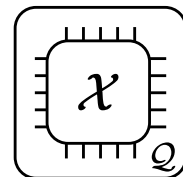
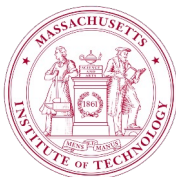


# Extractors: QLDPC Architectures for Efficient Pauli-Based Computation

Zhiyang He (Sunny), Alexander Cowtan, Dominic Williamson, Theodore Yoder



I. Motivation: A QLDPC-Based Quantum Computer

II. Code Surgery and Extractors

III. Extractor Architecture and Compilation

IV. Building an Extractor with Graph Theory

V. Discussions and Outlooks

# The Promise of QLDPC Codes

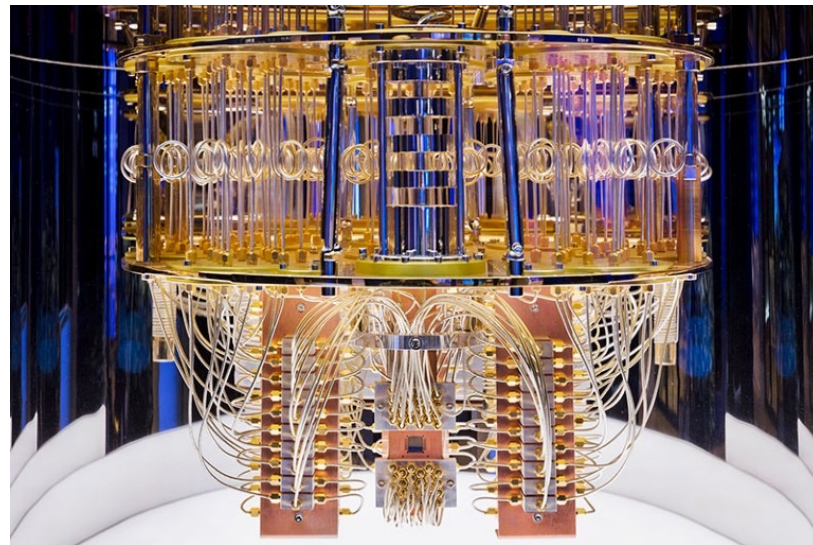
Surface code is **the leading candidate** for building a large-scale, fault tolerant quantum computer.

**Amazing properties:** high threshold, 2D connectivity, fast decoding, transversal gates, lattice surgery...

**Challenge:** Significant asymptotic space overhead, ~1000x for factoring.

Quantum LDPC codes promise to implement fault-tolerant computation with  **$O(1)$  space overhead**.

- **At what scale can we fulfill this promise to gain a practical advantage?**



# Fast Progress in QLDPC Memory

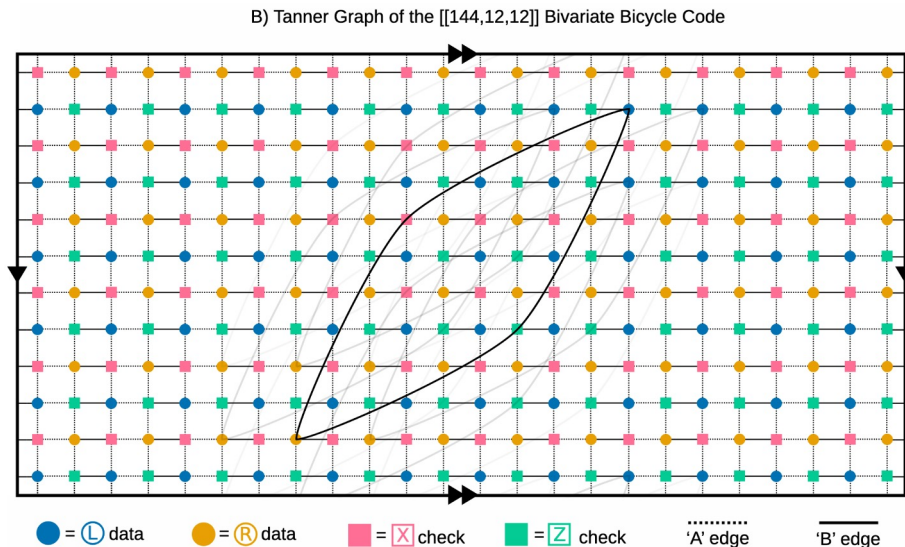
**Quantum Low-Density Parity-Check (LDPC) Codes:**  
stabilizers of  $O(1)$  weight, qubits in  $O(1)$  stabilizers.  
Better encoding rate than surface code!

Recent constructions:

- Bivariate Bicycle code  $[144, 12, 12]^*$
- Hypergraph product code  $[2500, 100, 12]^{**}$
- Lifted product code  $[544, 80, \leq 12]^{**}$

Surface code:  $[265, 1, 12]$ .

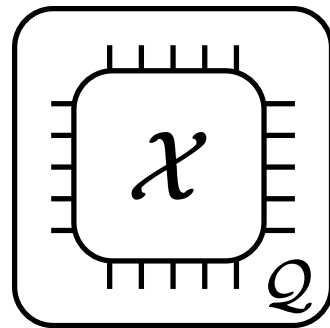
**Challenges:** Decoding algorithm & threshold, hardware, and logical computation.



# Extractor Architecture for QLDPC Computation

In this work, we present a solution to the QLDPC computation challenge: **Extractors**. Our solution has a few distinctive features:

1. **Any** quantum code can be augmented by an extractor system to become a computational block. I.e., **extractors augment memories into processors**.
2. Given **any** magic state factory, can **implement universal quantum circuits via parallelized logical operations**.
3. **Highly optimizable**, practical space and time overheads.
4. Can be implemented with **fixed, constant degree connectivity** (having movable qubits is certainly helpful but not necessary).



An Extractor-augmented computational (EAC) block.

