doi:10.11835/j.issn.1000-582X.2014.06.007

基于 Verilog 的 LDPC 编译码设计

何 伟,郭志欢
(重庆大学 通信工程学院,400044)

摘 要:LDPC(low density parity check code)码是目前最优秀的码字之一,其接近香农传输极限的性能使其成为第四代通信系统(4G)强有力的竞争者。论文通过 Verilog 实现 LDPC 编译码算法从而提高运算效率,选用了"π 旋转矩阵构造法"进行编码,"皇后算法"较好的避免了 H 矩阵中小环的出现。译码采用"UMP BP-Based(最小和或最大积)算法",其中的对数运算将小数控制在了−100 到 100 之间,对于运算过程中的大量小数均采用 Q8(定点数)格式表示,范围 $-128 \leqslant X \leqslant 127.996\ 093\ 75$,精度 0.003 906 25,从而避免了浮点数运算,因此可以完全采用 Verilog 语言描述 LDPC 译码算法。程序中没有使用任何公司的 IP 核,适用于所有 FPGA,可移植性好。

关键词:LDPC;Verilog 实现;Q 格式;π 矩阵;UMP BP-Based 算法
中图分类号:TP309 **文献标志码**:A **文章编号**:1000-582X(2014)06-045-06

LDPC encoding and decoding design based on Verilog HDL

HE Wei, *GUO Zhihuan*
(School of Communication Engineering, Chongqing University, Chongqing 400044, China)

Abstract: LDPC (low density parity check code) is one of the most excellent codes, and it is a strong competitor of the fourth generation communication system (4G) as its performance is close to Shannon transmission limit. The paper realizes LDPC encoding and decoding algorithm through Verilog to improve computational efficiency, and selects π rotation matrix construction method to encode. And queen algorithm avoids the appearance of Cyclotella in H matrix. The UMP BP-based algorithm is also used in decoding and all of decimal fractions are limited between −100 and 100. For a large number of decimal fractions, the Q8 (fixed-point) format is adopted to express them and the range is between −128 and 127.996 093 75 with the precision of 0.003 906 25. Thus floating-point arithmetic can be avoided and Verilog language can be used to describe the LDPC decoding algorithm. The program does not use any IP core, which means it can be applied to all FPGAs and has good transplantation. It has high application value in engineering.

Key words: LDPC; Verilog realization; format of Q; π matrix; UMP BP-Based algorithm

LDPC 码是美国 Gallager 教授在 1962 年提出来的一种线性分组码(n,k),码长为 n,信息序列长度为 k,可由校验矩阵 $\boldsymbol{H}$ 唯一定义,$\boldsymbol{H}$ 矩阵为稀疏矩阵。理论研究表明,1/2 码率的 LDPC 码在 BPSK 调制下的性能距离香农极限仅差 0.004 5 dB。但是,由于 LDPC 迭代译码算法具有较大的时间复杂度和空间复杂度,为了提高算法的执行效率,笔者采用 Verilog 语言以 LDPC(64,32)为例描叙了整个编译码算法,最终通过 FPGA 综合成了硬件电路,在成本上有所降低,在速度上有了很大的提高。

收稿日期:2013-12-02
基金项目:重庆市科委自然科学基金资助项目(CSTC2011BB2048);国家级大学生创新项目。
作者简介:何伟(1964-),男,教授,博士,主要从事信号与信息处理方向研究,(Tel)18651802606;(E-mail)715695317@qq.com。

