2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA) | 978-1-6654-2027-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/HPCA53966.2022.00027

AFS: Accurate, Fast, and Scalable Error-Decoding for Fault-Tolerant Quantum Computers

Poulami Das*§, Christopher A. Pattison†, Srilatha Manne‡, Douglas M. Carmean‡,
Krysta M. Svore‡, Moinuddin Qureshi*, and Nicolas Delfosse‡
*Georgia Institute of Technology, Atlanta, Georgia
†California Institute of Technology, Pasadena, California
‡Microsoft Quantum and Microsoft Research, Redmond, Washington
§poulami@gatech.edu

***Abstract*—Quantum computers promise computational advantages for many important problems across various application domains. Unfortunately, physical quantum devices are highly susceptible to errors that limit us from running most of these quantum applications. Quantum Error Correction (QEC) codes are required to implement *Fault-Tolerant Quantum Computers (FTQC)* on which computations can be performed without encountering errors. Error decoding is a critical component of quantum error correction and is responsible for transforming a set of qubit measurements generated by the QEC code, called the syndrome, into error locations and error types. For the feasibility of implementation, error decoders must not only identify errors with high accuracy, but also be fast and scalable to a large number of qubits. Unfortunately, most of the prior works on error decoding have focused primarily only on the accuracy and have relied on software implementations that are too slow to be of practical use. Furthermore, these studies only look at designing a single decoder and do not analyze the challenges involved in scaling the storage and bandwidth requirements when performing error correction in large systems with thousands of qubits.**

In this paper, we present *AFS*, an accurate, fast, and scalable decoder architecture that is designed to operate in the context of systems with hundreds of logical qubits. We present the hardware implementation of AFS, which is based on the Union Find decoding algorithm and employs a three-stage pipelined design. AFS provides orders of magnitude higher accuracy compared to recent SFQ-based hardware decoders (logical error rate of 6×10^{-10} for physical error rate of 10^{-3}) and low decoding latency (42ns on average), while being robust to measurement errors introduced while extracting syndromes during the QEC cycles. We also reduce the amount of decoding hardware required to perform QEC simultaneously on all the logical qubits by co-designing the micro-architecture across multiple decoding units. Our proposed *Conjoined-Decoder Architecture (CDA)* reduces the storage overhead by 70% (10MB to 2.8MB). Finally, we reduce the bandwidth overheads required to transmit syndromes from the qubits to the decoders by exploiting the sparsity in the syndromes and compressing the data. Our proposed *Syndrome Compression* reduces the bandwidth requirement by 30x, on an average.

***Keywords*-Quantum computing, Quantum error correction, Fault-tolerant quantum computing, Decoding, Union-Find decoding, Surface codes**

I. Introduction

Quantum computing promises significant speed-up over conventional computers for many important applications [71, 90, 101, 121]. The primary obstacle to practical quantum computing is the high error-rate in quantum devices. A *Fault-Tolerant Quantum Computer (FTQC)*, in which quantum bits, or qubits, are regularly refreshed by Quantum Error Correction (QEC), is necessary to perform useful computations on error-prone quantum hardware.

QEC is way more challenging than classical error correction because the error rates of qubits are orders of magnitude higher and their inherent properties impose many fundamental restrictions. For example, qubits cannot be copied [123] and lose their quantum state when measured. To tackle these challenges, QEC codes [1, 3, 46, 55, 100] encode *logical* qubits by using a set of *data* and *ancilla* qubits, as shown in Figure 1(a). The data qubits represent the quantum information, whereas the ancilla qubits periodically interact or entangle with the data qubits as per the QEC protocol and reveal information about errors on the data qubits upon measurement. This process is called syndrome measurement or extraction and the output of ancilla measurements is called a *syndrome*. An *error decoder* processes the syndrome to locate and identify the type of errors on the data qubits.

To be practical for implementation in an FTQC, a decoder must satisfy three design constraints: accuracy, latency, and scalability. The ***accuracy constraint*** suggests that the decoder must correctly identify all the errors (both in the data qubits as well as measurement operations in the QEC cycles) with a very high probability. The ***latency constraint*** requires decoders to correct the errors within an error correction cycle to prevent any backlog or accumulation of errors. The ***scalability constraint*** indicates that it must be feasible to implement the decoder in FTQCs with hundreds of logical qubits. To scale to such large FTQCs and simultaneously facilitate operation inside a constrained, potentially cryogenic [8], environment, decoders must be hardware efficient. The proximity to the physical qubits necessitates the use of minimal hardware to ensure that the thermal heat generated

978-1-6654-2027-3/22/$31.00 ©2022 IEEE
DOI 10.1109/HPCA53966.2022.00027

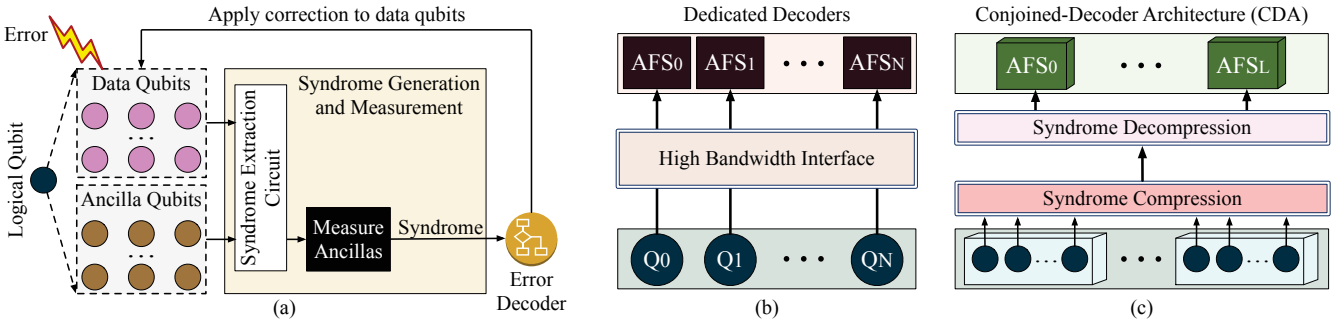

Figure 1. (a) The framework for quantum error correction and the process of error decoding (b) Existing designs use dedicated decoders and do not consider bandwidth constraints (c) Proposed optimizations of Conjoined-Decoder and Syndrome Compression to reduce the storage and bandwidth overheads.

from the decoding circuits do not introduce additional noise channels for the qubits. In this paper, we focus on the hardware design of decoders for quantum error correction.

The design of decoders for QEC has been an active area of research for the last two decades. However, these studies [4–7, 10, 12–17, 19, 20, 23, 29, 32–36, 38–40, 42, 44, 45, 47–49, 56–58, 61, 62, 65, 66, 70, 73, 82, 83, 97, 102, 104, 110–112, 114–116, 119, 120, 124, 125] have mainly focused on the accuracy constraint, whereas the latency constraint is typically studied only at an algorithmic level, by mainly relying on software implementations, and it is unclear if existing decoders can satisfy this constraint without degrading the accuracy [43]. More recently, Holmes et al. [59] proposed a hardware decoder using superconducting technology that provides high accuracy and low latency. Unfortunately, the proposed decoder design assumes perfect ancilla measurements. However, measurement operations are erroneous and may generate incorrect syndromes even if there are no errors in the data qubits. Failure to account for imperfect ancilla measurements and relying on erroneous syndromes causes miscorrections and severely degrades the accuracy of a decoder. Ideally, we want an error decoder that can tolerate errors in the data qubits as well as in the measurement of ancilla qubits (to mitigate measurement errors, the decoder must process multiple rounds of syndrome measurements [36], presenting even greater challenges to meet the latency constraint). Finally, all existing decoders study the decoder for a single logical qubit and implicitly assume dedicated decoders for each logical qubit, as shown in Figure 1(b). However, this architecture is not scalable as the hardware complexity of the decoders grows linearly with the number of logical qubits. Alternately, sharing decoders between multiple logical qubits reduces the hardware cost but can also impact the accuracy because any resource conflicts may deprive a logical qubit timely access to decoding circuits, increasing the probability of an error going undetected. Ideally, we want to co-design these decoders such that we can reduce the hardware while simultaneously maintaining the accuracy and latency.

To that end, this paper proposes the *AFS (Accurate, Fast, and Scalable) Decoder*, that is designed to provide high decoding accuracy, low latency (average latency of 42ns), and is optimized to operate with a system containing hundreds of logical qubits. We assume *Surface Code* [36, 46, 64, 89] which is widely considered to be one of the most promising candidates for QEC. AFS is based on the *Union Find Decoding (UFD)* algorithm [32, 34] as the accuracy, simplicity, and low time-complexity of the UFD makes it an ideal candidate for our design. We describe the implementation of the AFS microarchitecture, where the design comprises of three pipeline stages, based on the three key steps of the decoding algorithm: Graph Generator (Gr-Gen), Depth First Search (DFS), and Correction Engine (CORR).[1] Furthermore, we ensure that AFS is tolerant of measurement errors, by repeating the measurement operations d times (where d is the code distance and is the length of the shortest error chain that cannot be corrected). To analyze the effectiveness of the AFS decoder, we assume a phenomenological noise model which accounts for erroneous ancilla measurements in addition to errors on data qubits [36]. We observe that AFS provides orders of magnitude higher accuracy compared to recent SFQ-based decoders [59, 113] (*e.g.* a logical error rate of error rate of 6×10^{-10} for physical error rate of 10^{-3} at d=11). Furthermore, the average decoding latency of AFS is 42 nanoseconds and the 99.9 percentile decoding latency is 150 nanoseconds, both of which are below the syndrome generation cycle time of 400 ns [51, 59]. AFS requires a storage of 8.95 KB (for d=11) to 133 KB (for d=25).

Allocating individual AFS decoders to each qubit may require hardware resources and bandwidth in proportion to the number of qubits in large systems. We avoid this rapid increase in hardware cost by observing that not all parts of the decoder are used equally. For infrequently used components of the decoder, we can share the component across

[1]Our hardware implementation makes minor modification to the UFD algorithm, as our objective is to reduce the cost of hardware required for implementing the decoder and not establishing the theoretical bounds on the asymptotic complexity of the algorithm.

multiple logical qubits and reduce the overall cost. With this insight, we propose *Conjoined-Decoder Architecture (CDA)*, which reduces the storage overhead by 3.5x. CDA reduces the total storage required to implement error decoding for a reasonably large fault-tolerant system with 1000 logical qubits from 9.96 MB to 2.8 MB. As CDA shares decoding resources, a decoding failure may also result from the lack of access to decoding circuits, which we refer to as a *timeout failure* in this paper. Our analysis shows that for a period of 350 ns, CDA causes timeout failures with probability 2×10^{-11}, which is much smaller than the logical error rate 6×10^{-10}, thus causing negligible impact on accuracy.

