MCMit: Hardware-Software Co-Design for Mid-Circuit Measurement Error Mitigation

Emmanouil Giortamis, Felix Gust, Aleksandra Świerkowska, Sandra Stankovic, Innocenzo Fulginiti, Yanbin Chen, Xiaorang Guo, Benjamin Lienhard†, Martin Schulz, and Pramod Bhatotia

Technical University of Munich, Munich, Germany
†Also with the Walther-Meißner-Institut and the Munich Center for Quantum Science and Technology, Munich, Germany
Email: firstname.lastname@tum.de

***Abstract*—Distributed Quantum Computing (DQC) and Quantum Error Correction (QEC) rely on dynamic circuits that include Mid-Circuit Measurements (MCMs) and classical feedback. These operations present a major bottleneck: MCMs suffer from high error rates that lead to real-time branching errors, while MCM and classical feedback latencies amplify decoherence errors. Current hardware controllers, qubit-state discriminators, and software error mitigation techniques fail to address these challenges holistically.**

We propose MCMit, a hardware-software co-design to mitigate branching and latency-induced errors. MCMit introduces a scalable, constant-latency multi-control branch instruction for faster classical feedback and two qubit-state discriminators, a transformer, and a CNN, with high accuracy even under short measurement durations. On the software side, static MCM elimination and stochastic branching complement the hardware by mitigating residual branching errors that persist despite hardware improvements.

We implement MCMit on Qubic and evaluate it using experimentally extracted QPU readout traces. Our branch instruction reduces feedback latency by up to 70%, improving circuit depths by up to $7\times$ over Qubic. Our CNN discriminator achieves up to 62% higher accuracy for short measurement durations than the baselines, driving $1.2-9.4\times$ lower logical error rates in QEC. Last, our software mitigation improves fidelity by 18–30% over baseline methods in DQC workloads.

I. Introduction

Context and motivation. Current Noisy Intermediate-Scale Quantum (NISQ) processors face two primary obstacles hindering progress towards quantum advantage: inherent operational noise and limitations in controlling large qubit counts (scale) [1]. Overcoming noise fundamentally requires the implementation of Quantum Error Correction (QEC) [2]–[4], while transcending scale limitations necessitates Distributed Quantum Computing (DQC) architectures that interconnect multiple processors [5], [6]. Crucially, both QEC and DQC depend on dynamic quantum circuits, which incorporate Mid-Circuit Measurements (MCMs) and subsequent classical feedback [7]–[9]. These operations are indispensable for core operations, such as syndrome measurements in QEC and the classical communication inherent in quantum teleportation protocols, essential for DQC [10], [11].

Despite their utility, MCMs, particularly in superconducting architectures [12], exhibit both high error rates and substantial latency, often representing the slowest and most error-prone operations in a quantum circuit [13]–[15]. Furthermore, the subsequent classical feedback required for dynamic circuits introduces additional, often unaddressed, latency that ranks second only to the measurements themselves, further hindering performance, as we show in Table I.

The above limitations inherent in MCMs present significant challenges for DQC and QEC. Firstly, MCM errors (bitflips) manifest as *real-time branching errors*. Branching errors occur when an MCM error causes the classical control logic to deviate from the intended execution path, triggering an incorrect conditional quantum operation. Unlike terminal measurement errors, which can be mitigated post-execution via statistical corrections on the final outcome distribution [13], [14], [16], branching errors propagate irrecoverably: an MCM error in a teleportation protocol applies the wrong Pauli correction, while a syndrome MCM error in a QEC cycle can trigger a recovery gate that worsens rather than corrects the logical error.

Secondly, the combined latency of MCMs and classical feedback is detrimental; during teleportation, this delay allows the receiver qubit to decohere before the necessary correction can be applied, reducing fidelity [17], [18]. In QEC cycles, this latency extends the time during which logical errors can accumulate between syndrome measurement rounds, directly competing against the code's ability to correct errors. We quantitatively motivate these claims in § III.

Research gap. Unfortunately, current methods for enhancing the fidelity and speed of MCMs and feedback operations are not only *individually limited*, but are developed in isolation, with *no cross-layer co-design* that accounts for their mutual dependencies. First, real-time FPGA controllers [19] lack native support for complex multi-qubit conditional logic, essential for protocols in DQC and QEC [17], [20], leading to feedback *latency that scales with circuit complexity* rather than remaining constant. Second, qubit-state discriminators [21], [22] are optimized for *long readout* durations, making them ill-suited for the *short durations* required by fast QEC cycles, both to avoid decoder backlog under the 1μs latency budget [23] and to sustain high repetition rates, where their accuracy degrades significantly, directly increasing the logical error rate. Third, existing software error mitigation operates purely post-execution [13], [14], [16], adjusting final measurement distributions after the fact, being fundamentally oblivious to real-time branching errors that have already propagated through the circuit, and is designed with no knowledge of hardware calibration data or discriminator confidence. Crucially, none of these systems is aware of the others.

> **Research Question:** How can FPGA controllers, qubit-state discriminators, and software error-mitigation techniques be co-designed such that each layer exploits and compensates for the others, collectively removing the latency and error bottleneck of MCMs in DQC and QEC?

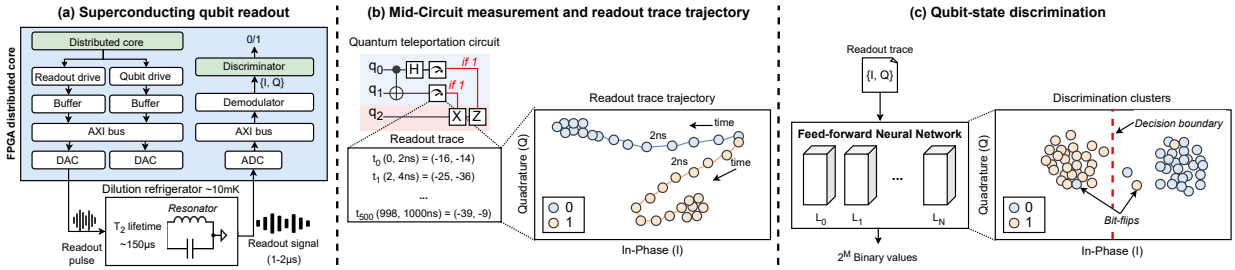


Fig. 1: Superconducting readout (§ II-A). ***(a)*** *Superconducting qubit readout pipeline on an FPGA controller.* ***(b)*** *The MCM in a teleportation circuit can produce a readout trace of 1μs: 500 samples spaced 2ns apart.* ***(c)*** *The readout trace is input to a Feed-forward Neural Network comprising N layers. The network discriminates 0 and 1 using a decision boundary that separates the states.*

Our approach. To answer this question, we propose MCMit, the first hardware-software co-design to mitigate errors and latency associated with MCMs and classical feedback. On the hardware side, MCMit *enables faster classical feedback* by introducing a scalable multi-control branch instruction that has constant-latency w.r.t. the number of qubits as input. MCMit features two powerful qubit-state discriminators, a transformer, and a CNN that achieve high accuracy even with short readout durations, thereby *enabling faster measurements*. Complementing the hardware, MCMit's software *mitigates MCM-induced errors* by employing static analysis to eliminate MCMs and their subsequent feedback where feasible, thereby reducing associated errors and delays, and utilizing stochastic branching to mitigate the impact of branching errors caused by MCM errors.

MCMit's design directly addresses key technical hurdles in MCM error mitigation across the stack, with each layer explicitly informed by the others. Specifically, the utility of fast branching depends on the discriminator's ability to produce reliable outcomes quickly. Short readout durations, necessary to keep the feedback window narrow, demand models that capture temporal signal features with limited data, motivating our transformer discriminator. In turn, even a fast and accurate hardware stack cannot eliminate MCM errors entirely; the software layer closes this gap by consuming hardware-provided confusion matrices to inform stochastic branching decisions at compile time, and by applying static MCM removal to reduce the number of branch instructions the controller must execute at runtime, directly feeding back into hardware latency. Together, these layers form a closed loop: the discriminator enables fast feedback, the branch instruction exploits it, and the software mitigates what hardware leaves unresolved.

We implement the MCMit controller on top of the open-source Qubic 2.0 framework [19] and deploy it on the Xilinx Alveo XCU280 FPGA card, the MCMit transformer and CNN using PyTorch and Torchvision [24], and train them with an Nvidia A40 GPU, and last, the MCMit software in Python using the Qiskit framework [25].

We evaluate MCMit using *real* readout traces from a five-qubit QPU described by Lienhard et al. [26], four micro-benchmarks that utilize MCMs and classical feedback, and two high-level applications, MECH [27] for DQC and the surface code for QEC [28]. On hardware, the MCMit branch instruction incurs up to 70% lower feedback latency, leading to 37%-57% higher fidelity and 7× depth improvement, compared to Qubic. The MCMit CNN outperforms Qubic's ML discriminator, QubiCML, by 11.6% and achieves up to 62% higher accuracy and 30.9% lower circuit depth for short readout durations (250ns-500ns) compared to the state-of-the-art qubit state discriminator HERQULES [21]. This leads to 1.2×-9.4× lower logical error rate for the surface QEC code under current circuit-level noise. Last, on the software side, MCMit achieves 18-30% higher fidelity compared to Qiskit mthree [29] and unmitigated circuits, respectively.

MCMit is available at https://doi.org/10.5281/zenodo.21358489.

Contributions. We make the following contributions:

- We introduce the first holistic hardware-software co-design for MCM and classical feedback error mitigation.
- We present an FPGA-based controller that implements a real-time and scalable branch instruction for constant-latency multi-qubit classical feedback (§ V).
- We present two qubit-state discrimination models designed for enabling high-accuracy short-duration readout (§ VI).
- We present software that simplifies dynamic circuits, mitigating their associating errors and latency, and suppresses branching errors stochastically (§ VII).

II. Background

In this section, we detail the superconducting qubit readout pipeline, the core process executed during each MCM, and then contextualize the importance of MCMs as pivotal operations enabling both scalability through DQC and fault-tolerance via QEC.

A. Qubit Readout in Quantum Architectures

To set the stage, we must first understand the qubit readout pipeline.

Quantum architecture readout pipeline [Fig. 1 (a)]. Qubit-state readout in superconducting QPUs is handled by FPGA control hardware [19], [22], [30], [31]. Fig. 1 (a) shows the pipeline of the Qubic open-source QPU control framework [19], where the core hardware components are shown in green. We refer interested readers to the remaining components in [26], [32], [32]–[35]. The distributed core instructs the readout drive to perform a measurement, and the readout pulse information is sent over the AXI bus to the FPGA's DAC. The pulse is converted into an analog microwave-frequency pulse and transmitted to the qubit's resonator within the dilution refrigerator. The resonator undergoes a qubit-state-dependent phase shift. The resulting transmitted signal is first demodulated and then

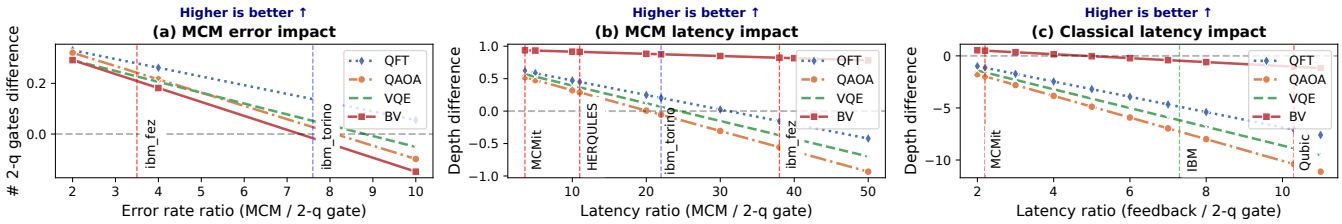


Fig. 2: MECH [27] performance analysis. ***(a)*** *Impact of MCM error rate (normalized to 2-q gate error rate) on MECH's effective 2-q gate count.* ***(c)*** *Impact of classical feedback latency as a ratio to the 2-qubit gate latency. There is a linear performance improvement with MCM error and (classical) latency reduction.*

digitized. The result is a *readout trace* that is the input to a qubit-state discriminator, which generally classifies the trace as 0 or 1.

Readout traces and trajectories [Fig. 1 (b)]. A readout trace is a sequence of In-Phase (I) and Quadrature (Q) $\{I, Q\}$ points forming a 2D plane, and shows the time evolution (trajectory) of the readout signal and the qubit's final state [26], [36]. Fig. 1 (b) shows a readout trace with length 1000ns: 500 samples spaced 2ns apart. On the right, we show an example readout trajectory, where the x-axis represents the I component and the y-axis represents the Q component of the trace points.

Qubit-state discrimination [Fig. 1 (c)]. Qubit-state discrimination outputs the final binary measurement outcome. To perform this task, state-of-the-art discriminators leverage Feed-forward Neural Networks (FNNs) [21], [22], [26], which achieve high accuracy and can be implemented efficiently on FPGA controllers. Fig. 1 (c) shows an example FNN with N layers (input, output, and $N-2$ hidden layers) that produces a 2^M-bit output, with M denoting the number of qubits. The output dimension depends on the design, where some designs output multi-qubit states [21] while others output single-qubit states [22]. Last, these networks create a decision boundary for discriminating states (red dashed line in Fig. 1 (c)) [22], [34].

B. MCMs for Scalability and Fault Tolerance

Scalability: Distributed Quantum Computing (DQC). Monolithic QPUs face key scalability limitations that prohibit the fabrication of QEC-scale chips [37]–[39]. This leads to the use of distributed architectures, such as chiplet designs, which connect multiple QPUs via short-range couplers [40], and DQC setups utilizing quantum networks [6], [41], that utilize components such as quantum repeaters, quantum switches, and quantum transduction [42], [43].

DQC communication primitives. Communication between two QPUs in DQC setups is based on two main communication primitives, which utilize MCMs and classical feedback operations. First, *quantum teleportation* enables communication between distant QPUs using one remote 2-qubit operation, and at least *two* MCMs and *two* classical feedback operations, with variants such as gate and state teleportation [10], [44], [45]. Second, *constant-depth long-range entanglement* consumption involves preparing and using entangled states (e.g., GHZ) at constant circuit depth using dynamic circuits, to apply remote operations efficiently and reduce inter-QPU communication [17], [27], [46], [47]. Crucially, preparing these states in constant depth requires non-trivial classical control flow: MCM outcomes across multiple qubits must be jointly processed via controller-level XOR or majority voting operations to propagate correction logic, before the corresponding feedback gates can be applied [17], [47].

