基于帧交错的 LDPC 译码器流水结构设计

韩国军，杨伟泽，叶震亮，翟雄飞*，史治平

（广东工业大学 信息工程学院，广州 510006）

摘要 低密度奇偶校验码（LDPC）的译码器通常采用基于节点置信度更新迭代的算法，这种算法可以并行实现，具有非常高的吞吐量。在此提出了一种具有高硬件利用效率（HUE）的帧交错译码结构，并提供了一种用于层内节点重排序的动态规划方法，解决内存访问冲突问题。与现有的结构相比，该结构可以实现更高的硬件利用效率。

关键词 帧交错；低密度奇偶校验码；内存访问冲突；节点重排序

中图分类号 TP911 **文献标志码** A DOI 10.12178/1001-0548.2023023

Pipeline Design of LDPC Decoder Based on Frame-Interleaving

HAN Guojun, YANG Weize, YE Zhenliang, ZHAI Xiongfei*, and SHI Zhiping

(School of Information Engineering, Guangdong University of Technology, Guangzhou 510006, China)

Abstract The decoder of low density parity check code (LDPC) generally adopts an iterative algorithm based on node confidence update, which can be implemented in parallel and has very high throughput. In this paper, we propose a frame-interleaving decoding structure with high hardware utilization efficiency (HUE) features and develop a dynamic planning method for node reordering within layers, which can solve the memory access conflict problems. Compared with the existing structures, the proposed structure shows more efficiency with respect to hardware utilization.

Key words frame-interleaving; low density parity check code; memory access conflict; node reordering

低密度奇偶校验（Low Density Parity Check, LDPC）码的性能逼近香农限[1]，其译码方法一般采用基于节点置信度更新迭代的算法，这种基于置信度传播的思想是一种逼近香农极限的非常有效的途径[2]。除此之外，准循环 LDPC 码的并行特性使其能够通过引入少量额外硬件来线性地提高译码器吞吐量[3]。正因为 LDPC 码的优异性能使得它成为近年来通信及存储领域编码研究的热点。在 2016 年 10 月的 3GPP 会议上确定了将 LDPC 作为第五代移动通信技术（the Fifth Generation, 5G）的数据信道编码方案[4]。

文献 [5-6] 采用完全并行结构，实现了上百 Gbps 级的吞吐量，虽然此结构能够满足 5G 标准的高速译码需求，但该结构硬件利用效率非常低，且会带来严重的路由阻塞问题。于是人们通常采用硬件利用效率更高的块并行结构，而 5G 标准规定的基图的行重及列重高度不对称，不同层之间的正交性较差，这就导致了基于块并行结构的译码器不可避免地存在内存访问冲突的问题。

针对内存访问冲突的问题，现有解决方案主要有以下两个。一是尽量避免冲突的发生，具体做法是对层顺序和层内节点进行重新排序[6-7]、分层半并行译码[8]、在流水阶级中插入空闲周期等[9-10]。其中层顺序调换操作会对译码性能产生影响[11]，分层半并行译码以较低吞吐量为代价，插入空闲周期会降

收稿日期：2023−01−16；修回日期：2023−03−29

基金项目：国家自然科学基金青年项目（62301166）；NSFC-广东省联合基金（U2001203）；2020 广东省重点领域研发计划“芯片、软件与计算（芯片类）”专项（2021B1101270001）；广州市基础与应用基础研究项目（202102020869）；广东省自然科学基金面上项目（2022A1515010153）；四川省自然科学基金（2022NSFSC0488）

作者简介：韩国军，教授，博士，主要从事差错控制编译码技术方面的工作。

*通信作者 E-mail: zhaixiongfei@gdut.edu.cn

低吞吐率，因此需尽量避免这些操作。二是降低发生冲突后对译码带来的影响，文献 [12] 采用一种新的调度方式，当节点是冲突节点时，用一种新的算法进行该节点更新，减小该冲突节点更新值的误差，但这种方法并不能完全解决内存访问冲突问题。

为此本文提出了一种基于帧交错的译码结构，采用多帧交错译码方法，同时最小和计算模块采用多级缓冲机制，结合层内节点重排序算法，能够大幅减少为避免内存访问冲突而设置的空白周期。并采用乒乓机制对初始 LLR 值进行预缓存，节省初始化所需要的时间。

