doi:10.11887/j.cn.202401015 http://journal.nudt.edu.cn

面向GPU的5G新型无线电的高吞吐率LDPC译码器

李荣春[1,2*],周 鑫[1,2],乔 鹏[1,2],王庆林[1,2]
(1. 国防科技大学 计算机学院,湖南 长沙 410073;
2. 国防科技大学 并行与分布计算全国重点实验室,湖南 长沙 410073)

摘 要:提出了一种基于图形处理单元(graphic processing unit,GPU)的5G软件无线电准循环低密度奇偶校验(low density parity check, LDPC)码译码器,为了节省片上和片下带宽,采用码字缩短和打孔技术、两级量化和数据打包方案,以提升数据带宽的利用率。实验基于Nvidia RTX 2080Ti GPU平台实现了高码率情况下的最小和近似译码算法的并行译码,通过分析GPU上的最优线程设置,将码率为5/6的(2 080, 1 760) LDPC算法的译码吞吐率提升至1.38 Gbit/s,译码吞吐率性能优于现有其他基于GPU的LDPC译码器。

关键词:低密度奇偶校验;5G;图形处理单元;软件无线电
中图分类号:TN014 **文献标志码**:A **文章编号**:1001-2486(2024)01-141-08

High-throughput LDPC decoder on GPU for 5G new radio

LI Rongchun[1,2*], *ZHOU Xin*[1,2], *QIAO Peng*[1,2], *WANG Qinglin*[1,2]
(1. College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China;
2. National Key Laboratory of Parallel and Distributed Computing, National University of Defense Technology, Changsha 410073, China)

Abstract: A GPU (graphic processing unit) based 5G software radio quasi cyclic LDPC (low-density parity check) code decoder was proposed. In order to save on chip and off chip bandwidth, code word shortening and punching techniques, two-stage quantization, and data packaging schemes were adopted to improve the utilization of data bandwidth. The experiment was based on the Nvidia RTX 2080Ti GPU platform to achieve parallel decoding of minimum and approximate decoding algorithms under high bit rates. By analyzing the optimal thread settings on the GPU, the decoding throughput of the 5/6 (2 080, 1 760) LDPC algorithm is improved to 1.38 Gbit/s, and the decoding throughput performance is better than other GPU based LDPC decoders.

Keywords: LDPC; 5G; GPU; software defined radio

低密度奇偶校验(low density parity check, LDPC)码[1]是一类前向纠错(forward error correction, FEC)码。LDPC码由于接近最佳性能,已经被WiFi、DVB-S2和WiMAX等通信系统所采用。LDPC码也被纳入5G标准,成为第五代新无线电的信道编码方案。目前,高吞吐率的LDPC译码器取得了一定的进展[2-5],但基于5G无线电的LDPC译码器工作目前并不多。首先,按照第三代合作伙伴计划(3rd generation partnership project, 3GPP)交付的标准描述,5G无线系统的速度将达到20 Gbit/s[6]。编码和译码的吞吐率必须足够高,才能满足高速数据传输的需求。因此,LDPC译码器需要提高译码吞吐率。在应用5G无线电时,需要更高吞吐率的LDPC译码器来实现高速信号传输。其次,5G新型无线电(new radio, NR)标准要求支持更广泛的码率和码长,利用一些新的结构特点,如码型设计和译码技术来实现更好的性能。

基于专用硬件平台(如ASIC和FPGA)的译码吞吐率可以非常高[7-11]。传统FPGA方案可以采用脉动阵列等方式来实现流水并行,提高译码吞吐率,降低译码延迟。同时其功耗较低,被广泛用于各种无线通信系统中。目前,图形处理单元(graphic processing unit, GPU)由于其强大的并行计算能力和灵活的编程形式,被广泛应用于各种科学计算、智能计算中。基于GPU的无线通信译码器也不断涌现,其吞吐率较高,甚至在某些情况下可以超越FGPA[12]。GPU译码器的优势之一是良好的灵活性,利用基于GPU的译码器,可以通过软件实现多标准、多模式的通信系统。基于GPU的译码器可以轻松实现对LDPC译码的

收稿日期:2022-04-29
基金项目:国家自然科学基金资助项目(61902415,62002365)
*第一作者:李荣春(1985—),男,安徽无为人,研究员,博士,硕士生导师,E-mail:rongchunli@nudt.edu.cn

不同码率和码长的支持,从而可以快速实现软件定义无线电平台。此外,开发时间短也使得基于GPU的译码器非常有吸引力。

WiFi和WiMAX系统采用准循环LDPC(quasi cyclic-LDPC, QC-LDPC)码,它是编码和译码复杂度较低的结构化LDPC码。5G无线电标准设计了新的QC-LDPC码。目前已有的基于GPU的QC-LDPC译码器都是特定码长的实现,如IEEE802.16e(2 304,1 152)的LDPC码[13],IEEE802.11n(1 944,972)的LDPC码[13]。但是其缺少支持5G无线电所需更长码长和更高码率的高吞吐率QC-LDPC译码器。

本文提出了一种基于GPU的5G新型无线电高通量QC-LDPC译码器。为了提高译码吞吐率,利用GPU实现了并行MSA算法。在消息更新过程中的最小值和符号计算在GPU上实现了并行计算。此外,实现了对数似然比(log-likelihood ratio, LLR)和中间值的量化,以合理的性能损失来节省片上和片下带宽。同时,还利用数据打包方法来降低GPU和中央处理器(central processing unit, CPU)之间数据传输的开销。在Nvidia RTX 2080Ti上测量了不同码长和码率的吞吐率。最终实验显示,基于GPU的译码器取得了较高的吞吐率和峰值性能利用率。最后,提出了一种评估方法来计算GPU上并行译码的最优线程设置。

