引用格式：韩国军, 董鹏, 曾子峰, 等. 基于奇偶校验和循环冗余校验的增强 Fast-SCL 译码算法 [J]. 电子科技大学学报, 2026, 55(4): 565-570.
Han Guojun, Dong Peng, Zeng Zifeng, et al. Enhanced fast-SCL decoding based on parity check and cyclic redundancy check[J]. Journal of University of Electronic Science and Technology of China, 2026, 55(4): 565-570. (in Chinese)

基于奇偶校验和循环冗余校验的增强 Fast-SCL 译码算法

韩国军[1]，董　鹏[1]，曾子峰[1]，翟雄飞[1*]，史治平[2]

（1. 广东工业大学信息工程学院，广州 510006；2. 电子科技大学通信抗干扰国家级重点实验室，成都 611731）

摘要：极化码主流的快速串行消除列表（Fast-SCL）译码算法虽能实现低时延，但在中短码长下性能较差。针对该问题，提出了一种结合奇偶校验（parity check, PC）与循环冗余校验（CRC）的 Fast-SCL 算法。在 Fast-SCL 算法中引入 PC 位以实现早期的路径验证，提升路径选择的可靠性；同时，在校验位总数不变的前提下，考虑 PC 位与 CRC 位的多种组合方式。仿真结果表明，在短码长、帧错误率（FER）为 10^{-2} 的条件下，与原始的 CRC 辅助 Fast-SCL 算法相比，该算法性能增益最高可达 0.35 dB，计算复杂度最多可降低 10%。

关键词：极化码；特殊节点；快速串行消除列表；奇偶校验；循环冗余校验

中图分类号：TP911　　**文献标志码：**A　　DOI: 10.12178/1001-0548.2025200

Enhanced fast-SCL decoding based on parity check and cyclic redundancy check

HAN Guojun[1], DONG Peng[1], ZENG Zifeng[1], ZHAI Xiongfei[1*], and SHI Zhiping[2]

(1. School of Information Engineering, Guangdong University of Technology, Guangzhou 510006, China;
2. National Key Laboratory of Wireless Communications, University of Electronic Science and Technology of China, Chengdu 611731, China)

Abstract: The fast successive cancellation list (Fast-SCL) decoding algorithm for polar codes can achieve low latency, but still performs poorly under medium and short code lengths. To deal with this issue, this paper proposes an improved Fast-SCL algorithm by combining parity check (PC) and cyclic redundancy check (CRC). Specifically, PC bits are introduced into the Fast-SCL algorithm to enable real-time path verification, which makes the path selection more reliable. Additionally, with the same total number of check bits, various combinations of PC bits and CRC bits are considered. The simulation results demonstrate that the proposed algorithm achieves a performance gain of 0.35 dB and a complexity reduction of 10% compared to the original CRC-assisted Fast-SCL algorithm under the short code lengths with the frame error rate (FER) being 10^{-2}.

Key words: polar code; special nodes; Fast-SCL; parity check; cyclic redundancy check

极化码[1]是被证明在码长足够大时能达到香农极限的信道编码方案，因其复杂度低，受到广泛关注。但在实际应用中，为保证编码效率，码长通常受限，因此在有限码长下提升性能成为极具潜力的研究方向之一。

为实现极化码的高效译码，串行消除列表（successive cancellation list, SCL）算法被提出[2]。与传统串行消除（successive cancellation, SC）算法相比，此算法性能显著提升。文献 [2-3] 利用循环冗余校验（cyclic redundancy check, CRC）进行最终路径选择，提出了 CRC 级联极化码及对应的 CRC 辅助 SCL（CA-SCL）译码算法。为进一步提升中短码长下 SCL 译码算法的性能，文献 [4-6] 分别采用分布式 CRC 码和奇偶校验码（parity check, PC）对译码结果执行早期的验证；文献 [7] 在极化码的 SCL 译码算法中联合考虑 CRC 与 PC，实现错误路径的早期检测与剪枝，在降低复杂度的同时提升了译码性能；文献 [8] 进一步分析了 PC 比特

收稿日期：2025−10−13
基金项目：国家自然科学基金（62301166, 62471151, U2001203, 62371101）；广州市科技计划项目（2025A04J2904）；2020 广东省重点领域研发计划“芯片、软件与计算（芯片类）”专项计划（2021B1101270001）
作者简介：韩国军，博士，教授，主要从事差错控制编译码技术方面的研究。
*通信作者 E-mail: zhaixiongfei@gdut.edu.cn

