

基于 3GPP 标准的 Turbo 码译码器设计与实现

刘晓明,解志强,彭芳芳

(重庆大学 通信工程学院,重庆 400030)

摘要:文中针对 3GPP 标准的 Turbo 码的性能进行仿真分析,基于课题的要求,根据性能和 FPGA 硬件实现复杂度提出了一种新颖的译码器方案。本方案采用在分量译码器计算前向递推的数据时,只对前向递推量进行存储,在后续过程中将同时计算出的分支度量和后向递推量结合已经存储的前向递推量直接更新信息比特的似然信息和外信息,节省了硬件存储器资源,提高了译码吞吐量,根据硬件系统时钟可推算出大致的译码吞吐量,达到课题要求。本方案的思想同样可推广应用于其他标准的 Turbo 码译码器。

关键词:Turbo 码;3GPP;FPGA;译码器

中图分类号:TP311

文献标识码:A

文章编号:1673-629X(2010)11-0022-03

Design and Implementation of Turbo Decoder Based on 3GPP

LIU Xiao-ming, XIE Zhi-qiang, PENG Fang-fang

(College of Communication Engineering, Chongqing University, Chongqing 400030, China)

Abstract: Analyse the performance of Turbo code based on 3GPP protocol through simulation, in accordance with the requirements of the subject, a novel decoder scheme is proposed considering the performance and the implementation complexity of FPGA. This scheme only stores the forward path metrics in the process of computing itself, in the next round it computes the branch metrics and backward path metrics simultaneously, which are combined with the forward path metrics stored in the first round immediately in the purpose of updating LLR and extrinsic information of the information bits. This scheme saves lots of hardware memory resources and improves the throughput of decoder which can be estimated according to the system clock. The requirement of the subject can be achieved using this scheme, and this scheme can also be used in the other Turbo decoders of different standards.

Key words: Turbo code; 3GPP; FPGA; decoder

0 引言

Turbo 码在 1993 年瑞士日内瓦召开的国际通信会议上,由 C. Berrou, A. Glavieux 和 P. Thitimajshima 首次提出^[1],它将卷积码和随机交织器结合,同时采用软输出迭代译码来逼近最大似然译码^[2,3],它以接近 Shannon 限的优异性能被 CCSDS 和 3GPP 等标准采用。近年来,同样被广泛应用于 OFDM 系统^[4]。Turbo 码的软译码中,计算过程复杂,存储量大,围绕其时延问题,有许多文章提出了解决方法^[5-9],从中可以看出分量译码器是设计的关键所在,不同标准中的 Turbo 码的结构不同,文中利用 FPGA 内在的电路并行结构特点设计了一种新颖的基于 3GPP 标准的译码器,这种方案的思想同样可推广到其他标准的 Turbo 码。

1 译码算法分析与仿真

3GPP 标准中 Turbo 码的生成矩阵为(13,15),约束长度为 4,交织器长度为 40~5114。Turbo 码编码器为线性编码,编码复杂度低,容易实现,对于编码器不做赘述。

对于译码算法,主要有两大类:一类是基于最大后验概率 MAP 的软输出算法,其中 Log-MAP 算法^[10]是理想情况,Max-Log-MAP 算法^[11]是其简化算法;另一类是基于 Viterbi 算法的软输出算法,包括 SOVA 算法^[12]等,下面对这三种算法进行过程和性能的分析。定义信息比特 u_k ,分量编码器输出为 $c_k = \{c_k^1, c_k^2\}$ 。经过信道传输后在接收端接收符号为 $y_1^N = \{y_1, y_2, \dots, y_N\}$,其中 $y_k = \{y_k^1, y_k^2\}$ 。参考信道为加性高斯白噪声(AWGN)信道。

1.1 Log-MAP 算法和 Max-Log-MAP 算法

在 Log-MAP 算法中,信息比特的对数似然比:

$$L(u_k) = \max_{\substack{(s',s) \\ u_k=1}} [A_{k-1}(s') + M_k(s',s) + B_k(s)]$$

