引用格式：韩国军, 曾子峰, 董鹏, 等. 基于特殊节点划分的低时延极化码 SCL 译码器架构 [J]. 电子科技大学学报, 2026, 55(1): 93-99.
HAN G J, ZENG Z F, DONG P, et al. A low-latency architecture of SCL decoder for FPGA based on special node division[J]. Journal of University of Electronic Science and Technology of China, 2026, 55(1): 93-99.

基于特殊节点划分的低时延极化码 SCL 译码器架构

韩国军[1]，曾子峰[1]，董　鹏[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全国重点实验室，成都 611731）

摘要：极化码基于信道极化理论提出，是唯一一种被理论证明可以达到香农限的信道编码方案。极化码主流的串行抵消列表（SCL）译码算法具有串行逐比特译码的特点，并且包含大量的路径度量和路径扩展，这导致译码时延较高。为了解决这一问题，提出了一种基于特殊节点划分的低时延 SCL 译码算法硬件架构，通过对不同类型的特殊节点分别使用剪枝或减分裂策略，以损失轻微译码性能为代价减少译码时延，提高吞吐量。为了提高译码器的灵活性，通过将不同码长码率的先验信息预先写入存储单元，从而实现编码参数可配置。可编程逻辑门阵列（FPGA）上的实验结果显示，在码长为 128～1 024 bit 以及码率为 0.3～0.5 的情况下，所提出的架构比标准 SCL 译码器降低了 40.32%～56.87% 的译码时延。

关键词：极化码；串行抵消列表；低时延；可配置译码器；可编程逻辑门阵列
中图分类号：TP911　　**文献标志码：**A　　DOI: 10.12178/1001-0548.2024291

A low-latency architecture of SCL decoder for FPGA based on special node division

HAN Guojun[1], ZENG Zifeng[1], DONG Peng[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: Polar code proposed based on channel polarization is the only one that is proven to achieve the Shannon limit. The mainstream successive cancellation list (SCL) decoding algorithm shows the extremely high latency due to its massive path splitting and path metrics. To solve this problem, a low-latency SCL decoding hardware architecture based on special nodes is proposed in this paper. Pruning or split-reducing is considered for different nodes to reduce the decoding latency and improve the throughput with slight performance degradation. Moreover, to enhance the flexibility, the prior information of different code lengths and code rates is stored in the read-only memory (ROM) and is selected based on the input configuration. The implementation results of the proposed architecture on field programmable gate array (FPGA) show that the decoding cycles are reduced by 40.32%~56.87% compared with the standard SCL decoder with the code length varying from 128 to 1 024 bit and the code rate varying from 0.3 to 0.5.

Key words: polar code; successive cancellation list; low latency; configurable decoder; FPGA

文献 [1] 在 2008 年首次提出信道极化理论，并证明了极化码在二进制离散无记忆信道（binary-input discrete memoryless channels, B-DMC）下可达香农限。2016 年，第三代合作伙伴计划（3rd generation partnership project, 3GPP）会议确定选择极化码作为第五代移动通信中增强移动宽带场景下控制信道的编码方案。

文献 [1] 在提出极化码的同时提出了一种低复杂度的译码算法，即串行抵消（successive cancellation, SC）译码算法，该算法在码长趋于无穷时可以达到香农限。然而，信道极化不充分导致中短码长下的 SC 译码性能不理想，逐比特译码会导致错误传

收稿日期：2024−10−23
基金项目：国家自然科学基金（62301166，62471151，62371101）
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

播。此外，串行译码会导致较高的时延。

为了提高译码性能，串行抵消列表（successive cancellation list, SCL）译码算法被提出[2]，该译码算法在译码信息位时执行路径扩展，并基于后验概率筛选出可靠路径，保留可能的候选路径。实验证明，SCL 译码器的性能随着列表数量的增加而逐渐接近最大似然（maximum likelihood, ML）译码性能。为了进一步提高性能，文献 [3] 将循环冗余校验（cyclic redundancy check, CRC）技术引入 SCL 译码算法中。文献 [4] 将奇偶校验（parity check, PC）引入具有 CRC 辅助的 SCL 译码，进一步提高了译码性能。上述算法虽然极大地提高了纠错性能，但都以较高的空间复杂度和计算复杂度为代价，不适用于低时延场景。