1 用于5G无线电的QC-LDPC码

1.1 QC-LDPC码奇偶校验矩阵

LDPC码由$M\times N$奇偶校验矩阵(parity check matrix, PCM)$\boldsymbol{H}$构造,其中M代表校验节点(check node, CN)的数量,N代表变量节点(variable node, VN)的数量。将K表示为信息位的长度,$K=N-M$。PCM中的每一行代表一个奇偶校验方程,如式(1)所示。

$$c_1\oplus c_2\oplus\cdots\oplus c_k=0 \tag{1}$$

式中,$\oplus$表示模2加,$c_1,c_2,\cdots,c_k$是行中的非零位。CN和VN之间的关系也可以用Tanner图[14]来描述。Tanner图中CN_i和VN_j之间的边对应于PCM中的非零条目,这意味着$H(ij)=1$,其中,$i=1,2,\cdots,M$;$j=1,2,\cdots,N$。

QC-LDPC码的PCM可以根据基础矩阵$\boldsymbol{H}_{\mathrm{b}}$来构造,$\boldsymbol{H}_{\mathrm{b}}$是一个$m_{\mathrm{b}}\times n_{\mathrm{b}}$矩阵,由$Z\times Z$右移标识和零矩阵组成。3GPP为NR定义了两个速率兼容的基础矩阵,即$\boldsymbol{BG}_1$和$\boldsymbol{BG}_2$。$\boldsymbol{BG}_1$的目标是编码长度较大($500\leqslant K\leqslant 8\ 448$)、码率较高($1/3\leqslant r\leqslant 8/9$);$\boldsymbol{BG}_2$的目标是编码长度较小($40\leqslant K\leqslant 2\ 560$)、码率较低($1/5\leqslant r\leqslant 2/3$)。$Z$是右移大小,代表右移单位矩阵的大小。3GPP对基本图使用8组右移尺度,值为$Z=2^j\times a$,其中$a\in\{2,3,5,7,9,11,13,15\}$,$j\in\{0,1,2,3,4,5,6,7\}$。表1显示了不同$a$和$j$变化情况下$Z$的大小变化及数值集合,$Z$最大为384。

表1 5G新型无线电协议中的右移系数

Tab. 1 Right shift coefficient in 5G new radio

a	右移系数Z集合	j的变化范围
2	{2,4,8,16,32,64,128,256}	$j=0,1,2,3,4,5,6,7$
3	{3,6,12,24,48,96,192,384}	$j=0,1,2,3,4,5,6,7$
5	{5,10,20,40,80,160,320}	$j=0,1,2,3,4,5,6$
7	{7,14,28,56,112,224}	$j=0,1,2,3,4,5$
9	{9,18,36,72,144,288}	$j=0,1,2,3,4,5$
11	{11,22,44,88,176}	$j=0,1,2,3,4$
13	{13,26,52,104,208}	$j=0,1,2,3,4$
15	{15,30,60,120,240}	$j=0,1,2,3,4$

假设偏移值

$$s=\boldsymbol{H}_{\mathrm{b}}(i_{\mathrm{b}},j_{\mathrm{b}})\in\{-1\}\cup\{0,1,\cdots,Z-1\} \tag{2}$$

式中,$1\leqslant i_{\mathrm{b}}\leqslant m_{\mathrm{b}}$, $1\leqslant j_{\mathrm{b}}\leqslant n_{\mathrm{b}}$。

PCM通过使用该映射对$\boldsymbol{H}_{\mathrm{b}}$进行扩充和单位矩阵的右移来得到。

$$s\rightarrow\begin{cases}\boldsymbol{0} & s=-1\\ \boldsymbol{I}_s & \text{其他}\end{cases} \tag{3}$$

其中:$\boldsymbol{I}_s$是一个$Z\times Z$单位矩阵,经过了s次的循环右移;$\boldsymbol{0}$是$Z\times Z$零矩阵。$\boldsymbol{H}$的大小为$(m_{\mathrm{b}}\times Z)\times(n_{\mathrm{b}}\times Z)$。对于$\boldsymbol{BG}_1$,$m_{\mathrm{b}}=46$,$n_{\mathrm{b}}=68$,信息列数量$k_{\mathrm{b}}=n_{\mathrm{b}}-m_{\mathrm{b}}=22$。在5G新型无线电中,$m_{\mathrm{b}}$和$n_{\mathrm{b}}$可以调整以适应码率。以下步骤描述了码率$r$和待编码的码长$W$条件下QC-LDPC的PCM构造过程。

步骤1:求满足以下条件的缩短长度N_{s}。

$$\frac{k_{\mathrm{b}}}{n_{\mathrm{b}}-N_{\mathrm{s}}-2}-d<r<\frac{k_{\mathrm{b}}}{n_{\mathrm{b}}-N_{\mathrm{s}}-2}+d \tag{4}$$

式中,k_{b}为每行的非零条目,n_{b}为校验矩阵宽度,N_{s}为缩短长度,d为最大码率偏差。按照上述构造方法,目标码率可能无法完全满足要求,因此允许最终码率与目标码率有微小的偏差。

步骤2:通过消除最右边的N_{s}列[15]来缩短基图,缩短后基图的大小为$(m_{\mathrm{b}}-N_{\mathrm{s}})\times(n_{\mathrm{b}}-N_{\mathrm{s}})$。