收稿日期:2010-03-04;修回日期:2010-06-11

基金项目:国家发改委 CNGI 示范工程项目(CNGI-04-4-2D)

作者简介:刘晓明(1963-),男,重庆市人,博士后,教授,研究方向为扩频通信、宽带无线通信。

$$- \max_{\substack{(s',s) \\ u_k=0}} [A_{k-1}(s') + M_k(s',s) + B_k(s)] \quad (1)$$

其中, $\max_x (f(x)) = \ln(\sum_x e^{f(x)})$, 如果 x 和 y 相差较大, 则得到 $\max(x, y) \approx \max(x, y)$, 就有了 Max-Log-MAP 算法。这样虽然减少了算法的计算量, 但同时也带来了 0.3dB ~ 0.5dB 的编码增益损失。

1.2 SOVA 算法

SOVA (Soft-Output Viterbi Algorithm), SOVA 算法是 Viterbi 算法的改进, 它对传统 Viterbi 算法上进行修正, 采用软判决计算度量值, 输出相应判决比特和可靠性信息。

1.3 算法性能仿真

仿真结果如图 1 所示, 基于 Matlab 环境下, 使用随机输入数据, 码率为 1/3, 帧长为 420, 迭代 5 次。由此可以看出 SOVA 算法虽然最简单, 但性能也最差, 而 Log-MAP 算法性能为三者中最好。Max-Log-MAP 算法的性能介于两者之间, 但计算量大大少于 Log-MAP 算法, 而且其中的算法非常利于 FPGA 的硬件实现。所以文中设计的译码器采用 Max-Log-MAP 算法。

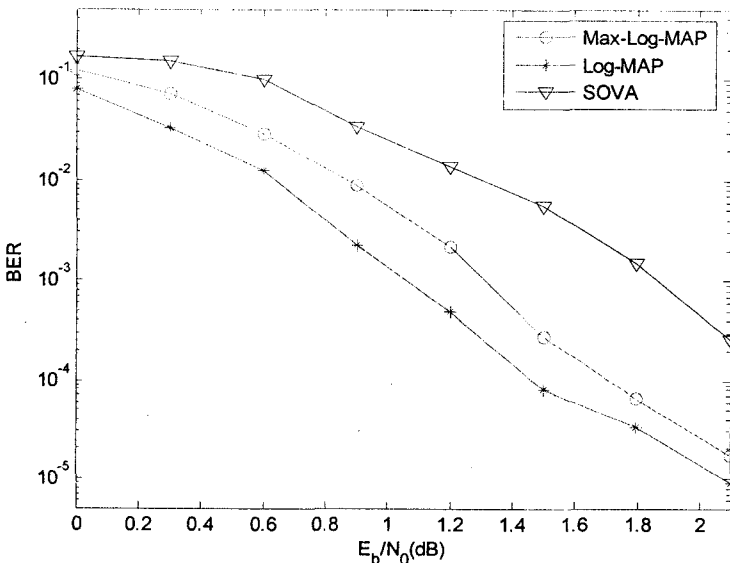


图 1 三种算法性能比较

2 译码器设计与实现

技术指标:

(1) 基带速率 10.24 Mb/s, 编码效率 1/3, 编码后比特速率 30.72 Mb/s;

