ERASER: Towards Adaptive Leakage Suppression for Fault-Tolerant Quantum Computing

Suhas Vittal
suhaskvittal@gatech.edu
Georgia Institute of Technology
Atlanta, GA, USA

Poulami Das
poulami.das@utexas.edu
University of Texas, Austin
Austin, Texas, USA

Moinuddin Qureshi
moin@gatech.edu
Georgia Institute of Technology
Atlanta, GA, USA

ABSTRACT

Quantum error correction (QEC) codes can tolerate hardware errors by encoding fault-tolerant logical qubits using redundant physical qubits and detecting errors using parity checks. *Leakage errors* occur in quantum systems when a qubit leaves its computational basis and enters higher energy states. These errors severely limit the performance of QEC due to two reasons. *First*, they lead to erroneous parity checks that obfuscate the accurate detection of errors. *Second*, the leakage spreads to other qubits and creates a pathway for more errors over time. Prior works tolerate leakage errors by using leakage reduction circuits (LRCs) that modify the parity check circuitry of QEC codes. Unfortunately, naively using LRCs *always* throughout a program is sub-optimal because LRCs incur additional two-qubit operations that (1) facilitate leakage transport, and (2) serve as new sources of errors.

Ideally, LRCs should *only* be used if leakage occurs, so that errors from both leakage as well as additional LRC operations are simultaneously minimized. However, identifying leakage errors in real-time is challenging. To enable the robust and efficient usage of LRCs, we propose *ERASER* that speculates the subset of qubits that *may* have leaked and only uses LRCs for those qubits. Our studies show that the majority of leakage errors typically impact the parity checks. We leverage this insight to identify the leaked qubits by analyzing the patterns in the failed parity checks. We propose ERASER+M that enhances ERASER by detecting leakage more accurately using qubit measurement protocols that can classify qubits into $|0\rangle$, $|1\rangle$ and $|L\rangle$ states. ERASER and ERASER+M improve the logical error rate by up to 4.3× and 23× respectively compared to always using LRC.

KEYWORDS

Quantum Error Correction, Leakage Suppression

ACM Reference Format:
Suhas Vittal, Poulami Das, and Moinuddin Qureshi. 2023. ERASER: Towards Adaptive Leakage Suppression for Fault-Tolerant Quantum Computing. In *56th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '23), October 28-November 1, 2023, Toronto, ON, Canada.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3613424.3614251

MICRO '23, October 28-November 1, 2023, Toronto, ON, Canada

ACM ISBN 979-8-4007-0329-4/23/10.
https://doi.org/10.1145/3613424.3614251

1 INTRODUCTION

Noisy quantum hardware and imperfect operations limit us from running most promising quantum applications [10, 13, 23, 28, 32, 40, 44, 45, 55, 57, 58]. Quantum Error Correction (QEC) can bridge the gap between quantum applications and qubit devices. *Fault-Tolerant Quantum Computers (FTQCs)* use QEC codes to encode *logical qubits* using several *physical qubits* such that the error rate of the logical qubits is lower than the physical error rate if the latter is below a certain *threshold*. Moreover, the logical error rate decreases exponentially with increasing redundancy, measured as a QEC code's *distance (d)*. This exponential suppression of errors enables QEC codes to achieve the target error rate necessary to run a given quantum application.

In this paper, we focus on *surface codes*, widely recognized as the most promising QEC code, which uses *data qubits* to store quantum information and *parity qubits* to detect errors [21, 25, 38, 39]. An FTQC executes *syndrome extraction circuits* that project errors on the data qubits onto the parity qubits and measure the parity qubits to obtain a bitstring of parity checks called a *syndrome*. A classical *decoder* uses the syndrome to identify errors. It sends the correction to the *control processor*, which corrects the errors. In practice, syndrome extraction uses quantum operations, which are also error-prone. To tolerate these errors, a decoder analyzes at least d consecutive rounds of syndromes, also known as a *QEC cycle*. FTQCs enable computations by interleaving QEC cycles in between logical operations.

Recent studies from IBM and Google show that *leakage errors* degrade QEC performance on real hardware [1, 41, 64]. Leakage errors occur when qubits leave the computational basis ($|0\rangle$ and $|1\rangle$) and enter a higher energy state $|L\rangle$ [1, 5–7, 24, 48, 52, 53, 63, 64]. As quantum operations are only calibrated for the computational basis, leakage errors deteriorate logical performance for two reasons. *First*, leaked qubits cause faulty operations during syndrome extraction, inducing random errors on their neighboring qubits and obfuscating other errors from being detected due to incorrect parity checks. *Second*, these faulty operations spread leakage onto other qubits via *leakage transport*. If not removed, leakage continues spreading, affecting more qubits over time and increasing the *leakage population ratio*, or the number of qubits leaked at any given time, as shown in Figure 1(a), making QEC codes increasingly vulnerable. For example, our studies show that leakage errors increase the logical error rate by 27× and 467× for a distance 7 surface code after one and five QEC cycles, respectively. *Thus, reducing the impact of leakage errors is crucial to improving the performance of QEC.*

Leakage errors arise from fundamental device-level imperfections and cannot be wholly eliminated despite improving qubit qualities. Instead, recent approaches actively remove them as they

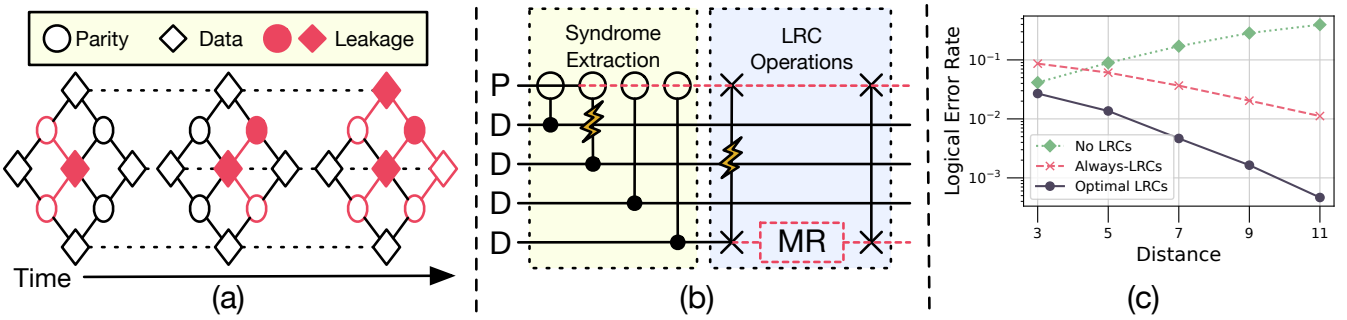

Figure 1: (a) Leakage errors spread over time (b) Regular syndrome extraction resets parity qubits every round, removing any leakage from them. Leakage reduction circuits (LRCs) swap data and parity qubits to remove leakage from the data qubit at the expense of five extra CNOTs (two extra SWAPs cost five CNOTs). (c) Logical error rate without LRCs, state-of-the-art Always-LRCs, and idealized LRC scheduling over 10 QEC cycles.

occur by resetting the leaked physical qubits. The most common technique uses leakage reduction circuits (LRCs) that modify the syndrome extraction circuit to *swap* the data and parity qubits [3, 6, 24, 63], as in Figure 1(b). Syndrome extraction rounds without LRCs proceed normally, where the parity qubits are measured and reset, eliminating any leakage from them. These rounds are followed by rounds with LRCs where the SWAPs remove leakage from the data qubits. Prior works schedule LRCs every alternate round throughout the duration of a program. *However, our studies show that always scheduling LRCs is sub-optimal and limits their efficacy.*

Always-LRCs scheduling throughout program execution done in prior work has the following drawbacks. *First*, leaked qubits facilitate leakage transport through the two-qubit operations intended to eliminate them. Our studies show that the *leakage population ratio (LPR)* continues to increase over QEC cycles despite using LRCs; ideally, we want the LPR to remain as low as possible to prevent performance degradation. *Second*, LRC operations increase the number of two-qubit operations in a syndrome extraction round from 4 to 9, as shown in Figure 1(b). Two-qubit operations are themselves error-prone and serve as new sources of errors even when there are no leakage errors. Our studies show that although the state-of-the-art *Always-LRCs* scheduling policy can improve the logical error rate, its performance is still far from an idealized policy that schedules LRCs only when leakage occurs. For example, Always-LRCs scheduling can improve the logical error rate by 4× for distance 7 surface codes, as shown in Figure 1(c). However, the idealized policy can improve the logical error rate by up to 10×. Furthermore, this gap consistently increases with the increasing code distance, heavily limiting the performance of QEC. This paper aims to bridge this gap via the optimal usage of LRCs such that leakage errors are maximally removed while limiting leakage spread and minimizing errors from LRC operations. We propose *ERASER* that achieves this goal.

ERASER comprises of three key components. (1) The *Leakage Speculation Block (LSB)* analyzes the current syndrome to speculate potentially leaked qubits, (2) the *Dynamic LRC Insertion (DLI)* block modifies the next syndrome extraction round to include LRC operations for this subset of qubits, and (3) the *QEC Schedule Generator (QSG)* issues the updated syndrome extraction schedules to the qubits. Designing efficient LSB logic is non-trivial because leakage errors may remain invisible during syndrome extraction while continuing to induce errors on other qubits. Even if they impact syndrome extraction, they create random parity qubit flip patterns, as a leaked data qubit can cause any arbitrary combination of its four neighboring parity qubits to flip. Efficient DLI logic design is also challenging as the QEC schedules must be adapted in real time. Failure to introduce LRCs in real-time causes leakage to persist, whereas waiting to determine which LRCs to use causes QEC cycles to slow down and errors to accumulate on qubits.

To overcome these challenges, we leverage the insight that most leakage errors become visible to syndrome extraction within a few syndrome extraction rounds, and thus, optimizing the LSB to tackle visible leakage is sufficient. To address the challenge related to arbitrary syndrome bit-flip patterns caused by leakage errors, we speculate a leakage has occurred if at least half of the neighboring parity qubits have flipped for a data qubit. This achieves a sweet spot between two extremes: a conservative prediction based on too few neighboring syndrome bit-flips introduces more LRC operations, whereas a more aggressive prediction may cause leakage to remain undetected. Note that during Always-LRCs scheduling, each data qubit swaps with a unique parity qubit. However, as ERASER schedules LRCs dynamically, two data qubits may request to swap with the same parity qubit, thus preventing their LRCs from being scheduled concurrently. To resolve this problem, DLI schedules the LRCs in the upcoming round to maximize the number of LRCs scheduled. By scheduling LRCs only for likely-leaked data qubits, ERASER removes leakage errors while minimizing any additional errors caused by LRCs.

Finally, recent device-level research has been increasingly exploring the efficacy of *multi-level* readout, which classifies additional states beyond $|0\rangle$ and $|1\rangle$ [12, 49]. While the accuracy of multi-level readout is worse than standard readout, the additional information granted by multi-level readout can enhance leakage detection accuracy. We propose ERASER+M that leverages multi-level readout to improve LRC scheduling further.

Our evaluations show that ERASER and ERASER+M improve the logical error rate by up to 4.3× and 23×, respectively, compared to Always-LRCs scheduling. Furthermore, ERASER requires <1%

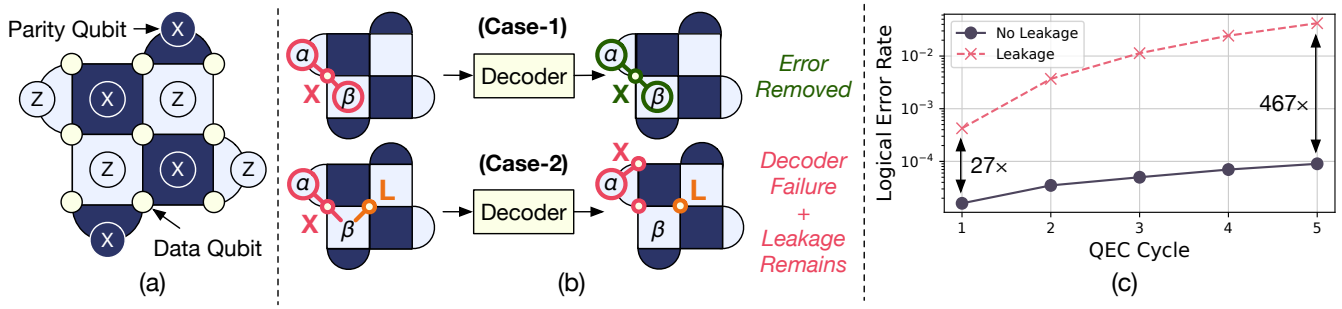

Figure 2: (a) Distance (d) 3 surface code. (b) In Case 1 (no leakage), there is an X error on a data qubit that causes two Z stabilizers (α and β) to flip. In Case 2 (with leakage), the same data qubit has an X error, but the central data qubit has a leakage error (L), which causes stabilizer β not to flip. The decoder fails to identify the actual data qubit error and instead assigns the X error to the boundary. (c) Logical performance comparison with and without leakage errors.

logic and 5ns latency on Xilinx FPGAs, demonstrating that real-time leakage suppression can be achieved at low cost. Further evaluations regarding the applicability of ERASER to alternatives for LRCs and an analysis of its performance under different noise models can be found in the Appendix.

