CaliQEC: In-situ Qubit Calibration for Surface Code Quantum Error Correction

Xiang Fang
University of California San Diego
La Jolla, USA
x8fang@ucsd.edu

Keyi Yin
University of California San Diego
La Jolla, USA
keyin@ucsd.edu

Yuchen Zhu
University of California San Diego
La Jolla, USA
ax009822@acsmail.ucsd.edu

Jixuan Ruan
University of California San Diego
La Jolla, USA
j3ruan@ucsd.edu

Dean Tullsen
University of California San Diego
La Jolla, USA
tullsen@ucsd.edu

Zhiding Liang
Rensselaer Polytechnic Institute
Troy, USA
liangz9@rpi.edu

Andrew Sornborger
LANL
Los Alamos, USA
sornborg@lanl.gov

Ang Li
PNNL
Richland, USA
ang.li@pnnl.gov

Travis Humble
Quantum Science Center, Oak Ridge National Laboratory
Oak Ridge, USA
humblets@ornl.gov

Yufei Ding
University of California San Diego
La Jolla, USA
yufeiding@ucsd.edu

Yunong Shi
AWS Quantum Technologies
New York, USA
shiyunon@amazon.com

Abstract

Quantum Error Correction (QEC) is essential for fault-tolerant, large-scale quantum computation. However, error drift in qubits undermines QEC performance during long computations, necessitating frequent calibration. Conventional calibration methods disrupt quantum states, requiring system downtime and rendering *in situ* calibration impractical. To address this challenge, we propose QECali, a novel framework that enables *in situ* calibration for surface codes. Our evaluation demonstrates that QECali introduces modest qubit overhead and negligible increases in execution time, offering the first practical solution for *in situ* calibration in surface code based quantum computation.

CCS Concepts

• **Computer systems organization → Quantum computing**; • **Hardware → Quantum error correction and fault tolerance**.

Keywords

Quantum error correction, Qubit Calibration

ACM Reference Format:
Xiang Fang, Keyi Yin, Yuchen Zhu, Jixuan Ruan, Dean Tullsen, Zhiding Liang, Andrew Sornborger, Ang Li, Travis Humble, Yufei Ding, and Yunong Shi. 2025. CaliQEC: In-situ Qubit Calibration for Surface Code Quantum Error Correction. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture (ISCA '25), June 21–25, 2025, Tokyo, Japan.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3695053.3731042

ISCA '25, Tokyo, Japan

ACM ISBN 979-8-4007-1261-6/25/06
https://doi.org/10.1145/3695053.3731042

1 Introduction

Quantum Error Correction (QEC) [26, 42, 55, 59] is essential to enable large-scale Fault-Tolerant Quantum Computing (FTQC) suitable for practical applications [13, 60]. Among various QEC schemes, *surface codes* [8, 12, 16, 20, 44] have emerged as the leading solution and have been successfully implemented on various quantum hardware [1, 2, 9, 72]. Surface codes work by encoding *logical qubits* into redundant physical qubits arranged in a 2D square or hexagonal lattice, effectively reducing the error rate of the logical qubits if the physical error rate remains below a certain *threshold* [20]. With increasing code size, surface codes provide exponential suppression of logical errors. Recent experiments on superconducting devices [11, 32] demonstrated this error suppression for the first time [2], underscoring the practical applicability of surface codes.

Despite their demonstrated effectiveness, surface codes are highly sensitive to noise levels — small increases in physical error rates can require substantial expansion in code size to maintain the same logical error rate [21]. Consequently, maintaining a low and stable physical error rate is essential to ensure the effectiveness of surface codes during long computations. However, qubit and gate conditions degrade over time, leading to increased physical error rates—a phenomenon known as *error drift* [56, 71]. Fig. 1(a) shows the error drift observed on an IBM quantum computer [17]: after just one day, over 90% of single qubit gates exhibit error rates exceeding the threshold of surface codes. Theoretical estimates suggest that practical quantum applications are expected to run for hours or even days [7, 23, 40, 45], assuming a fixed error rate. However, in practice, error drift can significantly undermine computational reliability, making such long computations infeasible.

Calibration [35, 39, 65, 66, 69] is the process of fine-tuning quantum hardware to maintain low physical error rates and counteract error drift. Quantum hardware installations frequently calibrate their devices-often several times a day-to keep qubits and gates aligned with optimal conditions. Calibration involves three key steps: characterization, control parameter adjustment, and validation. Importantly, the calibration process disrupts quantum states, making qubits under calibration unavailable for computation.

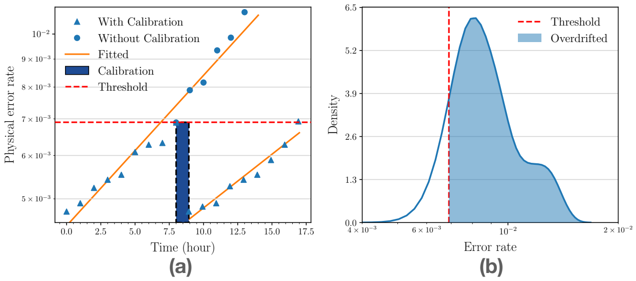


Figure 1: Error drift on an IBM's device: (a) Calibration maintains low physical error rates. (b) Without calibration, a majority of qubits exceed the threshold after 24 hours.

The fundamental challenge in surface code based quantum computation lies at the intersection of two competing requirements: maintaining consistently low physical error rates (through calibration) while simultaneously preserving quantum information for ongoing computation. This tension motivates our central research question: *How can we integrate runtime calibration with ongoing computation while maintaining the protection level of surface codes?*

A natural starting point draws inspiration from classical computing systems, where memory access must be coordinated with periodic DRAM refresh cycles [28]. Just as memory refresh temporarily interrupts access to maintain data integrity, qubit calibration requires periodic access to physical qubits while trying to preserve quantum information.

However, key differences exist between the classical and quantum settings. While classical memory can be refreshed at the bit level, quantum error correction creates a critical constraint: individual physical qubits cannot be naively removed for calibration without catastrophic error propagation. Simply mirroring classical strategies leads to a conservative approach, which we term *Logical Swap for Calibration* (LSC): transferring entire logical qubits to ancillary regions via logical SWAP operations before calibrating their physical components (Fig. 2c). This classically inspired approach suffers from two fundamental limitations. First, unlike classical bit copying, quantum logical SWAP operations require complex sequences of physical operations or quantum teleportation protocols [8, 44], introducing substantial overhead in both ancilla qubit count and execution time. Second, there is a critical granularity mismatch—calibration operates at the physical qubit level, while computation occurs at the logical qubit level. Since physical qubits drift at varying rates (Fig. 1b), LSC inefficiently forces entire logical qubits to pause when a single physical qubit requires calibration, leading to high overheads that scale with worst-case behavior. Other methods, such as [34], claim in-situ calibration but essentially rely on speculative estimation of control parameters rather than physical calibration. The derived parameters are often suboptimal and fail to meet the stringent physical error rates required for QEC.

To address these limitations, we propose **QECali** (Fig. 2d), a novel QEC framework that enables *in-situ* runtime calibration *concurrent* with ongoing computation, while maintaining the same level of QEC protection provided by surface codes. This approach eliminates the substantial overhead associated with quantum state transfer and ancillary qubit preparation required by LSC. Furthermore, it operates in a *fine-grained* manner, precisely isolating over-drifted physical qubits for calibration without stalling the entire logical qubit, effectively resolving the granularity mismatch issue of LSC.

The advantages of QECali are rooted in a key insight: *Surface codes allow for strategic runtime updates to their code structures, enabling qubit isolation while preserving logical information.* This is enabled by a theoretical tool known as *code deformation* [10, 67, 70]. Originally proposed for implementing logical operations on surface codes, we creatively repurpose it to address the conflicts between calibration and QEC-protected computation. Specifically, we apply the theory of code deformation to the most prevalent hardware architectures—square and hexagonal lattices [5, 17, 33]—and design novel code deformation instruction sets for these architectures (Sec. 6). These instruction sets enable two complementary operations: *selective qubit isolation*, which creates temporary boundaries to separate the calibrating qubits and program-running qubits, and *dynamic code enlargement*, which slightly expands affected patches to maintain QEC capabilities during the calibration process. QECali combines this instruction set with intelligent scheduling (Sec. 4, 5) to create an efficient, automated calibration system while preserving the original QEC protection level.

Our evaluation on both simulation and real quantum hardware demonstrates QECali's effectiveness across different architectures and use cases. On practical quantum benchmarks such as quantum chemistry applications, QECali maintains sub-threshold error rates with only 12-15% additional physical qubits and negligible impact on execution time, reducing retry risk by up to 85% compared to baselines. Component-wise analysis shows that QECali's adaptive scheduling achieves 91% reduction in calibration operations while minimizing space-time overhead. Our experiments on Rigetti and IBM processors validate QECali's key insight, showing that code deformation can effectively control error propagation.

In summary, this paper makes the following contributions:

- We propose a novel framework, **QECali**, that enables in-situ calibration and concurrent QEC-protected computation, achieving the desired retry risk level with minimal qubit count overhead and negligible execution time overhead.
- We design and formalize novel surface code deformation instruction sets for calibration on square and heavy-hexagon topologies supporting the QECali framework.
- We develop an adaptive scheduling method to efficiently coordinate calibration operations across physical qubits while effectively balancing various constraints.
- We introduce key device metrics—calibration times, drift rates, and crosstalk—to characterize quantum hardware, providing critical insights for optimizing calibration schedules and predicting performance.

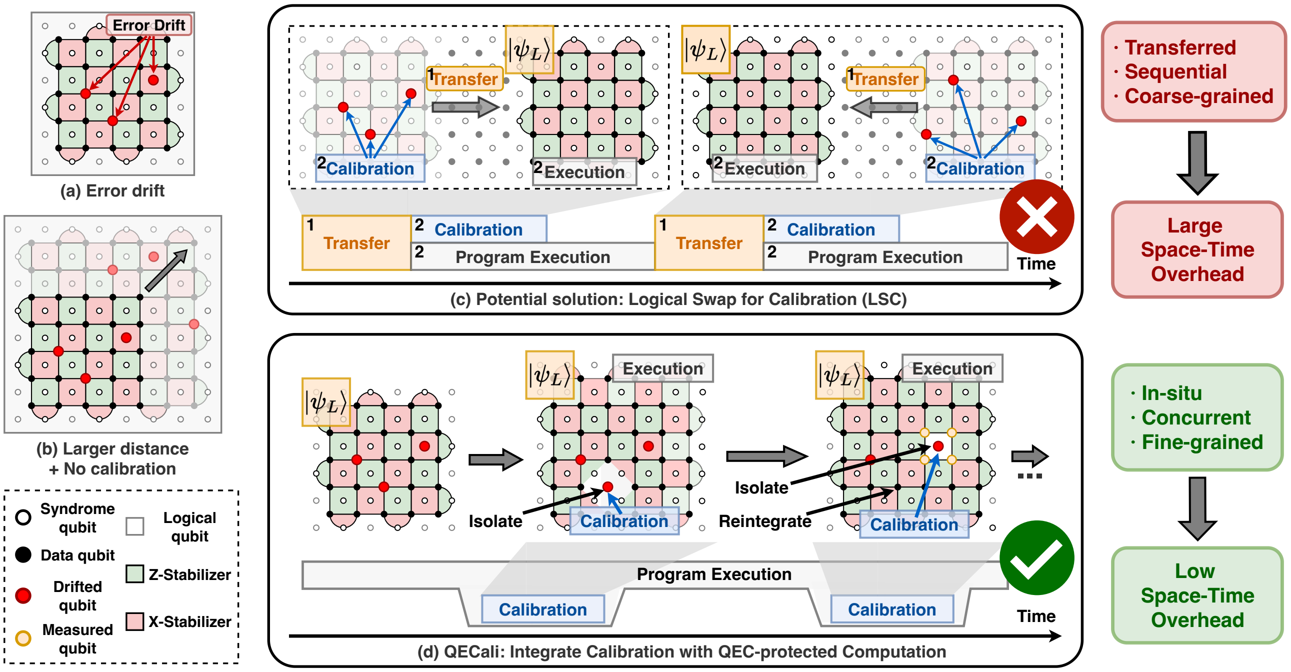


Figure 2: Comparison of QECali and LSC. (a) Error drift pattern in physical qubits. (b) Increased code distance approach. (c) LSC, a coarse-grained approach with high overhead. (d) QECali enables *in situ* calibration.