SC 类译码时延主要来自对数似然比（log-likelihood ratio, LLR）递归计算和大量的路径操作，为了减少 LLR 递归计算，简化 SC（simplified SC, SSC）和快速简化 SC（fast-simplified successive cancellation, Fast-SSC）译码算法相继被提出[5-6]。包括 Rate0、Rate1、SPC、REP 这 4 种特殊节点的并行译码方法，都大大降低了 SC 译码算法的时延。然而 Fast-SSC 并没有带来译码性能的提升，继而快速串行抵消列表（fast-successive cancellation list, Fast-SCL）译码算法[7] 被提出，将上述 4 种特殊节点引入 SCL 译码算法，但是特殊节点的路径扩展、度量、排序带来的极高硬件复杂度不适合硬件实现。在硬件实现方面，多比特串行抵消列表译码算法（multibit-decision successive cancellation list, MSCL）[8-9] 通过对长度为 M 的任意节点并行译码降低译码时延，这仍无法避免硬件复杂度问题。这主要由于在译码特殊节点时需要穷举出所有候选路径，导致路径排序的硬件复杂度随节点长度 M 的增大呈指数级增加，节点长度受限。此外，研究者们在 SC 类译码算法的硬件实现架构方面做了大量工作，如树型结构[10]、半并行结构[11]、流水线设计[12] 等。此外，一种可配置的帧间流水线 SCL（dynamic configurable interframe pipelined SCL, DCPSCL）译码器综合考虑了上述结构，有效提高了译码吞吐率[13]。然而，现有工作未能在译码性能、时延和硬件资源消耗之间取得较好的平衡。

与采用固定节点长度的 MSCL 不同[8]，本文提出了一种新的极化码结构，该结构由 Rate0、Rate1、SPC、REP、头部连续冻结位（head consecutive frozen, HCF）和尾部连续信息位（tail consecutive information, TCI）这 6 个节点组成，并据此设计了一种低时延串行抵消列表（low latency successive cancellation list, LLSCL）译码器硬件结构，在保持相近纠错性能的同时，具有较低的译码时延，避免了硬件复杂度高的问题。

1 传统译码算法

极化码是一种线性分组码，可以用 (N,K,A) 表示，其中 $N=2^n,K,A$ 分别表示码长、信息位数量、信息位索引集合。A^c 是 A 的补集，表示可靠性较低的 $N-K$ 个子信道索引集合，即冻结位索引集合，$R=K/N$ 表示码率。极化码的编码过程可以表示为 GF(2) 上的矩阵乘法，即 $\boldsymbol{x}_1^N=\boldsymbol{u}_1^N\boldsymbol{G}$。式中，$\boldsymbol{u}_1^N$ 为信源序列，$\boldsymbol{x}_1^N$ 为编码后的码字序列，$\boldsymbol{G}=\boldsymbol{F}^{\otimes n}$ 是生成矩阵，$\boldsymbol{F}=\begin{bmatrix}1&0\\1&1\end{bmatrix}$，$\otimes$ 表示克罗内克积。u_i，x_i 分别表示 $\boldsymbol{u}$ 和 $\boldsymbol{x}$ 的第 i 个元素，$\boldsymbol{u}_i^j=[u_i,u_{i+1},\cdots,u_j]$，$\boldsymbol{x}_i^j=[x_i,x_{i+1},\cdots,x_j](i\leqslant j)$ 表示子向量。

1.1 SC 译码算法

假设在 B-DMC 信道下的转移概率为 $W(y|x)$，接收端的码字向量为 $\boldsymbol{y}_1^N=[y_1,y_2,\cdots,y_N]$，$\hat{\boldsymbol{u}}_1^N=[\hat{u}_1,\hat{u}_2,\cdots,\hat{u}_N]$ 为信源序列的估计值，则极化子信道的 LLR 的定义为：

$$L_N^{(i)}(\boldsymbol{y}_1^N,\hat{\boldsymbol{u}}_1^{i-1})=\ln\frac{W_N^{(i)}(\boldsymbol{y}_1^N,\hat{\boldsymbol{u}}_1^{i-1}|0)}{W_N^{(i)}(\boldsymbol{y}_1^N,\hat{\boldsymbol{u}}_1^{i-1}|1)} \tag{1}$$

式中，N 表示向量维度；$L_N^{(i)}$ 表示该向量的第 i 个元素，即第 i 个子信道的 LLR。由此可得硬判决公式如下：

$$\hat{u}_i=\begin{cases}0,i\in A^c\text{ or }L_N^{(i)}(\boldsymbol{y}_1^N,\boldsymbol{u}_1^{i-1})\geqslant 0\\1,\text{其他}\end{cases} \tag{2}$$

