一种低译码复杂度的 Turbo 架构 LDPC 码

熊　磊　　谈振辉　　姚冬苹
(北京交通大学轨道交通控制与安全国家重点实验室　北京　100044)

摘　要：针对低密度奇偶校验(LDPC)码较大的译码复杂度和 RAM 占用，该文提出了一种低译码复杂度的 Turbo 架构 LDPC 码——并行交织级联 Gallager 码 (Parallel Interleaved Concatenated Gallager Code，PICGC)。该文给出了 PICGC 的设计方法和编译码算法，并分析比较了 PICGC 译码器与 LDPC 译码器所需的 RAM 存储量，推导出 RAM 节省比的上界。理论分析和仿真结果表明，PICGC 以纠错性能略微降低为代价，有效地降低译码复杂度和 RAM 存储量，且译码时延并未增加，是一种有效且易于实现的信道编码方案。

关键词：LDPC 码；级联码；译码复杂度
中图分类号：TN911.22　　**文献标识码**：A　　**文章编号**：1009-5896(2007)12-2907-05

A Low Decoding Complexity Gallager Code with Turbo Architecture

Xiong Lei　Tan Zhen-hui　Yao Dong-ping
(*State Key Laboratory of Rail Traffic Control and Safety*, *Beijing Jiaotong University*, *Beijing* 100044, *China*)

Abstract: Be aimed at lower complexity and RAM requirement of Low Density Parity Check (LDPC) decoder, a new class of concatenated codes called Parallel Interleaved Concatenated Gallager Code (PICGC), based on Turbo architecture and LDPC codes, is presented. In this paper, design, encoding and decoding algorithms of PICGC are studied. The RAM requirement for PICGC decoder is analyzed and compared to LDPC decoder, and an upper bound of memory-saving ratio is derived. The theoretical analysis and simulation results demonstrate that PICGC can reduce decoding complexity and RAM requirement significantly and maintain decoding delay with little sacrifice in performance in comparison to conventional LDPC codes. PICGC is an effective and feasible channel coding scheme.

Key words: LDPC codes; Concatenated codes; Decoding complexity

1　引言

1961 年Gallager首先提出了低密度奇偶校验(Low Density Parity Check，LDPC)码，因此LDPC码也被称为Gallager码[1]。1996 年Mackey和Neal进一步指出LDPC码具有逼近Shannon信道容量限的优秀性能[2, 3]。研究表明，码长为 10^7的LDPC码在采用置信传播(Belief Propagation, BP)迭代译码算法时，距Shannon限仅 0.0045dB[4]。作为一种非常优秀的线性分组码，LDPC码在二进制对称信道、加性高斯白噪声(Additive White Gaussian Noise，AWGN)信道、衰落信道下均具有良好的性能。然而LDPC码的译码复杂度和所需的RAM存储量随码长线性增长[5]，这在一定程度上制约了LDPC码的应用，特别是长码的应用。如何在保持LDPC良好性能的前提下，尽可能地降低译码复杂度和占用的RAM存储空间，成为当前迫切需要解决的问题之一。

当前降低译码复杂度的方法主要可分为两大类：(1)采用近似算法降低复杂度，如最小BP算法，归一化BP算法等[6, 7]；(2)设计性能较好，迭代收敛较快的码字，如采用渐进边增长 (Progressive Edge-Growth，PEG)法构造码字[8]。这两类方法虽然在一定程度上降低了译码复杂度，但进一步改进的余地有限。级联码通过将一个长码字的译码分解为若干个短码字的译码，从而以较低的复杂度实现了较好的纠错性能。因此，Behairy 和Chang提出了并行级联Gallager码(Parallel Concatenated Gallager Codes，PCGC) [9, 10]。

如图 1 所示，PCGC采用两个并行级联的LDPC编码器分别对信息序列s进行编码，p^1和p^2为相应的校验比特序列，最后将两个码字合并，输出码字为$C=\left[s,p^1,p^2\right]$。仿真结果表明，PCGC仅在信噪比(Signal to Noise Ratio，SNR)较低时(SNR<1dB)可以降低译码复杂度，而在SNR较高时，其译码复杂度反而要明显高于LDPC码。虽然PCGC的实用价值有限，但为降低译码复杂度开辟了一条新的思路。本文借鉴了PCGC的基本思想，将Turbo架构与LDPC码相结合，提出了一种新的级联码——并行交织级联Gallager码(Parallel Interleaved Concatenated Gallager Codes，PICGC)。