TABLE I: IBM Heron and Nighthawk processors' properties: Absolute values and normalized to those of a 2-qubit gate.[1]

Operation	Error rate	Duration
1-qubit gate	2e-4 ($\leq 0.1\times$)	32ns ($0.35-0.45\times$)
2-qubit gate	1.6e-2-2.7e-3 ($1\times$)	68-88ns ($1\times$)
Measurement	4e-3-3e-2 ($\mathbf{2-12}\times$)	1.5-2.5μs ($\mathbf{23-38}\times$)
Remote 2-q gate [40]	3.5e-2 ($10-18\times$)	235ns ($2.6-3.4\times$)
Classical feedback [19] [18]	0	205-701ns ($\mathbf{3-10}\times$)

Fault tolerance: Quantum Error Correction (QEC). Achieving quantum advantage is widely believed to require fault-tolerant quantum computers, making QEC indispensable for overcoming the intrinsic fragility of qubits and operational noise [3], [4], [28]. QEC works by redundantly encoding a logical qubit across multiple physical qubits, such that common physical errors transform the state into distinguishable error subspaces [48], [49]. Errors are identified by performing MCMs on ancilla qubits, extracting syndrome information that reveals which error subspace the system occupies without collapsing the logical state [50]; these MCMs must be performed cyclically to detect and prevent error propagation [49]. As such, the fidelity and speed of each MCM cycle critically dictate both the quality of the extracted syndrome information and the time available for further errors to accumulate [51].

III. MCM Errors in DQC and QEC Applications

To quantitatively motivate MCMit, we characterize MCM and feedback errors and latencies on state-of-the-art superconducting processors [12], and evaluate their impact on DQC scalability via MECH [27], a compiler for chiplet architectures, and on fault tolerance via the surface code under realistic IBM processor assumptions.

A. Readout Errors and Latency

In the following, we compare the readout error rates, as well as the readout and classical feedback latencies, to the other operations supported by state-of-the-art IBM QPUs.

Readout errors. In currently available superconducting QPUs, qubit readout is the most error-prone operation [12], [13], [15], as shown

in Table I.[2] On current IBM Heron and Nighthawk processors, readout error rates are typically $2-12\times$ higher than those of two-qubit gates, depending on the specific QPU [12]. Specifically, readout errors arise from *relaxation* during the relatively long measurement window [52]–[54], *crosstalk* in frequency-multiplexed setups [26], [55]–[57], qubit *excitations* induced by the readout pulse [57], [58], and imperfect suppression of *environmental noise* despite signal amplification and filtering [59], [60].

Readout latency. Currently, readout is optimized for maximum *terminal* measurement fidelity, not MCM fidelity. As such, measurements are also the slowest operation, taking approximately $20-40\times$ longer than a two-qubit gate on IBM Heron processors [12]. Concretely, readout length spans 1560-2584ns, while a two-qubit gate only spans 68-88ns. Interestingly, this latency "wall" renders dynamic circuits non-optimal compared to unitary circuits or post-selection, despite the use of dynamic decoupling [17].

Classical feedback latency. Interestingly, classical feedback operations are the *second slowest operation* [61]. In the seminal Google quantum error correction experiment, feedback takes 160ns, in contrast to 32ns two-qubit gates and 500ns readout ($5\times$ and 32%, respectively) [62], and IBM reports a 500ns feedback latency in their DQC experiment (20-30% of measurement duration) [18]. Last, the state-of-the-art open-source quantum controller framework, Qubic [19], supports classical feedback for 205-701ns latency ($3\times-10\times$), depending on the complexity of the control flow. Notably, classical feedback becomes first-order in MCMit's target regimes, where feedback latency scales with the number of jointly-processed MCMs and accumulates across rounds and patches.

> **Takeaway:** MCM errors and the combined latency of measurement and feedback severely hinder the performance of dynamic circuits, creating a major bottleneck for QEC and DQC applications.

B. MCM Impact on DQC

We begin our quantitative motivation with MECH [27], the state-of-the-art compiler for chiplet architectures.

Experimental setup. Without available DQC setups, the MECH compiler was originally evaluated at the compilation level using two "effective" metrics: effective CNOT count and effective depth. In this framework, MCMs and remote CNOTs are normalized to the cost of a local CNOT gate, scaling by error rate for gate count and by latency for depth. This normalization means that as MCM cost increases, the reported metric grows proportionally even when physical gate counts remain constant. We extend this evaluation with empirical error rates and latencies from IBM Heron devices [12] for a more realistic assessment of execution-time trade-offs in near-term chiplet-based architectures.

Problem #1: 2-q gate increase from MCM errors. Fig. 2 (a) shows the impact of MCM errors evaluated as the increasing ratio of the MCM to 2-q gate error rates. The y-axis shows MECH's performance difference as effective 2-q gate count difference, where $y>0$ means performance improvement. Here, a remote 2-q gate is $10\times$ noisier than a local 2-q gate. We also plot the MCM/2-q error rate ratio of the `ibm_fez` and `ibm_torino` QPUs [12]. Notably, lower MCM error rates linearly improve MECH performance, since there are $6.4\times$ more MCMs than remote 2-q gates ($5.1-8.6\times$), effectively contributing $1.2-2.3\times$ more errors compared to remote 2-q gates.

[2]Exact ratios depend on the specific device and calibration day; the reported ranges are representative rather than exhaustive bounds.

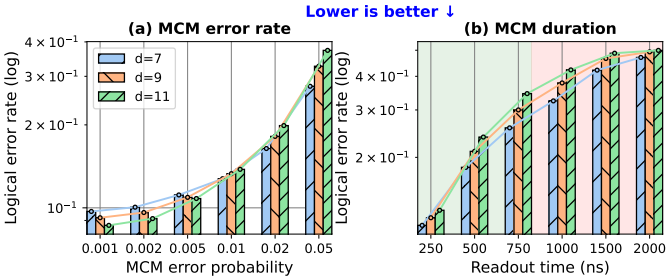


Fig. 3: Logical error rate of the surface code on the Google Sycamore architecture [63]. *(a) Increasing with the MCM error rate. (b) Increasing with the MCM duration. Green marks durations available with MCMit; red marks those available with HERQULES [21].*

Problem #2: Circuit depth increase from MCM latency. Fig. 2 (b) shows the impact of the increasing MCM/2-q gate latency ratio. The y-axis shows MECH's post-compilation depth difference, where every operation is normalized to a 2-q gate, and $y>0$ means lower depth. We assume an MCM/2-q gate error ratio of 2, which is better than ibm_fez in (a). Again, MCM latency reduction linearly increases MECH's depth performance. Interestingly, by enabling fast readout (<750ns), the latency impact of MCMs is lower; for instance, MCMit with high-accuracy 250ns readout would improve MECH's depth by $2.4\times$ and $11\times$ compared to `ibm_torino` and `ibm_fez`, respectively.

Problem #3: Circuit depth increase from classical latency. Last, Fig. 2 (c) shows the impact of increasing classical feedback/2-q gate latency ratio. We assume a MCM/2-q gate latency ratio of 10 (lower than ibm_torino in (b)). To our surprise, classical latency was considered to be *zero* in the original MECH evaluation and thus was *not* accounted for. We plot the feedback latencies of Qubic [19], and those of IBM [17], [18] and MCMit. As latency increases (decreases), MECH's performance deteriorates (improves) in a linear fashion. Notably, by enabling constant-latency feedback, MCMit can achieve a depth $\sim7\times$ lower than Qubic.

> **Takeaway:** MCM errors and latencies detrimentally impact the performance of DQC compilers, since the combined measurement and feedback latency increases the effective circuit depth. However, MCMit can lower the depth by $11\times$ and $7\times$ by enabling faster readout and implementing constant-latency feedback, respectively.

C. MCM Impact on QEC

We evaluate the impact of readout error rate and duration on the surface QEC code used in the Google Sycamore experiment [62].

Experimental setup. We simulate a fault-tolerant lattice-surgery merge within a lattice of surface code of distance 7, 9, and 11. After the merge, we perform $d+1$ rounds of stabilizer measurements. We simulate using the framework of Maurya et al. [64] with a *circuit-level noise model* derived from Google Sycamore [63].

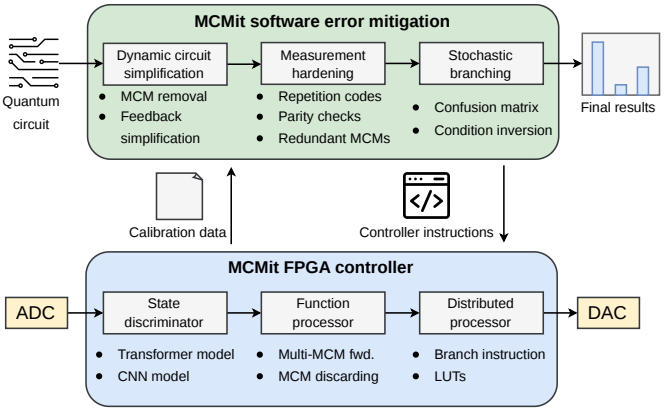


Fig. 4: MCMit overview (§ IV). *MCMit comprises the error mitigation software and the FPGA controller.*

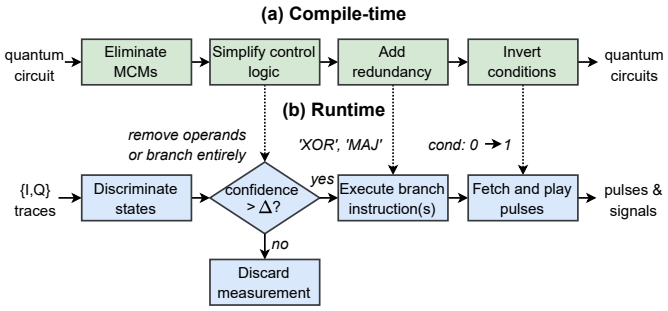


Fig. 5: MCMit workflow (§ IV-B). ***(a)*** *Compile-time workflow and* ***(b)*** *runtime workflow.*

Syndrome measurements are decoded using a minimum-weight perfect matching (MWPM) decoder [65], with the logical error rate defined as the fraction of unsuccessful corrections over 500000 trials.

Problem #1: MCM error effect on logical error rates. Fig. 3 (a) shows the logical error rate change with the MCM error rate. We observe that reducing the MCM error by $10\times$ improves the logical error rate by 47.68%, with the effect becoming more pronounced at higher measurement error probabilities.

Problem #2: MCM latency effect on logical error rates. Fig. 3 (b) shows how MCM duration affects the logical error rate during long execution. Compared to the 250ns duration, the standard 1000ns duration results in a logical error rate that is on average 212.98% higher, while the 2000ns duration is 308.78% higher. Interestingly, MCMit enables high-accuracy readout for short readout traces (<750ns), in contrast to the state-of-the-art HERQULES [21].

> **Takeaway:** MCM errors and latency reduce the effectiveness of QEC: MCM errors directly increase the probability of correction failure, while longer readout durations amplify decoherence effects.

IV. MCMit Overview

To address the MCM bottlenecks characterized in § III, we propose MCMit, a hardware-software co-design comprising three mutually reinforcing layers: real-time multi-qubit branching, fast and high-accuracy qubit-state discrimination, and circuit-level optimization of dynamic circuits.

A. MCMit Architecture

Fig. 4 shows the MCMit architecture, comprising the MCMit software layer for circuit-level error mitigation and the MCMit FPGA controller, which integrates the fast multi-qubit branching logic and the high-accuracy qubit-state discriminator model.

Fast classical feedback. MCMit introduces a fast multi-qubit branch instruction, which executes scalable conditional logic based on the joint outcomes of multiple qubits. This instruction enables more complex, constant-latency feedback protocols that are not possible with existing branching. This functionality is particularly advantageous for routines such as parity checks and majority voting [17], [20], where MCM results are used to directly address a pre-computed LUT of correction operations, guaranteeing low-latency execution.

High-accuracy qubit-state discrimination. We present two novel neural network architectures for qubit-state discrimination that operate directly on *raw* measurement traces, eliminating the need for hand-crafted feature extraction or pre-classifiers. The first is a transformer that leverages self-attention mechanisms to learn complex temporal correlations within the trace, while the second is a lightweight residual CNN that exploits the local temporal correlations inherent in I/Q traces. These models achieve high accuracy even for *short* readout traces, thereby enabling substantially faster measurement cycles that directly translate to lower logical error rates.

Software error mitigation. The MCMit software mitigates MCM-induced errors through three stages: *dynamic circuit simplification* tracks qubit-state evolution to identify MCMs that can be replaced with probabilistic equivalents, and simplifies the corresponding classical control logic; *measurement hardening* adds redundancy via repetition codes, parity checks, and repeated measurements; and *stochastic branching* probabilistically modifies feedback conditions using hardware-provided confusion matrices to suppress branching errors.

B. MCMit Workflow

Fig. 5 shows the compile-time and runtime workflows of MCMit. The software layer operates only at compile-time, while the FPGA controller layer operates only at runtime.

Compile-time workflow. To apply the first stage to the input quantum circuit, dynamic circuit simplification, we execute three substages: (1) we identify and remove up to M MCMs, where M dictates the computational complexity of our static analysis tool, and (2), we simplify the control logic that is based on those MCMs. This removes some operands (e.g., XOR) in subsequent branch instructions or eliminates the branches entirely, effectively reducing classical latency. (3) We apply the measurement hardening pass, which adds redundancy via repetition codes, parity checks, and/or repeated MCMs. This stage modifies branch instructions by swapping "equal" for XOR or majority voting conditions on MCM results. (4) The stochastic branching stage inverts the branch conditions stochastically based on the bitflip error probabilities specified in the confusion matrices.

Runtime workflow. For each set of MCMs that generates a set of $\{\mathrm{I}, \mathrm{Q}\}$ traces, (1) we first discriminate the qubit-states and extract the discrimination confidence. (2) If it is below a threshold Δ, we

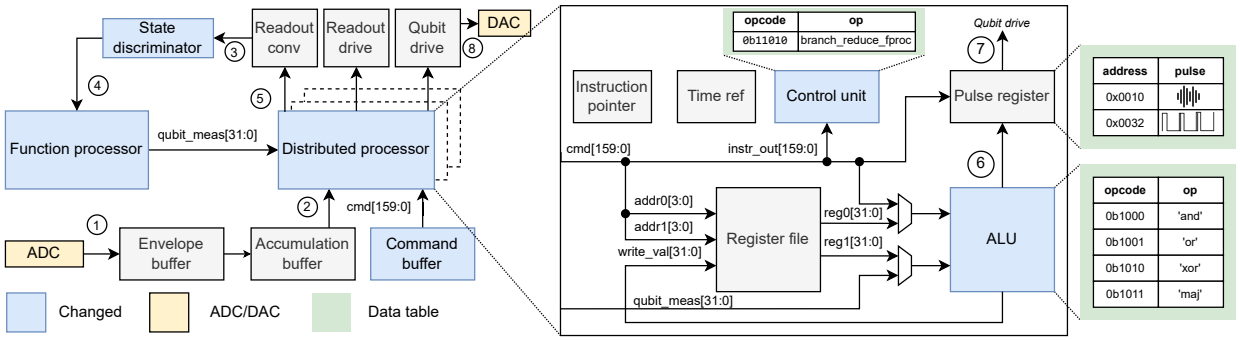


Fig. 6: MCMit controller (§ V). *Blue boxes show modified components, yellow boxes show ADC/DAC components, and green boxes show example data tables. Steps (1)-(8) are executed for a conditional feedback operation based on an MCM result.*

discard (some of) the measurements, and the shot must be rerun. (3) We execute the branch instruction(s) that execute the feedback operation. The instruction's operands and conditions are already modified by the software layers as previously discussed. (4) We fetch the pulses to play from memory, if any, and we send them to the qubit drive, where the pulses eventually become signals at the DAC.

