Composing Detector Error Models

Vadym Kliuchnikov, Justin Lietz, and Fernando Pastawski

NVIDIA Corporation

Fault-tolerant quantum programs thread together reusable logical operations, yet correcting errors requires comparing measurements across operation boundaries. Correlations tie each operation's error analysis to the computation around it. Adaptive computation makes this a runtime challenge: measurement results determine which operation comes next, so the error analysis must keep pace with execution.

We introduce extended detector error models (EDEMs) for stabilizer circuits with stochastic Pauli faults. EDEMs compose in sequence and in parallel, mirroring the composition of physical circuit realizations and their detector contracts. We prove how detectors, the measurement parities used to diagnose errors, can be split across circuit boundaries, and establish conditions under which composition recovers the assembled circuit's detector error model. This structure supports symbolic precompilation and analysis of entire families of circuits. It also defines error models for decoding windows and the boundary information through which committed corrections affect subsequent windows. The same interface thus connects the design of individual logical operations to the decoding of an unfolding quantum program.

1 Introduction

1.1 Overview and main result

Fault-tolerant quantum computation protects logical information with stabilizer codes and repeatedly measures redundant parities to diagnose noise [1–3]. In the absence of noise, certain parities of measurement outcomes are deterministic; these parities are called *detectors* [4–6]. A *detector error model* (DEM) records how each elementary fault flips detectors and logical observables, together with the fault probabilities. Decoders such as minimum-weight matching [7], union-find [8], and belief propagation with post-processing [9] are configured from this data, and established tools construct it from the complete physical circuit [4].

The fault-tolerant computations we target are adaptive: measurement outcomes and logical control decisions determine parts of the physical circuit during execution. Consequently, the complete physical circuit is not available in advance. Logical operations are chosen adaptively, based on decoded outcomes that fix Pauli or Clifford frame corrections [10–14], and decoding must keep pace with the physical circuit as it executes [3, 15–17]. The DEM must therefore be constructed incrementally, in tandem with the circuit. Physical circuits for such computations are assembled from *gadgets*: circuit fragments that provably implement specific logical operations on encoded information [18–20]. This suggests a division of labor: analyze each gadget type once, offline, and assemble the DEM of any logical circuit from the per-gadget results, without revisiting the gadgets' physical circuits.

The obstruction is that detectors do not respect gadget boundaries. A detector typically compares syndrome information extracted by different gadgets, so a gadget partition of the circuit does not induce a partition of its detectors. We address this in two steps. First, we prove that detectors can be cut at circuit boundaries: every detector of a serial composition splits into parts local to the pieces once the boundary is extended by a small number of extra outcome bits (Sec. 3). The proof is constructive and preserves a chosen detector basis, and the composed basis is complete whenever the component bases are. At gadget boundaries, these extra bits can be taken to

Vadym Kliuchnikov: vkliuchnikov@nvidia.com
Justin Lietz: jlietz@nvidia.com
Fernando Pastawski: fpastawski@nvidia.com

be *virtual syndromes*: outcomes of virtual stabilizer measurements inserted at the boundary, following Ref. [21, 22]. Second, we define the *extended detector error model* (EDEM) of a gadget (Sec. 4): a linear map over $\mathbb{F}_2$ that takes the gadget's elementary faults and its incoming interface data—virtual-syndrome flips and boundary-error coordinates (Sec. 4.2)—to detector and observable flips and the corresponding outgoing interface data. EDEMs compose the way gadgets do: wiring the per-gadget maps according to the logical circuit and contracting the result recovers the DEM of the full circuit, for the detector basis induced by the gadget detector contracts, provided connected ports carry matching codes and code presentations (Sec. 4.2).

Because composition is linear-algebraic, constructing a DEM becomes evaluating a wired diagram of sparse block matrices, and the same few blocks appear many times. This makes a symbolic approach effective (Sec. 5): assign a variable to each distinct block, contract the diagram once into polynomial expressions in these variables, and evaluate each distinct expression once. The symbolic form also supports structural arguments; for syndrome-extraction gadgets satisfying mild assumptions, we derive closed-form expressions establishing the banded structure of the repeated-extraction part of memory-experiment DEMs.

1.2 Consequences and applications

Modular logical instruction-set design. An EDEM provides a compositional error-model interface for a logical gadget. Gadget implementations can therefore be analyzed, compared, and optimized independently, then assembled into larger computations without flattening and reanalyzing their physical circuits, provided that connected code and code-presentation interfaces match (Sec. 4.2). In this way, EDEMs support the fault-tolerance and decoding layer of a logical instruction-set architecture built from reusable, independently characterized components. They also provide a self-similar structure naturally suited to describing concatenated fault-tolerant protocols.

Incremental construction. Per-gadget EDEMs can be computed and stored ahead of time, once per gadget type. Extending the DEM by one more gadget call at runtime then touches only these stored maps and the interface data of its ports, which suits adaptive computations whose instruction stream is decided on the fly (Sec. 7).

Sliding-window decoding. The same interfaces define the decoding problem of a window: the EDEM of any convex set of gadget calls (Sec. 7). Causality makes the channel-check matrix block lower triangular with respect to the gadget circuit, and detector span, read off the per-gadget blocks, makes it block local, reducing to block banded for chain-like schedules. Committed corrections act on later windows only through their interface outputs, so residual errors are carried forward as boundary and virtual-syndrome data rather than as retained detector history.

Compact representation. The composed DEM is a block matrix with few distinct blocks, so it can be stored as a grid of pointers into a small set of blocks. This reduces memory traffic, for example when transferring DEMs to accelerators for decoding.

Amortization across experiments. Families of experiments that share a fault-tolerance protocol share polynomial expressions, so symbolic evaluations can be cached and reused across many circuits using the same fault-tolerant instruction set, and many circuit instantiations, such as sweeps over distances, rounds, or noise parameters (Secs. 5.3 and 6).

1.3 Contributions and organization

After Sec. 2, which fixes notation, reviews stabilizer codes, stabilizer circuits, DEMs, gadgets, and the modularization of detectors by virtual syndromes [21, 22], and introduces the channel-level conventions used throughout (boundary constraints, detector and outcome maps, and the diagram notation for linear maps), our contributions are:

- a detector-cutting theorem showing that every detector of a composed stabilizer circuit splits across a serial cut, constructively and preserving completeness of the detector basis (Sec. 3);

- detector maps with virtual-syndrome interfaces and extended detector error models, together with the composition rule that recovers the DEM of a gadget composition from the per-gadget EDEMs (Secs. 3 and 4);

- a symbolic evaluation method for EDEM diagrams, illustrated by a closed-form derivation of the structure of repeated-syndrome-extraction DEMs (Sec. 5);

- a Rust prototype that implements the formalism for four code families and reproduces the DEMs Stim derives from the flat circuits (Sec. 6);

- window EDEMs as sub-diagram contractions, a commit-and-fold rule and interface dispositions for sequential decoding, a per-gadget criterion for detector span, and a distinction between quantum execution and decoding dependencies, with interface requirements for schedules that use different orders (Sec. 7).

Limitations and next steps are discussed in Sec. 8.

Relation to prior work. Modular, analyze-each-gadget-once error analysis is not new. The `deq` system [21, 22] and the framework of Ref. [20] modularize detectors with virtual stabilizer measurements, analyze each gadget type offline, and assemble decoding problems, including sliding windows, online; our detector decomposition builds directly on these ideas. Stim [4] provides stabilizer flows [23], and the Stim project's `stimflow` library [24] composes chunks carrying these relations and completes detectors across piece boundaries. In that workflow, composition produces a circuit from which the DEM is then derived monolithically. Our contribution relative to both is the uniform linear-map form of the gadget error model, the accompanying detector-cutting proofs, and the symbolic layer this uniformity enables. The linear-algebraic view of fault effects follows the spacetime and outcome-code perspectives on Clifford circuits [19, 20, 25]; DEMs as design objects appear in Ref. [6], and detectors as design objects in Ref. [5]. Compositional descriptions of fault-tolerant protocols at the operator level include logical blocks, fusion-based computation, and ZX-calculus approaches [26–29]; these compose the protocols themselves, whereas we compose their error models. Windowed and modular decoding are reviewed where we use them (Sec. 7).

2 Preliminaries

This section establishes notation and reviews the background used throughout the paper. Sections 2.1, 2.4, and 2.5 recall standard material: stabilizer codes and encoders, stabilizer circuits with Pauli faults [19, 20], and detector error models [4, 6]. Section 2.6 reviews gadgets [19], and Sec. 2.7 reviews the modularization of detectors using virtual syndromes introduced in Ref. [21]. Sections 2.2, 2.3, and 2.8 fix this paper's conventions: stabilizer channels with declared classical ports and their boundary constraints, which are the stabilizer flows of Ref. [23] written at the channel level; detectors, observables, and outcome maps in that language; and the diagram notation for linear maps over $\mathbb{F}_2$. Apart from these conventions, the material in this section is standard; our results begin in Sec. 3. A reader familiar with stabilizer codes, stabilizer circuits, and detector error models may skim Secs. 2.1, 2.4, and 2.5.

2.1 Notation and stabilizer codes

We recall standard facts about stabilizer codes; see, e.g., Ref. [3]. Let $\mathcal{P}_n$ denote the n-qubit Pauli group, with phases $\{\pm 1, \pm i\}$ retained because the signs of Pauli observables encode measurement outcomes and syndromes. A *Pauli observable* is a Hermitian element of $\mathcal{P}_n$, and $\mathcal{C}_n$ denotes the n-qubit Clifford group. For a bitvector $v \in \mathbb{F}_2^{\mathsf{R}}$ indexed by a register R, we write $X_{\mathsf{R}}^v := \bigotimes_{j\in\mathsf{R}} X_j^{v_j}$, and similarly Z_{R}^v. The *phase-free Pauli group* $\mathcal{P}_n/\langle i\rangle$ is identified with $\mathbb{F}_2^{2n}$ through $(x, z) \mapsto X^x Z^z$.

A *stabilizer group* $S \subseteq \mathcal{P}_n$ is an abelian group of Pauli observables that does not contain $-I$. Its *codespace* is the joint $+1$ eigenspace of S, with projector $\Pi_S = |S|^{-1} \sum_{s\in S} s$; if S has $n-k$ independent generators, the codespace encodes k logical qubits, and for $k = 0$ it is spanned by a single *stabilizer state*. Physical representatives of logical Pauli operators are the Paulis that commute with S, and two representatives that differ by an element of S act identically on the codespace. A *logical interpretation* of the code fixes an isomorphism

$$\iota_S : \mathcal{P}_k \longrightarrow S^\perp/S, \qquad \iota_S(\ell) = [\bar{\ell}] := \bar{\ell}S, \tag{1}$$

where $S^\perp$ denotes the group of Paulis that commute with S; it suffices to fix representatives $\bar{X}_1, \bar{Z}_1, \ldots, \bar{X}_k, \bar{Z}_k$. Throughout the paper, every

code has a fixed choice of generators $s_1, \dots, s_{n-k}$ and of logical representatives; gadgets compose only when these choices agree on connected ports (Sec. 2.6), and the code presentations of Sec. 4.2 additionally allow overcomplete generator lists.

The code and its logical interpretation specify an *encoding isometry* $E_{\mathbf{0}} : (\mathbb{C}^2)^{\otimes k} \to (\mathbb{C}^2)^{\otimes n}$, characterized by

$$\begin{aligned} E_{\mathbf{0}}^\dagger E_{\mathbf{0}} &= I, \\ sE_{\mathbf{0}} &= E_{\mathbf{0}} \quad (s \in S), \\ \bar{\ell} E_{\mathbf{0}} &= E_{\mathbf{0}} \ell \quad (\ell \in \mathcal{P}_k). \end{aligned} \tag{2}$$

As in Ref. [19], fix an *encoding unitary*, a Clifford unitary $E : (\mathbb{C}^2)^{\otimes k} \otimes (\mathbb{C}^2)^{\otimes n-k} \to (\mathbb{C}^2)^{\otimes n}$ extending $E_{\mathbf{0}}$, whose second input is the *syndrome register* associated with the chosen generators. Its restrictions $E_\sigma := E(I \otimes |\sigma\rangle)$ to syndrome values $\sigma \in \mathbb{F}_2^{n-k}$ satisfy

$$\begin{aligned} s_i E_\sigma &= (-1)^{\sigma_i} E_\sigma \quad (i = 1, \dots, n-k), \\ \bar{\ell} E_\sigma &= E_\sigma \ell \qquad\quad (\ell \in \mathcal{P}_k), \end{aligned} \tag{3}$$

so that $E_{\mathbf{0}}$ is the encoding isometry and the other E_σ identify every syndrome sector with the same logical space. The *encoder* is the channel with a k-qubit quantum input and a declared classical syndrome input that prepares a logical state in the requested syndrome sector,

$$\mathcal{E}(\rho_L \otimes |\sigma\rangle\langle\sigma|) := E_\sigma \rho_L E_\sigma^\dagger, \tag{4}$$

and the *unencoder* is its inverse, the channel

$$\mathcal{U}(\rho) := \sum_{\sigma \in \mathbb{F}_2^{n-k}} E_\sigma^\dagger \rho E_\sigma \otimes |\sigma\rangle\langle\sigma|, \tag{5}$$

whose quantum output is the k-qubit logical state and whose declared classical output is the syndrome of the chosen generators. Indeed, $\mathcal{U} \circ \mathcal{E}$ is the identity, and $\mathcal{E} \circ \mathcal{U}$ acts as the identity on every state supported in a single syndrome sector. Reading out the syndrome of the unencoder is the same as measuring the generators $s_1, \dots, s_{n-k}$, so an unencoder followed by an encoder is a nondestructive measurement of the generators.

Lemma 2.1 (Pull-back of Paulis through the encoding unitary)**.** *For every $P \in \mathcal{P}_n$, there are a logical Pauli $\ell_P \in \mathcal{P}_k$ and bitvectors $c(P), z(P) \in \mathbb{F}_2^{n-k}$ such that, up to a phase,*

$$E^\dagger P E = \ell_P \otimes X^{c(P)} Z^{z(P)}, \tag{6}$$

where $c_i(P) = 1$ precisely when P anticommutes with s_i. The pair $(c(P), \ell_P)$ depends only on the coset PS. On states whose syndrome register is in $|\mathbf{0}\rangle$, the factor $Z^{z(P)}$ acts trivially.

Proof. By Eq. (3), $E^\dagger s_i E = I \otimes Z_i$ and $E^\dagger \bar{\ell} E = \ell \otimes I$. The Clifford conjugate $E^\dagger P E$ is a Pauli, and it anticommutes with $I \otimes Z_i$ exactly when P anticommutes with s_i, which fixes its X-part on the syndrome register to be $X^{c(P)}$. Multiplying P by $s \in S$ changes only the Z-part on the syndrome register, and $Z^z|\mathbf{0}\rangle = |\mathbf{0}\rangle$. □

We call $c(P)$ the *syndrome* and ℓ_P the *logical label* of P; Secs. 4.2 and 5.2 use them as the coordinates of boundary errors.

2.2 Stabilizer channels and boundary constraints

Stabilizer circuits with measurements, discards, and classically controlled Paulis implement quantum channels, and we formulate our discussion in terms of these channels. For an operator $Q : \mathcal{H}_A \to \mathcal{H}_B$, let $|Q\rangle\!\rangle := (Q \otimes I)|\Omega_A\rangle$ be its vectorization, where $|\Omega_A\rangle = \sum_x |x\rangle_A |x\rangle_{A'}$ is the unnormalized maximally entangled vector with a reference copy A'; then $|LQR\rangle\!\rangle = (L \otimes R^T)|Q\rangle\!\rangle$, so left and right Pauli actions on Q become Pauli stabilizer constraints on $|Q\rangle\!\rangle$. We call a nonzero Q a *stabilizer operator* if $|Q\rangle\!\rangle$ is proportional to a stabilizer state; Clifford unitaries, stabilizer projectors, and encoding isometries are examples. A *stabilizer channel* is a completely positive trace-preserving map Φ whose Choi operator $J(\Phi) := (\Phi \otimes \mathrm{id}_{A'})(|\Omega_A\rangle\langle\Omega_A|)$ is a stabilizer operator [30]. Equivalently, Φ is fixed up to a scalar by $2(n_{\text{in}} + n_{\text{out}})$ independent equations of the four-slot form

$$P_{\text{out,L}} \Phi(P_{\text{in,L}} \rho P_{\text{in,R}}) P_{\text{out,R}} = \Phi(\rho), \tag{7}$$

holding for every ρ, and we write $\mathsf{Stab}(\Phi)$ for the group of all such equations. For a stabilizer operator Q, the map $\mathrm{Ad}_Q : \rho \mapsto Q\rho Q^\dagger$ is a stabilizer channel whenever it is trace preserving; tensor products and serial compositions of stabilizer channels are stabilizer channels [30]. Appendix A records the Choi-operator details, verifies that the elementary operations of Sec. 2.4 are stabilizer channels, and compares this notion with the trace-non-increasing *stabilizer maps* and the circuits of Ref. [19]. The one structural fact about stabilizer

channels used in the main text is that Pauli errors propagate through them as Pauli errors.

The *adjoint* $\Phi^\dagger$ of a channel Φ is the unital completely positive map defined by $\mathrm{Tr}(\Phi(X)^\dagger Y) = \mathrm{Tr}(X^\dagger \Phi^\dagger(Y))$ for all operators X and Y; it preserves Hermiticity and reverses the order of composition, $(\Phi_2 \circ \Phi_1)^\dagger = \Phi_1^\dagger \circ \Phi_2^\dagger$.

Lemma 2.2 (Pauli covariance)**.** *Let Φ be a stabilizer channel from register A to register B. For every Pauli R on A, there is a Pauli Q on B such that*

$$\Phi \circ \mathrm{Ad}_R = \mathrm{Ad}_Q \circ \Phi. \tag{8}$$

Dually, for every Pauli P on B, the adjoint $\Phi^\dagger(P)$ is either zero or a Pauli on A up to a sign.

The first statement is proved via a stabilizer dilation in Appendix A (Corollary A.2); the dual statement is Theorem 1 of Ref. [30].

Classical ports. We model classical bits as qubits so that a single formalism covers quantum wires and measurement records. With $\mathcal{T}_Z(\rho) := (\rho + Z\rho Z)/2$ the dephasing channel, which is a stabilizer channel, a register A_C of input ports or B_C of output ports is *declared classical* when

$$\Phi \circ \mathcal{T}_Z^{\otimes A_C} = \Phi, \qquad \mathcal{T}_Z^{\otimes B_C} \circ \Phi = \Phi, \tag{9}$$

respectively, where the twirl acts as the identity on the remaining ports. Since $\mathcal{T}_Z = (\mathrm{id} + \mathrm{Ad}_Z)/2$, these conditions are the stabilizer equations

$$\begin{aligned} \Phi(Z_i \rho Z_i) &= \Phi(\rho) \quad (i \in A_C), \\ Z_j \Phi(\rho) Z_j &= \Phi(\rho) \quad (j \in B_C), \end{aligned} \tag{10}$$

which are elements of $\mathsf{Stab}(\Phi)$. A declaration records that a port admits a classical implementation and fault model; it does not alter the map. The consequence we use below is that conjugating a declared classical output by Z leaves the image of Φ invariant, so measurement outcomes are affected by faults only through bit flips.

Boundary constraints. Suppose Φ maps an input register A to a quantum output register B and a declared classical outcome register with port set M, and write $m \in \mathbb{F}_2^{\mathsf{M}}$ for an outcome vector. For $a \in \mathbb{F}_2^{\mathsf{M}}$ and $b \in \mathbb{F}_2$, let $Z_{\mathsf{M}}(a,b) := (-1)^b Z_{\mathsf{M}}^a$, so that $Z_{\mathsf{M}}(a,b)|m\rangle = (-1)^{a^T m + b}|m\rangle$.

Definition 2.3 (Boundary constraint)**.** A tuple $(P_{\mathrm{in}}, P_{\mathrm{out}}, a, b)$ of a Pauli P_{in} on A, a Pauli P_{out} on B, and an affine outcome parity is a *boundary constraint* of Φ when

$$(P_{\mathrm{out}} \otimes Z_{\mathsf{M}}(a,b))\Phi(P_{\mathrm{in}}\rho) = \Phi(\rho) \quad \text{for every } \rho. \tag{11}$$

The boundary constraints form the *boundary-constraint group* $R_{\mathsf{M}}(\Phi)$, the subgroup of one-sided elements of $\mathsf{Stab}(\Phi)$, i.e., of the equations (7) with trivial right-action slots.

The Pauli parts determine how to interpret a boundary constraint. When $P_{\mathrm{in}} = P_{\mathrm{out}} = I$, it is a deterministic outcome parity; when only $P_{\mathrm{out}} = I$, it states that Φ measures the input Pauli P_{in} with outcome $a^T m + b$; when only $P_{\mathrm{in}} = I$, it states that P_{out} stabilizes the output with sign $(-1)^{a^T m + b}$; and when both are nontrivial, it transports a logical correlation from input to output. A boundary constraint is a *stabilizer flow* $P_{\mathrm{in}} \to P_{\mathrm{out}}$ with outcome parity $a^T m + b$ in the sense of Ref. [23], as implemented in Stim [4], written at the channel level; equivalently, it is a check of the outcome code or spacetime code of the circuit [19, 25]. The Paulis P_{out} of constraints with $P_{\mathrm{in}} = I$ form the *output stabilizer group* of Φ, with outcome-dependent signs. Likewise, the Paulis P_{in} of constraints with $P_{\mathrm{out}} = I$ form its *measured group*. Boundary constraints compose serially: if $(P_{\mathrm{in}}, P, a_1, b_1) \in R_{\mathsf{M}_1}(\Phi_1)$ and $(P, P_{\mathrm{out}}, a_2, b_2) \in R_{\mathsf{M}_2}(\Phi_2)$, then $(P_{\mathrm{in}}, P_{\mathrm{out}}, (a_1, a_2), b_1 + b_2) \in R_{\mathsf{M}_1 \sqcup \mathsf{M}_2}(\Phi_2 \circ \Phi_1)$, because $\Phi_2(P\sigma) = (P_{\mathrm{out}} \otimes Z_{\mathsf{M}_2}(a_2, b_2))\Phi_2(\sigma)$ for every σ. In particular, if Φ_1 prepares P with sign $a_1^T m_1 + b_1$ and Φ_2 measures P with outcome $a_2^T m_2 + b_2$, then $a_1^T m_1 + a_2^T m_2 + b_1 + b_2$ is a deterministic parity of $\Phi_2 \circ \Phi_1$.

2.3 Detectors, observables, and outcome maps

In the absence of faults, certain parities of measurement outcomes are deterministic; these parities are called *detectors* [4–6].

Definition 2.4 (Detectors)**.** A *detector* of Φ is an affine expression $\langle a, m\rangle + b$ in its outcomes, with $\langle a, m\rangle := a^T m$, that equals zero for every outcome vector that Φ can produce from any input state; equivalently, $(I, I, a, b) \in R_{\mathsf{M}}(\Phi)$. If Φ has declared classical input ports S, the expression may also involve the classical inputs s, taking the form $\langle a_0, s\rangle + \langle a, m\rangle + b$, and must vanish for every value of s; equivalently, $(Z_{\mathsf{S}}^{a_0}, I, a, b) \in R_{\mathsf{M}}(\Phi)$. A detector is *nontrivial* if its linear part

is nonzero and *supported on* a set of ports if its linear part vanishes outside that set. A finite family of detectors, labelled by a set D, is *declared* when it is supplied to the decoder; it is a *detector basis* if it is linearly independent, and *complete* if it generates every deterministic parity of Φ. The observed value of a declared detector is its *detection event*.