2006-06-13 收到，2006-10-25 改回
国家自然科学基金重点项目(6033202)资助课题

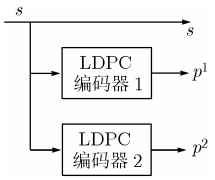

图 1 PCGC 编码器结构图

2 并行交织级联 Gallager 码

在PICGC中，通常采用码长和码率相同，而平均列重(Mean Column Weigh，MCW)不同的两个LDPC码作为成员码。假定两个成员码的码长为N，信息位长度为k，校验位为$M=N-k$，校验矩阵分别为$\boldsymbol{H}_1$和$\boldsymbol{H}_2$。MCW定义为

$$\mathrm{MCW}=\frac{1}{N}\sum_{n=1}^{N}d_v(n) \tag{1}$$

其中$d_v(n)$ 为校验矩阵中第n列的重量。

Behairy和Chang发现，MCW较低的LDPC码在SNR较低时具有较好的纠错性能，而MCW较高的LDPC码则具有较低的“误码平台”(error floor)[11]。在LDPC码的设计中，通常需要对两者进行权衡。而在PICGC的设计中，可以选择MCW较低的LDPC码作为其第一个成员码，而第二个成员码则选择MCW较高的LDPC码。因此，与LDPC码相比，PICGC的设计更为灵活。

如图 2 所示，PICGC 在 PCGC 编码器的基础上增加了数据分割器和交织器。编码时，数据分割器首先将信息序列 S 分隔为 L 段 $s_1s_2\cdots s_L$，每段包含 k 个比特。成员编码器 1 分别对 $s_1s_2\cdots s_L$ 进行编码，输出 L 个长度为 N 的子码字 $\left\{s_l p_l^1\right\}$；而交织后的信息序列 S^* 也同样被分割为 L 段 $s_1^*s_2^*\cdots s_L^*$，并在成员编码器 2 中进行编码，输出的子码字为 $\left\{s_l^* p_l^2\right\}$。最后，删除合并器将成员编码器 1 和编码器 2 输出的合计 $2L$ 子码字中冗余的信息比特删除后加以合并。PICGC 的输出码字为 $C=\left(S,P^1,P^2\right)$，其中 $S=\left(s_1s_2\cdots s_L\right)$，$P^1=\left(p_1^1p_2^1\cdots p_L^1\right)$ 和 $P^2=\left(p_1^2p_2^2\cdots p_L^2\right)$。显然，码字的长度为 $N_{\mathrm{PICGC}}=(2N-k)L$，码率为 $R=\dfrac{k\bullet L}{(2N-k)L}=\dfrac{k}{2N-k}$。此外，通过在删除合并器中删除部分校验比特也可以方便地实现可变码率 PICGC 码。

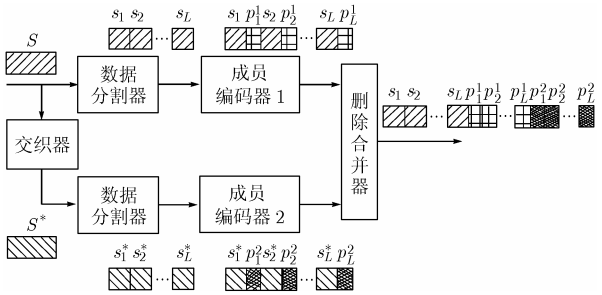

图 2 PICGC 编码器结构图

3 PICGC 译码算法

如图 3 所示 PICGC 译码器采用了两个级联的软输入-软输出成员译码器(DEC1 和 DEC2)。PICGC 译码算法的基本思路与 Turbo 码类似，整个译码过程主要可以分为两个部分：即 DEC1 和 DEC2 中的迭代译码与 DEC1 和 DEC2 之间的信息交换。但与 Turbo 码稍不同的是，PICGC 对 $2L$ 个子码字分别进行译码，即采用分段译码方式。