V. Enabling Scalable Classical Feedback

Existing controllers only support high-latency, single-qubit branching that cannot natively execute the complex multi-qubit conditional logic required by DQC and QEC (§ III). We address this by introducing a controller with a constant-latency multi-input branch instruction that natively supports multi-qubit operations.

A. MCMit Controller Overview

Fig. 6 shows the design of our FPGA controller, based on the open-source Qubic framework [19]. At a high level, the controller comprises a *function processor*, a *distributed processor*, a set of *buffers*, and a set of *interfaces*.

Function processor. The function processor is responsible for applying processing functions to qubit readout results and, as such, supports an interface between external components, such as state discriminators or dedicated QEC decoders, if needed.

Distributed processor. The distributed processor comprises multiple cores, each managing interfaces to the function processor and inter-core synchronization. Cores execute programs featuring precisely timed pulse commands, controlling AWG characteristics like frequency and envelope shape. Concurrently, cores implement arbitrary control flow, such as conditional branching, based on external inputs or internal state using a 16x 32-bit register bank and an ALU that performs integer comparisons and arithmetic. ALU results can update registers, adjust the timing reference, or modify the program counter, enabling dynamic execution paths. The distributed processor natively implements branching logic using LUTs, processing MCMs results in parallel as they are dispatched from the function processor, enabling simultaneous classical execution.

Buffers. There are four types of buffers: (1) the *envelope* buffer, which stores the real and imaginary parts of the pulses' envelopes, (2) the *accumulation* buffer, which stores the processed I/Q trace after down-converting and filtering, (3) the *command* buffer, which stores the detailed information for every command, and (4), the *acquisition* buffer, which we omit for simplicity in the context of this work.

Interfaces. There are three types of interfaces: (1) the *readout converter*, which performs the final trace conversion to a format compatible for state discrimination, (2) the *readout drive*, which sends readout pulses through the DACs, and (3) the *qubit drive*, which sends gate pulses through the DACs.

B. Multi-qubit Branch Instruction

To enable an efficient and scalable multi-qubit branch instruction, we modify the architecture described above, as shown in Fig. 6, and highlight the modifications in blue.

ALU. We modify the ALU with a wider opcode to support new operations, specifically `'and`, `'or`, `'xor`, and `'maj`, which performs majority voting. All new operations support up to 32 qubit inputs and 32-bit masks to select them. Specifically, the ALU receives a 32-bit mask that selects the qubit measurements to operate on as its second input.

Control unit. We modify the control unit to support a wider ALU opcode, and add our new branch instruction `branch_reduce_fproc` (Fig. 6 (top right)).

Command buffer. We modify the command buffer by extending the command length since the ALU requires new opcodes for operations on multiple qubits. As the command memory is organized into 32-bit blocks, we add a new block, increasing the command memory and bus width from 128 bits to 160 bits (5 x 32 bus).

Function processor. Originally, during the single-qubit branch instruction, the distributed processor core requests the qubit measurement from the function processor and receives it as the least significant bit of a 32-bit bus. We modify the function processor to send the whole 32-bit measurement register containing all qubit measurements to the ALU in the case of `branch_reduce_fproc`.

State discriminator. Lastly, we modify the state discriminator by implementing a transformer and a CNN, which we explain in § VI.

C. Controller Workflow

We enumerate the steps the controller will take during an MCM and a dependent feedback operation, as shown in Fig. 6. We assume that previously the readout drive sent a readout pulse; thus, we start from (1), the gathering of the ADC data, which are accumulated in the accumulation buffer. These data are passed to the distributed processor (2), then to the readout converter, which produces the final I, Q trace (3). The state discriminator outputs a 1 bit measurement result per qubit and sends it to the function processor

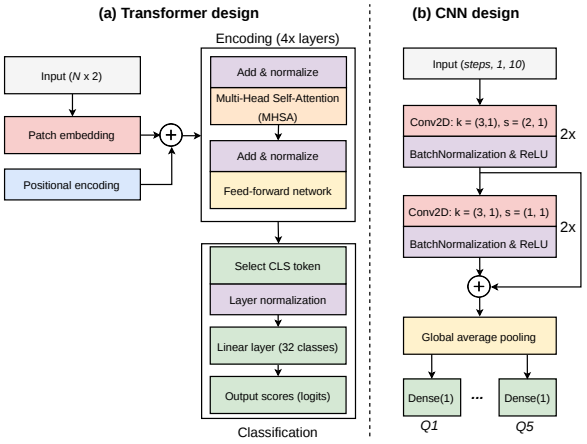


Fig. 7: The MCMit qubit-state discriminators (§ VI).

(4), which, in this case, simply forwards all qubit measurements to the distributed processor (5). Here, the control unit and the ALU fetch the `branch_reduce_fproc` with the appropriate ALU op (e.g., `'maj'`) (6). When the branch is decided, the branch's pulse is fetched from the memory using the pulse register (7), and the pulse is sent to the DAC by the qubit drive (8).

VI. High-Accuracy Qubit-State Discrimination

Fast readout and high-accuracy single-shot qubit-state discrimination lies at the foundation of the DQC and QEC paradigms (§ III). Any enhancement in the readout speed (i.e., shortening the readout length) or discrimination accuracy directly improves the overall fidelity of the entire quantum computation. In this section, we introduce two discriminator designs that operate directly on raw I/Q measurement traces: a transformer-based model that leverages self-attention to capture global temporal dependencies (§ VI-A), and a lightweight residual CNN that exploits local temporal correlations while remaining parameter-efficient (§ VI-B).

Limitations of existing approaches. Previous work on qubit-state discrimination has explored Match Filters (MFs), Support Vector Machines, and FNNs [26]. HERQULES [21] proposed a lightweight FNN combined with MFs to balance accuracy and hardware constraints. Unfortunately, these approaches share a fundamental limitation: they either aggregate the entire readout trace into a single feature vector via boxcar integration (MFs, FNNs) or operate on hand-crafted summary statistics, discarding the temporal structure of the signal entirely. As a result, such architectures are *not suitable for short readout traces*, as we show in Table IV. By contrast, a transformer's multi-head self-attention captures non-local temporal correlations, attending to the most informative segments of a noisy pulse even under minimal integration time, while a CNN's strided convolutional filters efficiently exploit local temporal structure directly on raw I/Q traces, without hand-crafted feature extraction or pre-classifiers, making both architectures inherently more suitable for the short-trace, low-SNR regime.

A. Transformer Design

Design rationale. Before committing to a lightweight architecture, we deliberately explore an expressive, parameter-rich model to establish an accuracy ceiling for the readout discrimination task. The key question is whether a model with hundreds of thousands of parameters and global self-attention can extract discriminative signal that smaller networks fundamentally miss, particularly at short readout durations where the SNR is low. If the transformer significantly surpassed existing discriminators such as HERQULES, this would indicate that long-range temporal correlations carry information that no current lightweight model captures, warranting further investigation into knowledge distillation or architectural insights. Conversely, if a subsequent lightweight design matches the transformer, we can be confident that the deployable model is not leaving accuracy on the table. As we show in § VIII-C, the latter turns out to be the case: our CNN matches the transformer at a fraction of the FPGA cost, validating it as the right design point.

Transformer architecture. Fig. 7 (a) shows the transformer encoder-only architecture. The transformer operates on input traces of shape $(N,2)$ (by default $N=500$). The trace is partitioned into non-overlapping temporal patches (10 samples each), which are linearly projected into an embedding space and prepended with a learnable [CLS] token that aggregates sequence-wide information. Positional encodings are added to preserve timing information, and the sequence is processed by four encoder layers of multi-head self-attention to capture long-range temporal dependencies. The final [CLS] state is fed to a feed-forward classifier to produce the qubit-state classification logits. We train the model using AdamW with weight decay and a Cosine Annealing scheduler, with label smoothing and dropout for regularization. The output represents five qubits, i.e., 32 binary states.

Limitation. Although the transformer achieves high qubit-state discrimination accuracy even for short readout traces (§ VIII-C), it also exhibits a large size that stresses FPGA resources (Table VII). To this end, we propose a CNN design.

B. CNN Design

We detail the architecture of the CNN discriminator (Fig. 7 (b)).

Input pre-processing. Because the raw multiplexed readout trace overlaps every qubit's response in both time and frequency, feeding it to the network unprocessed would waste model capacity on separating channels rather than discriminating states, so we isolate each qubit's channel before it reaches the CNN. To avoid depending on nominal hardware specifications, we calibrate this separation once per setup, picking for every qubit i the demodulation frequency $f_i^* = \operatorname{argmax}_f |S_1(f) - S_0(f)|$ that maximizes the spectral separation between its ground- and excited-state magnitude spectra. Using these fixed frequencies, each channel is then downconverted to baseband and low-pass filtered to suppress out-of-band noise and inter-channel interference, and decimated to shrink the memory footprint, before being stacked into a tensor $X \in \mathbb{R}^{T\times N\times 2}$ across N qubits' I/Q components. We evaluated several filter and decimation configurations for this pipeline and found accuracy relatively insensitive to the exact choice, so we treat them as tunable rather than fixed parameters.

Architecture. Because the FPGA controller operates under a strict latency and resource budget (§ VIII-C), we favor a compact, largely convolutional design over deeper or fully-connected alternatives; across the several such lightweight variants we evaluated, differences in discrimination accuracy were small, so we describe the configuration used throughout our evaluation

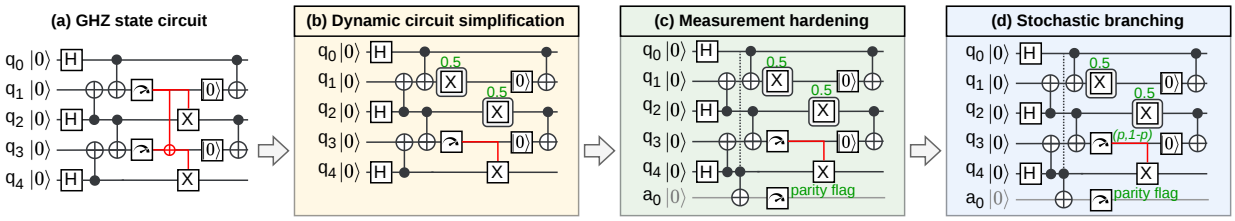


Fig. 8: Software MCM error mitigation (§ VII). ***(a)*** *GHZ state dynamic circuit.* ***(b)*** *Dynamic circuit simplification removes MCMs when possible and simplifies classical conditional logic.* ***(c)*** *Measurement hardening leverages repetition codes, parity checks, and flag qubits to detect and/or correct errors.* ***(d)*** *Stochastic branching factors MCM errors into branching decisions.*

(Fig. 7 (b)). The preprocessed tensor is first reshaped into a multi-channel representation, with each qubit's I/Q components mapped to its own channel, and passed through an input section of two strided convolutional layers (kernel size 3, stride 2); replacing pooling with strided, learnable downsampling lets the network decide how to compress the sequence rather than discarding signal nuances arbitrarily, while the small kernels keep the parameter count low. Since the upstream signal processing has already surfaced much of the discriminative structure, a single residual block is enough to refine it further, learning incremental corrections that capture subtle cross-channel effects such as measurement crosstalk without distorting the dominant per-qubit features. To keep the model compact and robust to temporal shifts in the readout pulse rather than relying on parameter-heavy fully connected layers over flattened features, global average pooling then collapses the temporal dimension into a fixed-length vector. Finally, so that the output scales linearly rather than exponentially with the number of qubits while still implicitly capturing inter-qubit correlations through the shared convolutional trunk, five independent sigmoid neurons (one per qubit) produce binary predictions under a multi-label formulation.

Training and inference. To let the network converge quickly while still fine-tuning in later epochs, we train with the Adam optimizer under a step-decay schedule (initial learning rate 5×10^{-4}, decayed by a factor of 0.95 every 3 epochs); since each qubit's state is only weakly coupled to the others, we supervise each output independently with binary cross-entropy loss rather than treating the 32 joint states as a single classification target. We use a 65/35 train/validation split, a batch size of 64 over 30 epochs, and track per-qubit accuracy and the geometric-mean fidelity across qubits, though we found these training hyperparameters had only a mild effect on the final accuracy across the configurations we tried. Because the FPGA controller cannot host a full-precision model within its latency and resource budget, we apply post-training INT8 quantization after training (Table VII), which costs negligible accuracy given the already-compact dynamic range of the preprocessed input. For inference, a single raw downsampled trace is fed directly into the model, and the five sigmoid outputs are thresholded at 0.5 to yield the per-qubit state predictions.

VII. Software MCM Error Mitigation

To mitigate MCM-induced branching errors that propagate through the circuit (§ III), we present a three-stage pipeline: (1) static MCM elimination via symbolic reasoning to remove measurements and their associated control logic (§ VII-A), (2) measurement hardening via repetition codes, parity checks, and repeated measurements to detect and correct bitflip errors (§ VII-B), and (3) stochastic branching, which probabilistically modifies control flow conditions using hardware-provided bitflip probabilities to reduce incorrect branching (§ VII-C).

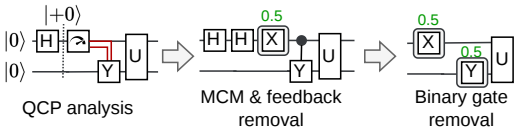


Fig. 9: Dynamic circuit simplification (§ VII-A). *The original dynamic circuit is replaced by a static sub-circuit containing a probabilistic gate, whose outcome is decided at compile time.*

A. Dynamic Circuit Simplification

The first stage simplifies dynamic circuits by removing MCMs when possible, and then simplifies or eliminates their dependent feedback operations. To remove MCMs, we track the qubit states from initialization until the MCM, without using full quantum simulation, based on the quantum constant propagation technique [66].Importantly, this technique is generally not applicable in QEC settings, where syndrome MCMs are crucial for decoding and subsequent corrections. However, it is applicable in any non-FTQC case.

Quantum constant propagation (QCP). QCP is introduced as a symbolic execution technique that improves circuit optimization by identifying and removing redundant gates based on static information available at compile time [66]. QCP tracks quantum states using entanglement-aware symbolic representations, grouping entangled qubits via a union-based data structure. To ensure scalability, QCP imposes a configurable upper bound on the size of tracked entangled sets (parameterized by N_{max}). When this threshold is exceeded, QCP conservatively abandons tracking and assigns the unknown value $\top$. As such, QCP avoids simulation overheads and maintains polynomial runtime complexity.

MCM removal. The work by Chen et al. [67], [68] extends QCP to support circuits with MCMs, demonstrating that many MCMs can be removed during compilation by treating them as static observations under certain conditions. The core idea is shown in Fig. 9: in specific sub-circuits, qubit values at measurement points can be inferred symbolically from initial states and prior gates. When symbolic execution reveals a deterministic value at the point of an MCM, the measurement and its associated classical feedback logic can be replaced by a probabilistic sub-circuit. Naturally, this

improves execution fidelity and reduces latency by eliminating unnecessary measurement and feedback operations.

