4.6 Design and Applications of Turbo Codes

4.6.1 Design of Constituent Codes

Turbo 码的最大似然译码性能分析指出，Turbo 码在高信噪比下的性能主要由它的自由距离所决定。因为 Turbo 码的自由距离主要由重量为 2 的输入信息序列所产生的码字间的最小距离所决定，用本原多项式作为反馈连接多项式的分量编码器所产生的码字的最小重量为最大，因此当 Turbo 码交织器的大小给定后，如果分量码的反馈连接多项式采用本原多项式，则 Turbo 码的自由距离会增加，从而 Turbo 码在高信噪比下的“错误平层(error floor)”会降低。错误平层效应，指的是在中高信噪比下，误码率曲线变平；也就是说，即使是再增大信噪比，误码率也降不下来。使用表 4.1 所给码参数的 turbo 码性能仿真结果如图 4.30 所示。

表 4.1　16 状态 Turbo 码的两种配置

说　明	分量码 RSC1 $G = (g_0, g_1)_8$	分量码 RSC2 $G = (g_0, g_1)_8$
NP16-NP16 (非本原--非本原)	（37，21）	（37，21）
P16-P16　(本原--本原)	（23，35）	（23，35）

图中 turbo 码的码率为 1/2，迭代次数为 6 次，N 为交织长度。由图可见，在低信噪比区域，使用非本原反馈多项式的 turbo 码的性能要优于采用本原多项式的 turbo 码；而在高信噪比区域，却正好相反，本原 turbo 码具有很低的错误平层。因此，为了兼顾 turbo 码在两个区域的性能，一个分量码可以采用本原分量码，另一个为非本原分量码这种**非对称编码器结构**。

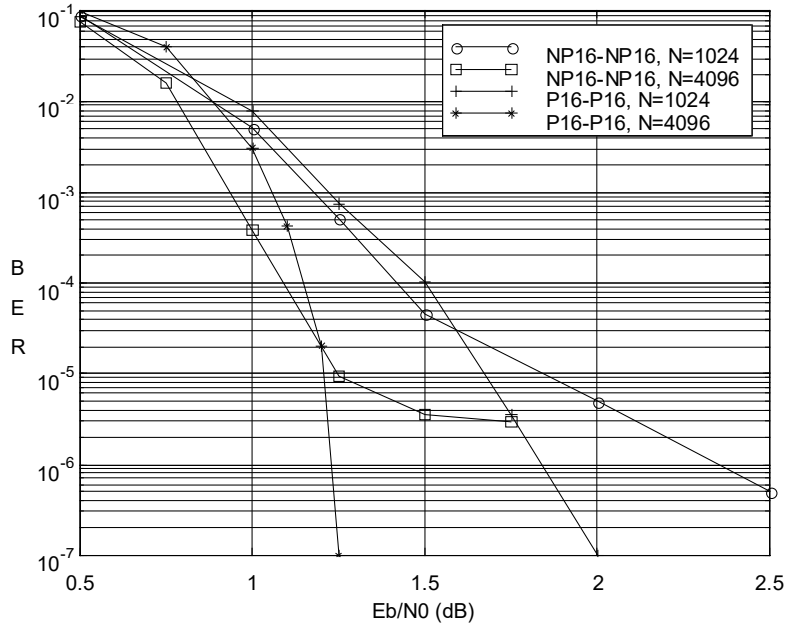


图 4.30　采用本原反馈多项式与非本原反馈多项式的 turbo 码性能的比较

为了得到高码率的 turbo 码，除采用删余技术外，还可以使用码率为 *k*/*n* (k>=2)的卷积码，如图 4.31 所示。表 4.2 列出了一些通过计算机搜索得到的性能优秀的卷积码 [Benedetto-Tcom96, no.5].

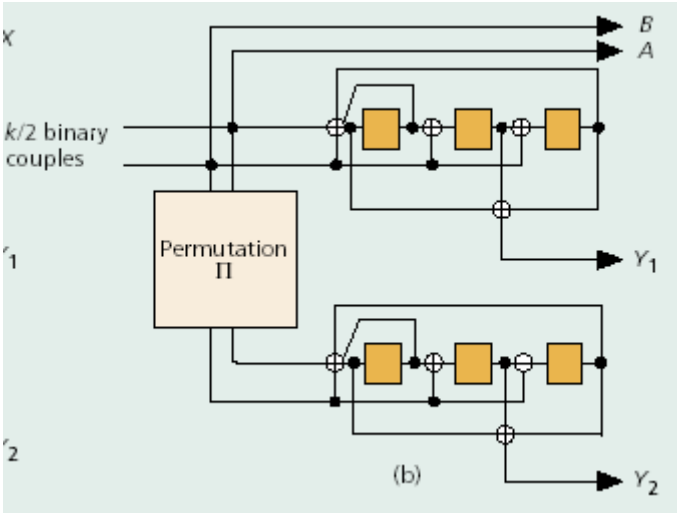


Fig. 4.31 8-state duobinary turbo encoder with polynomials (15, 13)

表 4.2

BEST RATE 1/2 CC FOR RATE 1/3 PCCC WITH INTERLEAVER LENGTH $N = 100$. THE POLYNOMIALS $d(D)$ AND $n(D)$ DEFINING THE FEEDBACK AND PARITY-CHECK CONNECTIONS OF THE ENCODERS ARE GIVEN IN OCTAL NOTATION WITH THE LEAST SIGNIFICANT BIT ON THE LEFT

ν	$d(D)$	$n(D)$	$d_{\text{free,eff}}$	d_{free}^{CP}	w_{free}^{CP}
1	3	2	4	4	2
2	7	5	10	7	3
3	15	17	14	8	4
4	31	33	22	9	5
	31	27	22	9	5
5	51	77	38	10	6
	51	67	38	12	4

- The rate of a turbo code without puncturing is

$$R = \frac{K}{K + \left(\frac{K}{R_1} - K\right) + \left(\frac{K}{R_2} - K\right)} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} - 1} = \frac{R_1 R_2}{R_1 + R_2 - R_1 R_2}$$

where K is the length of information bit sequence and R_i the code rate of the ith constituent code. For sake of simplicity, we can express it in the form

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} - 1$$

4.6.2 Trellis Termination

在 MAP 译码算法中，前向递推$\alpha_k(s)$与反向递推$\beta_k(s)$的初始值一般是根据分量编码器的初始状态和终止状态进行初始化的。对于普通的非系统卷积编码器，很容易将其初始状态和终止状态置为一已知状态，比如零状态。而对于 turbo 编码器，由于采用了递归结构，其分量编码器需要额外的结尾处理(trellis termination)才能达到终止于零状态，而且由于交织器的存在，将两个编码器同时归零就更为困难。为此，对$\alpha_k(s)$和$\beta_k(s)$的初始化一般采用表 4.3 所示的几种方法。

表 4.3　$\alpha_k(s)$和$\beta_k(s)$的初始化方法

	译码器 Dec1 的初始化条件	译码器 Dec2 的初始化条件
RSC1 和 RSC2 均不归零	$\alpha_0(0)=1,\alpha_0(s\neq 0)=0$ $\beta_N(s)=1/M, M=2^v$	$\alpha_0(0)=1,\alpha_0(s\neq 0)=0$ $\beta_N(s)=1/M, M=2^v$
RSC1 归零 RSC2 不归零	$\alpha_0(0)=1,\alpha_0(s\neq 0)=0$ $\beta_N(0)=1,\beta_N(s\neq 0)=0$	$\alpha_0(0)=1,\alpha_0(s\neq 0)=0$ $\beta_N(s)=1/M, M=2^v$
RSC1 和 RSC2 均归零	$\alpha_0(0)=1,\alpha_0(s\neq 0)=0$ $\beta_N(0)=1,\beta_N(s\neq 0)=0$	$\alpha_0(0)=1,\alpha_0(s\neq 0)=0$ $\beta_N(0)=1,\beta_N(s\neq 0)=0$

一种将两个分量编码器均归零的解决方案如图 4.32 所示。

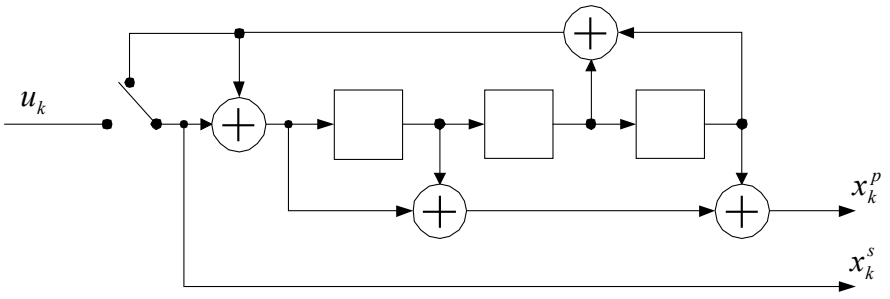


图 4.32　RSC 编码器的一种归零方法　G=(13,15)

4.6.3 Design of Interleavers

■　交织器的描述方法

交织器是一个单输入单输出设备，它的输入与输出符号序列有相同的字符集，只是各符号在输入与输出序列中的排列顺序不同。即它是整数 Z 上的置换：

$$\pi:\ \mathbb{Z}\to\mathbb{Z}:\ k'=\pi(k),\ k,k'\in\mathbb{Z}$$

目前，Turbo 码交织器有多种设计方法和具体实现形式，常用的有伪随机 S-交织器、分组交织器(block interleaver)、BG 非均匀交织器和 QPP 交织器等。

当分量编码器不归零时，从 MAP 算法的译码原理可知，分量译码器对一帧数据中的最后几比特译码的可信度较低，这样如果原帧数据中的最后几比特交织后仍处于帧数据的尾部，则整个 turbo 码性能的提高就受到了限制，我们称此为尾效应(tail effect)。为了避免这个问题，在交织器的设计中应该将原帧数据中的最后几比特置换到 RSC2 的输入序列的前部（非尾部位置）。当分量编码器均归零时，则不存在尾效应。

■ 伪随机 S-交织器

在伪随机S-交织器的设计中，每一个随机产生的置换位置$\pi(i)$均与它前边的S个值：

$\pi(i-1),\pi(i-2),\cdots,\pi(i-s)$进行比较，如果距离$|\pi(i)-\pi(i-j)|<s,\ \ j=1,2,\cdots s$，则$\pi(i)$被拒绝，必须重新产生。The definition in the previous paragraph translates into the following condition:

$$|\pi(i)-\pi(j)|\geq n_1 \text{ whenever } |i-j|<n_2$$

4.6.4 Examples of Applications

Turbo-like codes has already made their way into practical applications and standards. Typical examples include:

Application	**code**	**Termination**	**Polynomials**	**Rates**
CCSDS (deep space)	16-state	Tail bits	23, 33, 25, 37	1/6, 1/4, 1/3, 1/2
UMTS, cdma2000	8-state	Tail bits	13, 15, 17	1/4, 1/3, 1/2
DVB-RCS (for satellite)	duo-binary, 8-state	Circular	15, 13	1/3 up to 6/7
DVB-RCT (for terrestrial)	duo-binary,8-state	Circular	15, 13	1/2, 3/4
IEEE 802.16 (WiMax)	duo-binary, 8-state	Circular	15, 13	1/2 up to 7/8

(Note: “duo-binary” refers to a turbo code with rate-2/3 constituent code.)

■ The CCSDS Turbo Coding Standard

Given their outstanding performance, it is not surprising that turbo codes have been rapidly adopted for application in many commercial systems. The Consultative Committee for Space Data Systems (CCSDS) was among the first to develop a channel coding standard based on turbo codes. The CCSDS Telemetry Channel Coding standard [?] uses a turbo code with two component codes with selectable rates and blocklengths. The encoder for this code is shown in Figure 4.33. Two rate R = 1⁄4 recursive convolutional encoders are used to generate R = 1⁄2, 1⁄3, 1⁄4, and R = 1⁄6 turbo codes. The two switching matrices combine these rates, where a solid circle means every symbol is taken, and an open circle means every other symbol is taken. The feedback polynomial $h_0 = 1+D^3+D^4 = 23_8$ of both encoders is a primitive polynomial of degree 4 and the feedforward polynomials are $h_1 = 33$, $h_2 = 25$ and $h_3 = 37$. The various rates are achieved by using the connections shown on the right side of Figure 10.17. For example, to achieve R = 1⁄4 the four outputs are the systematic bit, the second and third parity bits from encoder 1 and the first parity bit from encoder 2.

The interleaver is a block permutation interleaver that has good spreading factors and, unlike the *S*-random interleavers, has an algorithmic implementation and is scalable. The CCSDS standard allows five interleaver lengths: $N1$ = 1784, $N2$ = 3568, $N3$ = 7136, $N4$ = 8920 and $N5$ = 16384. The first four block lengths are chosen to be compatible with an outer

Reed-Solomon code with interleaving depths of 1, 2, 4 and 5, respectively. The largest block length is allowed for those users requiring the most power efficiency and able to tolerate a larger decoder latency. The BER and frame error rate (FER) performance of the CCSDS turbo code with $N = 8920$ and various rates is shown in Figure 4.34.

