文章编号：1672-2892(2010)02-0201-06

基于 FPGA 的 Turbo 码译码器的设计

李　霞，王正彦

(青岛大学 自动化工程学院，山东 青岛 266071)

摘　要：介绍了一种基于现场可编程门阵列(FPGA)的 Turbo 码译码器的完整的设计方案和设计结果，采用 Max-Log-MAP 译码算法，用 Verilog 语言编程，提出了正序运算和逆序运算同时进行，以及采用数组型存储器存储中间运算结果的方案，使译码速度得到提高。文中给出了 Turbo 码译码原理、Max-Log-MAP 算法分析、基于 FPGA 的设计方案及实现框图、算法时序图及速度分析、仿真波形图及性能分析，结果表明，该方案正确可行，译码/纠错正确无误，且译码速度快。

关键词：Turbo 码；现场可编程门阵列；Max-Log-MAP 算法；Verilog 语言

中图分类号：TN911.72　　**文献标识码：**A

Design of the Turbo decoder with FPGA

LI Xia，WANG Zheng-yan

(Department of Automation Engineering，Qingdao University，Qingdao Shandong 266071，China)

Abstract：The design proposal and result of a FPGA-based Turbo decoder is introduced. Using the Max-Log-MAP decoding algorithm and the Verilog language for programming, the positive sequence operation and the reverse order operation can be simultaneously carried, and the intermediate results can be stored in array-based memory, which has enabled the decoding speed to be improved. The Turbo decoding theory, the Max-Log-MAP algorithmic analysis, the block diagram based on FPGA design and implementation, algorithm timing diagram and velocity analysis, are also presented. Results indicates that the program is correct and feasible, decoding/error correction is unmistakable, and the decoding speed is quick.

Key words：Turbo code；FPGA；Max-Log-MAP algorithm；Verilog

自 1993 年 Turbo 码提出，就一直是通信领域的研究热点。Turbo 码由 2 个递归循环卷积码(Recursive Systematic Convolutional code，RSC)通过交织器以并行级联的方式结合而成，是一种理论性能接近香农极限的纠错码。Turbo 码编码器是由 2 个反馈的系统卷积编码器通过 1 个交织器并行连接而成，编码后的校验位经过删余矩阵，产生不同码率的码字[1]，这些码字和校验序列复接后发送出去。Turbo 码的译码则较为复杂，本文首先介绍 Turbo 码的译码原理及相关的算法，接着介绍了译码器的 FPGA 实现框架，提出了正序运算和逆序运算同时进行的方案，以此来提高译码速度，最后对 FPGA 实现的译码器的功能进行验证并对性能进行了简要分析。

1　Turbo 码的译码原理和相应译码算法的公式

相对于由 2 个分量编码器构成的并行级联卷积码(Parallel Concatenated Convolutional Code，PCCC)的 Turbo 码译码结构如图 1。

