

# 数字逻辑电路实验

## 讲义

## 目录

实验 1: 组合逻辑电路(1)—用 74 系列芯片搭建电路并观测竞争与险象.....	4
一 实验准备 .....	4
二、实验内容 .....	5
三、实验报告 .....	8
实验 2: 组合逻辑电路(2)—译码器电路设计 .....	9
一 实验准备 .....	9
二 实验内容 .....	10
三 实验报告 .....	11
实验 3: 组合逻辑电路(3)—组合逻辑电路设计 .....	12
一 实验准备 .....	12
二 实验内容 .....	13
三 器件编程步骤 .....	15
四 实验报告 .....	19
实验 4: 时序逻辑电路(1)—双稳态元件功能测试 .....	20
一 实验准备 .....	20
二 实验内容 .....	20
三 实验报告 .....	20
实验 5: 时序逻辑电路(2)—计数器的设计与应用 .....	22
一 实验准备 .....	22
二 实验内容 .....	22
三 实验报告 .....	24
实验 6: 时序逻辑电路(3)—时序逻辑电路设计 .....	25
一 实验准备 .....	25
二 实验内容 .....	25
三 实验报告 .....	26
实验 7-实验 8: 数字系统设计 .....	27
一 实验准备 .....	27
二 实验内容 .....	27
三 实验报告 .....	32

附录 1：Quartus Prime18.1 软件设计流程.....	33
一、打开软件 .....	33
二、创建新工程.....	35
三、设计输入 .....	40
四、编译.....	45
五、仿真.....	47
六、管脚分配 .....	52
七、器件编程 .....	52
附录 2：Quartus Prime18.1 器件编程流程.....	53
附录 3：Verilog 语法初步 .....	57

# 实验 1：组合逻辑电路(1)—用 74 系列芯片搭建电路 并观测竞争与险象

## 一 实验准备

1. 查阅 示波器 使用手册，学习基本方法。型号 为 GDS -200 2E。
2. 查阅 信号源 使用 手册，学习基本使用方法。信号 源型号为 SDG 2122X 。
3. 74系列门电路简介

74 系列门电路指的是一个系列的数字集成电路，其中有 74XXX(现已不使用)、74SXXX、74LSXXX、74FXXX、74CXXX、74HCXXX、74HCTXXX、74AXXX、74ASXXX、74ACTXXX 等多种系列的芯片，对于“XXX”表示芯片的类型，是一串数字(如 00，08，20，138，245，373，573，4066 等)，只要数字相同，其逻辑功能就相同，只是性能的差异。