1　5G LDPC 和分层译码算法

1.1　5G LDPC 结构

5G 标准采用准循环 LDPC（Quasi-cyclic Low Density Parity Check, QC-LDPC）码，其校验矩阵可以用一个基图（Base Graph, BG）和相应的移位因子 Z 来表示，即基图中每个元素都展开为 $Z\times Z$ 大小的单位矩阵的循环矩阵，移位大小由 Z 和该元素值确定。特别地，元素“−1”展开为 $Z\times Z$ 的全零矩阵。3GPP 会议确定 5G LDPC 由一个高码率校验矩阵进行码字扩展得到，根据基图可对校验矩阵进行划分，如图 1 所示。

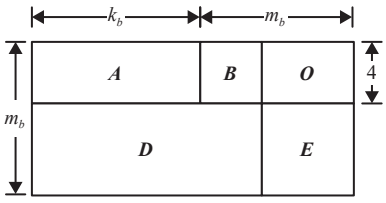

图 1　5G LDPC 基图划分

图中 $\boldsymbol{A}$ 是维度 $4\times k_b$ 的密集矩阵，对应信息比特，$\boldsymbol{B}$ 是维度 4×4 的双对角矩阵，对应校验比特，$\boldsymbol{O}$ 是维度 $4\times(m_b-4)$ 的全零矩阵，$\boldsymbol{D}$ 是维度 $(m_b-4)\times(k_b+4)$ 的稀疏矩阵，$\boldsymbol{E}$ 是维度 $(m_b-4)\times(m_b-4)$ 的单位矩阵。$[\boldsymbol{A},\boldsymbol{B}]$ 构成校验矩阵的核心 $\boldsymbol{H}_{\text{core}}$。5G 标准的校验矩阵通过 $\boldsymbol{H}_{\text{core}}$ 扩展得到。

考虑到应用场景的多样性，5G 给出了两个基图 BG1 和 BG2 以实现不同的码长和码率。表 1 列出了这两个基图的基本参数。从表中可以看出，两个矩阵的节点度分布不均匀，导致实际实现中存在内存访问冲突问题。

表 1　5G LDPC 基图特性

参数	基图	
	BG1	BG2
尺寸	46×68	42×52
$\boldsymbol{H}_{\text{core}}$	4×26	4×14
非零元素	316	197
列重	1～30	1～23
行重	3～19	3～10

1.2　分层译码算法

传统的 LDPC 迭代译码算法是基于全并行的泛洪调度[13]，这种调度方式是以较大的资源消耗来换取较高的吞吐量。为了实现吞吐量和硬件资源的折中，文献 [14] 提出了分层译码算法，其基本思想是将校验矩阵按行分层，依次更新每层节点的信息，上一层更新的节点信息可以立即用于下一层的更新，其收敛速度是全并行调度的两倍。

在硬件实现中通常以 Z 行作为一层，由于准循环特性，层内变量节点具有完美的正交性，因此能实现 Z 个节点并行译码。为了便于表述，本文以一行作为一层，表 2 对译码过程出现的符号做了详细说明，调制方式为 BPSK，在高斯信道下，译码过程如下。

表 2　符号说明

符号	说明
$\boldsymbol{H}$	校验矩阵
m, n	矩阵行数和列数
l	层序号
t	迭代次序
y_j	第j个变量节点的信道消息
σ^2	高斯白噪声方差
c_i, v_j	第i / j个变量/校验节点
N_{c_i}	与c_i有连接关系的变量节点的集合
N_{v_j}	与v_j有连接关系的校验节点的集合
$N_{c_i}\backslash v_j$	除了v_j，其他与c_i有连接关系的变量节点的集合
$N_{v_j}\backslash c_i$	除了c_i，其他与v_j有连接关系的校验节点的集合
$L_{v_j\to c_i}^{(t)}$	第t次迭代时，v_j传递给c_i的信息
$L_{c_i\to v_j}^{(t)}$	第t次迭代时，c_i传递给v_j的信息
$L_j^{(t)}(l)$	第t次迭代时，v_j在第l层更新的后验概率信息

