# 数字电子技术实验平台

## 硬件平台简介

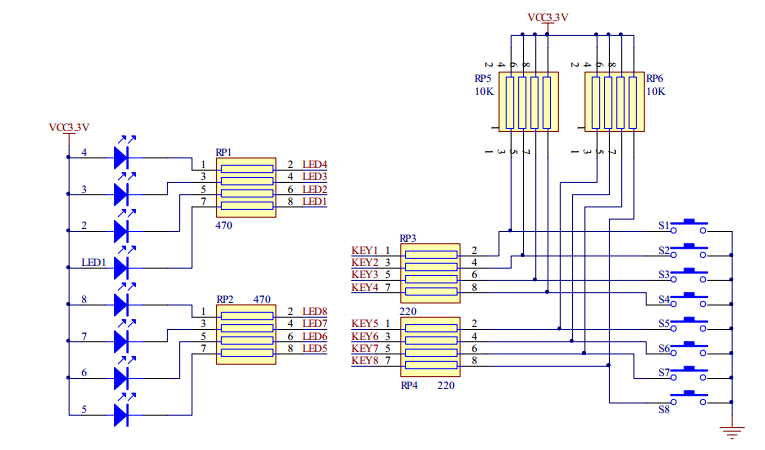
数字电子技术实验平台采用国际著名可编程逻辑器件公司Altera的Cyclone II系列芯片为核心，整个平台采用模块化设计，各种模块可以自由组合，同时提供丰富的扩展接口。开发工程师可利用VHDL语言、Verilog语言、原理图或方程式，结合Altera集成开发环境Quartus II，进行编辑、综合、仿真和布局布线，通过加载配置并进行设计验证。它可以满足绝大多数组合逻辑电路、时序逻辑电路设计需求。

## 开发板功能特点

* 提供了5V、3.3V板上电源
* 50 MHz的有源晶振，满足基本设计要求
* 208管脚封装， I/O资源丰富
* 芯片型号为Cyclone EP2C8Q208C8N
* 支持JTAG加载模式
* 8个7段数码管
* 8个用户LED灯
* 4×4键盘矩阵
* 8个开关量输入端口
* 蜂鸣器
* 独立矩形波发生器

## 功能模块介绍

### LED灯和8个按键开关

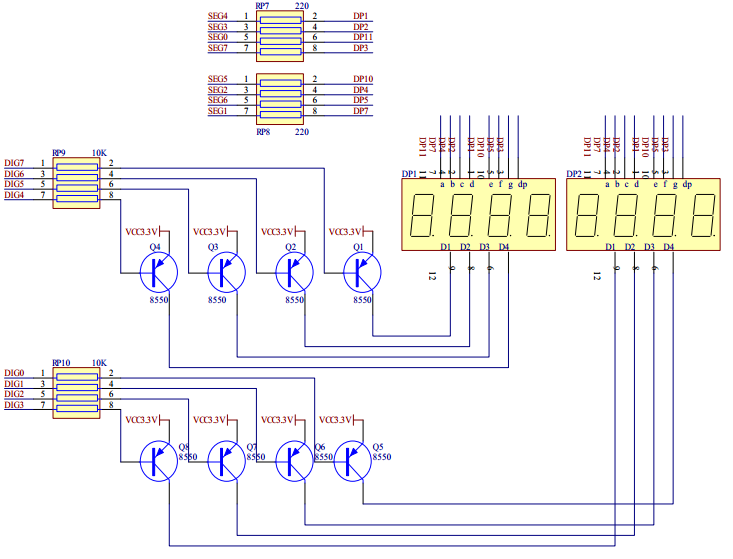


**图1-1.独立按键和LED电路**

### 八位数码管显示

8个数码管为共阳极接入，用8个三极管分别对其进行驱动。数码管的8个显示控制端分两组，每组4个共用，通过三极管基极选取当前显示的数码管。数码管的8段a-h的位置及与控制端的对应如所示，因此，若显示数字“1”，则控制字为“11111001”，其它字符以此类推。

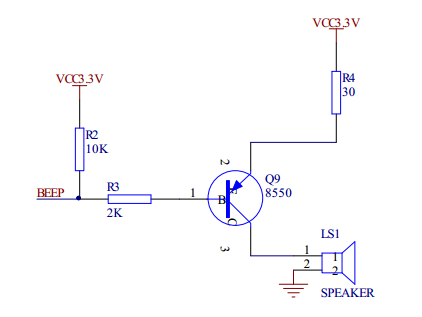
|  |  |  |
| --- | --- | --- |
|  | FPGA引脚 | LED字段 |
| 170 | a |
| 165 | b |
| 169 | c |
| 175 | d |
| 173 | e |
| 168 | f |
| 164 | g |
| 171 | h |



**图1-2.八位数码管电路**

### 蜂鸣器

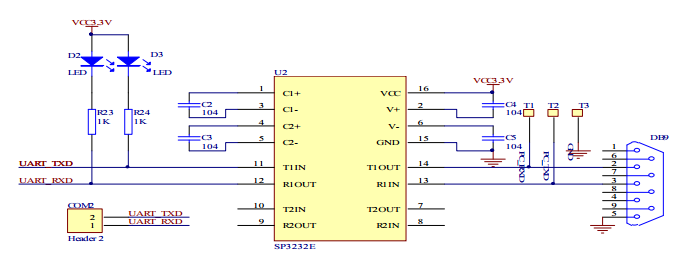
可作为报警信号发生器，用一个三极管做驱动。



**图1-3.蜂鸣器电路**

### 串口通信电路

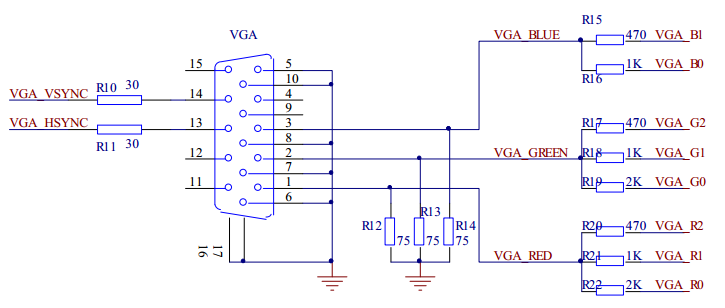
作为UART数据的收发。

****

**图1-4.RS232串口电路**

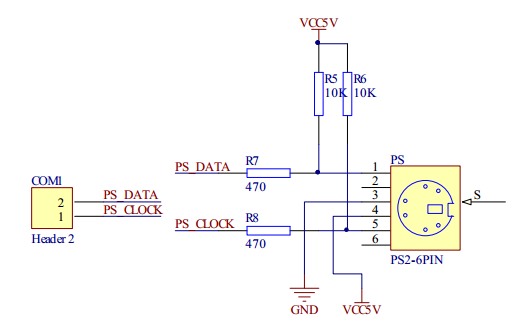
### VGA显示电路

作为VGA显示器的驱动接口。

**图1-5.VGA接口电路**

### PS/2接口电路

作为PS/2键盘或鼠标的驱动接口。

****

**图1-6.PS/2接口电路**

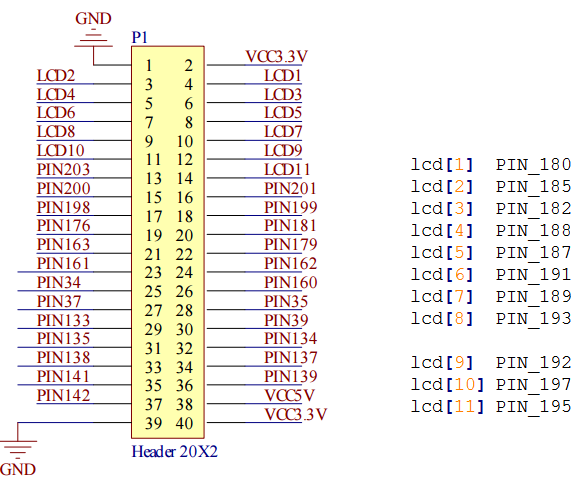
### A/D、D/A接口电路

用于完成数字量与模拟量之间的转换。

## 

**图1-7.D/A、A/D电路**

用于数字量的扩展接口电路

****

**图1-8.扩展接口电路**

**引脚分配表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 全局信号 | | 蜂鸣器 | | LCD | |
| 引脚号 | 信号名称 | 3 | beep | 180 | lcd[1] |
| 23 | 全局时钟 | 串行口 | | 185 | lcd[2] |
| 131 | 全局复位 | 143 | RXD | 182 | lcd[3] |
| 发光二极管 | | 144 | TXD | 188 | lcd[4] |
| 142 | led[1] | A/D | | 187 | lcd[5] |
| 141 | led[2] | 31 | adc\_clk | 191 | lcd[6] |
| 139 | led[3] | 33 | cs\_n | 189 | lcd[7] |
| 138 | led[4] | 30 | sdat\_in | 193 | lcd[8] |
| 137 | led[5] | D/A | | 192 | lcd[9] |
| 135 | led[6] | 47 | dac\_clk | 197 | lcd[10] |
| 134 | led[7] | 48 | dac\_data | 195 | lcd[11] |
| 133 | led[8] | 46 | dac\_ldac | PS/2接口 | |
| 按钮开关 | | 45 | dac\_load | 145 | ps2\_clk |
| 107 | S1 | VGA | | 146 | ps2\_dat |
| 108 | S2 | 5 | hsync |  |  |
| 110 | S3 | 4 | vsync |  |  |
| 112 | S4 | 14 | vga\_r[0] |  |  |
| 113 | S5 | 15 | vga\_r[1] |  |  |
| 114 | S6 | 12 | vga\_r[2] |  |  |
| 115 | S7 | 13 | vga\_g[0] |  |  |
| 116 | S8 | 10 | vga\_g[1] |  |  |
| 拨动开关 | | 11 | vga\_g[2] |  |  |
| 128 | K1 | 6 | vga\_b[0] |  |  |
| 127 | K2 | 8 | vga\_b[1] |  |  |
| 118 | K3 |  |  |  |  |
| 117 | K4 |  |  |  |  |
| LED数码管-位码 | | LED数码管-段码 | |  |  |
| 150 | dig[1] | 170 | a |  |  |
| 149 | dig[2] | 165 | b |  |  |
| 152 | dig[3] | 169 | c |  |  |
| 151 | dig[4] | 175 | d |  |  |
| 205 | dig[5] | 173 | e |  |  |
| 206 | dig[6] | 168 | f |  |  |
| 207 | dig[7] | 164 | g |  |  |
| 208 | dig[8] | 171 | dot |  |  |

# Quartus II 简介

## Quartus II软件

Quartus II软件是Altera公司的综合开发工具，它集成了Altera的FPGA/CPLD（复杂可编程逻辑器件Complex Programmable Logic Device,CPLD/现场可编程器件Field Programmable Gate Array,FPGA）开发流程中所涉及的所有工具和第三方接口。通过使用此综合开发工具，设计者可以创建、组织和管理自己的设计。

## Quartus II软件的设计流程

Quartus II 软件为设计流程的每个阶段提供Quartus II 图形用户界面、EDA 工具界面以及命令行界面。用户可以在整个流程中只使用这些界面中的一个，也可以在设计流程的不同阶段使用不同界面。下图1-1所示是利用Quartus II完成应用开发的流程。

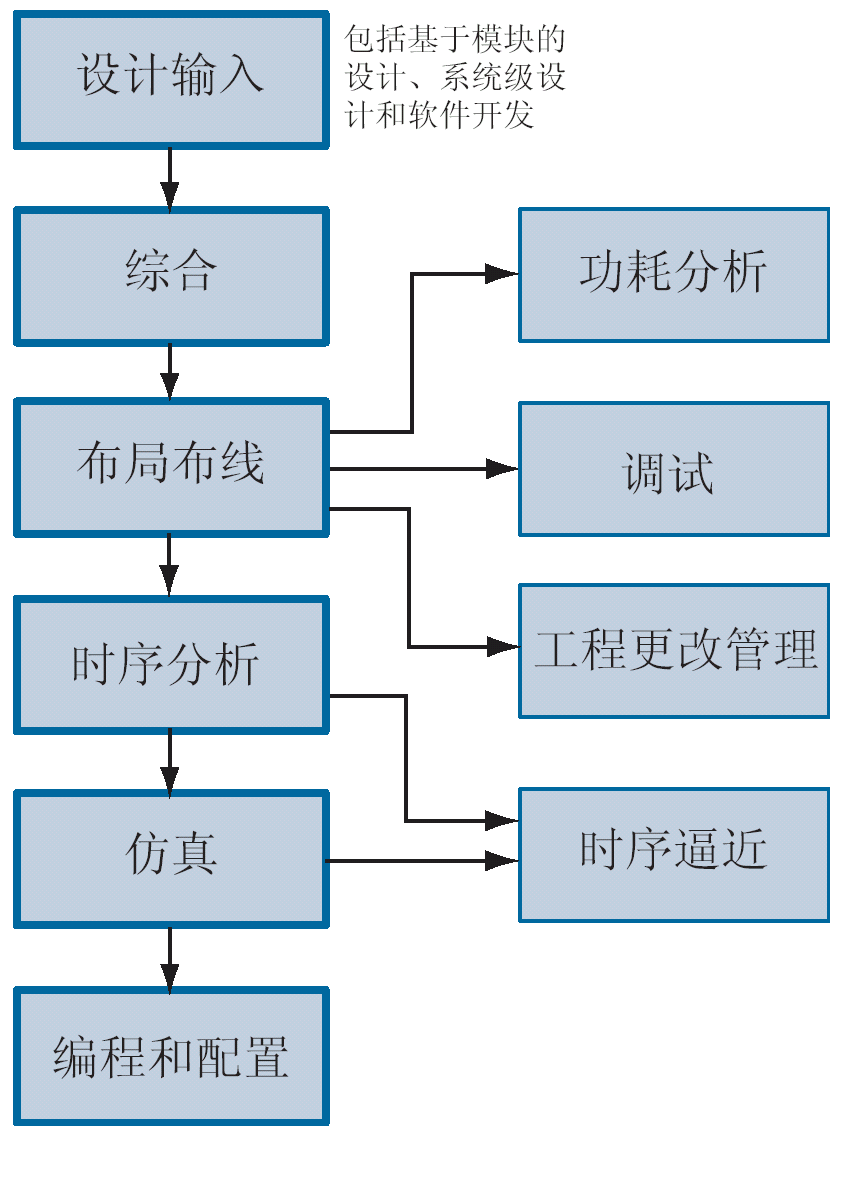


图2-1 设计流程

1. 设计输入

设计输入是CPLD／FPGA开发阶段的第一步．它完成了器件的硬件描述。Quartus II提供了以下多种设计输入的工具。

● 文本编辑器

利用该工具完成硬件描述语言程序文件的编写、修改与保存。

● 块与符号编辑器

该工具用于查看和编辑代表宏功能、宏功能模块、基本单元或设计文件的预定义符号。

● MegaWizard插件管理器

Quartus II提供了许多Altera公司的宏功能模块，可以在设计文件中与门和触发器等基本单元一起使用。利用MegaWizard插件管理器可将这些功能强大的宏功能模块插入到设计中。

2. 综合

Quartus II提供了如下综合工具：

● 分析和综合器

该工具调用了Quartus II的内置综合器，它支持最新版本的Verilog和VHDL，并最终生成EDIF网表文件(.edf)或VQM文件(.vqm)。

● 辅助工具

Quartus II在设计综合期间提供了辅助工具，用于检查设计的可靠性。

● RTL查看器

利用该工具可查看硬件描述最终的综合结果，并给出形象的电路元器件图表。

3. 布局连线

布局连线是将设计综合后的网表文件映射到实体器件的过程。该过程包括：将设计工程的逻辑和时序要求与器件的可用资源相匹配；将每个逻辑功能分配给最好的逻辑单元位置，进行布线和时序分析；选择相应的互连路径和引脚分配。Quartus II提供了以下丰富的布局连线工具。

● Fitter工具

如果设计者利用约束编辑器指定了编译约束条件，那么Fitter工具试图将设计约束与器件上的资源相匹配，并努力满足约束条件，然后试图优化设计中的其余逻辑。如果设计者未指定任何约束条件，那么Fitter工具将自动优化设计。

● 约束编辑器

利用该工具可为设计指定初始约束条件，例如引脚分配、器件选项、逻辑选项和时序等约束条件。

● 布局图编辑器

利用该工具可以查看上一次编译期间执行的资源分配和布线结果(该结果是只读的，不可通过编辑器进行更改)。

● 芯片编辑器

利用该工具可以显示芯片内部完整的布线信息，显示每个器件资源之间的所有可能和使用的布线路径。

● 增量布局连线工具

如果设计者所做的更改仅影响少数节点，可利用该工具避免运行全编译。Quartus II的增量布局连线工具将尽量保留以前编译的布局连线结果，以较快的速度完成新的编译。

在布局连线过程中，设计者还会遇到“整体设计工程更改管理”的情况，这种工程更改管理是指在完成全编译之后，使用芯片编辑器查看设计布局布线详细信息，并确定要更改的资源，从而避免了过多地修改设计源文件或Quartus II设置。

4. 时序分析

Quartus II提供了专用的时序分析器，可用于分析设计中的所有逻辑，并有助于指导Fitter工具达到设计的时序要求。时序分析的结果包括fMAX(最大频率)、tSU(时钟建立时间)、tH(时钟保持时间)、tCO(时钟至输出延时)、tPD(引脚至引脚延时)、最小tCO和最短tPD。

5. 仿真

Quartus II提供了功能仿真和时序仿真两种仿真工具，其功能十分强大。设计者视所需的信息类型而定，可以进行功能仿真以测试设计的逻辑功能，也可以进行时序仿真，在目标器件中测试设计的逻辑功能和最坏情况下的时序。在时序仿真过程中，Quartus II可根据设计者提供的向量波形文件( .vwf)、间量表输出文件(.tbl)、向量文件(.vec)和仿真基准文件(.tbl)格式的波形文件进行仿真，输出仿真波形。除此之外，Quartus II还可以估计在时序仿真期间当前设计所消耗的功率。

6. 器件编程与配置

Quartus II编译成功后，设计者就可以对器件进行编程或配置了。器件编程器使用编译过程中的Assembler工具生成的POF和SOF文件对器件进行编程，其编程模式有4种。

● 被动串行模式：该模式可实现对多个器件进行编程。

● JTAG模式：该模式也可实现对多个器件进行编程。

● 主动串行编程模式：该模式可实现对单个串行配置器件进行编程。

● 插座内编程模式：该模式可实现对单个CPLD或配置器件进行编程。

## Quartus II软件的用户界面

Quartus II软件启动后的主界面如图1-2所示，由标题栏、菜单栏、工具栏、资源管理窗、编译状态显示窗、信息显示窗和工程工作区等部分组成。

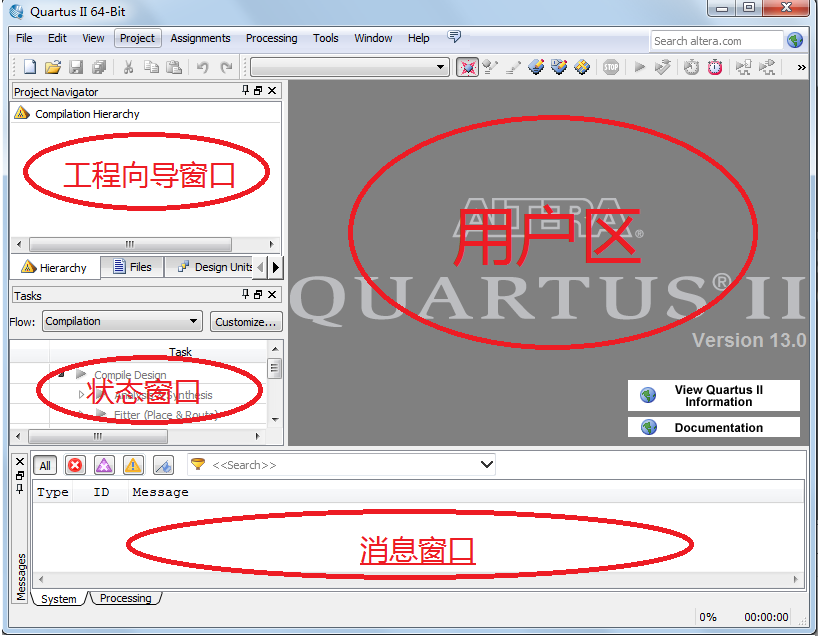


图2-2 Quartus II软件主界面

下面分别介绍各个部分的作用和使用方法。

1. 标题栏

标题栏显示当前工程的路径和程序的名称。

2. 菜单栏

菜单栏主要由文件（File）、视图（View）、工程（Project）、操作（Processing）、资源分配（Assignments）、调试（Debug）、工具（Tools）、窗口（Window）和帮助（Help）等下拉菜单组成。其中工程（Project）、资源分配（Assignments）、操作（Processing）、工具（Tools）集中了Quartus II软件较为核心的全部操作命令，下面分别介绍。