Overall, this paper makes the following contributions:

(1) Our studies show that always scheduling LRCs throughout program execution (Always-LRCs scheduling) has limited efficacy.

(2) We propose *ERASER*, a dynamic LRC scheduling policy that predicts the subset of qubits that may have leaked and only applies LRCs to those qubits in real time. To the best of our knowledge, this is the *first* paper that proposes real-time leakage suppression.

(3) We propose a *Leakage Speculation Block* to accurately detect leakage and *Dynamic LRC Insertion* to adapt the QEC schedules.

(4) We propose *ERASER+M*, which extends ERASER to leverage the capabilities of multi-level readout.

2 BACKGROUND AND MOTIVATION

2.1 The Surface Code

A distance d surface code encodes a logical qubit using an alternating lattice of d^2 data and d^2-1 parity qubits, as shown in Figure 2(a). Periodically, *syndrome extraction* circuits entangle the data qubits with their neighboring parity qubits to project any errors on the data qubits into *Pauli* errors or I (none), X (bit flip), Y (bit and phase flip), and Z (phase flip) errors on the parity qubits [21, 25, 38, 39, 54]. Each parity qubit and its neighboring data qubits execute a quantum circuit to measure a 4-qubit operator, called a *stabilizer*, in a *syndrome extraction round*. The surface code uses two types of stabilizers, X and Z, which correct Z and X errors, respectively. The surface code can correct arbitrarily many errors provided the errors do not form an *error chain* (a sequence of adjacent errors) of length more than $\lfloor (d-1)/2 \rfloor$.

2.2 Decoding Errors on the Surface Code

Errors on the logical qubit are detected by periodically executing syndrome extraction circuits, which measure the parity checks. These parity checks, also called *syndromes*, are used to identify errors on the logical qubit in *real-time* by pairing or matching the non-zero parity bits using graph algorithms, such as *Minimum-Weight Perfect Matching (MWPM)*. This process is known as decoding and is performed independently for X and Z stabilizers. For example, matching the non-zero or flipped Z syndrome bits α and β enables the decoder to accurately identify the X error on the data qubit shown in *Case-1* of Figure 2(b). In practice, decoders simultaneously decode d consecutive rounds of syndromes to tolerate operational errors in syndrome extraction. This constitutes a *logical* or *QEC cycle*. Amongst decoders for the surface code, MWPM is widely recognized as the gold standard for decoding surface codes because of its high accuracy [16, 17, 33, 56, 70].

2.3 Pauli+ Errors and Leakage Errors

Not all errors on data qubits can be classified as Pauli errors on real quantum hardware. In recent demonstrations of QEC codes, Google identified a class of errors in addition to decoherence and operational errors that reduce the performance of QEC [1]. These errors are referred to as *Pauli+* errors and are fundamentally hard to tackle for two reasons. First, it is difficult to characterize these errors in real systems. Second, these errors can cause correlated errors, and thus, a decoder unaware of such correlations may be unable to handle such errors [1, 6, 65].

Leakage errors, which occur when a qubit leaves the computational basis ($|0\rangle$ and $|1\rangle$) and enters a higher energy state $|L\rangle$, are the most damaging class of Pauli+ errors because leaked physical qubits spread errors onto other qubits through two-qubit operations [1, 3, 5–8, 24, 48, 53, 69]. As these two-qubit operations are calibrated for only the computational basis, performing an operation between a leaked qubit and an unleaked qubit can lead to either (1) a random error modifying the unleaked qubit's state (which can be modeled as a random Pauli error), or (2) the unleaked qubit becoming leaked through *leakage transport* from the leaked qubit [48, 49, 53, 69].

Although leakage errors are less frequent than gate and measurement errors, they significantly degrade the logical performance because the errors spread by leakage errors obfuscate other errors from getting detected. For example, *Case-2* of Figure 2(b) shows

how the leaked qubit at the center of the code lattice leads to faulty syndrome extraction on its adjacent Z stabilizer β, causing it not to flip. Now, the decoder observes only a single non-zero Z stabilizer α and assigns an X error to the data qubit on the boundary of the lattice, thereby introducing an error itself while the actual X error remains undetected. Moreover, the leaked qubit remains faulty, creating a pathway for the leakage to spread onto other qubits, leading to an even greater possibility of errors in future syndrome extraction rounds. Consequently, the logical error rate increases, degrading the performance of QEC.

For example, Figure 2(c) shows the logical error rate of a distance 7 surface code over multiple QEC cycles. After the first cycle, the logical error rate is 27× higher in the presence of leakage. Moreover, the impact of leakage accumulates over multiple cycles. For example, the logical error rate increases by only 5× after five QEC cycles in the absence of leakage errors, whereas it increases by nearly 100× in the presence of leakage errors. Leakage errors rapidly widen the gap in logical performance with increasing QEC cycles, going from 27× to 467× in just five cycles. The sharp decline in logical performance shows that *leakage errors pose a significant barrier to scaling up logical qubits and realizing fault tolerance.*

2.4 Prior Works on Leakage Error Reduction

There are several prior works that focus on leakage error mitigation that can be classified into three broad categories:

(1) **Post-processing**: This approach identifies leakage errors from stabilizer flips observed during syndrome extraction [8, 69]. The drawback of this approach is that it requires many rounds to determine leakage errors accurately, and thus, it is mainly used to post-select or filter trials that had leakage errors during memory experiments on real systems.

(2) **Calibrating operations on leaked states**: This approach mitigates leakage by using new operations on leaked qubits that interact with states ($|L\rangle$) outside the computational basis [49, 52, 53]. Such approaches are either inherently specific to the underlying quantum processor [53] or require calibrating custom pulses to interact with higher energy states [43, 49]. Thus, such approaches are not the focus of this paper, which tackles leakage in a manner *generalizable* to any processor.[1]

(3) **SWAP-Based Leakage Removal**: This involves swapping *leaked data* qubits with *unleaked parity* qubits during syndrome extraction. The modified syndrome extraction circuit is called a *leakage reduction circuit* or *LRC* [3, 6, 24, 63].[2] The measurement and reset operations post-SWAP eliminate leakage from the data qubit. LRCs are scheduled periodically to minimize both parity and data qubit leakage, as shown in Figure 3 for the $d = 3$ code. In round R_1, no LRCs are performed, and the parity qubits are measured and reset during usual syndrome extraction, thus removing any leakage from the $d^2 - 1$ parity qubits. In round R_2, $d^2 - 1$ data qubits are scheduled for LRCs (each data qubit is swapped with a unique parity qubit). The LRC in round R_3 removes leakage from the remaining data qubit. LRCs are a straightforward approach for leakage reduction readily implementable on any device as their only overhead is modifying the syndrome extraction circuit.

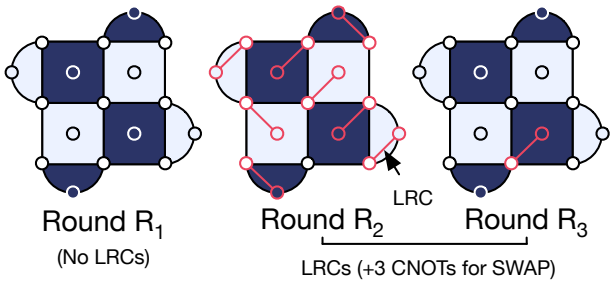

Figure 3: An example of a SWAP LRC schedule.

2.5 Limitations of LRCs

LRCs have two fundamental limitations. *First*, LRCs are unoptimized for reducing the impact of leakage transport as it has only been observed recently on real systems [52, 53]. *Second*, LRCs are inefficiently scheduled: qubits do not have leakage errors every round, so using LRCs only adds additional points of failure during syndrome extraction.

2.6 Goal

Ideally, we want greater accuracy while maintaining the simplicity of LRCs to mitigate leakage errors. Our proposed solution ERASER achieves this goal.

3 ARE ALWAYS-LRCS A GOOD IDEA?

We discuss the limitations of LRCs, specifically their poor performance in the presence of leakage transport.

3.1 LRCs facilitate leakage transport

An LRC, shown in Figure 4(b), removes leakage from a data qubit (D) by swapping it with a parity qubit (P). However, an LRC may introduce leakage onto P via leakage transport when D is leaked. In such a situation, the LRC may introduce leakage rather than remove it as intended. In the following section, we model the introduction of leakage errors during syndrome extraction with and without an LRC. A summary of the notation used in this section is shown in Table 1.

3.1.1 Leakage Errors Without LRCs. Consider a syndrome extraction round without an LRC, as shown in Figure 4(a), and suppose that the parity qubit P is leaked. During the round, P may transport leakage to one of its neighboring data qubits, which we denote D, whereas any leakage on P will be removed once it is reset. Thus, we are interested in the probability that *D becomes leaked by the end of the round, given P is leaked before the start of the round.* We denote this probability as $\mathsf{P}(L_{\text{data}}|L_{\text{parity}})$ as designated in Table 1.

D can only incur leakage through either (1) operation errors through CNOTs with neighboring parity qubits or (2) a transport error in the CNOT with P. For calculating (1), we note that the probability of the k-th CNOT causing leakage (not due to transport)

[1] We include results for ERASER combined with such specialized operations in the Appendix (Section A.2).

[2] In this paper, LRCs refer to SWAP LRCs. While there are other variants of LRCs which we discuss in Related Work (Section 7), these are impractical because they require denser device connectivity.

Table 1: Notation and Constants Used in Section 3.1

Notation	Explanation
$\mathsf{P}(L_{\text{data}}\|L_{\text{parity}})$	Probability that a data qubit leaks given a parity qubit is already leaked.
$\mathsf{P}(L_{\text{parity}}\|L_{\text{data}})$	Probability that a parity qubit leaks given a data qubit is already leaked.
p_ℓ	Probability of CNOT leakage error, equal to $0.1p = 1 \times 10^{-4}$.
p_{LT}	Probability of CNOT leakage transport, equal to 0.1.

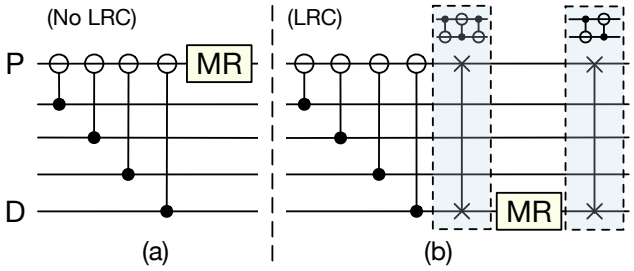

Figure 4: Syndrome extraction (a) without an LRC, and (b) with an LRC. In (b), one SWAP swaps the parity and data qubit states, and another swaps them back.

is $(1-p_\ell)^{k-1}p_\ell$, and so (1) is the sum of these probabilities for $1 \le k \le 4$. Then, combining (1) and (2), $\mathsf{P}(L_{\text{data}}|L_{\text{parity}})$ is found as in Equation (1), and we estimate this quantity to be about 10%.

$$\mathsf{P}(L_{\text{data}}|L_{\text{parity}}) = p_{\text{LT}} + \tilde{④}_{k=1}(1-p_\ell)^{k-1}p_\ell \quad (1)$$

3.1.2 Leakage Errors with LRCs. Now, we consider syndrome extraction with an LRC, as in Figure 4(b), and suppose that now the data qubit D is leaked. We are interested in the probability P becomes leaked by the end of the round, given D is leaked before the start of the round. We denote this probability as $\mathsf{P}(L_{\text{parity}}|L_{\text{data}})$ as in Table 1. However, unlike the situation without an LRC, we note that P is used in nine CNOTs. Furthermore, P interacts with D six times. However, only *four* of these CNOTs occur before D is reset and can thus cause leakage transport. The other two CNOTs occur after D is reset and are unlikely to cause leakage transport.

As with the prior calculation, we can separate $\mathsf{P}(L_{\text{parity}}|L_{\text{data}})$ into (1) a probability of leakage caused by operation error and (2) a probability of leakage caused by leakage transport. Thus, $\mathsf{P}(L_{\text{parity}}|L_{\text{data}})$ is found as in Equation (2), which we estimated to be about 34%.

$$\mathsf{P}(L_{\text{parity}}|L_{\text{data}}) = \tilde{⑨}_{k=1}(1-p_\ell)^{k-1}p_\ell + \tilde{④}_{k=1}(1-p_{\text{LT}})^{k-1}p_{\text{LT}} \quad (2)$$

3.1.3 Impact of Leakage Transport. As $\mathsf{P}(L_{\text{parity}}|L_{\text{data}})$ is about 3× larger than $\mathsf{P}(L_{\text{data}}|L_{\text{parity}})$, we expect that LRCs significantly contribute to increasing the amount of leakage on a logical qubit. This is indeed the case. Figure 5 shows the *leakage population ratio (LPR)*, or the probability that a given physical qubit on the logical qubit is leaked, over 10 QEC cycles in a $d = 7$ code at $p = 1 \times 10^{-3}$. We observe two trends corroborating our analytical results in Equation (1) and Equation (2). *First*, the LPR spikes after even rounds, which are rounds with LRCs. *Second*, each spike generally increases the LPR over the last spike, increasing the LPR over time.

