Overview for decoders

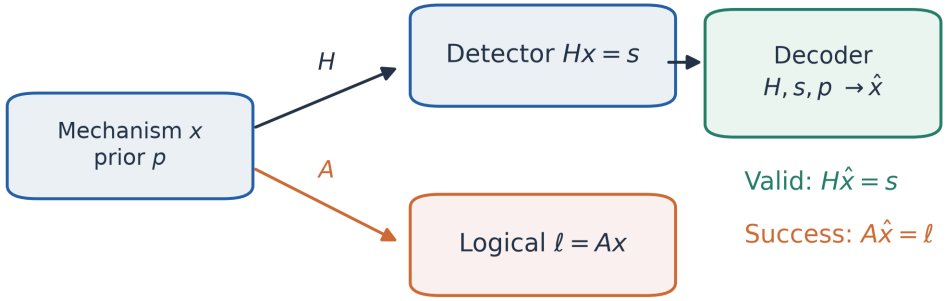
1. Decoder 的任务

- **mechanism**：候选错误机制及其二值变量，也称 error / variable。$x_i \in \{0,1\}$ 表示是否发生。x 是构型，p_i 是 prior probability。
- **detector**：$s_a \in \{0,1\}$ 表示是否触发。s 是取值向量，event 统一称为“触发的 detector”。
- **logical**：$\ell_\alpha \in \{0,1\}$ 表示是否翻转。ℓ 是 logical 翻转向量。

$$Hx = s, \qquad \ell = Ax \quad (\mathbb{F}_2).$$

- 输入 p, s, H。输出 $\hat{x}$，满足 $H\hat{x} = s$。
- **纠错成功**：$A\hat{x} = \ell$。

> H, A 是二元线性映射。detector 向量记为 s，code distance 记为 d。



All parity maps are over $\mathbb{F}_2$. Detector match and logical match are separate checks.

2. Noise models

- **Circuit-level**：线路中的 mechanism，如 idle Pauli Z、两比特门中的单 / 两比特误差。
- **Phenomenological**：只考虑 data / measurement mechanism。R 轮中的误差传播后归为这两类。

3. Objective · 固定 s，产生相同 ℓ 的 mechanism 构型属于同一 **logical class**。

近似最小权重构型，还是**概率最大的 logical class**？多数 MWPM / BP / beam search 关注前者。后者适于理论分析，但实现困难。

术语：mechanism / detector / logical。自绘示意图，保留大纲中的两类解码目标。

Minimum weight perfect matching

1. 优点与限制

- **特别快，能找到 weight 最低的解。**
- 一个 mechanism 最多影响两个 detectors：主要用于 surface code。

2. 从解码到 matching

1. 将每个 mechanism 看成连接 detectors 的 weighted edge。
2. Mechanisms 构成路径，触发的 detectors 位于路径端点，成对出现。
3. 给定一对 detectors，最似然的错误构型对应最短路径。
4. 预处理两两最短距离，得到 matching graph：将节点两两配对，使总 weight 最小。
5. 使用 Blossom 算法求解。

3. 时间复杂度

最短路预处理 + matching

- k：当前触发的 detectors 数量。
- n：detector graph 的节点总数。
- m：detector graph 的边总数。

$$O\big(k(m+n)\log n + k^3\big).$$

> 实际测量中，运行时间大致随节点数线性增长。

[1] Higgott and Gidney, 2025, arXiv:2303.15933.