(1) Project菜单

该菜单项主要完成对工程的一些操作。

● 【Add/Remove Files in Project】：添加或新建某种资源文件。

● 【Revisions】：创建或删除工程，在其弹出的窗口中单击“Create…”按钮创建一个新的工程；或者在创建好的几个工程中选中一个，单击“Set Current”按钮，就把选中的工程设置为当前工程。

● 【Archive Project】：为工程归档或备份。

● 【Generate Tcl File for Project】：产生工程的Tcl脚本文件，选择好要生成的文件名以及路径后，单击OK 按钮即可。如果选中了“Open generated file”则会在工程工作区打开该Tcl文件。

● 【Generate Tcl File for Project】：产生功率估计文件。

● 【HardCopy Utilities】：跟HardCopy 器件相关的功能。

● 【Locate】：将Assignment Editor 中的节点或原代码中的信号在Timing Closure Floorplan编译后布局布线图，Chip Editor或原文件中定位其位置。

● 【Hierarchy】:打开工程工作区显示的源文件的上一层或下一层的源文件以及顶层文件。

(2) Assignments菜单

该菜单项的主要功能是对工程的参数进行配置，如管脚分配、时序约束、参数设置等。

● 【Device】：是指目标器件型号。

● 【Assign Pins】：打开分配管脚对话框，给设计的信号分配IO管脚。

● 【Timing Settings】：打开时序约束对话框。

● 【EDA Tool Settings】：设置EDA工具，如Synplify等。

● 【Settings】：打开参数设置页面，可以切换到使用Quartus II软件开发流程的每个步骤所需的参数设置页面。

● 【Wizard】：启动时序约束设置、编译参数设置、仿真参数设置、Software Build参数设置。

● 【Assignment Editor】：分配编辑器，用于分配管脚、设定管脚电平标准、设定时序约束等。

● 【Remove Assignments】：用户可以使用它删除设定的类型的分配，如管脚分配、时序分配、SignalProbe信号分配等。

● 【Demote Assignments】：允许用户降级使用当前较不严格的约束，使编辑器更高效地编译分配和约束等。

● 【Back-Annotate Assignments】：允许用户在工程中反标管脚、逻辑单元、LogicLock区域、节点、布线分配等。

● 【Import Assignments】：给当前工程导入分配文件。

● 【Timing Closure Foorplan】：启动时序收敛平面布局规划器。

● 【LogicLock Region】：允许用户查看，创建和编辑LogicLock区域约束以及导入导出LogicLock 区域约束文件。

(3) processing 菜单

该菜单项包含了对当前工程执行各种设计流程，如开始综合、开始布局布线、开始时序分析等。

(4) Tools菜单

该菜单项调用Quartus II软件中集成的一些工具，如MegaWizard Plug-In manager(用于生成IP和宏功能模块)，ChipEditor、RTL Viewer、Programmer等工具。

3. 工具栏

工具栏中包含了常用命令的快捷图标。将鼠标移到相应图标时，在鼠标下方出现此图标对应的含义，而且每种图标在菜单栏均能找到相应的命令菜单。用户可以根据需要将自己常用的功能定制为工具栏上的图标，方便在Quartus II软件中灵活快速地进行各种操作。

4. 资源管理窗

资源管理窗用于显示当前工程中所有相关的资源文件。资源管理窗左下脚有三个标签，分别是结构层次(Hierarchy)，文件（Files ）和设计单元（Design Units）。结构层次窗口在工程编译之前只显示了顶层模块名，工程编译了一次后，此窗口按层次列出了工程中所有的模块，并列出了每个原文件所有资源的具体情况。顶层可以是用户产生的文本文件，也可以是图形编辑文件。文件窗口列出了工程编译后的所有文件，文件类型有设计器件文件(Design Device Files) 、软件文件(Software Files)和其他文件（Other Files）。设计单元窗口列出了工程编译后的所有单元，如AHDL单元，Verilog单元，VHDL单元等，一个设计器件文件对应生成一个设计单元，参数定义文件没有对应设计单元。

5. 工程工作区

器件设置、定时约束设置、底层编辑器和编译报告等均显示在工程工作区中，当Quartus II实现不同功能时此区域将打开相应的操作窗口，显示不同的内容，进行不同的操作。

6. 编译状态显示窗

编译状态显示窗主要时显示模块综合、布局布线过程及时间。模块（Module）列出工程模块，过程（Process）显示综合、布局布线进度条，时间（Time）表示综合、布局布线所耗费时间。

7. 信息显示窗

信息显示窗显示Quartus II软件综合、布局布线过程中的信息，如开始综合时调用源文件、库文件、综合布局布线过程中的定时、告警、错误等，如果是告警和错误，则会给出具体的引起告警和错误原因，方便设计者查找及修改错误。

## 

## 实验一 基本逻辑门电路

### 实验目的

#### 熟悉Quartus II平台开发环境,了解Verilog HDL语言的设计数字电路的基本流程。

#### 掌握基本逻辑门输入与输出之间的逻辑关系。

### 实验内容

#### 在实验开发板上实现二输入与门电路。

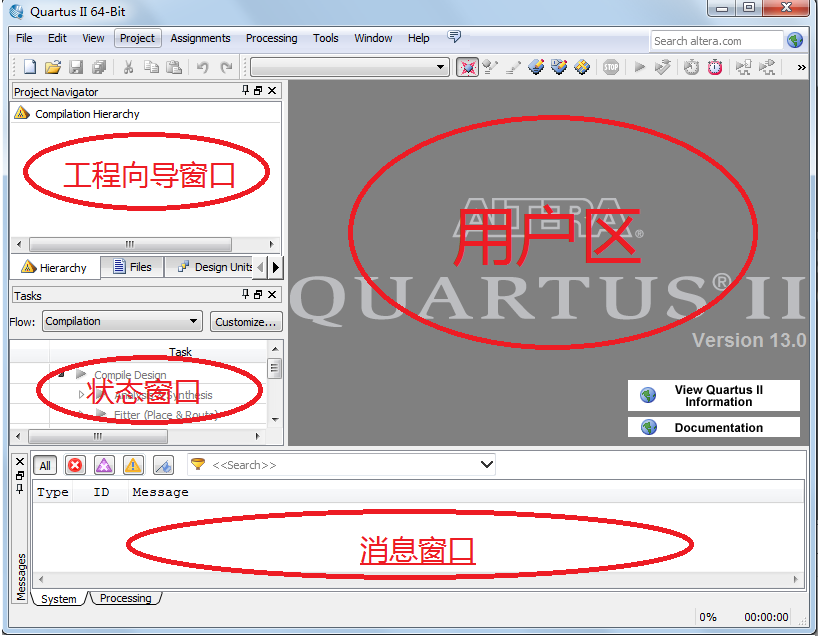
#### 在实验开发板实现异或、与非和或非等基本逻辑门电路。

### 实验原理和步骤

**1. 本章节以二输入与门为例，介绍在QuartusII环境下的开发流程**

(1) 启动quartus II

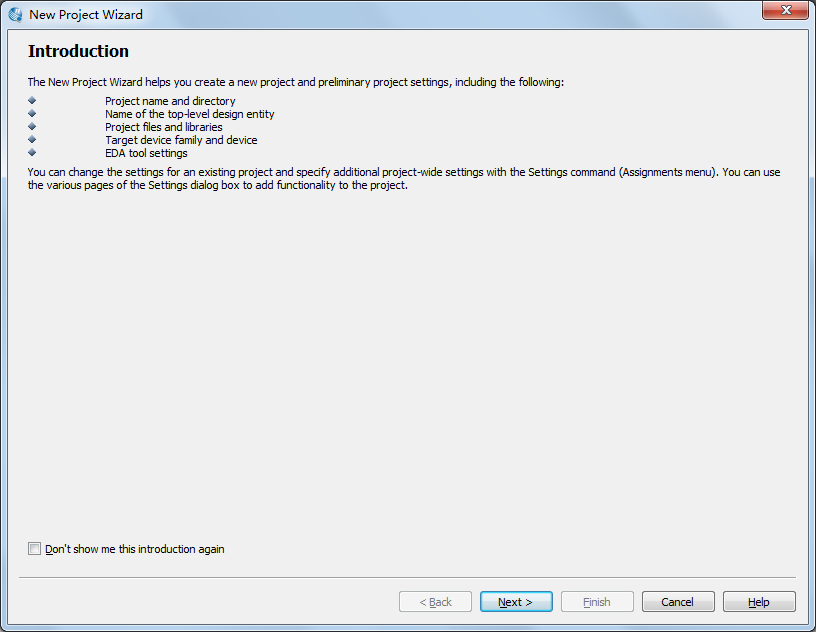
Quartus II 的主界面由：工程向导窗口、窗台窗口、消息窗口和用户区构成。



(2) 利用向导创建项目

在File菜单下选择New Project Wizard选项启动项目向导

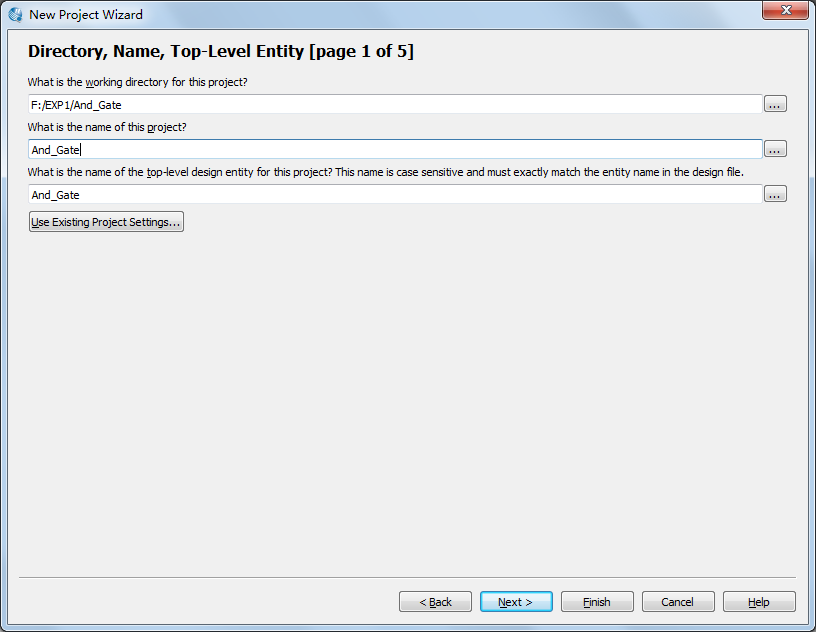
显示如下简介界面，说明新建项目的基本步骤，在此直接点击Next进入下一步。



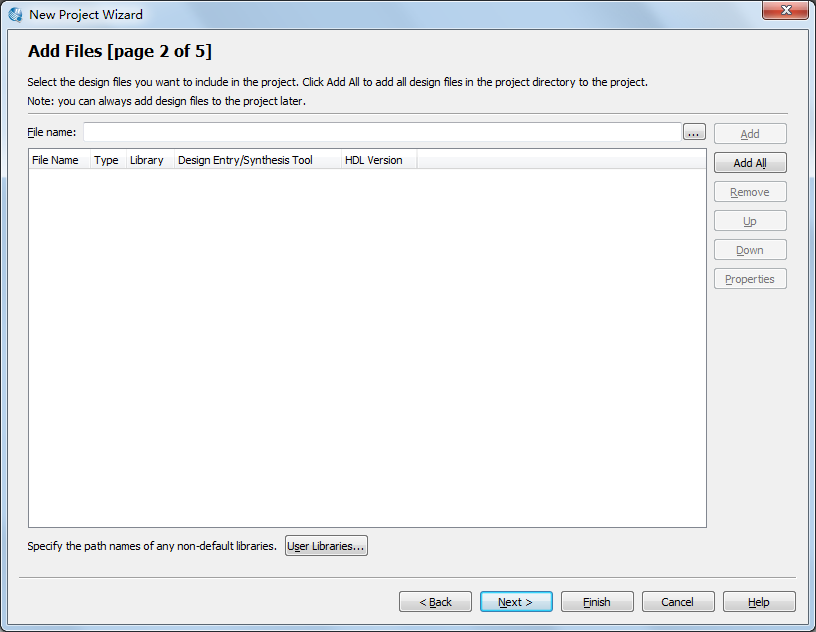
Step1.如下图所示，在此设定项目的存放路径、工程项目名称和顶层实体名称，设定完成后，点击Next进入下一步。

**注意1.关于存放路径：不要放在QuartusII开发平台的系统路径下，请在别处建立用于工程存放文件夹。**

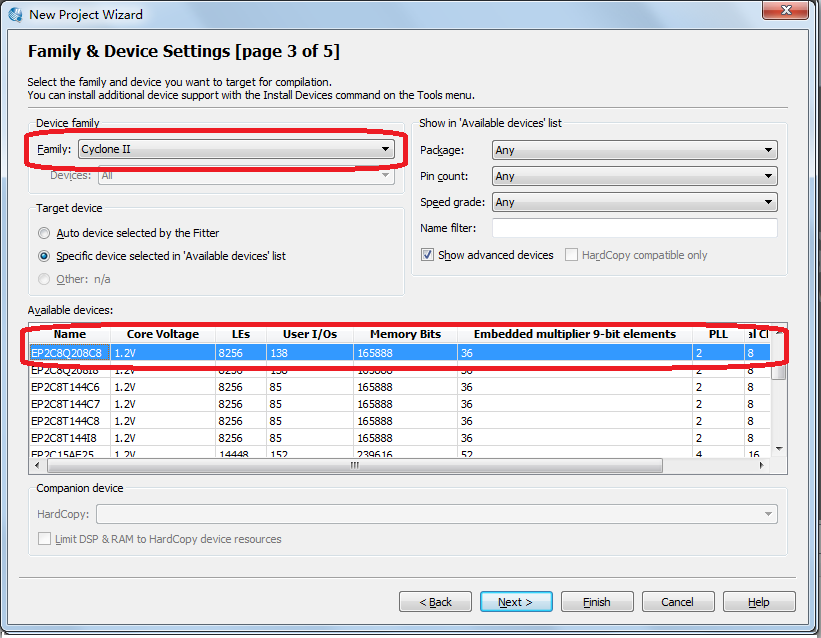
**2.上面三者的名称可由字母、数字和下划线构成，必须字母开头，不要使用汉字。**



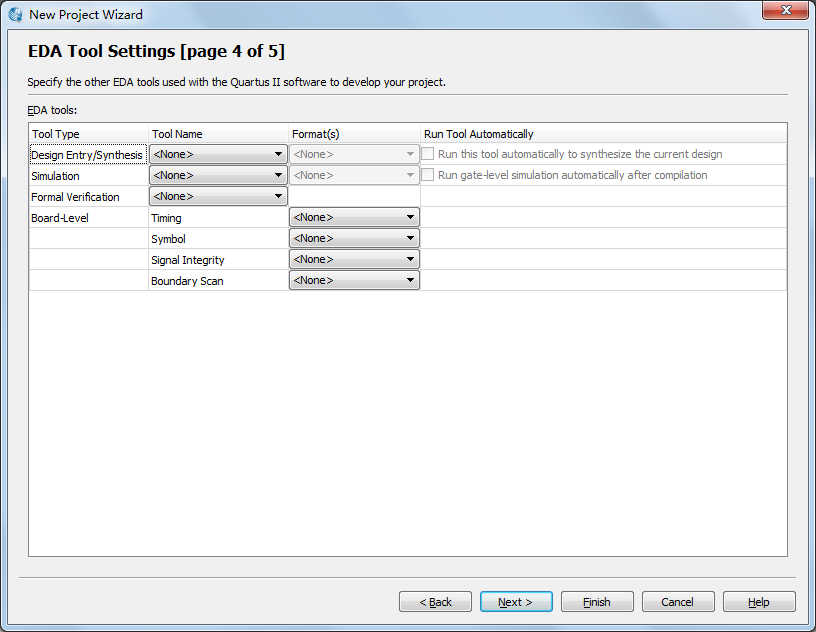
Step2. 此界面用于添加已有的设计源文件，本实验无需添加，直接点击Next进入下一步。



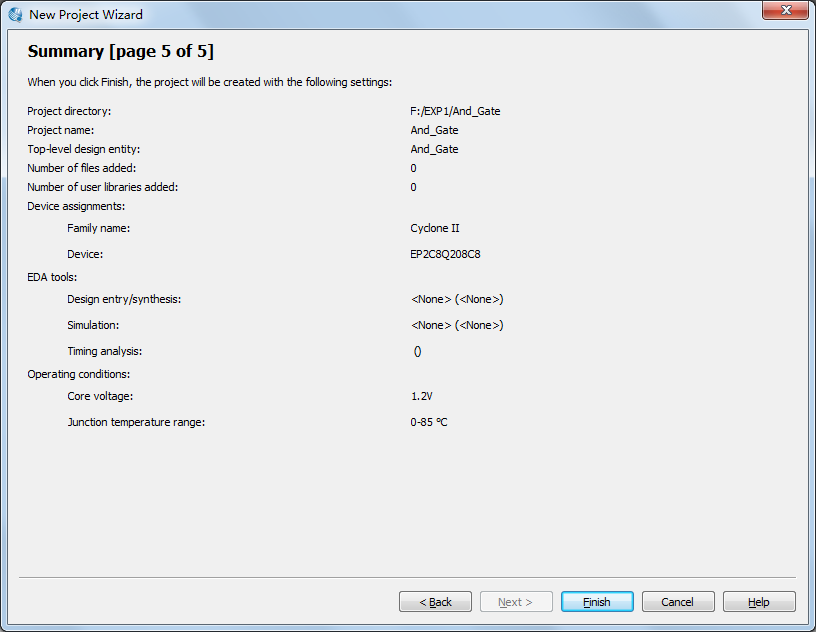
Step3. 此界面用于元器件的设定，元器件的选择与所用开发板硬件相关，我们实验用的开发板中的器件为**cycloneII系列，型号为EP2C8Q208C8**，选择相应器件，点击Next进入下一步



Step4. 此界面用于选择第三方工具软件，本实验无需选择，直接点击Next进入下一步

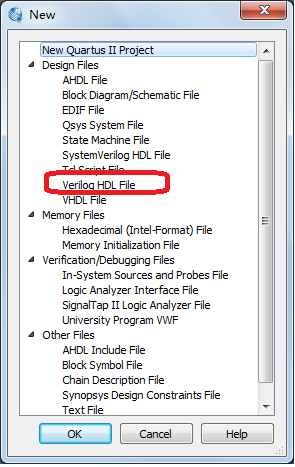


Step5.工程项目信息综述，无需任何操作，直接点击Finish完成工程项目的建立。



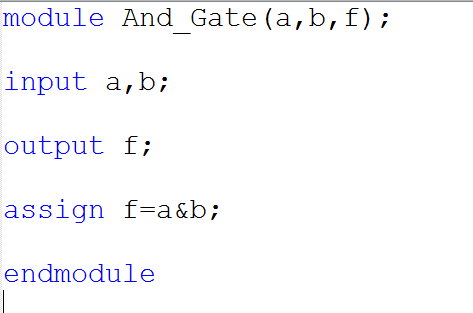
(3) 新建Verilog源文件

QuartusII提供完整的文本编辑工具，在主菜单File中选择New选项，新建verilog源文件；



在新建文件类型对话框中选择Verilog HDL文件类型，然后点击OK按钮。

(4) Verilog HDL程序编写

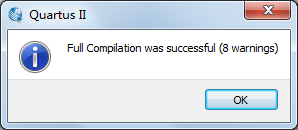


**注意**

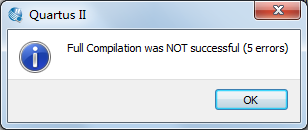
1. **保存文件时文件名需要与新建工程项目时设定的顶层实体名一致。**
2. **Verilog 语言大小写敏感，输入程序时注意字母的大小写。**

