



研究与开发

极化码的多比特译码算法研究

沈周青, 尚俊娜

(杭州电子科技大学通信工程学院, 浙江 杭州 310018)

摘要: 针对极化码的连续消除列表 (successive cancellation list, SCL) 译码算法的高时延问题, 提出了基于对数似然比的多比特 SCL (multi-bit SCL, MSCL) 译码算法, 可以在一个判决时刻同时译出多个码字比特, 在不损失译码性能的前提下, 将译码时延由 $3N-2$ 个时钟降为 $4N/M-2$ 个时钟, 相比于现有的多比特 SCL 译码算法, MSCL 译码算法具有更低的路径度量值计算复杂度。为了降低循环冗余校验 (cyclical redundancy check, CRC) 辅助的 SCL (CRC aided SCL, CA-SCL) 译码算法的译码时延以及存储空间, 提出了分段 CRC 辅助的 MSCL (segmented CRC-aided MSCL, SCA-MSCL) 译码算法, 并提出了分段信息码字长度修正算法, 来保证在信息位索引集 A 不变的前提下, 实现每一分段结尾处对应的信息位索引能够被 M 整除。SCA-MSCL 算法可以借助多个 CRC 判决来尽可能早地输出译码码字, 从而减少译码器的存储空间以及译码时延。

关键词: 极化码; 多比特编码; MSCL; 分段 CRC; 译码时延; 存储空间

中图分类号: TP393

文献标识码: A

doi: 10.11959/j.issn.1000-0801.2018261

Research on multi-bit decoding algorithms for polar codes

SHEN Zhouqing, SHANG Junna

College of Telecommunication Engineering, Hangzhou Dianzi University, Hangzhou 310018, China

Abstract: Aiming at the problem of high delay of successive cancellation lists (SCL) decoding algorithm for polar codes, a multi-bit SCL (MSCL) decoding algorithm based on log-likelihood ratio was proposed. The multiple codeword bits could be decoded simultaneously at a single decision time. Without loss of SCL decoding performance, the decoding delay was reduced from $3N-2$ clocks to $4N/M-2$ clocks. Compared with the existing multi-bit SCL decoding algorithm, the MSCL decoding algorithm had a lower complexity of the path metric calculation. In order to reduce the decoding delay and memory space of the cyclical redundancy check (CRC) assisted SCL (CA-SCL) decoding algorithm, a segmented CRC assisted MSCL (SCA-MSCL) decoding algorithm was proposed, and a segmentation information codeword length correction algorithm was proposed. By adjusting the length of the relevant segment, the information bit index set A was guaranteed. Under the same premise, the corresponding information bit index at the end of each segment could be divisible by M . The SCA-MSCL algorithm could output decoded code words as early as possible by using multiple CRC decisions, thereby reducing the decoder's storage space and decoding delay.

Key words: Polar code, multi bit decoding, MSCL, segmented cyclical redundancy check, decoding delay, storage space

收稿日期: 2018-03-12; 修回日期: 2018-09-28

基金项目: 国家自然科学基金资助项目 (No.11603041)

Foundation Item: The National Natural Science Foundation of China (No.11603041)



1 引言

极化码是唯一一种被证明可以在任意离散无记忆信道中实现信道容量的编码方式^[1]。但是,在连续消除(successive cancellation, SC)译码算法条件下,只有当码长足够长时,极化码才能达到信道容量^[2]。相比于SC译码算法,参考文献[3]提出的串行抵消列表(successive cancellation list, SCL)译码算法具有明显的译码性能提升。然而,SCL译码器具有类似于早期SC译码器的长延迟和低吞吐量问题,且相应的译码时延为 $3N-2$ 个时钟周期^[4]。

有很多技术^[5-7]已经被提出来缩短SC译码算法的译码时延,但是这些技术不能直接应用于SCL译码。所以,很多学者在SCL译码基础上做了改进译码时延的研究,参考文献[8]提出了基于符号的SCL译码算法,该译码算法是通过信道合并的方式来计算符号转移概率的。参考文献[9]提出了基于似然值的多个比特同时译码的SCL译码算法,从而有效地降低了SCL译码算法的译码时延。参考文献[10]对SCL译码器的硬件结构设计进行了一系列的研究,基于对数似然比(log-likelihood ratio, LLR)的SCL译码算法的提出可以有效地降低译码器的硬件复杂度^[14]。为了同时降低SCL译码器的译码时延以及硬件复杂度,参考文献[12]提出了一种降低时延的基于LLR的SCL译码算法。

为了降低SCL译码算法的时延,提出了一种新的基于LLR的多比特SCL(multi-bit SCL, MSCL)译码算法,该算法可以在一个判决时刻同时译出多个码字比特。同时,MSCL译码算法通过对候选路径正确性的判断来降低路径计算的复杂度。为了降低CA-SCL译码算法的译码时延以及存储空间,提出了分段CRC辅助的MSCL(segmented CRC-aided MSCL, SCA-MSCL)译码算法,该算法要求在编码时对信息比特进行相应的分段。SCA-MSCL译码算法可以借助多个CRC

校验码进行判断,在每一段译码结束时,即进行相应的CRC校验,并选择最符合条件的一条路径输出,同时只保存该路径的估计值供后一段译码使用,从而可以减少译码器的存储空间以及译码时延。