Limitations and our solutions. This previous work is characterized by two main limitations, which we address in our work: (1) a lack of multi-qubit classical control, since they assume branching based on a single MCM result, and (2) unnecessary 2q-gate overhead, where classical control is replaced with noisy and slow 2q-gates (Fig. 9, middle). For (1), we extend symbolic reasoning to support multi-qubit classical control logic, allowing the compiler to simplify complex feedback operations such as XOR or majority-vote outcomes, effectively relaxing FPGA hardware requirements and lowering classical feedback latency. For (2), we further eliminate 2q gates and replace them with probabilistic 1q-gates. This leads to removing the MCM, the associated classical feedback, and the 2-qubit gates entirely.

Example. Fig. 8 (b) shows the dynamic circuit simplification pass. The first MCM (on qubit q_1) and its dependent feedback X operation are replaced with two probabilistic X gates, with 0.5 probability each. As such, in every shot compilation, one out of four possible circuits will be compiled. Moreover, the propagation of the MCM result to q_3 and the XOR operator are removed entirely.

B. Measurement Hardening

To mitigate MCM errors, we adopt lightweight primitives inspired by QEC, including repetition codes, parity checks, and repeated measurements with majority voting. These techniques serve three purposes: (1) They reduce the likelihood of multiple simultaneous MCM errors through redundancy, (2) enable error detection by exposing bitflips of correlated measurements, and, in the case of repetition codes and repeated measurements, (3) allow for error correction.

Repetition codes. Repetition encoding detects bitflip errors in MCMs by redundantly encoding a logical qubit into d physical qubits and comparing measurement outcomes [69], [70]. Even d allows error detection via parity checks, while odd d enables single-bitflip correction by majority voting. Since CNOT gates used for encoding/decoding also introduce errors, these must be considered. As such, the resulting error probability for a distance-d code with N CNOTs and d measurements is $p_{err}^{rep} = 1 - (1 - p_{CNOT})^N (1 - p_{meas})^d$, where p_{CNOT} and p_{meas} are the per-gate and per-measurement error probabilities, respectively. All encoded qubits are measured simultaneously, and the final outcome is determined using a two-qubit logical AND (for detection) or a three-qubit majority vote (for correction). Notably, these operations are natively supported by the MCMit controller at a constant latency cost (§ V-B).

Parity checks. To protect long-range GHZ states [17], we use stabilizer-based parity checks that detect MCM-induced bitflip errors. For an n-qubit GHZ state $|GHZ_n\rangle$ the stabilizer group is generated by $\langle X_1X_2...X_n, Z_1Z_2, ..., Z_{n-1}Z_n\rangle$, and a subset of the Z_iZ_j stabilizers is non-destructively measured via the ancillae, yielding outcomes $m_k \in +1, -1$ for each $P_k = Z_{i_k}Z_{j_k}$. Any $m_k = -1$ signals a parity violation. We define the error-detection flag $D = 1 - \prod_{k=1}^{r} \frac{1+m_k}{2}$, such that $D = 1$ whenever a bitflip is detected, allowing faulty shots to be discarded and improving overall fidelity without requiring full error correction.

TABLE II: Qubic's latency overhead compared to MCMit for N-qubit conditional operations. Classical feedback includes DAC, ADC, pulse preparation, and state discrimination.

N-qubit input (Operation)	QubiC latency (ns)	
	Branch instr.	Classical feedback
N=1 (Single Branch)	16 ($\times\mathbf{1}$)	205 ($+\mathbf{0}\%$)
N=7 (XOR/Majority)	112 ($\times\mathbf{7}$)	301 ($+\mathbf{46}\%$)
N=8 (XOR)	128 ($\times\mathbf{8}$)	317 ($+\mathbf{54}\%$)
N=15 (XOR/Majority)	240 ($\times\mathbf{15}$)	429 ($+\mathbf{109}\%$)
N=16 (XOR)	256 ($\times\mathbf{16}$)	445 ($+\mathbf{117}\%$)
N=32 (XOR)	512 ($\times\mathbf{32}$)	701 ($+\mathbf{241}\%$)
N=128 (XOR)	2048 ($\times\mathbf{128}$)	2237 ($+\mathbf{991}\%$)

Example. Fig. 8 (c) shows the measurement hardening stage, specifically a parity check. Here, we add an ancilla qubit (a_0) and entangle it with qubits q_0,q_2,q_4 to perform a stabilizer-based parity check.

C. Stochastic Branching

To address branching errors caused by MCM bitflips in dynamic circuits, we leverage confusion matrix [29] data provided by the MCMit controller to inform stochastic compilation of control flow. Specifically, for measurement-conditioned operations, the compiler uses per-qubit bitflip probabilities to probabilistically invert the branching condition across repeated shot executions. For example, if a feedback gate is conditioned on a measured '1' with bitflip probability p, the compiler *partitions* the existing N shots: approximately $(1-p)\cdot N$ instances run with the original condition and $p\cdot N$ with the condition flipped, mitigating control flow divergence across shots. Crucially, the total shot count remains N; no additional shots are introduced.

Formally, for k independent branches, each shot independently samples its branch conditions from the joint distribution. The dominant configuration (no flips) retains $(1-p)^k \cdot N$ shots, while the most important minority configurations (single flips) each receive $p \cdot (1-p)^{k-1} \cdot N$ shots. Adequate statistical coverage requires $N \geq S_{\min}/(p\cdot(1-p)^{k-1})$, where $S_{\min}$ is the minimum number of shots per configuration. For small p this simplifies to $N \gtrsim S_{\min}/p$: with $p = 0.03$ and $S_{\min} = 30$, approximately $N \geq 1{,}000$ shots suffice, well within standard execution budgets (we use $N = 10{,}000$). Higher-order configurations ($j \geq 2$ simultaneous flips) occur with probability $\binom{k}{j} p^j (1-p)^{k-j}$, dropping rapidly and contributing negligibly to overall fidelity.

Example. Fig. 8 (d) shows qubit q_4 which is unprotected from branching errors, since the MCM at qubit q_3 can be erroneous. We compile the circuit as-is $p\cdot N$ times, and $(p-1)\cdot N$ times without the X gate, where p is the readout error probability for this qubit.

D. Applicability Across Application Classes

The three software stages target complementary application scenarios. Dynamic circuit simplification (§ VII-A) applies to DQC and non-FTQC workloads, where MCMs can sometimes be statically resolved, but is generally not applicable in QEC settings where syndrome MCMs are essential for decoding. In contrast, measurement hardening (§ VII-B) and stochastic branching (§ VII-C) apply broadly across all application classes, including QEC, as they mitigate errors in MCMs that must be retained. As a result, the relative contribution of each stage varies with the circuit's

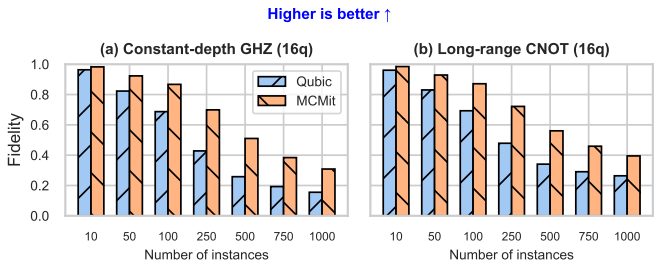


Fig. 10: Classical feedback latency impact (§ VIII-B). *The x-axis shows the number of instances of the GHZ/CNOT circuit in a higher-level application. MCMit achieves* 57.3% *and* 37.8% *higher fidelity than Qubic, on average.*

MCM structure, e.g., circuits with many eliminable MCMs (e.g., constant-depth GHZ) benefit most from simplification.

VIII. Evaluation

A. Experimental Methodology

Implementation. We implement the MCMit controller on top of Qubic 2.0 [19] by modifying the Qubic software (Qubic IR, compiler, assembler, and command generation), the Qubic distributed processor, and the Qubic gateware. We implement the MCMit transformer and CNN with PyTorch v1.7.0., TorchVision v.0.8.1., and hls4ml v.1.2.0 [71], [72]. Last, we implement the MCMit software error mitigation using Qiskit v.2.1.0 [25], Qiskit Aer v.0.17.1, Networkx v.3.5., and Qiskit mthree v.3.0.0.

Quantum hardware. We conduct experiments requiring quantum hardware on `ibm_fez`, an IBM Heron r2 QPU [12]. However, since IBM **does not** give public access to readout traces, we use the traces of Lienhard et al. [26]. Their QPU comprises five frequency-tunable superconducting transmon qubits with alternating target frequencies, capacitive nearest-neighbor coupling, and dispersive coupling to a dedicated readout resonator (bare frequencies 7 GHz); the measured qubit lifetimes (T_1) range from 12–41μs.

Readout trace dataset. The dataset comprises 1.6 million measurement traces experimentally extracted from [26]. Each trace consists of sequentially recorded$\{\mathrm{I},\mathrm{Q}\}$ values captured at a sampling rate of 500 MSamples/sec over a duration of 1μs, representing one of the $2^5=32$ possible computational basis states. The dataset includes 50,000 distinct traces for each of these 32 basis states.

Classical hardware. For classical tasks (such as ML training), we use a server with a 64-core Intel Xeon Gold 6326 CPU at 2.90GHz and 314GB RAM, and an A40 Nvidia GPU and 48GB VRAM. For our controller implementation, we use the Xilinx Alveo XCU280 data center acceleration card with 8GB HBM.

Benchmarks. We use four micro-benchmarks: **(1)** Constant-depth GHZ state and **(2)** Long-range CNOT (i.e., CNOT between non-local qubits), both using dynamic circuits as defined in [17], **(3)** quantum teleportation in a ladder structure, and **(4)** repeated teleportation (on the same qubits) [73]. The quantum state we teleport is $|\psi\rangle = \cos\frac{\pi}{8}\, e^{-i3\pi/8}\,|0\rangle - i\sin\frac{\pi}{8}\, e^{i3\pi/8}\,|1\rangle$, to ensure robustness against bitflip errors. Last, we use two high-level applications: MECH [27], the SOTA compiler for chiplet architectures, and the surface QEC code [74], as described in § III-C.

TABLE III: Qubit-state discrimination accuracy results (§ VIII-C).

Design	Q1	Q2	Q3	Q4	Q5	$\mathbf{F_{5Q}}$
Baseline FNN	0.968	0.746	0.941	0.945	0.970	0.909
QubiCML	0.968	0.731	0.895	0.936	0.957	0.892
HERQULES[3]	0.969	0.740	0.925	0.942	0.970	0.905
MCMit-T	0.970	0.751	0.941	0.945	0.971	0.911
MCMit-CNN	0.971	0.747	0.941	0.947	0.971	0.911

TABLE IV: Discrimination accuracy vs. trace length (§ VIII-C).

Design	Length	Q1	Q2	Q3	Q4	Q5	$\mathbf{F_{5Q}}$
HERQULES	750	0.831	0.721	0.922	0.895	0.969	0.863
	500	0.513	0.663	0.915	0.631	0.936	0.713
	250	0.500	0.550	0.506	0.500	0.500	0.511
QubiCML	750	0.960	0.725	0.905	0.936	0.963	0.892
	500	0.928	0.707	0.915	0.918	0.962	0.880
	250	0.803	0.655	0.878	0.806	0.806	0.786
MCMit-CNN	750	0.964	0.736	0.940	0.943	0.970	0.906
	500	0.939	0.715	0.934	0.927	0.968	0.892
	250	0.835	0.664	0.895	0.833	0.856	0.812

Metrics. We distinguish the evaluation metrics based on a component basis: **(1)** For the `branch_reduce_fproc` instruction, we measure *latency* (in ns), from when the branch opcode starts executing until the if/else pulse starts executing. **(2)** For the discriminators, we measure discrimination *accuracy* ($[0,1]$), discrimination *latency* (in ns), and FPGA resource utilization. **(3)** For the software error mitigation, we measure *Hellinger fidelity* as defined in [75], where it ranges in $[0,1]$ and higher is better.

Baselines. We distinguish our baselines based on the MCMit component we evaluate. For the `branch_reduce_fproc` instruction, we use Qubic [19] as the baseline, since we do not have access to proprietary industrial FPGA controllers. For the MCMit discriminators, we use the baseline FNN design by Lienhard et al. [26],Qubic's ML discriminator [22], and the state-of-the-art HERQULES system [21]. Last, for the MCMit software error mitigation, we use Qiskit mthree [29] as the baseline, since it is the SOTA readout error mitigation method consistently used by IBM [76]–[79].

Training methodology. For each readout length, we train MCMit from scratch on traces truncated to that length. For HERQULES, we instead follow its published "without re-training" protocol: a single network is trained on the full 1000 ns window and applied to shorter traces by truncating the integration window (matched-filter envelopes re-sliced, network weights frozen). Both systems are trained and evaluated on the same calibration dataset with a fixed train/test split, using a matched-filter preprocessing without cross-qubit shot mixing.

B. Classical Feedback Performance

We evaluate MCMit's classical feedback performance by (1) comparing the feedback latency of MCMit to Qubic, and (2) the impact of this latency on quantum applications.

Branch instruction latency. Table II shows the branch instruction and total classical feedback latencies (excluding readout) for Qubic compared to MCMit. On the left, we increase the input size for the XOR or majority voting operation. We additionally include 128-bit operations that our controller supports at the same constant latency. Notably, Qubic incurs up to 32× higher branch latency, which leads

TABLE V: Mean absolute cross-fidelity $\langle |F_{ij}^{\mathrm{CF}}| \rangle$ by qubit separation $|i-j|$ (lower is better; perfect crosstalk immunity $=0$). All-qubit joint readout, full 1000 ns window.

Design	$\lvert i-j\rvert$=1	$\lvert i-j\rvert$=2	$\lvert i-j\rvert$=3	$\lvert i-j\rvert$=4
HERQULES	**0.0033**	0.0111	**0.0015**	**0.0005**
MCMit-CNN	0.0035	**0.0077**	0.0020	0.0011

TABLE VI: Discrimination accuracy as the number of simultaneously measured qubits N increases, averaged over all $\binom{5}{N}$ active-qubit subsets.

Design	N=1	N=2	N=3	N=4	N=5
HERQULES	0.922	0.916	0.911	0.908	0.905
MCMit-CNN	**0.924**	**0.919**	**0.916**	**0.913**	**0.910**

to 241% higher feedback latency, for 32 qubits. Note that 701ns of classical feedback is comparable to measurement duration (Table I).

Feedback latency impact on fidelity. Fig. 10 shows the impact of feedback latency on the fidelities of 16-qubit constant-depth GHZ and long-range CNOT benchmarks. Here, we only model the thermal relaxation errors caused by qubit idling, for the sake of fairness. On the x-axis, we increase the number of instances of the GHZ state or CNOT within a higher-level application (e.g., MECH). Notably, MCMit achieves, on average, 57.3% and 37.8% higher fidelity for the two applications. Last, MCMit improves MECH's effective circuit depth by $\sim 7\times$ compared to Qubic (Fig. 2 (c)).