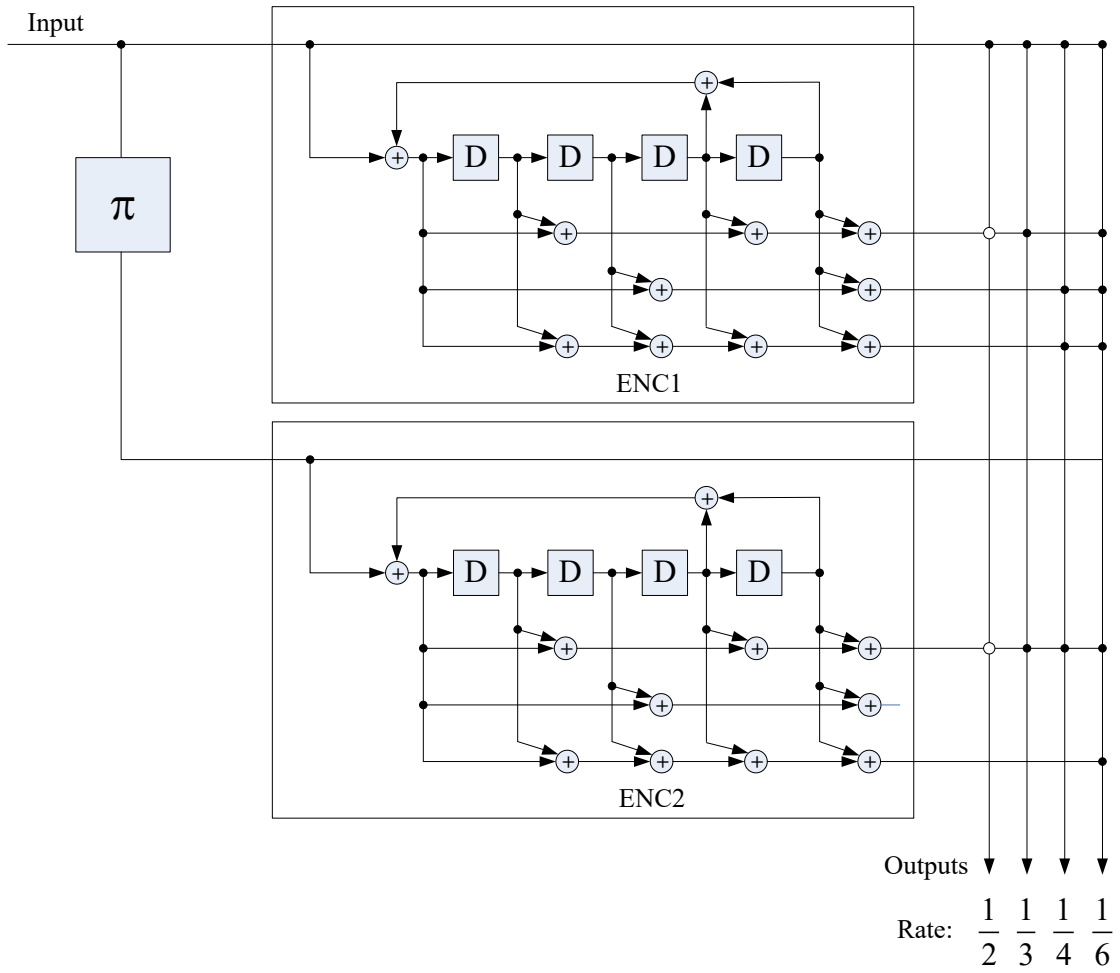


Figure 4.33: The CCSDS turbo encoder. A solid circle means every symbol is taken, and an open circle means every other symbol is taken.

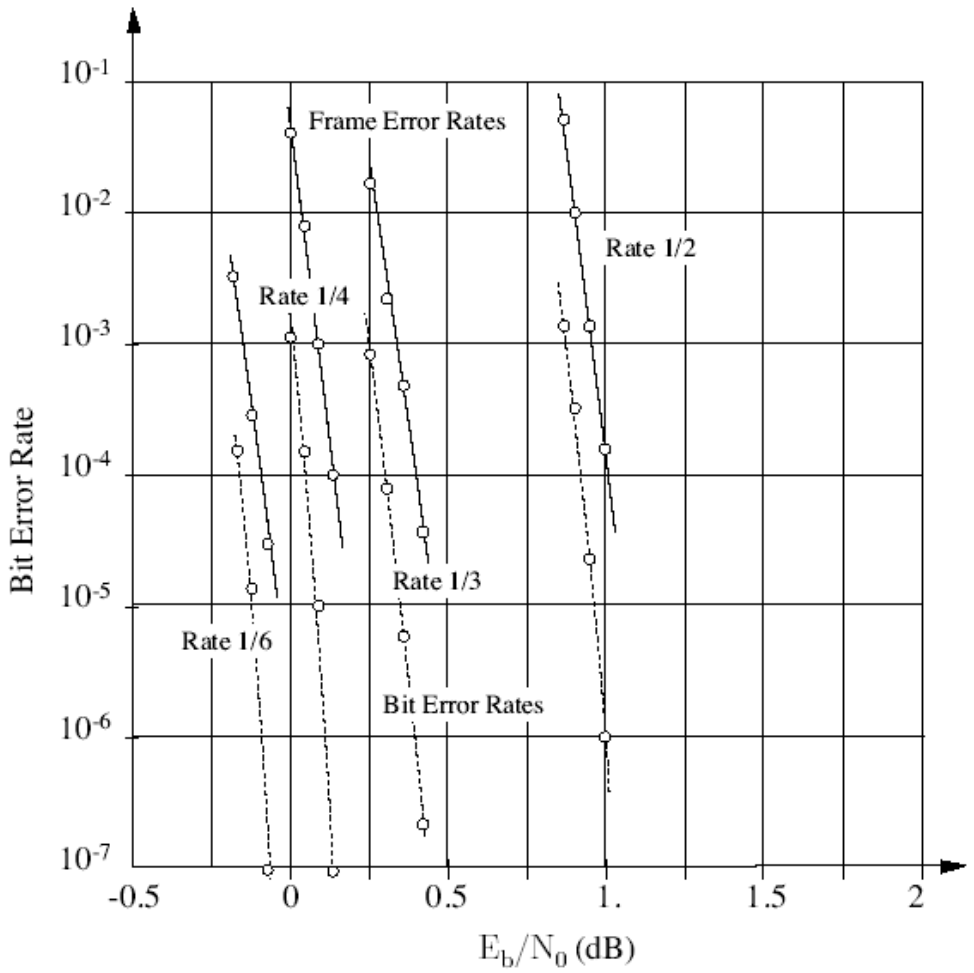


Figure 4.34: Frame and bit error rate performance of the CCSDS turbo code with $N = 8920$.

■ The Turbo Coding Scheme in 3GPP Standard

经过 10 多年的理论研究与实用化探索，Turbo 码在各方面都走向了实用阶段。在包括 3GPP 的 WCDMA、CDMA2000 等在内的第三代移动通信系统（IMT-2000）的信道编码方案中，都使用了 Turbo 码，用于高速率、高质量的通信业务。

在 3GPP 中，传输信道主要采用了三种信道编码方式：无信道编码，卷积码，Turbo 码。其具体应用范围及参数见表 4.4：

表 4.4 3GPP 中信道编码参数

信道类型	编码方式	码率
广播信道（BCH）	卷积码	1/2
无线寻呼信道（PCH）		
前向接入信道（FACH）		
随机接入信道（RACH）		
专用信道（DCH）		1/3,1/2 或无编码
专用信道（DCH）	Turbo 码	1/3 或无编码

第四代移动通信（LTE）同样采用了卷积码与 Turbo 码作为纠错编码方案，而且卷积码用于控制信道，Turbo 码用于数据信道。与 WCDMA 的纠错编码方案相比，LTE 对纠错编码方案进行了进一步优化。LTE 卷积编码采用 1/3 码率的咬尾卷积码（Tail-Biting Convolutional Codes, TBCC），约束长度为 7，生成多项式为$[133, 171, 165]_8$ 。

LTE Turbo 编码结构与 WCDMA 相近，都是基于两个 8 状态的生成多项式为 $g_0 = 13, g_1 = 15$ 的卷积码并行级联得到 1/3 码率的 Turbo 母码，但是分量码间的内部交织器与 WCDMA 完全不同。图 4.35 给出了 LTE 标准中 Turbo 码的编码器。两个分量编码器结构相同，其生成矩阵为

$$\mathbf{G}(D)=\left[1,\ \frac{1+D+D^3}{1+D^2+D^3}\right]$$

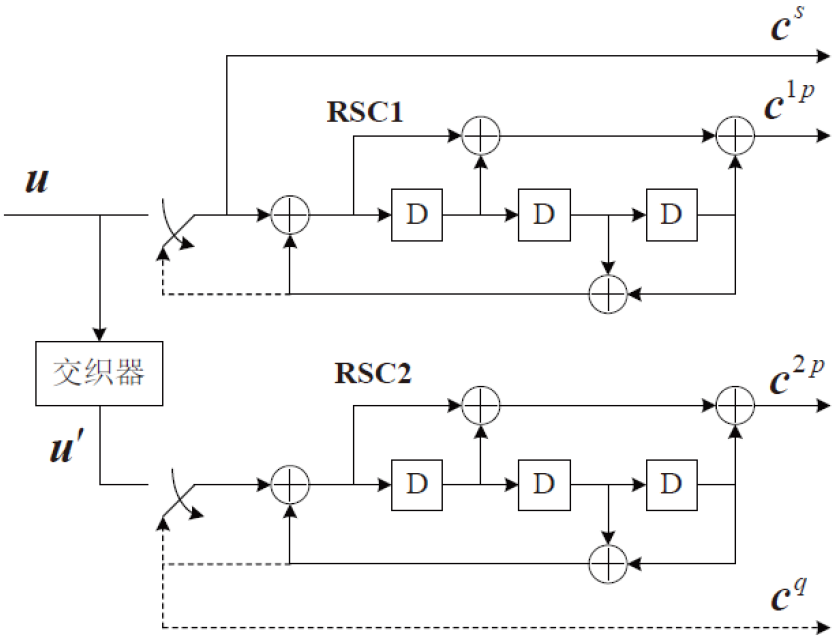


图4.35　LTE -Turbo码编码器结构（虚线仅在归零过程有效）

LTE Turbo 采用二次置换多项式（Quadratic Permutation Polynomials, QPP）交织器，其定义如下：

$$\pi(i)=\left(f_1\cdot i+f_2\cdot i^2\right)\bmod K$$

其中 i 代表交织后的序号，$\pi(i)$ 代表交织器输入的序号，K 是编码器输入比特序列的长度，其范围为 $40\leqslant K\leqslant 6144$。$f_1$ 和 f_2 均为交织器的参数，与信息位长度 K 有关，可以通过 3GPP 标准中的 Turbo 码交织器参数表查得。QPP 交织器的主要优点是易于实现，无冲突交织处理，能够支持并行译码，相对于 WCDMA Turbo 码的串行译码能够显著提高译码速度。由于 QPP 交织器存在平方运算，为了简化运算和便于硬件处理，在具体工程实现时通常对上式采用递推方式计算：

$$\begin{aligned}\pi(i+1) &= \left(f_1\cdot(i+1)+f_2\cdot(i+1)^2\right) \mod K \\ &= \left[\left(f_1\cdot i+f_2\cdot i^2\right)+\left(f_1+f_2+2f_2\cdot i\right)\right] \mod K \\ &= \left[\pi(i)+g(i)\right] \mod K\end{aligned}$$

其中 $g(i)=f_1+f_2+2f_2\cdot i$，可用类似方法递推计算得到：

$$g(i+1)=g(i)+2f_2 \mod K$$

进一步，由于 $\pi(i)$ 和$g(i)$都小于K，上述算法的求模操作可以简化为比较和减法运算。

QPP 交织器具有最大无冲突(Contention Free, CF) 特性，因此QPP交织器在并行译码上具有很大的优势，可以支持帧长为K的任意M个大小为L的子译码器($K=M\cdot L$) 同时进行独立的互不干扰的并行译码。例如，一个长度为$K=512$的码块，可以采用{1, 2, 4, 8, 16, 32, 64, 128, 256, 512}个子译码器同时进行无冲突译码，从而使译码器的设计更加灵活，这是实现并行译码来降低译码时延的关键。

对于码率为 1/3 的 Turbo 码，图 4.35 中不存在删余部分，其输出码字序列为 $c_1^s, c_1^{1p}, c_1^{2p}, c_2^s, c_2^{1p}, c_2^{2p}, \cdots$。