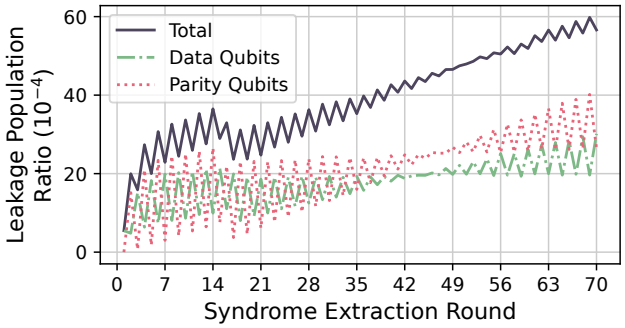

Figure 5: Leakage population ratios for $d = 7$ at $p = 1 \times 10^{-3}$ over 70 rounds (10 QEC cycles). Lower is better.

3.2 LRCs are inefficiently scheduled

The state-of-the-art LRC policy schedules LRCs every alternate round such that rounds without LRCs remove leakage from parity qubits, and rounds with LRCs remove leakage from data qubits. However, such scheduling is inefficient because the additional CNOTs in LRCs create new sources of failure. Ideally, we want to use LRCs only to remove leakage errors when they occur.

To assess the impact of the extra LRC operations on the logical performance (LPR and LER), we compare state-of-the-art LRC scheduling to an idealized scheduling policy that schedules LRCs for qubits as soon as they are leaked. Figure 6 shows the LPR and LER for both policies over 10 QEC cycles for a $d = 7$ code at $p = 1 \times 10^{-3}$. The LPR continues to increase for the state-of-the-art policy, resulting in 10× higher LER than the idealized policy. The performance gap is due to the idealized policy scheduling significantly fewer LRCs: the idealized policy schedules *one LRC every three QEC cycles* whereas the state-of-the-art policy schedules *24 LRCs every round.*

3.3 Characterizing the Spread of Leakage

To better understand how leakage spreads on a real system, we perform density matrix simulations of a Z stabilizer on the surface code. Our simulation implements the leakage phenomena observed by Google during their recent demonstration of a distance 5 surface code on their Sycamore processor [1, 53]. As Google Sycamore's leakage phenomena are reported to interact with the $|3\rangle$ state, our simulation uses *ququarts*,[3] where $|L\rangle$ corresponds to $|2\rangle$ and $|3\rangle$. Figure 7(a) provides an overview of our simulation, which simulates the spread of leakage originating from a single leaked data qubit q_0 across a Z stabilizer over an LRC round followed by a no-LRC round. During syndrome extraction, each CNOT can incur errors due to (1) leakage transport, (2) R_X error on unleaked qubits if one operand is leaked, and (3) leakage injection, as shown in Figure 7(b). The R_X

[3] Ququarts are a quantum superposition of $|0\rangle$, $|1\rangle$, $|2\rangle$, and $|3\rangle$.

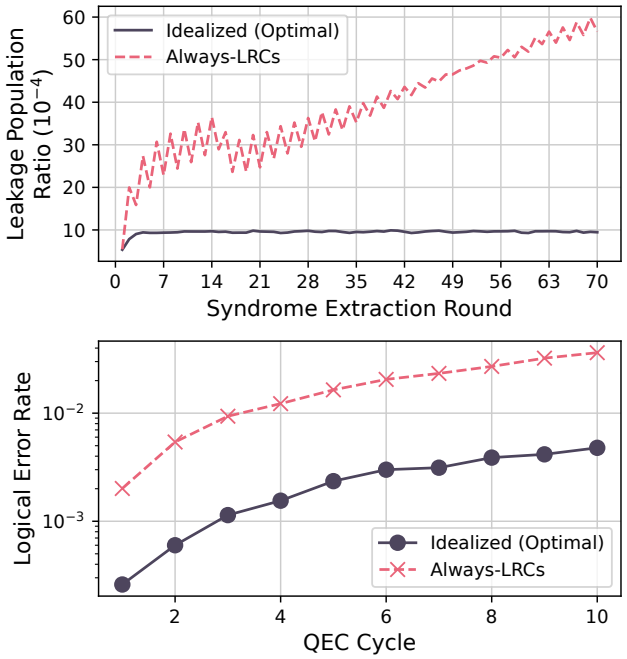

Figure 6: LPR (top) and LER (bottom) comparison between state-of-the-art and idealized LRC scheduling.

errors in our experiment are fixed to be $R_X(0.65\pi)$, which was the error measured during leakage studies on Google Sycamore [53].

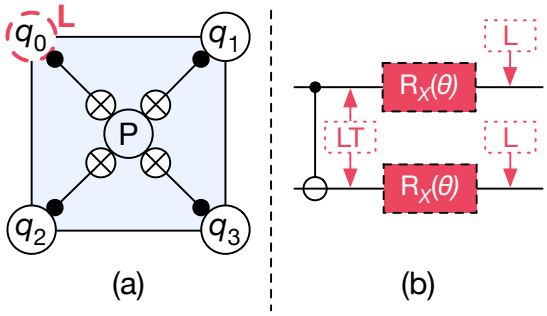

Figure 7: (a) The simulated Z stabilizer. The density matrix simulation starts with q_0 initialized in $|2\rangle$. (b) CNOTs are followed by leakage transport, R_X errors, and leakage injection.

Figure 8 shows the movement of leakage from the leaked parity qubit and the impact of leakage on the Z stabilizer measurement. We discuss three points of interest. At point Ⓐ, which marks the end of the LRC with q_0, we observe that the parity qubit P has significantly leaked due to interactions with q_0, confirming that *LRCs do facilitate leakage transport*. Consequently, P then spreads leakage errors onto the other data qubits during the no-LRC round, thus increasing the leakage population. Point Ⓑ shows the first point where P is affected by leakage during a CNOT with q_0 (CNOT #4). If P was measured at this point, we would get a random outcome; note that we ideally want to measure P as 0 as there are no X errors on the data qubits. As syndrome extraction continues, the measurement probabilities fluctuate. At point Ⓒ, before the measurement of P, the probability of measuring the correct outcome is slightly better than random. Thus, leakage errors interfere with syndrome extraction measurements by *inducing random measurement results*.

We note that as our simulations in this section are restricted to a single stabilizer, the results observed *understate* the impact of leakage. In reality, the leakage error on q_0 will also spread to the rest of the logical qubit and cause more errors. We refer to Google's recent studies on leakage and recently published qutrit simulations for more extensive analyses on the spread of leakage across an entire logical qubit [48, 53].

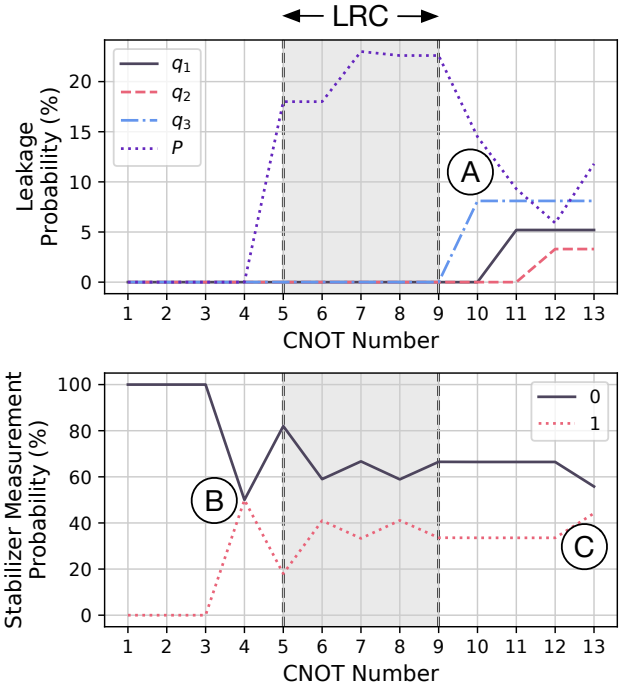

Figure 8: (top) Spread of leakage errors, and (bottom) the effect of leakage on stabilizer measurement probability. We do not show qubit q_0's leakage probability as it begins the simulation already initialized in $|2\rangle$.

4 ERASER: INSIGHTS AND DESIGN

We propose ERASER that judiciously schedules LRC operations such that errors from both leakage and LRC operations are simultaneously minimized. Figure 9 gives an overview of ERASER. The *Leakage Speculation Block (LSB)* uses the current syndrome to speculate a subset of qubits that may have leaked. The *Dynamic LRC Insertion (DLI)* block interrupts the *QEC Schedule Generator (QSG)* to modify the syndrome extraction circuits for the next round and issues LRC operations only for qubits speculated as leaked.

Enabling adaptive LRCs presents two key challenges. *First*, we must accurately speculate leakage. Failure to do so causes leakage to remain and degrade performance. *Second*, the control processor must integrate LRC operations into QEC schedules in real time to prevent QEC cycles from stalling. We discuss the insights to overcome these challenges.

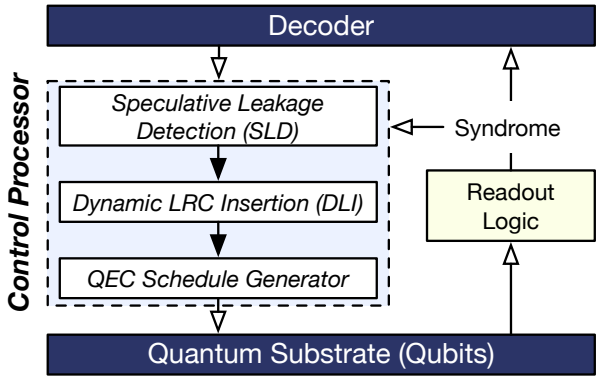

Figure 9: Overview of ERASER.

4.1 Leakage Speculation Block: Challenges

The only information available about the qubits during syndrome extraction that could be used to detect leakage errors is the measured syndrome. We discuss how often leakage errors impact syndrome extraction and the challenges with precisely detecting leakage errors using syndromes.

4.1.1 Is Leakage Visible or Invisible from Syndromes? Our analysis shows that leakage errors can be broadly classified into two categories: *visible* which immediately affect syndrome extraction, and *invisible* which persist for multiple rounds before affecting syndrome extraction. We discuss how long a data qubit potentially remains invisible without LRCs; note that parity qubit leakage does not accumulate as these qubits are reset every round. This happens in two scenarios:

(1) When the leaked data qubit causes an error in syndrome extraction. This can be modeled as a **depolarizing error** and affects parity qubit measurement with a 50% probability.

(2) When the leaked data qubit **transports leakage** resulting in the parity qubit accumulating leakage. When measured, the parity qubit will be randomly classified as a 0 or 1. There is a 50% probability that the error affects the measurement.

As a data qubit neighbors at most four parity qubits, the probability a leaked data qubit is invisible in a round is $(\frac{1}{2})^4 = \frac{1}{16}$. As a qubit remains invisible until it affects a parity qubit measurement (probability is $1 - \frac{1}{16} = \frac{15}{16}$), the probability a leaked data qubit remains invisible for r rounds is given by Equation (3).

$$\mathsf{P}_{\text{invis}}(r) = \frac{15}{16} \times \frac{1}{16}^{r} \tag{3}$$

Table 2 shows the probability of a leaked data qubit remaining invisible over multiple rounds. Note that more than 99% of leakage errors affect syndrome extraction within two rounds, resulting in most leakage errors becoming quickly visible.

ERASER: Insight #1

Visible leakage errors are the *most common variant* of leakage errors, and optimizing LRCs for them is sufficient.

Table 2: Invisible Leakage Error Probability

Rounds Spent Invisible	Probability ($\mathsf{P}_{\text{invis}}$) in %age
0	93.8
1	5.90
2	0.36
3	0.02

4.1.2 Challenges in Exact Leakage Detection. Syndrome bit flips not only result from leakage errors but also arise from other types of errors such as decoherence, gate, and measurement errors. This makes the reliance on syndromes to detect leakage errors extremely challenging. Furthermore, unlike other errors, leakage errors do not cause syndrome measurements to flip according to a specific pattern. For example, an X error on a data qubit only causes its adjacent Z syndromes to flip, whereas a measurement error causes the same syndrome measurement to flip across consecutive rounds. Unlike such errors, leakage errors cause random syndrome measurements to flip. For example, a leaked data qubit can cause any arbitrary combination of its four neighboring parity qubits to flip, making it difficult to detect the leakage during syndrome extraction.

