Distributed Quantum Error-Correcting Codes for Large-Scale, Fault-Tolerant Quantum Computing: Academic and Industrial Advances and Perspectives

Shahram Babaie, Chunming Qiao, *Fellow, IEEE*

***Abstract*—Distributed quantum computing (DQC), augmented with distributed quantum error-correcting codes (DQECCs), is emerging as a promising architectural paradigm for large-scale fault-tolerant quantum computing (FTQC). However, realization of large-scale FTQC is a full-stack challenge that requires systematically addressing both theoretical and engineering constraints across architectural design, scalable fault-tolerant frameworks, efficient classical–quantum co-processing, and control integration. In this paper, we provide a comprehensive, system-level overview of academic advances and industrial progress toward distributed and large-scale FTQC. We first clarify the theoretical foundations and practical realization of FTQC, emphasizing the opportunities and challenges introduced by DQC and heterogeneous DQC. We then explore recent developments in error suppression, quantum error correction, error mitigation, fault-tolerant universal gate sets, and modular architectures, along with industrial progress on each topic and across different qubit modalities. We further discuss benchmarking methodologies, simulation frameworks, decoding tools, and performance metrics for assessing system-level readiness for large-scale FTQC. We also explore emerging industrial roadmaps toward FTQC, quantum–HPC orchestration, real-time decoding, and modality-specific architectures. Finally, we outline open challenges and future research directions by connecting academic research trajectories with industrial roadmaps and commercialization efforts, highlighting the pivotal role of distributed fault-tolerant designs in shaping next-generation utility-scale quantum computing systems.**

***Index Terms*—Fault-tolerant quantum computing, quantum error correction, quantum error mitigation, quantum error suppression, QEC decoding, fault-tolerant gate operations.**

I. Introduction

QUANTUM computing leverages unique features of quantum mechanics, such as superposition, interference, and entanglement, to address computational problems that are intractable for state-of-the-art (SOTA) classical counterparts. This paradigm is poised to revolutionize materials science, national security, optimization, and drug discovery, provided that the intrinsic fragility of quantum states, scalability constraints, and integration challenges are effectively addressed. Quantum circuits are susceptible to both coherent errors, such as faulty gates, and incoherent errors arising from environmental interactions that induce decoherence, relaxation, and dephasing, as well as imperfect state preparation and measurement (SPAM) [1].

Shahram Babaie and Chunming Qiao are with the Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY, USA (e-mail: shahramb@buffalo.edu; qiao@computer.org).

Corresponding author: Shahram Babaie.

Despite significant advances across various qubit modalities, perfect, i.e., unit-fidelity, operations remain unattainable in the near term, and even small physical inaccuracies accumulate over circuit execution and substantially degrade computational fidelity [2]. For instance, the success probability of a *depth-D* circuit scales approximately as p^D, where $0 < p < 1$ denotes the success probability of each operation, implying exponential degradation with increasing circuit depth[1] [3]. Fault-tolerant measures play an indispensable role in bridging the gap between relatively high physical error rates, typically on the order of $p_q \approx 10^{-3}$, and much lower logical error rates, e.g., $p_L \approx 10^{-15}$, required for application-scale quantum algorithms. However, realizing FTQC requires coordinated progress across the full quantum computing stack, including hardware design, noise engineering, error suppression, quantum error correction (QEC), error mitigation, compilation, decoding, and hybrid classical–quantum orchestration. This progress must be supported by both theoretical advances in academia and practical implementation efforts from industry [4].

Fault-tolerant quantum computing can be viewed as a cross-layer challenge spanning both hardware/physical-layer and control/software-layer concerns. At the physical layer, advances are required to address system-level constraints in fabrication and architecture design, aiming to enhance qubit coherence, gate and measurement fidelity, fabrication yield, connectivity, and, in distributed settings, the generation and routing of high-fidelity EPR (Einstein–Podolsky–Rosen) pairs [17]. At the control layer, fault tolerance relies on pulse-level optimization and execution-time measures, including error containment, noise-aware compilation, adaptive feedforward, and efficient classical post-processing, particularly real-time decoding. These mechanisms are complemented by error engineering, error suppression, error mitigation, and quantum error correction before they result in logical-level errors [18]. Error engineering tailors complex or highly correlated noise processes into structured and more tractable forms, thereby reducing implementation overhead. Error suppression identifies processes that lead to certain types of errors and avoids them through improved circuit design, calibration, and noise-aware operation. Error mitigation analyzes how disruptive factors affect computational outcomes and extrapolates more accurate estimates [19]. Ultimately, error correction techniques

[1]Circuit depth refers to the number of sequential layers of quantum operations in a quantum circuit.

TABLE I: A summary of recent surveys on large-scale fault-tolerant quantum computing

Paper	Year	Publication	ES	Error Correction					Error Mitigation				HP	FT-G	LS-QC		II
				ALG	TOP	qLDPC	SUB	DYN	ZNE	PEC	TEM	PNA			HeDQC	HoDQC	
[5]	2025	IEEE TGCN	×	✓	✓	×	×	×	×	×	×	×	×	×	×	×	✓
[6]	2024	IEEE TCE	×	✓	✓	×	×	×	×	×	×	×	×	×	×	×	×
[7]	2025	IEEE Access	×	✓	✓	✓	✓	×	×	×	×	×	×	×	×	×	×
[8]	2024	IEEE ICAIT	×	✓	✓	×	×	×	×	×	×	×	✓	×	×	×	×
[9]	2026	IEEE WiOpt	×	✓	✓	✓	×	×	×	×	×	×	×	×	✓	✓	×
[10]	2023	IEEE QCE	×	✓	✓	×	✓	×	×	×	×	×	×	✓	×	×	×
[11]	2026	IEEE ISSCC	×	×	×	✓	×	×	×	×	×	×	✓	✓	✓	✓	✓
[12]	2024	ACM CS	✓	✓	✓	×	✓	×	×	×	×	×	✓	×	✓	×	×
[13]	2025	Elsevier ICT	×	✓	✓	×	×	×	×	×	×	×	✓	×	✓	✓	×
[14]	2025	Preprint	×	×	✓	✓	×	×	×	×	×	×	✓	✓	×	×	×
[15]	2025	Preprint	×	✓	✓	✓	✓	✓	✓	✓	×	×	✓	×	×	×	×
[16]	2026	QUICK	×	✓	✓	×	✓	×	×	×	×	×	×	×	×	×	×
This Survey	2026	–	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

ES: Error suppression; **ALG:** Algebraic codes; **TOP:** Topological codes; **SUB:** Subsystem codes; **DYN:** Dynamical codes; **ZNE:** Zero-noise extrapolation; **PEC:** Probabilistic error cancellation; **TEM:** Tensor error mitigation; **PNA:** Probabilistic noise amplification; **HP:** Hardware platforms; **FT-G:** Fault-tolerant gates; **LS-QC:** Large-scale quantum computing; **HeDQC:** Heterogeneous distributed quantum computing; **HoDQC:** Homogeneous distributed quantum computing; **II:** Industrial Insights.

encode quantum information across multiple physical qubits and utilize error-correcting codes to form robust, fault-tolerant logical qubits [20]. Hereafter, we refer to these complementary approaches collectively as *quantum error containment mechanisms* (QECMs), which include quantum error engineering, error suppression, error mitigation, and error correction.

Building sufficiently reliable monolithic quantum processors for utility-scale computation remains a formidable near-term challenge. Although fault-tolerant quantum computing does not fundamentally require distributed architectures, there is growing consensus across academia and industry that DQC, in which multiple small- to moderate-scale quantum processing units (QPUs) are interconnected through quantum data network (QDN) units [21], [22], offers the most viable path for overcoming the scalability limitations of *monolithic quantum computing* (MQC) [23]. Such architectures can improve manufacturing yield, localize correlated noises, enhance architectural flexibility, and support the parallelization of resource-intensive components, such as magic state factories [24]. Modular and distributed architectures not only mitigate intrinsic implementation challenges associated with MQC but also fundamentally reshape the design and realization of fault-tolerant mechanisms, including logical operations, stabilizer measurement scheduling, and inter-QPU entanglement distribution [25]. This architectural shift is increasingly reflected in both academic research directions and industrial roadmaps, as exemplified by emerging modular systems such as IBM Quantum System II and Rigetti's Cepheus-1-108Q QPU, which are discussed in subsequent sections of this paper, and point toward a scalable route toward large-scale FTQC [26].

A. Paper Motivation

Recent progress in quantum error-correcting codes, fault-tolerant designs, and modular quantum architectures has generated a growing body of survey literature on relevant topics. To position this work within the existing literature, we conduct an in-depth analysis of recent survey studies to identify existing gaps and clarify the research direction addressed in this paper. Table I summarizes representative recent surveys and compares their coverage across key dimensions relevant to large-scale FTQC. Although these studies provide valuable insights into specific aspects of the field, most of them focus on particular quantum code families or limited fault-tolerant mechanisms, without sufficiently addressing practical system-level FTQC concerns. As a result, the literature still lacks a unified and holistic survey that jointly examines error suppression, QEC, error mitigation, decoding, fault-tolerant gate operations, scalable architectures, and industrial implementation progress.

Several important dimensions remain insufficiently explored in existing surveys. First, prior works discussed QECCs without fully connecting them to the broader fault-tolerant stack, including syndrome extraction scheduling [27], decoding requirements, logical gate implementation, magic state distillation, and hybrid classical–quantum orchestration. Second, the opportunities and challenges introduced by DQC for achieving large-scale FTQC have not been systematically reviewed, particularly from the perspective of heterogeneous and homogeneous distributed architectures [28]. Third, while hardware progress is central to the practical realization of FTQC, existing surveys provide limited coverage of industrial advances across qubit modalities, modular processors, and industrial roadmaps. This omission is significant because each qubit modality exhibits distinct advantages and constraints in terms of coherence, connectivity, gate fidelity, measurement capability, scalability, fabrication yield, and compatibility with the specific requirements of scalable FTQC. Finally, simulation frameworks, benchmarking methodologies, and evaluation metrics for assessing fault-tolerance performance, decoding algorithms, and system-level readiness remain scattered across the literature rather than being treated as core components of an integrated FTQC survey.

Motivated by these gaps, this paper provides a compre-

hensive and integrated perspective on distributed, large-scale FTQC by connecting academic foundations with industrial advances and practical implementation constraints. We systematically review quantum error containment mechanisms, architectural designs, simulation and benchmarking tools, qubit modalities, and modular quantum architectures. By bridging these fragmented research directions, this work aims to provide a unified reference for understanding the current status, near-term challenges, and long-term pathways toward utility-scale fault-tolerant quantum computing.

B. Paper Contributions

In this paper, we explore the academic research landscape and industrial progress addressing the long-standing challenges of quantum computing associated with the error-prone nature of quantum systems and the scalability constraints of current QPUs. We investigate the theoretical foundations and experimental demonstrations shape the transition from *noisy intermediate-scale quantum* (NISQ) processors toward *intermediate-scale quantum (ISQ)*, *utility-scale quantum* (USQ), and ultimately FTQC through four stages, (i) pre-execution fault avoidance techniques, (ii) fault-tolerant universal gate set implementations, (iii) execution-time QECMs, and (iv) post-processing measures. As illustrated in Figure 1, the scope of this study spans three tightly coupled domains, i.e., the physical layer, the control layer, and fault-tolerant design. These domains jointly capture the requirements for utility-scale FTQC and highlight the pivotal role of QECMs in realizing utility-scale FTQC. The main contributions of this study are summarized as follows.

- Exploring the role and recent progress of quantum error containment mechanisms, including error suppression, error correction, and error mitigation, in enabling utility-scale FTQC, as well as the motivations for their DQC-friendly implementations under distributed architectural constraints.
- Explaining code construction, syndrome extraction, and distinctive features of major QECC families, including topological, Quantum low-density parity-check (qLDPC), subsystem, and dynamical codes, along with their latest experimental results across different quantum architectures and distributed implementations.
- Detailing recent industrial advancements and modality-specific features relevant to FTQC and QEC requirements, as well as their role in realizing modular and large-scale platforms.
- Elaborating fault-tolerant universal gate set implementations at the physical and logical levels, including code deformation, lattice surgery, code switching, gauge fixing, magic state injection, distillation, and cultivation, together with their architectural requirements.
- Highlighting distinctive characteristics of *heterogeneous DQC* (HeDQC) compared with *homogeneous DQC* (HoDQC) in enabling FTQC systems, as well necessity of classical–quantum orchestration in this domain.
- Providing a taxonomy of decoding algorithms and exploring their compatibility with classical acceleration platforms, including CPUs, GPUs, field-programmable gate arrays (FPGAs), and application-specific integrated circuits (ASICs).
- Reviewing simulation tools, frameworks, libraries, and benchmarking methodologies leveraged for the technical assessment of fault-tolerant designs, decoding algorithms, QECCs, and distributed FTQC architectures.
- Demonstrating the current status, near- and long-term industrial roadmaps, and perspectives of leading quantum companies, together with delineating cross-layer open challenges and research directions shaping the trajectory toward large-scale FTQC,

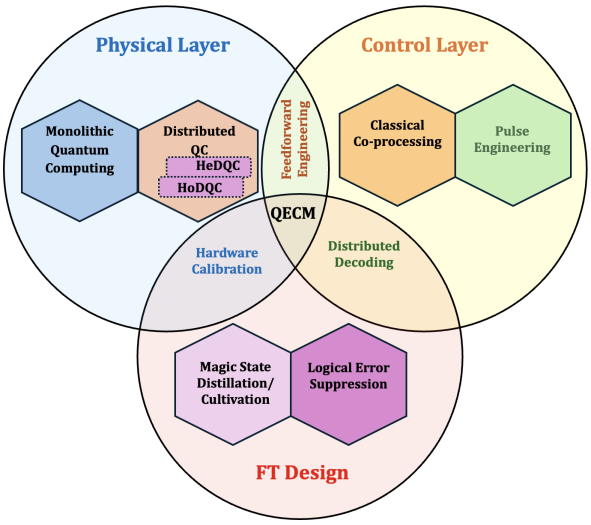


Fig. 1: Conceptual framework of this survey and its core domains for large-scale FTQC.

C. Paper Outline

The remainder of this paper is organized as follows. Section II introduces fundamental concepts related to error models, stabilizer formalism, and CSS codes. Section III discusses fault-tolerance criteria, including theoretical foundations, fault-avoidance techniques, execution-time measures, post-processing strategies, and the implementation of a fault-tolerant universal gate set. Section IV details hardware requirements for large-scale FTQC and heterogeneous DQC. Section V reviews theoretical advances and experimental demonstrations of major QECC families, including topological, qLDPC, subsystem, and dynamical codes, which is followed by decoding algorithms in Section VI. Section VII presents benchmarking methodologies and simulation tools, while Section VIII presents near- and long-term roadmaps of leading quantum companies. Section IX outlines open challenges and future research directions, and Section X concludes the paper.

II. Quantum Channels and Preliminaries

This section establishes the theoretical foundation for the subsequent discussion of QEC and FTQC. We first introduce representative quantum noise models and quantum channels to characterize how physical imperfections, decoherence, and

environmental interactions disturb quantum states and operations. We then present the stabilizer formalism, which provides the algebraic framework for the formal definition of QECCs.

A. Quantum Noise Models

The dynamical evolution of an open quantum system is formally described by a quantum channel, i.e., a *completely positive and trace-preserving* (CPTP) map $\mathcal{E}$ that transforms a density operator ρ as $\rho \mapsto \mathcal{E}(\rho)$ [29]. Quantum channels provide a time-domain mathematical framework for modeling imperfect control operations, decoherence, measurement imperfections, and system–environment interactions. Effective fault-tolerant design begins with a precise specification of the underlying noise model, which characterizes how errors arise, propagate, and manifest as detection events throughout a quantum circuit. Three commonly used noise models are *code-capacity*, *phenomenological*, and *circuit-level noise* [30]. The code-capacity noise model assumes that errors occur only on data qubits, while syndrome extraction and recovery operations are ideal. This model is useful for characterizing the intrinsic error correction capability of a quantum code under idealized syndrome extraction, independently of measurement errors and circuit-level imperfections. When system–environment interactions are memoryless, the noise is referred to as *Markovian noise*, where the quantum evolution depends only on the current state [31]. In the code-capacity noise model, data qubit errors may be modeled using physically motivated channels that capture mechanisms such as energy relaxation, dephasing, and thermal excitation, or using abstract channels such as Pauli, depolarizing, amplitude-damping, and erasure. In many analyses, errors across multiple qubits are assumed to be *independent and identically distributed* (i.i.d.); however, more realistic models may instead consider *independent but non-identically distributed* (i.ni.d.) errors, where error rates vary across qubits, gates, or spatial locations [32]. The decoherence time of a qubit, T_2, often measured using a *Hahn-echo sequence*, which is characterized by the relaxation time, T_1, and pure dephasing time T_ϕ, satisfying $\frac{1}{T_2} = \frac{1}{2T_1} + \frac{1}{T_\phi}$. In the presence of slow phase noise or low-frequency fluctuations, the Ramsey coherence time, T_2^*, captures additional dephasing effects and is typically shorter than the Hahn-echo coherence time, i.e., $T_2^* < T_2$ [33].

The phenomenological noise model extends the code-capacity model by incorporating measurement errors while still ignoring circuit-level errors in syndrome extraction circuits. Measurement errors are typically modeled using *positive operator-valued measures* (POVMs). For instance, the measurement outcome corresponding to the state $|0\rangle$ can be represented by $F_0 = (1 - P_M)\,|0\rangle\langle 0| + P_M\,|1\rangle\langle 1|$, where P_M denotes the readout error probability [34]. Although stabilizer measurements are typically repeated over multiple rounds to suppress uncertainty and distinguish data qubit errors from measurement errors, measurement-free paradigms can address challenges associated with measurement errors [35]. The circuit-level noise model provides the most realistic representation by accounting for errors during state preparation, idle periods, gate operations, and measurements, as well as leakage and correlated faults. Preparation errors may leave a qubit in a statistical ensemble of pure states, $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$, rather than the desired pure state [1]. Gate imperfections may include stochastic Pauli errors, incoherent relaxation and dephasing, or coherent control errors. In particular, coherent errors such as systematic over- or under-rotation by a small angle ϵ can accumulate under repeated operations. As a simple illustrative example, N consecutive imperfect identity operations acting on $|0\rangle$ result in $|\psi\rangle_{\text{out}} = \cos(N\epsilon)|0\rangle + i\sin(N\epsilon)|1\rangle$, leading to measurement probabilities $P(0) = \cos^2(N\epsilon) \simeq 1 - (N\epsilon)^2$ and $P(1) = \sin^2(N\epsilon) \simeq (N\epsilon)^2$, which deviate from the expected pure state [36].

B. Quantum Channels

The effects of physical disruptive factors can be modeled and analyzed using several canonical quantum channels, including *amplitude damping*, *phase damping*, and *Pauli* channels [4]. The amplitude damping channel captures energy relaxation processes, such as the decay of an excited state due to spontaneous emission, whereas the phase damping channel models the loss of quantum coherence without energy relaxation. In contrast, a Pauli channel represents noise processes as probabilistic applications of Pauli operators. For an arbitrary single-qubit state $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, a transformation to $|\psi'\rangle = \alpha\,|1\rangle + \beta\,|0\rangle$ is equivalent to applying the Pauli X operator, i.e., $|0\rangle$ and $|1\rangle$ are interchanged, which is called a bit-flip or X error. Similarly, a transformation to $|\psi'\rangle = \alpha\,|0\rangle - \beta\,|1\rangle$, i.e., $|1\rangle$ acquires a relative phase while $|0\rangle$ remains unchanged, corresponds to applying the Pauli Z operator and known as a phase-flip or Z error. Moreover, a transformation to $|\psi'\rangle = i\alpha\,|1\rangle - i\beta\,|0\rangle$, i.e., $|0\rangle$ and $|1\rangle$ are transformed to $i\,|1\rangle$ and $-i\,|0\rangle$, respectively, is equivalent to applying the Pauli Y operator and represents a bit-phase-flip or Y error [37].

The occurrence of a Pauli error is probabilistic, as represented in Equation 1, where the input state is mapped to a statistical mixture of the original state and the Pauli-transformed state.

$$\mathcal{E}_\zeta(\rho) = (1 - p_\zeta)\rho + p_\zeta \zeta\rho\zeta, \qquad \zeta \in \{X, Y, Z\}. \tag{1}$$

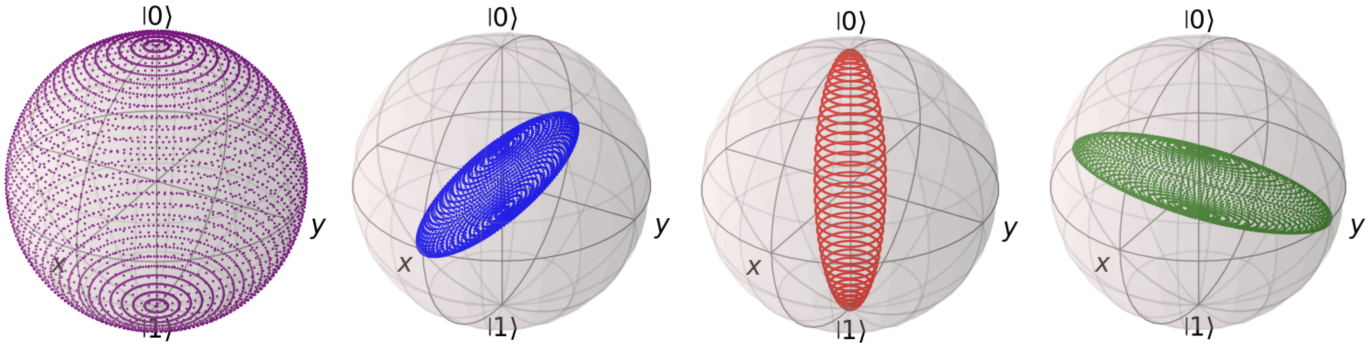
where p_ζ denotes the probability of the corresponding Pauli error. Under a Pauli-twirled approximation of relaxation and dephasing noise, the effective Pauli error probabilities can be expressed as Equation 2.

$$p_x = p_y = \frac{1}{4}(1 - e^{-\frac{t}{T_1}}), \quad p_z = \frac{1}{4}(1 + e^{-\frac{t}{T_1}} - 2e^{-\frac{t}{T_2}}) \tag{2}$$

A widely used instance of the Pauli channel is the depolarizing channel, in which the three non-identity Pauli errors are assumed to occur with equal probability. Let p denote the total probability of a non-identity Pauli error. Then, the single-qubit depolarizing channel can be represented as Equation 3.

$$\mathcal{E}_{\text{dep}}(\rho) = (1 - p)\rho + \frac{p}{3}\,(X\rho X + Y\rho Y + Z\rho Z) \tag{3}$$

It is worth noting that Pauli channels do not affect all quantum states uniformly. Indeed, each Pauli channel leaves



(a) A uniform distribution (b) Pauli X error (c) Pauli Z error (d) Pauli Y error

Fig. 2: Geometric representation of quantum states deformation under Pauli channels.

the eigenstates of its associated Pauli operator invariant, while contracting the Bloch vector components associated with the two remaining orthogonal axes. Figure 2(a) illustrates a uniform distribution of pure single-qubit quantum states over the surface of the Bloch sphere, representing the noiseless reference state ensemble before the application of Pauli error channels. Bit-flip error, which can be interpreted as a π rotation around the X-axis up to a global phase, since $X = R_{\vec{x}}(\pi) = e^{-i\pi\frac{X}{2}}$, preserves $|+\rangle$ and $|-\rangle$ states as the eigenstates of Pauli X operator, while contracting other states toward the X-axis, as illustrated in Figure 2(b). Similarly, phase-flip error corresponds to a π rotation around the Z-axis up to a global phase, since $Z = R_{\vec{z}}(\pi) = e^{-i\pi\frac{Z}{2}}$, preserves the eigenstates $|0\rangle$ and $|1\rangle$ while contracting other states toward the Z-axis, as shown in Figure 2(c). Likewise, as depicted in Figure 2(d), a bit-phase-flip error preserves the eigenstates of Y, $|+i\rangle = (|0\rangle + i\,|1\rangle)/\sqrt{2}$ and $|-i\rangle = (|0\rangle - i\,|1\rangle)/\sqrt{2}$, while contracting other states toward the Y-axis [38].

Beyond Pauli and decoherence-based errors, physical qubits are also subject to leakage and qubit loss, both of which can be modeled as erasure channels when the affected locations are detected or flagged [39]. Leakage occurs when a qubit leaves the intended two-dimensional computational subspace spanned by $\{|0\rangle, |1\rangle\}$ and occupies a non-computational energy level, such as $|2\rangle$ in a superconducting transmon qubits [40]. Leakage can occur in several qubit modalities [41], accumulate over multiple cycles, and propagate through subsequent multi-qubit interactions. Fault-tolerant architectures mitigate leakage through leakage detection and leakage removal techniques, including the quantum jump technique [42], leakage mobility [43], and coupling with ancilla qubits, commonly referred to as the *data qubit leakage removal* (DQLR) technique [44].

Qubit loss refers to the physical qubit loss from the system, such as photon absorption in photonic qubits or atom loss in neutral-atom systems [45]. Such processes can be modeled by an erasure channel, $\mathcal{N}_{\text{er}}(\rho) = (1-p_{er})\,\rho + p_{er}\,|e\rangle\langle e|$, where p_{er} denotes the erasure probability and $|e\rangle$ refers to an orthogonal erasure flag state outside the computational subspace, indicating that the corresponding qubit has been erased. Since erasure flags explicitly identify the locations of the affected qubits, erased qubits can typically be recovered more efficiently through re-initialization or replacement procedures. In contrast, *deletion errors* correspond to errors whose locations are unknown, making decoding and recovery more challenging. It is worth noting that, although most FTQC frameworks are formulated in terms of two-level qubits, higher-dimensional quantum systems, or *qudits*, are increasingly being explored for hardware-efficient simulation and computation. For instance, *qutrits* [46] and *ququints* are three- and five-level systems, respectively. Qudits are particularly well-suited for describing naturally high-dimensional gauge fields, allowing reduced register size and circuit complexity [47].

C. Stabilizer and CSS Codes

Stabilizer and CSS (Calderbank, Shor, Steane) codes form the algebraic foundation of many QECCs used in FTQC. They provide a compact operator-based framework for defining codespaces, detecting error through stabilizer measurements, and defining logical qubits and logical operations. The stabilizer formalism, introduced by Gottesman [48], provides an algebraic framework for describing a broad class of QECCs using operators rather than explicit state vector representations. In this formalism, the codespace is defined as the simultaneous $+1$ eigenstate of a set of mutually commuting Pauli operators, called stabilizer generators. The r independent stabilizer generators form an Abelian group $\mathcal{S} = \langle S_1, S_2, \ldots, S_r\rangle \subset \mathcal{P}_n \setminus \{\pm i, -1\}$, where $\mathcal{P}_n$ denotes the n-qubit Pauli group. A state $|\psi\rangle$ belongs to the codespace if and only if $S_i|\psi\rangle = |\psi\rangle$ for all $S_i \in \mathcal{S}$. Since each independent stabilizer constraint halves the dimension of the Hilbert space, the codespace has dimension $\dim(C) = 2^{n-r}$ and encodes $k = n - r$ logical qubits [49]. When an error $E \in \mathcal{P}_n$ occurs, stabilizer measurements produce a binary syndrome vector $\sigma(E) = (\sigma_1, \ldots, \sigma_r) \in \{0,1\}^r$, where $\sigma_i = 0$ if E commutes with S_i and $\sigma_i = 1$ if it anticommutes with S_i. Therefore, errors outside the normalizer of the stabilizer group are detected by at least one nonzero syndrome bit[2]. In contrast, errors in the normalizer commute with all stabilizer generators and therefore produce a trivial syndrome [50].

In stabilizer formalism, logical Pauli operators correspond to Pauli operators that commute with all stabilizer generators but are not themselves members of $\mathcal{S}$, i.e., members of $\mathcal{N}(\mathcal{S}) \setminus \mathcal{S}$, which act nontrivially on the codespace. Formally,

[2]The normalizer of the stabilizer group $\mathcal{S}$ in $\mathcal{P}_n$, denoted by $\mathcal{N}(\mathcal{S})$, consists of all Pauli operators that commute with every element of $\mathcal{S}$.

the logical Pauli group is represented by the quotient group $\mathcal{L}(\mathcal{S}) = \mathcal{N}(\mathcal{S})/\mathcal{S}$. The distance of a stabilizer code is the minimum weight[3] of a nontrivial logical Pauli operator, $d = \min_{P\in\mathcal{N}(\mathcal{S})\setminus\mathcal{S}} \mathrm{wt}(P)$. The detectable errors of a stabilizer code are $E(\mathcal{S}) = \mathcal{P}_n \setminus \mathcal{N}(\mathcal{S})$. A stabilizer code with parameters $[[n, k, d]]$ can detect all errors of weight up to $d-1$ and correct arbitrary errors of weight up to $\lfloor (d-1)/2 \rfloor$ [50]. In general, a stabilizer code encodes quantum information into a codespace C, which is a subspace of the full Hilbert space $\mathcal{H} = (\mathbb{C}^2)^{\otimes n}$. The Hilbert space can be decomposed into the codespace and its orthogonal complement as $\mathcal{H} = C \oplus C^{\perp}$, where $C^{\perp}$ contains orthogonal syndrome subspaces outside the codespace[4]. Stabilizer measurements project corrupted states onto syndrome subspaces associated with different error classes, a feature known as *discretization of error*[5]. This enables *active quantum error correction*, in which quantum information is protected through encoding, syndrome extraction, decoding, and recovery.

A notable subclass of stabilizer codes is the family of CSS codes, in which the stabilizer group contains only X-type and Z-type stabilizer generators. A CSS code can be defined by two binary parity-check matrices, H_X and H_Z, responsible for phase-flip and bit-flip error detection, respectively [51]. The commutativity of the associated stabilizer generators is guaranteed by the orthogonality constraint $H_X H_Z^T = 0 \pmod 2$. Equivalently, a CSS code, $\mathrm{CSS}(C_X, C_Z)$, can be constructed through two classical linear codes, i.e., $C_X = \ker(H_X)$ and $C_Z = \ker(H_Z)$, such that $C_Z^{\perp} \subseteq C_X$. The number of encoded qubits is $k = n - \mathrm{rank}(H_X) - \mathrm{rank}(H_Z)$. The code distance is $d = \min(d_X, d_Z)$, where $d_X = \min_{c\in C_Z\setminus C_X^{\perp}} \|c\|$ and $d_Z = \min_{c\in C_X\setminus C_Z^{\perp}} \|c\|$, corresponding to the minimum weight of vectors $(C_Z \setminus C_X^{\perp}) \cup (C_X \setminus C_Z^{\perp})$. Moreover, it has been shown that any $[[n, k, d]]$ stabilizer code can be mapped to a $[[4n, 2k, 2d]]$ CSS code [52], highlighting the central role of CSS constructions in quantum information theory.

The distributed implementation of CSS codes has been investigated by Babaie and Qiao [53]. They developed a distributed implementation of the Steane code, comprising six stabilizer generators, across seven QPUs through the use of a distributed *GHZ* (Greenberger–Horne–Zeilinger) state. The original quantum state is encoded as a superposition of even- and odd-weight codeword components associated with the underlying CSS construction. The authors demonstrated the effectiveness of their construction in detecting X, Z, and Y errors, which highlights its robustness and potential applicability in distributed quantum error correction architectures.

III. Fault-tolerant Criteria and Implementations

This section explores fault-tolerant measures across three stages, i.e., pre-execution, execution-time, and post-execution,

[3]The weight of a Pauli operator P, $\mathrm{wt}(P)$, is the number of qubits on which P acts nontrivially, i.e., as X, Y, or Z, rather than the identity.

[4]Shor's code encodes one logical qubit into a two-dimensional codespace within the nine-qubit Hilbert space, whose remaining dimensions form the orthogonal complement as $(\mathbb{C}^2)^{\otimes 9} = (\mathbb{C}^2)^{\otimes 1} \oplus (\mathbb{C}^2)^{\otimes 8}$.

[5]Discretization of error refers to the fact that arbitrary physical errors can be expanded in the Pauli basis. Hence, if a particular QECC detects both bit-flip and phase-flip errors, then it can also detect arbitrary quantum errors, including random Pauli, unitary, and non-unitary errors.

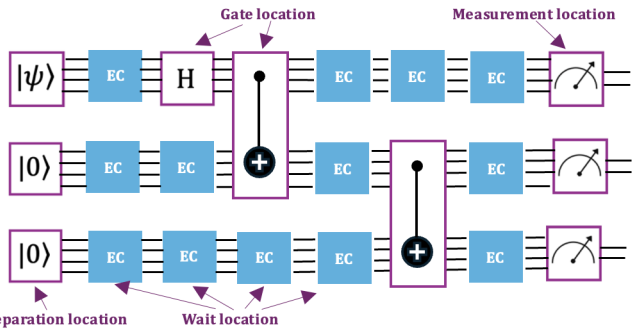


Fig. 3: A generic fault-tolerant model.

along with the mechanisms required to realize a fault-tolerant universal gate set. It also discusses the theoretical foundations and fundamental theorems that establish the feasibility of FTQC and define its principal limitations.

In general, fault-tolerant quantum computing refers to the ability to execute arbitrarily long quantum computations reliably despite the use of inherently error-prone quantum components operating in noisy environments. Realizing large-scale FTQC requires satisfying several key criteria that ensure reliable and scalable computation [54].

- *Fault-tolerant:* Physical errors must be prevented from transforming into uncorrectable logical errors, and logical errors must be sufficiently suppressed to support reliable quantum algorithm executions.
- *Addressable:* Encoded logical qubits must support independent preparation, manipulation, and measurement.
- *Universal:* The architecture must support a fault-tolerant universal gate set for general-purpose computation.
- *Adaptive:* The system must enable adaptive feedforward and conditional operations based on mid-circuit measurements (MCM) and syndrome stream.
- *Modular:* The architecture should support scalable integration of quantum modules or QPUs through QDN units.
- *Efficient:* The system must minimize physical qubit overhead, circuit depth, decoding latency, and classical–quantum orchestration complexity to maintain practical feasibility for near- and long-term quantum systems.

In the fault-tolerance literature, a quantum circuit is decomposed into several independent, indivisible error-prone operations, which are typically called *locations*. As depicted in Figure 3, there are five different types of locations, including state preparation, gate, measurement, wait/storage, and classical-processing locations, with the latter associated with classical–quantum orchestration. Each location may fail with a location-dependent error probability, although classical processing errors are often neglected or treated separately because their error rates are typically much lower than those of quantum operations. In idealized threshold analysis, errors can be modeled using a local stochastic noise process, where failures occur independently or with sufficiently weak correlations. However, in realistic applications, errors may be spatially or temporally correlated due to crosstalk, leakage, calibration drift, or shared control electronics [23]. Moreover, error propagation at the circuit-level can transform local er-

TABLE II: Error propagation through Clifford gates

Clifford Gate	Error	Propagated Error
CNOT	X_c	X_cX_t
	X_t	X_t
	Z_c	Z_c
	Z_t	Z_cZ_t
H	X	Z
	Z	X
S	X	Y
	Z	Z
X	X	X
	Z	$-Z$
Y	X	$-X$
	Z	$-Z$
Z	X	$-X$
	Z	Z
CZ	X_c	X_cZ_t
	X_t	Z_cX_t
	Z_c	Z_c
	Z_t	Z_t

rors into higher-weight correlated errors during computation. Such correlations complicate decoding, undermine assumptions used in simplified threshold estimates, and make error propagation a central challenge in FTQC.

During circuit execution, an error at one location may propagate through subsequent two-qubit gates and produce a higher-weight error within the same code block, potentially exceeding the code's correction capability. This risk is particularly significant in naively implemented high-weight stabilizer measurements, where a single error on an ancilla qubit can propagate to multiple data qubits and generate correlated *hook errors*. Fault-tolerant circuit design aims to ensure that any single error results in at most a correctable error within each code block [55].

Error propagation can be efficiently tracked through the Clifford group because Clifford gates map Pauli operators to Pauli operators under conjugation[6]. Therefore, if a Pauli error E occurs before a Clifford gate U, the corresponding propagated error after the gate is $UEU^\dagger$ [36]. Table II summarizes the transformation of single-qubit Pauli errors through common Clifford operations, which indicates how errors may either change type or propagate across multiple qubits. For instance, according to the table, a bit-flip error on the control qubit of a *CNOT* gate propagates to both control and target qubits, whereas a bit-flip error on the target qubit remains only on the target qubit. The propagation of multi-qubit Pauli errors can be derived by multiplying the propagated components of each Pauli factor[7]. Fault-tolerant implementations address this concern by preventing local physical errors from becoming logical errors.

In practice, FTQC relies on integration of complementary strategies, including (i) hardware- and control-level fault-avoidance and error suppression techniques, (ii) fault-tolerant operations and control mechanisms, (iii) execution-time measures, such as quantum error correction, and (iv) post-processing approaches, such as error mitigation and offline decoding. Since no single strategy can address all sources of errors, practical quantum systems must integrate these complementary techniques to reduce effective error rates and enable utility-scale FTQC systems [9].

A. Fault-avoidance Techniques

Over the past two decades, remarkable theoretical progress has established the foundations of FTQC; however, experimental demonstrations remain limited to relatively small-scale implementations. Therefore, quantum companies continuously strive to develop strategies that reduce the baseline physical error rates before active quantum error correction is applied. We collectively refer to these pre-execution and hardware-level measures as *fault-avoidance techniques* (FATs). Rather than correcting errors after they occur, FATs aim to prevent or suppress dominant error sources through advances in device architecture, materials engineering, environmental isolation, calibration, and quantum control [56]. Such techniques provide favorable architectural features that are more compatible with the implementation requirements of asymptotically good QECCs and fault-tolerant operations. Effective FATs depend on the understanding of the delicate nature of complex quantum systems, error sources, technological constraints, and fundamental theoretical bounds. Advances in materials purification and cryogenic engineering to suppress decoherence and unwanted excitations, precise calibration and control of microwave or laser pulses to minimize gate errors, improved wiring and shielding layouts to reduce electromagnetic interference (EMI) and crosstalk, and device geometry optimization to enhance connectivity and readout performance can be classified as fault-avoidance techniques [57].

Beyond architectural improvements during fabrication and design, the stochastic and time-varying nature of quantum hardware also necessitates continuous, real-time monitoring, adaptive control, and low-latency feedforward during system operation. This process requires fast and reliable acquisition of the measurement stream, including syndromes and calibration signals [58], followed by low-latency classical processing to infer system states, track parameter fluctuations, and trigger appropriate corrections. Machine learning and artificial intelligence techniques are increasingly being explored for adaptive control, decoding acceleration, noise characterization, and control optimization, thereby facilitating accurate and timely system stabilization under dynamic operating conditions [59].

