前言

信息论是整个信息科学发展的起源和基石，它是一门理论与应用都很强的一门学科。

提高信息传输的可靠性和有效性，始终是通信工作所追求的目标。编码分为信源编码和信道编码，信源编码主要是为了提高传输有效性，而信道编码是提高信息传输可靠性的一种重要手段。因此，学好编码课程对于从事通信方向的人员来说就显得非常重要。为了使大家能将理论用于实践，我们设置了信息论与编码技术实验课，其中信道编码实验均为综合性、设计性实验。

信源编码的作用之一是设法减少码元数目和降低码元速率，即通常所说的信息压缩。信源编码实验从数据压缩编码、语音压缩编码、音频压缩编码、图像压缩编码和视频压缩编码等几方面来帮助同学们掌握信源编。信道编码实验包括汉明编译码、循环码的编译码、RS编译码、卷积码的编译码及Turbo编译码器的设计与实现方法。

第六章 Quartus Ⅱ用法简介

&6.1 Quartus Ⅱ开发软件

Quartus Ⅱ是Altera公司为其可编程器件推出的开发软件， Quartus Ⅱ是用于10万门以上的器件。在这里仅对原理图设计输入法进行叙述，而其它方法类同，有兴趣的同学可以参考有关书籍。

原理图设计输入法的设计流程图如图6.1所示

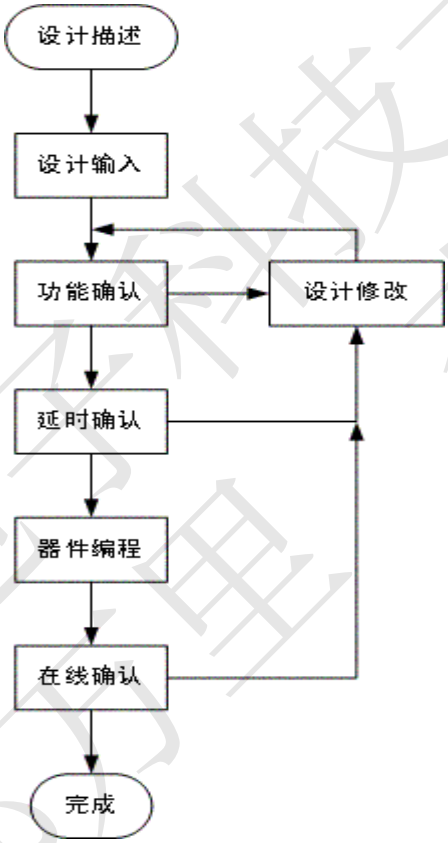


图6.1　原理图设计输入法的设计流程图

按如下步骤编译、仿真、下载文件：

⑴　设计输入: Quartus II 支持多种输入方式:图形输入文件，EDIF 网表文件，文本输入文件(.v，　.vhd，　.tdf)等。

⑵　功能仿真:将输入文件进行编译后，通过仿真器进行功能验证，如果设计规模较小，这一步可以省略。功能仿真没有模拟器件在时间上的迟延性。

⑶　逻辑综合:将源文件调入逻辑综合器进行综合，即把语言综合成最简的布尔表达式和信号的连接关系。逻辑综合软件会生成.edf(edif)的 EDA 工业标准文件。

⑷　时序仿真:对综合后的电路，加上延时信息后，验证电路的时序是否满足功能要

求。(也称后仿真)。

⑸ 布局布线:将工程的逻辑和时序要求与器件的可用资源相匹配，将每个逻辑功能分配给最好的逻辑单元位置，进行布线和时序，并选择相应的互连路径和引脚分配。

⑹ 编程下载:确认仿真无误后，将文件下载到芯片中。

下面将以由下向上设计的方法和由上向下的方法简单的介绍Quartus II的使用。由下向上设计的方法是先设计底层电路，再实现总体设计的方法。而由上向下设计的方法是先设计顶层电路，在添加各模块的功能。

&6.2 由下而上的设计方法

先设计各个局部电路，然后由各局部电路推出总体框图。

⑴ 新建原理图文件或VHDL\verilog文件，则点击FILE→New，弹出图6.2

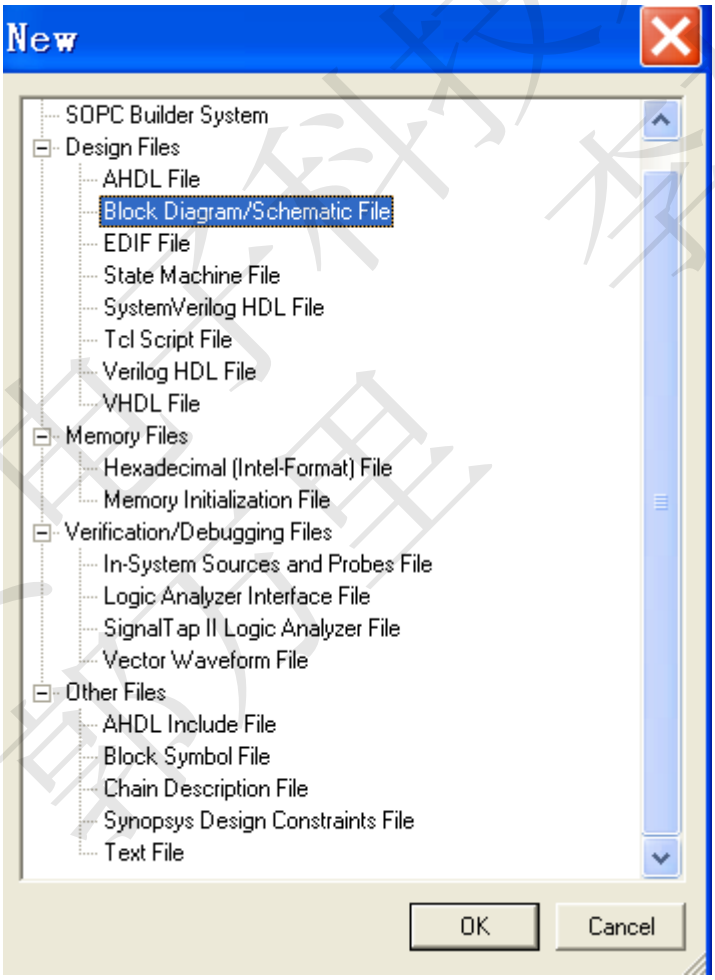


图 6.2 新建文件类型选择窗口

⑵ 如果是输入电路在图 6.2 中选中 Block Diagram/Schematic File（如果是程序可选择 Verilog HDL Files 或 VHDL Files 项），弹出图 6.3

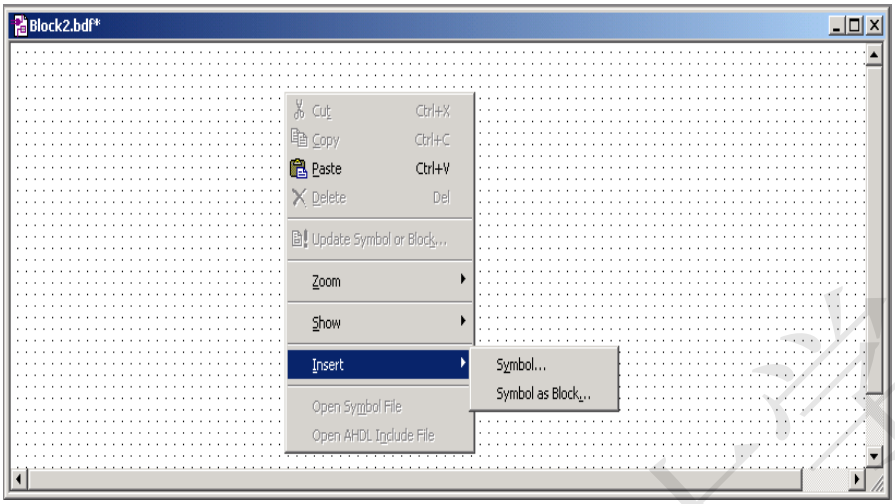


图 6.3　文件选项

在图 6.3 上点击鼠标右键，选 Insert→Symbal，弹出图 6.4

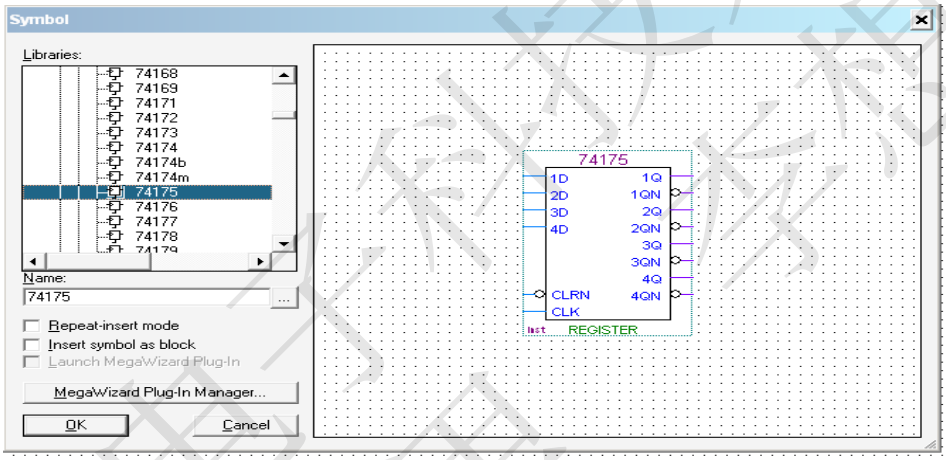


图 6.4　图形编辑窗口

(3)　在图 6.4 中选择所需器件后点击 OK 键，选中器件。这时在图 6.5 中拉动鼠标

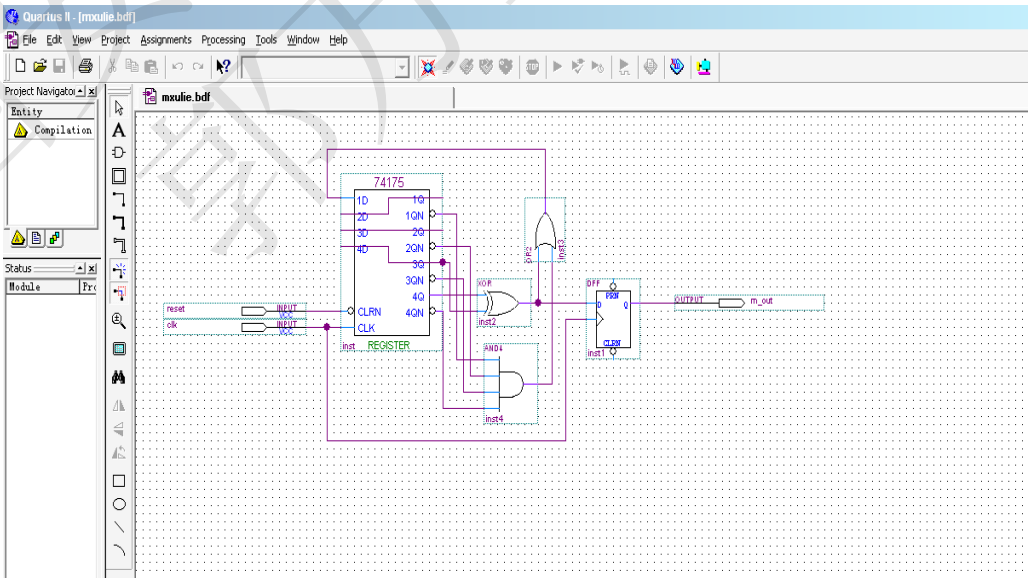


图 6.5　编辑 m 序列电路

连接你所设计的电路图。其中输入输出节点在图 6.4 中选择。输入输出节点选中之后，重新为其命名，命名的方法是直接在输入输出节点上双击鼠标右键，即可命名。直接拖动鼠标连接电路图，将你所设计好的电路连接好，如图 6.5 所示。存储电路图并命名，之后必须建立工程文件。

⑷ 编译工程文件选择暗红色向右指向的箭头，开始编译查找问题并生成一些备用的文件（其中.sof 为下载到 FPGA 中的文件，.pof 为下载到 Flash 中的文件）。特别要提示的是：工程文件里一般要包括你所要编译的有关文件，如*.bdf、*.sof、*.vwf 等，否则很可能会出现没有实体文件等这样的错误提示。要察看工程文件中所包含的具体文件时，可以点击 Projiect→add/Remove File in project，这时弹出图 6.6，在图 6.6 中你可以仔细察看一下工程文件中所含有的文件。如果没有你所要的文件，可以在图 6.6 中点击 add 进行添加，如果多余文件，可以点击 Remove 进行删除。

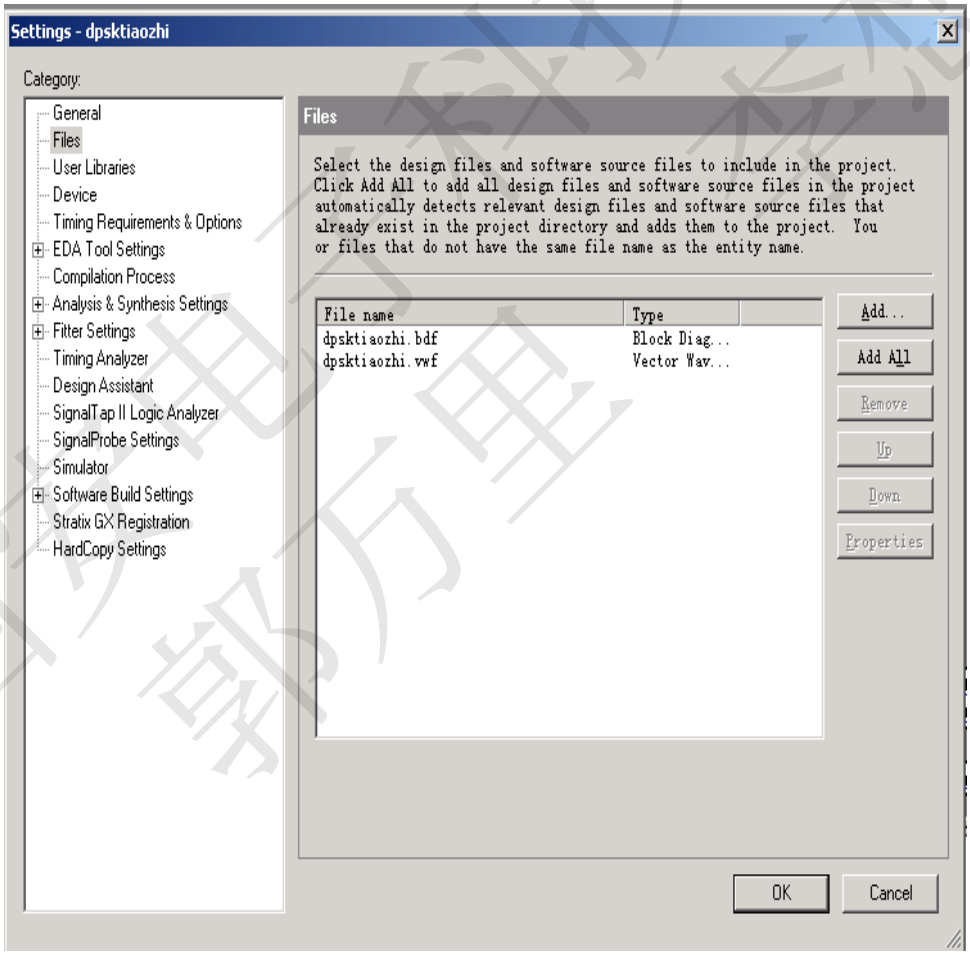


图 6.6 添加删除工程文件窗口

⑸ 建立波形文件，点击 File→New→Vector Wavefrom File。进入图 6.7，点击 Edit→End time，弹出图 6.7 中的小框，设置仿真结束时间。

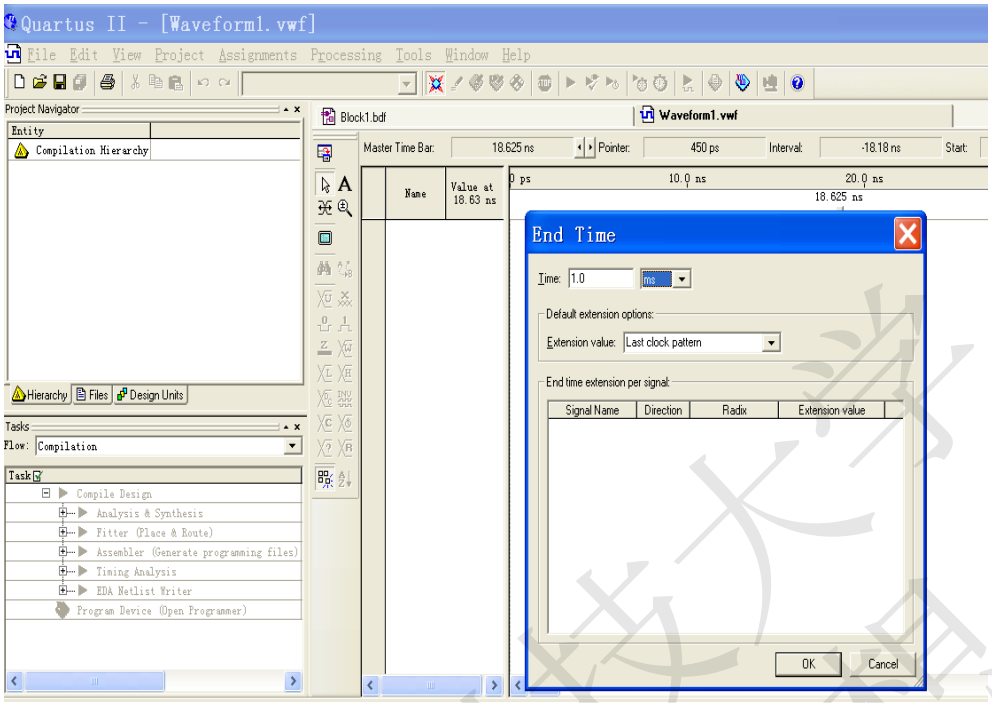


图 6.7　建立波形文件窗口

(6)　为波形文件分配输入输出节点，点击 Edit→Insert Node or Bus，进入图 6.8。在图 6.8 中，先按 List 键，然后将你要观察的输入、输出节点从图 6.8 的左框按“≥”按钮选到右框中。点击 OK 键进入图 6.9。

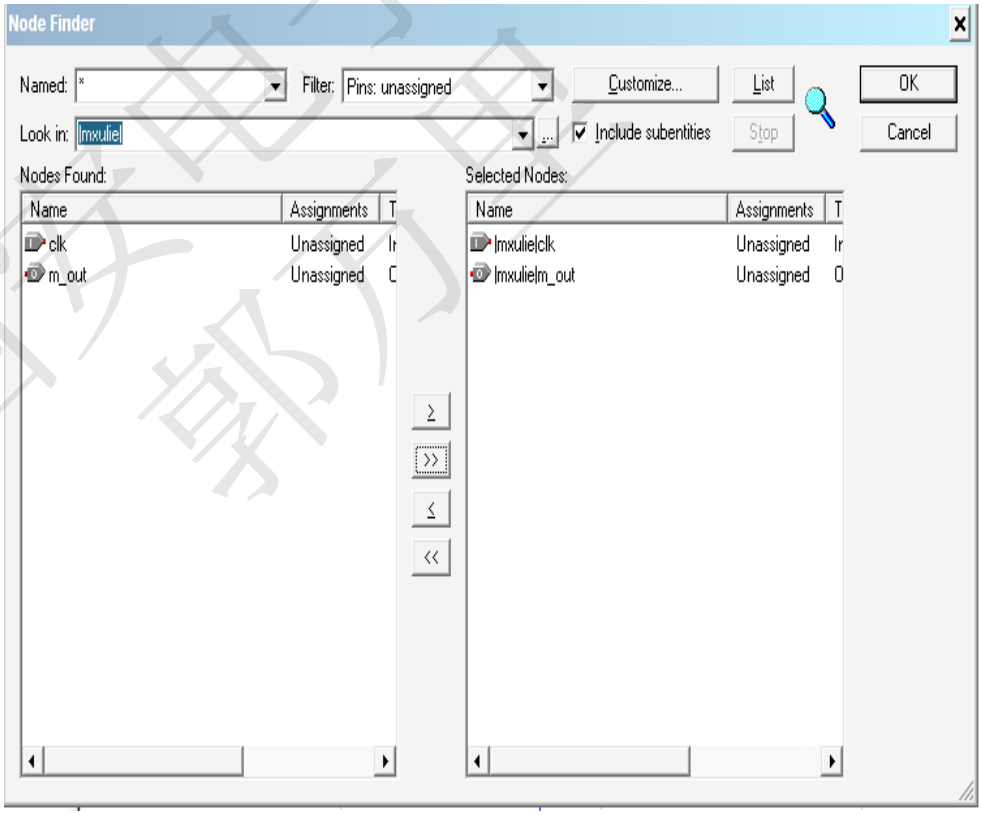


图 6.8　波形文件分配输入输出节点

在图 6.9 中，要给输入节点输入一个波形。例如给时钟 clk 输入时钟波形，首先选中它，然后点击图 6.10 中的时钟，就会出现 Clock 方框，在此方框中给定时钟一个值，例如

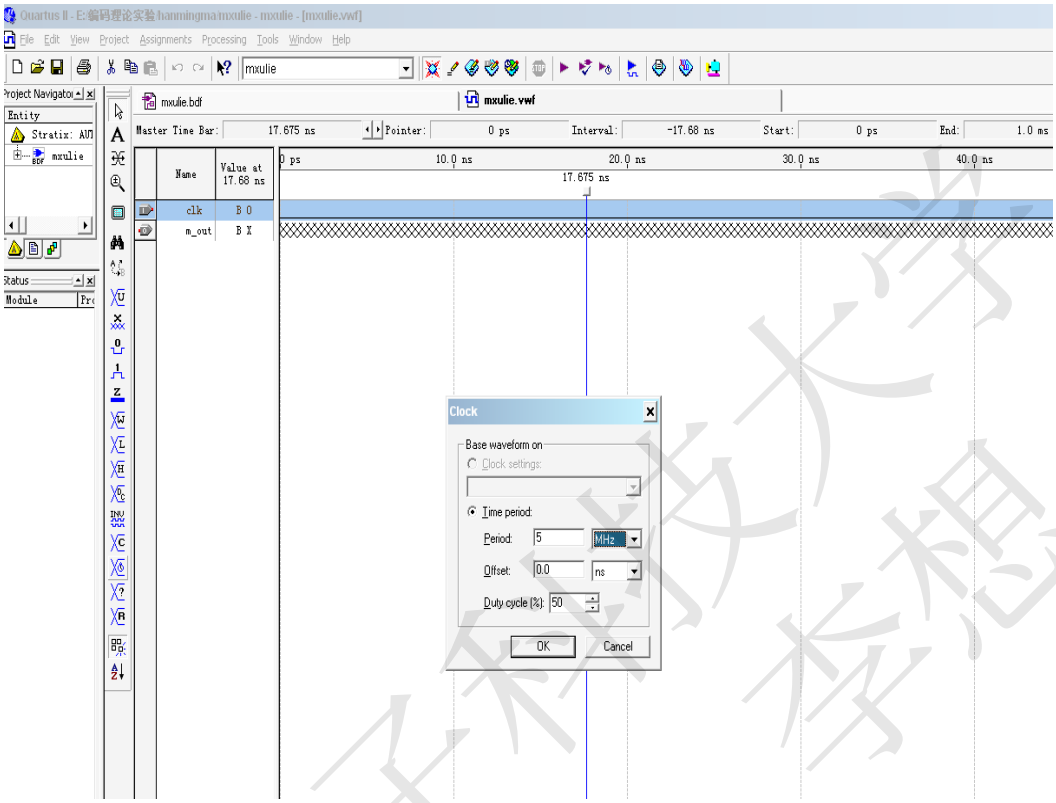


图 6.9　定义仿真时间

5MHz，点击 OK 键，clk 输入节点就有一个 20MHz 的时钟。给定时钟波形之后就可以时序仿真了。

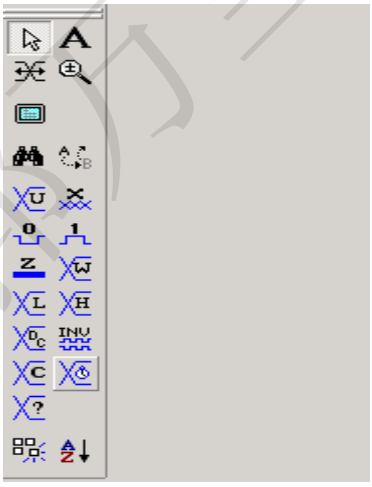


图 6.10 时钟选项

⑺ 仿真选择蓝色向右指向的箭头，开始仿真，这时的仿真是时序仿真（已设置好的），仿真完成后，你就可以看到仿真波形了。观察仿真波形是否正确，如图 6.11 所示。

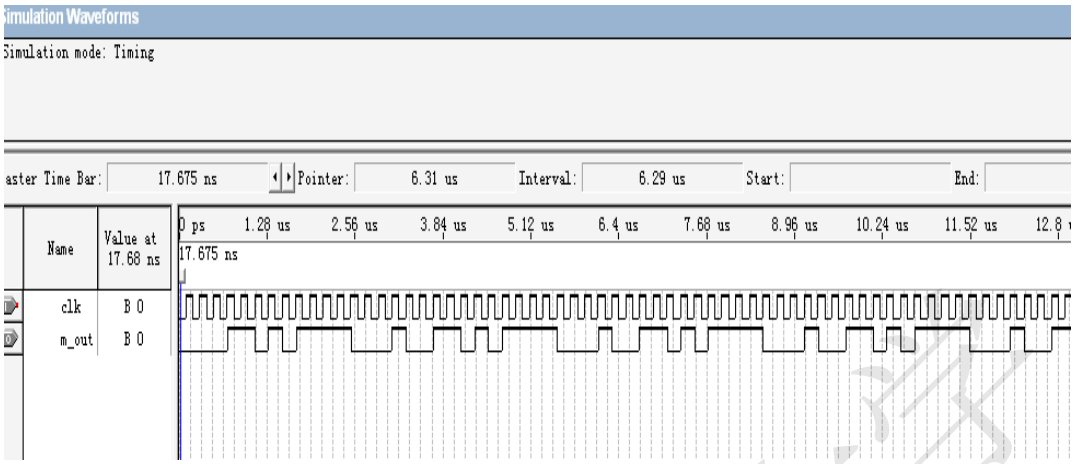


图 6.11　m 序列仿真波形

⑻　模块封装，就是将你的原理电路封装成独立的芯片，方法如下：在面板图上，点击 File→Create/Update→Create Symbol Files for Current File，可生成模块，方便以后调用。

