TCP-II 数字系统设计创新平台

实验指导书

EDA 版

2018-5

目录

第一章　TCP-II 数字系统设计创新平台简介

一、概述

TCP-II 数字系统设计创新平台是一款基于 Altera Cyclone IV 系列 FPGA 芯片的综合性实验开发平台。主实验板采用高密度核心板加底板双层架构，板载 USB-Blaster 下载器。整个实验系统采用模块化设计，系统配置丰富、操作灵活方便、没有复杂连线操作、可靠性强，同时还提供了丰富的外设资源和扩展接口。

二、核心板资源

1、核心板标配 Altera Cyclone IV FPGA 芯片 EP4CE30F23C8N，484 脚 BGA 封装，6 层高密度板，可选配升级为 EP4CE40F23C8N 核心板。
2、一片 8MByte 串行配置芯片 EPCS64。
3、JTAG、AS 编程接口。
4、一路 50M 高速时钟源、一路 25M 时钟源。
5、一片 512KByte SRAM IS61LV25616AL，可升级为 1M Byte。
6、一片 32MByte SDRAM MT48LC16M16A2。
7、一片 16MByte NOR FLASH AM29LV128ML。
8、四个轻触按键，1 个复位按键。
9、四个通用 LED 指示灯。
10、四个 50 脚 2.0 间距双排接口和一个 16 脚 2.0 间距双排接口，提供 197 个与核心板资源不复用的 IO 供底板资源和扩展资源使用。

三、底板资源

1、板载高速 USB-Blaster 下载器，仅需一根 USB 线，无需频繁插拔 IDC 排线，即能完成对 FPGA 核心板的 JTAG、AS 方式下载。
2、一路数字时钟源，可通过跳线选择向 FPGA 芯片提供 1Hz、10Hz、100Hz、1KHz、10KHz、100KHz、1MHz、10MHz 等多种时钟输入，便于开展实验教学。
3、一个 1602 字符液晶显示模块。
4、一路 16 位 VGA 接口，可输出 65536 色，1280×1024 分辨率。
5、一路并行 100M 高速以太网接口，支持以太网开发。
6、一个音频 CODEC 模块，含音频输入、输出接口和麦克输入接口。

7、一路 PS/2 键盘、鼠标接口。
8、一个 SD 卡接口，支持 SD 卡的两种读写方式。
9、一路 UART 串口，方便与上位机进行串口通信。
10、两路 40 针扩展 I/O 接口（兼容 DE2），无信号复用，方便进行扩展设计。
11、16 个拨动开关。
12、4 个轻触按键。
13、16 个蓝色 LED 灯。
14、一组 6 位动态七段数码管。
15、板载 9V、5V、3.3V、2.5V、1.2V 等多路电源模块。

四、产品特色

1、核心板采用 484 脚 BGA 封装的 Altera Cyclone IV FPGA 芯片 EP4CE30F23C8N，超 200 万门，6 层高密度板设计。
2、板载高速 USB-Blaster 下载器，仅需一根 USB 线，无需频繁插拔 IDC 排线，即能完成对 FPGA 核心板的 JTAG、AS 方式下载。
3、具备 1Hz、10Hz、100Hz、1KHz、10KHz、100KHz、1MHz、10MHz 等多时钟输入选择，便于进行计数，分频实验，节省了学生的开发时间，提高效率，便于开展实验教学。
4、两路 40 针扩展 I/O 接口，与 ALTERA DE2 板完全兼容，方便连接扩展模块，便于二次开发。
5、可根据用户需要选配或定制增加扩展模块。
6、实验主板及扩展模块上下均带亚克力保护板。

五、支持多门课程教学

1、数字逻辑设计
2、EDA 技术与 FPGA 设计
3、SOPC 技术与应用
4、数字电子技术课程设计
5、计算机组成原理
6、大学生电子设计大赛等

六、实验平台全貌

TCP-II 数字系统设计创新平台的实物图如图 1.1 所示：

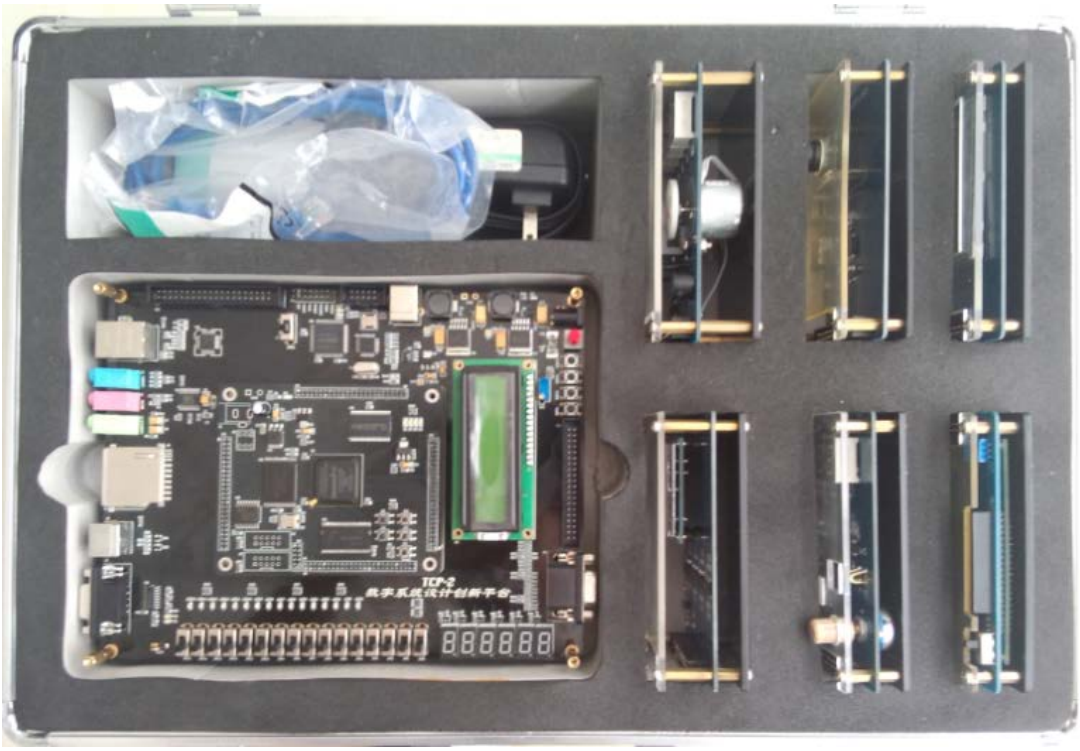


图 1.1　TCP-II 实物外观图

七、多时钟选择

TCP-II 实验平台板载 1 路数字时钟源，可通过跳线选择向核心板 FPGA 芯片提供 1Hz、10Hz、100Hz、1KHz、10KHz、100KHz、1MHz、10MHz 等多种时钟输入，便于开展实验教学。如图 1.2 所示。

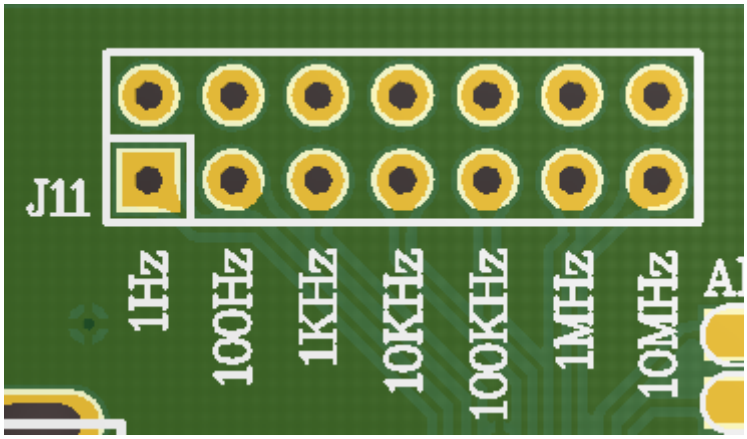


图 1.2　多时钟选择

八、下载方式说明

TCP-II 系统底板上侧板载高速 USB-Blaster 下载器，仅需一根 USB 线，无需频繁插拔 IDC 排线，通过拨动开关 SW16 选择完成对 FPGA 核心板的 JTAG 或 AS 方式下载。如图 1.3 所示。

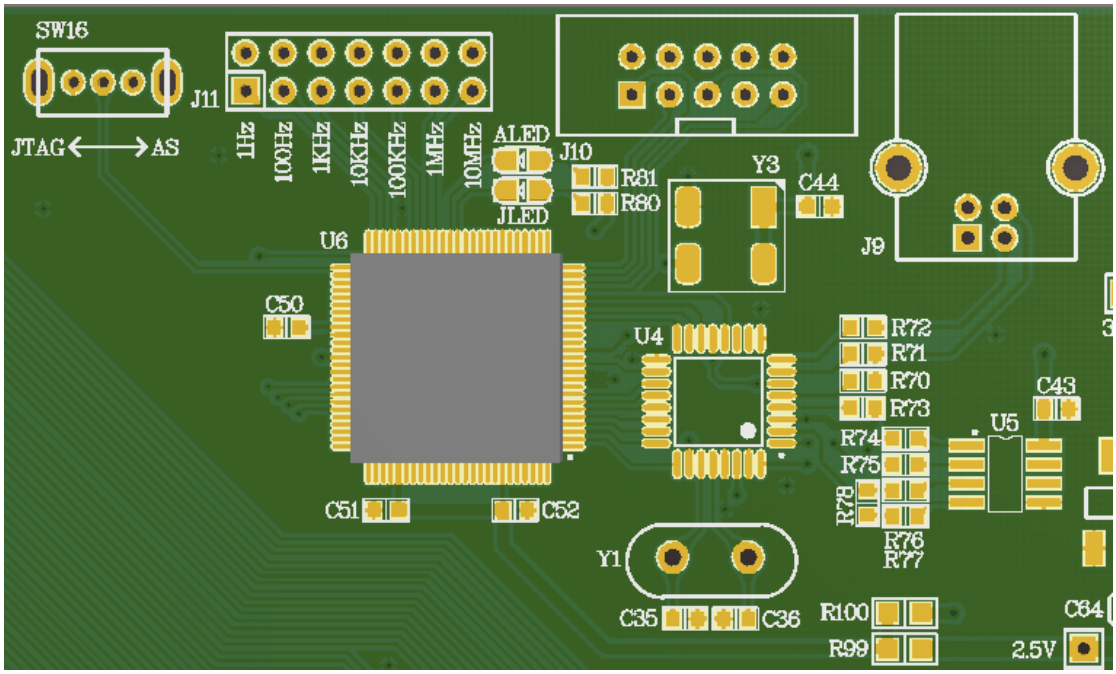


图 1.3 板载下载器

第二章　Quartus II 使用入门与 FPGA 实验过程简介

Quartus II 可编程逻辑开发软件是 Altera 公司为其 FPGA/CPLD 芯片设计推出的专用开发工具，是 Altera 公司最新一代功能更强的 EDA 开发软件，可完成从设计输入，综合适配，仿真到下载的整个设计过程。

Quartus II 提供了一个完整的多平台开发环境，它包含 FPGA 和 CPLD 整个设计阶段的解决方案。Quartus II 集成环境包括以下内容：系统级设计，嵌入式软件开发，可编程逻辑器件设计、综合、布局和布线，验证和仿真。

Quartus II 也可以直接调用 Synplify Pro、ModelSim 第三方 EDA 工具来完成设计任务的综合与仿真。Quartus II 与 MATLAB 和 DSP Builder 结合可以进行基于 FPGA 的 DSP 系统开发，方便快捷。Quartus II 还内嵌 SOPC Builder/Qsys，可实现 SOPC 系统的开发。

Quartus II 13.0 主界面如图 2.1 示。

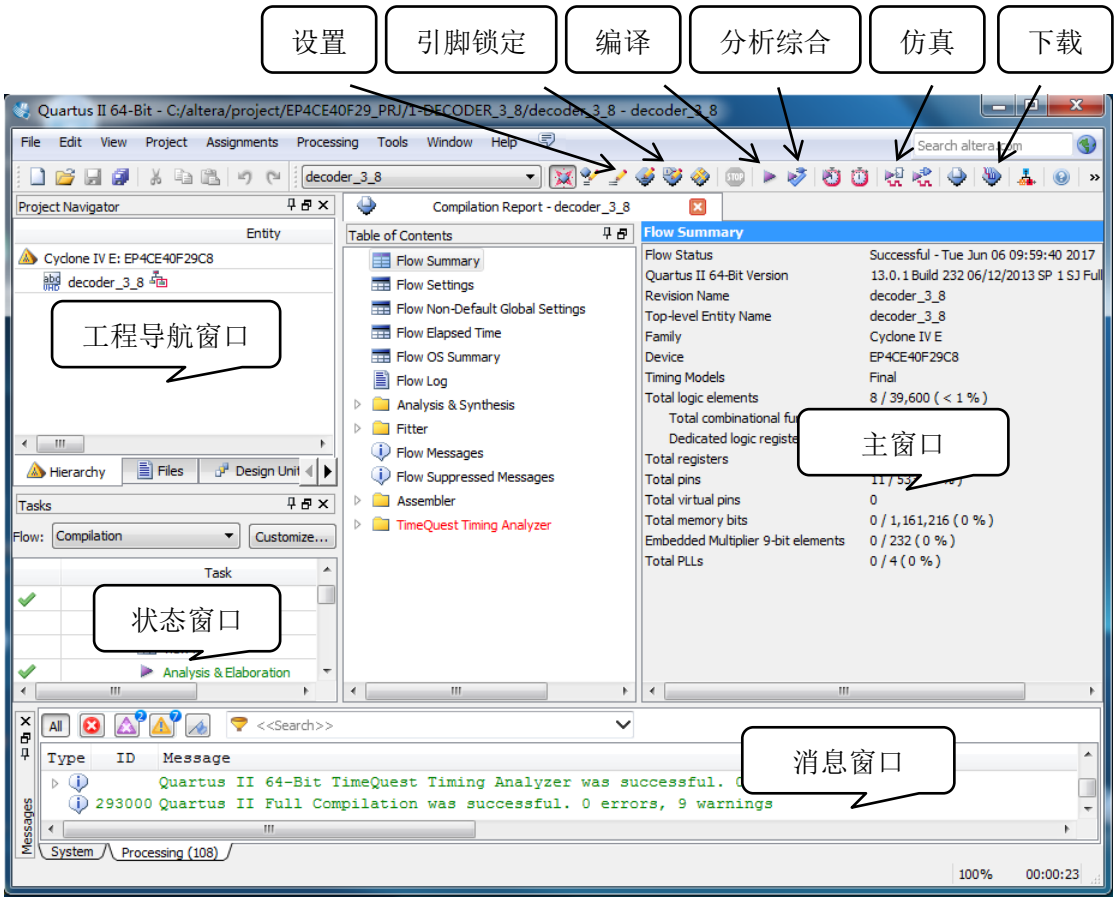


图 2.1　Quartus II 13.0 界面

一、Quartus II 基本设计流程

Quartus II 设计的主要流程包括创建工程、设计输入、分析综合、编译、仿真验证、编程下载等，其一般流程如图 2.2 所示。下面以硬件描述语言输入法设计计数器为例，说明 Quartus II 的设计流程。

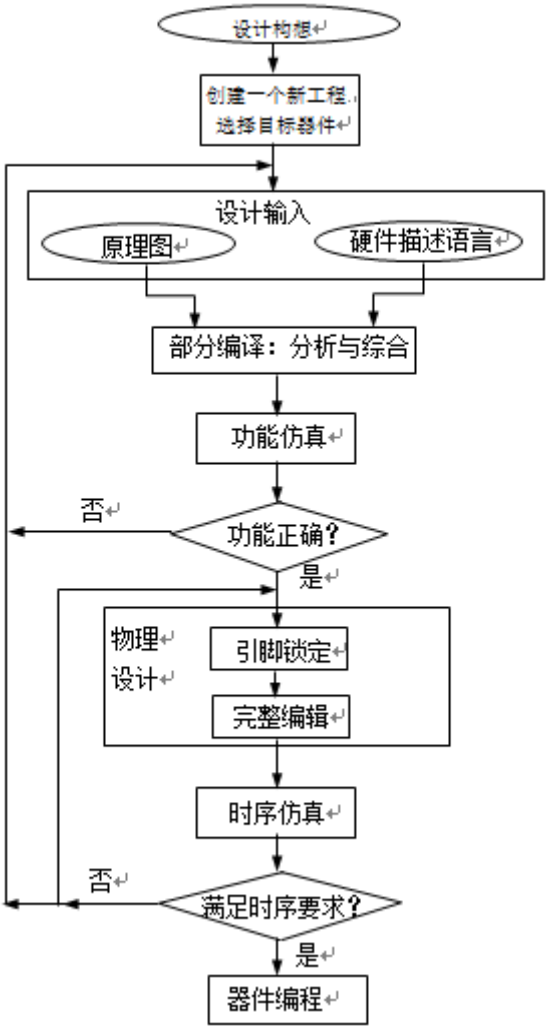


图 2.2 Quartus II 的设计流程

1、 创建工程

使用 Quartus II 设计电路被称作工程。Quartus II 每次只进行一个工程，并将该工程的全部信息保存在同一个文件夹中。开始一项新的电路设计，首先要创建一个文件夹，用以保存该工程的所有文件。之后便可通过 Quartus II 的文本编辑器编辑 VHDL/Verilog 源文件并存盘。

2、 设计输入

Quartus II 中包含原理图输入和硬件描述语言输入两种方法。

（1）原理图输入

原理图输入的优点是，设计者不必具有诸如编译技术、硬件描述语言等新知识就能迅速入门，完成较大规模的电路系统的设计，且具有直观，易于理解的特点，适合于初学者使用。但画电路图不如输入代码方便，所以在设计比较复杂电路时不宜采用。

按图 2.3 所示的步骤操作，即可进入原理图编辑器，此时出现原理图编辑器窗口，如图 2.4 所示。原理图编辑工具栏各按钮的功能如表 2.1 所示。

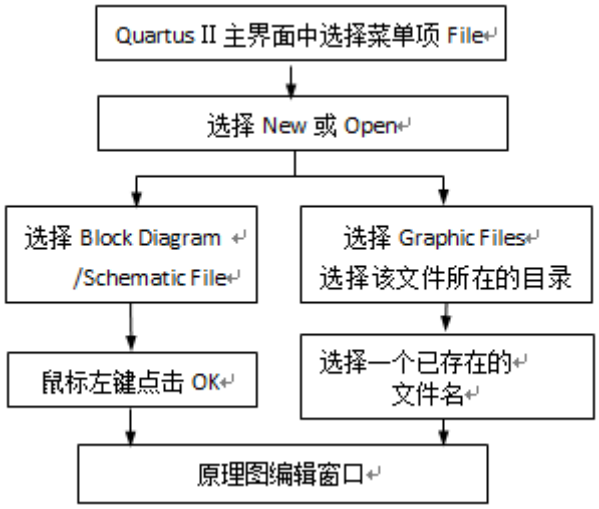


图 2.3 进入原理图编辑器操作方法

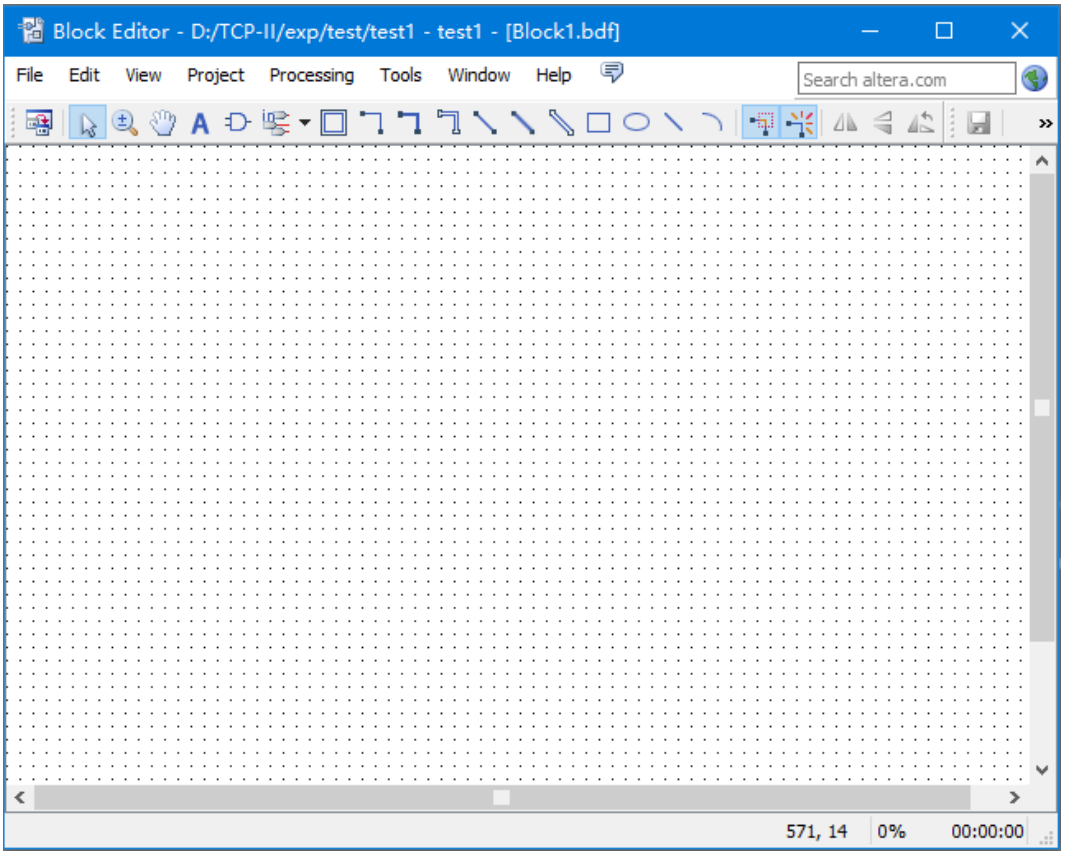


图 2.4 原理图编辑界面

图标	功能	图标	功能
	选择工具		文本工具
	插入符号		对角线工具
	单条连线		数组连线
	弧形工具		橡皮筋功能
	部分连线		放大缩小
	全屏显示		

表 2.1 原理图编辑工具栏各按钮功能说明

在原理图编辑器的空白处选择一个适当位置双击鼠标左键，或点击工具条上的与门符号（表示要插入一个元件符号），均可以调用库元件。

使用上述两种方法中的任何一个，将出现图 2.5 所示的窗口。在该窗口中可以选择库名，再在该库中选择库元件名，库元件符号便出现在窗口中，再用鼠标将其拖至适当的位置即可。所用的主要元件库如下：

① 基本元件库（primitives）

包括基本门电路（primitives\logic）、各种触发器、锁存器（primitives\storage）和输入/输出引脚（primitives\pin）。

② 其它元件库（others）

包括 74 系列器件（other\maxplus2）、宏功能模块评估（other\Opencore_plus）。

③ 参数化元件库（megafunctions）

包括算术组件（累加器、计数器、加法器、乘法器和 LPM 算术函数）、门电路（多路复用器、LPM 门函数）、I/O 组件、千兆位收发器块(GXB)、LVDS 接收器和发送器、存储组件（存储器、移位寄存器、LPM 存储器函数）。常用 I/O 组件有时钟数据恢复(CDR)、锁相环(PLL)、双数据速率(DDR)等。

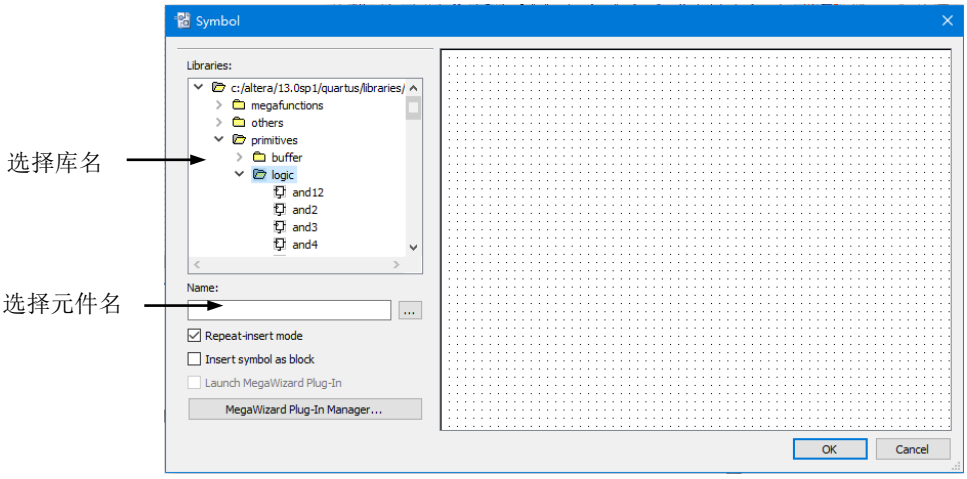


图 2.5 选择元件符号

创建/编辑一张原理图的主要工作包括：从库中调用元件符号，加入原理图；删除或复制所选中的元件符号；把各个元件符号用连线连接起来（或删除不需要的连线）；把电路的输入/输出引脚和电路内部相应元件的输入/输出端口连接起来；为输入/输出引脚、信号线等命名；移动元件或连线使图形美观；保存已经编辑好的原理图（.bdf 文件）等。

（2）硬件描述语言输入

如前所述，VHDL/Verilog 具有突出的优点，所以成为主流的硬件描述语言，广泛应用于电子设计之中。采用 VHDL/Verilog 输入的具体方法将结合下面的设计实例加以介绍。

3、 分析综合

综合是将硬件描述语言（或原理图）设计输入转化为由与、或、非门，RAM，触发器等基本逻辑单元组成的逻辑电路，并根据约束条件优化，生成门级逻辑电路，输出网表文件，供下一步的布线布局使用。性能优异的综合工具能够使所设计的电路占用芯片的面积更小、工作频率更高。这是评定综合工具优劣的两个重要指标。

在 Quartus II 中，从 Processing 菜单执行 Start / Start Analysis&Synthesis，启动分析与综合模块。该模块将检查工程的逻辑完整性和一致性，并检查边界连接和语法错误。它使用多种算法来减少门的数量，删除冗余逻辑以及尽可能有效地利用器件体系结构，产生用目标芯片的逻辑元件实现的电路，生成网表文件，构建工程数据库。

分析综合后，可执行 Tools /Netlist viewers/ RTL Viewer，查看 RTL 视图。

4、 仿真

仿真分为功能仿真和时序仿真。在 Assignments/Settings 对话框左侧列表中，选择 Simulator Settings，可将 Simulator mode（仿真模式）设为 Timing（时序仿真）或 Functional（功能仿真）。仿真之前需建立仿真波形文件。

（1）功能仿真

功能仿真主要是验证综合工具生成的电路是否符合设计要求。先执行 Processing/ Generate Functional Simulation Netlist，生成功能仿真网表，然后执行 Processing/Start Simulation，进行功能仿真。根据仿真得到的输出波形，分析电路是否满足要求。

（2）时序仿真

时序仿真包含了延时信息，它能较好地反映芯片的工作情况。对于一个实际的 PLD 设计项目，时序仿真不能省略，因为延时的存在，有可能影响系统的功能。

进行时序仿真之前，需要执行适配（Start Fitter）或全编译（Start

Compilation)。

功能仿真过程不涉及任何具体器件的时延特性。不经历适配阶段，在设计项目编辑编译（或综合）后即可进入仿真器进行模拟测试。直接进行功能仿真的好处是设计耗时短，对硬件库、综合器等没有任何要求。对于规模比较大的设计项目，综合与适配要花较长时间，如果每一次修改都要进行时序仿真，显然会大大降低工作效率。因此，通常的做法是，首先进行功能仿真，待确认设计文件所表达的功能满足要求后，再进行综合、适配和时序仿真，以便把握设计项目在硬件条件下的运行情况。

5、 编译

启动编译器可以对工程项目进行全编译。编译器是一个应用程序，它控制 Quartus II 中各个模块的运行。

编译器中包含如下 4 个主模块：

（1）分析和综合（Analysis & Synthesis）模块：产生用目标芯片的逻辑元件实现的电路；

（2）适配（Fitter）模块：将前一步确定的逻辑元件在目标芯片上分配精确的位置；

（3）组装（Assembler）模块：生成下载文件。

（4）时序分析（Timing Analyzer）模块。

编译器可以每次单独运行一个模块，也可以依次调用多个模块。

执行全编译的快捷方法是点击工具按钮 或执行 Processing / Start Compilation。

6、 编程下载

编译和仿真验证通过后，就可以进行下载了。在下载前，首先要通过综合器产生的网表文件配置于指定的目标器件中，使之产生最终的下载文件。把适配后生成的下载或配置文件通过下载器下载到 FPGA 或 CPLD 中，以便进行硬件调试和验证。并通过硬件测试来验证设计项目的实际工作情况。

通过以上的简要介绍，对一个逻辑电路的设计流程有了一个初步的了解。

二、Quartus II 设计示例

下面以建立一个一位十进制计数器为例来说明其使用方法。

1、 创建工程的准备工作

前已提及，开始一项新的电路设计，首先要创建一个文件夹，以便保存该工程的所有文件。通过QuartusII的文本编辑器编辑VHDL/Verilog源文件并存盘。

具体步骤如下：

（1） 新建一个文件夹。假设本项设计的文件夹取名为 counter，路径为 c:\altera\counter。

（2）输入源程序。打开 Quartus II，选择菜单 File/New。在 New 窗口中的 Device Design Files 中选择编辑文件的语言类型，这里选择“Verilog File”，如图 2.7 所示。然后在 Verilog 文本编译窗中输入 Verilog 示例程序，参见图 2.8。该程序的实体名为 CNT_10。

（3）文件存盘。选择 File/Save As 命令，找到已设立的文件夹 c:\altera\counter 即可存盘，存盘文件名最好与实体名一致，即 CNT_10.v。

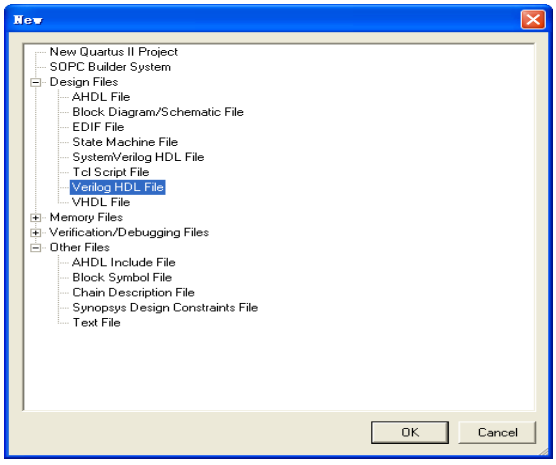


图 2.7 选择编辑文件语言类型

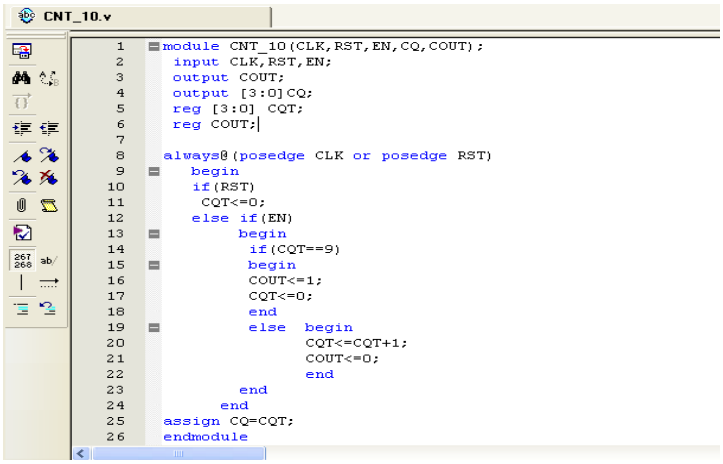


图 2.8 Verilog 文本编辑窗口及计数器程序

2、 创建工程

在菜单中选择 File/New Project Wizard，出现新建工程向导，三个输入栏中分别输入的是保存的路径及工程文件夹，工程的名称和顶层实体的名称。建议工程名与顶层实体名称保持一致。输入完毕点击 Next，将出现添加工程文件对话框，这时可将已经写好的 Verilog 文件加入到工程中。

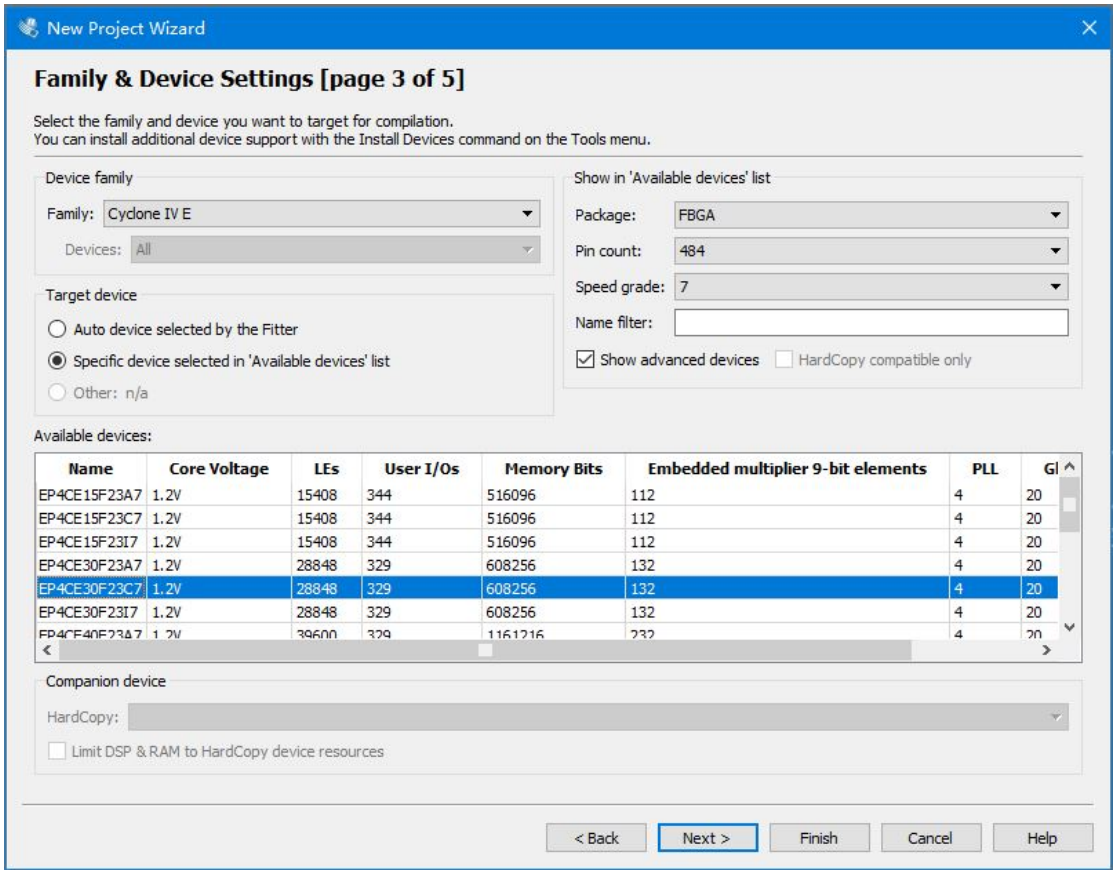


图 2.9　目标器件选择对话框

在图 2.9 所示的对话框中，完成选择器件的工作。这里以 TCP-II 数字系统设计创新平台为例，来说明具体操作方法。器件系列选择 Cyclone IV E，具体芯片型号为 EP4CE30F23C7，右面的三个下拉框用来限制芯片的封装形式 FBGA、管脚数 484 和速度等级 7。选择完成后，点击 Next，出现选用第三方 EDA 工具窗口。本例不选用第三方 EDA 工具，直接单击 Next。接下来出现的对话框给出了所生成工程的信息，单击 Finish 就完成了工程创建。这时工程导航窗口中的内容已经发生了改变。该窗口下面有三个页选项（图 2.10），Hierarchy 页中的内容是实体的层次结构，Files 页中的内容是工程包含的文档，这两个都是很常用的。