For example, take three data qubits in an arbitrary state, measure the checks Z_1Z_2 and Z_2Z_3 of the three-qubit repetition code twice, with outcomes $m_j^{(1)}$ and $m_j^{(2)}$ for $j = 1, 2$, and then measure the data qubits destructively with outcomes n_1, n_2, n_3. The detectors $m_j^{(1)} + m_j^{(2)}$ and $m_j^{(2)} + n_j + n_{j+1}$ form a complete basis, and n_1 reads out the logical Z of the code.

Definition 2.5 (Redundant outcomes and equivalence)**.** An outcome bit is *redundant* given a set of other outcomes if some detector contains it and is otherwise supported on that set; equivalently, it is an affine function of those outcomes on every attainable outcome vector. Two stabilizer circuits C and C' with the same quantum ports are *equivalent under an affine outcome map* f from the outcomes of C' to those of C if $[\![C]\!] = (\mathrm{id} \otimes \Phi_f) \circ [\![C']\!]$, where Φ_f applies f to the outcome register and discards the rest.

Inserting a measurement-only subcircuit whose outcomes are all redundant after, such as the virtual measurements of Sec. 2.7, yields an equivalent circuit.

For a closed experiment, deterministic parities not declared as detectors may be reported as deterministic logical observables instead. An *outcome map* reports declared logical outcomes, called logical observables, as affine functions of the measurement outcomes. For a gadget, these are the outcomes of its logical action (Def. 2.8), and they need not be deterministic. A *deterministic outcome map* reports selected deterministic parities of these logical outcomes, identified using detectors of the logical action. Let O label the declared logical observables. Collecting the declared detectors and observables gives the affine *detector map* and *outcome map*

$$\mathrm{DM} : \mathbb{F}_2^{\mathsf{M}} \longrightarrow \mathbb{F}_2^{\mathsf{D}}, \qquad \mathrm{OM} : \mathbb{F}_2^{\mathsf{M}} \longrightarrow \mathbb{F}_2^{\mathsf{O}}, \quad (12)$$

whose components are the declared detector and observable values. For an affine map $f : \mathbb{F}_2^{\mathsf{X}} \to \mathbb{F}_2^{\mathsf{Y}}$, we write Δf for its linear part,

$$(\Delta f)(u) := f(u) - f(0) = f(x + u) - f(x), \quad (13)$$

and Δx for a change in a bitvector x, so that $\Delta y = (\Delta f)(\Delta x)$ whenever $y = f(x)$ and $\Delta(g \circ f) = (\Delta g) \circ (\Delta f)$. Thus ΔDM and ΔOM map outcome flips to detector and observable flips. Definition 2.8 below extends the outcome map to gadgets, whose observables are the outcomes of a logical circuit, and Definition 3.6 extends the detector map to gadgets with virtual-syndrome ports; these extensions preserve the definitions above.

2.4 Stabilizer circuits and Pauli faults

Following Ref. [19], a stabilizer circuit is assembled from a fixed set $\mathcal{G}$ of *elementary operations*. Each operation $g \in \mathcal{G}$ has finite labelled sets of input and output ports, a *size* for each port (the number of qubits or bits it carries), and an interpretation $[\![g]\!]$ as a stabilizer channel between the corresponding registers. The sets $\mathcal{G}$ used in this paper consist of qubit preparations, Clifford unitaries, Pauli measurements with a declared classical outcome port, discards, fair-coin allocations, and Pauli corrections controlled by affine parities of earlier outcomes; Appendix A verifies that these are stabilizer channels.

Definition 2.6 (Stabilizer circuit)**.** A *stabilizer circuit* C over $\mathcal{G}$ consists of a finite set of boxes, each labelled by an operation $g \in \mathcal{G}$ and carrying its input and output ports; finite labelled sets of *boundary input ports* and *boundary output ports*, each with a specified size, which are the inputs and outputs of C itself; and a bijective *wiring* that pairs every source port (a boundary input port or a box output port) with a target port (a boundary output port or a box input port) of the same size. The directed graph on the boxes induced by the wires must be acyclic.

The boundary input ports carry the quantum inputs and declared classical inputs of the circuit, and the boundary output ports carry its quantum outputs and declared classical outcomes. A subnetwork of a stabilizer circuit is *convex* if no directed path of wires leaves it and then re-enters it. With the wires crossing its boundary treated as boundary ports, a convex subnetwork is itself a stabilizer circuit and may be treated as a single box, whose interpretation is the channel defined next. Convexity ensures that the resulting

network remains acyclic (Def. 4.7). The *interpretation* $[\![C]\!]$ of a stabilizer circuit is the stabilizer channel from its boundary input ports to its boundary output ports obtained by composing the channels $[\![g]\!]$ along the wires, in any topological order of the boxes; we write

$$\Phi_C := [\![C]\!] \tag{14}$$

and reuse the brackets for gadget circuits and map diagrams below, which are networks of the same shape. A circuit is *closed* if it has no quantum boundary ports, so its only outputs are declared classical outcomes; otherwise it is *open*.

Lemma 3.2 and the verification conditions of Secs. 4.2 and 6 use the following normal form for the outcomes of a stabilizer circuit, as computed by Clifford simulation [19].

Proposition 2.7 (General form of outcomes)**.** *The outcome vector of a stabilizer circuit is an affine function $m = M(r, l)$ of a uniformly random bitvector r and, for an open circuit, a bitvector l whose entries are eigenvalues of input Pauli observables measured by the circuit. The random bits of a circuit are independent of those of any circuit composed after it. The attainable outcome vectors form an affine code, the* outcome code *of Refs. [19, 25], whose check equations are exactly the deterministic parities, i.e., the parities constant in (r, l).*

The proof is the usual stabilizer-tableau argument: a Pauli measurement whose observable anticommutes with a current stabilizer yields a fresh fair coin, one whose observable lies in the current stabilizer group yields an affine function of earlier coins and the constant signs, and for open circuits the remaining case records an input observable.

Following Ref. [20], we separate the structural specification of the allowed faults from the probabilities assigned to them. An *elementary Pauli fault* is a Pauli insertion on a quantum wire, a bit flip on a classical wire, or a fixed collection of such insertions representing a correlated event; in the ambient qubit representation, a bit flip is a Pauli X, so a circuit with any fixed set of inserted faults is again a stabilizer circuit. Let $\mathcal{F}$ be a finite set of allowed elementary faults with label set F, $\alpha \mapsto f_\alpha$. A *fault configuration* is a bitvector $v \in \mathbb{F}_2^{\mathsf{F}}$ selecting the faults that occur, and Φ_v denotes the channel of the circuit with those faults inserted in causal order. The *reference fault set* $\mathcal{F}_0$ consists of Pauli X and Z insertions at every chosen quantum fault location and X flips at every chosen classical fault location; general Pauli faults are products of its elements.

2.5 Detector error models

We recall the detector error model of a closed experiment [4, 6, 20]. Treat each allowed fault as a Pauli conjugation of the circuit channel. By Lemma 2.2 and Eq. (10), its only effect on the declared outcomes is a bit flip, so there is a matrix $A_{\mathsf{M}} \in \mathbb{F}_2^{\mathsf{M}\times\mathsf{F}}$ whose column α is the outcome flip produced by f_α, and a fault configuration v produces the outcome flip $\Delta m = A_{\mathsf{M}} v$. This is the closed case of the raw fault-effect map of Sec. 4.1. The declared *channel-check matrix* A_{D} [20] and *observable-flip matrix* A_{O}, and the *bitmatrix part* of the detector error model, are

$$A_{\mathsf{D}} := (\Delta\mathrm{DM})A_{\mathsf{M}}, \quad A_{\mathsf{O}} := (\Delta\mathrm{OM})A_{\mathsf{M}},$$
$$M_{\mathrm{DEM}} := \begin{bmatrix} A_{\mathsf{D}} \\ A_{\mathsf{O}} \end{bmatrix} \in \mathbb{F}_2^{(\mathsf{D}\sqcup\mathsf{O})\times\mathsf{F}}, \tag{15}$$

so that v flips the detectors $A_{\mathsf{D}} v$ and the observables $A_{\mathsf{O}} v$. In the *independent stochastic Pauli noise model*, each fault f_α occurs independently with probability p_α. A *detector error model* (DEM) is the bitmatrix M_{DEM} together with the probability vector $p \in [0,1]^{\mathsf{F}}$, equivalently the sparse list of mechanisms $(p_\alpha, (A_{\mathsf{D}})_{:,\alpha}, (A_{\mathsf{O}})_{:,\alpha})_{\alpha\in\mathsf{F}}$ [4, 6]. Faults with identical columns are customarily merged into one mechanism whose probability equals the probability that an odd number of those faults occur,

$$p_{\mathrm{odd}}(\mathsf{U}) = \frac{1 - \prod_{\alpha\in\mathsf{U}}(1 - 2p_\alpha)}{2}. \tag{16}$$

The outcome flip assigned to a fault is unique only up to flips of uniformly random outcome bits; since deterministic parities do not depend on such bits, A_{D}, and A_{O} for deterministic observables, are intrinsic to the experiment.

2.6 Gadgets and gadget circuits

A gadget connects a logical action with its physical realization in a way that supports composition; the definition is implicit in the logical-action verification problem of Ref. [19] and, with input and output code blocks, in Ref. [21].

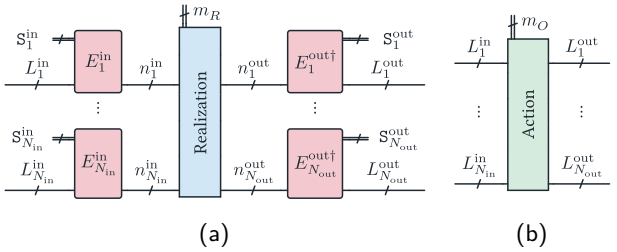


Figure 1: A gadget, shown as a realization sandwiched between encoders and unencoders in subfigure a and an equivalent logical action in subfigure b. The outcomes m_O are an affine function of m_R; this function is the outcome map.

Definition 2.8 (Gadget)**.** A *gadget* (Fig. 1) consists of two stabilizer circuits, an *action* (a logical circuit) and a *realization*, zero or more input and output stabilizer codes, each with its fixed generators and logical representatives, and an affine *outcome map* OM from the realization outcomes to the action outcomes, whose values are the gadget's logical observables. Two conditions must hold. First, the realization sandwiched between the encoders of its input codes and the unencoders of its output codes (Sec. 2.1), with the encoders' syndromes set to zero, is equivalent to the action under the outcome map (Def. 2.5). Second, in the sandwiched realization, the unencoders' syndromes are redundant given the realization outcomes and the encoders' syndromes, and are zero when encoders' syndromes are zero.

The correctness of a gadget can be checked with the algorithms of Ref. [19], as Sec. 4.2 discusses. Gadgets compose when connected output and input codes match, that is, have the same generators and logical representatives; Sec. 4.4 additionally requires matching code presentations. We call the register carried by a gadget port a *code block*.

Definition 2.9 (Gadget circuit)**.** A *gadget circuit* $\mathfrak{G}$ over a set of gadgets is a finite acyclic network in the sense of Definition 2.6 in which each box, called a *gadget call*, is labelled by a gadget g, its ports are bound to the input and output codes of g, and every wire connects an output code to a matching input code. The action and realization of its *composite gadget* G are the two stabilizer circuits obtained by wiring together the actions and the realizations, respectively, of its gadget calls. The input and output codes of G are the codes at the boundary ports of $\mathfrak{G}$, and its outcome map is

$$\mathrm{OM}_G = \bigoplus_g \mathrm{OM}_g. \tag{17}$$

Composition retains the full outcome maps, including random logical outcomes. Let $\mathrm{DM}_{\mathrm{act}(G)}$ collect selected detectors of the composite gadget's logical action. The corresponding *deterministic outcome map* is

$$\mathrm{OM}_G^{\mathrm{det}} = \mathrm{DM}_{\mathrm{act}(G)} \circ \mathrm{OM}_G. \tag{18}$$

Composition can introduce deterministic parities among individually random logical outcomes, as for the parity of the two logical readouts in a Bell experiment.

2.7 Virtual syndromes and virtual detectors

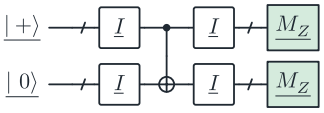


Figure 2: Physical realizations of logical Bell-pair preparation and product measurement, the running example of Secs. 3 and 4. The gadgets realize initialization in $|0\rangle$ and $|+\rangle$, syndrome extraction (a logical identity), a CNOT, and destructive logical-Z measurement. Declared classical outputs, including syndrome-extraction and transversal-measurement outcomes, are omitted.

We review the modularization of detectors introduced in Ref. [20, 21]. Consider a physical circuit partitioned into the realizations of logical operations, for instance, the Bell-pair experiment of Fig. 2. At every boundary between two operations, insert an unencoder immediately followed by an encoder that takes the unencoder's syndrome as its syndrome input. By Eq. (5), this pair implements a nondestructive measurement of the code generators whose outcome is fed forward. For the Pauli faults considered here, inserting this pair yields an equivalent circuit (Def. 2.5). We call the pair *virtual* because it is not executed. Its outcomes, the code syndrome at the boundary, are the *output virtual syndromes* of the preceding operation and the *input virtual syndromes* of the following one. Detectors of an operation sandwiched between its virtual encoders and unencoders may involve these bits; we call them *virtual detectors*. The detectors of the full circuit are recovered by substituting, for every input virtual syndrome, the expression for the matching

output virtual syndrome in terms of physical outcomes. In the scheme of Ref. [21], a virtual detector that contains an output virtual syndrome bit contains exactly one such bit, so that it expresses that bit as an affine function of the operation's outcomes and input virtual syndromes; these functions form the *output virtual syndrome map*, and substitution then proceeds in causal order. Section 3.2 shows where this per-operation scheme falls short and how the construction in Sec. 3.1 addresses its limitations.

Figure 3 illustrates the scheme. The code has the checks Z_1Z_2 and Z_2Z_3, indexed by $j \in \{1, 2\}$; round $x \in \{a, b, c, d\}$ measures them with outcomes x_j, has input virtual syndromes $s_j^{x,\mathrm{in}}$, and provides the virtual detectors $x_j + s_j^{x,\mathrm{in}}$ together with the output virtual syndrome map $s_j^x = x_j$. The CNOT realization has no physical outcomes; since it maps Z on the target block to $Z \otimes Z$ and leaves Z on the control block unchanged, its output virtual syndrome map is $s_j^c = s_j^a$ and $s_j^d = s_j^a + s_j^b$. Substituting along the wires turns the virtual detectors of rounds c and d into the detectors

$$c_j + a_j, \qquad d_j + a_j + b_j,$$

of the full circuit. Their sum $b_j + c_j + d_j$ is a detector supported on three rounds; we revisit it in Sec. 3.2.

2.8 Diagram conventions for linear and affine maps

We introduce the following conventions for maps with labelled bitvector input and output ports, closely related to [31]. Uppercase typewriter labels name classical bitvector ports, while lowercase mathematical symbols denote the values they carry; for example, m is a value at port $\mathtt{M}$ and Δm is a value at port $\Delta\mathtt{M}$. A *map diagram* is a network in the sense of Definition 2.6 whose boxes are linear or affine maps over $\mathbb{F}_2$ between labelled bitvector ports; composing the maps along the wires yields a single map $[\![\mathfrak{D}]\!]$ from the boundary inputs to the boundary outputs, and this computation is called *contracting* the diagram. Inputs are drawn as wires with incoming arrows at the top and left of a box, and outputs are drawn as wires with outgoing arrows at the bottom and right. Horizontal wires carry *interface ports*, which are connected between boxes when diagrams are composed, and vertical wires carry *free ports*, which survive contraction. The block matrix of a contracted diagram has one block per pair of boundary input and output ports, and each block is a polynomial in the blocks of the component maps, with a zero block whenever no path connects the two ports (Sec. 5).

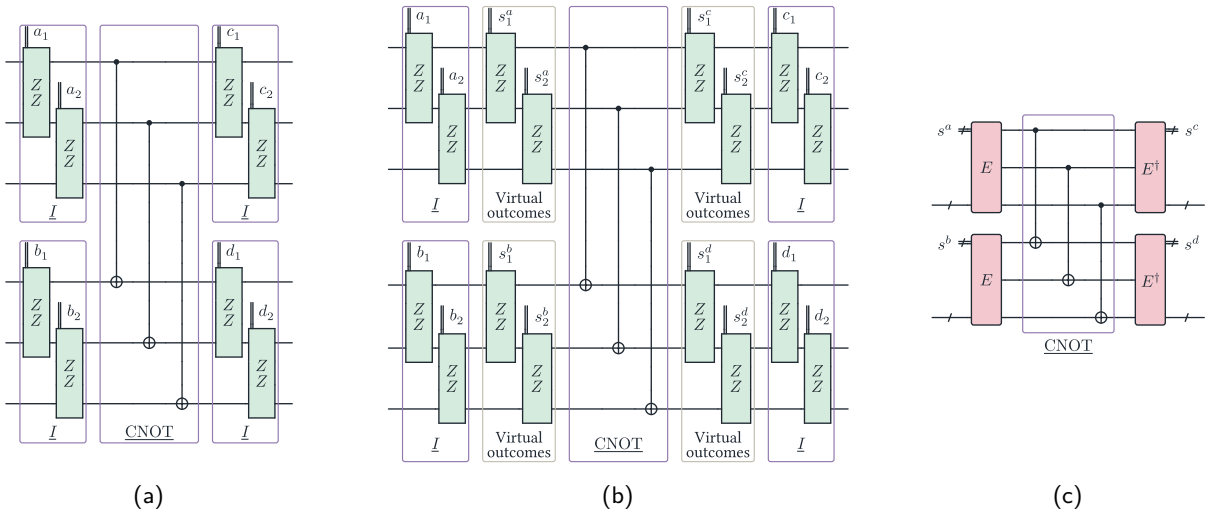
The map diagrams in this paper are obtained from gadget circuits by relabelling.

Definition 2.10 (Map diagram of a gadget circuit)**.** Let X assign to every gadget g a linear or affine map X_g with labelled bitvector ports. For each input code of g, the assignment designates a list of input ports of X_g; for each output code, it designates a list of output ports. These lists are the *interface ports* of X_g; they stand for the code blocks at the quantum input and output ports of g, and the two lists assigned to matching codes have the same length. All other ports of X_g are *free*. For a gadget circuit $\mathfrak{G}$ (Def. 2.9), the map diagram $X[\mathfrak{G}]$ is obtained by replacing every gadget call g with the box X_g and every code-block wire of $\mathfrak{G}$, which connects an output code of one call to an input code of another, with wires joining the output list of the first box to the input list of the second, entry by entry. The free ports of all boxes, together with the lists for the code blocks at the boundary of $\mathfrak{G}$, are the boundary ports of $X[\mathfrak{G}]$.

We consider three such assignments, each turning a gadget circuit into a map diagram whose contraction describes the composite gadget.

- *Detector maps* DM_g (Sec. 3.3, Def. 3.6): the list for a code is its virtual-syndrome port; measurement outcomes and detector values are free. The detector map of the composite gadget is defined as the contraction $\mathrm{DM}_G := [\![\mathrm{DM}[\mathfrak{G}]]\!]$, which substitutes output virtual syndromes into input virtual syndromes as in Sec. 2.7 (Fig. 6).

- *Raw fault-effect maps* RFE_g (Sec. 4.1): the list for a code is its boundary-error port; fault configurations and outcome flips are free. The raw fault-effect map of the composite gadget is defined as the contraction $\mathrm{RFE}_G := [\![\mathrm{RFE}[\mathfrak{G}]]\!]$, Eq. (22), after checking that the contraction is a valid raw fault-effect map (Fig. 9).



(a) (b) (c)

Figure 3: A transversal CNOT between two blocks of the three-qubit repetition code, preceded by the syndrome-extraction rounds a (control block) and b (target block) and followed by the rounds c and d; each round implements the logical identity. Subfigure a shows the physical circuit partitioned into logical operations, subfigure b adds virtual syndromes at the boundaries, and subfigure c shows the transversal CNOT sandwiched between virtual unencoders and encoders, with input virtual syndromes s^a, s^b and output virtual syndromes s^c, s^d.

- *Extended detector error models* EDEM_g (Sec. 4.3, Def. 4.4): the list for a code consists of its virtual-syndrome-flip port and its boundary-error port, or the boundary-error coordinates selected by the code presentation; fault configurations, detector flips, and observable flips are free. Here the contraction is not a definition but a result: $\mathrm{EDEM}_G = [\![\mathrm{EDEM}[\mathfrak{G}]]\!]$ is Theorem 4.6 in Sec. 4.4, and Sec. 5 evaluates such contractions symbolically.

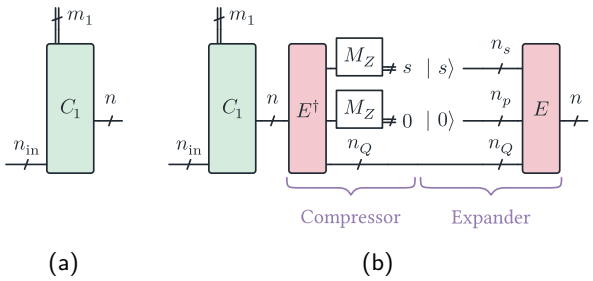
3 Decomposing detectors

This section develops tools for decomposing detectors across subcircuits of stabilizer circuits. We first establish formal detector-decomposition results, then relate them to the virtual detectors of Sec. 2.7. We conclude by extending the detector map of Sec. 2.3, which represents a list of detectors as an affine map, to gadgets. For gadgets with open code boundaries, these maps carry virtual-syndrome inputs and outputs; contracting the detector-map diagram of a gadget circuit yields the detector map of its composite gadget.

3.1 Formal foundations

To decompose the detectors of a serial composition of two stabilizer circuits C_1 and C_2 at the boundary between them, we impose three requirements on the boundary. (i) The connecting qubits carry no stabilizer of the output of C_1, including any with a deterministic sign. (ii) The outcome-dependent signs of the stabilizers removed from the connecting qubits are instead recorded as classical bits s, which are affine functions of the outcomes m_1 of C_1. (iii) Only independent such bits are recorded, so that no detector is supported on s. An unencoder of the code at the boundary (Sec. 2.1) satisfies these requirements except in the two situations discussed in Sec. 3.2. Definition 3.1 formalizes (i)–(iii) by introducing a *compressor* and pairing it with an *expander* that reverses its action.

Definition 3.1. Given a stabilizer circuit C_1 with n output qubits and outcome vector m_1 (Fig. 4a), a C_1-*compressor* is a stabilizer circuit that applies a Clifford unitary $E^\dagger$ to the n qubits and then measures $n_p + n_s$ of them destructively in the Z basis, obtaining n_p deterministic outcomes 0 and an outcome bitvector $s \in \mathbb{F}_2^{n_s}$ (Fig. 4b), such that the circuit consisting of C_1 followed by the compressor has a trivial output stabilizer group (Sec. 2.2) on its $n_Q = n - n_p - n_s$ remaining qubits, the compressor outcome is an affine function $s = \mathrm{CM}(m_1)$ of m_1, and no nontrivial detector is supported on s. The C_1-*expander* is the compressor run in reverse: it takes s as a declared classical input, prepares n_p fresh qubits



(a) (b)

Figure 4: (a) A stabilizer circuit C_1 with n_{in} input qubits, n output qubits, and outcome vector m_1. (b) An equivalent stabilizer circuit consisting of C_1 followed by a compressor and expander (Def. 3.1, Lemma 3.2). The compressor applies the Clifford unitary $E^\dagger$ and measures $n_p + n_s$ qubits in the Z basis, obtaining the deterministic outcomes 0 and the outcomes s; the expander prepares $n_p + n_s$ fresh qubits in $|\mathbf{0}\rangle$ and $|s\rangle$ and applies E.

in $|\mathbf{0}\rangle$ and n_s fresh qubits in $|s\rangle$ using X gates controlled by the bits of s, and applies E to these qubits and the n_Q remaining qubits (Fig. 4b).