Existing decoders assume availability of extremely large bandwidth, typically several hundreds of Gbps, for syndrome transmission. Syndrome data must be sent from millions of physical qubits to the decoders as quickly as possible so that the decoding completes within an error correction cycle. Since decoding is a time-sensitive problem, transmitting syndrome data at lower bandwidth reduces the time available for decoding and may result in errors going undetected. To address this challenge, we exploit the sparsity in syndromes (most syndrome bits tend to be zeros) and propose *Syndrome Compression*, which reduces the required bandwidth (typically between 200-2000 Gbps) by 30x, on average. An overview of the overall design is shown in Figure 1(c).

Overall, this paper makes the following contributions:

1) We propose AFS, a hardware-based error decoder for QEC, which provides high accuracy and low latency (average latency of 42 nanoseconds), while being tolerant to measurement errors.
2) We propose CDA, which avoids the rapid linear increase in decoding hardware cost by co-designing the decoder microarchitecture across multiple qubits (storage reduction of 3.5x).
3) We propose Syndrome Compression, which exploits the sparsity in syndrome values to reduce the bandwidth required to transmit the syndromes from qubits to the decoders by 30x.

II. Background and Motivation

A. Basics of qubits and types of errors

A qubit may be described by a vector $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$, representing a superposition of its basis states $|0\rangle$ and $|1\rangle$ with $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$. Qubits accumulate tiny rotations through their interactions with environmental noise. These unwarranted rotations can be projected onto three types of errors X, Y, and Z, called Pauli errors. The bit-flip (X) error swaps the probability amplitudes of the basis states $|0\rangle$ and $|1\rangle$ and maps the qubit $|\psi\rangle$ into $\beta|0\rangle + \alpha|1\rangle$. The phase-flip (Z) error introduces a relative phase between the two basis states, changing the qubit $|\psi\rangle$ to $\alpha|0\rangle - \beta|1\rangle$. The Y error corresponds to a simultaneous bit-flip and phase-flip error.

B. QEC and Surface Code

Quantum Error Correction (QEC) protects quantum states by encoding k logical qubits into a block of n data qubits ($n > k$). Additional ancilla qubits are used to obtain information about errors on the data qubits. QEC is a two-step process. First, a measurement circuit is executed on a block of qubits to produce a *syndrome*. In the second step, a *decoder* is used to identify the location and type of errors on data qubits based on the syndrome information so that errors can then be corrected. If the error rate of the physical qubits is lower than a threshold, QEC produces k logical qubits with lower error rate than the physical qubits at the expense of an increased number of physical qubits per logical qubit.

Surface code [36, 47, 64, 89] encodes a logical qubit into a square grid of $(2d-1) * (2d-1)$ alternating data and ancilla qubits, where d is the code distance that determines the length of the shortest uncorrectable error chain. Larger distance results in greater error tolerance but also increases the number of physical qubits consumed. The data qubits store the quantum information, whereas the ancilla qubits entangle with the neighboring data qubits and when measured, reveals information about errors on data qubits. For example, Figure 2(a) shows a distance-3 surface code lattice. An X-error on data qubit D_0 flips the neighboring Z-type ancilla qubits, whereas a Z-error on D_1 flips the adjacent X-type ancillas, as shown in Figure 2(b). When a data qubit encounters a Y-error, such as D_2, it flips both the X-type and Z-type ancilla qubits, as shown in Figure 2(c).

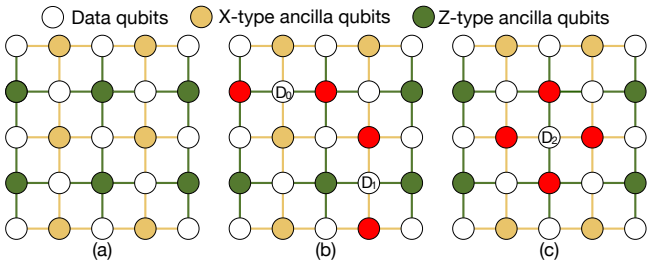

Figure 2. (a) Distance-3 surface code lattice (b) Bit-flip (X) error on D_0 and phase-flip (Z) error on D_1 flips adjacent Z-type and X-type ancilla qubits respectively (c) Both bit-flip and phase-flip (Y) error on D_2 flips both neighboring X-type and Z-type ancilla qubits

C. Error Decoding: Challenges

A decoder uses the output of ancilla measurements, the syndrome, to determine a set of corrections that must be applied to the data qubits. X-type and Z-type errors are corrected independently (then Y-errors are automatically corrected). A decoder must satisfy three constraints: *accuracy*, *latency*, and *scalability* for adoption in large FTQCs. To satisfy the accuracy constraint a decoder must correctly identify the errors with high probability. A decoder satisfies the latency constraint if it can be executed within one round of syndrome extraction (during which a syndrome bit

is extracted from each ancilla qubit), that is about 400ns for superconducting qubits [51, 59]. Failure to achieve the latency constraint causes errors to accumulate and may lead to a backlog problem [109]. For scalability, decoders must be implemented with minimal hardware to operate in constrained environments [8]. Prior software decoders are optimized for accuracy and are often too slow to meet the latency constraint [43]. Moreover, the latency constraint is always at tension with the accuracy constraint. A more accurate decoder is generally slower.

D. Impact of Measurement Errors

A recently proposed hardware-based error decoder [59] meets the latency constraint by leveraging micro-architectural optimizations and superconducting devices but assume perfect ancilla measurements. In reality, quantum measurement operations are error-prone and can significantly deteriorate the performance of the QEC code. Syndrome bits extracted by measuring the ancilla qubits might flip due to measurement errors, leading to incorrect syndrome data and failure to tolerate these errors reduces the logical error rate. Therefore, even small scale QEC experiments account for measurement errors [2]. For example, Figure 3 shows the logical error rate of surface code as a function of the physical error rate p for different code distances assuming (a) perfect measurements and (b) erroneous measurements, where each syndrome bit is also flipped independently with probability p. We observe that the logical error rate increases with the code distance in the presence of measurement errors even though it is expected to decrease exponentially fast with increasing code distance d.

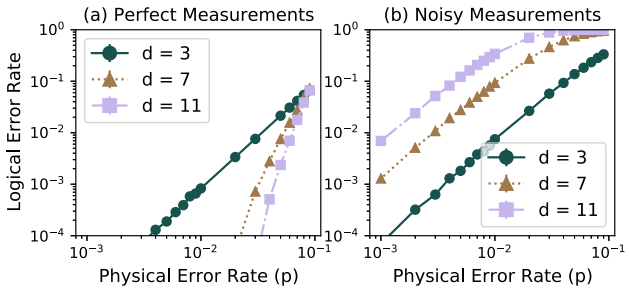

Figure 3. Logical error rate for different surface code distances (d) and physical error rates (p) assuming (a) perfect measurements and (b) noisy measurements estimated by using a Monte-Carlo simulation implementing the Minimum Weight Perfect Matching decoder [36].

Measurement errors can be corrected by performing multiple rounds of syndrome extraction [36]. We assume d consecutive rounds of noisy syndrome measurements are performed [36], and the entire sequence is referred to as a *logical cycle*. To tolerate measurement errors, the decoder must process the syndrome from all these syndrome extraction rounds together to identify the error which occurred. Consequently, the decoding process becomes even more computationally intense and time consuming. Therefore, we need a decoder design that is accurate (can tolerate erroneous ancilla measurements in addition to errors on data qubits) and simultaneously meets the latency constraint.

E. Impact of Scaling to Large FTQCs

Decoders must be accurate, fast, and scalable for implementation in FTQCs required to solve industry-size applications. Current decoders mainly focus on achieving higher accuracy and performance for a single logical qubit and implicitly assume each logical qubit in an FTQC will be assigned its dedicated decoders to maximize the error correction capability.

However, the hardware complexity of this architecture grows linearly with the system size. Alternately, the hardware cost of the decoding architecture can be significantly reduced by sharing decoders between multiple logical qubits. For example, in the extreme case, we may have only one decoder in the entire FTQC to decode all the errors on all the qubits. This design utilizes the least possible hardware, but such arbitrary resource sharing may deprive a logical qubit timely access to the decoding hardware if the decoder is processing a syndrome for another logical qubit. Consequently, such starvation caused by resource conflicts may result in a logical error. Therefore, there exists a trade-off between the error correction capabilities and hardware cost savings that can be achieved in a decoder architecture which shares decoders between logical qubits. Ideally, we want a hardware-efficient decoder architecture without compromising the accuracy.

Moreover, prior studies assume the availability of very high bandwidth, typically hundreds of Gbps, for syndrome transmission. For implementation in large FTQCs, we must not only have a fast and accurate decoder, but also solutions that can scale the decoder to a large number of qubits while minimizing the hardware and bandwidth requirements.

F. Goal

The goal of our paper is to develop decoders that satisfy all three design constraints: accuracy, latency, and scalability. We develop a hardware-based Accurate, Fast, and Scalable decoder, called *AFS*. We also develop *Conjoined-Decoder Architecture (CDA)* and *Syndrome Compression (SC)* to reduce the hardware and bandwidth requirements. We describe the evaluation methodology before discussing the solutions.

III. Evaluation Methodology

In this section, we discuss the simulation infrastructure used to evaluate the accuracy of the proposed decoder design. We defer the methodology for estimating the latency and storage overheads of the decoder design to the later sections of the paper.

A. Monte Carlo Simulation Infrastructure

Figure 4 shows an overview of our Monte-Carlo simulator, that uses random sampling for each configuration of physical error rate, code distance, and noise model for 10 million random trials and uses bootstrap techniques [126] for higher accuracy. We configure surface code with distances ranging from 3 to 25 and consider a default error rate of 10^{-3} (and below) because QEC cannot lower the logical error rate substantially unless the physical error rate is below the threshold (about 1% for surface codes). Also, higher physical error rates require larger code distance and hence, an increased number of physical qubits. Thus, it is challenging to run practical applications [90] on devices with error rates of 10^{-2} and hence, is not considered in this paper. For systems with 100 to 1000 logical qubits, error rates around 10^{-3}, far below threshold, are needed to run most applications.

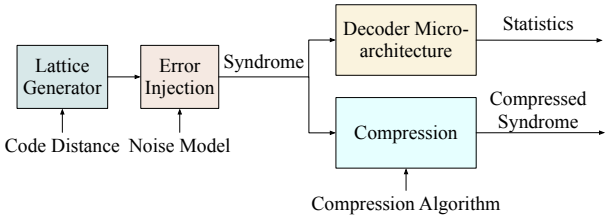

Figure 4. Overview of Monte Carlo simulation framework

The simulator generates a syndrome for a surface code lattice of given distance by injecting errors based on the noise model. The simulator uses the syndrome and models the proposed decoder micro-architecture, decodes the syndrome, estimates the performance, and utilization of each component in the design. The statistics from the simulator are used to draw insights and optimize the micro-architecture for systems with large number of qubits. The syndromes generated also help with determining the effectiveness of the compression algorithms described in Section VI.

