nature portfolio

https://doi.org/10.1038/s41586-024-08449-y

Supplementary information

Quantum error correction below the surface code threshold

In the format provided by the authors and unedited

Supplementary Information for "Quantum error correction below the surface code threshold"

Google Quantum AI and Collaborators
(Dated: December 10, 2024)

Contents

I. Device Characterization

A. Device Information

The experiments in this work are carried out on two separate Willow processors: one 72-qubit processor and one 105-qubit processor. As noted in the main text, these devices have improved operational fidelities compared to the Sycamore processors used in previous works. The T_1 and T_2 coherence times are significantly improved while the single- and two-qubit gate operation times are kept relatively constant. The improvement in T_1 can be attributed to a few critical changes in our device and fabrication. First, by adjusting the capacitor topology and increasing the overall capacitor size from $\sim$30 µm to $\sim$150 µm, we reduced the qubit exposure to lossy materials and interfaces. Second, we made numerous incremental cleanliness improvements by moving from a shared research cleanroom environment to a dedicated fabrication facility with dedicated tooling. To improve T_2, we reduced current noise in our flux bias lines, which is the largest source of qubit dephasing. Finally, we increased the coupling between our qubits and readout resonators, which reduces the probability of measurement induced state transitions.

B. Device Benchmarking

Single-qubit gates are implemented using microwave XY rotations and virtual Z rotations. On the 105-qubit processor, all XY rotations have 25 ns duration, while on the 72-qubit processor, π and $\pi/2$ rotations have 35 ns duration and 18 ns duration respectively. We characterize the single-qubit coherence times and gate performance, with results shown in Fig. S2 and Fig. S3 for the 72-qubit and 105-qubit processor, respectively. We measure all single-qubit gate errors simultaneously using randomized benchmarking [1].

We implement our entangling operations using CZ gates, as is further described in Section II C. Our CZ gates are performed in 42 ns on the 105-qubit processor and 37 ns on the 72-qubit processor. We characterize their performance using simultaneous cross-entropy benchmarking (XEB) [2]. Every CZ gate belongs to a single layer in the surface code circuit, and the CZ gates within each of these layers are benchmarked simultaneously. The central frequencies and performance of our CZ gates are shown in Fig. S4 and Fig. S5 for the 72-qubit and 105-qubit processor, respectively.

Our measurement chain is similar to that of [3], though with Lumped Element Snake Amplifiers (LESAs) [4] replacing the previous impedance matched parametric amplifiers. Resonator and readout chain parameters for the 72-qubit and 105-qubit processor are shown in Fig. S6 (a-d) and Fig. S7 (a-d), respectively. The qubit frequency during readout, readout pulse duration, and readout pulse power are jointly optimized [5] to attain simultaneous high fidelity mid-circuit readout. The resulting readout parameters and average readout fidelity are shown in Fig. S6 (e-i) and Fig. S7 (e-i) for the 72-qubit and 105-qubit processor, respectively.

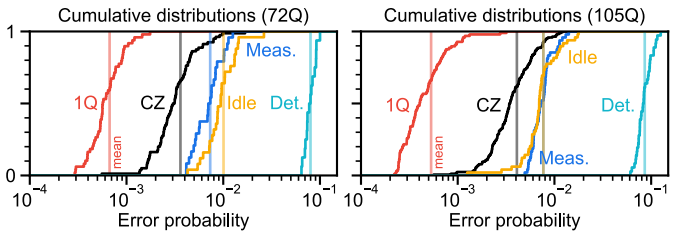

FIG. S1. **Operation error probabilities.** See Fig. 1b. Left: distribution of benchmark results for the 72-qubit processor configured to run the distance-5 surface code. Right: 105-qubit processor configured to run the distance-7 surface code (same as Fig. 1b).

In main text Fig. 1b, we show distributions of operation error probabilities for the 105-qubit processor. In Fig. S1, we repeat Fig. 1b alongside equivalent measurements on the 72-qubit processor.

II. Improvements in Control Techniques

A. Frequency Optimization

We optimize single- and two-qubit gate frequency trajectories via the Snake optimizer [6]. We leverage technology developed in Ref. [7], highlighting two strategies that were especially helpful in boosting data qubit coherence. First, we constructed our optimization model directly around the error correction circuit. This strategy prioritizes data qubit coherence to a much greater extent than optimizing for interleaved layers of single- and two-qubit gates as in cross-entropy benchmarking. Second, we employed intermediate-dimensional optimization up to 9 dimensions. This strategy enables more complex trade-offs between data and measure qubits than lower dimensional optimization, without significantly compromising runtime.

To ensure that qubit coherence remained stable over multiple days of datataking, two stability optimization strategies were included into the Snake optimizer. Both were intended to mitigate catastrophic losses in coherence from frequency collisions with TLS, which are known to have time-dependent variations in their transition frequencies [8]. Both strategies leveraged information extracted from spectrally and temporally resolved qubit T_1 data. The first strategy modified the relaxation error component to penalize single-qubit gate frequencies which historically had reduced T_1. The second strategy sought to prevent frequency collisions with TLS by extracting frequency-temporal trajectories of TLS from historical T_1 data, forecasting them to future times, and excluding these frequencies from allowed single-qubit gate frequencies.

B. Microwave Crosstalk

One requirement for practical quantum computation is the ability to selectively control the states of many qubits simultaneously and with high fidelity. These single-qubit operations are implemented via microwave pulses which must be carefully calibrated and routed through a signal chain onto the processor. One potential error mechanism is the spurious spreading of these microwave signals, which can cause unintentional rotations of other qubits' states and degrade computational performance [9]. This effect, known as microwave crosstalk, must be sufficiently mitigated via hardware or software in order to execute highly performant quantum error correction.

In order to characterize microwave crosstalk on our device, we employ a simple protocol illustrated in Fig. S8. This measurement quantifies the crosstalk experienced by qubit j when driving qubit i, which we parameterize via the crosstalk amplitude r_{ij} and phase θ_{ij} for each pair of qubits on the processor. First, we dynamically tune qubit j, called the target qubit, to the idle frequency of qubit i, called the adversary. Next, we apply a microwave pulse to the control line of the adversary qubit while measuring the response on the target. If this pair experiences appreciable levels of crosstalk, we can quantify the amplitude of the crosstalk via the frequency of the resulting Rabi oscillations, Ω_{adv}. Next, we directly drive the target qubit from its own control line to extract the direct Rabi rate, Ω_{dir}. The ratio of these rates, normalized by their respective drive amplitudes, contains information about the amplitude of the crosstalk r between this target-adversary pair. We also measure the phase of the crosstalk using a protocol described in Fig. S8. Here, we apply simultaneous drives to the adversary and target qubits, with amplitudes a and $r \times a$ respectively. We sweep the relative phase between the two drives and extract the phase θ which produces maximally destructive interference between them. Having measured r_{ij} and θ_{ij}, we can apply a compensation pulse for high-crosstalk pairs to cancel out the effect of crosstalk during single-qubit operations.

The amplitude and phase of the crosstalk between each pair both depend on the qubit frequency. After choosing the grid frequencies for all qubits, we measure the crosstalk amplitude and phase at each qubit's idle frequency. For the 72-qubit processor, we observe that microwave crosstalk is relatively widespread and nonlocal compared to previous device architectures.

C. CZ Error Budget

In this work, we employ diabatic CZ gates, which are implemented by bringing the energies of a qubit pair's $|11\rangle$ and $|20\rangle$ states near resonance while simultaneously increasing their mutual coupling. This is accomplished via flux control of both qubits and their shared coupler [10]. The gate duration t_g and coupling rate g are cho-

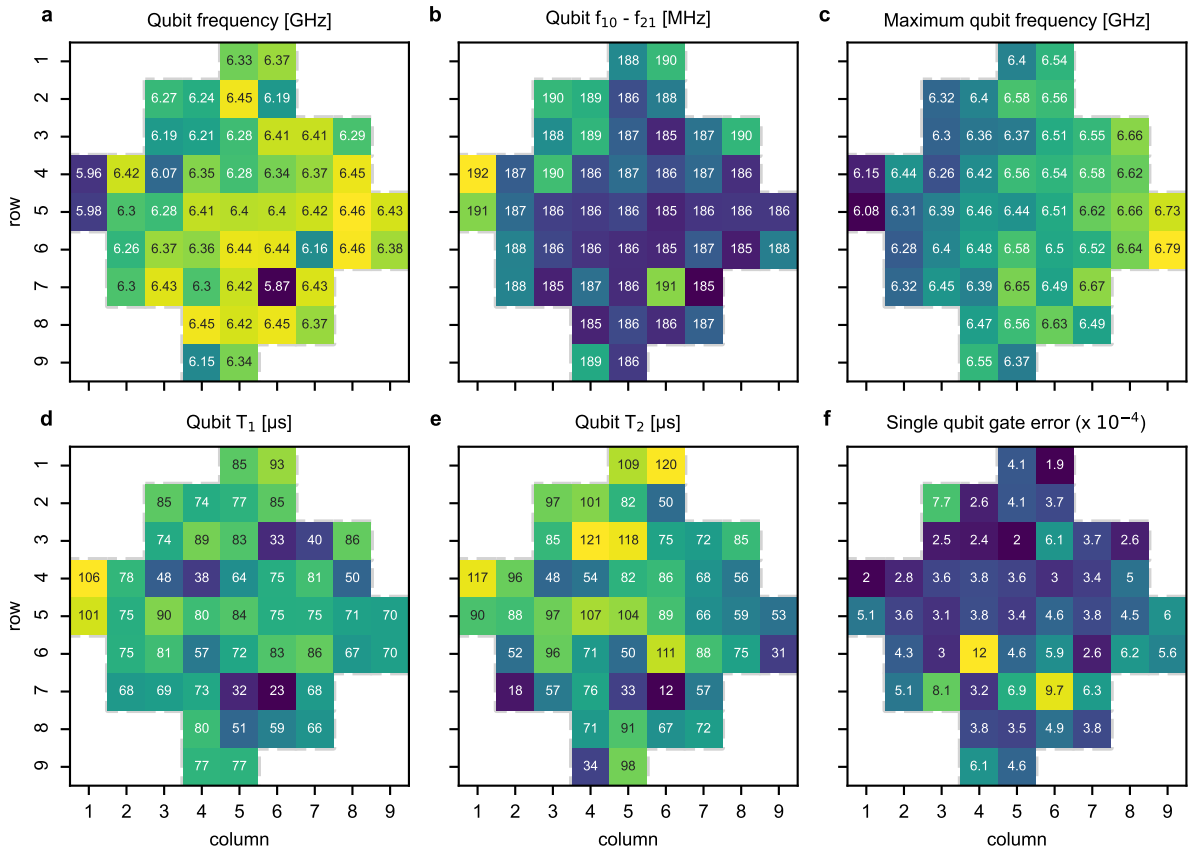

FIG. S2. **Heat map of single-qubit parameters on the 72-qubit processor. a,** Qubit operating frequency during idling and microwave gates. **b,** Qubit anharmonicity. **c,** Maximum qubit frequency. **d,** Qubit relaxation at the operating frequency. **e,** Qubit dephasing at the operating frequency as measured with CPMG echoing ($T_{2,\mathrm{CPMG}}$). **f,** Simultaneous single-qubit gate error as measured by randomized benchmarking.

sen such that at the end of the gate, a conditional phase of π has been acquired in the $|11\rangle$ state with minimal leakage into non-computational states. Given that many different combinations of t_g, g, and flux pulse shape parameters can yield a high fidelity CZ gate, we choose the optimal pulse parameters for our experiment using an empirical waveform optimization procedure. Next, we feed those parameters as inputs into our Snake optimizer, which selects the qubits' idle and interaction frequencies in the full grid [7].

We perform a detailed characterization of two-qubit gate performance on our 72-qubit processor via benchmarking protocols, such as XEB and context aware fidelity estimation (CAFE) [11], as well as careful error budgeting through a variety of experiments using periodic circuits. These measurement circuits include Floquet-style circuits for characterizing leakage [12], "matrix-element amplification using dynamical decoupling" (MEADD) circuits for computational subspace errors [13], and other periodic circuits for budgeting incoherent gate error (similar to Ref. [14]). We find the median CZ Pauli error to be 2.6×10^{-3}. A summary of our error budget can be found in Fig. S9, where we separate leakage errors from coherent and incoherent errors in the computational subspace. In this study, we report medians rather than mean values, due to the relative prevalence of unrepresentative outlier results. We note that our coherent error and leakage error each constitute roughly 10% of our error budget, with the remaining ∼80% of the error consisting of incoherent error. In the future, we believe that we can reduce leakage and coherent error through improved control techniques and substantially reduce the incoherent error through hardware improvements.

III. Decoding

A. List of Decoders

In this work, we use several different decoders depending on the context of the experiment. We use,

1. a neural network decoder introduced in Ref. [16],
2. a matching synthesis decoder Libra recently introduced in Ref. [15] with an ensemble size of 51,

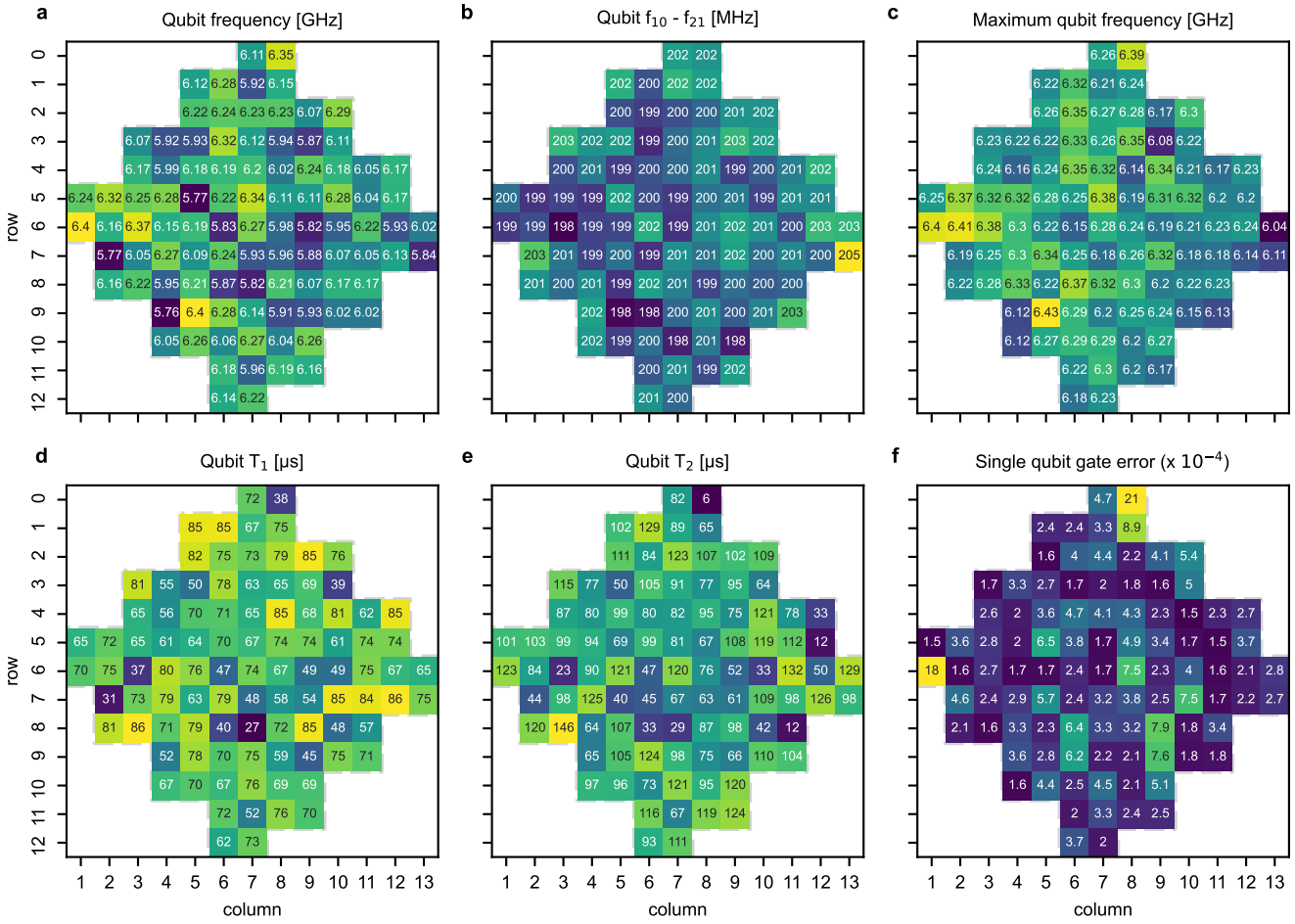

FIG. S3. **Heat map of single-qubit parameters on the 105-qubit processor. a,** Qubit operating frequency during idling and microwave gates. **b,** Qubit anharmonicity. **c,** Maximum qubit frequency. **d,** Qubit relaxation at the operating frequency. **e,** Qubit dephasing at the operating frequency as measured with CPMG echoing ($T_{2,\mathrm{CPMG}}$). **f,** Simultaneous single-qubit gate error as measured by randomized benchmarking.