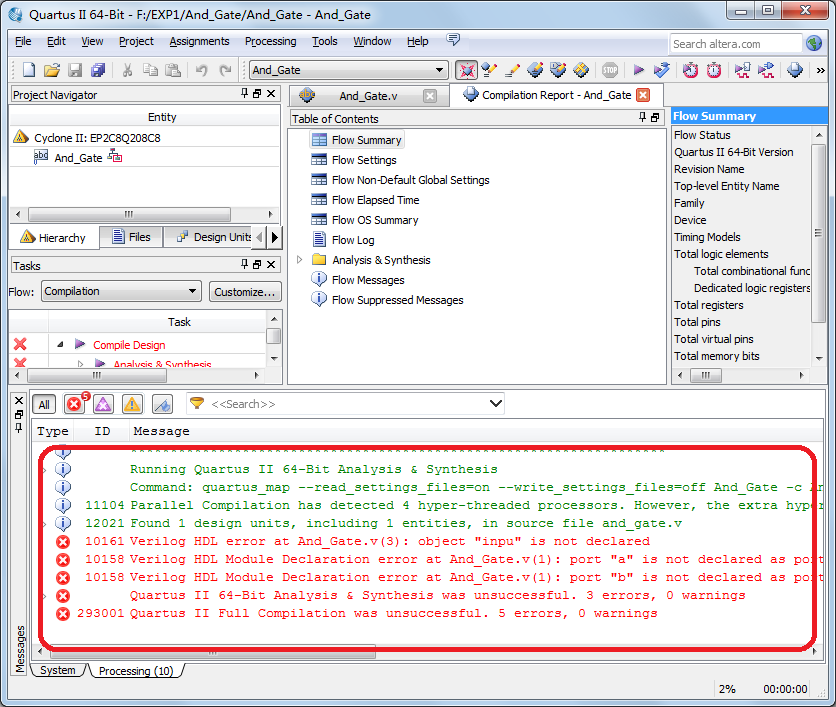
（5）对源程序进行语法检查和综合

在主菜单Processing中选择Start Compilation选项，对源文件进行语法检查和综合。

如果语法检查和综合成功，将显示如下提示窗口。 

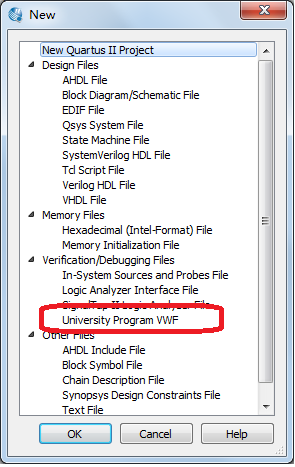
如果语法检查和综合发现错误，将显示如下提示窗口，提示有错误。

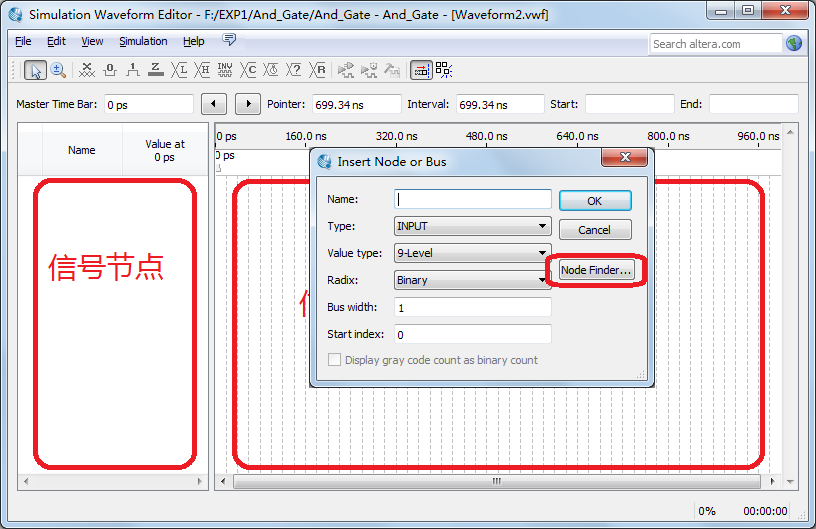


请根据如下消息窗口里的错误提示信息进行程序修改，直到语法检查和综合成功为止。

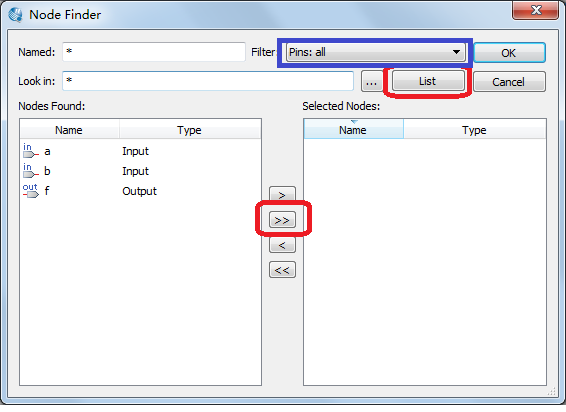
（6）仿真

Step1.在主菜单File中选择New选项，新建仿真文件，在新建文件类型窗口选择University Program VWF文件类型；然后点击OK按钮；



Step2.在下图，左边信号节点区域里，双击鼠标右键；弹出节点工具对话框，选择Node Finder按钮；

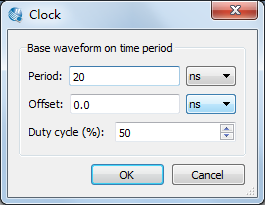
Step3.在如下界面中，首先Filter选择栏中选择，Pins：all选项，然后点击List按钮，获取信号节点列表，然后点击“〉〉”按钮选择需要仿真的信号节点，最后点击OK按钮。

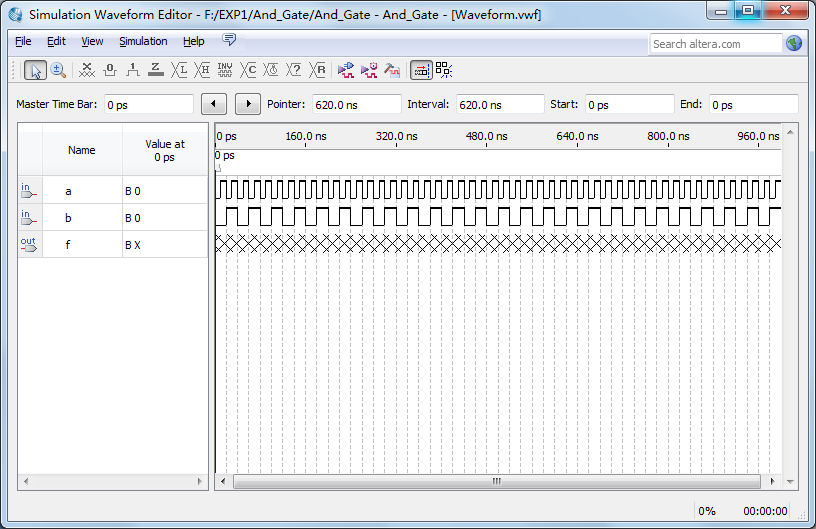


Step4.编辑波形

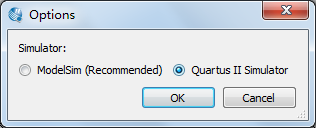
在如下界面里，鼠标在左侧信号节点窗口，点击需要编辑的节点，然后利用信号编辑工具，选择需要赋予的信号；此处我们选择信号为方波时钟信号。为a,b两个信号节点，分别设定20ns和40ns的时钟信号，然后将本文件保存。



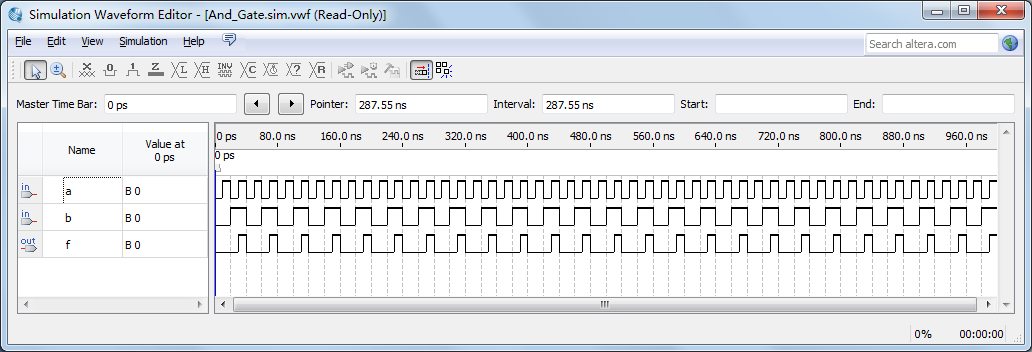


****

Step5.进行仿真，选择Simulation菜单下的option选项，在弹出的选择窗口中选择，右侧的仿真工具;

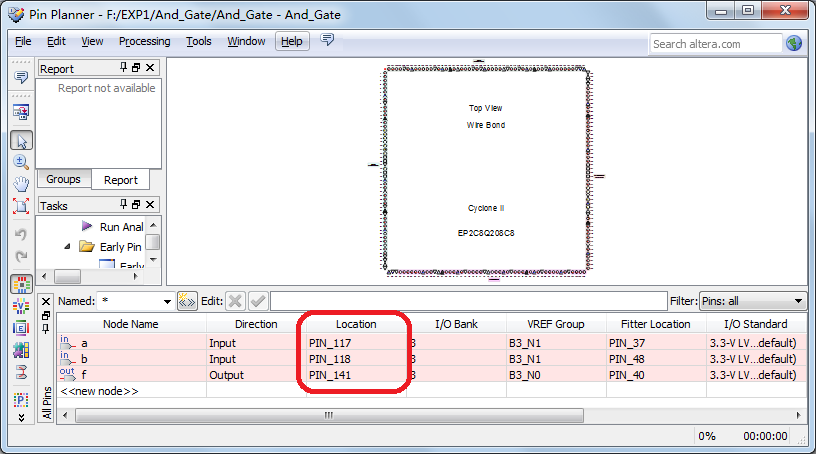


然选择Simulation菜单下的Run Functional Simulation选项,quartus将自动生成如下仿真结果；



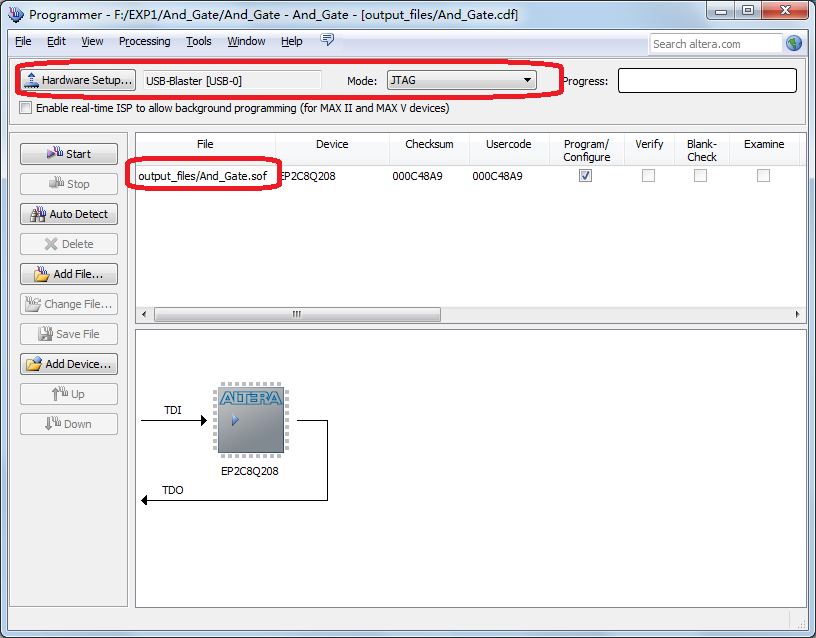
（7）引脚分配

关闭仿真界面，回到Quartus编辑设计界面；选择主菜单Assigments下的Pin Planner选项；在引脚分配界面里，参照指导书提供的引脚信息，为输入、输出分配对应引脚；然后关闭引脚分配界面,在主菜单Processing中选择Start Compilation选项，对源文件再次进行综合，以便将引脚信息综合到下载文件中。



（8）下载

关闭仿真界面，连接下载器USB Blaster一端连接到计算机，另一端连接到开发板的JTAG插口上；选择主菜单Tools下的Progrmmer选项；参照下图界面，注意显示出红框内USB-Blaster信息，并将设定下载类型Mode：JTAG；选择需要下载的目标.SOF文件，然后点击Start 按钮；进行下载；下载完毕将出现下载成功信息。



**2. 异或、与非、或非等基本逻辑门电路实现**

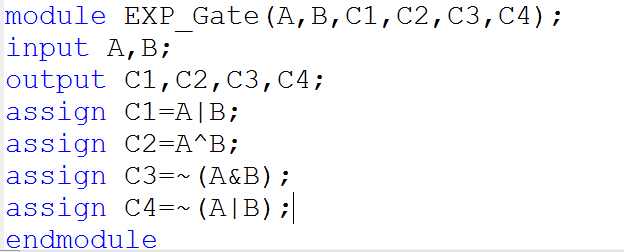
真值表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 输入信号 | | 输出信号 | | | |
| A | B | C1 | C2 | C3 | C4 |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

根据真值表，按照以下步骤和要求，实现或门、异或、与非、或非四种基本逻辑门实验

1. 利用工程向导，新建工程项目；
2. 新建一个Verilog HDL文件；
3. 在Verilog HDL 文件，根据真值表输入程序；
4. 进行语法检查和综合；综合成功后进入下一步，否则修改程序，直至综合成功；
5. 进行功能仿真，保存仿真结果；
6. 分配引脚，生成\*.sof下载文件；
7. 下载验证。

参考程序：



### 实验结果

1.两输入与门电路

引脚分配表

|  |  |  |
| --- | --- | --- |
| 程序中信号名称 | 实际引脚号 | 说明 |
| a |  | 拨动开关1 |
| b |  | 拨动开关2 |
| c |  | LED1 |

实验结果

|  |  |  |  |
| --- | --- | --- | --- |
| 拨动开关1  a | 拨动开关2  b | LED1  c | |
| 状态  （亮/灭） | 逻辑值 |
| 0 | 0 |  |  |
| 0 | 1 |  |  |
| 1 | 0 |  |  |
| 1 | 1 |  |  |

注：LED亮对应逻辑0；LED灭对应逻辑1

2.其他基本逻辑门电路

引脚分配表

|  |  |  |
| --- | --- | --- |
| 程序中信号名称 | 实际引脚号 | 说明 |
| A |  | 拨动开关1 |
| B |  | 拨动开关2 |
| C1 |  | LED1 |
| C2 |  | LED2 |
| C3 |  | LED3 |
| C4 |  | LED4 |

实验结果（结果同时填写逻辑值和亮灭状态）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 拨动开关1  A | 拨动开关2  B | LED1  C1 | LED2  C2 | LED3  C3 | LED4  C4 |
| 0 | 0 |  |  |  |  |
| 0 | 1 |  |  |  |  |
| 1 | 0 |  |  |  |  |
| 1 | 1 |  |  |  |  |

注：1. LED亮对应逻辑0；LED灭对应逻辑1

2. 拨动开关 ON对应逻辑0；OFF对应逻辑1

## 实验二 数字比较器和3线-8译码器

### 实验目的

1. 学习Verilog HDL语言基本语法。
2. 掌握module的基本结构和表述方法。

#### 掌握信号的类型和端口定义的方法和always 、if语句的使用。

### 实验内容

#### 3-8译码器设计。

#### 两位数字比较器设计。

### 实验原理和步骤

1. **3-8译码器设计**

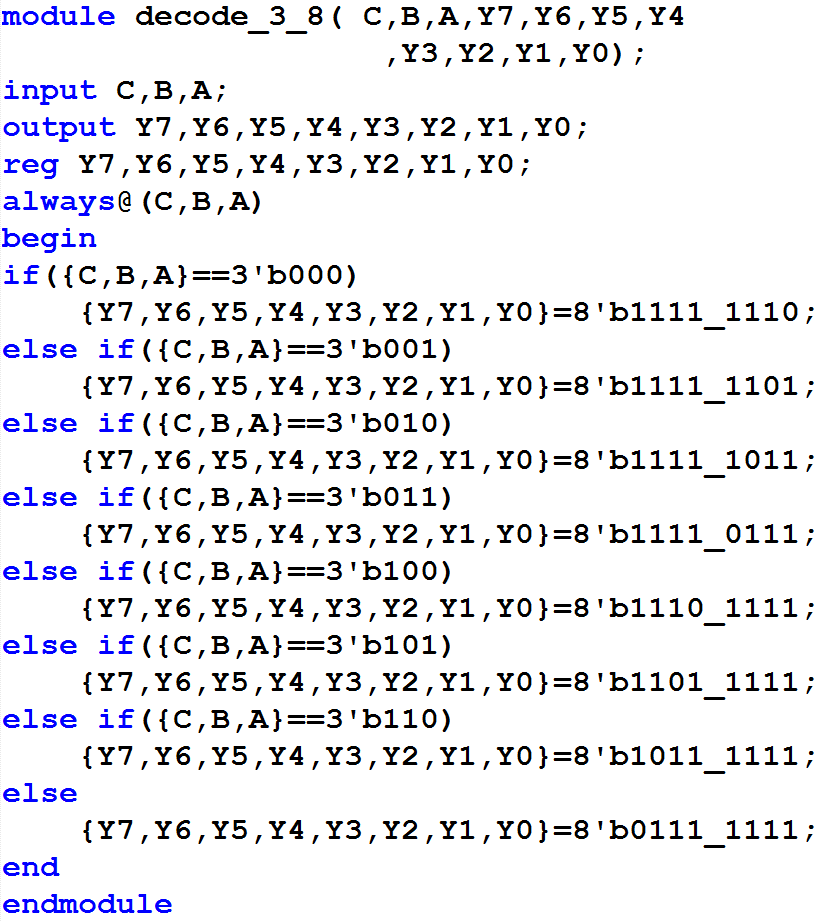
二进制译码器的输入是一组二进制代码,输出是一组与输入代码一一对应的高、低电平信号.三位二进制共有8种状态,所以对应的输出有8种状态.而每个状态的具体信息可由读者自行设置。本实验以3个拨码开关作为数据输入,用8个发光二极管表示译码后的信息。

3-8译码器真值表

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 输入 | | | 输出 | | | | | | | |
| C | B | A | Y7 | Y6 | Y5 | Y4 | Y3 | Y2 | Y1 | Y0 |
| **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** |
| **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** |
| **0** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** |
| **0** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** |
| **1** | **0** | **0** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** |
| **1** | **0** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** |
| **1** | **1** | **0** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** |
| **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |

根据真值表可知，该译码器为低电平输出有效；

参考程序：

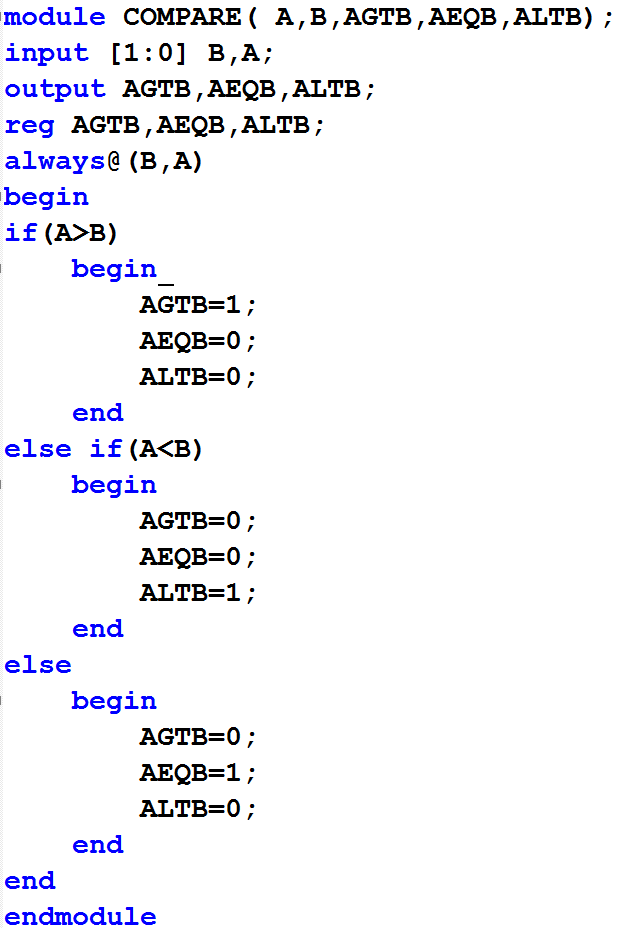
具体实验步骤如下：

1. 利用工程向导，新建工程项目，建议工程名为exp2\_1,顶层文件名为decode\_3\_8；
2. 新建一个Verilog HDL文件；
3. 在Verilog HDL 文件，根据真值表输入程序；
4. 进行语法检查和综合；综合成功后进入下一步，否则修改程序，直至综合成功；
5. 进行功能仿真,保存仿真结果；
6. 分配引脚，生成\*.sof下载文件；
7. 下载验证。
8. **数字比较器设计；**

