面向6G超可靠低时延通信的自同构集成极化码译码器

Claus Kestel*、Marvin Geiselhart†、Lucas Johannsen*‡、
Stephan ten Brink† 和 Norbert Wehn*
* 凯泽斯劳滕大学，德国凯泽斯劳滕 67663 邮箱： {kestel,
wehn}@eit.uni-kl.de
† 斯图加特大学电信研究所，德国斯图加特 70569 邮箱： {geiselhart,tenbrink}
@inue.uni-stuttgart.de
‡ 科布伦茨应用科学大学，德国科布伦茨 56075 邮箱： {
johannsen}@hs-koblenz.de

arXiv:2303.01235v1 [cs.IT] 2 Mar 2023

摘要—**即将出台的6G标准中的超可靠低时延通信场景要求低时延和超可靠传输，也就是说，纠错性能需接近最大似然性能。尤其对于短码块长度，实现接近最大似然的性能极具挑战。极化码是一种很有前景的候选方案，并且已纳入5G标准。连续消除列表（SCL）译码算法能够提供出色的纠错性能，但代价是较高的计算译码复杂度，进而导致时延较长、面积效率和能效较低。近年来，自同构集成译码（AED）因提升纠错能力而备受关注。与SCL不同，AED会并行执行多次低复杂度（例如SC）译码。然而，AED能否与复杂的SCL译码器相竞争仍是一个悬而未决的问题，尤其是从先进硅工艺技术的实现角度来看。本文提出了一种经过深入设计的AED架构，采用先进的基于路径度量的候选项选择，以降低实现复杂度，并在12纳米FinFET工艺下将其与先进的SCL译码器进行比较。我们的AED实现与先进的SCL译码器相比，时延最高降低4.4倍，面积效率最高提升8.9倍，能效最高提升4.6倍，同时提供相同甚至更优的纠错性能。**

索引词—**极化码、自同构、集成译码、路径度量、专用集成电路、12 nm、实现**

I. 引言

超可靠低时延通信（URLLC）场景将成为 6G 的关键推动因素之一。与 5G 相比，它要求更低的时延，并特别要求提升纠错能力 [1]，使其接近最大似然（ML）性能。短码块长度有助于降低时延，但通常会以牺牲纠错能力为代价 [2]。因此，关键挑战之一是找到能够实现低时延、高能效，同时使纠错性能接近最大似然性能的码、译码算法和高效译码器架构。

极化码是这一应用中很有前景的候选方案。它们是首类被证明能够实现二进制对称无记忆信道（BSMC）信道容量的纠错码，并已被5G增强型移动宽带（eMBB）用例中的控制信道采用。极化码可以使用连续消除（SC）算法进行译码，该算法是一种低复杂度译码算法，但其纠错性能较差，尤其是在短码块长度下。因此，人们开发了更复杂的译码算法。连续消除列表（SCL）译码算法 [4]同时考虑多条译码路径，并根据路径度量选择其中最佳的一条。在列表大小足够大时 L，SCL可逼近最大似然性能，但代价是极高的计算复杂度。

近年来，猜测随机加性噪声译码（GRAND） [5] 算法受到了广泛关注。与上述算法不同，GRAND 是一种适用于任意码的通用译码算法。其基本思想是猜测噪声模式，以找到正确的码字。原则上，只要猜测次数足够多，GRAND 就能达到最大似然性能。但由于事先无法确定猜测次数，时延和吞吐量可能在最坏情况与平均情况之间有很大差异。 [6] 可限制这种差异，但代价是纠错性能下降。此外，研究表明，GRAND 算法及其衍生算法仅适用于高信噪比区域和高码率。

自同构集成译码（AED）是近期发表的另一种新型译码方法，可应用于任何具有自同构的码。自同构用于置换码字。随后，可以通过低复杂度译码算法并行译码不同的码字。虽然里德–穆勒（RM）码存在许多自同构 [7]，但为极化码寻找有用的自同构颇具挑战。近期，[8] 已证明极化码同样可以找到自同构，并且能够显著提升纠错性能。