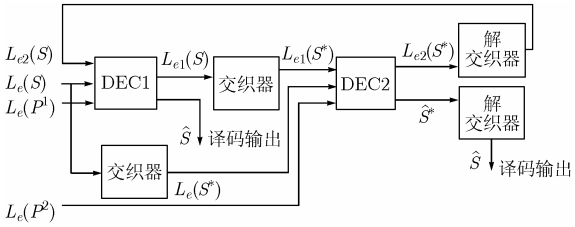

图 3 PICGC 译码器结构图

DEC1 和 DEC2 中的迭代译码通常采用 BP 算法(或其改进算法，如最小 BP 算法，归一化 BP 算法等)，也称为“自迭代”译码。DEC1 和 DEC2 在进行一定次数的自迭代译码后，将所得到的信息比特的外信息进行交换，作为对方接下来译码的先验信息，这一信息交换过程则被称为“互迭代”。

DEC1 和 DEC2 中的自迭代译码可以依次进行(串行模式)，也可行同时进行(并行模式)。串行模式的纠错性能更好，所需的硬件规模也较小，但其译码时延要大于并行模式。本文仅讨论串行模式。

DEC1 的自迭代译码如下所示：

假设子码字 $s_lp_l^1$ 采用BPSK调制，经AWGN信道传输，在接收端得到的序列为 $x=\left(x_1x_2\cdots x_N\right)$。令在第$i$次自迭代中，校验节点$m$传递给比特节点$n$的信息为$R^{(i)}(m,n)$，而比特节点$n$传递给校验节点$m$的信息为$Q^{(i)}(m,n)$。$N(m)$ 表示与校验节点m相邻的比特节点，$M(n)$ 则表示与比特节点n相邻的校验节点。$N(m)\setminus n$ 表示从 $N(m)$ 中删除n，而 $M(n)\setminus m$ 表示从 $M(n)$ 中删除m。

步骤 1 初始化

对于每一个 m 和 n，

$$Q^{(0)}(m,n)=\begin{cases}L_c(n)+L_{e2}^{(k-1)}(n), & 1\le n\le k\\ L_c(n)+L_{e1}^{(k-1)}(n), & k+1\le n\le N\end{cases} \tag{2}$$

其中 $L_c(n)=\dfrac{2}{\sigma^2}x_n$ 为比特节点 n 的信道对数似然比(Log-Likelihood Ratio, LLR)，σ^2 为信道噪声方差，$L_{e1}^{(k-1)}(n)$ 和 $L_{e2}^{(k-1)}(n)$ 分别为 DEC1 和 DEC2 在第 k–1 次互迭代所得比特 n 的外信息，而 $L_{e1}^{(0)}(n)=L_{e2}^{(0)}(n)=0$。

步骤 2 校验节点更新

对每个校验节点 m 和 $n\in N(m)$，

$$R^{(i)}(m,n)=-\prod_{n'\in N(m)\backslash n}\operatorname{sgn}\left(Q^{(i-1)}(m,n')\right)\cdot\Phi\left(\sum_{n'\in N(m)\backslash n}\Phi\left(Q^{(i-1)}(m,n')\right)\right)\tag{3}$$

其中 $\Phi(x)=\ln\left(\left|\tanh(\frac{x}{2})\right|\right)=\ln\left|\frac{e^x-1}{e^x+1}\right|$。

步骤 3　比特节点更新

对每个比特节点 n 和 $m\in M(n)$，

$$Q^{(i)}(m,n)=L_c(n)+\sum_{m'\in M(n)\backslash m}R^{(i)}(m',n)\tag{4}$$

步骤 4　尝试译码

各比特节点 n 的外信息 LLR 为

$$L_{e1}^{(k)}(n)=\sum_{m'\in M(n)}R^{(i)}(m',n)\tag{5}$$

各比特节点 n 的后验 LLR 为

$$\mathrm{LT}(n)=L_c(n)+L_{e1}^{(k)}(n)\tag{6}$$

根据后验 LLR 进行硬判决，生成码字矢量 $\hat{\boldsymbol{x}}=[\hat{x}_n]$

$$\hat{x}_n=\begin{cases}0, & \mathrm{LT}(n)>0\\ 1, & \mathrm{LT}(n)\le 0\end{cases}$$

