摘要

欧洲标准应答器是欧洲铁路控制系统四个主要系统之一，它将位置数据或其它信号数据传送到列车上。FFFS 标准中给出了欧洲标准应答器的编码标准，其编码过程由加扰、扩位、循环码编码和测试候选报文等几步构成。本文仔细研究了欧洲标准应答器的报文格式、编码标准和测试标准部分，根据循环码的特性，深入讨论了此类编码的性能，给出实现方案。同时，在此基础上提出了译码电路的硬件解决方案，并使用 FPGA 器件实现。最后对地面信号的发生部分作了一定的研究和探索，利用 Xilinx 公司的高性能 FPGA 器件，设计了一种全数字的 FSK 信号调制电路。

关键词：欧洲标准应答器、循环码编码和译码、FPGA

ABSTRACT

Eurobalise, one of the four main systems of European railway traffic management system (ERTMS), transmits location data or other signal data to the train. The document of "FFFS for Eurobalise" presents the coding strategy of Eurobalise. The coding process of Eurobalise consists of several steps such as scrambling, expanding, computing the check bits and testing candidate telegrams. This thesis studies telegram format, coding strategy and testing standard in detail, discusses the performance of this kind of coding thoroughly according to the characteristic of cyclic code and presents the realization scheme. Meanwhile, based on the scheme it puts forward the hardware solution of decoding circuit, which is implemented by FPGA device. Finally, some studies and research for the up-link signal generator are made and a full-digital FSK modulated circuit using high-performance FPGA device of Xilinx Company is designed.

Keywords: Eurobalise, cyclic coding and decoding, FPGA

铁道科学研究院学位论文

原创性声明

本人郑重声明： 所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品或成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

论文作者签名：赵鹏飞 日期：2005年 3月20日

第一章　绪论

1.1　欧洲标准应答器简介

欧洲标准应答器（Eurobalise）是欧洲铁路控制系统（ETCS）四个主要系统之一，是用于传输不连续数据的系统。根据设计的不同，通过欧洲标准应答器将有关位置的数据（位置标记、道路倾斜度、速度限制）或是其他相关的信号数据传送到列车上[1]。

在整个ETCS系统中，Eurobalise子系统具有特别重要的意义，因为它是完全引入ETCS系统的关键前提条件。1995年1月，在经过了多年的实验之后，最终确定了EURO-Balise的技术框架及其数据格式。它是一个电感耦合的射频识别系统，采用非谐性反馈频率[1]。

Balise的电源取自一部正在通过的牵引机车，利用27.095MHz的功率载波频率上的电感耦合来实现。Balise对牵引机车的数据传输在4.24MHz的频率上进行，并且当列车速度高达500km./h时，仍可确保数据报文能够正确读出。表1.1列出了Balise子系统的基本技术参数[1]。

表1.1　Eurobalise的基本参数

耦合方式	电感式
能量传输频率，列车→Balise	27.095MHz
数据传输频率，Balise→列车	4.24MHz
调制方式	FSK
调制指数	1
数据传输率	565 kb/s
电报长度	1023或341bit
有效数据范围	830或210bit
读取距离	230至450mm
最大侧偏移	180mm
雪、水、矿石覆盖	不影响

欧洲及日本等国不同标准的应答器已推广应用的数十年，欧洲统一标准的应答器也经历了10余年的标准制定、设备研制，2003年3月发布了最新版本的标准（2.2版）并已开始推广应用。

欧洲标准应答器的应用范围主要包括以下几个方面：

1、行车安全。包括防止列车冒进，防止列车超速运行，防止错误发车，列车报号，车种识别，自动预排进路，摆式列车控制，防止列车错误通过，定位停车，列车自动防护，辅助驾驶和缩短运行间隔等。

2、道口防护。包括告警与关闭时间控制，事故通知自动化。

3、运营服务。通知和控制车门开闭，隧道内车厢封闭控制，站台和车内广播等。

1.2 欧洲标准应答器的系统结构及功能

欧洲应答器传输系统由道旁应答器和车载传输设备组成（这是ERTMS/ETCS车载要素的一部分）。应答器或者是固定的，或者是可变的。车载传输设备包括天线单元和BTM。道旁信号设备由LEU和包含在道旁信号处理中的其他外部设备组成。图1.1给出了应答器传输系统的框图[2]。

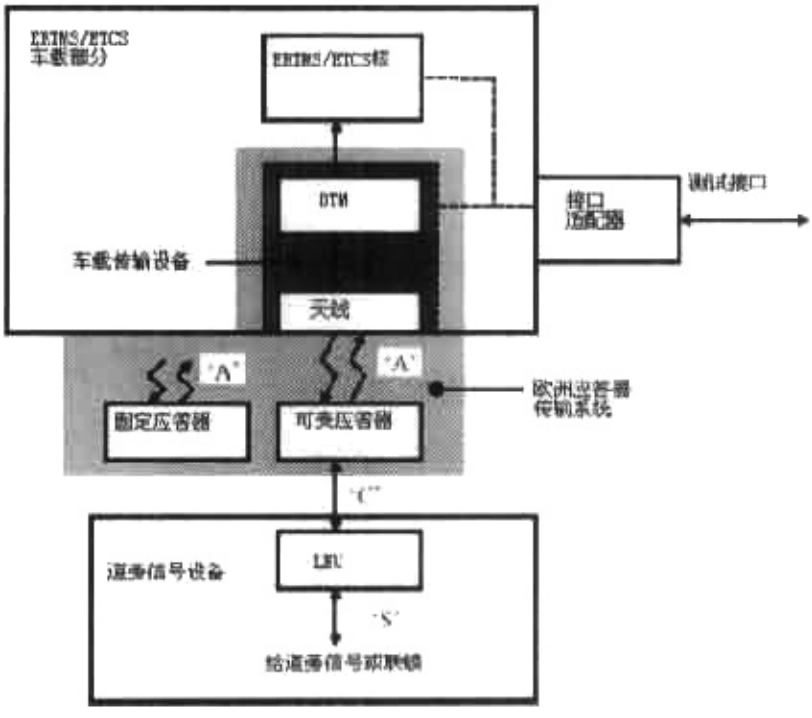

图1.1 欧洲应答器传输系统和接口

欧洲应答器传输系统包含了以下的基本功能[2]：

车载传输设备的必备功能：

- 产生远距离传递功率信号
- 确保远距离传递功率信号的电平和应答器可检测性
- 检测上行链路应答器
- 上行链路信号滤波和解调
- 防止串扰
- 阻止天线旁瓣
- 去除环境噪声
- 检查上行链路数据的编码要求
- 检测报文类型和译码
- 提取用户数据
- 报文滤波
- 管理在一个应答器通道中切换的上行链路报文
- 提供输出数据的时间戳、速度表
- 支持应答器定位
- 管理时间和速度表数据
- 检测位错误
- 启动时测试

车载传输设备的可选功能：

- 接收 KER 上行链路信号
- 检查和译码 KER 上行链路数据
- 报告 KER 上行链路数据
- 切换远距离功率模式（CW 或触发调制）
- 防止逻辑串扰
- 接收和处理下行链路信息
- 防止下行链路报文在应答器通道间进行切换
- 产生下行链路信号
- 自检

上行链路应答器的功能：

- 接收远程功率信号
- 产生上行链路信号
- 管理数据
- 启动时选择模式
- 限制上行链路区域
- 支持编程以及对操作或编程模式的管理
- 从接口 C 接收数据
- 控制 I/O 特性
- 防止和其他电缆的串扰
- 产生用于报文切换的闭塞的信号（可选）

1.3 欧洲标准应答器的信息内容和应用层次

欧洲应答器传输系统是一种安全点式传输，用于传送道旁和列车间和安全相关的信息，传输任务是通过应答器来完成的。

信息可以通过 2 种方式传输：上行链路和下行链路数据传输。由上行链路应答器发送至车载传输设备的信息根据应用的不同可以是固定的或可变的（上行链路数据传输）。信息也可以由列车发送、下行链路应答器接收（下行链路数据传输）。

点式传输是指地面设备和车载传输设备在离散位置存在的传输路径。只有当天线单元经过或停滞于相应的应答器上方的时候，信息才被会传输。

假定由轨道传输至列车的信息（上行链路信息）包括[2]：

a) 信号数据
b) 控制数据
c) 位置和地理信息
d) 功率采集信息
e) 列车行驶目标信息
f) 路由信息
g) 持久的速度限制

h) 暂时的速度限制

i) 固定的阻塞

j) 车行许可

k) 坡度

l) 联接数据

m) 其他信息

假定由列车至轨道传输的信息（下行链路信息）包括[2]：

a) 列车结构

b) 可能影响最大安全速度的列车状态的变化

c) 轨道到车载的粘着系数

d) 牵引控制的状态

e) 其他信息

欧洲应答器传输系统将用于ERTMS/ETCS中定义的应用的所有层次（分别称之为0级、1级、2级、3级和STM级）[2]。

在0级中，道旁的光学信号或ERTMS/ETCS中以外的信号的其他含义，用来给与司机通行权。除了固定的欧洲应答器，0级不使用其他的轨道至列车的信号传输来命令级别转换。因此，仍然需要读取欧洲标准应答器。除了特定的特殊命令，其他所有应答器数据都不做解释。

在1级中，通过使用可变的或固定的欧洲应答器来向列车发送列车控制信息。写入信息也可能由欧洲应答器系统通过电缆写入系统或无线写入系统提供。受控制的应答器直接和信号连接，或者和联锁相连接，或者经过欧洲应答器的接口C，和ERTMS/ETCS说明范围以外的器件相连接。

在2级中，双向的无线通信被用来传送所有的信息。点式传输道旁设备（应答器）包含了可以让列车检查、校验其速度表和辨别其实际位置的信息。有无道旁信号，都可以执行级别2。

在3级中，双向的无线通信被用来传送所有的信息。点式传输道旁设备（应答器）向列车提供信息，以使得列车可以检查和校验其速度表和辨别其实际位置。这一级别是在没有道旁信号的情况下执行的。

STM级用来运行在装备了国家列车控制和速度监督系统的线路上，按照

ERTMS/ETCS 标准装备的列车。由国家列车控制系统在道旁产生的列车控制信息，经过信道发送给列车。除了通知/命令级别转换和与应答器传输相关的特殊命令，STM 级不使用 ERTMS/ETCS 中的轨道至列车的数据传输。因此，仍然需要读取固定欧洲应答器。除了特定的特殊命令，其他所有应答器数据都不做解释。

1.4 欧洲标准应答器的应用前景和研究意义

应答器是现代铁路通信信号中非常重要的基础设施，在欧洲 ETCS 中的各个级别均有运用。我国的应答器系统也已经研制成功，并在城市轨道交通、驼峰场、及铁路干线的部分线路上得到运用。这是一种目前铁路通信信号系统中迫切需要的的设备，也是未来的高速铁路上急需的设备。欧洲标准应答器是铁道部重点引进的项目之一，欧洲应答器已经开始在秦沈线中实际运营。铁道部为了进一步进行线路提速，拟在多条线路上公开招标欧洲标准应答器项目，总合同额达上亿人民币，应用前景十分广阔。

目前国内的应答器在标准上与欧洲的有所差别，传输的数据量也与欧洲标准有一定的差距，同时在未来也存在互联互通的问题。为了在掌握应答器的核心技术，振兴民族产业，与世界先进技术接轨，在今后的设备招投标等过程中为国家节省宝贵的资金并取得主动，需要对该项欧洲标准的技术进行深入的研究和开发。大容量点式应答器的研究主要针对欧洲标准的应答器的核心技术进行相关方面的前期研究，包括该标准下的各种频率、传输速率、各个物理层及逻辑层的关系、相关数据的格式与定义、以及各个相关的接口方式。其中，车地间的相关频率、传输速率及相应的接口是该项目的基本点也是重点；然后初步研制出适用于中国铁路、符合 CTCS 要求的大容量点式应答器样机。

可以预测，欧洲标准应答器的国产化，将带来巨大的经济效益。首先，国产应答器将拥有自主知识产权，打破国外技术垄断，其研究成果可为全路服务；其次，其价格将低于国外同类产品，系统应用于工程将为国家节约巨额外汇；最后，售前、售后服务本地化，既有利于提高服务质量，又可降低服务成本。

1.5 论文内容及作者所完成的工作

本文首先介绍了欧洲标准应答器的报文格式、编解码标准和测试标准，在理论上深入讨论了编码的性能和实现方案。并在此基础上提出译码电路的硬件解决方案，并使用 FPGA 器件实现。同时对地面信号的发生部分作了一定的研究和探索。 论文章节安排如下：

第一章简要介绍了欧洲标准应答器的指标、性能和用途以及应用前景和研究意义。

第二章主要研究了欧洲标准应答器的编码标准，介绍了编码的各部分算法和测试方案，并讨论了这种编码方式的性能和实现方法。

第三章研究了欧洲标准应答器的译码标准，提出译码电路的硬件解决方案，以及 FPGA 的实现方法。

第四章给出了欧洲标准应答器的 FSK 信号特性，据此提出了几种调制方案，最后利用所指定的参数间的规律和关系，实现数字调制。

第二章　FFFS 编码策略、实现方案及其性能讨论

这一章主要阐述编码的算法和实现过程，并对编码的两个重要指标：最小距离和错误不可检出概率作了较深入的分析。

2.1　关于循环码

2.1.1 二元域多项式和求余运算

由于所有编码都是二进制的，这里仅讨论二进制编码理论中所涉及的二元域 GF(2)上的多项式，即系数取自 GF(2)上的多项式。在这里，我们把任何一个 n 位的二进制数 $v = [v_{n-1}, v_{n-2}, \cdots\cdots, v_1, v_0]$和一个多项式 $v(x) = v_{n-1}x^{n-1} + v_{n-2}x^{n-2} + \cdots + v_1x + v_0$一一对应。

在二元域中，只有 0 和 1 两个元素，两种代数运算分别为模 2 相加和模 2 相乘，其定义如下所示。

+	0	1
0	0	1
1	1	0

×	0	1
0	0	0
1	0	1

这样，可以在上述的加法和乘法下定义公式 $r(x) = R_{c(x)}[d(x)]$，用以表示 d(x)除以 c(x)的余式，即

$$d(x) = q(x)c(x) + r(x) \qquad (2.1)$$

并且，对于任何一个二进制多项式 f(x)有

$$f(x^2) = f(x)^2 \qquad (2.2)$$

2.1.2 循环码的特性

1、FFFS 编码策略给出的编码方式是一种循环码，对于一个循环码多项式

$$c(x) = c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_1x + c_0$$

若将其循环 i 次所得到的码矢记为 $c^{(i)}(x)$，则

$$c^{(i)}(x) = c_{n-1-i}x^{n-1} + c_{n-2-i}x^{n-2} + \cdots + c_0x^i + c_{n-1}x^{i-1} + \cdots + c_{n-i}$$

由循环码的特性可以得知

$$c^{(i)}(x) \equiv x^i c(x) \quad (\text{模 } x^n + 1) \qquad (2.3)$$

2、根据循环码的循环特性，可由一个码字的循环特性移位得到其它的非 0 码字。在(n, k)循环码的 2^k 个码字中，取前 k-1 位皆为 0 的码字 g(x)(其次数 r=n-k)，再经过 k-1 次循环移位，共得到 k 个码字：g(x)，xg(x)，…，

$x^{k-1}g(x)$。这 k 个码字显然是相互独立的，可作为生成矩阵的 k 行，于是得到(n，k)循环码的生成矩阵 G(x)

$$G(x)=\begin{pmatrix} x^{k-1}g(x) \\ x^{k-1}g(x) \\ \cdots \\ xg(x) \\ g(x) \end{pmatrix} \qquad (2.4)$$