的位置，提出一种优化的 PC 编码方法。

为进一步降低译码复杂度，文献 [9-10] 基于极化码子码中冻结位的特定分布模式，引入了 4 类特殊节点，即 R-0、R-1、SPC 和 REP 节点，并提出 4 种对应的快速最大似然（maximum likelihood, ML）译码方法，直接对 4 类子码进行码字序列估计。随后，上述特殊节点的应用被扩展到 SCL 算法中，形成了简化 SCL（simplified SCL, SSCL）算法[11]。但研究表明，其中针对 R-1 和 SPC 节点的译码步骤会导致性能损失。为解决该问题，文献 [12] 提出一种基于球形译码的 R-1 和 SPC 节点无损 SCL 译码方法。受文献 [12] 启发，文献 [13] 总结了适用于这 4 类特殊节点的快速 SCL（Fast-SCL）译码算法，降低了译码时延。此外，在硬件实现中，还可采用帧间流水线或多比特译码等硬件译码架构进一步改善译码时延[14-17]。

然而，目前缺乏针对 Fast-SCL 算法的早期高效的路径筛选方法。为此，本文提出一种基于 PC 和 CRC 的增强 Fast-SCL 译码算法。此方案实现了 PC-CRC 与极化码的混合编码，利用 PC 比特在路径筛选过程中实时检测并剪枝错误路径，利用 CRC 序列选择最终输出路径，实现中短码长下更可靠的译码结果。

1 极化码及传统译码算法介绍

1.1 极化码

极化码可表示为$(N,K,\mathcal{A})$的形式，其中$N=2^n$（n 为正整数）表示码长，K表示非冻结位个数，码率$R=K/N$；$\mathcal{A}$表示传输非冻结位的可靠子信道索引集，且$|\mathcal{A}|=K$，表示集合$\mathcal{A}$的基数。相应地，$\mathcal{A}^c$表示冻结位的子信道索引集，为集合$\mathcal{A}$的补集。

极化码的编码包括选择可靠子信道（极化码构造）、混合比特和构造生成矩阵 3 个步骤。1）选择K个可靠子信道传输非冻结位，其余子信道传输冻结位，本文的极化码构造采用极化权重法（polarization weight, PW）；2）将非冻结位置于指定位置，其余位置填充冻结位，得到混合信源码字序列$\boldsymbol{u}[0:N-1]=([u[0],u[1],\cdots,u[N-1])$；3）由核心矩阵$\boldsymbol{F}=\begin{bmatrix}1&0\\1&1\end{bmatrix}$构造生成矩阵$\boldsymbol{G}_N=\boldsymbol{F}^{\otimes n}$（其中$\otimes n$表示$n$次克罗内克积）。最终得到编码序列：

$$\boldsymbol{x}=\boldsymbol{u}\boldsymbol{G}_N=\boldsymbol{u}\boldsymbol{F}^{\otimes n} \tag{1}$$

1.2 译码算法

首先介绍 SC 译码算法的主要步骤。以$N=8(n=3)$的极化码为例（如图 1 所示），SC 算法可看作对一棵$n+1$级二叉树的遍历。在二叉树的第s级$(0\leqslant s\leqslant n)$，共有$2^{n-s}$个子节点，每个子节点对应一个长度为$2^s$的子码。具体而言，记$N_s^i$为第$s$级的第$i$个节点，该节点从其父节点接收对数似然比（log-likelihood ratio, LLR）向量$\boldsymbol{\lambda}_s^i$；遍历完所有子节点后，N_s^i向其父节点返回部分和（partial sum, PSUM）向量$\boldsymbol{\beta}_s^i$。当第 0 级的第k个叶节点接收到$\boldsymbol{\lambda}_0^k$时，非冻结位$u[k]$的估计值$\hat{u}[k]$为：

$$\hat{u}[k]=\mathrm{HD}(\boldsymbol{\lambda}_0^k)=\begin{cases}0 & k\in\mathcal{A}^c\\ \dfrac{1-\mathrm{sign}(\boldsymbol{\lambda}_0^k)}{2} & \text{其他}\end{cases} \tag{2}$$