Belief propagation：从 prior 到硬判决

1. Prior logit (Log likelihood ratio)

$$L_i = \log \frac{1 - p_i}{p_i}.$$

2. 初始化 mechanism → detector

$$q_{i\to a}^{(0)} = L_i.$$

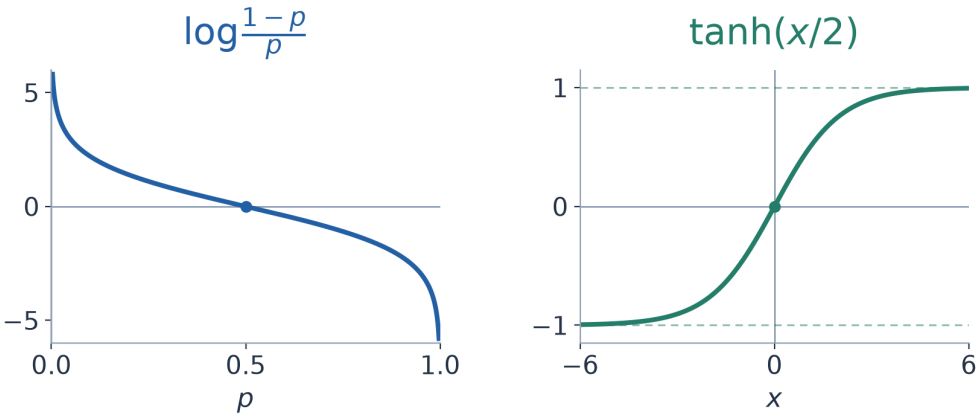
3. Detector → mechanism [1]

$$r_{a\to i}^{(t)} = 2\,\mathrm{atanh}\left[(-1)^{s_a} \prod_{j\in N(a)\setminus\{i\}} \tanh\left(\frac{q_{j\to a}^{(t-1)}}{2}\right)\right].$$

使用其它邻居 mechanism 的消息。

4. Mechanism → detector [1]

$$q_{i\to a}^{(t)} = L_i + \sum_{b\in N(i)\setminus\{a\}} r_{b\to i}^{(t)}.$$



5. Posterior logit

$$M_i = L_i + \sum_{a\in N(i)} r_{a\to i}.$$

6. 硬判决与检查

根据 M_i 得到 $\hat{x}_i$，检查：

$$H\hat{x} = s \pmod 2.$$

- 满足则输出。否则回到 **3**，直到最大迭代次数。
- 迭代间只继承 q。其余量现场计算。
- CSS code 分别得到 $\hat{x}_X, \hat{x}_Z$。

[1] Liu and Poulin, 2018, arXiv:1811.07835.

NeuralBP：学习消息强度与迭代反馈

沿用 sum-product BP 的标量消息。$k = k(i) \in \{\text{data}, \text{measurement}\}$；蓝色 标出相对普通 BP 的变化。

1. Prior 与初始化

$$L_i = \log \frac{1-p_i}{p_i},\ q_{i\to a}^{(0)} = M_i^{(0)} = L_i,\ r_{a\to i}^{(0)} = 0。$$

2. 原始 detector → mechanism 消息

$$\bar{r}_{a\to i}^{(t)} = 2\,\mathrm{atanh}\left[(-1)^{s_a} \prod_{j\in N(a)\setminus\{i\}} \tanh\left(\frac{q_{j\to a}^{(t-1)}}{2}\right)\right].$$

3. Check scale + damping

$$r_{a\to i}^{(t)} = \delta_{t,k} r_{a\to i}^{(t-1)} + (1-\delta_{t,k})\alpha_{t,k}\bar{r}_{a\to i}^{(t)}.$$

α：新消息的强度。δ：保留旧消息的比例。

4. Prior scale + memory

$$\Lambda_i^{(t)} = \beta_{t,k} L_i + \gamma_{t,k}\left(M_i^{(t-1)} - L_i\right).$$

β：prior 的强度。γ：历史偏移的反馈，可为负。

5. Extrinsic update 与 posterior

$$q_{i\to a}^{(t)} = \Lambda_i^{(t)} + \sum_{b\in N(i)\setminus\{a\}} r_{b\to i}^{(t)}.$$

$$M_i^{(t)} = \Lambda_i^{(t)} + \sum_{a\in N(i)} r_{a\to i}^{(t)}.$$

6. 输出软信息与硬判决

$$\tilde{p}_i = \mathrm{sigmoid}(-M_i),\ \hat{x}_i = \mathbf{1}[M_i < 0]。$$

参数共享：同一轮、同一类型共享，跨轮独立。**24 轮 × 2 类型 × 4 系数 = 192 个参数**。普通 BP 对应 $\alpha = \beta = 1$、$\delta = \gamma = 0$。