步骤 5　校验

如果 $\boldsymbol{H}_1\hat{x}=0$，即已收敛到合法码字，则终止译码，译码结果为 $\hat{\boldsymbol{s}}_l=(\hat{x}_1\hat{x}_2\cdots\hat{x}_k)$。

DEC1 输出信息比特的外信息为

$$L_{e1}^{(k)}(s_l)=\left\{L_{e1}^{(k)}(n)\right\},\quad 1\le n\le k$$

如果 $\boldsymbol{H}_1\hat{\boldsymbol{x}}\ne 0$，则返回步骤 2，继续进行迭代

如果已达到允许的最大自迭代次数，则输出外信息

$$L_{e1}^{(k)}(s_l)=\left\{L_{e1}^{(k)}(n)\right\},\quad 1\le n\le k$$

DEC2 的自迭代译码仅在初始化阶段与 DEC1 略有不同，即

$$Q^{(0)}(m,n)=\begin{cases}L_c(n)+L_{e1}^{(k)}(n), & 1\le n\le k\\ L_c(n)+L_{e2}^{(k-1)}(n), & k+1\le n\le N\end{cases}$$

$$L_{e2}^{(0)}(x_n)=0,\quad k+1\le n\le N$$

当成员码 1 的 L 个子码字译码全部结束后，对 DEC1 输出的外信息 $L_{e1}(S)=\left(L_{e1}^{(k)}(s_1)L_{e1}^{(k)}(s_2)\cdots L_{e1}^{(k)}(s_L)\right)$ 进行交织，交织后的外信息 $L_{e1}\left(S^*\right)$ 将作为 DEC2 译码的先验信息。同理，当成员码 2 的 L 个子码字译码结束后，DEC2 输出的外信息 $L_{e2}\left(S^*\right)$ 解交织后将传递给 DEC1。至此完成了一次互迭代译码。

如果成员码1(或成员码2)的全部 L 个子码字均收敛到合法码字，则输出 $\hat{S}=\{\hat{s}_l\}$ 作为译码结果，整个译码过程宣告结束。

4　PICGC 中的交织器

PICGC 与 PCGC 最大的不同就是引入了交织器和采用分段编译码。下面简要介绍交织器在 PICGC 中所起的作用。

假设 s_m 和 s_n 为成员码 1 的某个子码字中的两个信息比特，且参与了同一个校验式。由于交织器对信息序列进行了重新排列，s_m 和 s_n 在成员编码器 2 中将可能被编入不同的子码字之中。在DEC2 的译码中，s_m 和 s_n 将与其所在子码字中的比特进行信息交换，而在DEC1 的译码中，s_m 和 s_n 之间将进行信息交换，因此，此时它们之间交换的信息必然包含了它们之前在DEC2 中所获得的信息，从而实现了成员码 2 中两个子码字之间的信息交换。换句话说，s_{m} 和 s_{n} 为两个子码字之间的信息交换搭建了一座桥梁，从而使得看似分离的各个子码字成为了一个整体，而这正是由交织器发挥的作用。

而 PCGC 是由两个 LDPC 编码器直接对信息序列进行编码，并没有进行分段，可以认为 PCGC 是 PICGC 当 L=1 时的一个特例。PCGC 只包含两个成员码码字，而没有子码字，因此也就不需要交织器为子码字搭建信息交换的桥梁。仿真结果也证实了，在 PCGC 两个成员编码器之间加入交织器并不能改善 PCGC 性能。PICGC 将 LDPC 码分解为 $2L$ 个子码字，而 PCGC 仅将其分解为 2 个成员码码字，这是 PICGC 总体性能好于 PCGC 的关键。

5　RAM 存储量分析

5.1　LDPC 译码器

LDPC译码器需要一定的RAM用于存储 L_c，R和 Q等变量的值。通常 L_c，R和 Q等变量的位长相同，且取决于采用的量化方案,因此为了研究的方便,本节用占用RAM单元数,而不是比特数，作为计量存储量的单位。

显然，L_c占用的存储量为 N_{LDPC}，而 R和 Q中的每个元素分别对应校验矩阵 $\boldsymbol{H}$中的一个非零元素，而 $\boldsymbol{H}$中的非零元素总数等于

$$W=\sum_{n=1}^{N_{\mathrm{LDPC}}}d_v(n)=N_{\mathrm{LDPC}}\cdot\mathrm{MCW}_{\mathrm{LDPC}}$$