在第一次迭代时，变量节点的后验概率信息根据信道接收的似然信息进行初始化，校验节点信息初始化为零：

$$L_j^{(1)}(0)=\frac{2y_j}{\sigma^2} \tag{1}$$

$$L_{c_i\to v_j}^{(0)}=0 \tag{2}$$

初始化完成后开始译码迭代，逐层更新。

1）逐行更新第 l 层变量节点向校验节点传递的信息：

$$L_{v_j\to c_i}^{(t)}=L_j^{(t)}(l-1)-L_{c_i\to v_j}^{(t-1)} \tag{3}$$

2）逐行更新第 l 层校验节点向变量节点传递的信息：

$$L_{c_i\to v_j}^{(t)}=\left[\sum_{v_b\in N_{c_i}\backslash v_j}-\ln\left(\tan\frac{\left|L_{v_b\to c_i}^{(t)}\right|}{2}\right)\right]\times \prod_{v_b\in N_{c_i}\backslash v_j}\operatorname{sgn}\left(L_{v_b\to c_i}^{(t)}\right) \tag{4}$$

3）更新第 l 层后验概率：

$$L_j^{(t)}(l)=L_{v_j\to c_i}^{(t)}+L_{c_i\to v_j}^{(t)} \tag{5}$$

4）译码结果判决：

$$r_j=\begin{cases}1 & L_j^{(t)}(l)<0\\ 0 & L_j^{(t)}(l)\geqslant 0\end{cases} \tag{6}$$

令 $\boldsymbol{r}=[r_0,r_1,\cdots,r_n]^{\mathrm{T}}$，如果 $\boldsymbol{Hr}=0$，退出迭代，输出 $\boldsymbol{r}$ 为译码结果，否则返回步骤 1）继续迭代，直到达到最大迭代次数。

上述译码过程中，最复杂的是步骤 2），在实现中通常采用查表法，需要消耗大量硬件资源，因此一般采用低复杂度的归一化最小和算法（Normalized Minimum Sum, NMS）：

$$L_{c_i\to v_j}^{(t)}=\alpha M^{(t}(i)\prod_{v_b\in N_{c_i}\backslash v_j}\operatorname{sgn}\left(L_{v_b\to c_i}^{(t)}\right) \tag{7}$$

其中，

$$M^{(t)}(i)=\min_{v_b\in N_{c_i}\backslash v_j}\left|L_{v_b\to c_i}^{(t)}\right| \tag{8}$$

$\alpha\in[0.5,1]$称为归一化因子。相比于和积译码算法，NMS 仅需计算最小值和次小值，在硬件设计中容易实现。

1.3 传统译码结构

传统分层译码结构如图 2 所示，节点地址存储单元根据校验节点存储变量节点和校验节点的索引值，最小和计算单元和节点更新单元对应上述译码步骤 2）和 3），判决模块负责对译码结果进行检验，对应步骤 4）。

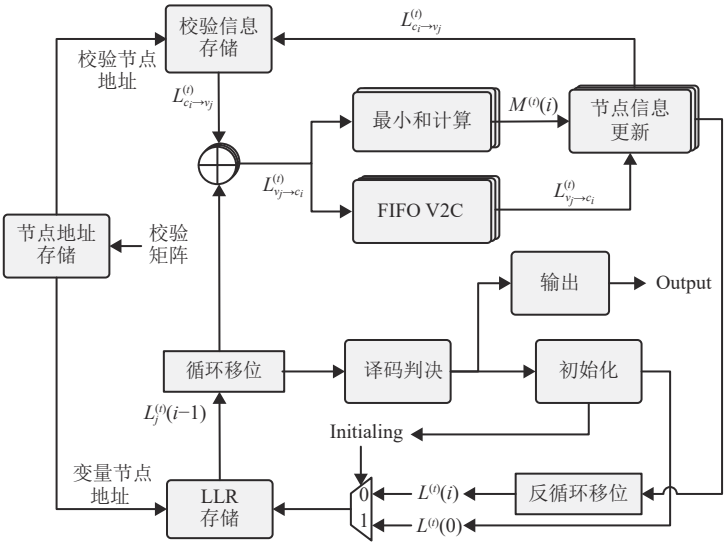

图 2 LDPC 传统译码结构

为了提高译码吞吐量，通常采用流水线设计，然而流水线设计常常会导致内存访问冲突问题，主要发生在以下两个方面。

1）当前层在读取节点信息时，上一层的 LLR 值尚未更新和写入到存储器中，因此当前层无法获得最新的 LLR 值，即式 (3) 参与运算的实际是 $L_j^{(t)}(l-2)$而非 $L_j^{(t)}(l-1)$。节点更新采用了错误的 LLR 值。