1 LDPC 编码

1.1 H 矩阵简介

H 矩阵对 LDPC 的性能有很大的影响，一个好的 H 矩阵要求：1）行重、列重远小于码长；2）H 矩阵中不能出现 4 环，即任意 4 个“1”相连不能是长方形；3）线性无关的列数尽量大。构造 H 矩阵的方法很多，如 Gallager 构造法[1]、单位矩阵循环移位法[1]、近似下三角矩阵构造法[2]等，本文采用 π 旋转矩阵构造法[3]，π 矩阵具有“任何一行、列以及正反对角线上均只有一个‘1’”的特性，一般采用“皇后算法”[6]实现，将此矩阵称为 π_A，对 π_A 顺时针旋转90°、180°、270°可以得到 π_B、π_C、π_D，如下所示

$$\pi_A=\begin{pmatrix}1&0&0\\0&0&1\\0&1&0\end{pmatrix},\pi_B=\begin{pmatrix}0&0&1\\1&0&0\\0&1&0\end{pmatrix},\pi_C=\begin{pmatrix}0&1&0\\1&0&0\\0&0&1\end{pmatrix},\pi_D=\begin{pmatrix}0&1&0\\0&0&1\\1&0&0\end{pmatrix},$$

由 π 矩阵构造出来的 H 具有$[H^d\vdots H^p]$形式，其中 H^d 是由 π 矩阵排列组合得到，H^p 即为双对角线矩阵。

$$H^d=\begin{pmatrix}\pi_A&\pi_A&\pi_A&\pi_A\\\pi_D&\pi_A&\pi_B&\pi_C\\\pi_C&\pi_D&\pi_A&\pi_B\\\pi_B&\pi_C&\pi_D&\pi_A\end{pmatrix},\quad H^p=\begin{pmatrix}1&0&0&0&0&0\\1&1&0&0&0&0\\0&1&1&0&0&0\\0&0&1&1&0&0\\0&0&0&1&1&0\\0&0&0&0&1&1\end{pmatrix},$$

论文选用的 π 矩阵为 8×8，由此构成的 H 矩阵为 32×64。

1.2 LDPC 编码 Verilog 描述

对于信息序列 S^d 是已知的，设编码后的校验序列为 S^p，那么系统码 $S=[S^d\vdots S^p]$，由 $S\times H^{\mathrm{T}}=0$ 可以得出$[S^dS^p]\times[H^dH^p]^{\mathrm{T}}=S^dH^d+S^pH^p=0$，由于是二值数据，因此 $S^dH^d=S^pH^p$。根据 H^p 特性，很容易得出

$$S_0^p=\sum_i^k(S_i^d\cdot H_{0,i}^d),\tag{1}$$

$$S_j^p=S_{j-1}^p+\sum_i^k(S_i^d\cdot H_{j,i}^d),\tag{2}$$

k 为信息序列 S^d 的长度，之后的 S_1^p 到 S_{k-1}^p 均可由如下递推得到

计算出的校验序列 S^p 与信息序列 S^d 相组合及完成了整个编码。

下面以 LDPC(64,32)为例描述编码过程

Step1：初始化 H 矩阵，将事先通过 MATLAB 或者 VC 计算好的 H 矩阵存成 reg 型变量 HD[0]=32′h80808080、HD[1]=32′h20202020…，如果 H 矩阵规模比较大，可以考虑生成 mif 文件初始化 RAM，这样就需要调用 IP 核，两大 FPGA 厂家 Altera 和 Xilinx 都提供这种 IP 核；

Step2：当检测到 START 信号时，读入 DATA_IN 数据到 databuf，且 $i<=0$；

Step3：databuf 与 HD[i]相与（完成二进制乘法），所得结果再缩减异或（完成二进制累加），存入到 dataout[31−i]位，硬件语言习惯将数据从最高位存起，此值作为递推初始值 S_0^p；

Step4：$i<=i+1$；

Step5：databuf 继续与 HD[i]进行与、缩减异或操作得到 newbit，$S_i^p=S_{i-1}^p+$newbit；

Step6：如果 i 等于 32，CODE 端口输出 dataout 的值，程序结束，否则跳转到 Step4。

编码模块消耗逻辑单元 250 个，仿真波形如下

模块输入的为信息位，输出的为校验位，两者合并即为 64 比特的系统码。

2 LDPC 译码

LDPC 译码方法有 2 大类：硬判决译码和软判决译码。硬判决译码是当校验方程 $R\times H^{\mathrm{T}}=0$ 不成立时，根据接收的信息和有关规则，改变一些比特的值（0 或者 1），然后继续验证直到方程满足或达到最大迭代次数。该判决方法用硬件比较容易实现，但性能要比软判决差，因此，目前几乎所有的 LDPC 译码均采用软判