从实现角度来看，AED 也很有吸引力。可以使用多个低复杂度译码器（例如 SC），而不是一个复杂的译码器（例如 SCL）。

作者感谢德国联邦教育与研究部通过“Open6GHub”项目（资助编号：16KISK019 和 16KISK004）提供的资金支持。

每个低复杂度译码器在时延、吞吐量、能效和面积效率方面都具有很高的实现效率。由于这些译码器并行运行且无需交换信息，局部性显著提升。这非常有利于提高实现效率，因此，面积效率和能效几乎随集成译码器实例的数量 M 呈线性增长，其中 M 由纠错性能要求决定。需要指出的是，AED 中的时延在一阶上独立于 M，而后者是高级译码所特有的。

本研究的创新贡献如下：

- 一种简单的、用于AED的数据驱动置换选择方法。
- 一种高效的AED算法，基于快速简化SC（SSC）算法和一种新的基于路径度量的候选项选择方法，可实现极高吞吐量和低时延。
- 在12纳米FinFET硅技术上，就纠错性能和硬件实现方面，对我们的AED架构与最先进的SCL译码器进行详细比较。据我们所知，这是首次实现AED，并将其与传统译码算法进行比较。

本文其余部分安排如下：我们在第二节介绍极化码及其译码算法所需的背景知识。第三节介绍AED的概念以及一种新的基于路径度量的候选项选择方法。第四节从纠错性能和实现成本方面，详细比较本文方法与SCL译码器，并在第五节总结全文。

II. 背景

A. 极化码

极化码 $\mathcal{P}(N,\ K)$ 是码长为 $N = 2^n$ 的线性分组码，可编码 K 信息比特。它利用极化现象生成 N 虚拟信道。随后，信息通过信息集 $\mathcal{I}$中列出的 K 可靠信道传输，而其余（不可靠的）信道被置为零，称为冻结比特（冻结集$\mathcal{F}$）[3]。本文始终采用以下记号：$\mathbf{x} \in \{0,1\}^N$ 表示码字，**$\hat{\mathbf{x}}$ </style}表示其译码后的估计值；$\mathbf{y}$** 表示接收信道对数似然比（对数似然比）的向量。

B. 连续消除译码

SC译码可描述为对极化因子树（PFT）[9]进行深度优先树遍历。PFT 有 $\log(N)+1$ 个阶段 s，并且 N 在阶段 $s = 0$设有叶节点，分别表示冻结比特和信息比特。每个节点 v 接收一个大小为 N_s 的对数似然比向量 α^v，首先根据向量 α^l 中的元素进行计算

$$\alpha_i^l = f\left(\alpha_i^v, \alpha_{i+N_v/2}^v\right) \tag{1}$$

通过 f 函数的硬件高效的最小和表达式

$$f(a,b) = \operatorname{sign}(a)\operatorname{sign}(b)\min\left(|a|,|b|\right). \tag{2}$$

所得向量 α^l（大小为 $N_s/2$）被传递给左子节点。利用从左子节点接收到的比特向量 β^l，通过 g函数计算 α^r 的$N_s/2$ 元素

$$\alpha_i^r = g\left(\alpha_i^v, \alpha_{i+N_v/2}^v, \beta_i^l\right) \tag{3}$$

with

$$g(a,b,c) = (1-2c)\cdot a + b, \tag{4}$$

并将其发送给右子节点。利用两个子节点的结果 β^l和 β^r，计算部分和 β^v，并将其发送给父节点。在叶节点中进行比特判决。根据定义，冻结比特始终为 0，信息比特节点返回

$$\beta^v = \operatorname{sign}(a^v), \tag{5}$$

其值在 $a^v \geq 0$ 时为 0，否则为 1。译码器输出 $\hat{\mathbf{x}} = \mathrm{SC}(y)$ 等于根节点 [9] 的输出 β^v。

