Turbo码编码/译码算法的FPGA实现

张新苗，赵雅兴
（天津大学电子信息工程学院，天津 300072）

摘要：提出了一种用FPGA实现Turbo码编码/译码的方法。利用简单的查找表可实现经过转化后的译码算法中复杂的运算。本设计使用Altera公司的FPGA器件实现。计算机模拟表明，本设计所实现的Turbo码具有良好的性能和实用价值。

关键词：Turbo码；迭代系统卷积码；Log-MAP算法；现场可编程逻辑器件
中图分类号：TN919.3 **文献标识码**：A **文章编号**：1003-353X(2001)05-0042-04

Method of Turbo coding performed by FPGA

ZHANG Xin-miao, ZHAO Ya-xing
(Institute of Electro Infor & Engineering, Tianjin Universty, Tianjin 300072, China)

Abstract: In this paper, a new implementation method of Turbo codes by FPGA was proposed. Using a simple look-up table, the complex arithmetic operation in the decoding algorithm can be completed; The whole system was implemented by FPGA technology from Altera Corp. The simulation result testified that the whole system has good performance, and it can be put into practical use.

Keywords: Turbo codes; RSC code; Log-MAP algorithm; FPGA

1 引 言

1993年C.Berron等人提出的并行级连码(Parellel Concatenated Code)亦称Turbo码，是一种接近香农极限的信道纠错编码。Turbo码在低于香农极限0.7dB的情况下，软件仿真可以得到10^{-5}的误码率，因此引起国际上广泛的重视。该码一经提出，研究人员就对把Turbo码应用于深海通信以及卫星通信等功率较低的通信方式进行了研究，并且在第三代蜂窝和个人通信等方面也有广泛的用途，将对第三代移动通信的发展带来巨大的冲击。Turbo码的三个最重要的组成部分为：(1) 可通过栅格图解码的线性码（分组码或卷积码）；(2) 交织器；(3) 迭代软输入/软输出解码器。

2 Turbo码编码器的实现

Turbo码是两个或多个系统码的并行级连。图1为通常使用的具有两个编码单元的Turbo码的编码原理图。其中迭代系统卷积（RSC）编码单元是把卷积码改造后得到的，其原理如图2所示。对于RSC编码单元同样也可以定义存储长度ν，ν与编码单元中的存储单元个数相对应。

交织器是Turbo交织器是Turbo编码器中一个重要的部分，不同的交织器使得所生成码字的最小码重不同，而最小码重是决定纠错码性能的一个重要指针，因此交织器是使Turbo码达到最佳性

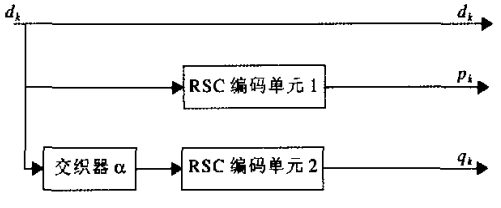

图1 具有两个编码单元的Turbo码编码器原理图

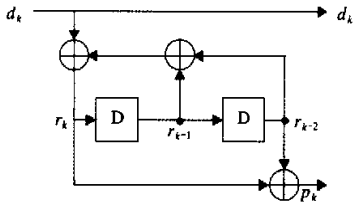

图2 迭代系统卷积编码器

能的关键。本设计采用螺旋奇偶交织器，它的交织序列和解交织序列是相同的，这样可以节省一部分资源，并具有良好的性能[1]。这种交织器的列数为v+1的偶数倍，行数为列数减1，按行的顺序依次将数据写入，而在读出时由左下角开始按照螺旋上升的顺序读出数据。例如，对于一个5×6=30的交织器，其输出序列如图3所示。交织器的FPGA实现实际上是把交织序列存储到一个查找表中，在输出时先从查找表中找到应输出数据的地址，再以这个地址作为存储数据的RAM的地址输入，从而得到应输出的数据。

d_1	d_2	d_3	d_4	d_5	d_6
d_7	d_8	d_9	d_{10}	d_{11}	d_{12}
d_{13}	d_{14}	d_{15}	d_{16}	d_{17}	d_{18}
d_{19}	d_{20}	d_{21}	d_{22}	d_{23}	d_{24}
d_{25}	d_{26}	d_{27}	d_{28}	d_{29}	d_{30}

d_{25}	d_{20}	d_{15}	d_{10}	d_5	d_{30}	d_{19}	d_{14}	d_9	d_4	d_{29}	d_{24}	d_{13}	d_8	d_3	…