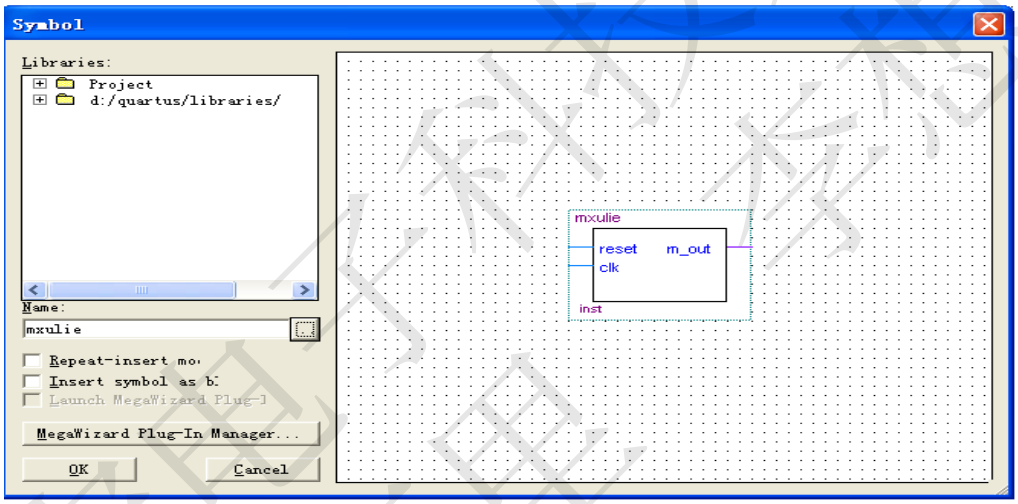


图 6.12　模块封装

⑼　下载程序到 FPGA 中。在下载程序之前，有时要设置计算机的 cmos 为：

Tntergrated Peripherhevals\Onboard Parallel Port:378/IRQ7

Tntergrated Peripherhevals\Parallel Port:ECP+EPP

设置好之后存盘并重新启动计算机。

⑽　进入 QuartusII 界面，选择 FPGA 的型号，点击 Assignment→Device，选择 FPGA 的型号为 Ep1c20F324c8。再为输入输出分配管脚，点击 Assignment→Assign Pins。

⑾　管脚分配好之后重新编译之后点击 Tools→Program，选择下载电缆，加载*.Sof（下载到 FPGA 中，Pof 下载到 flash 中）程序，点击 Start，开始下载，下载完之后，用示波器来测试输入输出。其中要下载到 Flash 中，首先将 FPGA 硬件板上的开关 2 置到 ON，然后将 Program 下的 mode 方式置到 Active Serial Program 上，点击 Start，将程序下载到

Flash 中，之后再将板子上的开关 1 置到 ON，即可。具体做法如下：

在 QuartusII 菜单中选 Assignments→Device，来选择实验板中使用的器件型号。实验板中使用的器件型号为 Cyclone 系列 EP1C20F324C8，选中它，如图 6.13 所示。

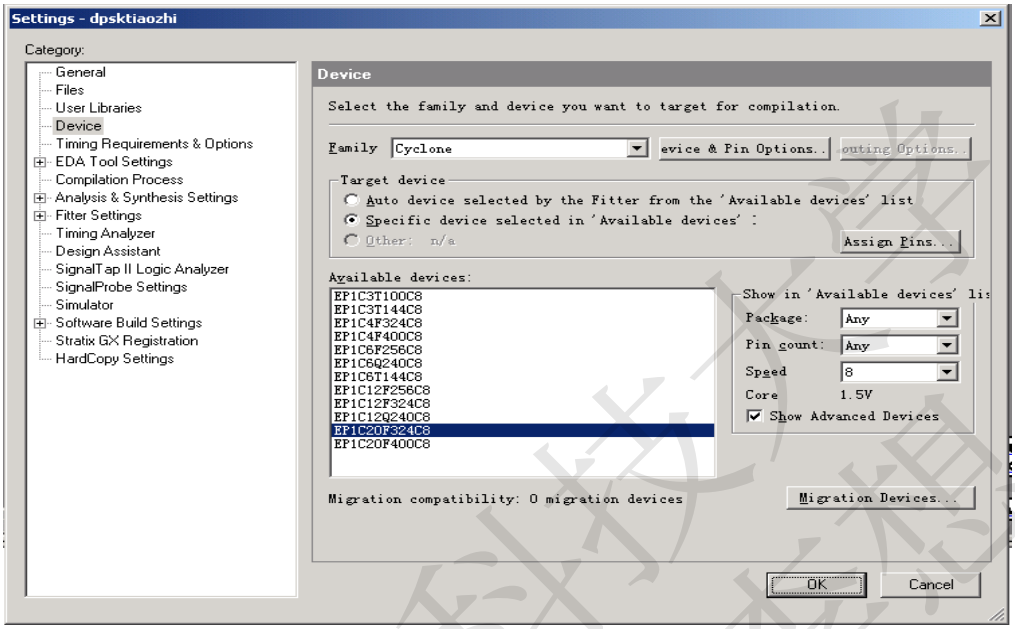


图 6.13　器件选择

为电路中的输入输出信号分配管脚，也就是每一个输入输出对应器件中的一个管脚。在图 6.13 中点击 Assign Pins，弹出图 6.14，在图 6.14 中选择器件的管脚和电路中的管脚配对，也就是说将电路的某个输入输出与器件的某个管脚相连接。

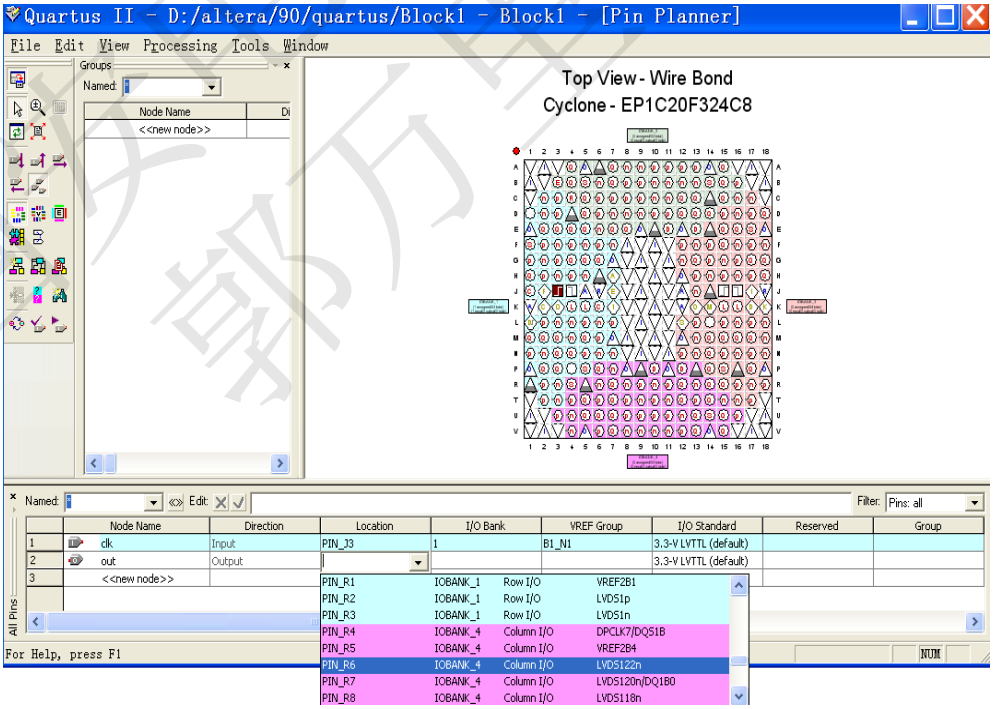


图 6.14　管脚分配

⑿ 实验板能够使用的器件管脚如下：

器件内部时钟为 20MHz：对应的管脚为 J3、J4

采样频率：对应的管脚为 A9

AD：对应的管脚为（从低到高）D5 A6 B6 C6 A7 D6 C7 B7 A8 D7 C8 B8

DA：对应的管脚为（从高到底）F2 G2 H3 H1 L3 M3 M2 L2 L4 H2 G1 G3

数字输入：对应的管脚为（在实验板上从 1 到 8）U6 T7 V7 U7 U8 T8 T9 V8

数字输出：对应的管脚为（在实验板上从 1 到 8）R6 T4 U5 V4 R5 R8 T6 V6

器件及管脚分配完之后将电路重新编译，系统将生成一个*.sof 文件，将此文件下载到实验板的 FPGA 中。具体做法是：选择 Tool→Programmer，将弹出图 6.16：

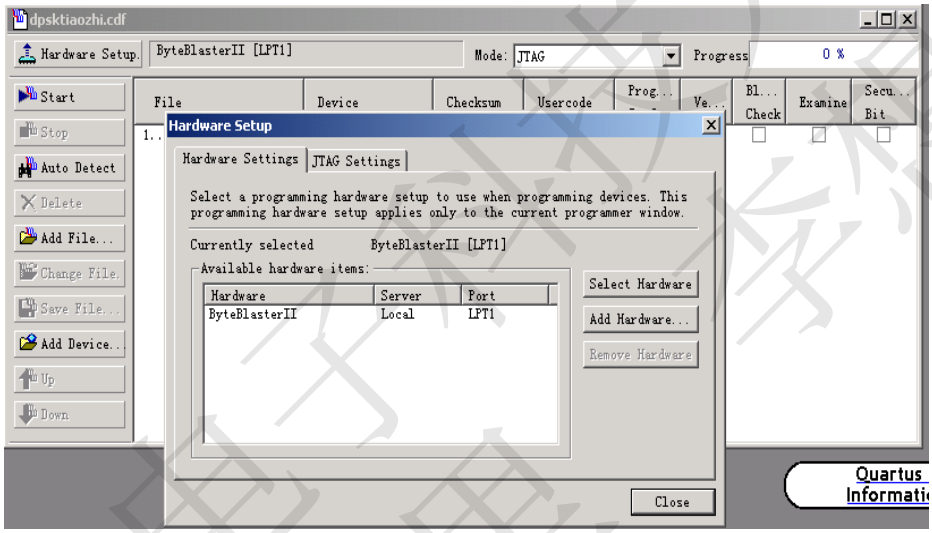


图 6.16 下载电路设置

在图 6.16 中选择 Hardware Setup 来选择并口电缆，并口电缆名为 ByteBlasterII LPT1，如图 6.17。

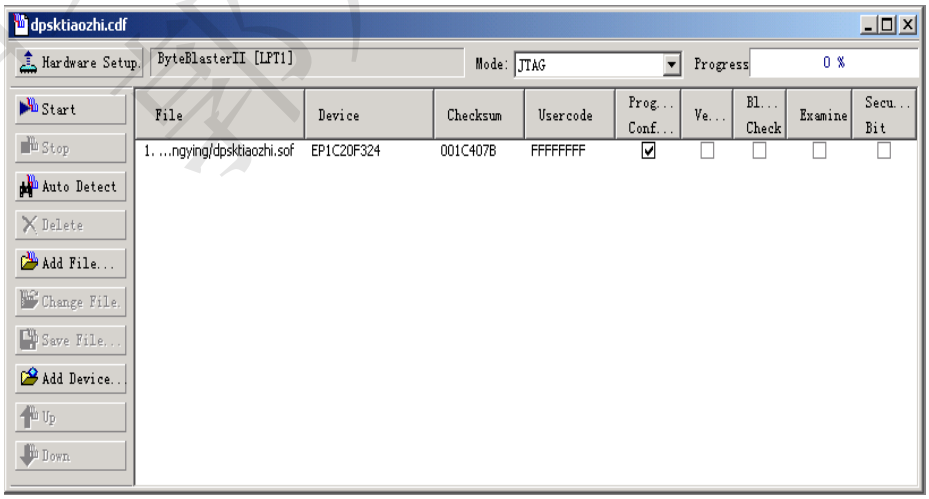
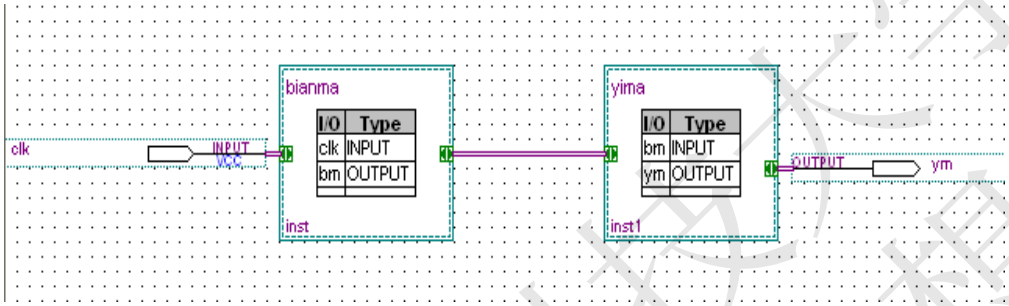


图 6.17 选择下载方式

这时在窗口中出现你的扩展名 sof 的文件名，点击 Prog 来选中它。按 Start 按钮下载你的程序。当 Progress 窗口先显示 100%时，程序下载完成，这时你可接入示波器观察波形。

&6.3 由上而下的设计方法

由上而下的设计方法是先建立图表模块。也就是先设计总体框图，然后再设计局部电路。如下图中的编译码器设计：



图表模块编辑器既可以编辑图表模块，又可以编辑原理图。图表模块编辑是顶层设计的主要方法，原理图编辑器是传统的设计输入方法。

(1) 用模块编辑器产生一个新的文件

选择File→New→Block Diagram/Schematic File项产生一个新的图表模块/原理图文件。然后点击OK按钮，并保存文件，如图6.18所示。

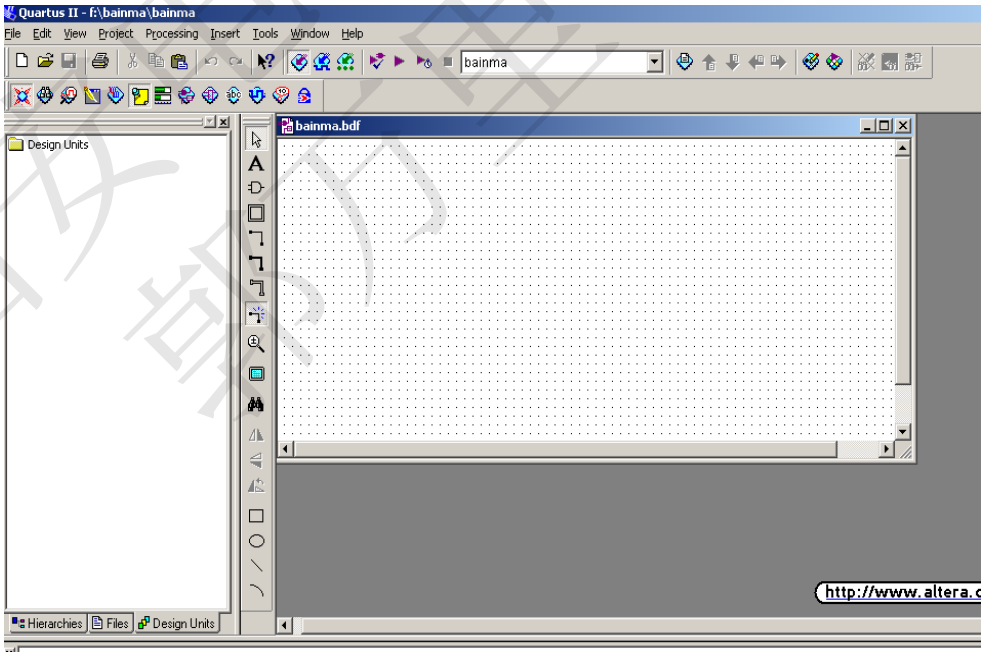


图6.18 新的图表模块/原理图文件

(2) 用模块编辑器设计模块

① 从工具兰中产生模块，工具栏包括很多工具，你可以根据需要选择。例如点击工具栏中的模块图表来画一个模块，鼠标托动确定模块的大小，点击鼠标右键，在弹出图6.19的对话框中选择属性。在图6.19中，其中

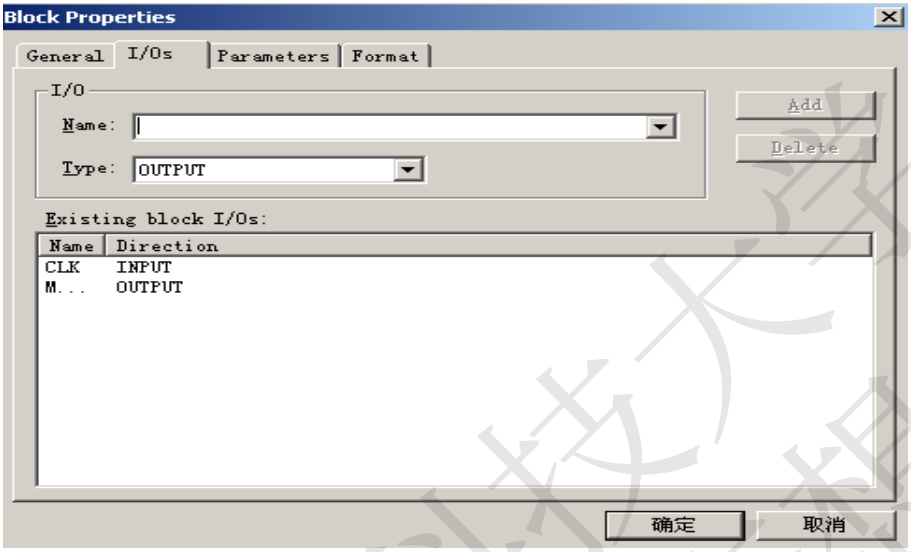


图6.19　定义模块输入输出

②　在模块属性对话框中点击General，输入模块名称，如mxulie。

③　在模块属性对话框中点击I/Os，可键入输入输出端口名并选择相应的端口类型，然后点击Add按钮添加。如在电路M序列发生器中，输入端口名为CLK，在Type栏中选择INPUT，同理，输出端口名为MXULIE，在Type栏中选择OUTPUT。

④　点击确定按钮，用鼠标右键点击模块，选择AutoFit，模块中的数据都可以看见，如图6.20所示。

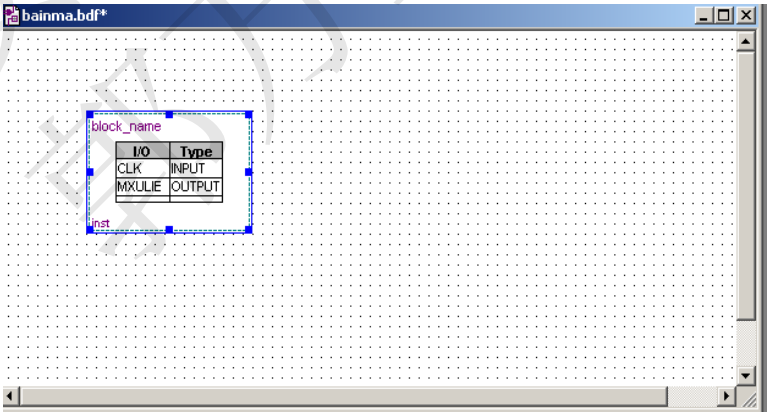


图6.20　确定模块

建好所有模块之后，建立工程文件。然后给每一个模块添加实际内容。其操作步骤是：在确定模块上点击鼠标右键点，选择create design file from selected block选项，在弹出图6.21的对话：

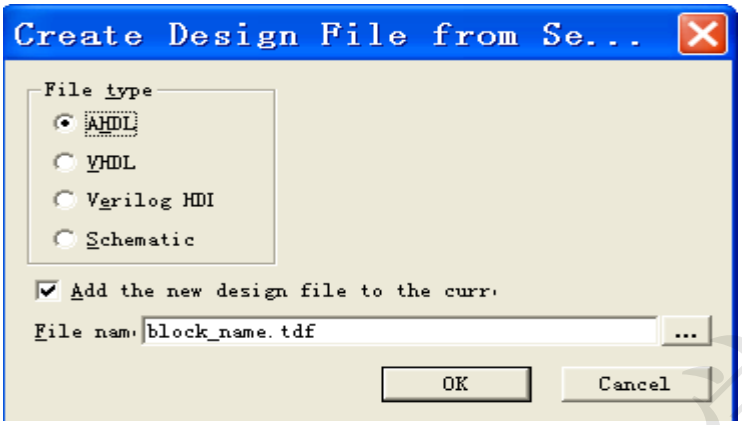


图6.21　选择模块属性

框中选择你需要的输入形式，然后点击 ok。这时就可以输入程序或电路图了。其它编译仿真的过程可以参考由下而上的设计方法，在这里就不再重复。

&6.4 用嵌入式逻辑分析仪测试数据

首先选择器件、分配管脚，并重新编译。之后做：

建立 STP 文件

在 QuartusII 软件中，选择 Tool→SignalTapII Logic Analyzer，弹出如图 6.22 窗口

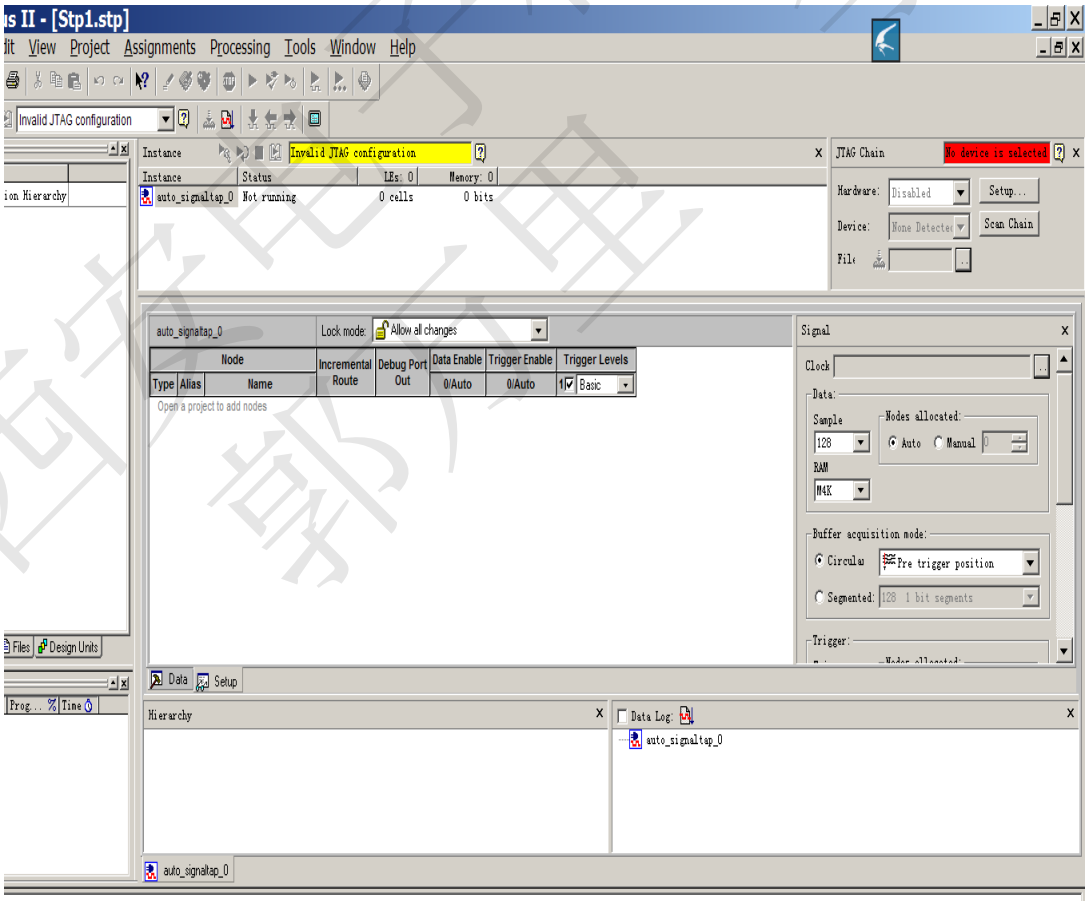


图 6.22　逻辑分析仪面板

在 Instance 栏，可将 auto_signaltap_0 改成你自己需要的名字。只需要在 auto_signaltap_0 处，点击鼠标右键，出现如图 6.23 所示：