In this regard, advances in qubit structure in terms of coherence time, gate fidelity, processing speed, and readout features are as critical as advances in QECC design for realizing utility-scale FTQC. Quantum processors can be realized through several computational paradigms and hardware technologies, including gate-based quantum computing [56], qudit-based quantum computing [38], measurement-based quantum computing [34], quantum annealing, and bosonic quantum computing [60]. Likewise, multiple qubit modalities are being pursued in parallel, including superconducting circuits, trapped-ions,

[6]A Clifford gate U maps the Pauli operator P to $UPU^\dagger \in \mathcal{P}_n$ for every $P \in \mathcal{P}_n$. A set of these gates constitutes a Clifford group, such as $\{H, S, \mathrm{CNOT}\}$.

[7]For instance, an input error $(X \otimes Y)$ on a *CNOT* gate propagates as $\mathrm{CNOT}(X\otimes Y)\mathrm{CNOT}^\dagger = i(X\otimes I)(Z\otimes Z) = Y\otimes Z$, up to a global phase. Moreover, propagation rules for Y errors can be inferred from $Y = iXZ$.

TABLE III: Architectural and operational characteristics of leading qubit modalities

Modality	Physical Encoding	Representative Companies	Coherence	1Q	2Q	Gate	Connectivity	Environment
Superconducting (Transmon)	JJ$^{\circ}$ oscillator	IBM, Google, Rigetti	$100-500\ \mu s$	$99.9\%+$	$99-99.7\%$	$10-200$ ns	NN† (2D)	$10-20$ mK
Trapped-ions	Hyperfine ion states	IonQ, Quantinuum	$sec.-min.$	$99.99\%+$	99.99%	$10-200\ \mu s$	A2A‡	UHV$^{\diamond}$ + lasers
Photonic	Pol.$^{\#}$/time-bin/dual-rail	PsiQuantum, Xanadu	Loss-dominated	$\sim 99.9\%$	Probabilistic	$\sim$ ns (optical)	Networked	Room temp.
Neutral-atoms (Rydberg)	Optically trapped atoms	QuEra, Pasqal, Infleqtion	$1-10$ s	$99.9\%+$	$99-99.96\%$	$0.1-1\ \mu s$	Reconfigurable 2D	Vacuum
Spin qubits (Quantum Dots)	Electron spin in silicon	Intel, Microsoft, Diraq	$10-100$ ms $(T_2^*)^{\S}$	$99.9\%+$	$98-99.5\%$	$10-100$ ns	NN† (2D/linear)	$10-100$ mK

$^{\circ}$**JJ**: Josephson junction; †**NN**: nearest-neighbor; ‡ **A2A**: all-to-all; $^{\#}$ **polarization**; § T_2^*: Ramsey coherence time; $^{\diamond}$ **UHV**:Ultra-high vacuum.

neutral-atoms, photonic platforms, and spin qubits [61]. Each modality exhibits distinct trade-offs in terms of scalability, noise characteristics, gate fidelity, readout features, entanglement distribution, and architectural constraints, and it remains unclear which platform, or combination of platforms, will ultimately provide the most viable path toward large-scale FTQC.

Table III compares the architectural characteristics of leading qubit modalities. Superconducting circuits offer fast gate operations and a high degree of integration [62]. Trapped-ion platforms provide a long coherence time and all-to-all connectivity, which can simplify the implementation of quantum operations [46]. Photonic systems inherently support long-distance entanglement distribution and room-temperature operation, making them promising for quantum networking and distributed architectures [63]. Neutral-atom architectures incorporate a long coherence time with reconfigurable geometries and programmable interaction topologies. Neutral-atom platforms, often referred to as *cold-atom* systems, typically operate in ultra-high-vacuum environments and do not require cryogenic cooling infrastructure [56]. The reconfigurable structure can be exploited to reduce space-time overhead[8] by enabling flexible qubit placement, parallel operations, and efficient realization of logical gates. For instance, the implementation of Shor's algorithm on superconducting circuits requires nearly one million qubits [64], whereas it can be implemented by approximately 10,000 atoms in neutral-atom architectures [65]. Semiconductor spin qubits benefit from nanoscale footprints and compatibility with CMOS (complementary metal–oxide–semiconductor) fabrication, offering a promising pathway toward dense integration and large-scale QPUs. These modality-specific features directly influence QECC design, syndrome extraction circuits, decoding strategies, and distributed FTQC architectures. Furthermore, analysis of single-qubit (1Q) and two-qubit (2Q) gate fidelities across different modalities indicates that many current platforms operate in the *KiloQuOp* regime, meaning they can reliably execute on the order of one thousand quantum operations (QuOps) before errors dominate. However, practical quantum advantage and large-scale FTQC will require progress toward the *MegaQuOp* and eventually *TeraQuOp* regimes, where application-driven quantum computation requires millions to trillions of reliable quantum operations [61].

B. Fault-tolerant Memories and Operations

Recent experimental demonstrations and industrial roadmaps indicate that realizing utility-scale FTQC with available quantum hardware remains beyond the near-term horizon. Fault-tolerant implementations aim to overcome this limitation by (i) encoding quantum information into blocks of physical qubits using QECCs, which are typically not decoded throughout computation, and (ii) replacing fragile locations with fault-tolerant modules, commonly referred to as *gadgets* [66]. These gadgets are small circuits composed of multiple physical operations and ancilla resources, designed to develop logical-level preparation, operations, measurements, and error correction while preventing a small number of physical errors from becoming uncorrectable logical errors. In Figure 3, the purple boxes indicate such fault-tolerant gadgets acting on encoded blocks of physical qubits. In general, the transition from physical- to logical-level computation introduces several fundamental challenges, including efficient qubit allocation, QEC overhead, logical gate implementation, and the design of fault-tolerant primitives (i.e., gadgets) [49].

1) Fault-tolerant Qubits: Physical qubits are typically microscopic quantum systems in which quantum information is encoded in delicate properties such as electron spin, atomic states, or photon polarization, making them highly fragile and susceptible to interaction with environmental noise and stray particles [67]. While hardware progress aims to enhance the performance of physical qubits (e.g., to achieve a physical qubit error rate below the QEC threshold), practical fault tolerance ultimately requires integrating these advances with effective quantum error correction. A near-term solution to protect quantum information is to encode logical quantum states across multiple entangled physical qubits to form robust, fault-tolerant logical qubits (FTLQs) whose effective error rate can be suppressed below the physical error rate when the noise is below threshold. The ratio of the number of physical qubits to logical qubits defines the *QEC overhead*, also referred to as *space overhead*. QEC overhead depends on several factors, including physical qubit quality, utilized code family, and decoding algorithm [68]. Moreover, QEC overhead is a fundamental metric for assessing the efficiency and scalability of QECCs, in which low QEC overhead is desirable, as it improves scalability, reduces hardware requirements and cost, and enhances resource efficiency.

2) Fault-tolerant Universal Gate Set: Quantum operations are fundamental to FTQC because logical circuits must support high-fidelity manipulation of encoded states. Unlike a naive approach that would decode a logical qubit, apply physical operations, and then re-encode it, fault-tolerant gadgets preserve encoded state throughout the computation. Moreover, their key objective is to ensure that a small number of physical errors during a logical operation produce only correctable errors within each code block. The simplest way to develop fault-tolerant gadgets is transversal implementations.

[8]The space-time overhead, quantified in *qubit.rounds*, captures the product of the number of physical qubits and the number of gates required by an algorithm.

In transversal single-qubit logical gates, independent physical gates are applied across the qubits of a code block, while in transversal multi-qubit gates, physical gates are applied qubit-wise between corresponding qubits on different code blocks. Since no physical gate couples two qubits within the same code block, a single physical error cannot spread to multiple qubits inside that block. Therefore, transversal gates propagate errors between corresponding qubits of different blocks, but they remain correctable by subsequent QEC cycles [69].

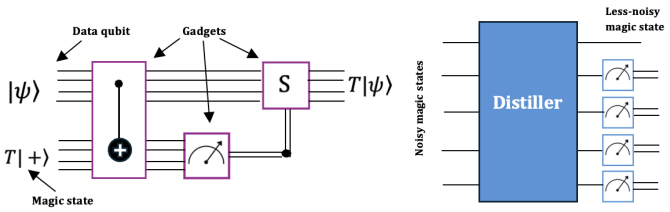
Although certain code families admit transversal implementations of specific non-Clifford gates, the *Eastin–Knill theorem* [70] establishes that no finite-dimensional quantum error-correcting code can support a universal logical gate set through transversal implementations. Therefore, FTQC requires additional techniques for implementing non-Clifford operations, such as code deformation (e.g., lattice surgery [71]), code switching [72], gauge fixing [73], and magic state injection [74]. This requirement is particularly important because Clifford circuits alone can be efficiently simulated classically, whereas augmenting the Clifford gate set with at least one non-Clifford gate, commonly the T gate, enables universal quantum computation.

The T-gate is essential for a universal gate set, but it dominates the space-time resource cost. For instance, Fowler et al. estimated that around 94% of the computation resources in surface codes may be devoted to T-state factories [75]. Although subsequent optimizations [76], [77] have reduced this fraction to approximately 30%–70%, these state factories can still constitute a substantial portion of the overall resource cost. Therefore, optimizing T-state preparation and distillation can substantially improve system-level performance, while the achievable speedup is bounded by Amdahl's law [78], which is given by Equation 4.

$$S_{\text{overall}} = \frac{1}{(1 - f_T) + \frac{f_T}{\gamma_T}} \tag{4}$$

where f_T denotes the fraction of the total cost dominated by T-gate resources, γ_T is the speedup achieved for T-gate, and $1 - f_T$ represents the fraction unaffected by the optimization[9], such as Clifford operations, measurement, and decoding.

A fault-tolerant T-gate can be implemented indirectly by consuming a T-type magic state, $|A\rangle = T|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$ along with Clifford gates, including *CNOT* and *S*, standard basis measurements, and classical feedforward. In this approach, known as *magic state injection* or gate teleportation, a *CNOT* is applied to the data qubit (i.e., the input state to which the T-gate is ultimately applied) as the control and the auxiliary magic state as the target [79]. The *CNOT* gate is followed by a measurement on the magic state, whose outcome determines the conditional correction on the data qubit. In particular, the desired T-gate is realized by either applying an S-gate[10] on the data qubit if the measurement outcome is 1 or applying no correction if the measurement outcome is 0 [80].



(a) Logical magic state injection (b) Distiller

Fig. 4: Magic state injection.

In logical-level circuits, an injected physical magic state $|T\rangle$ must be encoded into a logical magic state $|\bar{T}\rangle$, which can suppress the error rate to approximately $P_L \sim O(p/2)$ [81]. As illustrated in Figure 4(a), this transformation replaces physical operations with fault-tolerant gadgets, highlighting that magic state injection works at the *logical* level [82]. However, this level of suppression is insufficient for large-scale FTQC that requires a large number of logical T gates[11]. Therefore, additional magic state purification mechanisms are required to prepare high-fidelity logical T-type magic states [83].

A standard approach is *magic state distillation* (MSD), a probabilistic process that consumes multiple noisy magic states and generates fewer higher-fidelity magic states [84]. As depicted in Figure 4(b), a distillation circuit, or *distiller*, takes a collection of noisy encoded magic states and measures all outputs except one of them. If the measured syndrome is consistent with successful distillation, the top output is accepted as a higher-fidelity logical magic state, which may be consumed in a magic state injection circuit or used as an input to subsequent distillation rounds for further error suppression. Otherwise, the output is rejected, and the process must be repeated. Although several distillation protocols exist, they share the common principle of using stabilizer code structure and Clifford operations to probabilistically suppress non-Clifford magic state errors. Magic state injection has been experimentally validated in terms of overhead and practicality across several platforms, including superconducting qubits [79], trapped-ion qubits [85], and quantum dots [86].

Although magic state distillation enables fault-tolerant realization of non-Clifford gates, it introduces substantial space-time overhead. For example, in the standard *15-to-1* distillation protocol [69], fifteen noisy logical magic states $|\bar{T}\rangle$ are consumed to produce one higher-fidelity $|\bar{T}\rangle$ state, with output error scaling approximately as $P_{\text{out}} \sim O(35p^3)$. For instance, a physical magic state error rate of $p \approx 10^{-3}$ gives $P_L = 35(10^{-3}/2)^3 \approx 4.4 \times 10^{-9}$, where the factor of 2 in the denominator accounts for the mapping from the physical error rate to the logical error rate, as described above. Achieving even lower logical error rates, e.g., $P_L \approx 6 \times 10^{-10}$, requires a space-time footprint on the order of $\approx 3 \times 10^6$ qubit.rounds, highlighting the high overhead of MSD [87]. Several optimization techniques have been introduced to reduce this overhead, including postselecting the injection [88], lattice

[9]For example, if 50% of the computational cost is attributable to T-state factories, then even an arbitrarily large speedup on this component yields at most a $2\times$ overall speedup.

[10]The correction can be either an $S := \text{diag}(1, i)$ or $S^\dagger := \text{diag}(1, -i)$ operation depending on the injection circuit and measurement convention.

[11]For example, if $p \approx 10^{-3}$, the injected magic state may still have an error on the order of 5×10^{-4}, which is far too large for quantum algorithms requiring 10^9–10^{15} logical T-gates [83].

TABLE IV: Comparative overview of magic state cultivation and distillation

Metric	Magic state cultivation	Magic state distillation
Core idea	One-to-one	Many-to-one/ Many-to-fewer
Operation level	Physical	Logical
Input magic states	One	Multiple
Error reduction	Bounded fidelity	Arbitrarily low error rates (iteration)
Limitation	Postselection cost wall and limited repeatability	High space-time overhead
Trade-off	Cheaper ($\sim 10\times$)	Stronger error suppression ($\sim 10^{-12}$)
Code position	Inside a code block	Concatenated on top of other codes

surgery instead of braiding [89], reducing the code distance where the distillation code protects the computation code [90], and exploiting the complementary gap to get a better trade-off between attempts and residual logical errors [91]. Nevertheless, magic state distillation remains one of the dominant resource bottlenecks in large-scale FTQC.

This bottleneck motivates alternative approaches, such as *magic state cultivation* (MSC), which reduces reliance on large distillation factories and reduces space-time overhead[12] by gradually improving a single injected magic state at the *physical* level [92]. In contrast to MSD, which follows a many-to-one or many-to-fewer resource conversion, MSC follows a one-to-one refinement process in which a single noisy magic state is progressively improved through local stabilizer checks, code growth, and postselection procedures [93]. At a high level, MSC consists of three stages, i.e., *injection*, *cultivation*, and *escape*. In the injection stage, an initial low-distance encoded T-state is prepared using a QECC, such as distance-3 color codes. During the cultivation stage, the encoded state gradually grows in code distance, for example, from $d = 3$ to $d = 5$. In the escape stage, the code distance grows rapidly to the target code distance, such as $d = 15$, and then waits for the decoder to evaluate confidence information, including the complementary gap, to determine whether the cultivated state should be accepted or discarded and the protocol restarted. Although MSC can substantially reduce space-time overhead, it does not generally provide an arbitrarily low error rate through an iterative process [84]. Its reliance on postselection limits the repeatability and introduces a *postselection cost wall*, in which as the target state fidelity increases, the acceptance probability decreases, leading to more frequent rejection (i.e., by postselection) and restarts. Table IV summarizes the key differences between MSD and MSC across parameters that are critical for FTQC. Overall, cultivation provides a cheaper and potentially order-of-magnitude lower overhead pathway when moderate magic state fidelity is sufficient, which makes it a promising candidate for implementing non-Clifford gates in near-term FTQC systems.

Beyond magic state-based techniques, code switching, code deformation, and gauge fixing constitute complementary fault-tolerant mechanisms for realizing logical gate operations. Code switching refers to a fault-tolerant paradigm in which a logical quantum state is transferred between two distinct QECCs to enable logical operations that are non-transversal or resource-intensive in a single QEC code. Although code switching exploits the complementary properties of two distinct QECCs, it incurs nontrivial overhead due to the conversion procedures, whether they are based on teleportation, stabilizer re-encoding, or intermediate code deformation, which must be implemented fault-tolerantly and must preserve the logical information throughout the process [94]. Gauge fixing leverages a similar strategy within a subsystem code family, as discussed in section V, where distinct logical encodings are obtained by measuring (i.e., fixing) different sets of gauge operators rather than transferring the logical state between two distinct codes. The measurement projects the system into alternative stabilizer subspaces associated with the same underlying subsystem code. This enables the realization of a universal fault-tolerant gate set whose elements are not simultaneously transversal within a single subsystem code. In this sense, gauge fixing can reduce the overhead associated with full code switching by avoiding complete re-encoding or state transfer, while still exploiting multiple effective code structures.

Code switching can be realized between closely related code families with different parameters [94] or structures, such as between two- and three-dimensional color codes [95] or between two- and three-dimensional hypergraph product (HGP) codes [96], or between distinct code families with complementary transversal gate sets [97], [98], such as between color codes and surface codes. Experimental studies have validated the practical implementation of code switching across trapped-ion platforms [99]. These demonstrations highlight the practical relevance of code transformations as a complementary route to universal FTQC.

Code deformation is another fault-tolerant technique in which logical operations are implemented by dynamically modifying the stabilizer structure of a QECC, particularly in topological codes. The logical information remains protected throughout the process as the code is continuously transformed between valid stabilizer configurations. Code deformation can be realized through several hardware-friendly operations, including boundary deformation, defect (e.g., hole) movement, braiding, and lattice surgery [100]. Lattice surgery, a prominent form of code deformation in the surface code, realizes logical operations through merge-and-split procedures and joint Pauli measurements between code patches [101], [102]. The overhead and practical feasibility of lattice surgery have been experimentally validated on superconducting qubits [103] and trapped-ion qubits [104]. Furthermore, hybrid approaches that combine multiple fault-tolerant techniques, such as integrating magic state injection with code switching [85], as well as qubit modality-specific approaches, such as neutral-atom-based methods [105], have been developed to enable the implementation of a universal fault-tolerant gate set.

C. Execution-time Measures

Although fault-avoidance techniques aim to provide more reliable components, residual device, environmental, and op-

[12]For example, cultivation can produce one magic state every 3×10^3 qubit.rounds at a logical error rate of approximately $P_L \approx 6 \times 10^{-10}$, representing a substantially lower space-time overhead than MSD [92].

erational errors remain unavoidable during computations and must be continuously contained at execution time, particularly at the logical level, to prevent their accumulation. Accordingly, as illustrated by the blue boxes in Figure 3, error correction procedures are applied whenever there is an opportunity to contain accumulated errors. These approaches encompass a broad class of measures, including quantum error correction, error suppression, and error mitigation. It is important to emphasize that these techniques typically not only introduce additional overhead in the form of gate operations, circuit depth, and ancilla qubits, but also introduce new potential error sources and implementation complexity. Consequently, the usefulness of each technique is characterized by a break-even point, referred to as the *threshold*, that delineates the trade-off between the additional errors introduced by the QEC procedure and its error suppression capability.

Quantum error correction protects quantum information by encoding a logical qubit across multiple physical qubits, thereby enabling error detection and correction through stabilizer measurements without directly measuring or collapsing the encoded logical state[13]. QEC codes exhibit substantial diversity in terms of their construction, code parameters, stabilizer or gauge structure, and underlying algebraic or topological design. They differ in hardware requirements, including qubit connectivity, measurement capabilities, gate locality, and complexity of decoding algorithms [106]. These differences directly impact key performance metrics, including logical error rate, QEC threshold, QEC overhead, and scalability. From a structural perspective, QEC codes can be categorized into several families, which are discussed in section V.

Error suppression refers to the class of techniques that reduces the likelihood and impact of errors during quantum computation rather than explicitly detecting and correcting them through QEC. These techniques typically serve as the first layer of execution-time protection because they incur relatively low overhead while reducing effective physical error rates, thereby facilitating subsequent QEC measures. Error suppression can be conducted through several approaches, including dynamical decoupling (DD) [107], decoherence-free subspaces (DFS) [108], noise-aware control [31], and Hamiltonian engineering [109]. Dynamical decoupling suppresses decoherence by applying carefully designed sequences of control pulses that effectively average out environmental noise and extend qubit coherence without introducing redundancy [110]. For instance, qubits at idle or wait locations[14] are particularly susceptible to decoherence and error accumulation. Such errors can be partially suppressed by inserting sequences of unitary and their inverse operations, whose ideal net action is $UU^\dagger = I$. By dividing the wait location between U and $U^\dagger$, the accumulation of certain noise components can be suppressed. Decoherence-free subspaces provide a *passive error suppression* mechanism by encoding quantum information into subspaces or subsystems of the Hilbert space that are invariant under specific error channels, which eliminates the need for active error correction or stabilizer measurement [33]. In contrast, noise-aware control techniques, such as pulse shaping, enhance gate fidelity by tailoring time-dependent control fields to minimize sensitivity to noise, suppress control imperfections, and limit leakage outside the computational subspace [111]. Finally, Hamiltonian engineering improves system robustness by modifying the effective Hamiltonian through designed control fields, couplings, or encoding strategies. This can reduce sensitivity to noise, suppress dominant errors, enhance coherence, and improve operation fidelity [112].

[13]From a thermodynamic perspective, QEC can be viewed as an active process that removes entropy from encoded quantum states, analogous to a refrigerator or heat pump, thereby maintaining the integrity of logical qubits during computation.

[14]For instance, the time interval between initialization and a subsequent *CNOT* gate on the third qubit in Figure 3.

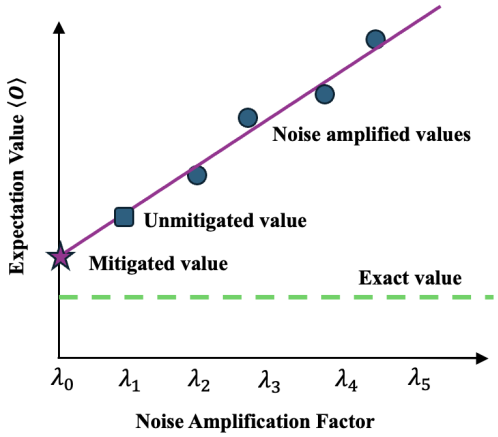


Fig. 5: Zero-noise extrapolation.

D. Post-processing Strategies

As the final layer in the pursuit of FTQC, post-processing strategies improve computational fidelity, particularly in near-term and pre-fault-tolerant regimes, where scalable QEC remains experimentally challenging, and the desired logical error rates required for fault-tolerant *application-scale quantum computing* are not yet attainable [64]. Post-processing methods compensate for noise-induced errors by classically processing measurement data collected *after each circuit execution*, rather than modifying the underlying quantum circuit or actively correcting errors during execution. Their effectiveness, however, depends on the accuracy of the noise model, the availability of calibration data, the structure of the observables, and the depth and size of the circuit. These methods typically increase sampling complexity, since they often transform systematic bias into statistical variance [113]. However, due to stochastic noise, these techniques require multiple executions (i.e., shots) and extensive sampling to average out additional uncertainties. A variety of post-processing approaches have been developed, including zero-noise extrapolation (ZNE) [114], probabilistic error cancellation (PEC) [115], tensor network error mitigation (TEM) [116], and propagated noise absorption (PNA) [117], all of which are broadly applicable across different qubit modalities.

Zero-noise extrapolation estimates the ideal, i.e., noise-free, expectation value of an observable by executing a given quantum circuit at multiple artificially amplified noise levels and extrapolating the measured results to the zero-noise limit.

Let $\langle O \rangle(\lambda)$ denote the expectation value of an observable under a noise amplification factor λ. As illustrated in Figure 5, the observable is first measured at the native noise level, typically when $\lambda = 1$, to obtain the unmitigated value, denoted by a square in Figure 5. The effective noise strength is then systematically increased, i.e., $\lambda > 1$, to generate a set of noisy measurements, represented by circles in Figure 5. By analyzing the dependence of $\langle O \rangle(\lambda)$ on λ, an extrapolation method, such as *linear*, *polynomial*, or *Richardson*, is applied to infer the ideal expectation value $\langle O \rangle_{\text{ideal}}$, in the zero-noise limit $\lambda \to 0$, indicated by a star in the Figure. Noise amplification can be implemented using several techniques, such as *gate folding*, where a gate U is replaced by $UU^\dagger U$ to increase accumulated error, and *pulse stretching*, where control pulses are lengthened to increase exposure to decoherence [114]. Despite its conceptual simplicity, ZNE introduces practical challenges, as it requires controlled noise amplification and repeated circuit execution. Nevertheless, when the extrapolation model is accurate, it can produce estimates substantially closer to the ideal expectation value [118].

Probabilistic error cancellation is a canonical error mitigation technique that reconstructs the ideal expectation value by expressing noisy operations as a linear combination of implementable operations and the inverse of the quantum channel obtained through quasi-probability sampling and classical post-processing. By combining measurement outcomes with appropriate weights, including negative coefficients, PEC yields an unbiased estimator of the ideal expectation value. Although each circuit execution is subject to noise, the aggregated result cancels noise statistically, thereby approximating the noise-free observable [115]. As illustrated in Figure 6, consider a width-n[15], depth-d circuit in which an ideal gate U_i is implemented as a noisy operation $\tilde{U}_i = \mathcal{E}_i \circ U_i$, where $\mathcal{E}_i$ denotes the associated quantum channel. Since physical quantum channels are generally not directly invertible on hardware, PEC approximates the inverse quantum channel through a linear decomposition of implementable noisy operations, $\mathcal{E}_i^{-1} \approx \sum_i q_i \mathcal{E}_i$, where q_i are quasi-probability coefficients and $\mathcal{E}_i$ are implementable operations. This introduces an *anti-noise* channel that counteracts the effect of $\mathcal{E}_i$. Accordingly, the ideal expectation value is reconstructed through classical post-processing as $\langle O \rangle_{\text{ideal}} \approx \sum_i q_i \langle O \rangle_i$. Although PEC can produce an unbiased estimator of the ideal expectation value, it incurs sampling overhead and depends on accurate noise characterization, which limits its scalability for utility-scale FTQC [119]. To address this limitation, Xie et al. [19] introduced space-time noise inversion, which avoids full noise characterization by leveraging a small set of accurately estimated error parameters. This approach reduces sampling overhead and provides robustness against parameter fluctuations, which offers a promising direction toward large-scale FTQC.

The sampling overhead of PEC can remain significantly large even under optimistic hardware assumptions. For example, even if hardware accelerations reduce the expected accumulated error count of a circuit to the order of 16, estimating

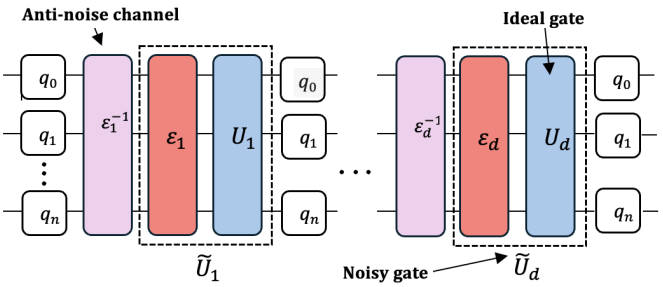


Fig. 6: Probabilistic error cancellation.

an observable to within about 5% accuracy may still require a sampling overhead of $\sim 10^6$. This corresponds to roughly 4×10^8 circuit executions, corresponding to approximately two to three days of quantum runtime when each execution takes about $0.5\,\text{ms}$[16] [116]. This motivates the development of methods that reduce the sampling burden of post-processing-based mitigation techniques. Moreover, the accuracy and sampling complexity of post-processing techniques are strongly influenced by the discrepancy between the measured outcomes and the corresponding ideal expectation values [120].

Recent advances have demonstrated that classical algorithms based on *Pauli propagation* (PP) can efficiently approximate expectation values of arbitrary observables associated with quantum circuits, which reduces mitigation overhead [55]. Building on this idea, IBM developed *Pauli-Prop*, a *Rust*-based Qiskit add-on[17] for approximating the evolution of operators in the Pauli basis under the action of quantum gates and quantum channels. Eddins et al. [121] leveraged Pauli propagation to introduce the concept of *shaded lightcone* (SLC), which refines PEC by reducing the sampling overhead. The key insight is that not all error channels contribute equally to the bias of a given observable. By identifying and selectively mitigating the subset of error channels within the relevant causal region that most strongly affects the target observable, SLC-based PEC improves the bias–variance trade-off. A Python-compatible implementation of this framework[18] facilitates its integration into practical quantum workflows.

Tensor network error mitigation is a hybrid classical–quantum post-processing technique that reduces the impact of noise with near-optimal sampling overhead. In this approach, noisy quantum circuits and states are represented as tensor networks [122], where quantum gates are mapped to tensors and quantum channels are incorporated as additional tensors. TEM approximates the inverse of the global quantum channel and applies it to measurement data obtained from informationally complete measurements. These data are then processed using classical tensor network contraction to estimate noise-free expectation values [123]. Because the mitigation is performed classically, TEM does not alter the underlying quantum circuit and can simultaneously estimate multiple observables from the same dataset. Furthermore, tensor network approximation and truncation schemes can be employed to control classical computational complexity by discarding weak correlations or

[15] Circuit width refers to the total number of qubits, including data and ancilla qubits, simultaneously involved in a quantum circuit.

[16] $4 \times 10^8 \times 0.5\,\text{ms} = 2 \times 10^5\,\text{s} \approx 2.3\,\text{days}$.

[17] https://github.com/Qiskit/pauli-prop

[18] https://github.com/Qiskit/qiskit-addon-slc

TABLE V: Comparative overview of quantum error mitigation techniques

Technique	Sampling Overhead†	Circuit Modification°	Classical Cost‡	Scalability*	Main Bottleneck
ZNE	Moderate (noise-level dependent)	Yes	Low	Limited (noise amplification)	Extrapolation error
PEC	High (depth-dependent)	Yes	Moderate	Poor (variance explosion)	Variance/sampling overhead
PNA	Problem-dependent ($<$ PEC)	No	Moderate–high	Structure-dependent	Operator growth
TEM	Near-optimal (information-theoretic)	No	High	Limited by tensor contraction	Classical complexity

† refers to the number of circuit executions required to achieve a target accuracy; ° indicates whether the original quantum circuit or its implementation must be modified; ‡ reflects the computational cost of classical post-processing; * captures the dominant limitation as system size, circuit depth, or noise strength increases.

suppressing negligible noise contributions. Filippov et al. have shown that TEM can achieve lower sampling overhead than PEC and ZNE under realistic noise models [116].

In a related direction, propagated noise absorption mitigates errors by incorporating the inverse effect of noise directly into the observable through operator propagation in the Pauli basis [117]. Rather than modifying the circuit or relying on quasi-probability sampling, PNA transforms the target observable into a noise-mitigated operator whose expectation value on the original noisy circuit approximates the ideal result. This approach shifts the mitigation burden to classical processing of operators, which can reduce stochastic circuit sampling and associated overhead. IBM has developed a relevant Python-compatible Qiskit add-on, *qiskit-addon-pna*[19]. However, PNA relies on accurate noise characterization and efficient operator propagation, and it may become computationally demanding for large-scale systems and non-Clifford-intensive circuits. Compared with SLC-based PEC, PNA offers a complementary trade-off between computational cost and mitigation accuracy [124]. Table V compares these mitigation techniques in terms of structure and scalability. ZNE is conceptually simple but requires controlled noise amplification and remains limited by extrapolation error. PEC can provide unbiased estimates but suffers from potentially severe sampling overhead. PNA avoids explicit circuit modification and can reduce sampling demands, but its effectiveness depends on efficient operator propagation. TEM can achieve favorable sampling behavior and reuse the measurement stream across observables, but its scalability is limited by tensor network contraction complexity.

Although quantum error mitigation is traditionally performed as a post-processing technique, Martin et al. [125] introduced a pre-processing framework in which the measurement operator itself is modified such that measurement of the noisy quantum state directly estimates the noiseless expectation value. Given a noisy state $\tilde{\rho}$ and a target observable O, the method constructs a surrogate observable $\tilde{O}$ satisfying $\mathrm{Tr}(\tilde{O}\tilde{\rho}) \approx \mathrm{Tr}(O\rho)$, where ρ denotes the ideal noiseless state. By exploiting prior knowledge of the underlying noise processes and tensor network representations [126], this framework can outperform conventional TEM in terms of average error, circuit depth, and computational complexity.

E. Theoretical Foundations and Theorems

Quantum error containment approaches and fault-tolerant implementations exploit various forms of redundancy, which also introduce new sources of error and degrade computational fidelity. The central question in FTQC is therefore whether the benefit of redundancy can outweigh the additional errors introduced by its implementation [127]. The *quantum fault–tolerance theorem* establishes that any quantum algorithm can be transformed into a fault-tolerant implementation if the physical error rate satisfies $p < p_{th}^{FT}$, where $p_{th}^{FT} > 0$ refers to a constant fault-tolerant threshold. In this case, an N-gate circuit can be simulated with arbitrarily small logical error using a fault–tolerant encoded circuit with $N\,\mathrm{polylog}(N)$ overhead[20] [128]. This theorem establishes the theoretical foundation for scalable quantum computing by guaranteeing that once physical qubits are sufficiently accurate (i.e., maintaining physical error rates below the threshold), scalable quantum computation can be realized by increasing the number of physical. More rigorous threshold analyses, such as the *extended-rectangle* (exRec) and *malignant-set* frameworks, further showed that the thresholds are not universal constants [129], [130].

A properly designed fault-tolerant implementation suppresses the logical error rate from $O(p)$ to $O(p^{t+1})$[21], where t denotes the number of correctable errors. Logical errors can be further suppressed through recursive encoding with concatenated quantum codes. For distance-3 codes, the logical error rate after r concatenation levels scales approximately as $P_L^{(r)} \approx (Cp)^{2^r}/C$, where C depends on the code and fault-tolerant gadgets. When $Cp < 1$, the logical error rate decreases rapidly with concatenation level, at the cost of increasing space-time overhead [131]. These results establish the theoretical scalability of FTQC while emphasizing that practical thresholds depend on the complete encoded architecture rather than on the QECC alone.

Beyond threshold conditions, no-go and locality theorems impose fundamental constraints on fault-tolerant logical operations. The *Bravyi–König theorem* extends the Eastin–Knill theorem, as discussed in section III, by showing that, for a topological stabilizer code in D spatial dimensions, any logical gate implementable by a constant-depth geometrically local circuit must lie within the D-th level of the Clifford hierarchy [132]. Therefore, two-dimensional topological stabilizer

[19] https://github.com/Qiskit/qiskit-addon-pna

[20] Typically, $p_{th}^{FT} \leq p_{th}^{QEC}$ because the fault-tolerant threshold depends not only on the underlying QEC code but also on the strategies leveraged for fault-tolerant implementations of state preparation, logical gates, syndrome extraction, and measurements. Moreover, the *pseudo-threshold*, denoted by p^*, and typically satisfies $P^* \leq p_{th}^{FT}$, refers to the physical error rate at which the logical error rate of a finite-size fault-tolerant scheme equals the physical error rate, i.e., $p_L(p^*) = p^*$.

[21] Fault tolerance improves reliability when $p \lesssim \frac{1}{C}$, where C denotes the malignant-pair constant inferred from the fault-tolerant gadget analysis. For the concatenated Steane code, C is typically on the order of 10^4 [129], yielding threshold estimates almost $p_{th}^{FT} \approx 10^{-4}$.

codes cannot realize a universal logical gate set using local constant-depth circuits alone, and practical FTQC architectures must incorporate multiple fault-tolerant mechanisms. Moreover, Gottesman [133] showed that the complexity of a fault-tolerant circuit depends not only on the number of physical qubits but also on the total number of *locations*.

Spatial locality also imposes fundamental constraints on QECC parameters. The *Bravyi–Terhal no-go theorem* [134] establishes that two-dimensional geometrically local stabilizer codes cannot provide a passive self-correction feature at finite temperature because their energy barrier remains bounded, thereby necessitating active QEC. Moreover, the distance of a geometrically local stabilizer code in D dimensions is bounded as $d = O(n^{1-\frac{1}{D}})$, which reduces to $d = O(\sqrt{n})$ in two-dimensional lattices. The *Bravyi–Poulin–Terhal (BPT) [135] bound* constrains both scaling of k and d in a D-dimensional lattice with local stabilizer interactions, as given in Equation 5.

$$k\, d^{\frac{2}{D-1}} \leq O(n) \tag{5}$$

In two-dimensional architectures, Equation 5 reduces to Equation 6.

$$k\, d^2 \leq O(n) \tag{6}$$

This bound precludes two-dimensional geometrically local QECC from simultaneously achieving constant rate and linear distance. Accordingly, the normalized quantity $\frac{kd^2}{n}$ is used as a *finite-size* performance metric for two-dimensional quantum codes, in particular compared with the toric code that has $k = 2$, $d = L$, and $n = 2L^2$, and therefore saturates the BPT scaling up to constant factors[22], as discussed in section V.

Regardless of geometric locality, several coding bounds characterize the trade-off among block length, dimension, and distance. The *quantum Singleton bound*, also referred to as the *Knill-Laflamme bound*, establishes that any $[[n, k, d]]$ quantum code satisfies $n - k \geq 2(d-1)$, showing that larger code distance necessarily requires additional physical qubits[23] [136]. Codes that saturate the quantum Singleton bound with equality are called quantum *maximum distance separable* (MDS) codes[24]. Moreover, a non-degenerate $[[n, k, 2t+1]]_q$ code[25] capable of correcting arbitrary errors on up to t qudits must satisfy the *quantum Hamming bound* [137], also known as the *sphere-backing bound*, given by Equation 7.

$$K\Big(\sum_{j=0}^{t}(q^2-1)^j\binom{n}{j}\Big) \leq q^n \tag{7}$$

where K is the dimension of the codespace. By considering $q = 2$ for quantum codes to cover all non-identity Pauli errors, i.e., X, Z, and Y errors, and $K = 2^k$, Equation 7 can be written as 8.

[22] $\frac{kd^2}{n} = \frac{(2)(L)^2}{2L^2} = 1$

[23] This bound is derived from the classical Singleton bound, $k \leq n-d+1$, which is multiplied by two due to the co-cloning theorem.

[24] For instance, the $[[5, 1, 3]]$ code is an MDS code. More generally, the family of $[[n, n-2, 2]]$ codes also constitutes quantum MDS codes for even n.

[25] Quantum Hamming bound for degenerate quantum codes remains an open problem.

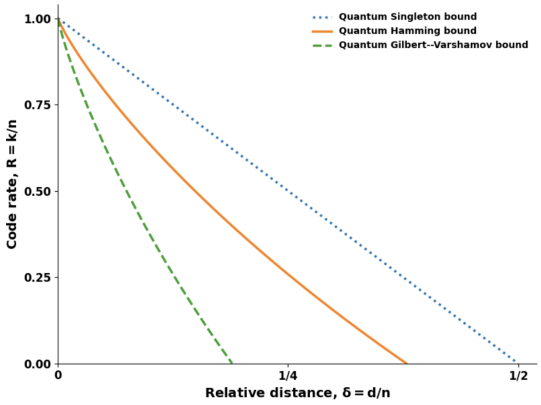


Fig. 7: Quantum error correction bounds for large n and $q = 2$.

$$2^k\Big(\sum_{j=0}^{t}3^j\binom{n}{j}\Big) \leq 2^n \tag{8}$$

where $\binom{n}{j}3^j$ counts weight j Pauli errors. A non-degenerate code that saturates this inequality is called a *perfect quantum code*. For instance, the $[[5, 1, 3]]$ code is both a quantum MDS code and a perfect quantum code because it saturates the quantum Singleton and Hamming bounds[26]. By contrast, the $[[7, 1, 3]]$ Steane code is neither a quantum MDS code nor a perfect code because there is a lot of *wasted space*[27].