本项目：`neuralbp_affine/src/analytic_search_proposal.py`，24 轮 analytic NeuralBP。

Post-processing for BP：OSD / LSD

BP 的后验概率 $\widetilde{p}$ 提供软信息，但硬判决未必满足 $H\hat{x} = s$。后处理寻找满足约束的 mechanism 构型。

1. OSD：全局求解 [1]

$r = \mathrm{rank}(H)$，n 为 mechanism 总数，$k = n - r$。

OSD-0

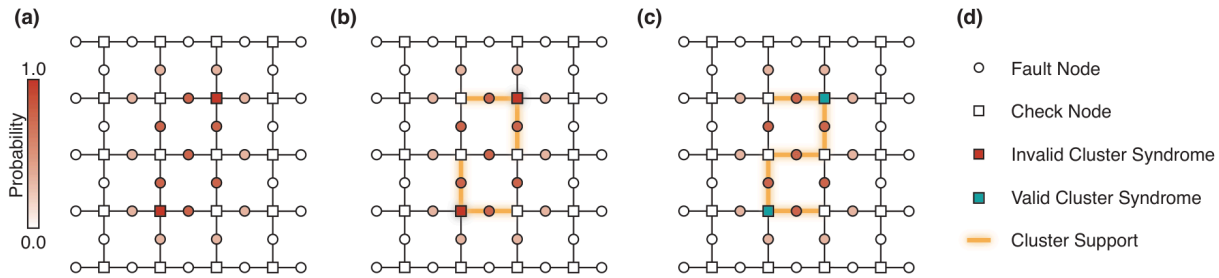
1. 按 $\widetilde{p}_i$ **从大到小**排列 H 的列。
2. 依次选出 r 个线性无关列，相关列跳过。
3. 其余 k 个 mechanism 设为 0。
4. 高斯消元求解基中的 mechanism，解唯一且满足约束。

高阶 OSD：在基外搜索至多 w 个非零 bits。每次重新求解基中的 mechanism，选择 weight 最低的候选。

$$N_{\mathrm{cand}} = \sum_{j=0}^{w} \binom{k}{j}.$$

2. LSD：局部 cluster [2]

- 将求解限制到部分行组成的 cluster。
- 每个触发的 detector 初始化一个 cluster。
- 初始 cluster 无法解释全部 detectors，需要加入 mechanism。
- 加入相连且后验概率最大的边。clusters 相连则合并。
- **LSD-0**：自洽、解释所有触发的 detectors 后停止。
- **LSD-μ**：合法 clusters 上再做**至多 μ 次额外增长**，再进行局部 OSD。



LSD cluster 生长，原论文 Fig. 2。[2]

[1] Roffe et al., 2020, arXiv:2005.07016. [2] Hillmann et al., 2025, arXiv:2406.18655.

BP 新进展：RelayBP / BF-OSD

RelayBP：类似退火，避免卡在不满足 syndrome 的点

1. Prior：$L_i = \log \frac{1-p_i}{p_i}$。**2. 初始化**：$q_{i\to a}^{(0)} = M_i^{(0)} = L_i$。

每轮先更新 dynamic bias：$\boldsymbol{\Lambda_i^{(t)} = (1-\gamma_i)L_i + \gamma_i M_i^{(t-1)}}$。

3. Detector → mechanism（MinSum）：

$$r_{a\to i}^{(t)} = (-1)^{s_a} \prod_{j\in N(a)\setminus\{i\}} \operatorname{sgn}(q_{j\to a}^{(t-1)}) \min_{j\in N(a)\setminus\{i\}} |q_{j\to a}^{(t-1)}|.$$