图3 螺旋奇偶交织器结构及其输出序列

Turbo码编码器顶层文件分为四个模块：移位寄存器模块、交织器模块和两个RSC编码器模块，其中移位寄存器模块的作用是使编码器中的各路输出同步。

3 Turbo码译码器的实现

Turbo码译码器的核心是MAP译码器，Turbo码的译码过程实际上是通过把MAP译码器的软输出作为下一个MAP译码器的输入，经过多次反复后，通过硬判决得到最终结果。图4为Turbo码译码器的原理图。

图中，$a_k=(2d_k-1)+n_k$；$b_k=(2p_k-1)+m_k$；$c_k=(2q_k-1)+l_k$。

d_k，p_k，q_k为Turbo码编码器的输出（见图1）；n_k，m_k，l_k是方差为σ^2、均值为0的高斯白噪声变量。图4中，在开始译码前，把c_0置0，为了提高性能，把译码器MAP2的外部信息(W_2)经过解交织后，利用反馈环，把它与a_k相加作为下一轮译码过程中译码器MAP1的一个输入，进行循环译码。

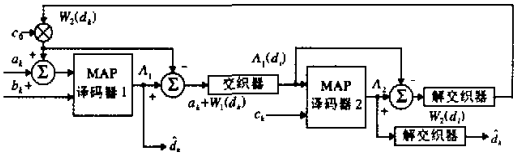

图4 Turbo码译码器原理图

本设计采用Log-MAP算法来实现MAP译码器。Log-MAP算法把复杂的乘除法转化为简单的加减法，同时通过调用参考文献[1]中的E函数实现转化后的幂运算，不但大大降低了运算复杂度，而且便于硬件实现。下面介绍Log-MAP算法。

对于图1所示的RSC编码器，可以用图5的栅格图表示其状态转移过程及相应的输出。这个栅格图中的每一个分支给出三种信息：分支的状态（当前状态和下一状态）、编码器的输出校验比特p_k和所输入的信息比特d_k。虚线表示输入的信息比特为0，实线表示输入的信息比特为1。例如，对于图中的实线，我们可得知在这个分支上输入的信息比特为1，编码器的状态从00变到01，并且编码器的输出为11(d_k=1，p_k=1)。

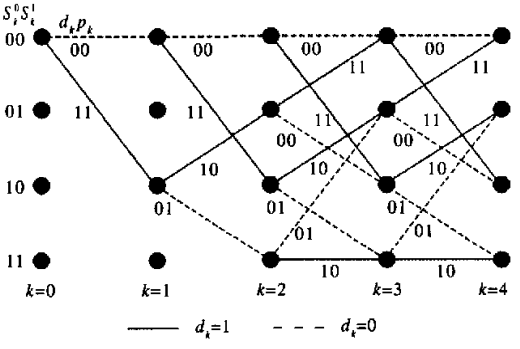

图5 v=2的RSC编码器的栅格图

信息比特经过RSC编码器后，再通过BPSK调制，经过加性高斯白噪声信道(AWGN)后，在接收端我们接收到的序列为$R_{1,N}$=（R_1，R_2，…R_k，…R_N），其中，R_k=（a_k，b_k）。

设m_k，n_k均是方差σ^2均值为0的高斯随机变量，d_k，p_k为编码器的输出。我们得到：

$$a_k=(2d_k-1)+n_k$$

$$a_k=(2p_k-1)+m_k$$

Log-MAP算法的目的是根据$R_{1,N}$找到最可能的d_k，我们定义对于每一个d_k，其相应的对数似然概率LLR（Log-likelihood ratio）Λ_k为

$$\Lambda_k(d_k)=\log\left(\frac{\Pr(d_k=1|R_{1,N})}{\Pr(d_k=0|R_{1,N})}\right)$$

我们通过计算上式得到的是一个软输出，可通过如下的判决得到最可能的d_k：

如果$\Lambda_k(d_k)\geq 0$那么$d_k=1$；

如果$\Lambda_k(d_k)<0$那么$d_k=0$。

根据参考文献[2]定义E运算为：

$xEy=-\frac{1}{L_c}\ln\left(e^{-L_c x}+e^{-L_c y}\right)$，其中$L_c=2/\sigma^2$。

使用E运算，我们定义分支向量$D_i(R_k,m)$，状态向量$A_{k,i}(m)$及$S_{b,j}(m)$。

分支向量$D_i(R_k,m)$的表达式依赖于传输信道特性，在加性高斯白噪声信道的情况下

$$D_i(R_k,m)=a_k i+b_k p_{k,i}(m) \quad (1)$$

分支向量$D_i(R_k,m)$对应图5所示栅格图中从时刻k到时刻$k+1$，编码器初始状态为m，输入信息比特为i的分支。$A_{k,i}(m)$代表在k时刻，当输入为i时，从状态m转移到下一状态的前向状态矢量，$S_{b,j}(m)$是当输入比特为j时而转移到状态m的前一个状态。例如根据图5 $S_{b,i}(00)=01$。