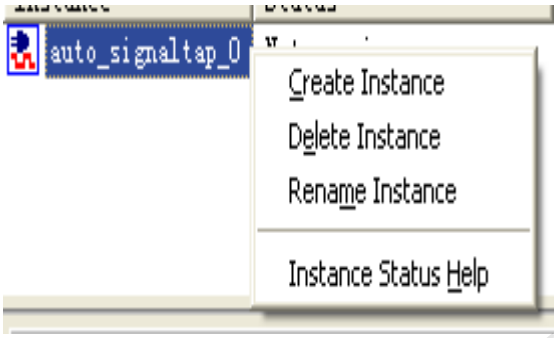


图 6.23　STP 文件更名

选择 Rename Instance，就可以改名了。

在使用 SignalTapII 逻辑分析仪进行采样之前，首先应该设置采样时钟。采样时钟在上升沿处采集数据。一般可以采用任意信号作为采集时钟，但原则上最好使用全局时钟。

设置 SignalTapII 采集时钟的步骤如下：

(1)　在 SignalTapII 逻辑分析仪窗口选择 Setup 标签页。

(2)　点击 Setup，选择硬件口-并口。

(3)　点击 clock 栏，弹出如图 6.24 所示窗口

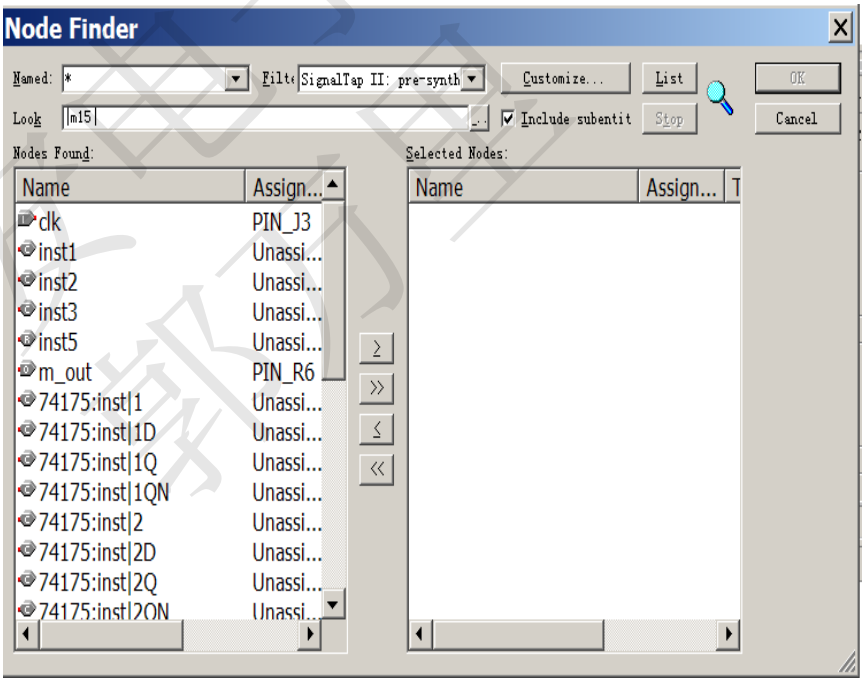


图 6.24　选择信号作为采集时钟

在该窗口中选择一个信号作为采集时钟。

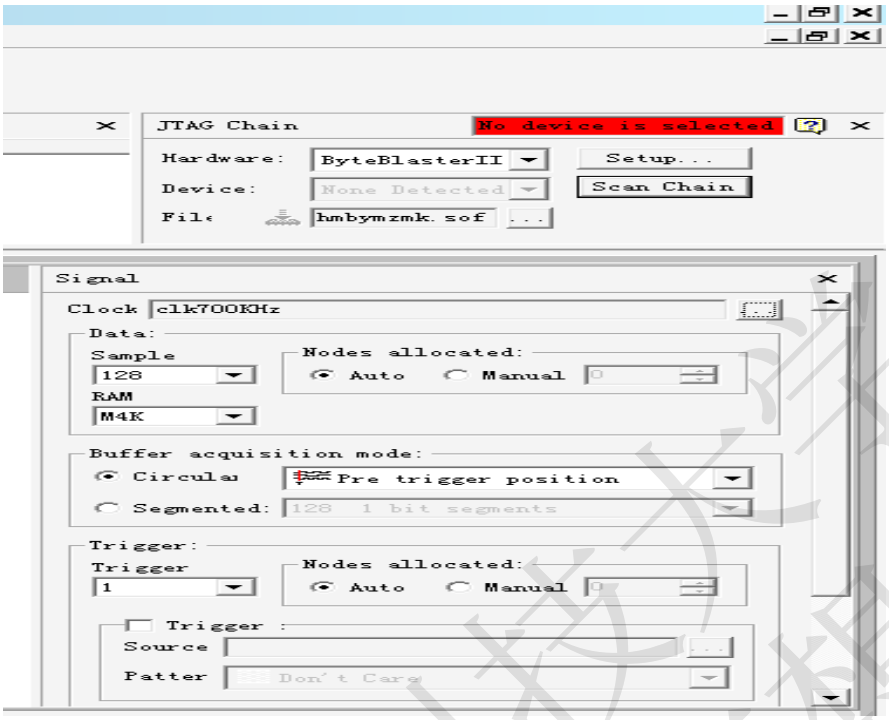


图 6.25　定义采集时钟

分配数据信号

在空白处双击鼠标，弹出图 6-26，在图中选择输入输出信号，如图 6.26 所示

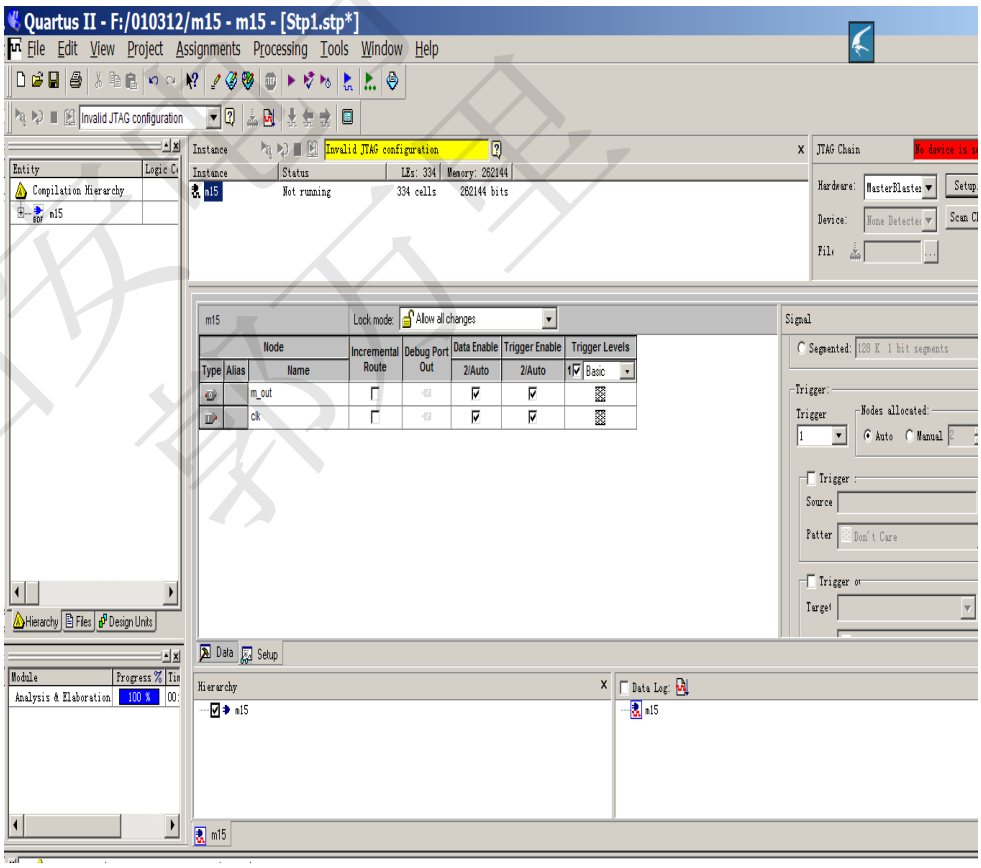


图 6.26　分配数据

设置触发类型：Basic 或 Advanced

设置采样点数：128K

重新编译逻辑分析仪，编译完成后，重新将.sof 文件下载到实验板中，重新打开 STP 文件，完成如下工作：

Setup：选择硬件接口 ByteBlaseterII(LPT1)

Device：选择目标器件

SOF：选择编程文件（.Sof）

察看 SignalTapII 逻辑分析仪采样数据

在 SignalTapII 逻辑分析仪窗口中，选择 Run Analysis 或 AutoRun Analysis 按钮启动 SignalTapII 逻辑分析仪。

Run Analysis：单步执行 SignalTapII 逻辑分析仪。

AutoRun Analysis：执行该命令后，SignalTapII 逻辑分析仪连续捕获数据，直到用户按下 Stop Analysis 为止。

Stop Analysis：停止 SignalTapII 逻辑分析仪。

Read Data：显示捕获的数据。如图 6.27 所示

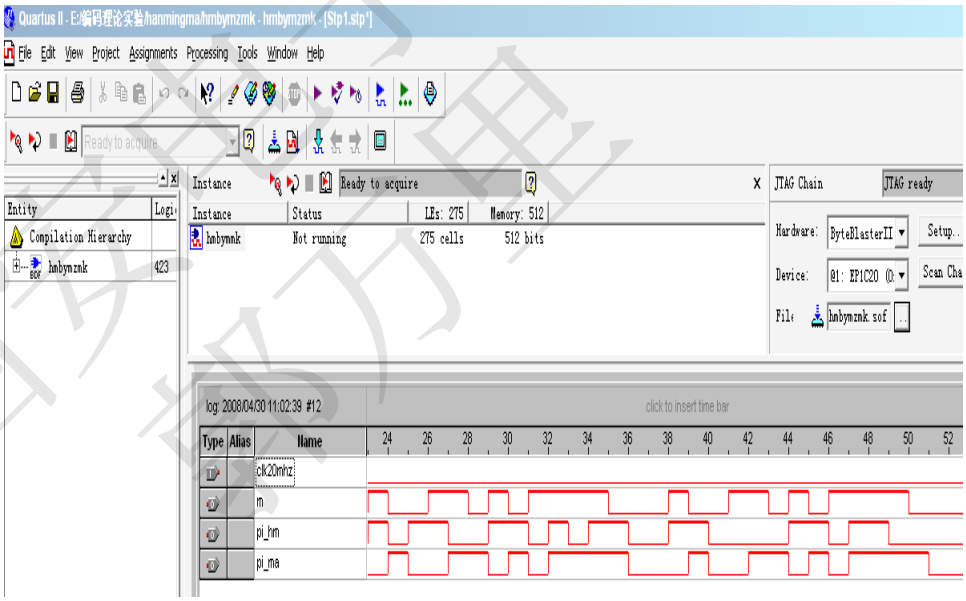


图 6.27　逻辑分析仪反馈 m 序列数据

&6.5 计数器的使用

计数器的使用

计数器电路可以自己设计，也可以使用 QuartusII4.0 库中的器件。在这里只讲述

QuartusII4.0 库中的计数器的用法。

首先进入图 6.28 窗口，按图 6.28 所示选中器件 LPM_COUNTER，接着按以下提示窗口做

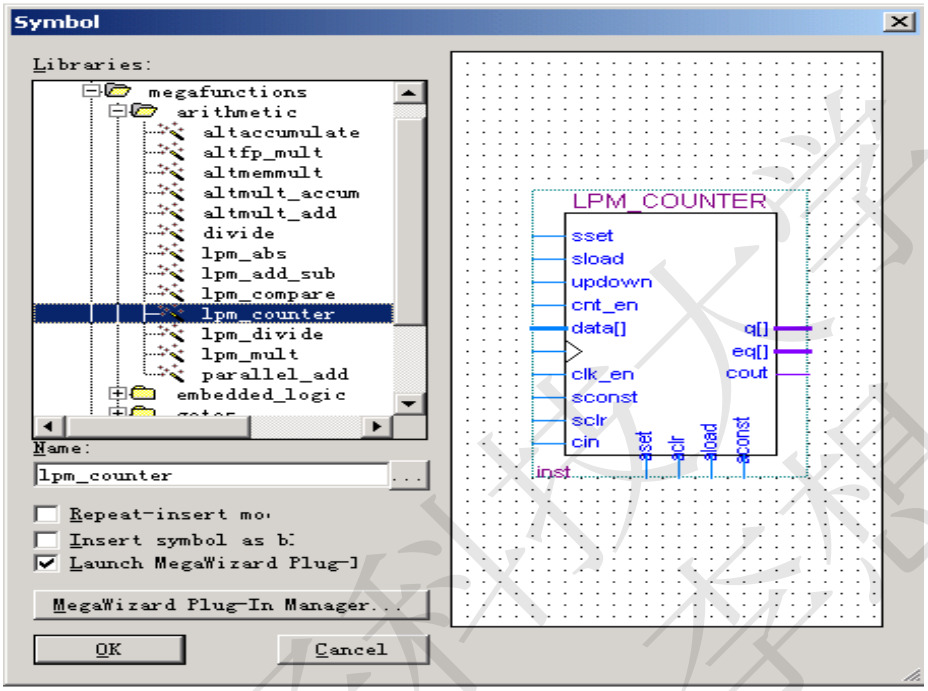


图 6.28　选择计数器模块

在图 6.29 窗口为计数器起名，盘符：文件名，点击 NEXT 进入图 6.30。

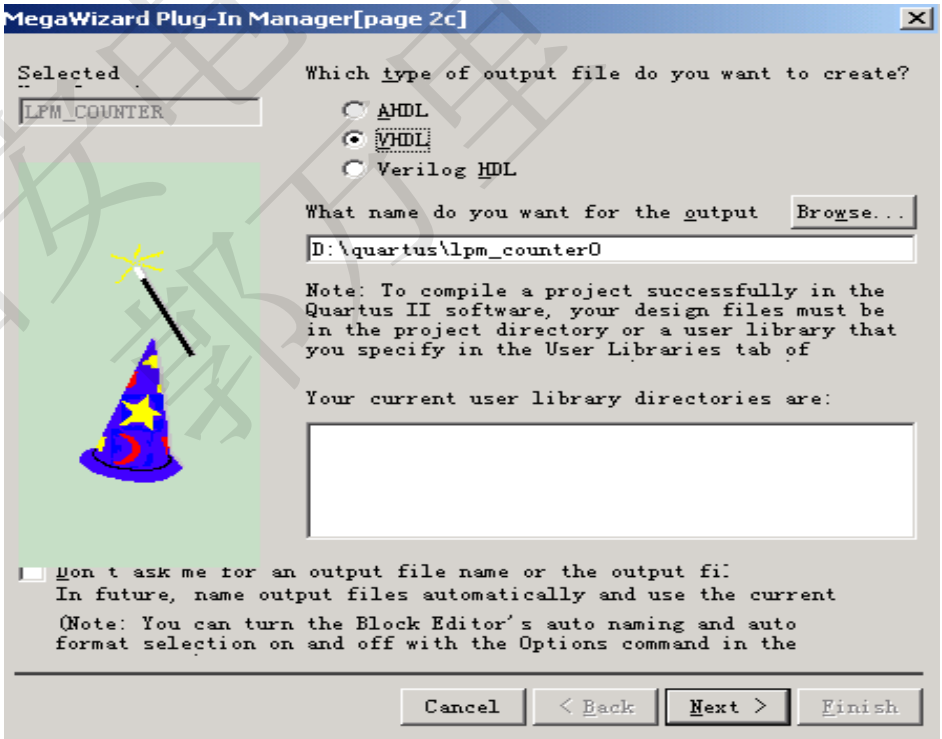


图 6.29　为计数器命名

在图 6.30 为计数器设计时钟（上升沿有效、下降沿有效、通过控制端设定上升沿有效或下降沿有效），选择的是上升沿有效。q 为计数器的输出位数，设为 5 位输出端、32 进制计数器。点击 NEXT 进入下一步。根据对计数器的不同要求，按照提示设定好所需的计数器。

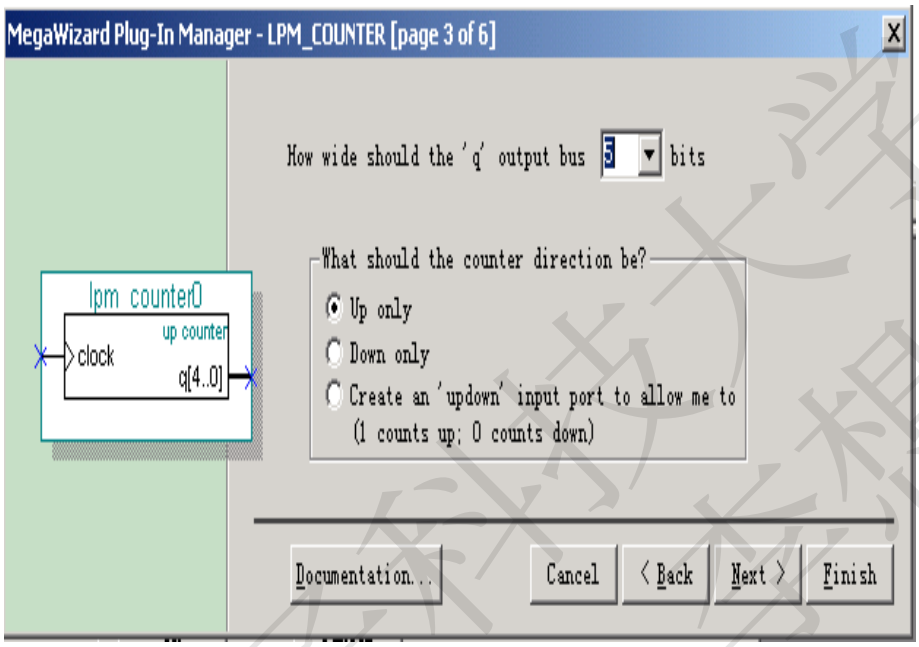


图 6.30　定义计数器输入输出

&6.6 ROM 的用法

进入图 6.31 窗口，按图所示选定器件 Lpm_rom，点击 OK 键进入图 6.32。

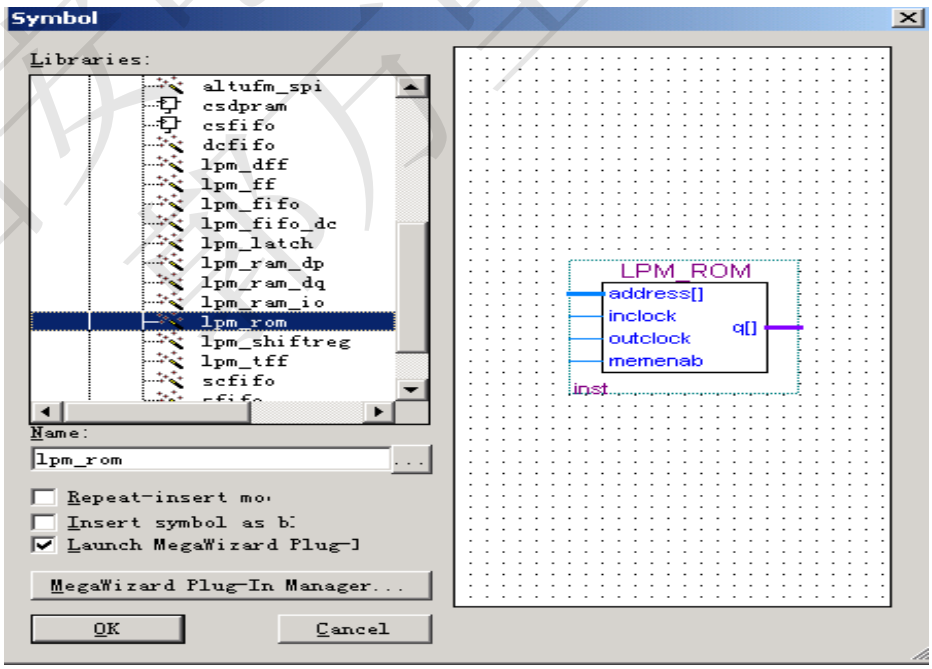


图 6.31　选择 rom 模块

在图 6.32 中，根据需要设定 ROM 的参数。将 ROM 设为器件系列 Cyclone、地址线为 5 位、输出端为 12 位、一个时钟输入。进入下一步，点击 NEXT，进入图 6.33，

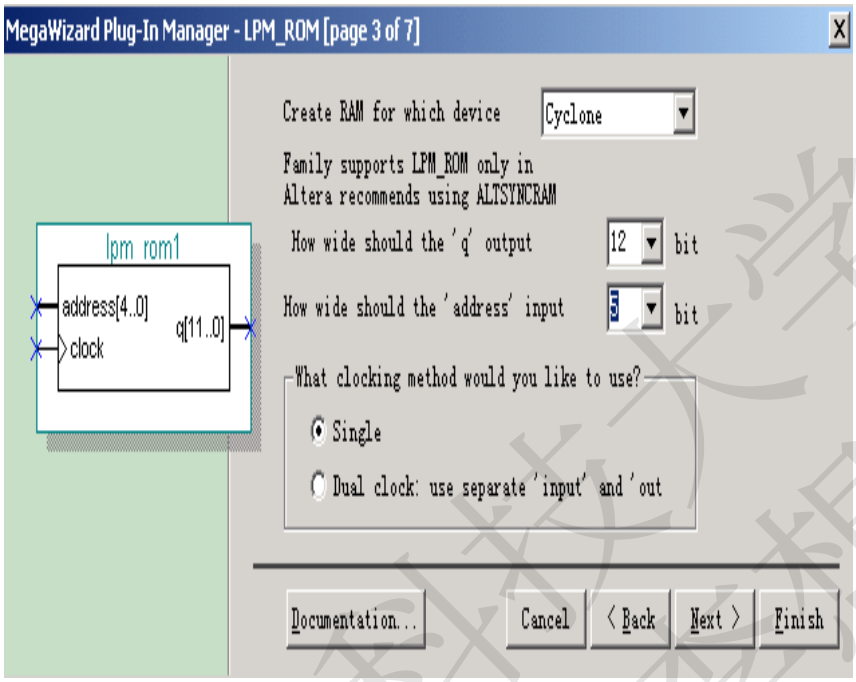


图 6.32　定义 rom 输入输出

设定时钟允许端，可以根据要求设置清零等。

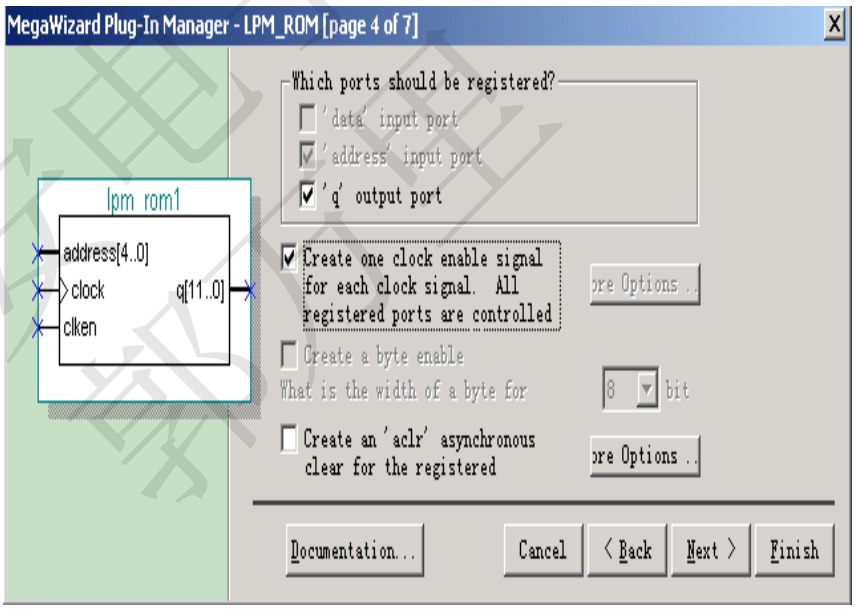


图 6.33　设定时钟允许端

进入图 6.34，写入 ROM 数据文件名，ROM 数据文件的扩展名是 mif。在这里仅仅是写入了 ROM 的文件名，ROM 文件的初始化内容有两种方法写入。其一是手工输入的方法，

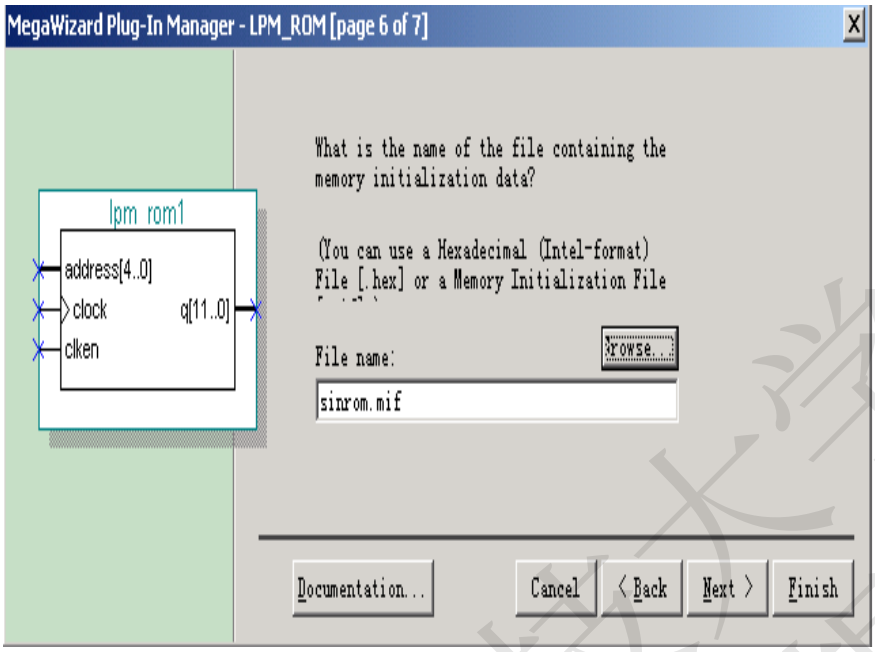


图 6.34　添加 rom 初始化文件

适用于数据量不大的情况；其二是利用计算机高级语言初始化 ROM，适用于数据量大的情况。

对于第一种方法，首先在 QuartusII4.0 菜单下选择 File→New→Other，进入 6.35，按图所示，双击 Memory Initialization File，进入　图 6.36。