4. Mechanism → detector：

$$q_{i\to a}^{(t)} = \Lambda_i^{(t)} + \sum_{b\in N(i)\setminus\{a\}} r_{b\to i}^{(t)}.$$

5. Posterior：

$$M_i^{(t)} = \Lambda_i^{(t)} + \sum_{a\in N(i)} r_{a\to i}^{(t)}.$$

6. 硬判决与检查：$\hat{x}_i = \mathbf{1}[M_i < 0]$，检查 $H\hat{x} = s$。

Relay：各 leg 更换 γ_i（可为负），继承 M_i；多解取 prior weight 最低者。

BF-OSD：似然度优先搜索 [2]

用 OSD-0 得到特解 x_0，消元得到 $\ker H$ 的基 g_j。所有满足 $Hx = s$ 的 x 都可以写成

$$x(T) = x_0 \oplus \bigoplus_{j\in T} g_j.$$

- 上述方法都是在 x 空间的邻域进行搜索。
- 在更加受限的 $\ker H$ 上搜索？BF-OSD 提供了这样一种方案。

$\mathcal{Q} \leftarrow \{\varnothing\}$，$x_{\text{best}} \leftarrow x_0$
while $\mathcal{Q} \neq \varnothing$ 且 query budget 未耗尽：
  $T \leftarrow \text{pop_max}_\ell(\mathcal{Q})$，消耗一次 query
  若 $x(T)$ 评分更高，则 $x_{\text{best}} \leftarrow x(T)$
  for $j > \max T$（$\max \varnothing = 0$）：
    $T' \leftarrow T \cup \{j\}$
    按 $\ell(T')$ 将 T' 加入 priority queue $\mathcal{Q}$
return x_{best}

[1] Müller et al., 2025, arXiv:2506.01779. [2] Banfi et al., 2026, arXiv:2605.25777.

SOTA：Tesseract

慢，但目前 logical 翻转率最低的 decoder。融合 **A*** 和 **beam search**。[1]

1. 搜索状态、代价与扩展

- F：当前选择的 mechanism 集合。$s(F)$：尚未解释的 detector 向量。
- $s(F) = 0$ 表示已解释全部观测，寻找代价最低的状态：

$$\min_{F:\, s(F)=0} g(F), \qquad g(F) = \sum_{j \in F} \log \frac{1-p_j}{p_j}.$$

- 从 $F = \varnothing$ 开始，每次加入一个 mechanism j：

$$F' = F \cup \{j\},$$
$$s(F') = s(F) + H_{\cdot j} \pmod 2,$$
$$g(F') = g(F) + \log \frac{1-p_j}{p_j}.$$

- 只扩展与**编号最小的剩余 detector**相连的 mechanism。用顺序规则去重。

2. A*：决定先搜谁

$$f(F) = g(F) + h(F).$$

- g：已经加入的 mechanism 的代价。
- h：解释剩余 detectors 所需代价的下界估计。
- 将剩余 detector 的估计修复代价相加。先探索 f 小的状态。

3. Beam：截断偏离进展的状态

$r(F) = \mathrm{wt}(s(F))$，$r_{\min}$ 为已访问状态中最小的 r。

$$r(F) > r_{\min} + B \quad \Rightarrow \quad \text{跳过 } F.$$

> A*决定搜索的方向，beam widths B决定搜索的宽度

[1] Beni et al., 2025, arXiv:2503.10988.