Instead of attempting to *identify exactly where* leakage errors have occurred, we use the insight that leveraging the flipped syndrome bits to *speculatively detect* a leakage with high accuracy is sufficient. However, even performing such speculative detection is nontrivial as there is an inherent trade-off between LRC scheduling frequency and performance. Speculating too conservatively schedules too many LRCs, degrading the QEC code's performance as the extra LRC operations increase errors during syndrome extraction. On the other hand, speculating too aggressively schedules LRCs too infrequently, also degrading performance as leakage is not removed in time. To maximize performance, ERASER achieves a sweet spot between the two and schedules LRCs on a data qubit when at least 50% of its neighboring parity qubits flip.

ERASER: Insight #2

Speculative leakage detection has an inherent trade-off between LRC scheduling frequency and performance. Scheduling LRCs too aggressively or too conservatively will degrade performance by causing more errors to occur.

4.2 Leakage Speculation Block: Design

ERASER uses the current syndrome to speculate the data qubits that may have encountered leakage. The *Leakage Speculation Block (LSB)* maintains a *Leakage Tracking Table (LTT)* with one entry per data qubit, as shown in Figure 10. The LSB analyzes the current syndrome and speculates if a qubit has leaked. If it identifies a leakage, the corresponding LTT entry is marked as *leaked*.[4] The LSB also maintains a *Parity qubit Usage Tracking Table (PUTT)* to track the allocation of parity qubits for LRC operations on the data qubits. In the Always-LRCs scheduling, LRCs span two consecutive syndrome rounds as the number of data qubits exceeds the number

[4]We use the term *previous* round to indicate the last syndrome extraction round. We refer to the syndrome obtained from this round as the *current* syndrome. The decision to introduce LRCs is made for the *next* syndrome extraction round.

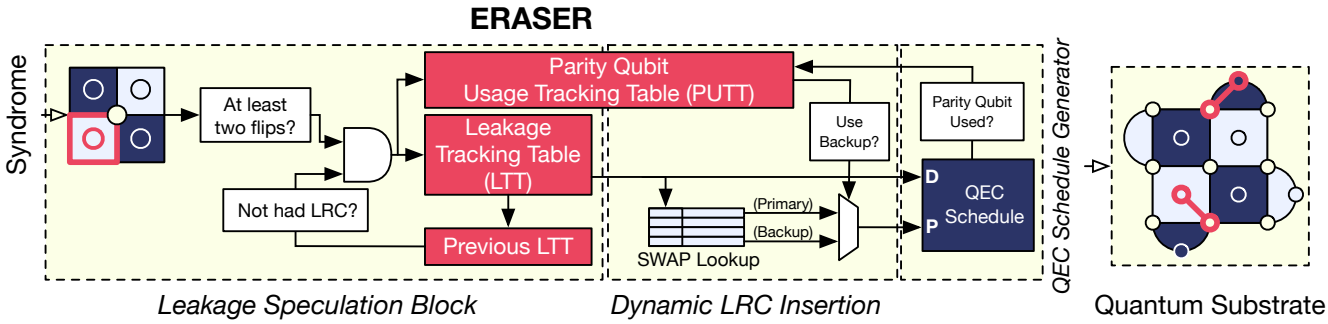

Figure 10: Overview of ERASER's microarchitecture.

of available parity qubits for swapping, and more than one data qubit may require swapping with the same parity qubit. As ERASER dynamically schedules LRC operations, the conflict is resolved by scheduling LRCs for the data qubits on any adjacent available parity qubit. The rules for handling the LTT and PUTT entries are discussed in the following subsection. The *Dynamic LRC Insertion (DLI)* block uses the LTT and PUTT entries to introduce LRCs.

4.2.1 Speculating Leakage on Data bits. A data qubit may have two, three, or four neighboring parity qubits. If no LRC operations were scheduled for a data qubit in the previous round (which yields the current syndrome) *and* at least half of the neighboring parity qubits flip, then the LSB block marks the LTT entry for the corresponding data qubit as *leaked* to schedule LRC operations in the next round. We choose half the number of parity qubits as a cutoff because, on average, half the parity qubits are expected to flip if there is a leakage error. Note that if LRC operations were scheduled on that particular data qubit in the previous round, any leakage on the qubit would have been removed, and we do not speculate any leakage even if 50% of its neighboring parity qubits flip.

4.2.2 Handling Parity bits Usage Tracking. In Always-LRCs, each data qubit has a *primary* parity qubit it swaps with to perform an LRC. However, as there are d^2 data qubits but only $d^2 - 1$ parity qubits, LRC operations cannot be scheduled for all data qubits in the same round. Instead, one LRC must be carried over into the next round. For example, Figure 11(a) shows how both the leaked qubits conflict with the same primary parity qubit, and the LRC operations for both of them cannot be scheduled in the same round. To overcome this limitation, we leverage the insight that as ERASER schedules LRCs dynamically, only a subset of data qubits will require LRC operations in the same round. Thus, LRCs need not be carried over to the next round. To facilitate this, we select one of the neighboring parity qubits for LRC operations based on availability at runtime instead of allocating *primary* parity qubits offline. The LSB allows each data qubit to use any neighboring parity qubit and marks it as *used* in the PUTT. Now, the LRC operations for both leaked data qubits in Figure 11(b) can be scheduled simultaneously.

However, a completely arbitrary selection of parity qubits may lead to the accumulation of leakage on parity qubits if the same parity qubit gets selected for LRCs over multiple consecutive rounds for different data qubits. This happens because the associated parity

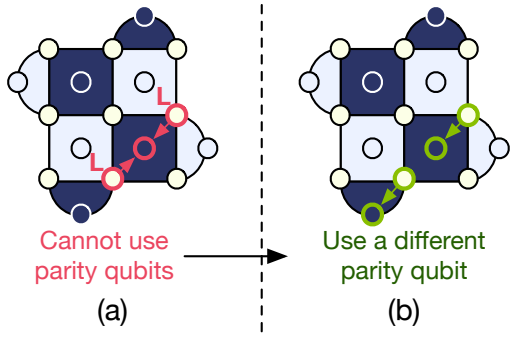

Figure 11: (a) Two leaked data qubits must perform an LRC but have the same primary parity qubit. (b) Arbitrarily assign data qubits to parity qubits.

qubit continuously gets swapped and is not reset for a prolonged duration. Note that each parity qubit may be used by up to four data qubits in case of such an arbitrary selection. To resolve this bottleneck, if a parity qubit has participated in an LRC in the previous round, it is marked as used in the PUTT and is not used for LRCs in the next round. The parity qubits that participated in LRC operations in the previous round will now be measured and reset in the next round, eliminating any leakage. The limited arbitrary selection of parity qubits enables us to schedule more LRCs in the same round and reduce leakage errors on both data and parity qubits.

4.3 Dynamic LRC Insertion: Challenges

Always-LRCs scheduling occurs offline before program execution by compiling syndrome extraction circuits down to the native gates of the quantum device [15, 51, 59]. During program execution, the control processor repeatedly executes these gates in each syndrome extraction round. However, as ERASER only schedules LRCs when needed, it must interrupt the instruction supplier or the *QEC Schedule Generator (QSG)* to update the schedule for the subset of qubits it has identified as leaked in the subsequent syndrome round. Note that the real-time constraint for scheduling ranges in the order of a few tens of nanoseconds. The QSG must know by the *fourth* CNOT in the syndrome extraction circuit whether to schedule an LRC, as

it will need to perform a SWAP after this CNOT to execute the LRC. Figure 12 shows this leaves about 120ns between obtaining the previous syndrome and the end of the fourth CNOT in the current round, assuming each CNOT takes 30ns (according to Sycamore latencies) [1, 2]. Failure to resolve whether or not to introduce LRC operations within this time either causes the qubits to idle until a decision is made or moves the LRC operations to the next round, causing the leakage to remain. Finally, as ERASER must be co-located within the control processor, it must fit on FPGAs to enable integration with existing quantum systems [2, 16, 26, 36, 50, 61].

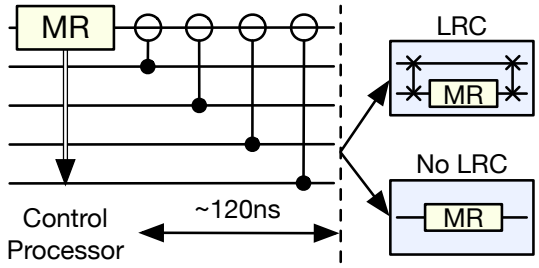

Figure 12: After a qubit is measured and the syndrome bit is sent to the control processor, there is a 120ns window to determine whether to schedule an LRC or not.

4.4 Dynamic Leakage Insertion: Design

After marking qubits as leaked in the LTT, ERASER attempts to schedule LRCs for all leaked data qubits while not scheduling parity qubits marked as used in the PUTT. We note that such scheduling is nontrivial as it requires solving a *maximum matching* problem in real time. We must pair each leaked data qubit with a unique unused parity qubit to swap with during an LRC. Also, we must maximize the number of leaked data qubits scheduled for LRC operations to ensure all leaked data qubits are reset in the next round.

To solve this problem efficiently, we propose using a lookup table containing pre-determined *primary* and *backup* SWAP neighbors for each data qubit; we call this lookup table the *SWAP Lookup Table*. For each leaked data qubit, ERASER uses the SWAP Lookup Table to get a neighboring parity qubit to swap with. If the parity qubit is already marked as used in the PUTT, ERASER looks through the backup parity qubits and repeats this process. By default, our design maintains one backup parity qubit for each data qubit.

4.5 QEC Schedule Generation: Design

After identifying LRCs that need to be scheduled, the control processor must execute the LRC operations in the next syndrome extraction round. By default, the control processor executes standard X and Z stabilizer circuits. The DLI interrupts the QEC Schedule Generation (QSG), appends the instruction schedules by inserting the extra CNOTs corresponding to the LRC operations, and replaces the measurement operations on the associated parity qubits with those on the data qubits selected for LRC operations.

4.6 Enhancing ERASER Using Multi-Level Readout Discriminators: ERASER+M

The performance of ERASER is limited by the LSB's ability to detect leakage using syndromes. ERASER uses a passive leakage detection strategy because existing measurement discriminators are two-level classifiers that can only classify a qubit into states $|0\rangle$ and $|1\rangle$. Consequently, leakage is never actively detected because a leaked qubit is randomly classified into a 0 or 1. Multi-level discriminators, on the other hand, can also classify leaked states, and recent studies at the device level indicate they are promising for tackling leakage [12, 41]. We propose ERASER+M to integrate multi-level discriminators with ERASER for enabling more accurate leakage detection and LRC scheduling. It requires modifications to the LSB and QSG blocks, as are discussed next and summarized in Figure 13(a, b), respectively. Note that the DLI block does not require any modifications.

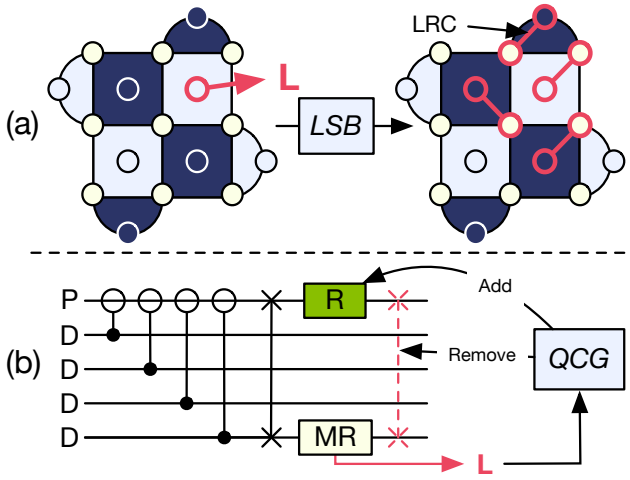

Figure 13: Key modifications for ERASER+M. (a) The LSB schedules LRCs for data qubits adjacent to a parity qubit measured in $|L\rangle$. (b) The QCG modifies LRC operations upon measuring data qubit leakage.

4.6.1 Modifications to the LSB. If a parity qubit is classified as $|L\rangle$ in the current round, we assume it has transported leakage to one or more of its neighboring data qubits. Therefore, we speculate all its adjacent data qubits have been potentially leaked and mark the corresponding entries in the LTT so that LRC operations can be scheduled on these qubits in the next round.

4.6.2 Modifications to the QSG. During syndrome extraction with an LRC, if the data qubit is classified as $|L\rangle$, we observe that the parity qubit has a meaningless state since the SWAP during the LRC would have failed due to the data qubit leakage. Either the parity qubit has leaked or has a random unleaked state. Consequently, performing the SWAP after the data qubit reset is unnecessary as no useful information will be returned to the data qubit. However, the SWAP was also the only way to return the parity qubit to $|0\rangle$, and this must be done before the next round. Thus, if the data qubit is classified as $|L\rangle$, the QSG (1) schedules a reset operation on the parity qubit and (2) squashes the second SWAP in the LRC circuit.

