应用于 LTE-OFDM 系统的 Viterbi 译码在 FPGA 中的实现

李小文，林　丹
（重庆邮电大学 计算机学院，重庆 400065）

摘　要：在 OFDM 系统中，为了获得正确无误的数据传输，要采用差错控制编码技术。LTE 中采用 Viterbi 和 Turbo 加速器来实现前向纠错。提出一种在 FPGA 中实现的基于软判决的 Viterbi 译码算法，并以一个(2，1，2)、回溯深度为 10 的软判决 Viterbi 译码算法为例验证该算法，在 Xilinx 的 XC3S500E 芯片上实现了该译码器，最后对其性能做了分析。

关键词：OFDM；Viterbi 译码；软判决；FPGA
中图分类号：TN492　　**文献标识码**：A

FPGA implementation of a viterbi decoder applied on LTE-OFDM system

LI Xiao Wen，LIN Dan
（College of Computer Science and Technology，ChongQing University of Posts and Telecommunications，Chongqing 400065，China）

Abstract：In OFDM system，the error control coding techniques is used for attaining accurate data transmitting. Viterbi and Turbo accelerators are applied on LTE to achieve the FEC. This paper presents an FPGA implementation of Viterbi algorithm based on soft-decision，and verifies it through an example which is(2，1，2) and the trace-back depth is 10. This decoder is implemented in Xilinx's XC3S500E chip. Finally，the algorithm performance is analyzed.

Key words：OFDM；Viterbi decoder；soft-decision；FPGA

LTE 采用下行正交频分多址(OFDM)、上行单载波频分多址(SC-FDMA)的方式。OFDM 是 LTE 系统的主要特点，其基本思想是把高速数据流分散到多个正交的子载波上传输，从而使子载波上的符号速率大大降低，符号持续时间大大加长，因而对时延扩展有较强的抵抗力，减小了符号间干扰的影响[1]。在 OFDM 系统中，为了获得正确无误的数据传输，需要采用差错控制编码技术。卷积编码和 Viterbi 译码就是一种有效的前向纠错方法，它具有一定的克服突发错误的能力[2]。LTE 中采用 Viterbi 和 Turbo 加速器实现前向纠错。

1 Viterbi 算法简介

Viterbi 译码算法是由 Viterbi 于 1967 年提出的降低计算复杂度的算法。它是计算网格图上在时刻 t_i 到达各个状态的路径和接收序列之间的相似度，或者说距离，去除不可能成为最大似然选择对象的网格上的路径。即如果有两条路径到达同一状态，则具有最佳度量的路径被选中，称为幸存路径。对所有状态都进行这样的选路操作，译码器不断在网格上深入，通过去除可能性最小的路径实现判决，从而降低译码器的复杂性[3]。

Viterbi 译码算法就是基于接收数据符号估计延时状态序列，重构通过整个网格的路径，实际包含度量更新和回溯过程。

2 Viterbi 算法实现流程

Viterbi 译码算法的实现流程如图 1。由图 1 可以看出，Viterbi 算法的主要实现过程可分为 4 大部分：分支

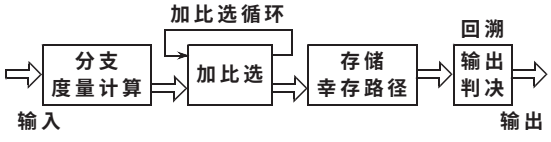

图 1　Viterbi 译码算法处理流程

度量计算(BMC);加比选(ACS);存储幸存路径存储器(SSM);输出判决(OD)。

3 Viterbi 译码实现过程

下面以一个(2,1,2)、回溯深度为 10 的软判决 Viterbi 译码算法为例(如图 2),通过这个例子可以比较清楚地了解 Viterbi 实现的过程。

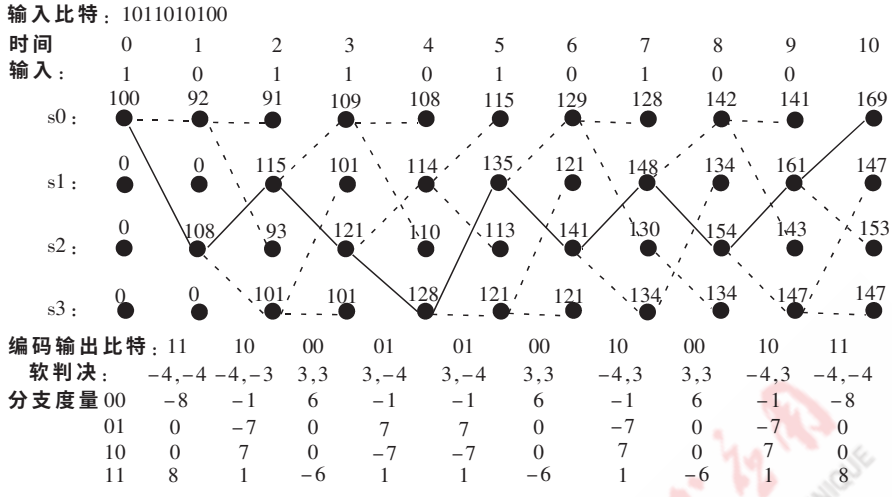

图 2 Viterbi 译码实现过程

在本例中,解码之前接收到 20 个编码后的符号。在某些应用中,Viterbi 译码根据接收到的符号逐帧进行译码,与邻帧之间相互独立,即在每个帧内进行维特比译码,各帧之间相互独立[4]。