2 Background

This section provides an overview of surface codes and their deformation operations, and background on error drift and calibration.

2.1 FTQC with Surface Code

Surface Code Basics. The surface code is a leading candidate for realizing FTQC due to its simple 2D grid structure and high error threshold (1%) [12, 20]. Fig. 3a presents a typical surface code patch, where multiple physical qubits encode a single *logical qubit*. The physical qubits are divided into *data qubits* (black dots), which store the logical quantum state, and *syndrome qubits* (white dots), which collect error information from neighboring data qubits. The surface code can also be implemented on a hexagonal topology, as shown in Fig. 3b, which reflects the architecture of some current hardware platforms (IBM's devices [17]). Unlike in the square lattice, where each stabilizer uses a single ancilla qubit as the syndrome qubit, the stabilizers in the heavy-hexagon structure use seven ancilla qubits arranged in an "S" shape to form a bridge connecting the four data qubits, as shown in Fig. 3(c). This "S"-shaped arrangement provides the connectivity needed to extract error syndromes while mitigating the frequency crowding issue.

QEC with Surface Codes. Each syndrome qubit is linked to a *stabilizer* [24, 25, 55], a Pauli operator involving adjacent data qubits, as shown in Fig.3b,c. The red and green colors indicate the two stabilizer types, comprising exclusively X- or Z-Pauli operators, respectively. The syndrome qubits provide measurement outcomes of these stabilizers, producing *error syndromes* to guide error correction. However, some errors can alter the logical state without being detected by stabilizer measurements, known as *logical errors*, resulting in program failure. The *logical error rate* (LER) is closely tied to the *code distance*, defined as the minimum number of single-qubit errors required to cause a logical error. The code distance of a perfect surface code equals the number of data qubits along its edge (a distance-5 surface code in Fig. 3a). The LER decays exponentially with increasing code distance, provided the physical error rate remains below a certain threshold. For further detail, see [16, 20, 36].

FTQC with Surface Codes. Current surface code architectures [44] for FTQC arrange code patches on a plane with interspace equal to the code distance d, as shown in Fig.3(e). This interspace functions as a communication channel, enabling logical operations such as CNOT gates and state transfer (Fig. 3(e)(f)) via lattice surgery and code deformation [10, 14, 29, 31, 67]. Its width is set to d to maintain the same level of QEC protection as the code patches. For a detailed introduction, see [8, 19, 44].

2.2 Deformation of Surface Codes

Code deformation [10, 67, 70] is a key technique in surface codes that modifies the shape of the code patch to achieve specific functions. This subsection introduces four key deformation instructions that, when combined, isolate qubits from the code patch while preserving the encoded quantum information and QEC capability, enabling calibration without disrupting ongoing computation.

1. `DataQ_RM`. As illustrated in Fig. 4a, to remove the data qubit q, it combines the original stabilizers into larger "superstabilizers" that exclude q [6, 53, 62, 63]. These superstabilizers, along with the remaining stabilizers, enable QEC to continue on the code patch consisting of the remaining qubits during subsequent QEC cycles.

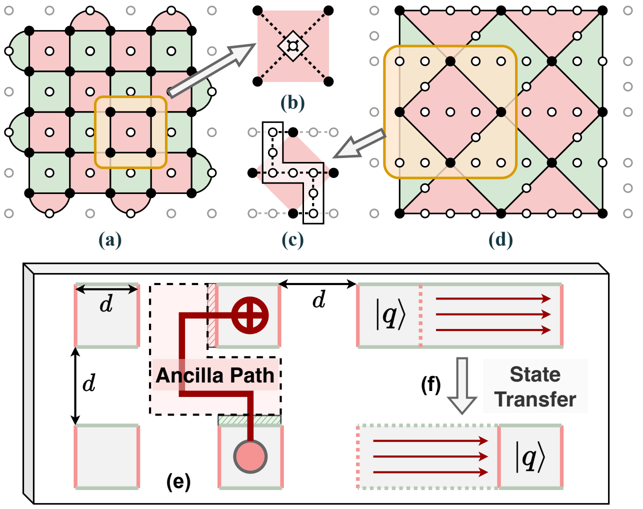


Figure 3: (a)(d) Surface codes on a square and hexagon lattice. (b)(c) Stabilizers of surface codes on two lattices. (e)(f) Layout for surface-code-based FTQC. Logical operations (e.g., logical CNOT in (e), logical state transfer in (f)) are enabled through the communication channel between the patches.

To reintegrate the data qubit q into the system, they are reset to state $|0\rangle$ or $|+\rangle$ and the stabilizers of the original code are measured, restoring the original structure.

2. SyndromeQ_RM. As shown in Fig. 4b, to remove the syndrome qubit q, the qubits involved in the stabilizer associated with q are measured in the X- or Z-basis, depending on the stabilizer type. The stabilizers are then updated to form "superstabilizers" that exclude q and are used in subsequent QEC cycles. Reintegrating the syndrome qubit into the system is accomplished by measuring the original stabilizers.

3. PatchQ_RM. This operation shrinks the code patch at the boundary by measuring the qubits to be excluded in either the Z- or X-basis, depending on the stabilizer they are associated with. To reintegrate the isolated qubits, they are reset to $|0\rangle$ or $|+\rangle$ and the original stabilizer (the red one in Fig. 4c) is measured.

4. PatchQ_AD. This operation expands the code patch at the boundary by preparing the qubits to be included in appropriate states ($|0\rangle$ or $|+\rangle$) and measuring the new stabilizer. In the example shown in Fig. 4d, the qubits are initialized in the $|0\rangle$ state, and a new X-stabilizer (red) is measured.

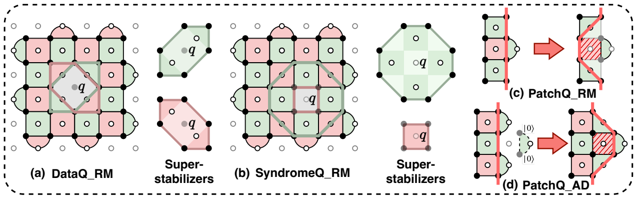


Figure 4: Four key deformation operations. (a) Data qubit removal, (b) Syndrome qubit removal, (c) Patch shrinkage at the boundary, and (d) Patch expansion at the boundary.

By combining DataQ_RM, SyndromeQ_RM, and PatchQ_RM, various patterns of drift-affected qubits can be isolated from the code patch. Repeated application of PatchQ_AD enables the desired code enlargement, restoring QEC capability after qubit isolation. Moreover, fault tolerance during code deformation is maintained using decoding approaches similar to those in regular surface codes [15, 18], with adjustments for dynamically changing stabilizers and syndrome transitions [53, 64, 67, 68]. Optimized decoders can efficiently handle these changes, ensuring minimal impact on decoding time [14, 43]. For a more comprehensive introduction to the deformation framework in surface codes, refer to [10, 67, 70].

2.3 Error Drift and Calibration

Error Drift. QEC schemes often assume a static error model where error rates remain constant. However, on real hardware, error rates fluctuate over time, potentially exceeding the QEC threshold—a phenomenon known as *error drift* [56, 58, 71]. In superconducting devices, a primary cause of error drift is the unwanted coupling of qubits to *two-level systems* (TLS) [38, 49, 51], along with thermal fluctuations in Josephson junctions [27], unpaired electrons [28, 38], and environmental noise [48, 50]. TLS defects, inherent to the fabrication process, occur randomly and are difficult to characterize or predict, leading to unique and heterogeneous impacts on each qubit. These factors cause qubit parameters (e.g., T_1 and T_2 coherence times) to vary unpredictably, necessitating frequent characterization and recalibration to maintain optimal performance.

Calibration. *Calibration* [5, 35, 37, 41, 46, 54, 57, 66, 69] is the process of readjusting control parameters (such as the duration and frequency of control pulses) for qubit operations to maintain optimal performance, specifically, low error rates. This process is lengthy, often taking several hours (e.g., 4 hours in [5]) to ensure accuracy. Moreover, the calibration order for different qubits must be carefully scheduled to mitigate crosstalk issues [52]. Importantly, calibration requires qubits to be in specific states for measurement, preventing them from storing information or supporting computation during calibration. As a result, calibration typically halts the program, making concurrent calibration and computation impossible and limiting the duration of reliable computations.

3 Overview

QECali achieves efficient in-situ calibration through three coordinated stages, as depicted in Fig. 5:

Preparation time. This stage thoroughly characterizes the quantum hardware, capturing key parameters for each gate, including *calibration duration*, *drift rate*, and *calibration crosstalk* (Section 4). This foundational data serves as a basis for optimizing our calibration strategies.

Compilation time. This stage aims to determine all the calibration processes based on the hardware characterization obtained during preparation stage. This stage consists of two steps.

Drift-based Calibration Grouping (Sec. 5.2). Based on gate parameters, we define calibration workloads by determining each gate's frequency and duration. Gates with similar drift characteristics are grouped into calibration intervals to avoid overlapping workloads, which could cause excessive distance loss from deformation.

Intra-Group Calibration Scheduling (Sec. 5.3): Within each interval, we sequence the calibration workloads based on their dependencies

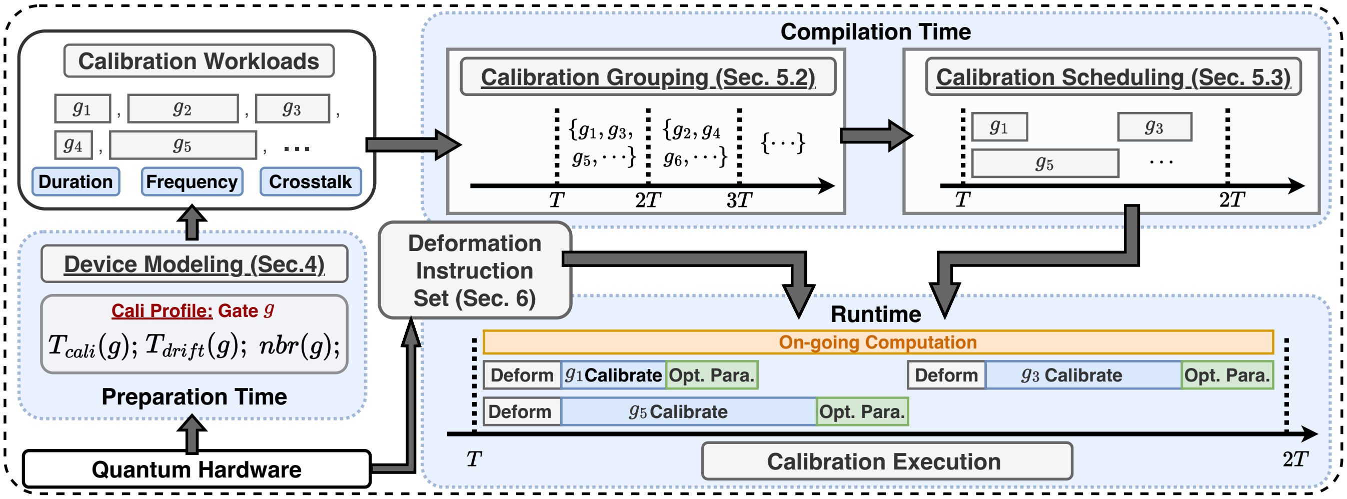


Figure 5: Overview of QECali: a deformation-based calibration framework.

and crosstalks. This scheduling minimizes the total calibration time while maintaining computational efficiency.

Runtime. The runtime follows the calibration schedule generated during compile time (Section 5), triggering the corresponding calibration operations for designated gates at specified intervals. For each gate, it executes the associated code deformation instructions from the QECali instruction set (Section 6). Meanwhile, logical computations on deformed logical qubits continue uninterrupted, with all operations pre-calculated during compile time.

4 Preparation-time Device Characterization

To enable optimized calibration scheduling, QECali first characterizes the quantum device pre-compilation by measuring and extracting key metrics of each qubit operations. These include:

Calibration Time (T_{cali}): We measure each gate's calibration duration through repeated experiments, as this directly impacts scheduling efficiency. Typically, individual gate calibration takes a few minutes, while full-device calibration spans several hours [4, 37].