由于码字可以由生成矩阵唯一确定，因此，（n，k）循环码可由它的一个（n-k）次多项式 g(x)来确定。所以称 g(x)为码的生成多项式。

可以证明，g(x)是唯一的，且为 x^n+1 的因式，即存在 k 次多项式 h(x)，满足

$$x^n+1=h(x)*g(x) \qquad (2.5)$$

满足上式的多项式 h(x)称为循环码的监督多项式。

3、设 H 为（n，k）循环码的一致监督矩阵

$$H=[h_{n-k-1}\ ,\ h_{n-k-2}\ ,\ \cdots,\ h_1\ ,\ h_0\ ]^T$$

其中 h_j (j = n-k-1，n-k-2，…,0)是 H 的行矢量。又设 C 为任一码字，R = (R_{n-1} ，R_{n-2} ，…，R_0)未接收矢量，其伴随式 $S^T=HR^T$ 的各行分量为

$$S_j=h_j R^T=\sum_{i=0}^{n-1}h_{j,i}R_i$$

$$j=n-k-1,\ n-k-2,\ \cdots,\ 0$$

如果 R 为一码字，则 S_j 为 0；如果 S_j \= 0，R 也一定不是一个码字。

如果 E = （E_{n-1} ，E_{n-2} ，…，E_0）为 R 的信道错误图样，由于 R = C + E，$HC^T=0^T$，所以

$$S_j=h_j E^T=\sum_{i=0}^{n-1}h_{j,i}E_i$$

$$j=n-k-1,\ n-k-2,\ \cdots,\ 0$$

因此，接收矢量的 S_j，实际上是对信道错误图样进行监督，而与发送的具体码字无关[4]。

由此，可以构造循环码的大数逻辑译码器[4]，如图 2.1 所示。第三章中译码器所使用的奇偶校验电路，就使用了类似的结构，只是不做纠错处理。

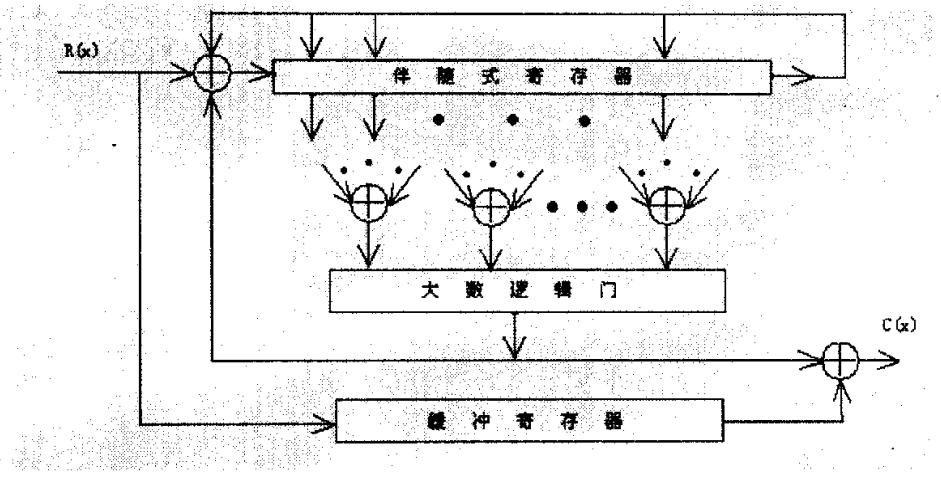

图 2.1　循环码大数逻辑译码器

2.2　报文格式和编码方式概述

2.2.1 欧标应答器编码的突出优点

FFFS 编码策略（Form Fit Function Specification Coding Strategy）是由 EUROSIG 联盟为推进欧洲铁路信号标准化制定的标准。它定义了欧洲标准应答器的报文格式，有如下突出的特点[2]：

1、兼容两种报文长度：1023 比特和 341 比特。

2、大量自由的信息位：分别为 830 比特和 210 比特（其中的某些位将被预留，作为报文传送系统的更高级的应用，如区分上行或下行报文）。

3、对于不同种类的传输错误具有可证明的安全保证；

4、报文的所有位反转后仍然可以被译码器识别；

5、不需要在报文的开始传送起始帧（或结尾帧）。

6、对于未知的将来的格式变化具有兼容性。

报文格式在设计时就被证明，对于随机的位错误、突发错误、丢位、插位以及这些错误的组合，此种编码方式都能够提供完全安全的防范。同时，它也特别注意了报文变化和报文误解等潜在的问题[2]。

然而，上述的所有的安全保证只有在使用明确指定的译码器时才会产生作用，因此，本文所讨论的译码器，都是根据 FFFS 编码策略中所规定的译码标准设计的。

2.2.2 报文格式

报文格式如图 2.2 所示。有两种兼容的格式：长度 $n_L = 1023$ (=93*11)的长格式和 $n_S = 341$ (= 31*11)的短格式。最左边是高位，定义为 b_{n-1}，最右边是低位，定义为 b_0[2]。应答器发送数据的顺序是从左至右，但并不一定从最左边的一位（b_{n-1}）开始，而可能是报文中的任何一位[2]。

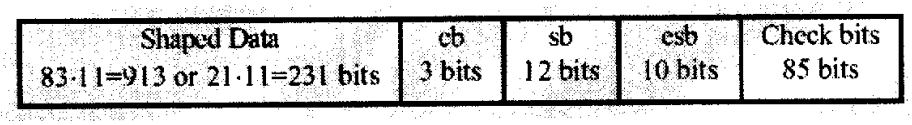

图 2.2 报文格式

报文的开头是一块“成型的数据”，是经过加扰和扩位后的用户数据。在长格式报文中，这个块包含 913 位（83 个 11 位字段）；短格式报文中，包含 231 位（21 个 11 位字段）。这样，一个长报文就包含了 830 位用户位，一个短报文包含了 210 位用户位。

下面的 3 个比特，$b_{109}...b_{107}$ 是控制位（cb）。第一个控制位 b_{109}，是一个反转位，通常置 0，如果置 1，那么在通过了校验位检验后，码矢的所有位必需反转，才能做下一步运算。值得注意的是，为了满足这一特性，编码和译码规则分别在扩位和同步上作了特殊的设计，在后面的 2.3.3 章节和 3.1.3 章节中分别有详细的讨论。另外 2 个控制位，b_{108} 和 b_{107}，当前没有使用，预留作将来的格式变更时使用。对于当前的格式，这两位备用位必须被置为 $b_{108} = 0$，$b_{107} = 1$。其他的值表明此码无效。

控制位后的 12 比特，$b_{106}...b_{95}$，是扰码位（sb），用来储存加扰前对数据进行操作的扰码器的初始状态（这个状态值必须送给解扰器，才能正确解扰）。

再下面的 10 位，$b_{94}...b_{85}$，是附加形成位（esb），它使得对于校验位的定型限制独立于加扰。除非译码时要求对定型限制做校验，否则译码器对这 10 位数据不做任何处理（收到后就丢弃）。

最后的 85 位，$b_{84}...b_0$，是校验位，包含错误检测码的 75 位奇偶位和用于

同步的 10 位[2]。

报文可能从 b_{n-1} 至 b_0 的任何一位开始被发送，一帧传送完毕后，从 b_{n-1} 开始再循环传送，直至地面应答器能量消失。因此，车载天线收到的最后一比特信息可能也是 b_{n-1} 至 b_0 中的任何一位。

2.2.3 编码方式概述

FFFS 规定的编码标准，经过了严格的数学推导和证明，使得这种编码有着多种优点。由于编码不需要在现场实时地完成（但译码需要），因此尽管编码和译码的算法结构非常相似，但它们是不对称的。编码过程中，相当大的工作量耗费在检测生成的码元是否符合编码规则上，只有通过这些测试，此编码才具有 1.2 中所述的良好性态，否则，需要重新编码。而译码过程相对容易（因为这里已经假定收到的编码是完全符合规则的），因此译码比编码需要的时间短的多。

编码的步骤由以下七步组成[2]：

1、选择 12 位扰码位。

2、对用户位做加扰运算（运算依赖于第 1 步中的扰码位）。

3、将加扰了的数据扩位，即按 10 比特分段，每一段映射成 11 位的字段。

4、检验形成条件，如果条件不满足，回到第 2 步。

5、选择 10 位附加形成位。（如果所有 2^{10} 个组合都已无法使用，回到第 1 步）。

6、形成校验位。

7、检验形成条件。如果满足，则输出报文；否则，回到第 5 步。

从纯逻辑的角度看，步骤 4 中的测试是可以省略的；步骤 7 会完成所有的测试。出于效率的原因，尽可能快的丢弃候选报文，并在只改变附加形成位的“5-6-7”内作循环是更优越的。

每一个报文都是一个循环码，并提供了大量的保护措施，以防止随机位错误和突发错误的发生。10-11 位的转换进一步防止了丢位、插位和连续长 0、长 1 的发生。4-7 步的测试可以排除掉存在潜在的丢位可能的报文，同时也排除位模式和短报文过于接近的长报文。加扰运算可以保证，对于给定的用户数据，可以产生足够多的可替代的候选报文，以至其中的一个最终能够通过测试。

然而，理论上可能存在某些由于形成位被耗尽而无法编码的用户数据，这

样的概率非常低（对于随机的数据，低于 10^{-100}）。如果真的发生了，那么只要对用户数据作微小的变动（比如将速度限制值降低 1km/h）就可以让这个数据的编码可实现。需要指出的是，在一个报文编码通过所有测试前，生成的候选报文的数量是非常巨大的。相反，接收器的工作就相对简单快速。

以下将详细讨论 FFFS 编码策略，包括如何加扰、扩位、计算校验位和测试报文等。

2.3 编码步骤

2.3.1 加扰运算

为了使码字具有随机性，必须对原始的用户码进行加扰运算。FFFS 编码策略中提供的加扰运算包括 3 步[2]：

1、将用户码的头 10 个比特替换为所有比特按一定规则运算得到的结果。

2、通过 12 位的扰码位计算 32 位的整数 S。

3、通过初始状态 S，利用 32 位的线性反馈移位寄存器进行加扰运算。

下面对每一个步骤作详细的阐述：

第一步：对于长报文，有 m = 830 位用户位；短报文，有 m = 210 位用户位。将用户位从左至右分成 k 个 10 位的块 $U_{k-1} = (u_{m-1}, \cdots, u_{m-10})$，$U_{k-2} = (u_{m-11}, \cdots, u_{m-20})$，$\cdots$，$U_0 = (u_9, \cdots, u_0)$，对于长报文，k = 83，短报文，k = 21。对 U_{k-1} ，$\cdots$ $U_{k-2}\cdots U_0$ 作如下运算：

$$U'_{k-1} = \sum_{i=0}^{k-1} U_i \bmod 2^{10}, \tag{2.6}$$

用 U'_{k-1} 代替 U_{k-1}，这样，就形成了新的序列 u'_{m-1} ，u'_{m-2} ，$\cdots$，u'_0，这个新序列中，除了头 10 位，其他均与原来的用户位相同。

第二步：通过 12 位扰码位（即 2.2.1 章中所说的 sb）计算 B：

$$B = b_{106} \cdot 2^{11} + b_{105} \cdot 2^{10} + \cdots + b_{96} \cdot 2 + b_{95} \tag{2.7}$$

32 位的整数 S 按如下公式计算：

$$S = (2801775573 \cdot B) \bmod 2^{32}$$

其中常数 $2801775573 = 69069^3 \bmod 2^{32}$ ；对于这种类型的随机数生成器，69069 是一个常用的选择。

第三步：使用如图 2.3 所示的移位寄存电路，图中，正方形表示寄存器，加号表示异或操作。系数 h_{31}，h_{30}，h_{29}，h_{27}，h_{25} 和 h_0 等于 1，其他所有的系数都为 0。S 以二进制的形式作为 32 位寄存器的初始值（最左端是 msb，右端是 lsb）。电路经过 m-1 次运算，输入是 u'_{m-1}，u'_{m-2}，…，u'_0，生成的扰码序列 s_{m-1}，s_{m-2}，…，s_0 串行输出。

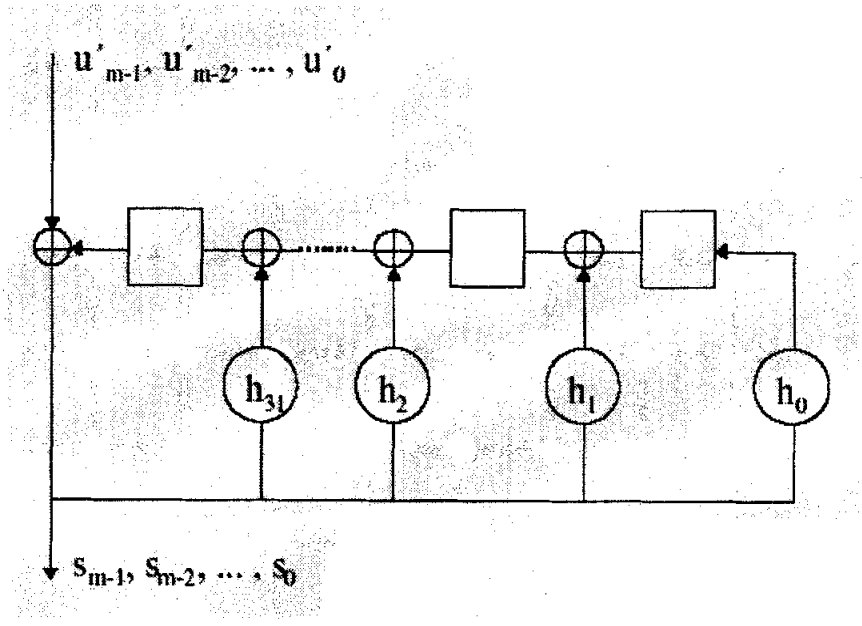

图 2.3　扰码生成器

2.3.2　扩位——10 到 11 位的变换

用户位加扰后，形成了扰码序列 s_{m-1}，s_{m-2}，…，s_0，我们把扰码序列分块，每块 10 位，即(s_{m-1}，s_{m-2}，…，s_{m-10})，(s_{m-11}，s_{m-12}，…，s_{20})，…，(s_9，s_8，…，s_0)。这样，长报文就有 83 个块，短报文有 21 个块。每一个 10 位的块将由一个置换表映射成一个 11 位的块。这 1024（2^{10}）个置换值在附录 A“10-11 字段转换表”中列出[2]。

置换规则为：每一个 10 位的块，作为一个 10 位的二进制数，被换算成一个整数 i（在 0-1023 间），于是这个块就对应了置换表中的第 i 个字段（一个 11 位字段，同时，表中字段的序号是从 0 开始的，到 1023 结束）。这样，表中第 i+1 个字段就一定比第 i 个字段的值要大。

扩位变化具有多个目的。

第一，是为了使码字具有更好的随机性，避免出现长0和长1。在这里，最多允许连续出现8个0或1。这是因为，在字段转换表中，以下的250个字段都是无效的（即表中没有如下数据出现）。

00000xxxxxx，（x表示0或1，因此代表2^6个不同的字段，下同）

xxxxxx00000，

10000000001，

11111xxxxxx，

xxxxxx11111，

011111111101。

第二，因为以下的20个字段都是无效的，所以附加位（即3.1章中关于译码算法第1步中的r）和处理窗口中前面那些相应的位之间的最多4次移位就可以被接收器检测到。

01010101010，

10101010101；（1）

00100100100, 01001001001, 10010010010,

11011011011, 10110110110, 01101101101；（2）

00010001000, 00100010001, 01000100010, 10001000100, 00110011001, 01100110011, 11101110111, 11011101110, 10111011101, 01110111011, 11001100110, 10011001100。（3）

注意到这3组数据的特性：在任何一个组中，总存在这样一个数据B，使得同组的任何一个数据A去掉头i位（或尾i位，1<=i<=4），在其右端（或左端）补齐数据B的头i位（或尾i位）后，得到数据C也在这个组中。这里数据A、B和C并不互异。因此以上的20个数据的弃用，可以保证不存在这样的编码：循环左移（或右移）4位后，仍与原编码相同。

第三，这个转换表（表是按经验选取的）支持不同步分析条件（不同步条件将在下文2.4.2章中详述）。特殊情况下，如果对任何一个有效字段左移或右移一位，移空的位置补上任意的一位，是不太可能产生一个有效字段的。所有字段中，左移一

位后在移空的位置上补上任意一位后，形成的新字段仍然有效的，只有316个；右移一位后在移空的位置上补上任意一位后，形成的新字段仍然有效的，只有324个。

第四，这个转换表支持反转，即对表中的任何一个字段取反，在表中仍可以找到相对应的字段。这个特性和2.2.1章编码格式中介绍的反转位（b_{109}）的定义是一致的，如果反转位为1，那么就要将当前码矢的所有位取反，再进行缩位的操作。这条限制保证了不会因为取反产生无效的字段。