A compressor minimizes the classical and quantum information present at the circuit output without losing information. Informally, the definition ensures that n_Q is minimal and, subject to that condition, that n_s is also minimal. If the output stabilizer group on the n_Q qubits were nontrivial, then we could decrease n_Q by one and increase either n_s or n_p by one, depending on whether the sign of the additional stabilizer depends on m_1. If there were a nontrivial detector supported on s, then we could decrease n_s by one and increase n_p by one.

Lemma 3.2. *Every stabilizer circuit C_1 has a C_1-compressor (Def. 3.1), and C_1 followed by a C_1-compressor and its expander is equivalent to C_1 (Def. 2.5) under the outcome map that retains m_1 and discards the compressor outcomes.*

Proof. The construction separates stabilizers with fixed signs from those whose signs must be recorded. Let G be the output stabilizer group of C_1 on its n output qubits (Sec. 2.2), and let G_0 be its subgroup of stabilizers whose signs are constant across all attainable outcomes m_1. Choose n_p independent generators of G_0 and Paulis $P_1, \ldots, P_{n_s}$ whose cosets form a basis of G/G_0. Take a Clifford unitary E such that $E^\dagger$ maps the chosen generators of G_0, taken with their fixed signs, to $Z_1, \ldots, Z_{n_p}$ and maps P_i to Z_{n_p+i} for $i = 1, \ldots, n_s$. Measuring these $n_p + n_s$ qubits in the Z basis yields the deterministic outcomes 0 and the outcomes s, whose bits are the signs of the P_i and hence affine functions of m_1.

On each outcome branch, the measured qubits are already in the state $|\mathbf{0}\rangle \otimes |\mathrm{CM}(m_1)\rangle$ after applying $E^\dagger$ and are therefore uncorrelated with the remaining qubits. Destructively measuring them and then preparing the same state leaves the branch unchanged. Applying E restores the original output, proving the claimed equivalence after discarding the redundant compressor outcomes.

It remains to check that neither the quantum output nor the classical record contains a redundant degree of freedom. If a Pauli P_Q stabilized the remaining n_Q qubits with a sign affine in (m_1, s), substituting $s = \mathrm{CM}(m_1)$ would make that sign affine in m_1. Thus $E(I \otimes P_Q)E^\dagger$ would belong to G. But $E^\dagger$ maps every element of G to a product of $Z_1, \ldots, Z_{n_p+n_s}$, which acts trivially on the remaining qubits, so $P_Q = I$. Likewise, a nontrivial detector $\langle w, s\rangle + c$ supported on s would give the product $\prod_i P_i^{w_i}$ a fixed sign. This product would then belong to G_0, contradicting the independence of the chosen cosets. Hence all three compressor properties hold. □

Compressors and expanders are not unique, but Lemma 3.2 guarantees their existence. We now use them to cut detectors across the serial composition of C_1 and C_2 (Fig. 5).

Detectors are affine expressions in the outcomes that vanish on every attainable outcome (Def. 2.4); for the expander followed by C_2, the free classical input s counts as a variable. One direction of the decomposition follows directly from the composition of boundary constraints (Sec. 2.2). A detector of C_1 is a detector of the composed circuit, and a detector $\langle v_0, s\rangle + \langle v_2, m_2\rangle + b$ of the expander followed by C_2 becomes the detector $\langle v_0, \mathrm{CM}(m_1)\rangle + \langle v_2, m_2\rangle + b$ of the composed circuit, because the compressor records $s = \mathrm{CM}(m_1)$ and inserting it together with the expander is an equivalence (Lemma 3.2).

We now prove the converse, stated in Lemma 3.4 below. The proof relies on Theorem 3.3, which factors detectors across a cut without using a compressor and requires only that C_2 be a stabilizer circuit. In terms of channels and observables, the theorem states that an observable on the outcome registers that stabilizes the output of a serial composition factors through a Pauli P at the cut, which the first channel prepares and the second measures. The Pauli P is

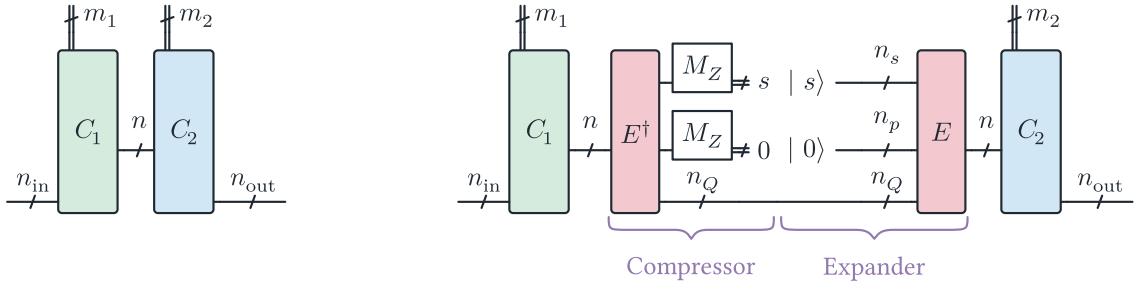


Figure 5: (a) A two-part open stabilizer circuit with n_{in} input qubits and n_{out} output qubits, composed of stabilizer circuits C_1 and C_2 connected by n qubits and having outcome vectors m_1 and m_2. (b) An equivalent circuit with a C_1-compressor and C_1-expander inserted between the two parts.

obtained by pulling the observable back through the second channel using its adjoint (Sec. 2.2).

Theorem 3.3 (Stabilized observables factor across a serial composition)**.** *Consider a channel Φ_1 from a register A to registers $\mathsf{M}_1 \otimes B$, a stabilizer channel Φ_2 from B to $\mathsf{M}_2 \otimes C$, and their serial composition $\Phi := (\mathrm{id}_{\mathsf{M}_1} \otimes \Phi_2) \circ \Phi_1$ from A to $\mathsf{M}_1 \otimes \mathsf{M}_2 \otimes C$. Let Z_{M_1} and Z_{M_2} be signed Z-type observables on M_1 and M_2 such that $Z_{\mathsf{M}_1} \otimes Z_{\mathsf{M}_2} \otimes I_C$ stabilizes the output of Φ, that is, $(Z_{\mathsf{M}_1} \otimes Z_{\mathsf{M}_2} \otimes I_C)\Phi(\rho) = \Phi(\rho)$ for all operators ρ. Then $P := \Phi_2^\dagger(Z_{\mathsf{M}_2} \otimes I_C)$ is a Pauli observable on B such that, for all operators ρ,*

$$(Z_{\mathsf{M}_1} \otimes P)\Phi_1(\rho) = \Phi_1(\rho),$$
$$(Z_{\mathsf{M}_2} \otimes I_C)\Phi_2(P\rho) = \Phi_2(\rho).$$

Proof. Pulling the stabilized observable back to the input gives the identity. Using Corollary B.2 (Appendix B) and the composition rule for adjoints, we obtain

$$\begin{aligned} I_A &= \Phi^\dagger(Z_{\mathsf{M}_1} \otimes Z_{\mathsf{M}_2} \otimes I_C) \\ &= \Phi_1^\dagger(Z_{\mathsf{M}_1} \otimes \Phi_2^\dagger(Z_{\mathsf{M}_2} \otimes I_C)) \\ &= \Phi_1^\dagger(Z_{\mathsf{M}_1} \otimes P). \end{aligned}$$

The stabilizer assumption on Φ_2 enters here: by the dual statement of Lemma 2.2, P is either zero or a Pauli up to a sign. The displayed identity excludes zero, and preservation of Hermiticity makes P a Hermitian Pauli, hence a unitary. We can therefore apply Corollary B.2 to both $\Phi_1^\dagger(Z_{\mathsf{M}_1} \otimes P) = I_A$ and $\Phi_2^\dagger(Z_{\mathsf{M}_2} \otimes I_C) = P$. These yield the first and second channel constraints in the statement, respectively. □

In the language of Sec. 2, let M_j be the declared classical outcome register of Φ_j and $Z_{\mathsf{M}_j} = Z_{\mathsf{M}_j}(a_j, b_j)$. The hypothesis of the theorem then says that $\langle a_1, m_1\rangle + \langle a_2, m_2\rangle + b_1 + b_2$ is a detector of Φ (Def. 2.4), and its conclusion says that (I_A, P, a_1, b_1) and (P, I_C, a_2, b_2) are boundary constraints (Def. 2.3): P stabilizes the output of Φ_1 with the sign given by Z_{M_1}, and Φ_2 measures P with the sign given by Z_{M_2}. A compressor records the signs of exactly these boundary observables in its outcomes s, which yields the detector decomposition. Declared classical boundary inputs of the composed circuit are counted among the outcomes m_1 of C_1. By Def. 2.4, a detector must vanish for every value of such an input, so replacing the input by a fair coin (Sec. 2.4) whose value is recorded in m_1 changes neither the detectors nor the argument below.

Lemma 3.4 (Cutting detectors)**.** *Consider the serial composition of two stabilizer circuits C_1 and C_2 with outcome vectors m_1 and m_2 (Fig. 5a), and fix a C_1-compressor and its expander (Def. 3.1). Every detector of the composed circuit is the sum of a detector d_1 of C_1 and a detector d_2 of the expander followed by C_2 (Fig. 5b), with the compressor outcome $\mathrm{CM}(m_1)$ substituted for the classical input s of the expander. If the detector is nontrivial, at least one of d_1 and d_2 is nontrivial. The decomposition is unique.*

Proof. The key point is that, after compression, the boundary Pauli supplied by Theorem 3.3 must be a parity of the classical bits s. Let $d = \langle a_1, m_1\rangle + \langle a_2, m_2\rangle + b$ be a detector, and insert the compressor and expander after C_1. This preserves d by Lemma 3.2. Let Φ_1 be C_1 followed by the compressor and Φ_2 the expander followed by C_2. At the cut, regard s as the classical register S passed to Φ_2 together with the remaining

qubits Q, while m_1 is retained in the outcome register M_1. Apply Theorem 3.3 with $Z_{\mathsf{M}_1}(a_1, b)$ and $Z_{\mathsf{M}_2}(a_2, 0)$. It gives a Pauli

$$P := \Phi_2^\dagger(Z_{\mathsf{M}_2}(a_2, 0) \otimes I_C)$$

that Φ_1 prepares with sign $(-1)^{\langle a_1, m_1\rangle + b}$ and Φ_2 measures with outcome $\langle a_2, m_2\rangle$.

Because S is a declared classical input of Φ_2, the image of $\Phi_2^\dagger$ is invariant under dephasing on S (Eq. (9)). Thus P has only I and Z factors on S, so write $P = (-1)^c P_Q \otimes Z_{\mathsf{S}}^w$ with P_Q Hermitian. On a branch with outcomes (m_1, s), the preparation constraint then says that P_Q stabilizes Q with sign $(-1)^{\langle a_1, m_1\rangle + b + \langle w, s\rangle + c}$. The compressor leaves Q with a trivial output stabilizer group, so $P_Q = I$, absorbing any overall sign into c. Consequently, the boundary observable is just $P = (-1)^c I_Q \otimes Z_{\mathsf{S}}^w$.

The preparation and measurement constraints now give the two detectors

$$d_1 := \langle a_1, m_1\rangle + \langle w, \mathrm{CM}(m_1)\rangle + b + c,$$
$$d_2 := \langle w, s\rangle + \langle a_2, m_2\rangle + c.$$

Here d_1 is affine in m_1 because CM is affine, and d_2 vanishes for every classical input s and every quantum input of Φ_2 by its measurement constraint. Substituting $s = \mathrm{CM}(m_1)$ cancels the shared boundary parity and gives $d_1 + d_2 = d$. If d is nontrivial, the two summands cannot both be zero, so at least one is nontrivial.

For uniqueness, the essential fact is that s is a free input of the expander. If $d = d_1 + d_2 = d_1' + d_2'$ are two decompositions, then $d_2 - d_2'$ has no m_2 part, so it is a detector of the expander followed by C_2 that depends on s alone. Such a detector vanishes for every value of s (Def. 2.4) and is therefore zero. Hence $d_2 = d_2'$ and $d_1 = d_1'$. □

The proof is constructive. Propagating $Z_{\mathsf{M}_2}(a_2, 0) \otimes I_C$ backwards through C_2 and the expander by a stabilizer-tableau computation gives P. Reading off its classical parity w and sign c then determines d_1 and d_2. Thus a detector basis chosen for the composed circuit can be decomposed detector by detector while preserving the basis supplied to the decoder. This matters because decoding performance depends on the choice of basis. The first two compressor properties make the decomposition possible: no outcome-dependent stabilizer remains on Q, and the recorded boundary parity is affine in m_1. The third property makes the classical interface minimal: since its attainable values form an affine code (Prop. 2.7), the absence of detectors on s means that s ranges over all of $\mathbb{F}_2^{n_s}$.

Corollary 3.5 (Completeness)**.** *Let B_1 be a complete detector basis of C_1 and B_2 a complete detector basis of the expander followed by C_2, and let B_2' be obtained from B_2 by substituting* $\mathrm{CM}(m_1)$ *for s. Then $B_1 \cup B_2'$ is a complete detector basis of the composed circuit.*

Proof. Each element of $B_1 \cup B_2'$ is a detector of the composed circuit by the composition of boundary constraints noted before Lemma 3.4. That lemma also establishes that the union spans the detector space: decompose any detector as $d_1 + d_2$, expand the two parts in B_1 and B_2, and substitute $s = \mathrm{CM}(m_1)$.

For independence, any linear relation among the elements of $B_1 \cup B_2'$ gives a decomposition of the zero detector into a sum from B_1 and a substituted sum from B_2. Uniqueness in Lemma 3.4, compared with the decomposition $0 + 0$, forces both sums to be zero before substitution. Independence of B_1 and B_2 then forces every coefficient in the relation to vanish. □

3.2 Virtual syndromes and virtual detectors

In the approach to modularizing detectors introduced in Ref. [21] and reviewed in Sec. 2.7, an unencoder does not always correspond to a compressor. This correspondence can fail in two common ways. First, in violation of requirement (iii) of Sec. 3.1, nontrivial detectors may be supported entirely on the output virtual syndrome bits. For example, in logical-zero state preparation, the output syndrome is commonly fixed to zero, so each output virtual syndrome bit s_j defines a detector $s_j = 0$. We handle this case by setting the expression for s_j to zero in the output virtual syndrome map. Second, in violation of requirement (i), the output stabilizer group (Sec. 2.2) may contain a logical operator of the code. A logical-zero preparation circuit again provides an example. This case is addressed by detectors in the logical circuit. For example, a logical circuit for a Z-memory experiment prepares the logical-zero state and measures Z. This circuit has a detector requiring the Z-measurement outcome to be zero. Whenever a logical operation produces logical measurement outcomes, its outcome

map (Def. 2.8, Eq. (12)) expresses them as affine functions of the physical outcomes. Substituting the outcome map into a detector of the logical circuit turns it into a detector of the physical circuit.

Another challenge for this virtual-detector approach is multiport composition. In this setting, not every choice of global detectors admits a simple decomposition. Consider the transversal CNOT flanked by syndrome-extraction rounds in Fig. 3. Section 2.7 displays its detector $b_j+c_j+d_j$, which is supported on three syndrome-extraction rounds. Applying Lemma 3.4 to the boundary following the CNOT gives $d_2 = s_j^c + s_j^d + c_j + d_j$ for the part after the cut. The parity $b_j + s_j^c + s_j^d$ recorded before the cut vanishes identically after substituting $s_j^c = a_j$ and $s_j^d = a_j + b_j$, so the detector d_1 of the part before the cut is trivial. The detector $s_j^c+s_j^d+c_j+d_j$ is split across two parallel gadgets and would not appear if we resolved output virtual syndromes in terms of past outcomes within individual gadgets.

Some quantum error-correcting codes have a natural overcomplete set of code generators, as in various versions of d-dimensional toric codes. The unencoder–encoder picture requires choosing a linearly independent set of generators. However, we can still define the syndrome using an overcomplete set of generators and specify virtual detectors in terms of the resulting overcomplete virtual syndromes, as already highlighted in Ref. [21]. The syndromes of the extra generators can be regarded as convenient intermediate variables.

3.3 Composing the detector basis using affine maps

To compose detectors across gadgets, we use the detector map of Eq. (12), which represents a list of detectors as an affine map from a bitvector of measurement outcomes to a bitvector of detector values, and extend it with virtual-syndrome ports.

A preliminary step in constructing an extended detector error model (EDEM, Sec. 4.3) is to introduce the *detector map* (DM) of a gadget.

Definition 3.6. The *detector map* DM of a gadget extends the detector map of Eq. (12) with virtual-syndrome ports; it is an affine map with the following bitvector input and output ports. The inputs are the gadget's measurement outcomes (`M`) and the input virtual syndrome bitvectors (`V`) for each input code block of the gadget. The outputs are the detector values (`D`) and the output virtual syndrome bitvectors (`V`) for each output code block of the gadget. Its linear part ΔDM maps the corresponding measurement and virtual-syndrome flips to detector and output virtual-syndrome flips. For a gadget without code boundaries, the virtual-syndrome ports are empty and the map reduces to the detector map of Eq. (12) from measurement outcomes to detector values.

Figures 6a and 6b show examples of detector maps for state preparation, syndrome extraction, and a transversal CNOT. For the detector map, the interface ports are the input and output virtual-syndrome ports, while measurement outcomes are free inputs and detector values are free outputs. For a gadget circuit $\mathfrak{G}$, the detector map of its composite gadget is obtained by contracting the detector-map diagram DM[$\mathfrak{G}$] (Def. 2.10), in which the individual gadget detector maps are wired along the code blocks of $\mathfrak{G}$ (Fig. 6c). The contraction substitutes output virtual syndromes for input virtual syndromes and thereby turns every virtual detector into a detector of the composite gadget, as in Sec. 2.7. We write $\mathrm{DM}_G = [\![\mathrm{DM}[\mathfrak{G}]]\!]$. After contraction, only the free ports and the virtual-syndrome ports at the boundary of $\mathfrak{G}$ remain (Fig. 6d). In the final detector map, detectors are naturally partitioned by the subcircuit in which they can be fully computed, and measurement outcomes are partitioned by the subcircuit to which they belong. When the partitions are indexed in temporal order, the matrix corresponding to the affine detector map is block lower triangular.

Success bits. When considering fault-tolerant protocols with postselection, it is convenient to introduce a special kind of detector called a *success bit*. A success bit is a declared deterministic parity whose violation causes the run to be discarded rather than decoded. Success bits can be added to detector maps as free inputs and outputs to support postselection. They are carried through the contraction as free ports rather than substituted along code blocks.

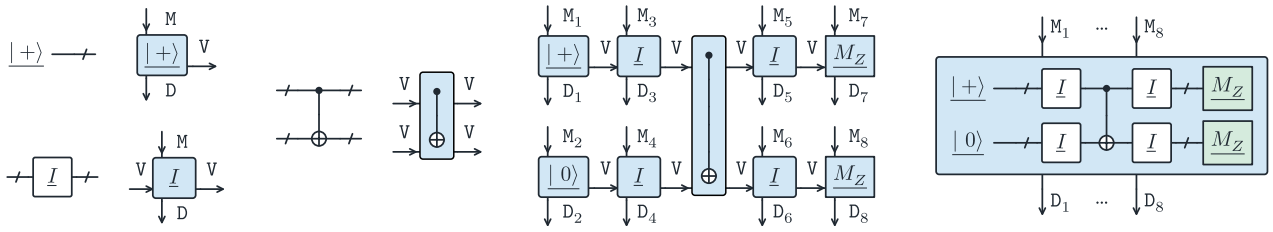


Figure 6: Detector maps with virtual-syndrome interfaces. Subfigures a and b show gadgets and their detector maps DM_g. Subfigure c shows the detector-map diagram $\mathrm{DM}[\mathfrak{G}]$ of the Bell experiment in Fig. 2, and subfigure d shows its contraction, the detector map DM_G of the composite gadget.

4 Extended detector error model

In this section we generalize detector error models to extended detector error models for gadgets with input and output qubits. Section 4.1 introduces boundary errors and the raw fault-effect map, which propagates faults and boundary errors through a gadget; Sec. 4.2 fixes the code presentations and gadget detector contracts that play the role of the detector basis; and Sec. 4.3 defines the extended detector error model. We show that the extended detector error model of the composite gadget of a gadget circuit is the contraction of the EDEM diagram of that circuit, in which the per-gadget extended detector error models are wired along the code blocks (Sec. 4.4). We also discuss optimizations that reduce the size of extended detector error models (Secs. 4.3 and 4.5).

4.1 Boundary errors and raw fault-effect maps

Faults occur at different locations within a circuit, but a detector error model retains only their action on the circuit boundary. To obtain this boundary description, pull the classical bit controlling each allowed fault f_α (Sec. 2.4) back to an input port labelled by $\alpha \in \mathsf{F}$, so that the faulty circuit becomes a single stabilizer channel

$$\begin{aligned} \Psi &: A \otimes \mathsf{F} \longrightarrow B \otimes \mathsf{M}, \\ \Phi_v(\rho) &= \Psi\big(\rho \otimes X_{\mathsf{F}}^v |\mathbf{0}\rangle\langle\mathbf{0}|_{\mathsf{F}} X_{\mathsf{F}}^v\big), \end{aligned} \tag{19}$$

from the quantum input register A and the fault ports to the quantum output register B and the outcome register M. A fault configuration v is then represented by the Pauli X_{F}^v on the fault ports.

Let E_{in} and E_{out} be the label spaces $\mathbb{F}_2^{2|A|}$ and $\mathbb{F}_2^{2|B|}$ of the phase-free Pauli groups on A and B (Sec. 2.1), with Pauli representatives $P_A(e_{\mathrm{in}})$ and $P_B(e_{\mathrm{out}})$. We call their elements *boundary errors*; an output boundary error is interpreted relative to the ideal output on the same outcome branch.

Definition 4.1 (Raw fault-effect map)**.** A *raw fault-effect map* compatible with Ψ is a linear map

$$\mathrm{RFE}_\Psi : E_{\mathrm{in}} \oplus \mathbb{F}_2^{\mathsf{F}} \longrightarrow \mathbb{F}_2^{\mathsf{M}} \oplus E_{\mathrm{out}} \tag{20}$$

such that, whenever $\mathrm{RFE}_\Psi(e_{\mathrm{in}}, v) = (\Delta m, e_{\mathrm{out}})$,

$$\Psi \circ \mathrm{Ad}_{P_A(e_{\mathrm{in}}) \otimes X_{\mathsf{F}}^v} = \mathrm{Ad}_{P_B(e_{\mathrm{out}}) \otimes X_{\mathsf{M}}^{\Delta m}} \circ \Psi. \tag{21}$$

We call Δm the *outcome flip* and e_{out} the *output boundary error* produced by (e_{in}, v).

Such a map exists by Lemma 2.2: for each basis element of $E_{\mathrm{in}} \oplus \mathbb{F}_2^{\mathsf{F}}$, the lemma supplies an output Pauli satisfying Eq. (21) up to a factor Z_{M}^z. This factor acts trivially on the declared classical output M by Eq. (10). Multiplying the chosen output Paulis extends the assignment linearly. The *raw fault-effect map of a gadget* g is a raw fault-effect map RFE_g of its realization with the gadget's fault set $\mathcal{F}_g$, where E_{in} and E_{out} are the spaces of phase-free Pauli labels on its input and output code blocks. For a gadget without code blocks, this map is the matrix A_{M} of Sec. 2.5.