I. Motivation: A QLDPC-Based Quantum Computer

II. Code Surgery and Extractors

III. Extractor Architecture and Compilation

IV. Building an Extractor with Graph Theory

V. Discussions and Outlooks

# Universal Computation via Logical Measurements

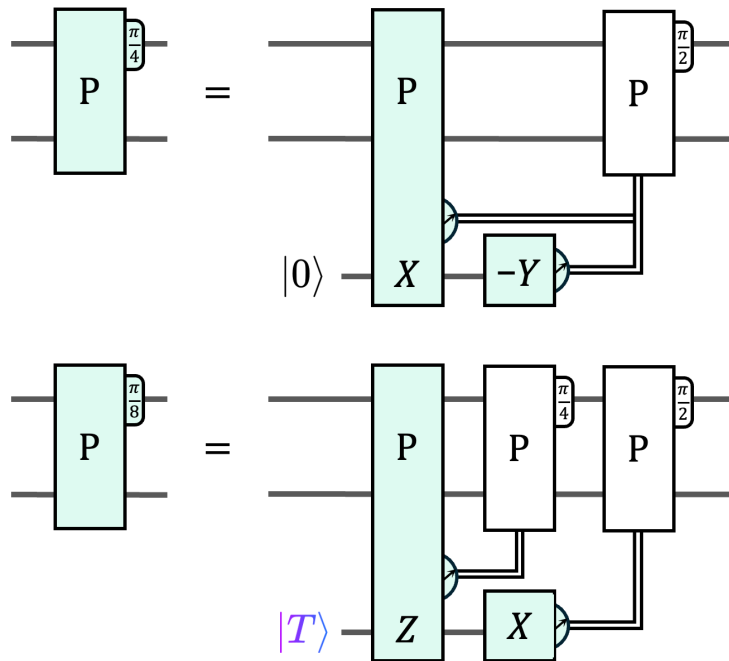
A Clifford + T circuit can be written in terms of Pauli rotations, where:

- Pauli gates  $\rightarrow$  Pauli  $\pi/2$  rotations,
- Clifford gates  $\rightarrow$  Pauli  $\pi/4$  rotations,
- T gates  $\rightarrow$  Pauli  $\pi/8$  rotations.

Pauli rotations can be implemented with Pauli measurements.

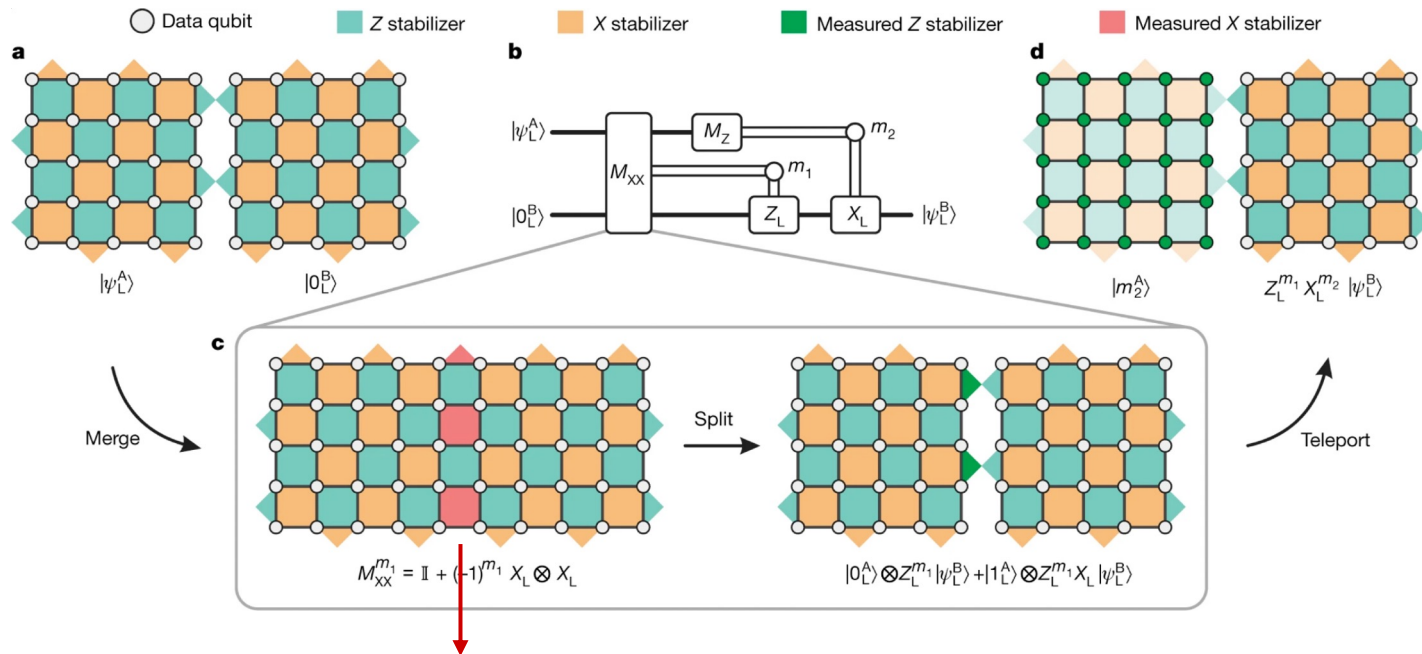
**Pauli-based computation:** Pauli measurements + magic states = universal computation.

- **Fault-tolerant measurements + magic state factory = universal FT computation!**



# Surface Code Lattice Surgery

Logical measurements on surface codes: **lattice surgery**, [Horsman et al, 1111.4022].



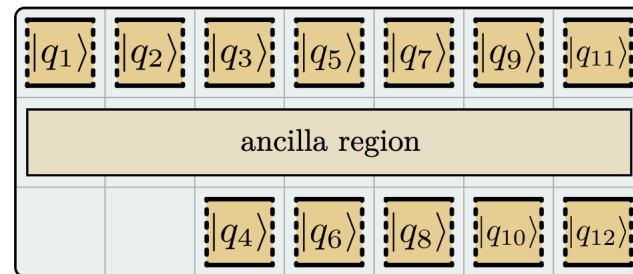
Product of **red** X-checks =  $X_L \otimes X_L$  – **obtain logical measurement result by measuring new stabilizers.**