5 EVALUATION METHODOLOGY

In this section, we discuss our evaluation methodology before discussing our results.

5.1 Surface Code Parameters

We consider *rotated surface codes* with code distances (d) ranging from $d = 3$ to $d = 11$. Rotated codes have lower resource overheads ($2d^2 - 1$ qubits) compared to the *unrotated codes* ($(2d-1)^2$) [35, 65] and, therefore, have been used in recent QEC studies and real-system demonstrations [1, 53].

5.2 Error Model

In this subsection, we discuss the error model used in our evaluations corresponding to different types of errors.

5.2.1 Modeling Operation Errors. We consider a physical error rate of $p = 1 \times 10^{-3}$ and a *circuit-level* error model that injects (1) *depolarizing* errors on data qubits with probability p at the start of a round, (2) *measurement* errors on qubits with probability p, (3) *depolarizing* errors on qubit operands after each CNOT or H gate with probability p, and (4) *initialization* errors on qubits after a reset with probability p [27, 42].

5.2.2 Modeling Leakage Errors. Modeling leakage in memory experiments is inherently challenging to approximate in a *tractable* manner [1, 48, 63]. To ensure our results are reflective of real systems, we design our leakage error model based on prior studies on real systems [1, 49, 53, 64, 69]. Our simulations also inject and track leakage in a manner consistent with prior work [6, 7, 24, 48, 63].

We extend the circuit-level error model to inject leakage errors (1) on data qubits at the beginning of each round with probability $0.1p$ to model ***environment-induced*** leakage and (2) on qubit operands after CNOT operations with probability $0.1p$ to model ***operation-induced leakage***.[5] When an unleaked qubit interacts with a leaked qubit through a CNOT, we inject a random Pauli error (I, X, Y, Z) on the unleaked qubit and apply a ***leakage transport*** with a 10% probability.

Our implementation of leakage transport conservatively assumes that the source qubit remains leaked after a leakage transport. Section A.1 reports results for an alternative implementation of leakage transport, where the source qubit may return to the computational basis provided the other qubit is not leaked.

5.2.3 Modeling the Measurement of Leaked bits. The output state of a qubit (0 or 1) is determined by a measurement discriminator [37, 50, 64]. If a standard two-level discriminator measures a leaked qubit, the outcome will be random because the discriminator is not trained to classify $|L\rangle$. We assume this for ERASER. For ERASER+M, we assume that a multi-level discriminator, which classifies $|0\rangle$, $|1\rangle$, and $|L\rangle$, is erroneous at a rate of $10p$ to be consistent with results on real systems [12, 49].

[5] Our error model also implements seepage, or the return of a leaked qubit to the computational basis, at the same rate as leakage. If a qubit is leaked, it can return to the computational basis in a randomly initialized state with probability $0.1p$.

5.3 Simulation Infrastructure

We use Google's *Stim* simulator [27], a state-of-the-art framework for performing state-preservation, or *memory*, experiments [1, 4, 9, 29, 30, 71], which we have extended to simulate leakage errors. Our evaluations go up to ten *QEC cycles* (each cycle is d rounds) to evaluate the efficacy of our design over time. We use Minimum-Weight Perfect Matching decoding [22], but any other decoder may be used as well.

5.4 Evaluation Metrics

We use the (1) *logical error rate* and (2) *leakage population ratio* to evaluate our policies. The logical error rate (LER) quantifies the ability to suppress errors [1, 6, 16, 17, 34, 46, 53, 63, 64, 66–68]. The LER is defined in Equation (4):

$$LER = \frac{n_{\text{logical\_errors}}}{n_{\text{experiments}}} \tag{4}$$

Leakage population ratio (LPR) quantifies the number of leaked qubits at any time [5, 7, 52, 53]. This metric is widely used in the devices community and is defined in Equation (5):

$$LPR = \frac{1}{n_{\text{experiments}}} \times \tilde{\mathrm{O}}_{\text{experiments}} \frac{n_{\text{leaked}}}{n_{\text{qubits}}} \tag{5}$$

5.5 Hardware Cost of ERASER

To evaluate the hardware overheads of our design, we target Xilinx's off-the-shelf Kintex UltraScale+ FPGA and synthesize our design using Vivado.

6 EVALUATIONS

In this section, we discuss the performance of our proposed designs ERASER and ERASER+M.

6.1 Impact on Logical Error Rate

Logical error rate (LER) denotes the capability of a QEC code to suppress errors. A lower LER and exponentially decreasing LER with increasing code distance are desirable. Figure 14 shows ERASER improves the LER consistently with increasing distance on average by 3.3× and up to 4.3× in the best-case. ERASER+M is even more effective and achieves near-optimal LER, improving the LER on average by 8.6× and up to 26× in the best case.

At lower physical error rates such as $p = 10^{-4}$, ERASER's performance improves, reducing the LER by 5.4× on average and up to 9× compared to Always-LRCs. Concurrently, ERASER's performance is now closer in performance to ERASER+M and optimal scheduling, as error events become sparser at lower physical error rates [11, 18–20, 60] and so leakage errors become more visible.

6.2 Impact on Leakage Population Ratio

A lower leakage population ratio or LPR means a greater reduction of leakage errors. Figure 15 shows the LPR of the default $d = 11$ configuration for the competing LRC scheduling policies. ERASER consistently maintains a lower LPR and decreases the LPR by 1.5× on average and up to 2.1×. Furthermore, ERASER+M bridges the gap between optimal LRC scheduling and reduces the LPR by 2.2×

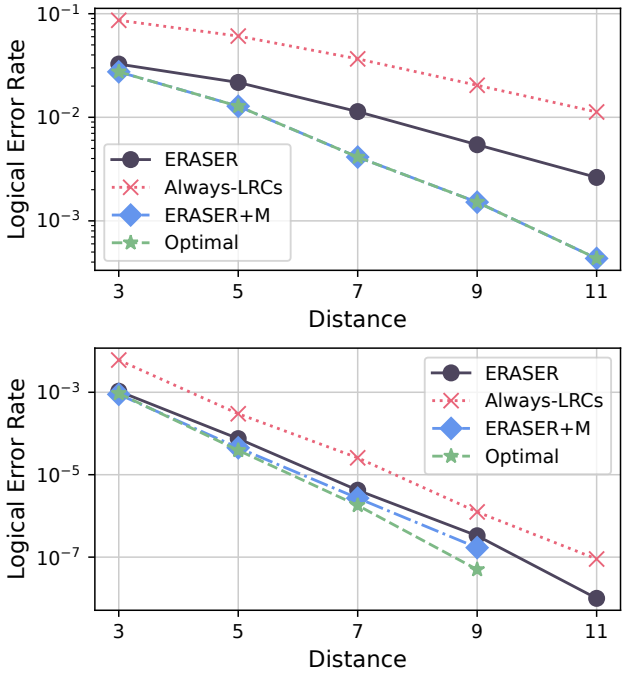

Figure 14: LER with increasing code distance for (top) $p = 10^{-3}$ and (bottom) $p = 10^{-4}$ for 10 QEC cycles. Data is not shown for $d = 11, p = 10^{-4}$ for ERASER+M and optimal LRC scheduling as it was too low to be measured accurately.

compared to ERASER, performing nearly identically to the idealized setting of scheduling LRCs.

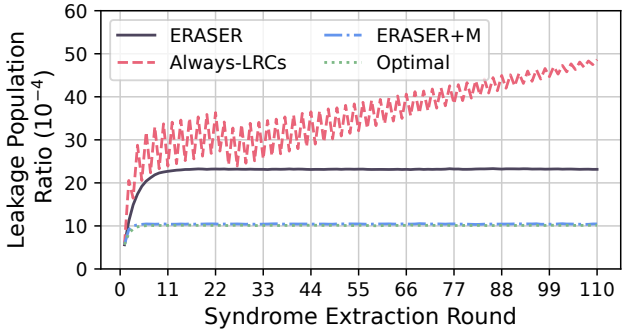

Figure 15: LPR for the baseline Always-LRCs, ERASER, ERASER+M, and optimal (idealized) LRC scheduling for default $d = 11$ configuration.

6.3 Hardware Implementation Cost

Table 3 shows the hardware resources required for implementing our proposed ERASER on standard off-the-shelf FPGAs as they are already being used to control and readout qubits on most existing quantum computers. Our implementation of ERASER requires less than 1% logic utilization up to $d = 11$ and has a worst-case latency of 5ns to speculate leakage and adapt the QEC schedules. This makes ERASER a very practical, low overhead, and accurate solution that eliminates leakage errors in real-time.

Table 3: FPGA Synthesis Results

d	LUT (%)	FF (%)
3	0.04	0.02
5	0.12	0.05
7	0.26	0.10
9	0.42	0.18
11	0.76	0.26

6.4 Performance Analysis of ERASER

ERASER is effective due to two key reasons. First, the LSB can accurately speculate most of the leakage errors. Second, ERASER schedules a significantly lower number of LRC operations. Figure 16(a) shows the average speculation accuracy of the LSB. Both ERASER and ERASER+M correctly use LRCs about 97% of the time, whereas Always-LRCs correctly speculates about 50% of the time. Table 4 further shows the average number of LRCs used per syndrome extraction round for all four policies. Both ERASER and ERASER+M reduce the number of LRCs scheduled by 16.0× on average and by up to 17.4× in the best-case.

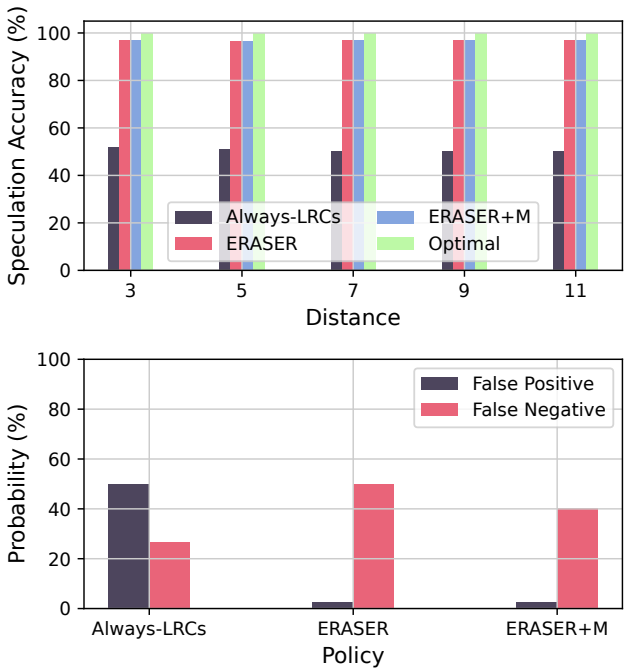

Figure 16: (top) LRC speculation accuracy, and (bottom) FPRs and FNRs for $d = 11$ over 10 QEC cycles. Data is not shown for optimal LRC scheduling as it has 100% speculation accuracy.

Table 4: Average LRCs Used Per Round

d	Always-LRCs	ERASER	ERASER+M	Optimal
3	4.2	0.27	0.26	0.005
5	12	0.81	0.79	0.015
7	24	1.52	1.50	0.034
9	40	2.40	2.38	0.058
11	60	3.45	3.41	0.089

6.4.1 Examining the 3% Gap. We further analyzed why there is a 3% accuracy gap between ERASER (and ERASER+M) and optimal LRC scheduling. Figure 16 shows the *false positive rates (FPR)* and *false negative rates (FNR)* for LRC usage across all policies. We make two observations. ERASER and ERASER+M can easily identify situations with no leakage errors, with a 3% FPR compared to a 50% FPR for Always-LRCs. Minimizing FPR is crucial as qubits are typically not leaked, so applying LRCs may create new errors. However, ERASER is not as accurate when detecting leakage, though ERASER+M can improve detection accuracy by up to 1.2×.

6.4.2 When does ERASER have False Negatives? The higher FNR of ERASER may appear alarming, but we observe that the false negatives incurred by ERASER are *hard-to-detect* leakage errors. By design, ERASER's false negatives are either (1) invisible leakage errors or (2) leakage errors that only flip one parity check, which go undetected as ERASER schedules LRCs when at least *two* parity checks have flipped. Such errors are hard to identify as they barely affect any syndrome measurements. Nevertheless, as shown with ERASER+M, which has an FNR of 40% compared to ERASER's 50%, even small reductions in the FNR can significantly improve the logical error rate, as ERASER+M has similar performance to optimal LRC scheduling.