> **Takeaway:** MCMit's constant feedback latency is up to 70% lower than Qubic's, which leads to 37.8%-57% higher fidelity and up to 7× lower depth for quantum applications, on average.

C. Qubit-State Discrimination Performance

We evaluate the performance of our qubit-state discriminator models w.r.t. (1) discrimination accuracy, (2) inference latency and FPGA resource utilization, and (3) impact on application fidelity.

Discrimination accuracy with long traces. Table III shows the fidelities of the baseline FNN, QubiCML, HERQULES, and the MCMit discriminators, for 1μs readout traces. We show the individual qubit fidelities, and the geometric mean fidelity for all five qubits.

Notably, both MCMit-T and MCMit-CNN are 2.1% more accurate than the baseline FNN, 17.5% more accurate than QubiCML, and 6.3% more accurate than HERQULES. Moreover, our models require no hand-crafted feature extraction or pre-classifiers, and achieve substantially higher accuracy for short readout traces, as we show next.

Discrimination accuracy with short traces. Table IV shows the accuracies for HERQULES, QubiCML, and MCMit-CNN for traces spanning 250–750ns. Interestingly, MCMit-CNN is substantially more robust to short readout traces compared to the baselines. Starting at 750ns (25% faster readout), MCMit-CNN is 31.4% more accurate than HERQULES; the latter drops fidelity by 4.6% while MCMit-CNN only by 0.5% (8.4× less). At 500 ns length (50% faster readout), MCMit-CNN achieves 62.4% higher accuracy than HERQULES (78.8% excluding Qubit 2); HERQULES' fidelity drops by 21.2% while MCMit-CNN by 2.1% (10.1× less). At 250ns length (75% faster readout), MCMit-CNN is 61.6% more accurate (70.8% excluding Qubit 2); HERQULES' fidelity drops by 43.5% while MCMit-CNN only by 10.9% (4.0× less). Last, MCMit-CNN achieves 12.9% (750ns), 9.6% (500ns), and 12.3% (250ns) lower infidelity than QubiCML, on average.

TABLE VII: Discriminator FPGA resource utilization and latency on XCU280 (8-bit quantization, 500 MHz clock).

Design	DSP (%)	BRAM (%)	FF (%)	LUT (%)	Latency (ns)
MCMit-T	23.6	9.0	1.9	29.1	80
MCMit-CNN	0	0	0.10	0.33	70
HERQULES	0.75	0.75	0.41	4.2	40
QubiCML	15	0.94	2.82	1.87	54

Crosstalk mitigation. We quantify crosstalk-induced cross-fidelity as a function of qubit separation/distance, k, i.e., the spurious change in a qubit's readout fidelity caused by the state of another qubit k sites away. HERQULES [21] reports this as a fidelity distance (deviation from the crosstalk-free baseline), so larger values mean stronger inter-qubit leakage and smaller values mean the classifier is more robust to neighboring-qubit activity. Table V shows that the two are essentially on par across all separations: they are nearly identical at short range (k=1: 0.0035 vs 0.0033), our CNN is more robust at k=2 (0.0077 vs 0.0111, about 1.4× smaller), while HERQULES is marginally better at k=3 (0.0020 vs 0.0015) and k=4 (0.0011 vs 0.0005). Both classifiers share the same non-monotonic dependence on k, peaking mildly at k=2, and reach comparably low cross-fidelity overall.

Accuracy scaling with simultaneously measured qubits. We evaluate how discrimination fidelity scales with the number of simultaneously measured qubits, N, by averaging the single-qubit assignment fidelity over all $\binom{5}{N}$ subsets of driven qubits while holding the remaining qubits in the ground state. Table VI shows that MCMit-CNN exhibits both a shallower slope and a higher accuracy at every load as the multiplexed register grows from N=1 to N=5: MCMit-CNN falls by 1.4 percentage points (0.924 → 0.910, a slope of 0.33 pp per added qubit) while HERQULES falls by 1.7 percentage points (0.922 → 0.905, 0.44 pp per qubit). MCMit-CNN stays ahead at every N, reaching 0.910 versus HERQULES' 0.905 at full N=5 while matching HERQULES in crosstalk robustness as the register fills.

FPGA resource utilization and inference latency. Table VII shows the XCU280 FPGA resource utilization and the inference latency for MCMit and the baselines, for five-qubit state discrimination and 1μs readout duration. Although the MCMit transformer incurs higher overheads compared to the baselines, the MCMit-CNN achieves 92% and 82.3% lower LUT utilization, and 75% and 96.4% lower FF utilization compared to HERQULES and QubiCML, respectively.

MCM latency impact on fidelity. Fig. 12 shows the impact of readout duration on the fidelity of quantum teleportation circuits. Both circuits scale in depth with the increasing number of teleportation steps (x-axis). Here, we only model the thermal relaxation errors caused by qubit idling, for the sake of fairness. Notably, a readout length of 250ns (500ns) achieves 5.8% (2.8%) higher fidelity than a readout length of 750ns, on average. Last, MCMit (250ns) achieves 30.9% lower depth than HERQULES (750ns) on MECH (Fig. 2 (c)).

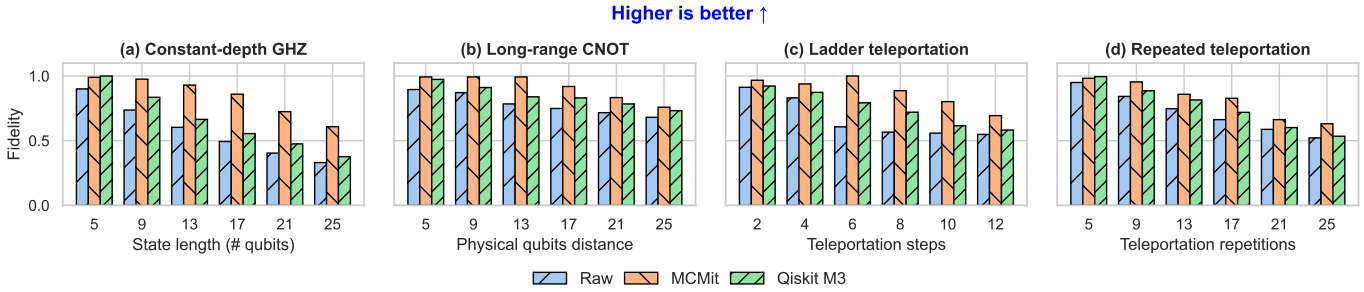


Fig. 11: MCMit software error mitigation impact on fidelity (§ VIII-D). *MCMit achieves 30.47% and 17.78% higher fidelity than raw execution and Qiskit M3, respectively, on average.*

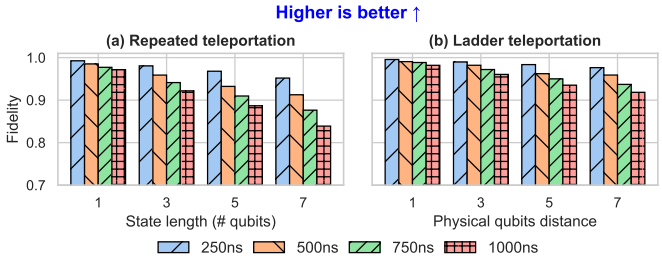


Fig. 12: Readout duration impact on fidelity (§ VIII-C). *The x-axis shows the number of teleportation steps. A 250ns readout achieves 6% higher fidelity than that of 750ns, on average.*

> **Takeaway:** MCMit achieves 62% and 12% higher accuracy for 250ns readout compared to HERQULES and QubiCML, respectively, leading to 30.9% lower depth for DQC applications, on average.

D. Software Mitigation Performance

We evaluate the effectiveness of the MCMit software error mitigation by (1) running the four micro-benchmarks on `ibm_fez` and measuring fidelity, (2) showing the impact of MCM error improvement on MECH and the surface QEC code.

Fidelity improvement. Fig. 11 shows the fidelity evaluation for our four micro-benchmarks. For the constant-depth GHZ state and long-range CNOT benchmarks, we increase the state (distance) length in number of qubits. In the case of quantum teleportation, we increase the number of consecutive teleportations (in a ladder or sequential manner, respectively). "Raw" indicates completely unmitigated results, and the MCMit and Qiskit M3 results do not use other error mitigation techniques, for fairness. Each experiment runs for 10,000 shots on the same calibration cycle of `ibm_fez`.

Notably, MCMit achieves 30.47% and 17.78% higher fidelity than raw execution and Qiskit M3, respectively, on average. Specifically, the improvements for the four applications are: 56.9% and 27.84%, 16.5% and 8%, 33.5% and 17.8%, and 14.6% and 8%. We observe the largest fidelity improvement in constant-depth GHZ, as it requires the most MCMs and classical feedback operations, which are effectively eliminated by our dynamic circuit simplification stage. Also, the GHZ state is protected by the parity checks added by the measurement hardening stage. In contrast, the repeated teleportation circuit achieves the smallest improvement over Qiskit M3, since it scales only with depth and circuit width; thus, our measurement hardening stage is less effective than wide circuits.

TABLE VIII: Compilation runtime to eliminate MCMs.

Application	small n	runtime	large n	runtime
Constant-depth GHZ	5	0.003 s	25	6.8 s
Long-range CNOT	5	0.002 s	25	2272 s (∼38 min)
Repeated teleportation	5	0.017 s	25	0.289 s
Ladder teleportation	5	0.013 s	12	0.113 s

Last, Table VIII shows the compilation runtime to statically eliminate MCMs. The runtime is dominated not by the number of MCMs but by the cost of classically simulating the circuit's intermediate state: it stays negligible for the teleportation variants (small, sparse states) and still scales for GHZ, but grows sharply for the long-range CNOT, whose protocol transiently builds a dense $\sim 2^{n/2}$-amplitude state. Thus the method's practical limit is set by the circuit's classical simulability rather than by its MCM count.

> **Takeaway:** The MCMit software error mitigation achieves $\sim$ 30% and $\sim$ 18% higher fidelity than unmitigated and Qiskit M3 results, respectively, on average, with up to $\sim$ 27.84% improvement over Qiskit M3 for the constant-depth GHZ circuit.

E. QEC Performance

We evaluate the impact of MCMit-CNN's readout fidelity on logical error rates by comparing it to HERQULES [21] under two noise models: (1) a realistic IBM Heron device [80], and (2) a futuristic model where gate errors are downscaled 10× and decoherence times prolonged 3×. The setup is explained in § III-C. We also examine how classical feedback, resulting from additional QEC patches, affects logical error rates and identify regimes of readout latency and fidelity that support effective fault-tolerant execution.

Logical error rate improvement. Fig. 13 (a) and (b) show the logical error rate (LER) achieved using the fidelity and readout duration of MCMit and HERQULES, averaged over $d=\{7,9,11\}$. MCMit achieves lower LERs than HERQULES at every readout length and in both noise regimes. In the current noise regime, MCMit reduces LERs over HERQULES by 1.2–1.5× (250ns), 7.4–9.4× (500ns), and 1.2–1.3× (1μs). Within MCMit, a fast 500ns readout is the sweet spot, lowering LERs by 32–43% (d=7/9/11) compared to 1μs readout. In the futuristic regime, the gap widens sharply: MCMit reduces LERs over HERQULES by 1.3–1.6× (250ns) and 20–41× (500ns), where HERQULES collapses to

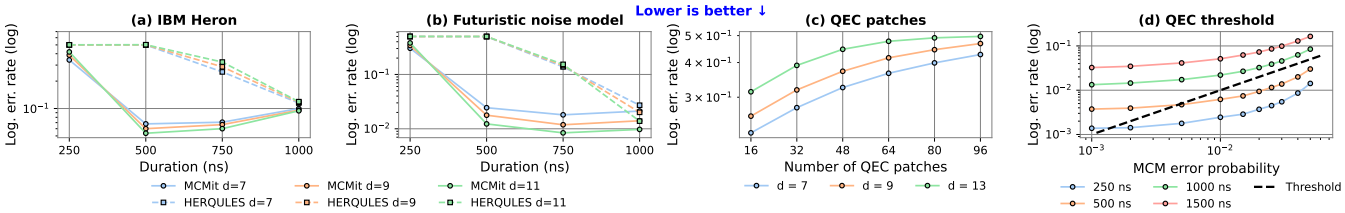


Fig. 13: Impact of varying readout duration and fidelity on logical error rate. *(a) and (b) compare MCMit and HERQULES under two noise models, (c) shows the additional latency in MCMit readout as the number of QEC patches d=7 grows, (d) explores different MCM durations and error rates, highlighting regimes where errors are effectively suppressed on surface code patches d=7.*

a near coin-flip LER (~ 0.50) at 500ns while MCMit stays at 0.012–0.024. In this low-error regime, the optimal readout instead shifts to 750ns (a 7.6–18$\times$ reduction over HERQULES), and a 500ns readout is 15–27% higher than 1μs.

Impact of classical delay from additional patches. For lattices exceeding 16 QEC patches, each additional patch introduces 16 ns of classical feedback overhead. Fig. 13 (c) illustrates the impact of this delay on logical error rate using MCMit readout parameters under the IBM Heron noise model. The degradation trend is consistent across all code distances, notably, with larger distances reaching failure earlier, indicating increased sensitivity to accumulated classical delay.

Exploration of varying latency and error regimes. Fig. 13 (d) presents logical error rates under different combinations of readout latency and error. The gray line marks the threshold where the logical error rate matches or falls below the MCM error probability, indicating regimes that effectively suppress errors. Our results highlight the critical importance of readout improvements, with 500 ns showing the most favorable trade-off between latency and error suppression.

> **Takeaway:** MCMit outperforms HERQULES at every readout length, achieving up to 9.4$\times$ and 41$\times$ lower logical error rates at 500ns readout in the current and futuristic noise regimes, respectively.

IX. Related Work

Readout error mitigation. Readout error mitigation through statistical and machine learning techniques is an active area of research [13], [14], [81]–[88]. However, these methods operate at the level of final execution results, adjusting measurement outcomes post-execution to approximate the correct output probabilities, and therefore are only effective for terminal measurements, not MCMs. In contrast, our work holistically mitigates MCM-induced errors by focusing on MCM and branching errors, and dynamic circuit latency, which causes decoherence errors.

FPGA-based controller frameworks. The development of high-performance, real-time FPGA-based quantum controllers is an active and constantly evolving area of research [19], [30], [31], [34], [61]. However, to our knowledge, existing open-source solutions offer limited support for fast, scalable classical feedback logic, such as multi-qubit conditional branching, and hardware acceleration, like LUTs for constant-time operations critical to QEC and DQC. In contrast, MCMit implements both a constant-latency branch operation and LUTs for complex ALU operations.

Qubit-state discrimination. High-accuracy qubit-state discrimination remains an open problem, with various approaches proposed to maximize readout fidelity [21], [22], [26], [89]–[91]. While some of these models are effective for *long* readout durations, they underperform for short. To this end, we introduce two discriminator designs: a transformer that leverages self-attention to capture non-local temporal dynamics even under short readout durations, and a lightweight residual CNN that exploits local temporal correlations directly from raw downsampled traces without hand-crafted feature extraction or pre-classifiers.