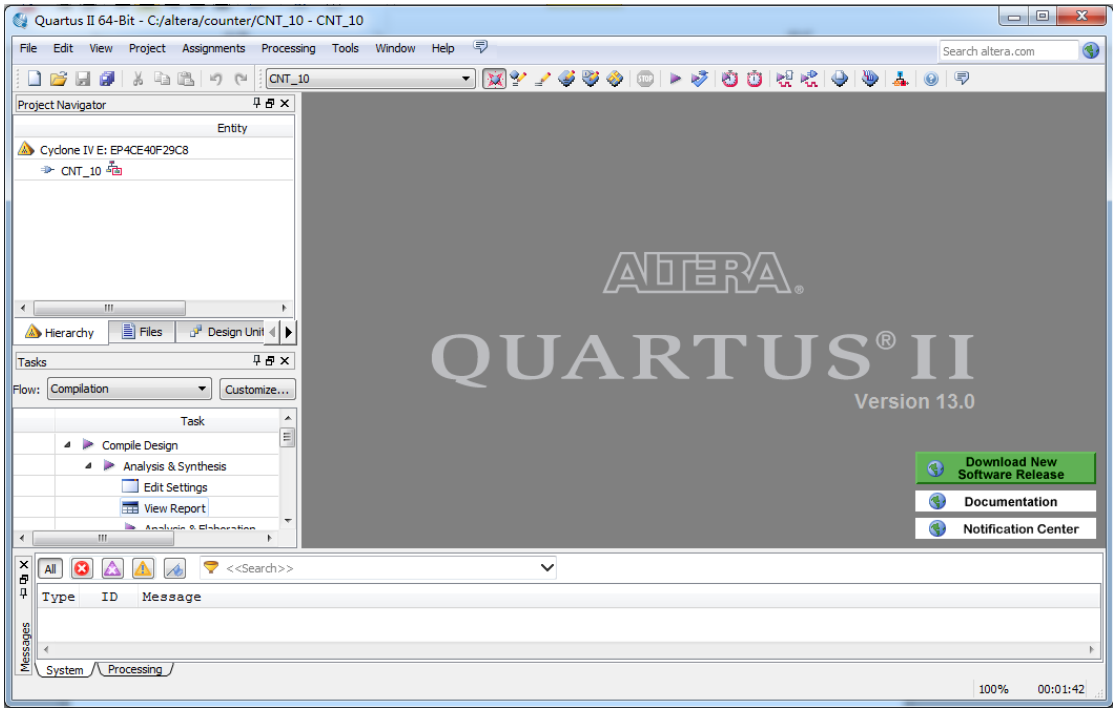


图 2-10 工程导航窗口

3、 添加/创建新文件

如果已经完成了 VHDL/Verilog 源文件的编写工作，只需将它加进工程中。

在 File 页中的 Device Design File 上点击鼠标右键，然后在 Add/Remove Files in Project 上点击鼠标左键，打开添加文件对话框，就可以添加文件了。也可以在 Quartus II 中创建 VHDL/Verilog 源文件。

4、 分析综合

在建立 VHDL/Verilog 文件以后，就可以进行分析综合，点击工具栏中 Start Analysis & Synthesis 按钮 启动分析综合过程。如果出现错误，则需要根据信息窗口的错误提示进行修改。文件存盘后重新启动分析综合过程。通过综合生成了逻辑电路网表文件，这时执行 Tool/Netlist Viewers/RTL Viewer 可以查看电路综合结果。

接下来可以对电路进行功能仿真，检查所综合的电路在功能上是否能够达到预期要求。

5、 建立激励波形文件

在 Quartus II 集成开发环境中无法对硬件描述语言进行仿真，只能对电路在激励波形的作用下进行仿真。因此进行仿真之前，首先需建立激励波形文件。选择命令 File/New，出现新建波形对话框，在该窗口中点击 Other Files 选项，选择 Vector Waveform File，点击 OK，出现波形编辑器窗口。选择命令 Edit/Insert/Insert Node or Bus，出现 Insert Node or Bus（插入节点或总

线）窗口，单击 Node Finder 按钮，出现节点查找器窗口，如图 2.11 所示。

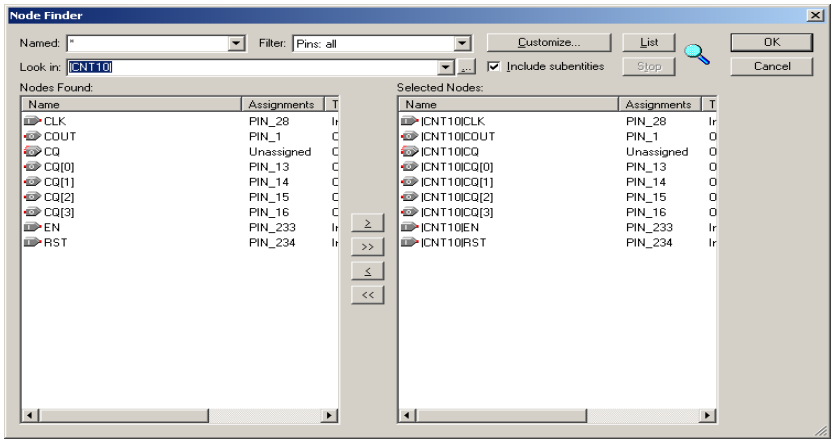


图 2.11　节点查找器窗口

节点查找器对被查找的节点类型有过滤功能，要找到所有输入/输出节点，在 Filter 栏选择 Pins: all，接着点击按钮 List，所有输入/输出节点的名字便出现在节点查找器窗口的左边的方框（Nodes Found）中。单击节点 CLK，接着点击符号“≥”，使该节点加入到窗口右边的方框（Selected Nodes）中。以同样方法选择节点 CQ，直至所需要观察的信号均加入到右边方框，再点击 OK 按钮，关闭节点查找器窗口，返回 Insert Node or Bus 窗口，再单击 OK 按钮，回到波形编辑器窗口。

波形编辑器窗口分为左右两个子窗口，左边为信号区，右边为波形区。最左侧为波形编辑查看工具栏。单击信号区 CLK 信号，在工具栏中单击时钟设置按钮，打开时钟设置对话框，接受默认设置，单击 OK 按钮，则输入信号 CLK 的激励波形设置完毕，参见图 2.12。当所有输入节点的激励波形设置完毕后，保存激励波形文件。

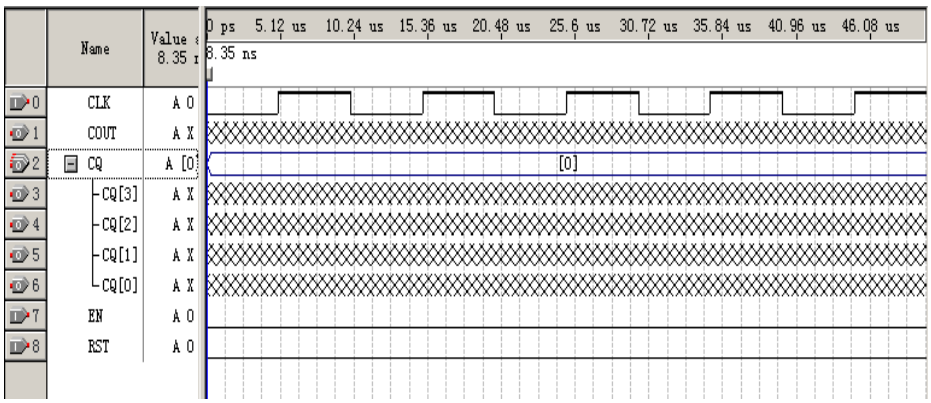


图 2.12　激励波形设置对话框

波形编辑工具栏各按钮的功能如表 2.2 所示。

图标	功能	图标	功能
	选择拾取按钮		文本按钮
	波形编辑按钮		缩放按钮
	全屏显示按钮		查找按钮
	替换按钮		设定选中波形为未初始化
	设定选中波形为未知电平		设定选中波形为低电平
	设定选中波形为高电平		设定选中波形为高阻状态
	设定选中波形为弱未知态		设定选中波形为弱低电平
	设定选中波形为弱高电平		设定选中波形为无关状态
	电平取反按钮		设定选中波形为计数脉冲
	时钟设置按钮		设定选中波形为任意指定值
	设定选中波形为随机值		设定波形对齐网格按钮
	设定波形排序按钮		

表 2.2　波形编辑工具栏各按钮功能说明

6、 功能仿真

选择 Processing 一栏的 Simulator Tools，打开仿真工具对话框，如图 2.13 所示。在 Simulation mode 栏中选择 Functional，即设为功能仿真。单击 Generate Functional Simulation Netlist 按钮，生成功能仿真网表。功能仿真网表生成后，将弹出成功提示信息框。单击 OK，关闭此信息框。

单击 Simulator Tools 对话框左下角 Start 按钮，开始功能仿真。仿真结束后，单击对话框右下角 Report 按钮，查看仿真结果。功能仿真网表生成后，也可关闭 Simulator Tools 对话框。单击上方工具栏仿真按钮  执行仿真，并查看仿真结果，参见图 2.14 。从仿真结果看，所综合的电路能正确实现一位十进制计数器的功能，证明了设计的正确性。

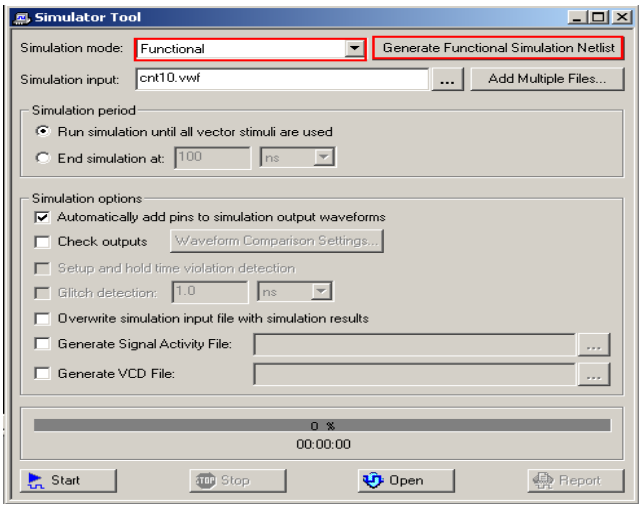


图 2.13　仿真设置对话框

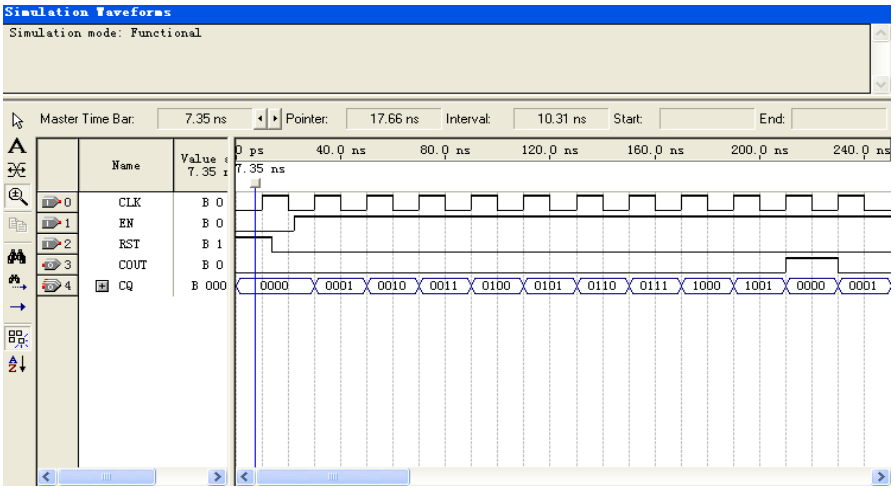


图 2.14　功能仿真结果

7、 引脚锁定

完成了功能仿真，在逻辑上实现了所需的功能电路，说明所设计的计数器是正确的。该计数器实体 CNT_10（图 2.8）有 5 组管脚 CLK、EN、RST、CQ 和 COUT，共 8 个。为使 CNT_10 这个实体配置到指定的 FPGA 芯片，需要将这 8 个管脚各自绑定到 FPGA 芯片上的一个管脚，这一过程称作“引脚锁定”。

在引脚锁定之前，首先要确定引脚锁定的方案，这与如何下载到实验平台上有关。本示例将设计结果下载到 TPC-IV 实验创新系统实验平台，计数时钟选 clock0（CLK），用按键 1 提供计数使能（En）信号，按键 2 提供复位清零（RST）信号，用数码管 1 显示计数结果（CQ[3]—CQ[0]），发光二极管 1 显示进位（COUT）。

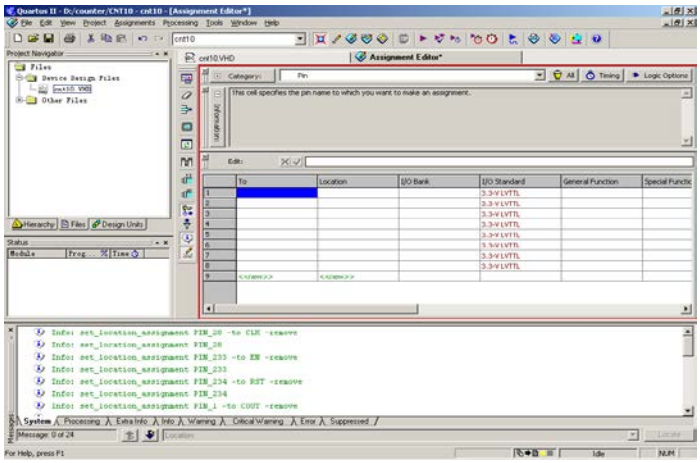


图 2.15　锁定引脚的界面

下面进行引脚锁定。首先选择 Assignments/Assignment Editor，将出现如图 2.15 所示的对话框。对于信号的输入，可以点击“To”这一列中的空格，直接输入名称即可，例如 CQ[3]。由于 CQ[3]（数码管）对应的是 FPGA 的引脚 16，在 Node Name 列，将会出现所有的引脚名，选择 CQ[3]所在的行，然后在 location 列单击鼠标选择引脚 16，如图 2.16 所示。

使用同样的方法可以完成全部的管脚锁定，并保存文件。重新点击 Start Compilation 按钮进行编译。编译结束后，工程目录下有一个与工程同名且后缀为.sof 的文件，这就是要用来下载的文件。事实上在第一次编译时也生成了这个文件，但由于之前没有管脚锁定，那时的管脚是随机分配的。注意，将没有进行管脚锁定的 sof 文件下载到 FPGA 是不允许的。

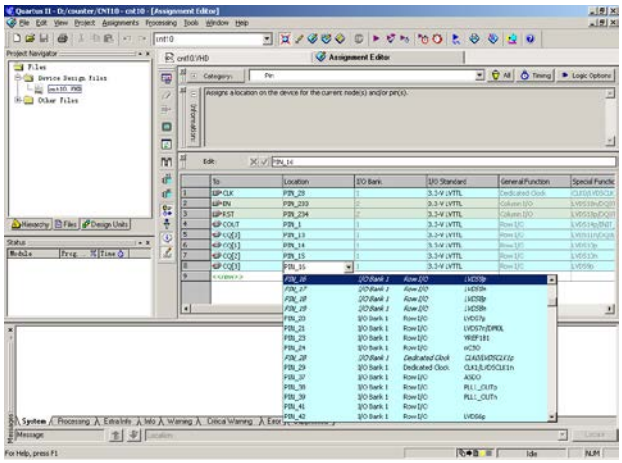


图 2.16　输入引脚名称

编译成功后，在主窗口显示的是编译报告。点击确定，就可以通过右面的窗口来观察资源占用情况。

如果编译过程中出现错误，Quartus II 会在下面的编译信息窗口中用红字显示出来，可以通过在错误信息上双击鼠标左键来定位错误。

8、 下载 FPGA

经过以上几步，得到了 FPGA 的下载文件，将 sof 文件下载到 FPGA 的操作步骤如下：

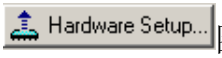
（1）用下载电缆连接主机和 TCP-IV 高级实验创新系统。

（2）打开 TCP-IV 高级实验创新系统电源，再按右下角复位建。

（3）在 Quartus II 中使用 Programmer 下载。

下面介绍一下 Quartus II 中 Programmer 的使用方法。

生成 sof 文件后，就可以使用 Quartus II 的 Programmer 进行下载。通过 Tools/Programmer 菜单命令进入 Programmer 对话框。第一次使用时，需要添加硬件，步骤如下：

点击 Programmer 对话框左上角的 

图标，弹出图 2.17 所示的对话框，在其中选择相应的硬件即可。如果 Available hardware items 当中没有选项，则点击上边 Currently selected hardware 右侧下箭头，选择 USB-Blaster[USB-0]即可，点击 Close 按钮关闭该窗口

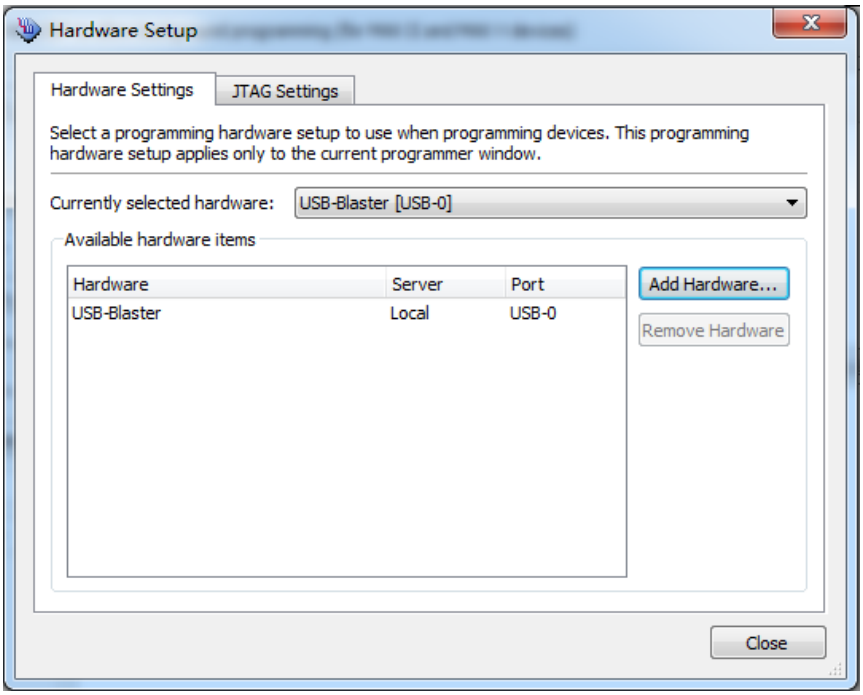


图 2.17 下载器 Hardware Setup 窗口

最后，添加要下载的 sof 文件。一般情况下，编译完成后，该文件会自动出现在 programmer 对话框中，如果没有出现，可以点击 Add Files 完成添加，然后在 Program/Configure 下的方框上打钩，点击 Start 开始下载。观察实验平台上的显示输出，以验证设计的正确性。

需要注意，如果重新编译了工程，在下载前必须先删掉上次加载的文件，然

后重新使用 Add File 添加下载文件。

三、LPM 宏功能模块应用

在实际应用中，必须通过宏功能模块才可以使用一些 Altera 公司特定器件的硬件功能，例如各类片上存储器、DSP 模块、LVDS 驱动器、嵌入式 PLL 电路模块等。设计者可以选择 LPM 库中的适当模块，并为其设定适当的参数，满足自己的设计要求，从而在自己的项目中十分方便地调用成熟的硬件设计成果。

在 Quartus II 中包含大量的 LPM 宏功能模块，每一模块的功能、参数含义、使用方法、硬件描述语言模块参数设置及调用方法都可以在 QuartusII 中的 Help 中查阅到，方法是选择“Help”菜单中的“Megafunctions/LPM”命令即可实现。

在 Quartus II 中调用 LPM 宏功能模块，可自动生成一些标准器件的 Verilog 代码。下面通过一个正弦信号发生器的设计示例介绍 LPM_ROM 的使用方法。

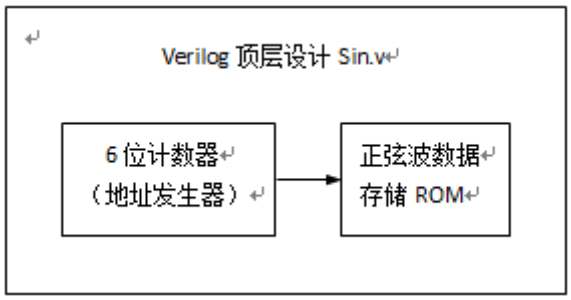


图 2.18　正弦信号发生器结构框图

图 2.18 所示的正弦信号发生器由以下两个部分组成：计数器或地址发生器（6 位）、正弦信号数据 ROM（6 位地址线，8 位数据线，即一个周期含有 64 个 8 位数据）。用 Verilog 完成顶层设计。图 2.18 所示的信号发生器结构图中，顶层文件 sin.v 在 FPGA 中实现，包含 ROM 的地址信号发生器（由 6 位计数器担任）、存贮正弦波数据的片内 ROM（由 LPM_ROM 模块构成）两部分。地址发生器的时钟 CLK 的输入频率 f0 与输出的正弦波频率 f 的关系是：f = f0 /64，64 为每周期的正弦波数据点数。

1、 定制初始化数据文件

首先建立 ROM 内的波形数据文件。

Quartus II 能接受的 LPM_ROM 中的初始化数据文件格式有 Memory Initialization File(.mif) 和 Hexadecimal（Intel－Format）File(.hex) 两种格式。下面以存储 64 点正弦波形数据的.mif 格式文件为例进行说明。

mif 文件是 FPGA 片内 ROM 的初始化文件。单击菜单 File/New/Memory File 项，选择 Memory Initialization File 项（见图 2.19），单击 OK。 确定 ROM

中数据数（Number of word）和字宽（Word size），单击 OK，打开 mif 文件。

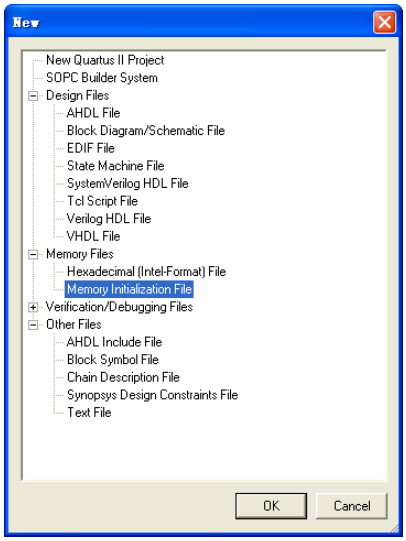


图 2.19　新建 mif 文件

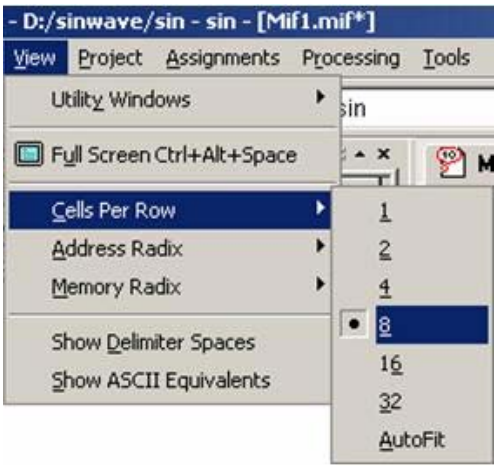


图 2.20 选择 mif 文件编辑窗口显示

通过 View/Cell Per Row 菜单设定每行显示 8 个存储单元（见图 2.21），这时 mif 文件编辑窗口如图 2.21 所示。在 mif 文件编辑窗口中填入正弦波数据，以文件名 sinwave.mif 保存到当前工程文件夹中，如图 2.22 所示。

Mif1.mif*

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	0	0	0	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
32	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0

图 2.21 mif 文件编辑窗口

sinwave.mif

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	128	140	153	165	177	188	199	209
8	219	227	235	241	246	250	253	255
16	255	254	252	248	244	238	231	223
24	214	204	194	183	171	159	147	134
32	121	108	96	84	72	61	51	41
40	32	24	17	11	7	3	1	0
48	0	2	5	9	14	20	28	36
56	46	56	67	78	90	102	115	127

图 2.22 将波形数据填入 mif 文件

2、 定制 ROM 元件

在设计正弦信号发生器前，必须首先完成存放波形数据 ROM 的设计。利用 MegaWizard Plug-In Manager 定制正弦信号数据 ROM 宏功能块，并将以上的波形数据文件 sinwave.mif 加载于此 ROM 中。设计步骤如下：

（1）通过菜单 Tools/MegaWizard Plug-In Manager 弹出对话框，选择第一项，即定制一个新的模块。单击 Next，打开图 2.23 所示对话框，在左栏选择 Memory Compiler 中的 ROM：1-PORT，再选择相应器件系列 Cyclone IV 和 Verilog 语言方式，最后输入 ROM 文件存放路径和文件名：D:\sinwave\drom.v，单击 Next。

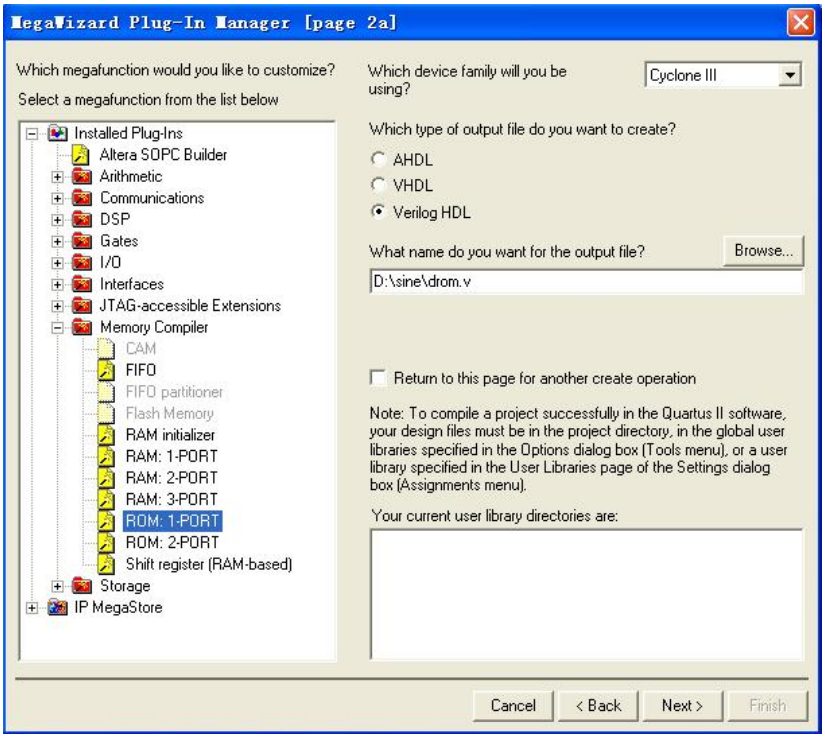


图 2.23 LPM 宏功能块设定

（2）选择 ROM 地址线、数据线和控制线。在如图 2.24 所示的对话框中选择地址线位宽和 ROM 中数据数分别为 6 和 64，默认地址锁存控制信号为 clock，单击 Next。在图 2.25 中，去掉输出锁存的勾选标记。

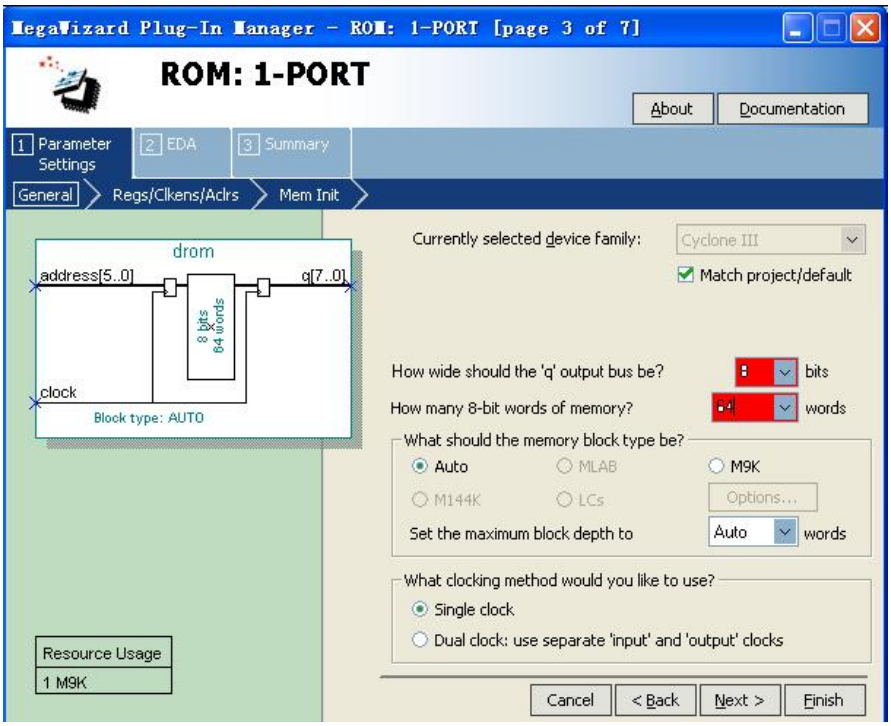


图 2.24　选择 ROM 模块数据线和地址线宽度

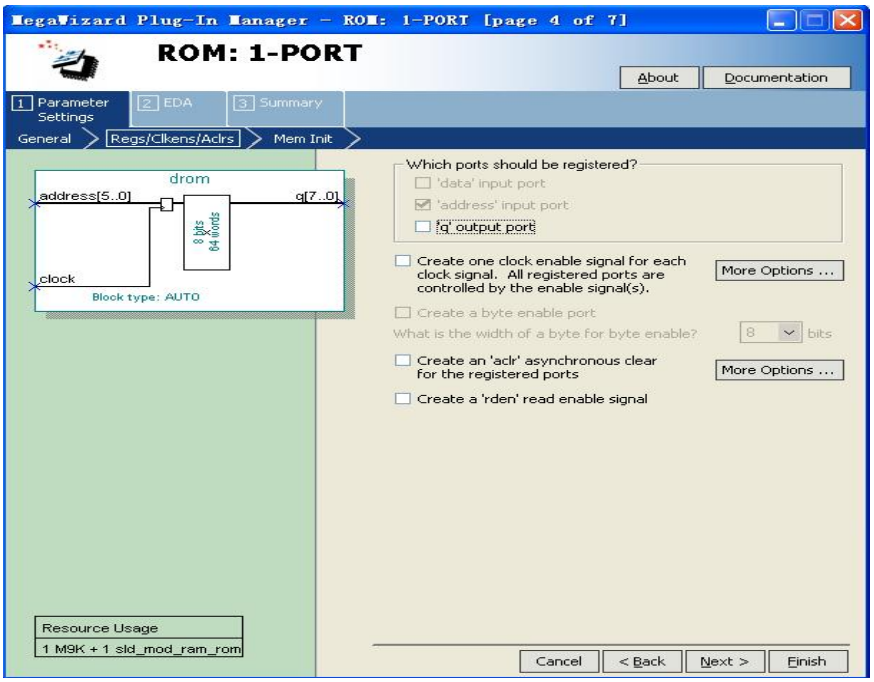


图 2.25　去除输出锁存信号 q

（3）指定 ROM 所对应的初始化文件 sinwave.mif。单击 Next 后，打开图 2.26 所示的对话框。在“Do you want to specify the initial content of the memory?”栏中选中“Yes，use this file for the memory content data”单选按钮，单击 Browse 按钮，选择指定路径上文件 sinwave.mif。