6.5 Analysis of Trade-Offs for ERASER+M

Although ERASER+M is significantly more effective compared to ERASER, it incurs overheads of using multi-level discriminators. The measurement discriminator of a qubit is prepared by initializing it into each possible state that we want to classify, measuring it, and using the output signal to train a classification function. Typically, each execution is repeated for a few thousand trials. Multi-level discriminators must be trained to classify $|L\rangle$ states in addition to the usual $|0\rangle$ and $|1\rangle$. This results in two sources of overheads: (1) we must calibrate a single-qubit operation that can initialize a qubit in a higher energy state (such as $|2\rangle$) and (2) additional executions to prepare and measure a qubit in the higher energy state to obtain the output signal for the leaked state. This process is required for each qubit. Assuming calibrating a single-qubit operation takes about 1K shots and another 1K shots are required to calibrate the classifier for the $|L\rangle$ state, we require $2NK$ extra trials where N is the number of physical qubits on the machine. Nevertheless, we note that other strategies also leverage multi-level discrimination, and thus ERASER+M naturally synergizes with such strategies [63].

Note that ERASER is already very effective, and integrating the modifications needed for ERASER+M can be managed solely in software. Hence, the choice of using ERASER versus ERASER+M can be left to the programmer.

7 RELATED WORK

In this section, we discuss related work and compare or contrast as appropriate.

7.1 Leakage errors and their impact on QEC

Improving device qualities and increasing system sizes have accelerated the demonstration of QEC codes in recent years [1, 41, 64]. These real system studies reveal that *leakage errors* significantly degrade the performance of QEC. For example, the studies performed on Google Sycamore rely on post-processing the results to eliminate experimental results from rounds with leakage errors. While post-processing can be used during experimentation, it cannot be used during program execution on a fault-tolerant quantum computer, where errors, including leakage errors, must be suppressed in real time. In contrast, ERASER actively removes leakage errors by efficiently scheduling leakage reduction circuits.

7.2 Handling leakage errors

Although strategies for mitigating leakage errors have been studied in the past, they are either low-cost but inaccurate or accurate with added overheads [1, 41, 52, 53, 64]. *Leakage Reduction Circuits (LRCs)* remove leakage from data qubits by executing SWAPs with other ancilla or parity qubits [3, 6, 63]. There are three varieties: *Full LRCs*, *Partial LRCs*, and *SWAP LRCs*. As the former two variants of LRCs require denser device connectivity, we consider SWAP LRCs in this paper, which remove leakage errors from data qubits by swapping them with parity qubits.

Recent works have provided new leakage reduction strategies through custom operations that interact with states outside the computational basis [49, 53]. While such operations may require modifications to the quantum system [49], additional calibration overheads [43], or are specific to the underlying device [53], their performance is rather promising as they offer better performance than SWAP-based LRCs. Nevertheless, as such operations can also be erroneous and introduce leakage themselves, we observe that ERASER can improve the fidelity of such approaches as well, which we discuss at length in Section A.2.

8 CONCLUSION AND DISCUSSION

Leakage errors present a significant barrier to realizing fault-tolerant quantum computing as they degrade the performance of quantum error correction (QEC) codes. These errors cause qubits to leave computational basis states and enter higher energy states. Leakage errors are not device-specific and have been observed in both superconducting processors [41, 49, 52, 53] and ion traps [5].

Prior works actively eliminate these errors by using leakage reduction circuits (LRCs) to periodically remove leakage from data qubits through SWAPs and resets. However, always using LRCs throughout a program is sub-optimal as they introduce additional two-qubit operations that facilitate leakage transport onto other qubits and may themselves fail. Ideally, LRCs should be scheduled so that leakage is wholly removed while ensuring minimal impact from the extra LRC operations.

We propose ERASER that detects the subset of qubits that may have leaked in real-time and judiciously applies LRC operations

only on those qubits. ERASER leverages the insight that most leakage errors cause arbitrary parity check failures during QEC cycles. By identifying patterns in the failed parity checks, ERASER speculates the subset of leaked qubits. Once, the potentially leaked qubits are identified, ERASER adjusts the syndrome extraction schedules for these qubits by introducing LRC operations in real-time. The accuracy of leakage identification can be further enhanced by modifying the qubit measurement protocols to classify leaked states in addition to computational basis states. We leverage this insight to enhance ERASER using multi-level measurement classifiers. ERASER improves logical error rate by up to 4.3× compared to Always-LRCs.

ERASER is the *first* work to consider *real-time leakage suppression*, and ERASER's superior performance to Always-LRCs demonstrates that real-time, or adaptive, leakage suppression provides significant benefits over static leakage suppression, where LRCs are scheduled offline at compile-time. Our results suggest that accurately speculating leakage in real-time is an important open problem. While ERASER's speculation accuracy is rather high, its poor FNR due to *hard-to-detect* leakage errors is a significant source of logical error. Fortunately, we observe that even minor improvements in speculation accuracy, particularly in the FNR, can significantly improve the logical error rate. Thus, more sophisticated speculation strategies for leakage detection appear to be a rich and promising area for future research.

Finally, we observe that qubit loss in ion traps and neutral atom systems have a similar signature to leakage on superconducting systems, which was the predominant focus of this work [14, 31, 47, 62, 72]. As qubit losses can cause operations to fail, such systems must be capable of detecting qubit loss through loss detection mechanisms to avoid errors. Given this parallel, we expect strategies similar to ERASER may be fruitful on such systems. Furthermore, as ion traps and neutral atom systems are much slower than superconducting systems, we note that time constraints for identifying leaked qubits are more generous, allowing for more sophisticated and accurate strategies.

ACKNOWLEDGMENTS

We thank the reviewers of MICRO 2023 for their feedback. We thank Cody Jones (Google) for helpful discussions regarding leakage errors and for providing feedback on the submitted draft. We also thank Andrew Cross (IBM) and Adam Meier (GTRI) for providing feedback on the submitted draft. This research was conducted using the *Partnership for an Advanced Computing Environment (PACE)* cluster at Georgia Tech.

A APPENDIX: ADDITIONAL RESULTS

A.1 Alternative Model for Leakage Transport

The leakage transport model used in the main text conservatively assumes that the source qubit remains leaked after a transport; that is, both qubits involved in the transport are leaked after the transport finishes. In this section, we consider an alternative model, where the source qubit and receiving qubit "exchange" leakage with each other. In such a model, if the receiving qubit is not leaked, it will become leaked, whereas the source qubit will return to the computational basis in a randomly initialized state. If the receiving qubit is leaked, then the transport essentially has no effect.

Figure 17 shows the LER for all policies under this alternative model for 10 QEC cycles. As expected, all models improve quite a bit under the alternative model. However, we further note that the gap between ERASER and Always-LRCs *widens* significantly, whereas the gap between ERASER and Optimal-LRCs has narrowed considerably. Now, ERASER improves the LER compared to Always-LRCs on average by 6.5× and up to 13.4×. Concurrently, ERASER+M improves the LER on average by 8.8× and up to 24.1×.

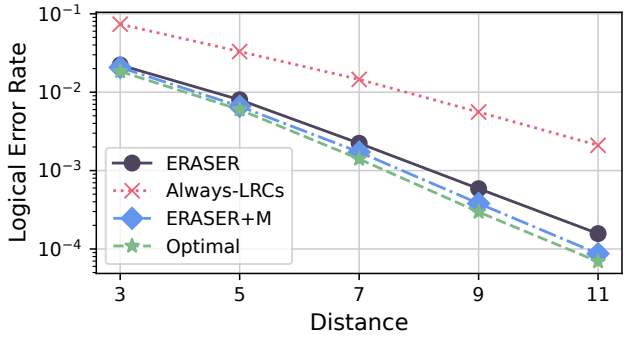

Figure 17: LER for 10 QEC cycles at $p = 10^{-3}$ for the alternative leakage transport model.

We believe that ERASER significantly improves under this alternative leakage transport model for two reasons. *First*, we note that the LPR for all policies is significantly lower. Figure 18 shows the LPR for all four policies. We note that the LPR is substantially lower under the alternative model as the number of leaked qubits is preserved during a leakage transport under this model. Hence, the LPR curves for all policies, except Always-LRCs, stabilizes. The LPR for Always-LRCs spikes after rounds with LRCs and reduces after rounds without LRCs because LRCs may fail to remove leakage due to leakage transport. *Second*, LRCs have lower error than in the original model. Consequently, the impact of ERASER's high FNR is much lower compared to the original model.

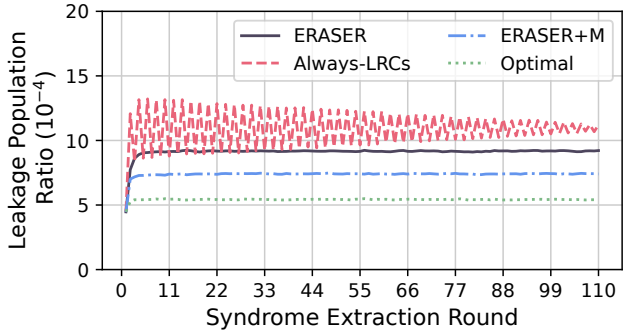

Figure 18: LPR over 110 rounds for a $d = 11$ at $p = 10^{-3}$ using the alternative leakage transport model.

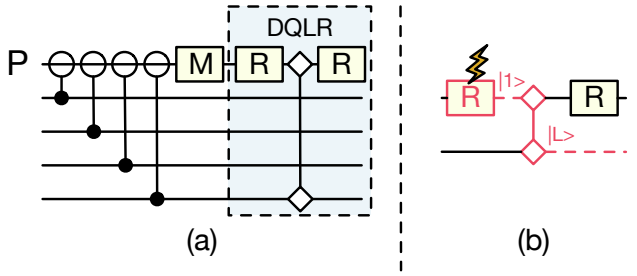

Figure 19: (a) The DQLR protocol, which leverages the two-qubit LeakageISWAP operation. (b) A reset failure on the parity qubit can cause the LeakageISWAP operation to excite the data qubit to $|L\rangle$.

A.2 Applicability of ERASER with DQLR

The evaluations in the main text consider the traditional SWAP-based LRC, which had been considered by much prior work [3, 6, 25, 63]. However, recent work has been moving towards LRCs using custom operations with tremendous success [49, 53]. As these operations exploit the underlying physics of the corresponding quantum processor, they can be calibrated for their respective systems without much difficulty. However, like any other quantum operation, these customized operations also may be erroneous. In this section, we analyze the applicability of ERASER for LRCs involving such operations. Specifically, we examine Google's *DQLR* approach [53] as shown in Figure 19(a), and we use the alternative leakage transport model from Section A.1 to ensure our results are reflective of Google Sycamore's leakage transport phenomena.

A.2.1 The DQLR Protocol. The DQLR protocol removes leakage from data and parity qubits every round by (1) performing syndrome extraction as usual, (2) resetting all parity qubits, which removes any leakage on the parity qubits, (3) using a custom operation known as a *LeakageISWAP* to remove data qubit leakage and move it to a parity qubit, and (4) resetting the parity qubits yet again. The fidelity of this operation is rather high, as (1) DQLR is not vulnerable to leakage transport, and (2) it only requires a single two-qubit operation to remove leakage. However, as shown in Figure 19(b), the DQLR protocol can introduce leakage on the data qubits if the first parity qubit reset fails (the parity qubit is initialized in $|1\rangle$ instead of $|0\rangle$), as then the data qubits may be excited to $|2\rangle$[6]. Thus, much like SWAP-based LRCs, overusing the DQLR protocol is risky, as it may introduce leakage even when there was no leakage to begin with.

A.2.2 Results. We examine the applicability of ERASER to the DQLR protocol and assume that the LeakageISWAP gate has the same fidelity as a CX gate. We compare the baseline DQLR policy, which executes the leakage removal protocol every syndrome extraction round; ERASER and ERASER+M, which schedule DQLR speculatively; and Optimal, which schedules DQLR whenever there is a data qubit leakage.

Figure 20 shows the LER for all four policies. We observe that ERASER improves upon the baseline DQLR protocol by 1.8× on average and up to 1.9×, whereas ERASER+M improves 2× on average and up to 2.6×. We note that there is about a 4.4× gap between the optimal scheduling of DQLR and the baseline DQLR protocol. These results demonstrate that custom approaches can benefit significantly from real-time scheduling.

We further examine the LPR of all four policies. Figure 21 shows the LPR for all four policies for $d = 11$ over 110 syndrome extraction rounds. Unlike SWAP-based LRCs, DQLR stabilizes the LPR rather quickly, as was reported in prior work [53]. However, as DQLR can cause leakage if the first reset fails, overusing DQLR can cause additional leakage. As ERASER and ERASER+M judiciously schedule DQLR, they reduce the LPR by about 1.4× and 1.5× respectively.

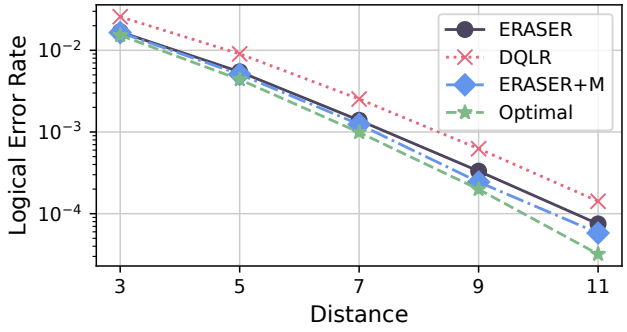

Figure 20: LER over 10 QEC cycles at $p = 10^{-3}$ using DQLR instead of SWAPs.