在[数字电路](https://baike.baidu.com/item/%E6%95%B0%E5%AD%97%E7%94%B5%E8%B7%AF/1705)中，经常需要对两个位数相同的[二进制数](https://baike.baidu.com/item/%E4%BA%8C%E8%BF%9B%E5%88%B6%E6%95%B0/108101)进行比较，以判断它们的相对大小或者是否相等，用来实现这一功能的逻辑电路就成为数值比较器。数值比较器就是对两数A、B进行比较，以判断其大小的逻辑电路。比较结果有A>B、A<B以及A=B三种情况。

|  |  |
| --- | --- |
|  | 引脚说明：  A,B：数据输入信号，均为两位二进制宽度  AGTB：当A>B，其值为1，否则为0  AEQB：当A>B，其值为1，否则为0  ALTB：当A>B，其值为1，否则为0 |

参考程序：



参考实验步骤如下：

1. 利用工程向导，新建工程项目，建议工程名为exp2\_2,顶层文件名为COMPARE；
2. 新建一个Vrilog HDL文件；
3. 在Verilog HDL 文件，根据真值表输入程序；
4. 进行语法检查和综合；综合成功后进入下一步，否则修改程序，直至综合成功；
5. 进行功能仿真,保存仿真结果；
6. 分配引脚，生成\*.sof下载文件；
7. 下载验证。

### 实验结果

1. **3线-8线译码器（if语句）**

实验代码：

仿真结果：

引脚分配：

|  |  |  |
| --- | --- | --- |
| 程序中信号名称 | 实际引脚号 | 说明 |
| A |  | 拨动开关K1 |
| B |  | 拨动开关K2 |
| C |  | 拨动开关K3 |
| Y0 |  | LED1 |
| Y1 |  | LED2 |
| Y2 |  | LED3 |
| Y3 |  | LED4 |
| Y4 |  | LED5 |
| Y5 |  | LED6 |
| Y6 |  | LED7 |
| Y7 |  | LED8 |

实验验证结果

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 输入 | | | 输出 | | | | | | | |
| K3 | K2 | K1 | LED8 | LED7 | LED6 | LED5 | LED4 | LED3 | LED2 | LED1 |
| C | B | A | Y7 | Y6 | Y5 | Y4 | Y3 | Y2 | Y1 | Y0 |
| **0** | **0** | **0** |  |  |  |  |  |  |  |  |
| **0** | **0** | **1** |  |  |  |  |  |  |  |  |
| **0** | **1** | **0** |  |  |  |  |  |  |  |  |
| **0** | **1** | **1** |  |  |  |  |  |  |  |  |
| **1** | **0** | **0** |  |  |  |  |  |  |  |  |
| **1** | **0** | **1** |  |  |  |  |  |  |  |  |
| **1** | **1** | **0** |  |  |  |  |  |  |  |  |
| **1** | **1** | **1** |  |  |  |  |  |  |  |  |

1. 数据比较器（if语句）

实验代码：

仿真结果：

引脚分配：

|  |  |  |
| --- | --- | --- |
| 程序中信号名称 | 实际引脚号 | 说明 |
| A0 |  | 拨动开关K1 |
| A1 |  | 拨动开关K2 |
| B0 |  | 拨动开关K3 |
| B1 |  | 拨动开关K4 |
| AGTB |  | LED1 |
| AEQB |  | LED2 |
| ALTB |  | LED3 |

实验验证结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 输入 | | | | 输出 | | |
| B1 | B0 | A1 | A0 | LED3 | LED2 | LED1 |
| **0** | **0** | **0** | **0** |  |  |  |
| **0** | **0** | **0** | **1** |  |  |  |
| **0** | **0** | **1** | **0** |  |  |  |
| **0** | **0** | **1** | **1** |  |  |  |
| **0** | **1** | **0** | **0** |  |  |  |
| **0** | **1** | **0** | **1** |  |  |  |
| **0** | **1** | **1** | **0** |  |  |  |
| **0** | **1** | **1** | **1** |  |  |  |
| **1** | **0** | **0** | **0** |  |  |  |
| **1** | **0** | **0** | **1** |  |  |  |
| **1** | **0** | **1** | **0** |  |  |  |
| **1** | **0** | **1** | **1** |  |  |  |
| **1** | **1** | **0** | **0** |  |  |  |
| **1** | **1** | **0** | **1** |  |  |  |
| **1** | **1** | **1** | **0** |  |  |  |
| **1** | **1** | **1** | **1** |  |  |  |

## 实验三 数字选择器和七段码译码器

### 实验目的

1. 掌握always进程语句是使用方法；
2. 掌握wire和reg信号类型的区别；
3. 掌握case语句的使用；

### 实验内容

#### 四位四选一数据选择器设计；

#### BCD-七段码译码器设计；

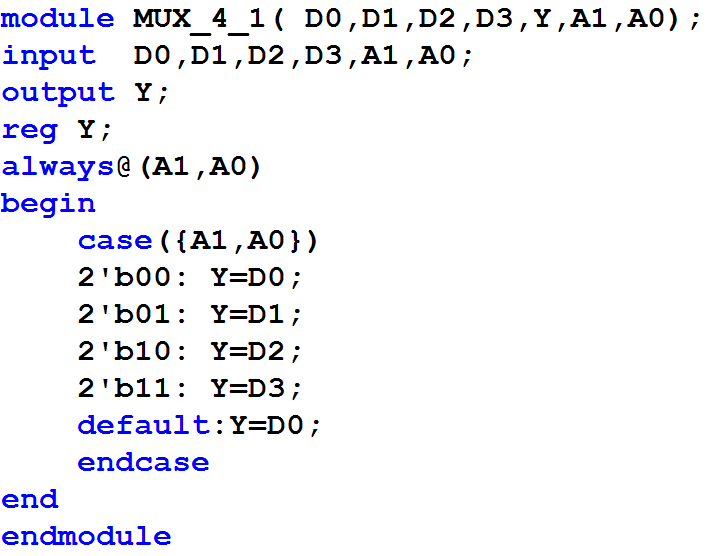
### 实验原理及步骤

1. **数据选择器设计**

数据选择器是指经过选择，把多个通道的数据传送到唯一的公共数据通道上去，实现数据选择功能的逻辑电路称为数据选择器。在多路[数据传送](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E4%BC%A0%E9%80%81)过程中，能够根据需要将其中任意一路选出来的电路，叫做数据选择器，也称[多路选择器](https://baike.baidu.com/item/%E5%A4%9A%E8%B7%AF%E9%80%89%E6%8B%A9%E5%99%A8)[开关](https://baike.baidu.com/item/%E5%A4%9A%E8%B7%AF%E5%BC%80%E5%85%B3)。工作原理是：给A1A0一组信号 比如1 0 那么就相当于给了他一个2进制数字2 也就相当于选通了D2这个输入端，这个时候 输出Y 输出的就是D2的信号；D2是什么，Y就输出什么.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| https://gss0.bdstatic.com/94o3dSag_xI4khGkpoWK1HF6hhy/baike/w%3D268/sign=ab438b029313b07ebdbd570e34d69113/d4628535e5dde711c05e4fd5a7efce1b9d166100.jpg | |  |  |  | | --- | --- | --- | | **选择控制端** | | **输出** | | **A1** | **A0** | **Y** | | **0** | **0** | **D0** | | **0** | **1** | **D1** | | **1** | **0** | **D2** | | **1** | **1** | **D3** | |

**参考程序（一位四选一数据选择器）：**

****

具体实验步骤如下：（注意：要求设计的是**四位四选一数据选择器**，即四个数据源都数据宽度均为四位，选择一路输出，数据宽度也是四位）

1. 利用工程向导，新建工程项目，建议工程名为exp3\_1顶层文件名为Mux\_4\_1；
2. 新建一个Verilog HDL文件；
3. 在Verilog HDL 文件，根据真值表输入程序；
4. 进行语法检查和综合；综合成功后进入下一步，否则修改程序，直至综合成功；
5. 进行功能仿真,保存仿真结果；
6. 分配引脚，生成\*.sof下载文件；
7. 下载验证
8. **BCD-7段码译码器设计**

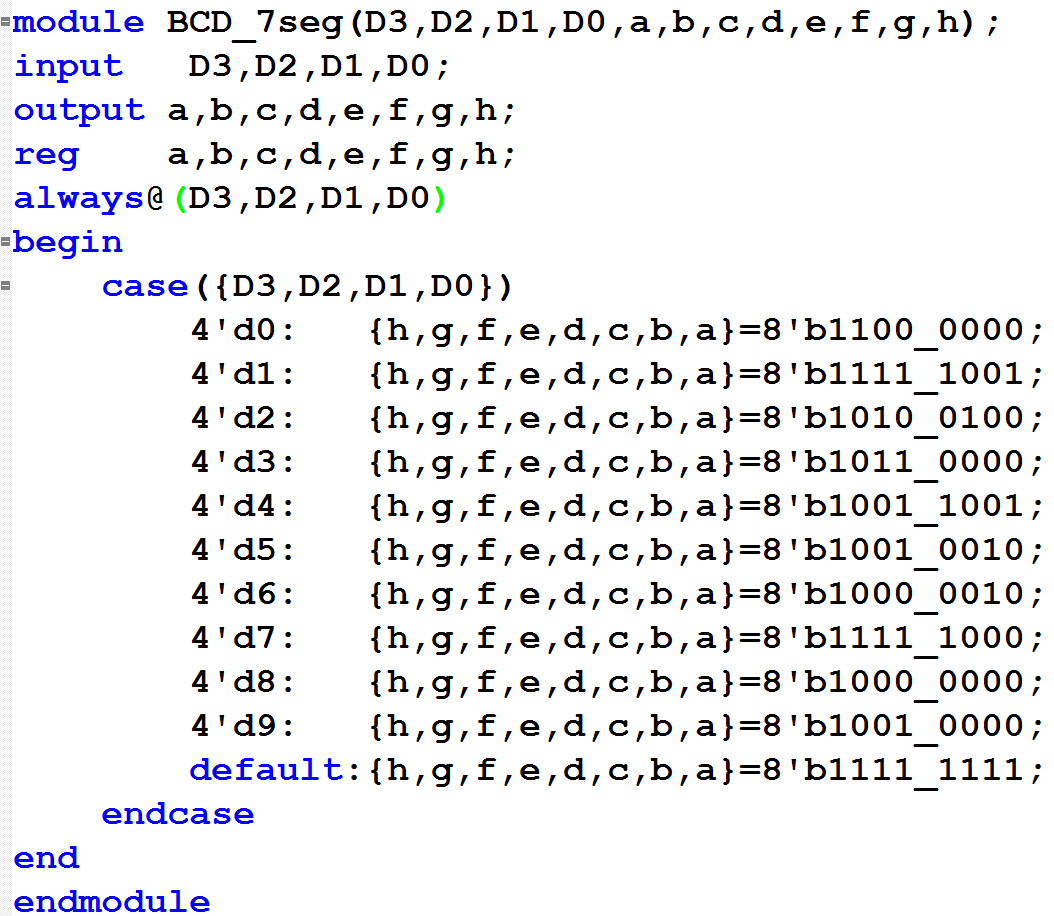
数码管分共阴和共阳连接，分别对应高有效和低有效。这里数码管采用共阳连接。数码管有七个LED段组成。可以分别对每一段进行置数以显示不同的数。

当数码管的位选信号选通后,给任意的数码管八段(小数点也算上对应H)输入低电平时,将会点亮相对应的段显示器。

|  |  |  |
| --- | --- | --- |
|  | FPGA引脚 | LED字段 |
| 170 | a |
| 165 | b |
| 169 | c |
| 175 | d |
| 173 | e |
| 168 | f |
| 164 | g |
| 171 | h |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 七段码真值表（共阳） | | | | | | | | | | | |  |
| **数字** | BCD | | | | 七段码 | | | | | | | |
| D3 | D2 | D1 | D0 | h | g | f | e | d | c | b | a |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 6 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 8 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

参考程序：

****

具体实验步骤如下：

1. 利用工程向导，新建工程项目，建议工程名为exp3\_2顶层文件名为BCD\_7seg；
2. 新建一个Verilog HDL文件；
3. 在Verilog HDL 文件，根据真值表输入程序；
4. 进行语法检查和综合；综合成功后进入下一步，否则修改程序，直至综合成功；
5. 进行功能仿真,保存仿真结果；
6. 分配引脚，生成\*.sof下载文件；
7. 下载验证
8. 实验结果
9. 四位四选一数据选择器

实验代码：

仿真结果：

1. BCD-七段码译码器

实验代码：

仿真结果：

引脚分配：

|  |  |  |
| --- | --- | --- |
| 程序中信号名称 | 实际引脚号 | 说明 |
| D0 |  | 拨动开关K1 |
| D1 |  | 拨动开关K2 |
| D2 |  | 拨动开关K3 |
| D3 |  | 拨动开关K4 |
| a |  |  |
| b |  |  |
| c |  |  |
| d |  |  |
| e |  |  |
| f |  |  |
| g |  |  |
| h |  |  |

实验验证结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 输入 | | | | 输出 |
| D3 | D2 | D1 | D0 | 显示字符 |
| **0** | **0** | **0** | **0** |  |
| **0** | **0** | **0** | **1** |  |
| **0** | **0** | **1** | **0** |  |
| **0** | **0** | **1** | **1** |  |
| **0** | **1** | **0** | **0** |  |
| **0** | **1** | **0** | **1** |  |
| **0** | **1** | **1** | **0** |  |
| **0** | **1** | **1** | **1** |  |
| **1** | **0** | **0** | **0** |  |
| **1** | **0** | **0** | **1** |  |

## 实验四 触发器和基本计数器

### 实验目的

1. 理解触发器和计数器的基本功能和设计原理。
2. 具备基于行为描述的方法设计计数器的能力。

### 实验内容

1. D触发器设计；
2. 计数器设计；

### 实验原理及步骤

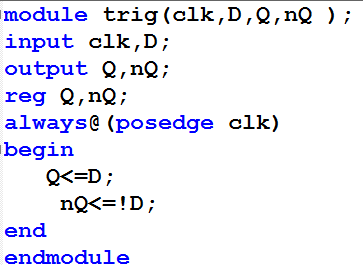
**1. D触发器的设计**

在各种复杂的数字电路屮，不但需要对输入信号进行逻辑运算和算术运算，还经常需要将这些结果保存下来。因此需要具有记忆功能的基本逻辑单元，存储数字信息的基本单元电路就被称为触发器。根据电路结构和制位方式，可将触发器分为D触发器、JK触发器、T触发器等，其中D触发器最为常用。在数字电路中，一个基本的上升沿触发的D触发器符号和功能表如下所示

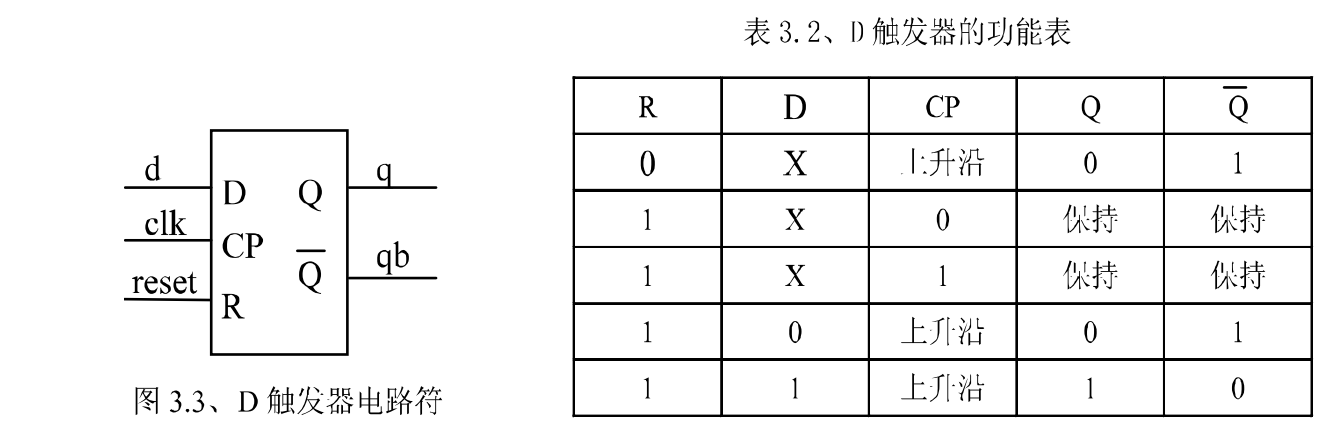
1. 基本D触发器

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| D触发器电路符号 | D触发器功能表   |  |  |  |  | | --- | --- | --- | --- | | D | CP | Q | /Q | | X | 0 | 保持 | 保持 | | X | 1 | 保持 | 保持 | | 0 | 上升沿 | 0 | 1 | | 1 | 上升沿 | 1 | 0 | |

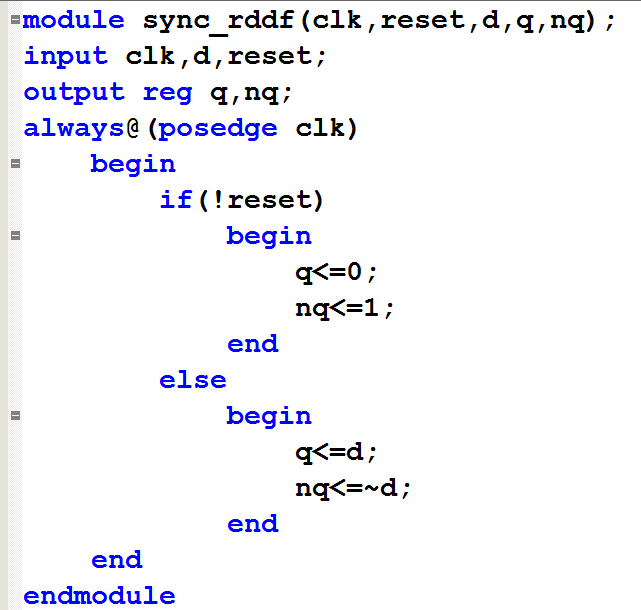
参考程序：



1. 同步复位的D触发器

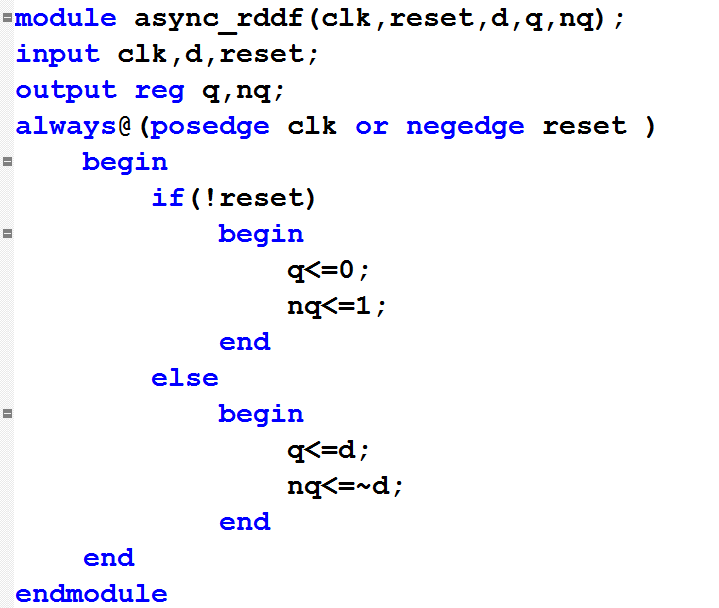


参考程序：



1. 异步复位的D触发器

参考程序



参考实验步骤：

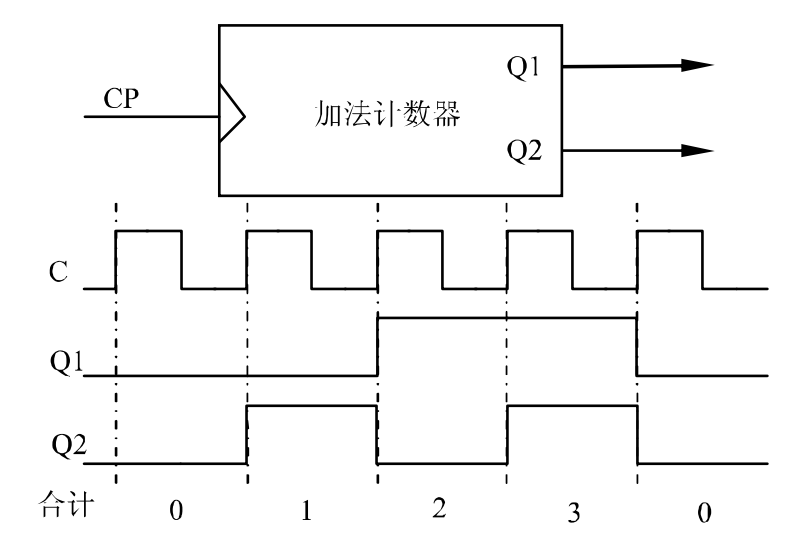
1. 利用工程向导，新建工程项目，建议工程名为exp4\_1顶层文件名为D\_trig；
2. 新建一个Verilog HDL文件；
3. 在Verilog HDL 文件，根据真值表输入程序；
4. 进行语法检查和综合；综合成功后进入下一步，否则修改程序，直至综合成功；
5. 进行功能仿真,保存仿真结果；
6. 分配引脚，生成\*.sof下载文件；
7. 下载验证