Drift Rate (T_{drift}): We characterize the error drift of a gate g using the following exponential scaling model:

$$p(g,t) = p_0[g] \cdot 10^{t/T_{\text{drift}}[g]} \tag{1}$$

where $p(g,t)$ is the error rate at time t, $p_0[g]$ is the initial error rate, and $T_{\text{drift}}[g]$ is the drift time constant, representing the time required for the error rate to increase tenfold. We adopt this exponential model as it best fits our experimental data from IBM's real machine (Fig.1) and is consistent with prior studies [56, 57], although some references report a linear drift model [4]. Specifically, we perform hourly measurements using the *interleaved randomized benchmarking* method [47], conducting three sets of tests with $[1, 10, 20, 50, 100, 150, 250, 400]$ repetitions. The resulting gate error data is fitted to Eqn. (1) to determine T_{drift}. Notably, this model can be replaced with other models based on specific hardware conditions and determine calibration periods for each gate accordingly, while the scheduling method in Sec. 5 remains applicable.

Calibration Crosstalk ($\text{nbr}(g)$): We introduce a new method to identify qubits affected by each gate's calibration using the circuit in Fig. 6. For each gate g, we initialize nearby qubits to random states, perform calibration, and measure their final states. Qubits exhibiting deviations beyond a threshold are added to $\text{nbr}(g)$, indicating crosstalk interference. Importantly, these qubits $\text{nbr}(g)$ are *isolated along with the calibrating qubit* during calibration, creating a protective barrier between calibrating and program-running qubits. After calibration, the isolated qubits are reset to $|0\rangle$ or $|+\rangle$ before reintegration into the system (Sec. 2.2), ensuring that calibration does not interfere with ongoing computations.

These metrics form the foundation for QECali's compilation-time scheduler (Section 5) and runtime execution engine, enabling generation of calibration schedules that optimize the critical trade-offs between frequency, parallelism, and interference. Moreover, a full calibration process is usually conducted before the program begins in practice, allowing device characterization parameters to be obtained. This makes this preparation stage an integral part of the overall computation process without adding extra time.

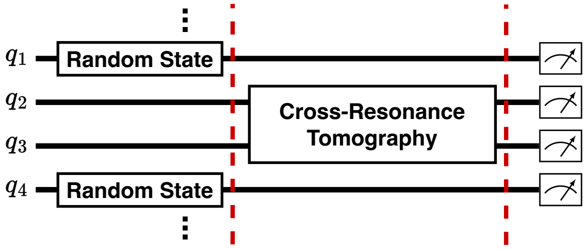


Figure 6: Circuit characterizing calibration crosstalk.

5 Compilation-time Calibration Scheduling

5.1 Problem Formulation

For reliable execution of large-scale quantum programs, we must maintain the logical error rate (LER) below a target value (say LER_{tar}) throughout computation. Formally, this requires $\text{LER}(t) \leq$

LER_{tar} for all t. The calibration schedule directly impacts this requirement by influencing the physical error rates: frequent calibration helps maintain lower physical error rates, which in turn supports a lower LER.

For a distance-d surface code to achieve the target LER_{tar}, the average physical error rate must not exceed a corresponding target, denoted as p_{tar}. Thus, each gate g must be calibrated within its drift time $T_{\text{drift},p_{\text{tar}}}[g]$—the time it takes for g's error rate to reach p_{tar}. To meet the reliability requirement, the calibration interval for g must satisfy $T_g \leq T_{\text{drift},p_{\text{tar}}}[g]$. The drift time $T_{\text{drift},p_{\text{tar}}}[g]$ can vary significantly depending on the specific gate and hardware characteristics, ranging from hours to days.

While frequent calibration is necessary for maintaining low LER, parallel calibration comes with substantial costs. Moreover, calibration-induced crosstalk between neighboring qubits further constrains the degree of parallelization possible.

To balance these competing requirements, we formulate our optimization objective as:

$$\min \sum_g \frac{1}{T_g} \quad \text{subject to} \quad \underbrace{T_g \leq T_{\text{drift},p_{\text{tar}}}[g],}_{\text{drift constraint}} \quad \underbrace{|C_t| \leq 1, \forall t}_{\text{crosstalk constraint}}$$

where T_g is the calibration period for gate g, C_t is the set of gates in C being calibrated at time t, and C represents a set of gates that cannot be calibrated simultaneously due to crosstalk. It aims to minimize calibration frequency, for reducing qubit overhead, while respecting both drift time limits and crosstalk constraints.

5.2 Drift-based Calibration Grouping

To solve this problem, we propose a heuristic solution that achieves high optimization quality with fast compilation speed. This compilation is performed before the program runs, taking only a few seconds, which is negligible compared to the program's execution time. Our approach groups gates with similar drift characteristics to share calibration intervals, effectively discretizing the scheduling space while respecting device physics. Specifically, we select a base calibration interval kT_{Cali} and assign each gate g to a calibration group k_g according to:

$$k_g \cdot T_{\text{Cali}} \leq T_{\text{drift},p_{\text{tar}}}[g] < (k_g + 1) \cdot T_{\text{Cali}} \tag{2}$$

This grouping strategy offers several advantages. First, gates within group k_g share the same calibration cycle $T_g = k_g \cdot T_{\text{Cali}}$, simplifying scheduling. Once the grouping is completed, the schedule for each group is determined: during the k-th interval, only gates belonging to Group k_g are executed (where $k \mod k_g = 0$). Second, this grouping enables opportunities for parallel scheduling of calibrations, as all gates within a group can be scheduled at any time within their assigned interval T_{Cali}. Third, it simplifies the issue of crosstalk, as we only need to address crosstalk within a single group. Crosstalk between groups is avoided since groups assigned to the same interval are trivially scheduled sequentially. In contrast, a naive approach without grouping can lead to crosstalk issues. As shown in Fig. 7(a), this method calibrates each gate individually until it reaches the error threshold. While it minimizes the overall calibration frequency, it may schedule gates with crosstalk interference simultaneously, causing conflicts. In summary, the resulting calibration frequency is:

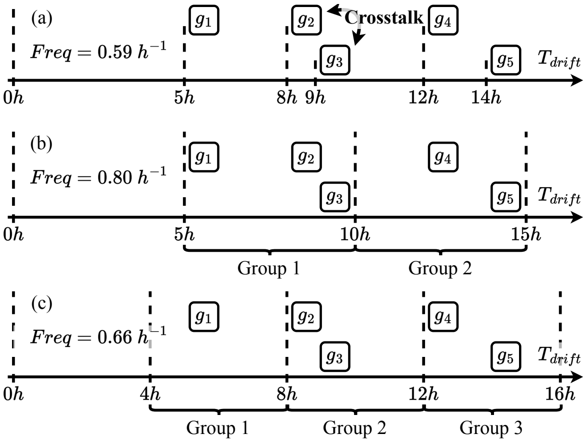


Figure 7: Impact of T_{Cali} on calibration frequency.

$$\sum_g \frac{1}{T_g} = \frac{1}{T_{\text{Cali}}} \sum_k \frac{n_k}{k} \tag{3}$$

where n_k denotes the number of gates in the k-th group.

Optimal Choice of Group Duration T_{Cali}: The choice of base calibration interval T_{Cali} significantly impacts grouping efficiency. While one might intuitively set T_{Cali} to the minimum drift time $\min_g T_{\text{drift},p_{\text{tar}}}[g]$, this often leads to suboptimal groupings. Consider the example in Fig. 7(b): with $T_{\text{Cali}} = 5h$, gates g_1, g_2, and g_3 form Group 1 while gates g_4 and g_5 form Group 2, resulting in 0.80 calibrations per hour. However, setting $T_{\text{Cali}} = 4h$, though increasing g_1's calibration frequency, enables better distribution of other gates into groups with lower frequencies, reducing overall cost to 0.66 calibrations per hour Fig. 7(c). To clarify, the calibration tasks will be executed periodically according to T_{Cali}. Fig. 7 illustrates different grouping strategies but does not depict the scheduling of calibration tasks.

Algorithm 1: Calibration Group Assignment

```
Input: Gate set $G$, Drift time $T_{\text{drift},p_{\text{tar}}}[g]$
Output: Calibration groups $Group[k]$
$T_{\min} = \min_g T_{\text{drift},p_{\text{tar}}}[g]$
$T_{\text{Cali}} = T_{\min}$
for $g \in G$ do
    $k = \lceil T_{\text{drift},p_{\text{tar}}}[g] / T_{\min} \rceil$
    $T = T_{\text{drift},p_{\text{tar}}}[g] / k$
    if $Freq(G, T_{Cali}) > Freq(G, T)$ then
        $T_{\text{Cali}} = T$
    end
end
Initialize $Group[k]$
for $g \in G$ do
    $k = \lfloor T_{\text{drift},p_{\text{tar}}}[g] / T_{\text{Cali}} \rfloor$
    Add $g$ into $Group[k]$
end
return $Group[k]$
```

The optimal T_{Cali} tends to occur when some $T_{\text{drift},p_{\text{th}}}[g]$ values align with integer multiples of T_{Cali}. If alignment does not occur, T_{Cali} can be increased without altering the grouping, thereby reducing the calibration frequency as described in Eq. (3). To determine the optimal value, we employ Algorithm 1, traversing the values of $T_{\text{drift},p_{\text{tar}}}[g]/k$ for each gate, particularly those slightly smaller than the minimum drift time T_{min}. The T_{Cali} that minimizes the calibration frequency is then selected. In cases where multiple intervals yield similar or identical calibration frequencies, a larger interval is preferred. A larger T_{Cali} allows for grouping more gates together, providing longer scheduling windows, increased flexibility for intra-group scheduling, and greater opportunities for parallelism.

Targeted Physical Error Rate Determination: Algorithm 1 assumed that the targeted physical error rate, p_{tar}, was known. Here we describe how our compiler determines p_{tar} based on the available physical qubit resources. Surface codes with larger code distances possess stronger error correction capabilities, allowing them to tolerate higher p_{tar} while maintaining the same targeted logical error rate (LER_{tar}).

The LER for a distance-d surface code is given by [19]:

$$\text{LER}(d, p_{\text{tar}}) = \alpha \left(\frac{p_{\text{tar}}}{p_{\text{th}}}\right)^{(d+1)/2}, \tag{4}$$

where α is a constant specific to the quantum error correction (QEC) code, typically around 0.03 for the rotated surface code. Here, p_{th} represents the physical error threshold for the surface code, approximately 0.01 under the circuit-level noise model.

To guarantee program fidelity, the condition $\text{LER} \leq \text{LER}_{\text{tar}}$ must be satisfied. With Eq. (1) and Eq. (4), this condition can be simplified:

$$\left(\log \frac{p_{\text{th}}}{p_0} - \log \frac{p_{\text{tar}}}{p_0}\right) \cdot d \geq \lambda, \tag{5}$$

where λ is a positive constant dependent on LER_{tar} and α. Importantly, this condition can only be satisfied if $p_{\text{tar}} < p_{\text{th}}$, which aligns with the intuitive requirement that no gate's error rate should exceed the physical error threshold during program execution.

Given a fixed number of physical qubits, the compiler calculates the maximum allowable code distance d that fits within the resource constraints. It then determines the largest p_{tar} that satisfies the target LER_{tar} while ensuring $\text{LER}(t) \leq \text{LER}_{\text{tar}}$ throughout program execution. This approach balances the trade-offs between code distance, p_{tar}, and physical qubit resources. Larger code distances exponentially suppress logical error rates but require more physical qubits. Conversely, higher p_{tar} allows for longer drift times and less frequent calibration but relies on more robust error correction. By leveraging these trade-offs, our compiler optimizes p_{tar} and d to achieve reliable and resource-efficient program execution.

5.3 Intra-Group Calibration Scheduling

After assigning calibrations to each time period T_{Cali} and determining which gates should be calibrated simultaneously, the next step involves finding an appropriate code deformation process to isolate these gates. Generally, for each gate, we apply code deformer to isolate all its affected neighboring qubits, enabling a region suitable for calibration. However, performing this sequentially may lead to excessive calibration time overhead, potentially causing the calibration time of a gate, t_{cali}, to exceed the calibration cycle T_{Cali}.

In this section, we analyze three challenges associated with calibration scheduling and propose an adaptive calibration scheduling approach to address them.

(1) Dependence between calibrations: Certain two-qubit gate calibrations may depend on the results of one-qubit gate calibrations. We address this by clustering such gates and scheduling their calibration collectively. This dependency typically arises from overlapping positions, i.e.,c their neighboring qubits $nbr(g)$ are highly overlapped. Code deformation to isolate these shared qubits facilitates the simultaneous calibration of multiple gates in the cluster.