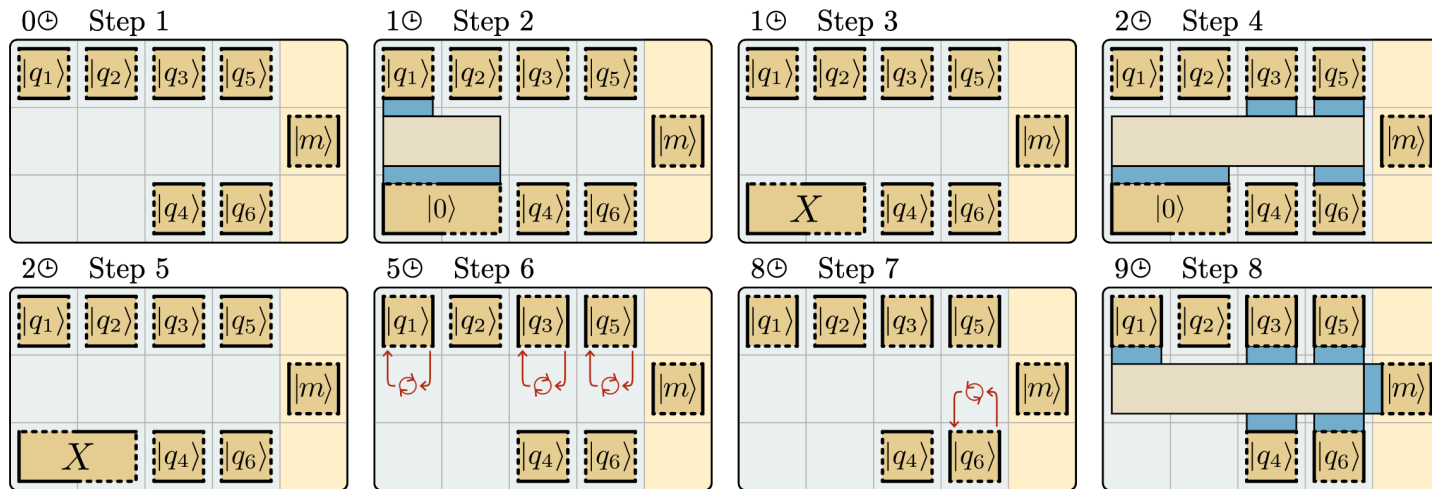
# Surface Code Architecture Based on Lattice Surgery

## Game of Surface Code [Litinski 1808.02892]

- Allocate surface code patches on a 2D plane into **memory**, **ancilla**, and **distillation** regions.
- **Any Pauli measurements can be performed** in 2D local connectivity with different protocols and ancilla patches.



(c)  $(Y \otimes 1 \otimes Y \otimes Z \otimes Y \otimes Y)_{\pi/8}$  rotation in  $9\ominus$

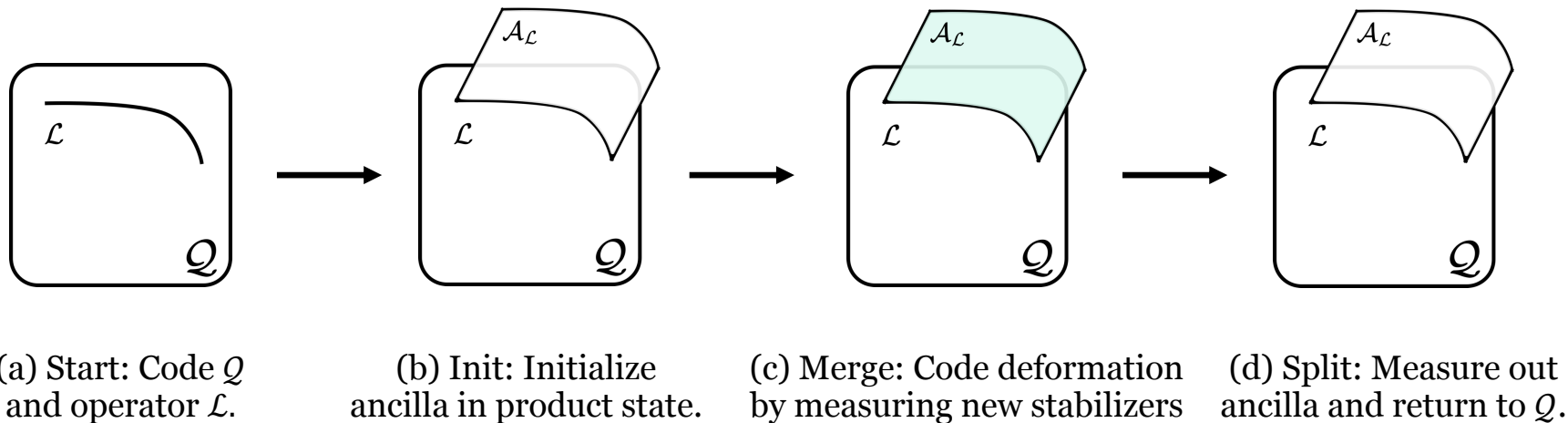


# QLDPC Code Surgery

First proposed by [Cohen et al., 2110.10794], > 10 papers on surgery in the past year.

➤ [Section 3.2 of the present work \[2503.10390\]](#) is a 1.5-page review.

High level description: for a quantum LDPC code  $Q$ , **for every logical operator  $\mathcal{L}$ , can construct ancilla system  $\mathcal{A}_{\mathcal{L}}$**  such that  $Q$  augmented by  $\mathcal{A}_{\mathcal{L}}$  can be used to measure  $\mathcal{L}$ .

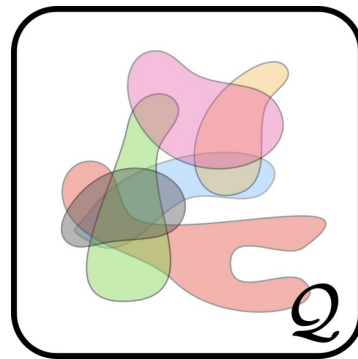


# Challenge: Compact Memory Has Many Operators

Prior works: for every logical operator  $\mathcal{L}$ , construct an ancilla system for measurement.

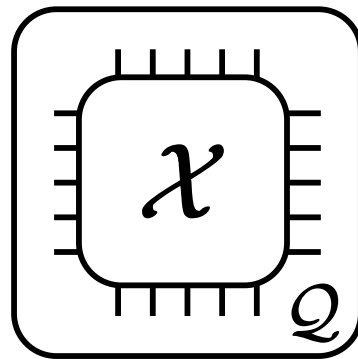
**Challenge:** High-rate codes have many operators, and they overlap.