步骤3:通过选择表1中的最小Z值来确定

Z,使 $k_b \times Z \geqslant K$。

步骤 4:根据修改后的基图,由右移的单位矩阵和零矩阵构成 PCM,如式(3)所述。

5G 新型无线电 LDPC 码的基图如图 1 所示。在 5G 新型无线电 QC-LDPC 码中,为了获得所需的信息块长度和高码率,对基图进行了缩短,剔除的列是扩展奇偶校验部分。

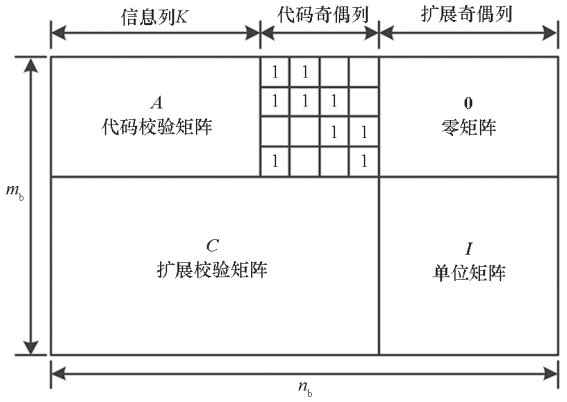

图 1　5G 新型无线电 LDPC 码的基图

Fig. 1　Base graph of LDPC code for 5G new radio

1.2　5G NR 标准 QC-LDPC 编码

给定 $\boldsymbol{H}$,LDPC 编码的目标是求解奇偶性方程

$$\boldsymbol{H}\boldsymbol{c}^{\mathrm{T}} = 0 \tag{5}$$

式中,$\boldsymbol{c}$ 是编码后的码字,由信息位和奇偶校验位组成。QC-LDPC 的编码方法包括高斯消除法、Richardson 等[16] 提出的 RU 方法以及 Nguyen 等[17] 提出的为 5G 无线电设计的高效编码方法。一旦 PCM 被确定,生成矩阵 $\boldsymbol{G}$ 可以从 $\boldsymbol{H}$ 中得到。

$$\boldsymbol{c} = \boldsymbol{u}\boldsymbol{G} \tag{6}$$

式中,$\boldsymbol{u}$ 是要编码的信息位数组。在 5G 无线电中,编码的 LDPC 码字在传输前会被打孔。前 $2 \times Z$ 位总是被打孔,而不是传输[18]。

2　QC-LDPC 的译码算法

LDPC 码可以采用置信传播的方法进行译码,其基本译码算法为和积算法[19](sum-product algorithm, SPA)。SPA 的近似算法是缩放最小和算法(min-sum algorithm, MSA)。由于 MSA 的复杂度比 SPA 低,所以更适合于实际实现。y_j 为接收值,v_j 为码字的第 j 位。对于 $1 \leqslant i \leqslant M$ 和 $1 \leqslant j \leqslant N$,假设变量节点 VN_j 到校验节点 CN_i(VN to CN, VTC)消息为 L_{ij},校验节点 CN_i 到变量节点 VN_j(CN to VN, CTV)消息为 R_{ij}。译码过程从 1 迭代到 M 层。根据每层的 Z 元素行来更新 $\boldsymbol{R}$ 和 $\boldsymbol{L}$。l 为消息矩阵的层索引,$l = 1, \cdots, m_b$。由于 $\boldsymbol{I}_s$ 是一个单位矩阵,每层内 VN_j 和其他 CN 之间只有一个联系,为了简化计算,L_{ij}^l 用 L_j^l 代替。L_j^l 的初始值为 VN_j 的后验概率(a posteriori probability, APP)比。译码的步骤描述如下。

步骤 1:初始化 APP 比和 CTV 消息。

$$L_j^0 = \ln \frac{P\{v_j = 0 \mid y_j\}}{P\{v_j = 1 \mid y_j\}} \qquad 1 \leqslant j \leqslant N \tag{7}$$

$$R_{ij} = 0, 1 \leqslant i \leqslant M \qquad 1 \leqslant j \leqslant N \tag{8}$$

步骤 2:按层计算 L_j^l 和 R_{ij}。首先,更新来自第 i 行所有 VN 的 L_j^l 值

$$L_j^{\mathrm{old},l} = L_j^{l-1} - R_{ij}^{\mathrm{old},l} \tag{9}$$

其次,更新 R_{ij}

$$R_{ij}^{\mathrm{new},l} = \alpha \prod_{k \in N(i) \setminus \{j\}} \mathrm{sign}(L_k^l) \min_{k \in N(i) \setminus \{j\}} \{ |L_k^l| \} \tag{10}$$

式中:$1 \leqslant i \leqslant M$;$k \in N(i) \setminus \{j\}$ 代表 CN_i 的 VN 邻居集,不包括 VN_j;α 是比例因子。最后,更新 L_j^l

$$L_j^{\mathrm{new},l} = L_j^{\mathrm{old},l} - R_{ij}^{\mathrm{new},l} \tag{11}$$

步骤 3:在计算了所有层的 CN 和所有 $1 \leqslant l \leqslant m_b$ 和 $1 \leqslant j \leqslant N$ 的 VN 后,信息位由最后一层的结果决定,其中,$l = m_b$,

$$v_j = \begin{cases} 0 & L_j^{m_b} \geqslant 0 \\ 1 & \text{其他} \end{cases} \tag{12}$$

