Vegapunk: Accurate and Fast Decoding for Quantum LDPC Codes with Online Hierarchical Algorithm and Sparse Accelerator

Kaiwen Zhou[1], Liqiang Lu[1]*, Debin Xiang[1], Chenning Tao[1], Anbang Wu[2], Jingwen Leng[2], Fangxin Liu[2], Mingshuai Chen[1], Jianwei Yin[1]*

[1]Zhejiang University, [2]Shanghai Jiao Tong University

MICRO 2025, Seoul, Korea

Outline

- **Background**
- Motivation
- Vegapunk Algorithm
- Vegapunk Accelerator
- Experiment

Quantum Computers are Noisy

- The most significant barrier for large-scale quantum computing: **Errors**

Current Physical Error Rate

$\approx 10^{-3}$ to 10^{-4}

Required Error Rate

$< 10^{-12}$

Quantum Error Correction (QEC) is crucial for practical, large-scale quantum computing.

Basics of QEC Code

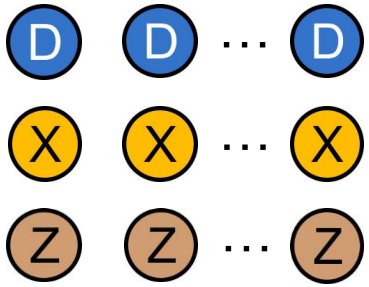
- The QEC code encodes multiple physical qubits into a single logical qubit to resist noise.

A single matchstick

stack

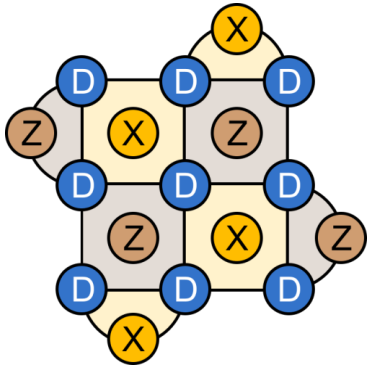
More Stable!

A pile of matchsticks



Multiple physical qubits

encode



A fault-tolerant logical qubit

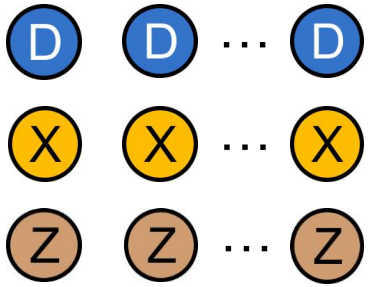
Basics of QEC Code

- The QEC code encodes multiple physical qubits into a single logical qubit to resist noise.

A single matchstick

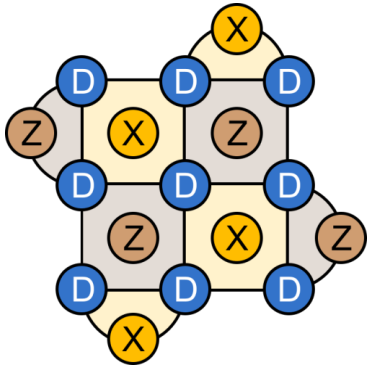
stack

A pile of matchsticks



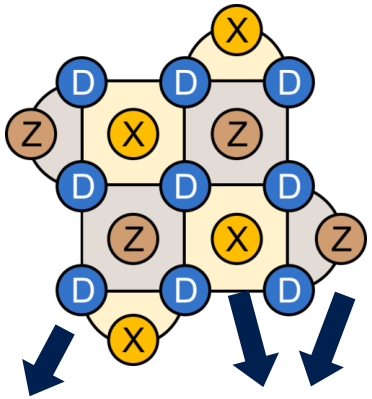
Multiple physical qubits

encode



A fault-tolerant logical qubit

- Consider surface code [[9, 1, 3]] as an example to illustrate key parameters [[n, k, d]] of QEC codes.



Data Qubit

Stabilizers (Parity Qubit)

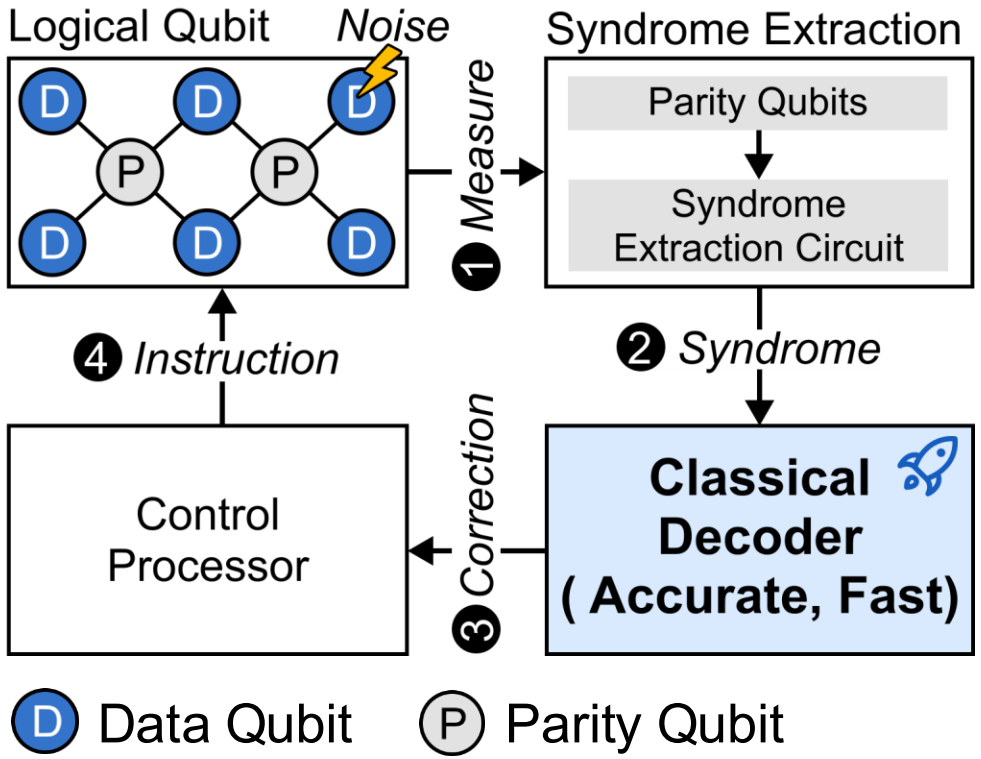
Key Parameters [[n, k, d]]

- **n**: the number of data qubits in a logical block.
- **k**: the number of encoded logical qubits.
- **d**: the code distance, quantifying the error tolerance.
- **Encoding Rate**: k / (number of physical qubits).

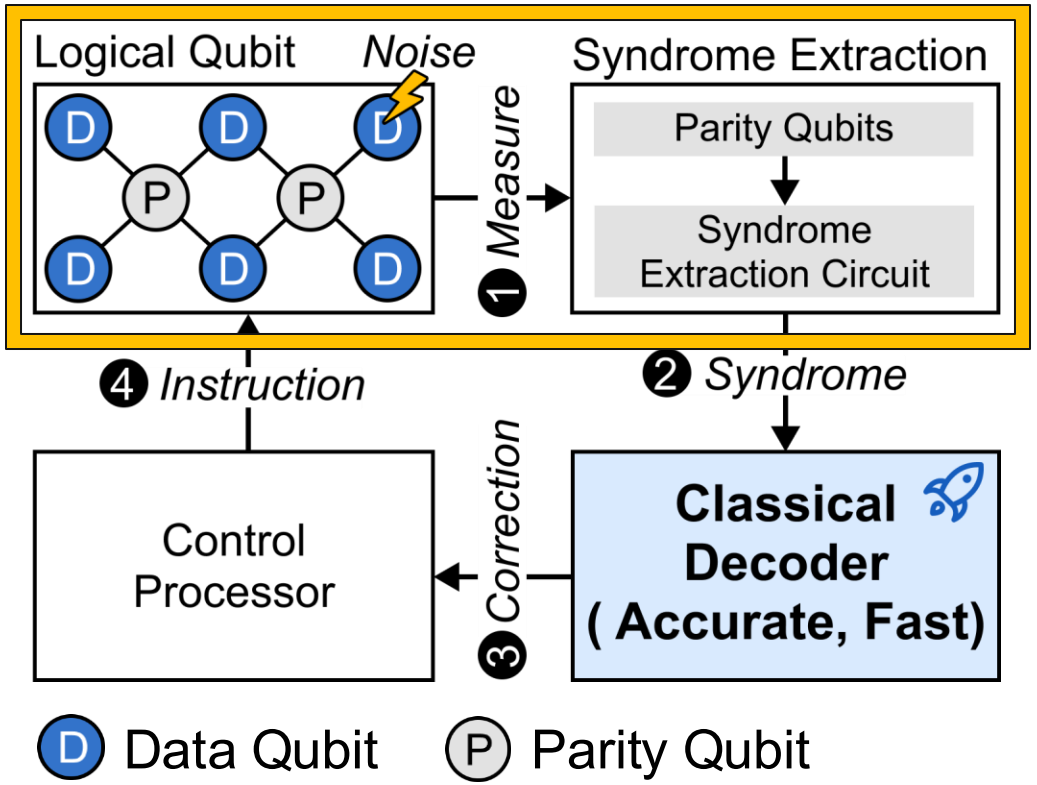
Error Detection

- Bit Flip (**X**) Errors → **Z** Stabilizers
- Phase Flip (**Z**) Errors → **X** Stabilizers

Overview of QEC Process

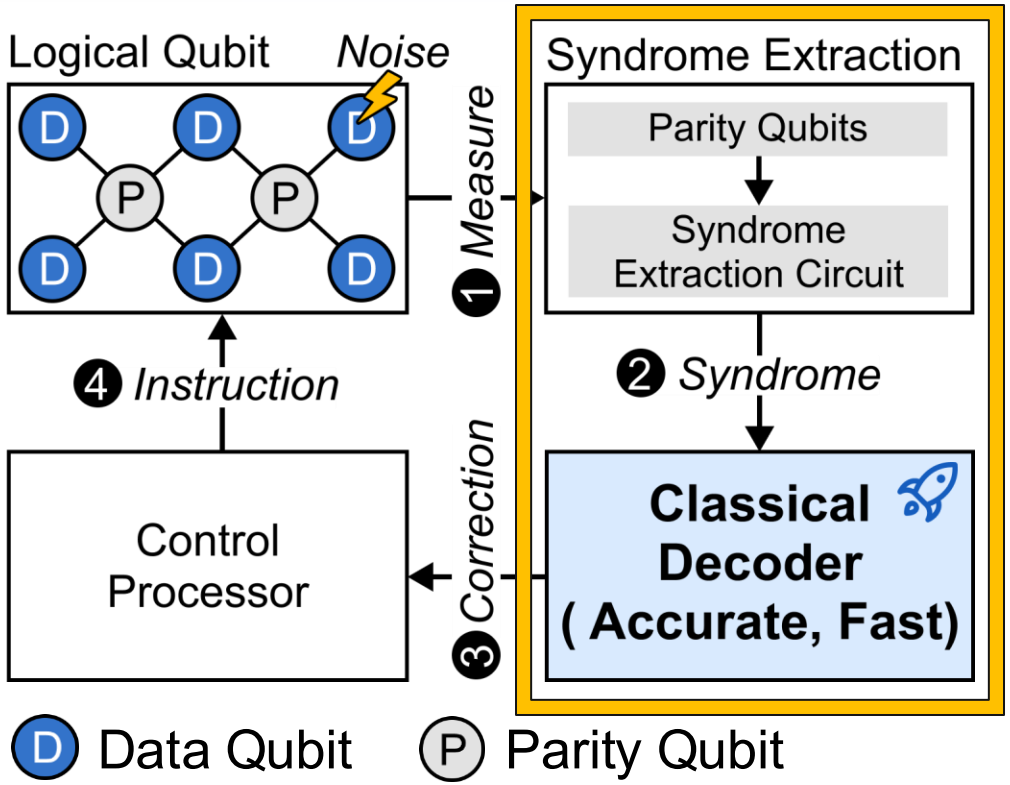


Overview of QEC Process



① The syndrome extraction circuit is executed on parity qubits to **generate the syndrome.**

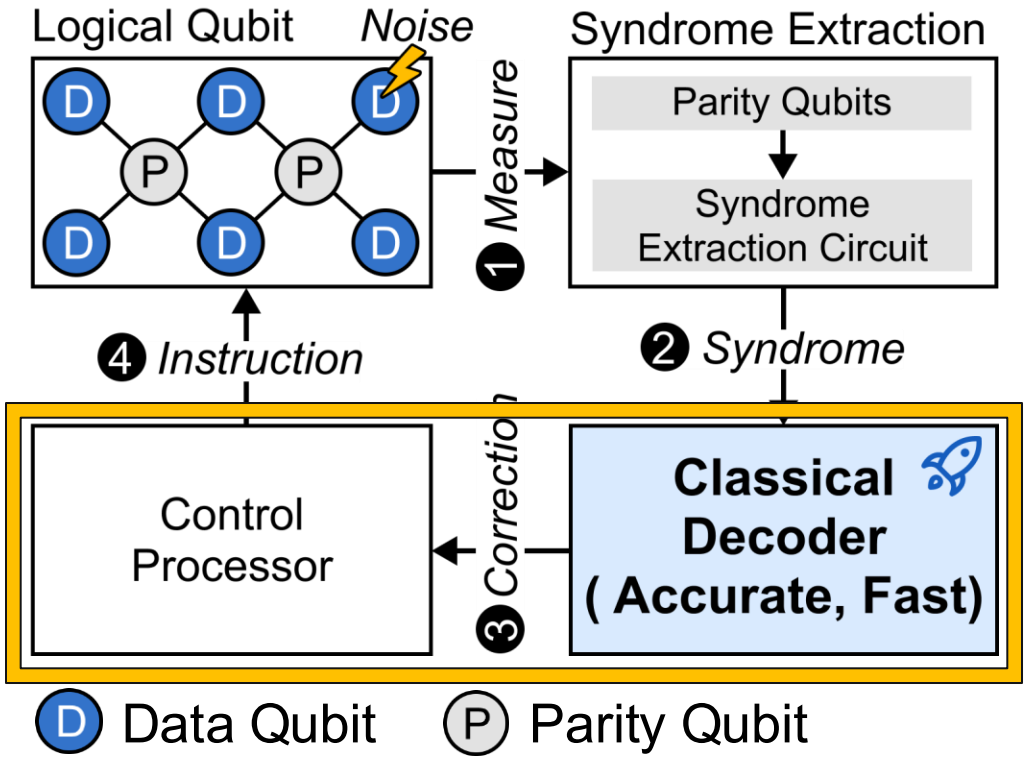
Overview of QEC Process



① The syndrome extraction circuit is executed on parity qubits to **generate the syndrome.**

② The syndrome is sent to a classical decoder, which **identifies error locations** on data qubits.