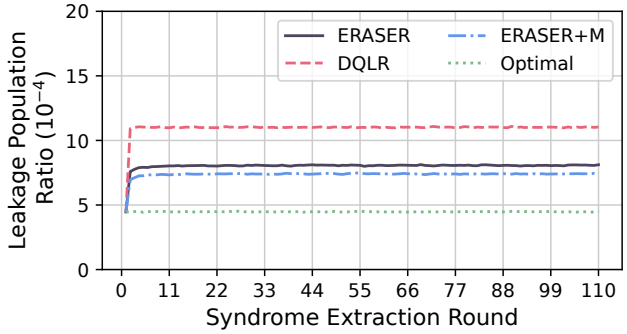

Figure 21: LPR over 110 rounds for $d = 11$ at $p = 10^{-3}$ using DQLR instead of SWAPs.

B APPENDIX: ARTIFACT

B.1 Abstract

The artifact contains the source code used to evaluate the designs proposed in this paper. We have listed how to reproduce the key results of our paper, namely those presented in Figures 5 and 6, which motivate the problem of inefficient LRC scheduling; Figures 14-16, which are our main results; and Table 2, which lists the utilization and timing results for our design (in RTL).

[6]This may occur as LeakageISWAP performs an *i*SWAP in the $|11\rangle, |20\rangle$ basis.

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

- **Algorithm:** ERASER, a leakage-detection algorithm.
- **Program:** `leakage`, `eraser_rtl_gen`
- **Compilation:** GCC
- **Hardware:** Tested on both Linux and MacOS
- **Execution:** Through command line
- **Metrics:** Logical Error Rate (LER) and Leakage Population Ratio (LPR)
- **Output:** RTL, data files, and figures.
- **Experiments:** Only three experiments (detailed later).
- **How much disk space required (approximately)?:** At most 1 GB.
- **How much time is needed to prepare workflow (approximately)?:** A minute or so of compilation.
- **How much time is needed to complete experiments (approximately)?:** 48 hours
- **Publicly available?:** Yes, on Zenodo
- **Code licenses (if publicly available)?:** Apache
- **Data licenses (if publicly available)?:** N/A
- **Workflow framework used?:** N/A
- **Archived (provide DOI)?:** 10.5281/zenodo.8224450

B.3 Description

B.3.1 How to access. The artifact for this work is available on Zenodo at https://doi.org/10.5281/zenodo.8224450.

B.3.2 Software dependencies. The code is built using CMake v3.20.3, though slightly older versions should be fine and can be enabled by modifying `CMakeLists.txt`. The compiler used in our evaluations was g++-12 and g++-13, and we also used OpenMPI v4.x.x to parallelize the experiments on computing clusters. All other dependencies have been packaged with the code and are referenced through CMake.

The provided plotting script has been tested using Python v3.10.6, and the following packages are dependencies: matplotlib v3.6.1, numpy v1.23.4, scipy v1.9.2, though it should work fine with newer versions.

For data involving RTL, we used Vivado 2023.1 to synthesize the design and obtain utilization and timing data, but older Vivado versions (i.e. 2022.x) should be sufficient.

B.4 Installation

We encourage using two build directories: `build` and `build_RTL` to avoid any issues. `build` is for creating the data for Figures 5, 6, 14, 15, and 16, whereas `build_RTL` is for generating the RTL (default distance 9). The executables `leakage` and `eraser_rtl_gen` can be generated as follows:

```
$ cd build
$ cmake .. -DCMAKE_BUILD_TYPE=Release
$ make -j8
$ cd ../build_RTL
$ cmake .. -DRTL=On -DCMAKE_BUILD_TYPE=Release
$ make -j8
```

B.5 Experiment workflow

B.5.1 Main Paper Figures. We explain how to generate the data for Figures 5, 6, 14, 15, and 16, which represent the main insights and results of the work. We have provided several bash scripts in the `leakage` folder: `figure_5_6.sh` and `figure_14_15_16.sh` which generate the data for the corresponding figures. We recommend running `figure_5_6.sh` first as it can be done on any laptop within an hour. For `figure_14_15_16.sh`, we recommend using a cluster with sufficient memory, as the larger distance codes require significant amounts of memory and may need many cores to complete in time. For reference, our evaluations for Figures 5 and 6 took five minutes on an ARM server using 64 cores using about 1GB per core. In contrast, our evaluations for Figures 14 through 16 took two days running on a cluster with 512 cores, with about 8GB per core.

For `figure_5_6.sh`, there are only two parameters: `proc`, the number of processors (used by MPI), and `shots`, which is the number of trials to use in the experiment. The number of processors can be set to the user's preference. For `shots`, we used 100K in the paper, though 10K would be fine also.

For `figure_14_15_16.sh`, there are three parameters: `p`, the physical error rate; `proc`, the number of processors; and `shots`, the number of trials in the experiment. In the paper, Figure 14 uses both $p = 10^{-3}$ and $p = 10^{-4}$, whereas Figure 15 and 16 both use $p = 10^{-3}$. We also note that the number of trials to obtain meaningful data *increases* with code distance (d) and lower physical error rate. We found that 10M trials (`shots`) is sufficient for all experiments at $p = 10^{-3}$, whereas 100M trials will provide the data reported for $p = 10^{-4}$ in the paper. We note that $d = 9$ and $d = 11$ will be incomplete as they require more trials, likely 1B or beyond which is intractable to perform with our setup.

In summary, to generate the data:

```
$ cd leakage
$ ./figure_5_6.sh <PROC> 100000
$ ./figure_14_15_16.sh 1e-3 <PROC> 10000000
$ ./figure_14_15_16.sh 1e-4 <PROC> 100000000
```

To plot the data, go to the `python` folder and call `plot.py`. This will generate PDFs for each figure in the `figures` folder.

B.5.2 RTL Statistics. To generate the RTL (which is in SystemVerilog), run:

```
$ cd build_RTL
$ ./eraser_rtl_gen <DISTANCE> > <RTL-FILE>
```

For example, `./eraser_rtl_gen 9 > eraser_d9.sv` will write the RTL for a distance 9 code to the `eraser_d9.sv` file. After obtaining the RTL for distances 3 to 11, make a project in Vivado with the source file as the sole file. Then, add a constraint file (`.xdc`) to drive the `clk` signal for the RTL. Our file contained the single line[7]:

```
create_clock -name clk -period <PERIOD> -waveform {0 <PERIOD/2>}
[get_ports clk]
```

where `PERIOD` (which is in nanoseconds) can be set to any value based on the desired frequency (i.e. for a frequency of 250MHz, set `PERIOD = 4`). Our designs generally have low critical path latencies,

[7] See here for more details on adding constraint files in Vivado.

so high frequencies can be used. However, we believe a practical frequency would be 500MHz (so `PERIOD = 2`). After defining the constraint file and adding it to the project, run *Synthesis* on the project. Utilization and timing results can be obtained at this point by generating different reports for the project.

Our design does not require any IP blocks. We used the Kintex UltraScale+ FPGA to evaluate our design, and the specific part used for evaluations in the paper is `xcku3p-ffvd900-3-e`.

To obtain RTL results for each distance, we recommend having one project that contains the RTLs for each distance. Then, obtaining results for each distance can be done by disabling the source files for other distances in Vivado.

We note that for $d = 11$, our design requires more IO pins than available on the device. IO saturation is more an evaluation detail than a design detail. Rather, it is more practical that ERASER operates as a logic block in the larger fabric. Consequently, ERASER will likely not be interacting with external inputs. While the design will fail to synthesize for $d = 11$, timing and utilization data can still be retrieved from the design. If the user wants to avoid errors during synthesis, one can add `-mode out_of_context` to the project constraints file.

B.6 Evaluation and expected results

The results for LER and LPR should be about the same as the reported values in the paper, perhaps with slight deviations due to randomness. The results for RTL utilization and timing should be similar to that which was reported in Table 3, with some deviation due to randomness.

B.7 Experiment customization

To modify ERASER and ERASER+M, see `quarch/src/fleece.cpp`. To modify the experiments, see `leakage/src/experiments.cpp`.

B.8 Methodology

Submission, reviewing and badging methodology:

- https://www.acm.org/publications/policies/artifact-review-and-badging-current
- http://cTuning.org/ae/submission-20201122.html
- http://cTuning.org/ae/reviewing-20201122.html

REFERENCES

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

[2] Google Quantum AI. Accessed: June 19, 2021. Quantum Computer Datasheet. https://quantumai.google/hardware/datasheet/weber.pdf.

[3] Panos Aliferis and Barbara M Terhal. 2005. Fault-tolerant quantum computation for local leakage faults. *arXiv preprint quant-ph/0511065* (2005).

[4] Lucas Berent, Lukas Burgholzer, and Robert Wille. 2022. Software Tools for Decoding Quantum Low-Density Parity Check Codes. https://doi.org/10.48550/ARXIV.2209.01180

[5] Natalie C. Brown and Kenneth R. Brown. 2019. Leakage mitigation for quantum error correction using a mixed qubit scheme. *Phys. Rev. A* 100 (Sep 2019), 032325. Issue 3. https://doi.org/10.1103/PhysRevA.100.032325

[6] Natalie C. Brown, Andrew Cross, and Kenneth R. Brown. 2020. Critical faults of leakage errors on the surface code. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. 286–294. https://doi.org/10.1109/QCE49297.2020.00043

[7] Natalie C Brown, Michael Newman, and Kenneth R Brown. 2019. Handling leakage with subsystem codes. *New Journal of Physics* 21, 7 (jul 2019), 073055. https://doi.org/10.1088/1367-2630/ab3372

[8] CC Bultink, TE O'Brien, R Vollmer, N Muthusubramanian, MW Beekman, MA Rol, X Fu, B Tarasinski, V Ostroukh, B Varbanov, et al. 2020. Protecting quantum entanglement from leakage and qubit errors via repetitive parity measurements. *Science advances* 6, 12 (2020), eaay3050.

[9] Ilkwon Byun, Junpyo Kim, Dongmoon Min, Ikki Nagaoka, Kosuke Fukumitsu, Iori Ishikawa, Teruo Tanimoto, Masamitsu Tanaka, Koji Inoue, and Jangwoo Kim. 2022. XQsim: Modeling Cross-Technology Control Processors for 10+K Qubit Quantum Computers. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (New York, New York) *(ISCA '22)*. Association for Computing Machinery, New York, NY, USA, 366–382. https://doi.org/10.1145/3470496.3527417

[10] Earl Campbell, Ankur Khurana, and Ashley Montanaro. 2019. Applying quantum algorithms to constraint satisfaction problems. *Quantum* 3 (jul 2019), 167. https://doi.org/10.22331/q-2019-07-18-167

[11] Christopher Chamberland, Luis Goncalves, Prasahnt Sivarajah, Eric Peterson, and Sebastian Grimberg. 2022. Techniques for combining fast local decoders with global decoders under circuit-level noise. https://doi.org/10.48550/ARXIV.2208.01178

[12] Liangyu Chen, Hang-Xi Li, Yong Lu, Christopher W Warren, Christian J Križan, Sandoko Kosen, Marcus Rommel, Shahnawaz Ahmed, Amr Osman, Janka Biznárová, et al. 2023. Transmon qubit readout fidelity at the threshold for quantum error correction without a quantum-limited amplifier. *npj Quantum Information* 9, 1 (2023), 26.

[13] Andrew M. Childs, Dmitri Maslov, Yunseong Nam, Neil J. Ross, and Yuan Su. 2018. Toward the first quantum simulation with quantum speedup. *Proceedings of the National Academy of Sciences* 115, 38 (sep 2018), 9456–9461. https://doi.org/10.1073/pnas.1801723115

[14] Jacob P Covey, Harald Weinfurter, and Hannes Bernien. 2023. Quantum networks with neutral atom processing nodes. *npj Quantum Information* 9, 1 (2023), 90.

[15] Poulami Das, Eric Kessler, and Yunong Shi. 2023. The Imitation Game: Leveraging CopyCats for Robust Native Gate Selection in NISQ Programs. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 787–801.

[16] Poulami Das, Aditya Locharla, and Cody Jones. 2022. LILLIPUT: A Lightweight Low-Latency Lookup-Table Decoder for near-Term Quantum Error Correction. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (Lausanne, Switzerland) *(ASPLOS '22)*. Association for Computing Machinery, New York, NY, USA, 541–553. https://doi.org/10.1145/3503222.3507707

[17] Poulami Das, Christopher A. Pattison, Srilatha Manne, Douglas M. Carmean, Krysta M. Svore, Moinuddin Qureshi, and Nicolas Delfosse. 2022. AFS: Accurate, Fast, and Scalable Error-Decoding for Fault-Tolerant Quantum Computers. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 259–273. https://doi.org/10.1109/HPCA53966.2022.00027

[18] Nicolas Delfosse. 2020. Hierarchical decoding to reduce hardware requirements for quantum computing. https://doi.org/10.48550/ARXIV.2001.11427

[19] Nicolas Delfosse, Vivien Londe, and Michael E. Beverland. 2022. Toward a Union-Find decoder for quantum LDPC codes. *IEEE Transactions on Information Theory* (2022), 1–1. https://doi.org/10.1109/TIT.2022.3143452

[20] Nicolas Delfosse and Naomi H Nickerson. 2017. Almost-linear time decoding algorithm for topological codes. *arXiv preprint arXiv:1709.06218* (2017).