(2) Crosstalk between calibrations: Crosstalk during calibrations prevents all calibrations from running simultaneously. To maximize calibration parallelism and minimize calibration time, we design a greedy scheduling. This approach sorts gate calibrations based on the size of their affected qubits. Starting with the largest, we select as many calibrations as possible without introducing crosstalk. When no more gates can be calibrated concurrently, we generate code deformation instructions for the selected gates and begin a new batch. This process is repeated until all gates are calibrated.

(3) Trade-off between calibration time and code distance loss: Code deformation reduces the code distance. To preserve the fidelity of the original code, the code must be enlarged to restore the original code distance. Calibrating more gates simultaneously requires isolating larger regions, leading to greater distance loss and increased physical qubit overhead for enlargement. To balance this trade-off, we define a new metric to evaluate scheduling: the space-time overhead of the enlarged region: $\text{Cost} = \Delta d \sum_g t_{\text{cali}}[g]$. For each Δd, we calculate the cost by constraining the greedy scheduling strategy to ensure that the code deformation for simultaneous calibrations does not exceed the maximal tolerable distance loss Δd. The optimal scheduling is then chosen based on this evaluation.

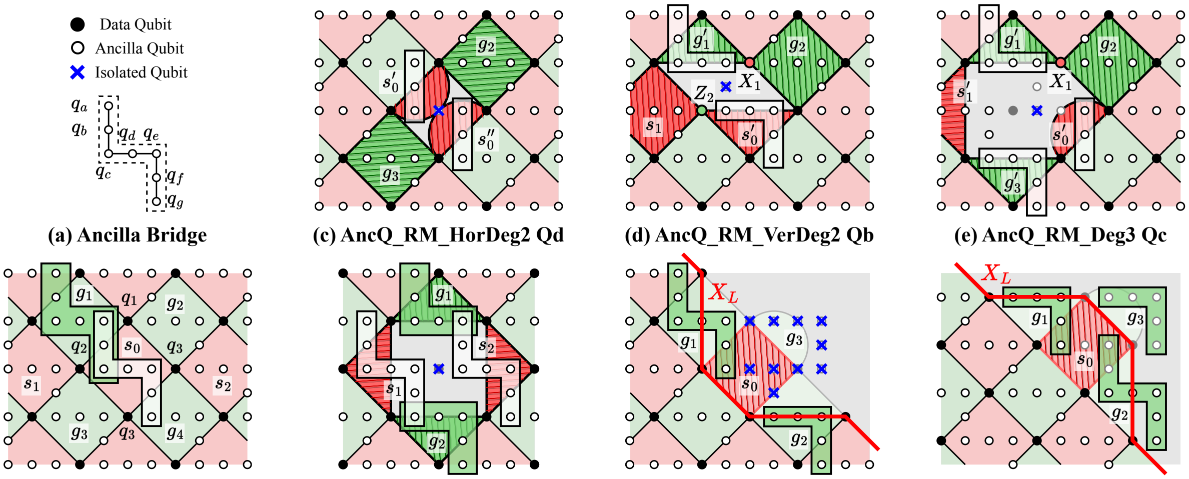
6 The QECali Instruction Set

A critical challenge in runtime calibration is the ability to isolate physical qubits for calibration without compromising the surface code's error correction capability. To address this challenge, QECali provides carefully designed instruction sets that enable safe transformation of the code structure during calibration.

Table 1 provides an overview of the two instruction sets. For surface codes on square lattices, QECali adopts the instruction set from [70], originally designed for handling defective qubits in surface codes. We identified that these instructions—`DataQ_RM`, `SyndromeQ_RM`, `PatchQ_RM`, and `PatchQ_AD`—can be repurposed for calibration, enabling selective qubit isolation while preserving error correction properties. For the details of these instructions, we refer readers to Section 2.2 or [70].

6.1 The QECali Instruction Set for the Heavy-Hexagon Topology

Modern quantum processors, including IBM's devices, utilize a heavy-hexagon topology (Fig. 3(d)). Existing code deformation instructions are designed for square lattices and cannot be directly applied to heavy-hexagon architectures. However, since square lattice instructions are fundamentally based on gauge fixing theory [10, 67], this theory provides a foundation for adapting deformation techniques to heavy-hexagon structures. Building on an



(a) Ancilla Bridge (c) AncQ_RM_HorDeg2 Qd (d) AncQ_RM_VerDeg2 Qb (e) AncQ_RM_Deg3 Qc

(b) Hev-Hex Surface Code (f) DataQ_RM Q0 (g) PatchQ_RM S0, XL (h) PatchQ_RM S0, XL

Figure 8: Code deformation instruction set for the surface code on the heavy-hexagon structure.

in-depth study of gauge fixing, we have designed and formalized a dedicated instruction set specifically for heavy-hexagon topologies.

Table 1: QECali instruction sets for square and heavy-hexagon surface codes

Code Topology	Instructions
Square	`DataQ_RM`, `SyndromeQ_RM`, `PatchQ_RM`, `PatchQ_AD`
Heavy-Hexagon	`DataQ_RM`, `AncQ_RM_HorDeg2`, `PatchQ_RM`, `AncQ_RM_VerDeg2` `PatchQ_AD`, `AncQ_RM_Deg3`,

Key Distinctions of the Topology: The heavy-hexagon topology has two key features that render the square lattice instructions inadequate:

1. Non-uniform ancilla roles: As shown in Fig. 8(a), the seven ancilla qubits associated with each stabilizer can be classified into two distinct types: (1) Degree-3 nodes (q_a, q_c, q_e, q_g), which connect to three other qubits, including one data qubit. (2) Degree-2 nodes (q_d, q_b, q_f), which connect to two other ancilla qubits. Degree-3 nodes directly connect to data qubits, while degree-2 nodes act as bridges to link them. This functional difference necessitates different code deformation instructions to ensure the deformed code remains functional after isolating specific ancilla qubits.

2. Shared ancilla qubits: As shown in Fig. 8(a), ancilla qubits associated with one stabilizer can also be shared with others. For instance, in Fig. 8(b), the X-stabilizer s_0 and the Z-stabilizer g_1 share three ancilla qubits. This overlap implies that isolating a single ancilla qubit may impact multiple stabilizers simultaneously which makes designing code deformation instructions under these circumstances becomes more complex.

Design Principles: The heavy-hexagon structure presents unique opportunities for code deformation compared to the square lattice. We outline the key principle that guided the design of our instruction set — *leveraging residual connectivity:* when an ancilla qubit is removed, the remaining structure often retains partial connectivity between data qubits. By utilizing this residual connectivity, we replace the original stabilizer with a product of smaller, localized measurements. This approach minimizes disruption to the overall code structure and preserves a greater number of stabilizers, ensuring robust error correction.

The Instructions: Building on the design principles outlined above, we redesign three square lattice instructions to adapt them to the heavy-hexagon architecture. Additionally, we introduce three distinct instructions of `AncQ_RM` category, each specifically tailored to remove ancilla qubits based on their unique positions and connectivity within the heavy-hexagon device.

(1) `AncQ_RM_HorDeg2`: This instruction targets a degree-2 horizontal ancilla qubit q_d, as illustrated in Fig. 8(c). Removing q_d divides the X-stabilizer s_0 into two parts, $s_0' = X_{1,2}$ and $s_0'' = X_{3,4}$, forming two new gauge measurements that replace the original stabilizer $s = s_0's_0''$. Additionally, the removal transforms the nearby Z-stabilizers g_2 and g_3 into new gauges, which combine to form a new Z-super-stabilizer g_2g_3.

(2) `AncQ_RM_VerDeg2`: This instruction removes a degree-2 vertical ancilla qubit q_b, as shown in Fig. 8(d). Unlike the horizontal case, q_b is shared by the X-stabilizer s_0 and the Z-stabilizer g_1. Removing q_0 divides s_0 into a three-qubit gauge $s_0' = X_{2,3,4}$ and a single-qubit gauge X_1. Similarly, g_1 is divided into a three-qubit gauge $g_1' = Z_{5,6,1}$ and a single-qubit gauge Z_2. These divisions also affect the nearby Z-stabilizer g_2 and X-stabilizer s_1, transforming them into new gauges. Collectively, these changes result in a new X-super-stabilizer $X_1s_0's_1$ and a new Z-super-stabilizer $Z_2g_1'g_2$.

(3) `AncQ_RM_Deg3`: This instruction removes a degree-3 ancilla qubit q_c, as illustrated in Fig. 8(e). Similar to `AncQ_RM_VerDeg2`, q_e is shared by the X-stabilizer s_0 and the Z-stabilizer g_1. The removal of q_0 leaves g_1 unchanged, dividing it into two parts: $g_1' = Z_{5,6,1}$ and a single-qubit gauge Z_2. However, removing q_0 divides s_0 into three components: a two-qubit gauge $s_0' = X_{3,4}$ and two single-qubit gauges X_1 and X_2. After this division, both X_2 and Z_2 exist as single-qubit gauges for qubit q_2. This indicates that q_2 becomes a gauge qubit isolated from the surface code, which should be removed. The removal of q_2 further impacts the nearby Z-stabilizers g_2 and g_3, as well as the X-stabilizer s_1, deforming and transforming them into new gauges. Collectively, these modifications result in a new X-super-stabilizer $X_1 s_0' s_1'$ and a new Z-super-stabilizer $g_1' g_2 g_3'$.

(4) `DataQ_RM`, `PatachQ_RM`, `PatachQ_ADD`: These three instructions deform the surface code in a manner similar to those for the square lattice (Fig. 4), ensuring that the stabilizers in the deformed code remain unchanged. However, because the heavy-hexagon surface code features unique ancilla bridges in its stabilizer circuits, it is also necessary to deform the ancilla bridges associated with the affected stabilizers during the deformation process.

The new instruction set not only make the deformation compatible with heavy-hexagon structures but also leverage the structure of the ancilla bridge to create more fine-grained strategies.

7 Experimental Setup

7.1 Setting and benchmark

Evaluation setting. We evaluate QECali through both hardware experiments and simulation-based analysis. Our hardware experiments are conducted on two quantum processors with distinct topologies: Rigetti's Ankaa-2 processor with square lattice connectivity and IBM's Eagle processor with heavy-hex connectivity. For large-scale logical error analysis, we employ the Stim quantum error simulator for surface code simulation [22] along with Pymatching [30] for error correction. Program compilation utilizes the lattice surgery framework [29] and implements logical T gates through magic state distillation [19].

Benchmark programs. We evaluate QECali using quantum programs designed for most promising applications in quantum chemistry and materials science. Our benchmarks include Hubbard model simulation [3], which provides essential insights into strongly correlated electronic systems with direct applications to high-temperature superconductivity; FeMo-co catalyst analysis [40] , which addresses the critical industrial challenge in nitrogen fixation; and Jellium simulation [61] , which serves as a fundamental model for understanding electronic structure in materials. Program variants are denoted by suffixes indicating problem size (e.g., Hubbard-16 for a 16-qubit system).

Metric. We evaluate QECali using four metrics. The *physical qubit count* encompasses all qubits required for the quantum program, including data qubit blocks for logical encoding, ancilla qubits for CX operations, and resource states for T gate implementation. *Execution time* measures the total runtime for program completion, with QEC cycle time set to $1\mu s$ (standard in FTQC studies [10, 35, 52, 56]), including all quantum operations and error correction cycles. *Logical error rate (LER)* represents the probability of logical errors occurring per quantum error correction (QEC) cycle for a logical surface code qubit. It reflects the effectiveness of error correction for physical errors, with a lower LER indicating superior fault-tolerant performance. *Retry risk* [23] quantifies the probability of encountering uncorrectable logical errors, providing a measure of program reliability and the likelihood of requiring computation restart. In general, it's computed by LER multiplied with the total number of logical operations.

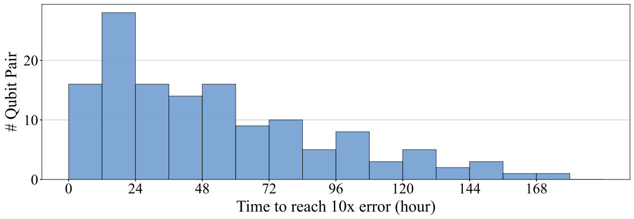


Figure 9: Probability distribution of error drift.

7.2 Error model

Physical error model. We adopt a standard circuit-level noise model [20, 22] where quantum operations are subject to different error channels: single-qubit gates experience depolarizing errors, two-qubit gates undergo two-qubit depolarizing errors, and measurement and reset operations are affected by Pauli-X errors, each with probability p. Initially, all operations start with a uniform error rate $p = 10^{-X}$, where X is chosen to be 10× below the surface code threshold (1%). This initialization represents an ideally calibrated device state.