Overview of QEC Process

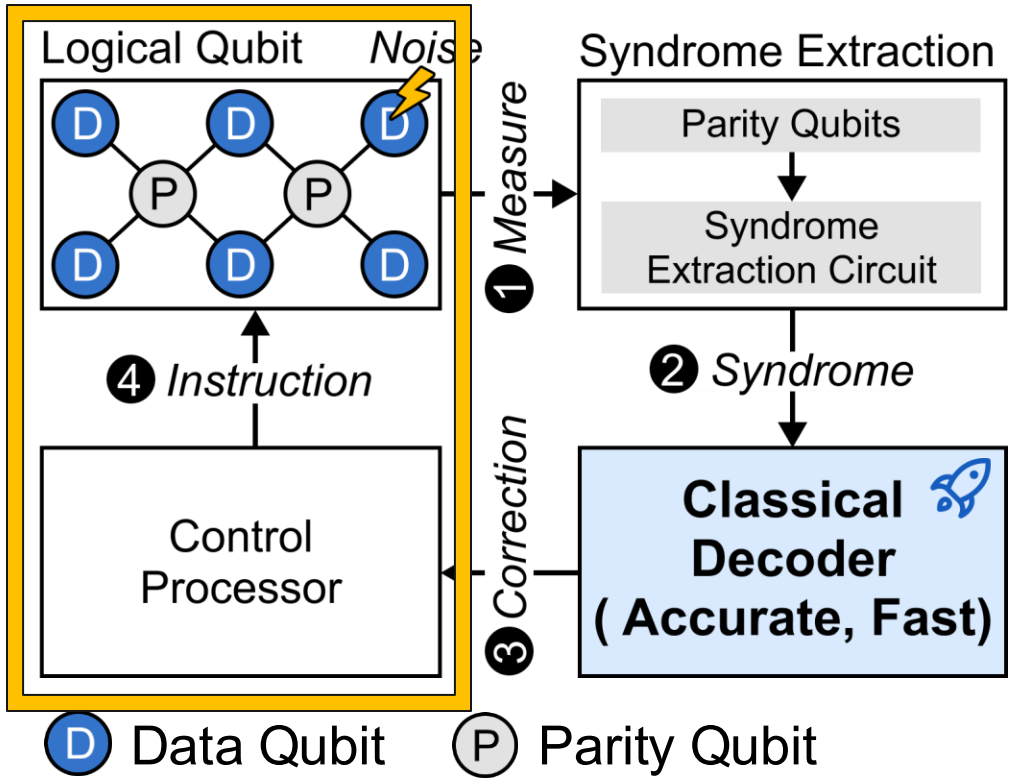


① The syndrome extraction circuit is executed on parity qubits to **generate the syndrome.**

② The syndrome is sent to a classical decoder, which **identifies error locations** on data qubits.

③ The decoder **computes a logical correction** and sends it to the control processor.

Overview of QEC Process



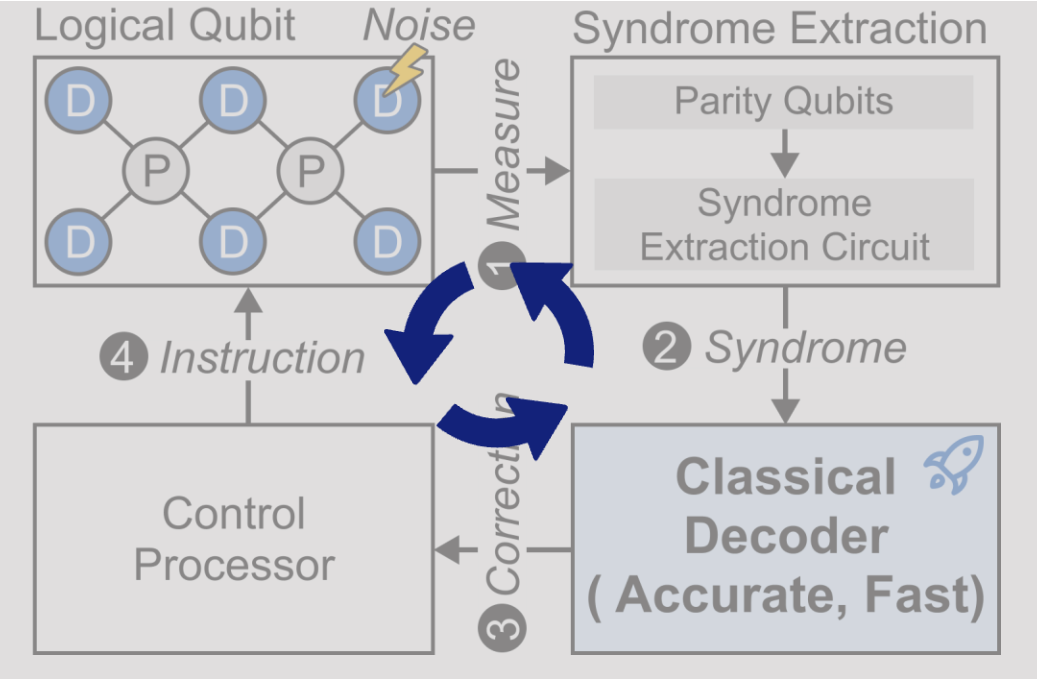
① The syndrome extraction circuit is executed on parity qubits to **generate the syndrome.**

② The syndrome is sent to a classical decoder, which **identifies error locations** on data qubits.

③ The decoder **computes a logical correction** and sends it to the control processor.

④ The control processor receives the correction and **sends QEC instructions to correct errors.**

Overview of QEC Process



① The syndrome extraction circuit is executed on parity qubits to **generate the syndrome.**

② The syndrome is sent to a classical decoder, which **identifies error locations** on data qubits.

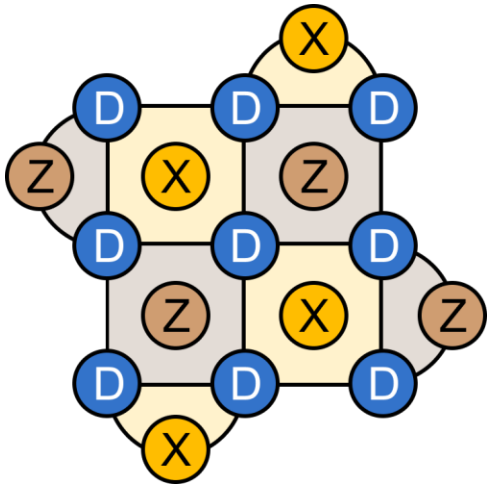
③ The decoder **computes a logical correction** and sends it to the control processor.

④ The control processor receives the correction and **sends QEC instructions to correct errors.**

- **The QEC process is executed in cycles to protect quantum information.**
- **Decoder identifies the location and type of errors, according to the syndrome.**

Scalable Architectural Solution: qLDPC Code

Surface Code

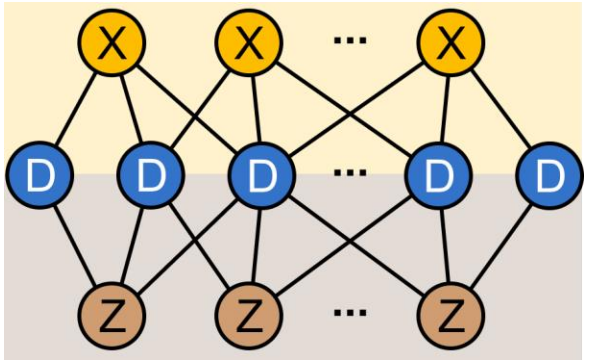


Resource-Intensive ✗

Low Encoding Rate!

$1 / 2n^2-1$

General qLDPC Code



Resource-Efficient ✓

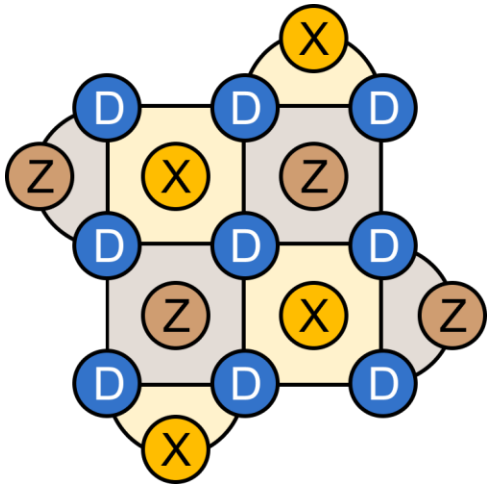
High Encoding Rate!

Asymptotically Constant

- Higher encoding rate means **fewer physical qubits** required for the same number of logical qubits.

Scalable Architectural Solution: qLDPC Code

Surface Code

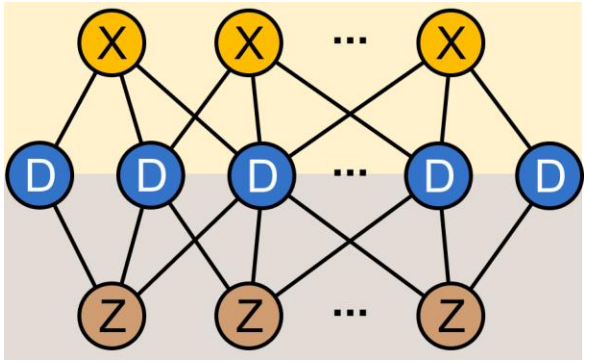


Resource-Intensive ✗

Low Encoding Rate!

$1 / 2n^2-1$

General qLDPC Code



Resource-Efficient ✓

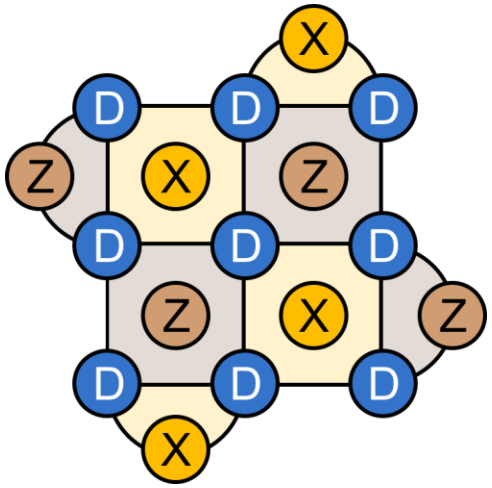
High Encoding Rate!

Asymptotically Constant

- Higher encoding rate means **fewer physical qubits** required for the same number of logical qubits.
- Let's take an example:
  - Surface code [[1452,12,11]] **requires 1452 data qubits** to encode 12 logical qubits.
  - Bivariate Bicycle (BB) code [[144,12,12]], a type of qLDPC codes, achieves the same number of logical qubits **using only 144 data qubits!**

Scalable Architectural Solution: qLDPC Code

Surface Code

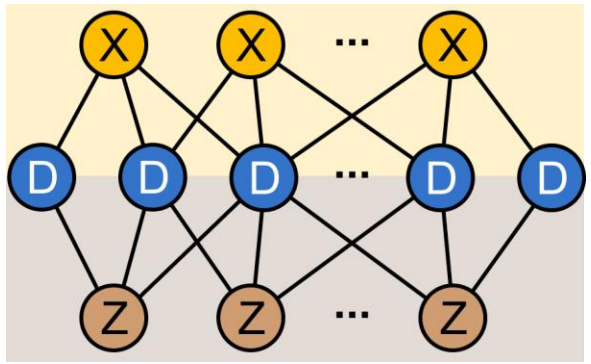


Resource-Intensive ✗

Low Encoding Rate!

$1 / 2n^2-1$

General qLDPC Code



Resource-Efficient ✓

High Encoding Rate!

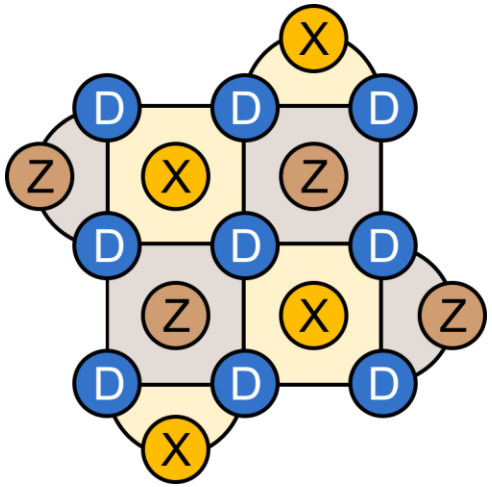
Asymptotically Constant

- Higher encoding rate means **fewer physical qubits** required for the same number of logical qubits.
- Let's take an example:
- Surface code [[1452,12,11]] **requires 1452 data qubits** to encode 12 logical qubits.

We focus on qLDPC code, since it offers a scalable solution for superconducting platforms.

qLDPC Decoding is Not Easy

Surface Code

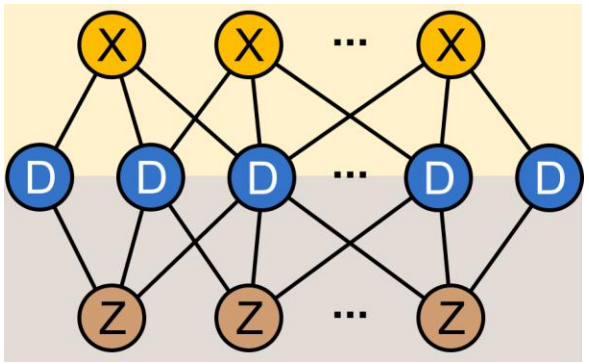


Resource-Intensive ✗

2D-Grid Structure ✓

Local Connectivity ✓

General qLDPC Code



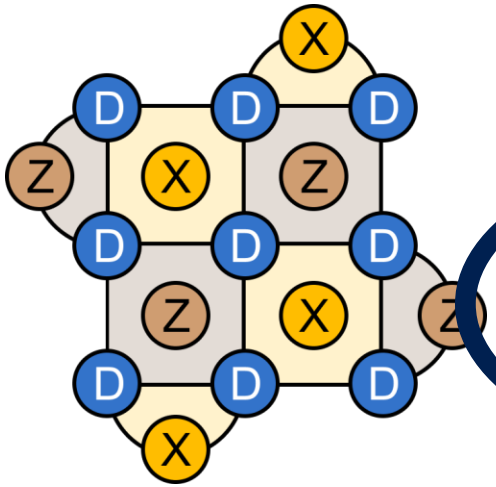
Resource-Efficient ✓

Hypergraph Structure ✗

Non-Local Connectivity ✗

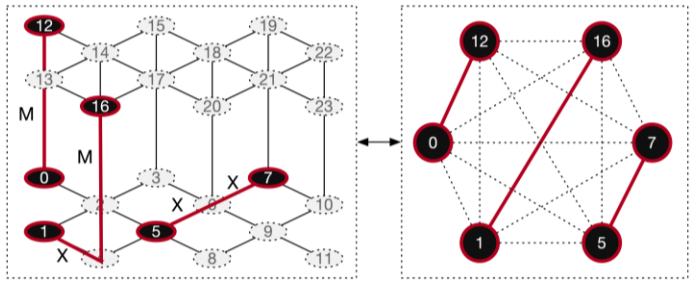
qLDPC Decoding is Not Easy

Surface Code



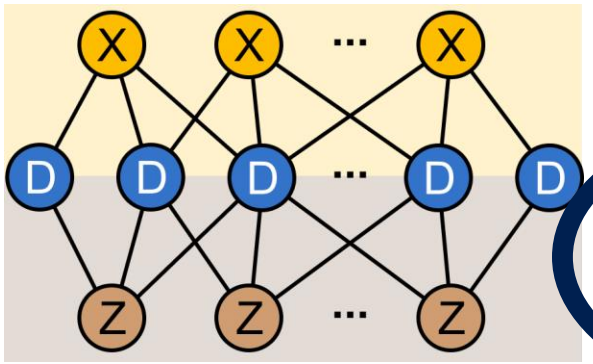
- Resource-Intensive ✗
- 2D-Grid Structure ✓
- Local Connectivity ✓

Efficient Decoding (Polynomial-Time)



Minimum Weight Perfect Matching (MWPM)

General qLDPC Code



- Resource-Efficient ✓
- Hypergraph Structure ✗
- Non-Local Connectivity ✗

Inefficient Decoding (NP-Hard)

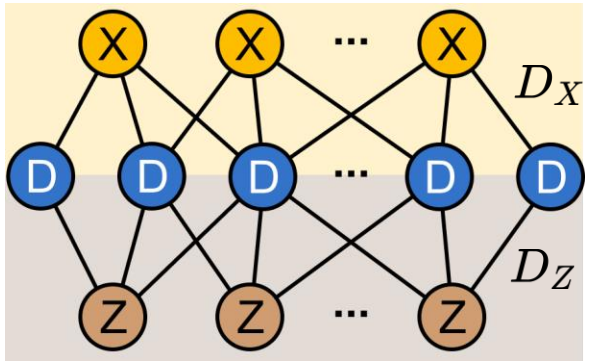
$$e^{\mathrm{ml}} = \arg\max_{e}\{\ln(P(e))\}$$
$$= \arg\max_{e}\left(C - \sum_{j} w_j e_j\right)$$
$$\text{s.t. } s = H_Z e,$$

Maximum Likelihood Decoding (MLD) [1]

[1] Iyer, Pavithran, and David Poulin. "Hardness of decoding quantum stabilizer codes." IEEE Transactions on Information Theory 61.9 (2015): 5209-5223.