2.3.3 计算校验位

在进行了加扰运算、扩位变换和选择了合适的附加形成位（esb）$b_{94}...b_{85}$后，候选报文的 $b_{n-1}...b_{85}$ 就已经确定。下面就要计算校验位 $b_{84}...b_0$。如果以多项式表示，校验位按照如下公式定义[2]：

$$b_{84}x^{84}+...+b_1x+b_0 = R_{f(x)g(x)}[\, b_{n-1}x^{n-1}+...+b_{85}x^{85}\,] + o(x) \tag{2.8}$$

可以看出，这是一个以 g(x)为生成多项式的循环码，f(x)在译码时作同步用，将在第三章详细讨论。

这里，多项式 f(x)，g(x)和 o(x)都取决于报文的格式。对于长格式报文，$f(x) = f_L(x)$，$g(x) = g_L(x)$，$o(x) = g_L(x)$：

$$f_L(x) = x^{10} + x^9 + x^7 + x^6 + x^4 + x^3 + x^2 + x + 1$$

$$\begin{aligned} g_L(x) = {} & x^{75} + x^{73} + x^{72} + x^{71} + x^{67} + x^{62} + x^{61} + x^{60} + x^{57} \\ & + x^{56} + x^{55} + x^{52} + x^{51} + x^{49} + x^{46} + x^{45} + x^{44} + x^{43} \\ & + x^{41} + x^{37} + x^{35} + x^{34} + x^{33} + x^{31} + x^{30} + x^{28} + x^{26} \\ & + x^{24} + x^{21} + x^{17} + x^{16} + x^{15} + x^{13} + x^{12} + x^{11} + x^9 \\ & + x^4 + x + 1. \end{aligned}$$

对于短格式报文，$f(x) = f_S(x)$，$g(x) = g_S(x)$，$o(x) = g_S(x)$：

$$f_S(x) = x^{10} + x^8 + x^7 + x^5 + x^3 + x + 1$$

$$\begin{aligned} g_S(x) = {} & x^{75} + x^{72} + x^{71} + x^{70} + x^{69} + x^{68} + x^{66} + x^{65} + x^{64} \\ & + x^{63} + x^{60} + x^{55} + x^{54} + x^{49} + x^{47} + x^{46} + x^{45} + x^{44} \\ & + x^{43} + x^{42} + x^{41} + x^{39} + x^{38} + x^{37} + x^{36} + x^{34} + x^{33} \\ & + x^{32} + x^{31} + x^{30} + x^{27} + x^{25} + x^{22} + x^{19} + x^{17} + x^{13} \\ & + x^{12} + x^{11} + x^{10} + x^6 + x^3 + x + 1. \end{aligned}$$

在解码时，一旦收到的信息位满足：

$$R_{g(x)}[b_{n-1}x^{n-1}+\cdots+b_0x^0]=0$$

即表示此 n 位信息满足校验条件，有可能是信息码元，应作进一步判断。

另外，注意到 $g_L(x)$和 $g_S(x)$满足如下关系

$$R_{gL(x)}[g_S(x)\cdot(x^{682}+x^{341}+1)]=0 \qquad (2.9)$$

2.9 式说明，连续重复 3 次的短格式报文，满足长格式报文的奇偶校验条件。

2.4 测试候选报文

每一个报文都必须满足下面的 4 个条件。如 2.1 章中所述的，如果一个候选报文不能满足所有的条件，它将被丢弃；然后通过改变附加形成位（这种做法仅仅改变校验位）、改变扰码位或改变用户数据（改变整个报文）来获得新的候选报文。

2.4.1 转换表条件

对于报文中所有 11 位的字段 $b_{i-1}\cdots b_{i-11}$，如果 i 是 11 的整数倍，那么这个字段必须是有效的（即在附录 A 的列表中存在）。很明显，这个条件在作扩位转换时，已经自动满足[2]。

2.4.2 不同步条件

不同步条件要求对这样的 11 位字段序列进行测试：（$b_{i-1}\cdots b_{i-11}$），（$b_{i-12}\cdots b_{i-22}$），（$b_{i-23}\cdots b_{i-33}$），…其中，i **不是** 11 的整数倍。如果 i + 1 或 i – 1 是 11 的整数倍，那么这个序列中连续的有效的字段的个数不能大于 2；如果 i、i + 1 或 i – 1 都不是 11 的整数倍，那么对于长报文，这个序列中连续的有效的字段的个数不能大于 10，短报文不能大于 6[2]。

注意，这个条件在编码过程中的任何一步都不是自动满足的。

2.4.3 长报文的非周期条件

只有长报文需要满足这个条件。即便在噪声和位丢失的情况下，这个条件可以通过测试两个相隔了 341 位的 11 位字段序列的汉明距离，来确保一个长报文的任何一部分都不会被译码器误当作是短报文[2]。

对于每一个是 11 整数倍的 i：

a) $b_{i-1}\cdots b_{i-11}$ 和 $b_{i-341-1}\cdots b_{i-341-22}$ 之间的汉明距离至少为 3

b) 对于 k = +1，-1，+2，-2，+3，-3，$b_{i-1}\cdots b_{i-11}$ 和 $b_{i-341-k-1}\cdots b_{i-341-k-22}$ 之间的汉明距离至少为 2。

2.4.4 欠采样条件

欠采样因子为 2 的欠采样（例如，把报文重新排列成了 b_{n-2}，b_{n-4}，…，b_1，b_{n-1}，b_{n-3}，…，b_2，b_0）会导致产生一个新的循环码序列，这样，通过检验校验位并不能发现这种错误。因为这种序列的改变可以由硬件的缺陷造成（如采样时钟慢了 1 倍），因此每个报文都必须经过测试，以确保这种序列变化不能满足变换表条件（条件 1）。

令 $v_j = b_{j2^k}$，j = 0，1，…，n-1（表示发生了欠采样因子为 2^k 的欠采样）。对于 k = 1，2，3，4，对于任何的 i，序列（$v_{i-1}\cdots v_{i-11}$），（$v_{i-12}\cdots v_{i-22}$），（$v_{i-23}\cdots v_{i-33}$），…中有效字段的个数不能大于 30。

注意到短报文的 11 位字段个数是 31，因此可以看出，这个条件使得发生了欠采样的短报文，和一个发生了欠采样的长报文的不超过 341 位的部分，即使任意划分字段，都无法满足转换表条件（条件 1）[2]。

2.5 编码的实现方案

对于固定应答器，码字的生成不需要在现场完成，只要预先将码编好，烧录到 ROM 中。应答器收到来自列车天线的信号时，把码元从 ROM 中读取出来，调制成 FSK 信号，发送给车载天线。因此，比较合适的编码办法是使用计算机，编制软件来实现。

图 2.4 给出了编码的流程图，建议使用 C 语言实现。

在本设计中，并没有完全按照 2.2 所示的流程图进行编码，而是只完成了前 4 步，即仅对用户数据进行编码，而不做条件测试。编码使用 VHDL 语言完成。这样做是出于 3 方面的考虑：

首先，使用 VHDL 实现的编码是基于硬件模型的，其中的加扰、扩位和校验位计算模块分别和译码电路中的解扰、缩位和校验位检验模块互为逆运算，这样就很好地验证了译码电路中各个模块的正确性；

其次，VHDL 是硬件描述语言，并不适合做算法或运算。软件编码工作量巨大，而本文主要讨论点式车地信息无线传输系统数字编码和解码的 FPGA 实

现，即硬件电路部分，因此不对编码过程作深入研究。这里用 VHDL 产生的编码，仅仅为测试译码电路用。

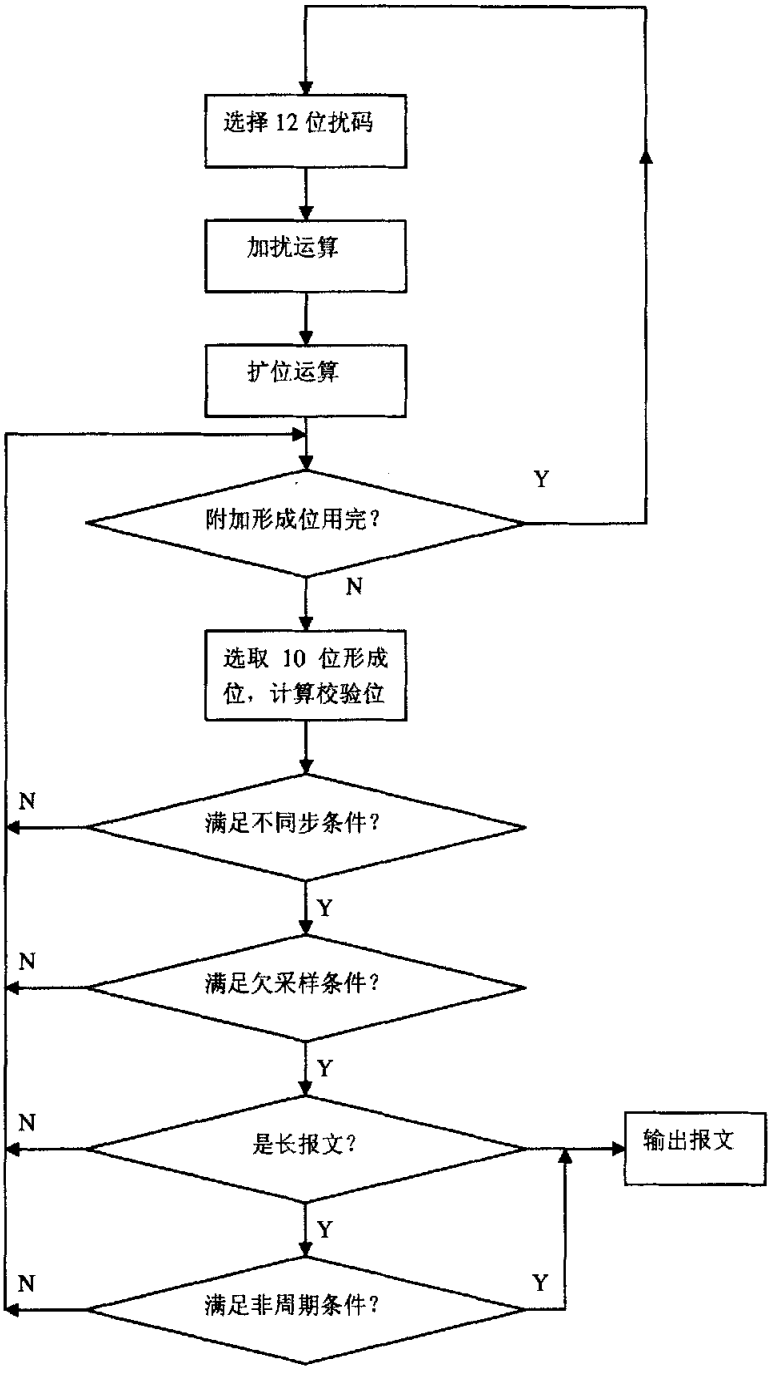

图 2.4　软件编码流程图

最后，如果令通过了条件测试的编码的集合为 A，未经过条件测试的编码的集合为 B，那么 A⊂B。因此如果译码电路可以对任意 b∈B 进行译码，那么一定可以对任意 a∈A 译码。

附录 B 中给出了几组和用户数据一一对应的编码[3]。在测试过程中，对编码做任意位的循环移位，译码器都应可以译出唯一对应的用户数据。

2.6 编码的性能讨论

2.6.1 码矢的最小距离

由 2.3.4 章可知，这是一个以 g(x)为生成多项式的（n，k）循环码，其中，n 为码矢的长度，k 为信息元的长度（长报文中，n 为 1023，k 为 938；短报文中，n 为 341，k 为 256），

那么存在 k 次多项式 h(x)，使得

$$x^n + 1 = h(x) * g(x)$$

这一点，可以通过程序进行检验。

令 $r = n - k$，

$$h(x) = h_k x^k + h_{k-1} x^{k-1} + h_{k-2} x^{k-2} + \cdots + h_1 x + h_0,$$

$$g(x) = g_r x^r + g_{r-1} x^{r-1} + g_{r-2} x^{r-2} + \cdots + g_1 x + g_0,$$

则有

$$\left.\begin{aligned} &x \text{ 的系数 } = h_1 g_0 = 0 \\ &x^2 \text{ 的系数 } = h_1 g_1 + h_2 g_0 + h_0 g_2 = 0 \\ &\quad \vdots \\ &x^{n-j} \text{ 的系数 } = \sum_i g_i h_{n-2-i} = 0, \end{aligned}\right\} \quad (2.10)$$

其中

$$0 <= i <= r-1 \text{ 且 } n-j-k+1 <= i <= n-j,\ 1 <= j <= n-1$$

将式 2.5 改写成矩阵形式

$$\begin{pmatrix} 0\,0\,0 \cdots h_0\,h_1 \cdots h_{k-1}\,h_k \\ 0\,0\,0 \cdots h_1\,h_2 \cdots h_k\,0 \\ \vdots \\ 0\,h_0\,h_1 \cdots 0\,0 \\ h_0\,h_1\,h_2 \cdots 0\,0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ g_r \\ g_{r-1} \\ \vdots \\ g_1 \\ g_0 \end{pmatrix} = 0$$

这样，就得到了这个循环码的监督矩阵，即

$$H_{(n,k)} = \begin{pmatrix} 0\,0\,0 \cdots h_0\,h_1 \cdots h_{k-1}\,h_k \\ 0\,0\,0 \cdots h_1\,h_2 \cdots h_k\,0 \\ \vdots \\ 0\,h_0\,h_1 \cdots 0\,0 \\ h_0\,h_1\,h_2 \cdots 0\,0 \end{pmatrix}$$

因为 $x^n + 1 = h(x) * g(x)$，所以 $h_0 = h_k = 1$。最终可以得到监督矩阵如下：

$$H_{(n,k)} = \begin{pmatrix} 0\,0 \cdots 0\,1\,h_1 \cdots h_{k-1}\,1 \\ 0\,0 \cdots 1\,h_1\,h_2 \cdots 1\,0 \\ \vdots \\ 0\,1\,h_1 \cdots 0\,h_{k-1}\,1 \cdots 0\,0 \\ 1\,h_1 \cdots h_{k-1}\,1\,0 \cdots 0\,0 \end{pmatrix}$$

注意到 $H_{(n,k)}$共有 r 行 n 列，易得知前 r 列是线性无关的，而前 r+1 列线性相关，根据编码理论关于线性码的定理[4]，可以推知码的最小距离为 r+1，即 76。

码的最小距离是衡量码的抗干扰能力（检、纠错能力）的重要参数，码的最小距离越大，码的抗干扰能力就越强。

由于（n，k）线性码能纠 t 个错误的充要条件是码的最小距离为[4]

$$d_{min} = 2t + 1$$

能发现 s 个错误的充要条件是码的最小距离为[4]

$$d_{min} = s + 1$$

因此这种码最多可以纠正37个错，发现75个错。对于长度最多为1023的码矢来说，这是一个相当高的纠（检）错率。

2.6.2 在BSC中的不可检测错误概率P(E)

考虑到码矢发生较少错误的概率较大，并且码长和最小距离已知，因而可用如下公式计算P(E)

对于一个长度为n，BSC转移概率为p的报文，发生i个错误的概率P(I)为：

$$P(I) = \binom{n}{i} p^i (1-p)^{n-i}$$

由于码的最小距离为76，因此，当报文中错误的比特输i满足$76 \leqslant i \leqslant n$时，错误无法被检测出。故在BSC中的不可检测错误概率P(E)为

$$P(E) = \sum_{i=76}^{n} \binom{n}{i} p^i (1-p)^{n-i}$$

其中，n为1023（长报文）或341（短报文）。

但是，这个式子的值并不易计算。不过我们可以只计算（n，k）码所有码字集合未检出错误的平均概率的上限[4]，即

$$P(E) < 2^{-(n-k)} = 2^{-75}$$

第三章　译码算法及其 FPGA 实现

3.1　译码器框图

FFFS 编码策略中也给出了建议的译码算法，总共包括 11 步，其中对于长格式报文，n=1023；短格式报文，n=341。

在本文讨论的译码器中，一个译码器必须包含 2 部分独立的电路——一部分译码长格式报文，另一部分译码短格式报文，两套译码器相互独立，且互不兼容。最终输出的结果由译码成功的那一部分提供，如下图所示：