B. Noise model

We use the *phenomenological noise model* [36] that accounts for data qubit errors and measurement errors. Given that X-type and Z-type errors are corrected independently, we focus on X-type errors. In the phenomenological noise model with parameter p, each round of syndrome measurement is preceded by a round of noise during which each data qubit is affected independently by a X error with probability p. Moreover, each syndrome bit is flipped independently with probability p to model measurement errors. This is the standard model used to study decoders and QEC for last two decades and has been proven to be effective even on recent QEC experiments [2]. Correlated errors or errors due to crosstalk leads to longer chains of errors on the surface code lattice and can be corrected by adjusting the code distance (increasing the lattice size) [84]. As our design is scalable across different code distances, it can handle such errors.

IV. AFS Decoder

In this paper, we propose the *Accurate Fast and Scalable (AFS)* decoder, which is based on the Union-Find decoding (UFD) algorithm [32, 34] for its accuracy, simplicity, and low time-complexity. We briefly describe the UFD algorithm before discussing the micro-architecture of the AFS decoder.

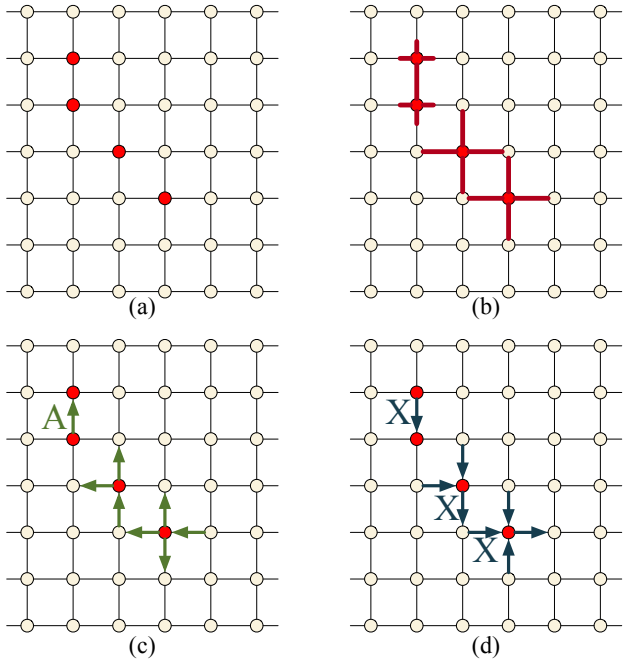

Figure 5. Edges and nodes on the graph represent data and ancilla qubits respectively (standard representation in QEC). (a) Syndrome extracted by the syndrome measurement circuit. Each node has a syndrome bit and red nodes indicate non-trivial or non-zero syndrome bits. (b) Cluster Growth: Clusters are grown by half-edge around the non-zero syndrome nodes until each cluster contains an even number of non-zero syndrome bits. (c) Spanning Forest Generation: A spanning tree is generated for each cluster. (d) Peeling: An error corresponding to the measured syndrome is produced for each cluster by reversing the spanning tree.

A. Background on Union-Find Decoding

UFD is a graph-based algorithm that processes a syndrome in almost-linear time. In the absence of measurement errors, decoding may be treated as a *matching* problem on a square grid (2-dimensional space). Figure 5 illustrates the decoding steps for distance-7 surface code. Our AFS decoder comprises of three pipeline stages that performs these steps.

1) **Cluster Growth**: clusters are grown around the non-zero syndrome bits, as shown in Figure 5(b), until all clusters cover an even number of non-zero syndrome bits. The Graph-Generator (Gr-Gen) stage of the pipeline performs this step.
2) **Spanning Forest Generation**: a spanning tree covering each cluster is created. The data qubits encountered along the traversal (for example, qubit A in

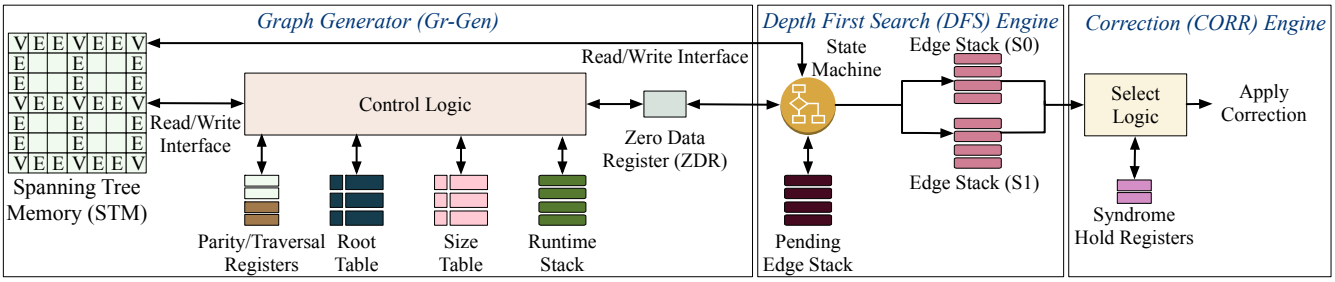

Figure 6. Micro-architecture of the AFS decoder, divided into three pipeline stages.

Figure 5(c)) may require a correction that has to be determined. The Depth First Search (DFS) Engine traverses the spanning trees and stores the list of visited edges which are used by the next pipeline stage of the design.

3) **Peeling**: the error within a cluster is estimated by reverse traversing the spanning tree and consulting the syndrome, as shown in Figure 5(d). The Correction (CORR) Engine in our design uses the list of the visited edges created by the DFS Engine to estimate the best correction for each cluster.

B. Tolerating Measurement Errors

Measurement errors flip syndrome bits and may mislead a decoder if not detected. For example, the non-zero syndrome bit S_0 in Figure 7(a) could be a result of measurement error and if the decoder uses this syndrome, it mistreats it as an error on the data qubit D_0 and degrades the performance of the QEC code by introducing a false correction. Our experiments (in Section II-D) show that failure to tolerate measurement errors can degrade the performance of QEC codes by orders of magnitude. To tolerate measurement errors, decoders analyze d rounds of syndrome measurements [36] simultaneously, resulting in a matching problem in a 3-dimensional space, as shown in Figure 7(b). Each edge on the 3-dimensional graph denotes a potential error: horizontal and vertical edges correspond to errors on data qubits and ancilla qubits respectively.

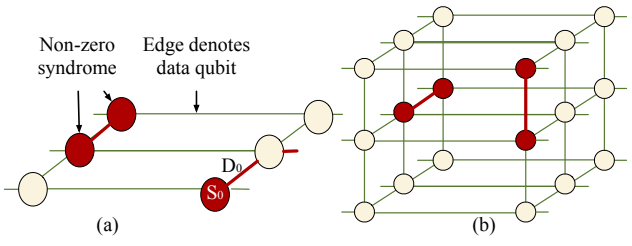

Figure 7. (a) Measurement error flips syndrome bit S_0 and if not detected, it misleads the decoder to treat it as an error on data qubit D_0 (b) To tolerate measurement errors, decoders process d syndrome rounds. A horizontal red edge represents a data qubit error, and a vertical red edge represents the flip of a measurement outcome.

C. AFS Decoder Micro-architecture

Figure 6 shows the micro-architecture of our AFS decoder, and we describe the functionality and optimizations next:

Graph Generator (Gr-Gen): The Gr-Gen accepts the syndrome and generates a spanning forest by growing clusters around non-zero syndrome bits. It consists of a Spanning Tree Memory (STM), Zero Data Register (ZDR), tables to store the root and sizes of clusters, parity registers, tree traversal registers, and a runtime stack, as shown in Figure 6. The STM stores a bit per node and 2 bits per edge (initialized to 0s).[2] The node bits are set for the non-zero syndrome bits and the edge bits flip to 1 when the corresponding edge is added to the spanning forest during cluster growth. To quickly grow odd clusters, a 1-bit parity information per cluster is stored in the parity registers. The Root and Size tables store the root and size of each cluster respectively to aid the *Union()* and *Find()* operations and merge clusters after the growth phase. The tree traversal registers stores the vertices visited during the *Find()* operation. These registers accelerate the *Find()* operation by quickly updating the table entries and compressing the depth of the tree continuously through *path compression*. When two clusters of different sizes are merged, the size table entries are used to add the smaller cluster to the larger one to reduce the overall number of updates. The ZDR is an optimization that stores a bit per row and is initialized to all 0s. When there are one or more non-zero bits in a row, the bit corresponding to the row is set to 1 in the ZDR. The ZDR is used by the next pipeline stage to quickly scan through the STM so that zero memory rows of the STM are not visited.

Depth First Search (DFS) Engine: It processes the data produced in the STM by the Gr-Gen using a *depth first search* algorithm and generates a list of edges by forming a spanning tree while traversing through each cluster. Note that a breadth first search works too but we prefer depth first search since it is generally more memory efficient. The DFS Engine uses stacks to store the edges visited while performing the DFS so that reverse traversal for the *peeling*

[2]Two bits are needed since clusters grow by half edge width [32].

step of the algorithm can be easily accomplished by *popping* off the stack. The ZDR is used to quickly traverse through the STM. Since the ZDR entries are filled when clusters are formed by the Gr-Gen, the DFS engine now only visits the non-zero rows, therefore accelerating the process. The edge stack stores the list of visited edges, whereas the runtime stack stores the edges that will be visited later in the ongoing DFS when a node diverges along multiple paths. Another optimization in this design is the use of an alternate edge stack (Edge Stack S1 in Figure 6) that enables us to fully pipeline the design. When the DFS for one of the clusters is complete, the CORR Engine can start generating its correction while the DFS Engine proceeds to process the data of another cluster.

Correction (CORR) Engine: This stage traverses the spanning forests in reverse direction and identifies the Pauli correction to be applied to the data qubits along the path by using the list of edges generated by the DFS Engine, stored on the stacks. The CORR Engine needs the syndrome to generate the correction and retrieving this data from the STM incurs additional logical complexity and read ports. Instead, the syndrome is stored on the stack along with the edge information by the DFS Engine at the overheads of 2 bits to store the direction and syndrome each, corresponding to the two vertices connected by the edge, per stack entry. Local changes to the syndrome during peeling are stored in the on-chip Syndrome Hold Registers (shown in Figure 6) instead of writing to the STM and reading back. This reduces the number of read and write ports required by the STM.