Decoding Objective and Quantum Degeneracy

- qLDPC decoding can be formulated as a minimum weight decoding problem, which is NP-hard.



Tanner Graph

$$D_Z = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mn} \end{bmatrix}$$

Parity Check Matrix

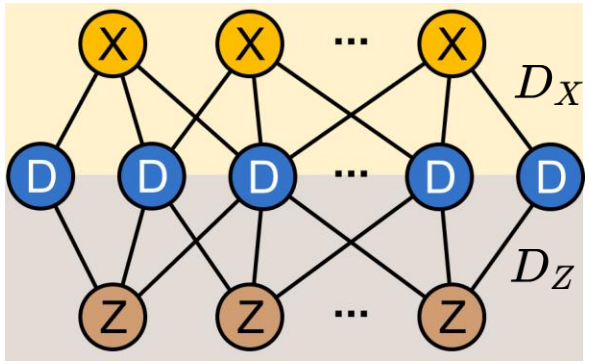
$$\vec{e}_{opt} = \arg\min_{\vec{e}} \sum_j w_j e_j$$

$$s.t.\ D_Z \cdot \vec{e} = \vec{s}$$

Decoding Objective
(minimum weight decoding)

Decoding Objective and Quantum Degeneracy

- qLDPC decoding can be formulated as a minimum weight decoding problem, which is NP-hard.



Tanner Graph

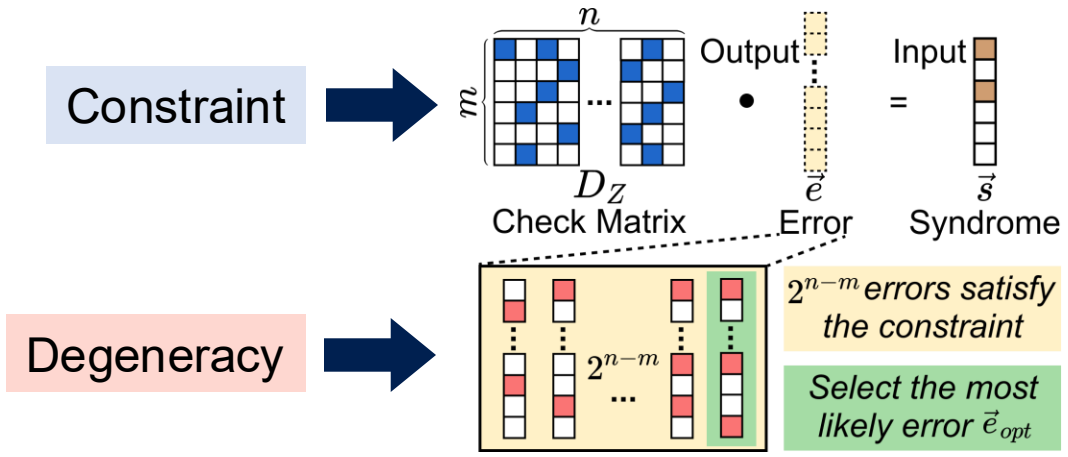
$$D_Z = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mn} \end{bmatrix}$$

Parity Check Matrix

$$\vec{e}_{opt} = \arg\min_{\vec{e}} \sum_j w_j e_j$$
$$s.t.\ D_Z \cdot \vec{e} = \vec{s}$$

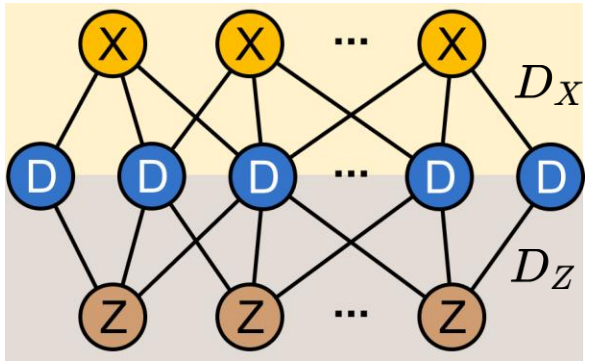
Decoding Objective
(minimum weight decoding)

- Decoding suffers from ***quantum degeneracy***, where multiple errors satisfy the check constraint.



Decoding Objective and Quantum Degeneracy

- qLDPC decoding can be formulated as a minimum weight decoding problem, which is NP-hard.



Tanner Graph

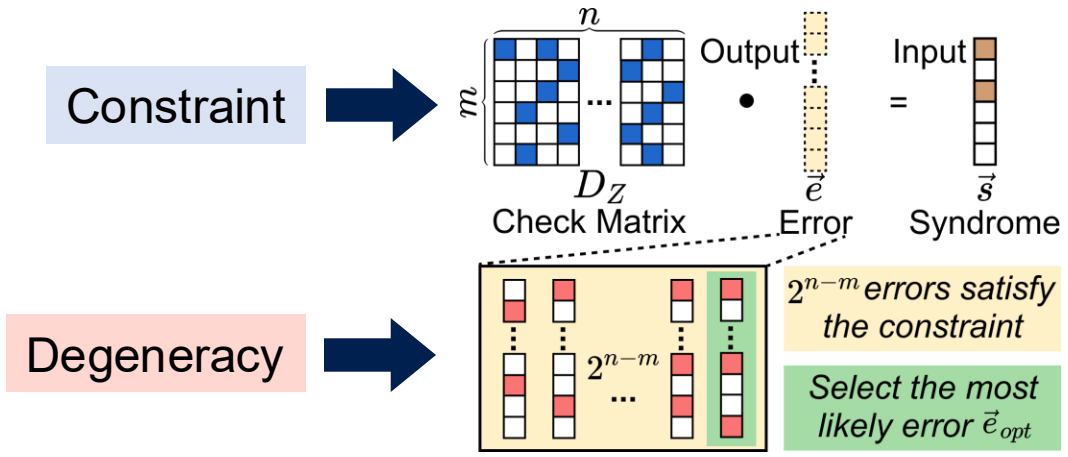
$$D_Z = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mn} \end{bmatrix}$$

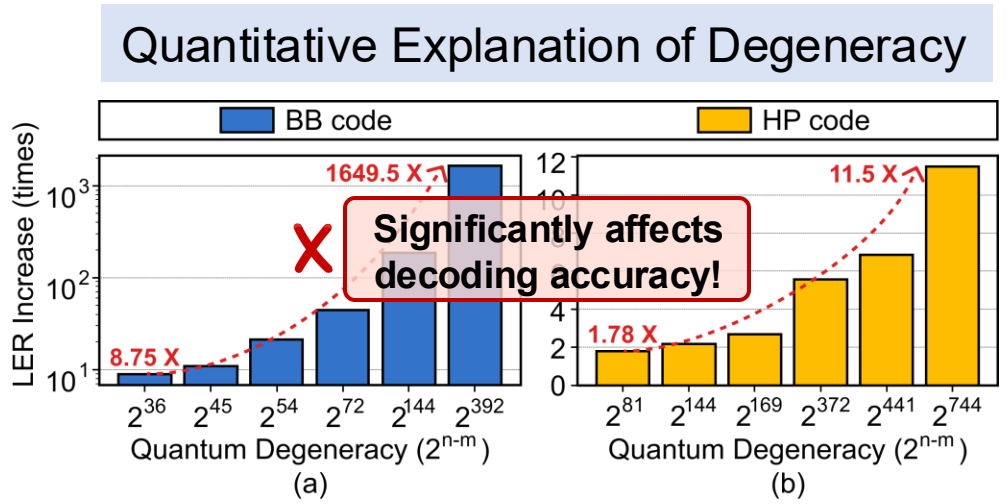
Parity Check Matrix

$$\vec{e}_{opt} = \arg\min_{\vec{e}} \sum_j w_j e_j$$
$$s.t.\ D_Z \cdot \vec{e} = \vec{s}$$

Decoding Objective
(minimum weight decoding)

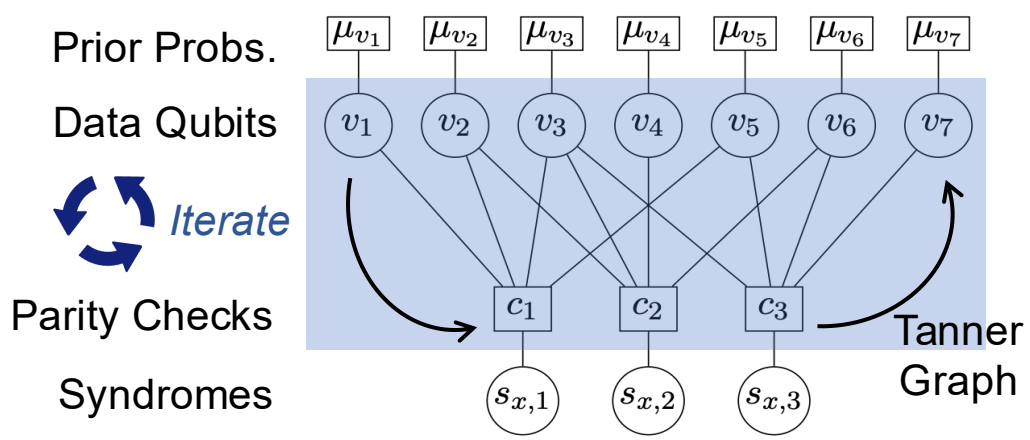
- Decoding suffers from ***quantum degeneracy***, where multiple errors satisfy the check constraint.





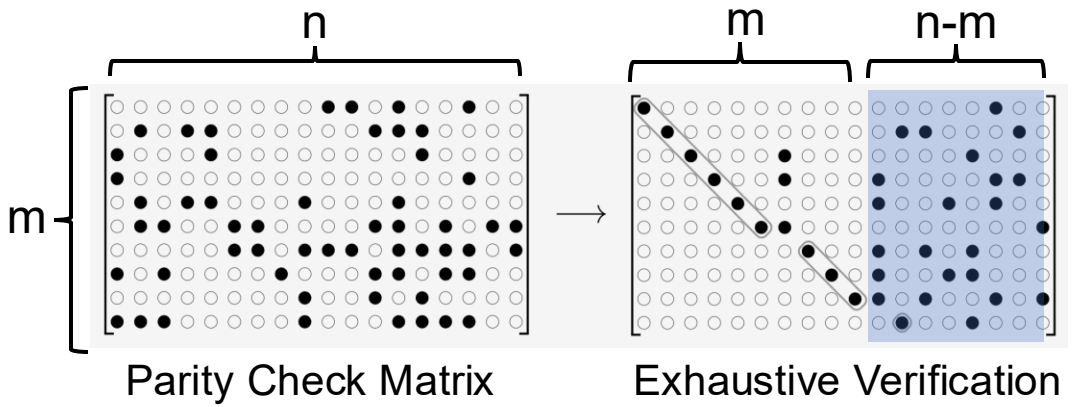
BP and BP+OSD Decoders

Belief Propagation (BP)



- BP **iteratively updates probability information** by passing messages on a Tanner graph to identify the most likely error pattern.
- Core characteristics: **high parallelism** and **low computational complexity**.

BP+Ordered Statistics Decoding (OSD)



- **If BP decoding fails, OSD part executes Gaussian elimination**, receiving probability information from BP to exhaustively verify possible solutions.
- Core characteristics: **high decoding accuracy** and **high computational complexity**.

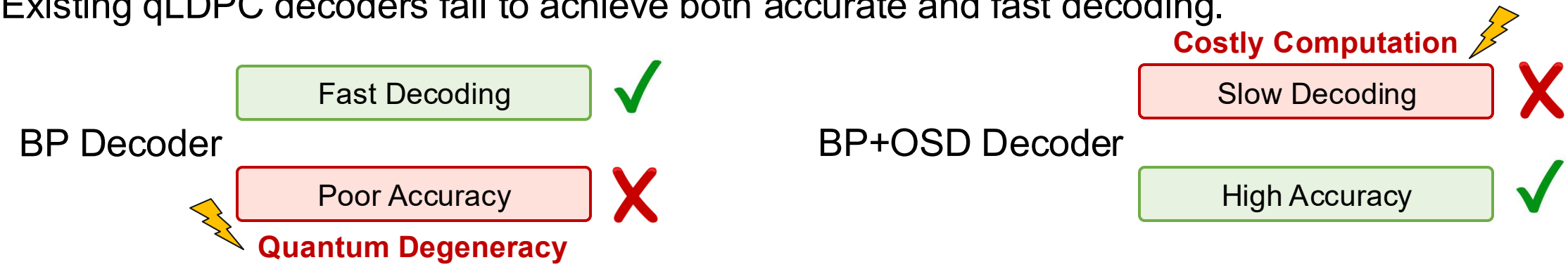
The vast majority of existing qLDPC decoders are built upon BP/OSD improvements.

Outline

- Background
- **Motivation**
- Vegapunk Algorithm
- Vegapunk Accelerator
- Experiment

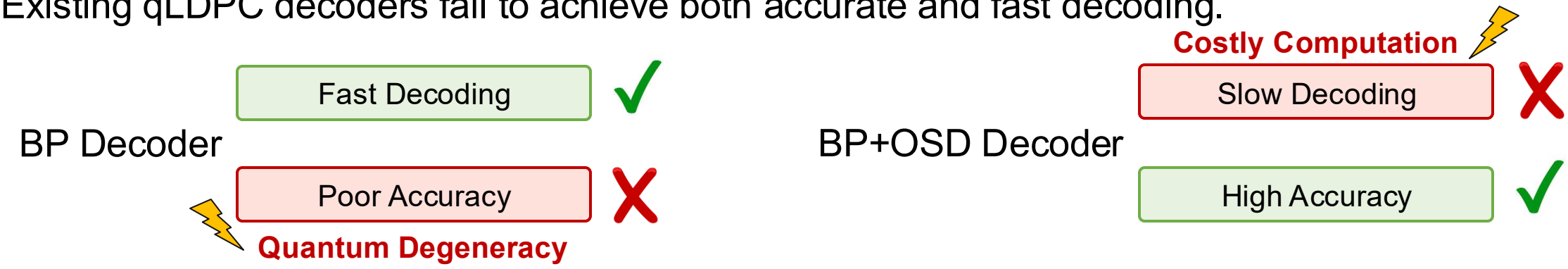
1. Prior Decoders: Accuracy-Latency Tradeoff

- Existing qLDPC decoders fail to achieve both accurate and fast decoding.