最终得到译码结果 v。

3　基于 GPU 的高吞吐率 LDPC 译码技术

在本节中,描述了几种用于 5G 新型无线电的高吞吐率译码技术。重点介绍使用 $\boldsymbol{BG}_1$ 构建的 QC-LDPC 码的并行译码过程。译码算法 MSA 是一种分层译码算法,可以在 GPU 上进行并行高效译码。为了实现高吞吐率译码,有必要进一步优化译码过程。此外,高码率和大码长对 GPU 的内存带宽提出了挑战。一般来说,内存访问的开销会影响译码速度。在 5G 新型无线电中,PCM 的大小显著增加,对 GPU 的内存资源需求也相应增加。在译码时,合理分配 GPU 上的资源至关重要。

3.1　压缩数据结构

$\boldsymbol{H}$ 是一个稀疏矩阵。一般来说,利用压缩结构,而不是将整个 PCM 传输到 GPU。定位 $\boldsymbol{H}$ 中非零项的常用方法是在译码前生成一个查找表,可以在每一行中存储非零条目的位置。然而,$\boldsymbol{H}$ 中每一行的非零条目数量并不完全相同。如果大多数行中的非零条目很少,那么执行相同次数的迭代会造成时间浪费。因此,不仅要记录每行非

零位的位置,还要记录非零项的数量。

将 P_{ik} 表示为 $\boldsymbol{H}$ 中第 i 行中第 k 个非零条目的指数。因为在 $\boldsymbol{H}$ 中每行最多有 k_b 个非零条目,那么 $1 \leqslant k \leqslant k_b$。设 C_i 为第 i 行的非零条目数,则 CN 和 VN 之间的联系可由 P_{ik} 和 C_i 决定。压缩 $\boldsymbol{H}$ 的结构可以节省内存空间和访问时间。而且,压缩在 GPU 上很容易实现。

除了压缩 $\boldsymbol{H}$ 外,CTV 消息 $\boldsymbol{R}$ 也可以被压缩。如上所述,VN 和 CN 之间的连接在译码前是已知的。如果 VN_j 和 CN_i 之间存在连接的话,则只需要在第 i 行中保存 VN 的消息值。$\boldsymbol{R}$ 的大小可以从 $(m_b \times Z) \times (n_b \times Z)$ 压缩到 $(m_b \times Z) \times k_b$。注意到 $\boldsymbol{R}$ 的压缩也是 GPU 上内存凝聚的一种优化方法。CTV 消息是译码过程的中间结果。将 $\boldsymbol{R}$ 压缩到连续空间有助于减少 GPU 上线程束内的块冲突。

3.2 并行层译码

本文采用的 LDPC 译码算法是 MSA 算法,MSA 算法适合在 GPU 上进行并行译码。PCM 是使用右移单位矩阵构建的,对于 $\boldsymbol{H}$ 中的每个 $Z \times Z$ 块,所有非零条目都来自不同的列。当计算 CTV 和 VTC 消息从 $i \times Z$ 到 $(i+1) \times Z$ 层时,不存在依赖性,其中 $i = 0, 1, \cdots, m_b - 1$。所以可以通过固定的线程分配来实现译码计算。本文为 MSA 的分层译码分配 Z 线程。对于 5G 新型无线电来说,最大的 Z 为 384,不超过 GPU 上每个块的最大线程数。

如上所述,R_{ij} 和 L_j^l 是由式(10)和式(11)更新的。计算 $\min\limits_{k \in N(i) \backslash (j)} \{ |L_k^l| \}$ 的复杂度为 $O(d_i^2)$,其中 d_i 是第 i 行的非零元素数。本文采用文献[3]中提出的一种线性复杂性方法来计算 $|L_{ik}^l|$ 的最小值;第一个步骤是全局计算,计算出第一个和第二个最小值;第二个步骤是局部计算,根据全局计算中得到的值确定 $|L_{ik}^l|$ 的最小值。

将 f, s 表示为 $|L_{ik}^l|, k \in N(i)$ 的第一个和第二个的最小值。设 f 和 s 的初始值为正最大值。在所有 VN 中的第一个和第二个最小值 f^l 和 s^l 通过式(13)~(14)计算。

$$f^l = \begin{cases} |L_{ik}| & |L_{ik}| < f^{l-1} \\ f^{l-1} & \text{其他} \end{cases} \tag{13}$$

$$s^l = \begin{cases} |L_{ik}| & f^{l-1} < |L_{ik}| < s^{l-1} \\ f^{l-1} & |L_{ik}| < f^{l-1} \\ s^{l-1} & \text{其他} \end{cases} \tag{14}$$

那么,可以通过式(15)得到 VN_j 的最小值。

$$L_{\min} = \begin{cases} f^l & |L_{ik}| \neq f^l \\ s^l & \text{其他} \end{cases} \tag{15}$$

类似地,可以得到 $\boldsymbol{S}$ 的全局符号位 $S_{\text{global}} = \prod_{k \in N(i)} \text{sign}(L_{ik}^l)$ 和 $S = \prod_{k \in N(i) \backslash \{j\}} \text{sign}(L_{ik}^l)$。在全局中,通过式(16)计算 S_{global}。

$$S_{\text{global}} = \begin{cases} -\text{sign} & L_{ik} < 0 \\ \text{sign} & \text{其他} \end{cases} \tag{16}$$

然后通过式(17)得到 $\boldsymbol{S}$。

$$S = \begin{cases} -S_{\text{global}} & L_{ik} < 0 \\ S_{\text{global}} & \text{其他} \end{cases} \tag{17}$$

因为只有负值才会影响 S_{global} 的符号,所以只在 $L_{ik} < 0$ 时更新 S_{global}。同时,如果 $L_{ik} \geqslant 0, S = S_{\text{global}}$。基于此,乘法中的式(10)被式(16)~(17)所取代。这种替换减少了 GPU 上的计算量。