LTE-Turbo 码的归零：

当一组信息位（交织器大小）经过编码器编码完后，LTE编码器通过移位寄存器的反馈信息进行网格图归零，如图4.35 中虚线部分所示。此时的输入比特等于反馈比特，这样在进行异或运算后输入到移位寄存器中的是全0比特。两个分量码编码器分别进行归零处理，整个Turbo 编码器所传输的编码后归零信息为 $(c_{K+1}^s, c_{K+1}^{1p}, c_{K+2}^s, c_{K+2}^{1p}, c_{K+3}^s, c_{K+3}^{1p}), (c_{K+1}^q, c_{K+1}^{2p}, c_{K+2}^q, c_{K+2}^{2p}, c_{K+3}^q, c_{K+3}^{2p})$。码字总长度为$(K/R+12)$ 比特。由于尾比特的加入，Turbo 码的实际码率有所降低。

4.6.5 Implementation Issues

FPGA-based Codec: 图 4.36 所示是一个生成多项式为$(15,11,13,17)_8$ 的双二进制 Turbo 码编码器。这里的 turbo 编码使用两个两输入三输出的（3,2,3）循环递归卷积编码器（CRSC），编码约束长度为 4，将$1+D+D^3$作为反馈多项式，故用生成多项式可以表示为：

$$G(D)=\left[1,\frac{1+D^3}{1+D+D^3},\frac{1+D^2+D^3}{1+D+D^3},\frac{1+D+D^2+D^3}{1+D+D^3}\right]$$

该双二进制 Turbo 码的分量编码器采用咬尾编码方式，保证编码器的初始状态和终止状态相同。交织器分为符号内交织和符号间交织，假定 N 为信息比特序列长度，则信息符号长度为 $K=N/2$。

① 先对同一符号中的两个比特进行内交织：对于 k 从 0 到 K-1，当 k 为偶数时，交换同一输入符号中的两个比特的顺序；

② 然后对内交织后的符号继续进行符号之间的交织，符号间交织映射表使用的是二次置换多项式函数 QPP 生成。

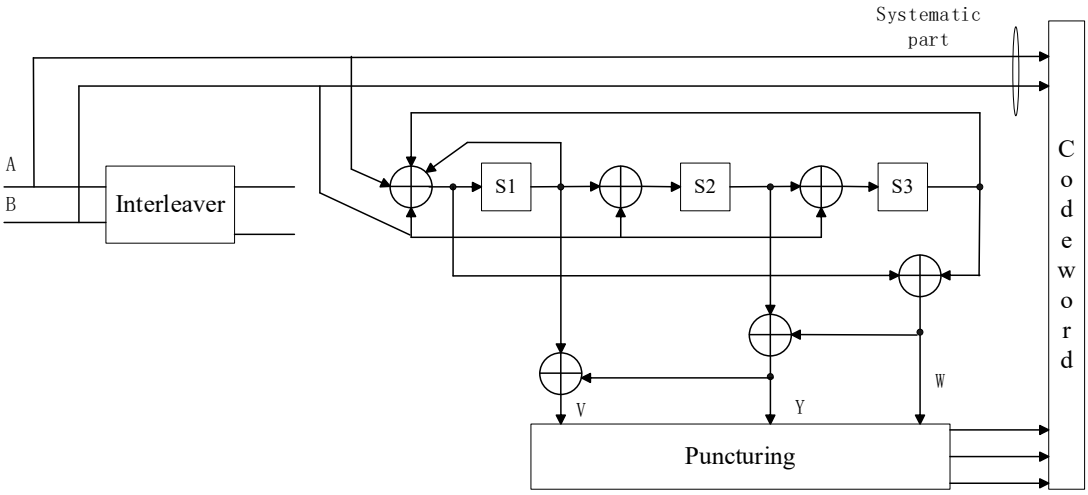


图 4.36　A duobinary turbo 码编码器

针对该 Turbo 码，可有如下的编码器的 FPGA 实现方案设计，如图 4.37a 所示。

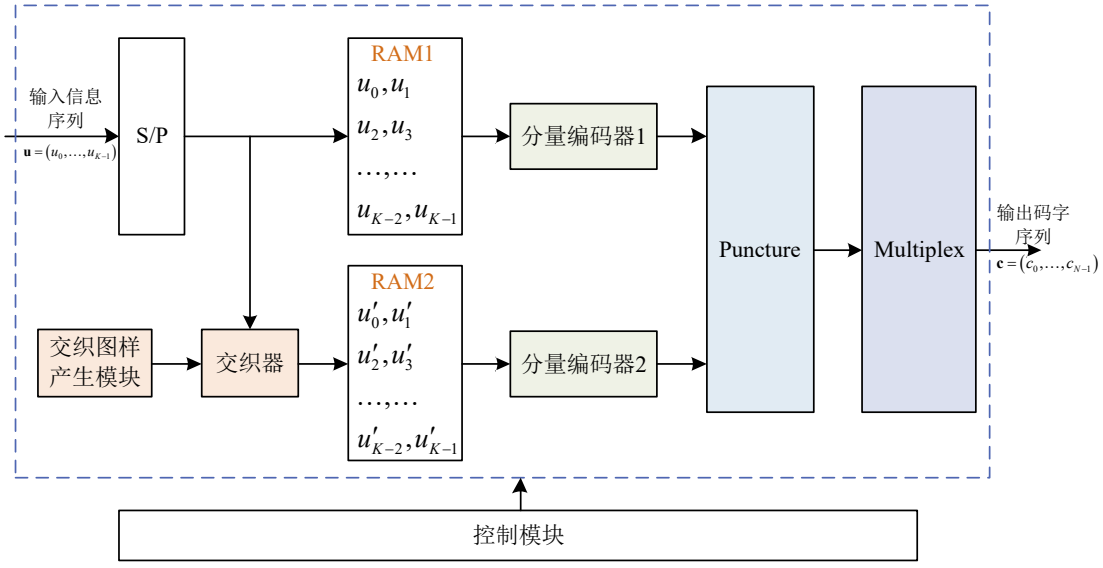


图 4.37a 双二进制 Turbo 码编码器 FPGA 实现方案设计框图

双二进制 Turbo 码译码器采用基于增强 Max-Log-MAP 算法的分段并行译码方案。译码器的 FPGA 实现方案设计，如图 4.37b 所示。

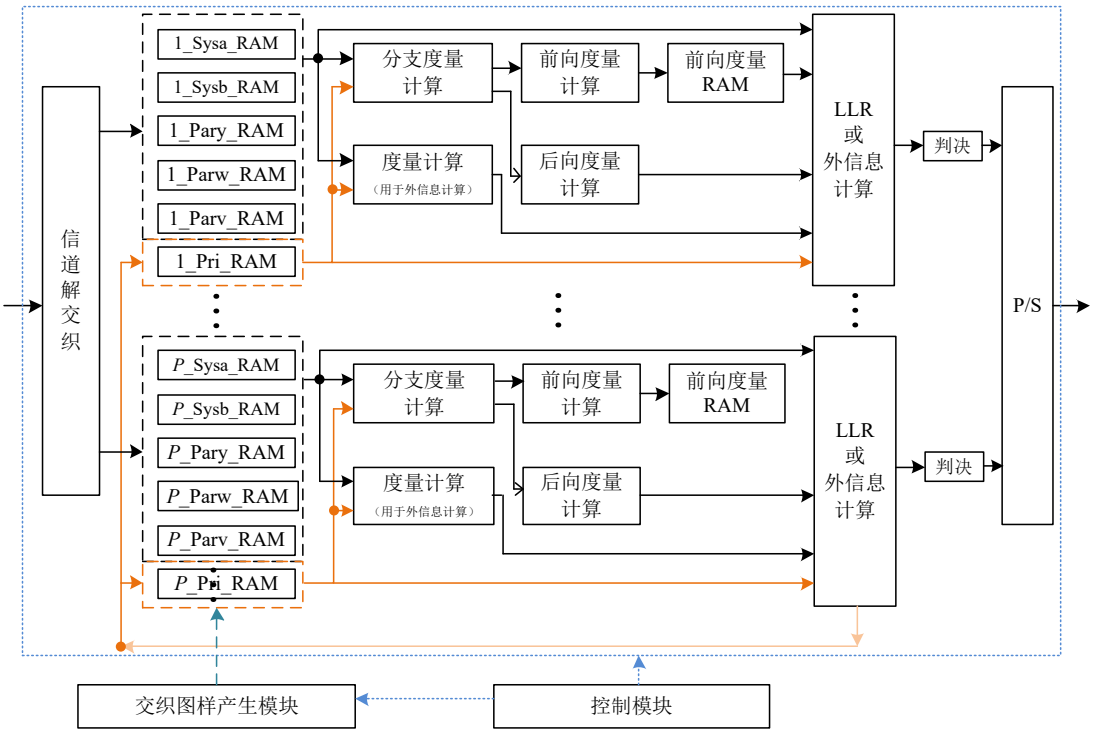


图 4.37b 双二进制 Turbo 码译码器 FPGA 实现方案设计框图

从图 4.37b 中可看出，解调后的数据先经过信道解交织模块，一边解信道交织一边根据控制模块给出的相关参数信号（并行度、码率、码长等）将解信道交织后的数据存入对应的 P 个系统位存储器 Sysa_RAM、Sysb_RAM 和校验位存储器 Pary_RAM、Parw_RAM、Parv_RAM 中。然后，根据控制模块给出的译码迭代次数计数信号判断输送到译码主模块（分支度量、前向度量、后向度量计算模块）的系统位信道数据和先验信息是否交织或解交织。在一次译码迭代中，前向度量值需存储在 RAM 中，当有效的后向度量值开始计算时，倒序读取前向度量存储器中数据，同后向度量值和其他相关值一起送入 LLR 或外信息计算模块，将计算的外信息经过交织或解交织送入先验信息存储器中，将系统位信道数据交织送入译码主模块，开始分量译码器 2 的运行。执行上述译码操作，直到达到译码迭代次数，将 LLR 值输出进行判决，并串转换后输出。

4.7 Serially Concatenated Codes

Serial concatenation of codes was first proposed by Forney in 1966 [], but the serial concatenation schemes using iterative decoding was introduced in [Benedetto-TIT98]. A block diagram of an encoder of a serial concatenation of two linear codes is shown in Figure 4.38. 这里，分量码（外码或内码）既可以是分组码，也可以是卷积码。用 $(2,1,6)$ 卷积码作内码，$(255,223)$ RS码作外码的级联差错控制系统在深空通信中取得了巨大的成功。Note that, as opposed to a turbo code, in a serially concatenated coding system there is no puncturing of redundant symbols. 如果外码的码率为 R_o，内码的码率为 R_i，则串行级联

码的总码率 $R = R_o R_i$ 。

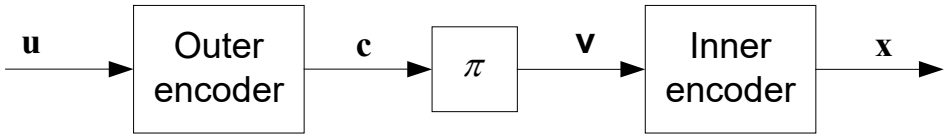


图4.38 串行级联码的编码器结构

下面，我们以串行级联卷积码（SCCC）为例来进行讨论。首先根据SCCC的ML译码性能分析结果[Benedetto-TIT98]，给出串行级联编码系统中分量码的设计原则：

- 为了获得交织增益，内码应该使用递归（系统）卷积码，且有较大的有效自由距离。
- The outer code should have large and, if possible, odd value of minimum distance.
- The outer code could be either nonrecursive or recursive. Usually, it is appropriate to choose a nonrecursive nonsystematic convolutional code as the outer code.

The last design criterion is needed in order to minimize the number of codewords of minimum weight (also known as the ***error exponent)*** and the weight input sequences resulting in minimum weight codewords. 由于非递归码有输入重量为1的错误事件，一般来说，即便发生重量为自由距离的错误事件，其中出错的信息比特个数也很少，因此选择非递归卷积码作为串行级联卷积码的外码是十分有效的。

Contrary to turbo codes, serially concatenated codes (SCCs) do not exhibit “interleaver gain saturation” (i.e., there is no error floor). As a result, SCCs outperform turbo codes in the SNR region where the error floor appears. At low SNR values, however, the better weight distribution properties of turbo codes lead to better performance than SCCs.