2）当前层节点更新还没有完成，下一层的节点最小值就已经准备好，此时节点更新用的是下一层的最小值，即式 (7) 参与运算的实际是$M^{(t)}(i+1)$而非$M^{(t)}(i)$。节点更新采用了错误的最小值。

2　帧交错译码器流水线设计

2.1　帧交错结构

本文提出的帧交错译码结构如图 3 所示，与传统译码结构相比，该结构主要有以下 4 点改进。

1） 插入一个帧控制单元，在多帧之间来回切换，根据帧索引信号选择进行译码的存储单元。

2） 采用一个 FIFO 作为多级缓冲结构，可以缓存多个最小值，当上层节点更新完成时，再将下层最小值从 FIFO 读出。该操作需要一个时钟周期，相应地增加一个流水阶级。

3） 插入一个译码输出选择器，使译码器可以在进行初始化时同时读出前一帧的译码结果，与传统结构相比，节省了额外用于存储译码结果的资源。

4） 采用乒乓式缓存机制，即将存储器分为两组，分别用于预存初始化信息和当前帧的译码，工作时交替使用，节省初始化信息需要的时间。

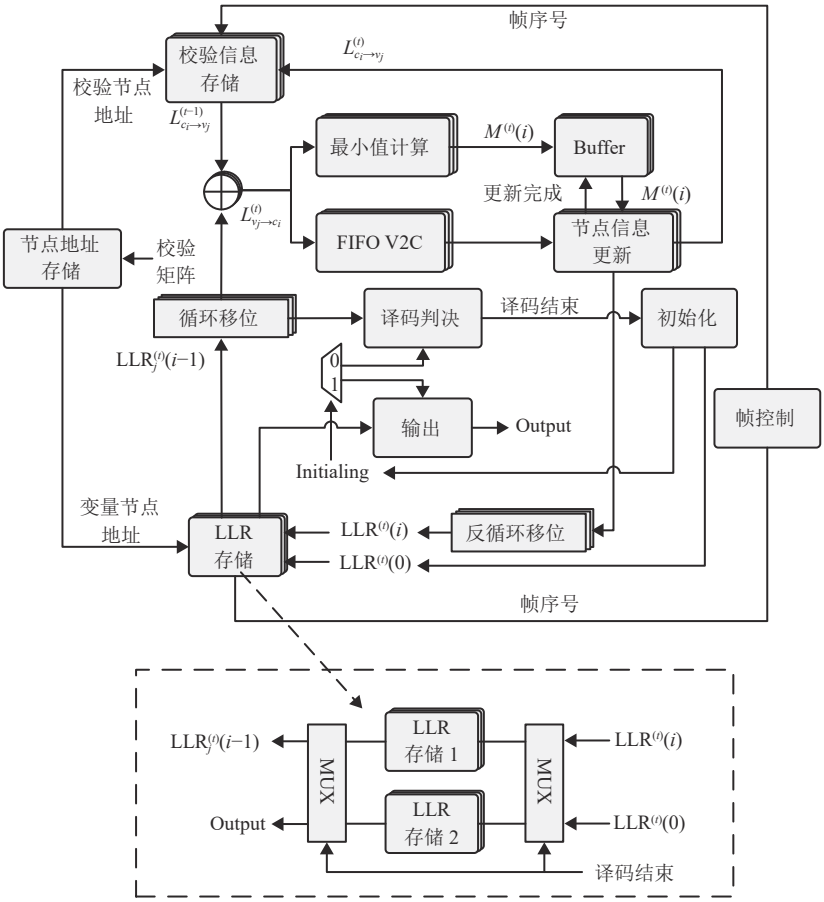

图 3　帧交错结构

2.2　动态规划

通常节点更新顺序是按照其在矩阵的位置逐个进行，若同一变量节点在连续两层中出现，其间距太小很容易发生访问冲突，因此有必要对节点进行层内重排序，使得同一变量节点间距足够长，且同一层的节点进行重排序不会影响译码性能。

若一个节点在连续两层中同时出现，称它在上一层的节点为父节点，在下一层的节点为子节点。一个节点可以同时是父节点和子节点。动态规划的基本思想是父节点提前更新，子节点推迟读取，使得父节点有足够的时钟周期完成更新。具体规划如算法 1 所示。