最终的并行层译码算法如算法 1 所示。算法的输入是由式(7)计算出的 L_j^0、非零条目的索引矩阵 $\boldsymbol{P}$ 和每层的非零条目数 C,$\boldsymbol{L}$ 是中间 LLR 的向量,$\boldsymbol{f}$ 和 $\boldsymbol{s}$ 是第一和第二最小值的向量。由于在

算法 1 并行层译码算法

Alg. 1 Parallel layered decoding algorithm

已知:L_j^0, $\boldsymbol{P}$, C

初始化,$\boldsymbol{R} \leftarrow \boldsymbol{0}$
$L[2Z, \cdots, m_b \times Z - 1] \leftarrow L_j^0[0, \cdots, (n_b - 2) \times Z]$
for $iter = 0 \rightarrow iter$ **do**
  for 层 $l = 1 \rightarrow m_b$ **do**
    for 线程 $t = 0 \rightarrow Z - 1$ **parallel do**
      $f^{l,t} \leftarrow \text{MAX}, s^{l,t} \leftarrow \text{MAX}$
      //全局轮
      for $j = 0 \rightarrow C^{l,t}$ **do**
        $k \leftarrow P(l, j)$
        $L(t, k) \leftarrow L(t, k) - R^l(t, k)$
        计算 $f^l(t), s^l(t)$
        更新 $S_{\text{global}}(t)$
      end for
      //局部轮
      for $j = 0 \rightarrow C^{l,t}$ **do**
        计算 $|L_{\min}|(t)$ 使用 $f^l(t), s^l(t)$
        更新 $S(t)$
        $k \leftarrow P(l, j)$
        $L(t, k) \leftarrow L(t, k) + \alpha \times |L_{\min}|(t) \times S(t)$
      end for
    end for
  end for
end for
//硬决策由 Z 线程并行执行

处理($n_b \times Z$) $\times Z$ 块中的条目消息时没有依赖性,$\boldsymbol{L}$ 的计算由译码器中的 Z 个线程并行处理,层的迭代次数为 m_b,开始层的迭代次数为 l_b。在第一次迭代中,所有 VN 的最小值都是 0,在本文实现中,起始层为 $l=1$,以减少译码时间。在并行译码器中,我们实现了两相法,简化了寻找最小值和符号乘积的过程。该方法消除了嵌套循环,大大提高了译码吞吐率。

3.3　多块译码

如上所述,每块有 Z 线程并行译码。但在 GPU 上,每块最大线程数为 1 024 或 2 048。虽然不建议超过 Z 的线程数,但最大 Z 是 384,远远没有超过每块线程的最大限制。因此,有可能在同一块内分配更多的线程来译码多个码字。因此,我们实现了多块译码,提高 GPU 的并行度。让 N_p 作为同一块中被译码的码字数量。那么每个块的线程数为 $N_p \times Z$。因此,用 N_c 表示输入码字的数量,为译码内核分配 N_c/N_p 块。有 N_c/N_p 块一起被译码,每个块由 $N_p \times Z$ 个线程进行译码。

3.4　存储层次

为了充分利用 GPU 上的内存资源,经常访问的值被存储在共享内存中。在本文实现中,$\boldsymbol{f}$、$\boldsymbol{s}$ 和 $\boldsymbol{L}$ 被存储到共享内存中。它们的大小分别是 $1 \times Z$、$1 \times Z$ 和 $Z \times n_b$。不建议将 $\boldsymbol{R}$ 放入共享内存中。因为 $\boldsymbol{R}$ 的大小是($m_b \times Z$) $\times k_b$,$\boldsymbol{BG}_1$ 中 $k_b=22$,最大 $m_b=46$。$\boldsymbol{R}$ 尺寸过大,以至于把它放在共享内存中可能会限制译码时常驻块的数量,所以本文将 $\boldsymbol{R}$ 存储在全局内存中。

$\boldsymbol{L}$ 的大小只有 $n_b \times Z$。本文只关心最后的结果,中间的结果会在后几层的迭代中被覆盖。由于译码时经常访问 $\boldsymbol{L}$,所以必须要将 $\boldsymbol{L}$ 放到离计算单元较近的存储层次中。在 5G 新型无线电中最大的 $n_b=68$,$Z_{max}=384$。$\boldsymbol{L}$ 的大小小于 GPU 共享存储的容量。因此将 $\boldsymbol{L}$ 的值存储到共享内存中,便于 GPU 的 CUDA 核心快速访问到 $\boldsymbol{L}$。

3.5　两级量化

以往工作很多都提出了量化方案来提高 LDPC 的译码吞吐率[20]。在本文实现中,为了更有效地利用带宽,同时减少对性能的损失,提出了一种两级量化方案。

一般来说,GPU 支持的内置类型的最小长度为 8 位。小于 8 位的量化方案不适合基于 GPU 的高吞吐率译码。因为较小的长度可能会引入许多类型转换,这些类型转换是为了适应 CUDA 中原生算术指令支持的最小长度而执行的。与加法和乘法等其他算术运算相比,类型转换在 GPU 上的吞吐率较低。此外,像 AND、XOR 以及移位等位元运算吞吐率也很低。

因此,本文在译码过程中使用 8 位值来节省带宽。虽然小于 8 位的数据类型的计算效率不高,但通过数据打包可以提高 CPU 和 GPU 之间的传输效率。基于此,本文将 L_j^0 量化为第一级的 4 位数。而 $\boldsymbol{L}$ 和 $\boldsymbol{R}$ 则在第二级量化为 8 位数。在两个层次中,第一个位是符号位。

