文章编号：2095-4980(2015)05-0722-07

LTE 链路中 Turbo 编译码原理及 FPGA 实现

李燕斌[1]，杨　杨[2]

(1.中国电子科技集团公司 第 10 研究所，四川 成都 610054；2.电子科技大学 通信与信息工程学院，四川 成都 610054)

摘　要：介绍了 Turbo 码的编译码原理和常用的译码算法。使用 Altera DE4 开发板对 Turbo 编译码进行硬件实现，给出了实现中功能模块的详细划分情况。本次实现的 Turbo 码的相关参数取自 3GPP 的长期演进(LTE)协议，针对 LTE 中不同场景下码长不同的情况，对硬件采取了可配置参数的设计，符合 LTE 链路中任意情况下的 Turbo 码的规范。最后搭建了验证平台，将硬件得到的数据和仿真的结果进行对比，验证了实现的正确性。

关键词：长期演进；Turbo 码；Log-MAP 算法；现场可编程门阵列

中图分类号：TN927　　**文献标识码：**A　　**doi：**10.11805/TKYDA201505.0722

FPGA implementation for Turbo encoding and decoding in LTE link

LI Yanbin[1], YANG Yang[2]

(1.The 10th Research Institute of CETC, Chengdu Sichuan 610054, China; 2. School of Communication and Information Engineering, University of Electronic Science and Technology of China, Chengdu Sichuan 610054, China)

Abstract: The principle of encoding and decoding for Turbo code and the Log-MAP decoding algorithm are introduced. The encoding and decoding processes are implemented on the Altera DE4 development board and the parameters of the encoder and decoder are based on the 3GPP Long Term Evolution(LTE) standard, which can work with any code length in LTE physical link. Finally, a simulation platform is built. The data from hardware implementation and simulation are compared to verify the feasibility of this implementation.

Key words: Long Term Evolution; Turbo; Log-MAP; Field Programmable Gate Array

Turbo 码又称为并行级联卷积码，它将卷积码和随机交织器巧妙地结合在一起。在实现随机编码思想的同时，通过交织实现了由短码构造长码的方法，并由软迭代译码来逼近最大似然译码，从而使 Turbo 码性能逼近了香农限。正是因为 Turbo 码具有这样优良的性能，它在很多应用中已成为首选的编码技术。这些应用包括遥测通道编码(Consultative Committee for Space Data Systems，CCSDS)，全球微波互联接入(WiMAX)和长期演进(LTE)。

本文给出了 Turbo 码的系统结构，详细分析了编译码原理，并给出了 Log-MAP 译码算法。然后用 FPGA 实现了针对 LTE 链路的 Turbo 码的编译码过程[1]。

1　Turbo 编译码原理

1.1 Turbo 编码原理

Turbo 码的编码器主要由分量码编码器、交织器、删余器和复用器构成，如图 1 所示。

分量码一般选择为递归系统卷积码，也可以选择分组码和非系统卷积码。

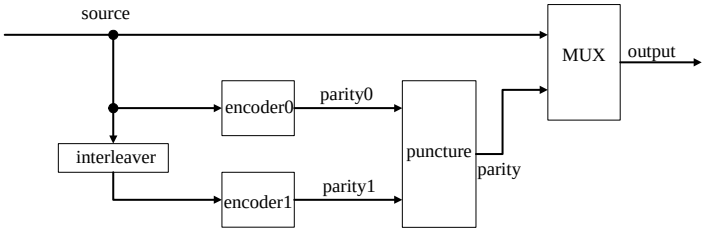

Fig.1 Block diagram of Turbo encoder
图 1 Turbo 编码器结构框图

交织器是 Turbo 码的一个关键部分，通过对输入信息序列的映射，对信息序列的比特位置重新排序，改变输出码字的重量。通过交织使得各分量码编码序列的低相关性能够快速收敛迭代译码。