DQC systems software. DQC compilers and runtimes focus on minimizing inter-QPU communication [10], [11], [92]–[105]. While these approaches reduce the frequency of cross-QPU communication, the number of MCMs remains proportionally larger than the number of remote gates and continues to impose a significant fidelity and latency cost (§ III-B). In contrast, MCMit focuses on removing MCMs and their feedback operations entirely, when possible, and mitigating their errors otherwise.

Compilers for QEC. Existing QEC compilers decompose quantum states and gates into fault-tolerant primitives, and manage their mapping and scheduling, primarily for surface code mechanisms such as lattice surgery [106]–[111] or braiding [112], [113]. Consequently, they operate at a higher level of abstraction, rather than directly addressing physical-level error mechanisms.

X. Conclusion

In this work, we presented MCMit, a compiler–controller co-design that addresses the critical bottlenecks of MCM error and feedback latency in DQC and QEC. Our software mitigates errors through static analysis and stochastic branching decisions, while our controller hardware introduces a constant-latency multi-control branch for fast feedback and two high-fidelity discriminators for fast, accurate qubit-state readout. As MCM-heavy dynamic circuits become central to both QEC and DQC, we believe this kind of tight hardware-software co-design will be essential to keeping classical control from becoming the dominant source of error and latency.

Acknowledgments

This work is funded by the Bavarian State Ministry of Science and the Arts as part of the Munich Quantum Valley (MQV), grant number 6090181. B.L. acknowledges support from the German Federal Ministry of Research, Technology and Space (BMFTR) through the program EQuIPS (Grant No. 13N17232).

References

[1] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018. [Online]. Available: https://doi.org/10.22331/q-2018-08-06-79

[2] E. Knill and R. Laflamme, "Theory of quantum error-correcting codes," *Phys. Rev. A*, vol. 55, pp. 900–911, Feb 1997. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.55.900

[3] C. Gidney and M. Ekerå, "How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits," *Quantum*, vol. 5, p. 433, Apr. 2021. [Online]. Available: http://dx.doi.org/10.22331/q-2021-04-15-433

[4] J. Preskill, "Fault-tolerant quantum computation," 1997. [Online]. Available: https://arxiv.org/abs/quant-ph/9712048

[5] M. Caleffi, M. Amoretti, D. Ferrari, J. Illiano, A. Manzalini, and A. S. Cacciapuoti, "Distributed quantum computing: A survey," *Computer Networks*, vol. 254, p. 110672, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128624005048

[6] S. Bravyi, O. Dial, J. M. Gambetta, D. Gil, and Z. Nazario, "The future of quantum computing with superconducting qubits," *Journal of Applied Physics*, vol. 132, no. 16, p. 160902, 10 2022. [Online]. Available: https://doi.org/10.1063/5.0082975

[7] A. D. Córcoles, M. Takita, K. Inoue, S. Lekuch, Z. K. Minev, J. M. Chow, and J. M. Gambetta, "Exploiting dynamic quantum circuits in a quantum algorithm with superconducting qubits," *Phys. Rev. Lett.*, vol. 127, p. 100501, Aug 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.127.100501

[8] T. M. Graham, L. Phuttitarn, R. Chinnarasu, Y. Song, C. Poole, K. Jooya, J. Scott, A. Scott, P. Eichler, and M. Saffman, "Midcircuit measurements on a single-species neutral alkali atom quantum processor," *Phys. Rev. X*, vol. 13, p. 041051, Dec 2023. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.13.041051

[9] Y. Wan, D. Kienzler, S. D. Erickson, K. H. Mayer, T. R. Tan, J. J. Wu, H. M. Vasconcelos, S. Glancy, E. Knill, D. J. Wineland, A. C. Wilson, and D. Leibfried, "Quantum gate teleportation between separated qubits in a trapped-ion processor," *Science*, vol. 364, no. 6443, pp. 875–878, 2019. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.aaw9415

[10] A. Wu, H. Zhang, G. Li, A. Shabani, Y. Xie, and Y. Ding, "Autocomm: A framework for enabling efficient communication in distributed quantum programs," in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. Los Alamitos, CA, USA: IEEE, Oct 2022, pp. 1027–1041.

[11] A. Wu, Y. Ding, and A. Li, "Qucomm: Optimizing collective communication for distributed quantum computing," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 479–493. [Online]. Available: https://doi.org/10.1145/3613424.3614253

[12] IBM Quantum, "Ibm quantum cloud compute resources," https://quantum.ibm.com/services/resources, 2025, accessed: 2025-08-20.

[13] S. S. Tannu and M. K. Qureshi, "Mitigating measurement errors in quantum computers by exploiting state-dependent bias," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '52. New York, NY, USA: Association for Computing Machinery, 2019, p. 279–290. [Online]. Available: https://doi.org/10.1145/3352460.3358265

[14] P. Das, S. Tannu, and M. Qureshi, "Jigsaw: Boosting fidelity of nisq programs via measurement subsetting," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 937–949. [Online]. Available: https://doi.org/10.1145/3466752.3480044

[15] T. Patel, A. Potharaju, B. Li, R. B. Roy, and D. Tiwari, "Experimental evaluation of nisq quantum computers: Error measurement, characterization, and implications," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. Los Alamitos, CA, USA: IEEE, Nov 2020, pp. 1–15.

[16] S. Dangwal, G. S. Ravi, P. Das, K. N. Smith, J. M. Baker, and F. T. Chong, "Varsaw: Application-tailored measurement error mitigation for variational quantum algorithms," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 4*, ser. ASPLOS '23. New York, NY, USA: Association for Computing Machinery, 2024, p. 362–377. [Online]. Available: https://doi.org/10.1145/3623278.3624764

[17] E. Bäumer, V. Tripathi, D. S. Wang, P. Rall, E. H. Chen, S. Majumder, A. Seif, and Z. K. Minev, "Efficient long-range entanglement using dynamic circuits," *PRX Quantum*, vol. 5, p. 030339, Aug 2024. [Online]. Available: https://link.aps.org/doi/10.1103/PRXQuantum.5.030339

[18] A. Carrera Vazquez, C. Tornow, D. Ristè, S. Woerner, M. Takita, and D. J. Egger, "Combining quantum processors with real-time classical communication," *Nature*, vol. 636, no. 8041, pp. 75–79, 2024.

[19] Y. Xu, G. Huang, J. Balewski, R. Naik, A. Morvan, B. Mitchell, K. Nowrouzi, D. I. Santiago, and I. Siddiqi, "Qubic: An open-source fpga-based control and measurement system for superconducting quantum information processors," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1–11, 2021.

[20] S. Xu, K. Liu, P. Rall, Z. He, and Y. Ding, "Distilling magic states in the bicycle architecture," in *International Symposium on Computer Architecture (ISCA)*, 2026, arXiv:2602.20546.

[21] S. Maurya, C. N. Mude, W. D. Oliver, B. Lienhard, and S. Tannu, "Scaling qubit readout with hardware efficient machine learning architectures," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ser. ISCA '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3579371.3589042

[22] N. Vora, Y. Xu, A. Hashim, N. Fruitwala, N. Nguyen, N. Goss, J. Balewski, B. Whaley, I. Siddiqi, P. V. Nguyen, and G. Huang, "Ml–enabled fpga framework for fast quantum state discrimination in mid-circuit measurement regimes," in *Proceedings of the Great Lakes Symposium on VLSI 2026 (GLSVLSI '26)*. ACM, 2026, arXiv:2406.18807.

[23] B. M. Terhal, "Quantum error correction for quantum memories," *Rev. Mod. Phys.*, vol. 87, pp. 307–346, Apr 2015. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.87.307

[24] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[25] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, "Quantum computing with qiskit," 2024. [Online]. Available: https://arxiv.org/abs/2405.08810

[26] B. Lienhard, A. Vepsäläinen, L. C. Govia, C. R. Hoffer, J. Y. Qiu, D. Ristè, M. Ware, D. Kim, R. Winik, A. Melville, B. Niedzielski, J. Yoder, G. J. Ribeill, T. A. Ohki, H. K. Krovi, T. P. Orlando, S. Gustavsson, and W. D. Oliver, "Deep-neural-network discrimination of multiplexed superconducting-qubit states," *Phys. Rev. Appl.*, vol. 17, p. 014024, Jan 2022. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevApplied.17.014024

[27] H. Zhang, K. Yin, A. Wu, H. Shapourian, A. Shabani, and Y. Ding, "Mech: Multi-entry communication highway for superconducting quantum chiplets," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ser. ASPLOS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 699–714. [Online]. Available: https://doi.org/10.1145/3620665.3640377

[28] 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, p. 032324, Sep 2012. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.86.032324

[29] P. D. Nation, H. Kang, N. Sundaresan, and J. M. Gambetta, "Scalable mitigation of measurement errors on quantum computers," *PRX Quantum*, vol. 2, p. 040326, Nov 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PRXQuantum.2.040326

[30] Y. Yang, Z. Shen, X. Zhu, Z. Wang, G. Zhang, J. Zhou, X. Jiang, C. Deng, and S. Liu, "Fpga-based electronic system for the control and readout of superconducting quantum processors," *Review of Scientific Instruments*, vol. 93, no. 7, p. 074701, 07 2022. [Online]. Available: https://doi.org/10.1063/5.0085467

[31] N. Messaoudi, C. Crocker, and M. Almendros, "A hardware-accelerated qubit control system for quantum information processing," in *2020 XXXV Conference on Design of Circuits and Integrated Systems (DCIS)*. Los Alamitos, CA, USA: IEEE, Nov 2020, pp. 1–5.

[32] J. Heinsoo, C. K. Andersen, A. Remm, S. Krinner, T. Walter, Y. Salathé, S. Gasparinetti, J.-C. Besse, A. Potočnik, A. Wallraff, and C. Eichler, "Rapid high-fidelity multiplexed readout of superconducting qubits," *Phys. Rev. Appl.*, vol. 10, p. 034040, Sep 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevApplied.10.034040

[33] R. Acharya, S. Brebels, A. Grill, J. Verjauw, T. Ivanov, D. P. Lozano, D. Wan, J. Van Damme, A. Vadiraj, M. Mongillo *et al.*, "Multiplexed superconducting qubit control at millikelvin temperatures with a low-power cryo-cmos multiplexer," *Nature Electronics*, vol. 6, no. 11, pp. 900–909, 2023.

[34] G. Di Guglielmo, B. Du, J. Campos, A. Boltasseva, A. V. Dixit, F. Fahim, Z. Kudyshev, S. Lopez, R. Ma, G. N. Perdue, N. Tran, O. Yesilyurt, and D. Bowring, "End-to-end workflow for machine-learning-based qubit readout

with qick and hls4ml," *IEEE Transactions on Quantum Engineering*, 2025, arXiv:2501.14663.

[35] R. Delaney, M. Urmey, S. Mittal, B. Brubaker, J. Kindem, P. Burns, C. Regal, and K. Lehnert, "Superconducting-qubit readout via low-backaction electro-optic transduction," *Nature*, vol. 606, no. 7914, pp. 489–493, 2022.

[36] E. Magesan, J. M. Gambetta, A. D. Córcoles, and J. M. Chow, "Machine learning for discriminating quantum measurement trajectories and improving readout," *Phys. Rev. Lett.*, vol. 114, p. 200501, May 2015. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.114.200501

[37] R. B. Staszewski, I. Bashir, E. Blokhina, and D. Leipold, "Cryo-cmos for quantum system on-chip integration: Quantum computing as the development driver," *IEEE Solid-State Circuits Magazine*, vol. 13, no. 2, pp. 46–53, 2021.

[38] S. Krinner, S. Storz, P. Kurpiers, P. Magnard, J. Heinsoo, R. Keller, J. Luetolf, C. Eichler, and A. Wallraff, "Engineering cryogenic setups for 100-qubit scale superconducting circuit systems," *EPJ Quantum Technology*, vol. 6, no. 1, p. 2, 2019.

[39] S. O. Valenzuela, W. D. Oliver, D. M. Berns, K. K. Berggren, L. S. Levitov, and T. P. Orlando, "Microwave-induced cooling of a superconducting qubit," *Science*, vol. 314, no. 5805, pp. 1589–1592, 2006.

[40] IBM Quantum, "IBM Quantum delivers on performance challenge made two years ago," https://www.ibm.com/quantum/blog/qdc-2024, 2024, accessed: 2025-08-20.

[41] J. Ang, G. Carini, Y. Chen, I. Chuang, M. Demarco, S. Economou, A. Eickbusch, A. Faraon, K.-M. Fu, S. Girvin, M. Hatridge, A. Houck, P. Hilaire, K. Krsulich, A. Li, C. Liu, Y. Liu, M. Martonosi, D. McKay, J. Misewich, M. Ritter, R. Schoelkopf, S. Stein, S. Sussman, H. Tang, W. Tang, T. Tomesh, N. Tubman, C. Wang, N. Wiebe, Y. Yao, D. Yost, and Y. Zhou, "Arquin: Architectures for multinode superconducting quantum computers," *ACM Transactions on Quantum Computing*, vol. 5, no. 3, Sep. 2024. [Online]. Available: https://doi.org/10.1145/3674151

[42] D. Barral, F. J. Cardama, G. Díaz-Camacho, D. Faílde, I. F. Llovo, M. Mussa-Juane, J. Vázquez-Pérez, J. Villasuso, C. Piñeiro, N. Costas, J. C. Pichel, T. F. Pena, and A. Gómez, "Review of distributed quantum computing: From single qpu to high performance quantum computing," *Computer Science Review*, vol. 57, p. 100747, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013725000231

[43] W. Kozlowski and S. Wehner, "Towards large-scale quantum networks," in *Proceedings of the Sixth Annual ACM International Conference on Nanoscale Computing and Communication*, ser. NANOCOM '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3345312.3345497

[44] A. Zeilinger, "Quantum teleportation," *Scientific American*, vol. 282, no. 4, pp. 50–59, 2000.

[45] S. Pirandola, J. Eisert, C. Weedbrook, A. Furusawa, and S. L. Braunstein, "Advances in quantum teleportation," *Nature photonics*, vol. 9, no. 10, pp. 641–652, 2015.

[46] H. Kang, J. F. Kam, G. J. Mooney, and L. C. Hollenberg, "Teleporting two-qubit entanglement across 19 qubits on a superconducting quantum computer," *Phys. Rev. Appl.*, vol. 23, p. 014057, Jan 2025. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevApplied.23.014057

[47] J.-B. Waring, S. L. Beux, and C. Pere, "Robust ghz state preparation via majority-voted boundary measurements," 2026. [Online]. Available: https://arxiv.org/abs/2602.19405

[48] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge: Cambridge University Press, 2010.