第一级量化、4 位 L_j^0 方案进一步节省了片外带宽。可以降低 CPU 和 GPU 之间的通信延迟。

4 位 L_j^0 方案避免了计算 $\boldsymbol{L}$ 时的溢出。如同在式(11)中所描述的,$\boldsymbol{L}$ 在每次迭代中与 $\boldsymbol{R}$ 相加。在多次迭代后,$\boldsymbol{L}$ 的数量有可能会溢出。而本文实验结果也证明了,当采用 8 位数据类型译码时,溢出现象很普遍。因此,我们将 L_j^0 量化为 4 位来限制初始值的范围。这个方案可以避免大部分计算过程中的溢出。

$\boldsymbol{L}$ 和 $\boldsymbol{R}$ 的两级量化可以节省片上带宽。为了实现高吞吐率译码,片上带宽也是需要考虑的重要因素。

两级量化也节省了共享内存和寄存器的空间。驻留块受共享内存占用率的限制。节省共享内存和寄存器有助于实现更多的并行性。与 8 位方案相比,两级方案的性能损失可以忽略。

3.6　数据打包

数据打包可以减少 CPU 和 GPU 之间数据传输造成的译码延迟。译码器的输出是二进制比特位。虽然在 GPU 上实现比特运算效率不高,但不难发现,我们可以对译码结果进行打包来节省带宽。在本文实现中,每 8 个译码位在硬性决定后打包成一个 8 位 char 值。然后将打包后的数据传输给 CPU。

除了对译码位进行打包外,本文还结合量化和数据打包来进一步节省带宽。如上所述,输入的 L_j^0 值被量化为 4 位固定数。因此,每两个 L_j^0 的值可以打包成一个 8 位的 char 值。一旦打包后的数据被传送到 GPU,它们就会被转换为 8 位的 char 值,以便进行下一个译码过程。

图 2 显示了译码器在 GPU 上的内存层次结构和数据打包策略。在译码前,输入数据被量化为 4 位值,中间值为 8 位值。$\boldsymbol{L}$、$\boldsymbol{f}$ 和 $\boldsymbol{s}$ 存储在共享内存中;$\boldsymbol{R}$ 存储在全局存储器中。8 个译码后的信息位被打包成一个 char 类型的值。然后将打包后的数据传送给 CPU。

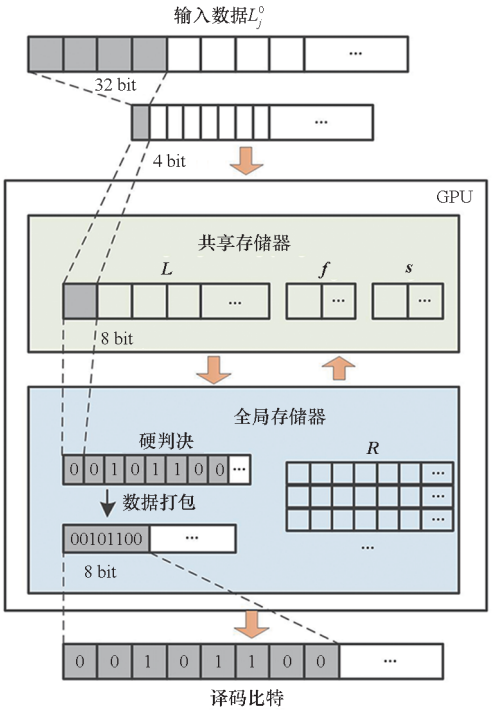

图 2 内存层次和数据打包

Fig. 2 Memory hierarchy and data package

4 实验结果

4.1 实验设置

本文在基于图灵架构的 Nvidia RTX 2080Ti 上实现了 QC-LDPC 译码器,该译码器拥有 4 352 个 CUDA 核心、68 个多处理器和 11 GB 的 GDDR6 内存,CPU 为 Inteli7-8700K,CUDA 版本为 10.2。

在生成编码值时,采用了正交相移键控调制方式,并用加法白高斯噪声作为信道噪声。MSA 的尺度因子为 0.75,信噪比为 3 dB,$d=0.15$。在测量译码吞吐率时,不采用提前终止法,所以上述参数设置为常用值。

4.2 译码吞吐率

表 2 描述了不同码率下的基图结构。LDPC 基图的码长、码率、右移参数配置会根据码率变化而变化。本文旨在提高由 $\boldsymbol{BG}_1$ 构建的 QC-LDPC 码的译码吞吐率,基码长度为 2 048。如表 2 所示,右移尺度随着码率的增加而增加。由于 $\boldsymbol{H}_b$ 的缩短,m_b 和 n_b 减小。吞吐率 T 定义为 $T=\frac{N_c}{t}$。其中,t 为译码时间。

采用不同优化方法的译码器的吞吐率结果如表 3 所示。采用两级量化的量化方式,吞吐率提升了 2.1 倍。采用多码块译码方法,吞吐率进一步提升了 18.6%。此外,数据打包方案也有很好的效果,与不进行数据打包的方法相比,吞吐率获得了 16.9% 的提高。

表 2 不同码率的基图

Tab. 2 Base graph under different code ratio

r	(N,K)	$m_b \times n_b$	Z
1/3	(2 112,704)	46 × 68	32
1/2	(2 160,1 056)	25 × 47	48
2/3	(2 112,1 408)	13 × 35	64
3/4	(2 088,1 584)	9 × 31	72
5/6	(2 080,1 760)	6 × 28	80

表 3 不同优化方法下吞吐率性能对比

Tab. 3 Comparison of throughput under different optimization methods

方法	吞吐率/(Mbit/s)	加速比
浮点	267	1.0
两级量化	557	2.1
两级量化 + 多码块译码	661	3.1
两级量化 + 多码块译码 + 数据打包	773	3.3