	Fig. 1d		**Fig. 4c**		
	Libra	Neural network	Real-time	Libra	Neural network
ε_3	$(7.12 \pm 0.06) \times 10^{-3}$	$(6.50 \pm 0.06) \times 10^{-3}$	$(7.1 \pm 0.3) \times 10^{-3}$	$(6.1 \pm 0.2) \times 10^{-3}$	$(5.9 \pm 0.1) \times 10^{-3}$
ε_5	$(3.49 \pm 0.04) \times 10^{-3}$	$(3.03 \pm 0.03) \times 10^{-3}$	$(3.5 \pm 0.1) \times 10^{-3}$	$(2.70 \pm 0.08) \times 10^{-3}$	$(2.69 \pm 0.08) \times 10^{-3}$
ε_7	$(1.71 \pm 0.03) \times 10^{-3}$	$(1.43 \pm 0.03) \times 10^{-3}$			

TABLE S1. **Logical errors per cycle for different decoding schemes.** Here we report the logical error per cycle, ε_d, for different code distances d and different decoding schemes (real-time, Libra [15], and neural network [16]). These correspond to the values plotted in Fig. 1d and Fig. 4c of the main text.

3. an ensembled matching decoder Harmony introduced in Ref. [17] with an ensemble size of 101 (unless otherwise specified),

4. a correlated matching decoder similar to that described in Ref. [18],

5. a real-time decoder with accuracy similar to Ref. [19].

Predictions for these decoders applied to various datasets reported in the paper, along with additional datasets not reported here, can be found at Ref. [20]. We note there are other accurate decoders that would be interesting to test on this data, including beliefmatching [21], BP+OSD [22], and tensor network decoding [3].

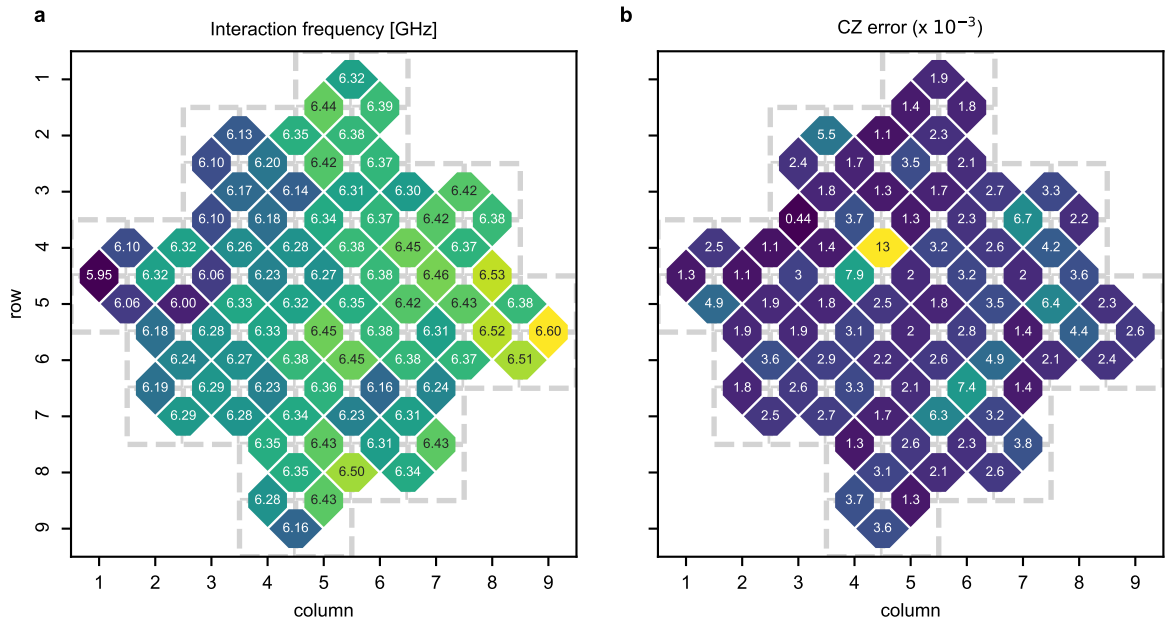

FIG. S4. **Heatmap of two-qubit parameters on the 72-qubit processor. a,** CZ interaction frequency as defined by the average of the two qubits' f_{10} transitions in the middle of the gate. **b,** Simultaneous CZ XEB gate error. Gate errors are measured in the context of applying all gates in a surface code gate layer.

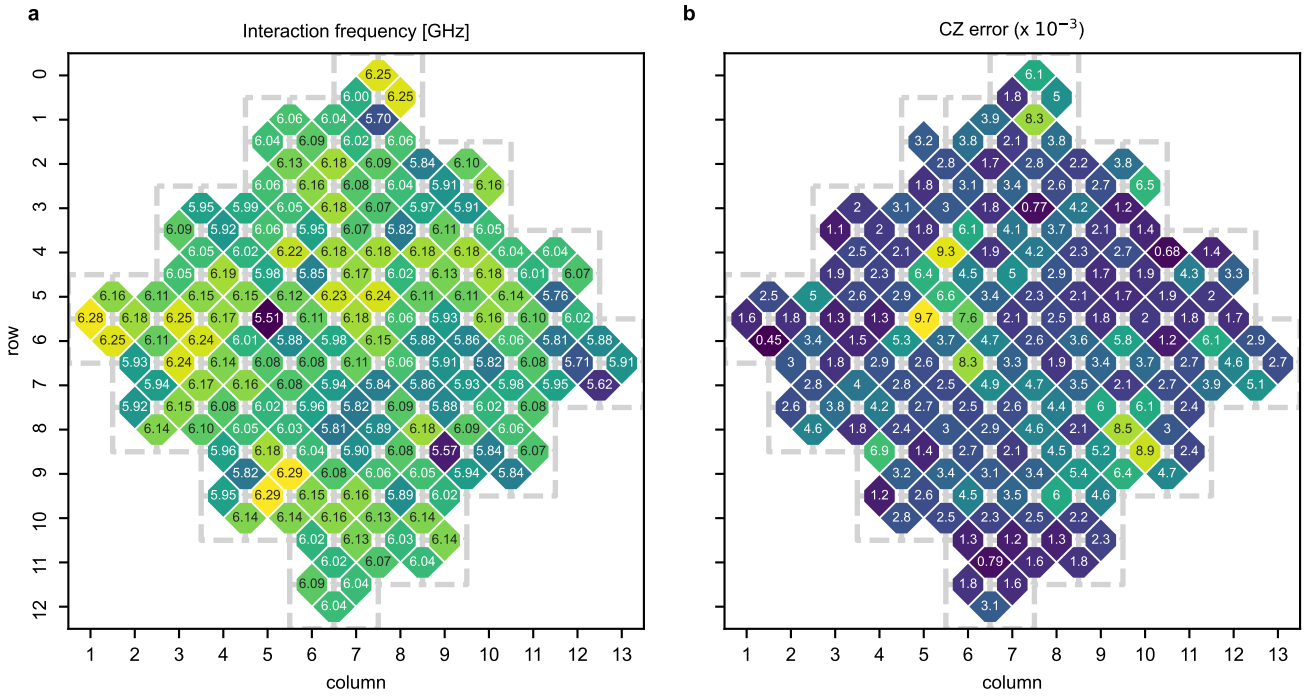

FIG. S5. **Heatmap of two-qubit parameters on the 105-qubit processor. a,** CZ interaction frequency as defined by the average of the two qubits' f_{10} transitions in the middle of the gate. **b,** Simultaneous CZ XEB gate error. Gate errors are measured in the context of applying all gates in a surface code gate layer.

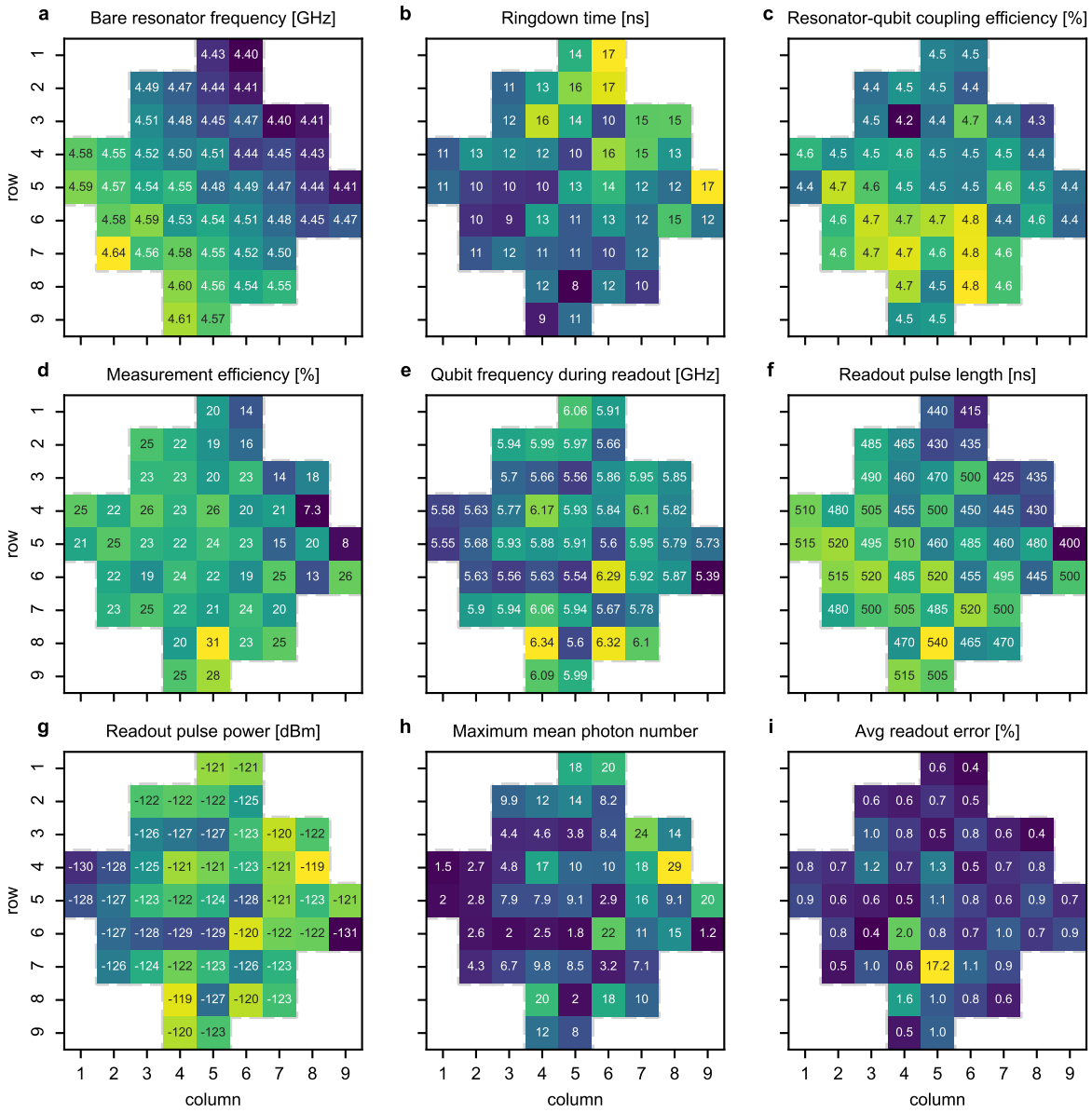

FIG. S6. **Heat map of readout parameters on the 72-qubit processor. a,** Bare frequency of each readout resonator. **b,** Ringdown time (equivalent to $1/\kappa$ of each readout resonator). **c,** The coupling efficiency (k_{qr}) between resonator and qubit. The qubit-resonator coupling strength is given by $g_{qr} = k_{qr}\sqrt{\omega_q \omega_r}/2$, where ω_q and ω_r are the qubit and resonator frequencies respectively. **d,** The measurement efficiency of the readout. This is the fraction of available information of the heterodyne measurement available after signal amplification and digitization. **e,** Qubit frequency during readout (in the absence of a readout drive). **f,** Length of the applied readout tone on each readout resonator. **g,** Applied power to each readout resonator. **h,** Simulated maximum mean photon number in each readout resonator, as calibrated using the AC-Stark shift. **i,** Average readout error, $(P(0|1) + P(1|0))/2$, when preparing random states on all qubits.

B. Decoder Priors

We use two methods for configuring decoder priors. In experiments where maximizing the performance is not the main objective (error injection, real-time decoding demo), we use the SI1000 error model [23]. This error model is inspired by the typical hierarchy of error rates in superconducting qubits, but is otherwise agnostic to the device and decoder.

In experiments that require the highest accuracy, we use the learning-based approach for calibrating the decoder priors [24]. In the learning-based approach, we adopt the hyperparameters from Ref. [24] for the surface code and train the prior on a "calibration dataset" of duration 13 cycles, which precedes the rest of the experiment. The learned parameters of the prior are then

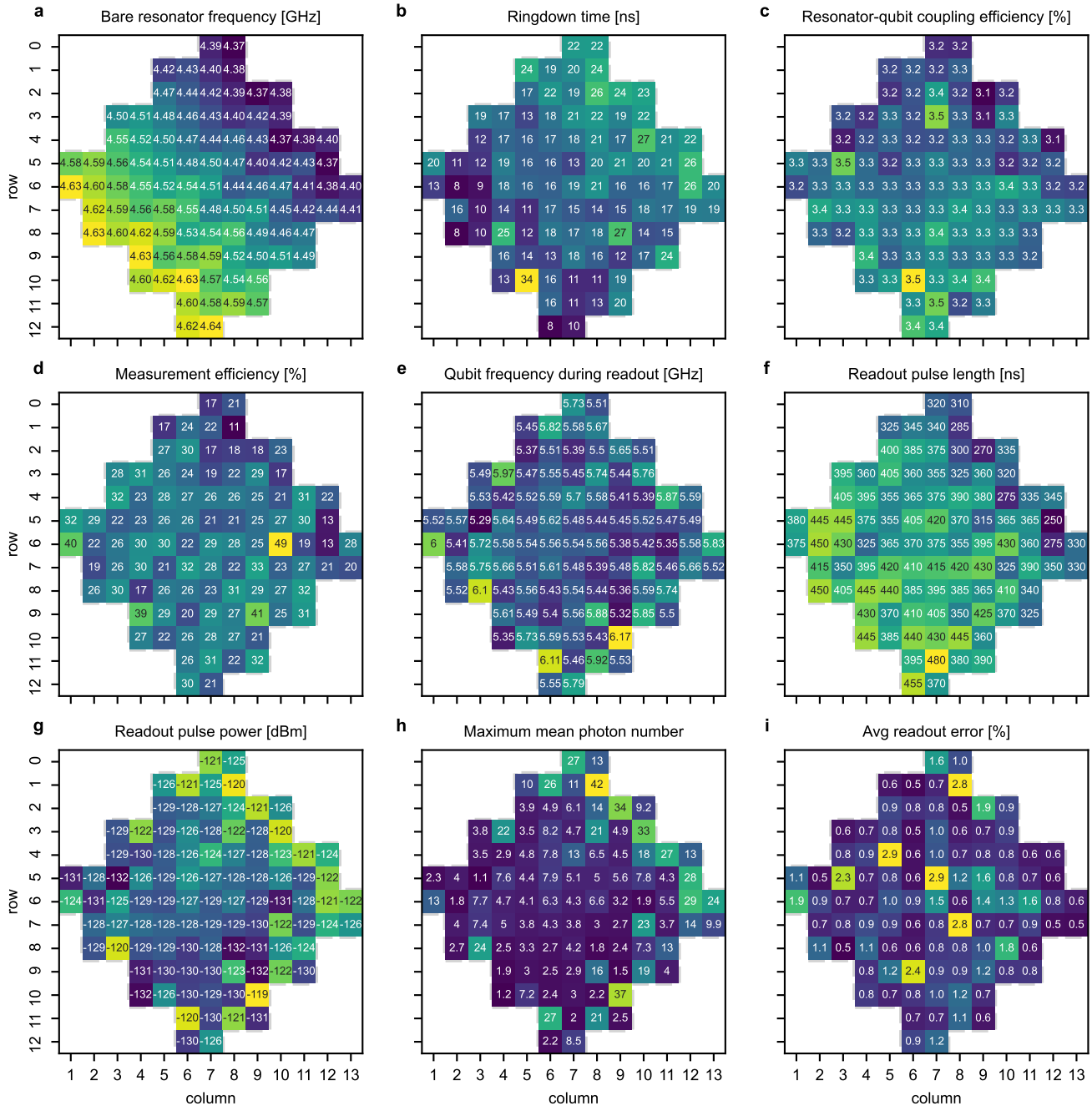

FIG. S7. **Heat map of readout parameters on the 105-qubit processor. a,** Bare frequency of each readout resonator. **b,** Ringdown time (equivalent to $1/\kappa$ of each readout resonator). **c,** The coupling efficiency (k_{qr}) between resonator and qubit. The qubit-resonator coupling strength is given by $g_{qr} = k_{qr}\sqrt{\omega_q \omega_r}/2$, where ω_q and ω_r are the qubit and resonator frequencies respectively. **d,** The measurement efficiency of the readout. This is the fraction of available information of the heterodyne measurement available after signal amplification and digitization. **e,** Qubit frequency during readout (in the absence of a readout drive). **f,** Length of the applied readout tone on each readout resonator. **g,** Applied power to each readout resonator. **h,** Simulated maximum mean photon number in each readout resonator, as calibrated using the AC-Stark shift. **i,** Average readout error, $(P(0|1) + P(1|0))/2$, when preparing random states on all qubits.