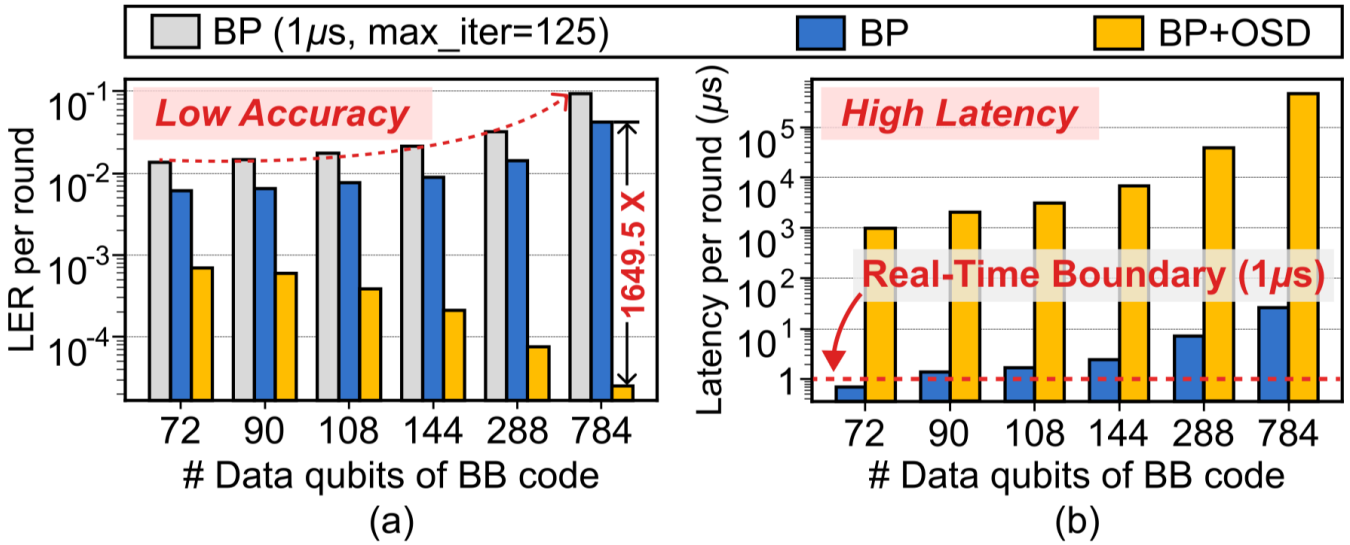


1. Prior Decoders: Accuracy-Latency Tradeoff

- Existing qLDPC decoders fail to achieve both accurate and fast decoding.



- Decoding performance profiling of BP and BP+OSD decoders under 0.1% circuit-level noise.

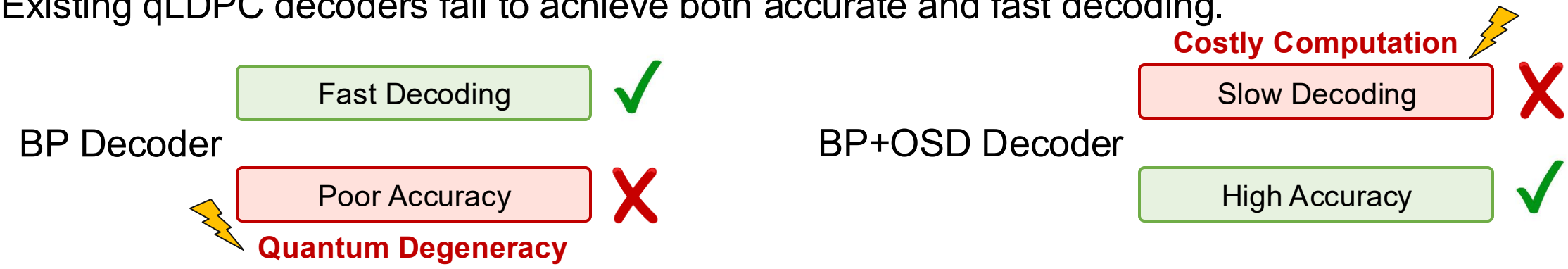


Observations:

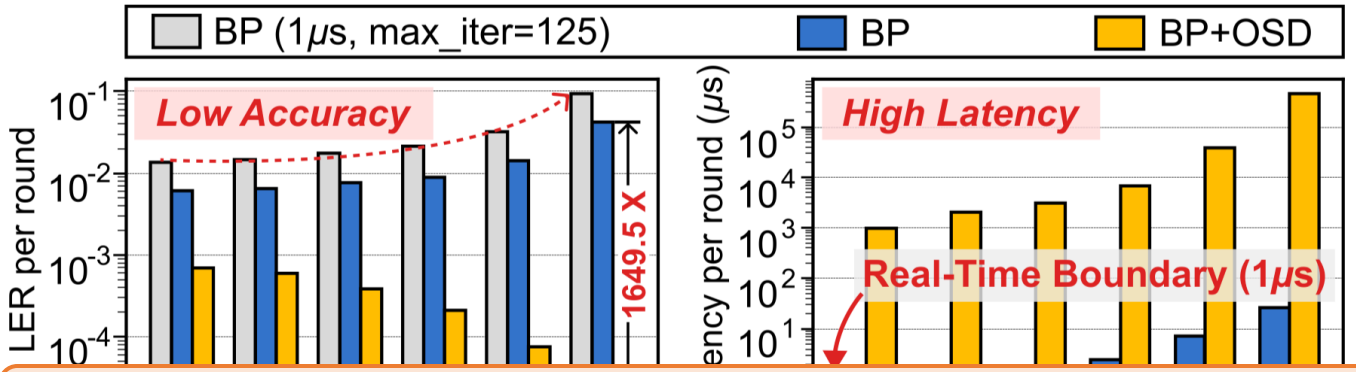
- BP's accuracy is significantly lower than BP+OSD's, rendering it unacceptable.
- Neither BP nor BP+OSD can meet the real-time decoding requirements (1μs) when scaled to large-scale qLDPC codes.

1. Prior Decoders: Accuracy-Latency Tradeoff

- Existing qLDPC decoders fail to achieve both accurate and fast decoding.



- Decoding performance profiling of BP and BP+OSD decoders under 0.1% circuit-level noise.



Observations:

- BP's accuracy is significantly lower than BP+OSD's, rendering it unacceptable.
- Neither BP nor BP+OSD can meet the

Goal: Develop an accurate qLDPC decoder that can process each round of syndrome in 1μs.

2. Low Latency: Why Dedicated Accelerator?

- General-purpose processors (e.g., CPUs or GPUs) are inefficient at handling the fine-grained and bit-level operations required in decoding.

CPU

GPU

Incompatible

Decoding Process

- Fine-grained Operation
- Bit-level Parallelism

2. Low Latency: Why Dedicated Accelerator?

- General-purpose processors (e.g., CPUs or GPUs) are inefficient at handling the fine-grained and bit-level operations required in decoding.

CPU

GPU

Incompatible

Decoding Process

Fine-grained Operation

Bit-level Parallelism

Lead to

Require

Low Utilization

Inefficient Memory Access

Mismatch

Low-latency Feedback

2. Low Latency: Why Dedicated Accelerator?

- General-purpose processors (e.g., CPUs or GPUs) are inefficient at handling the fine-grained and bit-level operations required in decoding.

CPU

GPU

Incompatible

Decoding Process

Fine-grained Operation

Bit-level Parallelism

Lead to

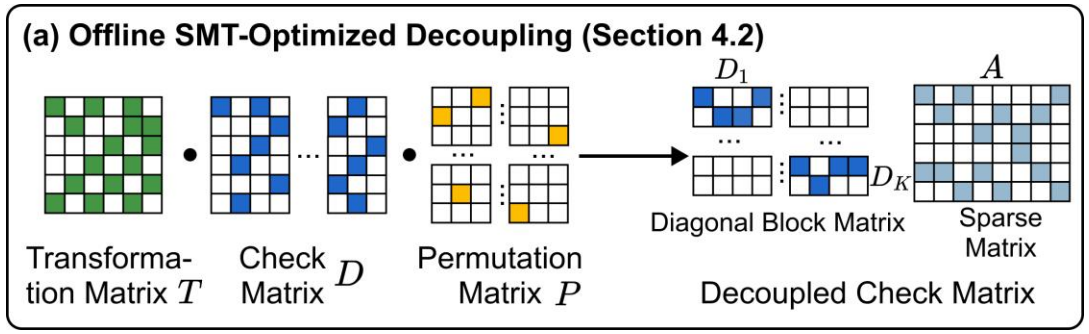
Require

We choose to design a dedicated accelerator for qLDPC decoding based on FPGAs.

Outline

- Background
- Motivation
- **Vegapunk Algorithm**
- Vegapunk Accelerator
- Experiment

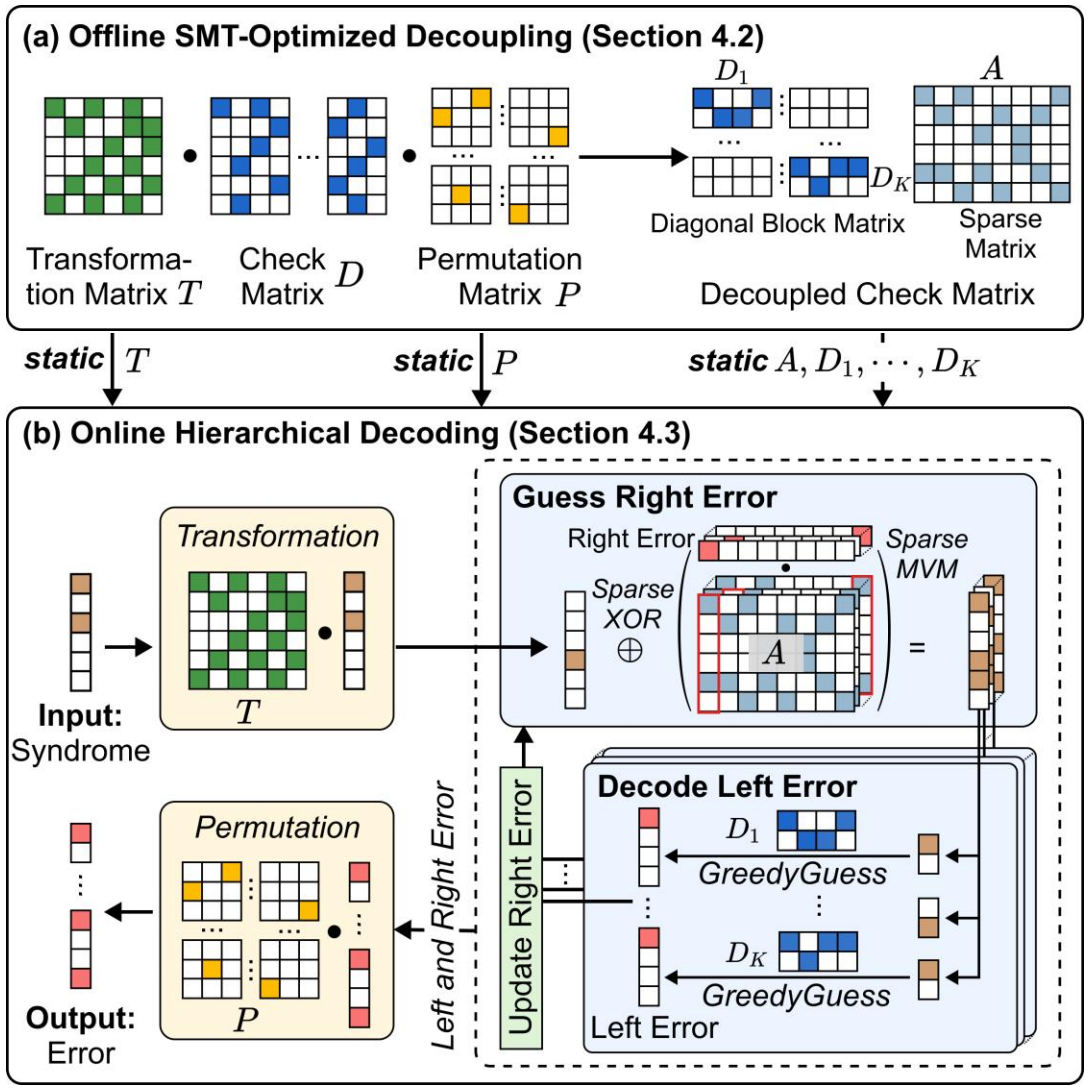
Workflow of Vegapunk Algorithm



To Address **Low Accuracy:**

- **Insight:** Degeneracy arises from the check matrix having far more columns than rows.
- **Solution:** Decouple the original check matrix into smaller sub-matrices, to balance the number of columns and rows.

Workflow of Vegapunk Algorithm



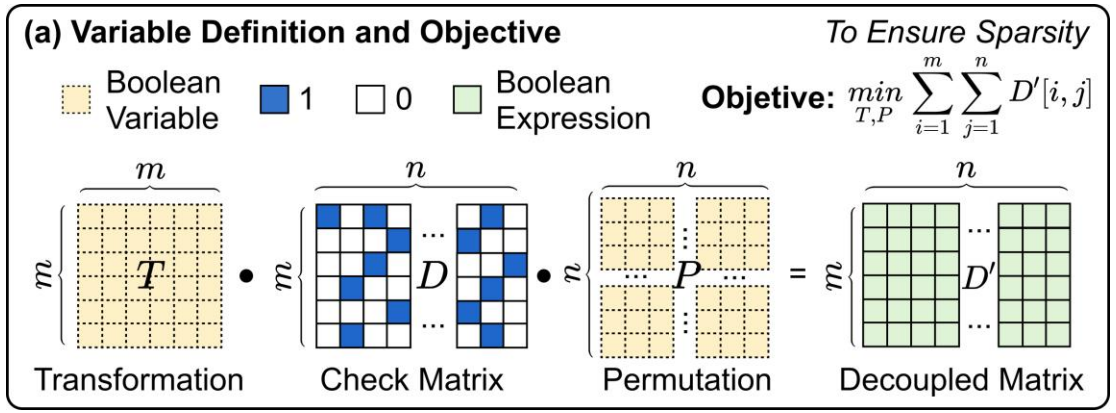
To Address **Low Accuracy:**

- **Insight:** Degeneracy arises from the check matrix having far more columns than rows.
- **Solution:** Decouple the original check matrix into smaller sub-matrices, to balance the number of columns and rows.

To Address **High Latency:**

- **Insight:** The probability of an error pattern exponentially decreases with the number of ‘1’s it contains.
- **Solution:** A highly parallel greedy decoding algorithm that incrementally increases the number of ‘1’s in error pattern, evaluates their likelihood at each step, and incorporates an early stopping mechanism.

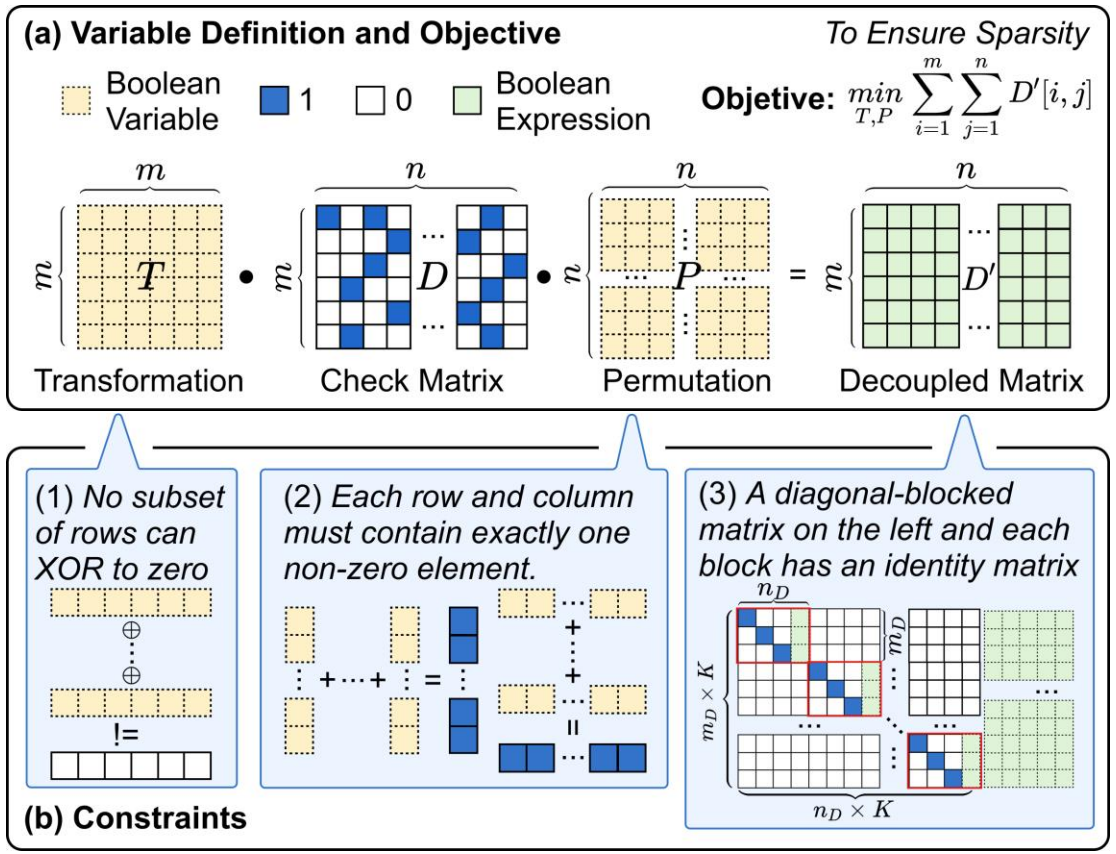
Offline SMT-Optimized Decoupling



Fundamental Idea: **Transform the Check Matrix**

- **Left-multiplying** it with a transformation matrix T.
- **Right-multiplying** it with a permutation matrix P.