4. 查阅 SN7400芯片手册，了解SN7400芯片的内部结构和封装后管脚与内部电路的对应关系

SN7400芯片的实物图和管脚图如下所示：



图1-1 SN7400实物图

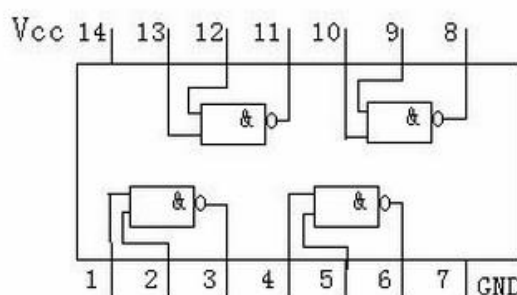


图1-2 SN7400管脚图

5. 实验箱的简单介绍

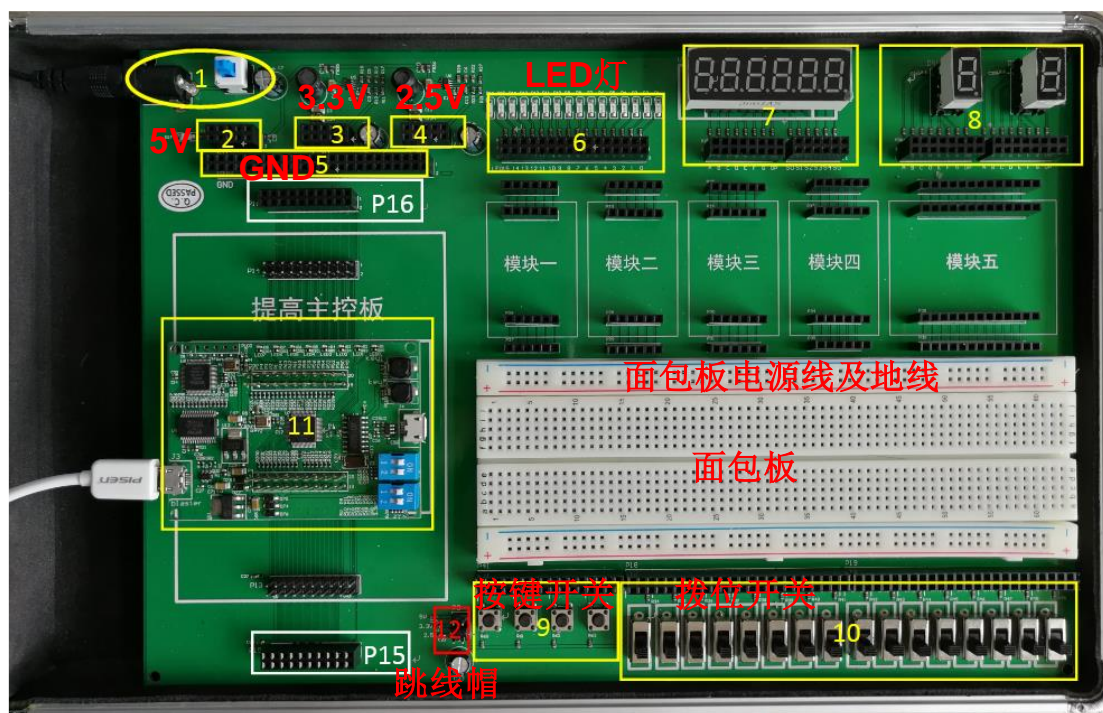


图1-3 实验箱全貌

1. 电源接口和电源开关
2. 5V电源插孔
3. 3.3V电源插孔
4. 2.5V电源插孔
5. GND信号插孔
6. LED灯及其信号插孔
7. 6位共阴极7段数码管及其插孔
8. 两位共阳极7段数码管及其插孔
9. 按键及其插孔
10. 拨位开关及其插孔
11. 核心主控板，芯片型号为5M160ZE64 （实验3中会详细介绍）
12. 跳线（用于配置按键和拨位开关的IO电平）

## 二、实验内容

### 实验内容1：测量示波器的探头补偿信号

打开示波器电源，将探头补偿信号接到CH1的输入端，按Autoset键，观察波形；

调整水平刻度，观察信号的变化；

调整垂直刻度，观察信号的变化；

调整触发电平，观察信号的变化。

**思考：若没有Autoset键，要如何设置水平刻度和垂直刻度来保证信号的正确显示？**

实验内容2：用示波器观察信号源输出的正弦信号

设置信号发生器的ch1输出1KHz、峰峰值5V的正弦信号；

连接示波器的ch1和信号发生器的ch1；

调整示波器水平刻度、垂直刻度及触发电平，使波形正确清晰的显示。记录此时的水平刻度和垂直刻度并记录波形；

改变信号发生器输出的正弦信号的频率，不改变峰峰值，观察示波器捕获的波形的峰峰值随频率的变化情况，分析原因。记录关键点的波形及示波器参数设置。

**思考：示波器的带宽对实际测量有什么影响？**

实验内容3：用示波器观察信号源输出的方波信号

设置信号发生器的ch2输出1KHz、峰峰值5V、占空比50%的方波信号；

连接示波器的ch2和信号发生器的ch2；

调整示波器的水平刻度、垂直刻度及触发电平，使波形正确显示。记录此时的水平刻度和垂直刻度以及示波器波形。

改变方波频率为1MHz、10MHz、20MHz，观察并记录波形。

**思考：示波器捕获的方波还那么方吗？分析原因**

实验内容4：测量示波器的带宽

用信号发生器输出5V峰峰值无直流分量的正弦波；

改变正弦波的频率，用示波器观察峰峰值降为 $5 \times 0.707 = 3.5\text{V}$ 时的正弦波的频率，该频率即为要测量的示波器的带宽。记录该频率值。