(2) 基带速率 40.96 Mb/s, 编码效率 2/3, 编码后比特速率 61.44 Mb/s。

Turbo 码译码过程主要由 2 个分量译码器完成, 其结构一致。译码器的总体结构如图 2 所示。

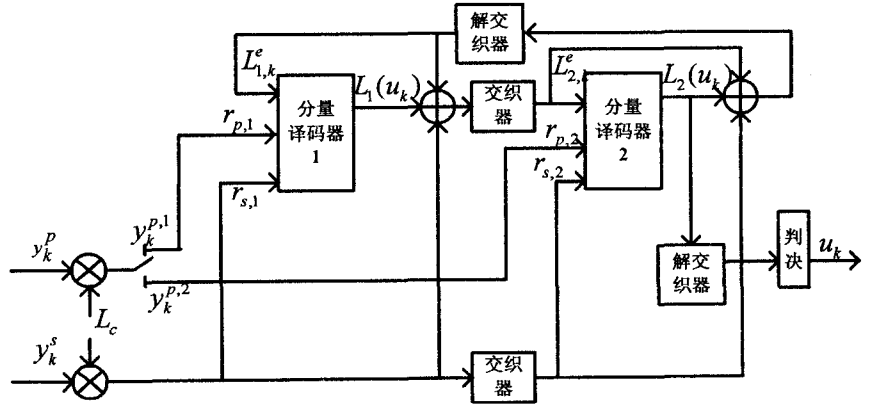


图 2 译码器实现结构图

分量译码器采用 Max-Log-MAP 算法, 计算详细步骤如下:

(1) 接收信息序列的同时计算出 $M_k(s', s)$ 和 $A_k(s)$;

$$M_k(s', s) = uL^e(u_k) + L_{\mathcal{C}_k^s}u_k + L_{\mathcal{C}_k^p}p_k \quad (2)$$

$$A_k(s) = \max_{s'} [A_{k-1}(s') + M_k(s', s)] \quad (3)$$

(2) 当接收到整个序列 Y_k^N 后, 再通过前向迭代得到 $B_k(s)$, $L(u_k)$ 和先验信息 L_k^e ;

$$B_{k-1}(s') = \max_s [B_k(s) + M_k(s', s)] \quad (4)$$

$$L(u_k) = \max_{\substack{(s',s) \\ u_k=1}} [A_{k-1}(s') + M_k(s', s) + B_k(s)] - \max_{\substack{(s',s) \\ u_k=0}} [A_{k-1}(s') + M_k(s', s) + B_k(s)] \quad (5)$$

$$L_k^e = L(u_k) - L_{\mathcal{C}_k^s}u_k - L_{\mathcal{C}_k^p}p_k \quad (6)$$

(3) 对 $L(u_k)$ 硬判决得到译码输出。

其中因为编码器起始状态均为零状态, $A_k(s)$ 和 $B_k(s)$ 初始值为 0。

Turbo 码译码过程中存在很大计算量, 有前向递推 $A_k(s)$, 分支转移概率 $M_k(s', s)$, 后向递推 $B_k(s)$, 信息比特后验概率对数似然比值 $L(u_k)$ 和先验信息 $L^e(u_k)$, 为了大量节省存储空间, 该结构不存储分支转移概率 $M_k(s', s)$ 和后向递推 $B_k(s)$, 下面分别介绍其详细实现过程:

(1) 分支转移概率 $M_k(s', s)$ 的计算。

3GPP 标准中生成矩阵为 (13, 15), 所以共有 8 个状态: $s_1 = (000)$, $s_2 = (001)$, $s_3 = (010)$, $s_4 = (011)$,

(2) 分支转移概率 $M_k(s', s)$ 的计算。

3GPP 标准中生成矩阵为 (13, 15), 所以共有 8 个状态:

$s_1 = (000)$, $s_2 = (001)$, $s_3 = (010)$, $s_4 = (011)$,

$s5 = (100), s6 = (101), s7 = (110), s8 = (111)$ 。

每一时刻 k 时, $M_k(s', s)$ 有 16 个值, 根据式(2), $M_k(s', s)$ 的计算如下:

$$M_S21 = M_S71 = M_S81 = M_S11 = L_e + Y_s + Y_p;$$

$$M_S41 = M_S51 = M_S61 = M_S31 = L_e + Y_s + 0;$$

$$M_S30 = M_S40 = M_S50 = M_S60 = 0 + 0 + Y_p;$$

$$M_S10 = M_S20 = M_S70 = M_S80 = 0。$$

由于 $M_k(s', s)$ 每时刻均有 16 个值, 其量化是 5bit, 则存储量最大是 $5114 \times 16 \times 5\text{bit}$, 而它计算过程简单, 因此为了节省存储空间, 第一次从前向后得出的 $N_k(s', s)$ 只用作 $A_k(s)$ 的计算, 并不存储。第二次从后向前读取信息数据和校验数据重新计算 $M_k(s', s)$, 并同时计算出 $B_k(s)$ 和 $L(u_k)$ 。