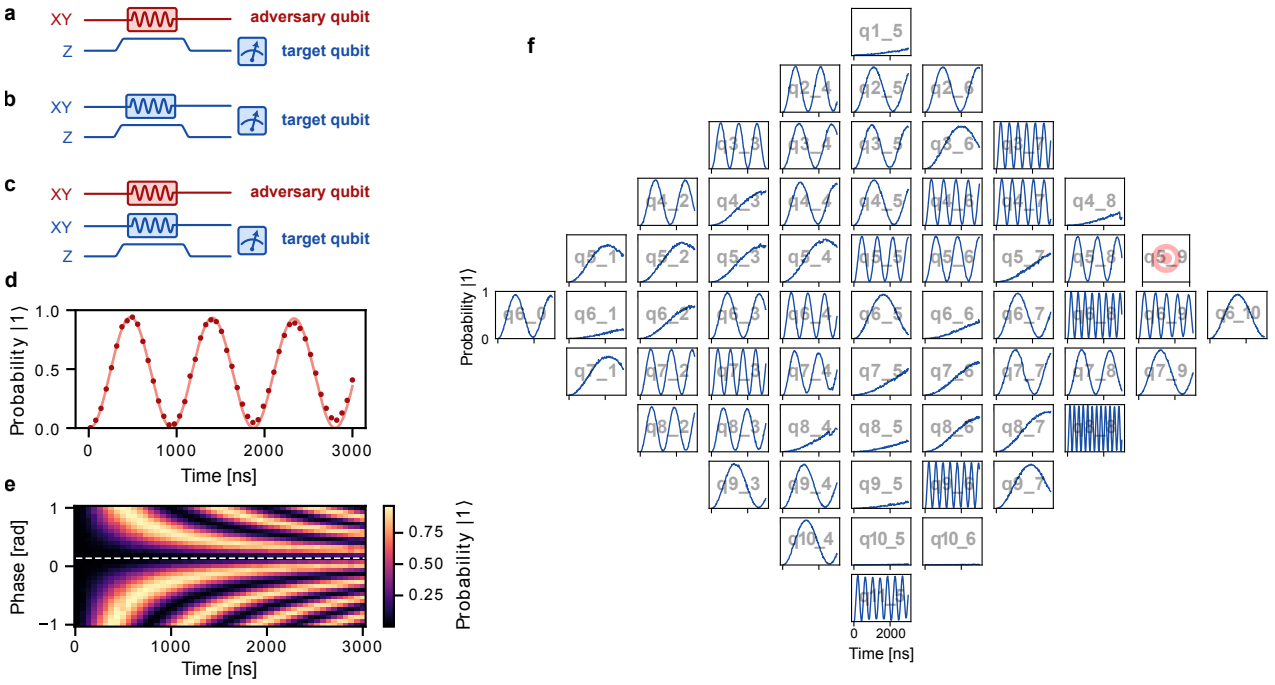

FIG. S8. **Microwave crosstalk measurement. a,** Pulse diagram for an adversarial Rabi measurement, showing the adversary qubit's control line (red) driven by a rectangular-envelope microwave pulse while the target qubit (blue) is dynamically tuned to the frequency of the drive. We measure the target qubit to observe the resulting population in $|1\rangle$ from the adversarial drive. **b,** Pulse diagram for a direct Rabi measurement, in which the target qubit is dynamically tuned to the desired frequency, driven via its own control line with a rectangular pulse, and subsequently measured to extract the $|1\rangle$ state population. **c,** Pulse diagram for a phase measurement, in which the adversary qubit i and target qubit j are driven simultaneously. These drives have amplitudes a and $r_{ij} \times a$, respectively, and the relative phase between the drives is swept. This allows us to extract the crosstalk phase θ_{ij} at which the two drives produce maximally destructive interference in the target qubit population (dotted line in **e**). **d,** Representative adversarial Rabi data (points) corresponding to the pulse sequence shown in **a**. Fitting this data to a sinusoid (line) allows us to extract the adversarial Rabi rate, $\Omega_{\mathrm{adv}}/2\pi = 1.074 \pm 0.001$ MHz. **e,** Representative phase data showing the evolving population in the target qubit (color) versus drive duration for each relative phase between the two drives in **c**. This allows us to extract the crosstalk phase θ_{ij} (dotted line) for this target-adversary pair. **f,** Raw data from adversarial Rabi measurements across the qubit grid for one target qubit. In this dataset, we measure the target (q5_9) while driving each of the other qubits in the grid sequentially. The grid location of each subplot corresponds to the adversary qubit.

used for decoding all datasets up to 250 cycles. Although we frequently decode the surface code using ensembles of matching decoders [15, 17], we use a faster correlated-matching decoder [25] for training. As was shown in [24], the learned priors generalize well across these two decoders.

Our optimization of the priors relies on the sensor technique detailed in Ref. [24]. For the distance-7 surface code, smaller patches of distance-3 and distance-5, shown in Fig. S17(a,b), act as local sensors of the error landscape in the device. These sensor-codes are used to optimize the parameters of the prior, which is then applied to decoding the target distance-7 code. The sensor technique is especially important in the repetition code, as using the logical error rate (LER) of the target distance-29 code in the optimization loop becomes computationally infeasible. Instead, we optimize the prior with the help of 25 distance-5 sensors subsampled from the target code. For the minimum-weight perfect matching (MWPM) decoder, we find that training the prior results in 10% average improvement of the LER at all distances relative to the SI1000 prior.

Additional characterization of decoders and priors on two datasets, `google_72Q_surface_code_d3_d5_set1` and `google_72Q_surface_code_d3_d5_set2`, is shown in Fig. S10. The summary of the datasets, which we made publicly available in Ref. [20], is presented in Table S2.

On the first dataset, we find that the neural network decoder [16] achieves the highest accuracy overall. Among the matching-based decoders we ran for Table S2, the most performant configuration was Harmony [17]. In the future, we plan to carefully compare Harmony to Libra to determine the relative accuracy and scalability benefits of each [15]. We also use a reinforcement learning optimized prior [24]. On the distance-5 code, just ensembling leads to a 1.14 times lower LER on average than correlated matching with SI1000 prior. We expect this improvement will apply to real-time experiments as well, once ensembled matching is layered, parallelized, and incorporated into the real-time decoder, and if optimization

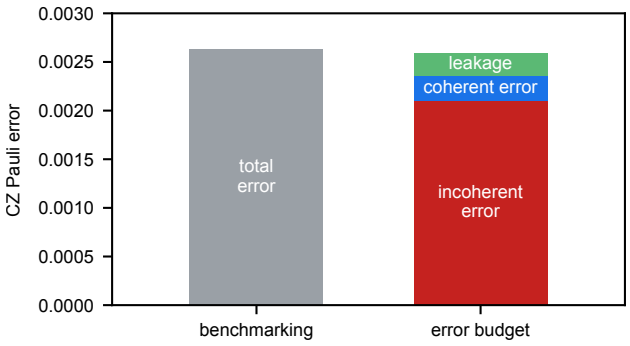

FIG. S9. **CZ error budget.** Error metrics from parallel CZ gates on the 72-qubit processor, plotted as Pauli error. The heights of the stacked bar plot correspond to medians of the measured values of total error, incoherent error, coherent error, and leakage across all CZ gates in our device. Note that incoherent error and coherent error refer specifically to errors in the computational subspace. The median value for total error is extracted from XEB experiments. We observe that benchmarking the gates using CAFE yields a similar median value for total error.

of the prior can be done on a time scale faster than the device drift [24]. We consider both of these requirements feasible. We also note that while this characterization uses Harmony as described in [17], the main text uses the recently-developed ensembled matching-synthesis decoder Libra [15].

C. Neural Network Decoder Details

The machine learning decoder is the recurrent attention-based neural network described in [16] which is trained to predict the logical error based on the stabilizer measurements of a surface code experiment. The network processes one cycle of stabilizer measurements at a time, updating an internal state representation which consists of a vector for each stabilizer. At the end of the experiment the state vectors are combined to produce a calibrated error probability.

As before, the networks (for both the 72-qubit and 105-qubit processors) are trained in two stages. We first pretrain on synthetic data from a generic noise model (SI1000 at $p = 0.4\%$ in this case) generated by Stim [26]. Separate models are trained for X and Z bases for distance-3, distance-5 and distance-7 experiments. We trained 5 separate models with different random seeds for each condition, resulting in 10 pretrained models per code distance that were the basis for all subsequent fine-tuning.

After pretraining, each network is further trained to match a specific experiment by fine-tuning using a limited quantity of experimental data. This fine-tuning stage is broken up into two steps: a first step during which we fine-tune with samples from a p_{ij} detector error model fitted to the 13 cycle dataset, taken before the actual experiment [3]. In a second step the model is fine-tuned on the experimental samples themselves.

For fine-tuning on experimental data, and as in previous experiments [16], we split the experimental data into multiple partitions for cross-validation. We further subdivide each partition into a training and validation set to choose an early-stopping time to prevent overfitting. Test performance is measured on the held out partition. Hyperparameters are chosen based on previous datasets, so no other evaluations are performed on this data.

We utilize various splits of the data. For the synthetic dataset, which is already divided into two equal-size parts, we fine-tune on one and test on the other. For the real-time and distance-7 datasets, of which there is each only one, we utilize an even/odd split. For the stability experiment shown in Fig. 2e in the main text, with multiple temporally spaced repeated memory experiments, we chose a truly causal scenario: for the N^{th} dataset, we fine-tune with the detector error model (DEM) data from the N^{th} set (since it was dependent only on the 13-cycle data captured at the start of that set) but subsequently fine-tune on the previous $(N-1)^{\text{th}}$ dataset. We find that this strict temporal split leads to similar accuracy as an even/odd split.

We use the architecture from Ref. [16] without attention bias, without the residual network before the recurrent neural network and with only two layers in the readout network. Because the experimental data in this paper is of much longer duration (250 cycles vs. 25 for the data in [3]) we add gated recurrence to the state update [28]. Instead of adding the state and incoming stabilizer representations, these are concatenated and projected through a 3-layer multilayer perceptron to compute element-wise update and reset gates. The reset-gated state and new embeddings are concatenated before projection and processing by the syndrome transformer layers. The final output and state are summed, weighted by the update gate. We find this makes the training more stable for long-duration experiments. We mostly train the network on synthetic data of up to 25 cycles, but periodically sample examples of lengths up to 200 to ensure generalization to long experiments. The simulated SI1000 examples had logical observable labels for every cycle up to and including the total length.

D. Real-Time Decoding System

1. *Technical Requirements*

In this section we provide a brief overview of general technical requirements for a real-time syndrome decoding system and describe the key aspects of the specific system employed in this work.

In the simplest quantum error correction experiments, syndrome data is saved in a file and processed by a decoder offline after the quantum circuit is finished execut-

Dataset	Number of experiments	Basis	Distance	Cycles	Shots
`google_72Q_surface_code_d3_d5_set1` This surface code dataset corresponds to Fig. 2e in the main text, which represents the last 16 experiments that were run consecutively.	21	X, Z	3, 5	1 to 250	5×10^4
`google_72Q_surface_code_d3_d5_set2` Samples in this surface code dataset capture different calibration states of the device, ranging from well-calibrated with LER of the distance-5 code smaller than 3×10^{-3} to poorly calibrated with LER approaching 10^{-2}. This selection is intended for testing the decoding algorithms under a wide range of experimental conditions.	35	X, Z	3, 5	5 to 50	6×10^4
`google_105Q_surface_code_d3_d5_d7` This surface code dataset corresponds to Fig. 1(c,d) in the main text.	1	X, Z	3, 5, 7	1 to 250	5×10^4
`google_72Q_repetition_code_d29` This repetition code dataset corresponds to Fig. 3a in the main text.	100	X, Z	29	10^3	10^5

TABLE S2. **QEC datasets.** Brief summary of the QEC datasets released in Ref. [20]. Further details can be found in each dataset's `README.md` file, including examples of data processing with open-source tools such as Stim [26] and PyMatching [27].

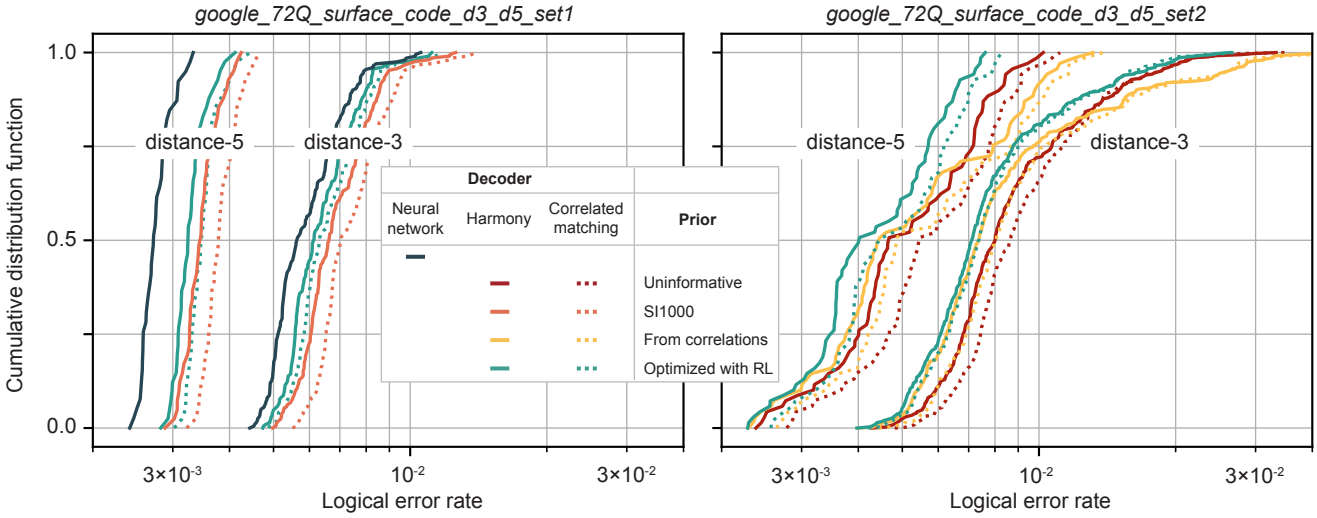

FIG. S10. **Characterization of offline decoding methods.** The decoder calibration methods benchmarked here are the same as in Ref. [24]. The first dataset corresponds to the 15-hour sweep in Fig. 2(e) of the main text. The second dataset contains experiments in which the device was well calibrated and experiments in which the calibrations have not been refreshed for several days. This selection is intended for testing the decoding algorithms under a wide range of experimental conditions.

ing on the quantum processor. This offline mode of syndrome decoding is insufficient for experiments involving fault-tolerant constructions, such as gate teleportation [29] that use feed-forward, i.e. the capability to condition quantum operations on measurement outcomes. In order to support these more advanced experiments, one needs a fast decoder capable of computing logical measurement outcomes in real-time during the execution of a quantum circuit.