参照上述步骤，完成根据老师要求从基本触发器，异步复位触发器，同步复位触发器中选择一个完成。

**2.计数器的设计**

（1）加法计数器

加法计数器，每次在时钟信号clk的上升沿时刻，计数器会将计数值加1，对于一个2bit的加法计数器，其计数值依次是 0,1,2,3,0,1,2…..,周而复始，



通过研究计数器我们可以得到的一下结论

① 对于n bit计数器，其基数范围是0 到 ；

② Q0和Q1输出信号的频率分别为时钟脉冲的1/2和1/4，可见可以利用计数器实现时钟信号进行分频。

参考程序：

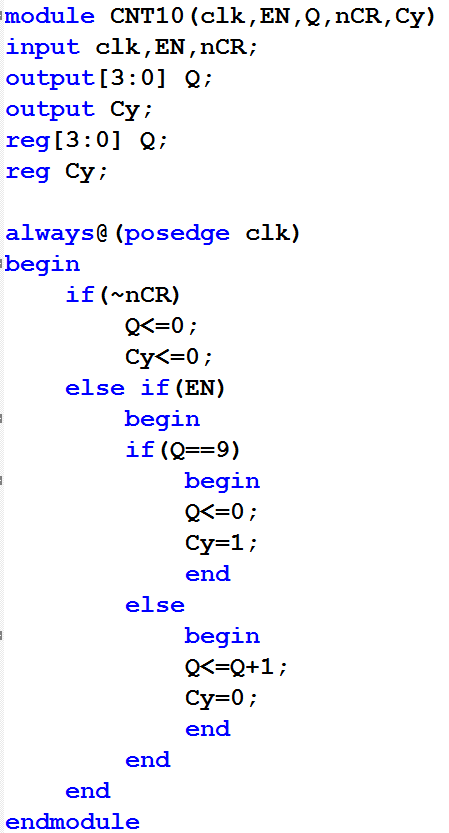
程序中clk：时钟信号；

EN：计数使能信号；

Q：计数值信号；

nCR：复位信号；

Cy：计数进位信号



实验步骤：

1. 利用工程向导，新建工程项目，建议工程名为exp4\_2顶层文件名为Count\_4bit；
2. 新建一个Verilog HDL文件；
3. 在Verilog HDL 文件，编写十进制加计数器，并结合前面实验完成的BCD-七段码译码器；以便可以将计数值在数码管上显示。
4. 进行语法检查和综合；综合成功后进入下一步，否则修改程序，直至综合成功；
5. 进行功能仿真,保存仿真结果；
6. 分配引脚，生成\*.sof下载文件；
7. 下载验证

### 实验结果

1. D触发器设计

实验代码：

仿真结果：

2. 计数器设计（4位二进制）

实验代码：

仿真结果：

## 实验五 LED动态扫描显示驱动电路

### 实验目的

1. 掌握层次化设计的基本设计方法。

2. 掌握LED动态扫描显示电路的基本原理。

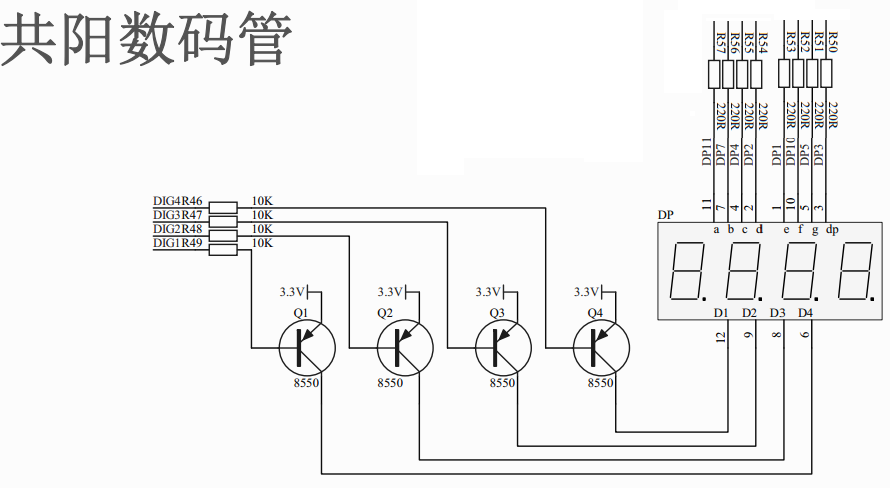
### 实验内容

#### 完成四位LED动态扫描显示电路设计；

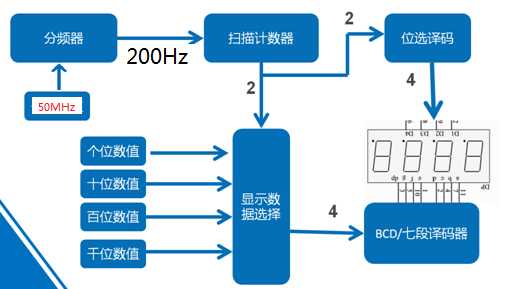
### 实验原理及步骤

由于LED静态显示需要占用较多的I/O口，且功耗较大，因此在大多数场合通常不采用静态显示，而采用动态扫描的方法来控制LED数码管的显示。动态显示的特点是将所有位数码管的段选线并联在一起，由位选线控制是哪一位数码管有效。点亮数码管采用动态扫描显示，即各数码管循环轮流显示，当循环显示频率较高时，利用人眼的暂留特性，看不出闪烁现象，这种显示需要一个接口完成字形码的输出（段码），另一接口完成各数码管的点亮（位选） LED数码管动态扫描显示即各数码管循环轮流显示，当循环显示频率较高时，利用人眼的暂留特性，看不出闪烁现象，这种显示需要一个接口完成字形码的输出（段码），另一接口完成各数码管的点亮（位选）。动态扫描显示时刷新频率最好大于50HZ,即每显示一轮的时间不超过20ms，每个数码管显示时间不能太长也不能太短，时间太长会影响刷新率，导致总体显示呈现闪烁的现象，时间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以上。

LED动态扫描显示外部电路原理图如下所示，四个数码管共用8位数据线用于段码输入，控制数码管显示的信息，四位位码控制任一时刻只有一个数码管被点亮。

****

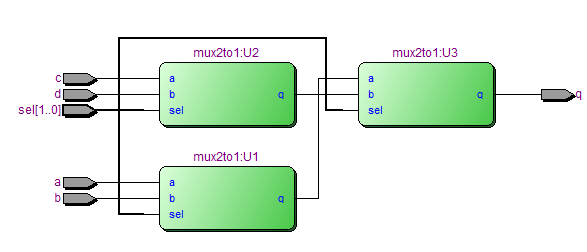
LED数码管外部电路原理图

****

LED数码管显示驱动电路结构图

LED扫描显示电路主要有分频器、扫描计数器、2线-四线译码器、四选一数据选择器和BCD-七段码译码器构成。分频器用于将外部50MHz时钟信号，分频产生200Hz的扫描时钟，因为有四个数码管所示数码管的刷新频率为200Hz/4=50Hz。2线-4线译码器产生数码管的位选信号，由于控制驱动PNP型三极管所以需要输出低电平为有效电平值，四选一数据选择器用于对特定的LED数码管从四个需显示的数据中选择出与之对应的显示数据。2线-4线译码器和四选一数据选择器输入控制信号都为两位数字信号而且需要同步，所以就需要产生两位数字信号，周而复始的产生00、01、10、11，显然这个信号可以用一个四进制的加一计数器产生。对于上述的五部分电路，在前面的实验已经设计过相似电路，请参考前面的实验设计适用于本实验的电路。

本实验将上面的基本单元电路，利用层次化设计的方法，构建上层电路将这五部分互联起来，实现完整的LED扫描显示驱动电路。

层次化设计范例：利用二选一数据选择器构成四选一数据选择器。如上图，四选一数据选择器可有三个二选一数据选择器构成，形成以二选一数据选择器为底层元件，在顶层将其互联起来构成四选一数据选择器。

参考程序：

**底层模块：二选一数据选择器：**

module mux21a (a, b ,sel，out );

intput a,b,sel;//定义信号方向

output out; //

wire nsl,sela,selb; //内部变量 相当于内部导线（线网类型）

assign nsl=~ sel;

assign sela=a & nsl;

assign selb=b & sel;

assign out=sela | selb;

endmodule

**顶层模块：四选一数据选择器**

module mux4to1（a,b,c,d,sel,q）；

input a,b,c,d;

input [1:0] sel;

output q;

wire q\_int1,q\_int2;

mux2to1 U1(a,b,sel[0],q\_int1);

mux2to1 U2(c,d,sel[0],q\_int2);

mux2to1 U3(q\_int1,q\_int2,sel[0],q);

Endmodule

**模块调用端口映射方法**

* 1. **位置映射**

顶层模块模块调用语句

mux2to1 U1(a,b,sel[0],q\_int1);

子模块定义语句

module mux21a (a, b ,sel，out );

信号变量根据位置先后一一对应

* 1. **名称映射**

mux2to1 U1( .a(a),

.b(b),

.sel(sel[0]),

.out(q\_int1)

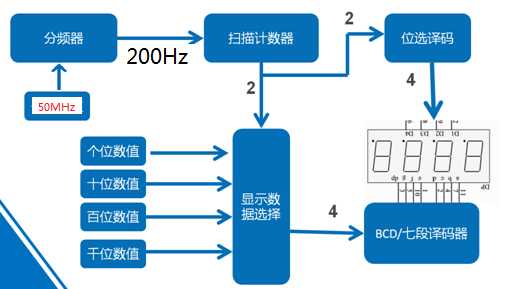
);

括号外.a、.b、.sel、.out为底层模块的端口名称

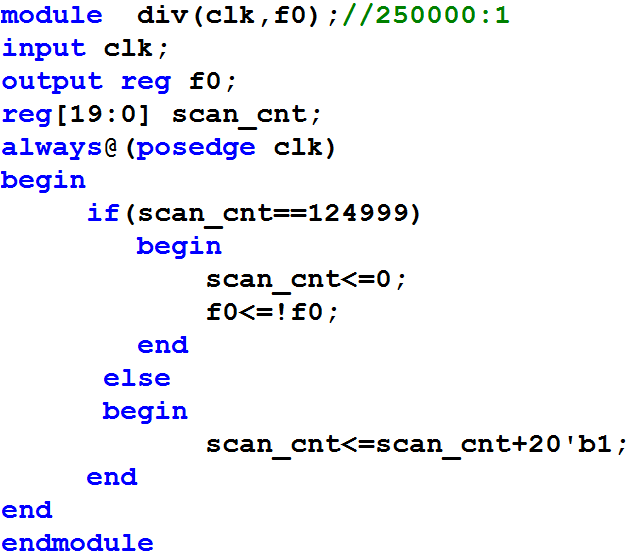
括号内（a）、（b）、（sel[0]）、（q\_int1）为顶层的送到底层的信号变量名称

注意端口映射时，所映射的信号必须是wire型

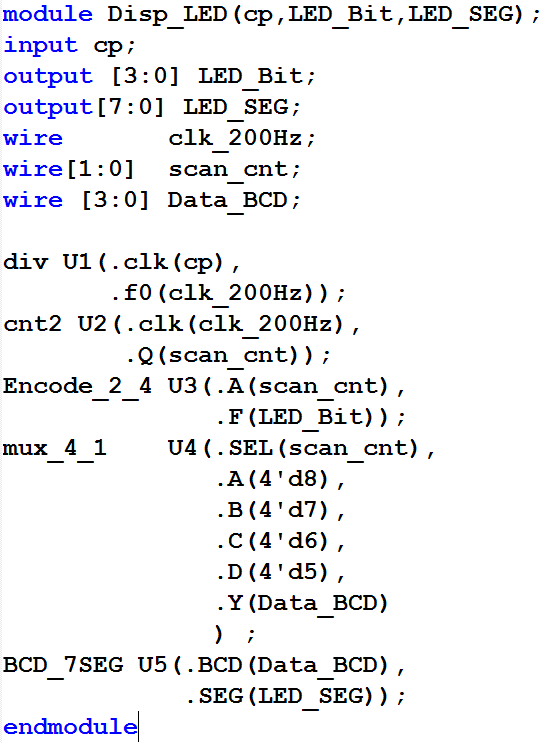
**扫描显示驱动电路顶层参考程序**



根据动态扫描结构图，结合先前所做实验，首先设计出2bit加法计数器、2线-4线译码器、四位四选一数据选择器和BCD-七段码译码器，结合下面给出的分频器和顶层参考设计程序，完成LED动态扫描显示电路的设计  
div.v



LED动态扫描顶层电路设计，显示固定值“1234”



参考实验步骤：

1. 利用工程向导，新建工程项目，建议工程名为exp5\_1顶层文件名为Disp\_LED；底层文件名可参考上面参考程序中的命名。
2. 新建一个Verilog HDL文件；
3. 在Verilog HDL 文件，根据真值表输入程序；
4. 进行语法检查和综合；综合成功后进入下一步，否则修改程序，直至综合成功；
5. 进行功能仿真,保存仿真结果；
6. 分配引脚，生成\*.sof下载文件；
7. 下载验证

### 实验结果

1. LED扫描显示驱动电路

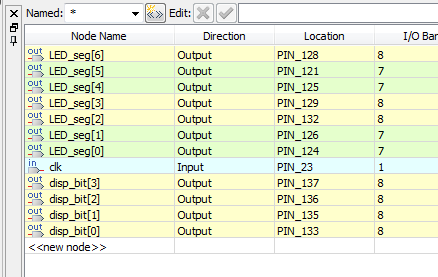
实验代码：（代码包括五个子模块代码和一个顶层模块代码）

仿真结果：

（完成扫描计数器、2线-四线译码器、四选一数据选择器和BCD-七段码译码器

各子模块的仿真图）

引脚分配表



LED\_seg[6]、LED\_seg[5]…LED\_seg[0],对应笔画g、f、e、d、c、b、a。Disp\_bit[0]最左边数码管。

4\*．下载实现验证的照片（可选）

## 

## 实验六 有限状态机

### 实验目的

1. 掌握有限状态机的设计方法。

#### 了解有限状态机的三个基本要素，掌握有限状态机的三段描述法。

### 实验内容

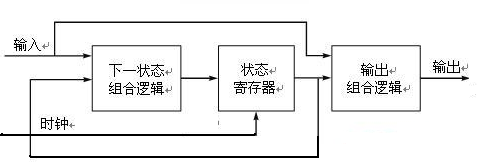
#### 利用有限状态机实现序列检测器；

### 实验原理及步骤

有限状态机是由寄存器组和组合逻辑构成的硬件时序电路，其状态（即由寄存器组的1和0的组合状态所构成的有限个状态）只可能在同一时钟跳变沿的情况下才能从一个状态转向另一个状态，究竟转向哪一状态还是留在原状态不但取决于各个输入值，还取决于当前所在状态。

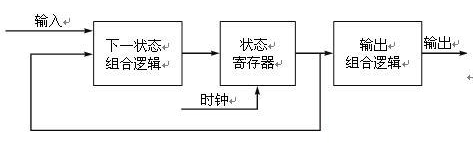
有限状态机有两种基本类型：米利（Mealy）机和摩尔（Moore）机：

①米利机的下一状态和输出取决于当前状态和当前输入；



米利型状态机框图

②摩尔机的下一状态取决于当前状态和当前输入，但其输出仅取决于当前状态。



摩尔型状态机框图

这两类有限状态机的下一状态和输出都是由组合逻辑电路形成的。

设计状态机的要点

第一，状态机要安全，是指FSM不会进入死循环，特别是不会进入非预知的状态，而且由于某些扰动进入非设计状态，也能很快的恢复到正常的状态循环中来。

第二，状态机的设计要满足设计的面积和速度的要求。

第三，状态机的设计要清晰易懂、易维护。

状态机描述方法

状态机描述的主要有三种方法：

① 一段式：

整个状态机写到一个always模块里面，在该模块中既描述状态转移，又描述状态的输入和输出。

② 二段式：

用两个always模块来描述状态机，其中一个always模块采用同步时序描述状态转移；另一个模块采用组合逻辑判断状态转移条件，描述状态转移规律以及输出。

③ 三段式：

在两个always模块描述方法基础上，增加第三个always模块。其中一个always模块采用同步时序描述状态转移，一个always采用组合逻辑判断状态转移条件，描述状态转移规律，另一个always模块描述状态输出。

在学习过程中，建议大家按照标准的三段式描述方式设计状态机

在此我们以序列检测器为例,研究如何使用Verilog语言用状态机模型设计一个能够检测输入序列中10010序列的序列检测器

首先根据检测10010序列的要求，绘制出状态转换图，从而确定状态机的状态个数。



序列检测器状态转换图

10010序列检测器程序

//=========================================================

// 定义模块，定义状态变量 ，定义符号状态值

//=========================================================

module seqdet(nrst,clk,x,z);  
 input nrst,clk;  
 input x;  
 output z;  
 reg z;  
 reg [4：0]CS,NS; // CS：初态 ,NS：次态

parameter [4：0] IDLE=5'b00000 , A=5'b00001 ,

B=5'b00010 ,C=5'b00100 ,

D=5'b01000 ,E=5'b10000 ;

//定义符号状态值

//=========================================================

/\*第一个always 描述状态机初始化和状态的翻转\*/

//=========================================================

always @(posedge clk or negedge nrst)  
 begin

if(!nrst)

CS <= IDLE;

else

CS <= NS; //现态到次态翻转

end

//=========================================================

/\*第二个always 描述状态机根据当前状态和输入情况确定次态\*/

//=========================================================

always @(nrst or CS or x)

begin

case(CS)  
 IDLE：

begin //初始空闲状态

if(x==1)

NS<=A;

else

NS<=IDLE;

end

A：

begin //A 状态

if(x==0)

NS<=B;

else

NS<=A;

end

B：

begin

if(x==0)

NS<=C;

else

NS<=A;

end

C：

begin  
 if(x==1)

NS<=D;

else

NS<=IDLE;

end

D：

begin

if(x==0)

NS<=E;

else

NS<=A;

end

E：

begin

if(x==0)

NS<=C;

else

NS<=A;