Although the Singleton and Hamming bounds restrict achievable code parameters, the *quantum Gilbert–Varshamov bound* [138] provides an existence guarantee[28], and defines the lower bound of quantum codes as Equation 9.

$$2^k\Big(\sum_{j=0}^{d-1}3^j\binom{n}{j}\Big) \geq 2^n \tag{9}$$

The normalized parameters $R = k/n$ and $\delta = d/n$, which represent the *code rate* and *relative distance*, often called *distance rate*, respectively, capture the fundamental trade-off between QEC overhead and error protection. For large n, the bound can be expressed as Equation 10.

$$R \geq 1 - h_2(\delta) - \delta\log_2 3, \tag{10}$$

where

$$h_q(\delta) = -\delta\log_q\delta - (1-\delta)\log_2(1-\delta) \tag{11}$$

is the *binary entropy function*. This bound establishes the existence of *asymptotically good* quantum code families with a nonzero constant encoding rate $R > 0$ and a nonzero constant relative distance $\delta > 0$, provided that $h_2(\delta) + \delta\log_2 3 < 1$. It is worth noting that Equation 10 is an existence bound than a necessary condition that every asymptotically good code family must satisfy. The contrast

[26] $2^1\left[\binom{5}{0}\times 3^0 + \binom{5}{1}\times 3^1\right] = 32 \leq 2^5$

[27] $2^1\left[\binom{7}{0}\times 3^0 + \binom{7}{1}\times 3^1\right] = 44 < 2^7$, it occupies only part of the available Hilbert space

[28] If n,k, and d satisfy in Equation 9, an $[[n, k, d]]_q$ exists.

TABLE VI: QEC-relevant characteristics of qubit modalities

Modality	Noise Bias	Correlated Errors	Measurement Speed	Topological Code Fit	qLDPC Fit
Superconducting	Dephasing-biased	Crosstalk + leakage	Fast (100s ns–μs)	Excellent	Moderate (locality constrained)
Trapped-ions	Almost symmetric	Global motional modes	Slow (μs–ms)	Good	Strong (long-range connectivity)
Photonic	Erasure-biased	Loss and source correlations	Fast (ns scale)	Good (MBQC)†	Promising (network-native)
Neutral-atoms	Platform-dependent bias	Laser-induced correlations	μs scale	Good	Promising (reconfigurable geometry)
Spin Qubits	Dephasing-biased	Capacitive coupling crosstalk	Fast ($< \mu$s)	Good (local layouts)	Promising (CMOS-scalable)

†**MBQC**: Measurement-based quantum computation.

between the Gilbert–Varshamov existence region and the BPT locality bound highlights a fundamental architectural trade-off. For an asymptotically good code family, let $k/n = C_1$ and $d/n = C_2$, where $C_1, C2 > 0$ are constants. Substituting these values into the two-dimensional locality constraint of Equation 6 yields $(C_1 n)(C_2 n)^2 = C_1 C_2 n^3$, which is not upper-bounded by $\leq O(n)$ (i.e., is incompatible with the requirement $kd^2 = O(n)$). Therefore, asymptotically good quantum codes cannot be realized using strictly geometrically local stabilizer checks in a two-dimensional Euclidean architecture [50]. This incompatibility motivates the development of qLDPC, long-range couplers, and non-local stabilizer measurements, as discussed in section V. Figure 7 illustrates the asymptotic rate–distance trade-off for binary quantum codes. The Singleton and Hamming bounds impose upper limits on achievable code parameters, whereas the Gilbert–Varshamov bound identifies a region in which code existence is guaranteed[29].

Collectively, these theorems and bounds establish the theoretical feasibility of scalable FTQC while exposing fundamental trade-offs among block length, encoding rate, code distance, locality, and implementation overhead. Recent experiments have begun to validate key FTQC requirements across multiple qubit modalities, as discussed in the subsequent sections.

IV. Quantum Hardware for FTQC

Alongside theoretical advances in QECCs and FTQC, progress in quantum hardware is equally essential for realizing utility-scale FTQC. Key requirements include long coherence times, high-fidelity gates and measurements, scalable qubit connectivity, high fabrication yield, low-latency control electronics, cryogenic and cabling engineering, and system-level interconnects. In this section, we explore the architectural and operational characteristics of major qubit modalities and evaluate their suitability for implementing fault-tolerant protocols and supporting recent theoretical advances.

A. Qubit Modalities for Scalable and heterogeneous FTQC

Beyond architectural features, each qubit modality is associated with intrinsic physical features that affect its compatibility with particular QEC code families. The QEC-relevant characteristics of the major qubit modalities are summarized in Table VI, which highlights how their features influence their compatibility with topological and qLDPC codes. Architectures with strong geometric locality and nearest-neighbor connectivity[30] naturally align with topological codes, whereas modalities supporting long-range interaction and reconfigurable geometries are suited to qLDPC constructions and non-local stabilizer generators [139]. In addition to connectivity constraints, dominant noise mechanisms further influence code compatibility. The movement ability of ions in trapped-ion systems enables *all-to-all* connectivity, which facilitates fault-tolerant operations with fewer physical qubit overhead [104]. Photonic systems are typically erasure-dominated due to photon loss [140], while several solid-state platforms, including superconducting and spin qubits, may exhibit dephasing-biased noise characterized by $T_\phi \ll T_1$, implying $p_z \gg p_x$. Such biased-noise models motivate tailored QEC codes and decoders that exploit bias to improve thresholds and reduce overhead [30]. Moreover, measurement speed also determines syndrome extraction latency, the feasibility of real-time decoding, and the latency of feedback control [35].

Recognizing modality-dependent strengths and limitations motivates hardware-aware FTQC and the use of heterogeneous quantum computing architectures [141]. In HeDQC architectures, QEC operations can be distributed across QPUs according to their native advantages. For instance, data qubits may reside in QPUs that offer longer coherence time (e.g., trapped-ion), while ancilla/check qubits may reside in QPUs that support fast measurement and reset operations (e.g., superconducting circuits). Figure 8 illustrates a generalized HeDQC architecture in which specialized *nodes* perform different tasks based on the best-suited qubit modality. *Compute nodes* leverage fast-gate modalities for rapid processing, *memory nodes* utilize long-coherence qubits for stable storage [142], and *communication interfaces* rely on photonic qubits to interconnect with other HeDQC modules [143]. Quantum transducers facilitate coherent conversion between different modalities, while a top-level classical control system coordinates system-level operations. It is worth noting that quantum switches are integral components of long-distance HeDQC systems. Recently, Cisco introduced a prototype universal quantum switch [67] capable of interconnecting quantum systems from different hardware vendors by translating between heterogeneous quantum information formats while preserving quantum coherence. Operating over existing room-temperature fiber-optic infrastructure with low signal degradation, such architectures are expected to provide a key interconnect layer for scalable networked FTQC.

[29]This figure provides a qualitative comparison rather than an exact numerical plot because the Hamming and Gilbert–Varshamov bounds contain the binary entropy function.

[30]Nearest-neighbor connectivity refers to architectures in which qubits can directly interact with adjacent qubits in a fixed layout (e.g., planar superconducting platforms). Therefore, interaction between non-adjacent qubits requires multiple additional operations (e.g., *SWAP* gates), which increases space-time overhead.

Du et al. introduced a related modular architecture in which a shared-quantum gate processing unit (S-QGPU) executes remote gate operations (RGOs) between distributed QPUs. The S-QGPU comprises a pool of small (2×2) processing units that mediate entangling operations across modules. According to the evaluations, this architecture can be more cost-effective than conventional entanglement-based interconnected paradigms by allowing each QPU to allocate more qubits to computations and enhance computational capacity [144].

Beyond the intrinsic suitability of qubit modalities for specific QEC families, a second wave of considerations concerns their scalability and system-level compatibility with DQC and DQECCs [145]. DQC-friendly and modular-compatible structure of each qubit modality depends on architectural scalability, interconnect mechanisms, fabrication maturity, and the ability to generate high-fidelity inter-module entanglement [146]. Table VII compares major qubit modalities in terms of their compatibility with DQC and DQEC. Superconducting circuits have demonstrated relatively large integrated processors, but are constrained by cryogenic wiring, packaging, and control I/O scaling [107]. Trapped-ion platforms provide long-range connectivity, useful for DQEC, but scaling is limited by laser control complexity, trap engineering, and motion management [34]. Photonic architectures are typically network-oriented and well-suited to distributed entanglement generation, yet remain challenged by photon loss, source efficiency, and detector requirements [63]. Neutral-atom platforms offer flexible geometries and large reconfigurable arrays, while they face constraints related to laser stability, atom reloading, and array control [105]. Semiconductor spin qubits are promising for dense CMOS-compatible integration and long-term manufacturability [57], but currently face fabrication variability and interconnect maturity challenges. These modality-dependent trade-offs determine how logical qubits can be partitioned, interconnected, and protected across distributed architectures, ultimately shaping their viability for large-scale FTQC [147].

Several studies have demonstrated the relevance of hardware-aware QEC and FTQC. Takeda et al. [148] introduced a three-qubit QEC scheme for protecting against dephasing errors in spin qubits. Their architecture leverages a *iToffoli* gate to perform a three-qubit conditional rotation, enabling direct correction of phase-flip errors within a compact device. By addressing the phase-flip errors, the dominant error of spin qubits, arising from quasi-static phase noise, their study highlights the potential of spin qubits for scalable FTQC. Xue et al. [149] demonstrated a silicon spin qubit processor with single- and two-qubit gate fidelities exceeding 99.5%. These fidelities place the corresponding physical error rates $\sim 1\%$, approaching the regime required for surface code correction and highlighting the features of semiconductor spin qubits in scalable FTQC implementations.

Neutral-atom architectures have also shown rapid progress toward large-scale programmable QEC schemes. Bluvstein et al. [150] demonstrated a 448-qubit neutral-atom processor supporting surface code correction below threshold. Moreover, they implemented universal logical operations using three-dimensional codes and illustrated logical entanglement, lattice surgery, and teleportation-based gates. Chiu et al. [151] also utilized an atom reloading mechanism to sustain continuous operation of a $3,000$-atom array for more than two hours, demonstrating the potential of neutral-atoms for maintaining large qubit registers over long computation times. These achievements position neutral-atoms as a promising candidate for scalable, universal FTQC architectures.

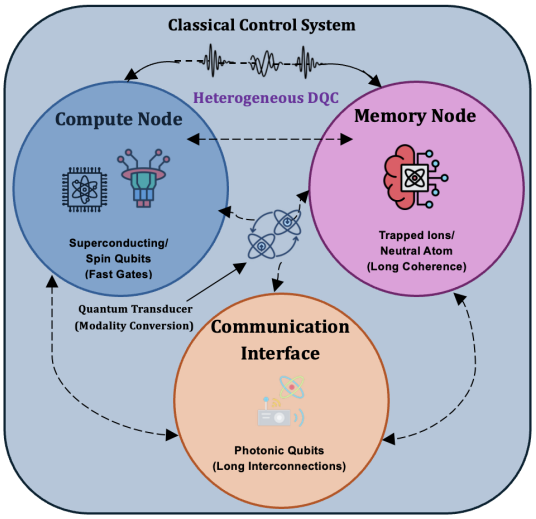


Fig. 8: General structure of a HeDQC architecture.

B. *Quantum Data Center Interconnects for FTQC*

Alongside advances in quantum processors, scalable FTQC requires advances in system-level control and interconnect architectures that can support both intra-QPU long-range interconnections within quantum modules and inter-QPU communication across multiple QPUs. For instance, Wang et al. [152] introduced a wireless terahertz cryogenic interconnect that connects millikelvin qubits with room-temperature control electronics while reducing the *heat-to-information* transfer ratio. Their architecture employs CMOS-based transceivers operating at 260 GHz, together with a cold-to-hot egress and a hot-to-cold ingress, illustrating how hardware interconnect innovation can support scalable FTQC systems.

As quantum systems scale beyond monolithic processors, fault-tolerant operations increasingly rely on the ability to distribute entanglement, execute non-local gates, and coordinate distributed stabilizer measurements with low latency and high fidelity [147]. Several classes of couplers are therefore required, including the *C-coupler* [153] for on-chip non-local connectivity within a quantum memory, the *M-coupler* for chip-to-chip connections to form a larger chip within a module, and the *l-coupler* for module-to-module communication through cable or microwave links. At a larger scale, efficient entanglement distribution architectures [154] are required to interconnect multiple QPUs and realize a *quantum data center* (QDC). A QDC provides this infrastructure by integrating QPUs, communication qubits, optical or microwave links, switching fabrics, routing mechanisms, and shared entanglement generation resources. From the perspective of large-scale FTQC, the QDC interconnect directly affects the latency, fidelity, synchronization overhead, and resource contention

TABLE VII: Modality-dependent characteristics associated with DQC and DQEC

Modality	Scaling Bottleneck	Connectivity Mechanism	Available QPU Size	DQEC Potential
Superconducting	Cryogenic wiring, control I/O	Microwave coupling	$\sim 100 - 1000$ qubits	Moderate (modular cryo emerging)
Trapped-ions	Laser control, trap scaling	Photonic interconnects	$\sim 10 - 100$ qubits per trap	Strong (remote entanglement)
Photonic	Source and detector efficiency	Native optical networking	Cluster-state dependent	Very strong (flying qubits)
Neutral-atoms	Laser stability, atom loss	Rydberg interactions	$\sim 100 - 1000$ atoms	Promising (reconfigurable geometry)
Spin qubits	Device non-uniformity	Electrical coupling	$\sim 10 - 100$ qubits	Emerging (CMOS integration)

associated with non-local gates, inter-QPU communication, and distributed stabilizer measurements [155].

In general, QDC interconnects can be classified into *switch-centric* and *server-centric* architectures. In switch-centric architectures, such as Clos [156], Fat-Tree [157], and HyperX [158] topologies, QPUs act primarily as endpoints connected through dedicated optical switching fabrics. As illustrated in Figure 9(a), QPUs may be organized into racks, with intra-rack communication supported by top-of-rack *near-infrared (NIR)* switches. These switches operate near the native resonant frequency of the communication qubits and may integrate photon detectors and quantum resources required for ebit[31] generation, including *Bell-state measurement* (BSM) modules and entanglement sources. Inter-rack communication is supported by higher-level *telecom* switches, which operate at fiber-compatible wavelengths and are compatible with standard photonic components. Although this architecture can provide direct or near-direct optical connectivity among QPUs and reduce reliance on intermediate quantum processors, its scalability may be limited by the cost and complexity of large switch fabrics, including switch port count requirements, optical channel availability, and BSM resources provisioning.

In server-centric architectures, such as QFly [146], BCube [159], and DCell [160] architectures, QPUs or quantum servers participate in establishing multi-hop entanglement paths. As shown in Figure 9(b), each QPU may be equipped with multiple communication ports or network interfaces. In contrast to switch-centric architecture, this architecture does not generally provide all-to-all direct connectivity. Instead, end-to-end ebits between some QPU pairs must be established through multi-hop repeater chains, where intermediate QPUs generate elementary entangled links and perform entanglement swapping. For example, in the BCube architecture [159], a level-k architecture with n-port switches can accommodate n^{k+1} QPUs and each QPU requires $k+1$ communication ports. This modularity improves scalability and path diversity, but it also introduces additional scheduling and resource management challenges because intermediate QPUs consume communication qubits and may become a bottleneck. Since QDC architectures are adapted from classical data-center networking, other scalable classical interconnects may also motivate quantum counterparts. For example, RNG [161], the first flat data-center architecture experimentally developed on Amazon servers, can motivate new directions for reducing hierarchical switching overhead in future QDC designs.

The QDC interconnects also require the link-level protocols to generate point-to-point entanglement. Depending on the hardware interface and communication distance, ebits may be generated using *emitter-emitter*[32], *emitter-scatterer*[33], or *scatterer-scatterer*[34] protocols. The emitter-emitter and emitter-scatterer protocols are primarily suitable for short-range intra-rack communication, whereas the scatterer-scatterer protocol can support longer inter-rack communication through telecom-compatible photons, non-degenerate entanglement sources, and BSM resources [154].

Although QDC architectures are inspired by classical data-center networks, they establish entanglement paths for tele-gate and tele-data operations, rather than forwarding quantum packets. Although bipartite Bell pairs support point-to-point interactions, coordinated operations across multiple modules may require multipartite entangled resources. A representative example is the n-qubit *GHZ* state [53], $|\mathrm{GHZ}_n\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle^{\otimes n} + |1\rangle^{\otimes n}\right)$, which provides strong global correlations suitable for distributed parity measurements and stabilizer-based operations. However, the *GHZ* state is highly fragile to qubit loss, as tracing out a single qubit destroys the entanglement among the remaining subsystems. In contrast, the W state [162], $|\mathrm{W}_n\rangle = \frac{1}{\sqrt{n}}\left(|100\cdots0\rangle + |010\cdots0\rangle + \cdots + |0\cdots01\rangle\right)$, offers a distinct multipartite resource whose residual subsystems can remain entangled after the loss of one qubit, promising for lossy quantum networks and distributed FTQC. Consequently, the suitability of a QDC architecture for FTQC depends not only on its connectivity graph but also on the type and quality of entanglement it can distribute, together with the entanglement generation rate, heralding probability, link fidelity, communication qubit availability, optical loss, switch latency, and resource contention [147].

V. Quantum Error Correction for FTQC

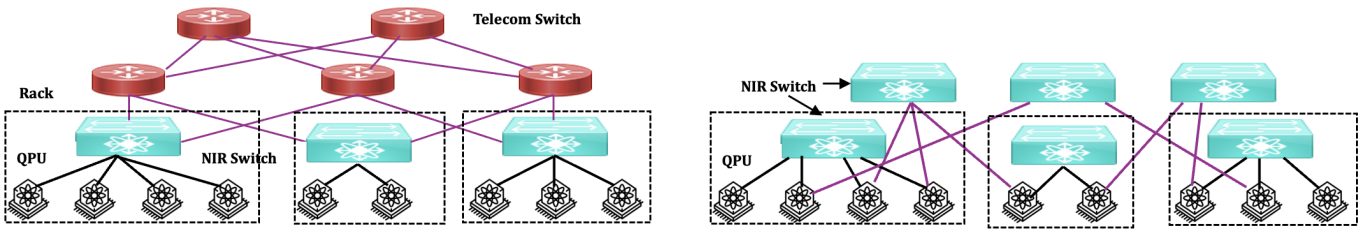
Quantum error correction is the central operation-time measure for suppressing physical errors to the logical error rates required for FTQC. A subspace quantum error-correcting code is typically characterized by parameters $[[n, k, d]]$, where n, k, and d denote the number of physical qubits, the number of encoded logical qubits, and the code distance, respectively. The distance is the minimum weight of a nontrivial logical operator, equivalently the minimum number of physical qubits

[31] One unit of entanglement corresponding to an entangled Bell pair.

[32] Both nodes emit a photon entangled with their local qubit state. The photons interfere at an intermediate station, where a successful Bell state measurement heralds entanglement between two QPUs.

[33] One node (i.e., emitter) generates a photon entangled with its local communication qubit and sends it to the second QPU (i.e., scatterer). A successful photon detection event at the second QPU heralds entanglement between the two communication qubits.

[34] Both communication qubits act as scatterers, while external entanglement sources generate photon pairs that are sent toward both QPUs and a BSM module; successful photon detections and a BSM outcome herald entanglement between the QPUs.



(a) Switch-centric (b) Server-centric

Fig. 9: Quantum data center architectures.

that must be acted on to transform one logical state into another [37].

For many QECC families operated below threshold, the logical error rate $\mathcal{E}_d$ decreases approximately exponentially with increasing code distance. This behavior is typically expressed by the *Empirical scaling* relation, given by Equation 12.

$$\mathcal{E}_d = C\left(\frac{p}{p_{\text{th}}}\right)^{\frac{d+1}{2}} \tag{12}$$

where p and p_{th} denote the physical error rate and the QEC threshold, respectively, and C is a constant that depends on the code family, syndrome extraction circuit, and noise model. Equivalently, the ratio $\frac{p}{p_{th}}$ can be interpreted as the inverse of the *logical error suppression factor*, i.e., $\frac{p}{p_{th}} = \frac{1}{\Lambda}$, where $\Lambda = \frac{\mathcal{E}_d}{\mathcal{E}_{d+2}}$ quantifies the improvement in logical error rate when the code distance is increased by two[35] [75]. Although increasing the code distance can, in principle, suppress the logical error rate, it also requires larger code blocks, additional stabilizer measurements, more ancilla qubits, and greater decoding complexity. These additional resources introduce new error locations and may offset the benefits of increasing code distance [163]. This trade-off can be analyzed in terms of QEC threshold, as discussed in section III.

In general, constructing logical qubits (LQs) is both a coding-theoretic and an architectural challenge because the placement of data and check qubits determines stabilizer measurement locality, decoding complexity, syndrome extraction scheduling, and control overhead, specifically in distributed QECCs [164]. Figure 10 illustrates how these architectural considerations differ between monolithic and distributed quantum computing. In MQC, as depicted in Figure 10(a), all physical qubits associated with an LQ block and corresponding syndrome extraction (SE) circuit reside within a single QPU, forming *local logical qubits* (LLQs) [9]. Although this arrangement benefits from localized operations, shorter intra-QPU communication, and simpler control, it is constrained by a finite qubit count, intra-QPU resources, and wiring density, which bound the number and size of LLQs. As depicted in Figure 10(b), in DQC, LQs may either remain local within individual QPUs (e.g., LQ#1 through LQ#i) or be organized as *distributed logical qubits* (DLQs), whose data and/or check qubits are partitioned across multiple QPUs (e.g., LQ #i+2 and LQ #j+1) [9]. DLQs can relax single-QPU capacity constraints and enable larger code blocks or non-local stabilizer structures, but they introduce new system-level requirements. In particular, DQECCs require cross-QPU coordination, entanglement-assisted stabilizer measurements, synchronization of local SE circuits, decentralized syndrome aggregation, and distributed decoding. These functions are supported by a control backbone that may include classical coordination electronics, syndrome processing units, inter-QPU communication links, and, at the cryogenic level, a shared cryogenic backplane for hosting multiple QPUs and routing control signals. Therefore, while DQC expands the architectural design space for scalable QEC, it shifts the overhead associated with local qubit resources to inter-QPU concerns.

The $[[9,1,3]]$ Shor code is the first introduced QECC capable of correcting an arbitrary single-qubit error [165]. This code is a concatenation of two 3-qubit codes to encode one logical qubit into nine physical qubits[36]. Subsequently, Steane [167] proposed a more efficient $[[7,1,3]]$ CSS code, which achieves the same error-correcting capability with lower QEC overhead. Laflamme et al. [168] later developed the $[[5,1,3]]$ code, in which saturates the quantum Hamming bound [137] and quantum Singleton bound [136].

These algebraic codes established the foundations of active QEC and motivated the development of structured code families with distinct parameters, stabilizer constructions, and logical operations. In the following subsections, we explore several prominent families, including topological, subsystem, qLDPC, and dynamical codes, with emphasis on their constructions, recent experimental demonstrations across different qubit modalities, and existing distributed implementations.

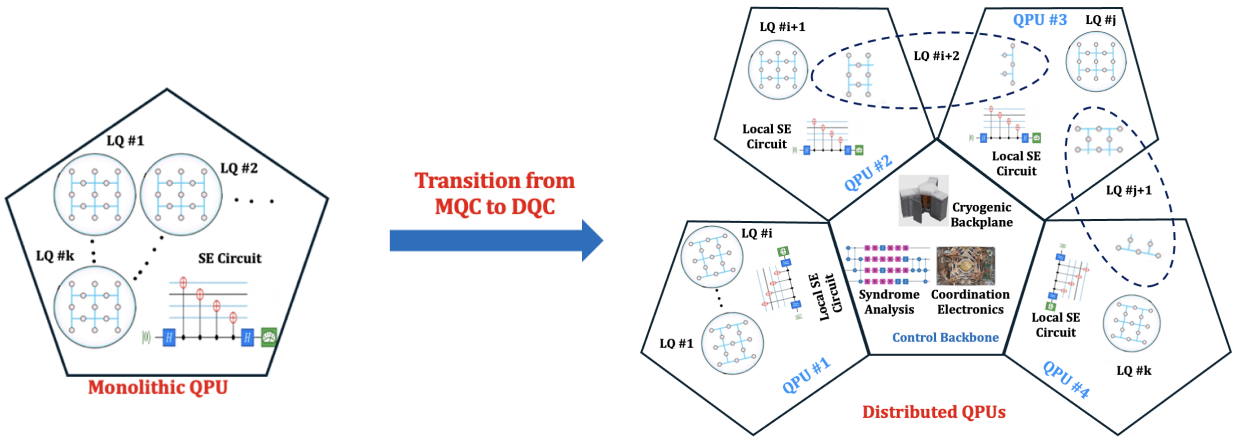
A. Topological Codes

Topological codes constitute a prominent class of CSS QECCs in which quantum information is encoded into global topological degrees of freedom of an underlying geometric manifold, such as a torus or a planar surface. In these constructions, stabilizer generators are geometrically local and logical operators correspond to non-contractible topological loops on the underlying lattice, which makes the encoded information robust against local and uncorrelated errors [169].

The structure of homological CSS codes is determined by the topological invariants of the underlying cell complex. For an orientable surface, the Euler characteristic (χ) is given by

[35]For example, $\Lambda_{35} = \frac{\mathcal{E}_3}{\mathcal{E}_5}$ represents the logical error suppression factor obtained by increasing the code distance from $d = 3$ to $d = 5$.

[36]Concatenation of two classical codes can yield robust, bias-adapted codes with reduced overhead, such as *elevator codes*, which are a concatenation of repetition phase-flip codes and high-rate bit-flip codes [166].



(a) Monolithic quantum computing

(b) Distributed quantum computing

Fig. 10: Logical qubit structure in monolithic and distributed quantum computing.

TABLE VIII: Topological properties of common surfaces relevant to homological CSS codes.

Surface	g	χ	b_0	b_1	b_2	k	Example Code
Sphere	0	2	1	0	1	0	None
Torus	1	0	1	2	1	2	Toric code
Double torus	2	-2	1	4	1	4	Toric code ($k=4$)
Triple torus	3	-4	1	6	1	6	Toric code ($k=6$)

$\chi = V - E + F$, where V, E, and F denote the number of vertices, edges, and faces, respectively. This property also holds for all convex *Platonic solids*, such as the tetrahedron, cube (hexahedron), and octahedron[37]. Equivalently, for a closed orientable surface of genus g, the Euler characteristic satisfies $\chi = 2 - 2g$, where g refers to the number of handles on the surface. It can also be expressed in terms of *Betti numbers* as $\chi = b_0 - b_1 + b_2 - b_3 + \cdots$, where b_i denotes the i-th Betti number of the surface. In two-dimensional homological CSS codes on closed orientable surfaces, b_0 corresponds to the number of connected components, b_1 represents the number of non-contractible loops (e.g., the number of encoded logical qubits), and b_2 defines the number of cavities (e.g., closed surfaces) [170]. Table VIII summarizes topological quantities for representative closed orientable surfaces and their implications for homological CSS codes.

From the perspective of code parameters, standard two-dimensional topological stabilizer codes obey the locality constraints imposed by the BPT bound. In particular, surface code and toric code families typically encode a constant number of logical qubits, $k = O(1)$, while achieving distance $d = O(\sqrt{n})$, thereby saturating the BPT scaling up to constant factors. This locality-preserving structure makes them well-suited to architectures with nearest-neighbor interactions, although it also implies substantial space overhead when very low logical error rates are required [75]. Motivated by this topological perspective, we briefly review several prominent families of topological QECCs, including toric codes, surface codes, and color codes.

[37]For instance, a cube has $V = 8$, $E = 12$, and $F = 6$, consistent with the Euler characteristic.

1) Toric Code: The intuition behind toric codes can be traced back to the classical repetition code, which can be viewed as a one-dimensional cyclic code. Kitaev [171] extended this idea to two dimensions and introduced the toric code as the Cartesian product of two cyclic codes that independently correct bit-flip and phase-flip errors. This code is defined on an $L \times L$ square lattice with periodic boundary conditions in both spatial directions, forming $[[2L^2, 2, L]]$ codes on a torus. Physical qubits are placed on $2L^2$ edges. The X-type stabilizer generators, often called *star operators*, are associated with vertices and act on the four incident edges, while Z-type stabilizer generators, often called *plaquette operators*, are associated with faces and act on the four boundary edges of each plaquette.

Due to global constraints, there are $L^2 - 1$ independent X-type and $L^2 - 1$ independent Z-type stabilizer generators[38]. Logical operators correspond to tensor products of Pauli operators along non-contractible loops around the two independent cycles of the torus, which cannot be expressed as products of stabilizer generators. Each lattice encodes two logical qubits[39], and the code distance[40] scales linearly with the lattice dimension as $d = \Theta(\sqrt{2L^2}) = \Theta(L)$.

The periodic boundary conditions of the toric code pose significant engineering challenges for planar quantum hardware because they require effective wrap-around connectivity. For this reason, planar surface code variants are often more practical for experimental implementation, while preserving the same stabilizer structure and asymptotic distance scaling. Nevertheless, toric code has been experimentally explored across several architectures, including superconducting [169], trapped-ion [46], neutral-atom [172], and photonic [140] platforms.

2) Surface Code: The surface code is a planar variant of the toric code in which periodic boundary conditions are

[38]A lattice contains L^2 star operators and L^2 plaquette operators, however, the product of each type of operator is the identity, $\prod_{v=1}^{L^2} S_v^{(X)} \prod_{f=1}^{L^2} S_f^{(Z)} = \mathbb{I}$. Therefore, one stabilizer from each set must be eliminated to achieve independent stabilizer generators.

[39]The number of logical qubits corresponds to the number of independent homologically nontrivial cycles.

[40]The length of the shortest non-contractible loop.

replaced by open boundaries, which eliminates the need for a toroidal layout and makes the code compatible with two-dimensional hardware architectures with local interactions. Although this reduces the number of logical qubits to *one*, yielding $[[2L^2, 1, L]]$ codes, it preserves favorable distance scaling [75]. Its geometric locality, low-weight stabilizer generators[41] (e.g., $\ell = 4$), high error thresholds (e.g., on the order of $\sim 1\%$ under the circuit-level noise model), and scalable code distance position the surface code as one of the leading candidates for QEC on superconducting and other nearest-neighbor hardware platforms.

From an experimental perspective, the surface code has been implemented and validated across multiple qubit modalities. Krinner et al. [173] implemented distance-3 surface codes on a 17-qubit superconducting QPU and demonstrated repeated QEC cycles. Their evaluations showed that the logical error probability was below 3% per cycle and logical fidelity reached approximately 99.6% with a minimum-weight perfect matching decoding algorithm. This work provides an important experimental validation of surface code capabilities even for small logical qubits, which are feasible on near-term quantum hardware. Zhao et al. [174] experimentally evaluated the repeated error correction capability of distance-3 surface codes on a 17-qubit superconducting processor, called *Zuchongzhi 2.1*. Their results demonstrated logical error suppression through consecutive error correction cycles.

Google experimentally validated below-threshold surface codes on its *Willow* superconducting processor. They reported that the logical error rate is suppressed by a factor of $\Lambda = 2.14 \pm 0.02$ by increasing the code distance by two. Moreover, the lifetime of the logical qubit exceeded that of the best physical qubit by a factor of 2.4 ± 0.3, highlighting operation beyond the break-even point. Additionally, repetition code experiments up to distance 29 further revealed that logical performance is ultimately limited by rare correlated error events, occurring approximately once every hour or every 3×10^9 cycles [175]. The results demonstrated significant hardware advances in Google's 105-qubit *Willow* QPU compared with its earlier 49-qubit *Sycamore* QPU [176]. In particular, the qubit relaxation time, T_1, improves from 20 μs to 70 μs, while the 2Q gate error decreases from 0.6% to 0.4% and the 1Q gate error from 0.1% to 0.05%. Furthermore, the idle error rate was reduced from 2.5% to 0.8% (i.e., through a DD technique as discussed in section III), and the readout error decreased from 2% to 0.8%. In addition to these hardware-level improvements, the experiment also demonstrated enhanced performance in terms of logical error suppression, longer coherence time, and improved detection probabilities.

Surface code implementations have also been investigated across various modalities beyond superconducting qubits. Berthusen et al. [177] implemented $[[33, 1, 4]]$ four-dimensional surface codes on *Quantinuum's H2* trapped-ion processor[42] and demonstrated single-shot QEC using bare ancilla qubits. Their experimental results showed that the 4D code outperforms 2D surface code memories in both fault-tolerant and single-shot error correction regimes. Iqbal et al. [1] leveraged mid-circuit measurement and feedforward on *Quantinuum's H1* trapped-ion processor to realize deterministic non-unitary dynamics and a constant-depth, real-time procedure for preparing a toric code ground state. In photonic systems, Yao et al. [178] experimentally demonstrated topological QEC on photonic qubits, which encoded state can be protected against arbitrary single-qubit errors.

Recent code design efforts have also endeavored to reduce the overhead of surface codes. The *Yoked* surface codes, introduced by Gidney et al. [179], concatenated surface code patches into a high-density parity-check code to form a hierarchical architecture arranged on a rectangular grid. The inter-patch parity checks, called *yokes*, are measured using lattice surgery, enabling the scheme to operate under nearest-neighbor connectivity constraints. The design targeted realistic physical error rates on the order of 10^{-3} and reduced the number of physical qubits per logical qubit by up to approximately a factor of three compared with conventional surface code memories, making it promising for two-dimensional FTQC architectures. Bone et al. [180] developed a distributed surface code implemented with nitrogen-vacancy centers in diamond, where a four-qubit *GHZ* state supports stabilizer measurements across separate nodes. Their model showed that realistic decoherence during entanglement generation reduces the gate and measurement error thresholds by at least a factor of three relative to idealized noise models and identified the entanglement generation to decoherence rate ratio of approximately 4×10^2 as a benchmark for fault-tolerant operations.

3) Color Code: Color codes are a family of two-dimensional topological stabilizer codes defined on trivalent lattices whose faces are three-colorable. In the standard construction, data qubits are placed at the lattice vertices, and each face supports both an X-type and a Z-type stabilizer acting on the qubits around that face. These codes can be constructed on a variety of three-colorable lattice structures embedded in different geometric surfaces (e.g., the 7-qubit color code can also be viewed as a geometric generalization of the Steane code [181]). Their geometric locality makes color codes compatible with two-dimensional hardware layouts, including superconducting and neutral-atom platforms.

A primary advantage of color codes is their ability to implement the full Clifford group transversally, which facilitates fault-tolerant logical gate constructions [182]. This property makes color codes promising for operations requiring frequent Clifford gates and for code switching techniques, detailed in section III, in which different code families are used for different logical tasks. However, color codes typically exhibit lower QEC threshold than surface codes and have higher-weight stabilizer generators, which can increase circuit depth, measurement complexity, and susceptibility to correlated hook errors during syndrome extraction. In addition, color code decoding is more complex than that of the surface code because it is not directly reducible to a single minimum-weight perfect matching problem, which can result in slower and often less accurate decoding procedures. Google Quantum AI's researchers, Gidney and Jones, have shown that color codes

[41] Stabilizer weight refers to the number of data qubits on which a stabilizer generator acts.

[42] The corresponding hardware specifications and datasets are available at https://github.com/Quantinuum/quantinuum-hardware-specifications.

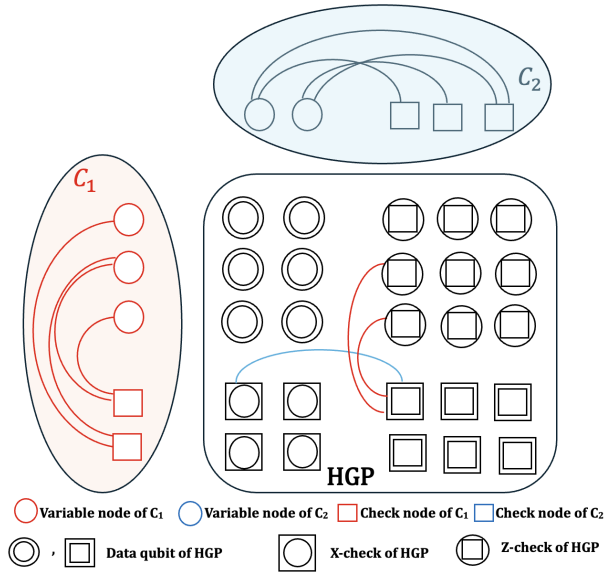


Fig. 11: HGP construction from two classical Tanner graphs.

require roughly twice as many physical qubits as the surface code to achieve a similar logical performance [183].

Recent studies have also explored the potential of color codes to reduce the overhead of non-Clifford resource preparation. Google has reported experimental implementations of color codes on superconducting qubits, demonstrating their potential for producing high-fidelity magic states for magic state cultivation [98]. Although surface codes remain the primary framework for computational tasks, color codes may offer advantages for specialized operations such as magic state generation and code switching. Experimental and platform-specific studies have further examined color code implementations in trapped-ion systems [85], [181] and neutral-atom [184] platforms, highlighting their potential role in hardware-aware FTQC architectures.

B. qLDPC Codes

Quantum low-density parity-check (LDPC) codes constitute a promising class of QECCs for scalable FTQC because they achieve a constant encoding rate with substantially improved distance scaling. In particular, combining two asymptotically good classical LDPC codes with parameters $[n, \Theta(n), \Theta(n)]$ via geometric or hypergraph product constructions yields a quantum LDPC (qLDPC) code with parameters $[[\Theta(n^2), \Theta(n), \Omega(\sqrt{n})]]$ [185]. In contrast to two-dimensional topological codes, which are constrained by geometric locality and imply the BPT trade-off, qLDPC codes relax this restriction by keeping stabilizer generators sparse while allowing them to be geometrically non-local, as discussed in section III. This flexibility improves the rate–distance trade-off and makes qLDPC codes promising for reducing QEC overhead, particularly in modular, reconfigurable, or distributed architectures that can support long-range connectivity and non-local stabilizer measurements. Interest in qLDPC codes further intensified after Gottesman's study [133], which demonstrated that families of constant-rate quantum codes can enable FTQC with constant overhead under some assumptions.