2 极化码的多比特译码算法

2.1 MSCL译码算法

参考文献[12]给出了一种降低时延的基于LLR的SCL译码算法,参考文献[12]的算法将这种译码算法称为LLR- 2^k b-SCL译码算法。同时,可以将SCL译码算法的译码时延从 $3N-2$ 降为 $N/2^{k-2}-2$ 个时钟。参考文献[12]中的算法首先需要计算 $2^k \cdot L$ 条候选路径的度量值,随后对所有候选路径的度量值进行修正操作,修正的条件为在当前所需译码的 2^k 比特中有几位是冻结比特,而在候选路径中,这几位冻结比特取得的值并不为0,即使这些候选路径对应的路径度量值足够大,都不应该被选择,所以将这些候选路径的度量值设置为 $-\text{inf}$ 。在完成候选路径的度量值计算跟修正以后,参考文献[12]的算法需要在 $2^k \cdot L$ 条候选路径中选择 L 条度量值最大的路径并保存。参考文献[12]的算法存在一定的缺陷,即当前需要译的 2^k 比特码字中,可能部分为冻结比特,或者全部为冻结比特,那么按照LLR- 2^k b-SCL算法同样需要从 $2^k \cdot L$ 条候选路径选择 L 条度量值最大的路径并保存。

针对上述情况,提出了一种新的基于LLR的 $M(M=2^m)$ 比特SCL译码算法(multi-bit successive cancellation list, MSCL),在扩展候选路径时,根据冻结比特删去无效的候选路径,使得每次扩展路径不多于 $2^M \cdot L$ 条。接下来,首先介绍MSCL算法如何通过同一时刻的LLR来实现 $\hat{u}_{M-(i-1)+1}, \dots, \hat{u}_{M-i}$ 共 M 比特码字同时译码的过程。从极化码的译码码树^[12]可知,SC译码器可以通过长为 $M \cdot (i-1)$ 路径的度量值直接计算得到长为 $M \cdot i$ 路径的度量值,MSCL译码器可以由 L 个相同的SC译码器组合而成。一般而言,可以通过一

个基于 LLR 的度量计算单元 (metric computation unit, MCU) 来完成长为 $M \cdot i$ 路径的度量值计算, 那么, MCU 可以代替 SC 译码器的最后 m 个阶段。接下来, 将推导如何实现 MSCL 译码过程中, 长为 $M \cdot i$ 路径的度量值计算。

假设 $M \cdot (i-1)$ 个之前被译出码字 $\hat{u}_{M \cdot (i-1)+1}, \dots, \hat{u}_{M \cdot i}$ 的值分别为 $z_{M \cdot (i-1)+1}, \dots, z_{M \cdot i}$, 且记为 $\hat{u}_1^{M \cdot (i-1)} = z_1^{M \cdot (i-1)}$, 那么在对数域中, 长为 $M \cdot i$ 路径的度量值可以表示为:

$$\begin{aligned} \text{PM}^{(i,M)}(\alpha_1 \dots \alpha_M, z_1^{M \cdot (i-1)}) = \\ \ln(\Pr(\hat{u}_{M \cdot (i-1)+1}^{M \cdot i} = \alpha_1^M, \hat{u}_1^{M \cdot (i-1)} = z_1^{M \cdot (i-1)})) \end{aligned} \quad (1)$$

其中, $\hat{u}_{M \cdot (i-1)+1}^{M \cdot i}$ 表示为 $(\hat{u}_{M \cdot (i-1)+1}, \dots, \hat{u}_{M \cdot i})$, 且 $\hat{u}_{M \cdot (i-1)+1}^{M \cdot i}$ 表示当前 M 比特译码集合。另外, α_1^M 表示为 $(\alpha_1, \dots, \alpha_M)$, 其元素都为二进制的数。

式 (1) 中包含着译码过程中未知的当前 M 比特译码码字的概率信息, 为了解决这一问题, 需要进一步说明路径的度量值与输入 MCU 的 LLR 之间的关系。参考文献[15]指出极化码译码过程本质上是通过编码来引导进行的, 如果 $u_{M \cdot (i-1)+1}^{M \cdot i}$ 的估计值表示为 α_1^M , 那么输出 $\widehat{\text{out}}_1^M = (\widehat{\text{out}}_1, \dots, \widehat{\text{out}}_M)$ 可以表示为 $\alpha_1^M \mathbf{G}_M$, 其中 $\mathbf{G}_M$ 表示的是 $M \times M$ 的生成矩阵。那么式 (1) 可以重写为式 (2):

$$\begin{aligned} \text{PM}^{(i,M)}(\alpha_1 \dots \alpha_M, z_1^{M \cdot (i-1)}) = \\ \ln\left(\Pr\left(\widehat{\text{out}}_1^M = \alpha_1^M \mathbf{G}_M, \hat{u}_1^{M \cdot (i-1)} = z_1^{M \cdot (i-1)}\right)\right) \end{aligned} \quad (2)$$