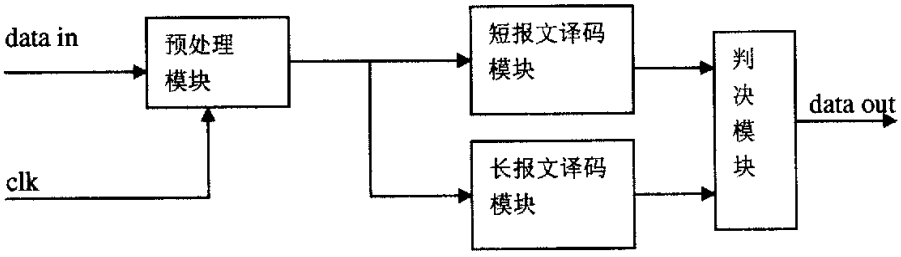

图 3.1　译码算法示意图

3.1.1 译码步骤

11 步的译码步骤如下所示[2]：

1、一个寄存器中存放 n+r 位连续的接收到的信息位（长格式报文：r=77；短格式报文：r=121。如果寄存器已经被移位 7500 位，则令 r=n）。

2、满足奇偶校验吗（即寄存器内的前 n 位，如果看成一个多项式，是否能被 g(x)整除）？如果不，寄存器移位（即最高位移出，最低位补下一个信息位），回到第 1 步。

3、n+r 位寄存器内的前 r 位和最后 r 位是否完全一致？如果不是，寄存器移位，回到第 1 步。

4、根据同步算法(将在下文中详细阐述)，找出报文的开头(即 b_{n-1} 的位置)，如果同步失败（即 $R_{f(x)}[v(x)]$是一个不可能的值），回到第 1 步。

5、检测是否所有 11 位的字段都有效(b_{n-1} …b_{n-11})，(b_{n-12} …b_{n-22})，…，(b_{10} … b_0)，如果不是，寄存器移位，回到第 1 步。

6、此时，报文被认为正确有效。

7、检测反转位 b_{109} 是否为 1，如果是，对报文的所有位取反。

8、检测另外 2 位控制位 b_{108} 和 b_{107}。在当前的应用中，这 2 位被固定设置

成01，其他组合作为将来的预留格式。因此如果 b_{108} 为 1 或者 b_{107} 为 0，则放弃报文并给出信息“无法识别的报文格式”。

9、把报文的所有 11 位字段，转换成 10 位字段。

10、 解扰（解扰算法将在下文中详述）。

11、 输出用户位和反转位 b_{109}。

下面对译码的 11 个步骤作必要的分析和解释。

1、对于第 1 步，短格式报文所需的附加存储位 r 大于长格式报文的 r，这是因为必须排除掉把长报文误认为是短报文的可能。当检测了 7500 位信息位却没有发现一个没有错误的报文时，令 r=n 的目的是获得每一个应答器通道未能检测出错误的概率的固定上限（当 r=n 时，有错却未能检出的概率是 0）。数字“7500”某种程度上是一个经验值。

2、对于第 2 步，所谓“寄存器移位”，一定是在还有附加的有效信息位（已收到，未处理）的情况下才能实现。一次移位可以移 1 位，也可以移几位。

3、测试附加的 r 位有多个目的。对于短报文，这将保证能够排除长报文；对于两种格式的报文，这是一个对丢位和插位基本检测。

4、第 4 步中，所谓“不可能的值”，对于一个无错的短报文，1024 个值（一个 10 位的无符号型二进制数）中只有 341 个是有效的。对于一个使用了长报文算法的译码器（只能对长报文译码），一个无错的长报文是不会使 $R_{f(x)}[v(x)]$ 为 0 的，而一个重复的无错的短报文却总能使 $R_{f(x)}[v(x)]$ 为 0。

5、在一个实际的执行中，第 5 步自然地是和第 9 步结合在一起的。所有字段都要进行检测，而不仅是报文中的定型数据部分。

FFFS 编码策略建议译码器纪录进入每一个不同的“回到第 1 步”的程序分支的频率。举例来说，我们推测除了第 2 步和第 4 步，从其他分支“回到第 1 步”几乎不会发生。知道这种来自译码器工作时的统计数据，对于未来的安全回顾是非常有意义的。

译码过程中，涉及到奇偶校验、同步和解扰的算法，下面给出这几种计算的数学依据和电路实现办法。

3.1.2 奇偶校验算法

根据 2.8 式，一个正确的码流 b(x)应该满足 $R_{g(x)}[b(x)]=0$，这就是校验算法的数学依据。我们要对码流作对 g(x)的整除运算，一旦余数为 0，则说明这是一个可能正确的码流，需要对其进行进一步运算。

对于求余运算，可用除法电路来实现如图 3.2 所示，这是一个反馈移位寄存器。移位寄存器的级数 r 等于除式 g(x)的次数，反馈抽头由除式的各项系数 g_i 决定。当电路完成 r+1 次移位时，寄存器内存放的即为求余的结果。

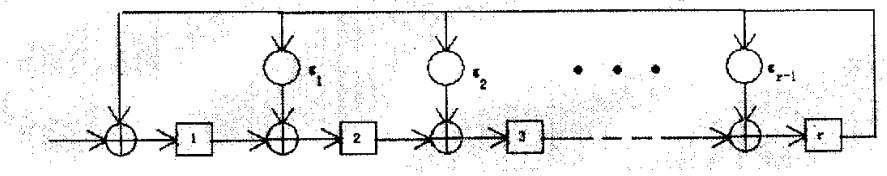

图 3.2 除式为 g(x)的除法（求余）电路

但是奇偶校验并不对循环码作纠错运算，只对其能力范围内的数量的错误进行检查。因此，必须消除收到的码矢中的错误对求余运算带来的影响。如果希望任何一个 n 位的完全正确的序列能够通过校验位检验，那么这 n 位之前的比特流对 g(x)的求余运算，结果应该是 0。也就是说，当一个长度为 n 的寄存器中的码元经校验位模块检验后，发现余式不为 0，那么就认为这 n 位的序列中存在错误，发生错误的位及其前面所有的位都将被弃用（因为这不是一个长度能够达到 n 的正确码流）。

由于 $R_{g(x)}[x^n+1]=0$，故

$$R_{g(x)}[x^n]=1$$

这个公式说明，一个 n 位的经校验位检验后余式不为 0 的序列，如果其最高位为 1，那么只需要对余式的最低位取反，就可以“消除”最高位对 g(x)整除产生的余项。序列的最高位为 0 时，不产生余项。

3.1.3 同步算法

同步运算是整个编、译码系统中最核心的一部分，应答器较小的数据传输量和循环发送的特点，决定了构造这种不需要帧头或帧尾的编码格式的可行性。下面详细阐述在这种编码规则下，同步运算是如何实现的。

令 $v=[v_{n-1}, \cdots\cdots, v_0]$表示由应答器发送的比特流中长度为 n 的“一块”

（长报文和短报文的 n 分别为 1023 和 341）。块 v 是发送报文的循环移位，如图 3.3 所示。图中，$b=[b_{n-1}, \cdots\cdots, b_0]$是发送的报文，整数 s 是 v 与 b“错开”的位的长度，即 $b_{n-1-s}=v_{n-1}$。用多项式表达，就是 $v(x)=R_x^n{}_{-1}[x^s b(x)]$。

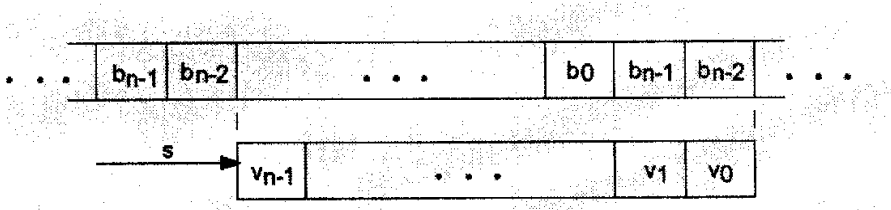

图 3.3 在一个长度为 n 的窗口中接收到的数据

为了计算 s（$0 <= s < n$），要计算集合 $s_f(x)$，表达式为：

$$s_f(x)=R_{f(x)}[v(x)], \qquad (3.1)$$

其中，对于长报文，$f(x)=f_L(x)$；短报文，$f(x)=f_S(x)$

下面要证明 $s_f(x)=R_{f(x)}[x^s o(x)]$ （3.2）

其中，$o(x)=g(x)$（g(x)的定义见 2.2 章编码策略）

前提： $R_{f(x)}[x^n+1]=0$（经过编程验证）

当 $s=0$ 时，$R_{f(x)}[v(x)]=R_{f(x)}[b(x)]=R_{f(x)}[g(x)]=R_{f(x)}[x^s o(x)]$

采用数学归纳法：

令 $v_k(x)=R_x^n{}_{+1}[x^k b(x)]$ $k=0, 1, \cdots\cdots, n-1$

假设 $R_{f(x)}[v_k(x)]=R_{f(x)}[x^k o(x)]$

则对于 $R_{f(x)}[xv_{k+1}(x)]$有：

1° $v_k(x)$首项为 0，则

$R_{f(x)}[v_{k+1}(x)]=R_{f(x)}[xv_k(x)]=R_{f(x)}[x^{k+1} o(x)]$

即相当于 $xv_k(x)$左移了一位

2° $v_k(x)$首项为 1，则

$R_{f(x)}[v_{k+1}(x)]=R_{f(x)}[v_k(x)x+x^n+1]$

由于 $R_{f(x)}[x^n+1]=0$，故

$R_{f(x)}[v_k(x)x+x^n+1]=R_{f(x)}[v_k(x)x]=R_{f(x)}[x^{k+1} o(x)]$

故 $R_{f(x)}[v_{k+1}(x)]=R_{f(x)}[x^{k+1} o(x)]$

因此，$R_{f(x)}[v(x)]=R_{f(x)}[x^s o(x)]$成立

这样，$s_f(x)$的值就唯一决定了 s 的值，以此实现了同步。

在实际的实现过程中，可以考虑由公式 $s_f(x) = R_{f(x)}[x^s o(x)]$事先生成一张 $s_f(x)$—>s 的映射表。在解码过程中，对收到的 v(x)，作 $R_{f(x)}[v(x)]$运算，从而得到 $s_f(x)$，然后通过查表得到同步需要进行的移位 s。

此外，可以证明 $R_{f(x)}[v(x)+x^{n-1}+x^{n-2}+\cdots+x^2+x+1] = R_{f(x)}[v(x)]$，

这个公式说明，如果把所有收到的信息位都取反，则以上同步公式依然成立，这从另一个方面验证了编码的算法，即 2.2.1 章中的反转位的功能。

对于一个没有错误的报文来说，式 3.2 的值是不会为 0 的。然而，对于任何一个周期为能被 n 整除的整数（但是必须大于 1），长度为 n 的向量 v(x)（即这个二进制序列的长度为 n，由 k 个相同的部分组成，每部分长度为 n/k，其中 k 为周期，1<k<=n），有：

$$R_{f(x)}[v(x)] = 0$$

有一种特殊的情况，即长报文解码器收到了循环的短报文，这时，v(x)的长度为 1023 位，由完全相同的 3 个 341 位组成，此时 $s_f(x)$的计算结果就是 0。因此，长报文解码器永远不会把循环发送的短报文误认为是长报文，以此保证了译码的可靠。

3.1.4 解扰算法

对比 2.2 章中的加扰算法，可以提出一套解扰算法，如图 3.4 所示。

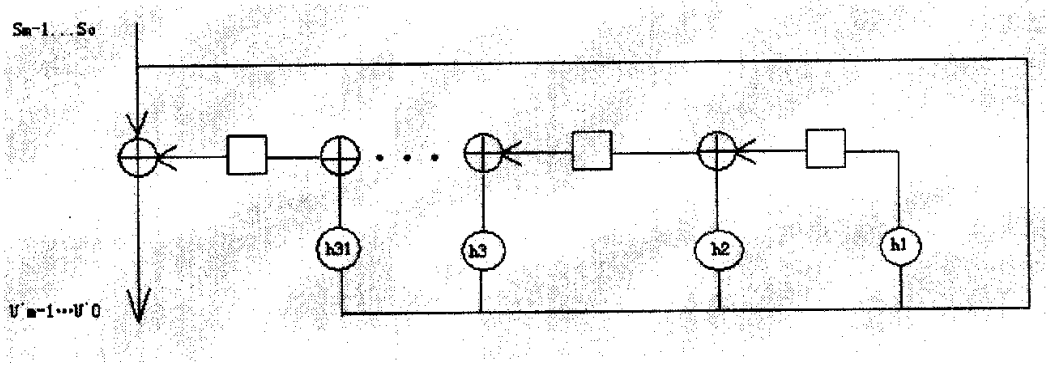

图 3.4　　解扰算法

图中，正方形表示寄存器，加号表示异或操作。系数 h_{31}，h_{30}，h_{29}，h_{27}，h_{25}和 h_0等于 1，其他所有的系数　都为 0。在进行解扰前，要先通过 12 位的 sb 计算 S，计算公式仍然为 2.2 式。之后 S 以二进制的形式作为 3.1 图中 32 位寄存器的初始值（最左端是 msb，右端　是 lsb）。电路经过 m-1 次运算，输入是 S_{m-1}，S_{m-2}，…，S_0，输出为 $\hat{u}_{m-1}$，$\hat{u}_{m-2}$，…，$\hat{u}_0$。

此时，除了 $\hat{u}_{m-1}$，报文与原始用户数据完全一样。回顾第 2 章中的 2.6 式，可以对 $\hat{u}_{m-1}$，$\hat{u}_{m-2}$，…，$\hat{u}_0$ 作如下运算：

$$U_{k-1} = \sum_{i=1}^{k-1} U_i^* \bmod 2^{10} \tag{3.3}$$

用 u_{k-1}，代替 $\hat{u}_{m-1}$，序列中的其余部分不变，即为原始用户数据。

3.2 译码过程中的特殊情况

译码过程中，会出现一些特殊情况，编码过程中对它们都已做了考虑，从而不会导致错误发生。

3.2.1 信道传输错误

码矢在信道中传播时，受干扰会产生各种各样的错误，如随机错误、突发错误、丢位和插位等。由 g(x)生成的循环码可以有效地防护随机错误和突发错误，正如 2.6.1 章中所讨论过的，译码器可以检测到 75 个以下的突发错误。而几个数量更小的突发错误的组合，也可以被检测。此外，在长度为 n+r 的寄存器内，总数不超过 3 次的丢位或插位（或者它们的组合），都可以被译码器检测到。

3.2.2 混合的格式

1、 长报文，短格式

接收器当短报文接收器收到长格式报文时，正如 2.4.3 章所讨论的，非周期条件会使得长报文的任何一部分都不会被当成短报文。3.1.1 章中，11 步译码步骤中的第 3 步，也会保证一个长格式报文在此无法通过测试。

2、 短报文，长接收器

一个循环发送的短报文，几乎会满足长报文的所有条件。2.4 式表明，3 个连续的短报文，可以通过长报文的校验位检验，而转换表条件、不同步条件和欠采样条件显然也满足。只有在做同步运算时，才能将这个“伪”长报文排除掉。3.1.3 章中已作详细阐述。

3.2.3 欠采样，过采样

欠采样的情况在 2.4.4 章节中已有论述。所谓采样因子为 k 的过采样，是指对 1 位数据连续 k 次的采样。这 2 种情况有可能发生在硬件有错误的时候。由于 2.2 式，循环码对于采样因子为 2 次方幂的欠采样和过采样，具有一种自身的缺

陷。因此必须考虑这 2 种错误情况。

编码过程中必需满足的“欠采样条件”可以保证在欠采样因子为 2，4，8 和 16 的情况下，都不会错误译码。

采样因子大于等于 8 的过采样，会导致 8 个或 8 个以上的连续的 0 或 1，这样就不会满足转换表条件。对于采样因子为小于 8 的偶数的过采样，将不会满足附加位条件，即 11 步译码步骤的第 1 步