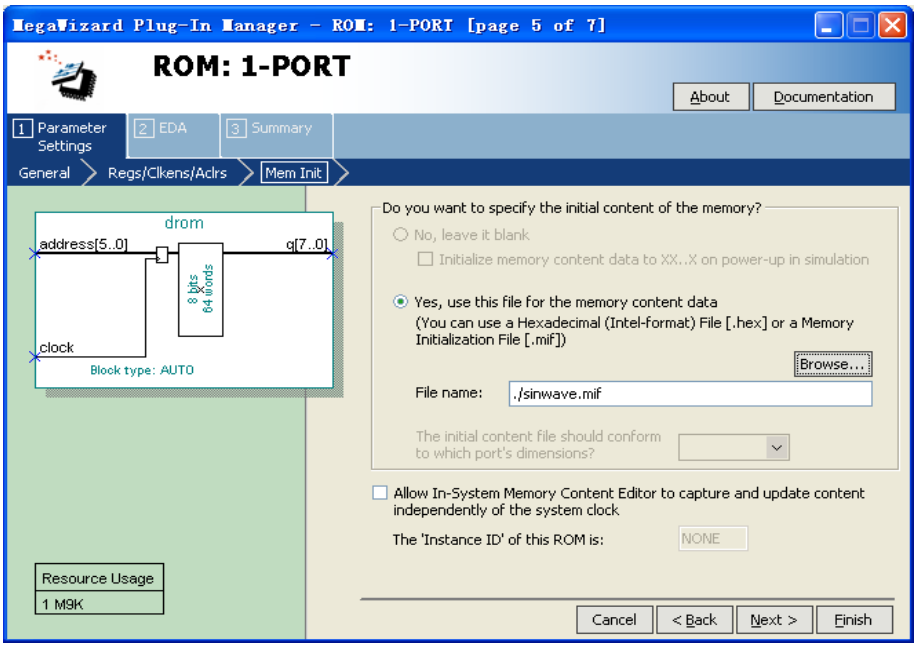


图 2.26　调入 ROM 初始化文件并选择在系统读写功能

建议在文件名对话框中将文件路径改为相对路径，如图 2.26 所示。选中“Alow In-System Memory...”栏复选框，并在“The Instance ID of this ROM is”栏键入“rom1”，作为此 ROM 的 ID 名称。通过这个设置，可在 In-System Memory Content Editor 窗口中查看编辑 ROM 中的内容。

（4）单击 Next 按钮两次，进入图 2.27 文件生成对话框，为便于搭建顶层原理图时调用 ROM 元件，建议勾选 rom1.bsf 选项，单击 Finish 按钮，完成 ROM 元件定制。

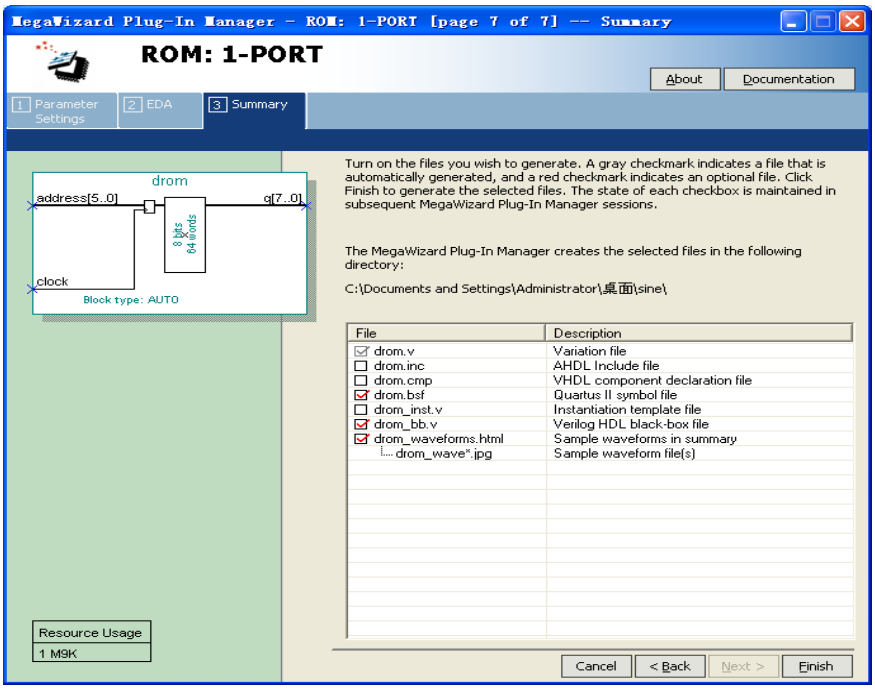


图 2.27　文件生成对话框

四、嵌入式逻辑分析仪 SignalTap II 的使用方法

SignalTap II 是集成于 Quartus II 中的一个软件工具，使用它可以在 FPGA 中形成嵌入式逻辑分析仪（Embedded Logic Analyzer，简称 ELA），用以捕捉目标芯片内部信号节点的信息，而不影响原硬件系统的正常工作。借助 SignalTap II 观察已编程器件中信号的实际波形，可以深入分析设计中是否存在缺陷。

假定原设计已经通过了编译和模拟验证，现在想加入 ELA 以观察某些信号的实际波形，其基本流程如图 2.28 所示。用户在.stp 文件中设置所需观察的信号节点名和触发条件，Quartus II 编译器在全程编译过程中，形成新的编程文件，其中包含原来的设计和根据.stp 文件生成的侦测逻辑 ELA。已被编程的 FPGA 器件在运行过程中，把采集的数据从 JTAG 口经 USB Blaster 送往主机，从而可以显示有关信号的实测波形。

ELA 将占用 FPGA 器件的资源，包括逻辑单元和存储单元，因此目标器件必须能够提供这些资源。设计者通过 SignalTap II 可选择通道数目和采样缓冲器的深度（0 到 128K 个采样），更有效地使用资源。

进入 SignalTap II 流程的前提是，该设计已通过编译和模拟验证，并且目标芯片已经安装在实验板上，通过下载线和主机相连。下面以项目名称为 sinwave 的正弦波发生器为例，介绍 SignalTap II 的使用方法，该正弦波发生器的 Verilog 代码如下。

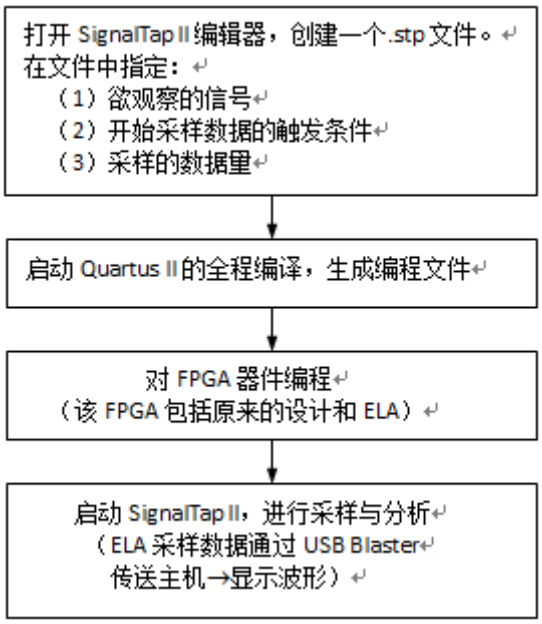


图 2.28　SignalTap 流程图

测试 FPGA 芯片中电路 sinwave 实际波形的步骤如下：

1、 打开 SignalTap II 编辑窗口

点击菜单 File/New，单击 Other Files，选择 SignalTap II File，再点击 OK，出现图 2.29 所示 SignalTap II 编辑窗口。

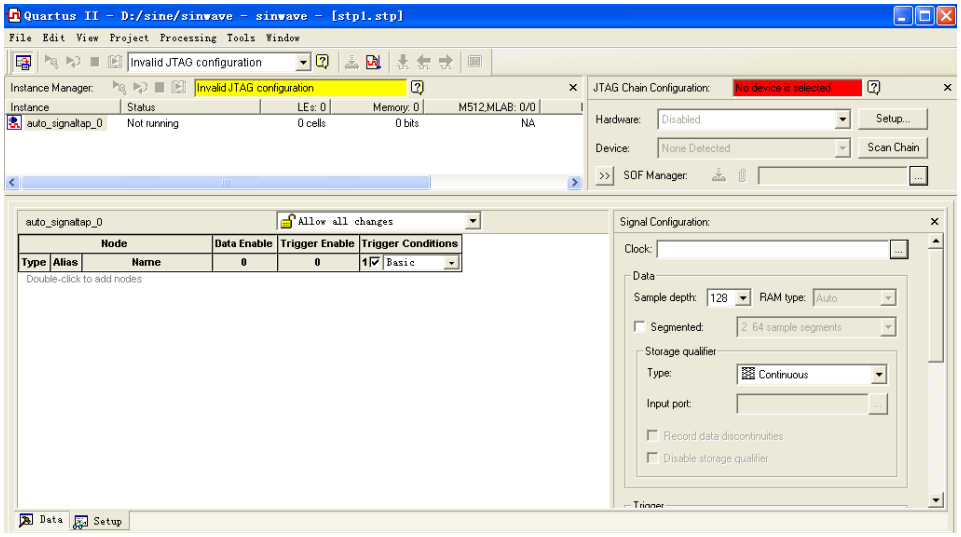


图 2.29　SignaTap II 编辑窗口

2、 调入电路的待测信号

在 SignalTapII 编辑窗口中，将 Instance 栏中的 auto_signaltap_0 重命名，把它改为本项目名 sinwave，于是图 2.29 变为图 2.30。在图 2.30 左下栏中的空白处双击鼠标左键，弹出 Node Finder 的窗口，单击 List 按钮可以得到相关的信号名，把需要观察的信号调入。注意调入信号的数量应根据实际需要来决定，不要调入过多的信号，以免占用过多 FPGA 芯片资源。

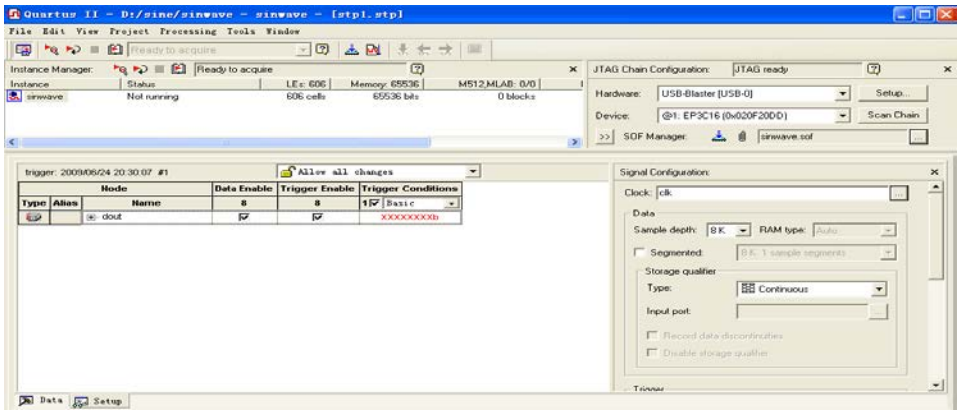


图 2.30　SignalTap II 参数设置

3、 设置 SignalTap II 参数

在 SignalTap II 编辑窗口中，单击左下角的 Setup 按钮，出现参数设置编辑窗口。

（1）设置 ELA 的采样时钟信号，该时钟可以是本设计的内部时钟，也可以是外部输入时钟。本例采用内部时钟 CLK。

（2）在 Sample depth 栏选择采样深度，例如 8K 位，表明所有的信号都采样 8K

个点。采样点数应根据需要和硬件资源消耗综合考虑确定。

（3）在 Buffer acquisition mode 选项区域选定 Circular，然后设定起始触发位置，比如选择前点触发“Pre trigger position”。

（4）选择采样的触发信号和触发方式，接受默认选项。

4、保存文件

选择 Save 以保存文件，如果出现“Do you want to enable SignalTap II…”，选择“是”就可以激活 SignalTap II，否则需要到图 2.31 所示窗口中进行设置。

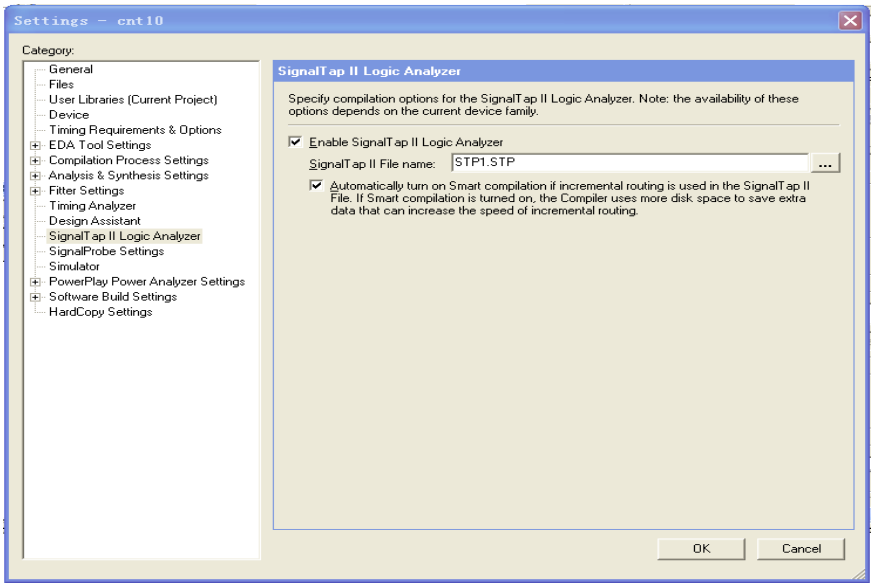


图 2.31　嵌入式逻辑分析仪设置

5、 编译并下载

启动全程编译，编译完成后，SignalTap II 的观察窗口会自动打开（若没有打开，可选择命令 Tools | SignalTap II Analyzer 将其打开，参见图 2.32）。然后将编译好的 sof 文件下载到 FPGA 器件。

图 2.32　启动 SignalTap II 采样

6、 启动 SignalTap II 进行采样与分析

单击 Autorun Analysis 按钮（图 2.32），启动 SignalTap II。点击左下角 data 页，这时就可以观察到来自实验板上 FPGA 内部的实测波形，如图 2.32 所示。

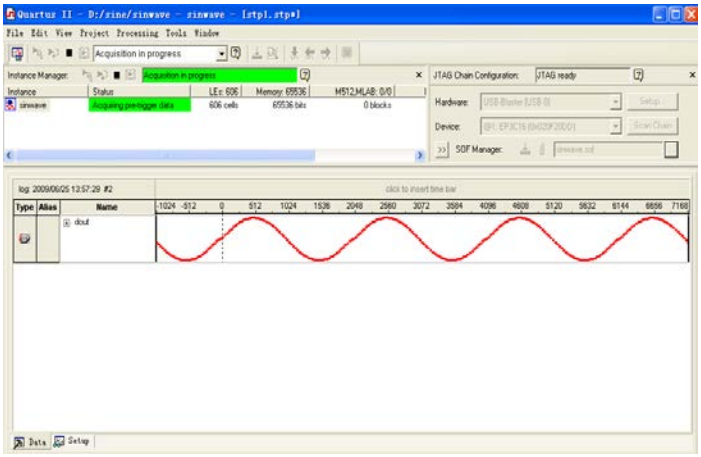


图 2.32　SignalTap II 实测波形

如果出现不了正弦波，鼠标点击 dout，按右键选择 Bus Display Format/Unsigned Line Chart，鼠标再点波形，点鼠标右键调节波形周期。如图 2.33 所示。

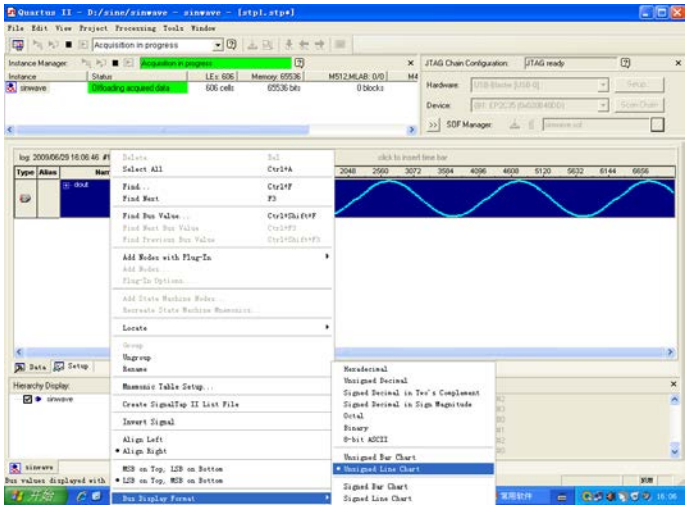


图 2.33　正弦波输出格式选择

五、FPGA 实验的一般过程与要求

FPGA 实验是一个综合性实验，涉及到硬件描述语言、软件开发工具、FPGA 开发系统和电路系统等多方面的知识。通过实验可以学习 FPGA 应用开发技术，熟悉用可编程逻辑器件设计数字电路和系统的基本方法，包括编程、设计输入、编译、仿真、下载和验证等，从而培养学生综合应用所学知识分析解决实际工程问题的能力。

完成一个 FPGA 应用实验一般过程和要求如下。

1、 设计准备

认真阅读实验指导书，深入理解实验题目所提出的任务与要求；查阅有关的

技术资料，学习相关的基本知识；分析开发系统的电路模式及输入输出资源，以选取合适的输入输出方案；对于综合实验任务，应画出设计原理图；编制出硬件描述语言程序。

2、 设计输入

将设计电路以开发工具所要求的形式输入到计算机中。Quartus II 软件允许用原理图形式、硬件描述语言的文本形式和混合输入方式进行设计输入。对于较复杂的电路系统，通常用硬件描述语言或调用宏功能模块的方式完成子模块的设计，并且通过 Quartus II 提供的转换工具，生成每一个子模块的电路符号，然后用原理图输入方式搭建顶层原理图。对于较复杂系统，应在顶层原理图中用文字标出每一模块名称，增加原理图的可读性。

在实际开发过程中，通常用 ModelSim 或 ActiveHDL 专用仿真工具进行 Verilog 仿真，验证 Verilog 程序是否满足设计要求。

3、 分析综合

设计输入完成后，利用开发工具提供的命令，对输入的源文件进行分析综合，生成电路网表。如果设计有误，则按所显示的错误信息，对源程序进行修改，最后综合出电路网表。这时可用 Quartus II 提供的 RTL Viewer 工具，观察所综合的电路。注意记录出现的错误及解决方法，以作为实验报告总结的基本材料。

4、 仿真

首先需建立激励波形文件，设置仿真类型。对于功能仿真，需生成功能仿真网表；对于时序仿真，需先完成引脚锁定，进行全编译。执行仿真，以波形图方式显示出仿真结果。如果仿真结果和原设计不符合，可检查修改原设计，再进行仿真，直至得到满意的结果为止，从而在理论上保证了下载后形成的电路符合设计要求。

5、 引脚锁定

根据预先拟定的输入和显示输出方案，对照电路结构图和引脚对照表进行引脚锁定。

6、 编译

通过编译完成电路的布局布线，并生成下载文件。实际上全编译包含了分析综合、适配、生成下载文件和时序仿真四个模块，编译过程耗时较长，如果出现错误，需按提示进行修改。

7、 下载

利用开发系统的资源，将编程配置文件从计算机下载到 FPGA 器件中，观察其输出，验证电路功能是否符合设计要求。

8、 硬件测试

根据实验要求决定是否进行硬件测试。硬件测试时，利用 SignalTap II 嵌入式逻辑分析仪，观察芯片中信号的实际波形，深入分析设计中是否存在缺陷。

9、撰写总结报告

FPGA 实验报告通常包含以下几部分：

（1）实验目的、实验要求。

（2）总体设计，包括总体设计思路、设计原理图、模块划分等。

（3）关键技术分析，包括分析设计中的主要技术要点和难点（例如协议实现、时序分析、设计优化、毛刺消除等）。

（4）波形仿真结果及简单说明。

（5）实验完成情况，实验演示说明。

（6）实验中遇到的问题及解决方法，包括对分析综合或编译过程中的错误、警告信息的分析。

（7）实验总结、体会。

实验报告还应提交实验程序，建议提交以下文件：Quartus II 工程文件（.qpf），VHDL 源文件（.vhd），原理图文件（.bdf），测试激励文件（.vwf），设置文件（.qsf），下载文件（.sof）。Verilog 程序应进行必要的注释，重点说明信号和进程的含义。

第三章　组合逻辑实验

实验一、3-8 译码器

一、实验目的

设计并实现一个 3-8 译码器，通过实验平台进行显示验证。

二、实验原理

通常的译码器，就是输入二进制编码，输出一路有效电平，类似选择电路。

译码器典型用途是实现存储器的地址译码、控制器中的指令译码、代码翻译、显示译码等。除此之外，还可用译码器实现各种组合逻辑功能。在 FPGA 实际应用中，译码器的功能就是把一种代码转换为另一种代码。常用的译码器有二进制译码器、二—十进制译码器和显示译码器三类。

3-8 译码器是常用的二进制译码器，其原理是对输入码（3 位码）进行译码。

3-8 译码器真值表如下所示：

输入	输出
A[2~0]	Y[7~0]
X X X	0 0 0 0 0 0 0 0
0 0 0	0 0 0 0 0 0 0 1
0 0 1	0 0 0 0 0 0 1 0
0 1 0	0 0 0 0 0 1 0 0
0 1 1	0 0 0 0 1 0 0 0
1 0 0	0 0 0 1 0 0 0 0
1 0 1	0 0 1 0 0 0 0 0
1 1 0	0 1 0 0 0 0 0 0
1 1 1	1 0 0 0 0 0 0 0

三、实验内容

1、编写 3-8 译码器的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译下载。

To	Assignment Name	Value	Enabled
keyin[2]	Location	PIN_F2	Yes
keyin[1]	Location	PIN_H2	Yes
keyin[0]	Location	PIN_H1	Yes
ledout[0]	Location	PIN_C19	Yes
ledout[1]	Location	PIN_D19	Yes
ledout[3]	Location	PIN_C17	Yes
ledout[2]	Location	PIN_D17	Yes
ledout[4]	Location	PIN_E16	Yes
ledout[5]	Location	PIN_G15	Yes
ledout[6]	Location	PIN_F15	Yes
ledout[7]	Location	PIN_D15	Yes

输入端口：KEYIN[2:0]分别对应 SW[2:0]。

输出端口：LEDOUT[7:0]分别对应 LED[7:0]。

4、拨动开关 SW2~0，观看 LED7~0 的变化。

四、参考程序

```
library ieee;
use ieee.std_logic_1164.all;
entity decoder_3_8 is
  port(
       keyin:in std_logic_vector(2 downto 0);
       ledout:out std_logic_vector(7 downto 0)
       );
end;
architecture decoder of decoder_3_8 is
 signal led:std_logic_vector(7 downto 0);
  begin
   process(keyin)
    begin
    case keyin is
        when "000"=>led<="00000001";
        when "001"=>led<="00000010";
        when "010"=>led<="00000100";
        when "011"=>led<="00001000";
        when "100"=>led<="00010000";
        when "101"=>led<="00100000";
        when "110"=>led<="01000000";
```

```
        when "111"=>led<="10000000";
        when others=>led<="00000000";
     end case;
    end process;
  ledout<=led;
end;
```

实验二、8-3 编码器

一、实验目的

设计并实现一个 8-3 编码器，通过实验平台进行显示。

二、实验原理

编码器的功能恰好与译码器相反，它是对输入信号按一定规律进行编排，使每组输出的二进制代码具有其特定的含义。目前经常使用的编码器有普通编码器和优先编码器。

常用的编码器有：4-2 编码器、8-3 编码器、16-4 编码器，下面我们用一个 8-3 编码器的设计来介绍编码器的设计方法。

8-3 编码器真知表如下所示：

输入	输出
0 0 0 0 0 0 0 0	X X X
0 0 0 0 0 0 0 1	0 0 0
0 0 0 0 0 0 1 0	0 0 1
0 0 0 0 0 1 0 0	0 1 0
0 0 0 0 1 0 0 0	0 1 1
0 0 0 1 0 0 0 0	1 0 0
0 0 1 0 0 0 0 0	1 0 1
0 1 0 0 0 0 0 0	1 1 0
1 0 0 0 0 0 0 0	1 1 1

三、实验内容

1、编写 8-3 编码器的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译下载。

To	Assignment Name	Value	Enabled
keyin[0]	Location	PIN_H1	Yes
keyin[1]	Location	PIN_H2	Yes
keyin[2]	Location	PIN_F2	Yes
keyin[3]	Location	PIN_F1	Yes
keyin[4]	Location	PIN_E1	Yes
keyin[5]	Location	PIN_D2	Yes
keyin[6]	Location	PIN_C2	Yes
keyin[7]	Location	PIN_C1	Yes
ledout[0]	Location	PIN_C19	Yes
ledout[1]	Location	PIN_D19	Yes
ledout[2]	Location	PIN_D17	Yes

输入端口：KEYIN[7:0]分别对应 SW[[7:0]

输出端口：LEDOUT[2:0]分别对应 LED[2:0]

4、拨动开关 SW7~0，观看 LED2~0 的变化。

四、参考程序

```
library ieee;
use ieee.std_logic_1164.all;
entity encoder_8_3 is
 port(
      keyin:in std_logic_vector(7 downto 0);
      ledout:out std_logic_vector(2 downto 0)
      );
end;
architecture encoder of encoder_8_3 is
  begin
    with keyin select
      ledout<="XXX" when "00000000",
              "000" when "00000001",
              "001" when "00000010",
              "010" when "00000100",
              "011" when "00001000",
              "100" when "00010000",
              "101" when "00100000",
              "110" when "01000000",
              "111" when "10000000",
              "ZZZ" when others;
```

```
end;
```

实验三、BCD 七段动态译码显示

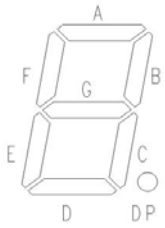
一、实验目的

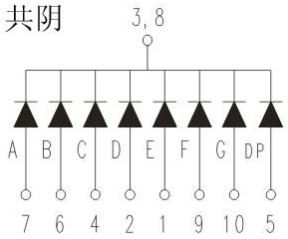
1、设计并实现一个 7 段动态扫描显示模块，通过实验平台完成 LED 数码管的动态显示。

2、掌握怎样分频从而实现慢速扫描。

二、实验原理

七段数码管是由 7 个独立的二极管采用共阴或共阳的方法连接而成。通常将这 7 个独立的二极管做成 a、b、c、d、e、f、g 这 7 个笔划，如下图所示。





由图可见，通过一个 7 位的二进制电平信号就可以显示出想要的结果。例如，点亮二极管 b、c，数码管将会显示数字 1，点亮 a、b、c、d、e、f、g，数码管将会显示数字 0。

七段共阴数码管真知表如下所示：

输入	输出
A[3:0]	Y[7~0]
0 0 0 0	0 0 1 1 1 1 1 1
0 0 0 1	0 0 0 0 0 1 1 0
0 0 1 0	0 1 0 1 1 0 1 1
0 0 1 1	0 1 0 0 1 1 1 1
0 1 0 0	0 1 1 0 0 1 1 0
0 1 0 1	0 1 1 0 1 1 0 1
0 1 1 0	1 1 1 1 1 1 0 1
0 1 1 1	0 0 0 0 0 1 1 1
1 0 0 0	0 1 1 1 1 1 1 1
1 0 0 1	0 1 1 0 1 1 1 1
1 0 1 0	0 1 1 1 0 1 1 1
1 0 1 1	0 1 1 1 1 1 0 0

1 1 0 0	0 0 1 1 1 0 0 1
1 1 0 1	0 1 0 1 1 1 1 0
1 1 1 0	0 1 1 1 1 0 0 1
1 1 1 1	0 1 1 1 0 0 0 1

通过一个时钟分频，使选通信号的扫描显示速率降低，从而观看到数码管的逐个变化。如果频率很高，扫描显示速率够快，由于人的视觉余辉效应，使我们仍会感觉所有的数码管都在同时显示。

动态扫描显示时刷新率最好大于 50HZ,每个数码管显示的时间不能太长也不能太短，太长会影响刷新率，导致总体显示呈现闪烁的现象，太短发光二极管的电流导通时间也就短，会影响总体的显示亮度。一般控制在 1ms 左右。

三、实验内容

1、编写 7 段动态显示译码器的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译下载。

To	Assignment Name	Value	Enabled
in clk	Location	PIN_G1	Yes
in swin[0]	Location	PIN_H1	Yes
in swin[1]	Location	PIN_H2	Yes
in swin[2]	Location	PIN_F2	Yes
in swin[3]	Location	PIN_F1	Yes
out hexout[0]	Location	PIN_J3	Yes
out hexout[1]	Location	PIN_J5	Yes
out hexout[2]	Location	PIN_G4	Yes
out hexout[3]	Location	PIN_J2	Yes
out hexout[4]	Location	PIN_G5	Yes
out hexout[5]	Location	PIN_J4	Yes
out hexout[6]	Location	PIN_G3	Yes
out hexout[7]	Location	PIN_J1	Yes
out hex_cnt[0]	Location	PIN_E3	Yes
out hex_cnt[1]	Location	PIN_J6	Yes
out hex_cnt[2]	Location	PIN_H6	Yes
out hex_cnt[3]	Location	PIN_H5	Yes
out hex_cnt[4]	Location	PIN_H7	Yes
out hex_cnt[5]	Location	PIN_H8	Yes

动态显示时钟：SEL_CLK=50M（CLK）/50K=1KHz

输入端口：SWIN[3:0]分别对应 SW[3:0]

输出端口：HEXOUT[7:0]分别对应数码管的段驱动端 LED7SEG_[DP:A]

HEX_CNT[5:0]分别对应数码管的位驱动端 LED7SEGDS[5:0]

4、设定输入，观看数码管 HEX0~HEX5 的输出变化。

四、参考程序

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity Hex7S_d is
  port(
       clk:in std_logic;
       swin:in std_logic_vector(3 downto 0);
       hexout:out std_logic_vector(7 downto 0);
       hex_cnt:out std_logic_vector(2 downto 0)
       );
end;
architecture Hex7S of Hex7S_d is
  signal div_cnt:integer range 0 to 49999999;
  signal sel_clk:std_logic;

   begin
    process(clk)
      begin
        if clk'event and clk='1' then
--          if div_cnt=49999999 then
            if div_cnt=49999 then
             div_cnt<=0;
             sel_clk<='1';
            else
             div_cnt<=div_cnt+1;
              sel_clk<='0';
          end if;
        end if;
    end process;
    process(sel_clk)
     variable sel_cnt:std_logic_vector(2 downto 0);
     begin
      if (sel_clk'event and sel_clk='1') then
         sel_cnt:=sel_cnt+1 ;
```

```
    end if;
    hex_cnt<=sel_cnt;
  end process;
  process(swin)
    begin
      case swin is
           when "0000"=>hexout<="00111111";
           when "0001"=>hexout<="00000110";
           when "0010"=>hexout<="01011011";
           when "0011"=>hexout<="01001111";
           when "0100"=>hexout<="01100110";
           when "0101"=>hexout<="01101101";
           when "0110"=>hexout<="01111101";
           when "0111"=>hexout<="00000111";
           when "1000"=>hexout<="01111111";
           when "1001"=>hexout<="01101111";
           when "1010"=>hexout<="01110111";
           when "1011"=>hexout<="01111100";
           when "1100"=>hexout<="00111001";
           when "1101"=>hexout<="01011110";
           when "1110"=>hexout<="01111001";
           when "1111"=>hexout<="01110001";
           when others=>hexout<="00000000";
        end case;
    end process;
end;
```

实验四、多路选择器

一、实验目的

设计并实现一个 1 位 4 选 1 的选择器，通过实验平台进行显示。

二、实验原理

数据选择器又叫“多路开关”，在地址码（或叫选择控制）电位的控制下，从几个数据输入中选择一个并将其送到一个公共的输出端，其功能类似一个多掷开关。数据选择器是目前逻辑设计中应用较为广泛的组合逻辑部件，常见电路有 2 选 1、4 选 1、8 选 1、16 选 1 等。

数据选择器除实现有选择的传送数据外，还可以产生逻辑函数、实现并行数据到串行数据的转换。

下面以 4 选 1 数据选择器为例进行设计。

4 选 1 真值表如下所示：

输入				选择控制	输出
a	b	c	d	0 0	a
a	b	c	d	0 1	b
a	b	c	d	1 0	c
a	b	c	d	1 1	d

三、实验内容

1、编写 1 位 4 选 1 选择器的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译下载。

To	Assignment Name	Value	Enabled
in en	Location	PIN_F13	Yes
in in0	Location	PIN_T14	Yes
in in1	Location	PIN_R14	Yes
in in2	Location	PIN_R16	Yes
in in3	Location	PIN_T15	Yes
out lout	Location	PIN_C19	Yes
in sel[1]	Location	PIN_H2	Yes
in sel[0]	Location	PIN_H1	Yes

输入端口：EN 对应 SW15　高使能

SEL[1:0]分别对应 SW[1:0]

IN3 对应 KEY4

IN2 对应 KEY3

IN1 对应 KEY2

IN0 对应 KEY1

输出端口：LOUT 对应 LED0

4、设定输入和选择控制端，观看输出。

四、参考程序

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity MUX_4 is
  port(
       en:in std_logic;
       in0:in std_logic;
       in1:in std_logic;
       in2:in std_logic;
       in3:in std_logic;
       sel:in std_logic_vector(1 downto 0);
       lout:out std_logic
       );
end;
architecture MUX of MUX_4 is
  begin
   process(en)
    begin
     if en='0' then
      lout<='0';
      else
        case sel is
          when "00"=>lout<=in0;
          when "01"=>lout<=in1;
          when "10"=>lout<=in2;
          when "11"=>lout<=in3;
          when others=>lout<='0';
        end case;
```

```
    end if;
  end process;
end;
```

实验五、数字比较器

一、实验目的

实现 1 位数值比较器，通过实验平台进行显示。

二、实验原理

用于比较两个数大小或相等的电路称为数值比较器。

1 位数值比较器是多位比较器的基础。这里就从 1 位数值比较器开始设计。

设计比较两个一位二进制数 A 和 B 大小的数字电路，输入变量是两个比较数 A 和 B，输出变量 Y(A＞B)、 Y(A＜B)、Y(A＝B)分别表示 A＞B、A＜B 和 A＝B 三种比较结果，1 位数值比较器真值表如下所示。

输入		输出		
A	B	y(A>B)	y(A<B)	y(A=B)
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

三、实验内容

1、编写 1 位数字比较器的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译下载。

To	Assignment Name	Value	Enabled
in a	Location	PIN_H1	Yes
in b	Location	PIN_H2	Yes
out c	Location	PIN_C19	Yes
out d	Location	PIN_D19	Yes
out e	Location	PIN_D17	Yes