- Building many ancilla systems will quickly blow up space and connectivity overhead.



**Extractors:** one ancilla system  $\mathcal{X}$ , can measure any logical operator.

- For any code of  $n$  qubits, can built LDPC extractor of size  $\tilde{O}(n)$ .
- In practice, expect space overhead to be a small constant. E.g., 103-qubit extractor for  $[144, 12, 12]$  gross code. [2407.18393]
- Any operator can be measured with  $O(d)$  syndrome rounds.
- Built using tools developed in [2407.18393], [2410.02213], and [2410.03628].



# Modularity: Bridges and Adapters

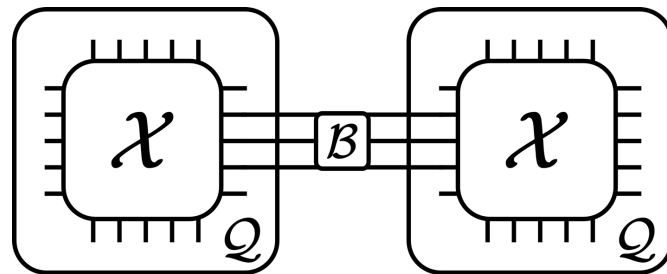
For a full PBC architecture, we need logical measurements on magic states and multiple QLDPC blocks. **Can we design our quantum computer modularly?**

**Bridge/Adapter:** primitive developed in [2407.18393] and [2410.03628].

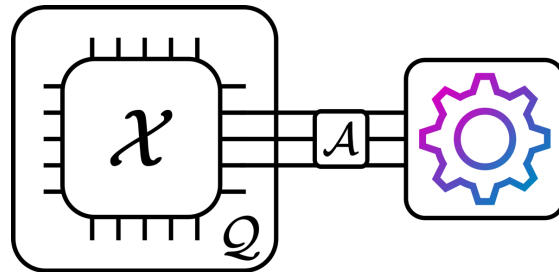
- LDPC Ancilla system that can connect two surgery ancillas together from **arbitrary codes**. Size  $O(d)$ .
- **Enables Pauli measurements across connected blocks.**

Two names for the same system:

- If it connects blocks of the same code, we call it **bridge**.
- If it connects blocks of different codes, we call it **adapter**.



Two EAC blocks joined by a **bridge**  $B$ .



An EAC blocks connected to a source of magic states by an **adapter**  $A$ .

I. Motivation: A QLDPC-Based Quantum Computer

II. Code Surgery and Extractors

**III. Extractor Architecture and Compilation**

IV. Building an Extractor with Graph Theory

V. Discussions and Outlooks

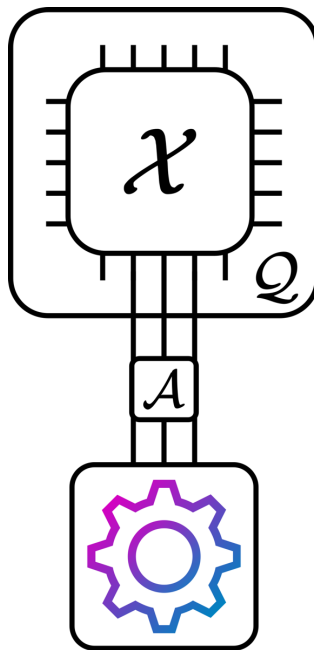
# Extractor Architecture: MWE

Let's start with a minimal working example (MWE) of extractor architectures.

A  $[n, k, d]$  code  $Q$ , augmented by an extractor  $\mathcal{X}$ . This is an EAC block with  $k-1$  computational qubits.

The extractor  $\mathcal{X}$  is connected to the factory by an adapter.

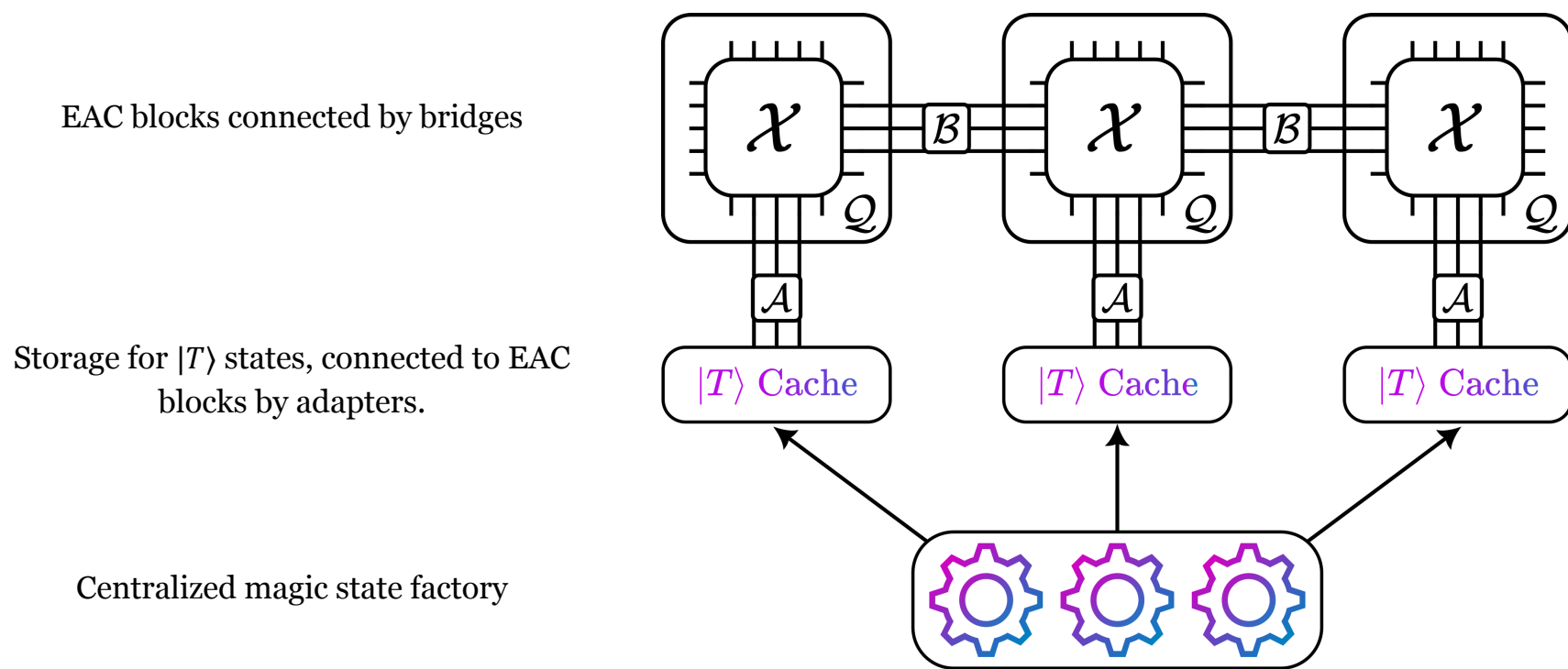
An arbitrary  $|T\rangle$  state factory. E.g., consider a magic state cultivation patch of surface code.