3.3 硬件实现方案

译码方案完全按照 3.1 中的 11 个步骤进行。考虑到车载部分对数据处理速度要求较高，且对数据所有的译码和处理都是基于数字逻辑的运算，决定使用 FPGA 器件实现译码电路。同时，也只有 FPGA 器件可以在一个单片系统中对数据进行并行的操作（这里表现为同时进行长报文和短报文的判断和处理）。

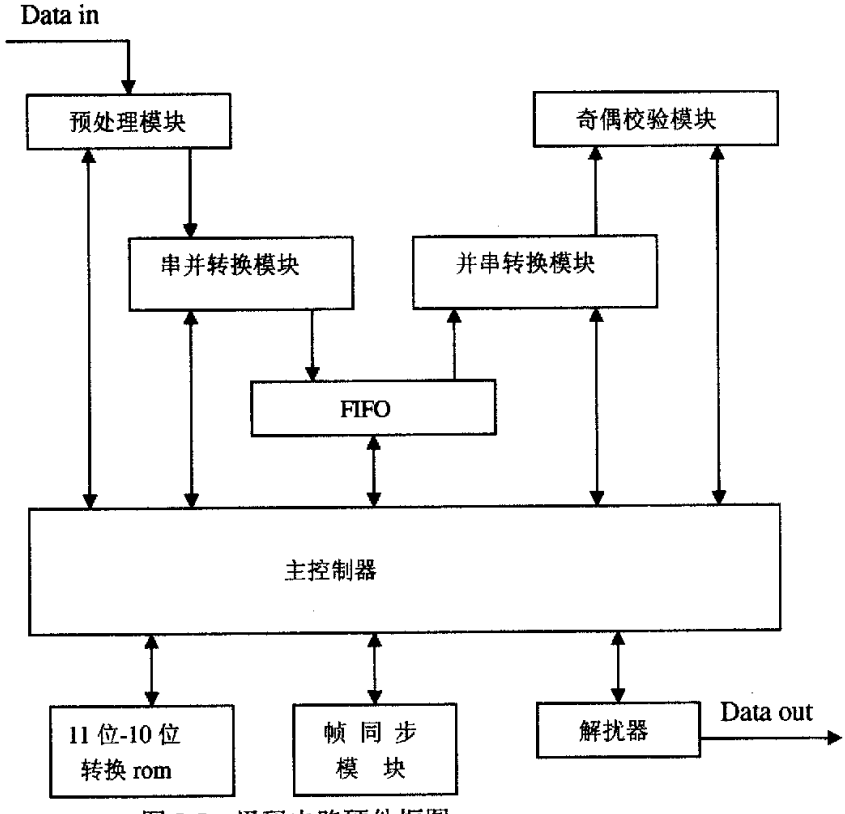

图 3.5 译码电路硬件框图

数据经天线接收、滤波和解调以后，数据以串行的数字信号形式进入译码电路。

对于长报文和短报文，其译码电路的结构是完全一样的，仅仅是在某些参数上存在细微的差异（比如寄存器的长度，求余运算式等）。译码电路的逻辑框图 3.5 所示。

3.3.1 预处理模块

数据首先要在预处理模块中进行一系列的处理，然后才能送去译码。预处理主要包括：

1） 除噪声，如果识别出是数据到来，则给出握手信号

2） 同步信号，防止亚稳态传播

3） 提取位同步信号，保证数据正确采样

其中，位同步的原理框图如图 3.6 所示。图中，边沿跳变检测脉冲作为相位比较器的触发脉冲，当位基准脉冲出现时，相位比较器就可以从可变模分频器中读出分频器计数值，并判别是同步、滞后或是超前状态，再根据判断结果来控制可变模分频器修改分频比。可变模分频器在完成一个分频周期输出位同步脉冲时，重新加载分频器模值存储器中的模值。

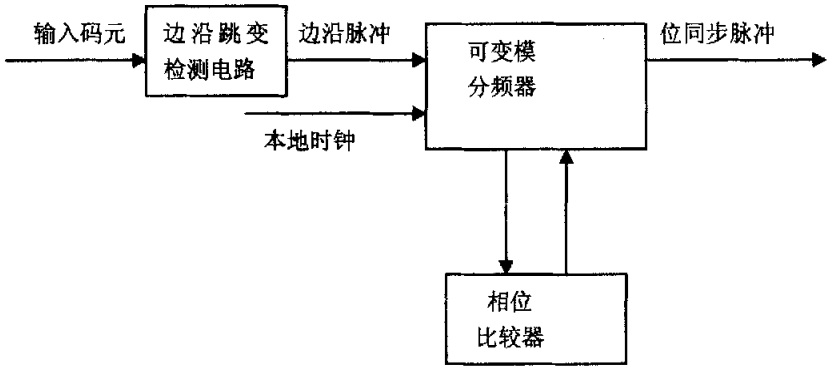

图 3.6 位同步原理图

3.3.2 FIFO

当主控制器收到来自预处理模块的握手信号以后，则认为有效数据已经到达，开始进行译码。由于接收数据的速率（564.48Kbps）大于译码的速率（每接收到 1 位数据需要 120 个左右的时钟周期进行处理，以 50M 时钟计算，处理能

力只能达到400k左右），因此采用了一个fifo，当数据到来时，先放到fifo中作缓冲，主控制器每处理完1位数据，就从fifo中读取新的数据。

主控制器控制fifo的读写，如果读和写的地址不同，那么读和写的操作可以同时进行。

fifo的宽度为11bit，深度为1024，可以储存11个长报文或33个短报文。需要指出的是，如果直到fifo溢出，仍没有一个连续的比特流能通过11步译码步骤的前5步，那么此次译码失败。

3.3.3 奇偶校验模块

奇偶校验模块是为了实现3.1章中11个解码步骤的第2步，是要对除数为g(x)的多项式进行求余运算。g(x)的定义见2.4章。整个电路是一个移位反馈寄存器。一旦余数为0，模块就通知主控制器做下一步运算。同时，电路必须自动清除错误的信息位对余项带来的影响，由此，实现框图如图3.7所示。

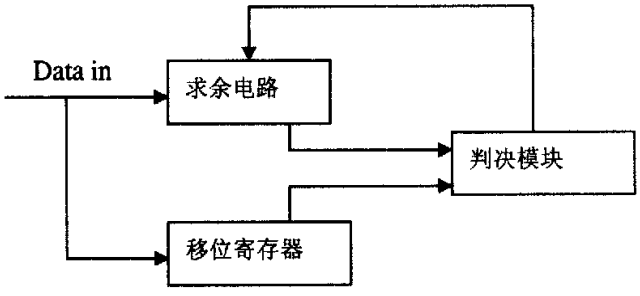

图3.7 改进的求余电路

3.3.4 帧同步模块

这个模块由2部分组成，一个是计算$R_{f(x)}[v(x)]$的电路，结构和计算校验位的电路一样；另一块实际上是块rom，其10位地址线就是$R_{f(x)}[v(x)]$的10位的结果，输出的数据即为9位的移位结果（长报文最多可能有1022位移位，短报文最多有340位移位）。采用rom的形式来实现同步是由原因的，因为如果使用硬件电路来计算$R_{f(x)}[v(x)] = R_{f(x)}[x^s o(x)]$中的s，将耗费很多个额外时钟周期。在这个过程中，其他模块无法工作，因为需要等待同步的结果。这个瓶颈使得整个系统无法采用流水线的方式作业，严重影响译码器的处理速度。采用rom虽然耗费了比较多的资源，但提高了整个系统的工作频率。

Rom中的内容是可以使用程序计算完成的。对于长格式报文，每循环左移

1 位后，都有一个对应的 $R_{f(x)}[v(x)]$值。由于长格式报文为 1023 位，因此共有 1023 个不同的值。即除了 0000000000 外的所有 10 位二进制数的组合。附录 C 中给出了长格式报文的 1023 个不同的 $R_{f(x)}[v(x)]$值。使用 linux 系统中的 sort/uniq 命令对文件排序，结果仍为 1023 行，表明没有重复的值。

3.3.5 11 位-10 位的转换 rom

用来把每个 11 位的字段转换成 10 位的字段。对于长格式报文，需要 83 个时钟周期完成转换；短格式报文需要 21 个。映射关系如 2.3 章所述。

3.3.6 解扰器

解扰模块要顺序执行 3 步操作。当解扰器收到来自主控制器的“开始解扰”命令后，首先根据主控制器提供的 12 位 sb 和 2.2 式计算出扰码器的 32 位初始状态；之后读入数据，根据图 3.3 的算法，进行 m-1 次移位反馈的操作，完成解扰计算；同时，用一个 10 位的寄存器存放累加和，每解扰出 10 位，就把它和累加器中的内容相加，最后累加器中的内容就是用户数据的前 10 位。用户数据最后是从解扰器输出的。

解扰器工作的时序是一个必须注意的问题。码流从 11-10 位转换 rom 中输出后，位速率降低了。之前的校验位检验、同步等模块，每个时钟周期都能处理 1 位数据，但是对于解扰器，由于数据位速率只有原来的 10/11，所以每 11 个时钟周期必须扣除一个脉冲。因此，主控制器中专门有一个 DescrambleRdy 信号作为解扰器的片选信号，以使解扰器的时序节拍可以和其他模块相谐调。

3.3.7 主控制器

主要是一个有限状态机，图 3.8 给出了状态转换图。可以看出，状态机共有 6 个状态：LoadToWindow ，LoadFifoData，RdSynOver，StoreCbSb，LoadS 和 Descramble。状态机通过读取各个模块的反馈信号实现状态间的转换，同时，每个状态输出的控制信号，使得各个模块的能够按照已定的时序工作。这样，整个系统以一种严格、精确的时序关系运作。

LoadToWindow 是初始状态，译码过程中任何一个状态出现错误，都必须回到 LoadToWindow 重新开始。在这个状态中，每次都有 1 位数据被读取到第 1 步译码步骤中的长度为 n+r 的寄存器 window 中。如果 window 中的寄存器不满足奇偶校验条件，那么就要从缓存中继续读取数据；如果缓存为空，那么就要跳

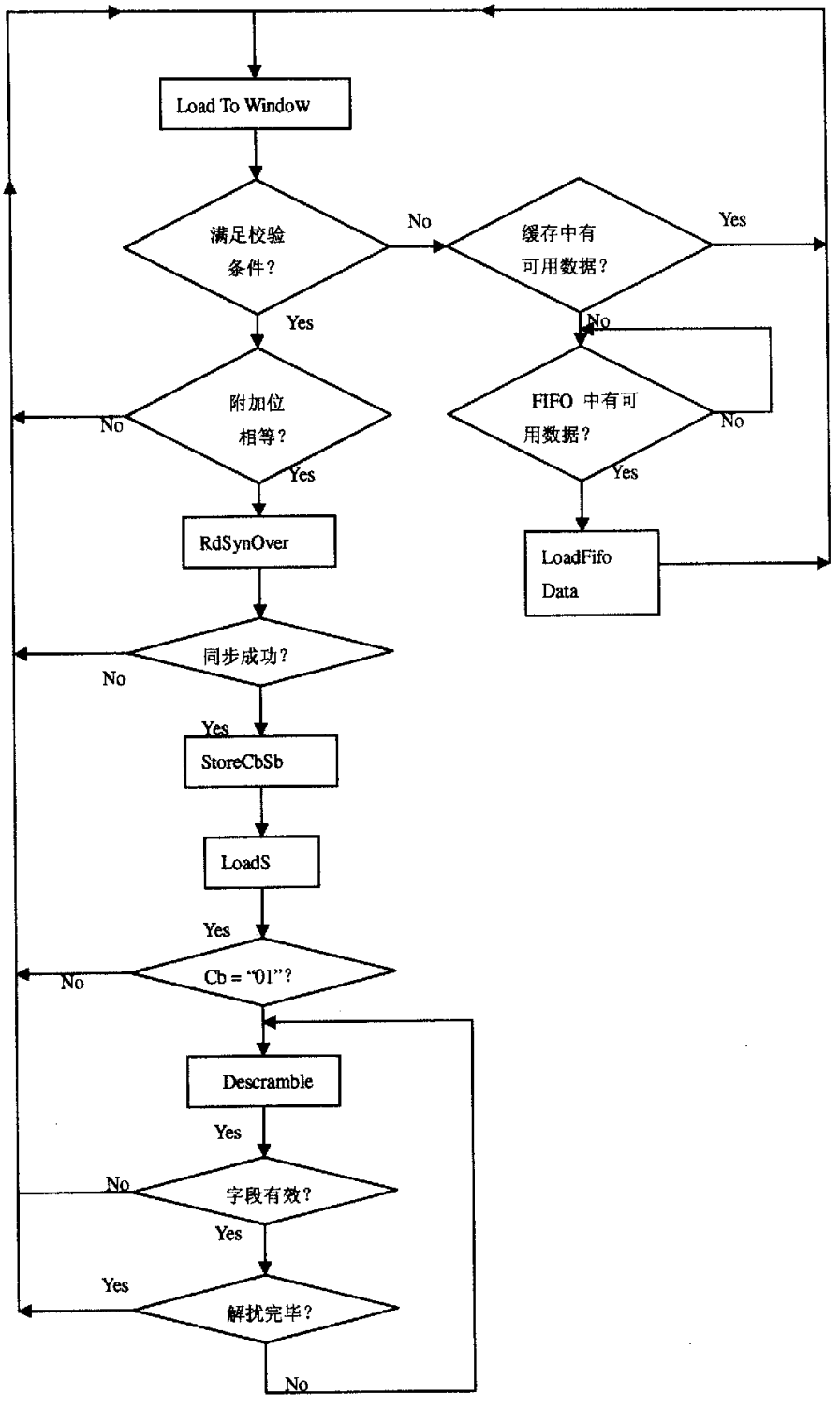

图 3.8　主控制器的状态转换图

转到 LoadFifoData 状态，从 FIFO 中读取数据，每次读取 11 位。

如果奇偶校验和附加位条件均满足，那么就需要等待同步值 S 计算结束。实际上，同步值的计算和奇偶校验是同时进行的，只是前者需多耗费几个时钟周期。

如果同步值 S 有效（如果是长报文，则非 0 有效），那么将进入到 StoreCbSb 状态。在这个状态，控制位 cb 被读取，同时扰码位 sb 被送到解扰器以计算解扰器的初始值。

如果报文格式标识为 01，则转至 Descramble 状态。此时，11 位-10 位转换 rom 开始从 window 的第 S 位开始读取数据。每当成功转换出一个 10 位字段，就把它串行送入解扰器（此时解扰器已经完成初始值计算，处于读取解扰数据的准备状态）。任何一个 11 位字段不满足转换表条件，状态机都会被复位，重新从缓存中读取数据，重新译码。

3.4 硬件设计中的优化问题和处理技巧

3.4.1 大容量 rom 的实现

在译码电路中，需要用到大量的 rom，即电路中的 11 位-10 位转换 rom 和同步器 rom。通常情况下，在 VHDL 中，rom 是可以用如下语句实现的：

```
data <= "xxxxx" when address = "xxxxx" else
        "xxxxx" when address = "xxxxx" else
          ⋮
        "xxxxx" when address = "xxxx" else
        "xxxxx";
```

但是，对于大容量的 rom，这样做会带来很大的问题。综合器（无论是 synplify pro 还是 xst）会把这样的电路完全使用 LUT（Look Up Table）资源实现，当 rom 容量过大时，LUT 资源的开销是惊人的。例如，如果用这种方法实现 11 位-10 位转换 rom，需要使用 670 个 LUT，占所选芯片 xc2s200 全部 LUT 资源的 14%，而同步器 rom 更是高达 25%。在 FPGA 中，每一个 slice 中寄存器和 LUT 的比例是固定的，如果仅仅大量使用 LUT，那么寄存器资源的“不均衡浪费”。而且如