A real-time syndrome decoding system must satisfy requirements concerning three key performance metrics: accuracy, latency, and throughput. Decoding accuracy is characterized by the per-cycle probability that the logical measurement outcome computed by the decoder is correct.

Decoding latency, T_{decode}, is the delay between when the measurement outcomes of the physical qubits comprising a logical qubit become known and when the measurement outcome of the logical qubit becomes known. This is an important contributor to the overall reaction

	Pretrain			Fine-tune (DEM)			Fine-tune (Experimental)		
	3×3	5×5	7×7	3×3	5×5	7×7	3×3	5×5	7×7
Parameter count	5.411M	5.419M	5.432M						
Channels	256	256	256						
Training steps	1.56M	5.86M	5.86M	195k	781k	10.9M	234k/15k*	234k/15k*	234k/15k*
Batch size	256	256	256	256	256	256	64/1024*	64/1024*	64/1024*
Training examples	400M	1500M	1500M	50M	200M	2800M	15M†	15M†	15M†
Learning rate	5e-6	5e-6	5e-6	1.7e-6	1.3e-6	2e-6	2e-7	2.5e-7	3e-7
Weight decay	1e-5	1e-5	1e-5	1e-5	1e-4	1e-3	7e-5	7e-5	7e-5
Cosine cycle	250M	300M	500M	40M	160M	640M	30M	80M	200M
EMA step	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4	8e-4	8e-4	8e-4

TABLE S3. **Hyperparameters for the machine learning decoder.** *For fine-tuning on the real-time and simulated data we choose a batch size of 1024; for the others a batch size of 64. †For fine-tuning with the even-odd (causal) split, we train on 258,440 (583,440) unique experimental samples: we keep 5120 validation set samples of each of the 25,000 (50,000) shots per split for each of the 13 experiment durations (10, 30, ..., 250 cycles). For the real-time experiment we only have 5,000 examples per duration, so we train on 1700 and tested on 800 from each split. Consequently, training presents each example many times. For pretraining and DEM fine-tuning we are sampling data from a simulator so each training example is different.

time, T_{react}, which is the time it takes the quantum computer's control system to react to a logical measurement by executing a dependent logical operation [30]. Inverse reaction time is the rate at which the quantum computer can execute logical operations that require feed-forward for fault-tolerance, such as the T-gate in the surface code. Thus, inverse reaction time plays the role of clock speed of a quantum computer.

In more detail, the reaction time is given by the sum

$$T_{\text{react}} = T_{\text{decode}} + T_{\text{control}} \tag{1}$$

of the decoding latency T_{decode} and the time T_{control} that the control system spends executing quantum gates, obtaining measurement outcomes for physical qubits, and reacting to the logical measurement outcomes provided by the real-time decoding system. In a non-error-corrected processor with feed-forward, we have $T_{\text{decode}} = 0$ and the reaction time is just T_{control}. Decoding latency T_{decode} constitutes most of the overhead of the quantum error correction layer.

In a software-based real-time syndrome decoding system, we can break decoding latency down into software latency and I/O latencies,

$$T_{\text{decode}} = T_{\text{input}} + T_{\text{software}} + T_{\text{output}} \tag{2}$$

where T_{software} is the latency of the software components of the decoding system, and T_{input} and T_{output} are the input and output latencies of the hardware-software interface which facilitates exchange of information between the quantum computer's control electronics and the decoding software.

Another requirement for a real-time decoding system concerns its throughput, i.e. the amount of syndrome information it is capable of processing per unit time. Decoding throughput must be at least as high as the amount of syndrome information generated per unit time. Failure to satisfy this requirement causes syndrome backlog to accumulate and leads to exponential slowdown of the quantum computer [31].

In order to establish real-time performance of our decoding system we measured T_{software} as a function of circuit depth, as shown for a single representative experiment using a quantum circuit with one million cycles in Fig. S11. We see that T_{software} is roughly constant as a function of time. Moreover, when the latency spikes the system is able to recover quickly. Measurements from many experiments with different number of cycles are shown at distance-5 in Fig. S12 and using both simulated and stitched together experimental data at distance-7 in Fig. S13. Because the real-time decoding infrastructure was not integrated with the 105-qubit device, we synthesized measurement data for long memory experiments by stitching together shots of 250-cycle memory experiments used for plots in the main text. To stitch together shots, we concatenate detection events from the bulk of each shot. At the interface between shots, we use edges obtained from decoding to ensure the detection events are consistent. Stability of T_{software} together with evidence that all syndrome information has been analyzed by the decoder proves that the system processes syndrome information at the rate at which it is generated.

Measurements indicate that in our current real-time decoding system $T_{\text{input}} < 10$ µs and $T_{\text{software}} = 63$ µs on average. In a hypothetical system with the addition of T_{output} and T_{control} in the range of a few microseconds, the reaction time would be within an order of magnitude from that assumed in Ref. [32]. Improving decoding latency while scaling the system to larger code distances is an outstanding engineering challenge on the path to large-scale fault-tolerant quantum computing.

There are trade-offs between accuracy, latency, and

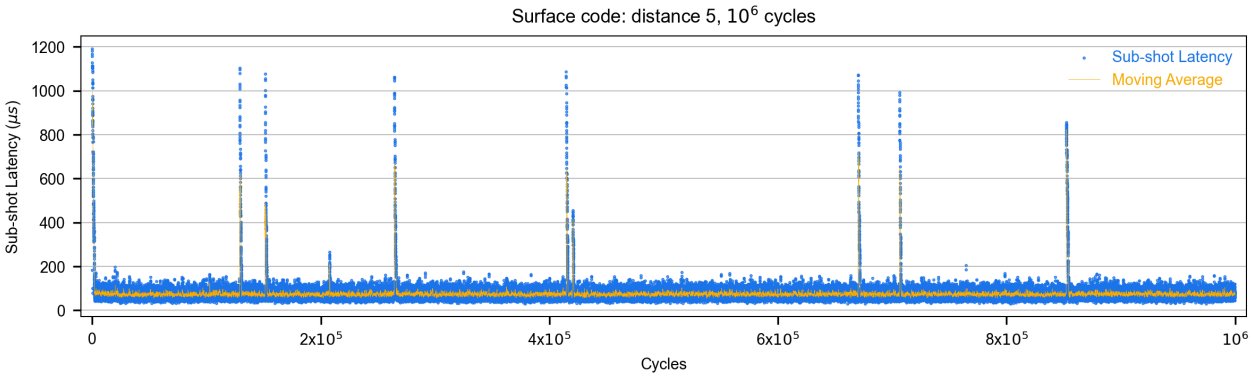

FIG. S11. **Sub-shot latency as a function of cycles within a single experiment.** Sub-shot latency is defined as the delay from the moment a ten-cycle block of syndrome data is acquired by the software decoding system to the point when the block is fully processed by the decoder during a single experimental run. We observe stable real-time decoding performance with occasional, brief latency spikes from which the system recovers rapidly.

throughput. For example, aggressive batching of syndrome or intermediate information may increase throughput at the expense of increased latency. Similarly, a decoder that short-circuits rare and computationally hard cases of syndrome decoding by making a random guess may achieve higher throughput and lower latency at the expense of reduced accuracy. The requirements interact in other ways, too. For example, in early fault-tolerance experiments, the logical circuit depth budget may be increased either by improving decoding accuracy or by reducing decoding latency.

A different kind of trade-off exists in the fundamental choice between decoders implemented in hardware and decoders implemented in software. The former promise greater performance while the latter offer greater flexibility. Our experiments demonstrate that a parallel and optimized C++ software decoder can achieve real-time performance when provided exclusive access to a subset of CPUs. In particular, we use a 64-core AMD Ryzen Threadripper PRO 5995WX with 62 GB of RAM and a real-time Linux operating system. However, we only use use 32 of the 64 available cores when decoding, as this already exceeds the necessary throughput. The flexibility of software-based decoding facilitates ongoing research in decoding algorithms and fault-tolerant constructions. In particular, software syndrome decoding can handle broken or disabled components [33], allows easing of hardware requirements [34], and promises support for novel and customized constructions for early fault-tolerant quantum algorithms.

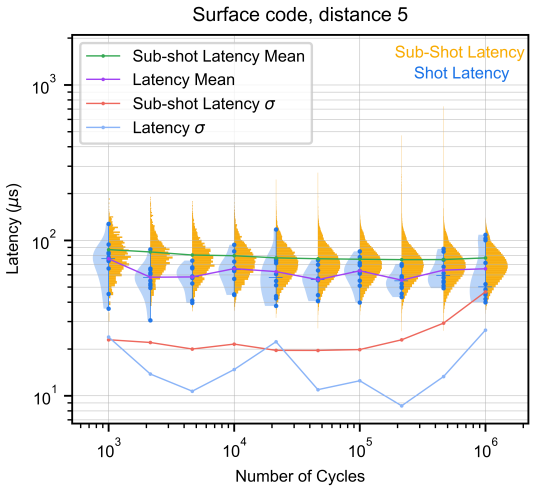

FIG. S12. **Software decoding latency at distance-5.** Here we show the latency associated with the software components, T_{software}, as a function of the number of cycles for ten independent experimental shot runs. Blue points indicate per shot latency (10 shots per number of cycles), with horizontal bars denoting the median, and the blue-shaded violin plot illustrating the shot latency distribution. The yellow histograms represent the distribution of sub-shot latencies, obtained by dividing each shot into sub-shots of ten-cycle blocks. Stable latency as a function of the number of cycles indicates the real-time decoding system achieves the required throughput.

2. *Correlated Parallel Blossom*

In this section we describe our real-time decoder, a multi-threaded implementation of a correlation-augmented minimum-weight perfect matching decoder capable of processing a stream of syndrome data, using a matching graph buffer that does not scale with the number of cycles. Our decoder uses Sparse Blossom [25] as the matching engine (main subroutine) and upgrades it to an accurate real-time decoder by making the following major changes:

a. Preweight correlations. Sparse Blossom, as described in Ref. [25], assumes that the noise model is graphlike (each error causes at most two detection events)

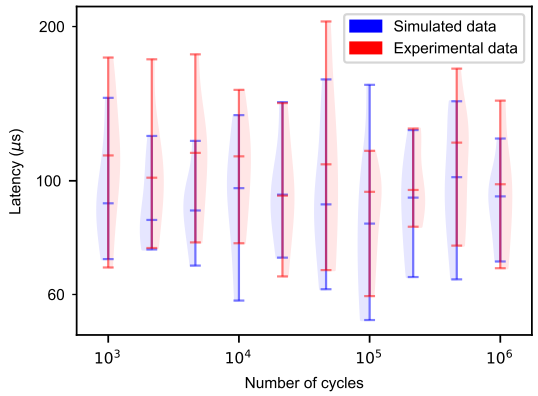

FIG. S13. **Software decoding latency at distance-7.** Horizontal bars denote the median. Because the real-time decoding infrastructure was not integrated with the 105-qubit device, this includes simulated and experimental data streamed into the decoder after being saved offline. The simulated data comes from a Pauli model of the 105-qubit device. The experimental data comprises 250-cycle experiments stitched together. Note that, due to slight differences in the simulated versus experimental pipeline, this data was streamed in at a cycle rate of ≈1.2 µs (instead of the device cycle rate of ≈1.1 µs). We observe an average decoder latency of 95 µs on simulated data and 108 µs on experimental data.

and ignores "hyperedge" error mechanisms that can cause more than two detection events, such as Y errors in the CSS surface code. We improve the accuracy of Sparse Blossom by reweighting the graph using a "preweight correlations" subroutine. This is similar to the reweight strategy introduced in Ref. [19], which itself is a fast approximation of the two-pass correlation strategy introduced in Ref. [18]. Concretely, we iterate over the detection events and identify any edge satisfying either of the following conditions:

1. The edge has a detection event at both of its endpoints.

2. The edge is a boundary edge that has a detection event v at its endpoint, and furthermore there are no detection events directly adjacent to v.

If an edge e satisfies either of these conditions then we add it to the set C of initially chosen edges. For each edge in C we then condition on an assumption that the associated error mechanism occurred to obtain a posterior probability that other related errors occurred, updating edge weights accordingly. For example, if an edge e in C corresponds to an X error at a particular location of a CSS surface code circuit, then from our knowledge of the noise model we know that a Y error could have occurred at the same location and hence we lower the edge weight corresponding to a Z error at the same circuit location.

b. Block-based parallelization. We parallelize Sparse Blossom following a similar strategy to how Fusion Blossom parallelizes Parity Blossom [35]. Our decoder partitions the matching graph into *blocks*, where each block consists of M syndrome measurement cycles. Each block is assigned to a thread, which first reweights the graph using the preweight correlations strategy described in Section III D 2 a. The same thread then runs Sparse Blossom on the block, with the Sparse Blossom timeline processed until every region is matched either to another region, the boundary of the graph, or the boundary of the block. Similar to Fusion Blossom [35], neighboring blocks are then *fused* in stages, again using Sparse Blossom as the matching engine.

The fusion graph we use is different from the ones proposed in Fusion Blossom. Specifically, we apply two layers of fuses (our fusion graph has height three): we first fuse each even block with the odd block immediately following it, and then we fuse each odd block with the even block immediately following it. Here, a block is even or odd if its sequential index is even or odd, respectively. Rather than adding a root node to the fusion graph, we instead declare a heralded error if any regions remain matched to a block boundary once fusing has completed. We ensure that the rate of heralded failures is negligible relative to other contributions to the logical error rate by choosing a sufficiently large number of cycles per block M. We find it is sufficient to set $M = 10$ for the distance-3 and distance-5 surface codes and $M = 90$ for the distance-29 repetition code.

Finally, we extract the predicted logical observable from each block as a bitmask and undo the edge reweights. The overall prediction of the decoder is then the bitwise XOR of the predicted logical observable bitmask for each block.

c. Constant-sized graph buffer. Rather than storing the full matching graph in memory, we instead use a constant-sized graph buffer. This graph buffer emulates the full matching graph while only storing 128 contiguous blocks at any instant. A separate *grapher* thread maintains the graph buffer, ensuring that the buffer always contains the region of the graph being acted on by the decoder. The grapher exploits the repeating pattern of the memory experiment matching graph and does not rewrite any of the graph buffer in the bulk of the experiment, with the only significant work being done at the beginning and end of the experiment where the local structure of the graph changes.

IV. Understanding Fidelity

A. Error Budget Simulations

1. *Surface Code Simulation Details*

We briefly summarize the methods and high-level workflow used to simulate the error correction experiments. Except when otherwise noted, the simulation details are equivalent to those discussed in Ref. [3]. The gate sequence matching each memory experiment is first

represented in a `cirq.Circuit` [36]. The circuit is then dressed with additional channels corresponding to different error mechanisms. Each error channel is represented numerically using Kraus operators, which are compiled specifically for each noise model and qubit based on experimental characterizations. The following mechanisms are included:

- Decoherence, quantified by decay (T_1), pure dephasing (T_ϕ), and passive heating of qubits to state $|2\rangle$.
- Readout and reset error.
- Dephasing-induced leakage of the higher frequency qubit during the CZ gate. (This is represented by a transition from state $|11\rangle$ to $|02\rangle$.)
- Stray coupling crosstalk between nearest-neighbor or diagonal-neighbor qubits during parallel CZ operation.
- Excess error (not accounted for by previous sources) on single-qubit and CZ gates, as well as during idling of data qubits during qubit measurement and reset.
- Transport of leakage between CZ-gate qubits through higher-excitation transitions (e.g., $|12\rangle \rightarrow |30\rangle$).

Beyond these errors already included in Ref. [3], we also allow for imperfect operation of the data qubit leakage removal operation (DQLR) [37]. For simplicity, this is implemented using a phenomenological model matching experimentally observed reset fidelity for state $|2\rangle$. The Kraus operators of this channel are given by

$$K_{ij} = \sqrt{P_{j\rightarrow i}}|i\rangle\langle j| \tag{3}$$

$$K_0 = \sqrt{I - \sum_{i,j} K_{ij}^\dagger K_{ij}} \tag{4}$$

where $P_{j\rightarrow i}$ is the probability that standard basis state $|j\rangle$ of the data qubit is reset to state $|i\rangle$. Notably, this channel acts trivially on the computational subspace.