节点的位置关系着该节点在硬件实现中读写的先后顺序，即同一层位置靠左的节点优先读写，图 4 给出了一个具体的例子，图中黄色标识的节点代表符合移动条件的节点。以序号为 7 的节点为例，在进行重排序之前，该节点在层 1 到层 2 的间距为 6，在层 2 到层 3 的距离为 4。经过重排序后间距都为 8，足够的间距能够有效避免内存访问冲突。

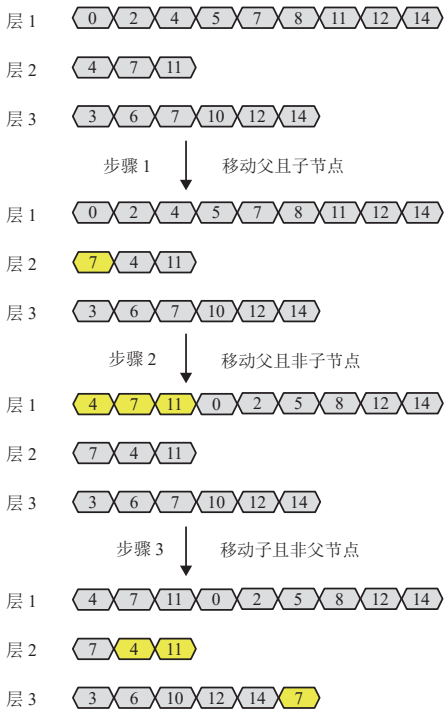

图 4　层内节点动态规划示例

算法 1　层内节点动态规划

for 校验节点每一层 do

　for 从右到左每个节点 do

　　if 该节点同时是父节点和子节点 then

　　移动该节点至最左端

　　end if

　end for

　for 从右到左每个节点 do

　　if 该节点是父节点且非子节点 then

　　移动该节点至最左端

　　end if

　end for

　for 从左到右每个节点 do

　　if 该节点是子节点且非父节点 then

　　移动该节点至最右端

　　end if

　end for

end for

2.3　时序分析

图 5 展示了在校验矩阵行重高度不对称的情况下，不同译码结构的时序分析。图中主要展示的是节点信息的读和写过程，整个流水结构包括 5 个流水阶级，在读取完一层节点的 5 个时钟周期后开始逐个写入节点信息。