果 LUT 资源耗尽，那么综合将无法完成，必须选用更大容量的芯片，导致成本升高、制板困难。

在 Xilinx 公司的 Spartan2 系列芯片中，都带有 BLOCK selectRam，即块状 ram，所有的存储器，如 ram、fifo 等，都可以用它实现，rom 也不例外。对于 Xilinx 的片子，如果 rom 的地址在 9 到 13 位之间，synplify pro 就可以把下面的 VHDL 语句描述的 rom 综合成块状 ram[9]：

```
subtype ROM_WORD is STD_LOGIC_VECTOR (DATA_WIDTH-1 downto 0);
type ROM_TABLE is array (0 to ROM_DEPTH-1) of ROM_WORD;
constant ROM : ROM_TABLE := ROM_TABLE'(
    ROM_WORD'("xxxxxxxxx"),
    ROM_WORD'("xxxxxxxxx"),
    ROM_WORD'("xxxxxxxxx"),
        ⋮
    ROM_WORD'("xxxxxxxxx")
);
process (CLK)
  begin
    if clk'event and clk = '1' then
      data <= ROM(conv_integer(address); -- Read from the ROM
    end if;
end process;
```

这样， 11 位-10 位转换 rom、同步器 rom 和 fifo 占用块状 ram 的情况分别为 20%、42%和 21%，共使用了 83%的 ram。而整个系统对 LUT 资源的使用为 72%。由此可以看出，芯片各种资源的分配非常优化，利用率很高。

3.4.2 多输入 mux 的实现

当得到了需要移位的偏移量 s 后，就可以找到报文的帧头。所用的办法是一个 n 选 1 的数据选择器（长报文 n 为 1023，短报文 n 为 341），待同步的 n 位数据作数据输入，s 做数据选择。这样做的原因仍然是希望整个设计能以一种流

水线的形式工作。如果采用移位寄存器的办法，就需要额外等待 s 个时钟周期，而且这种等待时间无法确定，因为 s 是变量。

现在的问题是，一个直接实现的 1023 位选 1 的数据选择器，是相当耗资源的，在最初的方案中，这个 1023 选 1 的 mux 占用了 28%的 LUT 资源，显然过于庞大了。FPGA 大多数是基于 4 输入 LUT 的，如果一个函数的输出取决于大于 4 个的输入变量时，就需要多个 LUT 组合、级联完成。1023 个数据输入变量非常巨大，而综合器在选择和组合 LUT 是无法做到最优化，如果把 1023 个输入数据按一定规则重新排列，使用多个更小的 32 选 1 的 mux，则可以显著地减少 LUT 资源的使用量。具体方法是，将数据依次按行排列，如表 3.1 中所示，最后一个 0 作补齐用。这样就有了一个 32*32 的矩阵。将矩阵的每一列作为一个 32 选 1 的 mux 的输入，共需 32 个 mux，这 32 个 mux 的选择输入为 s 的高 5 位；他们共有 32 个输出，再连接到一个 32 选 1 的 mux 的输入，这个 mux 的选择输入为 s 的低 5 位。这样，就用 33 个 32 选 1 的 mux 构造了一个 1024 选 1 的 mux。经 synplify pro 综合，所用 LUT 资源为 13%。

表 3.1　重新排列的 mux 输入

b_{1023}	b_{1022}	b_{1021}	……	b_{991}	b_{992}
b_{991}	b_{990}	b_{889}	……	b_{959}	b_{960}
……	……	……	……	……	……
b_{31}	b_{30}	b_{29}	……	b_1	0

3.4.3 有限状态机的实现

译码电路中的主控制器的主体，就是一个有限状态机。FPGA中的状态机设计，有一定的技巧。状态机一般包含三个模块，一个输出模块，一个状态转移模块和一个状态保存模块。组成三个模块所采用的逻辑也各不相同。输出模块通常既包含组合逻辑又包含时序逻辑；状态转移模块通常由组合逻辑构成；状态保存模块通常由时序逻辑构成。三个模块的关系如下图3.9所示。

实验结果表明，使用三个模块设计的状态机，在综合时比只使用一个模块（进程）的状态机少用近 50%的查表资源。

图3.9 状态机的组成

3.5 仿真结果

对于电路的仿真，可以利用自己编写的 testbench 文件产生仿真激励和检验结果，在 modelsim 软件中实现。

FPGA 的仿真分为功能仿真、综合后仿真和布局布线后仿真 3 种。功能仿真是指对 HDL 代码进行纯软件的仿真，即便是代码中存在不可综合成电路的语句，仿真也可以进行。功能仿真主要是检验代码的正确性。综合后仿真是在将代码综合成实际电路后进行的仿真。布局布线后的仿真是指考虑了电路在所选芯片中经过布局布线后的情况的仿真，仿真时需要反向标注由布线器产生的 sdf 延时文件。由于考虑了实际芯片内的各种连线资源的延时和使用情况，因此布局布线后仿真可以达到对实际电路完全模拟的效果。

在 testbench 文件中加入如下语句：

```
if output_data = USER_CODE then
    assert(false)
    report"Successfully Decoded "
    severity WARNING;
end if;
```

其中，output_data 用以储存解扰器输出的数据，USER_CODE 是常量，表示正确的用户数据。如果相等，则运行 assert 语句，测试在此中止，给出报告“Successfully Decoded”。

图 3.10 是译码成功后译码器主要信号的波形图。由图可见，当主控制器处

于 Descramble 状态时，表示解扰器正在解扰。这时，rvf（即 $R_{f(x)}[v(x)]$的值）为 0100011100，对应的 fail_s（即同步运算中移位的数量 S）的值为 00000110000，表明码矢是从第 975（1023-48）位开始被接受的。FIFO 的读写地址（rd_addr 和 wr_addr）都为 0001101000。cbsb 的值为 001100110001011，即控制位为 001，扰码位为 100110001011，这和编码时的预定值是一致的；因此 unknown 为 0，说明这不是一个“未知格式的报文”。over7500bits 为 0，表示接收到的码矢并为达到 7500 位，就被成功译码。

当 descramble_over 从 0 跳变到 1 时，表明解扰结束，至此，译码工作全部完成，主控制器从 Descramble 状态跳到 LoadToWindow 状态。window，rd_addr，wr_addr，rvf，fail_s，cbsb 等寄存器全部复位。用户数据开始从 dataout 管脚输出。done 信号在这里一直为 0，当 830 位用户数据全部输出完毕，done 信号被置 1。

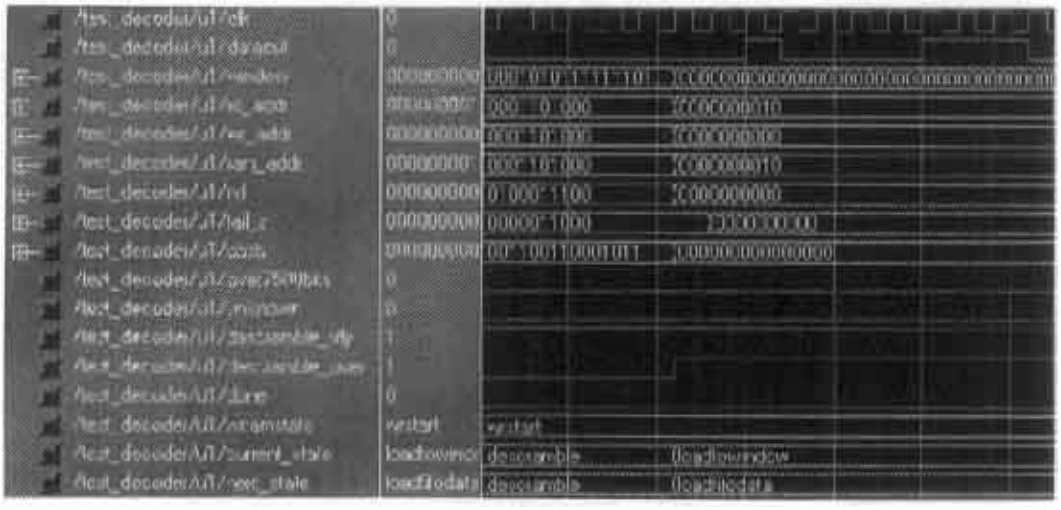

图 3.10　　译码器译出用户数据

第四章　欧洲标准应答器地面部分实现方案

4.1　应答器所用 FSK 信号的参数

在表 1.1 中我们介绍了欧洲标准应答器的主要参数，知道应答器信号是以 FSK 形式进行调制的。这里，我们给出应答器 FSK 信号的详细指标：

能量传输频率（f_P）：　27.095Mhz

逻辑 1（f_H）　：　4.516Mhz

逻辑 0（f_L）　：　3.951Mhz

中心频率（f_C）　：　$(f_H+f_L)/2 = 4.234Mhz \pm 200kHz$

数据速率（DR）　：　564.48Kbps

频偏（f_D）　：　$(f_H-f_L)/2 = 282.24khz \pm 5\%$

调制指数　：　1

注意到这几个参数之间的关系：

$f_H = 4.516Mhz = 27.095Mhz / 6 = f_P / 6$

$DR = 564.48khz = 4.516Mhz / 8 = f_H / 8$

$f_L = 3.951Mhz = 4.516Mhz - 564.48khz = f_H - DR$

4.2　FSK 调制方案的实现

4.2.1 调制方案的比较

实现如 4.1 章节中参数的 FSK 调制信号的关键在于几个不同频率信号的产生，即 f_H 、f_L 和 DR。考虑到它们之间存在一定的关系，故提出以下几种方案。

1、模拟混频

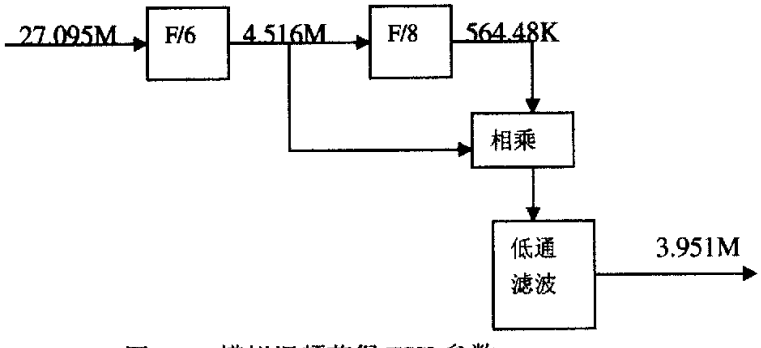

图 4.1　模拟混频获得 FSK 参数

能量传输频率是 27.095MHz，经过限流和整形电路后，得到可以利用的时钟信号。可以按图 4.1 中所示的方法得到所需频率。

这种方法非常直接，但是缺点也很明显。首先是模拟低通滤波器的设计，其系统函数要求下降很快，实现比较困难，因此几个频率的精度就很难保证；另外，乘法器和低通滤波器的使用，使得系统的响应时间比较长，功耗较大。此外，信号经低通滤波器输出后还需要一定的放大。

2、 利用本地时钟震荡器

定制一个特殊频率的晶振时钟是最简便的方法，如果本地晶振的频率为 31.611M，那么就可以使用图 4.2 中所示的方法方便地得到各种频率。

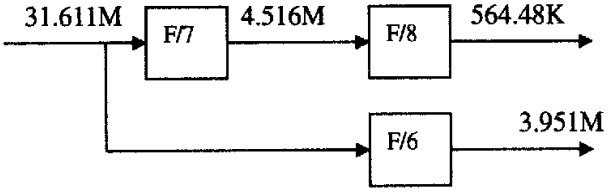

图 4.2 使用本地时钟获得 FSK 参数

这种办法产生的 FSK 信号，频率稳定度高，实现电路简单，功耗低，响应时间也很快。唯一的缺点是，由于本地时钟独立于车载设备，因此无法根据每台机车的 BTM 天线信号作相应的调整。如果本地时钟不准，或是因为温度、干扰等原因产生时钟的变化，会严重影响 FSK 信号的质量。

3、 全数字调制方案

因此，提出全数字的调制方案：仍利用来自车载设备的 27.095M 时钟，利用高性能 FPGA 中的数字锁相环产生 7 倍频信号，再经过分频操作，得到所需的 FSK 参数。这种方案具有很高的实用性，将在下文中详述。

4.2.2 关于 DCM

DCM（Digital Clock Manager）是 Xinlix 公司的 VirtexII、Spartan3 等系列的 FPGA 器件独有的模块，具有强大的时钟管理功能[8]。图 4.3 给出了 DCM 模块的示意图，其具体特性如下[7]：

1、除时钟信号延时，DCM 产生新的系统时钟，并和输入时钟是相位匹配的，因此消除了时钟分布延时；

2、频率合成，可以在很宽的范围内对输入时钟进行分频和倍频；

3、移相，通过动态相移控制提供粗略的或精细的移相。

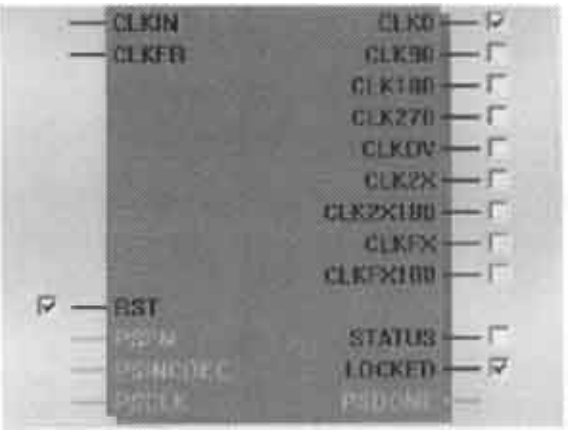

图 4.3 DCM 管脚示意

DCM 利用全数字延时线来实现对时钟相位和频率的高精度控制。同时，DCM 利用全数字反馈系统对电压和温度变化作动态补偿。在这个设计中，我们最关心的是 DCM 的频率合成功能。DCM 提供多种和输入时钟相关的频率输出，如图 4.3 所示，CLKIN 为外部时钟输入管脚，CLK90、CLK180 和 CLK270 分别为 CLKIN 移相 90 度、180 度和 270 度的输出。CLKDV 可以对 CLKIN 进行 1.5、2、2.5、3、3.5、4、4.5、5、5.5、6、6.5、7、7.5、8、9、10、11、12、13、14、15 和 16 分频。CLK2X 为 2 倍频输出。CLKFX 可以输出这样的频率：$FREQ_{CLKFX} = (M/D)* FREQ_{CLKIN}$，其中 M 和 D 是 2 个整数，一般 $24M < FREQ_{CLKFX} < 260M$，具体数值视芯片型号而定。LOCKED 为频率锁定标志：0 表示频率尚未锁定，1 表示频率已经锁定。利用 DCM 的频率合成特性，可以方便地得到所需的 FSK 信号的各种参数。

4.2.3 FSK 数字调制的实现

这里，我们采用 Xinlix 公司的 Spartan3 系列 XC3S50 芯片来实现 FSK 信号的数字调制。利用其中的 DCM 模块，可以方便地实现 7 倍频。这样，就可以通过简单的分频模块得到 FSK 信号的 3 个关键频率：f1，f0 和 DR。硬件框图如图 4.4 所示。

实现过程中，可以定义一个常量信号，用以存放编码，即图 4.X 中的数据存储器。在 564.48K 的数据时钟的触发下，编码数据被循环依次输出，以选择 f1 和 f0 间的切换。

在这个系统中，所有的子时钟（4.516M，3.951M，564.48K）都同步于 DCM 的输出时钟（189.665M），因而可以保证 FSK 信号在 f_H 和 f_L 之间切换时相位的连续。