假设 $\mathbf{G}_M(j)$ 代表 $\mathbf{G}_M$ 的第 j 列, 可以得到 $\widehat{\text{out}}_i = \alpha_{M \cdot (i-1)+1}^{M \cdot i} \mathbf{G}_M(j)$, 所以, 将式 (2) 改写为:

$$\begin{aligned} & \text{PM}^{(i,M)}(u_{M \cdot (i-1)+1}^{M \cdot i}, z_1^{M \cdot (i-1)}) \\ &= \ln\left(\Pr\left(\widehat{\text{out}}_1^M = u_{M \cdot (i-1)+1}^{M \cdot i} \mathbf{G}_M \mid \hat{u}_1^{M \cdot (i-1)} = z_1^{M \cdot (i-1)}\right)\right) \\ &= \ln\left(\Pr\left(\hat{u}_1^{M \cdot (i-1)} = z_1^{M \cdot (i-1)}\right)\right) \\ &= \sum_{j=1}^M \ln\left(\Pr\left(\widehat{\text{out}}_j = u_{M \cdot (i-1)+1}^{M \cdot i} \mathbf{G}_M(j) \mid \hat{u}_1^{M \cdot (i-1)} = z_1^{M \cdot (i-1)}\right)\right) \\ &+ \text{PM}^{(i-1,M)} \end{aligned} \quad (3)$$

其中, $\text{PM}^{(i-1,M)}$ 代表的是长度为 $M \cdot (i-1)$ 的路径对应的度量值。

由于采用的译码器是基于 LLR 执行的, 那么 MCU 的第 j 个输入可以表示为:

$$L_{N/M}^{(i,j)} = \ln\left(\frac{\Pr(\widehat{\text{out}}_j = 0 \mid \hat{u}_1^{M \cdot (i-1)} = z_1^{M \cdot (i-1)})}{\Pr(\widehat{\text{out}}_j = 1 \mid \hat{u}_1^{M \cdot (i-1)} = z_1^{M \cdot (i-1)})}\right) \quad (4)$$

参考 LLR-SCL 译码算法对于路径度量值的计算过程, 将式 (3) 改写为式 (5):

$$\begin{aligned} \text{PM}^{(i,M)} &= \text{PM}^{(i-1,M)} + \sum_{j=1}^M L_{N/M}^{(i,j)} \\ &= \text{PM}^{(i-1,M)} + \sum_{j=1}^M c_{i,j} \left| L_{N/M}^{(i,j)} \right| \end{aligned} \quad (5)$$

式 (5) 表示的是在进行 M 比特同时译码时, 第 $i(i \in \{1, 2, \dots, N/M\})$ 次判决时的度量值, 对长度为 N 的极化码进行 SC 译码时, 共需要经历 $n = \text{lb}N$ 个阶段, 假设 $L_N^{(i)}$ 表示的是阶段 n 输出的 LLR 值。因此, 式 (5) 中 $L_{N/M}^{(i,j)}$ 表示的是阶段 $(n-m)$ 输出的第 j 个 LLR 值。令 $\text{PM}^{(0,M)} = 0$, 即初始度量值为 0, 当 $\widehat{\text{out}}_{M \cdot (i-1)+j} = \frac{1}{2} \cdot (1 - \text{sgn}(L_{N/M}^{(i,j)}))$ 时, $c_{i,j} = 1$; 当 $\widehat{\text{out}}_{M \cdot (i-1)+j} \neq \frac{1}{2} \cdot (1 - \text{sgn}(L_{N/M}^{(i,j)}))$ 时, $c_{i,j} = 0$ 。

式 (5) 给出了如何从长度为 $M \cdot (i-1)$ 路径的度量值直接计算长度为 $M \cdot i$ 路径的度量值。MSCL 译码器是由 L 个相同的基于 LLR 的 SC 译码器组成的。为了译出连续的 M 个码字, 每个 SC 译码器模块首先需要像传统 SC 译码器一样执行, 直到阶段 $(n-m)$ 时停止, 在阶段 $(n-m)$ 时, SC 译码器可以向 MCU 提供 M 个 LLR 信息 $L_{N/M}^{(i,j)} (j=1, 2, \dots, M)$, 然后在每个 SC 译码器中的 MCU 可以根据式 (5) 来计算当前路径新的度量值。通过比较 L 个 SC 译码器计算产生的所有路径的度量值, 并选择 L 个具有最大的度量值路径作为幸存路径。综上, 完整的 MSCL 算法如下所示。

算法 1: MSCL 译码算法

输入: 每一个接收码字的 LLR



初始化: $PM^{(0,M)} = 0$; $i = 1$; $l = 1$

while: $i \leq N/M \ \& \ l \leq L$

对于第 l 条幸存路径, 假设 $M \cdot (i-1)$ 个之前被译出码字 $\hat{u}_{M \cdot (i-1)+1}, \dots, \hat{u}_{M \cdot i}$ 的值分别为 $z_{M \cdot (i-1)+1}, \dots, z_{M \cdot i}$, 且记为 $\hat{u}_1^{M \cdot (i-1)} = z_1^{M \cdot (i-1)}$

激活基于 LLR 的 SC 译码器的阶段 1 到阶段 $(n-m)$, 在第 $(n-m)$ 阶段输出 M 个 LLR 的值 $L_{N/M}^{(i,j)}$ ($j=1, 2, \dots, M$)

将幸存路径 $z_1^{M \cdot (i-1)}$ 扩展成 2^M 条候选路径 ($\hat{u}_1^{M \cdot i} = (\alpha_1^M, z_1^{M \cdot (i-1)})$)

如果 $\hat{u}_{M \cdot (i-1)+j}$ 是冻结比特, 将路径 $(\alpha_1 \alpha_2 \dots \alpha_{j-1} \dots \alpha_{j+1} \dots \alpha_M, z_1^{M \cdot (i-1)})$ 从候选路径中去除, 那么一条长度为 $M \cdot (i-1)$ 的路径至多可以扩展成 2^M 条长度为 $M \cdot i$ 的路径

