高技术通讯 2012 年 第22 卷 第10 期:1014 ~ 1020

doi:10.3772/j.issn.1002-0470.2012.10.003

可消除分层译码器访问冲突的 LDPC 码构造①

董明科② 张建军 吴建军 项海格③

(北京大学信息科学技术学院 区域光纤通信网与新型光通信系统国家重点实验室 北京 100871)

摘 要 针对普通低密度奇偶校验(LDPC)码校验矩阵导致的行分层译码器访问冲突问题,基于渐进边增长(PEG)算法提出了一种新的 LDPC 码构造算法,即行方向冲突规避块(RCAB)PEG 构码算法——RCAB-PEG 算法,该算法通过逐行建立基础校验矩阵来构造准循环 LDPC(QC-LDPC)码,并支持行间非零元素规避。新构造的码不会导致访问冲突延迟,故能使分层译码器免除有关复杂设计,节省硬件资源,提高译码速率。采用该算法所构码的典型分层译码器速率能提高到原来的 1.33 倍。仿真表明,该算法构造的码,误码性能与 Block-PEG 码及 WiMAX 和 DVB-SII 等标准码相当。

关键词 低密度奇偶校验(LDPC),准循环 LDPC,LDPC 分层译码,渐进边增长(PEG),LDPC 译码器,访问冲突

0 引 言

最初的低密度奇偶校验(low-density parity-check,LDPC)译码算法是由 Gallager[1]提出的,其置信度传播方式为泛洪调度[2,3](flooding schedule)。Mansour 等人提出了一种 Turbo 译码的置信度传播方式[4,5],Hocevar 等人[6-8]称这种基于串行操作更新置信度的传播方式为分层译码算法(layered decoding)。分层译码算法能够在不损失性能和降低运算复杂度的同时,使译码所需的迭代次数减少一半[7,8],因此受到广泛关注。块行分层译码算法采用块行内所有行并行运算的方式[3],适用于准循环 LDPC(quasi-cyclic LDPC,QC-LDPC)码,近年来很流行。文献[9-14]分别针对这种置信度传播方式提出了不同的实现形式。在硬件实现时一般采用修正的最小和(min-sum)算法与分层算法结合,以降低复杂度[9-12]。在译码器结构中一般要采用流水线技术,以提高分层译码器的吞吐率,但同时引入了访问冲突问题[11-15]。文献[11]在流水线中添加空闲节拍来解决访问冲突问题,却降低了译码器吞吐率。文献[12]添加信道信息镜像 RAM,由计算外信息改变量近似更新信道信息,提高流水线运算效率,但性能有所损失。文献[13]调度各块行的运算顺序和校验节点处理器中信道信息输入和输出顺序,使相邻运算行的运算顺序匹配,只能尽量减少空闲等待。文献[14]利用访问冲突的特点,利用尽可能多的耦合以减少读取信道 RAM 次数,降低功耗,但也受限于校验矩阵结构。文献[15]将 QC-LDPC 基矩阵分割为 S × S 的小基矩阵,然后重新排列,将流水线间资源冲突宽松化,但是降低了译码器并行度,使吞吐率降低。

以上文献[11-15]都从译码器设计角度不同程度地解决了访问冲突问题,但是都受限于 LDPC 码校验矩阵结构,设计过程复杂却并不能确保完全消除冲突问题,同时需付出资源、速率或性能代价。行分层译码实现中存在访问冲突问题的根本原因是码设计中没有考虑这一问题。为此,本文构造了能避免行分层译码器结构中的变量访问冲突的 LDPC 码,以使分层译码器的实现简化、速率提高,同时避免了为访问冲突付出硬件资源。

1 行分层译码器的数据访问冲突问题分析

依据行分层译码算法[9-12]和传统译码器设计[6]

① 国家自然科学基金(61071083,60972008)和国防基金(9140A220310)资助项目。
② 男,1973 年生,博士生;研究方向:通信中的信号处理;E-mail:mingke.dong@pku.edu.cn
③ 联系人,E-mail:xianghg@pku.edu.cn
(收稿日期:2011-10-25)