$N=8$ 的 SC 译码算法结构如图 1 所示，它由 3 个单元组成：f 运算单元、g 运算单元和硬判决单元。操作单元 f/g 上面的数字表示操作顺序。L1 到 L8 为信道输入 LLR。f/g 单元的操作可以表述为：

$$f(L_a,L_b)=\text{sign}(L_a)\text{sign}(L_b)\min\{|L_a|,|L_b|\} \tag{3}$$

$$g(L_a,L_b)=(-1)^{u_{\text{sum}}}L_a+L_b \tag{4}$$

式中，L_a 和 L_b 是输入 LLR 值；sign 表示取符号位；min 表示取最小值；u_{sum} 是上一层比特估计值的线性组合，称为部分和。

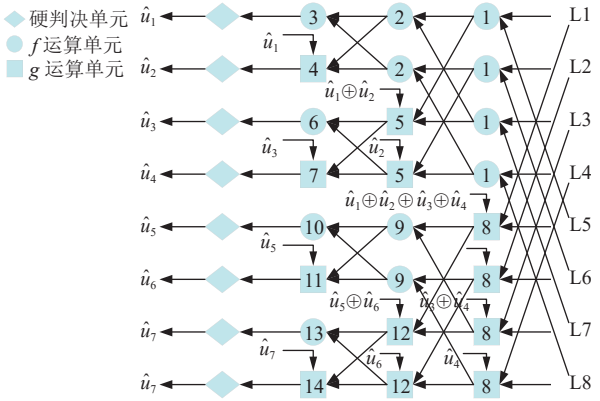


图 1　码长 N=8 的 SC 译码算法架构图

1.2　SCL 译码算法

与 SC 算法不同的是，SCL 算法保留了两种可能的结果，而不是直接执行硬判决。为了评估路径的可靠性，引入了路径度量（path metric, PM）。当候选路径的数目大于列表长度 L 时，只保留具有较小 PM 的 L 条路径。在完成最后一位的译码后，输出具有最小 PM 的路径作为译码结果。PM 的定义和 PM 的更新规则如式 (5) 和式 (6) 所示：

$$\mathrm{PM} = -\ln P(\boldsymbol{u}_1^i \mid \boldsymbol{y}_1^N) \tag{5}$$

$$\mathrm{PM}_l^{(i)} = \begin{cases} \mathrm{PM}_l^{(i-1)}, \hat{u}_i = \psi(L_{N,l}^{(i)}) \\ \mathrm{PM}_l^{(i-1)} + \left| L_{N,l}^{(i)} \right|, \hat{u}_i \neq \psi(L_{N,l}^{(i)}) \\ +\infty, i \in A^c \end{cases} \tag{6}$$

式中，N 表示向量维度；l 表示当前译码路径；$\mathrm{PM}_l^{(i-1)}$ 表示译码器中第 l 条路径第 i 个比特译码之前的路径度量值；PM_l^i 表示第 l 条路径第 i 个比特译码之后的路径度量值；$L_{N,l}^{(i)}$ 表示第 l 条路径的第 i 个比特的 LLR；$\psi(L_{N,l}^{(i)}) = \dfrac{1-\mathrm{sign}(L_{N,l}^{(i)})}{2}$。

N=4, K=4, L=2 的标准 SCL 译码过程如图 2 所示。SCL 译码器的基本思想是在译码过程中保留更多的候选路径，以避免丢失 PM 较小的路径，从而提高译码器的性能。然而，串行译码和频繁路径排序使得 SCL 译码非常耗时。

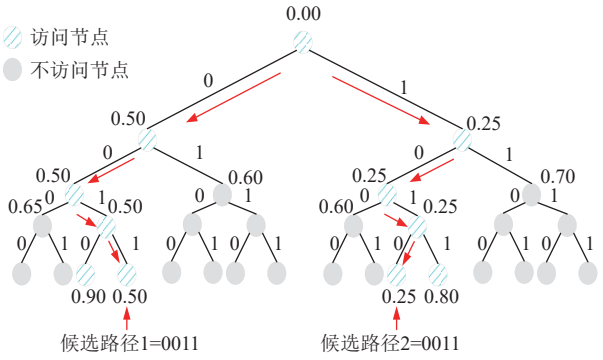


图 2　N=4, K=4, L=2 的 SCL 译码流程

2　LLSCL 译码算法及硬件架构

本节提出了一种基于特殊节点的 LLSCL 译码算法和相应的硬件架构。具体地，对于包含较多冻结位的 Rate0 节点和 REP 节点使用多比特译码；对于包含较多信息位的 Rate1 节点和 SPC 节点使用减分裂方法；同时，针对 HCF 节点和 TCI 节点提出了快速多比特译码方法。

码字的划分结构如图 3 所示，HCF/TCI 节点由头部/尾部连续的冻结/信息位来识别，然后将中间部分划分为 Rate0、REP、Rate1 和 SPC[7] 这 4 种特殊节点。