(2) 前向递推 $A_k(s)$ 的计算。

在一帧数据接收过程中, 同时计算出每一时刻的 16 个分支转移概率 $M_k(s', s)$ 值, 根据式(3) 计算出每时刻 $A_k(s)$ 的 8 个值, 并存储 $A_k(s)$ 所有值。

$A_k(s)$ 计算流程如图 3 所示。

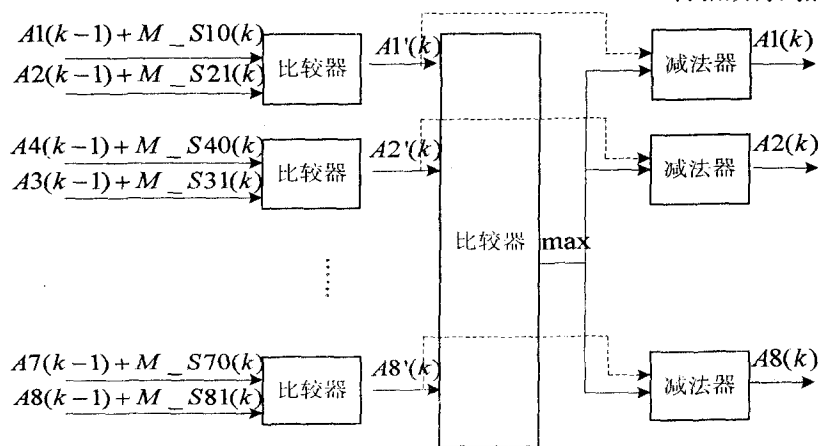


图 3 前向递推计算流程

首先根据前面计算出来的 k 时刻的分支转移概率 $M_k(s', s)$ 和 $k-1$ 时刻的前向递推 $A_{k-1}(s)$ 的值, 分别计算出 $A1(k-1) + M_S10(k)$, $A2(k-1) + M_S21(k)$ 到 $A8(k-1) + M_S81(k)$ 共 16 个值, 再分别通过比较器, 根据比较结果算出 $A1'(k)$, $A2'(k)$ 到 $A8'(k)$ 的值。由于前向递推 $A(k)$ 是一个递增的量, 为了防止运算溢出, 需要对 $A(k)$ 进行归一化处理。通过将 $A1'(k)$, $A2'(k)$ 到 $A8'(k)$ 送入比较器, 选取出其中最大的值 \max , 再将 $A1'(k)$, $A2'(k)$ 到 $A8'(k)$ 8 个值分别减去 \max , 即得到归一化后的 $A1(k)$, $A2(k)$

到 $A8(k)$ 8 个值, 达到防止溢出的作用。

对于 $A_k(s)$ 的存储情况, 由于 $A_k(s)$ 在每一时刻均有 8 个值, 为了便于同时读写这 8 个值, 以保证并行运算, 达到译码速度的要求, 这里将 8 个值并行排列存储到一个地址单元, 则一个地址单元将存储 $8 \times 5\text{bit}$ 。由于 $A_k(s)$ 值仅在计算 $L(u_k)$ 时用到, 且要经过减法得到最终 $L(u_k)$ 值, 为了进一步节省存储空间, $A_k(s)$ 值采用 4bit 位宽。 $A_k(s)$ 存储方式如图 4 所示。