收稿日期：2014-03-03；修回日期：2014-03-23

删余器通过周期性地删除分量码的校验位提高 Turbo 码编码码率，经过信道传输后在译码端通过复用器再次复接成一个完整的码字。

Turbo 码编码流程如下：

1) 信息序列(source)输入到第 1 个分量码编码器(encoder0)，编码输出校验序列(parity0)；

2) 信息序列(source)经过交织器(interleaver)交织后输入到分量码编码器(encoder1)，编码输出校验序列(parity1)；

3) 对校验序列 parity0 和 parity1 删余(puncture)后，与信息序列通过复用器(MUX)复接后生成一个 Turbo 码字。

经过上面 3 个步骤便完成了 Turbo 码的编码过程，交织器的选择以及删余操作的算法由具体的应用场景确定。

1.2 Turbo 译码原理

Turbo 码获得优异性能的根本原因之一是采用了迭代译码，通过分量译码器之间软信息的交换来提高译码性能。1 个由 2 个分量码构成的 Turbo 码译码器是由 2 个与分量码编码器对应的译码单元和交织器与解交织器组成的，将 1 个译码单元的软输出信息作为下一个译码单元的输入，为了获得更好的译码性能，将此过程迭代数次。这就是 Turbo 码译码器的基本工作原理。Turbo 码译码器框图如图 2 所示。

Turbo 码的译码流程如下：

1) 信道接收序列解复用后得到消息序列(source)和校验序列 parity0 和 parity1。

2) 消息序列(source)、先验信息(AP0)和相应校验序列(parity0)输入到分量译码器(decoder0)，译码器输出软判决对数似然比(Log Likelihood Ratio，LLR0)，经过运算后得到外部软信息(EX0)，外部软信息经过交织器(interleaver)交织后输入到分量译码器(decoder1)，作为分量译码器的先验信息(AP1)。

3) 消息序列(source)，经过交织器(interleaver)交织后和相应的校验序列、先验信息(AP1)和校验序列(parity1)输入到分量译码器(decoder1)，译码器输出软判决对数似然比(LLR1)，经过运算后得到外部软信息(EX1)，外部软信息经过解交织器(deinterleaver)解交织后输入到分量译码器(decoder0)，作为分量译码器的先验信息(AP0)。

4) 继续执行步骤 2),3)直到迭代次数结束，分量译码器(decoder1)输出的最后一次对数似然比硬判决作为译码输出。

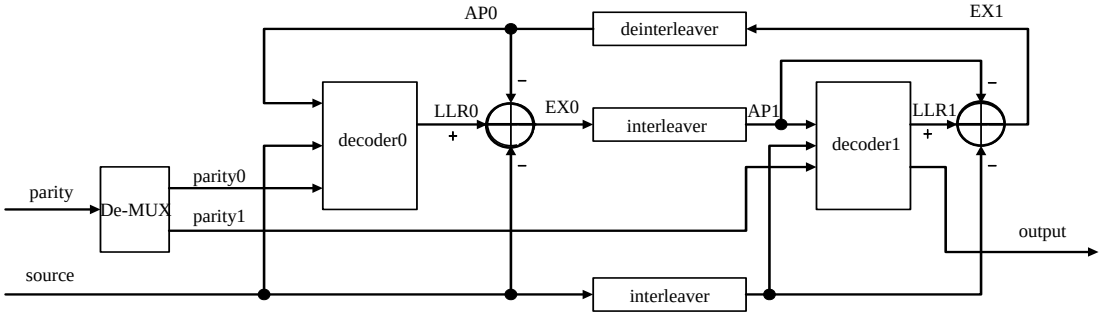

Fig.2 Block diagram of Turbo decoder

图 2 Turbo 译码器结构框图

2　Turbo 译码算法

Turbo 译码算法是决定 Turbo 码性能的关键算法。本小节着重介绍几种常用的软译码算法，以此作为硬件实现的理论基础。

2.1 MAP 译码器

最大后验概率(Maximum a Posteriori，MAP)译码算法也称为 BCJR(Bahl Cocke Jelinek and Raviv)算法[2]，它将原有的最小化网格图路径的差错概率的译码准则改为了最小化比特差错概率。比特后验 LLR 的定义为：

$$\Lambda\left(m\left|y\right.\right)=\ln\frac{p\left(m=+1\left|y\right.\right)}{p\left(m=-1\left|y\right.\right)} \tag{1}$$