式中，$\mathrm{HD}(\boldsymbol{\lambda}_0^k)$表示基于 LLR 的硬判决。

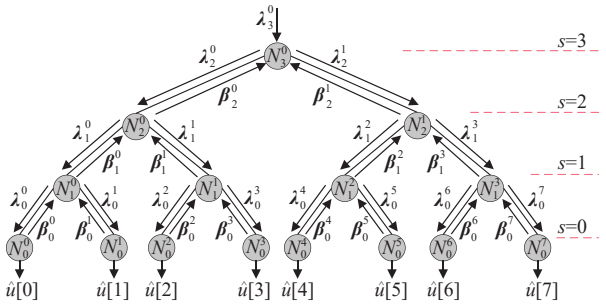


图 1 $N=8$极化码的 SC 译码树表示图

SC 算法基于硬判决且仅考虑一条最优路径，因此其译码性能欠佳。为提升译码准确性，SCL 算法基于路径度量（path metric, PM）保留L条译码路径，其中 PM 用于衡量路径可靠性。译码非冻结位时，对全部现存L条路径均保留“0”和“1”两种译码结果以实现路径扩展，随后按以下方式更新所有路径的 PM 值：

$$\mathrm{PM}_{0,l}^k\approx\begin{cases}\mathrm{PM}_{0,l}^{k-1} & \beta_{0,l}^k=\mathrm{HD}(\lambda_{0,l}^k)\\ \mathrm{PM}_{0,l}^{k-1}+|\lambda_{0,l}^k| & \text{其他}\end{cases} \tag{3}$$

式中，l为译码路径索引；$\boldsymbol{\lambda}_{0,l}^k$和$\mathrm{PM}_{0,l}^k$分别表示对应节点$\boldsymbol{N}_0^k$的第$l$条路径的 LLR 值和 PM 值；$\boldsymbol{\beta}_{0,l}^k$表示第$l$条路径的估计值。特别说明，冻结位无需进行路径扩展。

为降低译码时延，快速 SC（fast-SC）译码算法基于特殊节点的分布模式，无需遍历译码树即可直接估计节点的 PSUM。引入向量$\boldsymbol{d}=(d[0],d[1],\cdots,d[N-1])$（其中$d[k]$表示节点第$k$位是否为信息位），4 类特殊节点的定义如下[10]。

1）R-0 节点：所有位均为冻结位$\boldsymbol{d}=(0,0,\cdots,0)$；

2）R-1 节点：所有位均为非冻结位，$\boldsymbol{d}=(1,1,\cdots,1)$；

3）REP 节点：除最后一位外均为冻结位，$\boldsymbol{d}=(0,0,\cdots,1)$；

4）SPC 节点：除第一位外均为非冻结位，$\boldsymbol{d}=(0,1,\cdots,1)$。

上述 4 类特殊节点可通过特定的快速译码算法处理。若$\boldsymbol{N}_i^j$为 R-1 节点，可通过硬判决直接得到 PSUM：

$$\beta_j^i[k]=\mathrm{HD}\left(\lambda_j^i[k]\right)=\begin{cases}0 & \lambda_j^i[k]\geqslant 0\\ 1 & \text{其他}\end{cases} \tag{4}$$

若N_i^j为 SPC 节点，先进行硬判决，再通过模 2 加法计算输出的奇偶性；若不满足奇偶校验约束，则翻转可靠性最低的硬判决值。若N_i^j为 REP 节点，其 PSUM 向量只能为全 0 或全 1，具体取决于 LLR 值之和的符号位。后续将上述快速译码算法扩展到 SCL 译码中，译码器可基于输入的 LLR 向量，通过更少的步骤直接得到L个 PSUM[11-13]。

2　增强 Fast-SCL 译码算法

文献 [5] 验证了在 SCL 译码中 PC 位相比标准 CRC 辅助码在早期错误检测方面的有效性。受此启发，本文提出一种基于 PC 和 CRC 的增强 Fast-SCL 译码算法，通过双重路径验证，降低复杂度及帧错误率（frame error rate, FER）。具体而言，利用 PC 位对高速特殊节点进行实时错误路径删除，利用 CRC 位在最终路径选择中输出正确路径。在基于特殊节点划分的极化码结构中应用 PC 校验时存在两个挑战：一是如何为目标节点生成 PC 位并将其置于节点末尾；二是无法直接对所得 PSUM 进行 PC 校验。下面将针对这两个问题展开讨论。