## Features & Comments

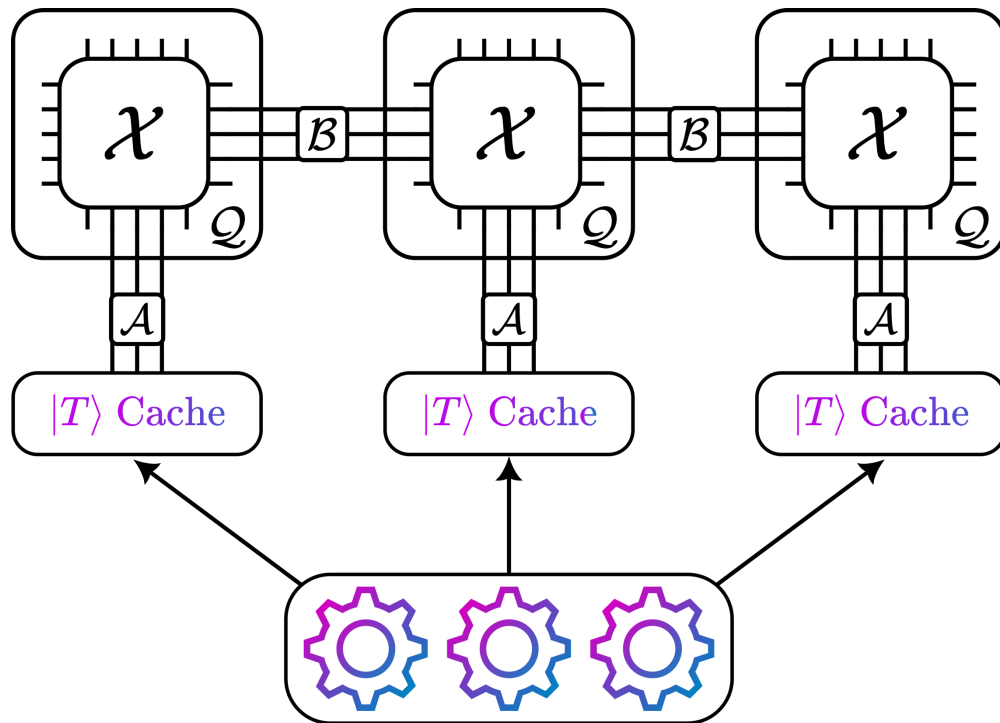
- Every logical measurement takes  $O(d)$  syndrome rounds, has fault distance  $d$ .
- Can be built with *any* code  $Q$  and *any*  $|T\rangle$  state factory.
- For near-term, can use small QLDPC code + magic state cultivation.
- Entire system has fixed, constant-degree connectivity.
- 1 logical qubit in  $Q$  is used as ancilla.
- Factory should have partial extractors, intentionally unspecified here.

# Extractor Architecture



# Extractor Architecture

- Any logical Pauli supported on blocks and caches connected by bridges and adapters can be measured in  $O(d)$  syndrome rounds, with fault distance  $d$ .
- Operators supported on disjoint blocks can be measured in parallel by deactivating bridges/adapters.
- Minimal cost for modularity: bridges/adapters have size  $O(d)$ .
- Flexibility: global architecture can be tailored to hardware or application.



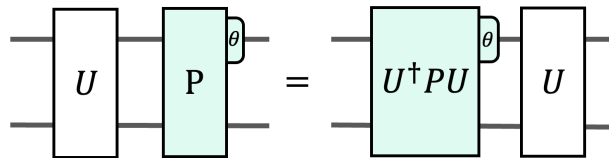


# Compilation for an Extractor Architecture

Compilation similar to Game of Surface Code. Given a logical circuit of Pauli rotations, we consider three types of gates:

1. Pauli  $\pi/8$  rotations,
2. Pauli  $\pi/4$  rotations supported within one EAC block (in-block Cliffords),
3. Pauli  $\pi/4$  rotations supported on two EAC blocks (cross-block Cliffords).

**We conjugate all in-block Cliffords to the end of the circuit.** This conjugation changes the Pauli axis of type 1 and 3 rotations, but do not expand their support to other EAC blocks.