Error Drift Model. Based on our measurements of IBM's 127-qubit Eagle processor, we observe that physical error rates increase exponentially over time, following the relation:

$$p(G, t) = p(G, 0)10^{t/T(G)}$$

where $p(G, t)$ is the error rate of operation or qubit G at time t, $p(G, 0)$ is the initial calibrated error rate, and $T(G)$ is the operation-specific drift time constant—the time required for the error rate to increase by a factor of 10. Our characterization shows that these drift time constants vary significantly across the device, following a log-normal distribution with a mean of 14.08 hours (Fig. 9). This heterogeneity stems from variations in qubit connectivity, gate implementation complexity, and device characteristics, necessitating individual calibration schedules for different device components.

Future Error Model. We account for potential advancements in hardware technology that may slow error drift or, equivalently, extend the constant $T(G)$. The reference [11] indicate that the physical fidelity of superconducting devices can be improved from 99.9% to 99.99%. We assume that the error drift effect will improve proportionally, leading to a longer calibration period. Specifically, a one-order-of-magnitude improvement in error rate translates to a doubling of the calibration duration. To model this, we assume a future error scenario where $T(G)$ follows a log-normal distribution with a doubled mean of 28.016. Our framework is evaluated under this future error model to demonstrate its adaptability and continued utility in evolving hardware environments.

7.3 Baseline Assumptions

In this section, we introduce the key assumptions of two baselines and our framework QECali.

Baseline 1. No calibration. In this case, benchmarks run without any calibration. This approach minimizes qubit resources and execution time, but leads to an expected retry risk approaching 100% due to error drift.

Baseline 2. LSC. In this case, whenever a gate within a surface code patch requires calibration, the logical state is transferred elsewhere and moved back to its original location once calibration is complete (Sec. 2.1). To accommodate these logical state transfers while ensuring uninterrupted computation, LSC has to expand the communication channels in both dimensions within the 2D surface-code-based FTQC architecture (Sec. 2.1). This results in a roughly 4x qubit overhead compared to Baseline 1. While this estimate may be somewhat conservative, reducing this qubit overhead would either extend execution time due to limited channel availability or necessitate complex scheduling strategies. Consequently, we adopt a straightforward 2D expansion in LSC.

QECali. Our framework adopts the same layout as Baseline 1 but increases the interspace by Δd to accommodate potential patch enlargement during calibration, ensuring QEC capability is maintained. Here, Δd represents the maximum tolerable distance loss, set to 4 in our experiments. This allows for either four single-qubit isolations or the isolation of a larger region with a diameter of 4 qubits, depending on the crosstalk-affected qubits $\mathrm{nbr}(g)$ identified during device characterization (Sec. 4). At runtime, gates are calibrated according to the pre-determined schedule (Sec. 5) using code deformation instructions. To preserve the target LER, the code patch is dynamically enlarged to compensate for distance loss. These operations occur concurrently with computation, introducing negligible execution time overhead. While scheduling calibrations earlier could prevent exceeding the target LER without patch enlargement, accurately predicting the impact of distance loss on LER and determining the optimal timing remains challenging, especially with irregular surface code structures. Therefore, we opted for the current approach.

8 Evaluation

In this section, we evaluate QECali by comparing with two baselines, analyze the effects of individual components, and perform experiments on real systems.

8.1 Overall Performance

We evaluated QECali against two baseline approaches: running without calibration and using Logical Swap for Calibration (LSC), under both the current and future error models described in Sec. 7.2. Our experiments, presented in Table 2, use surface codes with distances chosen to achieve target retry risk levels of 1% and 0.1% and benchmarks described in Section 7. Our evaluation reveals four critical observations:

1. Calibration is indispensable: attempting to run quantum programs without calibration leads to retry risks approaching 100%, demonstrating the severe impact of error drift. While programs start with low logical error rates, exponential drift in physical error rates quickly compromises computation reliability, making successful execution virtually impossible for long-running applications.

2. Coarse-grained calibration is impractical: Compared to the solution with no calibration (Table 2), the LSC approach reduces the retry risk to the target level but incurs a substantial 363% increase in qubit count and a 20% longer execution time. These inefficiencies arise from LSC's need for a 2D layout expansion to enable logical state transfer (Sec. 7.3), along with execution delays from logical SWAPs and program stalls during calibration. Its coarse-grained approach, dictated by the worst-performing qubits, further exacerbates overhead as system size increases.

3. QECali achieves low overhead calibration: Compared to the LSC solution, QECali sustains computation progress during calibration, dramatically reducing the 363% qubit overhead of LSC to just 24% and eliminating execution time overhead entirely. By performing in-situ calibration via code deformation, QECali avoids the computational stalls and ancilla overhead associated with state transfer approaches. Furthermore, QECali reduces retry risk by 79.4% compared to LSC, demonstrating the effectiveness of its fine-grained calibration strategy in preserving the system's error correction capabilities and suppressing the retry risk.

4. *In situ* calibration remains essential even with improved hardware: The lower half of Table 2 compares QECali with two baselines under the future error model. In this scenario, LSC still incurs a significant qubit count overhead (363%) because the physical error rate eventually exceeds the threshold, necessitating logical state transfers for calibration and requiring excessive additional qubit resource. While slower error drift reduces the frequency of calibration, LSC still suffers from a 304% higher retry risk, despite a modest increase in execution time. This analysis highlights that even with reduced error drift in future systems, large-scale quantum tasks will still require *in situ* calibration. The log-normal distribution of drift time constants $T(G)$ across gates ensures that some physical qubits will remain more vulnerable to drift. Our experiments confirm that even a small number of underperforming qubits can significantly increase logical error rates, reinforcing the importance of runtime calibration for reliable quantum computation.

8.2 Component-wise Analysis

We conducted a detailed analysis of QECali's key components to quantify their individual contributions to overall system performance. This ablation study focuses particularly on our adaptive calibration scheduling and resource management strategies.

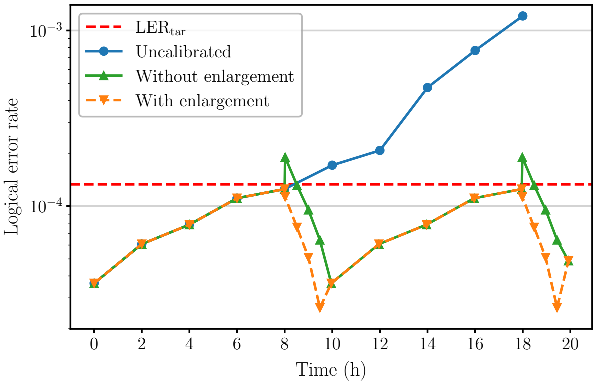


Figure 10: d=11 Logical error rate analysis with error drift.

8.2.1. In Situ Calibration's Impact on LER

We evaluate the impact of the two critical deformation steps in our calibration framework: qubit isolation and code enlargement.

In Fig. 10, we simulate the LER dynamics during calibration cycles for a $d = 11$ surface code. The red line represents the LER

Table 2: Comparison of performance for large scale programs

Error drift Model	Benchmark					No Calibration			LSC			QECali		
	Name	# *CX*	# *T*	# logical qubit	d	# physical qubit	Execution time (hour)	Retry risk	# physical qubit	Execution time (hour)	Retry risk	# physical qubit	Execution time (hour)	Retry risk
Current model	Hubbard -10-10	1.64×10^9	7.10×10^8	200	25	9.81×10^5	5.29	$\sim$ 100%	4.65×10^6	5.74	11.3%	1.53×10^6	5.29	3.13%
					27	1.14×10^6	5.50	$\sim$ 100%	5.43×10^6	5.95	1.22%	1.62×10^6	5.50	0.38%
	Hubbard -20-20	5.3×10^{10}	1.2×10^{10}	800	29	5.28×10^6	91.3	$\sim$ 100%	2.30×10^7	101.5	7.35%	7.11×10^6	91.3	1.88%
					31	6.03×10^6	94.3	$\sim$ 100%	2.63×10^7	108.5	0.79%	8.38×10^6	94.3	0.20%
	jellium -250	8.23×10^9	1.10×10^9	250	39	2.74×10^6	177	$\sim$ 100%	1.29×10^7	190.5	8.65%	4.87×10^6	177	2.40%
					41	3.03×10^6	182	$\sim$ 100%	1.42×10^7	195.95	0.91%	5.38×10^6	182	0.24%
	jellium -1024	1.25×10^{12}	4.30×10^{10}	1024	45	1.66×10^7	1870	$\sim$ 100%	7.17×10^7	2010.4	3.69%	2.22×10^7	1870	0.88%
					47	1.81×10^7	2140	$\sim$ 100%	7.82×10^7	2300	0.39%	2.42×10^7	2140	0.09%
	Grover-100	6.8×10^9	5.4×10^{10}	100	41	1.35×10^6	220	$\sim$ 100%	6.81×10^6	236.5	6.16%	3.03×10^6	220	0.98%
					43	1.48×10^6	237	$\sim$ 100%	7.49×10^6	243.67	0.92%	3.33×10^6	237	0.11%
Future model	Hubbard -10-10	1.64×10^9	7.10×10^8	200	25	9.81×10^5	5.29	$\sim$ 100%	4.65×10^6	5.29	3.13%	1.36×10^6	5.29	3.13%
					27	1.14×10^6	5.50	$\sim$ 100%	5.43×10^6	5.50	0.38%	1.59×10^6	5.50	0.38%
	Hubbard -20-20	5.3×10^{10}	1.2×10^{10}	800	29	5.28×10^6	91.3	$\sim$ 100%	2.30×10^7	94.7	7.35%	6.26×10^6	91.3	1.88%
					31	6.03×10^6	94.3	$\sim$ 100%	2.63×10^7	97.8	0.79%	7.16×10^6	94.3	0.20%
	jellium -250	8.23×10^9	1.10×10^9	250	39	2.74×10^6	177	$\sim$ 100%	1.29×10^7	183.3	8.65%	3.73×10^6	177	2.40%
					41	3.03×10^6	182	$\sim$ 100%	1.42×10^7	188.83	0.91%	4.12×10^6	182	0.24%
	jellium -1024	1.25×10^{12}	4.30×10^{10}	1024	45	1.66×10^7	1870	$\sim$ 100%	7.17×10^7	1960	3.69%	1.93×10^7	1870	0.88%
					47	1.81×10^7	2140	$\sim$ 100%	7.82×10^7	2220	0.39%	2.10×10^7	2140	0.09%
	Grover-100	6.8×10^9	5.4×10^{10}	100	41	1.35×10^6	220	$\sim$ 100%	6.81×10^6	228.25	6.16%	2.10×10^6	220	0.98%
					43	1.48×10^6	237	$\sim$ 100%	7.49×10^6	245.89	0.92%	2.31×10^6	237	0.11%

threshold, indicating the maximum allowable LER to maintain the desired retry risk level. The blue, green, and orange lines illustrate LER dynamics under three scenarios: (1) no calibration, (2) qubit isolation + calibration, and (3) qubit isolation + code enlargement + calibration. The results demonstrate: (1) Without calibration, the LER increases exponentially due to error drift. (2) With qubit isolation and calibration but no code enlargement, the LER briefly spikes above the threshold due to distance loss from qubit isolation, though it eventually falls below the threshold after calibration. (3) The complete QECali scheme, incorporating both qubit isolation and code enlargement, quickly restores error protection and keeps the LER consistently below the threshold. Importantly, this compensation mechanism of QECali proves highly efficient: the code distance reduction (Δd) during calibration requires only a $d + \Delta d$ expansion, resulting in 14% additional physical qubits. This modest overhead can be further optimized by adjusting calibration intervals to minimize distance loss. Moreover, since compensation qubits are only needed during calibration, they can be shared across different logical qubits through our flexible layout scheme. This sharing reduces the net qubit overhead to 6%, while maintaining sub-threshold logical error rates throughout computation.

8.2.2. Impact of Drift-based Calibration Grouping We compare three calibration grouping strategies: (1) *Ideal grouping*, where each gate is calibrated only when its error reaches the threshold, (2) *Uniform calibration*, where all qubits are calibrated whenever any qubit requires calibration, and (3) *QECali's adaptive grouping*. Our evaluation shows that QECali reduces the total number of calibration operations by 3.63x to 11.1x compared to uniform calibration (Fig. 11), significantly lowering operational overhead without compromising error protection. This reduction stems from avoiding unnecessary calibrations of stable qubits. Instead, QECali leverages the natural variation in qubit stability, as evidenced by the normal distribution of drift rates (Fig. 11), and assigns calibration schedules based on individual drift patterns.