2.1　基于节点的 PC-CRC 级联极化码

本节提出一种级联编码方案，即基于节点的 PC-CRC 极化码（node-based PC-CRC, NB-PC-CRC）。具体而言，将 PC 位置于部分特殊节点的末尾，将 CRC 序列置于非冻结位的末尾。NB-PC-CRC 极化码可表示为$(N,\mathcal{A},J,P)$，其中J表示 PC 位数量，P表示 CRC 阶数。

首先讨论哪些特殊节点需要执行 PC 校验。在文献 [5] 中，PC 位直接附在信息位末尾，PC 编码在极化码编码之前完成。但由于在本方案中进行 PC 校验的对象是特殊节点，该方法不适用于基于节点的极化码。根据 PW 构造法，在确定码长和信息位数时，特殊节点的划分便可以确定，此时，R-1 与 SPC 节点通常集中承载了绝大部分非冻结位（如在N=256，K=128 的条件下这个比例为 93%），它们的信息密度高且对后续译码决策影响显著。同时，受极化码的偏序关系影响，位于编码序列较靠前的子信道通常尚未完全极化，可靠性较低（更易发生早期判错），这在文献 [18] 中已有论证。此外，这两种节点最后一位必然为非冻结位，可以传输 PC 比特。因此将 PC 位置于靠前的J个 R-1/SPC 节点（称为校验节点）末尾，能最大化 PC 的早期检错效益，并降低错误路径累加对最终译码的负面影响。

另外，需要确定的是 NB-PC-CRC 极化码的 PC 位与 CRC 位的数量配置方式以及其作用机理。在本文 NB-PC-CRC 框架下，固定总校验位数$J+P$=16。实际通信系统中常用的 CRC 长度为一个离散值（典型取值为 4、8、12、16 位），因此，PC/CRC 的可选组合实际上是一个有限的离散集合（如：0/16、4/12、8/8、12/4、16/0），而并非一个连续的变量空间。在文献 [7-8] 以及类似的研究中，通常也选择了这几种配置方式作为研究和实验的对象。从译码原理上看， PC 作为对每个校验节点的局部线性检验，主要检测并剪枝早期产生的错误路径；而 CRC 作为最终端的全局检验，是为了降低最终路径选择错误的未被检出概率。文献 [19] 总结了译码中的两种错误类型主要是消失错误（译码中途正确路径被淘汰）和选择错误（最终选择了错误的译码路径）；同时验证了 PC 位的插入在低信噪比区域带来的性能增益效果最显著，而 CRC 的加入在高信噪比区域能抑制大部分的选择错误。对于一个包含m个信息比特且单个比特出错概率为p的校验节点，根据二项分布，PC 的检错概率可以近似表示为：

$$P_d=\sum_{k=1}^{m}\binom{m}{k}p^k(1-p)^{m-k}=\frac{1-(1-2p)^m}{2} \tag{5}$$

式中，k为奇数，表示节点中出错的信息比特个数。式 (5) 说明 PC 在p较大（即低 SNR 区域）时对检测多比特错误更有效。另一方面，长度为 r的 CRC 对随机错误向量的未检出概率可近似表示为2^{-r}，这说明 CRC 在抑制最终未检出错误的能力随r指数增大而降低。综上所述，在低 SNR 区域，误码主要由早期判错与路径扩展导致，增加 PC 位数有助于在中途剪枝错误路径，从而增加正确路径存活概率；在高 SNR 区域，路径扩展错误较少，最终路径选择错误占主导，增加 CRC 位数能显著降低未检出概率，从而改善高 SNR 的 FER。

NB-PC-CRC 极化码的编码框图如图 2 所示。首先，基于$\mathcal{A}$识别特殊节点，获取前J个需进行奇偶校验的 R-1 或 SPC 节点的信息（包括节点长度及其在信源码序列$\boldsymbol{u}$中的位置）；同时生成基

于节点的校验集$\mathcal{S}$，用于标识当前节点是否为校验节点；与此同时，从$\mathcal{A}$中剔除 PC 位和 CRC 位的索引，得到真实信息位的索引，真实信息比特数量为$K_m = K - J - P$；随后，将真实信息位序列$\boldsymbol{c}[0: K_m - 1]$与$F$个冻结位混合，得到不含 PC 位和 CRC 位的信源序列。由下式生成 PC 位：

$$p_k = \left(\sum_{q=0}^{K_j - 1} x_k[q] \right) \bmod (2) \quad k = 0, 1, \cdots, J - 1 \tag{6}$$