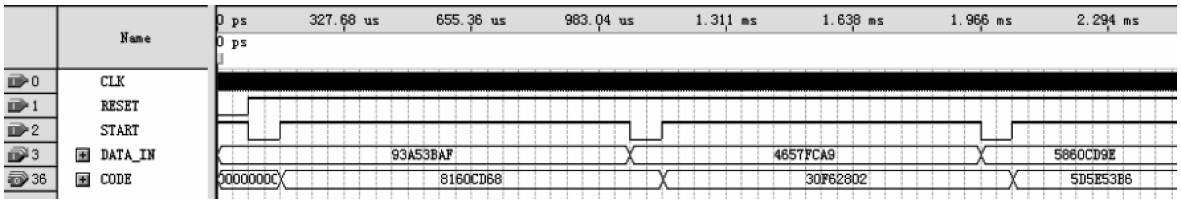

图 1　编码模块仿真波形

决方法,其中经典的算法有 BP(Belief propagation algorithm)译码算法[1]和 LLR BP 译码算法[1]。

2.1　BP 译码算法

BP 译码算法凭借优越的性能和合理的复杂度成为 LDPC 主流的译码算法,是一种建立在 Tanner 图上的译码方法。根据 $\boldsymbol{H}$ 矩阵可以绘制出表征 LDPC 码的 Tanner 图(如图 2)。

$$\boldsymbol{H}=\begin{pmatrix}1&0&1&1&0&0&1&0\\1&1&0&0&1&1&0&0\\0&1&1&0&1&0&0&1\\0&0&0&1&0&1&1&1\end{pmatrix},$$

以 Tanner 图为基础的 LDPC 译码主要包括两步:变量节点消息处理和校验节点消息处理。对于收到的数据,根据信道估值计算出其条件概率 $P(c/r)$;比如接收为 r,发送判断为 0 的概率即为 $P(c=0/r)$,发送判断为 1 的概率即为 $P(c=1/r)$,以此作为该变量节点的初始消息传向校验节点;校验节点将接收到的消息进行一定的运算处理后再传回变量节点;变量节点根据传回的数据连同初始信道估值数据一起运算后再传向校验节点。如此迭代下去直到变量节点满足校验方程或者达到最大迭代次数,判决输出的变量节点即为译码输出。

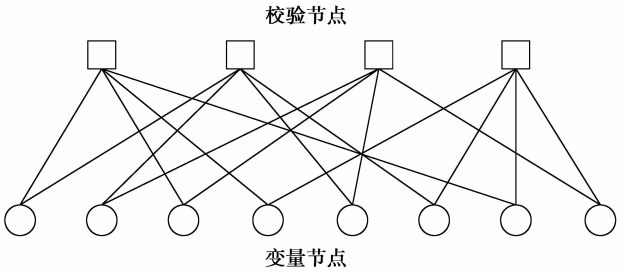

图 2　LDPC Tanner 图

为了便于具体描述,引入以下符号

$r_{ji}(b)$ (b=0,1):校验节点 j 传递给变量节点 i 的概率消息;

$q_{ij}(b)$ (b=0,1):变量节点 i 传递给校验节点 j 的概率消息;

$C(i)\backslash j$:除 j 外与变量节点 i 相连的校验节点的集合;