Remark 4.2 (Choice of output representative). A raw fault-effect map is not unique. The output representative $P_B(e_{\mathrm{out}}) \otimes X_{\mathsf{M}}^{\Delta m}$ may be changed by an output stabilizer (Sec. 2.2) or by a flip of an independently random outcome bit, since either acts trivially on the image of Ψ. These changes preserve the circuit semantics and the flips of all deterministic detector and observable parities. Dually, two input boundary errors that differ by a stabilizer of the code on that wire have

the same downstream effect on codespace states, by Lemma 2.1. We call this freedom the *choice of output representative*; every statement below holds for any fixed choice.

Fault propagation through a gadget circuit is described by a composition of linear maps. For raw fault-effect maps, the interface ports of a code are its boundary-error ports, and the fault and outcome-flip ports are free (Def. 2.10). Applying Eq. (21) successively along the wires of a gadget circuit $\mathfrak{G}$ yields the same compatibility relation for its composite gadget G, with fault set $\bigsqcup_g \mathcal{F}_g$. Thus, the contraction of the RFE diagram RFE[$\mathfrak{G}$] is a raw fault-effect map of G. We define the raw fault-effect map of the composite gadget by this contraction,

$$\mathrm{RFE}_G := [\![\mathrm{RFE}[\mathfrak{G}]]\!], \tag{22}$$

as illustrated in Fig. 9 of Sec. 4.4 for the Bell-pair experiment of Fig. 2.

4.2 Gadget detector contracts and code presentations

The standard detector error model is defined with respect to a choice of detector basis, observables, and elementary faults. The same three choices must be made to define the extended detector error model of a gadget, and gadget detector contracts and code presentations generalize the choice of detector basis.

Definition 4.3 (Code presentation and gadget detector contract)**.** A *code presentation* of a stabilizer code is a possibly over-complete list of stabilizer generators together with the syndrome and logical-effect coordinates used at the code boundary: for a boundary error e with representative P, the *syndrome coordinate* $s(e)$ is the commutation vector of P with the listed generators, and the *logical-effect coordinate* $l(e)$ is the phase-free label of the logical Pauli ℓ_P of P (Lemma 2.1). A *gadget detector contract*, or *gadget contract* for short, fixes the virtual detectors of the gadget, the affine expressions of its output virtual syndromes in terms of its measurement outcomes and input virtual syndromes, and the code presentations of its input and output codes; that is, it fixes a detector map (Def. 3.6) together with the code presentations.

Both coordinates depend only on the coset of P modulo the stabilizer group, by Lemma 2.1, so they are well defined on boundary errors up to stabilizers.

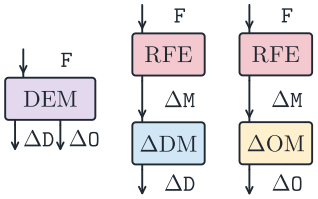


Figure 7: Bitmatrix part of the standard detector error model and its factorization through the raw fault-effect map (RFE) and the linear parts ΔDM and ΔOM of the detector and outcome maps.

For composition to work correctly, it is important that not only the input and output stabilizer codes of connected ports match, but also that their code presentations match.

Contract correctness. A gadget contract is *correct* when every declared virtual detector and every declared output virtual syndrome relation is a deterministic parity of the realization sandwiched between the encoders and unencoders of its codes (Fig. 1a), that is, a check of the outcome code of the sandwiched circuit (Prop. 2.7), and when the declared detectors, output virtual syndrome relations, success bits (Sec. 3.3) and deterministic logical outcomes together generate all deterministic parities of that circuit (rank condition). Deterministic observables are the deterministic parities of the action outcomes, taken modulo detectors. Both conditions are checked by Clifford simulation [19], as Sec. 6.2 describes.

4.3 Extended detector error model

Recall that the bitmatrix part of the standard detector error model, Eq. (15) in Sec. 2.5, can be viewed as a linear map from fault configurations to detector and observable flips (Fig. 7) for a gadget that has no inputs and no outputs. It is obtained by composing the gadget's raw fault-effect map, which in this closed case is the matrix A_{M}, with ΔDM and ΔOM, as shown in Fig. 7. EDEM composition propagates the full outcome map. Deterministic observable-flip rows are subsequently extracted using the linear part of the composite logical action's detector map, as in Eq. (18). An action outcome that is not deterministic, such as the random logical readout of Sec. 6.3, is nevertheless a legitimate observable,

whose row $\Delta\mathrm{OM}\ A_{\mathsf{M}}$ depends on the choice of output representative (Remark 4.2). Only the rows of deterministic parities are independent of that choice.

The above viewpoint leads to a natural generalization to gadgets with inputs and outputs. When composing gadgets in the diagrammatic representations such as Fig. 8, we add horizontal input and output wires to pass virtual syndromes and boundary errors. The virtual syndrome bits carry the signs of the stabilizers as expected by the providing gadget on the basis of its available classical information. Boundary-error bits represent the residual Pauli effect of earlier faults on the output boundary, in the chosen output representative (Remark 4.2).

Definition 4.4. A raw extended detector error model for a gadget (with associated gadget contract and elementary faults) is a linear map with the following bitvector input and output ports (Fig. 8). The inputs are a fault configuration ($\mathtt{F}$) and, for each gadget input code block, input virtual syndrome flips ($\Delta\mathtt{V}$) and an input boundary error ($\mathtt{E}$). The outputs are detector flips ($\Delta\mathtt{D}$), declared observable flips ($\Delta\mathtt{O}$), and, for each gadget output code block, output virtual syndrome flips ($\Delta\mathtt{V}$) and an output boundary error ($\mathtt{E}$).

Two structural properties are visible in Fig. 8: the output boundary error, and hence its coordinates introduced below, depends only on the fault configuration and the input boundary error, because the raw fault-effect map has no virtual-syndrome input; and the input virtual syndrome flips enter only the detector flips and the output virtual syndrome flips, through $\Delta\mathrm{DM}$.

The dimensions of the input and output wires of the raw extended detector error model can be further reduced by replacing boundary errors with coordinates selected by a code presentation.

Definition 4.5 (Code-presentation map)**.** The *code-presentation map* $\mathrm{CPM} : \mathtt{E} \to (\mathtt{S}, \mathtt{L})$ of a code presentation is the linear map sending a boundary error e to its syndrome and logical-effect coordinates $(s(e), l(e))$ of Definition 4.3. A *section* of CPM is a linear map $\sigma : \mathrm{im}(\mathrm{CPM}) \to \mathtt{E}$ with $\mathrm{CPM} \circ \sigma = \mathrm{id}$, which maps attainable coordinate pairs back to representative boundary errors.

The code-presentation map is obtained by pulling physical X- and Z-Pauli errors back through the encoder (Lemma 2.1) and mapping the resulting syndrome to the possibly overcomplete generator list of the code presentation. A convenient section sends the syndrome of each generator in an independent subset to a destabilizer of that generator, a Pauli anticommuting with it and commuting with the other generators, and each logical coordinate to the fixed logical representative of the code. The *extended detector error model* is the raw extended detector error model with its boundary-error ports compressed to boundary-error coordinates,

$$\mathrm{EDEM} := (\mathrm{id} \oplus \mathrm{CPM}_{\mathrm{out}}) \circ \mathrm{EDEM}^{\mathrm{raw}} \circ (\mathrm{id} \oplus \sigma_{\mathrm{in}}), \tag{23}$$

where $\mathrm{CPM}_{\mathrm{out}}$ and σ_{in} act on the boundary-error ports of the output and input codes and id on the remaining ports. Inserting $\sigma \circ \mathrm{CPM}$ on a boundary-error wire leaves the contraction of a diagram unchanged up to the choice of output representative, because e and $\sigma(\mathrm{CPM}(e))$ have the same coordinates and therefore differ by a stabilizer of the code on that wire (Remark 4.2). The raw form is simpler and more convenient for illustrating the key ideas; the implementation of Sec. 6 uses the compressed form.

For a gadget call g, write $b = (s, l)$ for the boundary coordinates on a wire and Δv for the virtual-syndrome flips, with subscripts in and out collecting all input and output wires of g. In block form, its EDEM is

$$\begin{aligned} \Delta d_g &= D_F^g f_g + D_B^g b_{\mathrm{in}} + D_V^g \Delta v_{\mathrm{in}}, \\ \Delta o_g &= O_F^g f_g + O_B^g b_{\mathrm{in}}, \\ \Delta v_{\mathrm{out}} &= V_F^g f_g + V_B^g b_{\mathrm{in}} + V_V^g \Delta v_{\mathrm{in}}, \\ b_{\mathrm{out}} &= B_F^g f_g + B_B^g b_{\mathrm{in}}. \end{aligned} \tag{24}$$

The virtual-syndrome interface Δv records fault-induced changes in the stabilizer signs inferred from classical data, while the error interface $b = (s(e), l(e))$ records the syndrome and logical effect of the residual Pauli error e. In particular, Δv need not equal $s(e)$: a flipped syndrome-measurement bit can change Δv without changing e, and a Pauli fault after the last check can change e without changing Δv. The absent blocks express the structural properties above: observable flips and outgoing boundary coordinates do not depend on incoming virtual-syndrome flips.

Tables 2 and 1 summarize the linear maps and named port types used in this section. As defined in Eq. (13), a label $\Delta\mathtt{X}$ denotes the fault-induced

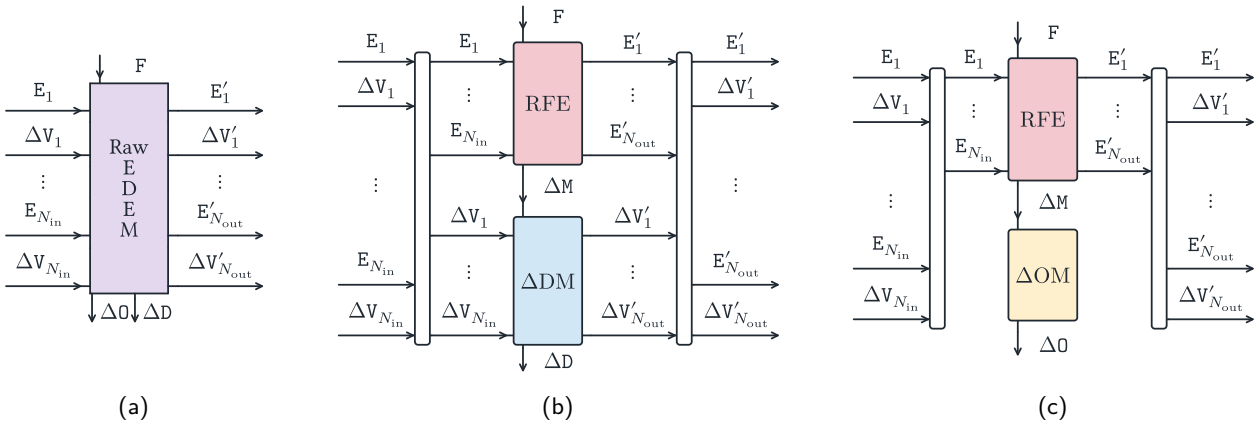


Figure 8: Raw extended detector error model (Def. 4.4). Subfigure 8a shows a diagram with labelled inputs and outputs, subfigure 8b relates the detector-flip output to the raw fault-effect map (RFE, Def. 4.1) and the linear part ΔDM of the detector map (Def. 3.6), and subfigure 8c relates the observable-flip output to the raw fault-effect map and the linear part ΔOM of the outcome map (Def. 2.8). Thin white blocks reorder wires.

change in the corresponding classical quantity $\mathtt{X}$.

4.4 Composition

Let $\mathfrak{G}$ be a gadget circuit (Def. 2.9) in which connected ports carry matching codes and code presentations, and let G be its composite gadget. The gadget circuit induces a gadget contract, a fault set, and a raw fault-effect map for G: the elementary faults are the disjoint union $\bigsqcup_g \mathcal{F}_g$ over the gadget calls, the detector map is the contraction $\mathrm{DM}_G = [\![\mathrm{DM}[\mathfrak{G}]]\!]$ (Sec. 3), the raw fault-effect map is the contraction $\mathrm{RFE}_G = [\![\mathrm{RFE}[\mathfrak{G}]]\!]$ of Eq. (22), and the code presentations at the boundary of $\mathfrak{G}$ are inherited from the corresponding gadget calls. We call this the *induced gadget contract* of $\mathfrak{G}$; all statements below refer to it. For the extended detector error model of a gadget, the interface ports for a code are its virtual-syndrome-flip port together with its boundary-error port, or its boundary-error-coordinate port in the compressed form; the fault, detector-flip, and observable-flip ports are free.

Theorem 4.6 (Composition of extended detector error models)**.** *The raw extended detector error model of the composite gadget is the contraction of the raw-EDEM diagram of the gadget circuit,*

$$\mathrm{EDEM}_G = [\![\mathrm{EDEM}[\mathfrak{G}]]\!], \tag{25}$$

in which each gadget call g is replaced by EDEM_g *and each code-block wire by a boundary-error wire and a virtual-syndrome-flip wire. For compressed extended detector error models, with boundary-error-coordinate wires in place of boundary-error wires, Eq. (25) holds up to the choice of output representative (Remark 4.2). In particular, the detector and deterministic observable matrices of a closed experiment agree exactly; interface coordinates and random-observable rows may depend on the representative.*

The proof assembles three facts about the diagram contractions. First, $\mathrm{RFE}_G = [\![\mathrm{RFE}[\mathfrak{G}]]\!]$ by Eq. (22); Fig. 9 shows the RFE diagram and its contraction for the Bell experiment. Second, $\mathrm{DM}_G = [\![\mathrm{DM}[\mathfrak{G}]]\!]$ holds by construction, and $\Delta\mathrm{DM}_G = [\![\Delta\mathrm{DM}[\mathfrak{G}]]\!]$ follows because taking linear parts commutes with composition, Eq. (13). Third, $\Delta\mathrm{OM}_G = \bigoplus_g \Delta\mathrm{OM}_g$, since the outcome map of the composite gadget is the direct sum of the outcome maps of its gadget calls, Eq. (17). Now draw EDEM_G as in Fig. 8 in terms of RFE_G, $\Delta\mathrm{DM}_G$, and $\Delta\mathrm{OM}_G$, and substitute the diagrams $\mathrm{RFE}[\mathfrak{G}]$, $\Delta\mathrm{DM}[\mathfrak{G}]$, and $\bigoplus_g \Delta\mathrm{OM}_g$ for them. The result has three vertices per gadget call, RFE_g, $\Delta\mathrm{DM}_g$, and $\Delta\mathrm{OM}_g$, and the only wires between different gadget calls are the boundary-error wires of $\mathrm{RFE}[\mathfrak{G}]$ and the virtual-syndrome-flip wires of $\Delta\mathrm{DM}[\mathfrak{G}]$, both inherited from $\mathfrak{G}$. Grouping the three vertices of each gadget call into a single vertex reproduces EDEM_g as drawn in Fig. 8. The wiring that remains between the grouped vertices is that of $\mathfrak{G}$, so the grouped diagram is $\mathrm{EDEM}[\mathfrak{G}]$.

For extended detector error models one further fact is needed: inserting a code-presentation map followed by a section (Def. 4.5) on a boundary-error wire of $\mathrm{RFE}[\mathfrak{G}]$ leaves the contraction un-

Table 1: Named classical bitvector port types, grouped by functionality.

Abbrev.	Full name	Description
`F`	Fault configuration	A bitvector selecting a combination of elementary faults; the compressed variant uses a smaller coordinate set with the same attainable effects.
`E`	Boundary error	A phase-free Pauli label carrying the residual Pauli effect of earlier faults across a gadget boundary, up to the choice of output representative (Remark 4.2).
`(S, L)`	Boundary-error coordinates	The syndrome and logical-effect components selected by a code presentation; (s, l) denotes a value of these ports.
`V`, `ΔV`	Virtual syndrome and flip	Stabilizer signs inferred from classical data, and their fault-induced changes.
`M`, `ΔM`	Measurement outcome and flip	A physical measurement outcome and its flip induced by faults and boundary errors.
`D`, `ΔD`	Detector value and flip	A declared detector parity and its fault-induced violation, obtained using DM and ΔDM, respectively.
`O`, `ΔO`	Observable value and flip	A declared logical-observable value and its fault-induced flip, obtained using OM and ΔOM, respectively.

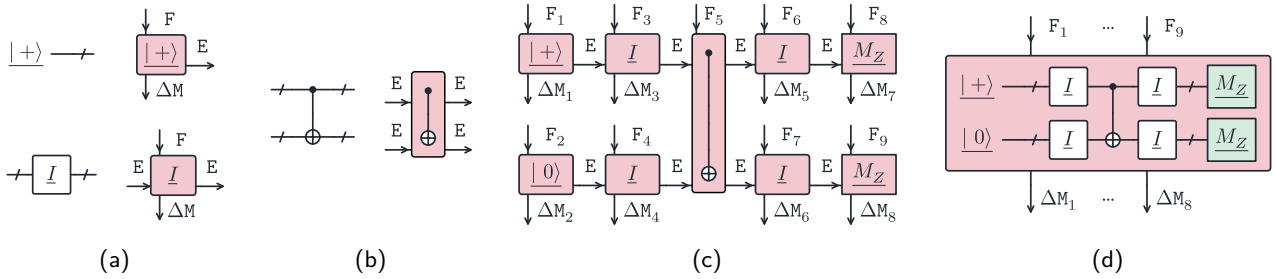


Figure 9: Raw fault-effect maps with boundary-error interfaces, the counterpart of Fig. 6 for fault propagation. Subfigures a and b show logical operations and their raw fault-effect maps RFE_g (Def. 4.1). Subfigure c shows the RFE diagram $\mathrm{RFE}[\mathfrak{G}]$ of the Bell experiment in Fig. 2, in which the per-gadget maps are wired along the code blocks by boundary-error wires, and subfigure d shows its contraction, the raw fault-effect map RFE_G of the composite gadget, Eq. (22).

changed up to the choice of output representative, because the two boundary errors differ by a stabilizer of the code on that wire, which acts trivially on the codespace state at the boundary (Remark 4.2). Applying Eq. (23) to every gadget call therefore compresses every boundary-error wire of $\mathrm{EDEM}[\mathfrak{G}]$ to a boundary-error-coordinate wire while preserving the represented fault effects up to this freedom.

For subsequent formulas involving compressed EDEMs, we choose the composite representative induced by contracting the fixed per-gadget compressed maps. Regrouping these fixed maps, as in the window construction below, then gives literal matrix equality.

The theorem applies to any part of a gadget circuit that can itself be regarded as a single gadget call.

Definition 4.7 (Window)**.** A set W of gadget calls of $\mathfrak{G}$ is *convex* if every directed path between two calls in W stays in W. The *restriction* $\mathfrak{G}|_W$ is the sub-network with boxes W, the wires with both ends in W, and a boundary port for every wire entering or leaving W (Sec. 2.4). A *window* is a convex set of gadget calls, identified with its restriction, and $\mathfrak{G}/W$ denotes the gadget circuit in which W is replaced by a single call whose gadget is the composite gadget G_W of $\mathfrak{G}|_W$.

Corollary 4.8 (Window EDEM)**.** *For a window W of $\mathfrak{G}$, $\mathrm{EDEM}_{G_W} = [\![\mathrm{EDEM}[\mathfrak{G}|_W]]\!]$, and replacing the boxes of W in $\mathrm{EDEM}[\mathfrak{G}]$ by the single box EDEM_{G_W} leaves the contraction unchanged.*

We call EDEM_{G_W} the *window EDEM* of W. Section 7 uses it as the decoding problem of a window of an adaptive computation.

Corollary 4.9 (Causality)**.** *Fix a topological order of the gadget calls of $\mathfrak{G}$ and assign every detector of the induced gadget contract to the gadget call in which it is fully computable, that is, to the box of $\mathrm{EDEM}[\mathfrak{G}]$ that outputs it. Then the fault-to-detector block of EDEM_G, and hence the channel-check matrix A_{D} of a closed experiment, is block lower triangular: a fault in a gadget call*

Table 2: Linear maps used to construct detector error models and extended detector error models. For an affine map f, Δf denotes its linear part, as defined in Eq. (13). The final column lists input and output ports explicitly; ports subscripted in and out are interface ports, attached to input and output codes and wired along the code blocks of a gadget circuit (Def. 2.10), and the remaining ports are free. Badge colors match the corresponding diagram components, with green reserved for compression maps.

Abbrev.	Full name	Description	Ports
DEM	Detector error model	Bitmatrix part of a closed-experiment DEM (Sec. 2.5), mapping fault configurations to detector and observable flips.	**in:** $\mathtt{F}$ **out:** $\Delta\mathtt{D}$, $\Delta\mathtt{O}$
Raw EDEM	Raw extended DEM	Open-boundary generalization of a DEM that retains boundary errors and virtual-syndrome flips.	**in:** $\mathtt{F}$, $\Delta\mathtt{V}_{\text{in}}$, $\mathtt{E}_{\text{in}}$ **out:** $\Delta\mathtt{D}$, $\Delta\mathtt{O}$, $\Delta\mathtt{V}_{\text{out}}$, $\mathtt{E}_{\text{out}}$
EDEM	Extended DEM	Raw EDEM whose boundary-error ports are compressed to the boundary-error coordinates selected by the code presentations, Eq. (23).	**in:** $\mathtt{F}$, $\Delta\mathtt{V}_{\text{in}}$, $(\mathtt{S},\mathtt{L})_{\text{in}}$ **out:** $\Delta\mathtt{D}$, $\Delta\mathtt{O}$, $\Delta\mathtt{V}_{\text{out}}$, $(\mathtt{S},\mathtt{L})_{\text{out}}$
ΔOM	Outcome map (linear part)	Maps physical measurement-outcome flips to observable flips.	**in:** $\Delta\mathtt{M}$ **out:** $\Delta\mathtt{O}$
ΔDM	Detector map (linear part)	Maps measurement-outcome and input virtual-syndrome flips to detector and output virtual-syndrome flips.	**in:** $\Delta\mathtt{M}$, $\Delta\mathtt{V}_{\text{in}}$ **out:** $\Delta\mathtt{D}$, $\Delta\mathtt{V}_{\text{out}}$
RFE	Raw fault-effect map	Propagates an input boundary error and a fault configuration to outcome flips and an output boundary error (Def. 4.1).	**in:** $\mathtt{F}$, $\mathtt{E}_{\text{in}}$ **out:** $\Delta\mathtt{M}$, $\mathtt{E}_{\text{out}}$
CPM	Code-presentation map	Converts a boundary error into the boundary-error coordinates selected by a code presentation.	**in:** $\mathtt{E}$ **out:** $(\mathtt{S},\mathtt{L})$
FCM	Fault-compression map	Expands compressed fault coordinates without changing the image of the raw fault-effect map.	**in:** compressed $\mathtt{F}$ **out:** $\mathtt{F}$

can flip only detectors assigned to that call or to gadget calls in its causal future.

Proof. By Theorem 4.6, the only wires between different gadget calls of EDEM[$\mathfrak{G}$] are the boundary-error and virtual-syndrome-flip wires inherited from the acyclic gadget circuit $\mathfrak{G}$, so a path from the fault port of a gadget call to a detector port of another exists only if the second call lies in the causal future of the first. The block of a contracted diagram between two ports that no path connects is zero (Sec. 2.8), which is the claim. □

4.5 Elementary-fault compression

We can further reduce the dimension of the fault-configuration wires by retaining a minimal set of coordinates with the same attainable effects.

Definition 4.10 (Fault-compression map)**.** For the restriction $R : \mathbb{F}_2^{\mathsf{F}} \to Y$ of a raw fault-effect map to its fault ports, a *fault-compression map* $\mathrm{FCM} : \mathbb{F}_2^{\mathsf{F}_c} \to \mathbb{F}_2^{\mathsf{F}}$ is a linear map satisfying

$$\mathrm{im}(R \circ \mathrm{FCM}) = \mathrm{im}(R). \tag{26}$$

When the compressed coordinates select elementary faults, a minimal choice selects effect columns forming a basis of $\mathrm{im}(R)$.