通过式 (5) 来计算每一条候选路径的度量值
end while

对至多 $2^M \cdot L$ 条长度为 $M \cdot i$ 的候选路径根据其度量值进行从大到小排序, 并选择 L 条度量值最大的路径作为新的幸存路径

end while

输出: 选择长度为 N 且度量值最大的路径

2.2 MSCL 译码算法的译码时延及复杂度分析

在进行一次判决时, MSCL 译码算法可以同时译出 $M = 2^m$ 个码字。与 SCL 译码算法相比, MSCL 译码算法可以通过一次路径选择替代 SCL 译码算法中的 M 次选择, 因此节省的时钟为 $t_1 = M - 1$ 。MSCL 译码算法需要额外的一个时钟进行路径度量值的计算, 因此增加的时钟为 $t_2 = 1$ 。因为在 MSCL 译码过程中, 最后的 m 阶段的 LLR 更新已经被去掉, 所以可以节省的时钟为 $t_3 = 2 \cdot M - 2$ 。因此, MSCL 译码算法的译码时延可以表示为:

$$\begin{aligned} T &= (3 \cdot N - 2) - (t_1 + t_2 + t_3) \\ &= (3 \cdot N - 2) - \sum_{i=1}^{N/M} (3 \cdot M - 4) \quad (6) \\ &= \frac{4 \cdot N}{M} - 2 = N/2^{m-2} - 2 \end{aligned}$$

由式 (6) 可知, MSCL 译码算法与参考文献[12]的算法降低时延效果一致。但是, MSCL 译码算法通过提前将不符合条件的候选路径去掉, 有效地降低了路径度量值计算以及度量值比较的复杂度。假设 MSCL 译码算法将译码码字分成 E 组, C_i 表示第 i 组中包含的信息比特数量, 其中 $i \in \{1, 2, \dots, E\}$, 且 $E = N/M$, 那么相比于参考文献[12]的算法, MSCL 译码算法对于候选路径度量值计算次数的减少量可以表示为:

$$\tau = 2^M \cdot L \cdot E - \sum_{i=1}^E 2^{C_i} \cdot L \quad (7)$$

3 极化码的分段 CRC 辅助的多比特译码算法 (SCA-MSCL)

3.1 SCA-MSCL 译码算法

CA-SCL 译码算法具有比 SCL 译码算法更好的性能, 然而, CA-SCL 译码算法同样具有类似于 SCL 译码算法的高时延问题, 同时, 在译码过程中需要保存 L 条路径, CA-SCL 译码算法需要占用较大的存储空间。针对以上问题, 在 MSCL 译码算法的基础上提出了分段 CRC 辅助的 MSCL 译码算法 (segmented CRC-aided MSCL, SCA-MSCL)。具体来说, 提出了一种新的基于 CRC 的极化码构造方法, 借助于码字中间多个 CRC 校验尽可能早地选择译码码字, 在每一段译码结束时, 即进行相应的 CRC 校验, 选择最符合条件的一条路径输出, 同时保存该路径的估计值, 从而节省存储空间以及进一步降低译码时延。

图 1 表示在不同译码算法条件下, 极化码编码器输入信息序列的结构。如图 1(a)所示, 传统的 CA-SCL 译码算法是直接将 CRC 校验码字放在信息比特的末尾; 而 SCA-MSCL 译码算法是将校验码字分插在信息比特段中, 且都处于每一信息段的末尾。如图 1(a)所示, 在极化码编码之前, 信息段应由一个 CRC 编码器处理, 其中 r 个 CRC 校验位用于传统 CA-SCL 译码算法。图 1(b)将信息段均匀地拆分为 S 个子段, 为保证相同的码率,

通过一个可以生成 r/S 个校验位的 CRC 编码器处理每个子块。

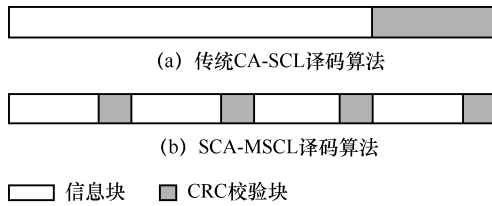


图1 极化码编码器输入信息序列的结构

在进行极化码编码时，需要将校验位设为相应的信息位，那么 $(N, K-r, S, r)$ 的极化码需要通过高斯近似方法^[16]来获得 K 个信息位索引集 A ，并对索引集 A 进行从小到大的排序。首先，对 K 个信息比特进行均匀分段，那么每一段的最后一位信息位对应的索引值为 $A_{i:K/S}$ ($1 \leq i \leq S$)，根据实际仿真会发现 $A_{i:K/S}$ ($1 \leq i \leq S$) 存在不能被 M 整除的情况，而在使用 MSCL 译码算法时，需要保证每一分段的长度都能被 M 整除。因此，提出了一种信息码字分段修正方法，在保证所选信息位索引集 A 不变的前提下，通过改变相关段信息比特的长度来使得每一分段结尾处对应的信息位索引都能被 M 整除。相应的算法实现如下所示，其中， D_i^S 表示的是各分段所含有的信息比特的长度，函数 $\text{mod}()$ 和 $\text{ceil}()$ 分别表示取余和向上取整操作。