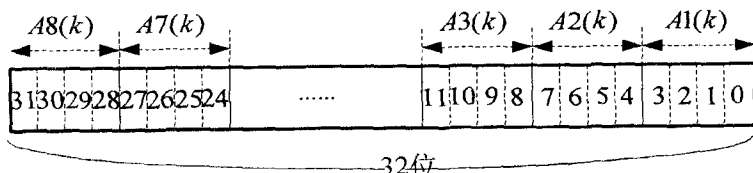


图 4 $A_k(s)$ 存储方式示意图

(3) 后向递推 $B_k(s)$ 的计算。

$B_k(s)$ 的计算同 $A_k(s)$ 一样, 原理如式(4)。计算出 $B_k(s)$ 后, 根据式(5) 和式(6) 用当前的 $M_k(s', s)$, $B_k(s)$ 及从存储器中读取的 $A_k(s)$ 计算出 $L(u_k)$ 和 $L^e(u_k)$ 。

(4) 后验概率 $L(u_k)$ 和先验信息 $L^e(u_k)$ 的计算。

根据 $L(u_k)$ 的计算公式(5), 将前面部分的 8 个值和后面部分的 8 个值分别送入比较器, 求出最大值后再相减得到后验概率 $L(u_k)$, 再根据式(6) 计算出先验信息 $L^e(u_k)$ 。最后一次迭代时将分量译码器 2 输出的后验概率 $L(u_k)$ 经交织后存储到 Info RAM 中, 经判决器输出译码后的信息数据。

从上面分析可以看到, 对于采用 5 次迭代的译码过程, 译出每个信息位需要 20 个系统时钟, 即对于 70Mbits/s 的 FPGA 系统时钟可达到约 3Mbits/s 译码吞吐量。采用 14 路译码器并行译码可达到 42Mbits/s 的吞吐量, 满足本课题最大到 40.96Mbits/s 的基带译码吞吐量要求。

3 结束语

仿真分析了各种算法下 3GPP 标准中 Turbo 码的性能, 从性能和实现复杂度两方面考虑, 选择出最适合于 FPGA 实现的算法。设计了一种新颖的分量译码器存储方案, 大大节省了硬件存储器资源, 并可以推广到其他标准的 Turbo 码。在系统时钟确定的情况可推算出单路译码器的吞吐量, 可采取多路并行译码的方式解决译码时延的问题, 从而达到高速译码的效果。

(下转第 28 页)

量数据的采集是每隔 1 秒采集一个数据,第一次采集了 256 组数据。

(2) 在 Matlab 6.5 中,分别实现小波卡尔曼滤波流量预测模型。

(3) 把步骤(1)中所采集得到的 MPLS 网络流量样本序列化为矩阵,代入 WKHEFA 网络流量预测进行流量预测仿真,分别进行 1 步预测、3 步预测、5 步预测、7 步预测。

(4) 将仿真结果利用 Matlab 6.5 绘图工具绘制成图。仿真结果如图 2 所示。

从图中可以看出,WKHEFA 预测模型一步预测曲线和真实 MPLS 网络流量曲线的拟合度较好,表示预测误差小、精度高,相对准确地预测出 MPLS 网络流量;随着预测步长增大,预测曲线相对偏离真实 MPLS 网络流量曲线,表明预测误差增大和预测精度下降。仿真结果表明,WKHEFA 预测模型可以用来进行实时 MPLS 网络流量预测,主要基于如下原因:WKHEFA 是在卡尔曼滤波的基础上引入小波变换过程,相对于传统的线性预测模型如 AR,MA,ARMA 等模型,卡尔曼滤波不需存储和计算大量的网络流量历史数据,只是基于实时的更新信息,也即是“新息”,便可以保证其预测精度;WKHEFA 运算复杂度与非线性预测模型如神经网络等相比,复杂度较低,便于实时预测。

3 结束语

文中针对 MPLS 网络流量的特征,结合小波与卡尔曼滤波提出了一种新的实时网络流量预测方法,并在 Matlab 6.5 仿真平台中实现,仿真结果表明,具备较

好的预测精度,误差率低。

参考文献:

