文章编号：2095-4980(2014)06-0837-05

一种低存储量需求的高效 LDPC 译码方法

赵　岭[1,2]，李　众[1]，朱曼洁[2]，王荣博[2]

(1.北京航空航天大学 电子信息工程学院，北京 100191；2.中国航天科技集团公司 卫星应用研究院，北京 100086)

摘　要：提出了一种高效的低密度奇偶校验码(LDPC)译码方法，相对于传统译码方法，仅需增加少量存储量便能获得接近 2 倍的吞吐率增益。本文将修正的最小和算法与分组双向消息传递(STMP)译码算法相结合，提出了基于最小和的分组双向消息传递算法(MS-STMP)，并给出了相应的迭代交叠方案。随后讨论了交叠过程中 2 种运算单元对存储器的访问。最后以中国地面数字电视传输(DTTB)标准中使用的一组 LDPC 码为例，计算了 MS-STMP 算法相对于传统双向消息传递(TPMP)算法的吞吐率增益和额外增加的存储量。计算结果表明，MS-STMP 算法平均增加 44%的存储量，而将吞吐率平均提高到 1.85 倍，相对于交叠(OMP)算法有明显的优势。

关键词：低密度奇偶校验码；译码方法；存储量；交叠

中图分类号：TN911.22　　**文献标识码：**A　　**doi：**10.11805/TKYDA201406.0837

An efficient decoding of LDPC codes with low memory requirement

ZHAO Ling[1,2]，LI Zhong[1]，ZHU Man-jie[2]，WANG Rong-bo[2]

(1.Electronic and Information Engineering School，Beihang University，Beijing 100191，China；
2.The Academy of Satellite Application, Beijing 100086，China)

Abstract: An efficient decoding with low memory requirement is presented in this paper. With some extra memory bits，the proposed method could achieve nearly 2 times throughput compared with the conventional decoding. This paper combines min-sum decoding with Sliced Two-phase Message Passing (STMP) decoding, and proposes an overlapping scheme. The accessing of the operation units to the memory units is also discussed in order to analyze the access violation. A group of Low Density Parity Check(LDPC) codes are selected from Chinese Digital Television Terrestrial Multimedia Broadcasting(DTTB) standard，and the throughput gain as well as the ratio of extra memory bits for the proposed decoding over the conventional Two Phase Message Passing(TPMP) are calculated. The calculated results indicate that MS-STMP decoding needs only extra 44% memory bits to achieve 1.85 times throughput in even，which is more efficient than the Overlapped Message Passing(OMP) decoding.

Key words: Low Density Parity Check codes；decoding；memory bits；overlapping

低密度奇偶校验码(LDPC)[1]由于其优异的纠错性能和可并行译码的特性近年来受到业界的广泛关注[2]，并已经应用于实际。传统的 Log-BP 译码算法主要包含 2 种运算：变量节点更新运算和校验节点更新运算，这 2 种运算交替进行。这类 2 种运算交替进行的算法可以归结为双向消息传递译码(TPMP)算法。TPMP 算法也可以用最小和(Min Sum，MS)算法实现，从而降低复杂度。传统的 TPMP 算法 2 种更新运算的结果相互依赖，因此硬件资源利用率只有 50%[3]，即变量节点更新单元工作时，校验节点更新单元不工作，否则反之。为提高 TPMP 算法的硬件资源效率，文献[4–5]提出了一种交叠译码(OMP)算法及其改进算法。在 OMP 算法中，通过改变矩阵中各行与各列的运算顺序，变量节点更新运算和校验节点更新运算可以部分交叠运行，从而提高了硬件资源效率，提高了译码吞吐率。但是，在交叠区变量节点更新单元和校验节点更新单元需要同时对迭代信息存储器进行访问，即同一时间对同一个存储器需要 2 次读操作和 2 次写操作，这就是 OMP 的存储器访问冲突问题。为了解决访问冲突，