**思考：分析这种测量方法存在的问题。**

实验内容5：74HC00 芯片功能测试

从芯片的器件手册中查找芯片各管脚的定义，在实验箱上将芯片插到面包板上的合适位置，用实验箱上的5V和GND信号给芯片供电。使用拨位开关作为输入，LED灯作为输出，验证7400的逻辑功能，根据验证的结果写出真值表，确认4个与非门是否正常工作。

## 实验内容6：竞争与险象的产生及观测

### 1.6.1 分析存在险象的电路

分析表达式 $Y = B + \bar{B}$  可能存在的竞争和险象，考虑用与非门如何搭建可以产生该险象的电路

### 1.6.2 实现竞争与险象测量电路

将 $Y = B + \bar{B}$  变换为用与非门表示的形式 $Y = B + \bar{B} = \overline{A * \bar{A}} = 1$ ，其中 $A = B$ 。

7400包含4个与非门，与非门可以用来实现非运算。实验中使用3级与非门来实现非运算（3级非门的延迟能让大家观察到清晰的险象），所以总共是使用4个与非门正好是一个7400芯片来实现 $Y = B + \bar{B} = \overline{A * \bar{A}} = 1$ 逻辑电路。

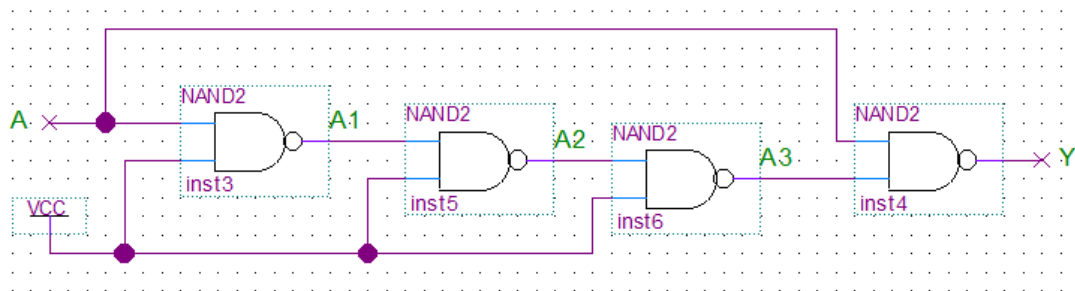


图1-4 竞争与险象实现电路原理图

### 1.6.3 通过示波器的单次触发功能捕获险象

搭建好电路后，使用拨位开关提供输入信号A，使用示波器的单次触发功能捕获各个关键信号，直到最后捕获到静态1险象。

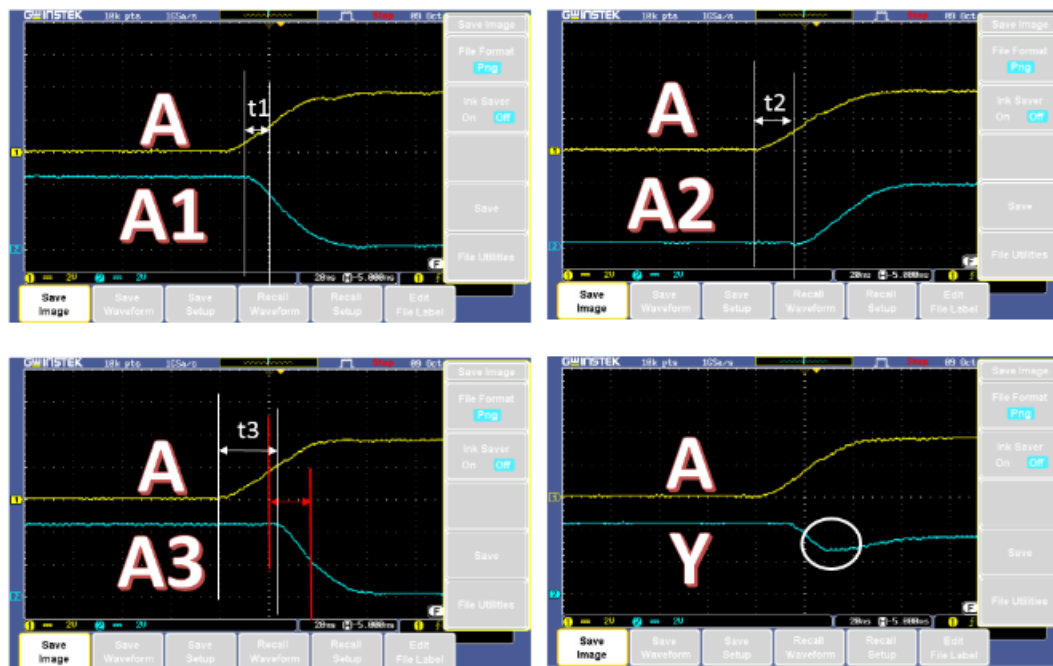


图1-5 竞争与险象测量中的四次观测结果

思考：A1从0变为1,或从1变为0都有竞争的发生，但是都会产生险象吗？分析原因

思考：通过7400能够设计动态险象？如果可以如何设计？

### 三、实验报告

#### 1、实验内容

#### 2、实验结果及波形记录

注意波形中应标明：横坐标、纵坐标、刻度值、关键点、幅度、频率

#### 3、实验内容5中7400的测试结果及真值表

#### 4、实验内容6中的电路逻辑表达式、险象分析、实验结果图

#### 5、思考题



## 实验 2：组合逻辑电路(2)—译码器电路设计

### 一 实验准备

1. 学习Quartus Prime软件基本使用。参考附录1。