C. 快速*SSC*译码

可以对 PFT 进行剪枝，以减少译码一个码字所需的运算量。只包含冻结比特的子树无需遍历，因为其译码结果已预先确定为全零向量。这些子树会被替换为零速率（R0）节点。同样，不含任何冻结比特的子树可以通过硬判决译码（HDD）直接译码，因为其中不包含任何奇偶校验信息。因此，这些节点称为单位速率节点 [9]。

随后又提出了进一步的优化方法 [10], , 称为 Fast-SSC 译码。如果子树只包含一个信息比特，则将其视为重复（REP）码，并替换为专用的 REP 节点。REP 节点中索引为 j 的所有比特均按以下方式译码：

$$\beta_j^{v,\mathrm{REP}} = \operatorname{sign}\left(\sum_{k=0}^{N_s-1}\alpha_k^v\right). \tag{6}$$

在仅包含一个冻结比特的子树中，该比特始终充当校验比特。因此，该子树的部分和表示一个单奇偶校验（SPC）码。专用SPC节点通过计算输入的奇偶校验 $\gamma^v \in \{0,1\}$ 来执行最大似然译码：

$$\gamma^v = \bigoplus_{j=0}^{N_s-1}\operatorname{sign}\left(\alpha_j^v\right), \tag{7}$$

找出最不可靠比特

$$j_{\min} = \underset{j\in[0,N_s)}{\arg\min}\left|\alpha_j^v\right| \tag{8}$$

并设置 β^v 以满足单奇偶校验约束：

$$\beta_j^{v,\mathrm{SPC}} = \begin{cases}\operatorname{sign}\alpha_j^v \oplus \gamma^v & \text{if } j = j_{\min}\\ \operatorname{sign}\alpha_j^v & \text{otherwise.}\end{cases} \tag{9}$$

III. 自同构集成译码

AED 在 [7] 中被提出，作为一种通用方法，通过运行由 M 个相互独立的 SC 译码器组成的集成来提升误码率性能。每个译码器 m 都对接收序列的一个置换版本进行处理，然后将每个译码结果反置换为

$$\hat{\mathbf{x}}_m = \pi_m^{-1}\left(\mathrm{SC}\left(\pi_m(\mathbf{y})\right)\right). \tag{10}$$

这些置换 π_m 源自该码的自同构群，即能够将每个码字映射到某个（不一定不同的）码字上的编码比特置换集合。因此，置换后的接收序列对应于不同的噪声实现，可能更易于译码。因此，通过从 $\hat{}\mathbf{x}_m$ 中选择最可能的候选码字，找到正确码字的概率得以提高。

A. 自同构与码构造

极化码的自同构群包含块下三角仿射（BLTA）群，其定义为

$$\mathbf{z}' = \mathbf{A}\mathbf{z} + \mathbf{b}, \tag{11}$$

其中$\mathbf{A}$ 是一个可逆的块下三角二进制矩阵，$\mathbf{b}$ 是任意二进制向量，而$\mathbf{z}$,$\mathbf{z}'$ 分别是置换前后比特索引的二进制表示 [8]。$\mathbf{A}$的块剖面以及自同构的数量取决于信息集的选择 $\mathcal{I}$。因此，为了在AED下取得良好性能，极化码设计至关重要。在 [11], 中发现，具有最小信息集 $\mathcal{P}$的（128,60）码特别适合AED，因为其BLTA群的块剖面为$\mathbf{s}$ = （3,4）。

B. 自同构选择

文献 [11] 表明，置换可以划分为等价类。在每个等价类（EC）中，置换译码会得到相同的码字估计。因此，在 AED 中，每个等价类至多使用一个置换。由于 $\mathcal{P}$ (128,60) 码具有 2205 个等价类，因此必须选择其中的一个子集用于 AED。

此外，使用一个统一的 M置换列表也很有益，这样较小的集成规模 $M' < M$ 可以使用该列表的一个子集。由于目前尚无关于如何选择优良等价类子集的理论，我们提出了一种贪心的数据驱动方法，具体如下：