This compression preserves attainable effects, but the compressed coordinates need not be independent stochastic faults. To retain the original noise model, record each elementary fault α by a compressed coordinate vector c_α satisfying $R\,\mathrm{FCM}\,c_\alpha = Re_\alpha$, where e_α is its unit fault vector, together with its original probability p_α. The compressed maps propagate c_α, while the probabilities remain attached to the original independent mechanisms. If the compressed coordinates themselves are used as random variables, their induced joint distribution must be retained; a single original mechanism can flip several coordinates and thereby correlate them.

Despite this compression, elementary faults in different gadgets can still have the same fault effect in the composite gadget. For this reason, the post-processing step that merges elementary faults with the same effect and updates the fault probabilities using Eq. (16) is still necessary.

5 Symbolic construction of extended detector error models

Theorem 4.6 expresses the extended detector error model of a gadget circuit as the contraction of a map diagram, which gives us flexibility in how to contract it, that is, how to compute the linear map $[\![\mathfrak{D}]\!]$ defined by a diagram $\mathfrak{D}$ (Sec. 2.8). In this section we explore a symbolic approach, in which contraction produces polynomial expressions for the blocks of the result and the evaluation of these expressions is deferred and shared.

5.1 Multi-matrix-variable polynomials

The key observation is that when we contract a diagram, the result is a block matrix whose blocks are multi-matrix-variable polynomials in variables corresponding to the blocks of the component maps (Fig. 10). The matrix variables generally do not commute, every product must have compatible dimensions, and polynomial coefficients lie in $\mathbb{F}_2$. When the blocks are sparse and many blocks are repeated, we can first evaluate the blocks as multi-matrix-variable polynomials and then evaluate the matrix products. This elucidates which blocks are repeated and ensures that the polynomial for each repeated block is evaluated only once.

The symbolic evaluation of diagrams proceeds in three steps. First, we analyze each distinct component diagram and assign a variable to each distinct matrix. At this stage, in addition to assigning variables, we also identify zero and identity matrices. Second, we contract the diagram in terms of the introduced variables and obtain a polynomial expression for each block. Third, we evaluate the distinct polynomial expressions. Interestingly, the second stage of symbolic evaluation can be the same across distances within the same family of fault-tolerant protocols, and even across families of fault-tolerant protocols, as we illustrate in the next subsection.

5.2 Syndrome extraction repetition example

In this section we symbolically analyze the extended detector error model of repeated syndrome extraction rounds. We consider a syndrome extraction gadget that, together with its gadget detector contract, satisfies the following assumptions:

1. the gadget has the same input and output code with the same code presentation, and the code presentation has an independent generator list G (Def. 4.3)

2. the action of the gadget's realization is equivalent to measuring the independent set of generators G; in particular, the sign of each measured generator is an affine function of the gadget's measurement outcomes

3. the gadget logical action is the identity on all logical qubits

4. the gadget has $|G|$ virtual detectors, each being the parity of the input syndrome bit and the measurement outcomes that determine the sign of the corresponding generator

5. each output virtual syndrome bit equals the parity of the measurement outcomes that determine the sign of the corresponding generator

Note that not every syndrome extraction gadget satisfies the above assumptions. Many common syndrome extraction circuits for CSS codes do satisfy them, and thus serve as a good illustration of symbolic evaluation techniques. Appendix C gives the corresponding analysis for subsystem and Floquet syndrome extraction, where the group measured from a cycle's input can differ from the stabilizer group prepared at its output and the ideal output may carry an outcome-dependent Pauli frame.

Next we analyze the repetition of the extended detector error model for the syndrome extraction gadget in Fig. 11; by Theorem 4.6, contracting this chained diagram yields the extended detector error model of the composite gadget of the consecutive rounds. We use the following bitvector variables:

f_k– fault configurations
l_k– logical coordinate of the boundary error
s_k– syndrome coordinate of the boundary error
v_k– virtual syndrome flips
d_k– detector flips

For some bitmatrices D_F, S_F and L_F the vari-

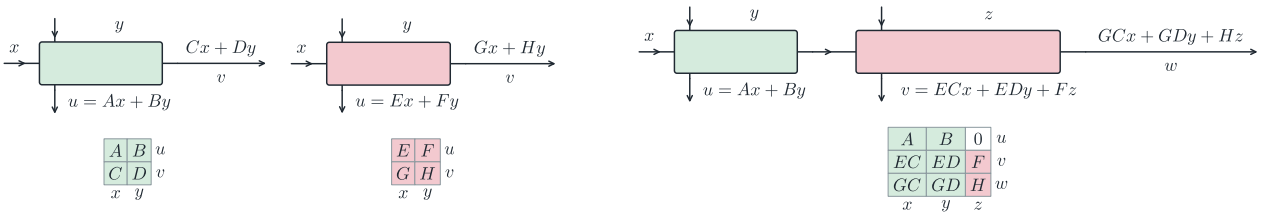


Figure 10: Linear map diagrams and matrices relating their input and output bitvectors. Diagram composition results in matrix addition and multiplication. The blocks of the composition contain multi-matrix-variable polynomials. Note how an output being insensitive to an input, such as u to z is self evident both from the diagram composition structure and the 0 block in the block-matrix representation.

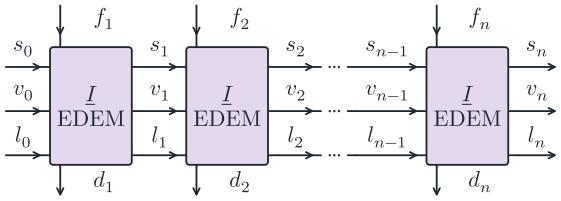


Figure 11: Extended detector error model repetition. The repeated diagram has the input and output structure of a syndrome extraction round. The variables f_k are fault configurations, s_k are the syndrome components of the boundary-error coordinates, v_k are virtual syndrome flips, l_k are the logical-effect components of the boundary-error coordinates, and d_k are detector flips. Here and in the following derivation, we suppress Δ in v_k and d_k.

ables above satisfy the following equations:

$$l_k = L_F f_k + l_{k-1} \tag{27}$$
$$s_k = S_F f_k + s_{k-1} \tag{28}$$
$$v_k = D_F f_k + s_{k-1} \tag{29}$$
$$d_k = D_F f_k + s_{k-1} + v_{k-1} \tag{30}$$

These follow from our assumption on the syndrome extraction gadget. Since every relation is linear, we check the contribution of each input separately. The matrices D_F, S_F, and L_F capture how the gadget's faults affect detectors and the output boundary-error coordinates.

Logical effect (27): a fault-free gadget carries the logical-effect component of the input boundary-error coordinates to the output unchanged, and this component does not depend on the input virtual syndrome flips—the latter holds for any gadget, because the output boundary-error coordinates of an extended detector error model depend only on the faults and the input boundary error (see the structural independences stated after Def. 4.4).

Syndrome (28): a fault-free gadget preserves the syndrome component of the input boundary-error coordinates because its action is to measure the code stabilizers. As with the logical-effect component, the syndrome component does not depend on the input virtual syndrome flips or on the logical-effect component.

Output virtual syndrome flips (29): by Def. 4.4 and our choice of expressions for these flips in the gadget contract, a fault-free gadget outputs the syndrome component of the input boundary-error coordinates—the same s_{k-1} dependence as Eq. (28), but carried forward as a virtual-syndrome flip rather than a boundary-error coordinate. Faults enter here and in the detector flips through the same matrix D_F, because both are determined by the parities of the measurements used to fix the stabilizer signs.

Detector flips (30): by the choice of virtual detectors, a fault-free gadget with zero incoming boundary-error coordinates reports the input virtual-syndrome flips, and a fault-free gadget with no input virtual-syndrome flips reports the incoming syndrome coordinate.

We can turn the matrix-vector recurrences into closed-form expressions and verify them symbolically (Appendix D), with the result given below:

$$v_k = D_F f_k + s_0 + S_F f^{\Sigma}_{k-1}, f^{\Sigma}_k = \sum_{j=1}^{k} f_j \tag{31}$$
$$s_k = s_0 + S_F f^{\Sigma}_k,\ l_k = l_0 + L_F f^{\Sigma}_k \tag{32}$$
$$d_k = D_F f_k + (D_F + S_F) f_{k-1}, \quad (k \geq 2) \tag{33}$$

We see in Eq. (33) that detector flips in step k depend only on faults from steps k and $k-1$ for $k \geq 2$. This is what gives the repeated-extraction part of memory-experiment detector error models its banded structure; Sec. 7.3 recasts it as a detector-span certificate obtained from the gadget blocks alone (Lemma 7.2).

A related use of repetition appears in Stim's `REPEAT` blocks [4], which compactly represent repeated circuit and detector-error-model instructions. Here, repeated gadget composition yields symbolic expressions for the model's matrix blocks.

The above example also illustrates that the block matrix corresponding to the EDEM of repeated syndrome extraction has substantial redundancy; many blocks are zero, many are identity and the rest repeat among S_F, L_F, D_F, $D_F + S_F$. In a software implementation, we can simply store pointers to the distinct blocks when representing the extended detector error model, which makes this representation compact and reduces both its memory and its communication requirements. For example, a decoder unit (such as a GPU, FPGA or ASIC) may have individual blocks pre-loaded[1]. When a decoding coordinator assigns it a specific decoding task, it need only describe the block structure, thus minimizing the communication requirement for specifying the EDEM.

5.3 Caching and pre-computation

When analyzing many logical circuits implemented using the same fault-tolerant protocols, their detector error models share blocks with the same polynomial expressions in the block matrices. To speed up the instantiation of many such experiments, one can cache the results of evaluating the expressions and amortize their cost across many circuits. The expressions can be reused across parameter choices that preserve the block structure and identities. Cached matrix values are specific to the block assignment and dimensions. We can also prime the cache by evaluating polynomials that occur in common sub-circuits, such as various logical operations surrounded by syndrome extraction rounds.

6 Examples and software implementation

We implemented the constructions of Secs. 3–5 in a Rust prototype and used it to build and check extended detector error models for several code families and experiments. The prototype covers gadgets and their verification, gadget detector contracts, per-gadget EDEM construction, EDEM composition by diagram contraction, and the symbolic evaluation of Sec. 5. Its instruction set covers the repetition code, the rotated surface code, the two-dimensional toric code, and the four-dimensional toric code; for each code it provides state-preparation, syndrome-extraction, destructive-measurement, and transversal-CNOT gadget types with verified contracts, and a transversal-Hadamard gadget type where the code admits one. Code presentations with overcomplete generator lists (Sec. 4.2) are supported and exercised by the four-dimensional toric code.

6.1 Implementation overview

The central data structure is the extended detector error model of Sec. 4.3 in its compressed form: a block bitmatrix whose columns are grouped into the input boundary-error coordinates (s, l), input virtual-syndrome flips, and elementary faults, and whose rows are grouped into detector flips and the corresponding output coordinates and virtual-syndrome flips. The column groups correspond to the variables s, v, l, and f of Sec. 5.2, and the detector rows to d. The observable-flip rows of Def. 4.4, obtained through $\Delta\mathrm{OM}$, are stored in the same block bitmatrix; beyond the definition, the implementation also carries success-bit rows for postselection (Sec. 3.3). Each block is stored once and shared by reference; zero and identity blocks require no storage.

Each gadget type is analyzed exactly once. Fault propagation traverses the circuit once in reverse, recording which detectors and boundary outputs each elementary fault flips.

The analyzed gadget types form an instruction set, and an experiment is a gadget circuit built from calls to these types. The experiment's DEM is assembled by wiring the per-call EDEMs along shared code blocks and contracting the syndrome, virtual-syndrome, and logical wires, as in Sec. 4.4. The composed bitmatrix retains the partition of detectors and faults by gadget call, which is the block structure used in Sec. 7; fault probabilities are tracked per gadget and follow the fault columns through composition.

The resulting diagram can be evaluated by direct dense contraction over $\mathbb{F}_2$ or by the symbolic method of Sec. 5. In the symbolic method, each

[1] This technique is reminiscent of how sprites were used in early 2D video games to keep animations fluid without redrawing every pixel.

distinct block is assigned a variable once per instruction set; contraction then produces a polynomial expression for each result block, and each distinct expression is evaluated once. An optional cache stores the block matrix values of expressions that recur across experiments (Sec. 5.3).

6.2 Correctness checks

Gadgets and gadget detector contracts are verified by Clifford simulation, following Ref. [19]. The realization sandwiched between encoders and unencoders is compared against the action under the outcome map (Def. 2.8). The declared detectors, output virtual syndrome relations, success bits and deterministic logical outcomes are checked for determinism and completeness using the contract-correctness conditions, including the rank condition, of Sec. 4.2.

A slower forward propagation of every fault provides an independent check of the reverse fault propagation; the two produce identical EDEMs on all gadget types of the instruction set.

Composition is tested directly. For every code in the instruction set and every tested round count n, the EDEM of n syndrome-extraction rounds built from the flattened circuit equals the contraction of n single-round EDEMs. The EDEMs of memory, Hadamard, and CNOT experiments built from gadget calls agree with contractions of the corresponding per-stage EDEMs. The block-level decomposition of an EDEM into the raw fault-effect map, ΔDM, ΔOM, and the code-presentation maps (Fig. 8 and Eq. (23)) is likewise reproduced for the surface and both toric codes.

Direct dense contraction and symbolic evaluation produce bit-for-bit identical DEMs on a test suite covering memory experiments on all four codes and Hadamard, CNOT, and GHZ experiments on the surface and toric codes.

Finally, the DEMs obtained by symbolic EDEM composition are cross-checked against Stim [4], which constructs the DEM from the complete physical circuit and its declared detectors and logical observables, without using the gadget decomposition. Table 3 specifies the tested code sizes, round counts, noise parameters, Stim version, and comparison criterion. The checks cover memory, transversal Hadamard and CNOT, and three- and five-block encoded GHZ experiments, with all 1,836 code–size–round–experiment–basis–noise configurations passing. The nonempty detector–observable effect sets agree exactly, and the largest absolute difference between merged effect probabilities is below 4.2×10^{-14}.

6.3 Examples

Memory and GHZ experiments illustrate the block structure of the composed DEMs.

Memory experiments. For a single code block undergoing preparation, r syndrome-extraction rounds, and destructive measurement, the composed DEM reproduces, on the syndrome-extraction rounds, the banded structure derived in Sec. 5.2: each gadget call contributes nonzero blocks only for itself and the preceding call.

GHZ states on n logical qubits. A ladder of transversal CNOTs across n code blocks, with syndrome-extraction rounds after each rung and a destructive measurement of every block, prepares and verifies an n-qubit logical GHZ state using $n^2 + 2n - 1$ gadget calls at one round per rung. This experiment exercises parallel composition: each CNOT rung contributes blocks that relate faults in one code block to detectors in another. For the tested values of n up to 12 at distance 3, the composed DEM has lower block bandwidth $b = n + 1$ ($b = n$ under Z-only noise) in the chosen topological order of gadget calls. Here b is the largest separation $t - s$ of a nonzero block relating detectors of call t to faults of call s, so nonzero blocks satisfy $0 \le t - s \le b$. Since $\mathrm{dist}(s, t) \le t - s$ for causally related calls, this also bounds the graph-distance detector span of Eq. (39) by b. This bound need not be tight for a multiport gadget circuit. Only about $2n^2$ of the $(n^2 + 2n - 1)^2$ block-grid entries are nonzero, so the block representation becomes sparser as n grows. The GHZ correlations appear as deterministic observables, the shared readout bit as a random one.

6.4 Prototype limitations

The current version only supports independent stochastic noise over the elementary Pauli faults of Sec. 2.4. These include Pauli insertions on the individual gadget circuits and outcome bit flips.

Table 3: Validation of symbolically composed EDEMs against Stim. Each row specifies the Cartesian product of the listed sizes, round counts, and experiments, tested in both bases under all three noise presets. Fractions count passing/tested DEM comparisons; every comparison checks the complete set of nonempty detector–observable effects and every associated probability.

Code	Size	SE rounds r	Experiments	Agreement
Repetition	$n = 3, 5, 7$	$2, 10, 30$	M, C, G_3, G_5	216/216
	$n = 15, 31, 63, 127, 255$	$10, 100$	M, C, G_3, G_5	240/240
	$n = 3, 5$	100	G_3, G_5	24/24
Rotated surface	$d = 3, 5, 7$	$2, 10, 30$	M, H, C, G_3, G_5	270/270
	$d = 11, 15, 21, 31$	$2, 10$	M, H, C, G_3, G_5	240/240
	$d = 5$	$100, 300, 1000$	M, H, C	54/54
	$d = 3, 5$	100	G_3, G_5	24/24
2D toric	$d = 3, 5, 7$	$2, 10, 30$	M, H, C, G_3, G_5	270/270
	$d = 11, 15, 21$	$2, 10$	M, H, C, G_3, G_5	180/180
	$d = 5$	$100, 300, 1000$	M, H, C	54/54
	$d = 3, 5$	100	G_3, G_5	24/24
4D toric	$L = 2, 3, 4$	$2, 10$	M, H, C, G_3, G_5	180/180
	$L = 2$	$30, 100$	M, H, C, G_3, G_5	60/60
Total DEM comparisons				**1,836/1,836**
Compared effect probabilities				496 696 358
Largest absolute probability difference				4.16×10^{-14}

Circuits and sizes. M, H, C, and G_m denote memory, Hadamard, CNOT, and m-block GHZ experiments. Here r counts explicit SE rounds per block in each segment: M has one segment, H and C have one before and one after the gate, and G_m has one after each rung of a nearest-neighbor CNOT ladder, starting from $|+\rangle|0\rangle^{\otimes(m-1)}$. Thus G_5 has $4r$ rounds per block. M and C use matching preparation/readout bases; H uses dual readout. GHZ readout checks adjacent logical ZZ parities in the Z basis and the global logical $X^{\otimes m}$ parity in the X basis, for every encoded logical index. Repetition size is the number n of data qubits; d is code distance for surface and 2D toric codes. For 4D toric, L is lattice side length, with distance L^2 and $6L^4$ data qubits per block; the largest five-block experiment contains 7,680 data qubits.

Noise and comparison. All runs use release builds and Stim [4] 1.16.dev0, with decomposition, loop folding, and gauge detectors disabled and disjoint-error approximation threshold zero. At $p = 10^{-2}$, the presets are independent X/Z faults and independent X/Y/Z faults, each at probability p per surviving operation-target qubit, and all nonidentity joint Pauli/readout patterns on an operation with q surviving targets and b output bits, each independently at $p/(4^q 2^b - 1)$. Identical effects are merged as $P_e = \frac{1}{2}[1 - \prod_{f:e_f=e}(1-2p_f)]$. The acceptance bound is $|P_e - P_e^{\text{Stim}}| \le 10^{-12}(1 + |P_e| + |P_e^{\text{Stim}}|)$. All effect sets agree exactly. The maximum discrepancy above is measured over every compared effect probability.

Including cross-gadget correlated noise is an ongoing line of work.

The prototype composes gadgets along matching ports, with no cross-gadget feed-forward beyond parity-controlled Pauli corrections. It currently produces monolithic DEMs. Although the output retains the per-call partition needed for windowing, the prototype does not yet extract window EDEMs (Cor. 4.8) or automate the detector-span criterion of Lemma 7.2. It also does not represent the interface dispositions and task ownership of Sec. 7.5.

7 Window EDEMs and streaming decoding

This section connects compositional EDEM construction to the decoding problem of a long, adaptive computation. It makes three points. First, the object a windowed decoder consumes is the *window EDEM*: the contraction of the EDEM diagram restricted to a convex set of gadget calls, which by Theorem 4.6 is again an EDEM. Second, the two structural properties windowing requires, causality and bounded detector span, can be read off the per-gadget EDEM blocks, so the symbolic analysis of Sec. 5 also certifies them. Third, decoding tasks need not follow the causal order of

the calls; we distinguish their data dependencies from quantum execution and explain the interface conditions needed to express them compositionally. We prescribe no decoding algorithm and do not analyze accuracy.

7.1 Incremental construction and outcome interpretation

In a fault-tolerant computation, decoded logical outcomes determine which logical gadgets the quantum processing unit (QPU) executes next. The gadget circuit $\mathfrak{G}$ (Def. 2.9) is thus unknown in advance; it grows one gadget call at a time, in a topological order fixed by the instruction stream. Recall that a gadget call is a box of the gadget circuit, one use of an elementary gadget at a specific place in the computation; we write *call* for short, and say a call is *issued* when the control system queues its physical instruction to the QPU.

Two consequences follow. The EDEM diagram $\mathrm{EDEM}[\mathfrak{G}]$ must be assembled incrementally, in step with the gadget calls issued to the QPU. Decoding must begin before the computation ends, so that logical outcomes can drive control flow within an acceptable *reaction time* [14], at a throughput that keeps pace with the growing decoding problem [3, 15].

The same per-gadget maps serve both needs. When the control system chooses the next gadget call g, it queues the physical instruction to the QPU and the precomputed EDEM_g, with its detector map DM_g and outcome map OM_g, to the decoding unit. Appending g to $\mathrm{EDEM}[\mathfrak{G}]$ touches only these stored maps and the wires at the ports of g.

When g completes, DM_g evaluated on its physical outcomes and on the virtual syndromes arriving at its input wires yields the detector values g declares and the virtual syndromes it passes on its output wires; OM_g yields the raw logical outcomes g declares. The structure that builds the EDEM thus also interprets QPU outcomes as they stream in. The decoding unit may receive either the physical outcomes of each call or only the detector values computed from them. Since DM_g and OM_g are affine (Def. 3.6), the latter is $\mathbb{F}_2$ linear logic that can run in an intermediate unit close to the QPU, which then forwards only detector values, output virtual syndromes, and observable values. This reduces the communication load when a realization produces many more measurement outcomes than detectors, such as in Floquet [32] and fusion-based [27] fault tolerance.

7.2 Window extended detector error models

A gadget circuit is an acyclic network of gadget calls (Def. 2.9). We use the directed graph its wires induce on the calls, with an edge from s to t whenever an output code of s is wired to an input code of t. We write $s \preceq t$ if $s = t$ or a directed path leads from s to t, call $\preceq$ the *causal order* on calls, and write $\mathrm{dist}(s, t)$ for the length of a shortest such path, or ∞ if there is none.

A window (Def. 4.7) is a convex set W of calls, a gadget circuit in its own right and at the same time a single call of the coarser circuit $\mathfrak{G}/W$. Its window EDEM_{G_W} is the contraction of the restricted diagram, and W may be replaced by this single box inside $\mathrm{EDEM}[\mathfrak{G}]$ without changing the contraction (Corollary 4.8). Its free ports are the fault vectors $\mathsf{F}_W = \bigsqcup_{g \in W} \mathsf{F}_g$, the detector flips D_W, and the observable flips of the calls in W. Its interface ports are a virtual-syndrome-flip port and a boundary-coordinate port for every wire entering or leaving W.

Two windows appear in each step of sequential decoding, and they play different roles. The *inference window* W fixes the decoding problem: its detectors supply the evidence and its elementary faults define the inference variables. The *commit window* $C \subseteq W$ fixes what is passed on. Its output boundary is the cut along which decoding composes, and this cut runs through the interior of W, between the committed calls and the *buffer* $W \setminus C$ (Fig. 12). A committed region has a past and a future, and the interface ports of its EDEM carry what crosses into the buffer and beyond; this is why the right error model for it is an EDEM and not a closed DEM. Per-gadget blocks supply the rows, columns, and interface maps of both windows, so no physical circuit is revisited and the DEM of the whole computation is never materialized.

We use the block notation of Eq. (24), with $b = (s, l)$ the boundary coordinates and Δv the virtual-syndrome flips. The same block form holds for a window EDEM with g replaced by W.