2 个分量译码器 decoder1 和 decoder2 通过互相交换外信息，进行迭代译码，从而提高译码性能。decoder1 对信息序列{*Ys*}、校

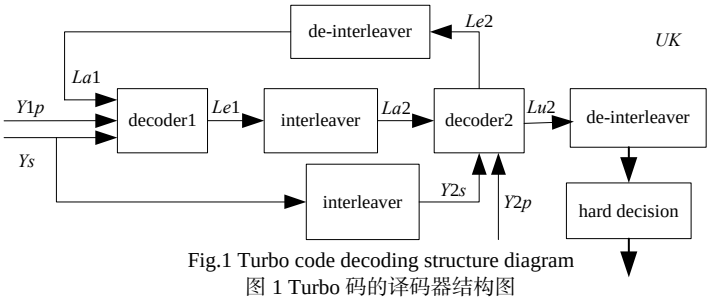

Fig.1 Turbo code decoding structure diagram
图 1 Turbo 码的译码器结构图

收稿日期：2009-08-10；修回日期：2009-10-26

验序列{$Y1p$}和先验信息序列{$La1$}(其中先验信息序列{$La1$}是由前一次 decoder2 生成的外信息序列{$Le2$}经过解交织(de-interleaving)而得的，第一次迭代时 decoder1 的外信息 $La1$ 为 0)进行译码运算，产生外信息序列{$Le1$}，经过交织器(interleaver)后得到 decoder2 的先验信息序列{$La2$}，连同校验序列{$Y2p$}和由信息序列{Ys}经交织所得的序列{$Y2s$}，一起送入 decoder2 进行译码运算，得到外信息序列{$Le2$}和对数似然比序列{$Lu2$}。其中，外信息序列{$Le2$}经过解交织得到 decoder1 下一次迭代的先验信息序列{$La1$}，至此完成 1 次迭代译码过程。当循环迭代译码达到一定迭代次数时，停止迭代运算，decoder2 的对数似然比序列{$Lu2$}经解交织后送硬判决(hard decision)，输出译码结果序列{UK}，从而完成整个译码过程。

Turbo 译码器的译码算法有 MAP 算法、Log-MAP 算法和Max-Log-MAP算法，其中Max-Log-MAP算法因为计算量小，是硬件实现 Turbo 码译码的最常用算法。在Max-Log-MAP译码算法中，需要计算分支转移概率 D_k、前向递推值 A_k、后向递推值 B_k、软输出似然比 $L(u_k)$ 和外信息 $Le(u_k)$。在编码采用生成矩阵为(7,5)的 RSC 码的情况下，分支转移概率 D_k 有 4 种值，计算公式为式(1)；前向和后向状态值的计算公式也有 4 个，分别为式(2)和式(3)；软输出似然比 $L(u_k)$ 和外信息 $Le(u_k)$ 计算公式为式(4)和式(5)。

$$\begin{cases} D_k^{00}(s',s)=(Lc/2)(y_k^s+y_k^p) \\ D_k^{01}(s',s)=(Lc/2)(y_k^s-y_k^p) \\ D_k^{10}(s',s)=-La(u_k)-(Lc/2)(y_k^s-y_k^p) \\ D_k^{11}(s',s)=-La(u_k)-(Lc/2)(y_k^s+y_k^p) \end{cases} \tag{1}$$

$$\begin{cases} A_k(0)=\min(A_{k-1}(0)+D_k^{00}(s',s),A_{k-1}(2)+D_k^{11}(s',s)) \\ A_k(1)=\min(A_{k-1}(2)+D_k^{00}(s',s),A_{k-1}(0)+D_k^{11}(s',s)) \\ A_k(2)=\min(A_{k-1}(3)+D_k^{01}(s',s),A_{k-1}(1)+D_k^{10}(s',s)) \\ A_k(3)=\min(A_{k-1}(1)+D_k^{01}(s',s),A_{k-1}(3)+D_k^{10}(s',s)) \end{cases} \tag{2}$$

$$\begin{cases} B_k(0)=\min(B_{k+1})(0)+D_{k+1}^{00}(s',s),B_{k+1}(1)+D_{k+1}^{11}(s',s)) \\ B_k(1)=\min(B_{k+1})(3)+D_{k+1}^{01}(s',s),B_{k+1}(2)+D_{k+1}^{10}(s',s)) \\ B_k(2)=\min(B_{k+1})(1)+D_{k+1}^{00}(s',s),B_{k+1}(0)+D_{k+1}^{11}(s',s)) \\ B_k(3)=\min(B_{k+1})(2)+D_{k+1}^{01}(s',s),B_{k+1}(3)+D_{k+1}^{10}(s',s)) \end{cases} \tag{3}$$

$$L(u_k)=\min_{u_k=0}(\tilde{A}_{k-1}(s')+D(s',s)+\tilde{B}_k(s))-\min_{u_k=1}(\tilde{A}_{k-1}(s')+D(s',s)+\tilde{B}_k(s)) \tag{4}$$

$$Le(u_k)=L(u_k)-La(u_k)-Lc\cdot y_k^s \tag{5}$$

式中：$A_k(s)$ 表示 k 时刻的 s 状态的前向递推项；$B_k(s)$ 表示 k 时刻的 s 状态的后向递推项；$D_k(s',s)$ 表示 k 时刻的分支转移概率；s' 表示 k–1 时刻状态；$\tilde{A}_k(s)$ 表示前向递推项的归一化公式；$\tilde{B}_k(s)$ 表示后向递推项的归一化公式；$La(u_k)$ 表示先验信息；Lc 为信道可靠值，表达式为：$Lc=4aE_b/N_0$ (等式中 a 为信道衰落因子，对于加性高斯白噪声信道 BPSK 调制而言，衰落因子 a=1；方差 $\sigma^2=N_0/2$；E_b 表示每一发送比特的能量)[2-3]。