1）从每个 EC 中生成一个置换。2）由于对称性，任意选择一个置换 π_1。3）收集一批无法通过任何已选置换正确译码的含噪码字。4）使用尚未选取的置换对这批码字进行译码，并选出能够正确译码其中最多含噪码字的置换。5）重复步骤 3 至 5，直至选出 M 个置换 π_1，……，π_M。

C. 传统候选选择

为了从所有 M 个译码结果 $\hat{}\mathbf{x}$ $\hat{}\mathbf{x}_m$ 中选出概率最高的码字，[7]提出了列表内最大似然方法

$$\hat{\mathbf{x}} = \underset{\hat{\mathbf{x}}_m, m\in\{1,\ldots,M\}}{\arg\max} \mathbf{y}^{\mathrm{T}}(1-2\hat{\mathbf{x}}_m) \tag{12}$$

该方法选择与接收到的对数似然比相关性最高的码字 $\hat{}\mathbf{x}_m$ 。

D. 路径度量

列表内最大似然方法的一个缺点是，必须使用接收到的信道对数似然比$\mathbf{y}$ 才能进行最终决策。对于深度流水线化的高吞吐量译码器，这需要一个大型缓冲区。根据 M，该缓冲区最多可占译码器总面积的 25%。这部分存储器也是功耗的一大来源。因此，找到避免使用该缓冲区的技术十分重要。我们提出一种基于新路径度量（PM）的候选选择方法，用于采用 SC子译码器的 AED。该 PM 技术取代了列表内最大似然选择，从而提高了 AED 实现中的局部性。此外，它还使缓冲区变得多余。

在 SCL译码中，PM 用于评估不同译码路径。 [12] 提出了一种高效的基于对数似然比的路径度量计算。特别是，研究表明，按如下方式计算的 PM

$$\mathrm{PM} = \sum_{i=0}^{N-1} \ln\left(1 + e^{-(1-2\beta_i^v)\alpha_i^v}\right) \tag{13}$$

是码字候选似然度的递减函数。因此，PM 最低的路径对应于最有可能的发送码字。（13）中的各项可采用便于实现的近似

$$\ln(1+e^x) \approx \max(0, x). \tag{14}$$

与SCL译码不同，SC译码不会分裂译码路径。对于信息比特（$i \in \mathcal{I}$），$\hat{}u_i$ 等于对 α_i^v的硬判决，因此它们不会影响概率测度（PM）。因此，对于SC译码，只需对冻结比特更新概率测度，其中只有 $\alpha_i^v < 0$ 表示发生了传输错误。这给出了叶节点中概率测度的以下代价函数：

$$\mathrm{PM} = \sum_{i\in\mathcal{F}} |\min(0, \alpha_i^v)| \tag{15}$$

随后选择概率测度最小的码字$\hat{}\mathbf{x}_m$ 作为AED的输出。

*E. Fast-SSC*路径度量

目前最先进的高吞吐量、低时延译码器架构基于Fast-SSC译码算法[13]。由于 Fast-SSC译码在 PFT 的较高层级上运行，因此不会计算冻结叶节点的对数似然比值。因此，前文介绍的概率测度计算方法并不直接适用。于是，我们推导出适用于专用节点的概率测度计算方法，类似于

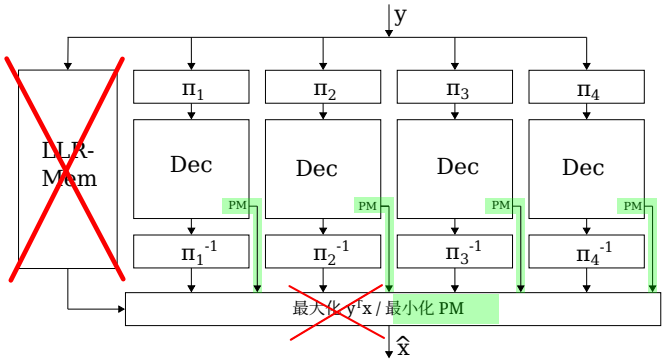

图1. 基于路径度量、包含 $M = 4$的AED架构。与传统AED相比，所提出的架构不需要用于存储对数似然比的记忆。新颖的基于路径度量的选择过程以绿色突出显示。