8.2.3. Impact of Intra-Group Calibration Scheduling

Calibration scheduling must balance speed against qubit overhead: more parallel calibrations reduce execution time but require more compensation qubits. Neither purely sequential nor fully parallel approaches are optimal for practical systems.

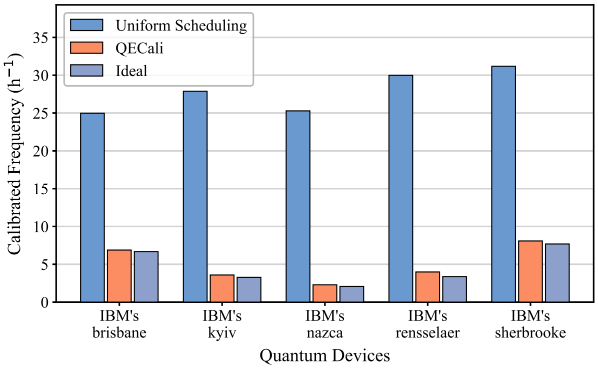


Figure 11: Reduction in calibration count through adaptive calibration assignment.

To quantify this trade-off, we evaluate scheduling strategies using a space-time overhead metric: Overhead $= \Delta d \times T(\text{Cali})$, where Δd represents the temporary reduction in code distance during calibration, and $T(\text{Cal})$ is the total calibration time. This metric captures both the spatial cost (additional physical qubits needed for code compensation, which scales as $O(\Delta d)$) and temporal cost (duration of reduced error protection) of the calibration process.

We compare three scheduling approaches: *sequential calibration*, which processes one gate at a time; *bulk calibration*, which calibrates as many gates as dependencies allow and achieves maximal parallelism; and QECali's adaptive scheduling, which optimizes the parallelism-overhead trade-off.

Our results (Fig. 12) show that QECali reduces space-time overhead by 2.89 times compared to sequential calibration and 3.8 times compared to bulk calibration. This improvement demonstrates that naive approaches to parallelization can be counterproductive—either consuming excessive qubit resources (bulk) or requiring unnecessarily long calibration times (sequential). QECali's adaptive scheduling finds an effective balance, minimizing both resource requirements and calibration duration.

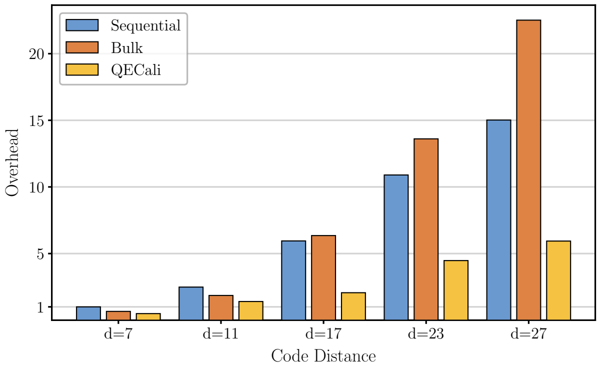


Figure 12: Space-time overhead of calibration of code with different code distance

8.3 QECali on real quantum device

Real quantum devices present additional challenges: non-uniform gate fidelities, complex error correlations, and hardware-specific constraints. To validate QECali's practicality, we implement $d = 3$ surface codes on two state-of-the-art quantum processors with distinct architectures: (1) Rigetti Ankaa-2 with a square lattice architecture, and (2) IBM-Rensselaer with heavy-hexagon architecture. We compare three scenarios: (1) *optimal* ("Original" column), (2) *drifted*, where a single gate's (either single-qubit or two-qubit gate) calibration parameters are replaced by those drifted after 8 hours ("drifted 1Q" and "drifted 2Q" columns), and (3) *drifted + qubit isolation* (two "isolated drifted" columns).

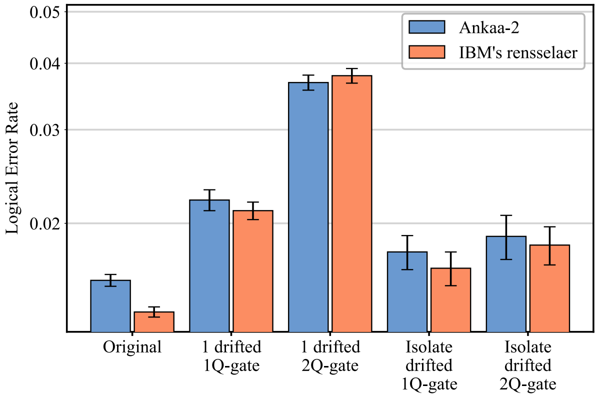


Figure 13: Logical error rate of a $d = 3$ surface code on Rigetti Ankaa-2 and IBM-Rensselaer

Standard surface code on Rigetti Ankaa-2: Our results (Fig. 13) reveal that even a single uncalibrated gate increases the LER by 41.6% for a single-qubit gate and 135.5% for a two-qubit gate. In contrast, QECali's qubit isolation and recalibration strategy incurs only a minor LER increase during calibration: 13.1% for single-qubit gates and 21.0% for two-qubit gates. This demonstrates that isolating high-error-rate qubits affected by error drift effectively suppresses the LER, requiring only a modest qubit overhead for code enlargement to restore QEC capability to the original LER level. Without isolating drifted qubits, significantly more qubits would be needed for enlargement, as the LER difference between drifted qubits ("drifted 1Q" and "drifted 2Q" columns) and the optimal ones ("Original" column) is much larger than that between the drift-removed qubits ("isolate drifted 1Q" and "isolated drifted 2Q" columns) and the optimal ones.

Heavy-hex surface code on IBM-Rensselaer: We observe similar results on the IBM-Rensselaer device. The drift of a single gate increases the LER by 55.0% for single-qubit gates and 178.2% for two-qubit gates. In contrast, removing the drifted qubits limits the LER increase to just 22.8% and 33.6%, significantly reducing the qubit resources required to restore the original QEC protection level. Notably, this device is more sensitive to drifted errors than the Rigetti Ankaa-2 device, as evidenced by the larger LER increases (55.0% and 178.2% compared to 41.6% and 135.5%). We speculate that this heightened sensitivity arises from the heavy-hex topology, where two-qubit gates are shared across multiple stabilizer measurements, amplifying the effect of individual gate errors.

These real-device experiments confirm that qubit isolation is an effective strategy for addressing drifted errors, requiring minimal qubit overhead for code enlargement. By combining qubit isolation with code enlargement, the LER can be kept sufficiently low to safeguard ongoing computation while enabling efficient calibration.

9 Conclusion

We present QECali to enable *in-situ* calibration of physical qubits while maintaining QEC in surface codes. Through selective qubit isolation and dynamic code enlargement, QECali achieves concurrent calibration and computation while preserving error correction capabilities. As quantum systems scale and computation times increase, QECali's approach to resource management provides a practical foundation for maintaining reliable quantum error correction during extended computations.

Acknowledgments

We thank the anonymous reviewers for their constructive feedback and AWS Cloud Credit for Research. This work is supported in part by NSF 2048144, NSF 2422169, NSF 2427109. This material is based upon work supported by the U.S. Department of Energy, Office of Science, National Quantum Information Science Research Centers, Quantum Science Center (QSC). This research used resources of the Oak Ridge Leadership Computing Facility (OLCF), which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231. The Pacific Northwest National Laboratory is operated by Battelle for the U.S. Department of Energy under Contract DE-AC05-76RL01830.

References

[1] 2023. Suppressing quantum errors by scaling a surface code logical qubit. *Nature* 614, 7949 (2023), 676–681.

[2] Rajeev Acharya, Laleh Aghababaie-Beni, Igor Aleiner, Trond I. Andersen, Markus Ansmann, Frank Arute, Kunal Arya, Abraham Asfaw, Nikita Astrakhantsev, Juan Atalaya, Ryan Babbush, Dave Bacon, Brian Ballard, Joseph C. Bardin, Johannes Bausch, Andreas Bengtsson, Alexander Bilmes, Sam Blackwell, Sergio Boixo, Gina Bortoli, Alexandre Bourassa, Jenna Bovaird, Leon Brill, Michael Broughton, David A. Browne, Brett Buchea, Bob B. Buckley, David A. Buell, Tim Burger, Brian Burkett, Nicholas Bushnell, Anthony Cabrera, Juan Campero, Hung-Shen Chang, Yu Chen, Zijun Chen, Ben Chiaro, Desmond Chik, Charina Chou, Jahan Claes, Agnetta Y. Cleland, Josh Cogan, Roberto Collins, Paul Conner, William Courtney, Alexander L. Crook, Ben Curtin, Sayan Das, Alex Davies, Laura De Lorenzo, Dripto M. Debroy, Sean Demura, Michel Devoret, Agustin Di Paolo, Paul Donohoe, Ilya Drozdov, Andrew Dunsworth, Clint Earle, Thomas Edlich, Alec Eickbusch, Aviv Moshe Elbag, Mahmoud Elzouka, Catherine Erickson, Lara Faoro, Edward Farhi, Vinicius S. Ferreira, Leslie Flores Burgos, Ebrahim Forati, Austin G. Fowler, Brooks Foxen, Suhas Ganjam, Gonzalo Garcia, Robert Gasca, Élie Genois, William Giang, Craig Gidney, Dar Gilboa, Raja Gosula, Alejandro Grajales Dau, Dietrich Graumann, Alex Greene, Jonathan A. Gross, Steve Habegger, John Hall, Michael C. Hamilton, Monica Hansen, Matthew P. Harrigan, Sean D. Harrington, Francisco J. H. Heras, Stephen Heslin, Paula Heu, Oscar Higgott, Gordon Hill, Jeremy Hilton, George Holland, Sabrina Hong, Hsin-Yuan Huang, Ashley Huff, William J. Huggins, Lev B. Ioffe, Sergei V. Isakov, Justin Iveland, Evan Jeffrey, Zhang Jiang, Cody Jones, Stephen Jordan, Chaitali Joshi, Pavol Juhas, Dvir Kafri, Hui Kang, Amir H. Karamlou, Kostyantyn Kechedzhi, Julian Kelly, Trupti Khaire, Tanuj Khattar, Mostafa Khezri, Seon Kim, Paul V. Klimov, Andrey R. Klots, Bryce Kobrin, Pushmeet Kohli, Alexander N. Korotkov, Fedor Kostritsa, Robin Kothari, Borislav Kozlovskii, John Mark Kreikebaum, Vladislav D. Kurilovich, Nathan Lacroix, David Landhuis, Tiano Lange-Dei, Brandon W. Langley, Pavel Laptev, Kim-Ming Lau, Loïck Le Guevel, Justin Ledford, Kenny Lee, Yuri D. Lensky, Shannon Leon, Brian J. Lester, Wing Yan Li, Yin Li, Alexander T. Lill, Wayne Liu, William P. Livingston, Aditya Locharla, Erik Lucero, Daniel Lundahl, Aaron Lunt, Sid Madhuk, Fionn D. Malone, Ashley Maloney, Salvatore Mandrá, Leigh S. Martin, Steven Martin, Orion Martin, Cameron Maxfield, Jarrod R. McClean, Matt McEwen, Seneca Meeks, Anthony Megrant, Xiao Mi, Kevin C. Miao, Amanda Mieszala, Reza Molavi, Sebastian Molina, Shirin Montazeri, Alexis Morvan, Ramis Movassagh, Wojciech Mruczkiewicz, Ofer Naaman, Matthew Neeley, Charles Neill, Ani Nersisyan, Hartmut Neven, Michael Newman, Jiun How Ng, Anthony Nguyen, Murray Nguyen, Chia-Hung Ni, Thomas E. O'Brien, William D. Oliver, Alex Opremcak, Kristoffer Ottosson, Andre Petukhov, Alex Pizzuto, John Platt, Rebecca Potter, Orion Pritchard, Leonid P. Pryadko, Chris Quintana, Ganesh Ramachandran, Matthew J. Reagor, David M. Rhodes, Gabrielle Roberts, Eliott Rosenberg, Emma Rosenfeld, Pedram Roushan, Nicholas C. Rubin, Negar Saei, Daniel Sank, Kannan Sankaragomathi, Kevin J. Satzinger, Henry F. Schurkus, Christopher Schuster, Andrew W. Senior, Michael J. Shearn, Aaron Shorter, Noah Shutty, Vladimir Shvarts, Shraddha Singh, Volodymyr Sivak, Jindra Skruzny, Spencer Small, Vadim Smelyanskiy, W. Clarke Smith, Rolando D. Somma, Sofia Springer, George Sterling, Doug Strain, Jordan Suchard, Aaron Szasz, Alex Sztein, Douglas Thor, Alfredo Torres, M. Mert Torunbalci, Abeer Vaishnav, Justin Vargas, Sergey Vdovichev, Guifre Vidal, Benjamin Villalonga, Catherine Vollgraff Heidweiller, Steven Waltman, Shannon X. Wang, Brayden Ware, Kate Weber, Theodore White, Kristi Wong, Bryan W. K. Woo, Cheng Xing, Z. Jamie Yao, Ping Yeh, Bicheng Ying, Juhwan Yoo, Noureldin Yosri, Grayson Young, Adam Zalcman, Yaxing Zhang, Ningfeng Zhu, and Nicholas Zobrist. 2024. Quantum error correction below the surface code threshold. arXiv:2408.13687 [quant-ph] https://arxiv.org/abs/2408.13687

[3] Daniel P Arovas, Erez Berg, Steven A Kivelson, and Srinivas Raghu. 2022. The hubbard model. *Annual review of condensed matter physics* 13, 1 (2022), 239–274.