[49] J. Roffe, "Quantum error correction: an introductory guide," *Contemporary Physics*, vol. 60, no. 3, p. 226–245, Jul. 2019. [Online]. Available: http://dx.doi.org/10.1080/00107514.2019.1667078

[50] F. Battistel, C. Chamberland, K. Johar, R. W. J. Overwater, F. Sebastiano, L. Skoric, Y. Ueno, and M. Usman, "Real-time decoding for fault-tolerant quantum computing: progress, challenges and outlook," *Nano Futures*, vol. 7, no. 3, p. 032003, Aug. 2023. [Online]. Available: http://dx.doi.org/10.1088/2399-1984/aceba6

[51] R. Harper, C. Lainé, E. Hockings, C. McLauchlan, G. M. Nixon, B. J. Brown, and S. D. Bartlett, "Characterising the failure mechanisms of error-corrected quantum logic gates," *Nature Communications*, vol. 17, p. 5039, 2026, arXiv:2504.07258.

[52] T. Picot, A. Lupaşcu, S. Saito, C. J. P. M. Harmans, and J. E. Mooij, "Role of relaxation in the quantum measurement of a superconducting qubit using a nonlinear oscillator," *Phys. Rev. B*, vol. 78, p. 132508, Oct 2008. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.78.132508

[53] T. Thorbeck, Z. Xiao, A. Kamal, and L. C. G. Govia, "Readout-induced suppression and enhancement of superconducting qubit lifetimes," *Phys. Rev. Lett.*, vol. 132, p. 090602, Feb 2024. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.132.090602

[54] F. Mallet, F. R. Ong, A. Palacios-Laloy, F. Nguyen, P. Bertet, D. Vion, and D. Esteve, "Single-shot qubit readout in circuit quantum electrodynamics," *Nature Physics*, vol. 5, no. 11, pp. 791–795, 2009.

[55] C. A. Ryan, B. R. Johnson, J. M. Gambetta, J. M. Chow, M. P. da Silva, O. E. Dial, and T. A. Ohki, "Tomography via correlation of noisy measurement records," *Phys. Rev. A*, vol. 91, p. 022118, Feb 2015. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.91.022118

[56] Y. Chen, D. Sank, P. O'Malley, T. White, R. Barends, B. Chiaro, J. Kelly, E. Lucero, M. Mariantoni, A. Megrant, C. Neill, A. Vainsencher, J. Wenner, Y. Yin, A. N. Cleland, and J. M. Martinis, "Multiplexed dispersive readout of superconducting phase qubits," *Applied Physics Letters*, vol. 101, no. 18, p. 182601, 11 2012. [Online]. Available: https://doi.org/10.1063/1.4764940

[57] T. Walter, P. Kurpiers, S. Gasparinetti, P. Magnard, A. Potočnik, Y. Salathé, M. Pechal, M. Mondal, M. Oppliger, C. Eichler, and A. Wallraff, "Rapid high-fidelity single-shot dispersive readout of superconducting qubits," *Phys. Rev. Appl.*, vol. 7, p. 054020, May 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevApplied.7.054020

[58] Z. Chen, J. Kelly, C. Quintana, R. Barends, B. Campbell, Y. Chen, B. Chiaro, A. Dunsworth, A. G. Fowler, E. Lucero, E. Jeffrey, A. Megrant, J. Mutus, M. Neeley, C. Neill, P. J. J. O'Malley, P. Roushan, D. Sank, A. Vainsencher, J. Wenner, T. C. White, A. N. Korotkov, and J. M. Martinis, "Measuring and suppressing quantum state leakage in a superconducting qubit," *Phys. Rev. Lett.*, vol. 116, p. 020501, Jan 2016. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.116.020501

[59] C. Macklin, K. O'Brien, D. Hover, M. E. Schwartz, V. Bolkhovsky, X. Zhang, W. D. Oliver, and I. Siddiqi, "A near–quantum-limited josephson traveling-wave parametric amplifier," *Science*, vol. 350, no. 6258, pp. 307–310, 2015. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.aaa8525

[60] E. A. Sete, J. M. Martinis, and A. N. Korotkov, "Quantum theory of a bandpass purcell filter for qubit readout," *Phys. Rev. A*, vol. 92, p. 012325, Jul 2015. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.92.012325

[61] W. Tian, L. Lu, S. Tan, Y. Liang, T. Li, K. Zhou, X. Jia, and J. Yin, "Artery: Fast quantum feedback using branch prediction," in *Proceedings of the 52nd Annual International Symposium on Computer Architecture*, ser. ISCA '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 285–298. [Online]. Available: https://doi.org/10.1145/3695053.3731086

[62] Google Quantum AI, "Suppressing quantum errors by scaling a surface code logical qubit," *Nature*, vol. 614, no. 7949, pp. 676–681, 2023.

[63] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, Oct 2019. [Online]. Available: https://doi.org/10.1038/s41586-019-1666-5

[64] S. Maurya and S. Tannu, "Synchronization for fault-tolerant quantum computers," in *Proceedings of the 52nd Annual International Symposium on Computer Architecture*, ser. SIGARCH '25. New York, NY, USA: ACM, Jun. 2025, p. 1370–1385. [Online]. Available: http://dx.doi.org/10.1145/3695053.3730991

[65] O. Higgott and C. Gidney, "Sparse blossom: correcting a million errors per core second with minimum-weight matching," *Quantum*, vol. 9, p. 1600, Jan. 2025. [Online]. Available: http://dx.doi.org/10.22331/q-2025-01-20-1600

[66] Y. Chen and Y. Stade, "Quantum constant propagation," in *Static Analysis*, M. V. Hermenegildo and J. F. Morales, Eds. Cham: Springer Nature Switzerland, 2023, pp. 164–189.

[67] Y. Chen, I. Fulginiti, and C. B. Mendl, "Reducing mid-circuit measurements via probabilistic circuits," in *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 01. Los Alamitos, CA, USA: IEEE, 2024, pp. 952–958.

[68] ——, "Optimization framework for reducing mid-circuit measurements and resets," in *Computational Science – ICCS 2025 Workshops*, M. Paszynski, A. S. Barnard, and Y. J. Zhang, Eds. Cham: Springer Nature Switzerland, 2025, pp. 150–164.

[69] J. M. Günther, F. Tacchino, J. R. Wootton, I. Tavernelli, and P. K. Barkoutsos, "Improving readout in quantum simulations with repetition codes," *Quantum Science and Technology*, vol. 7, no. 1, p. 015009, nov 2021. [Online]. Available: https://doi.org/10.1088/2058-9565/ac3386

[70] J. R. Wootton and D. Loss, "Repetition code of 15 qubits," *Phys. Rev. A*, vol. 97, p. 052313, May 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.97.052313

[71] FastML Team, "hls4ml concepts," https://fastmachinelearning.org/hls4ml/api/concepts.html, 2025, accessed: 2025-08-20.

[72] J. Duarte *et al.*, "Fast inference of deep neural networks in FPGAs for particle physics," *JINST*, vol. 13, no. 07, p. P07027, 2018.

[73] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, "Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels," *Phys. Rev. Lett.*, vol. 70, pp. 1895–1899, Mar 1993. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.70.1895

[74] D. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, "Surface code quantum computing by lattice surgery," *New Journal of Physics*, vol. 14, no. 12, p. 123011, dec 2012. [Online]. Available: https://dx.doi.org/10.1088/1367-2630/14/12/123011

[75] Qiskit, "Qiskit Hellinger Fidelity," https://docs.quantum.ibm.com/api/qiskit/0.31/qiskit.quantum_info.hellinger_fidelity, 2021, accessed: 2025-05-05.

[76] A. Javadi-Abhari, S. Martiel, A. Seif, M. Takita, and K. X. Wei, "Big cats: entanglement in 120 qubits and beyond," 2025. [Online]. Available: https://arxiv.org/abs/2510.09520

[77] E. Bäumer, V. Tripathi, A. Seif, D. Lidar, and D. S. Wang, "Quantum fourier transform using dynamic circuits," *Phys. Rev. Lett.*, vol. 133, p. 150602, Oct 2024. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.133.150602

[78] S. V. Barron, D. J. Egger, E. Pelofske, A. Bärtschi, S. Eidenbenz, M. Lehmkuehler, and S. Woerner, "Provable bounds for noise-free expectation values computed from noisy samples," *Nature Computational Science*, vol. 4, no. 11, pp. 865–875, 2024.

[79] M. Motta, G. O. Jones, J. E. Rice, T. P. Gujarati, R. Sakuma, I. Liepuoniute, J. M. Garcia, and Y.-y. Ohnishi, "Quantum chemistry simulation of ground- and excited-state properties of the sulfonium cation on a superconducting quantum processor," *Chemical Science*, vol. 14, no. 11, pp. 2915–2927, 2023.

[80] I. Quantum. (2025) Development & innovation roadmap. IBM. [Online]. Available: http://www.ibm.com/downloads/documents/us-en/131cf87ab63319bf

[81] S. Tannu, P. Das, R. Ayanzadeh, and M. Qureshi, "Hammer: Boosting fidelity of noisy quantum circuits by exploiting hamming behavior of erroneous outcomes," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 529–540. [Online]. Available: https://doi.org/10.1145/3503222.3507703

[82] E. van den Berg, Z. K. Minev, and K. Temme, "Model-free readout-error mitigation for quantum expectation values," *Phys. Rev. A*, vol. 105, p. 032620, Mar 2022. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.105.032620

[83] S. Bravyi, S. Sheldon, A. Kandala, D. C. Mckay, and J. M. Gambetta, "Mitigating measurement errors in multiqubit experiments," *Phys. Rev. A*, vol. 103, p. 042605, Apr 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.103.042605

[84] J. Kim, B. Oh, Y. Chong, E. Hwang, and D. K. Park, "Quantum readout error mitigation via deep learning," *New Journal of Physics*, vol. 24, no. 7, p. 073009, jul 2022. [Online]. Available: https://dx.doi.org/10.1088/1367-2630/ac7b3d

[85] A. W. R. Smith, K. E. Khosla, C. N. Self, and M. S. Kim, "Qubit readout error mitigation with bit-flip averaging," *Science Advances*, vol. 7, no. 47, p. eabi8009, 2021. [Online]. Available: https://www.science.org/doi/abs/10.1126/sciadv.abi8009

[86] B. Yang, R. Raymond, and S. Uno, "Efficient quantum readout-error mitigation for sparse measurement outcomes of near-term quantum devices," *Phys. Rev. A*, vol. 106, p. 012423, Jul 2022. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.106.012423

[87] A. S. Aasen, A. Di Giovanni, H. Rotzinger, A. V. Ustinov, and M. Gärttner, "Readout error mitigated quantum state tomography tested on superconducting qubits," *Communications Physics*, vol. 7, no. 1, p. 301, 2024.

[88] F. B. Maciejewski, Z. Zimborás, and M. Oszmaniec, "Mitigation of readout noise in near-term quantum devices by classical post-processing based on detector tomography," *Quantum*, vol. 4, p. 257, 2020.

[89] X. Guo, T. Bunarjyan, D. Liu, B. Lienhard, and M. Schulz, "Klinq: Knowledge distillation-assisted lightweight neural network for qubit readout on fpga," in *2025 62nd ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2025, arXiv:2503.03544.

[90] C. N. Mude, S. Maurya, B. Lienhard, and S. Tannu, "Efficient and scalable architectures for multi-level superconducting qubit readout," in *2025 62nd ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2025, arXiv:2405.08982.

[91] E. Giortamis, A. Świerkowska, S. Stankovic, F. Gust, B. Lienhard, and P. Bhatotia, "Oraqle: An empirical analysis of qubit readout and discriminators in quantum error correction," 2026. [Online]. Available: https://arxiv.org/abs/2608.01939

[92] P. Escofet, A. Ovide, M. Bandic, L. Prielinger, H. van Someren, S. Feld, E. Alarcon, S. Abadal, and C. Almudever, "Revisiting the mapping of quantum circuits: Entering the multi-core era," *ACM Transactions on Quantum Computing*, vol. 6, no. 1, Jan. 2025. [Online]. Available: https://doi.org/10.1145/3655029

[93] H. Zhang, Y. Xu, H. Hu, K. Yin, H. Shapourian, J. Zhao, R. R. Kompella, R. Nejabati, and Y. Ding, "Optimizing quantum communication for quantum data centers with reconfigurable networks," 2024. [Online]. Available: https://arxiv.org/abs/2412.04837

[94] W. Cambiucci, R. M. Silveira, and W. V. Ruggiero, "Hypergraphic partitioning of quantum circuits for distributed quantum computing," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 02. Los Alamitos, CA, USA: IEEE, Sep. 2023, pp. 268–269.

[95] P. Andrés-Martínez and C. Heunen, "Automated distribution of quantum circuits via hypergraph partitioning," *Phys. Rev. A*, vol. 100, p. 032308, Sep 2019. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.100.032308

[96] J. M. Baker, C. Duckering, A. Hoover, and F. T. Chong, "Time-sliced quantum circuit partitioning for modular architectures," in *Proceedings of the 17th ACM International Conference on Computing Frontiers*, ser. CF '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 98–107. [Online]. Available: https://doi.org/10.1145/3387902.3392617

[97] D. Dadkhah, M. Zomorodi, S. E. Hosseini, P. Plawiak, and X. Zhou, "Reordering and partitioning of distributed quantum circuits," *IEEE Access*, vol. 10, pp. 70 329–70 341, 2022.

[98] Z. Davarzani, M. Zomorodi-Moghadam, M. Houshmand, and M. Nouri-Baygi, "A dynamic programming approach for distributing quantum circuits by bipartite graphs," *Quantum Information Processing*, vol. 19, pp. 1–18, 2020.

[99] S. DiAdamo, M. Ghibaudi, and J. Cruise, "Distributed quantum computing and network control for accelerated vqe," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1–21, 2021.

[100] D. Ferrari, A. S. Cacciapuoti, M. Amoretti, and M. Caleffi, "Compiler design for distributed quantum computing," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1–20, 2021.

[101] T. Häner, D. S. Steiger, T. Hoefler, and M. Troyer, "Distributed quantum computing with qmpi," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3458817.3476172

[102] N. Tornow, E. Giortamis, and P. Bhatotia, "Qvm: Quantum gate virtualization machine," *Proc. ACM Program. Lang.*, vol. 9, no. PLDI, Jun. 2025. [Online]. Available: https://doi.org/10.1145/3729290

[103] E. Giortamis, F. Romão, N. Tornow, and P. Bhatotia, "Qos: quantum operating system," in *Proceedings of the 19th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI '25. USA: USENIX Association, 2025.