输入端口：a,b 分别对应 SW0,SW1。

输出端口： c,d,e 分别对应 LED[0:2]。

a>b　LED0 灯亮。

a<b　LED1 灯亮。

a=b　LED2 灯亮。

4、拨动开关 SW1~0，观看 LED[0:2]的变化。

四、参考程序

```
library ieee;
use ieee.std_logic_1164.all;
entity COMPARE is
 port(
      a,b:in std_logic;
      c,d,e:out std_logic
      );
end;
architecture COMPA of COMPARE is
  begin
   process(a,b)
    begin
    if a>b then
       c<='1';d<='0';e<='0';
      elsif a<b then
         c<='0';d<='1';e<='0';
        elsif a=b then
          c<='0';d<='0';e<='1';
    end if;
   end process;
end;
```

实验六、四位加法器

一、实验目的

设计并实现一个 4 位加法器，通过实验平台进行显示验证。

二、实验原理

加法器即是产生数的和的装置。加数和被加数为输入，和数与进位为输出的装置为半加器。若加数、被加数与低位的进位数为输入，而和数与进位为输出则为全加器。常用作计算机算术逻辑部件，执行逻辑操作、移位与指令调用。

常用的有逐位进位加法器和超前进位加法器

列出 1 位全加器真值表，如下所示。

A	B	Cin	Sum	Cout
0	0	0	0	0
1	*0*	*0*	*1*	*0*
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

三、实验内容

1、编写 4 位加法器的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译下载。

To	Assignment Name	Value	Enabled
a[0]	Location	PIN_H1	Yes
a[1]	Location	PIN_H2	Yes
a[2]	Location	PIN_F2	Yes
a[3]	Location	PIN_F1	Yes
b[0]	Location	PIN_E1	Yes
b[1]	Location	PIN_D2	Yes
b[2]	Location	PIN_C2	Yes
b[3]	Location	PIN_C1	Yes
s[0]	Location	PIN_C19	Yes
s[1]	Location	PIN_D19	Yes
s[2]	Location	PIN_D17	Yes
s[3]	Location	PIN_C17	Yes
ci	Location	PIN_B2	Yes
gg	Location	PIN_E16	Yes
pp	Location	PIN_G15	Yes

输入端口：a[3:0]分别对应 SW[3:0]

b[3:0]分别对应 SW[7:4]

ci 对应 SW8

输出端口：s[3:0]分别对应 LED[3:0]

gg 对应 LED4（进位）

pp 对应 LED5（进位传递信号）

4、设定不同的输入，观看输出变化。

四、参考程序

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY ADDER_4 IS
   PORT (

      a    : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
      b    : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
      ci   : IN STD_LOGIC;

      s    : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
      pp   : OUT STD_LOGIC;
      gg   : OUT STD_LOGIC
   );
END ADDER_4;
```

```
ARCHITECTURE trans OF ADDER_4 IS

   SIGNAL p : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL g : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL c : STD_LOGIC_VECTOR(2 DOWNTO 0);
BEGIN

   p(0) <= a(0) XOR b(0);
   p(1) <= a(1) XOR b(1);
   p(2) <= a(2) XOR b(2);
   p(3) <= a(3) XOR b(3);

   g(0) <= a(0) AND b(0);
   g(1) <= a(1) AND b(1);
   g(2) <= a(2) AND b(2);
   g(3) <= a(3) AND b(3);

   c(0) <= (p(0) AND ci) OR g(0);
   c(1) <= (p(1) AND c(0)) OR g(1);
   c(2) <= (p(2) AND c(1)) OR g(2);

   pp <= p(3) AND p(2) AND p(1) AND p(0);
   gg <= g(3) OR (p(3) AND (g(2) OR (p(2) AND (g(1) OR (p(1) AND g(0))))));

   s(0) <= p(0) XOR ci;
   s(1) <= p(1) XOR c(0);
   s(2) <= p(2) XOR c(1);
   s(3) <= p(3) XOR c(2);

END trans;
```

实验七、四位加减法器

一、实验目的

设计并实现 4 位加、减法器，通过实验平台进行显示验证。

二、实验原理

根据运算符号信号的不同状态，既能够进行加法运算，又能够进行减法运算的数字电路叫做加、减法器。四位加、减法器是可以对两个四位二进制数进行加、减运算。

对四位加、减法器的设计是在上个实训的基础上进行，把四个全加器、加减控制信号用异或门组合到一起。

四位加、减法器的真值表如下所示。

A[3:0]	B[3:0]	ADD	SUB	S[4~0]
A	B	1	0	A+B
A	B	0	1	A-B

三、实验内容

1、编写 4 位加、减法器的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译下载。

To	Assignment Name	Value	Enabled
in a[3]	Location	PIN_F1	Yes
in a[2]	Location	PIN_F2	Yes
in a[1]	Location	PIN_H2	Yes
in a[0]	Location	PIN_H1	Yes
in b[3]	Location	PIN_C1	Yes
in b[2]	Location	PIN_C2	Yes
in b[1]	Location	PIN_D2	Yes
in b[0]	Location	PIN_E1	Yes
in add	Location	PIN_G7	Yes
in sub	Location	PIN_F13	Yes
out output[3]	Location	PIN_C17	Yes
out output[2]	Location	PIN_D17	Yes
out output[1]	Location	PIN_D19	Yes
out output[0]	Location	PIN_C19	Yes

输入端口：a[3:0]分别对应 SW[3:0]

b[3:0]分别对应 SW[7:4]

add 对应 SW14

Sub 对应 SW15

输出端口： output[3:0] 对应 LED3--LED0

4、设定不同的输入，观看输出变化。

四、参考程序

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity ADD_SUB_4 is
port(
     a:in std_logic_vector(3 downto 0);
     b:in std_logic_vector(3 downto 0);
     add:in std_logic;
     sub:in std_logic;
     output:out std_logic_vector(3 downto 0)
     );
end;
architecture ADD_SUB of ADD_SUB_4 is
  signal output1:std_logic_vector(3 downto 0);
    begin
      process(add,sub)
        begin
          if add='0' or sub='0' then
             output1<="0000";
             if add='1' then
              output1<=a+b;
               elsif sub='1' then
                 if a>b then
                    output1<=a-b;
                  else
                    output1<=b-a;
                 end if;
             end if;
          end if;
          output<=output1;
```

```
    end process;
end;
```

第四章　时序逻辑实验

实验八、D 触发器

一、实验目的

设计并实现一个 D 触发器，通过实验平台进行显示验证。

二、实验原理

能够储存 1 位二值信号的基本单元统称为触发器，触发器具有记忆功能，是构成各种时序电路的最基本逻辑单元。

最常用的四种触发器分别为：T 型触发器、RS 触发器、JK 触发器及 D 触发器。

D 触发器的状态方程 Q(n+1)=D(n)

D 触发器的特征表，如下所示。

D	Q(n)	Q(n+1)
0	0	0
0	1	0
1	0	1
1	1	1

三、实验内容

1、编写带异步清 0 端 D 触发器的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译适配。

To	Assignment Name	Value	Enabled
in clk	Location	PIN_G1	Yes
in clrb	Location	PIN_H2	Yes
in d	Location	PIN_H1	Yes
out q	Location	PIN_C19	Yes
out qb	Location	PIN_D19	Yes

时钟 clk=50MHZ

输入端口：CLRB 对应 SW1

D 对应 SW0

输出端口：Q　对应 LED0

QB 对应 LED1

4、下载，拨动开关 SW1~0，观看 LED1~0 的变化。

四、参考程序

```
library ieee;
use ieee.std_logic_1164.all;
entity D_FF is
port(
     clk:in std_logic;
      d:in std_logic;
     clrb:in std_logic;
     q:out std_logic;
     qb:out std_logic
     );
end;
architecture D_F of D_FF is
  signal qq:std_logic;
   begin
    process(clk)
     begin
      if clk='1' then
       if clrb='0' then
          q<='0';
          qb<='1';
         else
           q<=d;
           qb<=not d;
       end if;
      end if ;
    end process;
end;
```

实验九、寄存器

一、实验目的

设计并实现一个基本寄存器，通过实验平台进行显示验证。

二、实验原理

在数字电路中，用来存放二进制数据或代码的电路称为寄存器。

寄存器是由具有存储功能的触发器构成的。一个触发器可以存储一位二进制代码，存放 N 位二进制代码的寄存器，需用 n 个触发器来构成。

按功能可分为：基本寄存器和移位寄存器。

寄存器被广泛地应用于各类数字系统和数字计算机中，通过寄存器可以改善时序，消除抖动，设计流水线，还可利用 I/O 块中的寄存器对 FPGA 的输入、输出信号进行锁存，提高板级的设计。

三、实验内容

1、编写基本寄存器的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译适配。

To	Assignment Name	Value	Enabled
in clk	Location	PIN_G1	Yes
in reset	Location	PIN_E4	Yes
in d[3]	Location	PIN_F1	Yes
in d[2]	Location	PIN_F2	Yes
in d[1]	Location	PIN_H2	Yes
in d[0]	Location	PIN_H1	Yes
out q[3]	Location	PIN_C17	Yes
out q[2]	Location	PIN_D17	Yes
out q[1]	Location	PIN_D19	Yes
out q[0]	Location	PIN_C19	Yes

时钟 clk=50MHZ

输入端口：reset 对应核心板上 K5,低复位

d[3:0]分别对应 SW[3:0]

输出端口：q[3:0]分别对应 LED[3:0]

4、下载，设定 reset 状态，拨动开关 SW3~0，观看 LED3~0 的变化。

四、参考程序

```
library ieee;
use ieee.std_logic_1164.all;
```

```
entity REG is
port(
     clk:in std_logic;
     reset:in std_logic;
     d:in std_logic_vector(3 downto 0);
     q:out std_logic_vector(3 downto 0)
     );
end;
architecture RE of REG is
 signal q_temp:std_logic_vector(3 downto 0);
   begin
     process(clk)
       begin
         if reset='0' then
            q_temp<="0000";
           elsif clk='1' then
             q_temp<=d;
         end if;
     end process;
   q<=q_temp;
end;
```

实验十、移位寄存器

一、实验目的

设计并实现一个移位寄存器，通过实验平台进行显示验证。

二、实验原理

移位寄存器的功能是当时钟控制脉冲有效时，寄存器中存储的数码同时顺序向高位（左移）或向低位（右移）移位一位。所以，移位寄存器的各触发器状态必须同时变化，为同步时序电路。

因为数据可以按序逐位从最低位或最高位串行输入移位寄存器，也可以通过置数端并行输入移位寄存器。所以移位寄存器的数据输入、输出方式有并行输入/并行输出、并行输入/串行输出、串行输入/并行输出、串行输入/串行输出。

移位寄存器主要应用于实现数据传输方式的转换（串行到并行或并行到串行）、脉冲分配、序列信号产生以及构成移位寄存器型计数器。

三、实验内容

1、编写移位寄存器的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译适配。

To	Assignment Name	Value	Enabled
clk	Location	PIN_G1	Yes
clr	Location	PIN_F13	Yes
en	Location	PIN_G7	Yes
in1[0]	Location	PIN_H1	Yes
in1[1]	Location	PIN_H2	Yes
in1[2]	Location	PIN_F2	Yes
in1[3]	Location	PIN_F1	Yes
in1[4]	Location	PIN_E1	Yes
in1[5]	Location	PIN_D2	Yes
in1[6]	Location	PIN_C2	Yes
in1[7]	Location	PIN_C1	Yes
out1[0]	Location	PIN_C19	Yes
out1[1]	Location	PIN_D19	Yes
out1[2]	Location	PIN_D17	Yes
out1[3]	Location	PIN_C17	Yes
out1[4]	Location	PIN_E16	Yes
out1[5]	Location	PIN_G15	Yes
out1[6]	Location	PIN_F15	Yes
out1[7]	Location	PIN_D15	Yes
set[0]	Location	PIN_B2	Yes
set[1]	Location	PIN_B1	Yes
set[2]	Location	PIN_T2	Yes

时钟 clk=50MHZ。

输入端口：clr 对应 SW15，高清 0。

en 对应 SW14，低使能。

In1[7:0]分别对应 SW[7:0]。

set[2:0]分别对应 SW[10:8]。

输出端口：out1[7:0]分别对应 LED[7:0]。

4、下载，设定输入状态和移位个数，观看 LED7~0 的变化。

四、参考程序

```
library ieee;
use ieee.std_logic_1164.all;
entity shift_r is
port(
     clk:in std_logic;
     en: in std_logic;
     clr:in std_logic;
     set:in std_logic_vector(2 downto 0);
      in1:in std_logic_vector(7 downto 0);
     out1:out std_logic_vector(7 downto 0)
     );
end;
architecture shift of shift_r is
  signal out2:std_logic_vector(7 downto 0);
    begin
      process(clk,clr,en)
        begin
          if clk'event and clk='1' then
          out2<=in1;
            if clr='1' then
               out2<="00000000";
              elsif en='0' then
                case set is
                  when "000"=>out2<=in1;
                  when "001"=>out2<=in1(6 downto 0)& in1(7);
```

```
                when "010"=>out2<=in1(5 downto 0)& in1(7 downto 6);
                when "011"=>out2<=in1(4 downto 0)& in1(7 downto 5);
                when "100"=>out2<=in1(3 downto 0)& in1(7 downto 4);
                when "101"=>out2<=in1(2 downto 0)& in1(7 downto 3);
                when "110"=>out2<=in1(1 downto 0)& in1(7 downto 2);
                when "111"=>out2<=in1(0)& in1(7 downto 1);
                when others=>out2<="00000000";
            end case;
          end if;
        end if;
    end process;
  out1<=out2;
end;
```

实验十一、计数器

一、实验目的

设计并实现一个可逆计数器（即加减计数器），通过实验平台进行显示验证。

二、实验原理

数字系统中使用得最多的时序电路就算是计数器了。计数器不仅能用于对时钟脉冲计数，还可以用于分频、定时、产生节拍脉冲和脉冲序列以及进行数字运算等。

计数器按不同的标准可分为同步、异步，加法、减法、可逆，二进制、二—十进制、循环码计数器等。

三、实验内容

1、编写可逆计数器的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译适配。

To	Assignment Name	Value	Enabled
in clk	Location	PIN_G1	Yes
in clr	Location	PIN_F2	Yes
in en	Location	PIN_H2	Yes
out count[0]	Location	PIN_C19	Yes
out count[1]	Location	PIN_D19	Yes
out count[2]	Location	PIN_D17	Yes
out count[3]	Location	PIN_C17	Yes
in updown	Location	PIN_H1	Yes

计数时钟 Clk1=clk/50M=1HZ

输入端口：clr 对应 SW2，高清 0

en 对应 SW1，低使能

updown 对应 SW0，计数器加减选择

输出端口：count[3:0]分别对应 LED[3:0]

4、下载，设定输入状态，观看 LED3~0 的变化。

四、参考程序

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity COUNTER is
```

```
port(
     clk:in std_logic;
     clr:in std_logic;
     en :in std_logic;
     updown :in std_logic;
     count:out std_logic_vector(3 downto 0)
     );
end;

architecture COUNTE of COUNTER is
  signal clk1:std_logic;
  signal count1:std_logic_vector(3 downto 0);
  begin
    process(clk)
     variable cnt:integer range 0 to 49999999;
      begin
        if clk'event and clk='1' then
          if cnt=49999999 then
             cnt:=0;
             clk1<='1';
            else
              cnt:=cnt+1;
              clk1<='0';
          end if;
        end if;
    end process;
    process(clk1)
      begin
        if clk1'event and clk1='1' then
          if clr='1' or  en='1' then
             count1<="0000";
            elsif updown='1' then
                  count1<=count1+'1';
                else
```

```
                    count1<=count1-'1';
          end if;
        end if;
       count<=count1;
    end process;
end;
```

实验十二、分频器

一、实验目的

用计数器设计并实现时钟分频器，在实验平台通过 LED 灯进行显示验证。

二、实验原理

通过计数器统计时钟脉冲数，清 0，来实现时钟的分频。

三、实验内容

1、编写 1M 分频和 25M 分频的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译适配。

To	Assignment Name	Value	Enabled
in CLK	Location	PIN_G1	Yes
out CLK0	Location	PIN_C19	Yes
out CLK1	Location	PIN_D19	Yes

时钟 CLK=50MHZ

1M 分频时钟 CLK0=CLK/2/1M=25HZ

25M 分频时钟 CLK1=CLK/2/25M=1HZ

输入端口：CLK 对应 PIN_G1

输出端口：CLK0 对应 LED0

CLK1 对应 LED1

4、下载，设定输入状态，观看 LED3~0 的变化。

四、参考程序

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY FRE_D IS
   PORT (
      CLK       : IN STD_LOGIC;
      CLK0      : OUT STD_LOGIC;
      CLK1      : OUT STD_LOGIC
   );
END FRE_D;
```

```
ARCHITECTURE trans OF FRE_D IS
   SIGNAL ct0        : STD_LOGIC_VECTOR(19 DOWNTO 0);
   SIGNAL ct         : STD_LOGIC_VECTOR(24 DOWNTO 0);

   SIGNAL CLK0_reg : STD_LOGIC;
   SIGNAL CLK1_reg : STD_LOGIC;
BEGIN
   CLK0 <= CLK0_reg;
   CLK1 <= CLK1_reg;

   PROCESS (CLK)
   BEGIN
      IF (CLK'EVENT AND CLK = '1') THEN
         IF (ct0 = 1000000) THEN
            ct0 <= "00000000000000000000";
            CLK0_reg <= NOT(CLK0_reg);
         ELSE
            ct0 <= ct0 + 1;
         END IF;
      END IF;
   END PROCESS;

   PROCESS (CLK)
   BEGIN
      IF (CLK'EVENT AND CLK = '1') THEN
         IF (ct = 25000000) THEN
            ct <= "00000000000000000000000000";
            CLK1_reg <= NOT(CLK1_reg);
         ELSE
            ct <= ct + 1;
         END IF;
      END IF;
   END PROCESS;
```

```
END trans;
```

实验十三、串行数据检测器

一、实验目的

设计并实现一个串行数据检测器，在实验平台进行显示验证。

二、实验原理

通过一个输入开关实现脉冲来控制输入端是否输出。

三、实验内容

1、编写 1M 分频和 25M 分频的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译适配。

To	Assignment Name	Value	Enabled
in clk	Location	PIN_A11	Yes
in data	Location	PIN_H2	Yes
out y	Location	PIN_C19	Yes

输入端口：clk 对应 CLK_CHOOSE；

data 对应 SW1。

输出端口：y 对应发光二极管 LED0。

4、下载，设定输入状态，观看底板上发光二极管 LED0 的变化。

四、参考程序

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity seqdect is
  port(data,clk:in std_logic;
       y:out std_logic);
end seqdect;

architecture a of seqdect is
  type state is(s0,s1,s2,s3,s4,s5);
  signal p: state;
  signal n: state;
begin
```

```
s:process(clk)
begin
  if clk'event and clk='1' then
     p<=n;
  end if;
end process;

process(data,p)
begin
  case p is
    when s0=>if data='1' then
      n<=s1;
    else
      n<=s0;
    end if;
    y<='0';
    when s1=>if data='1' then
      n<=s2;
    else
      n<=s0;
    end if;
    y<='0';
    when s2=>if data='1' then
      n<=s3;
    else
      n<=s0;
    end if;
    y<='0';
    when s3=>if data='1' then
      n<=s4;
    else
      n<=s0;
    end if;
    y<='0';
```

```
      when s4=>if data='1' then
        n<=s5;
      else
        n<=s0;
        y<='0';
      end if;
      y<='0';
      when s5=>
        n<=s0;
        y<='1';
      when others=>null;
    end case;
  end process;
end a;
```

实验十四、Moore 状态机

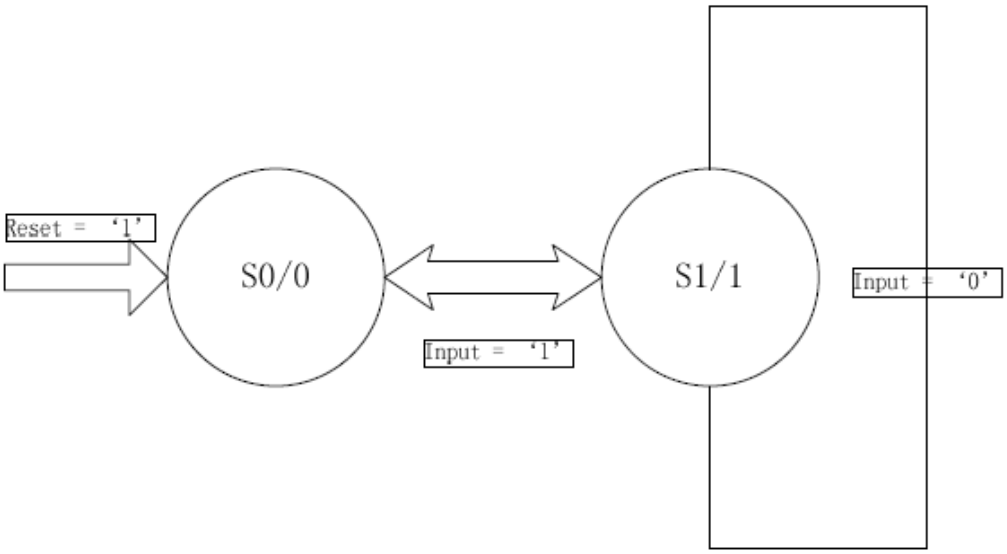
一、实验目的

设计并实现一个 Moore 状态机，通过实验平台进行显示验证。

二、实验原理

凡是输出与所处状态有关，而与输入信号无关的状态机类型，都可称为 Moore 状态机。

本实验实现一个简单的状态机，状态图如下图所示：



使用实验平台底板上的可选数字时钟源作为 clk 时钟输入，上升沿触发；由拨动开关 SW1 产生系统的复位信号，高电平复位；由拨动开关 SW0 产生输入信号，输出信号接到发光二极管 LED0。

三、实验内容

1、编写 Moore 状态机的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译适配。

To	Assignment Name	Value	Enabled
clk	Location	PIN_A11	Yes
input	Location	PIN_H1	Yes
output	Location	PIN_C19	Yes
reset	Location	PIN_H2	Yes

输入端口：clk 对应 CLK_CHOOSE；

input 对应 SW0；

reset 对应 SW1。

输出端口：output 对应发光二极管 LED0。

4、下载，设定输入状态，观看底板上发光二极管 LED0 的变化。

四、参考程序

```
library ieee;
use ieee.std_logic_1164.all;

entity statmach is
  port(clk : in bit;
       input : in bit;
       reset : in bit;
       output : out bit);
end statmach;

architecture a of statmach is
  type state_type is (s0,s1);
  signal st:state_type;
begin
  process(clk)
  begin
  if reset='1' then
     st<=s0;
  elsif (clk'event and clk='1')then
    case st is
      when s0=>
        st<=s1;
      when s1=>
        if input='1' then
        st<=s0;
        else
        st<=s1;
        end if;
    end case;
  end if;
```

```
  end process;
  output<='1' when st=s1 else '0';
end a;
```

实验十五、Mealy 状态机

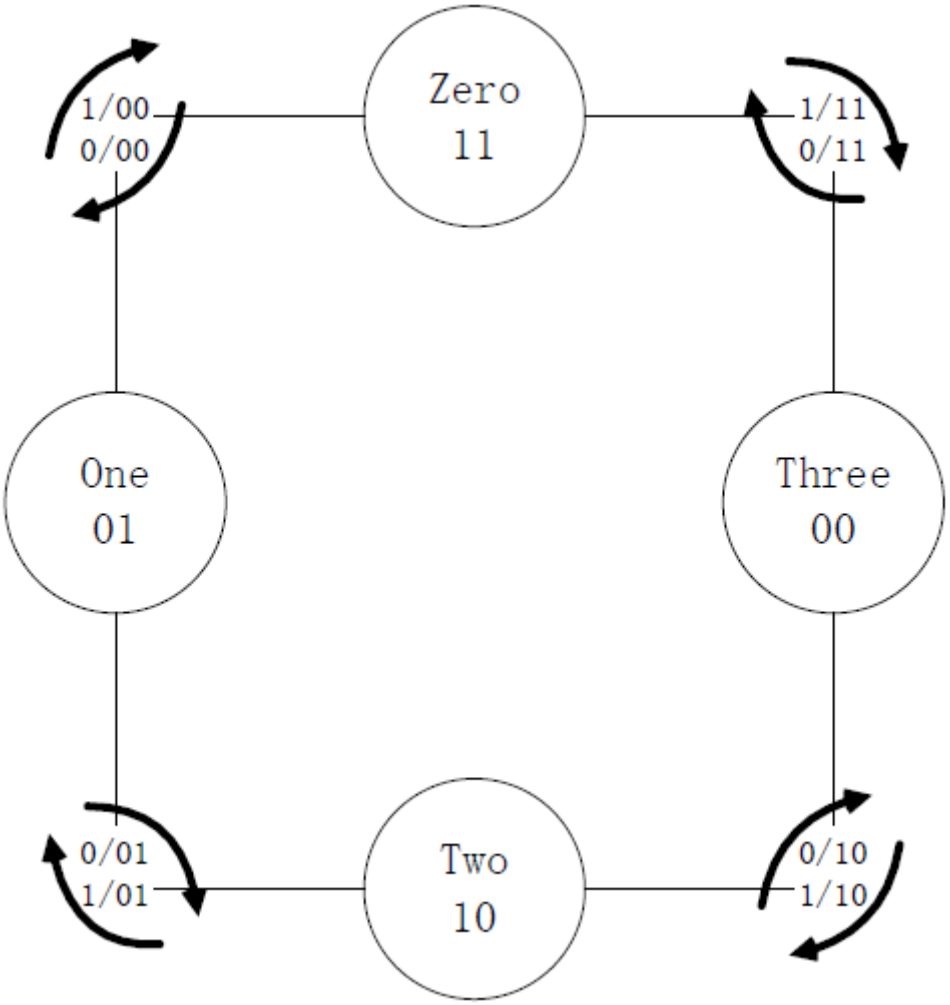
一、实验目的

设计一个 Mealy 状态机，实现计数器功能，通过实验平台进行显示验证。

二、实验原理

凡是输出与所处状态，以及输入信号有关的状态机类型，都可称为 Mealy 状态机。

使用实验平台底板上的可选数字时钟源作为 clk 时钟输入，上升沿触发；由拨动开关 SW0 的电平高低决定计数器是加计数还是减计数；输出信号连接到发光二极管 LED0、LED1。状态图如下图所示：



三、实验内容

1、编写 Mealy 状态机的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译适配。

To	Assignment Name	Value	Enabled
clk	Location	PIN_A11	Yes
updown	Location	PIN_H1	Yes
lsb	Location	PIN_C19	Yes
msb	Location	PIN_D19	Yes

输入端口：clk 对应 CLK_CHOOSE；

updown 对应 SW0。

输出端口：lsb 对应发光二极管 LED0；

msb 对应发光二极管 LED1。

4、下载，设定输入状态，观看底板上发光二极管 LED0、LED1 的变化。

四、参考程序

```
library ieee;
use ieee.std_logic_1164.all;

entity enumsmch is
  port(updown : in std_logic;
       clk : in std_logic;
      lsb,msb : out std_logic);
end enumsmch;

architecture a of enumsmch is
  type state is (zero,one,two,three);
  attribute enum_encoding:string;
  attribute enum_encoding of state:type is "11 01 10 00";
  signal p_state,n_state:state;
begin
  process (p_state,updown)
  begin
    case p_state is
        when zero =>
           if(updown='0')then
              n_state<=one;
              lsb<='0';
              msb<='0';
```

```
    else
       n_state<=three;
       lsb<='1';
       msb<='1';
    end if;
when one=>
    if(updown='0')then
       n_state<=two;
       lsb<='1';
       msb<='0';
    else
       n_state<=zero;
       lsb<='0';
       msb<='0';
    end if;
when two=>
    if(updown='0')then
       n_state<=three;
       lsb<='0';
       msb<='1';
    else
       n_state<=one;
       lsb<='1';
       msb<='0';
    end if;
when three=>
    if(updown='0')then
       n_state<=zero;
       lsb<='1';
       msb<='1';
    else
       n_state<=two;
       lsb<='0';
       msb<='1';
```