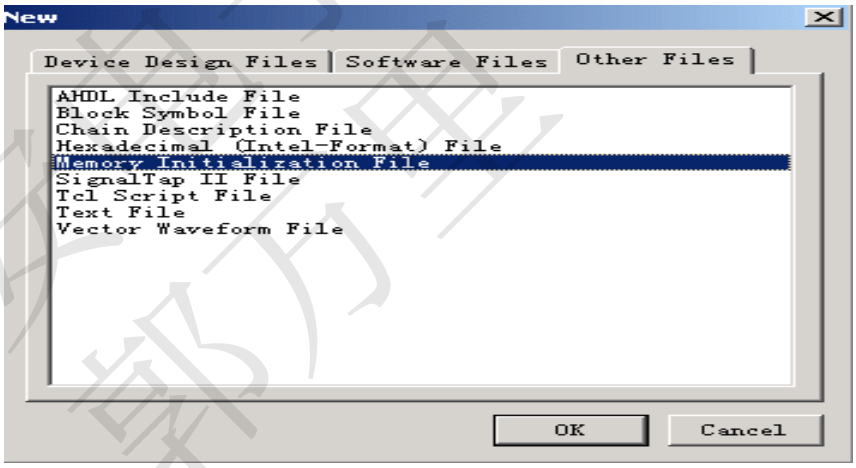


图 6.35　定义 rom 初始化文件

Number of:设存储单元的大小。

Word size：设置每个存储单元位数，在 图 6.36 中，设 32 个存储单元，为个单元由 12 位二进制数表示。点击 OK 进入 图 6.37。在 图 6.37 中就可以手工输入数据了，数据输入完之后，记住存储文件以防数据丢失。

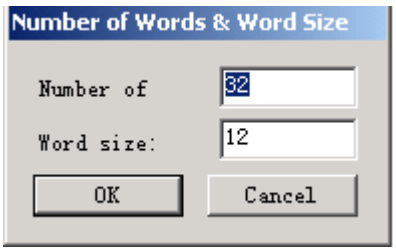


图 6.36　设置数据单元

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	4095	0	1024	0	0	0	0	0
8	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0

图 6.37　添加 rom 文件数据

到此 ROM 就可以用了。

对于第二种方法，就是利用高级语言程序编程，将结果计算好，然后将结果存储到.mif 文件中去。无论使用那种方法，其结果都是相同的。

&6.7 FPGA 实验板简介

信息系统综合实验硬件平台的结构框图如图 6.38 所示。

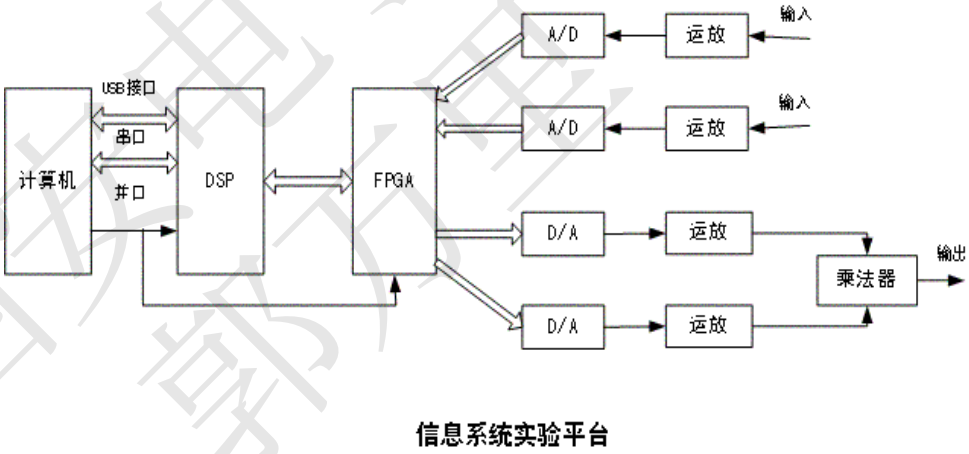


图 6.38　实验系统框图

实验平台包括如下结构：

(1)　母板：

母板包括信号采集与输出部分，FPGA 接口部分，DSP 接口部分，计算机接口部分，通用电源部分。

信号采集与输出部分包含模拟信号的采集与输出和数字信号采集与输出。模拟信号的

采集与输出的作用：一是通过 AD 转换器采集模拟信号传输给 FPGA 进行处理，母板上提供两个独立的采集通道；二是通过 DA 转换器将从 FPGA 输出的数据信号转换为模拟信号输出，母板上提供的是两路 DA 转换后进行乘法调制输出。数字信号采集与输出的作用是由母板提供数据信号到 FPGA 的输入和 FPGA 的数据信号输出接口。

FPGA 接口部分包含计算机下载接口，数据采集接口，数据输出接口，电源接口，DSP 接口。计算机下载接口的作用是提供标准的并口数据到 FPGA 的数据转换，完 FPGA 的程序下载功能；数据采集接口作用是提供从 AD 转换器和数字信号输入接口到 FPGA 的连接；数据输出接口的作用是提供 FPGA 到 DA 转换器和数字输出接口的连接；电源接口的作用是给 FPGA 提供 1.5V 和 3.3V 的电源供应；DSP 接口的作用是提供 F PGA 与 DSP 的连接。

DSP 接口部分包含计算机接口，FPGA 接口，电源接口。计算机接口的作用是提供计算机的并口、串口和 USB 接口与 DSP 子模快的连接；FPGA 接口的作用是提供 DSP 与 FPGA 之间的数据连接；电源接口的作用是给 DSP 提供 5V 和 3.3V 电源供应。

计算机接口部分包含并口，串口，USB 接口，作用是提供母板与计算机的数据连接。

通用电源部分的作用是提供系统所需的各种电源供应，包括+5VA、-5VA、+15VA、-15VA、+5V，+3.3V、+1.5V。

⑵ DSP 部分：

DSP 子板利用从主机下传的程序对从 FPGA 子板获得的数据进行处理，主机的程序是通过并口并采用 HPI 方式传输的。在程序的下传过程中，程序同时存储在了 FLASH 存储器中，这样，可以保证在出现断电等意外情况时，程序不会丢失。数据处理后的结果通过 USB 接口和串行接口回复给主机，完成了对数据的数字信号处理。

⑶ FPGA 子板功能：

FPGA 子板从母板接收到数据（数据采集接口），经处理后将数据送到 DSP 子板（DSP 接口）。DSP 子板对数据进行运算后将数据返回到 FPGA 子板，FPGA 将数据处理后再送回母板（数据输出接口）。

⑷ FPGA 子板接口：

计算机下载接口，数据采集接口，数据输出接口，DSP 接口，电源接口。

FPGA 的程序可经由母板上的并口下载到 FPGA 内或者直接通过 FPGA 子板上的 JTAG 接口下载到 FPGA 内。下载模式采用 AS 模式和 JTAG 模式相结合的方式。

FPGA 子板主要芯片：

EP1C20　　FPGA 芯片

EPCS4　　配置芯片

这块实验板能够完成：信道编码、随机信号分析、数据采样、信息处理、通信系统、自动控制、锁相环路、数字电路、虚拟仪器等，系统可扩展，在硬件平台不变的情况下，扩充软件使系统不断改进升级。

第七章　编码理论综合实验实例

实例：基于 FPGA 的汉明码编码与译码器的设计与实现。

& 7.1 实验目的

⑴　了解 FPGA 软硬件知识，学习利用 QuartusII 实现电路的方法，能初步掌握之。

⑵　通过汉明编译码初步掌握编译码实现的方法。

& 7.2 实验仪器

1、512MHz 以上内存微计算机。

2、20MHz 双踪示波器、信号源。

3、Quartus Ⅱ语言环境。

4、fpga 实验板、若干导线。

& 7.3 FPGA 软硬件基础知识、QuartusII 的使用举例

§7.3.1 FPGA 的发展状况

数字集成电路发展到现在，主要有三类电子器件：存储器、处理器、逻辑器件。实验中涉及到的 FPGA 就是逻辑器件。逻辑器件分两类：固定的或定制的、可编程的或可变的。其中固定的或定制的通常称为专用芯片（ASIC）而可编程的或可变的通常称为可编程逻辑器件。而 FPGA 属于可编程逻辑器件。

⑴　FPGA 是什么

FPGA 是英文 Field－Programmable Gate Array 的缩写，即现场可编程门阵列，是在 PAL、GAL、CPLD 等可编程器件的基础上进一步发展的产物。是作为专用集成电路（ASIC）领域中的一种半定制电路而出现的，既解决了定制电路的不足，又克服了原有可编程器件门电路数有限的缺点。 自 1985 年 Xilinx 公司推出第一片现场可编程逻辑器件（FPGA）至今，FPGA 已经走过了二十多年的发展历史。

FPGA 采用了逻辑单元阵列 LCA（Logic　Cell　Array）这样一个新概念 。现场可编程逻辑器件从最初的 1200 个可利用门,发展到 90 年代的 25 万个可利用门,至今,国际上现场可编程逻辑器件的著名厂商 Altera 公司、Xilinx 公司已陆续推出了上千万门的单片 FPGA 芯片,将现场可编程器件的集成度提高到一个新的水平。FPGA 不仅可以解决电子系统

小型化、低功耗、高可靠性等问题,而且其开发周期短、开发软件投入少、芯片价格不断降低,促使 FPGA 越来越多地取代了 ASIC 的市场,特别是对小批量、多品种的产品需求,使 FPGA 成为首选。

目前,FPGA 的主要发展动向是：芯片朝着高集成度、低功耗、高可靠性、低成本方向发展; FPGA/CPLD 与 CPU、DSP、EDA（电子设计自动化）和 RTOS（嵌入式实时系统）等结合，共创用于数字电子系统设计和开发的平台级 FPGA。对微处理器和先进的 ASIC 在处理和系统设计方法上有所突破和创新，现在已在 FPGA 技术上得到体现，平台级 FPGA 是在单个 FPGA 器件中集成多种硬和软 IP 核，在 FPGA 中嵌入处理器。FPGA 芯片内嵌入的处理器包括 CPU、微处理器和微控制器。

⑵ FPGA 的产品：

目前 FPGA 的品种很多，有 XILINX 的 XC 系列、TI 公司的 TPC 系列、ALTERA 公司的 FIEX 系列等。

§7.3.2 FPGA 的硬软件结构

⑴ FPGA 的硬件结构

FPGA 内部最主要的部件是 CLB（Configurable Logic Block，可配置逻辑块）、Input/Output Block（输入/输出块）和 BlockRAM（块 RAM）。

CLB 是 FPGA 具有可编程能力的主要承担者。通过配置这些 CLB 可以让 FPGA 实现各种不同的逻辑功能。

Input/Output Block 分布在 FPGA 的周边，也具有可编程特性，可以配置支持各种不同的接口标准，如 LVTTL（电平式晶体管-晶体管逻辑）、LVCMOS（低压互补金属氧化物半导体）、PCI（外设组件互联标准）和 LVDS（低压差分信号）等。BlockRAM 是成块的 RAM，可以在设计中用于存储数据，是设计的重要资源。在大规模设计选择 FPGA 时，RAM 资源是否够用是重要的考虑因素。除了 CLB、Input/Output Block 和 BlockRAM 以外，FPGA 还有很多其它的功能单元，例如布线资源、DCM（Digital Clock Manager，数字时钟管理器）和 Multiplier（乘法器）等。

⑵ FPGA 不同部件的分布图如图 7.1 所示

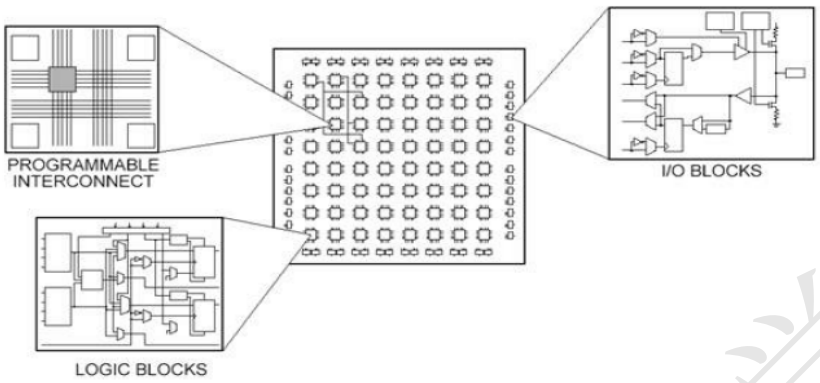


图 7.1　FPGA 不同部件的分布图

布线资源在 FPGA 内部占用硅片面积很大，为 FPGA 部件提供灵活可配的连接；DCM 模块提供各种时钟资源，包括多种分频、移相后的时钟；Multiplier 为 18bit×18bit 硬件乘法器，可以在一个时钟周期内完成乘法运算在高级的 FPGA 中，还包含了嵌入式处理器、DSP 模块、以太网 MAC、高速串行 IO 收发器等。

⑶　FPGA 的工作状态

FPGA 是由存放在片内 RAM 中的程序来设置其工作状态的。因此，工作时需要对片内的 RAM 进行编程。用户可以根据不同的配置模式，采用不同的编程方式。

加电时，FPGA 芯片将 EPROM 中数据读入片内编程 RAM 中，配置完成后，FPGA 进入工作状态。掉电后，FPGA 恢复成白片，内部逻辑关系消失，因此，FPGA 能够反复使用。FPGA 的编程无须专用的 FPGA 编程器，只须用通用的 EPROM、PROM 编程器即可。当需要修改 FPGA 功能时，只需换一片 EPROM 即可。这样，同一片 FPGA，不同的编程数据，可以产生不同的电路功能。因此，FPGA 的使用非常灵活。

⑷　FPGA 有多种配置模式

并行主模式为一片 FPGA 加一片 EPROM 的方式；主从模式可以支持一片 PROM 编程多片 FPGA；串行模式可以采用串行 PROM 编程 FPGA；外设模式可以将 FPGA 作为微处理器的外设，由微处理器对其编程。

⑸　我们在实验中所用到的 FPGA 芯片

型号：EPIC20F324C8 芯片

逻辑单元个数：20060

存储容量：288k

锁相环：2

管脚数目：233

(6) FPGA 的应用领域

FPGA 在通信、数据处理、网络、仪器、工业控制、军事和航空航天、消费电子、汽车等众多领域得到了广泛应用。

(7) EPIC20F324C8 芯片系统连接图如图 7.2 所示

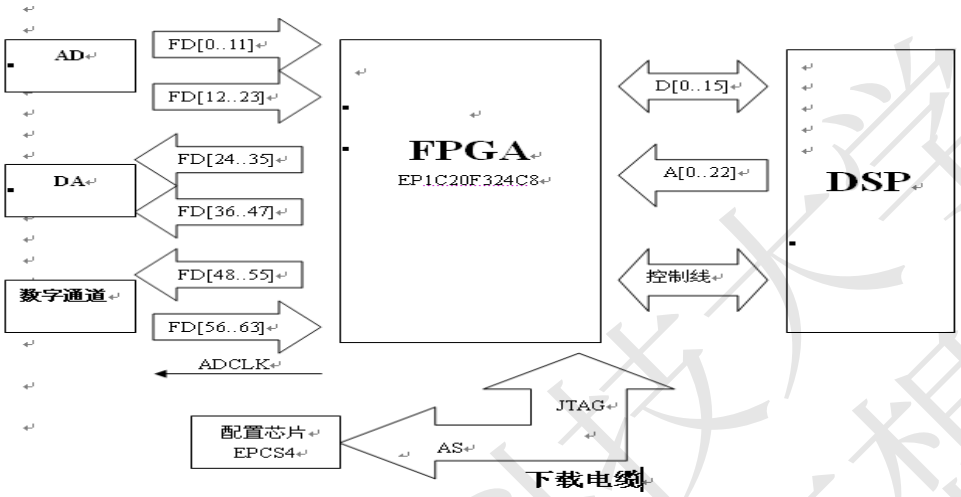


图 7.2 EPIC20F324C8 芯片系统连接图

(8) FPGA 集成开发环境

FPGA 软件是由 PLD/FPGA 芯片厂家提供，基本都可以完成所有的设计输入（原理图或 HDL)，仿真，综合，布线，下载等工作。器件的厂家不同，应用的软件开发环境有所不同。Xilinx 公司使用 XilinxFoundationF3.1 可编程器件开发工具 而 Altera 公司使用 MaxplusII 和 QuartusII 软件平台。这里我们只介绍厂商 Altera 公司的软件。

§7.3.3 QuartusII 举例、练习

下面通过几个实例，来学习使用 QuartusII（由教师指导学生完成）。

例子：

(1) 电路设计法设计：15 位 m 序列发生器电路及波形如图 7.3 图 7.4 所示

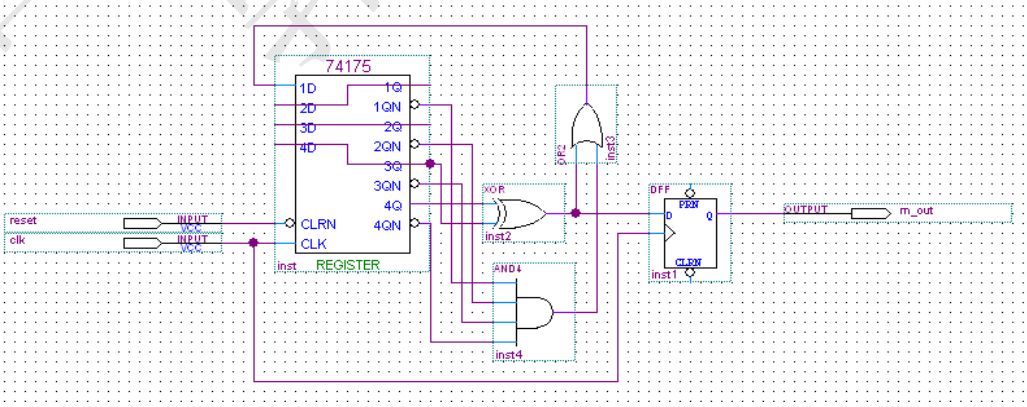


图 7.3 15 位 m 序列发生器

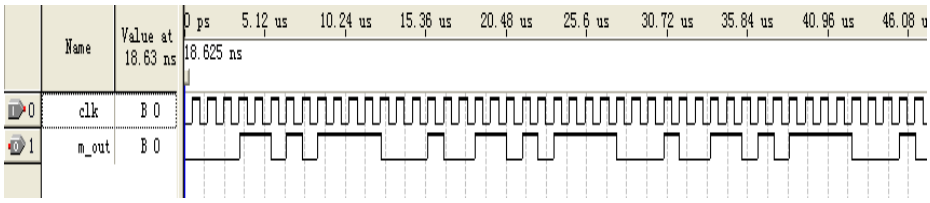


图 7.4　15 位 m 序列发生器波形

(2)　用 VHDL 实现四分频、五分频

四分频 VHDL 程序

```
-- 占空比 1：1 的通用分频模块
-- 修改 generic 中 div_size 的大小即可
library ieee;
use ieee.std_logic_1164.all;

entity jiclkdiv is
port(
  clkin : in std_logic;
  clkout : out std_logic);
end jiclkdiv;

architecture behavioral of jiclkdiv is
constant N:integer:=4;   --N 是奇数分频系数
signal coun:integer range 0 to N;
signal  up,down:std_logic;
begin
  process(clkin)
  begin
    if rising_edge(clkin) then
      if coun=N-1 then
        coun<=0;
        up<=not up;
      else
```

```
        coun<=coun+1;
      end if;
    end if;

    if falling_edge(clkin) then
      if coun=(N-1)/2 then
         down<=not down;
      end if;
    end if;
 end process;
  clkout<=up XOR down;
end behavioral;
```

波形如图 7.5 所示

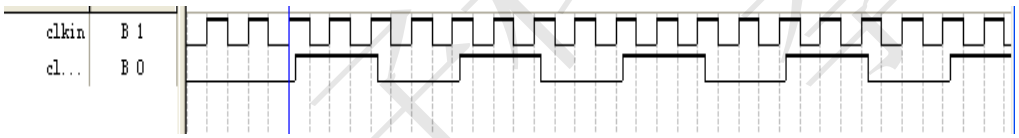


图 7.5　4 分频波形

5 分频程序:将 4 分频程序的分频系数 N 改为 5 即可，波形如图 7.6 所示

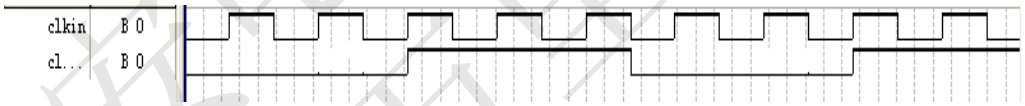


图 7.6　5 分频波形

(3)　正弦信号输出，其电路及波形如图 7.7 图 7.8 所示

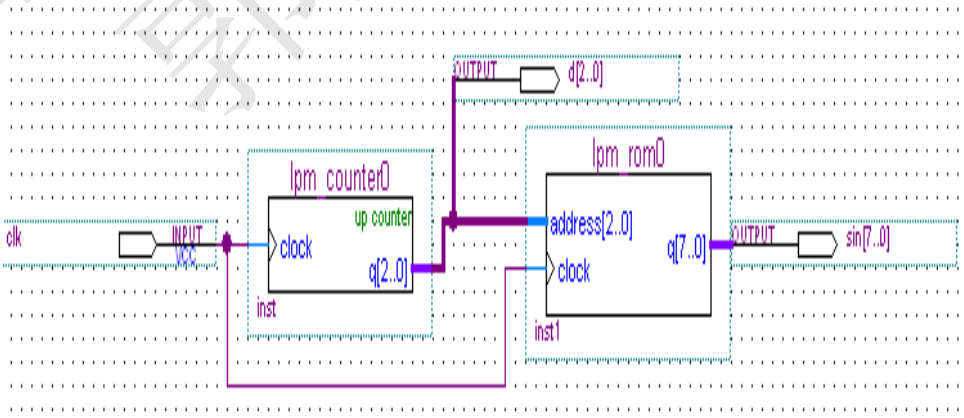


图 7.7　正弦信号输出电路

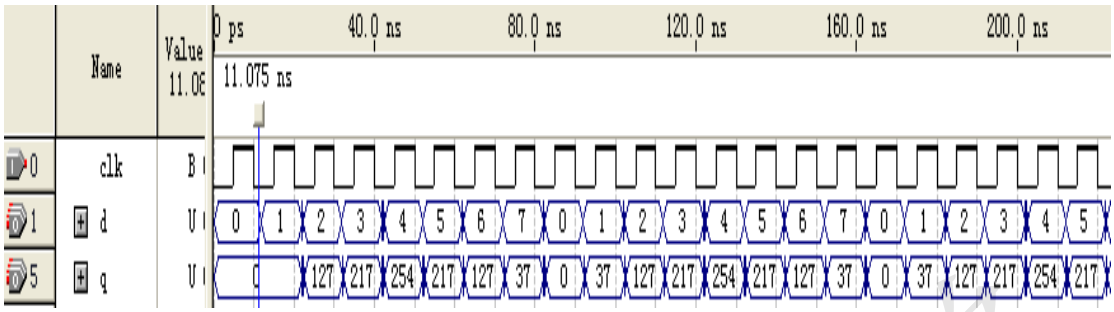


图 7.8　正弦信号输出波形

①　正弦波的一个周期由 8 个点组成，分别为 0，0.707，1，0.707，0，-0.707，-1，-0.707。

②　首先将这些数转换为正整数，每一个数分别加+1，为 1，1.707，2，1.707，1，0.293，0，0.293。再将这些数乘以 127。则有：127，217，254，217，127，37，0，37。将正弦波的值存入 ROM 中，然后按地址的顺序读出。

③　创建一个 3 位地址的计数器

④　建一个地址为 3 的 ROM

⑤　连接、编译、仿真

⑷　正弦波转变成方波，其电路及波形图如图 7.9 图 7.10 所示

以练习⑶正弦波输出电路作为输入端，将输出的正弦波转变成方波。使用比较器可实现之。首先要将练习⑶中的*.bsf、*.tdf、*.mif 复制到练习⑷中。

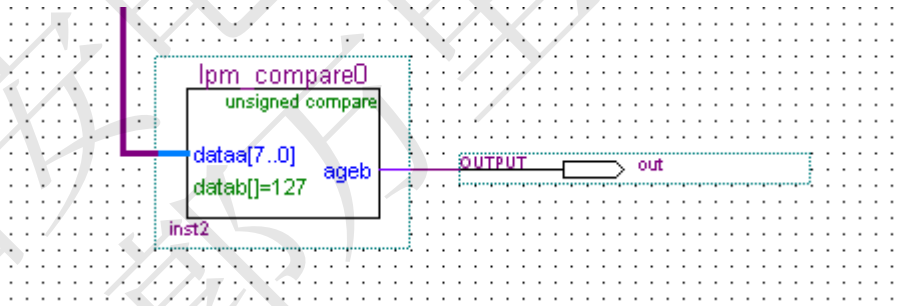


图 7.9　正弦信号转变成方波电路

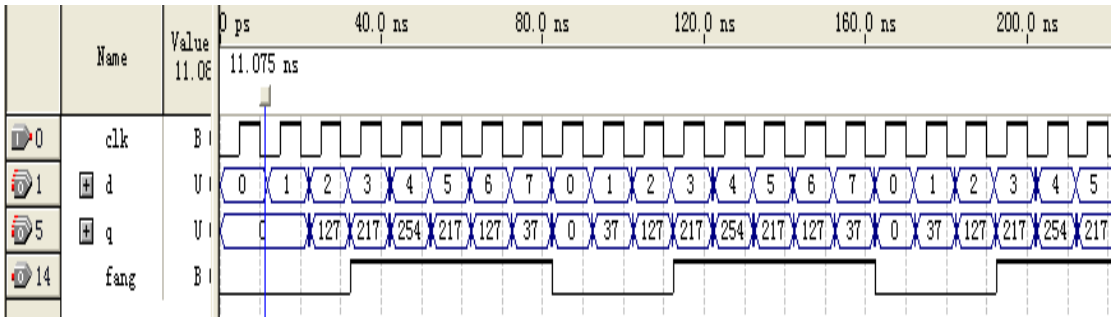


图 7.10　正弦信号转变成方波波形

& 7.4 汉明编码实现原理及框图

§7.4.1 分组码

分组码是一组固定长度的码组，可表示为（n ， k），通常它用于前向纠错。在分组码中，监督位被加到信息位之后，形成新的码。在编码时，k 个信息位被编为 n 位码组长度，而 n-k 个监督位的作用就是实现检错与纠错。当分组码的信息码元与监督码元之间的关系为线性关系时，这种分组码就称为线性分组码。