$$A_{k,j}(m)=D_i(R_k,m)+E_{j=0}^{1}A_{k-1,j}\left(S_{b,j}(m)\right) \quad (2)$$

$B_{k,i}(m)$叫做反向状态向量，表示在时刻k，当输入为i，从状态m转移到下一状态的反向状态矢量，$D_j(R_{k+1},S_{f,i}(m))$为以前定义过的分支向量，$S_{f,i}(m)$为当输入为i时由状态m转移到的下一个状态。例如根据图5，$S_{f,1}(01)=10$

$$B_{k,j}(m)=E_{j=0}^{1}\left[B_{k+1,j}\left(S_{f,i}(m)\right)+D_j\left(R_{k+1},S_{f,i}(m)\right)\right] \quad (3)$$

最后得到：

$$\Lambda_k=E_{m=0}^{2^v-1}\left[A_{k,1}(m)+B_{k,1}(m)\right]-E_{m=0}^{2^v-1}\left[A_{k,0}(m)+B_{k,0}(m)\right] \quad (4)$$

Log-MAP算法的计算步骤如下：

第一步：从$k=0$开始，用式(1)计算并存储所有的$D_i(R_k,m)$；

第二步：在$k=N-1$时初始化B，令

$B_{N-1,0}[S_{f,0}(00)]=B_{N-1,1}[S_{f,1}(00)]=0$。

对于其它的所有m和i，令

$$B_{N-1,0}(m)=B_{N-1,1}(m)=\infty;$$

第三步：使用式(3)，从时刻$k=N-2$开始到$k=0$计算并存储所有的B；

第四步：在时刻$k=0$初始化A，令$A_{0,0}(00)=A_{0,1}(00)=D_i(R_0,00)$，对于其它的所有$m$令$A_{0,0}(m)=A_{0,1}(m)=\infty$；

第五步：使用式(2)，从$k=1$开始，到$k=N-1$计算并存储所有的A；

第六步：从$k=0$开始，到$k=N-1$用(4)式计算Λ_k，并做出判决。

同时E函数可简化为

$$xEy=\min(x,y)-\frac{1}{L_c}\ln\left(1+e^{-L_c|y-x|}\right)。$$

由于$xEy=yEx$，函数$f(z)=\ln(1+e^{-z})$ $(z\geq 0)$的最大值为：当$z=0$时，$f(z)=\ln(2)$。随着z的增加，$f(z)$很快减小，当$z\geq 7$，$f(z)$趋于零。这些特性使得可以用一个很小的查找表实现E函数。

本设计采用定点数，数据宽度为8位。其中第1位为符号位，第2位到第5位为整数位，第6~8位为小数位。采用定点数可以使数据的一部分数据位直接作为地址对E函数进行查表。Log-MAP译码器的实现中，我们采用并行算法，分别用8个而不是1个大的RAM存储分支矢量以及状态矢量。例如，采用RAMD[7..0]存贮计算所得的$D_i(R_k,m)$值，把RAMD的序号用二进制表示一共有$\log_2 8=3$位。我们令第一位表示$D_i(R_k,m)$中的i，由于本设计所实现的RSC编码器中$v=2$，所以一共有四种状态，我们可以用第二位和第三位表示m。如我们用RAMD[5]，也就是RAMD[101B]，存储$D_i(R_k,01)$的值。这样可以用一个从0增加到$N-1$的计数器作为RAMD[7..0]的写入地址，对于每一个计数器的值k，每个RAMD各存储一个值，当计数器的值

从0变到$N-1$，就可以把所有的$D_i(R_k,m)$存储下来。同样，采用RAMA[7..1]和RAMB[7..1]分别存储$A_{k,t}(m)$和$B_{k,t}(m)$。在对Log-MAP算法进行深入分析的基础上，我们发现这种用8个RAM存储矢量的方法不仅有利于提高译码速度，而且$A_{k,t}(m)$和$B_{k,t}(m)$相邻时刻之间的值存在着一定的逻辑关系，这种关系可以通过控制不同RAMA和RAMB之间的输入输出实现。通过对(2)式和(3)式以及图5的分析，我们得到如图6中相邻时刻之间各个RAMA和RAMB之间的逻辑关系。