A canonical construction of qLDPC codes is the *hypergraph product* codes introduced by Tillich and Zémor, which are constructed via a Kronecker product of two classical binary codes ($C_1 \subseteq \mathbb{F}_2^n$ and $C_2 \subseteq \mathbb{F}_2^n$), represented by Tanner graphs[43], or equivalently by sparse parity-check matrices $H_1 \in \mathbb{F}_2^{r_1 \times n_1}$ and $H_2 \in \mathbb{F}_2^{r_2 \times n_2}$ [186]. The HGP code places data qubits on two spaces, $(V_1 \times V_2)$ and $(C_1 \times C_2)$, where V_i and C_i denote variable and check node sets of the Tanner graph associated with H_i, respectively. The X-type checks are indexed by $C_1 \times V_2$, while the Z-type checks are indexed by $V_1 \times C_2$, which can be written in product form, given in Equation 13.

$$\begin{aligned} H_X &= \left[H_1 \otimes I_{n_2} \;\middle|\; I_{r_1} \otimes H_2^T \right] \\ H_Z &= \left[I_{n_1} \otimes H_2 \;\middle|\; H_1^T \otimes I_{r_2} \right] \end{aligned} \tag{13}$$

The HGP structure guarantees the CSS commutativity condition $H_X H_Z^T = 0$ by construction. Figure 11 illustrates this product structure, in which data qubits occupy the product of variable–variable (i.e., double-circle) and check–check (i.e., double-square) node sets, while X- and Z-type stabilizer generators arise from the corresponding check–variable product spaces[44]. For input codes with parameters $[n_i, k_i, d_i]$, the HGP construction yields a quantum code with parameters $[[n = n_1 n_2 + (n_1 - k_1)(n_2 - k_2),\ k = k_1 k_2,\ d = \min\{d_1, d_2\}]]$.

Quantum LDPC codes, unlike their classical counterparts, cannot be constructed using naive random sparse constructions because of the commutation constraints of the stabilizer generators. Therefore, Evra et al. [187] broke the $\sqrt{n}$ distance barrier by incorporating the Kronecker product of good LDPC codes and high-dimensional expenders (HDX), in particular Ramanujan complexes. Their construction achieved $d \in \Omega(\sqrt{n} \log n)$ for three-dimensional Ramanujan complexes and $d \in \Omega\left(\sqrt{n \log n}\right)$ for two-dimensional Ramanujan complexes. Moreover, the distance of qLDPC codes has been improved to $d \in \Theta(\sqrt{n} \log^s n)$ using an iterated Kronecker product of two cosystolic expenders, introduced by Kaufman and Tessler [188]. Hastings et al. [189] introduced *fiber bundle codes*, which augment tensor product constructions with twisting to improve distance scaling. In this construction, a random classical code serves as the base and a cycle graph serves as a fiber, yielding distance scaling $d \in \Omega\left(\frac{n^{3/5}}{\text{polylog}(n)}\right)$ and the number of encoded qubits scaling $k \in \Theta(n^{3/5})$.

Panteleev and Kalachev [190] introduced *lifted product codes*, which improve the distance scaling of qLDPC constructions to $d \in \Omega\left(\frac{n^{1-\alpha/2}}{\log(n)}\right)$, for $0 \leq \alpha < 1$, while the number of logical qubits scales as $k \in \Theta(n^\alpha \log(n))$. Lifted product codes combine graph lifting techniques from classical LDPC codes and algebraic product constructions. Breuckmann and Eberhardt introduced *balanced product* (BP) codes, which combine two codes sharing a group symmetry through a quotient of their tensor product. This construction achieves distance and dimension scaling $d \in \Omega(n^{3/5})$ and $k \in \Theta(n^{4/5})$, respectively [191]. For classical codes C and D with a group

[43] A Tanner graph is a bipartite graph representation of a parity-check matrix, where variable nodes correspond to code bits and check nodes correspond to parity-check constraints.

[44] The constructed HGP code contains a total of 40 connections, of which only three representative connections are depicted in Figure 11 for clarity.

action H, acting on the left of C and on the right of D, the BP is denoted by $C \otimes_H D$. This construction can be viewed as a quotient of the tensor product $C \otimes D$ under the symmetry group action H, where elements belonging to the same orbit are identical. By factoring out these symmetries, the construction reduces the number of physical qubits while preserving the sparse structure and improving code parameters.

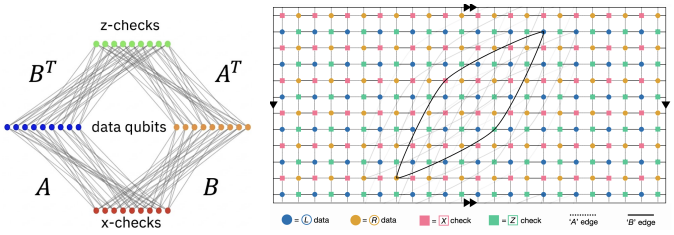
Despite their promising asymptotic properties, implementing qLDPC codes on available platforms, especially in superconducting architectures is challenging, because their stabilizer generators are non-local and incompatible with planar nearest-neighbor layouts. Several approaches have been proposed to bridge this gap. Tremblay et al. introduced an l-planar structure and stabilizer measurement scheduling via optimized edge coloring [192]. Moreover, long-range operations mediated by resonator-induced phase interactions offer another possible pathway for realizing qLDPC codes in superconducting systems [193]. Alternatively, reconfigurable atom-array platforms offer a complementary approach, as their dynamic connectivity can support non-local stabilizer measurements while maintaining relatively low decoherence [194].

Strikis and Berent [143] proposed a modular implementation of HGP codes by tailoring the hardware connectivity to the code structure. In their construction, a repetition code and a two-dimensional surface code define the intra-modular connectivity and inter-modular connectivity, respectively. The tensor product of these two codes constitutes a three-dimensional surface code, which can be applied to a special case of BP codes and to construct fiber bundle complexes. This framework could be generalized by permitting heterogeneous intra-modular connectivity across multiple modules.

Riverlane researchers introduced a novel family of qLDPC codes, known as *directional codes*, and mapped them onto commonly available hexagonal and square-grid topologies [195]. They have leveraged the *iSWAP* gate[45], which is well-suited to superconducting qubits, to reduce the need for long-range interactions. The simulation results indicated that directional codes outperform both the *rotated planar code* (RPC) and bivariate bicycle codes, requiring only 18.75%–45% of the physical qubits needed by these codes under the physical error rate of $p = 10^{-3}$ and code distances up to 10. This research not only demonstrated the compatibility of qLDPC codes with available hardware connectivity constraints, but also reduces the reliance on complex, long-range, high-connectivity architectures for FTQC.

Progress in semiconductor spin qubits also indicates the feasibility of qLDPC code implementations. Undseth et al. [196] developed a silicon spin qubit platform with a coherent shuttling bus that enables dynamic connectivity across multiple sites, and configured an effective five-qubit QPU capable of implementing X- and Z-type stabilizer generators of weight up to four. They further demonstrated a $|GHZ_5\rangle$ state, the largest state realized with gate-defined semiconductor spins, and leveraged shuttling and quantum non-demolition (QND) spin measurements to initialize, control, and tune all single- and two-qubit operations. This architecture provides a promising pathway for implementing qLDPC codes and dynamical codes. In a complementary direction, Tamiya et al. [197] proposed a hybrid fault-tolerant architecture combining qLDPC codes and concatenated Steane codes to address the implementation and space–time overheads. The analysis based on *partial circuit reduction*, demonstrated that the proposed construction achieved constant space overhead and polylogarithmic time overhead. This study shows promise for low-overhead FTQC architectures by (i) explicitly accounting for non-zero classical processing times and the associated error accumulation, (ii) employing a hybrid coding strategy that combines non-vanishing-rate qLDPC codes with concatenated codes, and (iii) enabling a modular assessment framework.

[45]It is a two-qubit quantum gate that exchanges the states $|01\rangle$ and $|10\rangle$ while introducing a phase factor i, leaving $|00\rangle$ and $|11\rangle$ unchanged.



(a) Tripartite Tanner graph (b) Periodic two-dimensional [12]

Fig. 12: Bivariate bicycle code representations.

1) Bivariate Bicycle Codes: Bravyi et al. [54], IBM researchers introduced bivariate bicycle (BB) codes, a structured family of qLDPC codes that generalizes earlier bicycle code constructions [198]. BB codes can achieve substantially lower QEC overhead than surface codes at comparable logical performance, provided that the underlying hardware supports the required long-range connectivity. These codes can be viewed as an extension of the two-dimensional toric code, with weight-six stabilizer generators that are not necessarily geometrically local. BB codes are CSS codes defined by parity-check matrices, $H_X = \left[A \mid B\right]$ and $H_Z = \left[B^T \mid A^T\right]$, where the sparse commuting binary matrices A and B are constructed from the x- and y-type cyclic shift operators given by Equation 14.

$$\begin{aligned} x &= S_\ell \otimes I_m & y &= I_\ell \otimes S_m \\ A &= x^{a_1} + y^{a_2} + y^{a_3} & B &= y^{b_1} + x^{b_2} + x^{b_3} \end{aligned} \tag{14}$$

where S_ℓ and I_m denote the $\ell \times \ell$ cyclic shift matrix and the $m \times m$ identity matrix, respectively, and all addition are performed over $\mathbb{F}_2$. Since x and y commute, the resulting matrices A and B also commute, which satisfy the CSS commutativity condition.

Figure 12 illustrates two complementary representations of BB codes. Figure 12(a) shows the tripartite Tanner graph representation, in which data qubits are divided into two equal registers of size $\frac{n}{2}$, corresponding to the two matrix blocks in H_X and H_Z. Figure 12(b) illustrates the two-dimensional representation with periodic boundary conditions, where each stabilizer acts on six data qubits, yielding a weight-six qLDPC code. The x-type and z-type permutations shift the basis vector along x-axis $(\mathrm{mod}\ \ell)$) and y-axis $(\mathrm{mod}\ m)$, respectively. For example, x^3 and y^2 mean three

vertical shifts and two horizontal shifts, respectively[46]. These shifts which stabilizer checks each data qubit. These shifts determine the incidence relationships between data qubits and stabilizer checks. Since A and B each contain three cyclic shift terms, every row of H_X and H_Z has weight six[47]. Various BB code families can be constructed by varying ℓ, m, and permutation degrees (i.e., a_i and b_i). Table IX summarizes BB codes and their finite-size performance. For instance, $H_X = \left[x^3 + y + y^2 \mid y^3 + x + x^2\right]$ for the first code, $[[72, 12, 6]]$, can be represented as Equation 15.

$$H_X = \begin{bmatrix} (S_6 \otimes I_6)^3 + (I_6 \otimes S_6) + (I_6 \otimes S_6)^2 & | \\ (I_6 \otimes S_6)^3 + (S_6 \otimes I_6) + (S_6 \otimes I_6)^2 & | \end{bmatrix} = \begin{bmatrix} (S_6^3 \otimes I_6) + (I_6 \otimes S_6) + (I_6 \otimes S_6^2) & | \\ (I_6 \otimes S_6^3) + (S_6 \otimes I_6) + (S_6^2 \otimes I_6) & | \end{bmatrix} \quad (15)$$

The total number of physical qubits and the hardware encoding density[48] are given by $n = 2\ell m$ and $r = \frac{k}{2n}$, respectively. Two highlighted codes in Table IX exhibit favorable finite-size performance, relatively high encoding rates, and moderate physical qubit requirements, supporting their implementation on available systems. The code highlighted in yellow is known as the *gross* code (i.e., denotes *dozen dozens*), while the code highlighted in purple is refereed to as the *two-gross* code. IBM evaluations demonstrated that BB codes offer promising performance on superconducting qubits. In particular, the gross code can achieve a QEC threshold of approximately 0.7% with shallow, i.e., depth-six, syndrome extraction circuits. A total of 288 physical qubits, consisting of 144 data qubits and 144 check qubits, can encode 12 logical qubits and preserve them for nearly 10^6 QEC cycles, while a comparable logical performance on the surface code would require on the order of $3,000$ physical qubits [54].

TABLE IX: Representative bivariate bicycle codes and finite-size performance

$[[n, k, d]]$	r	(ℓ, m)	A	B	kd^2/n
$[[72, 12, 6]]$	$1/12$	$(6, 6)$	x^3+y+y^2	y^3+x+x^2	6.0
$[[98, 6, 12]]$	$1/32$	$(7, 7)$	$x^3+y^5+y^6$	$y^2+x^3+x^5$	8.82
$[[108, 8, 10]]$	$1/27$	$(9, 6)$	x^3+y+y^2	y^3+x+x^2	7.4
$[[144, 12, 12]]$	$1/24$	$(12, 6)$	x^3+y+y^2	y^3+x+x^2	12.0
$[[162, 8, 14]]$	$1/40$	$(3, 27)$	$1+y^{10}+y^{14}$	$y^{12}+x+x^2$	9.68
$[[288, 12, 18]]$	$1/48$	$(12, 12)$	$x^3+y^2+y^7$	y^3+x+x^2	13.5
$[[180, 8, 16]]$	$1/45$	$(6, 15)$	x^3+y+y^2	$y^6+x^4+x^5$	11.38

From an architectural perspective, IBM [199] has demonstrated that BB codes naturally support modular QEC, providing a promising pathway toward large-scale FTQC. As illustrated in Figure 13, IBM proposed a scalable modular architecture in which each interconnected module comprises a quantum memory based on either the gross code or two-gross code and a *logical processing unit* (LPU). The LPU incorporates additional physical qubits to implement Clifford gates via generalized lattice surgery [139]. This modular architecture

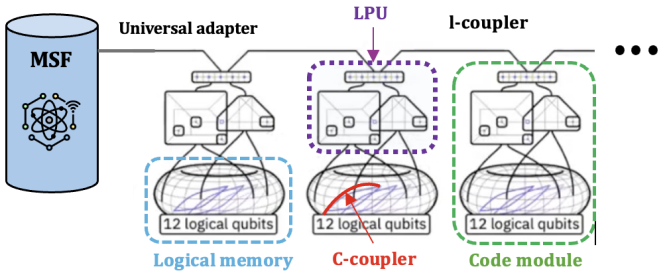


Fig. 13: Modular FTQC based on BB code.

relies on several classes of couplers, i.e., *C-coupler* [153] for on-chip non-local connectivity, the *M-coupler* for chip-to-chip connections, and the *l-coupler* for module-to-module interconnections. Different modules are connected to each other as well as to *magic state factory* (MSF) through *universal adapters* [200] that support *joint measurement* between logical qubits encoded in different quantum memories. These adapters can also support heterogeneous QECC, allowing logical operations between logical qubits encoded with different codes. Such interactions can be realized using inter-module microwave *l-couplers*. Moreover, an MSF is integrated for generation, preparation, and distillation of magic states required for fault-tolerant implementation of non-Clifford gates. Xu et al. [201] proposed a BB-specific MSF design that mitigates the error rate at a smaller qubit footprint while preserving a favorable space-time overhead.

Experimental and theoretical extensions of BB codes further demonstrate their relevance to hardware-aware qLDPC implementations. Wang et al. [202] experimentally implemented distance-4 BB codes and distance-3 punctured BB codes on a 32-qubit superconducting processor using overlapping long-range couplers in a two-dimensional layout. Their experiments enabled simultaneous measurement of non-local weight-6 stabilizer generators and achieved logical error rates of approximately 8.91% and 7.77% per logical qubit per cycle for the BB and punctured BB codes, respectively. Recent extensions further enhanced the capabilities of BB codes. Wang and Mueller [203] experimentally investigated a novel subclass of BB codes, referred to as *coprime-BB* codes, on cold atom qubits using product generating variables. Their construction yielded short- to medium-length codes and reduces both atom movement latency and the number of movements required for syndrome extraction. Voss et al. [204] developed *multivariate bicycle codes*, in particular *trivariate bicycle* (TB) codes, which reduce stabilizer weights from six to five. They demonstrated that this enhancement enables the encoding of four logical qubits with distance five using only 60 physical qubits, compared with 200 physical qubits required by the surface code construction. Shaw and Terhal [205] similarly reduced the connectivity degree of BB codes to five by introducing the generalized parity-check circuit design principle, known as *morphing circuits*. Guo et al. [206] introduced *Z semidirect Z* (ZSZ) codes, which not only exhibited BB-like performance (i.e., threshold around 0.5%) under the circuit-level depolarizing noise model, but also can be realized with neutral-atoms trapped in movable tweezer arrays. Therefore, the entire syndrome extraction cycle can be implemented

[46] $x : (i, j) \mapsto (i + 1 \bmod \ell,\ j), \quad y : (i, j) \mapsto (i,\ j + 1 \bmod m)$.

[47] The term *bivariate* refers to the two commuting variables x and y, while the *bicycle* reflects the cyclic-shift structure inherited from classical bicycle code constructions.

[48] Hardware encoding density rate accounts for both n data qubits and n check qubits, required to encode k logical qubits.

efficiently through simple global atomic array motions. Moreover, the proposed codes demonstrated substantially improved sustainable thresholds of approximately 0.095%, compared with roughly 0.06% for the four-dimensional toric codes.

Several studies have also generalized or adapted the BB code structure. Liang et al. [207] leveraged the twisting structure of BB codes, formulated via a ring-theoretic perspective, to generalize toric code constructions to *twisted tori* with larger logical dimensions. They developed a family of weight-six generalized toric codes, including $[[254, 14, 16]]$ and $[[294, 10, 20]]$ codes, which achieved improved finite-size performance values of 14.11 and 13.61, respectively. These values exceed the 13.5 finite-size performance of the two-gross code (i.e., the best reported BB code listed in Table IX). Mourad Halla [208] generalized this construction to qudit systems, enabling its application to higher-dimensional quantum codes. Jacoby et al. [209] introduced *Stairway codes* by applying dynamic Floquet-style measurement scheduling, as discussed later in this paper, to BB code structures, where each stabilizer generator is decomposed into a periodic sequence of pairwise measurements. These codes achieveed performance comparable to semi-hyperbolic Floquet codes using fewer than 300 physical qubits, in contrast to more than $1,300$ physical qubits required by the corresponding semi-hyperbolic constructions. Chandra et al. [210] proposed a distributed implementation of BB codes across multi-QPU architectures. They partitioned the gross code among 4, 6, and 12 QPUs, each assumed to support all-to-all internal connectivity, and evaluated the resulting logical error rates using Monte Carlo simulations with a BP+OSD decoding algorithm. Finally, Bhave et al. [211] introduced *BiBiEQ*, an enhanced variant of BB codes specifically designed to improve resilience against erasure errors.

C. Subsystem Codes

Although stabilizer codes have been experimentally validated across multiple qubit modalities, their syndrome extraction circuits may involve high-weight and asymmetric stabilizer measurements, which increase circuit depth and susceptibility to error propagation. For example, in Shor's code, the X-type stabilizer generators have weight six, while the Z-type stabilizer generators can be generated by weight-two parity checks [165]. Subsystem codes provide an alternative QEC construction in which high-weight stabilizer generators can be inferred from the measurement of *gauge operators*. This reduces syndrome extraction complexity and makes subsystem codes promising candidates for FTQC, particularly in architectures where direct high-weight measurements are costly. Their gauge structure also provides flexible gauge-operator measurements, which can be tailored to improve code performance under biased noise. For instance, XX-type check measurements can be performed more frequently when phase-flip errors dominate [212].

The theoretical foundation underlying subsystem codes is provided by *operator quantum error correction* (OQEC) [213]. In OQEC, quantum information is encoded into a protected subsystem A of a codespace $C = A \otimes B$, where B is an auxiliary *gauge subsystem* that may evolve under noise without affecting the logical information stored in subsystem A. In general, stabilizer codes, also known as *subspace codes*, correspond to the special case of subsystem codes in which the gauge subsystem B is one-dimensional [37]. Therefore, errors need only be corrected modulo transformations acting on B, since changes in B do not affect the protected subsystem A. This strategy unifies active quantum error correction protocols with passive error-avoidance schemes[49], such as *decoherence-free subspaces* (DFS) [33] and *noiseless subsystems*. In decoherence-free subspaces, the relevant noise acts trivially on the protected subspace, whereas in noiseless subsystems, noise may act only on the gauge subsystem while leaving the encoded logical information unaffected [55].

Subsystem codes constitute a stabilizer-based realization of OQEC codes through a generally non-Abelian gauge group $\mathcal{G}$. The stabilizer group is given by the center of the gauge group, $\mathcal{S} = Z(\mathcal{G})$, up to phases[50]. These codes decompose the Hilbert space as $\mathcal{H} = (\mathcal{C}_L \otimes \mathcal{C}_G) \oplus C^{\perp}$, where $\mathcal{C}_L$ is the protected logical subsystem that stores the encoded quantum information, $\mathcal{C}_G$ is the gauge subsystem, and $C^{\perp}$ denotes the orthogonal complement outside the codespace $C = \mathcal{C}_L \otimes \mathcal{C}_G$. A subsystem code is characterized by parameters $[[n, k, g, d]]$, where g denotes the number of *gauge qubits* and d is the minimum weight of a nontrivial logical operator in $\mathcal{N}(\mathcal{S}) \backslash \mathcal{G}$ [214]. A subsystem code with n physical qubits and r independent stabilizer generators satisfies the dimension relation given in Equation 16.

$$n - r = k + g \tag{16}$$

Gauge qubits do not store logical information and therefore do not require active protection. This gauge freedom allows high-weight stabilizer generators to be decomposed into products of lower-weight gauge operators, which reduces depth of syndrome extraction circuits and mitigates error propagation during stabilizer measurement, both of which are important requirements for FTQC.

A canonical example of subsystem codes is the *Bacon codes*, constructed by placing physical qubits on the vertices of an $L \times M$ rectangular lattice to encode one logical qubit. Let $O_{i,j}$, with $O \in \{X, Z\}$, denote a Pauli operator acting on the qubit located at position (i, j) on the lattice, where $i \in \{0, 1, \ldots, M-1\}$ and $j \in \{0, 1, \ldots, L-1\}$. The gauge operators correspond to weight-two nearest-neighbor Pauli checks, i.e., horizontal XX operators and vertical ZZ operators[51]. There are $M-1$ independent X-type stabilizer generators and $L-1$ independent Z-type stabilizer generators,

[49] Passive schemes protect quantum information by encoding it in subspaces or subsystems that are naturally immune to certain noise processes, rather than correcting errors after they occur.

[50] The center of a group is the set of operators that commute with every element of that group.

[51] The XX gauge checks act on neighboring qubits in adjacent rows, $X_{i,j}X_{i+1,j}$, while the ZZ gauge checks act on neighboring qubits in adjacent columns, $Z_{i,j}Z_{i,j+1}$.

which can be written as product of these gauge operators, given in Equation 17.

$$X_{i,*}X_{i+1,*} = \bigotimes_{j=0}^{L-1} X_{i,j}X_{i+1,j}, \quad \forall i \in \{0,1,\ldots,M-2\},$$
$$Z_{*,j}Z_{*,j+1} = \bigotimes_{i=0}^{M-1} Z_{i,j}Z_{i,j+1}, \quad \forall j \in \{0,1,\ldots,L-2\}. \tag{17}$$

Each stabilizer generator is therefore obtained as the product of weight-two gauge operators. Consequently, the eigenvalues of stabilizer generators can be inferred from the measurement outcomes of low-weight gauge operators. The stabilizer non-abelian group of the Bacon code can be written as Equation 18.

$$\mathcal{S} = \langle X_{i,*}X_{i+1,*},\, Z_{*,j}Z_{*,j+1}\rangle. \tag{18}$$

The code has $r = (L-1)+(M-1)$ independent stabilizer generators and encodes one logical qubit. Therefore, the number of gauge qubits is $(L-1)(M-1)$, which corresponds to the number of square plaquettes in the lattice[52]. Therefore, a subsystem code defined on an $L \times M$ lattice has parameters $[[LM, 1, (L-1)(M-1), \min(L,M)]]$, where the distance is determined by the minimum length of logical operators acting along rows or columns of the lattice. Moreover, Bacon and Casaccino [215] further generalized the construction of subsystem codes using pairs of classical linear codes.

Subsystem variants of well-known stabilizer codes have been developed to facilitate their practical implementation on near-term quantum hardware. The Bacon–Shor code [213] is the subsystem version of subspace Shor's code, where $L = M = 3$. The Bacon–Shor code encodes one logical qubit in nine physical qubits and has parameters $[[9,1,4,3]]$. Its four gauge degrees of freedom allow the high-weight stabilizer generators to be inferred from weight-two gauge measurements. The Bacon–Shor code has been experimentally demonstrated on a 13-qubit trapped-ion QPU, achieving logical SPAM errors of approximately 0.6% and logical Clifford gate errors of about 0.3% after offline error correction [216]. Bombín [212] introduced generalized subsystem color codes, which extend conventional subspace color codes by incorporating weight-two gauge degrees of freedom. In three dimensions, such codes can support transversal implementation of a universal logical gate set.

Bravyi et al. [217] introduced subsystem variants of toric and surface codes in which weight-four stabilizer measurements are replaced by weight-three triangle gauge measurements. In the subsystem toric code, physical qubits are placed on vertices and center of edges of an $L \times L$ lattice with periodic boundary conditions. This construction requires $3L^2$ physical qubits and introduces L^2 gauge qubits[53]. The stabilizer generators are products of triangle gauge operators, allowing syndrome information to be extracted through local weight-three gauge operator measurements. The subsystem surface code corresponds to the planar version of this construction with open boundary conditions and additional boundary stabilizer generators that appear along the lattice edges (e.g., weight-two X- and Z-type stabilizer generators on smooth and rough boundaries, respectively). This modification increases the number of physical qubits to $3L^2 + 4L + 1$ and the number of independent stabilizer generators to $2L^2 + 4L$[54]. Their evaluations demonstrated that under the circuit-level noise model, the subsystem surface code achieves a threshold of approximately 0.6%, compared with roughly 0.9% for the subspace toric code [218]). When three-qubit parity measurements, such as XXX and ZZZ, are supported by the hardware, the threshold can increase to approximately 0.97%, approaching the threshold of the subspace surface code.

A distinctive advantage of subsystem codes is their support for *gauge fixing* to manipulate encoded quantum information and improve the performance of subsystem codes. Gauge fixing was introduced by Paetznick and Reichardt [73] to switch between different codes during computation to implement universal fault-tolerant gate set. Gauge fixing refers to selecting a gauge operator g from the gauge group, $g \in \mathcal{G}$, promoting it to a stabilizer generator by adding it to the stabilizer group $\mathcal{S}$, and updating the gauge generating set by removing operators that anticommute with g. More recently, Higgott and Breuckmann [219] exploited schedule-induced gauge fixing to improve the error correction performance of subsystem codes. By modifying the measurement schedule, their method extracts additional syndrome information and facilitates decoding process. According to the simulation results, they increased the threshold of the subsystem toric code from 0.67% to 0.81% under the circuit-level depolarizing noise model and up to 2.22% under a biased circuit-level noise model. They further constructed finite-rate subsystem LDPC code families with weight-three check operators, achieving overhead reductions of approximately $4.3\times$ relative to the most efficient surface code and $5.1\times$ relative to the subsystem toric code at comparable error rates. These results highlight the flexibility of gauge fixing in tailoring subsystem codes to biased noise models. This capability is particularly important for FTQC, as different qubit modalities exhibit distinct noise biases, as shown in Table VI.

D. Dynamical Codes

Kitaev's honeycomb model demonstrated an early example in which the eigenvalues of weight-six stabilizer generators can be inferred from sequences of weight-two check measurements [220], which can be interpreted as a subsystem formulation of the toric code on a hexagonal lattice. In this construction, as illustrated in Figure 14, physical qubits are placed on the vertices of a hexagonal (i.e., honeycomb) lattice with periodic boundary conditions. Each edge corresponds to a two-body gauge operator, including XX (i.e., horizontal edges), YY (i.e., negative slope edges), or ZZ (i.e., positive slope edges), represented by the red, blue, and gray edges in the figure, respectively. The eigenvalues of the effective plaquette stabilizer generators are inferred from the measurement

[52]The number of gauge qubits equals $g = n - r - k$, which equals to $LM - (L-1) - (M-1) - 1 = (L-1)(M-1)$.

[53]The subsystem toric code has parameters $[[3L^2, 2, L^2, L]]$.

[54]The subsystem surface code has parameters $[[3L^2 + 4L + 1, 1, L^2, L]]$.

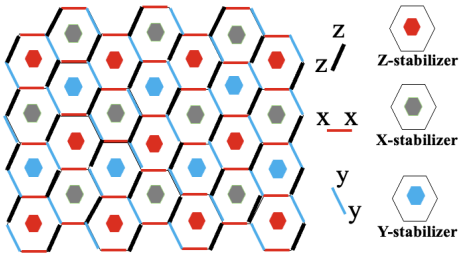


Fig. 14: Honeycomb structure for subsystem codes.

sequence of red, blue, and gray gauge operators, rather than by directly measuring weight-six operators.

A honeycomb lattice with n vertices contains $F = \frac{n}{2}$ stabilizer generators, associated with hexagonal plaquettes and $E = \frac{3n}{2}$ two-body gauge operators, associated with edges[55]. Moreover, the number of independent stabilizer generators is $r = \frac{n}{2}+1$, consisting of $\frac{n}{2}-1$ independent plaquette stabilizer generators and two additional stabilizer generators associated with nontrivial cycles of the torus [221]. Furthermore, the number of gauge qubits is $g = \frac{n}{2}-1$[56]. Substituting these values into the subsystem code dimension relation in Equation 16 yields $k = 0$, indicating that there is *no space* for logical qubits in the static description. Subsequent studies revealed that logical degrees of freedom (i.e., qubits) can emerge through a time-dependent measurement scheduling, motivating the concept of dynamically generated logical qubits.

Hastings and Haah [29] demonstrated that logical qubits do exist in the honeycomb subsystem codes, although they are hidden in the subsystem description. They formalized this idea by introducing a time-dependent measurement scheduling that generates logical qubits dynamically, leading to the class of *Floquet codes*. In this structure, the hexagonal lattice is three-colorable, with plaquettes labeled red (R), gray (G), and blue (B), as depicted in Figure 14. They proposed a three-round measurement schedule for measuring two-body gauge operators of the non-Abelian gauge group. In the first round, gauge operators connecting two red plaquettes are measured (e.g., BX, BY, BZ, where the first letter denotes the plaquette color and the second indicates the type of gauge operator). In the second round, gauge operators connecting two blue plaquettes are measured (e.g., GX, GY, GZ). In the third round, gauge operators connecting two gray plaquettes are measured (e.g., RX, RY, RZ)[57]. This measurement scheduling prevents measuring the homologically nontrivial cycles and therefore preserves two logical qubits on the torus. The evolution of the system can be described by the *instantaneous stabilizer group* (ISG), which denotes the group of operators that stabilize a quantum state after a given measurement round. As the measurement schedule proceeds, the ISG evolves dynamically, causing the code transition between related toric-like stabilizer configurations (e.g., between the embedded toric code and superlattice toric code). Hastings and Haah also showed that the Floquet code is fault tolerant because (i) eigenvalues of stabilizer generators can be inferred from a sequence of gauge measurements, (ii) single-qubit errors generate pairs of detection events, and (iii) the detection events can be decoded using matching or union-find techniques, yielding a finite error correction threshold.

Gidney et al. [222], Google researchers, experimentally evaluated the robustness of Floquet code logical qubits and benchmarked it against the rotated surface codes. Their results indicated that the QEC threshold of the Floquet code lies between 0.2% and 0.3%, whereas the rotated surface code achieves a higher threshold in the range of 0.5%–0.7% in the controlled-not circuit model. They also projected that Floquet codes could reach the *TeraQuOp footprint*[58] using only about 600 physical qubits. Although the Floquet code threshold is lower than that of optimized surface code layouts, Floquet codes remain promising because their low-weight measurements simplify syndrome extraction circuits and exploit dynamically evolving stabilizer generators. Sutcliffe et al. [223] introduced distributed hyperbolic Floquet codes based on pairwise gauge measurements, such that each nonlocal measurement requires only a single distributed Bell pair. The authors demonstrated that under the circuit-level noise model, local and nonlocal operation fidelities of approximately 99.97% and 99%, respectively, which makes this code family a promising candidate for distributed FTQC.

Dynamical code ideas have also been applied to subsystem codes. Alam and Rieffel [224] introduced *Floquet–Bacon–Shor codes* to generate dynamical logical qubits by developing a measurement scheduling for the Bacon–Shor code. A four-round measurement schedule involves both the stabilizer generators of the Bacon–Shor code and gauge operators, such that dynamical logical qubits emerge through the evolution between ISGs. The code not only saturates the two-dimensional subsystem code bound, $kd = O(n)$, which is not achieved by the conventional Bacon–Shor code, but also scales the code distance as $\Theta(L/\sqrt{k})$ on an $n = L \times L$ lattice with k dynamical logical qubits, while preserving a logical qubit of the Bacon–Shor code. Although the Floquet–Bacon–Shor code exhibits self-correcting behavior against certain error patterns, it requires more complex decoding algorithms and, in contrast to the conventional Bacon–Shor code, it does not exhibit a conventional QEC threshold[59].

Dynamic measurement scheduling has also been experimentally explored in surface code variants. Eickbusch et al. [226], Google researchers, experimentally demonstrated time-dynamic measurement circuits for three variants of the surface code, i.e., hexagonal lattice surface codes, walking surface codes, and *iSWAP*-based surface codes. The hexagonal

[55]Each vertex has degree three, so $3V = 2E$, each hexagonal plaquette has six edges, and each edge is shared by two plaquettes, so $6F = 2E$.

[56]The number of gauge qubits should not be confused with the number of independent gauge generators. For a subsystem code, the gauge group rank is $r + 2g$. Here, $r = \frac{n}{2} + 1$ and $g = \frac{n}{2} - 1$, giving $r + 2g = \frac{3n}{2} - 1$ independent gauge generators.

[57]Note that the mentioned measurement schedule is not unique; alternative cyclic measurement patterns can also be defined.

[58]A TeraQuOp footprint estimates the number of physical qubits required to reliably execute approximately 10^{12} logical quantum operations before failure.

[59]Note that Gidney and Bacon [225] discussed how removing certain gates from the circuit and skipping measurements of selected gauge operators can provide insights about the threshold of the Bacon–Shor code.

lattice construction reduces the qubit connectivity requirement from four couplings per qubit to three. The walking surface code periodically swaps the roles of data and measurement qubits at each round to eliminate accumulated non-computational errors, such as leakage. The *iSWAP*-based implementation replaced *CNOT* gates with *iSWAP* gates to minimize gate overhead. Their experimental results indicated that the error-suppression factors of these three structures were $\Lambda_{35,\text{hex}} = 2.15(2)$[60], $\Lambda_{35,\text{walk}} = 1.69(6)$, and $\Lambda_{35,\text{iSWAP}} = 1.56(2)$. These results demonstrate logical error suppression across dynamic surface code variants and highlight the potential of dynamic circuit implementations of the surface code for FTQC.

The implementation of dynamical codes requires hardware platforms capable of flexible spatial layouts, programmable, and parallel operations between desired qubits. Neutral-atom platforms are particularly relevant in this context because optical tweezers can dynamically rearrange atoms and support non-local connectivity. Bluvstein et al. [172] developed a neutral-atom-based QPU that provides a dynamic topology and non-local connectivity (i.e., based on optical tweezers) between distant qubits. By overcoming connectivity limitations, this architecture provides a promising platform for distributed DQECCs, dynamical codes and subspace codes.

Overall, the code families discussed in this section reveal that no single QECC construction is universally optimal or capable of satisfying all FTQC requirements. Each quantum code family offers distinct advantages and trade-offs in terms of overhead, threshold, decoding complexity, logical gate support, measurement weight, and hardware compatibility. For instance, BB codes are promising for low overhead quantum memories, surface codes remain among the most practical candidates for local two-dimensional implementations, and color codes provide a transversal implementation of the full Clifford group. Subsystem codes reduce syndrome extraction complexity through gauge degrees of freedom, while dynamical and Floquet codes introduce time-dependent stabilizer structures that can reduce measurement weight, adapt to hardware connectivity, and exploit flexible measurement scheduling. These complementary characteristics motivate hybrid quantum code architectures that combine multiple code families, code transformations, and hardware-aware implementations within a unified FTQC framework, thereby enabling greater flexibility and improved resource optimization on the path toward large-scale fault-tolerant quantum computing.

VI. Decoding Algorithms for QEC

Decoding algorithms constitute a fundamental component of QECCs as they infer likely error configurations from syndrome information and determine appropriate recovery operations to preserve quantum information. Therefore, the effectiveness of FTQC depends not only on the QECC parameters, thresholds, and overheads, but also on the accuracy, computational efficiency, latency, and scalability of the associated decoding algorithms. Formally, a decoding problem can be defined by a parity-check matrix $H \in \mathbb{F}_2^{M\times N}$, an action matrix $A \in \mathbb{F}_2^{K\times N}$,

[60] $\Lambda_{35,\text{hex}} = 2.15(2)$ denotes $\Lambda_{35,\text{hex}} = 2.15 \pm 0.02$.

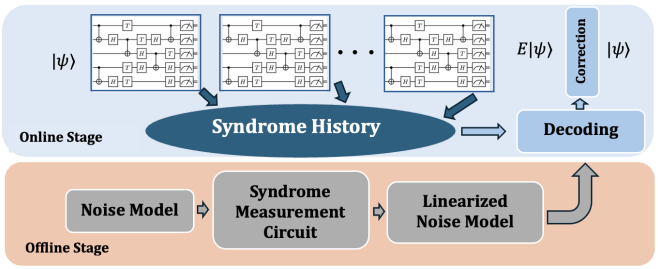


Fig. 15: Overview of the QEC decoding workflow.

and an error probability vector $p = (p_0, \ldots, p_{N-1})$ (i.e., describing the physical error probabilities). Let $e \in \mathbb{F}_2^N$ denote an unknown error vector and $\sigma = He \in \mathbb{F}_2^M$ represents the observed syndrome. The decoder aims to infer a recovery operator $\hat{e}$ satisfying $H\hat{e} = \sigma$ and $A\hat{e} = Ae$, ensuring both syndrome consistency and logical equivalence between the estimated error and actual error. Equivalently, the residual error $e + \hat{e}$ should act trivially on the protected logical information. In practical systems, syndromes are typically represented as *detection events*[61] and are mapped to a decoding graph or hypergraph, where vertices correspond to detection events (i.e., syndrome defects) and edges or hyperedges represent possible error mechanisms, including data errors, measurement errors, leakage, and correlated errors. The resulting graph is subsequently processed by classical processing either real-time during circuit execution, referred to as *online decoding*, or after execution, referred to as *offline decoding*[62] [227].

As illustrated in Figure 15, in the operational workflow of FTQC, a quantum processor continuously performs syndrome extraction cycles and generates a stream of syndromes, which are accumulated as the syndrome history and are passed to a classical decoding pipeline. On the classical side, the decoding process begins during the offline stage with the construction of a noise model that characterizes the physical error models affecting the quantum processor. The syndrome measurement circuit is then analyzed to determine how physical errors propagate through the circuit and manifest as observable syndrome patterns [228]. Subsequently, a linearized noise model maps individual error events to their corresponding detection events and evaluates their impact on the logical error rate (i.e., this is accomplished for both bit-flip and phase-flip errors). This information is then provided to the decoding algorithm, which compares the observed detection events with possible error configurations to infer an appropriate recovery operation or Pauli frame update. Overall, the offline stage transforms physical noise models into error syndromes, while the decoder maps syndromes to the most likely error configuration in order to determine an appropriate reaction [229].