式中，$\boldsymbol{x}_k[q]$表示节点的第q个信息位；$K_j = 2^s - 1$。PC 编码器将生成的 PC 放置于指定特殊节点的末尾比特（图 3 展示了两个独立校验节点的结构），特殊节点一旦被识别，将不再进一步划分。每个校验节点末尾放置 一个 PC 位，且该 PC 位仅与其对应节点形成校验关系（而非与所有前序信息位），可降低译码时延并便于硬件实现。

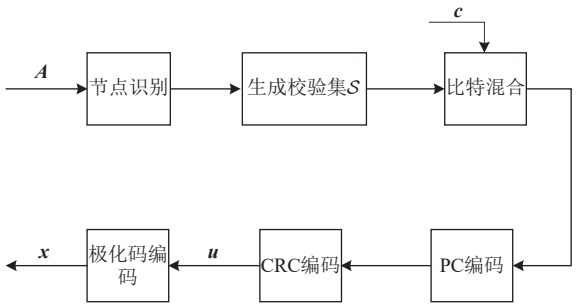


图 2 NB-PC-CRC 级联极化码编码框图

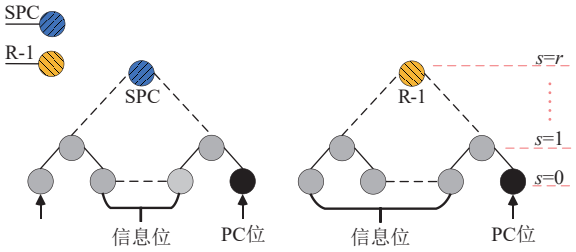


图 3 校验节点结构

PC 编码后，得到包含J个 PC 位的信息序列$\boldsymbol{c}_1[0: K_m + J - 1]$，经 CRC 编码器处理后，得到完整的信源码序列$\boldsymbol{u}[0: N - 1]$。最后通过极化码编码器生成最终的编码码字序列$\boldsymbol{x}$。

2.2 译码

现有基于节点的 Fast-SCL 译码算法中，译码一个节点后，仅基于 PM 值保留L条最可靠路径；即使引入 CRC，也仅在最终步骤进行 CRC 校验。与之不同，本文提出的基于 PC 和 CRC 的增强 Fast-SCL 译码算法采用双重路径校验策略：对位于不可靠位置的高速节点，此算法同时利用 PC 和 PM 值进行路径筛选。

详细译码流程如图 4 所示，其中M表示节点数量。特别地，为便于 PC 校验，需将 PSUM 向量转换为译码估计信息向量（通过 PSUM 向量与生成矩阵相乘可得）；译码一个校验节点后，执行 PC 校验：若存活路径数大于L，则利用 PM 值进一步筛选路径；否则保留 PC 校验后的所有存活路径。需注意，SPC 节点的译码过程涉及两类奇偶校验：一是传统子节点译码规则中规定码字判决值需满足奇偶约束；二是子节点译码完成后针对节点的信息比特估计值执行 PC 校验以实现实时路径验证，两类奇偶校验互不冲突。PC 校验过程涉及矩阵乘法，可在硬件（如现场可编程门阵列）中实现。此外，由于 SCL 类译码算法的复杂度与平均列表大小成正比，本文提出的算法通过中途奇偶校验剪枝错误路径，减小平均列表大小，从而降低计算复杂度。