进行优化设计后得到的 QC-LDPC 码译码器如图 1 所示,校验矩阵中的单位循环子矩阵大小为 $p \times p$。可以参照文献[10]中的分层译码算法式

$$u_{ij} = U_j - v_{ij} \tag{1}$$

$$v'_{ij} = \prod_{k \in N(i)/j} \mathrm{sign}(u_{ik}) \times \min_{k \in N(i)/j} | u_{ik} | \times \alpha \tag{2}$$

$$U'_j = u_{ij} + v'_{ij} \tag{3}$$

来理解图 1 译码器原理。式(1)中的 U_j 是信道信息,v_{ij} 是校验节点 i 传递给变量节点 j 的外信息。式(2)中 $N(i)$ 为校验节点 i 相邻的变量节点集合,$N(i)/j$ 是 $N(i)$ 集合排除节点 j 所得集合,sign(.)表示求符号运算,min(.)表示对集合求最小运算,$0 < \alpha < 1$。

图 1 中的信道信息 RAM 存储 U_j 变量,外信息 RAM 存储 v_{ij}变量,循环偏移模块根据 QC-LDPC 码基矩阵偏移量对 p 个 U_j 更新值进行所需的循环移位。校验节点处理器(check node processor, CNP)模块为行分层运算器,对应式(1)-(3),其详细实现结构如图 2 所示。CNP 组表示并行运算的 p 个 CNP,对应于 QC-LDPC 码的 p 行组成的块行。选择模块用来从信道信息 RAM 读取的 U_j 值和刚更新的信道信息 U'_j 之间进行选择[14]。选择模块可为后续行 CNP 运算立即要用到的 U_j 变量提供不需经过信道信息 RAM 的小延迟通道。

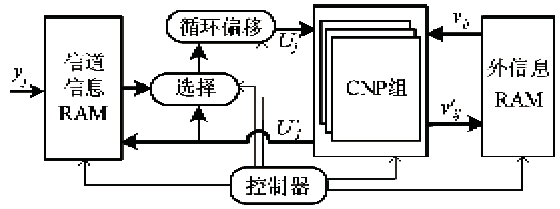

图 1 典型分层译码器结构框图

在图 2 所示的典型 CNP 设计中,可根据模拟退火算法[14]或基因算法[15],排列该层中 u_{ij}值输入与输出调度 RAM(SchRAM)模块的顺序,实现信道信息 U_j 的及时更新。图 2 中行运算模块表示用 u_{ij} 值、符号积(SgnAll)、最小与次最小(Min&SubMin)等值完成(2)式运算。延迟模块是对用 u_{iq} 等求 v'_{iq} 运算的逻辑延迟进行匹配。由于图 2 实现中存在流水线延迟,当输入为 v_{ij} 和 U_j,同时 CNP 的输出不对应公式(2)、(3)中所示的 v'_{ij}, U'_j,故标记为 v'_{iq}, U'_q。从总体上看,CNP 实现式(1)-(3)的计算,所以图 1 中的接口宏观上仍标注符号 v'_{ij}, U'_j,表示 CNP 对 v_{ij}, U_j 的更新运算。

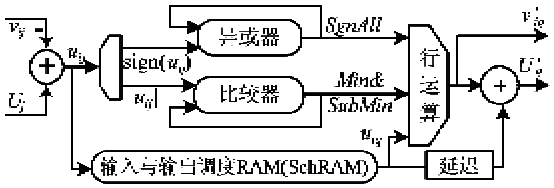

图 2 CNP 模块实现结构图

在图 2 中,当某行处理所需要的所有 U_j 数值都输入 CNP 后,此行运算更新后所得的第一个 U_j 值重新到达 CNP 入口的最短时间记为 DLY·T(T 为电路时钟周期)。典型情况下,从 SchRAM 输出到行运算模块,再经其后的加法器逻辑的流水线,用时为 3T;输出以后要经过循环偏移模块和选择模块组成的流水线,用时为 2T,所以 U_j 在更新完重新到达 CNP 入口至少需要延迟值 DLY·T=5T。其他一些低时钟速率的设计中,DLY 值可达到 2。在时钟速率高,或者 p 值大并行度高的设计中,DLY 值可达 10。DLY 是行分层译码器设计中的重要因素。