Although offline decoders are sufficient for many Clifford operations and benchmarking experiments, online decoding is

[61] A detection event corresponds to an unexpected parity outcome associated with a *detector*, where a detector typically refers to a pair of consecutive stabilizer measurements whose parity should remain unchanged in the absence of errors.

[62] In offline decoding, stabilizer measurement outcomes are collected and stored during circuit execution, then processed to infer the error configuration and determine the corresponding recovery or Pauli frame update.

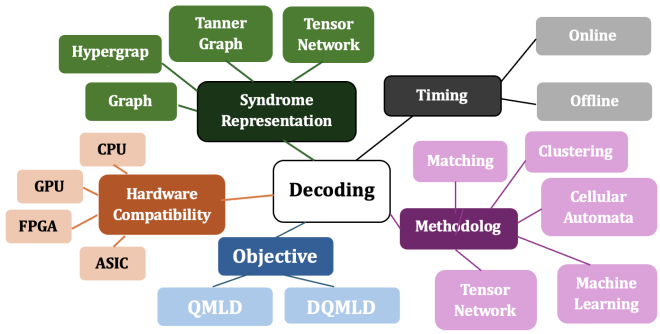


Fig. 16: Taxonomy of QEC decoding algorithms.

required whenever recovery operation affects subsequent quantum operations[63]. The main challenge is the finite throughput of classical processing, which introduces latency for online decoding. To prevent a growing syndrome backlog, the decoder must process the syndrome stream at a rate comparable to the syndrome generation rate. Let r_{gen} and r_{proc} denote syndrome generation and decoder processing rates (i.e., in bauds), respectively. If $f = \frac{r_{\text{gen}}}{r_{\text{proc}}} > 1$, the syndrome information accumulates faster than it can be decoded, causing processing backlogs and increasing feedback latency. Simulations have confirmed that, as quantum systems scale, decoders increasingly struggle to satisfy the required low-latency constraints [230]. This constraint is particularly stringent in superconducting platforms, where QEC cycle times can approach the microsecond scale. Therefore, practical real-time decoders for FTQC should satisfy four key requirements.

- *High accuracy:* to achieve sufficiently low logical error rates required by large-scale logical computations.
- *Low latency:* to prevent syndrome processing backlogs and support time sensitive feedforward operations.
- *Flexibility:* to supporting different code distances, noise models, and QECC families without requiring substantial modifications of the decoding architecture.
- *Hardware efficiency:* to enable FPGA- or ASIC-friendly implementation with a modest resource footprint[64].

General acceleration strategies, including sliding-window [232], parallelization [233], and modular decoding [234], [235], can reduce the exponential growth of decoding overhead to polynomial scaling as the size of the QEC code increases. However, the quantum decoding problem belongs to the $\#P$-complete complexity class because of code degeneracy[65], which makes it computationally more challenging than decoding of classical linear codes. Moreover, the diversity of noise models across different quantum computational paradigms and qubit modalities, retaining decoder latency below the syndrome generation timescale, classical–quantum orchestration overhead make quantum decoding highly challenging for large-scale FTQC.

Decoding algorithms of stabilizer codes are generally classified into two classes, i.e., *quantum maximum likelihood decoding* (QMLD) and *degenerate quantum maximum likelihood decoding* (DQMLD). QMLD, often called *non-degenerate decoding*, ignores the code degeneracy and aims to identify the most probable physical error configuration consistent with the syndrome stream[66]. In contrast, DQMLD, also referred to as *degenerate decoding*, explicitly accounts for code degeneracy by identifying the most probable equivalence class (i.e., coset) of errors that produce the same syndrome and induce the same logical effect on the encoded quantum state[67]. Although DQMLD provides the optimal decoding rule for stabilizer codes, it is generally *#P-complete* and therefore computationally intractable for large-scale instances. Therefore, many practical decoding algorithms rely on QMLD, which, although *NP-complete*, is often more amenable to efficient approximations and hardware-aware implementations [229].

Decoding of CSS codes can be decomposed into two independent decoding subproblems corresponding to bit- and phase-flip error correction. Let E denote the physical error acting on the encoded state, and R_X and R_Z denote the recovery operations produced by the corresponding X- and Z-decoders, respectively. The residual error is given by $\Gamma = R_X R_Z E$. The decoding algorithm succeeds when the residual error acts trivially on the codespace, i.e., $\Gamma \in \langle \mathcal{S} \rangle$, or equivalently the logical state is the $+1$ eigenstate of the residual error, $\Gamma|\psi\rangle_L = (+1)|\psi\rangle_L$. More generally, decoding succeeds when the recovery operation differs from the actual error only by a stabilizer element, thereby preserving the logical information. In contrast, decoding fails when Γ corresponds to a nontrivial logical operator, i.e., $\Gamma \in \mathcal{N}(\mathcal{S}) \setminus \mathcal{S}$, where $\mathcal{N}(\mathcal{S})$ denotes the normalizer, discussed in section II. In this case, the recovery procedure introduces a logical error on the encoded state.

Figure 16 illustrates a taxonomy of decoding algorithms organized in terms of syndrome representation, decoding methodology, execution timing, hardware compatibility, and decoding objectives. From the methodological perspective, decoding algorithms can be broadly classified into several categories, including matching-based decoders, clustering and graph-growth decoders, probabilistic inference decoders, tensor network decoders, machine learning decoders, cellular automata decoders, and optimization-based decoders. Each category exhibits distinct trade-offs among decoding accuracy, computational complexity, scalability, latency, and hardware compatibility, making it suitable for different code families, noise models, and implementation constraints.

Matching-based decoders: Matching-based decoders, such as *Minimum-Weight Perfect Matching* (MWPM), transform the decoding problem into a weighted graph matching[68] opti-

[63]For example, in the magic state injection-based implementation of the T gate, a real-time decoder may be required to determine whether a corrective Clifford operation must be applied before subsequent operations.

[64]Fast and real-time decoding often requires implementation on a field-programmable gate array (FPGA) or even an application-specific integrated circuit (ASIC) [231], rather than relying solely on general-purpose processors.

[65]Multiple distinct physical error configurations may produce the same error syndromes and have equivalent logical effects.

[66]Mathematically, QMLD, often called *maximum likelihood decoding* (MLD), formulated as $\hat{E} = \arg\max_{E \in \mathcal{P}_n} P(E \mid \bar{s})$, where $\bar{s}$ denotes the measured syndrome.

[67]Mathematically, DQMLD can be formulated as $\hat{L} = \arg\max_{L \in \mathcal{L}} P(L \mid \bar{s})$, where $\mathcal{L}$ denotes the set of logical error cosets associated with the stabilizer code.

[68]A matching of a graph is a subset of edges such that no two edges share a common vertex.

mization problem defined over detection events or syndrome defects. In these approaches, the decoding problem is formulated as a minimum weight perfect matching[69] problem that identifies the most likely (e.g., the lowest weight) configuration of error chains consistent with the detection events, which can be efficiently solved using *Blossom-based* algorithms, including fusion blossom [236] and sparse blossom [237]. Matching-based decoders are particularly effective for topological QECCs, such as surface codes, where local Pauli errors typically generate pairwise syndrome defect structures.

Graph-growth decoders: Graph-growth decoders, such as *union-find* (UF) decoders [238], formulate the decoding problem as a graph expansion and cluster merging procedure defined over detection events (i.e., often called *cluster decoders*). In these approaches, odd clusters[70] are iteratively expanded and merged until they either become even clusters or reach the code boundaries. Subsequently, a peeling algorithm is applied within each cluster to construct recovery paths (i.e., error chains) that neutralize the generated defects. Compared with MWPM, the UF decoder achieves significantly lower computational overhead and near-linear time complexity, although its accuracy is generally lower than that of MWPM. Some UF variants can match MWPM performance under certain conditions and noise models [239], however, UF is typically regarded as an approximate alternative to MWPM decoders and promising for large-scale FTQC.

Probabilistic inference decoders: Probabilistic inference decoders, such as *belief propagation* (BP), provide general decoding algorithms for sparse QECCs, particularly where the Pauli errors do not naturally generate pairwise syndrome defects, as in many qLDPC codes. These decoders formulate the decoding task as an iterative probabilistic inference problem over the Tanner graph of the underlying code [240]. In these approaches, probabilistic messages are exchanged iteratively between neighboring variable and check nodes to estimate the marginal error probabilities of individual qubits conditioned on the measured syndrome information; for this reason, BP is often referred to as a *message-passing* (MP) decoding algorithm. These decoders are scalable, flexible across different noise models, and have relatively low computational complexity. However, the performance of BP decoders can degrade in the presence of small girth[71], code degeneracy, absorbing sets[72], and trapping sets[73]. Consequently, BP is often combined with post-processing methods such as *ordered-statistics decoding* (OSD), which compensates for residual decoding inconsistencies produced by iterative message-passing inference [241].

Tensor network decoders: Tensor network decoders, such as DQMLD decoders, formulate decoding problem as a tensor network contraction problem defined by the graphical structure of the underlying code. In these approaches, stabilizer constraints, syndrome stream, and error probabilities are represented as interconnected tensors[74], commonly using *matrix product state* (MPS) representations [242]. The decoding task is then formulated as a tensor network contraction problem, where the contracted network approximates the most probable logical recovery operation conditioned on the observed stabilizer measurements. Tensor network decoders can capture long-range correlations and code degeneracy more effectively than local message-passing approaches, enabling near-optimal performance for several topological QECCs, including two- and three-dimensional surface codes [243]. However, the computational complexity of tensor network contraction generally increases rapidly with code distance and network connectivity, motivating the development of approximate contraction techniques and truncated tensor representations for scalable FTQC.

Machine learning decoders: Machine learning decoders formulate decoding problem as a data-driven inference task, in which a parameterized learning model is trained to map syndrome stream or syndrome histories to recovery operations or Pauli frame updates. Various machine learning architectures, including neural networks [244], convolutional neural networks (CNN) [245], recurrent neural networks, reinforcement learning [246], graph neural networks (GNN), and transformer-based models, have been explored for QEC decoding. These approaches can learn complex error correlations, leakage and crosstalk signatures, analog readout features, and hardware-specific noise patterns from training datasets[75]. However, their performance depends on the quality and diversity of the training set, and training cost may increase significantly with code distance and syndrome dimensionality. Despite these challenges, their adaptability and potential for hardware acceleration make machine learning decoders promising for scalable FTQC, particularly under realistic and correlated noise models. For instance, Google developed a machine learning decoder called *AlphaQubit* [244] for surface codes, which outperforms previous tensor network and correlated matching decoders for codes distances up to 11. However, AlphaQubit is not yet fast enough for superconducting real-time decoding.

Cellular automata decoders: Cellular automata (CA) decoders formulate the decoding problem as a local iterative update process over the lattice structure of the underlying code. In these approaches, local update rules are repeatedly applied to syndrome defects or auxiliary classical variables to propagate, annihilate, or neutralize error syndromes over successive decoding iterations [247]. Unlike global optimization-based decoding algorithms, CA decoders rely exclusively on local classical processing and nearest-neighbor communica-

[69] A perfect matching is a matching in which every vertex is incident to exactly one selected edge (i.e., pairs all relevant defects).

[70] A cluster refers to a connected set of detection events together with the associated edges in the decoding graph.

[71] The girth of a graph is the length of its shortest cycle, in which small girth indicates the presence of short loops.

[72] An absorbing set is a subgraph structure in iterative decoding that can trap the decoder in an incorrect fixed point, leading to persistent decoding failures.

[73] Trapping sets are small subgraphs of the Tanner graph whose local message correlations can cause iterative message-passing decoders to converge to incorrect decoding states, thereby contributing significantly to the decoding error floor.

[74] A tensor network is a collection of tensors connected through shared indices, where tensor contractions encode correlations and probabilistic relationships among variables.

[75] Training datasets typically consist of syndromes paired with corresponding physical error, recovery operation, or logical error labels generated by Monte Carlo simulation or experimental calibration.

tion, making them promising for hardware-efficient and fully parallel FTQC architectures. They are particularly well-suited to topological QECCs, whose geometric locality naturally aligns with local update operations. Although their accuracy is generally lower than near-optimal decoding algorithms, such as MWPM, CA decoders offer low computational overhead, local update rules, and strong scalability, making them well suited for large-scale real-time decoding algorithms [248].

Table X provides a qualitative comparison of the decoder families, discussed above, in terms of decoding algorithms features. These criteria reflect the key requirements of real-time QEC decoding in supporting different codes and noise models, achieving low logical error rates, minimizing resource footprint, and processing syndrome stream with sufficiently low latency. The ratings are context-dependent because each decoder family can behave differently depending on the underlying QECC, noise model, and implementation platform. Here, compactness refers to the suitability of a decoder for resource-efficient FPGA- or ASIC-based implementations, and search-based approaches are discussed later in this paper.

TABLE X: Comparison of major decoder families and implementation trade-offs

Decoder Family	Flexibility	Accuracy	Compactness	Speed
Matching	○	✓	×	○
Graph-growth	○	○	✓	✓
Message Passing	✓	○	✓	✓
MP + OSD	✓	✓	×	×
Tensor Network	○	✓	×	×
Cellular Automata	○	○	✓	✓
Machine Learning	✓	○	×	○
Search-based	✓	✓	×	×

✓: strong; ×: weak/unsuitable; ○: moderate or context-dependent.

A. Decoding of Topological Codes

In this subsection, we review decoding approaches for topological QECCs, including surface, toric, and color codes, since these families are experimentally mature codes for FTQC. We emphasize how geometric structure of these codes shapes the choice of decoding algorithm.

1) Decoding of Surface/Toric Codes: The grid-like nearest-neighbor structure of surface codes causes many Pauli errors to generate either zero or two nearby detection events in the corresponding space-time syndrome graph. This pairwise defect structure makes MWPM, implemented through solvers such as *sparse blossom* or *PyMatching*, a default decoding algorithm for such codes. Google employed a real-time sparse blossom decoder with greedy edge reweighting techniques to validate surface code operation on the *Willow* processor, achieving an average decoder latency of 63 μs for distance-5 codes over up to one million QEC cycles, with a cycle time of 1.1 μs. For larger codes, Google utilized two high-accuracy offline decoding approaches, i.e., a neural network decoder [244] and a harmonized ensemble [249] of correlated MWPM decoders augmented with matching synthesis [250], both executed on separate classical hardware platforms. Shutty et al. [251] further demonstrated that harmonized ensembles of MWPM decoders can outperform their individual constituent decoders on repetition code and surface code benchmarks in terms of logical error rates, which highlights decoder harmonization as a promising solution toward highly accurate near-optimal decoding.

Union-find and clustering decoders have also been developed for topological codes to reduce decoding latency while preserving scalable graph-based structure. Huang et al. [252] introduced a weighted union-find decoder for the toric code under the circuit-level depolarizing noise model. Their simulation results demonstrated that incorporating edge weighting improves the decoding threshold from 0.38% to 0.62%, while preserving the near-linear time complexity of the original union-find decoding algorithm. Liyanage et al. [253] proposed an FPGA-based distributed union-find decoder for surface codes, called *Helios*[76]. Their design exploited massive parallelism to achieve sublinear average decoding complexity with respect to the code distance using $O(d^3)$ parallel resources. Their implementation on a *Xilinx VCU129* FPGA achieved ultra-low decoding latency as low as 11.5 ns per measurement round under phenomenological noise, demonstrating a fast decoder while maintaining scalability to larger code distances. Chan et al. [254] extended the distributed union-find decoding algorithm on the Helios platform and developed *Actis*, a scalable distributed decoder architecture. Ziad et al. [255] introduced the *Local Clustering Decoder* (LCD) implemented on FPGAs to address both accuracy and latency associated with surface code decoding. Under a realistic circuit-level noise model dominated by leakage errors, their adaptive parallel decoder performs one million error-free quantum operations using $4\times$ fewer physical qubits than conventional non-adaptive decoding approaches, with latencies below 1 μs per QEC round using modest FPGA resources. Forlivesi et al. [256] introduced the *Bubble Clustering* (BC) decoder for surface codes, achieving decoding complexity proportional to the square of the number of syndrome defects, which corresponds to near-linear scaling with the number of data qubits.

Although naive BP decoder performs poorly for surface codes because of highly loopy Tanner graphs and code degeneracy, BP-derived decoders have been adopted for topological codes to exploit full noise information, improve decoding accuracy, and preserve the scalable structure of message-passing techniques. Hack et al. [257] proposed three BP-derived decoding algorithms, i.e., belief propagation for matching (BP4M), belief propagation for matching forced (BP4MF), and belief propagation for matching plus matching (BP4M+M) for surface codes and other graph-like QECCs. BP4M acts similar to MWPM under depolarizing noise, BP4MF achieves competitive logical error rates at larger code distances, and BP4M+M further integrates PyMatching to enhance decoding accuracy. Because these methods require relatively few iterations and reduce computational complexity, they provide a promising pathway toward scalable hardware-accelerated real-time decoding implementations. Higgott et al. [228] integrated

[76]Helios is an open-source software available at https://github.com/NamiLiy/Helios_scalable_QEC.

BP with matching- and clustering-based decoders to develop two hybrid frameworks, namely *belief-matching* and *belief-find*, which exploit full noise information to improve decoding accuracy and latency. Their results demonstrated a decoding threshold of approximately 0.94%, outperforming the 0.82% threshold achieved by standard MWPM decoders. Old and Rispler [258] proposed a generalized BP decoder augmented with an outer re-initialization loop to improve BP decoding algorithms for surface codes. Their approach achieved thresholds comparable to ideal thresholds under independent bit-/phase-flip and depolarizing noise models. Kaufmann and Arad [242] replaced the tensor network contraction stage of tensor network decoders with *blockBP*, an approximate contraction algorithm based on BP. Their numerical evaluations showed more than an order-of-magnitude improvement in logical error rate compared with MWPM decoders.

In general, neural network decoders can operate without prior knowledge of physical error rates and can outperform conventional MWPM in the decoding of surface codes. Varbanov et al. [259] developed a neural network decoder trained on both simulated and experimental data from a transmon processor. Their decoder captured correlated errors, including Y errors, and achieved approximately 25% lower logical error rates than MWPM on experimental data associated with Google Quantum AI [176]. By incorporating soft information from analog qubit readout signals, the decoder achieved an additional 10% reduction in logical error rates, highlighting the potential of machine learning decoding algorithms for near-term FTQC. Yang et al. [260] proposed an FPGA-based neural network decoder integrated into a real-time QEC control architecture for surface codes, which achieved a deterministic closed-loop latency[77] of 550 ns, including 124 ns for decoding, within a 1.25 μs QEC cycle. This decoder highlights the potential of FPGA-accelerated machine learning decoding for scalable FTQC systems.

Cellular automata decoders constitute one of the most locality-friendly decoding algorithms and are therefore promising for topological codes. The Harrington decoder [261] is a canonical early CA decoder for two-dimensional toric codes, although its performance is limited by the finite speed of local communication and computation. Breuckmann et al. [262] extended this framework to four-dimensional toric codes, achieving a decoding threshold of approximately 1.59%, substantially higher than the 0.133% threshold reported for the original Harrington decoder. Furthermore, Kubica and Preskill [247] introduced the *sweep rule*, a higher-dimensional generalization of Toom's rule[78], applicable to locally Euclidean lattices in $d \geq 2$ dimensions. They further provided rigorous threshold analyses for both three-dimensional toric codes and color codes.

2) Decoding of Color Codes: Color code decoding is generally more challenging than that of surface codes because a single Pauli error typically triggers three nearby syndrome checks rather than producing pairwise detection events, which results in more complex decoding graphs. However, in practice, the most established real-time-friendly color code decoders are still matching-derived families. Lee et al. [30] proposed an efficient matching-based decoder that combines two matching decoders for each color. Under both ideal bit-flip noise and realistic circuit-level noise models, the decoder demonstrated near-optimal logical error rate scaling proportional to $p^{d/2}$, where p and d denote the physical error rate and code distance, respectively. Although the achieved decoding thresholds, 8.2% for bit-flip noise model and 0.46% for circuit-level noise model, are comparable to existing matching-based color code decoders, the proposed approach substantially improves sub-threshold logical error suppression. Google utilized the neural network decoder *AlphaQubit* [244] for logical memory and randomized benchmarking analyses, and the Möbius matching decoder, called *Chromobius*, [263] for magic state injection experiments in their implementation of the color code on superconducting processors [98].

Several additional decoders have been developed to exploit the relationship between color codes and simpler topological code decoding problems. Kubica and Delfosse [264] introduced the *restriction decoder*, which maps color codes with $d \geq 2$ dimensions into multiple toric code decoding subproblems. Numerical simulations under bit- and phase-flip noise models with perfect syndrome extraction demonstrated a threshold of approximately 10.2% for the two-dimensional color code on the square-octagon lattice, comparable to the threshold of toric codes on square lattice. Kung et al. [265] introduced a BP-OSD decoder enhanced with quaternary reliability statistics for quantum topological codes, achieving a threshold of approximately 15.42% for hexagonal planar color codes. Berent et al. [266] reformulated color code decoding problem as a variation of the classical *LightsOut* puzzle and solved it using the MaxSAT optimization framework. In this approach, syndrome constraints and candidate recovery operations are translated into Boolean satisfiability constraints, enabling the decoder to search for corrections and minimize logical errors and achieve state-of-the-art numerical results for color code decoding.

Recent studies have also examined color code decoding in distributed and hardware-aware settings. Chandra et al. [267] proposed a distributed implementation of the $(6, 6, 6)$ color code, in which multiple color code patches hosted on separate QPUs are interconnected through entangled pairs. They evaluated the robustness of the distributed structure using both a tensor network decoder and a concatenated MWPM decoder. Their simulation results showed that while the tensor network decoder experiences a slight threshold degradation, the concatenated MWPM decoder maintains nearly unchanged thresholds, demonstrating robustness for distributed color code architectures with noisy QPU interconnects. Google Quantum AI scientists [183] introduced two circuits for color codes based on *superdense coding* and a middle-out strategy to reduce the performance gap between color codes and surface codes. They further developed the *Chromobius* decoder, which approximates the color code decoding problem as an MWPM problem. Their evaluations demonstrated that the proposed middle-out color code architecture achieved a *TeraQuOp foot-*

[77]Closed-loop latency refers to the sum of decoding, communication, and feedback delays.

[78]Toom's rule flips the value of a face in a two-dimensional square lattice when the associated north and east parity checks are both non-trivial.

print of approximately $1,250$ qubits, compared with roughly 650 qubits for surface codes decoded using correlated matching. Although surface codes remain more resource efficient in this comparison, the proposed circuits significantly reduce the historical overhead gap between color codes and surface codes.

Finally, color code variants tailored to biased noise can benefit from local decoding rules. Miguel et al. [248] proposed a CA decoder for the *XYZ* color code, a locally rotated color code variant designed to exploit highly biased dephasing noise. Because string-like logical operators are absent in the high-bias regime, the decoder exhibits partially self-correcting memory behavior, with memory lifetime scaling polynomially with system size without requiring global decoding. Their results demonstrated improved memory lifetime under realistic biased noise conditions, highlighting the potential of local CA decoders to reduce the bandwidth and computational demands of global decoding in FTQC systems.

B. Decoding of qLDPC Codes

Unlike topological codes, qLDPC codes generally do not possess a regular low-dimensional lattice structure. Instead, they are represented by sparse Tanner graphs whose stabilizer generators may have complex or non-local connectivity. Therefore, decoding of qLDPC codes must operate directly on the sparse parity-check structure rather than relying on pairwise syndrome matching. This subsection reviews representative qLDPC decoding algorithms, including BP algorithms, code-specific algebraic and local decoders, clustering and search-based methods, learning-based decoders, and hardware-accelerated implementations.

1) Message-passing and BP-based Decoders: Belief propagation combined with ordered-statistics decoding (BP+OSD) has become a standard baseline for general purpose qLDPC decoding problems [241]. BP exploits the sparse Tanner graph structure through iterative message passing, while OSD provides a post-processing step that resolves residual inconsistencies caused by degeneracy, short cycles, and convergence failures. Although BP decoders remain dominant because of their parallelism and low per-iteration complexity, the OSD stage can become computationally expensive and difficult to implement in low-latency hardware. Several recent variants aim to preserve the hardware efficiency of BP while reducing the reliance on the computationally expensive post-processing stage. These include stabilizer inactivation (SI) [268], belief-propagation guided decimation (BPGD) [269], [270], localized statistics decoding (BP+LSD) [271], ordered Tanner forest decoding (BP+OTF) [272], and Relay-BP [273]. Valentini et al. [240] introduced *Restart Belief*, an iterative BP decoder inspired by branch-and-bound optimization, to mitigate convergence limitations of conventional BP decoders while preserving strong error correction performance. IBM Quantum researchers introduced Relay-BP [273], a hardware-oriented heuristic decoder for qLDPC codes that achieved roughly an order-of-magnitude improvement in decoding accuracy over BP+OSD while preserving the compactness, parallelism, and computational efficiency of conventional BP decoders. Subsequent IBM research on the implementation of Relay-BP on an FPGA [274] demonstrated a BP iteration latency of only 24 ns and sub-microsecond average decoding latency per QEC cycle for the Gross code. These results position Relay-BP as a strong candidate for real-time decoding in FTQC systems.

2) Clustering and Search-based Decoders: Several code-specific local and algebraic decoders exploit either the geometric locality or algebraic structure of qLDPC codes to improve the decoding efficiency and accuracy of QECCs. Small-set-flip (SSF) decoding [275] is a linear-time decoder originally developed for HGP codes. Grospellier et al. [276] later extended this decoder by introducing a hybrid BP+SSF decoder to improve performance for large code sizes. Connolly et al. [277] proposed a fast decoder for erasure errors associated with HGP codes, achieving $O(N^2)$ bit complexity, where N denotes the code length, and further reduced the complexity to $O(N^{1.5})$ in a probabilistic variant. Moreover, Gu et al. [278] proposed a linear-time iterative decoder for quantum Tanner codes[79]. Their decoder performs localized corrections within constant-sized regions by iteratively minimizing an efficiently computable local cost function that approximates the residual error weight.

In recent years, clustering and search-based decoders have also been developed to achieve favorable accuracy–latency trade-off in qLDPC codes. Wolanski and Barber [280] introduced the *ambiguity clustering* (AC) decoder and evaluated it on BB codes. Their simulation results showed that AC can be up to $27\times$ faster than BP+OSD at comparable accuracy, with a decoding latency of approximately 135 μs per round for the gross code. Closed-branch decoding explores compact branch structures and achieves performance comparable to BP+OSD with substantially lower complexity on smaller BB codes, although degradation has been observed for larger instances [281]. Decision tree decoders [106] further exploit sparse decision trees over fault additions; the heuristic variant outperforms BP+OSD on the gross code, while the provable variant can return minimum-weight recovery solutions.

IonQ researchers developed a *beam search decoder* for qLDPC codes, demonstrating that IonQ's trapped-ion architecture can support fault-tolerant decoding using conventional CPU-based hardware without specialized accelerators [282]. Their decoder reduced the logical error rate by up to $17\times$ compared with BP+OSD, reduced the $99.9th$ percentile worst-case runtime by $26\times$, and achieved sub-millisecond decoding latency on standard commercial CPUs. In addition, Google Quantum AI researchers developed *Tesseract*, a most-likely error (MLE) decoder based on A^* search algorithm and pruning heuristics [283]. Tesseract achieved significantly faster decoding while maintaining competitive accuracy across surface, color, and BB codes, and further demonstrated that the gross code can provide $14\times$ to $19\times$ higher resource efficiency than surface codes under MLE decoding, compared with $10\times$ improvement when using correlated matching and BP+OSD decoders. The same research community [284] further optimized the Tesseract decoder through several low-level software and hardware-aware enhancements, including efficient mem-

[79]Quantum Tanner codes constitute a family of asymptotically good qLDPC codes with constant encoding rate and linear distance, which provides single-shot QEC [279].

ory layouts, improved data structures, and bitwise operations. Compared with the original Tesseract implementation, the optimized variant achieved approximately $2\times$–$5\times$ simulation speedups across multiple QECC families, particularly for computationally demanding BB code instances. Furthermore, MWPM-based [285] and UF-based [286] decoding algorithms have also been adapted for qLDPC codes to exploit the inherent efficiency, scalability, and low-latency characteristics of these decoding paradigms.

3) Learning-based and Hardware-accelerated Decoders: Machine learning decoders offer another promising direction for qLDPC decoding, especially for graph-structured and repetitive code families, such as BB codes. However, their practical applications require careful handling of training cost, dataset generation, uncertainty calibration, runtime predictability, and robustness across code sizes and noise models. Gu et al. [287] proposed *Cascade*, a convolutional neural network decoder for qLDPC codes, which exploits geometric structure in the underlying QEC lattice to achieve both high accuracy and throughput. Cascade achieved up to $17\times$ lower logical error rates than existing qLDPC decoders and reached a logical error rate near 10^{-10} at a physical error rate of 0.1% for the Gross code. Furthermore, the decoder exhibited a strong *waterfall regime*[80], indicating that the logical error rates required for large-scale FTQC can be achieved with relatively modest code sizes. In addition to improving decoding accuracy, Cascade achieved several orders of magnitude higher decoding throughput and generated calibrated confidence estimates that can reduce the overhead of *repeat-until-success* fault-tolerant protocols.

Gong et al. [288] introduced a hybrid BP+GNN decoder in which that the GNN leverages information from preceding BP iterations to improve the initialization of subsequent BP runs, compensating for suboptimal BP behavior caused by qLDPC graph structure. Maan and Paler [289] introduced *Astra*, a scalable GNN decoder that demonstrated higher decoding threshold and improved error correction performance compared with BP+OSD for surface codes trained up to distance-11 and BB codes trained up to distance-18. Furthermore, Blue et al. [290] proposed a recurrent transformer-based neural network decoder for BB codes under the circuit-level noise model. Their simulation results demonstrated nearly $5\times$ lower logical error rates than BP+OSD for the Gross code, together with more consistent runtime behavior and an order-of-magnitude reduction in worst-case decoding latency.

Hardware acceleration is essential for making qLDPC decoding compatible with real-time FTQC constraints. Maan et al. [291] proposed *graph augmentation and rewiring for inference* (GARI) for decoding of correlated errors in qLDPC codes, which removes 4-cycles associated with correlated Y errors to improve message-passing performance. Using normalized min-sum and parallel ensemble decoding, it achieved logical error rates as low as 6.70×10^{-9} for the distance-12 BB code, while its FPGA implementation demonstrated an average latency of only 273 ns per QEC round. Ferraz et al. [292] developed a GPU-accelerated parallel BP decoder with latency below 50 μs, reaching 23.3 μs for smaller codes. These results satisfy the stringent timing requirements of superconducting QEC cycles and outperform the sub-$63\,\mu$s decoding latency reported in Google's *Willow* processor experiments.

Overall, qLDPC decoding is evolving from generic BP+OSD baselines toward specialized, hybrid, and hardware-aware decoders. BP decoders provide scalable message passing, code-specific decoders exploit algebraic or geometric structure, search-based methods improve accuracy for finite-size BB codes, learning-based decoders capture complex circuit-level correlations, and FPGA/GPU implementations address real-time latency. These complementary directions are essential for realizing the low-overhead advantages of qLDPC codes in practical FTQC architectures.

VII. Benchmarks and Simulations

Benchmarking large-scale FTQC requires a systematic evaluation of how distributed quantum systems perform fault-tolerant operations, including their ability to suppress errors, execute logical operations, and scale under resource and connectivity constraints. However, benchmarking in this context remains challenging due to the lack of unified evaluation metrics, heterogeneity of computing architectures, absence of established DQEC benchmarks, and the limited metrics tailored to DQC and HeDQC, which make fair, reproducible, and architecture level comparisons challenging. In MQC, standard evaluation metrics include the logical error rate P_L, threshold p_{th}, resource overhead (e.g., number of physical qubits and gates), and space-time volume to capture the trade-off between spatial and temporal resources [230].

Distributed quantum computing extends this benchmarking landscape by introducing additional dimensions, in particular communication dependent metrics, such as the number of inter-QPU interactions, entanglement fidelity F_{en}, entanglement generation rate, entanglement routing overhead, and communication latency. These metrics are essential because distributed fault-tolerant operations depend not only on local gate fidelity and code performance, but also on the quality, availability, and timing of non-local entanglement resources. Furthermore, HeDQC and vendor-specific metrics [28] broaden this evaluation space by metrics that capture system-level metrics beyond HoDQC and cross-platform interactions among distinct qubit modalities. Relevant system-level metrics include modality utilization efficiency, cross-modality entanglement fidelity, interface and transduction overhead, and heterogeneous communication latency. Moreover, additional factors, such as cross-platform logical error rates and synchronization overhead across heterogeneous control systems become critical for assessing overall system performance. Collectively, these metrics reflect the intrinsic challenges of integrating diverse qubit modalities while balancing computation, communication, and control across heterogeneous quantum technologies [141].

Several quantum computing companies currently provide public cloud access to quantum processors through public, subscription-based, or partner-access platforms. Representative

[80]The waterfall regime refers to a rapid, often exponential, reduction in logical error rates as the code size increases below the threshold.

examples include IBM Quantum processors (e.g., Heron r2, Heron r3, and Nighthawk) accessible through *Qiskit*, Quantinuum trapped-ion systems (e.g., H2-1, H2-2, and Helios), IonQ processors (e.g., Forte and Forte Enterprise), Rigetti superconducting processors (e.g., Cepheus-1-108Q and Ankaa-3) accessible via *pyQuil*, D-Wave systems (e.g., Advantage2) accessible via *Ocean*, and QuEra neutral-atom processors (e.g., Aquila) accessible via *Amazon Braket*[81]. Recent modular hardware demonstrations, such as Rigetti's Cepheus-1-108Q processor and IBM's experiments on BB code, illustrate the growing relevance of multi-chip and multi-module architectures for scaling quantum systems. However, despite the increasing availability of quantum cloud, practical limitations motivate the development of quantum simulation tools and software libraries. Current quantum processors typically impose restrictions on execution time, shot count, queue priority, and the number of accessible physical qubits, which limits the execution of application-scale algorithms. Moreover, end users generally have limited control over hardware topology, noise processes, and time-dependent calibration drift. Therefore, simulation tools play a critical role in quantum computing research, idealized circuit verification, custom noise modeling, and reproducible benchmarking.

Simulation, compilation, and benchmarking tools for large-scale FTQC can typically be categorized into three classes.

Dedicated quantum programming languages: This class consists of dedicated quantum programming languages, often referred to as *standalone languages*, such as Scaffold [304], Q# [305], Qrisp [306], isQ [307], and Q|SI⟩ [308]. These languages provide high-level abstractions for quantum algorithms and, in some cases, resource estimation and compilation. However, their adoption is often limited by the need to learn new syntax, semantics, and programming paradigms that differ from conventional classical programming languages. At a lower abstraction level, quantum assembly languages (QASM), such as OpenQASM [309], eQASM [310], and NetQASM [311], serve as intermediate representations for hardware-level execution within the compilation stack.

Quantum software development kits (SDKs): This class comprises quantum software development kits embedded in widely used classical programming environments, including *Qiskit* [294], *Cirq*[82], Bloqade[83], and *Qulacs* [301], which allow users to design, transpile, simulate, and execute quantum circuits using classical programming languages such as Python. These SDKs are widely used because they integrate circuit design, backend access, noise modeling, transpilation, visualization, and hybrid classical–quantum workflows within familiar software ecosystems.

Specialized FTQC analysis and benchmarking tools: This class includes specialized tools targeting specific layers of the fault-tolerant stack to support technical analysis. Stabilizer simulators, such as *Stim*[84], enable efficient simulation of large Clifford circuits and are particularly important for QEC and decoding studies. Resource estimation and fault-tolerant analysis frameworks, such as *Qualtran*[85] [295], support high-level evaluation of algorithmic and architectural overhead. Other tools target different requirements, for instance, *tqec*[86] [312] developed for topological QEC, *Crumble*[87] introduced for error propagation analysis, *Sinter*[88], *qecsim*[89], *QTop*[90], *PanQEC*[91], *Plaqette*[92], and *ECCentric* [313] proposed for QEC simulation and benchmarking, and *NetSquid*[93] [296] designed for distributed quantum-networked simulations.

Table XI presents a feature-oriented comparison of representative tools for FTQC design and analysis in terms of several functional layers, including circuit level (CL) simulation, quantum error correction (QEC), error mitigation (EM), decoding (DEC), and distributed quantum computing (DQC), noise model (NM) support, fault-tolerant (FT) execution. The comparison highlights the diversity of simulation methods and implementation technologies that illustrate the rapidly evolving FTQC toolchains. Simulation methods refer to the classical computational techniques used to emulate the quantum state evolution, quantum circuits, and noisy quantum processes under different assumptions about state representation, noise modeling, circuit structure, and scalability. These methods are usually inherited from classical numerical linear algebra, stabilizer formalism, graph theory, tensor contraction, probabilistic sampling, and statistical simulation. Several classical methods are typically utilized to model quantum systems in quantum simulators, as outlined below. Each method provides a distinct trade-off among accuracy, computational cost, and applicability, making it suitable for different layers of FTQC analysis.

- *Statevector simulation:* represents an n-qubit pure state as a complex amplitude vector of dimension 2^n and updates the state through matrix vector operations. It provides exact simulation of ideal quantum circuits, but its memory and runtime scale exponentially with the number of qubits.
- *Density matrix simulation:* represents quantum states by a $2^n \times 2^n$ density operator, thereby supporting mixed states and general quantum channels. This makes it suitable for detailed noise modeling, but its memory complexity scales as $O(4^n)$, making it substantially more costly than statevector simulation.
- *Stabilizer simulation:* exploits the Gottesman–Knill theorem and represents Clifford circuits through stabilizer generators rather than full quantum state vectors. This enables efficient simulation of large Clifford circuits

[81] Amazon Braket, launched by AWS in 2020, is the largest multi-vendor quantum cloud at the time of writing that provides cloud-based access to quantum hardware across multiple modalities, simulators, and hybrid classical–quantum workflows.