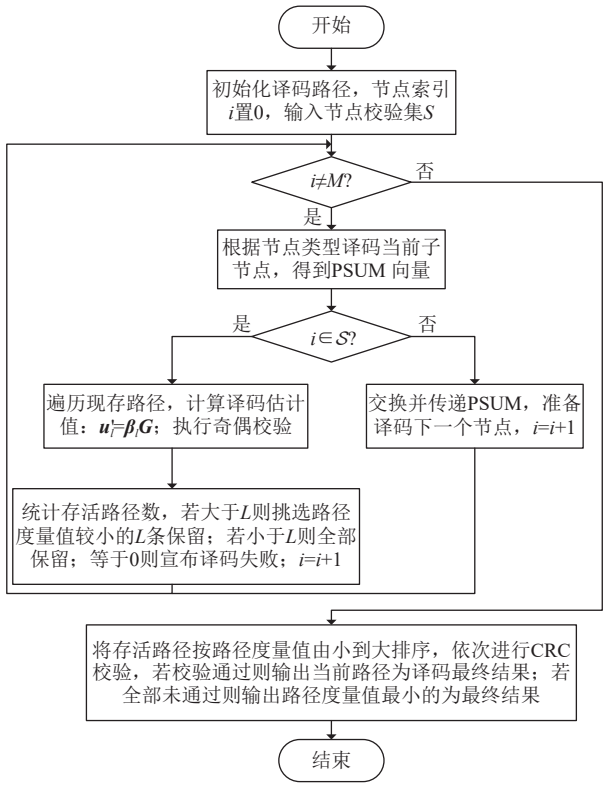


图 4 校验节点译码流程

3 仿真结果

将增强 Fast-SCL 译码算法与现有 CA-Fast-SCL 译码算法进行对比。仿真基于加性高斯白噪声（additive white gaussian noise, AWGN）信道模型，采用二进制相移键控（binary phase shift keying, BPSK）调制。符号（J-PC, P-CRC）表示 NB-PC-CRC 极化码的 PC 位与 CRC 位配置，同时也指代对应的译码算法。仿真中统一采用有效码率$R = (K - P - J)/N$，因此信噪比满足$E_b/N_0 = E_S/N_0 +$

$10\log_{10}(1/R)$。以含 16 位 CRC 的 CA-Fast-SCL 算法[11] 作为基准方案，该方案可视为本文算法的特殊配置，即（0-PC, 16-CRC）。同时对比了文献 [6-7] 的方案，其中文献 [6] 的方案采用 16 位 PC 位，与本文方案保持码率一致。

3.1　FER 性能

图 5 展示了在 N=128、R=0.4 时，不同列表大小下各算法的 FER 性能。由图 5 可知，在低 E_b/N_0区域，引入 PC 位可显著提升纠错性能。当 L=8、L=16、L=32 时，（8-PC, 8-CRC）算法相比 CA-Fast-SCL 算法的最大性能增益分别为 0.40、0.38、0.32 dB。当E_b/N_0<2 dB 时，L=16 的（8-PC, 8-CRC）算法性能甚至优于 L=32 的 CA-Fast-SCL 算法，性能增益达 0.2 dB。但随着E_b/N_0增大，（8-PC, 8-CRC）算法的性能增益逐渐消失，这与第 2 节的分析一致，即高E_b/N_0区域不应过度减少 CRC 位数量。且列表越大，这种性能下降越明显，如 L=32 时，（8-PC, 8-CRC）算法性能甚至劣于 CA-Fast-SCL 算法。相比之下，（4-PC, 12-CRC）算法在高 E_b/N_0区域表现更优。为了验证算法在更复杂信道条件下的鲁棒性，本文在图 5 中展示了 L=16 时 CA-Fast-SCL 算法和（8-PC, 8-CRC）算法在平坦块衰落瑞利信道（块长为 8）下的译码性能，其中还使用了二分集接收以及随机交织技术，对应的图例为 CA-Fast-SCL（L16, RF）和（8-PC, 8-CRC）（L16, RF）。结果表明，（8-PC, 8-CRC）算法在瑞利信道下仍保持显著的优势，说明所提 NB-PC-CRC 方案在常见慢衰落场景下具有一定的鲁棒性。

图 6 和图 7 分别展示了 N=128, 256，且 L=16 时，不同码率下各算法的 FER 性能。由图 6 可知，在不同 E_b/N_0区域，（8-PC, 8-CRC）与（4-PC, 12-CRC）算法仍表现出不同性能；且随着码率增大，所提算法的性能增益减小，这是因为码率越低，需进行奇偶校验的高速节点比例越高。具体而言，（8-PC, 8-CRC）算法相比 CA-Fast-SCL 算法，在不同码率下的性能增益分别约为 0.4 dB（低码率）和 0.32 dB（高码率）。从图 7 来看，码长越长，本文算法的性能增益越小：（8-PC, 8-CRC）算法相比 CA-Fast-SCL 算法的性能增益约为 0.18 dB。原因在于，短码长下信道极化不完全，部分传输信息的子信道未完全极化。而本文所提 NB-PC-CRC 极化码将这些信道纳入高速校验节点，可实现早期错误检测。此外可以看出，本文算法性能优于文献 [6]，与文献 [7] 相当，证明了 PC-CRC 编码结构在 Fast-SCL 译码架构中的有效性，即在特殊节点的基础上实现 PC 和 CRC 的级联，取得了译码时延与译码性能的平衡。