本文同时测量了表 2 中不同码率的译码吞吐率。对于高码率,$\boldsymbol{H}$ 会被缩短。如表 2 所示,m_b 和 n_b 的大小随着码率的增加而减小。如此,迭代次数在水平和垂直方向上都相应减少,这就造成并行译码器的译码时间减少,而吞吐率显著增加。图 3 描述了不同码率下的译码吞吐率。多码率译码的吞吐率比普通译码的吞吐率平均提高了 40% 以上。

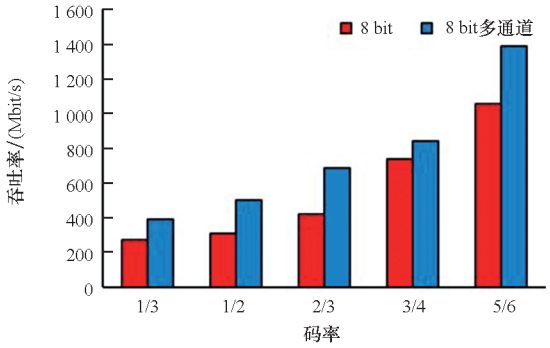

图 3 不同码率下的译码吞吐率

Fig. 3 Decoding throughput under different code ratio

4.3 线程配置分析

为了最有效地利用 GPU 上的资源,线程的分配是根据码率来调整的。众所周知,每个线程块

和每个多处理器的寄存器和共享内存的数量是有限的。此外,为了避免过载,还要考虑每个流多处理器(streaming multiprocessor, SM)的最大常驻块和线程数。本文将 N_p、N_r、N_m、N_t 表示为每个块的码字数、寄存器数、共享内存数和线程数,并用 R_{SM}、M_{SM}、T_{SM}、B_{SM} 表示每个 SM 的最大寄存器、共享内存、常驻线程和常驻块,N_{SM} 为 GPU 上 SM 的数量,N_b 为每个 SM 上的常驻块。

那么就可以得到相应的线程分配如下

$$N_b N_p \leqslant \min\left(\frac{R_{SM}}{N_r}, \frac{M_{SM}}{N_m}, \frac{T_{SM}}{N_t}, B_{SM}\right) \tag{18}$$

同时被译码的码块数量可以用以下方法来评估

$$N_c \leqslant N_b N_p N_{SM} \tag{19}$$

在此基础上,评估最合适的线程分配和被并行译码的码字块数量。设最优的线程数配置 $X = \min\left(\frac{R_{SM}}{N_r}, \frac{M_{SM}}{N_m}, \frac{T_{SM}}{N_t}, B_{SM}\right)$,译码的最佳码字块数量 $N_c^t = N_b N_p N_{SM}$。对于 RTX 2080Ti,实验中参数具体设置如表 4 所示。N_r 由 nvcc 编译器获得。输出数据 $L_o = Z \times k_b$,会根据 Z 和 N_b 大小来变化。

表 4　GPU 上资源的占用情况

Tab. 4　Resource occupancy on GPU

参数	数值
X	$\min\left(\frac{R_{SM}}{N_r}, \frac{M_{SM}}{N_m}, \frac{T_{SM}}{N_t}, B_{SM}\right)$
B_{SM}	16
N_{SM}	68
N_r	63
N_t	Z
N_m	$(N_b+2) \times Z$

表 5 描述了 RTX 2080Ti 上实现的线程参数配置情况下的吞吐率情况。为了实现更高的吞吐率,本文选择了一个合适的 N_p 参数来实现。

表 5　最佳线程分配和吞吐率

Tab. 5　The Optimal thread allocation and throughput

r	X	N_c^t	N_c	N_p	吞吐率/(Mbit/s)
1/3	16	1 088	1 000	5	389
1/2	16	1 088	1 000	5	506
2/3	16	1 088	1 000	5	684
3/4	14	952	1 000	5	838
5/6	12	816	800	2	1 385

图 4 显示了每个 SM 中的最大常驻块与 GPU 上的码率的变化关系。M、R、T、B 分别代表共享内存、寄存器、常驻线程和每个 SM 的常驻块。在低码率情况下,Z 很小,限制了每个 SM 的最大常驻块。在高码率情况下,Z 较大,N_p 和 N_c 受到每个块的最大线程量和最大共享内存量的限制。最终的常驻块是 M、R、T、B 四个数值的最小值。

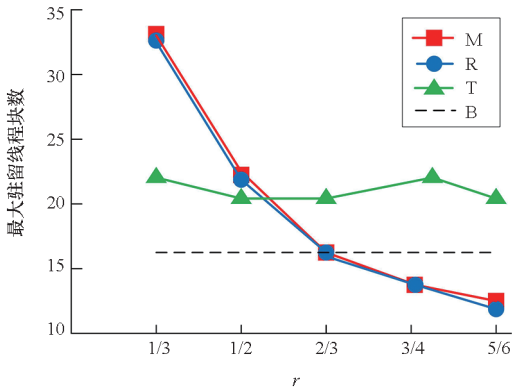

图 4　每个 SM 中的最大常驻块与 r 的变化关系

Fig. 4　Changing relationship between the maximum resident block in each SM and r

4.4　性能比较

目前,基于 GPU 的 5G 新型无线电的 QC-LDPC 译码器很少。在 5G 新型无线电中,信息列的数量 k_b 较多,达到 22 个,而 WiMAX 中最大的非零条目是 7 个。表 6 为本文 LDPC 译码器和其他 LDPC 译码器的比较。其中,迭代次数为 10 次,译码算法为 MSA 的分层译码,表中的吞吐率用 Mbit/s 表示。虽然我们在译码时需要处理更多的迭代,但本文提出的译码器在高码率下实现了 Gbit/s 的吞吐率,超过其他相关工作。另外本文利用吞吐率/峰值性能的比值来衡量同样的峰值性能情况下实现的吞吐率,以此来比较不同的工作对 GPU 峰值性能的利用率。从最终结果可以看出,本文算法峰值性能利用率也是最高的,证明了本文 GPU 并行算法及优化方法的优越性。