#### 2. 74系列38译码器（74LS138）

复习数电教材38译码器部分章节。下图是74 LS138的管脚图和真值表，其中H表示高，L表示低，X表示不确定的值。

				FUNCTION TABLE													
				INPUTS			OUTPUTS										
				ENABLE		SELECT											
				G1	G2A	G2B	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
A	1	16	V <sub>CC</sub>	X	H	X	X	X	X	H	H	H	H	H	H	H	H
B	2	15	Y0	X	X	H	X	X	X	H	H	H	H	H	H	H	H
C	3	14	Y1	L	X	X	X	X	X	H	H	H	H	H	H	H	H
G2A	4	13	Y2	H	L	L	L	L	L	L	H	H	H	H	H	H	H
G2B	5	12	Y3	H	L	L	L	L	H	H	L	H	H	H	H	H	H
G1	6	11	Y4	H	L	L	L	H	H	H	H	L	H	H	H	H	H
Y7	7	10	Y5	H	L	L	L	H	L	H	H	H	H	L	H	H	H
GND	8	9	Y6	H	L	L	L	H	L	H	H	H	H	H	L	H	H
				H	L	L	L	H	H	H	H	H	H	H	H	H	L

#### 3. 全加器的实现

复习数电教材上全加器的相关内容，及最大最小项的相关内容。

##### 3.1一位全加器的逻辑门实现

一个全加器的输入端信号分别为：被加数输入 $X_i$ 、加数输入 $Y_i$ 、低位向本位的进位输入 $C_{i-1}$ ，输出端信号分别为：本位的和输出 $S_i$ 、本位向高位的进位输出 $C_i$ 。它的真值表如下图左所示。根据真值表可以得到和输出以及进位输出的逻辑表达式如下图右。根据表达式可以得到逻辑门电路。

$C_{i-1}y_i x_i$	$S_i C_i$
0 0 0	0 0
0 0 1	1 0
0 1 0	1 0
0 1 1	0 1
1 0 0	1 0
1 0 1	0 1
1 1 0	0 1
1 1 1	1 1