算法 2：分段信息比特长度修正算法

输入：通过高斯近似方法来获得 K 个信息位索引集 A

初始化：对索引集 A 中的信道索引值进行从小到大的排序； $i=1$

```
while  $i \leq S$ 
    if  $\text{mod}(A_{i:K/S}, M) \neq 0$ 
        temp =  $\text{ceil}(A_{i:K/S} / M) \times M$ 
        while 当 temp 不在集合  $A$  中
            temp = temp +  $M$ 
        end while
         $B_i = \text{temp}$ 
```

else

$$B_i = A_{i:K/S}$$

end if

if $i=1$

D_i 等于 B_i 在 A 中的位置索引

else

D_i 等于 B_i 在 A 中的位置索引减去

B_{i-1} 在 A 中的位置索引

end if

end while

输出： D_1^S

在完成输入信息序列分段处理之后，进行相应的极化码编码，接下来将介绍 SCA-MSCL 的算法，该译码算法的主要思想为，在发送端根据分段信息比特长度修正算法对信息比特进行合理分段，在各分段的末尾加入同等长度的 CRC 校验位，将处理后的信息比特进行极化码编码。最后，在译码端，每一段译码结束时，即进行相应的 CRC 校验，选择最符合条件的一条路径输出，同时保存该路径的估计值。SCA-MSCL 的算法描述如下所示，其中 $\theta_s = \sum_{i=1}^s D_i$ 。

算法 3：SCA-MSCL 译码算法

输入：每一位接收码字的对数似然比

初始化： $s=1$

while $s \leq S$

if $s=1$

start = 1

else

$$\text{start} = A_{\theta_{s-1}} + 1$$

end if

if $s=S$

last = N

else

$$\text{last} = A_{\theta_s}$$

end if



$i = \text{start}$

while $i \leq \text{last}$

MSCL 译码器根据保存的前 $s-1$ 子段的路径信息以及当前译码保存的路径信息进行相应的译码, 并获得对应路径的度量值

if $i = \text{last}$

对于子段 s 中 L 条路径进行 CRC 校验, 选择并保存满足如下条件的路径, 并输出相应的路径估计值:

- 当 L 条路径中只有一条是可以通过 CRC 校验的, 则选择该路径;
- 当 L 条路径中没有一条是可以通过 CRC 校验的, 则选择所有路径中度量值最大的那一条;
- 当 L 条路中有多条路径是可以通过 CRC 校验的, 则选择所有通过校验的路径中度量值最大的那一条

将该路径的路径估计值作为子段 s 的译码码字输出, 并作为下一子段译码时的输入

end if

end while

end while

输出: 每一段的译码码字

3.2 SCA-MSCL 译码算法的存储空间及译码时延分析

图 2 是 CA-SCL 译码算法在译码过程中需要占用的内存空间。这里不考虑译码算法在执行过程中保存的路径数未达到 L 时所占存储空间的误差, 因为随着译码比特数的增加, 保存路径数是呈指数增加的, 即保存的路径数很快会达到 L 。译码执行到第 i 步时, 需要 $i \cdot L$ 的内存空间存储路径, 由于 CA-SCL 算法的译码特点, 只有当存储路径长度达到 N 时, 才会将所有的译码码字输出, 因此 CA-SCL 算法在译码的过程中需要占用的存储空间为 $N \cdot L$ 。

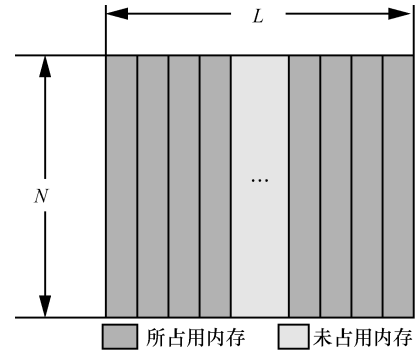


图 2 CA-SCL 译码的存储结构

对于 SCA-MSCL 译码算法, 图 3 是其在译码过程中所占内存空间的情况, 该算法将译码路径划分成了 S 个子段, 每译完一段, 便进行 CRC 校验, 选择并保留该段中一条符合要求的路径, 将不符合的路径舍弃, 同时将该段的译码比特输出。从图 3 可知, 由于在完成分段以后, 各分段所包含的信息位长度几乎一致, 但是分段的总码长并不一致, 所以选择长度最长的子段对应的内存空间作为 SCA-MSCL 算法在译码过程中所需占用的内存空间, 即所占用的内存空间为 $(L-1) \cdot \max\{(A_{D_{bs}} - A_{D_{bs-1}}), A_{D_1}\} + N$, 其中 $2 \leq s \leq S$ 。

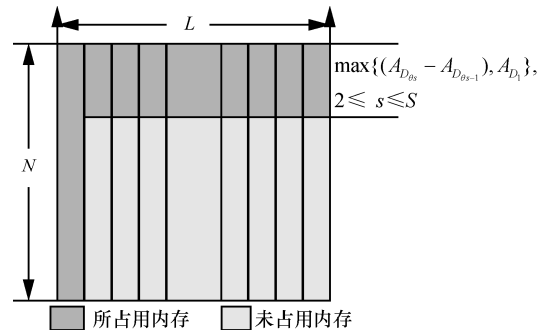


图 3 SCA-MSCL 译码的存储结构

主要是通过平均译码时延来探讨两种方案的性能。只要子段完成译码, SCA-MSCL 译码算法就会立即将该子段的码字序列输出。而 CA-SCL 译码算法需要将所有 N 个码字译完后才输出最终的码字。每一分段的平均时延 $\bar{T}$ 可以定义如下:

$$\bar{T} = \frac{1}{S} \cdot (t_1 + t_2 + \cdots + t_i + \cdots + t_S) \quad (8)$$