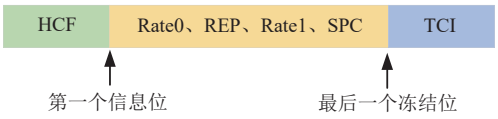


图 3　码字划分结构

不同节点在二叉树中的层数不同，以 v 代表其所在层数，那么 $N_v = 2^v$ 为该节点的长度。向量 $\boldsymbol{I}_v$ 用于表示节点所包含的叶子节点是否为冻结位，$I_v[i]=0$ 表示该位置为冻结位，$I_v[i]=1$ 表示该位置为信息位。那么 Rate0、REP、Rate1、SPC[7] 这 4 种特殊节点的 $\boldsymbol{I}_v$ 可表示为：

Rate0：$\boldsymbol{I}_v = [0,0,\cdots,0,0]$

REP：$\boldsymbol{I}_v = [0,0,\cdots,0,1]$

Rate1：$\boldsymbol{I}_v = [1,1,\cdots,1,1]$

SPC：$\boldsymbol{I}_v = [0,1,\cdots,1,1]$

可以看到 Rate0 节点不包含信息位，REP 节点只包含一个信息位，而 Rate1 和 SPC 节点包含大量的信息位。

2.1　HCF 节点

由于 HCF 节点包含的都是冻结位，所以可以直接在根节点处得到估计的码字序列为全零序列，并且该节点不需要更新路径度量值。因为此时译码器中仅存在一个译码路径，若更新路径度量值只能得到一个初始值，并在遇到第 1 个信息位时执行路径扩展赋值给第 2 条路径。如表 1 所示，不同码长码率下 HCF 节点长度占总码长的 12.5%～25%，可见上述译码方案可减少大量 LLR 递归计算和路径操作。

注意，当且仅当存在 1 条译码路径时，才可以省略冻结比特的 PM 更新，否则会影响最终的译码结果。这也是 HCF 和 Rate0 节点的主要区别。

表 1 不同码长码率下 HCF/TCI 节点长度

R	N			
	128	256	512	1 024
0.3	32/15	63/30	127/55	128/95
0.4	31/23	32/31	64/62	128/125
0.5	16/30	32/55	63/95	127/127

2.2 Rate0 和 REP 节点

文献 [8] 提出另一种 PM 值的计算方法：

$$\sum_{i=1}^{N_v}\ln(1+\mathrm{e}^{-(1-2\hat{u}_i)L_{N_v,l}^{(i)}})=\sum_{i=1}^{N_v}\ln(1+\mathrm{e}^{-(1-2\hat{x}_i)l_{N_v,l}^{(i)}}) \tag{7}$$

式中，N_v表示向量维度；l 表示当前译码路径；$l_{N_v,l}^{(i)}$是特殊根节点的输入 LLR 向量的第 i 个元素；$L_{N_v,l}^{(i)}$是特殊叶节点的输入 LLR 向量的第 i 个元素。由式 (7) 可知，$l_{N_v,l}^{(i)}$和$L_{N_v,l}^{(i)}$对于更新 PM 值具有同样效果。

在极化码编码规则下，Rate0 节点的合法码字为全零序列，REP 节点的合法码字为全零或全一序列。全零码字和全一码字的更新规则分别由式 (8) 和式 (9) 给出， PM_l是译码当前节点时的路径度量，PM_l^*是更新当前节点后的 PM 值：

$$\mathrm{PM}_l^*=\mathrm{PM}_l+\sum_{i=1}^{N_v}\left|l_{N_v,l}^{(i)}\right|,l_{N_v,l}^{(i)}<0 \tag{8}$$

$$\mathrm{PM}_l^*=\mathrm{PM}_l+\sum_{i=1}^{N_v}\left|l_{N_v,l}^{(i)}\right|,l_{N_v,l}^{(i)}\geqslant 0 \tag{9}$$

2.3 Rate1 和 SPC 节点

对于这两种包含较多信息位的特殊节点，若穷举出所有候选路径[8]，则一个长度为N_v的 Rate1 或 SPC 节点会扩展出2^{N_v}或2^{N_v-1}条候选路径，这会带来大量的路径操作。为了避免这种复杂度随节点长度呈指数增长的问题，本文提出了一种减分裂策略：仅在当前节点的前两个信息位上执行路径扩展，其余比特位上直接硬判决译码。图 4 为减分裂策略的示意图。