因此，译码器所需 RAM 存储量为

$$M_{\mathrm{LDPC}}=N_{\mathrm{LDPC}}\left(1+2\mathrm{MCW}_{\mathrm{LDPC}}\right)\tag{7}$$

其它变量所占用的 RAM 可以忽略不计。

5.2　PICGC 译码器

PICGC的译码包括两部分，即自迭代和互迭代。在自迭代中，同样需要存储 L_c，R和 Q的值。其中，L_c占用的存储量为 $N_{\mathrm{PICGC}}=(2N-k)L$。在DEC1 中，变量 R和 Q合计占用 $2N\cdot\mathrm{MCW}_1$ 个单位的RAM，而在DEC2 中，则占用 $2N\cdot\mathrm{MCW}_2$ 个单位，其中 MCW_1和 MCW_2分别为 $\boldsymbol{H}_1$和 $\boldsymbol{H}_2$的平均列重。由于采用串行模式译码，DEC1 和DEC2 轮流进行自迭代译码，因此 R和 Q所占用的存储区可以进行复用。因为 $\mathrm{MCW}_1<\mathrm{MCW}_2$，因此仅需 $2N\bullet\mathrm{MCW}_2$ 个单位的RAM即可。

所以，自迭代需要占用的 RAM 空间为

$$M_1=(2N-k)L+2N\cdot\mathrm{MCW}_2$$

在互迭代中，译码器需要存储 $L_{e1}(S)$，$L_{e1}(P^1)$，$L_{e2}(S^*)$

和$L_{e1}(P^2)$等变量的值，它们占用的RAM存储量如表 1 所示。$L_{e1}(S^*)$和 $L_{e2}(S)$的值由于通过地址变换可以分别从$L_{e1}(S)$和$L_{e2}(S^*)$中获得，因此不必另行存储。

表 1 互迭代中各变量占用的 RAM 存储量

	$L_{e1}(S)$	$L_{e2}(S^*)$	$L_{e1}(P^1)$	$L_{e1}(P^2)$
存储量	$k\cdot L$	$k\cdot L$	$(N-k)L$	$(N-k)L$

因此，互迭代所需 RAM 存储量为

$$M_2 = 2N\cdot L$$

PICGC 译码器总的 RAM 需求为

$$M_{\text{PICGC}} = M_1 + M_2 = 4N\cdot L - k\cdot L + 2N\cdot \text{MCW}_2 \quad (8)$$

与码长相同的 LDPC 译码器相比，PICGC 译码器的 RAM 节省比为

$$\mu = 1-\frac{M_{\text{PICGC}}}{M_{\text{LDPC}}} = 1-\frac{4N\cdot L - k\cdot L + 2N\cdot \text{MCW}_2}{(2N-k)\big(1+2\text{MCW}_{\text{LDPC}}\big)L}$$
$$= 1-\frac{4N-k+2N\cdot \text{MCW}_2/L}{(2N-k)\big(1+2\text{MCW}_{\text{LDPC}}\big)} \quad (9)$$

显然， μ 随 L 的增加而增加，其上界为

$$\lim_{L\to\infty}\mu = 1-\frac{4N-k}{(2N-k)(1+2\text{MCW}_{\text{LDPC}})} \quad (10)$$

但如果 L 取值过大，将会导致 PICGC 纠错性能一定程度的下降。因此，通常 L 取 10 ~ 20。

6 译码吞吐量分析

在 BP 算法中，各校验节点和比特节点的更新是相互独立的，可以同步进行，因此 BP 算法实质上是一种并行算法。在图 4 所示的全并行结构中，LDPC 码的 M 个校验节点 $c_1c_2\cdots c_M$ 和 N个比特节点 $b_1b_2\cdots b_N$ 都拥有独立的更新处理单元，即校验节点更新单元(Check Function Unit，CFU)和比特节点更新单元(Bit Function Unit，BFU)。全并行结构 LDPC 译码器的译码吞吐量非常高，可达几百 Mbps，甚至更高。但对于中长 LDPC 码，其硬件复杂度是无法承受的。因此，一般在实际应用中采用部分并行结构译码器。如图 5 所示，部分并行结构将校验节点和比特节点分为 L 组，各组节点依次进行更新，因此整个译码器只需要 M/L 个 CFU 和 N/L 个 BFU 即可。部分并行结构通过对 CFU 和 BFU 进行复用，有效地降低了硬件规模，然而其译码吞吐量也只有全并行结构的 $1/L$，在译码吞吐量与硬件复杂度之间取得折中。