$R(j)\backslash i$:除 i 外与校验节点 j 相连的变量节点的集合。

Step 1:计算初始消息。

根据信道模型,计算接收消息的条件概率,以 AWGN(高斯白噪声信道)为例,若采用 BPSK 调制、信号源等概率分布的情况下,BP 译码的初始消息为

$$q_{ij}^{(0)}(0)=P_i(0)=\frac{1}{1+e-2y_i/\sigma^2}\quad q_{ij}^{(0)}(1)=P_i(1)=\frac{1}{1+e2y_i/\sigma^2},\tag{3}$$

收符号 $y_i=x_i+n_i$,x_i 为发送消息符号,n_i 为信道噪声,服从高斯分布。$P_i(0)$表示接收为 y_i 判断发送为 0 的概率,$P_i(1)$表示接收为 y_i 判断发送为 1 的概率。

Step 2:计算校验节点消息。

校验节点 j 的消息要排除 i 到 j 这条连线,即由其他与 j 相连的变量节点(非 i 点)来决定,由此可以避免错误的传递

$$r_{ji}^{(n)}(0)=\frac{1}{2}+\frac{1}{2}\prod_{i\in R(j)\backslash i}(1-2q_{ij\backslash i}^{(n-1)}(1)),\tag{4}$$

$$r_{ji}^{(n)}(1)=1-r_{ji}^{(n)}(0)=\frac{1}{2}-\frac{1}{2}\prod_{i\in R(j)\backslash i}(1-2q_{ij\backslash i}^{(n-1)}(1))。\tag{5}$$

Step 3:计算变量节点消息。

变量节点 i 的消息同样要排除 j 到 i 这条线,即由其他与 i 相连的校验节点(非 j 点)来决定。

$$q_{ij}^{(n)}(0) = K_{ij}P_i(0)\prod_{j\in C(i)\backslash j} r_{ji\backslash j}^{(n)}(0), \tag{6}$$

$$q_{ij}^{(n)}(1) = K_{ij}P_i(1)\prod_{j\in C(i)\backslash j} r_{ji\backslash j}^{(n)}(1)。 \tag{7}$$

K_{ij} 为校正因子,确保 $q_{ij}^{(n)}(0)+q_{ij}^{(n)}(1)=1$。

Step 4:判决检测。

判决类似于变量节点消息的处理,不同的是它必须将所有与变量节点 i 相连的校验节点的消息全部取出并运算

$$q_i^{(n)}(0) = K_iP_i(0)\prod_{j\in C(i)} r_{ji}^{(n)}(0), \tag{8}$$

$$q_i^{(n)}(1) = K_iP_i(1)\prod_{j\in C(i)} r_{ji}^{(n)}(1), \tag{9}$$

其中 K_i 为校正因子,确保 $q_i^{(n)}(0)+q(n)_i(1)=1$。若 $q_i^{(n)}(0)>q_i^{(n)}(1)$,则 $c_i=0$,否则 $c_i=1$。

Step 5:迭代过程。

若 $c\times \boldsymbol{H}^{\mathrm{T}}=0$ 或者达到最大迭代次数则结束,否则跳转至 Step 2 继续迭代。

2.2 LLR BP 译码算法

由于每一个节点的消息都是成对存在的,因此引入对数可以减少消息传递量,同时将乘法运算变成加法运算从而降低复杂度。LLR BP 译码算法步骤如下

Step 1:计算初始消息。

引入对数后,变量节点的初始消息为

$$L^{(0)}(P_i) = \ln\frac{q_{ij}^{(0)}(0)}{q_{ij}^{(0)}(1)} = \ln\frac{P_i(0)}{P_i(1)} = \frac{2y_i}{\sigma^2}, \tag{10}$$

σ 与 y_i 意义均与 BP 算法相同。

Step 2:计算校验节点消息;

根据式(4)、(5)可得

$$r_{ji}(0) - r_{ji}(1) = 1 - 2r_{ji}(1) = \prod_{i\in R(j)\backslash i}(1-2q_{ij\backslash i}(1)), \tag{11}$$