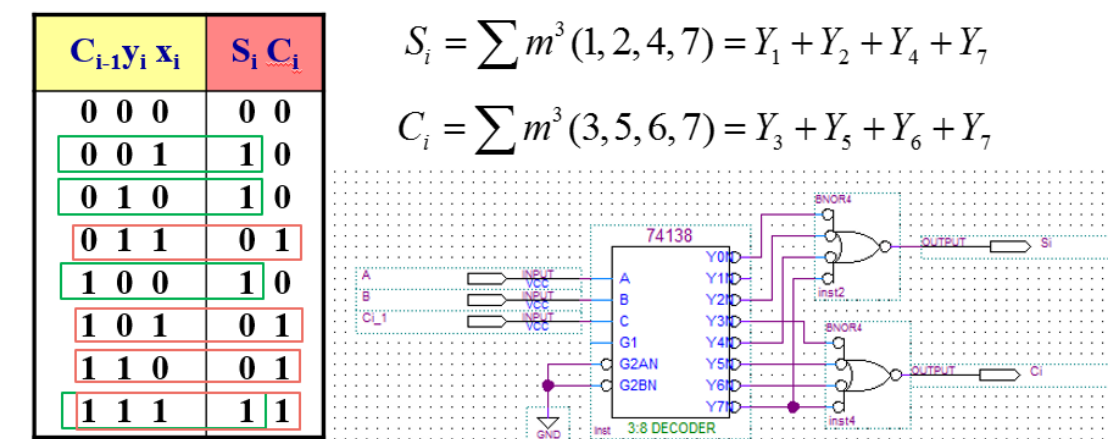
$$S_i = x_i \oplus y_i \oplus C_{i-1}$$

$$C_i = x_i y_i + C_{i-1}(x_i \oplus y_i)$$

$$= \overline{x_i y_i} * \overline{C_{i-1}(x_i \oplus y_i)}$$

### 3.2 利用38译码器实现一位全加器

38译码器也称为是最小项生成器。由一位全加器的真值表可以写出输出的最小项的与或式。借助38译码器和逻辑门电路就可以实现一位全加器。如下图所示。



## 二 实验内容

### 1. EDA软件Quartus Prime18.1软件的使用

在Quartus Prime中创建工程并添加原理图设计文件，实现与、或、非逻辑，通过波形仿真文件对设计进行仿真。

### 2. 在面包板上测试74LS138芯片的电路功能

### 3. 在Qaurtus Prime中用原理图实现138译码器功能，通过波形仿真进行验证

### 4. 在Quartus Prime中用38译码器实现一位全加器，通过波形仿真进行验证

## 三 实验报告

### 3.1实验内容

### 3.2实验步骤

### 3.3实验结果

#### 3.3.1 Quartus Prime软件设计流程

#### 3.3.2 与或非门电路及仿真结果

#### 3.3.3 74138芯片的引脚排列和测得的真值表

#### 3.3.4 用Quartus Prime设计的3-8译码器电路及仿真结果

#### 3.3.5 用Quartus Prime设计的一位全加器电路及仿真结果

### 3.4思考题

#### 3.4.1 Quartus Prime 有哪几种设计输入文件格式

#### 3.4.2 Quartus Prime中顶层实体的作用是什么？

## 实验 3：组合逻辑电路(3)—组合逻辑电路设计

### 一 实验准备

1. 学习数电教材上组合逻辑相关章节
2. 数电实验箱具体介绍

在实验2中已对实验箱进行了简单介绍

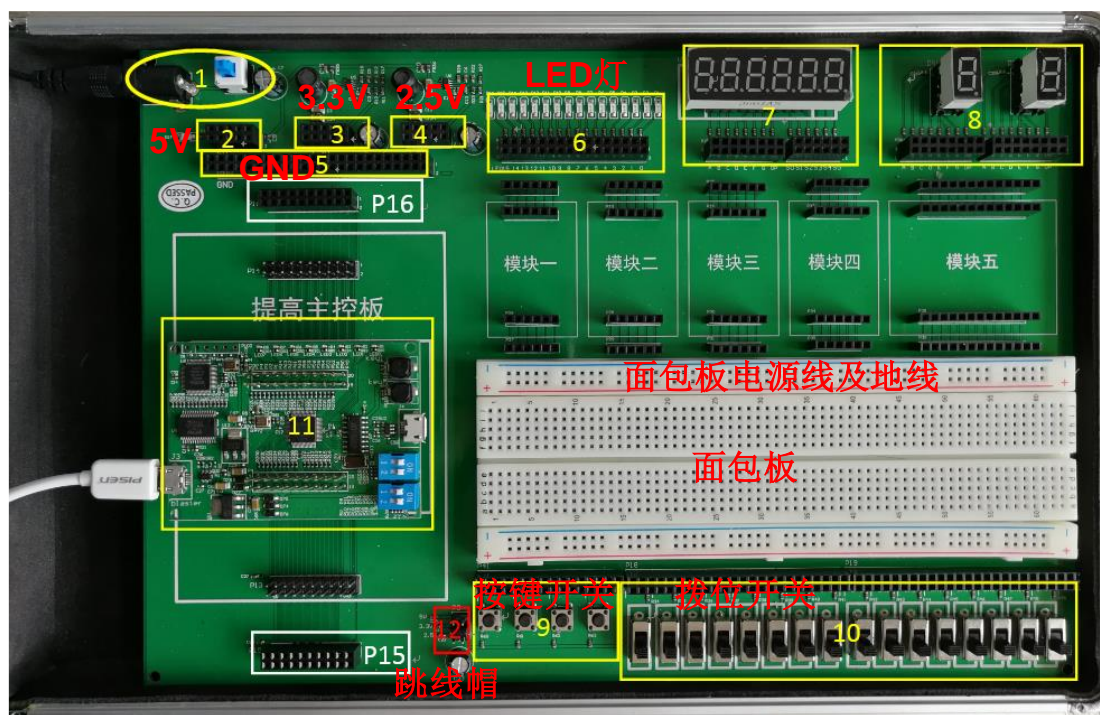


图1-3 实验箱全貌

1. 电源接口和电源开关
2. 5V电源插孔
3. 3.3V电源插孔
4. 2.5V电源插孔
5. GND信号插孔
6. LED灯及其信号插孔
7. 6位共阴极7段数码管及其插孔
8. 两位共阳极7段数码管及其插孔
9. 按键及其插孔
10. 拨位开关及其插孔
11. 核心主控板，芯片型号为5M160ZE64（实验3中会详细介绍）
12. 跳线（用于配置按键和拨位开关的IO电平）