收稿日期：2013-11-22；**修回日期：**2013-12-12

基金项目：国家自然科学基金资助项目(61401010)；中国航天科技集团公司卫星应用研究院开放基金资助项目(2012-1696)；中央高校基本科研业务费专项资金资助项目(YWF13TRSC062)

通常需要额外增加一倍的迭代信息存储器，使变量节点更新单元和校验节点更新单元对迭代信息存储器进行交替访问。文献[6]设计了一种高并行度的部分并行译码器结构，并采用 OMP 的思想使得 2 种迭代运算交叠进行，在高并行度下取得高的硬件资源利用率。文献[6]所提方法额外增加了一倍的迭代存储量，且需要使用双倍时钟对迭代存储器进行访问来解决访问冲突。文献[7]提出将 2 帧 LDPC 码同时译码，变量节点更新单元与校验节点更新单元同时工作，这种做法同样提高了译码器吞吐率，但需要多耗费一倍的存储量去存储另一帧码字的初始化信息、迭代信息和判决信息。文献[8]提出了一种 STMP 译码算法，将校验矩阵按照行块进行分层，各层的迭代译码运算可以交叠进行且没有访问冲突。但是文献[8]所提的 STMP 译码算法使用的是 log-BP 算法，需要耗费小块存储器实现迭代更新运算过程中的查找表。本文通过修改迭代译码算法运算流程，将修正的最小和算法与 STMP 算法结合，提出了基于修正最小和的 STMP(MS-STMP)算法，降低了实现复杂度。文章以中国 DTTB 标准中使用的 LDPC 码为例，计算了 MS-STMP 算法相对于传统 TPMP 算法的吞吐率增益和额外增加的存储量，计算结果表明，针对 DTTB 标准中的 3 种码率的码字，MS-STMP 算法平均增加 44%的存储量，而将吞吐率平均提高到 1.85 倍，相对于 OMP 算法具有明显的优势。

1 基于修正的最小和 STMP 算法及其交叠方案

1.1 译码算法的描述

与文献[8]中基于 log-BP 的 STMP 算法类似，本文所提的 MS-STMP 算法也需要对校验矩阵按照行进行分层，然后针对分层进行迭代译码。分层的大小通常设置为子矩阵的大小，因此分层数目通常为校验矩阵的行块数，为便于描述，此处将分层数目定义为 L。另外，定义 R_{mj} 为与第 m 个校验节点第 j 个变量节点相关联的迭代信息，定义 R_{mj}^{k} 为第 k 次迭代时的 R_{mj} 信息，$R_{mj}^{k,t}$ 表示第 k 次迭代第 t 层对应的 R_{mj} 信息。定义 S_j 表示与第 j 个变量节点相关联的所有 R_{mj} 与 I_j 的求和结果，定义 $S_j^{k,t}$ 表示第 k 次迭代前 t 层与第 j 个变量节点相关联的所有 R_{mj} 与 I_j 的求和结果。那么 MS-STMP 算法可以表述如下：

1) 行处理过程(Row Processing Stage，RPS)，由行处理单元(Row Processing Unit，RPU)完成。在第 $k(k=1,2,\cdots,N)$ 次迭代时，对于每一个校验节点 m，计算 R_{mj} 如下：

$$L_{mj}^{k} = S_j^{k-1,L} - R_{mj}^{k-1} \tag{1}$$

$$R_{mj}^{k} = a \times \prod_{n\in N(m)/j} sign\left(L_{mn}^{k}\right) \times \min_{n\in N(m)/j}\left|L_{mn}^{k}\right| \tag{2}$$

式中：$N(m)$表示与校验节点 m 相关联的所有变量节点的集合；a 为最小和算法的修正系数。

当 k 等于 1，即第 1 次迭代时，将 L_{mj} 设置为 I_j。

同时，RPU 根据 S_j 的符号位对每一个校验节点计算校验关系，当判决结果 $\boldsymbol{X}$ 能满足 $\boldsymbol{H}^{\mathrm{T}}\cdot\boldsymbol{X}=0$ 或者达到预设的最大迭代次数时，迭代停止。