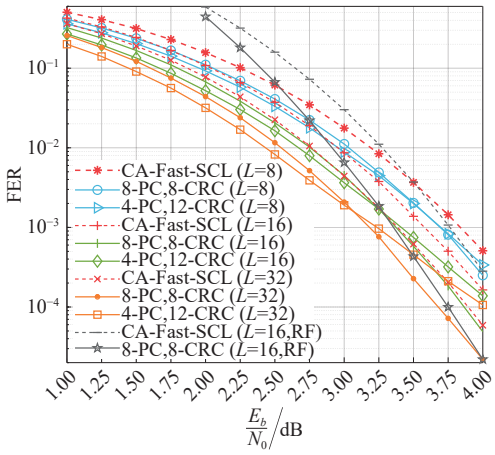


图 5　N=128、R=0.4 时各算法的 FER 性能

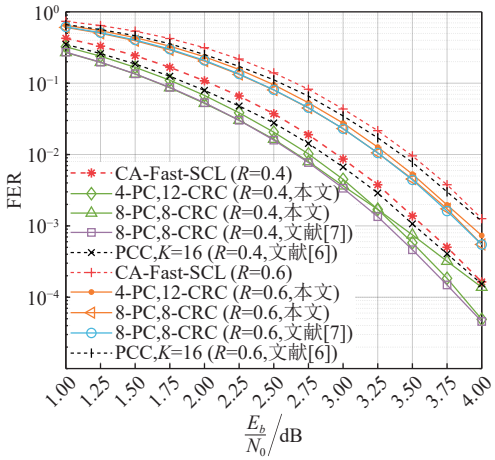


图 6　N=128、L=16 时各算法的 FER 性能

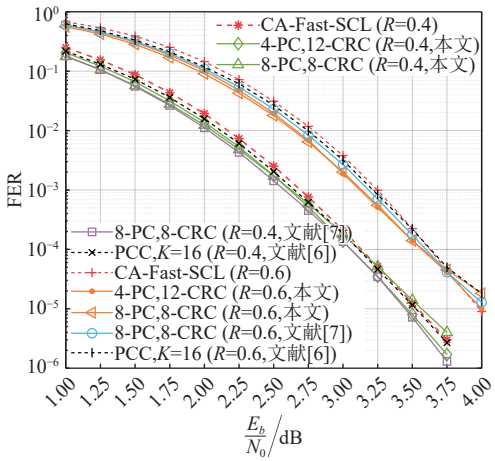


图 7　N=256、L=16 时各算法的 FER 性能

3.2　复杂度分析

表 1 展示了在 L=16, 32, N=128, R=0.4, E_b/N_0= 1 dB 时，（4-PC, 12-CRC）、（8-PC, 8-CRC）算法相比传统 CA-Fast-SCL 算法的平均列表大小（CA-Fast-SCL 算法的平均列表大小近似等于 L）、减少

的路径排序以及软件实现的归一化算法平均运行时间（以 CA-Fast-SCL 算法的平均运行时间归一化）。从表中可以看出，由于平均列表大小减小，路径排序次数有所减少。但是，在硬件实现中，本方案的早期路径校验必定会带来额外的剪枝操作，也意味着产生更多的路径复制及扩展。另外， PC 剪枝可能导致存活路径数变为非 2 的幂次，引入动态路径管理开销。从表 1 中的归一化算法平均运行时间也可以看出本文算法并不占优，且 PC 位的增加会显著提高算法运行时间。虽然平均列表大小的减小可以在一定程度上抵消该额外复杂度，但仍需设计高效硬件架构以充分发挥本文算法的优势，这是未来的研究方向之一。

表 1　各算法的复杂度对比情况

算法	平均列表大小		减少的路径排序/%		归一化算法平均运行时间	
	L=16	*L*=32	*L*=16	*L*=32	*L*=16	*L*=32
4-PC, 12-CRC	15.034 8	29.581 6	4.333	3.222	1.069	1.018
8-PC, 8-CRC	14.806 6	28.623 2	4.444	4.666	1.621	1.063

4　结束语

本文提出一种基于 PC 和 CRC 的增强 Fast-SCL 译码算法，以提升极化码译码的可靠性。通过在特殊节点中引入 PC 位实现早期路径验证，提前剪枝错误路径，减小平均列表大小；同时分析 PC 位与 CRC 位在不同配置下的优化算法性能。相比传统 CA-Fast-SCL 算法，本文算法的性能增益最高可达 0.35 dB，平均列表大小最多可降低 10%。

参 考 文 献

[1] Arikan E. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels[J]. IEEE Transactions on Information Theory, 2009, 55(7): 3051-3073.