Recursive Transformer

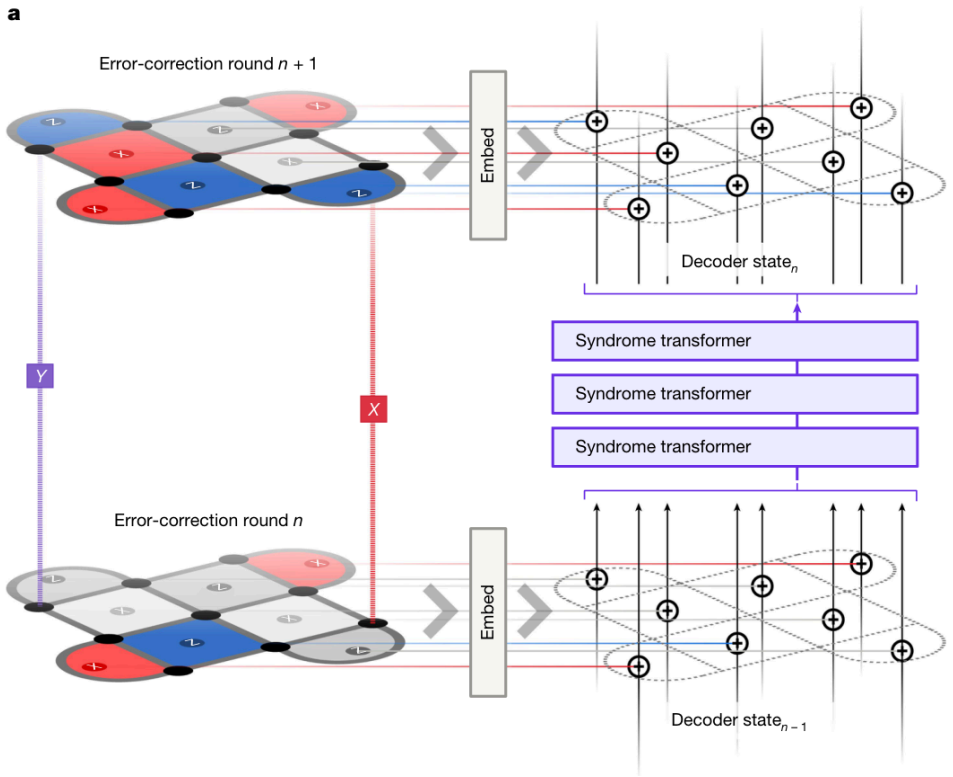
1. Transformer 的三个组成部分

- **Embedding**：将 detector 与 data nodes 映射为特征向量。
- **Attention**：a 的 query 与 b 的 key 做内积，决定 a 对 b 的 attention。据此把 b 的 value 信息加入 a。
- **MLP**：每个 node 独立经过简单网络演化。

> Query / key 决定关联。value 承载被加入的信息，与 node 的特征向量区分。

2. QEC 中的空间与时间处理

- 空间 attention 一般局限于同一 cycle。
- **AlphaQubit**：将 detector 信息映射成向量，加入hidden vector。[1]
- **GraphQEC**：时间方向也是Transformer，使用 linear attention。[2]
- Recurrent hidden state 将上一 cycle 的历史信息传给当前 cycle，逐轮处理 detector 数据。



AlphaQubit recurrent architecture，Fig. 2a。[1]

3. Readout

- 从 mechanism 上的 hidden vectors readout 出 **logical 翻转**。
- 目前未看到直接 readout **mechanism 构型**的 transformer decoder。

[1] Bausch et al., 2024. [2] Hu et al., 2025, arXiv:2502.19971.