4.7.1 Iterative Decoding of Serially Concatenated Codes

串行级联卷积码的迭代译码器如图4.39所示。可以看出，其结构和Turbo码译码器十分相似；所不同的是，外码译码器不接收信道信息。这里，内码的SISO译码器与Turbo码中的分量译码器相同，only the LLR of the information symbols are updated。而对于外码的SISO译码器来说，其先验LLR总是设置为零，它基于MAP或max-log-MAP算法计算其信息符号与校验符号的LLR，并在此基础上获得关于外码的所有编码符号的外信息。在迭代译码中，该外信息经解交织后，作为先验LLR送给内码译码器。

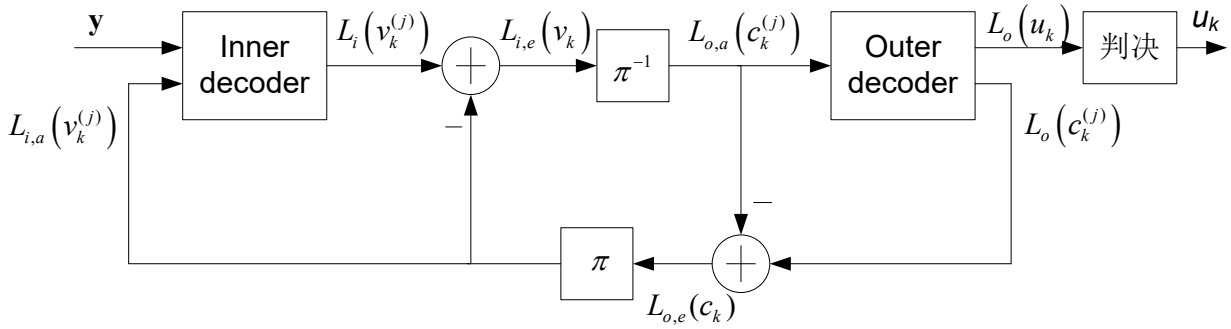


Figure 4.39 Iterative decoder of SCCC

下面我们具体讨论其迭代译码算法。根据前面的串行级联码设计原则，假定内码是一个 (n_i,k_i) RSC 码，而外码一般是一个 (n_o,k_o) 前向非系统卷积码。假定外码的码率 $R_o=1/2$，且记其编码器的输入为 $\mathbf{u}=(u_1,u_2,...,u_K)$，输出码字为 $\mathbf{c}=(\mathbf{c}_1,\mathbf{c}_2,...,\mathbf{c}_K)$ with $\mathbf{c}_k=\left(c_k^{(1)},c_k^{(2)}\right)$。由于内码译码器和 Turbo 码的子译码器一致，其译码方法可沿用 Turbo 码的方法。即在迭代译码中，内码译码器根据 MAP 算法计算 LLR：

$$L_i\left(v_k^{(j)}\right)=L_c y_k^s+L_{i,a}(v_k^{(j)})+L_{i,e}^{only}(v_k^{(j)})$$

其中，$j=0,1,...,k_i-1$，y_k^s 是与 $v_k^{(j)}$ 相对应的信道输出值。在首次迭代中，假定信息符号的先验概率相同，即 $L_{i,a}\left(v_k^{(j)}\right)=0$。

对外码译码器而言，其译码器输入只有来自于 inner decoder 的外信息，即

$$L_{o,a}=\pi^{-1}\left(L_{i,e}\right)，\text{ and}$$

$$L_{i,e}(v_k^{(j)})=L_i\left(v_k^{(j)}\right)-L_{i,a}(v_k^{(j)})=L_c y_k^s+L_{i,e}^{only}(v_k^{(j)})$$

由于外码译码器并不直接接收信道信息，用于外码译码器的分支度量需要从原来的形式中去掉信道信息那一部分，同时加入校验比特的先验信息(同样由内码译码器提供，其实是内码译码器的一部分信息比特的信息)。The branch 转移概率 is computed as

$$\gamma_k(s',s)\propto\exp\left\{\left(2c_k^{(1)}-1\right)L_{o,a}\left(c_k^{(1)}\right)+\left(2c_k^{(2)}-1\right)L_{o,a}\left(c_k^{(2)}\right)\right\}$$

(若外码为系统码，假定 $c_k^{(1)}=u_k$，则 $\gamma_k(s',s)\propto\exp\left\{\left(2u_k-1\right)L_{o,a}\left(c_k^{(1)}\right)+\left(2c_k^{(2)}-1\right)L_{o,a}\left(c_k^{(2)}\right)\right\}$)

在迭代译码过程中，外码译码器计算其输出编码比特 c_k 的 LLR 及外信息。c_k 的似然信息可以用类似于信息比特的方法得到，即外码译码器给出的各编码符号的 LLR 为：

$$L_o\left(c_k^{(j)}\right)=\ln\frac{\sum_{\forall b_k\in B_k:c_k^{(j)}=1}\tilde{\alpha}_{k-1}(s')\cdot\gamma_k^i(s',s)\cdot\tilde{\beta}_k(s)}{\sum_{\forall b_k\in B_k:c_k^{(j)}=0}\tilde{\alpha}_{k-1}(s')\cdot\gamma_k^i(s',s)\cdot\tilde{\beta}_k(s)}$$

$$=L_{o,a}(c_k^{(j)})+L_{o,e}(c_k^{(j)})$$

其中，$j=0,1,...,n_o-1$，并且

$$\gamma_k^i(s',s)=K\exp\left\{\sum_{j=0}^{n_o-1}c_k^{(j)}L_{o,a}\left(c_k^{(j)}\right)\right\}P(S_k=s\mid S_{k-1}=s',u_k=i)$$

迭代译码完毕，最后判决时外码译码器也必须计算关于输入 u_k 的 LLR，做判决 $\hat{u}_k=sign\left[L_o(u_k)\right]$。

4.7.2 性能仿真与应用

图 4.40 显示了码率为 1/3 的串行级联卷积码的误码性能。为了比较，图中还同时给出了 Turbo 码的性能曲线。用星号标识的代表串行卷积码，用方框标识的代表 Turbo 码。串行级联卷积码用码率为1/2 的 4 状态非递归卷积码作外码，码率为2/3 的 4 状态递归系统卷积码作内码。外码的生成多项式为

$$\mathbf{G}^{(o)}(D)=\left(1+D+D^2,1+D^2\right)$$

内码的生成多项式为

$$\mathbf{G}^{(i)}(D)=\begin{bmatrix}1 & 0 & 1+D^2/1+D+D^2\\ 0 & 1 & 1+D/1+D+D^2\end{bmatrix}$$

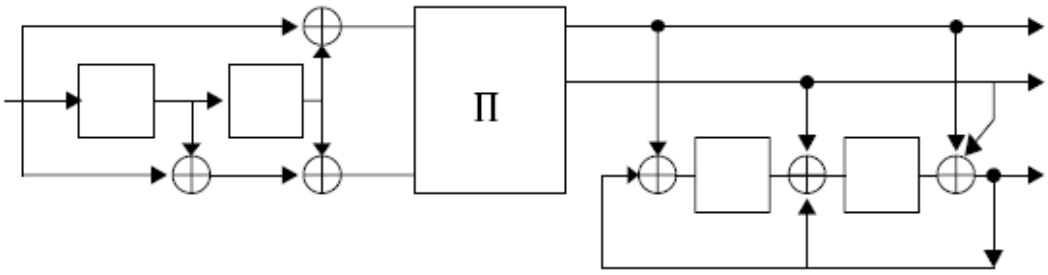


Figure 4.40a. Encoder for SCCC

Turbo 码的两个子码均为$(1,1+D^2/1+D+D^2)$码，从而两种方案的码率均为 1/3。因为外码的码率为1/2，所以用于串行级联卷积码的交织器应该比用于 Turbo 码的交织器大一倍。这里，分别将它们取为 8192 和 4096，采用随机交织。译码过程使用 MAX-LOG-MAP 算法，迭代 18 次。

从图 4.40 可以看出，中低信噪比时 Turbo 码的性能好于串行级联卷积码，但是在 1.3dB 之后，串行级联卷积码的性能更优，也就是说，串行级联卷积码具有更低的错误平层(error floor)。

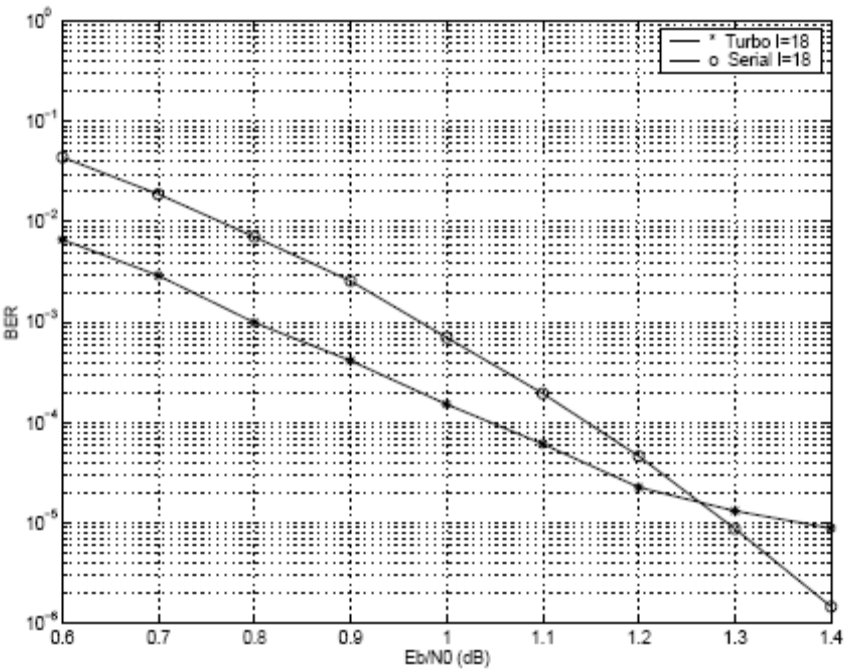


图 4.40 串行级联卷积码和 Turbo 码的误码性能比较，码率均为 1/3。

- **应用（CCSDS SCCC）**：外码、内码均为(7, 5) 4 状态 RSC encoder, s-random interleaver is used. 外码通过 puncturing 后码率 R_o=2/3。 (该编码方案用于 3GPP 第三代移动通信和 CCSDS 标准（CCSDS 131.2-B-1）建议的高速遥测应用，交织器长度依赖于编码调制模式，可在 8640—65520 比特之间变化。)

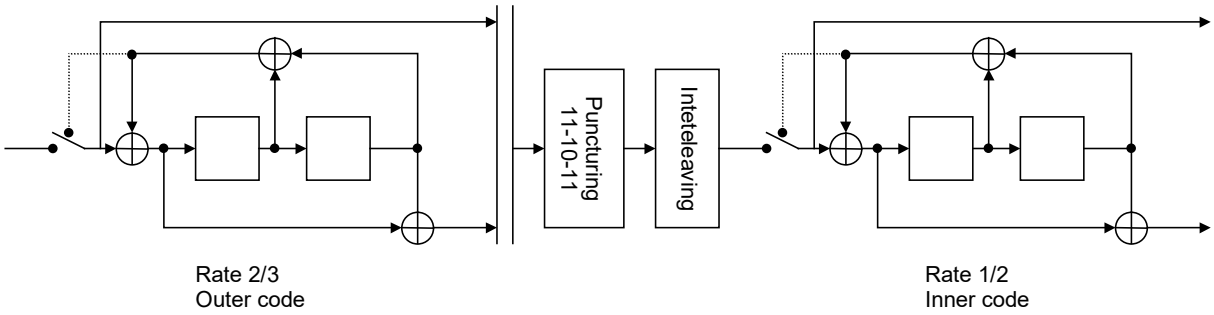


图 4.40b 1/3 码率 4 状态 SCCC 编码器结构图

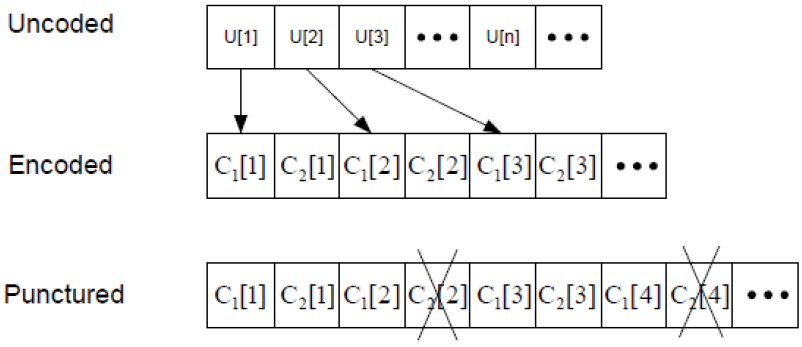


Figure 4.40c: Outer Code Puncturing Scheme

4.8 Trellis Representation of Binary Linear Block Codes

线性分组码的软判决译码算法对于采用分组码作为分量码的级联码或后面将介绍的乘积码的迭代译码起着至关重要的作用。分组码的软判决译码算法有多种，它们与码的表示方法有关。对于简单的线性分组码，我们可以用 Trellis 图来表示，并用 Viterbi 或 BCJR 算法进行软判决译码。

4.8.1 Trellis representation

A trellis $\mathcal{T}$ of length n is a directed graph with the following property:

1) the state (vertex) set $\mathcal{S}$ can be partitioned into n+1 disjoint subsets, $\mathcal{S}=\bigcup_{k=0}^{n}\mathcal{S}_k$;

2) the branch (edge, state transition) set $\mathcal{B}$ can be partitioned into n disjoint subsets, $\mathcal{B}=\bigcup_{k=1}^{n}\mathcal{B}_k$;

3) every branch b in $\mathcal{B}_k$ starts at a state in $\mathcal{S}_{k\text{-}1}$ and ends at a state in $\mathcal{S}_k$.