快速简化 SCL（SSCL）译码 [14]。概率测度 $\mathrm{PM}^{(\ell)}$ 按顺序累加，其中 ℓ 表示剪枝 PFT 中叶节点的索引，初始值为 $\mathrm{PM}^{(0)} = 0$。

如上文所述，SC 译码器不会拆分译码路径，并且PM计算方法仅适用于包含冻结比特的专用节点，即R0、REP和SPC节点：

1) 零速率节点： 对于R0节点，我们遵循（15），但直接在节点大小为 N_s的节点输入对数似然比向量 α^v上执行计算：

$$\mathrm{PM}^{(\ell)} = \mathrm{PM}^{(\ell-1)} + \sum_{j=0}^{N_s-1} |\min(0, \alpha_j^v)|. \tag{16}$$

2) 重复节点： 根据（6），重复比特判决基于与REP节点相关的所有对数似然比之和的符号。因此，校正比特位于位置 j，对于这些位置，$\mathrm{sign}(\alpha_j^v) \neq \beta_j^v$。因此，PM计算为

$$\mathrm{PM}^{(\ell)} = \mathrm{PM}^{(\ell-1)} + \sum_{\substack{j=0 \\ \mathrm{sign}(\alpha_j^v) \neq \beta_j^v}}^{N_s-1} |\alpha_j^v|. \tag{17}$$

3) 单奇偶校验节点： 对于SPC节点，译码包括校正最不可靠比特，以满足单奇偶校验约束（（7）-（9））。因此，PM可推导为：

$$\mathrm{PM}^{(\ell)} = \mathrm{PM}^{(\ell-1)} + \gamma^v \cdot |\alpha_{j_{\min}}^v|. \tag{18}$$

仿真验证了，与基于式 (12) 的传统候选选择相比，这种 Fast-SSC 路径度量具有相同的纠错性能。

F. 架构

图 1 展示了一种 AED 架构，其中包含 $M = 4$ M 个相同的 SC 译码器。置换 π_m和 π_m^{-1}分别将对数似然比 $\mathbf{y}$ 和码字映射到各个 SC 译码器实例，并依据自同构进行映射（第三节-B）。此映射无需任何逻辑或寄存器。所有译码器均独立工作，并输出各自的译码结果和概率测度。每个译码器按照第三节-E 中所述计算该概率测度，并将其馈送至选择单元；选择单元随后输出概率测度值最低的码字估计 $\hat{x}$。所有译码器均由一个框架自动生成，采用完全展开的流水线架构。

为说明所提出的优化，图 1 还展示了传统 AED 架构中的码字选择方式：该架构使用对数似然比存储器和列表内最大似然方法（图中以红色划掉），而新颖的基于路径度量的选择方式则以绿色突出显示。

IV. 结果

为进行公平比较，所有译码器（AED 和 SCL 译码器）的吞吐量均优化至相同水平，即 64 Gbit/s。我们研究了 AE 译码器的不同并行度 $M \in \{2, 4, 8, 16\}$，以及 SCL 译码器的不同列表大小 $L \in \{2, 4, 8, 16\}$。这些译码器分别记为 AED-M 和 SCL-L。SCL 译码器采用了 [15] 中的优化；据我们所知，它们是适用于极高吞吐量的最高效 SCL 译码器实现。

尽管这种高吞吐量对于超可靠低时延通信（URLLC）应用场景而言或许并非必要，但所研究的展开式和流水线架构不仅面积和能效都很高，而且还具有极低的时延，而这正是超可靠低时延通信的核心要求之一。降低频率可以减少吞吐量，从而降低功耗，但代价是时延增加。

所有设计均使用 *Design Compiler* 进行综合，并使用来自新思科技 的 *IC-Compiler*完成布局布线；所采用的工艺为格芯的12纳米FinFET工艺，且设计均在最坏情况下的工艺、电压和温度（PVT）条件下进行（时序分析：125 °C、0.9 V；功耗分析：1.0 V）。