由恒等式 $\tan hx=\dfrac{e^x-e^{-x}}{e^x+e^{-x}}$,$\tan h\left(\dfrac{1}{2}\log(p_0/p_1)\right)=p_0-p_1=1-2p_1$,$(p_0+p_1=1)$可得

$$1-2r_{ji}(1) = \tanh\left(\frac{1}{2}\ln\left(\frac{r_{ji}(0)}{r_{ji}(1)}\right)\right) = \prod_{i\in R(j)\backslash i}\tanh\left(\frac{1}{2}\ln\left(\frac{q_{ij\backslash i}(0)}{q_{ij\backslash i}(1)}\right)\right),$$

令$L(r_{ji})=\ln\left(\dfrac{r_{ji}(0)}{r_{ji}(1)}\right)$,$L(q_{ij})=\ln\left(\dfrac{q_{ij}(0)}{q_{ij}(1)}\right)$,

则有

$$L^{(n)}(r_{ji}) = 2\tan h-1\left(\prod_{i\in R(j)\backslash i}\tan h\left(\frac{1}{2}L^{(n-1)}(q_{ij\backslash i})\right)\right)。 \tag{12}$$

Step 3:计算变量节点消息。

对式(6)、(7)进行对数处理如下

$$L^{(n)}(q_{ij}) = \ln\left(\frac{q_{ij}^{(n)}(0)}{q_{ij}^{(n)}(1)}\right) = L(P_i) + \sum_{j\in C(i)\backslash j} L^{(n)}(r_{ji\backslash j}), \tag{13}$$

Step 4:判决检测。

对式(8)、(9)进行对数处理如下

$$L^{(n)}(q_i) = L(P_i) + \sum_{j\in C(i)} L^{(n)}(r_{ji}), \tag{14}$$

若 $L^{(n)}(q_i)>0$,则 $c_i=0$,否则 $c_i=1$。

Step 5:迭代过程。

若 $c\times \boldsymbol{H}^{\mathrm{T}}=0$ 或者达到最大迭代次数则结束,否则跳转至 Step 2 继续迭代。

2.3 UMP BP 译码算法

三角函数的计算是十分耗时的,如何简化 LLR BP 算法的 Step 2 将是提速的关键,由于 $\tanh(x)$与 $\tanh-1(x)$均为奇函数,即:$\tanh(x)=\mathrm{sgn}(x)\cdot\tanh(|x|)$、$\tanh-1(x)=\mathrm{sgn}(x)\cdot\tanh-1(|x|)$,$\mathrm{sgn}(x)$为符号函数,取值±1 和 0;则式(12)变为

$$L^{(n)}(r_{ji})=2\Big(\prod_{i\in R(j)\backslash i}\mathrm{sgn}(L^{(n-1)}(q_{ij\backslash i}))\Big)\cdot\tanh-1\Big(\prod_{i\in R(j)\backslash i}\tanh\Big(\frac{1}{2}\,|\,L(n-1)(q_{ij\backslash i})\,|\Big)\Big),\tag{15}$$

由于 $0\leqslant\tanh(|x|)<1$ 且为单调递增函数,因此有下列约等式

$$\prod_{i\in R(j)\backslash i}\tanh\Big(\frac{1}{2}\,|\,L(n-1)(q_{ij\backslash i})\,|\Big)\approx\min_{i\in R(j)\backslash i}\tanh\Big(\frac{1}{2}\,|\,L(n-1)(q_{ij\backslash i})\,|\Big)=$$

$$\tanh\Big(\min_{i\in R(j)\backslash i}\frac{1}{2}\,|\,L(n-1)(q_{ij\backslash i})\,|\Big),$$

则式(15)可简化为

$$L^{(n)}(r_{ji})=\prod_{i\in R(j)\backslash i}\mathrm{sgn}(L^{(n-1)}(q_{ij\backslash i}))\cdot\min_{i\in R(j)\backslash i}|\,L(n-1)(q_{ij\backslash i})\,|。\tag{16}$$