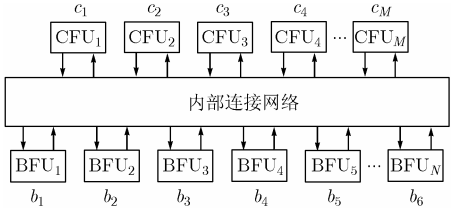

图 4 全并行结构 LDPC 译码器结构图

在 PICGC 译码器设计中，由于子码字相对较短，DEC1 和 DEC2 可以采用全并行结构。PICGC 将码字分解为若干个子码字，依次在 DEC1 和 DEC2 中进行译码；而部分并行结构译码器将节点分成若干个节点组，依次在 CFU 和 BFU 中进行更新。因此，在相同硬件规模的条件下， PICGC 译码器与部分并行结构 LDPC 译码器吞吐量相当，可达十几～几十 M bps。因此，PICGC 可以应用于高速译码。

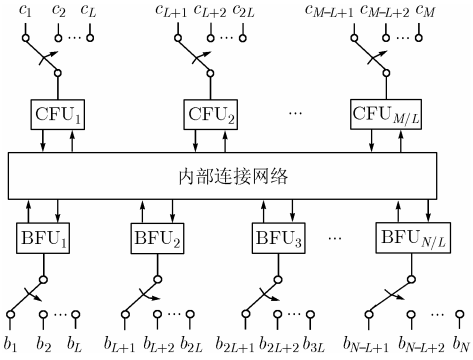

图 5 部分并行结构 LDPC 译码器结构图

7 仿真结果

本节的仿真基于 AWGN 信道，发射端采用 BPSK 调制。

PICGC 码长 $N_{\text{PICGC}}=11560$，码率 $R=0.7$，L =20。两个成员码分别为(N=490, $R=0.82$， MCW_1=2.21)和(N=490, $R=0.82$， MCW_1=2.87)。PICGC 译码器的最大自迭代次数为 10，最大互迭代次数为 6。仿真采用的 LDPC 码为($N_{\text{LDPC}}=11560$， $R=0.7$， $\text{MCW}_{\text{LDPC}}=2.87$)。LDPC 码译码器的最大迭代次数为 120。PICGC 的成员码与 LDPC 码均为 PEG 法构造的非正则码。仿真时，在每个仿真点至少收集 50 个错误译码码字。

图 6 所示为PICGC和LDPC码在AWGN信道下的误比特(Bit Error Rate，BER)曲线。可以看出，PICGC的纠错性能略差于LDPC码，但随着SNR的增加，差距不断缩小，在BER为 10^{-4}时，仅相差约 0.15 dB。性能略有下降的主要原因是信息在DEC1 和DEC2 之间交换时存在一定的损失。

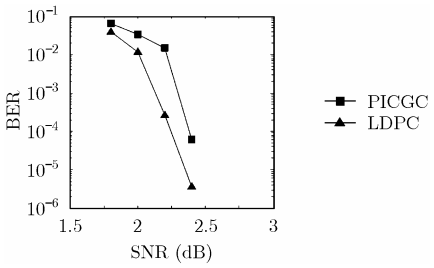

图 6 AWGN 信道下 PICGC 与 LDPC 码误比特率比较

译码迭代中，校验节点更新是最为复杂的部分，因此可用校验节点更新的次数作为衡量复杂度的尺度。LDPC 码每

进行一次迭代译码，需要进行 $N_{\mathrm{LDPC}}\cdot \mathrm{MCW}_{\mathrm{LDPC}}=11560\times 2.87=33177$ 次校验节点更新；对于 PICGC，DEC1 完成一次自迭代需要进行 $N\bullet\mathrm{MCW}_1=490\times 2.21=1083$，而 DEC2 需要进行 $N\bullet\mathrm{MCW}_2=490\times 2.87=1406$ 次校验节点更新。