其中, t_i 是第 i 个子段完成译码的时延。SCA-MSCL 译码算法是在 MSCL 译码算法的基础上提出的, 该算法在每个分段译码完成时需要额外的时延进行 CRC 校验, 假设 CRC 校验所需要的时延与 CRC 校验码的码字长度一致^[17]。因此, 已知 MSCL 译码算法的时延为 $4 \cdot N/M - 2$, 那么 SCA-MSCL 译码算法的总译码时延为 $4 \cdot N/M - 2 + r$ 。因此, 第 i 个子段的译码时延可以定义如下:

$$t_i = \frac{A_{\theta_i}}{N} \cdot (4 \cdot N/M - 2 + r) \quad (9)$$

综上, 可以将 SCA-MSCL 译码算法的平均译码时延改写为:

$$\bar{T} = \frac{1}{S} \cdot \sum_{i=1}^S \frac{A_{\theta_i}}{N} \cdot (4 \cdot N/M - 2 + r) \quad (10)$$

CA-SCL 译码是在 SCL 译码算法的基础上提出的, CA-SCL 译码算法在所有译码完成时需要额外的时延进行 CRC 校验。因此, 相较于 SCL 译码算法的 $3 \cdot N - 2$ 的时延, CA-SCL 的译码算法的总译码时延为 $3 \cdot N - 2 + r$ 。

4 算法的仿真分析

4.1 MSCL 译码算法的仿真结果与分析

图 4 表示的是与 SCL 译码算法相比, MSCL 译码算法的时延降低程度。由图 5 可知, 在 M 一致时, 随着码长的增加, 对应的译码时延降低的百分比基本不变。而当 M 增加时, 相应的译码时延降低的百分比增加。由图 4 可知, 当 M 分别为 2、4、8、16、32 时, MSCL 译码算法相比于 SCL 译码算法分别可以降低 35%、69%、85%、95%、98% 的译码时延。

图 5 表示的是与参考文献[12]的算法相比, MSCL 译码算法的路径度量值计算减少的程度, 其中极化码码长都为 1 024。由图 5 可知, 当 M

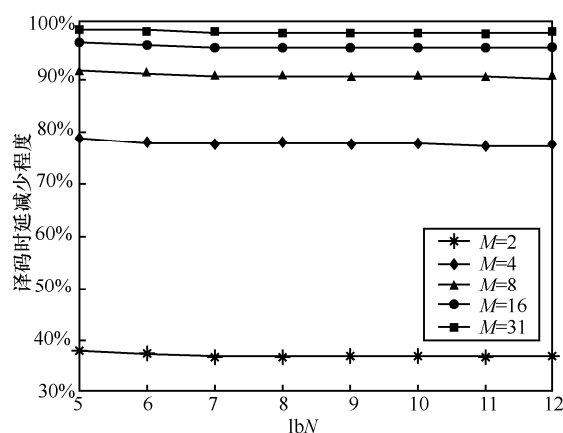


图 4 MSCL 译码算法时延降低的程度

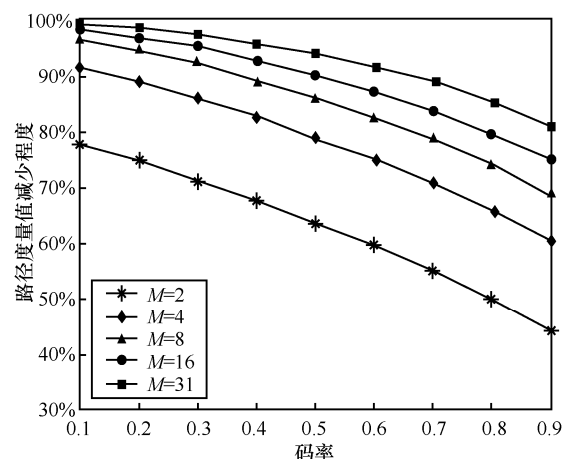


图 5 MSCL 译码算法路径度量值计算减少的程度

分别为 2、4、8、16、32 时, MSCL 译码算法相比于参考文献[12]的算法分别至少可以减少 38%、51%、58%、64%、72% 的路径计算复杂度。其中, 当 $M=8$, 极化码码率取值为 0.1~0.9 时, MSCL 译码算法相比参考文献[12]的算法减少的路径计算复杂度为 58%~94%。

在 AWGN 信道下, 采用码长为 1 024、码率为 0.5 的极化码, M 分别为 2、4、8, 对 MSCL 译码算法与 SCL 译码算法进行相应的性能对比, 其中 $L=4$, 采用的调制方式为 BPSK 调制。图 6 表示的是 MSCL 译码算法与 SCL 译码算法的误帧率性能对比, 由图 6 可以发现, MSCL 译码算法在不同的 M 取值下, 译码性能几乎与 SCL 译码性能一致。