行分层译码算法流水线是逐行(对 QC-LDPC 码则是逐块行并行)进行运算,读入前一行用的所有 U_j 变量后,紧接着要读入下一行要用的 U_j 变量,而下一行运算经常要用到上一行刚更新的 U_j 变量。当延迟 DLY 导致不能及时读入上一行刚更新的 U_j 变量时就会发生访问冲突,这可通过表 1 说明。假设 DLY=5。表 1 的基础矩阵 ***Hb*** 第 2 行对应的式(1)-(3)行运算中,CNP 可先连续地从信道信息 RAM 读入非最近更新的 U_j 值 U_4, U_{26}, U_{28},这需经过 3T,然后就需要输入第 1 行运算更新过的 U_2, U_6, U_{25},可它们至少要再经过DLY·T-3T=2T后才能

表 1 译码器流水线第 1、2 行运算都更新的 U_2 值访问冲突

列号	1	2	3	4	5	6	7	…	25	26	27	28
Hb 第 1 行元素	1	1	1	0	0	1	0	…	1	0	1	0
Hb 第 2 行元素	0	1	0	1	0	1	0	…	1	1	0	1
第 1 行用 U_j 值	U_1	U_2	U_3			U_6		…	U_{25}		U_{27}	
第 2 行用 U_j 值		U_2		U_4		U_6		…	U_{25}	U_{26}		U_{28}

到达 CNP 入口，这样就造成了对 U_2 等变量的访问冲突，所以此时流水线就被迫插入 2T 空闲等待节拍，造成多余延迟。不同结构的 LDPC 码造成的延迟不同，最多的时候每行要造成 DLY 拍的浪费，甚至超过行维度，这些浪费很可观，复杂的 U_j 调度设计也很难消除这些延迟，因而需要用新的码设计方法确保码结构能避免 U_j 访问冲突。

可将 LDPC 码设计为如表 2 中的情形，基础矩阵 ***Hb*** 第 6 行中有大于或等于 DLY 个 U_j 值与第 5 行避开，即有 U_1，U_4，U_5，U_{25}，U_{27} 共 5（= DLY）个 U_j 值和第 5 行规避，这样就可以在对第 6 行运算时先让 CNP 输入 5 个 U_j 值 U_1，U_4，U_5，U_{25}，U_{27}，然后输入第 5 行运算刚更新过的 U_2 值，从而流水线在第 5 行到第 6 行处理转换期间不需要多余空闲等待。定义后面行中非零元素中与前一行非零元素处于不同列的元素个数为规避度（avoidence degree，*ADeg*）。比如表 1 中第 2 行对第 1 行的规避度为 3，而表 2 中 5、6 行间规避度为 5。只要让 *ADeg* ≥ DLY 就可避免对 U_j 变量的访问冲突，可要求 *ADeg* = DLY，本文提出的新构造方法将采用此约束条件。从 DLY 经验值看，*ADeg* 的范围为 2 到 10，典型值为 5 到 8。

表 2 译码流水线第 5、6 行运算都更新的 U_2 值访问无冲突

列号	1	2	3	4	5	6	7	…	25	26	27	28
Hb 第 5 行元素	0	1	1	0	0	1	1	…	0	1	0	1
Hb 第 6 行元素	1	1	0	1	1	0	0	…	1	0	1	0
第 5 行用 U_j 值		U_2	U_3			U_6	U_7	…		U_{26}		U_{28}
第 6 行用 U_j 值	U_1	U_2		U_4	U_5			…	U_{25}		U_{27}	

2 按行方向填充的 RPEG 算法

传统的渐进边增长[16]（progressive edge-growth，PEG）类算法是一种基于局部环长最大化的 LDPC 码构造方法，它是逐列建立连接的列方向 PEG（column-direction PEG，CPEG）构造算法，构造过程可兼顾列维度分布和局部环长最大化，所构码性能优良。*ADeg* 约束条件是基于基矩阵行之间的约束，但传统 PEG 及其推广都是按照列方向进行校验矩阵构造，不便于构造具有行间约束关系的 LDPC 码，因此改为在 PEG 算法中逐行添加边的方式，能够更加方便地构造满足要求的 LDPC 码。故本文提出行方向 PEG（row-direction PEG，RPEG）构造算法。可以比照 CPEG 算法[16]理解 RPEG 算法。RPEG 算法流程如表 3 所示：