D. Accuracy of the AFS Decoder

Encoding qubits with a physical error rate p using a distance-d surface code, we obtain a logical qubit whose error rate is given by Equation (1).

$$\text{Logical error rate } = p_{log}(d,p) = 0.15.(40p)^{\frac{(d+1)}{2}} \quad (1)$$

This is referred to as the *logical error rate*. The heuristic formula is derived from the numerical results of [32] and provides a good estimate of the logical error rate for the Union-Find decoder in the regime of low error rate ($p \ll 10^{-2}$). It is valid in the context of the AFS decoder implementing the UFD algorithm and for the phenomenological noise model considered in this paper. Figure 8 shows the logical error rate for different surface code distances and physical error rates. In this paper, we illustrate our design with numerical results for distance-11 surface codes which is a reasonable distance for a first generation of fault-tolerant quantum computers as it allows us to implement non-trivial quantum algorithms on logical qubits while keeping the qubit overhead to a few hundred qubits per logical qubit. For a physical error rate of $p = 10^{-3}$, the logical qubits error rate drops to $p_{log} \approx 6 \times 10^{-10}$ allowing for the implementation of large depth quantum algorithms.

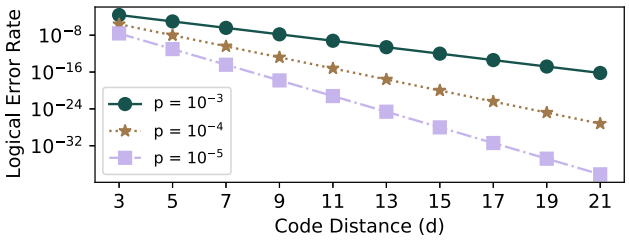

Figure 8. Logical error rate with the AFS decoder for different physical error rates (p) and code distances (d).

E. Latency of the AFS Decoder

The latency is dominated by read operations to the memory structures. The AFS decoder requires up to three sequential memory reads every cycle. This helps us to accurately model the performance of the design. The write operations performed are read-modify-write, and the micro-architecture is designed such that memory write-backs are not on the critical path. The other key operations are flipping adjacent memory locations if a non-zero syndrome bit is encountered (in cluster growth step) and updating the parity bits. Note that the AFS does not require any floating-point arithmetic or matrix operations. We assume a latency of 1 nanosecond (4 cycles latency and a 4 GHz clock frequency) for 32-bit memory accesses from 1 KB on-chip memory [77]. The number of memory accesses in the Gr-Gen for a syndrome is proportional to the total number of clusters and diameter of each cluster and the time spent in this stage (τ_{GG}) is given by Equation (2), where C_i is the i^{th} cluster and m is the total number of clusters in the syndrome.

$$\tau_{\text{GG}} = \sum_{i=1}^{m} \sum_{j=1}^{diam(C_i)} j^2 \quad (2)$$

The number of memory operations in the DFS Engine and CORR Engine is proportional to the size of a cluster (or the number of vertices $|V(C_i)|$ in the i^{th} cluster C_i). Equation (3) denotes the time spent in the DFS Engine (τ_{DFS}) and CORR Engine (τ_{CE}) for a syndrome with m clusters.

$$\tau_{\text{DFS}} = \tau_{\text{CE}} = \sum_{i=1}^{m} |V(C_i)| \quad (3)$$

Note that there is no single number that can quantify the latency of any decoder, as the latency is dependent on the syndrome (easier syndromes take less decoding time). Using the above model, for the AFS decoder, we obtain an average latency of 42 ns for a single logical qubit encoded using distance 11 surface codes. Based on the likelihood of complex syndromes, we observe that the 99.9th percentile latency of decoding is less than 150 ns. Note that these values are significantly lower than the timing budget of 400 ns typically provisioned for decoding operations [51, 59].

For the latency analysis, we assume a dedicated decoder design where there are no resource conflicts across decoding operations from different logical qubits. In Section V, we optimize the decoder design to be more hardware efficient, which can cause resource conflicts and increase the latency. We defer the detailed evaluation of latency to Section V-E.

F. Storage Overhead for the AFS Decoder

It is preferred that decoders operate close to the quantum substrate in a cryogenic regime. Therefore, our main design constraint is limited hardware resources, particularly memory available in the cold environment. Memory constitutes for the major part in the proposed design and the AFS micro-architecture requires 4.5 KBytes of memory for physical error rate 10^{-3} and surface code distance 11. Note that two decoders are needed per logical qubit to decode X and Z errors independently. Table I shows the memory required by AFS decoder for a logical qubit in distances 11 and 25.

Table I
MEMORY REQUIRED FOR A LOGICAL QUBIT ENCODED USING DISTANCE d SURFACE CODE AND PHYSICAL ERROR RATE 10^{-3}

Design Component	$d = 11$	$d = 25$
STM (Gr-Gen) (KB)	2.07	25.6
Root Table (Gr-Gen) (KB)	3.25	51.3
Size Table (Gr-Gen) (KB)	3.54	54.9
Stacks (DFS Engine) (KB)	0.08	1.41
Total (KB)	8.95	133

V. CONJOINED-DECODER ARCHITECTURE

Existing decoders only focus on improving the accuracy and latency of a single logical qubit, ignoring the scalability constraint. However, an FTQC for practical applications require hundreds of logical qubits and each of them must constantly undergo QEC. For example, a quantum chemistry application to study nitrogen fixation requires 100 to 1000s of logical qubits [90]. Failure to detect an error on any one of the qubits may produce an incorrect output for the program. Prior studies implicitly assume a simple system-level decoder architecture where each logical qubit accesses its dedicated decoders. While this design maximizes the error correction capability, its hardware complexity grows linearly with the number of qubits, as shown in Figure 9, and hence become resource intensive for large FTQCs.

Sharing decoders between logical qubits improves the scalability but may result in an error going undetected due to lack of timely access to the decoding hardware units. Thus, designing decoder architectures that reduce the hardware cost through resource-sharing is non-trivial. Ideally, the hardware complexity must be minimized without impacting the accuracy and latency, and our proposed *Conjoined-Decoder Architecture (CDA)* achieves this goal.

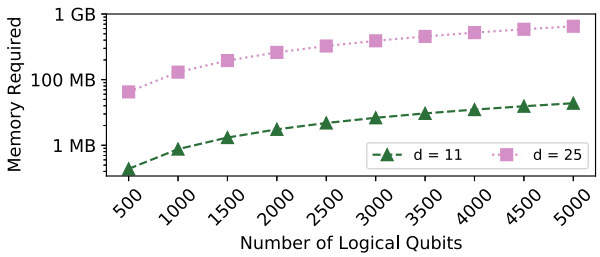

Figure 9. Memory required for the total number of decoders with the number of logical qubits (each qubit requires two decoders, X and Z).

A. Optimized CDA Micro-architecture

The key insight in CDA is that all units of the AFS decoder are not equally utilized, and we can reduce the hardware complexity by sharing infrequently used units across logical qubits. Figure 10 shows the micro-architecture of a decoder block, fundamental unit of CDA, that uses a reduced number of pipeline units of each type. We use non-uniform number of pipeline stages because we observe that certain stages are under-utilized than others when we study the utilization of each stage of our AFS decoder. In a decoder block, groups of Gr-Gen share a DFS Engine and groups of DFS Engines share a CORR Engine. The *Select* logic prioritizes the first-ready component and uses round robin arbitration (for fairness) to generate select signals for the multiplexers. For example, if four Gr-Gen units share a DFS Engine, and the second Gr-Gen finishes cluster generation at the earliest, it gets access to the DFS Engine.

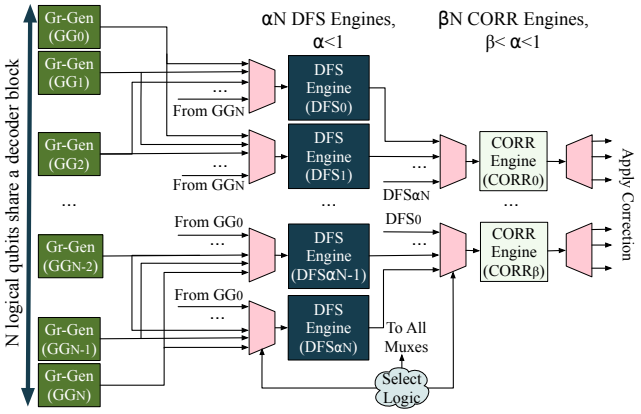

Figure 10. Decoder Block shared by N logical qubits using non-uniform number of Gr-Gen, DFS Engine, and CORR Engine units and resource savings depend on parameters α and β.

CDA reduces hardware complexity because decoder blocks require fewer resources than individual decoders. Figure 11 shows the CDA of an FTQC with L logical qubits, where N logical qubits share a decoder block. Hardware savings depend on parameters α and β and our goal is to design hardware efficient CDA by optimizing these parameters, subject to meeting the accuracy and latency constraints.

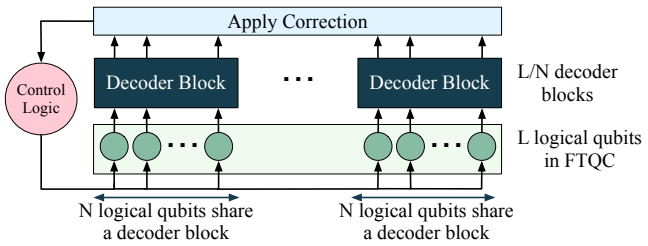

Figure 11. A decoder block in the CDA generates the correction that must be applied for each of its N logical qubits every error correction cycle.

B. Modeling of Error Correction Capability

Error correction of a logical qubit is successful when d rounds of syndrome measurements are decoded within a single round of measurement, which limits the maximum latency that can be tolerated. When a decoder fails to decode all the syndromes within a cycle, errors go undetected.

Sources of system failure: There are two sources of *system failure* in CDA: (a) logical error or (b) timeout failure. No decoder can correct all possible errors. The surface code and Union-Find decoder are designed such that the probability of a logical error is small and decays exponentially with code distance. CDA has an extra source of failure which occurs when an error cannot be detected because a logical qubit is denied timely access to the decoders. We call this a *timeout failure*, and it arises due to conflicts during resource sharing and depends on the architecture.

Accuracy Constraint: The system failure rate does not get impacted if the probability of a timeout failure (p_{tof}) is lower than the probability of a logical error (p_{log}). The design goal is to satisfy this constraint, given by Equation (4), by using minimal decoder blocks to ensure the accuracy of the system is not compromised. The p_{tof} is obtained from the performance model embedded in our simulator.

$$p_{tof} \ll p_{log} \tag{4}$$

C. Optimization Results and Performance

An FTQC with L logical qubits that allocates dedicated decoders to each logical qubit requires 2L AFS decoders to correct both X and Z syndromes. Alternately, in CDA, a Gr-Gen unit grows clusters for both X and Z syndromes and two Gr-Gen units share a DFS Engine and CORR Engine. The optimal number of units for sharing are determined from the fraction of the execution time spent in each stage. Most of the time is spent in the Gr-Gen since spanning forest generation and peeling cannot proceed until clusters are fully grown [32]. The DFS Engine takes lesser time by using two stacks and not waiting for the CORR Engine. Since the design is fully pipelined, the CORR Engine performs the peeling step when a cluster traversal is complete while another cluster is being traversed through by the DFS Engine. Also, the Zero Data Register ensures that the DFS Engine only visits non-zero STM rows, unlike the Gr-Gen.