CNN/GNN: Cascade

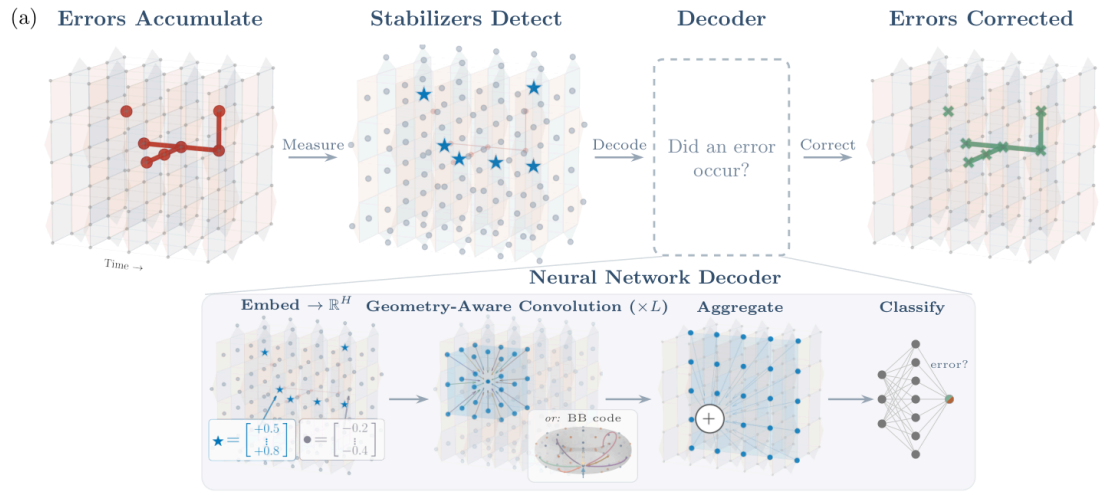
1. 图结构与 embedding [1]

- Surface code 用 CNN，nonlocal LDPC 用 GNN。Tanner graph 的邻接关系决定卷积的邻居集合。
- Spacetime detector $v = (c, t)$ 的取值 s_v embedding 为 $h_v^{(0)} \in \mathbb{R}^H$。

2. L 个 bottleneck residual blocks

$$\begin{aligned}&\mathrm{BN{+}SiLU} \to (H \to H/4) \to \\ &\mathrm{BN{+}SiLU} \to \mathrm{convolution} \to \\ &\mathrm{BN{+}SiLU} \to (H/4 \to H) \to \mathrm{residual}.\end{aligned}$$

- $H \to H/4$ 由投影矩阵实现。
- **BN** = batch norm：在样本层面归一化权重矩阵。特征各维仍有尺度差异。
- **Residual**：将本层结果加回上一层。



Cascade architecture，Fig. 1a。[1]

3. 卷积与权重共享

$$h'_v = W_{\mathrm{self}} h_v + \sum_{u \in \mathcal{N}(v)} W_{\delta(u,v)} h_u.$$

- $\delta(u, v)$ 表示几何 / 连接关系。同一 relation type 共享 W_δ，每条边没有独立矩阵。
- 每层感受野扩大常数量级，需 $O(d)$ 层局域传播。

4. Readout

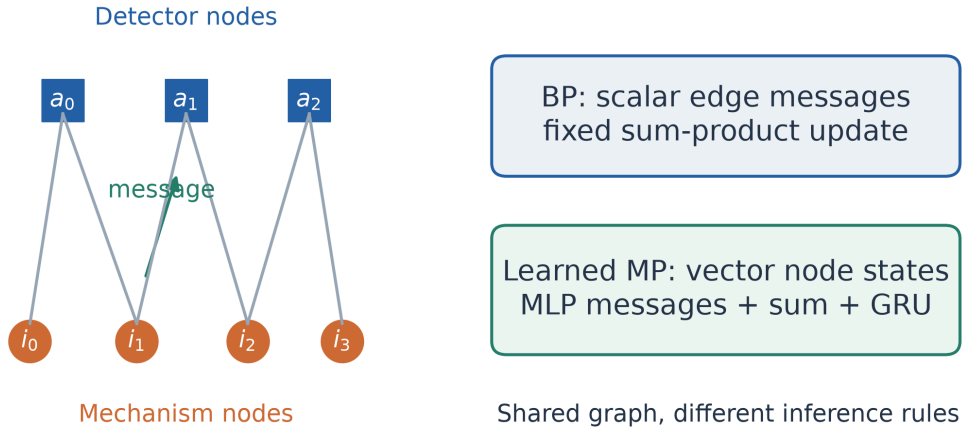
Detector representations → data qubits → logical support pooling → MLP 输出 logical 翻转概率。

[1] Gu et al., 2026, arXiv:2604.08358.