图 3 为 CPEG 算法和 RPEG 算法的展开图的对比，其中实心圆表示变量节点，空心方框表示校验节点。本文采用的 RPEG 算法与 CPEG[16]对偶，按校验节点一行一行填充连接，算法处理过程中图展开处理是以校验节点（行）为根节点进行的。

表 3 RPEG 算法流程

For $i = 0: m-1$（m 为校验矩阵的行数）
  For $k = 0: d_c^i - 1$（d_c^i 为校验节点 i 的维度）
    IF $k = = 0$
    则选择度数增长最慢的变量节点与当前校验节点 i 建立连接，在 Tanner 图中添加相应的边。
    ELSE
    在当前 Tanner 图基础上，以校验节点 i 为根节点展开成深度为 l 的子图（参图 3），直到第 l 层的当前子图变量节点总数目达到校验矩阵总列数 n，或者第 $l+1$ 层的变量节点集合为空。然后优先选用没有连接到子图中的变量节点，否则在 l 层的变量节点集合中选择具有最低度数的变量节点与根节点建立连接，并在 Tanner 图中添加相应的边。
    End
  End
End

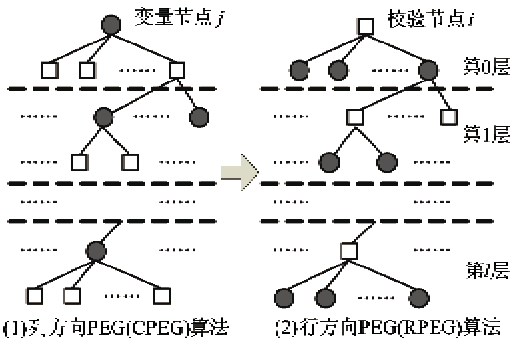

图 3 列方向 PEG（CPEG）算法与 RPEG 算法的展开图比较

3 消除行间访问冲突的 RCAB-PEG 构造算法

据第 1 节分析,为避免 U_j 访问冲突,准循环 LDPC 码的基矩阵需满足的 *ADeg* 约束条件如下:基础矩阵 ***Hb*** 的任意一行中至少要有 *ADeg* 个非零元素与上一行中的所有非零元素处于不同列,规避度 *ADeg* 典型值取 5 或 8。本节提出相应的构造算法。

块渐进边增长(block-PEG, BPEG[17])构造算法是推广的 PEG 算法,可构造 QC-LDPC 码。将 RPEG 算法与 BPEG 算法结合,同时在构造算法中添加 *ADeg* 约束条件,可得行向冲突规避块 PEG(row-direction conflict avoidance block PEG, RCAB-PEG)构造算法。为简单起见,RCAB-PEG 算法流程表述中不区分校验节点和行,不区分变量节点和列。

RCAB-PEG 构造算法流程如表 4 所示:

总结 RCAB-PEG 构造法,第 3-a)步体现了 PEG 算法的本地环长最大化准则;第 3-b)步体现了行方向 PEG 方法,并支持按规划的行和列维度分布进行码构造;3-c)体现了 BPEG QC-LDPC 码构造算法;3-d)体现了 *ADeg* 约束的要求。本算法优先支持环长等性能因素,兼顾 *ADeg* 结构约束,所造码属于 BPEG 码[17]子集,具有 PEG 码优秀的误码率性能。

可用图 4 偏移量矩阵为例来说明码构造。该 QC-LDPC 码长为 1344,码率为 0.5,对应的循环移位子矩阵为 84×84。该码为系统码,基矩阵左 8 列对应校验位,右 8 列对应信息位。为了编码实现方便,该码采用双对角线结构,这个可在算法第 2 步初始化阶段加入,可对第 8 列的偏移量为 84 的子矩阵扣掉一个 1,得到编码入手点。设定的译码块行顺序为{1,2,3,4,5,6,7,8}循环。构造过程冲突规避约束为 *ADeg* =5,比如图 4 中第 2 行非零元素位置集合{2,10,11,13,16},共有 5 个元素和第 1 行的非零元素不共列,所以该码可以保证 DLY≤5 的行分层译码器的流水线中无空闲节拍,实现高效译码。