Turning an inference window into a decoding problem requires a *disposition* for every wire that touches it. For the sequential orientation considered here, there are three.

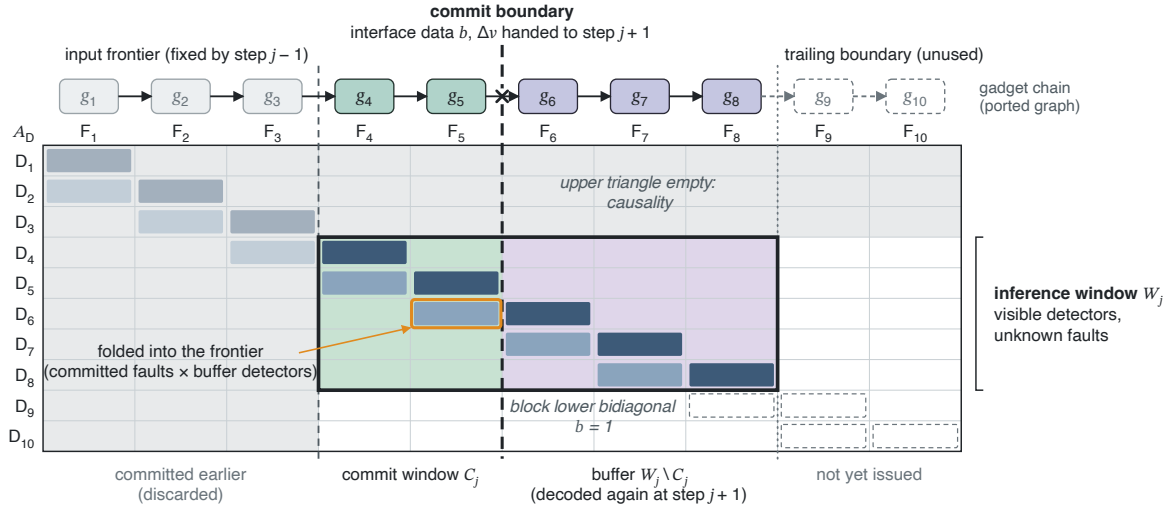


Figure 12: One step of sequential sliding-window decoding on the block-banded channel-check matrix A_{D} of a chain of gadget calls $g_1 \to \cdots \to g_{10}$ (top strip). Column blocks are the fault partitions F_t and row blocks the detector partitions D_t. Causality empties the upper triangle. The figure assumes detector span $w = 1$, so nonzero blocks lie on the diagonal and the first subdiagonal; Sec. 5.2 and App. C provide locality conditions for the two-block band for repeated stabilizer, subsystem, and Floquet syndrome extraction, and Sec. 7.3 discusses gadgets that widen it. At step j the inference window $W_j = \{g_4, \ldots, g_8\}$ (black outline) supplies the visible detectors and the fault variables included in the inference problem. Its commit window $C_j = \{g_4, g_5\}$ (green) is separated from the buffer (purple) by the commit boundary (bold dashed line), which cuts the wire between g_5 and g_6 in the interior of W_j; the interface data b and Δv on that wire are handed to step $j+1$. The committed correction reaches later detectors only through the orange block $(\mathsf{D}_6, \mathsf{F}_5)$, which Proposition 7.1 folds into the frontier. Calls g_1 to g_3 (grey) were committed at earlier steps and discarded; their block $(\mathsf{D}_4, \mathsf{F}_3)$ was folded at step $j-1$ and fixes the input frontier of W_j. The trailing boundary of W_j plays no role in composition, and calls g_9 and g_{10} (dashed) are not yet issued.

- *Stitched.* Both ends of the wire lie in W. The interface disappears in the contraction of Corollary 4.8; this is composition.

- *Closed.* The wire enters W from a committed region. Its boundary coordinates b_{in} and virtual-syndrome flips Δv_{in} encode the interface effect of the previously committed correction and deterministically shift the detector values of W (Proposition 7.1 below).

- *Open.* The wire leaves W, or enters it from a region that is neither in W nor committed. Its interface data are unconstrained: fault configurations in W that differ only there are indistinguishable from the detectors of W, so the inference stops at the wire. An open wire acquires a disposition of another kind when its far side is decoded later.

A preparation or readout gadget inside W is the degenerate closed case: it has no code port on that side, so the window is closed there without boundary data from any other call. In sequential decoding the input wires of W are closed, its trailing wires are open, and all wires between its calls are stitched. Given the fixed interface data Δv_{in} and b_{in} of its closed wires, the window decoder receives the observed detector flips Δd_W and solves for a fault vector $f^\star_W$ with

$$D^W_F f^\star_W = \Delta d_W + D^W_B b_{\text{in}} + D^W_V \Delta v_{\text{in}}, \qquad (34)$$

using the fault probabilities of F_W as prior. This is an ordinary decoding problem for the channel-check matrix D^W_F, with the detector values shifted by known interface data. The logical component l of b_{in} is the accumulated Pauli frame, and $O^W_B b_{\text{in}}$ is its contribution to the logical outcomes declared in W.

A window EDEM is thus consumed by a standard decoder through a closed DEM in the sense of Sec. 2.5: the fault set F_W with its probabilities, the check matrix D^W_F, and the observable matrix O^W_F. The interface blocks never reach the decoder. Closed wires enter as the offsets $D^W_B b_{\text{in}} + D^W_V \Delta v_{\text{in}}$ on the detector values and $O^W_B b_{\text{in}}$ on the observables; open outgoing wires

contribute nothing, since their rows are not detectors; and an open past-facing wire is handled by an explicit boundary treatment, as discussed in Sec. 7.5. Decoder-specific conversions, such as decomposing hyperedges into a graphlike model, apply to this DEM as to any other. The decoder returns a fault vector on all of F_W, which is restricted to the commit window and folded. Faults of the window DEM with identical columns may be merged only if they agree on commit membership and on their interface and observable effects; otherwise the fold after decoding is wrong.

The decoder commits to a correction, or to an equivalence class of fault configurations with the same relevant effect; it does not claim to have identified the faults that occurred. The next window needs only the effect of the committed correction on the interface, and evaluating the EDEM supplies it without any retained detector history.

Proposition 7.1 (Commit and fold)**.** *Let C be a window of $\mathfrak{G}$, let $\kappa \in \mathbb{F}_2^{\mathsf{F}_C}$ be a committed fault vector on C, and let $(b_{\text{out}}, \Delta v_{\text{out}})$ be the interface outputs of EDEM_{G_C} evaluated on κ with zero interface inputs. Then for every gadget call $t \notin C$,*

$$(A_{\mathsf{D}})_{\mathsf{D}_t,\mathsf{F}_C}\,\kappa \tag{35}$$

equals the detector flip obtained by feeding $(b_{\text{out}}, \Delta v_{\text{out}})$ into the wires leaving C in $\text{EDEM}[\mathfrak{G}/C]$, with all other inputs zero. The same holds for observable flips.

Proof. By Corollary 4.8, $\text{EDEM}[\mathfrak{G}]$ and $\text{EDEM}[\mathfrak{G}/C]$ with C replaced by EDEM_{G_C} have the same contraction. Faults propagate only forward along wires, and by convexity a wire leaving C never re-enters it, so every path from F_C to a detector outside C passes through a wire leaving C; by linearity, κ contributes through its interface outputs alone. □

The proposition concerns the commit window, not the inference window. The frontier passed to the next step is the output boundary of C; the trailing boundary of the inference window plays no role in composition. To fold a committed correction into the frontier, add b_{out} and Δv_{out} to the interface data on the wires leaving C and discard C; the interface inputs of C, fixed by earlier commits, are transported to the same wires by the fault-free part of EDEM_{G_C}. In matrix terms this is the residual update of Eq. (43) below, computed through the interface instead of the global matrix. The two halves of the handoff differ in what they carry. If every fault pattern invisible to the detectors of C is also invisible to its outgoing virtual syndromes, $\ker D_F^C \subseteq \ker V_F^C$, then a κ that reproduces every detector of C also reproduces the virtual syndromes C hands on, and the boundary coordinates b_{out} are the only part of the handoff that carries the decoder's belief. Syndrome-extraction gadgets whose virtual syndromes are the measured stabilizer signs satisfy this with $D_F^C = V_F^C$ (Sec. 5.2). In the sequential schedule, a logical outcome declared by a call $g \in C$ is final once C and its causal past have been committed, because every contributing fault then has a committed correction.

For a chain, with call t immediately following the committed region C, the proposition gives the handoff factorization

$$(A_{\mathsf{D}})_{\mathsf{D}_t,\mathsf{F}_C} = \begin{bmatrix} D_B^t & D_V^t \end{bmatrix} \begin{bmatrix} B_F^C \\ V_F^C \end{bmatrix}. \tag{36}$$

More distant calls require intervening interface transfers; general gadget circuits require the path sum described in Sec. 7.3. A decoding schedule may therefore pass any of three payloads across a cut. Passing the committed fault vector κ leaves both factors to the consumer and exposes the largest basis. Passing the interface data $(b_{\text{out}}, \Delta v_{\text{out}})$ lets the producer apply its own blocks and the consumer its own, so the two sides are coupled only through the code interface and either can be exchanged for another window EDEM with the same gadget detector contract. Passing the detector delta $(A_{\mathsf{D}})_{\mathsf{D}_t,\mathsf{F}_C}\kappa$ uses the product, computed once per producer and consumer pair, and is the smallest payload, but is specialized to one consumer. The three are equivalent only if the interface data include the boundary coordinates. For the syndrome-extraction chain the handoff block is $D_F + S_F$, whereas the virtual-syndrome factors alone give D_F; the S_F term travels through b, which is why the residual cannot be handed on as virtual syndromes only.

7.3 Causality and detector span from gadget blocks

Exactly one gadget call declares each detector, namely the call whose detector map outputs it (Corollary 4.9). The detector-cutting construction of Sec. 3 thus induces canonical partitions

over gadget calls,

$$\mathsf{D} = \bigsqcup_t \mathsf{D}_t, \qquad \mathsf{F} = \bigsqcup_t \mathsf{F}_t. \tag{37}$$

Here A_{D} is the fault-to-detector block of the EDEM of the computation executed so far (Sec. 4.3), which for a closed experiment is the channel-check matrix of Sec. 2.5. Every path in EDEM[$\mathfrak{G}$] from a fault port of s to a detector port of t follows wires of the gadget circuit, so by Corollary 4.9

$$(A_{\mathsf{D}})_{\mathsf{D}_t,\mathsf{F}_s} = 0 \qquad \text{unless } s \preceq t. \tag{38}$$

In any topological order of the gadget calls the channel-check matrix is block lower triangular: a fault cannot flip a detector outside its causal future.

Causality alone does not bound the size of a window, because a fault could flip detectors arbitrarily far downstream. We say that $\mathfrak{G}$ with its induced gadget detector contract has *detector span* w if

$$(A_{\mathsf{D}})_{\mathsf{D}_t,\mathsf{F}_s} = 0 \qquad \text{whenever } \operatorname{dist}(s,t) > w. \tag{39}$$

Together, Eqs. (38) and (39) make A_{D} block local with respect to the gadget circuit. For a chain in its natural order, detector span w bounds the lower block bandwidth $b \le w$, so the matrix occupies at most $w+1$ block diagonals, including the main diagonal, as in Fig. 12. Bandwidth depends on the chosen serialization of the causal order; detector span is intrinsic to the gadget graph and its detector contract.

Detector span is a property of the per-gadget blocks, not an assumption imposed from outside. In the notation of Eq. (24), define the *interface transfer* of a call g as the map from its incoming to its outgoing interface data with faults set to zero,

$$T^g = \begin{bmatrix} B_B^g & 0 \\ V_B^g & V_V^g \end{bmatrix} : \begin{bmatrix} b_{\text{in}} \\ \Delta v_{\text{in}} \end{bmatrix} \longmapsto \begin{bmatrix} b_{\text{out}} \\ \Delta v_{\text{out}} \end{bmatrix}. \tag{40}$$

Lemma 7.2 (Detector span of a chain)**.** *For a chain of gadget calls $g_1 \to g_2 \to \cdots$ and $s < t$,*

$$(A_{\mathsf{D}})_{\mathsf{D}_t,\mathsf{F}_s} = \begin{bmatrix} D_B^t & D_V^t \end{bmatrix} T^{t-1} \cdots T^{s+1} \begin{bmatrix} B_F^s \\ V_F^s \end{bmatrix}. \tag{41}$$

In particular, chains of identical calls g have detector span w uniformly in their length if and only if

$$\begin{bmatrix} D_B^g & D_V^g \end{bmatrix} (T^g)^k \begin{bmatrix} B_F^g \\ V_F^g \end{bmatrix} = 0 \qquad \text{for all } k \ge w. \tag{42}$$

If the interface transfer T^g acts on a q-dimensional space, the Cayley–Hamilton theorem reduces this condition to the finite checks $k = w, \dots, w+q-1$.

Proof. Contracting the chain diagram multiplies matrices along the interface wires. Detector and observable ports are free and do not feed forward. The only path from F_s to D_t therefore enters the interface through the fault columns B_F^s and V_F^s of s, travels through the fault-free transfers of the intermediate calls, and exits through the interface columns D_B^t and D_V^t of the detector rows of t. □

Bounded detector span does not require the Pauli effect of a fault to vanish. The interface transfer may carry the residual forward indefinitely; what Eq. (42) demands is that the detector rows of later calls be blind to the transported residual, which happens when the boundary coordinates and the virtual syndromes report it in a way that cancels in $\begin{bmatrix} D_B^g & D_V^g \end{bmatrix}$. Only the support of the residual on newly declared detector rows closes, and this is what lets the global EDEM grow without retained detector history. Repeated stabilizer syndrome extraction is the simplest instance: the blocks of Sec. 5.2 satisfy Eq. (42) with $w = 1$, and Eq. (41) reproduces the closed form of Eq. (33). Appendix C shows that subsystem and Floquet extraction cycles satisfy the same criterion under the locality condition of Eq. (85), so the two-block band of Fig. 12 covers these cases as well.

A gadget that measures no stabilizers, such as a transversal Clifford gadget, declares no detectors and transports the incoming virtual-syndrome flips to its outputs through a nonzero block V_V^g, which relabels and may merge them. Each such call extends the span by one edge until a later call measures the transported stabilizers; the GHZ experiment of Sec. 6.3 has measured lower block bandwidth $b = n+1$ for n code blocks, which bounds its detector span. Windowing remains possible as long as the span stays bounded. For a general gadget circuit, the entry $(A_{\mathsf{D}})_{\mathsf{D}_t,\mathsf{F}_s}$ is a sum over directed paths from s to t of products of the form in Eq. (41), and detector span w follows

if every path product of length exceeding w vanishes. Since the blocks in Eq. (41) are the matrix variables of Sec. 5, the span of a fault-tolerant instruction set can be verified symbolically, once per instruction set, with the machinery of Sec. 5.

7.4 Sequential sliding-window decoding

Sequential sliding-window decoding chooses the decoding order to agree with the causal order of the gadget calls. This is a scheduling choice, additional to the EDEM construction. Windowed decoding has a classical counterpart for convolutional codes [33] and quantum variants including modular, parallel-window, and sandwich decoding [34–37].

At step j, the inference window W_j contains the commit window C_j and a buffer $W_j \setminus C_j$. The cumulative committed region is a down-set of the causal order: every strict causal predecessor of a call in C_j lies either in C_j or in an earlier commit window. Consequently, every wire entering C_j is fixed by an earlier commit or by a physical preparation boundary. For a chain, the input frontier of W_j is the commit boundary of C_{j-1}. The decoder solves Eq. (34), commits the restriction κ_j of its estimate to C_j, and folds its effect into the frontier using Proposition 7.1. The buffer estimates remain provisional: their fault variables are included again in the next inference problem, together with newly available calls. The next window EDEM is assembled from the retained and newly appended per-gadget blocks. A corrected logical outcome is released once its measurement data are available and every fault contributing to its flip has a committed correction.

Detector span bounds the forward reach of a committed fault's direct detector effects. Including the forward neighbourhood of C_j through depth w therefore includes every detector row that its faults can flip. This gives a structural coverage condition for the inference window; the buffer needed for reliable commitment against multi-fault configurations also depends on the code, noise model, and decoder. For the matchable examples of Ref. [34], sufficiently large buffers approach monolithic-decoder performance; qLDPC examples with single-shot redundancy can benefit from windows containing only a few rounds [37].

Streaming decoders such as *Frontier* [38] and *Snowflake* [39], and the sliding-window wrapper of CUDA-Q QEC [40], motivate this interface between incrementally supplied error-model blocks and a retained decoding state. The EDEM interface describes deterministic fault effects; it does not by itself specify a decoder's state or the probabilistic information needed for noise correlations across windows.

7.5 Quantum execution and decoding order

The quantum circuit and the decoding tasks have different dependency graphs. Quantum execution follows the causal order $\preceq$ of gadget calls. Decoding follows the availability of measurement data and of interface information supplied by other decoding tasks. Both orders are acyclic, but they need not agree: a task may use data from a physically later region before an earlier region has been decoded [34–36].

A decoding task τ has visible detectors $V_\tau \subseteq \mathsf{D}$, an *inference fault set* $U_\tau \subseteq \mathsf{F}$, and a commit set $C_\tau \subseteq U_\tau$ that it owns. The variables indexed by $U_\tau \setminus C_\tau$ provide buffer context; only the estimate on C_τ is committed. Transfers of committed interface information define the edges of an acyclic task graph. Write $\rho \rightsquigarrow \tau$ when τ depends on ρ through a directed path in this graph. A task can start once its measurement data and all required predecessor outputs are available. Including another task's fault variables provisionally in a buffer does not create a dependency on that task's eventual commit. For fixed global detector coordinates, predecessor commits enter as

$$\sigma_\tau = \Delta d_{V_\tau} + \sum_{\rho \rightsquigarrow \tau} (A_{\mathsf{D}})_{V_\tau, C_\rho} \kappa_\rho. \tag{43}$$

Faults already accounted for by these offsets are excluded from U_τ. Every fault has one owner, and the schedule must arrange that each detector is finally satisfied after all commits that affect it. At an open boundary, contributions not supplied by predecessors must be included in the local inference model or the affected detector constraints deferred. These are requirements on the decoding decomposition; EDEM composition supplies fault-effect maps, not a guarantee of decoding accuracy.

The edge–vertex construction of Ref. [34] gives a task graph with two layers. Edge tasks first decode interface regions, using neighbouring block interiors as buffers. They have no decoding predecessors and can run independently as soon as

their required measurement data are available. Their commits provide boundary conditions to vertex tasks, which then decode the remaining block interiors. The committed interfaces separate these remaining problems, so the vertex tasks have closed boundaries and require no further buffers. In a task-oriented interface description, the incoming wires of each vertex task are thus all closed, including those receiving information from physically later regions. The parallel-window and sandwich constructions of Refs. [35, 36] similarly perform independent buffered tasks followed by tasks that reconcile the intervening regions. These schedules have two decoding layers, independently of the depth of the quantum circuit. A general decoding task graph may have greater depth; its dependencies are determined by the chosen handoffs. In every case, a task must also wait for its required measurement data, and adaptive quantum operations must wait for the decoded outcomes that control them.

EDEMs provide the local fault effects, virtual syndromes, and boundary-error coordinates needed for such handoffs. The presentation in Secs. 2.8–4, however, assigns interface inputs and outputs according to the physical input and output codes of each gadget. Section 7.4 uses this agreement between physical direction and decoding information flow. For more general schedules, interface information should flow from the task that produces it to the task that consumes it, whichever physical side of a cut they occupy. This concerns both virtual-syndrome information and the boundary-error effects represented in a committed correction.

We expect the EDEM perspective to extend to this separate task orientation when the required handoffs admit sufficient interface summaries and well-defined local maps, preserve the composed detector and observable effects, and have acyclic data dependencies. For the physical orientation, Proposition 7.1 establishes such a factorization; a different orientation requires its own compatible interface maps. For each chosen task orientation, the exported interface data must be computable from the task's available measurements, incoming interface data, and committed correction. The resulting handoffs must also form an acyclic dependency graph. The required generalization is therefore to the choice of interface inputs and outputs and their maps, while retaining the compositional description of fault effects. We leave its general conditions and implementation to future work.

8 Conclusion

We have shown how to construct the detector error model of a fault-tolerant circuit from the extended detector error models of its gadgets. The detector-cutting lemmas of Sec. 3 justify splitting detectors at gadget boundaries using virtual syndromes, and the extended detector error model of Sec. 4 packages each gadget's fault effects, together with virtual-syndrome interfaces and boundary-error coordinates, into a linear map over $\mathbb{F}_2$ that composes the way the gadgets do. Because the global DEM is then a contraction of a diagram of linear maps, it can be evaluated symbolically: blocks become matrix variables, repeated blocks are evaluated once, and structural facts, such as the banded structure of repeated-syndrome-extraction DEMs, can be derived in closed form (Sec. 5). A Rust prototype reproduces the DEMs that Stim derives from the corresponding flat circuits, for experiments spanning four code families, and realizes the practical benefits of analyzing each gadget type once (Sec. 6). Finally, the EDEM of any convex set of gadget calls is the decoding problem of a window; causality and detector span, the latter certified from the gadget blocks, make the channel-check matrix block local with respect to the gadget circuit, and committed corrections are folded into the frontier through the interface (Sec. 7).

The remaining limitations concern both the scope of the formalism and the capabilities of the prototype. The formalism assumes stochastic Pauli noise on stabilizer circuits, and composition recovers the detector basis induced by the chosen gadget detector contracts rather than an arbitrary externally specified basis. Noise correlations that cross gadget boundaries must be carried as explicit interface data. On the practical side, the prototype does not yet extract window EDEMs and supports no cross-gadget feed-forward beyond parity-controlled Pauli corrections. Quantum execution and decoding tasks can follow different dependency orders; expressing their handoffs through task-oriented EDEM interfaces requires compatible maps for virtual-syndrome information and boundary-error effects (Sec. 7.5).

A natural next step is to move beyond stabilizer logical action. Other directions include richer noise models, implementing window extraction on top of the per-call partition the composition already produces, allowing EDEM interface inputs and outputs to follow decoding dependencies, together with an appropriate probabilistic treatment of open cuts, and sharing symbolic evaluations across protocol families.

Contributions and use of AI tools

VK conceived the entirety of the original project, including the Rust prototype and SymPy verification scripts, which were developed with extensive AI coding tool assistance. VK presented the work with supporting slides to a broader team at Nvidia including JL and FP. AI tools were used to generate the initial drafts of Secs. 1, 6, and 8 after a draft of the rest of the paper was written. They also assisted with language and grammar editing throughout the manuscript, reorganizing material between the preliminaries and appendices, and identifying gaps in math, presentation and unclear passages. FP, VK and JL reviewed, discussed, and generalized ideas in the article and presentation thereof. FP used AI tools as a co-editor, prompting changes and alternating between role of reviewer and reviewed, requesting mechanical changes and deeper feedback. The proof exposition in Sec. 3.1 was revised with AI assistance. The proofs in Appendix B were drafted with assistance from a language model and checked by the authors. Most figures were derived from hand-drawn originals using AI tools and iterated on. Figure 12 was prompted into existence through several iterations.

Acknowledgments

We thank members of the NVIDIA Quantum teams for many helpful discussions.

References

[1] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. "Topological quantum memory". Journal of Mathematical Physics **43**, 4452–4505 (2002).

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

[3] Barbara M. Terhal. "Quantum error correction for quantum memories". Reviews of Modern Physics **87**, 307–346 (2015).

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

[5] Matt McEwen, Dave Bacon, and Craig Gidney. "Relaxing hardware requirements for surface code circuits using time-dynamics". Quantum **7**, 1172 (2023).