Offline SMT-Optimized Decoupling

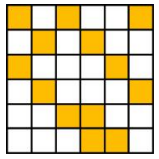
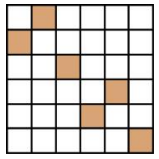
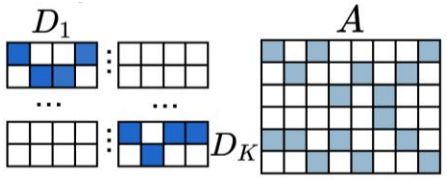


Fundamental Idea: **Transform the Check Matrix**

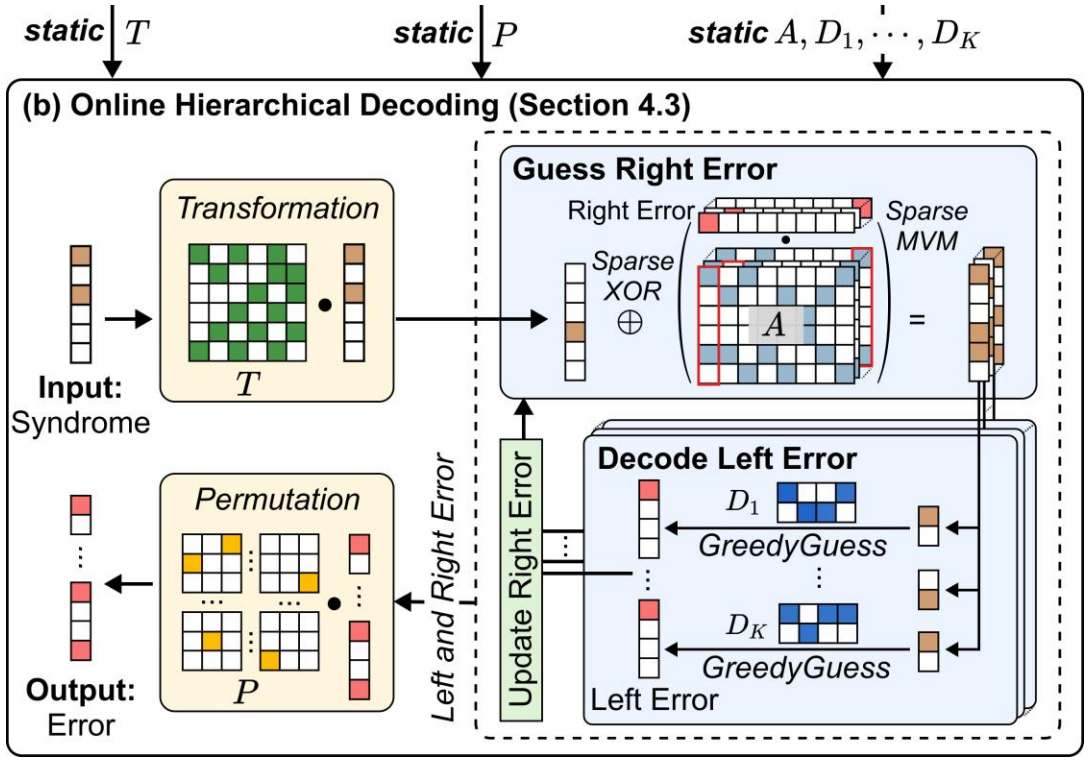
- **Left-multiplying** it with a transformation matrix T.
- **Right-multiplying** it with a permutation matrix P.

Fundamental Idea: **Reformulate into an SMT Problem**

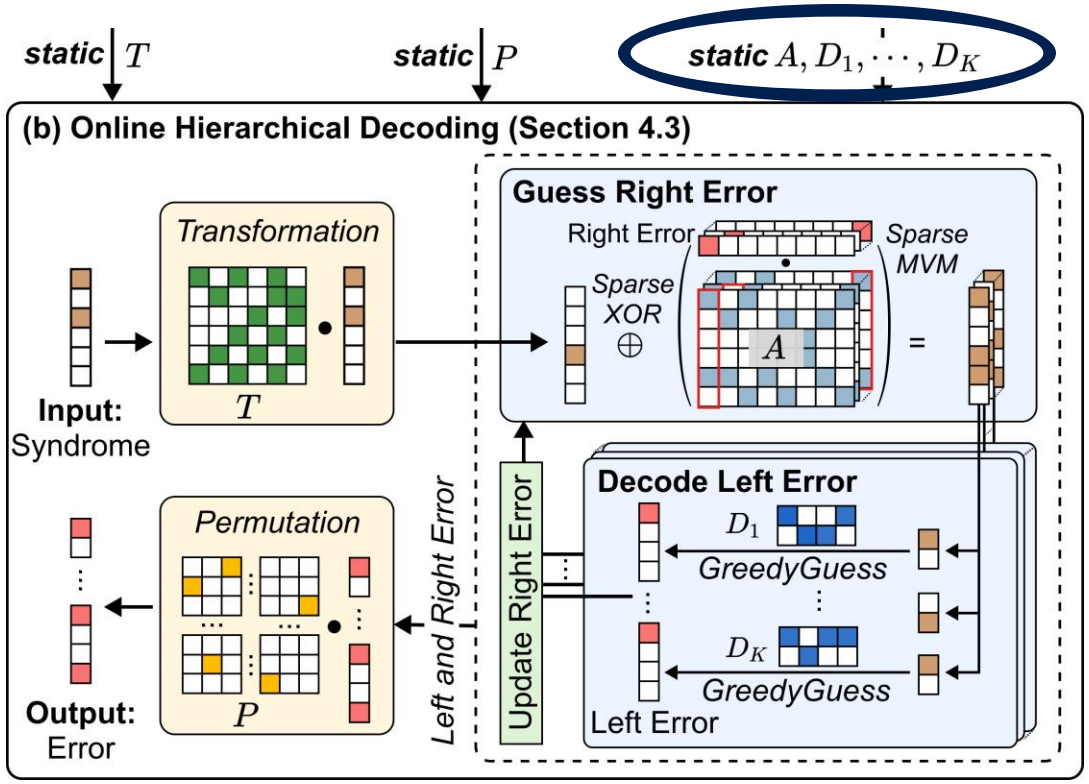
- By **setting constraints**, the decoupled matrix consists of a diagonal-block matrix and a sparse matrix.
- The objective of our SMT solver is to **maximize the sparsity** of the decoupled matrix.

SMT Solver → Permutation Matrix P  Transformation Matrix T  Decoupled Matrix D'  → **Online Decoding**

Online Hierarchical Decoding



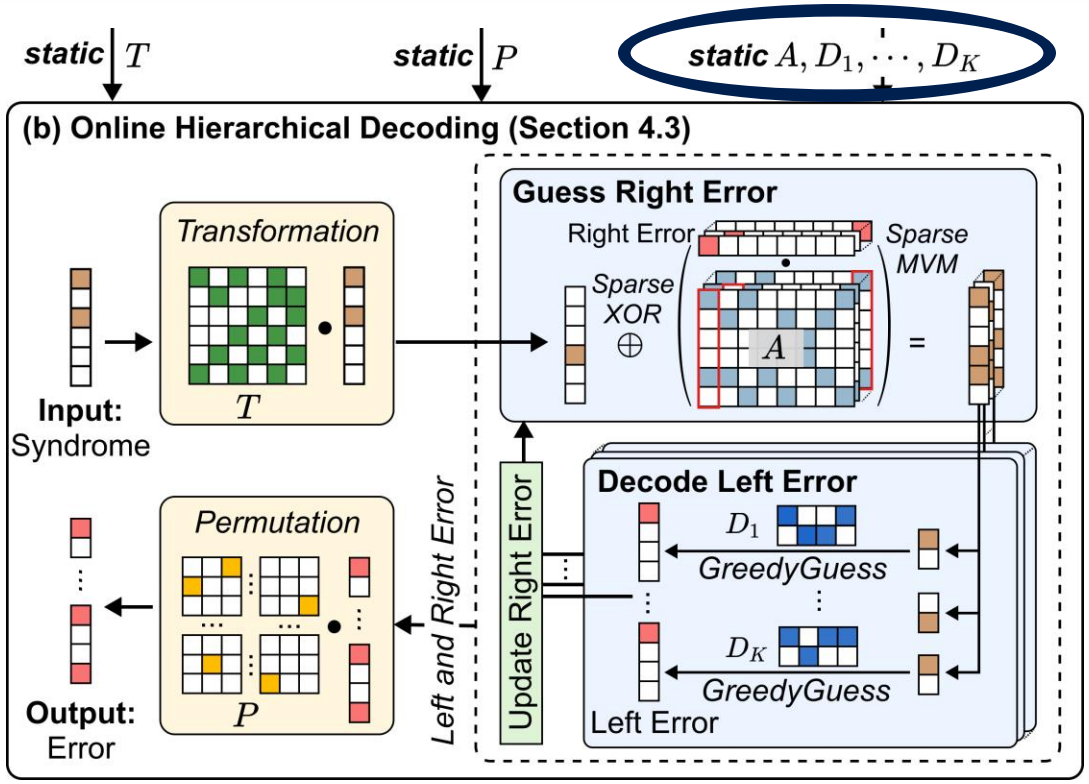
Online Hierarchical Decoding



The check matrix is decomposed into two parts:

$$\mathrm{diag}(D_1, D_2, \cdots, D_K) + A$$

Online Hierarchical Decoding



The check matrix is decomposed into two parts:

$$\mathrm{diag}(D_1, D_2, \cdots, D_K) + A$$

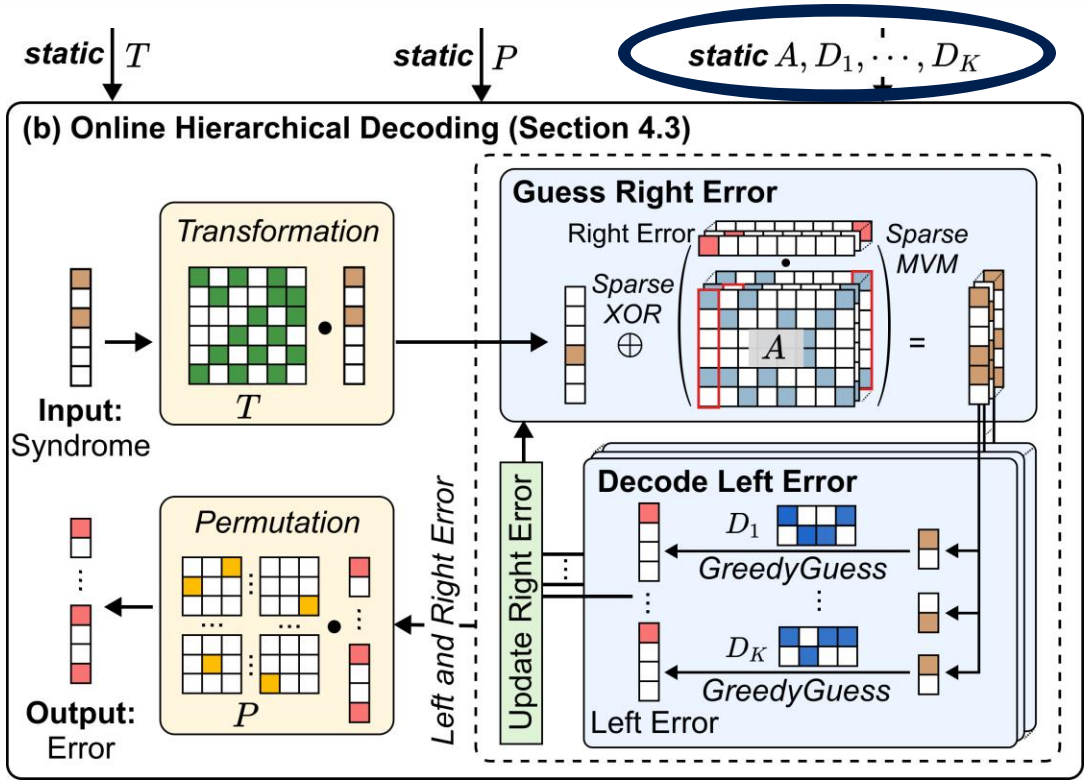
Therefore, we propose to partition the full error into ***left error*** and ***right error*** accordingly:

Left Error l + Right Error r

① Guess Right Error

How to compute the syndrome for the left error?

Online Hierarchical Decoding



The check matrix is decomposed into two parts:

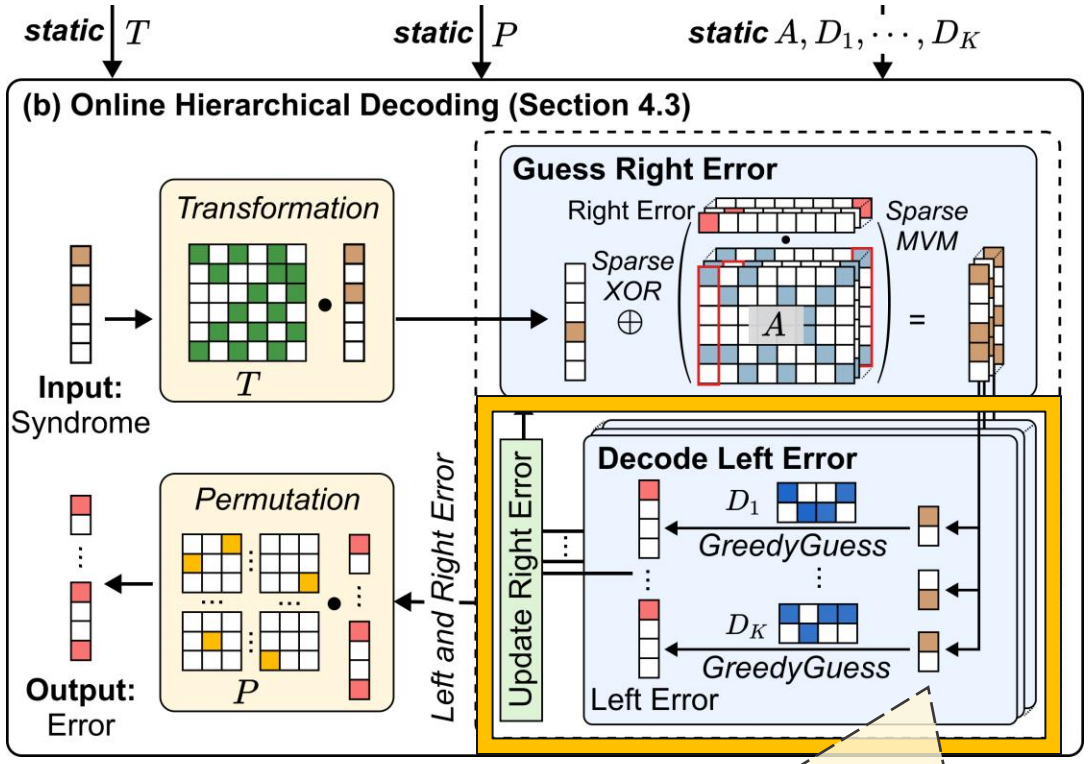
$$\mathrm{diag}(D_1, D_2, \cdots, D_K) + A$$