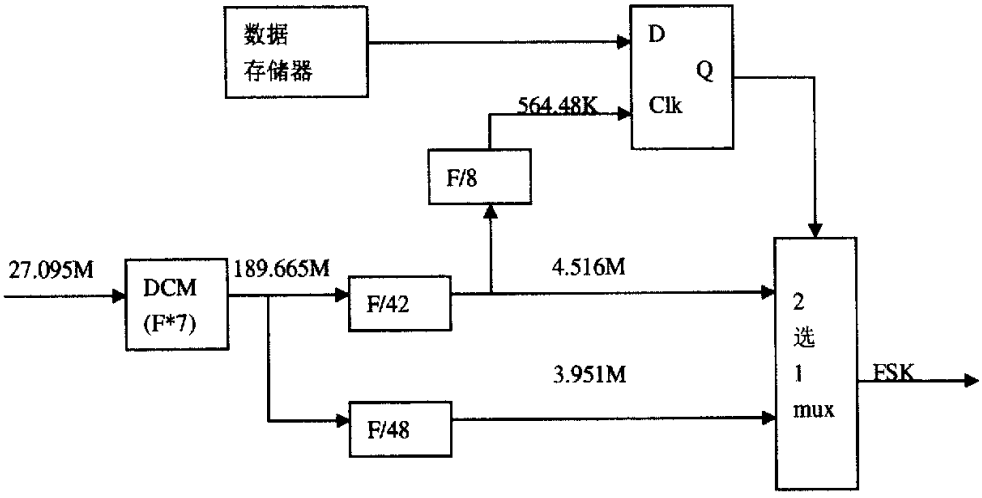

图 4.4　利用 DCM 实现 FSK 信号调制

4.3　仿真结果

这里给出了调制电路的仿真结果，由于是布局布线后的仿真，因此能够真实地反映出电路运行的结果。

在图 4.5 中，clk 为 27.095M 的输入时钟，当 DCM 的 locked 信号为高后（即 DCM 内的数字锁相环路锁定后），7 倍频信号（图中的 test_clk7）稳定建立。与此同时，f_H 、f_L 和 FSK 调制信号（分别为图中的 test_f1，test_f0 和 fsk_out）开始可靠输出。由于列车高速通过地面点的时间只有 10ms 的数量级，所以 FSK 信号建立的时间越短越好，而从第一个 clk 信号输入到 7 倍频信号建立，仅仅不到 500ns 的时间，非常符合要求。

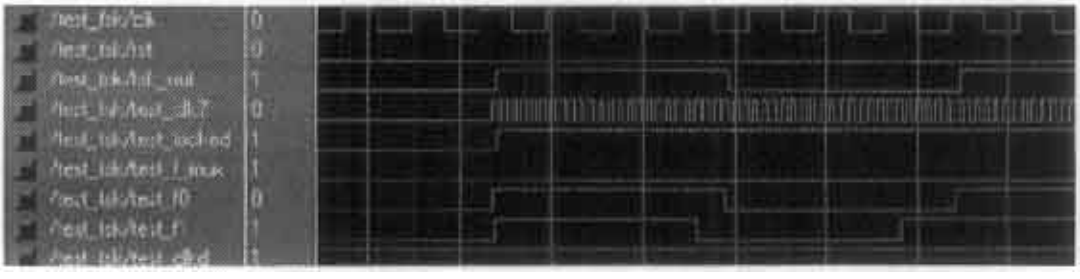

图 4.5　数字锁相环锁定

图 4.6 是将图 4.5 缩小后观察的结果，图中，test_f_mux 输出需要调制的数据，从图中可以清楚地观察到 fsk_out 随 test_f_mux 高低电平的不同而产生的疏密的变化。如果在数据 1 和 0 间的跳变点放大观察，可以看到 FSK 信号在 2 个频率间切换时，相位是连续的。

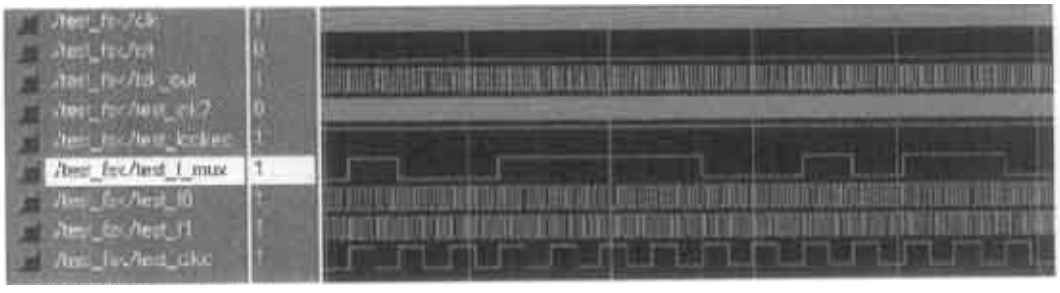

图 4.6　FSK 信号的输出

结束语

欧洲标准应答器的在国内的推广使用是近两年来铁路信号领域内被关注的一个热点。这套技术在欧洲几家大学和公司的多年研究下已经非常成熟和完善，欧洲多家铁路信号公司都已生产出可靠的产品，在欧洲广泛使用，并互相兼容。国内已有数家企业和研究机构开始大力着手研究开发欧洲标准应答器的整套设备，有的已见初步成果。

本文以 Form Fit Function Specification Coding Strategy 为基础，对欧洲标准应答器的编码和译码理论展开研究，对其中的奇偶校验、同步和解扰等关键算法给予了严格的证明，提出了实现模型，并且使用 Xilinx 公司的 FPGA 芯片实现了其中的译码部分。同时，也对欧洲标准应答器的地面部分也作了一定探索，提出了全数字的 FSK 信号调制方案，同样使用 Xilinx 的 FPGA 芯片实现。

本文是我对研究生学习阶段的总结，希望其中的成果能对他人的研究提供一定的参考价值。但由于时间仓促，个人研究水平有限，在以下方面还需要更进一步的研究和探索。

1、对于编码结构更深入的理解，对其在防护各种突发错误、随机错误、丢位和插位错误方面的严格数学证明；

2、编制软件生成完善的编码；

3、对译码器更全面的测试；

4、欧洲标准应答器车载和地面天线的结构和实现办法，以及能够对天线进行操作和控制的相关电路。

参考文献

[1] Klaus Finkenzeller. 射频识别（RFID）技术[M]. 北京：电子工业出版社，2001.

[2] UNISIG Consortium. FFFIS for Eurobalise[S]. 2003.

[3] UNISIG Consortium. Test Specification for Eurobalise FFFIS[S]. 2003.

[4] 张鸣瑞 邹世开. 编码理论[M]. 北京：北京航空航天大学出版社，1990.

[5] 林敏 方颖立. VHDL 数字系统设计与高层次综合[M]. 北京：电子工业出版社，2002.

[6] 段吉海 黄智伟. 基于 CPLD/FPGA 的数字通信系统建模与设计[M]. 北京：电子工业出版社，2004.

[7] 王诚 薛小刚 钟信潮. FPGA/CPLD 设计工具 Xilinx ISE 5.X 使用详解[M]. 北京：人民邮电出版社，2003.

[8] XILINX. Spartan3 Platform FPGAs: Complete Data Sheet [S]. 2004.

[9] XILINX. Software Manuals[S]. 2004.

致谢

感谢我的导师杨志杰研究员在课题研究和论文撰写的过程中对我在理论上悉心的指导和物力上的大力帮助，正是在杨老师的全力帮助下，我才顺利地完成了毕业论文，完满结束了硕士研究生学业。

感谢范明主任在实验条件和专业资料上的慷慨支援，使我具有一个良好的研究环境，能够充分发挥专业上的才智。

感谢我身边所有帮助过我的同学、同事和朋友们！感谢他们在资料翻译、硬件调试等诸多方面给予我的提示、建议和无私的帮助！

附录 A：以 8 进制显示的 10-11 字段转换表

00101, 00102, 00103, 00104, 00105, 00106, 00107, 00110, 00111, 00112,
00113, 00114, 00115, 00116, 00117, 00120, 00121, 00122, 00123, 00124,
00125, 00126, 00127, 00130, 00131, 00132, 00133, 00134, 00135, 00141,
00142, 00143, 00144, 00145, 00146, 00147, 00150, 00151, 00152, 00153,
00154, 00155, 00156, 00157, 00160, 00161, 00162, 00163, 00164, 00165,
00166, 00167, 00170, 00171, 00172, 00173, 00174, 00175, 00176, 00201,
00206, 00211, 00214, 00216, 00217, 00220, 00222, 00223, 00224, 00225,
00226, 00231, 00233, 00244, 00245, 00246, 00253, 00257, 00260, 00261,
00272, 00273, 00274, 00275, 00276, 00301, 00303, 00315, 00317, 00320,
00321, 00332, 00334, 00341, 00342, 00343, 00344, 00346, 00352, 00353,
00357, 00360, 00374, 00376, 00401, 00403, 00404, 00405, 00406, 00407,
00410, 00411, 00412, 00413, 00416, 00417, 00420, 00424, 00425, 00426,
00427, 00432, 00433, 00442, 00443, 00445, 00456, 00457, 00460, 00461,
00464, 00465, 00470, 00471, 00472, 00474, 00475, 00476, 00501, 00502,
00503, 00504, 00505, 00506, 00507, 00516, 00517, 00520, 00521, 00522,
00523, 00524, 00525, 00530, 00531, 00532, 00533, 00534, 00535, 00544,
00545, 00546, 00547, 00550, 00551, 00552, 00553, 00554, 00555, 00556,
00557, 00560, 00561, 00562, 00563, 00571, 00573, 00576, 00601, 00602,
00604, 00605, 00610, 00611, 00612, 00613, 00614, 00615, 00616, 00617,
00620, 00621, 00622, 00623, 00624, 00625, 00626, 00627, 00630, 00634,
00635, 00644, 00645, 00646, 00647, 00650, 00651, 00652, 00653, 00654,
00655, 00656, 00657, 00660, 00661, 00662, 00663, 00666, 00667, 00672,
00674, 00675, 00676, 00701, 00712, 00713, 00716, 00717, 00720, 00721,
00722, 00723, 00730, 00731, 00732, 00733, 00734, 00735, 00742, 00743,
00744, 00745, 00746, 00747, 00750, 00751, 00752, 00753, 00754, 00755,
00756, 00757, 00760, 00761, 00764, 00765, 00766, 00767, 00772, 00773,
00776, 01001, 01004, 01005, 01016, 01017, 01020, 01021, 01022, 01023,
01024, 01025, 01030, 01031, 01032, 01033, 01034, 01035, 01043, 01044,

01045, 01046, 01047, 01054, 01057, 01060, 01061, 01062, 01075, 01076,
01101, 01102, 01103, 01110, 01114, 01115, 01116, 01117, 01120, 01121,
01122, 01123, 01124, 01125, 01126, 01127, 01130, 01131, 01132, 01133,
01142, 01143, 01144, 01145, 01146, 01147, 01151, 01152, 01153, 01154,
01155, 01156, 01157, 01160, 01164, 01166, 01167, 01176, 01201, 01214,
01217, 01220, 01221, 01222, 01223, 01224, 01225, 01226, 01227, 01230,
01231, 01232, 01233, 01243, 01244, 01245, 01253, 01254, 01255, 01256,
01257, 01260, 01261, 01272, 01273, 01274, 01275, 01276, 01301, 01302,
01303, 01305, 01306, 01307, 01317, 01320, 01321, 01332, 01334, 01335,
01342, 01343, 01344, 01345, 01350, 01351, 01352, 01353, 01355, 01356,
01357, 01360, 01361, 01364, 01365, 01370, 01371, 01372, 01373, 01374,
01376, 01401, 01403, 01406, 01407, 01414, 01415, 01416, 01417, 01420,
01424, 01425, 01431, 01433, 01434, 01435, 01443, 01445, 01456, 01457,
01460, 01462, 01474, 01475, 01476, 01501, 01502, 01503, 01504, 01505,
01516, 01517, 01520, 01524, 01532, 01533, 01544, 01546, 01550, 01551,
01552, 01553, 01554, 01557, 01560, 01561, 01562, 01563, 01566, 01567,
01576, 01601, 01603, 01604, 01605, 01606, 01607, 01610, 01611, 01612,
01613, 01614, 01615, 01616, 01617, 01620, 01621, 01622, 01623, 01624,
01625, 01626, 01630, 01631, 01632, 01633, 01635, 01643, 01644, 01645,
01650, 01651, 01652, 01653, 01654, 01655, 01656, 01657, 01660, 01661,
01672, 01674, 01675, 01676, 01701, 01720, 01744, 01745, 01746, 01747,
01750, 01751, 01752, 01753, 01754, 01755, 01756, 01757, 01760, 01761,
01762, 01763, 01764, 01765, 01766, 01767, 01770, 01771, 01772, 01773,
01774, 01775, 02002, 02003, 02004, 02005, 02006, 02007, 02010, 02011,
02012, 02013, 02014, 02015, 02016, 02017, 02020, 02021, 02022, 02023,
02024, 02025, 02026, 02027, 02030, 02031, 02032, 02033, 02057, 02076,
02101, 02102, 02103, 02105, 02116, 02117, 02120, 02121, 02122, 02123,
02124, 02125, 02126, 02127, 02132, 02133, 02134, 02142, 02144, 02145,
02146, 02147, 02151, 02152, 02153, 02154, 02155, 02156, 02157, 02160,
02161, 02162, 02163, 02164, 02165, 02166, 02167, 02170, 02171, 02172,

02173, 02174, 02176, 02201, 02210, 02211, 02214, 02215, 02216, 02217,
02220, 02223, 02224, 02225, 02226, 02227, 02231, 02233, 02244, 02245,
02253, 02257, 02260, 02261, 02272, 02273, 02274, 02275, 02276, 02301,
02302, 02303, 02315, 02317, 02320, 02321, 02332, 02334, 02342, 02343,
02344, 02346, 02352, 02353, 02357, 02360, 02361, 02362, 02363, 02370,
02371, 02374, 02376, 02401, 02403, 02404, 02405, 02406, 02407, 02412,
02413, 02416, 02417, 02420, 02421, 02422, 02424, 02425, 02426, 02427,
02432, 02433, 02434, 02435, 02442, 02443, 02445, 02456, 02457, 02460,
02470, 02471, 02472, 02474, 02475, 02476, 02501, 02502, 02503, 02504,
02505, 02516, 02517, 02520, 02521, 02522, 02523, 02524, 02532, 02533,
02534, 02544, 02545, 02546, 02547, 02550, 02551, 02552, 02553, 02554,
02555, 02556, 02557, 02560, 02563, 02576, 02601, 02610, 02611, 02613,
02617, 02620, 02621, 02622, 02623, 02624, 02625, 02626, 02630, 02631,
02632, 02633, 02634, 02635, 02644, 02645, 02646, 02647, 02650, 02651,
02652, 02653, 02654, 02655, 02656, 02657, 02660, 02661, 02662, 02663,
02667, 02674, 02675, 02676, 02701, 02702, 02715, 02716, 02717, 02720,
02723, 02730, 02731, 02732, 02733, 02734, 02742, 02743, 02744, 02745,
02746, 02747, 02752, 02753, 02754, 02755, 02756, 02757, 02760, 02761,
02772, 02773, 02776, 03001, 03004, 03005, 03010, 03011, 03012, 03013,
03016, 03017, 03020, 03021, 03022, 03023, 03024, 03025, 03026, 03027,
03030, 03031, 03032, 03033, 03034, 03035, 03042, 03043, 03044, 03045,
03046, 03047, 03054, 03055, 03056, 03057, 03060, 03061, 03064, 03065,
03076, 03101, 03102, 03103, 03105, 03110, 03111, 03114, 03115, 03116,
03117, 03120, 03121, 03122, 03123, 03124, 03125, 03126, 03127, 03130,
03131, 03132, 03133, 03142, 03143, 03147, 03150, 03151, 03152, 03153,
03154, 03155, 03156, 03157, 03160, 03161, 03162, 03163, 03164, 03165,
03166, 03167, 03172, 03173, 03175, 03176, 03201, 03204, 03206, 03214,
03215, 03216, 03217, 03220, 03221, 03222, 03223, 03224, 03225, 03226,
03227, 03230, 03231, 03232, 03233, 03242, 03243, 03244, 03245, 03246,
03247, 03252, 03253, 03254, 03255, 03256, 03257, 03260, 03261, 03270,