表 4 RCAB-PEG 构造算法流程

1. 初始化 LDPC 校验矩阵参数:基础矩阵 ***Hb*** 的大小 $m \times n$、Block 矩阵的大小 $p \times p$、变量节点维度分布(d_v),规定行分层译码处理的行顺序。

2. 根据变量节点维度分布确定校验节点的均匀化维度分布(d_c);向 Tanner 图添加所有 n 个变量节点。

3. 向 Tanner 图中逐个添加校验节点,挑选变量节点建立连接,并确定边线权重。

For i(按规定的行处理顺序的行号进行遍历)

For $k=0$: d_c^i-1(d_c^i 为第 i 个校验节点的维度,此步为每个校验节点安排变量节点连接)

a) 如果该校验节点的当前维度等于 **0**,则从变量节点中随机选择一个节点作为目的节点;否则以第 i 个校验节点为根节点展开二分图,选择距离根节点最远的变量节点作为备选集合 $\mathbf{S}_1$。

b) 优先选取列重增长慢的列做候选变量节点。计算 $\mathbf{S}_1$ 集合对应各列当前列重与其预期列重量比值 *Rcw*,选出其中最小值 *Rcw _ min*。从集合 $\mathbf{S}_1$ 中选择 $Rcw \leqslant Rcw\_min \cdot C_{\Delta}$($C_{\Delta}$经验值为 **1.5**)的变量节点形成备选集合 $\mathbf{S}_2$。

c) 遍历根节点到备选变量节点的所有路径,根据 $s=\sum_{k=0}^{2l-2}(-1)^k p_{i_k,j_k}$,计算路径累积权重 s,其中 p_{i_k,j_k} 为已有边线权重,$2l$ 为路径环长;并按照随机原则挑选边线权重 p_0[17],选择 $\mathbf{S}_2$ 中所有路径都能保证 LDPC 码不出现长为 **4** 的环,即 $\mathrm{mod}(s-p_0,p)\neq 0$ 的变量节点,形成备选集合 $\mathbf{S}_3$(此步可反复挑选 p_0,以获得非空 $\mathbf{S}_3$)。如果 $\mathbf{S}_3$ 为空集,则退出循环,回到 **3**;否则继续到 d)。

d) 根据上一个校验节点的连接关系,从集合 $\mathbf{S}_3$ 中选择支持 *ADeg* 约束条件的变量节点形成集合 $\mathbf{S}_4$。即在 $\mathbf{S}_3$ 中选择变量节点时,要优先选出避开上行非零元素位置的元素。如果选不到避开的,查当前校验节点维度规划,看还剩多少个变量节点没有连接,如果后续还有可能凑够 *ADeg* 个规避变量节点,就让 $\mathbf{S}_4=\mathbf{S}_3$,否则 $\mathbf{S}_4$ 空集。如果 $\mathbf{S}_4$ 为空集,则退出循环,回到 **3**;否则继续到 e)。

e) 从集合 $\mathbf{S}_4$ 中随机选择一个节点,与根节点建立连接,并设置相应的权重 p_0。

End

End

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	0	0	0	0	0	**0**	84	22	0	0	49	0	81	58	0
2	1	1	0	0	0	0	**0**	0	0	44	81	0	53	**0**	0	50
3	0	1	1	0	0	0	**0**	0	45	0	0	20	0	29	**44**	0
4	0	0	1	1	0	0	**0**	0	0	4	55	0	37	**0**	0	18
5	0	0	0	1	1	0	**0**	0	53	0	0	11	0	19	67	0
6	0	0	0	0	1	1	**0**	0	0	29	49	0	57	**0**	0	53
7	0	0	0	0	0	1	1	0	8	0	0	76	0	45	53	0
8	0	0	0	0	0	0	1	1	0	51	17	0	27	**0**	0	59