对于长度为 n 的二进制线性分组码，它有2^n种可能的码组2^n，从种码组中，可以选择 M= 2^k个码组（k<n）组成一种码。这样，一个 k 比特信息的线性分组码可以映射到一个长度为 n 码组上，该码组是从 M= 2^k个码组构成的码集中选出来的，这样剩下的码组就可以对这个分组码进行检错或纠错。

线性分组码是建立在代数群论基础之上的，各许用码的集合构成了代数学中的群，它们的主要性质如下：

① 任意两许用码之和（对于二进制码这个和的含义是模二和）仍为一许用码，也就是说，线性分组码具有封闭性；

② 码组间的最小码距等于非零码的最小码重。

奇偶监督码，就是一种最简单的线性分组码，由于只有一位监督位通常可以表示为（n,n-1）。在接收端解码时，实际上就是在计算：

$$S = b_{n-1} + b_{n-2} + \cdots + b_1 + b_0 \qquad (1)$$

其中，b_{n-1} b_{n-2}…b_1表示接收到的信息位，b_0表示接收到的监督位，若 S＝0，就认为无错；若 S＝1 就认为有错。式（1）被称为监督关系式，S 是校正子。由于校正子 S 的取值只有“0”和“1”两种状态，因此，它只能表示有错和无错这两种信息，而不能指出错码的位置。

设想如果监督位增加一位，即变成两位，则能增加一个类似于式（1）的监督关系式，计算出两个校正子S_1和S_2，S_1 S_2而共有 4 种组合：00，01，10，11，可以表示 4 种不同的信息。除了用 00 表示无错以外，其余 3 种状态就可用于指示 3 种不同的误码图样。

同理，由 r 个监督方程式计算得到的校正子有 r 位，可以用来指示 2^r -1 种误码图样。对于一位误码来说，就可以指示 2^r-1 个误码位置。对于码组长度为 n、信息码元为 k 位、监督码元为 r＝n － k 位的分组码（常记作（n，k）码），如果希望用 r 个监督位构造出 r 个监督关系式来指示一位错码的 n 种可能，则要求：

$$2^r-1\geq n \quad 或 \quad 2^r\geq k+r+1 \qquad (2)$$

下面通过一个例子来说明线性分组码是如何构造的。设分组码（n ， k）中 k = 4，为了能够纠正一位错误，由式（2）可以看到，要求 r ≥ 3，若取 r = 3，则 n = k+r = 7。因此，可以用 $a_6a_5a_4a_3a_2a_1a_0$ 表示这 7 个码元，用 S_3、S_2、S_1 表示利用三个监督方程，通过计算得到的校正子，并且假设 S_3、S_2、 S_1 三位校正字码组与误码位置的关系如表 1（当然，也可以规定成另一种对应关系，这并不影响讨论的一般性）：

由表中规定可已看到，仅当一错码位置在 a_2、 a_4、 a_5 或 a_6 时，校正子 S_1 为 1；否则 S_1 为 0。这就意味着 a_2、 a_4、 a_5 和 a_6 四个码元构成偶数监督关系：

$$S_1=a_6+a_5+a_4+a_2 \qquad (3a)$$ 同理，

$$S_2=a_6+a_5+a_3+a_1 \qquad (3b)$$

$$S_3=a_6+a_4+a_3+a_0 \qquad (3c)$$

表 1 校正字与误码位置

表 1　校正子与误码位置

S1 S2 S3	误码位置	S1 S2 S3	误码位置
001	a0	101	a4
010	a1	110	a5
100	a2	111	a6
011	a3	000	无错

在发送端编码时 a_6、 a_5、 a_4 和 a_3 是信息码元，它们的值取决于输入信号，因此是随机的。 a_2、 a_1 和 a_0 是监督码元，它们的取值由监督关系来确定，即监督位应使式（3）的

三个表达式中的S_3、 S_2和S_1的值为零（表示编成的码组中应无错码），这样式（3）的三个表达式可以表示成下面的方程组形式：

$$\begin{cases} a_6+a_5+a_4+a_2=0 \\ a_6+a_5+a_3+a_1=0 \\ a_6+a_4+a_3+a_0=0 \end{cases} \quad (4)$$

由上式经移项运算，接出监督位

$$\begin{cases} a_6+a_5+a_4=a_2 \\ a_6+a_5+a_3=a_1 \\ a_6+a_4+a_3=a_0 \end{cases} \quad (5)$$

根据上面两个线性关系，可以得到 16 个许用码组如表 2 所示：

表 2　　许用码组

信息位	监督位	信息位	监督位	信息位	监督位	信息位	监督位
a6a5a4a3	a2a1a0	a6a5a4a3	a2a1a0	a6a5a4a3	a2a1a0	a6a5a4a3	a2a1a0
0000	000	0100	110	1000	111	1100	001
0001	011	0101	101	1001	100	1101	010
0010	101	0100	011	1010	010	1100	100
0011	110	0111	000	1011	001	1111	111

接收端收到每个码组后，计算出S_3、S_2和S_1，如不全为 0，则可按表 1 确定误码的位置，然后予以纠正。例如，接收码组为 0000011，可算出S_3 S_2 S_1＝011，由表 1 可知在 a_3 位置上有一误码。

不难看出，上述（7，4）码的最小码距$d_{\min}=3$，因此，它能纠正一个误码或检测两个误码。如超出纠错能力，则反而会因“乱纠”而增加新的误码。

监督矩阵 H 和生成矩阵 G

式（4）所述（7，4）码的三个监督方程式可以重新改写为如下形式：

$$\begin{cases} 1\cdot a_6+1\cdot a_5+1\cdot a_4+0\cdot a_3+1\cdot a_2+0\cdot a_1+0\cdot a_0=0 \\ 1\cdot a_6+1\cdot a_5+0\cdot a_4+1\cdot a_3+0\cdot a_2+1\cdot a_1+0\cdot a_0=0 \\ 1\cdot a_6+0\cdot a_5+1\cdot a_4+1\cdot a_3+0\cdot a_2+0\cdot a_1+1\cdot a_0=0 \end{cases} \quad (6)$$

对于式（6）可以用矩阵形式来表示：

$$\begin{bmatrix}1&1&1&0&1&0&0\\1&1&0&1&0&1&0\\1&0&1&1&0&0&1\end{bmatrix}\cdot[a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a_0]^T=\begin{bmatrix}0\\0\\0\end{bmatrix}\quad(7)$$

上式可以记作：$HA^T=\theta^T$ 或 $AH^T=\theta^T$，其中

$$\mathbf{H}=\begin{bmatrix}1&1&1&0&1&0&0\\1&1&0&1&0&1&0\\1&0&1&1&0&0&1\end{bmatrix}=[\mathbf{P}\quad\mathbf{I}_r]\quad(8a)$$

$$\mathbf{A}=[a_6\quad a_5\quad a_4\quad a_3\quad a_2\quad a_1\quad a_0]\quad(8b)$$

$$\theta=[0\quad 0\quad 0]\quad(8c)$$

通常 H 称为监督矩阵，A 称为信道编码得到的码字。在这个例子中 H 为 r×n 阶矩阵，P 为 r×k 阶矩阵，Ir 为 r×r 阶单位矩阵，具有这种特性的 H 矩阵称为典型监督矩阵，这是一种较为简单的信道编译码方式。典型形式的监督矩阵各行是线性无关的，非典型形式的监督矩阵可以经过行或列的运算化为典型形式。

对于式（5）也可以用矩阵形式来表示：

$$\begin{bmatrix}a_2\\a_1\\a_0\end{bmatrix}=\begin{bmatrix}1&1&1&0\\1&1&0&1\\1&0&1&1\end{bmatrix}\cdot\begin{bmatrix}a_6\\a_5\\a_4\\a_3\end{bmatrix}$$

或者

$$[a_2\quad a_1\quad a_0]=[a_6\quad a_5\quad a_4\quad a_3]\cdot\begin{bmatrix}1&1&1\\1&1&0\\1&0&1\\0&1&1\end{bmatrix}=[a_6\quad a_5\quad a_4\quad a_3]\cdot Q\quad(9)$$

比较式（8a）和式（9）可以看到 $Q=P^T$，如果在 Q 矩阵的左边在加上一个 k×k 的单位矩阵，就形成了一个新矩阵 G:

$$\mathbf{G}=[\mathbf{I}_k\quad\mathbf{Q}]=\begin{bmatrix}1&0&0&0&1&1&1\\0&1&0&0&1&1&0\\0&0&1&0&1&0&1\\0&0&0&1&0&1&1\end{bmatrix}\quad(10)$$

这里 G 称为生成矩阵，利用它可以产生整个码组

$$\mathbf{A}=\mathbf{M}\cdot\mathbf{G}=[a_6\quad a_5\quad a_4\quad a_3]\cdot\mathbf{G}\quad(11)$$

由式（10）表示的生成矩阵形式称为典型生成矩阵，利用式（11）产生的分组码必为系统码，也就是信息码元保持不变，监督码元附加在其后。

校验子 S

发送端信息码元 M 利用式（11），实现信道编码，产生线性分组码 A；在传输过程中有可能出现误码，设接收到的码组为 B。则收发码组之差为：

$$\begin{aligned}\mathbf{B}-\mathbf{A}&=\begin{bmatrix}b_{n-1} & b_{n-2} & \cdots & b_0\end{bmatrix}-\begin{bmatrix}a_{n-1} & a_{n-2} & \cdots & a_0\end{bmatrix}\\&=\mathbf{E}=\begin{bmatrix}e_{n-1} & e_{n-2} & \cdots & e_0\end{bmatrix}\end{aligned} \tag{12}$$

这里 $e_i=\begin{cases}0 & b_i=a_i\\1 & b_i\neq a_i\end{cases}$，$e_i=1$，表示 i 位有错，$e_i=0$，表示 i 位无错。基于这样的原则接收端利用接收到的码组 B 计算校正子：

$$S=BH^T=(A+E)H^T=AH^T+EH^T=EH^T \tag{13}$$

因此，校正子仅与 E 有关，即错误图样与校正子之间有确定的关系。

对于上述（7，4）码，校正子 S 与错误图样的对应关系可由式（13）求得，其计算结果见表 3 所示。在接收端的译码器中有专门的校正子计算电路，从而实现检错和纠错。

表 3（7，4）码校正子与错误图样的对应关系

表 3　（7，4）码校正子与错误图样的对应关系

序号	错误码位	E	S
		e6 e5 e4 e3 e2 e1 e0	S3 S2 S1
0	/	0 0 0 0 0 0 0	0 0 0
1	b0	0 0 0 0 0 0 1	0 0 1
2	b1	0 0 0 0 0 1 0	0 1 0
3	b2	0 0 0 0 1 0 0	1 0 0
4	b3	0 0 0 1 0 0 0	0 1 1
5	b4	0 0 1 0 0 0 0	1 0 1
6	b5	0 1 0 0 0 0 0	1 1 0
7	b6	1 0 0 0 0 0 0	1 1 1

§7.4.2　汉明码

汉明码是一种能够纠正单个错误的线性分组码。它有以下特点：

（1）最小码距$d_{min}=3$，可以纠正一位错误；

（2）码长 n 与监督元个数 r 之间满足关系式：$n=2^r-1$。

如果要产生一个系统汉明码，可以将矩阵 H 转换成典型形式的监督矩阵，进一步利用 Q = PT 的关系，得到相应的生成矩阵 G。通常二进制汉明码可以表示为：

$$(n,k)=(2^r-1\quad,\quad 2^r-1-r) \qquad (14)$$

根据上述汉明码定义可以看到，（7，4）线性分组码实际上就是一个汉明码，它满足了汉明码的两个特点。

实验中要实现的汉明编译码总模块如 7.11 所示

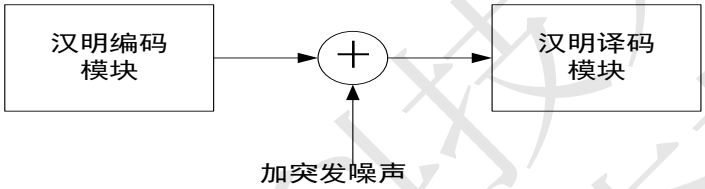


图 7.11　汉明编译码总模块

§7.4.3 汉明码编码器的实现

汉明码编码器实现原理框图如 7.12 所示

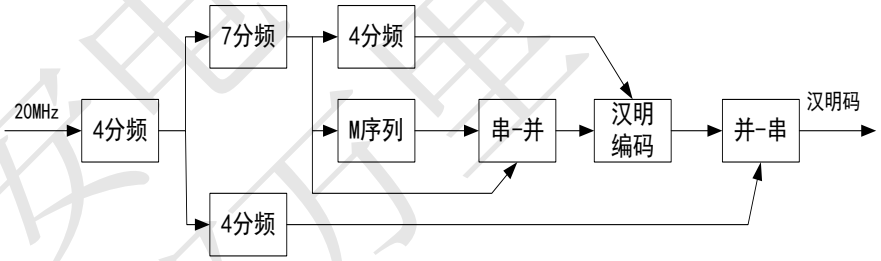


图 7.12　汉明码编码器实现原理框图

(1)　时钟：时钟信号由实验板提供 20MHz 时钟（仿真时也选用 20MHz 时钟）经 4 分频模块得到 5MHz，电路及波形如 7.13 图 7.14 所示。

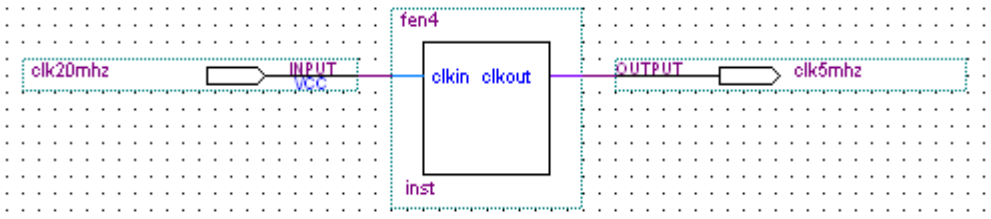


图 7.13　时钟产生电路

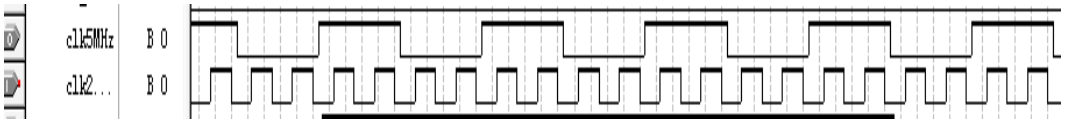


图 7.14　时钟波形

(2)　信源：产生周期为 15 的 M 序列作为消息的输入，电路及波形如图 7.15 图 7.16 所示。信源时钟为 5MHz 经 7 分频模块得到（5MHz/7）。

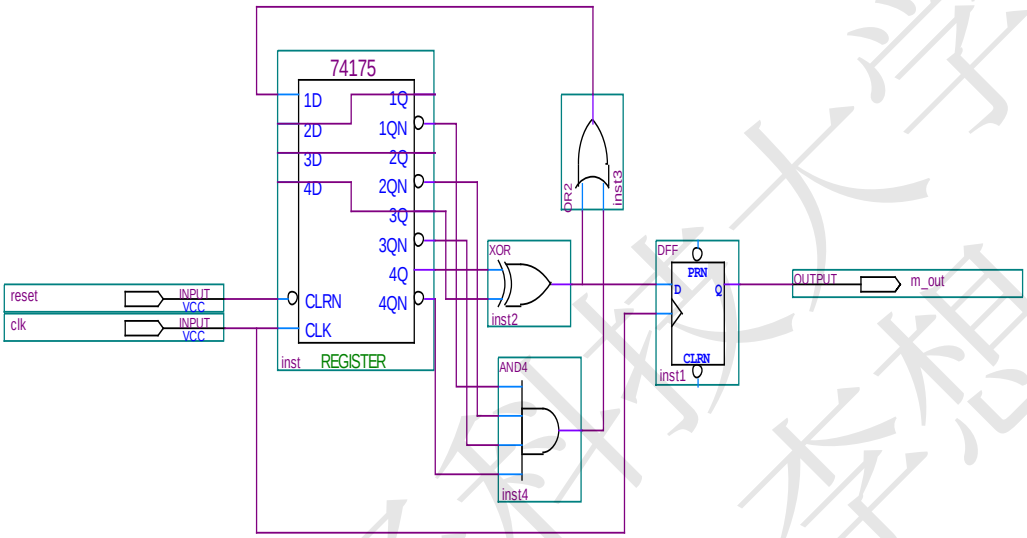


图 7.15　m 序列发生器电路

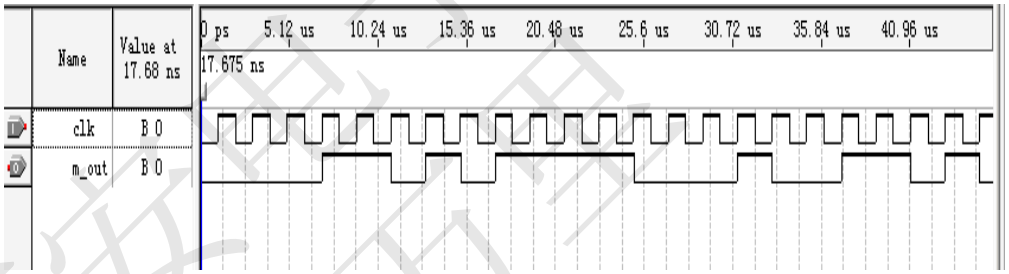


图 7.16　m 序列波形

(3)　串并转换模块：将信源 m 序列串行输出变换成 4 位并行码（verilog 编写），时钟是 5/7MHz。程序及波形如图 7.17。

```
module bm_ipo11(mxulie,b,clk5000);
    input clk5000;
    input mxulie;
    output [3:0] b;
    reg [3:0] b;
    reg [3:0] temp;
    integer counter = 0;
```

```
always @(posedge clk5000)
    begin
     if(counter ==4)
        begin
            b <= temp;
            counter = 0;
        end
        temp[3-counter] <= mxulie;
        counter = counter + 1;
     end
endmodule
```

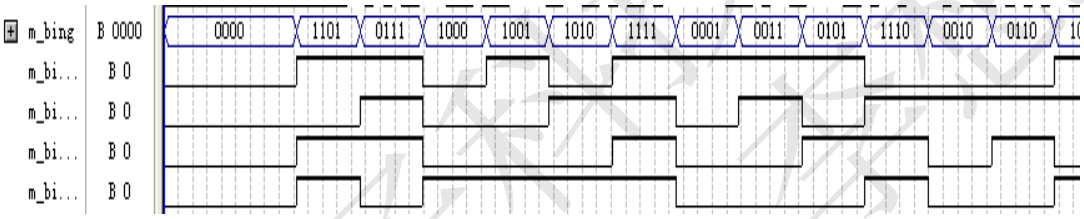


图 7.17　串并转换波形

⑷ 汉明编码：设计一种（7，4）汉明码，即 4 位信息码，3 位监督码。汉明编码器实现的具体电路需要自己去设计。下面给出一个具体的例子仅供参考。下面的a_6、a_5、a_4、a_3为信息码元，a_2、a_1、a_0为监督码元。它们各关系如下：

$$a_2 = a_6 + a_5 + a_4$$

$$a_1 = a_6 + a_5 + a_3$$

$$a_0 = a_6 + a_4 + a_3$$

汉明编码电路（VHDL 编写），时钟是 5/28MHz。汉明编码原理图、程序及波形如图 2.18 图 7.19 所示

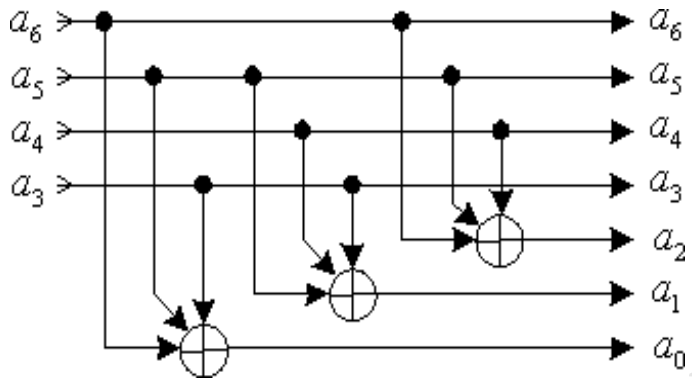


图 7.18　汉明编码原理图

汉明编码程序：

```
library ieee;
 use ieee.std_logic_1164.all;
 use ieee.std_logic_unsigned.all;
 entity hmbm is
       port (b: in std_logic_vector(3 downto 0);
             clk4:in std_logic;
             hm_out: out std_logic_vector(6 downto 0));
 end hmbm;
   architecture myarch of hmbm is
 begin
 process(clk4,b)
   begin
    if rising_edge(clk4) then
      hm_out(0)<=b(3) xor b(1) xor b(0);
     hm_out(1)<=b(3) xor b(2) xor b(0);
      hm_out(2)<=b(3) xor b(2) xor b(1);
       hm_out(3)<=b(0) ;
       hm_out(4)<=b(1);
       hm_out(5)<=b(2);
       hm_out(6)<=b(3);
     end if;
 end process;
```

end myarch;

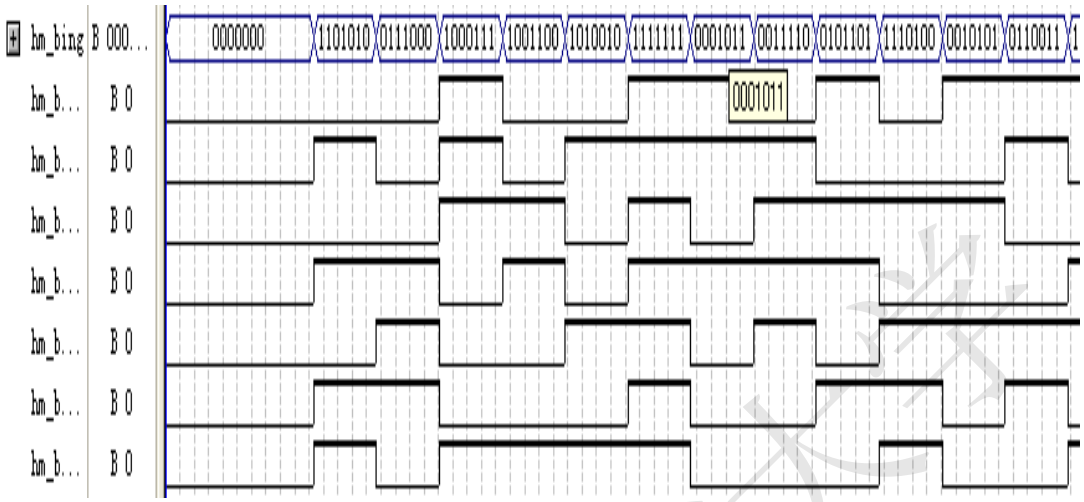


图 7.19　汉明编码波形

(5)　并串转换：经汉明编码器得到 7 位并行码，需要将汉明并行码砖换成串码，时钟是 5MHz/4，1.25MHz。发往信道。并串转换电路及波形如图 7.20 图 7.21 所示。