式中 y 是译码器的输入序列。接下来会用双极性的表示方法来表示比特序列，即：$m\in\left\{+1,-1\right\}$，由此得到的 LLR 代表了比特 m 的二进制值，LLR 的幅度代表了这个双极性比特值的置信度。LLR 的幅度值越大，对应的比特判

决的置信度越大。因为 $p\left(m=+1\left|y\right.\right)+p\left(m=-1\left|y\right.\right)=1$，由比特 LLR 计算相应比特概率的公式为：

$$p\left(m\left|y\right.\right)=\frac{\mathrm{e}^{(1+m)\Lambda\left(m\left|y\right.\right)/2}}{1+\mathrm{e}^{\Lambda\left(m\left|y\right.\right)}},\quad m=+1,-1 \tag{2}$$

MAP 译码器主要功能就是计算后验 LLR $\Lambda\left(m\left|y\right.\right)$，在进行 MAP 译码时第 i 个比特的后验概率的计算公式为：

$$p\left(m_i=m\left|y\right.\right)=\frac{p\left(m_i=m\right)}{p\left(y\right)}=\frac{1}{p\left(y\right)}\sum_{\substack{S_{j2}\to S_{j1}\\ m_i=m}}\alpha_{i-1}\left(S_{j2}\right)\gamma_i\left(S_{j1},S_{j2}\right)\beta_i\left(S_{j1}\right) \tag{3}$$

式(3)中关键是将输入序列 y 根据第 i 比特分为 3 个部分：输入序列以前的部分、当前第 i 个译码阶段相关的部分以及将来的输入部分。然后应用贝叶斯准则，联合概率密度函数 $p\left(m_i=m\left|y\right.\right)$，式(3)可以分解为 3 个概率的乘积，即：$\alpha$，$\beta$ 和 γ，详细的推导可以参考文献[3]。

2.2 Log-MAP 译码器

实际应用中，MAP 译码器的计算都是在对数域进行的，这种译码器称为 Log-MAP 译码器[4]。利用雅克比对数式计算 2 个指数之和的对数，计算公式如下：

$$\ln\left(\mathrm{e}^{\tilde{x}_1}+\mathrm{e}^{\tilde{x}_2}\right)=\max\left(\tilde{x}_1,\tilde{x}_2\right)+\ln\left(1+\mathrm{e}^{-\left|\tilde{x}_1-\tilde{x}_2\right|}\right)=g\left(\tilde{x}_1,\tilde{x}_2\right) \tag{4}$$

由此可知，倘若将前向、后向以及转移概率用式(5)计算，则前面的乘法计算都可以用加法来代替：

$$\tilde{\alpha}=\ln\left(\alpha\right),\tilde{\beta}=\ln\left(\beta\right),\tilde{\gamma}=\ln\left(\gamma\right) \tag{5}$$

需要注意的是，转移概率的对数值计算可以进一步分解为：

$$\tilde{\gamma}_i\left(S_{j1},S_{j2}\right)=\ln\left(p\left(m_i\right)p\left(y_i\left|c\right.\right)\right)=\ln\left(p\left(m_i\right)\right)+\ln\left(p\left(y_i\left|c\right.\right)\right) \tag{6}$$

2.3 Max-Log-MAP 译码器

Max-Log-MAP 译码器就是将对数译码器的算法用式(7)进行近似计算：

$$\ln\left(\mathrm{e}^{\tilde{x}_1}+\mathrm{e}^{\tilde{x}_2}\right)\approx\max\left(\tilde{x}_1,\tilde{x}_2\right) \tag{7}$$

和式(4)相比较，差别只有 $\ln\left(1+\mathrm{e}^{-\left|\tilde{x}_1-\tilde{x}_2\right|}\right)$项被去掉了，因此 $\ln\left(1+\mathrm{e}^{-\left|\tilde{x}_1-\tilde{x}_2\right|}\right)$这项也被称为修正项，加上这一项，Max-Log-MAP 译码器就可以取得和 Log-MAP 相同的性能。