2 基于 FPGA 的 Turbo 码译码器设计

由 Turbo 译码原理和译码算法可知，基于 FPGA 的译码器的设计关键在于Max-Log-MAP算法的实现、交织器设计与时序控制电路设计、硬件实现时性能指标追求速度和规模的平衡[4]。设计总体框图如图 2，与图 1 相比将交织器整合于 decoder1，解交织器整合于 decoder2。由信道传递过来的序列解复接后分别存放于信息序列存储器(message sequence)、校验序列存储器(check sequence)中，送 Max-Log-MAP 算法模块进行运算；Max-Log-MAP 算法模块计算结果送交织器(interleaver)或者是解交织(de-interleaver)；decoder1 和 decoder2 的时序配合由时序控制单元(timing control unit)完成。

2.1 Max-Log-MAP 算法实现

由 Max-Log-MAP 算法可得：分支转移概率不存在递推计算，所以用组合电路实现即可。而前向递推值和后向递推值都属于递推计算，所以采用时序电路实现。因为前向递推值从前往后递推，后向递推值从后往前递推，

为提高速度,将信息序列和校验序列同时按正序和逆序从存储器中取出,先验信息序列也同时按正序和逆序输入,在计算分支转移概率值的正序和逆序的同时也计算前向递推值和后向递推值。为保证前向和后向递推值在计算完后立即计算对数似然比和外信息,需要将计算结果存入寄存器备用。以子译码器 decoder1 为例,介绍具体的 FPGA 实现框图如图 3。