UMP BP 算法主要是针对 LLR BP 算法的 Step2 进行优化,经过一定的近似,省去了三角函数的累乘计算,通过比较即可完成校验节点消息的更新,仅仅牺牲少量的译码准确率换取速度的极大提升。

2.4 LDPC 译码 Verilog 描述

经过前面的理论分析,可以利用式(10)、(13)、(14)、(16)进行译码,译码过程只涉及加法运算,复杂度极低,数据均为小数,数据源为 $2y_i/\sigma^2$,其中 $y_i=\pm1$,而 $0\leqslant\sigma\leqslant1$ 为 AWGN 信道表征值,取 $\sigma=0.45$(信道环境差,误码率 4.2%),初始消息为±9.88。

为了便于 Verilog 描述,这里引入 Q 格式[4](定点数),数据位宽 16 比特,对 LDPC(64,32)译码过程中的小数分析可得,其分布范围在−100～100 之间,因此选用 Q8 格式,即最高位为符号位,中间 7 比特为整数,最低 8 比特为小数,范围$-128\leqslant X\leqslant127.996\,093\,75$,精度 0.003 906 25。

Q8	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Q 格式与浮点数的互换公式为:$X_q=(\mathrm{int})(X_f\times2^Q)$,$X_f=(\mathrm{float})(X_q\times2^{-Q})$,这里 $Q=8$。运算中的负数均以补码存在(正数取反加 1),以初始值为例:$9.88\xrightarrow{\text{float2Q}}0\text{x}9\text{E}1$,$-9.88\xrightarrow{\text{float2Q}}0\text{xF}61\text{F}$,此后算法中的数据将全部以 Q8 格式参与运算,由于全是加法运算,因此不会出现其他 Q 格式。

由于 $\boldsymbol{H}(32\times64)$矩阵比较大,$L(q_{ij})$和$L(r_{ji})$均与 $\boldsymbol{H}$ 相关联,因此需要开辟较大的存储空间,并且采用一维数组表示二维数组,如 $q[i][j]=q[i\times\mathrm{COL}+j]$,COL 为矩阵列数,在 FPGA 中,开辟的 reg 类型的数组消耗的是逻辑资源 LET(logic elements),而并非其内部的 RAM(on chip memory),定义变量如下

```
module DECODE (CLK,START,CODE,DECODE);
  input CLK,START;
  input [63:0] CODE;
  output reg [63:0] DECODE;

  reg [15:0] q[2047:0];
  reg [15:0] next_q[2047:0];
  reg [15:0] r[2047:0];
  reg [63:0] HD [31:0];
  reg [15:0] ini_q[63:0];

  reg [7:0] i,j,k,state;
  reg [4:0] iter;
  reg [63:0] databuf;
  reg [15:0] min,sgn,r_abs;
  reg [63:0] r_buf;
  reg [31:0] q_buf;
  reg [15:0] q_sum,r_sum;
  reg [63:0] decodebuf,flag;
  reg flag_bit;
```

解决了数据表示难题后,即可按照 LLR BP 算法流程编写状态机程序,由于算法具有“串行”流水性,而

Verilog 语言是并行处理的，因此，用到了 25 种状态描述初始化 q_{ij}、计算 r_{ji}、更新 q_{ij}、判决、迭代的过程。虽然整个算法是顺序执行的，但每一状态的赋值语句均是并行操作的。译码模块运用了大量的数组，消耗逻辑资源 102 402 个，仿真波形如下：