类似地可以推导出前向、后向概率的近似对数值：

$$\begin{cases}\tilde{\alpha}_i\left(S_{j1}\right)=\max\limits_{S_{j2}}\left(\tilde{\gamma}_i\left(S_{j1},S_{j2}\right)+\tilde{\alpha}_{i-1}\left(S_{j2}\right)\right)\\ \tilde{\beta}_{i-1}\left(S_{j2}\right)=\max\limits_{S_{j1}}\left(\tilde{\gamma}_i\left(S_{j1},S_{j2}\right)+\tilde{\beta}_{i-1}\left(S_{j1}\right)\right)\end{cases} \tag{8}$$

最大对数近似也可以计算后验 LLR，即：

$$\Lambda_i\left(m\right)\approx\max_{\substack{S_{j2}\to S_{j1}\\ m_i=+1}}\left(\tilde{\alpha}_{i-1}\left(S_{j2}\right)+\tilde{\gamma}_i\left(S_{j1},S_{j2}\right)+\tilde{\beta}_i\left(S_{j1}\right)\right)-\max_{\substack{S_{j2}\to S_{j1}\\ m_i=-1}}\left(\tilde{\alpha}_{i-1}\left(S_{j2}\right)+\tilde{\gamma}_i\left(S_{j1},S_{j2}\right)+\tilde{\beta}_i\left(S_{j1}\right)\right) \tag{9}$$

实际的应用中常常使用 Max-Log-MAP 译码器，因为这类译码器大大降低译码的复杂度且对性能的影响不是很大[5−6]。本次实现译码器使用的就是 Max-Log-MAP 算法。

3 Turbo 编译码器的硬件实现

3.1 LTE 中的 Turbo 码

3.1.1 编码器

LTE 协议中使用的 Turbo 编码方案是：并行级联卷积编码(Parallel Concatenated Convolutional Code，PCCC)[7−8]使用了 2 个 8 状态子编码器和 1 个 Turbo 码内交织器。Turbo 编码器的码率为 1/3，结构如图 3 所示。PCCC 中 8 状态子编码器的传输函数为：

$$G(D)=\left[1,\frac{g_1(D)}{g_0(D)}\right] \tag{10}$$

式中 $g_0(D)=1+D^2+D^3$，$g_1(D)=1+D+D^3$。

输入 Turbo 编码器的比特表示为 $c_0,c_1,\cdots,c_{k-1}$，第 1 个和第 2 个 8 状态子编码器的输出比特分别为 $z_0,z_1,\cdots,z_{k-1}$ 和 $z'_0,z'_1,\cdots,z'_{k-1}$。从交织器的输出比特表示为 $c'_0,c'_1,\cdots,c'_{k-1}$，这些比特将输入第 2 个 8 状态子编码器。

开始进行编码时，8 状态子编码器中移位寄存器的初始值为 0。Turbo 编码器输出为：

$$d_k^{(0)}=x_k,d_k^{(1)}=z_k,d_k^{(2)}=z'_k \tag{11}$$

式中 $k=0,1,\cdots,k-1$。

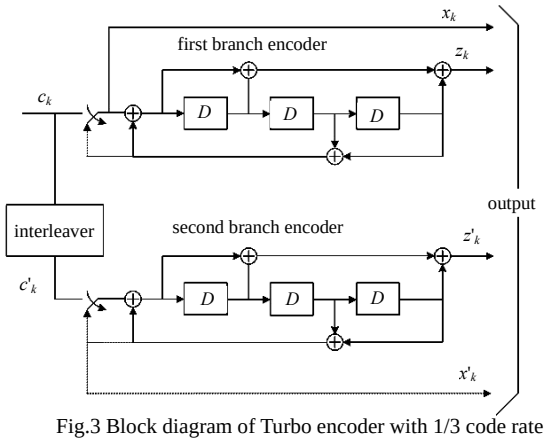

Fig.3 Block diagram of Turbo encoder with 1/3 code rate
图 3 码率为 1/3 的 Turbo 编码器结构

3.1.2Turbo 编码器的迫零处理

Turbo 码编码器对数据进行编码之后，6 个寄存器的状态往往处于不定态，为此常常需要对编码器进行迫零处理，使得下一次的编码其真实状态能够是一个确定的状态。LTE 中迫零处理是通过从所有信息比特编码之后的移位寄存器反馈中获取尾比特来完成的，迫零过程中输出的尾比特在信息比特编码之后添加。