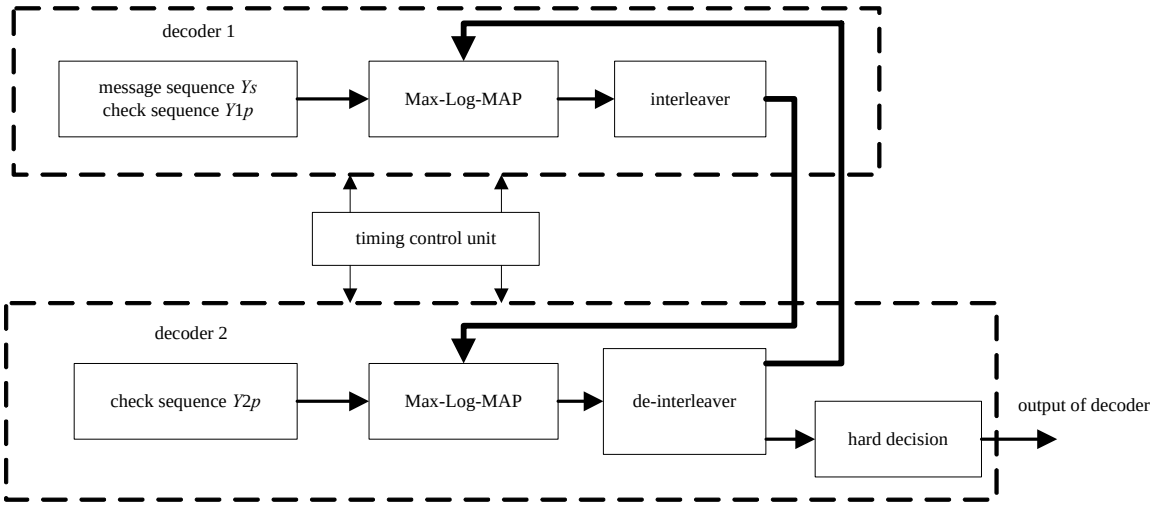

Fig.2 Total block diagram of Turbo code decoding based on FPGA

图2 基于FPGA的Turbo码译码器的总体框图

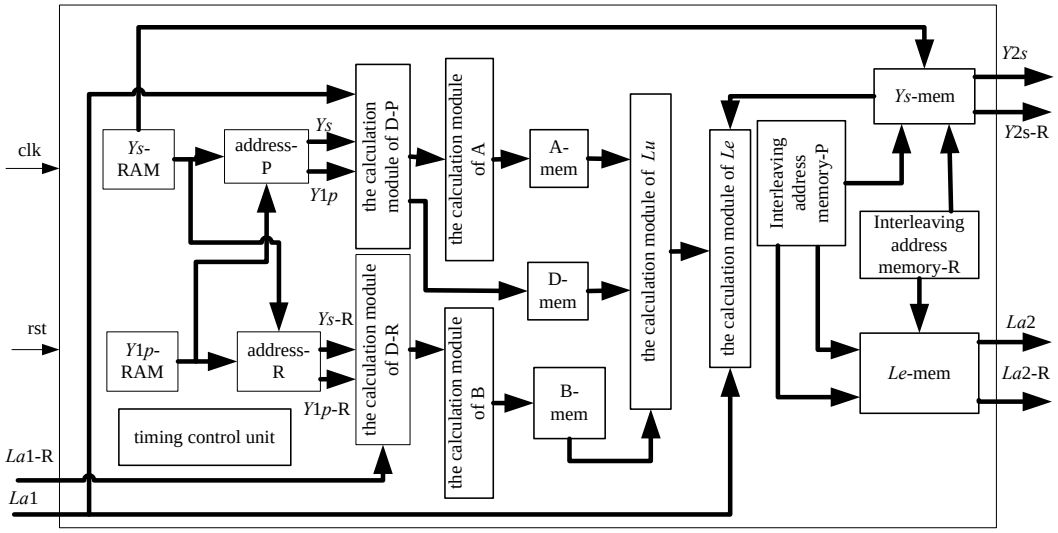

Fig.3 FPGA implementation block diagram of decoder1

图3 子译码器1的FPGA实现框图

decoder1 主要包括 2 个交织地址存储器 Interleaving address memory-P(正序)和 Interleaving address memory-R(逆序);2个地址控制模块address-P(正序)和address-R(逆序);2个定制存储器*Ys*-RAM和*Y*1*p*-RAM;2个分支转移概率计算模块the calculation module of D-P(正序)和the calculation module of D-R(逆序);前向递推计算模块the calculation module of A;后向递推计算模块the calculation module of B;对数似然比计算模块the calculation module of *Lu*;外信息计算模块the calculation module of *Le* 及其相应的存储器D-mem,A-mem,B-mem,*Le*-mem;信息序列存储器*Ys*-mem和时序控制模块timing control unit。

分支转移概率计算模块P,通过地址模块address-P顺序读取信息位*Ys*、校验位*Y*1*p*和先验信息位*La*1,计算出一帧数据的每一位所对应的4个分支转移概率值,然后直接送入前向递推计算模块,计算出前向递推的每一位所对应的4个状态值,并行存储于前向递推存储器A-mem;同时把P模块的结果并行存储于分支转移概率寄存器D-mem,以备计算软输出数似然比。

分支转移概率计算模块R通过地址模块address-R顺序读取信息位*Ys*-R、校验位*Y*1*p*-R和先验信息位*La*1-R,逆

序计算出一帧数据的每一位所对应的4个分支转移概率值，然后直接送入后向递推计算模块，并行计算后向递推每一位所对应的4个状态值，同时并行存储于后向递推存储器B-mem。

对数似然比计算模块，直接从D-mem,A-mem和B-mem中读取4个相应的状态值，计算对数似然比然后直接送入外信息的计算模块，与相应的信息位和先验信息位一起计算出外信息的值，同时并行存储于外信息寄存器*Le*-mem。

信息序列存储器*Ys*-mem按交织地址存储器P给出的地址输出decoder2所需的正序信息序列{*Y*2*s*}，按交织地址存储器R给出的地址输出decoder2所需的逆序信息序列{*Y*2*s*-R}，同时外信息寄存器Le-mem中的数据按交织地址存储器P给出的地址输出decoder2所需的正序先验信息序列{*La*2}，按交织地址存储器R给出的地址输出decoder2所需的逆序先验信息序列{*La*2-R}，至此decoder1完成1次迭代译码计算。

decoder2的Max-Log-MAP算法模块的实现与decoder1略有不同，在方框图上的表现为：把交织地址存储器变为解交织地址存储器；decoder2要有1个外信息存储器*L*-mem，但不需要信息序列存储器*Ys*-mem；decoder2有3路输出序列：*La*1,*La*1-R和*Lu*2。其中*La*1和*La*1-R直接送入decoder1，作为先验信息的正序和逆序，*Lu*2则是从*L*-mem中按解交织地址读取的数据，直接送入硬判决器，经过硬判决输出判决结果*UK*。