2) 列处理过程(Column Processing Stage，CPS)，由列处理单元(Column Processing Unit，CPU)完成。在第 k 次迭代对第 $t(t=1,2,\cdots,L)$层进行运算时，对于每一个变量节点 j，计算 $S_j^{k,t}$ 如下：

$$S_j^{k,t} = S_j^{k,t-1} + \sum_{m\in M(j)} R_{mj}^{k,t} \tag{3}$$

式中当 $S_j^{k,t-1}$ 不存在，即当前列是首次进行列更新运算时，将 $S_j^{k,t-1}$ 由初始化信息 I_j 替代。在每一次迭代的最后，译码器通过 S_j 进行判决。

步骤 1)与步骤 2)反复进行，直到满足 $\boldsymbol{H}^{\mathrm{T}}\cdot\boldsymbol{X}=0$ 或者达到预设的最大迭代次数。

相对于传统的 TPMP 算法[3]，MS-STMP 只是将列更新过程中的减法放到行更新过程中，并将列更新过程中的求和运算拆成多次完成，因此，MS-STMP 相对于传统 TPMP 不会有抗误码性能的损失。

1.2 迭代译码的交叠方案

从式(1)~式(2)可以看出，在第 k 次迭代期间，RPU 读取由 RPU 在 k–1 次迭代过程中产生的 R_{mj} 信息，与由 CPU 在 k–1 次迭代过程中得到的 S_j 一起进行计算，然后更新 R_{mj}。因此，在一次迭代中，RPS 过程可以逐层进行，而不需要使用 CPU 在本次迭代中的计算结果。

另一方面，根据式(3)，CPU 读取由 RPU 在本次迭代中更新的 R_{mj}，然后将它们一层一层累加。因此，只要第 t 层的 RPS 运算完成即可开始第 t 层的 CPS。

根据上述分析可知，在一次迭代过程中，R_{mj} 信息单向地由 RPU 传递给 CPU，而累加和结果 S_j 也是单向地在每一次迭代的最后由 CPU 传递给 RPU，因此，MS-STMP 算法打破了传统 TPMP 译码算法中 2 种更新运算之间的顺序约束。当第 t 层的行处理过程完成后就可以开始第 t 层的列处理过程，第 t 层的列处理过程可以和第 t+1 层的行处理过程交叠进行。并且，由于 MS-STMP 算法中各层的更新运算相对独立，相邻层的水平(垂直)过程可以流水操作，不需要额外的等待时间。

以运算并行度为 1 为例，MS-STMP 算法的交叠方案可以用图 1 中的时序图表示。

其中，图 1 中 b 表示校验矩阵子矩阵大小，P_1 表示行运算单元插入的流水线级数。从图 1 可以看出，MS-STMP 算法一次迭代的时间是 $(L+1)b+P_1$ 个时钟周期，如果提高运算并行度，假定运算并行度为 c，则一次迭代的时间是 $(L+1)b/c+P_1$ 个时钟周期。若 $c=L$，即译码器设置 L 个行运算单元，此时与传统 TPMP 算法的部分并行结构并行度相当，而部分并行结构一次迭代的时间为 $2b+P_1$，因此，MS-STMP 算法相对于传统 TPMP 算法的吞吐率增益可近似表示为：