前 3 个尾编码用于终止第 1 个编码器(图 3 中上面的那一个开关处于低端位置),此时第 2 个子编码器被禁用。最后 3 个尾比特用于终止第 2 个子编码器(图 3 中下面的那一个开关处于低端位置),此时第 1 个子编码器被禁用。则用于 Trellis Termination[7]的传输比特为：

$$\begin{cases} d_k^{(0)}=x_k,d_{k+1}^{(0)}=z_{k+1},d_{k+2}^{(0)}=x'_k,d_{k+3}^{(0)}=z'_{k+1} \\ d_k^{(1)}=z_k,d_{k+1}^{(1)}=x_{k+2},d_{k+2}^{(1)}=z'_k,d_{k+3}^{(1)}=x'_{k+2} \\ d_k^{(2)}=x_{k+1},d_{k+1}^{(2)}=z_{k+2},d_{k+2}^{(2)}=x'_{k+1},d_{k+3}^{(2)}=z'_{k+2} \end{cases} \tag{12}$$

3.2 Turbo 编码器的硬件实现

虽然 Turbo 编码的实现可以直接采用图 3 的硬件结构，但由于每一位编码的比特，交织后对应的位置不一样，导致交织器只能逐位进行，而不能多位进行。因此，为了提高并行度，考虑在交织部分复用 8 路交织存储器取址。因为交织表是一样的，为了减小存储空间，采用双口 ROM 来进行交织。则只需要 4 个双口 ROM。而取址的 RAM 读写的地址不一样，输入(8 位)输出(1 位)位宽不一样，只能采用双口 RAM，且需要 8 个双口 RAM，另外系统位和校验位 1 可以共用一个双口 RAM。本设计对尾码处理时将 12 位尾码全部输出。图 4 是 Turbo 编码器的实现框图，主要由主控单元、双口 RAM、交织器、ROM、RSC 等模块构成，各模块的功能如下。

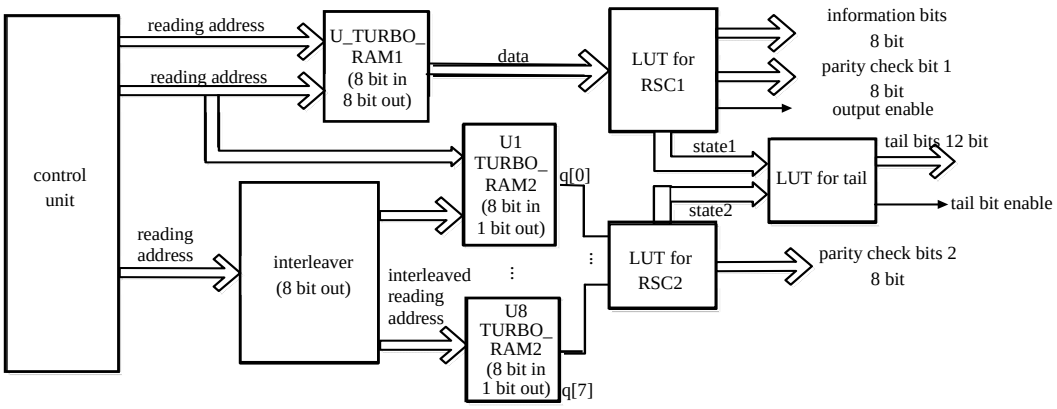

Fig.4 Block diagram of Turbo encoder implementation
图 4 Turbo 编码实现结构框图

主控单元：生成读写使能信号(Wr_rdy)，写地址信号(Addr_w)和交织器开始交织的一个高脉冲使能信号。实际设计中使用同步计数器生成顺序写地址信号，写数据结束，产生一个高脉冲信号。

双口 RAM：U_TURBO_RAM1 作为系统数据缓存，保证与交织后的数据同步进入 U_RSC2(Recursive System Code)中编码；8 个 TURBO_RAM2 保证交织地址能即时读到待编码 8 位比特中每一位交织后的地址。合并成 8 位后送到 U_RSC2 中编码。