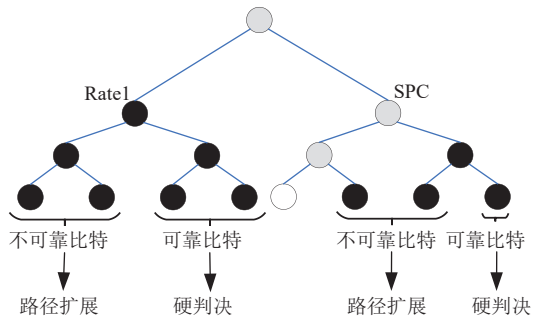


图 4 减分裂策略

在比特翻转 SC 译码算法[14] 中，LLR 最小的位置被认为是不可靠的位置，当译码错误时对 LLR 排序得到不可靠位置并将其翻转，然后重新执行译码过程。受这个想法的启发，本文仅对不可靠的比特执行路径扩展。基于极化信道的偏序关系，信道序号较小的子信道往往不可靠，所以考虑只对当前节点中的前两个信息位执行路径扩展。同时，不可靠的信息位只与子信道序号有关，这避免了对 LLR 值的排序。

2.4 TCI 节点

针对 TCI 节点，本文直接根据其根节点的输入 LLR 序列进行硬判决，即 $\hat{x}_1^{N_v}=\psi(l_{N_v,l}^{(i)}),\forall i\in\{1,2,\cdots,N_v\}$。这只需要一个周期，节省了大量的 LLR 递归计算和路径操作。

考虑到 TCI 是最后一个节点，只需要筛选出一条最优路径作为译码结果。由式 (7) 可知，硬判决得到的路径不会带来 PM 值的增加，而路径扩展一定会带来 PM 值的增加，即硬判决得到的路径为 PM 值最小的最优路径，所以本文不对 TCI 节点包含的信息位执行路径扩展。这与 Rate1 和 SPC 节点不同，因为 Rate1 节点和 SPC 节点不是最后一个节点，需要通过路径扩展和排序来筛选出 L 条候选路径继续译码。如表 1 所示，不同码长码率下的 TCI 节点长度占总码长的 11%～24%，直接执行硬判决将大大降低译码复杂度。

2.5 硬件架构

为了兼容多种码长码率，本文首先将这几种码长码率的冻结位/信息位位置、LLR 计算开始执行层数以及按照图 3 的划分方式划分出的节点信息写入只读存储器中，然后根据输入的配置信息选择对应的先验信息进行译码。本文基于最大码长 1 024 设计可配置译码器，向下兼容短码长，以节省硬件资源。图 5 为本文提出的译码器整体硬件架构。

在标准的 SCL 译码算法[3] 或多比特 SCL 译码算法[8] 中，其部分和计算开始的位置固定在判决层或某一层，但本文所提出的极化码结构中节点长度是不固定的，所以部分和开始执行的层数是不确定的。为此，本文提出一种可兼容各种节点长度的部分和架构，如图 6 所示，通过节点长度控制部分和开始计算的位置，并通过当前的二进制索引判断部分和结束的位置。

phi[i] 表示当前的二进制索引第 i 位，S_i 表示存储第 i 层节点部分和的数组。图 6 中左边的方框

为控制模块，本文用节点长度控制部分和开始执行的位置。右边方框为计算模块，目标寄存器用来存储部分和的最终结果，缓冲寄存器用来存储计算部分和的中间结果。从开始位置即第 v 层，在第 1 个索引为 0 的位置结束部分和的执行并将部分和的结果写入目标寄存器。若索引为 1 则执行部分和，然后将结果写入缓冲寄存器，直至索引为 0。

如当前节点长度为 8，v=3，第 1 个比特的索引值为 phi=1 010 011 110，则 phi[3]=1，phi[4]=1，phi[5]=0，因此，从 S_3 开始执行部分和，到 S_5 结束，中间计算结果存储在缓冲寄存器对应位置的寄存器中，并将最终计算结果存储到目标寄存器 S_5 中。

根据本文所提出的极化码结构，由于节点长度不固定，需要对不同长度的节点更新 PM 值。为了降低复杂度和提高兼容性，本文提出了可兼容多种节点长度的 PM 更新模块，如图 7 所示。本文采用 8 bit 量化，节点长度最大为 128 bit，则输入到节点的 LLR 最多为 1 024 bit，向下兼容较短节点的 PM 更新。通过节点长度灵活配置寄存器右移次数，将 LLR 的低位 8 bit 和 PM 相加 2^v 次即可完成 PM 值的更新，避免为每一种节点长度单独例化模块。这里牺牲了少许时延性能，但提高了资源利用率和可配置性。