Therefore, we propose to partition the full error into ***left error*** and ***right error*** accordingly:

Left Error l + Right Error r ① Guess Right Error

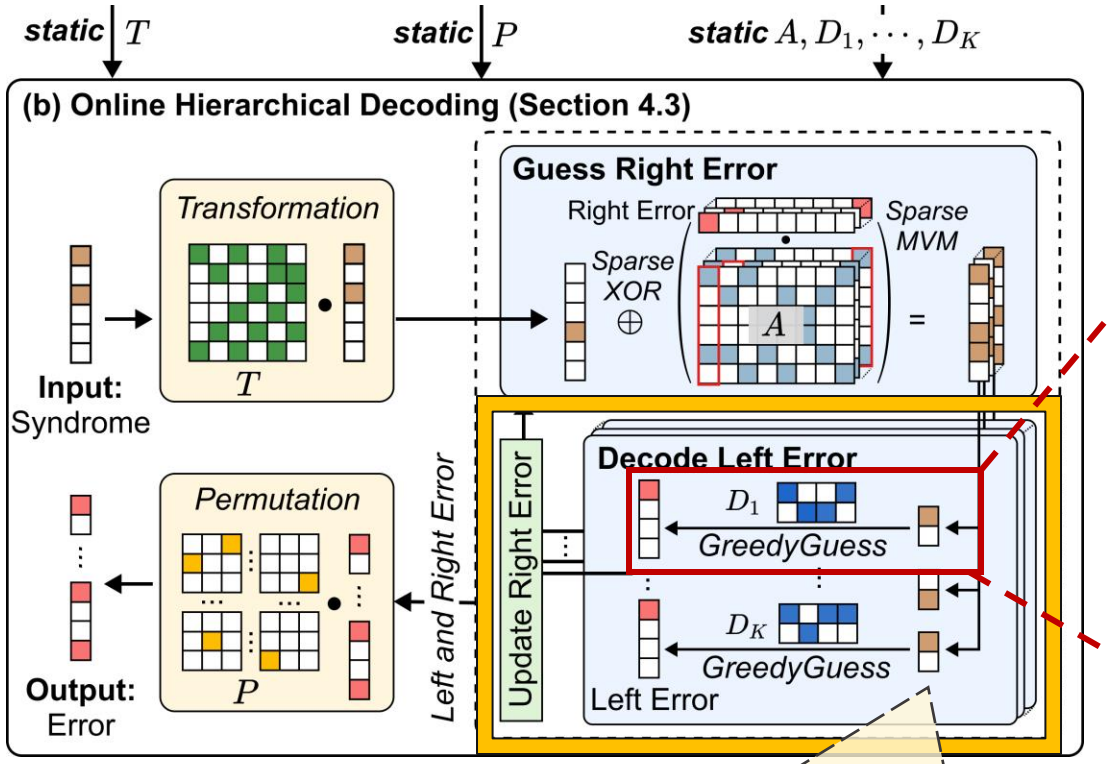
③ Left-part Syndrome s_l = Transformed Syndrome $\oplus s_r$ = $A \cdot r$ s_r ② Right-part Syndrome

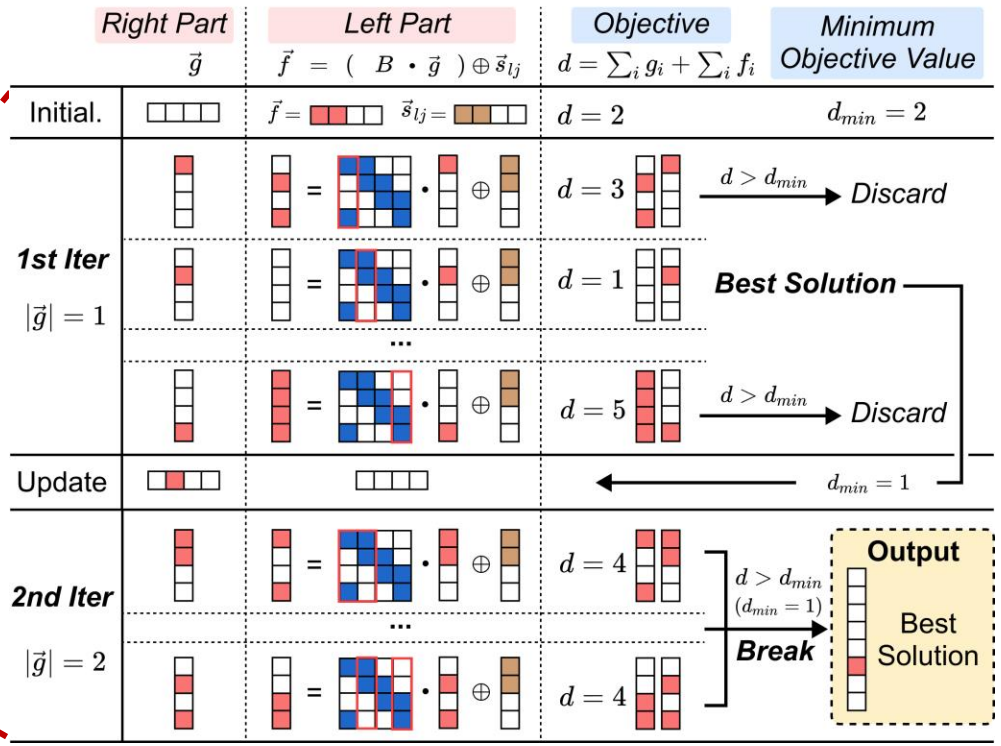
Online Hierarchical Decoding



Decoding procedures for $D_1, \ldots, D_K$ are executed parallelly.

Online Hierarchical Decoding





Decoding procedures for $D_1, \ldots, D_K$ are executed parallelly.

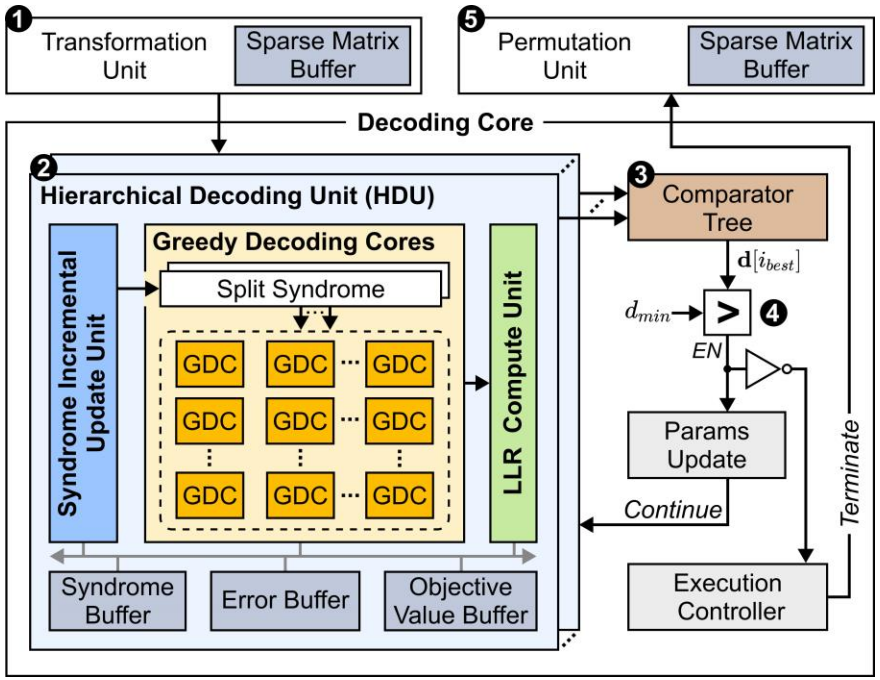
- **1st Iter:** The optimal solution with objective value d=1 is selected as the initial error for the next iteration.
- **2nd Iter:** No better solution than d=1 is found, so the ***GreedyGuess*** algorithm terminates.

Outline

- Background
- Motivation
- Vegapunk Algorithm
- **Vegapunk Accelerator**
- Experiment

Accelerator Architecture

Accelerator Dataflow

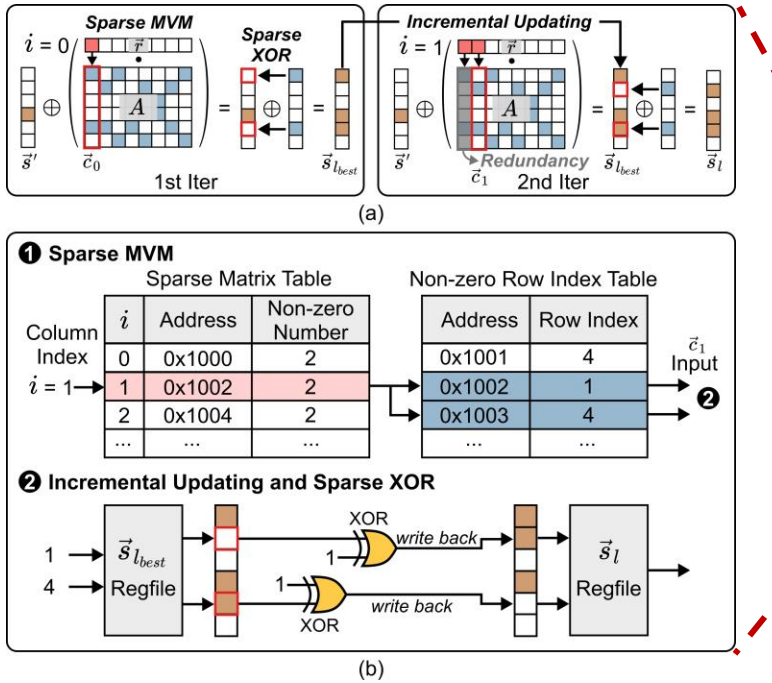


Customized design based on the online hierarchical algorithm.

- The accelerator consists of a decoding core, a transformation unit, and a permutation unit.
- The execution controller determines whether the decoding should continue.

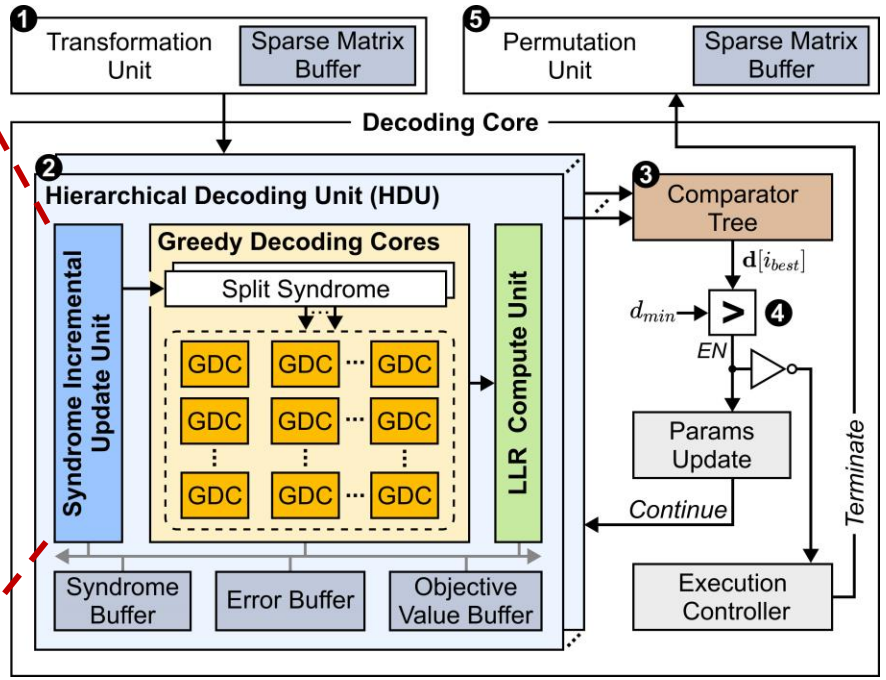
Accelerator Architecture

Syndrome Update Unit



- **Sparse MVM** and **sparse XOR** operations are employed.
- **Incremental updating** greatly reduces redundant computations during decoding.

Accelerator Dataflow

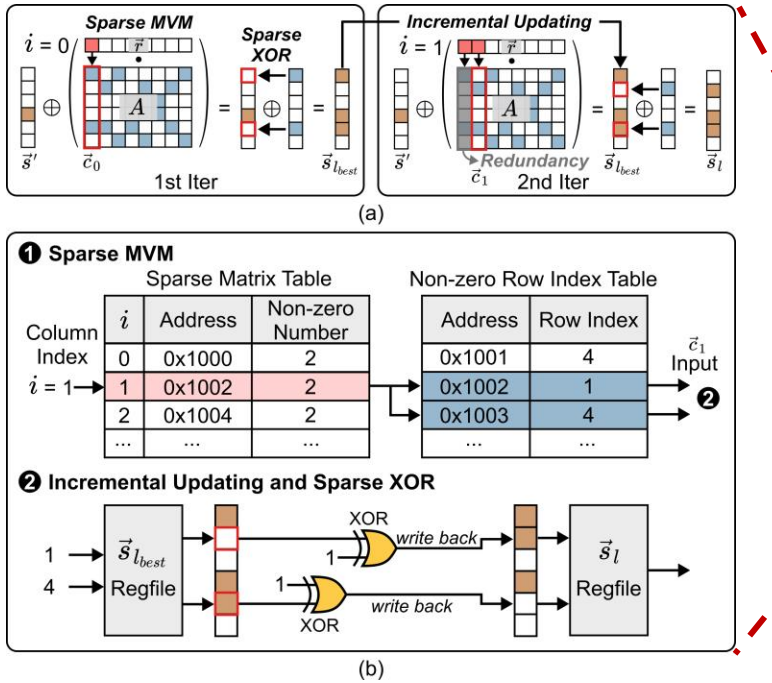


- The accelerator consists of a decoding core, a transformation unit, and a permutation unit.
- The execution controller determines whether the decoding should continue.

Customized design based on the online hierarchical algorithm.

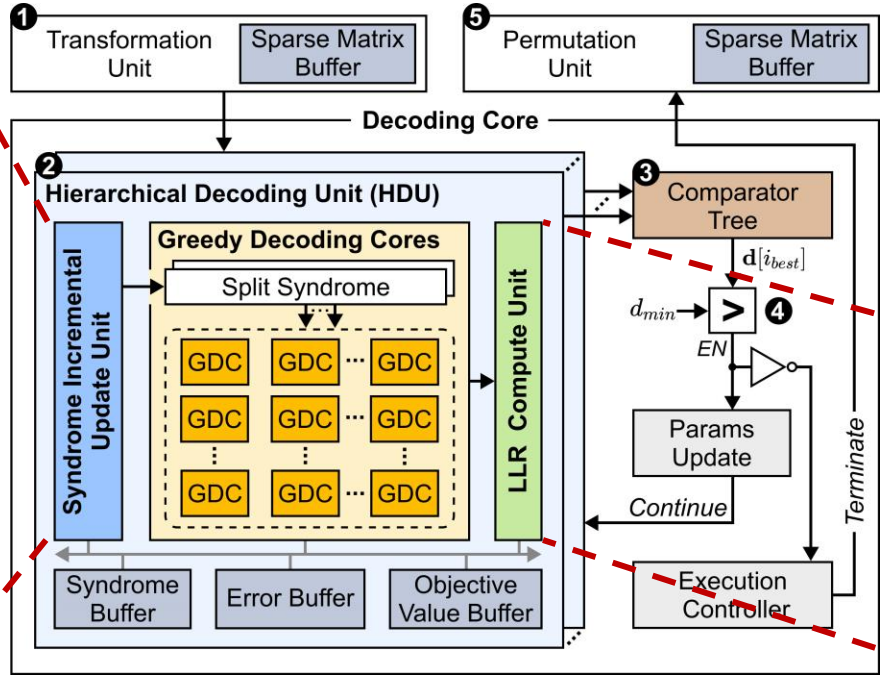
Accelerator Architecture

Syndrome Update Unit



- **Sparse MVM** and **sparse XOR** operations are employed.
- **Incremental updating** greatly reduces redundant computations during decoding.