The trellis $\mathcal{T}$ represents a block code of length over GF(q) if each branch is labeled by an m-tuple over GF(q) representing output symbols. The set $\{k\in\mathcal{I}=[0,n]\}$ represents the discrete time indexes. A path of length n through the trellis $\mathcal{T}$ is described by a sequence of branches $b_1,b_2,...,b_n$. The set of all possible codewords is one-to-one correspondence with the set of all possible paths (of length n) through the trellis. The codeword associated with any particular path is the sequence of corresponding m-tuple labels.

With a trellis representation of a block code, we can use VA (or BCJR) to perform ML decoding of the code, as shown for convolutional codes. VA yields the most likely sequence, while the BCJR algorithm yields the most likely symbol, along with its reliability at each time.

- Example (SPC codes): Any (n, n−1, 2) single-parity-check (SPC) code has a two-state trellis representation like that shown in Figure 4.51.

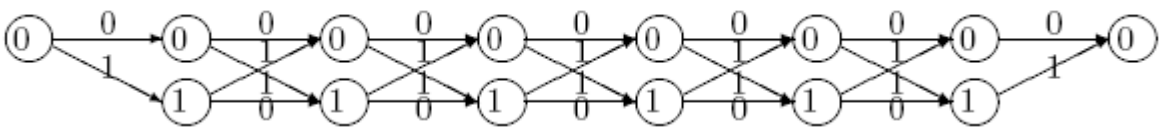


Figure 4.51 Two-state trellis for a binary (n, n−1, 2) single-parity-check code (n =7).

Question: Trellis for (n, 1, n) repetition code.

4.8.2 BCJR Trellis of a binary linear block code

Consider a binary (n, k, d) linear block code $\mathcal{C}$ with parity-check matrix $\mathbf{H} = [\mathbf{h}_0 \quad \mathbf{h}_1 \quad \cdots \quad \mathbf{h}_{n-1}]$. Let $\mathbf{x} = (x_0, x_2, ..., x_{n-1}) \in \mathcal{C}$ be an arbitrary codeword, and let r=n-k. We use the column vectors $\mathbf{s}_i \in \mathcal{S}_i$, $0 \le i \le n$-1, to denote the state variables of the trellis corresponding to $\mathcal{C}$. For BCJR trellis, $\mathbf{s}_i \in \mathbb{F}_2^r$ is defined by

$$\mathbf{s}_0 = \mathbf{0},$$

$$\mathbf{s}_{i+1} = \mathbf{s}_i + x_i \mathbf{h}_i = \sum_{t=0}^{i} x_t \mathbf{h}_t, \qquad 0 \le i < n\text{-}1 \tag{A1}$$

Clearly, $\mathbf{s}_n$ is the syndrome of $\mathbf{x}$ and $\mathbf{s}_n = \mathbf{0}$ for all codewords. The next state $\mathbf{s}_{i+1}$ is determined by the current state $\mathbf{s}_i$ and the current input x_i. The tripe ($\mathbf{s}_i$, $\mathbf{s}_{i+1}$, x_i) specifies the trellis branches in the trellis diagram. Thus a trellis can be constructed using (A1). The resulting trellis structure is irregular compared to the trellis of a convolutional code. The maximum number of states in the BCJR trellis is $\min\left\{2^k, 2^{n-k}\right\}$. The BCJR trellis has the property that it has the smallest number of states. A trellis with this property is called a *minimal trellis*.

As an example, the BCJR trellis for the (7, 4) cyclic Hamming code with

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

is shown in Fig. A1, where the trellis state $\mathbf{s}_i = (s_0, s_1, s_2)^T$ is labeled with the corresponding r-tuples over GF(2). The trellis has maximum of 8 states at time i = 3 and i = 4.

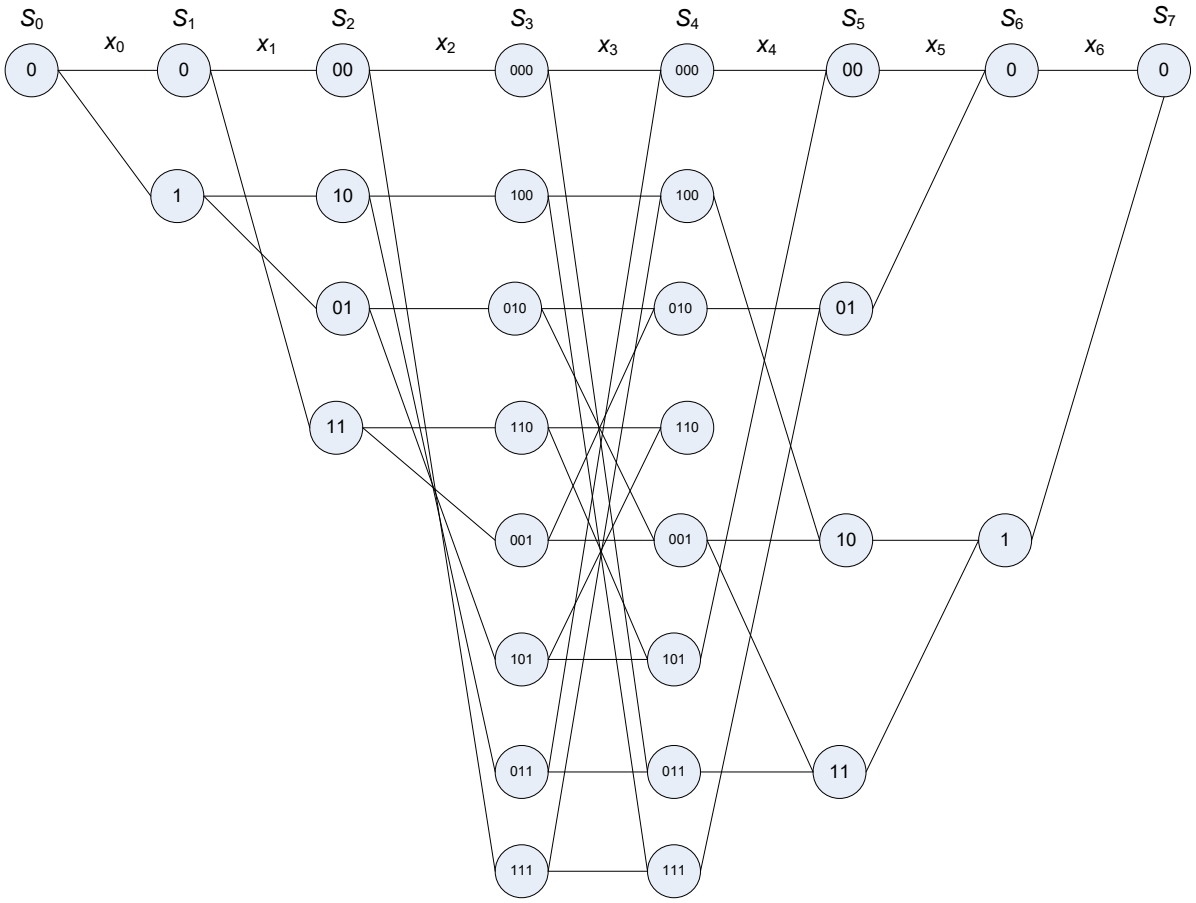


Figure A1. A trellis diagram for the (7, 4) Hamming code.

The trellis diagram represents the 16 codewords of the Hamming code as the set of labeled paths obtained starting from the leftmost vertex and proceeding rightwards in the graph. The *i*th trellis section constrains the possible combinations of $(\mathbf{s}_{i-1}, x_{i-1}, \mathbf{s}_i)$; in fact, in this linear trellis example, these triples always form a linear code. For example, the 3rd section forms the local code that consists of the eight binary linear combinations of (00, 0, 000), (00, 1, 111), (10, 0, 100), (10, 1, 011), and etc. This code may be regarded as a binary (6, 3) code.

Fig. A4 is the trellis of the (5,3) block code defined by $\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}$.

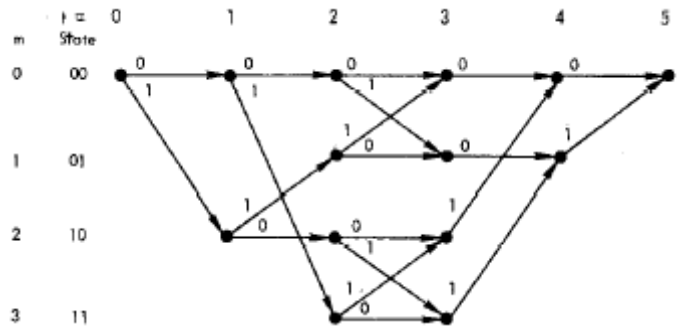


Figure A2

It is interesting to note that with a syndrome trellis there is no need to label the branches with the coded bits. A transition between two states with the same label (i.e., a horizontal branch) corresponds to coded bit 0, as seen from equation (A1)*: If the coded bit is 0, then the sum does not change*.

For some classes of codes, such as extended BCH codes and Reed-Muller codes, the trellis can be divided into sections. This results in a more regular and symmetric trellis structure with many parallel subtrellises, which may be used to build very high-speed Viterbi decoders for block codes.

4.8.3 Trellis complexity and Minimal trellis

There are various measures of the complexity of a trellis, which are all related to VA decoding complexity. The most common is perhaps the *state complexity profile*, which is just the sequence of state space sizes. (We define *state complexity profile* as the sequence of the numbers of trellis states $|\mathcal{S}_k|$ at various times $k \in \mathcal{I} = [0, n]$) For example, the state complexity profile of the trellis of Figure 4.51 is {1,2,2,2,2,2,2,1}, and that of the trellis of Figure A1 is {1,2,4,8,8,4,2,1}.The state complexity of a trellis is often defined as the maximum state space size $\max_{k \in I} |\mathcal{S}_k|$; e.g., 2 or 8, respectively, for these two trellises. Another possible complexity measure is the *branch complexity profile*, defined as the sequence of numbers of branches in each section— e.g., {2,4,4,4,4,4,2} and {4,8,8,4} for the trellises of Figures A1 and A2, respectively. The branch complexity is the maximum number of branches in any section— e.g., 4 or 8, respectively. The complexity of VA decoding is more precisely measured by the branch complexity profile, but any of these measures gives a good idea of VA decoding complexity.

Definition [Minimal trellis]:

A minimal trellis for a given linear block code is defined as a trellis that has the minimum possible state space size at each time. In other words, it satisfies the property that for each *k*, the number of states $|\mathcal{S}_k|$ in T≤the number of states at time *k* in any other trellis representation.

For obtaining the minimal trellis for a linear code, refer to the book [Lin&Costello2004, Ch.9][Lin&Fossorier1998].