[4] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. 2019. Quantum supremacy using a programmable superconducting processor. *Nature* 574, 7779 (2019), 505–510.

[5] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. 2019. Quantum supremacy using a programmable superconducting processor. *Nature* 574, 7779 (Oct. 2019), 505–510. doi:10.1038/s41586-019-1666-5

[6] James M Auger, Hussain Anwar, Mercedes Gimeno-Segovia, Thomas M Stace, and Dan E Browne. 2017. Fault-tolerance thresholds for the surface code with fabrication errors. *Physical Review A* 96, 4 (2017), 042316.

[7] Ryan Babbush, Craig Gidney, Dominic W Berry, Nathan Wiebe, Jarrod McClean, Alexandru Paler, Austin Fowler, and Hartmut Neven. 2018. Encoding electronic spectra in quantum circuits with linear T complexity. *Physical Review X* 8, 4 (2018), 041015.

[8] Michael Beverland, Vadym Kliuchnikov, and Eddie Schoute. 2022. Surface code compilation via edge-disjoint paths. *PRX Quantum* 3, 2 (2022), 020342.

[9] Dolev Bluvstein, Simon J. Evered, Alexandra A. Geim, Sophie H. Li, Hengyun Zhou, Tom Manovitz, Sepehr Ebadi, Madelyn Cain, Marcin Kalinowski, Dominik Hangleiter, J. Pablo Bonilla Ataides, Nishad Maskara, Iris Cong, Xun Gao, Pedro Sales Rodriguez, Thomas Karolyshyn, Giulia Semeghini, Michael J. Gullans, Markus Greiner, Vladan Vuletić, and Mikhail D. Lukin. 2024. Logical quantum processor based on reconfigurable atom arrays. *Nature* 626, 7997 (01 Feb 2024), 58–65. doi:10.1038/s41586-023-06927-3

[10] Héctor Bombín and Miguel Angel Martin-Delgado. 2009. Quantum measurements and gates by code deformation. *Journal of Physics A: Mathematical and Theoretical* 42, 9 (2009), 095302.

[11] Sergey Bravyi, Oliver Dial, Jay M Gambetta, Darío Gil, and Zaira Nazario. 2022. The future of quantum computing with superconducting qubits. *Journal of Applied Physics* 132, 16 (2022).

[12] Sergey B Bravyi and A Yu Kitaev. 1998. Quantum codes on a lattice with boundary. *arXiv preprint quant-ph/9811052* (1998).

[13] Yudong Cao, Jonathan Romero, Jonathan P. Olson, Matthias Degroote, Peter D. Johnson, Mária Kieferová, Ian D. Kivlichan, Tim Menke, Borja Peropadre, Nicolas P. D. Sawaya, Sukin Sim, Libor Veis, and Alán Aspuru-Guzik. 2019. Quantum Chemistry in the Age of Quantum Computing. *Chemical Reviews* 119, 19 (Oct. 2019), 10856–10915. doi:10.1021/acs.chemrev.8b00803

[14] Christopher Chamberland and Earl T Campbell. 2022. Universal quantum computing with twist-free and temporally encoded lattice surgery. *PRX Quantum* 3, 1 (2022), 010331.

[15] Nicolas Delfosse and Naomi H Nickerson. 2021. Almost-linear time decoding algorithm for topological codes. *Quantum* 5 (2021), 595.

[16] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. 2002. Topological quantum memory. *J. Math. Phys.* 43, 9 (2002), 4452–4505.

[17] Brayden Thomas Edman. 2024. A Hardware-Focused Tour of IBM's 127-Qubit Eagle Processor. *Vanderbilt Undergraduate Research Journal* 14, 1 (2024).

[18] Austin G Fowler. 2013. Minimum weight perfect matching of fault-tolerant topological quantum error correction in average $O(1)$ parallel time. *arXiv preprint arXiv:1307.1740* (2013).

[19] Austin G Fowler and Craig Gidney. 2018. Low overhead quantum computation using lattice surgery. *arXiv preprint arXiv:1808.06709* (2018).

[20] Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. 2012. Surface codes: Towards practical large-scale quantum computation. *Physical Review A* 86, 3 (2012), 032324.

[21] Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. 2012. Surface codes: Towards practical large-scale quantum computation. *Physical Review A* 86, 3 (2012), 032324.

[22] Craig Gidney. 2021. Stim: a fast stabilizer circuit simulator. *Quantum* 5 (2021), 497.

[23] Craig Gidney and Martin Ekerå. 2021. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum* 5 (April 2021), 433. doi:10.22331/q-2021-04-15-433

[24] Daniel Gottesman. 1996. Class of quantum error-correcting codes saturating the quantum Hamming bound. *Physical Review A* 54, 3 (1996), 1862.

[25] Daniel Gottesman. 1998. The Heisenberg Representation of Quantum Computers. arXiv:quant-ph/9807006 [quant-ph]

[26] Daniel Gottesman. 1998. Theory of fault-tolerant quantum computation. *Physical Review A* 57, 1 (1998), 127.

[27] E. Gümüş, D. Majidi, D. Nikolić, P. Raif, B. Karimi, J. T. Peltonen, E. Scheer, J. P. Pekola, H. Courtois, W. Belzig, and C. B. Winkelmann. 2023. Calorimetry of a phase slip in a Josephson junction. *Nature Physics* 19, 2 (2023), 196–200.

[28] Simon Gustavsson, Fei Yan, Gianluigi Catelani, Jonas Bylander, Archana Kamal, Jeffrey Birenbaum, David Hover, Danna Rosenberg, Gabriel Samach, Adam P. Sears, Steven J. Weber, Jonilyn L. Yoder, John Clarke, Andrew J. Kerman, Fumiki Yoshihara, Yasunobu Nakamura, Terry P. Orlando, and William D. Oliver. 2016. Suppressing relaxation in superconducting qubits by quasiparticle pumping. *Science* 354, 6319 (2016), 1573–1577.

[29] Daniel Herr, Franco Nori, and Simon J Devitt. 2017. Lattice surgery translation for quantum computation. *New Journal of physics* 19, 1 (2017), 013034.

[30] Oscar Higgott. 2022. PyMatching: A Python package for decoding quantum codes with minimum-weight perfect matching. *ACM Transactions on Quantum Computing* 3, 3 (2022), 1–16.

[31] Clare Horsman, Austin G Fowler, Simon Devitt, and Rodney Van Meter. 2012. Surface code quantum computing by lattice surgery. *New Journal of Physics* 14, 12 (2012), 123011.

[32] He-Liang Huang, Dachao Wu, Daojin Fan, and Xiaobo Zhu. 2020. Superconducting quantum computing: a review. *Science China Information Sciences* 63 (2020), 1–32.

[33] Petar Jurcevic, Ali Javadi-Abhari, Lev S Bishop, Isaac Lauer, Daniela F Bogorin, Markus Brink, Lauren Capelluto, Oktay Günlük, Toshinari Itoko, Naoki Kanazawa, Abhinav Kandala, George A Keefe, Kevin Krsulich, William Landers, Eric P Lewandowski, Douglas T McClure, Giacomo Nannicini, Adinath Narasgond, Hasan M Nayfeh, Emily Pritchett, Mary Beth Rothwell, Srikanth Srinivasan, Neereja Sundaresan, Cindy Wang, Ken X Wei, Christopher J Wood, Jeng-Bang Yau, Eric J Zhang, Oliver E Dial, Jerry M Chow, and Jay M Gambetta. 2021. Demonstration of quantum volume 64 on a superconducting quantum computing system. *Quantum Science and Technology* 6, 2 (2021), 025020.

[34] J. Kelly, R. Barends, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Yu Chen, Z. Chen, B. Chiaro, A. Dunsworth, E. Lucero, M. Neeley, C. Neill, P. J. J. O'Malley, C. Quintana, P. Roushan, A. Vainsencher, J. Wenner, and John M. Martinis. 2016. Scalable in situ qubit calibration during repetitive error detection. *Physical Review A* 94, 3 (2016), 032321.

[35] Julian Kelly, Peter O'Malley, Matthew Neeley, Hartmut Neven, and John M Martinis. 2018. Physical qubit calibration on a directed acyclic graph. *arXiv preprint arXiv:1803.03226* (2018).

[36] Younghun Kim, Jeongsoo Kang, and Younghun Kwon. 2023. Design of quantum error correcting code for biased error on heavy-hexagon structure. *Quantum Information Processing* 22, 6 (2023), 230.

[37] Paul V. Klimov, Andreas Bengtsson, Chris Quintana, Alexandre Bourassa, Sabrina Hong, Andrew Dunsworth, Kevin J. Satzinger, William P. Livingston, Volodymyr Sivak, Murphy Yuezhen Niu, Trond I. Andersen, Yaxing Zhang, Desmond Chik, Zijun Chen, Charles Neill, Catherine Erickson, Alejandro Grajales Dau, Anthony Megrant, Pedram Roushan, Alexander N. Korotkov, Julian Kelly, Vadim Smelyanskiy, Yu Chen, and Hartmut Neven. 2024. Optimizing quantum gates towards the scale of logical qubits. *Nature Communications* 15, 1 (2024), 2442.

[38] P. V. Klimov, J. Kelly, Z. Chen, M. Neeley, A. Megrant, B. Burkett, R. Barends, K. Arya, B. Chiaro, Yu Chen, A. Dunsworth, A. Fowler, B. Foxen, C. Gidney, M. Giustina, R. Graff, T. Huang, E. Jeffrey, Erik Lucero, J. Y. Mutus, O. Naaman, C. Neill, C. Quintana, P. Roushan, Daniel Sank, A. Vainsencher, J. Wenner, T. C. White, S. Boixo, R. Babbush, V. N. Smelyanskiy, H. Neven, and John M. Martinis. 2018. Fluctuations of energy-relaxation times in superconducting qubits. *Physical review letters* 121, 9 (2018), 090502.

[39] Paul V Klimov, Julian Kelly, John M Martinis, and Hartmut Neven. 2020. The snake optimizer for learning quantum processor control parameters. *arXiv preprint arXiv:2006.04594* (2020).

[40] Joonho Lee, Dominic W. Berry, Craig Gidney, William J. Huggins, Jarrod R. McClean, Nathan Wiebe, and Ryan Babbush. 2021. Even More Efficient Quantum Computations of Chemistry Through Tensor Hypercontraction. *PRX Quantum* 2 (Jul 2021), 030305. Issue 3. doi:10.1103/PRXQuantum.2.030305

[41] Tian-Ming Li, Jia-Chi Zhang, Bing-Jie Chen, Kaixuan Huang, Hao-Tian Liu, Yong-Xi Xiao, Cheng-Lin Deng, Gui-Han Liang, Chi-Tong Chen, Yu Liu, Hao Li, Zhen-Ting Bao, Kui Zhao, Yueshan Xu, Li Li, Yang He, Zheng-He Liu, Yi-Han Yu, Si-Yun Zhou, Yan-Jun Liu, Xiaohui Song, Dongning Zheng, Zhong-Cheng Xiang, Yun-Hao Shi, Kai Xu, and Heng Fan. 2024. High-precision pulse calibration of tunable couplers for high-fidelity two-qubit gates in superconducting quantum processors. *arXiv preprint arXiv:2410.15041* (2024).