03271, 03272, 03273, 03274, 03275, 03276, 03301, 03302, 03303, 03305, 03306, 03307, 03312, 03313, 03316, 03317, 03320, 03321, 03332, 03334, 03335, 03344, 03345, 03350, 03351, 03352, 03353, 03357, 03360, 03361, 03364, 03365, 03366, 03367, 03370, 03371, 03372, 03373, 03374, 03376, 03401, 03403, 03417, 03420, 03424, 03425, 03431, 03433, 03434, 03435, 03436, 03443, 03445, 03456, 03457, 03460, 03462, 03474, 03476, 03501, 03502, 03503, 03504, 03505, 03516, 03517, 03520, 03524, 03531, 03532, 03533, 03544, 03546, 03551, 03552, 03553, 03554, 03555, 03557, 03560, 03561, 03563, 03566, 03571, 03576, 03601, 03602, 03603, 03604, 03605, 03606, 03607, 03610, 03611, 03612, 03613, 03614, 03615, 03616, 03617, 03620, 03621, 03622, 03623, 03624, 03625, 03626, 03627, 03630, 03631, 03632, 03633, 03634, 03635, 03636, 03642, 03643, 03644, 03645, 03646, 03647, 03650, 03651, 03652, 03653, 03654, 03655, 03656, 03657, 03660, 03661, 03662, 03663, 03664, 03665, 03666, 03667, 03670, 03671, 03672, 03673, 03674, 03675, 03676

附录B：可供测试的报文及其对应的编码

报文编号	报文内容（16进制显示）
1	80 10 01 1C D8 00 00 42 91 D3 08 59 4D 17 9D D5 88 1A 3F FF FF FF FF FF FF FF C0
2	80 20 01 1C 78 00 00 81 1C 15 0D 4D 7F FF FF FF FF FF FF FF FF FF FF FF FF FF C0
3	80 10 01 1D 38 00 00 C4 17 8C 99 5E 09 CD 8D DB D4 17 1C 54 DC 92 9C D1 3F FF C0
4	80 20 01 1C A8 00 01 01 CA D8 52 0B 49 0D CA FF FF FF FF FF FF FF FF FF FF FF C0
5	80 00 01 1D 18 00 02 03 9A CF D6 D0 1A 8F DE 95 95 9A 90 CB 1E 1C 7F FF FF FF C0
6	80 10 01 1C B8 00 02 42 19 89 DC CF D8 11 1D 19 3F FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FC
7	80 20 01 1E 04 00 02 87 58 8B D6 48 9C D8 99 51 D4 55 8D 49 5A 18 53 D4 D3 4B D7 5A 9C 1C 0D 8C D4 54 91 CD DA 3F FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FC
8	80 10 01 1E AC 00 02 D1 C9 53 D5 9C D2 1C 4B 14 0C 0B C9 58 9C CD 8D 8D 14 4E 1A 1E 4D D0 DA 89 1D DA 0E 99 59 1E 8F 89 5A 0D 56 4C 91 91 8B 96 11 08 0A 49 55 4E 49 DD D7 1D 95 4A CF 1A D8 95 92 9C 4B 19 DD 54 8B DC 4D 0B 11 53 D5 17 0F 7F FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FC
9	80 20 01 1D B8 00 03 06 08 16 54 0B 8B 98 1C 52 CD 55 DD CC D3 D5 D8 1E 12 11 5B 4C 89 1D 8F D5 3F FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FC
10	80 00 01 1E D4 00 03 54 5B 16 5B D2 8B CE 4C 5D 57 4D C8 51 58 09 0D 4D 09 9A 57 9E 59 4A 5E 1C 1E 9B 17 4E 92 D3 57 55 9C 14 9B 4D 57 CC DD 49 CA CF 17 5D D2 55 8C 54 54 4A 97 10 94 1C CC 4D 5D 09 9D 17 5B 5A 13 9C 54 1D D6 DA 09 D9 49 8F 14 57 D5 9D CA 19 15 D8 D5 FF FF FF FF FF FF FF FF FF FF FF FF FF FC

编码编号	编码内容（16进制显示）
1	75 BB BF 37 1E 7F 76 B9 25 D3 FD ED 46 BA D7 69 85 7B ED 37 F6 A5 F2 F4 3E FD 8C DF 5A 40 DA 7F BB 71 22 C5 BF 22 CD A3 9B 07 78
2	DD 77 C1 57 C7 E7 F9 79 54 67 A8 B9 BE B1 FE 5F A8 37 D3 B7 A8 CD 2C EF F7 58 DC E6 D6 40 7C E7 B3 25 F8 B3 7B EF 6B 7C DA A8 E0
3	0E 50 48 31 9E 85 A1 6B E4 A5 A4 88 A8 06 0A 70 18 A5 73 F0 2B 10 31 08 40 72 30 C0 90 40 72 80 89 22 04 E5 0E 6C 51 57 03 44 80
4	3B 47 71 49 18 C9 02 16 15 84 09 0C 49 31 C0 C9 0A E7 17 0C A0 44 53 12 66 08 05 83 94 40 48 77 B1 80 8A 58 F2 37 02 07 49 71 30
5	E8 AB AD 7C AA CD 75 66 23 49 71 75 3E 09 19 2E DA 8A 4D 5F 6C 4F 5B A5 51 09 56 CE 5E 40 4D 77 AA 52 B5 98 32 6A 47 51 76 ED 28
6	98 3B E6 32 7B 23 75 ED 96 19 46 9A 3D 0E F2 A6 3D 2D 7C 37 88 CD F7 77 C3 DD EB D1 82 65 A7 F5 22 D4 BB D4 75 3A DC 4D 34 2F 5E 63 91 C7 B3 92 96 BA 7D 7B EC DB 14 2F 24 5C 87 F8 EA 7D 3E 0D 2B F6 F2 F1 AB 99 5B 7E DF 45 3C 41 3C AE 77 C2 3B E9 7C 47 5A 7D F2 C5 5D 49 AA F3 30 67 74 FC 4A C7 59 F2 D9 ED 5E F9 13 E2 E6 17 85 92 CD 7B 0F D9 10 1B 51 67 29 2F B5 DF 89 B8 AC DE DA 7C
7	BE 09 4B AD 38 41 5F E7 42 D4 6B A0 79 61 36 1B 64 27 9C DF 2B ED 3D E5 BC D9 E2 F6 DF 78 CC 52 E2 B3 16 BC B9 5D CD 8E 73 D7 5B F6 08 B9 0E F5 A3 E5 03 5B 3D D7 B3 15 E4 0C FB 45 55 D6 8F 75 5F 6D 75 1A 4F 7F 69 7C F6 7B 6D 8A 19 3F 4C 07 B7 A9 6E FC 73 F4 C6 EF D4 D8 70 DB F7 EF 79 90 C9 A1 25 8E 67 6E CA D0 E1 C2 69 CE D9 25 AE 0F F1 21 90 11 CD A9 7D BF A5 60 F6 F5 F4 BD 0B D8
8	70 88 05 88 70 13 48 B4 43 C0 F0 09 0A 45 D4 CC 19 02 54 BB 81 23 A8 91 CC 2A 41 35 03 DA F1 65 AE 5A 01 4B A8 7A C1 1C 56 1C 1D 6A 22 EE 6A 5D 15 37 E8 44 3D 52 62 1D 41 9E 2B 05 9F 0B 99 F2 8D 95 91 82 97 D6 10 1C 45 39 83 CD 5A 1D 46 F1 51 C0 96 8A 51 32 93 C1 74 81 8E A9 6B 12 03 9B 4E C8 4C F4 B8 6B 3F 68 40 F0 12 C3 2E 86 BE 38 B0 FB 90 3F 09 55 15 86 41 20 6C FA E7 29 86 20
9	70 DA 90 A3 DA 44 AF 4B CA E8 D0 F7 08 69 A2 85 7B DC A4 12 C4 88 22 BA F8 29 C1 0C 60 90 F2 1E 23 09 58 21 A0 B8 91 DE 1A E0 21 D2 63 84 51 FD 47 24 A3 84 74 DC A5 ED 37 08 9C 77 12 03 EE 4D B2 97 30 85 40 C4 85 02 E6 E3 48 B5 88 2B 46 31 34 F8 87 09 3F 0C 11 50 CB 28 70 9F 70 11 1E 95 42 10 E4 3A 3C 74 83 51 13 A7 2C 12 D7 69 62 95 08 47 10 10 9F 4F 1F 52 30 D0 3D 0E 4A 10 74 A6
10	A1 E4 F4 35 1A 93 5B 15 44 A2 AB 75 B5 B7 56 B6 80 D5 66 DF 95 EA D3 30 3E 47 CD 15 BB BC 73 85 AC B6 7A 6F 31 DD 37 E8 9B DB 61 EA 24 BD 92 A2 2A 53 A4 69 6A 2A 64 3C CA F6 F6 5D 86 2A 61 7D A7 45 71 38 52 5D 3F C4 83 5A A4 AE 12 AD 15 6E 71 A6 6B 53 96 35 DA 01 F6 6A 93 B4 BC 14 E1 6C DA 7A DA 24 64 D1 0C 6B D2 8D E5 96 EA 18 F4 C4 F8 97 90 12 1C 94 E9 16 F9 8A A3 BC 29 56 63 CC

附录 C：长格式报文每次循环一位后 $R_{r(x)}[v(x)]$ 的值

303 0D9 1B2 364 017 02E 05C 0B8 170 2E0 31F 0E1

1C2 384 1D7 3AE 183 306 0D3 1A6 34C 047 08E 11C

238 2AF 381 1DD 3BA 1AB 356 073 0E6 1CC 398 1EF

3DE 163 2C6 353 079 0F2 1E4 3C8 14F 29E 3E3 119

232 2BB 3A9 18D 31A 0EB 1D6 3AC 187 30E 0C3 186

30C 0C7 18E 31C 0E7 1CE 39C 1E7 3CE 143 286 3D3

179 2F2 33B 0A9 152 2A4 397 1F1 3E2 11B 236 2B3

3B9 1AD 35A 06B 0D6 1AC 358 06F 0DE 1BC 378 02F

05E 0BC 178 2F0 33F 0A1 142 284 3D7 171 2E2 31B

0E9 1D2 3A4 197 32E 083 106 20C 2C7 351 07D 0FA

1F4 3E8 10F 21E 2E3 319 0ED 1DA 3B4 1B7 36E 003

006 00C 018 030 060 0C0 180 300 0DF 1BE 37C 027

04E 09C 138 270 23F 2A1 39D 1E5 3CA 14B 296 3F3

139 272 23B 2A9 38D 1C5 38A 1CB 396 1F3 3E6 113

226 293 3F9 12D 25A 26B 209 2CD 345 055 0AA 154

2A8 38F 1C1 382 1DB 3B6 1B3 366 013 026 04C 098

130 260 21F 2E1 31D 0E5 1CA 394 1F7 3EE 103 206

2D3 379 02D 05A 0B4 168 2D0 37F 021 042 084 108

210 2FF 321 09D 13A 274 237 2B1 3BD 1A5 34A 04B

096 12C 258 26F 201 2DD 365 015 02A 054 0A8 150

2A0 39F 1E1 3C2 15B 2B6 3B3 1B9 372 03B 076 0EC

1D8 3B0 1BF 37E 023 046 08C 118 230 2BF 3A1 19D

33A 0AB 156 2AC 387 1D1 3A2 19B 336 0B3 166 2CC

347 051 0A2 144 288 3CF 141 282 3DB 169 2D2 37B

029 052 0A4 148 290 3FF 121 242 25B 269 20D 2C5

355 075 0EA 1D4 3A8 18F 31E 0E3 1C6 38C 1C7 38E

1C3 386 1D3 3A6 193 326 093 126 24C 247 251 27D

225 295 3F5 135 26A 20B 2C9 34D 045 08A 114 228

28F 3C1 15D 2BA 3AB 189 312 0FB 1F6 3EC 107 20E

2C3 359 06D 0DA 1B4 368 00F 01E 03C 078 0F0 1E0
3C0 15F 2BE 3A3 199 332 0BB 176 2EC 307 0D1 1A2
344 057 0AE 15C 2B8 3AF 181 302 0DB 1B6 36C 007
00E 01C 038 070 0E0 1C0 380 1DF 3BE 1A3 346 053
0A6 14C 298 3EF 101 202 2DB 369 00D 01A 034 068
0D0 1A0 340 05F 0BE 17C 2F8 32F 081 102 204 2D7
371 03D 07A 0F4 1E8 3D0 17F 2FE 323 099 132 264
217 2F1 33D 0A5 14A 294 3F7 131 262 21B 2E9 30D
0C5 18A 314 0F7 1EE 3DC 167 2CE 343 059 0B2 164
2C8 34F 041 082 104 208 2CF 341 05D 0BA 174 2E8
30F 0C1 182 304 0D7 1AE 35C 067 0CE 19C 338 0AF
15E 2BC 3A7 191 322 09B 136 26C 207 2D1 37D 025
04A 094 128 250 27F 221 29D 3E5 115 22A 28B 3C9
14D 29A 3EB 109 212 2FB 329 08D 11A 234 2B7 3B1
1BD 37A 02B 056 0AC 158 2B0 3BF 1A1 342 05B 0B6
16C 2D8 36F 001 002 004 008 010 020 040 080 100
200 2DF 361 01D 03A 074 0E8 1D0 3A0 19F 33E 0A3
146 28C 3C7 151 2A2 39B 1E9 3D2 17B 2F6 333 0B9
172 2E4 317 0F1 1E2 3C4 157 2AE 383 1D9 3B2 1BB
376 033 066 0CC 198 330 0BF 17E 2FC 327 091 122
244 257 271 23D 2A5 395 1F5 3EA 10B 216 2F3 339
0AD 15A 2B4 3B7 1B1 362 01B 036 06C 0D8 1B0 360
01F 03E 07C 0F8 1F0 3E0 11F 23E 2A3 399 1ED 3DA
16B 2D6 373 039 072 0E4 1C8 390 1FF 3FE 123 246
253 279 22D 285 3D5 175 2EA 30B 0C9 192 324 097
12E 25C 267 211 2FD 325 095 12A 254 277 231 2BD
3A5 195 32A 08B 116 22C 287 3D1 17D 2FA 32B 089
112 224 297 3F1 13D 27A 22B 289 3CD 145 28A 3CB
149 292 3FB 129 252 27B 229 28D 3C5 155 2AA 38B
1C9 392 1FB 3F6 133 266 213 2F9 32D 085 10A 214

2F7 331 0BD 17A 2F4 337 0B1 162 2C4 357 071 0E2
1C4 388 1CF 39E 1E3 3C6 153 2A6 393 1F9 3F2 13B
276 233 2B9 3AD 185 30A 0CB 196 32C 087 10E 21C
2E7 311 0FD 1FA 3F4 137 26E 203 2D9 36D 005 00A
014 028 050 0A0 140 280 3DF 161 2C2 35B 069 0D2
1A4 348 04F 09E 13C 278 22F 281 3DD 165 2CA 34B
049 092 124 248 24F 241 25D 265 215 2F5 335 0B5
16A 2D4 377 031 062 0C4 188 310 0FF 1FE 3FC 127
24E 243 259 26D 205 2D5 375 035 06A 0D4 1A8 350
07F 0FE 1FC 3F8 12F 25E 263 219 2ED 305 0D5 1AA
354 077 0EE 1DC 3B8 1AF 35E 063 0C6 18C 318 0EF
1DE 3BC 1A7 34E 043 086 10C 218 2EF 301 0DD 1BA
374 037 06E 0DC 1B8 370 03F 07E 0FC 1F8 3F0 13F
27E 223 299 3ED 105 20A 2CB 349 04D 09A 134 268
20F 2C1 35D 065 0CA 194 328 08F 11E 23C 2A7 391
1FD 3FA 12B 256 273 239 2AD 385 1D5 3AA 18B 316
0F3 1E6 3CC 147 28E 3C3 159 2B2 3BB 1A9 352 07B
0F6 1EC 3D8 16F 2DE 363 019 032 064 0C8 190 320
09F 13E 27C 227 291 3FD 125 24A 24B 249 24D 245
255 275 235 2B5 3B5 1B5 36A 00B 016 02C 058 0B0
160 2C0 35F 061 0C2 184 308 0CF 19E 33C 0A7 14E
29C 3E7 111 222 29B 3E9 10D 21A 2EB 309 0CD 19A
334 0B7 16E 2DC 367 011 022 044 088 110 220 29F
3E1 11D 23A 2AB 389 1CD 39A 1EB 3D6 173 2E6 313
0F9 1F2 3E4 117 22E 283 3D9 16D 2DA 36B 009 012
024 048 090 120 240 25F 261 21D 2E5 315 0F5 1EA
3D4 177 2EE