P15: 核心板下面两排针脚的引出接口（与核心板下面两排针脚的信号一一对应）

P16: 核心板上上面两排针脚的引出接口（与核心板上上面两排针脚的信号一一对应）

核心板上两排针脚对应的信号：

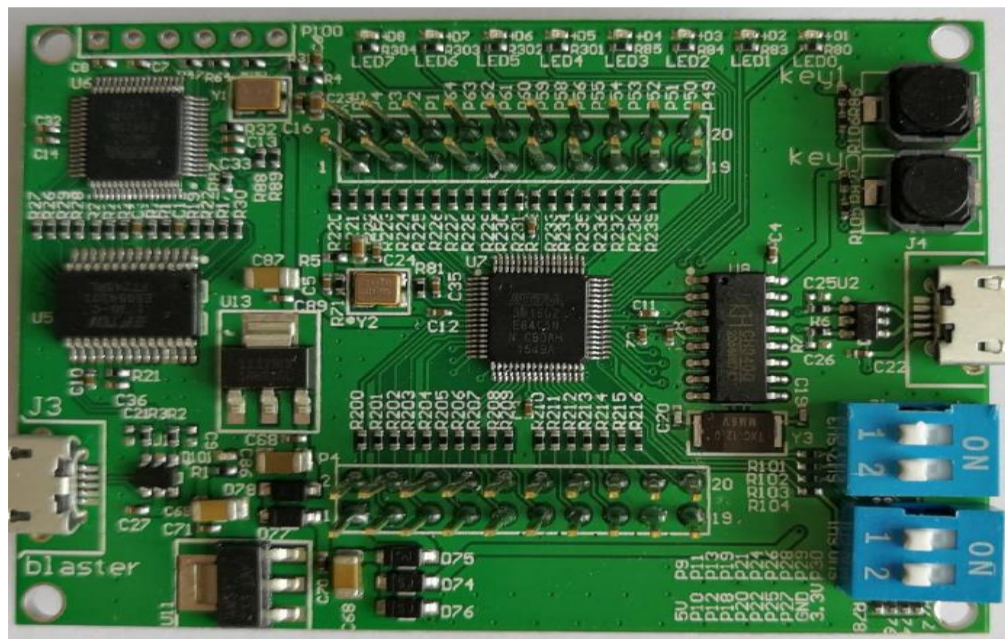


图3-1 实验箱的核心板

插针的管脚对应关系如下：

上端：第一排 P5 P3 P1 P63 P61 P59 P56 P54 P52 P50

上端：第二排 P4 P2 P64 P62 P60 P58 P55 P53 P51 P49

下端：第一排 P9 P11 P13 P19 P21 P24 P26 P28 P29 1P30

下端：第二排 5V P10 P12 P18 P20 P22 P25 P27 GND 3.3V

## 二 实验内容

在以下几个题目中选取一个设计和实现，并使用数电实验箱验证。也可以自己命题自己设计和实现并最终通过数电实验箱验证。

### 题目一：海明校验码

在通信系统中，因为干扰和噪声，经常会发生数据在传输到另一方后被识别错误。那么，就要考虑采用一些方法来加强通信的可靠性，74海明码是可以检错并且可以纠错的编码，在通信系统中用的较多。可以查阅教材1.2.4节可靠性编码中的海明校验码复习相关内容，并考虑如何使用Quartus Prime软件设计电路

并用实验箱验证。

要考虑的问题可能会包含以下几个方面：

### 1. 输入输出信号怎么选取？

在编码时，输入信号为4位信息码：**B8B4B2B1**（可用实验箱上的4个拨位开关来输入）；输出信号为3位校验码**P3P2P1**，（4位信息码+3位校验码）构成7位编码输出（连接至实验箱上的7个LED灯显示结果）。

在解码时，输入信号为7位接收码（可能含有错误位）：**B8B4B2P3B1P2P1**（可用实验箱上的7个拨位开关来输入）；输出信号为校验后的4位信息码（连接至实验箱上的4个LED灯显示结果）。

### 2. 怎么用数电实验箱来展示？

编码时，用四位拨位开关作为4位信息码，7个LED灯显示编码后的7位信号**B8B4B2P3B1P2P1**；解码时，7个拨位开关对应7个码位，输出4个LED灯对应解码后的4个信息位。在只错一位码的情况下，解码后的4个LED灯的状态应该与编码时输入的4位拨位开关的状态一致。