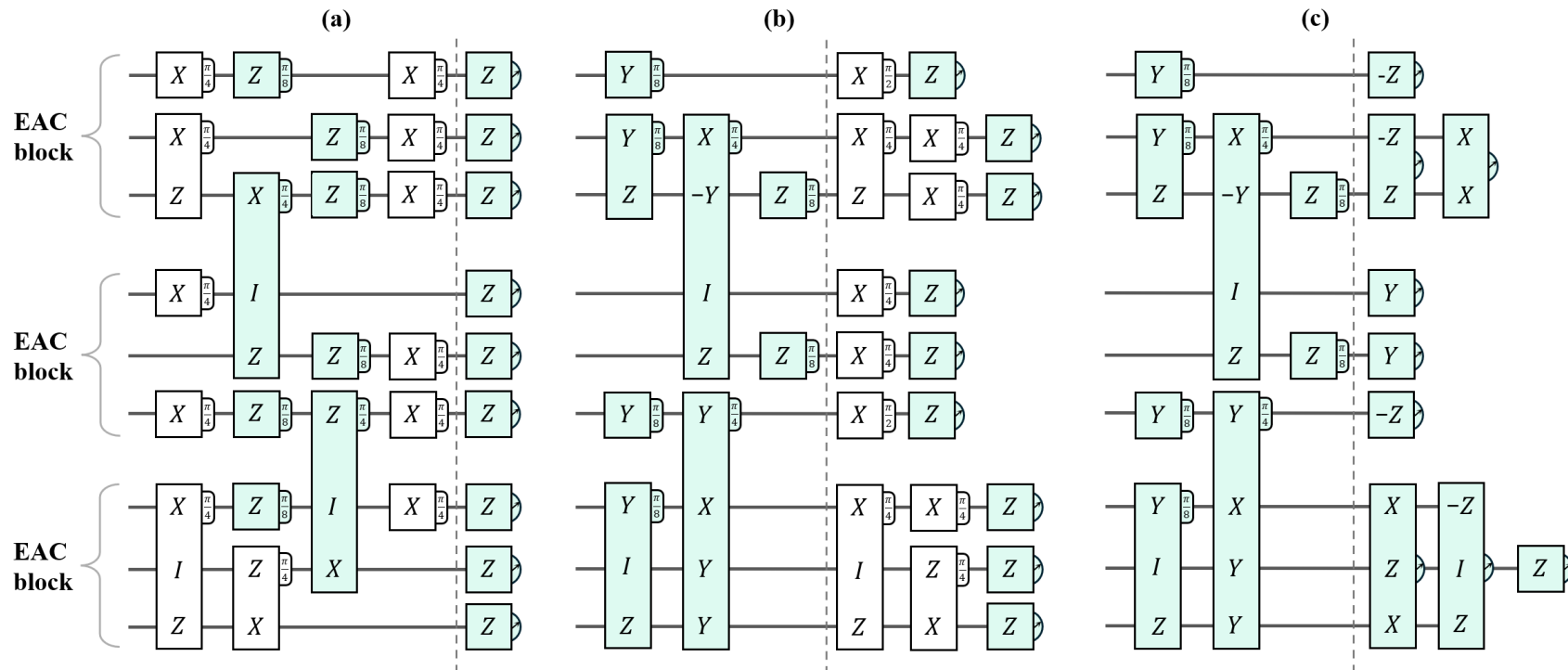


1. We implement type 1 & 3 rotations via logical measurements;
  2. In-block Cliffords are absorbed by standard basis measurements at the end of the computation.
-

# Circuit Example

1. Conjugate all in-block Cliffords to the end of the circuit.

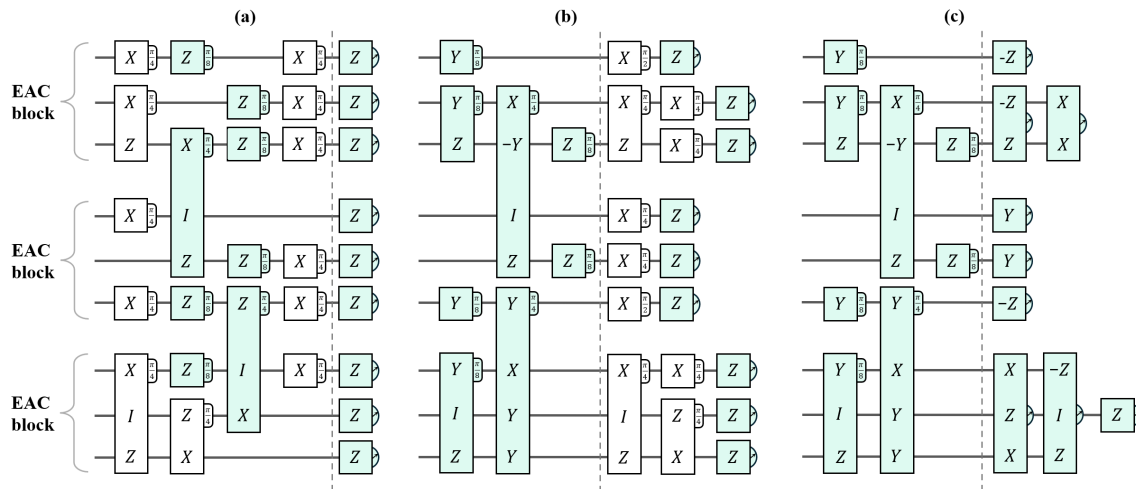
2. Absorb in-block Cliffords by the final measure-out.



Green operations are what we compile and implement. White operations are in-block Clifford that are compiled away.

# Remarks

- In-block Clifford gates are essentially free.
- This compilation heavily relies on the fact extractors can measure any logical Pauli.
- Bottleneck: magic state supply speed and number of cross-block gates.
- Highly optimizable for specific applications.