CDA for an FTQC with L logical qubits requires L Gr-Gen units, L/2 DFS Engines, and L/2 CORR Engines. Thus, the total number of Gr-Gen units, DFS Engines, and CORR Engines are reduced by 2x, 4x, and 4x respectively, effectively reducing the total number of decoders by more than 2x. Consequently, the total memory required is reduced by 50% compared to the baseline. The root and size tables can be shared between two Gr-Gen units. This prevents the Gr-Gen units from simultaneously growing clusters, but the two STMs can still be used in parallel. When one of the STMs is being traversed by a DFS engine, the other can be used by a Gr-Gen unit to grow clusters. This further reduces the memory capacity to 70% at the cost of slight slowdown. The number of qubits sharing a decoder is fixed throughout the architecture for this study to minimize the communication overheads between the logical qubits and the decoding logic. In the current design we minimize this overhead by allowing only limited sharing so that the overhead is few multiplexors and their select logic.

D. Accuracy of CDA

CDA mitigates any adverse impact of resource sharing by ensuring that the probability of the decoder timeout failure is less than the logical error rate. Figure 12 shows the estimated execution time required for different syndromes and their probability of occurrence. We use a timeout threshold of 350ns, which is less than the decoding latency target of 400ns [51, 59]. The shaded region in Figure 12 shows events that lead to a timeout failure and we observe that the probability of a timeout failure p_{tof} is equal to 2 x 10^{-11}, which is significantly lower than the logical error rate ($\approx$ 6 x 10^{-10}). Therefore, the overall failure rate with CDA remains approximately the same as the logical error rate of the dedicated AFS decoder design i.e. $\approx$ 6 x 10^{-10}.

E. Latency of CDA

Figure 12 shows the distribution of estimated execution time for distance 11 and error rate 10^{-3}. The average, median, and 99.9th percentile latencies are 95ns, 85ns, and 190ns respectively. Events that require more than 350ns result in timeout errors and occur with a probability of 2×10^{-11}. Generally, software decoders are too slow [43] and there is increasing interest in the QEC community to develop faster decoders in hardware. Although CDA increases decoding latency compared to a traditional architecture with dedicated decoders, it still meets the target latency. Dedicated decoders are faster but needs more hardware, which is limited in cryogenic environment. The software implementation of Union-Find decoder incurs an average latency of 11 microseconds for distance 11 and error rate 10^{-3} [31] and would not meet our latency target ($\approx 400ns$). Thus, it is not the algorithm itself but our hardware optimization that enables AFS to meet our latency target (our design is 115x faster than the software implementation).

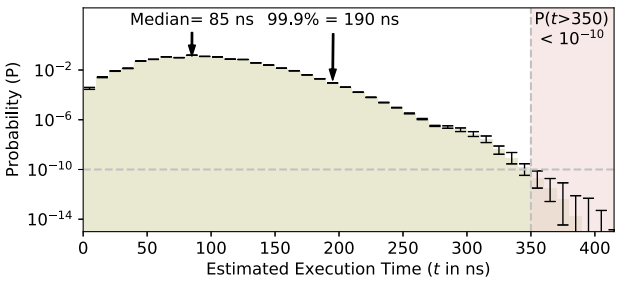

Figure 12. Distribution of execution time for distance 11 and error rate 10^{-3}. Shaded area indicates events that cause timeout failures which occurs with probability 2 x 10^{-11} which is less than logical error rate 6 x 10^{-10}.

F. Comparison with Prior SFQ Decoders

We compare AFS with recent SFQ decoders [59, 113].

(1) Accuracy: The logical error rate of QECOOL for distance 11 and physical error rate 10^{-3} is below 10^{-6} [113] compared to 6×10^{-10} of AFS, making AFS about 4 orders of magnitude more accurate compared to QECOOL. The accuracy differences arise mainly from the way each of these decoders handle measurement errors. To accurately correct measurement errors, a decoder must process d syndrome rounds at once, where d is the distance, resulting in a 3-dimensional decoding graph. The AFS decoder is capable of handling 3-dimensional graphs and therefore, can correct these errors. The NISQ+ decoder [59] on the other hand is only designed for 2-dimensional planar graphs. Its inability to handle measurement errors results in poor accuracy. More recently, Ueno et al. [113] proposed, QECOOL, that extends NISQ+ to tolerate measurement errors. But QECOOL uses a divide-and-conquer approach and does not process the entire 3-dimensional graph at once. Instead, it uses only three syndrome rounds at once and is thus not capable of correcting larger vertical chains of errors that span beyond 3 rounds in the decoding graph. This sub-optimal matching results in poor accuracy of the QECOOL decoder and its inability to lower the logical error rate beyond distance 7.

(2) Threshold: QECOOL results in slightly lower threshold (1%) compared to AFS (about 2.6%) [113]. Note that higher thresholds and lower logical error rates are desirable.

(3) Latency: All three decoders meet the target latency of 400ns. NISQ+ and QECOOL designs benefit from the computational advantages of SFQ technology. On the other hand, the AFS decoder is fully pipelined and breaks down the decoding graph into multiple smaller clusters which are decoded in parallel. The AFS decoder is a memory dominated design with memory read every cycle, followed by few logic gates to perform bit-flips, and write. As the operations in AFS have low logical complexity, the technology for implementing logic (SFQ or CMOS) has negligible effect on the overall latency.

(4) Scalability: NISQ+ and QECOOL decoders may be challenging to scale if the device densities for superconducting logic families do not improve in the next few years. On the contrary, AFS avoids any technology-specific assumptions, therefore even if superconducting technology does not catch up, AFS can still be implemented using CMOS technology.

To summarize, all of the three decoders meet the latency constraint, but the AFS decoder offers higher accuracy and greater scalability and therefore, satisfies the three design constraints of an ideal decoder (described in Section II-C).

G. Storage Overhead for CDA

Table II shows the memory required for each component of the AFS decoder for an FTQC with 1000 logical qubits encoded in distance 11. We observe that an FTQC using AFS decoders with CDA requires 3.5x lesser memory compared to a system with dedicated decoders. This reduces the required memory for this FTQC from about 10MB to 2.8MB. Alternately, for a given memory budget, CDA can support 3x more logical qubits compared to a dedicated decoder design.

Table II
MEMORY FOR AN FTQC WITH 1000 LOGICAL QUBITS.

Design Component	AFS without CDA (MB)	AFS with CDA (MB)
STM (Gr-Gen)	1.97	0.99 (2X)
Root Table (Gr-Gen)	3.17	0.79 (4X)
Size Table (Gr-Gen)	3.46	0.87 (4X)
Stacks (DFS Engine)	1.35	0.34 (4X)
Total	9.96	2.81 (3.5X)

H. Device Technology Considerations

Fault-tolerant quantum computers for most practical quantum applications require hundreds to thousands of qubits. Currently, both superconducting [50, 59, 105] and CMOS devices [8, 87] are being investigated as potential choices for implementing control processor and decoders for such large-scale FTQCs. In our paper, rather than focusing on technology specific choices, we use a simple model to derive latency based on memory accesses (which dominates the latency of AFS). Therefore, our results are applicable to both device technologies and our decoder latency would reduce if a faster memory technology were to arrive. Moreover, both SFQ and CMOS device technologies operating at cryogenic temperatures can support limited memory capacities due to device fabrication and power dissipation challenges respectively. Our system-level optimizations would therefore remain useful irrespective of the device technology used for the actual implementation in the future.

VI. Reducing Bandwidth via Syndrome Compression

The decoders must process the syndromes and provide an estimation of error by the end of the error correction cycle, failure to do which leads to a backlog. As error decoding is a time-sensitive problem, the syndrome data must be supplied to the decoders quickly for immediate decoding.

A. Challenges in Syndrome Transmission

Syndrome transmission requires few hundreds/thousands of gigabytes per second in large FTQCs. As prior decoders are studied at the granularity of a single qubit, they assume the availability of such large bandwidth. To understand the bandwidth requirements for decoding, consider an FTQC with L logical qubits encoded using distance d surface code. This system requires $2d(d-1)L$ bits to be sent at the end of every syndrome measurement round. Any delay in transmission reduces the effective time remaining for decoding. Assuming a syndrome measurement round of 400 ns [51], even if we were to dedicate the entire 400 ns to transmitting syndromes, we would need a bandwidth of 550 Gbps for a system with 1000 logical qubits encoded in distance 11 surface code.[3] Here, we consider a first-order model and report the aggregate bandwidth for simplicity. In reality, each transmission link has limited bandwidth and imposes further challenges.

Figure 13 shows the bandwidth required to transmit syndromes as the time duration for transmission is varied from 100ns to 400ns [51] to 1 microsecond [47]. The bandwidth requirement for distance 11 increases from 220 Gbps at 1 microsecond window, to 550 Gbps for a 400ns window, to 2200 Gbps for a 100ns window. Provisioning for such high bandwidth at cryogenic environment poses a significant challenge owing to the constraint on the number of wires due to thermal leakage. While we discuss the bandwidth issue in the context of AFS, this problem holds true even for any other QEC code, qubit technology, and decoding algorithm.

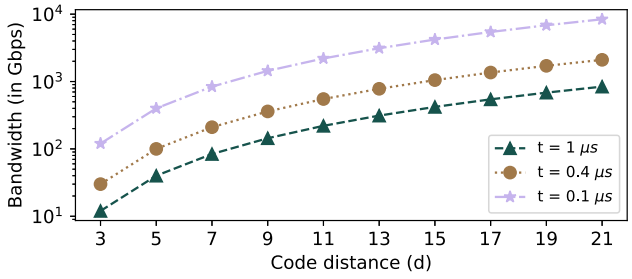

Figure 13. Bandwidth required to transmit syndrome data from qubits to the decoders for a given code distance (d) and an FTQC with 1000 logical qubits for different time duration (t) spent on the data transmission.

[3]The bandwidth requirement for syndromes is different from the instruction bandwidth problem studied by Tannu et al. [105], which can be mitigated by microcode. We discuss this in related work.

B. Insight: Sparsity in Syndrome Values

Syndrome contains information about failed parity checks and, in the common case, we expect them to be zero. Exploiting sparsity can allow us to reduce the bandwidth requirements by using compression. To understand the sparsity in syndrome values, we develop an analytical model. The 3-dimensional decoding graph for X-type errors comprises of $O(3d^3)$ possible error locations (including both X errors and measurement errors). Assuming a physical error rate of p, the expected number of non-zero syndrome bits is at most $6d^3p$ because each error is detected by at most two non-zero syndrome bits. Thus, syndrome is typically sparse. For example, for distance-11 surface code and error rate $p = 10^{-3}$, the syndrome is a binary vector of length $\approx 1,000$ and its average Hamming weight is ≤ 8. The same analysis is true for Z-type errors too. We propose *Syndrome Compression* to exploit this sparsity and reduce the bandwidth requirements.