End

//----------------------------------------------------------------------------------

default：

NS<=IDLE;

endcase

end

//=======================================================

/\*第三个always 描述状态机输出\*/

//=========================================================

always @(posedge clk or negedge nrst)

begin

if (!nrst)

z<=1'b0;

else

begin

case(NS)  
 IDLE，A，B，C，D： z<=0;  
 E： z<=1;

default： z<=0;

endcase

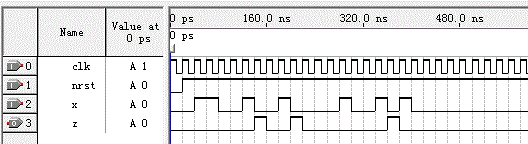
end

end

endmodule

//============================================================

仿真结果

****

状态序列检测其仿真图

参考实验步骤：

1. 利用工程向导，新建工程项目，建议工程名为exp6\_1顶层文件名为Seqdet；
2. 新建一个Verilog HDL文件，参照范例设计一个检测11011的序列检测器；
3. 在Verilog HDL 文件，根据真值表输入程序；
4. 进行语法检查和综合；综合成功后进入下一步，否则修改程序，直至综合成功；
5. 进行功能仿真,保存仿真结果；
6. 分配引脚，生成\*.sof下载文件；
7. \*下载验证

### 实验结果

1. 11011序列检测器

状态图：

实验代码：

仿真结果：

## 实验七 电子计时秒表

### 实验目的

1. 了解电子计时秒表的结构原理。
2. 掌握复杂时序电路的设计。
3. 熟悉层次化设计的方法。

### 实验内容

1. 完成电子计时秒表设计，秒表的结果显示形式为 000.00-999.99。
2. 具有开始\暂停和复位功能。

### 实验原理及步骤

1. 工作原理

秒表主要由显示时钟分频模块、计时模块、显示模块组成。

①时钟分频模块： 输入： 50MHz时钟信号。

输出：400Hz时钟信号，显示电路的驱动时钟；

100HZ时钟信号，秒表的计时标准信号。

②计时模块：输入：

100Hz时钟信号 ，秒表计时标准信号；

开始\暂停按钮 KEY1；

复位按钮 KEY2。

输出：

五位数码管第一位显示数据 BCD1；

五位数码管第二位显示数据 BCD2；

五位数码管第三位显示数据 BCD3；

五位数码管第四位显示数据 BCD4；

五位数码管第五位显示数据 BCD5；

③ 显示模块：输入： 400HZ时钟信号，显示电路的驱动时钟；

五位数码管第一位显示数据 BCD1；

五位数码管第二位显示数据 BCD2；

五位数码管第三位显示数据 BCD3；

五位数码管第四位显示数据 BCD4；

五位数码管第五位显示数据 BCD5；

输出：数码管显示位选通控制信号SG1，SG2，SG3，SG4,SG5

数码管段码驱动信号Seg

实验步骤：

1. 利用工程向导，新建工程项目，建议工程名为exp7\_1顶层文件名为Timer
2. 新建一个Verilog HDL文件，完成各模块和顶层电路的设计；
3. 进行语法检查和综合；综合成功后进入下一步，否则修改程序，直至综合成功；
4. 进行功能仿真,保存仿真结果；
5. 分配引脚，生成\*.sof下载文件；
6. \*下载验证

### 实验结果

实验代码：

仿真结果：

引脚分配表

\*下载验证（照片）

## 实验八 电子音乐盒

### 实验目的

* 1. 了解电子音乐盒设计原理；
  2. 掌握数控分频器的原理和设计方法

### 实验内容

1. 完成电子音乐盒电路的设计，所奏音乐自由选择，音乐播放不低于30秒。

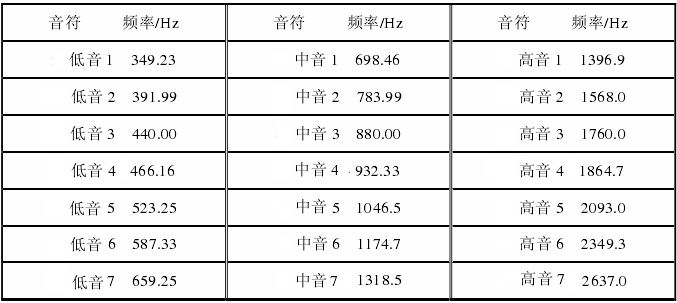
### 实验原理及步骤

1.实验背景知识

乐曲演奏的两个基本参数是每个音符的频率值（音调）及其持续的时间（音长）。因此只要控制输出到扬声器的激励信号的频率和持续时间，就可以发出连续的音乐声。

**音调的控制**： 频率的高低决定音调的高低。简谱中从低音1到高音1 的每个音名对应的频率如下表所示。

表8-1 音名与频率对照表



利用可控分频器对50MHz信号进行可控分频，产生不同音名需要的频率，由于50MHz信号频率较高，可以考虑用一个固定分频器将50MHz基频信号进行预分频，然后在送入可控分频器，由于可控分频器的输出信号占空比很小，需要在后面接入2分频器，将信号占空比调整成占空比为50%的信号驱动扬声器。

在程序中设置了一个状态机，每 250ms 改变一个状态（乐曲的每个一个节拍），组成乐曲的每个音符的频率值（音调）相对应于状态机的每一个状态。只要让状态机的状态按顺序转换，每个状态给数控分频器设定不同的分频比，就可以自动播放音乐了。

程序清单 **8.1 beep1.v**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// 《友谊地久天长》乐曲 //

// 时钟：50MHz //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//音高与频率的对应关系

//----------------------------------------------------------------------

//| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

//|低音 |261.6Hz |293.7Hz |329.6Hz |349.2Hz | 392Hz | 440Hz |493.9Hz |

//|中音 |523.3Hz |587.3Hz |659.3Hz |698.5Hz | 784Hz | 880Hz |987.8Hz |

//|高音 |1045.5Hz|1174.7Hz|1318.5Hz|1396.9Hz| 1568Hz | 1760Hz |1975.5Hz|

//----------------------------------------------------------------------

module song(clk,beep); //模块名称 song

input clk; //系统时钟 50MHz

output beep; //蜂鸣器输出端

reg beep\_r; //寄存器

reg[7:0] state; //乐谱状态机

reg[15:0]count,count\_end;

reg[23:0]count1;

//乐谱参数:D=F/2K (D:参数,F:时钟频率,K:音高频率)

parameter L\_5 = 16'd63776, //低音 5

L\_6 = 16'd56818, //低音 6

M\_1 = 16'd47774, //中音 1

M\_2 = 16'd42567, //中音 2

M\_3 = 16'd37919, //中音 3

M\_5 = 16'd31888, //中音 5

M\_6 = 16'd28409, //中音 6

H\_1 = 16'd23912; //高音 1

parameter TIME = 12500000; //控制每一个音的长短(250ms)

assign beep = beep\_r; //输出音乐

// 数控分频器，分频比由count\_end决定//

always@(posedge clk)

begin

count <= count + 1'b1; //计数器加 1

if(count == count\_end)

begin

count <= 16'h0; //计数器清零

beep\_r <= !beep\_r; //输出取反

end

end

always @(posedge clk)

begin

if(count1 < TIME) //一个节拍 250mS

count1 = count1 + 1'b1;

else

begin

count1 = 24'd0;

if(state == 8'd147)

state = 8'd0;

else

state = state + 1'b1;

case(state)

8'd0,8'd1: count\_end = L\_5;//低音"5",持续 2 个节拍

8'd2,8'd3,8'd4,8'd5,8'd6,8'd7,8'd8: count\_end = M\_1;//中音"1",持续 7 个节拍

8'd9,8'd10: count\_end = M\_3;//中音"3",持续 2 个节拍

8'd11,8'd12,8'd13,8'd14: count\_end = M\_2;

8'd15: count\_end = M\_1;

8'd16,8'd17: count\_end = M\_2;

8'd18,8'd19: count\_end = M\_3;

8'd20,8'd21,8'd22,8'd23,8'd24: count\_end = M\_1;

8'd25,8'd26: count\_end = M\_3;

8'd27,8'd28: count\_end = M\_5;

8'd29,8'd30,8'd31,8'd32,8'd33: count\_end = M\_6;

8'd34,8'd35,8'd36,8'd37,8'd38: count\_end = M\_6;

8'd39,8'd40,8'd41,8'd42: count\_end = M\_5;

8'd43,8'd44,8'd45: count\_end = M\_3;

8'd46,8'd47: count\_end = M\_1;

8'd48,8'd49,8'd50,8'd51: count\_end = M\_2;

8'd52: count\_end = M\_1;

8'd53,8'd54: count\_end = M\_2;

8'd55,8'd56: count\_end = M\_3;

8'd57,8'd58,8'd59,8'd60: count\_end = M\_1;

39理念教仪 EDA 实验指导书

8'd61,8'd62,8'd63: count\_end = L\_6;

8'd64,8'd65: count\_end = M\_5;

8'd66,8'd67,8'd68,8'd69: count\_end = M\_1;

8'd70,8'd71,8'd72,8'd73: count\_end = M\_1;

8'd74,8'd75: count\_end = M\_6;

8'd76,8'd77,8'd78,8'd79: count\_end = M\_5;

8'd80,8'd81,8'd82: count\_end = M\_3;

8'd83,8'd84: count\_end = M\_1;

8'd85,8'd86,8'd87,8'd88: count\_end = M\_2;

8'd89: count\_end = M\_1;

8'd90,8'd91: count\_end = M\_2;

8'd92,8'd93: count\_end = M\_6;

8'd94,8'd95,8'd96,8'd97: count\_end = M\_5;

8'd98,8'd99,8'd100: count\_end = M\_3;

8'd101,8'd102: count\_end = M\_5;

8'd103,8'd104,8'd105,8'd106: count\_end = M\_6;

8'd107,8'd108,8'd109,8'd110: count\_end = M\_6;

8'd111,8'd112: count\_end = H\_1;

8'd113,8'd114,8'd115,8'd116: count\_end = M\_5;

8'd117,8'd118,8'd119: count\_end = M\_3;

8'd120,8'd121: count\_end = M\_1;

8'd122,8'd123,8'd124,8'd125: count\_end = M\_2;

8'd126: count\_end = M\_1;

8'd127,8'd128: count\_end = M\_2;

8'd129,8'd130: count\_end = M\_3;

8'd131,8'd132,8'd133,8'd134: count\_end = M\_1;

8'd135,8'd136,8'd137: count\_end = L\_6;

8'd138,8'd139: count\_end = M\_5;

8'd140,8'd141,8'd142,8'd143: count\_end = M\_1;

8'd144,8'd145,8'd146,8'd147: count\_end = M\_1;

default:count\_end = 16'h0;

endcase

end

end

endmodule

1. 利用工程向导，新建工程项目，建议工程名为exp6\_1顶层文件名为Seqdet；
2. 新建一个Verilog HDL文件，参照范例设计一个检测11011的序列检测器；
3. 在Verilog HDL 文件，根据真值表输入程序；
4. 进行语法检查和综合；综合成功后进入下一步，否则修改程序，直至综合成功；
5. 进行功能仿真,保存仿真结果；
6. 分配引脚，生成\*.sof下载文件；
7. \*下载验证

### 实验结果

实验代码：

仿真结果(数控分频器)：

引脚分配表

## 实验九 PWM-脉冲宽度调制器

### 实验目的

#### 学习使用 PWM 进行 DAC 转换，了解 PWM-DAC 电压转换的原理。

#### 掌握PWM控制器的基本结构和设计方法。

### 实验内容

#### 采用分频器,计数器和比较器进行组合完成PWM模块的设计。

### 实验原理和步骤

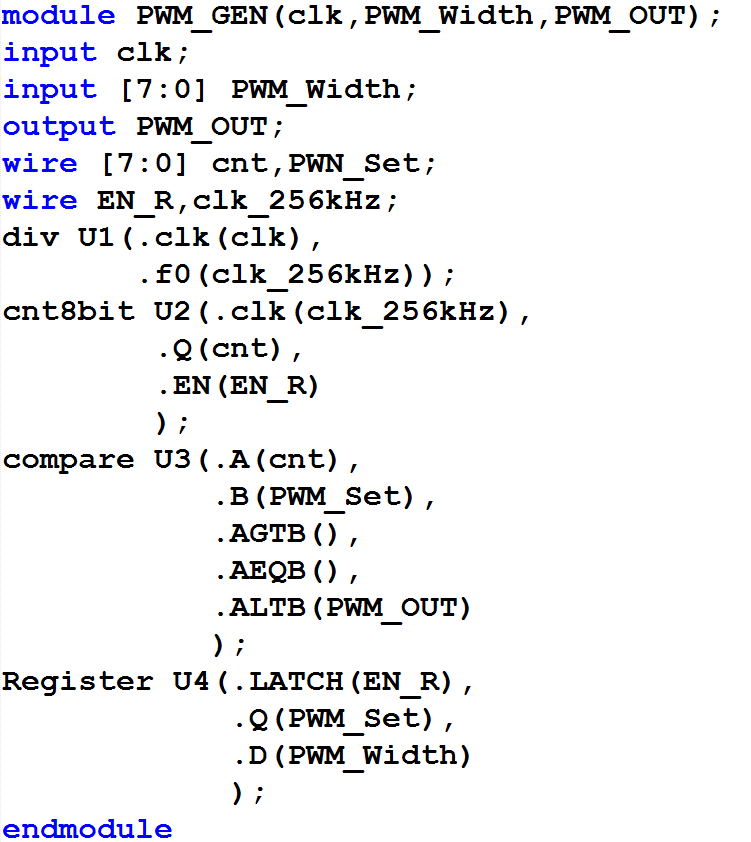
如图所示在时钟脉冲的作用下，循环计数器的5位输出逐次增大。5位数字调制信号用一个寄存器来控制，不断与循环计数器的输出进行比较，当调制信号大于循环计数器的输出时，比较器输出高电平，否则输出低电平。循环计数器循环一个周期后，向寄存器发出一个使能信号EN，寄存器送入下一组数据。在每一个计数器计数周期，由于输入的调制信号的大小不同，比较器输出端输出的高电平个数不一样，因而产生出占空比不同的脉冲宽度调制波。



采用预分频器寄存器,计数器、数字比较器和寄存器进行组合完成PWM模块的设计，根据要求设计模型要求。

1. 分频器：采用数控分频器完成频率可调。若输入信号为50MHZ,要求PWM输出信号频率为1000HZ ,分频器分频比为195:1;
2. 计数器：采用8位计数器对频率可调信号进行计数，脉宽分辨率为1/28 即1/256。计数其周而复始的从0计数到255，再复位到0，重新开始计数，每次复位到0时，产生寄存器更新使能信号EN，控制脉宽设置寄存器的数据更新。
3. 比较器：其计数结果与占空比调整字相比较,若计数器值大于调整字，输出为1,否则输出为0.改变调整字大小,改变了输出信号的脉宽.
4. 脉宽设置寄存器，在更新使能信号EN的控制下实现数据的更新，该寄存器设计可参照基本D触发器的设计，将其扩展成8位即可得到。

参考设计：



### 实验结果

实验代码：

仿真结果(数控分频器)：

引脚分配表：

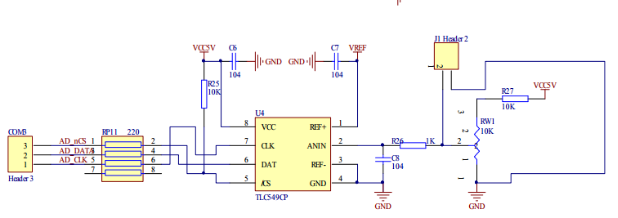
输出波形图：

## 实验十 A/D 接口电路

### 实验目的

1. 熟悉串行模数转换芯片 TL549 的使用方法。
2. 掌握利用有限状态机实现时序逻辑电路设计的方法
3. 一般状态机的设计与应用

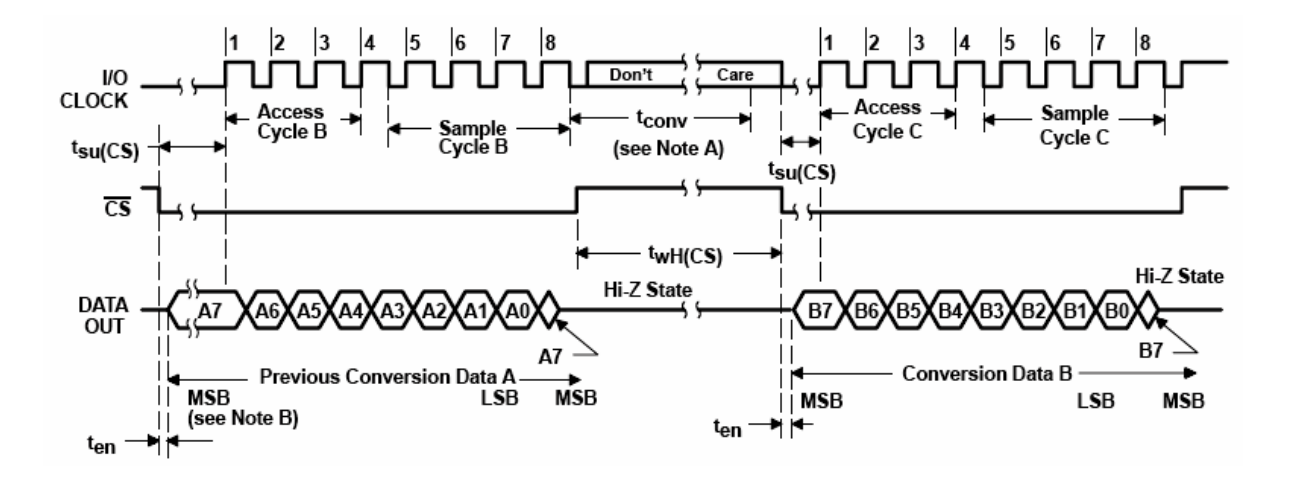
### 实验内容

本实验的内容是使用状态机实现对 TL549 的采样控制，实现一个简单的电压表。模数转换电路原理图如图所示。利用 QuartusⅡ完成设计，仿真等工作，最后在实验平台上进行硬件测试。实验时通 过调节电位器 RW1 改变 ADC 的模拟输入值，数据采样读取后由数码管 显示。最后用万用表测量 TP1 的输入电压，并与读取到的数据（经换算后的数据）做比较。 

### 实验原理及步骤

TLC549 是一个 8 位的串行模数转换器，AD 转换时间最大 19 微秒，I/O 时钟可达 1.1MHz。如图所示为 TLC549 的访问时序，从图中可以看出由当 CS/拉低低，ADC 前一次的转换数据（A）的最高位 A7 立 即出现在数据线 DATA OUT 上，之后的数据在时钟 I/O CLOCK 的下沿改变，可在 I/O CLOCK 的上升沿读取 数据。读完 8 位数据后，ADC 开始转换这一次采样的信号（B），以便下一次读取。转换时片选信号 CS 要置 高电平。设计操作时序是要注意 Tsu(CS),Tconv 及 I/O CLOCK 的频率几个参数。Tsu(CS)为 CS/拉低到 I/O CLOCK 第一个时钟到来时间，至少要 1.4 微秒；Tconv 为 ADC 的转换时钟，不超过 17 微秒；I/O CLOCK 不能超过 1.1MHz。其他的参数参考数据手册。

由于 ADC 是 8 位的，所以采样的电压值为 其中 V 为采样的电压值，D 为 ADC 转换后读取的 8 位二进制书，*Vref* 为参考电压值，



实验步骤  
1）启动 QuartusⅡ建立一个空白工程，然后命名为 tlc549adc.qpf  
2）新建 Verilog HDL 源程序文件 adc.v,写出程序代码并保存（完整的 Verilog HDL 程序）， 进行综合编译，若在编译程序中发现错误，则找出并更正错误，直接编译成功为止。  
3）建立波形仿真文件并对 adc.v 进行仿真验证，验证仿真结果是否与设计相符合，如果不符合，重  
新设计文件，再进行综合编译，仿真。直到仿真结果与设计相符合为止。