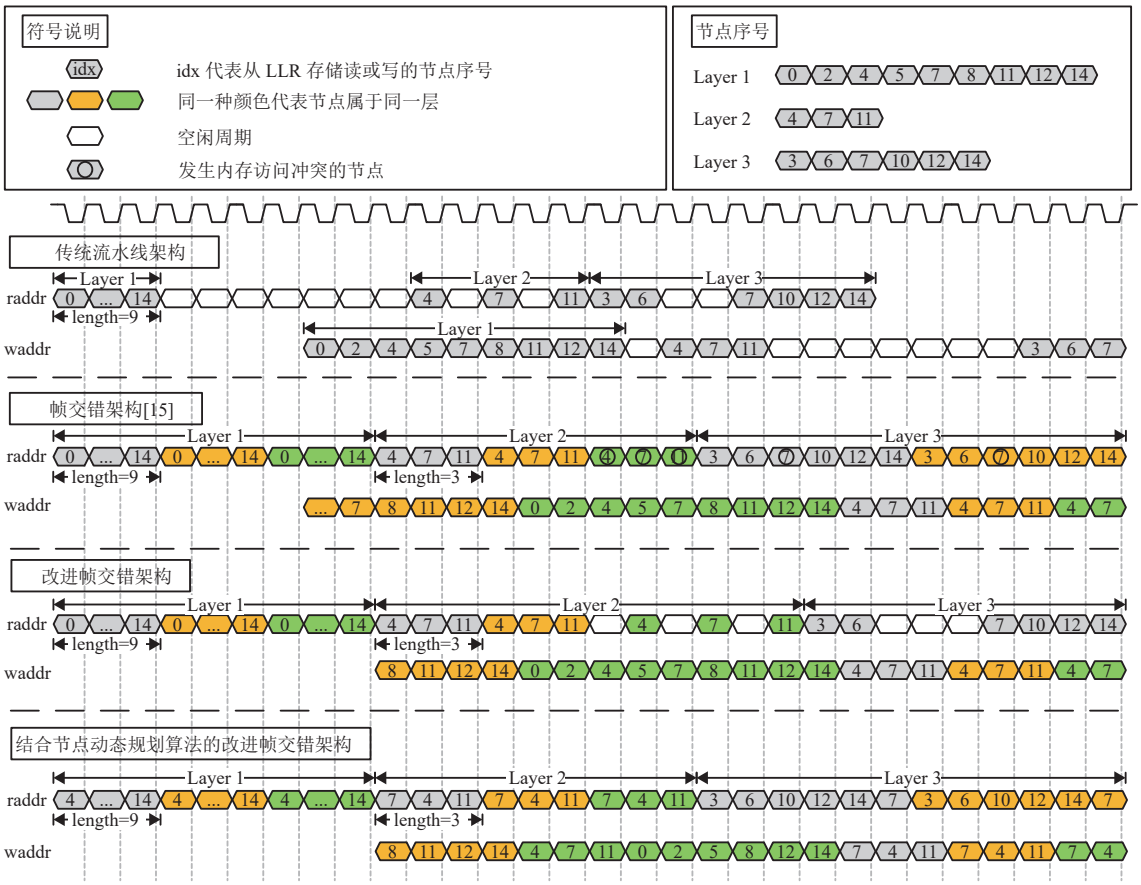

图 5　不同结构在行重高度不均匀情况下的时序分析

可以看到，在没有采用帧交错结构的情况下，传统译码结构完成一次译码迭代需要插入 11 个空闲周期。文献 [16] 提出了简单的帧交错结构，但在行重高度不对称的情况下仍然存在内存访问冲突问题。本文在此基础上做了改进，插入了必要的空闲周期，平均每一帧插入两个空周期，相比传统结构提升了吞吐量。

除了以上对于帧交错结构的改进，结合本文提出节点动态规划问题，能够进一步减少空闲周期的插入，极大地提高吞吐量。

3　实验与对比

本实验选择长度为 2 176，速率为 1/3，基图为 BG1 的 LDPC 码，交错帧数为 3。根据上述推导，本结构的吞吐量 Th 为：

$$\mathrm{Th}=\frac{F_{\mathrm{clk}}n}{n_c\bar{n}_l} \tag{9}$$

式中，F_{clk} 代表译码器工作频率；n 代表码字长度；n_c 代表一次迭代所需要的周期，即基图的非零元数目；$\bar{n}_l$ 代表平均迭代次数。为了实验更具有可比性，采用归一化吞吐量进行对比，有：

$$\mathrm{Th}_{\mathrm{norm}}=\mathrm{Th}\bar{n}_l \tag{10}$$

表 3 对比了本文提出的结构与已有文献在资源利用率和归一化吞吐量的比较，表中量化等级（6,8,6）表示 6 位初始 LLR 值，8 位变量信息，6 位校验信息；HUE 表示单位资源的归一化吞吐量。从表中可以看到，虽然现有文献能够实现比本结构更高的吞吐量，但消耗了大量的资源。与已有文献相比，本文提出的结果能显著提高单位资源利用率。

表 3　不同结构 FPGA 实现数据对比

方法	结构								HUE		
	FPGA 开发板	码长	量化等级	F_{clk} /MHz	$\mathrm{Th}_{\mathrm{norm}}$ /Mb·s^{-1}	Slice LUTs	Slice Registers	36 k BRAMs	千单位LUT 吞吐量/Mb·s^{-1}	千单位 Registers吞吐量 /Mb·s^{-1}	单位BRAMs 吞吐量/Mb·s^{-1}
本文帧交错结构	Zynq-7 000	(2 176, 704)	(6,8,6)	250	1 721.5	5 910 (14.56%)	2 130 (2.62%)	38 (35.51%)	**291.3**	**808.21**	45.3
混合调度[12]	Virtex-7	(26 112, 8 448)	(8,8,6)	261	20 900	100 929 (23.3%)	85 431 (9.86%)	136.5 (9.3%)	207.08 (40.7%)	244.64 (230.4%)	153.11
灵活帧并行[15]	Kintex-7	(10 368, 8 448)	(8,8,6)	160	11 740	74 373 (73%)	46 517 (23%)	198.5 (30%)	157.85 (84.5%)	252.38 (220.2%)	59.14
全行并行[17]	Virtex Ultrascale+	(26 112, 8 448)	(7,7,4)	102.45	29 000	1 448 762 (35.46%)	211 238 (2.59%)	-	20.17 (1 344.2%)	137.286 (488.7%)	-