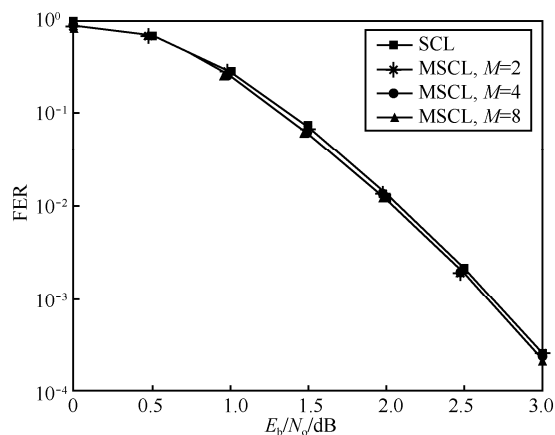


图6 误帧率性能对比

4.2 SCA-MSCL 译码算法的仿真结果与分析

对提出的SCA-MSCL译码算法的性能进行仿真,分别从不同码长和不同分段数进行误帧率讨论,同时与传统的CA-SCL译码性能进行对比。在仿真的同等码长条件下,令极化码的编码码率相同,且保证信息比特与CRC码字组合后,相应的码率都达到50%。表1是仿真中不同码长的码字构造方案。构造的码字都是在AWGN信道、BPSK调制下进行性能比较的。图7给出的是在码长分别为128、256、512和1024的情况下,不同的码字构造方案的误帧率性能。

表1 不同码长的码字构造方案

极化码 码长 N	分段 数 S	CRC 码长	译码列表 长度 L	同时译 码 M	译码算法
128	1	8	8	8	CA-SCL
128	2	4	8	8	SCA-MSCL
256	1	16	8	8	CA-SCL
256	2	8	8	8	SCA-MSCL
256	4	4	8	8	SCA-MSCL
512	1	16	8	8	CA-SCL
512	2	8	8	8	SCA-MSCL
512	4	4	8	8	SCA-MSCL
1024	1	32	8	8	CA-SCL
1024	2	16	8	8	SCA-MSCL
1024	4	8	8	8	SCA-MSCL
1024	8	4	8	8	SCA-MSCL

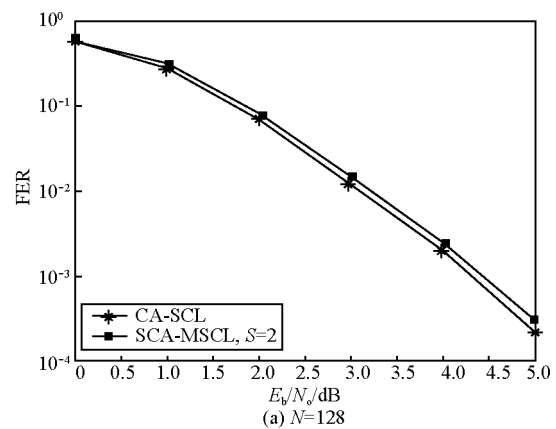
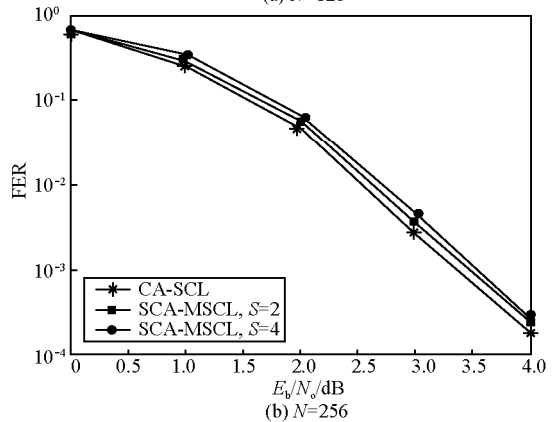
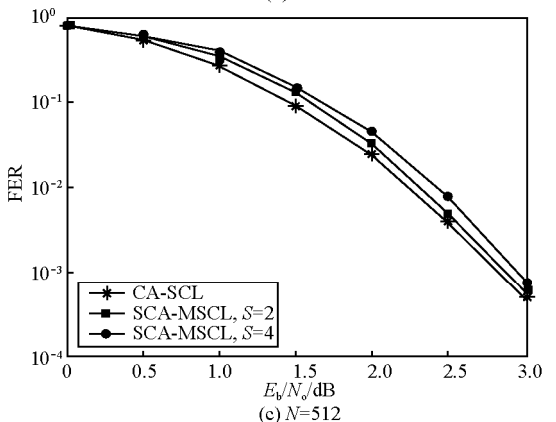
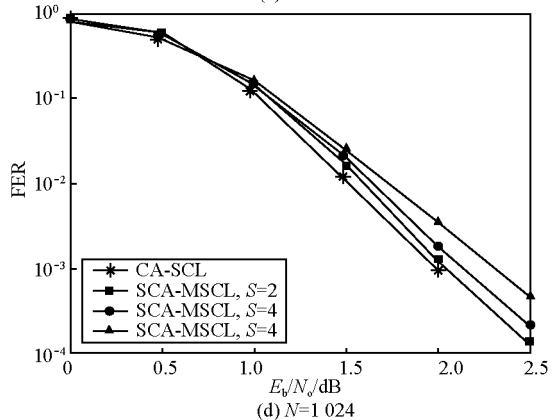
(a) $N=128$ (b) $N=256$ (c) $N=512$ (d) $N=1024$

图7 不同码长下各码字构造方案的误帧率性能对比

表 2 SCA-MSCL 空间存储及时延减小的统计

类型	较 CA-SCL 的时延降低	较 CA-SCL 的空间存储减少
$N=128, S=2$	84.29%	21.87%
$N=256, S=2$	84.11%	21.88%
$N=256, S=4$	85.81%	43.75%
$N=512, S=2$	84.89%	23.24%
$N=512, S=4$	86.46%	43.75%
$N=1\ 024, S=2$	84.85%	23.24%
$N=1\ 024, S=4$	86.42%	43.75%
$N=1\ 024, S=8$	87.00%	50.01%