4.9 Turbo 乘积码(TPC/BTC)及其迭代译码

乘积码是一种多维阵列编码技术，是由Elias于1954年首先提出的由简单线性分组码构造复杂长码的一种方法。乘积码不但可以纠正随机错误，而且可以纠正大量的突发错误。常用的是二维乘积码，其译码可以采用行码和列码单独译码，这样可以简化译码的复杂性。1993/1994年提出了乘积码的迭代译码技术，其基本原理类似于Turbo码，即基

于行码和列码的网格结构，采用最大后验概率（MAP）译码算法。这样的乘积码称为Turbo乘积码(TPC)或分组Turbo码(BTC）。1998年，R. M. Pyndiah提出了采用扩展BCH码的乘积码及一种基于Chase type-II 算法的迭代译码方法[Pyndiah98]。这种译码方法和最优的MAP迭代译码方法相比性能损失很小，复杂性较小，而且无需信道的噪声方差，这种译码方法为Turbo 乘积码在无线通信系统中的应用提供了极大的方便。目前许多公司已经推出商业化的Turbo乘积码的编译码器。

Turbo 乘积码在高码率下的性能比同样码率的（卷积）Turbo 码更加接近 Shannon 容量限，比通常的 Turbo 码具有更低的错误平层。1997 年 Nickl 和 Hagenauer 等人利用 Hamming 码（1023, 1013, 3）作为行码构成的 Turbo 乘积码 $(1023,1013,3)^2$ 在 50 次迭代译码下，Turbo 乘积码的性能与 Shannon 容量限只有 0.27dB 的差距[CL97]。

4.9.1 Turbo 乘积码的编码方法

假设两个系统分组码 $\mathcal{C}^{(1)}$ 和 $\mathcal{C}^{(2)}$，参数分别为 (n_1,k_1,d_1) 和 (n_2,k_2,d_2)，其中 n_i 为码长，k_i 为信息比特数目，d_i 为最小 Hamming 距离。乘积码 $\mathcal{C}=\mathcal{C}^1\otimes\mathcal{C}^2$ 的构造方法如图 4.9-1 所示。乘积码编码步骤如下：

（1）将 $k_1\times k_2$ 个信息比特排列成 k_2 行和 k_1 列的矩阵。

（2）用 $\mathcal{C}^{(1)}$ 对 k_2 行进行行编码。

（3）用 $\mathcal{C}^{(2)}$ 对 n_1 列进行列编码。

由此得到的乘积码$\mathcal{C}$的参数为 (n_1n_2,k_1k_2,d_1d_2)，它可以看成是 $\mathcal{C}^{(1)}$ 和 $\mathcal{C}^{(2)}$ 构成的串行级联码。如果在第（3）步 时，我们用 $\mathcal{C}^{(2)}$ 只对 k_1 列进行列编码，则得到的是 punctured 的乘积码，它等价于 $\mathcal{C}^{(1)}$ 和 $\mathcal{C}^{(2)}$ 构成的并行级联码。

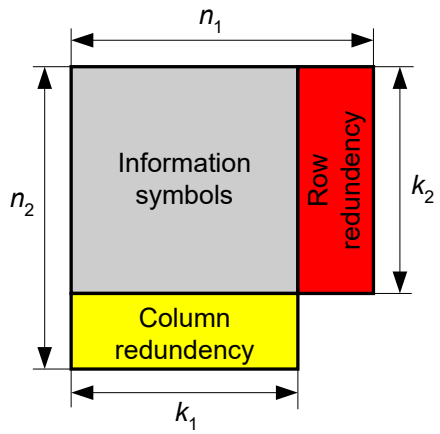


图 4.9-1：乘积码的构造

- The rate of a TPC code is

$$R=\frac{K}{N}=\frac{k_1k_2}{n_1n_2-(n_1-k_1)(n_2-k_2)}$$

$$= \frac{R_1 R_2}{R_1 + R_2 + R_1 R_2}$$

where $R_i = k_i / n_i$, $i = 1, 2$.

传统的乘积码的硬判决译码是先逐列译码，然后逐行译码，而行码和列码通常是简单的系统码，因此乘积码的译码复杂度只是随构成它的子码$\mathcal{C}^{(1)}$和$\mathcal{C}^{(2)}$的译码复杂度线性增加。如果t_1， t_2分别$\mathcal{C}^{(1)}$和$\mathcal{C}^{(2)}$的纠错能力，即：

$$t_1 = \lfloor (d_1 - 1)/2 \rfloor, \quad t_2 = \lfloor (d_2 - 1)/2 \rfloor \tag{4.9-1}$$

则乘积码可以纠正$t = \lfloor (d_1 d_2 - 1)/2 \rfloor$个随机错误，也可以纠正一个长度为

$$b \le \max(n_1 t_1, n_2 t_2) \tag{4.9-2}$$

的突发错误[]。

4.9.2 Turbo 乘积码的迭代译码

乘积码的迭代译码器结构与并行/串行级联码的译码器结构相同，可以基于分组码的trellis使用MAP算法或Max-Log-MAP算法进行迭代译码，这方面的详细讨论见[Hagenauer-TIT, no.2, 1996]。下面我们主要讨论使用Chase算法的迭代译码原理与有关算法。

4.9.2.1 The Chase Algorithm

为了能够对一般线性分组码进行软判决译码，1972 年 Chase 提出了一种次优的软判决译码算法，具有较低的复杂度。

The Chase algorithm uses a *set or list of most likely error patterns.* These error patterns are selected based on the *reliability* of the received symbols. Each error pattern is added to the hard-decision received word and decoded using a *hard-decision decoder.* Each decoded codeword is scored by computing its *metric* with respect to the received (soft-decision) sequence. The codeword with the *best metric* is selected as the most likely.

令$\mathcal{C}$ 是一个(n,k,d)二进制线性分组码，它能够纠正小于或等于$t = \lfloor (d-1)/2 \rfloor$个随机错误。考虑加性白高斯噪声（AWGN）信道上的 BPSK 数据传输。假设发送的码字为$\mathbf{c} = (c_0, c_1, ..., c_{n-1}) \in \mathcal{C}$ ，则在信道输出端的接收向量为$\mathbf{y} = (y_0, y_1, ..., y_{n-1})$，其中

$$y_j = (-1)^{c_j} + n_j, \quad j = 0, 1, ..., n-1$$

$$= x_j + n_j$$

这里 n_j 是均值为零，方差为σ^2的高斯随机变量。对 $\mathbf{y}$ 进行判决，可以得到硬判决接收字为$\mathbf{z}_0 = (z_{0,0}, z_{0,1}, ..., z_{0,n-1})$，其中

$$z_{0,j} = 0.5\left(1 - \mathrm{sgn}(y_j)\right) = \begin{cases} 0, & \text{if } y_j \geq 0 \\ 1, & \text{otherwise} \end{cases}, \quad j = 0,1,...n-1 \qquad (4.9\text{-}5)$$

在 Chase 算法中，我们需要确定每个接收比特的可靠度（Reliability）。对于 AWGN 信道上的二进制传输，接收符号的可靠性可以由其对数似然比计算得到：

$$\Lambda(z_{0,j}) = \ln \frac{P(y_j \mid c_j = 0)}{P(y_j \mid c_j = 1)} = \left(\frac{2}{\sigma^2}\right) y_j \qquad (4.9\text{-}6)$$

对于噪声方差为常数的平稳信道，比特的可靠性可简单地用其幅度值$|y_j|$来度量。在 Chase 算法中，我们将接收符号的可信/靠度按下式进行排序：

$$|y_{I_1}| \leq |y_{I_2}| \leq \cdots \leq |y_{I_n}|$$

其输出是一个排序的索引列表：$\{I_j, j = 1,2,...,n\}$。

在第一轮译码中，接收序列 $\mathbf{z}_0$ 做为硬判决译码器的输入，它所对应的译码器输出码字记为 $\mathbf{v}_0 = (v_{0,0}, v_{0,1}, ..., v_{0,n-1})$。我们按照下式计算 $\mathbf{v}_0$ 的度量，并将它作为最大值保存起来：

$$M_0 \triangleq m(\mathbf{y}, \mathbf{v}_0) = \sum_{j=0}^{n-1} (-1)^{v_{0,j}} \times y_j \qquad (4\text{-}100)$$

Chase 算法是基于如下事实：在高信噪比下，最大似然译码输出的结果以极大概率位于半径为$(d-1)$球心为$\mathbf{z} = (z_1, z_2, ..., z_n)$的超球内，其中$z_j = 0.5\left(1 - \mathrm{sgn}(y_j)\right)$。根据上述事实，搜索最大似然码字过程可在以 $\mathbf{z}$ 为球心，$(d-1)$为半径的超球中进行。

根据测试错误图样的不同，Chase 算法分为三类，常用的是 II-型算法。它测试所有下列错误图样 $\mathbf{e}_i$：在$\lfloor d/2 \rfloor$个具有最低可靠度的比特之外最多有$t = \lfloor (d-1)/2 \rfloor$个错误。通过测试这样的错误图样，II-型 Chase 算法在算法复杂度与译码性能之间达到了较好的折中。

■ II-型 Chase 算法：

1）形成硬判决接收序列 $\mathbf{z}_0$ from $\mathbf{y}$. 计算硬判决接收序列 $\mathbf{z}_0$ 中每个比特的可靠度，并确定其中$\lfloor d/2 \rfloor$个可靠性最小的比特的位置。

2）生成测试错误图样集合$T = \{\mathbf{e}_i, i = 1,2,...,2^t - 1\}$：The error patterns are generated among the t least reliable positions; i.e., positions $\{I_1, I_2, ..., I_t\}$ for which reliabilities (amplitudes) are the smallest。这是由以下二进制矢量构成的集合：考虑在 $\mathbf{z}_0$ 的 t 个可靠性最低的位置上“0”和“1”的所有可能组合。

3）For $i = 1,2,...,2^t - 1$, an error pattern $\mathbf{e}_i$ is added to the hard-decision received word $\mathbf{z}_0$, 生

成测试序列（modified received vector）： $\mathbf{z}_i = \mathbf{e}_i \oplus \mathbf{z}_0$.

4）Each $\mathbf{z}_i$ is input to a hard-decision decoder, producing a codeword $\mathbf{v}_i \in \mathcal{C}$, $i = 1, 2, ..., 2^t - 1$.

5）Steps 3) and 4) result in a list of candidate codewords. Compute the soft-decision decoding metrics of these candidates according to (4.100) and find the one with the largest metric, which is the decoded codeword as the most likely.

线性分组码的 Chase 译码算法相对于穷搜索而言，计算复杂性大大降低。但是这种译码方法输出的是硬判决信息，而迭代译码中每个译码步骤必须是软输入软输出译码，因此不能直接应用于 Turbo 乘积码的迭代译码。

4.9.2.2　基于 Chase 算法的软输入软输出译码原理

根据 Chase 算法虽然可以得到 Turbo 乘积码中行码或列码的译码输出 $\mathbf{v} \in \mathcal{C}$，但是要实现迭代译码必须计算 **v** 中各个比特的软信息。比特 v_j 的软信息定义为发送符号为 c_j 时的对数似然比，即：

$$\Lambda(v_j) = \ln\left(\frac{P(c_j = 0 \mid \mathbf{y})}{P(c_j = 1 \mid \mathbf{y})}\right) \tag{4.9-7}$$

考虑到 **v** 是 $\mathcal{C}$ 的 2^k 个码字中的一个，式(4.9-7)中的分子部分可以写成：

$$P(c_j = 0 \mid \mathbf{y}) = \sum_{\mathbf{c}_i \in S_j^0} P(\mathbf{c} = \mathbf{c}_i \mid \mathbf{y}) \tag{4.9-8}$$

其中 $S_j^0 = \left\{\mathbf{c} \in \mathcal{C} \mid c_j = 0\right\}$ 表示 $c_j = 0$ 的码字的集合。同样，式(3-7)中分母部分可以写成：

$$P(c_j = 1 \mid \mathbf{y}) = \sum_{\mathbf{c}_i \in S_j^1} P(\mathbf{c} = \mathbf{c}_i \mid \mathbf{y}) \tag{4.9-9}$$

其中 $S_j^1 = \left\{\mathbf{c} \mid c_j = 1\right\}$ 表示 $c_j = 1$ 的码字的集合。考虑到不同码字是等概分布，利用 Bayes 公式可以将式(4.9-7)写成：

$$\Lambda(v_j) = \ln\left(\frac{\sum_{\mathbf{c}_i \in S_j^0} p(\mathbf{y} \mid \mathbf{c} = \mathbf{c}_i)}{\sum_{\mathbf{c}_i \in S_j^1} p(\mathbf{y} \mid \mathbf{c} = \mathbf{c}_i)}\right) \tag{3-10}$$

其中

$$p(\mathbf{y} \mid \mathbf{c} = \mathbf{c}_i) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{|\mathbf{y} - \mathbf{x}_i|^2}{2\sigma^2}\right) \tag{3-11}$$

是 **y** 的条件概率密度函数，并且 $\mathbf{x}_i = 1 - 2\mathbf{c}_i$ 是对应于 $\mathbf{c}_i$ 的信号序列（已调码字）。令

$\mathcal{X}_j(+1)=\{\mathbf{x}\,|\,x_j=+1\}$ 和 $\mathcal{X}_j(-1)=\{\mathbf{x}\,|\,x_j=-1\}$ 分别表示与 S_j^0 和 S_j^1 相对应的已调码字（modulated codewords）集合。令 $\mathbf{x}_j^l$，$l\in\{+1,-1\}$ 是集合 $\mathcal{X}_j(l)$ 中和 $\mathbf{y}$ 具有最小平方欧氏距离的已调码字，即

$$\mathbf{x}_j^l=\arg\min_{\mathbf{x}\in\mathcal{X}_j(l)}|\mathbf{y}-\mathbf{x}|^2$$

合并(3-10)和(3-11)可以得到：

$$\Lambda(v_j)=\frac{1}{2\sigma^2}\left(\left|\mathbf{y}-\mathbf{x}_j^{-1}\right|^2-\left|\mathbf{y}-\mathbf{x}_j^{+1}\right|^2\right)+\ln\left(\frac{\sum_i A_i}{\sum_i B_i}\right) \tag{3-12}$$

其中

$$A_i=\exp\left(\frac{\left|\mathbf{y}-\mathbf{x}_j^{+1}\right|^2-\left|\mathbf{y}-\mathbf{x}_i\right|^2}{2\sigma^2}\right)\le 1，\text{并且}\,\mathbf{x}_i\in\mathcal{X}_j(+1) \tag{3-13}$$

$$B_i=\exp\left(\frac{\left|\mathbf{y}-\mathbf{x}_j^{-1}\right|^2-\left|\mathbf{y}-\mathbf{x}_i\right|^2}{2\sigma^2}\right)\le 1，\text{并且}\,\mathbf{x}_i\in\mathcal{X}_j(-1) \tag{3-14}$$