[82] https://github.com/quantumlib/Cirq

[83] https://github.com/QuEraComputing/bloqade

[84] https://github.com/quantumlib/Stim?utm_source=chatgpt.com

[85] https://github.com/quantumlib/Qualtran

[86] https://github.com/tqec

[87] https://algassert.com/crumble

[88] https://github.com/quantumlib/Stim/wiki/Sinter-v1.13-Python-API-Reference

[89] https://github.com/qecsim/qecsim

[90] https://github.com/jacobmarks/QTop

[91] https://github.com/panqec/panqec

[92] https://github.com/qubit-ulm/ProtoPlaqette

[93] https://github.com/ccicconetti/netsquid

TABLE XI: Comparison of representative tools for fault-tolerant quantum computing analysis and design

Tool	Type	Developer	Features							Implementation Language	Simulation Method
			CL	QEC	DEC	EM	NM	DQC	FT		
Stim [293]	Simulator	Google	✓	✓	×	×	✓	×	✓	Python/C++	Stabilizer
Qiskit (Aer) [294]‡	SDK	IBM	✓	✓	×	✓	✓	×	×	Python/C++	Statevector
Qualtran [295]	Library	Google	✓	✓	×	×	×	×	✓	Python	Analytical
NetSquid [296]	Simulator	QuTech	×	×	×	×	✓	✓	×	Python	Discrete-event
Mitiq [297]§	Library	Unitary Fund	✓	×	×	✓	×	×	×	Python	Sampling
Sinter [298]	Library	Google	✓	✓	✓	×	✓	×	✓	Python	Monte Carlo
TKET [299]*	Compiler	Quantinuum	✓	✓	×	✓	✓	×	✓	Python/C++	Compilation
AQRE† [300]	Framework	Microsoft	✓	✓	×	×	×	×	✓	Python/Q#	Analytical
Cirq	SDK	Google	✓	✓	×	✓	✓	×	×	Python	Statevector
ABS°,♣	Library	Amazon	✓	✓	×	✓	✓	×	×	Python	Tensor network
CUDA-Q△	Framework	NVIDIA	✓	✓	×	✓	✓	✓	×	Python/C++	Tensor network
Qulacs [301]#	Simulator	Qulacs Team◇	✓	✓	×	×	✓	×	×	Python/C++	Statevector
qecsim•	Simulator	EQUS□	✓	✓	✓	×	✓	×	✓	Python	Monte Carlo
SeQUeNCe [302]♮	Simulator	Argonne Lab	×	×	×	×	✓	✓	×	Python	Discrete-event
PECOS♠	Library	Quantinuum	✓	✓	✓	×	✓	×	✓	Python	Stabilizer
Tsim [303]♦	Simulator	QuEra	✓	✓	×	×	✓	×	✓	Python/JAX	ZX-calculus⊛

†**AQRE:** Azure Quantum Resource Estimator; °**ABS:** Amazon Braket simulators; ◇**Qulacs Team:** QunaSys, Osaka University, NTT, and Fujitsu; □**EQUS:** Australian Research Council Centre of Excellence for Engineered Quantum Systems.
‡: https://github.com/Qiskit/qiskit-aer; ∗: https://github.com/Quantinuum/tket; §: https://github.com/unitaryfoundation/mitiq;
♣: https://github.com/amazon-braket/amazon-braket-default-simulator-python; △: https://github.com/NVIDIA/cuda-quantum;
#: https://github.com/qulacs/qulacs; •: https://github.com/qecsim/qecsim; ♮: https://github.com/sequence-toolbox/SeQUeNCe;
♠: https://github.com/PECOS-packages/PECOS; ♦: https://github.com/QuEraComputing/tsim
⊛: **ZX-calculus** is a diagrammatic formalism and an alternative way to represent quantum states, circuits, and elements using spiders, equipped with a set of rewrite rules. QuEra is developing *PPVM* simulator as an extended version of Tsim, supporting atom loss and the next generation of FTQC.

with Pauli noise models, making stabilizer simulation particularly useful for QEC and FTQC analysis.

- *Tensor network simulation:* represents quantum states or circuits as networks of tensors and evaluates them through tensor contraction. This exploits limited entanglement, low circuit depth, or favorable circuit geometry to simulate structured quantum circuits that are infeasible for full statevector simulation.
- *Monte Carlo and sampling-based techniques:* approximate noisy quantum processes through repeated randomized trials, quantum trajectories, stochastic error realizations, or error sampling. These methods provide a practical approach for estimating output distributions, logical error rates, decoder performance, and fault-tolerant thresholds under probabilistic noise models.

Given the critical role of decoding in FTQC, both academic and industrial communities have developed a broad range of simulation tools and software packages for implementing, modeling, and benchmarking decoding algorithms. Representative tools are summarized in Table XII, which compares their decoding families, implementation languages, hardware compatibility, and developers. Several open-source repositories provide implementations of MWPM and other matching-based decoders, including *PyMatching*[94] [314], *Fusion Blossom*[95], and *Chromobius* [263]. Moreover, *Qsurface*[96] is a Python-based framework for union-find decoding, while the *LDPC* package[97] includes message-passing and post-processing decoders, such as BP+OSD and BP+LSD, for qLDPC codes. Tensor network decoding is supported by *SweepContractor.jl*[98], and cellular automata decoders are represented by *Sweep-Decoder-Boundaries*[99], which implements the sweep rule, and *LocalToricDecoder*[100], which is based on Toom's rule. Machine learning decoders include *neural network decoder*[101], *deep neural decoder*[102], *Ising*[103], and a dedicated neural network decoder for color codes[104]. Furthermore, an open-source MaxSAT-based decoder has been developed as part of the Munich quantum toolkit (MQT)[105]. These tools differ not only in their underlying decoding algorithms, but also in their programming languages and hardware targets, ranging from CPU-friendly to GPU-friendly decoders.

VIII. Strategic Roadmaps Toward FTQC

Leading quantum enterprises have released comprehensive, multi-layered roadmaps for the transition from

[94]The latest version of PyMatching incorporates the sparse blossom solver.
[95]https://github.com/yuewuo/fusion-blossom
[96]https://github.com/watermarkhu/qsurface
[97]https://github.com/quantumgizmos/ldpc
[98]https://github.com/chubbc/SweepContractor.jl
[99]https://github.com/MikeVasmer/Sweep-Decoder-Boundaries
[100]https://github.com/kduivenvoorden/LocalToricDecoder
[101]https://github.com/Krastanov/neural-decoder
[102]https://github.com/pooya-git/DeepNeuralDecoder
[103]https://github.com/NVIDIA/ising-decoding
[104]https://github.com/baireuther/neural_network_decoder
[105]https://github.com/munich-quantum-toolkit/qecc

TABLE XII: Representative decoding tools and frameworks for QEC

Decoding Tool	Decoding Family	Language	Hardware Compatibility	Developer
PyMatching v.2	MWPM	Python/C++	CPU	Oscar Higgott
Fusion Blossom	MWPM	Rust	One- or Multi-core CPU	Yue Wu
ProtoPlaqette	MWPM, Lookup	Python	CPU	UU & FJ[‡]
Qsurface	MWPM, UF, UFNS[°]	Python	One- or Multi-core CPU	Academic Community
MQT[†]-QECC	UF, MaxSAT	Python	CPU	Academic Community
Local Toric Decoder	Cellular Automata	Python	One- and Multi-core CPU	Kasper Duivenvoorden
Sweep Decoder Boundaries	Cellular Automata (CA)	C++	One- and Multi-core CPU	Michael Vasmer
LDPC v.2	UF, MP, BP+OSD, BP+LSD, BF[◇]	Python/C++	CPU	Joschka Roffe
Sweep Contractor.jl	Tensor Network	Julia	CPU	Christopher T. Chubb
Neural Network Decoder	Machine Learning	Python	GPU	Stefan Krastanov
Deep Neural Decoder	Machine Learning	Python	GPU	Pooya Ronagh
Tesseract	Search-based	Python & C++	CPU	Google Quantum AI
Ising	Machine Learning	Python & CUDA-Q	GPU	NVIDIA
Chromobius	Matching-based	Python/C++	One- or Multi-core CPU	Google Quantum AI

[‡]**UU & FJ:** Ulm University, Forschungszentrum Jülich; [°]**UFNS:** Union-Find Node-Suspension; [†]**MQT:** Munich Quantum Toolkit; [◇]**BF:** Belief-Find.

noisy intermediate-scale quantum (NISQ) processors toward *intermediate-scale quantum (ISQ)*, *utility-scale quantum* (USQ), and ultimately FTQC systems. As illustrated in Figure 17[106], these roadmaps no longer project solely qubit counts. Instead, they detail key architectural considerations, including logical qubits, fault-tolerant logical operations, real-time decoding, modular implementation, and integration with high-performance computing (HPC) infrastructures [26]. These roadmaps can also be interpreted by examining how each company views the role of QEC in enabling FTQC and how deeply QEC is embedded within its long-term technology strategy. Riverlane categorized quantum computing companies into five categories based on their level of engagement with QEC, (i) *not pursuing*, i.e., no visible QEC activity, (ii) *nascent*, i.e., exploring basic error mitigation while outlining an initial QEC roadmap, (iii) *prioritization*, i.e., establishing a structured QEC strategy, publishing research, and recruiting domain experts, (iv) *implementation*, i.e., actively developing and integrating QEC into their systems, and (v) *differentiation*, i.e., treating QEC as a central strategic focus and a key competitive advantage [68].

Although these industrial roadmaps differ in hardware modality, scaling strategy, and error correction approach, they share a common objective, the transition from noisy physical qubits to reliable logical qubits capable of supporting fault-tolerant execution. To clarify these differences, we organize representative company roadmaps according to their dominant physical platforms as follows.

A. Superconducting Qubit Roadmaps

IBM Quantum outlines a tightly structured, architecture-driven roadmap based on modularity to deliver an error-corrected quantum system, called *Starling*, by 2029 and enable practical FTQC access to users by the end of the decade. In 2025, IBM focused on the *Loon* architecture to demonstrate a quantum memory based on BB codes, including ∼50 data qubits and implementation of C-couplers. In 2026, this progression is followed by the *Kookaburra processor*, which introduces a complete module comprising a logical processing unit and quantum memory, while targeting real-time decoding and improved logical qubit coherence time. In 2027, the *Cockatoo* processor is expected to enable inter-module entanglement via quantum interconnects, demonstrating scalable modular coupling (i.e., M- and l-couplers) and distributed logical operations. By 2028, an early *Starling* system will integrate multiple modules to realize a fault-tolerant *instruction set architecture* (ISA), including key primitives such as magic state distillation [79]. These developments culminate in a fully fault-tolerant Starling system by 2029, targeting approximately 100 million fault-tolerant operations on around 200 LQs, with further scaling toward billion gates and thousands of logical qubits in the *Blue Jay* architecture envisioned for the 2033+ timeframe [24]. In parallel, IBM pursues near-term quantum advantage by end of 2026 through modular platforms such as *Quantum System II*[107] and improved processors (e.g., *Heron* and *Nighthawk*), bridging the gap between NISQ-era and fully fault-tolerant architectures. As part of its roadmap, IBM announced in May 2026, in collaboration with the *U.S. Department of Commerce*, the establishment of *Anderon*, the first purpose-built quantum foundry in the United States. This initiative aims to enable industrial-scale fabrication of quantum hardware and support a broader ecosystem of quantum technology providers beyond IBM.

Google Quantum AI follows a QEC-driven roadmap toward large-scale FTQC based on superconducting processors and surface code error correction. Its roadmap is structured around six technical milestones that progress from physical qubit demonstrations to scalable logical systems in the late 2020s and beyond. The first two milestones have already been

[106]This figure is not intended to provide an exhaustive list of quantum computing companies with published roadmaps, nor does it include all organizations actively pursuing FTQC without publicly disclosed plans.

[107]IBM classifies its quantum processors into two generations, i.e., *Quantum System I*, monolithic processors such as *Condor* and *Osprey*, and *Quantum System II*, modular architectures such as *Heron*.

achieved through the demonstration of quantum supremacy on the *Sycamore* processor in 2019 [176] and the experimental validation of logical error suppression and below-threshold operation on the *Willow* processor in 2023 [175]. The subsequent milestones focus on realizing long-lived logical qubits, for which partial progress has already been demonstrated, and on implementing fault-tolerant logical gates via lattice surgery and magic state distillation. Beyond 2028, its trajectory targets scaling the number of logical qubits from 10^3 to 10^6, while reducing the logical error rate from 10^{-6} to 10^{-13}.

Rigetti Computing follows a hardware-centric roadmap, specifically *chiplet-based* architectures. Its strategy prioritizes the modular tiling of smaller processors, improvements in two-qubit gate fidelity, and inter-module connectivity as prerequisites for scalable quantum systems. Rigetti's modular design trajectory began with the *Aspen* 80-qubit chip family, which relies on fixed coupler architectures, followed by the *Ankaa* 84-qubit architecture based on tunable couplers. Rigetti also introduced *Novera*, a commercially available 9-qubit chip featuring a median two-qubit gate fidelity of approximately 99.5% and a median single-qubit gate fidelity of 99.93%. A major step in Rigetti's modular roadmap is *Cepheus-1-108Q*[108], a 108-qubit modular multi-chip processor comprising a 3×4 array of twelve interconnected Novera chiplets based on tunable-coupler technology. Rigetti's near-term milestone targets a $150+$ qubit chiplet with two-qubit gate fidelity of almost 99.7% by around the end of 2026, followed by a $1,000+$ qubit system with an anticipated median two-qubit gate fidelity of approximately 99.8% by around the end of 2027. Beyond increasing chiplet size and density, Rigetti plans to scale its modular architecture through *passive carrier substrates* and multi-chip packaging, aligning its roadmap with broader FTQC strategies based on modular integration and scalable interconnects.

B. Trapped-ion Roadmaps

Quantinuum outlines an accelerated, hardware–software co-designed roadmap toward universal FTQC by 2030, building on the progress of its *H-Series* processors. In November 2025, Quantinuum introduced *Helios*[109], a 98-qubit trapped-ion processor based on the quantum charge-coupled device (QCCD) architecture. Helios supports 98 physical qubits and $\sim$ 48 logical qubits[110], with logical error rates on the order of $\leq 10^{-4}$. Helios achieves average two-qubit and single-qubit gate fidelities of approximately 99.921% and 99.9975%, respectively.[111]. Quantinuum further targets the development of the *Sol* processor by 2027, aiming for 192 physical qubits in a two-dimensional QCCD architecture while further suppressing logical error rates to approximately $P_L \approx 10^{-5}$. By 2029, the *Apollo* system is expected to extend this trajectory to $\sim 1000s$ physical qubits with improved logical error rates of $P_L < 10^{-5}$, thereby supporting more complex error-corrected computations [177].

IonQ follows a performance- and scalability-driven roadmap toward FTQC, leveraging high-fidelity trapped-ion gates, all-to-all connectivity, and modular architectures. Its roadmap progresses from the *Forte Enterprise* system, with $36+$ physical qubits and approximately 99.6% physical qubit fidelity in 2024, to the *Tempo* architecture, with $64 - 100+$ physical qubits in 2025 and achieving an algorithmic qubit milestone of #AQ 64. By the end of 2026, IonQ targets $100-256+$ physical qubits based on an *EQC system*[112], with logical error rates below 10^{-7}. In October 2025, IonQ reported the world record for two-qubit gate fidelity of almost 99.99%. In the longer term, IonQ envisions scaling through photonic interconnects and multi-core architectures, targeting approximately 2×10^5 physical qubits, $8{,}000$ logical qubits, and logical error rates below 10^{-12} by 2029, ultimately aiming to enable large-scale fault-tolerant computation [282].

C. Neutral-atom Roadmaps

QuEra follows an architecture-driven roadmap toward commercially valuable fault-tolerant computing based on neutral-atom platforms. It launched *Aquila* on Amazon Braket in November 2022 as the world's first publicly available neutral-atom quantum computer. Aquila is a 256-qubit analog Rydberg quantum processor and has maintained approximately 99.9% uptime on Amazon Braket for more than three years [315]. QuEra subsequently introduced *Gemini*, described as the world's first on-premises neutral-atom logical quantum computer, with 260 physical qubits and a two-qubit gate fidelity of approximately 99.5%, which was installed at AIST in Japan in 2025 [82]. In parallel, QuEra reported two-qubit gate fidelities of 99.96% using loss post-selection and 99.74% over a large zone involving many qubits. Looking ahead, QuEra's roadmap envisions the launch of *Libra*, a *MegaQuOp* system expected to be accessible through Amazon Braket by 2028. Libra is projected to scale to $\geq$ 10,000 physical qubits and up to 256 logical qubits, with a logical error rate of $\leq 10^{-6}$. The roadmap further emphasizes continuous operation, magic state generation [316], accelerated decoding [245], and reduced space–time overhead[113] as key requirements for a next-generation *GigaQuOp* architecture. This longer-term system is expected to include $\geq$ 20,000 physical qubits and more than $1,000$ logical qubits, with a logical error rate of $\leq 10^{-9}$, thereby supporting increasingly large and commercially relevant fault-tolerant workloads.

Infleqtion follows a neutral-atom, full-stack roadmap toward utility-scale FTQC, centered on its *Sqale* platform. The roadmap is organized around three major milestones. Infleqtion targets an error-corrected system with more than 30 logical qubits, approximately $4,000$ physical atoms, and gate

[108]Cepheus-1-108Q has been available through Rigetti *Quantum Cloud Services* (QCS) and Amazon Braket since April 2026, and has also been accessible through the *Open Quantum* platform since May 2026.

[109]*Helios* is based on barium-ion qubits rather than ytterbium-ion qubits, in which ions can be manipulated using visible light lasers.

[110]Its physical-to-logical qubit ratio is 2:1.

[111]Quantinuum announced the deployment of Helios in a new Singapore R&D center in March 2026.

[112]Electronic qubit control (EQC) replaces laser-based qubit control with microwave and radio frequency electronic fields delivered through CMOS integrated ion-trap chips, making trapped-ion systems potentially easier to scale and integrate.

[113]QuEra introduced ultra-high rate QECCs with an encoding rate of $1/2$ on reconfigurable atom arrays [317].

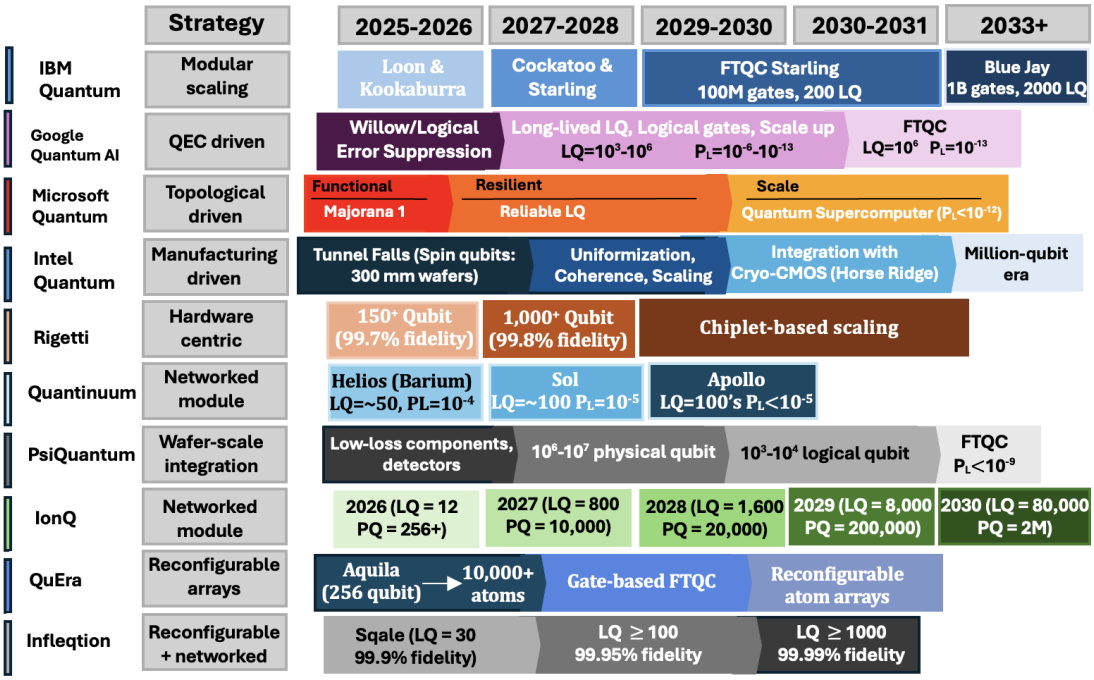


Fig. 17: Industrial roadmaps toward large-scale FTQC.

fidelities around 99.90%, enabled by optimized compilation of fault-tolerant circuits, individual qubit addressing, and dynamic reconfigurable structure by the end of 2026. Infleqtion roadmap envisions reaching an *inflection point* with more than 100 logical qubits, ≥ 1000 physical qubits, and gate fidelities around 99.95%, supported by exponential speed-up demonstrations and real-time atom reloading. The roadmap further emphasizes utility-scale operation with more than $1,000$ logical qubits, $\geq 100,000$ physical qubits, and gate fidelities exceeding 99.99%, enabled by soft-decoding enhancements and large-scale advanced photonic beam steering. Beyond these system-level targets, Infleqtion's roadmap emphasizes a full-stack neutral-atom strategy that integrates hardware-aware software, QEC-enabling architectures, high-fidelity entangling gates, atom transport, and resource estimation. Recent progress includes the release of *resource-superstaq*, an open-source architecture-level resource estimation tool, together with advances in dual-species rubidium–cesium entangling gates, gate-design theory, and scalable optical atom transport. These developments are intended to reduce resource overhead, improve magic state generation, support fast in-place syndrome extraction, and enable scalable fault-tolerant operation. In this roadmap, the combination of individual addressing, dynamic atom-array reconfiguration, real-time atom reloading, and photonic beam steering provides a pathway toward large neutral-atom systems capable of supporting commercially relevant FTQC workloads.

D. Photonic Qubit Roadmapds

PsiQuantum adopts a fusion-based photonic roadmap toward large-scale FTQC, targeting a utility-scale system in the late 2020s. Its approach is based on silicon photonics, wafer-scale fabrication, and integrated optical circuits, which targets scaling to millions of physical qubits from the outset. Early milestones focus on optical loss reduction, high-efficiency photon sources and detectors, with efficiencies $> 99\%$, and highly integrated optical circuits, supported by semiconductor manufacturing at scale. The subsequent phase aims to assemble these components into modular, error-corrected architectures comprising $\sim 10^6 - 10^7$ physical qubits, enabling $\sim 10^3 - 10^4$ logical qubits. Ultimately, PsiQuantum seeks to realize a fully fault-tolerant system capable of executing application-scale quantum algorithms with logical error rates below 10^{-9} [63].

E. Silicon and Topological Qubits Roadmaps

Microsoft Quantum outlines a topological qubit-driven roadmap based on hardware-level error protection using *Majorana* zero modes in hybrid semiconductor–superconductor devices such as InAs/Al nanowires (i.e., indium arsenide/aluminum nanowires). Microsoft aims to realize intrinsically protected qubits that could reduce the overhead required for FTQC, instead of relying solely on conventional physical qubits protected by high overhead QECCs. Its roadmap is structured across three-level measures, i.e., foundational (i.e., noisy qubits), resilient (i.e., reliable logical qubits), and scale (i.e., scalable quantum systems). Following recent demonstrations regarding Majorana mode control on the *Majorana 1* processor in February 2025, Microsoft's roadmap focuses on improving system fidelity and scaling toward multi-qubit topological processors. This materials- and physics-driven roadmap is tightly integrated with the *Azure Quantum* stack, targeting a *quantum supercomputer* capable of supporting millions of reliable qubits with projected error rates below 10^{-12} and high operational throughput.

Intel follows a manufacturing-driven roadmap based on silicon spin qubits fabricated using CMOS-compatible processors. Early milestones include the development of research processors such as *Tunnel Falls*, a 12-qubit silicon spin chip, for device validation and benchmarking, along with demonstrations of high-yield qubit fabrication on 300-mm wafers. Intel's longer-term roadmap focuses on improving qubit uniformity, coherence, and two-qubit fidelity while scaling through tile-based architectures that can be manufactured and interconnected using semiconductor techniques. This strategy is complemented by the integration of cryo-CMOS control electronics such as *Horse Ridge*, enabling a transition toward the *million-qubit era*, where quantum processors act as specialized accelerators within a hybrid classical–quantum stack. In parallel, Intel advances co-designed classical–quantum systems to support real-time control and decoding.

Overall, industrial roadmaps toward FTQC indicate a broad shift from physical qubit scaling toward logical qubit quality, logical error suppression, and fault-tolerant operation depth. While each modality follows a distinct scaling pathway, several cross-modality solutions are identified. For instance, QEC is increasingly treated as a core architectural requirement, modularity has become a dominant scaling principle, real-time decoding and classical–quantum co-processing are becoming integral to FTQC development. While the stated timelines and performance targets should be interpreted as strategic projections rather than guaranteed outcomes, these roadmaps collectively indicate convergence toward error-corrected, modular, and full-stack architectures for large-scale FTQC.

IX. Open Challenges and Research Directions for Scalable FTQC

Despite substantial progress in quantum hardware, QECCs, decoding algorithms, and classical–quantum platforms, the realization of large-scale FTQC remains a full-stack challenge. Practical FTQC requires not only low physical error rates and high-quality logical qubits, but also scalable architectures, efficient interconnects, real-time decoding, classical–quantum collaboration, and hardware-aware compilation. In alignment with the structure of this survey, we classify the major open challenges into five interrelated domains, i.e., fault-avoidance and architecture-aware design, fault-tolerant implementations, execution-time error containment, post-processing and hybrid classical–quantum integration, and modality-specific manufacturing and deployment constraints.

A. Fault Avoidance and Architecture-aware Design

Fault-avoidance strategies aim to proactively reduce the probability of error occurrence before errors occur or propagate during computation by leveraging noise resilient hardware, noise-aware control and calibration, hardware adopted code construction, and architectural optimizations. Despite significant advances, the realization of utility-scale FTQC remains constrained by incomplete noise characterization, limited control of microscopic error mechanisms[114], hardware limitations at subatomic scales, and challenges associated with architectural scalability. These limitations are further exacerbated in DQC, where inter-QPU communication overhead, heterogeneous hardware characteristics, and cross-platform features introduce additional factors and constraints [31].

- **Modular and heterogeneous quantum architectures**

Although modular architectures are increasingly viewed as a promising paradigm toward scalable quantum computing, heterogeneous DQC further enables the integration of diverse qubit modalities to employ the unique feature of each platform. HeDQC can be configured not only to exploit the unique strengths of different qubit modalities for specific computational tasks, but also to mitigate the limitations of each modality through complementary capabilities of others within the same computational fabric. However, modular and HeDQC systems introduce a new design space in which scalability is determined not only by qubit count, but also by communication efficiency, inter-module entanglement generation, calibration compatibility, and control integration. Heterogeneity further complicates system design due to incompatible control electronics, calibration complexities, timing and clock mismatches, diverse noise characteristics, and inconsistent coherence times. In addition, high fidelity, low latency, and noise resilient module-to-module couplers remain underdeveloped, particularly for communication across distinct qubit modalities [28].

Several research directions can address the aforementioned challenges. First, architecture-aware role assignment is required to to find the maximum match between underlying HeDQC features and each quantum task's requirements. Second, unified programming, control, and calibration abstractions are required to hide device-specific and modality-specific complexity while still exposing hardware-relevant constraints to compilers and schedulers. Third, efficient quantum transducers, chip-to-chip couplers, and interconnects are essential for cross-modality communication. Finally, standardized cross-platform error models and performance metrics are required to evaluate trade-offs among conversion overhead, communication cost, logical fidelity, latency, and scalability. Future HeDQC systems will therefore require tight hardware–software co-design, standardized interfaces, and prototype level benchmarking methodologies that evaluate both local quantum performance and distributed system behavior [21].

B. Fault-Tolerant Implementation Challenges

Fault-tolerant implementation requires the reliable realization of quantum memories, logical gates, syndrome extraction, decoding, and inter-QPU communication under strict noise and latency constraints. Fault-tolerant memories address code distance, QEC overhead, threshold requirements, coherence time, and compatibility with the underlying hardware topology. Fault-tolerant logical operations prevent uncontrolled error propagation during computations. These implementations introduce substantial redundancy in terms of ancilla qubits, additional gates, repeated syndrome extraction, and classical post-processing, which increases circuit depth, latency, and exposure to decoherence. In distributed architectures,

[114]This stems from the limited understanding of quantum dynamics and control of complex cause-and-effect relationships within quantum systems.

these challenges extend beyond local QEC cycles. Logical operations may require non-local gates, distributed stabilizer measurements, or entanglement-assisted teleportation between QPUs. Therefore, the cost and reliability of FTQC depend strongly on the number of inter-QPU links, the physical or optical distance between modules, the fidelity of distributed entanglement, and the scheduling strategy used to coordinate remote operations [49].

• **Entanglement distribution and connectivity constraints**

The realization of FTQC in distributed architectures is fundamentally constrained by connectivity features between QPUs, such as inter-QPU connectivity, spatial distance between QPUs, and entanglement distribution mechanisms, which are exacerbated in HeDQC platforms. Distributed FTQC depends on the generation, distribution, purification, and consumption of high-fidelity entangled states across QPUs. Even when distance is short, excessive reliance on non-local operations can substantially increase latency and resource overhead. Reducing inter-QPU communication is therefore essential for improving scalability. Although small multipartite entangled states, such as a *GHZ* state, can be combined to form larger ones through techniques such as *fusion-retain-only* (FRO) and *general fusion* (GF), these processes must be carefully integrated with entanglement purification protocols and entanglement swapping mechanisms to preserve high fidelity and avoid noise accumulation [154].

Addressing these limitations requires coordinated advances across connectivity, distance-aware distribution, and communication-aware compilation. The logical qubit allocation across QPUs can be formulated as a graph partitioning or min-cut problem, where qubits and interactions are represented as vertices and edges, respectively. Classical algorithms such as Ford–Fulkerson, Edmonds–Karp, as well as global min-cut approaches, including Stoer–Wagner, and Karger–Stein, can help identify communication bottlenecks and optimize qubit allocation. These methods can be complemented by machine learning heuristics, graph neural networks, and reinforcement learning to adapt partitioning and routing decisions under dynamic hardware constraints. Distance-aware protocols should further incorporate adaptive routing, loss-aware scheduling, and entanglement purification to mitigate fidelity degradation over long distances. Moreover, scalable entanglement distribution frameworks are required to support both pre-distributed and on-demand distribution among QPUs. Graph-state representations provide a flexible foundation for constructing multipartite entangled states across distributed systems, where intermediate qubits facilitate entanglement propagation; however, noise accumulation in multi-step entanglement generation must be carefully controlled. Ultimately, distributed FTQC requires entanglement routing frameworks that account not only for the availability of entanglement resources, but also for their fidelity, coherence, and compatibility with the required logical operations [17].

C. Execution-time Error Containment and DQEC

Execution-time error containment focuses on measures applied during gate execution, syndrome extraction, mid-circuit measurement, reset, feedforward, and real-time system evolution. Although QECCs discretize and correct errors through repeated stabilizer measurements, their practical effectiveness depends on gate fidelities, syndrome extraction scheduling, decoding, and feedforward latency. Therefore, operation-time error containment depends not only on physical gate fidelity, but also on the temporal coordination of quantum operations, classical processing, and control feedback [3].

• **DQEC code design for distributed architectures**

The transition from MQC to DQC necessitates the redesign of FTQC primitives, particularly distributed QEC codes. In distributed settings, error suppression, syndrome extraction circuits, and correction mechanisms must be carefully partitioned and executed across multiple QPUs while preserving fault tolerance guarantees. This introduces a fundamentally new design paradigm in which code blocks can be implemented as fully local, fully distributed across several QPUs, or hybridized such that only selected stabilizer generators or logical operations require inter-QPU communication. Beyond code block distribution, distributed syndrome extraction and decoding further complicate system design. These challenges are exacerbated in large-scale systems, where increasing code complexity and stringent real-time error correction impose significant demands on communication, synchronization, and computational resources [210].

Despite recent progress, several limitations hinder the efficient deployment of DQECCs. A primary bottleneck is the implementation of non-local stabilizer measurements. Such measurements often require entanglement distribution, teleportation, or remote parity checks, all of which introduce additional error channels, latency, and resource overhead. Moreover, mismatches between code geometry and hardware connectivity can increase the number of remote operations, degrade load balancing, and reduce the effective threshold. These challenges become more severe in HeDQC systems, where different QPUs may have distinct noise models, operation latency, and measurement mechanisms. Moreover, distributed and hierarchical decoding and syndrome extraction scheduling remain unresolved challenges. Distributed and parallel decoders can reduce latency, but they must exchange boundary information among QPUs and maintain consistency across local decoding decisions. Machine learning decoders, including neural network and reinforcement learning approaches, may capture complex spatiotemporal error correlations and adapt to nontrivial noise models, but their generalization, interpretability, hardware efficiency, and code family dependence remain open issues [289].

These limitations also highlight several open challenges in DQEC code design. Future work on DQEC should focus on architecture-aware QECC constructions that explicitly account for QPU topology, interconnect constraints, communication latency, and decoding algorithms. Communication-efficient syndrome extraction scheduling must be developed to minimize non-local stabilizer measurements while preserving fault-tolerance thresholds. Adaptive qubit allocation, code partitioning, and load-balancing strategies can reduce inter-QPU traffic and improve resource utilization. Dynamic and reconfigurable QECCs, whose parameters or layouts adapt to hardware fea-

tures, noise bias, or workload requirements, are especially promising for heterogeneous architectures. Finally, distributed decoding must be co-designed with hardware accelerators such as FPGAs, GPUs, and ASICs to meet real-time feedback deadlines in large-scale FTQC systems [231].

D. Post-Processing and Error Mitigation

Post-processing and error mitigation include a class of classical techniques applied alongside or after quantum computations to suppress noise and extract more accurate results without requiring fully fault-tolerant circuits. These techniques rely on classical processing to approximate noise-free expectation values. These methods are valuable in near-term and early logical regimes, but their effectiveness depends strongly on accurate noise characterization. In practice, noise is dynamic due to calibration drift, temperature fluctuations, laser or microwave instability, crosstalk, and hardware instability. These challenges are further exacerbated in HeDQC, where different qubit modalities exhibit diverse noise characteristics, bias profiles, leakage mechanisms, measurement errors, and temporal correlations [121].

Scalable post-processing frameworks must therefore be adaptive and hardware-aware. They must capture correlated and time-dependent noise, integrate calibration data, and update mitigation strategies as device conditions change. In addition, post-processing must be implemented efficiently on classical hardware. FPGAs, GPUs, and ASICs may be required for real-time mitigation, decoding, filtering, or feedforward, but algorithms designed for CPUs do not always map efficiently onto specialized hardware. CPU-optimized algorithms often rely on irregular memory access, dynamic branching, and sequential control flow, whereas FPGA and ASIC implementations favor predictable data paths, parallelism, pipelining, and deterministic memory access. This mismatch imposes a major co-design challenge for real-time FTQC [227].

- **Hybrid classical–quantum systems (HCQSs)**

Quantum computing systems are inherently hybrid because quantum processors must operate in close coordination with classical computing resources. In this workflow, as illustrated in Figure 18, classical systems translate high-level user requests into executable quantum circuits, compile them into hardware-compatible instructions, schedule operations, generate pulse-level controls, perform calibration, process measurement outcomes, and issue feedback or corrective instructions. The quantum processors execute the circuit and returns measurement outcomes, which are then processed by classical resources to produce final results or to generate subsequent quantum instructions. The interaction is critical in *variational quantum algorithms* (VQAs) [318], where quantum executions are repeatedly interleaved with classical optimization[115], conditional instructions, real-time decoding, error mitigation, syndrome processing, feedforward control, and corrective actions. For instance, in superconducting architectures based on surface code, a syndrome stream may be generated on a microsecond scale, in which a decoder must process them and return corrections within roughly 10 cycles, i.e., 10 μs. These requirements impose strict constraints on processing speed, communication latency, bandwidth, and synchronization, as classical delays can directly degrade quantum fidelity and increase decoherence and the logical error rate.

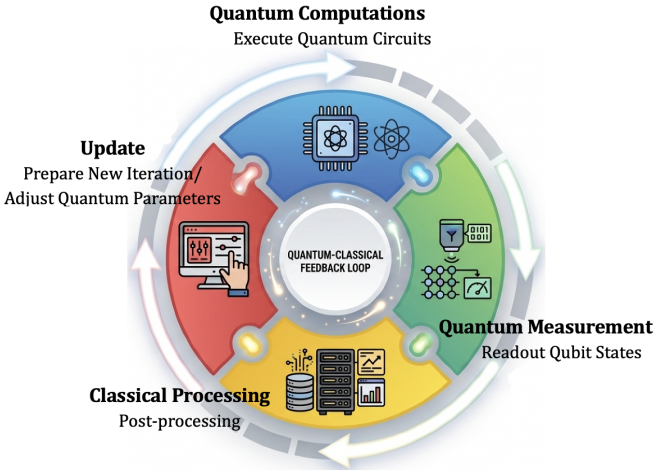


Fig. 18: The hybrid classical–quantum computing cycle.

Despite their importance, HCQSs introduce several system-level challenges. The mismatch between quantum and classical processing speeds, combined with conversion overhead and communication latency, can impose a classical bottleneck that limits scalability. Classical resources must coordinate QPUs that may differ in modality, calibration state, noise profile, connectivity, and timing constraints. Low-latency communication networks, HPC infrastructure, and parallel classical processing will be required to prevent classical control from becoming the bottleneck. Latency-aware compilation, precise synchronization, and incremental syndrome encoding[116] are promising directions for reducing overhead. Moreover, machine learning may also support noise-adaptive and bias-aware error mitigation, anomaly detection, workload prediction, approximate pre-processing, and hardware-aware decoder design. Beyond this, emerging agentic AI workflows may further enable autonomous orchestration of distributed classical–quantum pipelines, where backend selection, compilation, scheduling, monitoring, and post-processing are adapted to device status and application requirements. However, communication and computational overhead of the classical control stack of scalable HCQSs must scale approximately linearly with the number of QPUs, ensuring that classical processing does not become a bottleneck in large-scale FTQC [290].

- **Classical–quantum resource orchestration**

A related open issue is classical–quantum resource orchestration. Modern quantum systems are heterogeneous multi-node platforms that require protocols, APIs, schedulers, monitoring services, and resource management interfaces. In near-term deployments, many quantum processors are accessed through quantum cloud platforms, such as Amazon Braket, where users submit jobs through cloud APIs rather than

[115]For example, the variational quantum eigensolver (VQE) [318] can be utilized to estimate the ground-state energy of quantum systems, particularly molecular Hamiltonians.

[116]Transmitting incremental measurement updates rather than full measurement outcomes.

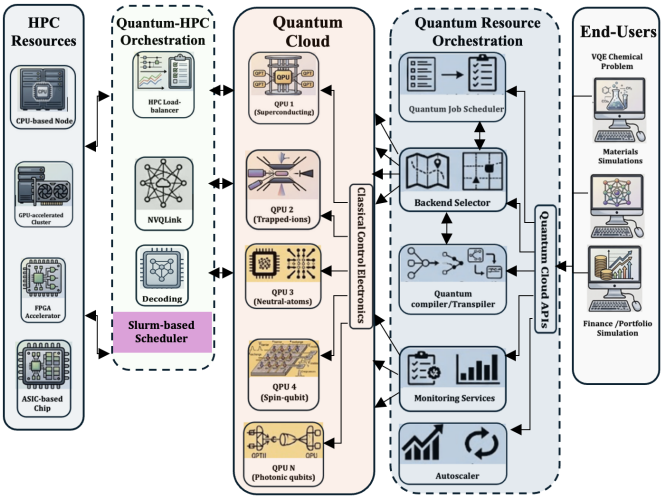


Fig. 19: A hybrid classical–quantum system and orchestration layers

interacting directly with hardware. As depicted in Figure 19, a HCQS can be viewed as a three-layer architecture, i.e., *end-users*, *heterogeneous quantum processors*, and *heterogeneous classical/HPC resources*, with two orchestration layers, i.e., *quantum resource orchestration* (QRO), which manages the interaction between users and quantum processors, and *quantum–HPC orchestration* (QHO), which manages the interaction between QPUs and classical acceleration resources [284].