[6] Peter-Jan H.S. Derks, Alex Townsend-Teague, Ansgar G. Burchards, and Jens Eisert. "Designing fault-tolerant circuits using detector error models". Quantum **9**, 1905 (2025).

[7] Oscar Higgott and Craig Gidney. "Sparse Blossom: correcting a million errors per core second with minimum-weight matching". Quantum **9**, 1600 (2025).

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

[9] Pavel Panteleev and Gleb Kalachev. "Degenerate quantum LDPC codes with good finite length performance". Quantum **5**, 585 (2021).

[10] Emanuel Knill. "Quantum computing with realistically noisy devices". Nature **434**, 39–44 (2005).

[11] Christopher Chamberland, Pavithran Iyer, and David Poulin. "Fault-tolerant quantum computing in the Pauli or Clifford frame with slow error diagnostics". Quantum **2**, 43 (2018).

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

[13] Daniel Litinski. "A Game of Surface Codes: large-scale quantum computing with lattice surgery". Quantum **3**, 128 (2019).

[14] Craig Gidney and Austin G. Fowler. "Flexible layout of surface code computations using AutoCCZ states" (2019). arXiv:1905.08916.

[15] Francesco Battistel, Christopher Chamberland, Kauser Johar, Ramon W. J. Overwater, Fabio Sebastiano, Luka Skoric, Yosuke Ueno, and Muhammad Usman. "Real-time

decoding for fault-tolerant quantum computing: progress, challenges and outlook". Nano Futures **7**, 032003 (2023).

[16] Google Quantum AI and Collaborators. "Quantum error correction below the surface code threshold". Nature **638**, 920–926 (2025).

[17] Laura Caune, Luka Skoric, Nick S. Blunt, Archibald Ruban, Jimmy McDaniel, Joseph A. Valery, Andrew D. Patterson, Alexander V. Gramolin, Joonas Majaniemi, Kenton M. Barnes, Tomasz Bialas, Okan Buğdaycı, Ophelia Crawford, György P. Gehér, Hari Krovi, Elisha Matekole, Canberk Topal, Stefano Poletto, Michael Bryant, Kalan Snyder, Neil I. Gillespie, Glenn Jones, Kauser Johar, Earl T. Campbell, and Alexander D. Hill. "Demonstrating real-time and low-latency quantum error correction with superconducting qubits". Nature Communications **17**, 7383 (2026).

[18] Panos Aliferis, Daniel Gottesman, and John Preskill. "Quantum accuracy threshold for concatenated distance-3 codes" (2005). arXiv:quant-ph/0504218.

[19] Vadym Kliuchnikov, Michael Beverland, and Adam Paetznick. "Stabilizer circuit verification" (2023). arXiv:2309.08676.

[20] Michael E. Beverland, Shilin Huang, and Vadym Kliuchnikov. "Fault tolerance of stabilizer channels" (2024). arXiv:2401.12017.

[21] Microsoft (2026). code: microsoft/qdk-ec v0.4.2 commit:96dc7de.

[22] Wu et al. "deq: Universal quantum decoding at runtime". In preparation.

[23] Craig Gidney, Noah Shutty, and Cody Jones. "Magic state cultivation: growing T states as cheap as CNOT gates" (2024). arXiv:2409.17595.

[24] Craig Gidney. "stimflow: annealed utilities for creating QEC circuits". url: `https://github.com/quantumlib/Stim/tree/main/glue/stimflow`. Accessed 10 September 2026.

[25] Nicolas Delfosse and Adam Paetznick. "Spacetime codes of Clifford circuits" (2023). arXiv:2304.05943.

[26] Héctor Bombín, Chris Dawson, Ryan V. Mishmash, Naomi Nickerson, Fernando Pastawski, and Sam Roberts. "Logical blocks for fault-tolerant topological quantum computation". PRX Quantum **4**, 020303 (2023).

[27] Sara Bartolucci, Patrick Birchall, Héctor Bombín, Hugo Cable, Chris Dawson, Mercedes Gimeno-Segovia, Eric Johnston, Konrad Kieling, Naomi Nickerson, Mihir Pant, Fernando Pastawski, Terry Rudolph, and Chris Sparrow. "Fusion-based quantum computation". Nature Communications **14**, 912 (2023).

[28] Hector Bombin, Daniel Litinski, Naomi Nickerson, Fernando Pastawski, and Sam Roberts. "Unifying flavors of fault tolerance with the ZX calculus". Quantum **8**, 1379 (2024).

[29] Benjamin Rodatz, Boldizsár Poór, and Aleks Kissinger. "Fault tolerance by construction" (2025). arXiv:2506.17181.

[30] Vsevolod I. Yashin and Maria A. Elovenkova. "Characterization of non-adaptive clifford channels". Quantum Information Processing **24**, 99 (2025).

[31] Yves Lafont. "Towards an algebraic theory of boolean circuits". Journal of Pure and Applied Algebra **184**, 257–310 (2003).

[32] Matthew B. Hastings and Jeongwan Haah. "Dynamically generated logical qubits". Quantum **5**, 564 (2021).

[33] Aravind R. Iyengar, Marco Papaleo, Paul H. Siegel, Jack Keil Wolf, Alessandro Vanelli-Coralli, and Giovanni E. Corazza. "Windowed decoding of protograph-based LDPC convolutional codes over erasure channels". IEEE Transactions on Information Theory **58**, 2303–2320 (2012).

[34] Héctor Bombín, Chris Dawson, Ye-Hua Liu, Naomi Nickerson, Fernando Pastawski, and Sam Roberts. "Modular decoding: parallelizable real-time decoding for quantum computers" (2023). arXiv:2303.04846.

[35] Luka Skoric, Dan E. Browne, Kenton M. Barnes, Neil I. Gillespie, and Earl T. Campbell. "Parallel window decoding enables scalable fault tolerant quantum computation". Nature Communications **14**, 7040 (2023).

[36] Xinyu Tan, Fang Zhang, Rui Chao, Yaoyun Shi, and Jianxin Chen. "Scalable surface-code decoders with parallelization in time". PRX Quantum **4**, 040344 (2023).

[37] Shilin Huang and Shruti Puri. "Increasing memory lifetime of quantum low-density par-

ity check codes with sliding-window noisy syndrome decoding". Physical Review A **110**, 012453 (2024).

[38] Anthony Leverrier and Rüdiger Urbanke. "Approximating optimal decoding of quantum LDPC codes with narrow frontiers" (2026). arXiv:2606.20513.

[39] Tim Chan. "Snowflake: A distributed streaming decoder". Quantum **10**, 2033 (2026).

[40] NVIDIA Corporation, CUDA-QX Development Team. "CUDA-Q QEC C++ API: Sliding window decoder". CUDA-QX documentation, version 0.7.0 (2026). Accessed 27 August 2026.

[41] Man-Duen Choi. "A schwarz inequality for positive linear maps on C^*-algebras". Illinois Journal of Mathematics **18**, 565 – 574 (1974).

[42] Caroline de Groot, Alex Turzillo, and Norbert Schuch. "Symmetry Protected Topological Order in Open Quantum Systems". Quantum **6**, 856 (2022).

[43] Héctor Bombín. "Gauge color codes: optimal transversal gates and gauge fixing in topological stabilizer codes". New Journal of Physics **17**, 083002 (2015).

A Further stabilizer preliminaries

This appendix records the Choi-operator facts behind Sec. 2.2, proves Lemma 2.2 through a stabilizer dilation, verifies that the elementary operations of Sec. 2.4 are stabilizer channels, and compares our stabilizer channels with the outcome-complete circuits of Ref. [19].

A.1 Stabilizer Choi operators and dilations

For a superoperator $\Phi : \mathcal{L}(\mathcal{H}_A) \to \mathcal{L}(\mathcal{H}_B)$, the Choi operator of Sec. 2.2 is $J(\Phi) := (\Phi \otimes \mathrm{id}_{A'})(|\Omega_A\rangle\langle\Omega_A|)$, where $|\Omega_A\rangle$ is the unnormalized maximally entangled vector between A and its reference copy A'. We call a nonzero Φ a *stabilizer superoperator* if $J(\Phi)$ is a stabilizer operator, or equivalently if $|J(\Phi)\rangle\!\rangle$ is proportional to a pure stabilizer state. An n_{in}-qubit to n_{out}-qubit stabilizer superoperator is therefore determined up to a scalar by $2(n_{\mathrm{in}} + n_{\mathrm{out}})$ independent Pauli stabilizer equations on $|J(\Phi)\rangle\!\rangle$, and after absorbing the computational-basis transposes arising from vectorization into the input Paulis each of them takes the four-slot form of Eq. (7). Complete positivity and trace preservation are additional conditions, equivalent respectively to $J(\Phi) \geq 0$ and $\mathrm{Tr}_B J(\Phi) = I_{A'}$. A completely positive, trace-non-increasing stabilizer superoperator is a *stabilizer map*, with the zero map included by convention, and a trace-preserving stabilizer map is a stabilizer channel in the sense of Sec. 2.2. Stabilizer maps include the selective projections $\rho \mapsto \Pi_S \rho \Pi_S$ and the stabilizer effects $\rho \mapsto \mathrm{Tr}(\Pi_S \rho)$, which need not be channels. Every stabilizer operator Q induces the stabilizer superoperator $\mathrm{Ad}_Q : \rho \mapsto Q\rho Q^\dagger$ with the single Kraus operator Q, and a *stabilizer isometry* is an isometry that is a stabilizer operator. Tensor products and nonzero serial compositions of stabilizer superoperators are again stabilizer superoperators, since they correspond to tensor products and stabilizer contractions of their vectorized Choi operators.

The rest of this subsection compares two descriptions of a stabilizer channel. The Choi description used in Sec. 2.2 is intrinsic, whereas a Stinespring dilation introduces a nonunique environment that records discarded information. For completely positive, trace-preserving maps, the two descriptions define the same class [30, Theorem 1, items (4)–(5)]; we include a self-contained proof below.

Proposition A.1 (Stabilizer dilation)**.** *Let $\Phi : \mathcal{L}(\mathcal{H}_A) \to \mathcal{L}(\mathcal{H}_B)$ be completely positive and trace preserving. Then $J(\Phi)$ is a stabilizer operator if and only if there exist an environment E and a stabilizer isometry*

$$V : \mathcal{H}_A \longrightarrow \mathcal{H}_B \otimes \mathcal{H}_E \tag{44}$$

such that

$$\Phi(\rho) = \mathrm{Tr}_E\left(V \rho V^\dagger\right). \tag{45}$$

Proof. Suppose first that V is a stabilizer isometry satisfying Eq. (45). Its vectorization $|V\rangle\!\rangle$ is a pure stabilizer state, and

$$J(\Phi) = \mathrm{Tr}_E\left(|V\rangle\!\rangle\langle\!\langle V|\right). \tag{46}$$

To see the form of this reduced operator, write the density operator of a normalized pure stabilizer state with stabilizer group S as $2^{-n}\sum_{s\in S} s$. The partial trace removes every term acting nontrivially on E, leaving a scalar multiple of the

projector onto the common eigenspace of the surviving stabilizers. Thus, $J(\Phi)$ is proportional to a stabilizer-subspace projector and is therefore a stabilizer operator.

Conversely, suppose that $J(\Phi)$ is a stabilizer operator. We first note that every positive stabilizer operator K is proportional to a stabilizer-subspace projector. Indeed, tracing one copy out of the pure stabilizer state $|K\rangle\!\rangle$ gives $KK^\dagger$, which by the preceding stabilizer-group argument is proportional to such a projector Π_S. Since $K \geq 0$, the positive square root is unique, so $K = \lambda\Pi_S$ for some $\lambda > 0$. Applying this observation to the positive Choi operator gives

$$J(\Phi) = \lambda\Pi_S. \tag{47}$$

Let C be a stabilizer encoding isometry whose image is the support of Π_S, and let E be a reference copy of its logical input. The encoded maximally entangled vector

$$|\Gamma\rangle := \sqrt{\lambda}(C \otimes I_E)|\Omega\rangle \tag{48}$$

is a stabilizer purification of $J(\Phi)$. After regrouping its systems as $(B \otimes E) \otimes A'$, write this vector as $|V\rangle\!\rangle$ for an operator $V : A \to B \otimes E$. Since $\mathrm{Tr}_{BE}(|V\rangle\!\rangle\langle\!\langle V|) = (V^\dagger V)^T$ and $\mathrm{Tr}_E(|V\rangle\!\rangle\langle\!\langle V|) = J(\Phi)$, trace preservation implies

$$(V^\dagger V)^T = \mathrm{Tr}_B\, J(\Phi) = I_{A'}, \tag{49}$$

and hence $V^\dagger V = I_A$. Therefore, V is a stabilizer isometry. Tracing out E recovers $J(\Phi)$, so injectivity of the Choi representation gives Eq. (45). □

Corollary A.2 (Pauli covariance)**.** *For every input Pauli R of a stabilizer channel Φ, there exists an output Pauli Q such that*

$$\Phi \circ \mathrm{Ad}_R = \mathrm{Ad}_Q \circ \Phi. \tag{50}$$

Proof. Choose the stabilizer isometry $V : A \to B \otimes E$ supplied by Proposition A.1. The input-reference marginal of $|V\rangle\!\rangle$ is maximally mixed because

$$\mathrm{Tr}_{BE}\left(|V\rangle\!\rangle\langle\!\langle V|\right) = (V^\dagger V)^T = I_{A'}. \tag{51}$$

Consequently, the projection of the phase-free stabilizer group of $|V\rangle\!\rangle$ onto the Pauli space of A' is surjective. Otherwise, nondegeneracy of the Pauli commutation form would give a nonidentity Pauli T on A' commuting with every projected stabilizer. Then $I_{BE} \otimes T$ would commute with the complete stabilizer group and hence belong to it, because the stabilizer group of a pure stabilizer state is maximal. This would make T a stabilizer of the reduced state on A', contradicting Eq. (51).

It follows that for every input Pauli R, there is a Pauli $\widetilde{W}$ on $B \otimes E$ such that, after absorbing a possible sign into $\widetilde{W}$,

$$(\widetilde{W} \otimes R^T)|V\rangle\!\rangle = |V\rangle\!\rangle. \tag{52}$$

The vectorization identity converts this equation into $VR = WV$, where $W := \widetilde{W}^\dagger$. Factor $W = Q \otimes S$ into Paulis on B and E. For every input operator ρ,

$$\begin{aligned}\Phi(R\rho R^\dagger) &= \mathrm{Tr}_E\left[(Q \otimes S)V\rho V^\dagger(Q^\dagger \otimes S^\dagger)\right] \\ &= Q\,\mathrm{Tr}_E(V\rho V^\dagger)Q^\dagger = Q\Phi(\rho)Q^\dagger,\end{aligned}$$

which proves Eq. (50). □

This proves Lemma 2.2; the dual statement is Theorem 1 of Ref. [30].

The trace-preserving assumption in Proposition A.1 is essential. A trace-nonincreasing stabilizer map may have a stabilizer Choi operator, but an isometry followed only by a partial trace is necessarily trace preserving. Selective stabilizer projections and effects therefore require postselection or a corresponding effect in their dilation description.

Computational-basis dephasing illustrates why the Choi condition must allow mixed stabilizer operators. With the unnormalized maximally entangled convention,

$$\begin{aligned}J(\mathcal{T}_Z) &= |00\rangle\langle 00| + |11\rangle\langle 11|, \\ |J(\mathcal{T}_Z)\rangle\!\rangle &= |0000\rangle + |1111\rangle.\end{aligned} \tag{53}$$

The Choi operator is mixed, while its vectorization is a pure GHZ stabilizer state. An equivalent stabilizer dilation is

$$V|z\rangle = |z\rangle_B|z\rangle_E, \qquad \mathcal{T}_Z(\rho) = \mathrm{Tr}_E(V\rho V^\dagger). \tag{54}$$

Thus, requiring the normalized Choi operator itself to be pure would exclude information-losing stabilizer channels such as dephasing.

A.2 Stabilizer and outcome-complete stabilizer circuits

Our stabilizer channels with declared classical ports are, by design, very similar to the objects

represented by the stabilizer circuits of Definition 2.3 in Ref. [19], which are built from zero-state and fair-coin allocations, Clifford unitaries, nondestructive Pauli measurements, Pauli corrections controlled by affine parities of earlier outcomes, and deallocation of wires promised to be in the zero state. Such a circuit retains every measurement outcome and every allocated random bit in its outcome vector, and we therefore call it *outcome-complete* in this subsection; the qualifier is ours rather than terminology of the reference. With the outcome vector represented as a declared classical output register, the channel of an outcome-complete stabilizer circuit has the form

$$\widehat{\Phi}(\rho) = \sum_o Q_o \rho Q_o^\dagger \otimes |o\rangle\langle o|, \tag{55}$$

where every conditional map has a single stabilizer Kraus operator Q_o; in the language of quantum instruments, every conditional map has Kraus rank at most one, so for a pure input the quantum output conditioned on the complete outcome vector is pure whenever it is nonzero. A stabilizer channel in the sense of Sec. 2.2 need not have this property, because it may discard a classical record or a quantum subsystem. Moreover, the stabilizer maps of Appendix A.1 may be trace non-increasing, whereas the instrument represented by an outcome-complete circuit is trace preserving; a selective stabilizer map corresponds to one branch of such an instrument. Below we verify that every elementary operation of Definition 2.3 is a stabilizer channel with the appropriate classical port declarations, and that conversely every stabilizer circuit built from these operations agrees with an outcome-complete one once the discarded records are accounted for.

We first record two basic constructions with classical interfaces. A deterministic affine map $f(x) = Ax + b$ between classical bit strings is represented by

$$\Phi_f(\rho) = \sum_{x \in \mathbb{F}_2^{m_{\text{in}}}} \langle x|\rho|x\rangle |Ax+b\rangle\langle Ax+b|. \tag{56}$$

Indeed, $|J(\Phi_f)\rangle\!\rangle$ is the uniform superposition over an affine binary subspace and is therefore a stabilizer state.

A nondestructive projective Z measurement is obtained by preparing a fresh ancilla in $|0\rangle$, applying a controlled-X from the measured qubit to the ancilla, and declaring the ancilla output classical:

$$\mathcal{M}_Z = (\mathrm{id} \otimes \mathcal{T}_Z) \circ \mathrm{Ad}_{\mathrm{CX}} \circ (\rho \mapsto \rho \otimes |0\rangle\langle 0|)\,. \tag{57}$$

The ancilla is a quantum wire during the controlled-X and becomes a classical carrier once $\mathcal{T}_Z$ is applied. Thus, $\mathcal{M}_Z$ is a stabilizer channel with a declared classical outcome port. The unencoder of Eq. (5) is likewise a stabilizer channel with a declared classical syndrome output, since it is the Clifford unitary $E^\dagger$ followed by $\mathcal{T}_Z$ on every qubit of the syndrome register.

We now check that every elementary operation of Definition 2.3 in Ref. [19] is represented by a stabilizer channel with the appropriate classical port declarations. The allocation of a qubit initialized in the zero state is

$$\mathcal{A}_0(1) := |0\rangle\langle 0|. \tag{58}$$

Its output may be declared either quantum or classical because $\mathcal{T}_Z \circ \mathcal{A}_0 = \mathcal{A}_0$. The two declarations respectively describe a qubit initialized in $|0\rangle$ and a deterministic classical bit initialized to zero, and the preparation of any other single-qubit stabilizer state is $\mathcal{A}_0$ followed by a Clifford unitary.

The allocation of a classical random bit distributed as a fair coin is

$$\mathcal{R}(1) := \frac{I_C}{2}. \tag{59}$$

The vectorization of its Choi operator is proportional to $|00\rangle + |11\rangle$, and its declared classical output satisfies $\mathcal{T}_Z \circ \mathcal{R} = \mathcal{R}$ so it can be interpreted as either classical or quantum.

Every Clifford unitary U, including every Pauli unitary, induces the stabilizer channel Ad_U without classical port declarations. Indeed,

$$J(\mathrm{Ad}_U) = |U\rangle\!\rangle\langle\!\langle U|, \tag{60}$$

and $|U\rangle\!\rangle$ is a stabilizer state.

For a Pauli observable P, its nondestructive measurement is the quantum-classical channel

$$\begin{aligned} \mathcal{M}_P(\rho) &:= \sum_{o \in \mathbb{F}_2} \Pi_o \rho \Pi_o \otimes |o\rangle\langle o|, \\ \Pi_o &:= \frac{I + (-1)^o P}{2}. \end{aligned} \tag{61}$$

Choosing a Clifford unitary that maps P to a single-qubit Z reduces this channel to the construction in Eq. (57), so $\mathcal{M}_P$ is a stabilizer channel whose explicitly recorded outcome bit is a declared classical output port.

The deallocation of a quantum or classical two-level wire is represented directly by the partial trace Tr_w. For a discarded wire with no other input, $J(\mathrm{Tr}_w) = I_{w'}$, whose vectorization is a Bell state; discarding one part of a larger input is its tensor product with an identity channel. Its input may be declared either quantum or classical because $\mathrm{Tr}_w \circ \mathcal{T}_Z = \mathrm{Tr}_w$. Definition 2.3 permits deallocation only when the wire is known to be in the zero state, whereas our discard channel is defined on arbitrary inputs. Tracing out a factor of a product state (which the promise guarantees) preserves the purity of the state, which is the invariant that outcome-complete stabilizer circuits maintain. However, the stabilizer channels of Sec. 2.2, and hence our stabilizer circuits, allow unconditional tracing out and dephasing of arbitrary subsystems and can therefore "forget" information. On the promised zero-state input, the discard channel agrees with the stabilizer effect $\rho \mapsto \langle 0|\rho|0\rangle$, which is a purity-preserving stabilizer map but is not trace preserving without the promise. A simple way to guarantee that a stabilizer circuit represents a trace-preserving stabilizer channel is to build it from trace-preserving elementary stabilizer channels only, as Definition 2.6 does.

Finally, let $c \in \mathbb{F}_2^m$ collect earlier classical outcomes, and let $p(c) := a^T c + b$ be an affine parity. The Pauli unitary P conditioned on this parity is the channel

$$\mathcal{C}_{P,p}(|c\rangle\langle c| \otimes \rho_c) := |c\rangle\langle c| \otimes P^{p(c)} \rho_c P^{p(c)}. \quad (62)$$

It is a stabilizer channel because affine classical control can be realized by Clifford operations on dephased classical controls and the quantum target. Tensoring and composing these stabilizer channels therefore maps every outcome-complete stabilizer circuit to a stabilizer circuit with the same retained outcome register.

Conversely, a stabilizer circuit built from these operations can be refined to an outcome-complete stabilizer circuit by retaining, as an explicit outcome, every record that it discards. For example, with $\Pi_o^Z := (I + (-1)^o Z)/2$, dephasing is a nondestructive measurement whose outcome is forgotten,

$$\mathcal{T}_Z(\rho) = \mathrm{Tr}_M \left[\sum_{o \in \mathbb{F}_2} \Pi_o^Z \rho \Pi_o^Z \otimes |o\rangle\langle o|_M \right], \quad (63)$$

and tracing out a wire is a destructive measurement of it in any Pauli basis whose result is forgotten. If M collects all records introduced in this way, the stabilizer channel and its outcome-complete refinement satisfy

$$\Phi = \mathrm{Tr}_M \circ \widehat{\Phi}. \quad (64)$$

Thus the two channels are equal once the record-discarding map Tr_M is included; in the terms of Definition 2.5, the two circuits are equivalent under the outcome map that keeps the retained outcomes and discards M. Together with the preceding paragraph, this translates each formalism into the other for the circuit class considered here; for a general stabilizer channel, Proposition A.1 supplies the corresponding dilation by a stabilizer isometry followed by a discard. This forgetting is more general than the instrument equivalence of Ref. [19], which only merges outcome branches whose conditional maps are proportional; the two branches of a dephasing measurement are not proportional and cannot be merged in that way.