```
            end if;
    end case;
  end process;
  process
  begin
    wait until clk'event and clk='1';
    p_state<=n_state;
  end process;
end a;
```

第五章　外设应用实验

实验十六、VGA 显示控制器

一、实验目的

实现 1024x768@60Hz 分辨率的 VGA 图像显示。

二、实验原理

VGA（Video Graphics Array）即视频图形阵列，是 IBM 在 1987 年推出的使用模拟信号的一种视频传输标准，在当时具有分辨率高、显示速率快、颜色丰富等优点，在彩色显示器领域得到了广泛的应用。这个标准对于现今的个人电脑市场已经十分过时。即使如此，VGA 仍然是最多制造商所共同支持的一个标准，个人电脑在加载自己的独特驱动程序之前，都必须支持 VGA 的标准。例如，微软 Windows 系列产品的开机画面仍然使用 VGA 显示模式，这也说明其在显示标准中的重要性和兼容性。

VGA 最早指的是显示器 640X480 这种显示模式。而今天的 VGA 其实已经不仅仅局限于 640*480 这种分辨率了，通常情况下，各种各样适用于 VGA 接口传输的分辨率都可以统称为 VGA。

VGA 接口共有 15 针，分成 3 排，每排 5 个孔，显卡上应用最为广泛的接口类型，绝大多数显卡都带有此种接口。它传输红、绿、蓝模拟信号以及同步信号(水平和垂直信号)。本实验将介绍在 EDA/SOPC 实验平台上如何实现 VGA 图像的显示，在实验之前我们首先来了解 VGA 显示时序方面的一些知识。

1、VGA 时序标准

VGA 显示器扫描方式从屏幕左上角一点开始，从左像右逐点扫描，每扫描完一行,电子束回到屏幕的左边下一行的起始位置，在这期间，CRT 对电子束进行消隐，每行结束时，用行同步信号进行同步；当扫描完所有的行，形成一帧，用场同步信号进行场同步，并使扫描回到屏幕左上方，同时进行场消隐，开始下一帧。

完成一行扫描的时间称为水平扫描时间，其倒数称为行频率；完成一帧（整屏）扫描的时间称为垂直扫描时间，其倒数称为场频率，即刷新一屏的频率，常见的有 60Hz，75Hz 等等。标准的 VGA 显示的场频 60Hz。

时钟频率：以 1024x768@59.94Hz(60Hz)为例，每场对应 806 个行周期，其中 768 为显示行。每显示行包括 1344 点时钟,其中 1024 点为有效显示区。由

此可知：需要点时钟频率：806*1344*60 约 65MHz。

VGA 的行时序如图 16-1 所示：

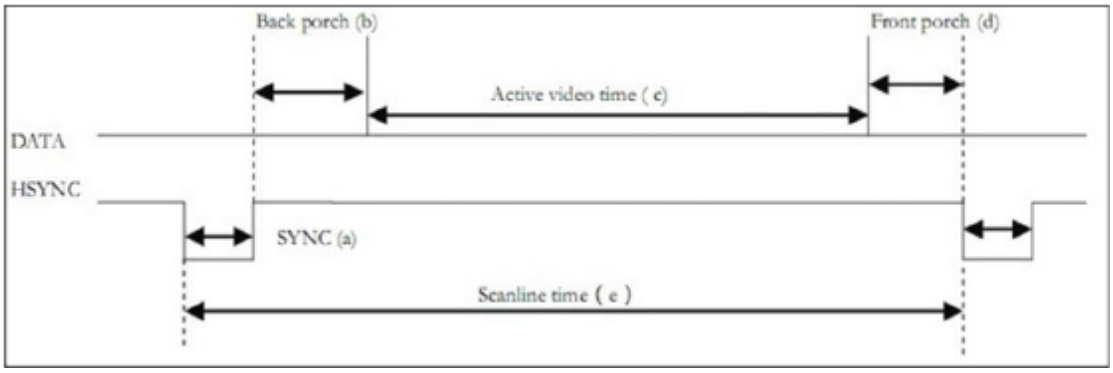


图 16-1 行时序

VGA的行场序如图16-2所示：

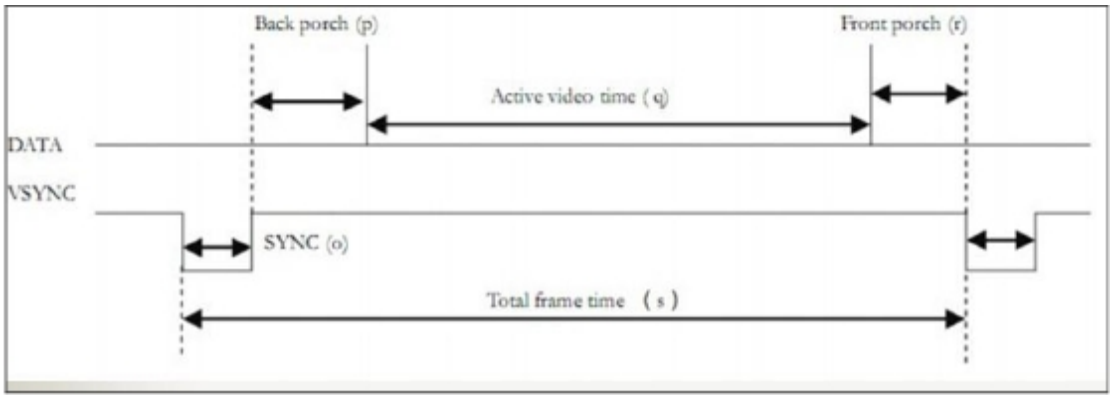


图 16-2 场时序

VESA 中定义行时序和场时序都需要同步脉冲（Sync a）、显示后沿（Back porch b）、显示时序段（Display interval c）和显示前沿（Front porch d）四部分。VGA 工业标准显示模式要求：行同步，场同步都为负极性，即同步脉冲要求是负脉冲。

下表以 1024x768@60Hz 分辨率的VGA时序参数:

1024 x 768 x 60Hz	a 段	b 段	c 段	d 段	e 段-总共 n 个列像素
HSYNC Signal 列像素	137	160	1024	24	1344
1024 x 768 x 60Hz	o 段	p 段	q 段	r 段	s 段-总共 n 个行像素
VSYNC Signal 行像素	6	29	768	3	806

2、硬件设计

本EDA/SOPC实验平台使用常用的 RGB565 数据输出，也就是 16 位的高彩色 VGA 显示，其中红色信号占 5 位，绿色信号占 6 位，蓝色信号占 5 位。电路

设计上通过电阻匹配网络实现 RGB 数字信号到模拟信号的转换。

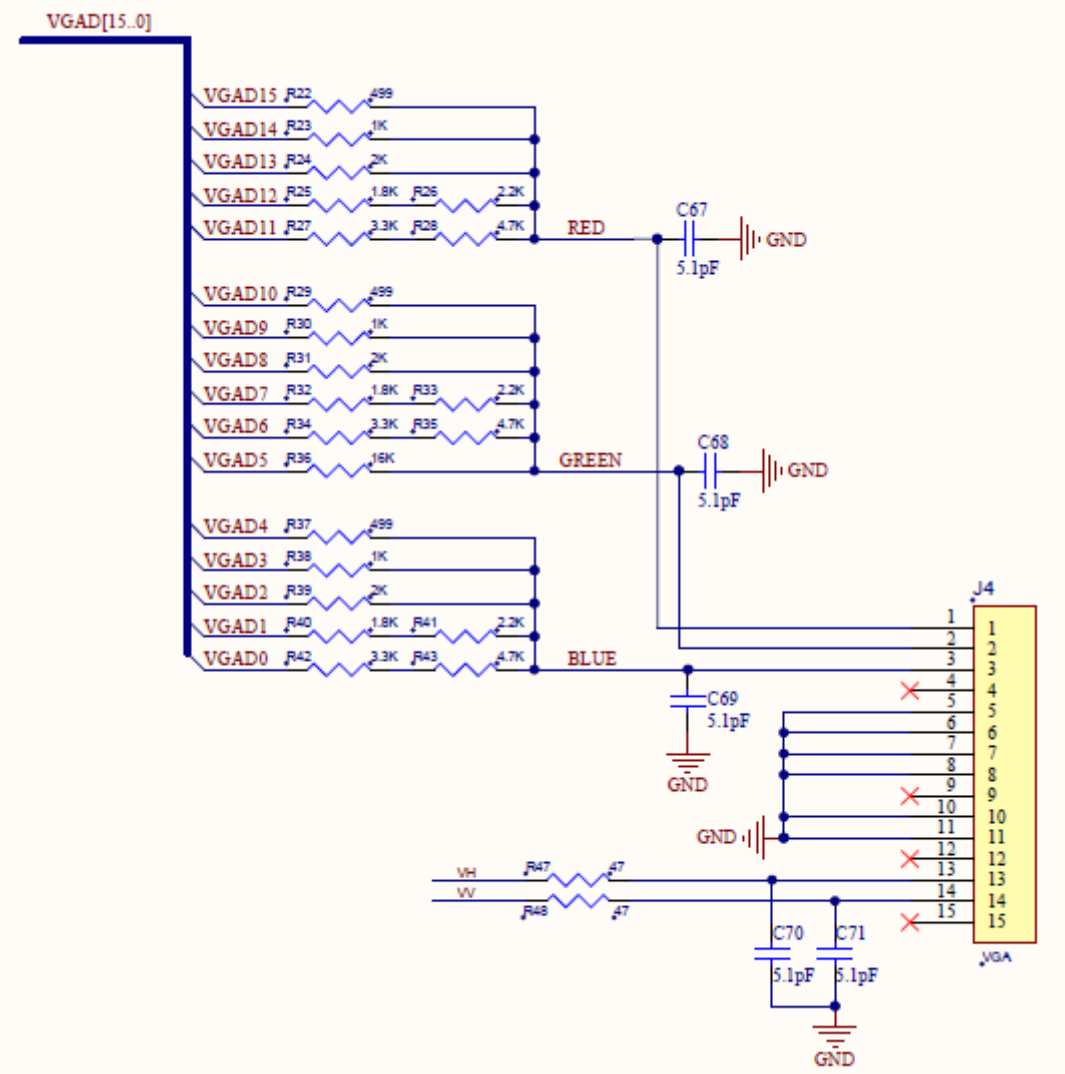


图 16-3 VGA接口电路

三、实验内容

1、新建工程，编写 VGA 显示控制器的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译适配。

To	Assignment Name	Value	Enabled
clk	Location	PIN_G1	Yes
clk_test	Location	PIN_AB13	Yes
key1	Location	PIN_T14	Yes
rstn	Location	PIN_E4	Yes
vga_b[0]	Location	PIN_K7	Yes
vga_b[1]	Location	PIN_K8	Yes
vga_b[2]	Location	PIN_L8	Yes
vga_b[3]	Location	PIN_M8	Yes
vga_b[4]	Location	PIN_R6	Yes
vga_g[0]	Location	PIN_Y3	Yes
vga_g[1]	Location	PIN_N8	Yes
vga_g[2]	Location	PIN_Y4	Yes
vga_g[3]	Location	PIN_V6	Yes
vga_g[4]	Location	PIN_W6	Yes
vga_g[5]	Location	PIN_Y6	Yes
vga_hs	Location	PIN_J7	Yes
vga_r[0]	Location	PIN_V7	Yes
vga_r[1]	Location	PIN_V5	Yes
vga_r[2]	Location	PIN_T7	Yes
vga_r[3]	Location	PIN_U7	Yes
vga_r[4]	Location	PIN_U9	Yes
vga_vs	Location	PIN_J8	Yes

时钟 clk=50MHz

输入端口：key1 对应轻触按键 KEY1

rstn 对应复位信号 nRESET

输出端口：vga_b[0:4]对应 VGA 接口的数据信号 VGAD[0:4]

vga_g[0:5]对应 VGA 接口的数据信号 VGAD[5:10]

vga_r[0:4]对应 VGA 接口的数据信号 VGAD[11:15]

vga_hs 对应 VGA 接口的行同步信号 VH

vga_vs 对应 VGA 接口的场同步信号 VV

4、下载，按下轻触按键 KEY1，观察 VGA 接口连接的显示屏的显示变化。

四、参考程序

略

实验十七、PS/2 键盘控制

一、实验目的

了解 PS/2 键盘工作原理，学会如何在 FPGA 内检测简单的键盘扫描码。

二、实验原理

PS/2设备履行一种双向同步串行协议，其接口的引脚定义如图17-1所示：

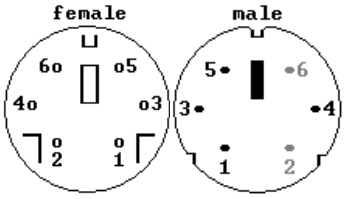


图 17-1　PS/2键盘接口

PS/2 接口中最重要的 4 根线是：1 脚的数据线、3 脚的地线、4 脚的电源线和 5 脚的时钟线。PS/2 接口的设备跟主机（本实验中 FPGA 可称为主机）间的通信主要是通过改变时钟线和数据线的状态来实现的。

完整的 PS/2 协议时比较复杂的，本实验中仅涉及到 PS/2 键盘跟主机间的通信，下面仅介绍一下本实验所用到的部分。

在键盘内部，有一专门负责扫描按键的处理器。它能检测出某个键被按下或者按下后被释放，并根据按键的类型长生相应的扫描码。键盘发送的扫描码一共有两种类型：通码(Make)和断码(Break)。当键盘上的一个键被按下时，键盘会根据按键类型产生一段通码；当键盘上的一个键按下后被松开时，键盘会根据按键类型再产生一段断码。此处理器为每个按键被分配了唯一的通码和断码，这样主机通过查找唯一的扫描码就可以测定是哪个键被按下或释放。

键盘的扫描码最终是要发送给主机的，这些扫描码包含在键盘发送给主机的数据帧中。每个数据帧包括： 1位起始位（总是低电平）、8位数据位（即扫描码，从低位开始发送）、1位奇偶校验位（奇校验）和1位结束位（总是高电平）。奇校验的原则是：如果扫描码中1的个数为奇数，该位取0；如果扫描码中1的个数为偶数，该位取1。图17-2是键盘发送扫描码给主机的过程。键盘总是产生时钟信号，从键盘发送到主机的数据在时钟信号的下降沿被读取。

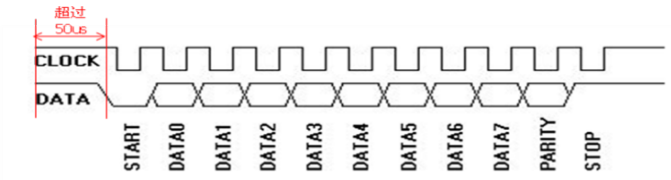


图 17-2　键盘发送扫描码给主机的过程

当键盘检测到有键按下或释放时，首先会判断主机不会给自己发送数据或命令，这是通过检测时钟线CLOCK持续高电平（没有被主机拉低）50us实现的。然后键盘开始给主机发送扫描码。键盘与主机间的通讯还包含很重要的一块，在本实验中并未涉及，就是主机发送数据给键盘。在后续的PS/2鼠标实验中会涉及到相关内容，在此，实验仅做一个铺垫，希望大家认真学习，为以后的实验打基础。表17-1列出了键盘上每个按键对应的扫描码。

KEY	MAKE	BREAK	KEY	MAKE	BREAK	KEY	MAKE	BREAK
A	1C	F0,1C	9	46	F0,46	[	54	F0,54
B	32	F0,32	`	0E	F0,0E	INSERT	E0,70	E0,F0,70
C	21	F0,21	-	4E	F0,4E	HOME	E0,6C	E0,F0,6C
D	23	F0,23	=	55	F0,55	PG UP	E0,7D	E0,F0,7D
E	24	F0,24	\	5D	F0,5D	DELETE	E0,71	E0,F0,71
F	2B	F0,2B	BKSP	66	F0,66	END	E0,69	E0,F0,69
G	34	F0,34	SPACE	29	F0,29	PG DN	E0,7A	E0,F0,7A
H	33	F0,33	TAB	0D	F0,0D	U ARROW	E0,75	E0,F0,75
I	43	F0,43	CAPS	58	F0,58	L ARROW	E0,6B	E0,F0,6B
J	3B	F0,3B	L SHFT	12	F0,12	D ARROW	E0,72	E0,F0,72
K	42	F0,42	L CTRL	14	F0,14	R ARROW	E0,74	E0,F0,74
L	4B	F0,4B	L GUI	E0,1F	E0,F0,1F	NUM	77	F0,77
M	3A	F0,3A	L ALT	11	F0,11	KP /	E0,4A	E0,F0,4A
N	31	F0,31	R SHFT	59	F0,59	KP *	7C	F0,7C
O	44	F0,44	R CTRL	E0,14	E0,F0,14	KP -	7B	F0,7B
P	4D	F0,4D	R GUI	E0,27	E0,F0,27	KP +	79	F0,79
Q	15	F0,15	R ALT	E0,11	E0,F0,11	KP EN	E0,5A	E0,F0,5A
R	2D	F0,2D	APPS	E0,2F	E0,F0,2F	KP .	71	F0,71
S	1B	F0,1B	ENTER	5A	F0,5A	KP 0	70	F0,70
T	2C	F0,2C	ESC	76	F0,76	KP 1	69	F0,69
U	3C	F0,3C	F1	05	F0,05	KP 2	72	F0,72
V	2A	F0,2A	F2	06	F0,06	KP 3	7A	F0,7A
W	1D	F0,1D	F3	04	F0,04	KP 4	6B	F0,6B
X	22	F0,22	F4	0C	F0,0C	KP 5	73	F0,73
Y	35	F0,35	F5	03	F0,03	KP 6	74	F0,74
Z	1A	F0,1A	F6	0B	F0,0B	KP 7	6C	F0,6C
0	45	F0,45	F7	83	F0,83	KP 8	75	F0,75
1	16	F0,16	F8	0A	F0,0A	KP 9	7D	F0,7D
2	1E	F0,1E	F9	01	F0,01	]	5B	F0,5B
3	26	F0,26	F10	09	F0,09	;	4C	F0,4C
4	25	F0,25	F11	78	F0,78	'	52	F0,52
5	2E	F0,2E	F12	07	F0,07	,	41	F0,41
6	36	F0,36	PRNT SCRN	E0,12, E0,7C	E0,F0, 7C,E0, F0,12	.	49	F0,49
7	3D	F0,3D	SCROLL	7E	F0,7E	/	4A	F0,4A
8	3E	F0,3E	PAUSE	E1,14,77, E1,F0,14, F0,77	-NONE-			

表17-1 键盘上每个按键对应的扫描码

三、实验内容

1、新建工程，编写 FPGA 接收来自 PS/2 键盘的扫描码的 VHDL 程序，要求对不同的按键做出响应。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译适配。

To	Assignment Name	Value	Enabled
KBCLK	Location	PIN_A13	Yes
KBdata	Location	PIN_B14	Yes
NUM[6]	Location	PIN_G3	Yes
NUM[5]	Location	PIN_J4	Yes
NUM[4]	Location	PIN_G5	Yes
NUM[3]	Location	PIN_J2	Yes
NUM[2]	Location	PIN_G4	Yes
NUM[1]	Location	PIN_J5	Yes
NUM[0]	Location	PIN_J3	Yes
rst	Location	PIN_F13	Yes
SEL[0]	Location	PIN_E3	Yes
SEL[1]	Location	PIN_J6	Yes
SEL[2]	Location	PIN_H6	Yes
SEL[3]	Location	PIN_H5	Yes
SEL[4]	Location	PIN_H7	Yes
SEL[5]	Location	PIN_H8	Yes

输入端口：rst 对应拨动开关 SW15

KBCLK 对应 PS/2 的时钟信号 PS2_A_CLK

KBdata 对应 PS/2 的数据信号 PS2_A_DAT

输出端口：NUM[0:6]对应数码管的段驱动信号 LED7SEG_[A:G]

SEL[0:5]对应数码管的位选择信号 LED7SEGDS[0:5]

4、下载，按下 PS/2 键盘上的按键，观察数码管的显示变化。

四、参考程序

略

实验十八、PS/2 鼠标接口控制

一、实验目的

了解 PS/2 鼠标协议，实现 PS/2 鼠标驱动，控制 VGA 屏幕的不同颜色显示。

二、实验原理

1、PS/2鼠标的引脚定义

PS/2鼠标设备接口的引脚定义如图18-1所示：

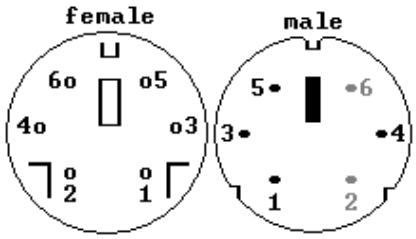


图 18-1　PS/2鼠标接口

PS/2 各个引脚的定义为：1 脚为数据输入、输出；2 脚未定义；3 脚为 GND ；4 脚为 VCC；5 脚为鼠标时钟 ；6 脚未定义 。

对于在实验平台上的编程来说，只需要关注1脚（数据输入输出）和5脚（鼠标时钟）就可以了。

2、PS/2鼠标的通信协议

PS/2设备履行一种双向同步串行协议，该协议支持以下传输模式：a） 鼠标——〉主机；b） 主机——〉鼠标 。主机可以是计算机、实验板等，这里我们特指FPGA实验平台。

（1）主机向鼠标发送数据

主机以数据帧的方式向鼠标发送数据，数据帧结构如图18-2所示：

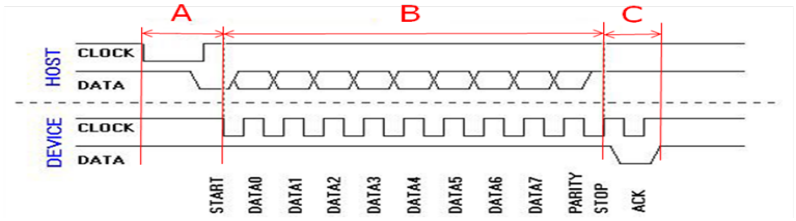


图 18-2　主机向鼠标发送数据帧的结构

为理解方便，将主机对鼠标的通讯分为 A,B,C 三段：

A 段：主机通知鼠标准备接收信息。主机端拉低时钟信号超过 100us，然后释放时钟线并同时将数据线拉低（发送起始位，起始位始终为 0），鼠标端在检测到以上变化后接管时钟线，开始发送时钟信号，并准备接受来自主机的数据

B 段：主机发送数据信息。主机端在鼠标发送的时钟信号为低电平时改变数

据线的电平向鼠标发送数据。B 段发送的数据为 8 位数位(低位在前)，1 位奇校验位及 1 位停止位（停止位为“1”)，鼠标端在时钟信号为高电平时采样数据线获得主机端发送的数据信息。主机在发送完成停止位后应在下一时钟上升沿前释放数据线。

C 段：鼠标应答主机的通信。鼠标端在接收完主机发送的停止位后，将发送 1 位应答位（“0”）通知主机已成功收到数据数据帧（因此再次之前主机必须释放数据线)。主机可以通过采样应答位来确认数据帧的传输，如果没有收到应答位，主机应需重发数据帧。

整个数据帧的结构为1位起始位（0)，8位数位(低位在前)，1位奇校验位及1位停止位（停止位为“1”）及1位鼠标应答位（0)，共12位。

（2）鼠标向主机发送数据

鼠标以数据帧的格式向主机放送数据，该数据帧的结构如图 18-3 所示。

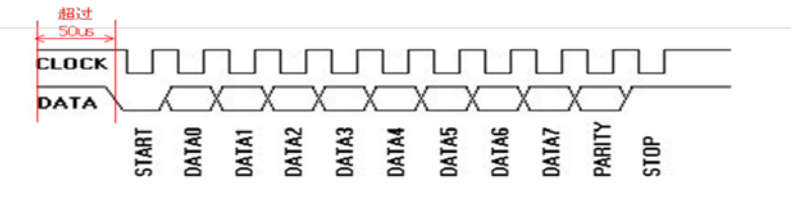


图18-3　鼠标向主机发送数据帧的结构

鼠标检测到时钟线高并维持超过 50us 则认定主机不会向鼠标发送数据，鼠标随即接管数据线并发出时钟信号向主机发送数据(如果鼠标确实有数据要发送的话)。在鼠标向主机发送数据帧的过程中，主机可以随时通过强行拉低时钟线抑制鼠标的通信(鼠标将会立即停止数据的发送,不管该数据帧是否发送完成)。

鼠标发送的数据帧有 11 位包括：1 位起始位（总为 0)；8 位数据位（最低位在先)；1 位奇偶校验位（奇校验)；1 位结束位（总为 1)。

鼠标在时钟高电平时发送数据，数据在时钟低电平时达到稳定状态。因此主机应在时钟信号为低时对数据线进行采样，以获得正确的数据。

3、鼠标通信流程

鼠标通信流程如图18-4所示：

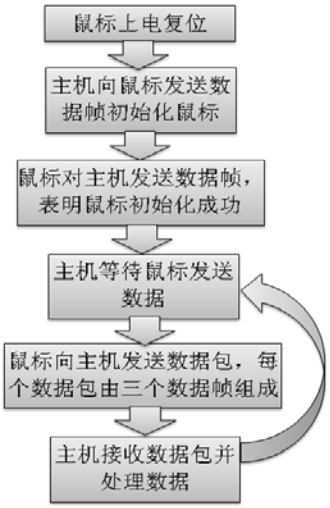


图18-4　鼠标通信流程

（1）鼠标上电复位：

鼠标在上电后（通过接入 PS/2 接口上电）自动复位，并发送设备信息。复位操作完成后鼠标自动进入 Stream 模式，该模式在默认状态下不发送数据包（鼠标的其他工作模式在此不做详述）。

（2）主机发送初始信息：

主机向鼠标发送初始化数据 0xF4h（以上文介绍过的数据帧格式）。

（3）鼠标向主机发送数据表明其初始化完成：

鼠标复位内部的位移计数器后发送数据 0xFAh （以数据帧格式），应答主机，之后鼠标会在移动或按键操作时向主机发送数据包。主机应接收并校验鼠标发送的数据帧 0xFAh，以确保鼠标已经完成了初始化，如果鼠标的应答数据不正确，主机应重新向鼠标发送初始化数据 0xF4h 直到鼠标正确返回数据帧 0xFAh 为止。

（4）主机等待鼠标发送数据：

鼠标在 Stream 模式下完成初始化后会在其位置发生变化或发生按键动作时向主机发送数据包（每个数据包由三个数据帧组成，数据包的内容将在下文中详述）。因此主机在确认鼠标完成初始化（返回 0xFAh）后，应检测时钟线，在时钟到来时同步采样数据线接收鼠标发送的数据包。

（5）主机处理来自鼠标的数据包：

主机接收完成数据包后要对数据包进行奇偶校验，在校验正确的情况下提取数据包内的数据信息（数据包错误时，主机应忽略该数据包，并等待下一数据包）。

4、 PS/2鼠标向主机发送的数据包

标准鼠标数据包如图 18-5 所示，由三个比特的数据组成，每个比特以一个上文介绍的数据帧的形式发送（即构成数据帧中的 8 个数据位），即一个数据包由三个数据帧组成。数据包包括鼠标的位移情况，左、中、右键的点击信息。

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	Y overflow	X overflow	Y sign bit	X sign bit	Always 1	Middle Btn	Right Btn	Left Btn
Byte 2	X Movement							
Byte 3	Y Movement							

图18-5　标准鼠标数据包

X movement 为上一个数据包发送后鼠标在 X 方向上的位置变化绝对值。X sign bit 为上一个发送后鼠标在 X 方向上的位置变化值的符号。Y 方向上的变化类似。Middle Btn、 Right Btn、 Left Btn 记录了鼠标中、右、左三键的点击情况。Y overflow, X overflow 记录 Y movement 、X movement 是否有溢出。

PS/2 时钟信号存在毛刺，应对其进行滤波操作，以免影响接口电路的稳定性。

三、实验内容

1、新建工程，编写 PS/2 鼠标控制 VGA 显示的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译适配。

To	Assignment Name	Value	Enabled
m_data	Location	PIN_B14	Yes
m_clk	Location	PIN_A13	Yes
clk	Location	PIN_G1	Yes
reset	Location	PIN_F13	Yes
hs	Location	PIN_J7	Yes
vs	Location	PIN_J8	Yes
r[9]	Location	PIN_U9	Yes
g[9]	Location	PIN_Y6	Yes
b[9]	Location	PIN_R6	Yes

时钟 clk=50MHz

输入端口：reset 对应拨动开关 SW15，高电平复位

双向端口：m_clk 对应 PS/2 的时钟信号 PS2_A_CLK

m_data 对应 PS/2 的数据信号 PS2_A_DAT

输出端口：b[9]对应 VGA 接口的数据信号 VGAD[4]

g[9]对应 VGA 接口的数据信号 VGAD[10]

r[9]对应 VGA 接口的数据信号 VGAD[15]

hs 对应 VGA 接口的行同步信号 VH

vs 对应 VGA 接口的场同步信号 VV

4、下载，依次按下 PS/2 鼠标的左键、右键、滚轮，观察 VGA 显示屏的显示变化。

四、参考程序

略

实验十九、UART 串口收发

一、实验目的

了解异步串口的工作原理并设计简单的 UART 发送器和接收器。

二、实验原理

1、UART结构

UART主要由数据总线接口、控制逻辑、波特率发生器、发送部分和接收部分等组成。 功能包括微处理器接口，发送缓冲器（tbr）、发送移位寄存器（tsr）、帧产生、奇偶校验、并转串、数据接收缓冲器（rbr）、接收移位寄存器（rsr）、帧产生、奇偶校验、串转并。图19-1是UART的典型应用。

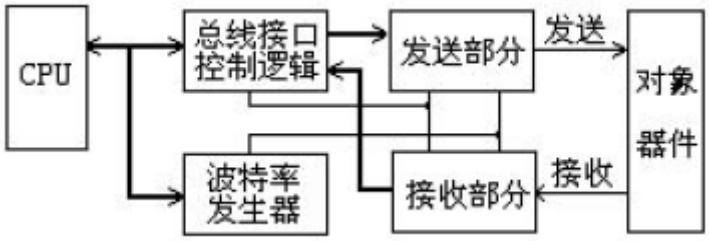


图 19-1　UART的典型应用

2、UART的帧格式

UART的帧格式如图19-2所示。

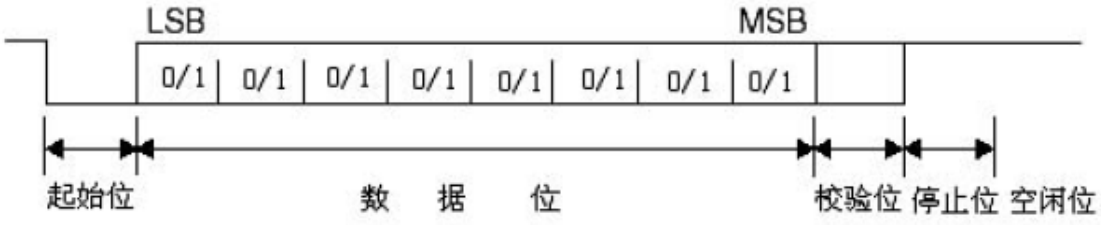


图 19-2　UART的帧格式

UART的帧格式包括线路空闲状态（idle，高电平）、起始位(start bit，低电平)、5～8 位数据位(data bits)、校验位(parity bit，可选)和停止位(stop bit，位数可为1、1.5、2 位)。 这种格式是由起始位和停止位来实现字符的同步。

UART内部一般有配置寄存器，可以配置数据位数（5～8 位）、是否有校验位和校验的类型、停止位的位数（1，1.5，2）等设置。

3、波特率发生器

UART的接收和发送是按照相同的波特率进行收发的。波特率发生器产生的时钟频率不是波特率时钟频率，而是波特率时钟频率的16倍，目的是为在接收时进行精确的采样，以提出异步的串行数据。

根据给定的晶振时钟和要求的波特率算出波特率分频数。

假设数据的波特率为p，则所需时钟的频率为16*p。以波特率p为9600为例，系统时钟为50MHz，则分频系数为50000000/(16*9600) = 325.52，取整为326。

4、UART 发送器

发送器每隔16个CLK16 时钟周期输出1 位，次序遵循1位起始位、8位数据位（假定数据位为8位）、1位校验位（可选）、1位停止位。

CPU 何时可以往发送缓冲器tbr 写入数据，也就是说CPU 要写数据到tbr 时必须判断当前是否可写，如果不判这个条件，发送的数据会出错。

数据的发送是由微处理器控制，微处理器给出wen信号，发送器根据此信号将并行数据din[7..0]锁存进发送缓冲器tbr[7..0]，并通过发送移位寄存器tsr[7..0]发送串行数据至串行数据输出端dout。在数据发送过程中用输出信号tre作为标志信号，当一帧数据发送完毕时，tre 信号为1，通知CPU 在下个时钟装入新数据。

5、UART 接收器

串行数据帧和接收时钟是异步的，发送来的数据由逻辑1 变为逻辑0 可以视为一个数据帧的开始。接收器先要捕捉起始位，确定rxd 输入由1 到0，逻辑0 要8个CLK16 时钟周期，才是正常的起始位，然后在每隔16个CLK16 时钟周期采样接收数据，移位输入接收移位寄存器rsr，最后输出数据dout。还要输出一个数据接收标志信号标志数据接收完。

三、实验内容

1、新建工程，编写 UART 串口发送和接收的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译适配。

To	Assignment Name	Value	Enabled
in clk50	Location	PIN_G1	Yes
in rx	Location	PIN_B11	Yes
out tx	Location	PIN_B13	Yes
in rst_n	Location	PIN_E4	Yes

时钟 clk=50MHz

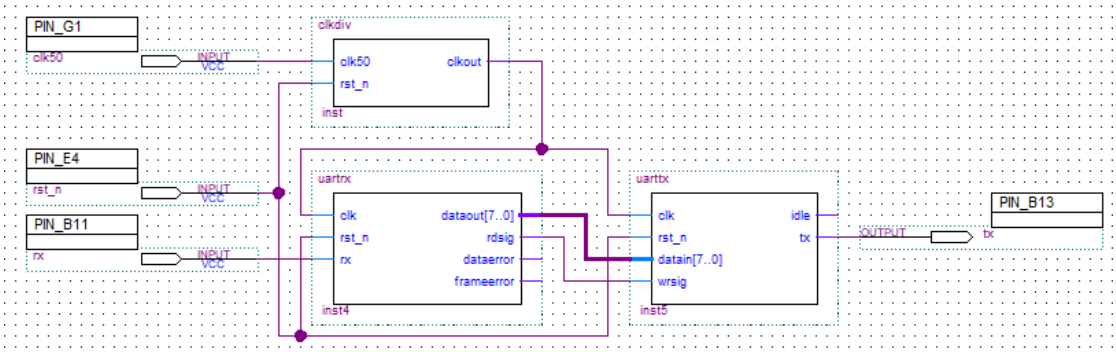
输入端口：rst_n 对应 FPGA 的复位信号 nRESET

rx 对应 UART 的接收信号 UART_RXD1

输出端口：tx 对应 UART 的发送信号 UART_TXD1

4、下载，通过串口调试助手，观察 UART 的发送和接收数据是否一致。

四、参考程序


PIN_G1
clk50
INPUT
VCC
clkdiv
clk50
rst_n
clkout
inst
PIN_E4
rst_n
INPUT
VCC
PIN_B11
rx
INPUT
VCC
uartrx
clk
rst_n
rx
dataout[7..0]
rdsig
dataerror
frameerror
inst4
uarttx
clk
rst_n
datain[7..0]
wrsig
idle
tx
inst5
OUTPUT
tx
PIN_B13

略

实验二十、LCD1602 液晶显示

一、实验目的

了解 LCD1602 液晶工作原理，在 LCD1602 液晶上实现字符显示。

二、实验原理

1、 1602 LCD 简介

1602 液晶也叫 1602 字符型液晶，它是一种专门用来显示字母、数字、符号等地点阵型液晶模块，它有若干个 5×7 或者 5×11 等点阵字符位组成，每个点阵字符位都可以显示一个字符。每位之间有一个点距的间隔，每行之间也有间隔，起到了字符间距和行间距的作用，正因为如此它不能显示图形。

1602 LCD 是指显示的内容为 16×2，即可以显示两行，每行 16 个字符液晶模块（显示字符和数字）。

1602 LCD 的特性：①+5V 电压，对比度可调；②内含复位电路；③提供各种控制命令，如：清屏、字符闪烁、光标闪烁、显示移位等多种功能；④有 80 字节显示数据存储器 DDRAM；⑤内建有 160 个 5×7 点阵的字型的字符发生器 CGROM；⑥8 个可由用户自定义的 5×7 的字符发生器 CGRAM。

2、1602 LCD 的基本参数及引脚功能

1602 LCD 分为带背光和不带背光两种，是否带背光在应用中并无差别。1602LCD 主要技术参数：显示容量为 16×2 个字符；芯片工作电压为 4.5~5.5V；工作电流为 2.0mA(5.0V)；模块最佳工作电压为 5.0V；字符尺寸为 2.95×4.35（W×H）mm。

1602 LCD 采用标准的 16 脚接口，各引脚接口说明如表 20-1 所示：

编号	符号	引脚说明	编号	符号	引脚说明
1	VSS	电源地	9	D2	数据
2	VDD	电源正极	10	D3	数据
3	VL	液晶显示偏压	11	D4	数据
4	RS	数据/命令选择	12	D5	数据
5	R/W	读/写选择	13	D6	数据
6	E	使能信号	14	D7	数据
7	D0	数据	15	BLA	背光源正极
8	D1	数据	16	BLK	背光源负极

表 20-1　引脚说明

3、1602 LCD 的指令说明

1602 液晶模块内部的控制器指令共有 11 条，如表 20-2 所示:

序号	指令	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
1	清显示	0	0	0	0	0	0	0	0	0	1
2	光标返回	0	0	0	0	0	0	0	0	1	*
3	置输入模式	0	0	0	0	0	0	0	1	I/D	S
4	显示开/关控制	0	0	0	0	0	0	1	D	C	B
5	光标或字符移位	0	0	0	0	0	1	S/C	R/L	*	*
6	置功能	0	0	0	0	1	DL	N	F	*	*
7	置字符发生存贮器地址	0	0	0	1	字符发生存贮器地址					
8	置数据存贮器地址	0	0	1	显示数据存贮器地址						
9	读忙标志或地址	0	1	BF	计数器地址						
10	写数到 CGRAM 或 DDRAM）	1	0	要写的数据内容							
11	从 CGRAM 或 DDRAM 读数	1	1	读出的数据内容							

表 20-2　1602 液晶内部控制器指令

与 HD44780 相兼容的芯片的时序如表 20-3 如下：

读状态	输入	RS=L，R/W=H，E=H	输出	D0-D7=状态字
写指令	输入	RS=L，R/W=L，D0-D7=指令码， E=高脉冲	输出	无
读数据	输入	RS=H，R/W=H，E=H	输出	D0-D7=数据
写数据	输入	RS=H，R/W=L，D0-D7=数据， E=高脉冲	输出	无

表 20-3　与 HD44780 相兼容的芯片时序

4、如何定义一个字符

1602 液晶显示模块是一个慢显示器件，所以在执行每条指令之前一定要确认模块的忙标志位低电平，表示不忙，否则此指令失效。要显示字符时要先输入

显示字符地址，也就是要告诉模块在哪里显示字符。例如第二行第一个字符的地址是 40H，那么是否直接写入 40H 就可以将光标定位在第二行第一个字符的位置呢？这样不行，因为写入显示地址时要求最高位 D7 恒定为高电平，所以实际写入的数据应该是 01000000B（40H）+10000000B（80H）=11000000B（C0H）。在对液晶模块的初始化中要先设置其显示模式，在液晶显示模块显示字符时光标是自动右移的，无需人工干预。每次输入指令钱都要判断液晶模块是否处于忙状态。1602 液晶模块内部的字符发生器已经存储了 160 个不同的点阵字符图形，每一个字符都有一个固定的代码，比如大写的英文字母“A”的代码是 01000001B（41H），显示模块是把地址 41H 中的点阵字符图形显示出来，我们就能看到“A”。

三、实验内容

1、新建工程，编写用 LCD1602 实现字符串显示的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译适配。

To	Assignment Name	Value	Enabled
in clk	Location	PIN_G1	Yes
out LCD_D[0]	Location	PIN_T10	Yes
out LCD_D[1]	Location	PIN_T9	Yes
out LCD_D[2]	Location	PIN_U17	Yes
out LCD_D[3]	Location	PIN_U16	Yes
out LCD_D[4]	Location	PIN_T16	Yes
out LCD_D[5]	Location	PIN_W15	Yes
out LCD_D[6]	Location	PIN_V15	Yes
out LCD_D[7]	Location	PIN_V13	Yes
out LCD_RS	Location	PIN_T13	Yes
out LCD_RW	Location	PIN_T12	Yes
out LCD_E	Location	PIN_T11	Yes
in rest	Location	PIN_E4	Yes

时钟 clk=50MHz

输入端口：reset_n 对应 FPGA 的复位信号 nRESET

输出端口：

LCD_D[7:0]对应 LCD1602 的数据信号 LCD_DB[7:0]

LCD_RS 对应 LCD1602 的数据/命令选择信号 LCD_RS

LCD_RW 对应 LCD1602 的读/写选择信号 LCD_R/W

LCD_E 对应 LCD1602 的使能选择信号 LCD_E

4、下载，观察 LCD1602 液晶上的字符显示。

四、参考程序

略

第六章　存储器实验

实验二十一、ROM

一、实验目的

1、熟悉 LPM 宏功能模块应用；

2、熟练掌握 ROM 的设计；

3、掌握 SignalTapII 的使用方法。

二、实验原理

1、基本原理

将正弦信号存放在波形数据表 ROM 中，mif 文件可以自己制作填充数据，也可以通过 mif 生成工具生成。通过 QuartusⅡ中内嵌的 SignalTapII 进行分析。

2、实验任务

借助 SignalTapII 观察 ROM 中正弦信号发生器的实际波形。

三、实验内容

1、建立 mif 文件

启动 QuartusⅡ建立空白工程，然后命名为 sinwave.qpf。建立 ROM 内的波形数据文件。单击菜单 File/New/Memory File 项，选择 Memory Initialization File 项（见图 21-1），单击 OK。确定 ROM 中数据数（Number of word）和字宽（Word size），单击 OK，打开 mif 文件。

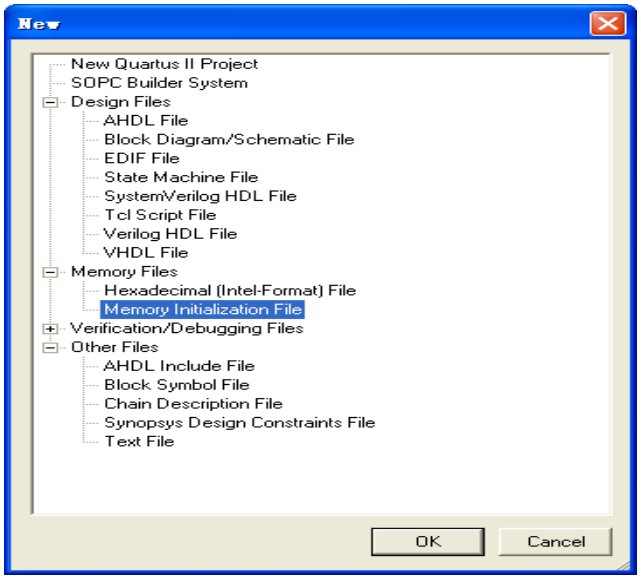


图 21-1　新建 mif 文件

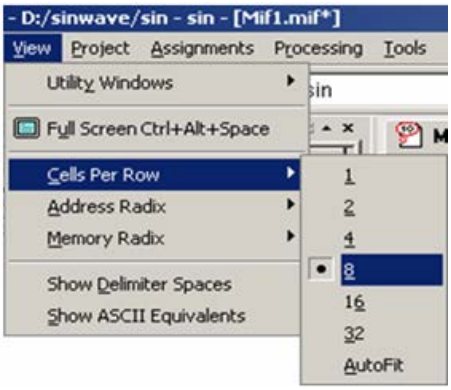


图 21-2　选择 mif 文件编辑窗口显示宽度

通过 View/Cell Per Row 菜单设定每行显示 8 个存储单元（见图 21-2），这时 mif 文件编辑窗口如图 21-3 所示。在 mif 文件编辑窗口中填入正弦波数据，以文件名 sinwave.mif 保存到当前工程文件夹中，如图 21-4 所示。

Mif1.mif*

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	0	0	0	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
32	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0

图 21-3　mif 文件编辑窗口

sinwave.mif

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	128	140	153	165	177	188	199	209
8	219	227	235	241	246	250	253	255
16	255	254	252	248	244	238	231	223
24	214	204	194	183	171	159	147	134
32	121	108	96	84	72	61	51	41
40	32	24	17	11	7	3	1	0
48	0	2	5	9	14	20	28	36
56	46	56	67	78	90	102	115	127

图 21-4　将波形数据填入 mif 文件表中

2、 完成存放波形数据 ROM 的设计

利用 MegaWizard Plug-In Manager 定制正弦信号数据 ROM 宏功能块，并将以上的波形数据文件 sinwave.mif 加载于此 ROM 中。设计步骤如下：

（1）通过菜单 Tools/MegaWizard Plug-In Manager 弹出对话框，选择第一项，即定制一个新的模块。单击 Next，打开图 21-5 所示对话框，在左栏选择 Memory Compiler 中的 ROM：1-PORT，再选择相应器件系列 Cyclone IV E 和 VHDL 语言方式，最后输入 ROM 文件存放路径和文件名：D:/TCP-II/exp/21-ROM/drom.vhd，单击 Next。

（2）选择 ROM 地址线、数据线和控制线。在如图 21-6 所示的对话框中选择地址线位宽和 ROM 中数据数分别为 8 和 64，默认地址锁存控制信号为 clock，单击 Next。在图 21-7 中，去掉输出锁存的勾选标记。

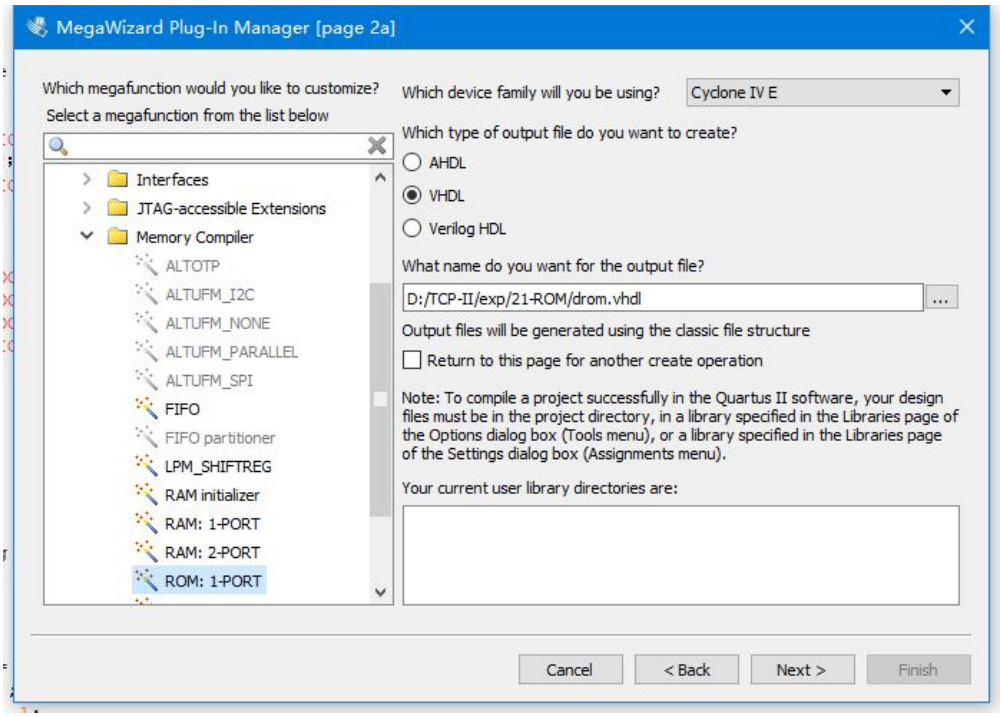


图 21-5　LPM 宏功能块设定

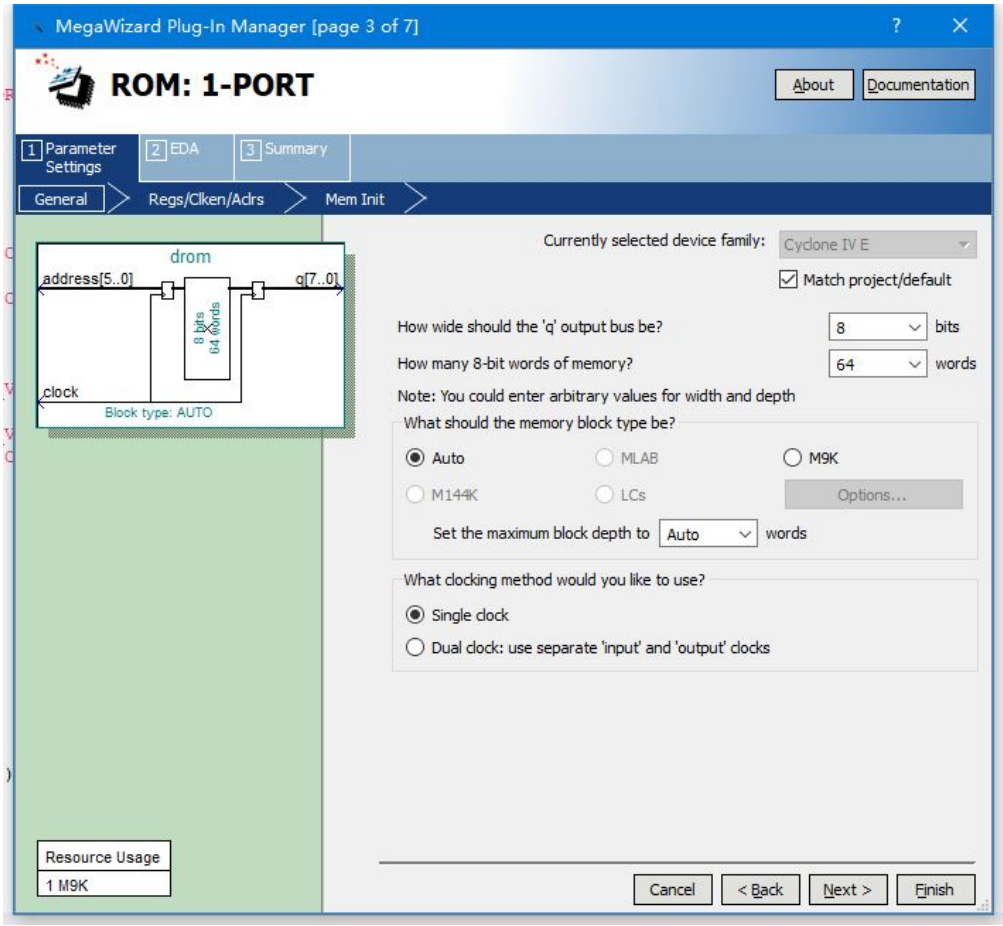


图 21-6　选择 ROM 模块数据线和地址线宽度

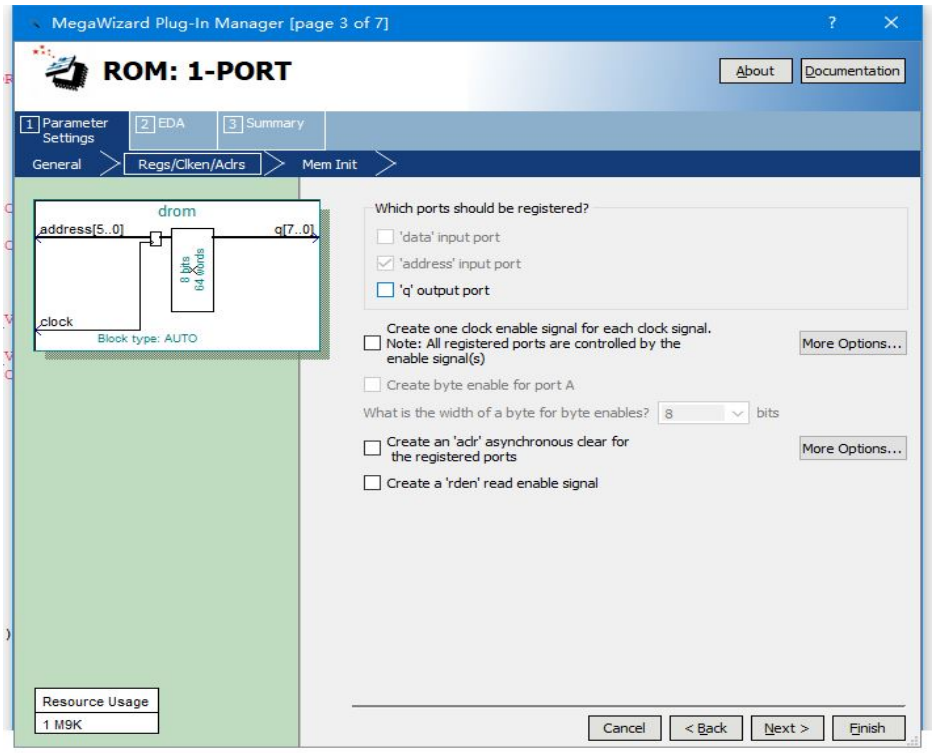


图 21-7　去除输出锁存信号 q

（3）指定 ROM 所对应的初始化文件 sinwave.mif。单击 Next 后，打开图 21-8 所示的对话框。在“Do you want to specify the initial content of the memory?”栏中选中“Yes，use this file for the memory content data”单选按钮，单击 Browse 按钮，选择指定路径上文件 sinwave.mif。

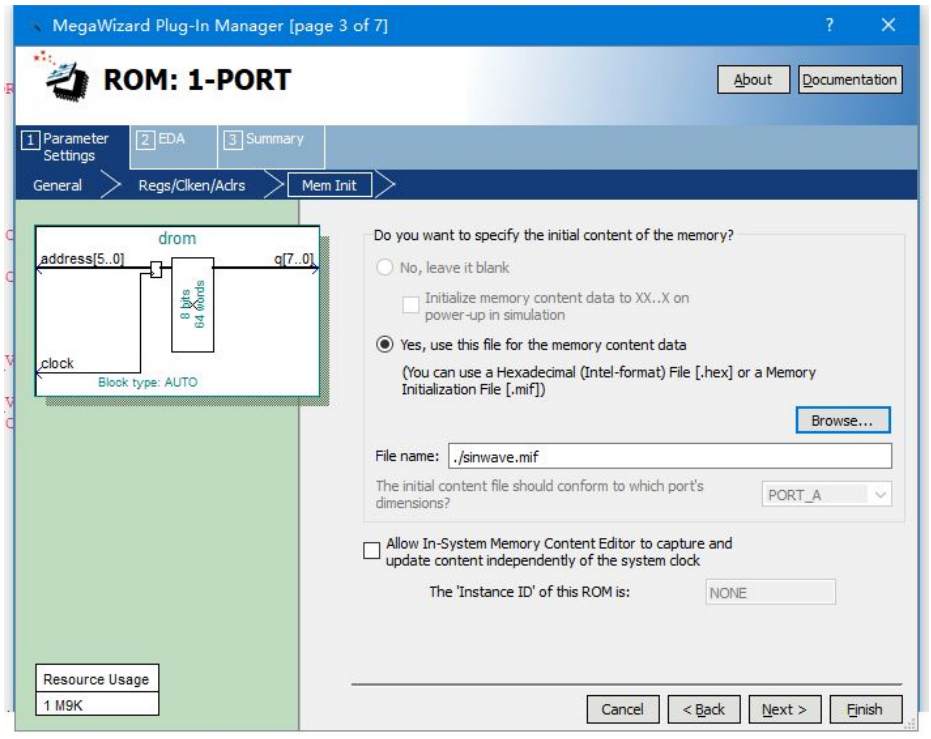


图 21-8　调入 ROM 初始化文件并选择在系统读写功能

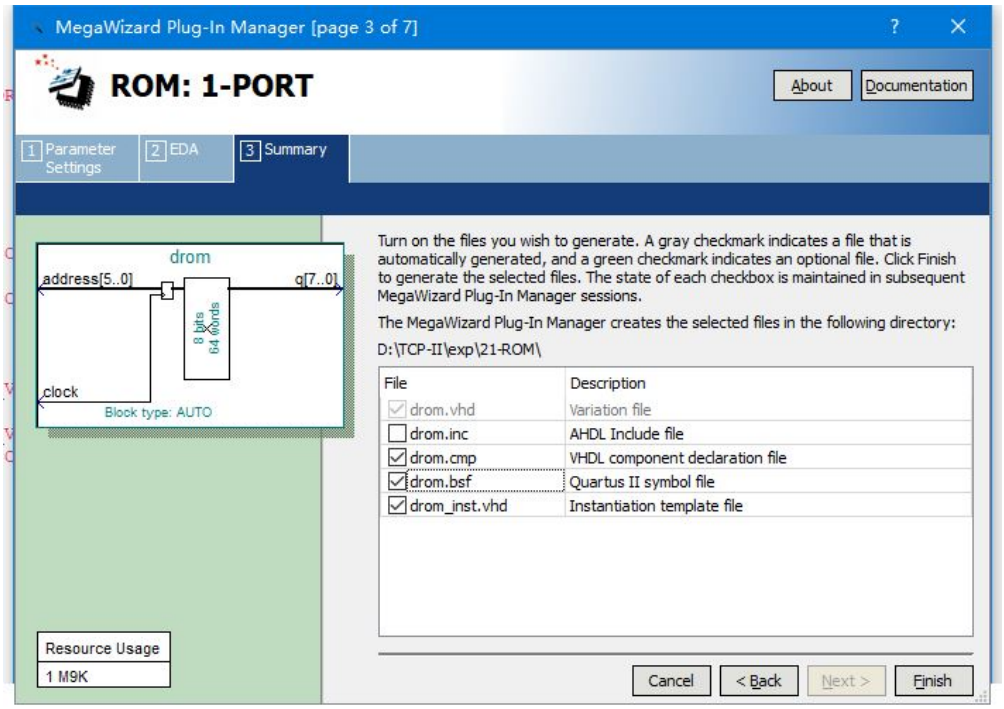


图 21-9　文件生成对话框

建议在文件名对话框中将文件路径改为相对路径，如图 21-8 所示。选中“Alow In-System Memory...”栏复选框，并在“The Instance ID of this ROM is”栏键入“rom1”，作为此 ROM 的 ID 名称。通过这个设置，可在 In-System Memory Content Editor 窗口中查看编辑 ROM 中的内容。

（4）单击 Next 按钮两次，进入图 21-9 文件生成对话框，为便于搭建顶层原理图时调用 ROM 元件，建议勾选 rom1.bsf 选项，单击 Finish 按钮，完成 ROM 元件定制。

3、 编写正弦波发生器的 VHDL 代码如下：

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY sinwave IS
    PORT (
        clk     : IN STD_LOGIC;
        dout    : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
    );
END sinwave;
```

```
ARCHITECTURE trans OF sinwave IS
COMPONENT drom
    PORT (
        address         : IN STD_LOGIC_VECTOR (5 DOWNTO 0);
         clock          : IN STD_LOGIC;
         q          : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
    );
END COMPONENT;

    SIGNAL q1                : STD_LOGIC_VECTOR(5 DOWNTO 0);
    SIGNAL newclk          : STD_LOGIC;
    SIGNAL clk_cnt        : STD_LOGIC_VECTOR(4 DOWNTO 0);

BEGIN
  PROCESS (clk)
    BEGIN
        IF (clk'EVENT AND clk = '1') THEN
            newclk <= clk_cnt(4);
            clk_cnt <= clk_cnt + 1;
        END IF;
    END PROCESS;

    PROCESS (newclk)
    BEGIN
        IF (newclk'EVENT AND newclk = '1') THEN
            q1 <= q1 + 1;
        END IF;
    END PROCESS;

    u1 : drom
        PORT MAP (
            address    => q1,
            clock       => clk,
            q              => dout
```