I. Motivation: A QLDPC-Based Quantum Computer

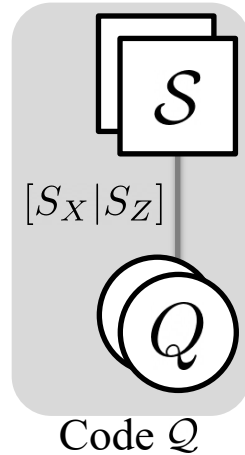
II. Code Surgery and Extractors

III. Extractor Architecture and Compilation

IV. Building an Extractor with Graph Theory

V. Discussions and Outlooks

# Scalable Tanner Graphs



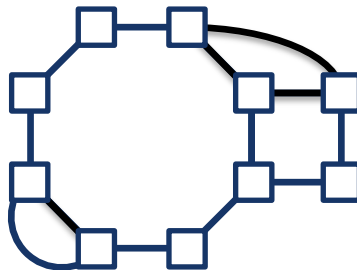
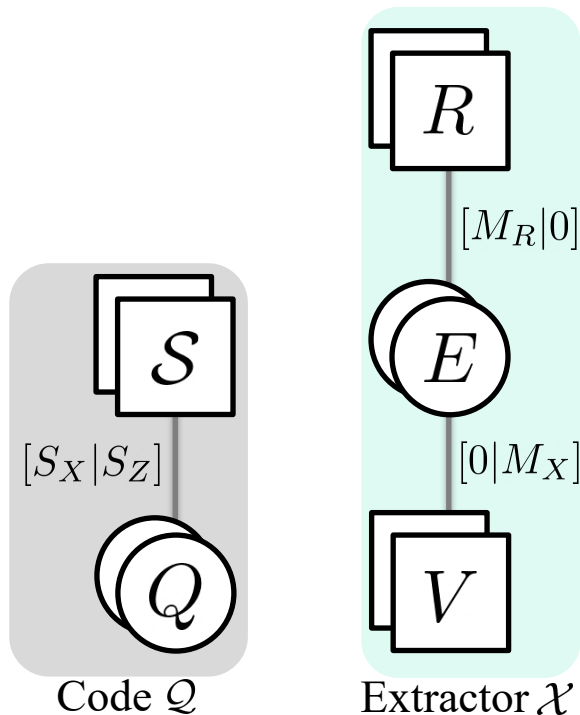
Stabilizers of the code  $Q$

Symplectic check matrix

Physical qubits of the code  $Q$

Code  $Q$

# Building an Extractor from a Graph

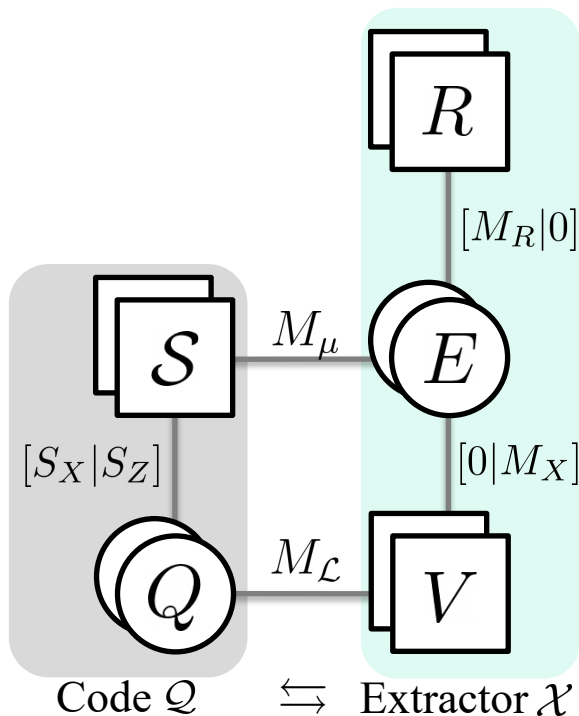


Let  $X = (V, E)$  be a graph.

1. For every edge in  $E$ , create an ancilla qubit.
2. For every vertex in  $V$ , create an ancilla check, which act on adjacent edge qubits by Pauli  $Z$ .
3. Pick a cycle basis  $R$  of  $X$ . For every cycle  $C$  in  $R$ , create an ancilla check, which act on edges in  $C$  by Pauli  $X$ .

**This ancilla system, the extractor system, commutes.**

# Building an Extractor from a Graph



We will build **fixed connections** between:

1. Vertex checks  $V$  and qubits of  $\mathcal{Q}$ ;
2. Stabilizers  $S$  and ancilla edge qubits  $E$ .

What's their Pauli action?

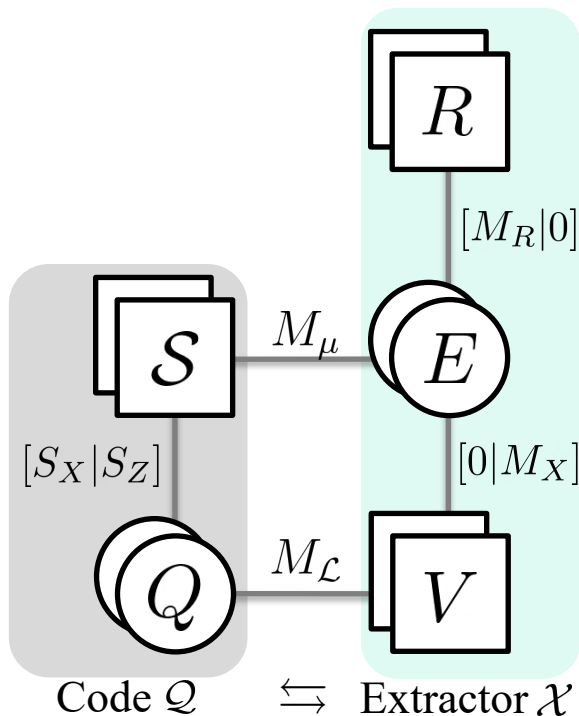
**Depends on the operator we want to measure!**

Given operator  $\mathcal{L}$ , we will pick symplectic matrices  $M_{\mathcal{L}}$  and  $M_\mu$  so that

1. Entire system in EAC block commutes. I.e, we have a well-defined measurement code  $\mathcal{Q}_{\mathcal{L}}$ .
2. Product of vertex checks  $V$  equals to  $\mathcal{L}$ .

**Measuring stabilizers of  $\mathcal{Q}_{\mathcal{L}}$  for  $O(d)$  rounds gives logical measurement of  $\mathcal{L}$  fault-tolerantly.**

# Many Important Details...



Many details not discussed in this talk:

1. Why is this system **LDPC**?
  2. How to connect  $S$  with  $E$  and  $Q$  with  $V$ ?
  3. How to choose matrices  $M_\mathcal{L}$  and  $M_\mu$ ?
  4. How to prove **fault-tolerance** of this code-switching process?
  5. **How to upper bound size of extractors by  $\tilde{O}(n)$ ?**
  6. **Most importantly, how to build this in practice?**
- All proved & discussed in the paper with graph theory.



I. Motivation: A QLDPC-Based Quantum Computer

II. Code Surgery and Extractors

III. Extractor Architecture and Compilation

IV. Building an Extractor with Graph Theory

V. Discussions and Outlooks

# The Landscape of QLDPC Computation

## Symmetry

- Transversal gates;
- Automorphisms gates;
- Fold-transversal gates.