根据图6中各RAM相邻时刻之间的逻辑关系，可以通过控制上一时刻所得计算结果通过(2)和(3)式定义的计算之后，在下一时刻存储到相应的RAM中去，通过迭代计算可以得到所有的状态矢量。

Turbo码译码器由三个模块组成：MAP译码器模块、交织器/解交织器模块和时序控制模块。时序控制模块的作用是产生各使能信号，使另外两个模块时序严密配合，完成Turbo码的译码功能。在本设计中，通过合理的模块分割以及参数化设计，使本设计具有相当的灵活性，同时采用自底向上与自顶向下相结合的设计方法，将各模块连接起来，构成总的工程设计文件。由于系统规模较大，MAX+PLUSII在编译时自动将译码器分割为三个器件：EPF10K200SBC356-1、EPF10K130EQC240-1和EPF10K30EQC208-1。表1为整个系统资源使用情况。

表1 Turbo码译码器的资源利用情况

Device	EPF10K200SBC356-1	EPF10K130EQC240-1	EPF10K30EQC208-1
Total dedicated input pins used	6/6(100%)	6/6(100%)	6/6(100%)
Total I/O pins used	197/258(73%)	155/180(86%)	115/141(81%)
Total Logic cells used	2472/9985(24%)	5739/6656(86%)	1048/1782(60%)
Total embedded cells used	165/384(42%)	12/256(4%)	21/96(21%)
Total EABs used	24/24(100%)	4/16(25%)	4/6(66%)
Total flipflops required	190	4995	48

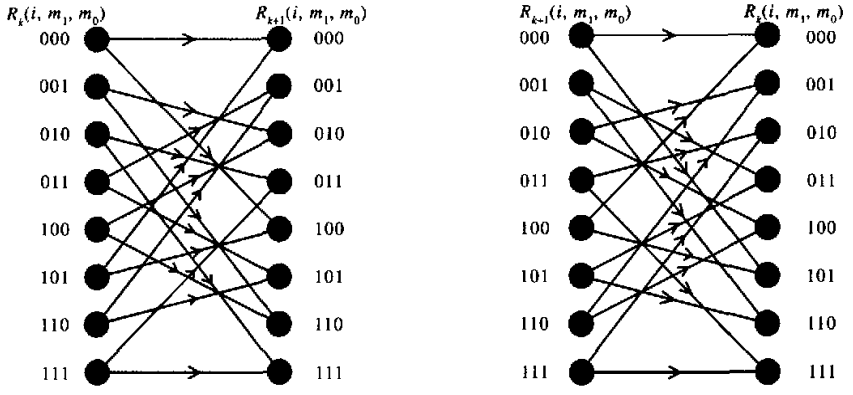

图6 RAMA和RAMB相邻时刻数据输入输出转移逻辑关系图

4 结论

本文提出了一种用FPGA实现Turbo码编码/译码的方法，采用Log-MAP算法作为Turbo码的译码算法，并针对FPGA实现进行了优化。计算机仿真结果表明，本设计所实现的Turbo码编码/译码系统性能较好，具有广泛的用途，也为今后硬件实现性能更好的Turbo码作了技术准备。

参考文献

1 Barbulescu A. Iterative decoding of turbo codes and other

（下转第60页）

（上接第45页）

concatenated codes. Ph D Dissertation, University of South Australia, 1996

2 Berrou C, Glavieux A. Near optimum error-correcting coding and decoding: Turbo Codes. IEEE Trans Commun, 1996; 44(10): 1261

3 Benedetto S, Montorsi G. Unveiling Turbo codes: some results on parallel concatenated coding schemes. IEEE Trans Commun, 1996; 42(2): 409

4 Valenti M. Iterative detection and decoding for wireless communications Ph D Dissertation, Virginia Polytechic Institute and State University, 1999

5 Herzberg H. multilevel Turbo coding with short interleavers. IEEE Journal Commun, 1998, 16: 303

6 Hagenauer J. Iterative decoding of binary block and convolutional codes. IEEE Trans Inform Theory, 1996; 42(2): 429

7 Viterbi A J. An Intuitive justification and a simplified implementation of MAP decoder for convolutional codes IEEE Journal Commun, 1998; 16(2): 260

8 赵雅兴. FPGA 原理设计与应用. 天津大学出版社. 1999

9 Data Book 1999, ALTERA.