[21] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. 2002. Topological quantum memory. *J. Math. Phys.* 43, 9 (Sep 2002), 4452–4505. https://doi.org/10.1063/1.1499754

[22] Jack Edmonds. 1965. Paths, trees, and flowers. *Canadian Journal of mathematics* 17 (1965), 449–467.

[23] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028* (2014).

[24] Austin G. Fowler. 2013. Coping with qubit leakage in topological codes. *Phys. Rev. A* 88 (Oct 2013), 042308. Issue 4. https://doi.org/10.1103/PhysRevA.88.042308

[25] 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.

[26] Jay Gambetta. 2022. Quantum-centric supercomputing: The Next Wave of computing. https://research.ibm.com/blog/next-wave-quantum-centric-supercomputing

[27] Craig Gidney. 2021. Stim: a fast stabilizer circuit simulator. *Quantum* 5 (July 2021), 497. https://doi.org/10.22331/q-2021-07-06-497

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

[29] Craig Gidney, Michael Newman, Austin Fowler, and Michael Broughton. 2021. A Fault-Tolerant Honeycomb Memory. *Quantum* 5 (Dec. 2021), 605. https://doi.org/10.22331/q-2021-12-20-605

[30] Craig Gidney, Michael Newman, and Matt McEwen. 2022. Benchmarking the Planar Honeycomb Code. *Quantum* 6 (Sept. 2022), 813. https://doi.org/10.22331/q-2022-09-21-813

[31] TM Graham, Y Song, J Scott, C Poole, L Phuttitarn, K Jooya, P Eichler, X Jiang, A Marra, B Grinkemeyer, et al. 2022. Multi-qubit entanglement and algorithms on a neutral-atom quantum computer. *Nature* 604, 7906 (2022), 457–462.

[32] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. 2009. Quantum Algorithm for Linear Systems of Equations. *Physical Review Letters* 103, 15 (oct 2009). https://doi.org/10.1103/physrevlett.103.150502

[33] Oscar Higgott. 2021. PyMatching: A Python package for decoding quantum codes with minimum-weight perfect matching. arXiv:2105.13082 [quant-ph]

[34] Adam Holmes, Mohammad Reza Jokar, Ghasem Pasandi, Yongshan Ding, Massoud Pedram, and Frederic T. Chong. 2020. NISQ+: Boosting quantum computing power by approximating quantum error correction. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. 556–569. https://doi.org/10.1109/ISCA45697.2020.00053

[35] 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.

[36] IBM. 2021. IBM Quantum breaks the 100-qubit processor barrier. https://research.ibm.com/blog/127-qubit-quantum-processor-eagle.

[37] 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 (mar 2021), 025020. https://doi.org/10.1088/2058-9565/abe519

[38] A Yu Kitaev. 1997. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys* 52, 6 (dec 1997), 1191. https://doi.org/10.1070/RM1997v052n06ABEH002155

[39] A Yu Kitaev. 2003. Fault-tolerant quantum computation by anyons. *Annals of Physics* 303, 1 (2003), 2–30.

[40] Ian D. Kivlichan, Craig Gidney, Dominic W. Berry, Nathan Wiebe, Jarrod McClean, Wei Sun, Zhang Jiang, Nicholas Rubin, Austin Fowler, Alá n Aspuru-Guzik, Hartmut Neven, and Ryan Babbush. 2020. Improved Fault-Tolerant Quantum Simulation of Condensed-Phase Correlated Electrons via Trotterization. *Quantum* 4 (jul 2020), 296. https://doi.org/10.22331/q-2020-07-16-296

[41] Sebastian Krinner, Nathan Lacroix, Ants Remm, Agustin Di Paolo, Elie Genois, Catherine Leroux, Christoph Hellings, Stefania Lazar, Francois Swiadek, Johannes Herrmann, et al. 2022. Realizing repeated quantum error correction in a distance-three surface code. *Nature* 605, 7911 (2022), 669–674.

[42] Argonne National Laboratory. 2018. INTRODUCTION TO QUANTUM ERROR CORRECTION. https://cpb-us-w2.wpmucdn.com/voices.uchicago.edu/dist/0/2327/files/2019/11/QECIntro.pdf.

[43] Nathan Lacroix, Luca Hofele, Ants Remm, Othmane Benhayoune-Khadraoui, Alexander McDonald, Ross Shillito, Stefania Lazar, Christoph Hellings, Francois Swiadek, Dante Colao-Zanuz, et al. 2023. Fast Flux-Activated Leakage Reduction for Superconducting Quantum Circuits. *arXiv preprint arXiv:2309.07060* (2023).

[44] 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, 3 (jul 2021). https://doi.org/10.1103/prxquantum.2.030305

[45] Jessica Lemieux, Guillaume Duclos-Cianci, David Sé néchal, and David Poulin. 2021. Resource estimate for quantum many-body ground-state preparation on a quantum computer. *Physical Review A* 103, 5 (may 2021). https://doi.org/10.1103/physreva.103.052408

[46] Wang Liao, Yasunari Suzuki, Teruo Tanimoto, Yosuke Ueno, and Yuuki Tokunaga. 2023. WIT-Greedy: Hardware System Design of Weighted ITerative Greedy Decoder for Surface Code. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference* (Tokyo, Japan) *(ASPDAC '23)*. Association for Computing Machinery, New York, NY, USA, 209–215. https://doi.org/10.1145/3566097.3567933

[47] Andrii Maksymov, Jason Nguyen, Vandiver Chaplin, Yunseong Nam, and Igor L Markov. 2022. Detecting Qubit-coupling faults in ion-trap quantum computers. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 387–399.

[48] Hidetaka Manabe, Yasunari Suzuki, and Andrew S Darmawan. 2023. Efficient Simulation of Leakage Errors in Quantum Error Correcting Codes Using Tensor Network Methods. *arXiv preprint arXiv:2308.08186* (2023).

[49] JF Marques, H Ali, BM Varbanov, M Finkel, HM Veen, SLM van der Meer, S Valles-Sanclemente, N Muthusubramanian, M Beekman, N Haider, et al. 2023. All-microwave leakage reduction units for quantum error correction with superconducting transmon qubits. *Physical Review Letters* 130, 25 (2023), 250602.

[50] Satvik Maurya, Chaithanya Naik Mude, William D. Oliver, Benjamin Lienhard, and Swamit Tannu. 2022. Hardware Efficient Neural Network Assisted Qubit Readout. arXiv:2212.03895 [quant-ph]

[51] Matt McEwen, Dave Bacon, and Craig Gidney. 2023. Relaxing hardware requirements for surface code circuits using time-dynamics. *arXiv preprint arXiv:2302.02192* (2023).

[52] Matt McEwen, Dvir Kafri, Z Chen, Juan Atalaya, KJ Satzinger, Chris Quintana, Paul Victor Klimov, Daniel Sank, C Gidney, AG Fowler, et al. 2021. Removing leakage-induced correlated errors in superconducting quantum error correction. *Nature communications* 12, 1 (2021), 1761.

[53] Kevin C Miao, Matt McEwen, Juan Atalaya, Dvir Kafri, Leonid P Pryadko, Andreas Bengtsson, Alex Opremcak, Kevin J Satzinger, Zijun Chen, Paul V Klimov, et al. 2022. Overcoming leakage in scalable quantum error correction. *arXiv preprint arXiv:2211.04728* (2022).

[54] Michael A. Nielsen and Isaac L. Chuang. 2000. *Quantum Computation and Quantum Information*. Cambridge University Press.

[55] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. 2014. A variational eigenvalue solver on a photonic quantum processor. *Nature communications* 5 (2014), 4213.

[56] Gokul Subramanian Ravi, Jonathan M. Baker, Arash Fayyazi, Sophia Fuhui Lin, Ali Javadi-Abhari, Massoud Pedram, and Frederic T. Chong. 2022. Have your QEC and Bandwidth too!: A lightweight cryogenic decoder for common / trivial errors, and efficient bandwidth + execution management otherwise. https://doi.org/10.48550/ARXIV.2208.08547

[57] Markus Reiher, Nathan Wiebe, Krysta M. Svore, Dave Wecker, and Matthias Troyer. 2017. Elucidating reaction mechanisms on quantum computers. *Proceedings of the National Academy of Sciences* 114, 29 (2017), 7555–7560. https://doi.org/10.1073/pnas.1619152114 arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.1619152114

[58] Peter W Shor. 1999. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review* (1999).

[59] Robert S Smith, Eric C Peterson, Mark G Skilbeck, and Erik J Davis. 2020. An open-source, industrial-strength optimizing compiler for quantum programs. *Quantum Science and Technology* 5, 4 (2020), 044001.

[60] Samuel C Smith, Benjamin J Brown, and Stephen D Bartlett. 2022. A local predecoder to reduce the bandwidth and latency of quantum error correction. *arXiv preprint arXiv:2208.04660* (2022).

[61] Matthias Steffen, Jerry Chow, Sarah Sheldon, and Doug McClure. 2022. IBM Quantum's highest performant system, yet. https://research.ibm.com/blog/eagle-quantum-error-mitigation

[62] Roman Stricker, Davide Vodola, Alexander Erhard, Lukas Postler, Michael Meth, Martin Ringbauer, Philipp Schindler, Thomas Monz, Markus Müller, and Rainer Blatt. 2020. Experimental deterministic correction of qubit loss. *Nature* 585, 7824 (2020), 207–210.

[63] Martin Suchara, Andrew W. Cross, and Jay M. Gambetta. 2015. Leakage suppression in the toric code. In *2015 IEEE International Symposium on Information Theory (ISIT)*. 1119–1123. https://doi.org/10.1109/ISIT.2015.7282629

[64] Neereja Sundaresan, Theodore J. Yoder, Youngseok Kim, Muyuan Li, Edward H. Chen, Grace Harper, Ted Thorbeck, Andrew W. Cross, Antonio D. Córcoles, and Maika Takita. 2022. Matching and maximum likelihood decoding of a multi-round subsystem quantum error correction experiment. https://doi.org/10.48550/ARXIV.2203.07205

[65] Yu Tomita and Krysta M. Svore. 2014. Low-distance surface codes under realistic quantum noise. *Physical Review A* 90, 6 (Dec 2014). https://doi.org/10.1103/physreva.90.062320

[66] Yosuke Ueno, Masaaki Kondo, Masamitsu Tanaka, Yasunari Suzuki, and Yutaka Tabuchi. 2021. QECOOL: On-Line Quantum Error Correction with a Superconducting Decoder for Surface Code. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. 451–456. https://doi.org/10.1109/DAC18074.2021.9586326

[67] Yosuke Ueno, Masaaki Kondo, Masamitsu Tanaka, Yasunari Suzuki, and Yutaka Tabuchi. 2022. NEO-QEC: Neural Network Enhanced Online Superconducting Decoder for Surface Codes. https://doi.org/10.48550/ARXIV.2208.05758

[68] Yosuke Ueno, Masaaki Kondo, Masamitsu Tanaka, Yasunari Suzuki, and Yutaka Tabuchi. 2022. QULATIS: A Quantum Error Correction Methodology toward Lattice Surgery. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 274–287. https://doi.org/10.1109/HPCA53966.2022.00028

[69] Boris Mihailov Varbanov, Francesco Battistel, Brian Michael Tarasinski, Viacheslav Petrovych Ostroukh, Thomas Eugene O'Brien, Leonardo DiCarlo, and Barbara Maria Terhal. 2020. Leakage detection for a transmon-based surface code. *npj Quantum Information* 6, 1 (2020), 102.

[70] Suhas Vittal, Poulami Das, and Moinuddin Qureshi. 2023. Astrea: Accurate Quantum Error-Decoding via Practical Minimum-Weight Perfect-Matching. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*. 1–16.

[71] Anbang Wu, Gushu Li, Hezi Zhang, Gian Giacomo Guerreschi, Yufei Ding, and Yuan Xie. 2022. A Synthesis Framework for Stitching Surface Code with Superconducting Quantum Devices. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (New York, New York) *(ISCA '22)*. Association for Computing Machinery, New York, NY, USA, 337–350. https://doi.org/10.1145/3470496.3527381

[72] Yue Wu, Shimon Kolkowitz, Shruti Puri, and Jeff D Thompson. 2022. Erasure conversion for fault-tolerant quantum computing in alkaline earth Rydberg atom arrays. *Nature communications* 13, 1 (2022), 4657.