After the circuit is dressed with all relevant error channels, we apply a Generalized Pauli Twirling Approximation to each noise channel. This converts each noise channel to a generalized Pauli channel that also includes leakage, thereby making it compatible with Clifford simulation methods. Finally, the noisy circuit is parsed and passed to the `Pauli+` simulator [3], which generates samples of simulated experimental data.

2. *Surface Code Performance at Large Code Distances*

In the main text, we have discussed surface code experiments for code distances $d = 3$, 5 and 7. Experimental characterizations of the qubits and gates of the 72-qubit and 105-qubit processors reveal that our component errors (see Tables S4 and S5) are roughly a factor of two better than those of Ref. [3]. In this section, we simulate the performance of surface codes at code distances larger than those realized experimentally. In particular, we discuss (a) the effect of non-uniform component errors on the logical error rate (LER) and the error suppression factor Λ; (b) the achievable logical error rates if we improve our components' fidelity by another factor of two; (c) the effect of foregoing DQLR on the exponential suppression of the logical error rate with code distance d; and (d) the crossover regime where we can observe finite-size error suppression above threshold. We point out that, for efficiency, all results of this section (except when noted otherwise) are obtained using a correlated minimum-weight perfect matching decoder [3] that is less accurate than the one used in the main text. Moreover, we carry out the simulations using the component error parameters of those of the 72-qubit processor, summarized in Table S4, and a slightly different cycle time of 1086 ns.

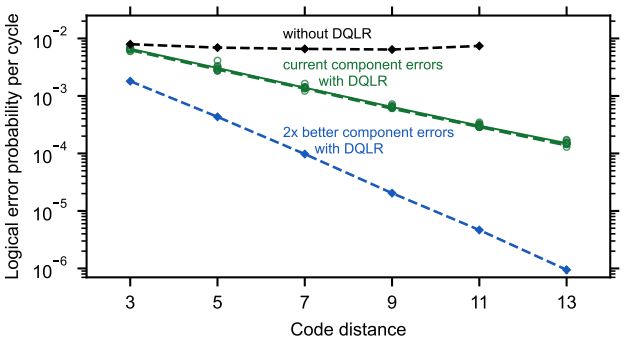

FIG. S14. **Simulated logical error rates for larger code distances.** For a given code distance, the green open circles indicate the LERs of ten different non-uniform error models obtained by re-sampling (bootstrapping) the measured component error probabilities of our distance-5 surface code on the 72-qubit processor. Note that the circles are overlapping. The green solid line denotes the geometric mean of the LERs indicated by the green circles. The dashed lines depict LERs for error models that, except for stray crosstalk errors, are uniform over each qubit. The dashed green and black lines respectively show the LERs with and without data qubit leakage removal (DQLR) gates at the end of each error correction cycle. Finally, the dashed blue line shows the logical error rates expected for a device with twice better component error probabilities and with DQLR.

a. Effect of non-uniform component errors on surface code performance. The component error probabilities of our devices are intrinsically non-uniform. For instance, the T_1 time varies from qubit to qubit with a mean value of 73 µs and standard deviation of 18 µs (for other component errors, see Table S4). To investigate the effect of the non-uniformity of component errors on the performance of surface codes at large code dis-

Component	Metrics for this work	Metrics for Ref. [3]
CZ gates (incl. CZ crosstalk & CZ leakage)	$(3.5 \pm 1.4) \times 10^{-3}$ [total error]	6.05×10^{-3}
CZ crosstalk	$\simeq 5.5 \times 10^{-4}$	9.5×10^{-4}
CZ leakage probability	2.0×10^{-4}	2.0×10^{-4}
Data qubit idle	$(0.9 \pm 0.2) \times 10^{-2}$	2.46×10^{-2}
Reset	1.5×10^{-3}	1.86×10^{-3}
Readout	$(0.8 \pm 0.3) \times 10^{-2}$	1.96×10^{-2}
Single-qubit gates	$(6.2 \pm 4.9) \times 10^{-4}$	10.9×10^{-4}
Leakage from heating	2.5×10^{-4} [assumes $\Gamma_{1\to2}^{-1} = 2\,\mathrm{ms}$]	6.4×10^{-4} [$\Gamma_{1\to2}^{-1} = 0.7\,\mathrm{ms}$]
T_1	(73 ± 18) µs	23 µs
$T_{2,\,\mathrm{cpmg}}$	(80 ± 28) µs	30 µs
Cycle time	1076 ns	921 ns

TABLE S4. **Qubit and gate error metrics for the 72-qubit processor.** Note that the left column (corresponding to this work) reports mean values, while the right column (corresponding to Ref. [3]) reports median values. Compared to Ref. [3], our present device exhibits roughly twice better qubit and gate performance. Reported uncertainties indicate one standard deviation from the mean value.

Component	Metrics for this work
CZ gates (incl. CZ crosstalk & CZ leakage)	$(4.1 \pm 2.2) \times 10^{-3}$ [total error]
CZ crosstalk	$\simeq 3.3 \times 10^{-4}$
CZ leakage probability	2.0×10^{-4}
Data qubit idle	$(0.8 \pm 0.3) \times 10^{-2}$
Reset	$(2.3 \pm 1.6) \times 10^{-3}$
Readout	$(0.8 \pm 0.2) \times 10^{-2}$
Single-qubit gates	$(5.3 \pm 4.3) \times 10^{-4}$
Leakage from heating	2.8×10^{-4} [assumes $\Gamma_{1\to2}^{-1} = 2\,\mathrm{ms}$]
T_1	(68 ± 13) µs
$T_{2,\,\mathrm{cpmg}}$	(89 ± 28) µs
Cycle time	1108 ns

TABLE S5. **Qubit and gate error metrics for the 105-qubit processor.** We report mean values and the uncertainties indicate one standard deviation from the mean value.

tances, we generate synthetic non-uniform error models by resampling (bootstrapping) the measured component errors of the $d = 5$ surface code grid. For codes of each distance $d = 3, 5, 7, 9, 11$ and 13, we generate ten statistically independent synthetic error models and compute the corresponding LERs.

The results are depicted by the open green circles in Fig. S14. The solid green line joins the geometric means of the computed LERs. For comparison, we also perform surface code simulations where, for each error type, we homogenize the component errors by taking the arithmetic mean (e.g. we assign all qubits a T_1 of 73 µs). In these simulations, however, the stray coupling crosstalk errors are still non-uniform, since they depend on the qubit frequencies. The LERs for these (almost) uniform error models are shown by the dashed green line in Fig. S14. We find that these LERs show good agreement with the geometric means of the LERs from the non-uniform error models. Both dashed and solid green lines indicate that logical errors are exponentially suppressed at large code distances with a suppression factor $\Lambda \approx 2$. From the solid green line, we have $\Lambda_{3/5} = 2.16, \Lambda_{5/7} = 2.2, \Lambda_{7/9} = 2.16, \Lambda_{9/11} = 2.12$ and $\Lambda_{11/13} = 2.01$, where $\Lambda_{d/d+2} \equiv \mathrm{LER}(d)/\mathrm{LER}(d+2)$. We perform all these calculations with DQLR gates that remove leakage states $|2\rangle$ and $|3\rangle$ with 100% and 50% fidelities, respectively. In Eq. (3), this corresponds to nonzero transition probabilities $P_{2\to1} = 1, P_{3\to2} = 0.5, P_{3\to1} = 0.5$. These results show that the error non-uniformity of our surface code components induces a relatively small spread in the logical error rates at large code distances. Additionally, the observed logical error suppression factor $\Lambda_{3/5} \gtrsim 2$ holds at larger code distances, despite the non-uniformity of component errors.

b. Performance forecast for devices with twice better component errors. By fitting the data of the solid green line of Fig. S14 with $\mathrm{LER}(d) = C \cdot [1/\Lambda]^{(d+1)/2}$, we obtain $\Lambda \approx 2.14$ and coefficient $C \approx 3.0 \times 10^{-2}$. This implies that to achieve a logical error rate of, e.g., $\mathrm{LER} = 10^{-6}$ would require a device that could hold a surface code with a minimum distance $d = 27$ (1457 qubits). An alternative path to achieving a LER of 10^{-6} is to continue

improving the surface code component fidelities. Since we have demonstrated a factor of two improvement in the component errors with respect to our previous device [3], we perform a simulation to forecast the performance of surface codes with component errors that are twice better than what we have demonstrated in this work. The results are indicated by the blue dashed line in Fig. S14. The predicted logical error suppression factor is $\Lambda \approx 4.5$ and a 10^{-6} logical error rate can be achieved with a $d = 13$ surface code (337 qubits). As shown in Section IV A 3, the most important component errors to improve are CZ errors and data qubit idle errors.

c. Importance of DQLR in surface codes. In this section, we demonstrate through numerical simulations the importance of the DQLR operation for logical device performance. In Ref. [37], it was shown that data qubit leakage removal is instrumental to achieving the exponential suppression of logical errors with code distance when physical errors are well below the error threshold. The reason for this is that leakage states (e.g. $|2\rangle$, $|3\rangle$, etc.) that survive over several error correction cycles create time correlations in error detection events that confuse the error decoder. DQLR limits the growth of leakage populations and the lifetime of leakage states (ideally to one cycle), recovering the exponential logical error suppression with code distance promised by quantum error correction.

The green and black dashed lines of Fig. S14 show the logical error rate as a function of code distance with and without DQLR, respectively. We use component errors that are uniform, except for the errors due to stray coupling crosstalk. The error parameters used in these simulations are given in Table S4, corresponding to experimental values of the 72-qubit processor. Without DQLR, we find that the logical error suppression factor $\Lambda_{d/d+2}$ saturates to a value around 1 at large code distances, while with DQLR, exponential suppression of logical errors holds to large code distances with a $\Lambda_{d/d+2}$ that does not exhibit a significant dependence on code distance, d.

d. The crossover regime. It is worth noting that the assumption that Λ is distance-independent can break down at $\Lambda_{3/5} \approx 1$, as was reported in Ref. [3]. This can occur even in a simple Pauli simulation, where one can achieve error suppression at finite sizes above threshold that will eventually turn around. To illustrate this crossover regime, we employ a simple Pauli simulation of the error rates reported in Table S4, see Fig. S15. There, we observe that for $\Lambda_{3/5} \leq 1.3$, the initial error suppression diminishes rapidly with distance. We note that the threshold is much better behaved when reporting the error rate per d rounds [38], which is relevant to the logical operation of the surface code.

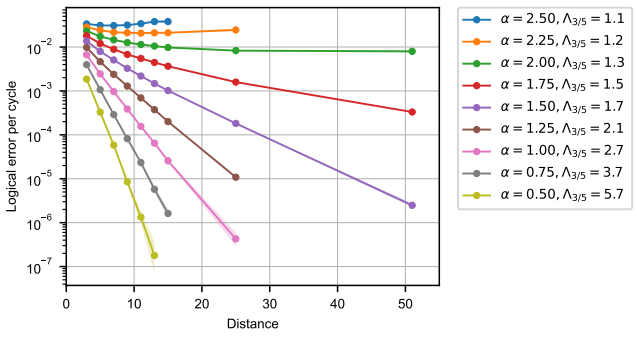

FIG. S15. **The crossover regime.** A Pauli simulation of the error model described in Table S4 using $3d$ rounds and correlated matching. The factor α scales the error rates of all the Pauli channels uniformly. We observe that for a $\Lambda_{3/5} \leq 1.3$, error suppression eventually saturates with distance, before turning around. However, for $\Lambda_{3/5} \geq 1.5$, we see the characteristic below-threshold error suppression continuing.

3. Surface Code $1/\Lambda$ Error Budget

We construct a $1/\Lambda$ error budget for our 72-qubit processor in a manner similar to Ref. [3]. We write $(\Lambda_{3/5})^{-1}$ as a sum of contributions from each error channel i that is considered in the simulation:

$$(\Lambda_{3/5})^{-1} = \sum_i w_i \, p_{\text{expt}}^{(i)}. \tag{5}$$

The weights w_i are the sensitivities of $(\Lambda_{3/5})^{-1}$ to a uniform increase of the error probabilities of the ith error channel from the baseline error operation point (specified in Table S4). From Table S6, we find that errors related to CZ gates (the first three rows) contribute about 60% of the $1/\Lambda$ budget. The next important contributor to $1/\Lambda$ is data qubit idle errors (about 20%), followed by readout and reset errors (12%) and single-qubit gate errors (9%). We remark that the sensitivities (w_i) are obtained assuming perfect DQLR and using a correlated matching decoder for efficiency [18]. The predicted value of Λ from a linear combination of these component errors and sensitivities, following Eq. (5) and Table S6, yields a calculated $\Lambda_{3/5}^{\text{linear}} = 2.25$.

We can compare this to the prediction from the `Pauli+` simulations (described in Section IV A 1), which directly model distance-3 and distance-5 surface code circuits with the component errors incorporated as Kraus operators, and return simulated logical error rates for each code distance. Taking the ratio of these simulated logical error rates yields $\Lambda_{3/5}^{\texttt{Pauli+}} = 2.17$. We note that this agrees reasonably well with the prediction of $\Lambda_{3/5}^{\text{linear}}$. This shows that in the presence of DQLR, which prevents leakage from propagating to subsequent cycles and causing long-time correlations in error detection events, the linear error model provides an accurate estimate of Λ.

We also compare these Λ predictions to the experimentally observed value, $\Lambda_{3/5}^{\text{expt.}} = 1.97$, obtained using the correlated minimum-weight perfect matching decoder. We find that the $1/\Lambda$ error budget overpredicts $\Lambda_{3/5}^{\text{expt}}$ by

Component	$p^{(i)}_{\text{expt}}$	w_i	$1/\Lambda$ contrib.
CZ gates (doesn't include CZ crosstalk & CZ leakage)	2.8×10^{-3}	65	0.182 (41%)
CZ crosstalk	5.5×10^{-4}	91	0.05 (11%)
CZ leakage	2.0×10^{-4}	108	0.022 (5%)
Data qubit idle	0.9×10^{-2}	10	0.09 (20%)
Readout	0.8×10^{-2}	6	0.048 (11%)
Reset	1.5×10^{-3}	6	0.009 (2%)
SQ gates	6.2×10^{-4}	63	0.039 (9%)
Leakage (heating)	2.5×10^{-4}	18	0.005 (1%)

TABLE S6. **Error budget calculation for** $(\Lambda_{3/5})^{-1}$ using device parameters from the 72-qubit processor when decoding with correlated matching. We multiply the component errors $p^{(i)}_{\text{expt}}$ by the sensitivities (weights) w_i at the error operation point, specified in Table S4, resulting in the relative contributions to $(\Lambda_{3/5})^{-1}$ shown in the right-most column.

roughly 14%, where the discrepancy arises mainly due to additional experimental errors not included in the simulation, and not due to inaccuracy of the $1/\Lambda$ error budget itself.

B. Comparison to Physical Qubit Lifetime

The average channel fidelity of a quantum channel $\mathcal{E}$: $\rho \to \mathcal{E}(\rho)$ to a target identity channel is defined as

$$\overline{\mathcal{F}}[\mathcal{E}] = \int d\psi \langle\psi|\mathcal{E}(|\psi\rangle\langle\psi|)|\psi\rangle, \tag{6}$$

where the integral is over the uniform measure on the state space, normalized so that $\int d\psi = 1$. This fidelity characterizes the closeness of a quantum channel to the identity without giving preference to any particular axis on the Bloch sphere.

As shown in Ref. [39], the uniform averaging in Eq. (6) is equivalent to averaging over the six cardinal points on the qubit Bloch sphere, the eigenstates of the Pauli operators,

$$\overline{\mathcal{F}}[\mathcal{E}] = \frac{1}{12}\sum_{P=X,Y,Z} \text{Tr}\left[P\mathcal{E}(P)\right] + \frac{1}{2}, \tag{7}$$

which becomes evident by using the linearity of the quantum channels, $\mathcal{E}(P) = \mathcal{E}(|+P\rangle\langle+P|) - \mathcal{E}(|-P\rangle\langle-P|)$. Eq. (7) provides an experimental protocol for extracting fidelity of an *arbitrary* channel to the identity.