表 6　不同 GPU 的译码器性能比较

Tab. 6　Performance comparison of different GPU-based decoder

工作	标准	GPU	吞吐率/(Mbit/s)	峰值性能/TFLOPS	利用率
文献[3]	WiMAX	GTX580	620	6.5	95.4
文献[12]	WiMAX	Titan	316	12.1	26.1
文献[4]	WiFi	Titan X	913	12.1	75.4
文献[5]	WiFi	1080Ti	971	10.8	89.9
本文	5G NR	2080Ti	1 385	14.2	97.3

5 结论

本文提出了一种基于GPU的5G新型无线电的LDPC译码器。同时提出了基于GPU的并行MSA算法。在实现细节上,本文利用多块译码方法实现了码块内的并行译码,提出了两级量化方案,以进一步节省GPU的片上和片下带宽。通过评估目标GPU的最佳线程分配,最终实现了基于GPU的高吞吐率5G新型无线电LDPC译码器。在(2 080,1 760),$r=5/6$的配置下,本文提出的译码器吞吐率最高可达到1.38 Gbit/s。

参考文献(References)

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

[2] LI R C, DOU Y, ZOU D, et al. Efficient graphics processing unit based layered decoders for quasicyclic low-density parity-check codes[J]. Concurrency and Computation: Practice and Experience, 2015, 27(1): 29-46.

[3] MHASKE S, KEE H, LY T, et al. High-throughput FPGA-based QC-LDPC decoder architecture[C]//Proceedings of the IEEE 82nd Vehicular Technology Conference, 2015.

[4] KESKIN S, KOCAK T. GPU-based gigabit LDPC decoder[J]. IEEE Communications Letters, 2017, 21(8): 1703-1706.

[5] YUAN J Y, SHA J. 4.7-Gb/s LDPC decoder on GPU[J]. IEEE Communications Letters, 2018, 22(3): 478-481.

[6] 3GPP. Release 15[EB/OL]. (2019-04-26)[2022-04-01]. https://www.3gpp.org/specifications-technologies/releases/release-15.

[7] FERRAZ O, SUBRAMANIYAN S, WANG G H, et al. Gbit/s non-binary LDPC decoders: high-throughput using high-level specifications[C]//Proceedings of the IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines, 2020.

[8] HERRMANN M, WEHN N, THALMAIER M, et al. A 336 Gbit/s full-parallel window decoder for spatially coupled LDPC codes[C]//Proceedings of Joint European Conference on Networks and Communications & 6G Summit, 2021.

[9] GORIUSHKIN R, NIKISHKIN P, LIKHOBABIN E, et al. FPGA implementation of LDPC decoder architecture for wireless communication standards[C]//Proceedings of the 10th International Conference on Modern Circuits and Systems Technologies, 2021.

[10] 谢天娇, 李波, 杨懋, 等. 高速码率兼容DVB-S2的LDPC译码器的FPGA实现[J]. 西北工业大学学报, 2019, 37(2): 299-307.
XIE T J, LI B, YANG M, et al. LDPC decoder of high speed multi-rate DVB-S2 based on FPGA[J]. Journal of Northwestern Polytechnical University, 2019, 37(2): 299-307. (in Chinese)

[11] WANG Y, WANG Q L, ZHANG Y, et al. An area-efficient hybrid polar decoder with pipelined architecture[J]. IEEE Access, 2020, 8: 68068-68082.

[12] FALCÃO G, SILVA V, SOUSA L. How GPUs can outperform ASICs for fast LDPC decoding[C]//Proceedings of the 23rd International Conference on Supercomputing, 2009.

[13] WANG G H, WU M, YIN B, et al. High throughput low latency LDPC decoding on GPU for SDR systems[C]//Proceedings of the IEEE Global Conference on Signal and Information Processing, 2013.

[14] TANNER R. A recursive approach to low complexity codes[J]. IEEE Transactions on Information Theory, 1981, 27(5): 533-547.

[15] RICHARDSON T, KUDEKAR S. Design of low-density parity check codes for 5G new radio[J]. IEEE Communications Magazine, 2018, 56(3): 28-34.

[16] RICHARDSON T J, URBANKE R L. Efficient encoding of low-density parity-check codes[J]. IEEE Transactions on Information Theory, 2001, 47(2): 638-656.

[17] NGUYEN T T B, TAN T N, LEE H. Efficient QC-LDPC encoder for 5G new radio[J]. Electronics, 2019, 8(6): 668.

[18] BAE J H, ABOTABL A, LIN H P, et al. An overview of channel coding for 5G NR cellular communications[J]. APSIPA Transactions on Signal and Information Processing, 2019, 8: e17.

[19] KSCHISCHANG F R, FREY B J, LOELIGER H A. Factor graphs and the sum-product algorithm[J]. IEEE Transactions on Information Theory, 2001, 47(2): 498-519.

[20] CHEN J, DHOLAKIA A, ELEFTHERIOU E, et al. Reduced-complexity decoding of LDPC codes[J]. IEEE Transactions on Communications, 2005, 53(8): 1288-1299.

(编辑:王颖娟,杨琴)