[104] E. Giortamis, F. Romao, N. Tornow, D. Lugovoy, and P. Bhatotia, "Qonductor: A cloud orchestrator for quantum computing," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 728–745. [Online]. Available: https://doi.org/10.1145/3712285.3759785

[105] N. Tornow, E. Giortamis, D. Sprokholt, C. Mendl, and P. Bhatotia, "qTPU: Hybrid tensor networks for Quantum-Classical acceleration," in *20th USENIX Symposium on Operating Systems Design and Implementation (OSDI 26)*. Seattle, WA: USENIX Association, Jul. 2026, pp. 2367–2388. [Online]. Available: https://www.usenix.org/conference/osdi26/presentation/tornow

[106] A. Paler, I. Polian, K. Nemoto, and S. J. Devitt, "Fault-tolerant, high-level quantum circuits: form, compilation and description," *Quantum Science and Technology*, vol. 2, no. 2, p. 025003, Apr. 2017. [Online]. Available: http://dx.doi.org/10.1088/2058-9565/aa66eb

[107] D. Herr, F. Nori, and S. J. Devitt, "Lattice surgery translation for quantum computation," *New Journal of Physics*, vol. 19, no. 1, p. 013034, Jan. 2017. [Online]. Available: http://dx.doi.org/10.1088/1367-2630/aa5709

[108] G. Watkins, H. M. Nguyen, K. Watkins, S. Pearce, H.-K. Lau, and A. Paler, "A high performance compiler for very large scale surface code computations," *Quantum*, vol. 8, p. 1354, May 2024. [Online]. Available: http://dx.doi.org/10.22331/q-2024-05-22-1354

[109] T. Leblond, R. S. Bennink, J. G. Lietz, and C. M. Seck, "Tiscc: A surface code compiler and resource estimator for trapped-ion processors," in *Proceedings of the SC '23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*, ser. SC-W 2023. New York, NY, USA: ACM, Nov. 2023, p. 1426–1435. [Online]. Available: http://dx.doi.org/10.1145/3624062.3624214

[110] S. Liu, N. Benchasattabuse, D. Q. Morgan, M. Hajdušek, S. J. Devitt, and R. Van Meter, "A substrate scheduler for compiling arbitrary fault-tolerant graph states," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Los Alamitos, CA, USA: IEEE, Sep. 2023, p. 870–880. [Online]. Available: http://dx.doi.org/10.1109/QCE57702.2023.00101

[111] M. Beverland, V. Kliuchnikov, and E. Schoute, "Surface code compilation via edge-disjoint paths," *PRX Quantum*, vol. 3, no. 2, May 2022. [Online]. Available: http://dx.doi.org/10.1103/PRXQuantum.3.020342

[112] Y. Ding, A. Holmes, A. Javadi-Abhari, D. Franklin, M. Martonosi, and F. T. Chong, "Magic-state functional units: mapping and scheduling multi-level distillation circuits for fault-tolerant quantum architectures," in *Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-51. Los Alamitos, CA, USA: IEEE Press, 2018, p. 828–840. [Online]. Available: https://doi.org/10.1109/MICRO.2018.00072

[113] A. Świerkowska, J. Pflieger, E. Giortamis, and P. Bhatotia, "Eccentric: An empirical analysis of quantum error correction codes," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 10, no. 2, May 2026. [Online]. Available: https://doi.org/10.1145/3805635

APPENDIX A
ARTIFACT APPENDIX

A. Abstract

The MCMit artifact contains the source code, evaluation drivers, result CSVs, and plotting scripts behind every figure and table of the paper. It provides **(1)** the FPGA controller implementing the `branch_reduce_fproc` constant-latency multi-qubit branch instruction, as three forks of Qubic 2.0 [19] (software, distributed processor, and gateware); **(2)** the MCMit-CNN and MCMit-T qubit-state discriminators with their training, crosstalk, and trace-truncation pipelines, plus reproductions of the baseline FNN [26], QubiCML [22], and HERQULES [21]; **(3)** the software error-mitigation compiler passes (static MCM elimination, measurement hardening, and stochastic branching); and **(4)** the vendored `lattice-sim` surface-code simulator and MECH [27] chiplet compiler used for the motivation and QEC studies. Together these reproduce Fig. 2, Fig. 3, Figs. 10–13, and Tables II–VIII.

B. Artifact check-list (meta-information)

- **Algorithm:** constant-latency multi-qubit branch reduction (XOR / majority vote); CNN- and transformer-based qubit-state discrimination; quantum constant propagation for static MCM elimination; measurement hardening; stochastic branching; surface-code lattice simulation.
- **Program:** Python (Qiskit, PyTorch), Verilog/SystemVerilog HDL, Jupyter notebooks.
- **Compilation:** none for the Python stack; Vivado 2022.1 for the Qubic gateware; `hls4ml` for the discriminator FPGA synthesis.
- **Transformations:** Qiskit compiler passes for dynamic circuit simplification, measurement hardening, and stochastic branching.
- **Binary:** none shipped; FPGA bitstreams are built from source.
- **Model:** MCMit-CNN and MCMit-T, plus the baseline FNN, QubiCML, and HERQULES reproductions.
- **Data set:** 1.6M experimentally extracted five-qubit readout traces (1μs, 500 MSamples/sec,{I,Q}) from Lienhard et al. [26]; IBM Heron calibration data; the MECH QFT/QAOA/VQE/BV benchmarks.
- **Run-time environment:** Linux (Ubuntu 22.04), Python 3.9+, `pip`/`venv`, Docker (for `lattice-sim`), and an optional IBM Quantum account for `ibm_fez`.
- **Hardware:** x86-64 server; an NVIDIA GPU for discriminator training; a Xilinx Alveo XCU280 for FPGA resource utilization; QPU access for the hardware fidelity numbers (a simulator fallback is provided).
- **Run-time state:** the backend calibration cycle in use, for the hardware experiments.
- **Execution:** Python entry points and notebooks, run from the repository root; no exclusive machine access required.
- **Metrics:** latency (ns), discrimination accuracy, FPGA resource utilization (DSP/BRAM/FF/LUT), Hellinger fidelity, logical error rate, and compilation runtime.
- **Output:** CSV files under `results/`, PDF figures, and stdout tables; reference copies of every CSV are included.
- **Experiments:** one entry point per paper result, listed in `evaluation/README.md`.
- **How much disk space required (approximately)?:** $\sim$2 GB for the repository and the Python/Docker environments; $\sim$10 GB including the readout-trace dataset.
- **How much time is needed to prepare workflow (approximately)?:** $\sim$1 hour for the Python and Docker environments; several hours more if the FPGA gateware is rebuilt in Vivado.
- **How much time is needed to complete experiments (approximately)?:** $\sim$1 day for the full set: a few hours for the simulation-only figures, hours on a GPU for the discriminator sweeps, with the d=11 QEC sweeps and the 25-qubit long-range CNOT elimination ($\sim$38 min, Table VIII) the long poles.
- **Publicly available?:** Yes.
- **Code licenses (if publicly available)?:** CC BY 4.0; each vendored subproject retains its own upstream license.
- **Data licenses (if publicly available)?:** the readout-trace dataset is not redistributed and is obtained from the authors of [26]; the result CSVs are CC BY 4.0.
- **Workflow automation framework used?:** none; plain Python entry points, Jupyter notebooks, and Docker Compose.
- **Archived (provide DOI)?:** https://doi.org/10.5281/zenodo.21485912

C. Description

1) How to access: The artifact is archived on Zenodo at https://doi.org/10.5281/zenodo.21485912 (32.5 MB archive; the concept DOI `10.5281/zenodo.21358489` always resolves to the latest version). The same content is on GitHub at https://github.com/manosgior/MCMit, tag `v3`. The top-level `README.md` maps every directory to the paper section it implements, and `evaluation/README.md` gives one command per figure or table.

2) Hardware dependencies: An x86-64 Linux server suffices for the compiler, simulation, and plotting experiments. The discriminator training and FPGA numbers additionally need the hardware of §VIII: a 64-core Intel Xeon Gold 6326 with 314 GB RAM and an NVIDIA A40 GPU (48 GB VRAM) for training, and a Xilinx Alveo XCU280 for Table VII; the Qubic gateware builds for a ZCU216 RFSoC target. Fig. 11 was measured on `ibm_fez`, which needs an IBM Quantum account; the scripts otherwise fall back to a noisy Aer simulator.

3) Software dependencies: MCMit's own code (`applications/`, `backends/`, `compiler/`, `evaluation/`, `plotting/`) pins its dependencies in the top-level `requirements.txt`: Qiskit 2.1.0, Qiskit Aer 0.17.1, `qiskit-ibm-runtime` 0.40.1, `mthree` 3.0.0, NumPy 2.3.1, pandas 2.3.0, NetworkX 3.5, Matplotlib 3.10.3, and seaborn 0.13.2. Each vendored subproject pins its own environment separately, since they are independent tools: `lattice-sim/` (Stim 1.14.0, PyMatching 2.2.1, sinter 1.14.0), `qubit_state_discrimination/` (PyTorch, `h5py`, scikit-learn, Optuna), `evaluation/motivation/MECH/` (Qiskit 0.45.1, Python $\geq$3.9.6), and `branch_instruction/qubic-software/pyproject.toml`. The discriminators use PyTorch v1.7.0, TorchVision v0.8.1, and `hls4ml` v1.2.0; the gateware build needs Vivado 2022.1.

4) Data sets: The discriminator tables need the raw readout-trace dataset of Lienhard et al. [26]: 1.6M traces, 50,000 per each of the 2^5 basis states, each 1μs long at 500 MSamples/sec. It is **not** redistributed in the artifact for size and policy reasons; the `data/five_qubit_data/` directory holds placeholders, and the scripts expect the real HDF5 files bind-mounted at `/data/five_qubit_data`. The backend calibration data and the MECH benchmarks are included in the repository.

5) Models: `qubit_state_discrimination/networks/` holds all five designs of Table III: `CNN.py`

(MCMit-CNN), `TransformerMF.py` (MCMit-T), `SingleQubitFNN.py` (baseline FNN), `Qubic.py` (QubiCML), and `HERQULES.py`. The subproject README maps each design to its trainer.

D. Installation

```
git clone --branch v3 \
    https://github.com/manosgior/MCMit.git
cd MCMit
python3 -m venv venv && source venv/bin/activate
pip install -r requirements.txt
```

This covers every top-level driver. The vendored subprojects are installed on demand from their own `requirements.txt`; `lattice-sim` instead ships a container (`docker build -t $USER/lattice-sim .`, then `docker-compose run app`). Run everything from the repository root.

E. Experiment workflow

Each paper result has exactly one entry point, tabulated in `evaluation/README.md`; results land as CSVs in `results/`, which `plotting/plot.py` and the subproject notebooks render.

- **Fig. 2** (§III-B): `MECH_line_plot.ipynb` and `python motivation/MECH/export_fig2_csv.py`.
- **Fig. 3** (§III-C): `python lattice-sim/evaluate_mcm_error.py --preset google` and `evaluate_mcm_latency.py --preset google`.
- **Table II** (§VIII-B): the cocotb testbenches and Vivado build under `branch_instruction/`, on the `branch-reduce` branch of the three Qubic forks.
- **Fig. 10** (§VIII-B): `python -m evaluation.evaluate_feedback_latency` (16-qubit simulation) and `python -m evaluation.evaluate_decoherence` (32-qubit analytic).
- **Tables III–VI** (§VIII-C): the per-table runners under `qubit_state_discrimination/runners/`, e.g. `CNN_LEN={125,250,375} python runners/_cnn_length_sweep.py`for Table IV and `python runners/_xtalk_cnn.py` for Tables V and VI.
- **Fig. 12** (§VIII-C): `python -m evaluation.evaluate_readout_duration`.
- **Table VII** (§VIII-C): `hls4ml` synthesis of the trained networks, 8-bit quantization at a 500 MHz clock, targeting the XCU280.
- **Fig. 11** (§VIII-D): `python -m evaluation.evaluate_software_mitigation --benchmark all --max-n 25`.
- **Table VIII** (§VII-A): `python mcm_count.py` and `branching_merged.py` in `compiler/removal/`.
- **Fig. 13** (§VIII-E): the `lattice-sim/` runner scripts, rendered by `lattice-sim/plotting/_render_four_panel.py`.

F. Evaluation and expected results

1) Steps for evaluation: **(1)** Install the top-level environment as above. **(2)** Run the simulation-only drivers — `evaluate_feedback_latency`, `evaluate_decoherence`, `evaluate_readout_duration`, and `evaluate_software_mitigation` — which need nothing beyond the pinned Python environment; this reproduces Figs. 10, 11, and 12. **(3)** Run `compiler/removal/mcm_count.py` and `branching_merged.py` for Table VIII. **(4)** Build the `lattice-sim` container and run its presets for Fig. 3 and Fig. 13, and the MECH notebook for Fig. 2. **(5)** With the readout-trace dataset and a GPU, run the discriminator runners for Tables III–VI. **(6)** Render the figures with `plotting/plot.py` and the subproject notebooks, and diff the regenerated CSVs against the reference copies shipped in `results/`, which `results/README.md` maps file-by-file to the figure each one feeds.

2) Results: The artifact supports the paper's four main claims. **(1) Classical feedback** (Table II, Fig. 10): `branch_reduce_fproc` holds branch latency constant where Qubic's grows with the operand count — up to $32\times$ lower branch latency and 70% lower feedback latency at 32 qubits — yielding 57.3% and 37.8% higher fidelity for the constant-depth GHZ and long-range CNOT benchmarks and $\sim 7\times$ lower effective depth on MECH. **(2) Qubit-state discrimination** (Tables III–VII, Fig. 12): both discriminators reach $F_{5Q} = 0.911$ at the full 1μs trace, and MCMit-CNN holds $F_{5Q} = 0.812$ at 250 ns — 62% and 12% above HERQULES and QubiCML, the former collapsing to near chance (0.511) — at 92% and 82.3% lower LUT utilization. **(3) Software mitigation** (Fig. 11, Table VIII): 30.47% and 17.78% higher Hellinger fidelity than raw execution and Qiskit M3 on average, peaking at 27.84% over M3 for constant-depth GHZ. **(4) QEC** (Fig. 13): 1.2–$9.4\times$ lower logical error rates than HERQULES in the current noise regime and up to $41\times$ in the futuristic one, both at a 500 ns readout.

G. Experiment customization

All drivers expose their parameters on the command line. The latency experiments default to Table II's measured values and §VIII's thermal-relaxation model (T_1=170μs, T_2=130μs), all overridable. `evaluate_software_mitigation` takes `--benchmark`, `--max-n`, `--removal`, and `--calibrations`, plus a `--noiseless` sanity mode where every fidelity should be ≈ 1.0. The discriminator sweeps are parameterized by trace length (`CNN_LEN`, `TRACE_LENGTHS`), and `lattice-sim`'s `circuit` class exposes code distance, patch count, and per-gate latencies and error rates.

H. Notes

Three caveats affect bit-level reproducibility. First, the Fig. 11 numbers were measured on `ibm_fez` over 10,000 shots in a single calibration cycle; the driver defaults to a noisy Aer simulator so it runs anywhere, reproducing the qualitative behavior but not the exact values — faithful numbers need real hardware and a fresh M3 calibration of *that* backend, since stochastic branching only helps when the confusion matrix matches the machine. Second, the GHZ and CNOT generators support only odd qubit counts, so the "16-qubit" configuration runs at 17. Third, HERQULES varies significantly across retrainings ($F_{5Q} = 0.905 \pm 0.007$); the `*_deploy/*_pertrace` scripts are the ones matching Table III. Finally, the three Qubic forks are snapshots of their `branch-reduce` branches; their git history and the four unmodified LBL-Qubic submodules a gateware build needs are linked from `branch_instruction/README.md`.

I. Methodology

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