);

END trans;

4、 测试 FPGA 芯片中电路 sinwave 实际波形的步骤如下：

(1) 打开 SignalTapII 编辑窗口

点击菜单 File/New，单击 Verification/Debugging Files 选择 SignalTap II File，再点击 OK，出现图 21-10 所示 SignalTap II 编辑窗口。

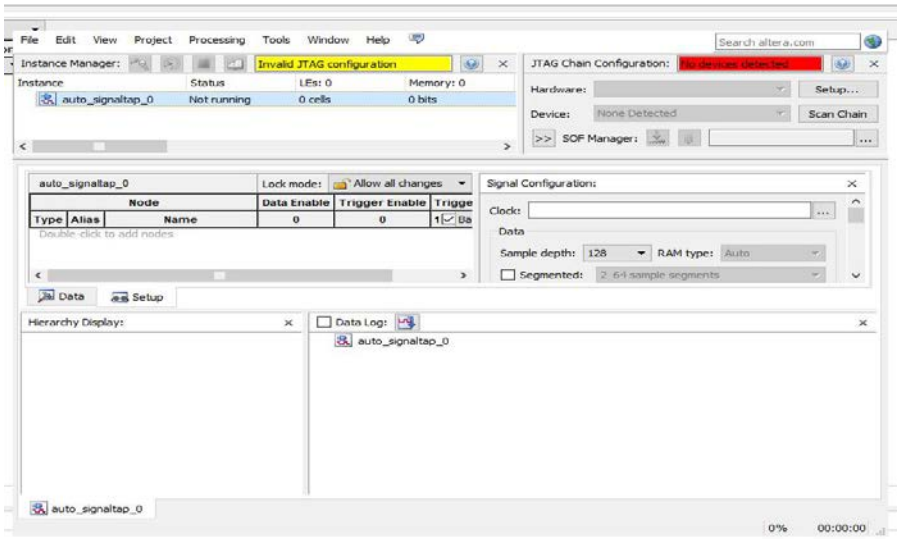


图 21-10　SignalTap II 编辑窗口

（2）在 SignalTapII 编辑窗口中，将 Instance 栏中的 auto_signaltap_0 重命名，把它改为本项目名 sinwave，于是图 21-10 变为图 21-11。在图 21-11 左下栏中的空白处双击鼠标左键，弹出 Node Finder 的窗口，单击 List 按钮可以得到相关的信号名，把需要观察的信号调入。注意调入信号的数量应根据实际需要来决定，不要调入过多的信号，以免占用过多 FPGA 芯片资源。

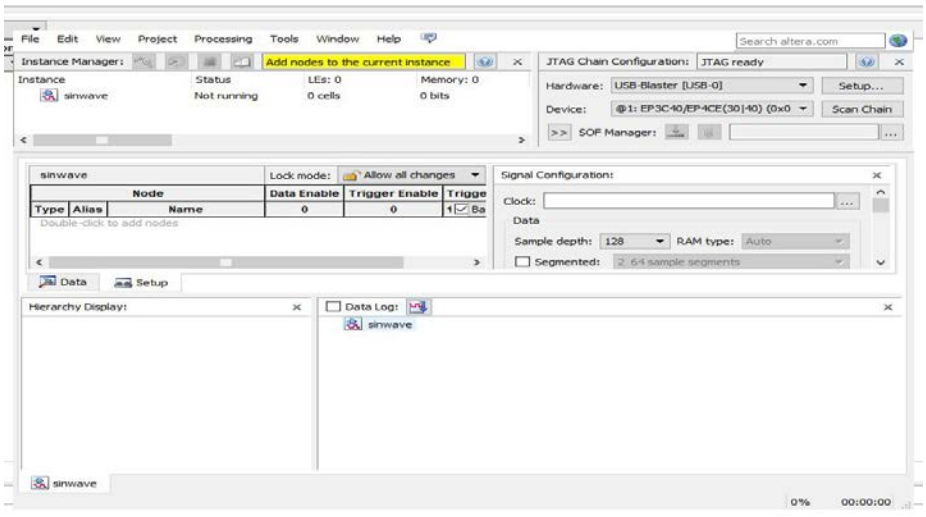


图 21-11　SignalTap II 参数设置

设置 SignalTapII 参数，在 SignalTapII 编辑窗口中，单击左下角的 Setup 按钮，出现参数设置编辑窗口。

① 设置 ELA 的采样时钟信号，该时钟可以是本设计的内部时钟，也可以是外部输入时钟。本例采用内部时钟 CLK。

② 在 Sample depth 栏选择采样深度，例如 8K 位，表明所有的信号都采样 8K 个点。采样点数应根据需要和硬件资源消耗综合考虑确定。

③设定起始触发位置，比如选择前点触发“Pre trigger position”。

④ 选择采样的触发信号和触发方式，接受默认选项。

（3）保存文件

选择 Save 以保存文件，如果出现“Do you want to enable SignalTap II…”，选择“是”就可以激活 SignalTapII。

（4）编译并下载

启动全程编译，编译完成后，SignalTapII 的观察窗口会自动打开（若没有打开，可选择命令 Tools | SignalTap II Analyzer 将其打开）。然后将编译好的 sof 文件下载到 FPGA 器件。

（5）启动 SignalTapII 进行采样与分析

在图 21-12 中单击 Autorun Analysis 红色带箭头按钮，启动 SignalTapII。点击左下角 data 页，这时就可以观察到来自实验板上 FPGA 内部的实测波形，如图 21-13 所示。

图 21-12　启动 SignalTapII

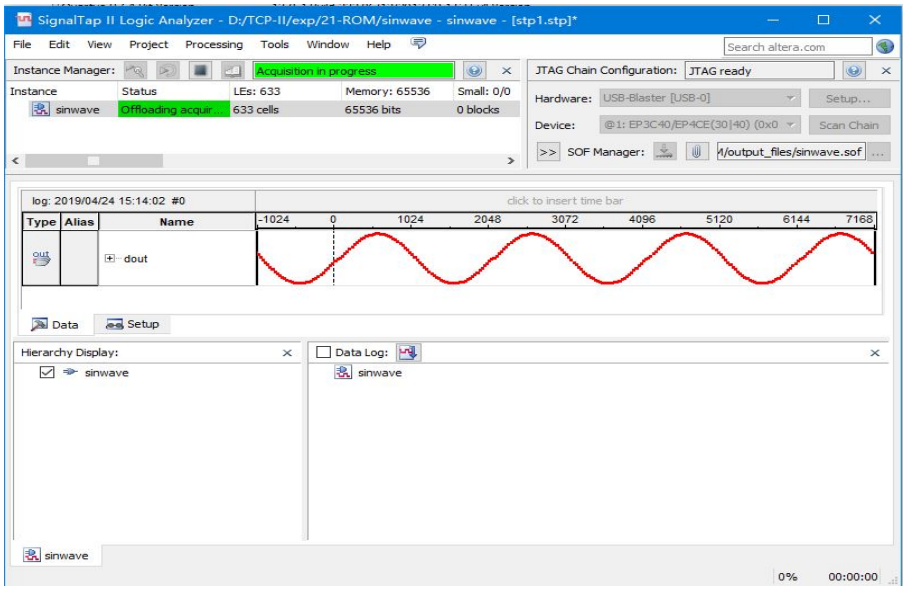


图 21-13　SignalTapII 实测波形

如果出现不了正弦波，鼠标点击 dout，按右键选择 Bus Display Format/Unsigned Line Chart，如图 21-15。鼠标再点波形，点鼠标右键调节波形周期。

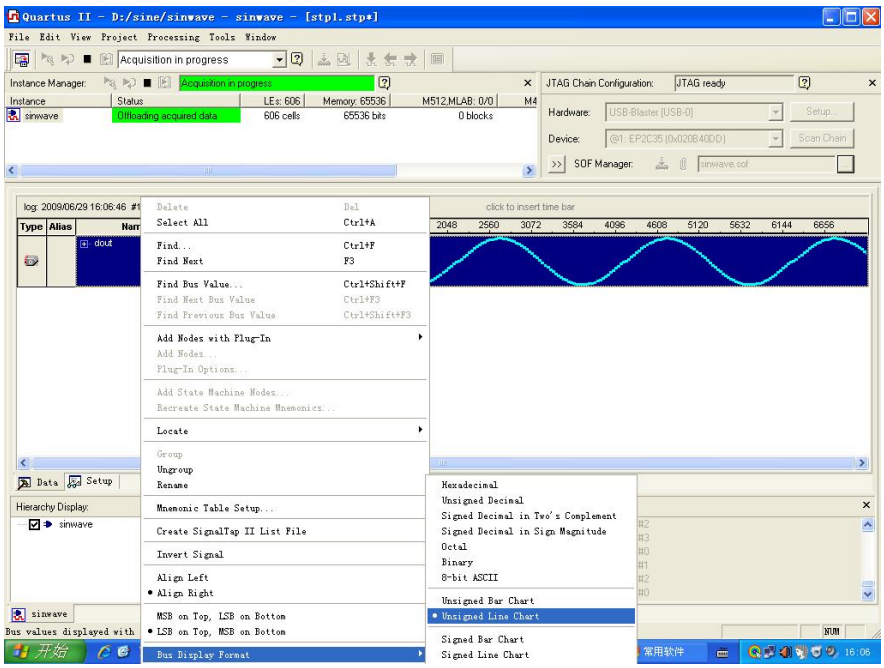


图 21-15　正弦波输出格式选择

实验二十二、SRAM 读写控制

一、实验目的

1、了解 SRAM 工作原理；

2、学习 SRAM 读写控制方法。

二、实验原理

1、SRAM简介

SRAM（静态随机存取存储器）是一种具有静止存取功能的掉电易失性存储器，不需要刷新电路即能保存它内部存储的数据。

2、IS61LV25616 的读写实现

表 22-1 列出了 IS61LV25616 的主要引脚控制端，其功能如表所述。

A0-A17	输入	地址总线
IO0-IO15	双向	数据总线
CE	输入	片选，低电平有效
OE	输入	读使能，低电平有效
WE	输入	写使能，低电平有效
LB	输入	低字节使能，低电平有效
UB	输入	高字节使能，低电平有效

表 22-1　IS61LV25616 的主要信号线

IS61LV25616 是一种高速 256K X 16 比特容量的 CMOS 静态随机存储器，其内部主要是组合逻辑电路。对它的读写控制主要是通过控制 CE、OE、WE、LB 和 UB 来实现的。芯片的各种读写控制的真值表如图 22-1 所示。

						I/O PIN	
Mode	$\overline{WE}$	$\overline{CE}$	$\overline{OE}$	$\overline{LB}$	$\overline{UB}$	I/O0-I/O7	I/O8-I/O15
Not Selected	X	H	X	X	X	High-Z	High-Z
Output Disabled	H	L	H	X	X	High-Z	High-Z
	X	L	X	H	H	High-Z	High-Z
Read	H	L	L	L	H	DOUT	High-Z
	H	L	L	H	L	High-Z	DOUT
	H	L	L	L	L	DOUT	DOUT
Write	L	L	X	L	H	DIN	High-Z
	L	L	X	H	L	High-Z	DIN
	L	L	X	L	L	DIN	DIN

图 22-1　IS61LV25616 的读写真值表

IS61LV25616 的数据线总线 DQ 是双向的，当对进行写操作时，数据线作输入

用，当对它进行些操作时，数据线作输出用。一些典型的 SRAM 读、写操作如图 22-2 和图 22-3 所示。地址线在 OE、CE、WE 有效期间应保持稳定。

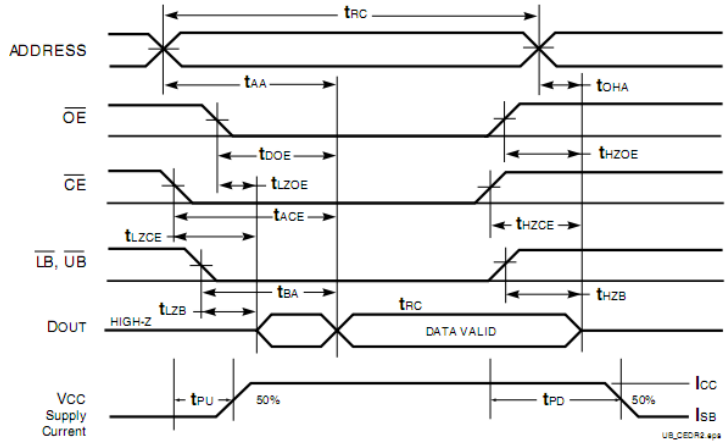


图 22-2　SRAM 的读操作

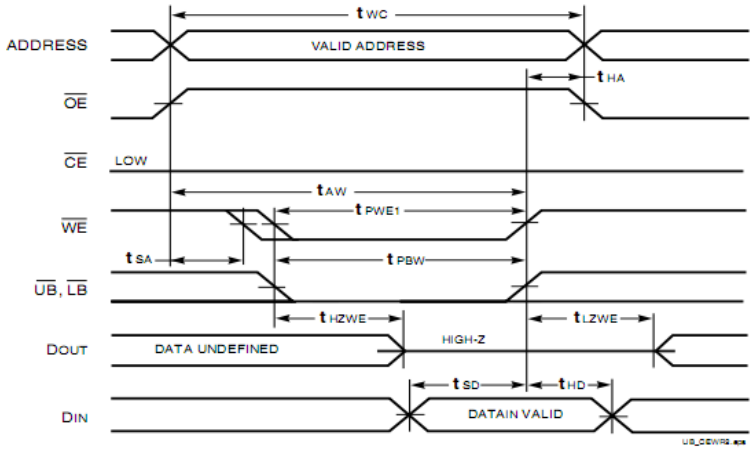


图 22-2　SRAM 的写操作

三、实验内容

1、启动 Quartus Ⅱ，建立 sram_ctrl.qpf 工程。

2、编写对 SRAM 进行读写控制的 VHDL 程序。

3、点击工具栏中的图标，对工程进行综合和分析，检查程序是否有错误，如有错误根据提示修改程序，直到成功为止。

4、点击工具栏中的图标，在弹出的窗口中为器件分配输入输出管脚，分配完成后点击图标对工程进行编译。

To	Assignment Name	Value	Enabled
CE_N	Location	PIN_E4	Yes
clk	Location	PIN_A11	Yes
RD_Data_OUT[0]	Location	PIN_C19	Yes
RD_Data_OUT[1]	Location	PIN_D19	Yes
RD_Data_OUT[2]	Location	PIN_D17	Yes
RD_Data_OUT[3]	Location	PIN_C17	Yes
RD_Data_OUT[4]	Location	PIN_E16	Yes
RD_Data_OUT[5]	Location	PIN_G15	Yes
RD_Data_OUT[6]	Location	PIN_F15	Yes
RD_Data_OUT[7]	Location	PIN_D15	Yes
RD_Data_OUT[8]	Location	PIN_C15	Yes
RD_Data_OUT[9]	Location	PIN_E15	Yes
RD_Data_OUT[10]	Location	PIN_E14	Yes
RD_Data_OUT[11]	Location	PIN_C13	Yes
RD_Data_OUT[12]	Location	PIN_E11	Yes
RD_Data_OUT[13]	Location	PIN_F11	Yes
RD_Data_OUT[14]	Location	PIN_E12	Yes
RD_Data_OUT[15]	Location	PIN_D13	Yes
WR_Data_IN[0]	Location	PIN_H1	Yes
WR_Data_IN[1]	Location	PIN_H2	Yes
WR_Data_IN[2]	Location	PIN_F2	Yes
WR_Data_IN[3]	Location	PIN_F1	Yes
WR_Data_IN[4]	Location	PIN_E1	Yes
WR_Data_IN[5]	Location	PIN_D2	Yes
WR_Data_IN[6]	Location	PIN_C2	Yes
WR_Data_IN[7]	Location	PIN_C1	Yes
WR_Data_IN[8]	Location	PIN_B2	Yes
WR_Data_IN[9]	Location	PIN_B1	Yes
WR_Data_IN[10]	Location	PIN_T2	Yes
WR_Data_IN[11]	Location	PIN_T1	Yes
WR_Data_IN[12]	Location	PIN_C6	Yes
WR_Data_IN[13]	Location	PIN_F7	Yes
WR_Data_IN[14]	Location	PIN_G7	Yes
WR_Data_IN[15]	Location	PIN_F13	Yes
SRAM_Addr[0]	Location	PIN_U22	Yes
SRAM_Addr[1]	Location	PIN_V21	Yes