## Teleportation-based

- Gate teleportation: magic states and Clifford states
- Homomorphic measurements

## Code deformation

- Code surgery and extractors
- Punctures?
- Code-switching?

Universal Computation = Symmetry + magic state factory + transversal CNOT + (multiple) Clifford state factories.

- Standard solution: multiple factories for different gates incurs heavy overhead.

Universal Computation = QLDPC memory + surgery + surface code computation.

- Hybrid architecture: surface code computation will quickly erase space advantage.

Universal Computation = Magic state factory + extractors (+ symmetry).

- Extractor architecture: in-block Cliffords are free, fixed & LDPC connectivity. Larger decoding instance.

Universal Computation = Symmetry + magic state factory + transversal CNOT/partial extractors.

- [Malcolm et al. 2502.07150]: inverse-exponential rate.

# Where does automorphism gates fit?

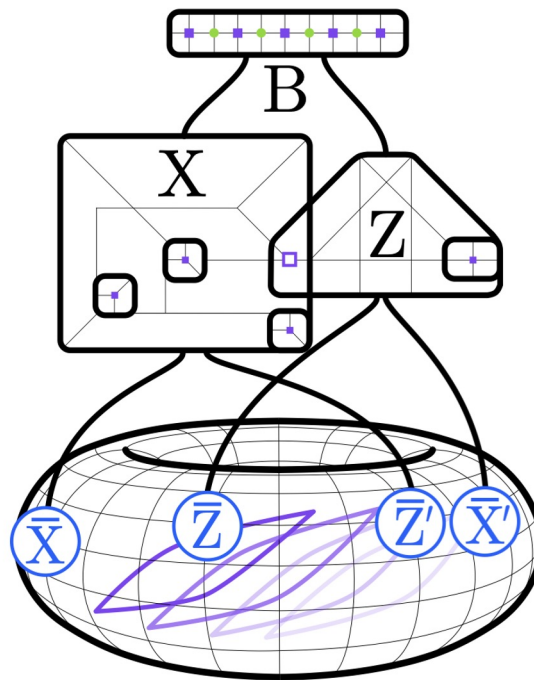
For a code with automorphism gates  $\mathcal{U}$ , we don't need to build a full extractor.

Instead, we can build a **partial extractor** which can measure Pauli operators in the set  $\mathcal{O}$ , such that  $\mathcal{U}^\dagger \mathcal{O} \mathcal{U}$  generate the full  $k-1$  qubit Pauli group.

➤ All  $\pi/4$  rotations on  $k-1$  qubits!

This is precisely the case for the 103-qubit system on the  $[[144, 12, 12]]$  gross code.  
[2407.18393]

Same applies if the code has other low-overhead logical operations.



Bridge used in  
partial extractor

Partial extractor

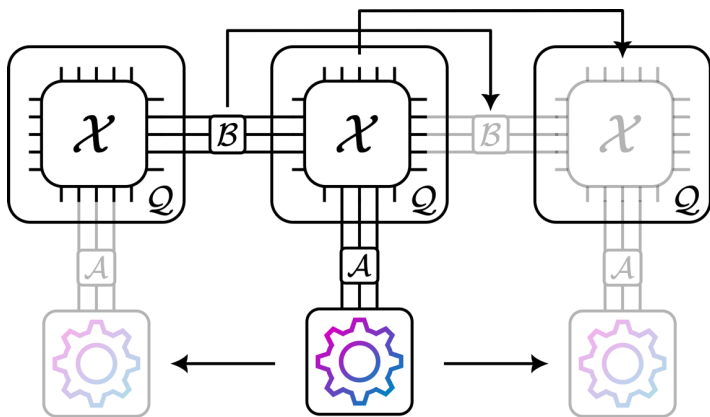
Gross code

# What if the hardware supports qubits movement?

Everything can move: factories, bridges, adapters, extractors.

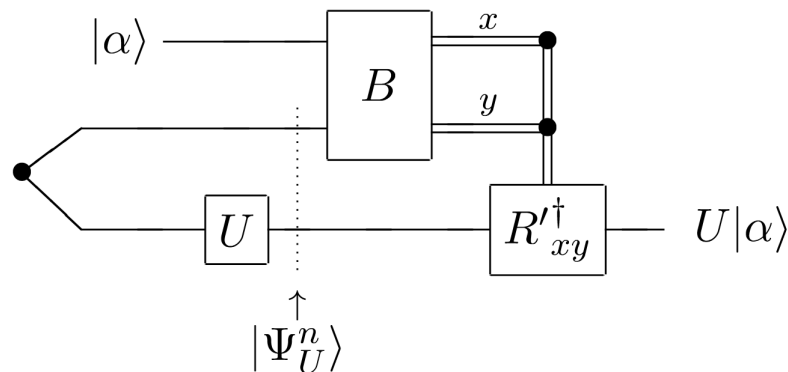
- Space overhead can be bounded by 'active' components.

Within one EAC block, a **moving partial extractor** can act as a full extractor.



Movement makes transversal CNOT easy.

- Extractor can prepare **arbitrary logical stabilizer state** 'offline', effectively as a '**Clifford factory**'.
- Gate teleportation lets us **perform arbitrary Clifford operation in  $O(1)$  'online' step**.
- Universal = **addressable non-Clifford(?)**\* + extractor Clifford factory + transversal CNOT.



\* See discussion & constructions in [2502.01864]

# Ending Remarks & Hot Takes

🌶️ Assuming a source of magic states (which is an almost default assumption), Extractor is a one-stop solution to perform universal FT computation on any stabilizer code with low overhead.

☀️ A new and open frontier for explorations and optimizations

- Build extractors on well-studied codes;
- Design extractor architectures for specific applications/algorithms;
- Better & hardware-specific compilation.

🌶️ Design the best possible memory, extractors will take care of the rest.

- Code – decoder – extractor codesign.

🌶️ Extractors will likely scale before the standard solution. If optimized and built, it could compete with surface code architectures in a practically large-scale regime.

- Challenge ahead: **LDPC hardware, fast and accurate decoding.**

# Extractors: QLDPC Architectures for Efficient Pauli-Based Computation

Zhiyang He (Sunny), Alexander Cowtan, Dominic Williamson, Theodore Yoder