[2] Tal I, Vardy A. List decoding of polar codes[J]. IEEE Transactions on Information Theory, 2015, 61(5): 2213-2226.

[3] Niu Kai, Chen Kai. CRC-aided decoding of polar codes[J]. IEEE Communications Letters, 2012, 16(10): 1668-1671.

[4] Chen Jie, Chen Yu, Jayasinghe K, et al. Distributing CRC bits to aid polar decoding[C]//2017 IEEE Globecom Workshops. Singapore: IEEE, 2017: 1-6.

[5] Guo Jianfeng, Shi Zhiping, Liu Zilong, et al. Multi-CRC polar codes and their applications[J]. IEEE Communications Letters, 2016, 20(2): 212-215.

[6] Wang Tao, Qu Daiming, Jiang Tao. Parity-check-concatenated polar codes[J]. IEEE Communications Letters, 2016, 20(12): 2342-2345.

[7] Yu Qingping, Shi Zhiping, Yan Qinhuai, et al. Hybrid parity-check and CRC aided SCL decoding for polar codes[C]//2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). Chengdu: IEEE, 2016: 711-716.

[8] Li Wanzhen, He Zhongqiu. An efficient CRC-aided parity-check concatenated polar coding[C]//2021 IEEE Asia Conference on Information Engineering. Sanya: IEEE, 2021: 1-5.

[9] Alamdar-Yazdi A, Kschischang F R. A simplified successive-cancellation decoder for polar codes[J]. IEEE Communications Letters, 2011, 15(12): 1378-1380.

[10] Sarkis G, Giard P, Vardy A, et al. Fast polar decoders: Algorithm and implementation[J]. IEEE Journal on Selected Areas in Communications, 2014, 32(5): 946-957.

[11] Sarkis G, Giard P, Vardy A, et al. Fast list decoders for polar codes[J]. IEEE Journal on Selected Areas in Communications, 2016, 34(2): 318-328.

[12] Hashemi S A, Condo C, Gross W J. A fast polar code list decoder architecture based on sphere decoding[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2016, 63(12): 2368-2380.

[13] Hashemi S A, Condo C, Gross W J. Fast and flexible successive-cancellation list decoders for polar codes[J]. IEEE Transactions on Signal Processing, 2017, 65(21): 5756-5769.

[14] Ye Longjian, Zhai Xiongfei, Zeng Zifeng, et al. High-throughput polar code SCL decoding with dynamically configurable pipeline architecture[C]//2023 IEEE/CIC International Conference on Communications in China. Dalian: IEEE, 2023: 1-6.

[15] Zheng Haotian, Balatsoukas-Stimming A, Cao Zizheng, et al. Implementation of a high-throughput fast-SSC polar decoder with sequence repetition node[C]//2020 IEEE Workshop on Signal Processing Systems. Coimbra: IEEE, 2020: 1-6.

[16] Jeong S L, Bae J H, Sunwoo M H. Fast multibit decision polar decoder for successive-cancellation list decoding[J]. Journal of Signal Processing Systems, 2021, 93(1): 127-136.

[17] 张宇. Polar 码译码算法研究及 FPGA 实现[D]. 成都: 电子科技大学, 2023.
Zhang Yu. The research and FPGA implementation of polar code decoding algorithm[D]. Chengdu: University of Electronic Science and Technology of China, 2023. (in Chinese)

[18] 曾子峰. 极化码低时延 SCL 译码器研究与设计[D]. 广州: 广东工业大学, 2025.
Zeng Zifeng. Research and design of low-latency SCL decoder for polar codes[D]. Guangzhou: Guangdong University of Technology, 2025. (in Chinese)

[19] 王涛, 屈代明, 江涛. 降低 SCL 译码错误的级联极化码[J]. 中兴通讯技术, 2019, 25(1): 5-11.
Wang Tao, Qu Daiming, Jiang Tao. Concatenated polar codes for SCL decoding error reduction[J]. ZTE Technology Journal, 2019, 25(1): 5-11. (in Chinese)

责任编辑　税　红