在高信噪比下，即 $\sigma\to 0$ 时，有：

$$\sum_i A_i\approx\sum_i B_i\to 1 \tag{3-15}$$

因此式(3-12)右侧的第二项趋于零，将其忽略，可以得到比特 v_j 的近似对数似然比：

$$\Lambda'(v_j)=\frac{1}{2\sigma^2}\left(\left|\mathbf{y}-\mathbf{x}_j^{-1}\right|^2-\left|\mathbf{y}-\mathbf{x}_j^{+1}\right|^2\right) \tag{3-16}$$

将式(3-16)展开得到：

$$\Lambda'(v_j)=\frac{2}{\sigma^2}\left(y_j+\sum_{m=0,m\ne j}^{n-1}y_m x_{j,m}^{+1}p_m\right) \tag{3-17}$$

其中

$$p_m=\begin{cases}0 & \text{if } x_{j,m}^{+1}=x_{j,m}^{-1}\\ 1 & \text{if } x_{j,m}^{+1}\ne x_{j,m}^{-1}\end{cases} \tag{3-18}$$

$x_{j,m}^l$ 表示 $\mathbf{x}_j^l$ 的第 m 位已调符号。如果信道的噪声方差 σ^2 是常数，可将对数似然比进行归一化，省去因子 $2/\sigma^2$，得到：

$$y_j'\triangleq\Lambda'(v_j)\cdot\frac{\sigma^2}{2}=y_j+w_j \tag{3-19}$$

其中

$$w_j = \sum_{m=1,m\neq j}^{n} y_m x_{m,j}^{+1} p_m \tag{3-20}$$

归一化的对数似然比 y'_j 就是译码器输出的软信息，它和 x_j 具有相同的符号，它的绝对值表示判决可靠性的度量。从(3-19)式可以看出，软信息等于软输入 y_j 加上一个修正项 w_j，这个修正项是两个和接收字 **y** 具有最小平方欧式距离的码字的函数，起着和 Turbo 码中外信息（Extrinsic Information）相同的作用。由于这个外信息是一组独立同分布的随机变量之和，因此近似为高斯随机变量，而且和软输入 y_j 是不相关的。

4.9.2.3　实用软输出算法

利用(3-16)式计算译码输出的软信息时，必须确定两个码字 $\mathbf{x}_j^{+1}$ 和 $\mathbf{x}_j^{-1}$ 。显然，与 Chase 算法判决结果 **v** 相对应的 **x**=1-2**v** 是其中之一，我们必须寻找另一个码字，称之为 $\tilde{\mathbf{x}}$，which can be viewed as a competing codeword of **x** at minimum Euclidean distance from **y** with $\tilde{x}_j \neq x_j$. $\tilde{\mathbf{x}} \in \left\{\mathbf{x}_j^{+1}, \mathbf{x}_j^{-1}\right\}$。对于给定的 **x** 和 $\tilde{\mathbf{x}}$，译码器输出的软信息为：

$$y'_j = \left(\frac{\left\|\mathbf{y}-\tilde{\mathbf{x}}\right\|^2 - \left\|\mathbf{y}-\mathbf{x}\right\|^2}{4} \right) x_j \tag{3-21}$$

迭代译码时，译码器输出的外信息为

$$L_e(v_j) = \Lambda'(v_j) - 2y_j/\sigma^2 \qquad \text{（对数似然比 LLR）}$$

或者归一化后为

$$w_j = y'_j - y_j$$

为了寻找码字 $\tilde{\mathbf{x}}$，必须增加 Chase 算法中的搜索空间的范围，即增加 Chase 算法可靠性最小的比特数目 t 和增加测试图案的数目。显然增加 t 可以增加搜索到 $\tilde{\mathbf{x}}$ 的概率，但另一方面，增加 t 导致译码复杂性呈指数方式增加，因此必须在译码复杂性和性能之间折衷选择 t 的值。这样，在某些情况码字 $\tilde{\mathbf{x}}$ 在选择的搜索空间中可能搜索不到，因此必须寻找其他方法来计算软信息。如果对于给定位置 j, 1<=j<=n, no pair of sequences $\mathbf{x}_j^{+1}$ and $\mathbf{x}_j^{-1}$ can be found by Chase 算法，一个比较合适的方法是：

$$y'_j = y_j + w_j, \quad w_j = \beta_c \cdot x_j \tag{3-22}$$

其中 $\beta_c \geq 0$ 是一个用来补偿外信息中近似的修正因子，其取值通过仿真确定：

$$\beta_c = \left| \log\left(\frac{P(c_j\text{正确})}{P(c_j\text{不正确})} \right) \right|$$

(3-22)这个方法非常简单，而且在迭代译码过程中非常有效，这种近似计算软信息是基于以下事实：

（1）软信息 y_j' 的符号和 x_j 相同，只是绝对值是 $\tilde{\mathbf{x}}$ 的函数。

（2）如果 Chase 算法在规定的搜索空间未能搜索到码字 $\tilde{\mathbf{x}}$，则 $\tilde{\mathbf{x}}$ 可能以极大概率远离接收字 **y**。

（3）如果 $\tilde{\mathbf{x}}$ 远离 **y**，正确判决 x_j 的概率可能很大，因此 x_j 的可靠性相应很大。

(3-22)式中的β可以看作为一个衡量码字 $\tilde{\mathbf{x}}$ 不在搜索空间Ω中时软信息的平均值。对于逐比特计算软信息，利用(3-22)计算软信息虽然没有(3-21)式计算软信息的结果准确，但是这种可靠性平均值的方法确实是有效的。

4.9.2.4 Turbo 乘积码的迭代译码

根据前面介绍的软输入软输出译码，可以实现 Turbo 乘积码的迭代译码。假设采用 BPSK 调制，而且信道是 AWGN 信道。令 **c** 为发送码字，**y** 为接收向量。译码器首先对接收的 **y** 按照行码进行译码，译码算法采用 4.8.2.3 中介绍的软输入软输出译码。将译码输出的软信息减去原始输入信息可以得到外信息 **w**，第二次译码（即列译码）的软输入为：

$$\mathbf{y}' = \mathbf{y} + \alpha_c \mathbf{w} \tag{3-23}$$

其中 α_c 是一个比例因子，用来补偿高斯变量 **y** 和 **y**'的方差之间的差别。由于外信息 **w** 的方差在开始迭代时很大，并且随着迭代次数的增加逐渐减小。选取因子 α_c 的目的是为了减小误比特率较高时外信息对软输入软输出译码的影响。在译码迭代开始时，α_c 应该取较小的值，随着译码迭代次数的增加，误比特率趋于零，α_c 的取值也应该增加。迭代译码器结构如图 3-2 所示。

构成 Turbo 乘积码的行码和列码应选择结构简单的线性分组码，这样可以简化软输入软输出译码算法的复杂度。通常选择的是 BCH 码和扩展 BCH 码，因为 BCH 码和扩展 BCH 码具有良好的代数结构，硬判决译码有快速算法，而且由 BCH 码和扩展 BCH 码构成的 Turbo 乘积码具有良好的性能。数值仿真结果表明，由最小 Hamming 距离 $d = 4$ 和 $d = 6$ 的扩展 BCH 码构成的 Turbo 乘积码的性能许多都在截止速率（Cut-off Rate）以内，因此通常选择的 Turbo 乘积码的行码和列码都是扩展 Hamming 码（BCH 码）或扩展 BCH 码。

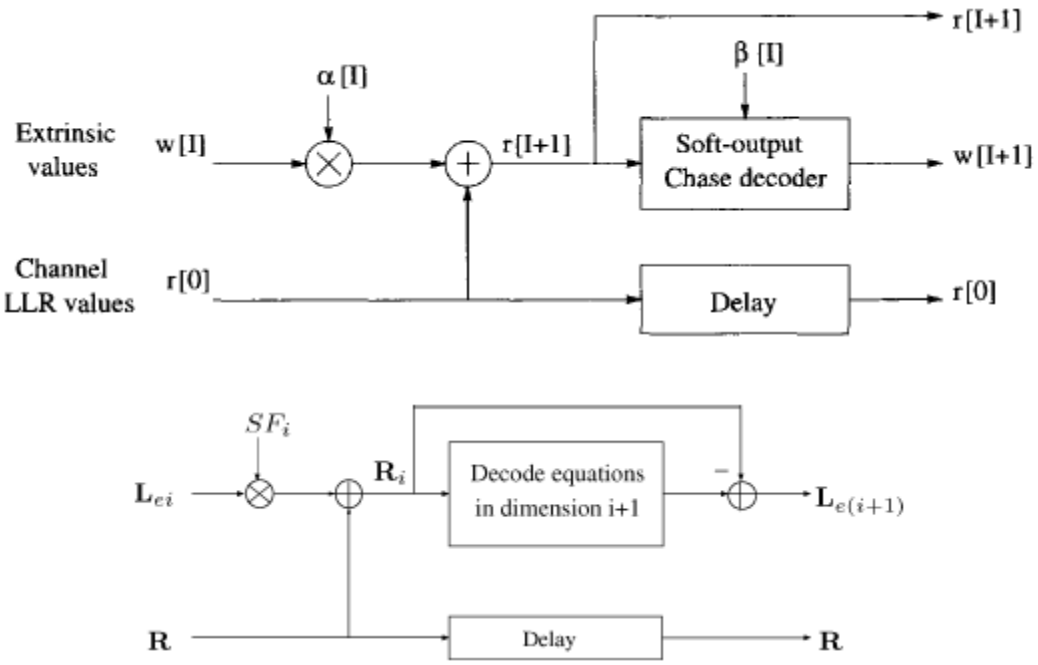


图 3-2：软输出 Chase 译码器结构框图

■ 小结：基于 Chase type-II 算法的软输出迭代译码方法[Zaragoza02]

1）初始化：

Set iteration counter $I = 0$. Let $\mathbf{y}^{(0)} = \mathbf{y}$ (the received channel values)

2）计算软输入：

For $j = 1, 2, \ldots, n$ do

$$y_j^{(I+1)} = y_j^{(0)} + \alpha_c^{(I)} w_j^{(I)}$$

3）执行 Chase 算法：

将 $\mathbf{y}^{(I+1)}$ 作为译码器输入，利用 Chase type-II 算法进行译码。令 cnt 表示所找到的码字个数，并保存在欧氏距离上与 $\mathbf{y}^{(I+1)}$ 相距最小的两个已调码字 $\mathbf{x}$ 和 $\tilde{\mathbf{x}}$。

4）求外信息：

For $j = 1, 2, \ldots, n$ do

If `cnt` >=2

$$w_j^{(I+1)} = x_j \sum_{\substack{m=1, m\neq j \\ x_m \neq \tilde{x}_m}}^{n} y_m^{(I+1)} x_m$$

else

$$w_j^{(I+1)} = \beta_c^{(I)} x_j$$

End

5）计算软输出：

Let $I = I + 1$. If $I < I_{\max}$ or a stopping criterion is not satisfied, then go to step 2).