交织器：将 LTE 中 QPP 交织公式，进行简单的变形，得到迭代公式。按照 188 种帧长规模将迭代公式的初值制表。另外为了保证与后端 8 位交织取址相匹配，在单个地址交织模块之后添加一个输入 8 位、输出 104 位的双口 RAM，将输出的 104 位分成 8 个交织地址输出。

RSC：RSC 的实现是通过查表法，为了提高速度，采用 8 路并行处理。输入 8 位数据与编码状态作为查表地址，输出 8 位与输出后编码状态作为查表输出，输出的状态下一时钟反馈给输入，即可连续编码。相应的状态和数据表可用 Matlab 生成为 mif 文件初始化 ROM，然后设计时直接调用 Altera 的 IP 核。

并行输出：将 RAM1,RSC1,RSC2 作为 3 路输出，尾码为 12 bit 单独输出，并有 1 个尾码输出使能信号；其中 RSC1,RSC2 输出最后的编码状态到尾码表 Tail，通过 case 语句选择即可得到尾码。

设计中用到的查找表是将初始状态和输入合并作为输入地址，输出为下一状态和输出的合并值，令 Cur_state =Next_state 即可实现连续赋值；尾表只需读取 2 个 RSC 最后状态即可得到输出，本设计中一次将尾码 12 位全部输出。

3.3 Turbo 译码器的硬件实现

3.3.1 结构框图

译码算法采用的是 Max-Log-MAP 算法，通过多次迭代实现译码，硬件设计框图如图 5 所示。

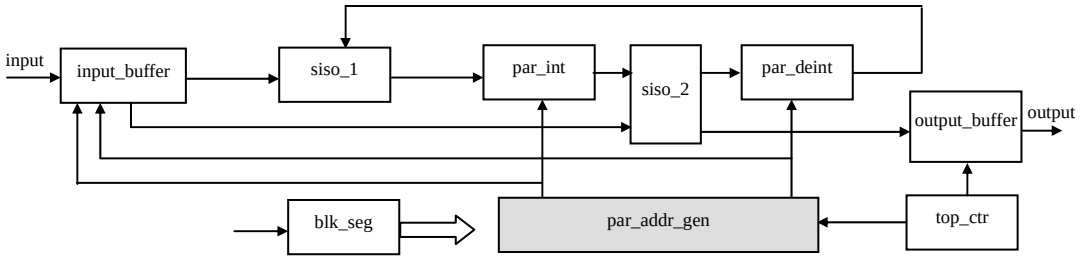

Fig.5 Realization block diagram of Turbo decoder
图 5 Turbo 译码实现结构框图

input_buffer 模块：对输入的数据进行缓存，并且将数据分割成 8 组分别存储在 8 片 RAM 中；同时产生一些反压信号 input_rdy 和数据存储结束、后一级可读取数据信号 rd_val。

siso 模块：Turbo 译码器中最核心的计算部分。这里 siso 模块由 8 个相同的 Max-Log-MAP 模块来实现，8 个 Max-Log-MAP 模块分别完成一个并行度的 Turbo 译码的计算，并且相互交换信息以提高译码的准确性。

par_addr_gen 模块：产生 siso 模块计算所需要数据的读地址和相应的使能信号，并且配合 swt_net_int 和 swt_net_deint 完成并行交织和并行解交织。