C. Proposal: Syndrome Compression

Syndrome Compression combines both classical compression schemes and our proposed domain specific scheme. Figure 14 provides an overview of Syndrome Compression (SC). The design contains three compression schemes, namely dynamic-zero compression, sparse representation, and geometry-based compression. SC uses the scheme that provides the highest compression ratio for a given syndrome. *Compression Ratio* is defined as the ratio of the size of the actual syndrome to the size of the compressed syndrome. The three compression schemes used are described next.

1) Dynamic Zero Compression (DZC): We adopt a DZC technique [118] in which the syndrome is grouped into K blocks of W bits each. A K-bit wide Zero Indicator Bit (ZIB) vector consists of 1-bit per block. If all the bits of the i^{th} block are 0, the corresponding ZIB (ZIB[i]) is set to 1. The compressed data consists of the non-zero blocks and the ZIB vector, as shown in Figure 14.

2) Sparse Representation: This scheme stores the indices of only the non-zero elements of sparse matrices. A Sparse Representation Bit (SRB) is used to indicate if all the syndrome bits are 0s. If there are one or more non-zero bits in the syndrome, the SRB is unset and the indices of the non-zero bits are sent with the SRB, as shown in Figure 14.

3) Geometry-based compression (Geo-Comp): This is a domain-specific adaptation of DZC that accounts for the surface code lattice geometry and compresses X and Z syndromes together. It is based on the insight that non-zero syndrome bits generally appear in pairs of neighboring bits and Y errors flip both X-type and Z-type ancilla qubits in the same neighborhood. The compression ratio can be increased by using blocks that respect the lattice structure. With a block decomposition, two neighboring bits fall in the same block (except those on the block boundaries) as shown in Figure 14 and thus, reduces the number of non-zero blocks.

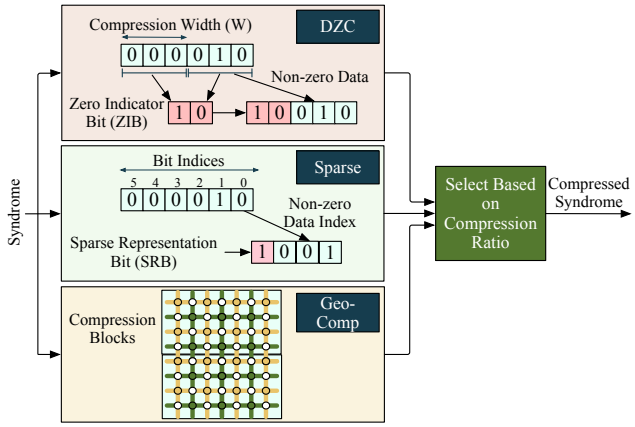

Figure 14. Syndrome Compression (SC) using hybrid algorithm that implements dynamic zero compression, sparse representation, and geometry-based compression and selects the one with the highest compression ratio.

D. Effectiveness of Syndrome Compression

Figure 15 shows the average compression ratio using Syndrome Compression. As the best compression scheme depends on the noise model, physical error rates, and code distance, future FTQCs need hybrid policies, similar to our implementation. We observe that the overall reduction in bandwidth varies between 5x to 380x depending upon the distance and physical error rate. Typically, syndromes for lower error rates have higher sparsity and therefore, sparse representation typically outperforms DZC and Geo-Comp. For the default system considered in this paper with 1000 logical qubits using distance 11 and 10^{-3} physical error rate, the bandwidth, which ranges between 200-2000 Gbps, is reduced by 30x.

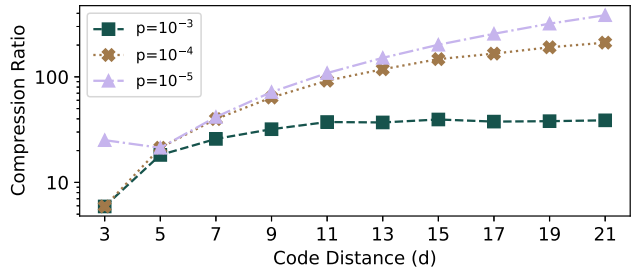

Figure 15. Compression Ratio with Syndrome Compression. A lower physical error-rate increases sparsity and improves the benefit from compression overall.

Syndrome compression has negligible impact on the decoding latency because we use low-latency schemes whose circuit depth scales $O(\log_2 d)$ with the lattice size (d). Also, we assume all syndromes are extracted in parallel [117] and therefore, SC is parallel. We assume local compression as real systems use dedicated transmission links with only a limited amount of bandwidth.

VII. Related Work

Quantum computers require full-stack solutions and interdisciplinary research [18, 72, 96]. This has led to developments in programming languages [22, 54, 93, 103, 122], compilers [21, 63, 95], micro-architecture [37, 50, 60, 105], control circuits [9, 69, 75, 76, 88, 91], and quantum devices. Although near-term quantum systems are promising for some domain-specific problems [41, 74] with support from software error-mitigation [26–28, 52, 53, 67, 68, 78–81, 85, 86, 98, 99, 106–108], FTQCs can solve a broader class of applications. Hence, designing FTQCs is important and recent QEC studies [2, 92, 94] is a step towards this direction. In this section, we describe the works in decoding and system-level studies for FTQCs.

A. Related Work on Decoding Algorithms

Error decoding has been an active area of research, with several decoding algorithms proposed in the literature.

Lookup Table (LUT) Decoder [110]: It uses a lookup table (LUT) to store corrections for every possible syndrome. The LUT is indexed by the syndrome bits and the corresponding entry stores the correction. These decoders are particularly attractive for near-term QEC studies [24] and have been used for demonstrating real-time decoding of color codes [94].

Minimum Weight Perfect Matching (MWPM) [36]: This decoder uses a graph pairing algorithm and is considered to be one of the most effective in terms of accuracy. However, its time complexity ranges from $O(n^3)$ to $O(n^7)$ and implementing is in less than 800 ns is an open problem [59]. Fowler proposed a parallel implementation of this decoder that reduces the average time complexity to $O(1)$, although the worst case complexity remains significant [45].

Machine Learning (ML) Decoders: They train neural networks with the underlying error probability distribution. During decoding, the syndrome data is an input to the neural network that infers the correction [4, 6, 7, 15–17, 19, 29, 65, 70, 73, 82, 104, 111, 114, 116, 119]. They require substantial computational and memory resources to store and process large amounts of training data (in the order of GBs) depending on the code distance. However, many of the design principles described in our paper can be applied to ML-Decoders to improve their performance and scalability.

B. Related Work on Hardware Decoders

Recently, hardware decoders are proposed to improve decoding latency. We discuss some of these studies next:

Superconducting Decoders: SFQ-based decoders [59, 113] represent significant milestones in the field of fast decoders. However, they trade-off accuracy for lower decoding latency and are reliant on technology-specific assumptions.

Hierarchical Decoder: Delfosse et al. propose a hierarchical decoder [31] that uses low-cost decoders for the average-case errors and sophisticated decoders for worst-case errors.

C. Bandwidth Challenges for FTQCs

Prior studies [50, 105] have identified bandwidth bottleneck for supplying instructions from the control processor to the qubits. This bottleneck is alleviated by using micro-code to supply instructions. However, the bandwidth problem we identify differs from prior work, in that we focus on the bottleneck in transmitting syndromes from the qubits to the decoders, and microcode would be ineffective at solving this. Syndrome Compression is inspired from cache and memory compression, however, those designs focus on compressing data values (which may or may not be sparse), whereas we focus on syndromes, which tend to be quite sparse.

D. Applicability beyond Surface Codes

We focus on surface code because it is considered as the most promising quantum error correction code. However, our design can be applied to color codes [11] by projecting them onto surface codes [30]. Moreover, our proposal of co-designing decoders at the system-level and compression to reduce bandwidth requirements is applicable to any decoder and QEC code used for implementation.

VIII. Conclusion

Decoders play a vital role in Fault-Tolerant Quantum Computers and must satisfy three key design aspects: accuracy, latency, and scalability. Most existing decoders focus on achieving higher accuracy, rely on software implementations that are too slow, and neglect the scalability aspect. They do not account for the system-level challenges in scaling the storage and bandwidth required to perform error correction on hundreds of logical qubits in practical FTQCs.

In this paper, we present AFS, an Accurate, Fast, and Scalable decoder that is designed to meet all the three design constraints of a decoder. For physical error rate of 10^{-3}, AFS offers a logical error rate of 6×10^{-10} for logical qubits encoded in distance-11 surface codes and achieves an average decoding latency of 42ns. We propose a *Conjoined-Decoder Architecture (CDA)* that reduces the complexity of decoding hardware by sharing resources across different multiple logical qubits, effectively reducing the memory capacity by 3.5x. Lastly, we propose *Syndrome Compression* that reduces the effective bandwidth required to transmit syndrome data from the quantum substrate to the decoders by 30x on average.

Acknowledgement

An earlier version of this paper was uploaded on arXiv [25] in January 2020. We thank the reviewers of HPCA-2020, ISCA-2020, ISCA-2021, MICRO-2021, and HPCA-2022 for their comments and feedback. We also thank Dave Wecker, Dave Probert, Michael Beverland, and Helmut Katzgraber for the technical discussions. Poulami Das was supported by the Microsoft PhD fellowship.

References

[1] D. Aharonov and M. Ben-Or, "Fault-tolerant quantum computation with constant error rate," *arXiv preprint:9906129*, 1999.

[2] G. Q. AI, "Exponential suppression of bit or phase errors with cyclic error correction," *Nature*, 2021.

[3] P. Aliferis, D. Gottesman, and J. Preskill, "Quantum accuracy threshold for concatenated distance-3 codes," *arXiv preprint quant-ph/0504218*, 2005.

[4] P. Andreasson, J. Johansson, S. Liljestrand, and M. Granath, "Quantum error correction for the toric code using deep reinforcement learning," *Quantum*, vol. 3, p. 183, 2019.

[5] H. Anwar, B. J. Brown, E. T. Campbell, and D. E. Browne, "Fast decoders for qudit topological codes," *New J. Phys.*, vol. 16, no. 6, p. 063 038, 2014.

[6] P. Baireuther, M. Caio, B. Criger, C. W. Beenakker, and T. E. O'Brien, "Neural network decoder for topological color codes with circuit level noise," *New J. Phys.*, vol. 21, 2019.

[7] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. W. Beenakker, "Machine-learning-assisted correction of correlated qubit errors in a topological code," *Quantum*, 2018.