3.1 分支度量

本文按以下方法决定分支度量:

考虑接收到的编码后的比特 $G_0(n)G_1(n)$。由于这里是进行软判决的维特比译码,因此在译码器的输入端将编码输出进行 3 bit 的量化,量化范围为-4~3。假设这里传送的是 11,则将 11 量化为(-4,-4)。若传送的为 00,则量化为(3,3),即将 0 量化为 3,1 量化为-4。

如图 2,在 0 时刻和 1 时刻接收到的比特为(1,1),若$(G_0(n)G_1(n))=(0,0)$,则对应的分支度量为$-4(1)-4(1)=-8$;若$(G_0(n),G_1(n))=(1,0)$,对应的分支度量为$-4(-1)-4(1)=0$;$(G_0(n),G_1(n))=(1,1)$,则为$-4(-1)-4(-1)=8$。

3.2 路径度量

这里计算路径度量使用的是欧氏距离。

$$PM(n,i)=\sum_j [S_j(n)-G_j(n)]2,\quad j\in\{\text{与输入相关的输出比特}\}$$

n 指示时间,i 指示要计算的路径。上式可扩展为:

$$PM(n,i)=\Sigma[S_j^2(n)-2S_j(n)G_j(n)+G_j^2(n)]^2,\quad j\in\{\text{与输入相关的输出比特}\}$$

因为 $S_j^2(n)$和 $G_j^2(n)$在给定的符号周期内为常数,在寻找具有最小路径度量的过程中可以忽略这两个常数,因此可由下式决定路径度量:

$$PM'(n,i)=\Sigma S_j(n)G_j(n)$$

注意,上式中系数-2 已被剔除,所以寻找具有最小路径度量 $PM(n,i)$的过程就转换为寻找具有最大路径度量 $PM'(n,i)$的过程。

在加比选的过程中做如下约定:凡是从上面状态转移过来的幸存路径,称之为"0"路径,如图 3 中实线所示;凡是从下面状态转移过来的幸存路径,称之为"1"路径,如图 3 中虚线所示。将这些幸存路径分别存储在相应的寄存器单元。

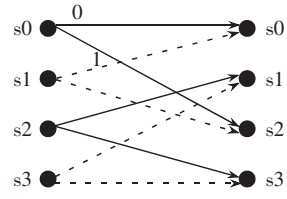

图 3 状态转移蝶形图

3.3 幸存路径存储器

由以上分析可知,ACS 在计算出新状态的路径度量的同时,也得出了到达这个状态的转移信息(用 1 bit 表示),该路径转移信息就需要保存在幸存路径存储器中,这样多个转移信息就可以"串成"一条完整的幸存路径。在本设计中,幸存路径的长度 $L=10$,所以使用 4 个 10 位的寄存器存储幸存路径。幸存路径存储模块如图 4,每一个小方框是寄存器的一个位。对于每一个状态,由 ACS 模块得到的路径转移信息都移入各自的幸存路径。这样每一条幸存路径存储单元就是一个长度为 10 的移位寄存器。

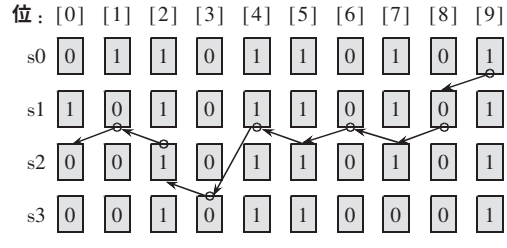

图 4 回溯过程

3.4 回溯

本设计的回溯算法如下:

(1)在 4 个状态的路径度量值中选取一个最大值;

(2)判断最大值对应的状态,作为回溯的起始状态,如果有多个,就根据状态的编号,按从小到大的顺序选取;

(3)根据幸存路径寄存器里存储的值进行判断,若是"0",则是由上面状态转移过来;若为"1",则是下面状态转移过来(如图 3)。根据这些信息回溯到对应的前一状态。回溯节点逐级前移,从与之对应的幸存路径寄存