par_int 模块:将 siso_1 模块产生的外部信息顺序储存起来,并交织读出传递给下一个 siso_2 译码器进行译码。

par_deint 模块:将 siso_2 模块产生的外部信息解交织储存起来,并顺序传递给下一个 siso_1 译码器进行译码。

top_ctr 模块：产生使能信号，控制 par_addr_gen 模块工作；同时控制迭代次数和输出数据的使能信号。

output_buffer 模块：对软信息进行硬判决，然后对比特信息进行解交织储存后串行输出。

blk_seg 模块:产生图中灰色模块所需要的码块分割的边界值 $k1$~$k8$ 和 top_ctr 模块所需要的最大迭代次数。

4 仿真及测试结果

为了验证硬件实现的正确性，设计构建了一个与 Matlab 仿真进行对比的验证平台，如图

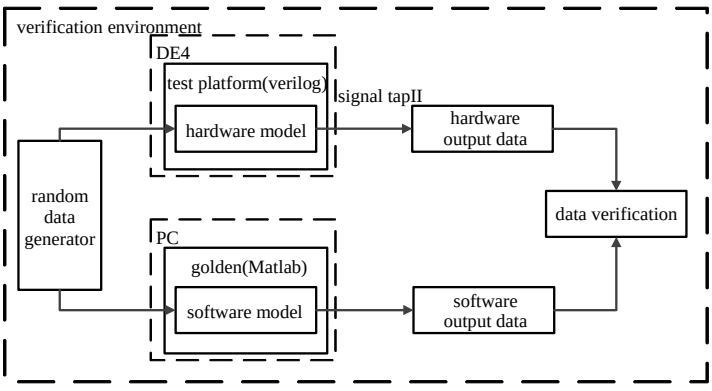

Fig.6 Schematic diagram of verification platform
图 6 验证平台示意图

6 所示。

利用 Matlab 随机生成一组测试向量，分别导入软件和硬件模型进行仿真，并将结果进行比对，验证设计的实现是否正确。针对 LTE 的应用需求，选取了 5 组数据帧长，即 frame_len 分别取 3 904,4 032,5 312,5 824,6 144，为了便于结果的观察，这里只给出了 frame_len 为 40 的情况。

经过 Matlab 软件仿真，得到的编码数据如图 7 所示。图 8 为 Turbo 编码模块下板测试，signal tapII 抓回的结果。

对译码模块采用经过软处理的 Turbo 编码模块的输出结果，下板测试，signal tapII 抓回的结果如图 9 所示。

由上面的结果可以看出本次设计实现的编译码器较好地完成了编译码的任务。

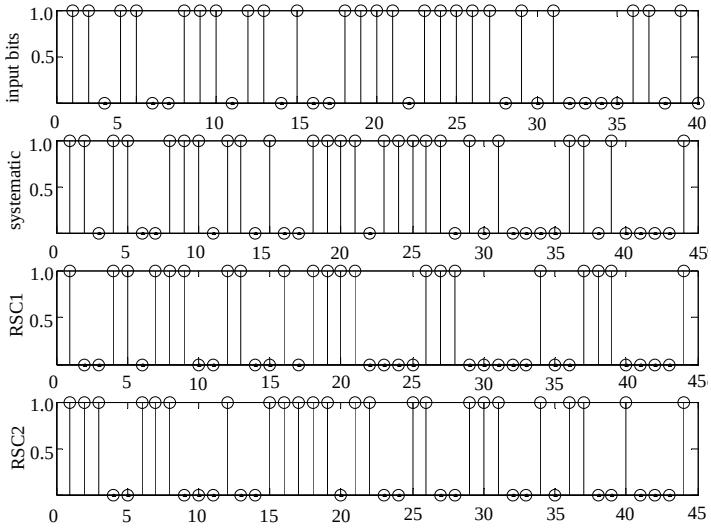

Fig.7 Simulation for Turbo encoder with Matlab
图 7 Turbo 编码 Matlab 仿真结果

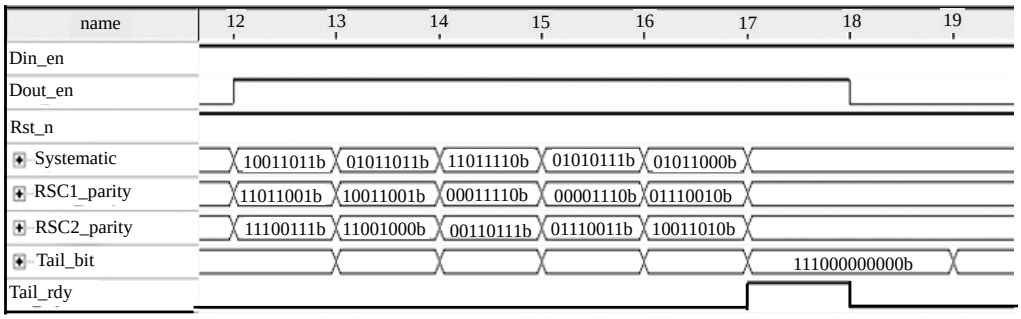

Fig.8 Output waveform of Turbo encoder with 40 bit/frame
图 8 Signal tap 抓取测试规模为 40 的一帧编码波形