End-users generate quantum jobs, which may include quantum circuit, shot counts, required gate set, desired logical error rate, output fidelity constraints, execution deadline, and user's budget constraints, and submit them to the quantum cloud through quantum cloud APIs. The QRO layer processes the received quantum tasks, selects an appropriate quantum backend according to the underlying quantum hardware features, processor availability, physical error rates, connectivity, gate fidelities, coherence times, queue length, and current workload, and provides the results back to the end-users. This interface may include quantum compilation/transpilation, backend selection, scheduling, service discovery, result aggregation, and security. During execution, monitoring services must track calibration data, noise levels, error rates, queue status, MCM data, and syndrome streams to detect execution anomalies and trigger corrective actions when needed [58].

Although early instances of QRO are emerging in cloud and industrial settings, such as Dell's quantum integration with QuEra, efficient QRO orchestration for heterogeneous quantum clouds remains immature. Open problems include splitting large quantum tasks across multiple processors, batching short jobs to reduce re-initialization and reconfiguration overhead, priority-aware scheduling, load balancing across heterogeneous backends, and quantum *autoscaling* that can adaptively allocate or release QPU resources based on monitoring feedback. Classical resource managers such as *Slurm* provide mature scheduling for heterogeneous HPC systems, but analogous orchestration tools for HCQSs remain at an early stage. Application-aware orchestration, where all QRO services can be application adaptive, is a particularly important research direction [163].

In HCQSs, each quantum processor may be incorporated with multiple HPC systems for post-processing purposes, either online or offline, which is managed by quantum–HPC orchestration. The QHO layer coordinates QPUs with CPUs, GPUs, FPGAs, ASICs, and other classical accelerators used for decoding, post-processing, and simulation. QHO must track quantum side features/requirements such as syndrome generation rate, feedback deadline, and quality-of-service (QoS) constraints, as well as classical side availability and accelerator-dependent features, such as throughput, latency, programmability, power consumption, and deployment complexity. Recent industrial efforts, including GPU-accelerated decoding algorithms, reflect the growing importance of QPU–HPC integration. Interfaces, such as NVIDIA's *NVQLink*, aim to connect QPUs with GPU-accelerated HPC systems through low-latency, high-throughput links[117]. In this context, QEC-ready platforms may require dedicated classical acceleration nodes tightly coupled to quantum control electronics and NVQLink-compatible control electronics to enable real-time processing and feedforward [79]. Similarly, emerging QPU–HPC integrations, developed by *Pasqal* in May 2026, based on neutral-atom QPUs, CUDA-Q, and Slurm-based [319] schedulers suggest that future quantum processors may increasingly be exposed as specialized accelerators within HPC environments.

E. Manufacturing and Engineering Constraints

The realization of FTQC is not limited by qubit quality alone; it also depends on manufacturability, packaging, control infrastructure, facility integration, reliability, and operational cost. While all modalities aim to improve coherence, gate fidelity, readout fidelity, and QEC compatibility, the dominant engineering bottlenecks differ substantially across platforms. These modality-specific constraints shape not only the physical implementation of quantum processors, but also their scalability, cost, system architecture, and suitability for large-scale deployment.

Superconducting quantum computers face a distinctive cryogenic infrastructure bottleneck. Large-scale superconducting and silicon-spin systems typically require dilution refrigerators, whose working fluid depends on scarce helium-3[118]. Cryogenic input/output density is another major constraint. Each qubit typically requires multiple RF/DC control and readout paths from room temperature electronics to the millikelvin cryostat. Conventional coaxial cables are bulky, thermally conductive, and mechanically rigid, making them difficult to scale to thousands or millions of qubits per cryostat. Emerging flexible cryogenic interconnects, such as Delft Circuits' Cri/oFlex technology, aim to increase channel density from the

[117]NVQLink, announced in October 2025, supports 17 QPU builders and nine U.S. national laboratories, and provides $\sim$ 400 Gb/s GPU–QPU throughput and round-trip latency of 3.96 μs.

[118]A dilution refrigerator holds approximately 40 liters of helium-3 that volume of helium-3 is worth roughly $100,000. Helium-3 is produced primarily from the radioactive decay of tritium in nuclear weapons stockpiles, while *Interlune* planned for lunar-regolith-extracted helium-3 in the early 2030s.

current 256 channels to 4,096 channels by 2029, but low loss, low thermal conductance, and high-density wiring remains a central engineering challenge for beyond small- and medium-scale processors. Superconducting systems also impose facility level requirements, including vibration isolation[119], chilled-water support for cryogenic compressors, magnetic shielding, reinforced flooring[120], and reliable backup power. Thermal excursions can require long recovery times, affecting availability and operational cost[121]. Therefore, standardized quantum infrastructure requirements and data center remediation strategies will be necessary for large-scale deployment of superconducting FTQC systems [62].

Trapped-ion systems typically avoid many millikelvin cryogenic constraints, although some platforms may still employ cryogenic operation to suppress thermal noise, electric field noise, and improve vacuum conditions. Their dominant engineering challenges instead arise from optical control complexity, motional mode scalability, vacuum reliability, and modular interconnects. As the number of ions in a single chain increases, the collective motional spectrum becomes denser, ion spacing decreases, crosstalk increases, and structural instabilities, such as zig-zag transitions, become more likely [177]. Consequently, scalable trapped-ion architectures are unlikely to rely on a single long ion chain as a global register. Future systems will require segmented traps, localized motional modes, dynamically reconfigurable optical potentials, or modular trap arrays. At the same time, trapped-ion control stacks require stable lasers, modulators, imaging systems, beam delivery, stabilization cavities, and zone-selective addressing. Integrated photonics, chip-scale lasers, and electronic or RF-based control may reduce optical complexity, but these technologies remain active engineering challenges. Moreover, scaling beyond a single trap or wafer is still unresolved, since current inter-module entanglement fidelities and effective rates remain below those of local trapped-ion gates. A promising direction is hierarchical HeDQC, in which deterministic matter links connect nearby modules, while photonic interconnects provide longer-range and reconfigurable connectivity [35].

Neutral-atom systems replace cryogenic bottlenecks with challenges in deterministic loading, atom loss, optical stack scaling, crosstalk, Rydberg gate fidelity, and readout latency. Many neutral-atom platforms begin with stochastic atom loading, requiring re-loading procedures to form defect-free arrays, while the loading probability is around 50%. Atom loss due to vacuum events, imaging-induced loss, transport, or gate operations is a first-order system concern. Therefore, atom reservoirs, ancilla replacement, loss-aware compilation, and continuous reloading are important for long-duration FTQC [105]. Similar to trapped-ion systems, neutral-atom processors rely heavily on optical infrastructure. Therefore, several academic and industrial roadmaps now emphasize integrated photonics, fiber array delivery, MEMS (micro-electro-mechanical systems)- or SLM (spatial light modulator)-based beam control, and closed-loop calibration to reduce optical complexity and address crosstalk. In addition, neutral-atom processors may become readout-bound, in which the total latency of measurement, decoding, feedforward, and reset limits the QEC cycle rate. Promising directions include zoned readout, non-destructive and loss resolved measurement, cavity enhanced readout, and hardware-accelerated classical control [150].

Spin-qubit platforms share some engineering constraints with superconducting quantum computers, including cryogenic operation, dense electrical control, microwave or RF signaling, packaging complexity, and the need for close integration with classical control electronics. However, spin qubits should not be treated as miniature superconducting qubits. Spin-qubit hardware comprises multiple families, including *Si MOS*, *Si/SiGe quantum dots*, *GaAs quantum dots*, and *donor spins in silicon*, each with distinct fabrication and materials challenges. Scalable spin-qubit systems require not only high isolated gate fidelities, but also wafer-scale process control, low disorder heterostructures, isotopic purification, and automated calibration across large arrays [57]. Similar to the superconducting modality, cryogenic I/O density remains a central constraint, but the mitigation path differs from superconducting systems. Spin-qubit architectures motivate stronger integration with cryo-CMOS controllers, multiplexed gate biasing, crossbar addressing, three-dimensional integration, flip-chip bonding, and thermally isolated control planes to reduce wiring overhead. Moreover, although high-fidelity spin-to-charge conversion has been demonstrated in small devices, compact, multiplexed, QEC-compatible readout with low latency and limited chip-area overhead remains challenging [148].

These engineering challenges highlight that the path to FTQC is fundamentally modality dependent. Manufacturing, packaging, control electronics, optical infrastructure, cryogenic systems, and facility level integration will likely dominate practical scalability alongside improvements in qubit fidelity and QECC performance. Therefore, answering *how to build FTQC* requires not only better qubits and better QECCs, but also modality-aware engineering roadmaps that integrate devices, control stack, packaging, infrastructure, orchestration, and error correction requirements from the beginning.

X. Conclusion

Large-scale fault-tolerant quantum computing is not merely a matter of increasing the number of physical qubits or designing more effective QECCs; rather, it is a full-stack engineering and orchestration challenge involving DQEC, HeDQC, real-time decoding, QPU–HPC integration, and modality-dependent implementation constraints. In this paper, we have explored the realization of FTQC through four main phases, i.e., fault-avoidance techniques, fault-tolerant universal gate set implementations, operation-time measures, and post-processing strategies. We first reviewed pre-execution and hardware-level strategies associated with the design, development, and configuration of quantum systems, which aim to

[119] Cryostats should be isolated from mechanical vibration sources such as elevators, trains, compressors, and freight docks.

[120] A fully loaded cryostat can weigh approximately 750 kilograms concentrated on a small footprint.

[121] If a grid failure or cooling interruption drives the cryostat above its low-temperature operating regime, recovery may take five to ten days because the system must be cooled down gradually.

suppress imperfections before they occur or propagate during computation. We then discussed operation-time measures, including efficient QECC construction, syndrome scheduling, real-time and HPC-accelerated decoding, and hybrid classical–quantum co-processing.

We comprehensively surveyed major QECC families, including topological, qLDPC, subsystem, and dynamical codes, together with a broad range of decoding algorithms, including matching-based, clustering and graph-growth, probabilistic-inference, tensor network, machine learning, cellular automata, and optimization-based decoders. We also reviewed fault-tolerant logical operations, including code deformation, lattice surgery, code switching, gauge fixing, magic state injection, along with distillation and cultivation techniques. All the aforementioned topics were examined from both academic theoretical achievements and industrial experimental demonstrations across major qubit modalities, which clarify the current status of leading quantum companies and their near- and long-term roadmaps.

Beyond QEC and gate-level considerations, we highlighted DQC-enabled strategies, heterogeneous DQC, classical–quantum orchestration layers, and modality-specific engineering constraints, including cryogenic infrastructure, optical control, packaging, manufacturability, and deployment requirements. We further compared benchmarking methodologies, simulation tools, frameworks, and libraries available for evaluating designs relevant to large-scale FTQC. Finally, we identified open theoretical, architectural, technological, and practical challenges and outlined promising research directions. These challenges indicate that the path toward utility-scale FTQC will require sustained collaboration among academic institutions, industry, and government-supported programs, although government investments, such as the U.S. $2 billion *CHIPS and Science Act* funding announced in May 2026, are expected to further accelerate this progress.

REFERENCES

[1] M. Iqbal, N. Tantivasadakarn, T. M. Gatterman, *et al.*, "Topological order from measurements and feed-forward on a trapped ion quantum computer," *Communications Physics*, vol. 7, pp. 1–8, 2024.

[2] Y. Zhao, G. Zhao, and C. Qiao, "E2e fidelity aware routing and purification for throughput maximization in quantum networks," in *IEEE Conference on Computer Communications (INFOCOM)*. Piscataway, NJ, USA: IEEE, 2022, pp. 480–489.

[3] K. Sen and M. A. Martin-Delgado, "Homomorphic quantum error correction," 2026. [Online]. Available: https://arxiv.org/abs/2605.25692

[4] E. Campbell, "A series of fast-paced advances in quantum error correction," *Nature Reviews Physics*, vol. 6, no. 3, pp. 160–161, 2024.

[5] V. S. Thakur, A. Kumar, and K. Dev, "Quantum error correction in green communications: Toward sustainable quantum networks," *IEEE Transactions on Green Communications and Networking*, vol. 9, no. 4, pp. 1566–1576, 2025.

[6] V. S. Thakur, A. Kumar, J. Das, K. Dev, and M. Magarini, "Quantum error correction codes in consumer technology: Modeling and analysis," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 4, pp. 7102–7111, 2024.

[7] I. Mahmud and A. Abdelhadi, "Quantum codes: A comprehensive survey of techniques, challenges, and future directions," *IEEE Access*, vol. 13, pp. 214 352–214 386, 2025.

[8] V. Karthick, M. R. Prabhu, P. Nancy, A. Devipriya, and R. A. A. Rosaline, "Advancements in fault-tolerant quantum error correction for current progress and future directions," in *2nd International Conference on Advances in Information Technology (ICAIT)*. Piscataway, NJ, USA: IEEE, 2024, pp. 1–5.

[9] S. Babaie, S. Grzenda, and C. Qiao, "Distributed quantum error correction: Advancements and future research directions," in *24th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*. Piscataway, NJ, USA: IEEE, 2026, pp. 1–8.

[10] A. Chatterjee, K. Phalak, and S. Ghosh, " Quantum error correction for dummies ," in *IEEE International Conference on Quantum Computing and Engineering (QCE)*. Piscataway, NJ, USA: IEEE, 2023, pp. 70–81.

[11] H. Riel, "Toward large-scale fault-tolerant quantum computing," in *IEEE International Solid-State Circuits Conference (ISSCC)*. Piscataway, NJ, USA: IEEE, 2026, pp. 16–22.

[12] S. Venkatesha and R. Parthasarathi, "Survey on redundancy based-fault tolerance methods for processors and hardware accelerators - trends in quantum computing, heterogeneous systems and reliability," *ACM Computing Surveys*, vol. 56, no. 11, 2024.

[13] H. T. Larasati and B.-S. Choi, "Towards fault-tolerant distributed quantum computation (FT-DQC): Taxonomy, recent progress, and challenges," *ICT Express*, vol. 11, no. 3, pp. 417–435, 2025.

[14] V. Erol, "A comprehensive review of recent progress in quantum error correction: Codes, decoders, and fault-tolerant architectures," Sep. 2025, version 2, preprint, not peer reviewed. [Online]. Available: https://doi.org/10.20944/preprints202509.1037.v2

[15] V. S. Thakur, A. Kumar, M. Magarini, K. Dev, and O. A. Dobre, "Quantum communication and information technologies: A survey on foundation, error correction, NISQ, and networks," Sep. 2025, preprint. [Online]. Available: https://www.techrxiv.org/doi/10.36227/techrxiv.175735780.05700466/v1

[16] S. L. Deepika and D. Ch, "A review on recent advances in quantum error correction: From theory to practical applications," *EPJ Web of Conferences*, vol. 360, p. 01023, 2026.

[17] X. Wang, Y. Zhao, H. Xu, C. Tian, K. Yang, and C. Qiao, "Ai-powered persistent entanglement distribution in quantum networks," *IEEE Transactions on Networking*, vol. 34, pp. 5772–5787, 2026.

[18] D. Khu, A. Tanggara, C. Jin, and K. Bharti, "Contextuality of quantum error-correcting codes," *PRX Quantum*, vol. 7, pp. 1–29, 2026.

[19] H. Xie, N. Yoshioka, K. Tsubouchi, and Y. Li, "Noise-agnostic unbiased quantum error mitigation for logical qubits," *Physical Review Letters*, vol. 136, pp. 1–9, 2026.

[20] J. F. Marques, B. M. Varbanov, M. S. Moreira *et al.*, "Logical-qubit operations in an error-detecting surface code," *Nature Physics*, vol. 18, no. 1, pp. 80–86, 2022.

[21] X. Wang, Y. Zhao, H. Xu, C. Tian, K. Yang, and C. Qiao, "Social utility maximization via entanglement connection provisioning in quantum networks," *IEEE Journal on Selected Areas in Communications*, vol. 44, pp. 4718–4732, 2026.

[22] Y. Zhao, Y. Wang, E. Wang, H. Xu, L. Huang, and C. Qiao, "An asynchronous transport protocol for quantum data networks," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 7, pp. 1885–1899, 2024.

[23] A. Potočnik, "How to scale the electronic control systems of a quantum computer," *Nature Electronics*, vol. 8, pp. 3–4, 2025.

[24] R. S. Gupta, N. Sundaresan, T. Alexander *et al.*, "Encoding a magic state with beyond break-even fidelity," *Nature*, vol. 625, no. 7994, pp. 259–263, 2024.

[25] X. Wang, Y. Zhao, E. Wang, C. Tian, K. Yang, and C. Qiao, "Deep reinforcement learning-based deferred entanglement path selection in quantum networks," *IEEE Transactions on Networking*, vol. 34, pp. 4668–4683, 2026.

[26] D. Barral, F. J. Cardama, G. Díaz-Camacho *et al.*, "Review of distributed quantum computing: From single qpu to high performance quantum computing," *Computer Science Review*, vol. 57, p. 100747, 2025.

[27] S. Gupta, S. Bhambay, N. Alavisamani, N. Walton, and T. Vasantam, "Boundary-aware stabilizer scheduling for distributed quantum error correction," 2026. [Online]. Available: https://arxiv.org/abs/2604.22471

[28] M. G. Garces, C. Heunen, and M. Marina, "Distributed quantum computing across heterogeneous hardware with hybrid dependency hypergraphs," in *Proceedings of the ACM SIGCOMM Posters and Demos*. New York, NY, USA: Association for Computing Machinery, 2025, pp. 49–51.

[29] M. B. Hastings and J. Haah, "Dynamically generated logical qubits," *Quantum*, vol. 5, p. 564, 2021.

[30] S.-H. Lee, A. Li, and S. D. Bartlett, "Color code decoder with improved scaling for correcting circuit-level noise," *Quantum*, vol. 9, p. 1609, 2025.

[31] Y. Hu, C. Zheng, X. Wang, F. Meng, X. Yu, and Z. Zhang, "Suppressing quantum errors by noise-aware circuit design," *Quantum Science and Technology*, vol. 10, no. 4, p. 045040, 2025.

[32] B. C. Sanders, "The success and failure of quantum computing start-ups," *Nature Electronics*, vol. 8, pp. 5–7, 2025.

[33] Z.-M. Li and Y.-x. Liu, "Efficient preparation of decoherence-free subspace basis states," *Physical Review A*, vol. 113, p. 012410, 2026.

[34] M. Ringbauer, M. Hinsche, T. Feldker *et al.*, "Verifiable measurement-based quantum random sampling with trapped ions," *Nature Communications*, vol. 16, no. 1, pp. 1–9, 2025.

[35] F. Butt, I. Pogorelov, R. Freund, A. Steiner, M. Meyer, T. Monz, and M. Müller, "Demonstration of measurement-free universal logical quantum computation," *Nature Communications*, vol. 17, no. 1, p. 995, 2026.

[36] N. Delfosse and E. Tham, "Low-cost noise reduction for clifford circuits," *Physical Review Letters*, vol. 134, pp. 1–8, 2025.

[37] S. Babaie and C. Qiao, "Towards distributed quantum error correction for distributed quantum computing," in *IEEE International Conference on Quantum Communications, Networking, and Computing (QCNC)*. Piscataway, NJ, USA: IEEE, 2025, pp. 66–73.

[38] B. L. Brock, S. Singh, A. Eickbusch, V. V. Sivak, A. Z. Ding, L. Frunzio, S. M. Girvin, and M. H. Devoret, "Quantum error correction of qudits beyond break-even," *Nature*, vol. 641, no. 8063, pp. 612–618, 2025.

[39] K.-Y. Kuo and Y. Ouyang, "Degenerate quantum erasure decoding," *npj Quantum Information*, vol. 12, no. 1, pp. 1–14, 2026.

[40] N. Lacroix, L. Hofele, A. Remm *et al.*, "Fast flux-activated leakage reduction for superconducting quantum circuits," *Physical Review Letters*, vol. 134, pp. 1–8, 2025.

[41] M. N. H. Chow, V. Buchemmavari, S. Omanakuttan, B. J. Little, S. Pandey, I. H. Deutsch, and Y.-Y. Jau, "Circuit-based leakage-to-erasure conversion in a neutral-atom quantum processor," *PRX Quantum*, vol. 5, pp. 1–14, 2024.

[42] J. F. Marques, H. Ali, B. M. Varbanov *et al.*, "All-microwave leakage reduction units for quantum error correction with superconducting transmon qubits," *Physical Review Letters*, vol. 130, pp. 1–6, 2023.

[43] J. Camps, O. Crawford, G. P. Gehér, A. V. Gramolin, M. P. Stafford, and M. Turner, "Leakage mobility in superconducting qubits as a leakage reduction unit," *Physica Scripta*, vol. 101, no. 11, p. 115107, 2026.

[44] K. C. Miao, M. McEwen, J. Atalaya *et al.*, "Overcoming leakage in quantum error correction," *Nature Physics*, vol. 19, pp. 1780–1786, 2023.

[45] J. Zhang, S. H. Cantú, F. Liu *et al.*, "Probing quantum floating phases in rydberg atom arrays," *Nature Communications*, vol. 16, no. 1, p. 712, 2025.

[46] M. Iqbal, A. Lyons, C. F. B. Lo *et al.*, "Qutrit toric code and parafermions in trapped ions," *Nature Communications*, vol. 16, no. 1, p. 6301, 2025.

[47] M. Meth, J. Zhang, and J. F. o. Haase, "Simulating two-dimensional lattice gauge theories on a qudit quantum computer," *Nature Physics*, vol. 21, pp. 570–576, 2025.

[48] D. Gottesman, "Stabilizer codes and quantum error correction," 1997. [Online]. Available: https://arxiv.org/abs/quant-ph/9705052

[49] S. Babaie, S. Grzenda, and C. Qiao, "Need one bell-pair only (nobol) for low-overhead fault-tolerant quantum computing," in *IEEE International Conference on Quantum Computing and Engineering (QCE)*. Piscataway, NJ, USA: IEEE, 2026.

[50] D. Gottesman, "Surviving as a quantum computer in a classical world," 2024, 2024 draft textbook, University of Maryland.

[51] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Physical Review A*, vol. 54, pp. 1098–1105, 1996.

[52] V. Guemard, "Lifts of quantum CSS codes," *IEEE Transactions on Information Theory*, vol. 71, no. 7, pp. 5418 – 5442, 2025.

[53] S. Babaie and C. Qiao, "Distributed quantum bit-phase-flip error correction based on css codes and ghz states," in *IEEE 45th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. Piscataway, NJ, USA: IEEE, 2025, pp. 195–200.

[54] S. Bravyi, A. W. Cross, J. M. Gambetta, D. Maslov, P. Rall, and T. J. Yoder, "High-threshold and low-overhead fault-tolerant quantum memory," *Nature*, vol. 627, no. 8005, pp. 778–782, 2024.

[55] A. Angrisani, A. Schmidhuber, M. S. Rudolph, M. Cerezo, Z. Holmes, and H.-Y. Huang, "Classically estimating observables of noiseless quantum circuits," *Physical Review Letters*, vol. 135, p. 170602, 2025.

[56] W. C. Chung, D. C. Cole, P. Gokhale *et al.*, "Fault-tolerant operation and materials science with neutral atom logical qubits," *npj Quantum Information*, vol. 11, no. 1, pp. 1–15, 2025.

[57] S. K. Bartee, W. Gilbert, K. Zuo *et al.*, "Spin-qubit control with a milli-kelvin cmos chip," *Nature*, vol. 643, no. 382-387, 2025.

[58] H. Zhou, C. Zhao, M. Cain *et al.*, "Low-overhead transversal fault tolerance for universal quantum computation," *Nature*, vol. 646, no. 8084, pp. 303–308, 2025.

[59] Z. Wang and H. Tang, "Artificial intelligence for quantum error correction: A comprehensive review," 2024, arXiv preprint.

[60] Z. Jia, "Quantum logic with bosonic error correction," *Nature Physics*, vol. 21, no. 10, pp. 1528–1529, 2025.

[61] N. Lacroix, A. Bourassa, F. J. H. Heras *et al.*, "A qubit update," *Nature Electronics*, vol. 8, pp. 547–547, 2025.

[62] A. D. King, A. Nocera, M. M. Rams *et al.*, "Beyond-classical computation in quantum simulation," *Science*, vol. 388, no. 6743, pp. 199–204, 2025.

[63] K. Alexander, A. Benyamini, D. Black *et al.*, "A manufacturable platform for photonic quantum computing," *Nature*, vol. 641, pp. 876–883, 2025.

[64] C. Gidney, "How to factor 2048 bit RSA integers with less than a million noisy qubits," 2025. [Online]. Available: https://arxiv.org/abs/2505.15917

[65] M. Cain, Q. Xu, R. King *et al.*, "Shor's algorithm is possible with as few as 10,000 reconfigurable atomic qubits," 2026. [Online]. Available: https://arxiv.org/abs/2603.28627

[66] A. Chakraborty and D. Gottesman, "No-go theorem on fault tolerant gadgets for multiple logical qubits," 2026. [Online]. Available: https://arxiv.org/pdf/2602.13395v1

[67] A. N. Craddock, T. Cowan, N. Bigagli *et al.*, "High-rate scalable entanglement swapping between remote entanglement sources on deployed new york city fibers," 2026. [Online]. Available: https://arxiv.org/abs/2602.15653

[68] Riverlane. (2025) The quantum error correction report 2025. [Online]. Available: https://www.riverlane.com/quantum-error-correction-report-2025

[69] S. Bravyi and A. Kitaev, "Universal quantum computation with ideal clifford gates and noisy ancillas," *Physical Review A*, vol. 71, p. 022316, 2005.

[70] B. Eastin and E. Knill, "Restrictions on transversal encoded quantum gate sets," *Physical Review Letters*, vol. 102, no. 11, p. 110502, 2009.

[71] D. B. Tan, M. Y. Niu, and C. Gidney, " A SAT Scalpel for Lattice Surgery: Representation and Synthesis of Subroutines for Surface-Code Fault-Tolerant Quantum Computing ," in *ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. New York, NY, USA: Association for Computing Machinery, 2024, pp. 325–339.

[72] E. Weilandt, T. Peham, and R. Wille, "Minimizing the number of code switching operations in fault-tolerant quantum circuits," 2025. [Online]. Available: https://arxiv.org/abs/2512.04170

[73] A. Paetznick and B. W. Reichardt, "Universal fault-tolerant quantum computation with only transversal gates and error correction," *Physical Review Letters*, vol. 111, p. 090505, 2013.

[74] R. Ismail, I.-C. Chen, C. Zhao, R. Weiss *et al.*, "Transversal architecture for megaquop-scale quantum simulation with neutral atoms," *PRX Quantum*, vol. 7, p. 020343, 2026.

[75] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A*, vol. 86, no. 3, p. 032324, 2012.

[76] D. Litinski, "A game of surface codes: large-scale quantum computing with lattice surgery," *Quantum*, vol. 3, pp. 1–37, 2019.

[77] C. Gidney and M. Ekerå, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits," *Quantum*, vol. 5, p. 433, 2021.

[78] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the Spring Joint Computer Conference*. New York, NY, USA: Association for Computing Machinery, 1967, p. 483–485.

[79] Y. Kim, M. Sevior, and M. Usman, "Magic state injection on IBM quantum processors above the distillation threshold," *Scientific Reports*, vol. 16, p. 11189, 2026.

[80] D. Ruiz, J. Guillaud, C. Vuillot, and M. Mirrahimi, "Unfolded distillation: very low-cost magic state preparation for biased-noise qubits," *npj Quantum Information*, vol. 12, no. 1, p. 53, 2026.

[81] K. H. Wan, Z. Zhong, and A. Zapirain, "Simulating magic state cultivation with few Clifford terms," *Quantum*, vol. 10, p. 2134, 2026.

[82] P. Sales Rodriguez, J. M. Robinson, P. N. Jepsen *et al.*, "Experimental demonstration of logical magic state distillation," *Nature*, vol. 645, no. 8081, pp. 620–625, 2025.

[83] K. H. Wan and Z. Zhong, "Pauli web of the $|y\rangle$ state surface code injection," 2025. [Online]. Available: https://arxiv.org/abs/2501.15566

[84] Z.-H. Chen, M.-C. Chen, C.-Y. Lu, and J.-W. Pan, "Efficient magic state cultivation on $\mathbb{RP}^2$," *PRX Quantum*, vol. 7, p. 010315, 2026.

[85] L. Daguerre, R. Blume-Kohout, N. C. Brown, D. Hayes, and I. H. Kim, "Experimental demonstration of high-fidelity logical magic states from code switching," *Physical Review X*, vol. 15, p. 041008, 2025.

[86] S. Jacoby, Y. Vaknin, A. Retzker, and A. L. Grimsmo, "Magic state injection with erasure qubits," *PRX Quantum*, vol. 6, p. 040323, 2025.

[87] C. Gidney, N. Shutty, and C. Jones, "Magic state cultivation: growing T states as cheap as cnot gates," 2024. [Online]. Available: https://arxiv.org/abs/2409.17595

[88] Y. Li, "A magic state's fidelity can be superior to the operations that created it," *New Journal of Physics*, vol. 17, no. 2, p. 023037, 2015.

[89] D. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, "Surface code quantum computing by lattice surgery," *New Journal of Physics*, vol. 14, no. 12, p. 123011, 2012.

[90] D. Litinski, "Magic state distillation: not as costly as you think," *Quantum*, vol. 3, p. 205, 2019.

[91] H. Bombín, M. Pant, S. Roberts, and K. I. Seetharam, "Fault-tolerant postselection for low-overhead magic state preparation," *PRX Quantum*, vol. 5, p. 010302, 2024.

[92] Y. Hirano, T. Itogawa, and K. Fujii, "Leveraging zero-level distillation to generate high-fidelity magic states," in *IEEE International Conference on Quantum Computing and Engineering (QCE)*. Piscataway, NJ, USA: IEEE, 2024, pp. 843–853.

[93] Y. Vaknin, S. Jacoby, A. Grimsmo, and A. Retzker, "High rate magic state cultivation on the surface code," *PRX Quantum*, vol. 7, p. 010353, 2026.

[94] S. Wu, D. Zhong, T. A. Brun, and D. A. Lidar, "Universal weakly fault-tolerant quantum computation via code switching in the [[8,3,2]] code," 2026. [Online]. Available: https://arxiv.org/abs/2603.15610

[95] F. Butt, S. Heußen, M. Rispler, and M. Müller, "Fault-tolerant code-switching protocols for near-term quantum processors," *PRX Quantum*, vol. 5, p. 020345, 2024.

[96] S. J. S. Tan, Y. Hong, T.-C. Lin, M. J. Gullans, and M.-H. Hsieh, "Single-shot universality in quantum ldpc codes via code-switching," 2025. [Online]. Available: https://arxiv.org/abs/2510.08552

[97] Y. Ouyang, Y. Jing, and G. K. Brennen, "Measurement-free code-switching protocol for low-overhead quantum computation using permutation-invariant codes," *PRX Quantum*, vol. 6, p. 040341, 2025.

[98] N. Lacroix, A. Bourassa, F. J. H. Heras *et al.*, "Scaling and logic in the colour code on a superconducting quantum processor," *Nature*, vol. 645, p. 614–619, 2025.

[99] I. Pogorelov, F. Butt, L. Postler, C. D. Marciniak, P. Schindler, M. Müller, and T. Monz, "Experimental fault-tolerant code switching," *Nature Physics*, vol. 21, no. 2, pp. 298–303, 2025.

[100] A. Márton, L. Colmenarez, L. Bödeker, and M. Müller, "Lattice surgery-based logical state teleportation via noisy links," *Physical Review Research*, vol. 7, p. 033238, 2025.

[101] K. Hamada, Y. Suzuki, and Y. Tokunaga, "Efficient and high-performance routing of lattice-surgery paths on three-dimensional lattice," *Quantum*, vol. 10, p. 2061, 2026.

[102] S. Yarkoni, C. Scheim, D. Hakshuri, and N. Katz, "The pangaea architecture: Fault-tolerant heterogeneous topological codes via a quantum bus," 2026. [Online]. Available: https://arxiv.org/abs/2608.01887

[103] I. Besedin, M. Kerschbaum, J. Knoll *et al.*, "Lattice surgery realized on two distance-three repetition codes with superconducting qubits," *Nature Physics*, vol. 22, no. 2, pp. 189–194, 2026.

[104] H. Leone, T. Le, S. Srikara, and S. Devitt, "Resource overheads and attainable rates for trapped-ion lattice surgery," *Physical Review Research*, vol. 7, p. 023088, 2025.

[105] A. Radnaev, W. Chung, D. Cole *et al.*, "Universal neutral-atom quantum computer with individual optical addressing and nondestructive readout," *PRX Quantum*, vol. 6, p. 030334, 2025.

[106] K. R. Ott, B. Hetényi, and M. E. Beverland, "Decision-tree decoders for general quantum ldpc codes," 2025. [Online]. Available: https://arxiv.org/abs/2502.16408

[107] V. Tripathi, N. Goss, A. Vezvaee, L. B. Nguyen, I. Siddiqi, and D. A. Lidar, "Qudit dynamical decoupling on a superconducting quantum processor," *Physical Review Letters*, vol. 134, p. 050601, 2025.

[108] W. Jang, J. Kim, J. Park *et al.*, "True decoherence-free-subspace derived from a semiconductor double quantum dot heisenberg spin-trimer," *npj Quantum Information*, vol. 12, no. 1, p. 1, 2025.

[109] C. Luo, H. Zhang, A. Chu, C. Maruko, A. M. Rey, and J. K. Thompson, "Hamiltonian engineering of collective XYZ spin models in an optical cavity," *Nature Physics*, vol. 21, no. 6, pp. 916–923, 2025.

[110] A. Vezvaee, V. Tripathi, M. Morford-Oberst, F. Butt, V. Kasatkin, and D. A. Lidar, "Demonstration of high-fidelity entangled logical qubits using transmons," *Nature Communications*, vol. 17, p. 3281, 2026.

[111] L. Hour, M. Go, and Y. Han, "Improving zero-noise extrapolation for quantum-gate error mitigation using a noise-aware folding method," *IEEE Access*, vol. 14, pp. 37 362–37 372, 2026.

[112] S. H. Park, G. Choi, E. Kim, G. Park, J. Choi, J. Choi, Y. Chong, Y.-H. Lee, and S. Hahn, "Kinetic-inductance-incorporated quantization for accurate hamiltonian prediction in superconducting circuits," *npj Quantum Information*, vol. 12, p. 58, 2026.

[113] K. Chiv, L. Hour, S. Lee, T. Kit, T.-K. Kim, and Y. Han, "Strategies for noise-resilient quantum approximate optimization algorithms: A review and classification of error mitigation," *IEEE Access*, vol. 13, pp. 216 916–216 936, 2025.

[114] M. G. Garces, "Distributed quantum error mitigation: Global and local zne encodings," in *IEEE Conference on Computer Communications (INFOCOM)*. Piscataway, NJ, USA: IEEE, 2026, p. 1–6.

[115] T. Scheiber, P. Haubenwallner, and M. Heller, "Reduced Sampling Overhead for Probabilistic Error Cancellation by Pauli Error Propagation," *Quantum*, vol. 9, p. 1840, 2025.

[116] S. N. Filippov, S. Maniscalco, and G. García-Pérez, "Scalability of quantum error mitigation techniques: from utility to advantage," 2024. [Online]. Available: https://arxiv.org/abs/2403.13542

[117] E. Van den Berg and P. Wocjan, "Techniques for learning sparse Pauli-Lindblad noise models," *Quantum*, vol. 8, p. 1556, 2024.

[118] A. Zhang, H. Xie, Y. Gao *et al.*, "Demonstrating quantum error mitigation on logical qubits," *Nature Communications*, vol. 17, no. 1, p. 1021, 2025.

[119] T.-R. Jin, Y.-R. Zhang, K. Xu, and H. Fan, "Noisy probabilistic error cancellation and generalized physical implementability," *Communications Physics*, vol. 8, no. 1, p. 296, 2025.

[120] P. Bermejo, P. Braccia, M. S. Rudolph, Z. Holmes, L. Cincio, and M. Cerezo, "Quantum convolutional neural networks are effectively classically simulable," *PRX Quantum*, vol. 7, p. 020304, 2026.

[121] A. Eddins, M. C. Tran, and P. Rall, "Lightcone shading for classically accelerated quantum error mitigation," 2024. [Online]. Available: https://arxiv.org/abs/2409.04401

[122] G. Torlai, C. J. Wood, A. Acharya, G. Carleo, J. Carrasquilla, and L. Aolita, "Quantum process tomography with unsupervised learning and tensor networks," *Nature Communications*, vol. 14, no. 1, p. 2858, 2023.

[123] S. Mangini and D. Cavalcanti, "Low variance estimations of many observables with tensor networks and informationally-complete measurements," *Quantum*, vol. 9, p. 1812, 2025.

[124] E. van den Berg, Z. K. Minev, A. Kandala, and K. Temme, "Probabilistic error cancellation with sparse pauli–lindblad models on noisy quantum processors," *Nature Physics*, vol. 19, no. 8, pp. 1116–1121, 2023.

[125] J. F. Martin, G. Cocco, and J. Fonollosa, "Quantum error mitigation at the pre-processing stage," 2026. [Online]. Available: https://arxiv.org/abs/2602.05916

[126] A. Berezutskii, M. Liu, A. Acharya *et al.*, "Tensor networks for quantum computing," *Nature Reviews Physics*, vol. 7, no. 10, pp. 581–593, 2025.

[127] L. Valentini, D. Forlivesi, and M. Chiani, "Performance limits of fault-tolerant quantum error correction schemes," *IEEE Journal on Selected Areas in Communications*, vol. 44, pp. 4704–4717, 2026.

[128] D. Gottesman, "Opportunities and challenges in fault-tolerant quantum computation," 2022. [Online]. Available: https://arxiv.org/abs/2210.15844

[129] P. Aliferis, D. Gottesman, and J. Preskill, "Quantum accuracy threshold for concatenated distance-3 codes," *Quantum Information & Computation*, vol. 6, no. 2, p. 97–165, 2006.

[130] A. W. Cross, D. P. Divincenzo, and B. M. Terhal, "A comparative code study for quantum fault tolerance," *Quantum Information & Computation*, vol. 9, no. 7, p. 541–572, 2009.

[131] D. Aharonov and M. Ben-Or, "Fault-tolerant quantum computation with constant error," in *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*. New York, NY, USA: Association for Computing Machinery, 1997, p. 176–188.

[132] S. Bravyi and R. König, "Classification of topologically protected gates for local stabilizer codes," *Nature Communications*, vol. 4, p. 2007, 2013.

[133] D. Gottesman, "Fault-tolerant quantum computation with constant overhead," 2014. [Online]. Available: https://arxiv.org/abs/1310.2984

[134] S. Bravyi and B. Terhal, "A no-go theorem for a two-dimensional self-correcting quantum memory based on stabilizer codes," *New Journal of Physics*, vol. 11, no. 4, p. 043029, 2009.

[135] S. Bravyi, D. Poulin, and B. Terhal, "Tradeoffs for reliable quantum information storage in 2d systems," *Physical Review Letters*, vol. 104, p. 050503, 2010.