时钟 clk 对应 CLK_CHOOSE

CE_N 对应 FPGA 的复位信号 nRESET

RD_Data_OUT[0:15]对应 LED[0:15]

WR_Data_IN[0:15]对应 SW[0:15]

SRAM_RDO_WR1 对应 KEY1

其它信号分别对应 SRAM 的地址、数据和控制信号

SRAM_Addr[2]	Location	PIN_V22	Yes
SRAM_Addr[3]	Location	PIN_W21	Yes
SRAM_Addr[4]	Location	PIN_W22	Yes
SRAM_Addr[5]	Location	PIN_AB17	Yes
SRAM_Addr[6]	Location	PIN_AA16	Yes
SRAM_Addr[7]	Location	PIN_AB16	Yes
SRAM_Addr[8]	Location	PIN_AA15	Yes
SRAM_Addr[9]	Location	PIN_AB15	Yes
SRAM_Addr[10]	Location	PIN_V16	Yes
SRAM_Addr[11]	Location	PIN_U15	Yes
SRAM_Addr[12]	Location	PIN_Y17	Yes
SRAM_Addr[13]	Location	PIN_W17	Yes
SRAM_Addr[14]	Location	PIN_W19	Yes
SRAM_Addr[15]	Location	PIN_R21	Yes
SRAM_Addr[16]	Location	PIN_P22	Yes
SRAM_Addr[17]	Location	PIN_P21	Yes
SRAM_CE_N	Location	PIN_Y21	Yes
SRAM_OE_N	Location	PIN_R22	Yes
SRAM_WE_N	Location	PIN_AA17	Yes
SRAM_DQ[0]	Location	PIN_Y22	Yes
SRAM_DQ[1]	Location	PIN_AA21	Yes
SRAM_DQ[2]	Location	PIN_AA20	Yes
SRAM_DQ[3]	Location	PIN_AB20	Yes
SRAM_DQ[4]	Location	PIN_AA19	Yes
SRAM_DQ[5]	Location	PIN_AB19	Yes
SRAM_DQ[6]	Location	PIN_AA18	Yes
SRAM_DQ[7]	Location	PIN_AB18	Yes
SRAM_DQ[8]	Location	PIN_U19	Yes
SRAM_DQ[9]	Location	PIN_U20	Yes
SRAM_DQ[10]	Location	PIN_T18	Yes
SRAM_DQ[11]	Location	PIN_T17	Yes
SRAM_DQ[12]	Location	PIN_R20	Yes
SRAM_DQ[13]	Location	PIN_R19	Yes
SRAM_DQ[14]	Location	PIN_R18	Yes
SRAM_DQ[15]	Location	PIN_R17	Yes
SRAM_UB_N	Location	PIN_U21	Yes
SRAM_LB_N	Location	PIN_P17	Yes
SRAM_RD0_WR1	Location	PIN_T14	Yes

5、创建 Signal Tap 文件，在 Setup 一栏中，双击上方的空白处，从弹出的窗口中加入待观测的信号，在“Signal Configuration”栏目的 Clock 一栏中使用 clk 作为参考时钟，设定完成后的界面如图 22-4 所示。

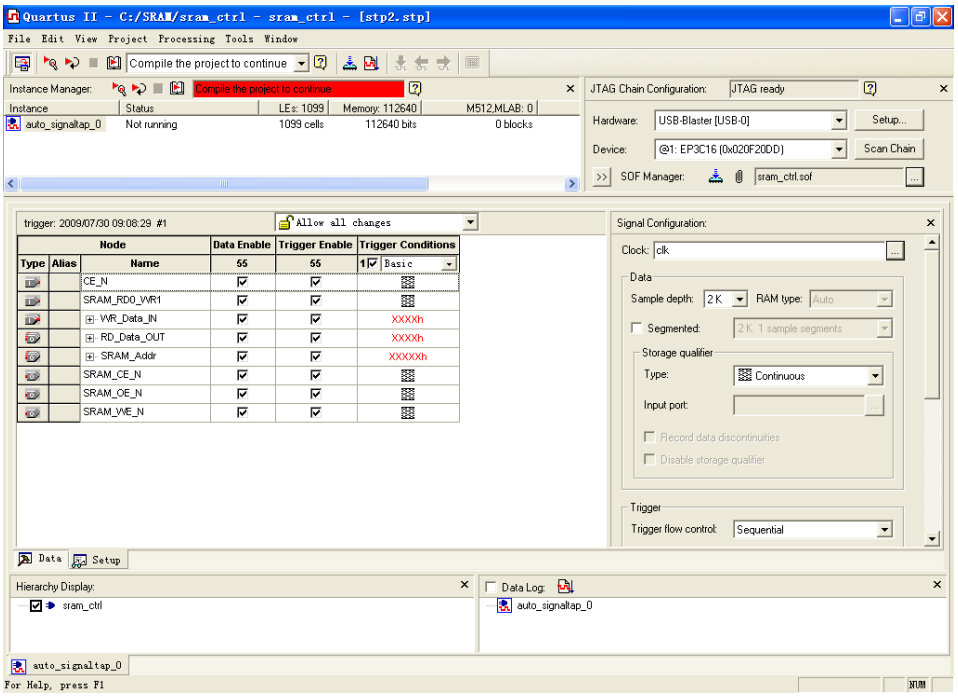


图 22-4　设定完成后的 Signal Tape 界面

6、下载配置文件至实验板，然后点击 Signal Tape 界面工具栏中的“Audio Analysis”图标开始运行 Signal Tape，观察不同的操作对应的不同现象。

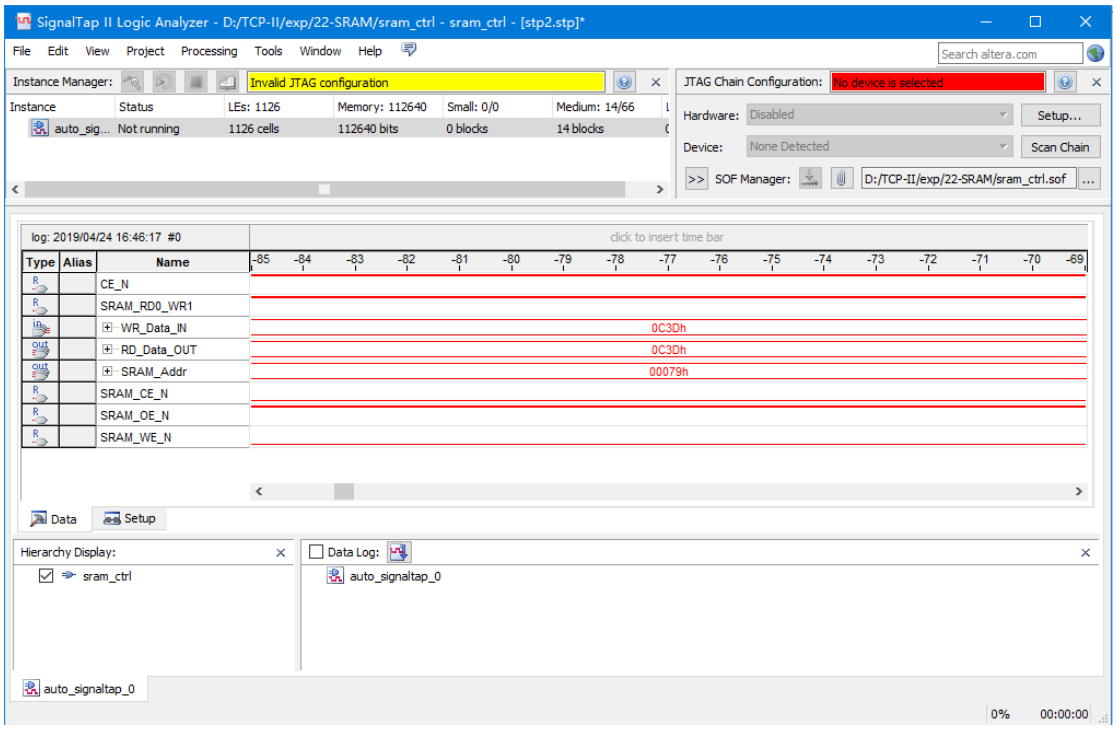


图 22-5　实验现象

实验二十三、FIFO 控制器

一、实验目的

1、了解 FIFO 的基本用途以及使用场合；

2、学会控制 FIFO。

二、实验原理

FIFO 是英文 First In First Out 的缩写，是一种先进先出的数据缓存器，他与普通存储器的区别是没有外部读写地址线，这样使用起来非常简单。FIFO 一般用于不同时钟域之间的数据传输，比如 FIFO 的一端是 AD 数据采集，另一端是计算机的 PCI 总线，假设其 AD 采集的速率为 16 位 100K SPS，那么每秒的数据量为 100K×16bit=1.6Mbps,而 PCI 总线的速度为 33MHz，总线宽度 32bit,其最大传输速率为 1056Mbps,在两个不同的时钟域间就可以采用 FIFO 来作为数据缓冲。

一般的 FIFO，有以下性能指标：

1）FIFO 的宽度：也就是英文资料里常看到的 THE WIDTH，它指的是 FIFO 一次读写操作的数据位，就像 MCU 有 8 位和 16 位，ARM 32 位等等。

2）FIFO 的深度：THE DEEPTH，它指的是 FIFO 可以存储多少个 N 位的数据（如果宽度为 N）。如一个 8 位的 FIFO，若深度为 8，它可以存储 8 个 8 位的数据，深度为 12 ，就可以存储 12 个 8 位的数据，FIFO 的深度可大可小，个人认为 FIFO 深度的计算并无一个固定的公式。在 FIFO 实际工作中，其数据的满/空标志可以控制数据的继续写入或读出。在一个具体的应用中也不可能由一些参数算出精确的所需 FIFO 深度为多少，这在写速度大于读速度的理想状态下是可行的，但在实际中用到的 FIFO 深度往往要大于计算值。一般来说根据电路的具体情况，在兼顾系统性能和 FIFO 成本的情况下估算一个大概的宽度和深度就可以了。而对于写速度慢于读速度的应用，FIFO 的深度要根据读出的数据结构和读出数据是由那些具体的要求来确定。

3）满标志：FIFO 已满或将要满时由 FIFO 的状态电路送出的一个信号，以阻止 FIFO 的写操作继续向 FIFO 中写数据而造成溢出（overflow）。

4）空标志：FIFO 已空或将要空时由 FIFO 的状态电路送出的一个信号，以阻止 FIFO 的读操作继续从 FIFO 中读出数据而造成无效数据的读出（underflow）。

5）读时钟：读操作所遵循的时钟，在每个时钟沿来临时读数据。

6）写时钟：写操作所遵循的时钟，在每个时钟沿来临时写数据。有的 FIFO 的读写时钟是共用的，但需要读写标志位以明确当前是读还是写。

7）读指针：指向下一个读出地址。读完后自动加 1。

8）写指针：指向下一个要写入的地址的，写完自动加 1。读写指针其实就是读写的地址，只不过这个地址不能任意选择，而是连续的。

三、实验内容

1、为实验的工程创建一个目录（比如：D:\TCP-II\exp\23-FIFO），打开 Quartus 软件，在创建的目录中新建一个工程并命名（如“FIFO”），注意选择正确的器件。

2、点击 File->New，弹出图 23-1 所示对话框，选择 Block Diagram/Schematic File，创建原理图文件。

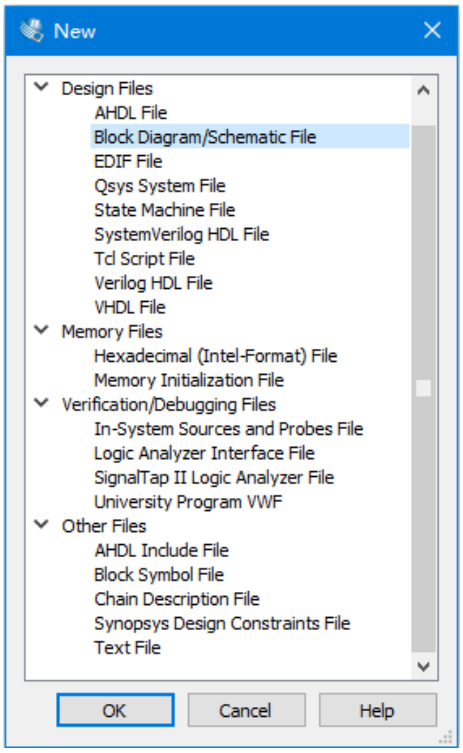


图 23-1　创建原理图文件

3、我们将使用宏功能模块添加一个 FIFO 器件至原理图中。在原理图文件的空白处双击，在图 23-2 的对话框中点击“MegaWizard Plug-In Manager”，然后在弹出的对话框中点击“next”。

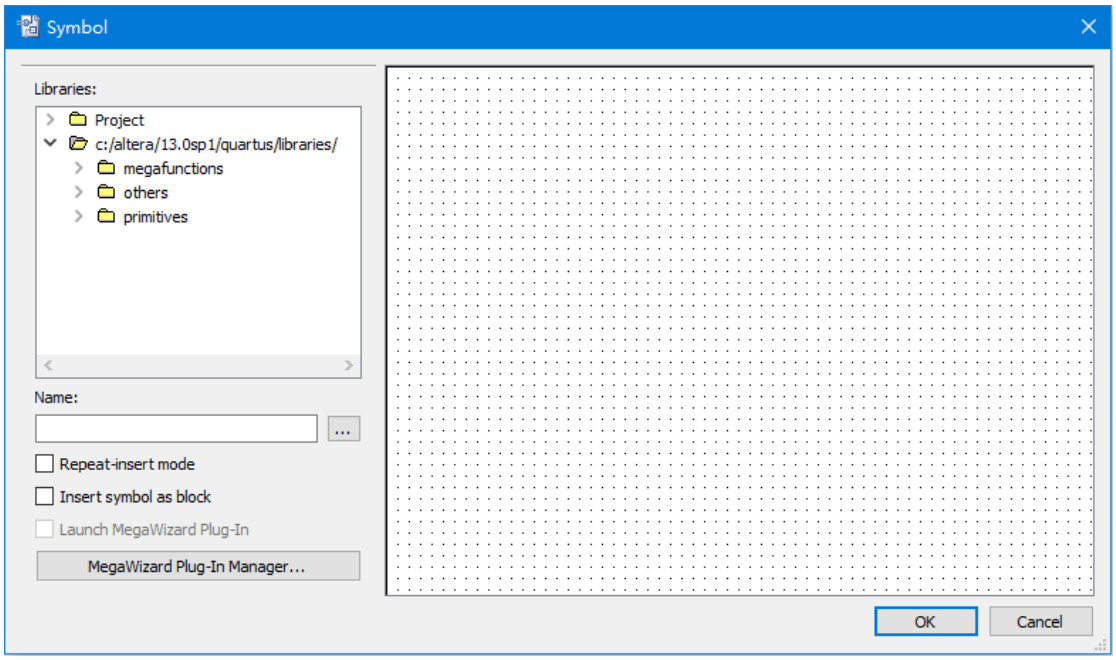


图 23-2　打开宏功能模块工具

4、在如图 23-3 所示的窗口中为添加的 FIFO 器件命名(如“myfifo”)，然后在左方的下拉列表的 Memory Compiler 中选择“FIFO”，点击“Next”。

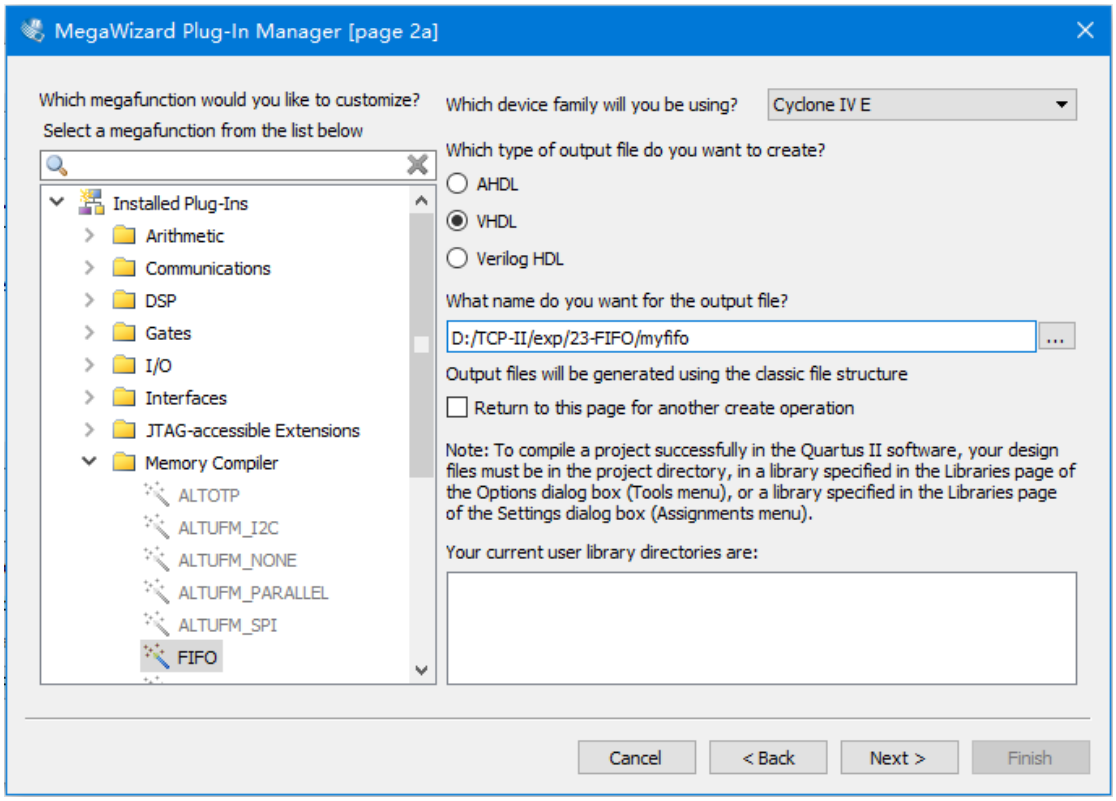


图 23-3　打开宏功能模块工具

5、在的对话框选择 FIFO 的宽度和深度，然后点击“next”，如图 23-4 所示。

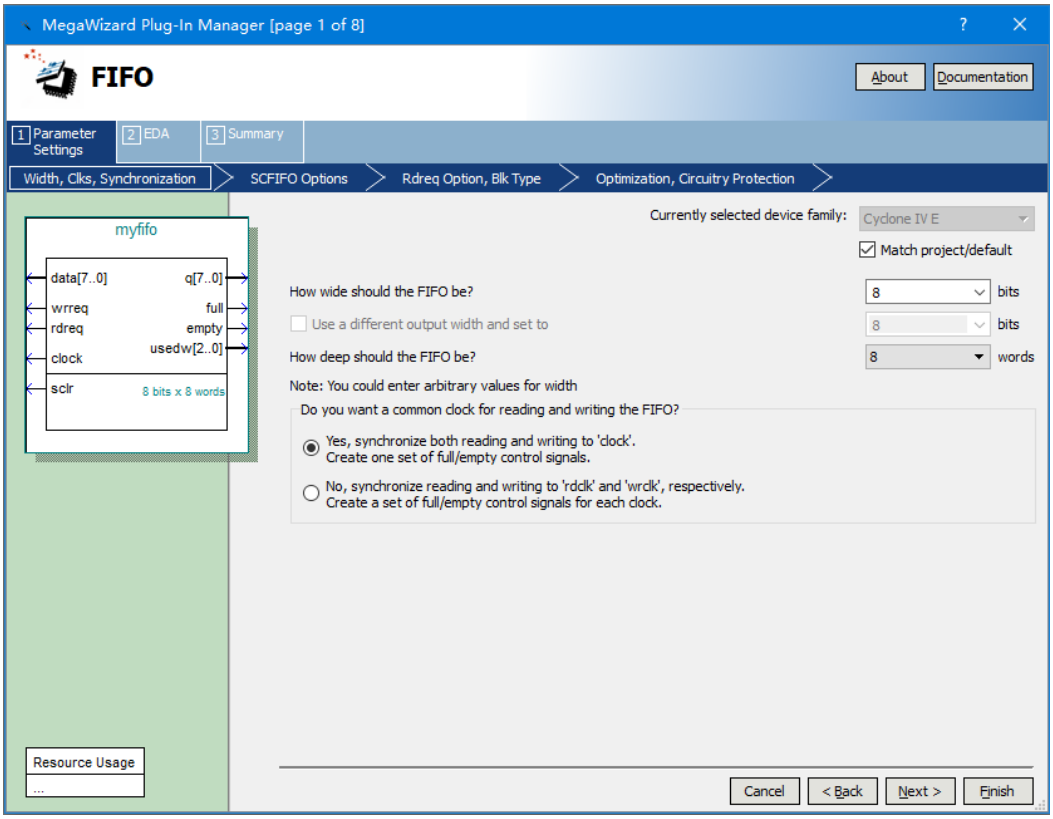


图 23-4　设置 FIFO 的宽度和深度

6、在图 23-5 所示的对话框中，选中“Synchronous clear”，然后点击“Next”。

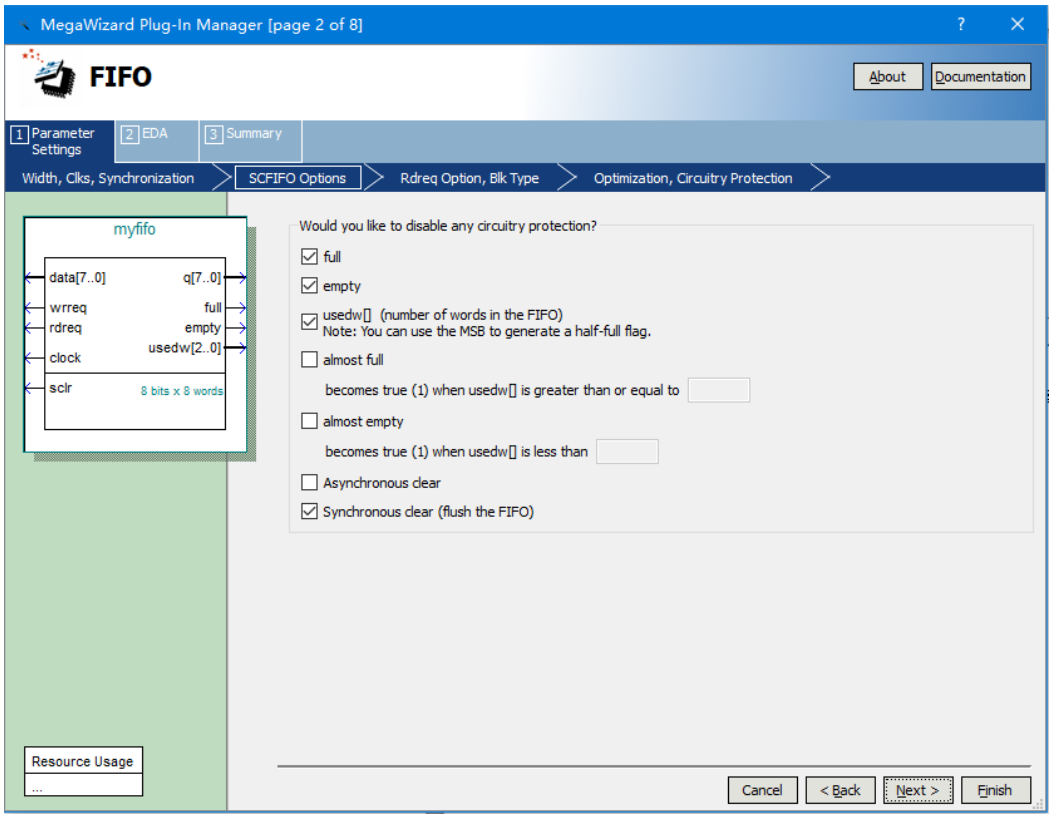


图 23-5　选择同步清零端

7、在剩下的步骤中，均按照默认的设置，点击“next”，最后点击“Finish”。

8、点击确定，则出现图 23-6 所示界面点击 OK，将此模块添加到工程原理图中。

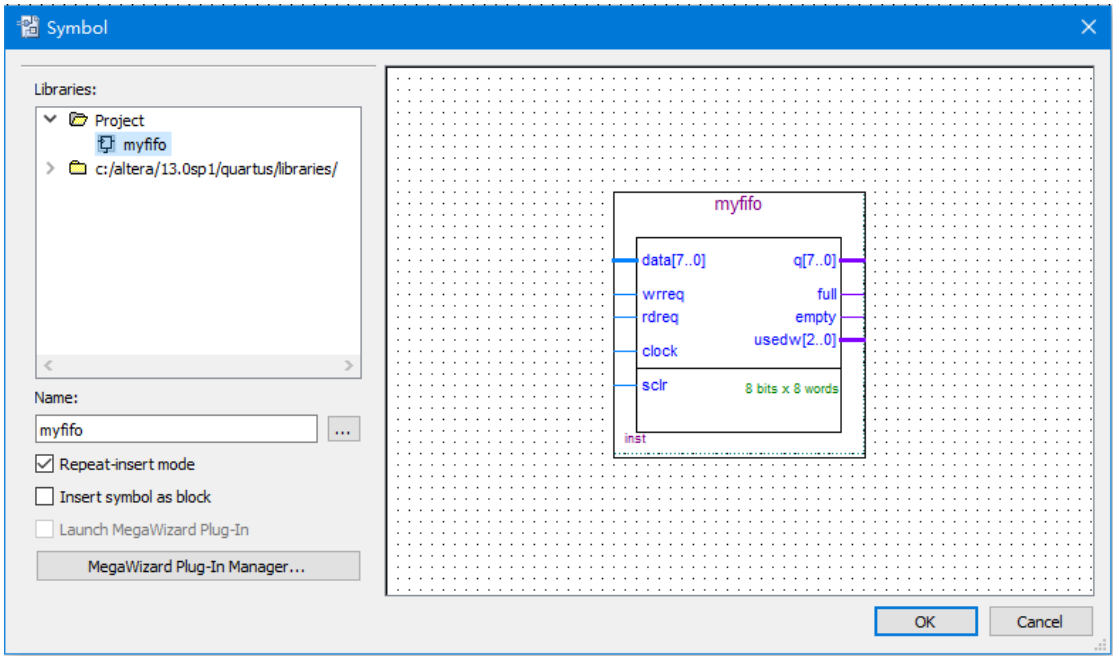


图 23-6　生成的 FIFO 模块

9、在原理图中添加输入输出端口，如图 23-7 所示。

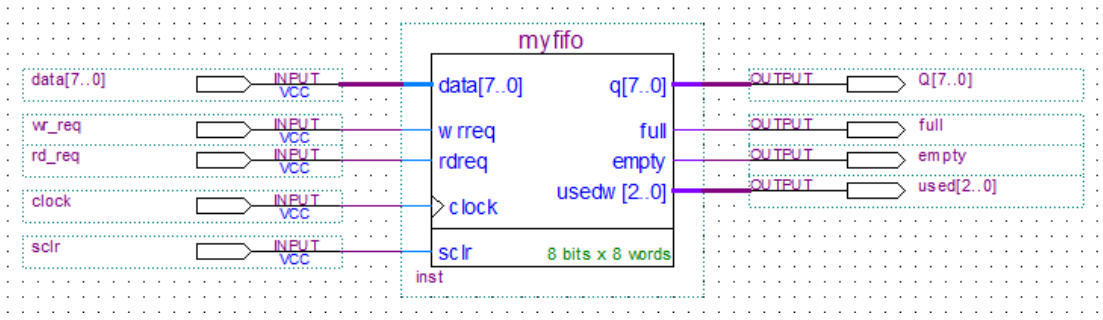


图 23-7　添加端口

10、保存原理图文件后，点击▶图标对工程进行全编译。

11、添加波形向量文件，选择 File->New，弹出图 23-1 所示对话框，在 Verification/Debuging Files 中选择 University Program VWF，点击“OK”。

12、在波形向量文件里添加仿真的输入输出端口，然后设置仿真的激励，如图 23-8 所示。

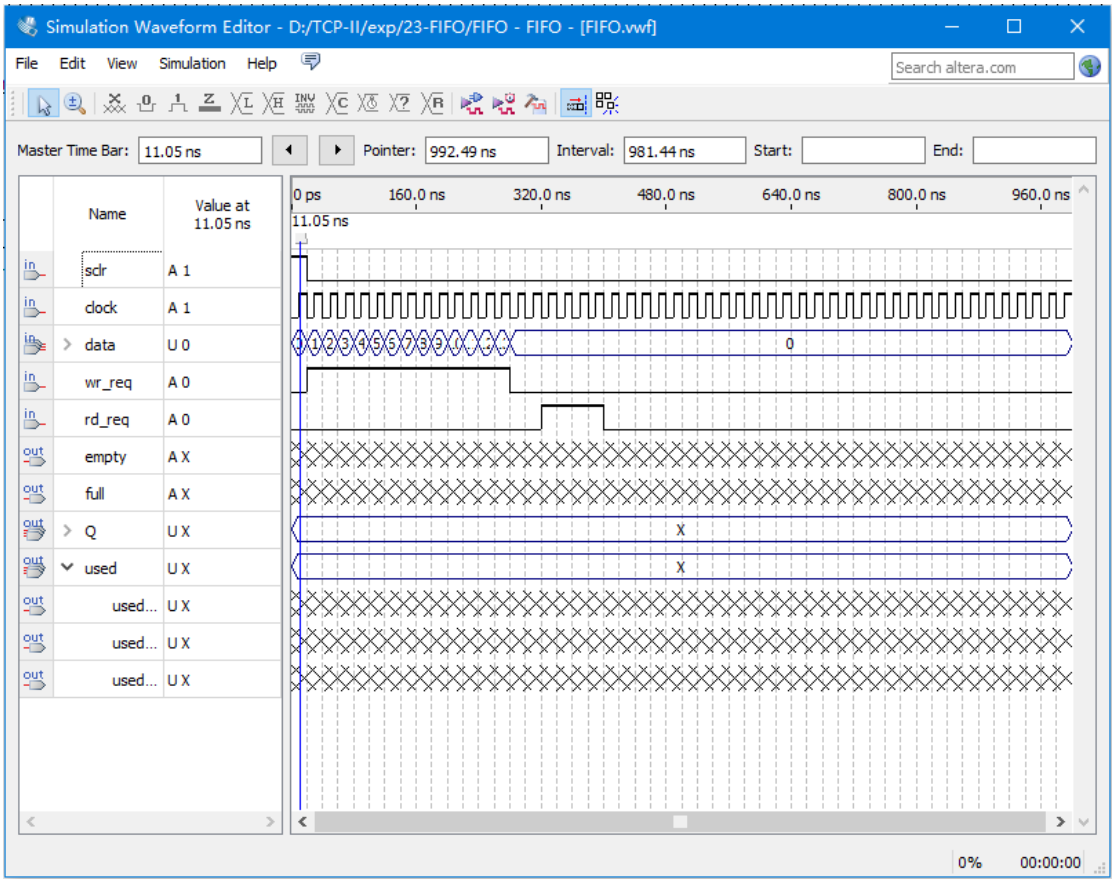
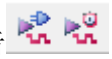


图 23-8　添加端口

13、保存该波形向量文件，然后点击  图标，开始功能/时序仿真，仿真完成后在自动弹出的仿真结果文件中观察仿真结果，仿真结果如图 23-9 所示。

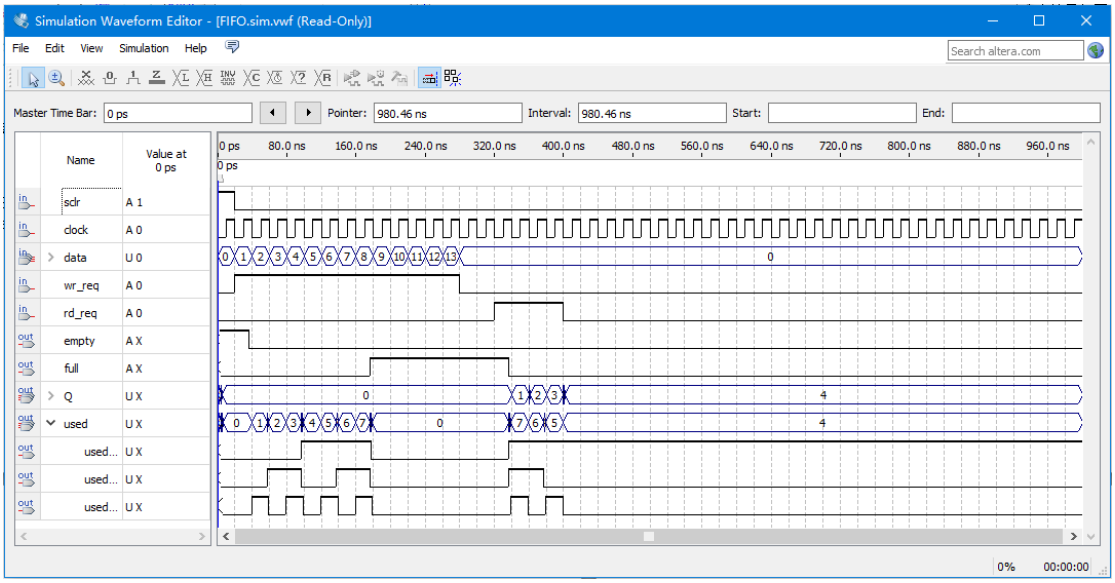


图 23-9 仿真结果

第七章　综合实验

实验二十四、数字钟

一、实验目的

了解数字钟工作原理，设计并实现基于数码管显示的多功能数字钟。

二、实验原理

1、基本原理

多功能数字钟应该具有的功能有：显示时－分－秒、整点报时、小时和分钟可调等基本功能。首先要知道钟表的工作机理，整个钟表的工作应该是在1Hz信号的作用下进行，这样每来一个时钟信号，秒增加1秒，当秒从59秒跳转到00秒时，分钟增加1分，同时当分钟从59 分跳转到00分时，小时增加1小时，但是需要注意的是，小时的范围是从0～23 时。