由图 7 (a)~图 7 (d) 可得, 当分段数 S 不大于 4 时, SCA-MSCL 算法误帧率性能与 CA-SCL 算法的误帧率性能比较接近, 即没有明显的性能损失。但是, 在仿真过程中, 为了保证码率相等, 随着分段数的增加, 相应的 CRC 译码长度也会变小, 因为 CRC 检验错误的能力与其码长有一定的关系, 所以, 在仿真过程中保证了 CRC 的码长不小于 $4^{[16]}$ 。

表 2 表示的是与 CA-SCL 译码算法相比, SCA-MSCL 译码算法在不同的条件下对应的空间存储及时延降低程序。由表 2 可知, 随着分段数 S 的增加, 相应的空间存储将会进一步减少, 其中当分段数 S 为 2 时, SCA-MSCL 译码算法至少可以减少 21% 的空间存储, 而相比于 CA-SCL 译码算法, SCA-MSCL 译码算法至少可以降低 84% 的译码时延。

5 结束语

本文对极化码的多比特译码进行了研究, 提出了 MSCL 译码算法, 通过仿真表明, MSCL 译码算法在不损失译码性能的前提下, 可以将对应的译码时延从 $3 \cdot N - 2$ 个时钟降为 $4 \cdot N / M - 2$ 个时钟, 随着 M 的增加, MSCL 译码可以减少更多的译码时延, 其中, 当 $M=8$ 时, 相对于 SCL 译码算法降低的译码时延为 83%。同时, 相比于

LLR- 2^k b-SCL 译码算法, MSCL 译码算法具有更低的路径度量值计算复杂度, 其中当 $M=8$, 极化码码率为 0.1~0.9 时, 可以减少 58%~94% 的路径计算复杂度。CA-SCL 译码算法可以使有限码长的极化码性能提高, 但是其占用的存储空间较大, 因此, 提出了 SCA-MSCL 译码算法, SCA-MSCL 译码算法可以通过多个 CRC 辅助及时输出每一段最符合条件的码字, 从而可以达到节省存储空间、降低时延以及保证译码性能的效果。当分段数 $S=2$ 、 $M=8$ 时, SCA-MSCL 译码算法相比于 CA-SCL 译码算法至少可以降低 21% 的存储空间, 随着分段数的增加, 相应的存储空间将会进一步降低, 同时, 相比于 CA-SCL 译码算法, 可以降低至少 84% 的译码时延。

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] LEROUX C, RAYMOND A J, SARKIS G, et al. A semi-parallel successive-cancellation decoder for polar codes[J]. IEEE Transactions on Signal Processing, 2013, 61(2): 289-299.
- [3] TAL I, VARDY A. List decoding of polar codes[J]. IEEE Transactions on Information Theory, 2015, 61(5): 2213-2226.
- [4] BALATSOUKASSTIMMING A, RAYMOND A J, GROSS W J, et al. Hardware architecture for list sc decoding of polar codes[J]. Computer Science, 2014.



- [5] RAYMOND A J, GROSS W J. Scalable successive-cancellation hardware decoder for polar codes[J]. *Computer Science*, 2013, 62(20): 1282-1285.
- [6] ZHANG C, PARHI K K. Low-latency sequential and overlapped architectures for successive cancellation polar decoder[J]. *IEEE Transactions on Signal Processing*, 2013, 61(10): 2429-2441.
- [7] SARKIS G, GROSS W J. Increasing the throughput of polar decoders[J]. *IEEE Communications Letters*, 2013, 17(4): 725-728.
- [8] XIONG C, LIN J, YAN Z. Symbol-based successive cancellation list decoder for polar codes[C]//*Signal Processing Systems*, Oct 20-22, 2014, Belfast, UK. Piscataway: IEEE Press, 2014: 675-687.
- [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 Systems*, 2015, 23(10): 2268-2280.
- [10] LIN J, YAN Z. An efficient list decoder architecture for polar codes[J]. *IEEE Transactions on Very Large Scale Integration Systems*, 2015, 23(11): 2508-2518.
- [11] 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.
- [12] BO YUAN, KESHAB K. LLR-based successive-cancellation list decoder for polar codes with multibit decision[J]. *IEEE Transactions on Very Large Scale Integration Systems*, 2017, 23(10): 2268-2280.
- [13] NIU K, CHEN K. CRC-aided decoding of polar codes[J]. *IEEE Communications Letters*, 2012, 16(10): 1668-1671.
- [14] EL-KHAMY M, LEE J, KANG I. Detection analysis of CRC-assisted decoding[J]. *IEEE Communications Letters*, 2015, 19(3): 483-486.
- [15] YUAN B, PARHI K K. Low-latency successive-cancellation polar decoder architectures using 2-bit decoding[J]. *IEEE Transactions on Circuits & Systems I Regular Papers*, 2014, 61(4): 1241-1254.
- [16] LI H, YUAN J. A practical construction method for polar codes in AWGN channels[C]//*Tencon Spring Conference*, April 17-19, 2013, Sydney, Australia. Piscataway: IEEE Press, 2013: 223-226.
- [17] EL-KHAMY M, LEE J, KANG I. Detection analysis of CRC-assisted decoding[J]. *IEEE Communications Letters*, 2015, 19(3): 483-486.

[作者简介]



沈周青（1992-），男，杭州电子科技大学通信工程学院硕士生，主要研究方向为信道编码。



尚俊娜（1979-），女，博士，杭州电子科技大学通信工程学院副教授，主要研究方向为通信信号处理、信道编码。