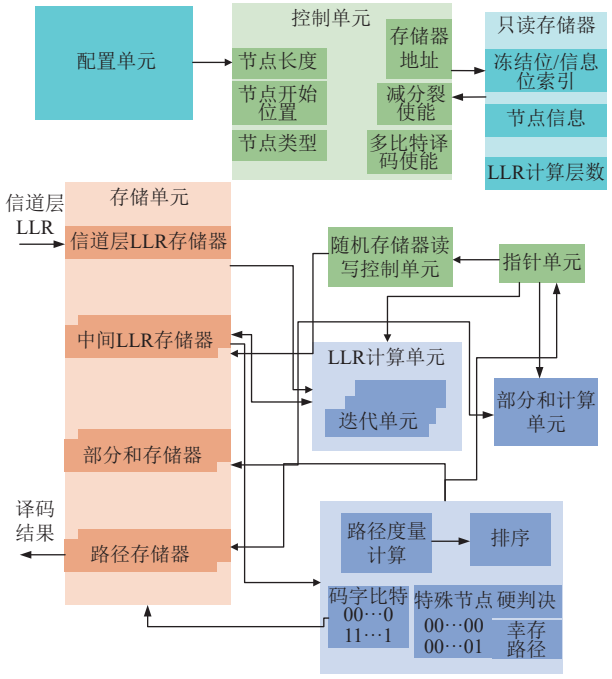


图 5　提出的低时延 SCL 译码器架构

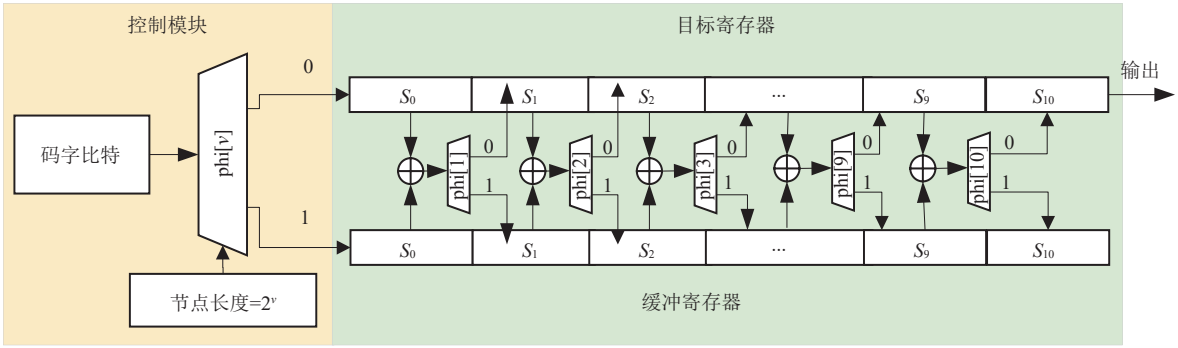


图 6　部分和计算模块

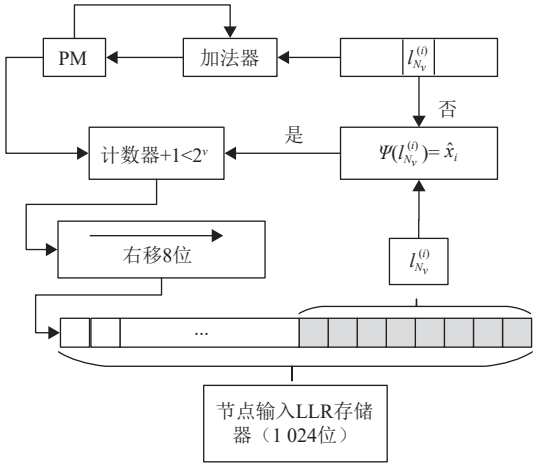


图 7　路径度量更新模块

该兼容多种节点长度的硬件架构的资源消耗主要集中在部分和更新模块以及路径度量更新模块。表 2 展示了在支持 10 种不同节点长度的情况下，兼容性硬件架构和单独例化架构的资源消耗情况以及时间复杂度。

表 2　兼容性硬件架构和单独例化架构的资源消耗以及时间复杂度

模块	资源消耗及复杂度	本文	单独例化
部分和	查找表（LUTs）	1 549	13 192
	寄存器（Registers）	1 679	10 306
	时间复杂度	$\Theta(\log_2(N))$	$\Theta(\log_2(N))$
路径度量	查找表（LUTs）	11 513	14 911
	寄存器（Registers）	4 476	4 860
	时间复杂度	$\Theta(2N)$	$\Theta(N)$

在资源消耗上，兼容性硬件架构的 2 个模块共减少了 53.52% 查找表以及 59.42% 寄存器的资源占用。此外，可以看到 2 种架构部分和更新模块的时间复杂度相同，但兼容性硬件架构的路径度量更新模块的时间复杂度增加了 1 倍，这来源于节点长度的判断。