### 实验结果

实验代码：

仿真结果(数控分频器)：

引脚分配表：

输出波形截图：

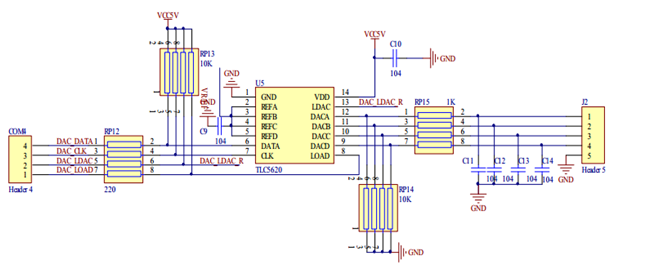
## 实验十 D/A 接口电路

### 实验目的

1. 熟悉串行数模转换芯片 TLC5620 的使用方法。
2. 掌握利用有限状态机实现时序逻辑电路设计的方法
3. 一般状态机的设计与应用

### 实验内容

实验平台有一个 4 通道 8 位 D/A 转换器 TLC5620。对 TLC5620 的控制基本上和实验十A/D 转换器 TLC549的控制一样。TLC5620 数模转换电路原理图如图 14.1 所示。本实验的内容是使用状态机时序实现对 TLC5620的控制，使 A，B，C，D 四个通道分别输出期望的电压值，用万用表测量输出电压并与理论值做比较。利用 QuartusⅡ完成设计，仿真等工作，最后在实验平台上进行硬件测试。



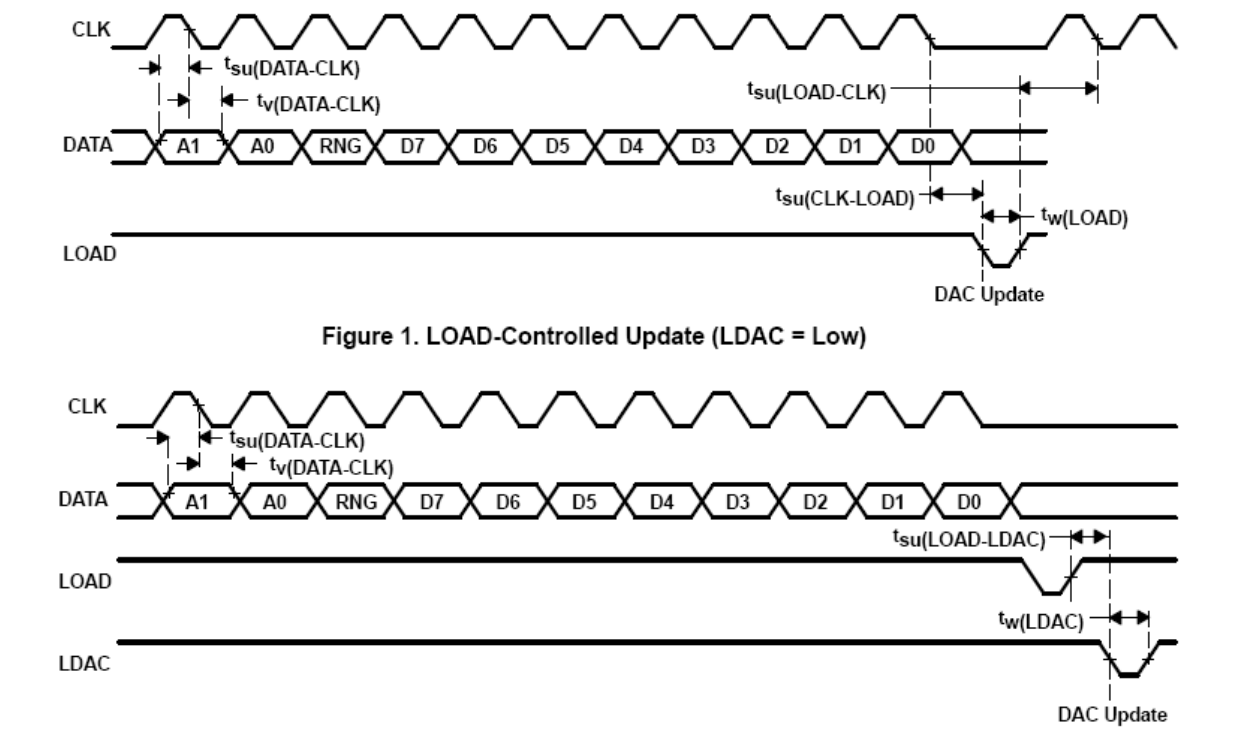
### 实验原理及步骤

## TLC5620 是一个 4 通道 8 位的串行 D/A 转换器，时钟频率最大可达到1MHz.TLC5620C 传输的一帧数据为 11 位，高位先传送。D10，D9 位为通道选择位，D8 位为输出电压模式选择，若为“0”则输出电压范围为 0-VREF， 若为“1”则输出电压范围为 0-2VREF，其中 VREF 为参考电压，本实验平台提供参考电压值为 2.5V：D7-D0 为数据位。

## TLC5620 的访问时序如下图所示，其中 CLK 为 DAC 的时钟：DATA 为串行数据，锁存于时钟的下沿；LOAD 为数据锁存控制信号，当一帧数据传送完毕时。拉低 LOAD 信号线，即把该帧数据锁存起来；LDAC 为DAC 输出更新控制信号，当 LDAC 为低电平时，则把锁存在数据器的数据传送到 DAC 转换器并转换输出。本本实验始终置 LDAC 为低电平，也就是说加载信号（LDAD）一旦产生，数据即刻转换输出。详细的时序控制，参数设置请参考数据手册。

TLC5620 的输出电压为:

Vo(DACA︱B︱C︱D)=REF×CODE/256×(1+RNG)  
其中 Vo 为输出电压值，DACA︱B︱C︱D 代表四个不同的通道，REF 为参考电压，CODE 为 8 位二进制数，范围为 0-255，RNG 为 RNG 位，数值为 0 或 1。



## 实验步骤 1）启动 QuartusⅡ建立一个空白工程，然后命名为 tlc5620adc.qpf 2）新建 Verilog HDL 源程序文件 dac.v,写出程序代码并保存进行综合编译，若在编译程序中发现错误，则找出并更正错误，直接编译成功为止。 3）建立波形仿真文件并对 adc.v 进行仿真验证，验证仿真结果是否与设计相符合，如果不符合，重新设计文件，再进行综合编译，仿真。直到仿真结果与设计相符合为止。

### 实验结果

实验代码：

仿真结果(数控分频器)：

引脚分配表：

## 实验十二 VGA接口电路

### 实验目的

1. 了解VGA信号的特性及接口电路的原理。
2. 掌握VGA接口电路的设计方法。

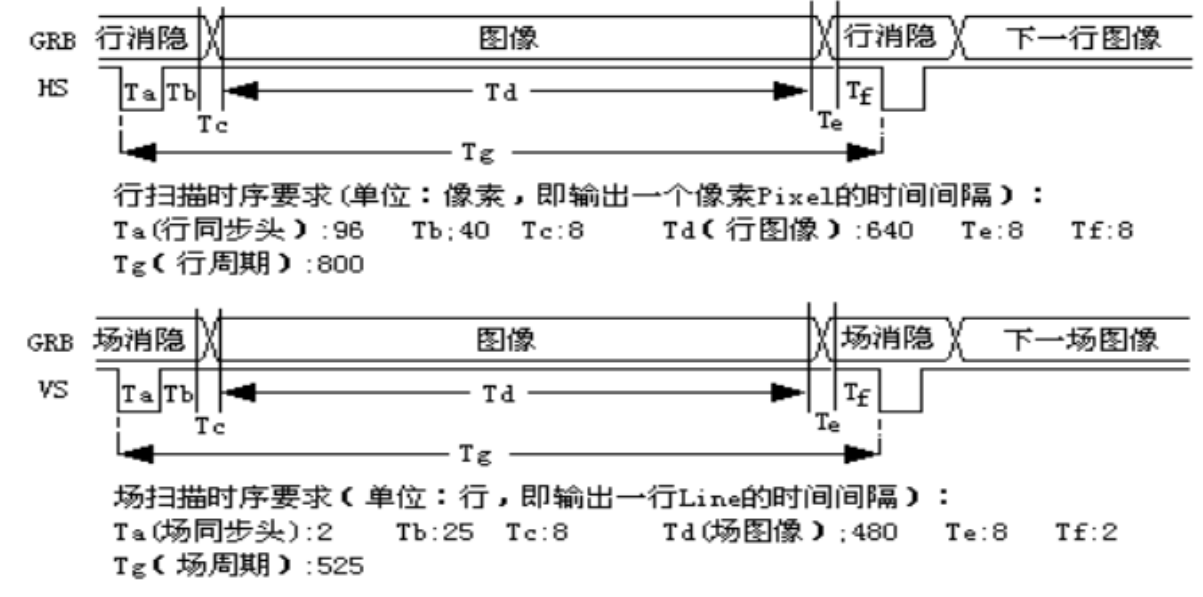
### 实验内容

本实验的内容是用 FPGA 来实现 VGA 图象控制器，控制显示器显示彩色信号，分别显示横彩条，竖彩条和横盘格。利用 QuartusⅡ完成设 计，仿真工作等最后在进行下载硬件测试

### 实验原理及步骤

计算机显示器的显示有许多标准，常见的有 VGA，SVGA 等.在这里我们用 VGA 接口来控制显示器，VGA 是 Video Graphics Adapter 的缩写，即视频图形阵列.作为一种标准的控制显示接口得到广泛的应用.VGA 接口的引脚分配如后脚本清单。常见的彩色显示器一般由 CRT(阴极射线管)构成，色彩是由 R，G，B(红：red 绿：green 蓝 blue)三

基色组成.显示是用逐行扫描的方式解决，阴极射线枪发出电子束打在涂有银光粉的荧光频上，产生 RGB 三基色，合成一个彩色像素。扫描从频幕的左上方开始，从左到右，从上到下，进行扫描，每扫完一行， 电子束回到频幕左边的下一行的起始位置，在这期间，CRT 对电子束进行消隐，没行结束时，用行同步信 号进行行同步；扫描完所有行，用场同步信号进行场同步，并使扫描回到屏幕的左上方，同步进行场消隐， 预备下一场的扫描。 对于普通的 VGA 显示器，共有 5 个信号：R、G、B 三基色信号；HS：行同步信号； VS：场同步信号。 对于时序驱动，VGA 显示器要严格遵循“VGA 工业标准”，即 640ⅹ480ⅹ60Hz 模式，否则可能会损害 VGA显示器。 通常我们用的显示器都满足工业标准，应此我们设计VGA控制器时要参考显示器的技术规格。如图16.2 所示是 VGA 显示器行扫描、场扫描的时序图。



VGA 工业标准要求的频率如下：

时钟频率 （Clock frequency） 25.175MHz

行频 (Line frequency) 31.459KHz

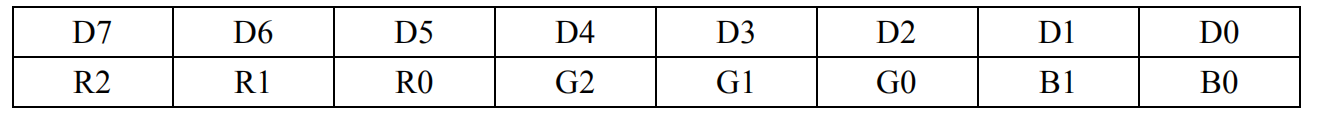
场频 (Field frequency) 59.94Hz

VGA 工业标准要求：行，场同步为负极，即同步头脉要求是负脉冲。

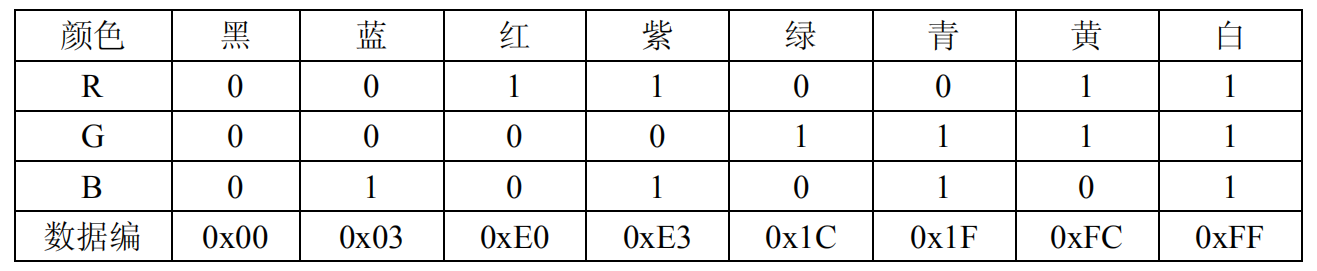
设计时要注意时序及电平驱动，详细情况可参照相关资料

如图 16.2 所示为 VGA 图象显示扫描示意图，在设计时，可用两个计数器进行记数（行，场扫描计数器）， 行计数器的驱动时钟为 25MHz，场计数器的驱动时钟为行计数器的益处信号。计数的同时控制行，场同步 信号输出，并在适当的时候送出数据，就能显示相应的图象。注意消隐期间送出的数据应为 0x00。显示器 的刷新频率为 25MHz/800/525=59.52Hz，接近 VGA 工业标准场频 59.94Hz。

实验平台的 256 色 VGA 接口提供 8 位数据输入，三基色信号 R、G、B 共占用 8 位（分别为 R：3 位、G： 3 位、B：2 位），因此可以显示 256 种颜色，RGB 数据的格式如表 16.1 所示。



本设计是产生 8 种颜色的彩条信号。分四种显示模式，分别是横彩条信号、竖彩条信号和两种模式的 棋盘格。颜色编码表如表 16.2 所列。



实验步骤

1）启动 QuartusⅡ建立一个空白工程，然后命名为 vga.qpf。

2）建立 PLL 宏单元，命名为 pll，设置输入频率为 50MHz，c0 输出频率为 25MHz(VGA 时钟)。

3）新建 VerilogHDL 源程序文件 vga.v，查阅文献资料，用Verilog语言设计电路写出程序代码并保存进行综合编译，若在编译过程中发现错误，则找出并更正错误，直至编译成功为止。

4）建立波形仿真文件并进行仿真验证，验证仿真结果是否与设计相符，如果不符合，需重新设计文 件，再进行综合编译、仿真。直到仿真结果与设计相符合为止。

5）选择目标器件并对相应的引脚进行锁定，引脚锁定方法脚本清单 16.1。将未使用的引脚设置为三 态输入（一定要设置，否则可能会损坏芯片）。

6）对该工程文件进行全程编译处理，若在编译过程中发现错误，则找出错误，直至编译成功为止。

7）显示器的 VGA 接口线接到实验平台上的 256 色 VGA 接口上，并将程序下载到 FPGA 器件中。观察显示器的变化。

### 实验结果

实验代码：

仿真结果：

引脚分配表

\*下载验证（照片）

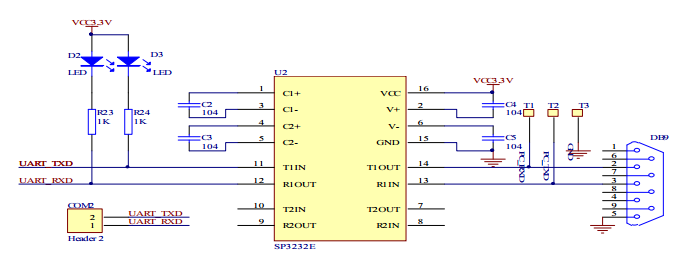
## 实验十三 通用异步收发器（ＵＡＲＴ）

### 实验目的 1.掌握通用收发器的设计。

### 实验内容

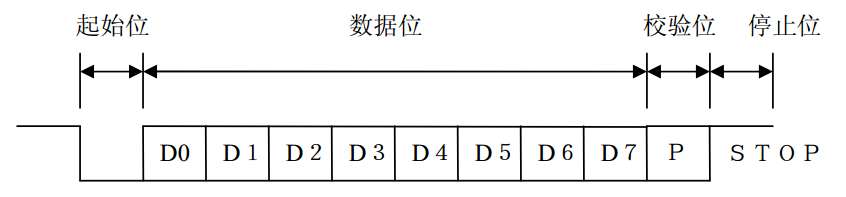
1. 利用 Quartus II 完成设计、仿真等工具，完成通用异步收发器（ＵＡＲＴ）的设计；
2. 要求收发器的波特率为 9600、8位数据、1位起始位和1位停止位的格式进行数据传输；

### 实验原理和步骤

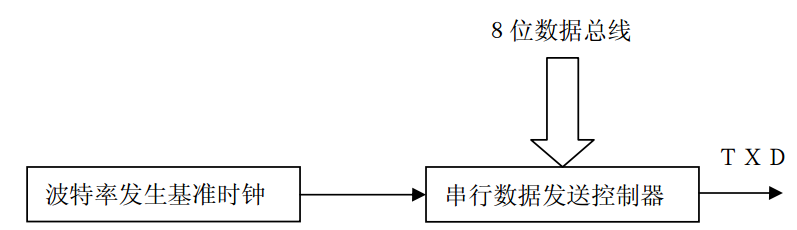


RS232外围硬件原理图

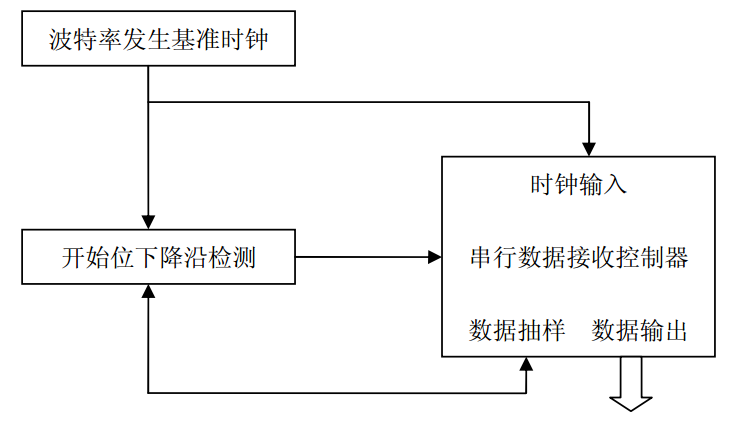
通用异步收发器，是一种广泛的短距离串行传输接口。下面简单介绍数据传输的基本原理。由于波特率为 9600 的一个信息位所需要的时间为 ， 也就是说要每隔发送一位数据，剩下就是要求发送的数据符合串行数据的格式。串行通信的数据格式如下图2 所示。首先发开始位低电平，接着从 D0 到 D7 发送８位数据，最后发送停止位的高电平。



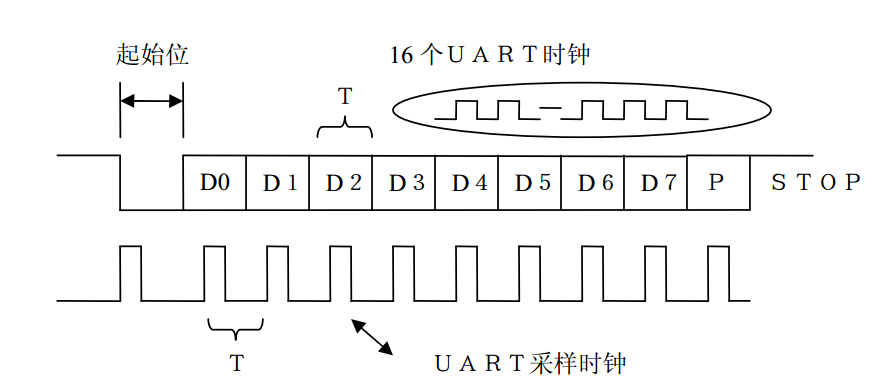
**串行数据发送**的结构图如所示，由基准时钟模块产生一个的时间，当要发送数据时，串行数据发送控制器把数据总线上的内容加上开始位和结束位，然后进行移位发送。



**串口数据接收**的结构如下图所示。由基准时钟产生一个 16 倍于波特率的频率，这样就把一个位的数据分成１６份了，当检测到开始位的下降沿，就开始进行数据采样。采样的数据为一个位的第６、７、８三个状态，然后三个里面取两个以上相同的值作为采样的结果，这样可以避免干扰。当开始位的采样结果不是０的时候就判定接收为错，把串行数据接收控制器的位计数器复位。当接收完１０位数据后就进行数据的输出，并把串行数据接收控制器的位计数器复位，等待下一数据的到来。



以9600波特率为例，则波特率基准时钟频率为，此频率为数据的采样时钟，即每个数据位进行16次采样。主时钟为50MHz情况下，需要分频系数为325:1的分频器，产生153600Hz的波特率基准时钟。