在加性高斯白噪声（AWGN）信道和二进制相移键控（BPSK）调制下，对纠错性能进行了仿真。仿真中至少包含100个错误码字。为评估这些码的纠错能力，我们还评估了最大似然（ML）性能。为此，我们仿真了两个界：最大似然上界和最大似然下界。如果两个界相等，则可获得最大似然性能。最大似然上界通过使用 $L = 128$ 进行 SCL译码，并统计码字与接收信道值的距离小于正确码字与接收信道值距离的情况得出。下界的推导方法相同，但还要求正确码字必须在候选项列表中。

A. 与 *SCL* 的比较

我们首先比较 AED 与 SCL译码（图 2）的纠错性能，二者均采用第二节-A 中所述的相同 $\mathcal{P}$(128,60)。在 L / M 的整个范围内，两种译码算法的表现都非常相似。它们

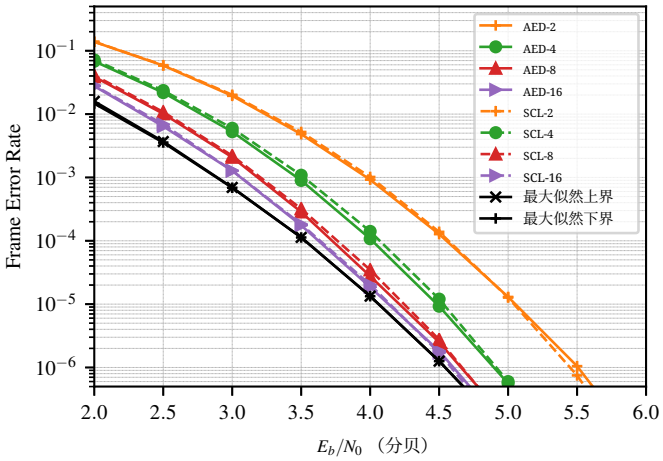

图 2. 相同 $\mathcal{P}(128,60)$下 AED 与 SCL 译码的比较

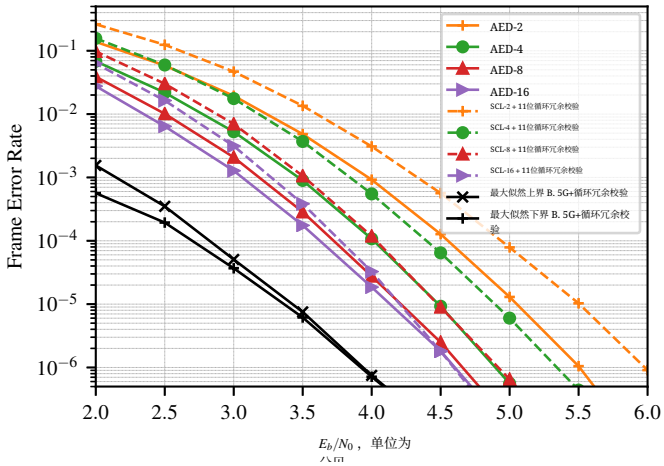

图3. $\mathcal{P}(128,60)$的AED与5G$\mathcal{P}(128,60+C)$的CRC辅助SCL译码相比

都受益于较大的最小距离。因此，我们省略了额外的循环冗余校验（CRC），因为它无法提升所选码的性能。对于$L = M = 16$ 这两种译码算法，其性能都接近最大似然界。

表 I 展示了实现结果。我们观察到，在所有给出的指标上，AED 均优于对应的 SCL 实现。随着 L / M 增大，AED 与 SCL 之间的差距也随之扩大，因为 AED 实现能从局部性中显著受益。对于 $L = M = 16$，观察到面积效率提升超过 8.9×、能效提升 4.5× ，且时延缩短 5.3× 。