[42] Daniel A Lidar and Todd A Brun. 2013. *Quantum error correction.* Cambridge university press.

[43] Sophia Fuhui Lin, Eric C Peterson, Krishanu Sankar, and Prasahnt Sivarajah. 2024. Spatially parallel decoding for multi-qubit lattice surgery. *arXiv preprint arXiv:2403.01353* (2024).

[44] Daniel Litinski. 2019. A game of surface codes: Large-scale quantum computing with lattice surgery. *Quantum* 3 (2019), 128.

[45] Daniel Litinski. 2023. How to compute a 256-bit elliptic curve private key with only 50 million Toffoli gates. *arXiv preprint arXiv:2306.08585* (2023).

[46] Yiding Liu, Zedong Li, Alan Robertson, Xin Fu, and Shuaiwen Leon Song. 2023. Enabling efficient real-time calibration on cloud quantum machines. *IEEE Transactions on Quantum Engineering* 4 (2023), 1–17.

[47] Easwar Magesan, Jay M. Gambetta, B. R. Johnson, Colm A. Ryan, Jerry M. Chow, Seth T. Merkel, Marcus P. Da Silva, George A. Keefe, Mary B. Rothwell, Thomas A. Ohki, Mark B. Ketchen, and M. Steffen. 2012. Efficient measurement of quantum gate error by interleaved randomized benchmarking. *Physical review letters* 109, 8 (2012), 080505.

[48] John M Martinis. 2021. Saving superconducting quantum processors from decay and correlated errors generated by gamma and cosmic rays. *npj Quantum Information* 7, 1 (2021), 90.

[49] John M. Martinis, K. B. Cooper, R. McDermott, Matthias Steffen, Markus Ansmann, K. D. Osborn, K. Cicak, Seongshik Oh, D. P. Pappas, R. W. Simmonds, and Clare C. Yu. 2005. Decoherence in Josephson qubits from dielectric loss. *Physical review letters* 95, 21 (2005), 210503.

[50] Matt McEwen, Lara Faoro, Kunal Arya, Andrew Dunsworth, Trent Huang, Seon Kim, Brian Burkett, Austin Fowler, Frank Arute, Joseph C. Bardin, Andreas Bengtsson, Alexander Bilmes, Bob B. Buckley, Nicholas Bushnell, Zijun Chen, Roberto Collins, Sean Demura, Alan R. Derk, Catherine Erickson, Marissa Giustina, Sean D. Harrington, Sabrina Hong, Evan Jeffrey, Julian Kelly, Paul V. Klimov, Fedor Kostritsa, Pavel Laptev, Aditya Locharla, Xiao Mi, Kevin C. Miao, Shirin Montazeri, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Alex Opremcak, Chris Quintana, Nicholas Redd, Pedram Roushan, Daniel Sank, Kevin J. Satzinger, Vladimir Shvarts, Theodore White, Z. Jamie Yao, Ping Yeh, Juhwan Yoo, Yu Chen, Vadim Smelyanskiy, John M. Martinis, Hartmut Neven, Anthony Megrant, Lev Ioffe, and Rami Barends. 2022. Resolving catastrophic error bursts from cosmic rays in large arrays of superconducting qubits. *Nature Physics* 18, 1 (2022), 107–111.

[51] Clemens Müller, Jared H Cole, and Jürgen Lisenfeld. 2019. Towards understanding two-level-systems in amorphous solids: insights from quantum circuits. *Reports on Progress in Physics* 82, 12 (2019), 124501.

[52] Prakash Murali, David C McKay, Margaret Martonosi, and Ali Javadi-Abhari. 2020. Software mitigation of crosstalk on noisy intermediate-scale quantum computers. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems.* 1001–1016.

[53] Shota Nagayama, Austin G Fowler, Dominic Horsman, Simon J Devitt, and Rodney Van Meter. 2017. Surface code error correction on a defective lattice. *New Journal of Physics* 19, 2 (2017), 023050.

[54] C. Neill, P. Roushan, K. Kechedzhi, S. Boixo, S. V. Isakov, V. Smelyanskiy, A. Megrant, B. Chiaro, A. Dunsworth, K. Arya, R. Barends, B. Burkett, Y. Chen, Z. Chen, A. Fowler, B. Foxen, M. Giustina, R. Graff, E. Jeffrey, T. Huang, J. Kelly, P. Klimov, E. Lucero, J. Mutus, M. Neeley, C. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. C. White, H. Neven, and J. M. Martinis. 2018. A blueprint for demonstrating quantum supremacy with superconducting qubits. *Science* 360, 6385 (2018), 195–199.

[55] Michael A Nielsen and Isaac L Chuang. 2010. *Quantum computation and quantum information.* Cambridge university press.

[56] Timothy Proctor, Melissa Revelle, Erik Nielsen, Kenneth Rudinger, Daniel Lobser, Peter Maunz, Robin Blume-Kohout, and Kevin Young. 2020. Detecting and tracking drift in quantum information processors. *Nature communications* 11, 1 (2020), 5396.

[57] Jiaan Qi and Hui Khoon Ng. 2021. Randomized benchmarking in the presence of time-correlated dephasing noise. *Physical Review A* 103, 2 (2021), 022607.

[58] Gokul Subramanian Ravi, Kaitlin Smith, Jonathan M Baker, Tejas Kannan, Nathan Earnest, Ali Javadi-Abhari, Henry Hoffmann, and Frederic T Chong. 2023. Navigating the dynamic noise landscape of variational quantum algorithms with QISMET. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2.* 515–529.

[59] Peter W Shor. 1996. Fault-tolerant quantum computation. In *Proceedings of 37th conference on foundations of computer science.* IEEE, 56–65.

[60] Peter W Shor. 1999. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review* 41, 2 (1999), 303–332.

[61] M. Springborg and Y. Dong. 2006. Chapter 4 The Jellium Model. In *Metallic Chains/Chains of Metals*, Michael Springborg and Yi Dong (Eds.). Handbook of Metal Physics, Vol. 1. Elsevier, 37–44. doi:10.1016/S1570-002X(06)01004-4

[62] Thomas M Stace and Sean D Barrett. 2010. Error correction and degeneracy in surface codes suffering loss. *Physical Review A* 81, 2 (2010), 022317.

[63] Thomas M Stace, Sean D Barrett, and Andrew C Doherty. 2009. Thresholds for topological codes in the presence of loss. *Physical review letters* 102, 20 (2009), 200501.

[64] Armands Strikis, Simon C Benjamin, and Benjamin J Brown. 2023. Quantum computing is scalable on a planar array of qubits with fabrication defects. *Physical Review Applied* 19, 6 (2023), 064081.

[65] Swamit S Tannu and Moinuddin K Qureshi. 2019. Not all qubits are created equal: A case for variability-aware policies for NISQ-era quantum computers. In *Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems.* 987–999.

[66] Caroline Tornow, Naoki Kanazawa, William E Shanks, and Daniel J Egger. 2022. Minimum quantum run-time characterization and calibration via restless measurements with dynamic repetition rates. *Physical Review Applied* 17, 6 (2022), 064061.

[67] Christophe Vuillot, Lingling Lao, Ben Criger, Carmen García Almudéver, Koen Bertels, and Barbara M Terhal. 2019. Code deformation and lattice surgery are gauge fixing. *New Journal of Physics* 21, 3 (2019), 033028.

[68] Zuolin Wei, Tan He, Yangsen Ye, Dachao Wu, Yiming Zhang, Youwei Zhao, Weiping Lin, He-Liang Huang, Xiaobo Zhu, and Jian-Wei Pan. 2024. Low-Overhead Defect-Adaptive Surface Code with Bandage-Like Super-Stabilizers. *arXiv preprint arXiv:2404.18644* (2024).

[69] Nicolas Wittler, Federico Roy, Kevin Pack, Max Werninghaus, Anurag Saha Roy, Daniel J Egger, Stefan Filipp, Frank K Wilhelm, and Shai Machnes. 2021. Integrated tool set for control, calibration, and characterization of quantum devices applied to superconducting qubits. *Physical Review Applied* 15, 3 (2021), 034080.

[70] Keyi Yin, Xiang Fang, Travis S Humble, Ang Li, Yunong Shi, and Yufei Ding. 2024. Surf-Deformer: Mitigating dynamic defects on surface code via adaptive deformation. (2024).

[71] Shuaining Zhang, Yao Lu, Kuan Zhang, Wentao Chen, Ying Li, Jing-Ning Zhang, and Kihwan Kim. 2020. Error-mitigated quantum gates exceeding physical fidelities in a trapped-ion system. *Nature communications* 11, 1 (2020), 587.

[72] Youwei Zhao, Yangsen Ye, He-Liang Huang, Yiming Zhang, Dachao Wu, Huijie Guan, Qingling Zhu, Zuolin Wei, Tan He, Sirui Cao, Fusheng Chen, Tung-Hsun Chung, Hui Deng, Daojin Fan, Ming Gong, Cheng Guo, Shaojun Guo, Lianchen Han, Na Li, Shaowei Li, Yuan Li, Futian Liang, Jin Lin, Haoran Qian, Hao Rong, Hong Su, Lihua Sun, Shiyu Wang, Yulin Wu, Yu Xu, Chong Ying, Jiale Yu, Chen Zha, Kaili Zhang, Yong-Heng Huo, Chao-Yang Lu, Cheng-Zhi Peng, Xiaobo Zhu, and Jian-Wei Pan. 2022. Realization of an error-correcting surface code with superconducting qubits. *Physical Review Letters* 129, 3 (2022), 030501.

A Artifact Appendix

A.1 Abstract

The artifact contains the source code used to generate, simulate, and evaluate the in-situ calibration methods presented in this paper. Since certain results presented in this work utilized premium quantum hardware that require access tokens, this artifact provides a simulation method for certain tasks. Users who have access to different quantum hardware platforms can take the circuits generated within the artifact and manually execute them. The artifact provides Jupyter notebooks and python files to reproduce major results in Figure 10, Figure 12, and Table 2.

A.2 Artifact check-list (meta-information)

- **Program: Stim**
- **Run-time environment:** Jupyter Kernel
- **Hardware:** AMD EPYC 9534 64-Core
- **Execution:** Stim circuit simulation
- **Output:** Error rate, space-time overhead, qubit usage, and related metrics
- **Experiments:** Surface code and in-situ calibration simulation
- **How much disk space required (approximately)?:** 1 GB to store the artifact directory and python virtual environment.
- **How much time is needed to prepare workflow (approximately)?:** 10 minutes
- **How much time is needed to complete experiments (approximately)?:** 1 hour
- **Publicly available?:** Yes
- **Code licenses (if publicly available)?:** MIT License
- **Workflow automation framework used?:** Jupyter notebook
- **Archived (provide DOI)?:** https://doi.org/10.5281/zenodo.15104546

A.3 Description

A.3.1 How to access. The artifact is available on Zenodo https://doi.org/10.5281/zenodo.15104546.

A.3.2 Hardware dependencies. The artifact relies on quantum circuit simulation available through the Stim. Any system which can run python programs should be able to evaluate the artifact.

A.3.3 Software dependencies. The dependencies are listed within requirements.txt.

A.4 Installation

The README.md contains detailed instructions to prepare the python environment. After downloading the artifact zipfile, and extracting the contents, the environment can be installed via:

# cd QECali_ISCA_Artifact

# pip install -r requirements.txt The user can then open the jupyter lab with the command:

# jupyter lab

The jupyter notebook files contain major results used in this paper.

A.5 Evaluation and expected results

The notebook space_time_overhead.ipynb contains examples how the space time are computed using the results of the code deformation and calibration time. The notebook stim_error.ipynb simulates the logical error rate analysis with error drifts, utilizing real-machine data including initial error rate and error growing rate. The python file evaluation.py calculates the error rate of application-oriented benchmarks listed in Table 2. The calculation utilizes a custom simulator based on the path finding process of lattice suggery. A folder ben_gen_example is provided to generate benchmarks in Table 2. Also, pre-generated benchmarks are stored in bench folder. The python file heavy_hex.py contains the simulation and deformation of surface code on heavy-hex topology.

A.6 Methodology

Submission, reviewing and badging methodology:

- https://www.acm.org/publications/policies/artifact-review-and-badging-current
- https://cTuning.org/ae