实验步骤

1. 启动 Quartus II 建立一个空白工程，然后命名为 uart.qpf。
2. 新建 Verilog HDL 源程序文件 rec.v 和 send.v，写出程序代码并保存， 进行综合编译，若在编译过程中发现错误，则找出并更正错误，直至编译成功为止。
3. 将光盘中 uart 目录下的 uart\_test.bsf、uart\_test.v 拷贝到工程目录。新建图形设计文件进行硬件测试，命名为 uart.bdf 并保存。
4. 把程序下载到 FPGA 器件中，用串口线将开发板和计算机相连，打开串口调试软件，设置使用串口 COM１，波特率 9600bps，８位数据位和１位停止位，在发送窗口输出两个字符，设置为 16 进制格式，按“发送”按钮，观察数码管 1、2 的状态。在实验平台上按 KEY2/RESET，输入两位 16 进制数，按 KEY3 发送，观察串口调试软件的接收窗口。

### 实验结果

实验代码：

引脚分配表

\*下载验证（照片）

## 实验十四 数字锁相环PLL的应用

### 实验目的 1.掌握 CycloneⅡ器件内嵌相环的使用。

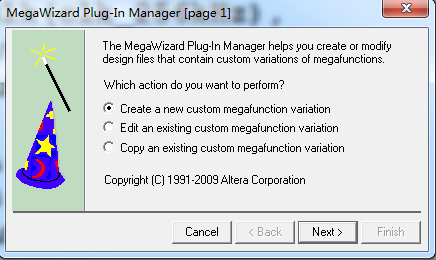
### 实验内容

1. 调用嵌入锁相环 PLL，产生时钟的倍频信号；
2. 调用频率计模块，对倍频后的时钟信号频率进行测量；
3. 改变锁相环PLL的倍频系数，观察频率计的变化；

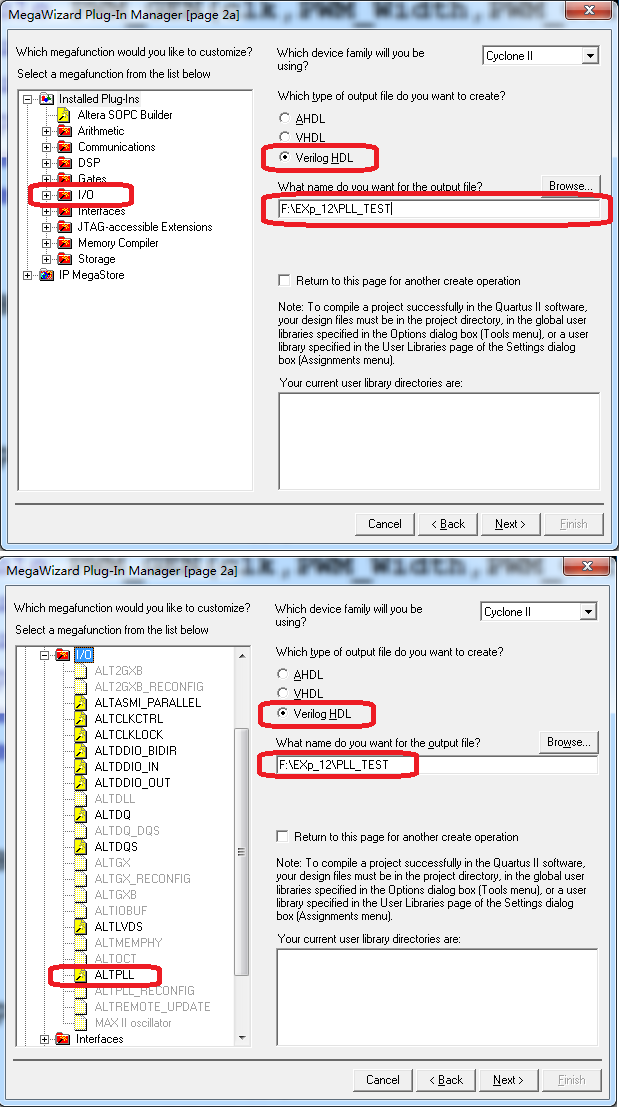
### 实验原理和步骤

1）启动 QuartusⅡ建立一个空白工程，然后命名为 pll\_test.qpf。  
2）利用MEGA Function向导建立数字锁相环模块。

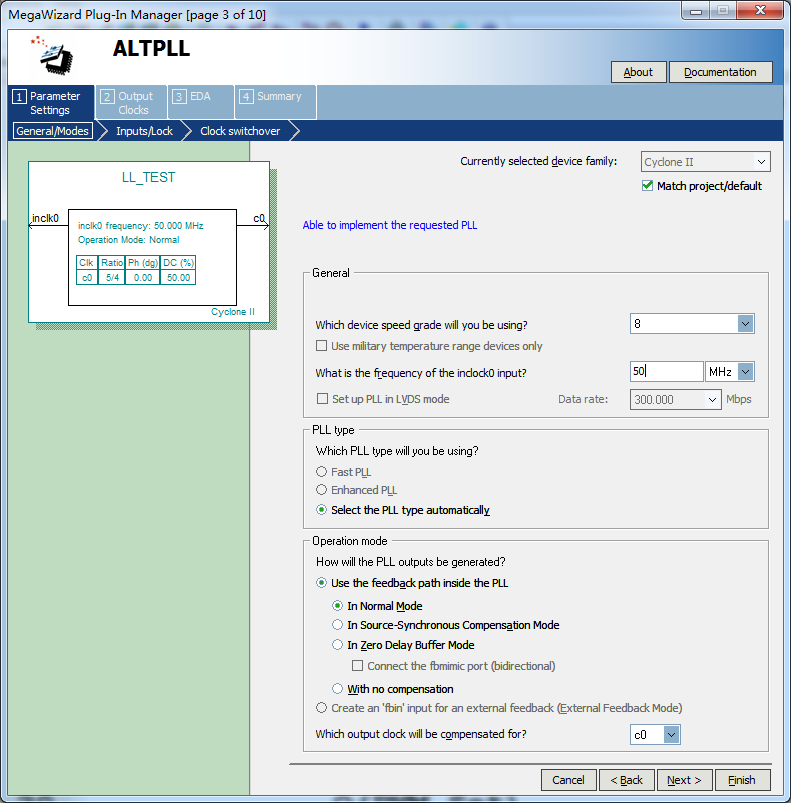
Step1：打开 QuartusⅡ工程，选择 Tool〉〉 MegaWizard Plug-In Manager 如下图所示的添加宏单元的向导Page1，选择Create a New……，然后点击 Next 进入向导page2。



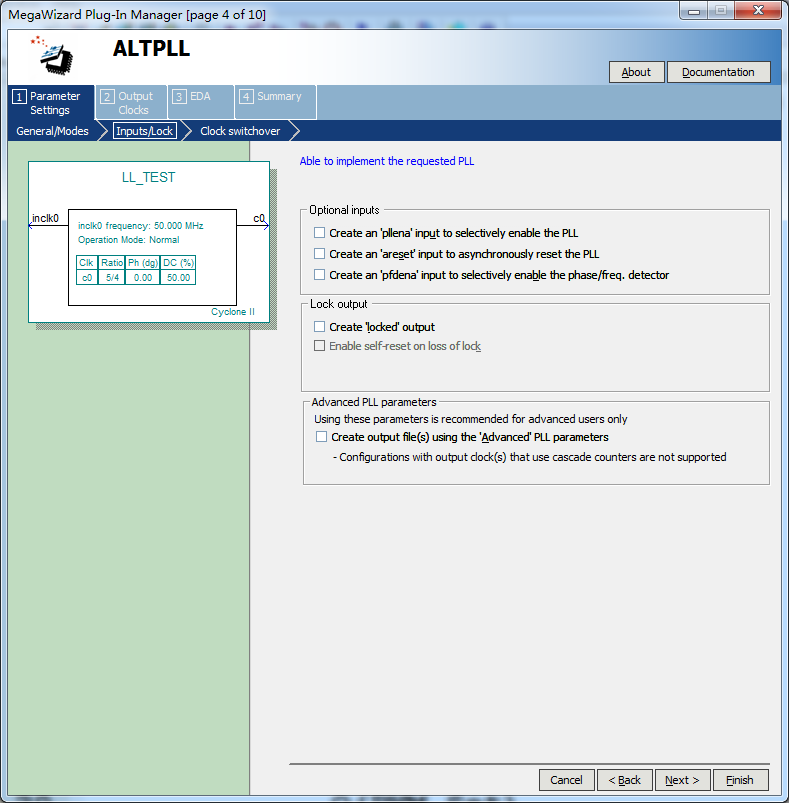
Step2:如下图选择ALTPLL模块，设定输出语言类型和文件名，然后点击next，进入Page3。

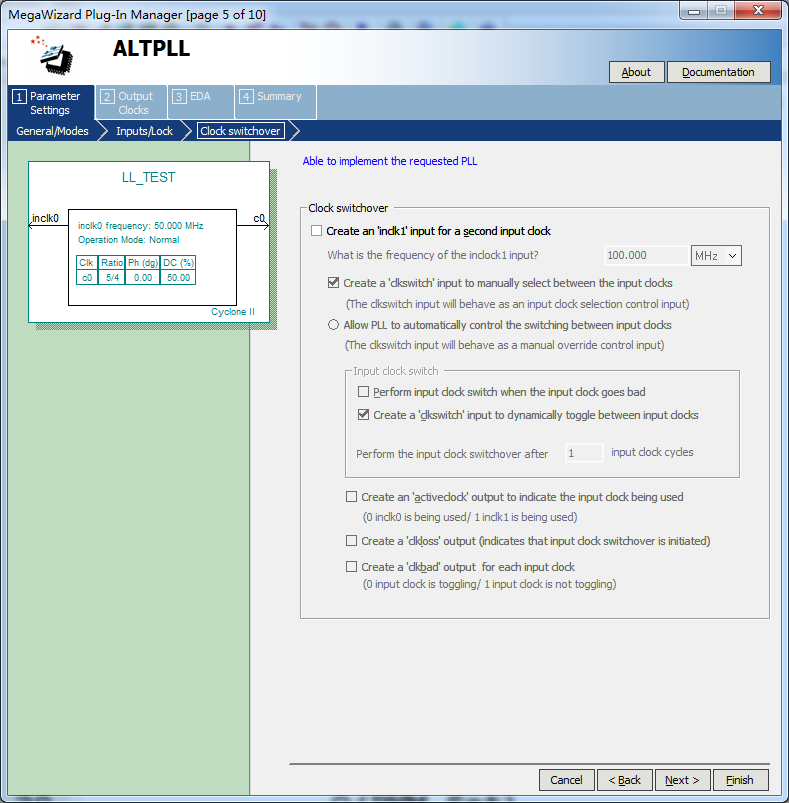


Step3:如下图根据芯片信号设定芯片的速度等级，在此我们选择等级“8”，，然后根据开发板提供的主时钟信号的频率，设定锁相环输入时钟频率，在此设定位“50MHz”，然后点击next，进入Page4。

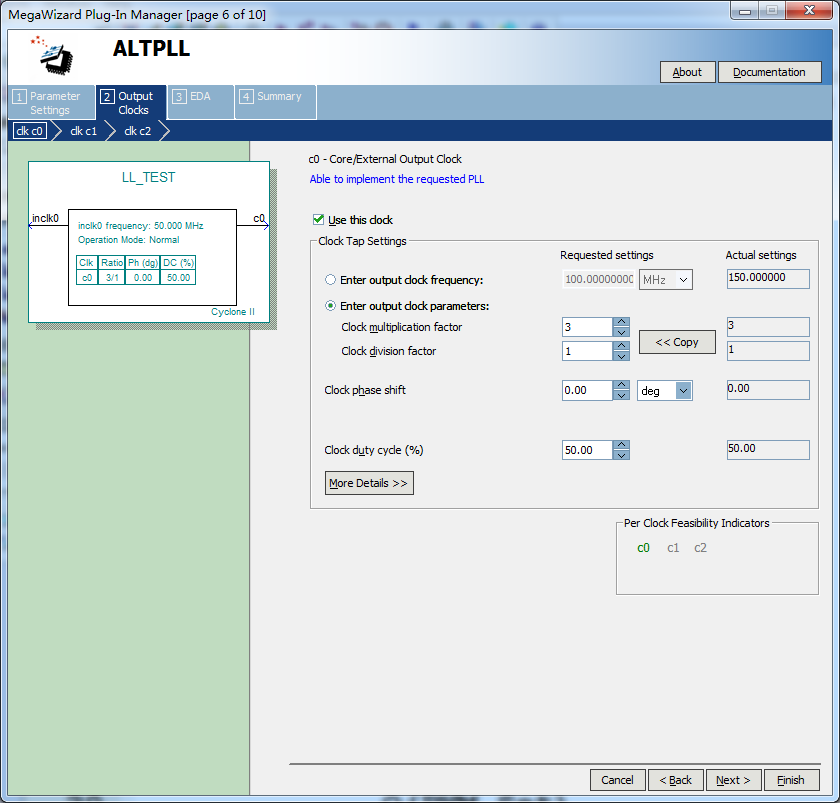


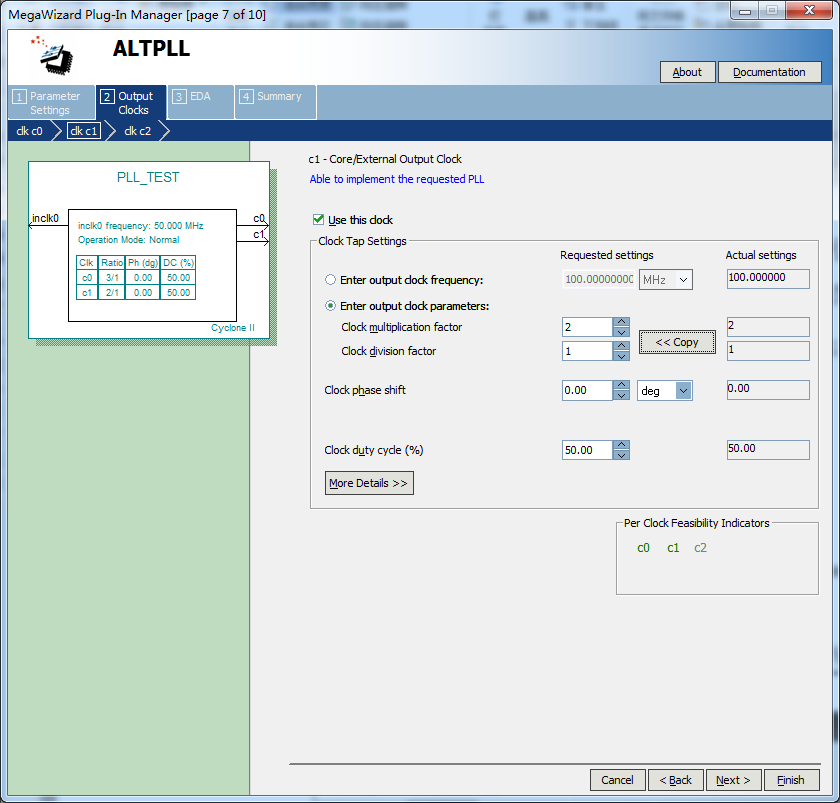
Step4:在Page4中取消任何控制信号的选择，点击next，在Page5中无需做其他任何操作，直接点击next，进入Page6。



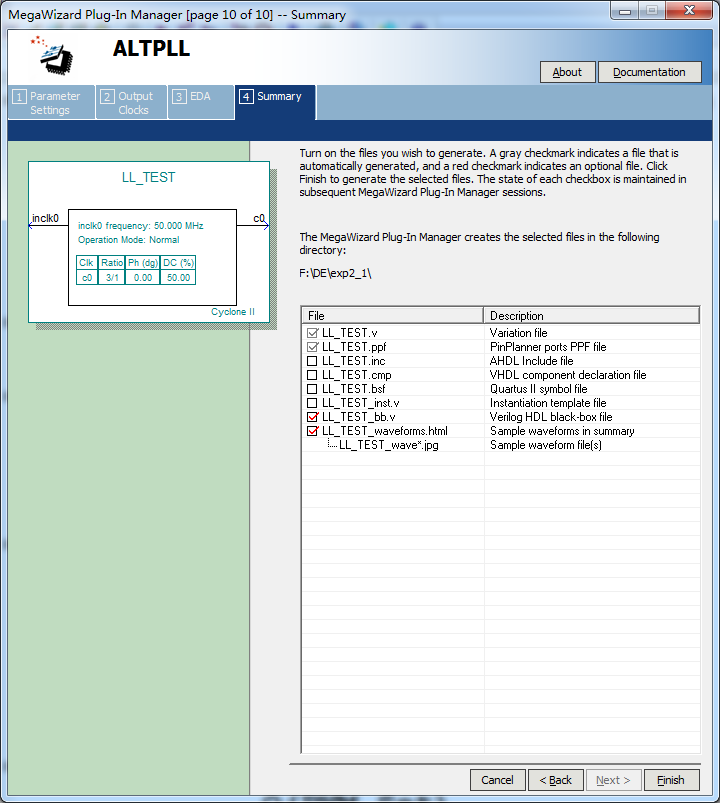


Step5: Page6中，根据输入时钟频率和倍频后期望的输出频率，设定倍频系数和分频系数，例如我们输出信号频率50MHz，输出频率150MHz，就可以设定倍频系数为3，分频系数为1，即将输出频率设定为150MHz然后点击next，进入Page7。



Step6：在Page7中设定锁相环产生另一路倍频信号输出，设定频率输出100MHz，选择合适倍频系数和分频系数，作为频率计的基准时钟信号，然后点击next，进入Page8；

Step7:Page8和Page9界面中无需其他操作，直接点击next，在Page10中点击Finishing，完成数字锁相环模块的生成。

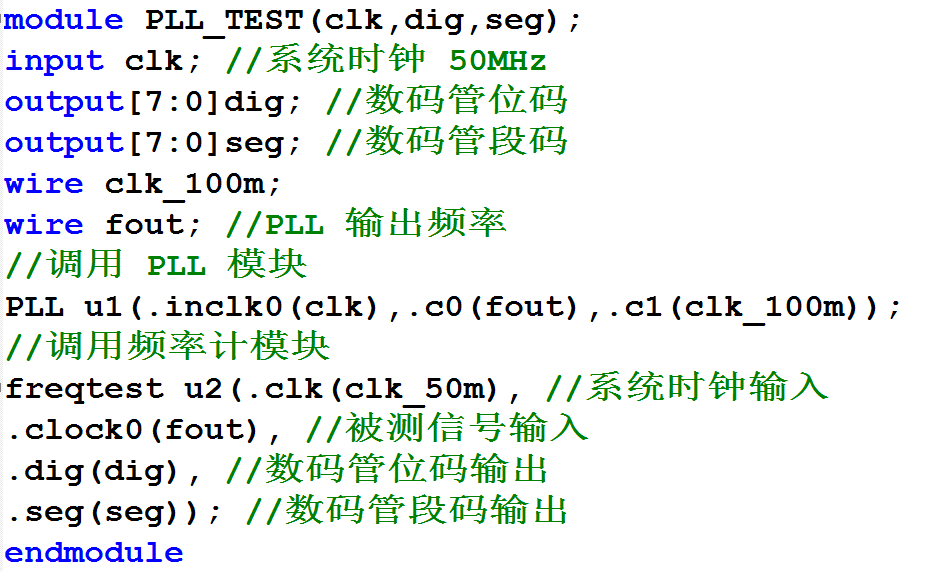


3）新建 Verilog HDL 源程序文件 pll\_test.v,编写顶层模块，调用 pll.v 和频率计模块 freqtest.v，进行综合编译，若在编译程序中发现错误，则找出并更正错误，直接编译成功为止。

4)选择目标器件并对应的引脚进行锁定，对该工程文件 进行全程编译处理，若在编译过程中发现错误，则找出并更正错误，直至编译成功为止，最后将程序下载到 FPGA 器件中。观察测得的频率并与锁相环设计的频率值做比较。

5) 更改 PLL 的 C0 输出频率值分别为 20MHz，50MHz，100MHz，120MHz，150MHz 并观察数码管的显示值。

**参考顶层程序：**



### 实验结果

实验代码：

引脚分配表

\*下载验证（照片）