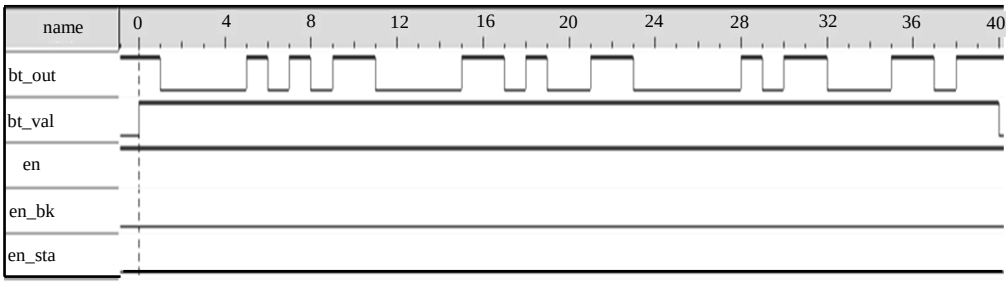

Fig.9 Output waveform for Turbo decoder with 40 bit/frame
图 9 Signal tap 抓取测试规模为 40 的一帧译码波形

5　结论

本文介绍了 Turbo 码的编译码原理，着重介绍了 Log-MAP 译码算法，并对 Turbo 编译码进行了 FPGA 硬件实现。实现的硬件平台是 Altera DE4 开发板，其中 Turbo 编码器的参数取自 3GPP 的 LTE 协议，针对 LTE 中不同码长的情况，对硬件采取了可配置参数的设计，符合 LTE 链路中任意情况下的 Turbo 码的规范；译码器采用的算法是基于迭代的 Log-MAP 算法，在算法性能和复杂度之间取得了较好的平衡。最后本文使用 signal tapII 和 Matlab 搭建了验证平台来验证硬件实现的结果。经过验证，本文实现的 Turbo 编译码器较好地完成了 LTE 链路所需的 Turbo 编译码的全部功能。

参考文献：

[1] 陈发堂,谢璘珂. LTE 系统中 Turbo 编译码仿真与性能分析[J]. 电视技术, 2011,35(7):81-84. (CHEN Fatang,XIE Linke. Simulation and performance analysis of LTE Turbo coding and decoding[J]. Video Engineering, 2011,35(7):81-84.)

[2] 叶懋. Turbo 码译码算法的研究[D]. 贵州:贵州大学, 2008. (YE Mao. The research of Turbo decoding algorithm[D]. Guizhou,China:Guizhou University, 2008.)

[3] Chiueh T D,Tsai P Y,Lai I W. Baseband receiver design for wireless MIMO-OFDM communications[M]. [S.l.]:Wiley-IEEE Press, 2012.

[4] 姚远,邱天爽. MAP 算法在 Turbo 码译码中的应用和研究进展[J]. 电讯技术, 2004(4):6-9. (YAO Yuan,QIU Tianshuang. Application and development of MAP algorithm in Turbo decoding[J]. Telecommunication Engineering, 2004(4):6-9.)

[5] 洪瑜. Turbo 码的编解码研究[D]. 浙江:浙江大学, 2006. (HONG Yu. The research of Turbo encoding and decoding[D]. Zhejiang,China:Zhejiang University, 2006.)

[6] 宋英杰. Turbo 高速编译码技术研究[J]. 现代导航, 2015,6(1):47-52. (SONG Yingjie. Research on high-speed Turbo encoding and decoding[J]. Modern Navigation, 2015,6(1):47-52.)

[7] 3GPP(2012)Evolved Universal Terrestrial Radio Access(E-UTRA). Multiplexing and channel coding[S]. TS36.212.V11.0. 2012.

[8] 3GPP(2012)Evolved Universal Terrestrial Radio Access(E-UTRA). Physical layer procedures[S]. TS36.213.V11.0. 2012.

作者简介：

李燕斌(1970–)，男，四川省营山县人，研究员，主要研究方向为航空通信.email:lyb_cd@163.com.

杨　杨(1989–)，女，安徽省寿县人，在读硕士研究生，主要研究方向为通信信号与信息处理.