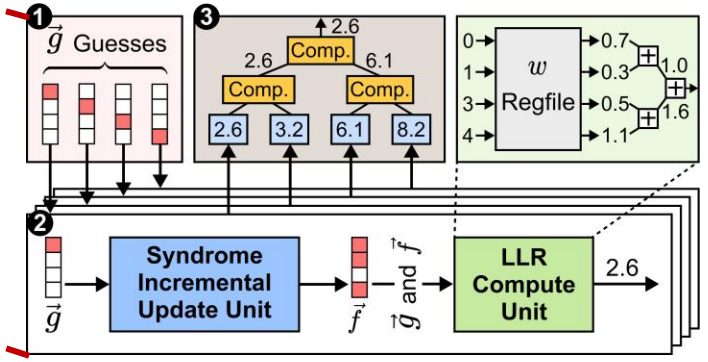
Accelerator Dataflow



- The accelerator consists of a decoding core, a transformation unit, and a permutation unit.
- The execution controller determines whether the decoding should continue.

Customized design based on the online hierarchical algorithm.

LLR Compute Unit



- The adder tree replaces floating-point multiplications for **fast objective value computation**.
- The comparison tree **efficiently identifies** the minimum value.

Outline

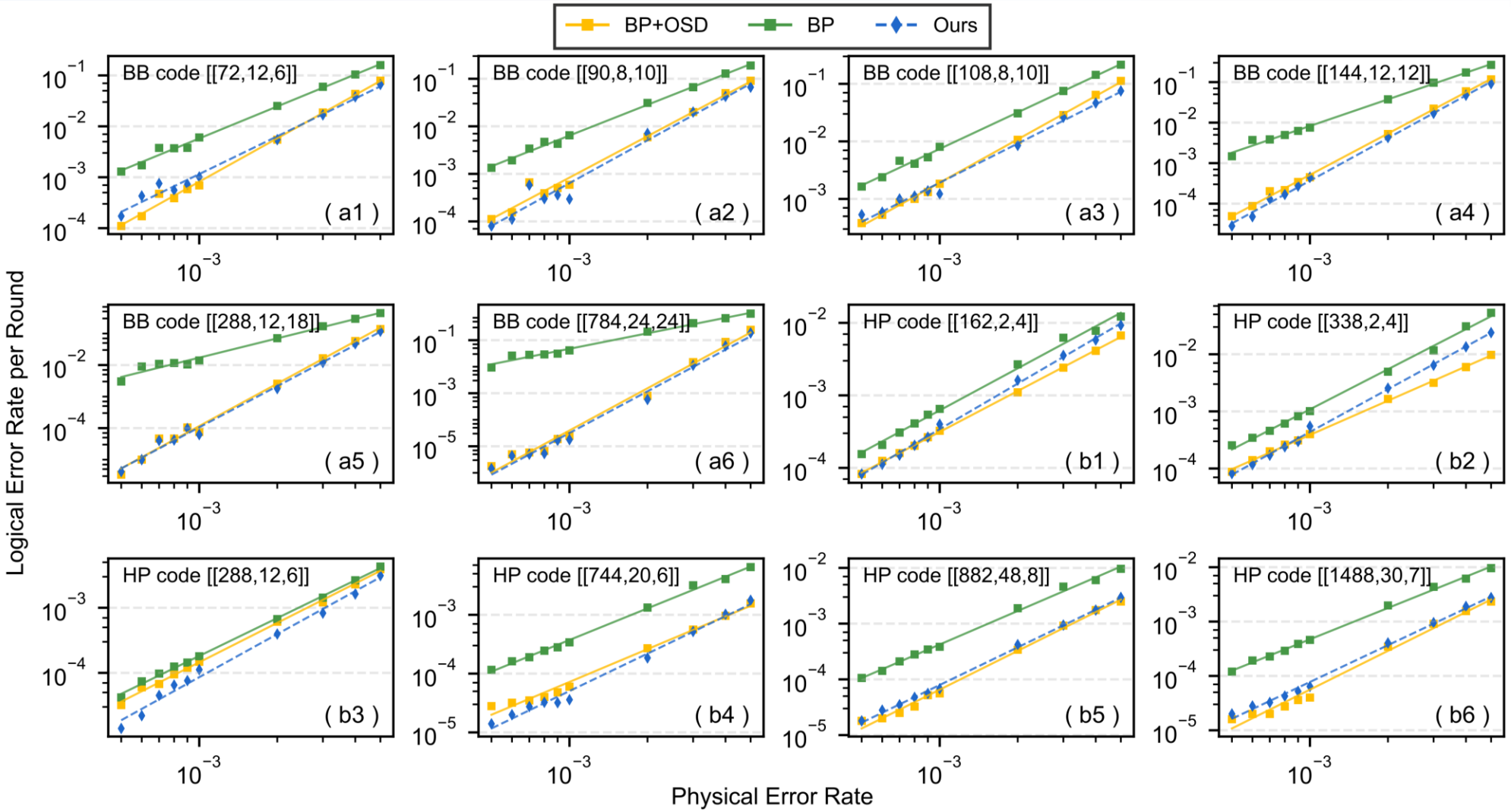
- Background
- Motivation
- Vegapunk Algorithm
- Vegapunk Accelerator
- **Experiment**

Evaluation Setup

- **Baselines:**
- BP[1], BP+OSD[2] (CS(7) version), BP+LSD[3], and BPGD[4].
- **Benchmarks:**
- (1) Bivariate Bicycle (BB) codes; (2) Hypergraph Product (HP) codes.
- **Implementation:**
- Vegapunk: High-Level Synthesis (HLS) C++, Xilinx Alveo U50 FPGA at 250 MHz.
- **Noise Model:**
- BB code: circuit-level noise with initial errors, gate errors, measurement errors, and reset errors.
- HP codes: phenomenological noise with bit-flip errors and measurement errors.
- **Evaluation Metrics:**
- (1) Logical Error Rate (LER) per round p_L ; (2) latency; (3) accuracy threshold p_T.
- We calculate accuracy threshold p_T by $\ln p_L = k \ln p + (1-k) \ln p_T$, where p is physical error rate and k is constant parameter. A higher accuracy threshold indicates greater tolerance to physical error rates.

[1] Pearl, Judea. "Reverend Bayes on inference engines: A distributed hierarchical approach." 2022.
[2] Fossorier, et al. "Soft-decision decoding of linear block codes based on ordered statistics." 2002.
[3] Hillmann, et al. "Localized statistics decoding: A parallel decoding algorithm for qLDPC codes." 2024.
[4] Yao, Hanwen, et al. "Belief propagation decoding of qLDPC codes with guided decimation." 2024.

Logical Error Rate (LER)



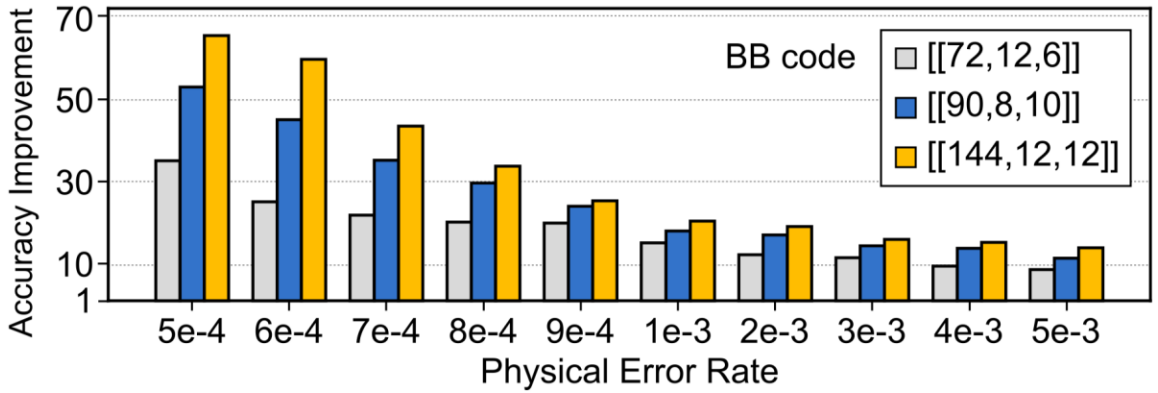
Vegapunk achieves decoding accuracy comparable to BP+OSD across 12 qLDPC codes.

Decoupled Result and Decoding Performance

Code Type	Code Notation	Check Matrix Shape	Decoupled Matrices		Accuracy Threshold (%)			Latency per Round		
			A shape (Spars.*)	D_i shape (Spars.*)	BP	BP+OSD-CS(7)	Vegapunk	BP	BP+OSD-CS(7)	Vegapunk
BB Codes	[[72,12,6]]	[36,360]	[36,108] (6)	[6,42] (3)	0.020	0.112	0.091	694ns	0.98ms	720ns
	[[90,8,10]]	[45,450]	[45,135] (6)	[9,63] (3)	0.020	0.110	0.125	1104ns	2.02ms	732ns
	[[108,8,10]]	[54,540]	[54,162] (6)	[6,42] (3)	0.016	0.065	0.060	1400ns	3.03ms	732ns
	[[144,12,12]]	[72,720]	[72,216] (6)	[6,42] (3)	0.017	0.132	0.149	2387ns	6.77ms	732ns
	[[288,12,18]]	[144,1440]	[144,432] (6)	[12,84] (3)	0.006	0.186	0.196	7009ns	38.6ms	780ns
	[[784,24,24]]	[392,3920]	[392,1176] (6)	[28,196] (3)	0.002	0.213	0.227	25881ns	449ms	840ns
Average	-	-	-	-	**0.013**	**0.136**	**0.141**	**6412ns**	**83.4ms**	**756ns**
HP Codes	[[162,2,4]]	[81,243]	[81,81] (2)	[9,18] (2)	0.167	0.377	0.268	151ns	0.18ms	264ns
	[[338,2,4]]	[169,507]	[169,169] (2)	[13,26] (2)	0.094	0.261	0.175	282ns	0.43ms	276ns
	[[288,12,6]]	[144,432]	[144,144] (4)	[12,24] (4)	0.622	0.674	0.777	72ns	0.19ms	432ns
	[[744,20,6]]	[372,1116]	[372,372] (4)	[31,62] (4)	0.350	2.176	1.455	578ns	1.93ms	480ns
	[[882,48,8]]	[441,1323]	[441,441] (5)	[63,126] (3)	0.237	0.798	0.768	735ns	6.66ms	526ns
	[[1488,30,7]]	[744,2232]	[744,744] (4)	[31,62] (4)	0.228	0.819	0.775	1875ns	11.7ms	492ns
Average	-	-	-	-	**0.283**	**0.851**	**0.703**	**616ns**	**3.52ms**	**412ns**

Ablation Study

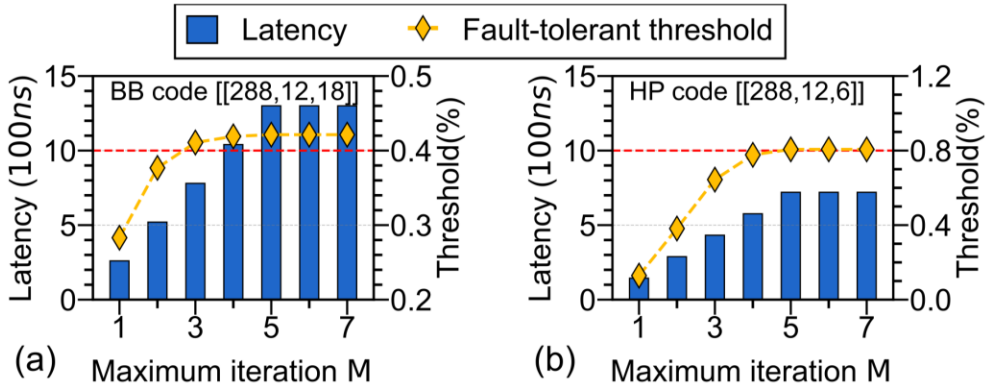
Offline Decoupling Strategy



Result: On three BB codes, the offline decoupling strategy yields accuracy improvements of **17.9×, 26.1×, and 31.1×**, respectively.

Analysis: The decoupling strategy partitions the check matrix into smaller submatrices, which not only **mitigates quantum degeneracy** but **also reduces the search space for greedy decoding**.

Maximum Iteration M

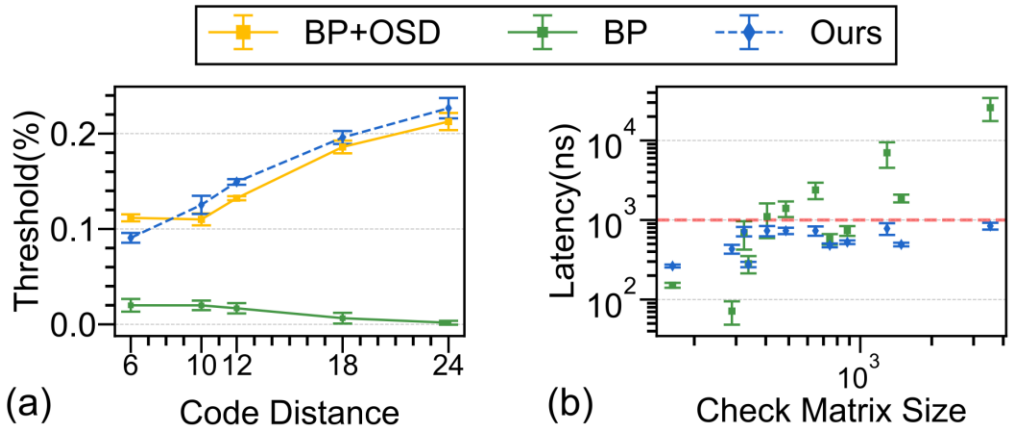


Result: As M increases, both decoding latency and accuracy threshold improve, with **M=3 offering the best trade-off**.

Analysis: A larger M expands the greedy decoding's search space, enhancing accuracy to some extent **but also increasing latency, with diminishing returns.**

Scalability and Comparison with SOTA

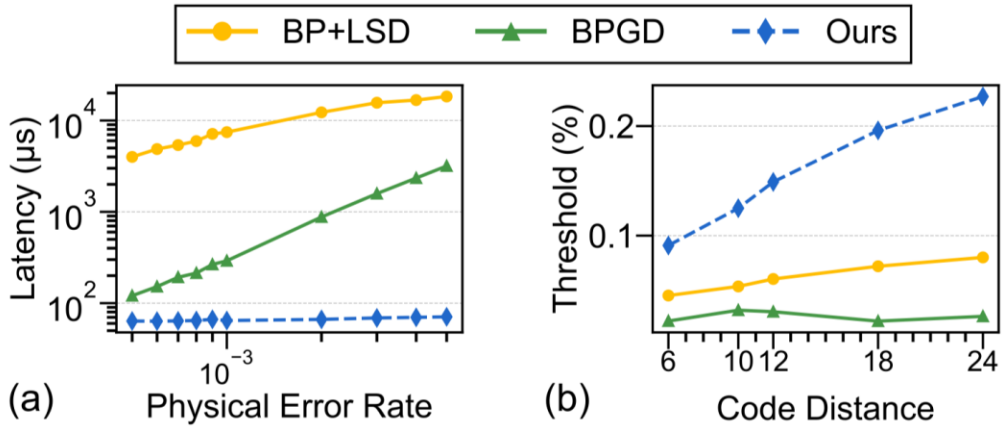
Scalability



Result: both Vegapunk and BP+OSD exhibit **increasing thresholds** as the code distance increases. In contrast, BP shows **a decreasing trend**.