2.2 交织器设计

交织器设计采用将待交织数据和交织图样分别存放于RAM和ROM中，按顺序地址将待交织数据存入RAM，读出时以交织地址读出，即将ROM的内容作为RAM的读地址，从而完成数据交织[5]。具体实现如图4。

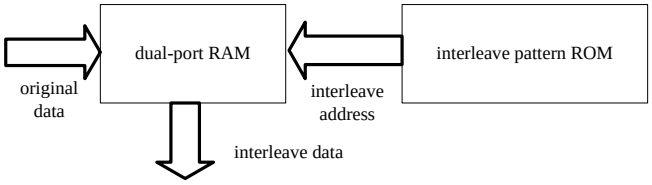

Fig.4 Interleaving process

图4 交织过程

2.3 时序控制

decoder1和decoder2是交替工作的，即decoder1输出先验信息*La*2和*La*2-R时，decoder2开始计算分支转移、前向和后向概率；decoder2输出先验信息*La*1和*La*1-R时，decoder1开始计算它的分支转移、前向和后向概率。具体时序关系如图5。

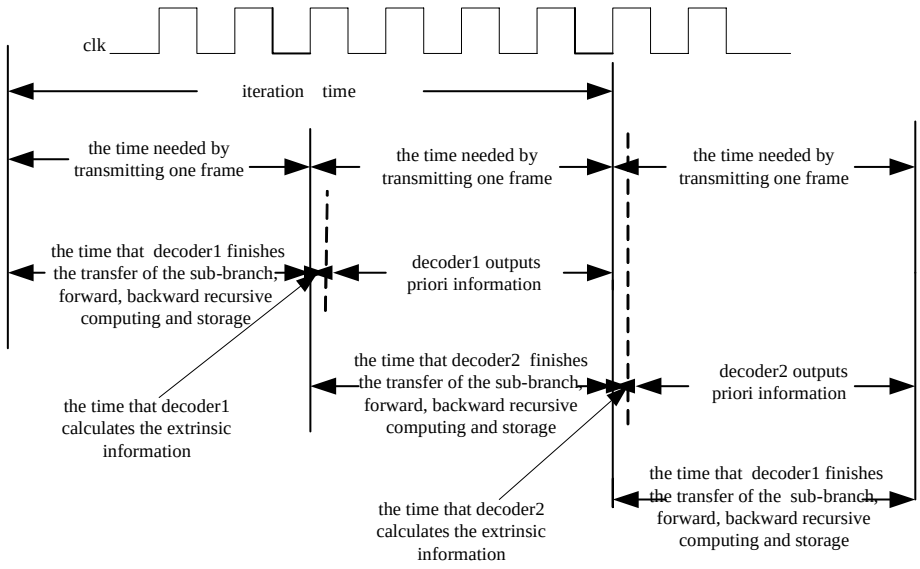

Fig.5 Diagram of sequential relationship

图5 时序关系图

2个译码器的交替工作控制采用计数器和使能控制完成。由于分支转移概率的运算采用正序和逆序同时进行方案，故一帧数据传输时间即可完成前向和后向概率计算，并将其存储于数组型存储器mem。而软输出似然比和外信息的计算，是从mem中取出数据并行完成的。故decoder1和decoder2完成1次迭代所需要的时间是一帧数据传输时间的2倍，提高了译码速度。由于采用了较多数组型存储器保存中间运算结果，使得FPGA的硬件资源占用较多，因此该方案适合于选用大容量FPGA芯片。