由表 2 可见，当 SNR 为 1.8dB，2.0dB 和 2.2dB 时，PICGC 的译码复杂度均低于 LDPC 码。而当 SNR=2.4dB 时，PICGC 的复杂度则略高于 LDPC 码。与 LDPC 码相比，PICGC 译码复杂度随 SNR 变化较小，这在无线通信中具有特别重要的意义。

表 2　AWGN 信道下 PICGC 与 LDPC 码译码复杂度比较

SNR (dB)	平均迭代次数			校验节点更新次数	
	LDPC 码	PICGC		LDPC 码	PICGC
		DEC1	DEC2		
1.8	115.7	678	665	3.84E6	1.67E6
2	74.2	535	513	2.46E6	1.30E6
2.2	36.3	446	417	1.20E6	1.07E6
2.4	15.9	234	211	5.28E5	5.50E5

由式(8)和式(9)可得 PICGC 和 LDPC 译码器所需 RAM 的存储量。表 3 列出了采用 5 比特量化时，PICGC 和 LDPC 译码器所需的 RAM。由表 3 可知，PICGC 译码器可以节省约 56.4%的 RAM，这有效地降低了译码器的成本和复杂度。

表 3　PICGC 与 LDPC 码译码器所需 RAM 存储量

	PICGC	LDPC 码	RAM 节省比 μ
所需 RAM 存储量 (bit)	169,860	389,570	56.4%

8　结束语

本文将 Turbo 架构与 LDPC 码相结合，提出了一种新的级联码——并行交织级联 Gallager 码(PICGC)。PICGC 将长码的译码分解为 $2L$ 个子码字的译码，并通过交织器实现子码字之间的信息交换，从而基本保持了 LDPC 码优异的纠错性能。仿真结果表明，PICGC 在中低 SNR 时，可以有效地降低译码复杂度。此外 PICGC 还可以减少译码器 RAM 存储量 50%以上。PICGC 性能优异，易于硬件实现，是一种很有应用价值的信道编码方案。

参 考 文 献

[1] Gallager R G. Low-density parity-check codes. *IRE Trans. on Inform. Theory*, 1962, IT-8(1): 21–28.

[2] MacKay D J C and Neal Near R M. Shannon limit performance of low density parity check codes. *IEEE Electron. Lett.*, 1996, 32(18): 1645–1646.

[3] MacKay D J C. Good error-correcting codes based on very sparse matrices. *IEEE Trans. on Information Theory*, 1999, 45(2): 399 –431.

[4] Chung S Y, Forney G D J, and Richardson T J, *et al.*. On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit. *IEEE Commun. Lett.*, 2001, 5(2): 58–60.

[5] MacKay D J C. Gallager codes that are better than Turbo codes. 36th Allerton Conf. Communications, Control, and Computing, Monticello, USA, Sept. 1998: 23–25.

[6] Chen J and Fossorier M P C. Density evolution for two improved BP-based decoding algorithms of LDPC codes. *IEEE Commun. Lett.*, 2002, 6(5): 208–210.

[7] Rusmevichientong P and Van Roy B. An analysis of belief propagation on the Turbo decoding graph with Gaussian densities. *IEEE Trans. on Information Theory*, 2001, 47(2): 745–765.

[8] Hu X-Y, Eleftheriou E, and Arnol D M. Regular and irregular progressive edge-growth Tanner graphs. *IEEE Trans. on Information Theory*, 2005, 51(1): 386–398.

[9] Behairy H and Chang S C. Parallel concatenated Gallager codes. *IEEE Electron. Lett.*, 2000, 36(24): 2025–2026.

[10] Behairy H and Chang S C. Parallel concatenated Gallager codes for CDMA applications. IEEE GLOBECOM 2001, San Antonio, USA, Nov. 2001: 1002–1006.

[11] Behairy H and Chang S C. Analysis and design of parallel concatenated Gallager codes. *IEEE Electron. Lett.*, 2002, 38(18): 1039–1040.

熊　磊：男，1978 年生，博士生，研究方向为移动通信、智能交通、纠错编码技术等.

谈振辉：男，1944 年生，教授，博士生导师，主要研究方向为宽带移动通信、智能交通、个人通信、CDMA 技术、正交频分复用等.

姚冬苹：女，1962 年生，副教授，硕士生导师，主要研究方向为移动通信、纠错编码技术等.