Analysis: Vegapunk and BP+OSD can fully exploit the error correction capacity of qLDPC codes, while BP is unable to achieve **this due to its low decoding accuracy.**

Comparison with BP+LSD and BPGD



Result: Vegapunk achieves average speedups of **147.6×** and **13.9×**, and average accuracy threshold improvements of **2.53×** and **7.11×**.

Analysis: Vegapunk employs a **greedy decoding strategy** and **effectively leverages the sparsity** in decoupled check matrices. The higher accuracy stems from its offline decoupling strategy.

Conclusion

- **Problem:**
- Accurate and fast decoder for qLDPC codes in the era of fault-tolerant quantum computing.
- **Motivation:**
- Existing qLDPC decoders face two challenges: (1) low decoding accuracy and (2) high latency.
- **Solution: Vegapunk**
- (1) Offline SMT-optimized decoupling; (2) online hierarchical decoding; (3) customized accelerator.
- **Results:**
- Vegapunk achieves decoding accuracy on par with BP+OSD.
- Across 12 types of qLDPC codes, Vegapunk consistently achieves decoding latency below 1 μs.

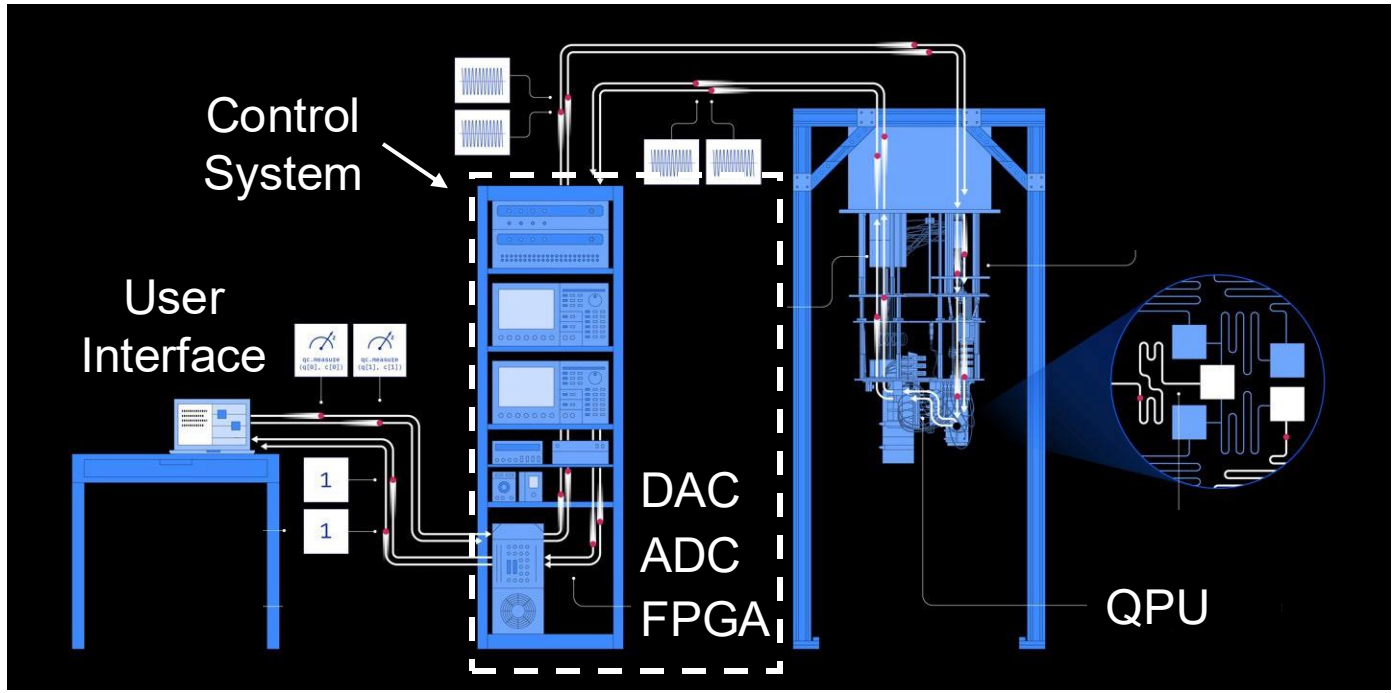
Thanks for Listening

Q & A

Backup Materials

2. Low Latency: Why Dedicated Accelerator?

- FPGA-based accelerators integrate naturally with superconducting quantum systems.



The need for low-latency feedback in QEC

Support

Seamless Integration with Quantum Hardware ✓

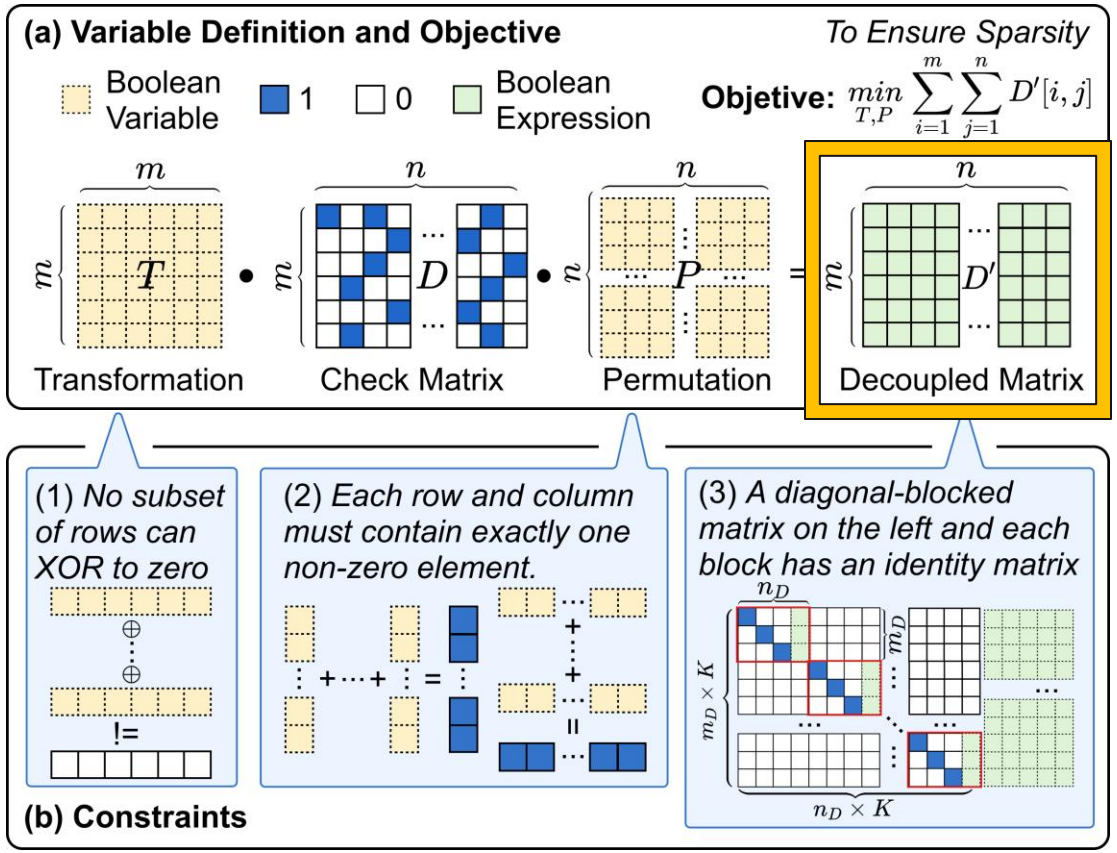
Low-Latency On-Chip Data Access ✓

FPGA

Provide

* The figure's source: IBM Research Quantum Experience

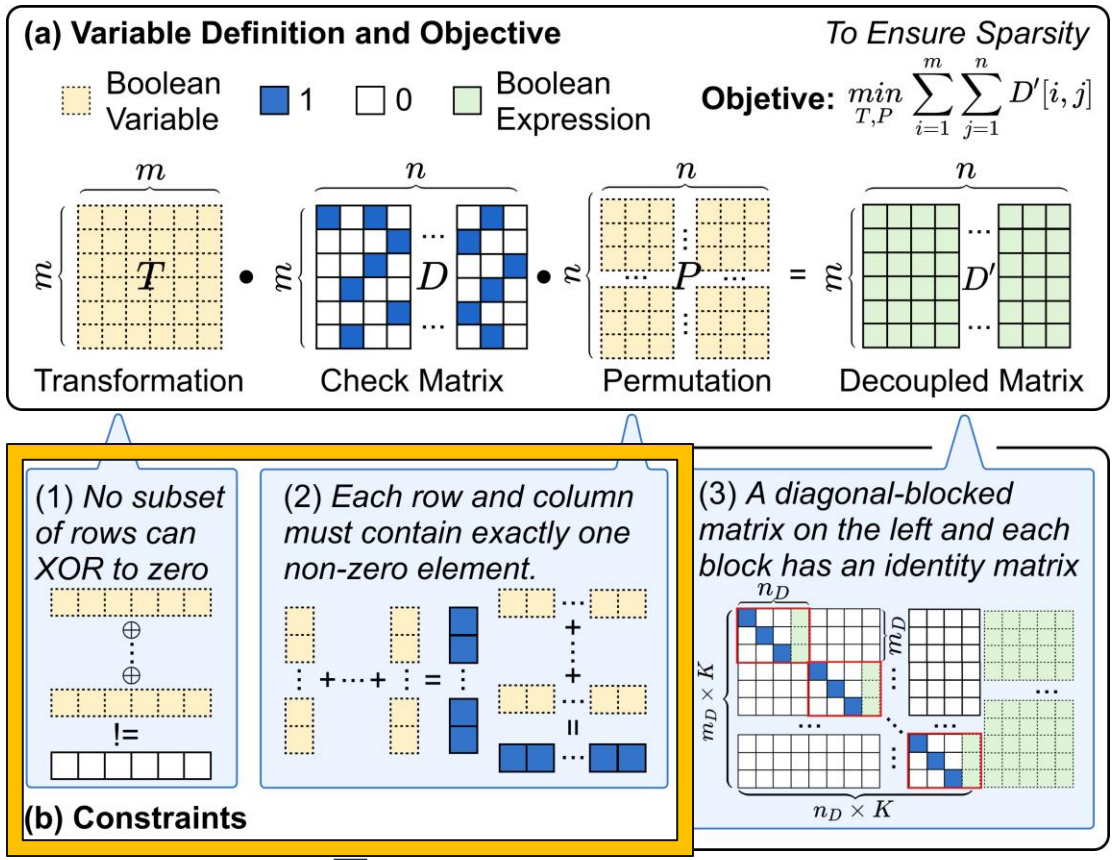
Offline SMT-Optimized Decoupling



Mathematical Formulation

Variables: $D'[i,j] = \sum_{q=1}^{n} \sum_{k=1}^{m} T[i,k]D[k,q]P[q,j]$

Offline SMT-Optimized Decoupling



Example

Mathematical Formulation

Variables: $D'[i,j] = \sum_{q=1}^{n}\sum_{k=1}^{m} T[i,k]D[k,q]P[q,j]$

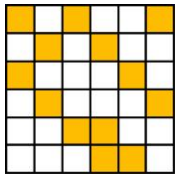
Constraints:

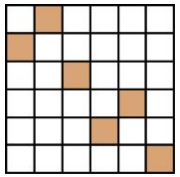
- Transformation matrix T:

$$\sum_{j=1}^{m} \oplus_{i\in\Delta} T[i,j] > 0,\ \forall \Delta \subseteq \{1,\cdots,m\}/\emptyset$$

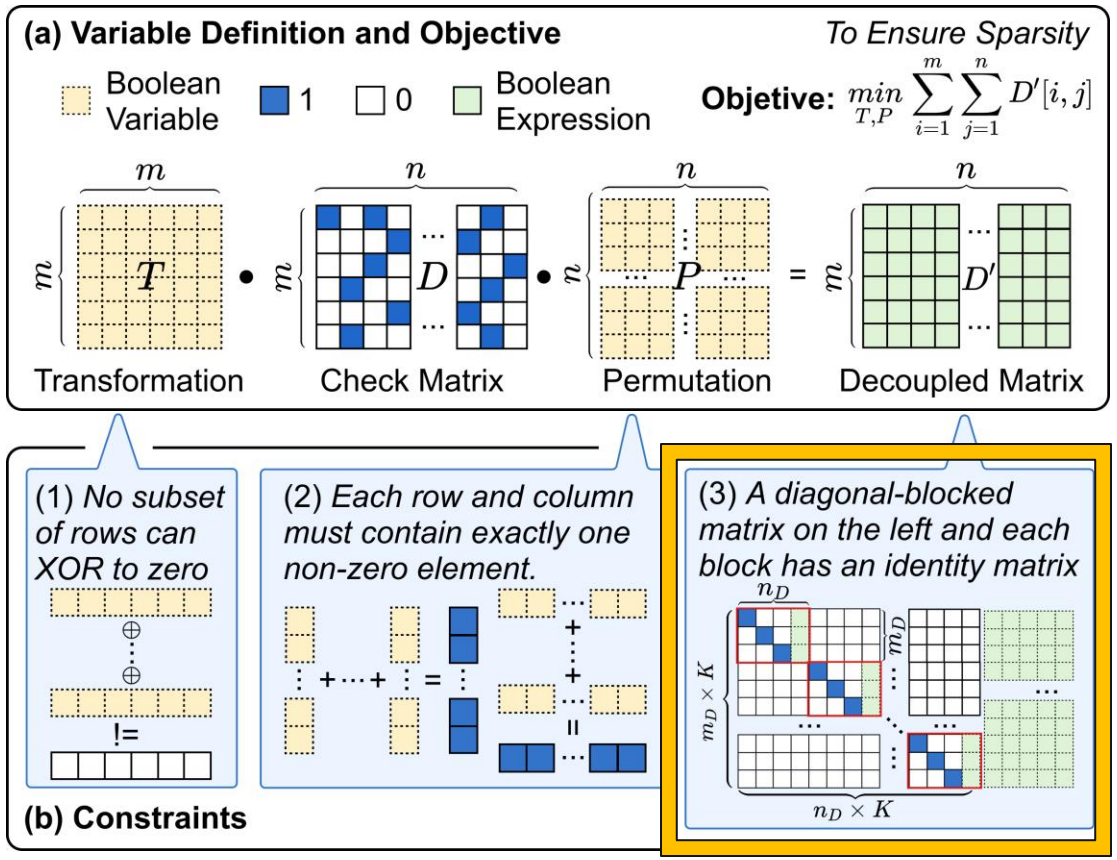
- Permutation matrix P:

$$\sum_{j=1}^{n} P[i,j] = 1,\ \forall i \in \{1,\cdots,n\},\quad \sum_{i=1}^{n} P[i,j] = 1,\ \forall j \in \{1,\cdots,n\}$$

Transformation matrix T 

Permutation matrix P 

Offline SMT-Optimized Decoupling



Mathematical Formulation

Constraints:

- Decoupled matrix D':

$$m_D \cdot K = m, m \le n_D \cdot K \le n$$

$$D'[i \cdot m_D + t, j \cdot n_D + k] = 0, \forall i, j(i \ne j) \in \{1, \cdots, K\}$$
$$\forall t \in \{1, \cdots, m_D\}, \forall k \in \{1, \cdots, n_D\}$$

$$D'[i \cdot m_D + t, i \cdot n_D + t] = 1, \forall i \in \{1, \cdots, K\}, \forall t \in \{1, \cdots, m_D\}$$
$$D'[i \cdot m_D + t, i \cdot n_D + k] = 0, \forall i \in \{1, \cdots, K\}\ \forall t, k(t \ne k) \in \{1, \cdots, m_D\}$$

Example