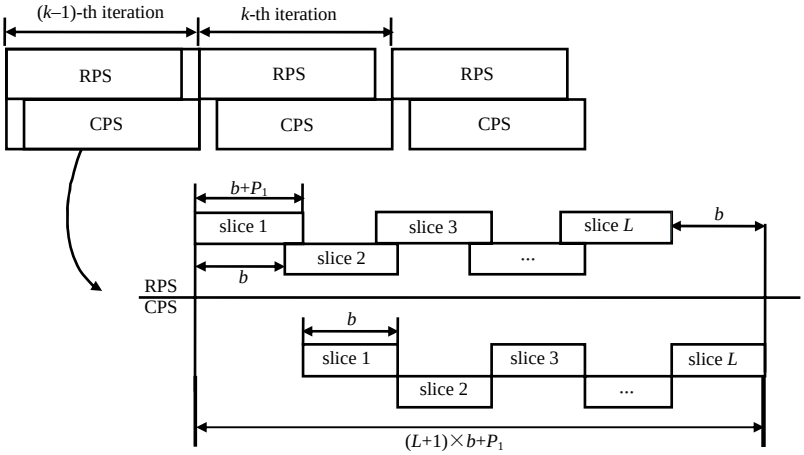

Fig.1 Overlapping scheme of MS-STMP decoding
图 1 MS-STMP 算法交叠方案时序图

$$Speedup = \frac{2b + P_1}{b + b / L + P_1} \tag{4}$$

从式(4)可以看出，在运算并行度相当的情况下，校验矩阵的行块数越多，MS-STMP 算法相对于传统 TPMP 算法的吞吐率增益越大，在 $b>>P_1$ 时，吞吐率增益近似为 $2L/(L+1)$，因此，当 $L>10$ 时，吞吐率增益便能超过 1.8，当 $L>30$ 时，吞吐率增益便能超过 1.93。而且，MS-STMP 算法的吞吐率增益与校验矩阵子矩阵的结构无关，这与 OMP 算法有本质的区别。

1.3 交叠方案中运算单元对存储器的访问

传统 TPMP 算法中，2 种更新运算的结果相互依赖，所以 2 种更新运算需要交替进行，导致了硬件资源利用率的低下。而 OMP 算法虽然通过变换 2 种更新运算对于校验矩阵的执行顺序，从理论上使得 2 种更新运算可以部分交叠进行，但具体实现时，在交叠部分行运算单元与列运算单元需要同时对迭代存储器进行读操作与写操作，造成访问冲突。解决此冲突现有 2 种方式，增加一倍的迭代存储器与使用双倍时钟访问迭代存储器[6]，两者是等价的。从图 1 中可以看出，MS-STMP 的交叠方案中有 2 种交叠，即列更新运算仅与下一层的行更新运算交叠，以及列更新运算与下 2 层的行更新运算交叠。这 2 种情况下，运算单元对存储器的访问情况如图 2 所示。

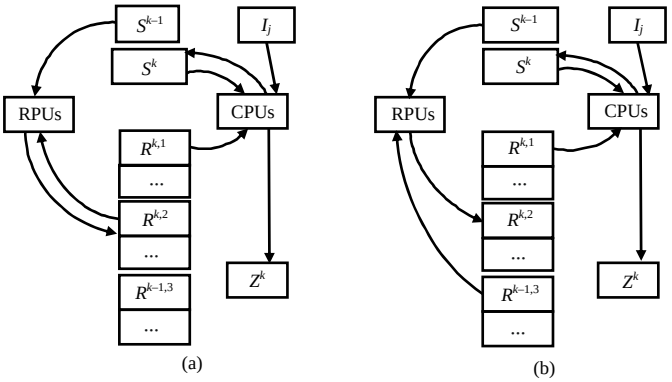

Fig.2 Access of RPUs and CPUs to the memories
图 2 RPU 与 CPU 对存储器的访问过程

如图 1 所示，在 CPS 过程中，前 $b–P_1$ 个时钟周期，CPS 仅与下一层的 RPS 交叠，这段时期，RPU 读取下一层的 R_{mj} 与对应的 S_j，更新下一层的 R_{mj}，而 CPU 读取本层的 R_{mj} 与上一层的 S_j 或者 I_j 更新对应的 S_j，这段时期