The simplest way to implement an identity is via an idling operation. For the physical qubits subject to amplitude damping at rate γ_1 and white-noise dephasing at rate $\gamma_\varphi = \gamma_2 - \gamma_1/2$, idling for time t results in

$$\overline{\mathcal{F}}(t) = \frac{e^{-\gamma_1 t} + 2e^{-\gamma_2 t}}{6} + \frac{1}{2}. \tag{8}$$

On the other hand, for a qubit subject to Pauli channel with Pauli error rates γ_X, γ_Y and γ_Z, we have:

$$\overline{\mathcal{F}}(t) = \frac{e^{-2(\gamma_Y+\gamma_Z)t} + e^{-2(\gamma_X+\gamma_Z)t} + e^{-2(\gamma_X+\gamma_Y)t}}{6} + \frac{1}{2}. \tag{9}$$

Hence, qubits subject to different error channels experience different decay of fidelity in time, which complicates their direct comparison. However, at short times this decay can be approximated as

$$\overline{\mathcal{F}}(\delta t) = 1 - \frac{1}{2}\Gamma\delta t, \tag{10}$$

where Γ is an effective depolarization rate and $1/\Gamma$ is an effective lifetime of a uniformly sampled pure state. This quantity can be directly compared among qubits with different error channels. Note that this metric does not favor qubits with extremely biased noise [40–42], for which Γ is dominated by the largest error rate. For physical qubits, from Eq. (8) we obtain

$$\Gamma_{\text{physical}} = \frac{\gamma_1 + 2\gamma_2}{3}. \tag{11}$$

For a qubit encoded in the surface code, we model a logical Pauli channel [43, 44] with error probabilities p_X, p_Y and p_Z per cycle. When these error probabilities are small, the channel can be modeled as continuous in time, with rates $\gamma_X \approx p_X/t_c$, $\gamma_Y \approx p_Y/t_c$, and $\gamma_Z \approx p_Z/t_c$, where t_c is the cycle duration. For such a continuous logical Pauli channel, from Eq. (9) we obtain

$$\Gamma_{\text{logical}} = \frac{4}{3}(\gamma_X + \gamma_Y + \gamma_Z) \tag{12}$$

Note that to first order $p_X + p_Y + p_Z = 2\varepsilon_d - p_Y$, with ε_d defined in the main text as the logical error probability per cycle averaged over the X and Z eigenstates. This equation allows us to establish a strict upper bound on Γ_{logical}, since $p_X + p_Y + p_Z \leq 2\varepsilon_d$, resulting in

$$\Gamma_{\text{logical}} \leq \frac{8\varepsilon_d}{3t_c}. \tag{13}$$

Simulations suggest [45, 46] that in the surface code $p_Y \sim p_X p_Z$ is suppressed compared to p_X and p_Z, because Y errors predominantly occur as independent X and Z errors. Hence, we expect that this bound is also tight. One illustrative consequence of this is that the

lifetime of Y eigenstates in the surface code would be approximately twice shorter than the X and Z eigenstates, although due to a more complicated preparation [45], we do not measure this lifetime in our experiment.

Following Refs. [44, 47, 48], we define the gain G as an improvement of the effective lifetime of an error-corrected logical qubit over the best physical qubit in the same system (with the break-even point corresponding to $G = 1$). This metric represents the usefulness of QEC in protecting quantum information in time; however, it does not take into account the cost of state preparation, measurement, or gates.

In our system, comparing against the qubit with the longest measured physical lifetime, we have

$$1/\Gamma_{\text{physical}} = 119 \pm 13\,\mu\text{s}, \tag{14}$$
$$1/\Gamma_{\text{logical}} = 291 \pm 6\,\mu\text{s}, \tag{15}$$

which leads to $G = 2.4 \pm 0.3$. This demonstrates a beyond break-even quantum memory realized with a multi-qubit code, extending previous results with bosonic codes [44, 47, 48].

V. Error Correction Experimental Details

A. Low Probability Events in the Repetition Code

In the distance-29 repetition code experiment, we achieve $\Lambda > 8$ and observe an apparent logical-error-per-cycle floor of 10^{-10} at large distances. This result constitutes a four orders-of-magnitude improvement compared to the lowest logical error per cycle achieved in previous experiments [41]. However, even this lower error rate floor will be limiting for larger fault-tolerant circuits. We find that the logical errors at high distances are caused by rare events with distinct detection event probability signatures. In this section, we report relevant metrics for these rare events.

One type of event presents as a correlated burst of detection event probability, with a sharp rise followed by an exponential decay to baseline levels over the course of several shots. An example is given in Fig. 3d, and another is presented in Fig. S16. These events account for all the distance-27 logical errors we observe, and half of the distance-21 to distance-25 errors. The measure qubits corresponding to the firing detectors are grouped spatially, not necessarily in sequence with the snaking repetition code order. The bursts occur roughly once per hour, or once every 3×10^6 shots, in both bit- and phase-flip codes. The decay timescale from fitting the detection fractions to a decaying exponential is around 400-700 µs. This is in stark contrast to the detection fraction bursts caused by quasiparticles observed in previous work [3, 49], which happened once every 10 seconds with a recovery time of $25 - 30$ ms.

The other type of limiting event presents as a single noisy detector, whose likelihood of firing rapidly increases and stays high for 1-2 ms. This event causes a high-distance (defined here as $d > 19$) error also about once per hour of data acquisition. The detector which is noisy varies from event to event. An example is presented in Fig. S16.

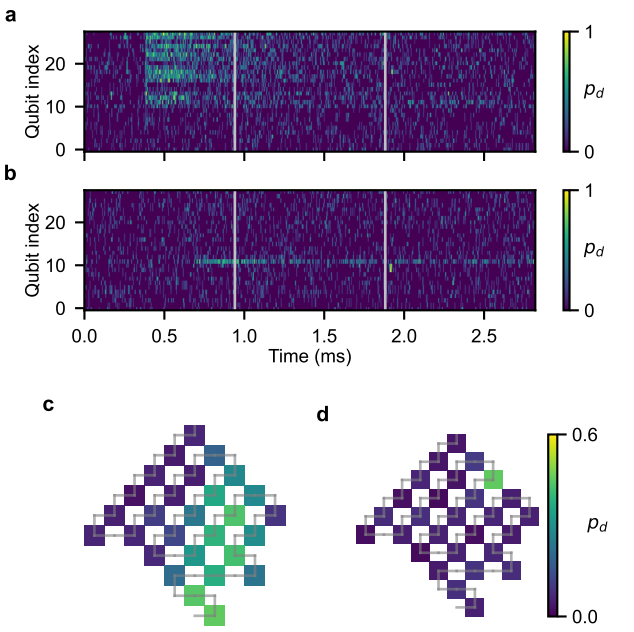

FIG. S16. **Repetition code. a,** Detection probability as a function of time for the measure qubit corresponding to each detector (indexed along repetition code direction), during a burst event. Detections are smoothed with a $\sigma = 2$ Gaussian filter. Vertical lines represent boundaries between shots. **b,** Same as **a**, but for the case of a single noisy detector. **c** and **d,** Detection probability averaged over the first 200 ns of the burst event and single noisy detector event, respectively, plotted on the qubit grid. Grey lines represent the repetition code gate direction.

B. Grid and Circuit Details

In this section, we present experimental details associated with the qubit grids and quantum circuits used for the surface code experiments. Fig. S17 shows the placement of distance-3 and distance-5 grids on both processors. To compare smaller-distance codes to larger ones, we average over several smaller codes chosen to cover the larger code with minimal overlap, extending the methodology of Ref. [3].

Fig. S18 shows a representative dataset of logical error versus cycles for the 72-qubit processor taken prior to the time series shown in Fig. 2e. This dataset gives $\Lambda = 2.31 \pm 0.02$. Fig. S19 shows the detection event probabilities from various experiments, including distance-3 and distance-5 codes corresponding to Fig. S18, distance-29 repetition codes on the 72-qubit processor, and distance-

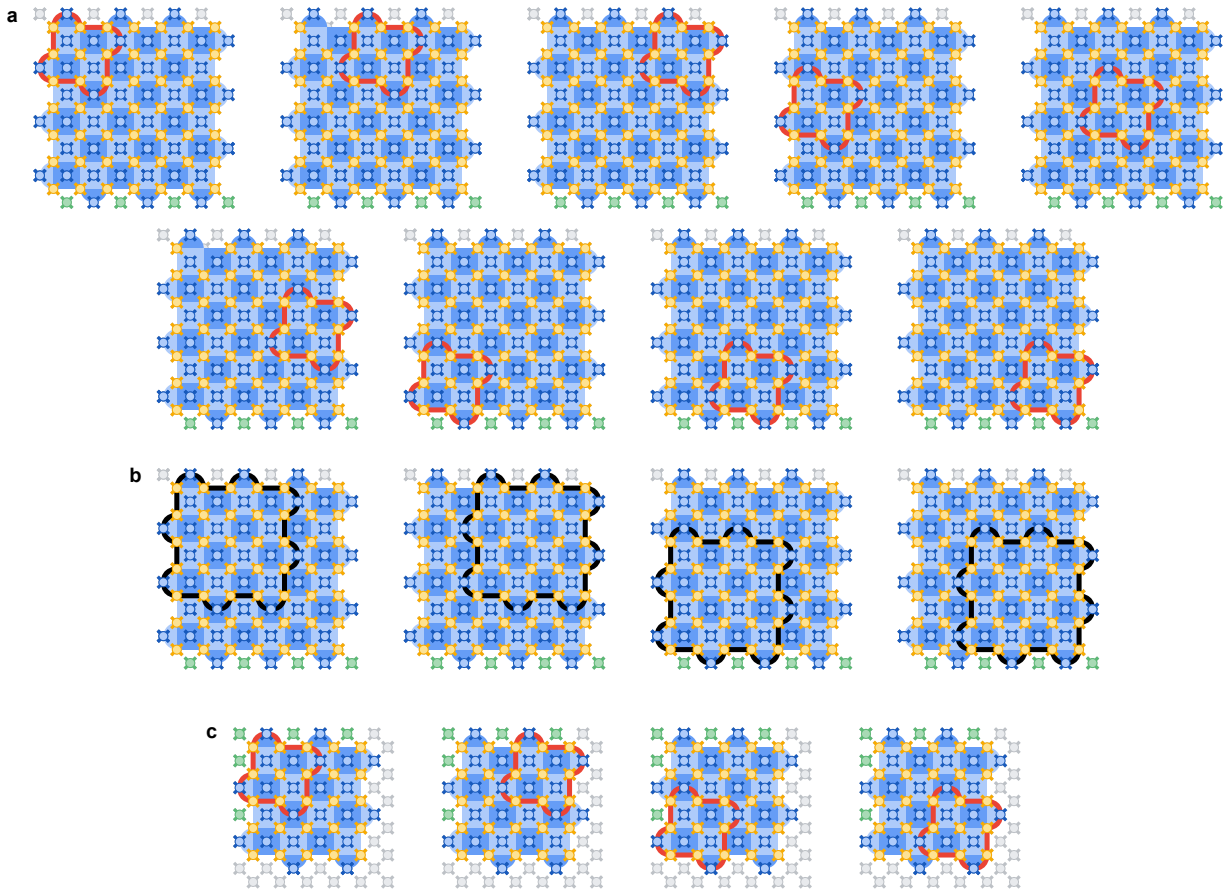

FIG. S17. **Surface code grid layout details. a,** Layout of distance-3 grids used on the 105-qubit processor. **b,** Layout of distance-5 grids used on the 105-qubit processor. **c,** Layout of distance-3 grids used on the 72-qubit processor.

3, distance-5, and distance-7 codes on the 105-qubit processor. Figure S22 shows the detection probabilities predicted from a Pauli simulation of the 105-qubit device. We see good agreement, with the simulation predicting detection probabilities for $d = (3, 5, 7)$ of $p_{\text{det}} = (8.0\%, 8.5\%, 8.6\%)$ compared to experiment which has $p_{\text{det}} = (7.7\%, 8.5\%, 8.7\%)$. The natural rise in detection probability with system size can be ascribed to reducing finite size effects – for example, each corner, edge, and bulk data qubit is involved in 2, 3, and 4 CZ gates, respectively. In Fig. S20 we show a comparison of detection probabilities from the data qubit leakage removal (DQLR) test from Fig. 2, noting the rise in detections for the experiments without DQLR, which we attribute to leakage accumulation.

Fig. S21 outlines the steps of a surface code error correction circuit, broken down into layers of single- and two-qubit gates as well as dynamical decoupling and measurement operations.

For experiments sweeping over several codes and numbers of cycles, such as Fig. 1c, we interleave experiments of different codes, as in Ref. [3]. We acquire one (number of cycles, basis, code) dataset at a time, each with 50,000 repetitions (10 initialization bitstrings times 5,000 repetitions). The initialization bitstrings are 5 random bitstrings and their complements; complementary bitstrings have opposite logical eigenvalues for odd-distance surface codes. We shuffle the order in which we measure each number of cycles. The order of enumeration is the following.

```
for n in num_cycles:
    for basis in (Z, X):
        for code in codes:
            take_data(n, basis, code)
```

VI. Uncertainty Analysis

A. Logical Performance

From a statistical point of view, the error suppression factor Λ is a random variable, defined with respect to a given collection of codes, whose randomness comes from sampling noise and performance drift for each of the given codes. Evaluating the uncertainty in our estimate of Λ

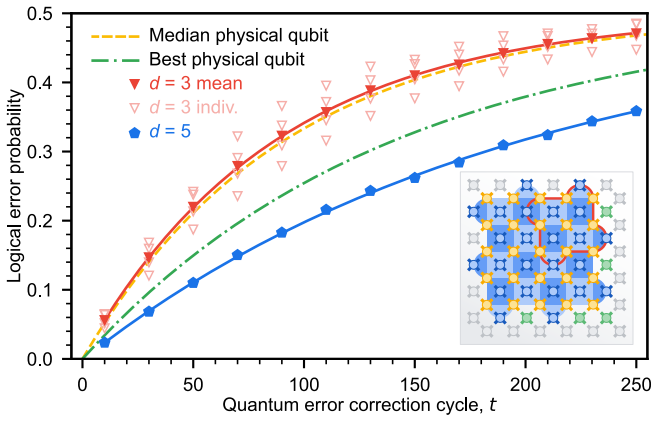

FIG. S18. **Logical performance on the 72-qubit processor.** Here we show a similar result to Fig. 1c of the main text, except on the 72-qubit processor. Decoding is performed with the neural network. The neural network decoder achieves $\varepsilon_5 = 0.252\% \pm 0.002\%$ and $\Lambda = 2.31 \pm 0.02$. It also achieves a break-even memory at distance-5, with a median and best physical qubit lifetime of approximately 73 µs and 113 µs, respectively, compared to a logical qubit lifetime of approximately 160 µs.

	X basis		Z basis	
Grid	Error rate	Std. dev.	Error rate	Std. dev.
distance-3 (0, 0)	0.00561	0.00013	0.00516	0.00008
distance-3 (0, 4)	0.00562	0.00015	0.00526	0.00014
distance-3 (0, 8)	0.00519	0.00012	0.00475	0.00021
distance-3 (4, 0)	0.00672	0.00021	0.00596	0.00023
distance-3 (4, 4)	0.01044	0.00045	0.00800	0.00030
distance-3 (4, 8)	0.00641	0.00028	0.00516	0.00039
distance-3 (8, 0)	0.00678	0.00030	0.00724	0.00024
distance-3 (8, 4)	0.00982	0.00021	0.00854	0.00012
distance-3 (8, 8)	0.00507	0.00015	0.00533	0.00014
distance-5 (0, 0)	0.00294	0.00008	0.00229	0.00008
distance-5 (0, 4)	0.00311	0.00007	0.00238	0.00009
distance-5 (4, 0)	0.00362	0.00008	0.00326	0.00014
distance-5 (4, 4)	0.00352	0.00008	0.00309	0.00009
distance-7 (0, 0)	0.00155	0.00004	0.00130	0.00004

TABLE S7. **Logical errors per cycle for Fig. 1c.** Both logical error per cycle and standard deviation is shown for each grid and basis using the neural network decoder. Grid labels correspond to the open source [20] labeling, e.g. distance-3 $(0, 4) \leftrightarrow$ d3_0+4j.

involves evaluating the uncertainty in our estimates of the mean logical error per cycle for the collection of codes with a particular distance, and then propagating that through to uncertainty in that scaling with respect to depth, which we detail below.

In Fig. 1c of the main text, we consider logical error probability p_L for a variety of codes for different numbers of error-correction cycles t. Each point has statistical uncertainty $\sqrt{p_L(1-p_L)/N}$ with $N = 10^5$ repetitions. For example, $p_L = 0.1$ corresponds to statistical uncertainty 10^{-3}.