3 实验与对比

本节基于 Xilinx xezu9eg-ffvb1156-2-i 和 Vivado 2020.1 对所提出的基于特殊节点的 LLSCL 译码器进行硬件实现，并与现有的译码器进行性能对比。在实验中，本文的码构造方法采用极化权重法，使用二进制相移键控（binary phase shift keying, BPSK）调制，列表宽度设为 4。

3.1 硬件实现结果对比

表 3 展示了本文所提出的 LLSCL 译码器和现有译码器的硬件实现结果。由于所有方案都没有使用 DSP 资源，因此没有列出 DSP 的使用率。可以看到，与现有工作相比[15-17]，译码时钟周期分别减少了 67.21%、98.67% 和 66.67%。相比于现有译码器，本文的 LLSCL 译码器占用了更多的查找表和寄存器资源，这是因为本文提出的架构支持多种码长码率。本文译码器架构的吞吐量低于 DCPSCL[13]，这是由于 DCPSCL 使用了帧间流水线架构，这将作为我们以后的改进方向。

表 3 本文所提出的译码器架构和现有译码器的硬件实现结果

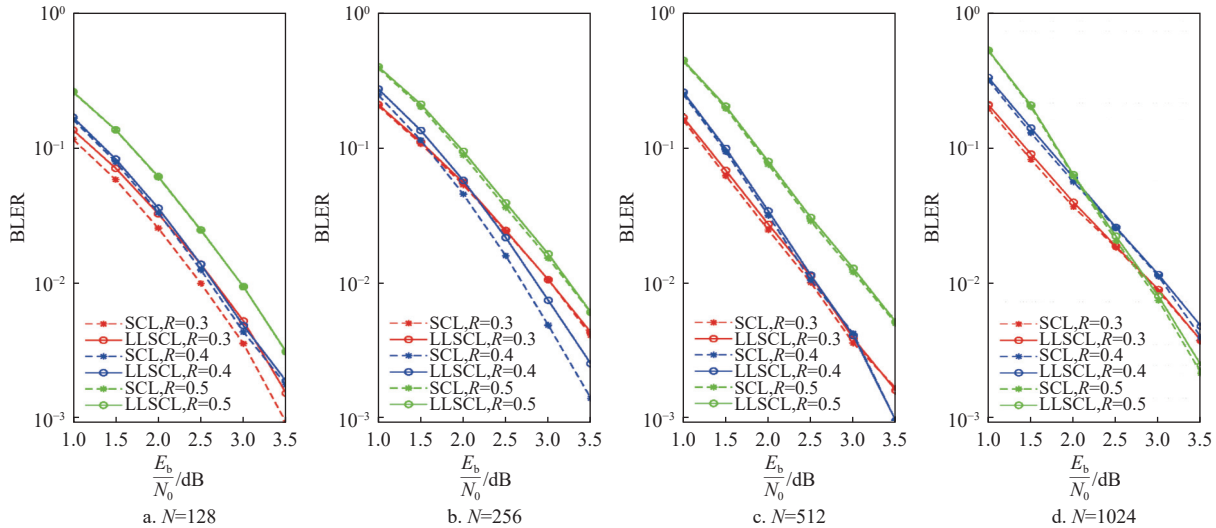
硬件实现结果	本文	文献[13]	文献[15]	文献[16]	文献[17]	标准SCL译码器
采用的硬件开发平台（N, R, L）	Zynq UltraScale+（1 024, 0.5, 4）	Zynq UltraScale+（1 024, 0.5, 4）	Zynq-7000（1 024, 0.5, 8）	Stratix V（1 024, 0.5, 4）	Zynq-7000（1 024, 0.5, 8）	Zynq UltraScale+（1 024, 0.5, 4）
查找表（LUTs）	83 976	89 088	2 782	8 146	6 571	76 117
寄存器（Registers）	33 847	30 322	1 163	2 862	1 265	31 650
译码所需时钟周期	6 999	10 436	21 344	524 288	20 992	12 460
工作频率/MHz	166.7	216.7	154	445.2	156	100
吞吐量/Mbps	24.39	119.2	7.39	0.871	7.43	8.22

3.2 时延和译码性能分析

表 4 展示了本工作和标准 SCL 译码器译码周期减少比例对比。可以看到，译码周期在不同码长和码率下减少了 40.32%～56.87%，这是由于针对特殊节点使用了剪枝和减分裂方法。图 8 展示了所提出的 LLSCL 译码器和标准 SCL 译码器在加性高斯白噪声信道下的误块率（block error ratios, BLER）。可以看出，该译码器的性能非常接近于标准的 SCL 译码器，特别是在码率较高的情况下。最大的性能差距出现在 N=128 和 R=0.3 时，但也只有 0.2 dB。因此，所提出的低时延 SCL 译码器可以在几乎不损失译码性能的情况下显著降低时延，并支持多种码长和码率。