Else compute the soft output:

For j = 1, 2, …, n do

$$\Lambda(u_i) = y_j^{(0)} + \alpha_c^{(I)} w_j^{(I)}$$

Stop

■ 另外，根据实验结果，在 Turbo 乘积码的译码过程中应注意以下三个方面：

（1）测试图案的选择

在译码复杂性和性能中折衷选择，测试图案的数目选择 16 是比较合适的，最小可靠性比特的数目选择为 4。

（2）加权因子 α_c 和可靠性因子 β_c 的选择

对于 BPSK 调制，随着译码迭代次数的增加，前 4 次迭代中 α_c and β_c 的取值通常如下（two values for the 1st and 2nd decoders）[Pyndiah-Tcom98]：

$$\alpha_c = (0.0 \quad 0.2 \quad 0.3 \quad 0.5 \quad 0.7 \quad 0.9 \quad 1.0 \quad 1.0)$$
$$\beta_c = (0.2 \quad 0.4 \quad 0.6 \quad 0.8 \quad 1.0 \quad 1.0 \quad 1.0 \quad 1.0)$$

实际应用中，β 在±10% 范围内变化对 Turbo 乘积码性能没有明显影响。为方便硬件实现，可对 α和β 的取值为恒定值，通常设置 $\alpha = 0.75, \beta = 1.0$ 。

（3）迭代次数的选择：在 Turbo 乘积码的迭代译码过程中，一般情况下迭代 4 次以后，再增加迭代次数，性能提高非常少。

4.9.3 Turbo 乘积码的性能仿真

本节对 AWGN 信道中典型的 Turbo 乘积码的性能进行数值仿真。TPC 的参数如下表所示：

TPC	生成多项式	码长	码率
(32,21)×(32,21)	$g(x) = x^{10} + x^9 + x^8 + x^6 + x^5 + x^3 + 1$	1024	0.431
(32,26)×(32,21)	$g_1(x) = x^5 + x^2 + 1$ $g_2(x) = x^{10} + x^9 + x^8 + x^6 + x^5 + x^3 + 1$	1024	0.533
(32,26)×(32,26)	$g(x) = x^5 + x^2 + 1$	1024	0.660
(64,57)×(64,57)	$g(x) = x^6 + x + 1$	4096	0.793
(128,120)×(128,120)	$g(x) = x^7 + x^3 + 1$	16384	0.879

上述 TPC 码在 AWGN 信道上的性能如下图所示。调制方式为 BPSK，译码采用 4 次迭代，不可靠位置数设为 4，α和β 的值随着迭代次数逐渐变化，从 0 增加到 1。

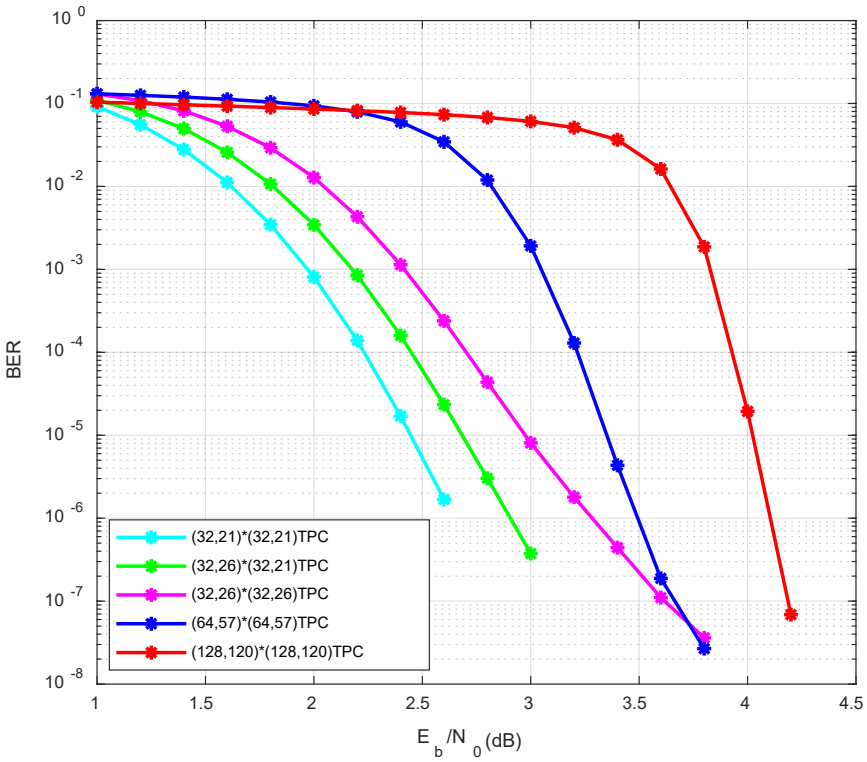


图 4.9-x Turbo 乘积码在 AWGN 信道中的性能

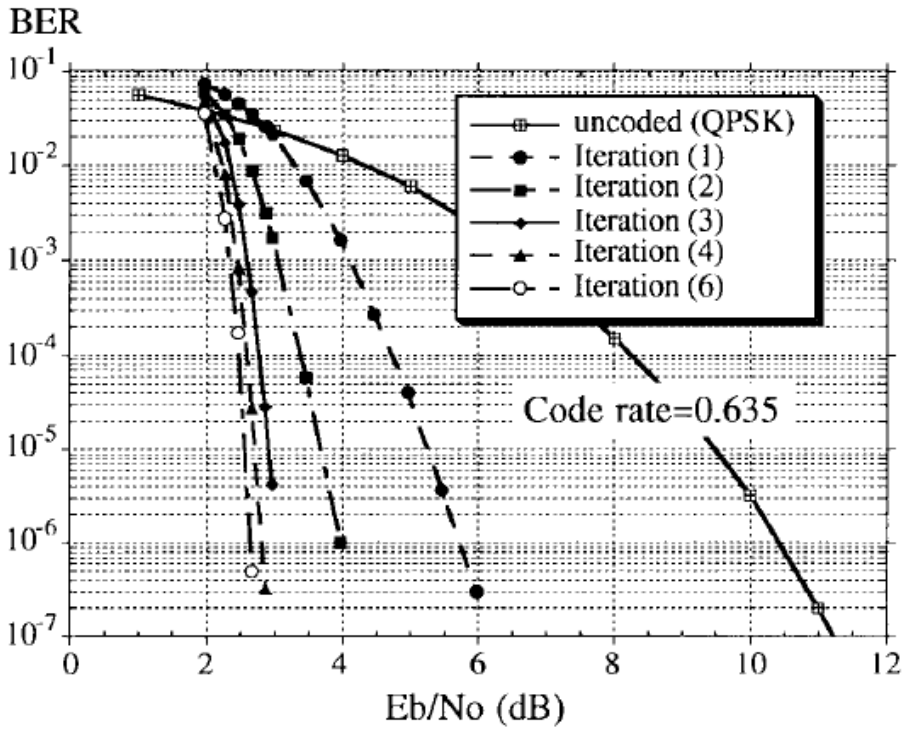
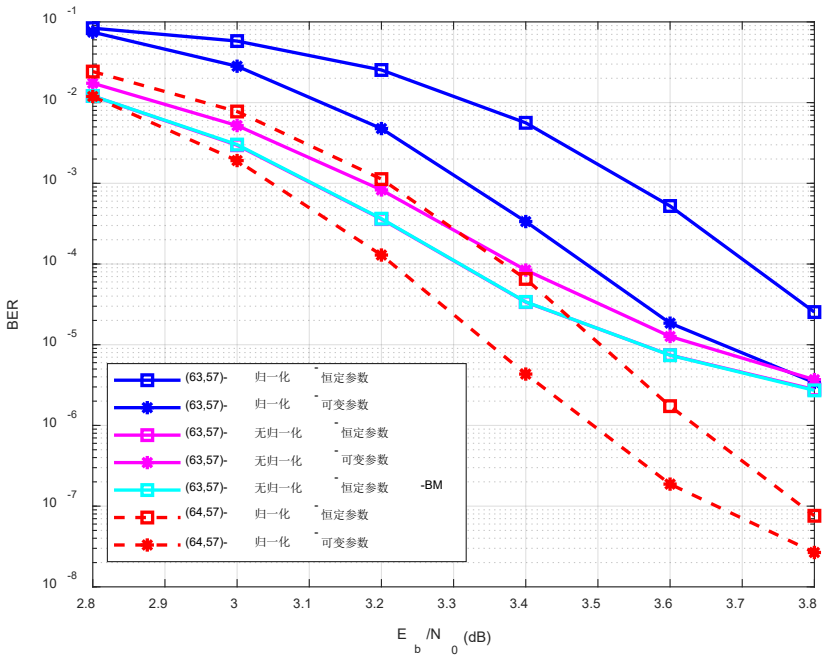


Figure x: BER vs Eb/N0 of product code [BCH(64, 51, 6)]^2 on AWGN channel with QPSK signaling [Pyndiah98, fig.3]

α和β参数选择对性能的影响：

上述仿真中α和β为可变参数，通常为方便硬件实现，我们对α和β取为恒定值，通常设置$\alpha=0.75,\beta=1.0$。以（63,57）BCH 码和（64, 57）扩展 BCH 码为例，分别作为 TPC 的分量码，我们选择不同的参数对其进行仿真，结果如下图所示。



从仿真结果可以看到，在扩展 BCH 码作为分量码时，可变参数α和β的取值对应的性能更优，但该结论对于非扩展 BCH 码作为分量码时并不成立。同时归一化操作适用于扩展 BCH 码作为分量码时的情形，对于非扩展时并不成立。

References

[1] K. Gracie and M.-H. Hamon, “Turbo and Turbo-like codes: Principles and applications in telecommunications,” *Proc. of The IEEE*, vol.95, no.6, June 2007.

[2] S. Benedetto, et al. “Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding.” *IEEE Transactions on Information Theory*, vol. 44, no. 4 (May 1998)

[3] CLAUDE BERROU, “Turbo-Information Processing: Algorithms, Implementations & Applications,” *Proc. of The IEEE*, vol.95, no.6, June 2007.

[4] Pyndiah R M. “Near-optimum decoding of product codes: block turbo codes,” *IEEE Transactions on Communications*, 1998, 46(8) : 1003 – 1010.

[5] Lin S, Fossorier M. *Trellises and Trellis-Based Decoding Algorithms for Linear Block Codes*, Kluwer Academic Publishers, 1998.

附录：LTE-Turbo 码使用的 QPP 交织器参数

i	*K*	f_1	f_2	*i*	*K*	f_1	f_2	*i*	*K*	f_1	f_2	*i*	*K*	f_1	f_2
1	40	3	10	48	416	25	52	95	1120	67	140	142	3200	111	240
2	48	7	12	49	424	51	106	96	1152	35	72	143	3264	443	204
3	56	19	42	50	432	47	72	97	1184	19	74	144	3328	51	104
4	64	7	16	51	440	91	110	98	1216	39	76	145	3392	51	212
5	72	7	18	52	448	29	168	99	1248	19	78	146	3456	451	192
6	80	11	20	53	456	29	114	100	1280	199	240	147	3520	257	220
7	88	5	22	54	464	247	58	101	1312	21	82	148	3584	57	336
8	96	11	24	55	472	29	118	102	1344	211	252	149	3648	313	228
9	104	7	26	56	480	89	180	103	1376	21	86	150	3712	271	232
10	112	41	84	57	488	91	122	104	1408	43	88	151	3776	179	236
11	120	103	90	58	496	157	62	105	1440	149	60	152	3840	331	120
12	128	15	32	59	504	55	84	106	1472	45	92	153	3904	363	244
13	136	9	34	60	512	31	64	107	1504	49	846	154	3968	375	248
14	144	17	108	61	528	17	66	108	1536	71	48	155	4032	127	168
15	152	9	38	62	544	35	68	109	1568	13	28	156	4096	31	64
16	160	21	120	63	560	227	420	110	1600	17	80	157	4160	33	130
17	168	101	84	64	576	65	96	111	1632	25	102	158	4224	43	264
18	176	21	44	65	592	19	74	112	1664	183	104	159	4288	33	134
19	184	57	46	66	608	37	76	113	1696	55	954	160	4352	477	408
20	192	23	48	67	624	41	234	114	1728	127	96	161	4416	35	138
21	200	13	50	68	640	39	80	115	1760	27	110	162	4480	233	280
22	208	27	52	69	656	185	82	116	1792	29	112	163	4544	357	142
23	216	11	36	70	672	43	252	117	1824	29	114	164	4608	337	480
24	224	27	56	71	688	21	86	118	1856	57	116	165	4672	37	146
25	232	85	58	72	704	155	44	119	1888	45	354	166	4736	71	444
26	240	29	60	73	720	79	120	120	1920	31	120	167	4800	71	120
27	248	33	62	74	736	139	92	121	1952	59	610	168	4864	37	152
28	256	15	32	75	752	23	94	122	1984	185	124	169	4928	39	462
29	264	17	198	76	768	217	48	123	2016	113	420	170	4992	127	234
30	272	33	68	77	784	25	98	124	2048	31	64	171	5056	39	158
31	280	103	210	78	800	17	80	125	2112	17	66	172	5120	39	80
32	288	19	36	79	816	127	102	126	2176	171	136	173	5184	31	96
33	296	19	74	80	832	25	52	127	2240	209	420	174	5248	113	902
34	304	37	76	81	848	239	106	128	2304	253	216	175	5312	41	166
35	312	19	78	82	864	17	48	129	2368	367	444	176	5376	251	336
36	320	21	120	83	880	137	110	130	2432	265	456	177	5440	43	170
37	328	21	82	84	896	215	112	131	2496	181	468	178	5504	21	86
38	336	115	84	85	912	29	114	132	2560	39	80	179	5568	43	174
39	344	193	86	86	928	15	58	133	2624	27	164	180	5632	45	176
40	352	21	44	87	944	147	118	134	2688	127	504	181	5696	45	178
41	360	133	90	88	960	29	60	135	2752	143	172	182	5760	161	120
42	368	81	46	89	976	59	122	136	2816	43	88	183	5824	89	182

43	376	45	94	90	992	65	124	137	2880	29	300	184	5888	323	184
44	384	23	48	91	1008	55	84	138	2944	45	92	185	5952	47	186
45	392	243	98	92	1024	31	64	139	3008	157	188	186	6016	23	94
46	400	151	40	93	1056	17	66	140	3072	47	96	187	6080	47	190
47	408	155	102	94	1088	171	204	141	3136	13	28	188	6144	263	480