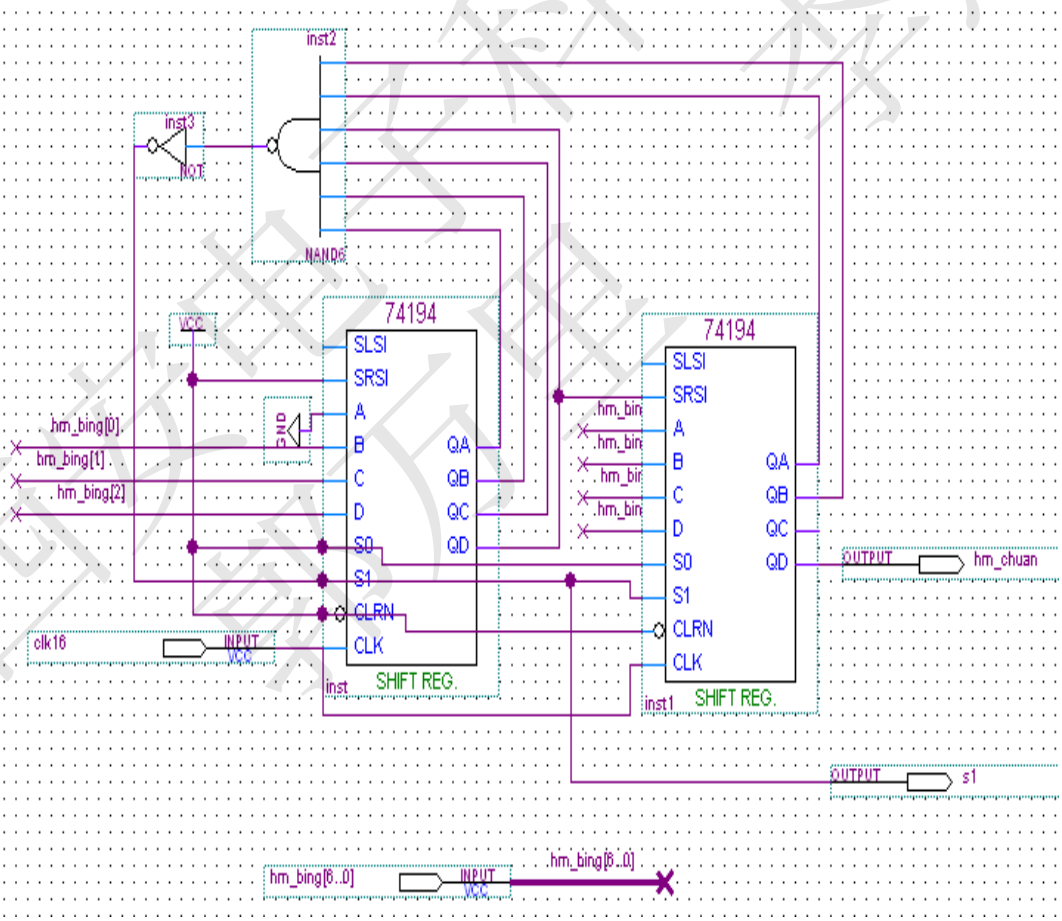


图 7.20　并串转换电路

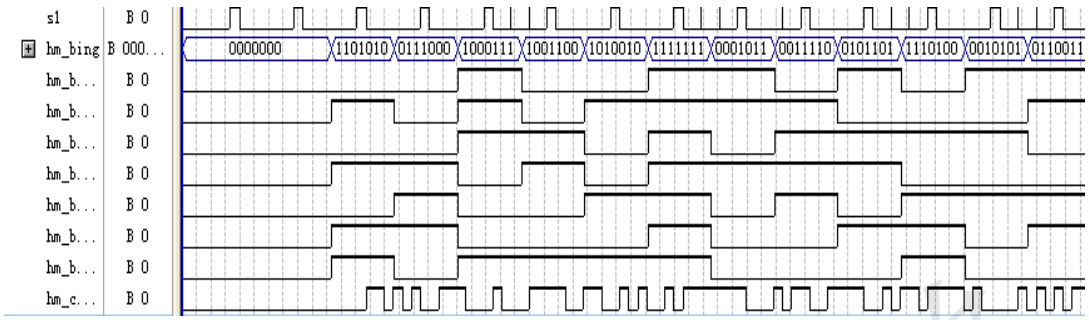


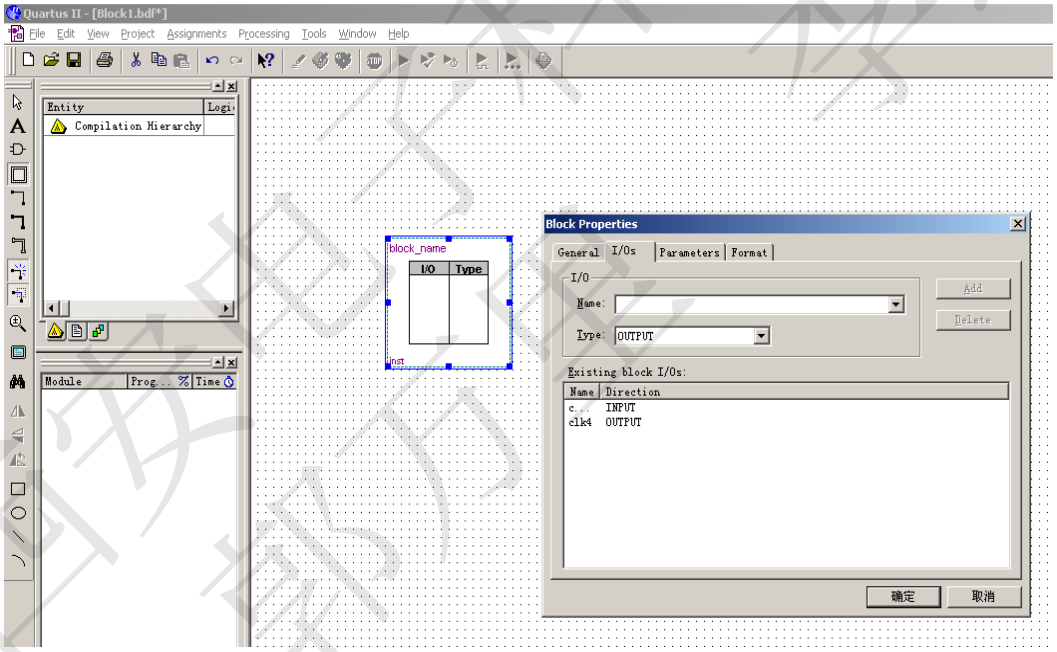
图 7.21　并串转换波形

设计好的汉明编码器先仿真得到正确的结果，然后需要下载到实验板中用示波器观察编码信号。

汉明编码器由上而下的设计方法：先建模块，再填入数据。

(1)　用模块编辑器产生一个新的文件

选择File→New→Block Diagram/Schematic File项产生一个新的图表模块/原理图



文件。然后点击OK按钮，并保存文件，如图7.22所示。

图7.22　建立图表文件

(2)　用模块编辑器设计模块

★ 从工具兰中产生模块，工具栏包括很多工具，你可以根据需要选择。例如点击工具栏中的模块图表来画一个模块，鼠标托动确定模块的大小，点击鼠标右键，在弹出图2.23的对话框中选择属性。在图7.23中，其中：

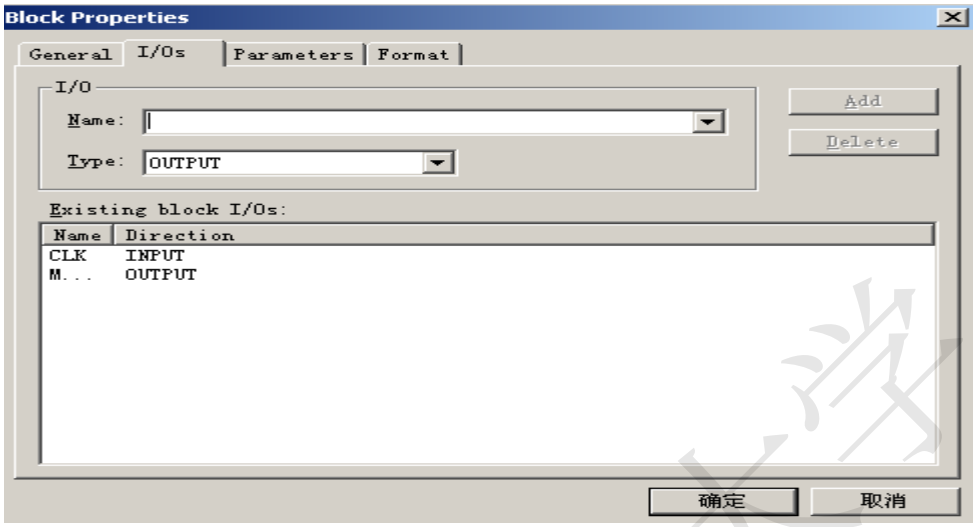


图7.23　定义输入输出端

★　在模块属性对话框中点击General，输入模块名称，如mxulie。

★　在模块属性对话框中点击I/Os，可键入输入输出端口名并选择相应的端口类型，然后点击Add按钮添加。如在电路M序列发生器中，输入端口名为CLK，在Type栏中选择INPUT，同理，输出端口名为MXULIE，在Type栏中选择OUTPUT。

★　点击确定按钮，用鼠标右键点击模块，选择AutoFit，模块中的数据都可以看见，如图7.24所示。

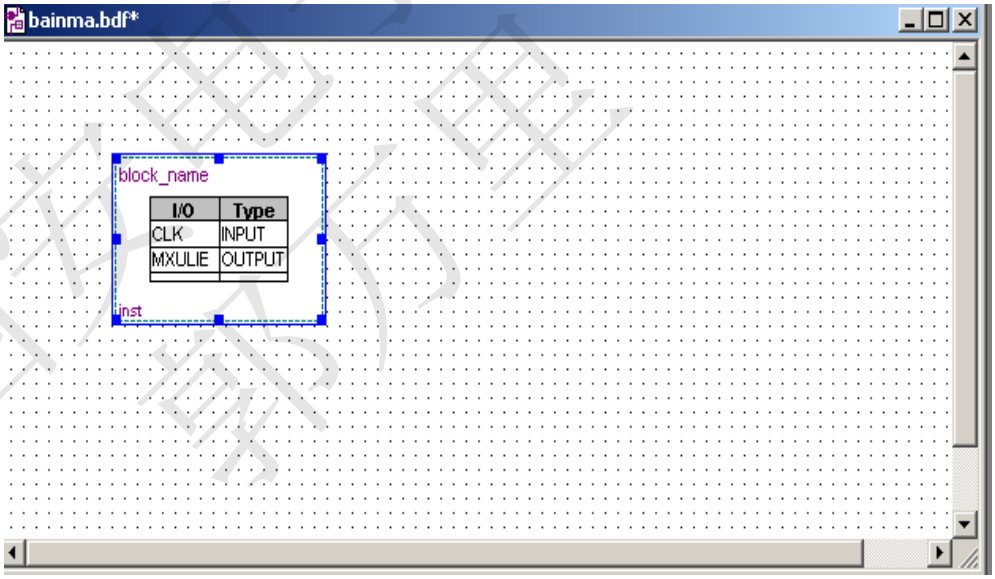


图7.24　确定图表模块

建好所有模块之后，建立工程文件。然后给每一个模块添加实际内容。其操作步骤是：在确定模块上点击鼠标右键点，选择create design file from selected block选项，在弹出图7.25的对话框

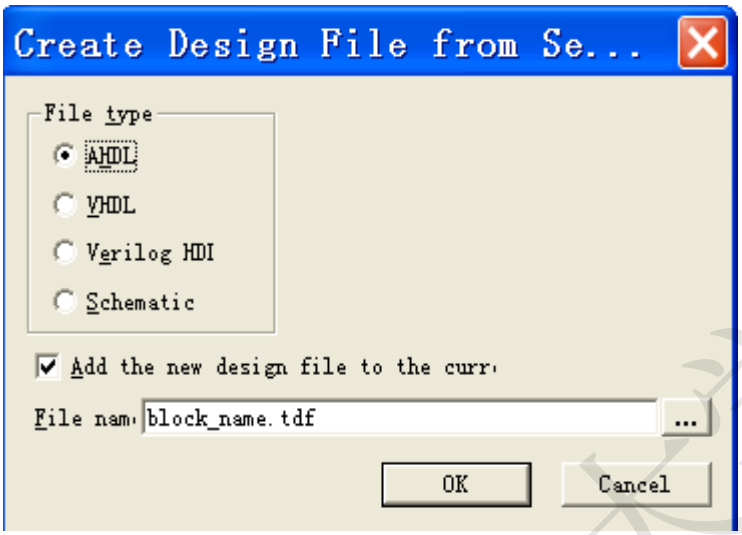
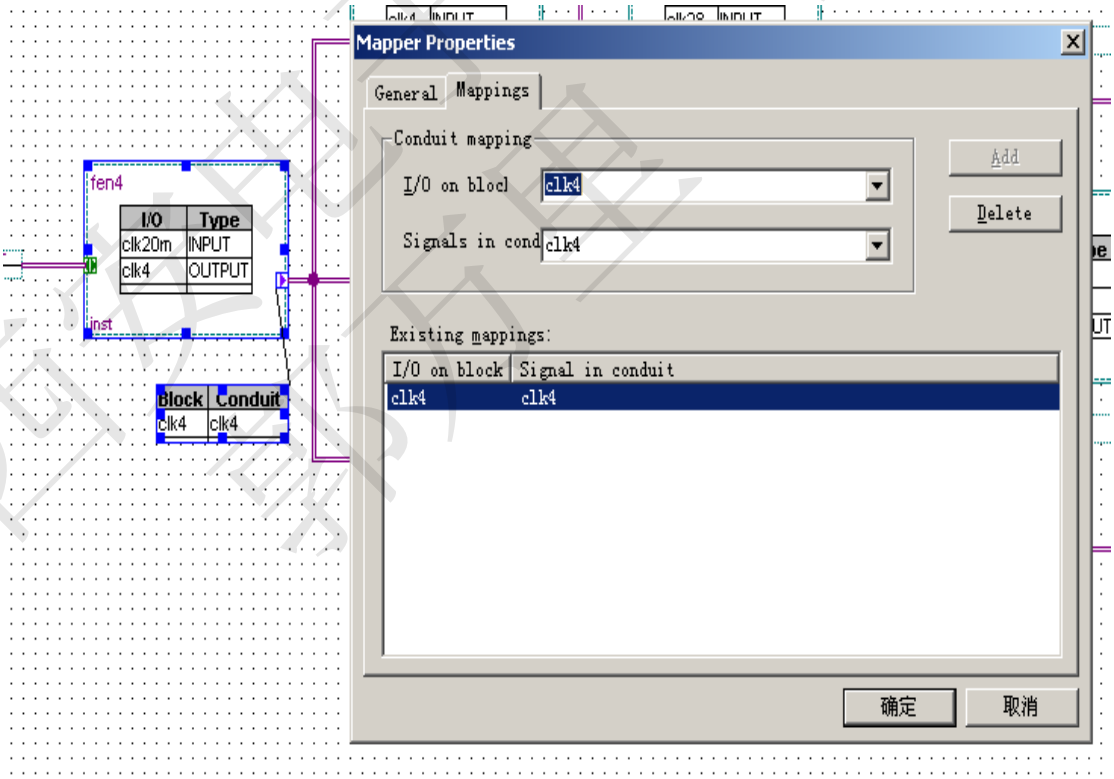


图7.25 选则设计文件类型

框中选择你需要的输入形式，然后点击 ok。这时就可以输入程序或电路图了。数据连接线为空的管道线。

（3） 指定节点

鼠标在模块处点击右键，点击 mapper properties 选择节点性质：输入、输出、BIDIR。



然后点击 mappings 指定节点名，如下图 7.26 所示

图 7.26 制定节点属性

其它编译仿真的过程可以参考由下而上的设计方法，在这里就不再重复。

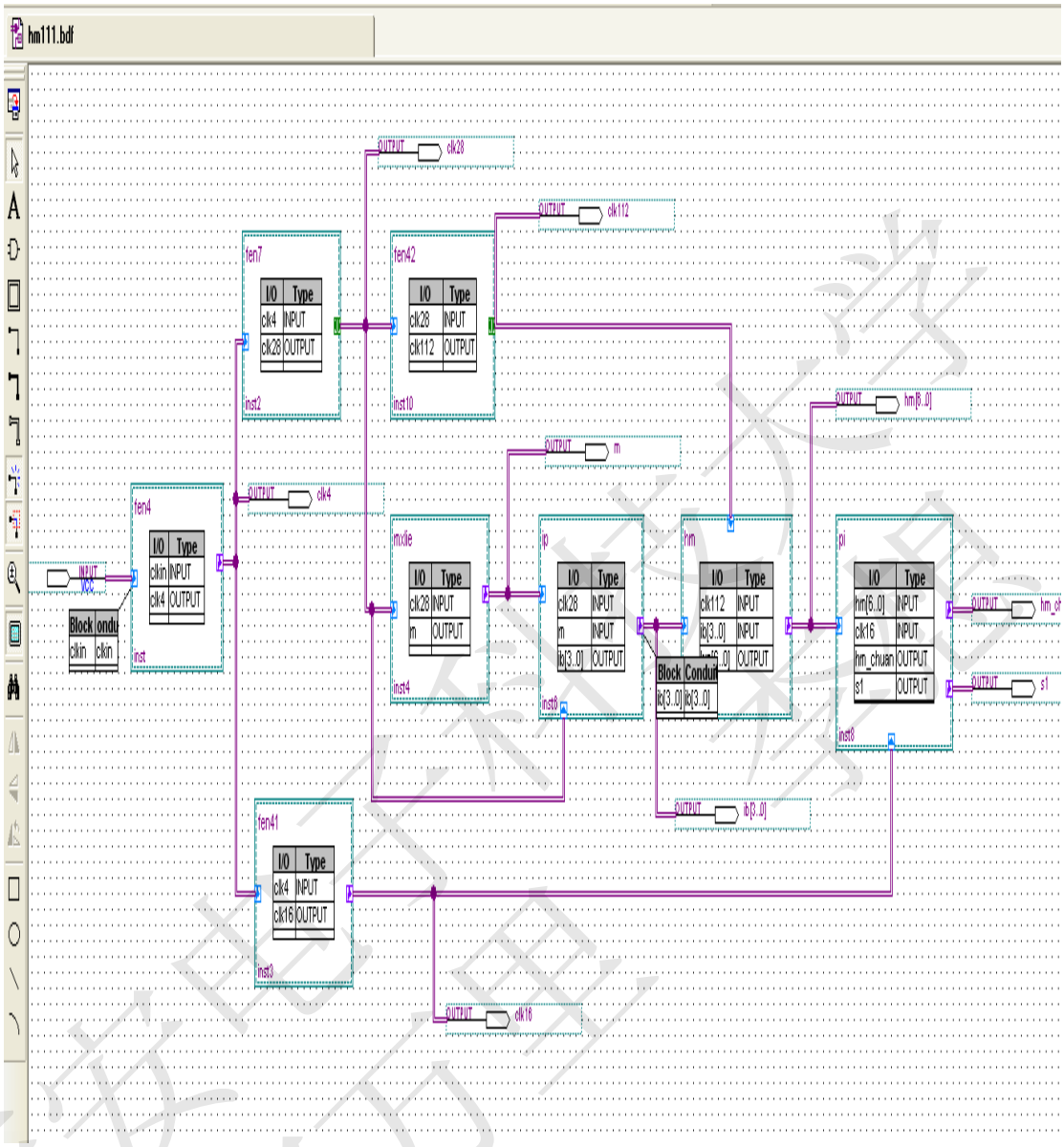


图 7.27　汉明编码模块

§7.4.4 汉明译码实现原理及框图

在设计汉明编码之前，将汉明编码封装成模块如图 7.28 所示。

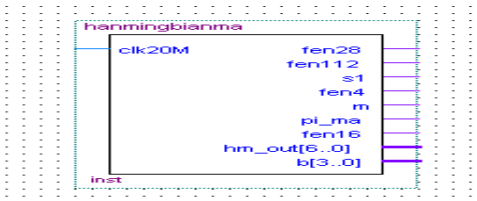


图 7.28　封转后的汉明编码模块

(1)　突发噪声产生模块

突发噪声+汉明编码电路及波形如图 7.29 图 7.30 图 7.31 图 7.32 所示

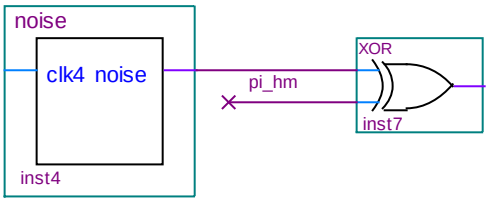


图 7.29　突发噪声+汉明编码

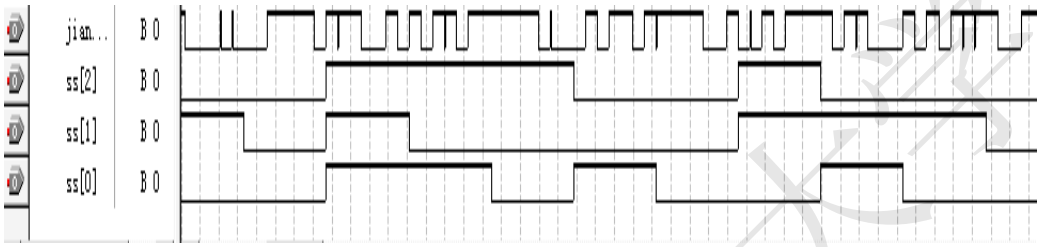


图 7.30 突发噪声+汉明编码波形

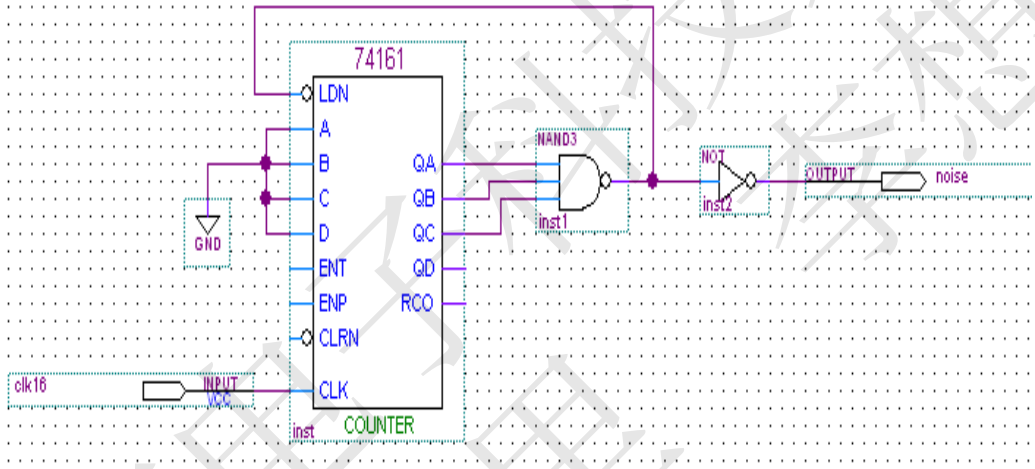


图 7.31　突发噪声电路

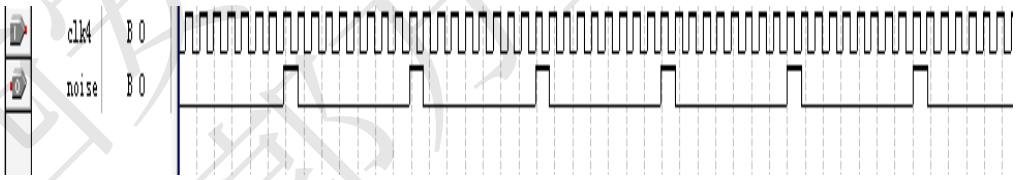


图 7.32　突发噪声波形

(2)　汉明码译码模块如图 7.33 所示

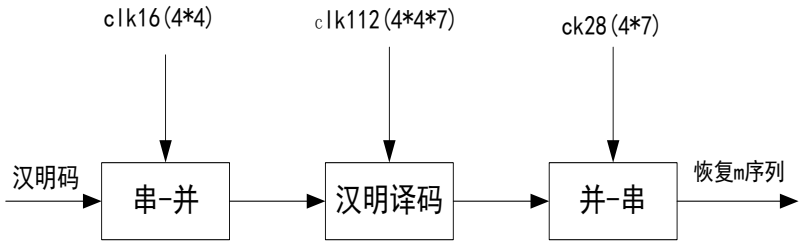


图 7.33　汉明译码模块

① 串并转换：将 7 位串并行码转换成 7 位并行码，程序及波形图如图 7.34。

串并转换程序：

```
module ym_ipo11(pi_hm,clk,cb_out);
   input clk;
   input pi_hm;
   output [6:0] cb_out;
   reg [6:0] cb_out;
   reg [6:0] temp;

   integer counter = 0;
   always @(posedge clk)
       begin
        if(counter ==7)
           begin
               cb_out <= temp;
               counter = 0;
           end
           temp[6-counter] <= pi_hm;
           counter = counter + 1;
        end
   endmodule
```

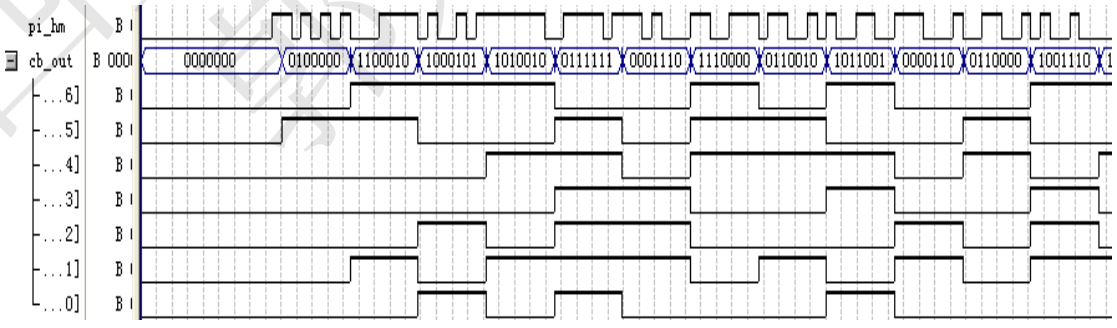


图 7.34 串并转换波形

② 汉明译码：电路及波形图如图 7.35 图 7.36 所示。

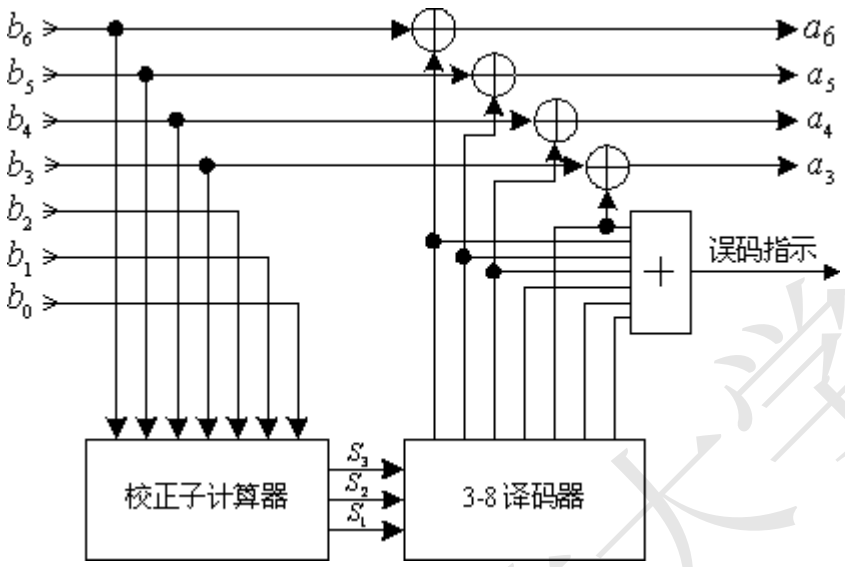


图 7.35　汉明译码模块

校正字产生程序：

校正字 ss 与汉明码有以下关系：

Ss2=cb_out6⊕cb_out5⊕cb_out4⊕cb_out2;