For a finite unitary group G, the *twirling channel* is $\mathcal{T}_G(\rho) := |G|^{-1} \sum_{U \in G} U \rho U^\dagger$, so that the dephasing channel $\mathcal{T}_Z$ of Sec. 2.2 is the twirl over $\langle Z \rangle$. For a qubit register A, let $\mathcal{P}(A)$ denote its Pauli group; the twirl over $\mathcal{P}(A)$ is the *fully depolarizing channel*

$$\mathcal{T}_{\mathcal{P}(A)}(\rho) = \mathrm{Tr}(\rho) \frac{I_A}{2^{|A|}}, \quad (65)$$

which is a stabilizer channel because $J(\mathcal{T}_{\mathcal{P}(A)}) = (I_A/2^{|A|}) \otimes I_{A'}$, whose vectorization is proportional to a product of Bell states. Replacing dephasing by full depolarization in Eq. (9) gives stronger interface conditions: rather than discarding only coherence, they discard all information carried by the selected port. For an input register A_I and an output register B_R of Φ, with the identity action on the remaining ports understood, we call A_I an *irrelevant input* and B_R an *independently random output* when, respectively,

$$\Phi \circ \mathcal{T}_{\mathcal{P}(A_I)} = \Phi, \qquad \mathcal{T}_{\mathcal{P}(B_R)} \circ \Phi = \Phi. \quad (66)$$

The first condition states that Φ does not depend on the input on A_I, and the second that B_R is maximally mixed and uncorrelated with the input and with all other outputs, that is, $\Phi(\rho) = (I_{B_R}/2^{|B_R|}) \otimes \widetilde{\Phi}(\rho)$ for a channel $\widetilde{\Phi}$ onto the remaining outputs. Each condition is generated

within $\mathsf{Stab}(\Phi)$ by the X- and Z-conjugation versions of the corresponding equation in Eq. (10).

An independently random output can therefore be discarded and regenerated by allocating fresh fair coins, Eq. (59), without changing the channel. The condition is stronger than mere randomness: an outcome that is uniformly random but correlated with another output is not independently random, because applying $\mathcal{T}_{\mathcal{P}(B_R)}$ would destroy the correlation and change the channel. For example, a fair coin allocated as in Definition 2.3 is an independently random output, whereas a physical wire that is randomized by measuring it, with the result retained in a distinct register M, becomes independently random only after M is forgotten.

B Further mathematical preliminaries

Here we provide a proof of the finite-dimensional multiplicative-domain theorem; the general version is due to Choi [41].

Theorem B.1 (Multiplicative domain in finite dimensions)**.** *Let*

$$\Psi : \mathrm{L}(\mathcal{Y}) \longrightarrow \mathrm{L}(\mathcal{X})$$

be a unital completely positive map, and let $A \in \mathrm{L}(\mathcal{Y})$*. Suppose that*

$$\Psi(A^\dagger A) = \Psi(A)^\dagger \Psi(A)$$

and

$$\Psi(AA^\dagger) = \Psi(A)\Psi(A)^\dagger.$$

Then, for every $B \in \mathrm{L}(\mathcal{Y})$*,*

$$\Psi(AB) = \Psi(A)\Psi(B)$$

and

$$\Psi(BA) = \Psi(B)\Psi(A).$$

Proof. Choose a Kraus representation

$$\Psi(B) = \sum_j K_j^\dagger B K_j, \qquad \sum_j K_j^\dagger K_j = I.$$

Because Ψ is completely positive, it is Hermiticity preserving, and hence

$$\Psi(A^\dagger) = \Psi(A)^\dagger.$$

Consider

$$\sum_j (AK_j - K_j\Psi(A))^\dagger (AK_j - K_j\Psi(A)).$$

Expanding this expression gives

$$\begin{aligned}
&\sum_j K_j^\dagger A^\dagger A K_j - \sum_j K_j^\dagger A^\dagger K_j \Psi(A) \\
&\quad - \Psi(A)^\dagger \sum_j K_j^\dagger A K_j \\
&\quad + \Psi(A)^\dagger \left(\sum_j K_j^\dagger K_j \right) \Psi(A) \\
&= \Psi(A^\dagger A) - \Psi(A)^\dagger \Psi(A).
\end{aligned}$$

By assumption, the right-hand side is zero. Every summand on the left-hand side is positive semidefinite, so each summand vanishes. Therefore,

$$AK_j = K_j \Psi(A)$$

for every j. Consequently,

$$\begin{aligned}
\Psi(BA) &= \sum_j K_j^\dagger B A K_j \\
&= \sum_j K_j^\dagger B K_j \Psi(A) \\
&= \Psi(B)\Psi(A).
\end{aligned}$$

Apply the same argument to $A^\dagger$. The second assumed equality may be written as

$$\Psi((A^\dagger)^\dagger A^\dagger) = \Psi(A^\dagger)^\dagger \Psi(A^\dagger).$$

It follows that

$$A^\dagger K_j = K_j \Psi(A^\dagger) = K_j \Psi(A)^\dagger.$$

Taking adjoints gives

$$K_j^\dagger A = \Psi(A) K_j^\dagger.$$

Hence

$$\begin{aligned}
\Psi(AB) &= \sum_j K_j^\dagger A B K_j \\
&= \Psi(A) \sum_j K_j^\dagger B K_j \\
&= \Psi(A)\Psi(B).
\end{aligned}$$

□

The following corollary shows how constraints on a dual channel translate into constraints on the primal channel. This is similar to the argument around Eqs. (6.17)–(6.18) in Ref. [42].

Corollary B.2. *Let*

$$\Phi : \mathrm{L}(\mathcal{X}) \longrightarrow \mathrm{L}(\mathcal{Y})$$

be completely positive and trace preserving, and let $P \in \mathrm{L}(\mathcal{Y})$ *and* $Q \in \mathrm{L}(\mathcal{X})$ *be Hermitian unitaries. Then the following statements are equivalent:*

1. $\Phi^\dagger(P) = Q$.
2. *For every* $X \in \mathrm{L}(\mathcal{X})$, $P\Phi(QX) = \Phi(X)$.

Proof. Set $\Psi = \Phi^\dagger$, and suppose first that

$$\Psi(P) = Q.$$

Since Φ is completely positive and trace preserving, Ψ is unital and completely positive. Because P and Q are unitary,

$$\Psi(P^\dagger P) = \Psi(I) = I = Q^\dagger Q = \Psi(P)^\dagger \Psi(P),$$

and

$$\Psi(PP^\dagger) = I = QQ^\dagger = \Psi(P)\Psi(P)^\dagger.$$

The multiplicative-domain theorem B.1 therefore gives

$$\Psi(PB) = Q\Psi(B)$$

for every $B \in \mathrm{L}(\mathcal{Y})$.

Using the Hilbert–Schmidt inner product

$$\langle A, B\rangle = \mathrm{Tr}(AB^\dagger),$$

we obtain, for arbitrary $B \in \mathrm{L}(\mathcal{Y})$,

$$\begin{aligned}\langle \Phi(QX), B\rangle &= \langle QX, \Psi(B)\rangle \\ &= \langle X, Q\Psi(B)\rangle \\ &= \langle X, \Psi(PB)\rangle \\ &= \langle \Phi(X), PB\rangle \\ &= \langle P\Phi(X), B\rangle.\end{aligned}$$

Therefore,

$$\Phi(QX) = P\Phi(X).$$

Multiplying on the left by P, and using $P^2 = I$, gives

$$P\Phi(QX) = \Phi(X).$$

Conversely, suppose that

$$P\Phi(QX) = \Phi(X)$$

for every $X \in \mathrm{L}(\mathcal{X})$. Multiplying on the left by P gives

$$\Phi(QX) = P\Phi(X).$$

Taking the trace and using that Φ is trace preserving, we obtain

$$\mathrm{Tr}(QX) = \mathrm{Tr}(\Phi(QX)) = \mathrm{Tr}(P\Phi(X)).$$

By the definition of the dual map,

$$\begin{aligned}\mathrm{Tr}(P\Phi(X)) &= \langle \Phi(X), P\rangle = \langle X, \Phi^\dagger(P)\rangle \\ &= \mathrm{Tr}(X\Phi^\dagger(P)^\dagger).\end{aligned}$$

Thus,

$$\mathrm{Tr}(XQ) = \mathrm{Tr}(X\Phi^\dagger(P)^\dagger)$$

for every X. By nondegeneracy of the trace pairing,

$$\Phi^\dagger(P)^\dagger = Q.$$

Taking adjoints and using $Q^\dagger = Q$ yields

$$\Phi^\dagger(P) = Q.$$

□

C Repeated subsystem and Floquet syndrome extraction

Section 5.2 considers the particularly simple case in which every cycle measures the same stabilizer generators that define both of its code boundaries. This appendix retains the requirement that each cycle prepares its declared output code, up to an outcome-dependent Pauli frame, while separating the Pauli group measured from the input from the stabilizer group prepared at the output. This separation accommodates subsystem-code gauge fixing [43] and Floquet syndrome extraction [32] without changing the extended detector error model formalism (Def. 4.4).

C.1 Measured and prepared Pauli groups

Consider a sequence of stabilizer channels

$$\Phi_t : \mathcal{L}(\mathcal{H}_A) \longrightarrow \mathcal{L}(\mathcal{H}_A \otimes \mathcal{H}_{\mathsf{M}_t}), \tag{67}$$

where the quantum output of Φ_t is the quantum input of Φ_{t+1}, and M_t labels the outcomes declared by cycle t. The channels and their code presentations may depend periodically on t. For example, the phases of a Floquet protocol may be represented separately, or one full period may be packaged as a single cycle.

The boundary-constraint group $R_{\mathsf{M}_t}(\Phi_t)$ of Def. 2.3 distinguishes two phase-free Pauli groups,

$$\mathcal{G}_t^{\mathrm{meas}} := \left\{ \bar{P} \,\middle|\, \exists a, b : \ (\bar{P}, I, a, b) \in R_{\mathsf{M}_t}(\Phi_t) \right\}, \tag{68}$$

$$\mathcal{S}_t := \left\{ \bar{Q} \,\middle|\, \exists a, b : \ (I, \bar{Q}, a, b) \in R_{\mathsf{M}_t}(\Phi_t) \right\}. \tag{69}$$

The group $\mathcal{G}_t^{\mathrm{meas}}$ consists of input Paulis measured by the cycle, whereas $\mathcal{S}_t$ consists of output Paulis prepared by the cycle; they are the measured group and the output stabilizer group of Φ_t in the terminology of Sec. 2.2. Their signed versions $\mathcal{G}_t^{\mathrm{meas}}(m_t)$ and $\mathcal{S}_t(m_t)$ attach the sign $(-1)^{a^T m_t + b}$ supplied by the corresponding boundary constraint.

Let S_t^{code} be the phase-free stabilizer group of the output code declared after cycle t. The preparation assumption used here is

$$S_t^{\mathrm{code}} \leq \mathcal{S}_t. \tag{70}$$

Equivalently, for every $\bar{S} \in S_t^{\mathrm{code}}$, there are a_S, b_S such that

$$(\bar{S} \otimes Z_{\mathsf{M}_t}(a_S, b_S))\Phi_t(\rho) = \Phi_t(\rho) \qquad \text{for every } \rho. \tag{71}$$

Thus the sign of every declared output stabilizer is determined by outcomes of the present cycle, without invoking an input stabilizer. This condition is not equivalent to measuring all input-code stabilizers: measurement and preparation are different projections of the boundary-constraint group.

Relations with nontrivial Pauli components on both the input and output describe transported logical correlations and, for a subsystem presentation, may also retain gauge correlations. These input–output relations determine the fault-free transformation of the remaining boundary information.

C.2 Detector closure between unequal groups

The stabilizer relations that close immediately across the boundary preceding cycle t belong to

$$\mathcal{I}_t := \mathcal{S}_{t-1} \cap \mathcal{G}_t^{\mathrm{meas}}, \tag{72}$$

or to the corresponding subgroups selected by the adjacent code presentations. For $\bar{H} \in \mathcal{I}_t$, choose constraints

$$(I, \bar{H}, a^{\mathrm{S}}, b^{\mathrm{S}}) \in R_{\mathsf{M}_{t-1}}(\Phi_{t-1}), \tag{73}$$

$$(\bar{H}, I, a^{\mathrm{M}}, b^{\mathrm{M}}) \in R_{\mathsf{M}_t}(\Phi_t). \tag{74}$$

Their composition gives the detector relation

$$d_{t,\bar{H}}(m_{t-1}, m_t) = (a^{\mathrm{S}})^T m_{t-1} + (a^{\mathrm{M}})^T m_t + b^{\mathrm{S}} + b^{\mathrm{M}} = 0. \tag{75}$$

for every fault-free realization. By contrast, an element of $\mathcal{S}_{t-1} \setminus \mathcal{G}_t^{\mathrm{meas}}$ need not yield a detector that closes in a later cycle. It may instead be randomized when cycle t measures an anticommuting Pauli, as occurs during gauge fixing. Any longer-range detector must follow from a boundary constraint of the composed channel rather than from an assumption that such a stabilizer persists.

The cycle may also have intrinsic detectors supported entirely on its own outcomes. The profile can present intrinsic and boundary-closing detectors together by assigning a zero preceding-boundary component to each intrinsic row.

C.3 Outcome-conditioned boundary coordinates

Intrinsically random outcomes can select the signs of the prepared stabilizers and a known logical or gauge Pauli frame. Consequently, the absolute output state is not generally a deterministic function of the preceding boundary state. Let $\widehat{z}_t(m_{\leq t})$ denote the ideal boundary coordinates obtained by propagating the preceding ideal branch through the code and Pauli-frame assignments declared for the outcome history $m_{\leq t}$. Here the Pauli-frame assignment is the outcome-dependent sign data (a, b) of the output-stabilizer and transported-logical boundary constraints of Def. 2.3, that is, the parities $a^T m_t + b$ that fix the signs of the prepared stabilizers and of the transported logical operators. If z_t denotes the actual boundary coordinates in the same binary presentation, define the residual boundary error coordinates by

$$\xi_t := z_t + \widehat{z}_t(m_{\leq t}). \tag{76}$$

Thus $\xi_t = 0$ on every ideal outcome branch when the input boundary error is zero. The syndrome and logical variables in Sec. 5.2 are components of this outcome-conditioned residual, rather than absolute stabilizer signs or an absolute logical

frame. Any random gauge label that can affect a later cycle must likewise be retained by the profile as virtual boundary information instead of being averaged out.

Let f_t be the fault configuration of cycle t (Sec. 2.4), and let v_t be the output virtual-syndrome *flip* in the presentation of $\mathcal{S}_t$. Let u_t collect the current-cycle parity flips appearing in the chosen detector rows, including measured-input parities for $\mathcal{I}_t$ and intrinsic detector parities. In the recurrence below, A_t, B_t, C_t, D_t, E_t, P_t, and Q_t denote blocks of the extended detector error model of cycle t (Def. 4.4); they are unrelated to the matrices A_{M}, A_{D}, A_{O} of Sec. 2.5 and to the encoder E of Sec. 2.1. The most general linear recurrence needed below has the form

$$\xi_t = A_t\xi_{t-1} + B_t f_t, \tag{77}$$
$$u_t = Q_t\xi_{t-1} + D_t f_t, \tag{78}$$
$$v_t = C_t\xi_{t-1} + E_t f_t, \tag{79}$$
$$d_t = u_t + P_t v_{t-1} = D_t f_t + Q_t\xi_{t-1} + P_t v_{t-1}. \tag{80}$$

Here P_t expresses the preceding prepared-stabilizer presentation in the detector-row presentation; its rows vanish for detectors intrinsic to cycle t. The matrices Q_t, D_t describe the measurement side of the detector relations, whereas C_t, E_t describe the independently chosen output-stabilizer presentation. Unlike the specialization in Sec. 5.2, these pairs of matrices need not agree. Known affine offsets have disappeared because Eqs. (77)–(80) describe flips relative to the outcome-conditioned ideal branch.

Define the ordered product

$$A_{t:j} := A_t A_{t-1}\cdots A_j, \tag{81}$$

with an empty product equal to the identity. Iteration of the boundary recurrence gives

$$\xi_t = A_{t:1}\xi_0 + \sum_{j=1}^{t} A_{t:j+1}B_j f_j, \tag{82}$$

and hence

$$v_t = C_t A_{t-1:1}\xi_0 + E_t f_t + \sum_{j=1}^{t-1} C_t A_{t-1:j+1}B_j f_j. \tag{83}$$

Substitution of the preceding-cycle recurrences into Eq. (80) gives, for $t \geq 2$,

$$\begin{aligned} d_t = {} & D_t f_t + (Q_t B_{t-1} + P_t E_{t-1}) f_{t-1} \\ & + (Q_t A_{t-1} + P_t C_{t-1})\xi_{t-2}. \end{aligned} \tag{84}$$

A detector profile with one cycle of boundary memory satisfies the compatibility condition

$$Q_t A_{t-1} + P_t C_{t-1} = 0. \tag{85}$$

This condition states that a boundary error present at the input of cycle $t-1$ contributes identically to the two sides of each closure relation and therefore cancels. Under this condition,

$$d_t = D_t f_t + (Q_t B_{t-1} + P_t E_{t-1}) f_{t-1}, \qquad t \geq 2. \tag{86}$$

Equation (85), rather than equality of $\mathcal{G}_t^{\mathrm{meas}}$ and $\mathcal{S}_t$, is the algebraic condition responsible for the block lower bidiagonal structure of the detector error model, cf. Eq. (33) and the detector span of Sec. 7. If a chosen elementary Floquet phase has a longer detector window, one may either retain the corresponding additional virtual boundary data or group a full period into one syndrome extraction cycle.

C.4 Reduction to the code-preserving example

The equations in Sec. 5.2 are recovered when

$$\xi_t = \begin{bmatrix} s_t \\ l_t \end{bmatrix}, \qquad A_t = I, \qquad B_t = \begin{bmatrix} S_F \\ L_F \end{bmatrix}, \tag{87}$$

and

$$\begin{aligned} Q_t = C_t = \begin{bmatrix} I & 0 \end{bmatrix}, \qquad & P_t = I, \\ D_t = E_t = D_F. & \end{aligned} \tag{88}$$

Equations (80) and (79) then become Eqs. (30) and (29), respectively. Moreover, Eq. (85) reduces to $I + I = 0$, and Eq. (86) becomes Eq. (33).

For a protocol with period p, grouping its phases into a full cycle gives the boundary matrices

$$A_{\mathrm{cyc}} := A_p A_{p-1}\cdots A_1, \tag{89}$$
$$B_{\mathrm{cyc}} := \begin{bmatrix} A_p\cdots A_2 B_1 & \cdots & A_p B_{p-1} & B_p \end{bmatrix}. \tag{90}$$

D Symbolic verification of the closed forms

Listing below verifies, with the SymPy computer-algebra system, the base case $k = 1$ and the induction step $k - 1 \to k$ of the closed forms in Eqs. (31)–(33), starting from the recurrences (27)–(30). The matrices D_F, S_F, L_F and all bitvectors are symbolic and coefficients are reduced modulo 2, so the single induction-step check covers every $k \geq 2$; the base case additionally records $d_1 = D_F f_1 + s_0 + v_0$, which is Eq. (30) at $k = 1$.

```
"""Prove the closed forms of the recurrences by induction.

RECURRENCE (sections/05_symbolic_construction.tex):
    d_k = D_F f_k + s_{k-1} + v_{k-1}      (eq:detector-flip)
    s_k = s_{k-1} + S_F f_k                (eq:syndrome-bpe)
    l_k = l_{k-1} + L_F f_k                (eq:logical-effect-bpe)
    v_k = D_F f_k + s_{k-1}                (eq:virtual-syndrome-flip)

CLOSED FORMS:
    s_k = s_0 + S_F F_k
    l_k = l_0 + L_F F_k
    v_k = D_F f_k + s_0 + S_F F_{k-1}
    d_1 = D_F f_1 + s_0 + v_0
    d_k = D_F f_k + (D_F + S_F) f_{k-1}    (k >= 2:)
where F_k = f_1 + ... + f_k.
"""

from sympy import Add, MatrixSymbol, Mul, Symbol, ZeroMatrix

N_F, N_G, N_L = (Symbol(x, integer=True, positive=True)
                 for x in ('N_F', 'N_G', 'N_L'))
D_F = MatrixSymbol('D_F', N_G, N_F)        # faults -> detectors
S_F = MatrixSymbol('S_F', N_G, N_F)        # faults -> syndrome
L_F = MatrixSymbol('L_F', N_L, N_F)        # faults -> logical effect
ZG, ZL, ZF = ZeroMatrix(N_G, 1), ZeroMatrix(N_L, 1), ZeroMatrix(N_F, 1)

s_0 = MatrixSymbol('s_0', N_G, 1)
l_0 = MatrixSymbol('l_0', N_L, 1)
v_0 = MatrixSymbol('v_0', N_G, 1)


def mod2(expr):
    """Reduce integer coefficients of a matrix expression mod 2."""
    expr = expr.doit().expand()
    terms = []
    for t in (expr.args if isinstance(expr, Add) else (expr,)):
        head, *rest = t.args if isinstance(t, Mul) else (None,)
        if not (head is not None and head.is_Integer):
            terms.append(t)
        elif int(head) % 2:
            terms.append(Mul(*rest))
    return Add(*terms) if terms else ZeroMatrix(expr.rows, expr.cols)


def step(s, l, v, f_k):
    """One step of the RECURRENCE: state at k-1 and f_k -> state at k, d_k."""
    d_k = D_F * f_k + s + v
    v_k = D_F * f_k + s
    s_k = s + S_F * f_k
    l_k = l + L_F * f_k
    return s_k, l_k, v_k, d_k


def check(name, got, want, zero):
    assert mod2(got - want) == zero, f'{name} does not match'


# --- base case: k = 1, state at k-1 = 0 is the initial state, F_0 = 0 -------
f_1 = MatrixSymbol('f_1', N_F, 1)
s_1, l_1, v_1, d_1 = step(s_0, l_0, v_0, f_1)

check('s_1', s_1, s_0 + S_F * f_1, ZG)                  # F_1 = f_1
check('l_1', l_1, l_0 + L_F * f_1, ZL)
check('v_1', v_1, D_F * f_1 + s_0 + S_F * ZF, ZG)       # F_0 = 0
check('d_1', d_1, D_F * f_1 + s_0 + v_0, ZG)

# --- induction step: assume the closed forms at k-1, derive them at k -------
# k is symbolic: f_km1, f_k and the accumulator F_km2 = f_1 + ... + f_{k-2}
# are unconstrained, so this single check covers every k >= 2 at once.
f_km1 = MatrixSymbol('f_{k-1}', N_F, 1)
f_k = MatrixSymbol('f_k', N_F, 1)
F_km2 = MatrixSymbol('F_{k-2}', N_F, 1)
F_km1 = F_km2 + f_km1                                   # F_{k-1} = F_{k-2} + f_{k-1}
F_k = F_km1 + f_k                                       # F_k     = F_{k-1} + f_k

# induction hypothesis: the closed forms hold at k-1 (with k-1 >= 1)
s_km1 = s_0 + S_F * F_km1
l_km1 = l_0 + L_F * F_km1
v_km1 = D_F * f_km1 + s_0 + S_F * F_km2

s_k, l_k, v_k, d_k = step(s_km1, l_km1, v_km1, f_k)

check('s_k', s_k, s_0 + S_F * F_k, ZG)
check('l_k', l_k, l_0 + L_F * F_k, ZL)
check('v_k', v_k, D_F * f_k + s_0 + S_F * F_km1, ZG)
check('d_k', d_k, D_F * f_k + (D_F + S_F) * f_km1, ZG)

print('base case k = 1 and induction step k-1 -> k verified: '
      'the closed forms hold for all k >= 1')
```