3　仿真结果

选用Altera公司的StratixII芯片系列作为目标器件，用Verilog语言描述了整个设计过程，并用Quartus II 6.1软件完成了整个设计的综合和时序仿真[6-7]。为简化仿真，帧长采用5位(可扩展为128位等)，*Lc* 设为2，并假设编码部分无删余处理和归零处理(因此译码部分递推运算初值采用等概率计算)；交织方案采用简单的奇偶交织，数据量化采用9位定点数，1位符号位，5位整数，3位小数；迭代次数为3次。

仿真方案 1：假设传输无误码产生。设编码部分的信息码为{1,1,0,0,1}，经过生成矩阵为(7,5)的 RSC 编码产生校验序列 1 为{1,0,0,1,0}、校验序列 2 为{1,1,0,0,1}，经复接，BPSK 调制，高斯白噪声信道，在接收端量化，解复接，得到的信息序列为{*Ys*}={8,8,–8,–8,8}，校验序列 1 为{*Y*1*p*}={8,–8,–8,8,–8}，校验序列 2 为{*Y*2*p*}={8,8,–8,–8,8}，仿真结果如图 6 所示。

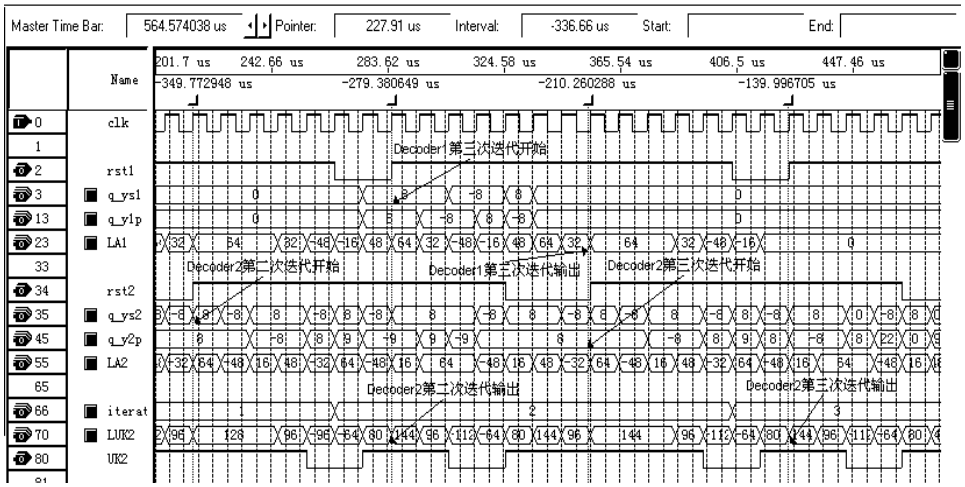

Fig.6 Simulation waveform of no error code by three times iteration
图 6 无误码经 3 次迭代的仿真波形图

图 6 中 q_ys1,q_y1p,LA1 分别为 decoder1 所需的信息序列、校验序列 1 和先验信息；q_ys2,q_y2p,LA2 分别为 decoder2 所需的信息序列、校验序列 2 和先验信息；rst1 和 rst2 为时序控制信号；*LUK*2 为 decoder2 输出的对数似然比解交织后的数据，*UK*2 为硬判决输出。由图 6 可以看出，3 次迭代后经过判决器输出的判决结果 *UK*2={1,1,0,0,1}，这与所发送的信息序列相同，证明经过 3 次迭代，用 Verilog 语言实现的 Turbo 码译码器的 Max-Log-MAP 算法判决输出正确。

仿真方案 2：假设传输过程中有噪声并有误码产生。设编码部分的信息码仍为{1,1,0,0,1}，但接收到的序列分别为：{*Ys*}={2,–5,–3,–6,4},{*Y*1*p*}={6,–6,–9,12,–4},{*Y*2*p*}={4,7,–5,–10,–3}[8]，其中信息序列中的–5 和校验序列 2 中的–3 为错码。仿真波形如图 7 所示。由图 7 可看出经过 3 次迭代后经判决器输出的判决结果 *UK*2={1,1,0,0,1}，与发送的信息序列相同，表明经过 3 次迭代，该译码器纠错正确。