Ss1=cb_out6⊕cb_out5⊕cb_out3⊕cb_out1;

Ss0=cb_out6⊕cb_out4⊕cb_out3⊕cb_out0;

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity ss is
     port (cb_out: in std_logic_vector(6 downto 0);
           clk:in std_logic;
           ss: out std_logic_vector(2 downto 0));
end ss;


  architecture myarch of ss is
  begin
      process(clk,cb_out)
      begin
           if rising_edge(clk) then
```

```
            ss(2)<=cb_out(6)  xor cb_out(5) xor cb_out(4) xor cb_out(2);
            ss(1)<=cb_out(6)  xor cb_out(5) xor cb_out(3) xor cb_out(1);
            ss(0)<=cb_out(6)  xor cb_out(4) xor cb_out(3) xor cb_out(0);
        end if;
    end process;
end myarch;
```

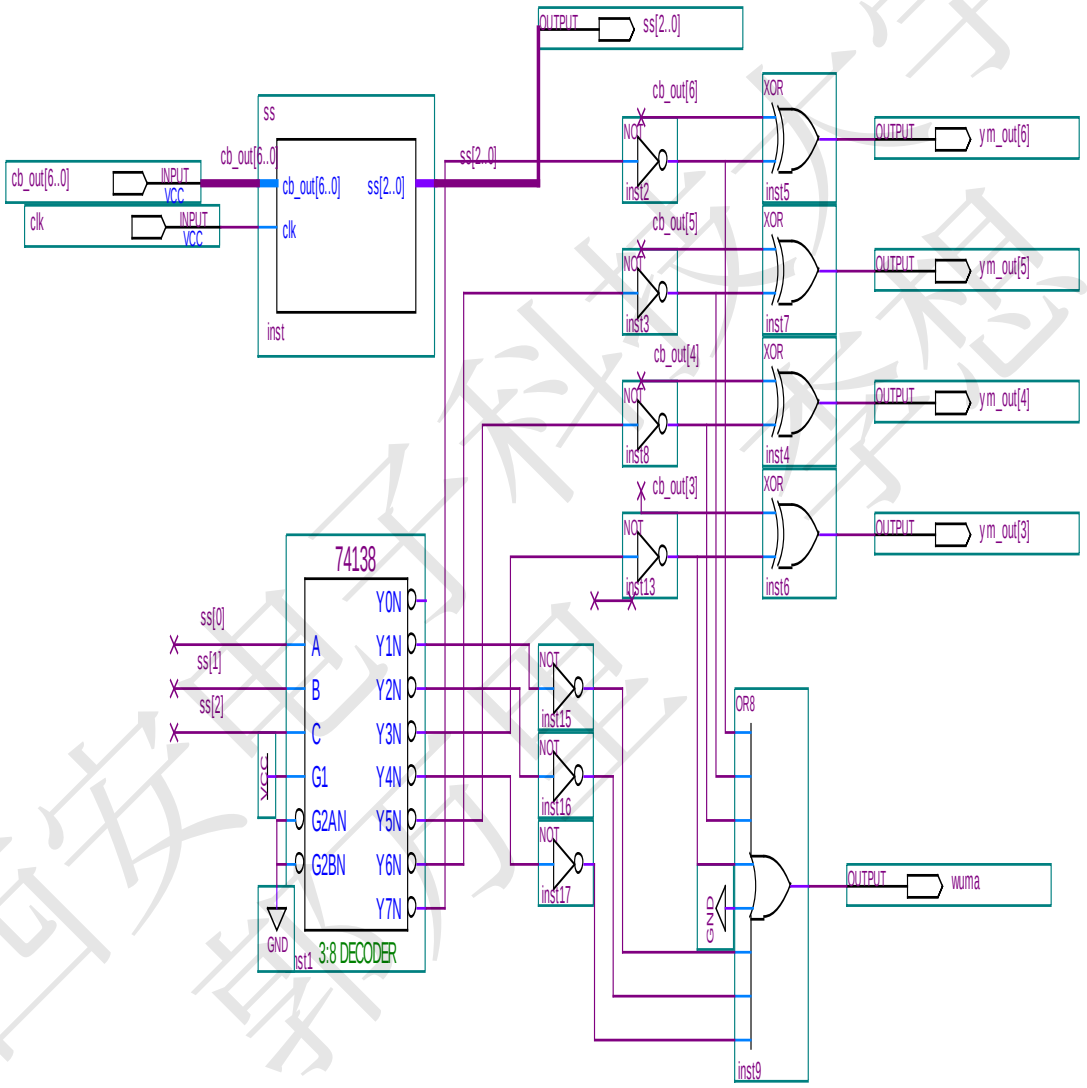


图 7.36　汉明译码电路

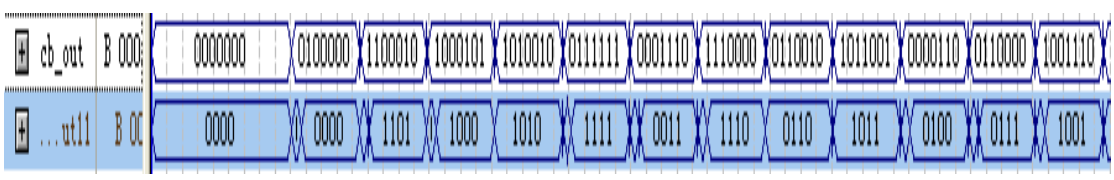


图 7.37　汉明译码波形

③　并串转换，电路及波形图如图 7.38 图 7.39 所示。

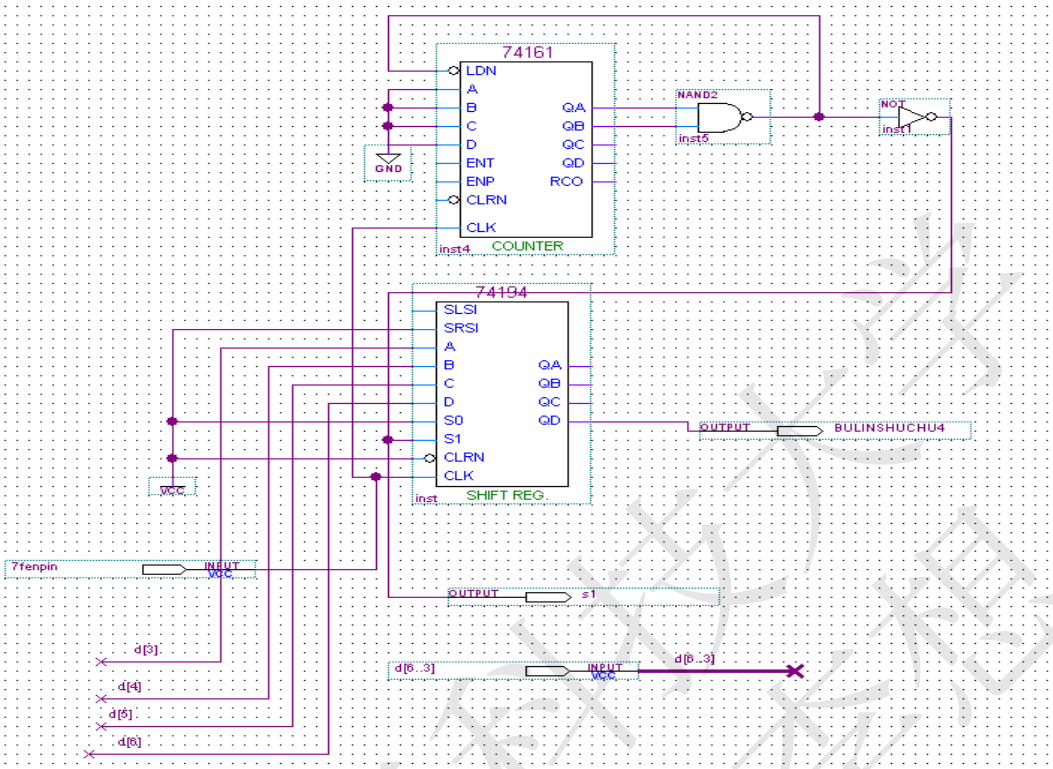


图 7.38　并串转换电路

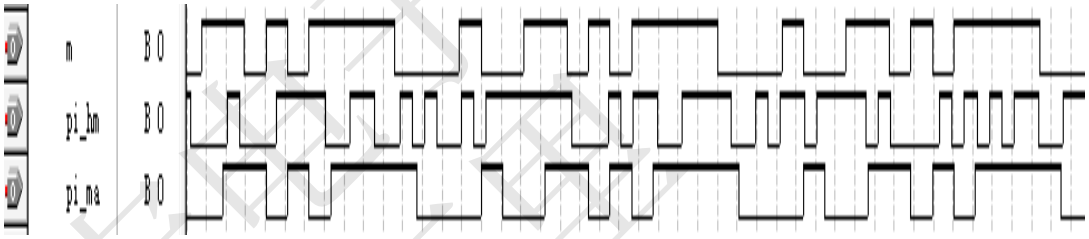


图 7.39　并串转换（恢复序列）波形

到此汉明编译码就设计完了，但是设计中存在着一个致命性的问题：信息码元起点识别没有设计，这样会带来致命性的错误。错误接收汉明编码而导致译码出错。如何解决这个问题，希望大家去思考，想出一个解决方案来。

§7.4.5 测试

① 设计好的译码器和编码器连接起来先仿真得到正确结果，最终将编译码器下载到实验板中，在示波器上观察编译码器的信号。

② 按系统方框图，将系统分成两部分：编码模块和译码模块。每部分都打包成模块，编译码模块又都有它自己的下层模块，最终完成汉明编译码器。

③ 将程序下载到实验板中调式的方法：可参考第八章 Quartus Ⅱ开发软件部分。

第八章 编码理论综合实验选题

以下所有课题仿真，数字电路都基于 FPGA，并使用 QuartusⅡ作为设计工具设计仿真、实现。由学生独立完成设计与仿真。所有模块用电路输入法或 VHDL/verilog 语言程序实现（建议复杂的算法用 VHDL/verilog 语言实现而简单电路用电路输入法实现）。在文中所有涉及到的系统框图及电路都仅作参考，我们不希望由于提供了参考电路而束缚了同学的思维和创新能力的发挥。

& 8.1（7，3）循环编译码器设计与实现

§8.1.1 实验目的

⑴ 了解 FPGA 软硬件知识，学习利用 QuartusII 实现电路的方法，能初步掌握之。

⑵ 了解循环码的生成多项式 g(x)与编、译码器之间的关系，码距与纠、检错能力之间的关系，掌握循环码的编码及译码方法。

⑶ 提高将编码理论应用到实践中的能力。

§8.1.2 实验仪器

1、512MHz 以上内存微计算机。

2、20MHz 双踪示波器、信号源。

3、QuartusⅡ语言环境。

4、fpga 实验板、若干导线。

§8.1.3 实现原理及框图

循环码是线性分组码的一个重要子集，是目前研究得最成熟的一类码。它有许多特殊的代数性质，这些性质有助于按所要求的纠错能力系统地构造这类码，且易于实现。同时循环码的性能也较好，具有较强的检错和纠错能力。

循环码的特点就是码字的循环特性，所谓循环特性是指：循环码中任意位许用码组经过循环移位后，所得到的码组仍然是许用码组。若（a_{n-1} a_{n-2} … a_1 a_0）为一循环码组，则（a_{n-2} a_{n-3}…a_0 a_{n-1}）、（a_{n-3} a_{n-4}…a_{n-1} a_{n-2}）、……还是许用码组。也就是说，不论是左移还是右移，也不论移多少位，仍然是许用的循环码组。下表给出了一种（7，3）循环码的全部码字。由此表可以直观地看出这种码的循环特性。例如，表 1 中的第 2 码字向右移一位，即得到第 5 码字；第 6 码字组向右移一位，即得到第 3 码字。

表 1　一种（7，3）循环码的全部码字

序号	码　字		序号	码　字	
	信息位	监督位		信息位	监督位
	a6 a5 a4	a3 a2 a1 a0		a6 a5 a4	a3 a2 a1 a0
1	0 0 0	0 0 0 0	5	1 0 0	1 0 1 1
2	0 0 1	0 1 1 1	6	1 0 1	1 1 0 0
3	0 1 0	1 1 1 0	7	1 1 0	0 1 0 1
4	0 1 1	1 0 0 1	8	1 1 1	0 0 1 0

例如，表 1 中的第 7 码字可以表示为：

$$A_7(x)=1\cdot x^6+1\cdot x^5+0\cdot x^4+0\cdot x^3+1\cdot x^2+0\cdot x+1$$
$$=x^6+x^5+x^2+1$$

一、循环编码器的工作原理：

在编码时，首先需要根据给定循环码的参数确定生成多项式 $g(x)$，也就是从x^n+1的因子中选一个（$n-k$）次多项式作为 $g(x)$；然后，利用循环码的编码特点，即所有循环码多项式 $A(x)$都可以被 $g(x)$整除，来定义生成多项式 $g(x)$。

根据上述原理可以得到一个较简单的系统循环编码方法：设要产生（n, k）循环码，$m(x)$表示信息多项式，则其次数必小于 k，而 $x^{n-k}\cdot m(x)$的次数必小于 n，用 $x^{n-k}\cdot m(x)$除以 $g(x)$，可得余数 $r(x)$，$r(x)$的次数必小于（$n-k$），将 $r(x)$加到信息位后作监督位，就得到了系统循环码。下面就将以上各步处理加以解释。

（1）用 x^{n-k}（监督码的最高位）乘 $m(x)$（信息码）。这一运算实际上是把信息码后附加上（$n-k$）个“0”。例如，信息码为 110，它相当于 $m(x)=x^2+x$。当 $n-k=7-3=4$ 时，$x^{n-k}\cdot m(x)=x^6+x^5$，它相当于 1100000。而希望的到得系统循环码多项式应当是 $A(x)=x^{n-k}\cdot m(x)+r(x)$。

（2）求 $r(x)$。由于循环码多项式 $A(x)$都可以被 $g(x)$整除，也就是

$$\frac{A(x)}{g(x)}=Q(x)=\frac{x^{n-k}\cdot m(x)+r(x)}{g(x)}=\frac{x^{n-k}\cdot m(x)}{g(x)}+\frac{r(x)}{g(x)}$$

因此，用 $x^{n-k}\cdot m(x)$除以 $g(x)$，就得到商 $Q(x)$和余式 $r(x)$，即

$$\frac{x^{n-k}\cdot m(x)}{g(x)}=Q(x)+\frac{r(x)}{g(x)}$$

(3) 编码输出系统循环码多项式 $A(x)$为：

$$A(x)=x^{n-k}\cdot m(x)+r(x)$$

上述三步编码过程，在硬件实现时，可以利用除法电路来实现，这里的除法电路采用一些移位寄存器和模 2 加法器来构成。下面将以（7,3）循环码为例，来说明其具体实现过程。设该（7,3）循环码的生成多项式为：$g(x)=x^4+x^2+x+1$，则构成的系统循环码编码器如 8.1 所示，图中有 4 个移位寄存器，一个双刀双掷开关 e、f。当信息位输入时，开关位置“e”接 2,“f”接 2，输入的信息码一方面送到除法器进行运算，一方面直接输出；当信息位全部输出后，开关位置“e”接 1,“f”接 1，这时输出端接到移位寄存器的输出，是除法的余项，也就是监督位依次输出。

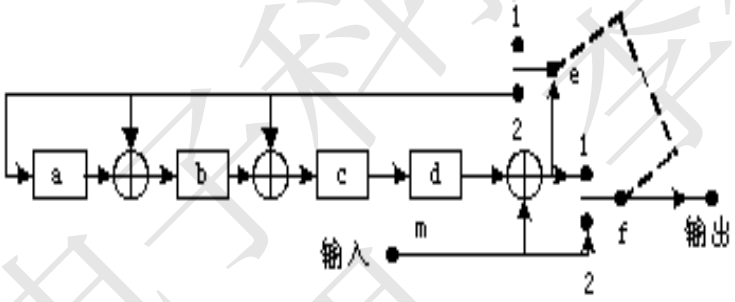


图 8.1　循环码编码器原理

二、循环译码

对于接收端译码的要求通常有两个：检错与纠错。达到检错目的的译码十分简单，通过判断接收到的码组多项式 $B(x)$是否能被生成多项式 $g(x)$整除作为依据。当传输中未发生错误时，也就是接收的码组与发送的码组相同，即 $A(x)=B(x)$，则接收的码组 $B(x)$必能被 $g(x)$整除；若传输中发生了错误，则 $A(x)\neq B(x)$，$B(x)$不能被 $g(x)$整除。因此，可以根据余项是否为零来判断码组中有无错码。

需要指出的是，有错码的接收码组也有可能被 $g(x)$整除，这时的错码就不能检出了。这种错误被称为不可检错误，不可检错误中的错码数必将超过这种编码的检错能力。

在接收端为纠错而采用的译码方法自然比检错要复杂许多，因此，对纠错码的研究大都集中在译码算法上。我们知道，校正子与错误图样之间存在某种对应关系。如同其它线性分组码，循环码的译码可以分三步进行：

（1）由接收到的码多项式 $B(x)$计算校正子（伴随式）多项式 $S(x)$；

（2）由校正子 $S(x)$ 确定错误图样 $E(x)$；

（3）将错误图样 $E(x)$ 与 $B(x)$ 相加，纠正错误。

上述第（1）步运算和检错译码类似，也就是求解 $B(x)$ 整除 $g(x)$ 的余式，第（3）步也很简单。因此，纠错码译码器的复杂性主要取决于译码过程的第（2）步。

基于错误图样识别的译码器称为梅吉特译码器，它的原理图如图 3-2 示。错误图样识别器是一个具有（$n-k$）个输入端的逻辑电路，原则上可以采用查表的方法，根据校正子找到错误图样，利用循环码的上述特性可以简化识别电路。梅吉特译码器特别适合于纠正 2 个以下的随机独立错误。

图 8.2 中 k 级缓存器用于存储系统循环码的信息码元，模 2 加电路用于纠正错误。当校正子为 0 时，模 2 加来自错误图样识别电路的输入端为 0，输出缓存器的内容；当校正子不为 0 时，模 2 加来自错误图样识别电路的输入端在第 i 位输出为 1，它可以使缓存器输出取补，即纠正错误。

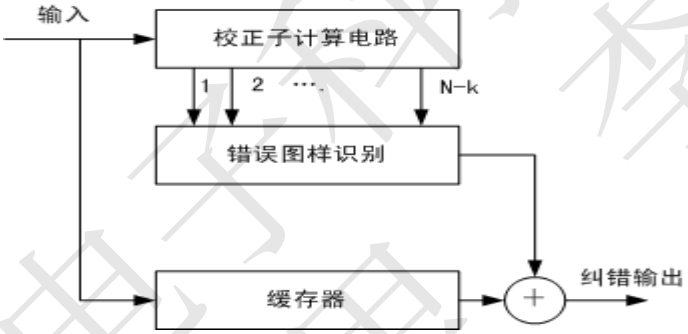


图 8.2　梅吉特译码器原理

§8.1.2 设计任务与要求

实验中要实现的循环编译码总体框图如图 8.3 所示

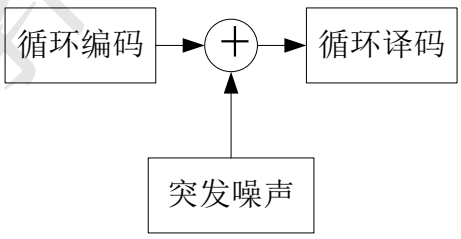


图 8.3　循环编译码总体框图

一、 （7，3）循环编码器，其原理框图如图 8.4 所示。

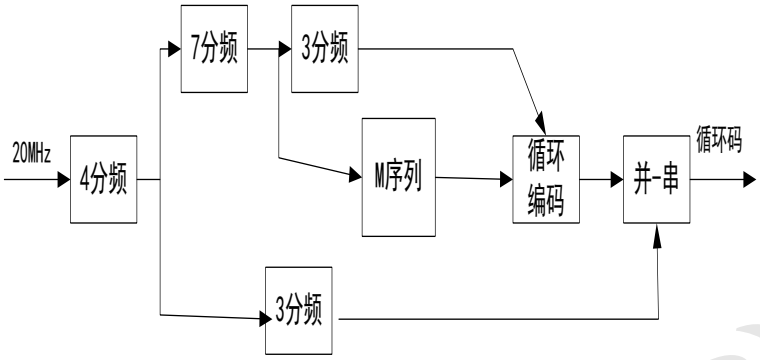


图 8.4　循环编码器的模块框图

① 时钟：时钟信号由实验板提供 20MHz 时钟（仿真时也选用 20MHz 时钟），经 4 分频得到 5MHz 方波信号。

② 信源：产生周期为 15 的 M 序列作为消息的输入。

③ 循环编码：设计一种（7，3）循环码，即 3 位信息码，4 位监督码。循环编码器实现的具体电路需要自己去设计。需要注意的是在电路设计时要考虑信号的连续性问题，不能由于插入监督码而中断信号。下面给出循环编码器的模块框图如图 3.4 所示，仅供大家参考。

编码电路如图 8.5 所示

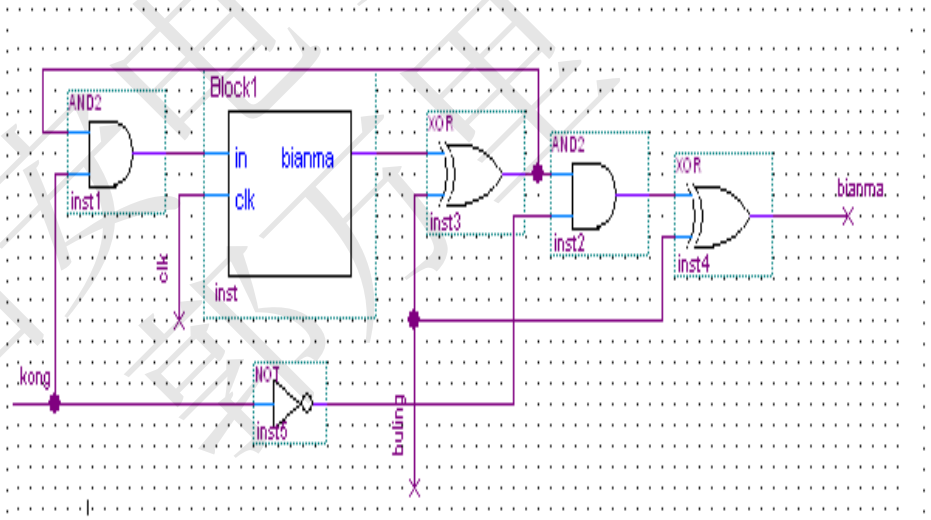


图 8.5　循环编码器电路

其中 bing 为补零输入：将 3 位信息位后补 4 个零，电路如图 3.6 所示。

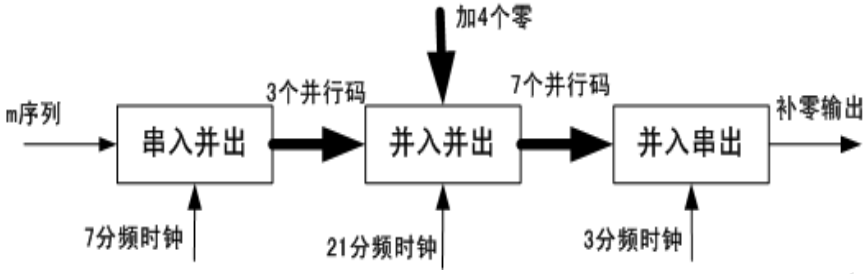


图 8.6　补 4 个零模块图

Block 模块为 4 位监督码生成模块（除法电路），如图 8.7 所示。

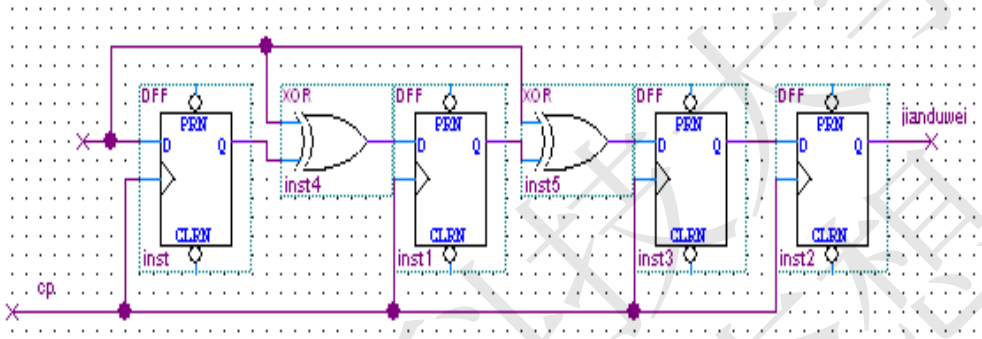


图 8.7　4 位监督码生成电路

④　并串转换：将 7 位并行码转换成 7 位串行码。

二、突发噪声产生：突发噪声用数字电路实现，使 m 序列的每一个周期出现一个或俩个错误。图 8.8 所示

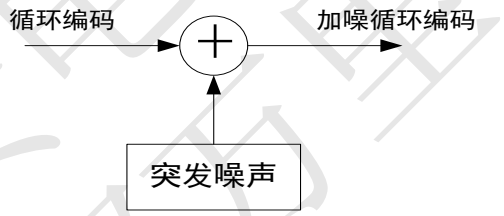


图 8.8　突发噪声产生

三、　（7，3）循环译码器，其原理框图如图 8.9 所示

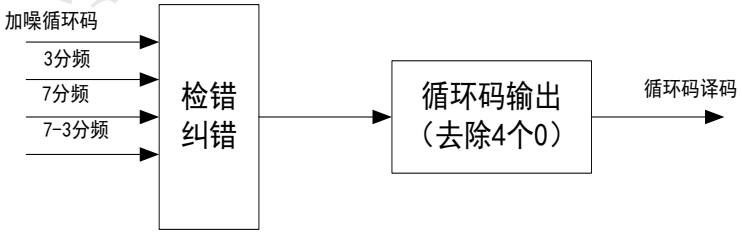


图 8.9　　循环译码器的模块图