图 4 规避度 *ADeg* =5 的 QC-LDPC 码基矩阵例子

表 5 给出 RCAB-PEG 构造的常见码长、码率 QC-LDPC 码例子，表中包括码参数、布通率、行分层译码算法所需的平均迭代次数。表 5 中布通率表示在表中的约束条件下，能够成功构造的码数目与构造尝试次数的比值。表 5 的构造终止条件为成功构造 100 个 LDPC 码或者尝试次数达到 5000。平均迭代次数表示在各行成功构造的 100 个 LDPC 码，采用相同的典型信噪比条件（即能使平均迭代次数取在 10 左右的一个信噪比）所需的平均迭代次数。根据平均迭代次数能较快地初步判断生成的 LDPC 码的误码性能，并为后续误码率仿真选出候选码。

表 5 RCAB-PEG 算法构造的常见码结果表

码长 N	码率 R	块矩阵大小 p	规避度 $ADeg$	RCAB-PEG 布通率	BPEG 布通率	RCAB- PEG 平均迭代次数	BPEG 平均迭代次数
2016	0.5	24	5	1.000	1.000	12.28	12.96
2016	0.5	42	5	0.980	0.962	12.33	12.94
2016	0.5	84	5	0.109	0.893	12.43	12.88
2016	0.75	84	5	0.025	0.343	12.23	12.47
8064	0.5	48	5	1.000	1.000	12.22	12.19
8064	0.5	112	5	0.158	1.000	12.23	12.27
8064	0.5	252	5	0.124	0.962	12.23	12.23
8064	0.5	504	5	0.000	0.952	N/A	12.20
8064	0.75	48	5	0.550	0.719	11.29	10.83
8064	0.75	48	8	0.383	0.719	11.24	10.83
8064	0.75	112	5	0.174	0.658	11.27	10.78
8064	0.75	112	8	0.105	0.658	11.22	10.78
8064	0.75	252	5	0.004	0.516	11.78	10.79
8064	0.75	252	8	0.000	0.516	N/A	10.79
16200	4/9	360	5	1.000	0.990	10.5	12.5
16200	11/15	360	5	0.625	0.971	8.18	6.35
16200	11/15	360	8	0.246	0.971	7.21	6.35
64800	0.5	360	5	1.000	1.000	8.07	9.53
64800	0.75	360	5	0.980	0.980	6.22	7.55
64800	0.75	360	8	0.971	0.980	6.26	7.55

表 5 的构造结果显示，RCAB-PEG 算法在相同的码长码率下，子矩阵越大越不易构造成功，$ADeg$ 值越大越不易构造成功。RCAB-PEG 对所构造码进行 ADeg 约束，得到的 LDPC 码的平均迭代次数基本与 Block PEG 算法码相当。这一点也是 RCAB-PEG 支持行列维度分布约束和环长最大化带来的好处。

从表 5 中也可看出 RCAB-PEG 算法增加了 $ADeg$ 结构化约束后，算法布通率相对于 BPEG 算法有所降低，这和维度分布、码长、码率、p 值等综合因素有关。但算法中存在随机选择环节，码构造可经过多次尝试达到成功。适当放宽约束也可提高布通率，也不失为次优解。比如增大算法第 3-b）步 C_{∞} 值，扩大预选集，弱化了维度要求，对于很多长码性能没有恶化。也可将 $ADeg$ 值约束调小一些，以支持 DLY 值小的译码电路。$Adeg$ 值即使达不到图 1 中要求的 DLY 值，所获的尽量大 $ADeg$ 值的码也能最大限度地消除空闲节拍。总之，可根据实际需求对 RCAB-PEG 算法做出合理的调整。

4 RCAB-PEG 算法码性能分析与讨论

本节给出用 RCAB-PEG 算法构造的 LDPC 码的误码率性能仿真情况。首先利用 RCAB-PEG 算法构造得到 $ADeg=5$，码长分别为 8064（$p=252$），2016（$p=84$），64800（$p=360$），码率分别为 0.5 和 0.75 的 6 个 LDPC 码。然后用 20 次迭代的行分层译码算法仿真其误码率性能，分别与 BPEG 构造得到的 8064 码长、WiMAX 中 2016 码长、DVB-SII 中 64800 码长的 LDPC 码的误码性能进行对比，仿真所得的误码率曲线参见图 5 和图 6。