B. 与 *5G* 的比较

我们还将我们的实现与针对 5G$\mathcal{P}(128,\ 60+C)$、采用 $C = 11$ 位 CRC 的优化型先进 CRC 辅助 SCL译码器进行了比较。如图 3 所示，对于所有 L / M，AED 的性能均优于 CRC 辅助 SCL译码；而对于 $M = L = 16$ ，随着信噪比升高，性能差距逐渐缩小。值得注意的是，AED-4 的性能与 CRC 辅助 SCL-8 非常接近，并且在此纠错要求下还能进一步节省硬件。尽管 5G$\mathcal{P}(128,\ 60+C)$ 的最大似然界低于 $\mathcal{P}(128,\ 60)$ 码，但 SCL译码器无法利用这一优势。

分析 AED 和 5G SCL 译码器的实现成本（表 II）可知，其实现表现与第四节-A 中的情况相似。不过，与不带 CRC 的 $\mathcal{P}(128,\ 60)$ 码相比，5G 码的译码器在面积和能效方面略有优势。这是由不同的码结构所致，它会在 PFT遍历中形成不同的构建模块。

值得一提的是，5G码针对广泛的码率和使用场景进行了优化，而不仅仅着眼于纠错性能和实现效率。

C. 与*GRAND*的比较

由于相关实现寥寥无几，因此很难与 GRAND 或有序可靠比特 GRAND（ORBGRAND）算法进行比较。在 [6]作者为 a$\mathcal{P}(128,\ 105)$ 提出了一种基于 7 nm 工艺的优化架构。其以时钟周期数计量的最小时延仍超过我们提出的 AE 译码器的 2× 时延。由于所用代码不同，且作者仅提供了综合结果，而我们则在最坏情况 PVT 条件下完成了全流程布局布线，因此很难进行进一步比较。

V. 结论

本文首次实现了AED，并将其与最先进的SCL译码器进行了比较。我们表明，AED译码器在时延、面积和能效方面均优于SCL译码器。我们还证明，其纠错性能与SCL译码器相当；对于5G码，其性能甚至优于与CRC的级联方案。因此，AED及所提出的架构非常适用于6G超可靠低时延通信场景，因为该场景要求低时延和高纠错能力。

参考文献

[1] W. Saad、M. Bennis 和 M. Chen，“6G无线系统愿景：应用、趋势、技术与开放性研究问题”，*IEEE Network*，第34卷第3期，第134–142页，2020年。[2] C. Yue、V. Miloslavskaya、M. Shirvanimoghaddam、B. Vucetic 和 Y. Li，“6G超可靠低时延通信中的短码长码高效译码器”，2022 [ ] 2206 09572 。在线获取：[3] https://arxiv.org/abs/。E. Arıkan，“信道极化：一种为对称二进制输入无记忆信道构造达到容量码的方法”，*IEEE Trans. Inf. Theory*，第55卷第7期，第3051–3073页，2009年。[4] I. Tal 和 A. Vardy，“极化码的列表译码”，*IEEE Trans. Inf. Theory*，第61卷第5期，第2213–2226页，2015年5月。[5] K. R. Duffy、J. Li 和 M. M´edard，“通过猜测噪声实现达到容量的译码”，2018年。 [在线]获取：http://arxiv.org/abs/1802. 07010[6] C. Condo，“具有查找表辅助错误模式调度的固定时延ORBGRAND译码器架构”，*IEEE Transactions on Circuits andSystemsI:Regular Papers*，第69卷第5期，第2203–2211页，2022年。

表 I 相同 $\mathcal{P}(128,\ 60)$ 下 AED 和 SCL 译码器的实现结果

	AED-2	SCL-2	AED-4	SCL-4	AED-8	SCL-8	AED-16	SCL-16
频率 [MHz]	498	476	497	462	498	478	495	403
吞吐量 [Gbps]	63.7	61.0	63.6	59.1	63.7	61.2	63.3	51.6
时延 [CC]	11	21	11	27	11	39	11	48
时延 [ns]	**22.1**	**44.1**	**22.1**	**58.5**	**22.1**	**81.5**	**22.2**	**119.1**
面积 [mm^2]	0.045	0.064	0.087	0.154	0.170	0.450	0.338	2.464
面积效率 [Gbps/mm^2]	**1407.9**	**956.0**	**733.8**	**385.0**	**375.1**	**136.2**	**187.0**	**20.9**
利用率 [%]	73	73	73	72	74	74	73	47
总功耗 [W]	0.07	0.09	0.15	0.22	0.32	0.61	0.64	2.38
能效 [pJ/bit]	**1.16**	**1.48**	**2.32**	**3.73**	**5.01**	**10.02**	**10.12**	**46.21**
功率密度 [W/mm^2]	1.63	1.42	1.70	1.44	1.88	1.36	1.89	0.97