在实验中为了显示的方便，由于分钟和秒钟显示的范围都是从 0～59，所以可以用一个3 位的二进制码显示十位，用一个四位的二进制码（BCD 码）显示个位，对于小时因为它的范围是从 0～23，所以可以用一个 2 位的二进制码显示十位，用4 位二进制码（BCD 码）显示个位。

实验中由于七段码管是扫描的方式显示,所以虽然时钟需要的是1Hz 时钟信号，但是扫描确需要一个比较高频率的信号，因此为了得到准确的1Hz 信号，必须对输入的系统时钟进行分频。

对于整点报时功能，用户可以根据系统的硬件结构和自身的具体要求来设计。本实验设计的是当进行整点的倒计时5 秒时，让LED 来闪烁进行整点报时的提示。

2、实验任务

本实验的任务就是设计一个多功能数字钟，要求显示格式为 小时－分钟－秒钟，整点报时，报时时间为5 秒，即从整点前5 秒钟开始进行报时提示，LED 开始闪烁，过整点后，停止闪烁。系统时钟选择时钟模块的10KHz，要得到1Hz时钟信号，必须对系统时钟进行 10,000 次分频。调整时间的按键用拨动开关SW0和SW1。SW0调整小时，拨到上端，小时增加，拨到下端，调整停止；SW1调整分钟，拨到上端，分钟增加，拨到下端，调整停止。另外用SW2作为系统时钟复位，复位后全部显示00 00 00。

实验系统中用到的数字时钟源、拨动开关、LED 灯、动态数码管与FPGA的

接口电路和管脚连接在以前的实验中都做了详细说明，这里不在赘述。

三、实验内容

1、新建工程，编写多功能数字钟及数码管显示的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译适配。

To	Assignment Name	Value	Enabled
in Clk	Location	PIN_A11	Yes
out led[0]	Location	PIN_C19	Yes
out led[1]	Location	PIN_D19	Yes
out led[2]	Location	PIN_D17	Yes
out led[3]	Location	PIN_C17	Yes
out Display[0]	Location	PIN_J3	Yes
out Display[1]	Location	PIN_J5	Yes
out Display[2]	Location	PIN_G4	Yes
out Display[3]	Location	PIN_J2	Yes
out Display[4]	Location	PIN_G5	Yes
out Display[5]	Location	PIN_J4	Yes
out Display[6]	Location	PIN_G3	Yes
out SEG_SEL[0]	Location	PIN_E3	Yes
out SEG_SEL[1]	Location	PIN_J6	Yes
out SEG_SEL[2]	Location	PIN_H6	Yes
out SEG_SEL[3]	Location	PIN_H5	Yes
out SEG_SEL[4]	Location	PIN_H7	Yes
out SEG_SEL[5]	Location	PIN_H8	Yes
in Rst	Location	PIN_F2	Yes
in S1	Location	PIN_H1	Yes
in S2	Location	PIN_H2	Yes

时钟 Clk=10KHz

输入端口：RST 对应拨动开关 SW2

S1 对应拨动开关 SW0

S2 对应拨动开关 SW1

输出端口：Display[0:6]对应数码管的段驱动信号 LED7SEG_[A:F]

SEG_SEL[0:5]对应数码管的位驱动信号 LED7SEGDS[0:5]

led[0:3]对应 LED[0:3]

4、下载，观察动态数码管上的时、分、秒显示，用拨动开关 SW0、SW1 调节时、分数值，查看 LED 灯的整点报时。

四、参考程序

略

实验二十五、频率计

一、实验目的

了解数字频率计工作原理，设计并实现基于数码管显示的数字频率计。

二、实验原理

1、基本原理

所谓频率就是周期性信号在单位时间（1s）内变化的次数。若在一定时间间隔 *T*（也称闸门时间）内测得这个周期性信号的重复变化次数为 *N*，则其频率可表示为

$$f=N/T$$

由上面的表示式可以看到，若时间间隔 *T* 取 1s，则 $f=N$。由于闸门的起始和结束的时刻对于信号来说是随机的，将会有一个脉冲周期的量化误差。进一步分析测量准确度：设待测信号脉冲周期为Tx，频率为Fx，当测量时间为T=1s 时，测量准确度为 $\delta=Tx/T=1/Fx$。由此可知这种直接测频法的测量准确度与被测信号的频率有关，当待测信号频率较高时，测量准确度也较高，反之测量准确度较低。因此，这种直接测频法只适合测量频率较高的信号，不能满足在整个测量频段内的测量精度保持不变的要求。若要得到在整个测量频段内的测量精度保持不变的要求，应该考虑待精度频率测量等其它方法。

等精度频率测频的实现方法，可以用图32-1所示的框图来实现。

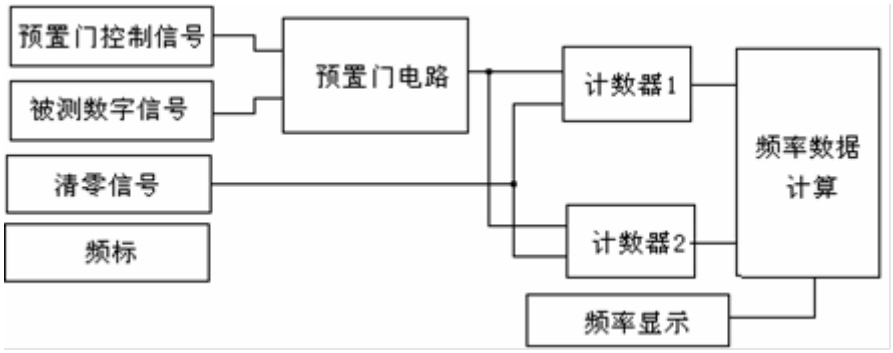


图32-1 等精度测频实现框图

所谓等精度是指该频率计在所测量的整个频段内部，均可实现相同精度的测量，即测量精度与频率无关。图中预置门信号通常为1s。其内部包括一个同步门电路，用来实现被测频标与被测频率的同步，提高测量精度，减少基本误差。该部分与清零脉冲协调工作用来控制两个计数器的启动脉冲。计数器 1 和计数器 2 分别用来给频标和被测数字脉冲计数，设在同步门控制结束时计数器1计数N1，计数器2计数N2，假设频标频率为F1，被测频率位Fx，则可写出公式：

Fx/N2=F1/N1；……（1）

Fx＝（F1/N1）* N2……（2）

由公式可以看出，测量精度与预置门时间无关，主要由 F1的频率稳定度来确定，所以为了提高测量精度，主要是提高频标的频率稳定度，换句话说，测量精度基本上近似于频标的稳定度，若频标的稳定度位 ，则测量误差边可达到。在该电路中，为了确保频标计数与被测频率完全同步（即被测频率的上升沿开始计数，1s 以后，被测频率的下跳沿停止计数），同步门必须由被测信号来控制，设计 方法多种多样，可由学生独立完成。

本实验采用直接测频法进行频率测量。闸门时间固定为 1s，闸门信号是一个 0.5Hz 的方波，在闸门有效（高电平）期间，对输入的脉冲进行计数，在闸门信号的下降沿时刻，所存当前的计数值，并且清零所有的频率计数器。由于闸门时间是 1s（0.5Hz 方波），所以显示的频率是 1s 钟更新一次，且显示的内容是闸门下降沿时锁存的值。

因为闸门时间我们设定为 1s，所以这种频率计仅能测出频率大于或者等于 1Hz 的情况，且频率越高，精度也越高。实际应用中，频率计的闸门时间是个可变量，当频率小于1Hz时，闸门时间就要适当放大。采用一个标准的时钟，在单位时间内如：0.1秒对被测信号的脉冲进行计数，即为信号的频率。

在设计频率计的时候，八个七段码管最多可以显示99,999,999Hz，因此在设计时候用八个 4 位二进制码（BCD 码）来表示，另外还必须有同样的八个 4 位二进制码来对输入的频率进行计数，在闸门下降沿的时候，将后者的值锁存到寄存器中。其信号的时序关系如下图32-2 所示：

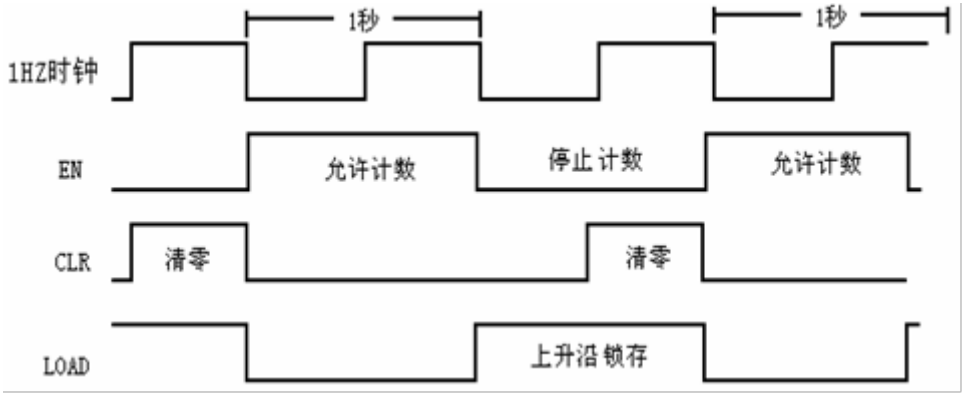


图32-2 控制信号时序关系

2、实验任务

本实验要完成的任务就是设计一个频率计，系统时钟选择核心板上的50M 的时钟，闸门时间为 1s（通过对系统时钟进行分频得到》，在闸门为高电平期间，对输入的频率进行计数，当闸门变低的时候，记录当前的频率值，并将频率

计数器清零，频率的显示每过2 秒刷新一次。被测频率通过一个拨动开关来选择是使用系统中的数字时钟源模块的时钟信号还是从外部通过系统的输入输出模块的输入端输入一个数字信号进行频率测量。当拨动开关为高电平时，测量从外部输入的数字信号，否则测量系统数字时钟信号模块的数字信号。其实现框图如下图32-3 所示：

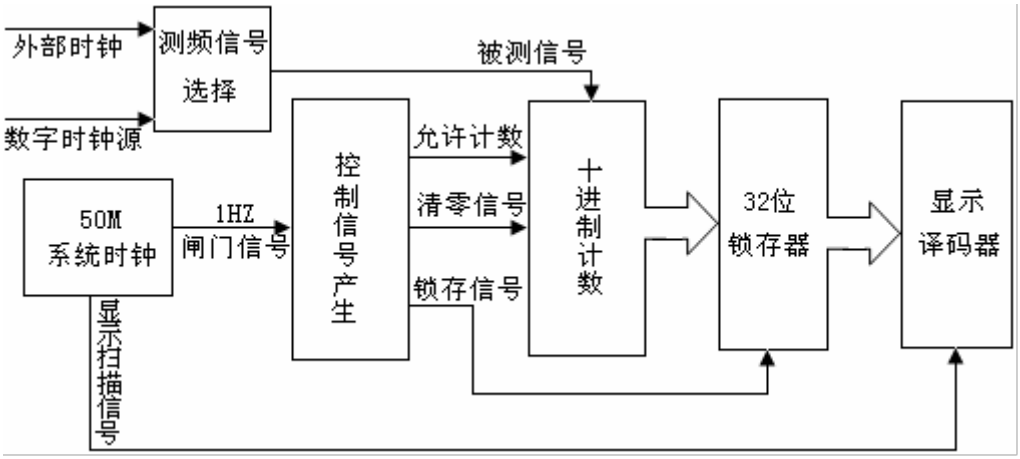


图 32-3　测频实现框图

在本实验中，用到的模块有数字信号源模块、拨动开关模块、50M 系统时钟源模块、数码管显示模块等。其中数码管、数字信号源、拨动开关与 FPGA 的连接电路和管脚连接在以前的实验中都做了详细说明，这里不在赘述。

三、实验内容

1、新建工程，编写数字频率计及数码管显示的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译适配。

To	Assignment Name	Value	Enabled
clk50M	Location	PIN_G1	Yes
clkin1	Location	PIN_A11	Yes
key	Location	PIN_H1	Yes
ledag[0]	Location	PIN_J3	Yes
ledag[1]	Location	PIN_J5	Yes
ledag[2]	Location	PIN_G4	Yes
ledag[3]	Location	PIN_J2	Yes
ledag[4]	Location	PIN_G5	Yes
ledag[5]	Location	PIN_J4	Yes
ledag[6]	Location	PIN_G3	Yes
sel[0]	Location	PIN_E3	Yes
sel[1]	Location	PIN_J6	Yes
sel[2]	Location	PIN_H6	Yes
sel[3]	Location	PIN_H5	Yes
sel[4]	Location	PIN_H7	Yes
sel[5]	Location	PIN_H8	Yes

时钟 Clk=50MHz

输入端口：key 对应拨动开关 SW0

clkin1 对应数字时钟源的输出信号 CLK_CHOOSE

输出端口：ledag[0:6]对应数码管的段驱动信号 LED7SEG_[A:F]

sel[0:5]对应数码管的位驱动信号 LED7SEGDS[0:5]

4、下载，拨动开关 SW0 拨到上端，观察动态数码管上的被测时钟信号的频率值显示。

四、参考程序

略

实验二十六、交通灯

一、实验目的

了解交通灯工作原理，设计并实现基于数码管和 LED 灯显示的简单交通灯控制器。

二、实验原理

1、基本原理

交通灯的显示有很多方式，如十字路口、丁字路口等，而对于同一个路口又有很多不同的显示要求，比如十字路口，车辆如果只要东西和南北方向通行就很简单，而如果车子可以左右转弯的通行就比较复杂，本实验仅针对最简单的南北和东西直行的情况。

要完成本实验，首先必须了解交通路灯的燃灭规律。本实验需要用到实验系统上的发光二极管。依人们的交通常规，“红灯停、绿灯行、黄灯提醒”。其交通灯的燃灭规律为：初始态是两个路口的红灯全亮之后，东西路口的绿灯亮，南北路口的红灯亮，东西方向通车，延时一段时间后，东西路口绿灯灭，黄灯开始闪烁。闪烁若干次后，东西路口红灯亮，而同时南北路口的绿灯亮，南北方向开始通车，延时一段时间后，南北路口的绿灯灭，黄灯开始闪烁。闪烁若干次后，再切换到东西路口方向，重复上述过程。

在实验中使用八个七段码管中的任意两个数码管显示时间。东西路和南北路的通车时间均设定为30 秒。数码管的时间总是显示为29、28、27……2、1、0、29、28……。在显示时间小于5秒的时候，通车方向的黄灯闪烁。

2、实验任务

本实验要完成任务就是设计一个简单的交通灯控制器，交通灯显示用实验系统的LED灯和七段码管中的任意两个来显示。系统时钟选择时钟模块的1KHz时钟，黄灯闪烁时钟要求为2Hz，七段码管的时间显示为1Hz 脉冲，即每1秒中递减一次，在显示时间小于5秒的时候，通车方向的黄灯以2Hz的频率闪烁。系统中用核心板上的按键K5进行复位。

实验系统中用到的数字时钟模块、按键开关、数码管与 FPGA 的接口电路，以及数字时钟源、按键开关、数码管与 FPGA 的管脚连接在以前的实验中都做了详细说明，这里不再赘述。

三、实验内容

1、新建工程，编写简单交通灯控制器及数码管显示的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译适配。

To	Assignment Name	Value	Enabled
in Clk	Location	PIN_A11	Yes
out Display[0]	Location	PIN_J3	Yes
out Display[1]	Location	PIN_J5	Yes
out Display[2]	Location	PIN_G4	Yes
out Display[3]	Location	PIN_J2	Yes
out Display[4]	Location	PIN_G5	Yes
out Display[5]	Location	PIN_J4	Yes
out Display[6]	Location	PIN_G3	Yes
out SEG_SEL[0]	Location	PIN_E3	Yes
out SEG_SEL[1]	Location	PIN_J6	Yes
out SEG_SEL[2]	Location	PIN_H6	Yes
out SEG_SEL[3]	Location	PIN_H5	Yes
out G1	Location	PIN_C19	Yes
out Y1	Location	PIN_D19	Yes
out R1	Location	PIN_D17	Yes
out G2	Location	PIN_C17	Yes
out Y2	Location	PIN_E16	Yes
out R2	Location	PIN_G15	Yes
in Rst	Location	PIN_E4	Yes

时钟 Clk=1KHz，对应数字时钟源的输出信号 CLK_CHOOSE

输入端口：Rst 对应系统的复位信号 nRESET

输出端口：Display[0:6]对应数码管的段驱动信号 LED7SEG_[A:F]

SEG_SEL[0:3]对应数码管的位驱动信号 LED7SEGDS[0:3]

R1、Y1、G1 依次对应 LED2、LED1、LED0

R2、Y2、G2 依次对应 LED5、LED4、LED3

4、下载，观察动态数码管和 LED 灯的显示变化。

四、参考程序

略

实验二十七、音频录放音

一、实验目的

了解 WM8731 工作原理，设计并实现基于 WM8731 的语音采样与回放。

二、实验原理

WM8731 是 Wolfson 公司推出的一款高性能、低功耗的音频编解码芯片。其引脚分配如图 34-1 所示。

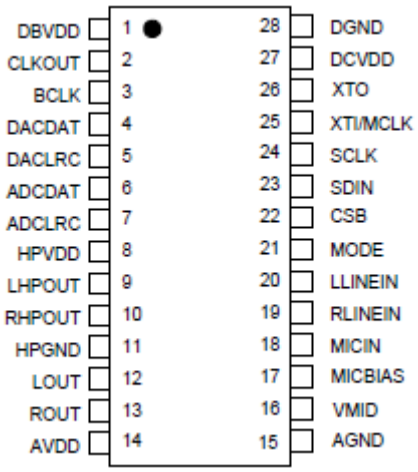


图 34-1　WM8731 引脚图

MICIN 是来自麦克风的模拟量输入，LLINEIN 和 RLINEIN 分别是左右线级音频信号输入，LHPOUT 和 RHPOUT 分别是左右音响的电压输出，LOUT 和 ROUT 分别是左右线级音频信号输出，ADCLRC 和 DACLRC 分别是 WM8731 内部 ADC 和 DAC 的左右声道采样标志位，ADCDAT 和 DACDAT 分别是 WM8731 内部 ADC 和 DAC 的采样数据。它的通信方式是串行的。WM8731 具有较完备的语音输入输出电路。其片内的 ADC 和 DAC 是独立的，可以分别工作在不同的频率（8 KHz 到 96 KHz）。WM8731 的内部电路有很多种工作模式，可以适应不同的工作场合。对其工作模式的控制是通过在向其内部寄存器的写入不同的值来实现的。

WM8731 的内部结构如图 34-2 所示。从图中可以看出，模拟音频信号在输出时，经过一个加法器，此加法器的前端有三个通道，这三个通道分别来自于 DAC、MICIN、LINEIN。然而，这三个通道的信号都可以被阻断，这取决于 WM8731 电路中相关寄存器的值。对于这几个通道，我们还可以分别设置它们的增益，这也是通过控制 WM8731 内部寄存器的值来实现的。WM8731 的音频输出有两种方式，一种是线级电平输出，用来驱动耳机（图中的 LOUT 和 ROUT）；另外一种是喇叭输出，用来驱动无源音响设备(图中的 LHPOUT 和 RHPOUT)。同样的，我们也可通过设置 WM8731 内部某些寄存器的值来改变输出增益的大小。

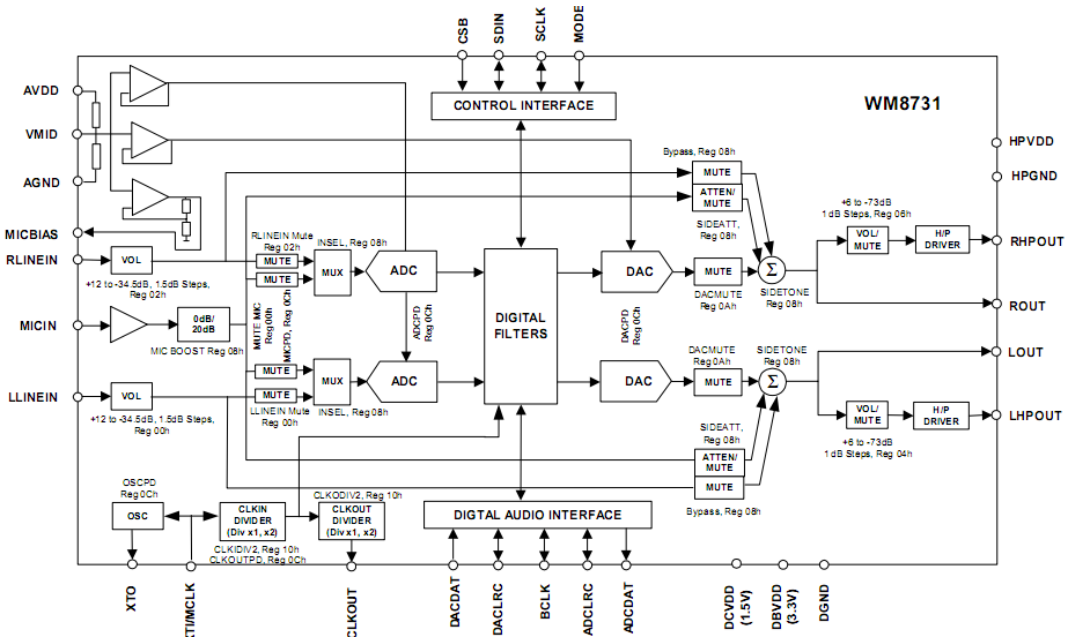


图 34-2　WM8731 内部结构图

在 ADC 电路的前端，有一个选择器，它决定 AD 转换的信号来自 MIC 输入还是 LINEIN 输入。在 AD 转换之前，电路还可以改变信号的增益。电路中的 ADC、DAC 和 Digtal Filters 模块也是 WM8731 电路的重要部分，它们直接影响器件性能的好坏。本实验的重点是实现不同的音频接口设计，表 34-1 中反应的是 WM8731 内部模拟通道设置寄存器的值与不同接口的对应情况。

0000100 Analogue Audio Path Control	0	MICBOOST	0	Microphone Input Level Boost 1 = Enable Boost 0 = Disable Boost
	1	MUTEMIC	1	Mic Input Mute to ADC 1 = Enable Mute 0 = Disable Mute
	2	INSEL	0	Microphone/Line Input Select to ADC 1 = Microphone Input Select to ADC 0 = Line Input Select to ADC
	3	BYPASS	1	Bypass Switch 1 = Enable Bypass 0 = Disable Bypass
	4	DACSEL	0	DAC Select 1 =Select DAC 0 = Don't select DAC
	5	SIDETONE	0	Side Tone Switch 1 = Enable Side Tone 0 = Disable Side Tone
	7:6	SIDEATT[1:0]	00	Side Tone Attenuation 11 = -15dB 10 = -12dB 01 = -9dB 00 = -6dB

表 34-1　WM8731 内部模拟通道寄存器

如表 34-1 所示，纵向第 1 栏中的“0000100”是寄存器的地址，第 2 栏中的数字代表寄存器值的第 n 位。MICBOOST 是麦克风输入增强，MUTEMIC 是麦克风静

音使能，INSEL 是 ADC 输入选择端，BYPASS 线级电平输入至输出选通端，SIDETONE 是麦克风输入至输出选通端，SIDEATT 是 SIDETONE 通道的增益。

要实现不同的音频接口，需要在 WM8731 在上电向其内部寄存器写入特定的值，是他工作在我们需要的状态，我们把这个过程成为对 WM8731 的编程。WM8731 的编程方式有两种:一种是 3 线方式，一种是 2 线方式。当 WM8731 的第 21 脚 MODE 的输入为高电平时，对其编程时需采用 3 线控制方式；当 MODE 的输入为低电平时，对其编程需采用 2 线控制方式。

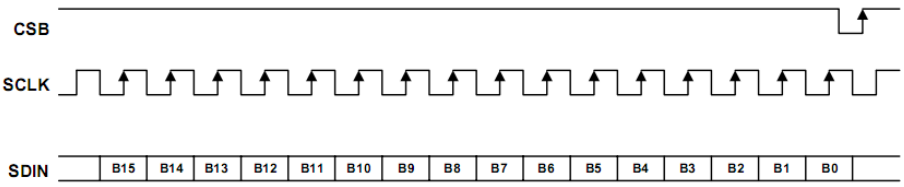


图 34-3　3 线编程方式

3 线编程方式的时序如图 34-3 所示，编程时用到 WM8731 的 SCB、SCLK、SDIN 管脚：SDIN 是输入的串行数据；B[15..9]代表寄存器的地址位；B[8..0]代表将要写入寄存器内的数据；SCK 时串行数据的同步时钟，CSB 的上升沿时，所有串行输入数据被 WM8731 锁存。

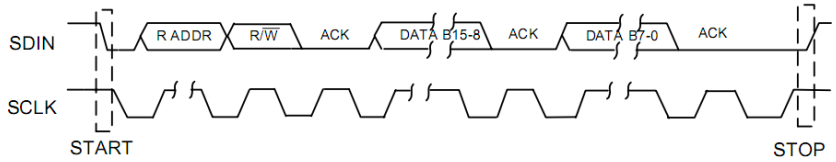


图 34-4　2 线编程方式

实验系统上的 MODE 端是接地的，所以我们需要采用 2 线编程方式。2 线编程方式的时序如图 34-4 所示。 编程时用到 WM8731 的 SCLK、SDIN 管脚：SDIN 是输入的串行数据； B[15..9]代表寄存器的地址位；B[8..0]代表将要写入寄存器内的数据；SCK 是串行数据的同步时钟。

从图 34-4 中可以看出这一总线控制方式属于 I^2C 总线控制方式。数据在传输时需要一个开始状态，当 SCLK 为高电平时，若检测到 SDIN 由高电平变为低电平，便开始通信（start 时间段）。SDIN 变为低电平后，随之而来的将是器件的通信地址 RADDR。WM8731 的通信地址有两个，由此时的 SCB 端口电平控制：当 CSB 的电平为高电平时，RADDR 应该是 0011011；当 CSB 的电平为低电平时，RADDR 应该是 0011010。当 WM8731 收到正确的地址（“0011010”）后，会将其 SDIN 脚设为低电平（对应第一个 ACK 时间段），如此时主控器件若检测到此低电平，则表示 WM8731 器件已收到正确的地址，可以继续通信。否则结束本次通信，返回 START 时刻的状态开始准备下一次通信。第 1 个 ACK 之后，主控器件开始向 SDIN 端口发送数据位 B[15..8]，发送完毕后，进入第 2 个 ACK 时间段，此时 WM8731

会对前一段时间接收的数据位进行一次回应，若器件收到正确的数据则回应 0，否则回应 1。第 2 个 ACK 期间，主设备在 SCLK 上升沿到来时，会检测 SDIN 端口状态，以确认 WM8731 接收的是正确的数据，否则将终止本次通信，返回 start 状态准备再次通信。若数据位 B[15..8]被成功地发送出，主控器件会在下一周期继续开始发送 B[7..0]位，数据位发送完毕后，进入第 3 个 ACK 时间段，此期间的通信跟第 2 个 ACK 时间段一样。第 3 个 ACK 结束后，便可结束通信，通信结束的标志是 SCLK 的高电平期间时，检测到 SDIN 端口电平由低电平变为高电平。

WM8731 有两种方式产生工作时钟，一种为 NORMAL MODE，另一种为 USB MODE。在 NORMAL 模式下，WM8731 通过器件内部的晶振系统产生工作时钟，并通过此时钟产生 ADC 和 DAC 模块的采样时钟。此种模式下，WM8731 的几种采样率都是符合工业标准的，因此也比较固定。在 USB 模式下，需要为 WM8731 提供工作时钟 MCLK,在这种模式下，WM8731 也可以产生这几种标准的采样时钟。此种模式是比较灵活的，我们可以在不用再次对 WM8731 编程的前提下，实现采样率的改变，此时只需改变它的工作时钟即可。表 34-2 是 USB 模式下，当 MCLK 为 12.00MHz 时，相关存储器的值与不同采样率的对应情况。

SAMPLING RATE		MCLK FREQUENCY	SAMPLE RATE REGISTER SETTINGS					DIGITAL FILTER TYPE
ADC	DAC							
kHz	kHz	MHz	BOSR	SR3	SR2	SR1	SR0	
48	48	12.000	0	0	0	0	0	0
44.1 (Note 2)	44.1 (Note 2)	12.000	1	1	0	0	0	1
48	8	12.000	0	0	0	0	1	0
44.1 (Note 2)	8 (Note 1)	12.000	1	1	0	0	1	1
8	48	12.000	0	0	0	1	0	0
8 (Note 1)	44.1 (Note 2)	12.000	1	1	0	1	0	1
8	8	12.000	0	0	0	1	1	0
8 (Note 1)	8 (Note 1)	12.000	1	1	0	1	1	1
32	32	12.000	0	0	1	1	0	0
96	96	12.000	0	0	1	1	1	3
88.2 (Note 3)	88.2 (Note 3)	12.000	1	1	1	1	1	2

表 34-2　USB 模式下不同采样率对照表

事实上，这些采样频率均是 WM8731 内部的时钟电路根据不同的寄存器组合，对主时钟 MCLK 分频后得到的。举个例子，在 USB 模式下 ，若 BOSR=0，SR[3..0]=“0000”，MCLK=24MHz,那么 ADC 和 DAC 得采样率均为 96KHz。

另外，值得一提的是，WM8731 在进行 A/D 和 D/A 转换时，采用的数据传输模式。WM8731 在设计时，为音频接口电路提供了 4 种数据传输模式：Right justified、Left justified、I2S 和 DSP MODE。每种模式都是串行通信的，只是在数据传输时，有效数据的相对位置不一样。本试验仅介绍 I2S 的传输模式，

图 34-5 是 I²S 通信模式的时序图。

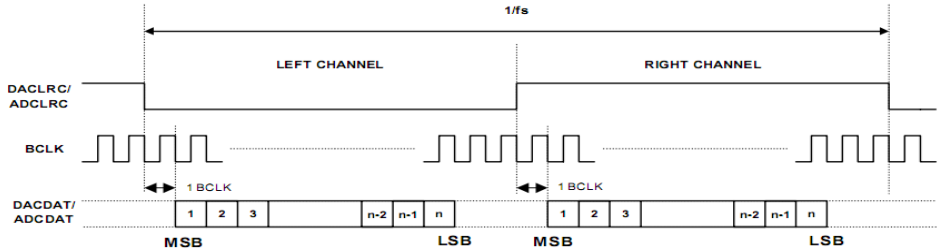


图 34-5　I²S 通信模式时序图

图 34-5 中 DACLRC/ADCLRC 表示当前传输的声道（高电平代表右声道，低电平代表左声道），跟其它模式一样，每个声道数据传输时 MSB 在前，LSB 在后。不同的是 MSB 出现在每声道开始的第 2 个下降沿（上升沿读取数据较为安全），后面紧跟的是次高位，…，最后是最低位。在 USB 模式下，BCLK 的频率始终等于 MCLK 的频率。由于 WM8731 的 ADC 和 DAC 的数字信号都是串行的。因此，在 ADC 的数据输出端和 DAC 的数据输入端分别需要数据进行串并转换和并串转换。

三、实验内容

1、新建工程，编写基于 WM8731 的语音采样与回放的 VHDL 程序。

2、分析综合，用 RTL Viewer 观察电路综合结果。

3、引脚锁定，编译适配。

To	Assignment Name	Value	Enabled
in AUD_ADCDAT	Location	PIN_B19	Yes
in AUD_ADCLRC	Location	PIN_A19	Yes
in AUD_BCLK	Location	PIN_A17	Yes
out AUD_DACDAT	Location	PIN_B18	Yes
in AUD_DACLRC	Location	PIN_A18	Yes
io AUD_I2C_DAT	Location	PIN_B20	Yes
out AUD_I2C_SCK	Location	PIN_A20	Yes
out AUD_MCLK	Location	PIN_G11	Yes
in KEY[1]	Location	PIN_R14	Yes
in KEY[0]	Location	PIN_T14	Yes
in CLK50M	Location	PIN_G1	Yes

时钟 CLK=50MHz

输入端口：KEY[0:1]对应轻触按键 KEY1、KEY2

AUD_ADCDAT 对应 WM8731 的 AUDIO_ADCDAT

AUD_ADCLRC 对应 WM8731 的 AUDIO_ADCLRC

AUD_BCLK 对应 WM8731 的 AUDIO_BCLK

AUD_DACLRC 对应 WM8731 的 AUDIO_DACLRC

输出端口：AUD_DACDAT 对应 WM8731 的 AUDIO_DACDAT

AUD_MCLK 对应 WM8731 的 AUDIO_MCLK

AUD_I2C_SCK 对应 WM8731 的 I2C_SCLK

双向信号：AUD_I2C_DAT 对应 WM8731 的 I2C_SDAT

4、用音频线连接电脑音频输出插孔与实验系统音频模块的 LINEIN 插孔，用耳机连接实验系统音频模块的 LINEOUT 插孔；下载，用电脑播放 MP3 音乐，听插到实验系统上的耳机中是否有音乐输出；用轻触按键 KEY1、KEY2 调节音乐的音量输出大小。

四、参考程序

略