从图5和图6的对比中可看到，RCAB-PEG算

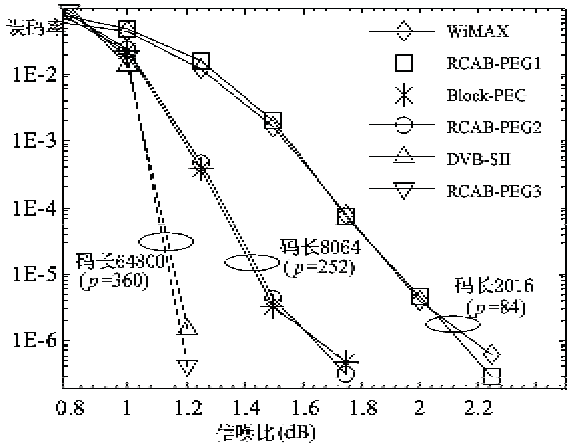

图 5　码率为 0.5 的 RCAB-PEG 码与标准码的误码率对比

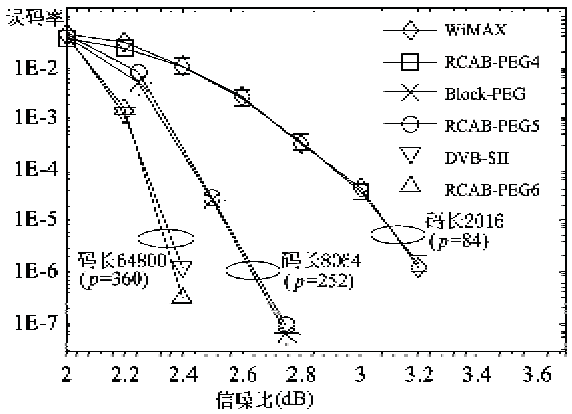

图 6　码率为 0.75 的 RCAB-PEG 码与标准码的误码率对比

法得到的 LDPC 码与 BPEG 算法得到的 LDPC 码以及 WiMAX、DVB-SII 标准中的 LDPC 码具有相当的性能。

采用图 1、图 2 优化结构来实现 DLY = 5 的 QC-LDPC 流水线译码器,可比较 RCAB-PEG 码和普通码的译码速率。采用图 4 的码,*ADeg* = 5,需在换行期间插入 DLY − *ADeg* = 0 个空闲节拍,行维度 dc = 6,故每行处理所需节拍数为 dc + DLY − *ADeg* = 6T。而和图 4 相同大小的普通 QC-LDPC 码典型 *ADeg* = 3,需要插入 DLY − *ADeg* = 2 拍的空闲等待,一行处理所需要的总拍数为 dc + DLY − *ADeg* = 8T。所以图 4 码对应译码器的速率为普通码的 8T/6T≈1.33 倍,并且图 4 码在设计过程中已考虑了行间关系,不需复杂的逐行调度设计,使译码器的实现更加简单。

5　结 论

本文针对行分层译码算器中的变量访问冲突问题,提出了一种消除行分层译码器访问冲突的 QC-LDPC 码构造算法(RCAB-PEG 算法),该算法采用行方向 PEG 算法,并支持行间规避约束。该构造算法提前消除了访问冲突问题,故分层译码器不需要调度等有关复杂设计,也避免付出有关的资源速率代价。RCAB-PEG 码能在不增加资源的情况下充分发挥行分层译码器的吞吐率潜力,典型情形下,采用 RCAB-PEG 码使译码器速率可提高到 1.33 倍。而仿真结果表明,生成的 LDPC 码的性能与 BPEG 码及 WiMAX、DVB-SII 标准码的性能具有可比性。本文属于针对译码器实现的码构造算法研究,参照本文工作,可探索适于不同译码方式的 LDPC 码构造方法。

参考文献

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

[2] Kschischang F, Frey B. Iterative decoding of compound codes by probability propagation in graphical models. *IEEE Journal on Selected Areas in Comm*, 1998, 16(2): 219-230