To determine the logical error per cycle ε_d, we fit exponentials to p_L versus number of cycles t (technically, by fitting a line to $\ln(1-2p_L)$ versus t) for each code. We typically see residuals greater than those expected by statistical uncertainty in p_L, which we attribute to system performance drift over the course of the experimental sweep. Along with uncertainty in the fitted log slope obtained from linear regression, we use the excess residuals to estimate the uncertainty in ε_d including this drift. See Ref. [3] for details. We compute ε_d and an uncertainty for each code and logical basis and then average over basis and code for data reported in the manuscript. Since the suppression factor Λ is defined with respect to the mean logical error per cycle for each distance, we compute the uncertainty in the mean from each individual uncertainty σ_i, specifically $\sqrt{\Sigma_i \sigma_i^2}/n$. Note this amounts to considering the particular ensemble of codes as fixed, with all the randomness coming from noise in sampling from a given code rather than sampling different codes.

Note that p_L, and thus ε_d, depend on the decoding scheme used. In Fig. 1d, we plot ε_d values for the neural network decoder. The specific values for the neural network decoder and the ensembled synthetic matcher Libra [15] are reported in the left-hand column of Table S1. Furthermore, the specific ε_d for each grid and basis of the neural network decoder are reported in Table S7.

Similarly, the values corresponding to the accuracy tests on our real-time data in Fig. 4c are reported in the right-hand column of Table S1. Those experiments included sweeps from 10 to 250 cycles (as in Fig. 1c) but with fewer repetitions (10^4 for each code and each number of cycles, split over logical X and Z bases). The smaller number of repetitions increases statistical uncertainty. We fit ε_d for cycles $t > 90$, reserving the shorter experiments for decoder training. The underlying logical error probability data is plotted in Fig. S23.

We compute Λ using linear regression of $y = \ln \varepsilon_d$ versus $x = (d+1)/2$, including the uncertainty of the ε_d values. The slope $m = -\ln \Lambda$, from which we compute $\Lambda = e^{-m}$. Regression gives us uncertainty δm, and the uncertainty $\delta\Lambda = e^{-m}\delta m$.

We use the same regression analysis for the repetition code Λ. In this case, we consider each data qubit idle basis (X and Z) separately, as those two cases are sensitive to different physical error mechanisms. Fitting for d from 5 to 11 (see main text), we obtain $\Lambda_x = 8.27 \pm 0.02$ and $\Lambda_z = 8.55 \pm 0.02$. To report an overall value, we use the mean $\Lambda = (\Lambda_x + \Lambda_z)/2$ and estimate the uncertainty $|\Lambda_x - \Lambda_z|/2$.

For the distance-5 experiments on the 72-qubit processor, we instead consider $\Lambda_{3/5} = \varepsilon_3/\varepsilon_5$, which is equivalent to the Λ fit discussed above for two points.

We compute uncertainties in physical qubit lifetimes by propagating uncertainties in T_1 and $T_{2\mathrm{CPMG}}$, obtained

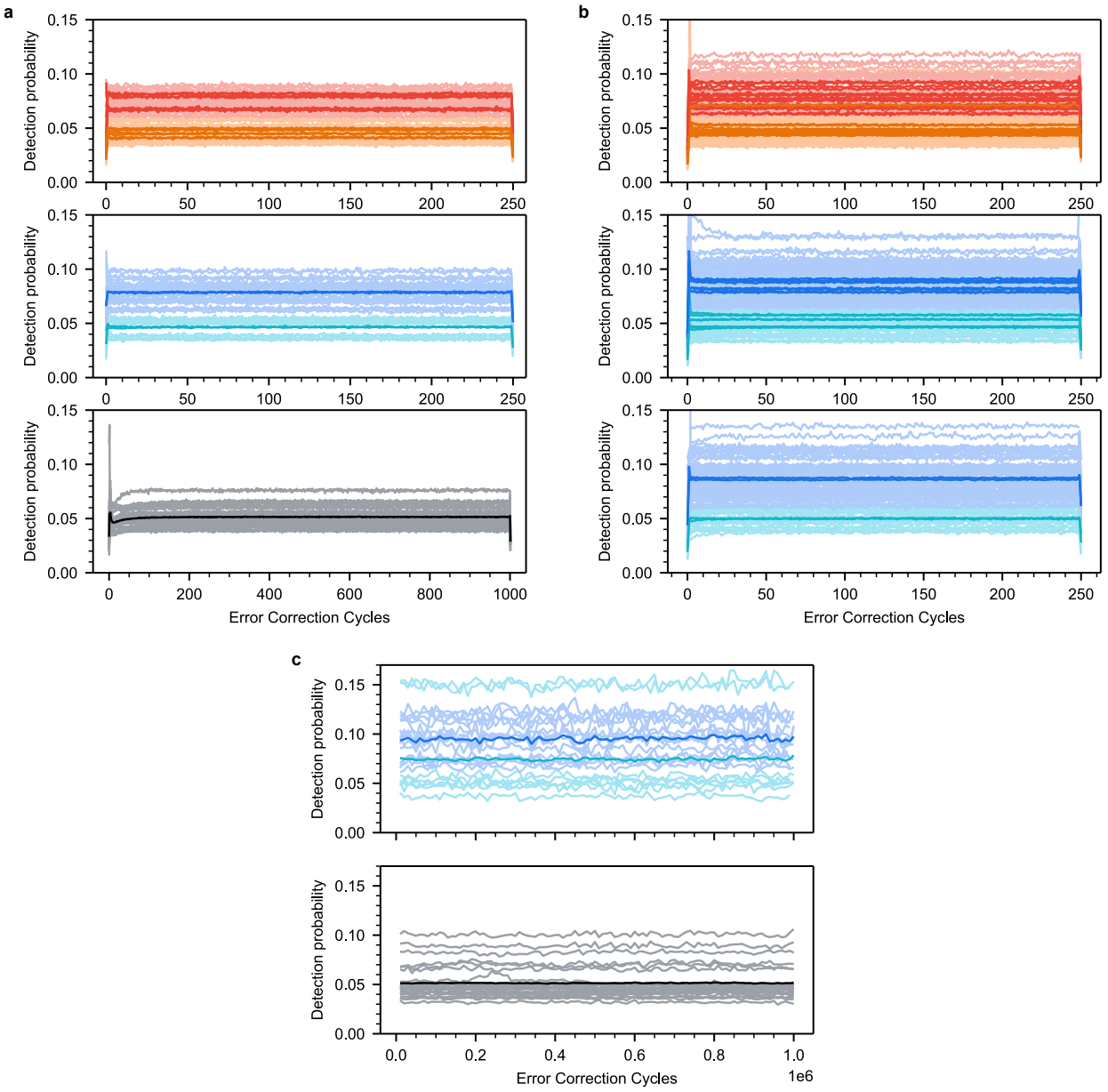

FIG. S19. **Detection event probabilities. a,** Average detection probability as a function of error correction cycles for all detectors in the different grids in the 72-qubit processor. (top) Distance-3 grids, with weight-4 stabilizers in red and weight-2 stabilizers in orange. The average X and Z basis detectors are plotted in bold. (middle) Distance-5 grid, with weight-4 in dark blue and weight-2 in light blue. (bottom) distance-29 repetition code. **b,** Detection probabilities for different grids in the 105-qubit processor; (top) distance-3 grids, (middle) distance-5 grids, (bottom) distance-7 grid. **c,** Detection probability as a function of error correction cycles, run continuously for one million cycles, for (top) the distance-5 grid and (bottom) the distance-29 repetition code in the 72-qubit processor. Here we plot one shot, and apply a rolling average of 10,000 cycles.

with `scipy.curve_fit`.

B. Logical Error per Cycle From One Point

In Fig. 2b and 3 of the main text, we estimate the logical error per cycle from experiments with a single number of cycles. The error-injection experiments in Fig. 2b and 3b are 10 cycles long; we choose shorter experiments so we can resolve high logical error $\varepsilon_d \sim 0.1$. The repetition code experiments in Fig. 2a are 1000 cycles so we can resolve $\varepsilon_3 \sim 10^{-3}$ while maximizing experiment length. To estimate the error per cycle ε_d from a single point (t, p_L), we consider a binomial problem with logical error probability ε_d at each step. The cumulative error probability p_L after t cycles is the probability of an odd number of

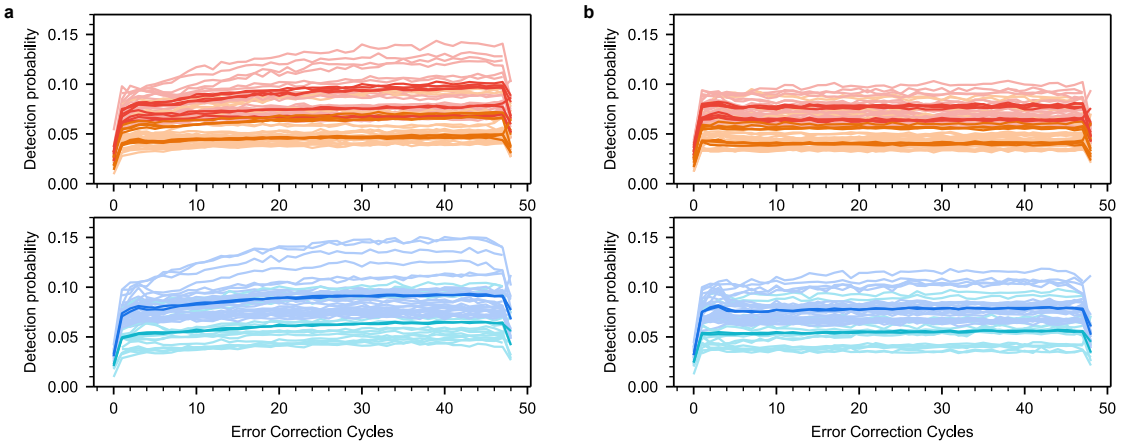

FIG. S20. **Detection event probabilities without and with data qubit leakage removal.** **a,** Average detection probability as a function of error correction cycles without data qubit leakage removal (DQLR) for all detectors in the distance-3 grids (top) and distance-5 grid (bottom) in the 72-qubit processor. We attribute the rise in detections to leakage accumulation. **b,** Average detections as a function of error correction cycles with DQLR; the data qubit leakage removal step successfully removes the rise in detections.

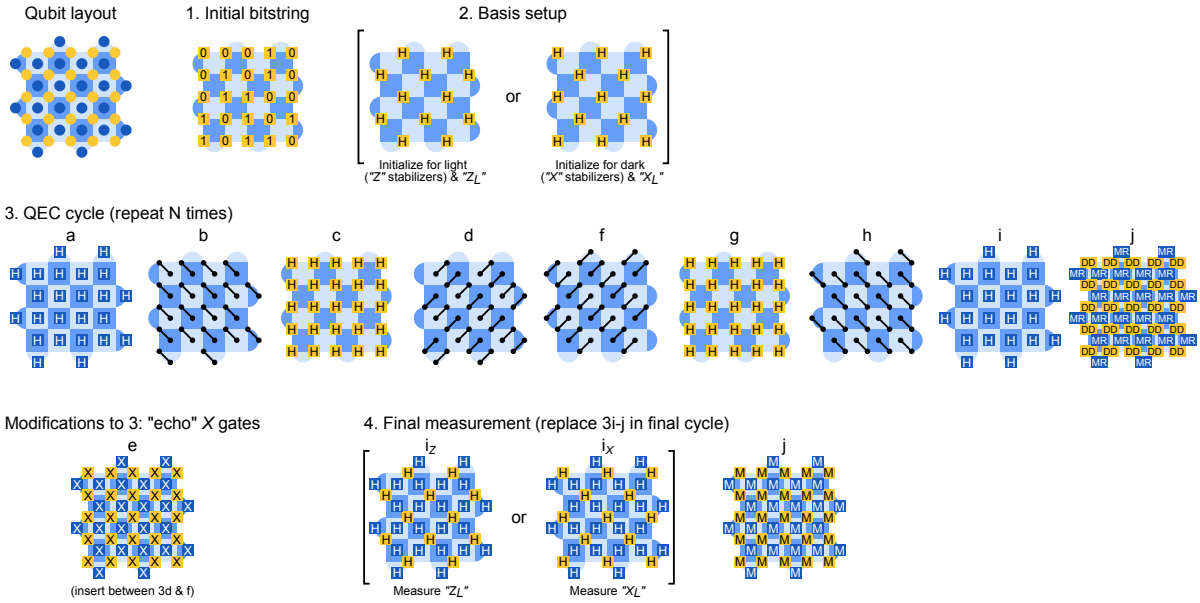

FIG. S21. **Distance-5 surface code circuit.** Spatial layout of data qubits (gold dots) and measure qubits (blue dots) are shown here on a distance-5 grid. 1. We reset all qubits to $|0\rangle$ and then prepare an initial bitstring state on the data qubits using X gates. 2. We apply H (Hadamard) gates to some of the data qubits to convert the initial bitstring (eigenstate of all Z operators) into an eigenstate of half the $ZXXZ$ stabilizers, the half matching to logical operator of interest (X_L or Z_L) for the specific experiment. Note steps (1) and (2) could be combined into a single moment of Clifford gates, but for simplicity we execute them in two moments as shown. 3. QEC cycle showing the explicit patterns for Hadamard gates and CZ gates as well as dynamical decoupling (DD) and measurement and reset operations (MR). Note that although all stabilizers measure $ZXXZ$, the "X" stabilizers and "Z" stabilizers apply their CZs in a different order (specifically in 3d and f). This pattern is carefully designed to manage "hook" errors. We modify this gate sequence with additional "echo" or "dynamical decoupling" gates, adding X gates to all qubits in 3e (inserted between the middle two CZ moments). 4. For the final measurement, we replace 3i-j with a different Hadamard pattern (transforming the data qubit state to the appropriate basis for logical measurement) and measure all qubits simultaneously (M). This final measurement is used for detectors for the penultimate cycle (measure qubit results) and the final cycle (data qubit results, converted to parities in the relevant basis).

logical errors, giving this expression for p_L:

$$p_L = \frac{1}{2}\left(1 - (1 - 2\varepsilon_d)^t\right).$$

Solving for ε_d,

$$\varepsilon_d = \frac{1}{2}\left(1 - (1 - 2p_L)^{1/t}\right).$$

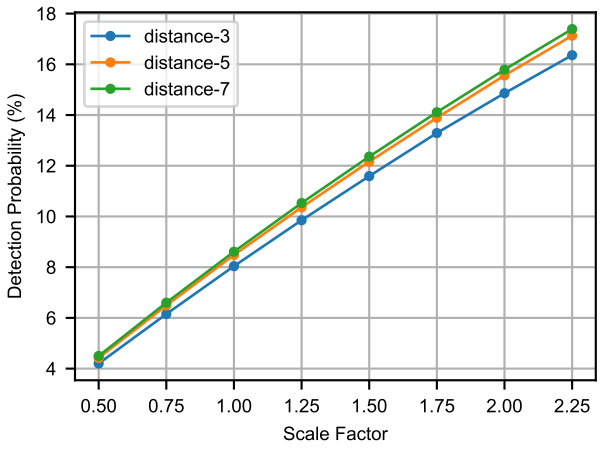

FIG. S22. **Pauli simulated detection event probabilities.** Average detection probability for weight-4 stabilizers predicted from a Pauli simulation of the 105-qubit processor based on Fig. S1. Each point is 10^4 shots of a 100 cycle experiment. Pauli error rates are obtained from the 105-qubit processor. All Pauli errors are scaled uniformly – 1.0 corresponds to the device Pauli rates. There, for $d = (3, 5, 7)$ we obtain predicted detection probabilities $p_{\text{det}} = (8.0\%, 8.5\%, 8.6\%)$. We attribute the upwards trend to diminishing finite-size effects. We see reasonable agreement with the experimental detection probabilities $p_{\text{det}} = (7.7\%, 8.5\%, 8.7\%)$.