[8] J. C. Bardin, E. Jeffrey, E. Lucero, T. Huang, O. Naaman, R. Barends, T. White, M. Giustina, D. Sank, *et al.*, "29.1 a 28nm bulk-cmos 4-to-8ghz¡ 2mw cryogenic pulse modulator for scalable quantum computing," in *ISSCC*, IEEE, 2019.

[9] J. C. Bardin, E. Jeffrey, E. Lucero, T. Huang, O. Naaman, R. Barends, T. White, M. Giustina, D. Sank, P. Roushan, *et al.*, "29.1 a 28nm bulk-cmos 4-to-8ghz‘ 2mw cryogenic pulse modulator for scalable quantum computing," in *ISSCC*, IEEE, 2019.

[10] S. D. Barrett and T. M. Stace, "Fault tolerant quantum computation with very high threshold for loss errors," *PRL*, 2010.

[11] H. Bombin and M. A. Martin-Delgado, "Topological quantum distillation," *PRL*, vol. 97, 2006.

[12] S. Bravyi and J. Haah, "Quantum self-correction in the 3d cubic code model," *PRL*, vol. 111, 2013.

[13] S. Bravyi, M. Suchara, and A. Vargo, "Efficient algorithms for maximum likelihood decoding in the surface code," *Phys. Rev. A*, vol. 90, 2014.

[14] N. P. Breuckmann, K. Duivenvoorden, D. Michels, and B. M. Terhal, "Local decoders for the 2d and 4d toric code," *arXiv preprint arXiv:1609.00510*, 2016.

[15] N. P. Breuckmann and X. Ni, "Scalable neural network decoders for higher dimensional quantum codes," *Quantum*, vol. 2, 2018.

[16] C. Chamberland and P. Ronagh, "Deep neural decoders for near term fault-tolerant experiments," *Quantum Science and Technology*, vol. 3, 2018.

[17] C. Chinni, A. Kulkarni, and D. M. Pai, "Neural decoder for topological codes using pseudo-inverse of parity check matrix," *arXiv:1901.07535*, 2019.

[18] F. T. Chong, D. Franklin, and M. Martonosi, "Programming languages and compiler design for realistic quantum hardware," *Nature*, vol. 549, 2017.

[19] L. D. Colomer, M. Skotiniotis, and R. Muñoz-Tapia, "Reinforcement learning for optimal error correction of toric codes," *arXiv preprint:1911.02308*, 2019.

[20] B. Criger and I. Ashraf, "Multi-path summation for decoding 2d topological codes," *Quantum*, 2018.

[21] A. Cross, "The ibm q experience and qiskit open-source quantum computing software," https://qiskit.org/, 2018.

[22] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, "Open quantum assembly language," *arXiv:1707.03429*, 2017.

[23] A. S. Darmawan and D. Poulin, "Linear-time general decoding algorithm for the surface code," *Phys. Rev. E*, vol. 97, no. 5, p. 051 302, 2018.

[24] P. Das, A. Locharla, and C. Jones, "Lilliput: A lightweight low-latency lookup-table based decoder for near-term quantum error correction," *arXiv:2108.06569*, 2021.

[25] P. Das, C. A. Pattison, S. Manne, D. Carmean, K. Svore, M. Qureshi, and N. Delfosse, "A scalable decoder micro-architecture for fault-tolerant quantum computing," *arXiv preprint arXiv:2001.06598*, 2020.

[26] P. Das, S. Tannu, S. Dangwal, and M. Qureshi, "Adapt: Mitigating idling errors in qubits via adaptive dynamical decoupling," in *MICRO-54*, 2021.

[27] P. Das, S. Tannu, and M. Qureshi, "Jigsaw: Boosting fidelity of nisq programs via measurement subsetting," in *MICRO-54*, 2021.

[28] P. Das, S. S. Tannu, P. J. Nair, and M. Qureshi, "A case for multi-programming quantum computers," in *MICRO*, ACM, 2019.

[29] A. Davaasuren, Y. Suzuki, K. Fujii, and M. Koashi, "General framework for constructing fast and near-optimal machine-learning-based decoder of the topological stabilizer codes," *arXiv:1801.04377*, 2018.

[30] N. Delfosse, "Decoding color codes by projection onto surface codes," *Phys. Rev. A*, vol. 89, 2014.

[31] N. Delfosse, "Hierarchical decoding to reduce hardware requirements for quantum computing," *arXiv:2001.11427*, 2020.

[32] N. Delfosse and N. H. Nickerson, "Almost-linear time decoding algorithm for topological codes," *arXiv preprint arXiv:1709.06218*, 2017.

[33] N. Delfosse and J.-P. Tillich, "A decoding algorithm for css codes using the x/z correlations," in *Intl. Symposium on Information Theory*, IEEE, 2014.

[34] N. Delfosse and G. Zémor, "Linear-time maximum likelihood decoding of surface codes over the quantum erasure channel," *arXiv preprint arXiv:1703.01517*, 2017.

[35] E. Dennis, "Purifying quantum states: Quantum and classical algorithms," *arXiv preprint quant-ph/0503169*, 2005.

[36] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory," *Journal of Mathematical Physics*, vol. 43, no. 9, pp. 4452–4505, 2002.

[37] Y. Ding, A. Holmes, A. Javadi-Abhari, D. Franklin, M. Martonosi, and F. Chong, "Magic-state functional units: Mapping and scheduling multi-level distillation circuits for fault-tolerant quantum architectures," in *MICRO*, IEEE, 2018, pp. 828–840.

[38] G. Duclos-Cianci and D. Poulin, "Fast decoders for topological quantum codes," *PRL*, vol. 104, 2010.

[39] G. Duclos-Cianci and D. Poulin, "Fault-tolerant renormalization group decoder for abelian topological codes," *arXiv preprint:1304.6100*, 2013.

[40] G. Duclos-Cianci and D. Poulin, "Kitaev's z d-code threshold estimates," *Phys. Rev. A*, vol. 87, 2013.

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

[42] A. J. Ferris and D. Poulin, "Tensor networks and quantum error correction," *PRL*, vol. 113, 2014.

[43] A. Fowler, "Towards sufficiently fast quantum error correction," Conference QEC 2017, 2017.

[44] A. G. Fowler, "Optimal complexity correction of correlated errors in the surface code," *arXiv preprint arXiv:1310.0863*, 2013.

[45] A. G. Fowler, "Minimum weight perfect matching of fault-tolerant topological quantum error correction in average $O(1)$ parallel time," *Quantum Information and Computation*, 2015.

[46] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Phys. Rev. A*, vol. 86, 2012.

[47] A. G. Fowler, A. C. Whiteside, and L. C. Hollenberg, "Towards practical classical processing for the surface code," *PRL*, vol. 108, no. 18, 2012.

[48] A. G. Fowler, A. C. Whiteside, and L. C. Hollenberg, "Towards practical classical processing for the surface code: Timing analysis," *Phys. Rev. A*, 2012.

[49] A. G. Fowler, A. C. Whiteside, A. L. McInnes, and A. Rabbani, "Topological code autotune," *Physical Review X*, vol. 2, no. 4, p. 041 003, 2012.

[50] X. Fu, M. Rol, C. Bultink, J. Van Someren, N. Khammassi, I. Ashraf, R. Vermeulen, J. De Sterke, W. Vlothuizen, R. Schouten, *et al.*, "An experimental microarchitecture for a superconducting quantum processor," in *MICRO*, 2017.

[51] J. Ghosh, A. G. Fowler, and M. R. Geller, "Surface code with decoherence: An analysis of three superconducting architectures," *Phys. Rev. A*, 2012.

[52] P. Gokhale *et al.*, "Partial compilation of variational algorithms for noisy intermediate-scale quantum machines," in *MICRO*, ACM, 2019, pp. 266–278.

[53] P. Gokhale, A. Javadi-Abhari, N. Earnest, Y. Shi, and F. T. Chong, "Optimized Quantum Compilation for Near-Term Algorithms with OpenPulse," *arXiv:2004.11205*, 2020.

[54] Google, *Cirq: An open source framework for programming quantum computers*, https://github.com/quantumlib/Cirq.

[55] D. Gottesman, "An introduction to quantum error correction and fault-tolerant quantum computation," in *Quantum information science and its contributions to mathematics, Proc. of Symposia in Applied Mathematics*, vol. 68, 2010, pp. 13–58.

[56] J. W. Harrington, "Analysis of quantum error-correcting codes: Symplectic lattice codes and toric codes," Ph.D. dissertation, Caltech, 2004.

[57] B. Heim, K. M. Svore, and M. B. Hastings, "Optimal circuit-level decoding for surface codes," *arXiv:1609.06373*, 2016.

[58] M. Herold, M. J. Kastoryano, E. T. Campbell, and J. Eisert, "Fault tolerant dynamical decoders for topological quantum memories," *arXiv preprint arXiv:1511.05579*, 2015.

[59] A. Holmes, M. R. Jokar, G. Pasandi, Y. Ding, M. Pedram, and F. T. Chong, "Nisq+: Boosting quantum computing power by approximating quantum error correction," in *ISCA*, 2020.

[60] F. Hua, Y. Chen, Y. Jin, C. Zhang, A. Hayes, Y. Zhang, and E. Z. Zhang, "Autobraid: A framework for enabling efficient surface code communication in quantum computing," in *MICRO-54*, 2021.

[61] A. Hutter, D. Loss, and J. R. Wootton, "Improved hdrg decoders for qudit and non-abelian quantum error correction," *New J. Phys.*, vol. 17, 2015.

[62] A. Hutter, J. R. Wootton, and D. Loss, "Efficient markov chain monte carlo algorithm for the surface code," *Phys. Rev. A*, 2014.

[63] A. JavadiAbhari, S. Patil, D. Kudrow, J. Heckey, A. Lvov, F. T. Chong, and M. Martonosi, "Scaffcc: A framework for compilation and analysis of quantum computing programs," in *Computing Frontiers*, 2014.

[64] A. Y. Kitaev, "Fault-tolerant quantum computation by anyons," *Annals of Physics*, vol. 303, 2003.

[65] S. Krastanov and L. Jiang, "Deep neural network probabilistic decoder for stabilizer codes," *Scientific reports*, vol. 7, 2017.

[66] A. Kubica and J. Preskill, "Cellular-automaton decoders with provable thresholds for topological codes," *PRL*, vol. 123, no. 2, p. 020 501, 2019.

[67] G. Li, Y. Ding, and Y. Xie, "Tackling the qubit mapping problem for nisq-era quantum devices," in *ASPLOS*, 2019, pp. 1001–1014.

[68] G. Li, A. Wu, Y. Shi, A. Javadi-Abhari, Y. Ding, and Y. Xie, "On the co-design of quantum software and hardware," in *Nanoscale Computing and Communication Conference*, 2021, pp. 1–7.