[3] Wang Z, Cui Z, Sha J. VLSI design for low-density parity-check code decoding. *IEEE Circuits and Systems Magazine*, 2011, 11(1): 52-69

[4] Mansour M M, Shanbhag N R. Turbo decoder architectures for low-density parity-check codes. In: Proceedings of the IEEE Global Telecommunications Conference, Taipei, China, 2002. 1383-1388

[5] Mansour M M, Shanbhag N R. High-throughput LDPC decoders. *IEEE Trans on VLSI Systems*, 2003, 11(6): 976-996

[6] Hocevar D E. A reduced complexity decoder architecture via layered decoding of LDPC codes. In: Proceedings of the IEEE Workshop on Signal Processing Systems, Austin, USA, 2004. 107-112

[7] Zhang J, Fossorier M. Shuffled belief propagation decoding. In: Proceedings of the 36th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, USA, 2002. 8-15

[8] Kfir H, Kanter I. Parallel versus sequential updating for belief propagation decoding. *Physica A*, 2003, 330: 259-270

[9] Wang Z, Mu Q. A Low-complexity layered decoding algorithm for LDPC codes. In: Proceedings of the IEEE International Forum on Computer Science-Technology and Applications, Chengdu, China, 2009. 318-320

[10] He Z, Roy S, Fortier P. FPGA implementation of LDPC decoders based on joint row-column decoding algorithm.

In: Proceedings of the IEEE International Symposium on Circuits and Systems, New Orleans, USA, 2007. 1653-1656

[11] Sun Y, Karkooti M, Cavallaro J. High throughput, parallel, scalable LDPC encoder/decoder architecture for OFDM systems design. In: Proceedings of the IEEE Dallas/CAS Workshop on Applications, Integration and Software, Dallas, USA, 2006. 39-42

[12] Bhatt T, Sundaramurthy V, Stolpman V, et al. Pipelined block-serial decoder architecture for structured LDPC codes. In: Proceedings of the IEEE International Conference on Speech and Signal Processing, Toulouse, France, 2006. 225-228

[13] Rovini M, Gentile G, Rossi F, et al. A minimum-latency block-serial architecture of a decoder for IEEE 802.11n LDPC codes. In: Proceedings of the IFIP International Conference on Very Large Scale Integration, Atlanta, USA, 2007. 236-241

[14] Jin J, Tsui C. An energy efficient layered decoding architecture for LDPC decoder. *IEEE Trans on VLSI Systems*, 2010, 18(8): 1185-1195

[15] Marchand C, Dore J, Conde-Canencia L, et al. Conflict resolution for pipelined layered LDPC decoders. In: Proceedings of the IEEE workshop on Signal Processing Systems, Tampere, Finland, 2009. 220-225

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

[17] 乔华,管武,董明科等. 一种基于循环移位矩阵的 LDPC 码构造方法. 电子与信息学报,2008, 30(10): 2384-2387

An LDPC code construction method for cancelling access conflict in a layered decoder

Dong Mingke, Zhang Jianjun, Wu Jianjun, Xiang Haige

(State Key Laboratory of Advanced Optical Communication Systems & Networks,
School of Electronics Engineering and Computer Science, Peking University, Beijing 100871)

Abstract

Aiming at the access conflict problem existing in row-layered decoder for ordinarily structured low-density parity check (LDPC) codes, a novel LDPC code construction method based on the progressive edge-growth (PEG) algorithm, called the RCBA (row-direction conflict avoidance block)-PEG algorithm, is proposed. The new RCBA-PEG algorithm constructs the base parity check matrix for quasi-cyclic (QC)-LDPC codes row-by-row with the row-conflict-avoidance constraint. The new constructed codes do not induce access conflicts, which can simplify the layered decoder design, save resources and enhance throughput. The rate of the typical decoder for the constructed codes speeds up to 1.33 times of the ordinary one. The simulations show that the BER performance of the new constructed codes is similar to that of the Block-PEG codes and WiMAX, DVB-SII codes.

Key words: low-density parity check (LDPC), quasi-cyclic (QC)-LDPC, LDPC layered decoding, progressive edge-growth (PEG), LDPC decoder, access conflict