4　结束语

本文针对 5G LDPC 码中行重高度不均匀导致的内存访问冲突问题，提出了一种多帧交错译码结构，结合了节点多级缓冲机制、乒乓机制和节点动态规划算法。该结构能很好地解决内存访问冲突问题，提高吞吐量。未来，将继续优化帧交错译码器结构，主要集中于提高译码器的灵活性和兼容性，支持 5G LDPC 更多矩阵更多并行度的译码。

参 考 文 献

[1] CHUNG S Y. On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit[J]. IEEE Communications Letters, 2002, 5(2): 58-60.

[2] GALLAGER R. Low-Density parity-check codes[J]. IRE Transactions on Information Theory, 1962, 8(1): 21-28.

[3] WANG Z, CUI Z. Low-Complexity high-speed decoder design for quasi-cyclic LDPC codes[J]. IEEE Transactions on Very Large Scale Integration Systems, 2007, 15(1): 104-114.

[4] NR. Multiplexing and channel coding(Release 15). 3GPP TS 38.212, 2018 [EB/OL]. [2022-12-29]. https://www.3gpp.org/ftp/Specs/archive/38_series/38.212.

[5] BLANKSBY A J, HOWLAND C J. A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder[J]. IEEE Journal of Solid-State Circuits, 2002, 37(3): 404-412.

[6] GHANAATIAN R, BALATSOUKAS-STIMMING A, MÜLLER T C, et al. A 588-Gb/s LDPC decoder based on finite-alphabet message passing[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2017, 26(2): 329-340.

[7] 袁建国, 曾磊, 孙雪敏, 等. 一种基于交错行列消息传递的 LDPC 码改进译码算法[J]. 半导体光电, 2018, 39(1): 109-112.
YUAN J G, ZENG L, SUN X M, et al. An improved decoding algorithm of LDPC codes based on interlaced column row message-passing[J]. Semiconductor

Optoelectronics, 2018, 39(1): 109-112.
[8] 丁太云. 分层部分并行 LDPC 编译码器研究及 FPGA 实现[D]. 重庆: 重庆邮电大学, 2021.
DING T Y. Research and FPGA implementation of layered partial parallel LDPC codec[D]. Chongqing: Chongqing University of Posts and Telecommunications, 2021.
[9] ZHAO M, ZHANG X, ZHAO L, et al. Design of a high-throughput QC-LDPC decoder with TDMP scheduling[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2015, 62(1): 56-60.
[10] HOCEVAR D E. A reduced complexity decoder architecture via layered decoding of LDPC codes[C]//IEEE Workshop on Signal Processing Systems. [S.l.]: IEEE, 2004: 107-112.
[11] FARSIABI A, BANIHASHEMI A H. Error floor analysis of LDPC row layered decoders: 10.1109/TIT.2021.3099020[P]. 2021.
[12] PETROVIĆ V L, MARKOVIĆ, M M, MEZENI D, et al. Flexible high throughput QC-LDPC decoder with perfect pipeline conflicts resolution and efficient hardware utilization[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2020, 67(12): 5454-5467.
[13] 茅迪. 一种全并行 LDPC 译码器及 FPGA 实现方法[J]. 现代导航, 2019, 10(5): 362-367.
MAO D. Method of parallel LDPC decoder and FPGA implementation[J]. Modern Navigation, 2019, 10(5): 362-367.
[14] MANSOUR M. Turbo decoder architectures for low-density parity-check codes[J]. IEEE Transactions on Very Large Scale Integration Systems, 2002, 7(2): 212-221.
[15] NADAL J, BAGHDADI A. Parallel and flexible 5G LDPC decoder architecture targeting FPGA[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2021, 29(6): 1141-1151.
[16] LI M, DERUDDER V, BERTRAND K, et al. High-Speed LDPC decoders towards 1 Tb/s[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2021, 68(5): 2224-2233.
[17] VERMA A, SHRESTHA R. A new VLSI architecture of next-generation QC-LDPC decoder for 5G new-radio wireless-communication standard[C]//2020 IEEE International Symposium on Circuits and Systems (ISCAS). [S.l.]: IEEE, 2020, DOI: 10.1109/ISCAS45731.2020.9181188.

编 辑 税 红