CNN/GNN: Machine Learning Message-Passing

1. 从 BP 到 message passing

- BP 是一种特化的 message passing。图神经网络已有成熟理论。
- 每个节点维护特征向量 h_i^t，通过更新网络传播信息。



相同图结构：BP 标量消息 / learned MP 向量状态。

初始化（z_j 是节点的初始输入）

- Mechanism：$z_i = 0$，$h_i^{(0)} = 0$。
- Detector：$z_a = s_a$，用于初始化 $h_a^{(0)}$。

2. 三个基本函数 [1]

Message：由发送 / 接收节点的状态决定消息。

$$m_{ij}^t = f(h_i^{t-1}, h_j^{t-1}).$$

$$[h_i^{t-1} \| h_j^{t-1}] \to \mathrm{MLP}_f \to m_{ij}^t.$$

Node update：收集邻居消息，更新特征向量。

$$h_j^t = g(h_j^{t-1}; m_{i_1 j}^t, m_{i_2 j}^t, \ldots).$$

先加总 $m_j^t = \sum_{i \in N(j)} m_{ij}^t$，再用 **GRU**（Gated Recurrent Unit，门控循环单元）更新：

$$(h_j^{t-1}, z_j, m_j^t) \to \mathrm{GRU} \to h_j^t.$$

Readout：由当前特征向量通过 MLP 得到输出。

$$o_i^t = r(h_i^t).$$

[1] Maan and Paler, 2025.

Why do we need online decoding?

1. 平台决定吞吐速度要求

两比特门时间、门数和并行度决定物理 cycle。长期运行需 **decoder 均摊时间 $<$ 物理 cycle 时间**。

超导平台

- Gross $[[144, 12, 12]]$：144 data + 144 measurement qubits。144 个 weight-6 checks，共 **864 个两比特门**。[1]
- X、Z 分组的 12-layer 示例：每层 72 个并行门。
- Google 实测 $1.1\,\mu s$/cycle，作为超导量级参考。[2]

离子阱：Quantinuum Helios

原生 $R_{ZZ}(\pi/2)$ 约 $70\,\mu s$ [3]。1 个 CNOT 仅计 1 个原生纠缠操作。

并行假设	Gross code：只计两比特操作
充分并行：7 层，峰值 144 个门	$7 \times 70\,\mu s = \mathbf{0.49\,ms}$
完全串行	$864 \times 70\,\mu s = \mathbf{60.48\,ms}$

中性原子：代表性架构预算 $O(1\,ms)$/cycle。[4]