表 4 本工作和标准 SCL 译码器相比译码周期减少比例

R	N			
	128	256	512	1 024
0.3	56.74%	41.51%	53.29%	56.87%
0.4	40.32%	51.06%	47.13%	48.31%
0.5	43.02%	42.64%	40.80%	43.83%



a. N=128 b. N=256 c. N=512 d. N=1024

图 8 LLSCL 和标准 SCL 译码器的误块率性能对比

4　结束语

为了在不降低 SCL 译码器性能的前提下减少译码时延，本文提出了一种 LLSCL 译码器。本文对码字设计了新的特殊节点划分方式，并针对特殊节点提出多种译码策略：对 HCF、TCI、REP 和 Rate0 节点进行多比特译码；为了避免现有多比特译码方案带来的硬件复杂度极大增加，在 SPC 和 Rate1 节点上执行减分裂策略。所提出的硬件架构基于最大节点长度设计，兼容较短节点长度的路径度量和部分和的更新，以提高资源利用率，实验结果验证了该架构的优越性能。此外，本文对译码器进行可配置设计，以兼容多种码长码率。实验结果表明，与现有的译码器相比，该译码器在不同码长和码率下均能有效降低译码时延，且几乎没有性能损失。

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 K, CHEN K. CRC-aided decoding of polar codes[J]. IEEE Communications Letters, 2012, 16(10): 1668-1671.

[4] 太云飞. 极化码与 CRC 码的级联方法及其译码算法研究[D]. 成都: 电子科技大学, 2023.
TAI Y F. Research on the cascading method of polar and CRC codes and the corresponding decoding algorithm[D]. Chengdu: University of Electronic Science and Technology of China, 2023.

[5] ALAMDAR-YAZDI A, KSCHISCHANG F R. A simplified successive-cancellation decoder for polar codes[J]. IEEE Communications Letters, 2011, 15(12): 1378-1380.

[6] 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.

[7] SARKIS G, GIARD P, VARDY A, et al. Fast list decoders for polar codes[J]. IEEE Journal on Selected Areas in Communications, 2015, 34(2): 318-328.

[8] YUAN B, PARHI K K. LLR-based successive-cancellation list decoder for polar codes with multibit decision[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2016, 64(1): 21-25.

[9] YUAN B, PARHI K K. Low-latency successive-cancellation list decoders for polar codes with multibit decision[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2014, 23(10): 2268-2280.

[10] LEROUX C, TAL I, VARDY A, et al. Hardware architectures for successive cancellation decoding of polar codes[C]//2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Prague: IEEE, 2011: 1665-1668.

[11] MISHRA A, RAYMOND A J, AMARU L G, et al. A successive cancellation decoder ASIC for a 1024-bit polar code in 180nm CMOS[C]//2012 IEEE Asian Solid State Circuits Conference (A-SSCC). Kobe: IEEE, 2012: 205-208.

[12] HUANG Y, ZHANG Z, YOU X, et al. Efficient folded SC polar line decoder[C]//2018 IEEE 23rd International Conference on Digital Signal Processing (DSP). Shanghai: IEEE, 2018: 1-5.

[13] YE L, ZHAI X, ZENG Z, et al. High-throughput polar code SCL decoding with dynamically configurable pipeline architecture[C]//2023 IEEE/CIC International Conference on Communications in China (ICCC). Dalian: IEEE, 2023: 1-6.

[14] AFISIADIS O, BALATSOUKAS-STIMMING A, BURG A. A low-complexity improved successive cancellation decoder for polar codes[C]//2014 48th Asilomar Conference on Signals, Systems and Computers. Pacific Grove: IEEE, 2014: 2116-2120.

[15] FENG Z, NIU C, ZHANG Z, et al. List-serial pipelined hardware architecture for SCL decoding of polar codes[J]. China Communications, 2023, 20(3): 175-184.

[16] LIANG X, WANG H, SHEN Y, et al. Efficient stochastic successive cancellation list decoder for polar codes[J]. Science China Information Sciences, 2020, 63: 1-19.

[17] BALATSOUKAS-STIMMING A, PARIZI M B, BURG A. LLR-based successive cancellation list decoding of polar codes[J]. IEEE Transactions on Signal Processing, 2015, 63(19): 5165-5179.

责任编辑　张　莉