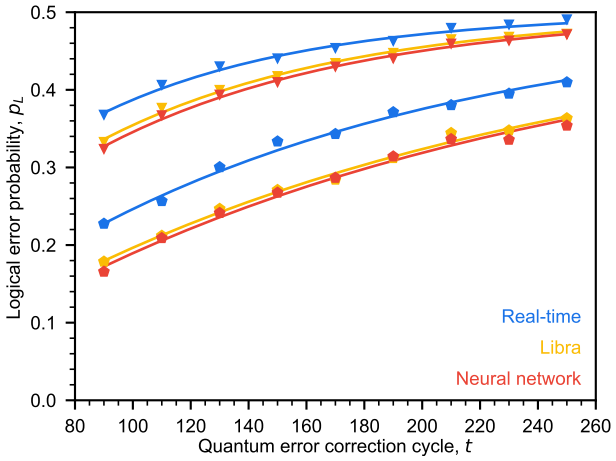

FIG. S23. **Decoder accuracy on the real-time dataset.** Underlying logical error probability data for Fig. 4c. Triangles: distance-3 (averaged over four quadrants and X and Z basis). Pentagons: distance-5 (averaged over X and Z basis). We compare the accuracy of three different decoding strategies. Fits begin from 90 cycles (rather than the usual 10) because these experiments only featured 5,000 shots per individual experiment, rather than the usual 50,000. Consequently, more of the early-cycle samples were used to train the decoder priors.

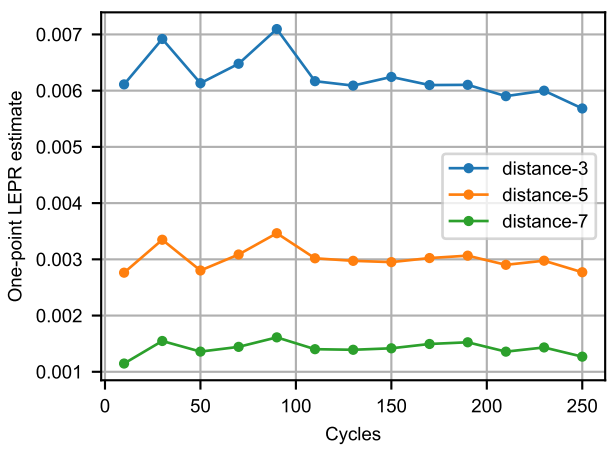

FIG. S24. **Error rates from one-point fits.** Here we plot the data from the Fig. 1 of the main text decoded with the neural network. For each cycle count r, we compute the one-point estimate of the logical error per cycle at that cycle count as $\epsilon_d = 0.5\left(1-(1-2p_L)^{1/r}\right)$, where p_L is the logical error fraction at r rounds, and average over basis and grid. We observe that the estimates remain relatively constant for different numbers of cycles, indicating that performance is not degrading for longer cycle counts.

d	$\ln A_d$	k	$k/((d+1)/2)$
3	-0.684 ± 0.06	1.708 ± 0.03	0.854 ± 0.015
5	0.198 ± 0.08	2.291 ± 0.04	0.764 ± 0.01
7	1.375 ± 0.1	3.023 ± 0.05	0.756 ± 0.01

TABLE S8. Fit values from Fig. 3b, error injection in the surface code. These are power law fits $\varepsilon_d = A_d p_{\text{det}}^k$. We fit a line to $\ln \varepsilon_d = \ln A_d + k \ln p_{\text{det}}$. The ratios in the final column average to approximately 0.8, leading to the observation of about 80% of the ideal value $(d+1)/2$ predicted by Eq. 1 of the main text.

We then propagate statistical uncertainties accordingly. The uncertainties become more noticeable for the low-error experiments (where relatively few errors are observed), such as the higher-distance points in Fig. 3a, where we have a statistical floor based on the total number of cycles in the dataset, 2×10^{10}. We also plot the logical error per cycle for Fig. 1c estimated from one cycle count for different cycles, and then averaged over grid and basis, in Fig. S24.

For clarity, we report the power law fits from the injection experiments reported in Figs. 2b and 3b, in Tables S8 and S10. We also report the fit values from the Pauli simulation of error injection in the surface code in Table S9.

d	$\ln A_d$	k	$k/((d+1)/2)$
3	0.462 ± 0.02	2.233 ± 0.009	1.117 ± 0.005
5	1.737 ± 0.02	3.136 ± 0.01	1.045 ± 0.003
7	3.390 ± 0.04	4.168 ± 0.02	1.042 ± 0.005

TABLE S9. Fit values from simulated error injection in the surface code (Pauli noise). These are power law fits $\varepsilon_d = A_d p_{\text{det}}^k$. We fit a line to $\ln \varepsilon_d = \ln A_d + k \ln p_{\text{det}}$. The ratios in the final column are close to 1, actually exceeding it slightly (compare to Table S8).

d	$\ln A_d$
3	-0.538 ± 0.001
5	0.078 ± 0.001
7	0.813 ± 0.002
9	1.635 ± 0.002
11	2.505 ± 0.003
13	3.396 ± 0.004
15	4.301 ± 0.005
17	5.223 ± 0.006
19	6.150 ± 0.007
21	7.077 ± 0.008
23	8.007 ± 0.010
25	8.955 ± 0.012
27	9.873 ± 0.015
29	10.783 ± 0.018

TABLE S10. Fit values from Fig. 3b, error injection in the repetition code. These are power law fits $\varepsilon_d = A_d p_{\text{det}}^{(d+1)/2}$. We fit a line to $\ln \varepsilon_d = \ln A_d + ((d+1)/2) \ln p_{\text{det}}$, just fitting the offset $\ln A_d$.

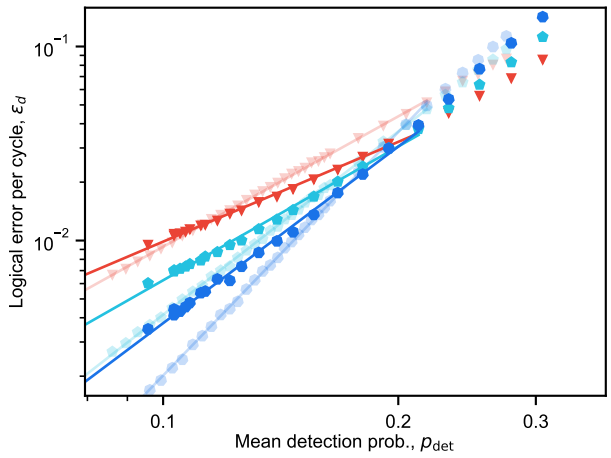

FIG. S25. **Experimental versus simulated injection.** Fig. 2b of the main text, but with simulated data overlaid semi-transparently. Note that the simulation uses stochastic Pauli channels rather than coherent error to make simulation efficient. Distance-3, -5, -7 shown in red, cyan, and blue. The fit parameters are presented in Table S8 and Table S9. Notably, the experimental error suppression does not achieve the full suppression predicted by the code distance, which we attribute to correlated error present in the device.

[1] Emerson, J., Alicki, R. & Życzkowski, K. Scalable noise estimation with random unitary operators. *Journal of Optics B: Quantum and Semiclassical Optics* **7**, S347 (2005). URL https://dx.doi.org/10.1088/1464-4266/7/10/021.

[2] Arute, F. *et al.* Quantum supremacy using a programmable superconducting processor. *Nature* **574**, 505–510 (2019). URL https://doi.org/10.1038/s41586-019-1666-5.

[3] Google Quantum AI. Suppressing quantum errors by scaling a surface code logical qubit. *Nature* **614**, 676–681 (2023). URL https://doi.org/10.1038/s41586-022-05434-1.

[4] Kaufman, R. *et al.* Josephson parametric amplifier with Chebyshev gain profile and high saturation. *Physical Review Applied* **20**, 054058 (2023). URL https://link.aps.org/doi/10.1103/PhysRevApplied.20.054058.

[5] Bengtsson, A. *et al.* Model-based optimization of superconducting qubit readout. *Physical Review Letters* **132**, 100603 (2024). URL https://link.aps.org/doi/10.1103/PhysRevLett.132.100603.

[6] Klimov, P. V., Kelly, J., Martinis, J. M. & Neven, H. The Snake optimizer for learning quantum processor control parameters. *arXiv preprint arXiv:2006.04594* (2020). URL https://doi.org/10.48550/arXiv.2006.04594.

[7] Klimov, P. V. *et al.* Optimizing quantum gates towards the scale of logical qubits. *Nature Communications* **15**, 2442 (2024). URL https://doi.org/10.1038/s41467-024-46623-y.

[8] Klimov, P. *et al.* Fluctuations of energy-relaxation times in superconducting qubits. *Physical Review Letters* **121**, 090502 (2018). URL https://link.aps.org/doi/10.1103/PhysRevLett.121.090502.

[9] Huang, S. *et al.* Microwave package design for superconducting quantum processors. *PRX Quantum* **2**, 020306 (2021). URL https://link.aps.org/doi/10.1103/PRXQuantum.2.020306.

[10] Foxen, B. *et al.* Demonstrating a continuous set of two-qubit gates for near-term quantum algorithms. *Phys. Rev. Lett.* **125**, 120504 (2020). URL https://link.aps.org/doi/10.1103/PhysRevLett.125.120504.

[11] Debroy, D. M. *et al.* Context-aware fidelity estimation. *Phys. Rev. Res.* **5**, 043202 (2023). URL https://link.aps.org/doi/10.1103/PhysRevResearch.5.043202.

[12] Arute, F. *et al.* Observation of separated dynamics of charge and spin in the fermi-hubbard model. *arXiv preprint arXiv:2010.07965* (2020). URL https://doi.org/10.48550/arXiv.2010.07965.

[13] Gross, J. A. *et al.* Characterizing coherent errors using matrix-element amplification. *arXiv preprint arXiv:2404.12550* (2024). URL https://doi.org/10.48550/arXiv.2404.12550.

[14] McCourt, T. *et al.* Learning noise via dynamical decoupling of entangled qubits. *Phys. Rev. A* **107**, 052610 (2023). URL https://link.aps.org/doi/10.1103/PhysRevA.107.052610.

[15] Jones, C. Improved accuracy for decoding surface codes with matching synthesis. *arXiv preprint arXiv:2408.12135* (2024). URL https://doi.org/10.48550/arXiv.2408.12135.

[16] Bausch, J. *et al.* Learning to decode the surface code with a recurrent, transformer-based neural network. *arXiv preprint arXiv:2310.05900* (2023). URL https://doi.org/10.48550/arXiv.2310.05900.

[17] Shutty, N., Newman, M. & Villalonga, B. Efficient near-optimal decoding of the surface code through ensembling. *arXiv preprint arXiv:2401.12434* (2024). URL https://doi.org/10.48550/arXiv.2401.12434.

[18] Fowler, A. G. Optimal complexity correction of correlated errors in the surface code. *arXiv preprint arXiv:1310.0863* (2013). URL https://doi.org/10.48550/arXiv.1310.0863.

[19] Paler, A. & Fowler, A. G. Pipelined correlated minimum weight perfect matching of the surface code. *Quantum* **7**, 1205 (2023). URL https://doi.org/10.22331/q-2023-12-12-1205.

[20] Google Quantum AI (2024). URL https://doi.org/10.5281/zenodo.13273331.

[21] Higgott, O., Bohdanowicz, T. C., Kubica, A., Flammia, S. T. & Campbell, E. T. Improved decoding of circuit noise and fragile boundaries of tailored surface codes. *Physical Review X* **13**, 031007 (2023). URL https://link.aps.org/doi/10.1103/PhysRevX.13.031007.

[22] Roffe, J., White, D. R., Burton, S. & Campbell, E. Decoding across the quantum low-density parity-check code landscape. *Physical Review Research* **2** (2020). URL https://link.aps.org/doi/10.1103/PhysRevResearch.2.043423.

[23] Gidney, C., Newman, M., Fowler, A. & Broughton, M. A fault-tolerant honeycomb memory. *Quantum* **5**, 605 (2021). URL https://doi.org/10.22331/q-2021-12-20-605.

[24] Sivak, V., Newman, M. & Klimov, P. Optimization of decoder priors for accurate quantum error correction. *arXiv preprint arXiv:2406.02700* (2024). URL https://doi.org/10.48550/arXiv.2406.02700.

[25] Higgott, O. & Gidney, C. Sparse Blossom: correcting a million errors per core second with minimum-weight matching. *arXiv preprint arXiv:2303.15933* (2023). URL https://doi.org/10.48550/arXiv.2303.15933.

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

[27] Higgott, O. PyMatching: A python package for decoding quantum codes with minimum-weight perfect matching. *ACM Transactions on Quantum Computing* **3**, 1–16 (2022). URL https://doi.org/10.1145/3505637.

[28] Cho, K. *et al.* Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734 (Association for Computational Linguistics, Stroudsburg, PA, USA, 2014). URL https://aclanthology.org/D14-1179.

[29] Gottesman, D. & Chuang, I. L. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature* **402**, 390–393 (1999). URL https://doi.org/10.1038/46503.

[30] Gidney, C. & Fowler, A. G. Flexible layout of surface code computations using AutoCCZ states. *arXiv preprint arXiv:1905.08916* (2019). URL https://doi.org/10.48550/arXiv.1905.08916.

[31] Terhal, B. M. Quantum error correction for quantum memories. *Reviews of Modern Physics* **87**, 307–346 (2015). URL https://link.aps.org/doi/10.1103/RevModPhys.87.307.

[32] Gidney, C. & Ekerå, M. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum* **5**, 433 (2021). URL https://doi.org/10.22331/q-2021-04-15-433.

[33] Nagayama, S., Fowler, A. G., Horsman, D., Devitt, S. J. & Van Meter, R. Surface code error correction on a defective lattice. *New Journal of Physics* **19**, 023050 (2017). URL https://doi.org/10.1088/1367-2630/aa5918.

[34] McEwen, M., Bacon, D. & Gidney, C. Relaxing hardware requirements for surface code circuits using time-dynamics. *Quantum* **7**, 1172 (2023). URL https://doi.org/10.22331/q-2023-11-07-1172.

[35] Wu, Y. & Zhong, L. Fusion Blossom: Fast MWPM decoders for QEC. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 1, 928–938 (IEEE, 2023). URL https://doi.ieeecomputersociety.org/10.1109/QCE57702.2023.00107.

[36] Cirq Developers. Cirq (2024). URL https://doi.org/10.5281/zenodo.11398048.

[37] Miao, K. C. *et al.* Overcoming leakage in quantum error correction. *Nature Physics* **19**, 1745–2481 (2023). URL https://doi.org/10.1038/s41567-023-02226-w.

[38] Stephens, A. M. Fault-tolerant thresholds for quantum error correction with the surface code. *Physical Review A* **89**, 022321 (2014). URL https://link.aps.org/doi/10.1103/PhysRevA.89.022321.

[39] Nielsen, M. A. A simple formula for the average gate fidelity of a quantum dynamical operation. *Physics Letters A* **303**, 249–252 (2002). URL https://doi.org/10.1016/S0375-9601(02)01272-0.

[40] Grimm, A. *et al.* Stabilization and operation of a Kerr-cat qubit. *Nature* **584**, 205–209 (2020). URL https://doi.org/10.1038/s41586-020-2587-z.

[41] Chen, Z. *et al.* Exponential suppression of bit or phase errors with cyclic error correction. *Nature* **595**, 383–387 (2021). URL https://doi.org/10.1038/s41586-021-03588-y.

[42] Réglade, U. *et al.* Quantum control of a cat qubit with bit-flip times exceeding ten seconds. *Nature* 1–6 (2024). URL https://doi.org/10.1038/s41586-024-07294-3.

[43] Bravyi, S., Englbrecht, M., König, R. & Peard, N. Correcting coherent errors with surface codes. *npj Quantum Information* **4**, 55 (2018). URL https://doi.org/10.1038/s41534-018-0106-y.

[44] Sivak, V. *et al.* Real-time quantum error correction beyond break-even. *Nature* **616**, 50–55 (2023). URL https://doi.org/10.1038/s41586-023-05782-6.

[45] Gidney, C. Inplace access to the surface code Y basis. *Quantum* **8**, 1310 (2024). URL https://doi.org/10.22331/q-2024-04-08-1310.

[46] Gidney, C., Newman, M., Brooks, P. & Jones, C. Yoked surface codes. *arXiv preprint arXiv:2312.04522* (2023). URL https://doi.org/10.48550/arXiv.2312.04522.

[47] Ofek, N. *et al.* Extending the lifetime of a quantum bit with error correction in superconducting circuits. *Nature* **536**, 441–445 (2016). URL https://doi.org/10.1038/nature18949.

[48] Ni, Z. *et al.* Beating the break-even point with a discrete-variable-encoded logical qubit. *Nature* **616**, 56–60 (2023). URL https://doi.org/10.1038/s41586-023-05784-4.

[49] McEwen, M. *et al.* Resolving catastrophic error bursts from cosmic rays in large arrays of superconducting qubits. *Nature Physics* **18**, 107–111 (2022). URL https://doi.org/10.1038/s41567-021-01432-8.