[69] K. Li, R. McDermott, and M. G. Vavilov, "Hardware-efficient qubit control with single-flux-quantum pulse sequences," *Phys. Rev. Applied*, vol. 12, no. 1, p. 014 044, 2019.

[70] Y.-H. Liu and D. Poulin, "Neural belief-propagation decoders for quantum error-correcting codes," *PRL*, vol. 122, 2019.

[71] S. Lloyd, "Universal quantum simulators," *Science*, 1996.

[72] M. Martonosi and M. Roetteler, "Next Steps in Quantum Computing: Computer Science's Role," *arXiv:1903.10541*, 2019.

[73] N. Maskara, A. Kubica, and T. Jochym-O'Connor, "Advantages of versatile neural-network decoding for topological codes," *Phys. Rev. A*, vol. 99, 2019.

[74] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics*, vol. 18, no. 2, p. 023 023, 2016.

[75] R. McDermott and M. Vavilov, "Accurate qubit control with single flux quantum pulses," *Phys. Rev. Applied*, vol. 2, 2014.

[76] R. McDermott, M. Vavilov, B. Plourde, F. Wilhelm, P. Liebermann, O. Mukhanov, and T. Ohki, "Quantum–classical interface based on single flux quantum digital logic," *Quantum science and technology*, vol. 3, no. 2, p. 024 004, 2018.

[77] D. Min, I. Byun, G.-H. Lee, S. Na, and J. Kim, "Cryocache: A fast, large, and cost-effective cache architecture for cryogenic computing," in *ASPLOS*, 2020, pp. 449–464.

[78] P. Murali, J. M. Baker, A. Javadi-Abhari, F. T. Chong, and M. Martonosi, "Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers," in *ASPLOS*, 2019.

[79] P. Murali, N. M. Linke, M. Martonosi, A. J. Abhari, N. H. Nguyen, and C. H. Alderete, "Architecting Noisy Intermediate-Scale Quantum Computers: A Real-System Study," *IEEE Micro*, vol. 40, 2020.

[80] P. Murali, N. M. Linke, M. Martonosi, A. J. Abhari, N. H. Nguyen, and C. H. Alderete, "Full-stack, real-system quantum computer studies: Architectural comparisons and design insights," in *ISCA*, IEEE, 2019, pp. 527–540.

[81] P. Murali, D. C. McKay, M. Martonosi, and A. Javadi-Abhari, "Software mitigation of crosstalk on noisy intermediate-scale quantum computers," in *ASPLOS*, 2020, pp. 1001–1016.

[82] X. Ni, "Neural network decoders for large-distance 2d toric codes," *arXiv:1809.06640*, 2018.

[83] N. H. Nickerson and B. J. Brown, "Analysing correlated noise on the surface code using adaptive decoding algorithms," *Quantum*, vol. 3, p. 131, 2019.

[84] E. Novais and E. R. Mucciolo, "Surface code threshold in the presence of correlated errors," *PRL*, vol. 110, no. 1, 2013.

[85] T. Patel and D. Tiwari, "Veritas: Accurately estimating the correct output on noisy intermediate-scale quantum computers," in *SC20*, IEEE, 2020.

[86] T. Patel and D. Tiwari, "Qraft: Reverse your quantum circuit and know the correct program output," in *ASPLOS*, 2021.

[87] S. Pauka, K. Das, J. Hornibrook, G. Gardner, M. Manfra, M. Cassidy, and D. Reilly, "Characterizing quantum devices at scale with custom cryo-cmos," *Phys. Rev. Applied*, vol. 13, no. 5, p. 054 072, 2020.

[88] S. Pauka, K. Das, R. Kalra, A. Moini, Y. Yang, M. Trainer, A. Bousquet, C. Cantaloube, N. Dick, G. Gardner, *et al.*, "A cryogenic interface for controlling many qubits," *arXiv:1912.01299*, 2019.

[89] R. Raussendorf and J. Harrington, "Fault-tolerant quantum computation with high threshold in two dimensions," *PRL*, vol. 98, no. 19, p. 190 504, 2007.

[90] M. Reiher, N. Wiebe, K. M. Svore, D. Wecker, and M. Troyer, "Elucidating reaction mechanisms on quantum computers," *Proc. of the National Academy of Sciences*, vol. 114, 2017.

[91] D. Reilly, "Challenges in scaling-up the control interface of a quantum computer," in *IEDM*, IEEE, 2019, pp. 31–7.

[92] A. Remm, C. K. Andersen, S. Lazar, S. Krinner, N. Lacroix, C. Hellings, A. Di Paolo, F. Swiadek, G. Norris, J. Hermann, *et al.*, "Quantum error correction using a distance three surface code with superconducting qubits.," *American Physical Society*, 2021.

[93] Rigetti, "Pyquil documentation," pp. 64–65, 2019, http://pyquil.readthedocs.io/en/latest.

[94] C. Ryan-Anderson, J. Bohnet, K. Lee, D. Gresh, A. Hankin, J. Gaebler, D. Francois, A. Chernoguzov, D. Lucchetti, N. Brown, *et al.*, "Realization of real-time fault-tolerant quantum error correction," *arXiv preprint arXiv:2107.07505*, 2021.

[95] "Scaffold: Quantum programming language," Princeton Univ. Dept of Computer Science, Tech. Rep., 2012.

[96] N. A. of Sciences Engineering and Medicine, *Quantum Computing: Progress and Prospects*, E. Grumbling and M. Horowitz, Eds. The National Academies Press, 2019, ISBN: 978-0-309-47969-1.

[97] M. Sheth, S. Z. Jafarzadeh, and V. Gheorghiu, "Neural ensemble decoding for topological quantum error-correcting codes," *arXiv:1905.02345*, 2019.

[98] Y. Shi *et al.*, "Resource-Efficient Quantum Computing by Breaking Abstractions," *Proc. of IEEE*, 2020.

[99] Y. Shi, N. Leung, P. Gokhale, Z. Rossi, D. I. Schuster, H. Hoffmann, and F. T. Chong, "Optimized compilation of aggregated instructions for realistic quantum computers," in *ASPLOS*, 2019.

[100] P. W. Shor, "Fault-tolerant quantum computation," in *Conf. on Foundations of Computer Science*, 1996.

[101] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computers," *SIAM review*, vol. 41, 1999.

[102] T. M. Stace and S. D. Barrett, "Error correction and degeneracy in surface codes suffering loss," *Phys. Rev. A*, vol. 81, 2010.

[103] K. M. Svore, A. Geller, M. Troyer, J. Azariah, C. Granade, B. Heim, V. Kliuchnikov, M. Mykhailova, A. Paz, and M. Roetteler, "Q#: Enabling scalable quantum computing and development with a high-level domain-specific language," *arXiv preprint arXiv:1803.00652*, 2018.

[104] R. Sweke, M. S. Kesselring, E. P. van Nieuwenburg, and J. Eisert, "Reinforcement learning decoders for fault-tolerant quantum computation," *arXiv preprint:1810.07207*, 2018.

[105] S. S. Tannu, Z. A. Myers, P. J. Nair, D. M. Carmean, and M. K. Qureshi, "Taming the instruction bandwidth of quantum computers via hardware-managed error correction," in *MICRO*, IEEE, 2017.

[106] S. S. Tannu and M. Qureshi, "Ensemble of diverse mappings: Improving reliability of quantum computers by orchestrating dissimilar mistakes," in *MICRO*, ACM, 2019.

[107] S. S. Tannu and M. K. Qureshi, "Mitigating measurement errors in quantum computers by exploiting state-dependent bias," in *MICRO*, ACM, 2019.

[108] S. S. Tannu and M. K. Qureshi, "Not all qubits are created equal: A case for variability-aware policies for nisq-era quantum computers," in *ASPLOS*, 2019.

[109] B. M. Terhal, "Quantum error correction for quantum memories," *RMP*, vol. 87, 2015.

[110] Y. Tomita and K. M. Svore, "Low-distance surface codes under realistic quantum noise," *Phys. Rev. A*, vol. 90, no. 6, 2014.

[111] G. Torlai and R. G. Melko, "Neural decoder for topological codes," *PRL*, vol. 119, 2017.

[112] D. K. Tuckett, C. T. Chubb, S. Bravyi, S. D. Bartlett, and S. T. Flammia, "Tailoring surface codes for highly biased noise," *arXiv:1812.08186*, 2018.

[113] Y. Ueno, M. Kondo, M. Tanaka, Y. Suzuki, and Y. Tabuchi, "Qecool: On-line quantum error correction with a superconducting decoder for surface code," *arXiv preprint:2103.14209*, 2021.

[114] S. Varsamopoulos, K. Bertels, and C. G. Almudever, "Designing neural network based decoders for surface codes," *arXiv:1811.12456*, 2018.

[115] S. Varsamopoulos, K. Bertels, and C. G. Almudever, "Decoding surface code with a distributed neural network based decoder," *arXiv preprint arXiv:1901.10847*, 2019.

[116] S. Varsamopoulos, B. Criger, and K. Bertels, "Decoding small surface codes with feedforward neural networks," *Quantum Science and Technology*, vol. 3, no. 1, p. 015 004, 2017.

[117] R. Versluis, S. Poletto, N. Khammassi, B. Tarasinski, N. Haider, D. J. Michalak, A. Bruno, K. Bertels, and L. DiCarlo, "Scalable quantum circuit and control for a superconducting surface code," *Phys. Rev. Applied*, vol. 8, 2017.

[118] L. Villa, M. Zhang, and K. Asanovic, "Dynamic zero compression for cache energy reduction," in *MICRO*, 2000.

[119] T. Wagner, H. Kampermann, and D. Bruß, "Symmetries for a high level neural decoder on the toric code," *arXiv:1910.01662*, 2019.

[120] F. H. Watson, H. Anwar, and D. E. Browne, "Fast fault-tolerant decoder for qubit and qudit surface codes," *Phys. Rev. A*, 2015.

[121] D. Wecker, M. B. Hastings, N. Wiebe, B. K. Clark, C. Nayak, and M. Troyer, "Solving strongly correlated electron models on a quantum computer," *Phys. Rev. A*, vol. 92, 2015.

[122] D. Wecker and K. M. Svore, "Liqui|⟩: A software design architecture and domain-specific language for quantum computing," *arXiv preprint arXiv:1402.4467*, 2014.

[123] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, 1982.

[124] J. Wootton, "A simple decoder for topological codes," *Entropy*, vol. 17, no. 4, pp. 1946–1957, 2015.

[125] J. R. Wootton and D. Loss, "High threshold error correction for the surface code," *PRL*, vol. 109, 2012.

[126] P. Young, "Everything you wanted to know about data analysis and fitting but were afraid to ask," *arXiv preprint arXiv:1210.3781*, 2012.