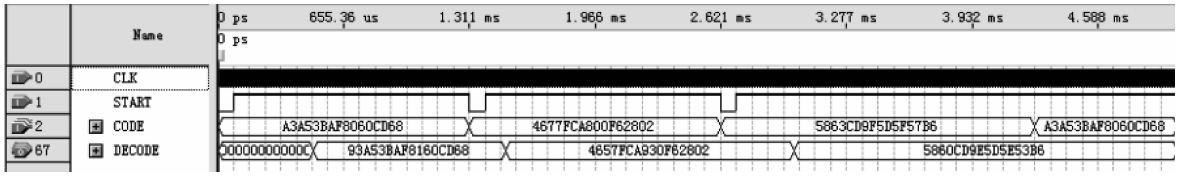

图 3 译码模块仿真波形

对照图 1 和图 3，输入模块的信息位和校验位合并后人为添加随机的错误，经过译码模块后可得到正确码字。

3 总 结

LDPC 编码过程比较简单，Verilog 的并行处理将乘累加运算缩短为 2 个 CLK 节拍，整个编码过程在 130 个 CLK 节拍完成。

LDPC 译码过程比较复杂，由编译报告就可以看出其消耗的资源巨大，而换来的益处就是译码速度的极大提升，以 LDPC(64,32)为例，25 个状态机在双重 for 循环的作用下进行消息的刷新和传递，迭代一次耗时 64×32×25＝51 200 个 CLK，最大迭代次数设为 10，一般出错 1 到 2 比特的情况下迭代 1 次即可正确译码，出错 3 到 4 比特数据源迭代 2 到 3 次可正确译码，超过 4 比特偶尔会不能正确译码，在完全正确译码的情况下平均迭代次数为 2，平均耗时 102 400 个 CLK，FPGA 系统时钟设为 100 M，经计算译码平均耗时 1.024 ms，数据位宽 64 bit，则速率为 64 Kbps。LLR BP 算法收敛较快，如果数据位宽成倍数增长，传输速率也将倍增，配合更高效的调制技术，速度可进一步提升，可满足高速通信系统的要求。

笔者提出了一种用 Verilog 语言实现复杂算法的思想，凡是标准 C 语言能实现的算法，在引入 Q 格式(定点数)后用 Verilog 语言也能将之实现，对于 C 语言中的开方、三角函数等可以用相应的 Q 格式算法或者查找表等方式来实现，甚至可以引入泰勒级数近似。用 Verilog 编写的硬件模块其运算效能将大大高于 C 语言算法。

参考文献：

[1] 李少谦. LDPC 码的原理与应用[M]. 北京：国防工业出版社，2006：129-138.

[2] 哈聪颖. LDPC 码的编码实现研究[D]. 哈尔滨：哈尔滨工业大学硕士学位论文，2006：21-25.

[3] 谢红梅. 迭代译码算法的研究[D]. 西安：西安电子科技大学硕士学位论文，2008：19-24.

[4] 王潞钢. DSP C2000 程序员高手进阶[M]. 北京：机械工业出版社，2005：33-38.

[5] 卢开澄，卢华明. 组合数学[M]. 北京：清华大学出版社，2002.

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

[7] Chen J H，Dholakia A，Eleftheriou E，et al. Reduced－complexity decoding of LDPC codes[J]. IEEE Transactions on Communications，2009，53(8)：1288-1299.

[8] Zhang J T，Fossorier M. Shuffled iterative decoding[J]. IEEE Transactions on Communications，2008，53(2)：209-213.

[9] Peterson W W，Weldon E J. Error correcting codes cambridge[M]. MA：MIT Press，2010.

[10] Tang H，Xu J，Kou Y，et al. On algebraic construction of Gallager and circulant low density parity check codes[J]. IEEE Transactions on Information Theory，2004，50(6)：1269-1279.

[11] Luby M G，Mitzenmacher M，Shokrollah M A，et al. Improved low-density parity-check codes using irregular graphs [J]. IEEE Transactions on Communication，2001，47(2)：585-598.

[12] Richardson T J，Shokrollahi M，Urbanke R. Design of capacity approaching irregular low-density parity-check codes[J]. IEEE Transactions on Information Theory，2001，47(2)：619-637.

(编辑 侯 湘)