表 II $\mathcal{P}(128,\ 60)$ 的 AED 与 5G $\mathcal{P}(128,\ 60+C)$ 的 CRC 辅助 SCL 译码器的实现结果

	AED-2	SCL-2	AED-4	SCL-4	AED-8	SCL-8	AED-16	SCL-16
频率 [MHz]	498	478	497	475	498	478	495	412
吞吐量 [Gbps]	63.7	61.2	63.6	60.8	63.7	61.1	63.3	52.8
时延 [CC]	11	21	11	26	11	36	11	47
时延 [ns]	**22.1**	**43.9**	**22.1**	**54.7**	**22.1**	**75.4**	**22.2**	**114.0**
面积 [mm^2]	0.045	0.063	0.087	0.138	0.170	0.378	0.338	1.990
面积效率 [Gbps/mm^2]	**1407.9**	**968.8**	**733.8**	**440.0**	**375.1**	**161.8**	**187.0**	**26.5**
利用率 [%]	73	72	73	73	74	77	73	47
总功耗 [W]	0.07	0.08	0.15	0.21	0.32	0.62	0.64	1.84
能效 [pJ/bit]	**1.16**	**1.39**	**2.32**	**3.52**	**5.01**	**10.08**	**10.12**	**34.82**
功率密度 [W/mm^2]	1.63	1.35	1.70	1.55	1.88	1.63	1.89	0.92

[7] M. Geiselhart、A. Elkelesh、M. Ebada、S. Cammerer 和 S. ten Brink，“Reed–Muller码的自同构集成译码”，*IEEE Transactions on Communications*，第69卷，6438，2021年10月，第6424页，否。[8] ——，“关于极化码的自同构群”，载于*2021 IEEE International Symposium on Information Theory (ISIT)*，2021年，第1230–1235页。[9] A. Alamdar-Yazdi 和 F. R. Kschischang，“极化码的简化连续消除译码器”，*IEEE Commun. Lett.*，第15卷第12期，第1378–1380页，2011年12月。[10] G. Sarkis、P. Giard、A. Vardy、C. Thibeault 和 W. J. Gross，“快速极化码译码器：算法与实现”，*IEEE J. Sel. Areas Commun.*，第32卷第5期，第946–957页，2014年5月。[11] C. Pillet、V. Bioglio 和 I. Land，“用于极化码译码的自同构分类”，载于*ICC 2022 - IEEE International Conference on Communications*，2022年，第110–115页。[12] A. Balatsoukas-Stimming、M. B. Parizi 和 A. Burg，“基于对数似然比的极化码连续消除列表译码”，*IEEE Transactions on Signal Processing*，第63卷第19期，第5165–5179页，2015年10月。[13] P. Giard、A. Balatsoukas-Stimming、T. C. M¨uller、A. Bonetti、C. Thibeault、W. J. Gross、P. Flatresse 和 A. Burg，“PolarBear：用于极化码译码的28纳米FD-SOI专用集成电路”，*IEEE Journal on Emerging and Selected Topics in Circuits and Systems*，第7卷第4期，第616–629页，2017年12月。[14] S. A. Hashemi、C. Condo 和 W. J. Gross，“快速且灵活的极化码连续消除列表译码器”，*IEEE*，第65卷第21期，第5756–5769页，2017年11月。[15] L. Johannsen、C. Kestel、O. Griebel、T. Vgt 和 N. Wehn，“快速简化连续消除列表译码器中单位速率节点的基于偏序的译码”，*Electronics*，第 11， 4, 卷。在线。

获取：https://www.mdpi.com/ - / / /