- [1] 刘正茂,王永骥. MPLS: ATM 与 IP 结合的新模式[J]. 河北省科学院学报, 2003, 20(1): 45 - 47.
- [2] 孟祥迪,郭静寰,熊木地. 电信级路由器的 MPLS 技术原理及性能测试[J]. 计算机技术与发展, 2006, 16(10): 207 - 210.
- [3] 刘广义,周海军,林孝康. MPLS 关键技术研究[J]. 计算机工程与应用, 2002(15): 130 - 133.
- [4] 陈勇,陈建勋,张勇. 基于 MPLS 流量工程的网络仿真技术[J]. 武汉科技大学学报, 2006, 29(3): 273 - 276.
- [5] Awduche D, Malcolm J, Agogbua J, et al. Requirements for Traffic Engineering over MPLS[S]. RFC 2702, 1999.
- [6] Swallow G. MPLS advantages for traffic engineering[J]. IEEE Communications Magazine, 1999, 37(12): 54 - 57.
- [7] Xiao Xipeng, Hannan A, Bailey B, et al. Traffic engineering with MPLS in the internet[L]. IEEE Network, 2000, 14(2): 28 - 33.
- [8] Zheng Tongxin, Girgis A, Makram B. A hybrid wavelet - Kalman filter method for load forecasting[J]. Electric Power Systems Research, 2000, 54(1): 11 - 17.
- [9] 任慧玉. 最优估计与小波分析理论在经济分析中的应用研究[D]. 河南: 河南大学, 2004.
- [10] 李海霞,宋玲. 基于 NS2 的 MPLS 仿真模拟技术分析[J]. 计算机技术与发展, 2006, 16(12): 26 - 28.
- [11] 李蓬,黄河. 基于 NS2 的 MPLS 流量工程仿真研究[J]. 计算机技术与发展, 2008, 18(9): 53 - 56.
- [12] Kolarov A, Atai A, Hui J. Application of Kalman filter in high - speed networks[C]//GLOBECOM '94. San Francisco, U.S.A.: [s. n.], 1994: 624 - 628.
- [7] 蔡剑卿. 基于 FPGA 的改进 Turbo 译码器的设计与实现[J]. 福建电脑, 2009(11): 133 - 134.
- [8] 赵旦峰,焉晓贞. 一种改进 Turbo 码译码器的 FPGA 设计与实现[J]. 电子技术应用, 2008, 34(3): 32 - 35.
- [9] 张桂华,桑会平,姬红兵. 基于 FPGA 的 Turbo 码译码算法实现[J]. 系统工程与电子技术, 2008, 30(8): 1584 - 1587.
- [10] Pietrobon S S, Barbulescu A S. A simplification of the modified bahl decoding algorithm for systematic convolutional codes [C]//Proc. Int. Symp on Inform. Theory and its Applie. Sydney, NSW: [s. n.], 1994: 1073 - 1077.
- [11] Robertson P, Villebrun E, Huber P. A comparison of optimal and sub - optimal MAP decoding algorithms operating in the log domain[C]//Proc. ICC'95. Seattle, USA: [s. n.], 1995: 1009 - 1013.
- [12] Hagenauer J, Hoehner P. A viterbi algorithm with soft - decision outputs and its applications[C]//Proc. Globecom'89. [s. l.]: IEEE, 1989: 1680 - 1686.

(上接第 24 页)

参考文献:

- [1] Berrou C, Glavieux A, Thitimajshima P. Near Shannon limit error - correcting coding and decoding: Turbo - codes (1) [C]//Proc. ICC'93. Geneva, Switzerland: [s. n.], 1993: 1064 - 1074.
- [2] 刘东华. Turbo 码原理与应用技术[M]. 北京: 电子工业出版社, 2000.
- [3] 邢莉,王忠,李兴国. 基于 Matlab 的 Turbo 码仿真研究[J]. 现代电子技术, 2009(3): 19 - 21.
- [4] 彭芳芳. OFDM 系统中的 Turbo 码编译码技术研究及实现[D]. 重庆: 重庆大学, 2009.
- [5] 张中培,周亮,靳蕃. 低复杂度 Turbo 码译码并行实现[J]. 电子学报, 2001(2): 272 - 274.
- [6] 王坤,张青青,冯加建. Turbo 码高速译码器设计[J]. 现代电子技术, 2008, 31(18): 171 - 173.