[136] E. Knill and R. Laflamme, "Theory of quantum error-correcting codes," *Physical Review A*, vol. 55, pp. 900–911, 1997.

[137] A. Ekert and C. Macchiavello, "Quantum error correction for communication," *Physical Review Letters*, vol. 77, pp. 2585–2588, 1996.

[138] C. Yuan and R. Zhu, "Improvement of the gilbert-varshamov bound for linear codes and quantum codes," 2026. [Online]. Available: https://arxiv.org/abs/2601.18590

[139] A. W. Cross, Z. He, P. J. Rall, and T. J. Yoder, "Improved qldpc surgery: Logical measurements and bridging codes," 2025. [Online]. Available: https://arxiv.org/abs/2407.18393

[140] C. Liu, H.-L. Huang, C. Chen *et al.*, "Demonstration of topologically path-independent anyonic braiding in a nine-qubit planar code," *Optica*, vol. 6, no. 3, pp. 264–268, 2019.

[141] T. A. Aditto, J. S. Ifty, and K. Zahin, "Toward scalable heterogeneous quantum networks: Microwave-optical transduction across platforms," 2026. [Online]. Available: https://arxiv.org/abs/2605.26976

[142] F. Shen, Y. Ji, D. Xiang *et al.*, "A bucket-brigade quantum random access memory," *Nature Physics*, vol. 22, no. 5, pp. 745–750, 2026.

[143] A. Strikis and L. Berent, "Quantum low-density parity-check codes for modular architectures," *PRX Quantum*, vol. 4, no. 2, p. 020321, 2023.

[144] S. Du, Y. Ding, and C. Qiao, "S-QGPU: Shared quantum gate processing unit for distributed quantum computing," *AVS Quantum Science*, vol. 7, no. 1, p. 013803, 2025.

[145] C. Clayton and B. Avritzer, "Distributed quantum error correction with permutation-invariant approximate codes," 2025. [Online]. Available: https://arxiv.org/abs/2509.25093

[146] D. Sakuma, T. Tsuno, H. Shimizu *et al.*, "Q-fly: An optical interconnect for modular quantum computers," 2025. [Online]. Available: https://arxiv.org/abs/2412.09299

[147] J. Knörzer, X. Liu, B. F. Schiffer, and J. Tura, "Distributed quantum information processing: a review of recent progress," *Reports on Progress in Physics*, vol. 89, no. 7, p. 074401, 2026.

[148] K. Takeda, A. Noiri, T. Nakajima, T. Kobayashi, and S. Tarucha, "Quantum error correction with silicon spin qubits," *Nature*, vol. 608, pp. 682–686, 2022.

[149] X. Xue, M. Russ, N. Samkharadze, B. Undseth, A. Sammak, G. Scappucci, and L. M. K. Vandersypen, "Quantum logic with spin qubits crossing the surface code threshold," *Nature*, vol. 601, no. 7893, pp. 343–347, 2022.

[150] D. Bluvstein, A. A. Geim, S. H. Li *et al.*, "A fault-tolerant neutral-atom architecture for universal quantum computation," *Nature*, vol. 649, pp. 39–46, 2026.

[151] N.-C. Chiu, E. C. Trapp, J. Guo *et al.*, "Continuous operation of a coherent 3,000-qubit system," *Nature*, vol. 646, pp. 1075–1080, 2025.

[152] J. Wang, I. Harris, M. Ibrahim, D. Englund, and R. Han, "A wireless terahertz cryogenic interconnect that minimizes heat-to-information transfer," *Nature Electronics*, vol. 8, pp. 426–436, 2025.

[153] L. Z. Cohen, I. H. Kim, S. D. Bartlett, and B. J. Brown, "Low-overhead fault-tolerant quantum computing using long-range connectivity," *Science Advances*, vol. 8, no. 20, p. eabn1717, 2022.

[154] S. Singh, F. Gu, S. de Bone, E. Villaseñor, D. Elkouss, and J. Borregaard, "Modular architectures and entanglement schemes for error-corrected distributed quantum computation," *npj Quantum Information*, vol. 12, no. 1, p. 3, 2025.

[155] H. Shapourian, E. Kaur, T. Sewell, J. Zhao, M. Kilzer, R. Kompella, and R. Nejabati, "Quantum data center infrastructures: A scalable architectural design perspective," 2025. [Online]. Available: https://arxiv.org/abs/2501.05598

[156] C. Clos, "A study of non-blocking switching networks," *The Bell System Technical Journal*, vol. 32, no. 2, pp. 406–424, 1953.

[157] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," *IEEE Transactions on Computers*, vol. C-34, no. 10, pp. 892–901, 1985.

[158] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, "Hyperx: topology, routing, and packaging of efficient large-scale networks," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*. New York, NY, USA: Association for Computing Machinery, 2009.

[159] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: a high performance, server-centric network architecture for modular data centers," in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*. New York, NY, USA: Association for Computing Machinery, 2009, p. 63–74.

[160] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*. New York, NY, USA: Association for Computing Machinery, 2008, p. 75–86.

[161] G. Bernardi, R. Mahajan, C. Seshadhri *et al.*, "Rng: Flat datacenter networks at scale," 2026. [Online]. Available: https://arxiv.org/abs/2604.15261

[162] W. Dür, G. Vidal, and J. I. Cirac, "Three qubits can be entangled in two inequivalent ways," *Physical Review A*, vol. 62, p. 062314, 2000.

[163] J. A. Muniz, D. Crow, H. Kim *et al.*, "Repeated ancilla reuse for logical computation on a neutral atom quantum computer," *Physical Review X*, vol. 15, p. 041040, 2025.

[164] P. V. Klimov, A. Bengtsson, C. Quintana *et al.*, "Optimizing quantum gates towards the scale of logical qubits," *Nature Communications*, vol. 15, no. 1, p. 2442, 2024.

[165] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Physical Review A*, vol. 52, pp. R2493–R2496, 1995.

[166] P. Shanahan and D. Ruiz, "Elevator codes: Concatenation for resource-efficient quantum memory under biased noise," 2026. [Online]. Available: https://arxiv.org/abs/2601.10786

[167] A. M. Steane, "Error correcting codes in quantum theory," *Physical Review Letters*, vol. 77, pp. 793–797, 1996.

[168] R. Laflamme, C. Miquel, J. P. Paz, and W. H. Zurek, "Perfect quantum error correcting code," *Physical Review Letters*, vol. 77, pp. 198–201, 1996.

[169] J. Niu, Y. Li, L. Zhang *et al.*, "Demonstrating path-independent anyonic braiding on a modular superconducting quantum processor," *Physical Review Letters*, vol. 132, p. 020601, 2024.

[170] S. Lloyd, S. Garnerone, and P. Zanardi, "Quantum algorithms for topological and geometric analysis of data," *Nature Communications*, vol. 7, p. 10138, 2016.

[171] A. Kitaev, "Fault-tolerant quantum computation by anyons," *Annals of Physics*, vol. 303, no. 1, pp. 2–30, 2003.

[172] D. Bluvstein, H. Levine, G. Semeghini *et al.*, "A quantum processor based on coherent transport of entangled atom arrays," *Nature*, vol. 604, pp. 451–456, 2022.

[173] S. Krinner, N. Lacroix, A. Remm *et al.*, "Realizing repeated quantum error correction in a distance-three surface code," *Nature*, vol. 605, pp. 669–674, 2022.

[174] Y. Zhao, Y. Ye, H.-L. Huang *et al.*, "Realization of an error-correcting surface code with superconducting qubits," *Physical Review Letters*, vol. 129, p. 030501, 2022.

[175] R. Acharya, D. A. Abanin, L. Aghababaie-Beni *et al.*, "Quantum error correction below the surface code threshold," *Nature*, vol. 638, pp. 920–926, 2025.

[176] R. Acharya, I. Aleiner, R. Allen *et al.*, "Suppressing quantum errors by scaling a surface code logical qubit," *Nature*, vol. 614, pp. 676–681, 2023.

[177] N. Berthusen, J. Dreiling, C. Foltz, J. P. Gaebler, T. M. Gatterman, D. Gresh, N. Hewitt, M. Mills, S. A. Moses, B. Neyenhuis, P. Siegfried, and D. Hayes, "Experiments with the four-dimensional surface code on a quantum charge-coupled device quantum computer," *Physical Review A*, vol. 110, p. 062413, 2024.

[178] X.-C. Yao, T.-X. Wang, H.-Z. Chen *et al.*, "Experimental demonstration of topological error correction," *Nature*, vol. 482, no. 7386, pp. 489–494, 2012.

[179] C. Gidney, M. Newman, P. Brooks, and C. Jones, "Yoked surface codes," *Nature Communications*, vol. 16, no. 1, p. 4498, 2025.

[180] S. de Bone, P. Möller, C. E. Bradley, T. H. Taminiau, and D. Elkouss, "Thresholds for the distributed surface code in the presence of memory decoherence," *AVS Quantum Science*, vol. 6, no. 3, p. 033801, 2024.

[181] L. Postler, F. Butt, I. Pogorelov, C. D. Marciniak, S. Heußen, R. Blatt, P. Schindler, M. Rispler, M. Müller, and T. Monz, "Demonstration of fault-tolerant steane quantum error correction," *PRX Quantum*, vol. 5, p. 030326, 2024.

[182] M. E. Beverland, A. Kubica, and K. M. Svore, "Cost of universality: A comparative study of the overhead of state distillation and code switching with color codes," *PRX Quantum*, vol. 2, p. 020341, 2021.

[183] C. Gidney and C. Jones, "New circuits and an open source decoder for the color code," 2023. [Online]. Available: https://arxiv.org/abs/2312.08813

[184] D. Bluvstein, S. J. Evered, A. A. Geim *et al.*, "Logical quantum processor based on reconfigurable atom arrays," *Nature*, vol. 626, no. 7997, pp. 58–65, 2024.

[185] N. P. Breuckmann and J. N. Eberhardt, "Quantum low-density parity-check codes," *PRX Quantum*, vol. 2, no. 4, p. 040101, 2021.

[186] J.-P. Tillich and G. Zemor, "Quantum ldpc codes with positive rate and minimum distance proportional to the square root of the block-length," *IEEE Transactions on Information Theory*, vol. 60, no. 2, p. 1193–1202, 2014.

[187] S. Evra, T. Kaufman, and G. Zémor, "Decodable quantum ldpc codes beyond the $\sqrt{n}$ distance barrier using high-dimensional expanders," *SIAM Journal on Computing*, vol. 53, no. 6, pp. FOCS20–276–FOCS20–316, 2024.

[188] T. Kaufman and R. J. Tessler, "New cosystolic expanders from tensors imply explicit quantum ldpc codes with $\Omega(\sqrt{n}\log^k n)$ distance," in *53rd Annual ACM SIGACT Symposium on Theory of Computing*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 1317–1329.

[189] M. B. Hastings, J. Haah, and R. O'Donnell, "Fiber bundle codes: breaking the n1/2 polylog(n) barrier for quantum ldpc codes," in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. New York, NY, USA: Association for Computing Machinery, 2021, p. 1276–1288.

[190] P. Panteleev and G. Kalachev, "Quantum ldpc codes with almost linear minimum distance," *IEEE Transactions on Information Theory*, vol. 68, no. 1, p. 213–229, 2022.

[191] N. P. Breuckmann and J. N. Eberhardt, "Balanced product quantum codes," *IEEE Transactions on Information Theory*, vol. 67, no. 10, p. 6653–6674, 2021.

[192] M. A. Tremblay, N. Delfosse, and M. E. Beverland, "Constant-overhead quantum error correction with thin planar connectivity," *Physical Review Letters*, vol. 129, p. 050504, 2022.

[193] X. Deng, W. Zheng, X. Liao, H. Zhou, Y. Ge, J. Zhao, D. Lan, X. Tan, Y. Zhang, S. Li, and Y. Yu, "Long-range zz interaction via resonator-induced phase in superconducting qubits," *Physical Review Letters*, vol. 134, no. 2, p. 020801, 2025.

[194] Q. Xu, J. P. B. Ataides, C. A. Pattison, and N. Raveendran, "Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays," *Nature Physics*, vol. 20, no. 7, pp. 1084–1090, 2024.

[195] G. P. Gehér, D. Byfield, and A. Ruban, "Directional codes: a new family of quantum ldpc codes on hexagonal- and square-grid connectivity hardware," 2025. [Online]. Available: https://arxiv.org/abs/2507.19430

[196] B. Undseth, N. Meggiato, Y.-H. Wu, S. R. Katiraee-Far, L. Tryputen, S. L. de Snoo, D. D. Esposti, G. Scappucci, E. Greplová, and L. M. K. Vandersypen, "Weight-four parity checks with silicon spin qubits," 2026. [Online]. Available: https://arxiv.org/abs/2601.23267

[197] S. Tamiya, M. Koashi, and H. Yamasaki, "Fault-tolerant quantum computation with polylogarithmic time and constant space overheads," *Nature Physics*, vol. 22, no. 1, pp. 27–32, 2026.

[198] A. A. Kovalev and L. P. Pryadko, "Quantum kronecker sum-product low-density parity-check codes with finite rate," *Physical Review A*, vol. 88, p. 012311, 2013.

[199] T. J. Yoder, E. Schoute, P. Rall, E. Pritchett, J. M. Gambetta, A. W. Cross, M. Carroll, and M. E. Beverland, "Tour de gross: A modular quantum computer based on bivariate bicycle codes," 2025. [Online]. Available: https://arxiv.org/abs/2506.03094

[200] E. Swaroop, T. Jochym-O'Connor, and T. J. Yoder, "Universal adapters between quantum low-density parity check codes," *PRX Quantum*, vol. 7, p. 010324, 2026.

[201] S. Xu, K. Liu, P. Rall, Z. He, and Y. Ding, "Distilling magic states in the bicycle architecture," 2026. [Online]. Available: https://arxiv.org/abs/2602.20546

[202] K. Wang, Z. Lu, C. Zhang *et al.*, "Demonstration of low-overhead quantum error correction codes," *Nature Physics*, vol. 22, no. 2, pp. 308–314, 2026.

[203] M. Wang and F. Mueller, "Coprime Bivariate Bicycle Codes and Their Layouts on Cold Atoms," *Quantum*, vol. 10, p. 2009, 2026.

[204] L. Voss, S. J. Xian, T. Haug, and K. Bharti, "Multivariate bicycle codes," *Physical Review A*, vol. 111, p. L060401, 2025.

[205] M. H. Shaw and B. M. Terhal, "Lowering connectivity requirements for bivariate bicycle codes using morphing circuits," *Physical Review Letters*, vol. 134, p. 090602, 2025.

[206] J. Guo, Y. Hong, A. Kaufman, and A. Lucas, "Toward self-correcting quantum codes for neutral atom arrays," *PRX Quantum*, vol. 7, p. 010301, 2026.

[207] Z. Liang, K. Liu, H. Song, and Y.-A. Chen, "Generalized toric codes on twisted tori for quantum error correction," *PRX Quantum*, vol. 6, p. 020357, 2025.

[208] M. Halla, "Qudit twisted-torus codes in the bivariate bicycle framework," 2026. [Online]. Available: https://arxiv.org/abs/2602.04443

[209] S. Jacoby, A. Retzker, and F. Pastawski, "Stairway codes: Floquetifying bivariate bicycle codes and beyond," 2026. [Online]. Available: https://arxiv.org/abs/2603.00228

[210] N. K. Chandra, E. Kaur, R. Nejabati, and K. P. Seshadreesan, "Distributed quantum error correction with bivariate bicycle codes in a modular architecture," 2026. [Online]. Available: https://arxiv.org/abs/2605.04663

[211] A. S. Bhave, N. Choudhury, A. Nemec, and K. Basu, "Bibieq: Bivariate bicycle codes on erasure qubits," 2026. [Online]. Available: https://arxiv.org/abs/2602.07578

[212] H. Bombín, "Gauge color codes: optimal transversal gates and gauge fixing in topological stabilizer codes," *New Journal of Physics*, vol. 17, no. 8, p. 083002, 2015.

[213] D. Bacon, "Operator quantum error-correcting subsystems for self-correcting quantum memories," *Physical Review A*, vol. 73, p. 012340, 2006.

[214] D. Poulin, "Stabilizer formalism for operator quantum error correction," *Physical Review Letters*, vol. 95, p. 230504, 2005.

[215] D. Bacon and A. Casaccino, "Quantum error correcting subsystem codes from two classical linear codes," 2006. [Online]. Available: https://arxiv.org/abs/quant-ph/0610088

[216] L. Egan, D. M. Debroy, C. Noel *et al.*, "Fault-tolerant control of an error-corrected qubit," *Nature*, vol. 598, pp. 281–286, 2021.

[217] S. Bravyi, G. Duclos-Cianci, D. Poulin, and M. Suchara, "Subsystem surface codes with three-qubit check operators," *Quantum Information & Computation*, vol. 13, no. 11–12, p. 963–985, 2013.

[218] A. G. Fowler, A. C. Whiteside, and L. C. L. Hollenberg, "Towards practical classical processing for the surface code," *Physical Review Letters*, vol. 108, p. 180501, 2012.

[219] O. Higgott and N. P. Breuckmann, "Subsystem codes with high thresholds by gauge fixing and reduced qubit overhead," *Physical Review X*, vol. 11, p. 031039, 2021.

[220] A. Kitaev, "Anyons in an exactly solved model and beyond," *Annals of Physics*, vol. 321, no. 1, p. 2–111, 2006.

[221] G. Kells, J. K. Slingerland, and J. Vala, "Description of kitaev's honeycomb model with toric-code stabilizers," *Physical Review B*, vol. 80, p. 125415, 2009.

[222] C. Gidney, M. Newman, A. Fowler, and M. Broughton, "A Fault-Tolerant Honeycomb Memory," *Quantum*, vol. 5, p. 605, 2021.

[223] E. Sutcliffe, B. Jonnadula, C. Le Gall, A. E. Moylett, and C. M. Westoby, "Distributed quantum error correction based on hyperbolic floquet codes," in *IEEE International Conference on Quantum Computing and Engineering (QCE)*. Piscataway, NJ, USA: IEEE, 2025, pp. 649–657.

[224] M. S. Alam and E. Rieffel, "Dynamical logical qubits in the bacon-shor code," *Physical Review A*, vol. 112, p. 022436, 2025.

[225] C. Gidney and D. Bacon, "Less bacon more threshold," 2023. [Online]. Available: https://arxiv.org/abs/2305.12046

[226] A. Eickbusch, M. McEwen, V. Sivak *et al.*, "Demonstration of dynamic surface codes," *Nature Physics*, vol. 21, pp. 1994–2001, 2025.

[227] S. Maurya, T. Maurer, M. Bühler, D. Vandeth, and M. E. Beverland, "Fpga-tailored algorithms for real-time decoding of quantum ldpc codes," 2026. [Online]. Available: https://arxiv.org/abs/2511.21660

[228] O. Higgott, T. C. Bohdanowicz, A. Kubica, S. T. Flammia, and E. T. Campbell, "Improved decoding of circuit noise and fragile boundaries of tailored surface codes," *Physical Review X*, vol. 13, p. 031007, 2023.

[229] A. deMarti iOlius, P. Fuentes, R. Orús, P. M. Crespo, and J. Etxezarreta Martinez, "Decoding algorithms for surface codes," *Quantum*, vol. 8, p. 1498, 2024.

[230] Y. Kurman, L. Ella, R. Szmuk, O. Wertheim, B. Dorschner, S. Stanwyck, and Y. Cohen, " Benchmarking the Ability of a Controller to Execute Quantum Error Corrected Non-Clifford Circuits ," *IEEE Transactions on Quantum Engineering*, vol. 6, no. 01, pp. 1–14, 2025.

[231] B. Barber, K. M. Barnes, T. Bialas *et al.*, "A real-time, scalable, fast and resource-efficient decoder for a quantum computer," *Nature Electronics*, vol. 8, no. 1, pp. 84–91, 2025.

[232] X. Tan, F. Zhang, R. Chao, Y. Shi, and J. Chen, "Scalable surface-code decoders with parallelization in time," *PRX Quantum*, vol. 4, p. 040344, 2023.

[233] L. Skoric, D. E. Browne, K. M. Barnes, N. I. Gillespie, and E. T. Campbell, "Parallel window decoding enables scalable fault tolerant quantum computation," *Nature Communications*, vol. 14, no. 1, p. 7040, 2023.

[234] H. Bombín, C. Dawson, Y.-H. Liu, N. Nickerson, F. Pastawski, and S. Roberts, "Modular decoding: parallelizable real-time decoding for quantum computers," 2023. [Online]. Available: https://arxiv.org/abs/2303.04846

[235] N. Liyanage, Y. Wu, S. Tagare, and L. Zhong, "Fpga-based distributed union-find decoder for surface codes," *IEEE Transactions on Quantum Engineering*, vol. 5, pp. 1–18, 2024.

[236] Y. Wu and L. Zhong, "Fusion blossom: Fast mwpm decoders for qec," in *IEEE International Conference on Quantum Computing and Engineering (QCE)*. Piscataway, NJ, USA: IEEE, 2023, pp. 928–938.

[237] O. Higgott and C. Gidney, "Sparse Blossom: correcting a million errors per core second with minimum-weight matching," *Quantum*, vol. 9, p. 1600, 2025.

[238] N. Delfosse and N. H. Nickerson, "Almost-linear time decoding algorithm for topological codes," *Quantum*, vol. 5, p. 595, 2021.

[239] J. P. Bonilla Ataides, D. K. Tuckett, S. D. Bartlett, S. T. Flammia, and B. J. Brown, "The XZZX surface code," *Nature Communications*, vol. 12, no. 1, p. 2172, 2021.

[240] L. Valentini, D. Forlivesi, A. Talarico, and M. Chiani, "Restart belief: A general quantum ldpc decoder," *IEEE Communications Letters*, vol. 30, pp. 1185–1189, 2026.

[241] J. Roffe, D. R. White, S. Burton, and E. Campbell, "Decoding across the quantum low-density parity-check code landscape," *Physical Review Research*, vol. 2, p. 043423, 2020.

[242] A. Kaufmann and I. Arad, "Blockbp decoder for the surface code," *Physical Review Research*, vol. 7, p. 043025, 2025.

[243] C. Piveteau, C. T. Chubb, and J. M. Renes, "Tensor-network decoding beyond 2D," *PRX Quantum*, vol. 5, p. 040303, 2024.

[244] J. Bausch, A. W. Senior *et al.*, "Learning high-accuracy error decoding for quantum processors," *Nature*, vol. 635, no. 8040, pp. 834–840, 2024.

[245] A. Gu, J. P. B. Ataides, M. D. Lukin, and S. F. Yelin, "Scalable neural decoders for practical fault-tolerant quantum computation," 2026. [Online]. Available: https://arxiv.org/abs/2604.08358

[246] V. Sivak, M. Newman, and P. Klimov, "Optimization of decoder priors for accurate quantum error correction," *Physical Review Letters*, vol. 133, p. 150603, 2024.

[247] A. Kubica and J. Preskill, "Cellular-automaton decoders with provable thresholds for topological codes," *Physical Review Letters*, vol. 123, p. 020501, 2019.

[248] J. F. S. Miguel, D. J. Williamson, and B. J. Brown, "A cellular automaton decoder for a noise-bias tailored color code," *Quantum*, vol. 7, p. 940, 2023.

[249] N. Shutty, M. Newman, and B. Villalonga, "Efficient near-optimal decoding of the surface code through ensembling," *Physical Review Letters*, vol. 136, p. 070603, 2026.

[250] C. Jones, "Improved accuracy for decoding surface codes with matching synthesis," 2024. [Online]. Available: https://arxiv.org/abs/2408.12135

[251] N. Shutty, M. Newman, and B. Villalonga, "Efficient near-optimal decoding of the surface code through ensembling," *Physical Review Letters*, vol. 136, p. 070603, 2026.

[252] S. Huang, M. Newman, and K. R. Brown, "Fault-tolerant weighted union-find decoding on the toric code," *Physical Review A*, vol. 102, p. 012419, 2020.

[253] N. Liyanage, Y. Wu, S. Tagare, and L. Zhong, "Fpga-based distributed union-find decoder for surface codes," *IEEE Transactions on Quantum Engineering*, vol. 5, pp. 1–18, 2024.

[254] T. Chan and S. C. Benjamin, "Actis: A Strictly Local Union–Find Decoder," *Quantum*, vol. 7, p. 1183, 2023.

[255] A. B. Ziad, A. Zalawadiya, C. Topal, J. Camps, G. P. Gehér, M. P. Stafford, and M. L. Turner, "Local clustering decoder as a fast and adaptive hardware decoder for the surface code," *Nature Communications*, vol. 16, no. 1, p. 11048, 2025.

[256] D. Forlivesi, L. Valentini, and M. Chiani, "Bubble clustering decoder for quantum topological codes," *IEEE Transactions on Communications*, vol. 73, no. 10, pp. 8470–8483, 2025.

[257] P. Hack, L. Menti, F. Lazaro, and A. Paler, "Achieving thresholds via standalone belief propagation on surface codes," 2026. [Online]. Available: https://arxiv.org/abs/2603.05381

[258] J. Old and M. Rispler, "Generalized Belief Propagation Algorithms for Decoding of Surface Codes," *Quantum*, vol. 7, p. 1037, 2023.

[259] B. M. Varbanov, M. Serra-Peralta, D. Byfield, and B. M. Terhal, "Neural network decoder for near-term surface-code experiments," *Physical Review Research*, vol. 7, p. 013029, 2025.

[260] X. Yang, X. Sun, Z. Wu *et al.*, "Real-time surface-code error correction using an fpga-based neural-network decoder," 2026. [Online]. Available: https://arxiv.org/abs/2605.04892

[261] J. W. Harrington, "Analysis of quantum error-correcting codes: Symplectic lattice codes and toric codes," Ph.D. dissertation, California Institute of Technology, Pasadena, California, 2004, ph.D. dissertation.

[262] N. P. Breuckmann, K. Duivenvoorden, D. Michels, and B. M. Terhal, "Local decoders for the 2D and 4D toric code," *Quantum Information & Computation*, vol. 17, no. 3–4, p. 181–208, 2017.

[263] P. Baireuther, M. D. Caio, B. Criger, C. W. J. Beenakker, and T. E. O'Brien, "Neural network decoder for topological color codes with circuit level noise," *New Journal of Physics*, vol. 21, no. 1, p. 013003, 2019.

[264] A. Kubica and N. Delfosse, "Efficient color code decoders in $d \geq 2$ dimensions from toric code decoders," *Quantum*, vol. 7, p. 929, 2023.

[265] C.-F. Kung, K.-Y. Kuo, and C.-Y. Lai, "On belief propagation decoding of quantum codes with quaternary reliability statistics," in *12th International Symposium on Topics in Coding (ISTC)*. Piscataway, NJ, USA: IEEE, 2023, pp. 1–5.

[266] L. Berent, L. Burgholzer, P.-J. H. Derks, J. Eisert, and R. Wille, "Decoding quantum color codes with MaxSAT," *Quantum*, vol. 8, p. 1506, 2024.

[267] N. K. Chandra, D. Tipper, R. Nejabati, E. Kaur, and K. P. Seshadreesan, "Distributed realization of color codes for quantum error correction," in *IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 01. Piscataway, NJ, USA: IEEE, 2025, pp. 2482–2492.

[268] J. D. Crest, M. Mhalla, and V. Savin, "Stabilizer inactivation for message-passing decoding of quantum ldpc codes," in *IEEE Information Theory Workshop (ITW)*. Piscataway, NJ, USA: IEEE, 2022, pp. 488–493.

[269] A. Gong, S. Cammerer, and J. M. Renes, "Toward low-latency iterative decoding of qldpc codes under circuit-level noise," 2024. [Online]. Available: https://arxiv.org/abs/2403.18901

[270] H. Yao, W. A. Laban, C. Häger, A. G. i. Amat, and H. D. Pfister, "Belief propagation decoding of quantum ldpc codes with guided decimation," in *IEEE International Symposium on Information Theory (ISIT)*. Piscataway, NJ, USA: IEEE, 2024, pp. 2478–2483.

[271] T. Hillmann, L. Berent, A. O. Quintavalle, J. Eisert, R. Wille, and J. Roffe, "Localized statistics decoding for quantum low-density parity-check codes," *Nature Communications*, vol. 16, no. 1, p. 8214, 2025.

[272] A. deMarti iOlius, I. E. Martinez, J. Roffe, and J. E. Martinez, "An almost-linear time decoding algorithm for quantum ldpc codes under circuit-level noise," 2025. [Online]. Available: https://arxiv.org/abs/2409.01440

[273] T. Müller, T. Alexander, M. E. Beverland, M. Bühler, B. R. Johnson, T. Maurer, and D. Vandeth, "Improved belief propagation is sufficient for real-time decoding of quantum memory," 2025. [Online]. Available: https://arxiv.org/abs/2506.01779

[274] T. Maurer, M. Bühler, M. Kröner, F. Haverkamp, T. Müller, D. Vandeth, and B. R. Johnson, "Real-time decoding of the gross code memory with fpgas," 2025. [Online]. Available: https://arxiv.org/abs/2510.21600

[275] A. Leverrier, J.-P. Tillich, and G. Zémor, "Quantum expander codes," in *IEEE 56th Annual Symposium on Foundations of Computer Science*. Piscataway, NJ, USA: IEEE, 2015, pp. 810–824.

[276] A. Grospellier, L. Grouès, A. Krishna, and A. Leverrier, "Combining hard and soft decoders for hypergraph product codes," *Quantum*, vol. 5, p. 432, 2021.

[277] N. Connolly, V. Londe, A. Leverrier, and N. Delfosse, "Fast erasure decoder for hypergraph product codes," *Quantum*, vol. 8, p. 1450, 2024.

[278] S. Gu, C. A. Pattison, and E. Tang, "An efficient decoder for a linear distance quantum ldpc code," in *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*. New York, NY, USA: Association for Computing Machinery, 2023, p. 919–932.

[279] S. Gu, E. Tang, L. Caha, S. H. Choe, Z. He, and A. Kubica, "Single-shot decoding of good quantum ldpc codes," *Communications in Mathematical Physics*, vol. 405, no. 3, p. 85, 2024.

[280] S. Wolanski and B. Barber, "Introducing ambiguity clustering: An accurate and efficient decoder for qldpc codes," in *IEEE International Conference on Quantum Computing and Engineering (QCE)*. Piscataway, NJ, USA: IEEE, 2024, pp. 402–403.

[281] A. deMarti iOlius and J. E. Martinez, "The closed-branch decoder for quantum ldpc codes," 2024. [Online]. Available: https://arxiv.org/abs/2402.01532

[282] M. Ye, D. Wecker, and N. Delfosse, "Beam search decoder for quantum ldpc codes," 2025. [Online]. Available: https://arxiv.org/abs/2512.07057

[283] L. A. Beni, O. Higgott, and N. Shutty, "Tesseract: A search-based decoder for quantum error correction," 2025. [Online]. Available: https://arxiv.org/abs/2503.10988

[284] D. Grbic, L. A. Beni, and N. Shutty, "Accelerating the tesseract decoder for quantum error correction," 2026. [Online]. Available: https://arxiv.org/abs/2602.02985

[285] K. Sahay, D. J. Williamson, and B. J. Brown, "A matching decoder for bivariate bicycle codes," 2026. [Online]. Available: https://arxiv.org/abs/2602.22770

[286] N. Delfosse, V. Londe, and M. E. Beverland, "Toward a union-find decoder for quantum ldpc codes," *IEEE Transactions on Information Theory*, vol. 68, no. 5, pp. 3187–3199, 2022.

[287] A. Gu, J. P. B. Ataides, M. D. Lukin, and S. F. Yelin, "Scalable neural decoders for practical fault-tolerant quantum computation," 2026. [Online]. Available: https://arxiv.org/abs/2604.08358

[288] A. Gong, S. Cammerer, and J. M. Renes, "Graph neural networks for enhanced decoding of quantum LDPC codes," in *Proceedings of the 2024 IEEE International Symposium on Information Theory (ISIT)*. Piscataway, NJ, USA: IEEE, 2024, pp. 2700–2705.

[289] A. S. Maan and A. Paler, "Machine learning message-passing for the scalable decoding of qldpc codes," *npj Quantum Information*, vol. 11, no. 1, p. 78, 2025.

[290] J. Blue, H. Avlani, Z. He, L. Ziyin, and I. L. Chuang, "Machine learning decoding of circuit-level noise for bivariate bicycle codes," 2025. [Online]. Available: https://arxiv.org/abs/2504.13043

[291] A. S. Maan, F. M. Garcia Herrero, A. Paler, and V. Savin, "Decoding correlated errors in quantum ldpc codes," *Nature Communications*, vol. 17, no. 1, p. 3965, 2026.

[292] O. Ferraz, B. Coutinho, G. Falcao, M. Gomes, F. A. Monteiro, and V. Silva, "Gpu-accelerated syndrome decoding for quantum ldpc codes below the 63 μs latency threshold," 2025. [Online]. Available: https://arxiv.org/abs/2508.07879

[293] C. Gidney, "Stim: a fast stabilizer circuit simulator," *Quantum*, vol. 5, p. 497, 2021.

[294] A. Javadi-Abhari, M. Treinish, K. Krsulich *et al.*, "Quantum computing with qiskit," 2024. [Online]. Available: https://arxiv.org/abs/2405.08810

[295] M. P. Harrigan, T. Khattar, C. Yuan, A. Peduri, N. Yosri, F. D. Malone, R. Babbush, and N. C. Rubin, "Expressing and analyzing quantum algorithms with qualtran," 2024. [Online]. Available: https://arxiv.org/abs/2409.04643

[296] T. Coopmans, R. Knegjens, A. Dahlberg *et al.*, "Netsquid, a network simulator for quantum information using discrete events," *Communications Physics*, vol. 4, no. 1, p. 164, 2021.

[297] R. LaRose, A. Mari, S. Kaiser *et al.*, "Mitiq: A software package for error mitigation on noisy quantum computers," *Quantum*, vol. 6, p. 774, 2022.

[298] C. Gidney, "Sinter: Fast monte carlo sampling of quantum error correction circuits," https://pypi.org/project/sinter/, 2025.

[299] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, "t|ket⟩: a retargetable compiler for nisq devices," *Quantum Science and Technology*, vol. 6, no. 1, p. 014003, 2020.

[300] W. van Dam, M. Mykhailova, and M. Soeken, "Using azure quantum resource estimator for assessing performance of fault tolerant quantum computation," in *Proceedings of the Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*. New York, NY, USA: Association for Computing Machinery, 2023, p. 1414–1419.

[301] Y. Suzuki, Y. Kawase, Y. Masumura *et al.*, "Qulacs: a fast and versatile quantum circuit simulator for research purpose," *Quantum*, vol. 5, p. 559, 2021.

[302] X. Wu, A. Kolar, J. Chung, D. Jin, T. Zhong, R. Kettimuthu, and M. Suchara, "Sequence: a customizable discrete-event simulator of quantum networks," *Quantum Science and Technology*, vol. 6, no. 4, p. 045027, 2021.

[303] R. Haenel, X. Luo, and C. Zhao, "Tsim: Fast universal simulator for quantum error correction," 2026. [Online]. Available: https://arxiv.org/abs/2604.01059

[304] A. Javadi Abhari, A. Faruque, D. M. Javad *et al.*, "Scaffold: Quantum programming language," Princeton University, Department of Computer Science, Tech. Rep. TR-934, 2012. [Online]. Available: https://www.cs.princeton.edu/techreports/2012/934.pdf

[305] K. Svore, A. Geller, M. Troyer *et al.*, "Q#: Enabling scalable quantum computing and development with a high-level dsl," in *Proceedings of the Real World Domain Specific Languages Workshop 2018*. New York, NY, USA: Association for Computing Machinery, 2018.

[306] J. Krzyszkowski and M. Niemiec, "Analysis of surface code algorithms on quantum hardware using the Qrisp framework," *Electronics*, vol. 14, no. 23, p. 4707, 2025.

[307] J. Guo, H. Lou, J. Yu, R. Li, W. Fang, J. Liu, P. Long, S. Ying, and M. Ying, "isq: An integrated software stack for quantum programming," *IEEE Transactions on Quantum Engineering*, vol. 4, pp. 1–16, 2023.

[308] S. Liu, X. Wang, L. Zhou *et al.*, *$Q|SI\rangle$: A Quantum Programming Environment*, ser. Lecture Notes in Computer Science. Cham, Switzerland: Springer International Publishing, 2018, vol. 11180, pp. 133–164.

[309] A. Cross, A. Javadi-Abhari, T. Alexander *et al.*, "OpenQASM 3: A broader and deeper quantum assembly language," *ACM Transactions on Quantum Computing*, vol. 3, no. 3, 2022.

[310] X. Fu, L. Riesebos, M. A. Rol *et al.*, "eqasm: An executable quantum instruction set architecture," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*. Piscataway, NJ, USA: IEEE, 2019, pp. 224–237.

[311] A. Dahlberg, B. van der Vecht, C. Delle Donne, and otehrs, "Netqasm—a low-level instruction set architecture for hybrid quantum–classical programs in a quantum internet," *Quantum Science and Technology*, vol. 7, no. 3, p. 035023, 2022.

[312] A. Suau, Y. Zhang, T. Purva *et al.*, "tqec: A python package for topological quantum error correction," *Journal of Open Source Software*, vol. 11, no. 120, p. 9142, 2026.

[313] A. Świerkowska, J. Pflieger, E. Giortamis, and P. Bhatotia, "Eccentric: An empirical analysis of quantum error correction codes," 2025. [Online]. Available: https://arxiv.org/abs/2511.01062

[314] O. Higgott, "Pymatching: A python package for decoding quantum codes with minimum-weight perfect matching," *ACM Transactions on Quantum Computing*, vol. 3, no. 3, 2022.

[315] D. González-Cuadra, M. Hamdan, T. V. Zache *et al.*, "Observation of string breaking on a (2 + 1)d rydberg quantum simulator," *Nature*, vol. 642, no. 8067, pp. 321–326, 2025.

[316] V. Menon, J. P. Bonilla Ataides, R. Mehta, A. Gu, D. B. Tan, and M. D. Lukin, "Magic tricycles: Efficient magic-state generation with finite block-length quantum ldpc codes," *Physical Review X*, vol. 16, p. 021014, 2026.

[317] C. Zhao, C. Duckering, A. Gu, N. Maskara, and H. Zhou, "Towards ultra-high-rate quantum error correction with reconfigurable atom arrays," 2026. [Online]. Available: https://arxiv.org/abs/2604.16209

[318] P. Mathiot, E. Garnaoui, A.-U. Leriche *et al.*, "Benchmarking a machine-learning differential equations solver on a neutral-atom logical processor," 2026. [Online]. Available: https://arxiv.org/abs/2605.21276

[319] A. B. Yoo, M. A. Jette, and M. Grondona, "SLURM: Simple linux utility for resource management," in *Job Scheduling Strategies for Parallel Processing*, ser. Lecture Notes in Computer Science, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds., vol. 2862. Berlin, Heidelberg: Springer, 2003, pp. 44–60.