RPU 与 CPU 对存储器的访问如图 2(a)所示。

在剩下的 P_1 个时钟周期里，CPS 与下 2 层的 RPS 交叠，这段时期，RPU 读取第 3 层的 R_{mj} 与对应的 S_j，并更新第 2 层的 R_{mj}，而 CPU 读取本层的 R_{mj} 与上一层的 S_j 或者 I_j 更新对应的 S_j，这段时期 RPU 与 CPU 对存储器的访问如图 2(b)所示。

由图 2 可以看出，只要将相邻 3 层的迭代信息分开存储，以及将前后 2 次迭代的求和信息分开存储，每个存储器最多同时有 1 次读操作与 1 次写操作，因此迭代存储器可以使用准双口 RAM 实现，没有访问冲突。

从图 2 还可以看出，相对于传统的 TPMP 算法，MS-STMP 算法需要增加存储量存储相邻 2 次迭代的求和信息 S_j。如果 I_j 和 R_{mj} 的量化比特数为 f，那么在实际应用中，求和信息 S_j 需要使用(f+2)比特，所以，MS-STMP 算法所耗费的存储量可以计算如下：

$$M = M_{\text{初始化}} + M_{\text{求和}} + M_{\text{迭代}} + M_{\text{判决}} = qbf + 2qb(f+2) + Wbf + qb \tag{5}$$

式中：q 为校验矩阵的列块数目；W 为校验矩阵总重量。相对于传统 TPMP 算法，MS-STMP 算法增加的存储量的百分比可计算如下：

$$M_{\text{百分比}} = \frac{M_{\text{求和}}}{M_{\text{初始化}} + M_{\text{迭代}} + M_{\text{判决}}} = \frac{2qb(f+2)}{qbf + Wbf + qb} = \frac{2q(f+2)}{qf + Wf + q} \tag{6}$$

另一方面，对于 OMP 算法[5−6]，需要增加 1 倍的迭代信息存储器避免访问冲突，因此相对于传统 TPMP 算法，OMP 算法增加的存储量的百分比可计算如下：

$$M_{\text{百分比}} = \frac{M_{\text{迭代}}}{M_{\text{初始化}} + M_{\text{迭代}} + M_{\text{判决}}} = \frac{Wbf}{qbf + Wbf + qb} = \frac{Wf}{qf + Wf + q} \tag{7}$$

对于最小和算法，量化比特数 f 通常取为 6[6]，此时，根据式(6)和(7)，可以推导出只要 W 大于 2.7 倍 q，MS-STMP 算法消耗的存储量便少于 OMP 算法，且 W 与 q 相差越大，MS-STMP 算法的优势越明显。

2 MS-STMP 算法与现有算法的对比

为了验证 MS-STMP 算法的有效性，本文选用了中国 DTTB 标准中的 3 种码率的 LDPC 码[9]，这些码字的基本信息列在表 1 中。根据式(4)、(6)~(7)可以分别计算出在并行度相当的前提下，MS-STMP 算法相对于传统 TPMP 算法的吞吐率增益、存储量增加的百分比，以及 OMP 算法相对于传统 TPMP 算法存储量增加的百分比，上述计算结果见表 1。

表 1 MS-STMP 算法与现有算法性能对比

Table1 Comparisons between MS-STMP,TPMP and OMP

	code	p	q	b	W	P_1	speedup	M/%	M/%(OMP)
1	(7 493,3 048)	35	59	127	275	5	1.904	45.8	80.0
2	(7 493,4 512)	23	59	127	296	6	1.871	43.1	81.1
3	(7 493,6 096)	11	59	127	294	7	1.788	43.4	81.0

表 1 中第 7 列数据为 MS-STMP 算法相对于传统 TPMP 算法的译码速率增益，第 8 列数据为 MS-STMP 算法相对于传统 TPMP 算法存储量增加的百分比，第 9 列数据为 OMP 算法相对于传统 TPMP 算法存储量增加的百分比。