2. 软件层面的延迟

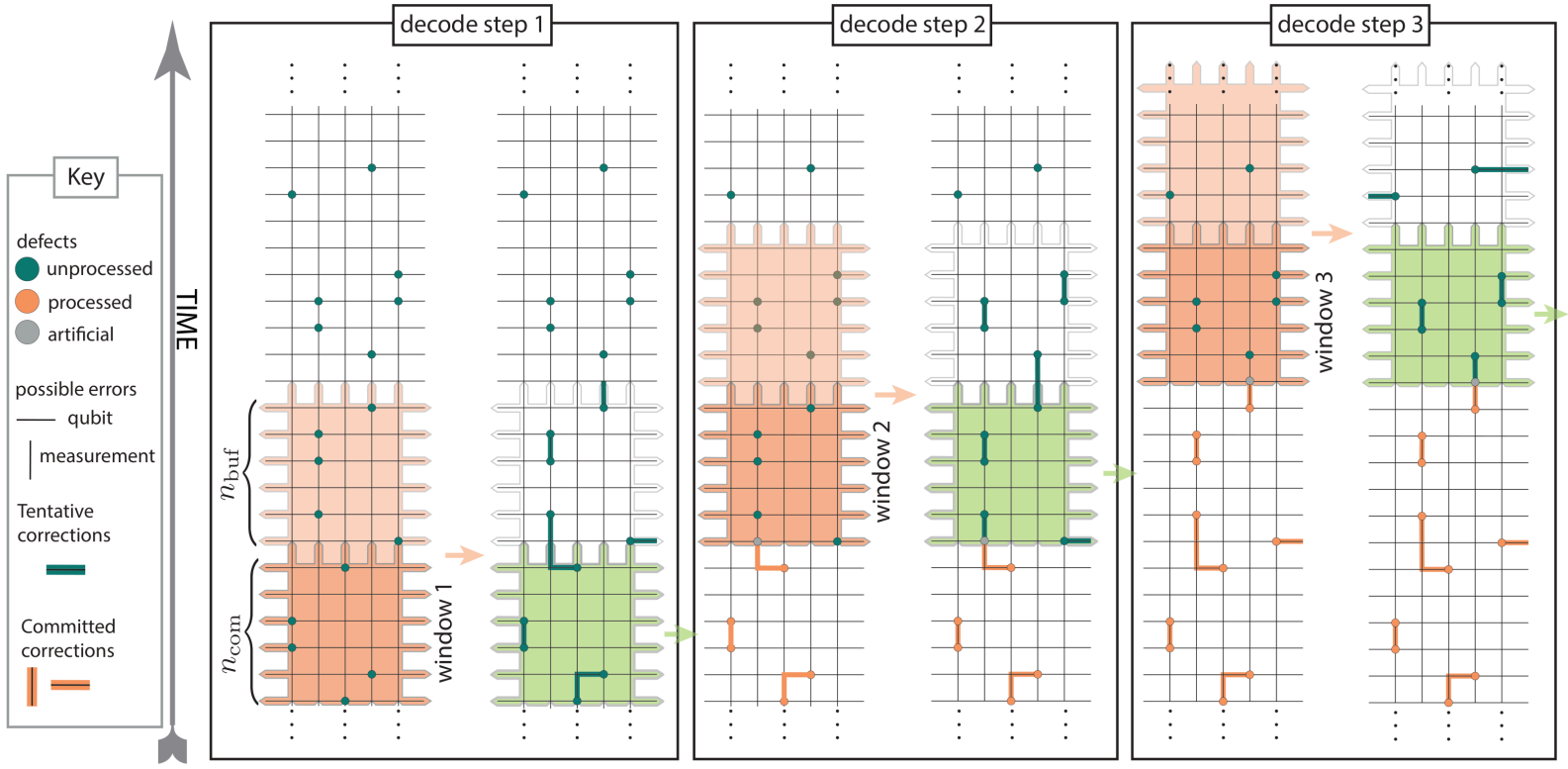
- 有测量误差时，一轮纠错不够，除非 **single-shot QEC**。多轮纠错的最后一轮同理。
- 多轮纠错的意义：准确知道最后一个 cycle 之前约 $d/2$ 轮发生了什么，而非马上知道所有 cycles 的 Pauli frame。
- 所以我们需要等待额外的 $d/2$ 轮才能知道当前的 Pauli frame

> 既要跟上 detector 数据的产生速度，也要在有限延迟内持续输出**可提交的 Pauli frame 更新**。

[1] Bravyi et al., 2024, arXiv:2308.07915. [2] Acharya et al., 2024. [3] Ransford et al., 2026, arXiv:2511.05465. [4] Sunami et al., 2025, arXiv:2407.11111.

Sliding window decoding

- **Sliding window** 是主流 online decoding 方法。
- 初始 boundary 的 Pauli frame 已知，取 $R = 2d$ 轮。
- 前 d 轮可信，由后续 d 轮排除测量错误。
- 只提交前 d 轮，再后移 d 轮，重复。



Sliding window：保留 commit region，由后续 buffer 支撑。原论文 Fig. 2。[1]

Parallel window：第一层并行提交各 window 的中间区域。第二层利用已提交结果作为 boundary，处理剩余区域。适合计算资源充足的情况。

[1] Skoric et al., 2023, arXiv:2209.08552.