2 次仿真的结果都和用 C 程序计算结果相同，表明结果正确。FPGA 的硬件资源占用情况如表 1 所示。

本文设计的译码器可从以下几方面改进，以得到更好的译码性能：帧长扩展为 128 位或更多，选择更具实用性能的交织方案，增加迭代次数，数据量化采用更多位数。

表 1 Turbo 码译码器逻辑资源占用表
Table1 Logic resource consumption of Turbo decoder

device utilization summary		
device:EP2S60F1020C3		
dedicated logic registers	1 851/48 352	4%
combinational ALUTs	2 793/48 352	6%
logic utilization		7%

4　结论

本文通过 2 组存储器同时计算 1 个子译码器的分支转移概率的正序和逆序，同时完成的还有此译码器的前向递推和后向递推的计算及相应值的存储；通过读取存储的分支转移概率、前向递推和后向递推值使对数似然比和外信息同时计算并存储，此时另外 1 个子译码器的分支转移概率、前向递推和后向递推值也可以计算存储，这样 2 个子译码器交替工作，使译码速度得以提高。虽然这要求存储单元增多，但是就目前大容量 FPGA 的发展来看，

这一点并不难实现。时序控制也比较简单，仿真结果正确，表明此方案可行。

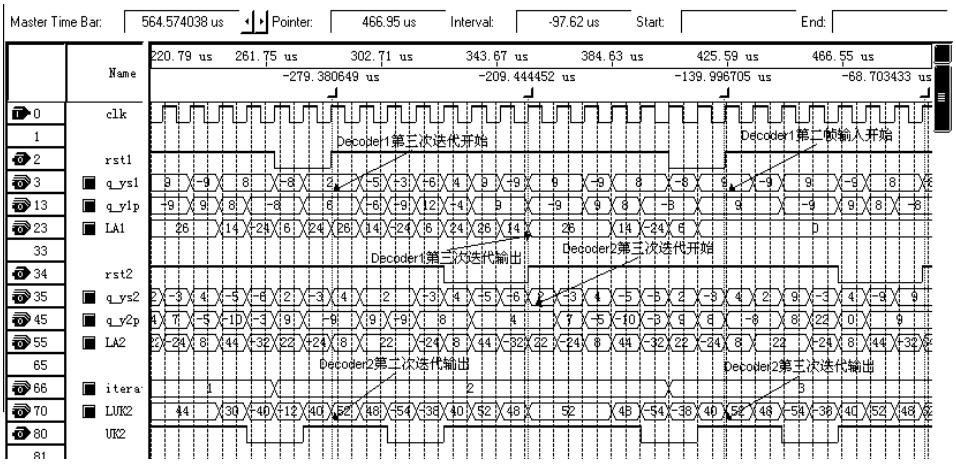

Fig.7 Simulation waveform after three times iteration with error code

图 7 有误码经 3 次迭代的仿真波形图

参考文献：

[1] Berron C,Glavicus A,Thitimaajshima P. Near Shannon limit error-correcting coding and decoding:Turbo-codes(1)[J]. Proc. ICC´, 1993,2(1):1064–1074.

[2] 杨海芬. Turbo 码编译码方法研究与实现[D]. 成都:西南交通大学, 2003.

[3] 王新梅,肖国镇. 纠错码–原理与方法[M]. 西安:西安电子科技大学出版社, 2001.

[4] Sharm S,Attri S,Chauhan F C. A Simplified and Efficient Implementation of FPGA-Based Turbo Decoder[C]// Proc. IEEE. Conference on Performance, Computing and Communications. 2003:207–213.

[5] 赵旦峰,雷李云. 基于 FPGA 的 Turbo 译码交织器设计[J]. 信息与电子工程, 2007,5(3):186–189.

[6] 褚振勇,翁木云. FPGA 设计应用[M]. 西安:西安电子科技大学出版社, 2002.

[7] 夏宇闻. Verilog 数字系统设计[M]. 北京:北京航空航天大学出版社, 2003.

[8] 刘东华. Turbo 码原理与应用技术[M]. 北京:电子工业出版社, 2004.

作者简介：

李 霞(1982–)，女，山东菏泽人，在读硕士研究生，研究方向为信号与信息处理.email: franky1006@sohu.com.

王正彦(1970–)，女，江苏沛县人，硕士，教授，主要研究方向为信号与信息处理、FPGA 技术应用.