从表 1 中可以看出，相对于传统 TPMP 算法，MS-STMP 算法以增加不到 50%的存储量提高了近 1 倍的吞吐率，远远小于 OMP 算法所增加的约 80%的存储量。而且，MS-STMP 算法的吞吐率增益基本只与校验矩阵的行块数有关，与校验矩阵的子矩阵的移位偏移量无关，因此，MS-STMP 算法相对于 OMP 算法更为简单、灵活。

3 结论

本文提出了一种基于修正最小和的 LDPC 码译码方法——MS-STMP 算法，该算法仅需增加少量存储量便能实现迭代译码 2 种更新运算的高度交叠，并且没有存储器访问冲突，从而提高译码吞吐率。本文以中国 DTTB 使用的 3 种 LDPC 码为例，将 MS-STMP 算法与传统 TPMP 算法和 OMP 算法进行了比较，比较结果表明，在并行度相当的情况下，通过交叠，MS-STMP 算法能将译码吞吐率提高近 1 倍，而存储量增加不到 50%，相对于 OMP 算法需要增加超过 80%的存储量具有明显的优势。

参考文献:

[1] Gallager R G. Low density parity check codes[J]. IEEE Trans. Information Theory, 1962,8(1):21–28.

[2] 周昱,刘荣科,侯毅. 一种提高 LDPC 译码层内并行度的方法[J]. 太赫兹科学与电子信息学报, 2012,10(6):719–724. (ZHOU Yu,LIU Rong-ke,HOU Yi. An approach to improve parallelism inside layer in LDPC decoding[J]. Journal of Terahertz Science and Electronic Information Technology, 2012,10(6):719–724).

[3] ZHANG T,Parhi K K. VLSI implementation oriented(3,k)-regular low-density parity-check codes[C]// IEEE Proc. of SIPS. [S.l.]:IEEE, 2001:25–36.

[4] CHEN Y,Parhi K K. Overlapped message passing for quasi-cyclic low-density parity check codes[J]. IEEE Trans. Circuits and Systems, 2004,51(6):1106–1113.

[5] DAI Y M,YAN Z Y,CHEN N. Optimal overlapped message passing decoding of quasi-cyclic LDPC codes[J]. IEEE Trans. VLSI, 2008,16(5):565–578.

[6] CHEN X,KANG J,LIN S,et al. Memory system optimization for FPGA-based implementation of quasi-cyclic LDPC codes decoders[J]. IEEE Trans. Circuits and Systems, 2011,58(1):98–111.

[7] Darabiha A,Carusone C,Kschischang F R. Block-interlaced LDPC decoders with reduced interconnect complexity[J]. IEEE Trans. Circuits and SystemsII, 2008,55(1):74–78.

[8] ZHAO L,LIU R K,HOU Y,et al. High hardware utilization and low memory block requirement decoding of QC-LDPC codes[J]. Chinese Journal of Aeronautics, 2012,25(5):747–756.

[9] GB 20600–2006. Framing structure,channel coding and modulation for digital television terrestrial broadcasting system[S]. Beijing:[s.n.], 2006.

作者简介:

赵　岭(1980–),男,湖南省益阳市人,博士,讲师,主要研究方向为无线通信、信道编译码.email:zhaoling@buaa.edu.cn.

王荣博(1982–),男,北京市人,硕士,主要研究方向为卫星通信系统.

李　众(1992–),男,石家庄市人,在读硕士研究生,主要研究方向为信道编码.

朱曼洁(1981–),女,北京市人,硕士,主要研究方向为卫星通信系统.

(上接第 836 页)

作者简介:

李　丹(1987–),女,吉林省德惠市人,在读硕士研究生,主要研究方向为信道编码参数盲识别.email:lidan2702@gmail.com.

廖红舒(1978–),女,广西壮族自治区贵港市人,副教授,主要研究方向为通信信号分析识别、信道编码盲识别.

甘　露(1974–),男,成都市人,教授,博士生导师,主要研究方向为自适应及阵列信号处理、高速实时信号处理技术、非合作信号处理、非线性信号处理技术、混沌通信等.