②　　时钟：时钟信号由实验板提供 20MHz 时钟（仿真时也选用 20MHz 时钟），经分频

得到。

② 循环译码：应用梅吉特译码器的原理进行译码，循环译码器实现的具体电路需要自己去设计（或编写程序完成），这里给出循环译码器的模块图仅供大家参考。 译码电路原理图如图 8.10 所示。

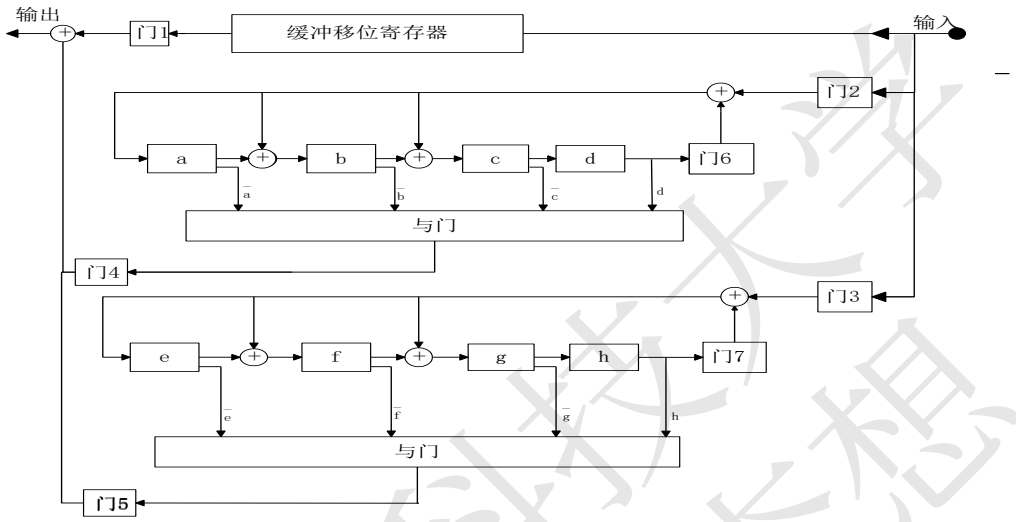


图 8.10　循环译码电路

门 1、门 2、门 3、门 4、门 5、门 6、门 7 在电路中的作用：

这些门的电路及输出波形如图 8.11、图 8.12、图 8.13 所示。

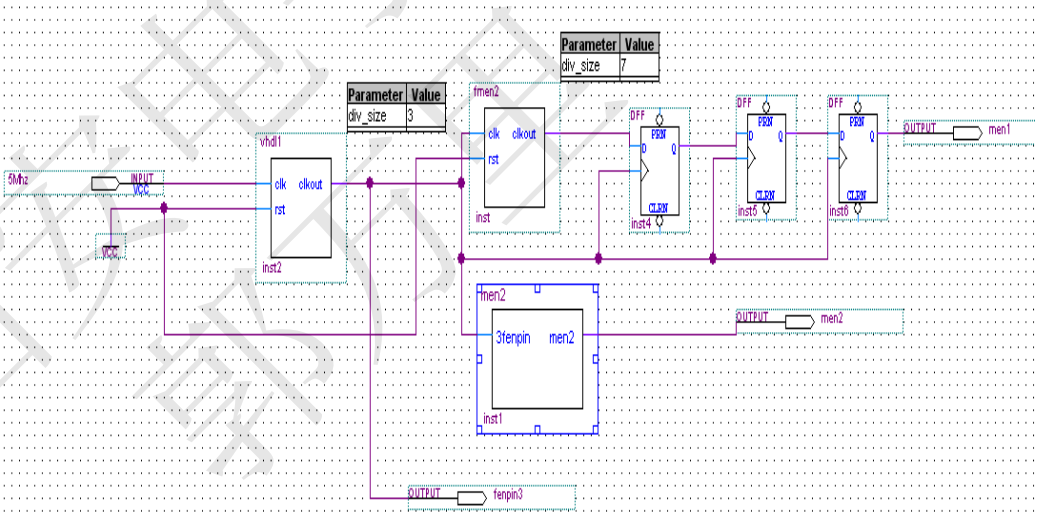


图 8.11　men1、men2 输出电路

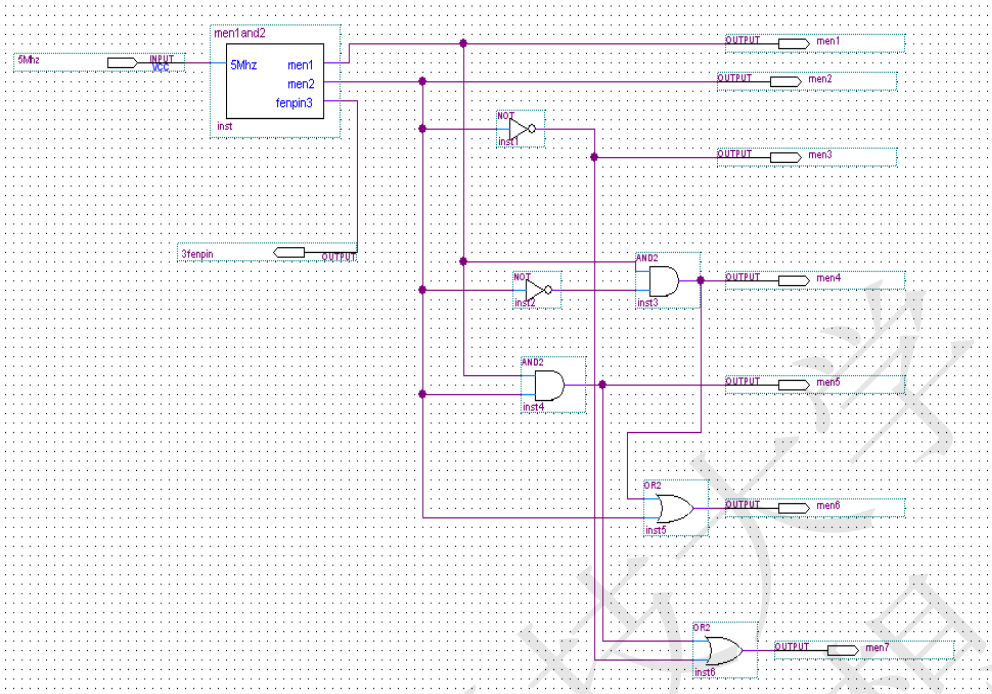


图 8.12　men3-men7 输出电路

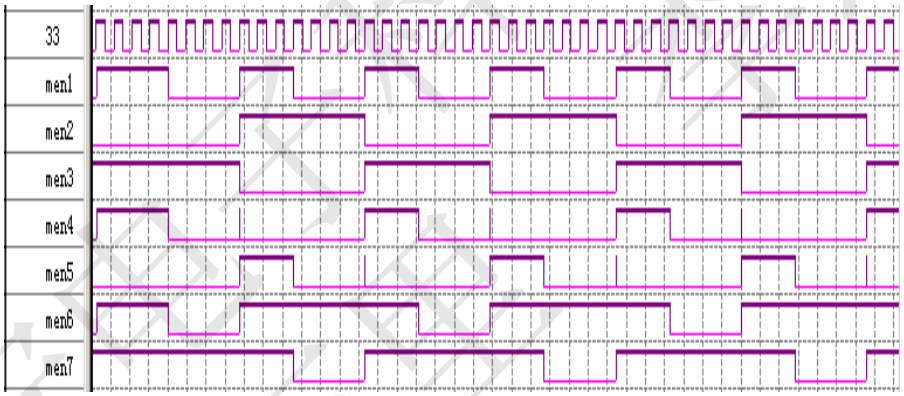


图 8.13 这些门的输出波形

其中门 1 控制信息码元的输出，由于信息码元有 3 位，为了只输出信息码元，所以门 1 的控制信号为 3 高 4 低。

门 2、门 3 交替控制 7 位码元的输出。门 4、门 5 控制信息位码元的输出。门 6、门 7 控制除法器通断。

④　并串转换：将 3 位并码转换成串码。

§8.1.3 测试

①　按系统方框图，将系统每部分都打包成模块，编译码模块又都有它自己的下层模块，最终完成循环编译码器。

②　设计好的循环编译码器先仿真得到正确的结果，然后需要下载到实验板中用示波

器和嵌入式逻辑分析仪观察编码信号。

③ 实现此系统可分 3 步完成：

第一步：电路设计或程序设计。

第二步：Quartus Ⅱ软件仿真。

第三步：在 FPGA 实验板中下载并用示波器观察循环编译码器信号。

§8.1.4 实验报告要求

⑴ 简述实验目的及实验原理。

⑵ 写出汉明编译码器的设计思想，画出测试的相应个点的波形图，分析译码后，汉明译码的纠错能力。

⑶ 在完成汉明编译码系统中出现的问题及解决的方法。

⑷ 循环编译码器通常应用在什么样的通信环境中。

⑸ 谈谈做设计性的实验对你学习的提高有没有帮助，你感觉设计性实验应该怎样做才能充分调动同学的积极性去参与。

& 8.2 (2,1,2)卷积编译码器的设计与实现

§8.2.1 实验目的

⑴ 了解 FPGA 软硬件知识，学习利用 QuartusII 实现电路的方法，能初步掌握之。

⑵ 了解卷积码的产生与译码的过程，掌握卷积码编码及译码方法。

⑶ 提高将编码理论应用到实践中的能力。

§8.2.2 实验仪器

1、512MHz 以上内存微计算机。

2、20MHz 双踪示波器、信号源。

3、Quartus Ⅱ语言环境。

4、fpga 实验板、若干导线。

§8.2.3 实现原理及框图

卷积码一般采用（n,k,m）来表示，其中 k 表示输入码元个数，n 为输出码元个数，而且 m 为编码器存储的存储器数。典型的卷积码一般会选择 n 和 k（〈n）值较小，但存储器可取较大值（m〈10〉，以获得既简单又高性能的信道编码。

卷积码与一般的分组码的区别是：在卷积码中，任何特定的时间单位内进入的信息组与前 m 个输入（记忆器中的存储器中存储的）有关。

由于在卷积码的束缚长度内，前后各组是密切相关的，一个组的监督元不仅取决于本组的信息元，而且也取决于前面 m 组的信息元。因此，虽然 n，k 比较小，但它的纠错能力却较强。

正是卷积码充分利用了各组之间的相关性，n 和 k 才可以取较小的数，因此在与分组码同样的传输速信率和设备复杂性相同的条件下，卷积码的性能一般比分组码好。

卷积码的编码器是一个有 k 个输入符号，n 个输出符号，编码速率为 R=k/n，且具有 m 节移位寄存器所构成的有限状有记忆系统，通常称作时序网络。描述这类时序网络的方法很多，大致可分为两大类型，即：解析表示法与图形表示法。在解析法中有可分为离散卷积法、数图法、格图法等。

卷积译码器的作用是采用可以使用误码率为最小的某种方法或准则，估计被编码的输入信息。在信息序列之间存在一对一的对应关系。卷积码的译码基本上可画分为两大类型：

代数译码和概率译码。在我们的实验里使用概率译码，因为在实际中最常采用的卷积码的译码方法。

Viterbi 译码属于概率译码，具有最佳性能。Viterbi 译码的主要思想就是根据产生卷积码的网格图[2] (或称为状态图) ,确定最可能的发送序列,这个最可能的发送序列对应网格图上的某一特殊路径。译码时,将接收序列与网格图上所有可能的传输路径相比较,求其汉明距离,从中选择具有最小距离的发送序列,距离最小表示接收码取此序列的可能性最大。

卷积编译码的详细原理请参考有关教科书。

§8.2.4 设计任务与要求

一、(2,1,2)卷积码编码器

实验中要实现的卷积编译码总体框图如图 8.14 所示

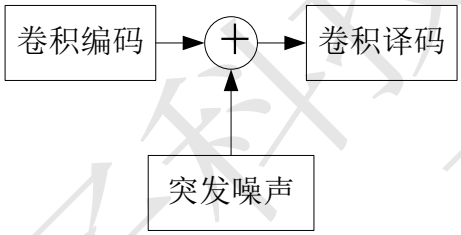


图 8.14 卷积编译码总体框图

设计一种(2,1,2)卷积码编码器,其原理方框图如图 8.15 所示。

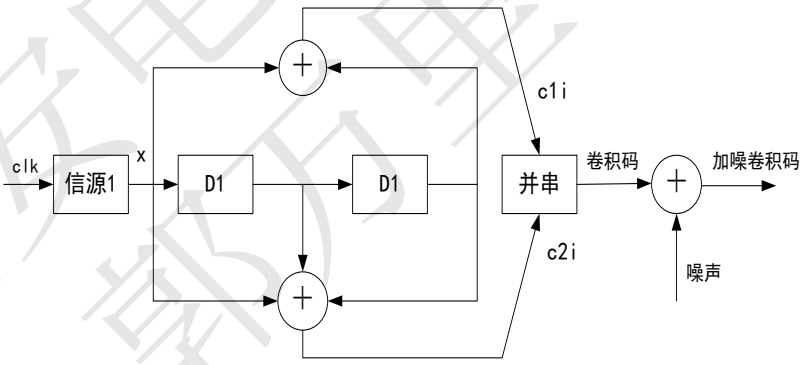


图 8.15 (2,1,2)卷积码编码器原理方框图

① 时钟信号为 5KHz。由实验板提供 20MHz 时钟，经分频得到（仿真时可由 20MHz 时钟分频得到）。

② 信源（x）：设计一个周期为 15 的小 M 序列作为信源。信源码率 5000bit/s 时钟信号。

③ 分频器：最好自己编写一个通用的奇偶分频程序，使用起来比较方便。分频次数

则由载波频率而定。

④ 卷积编码：由 D1、D2 两个移位寄存器以及莫尔加构成卷积编码器。设 D1、D2 两个移位寄存器的初始状态为 0，卷积编码器输入输出关系为：

输入：x1 x2 x3 x4 x5 x6 x7 x8 x9.....

输出：c11 c21 c12 c22 c13 c23 c14 c24 c15 c25 c16 c26……

⑤ 并串转换:将 2 位并码转换成串码。

⑥ 按系统方框图，将系统打包成模块，在顶层文件中调用各模块。

二、突发噪声产生

突发噪声用数字电路实现，使 m 序列的每一个周期出现一个或俩个错误。图 8.16 所示

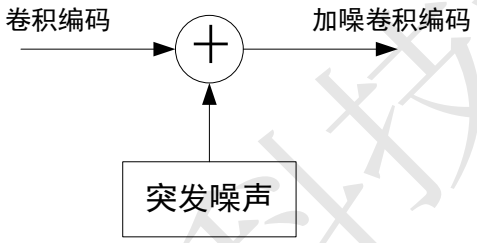


图 8.16 突发噪声产生

三、(2,1,2)卷积译码器

① 所需时钟：由实验板提供 20MHz 时钟（仿真时也选用 20MHz 时钟），经分频得到。

② 卷积编码信号：由选择卷积编码器设计组的学生提供。

③ 卷积译码：卷积译码器采用维特比算法实现，由于算法比较复杂，最好选用硬件语言实现。具体步骤如下：

(a) 计算进入每一状态的各个分支路径的度量(称为分支度量)，其中分支度量定义为接收码元与可能码元之间的欧几里得距离(软判决) 或汉明距离 (硬判决) 。

(b) 把各分支度量同各自相应的前一时刻状态度量相加求和,得到路径度量。在每个状态中,从到达这一状态的路径度量里选出并保留最小者作为当前时刻的状态度量。同时保留与之相应的路径作为幸存路径。

(c) 在各状态的幸存路径中选出状态度量最小的一条,顺此回溯,得到译码输出。

上述步骤(a)，步骤(b) 通常称之为加比选运算(ACS)，步骤(c) 为回溯(Traceback) 。参考远离框图如图8.17所示。

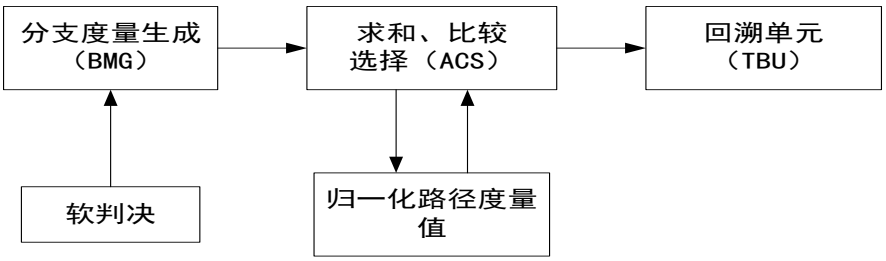


图 8.17　(2,1,2)卷积码译码器原理方框图

④　设计好的译码器和编码器连接起来先仿真得到正确结果，最终将编译码器下载到实验板中，在示波器上观察编译码器的信号。

⑤　按系统方框图，将系统分成两部分：编码模块和译码模块。每部分都打包成模块，编译码模块又都有它自己的下层模块，最终完成卷积编译码器。

§8.2.5 测试

实现此系统可分 3 步完成：

①　电路设计或程序设计。

②　Quartus Ⅱ软件仿真。

⑤　　在 FPGA 实验板中下载并用示波器观察时钟同步信号。

§8.2.6 实验报告要求

⑴ 简述实验目的及实验原理。

⑵ 写出卷积编译码器的设计思想，画出测试的相应个点的波形图，分析译码后，卷积译码的纠错能力。

⑶ 在完成卷积编译码系统中出现的问题及解决的方法。

⑷　卷积编译码器通常应用在什么样的通信环境中。

⑸ 谈谈做设计性的实验对你学习的提高有没有帮助，你感觉设计性实验应该怎样做才能充分调动同学的积极性去参与。

& 8.3 （7，3,5）RS 编译码器的设计与实现

§8.3.1 实验目的

⑴ 了解 FPGA 软硬件知识，学习利用 QuartusII 实现电路的方法，能初步掌握之。

⑵ 了解 RS 码的生成多项式 g(x)与编、译码器之间的关系，码距与纠、检错能力之间的关系，掌握 RS 码的编码及译码方法。

⑵ 提高将编码理论应用到实践中的能力。

§8.3.2 实验仪器

1、512MHz 以上内存微计算机。

2、20MHz 双踪示波器、信号源。

3、Quartus Ⅱ语言环境。

4、fpga 实验板、若干导线。

§8.3.3 实现原理及框图

RS（n，k，t）码是一种有很强纠错能力的多进制 BCH 码，是多进制编码器，也属于线性分组码。它首先由里德(Reed)和索洛蒙(Solomon) 应用 MS 多项式于 1960 年构造出来的。它不但可以纠正随机差错，而且对突发错误的纠错能力也很强，因此广泛用于差错控制系统中，以提高数据传输的可靠性。在 RS（n，k,t）码中，n 表示编码长度，k 表示数据信息,t 为纠错能力。见表 1 所示。

表 1 RS（n，k,t）码符号的意义

码长	$n = 2^m - 1$
信息码	k 符号
监督码	n-k=2t 符号
最小码距	D=2t+1

满足关系：t=(n-k)/2,m=log2n，RS 码的所有元素都是定义在 GF（2m）上。对于不同的 m 对应着一个本原多项式，从本原多项式就可以得到有限域各元素。

RS 码的详细原理论述可参考王新梅、肖国镇编著：《纠错码-原理与方法》，电子科技大学出版社，1991。

§8.3.4 设计任务与达到的目标

实验中要实现的 RS 编译码总体框图如图 8.18 所示

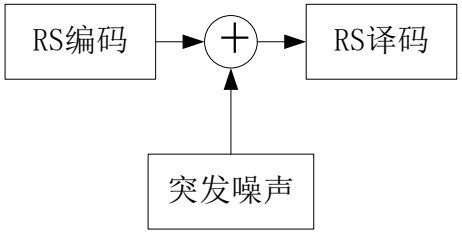


图 8.18　RS 编译码总体框图

一、RS(7,3,5)编码器设计，原理框图如图 8.19 所示。

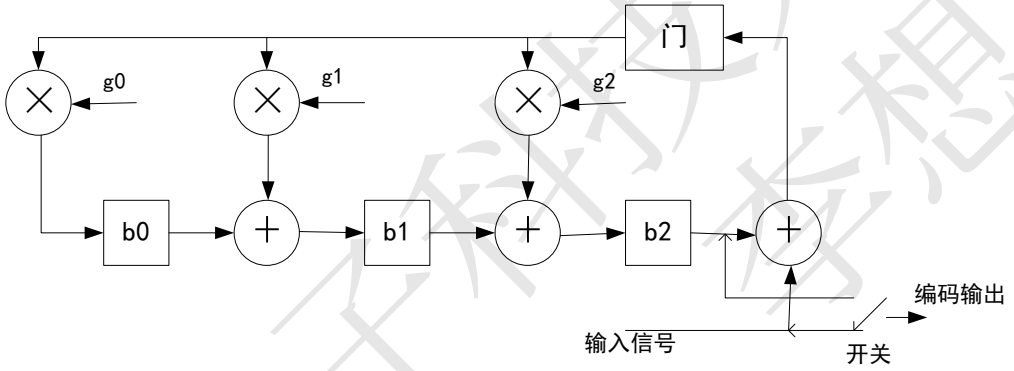


图 8.19　RS(7,3,5)编码器原理框图

①　信源：设计一个周期为 15 的小 M 序列作为信源。信源码率 5000bit/s。在经串并转换变为 3 位并行码（八进制码）。

②　时钟信号：时钟信号为 5KHz。由实验板提供 20MHz 时钟，经分频得到（仿真时可由 20MHz 时钟分频得到），其它所需时钟也由分频得到。

③　分频器：最好自己编写一个通用的奇偶分频程序，使用起来比较方便。分频次数则由载波频率而定。

④　RS(7,3,5)编码器，原理框图如图8.12所示。RS(7, 3) 码的生成多项式为

$$g(x) = x^4 + a^3x^3 + x^2 + ax + a^3$$

图 2.12 中 b0、b1、b2 为八进制移位寄存器，g0、g1、g2 为多项式。

⑤　在编码过程中输入信号不能间断。

⑥　将设计的 M 序列发生器、RS 编码器、噪声发生器封装成模块，构成原理框图。先仿真，经测试后如果没有错误，下载到实验板中检验 RS 码的纠错能力。

二、突发噪声产生

突发噪声用数字电路实现，使 m 序列的每一个周期出现一个或俩个错误。图 8.20 所示

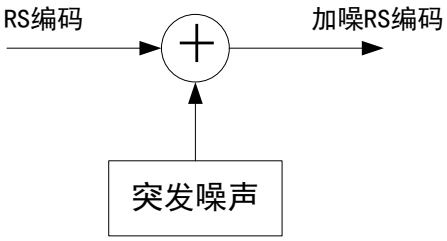


图 8.20　突发噪声产生

三、RS(7,3，5)译码器设计

①　信源（RS 编码）：由选择 RS 编码器设计组的学生提供。

②　时钟信号：由实验板提供 20MHz 时钟，经分频得到（仿真时可由 20MHz 时钟分频得到）。

③　分频器：最好自己编写一个通用的奇偶分频程序，使用起来比较方便。分频次数则由载波频率而定。

④　RS（7，3，5）译码器设计。选择一种译码算法将 RS 编码器解码。

⑤　将各部分电路封装成模块。

⑥　 与编码组一起完成编译码的测试。

§8.3.5 实验报告要求

⑴ 简述实验目的及实验原理。

⑵ 写出 M 序列发生器、RS 编码器、噪声发生器的设计思想，画出测试的相应个点的波形图，分析译码后，RS 译码的纠错能力。

⑶ 在完成通信系统中出现的问题及解决的方法。

⑷　RS 编译码器通常应用在什么样的通信环境中。

⑸ 谈谈做设计性的实验对你学习的提高有没有帮助，你感觉设计性实验应该怎样做才能充分调动同学的积极性去参与？