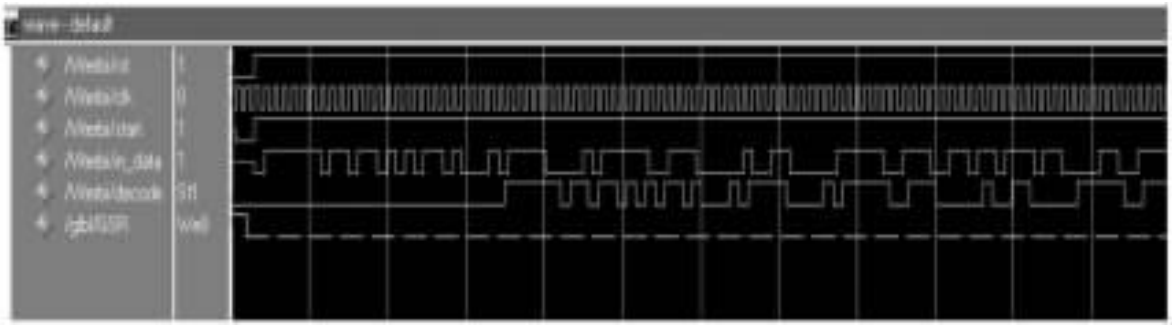

图 5 Viterbi 译码的时序仿真图

器中获得路径转移信息；

(4)重复步骤(3)，直到最后一级；

(5)输出译码结果。

从图 4 可以看出整个回溯的过程，这里假设路径度量值最大的是 s0。图 4 中的译码输出比特流为 0010110101

3.5 实现过程

从图 2 可以清楚看到，对输入的数据通过编码器进行卷积编码，到最后的输出译码结果，经历了以下几个过程：

(1)对输入的数据进行卷积编码，编码速率为 1/2，即每输入 1 个比特编码输出 2 个比特。

(2)将每次编码输出的 2 个比特量化为相应的数值，通过每一组数值计算出该组 4 个状态(s0，s1，s2，s3)的分支度量值，即 BM 值。

(3)进行加比选(ACS)运算，同时保存路径信息。首先在 0 时刻给 4 个状态(s0，s1，s2，s3)赋初始路径向量值(PM)：假如起始点为状态 s0，则状态 s0 的初始路径向量值为 PM0=100(该数值根据实际的情况来定，如回溯深度和分支度量值等，以便计算)，状态 s1、状态 s2、状态 s3 的初始路径向量赋值为 PM1=PM2=PM3=0。

(4)ACS 过程。因为到达每一个状态有两条路径(如图 3)，例如到达状态 s0(00)的两条路径分别是 s0(00)和 s1(01)，从中选出到达 s0 路径度量值最大的一条路径作为幸存路径。如图 2，若从 0 时刻到 1 时刻：BM0=−8，BM1=0，max{PM0+BM0，PM1+BM1}=PM0+BM0=92，所以 1 时刻到达状态 s0 的保留路径为 0 时刻从状态 s0 来的路径，从而更新 1 时刻 s0 的 PM0=92；同时由于 1 时刻到达 s0 的是“0”路径，所以保存的该时刻 s0 的路径信息是 0(若是“1”路径，则保存的该时刻 s0 的路径信息为 1)。以此类推，可求出该时刻到达状态 s1、s2、s3 的幸存路径，存储该路径信息，更新其路径度量值 PM。

(5)输出判决(OD)，即回溯过程，就是根据回溯深度以及 ACS 过程中所保存的 PM 值和幸存路径信息进行相应的算法回溯出译码结果。

4 时序仿真结果

使用 Verilog 语言在 ISE[5]中进行综合和实现，布线后的时序仿真图如图 5。图 5 为正确的维特比译码时序仿真图，输入的比特序列为一串随机数，经过卷积编码后输入到 Viterbi 译码器，最后输出的译码序列与输入序列一致。本译码器实现了正确的译码功能。

5 结果分析

考虑回溯深度对译码错误性能的影响，采用软判决译码，回溯深度为 9、18、45，测试帧数为 3 000，仿真结果如图 6。从仿真结果可以看出，回溯深度对 BER 有影响。回溯深度越深，性能越好。

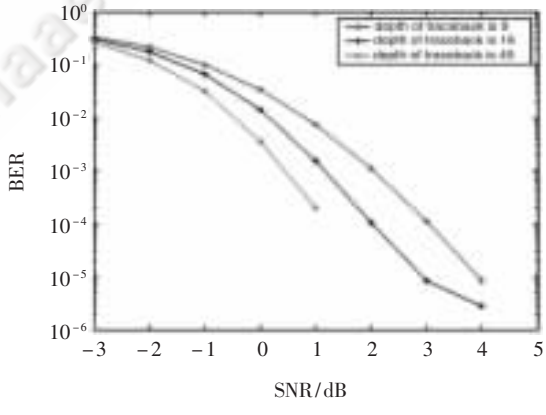

图 6 回溯深度对 Viterbi 的性能影响

参考文献

[1] 3GPP TS 36.201 v8.1.0：LTE Physical Layer−General Description(Release 8)[S].2008(5)：7−8.

[2] 郑宇驰，周晓方，闵昊.OFDM 系统中 Viterbi 译码器的设计及 FPGA 验证[J].复旦大学学报，2005，44(6)：923−927.

[3] 张宗橙.纠错编码原理和应用[M].西安：西安电子科技大学出版社，2001.

[4] Martin Röder and Raouf Hamzaoui.Fast Tree−Trellis List Viterbi Decoding[J].IEEE Trans Communications，2006，53(3)：453−460.

[5] 朱明程.Xilinx 数字系统现场集成技术[M].南京：东南大学出版社，2001.

(收稿日期：2009−01−16)