如何验收？先做编码，再做解码。编码部分任意输入4位信息码（通过拨位开关设置），这时7个LED灯是编码后的码字。解码部分，解码时对照编码部分验证后的7个LED灯的状态输入到7位拨位开关上（注意要反转其中某一位，剩余6位不变），这时4个LED灯的输出状态应是纠错后的正确编码，这4个LED灯的状态应和编码部分输入所设置的4个输入拨位开关的状态一致。

思考：

- 1、所设计的海明校验码是否能纠正一位错？
- 2、所设计的海明校验码是否能纠正两位错？
- 3、当前设计的海明校验码为什么只能纠一位错？
- 4、如何设计出可以纠多位错的海明码？

## 题目二：超前进位加法器

在计算机中都是二进制数及其运算，最本质的运算单元是加法器。串行加法器比较好理解，但是高位的计算要等待低位的计算结果，这样会导致多位的二进制加法速度很慢，所以提出了并行加法器，可有效降低延迟提高运算速度。

可参照教材85页2.4.4章节中的超前进位加法器相关章节，设计4位二进制超前进位加法器，并在实验箱上验证。设计实现2个4位二进制数的加法器，并添加额外低位向本位的进位，使用9个拨位开关作为输入，5个LED灯作为输出。



通过观察输入输出的关系验证超前进位加法器的结果。

思考：

- 1、与行波加法器相比，超前进位加法器最明显优势在哪里？劣势在哪里？
- 2、思考超前电路解决问题的思路和方法

### 题目三：多路表决器

在生活中常见到需要评委为选手进行表决的场景，通过评委的表决来判断是否选手最终是否通过选举。评委通常给出的结果是不尽相同的，所以可以通过设计一个多路表决器电路，将评委的评定结果作为电路的输入，电路给出选手是否通过选举的结果。

本例中要求设计一个5路的表决器，具体功能要求如下：

- 1、有多位表决者（如5位），每位表决者可以选择支持或不支持，也可以弃权
- 2、如果有超过一半的表决者赞成，则结果为通过（绿灯亮，红灯灭），否则结果为不通过（绿灯灭，红灯亮）

通过参考数电教材中的译码器章节自己给出设计方案。在实验箱上验证时，将评委的结果作为输入用拨位开关表示，最终的评定结果由LED灯显示。

思考：

- 1、如何从需求中抽象出电路？
- 2、是否可以扩展为更多路的表决器？如果可以，以7位表决器为例，在现有的方案下，如何扩展？

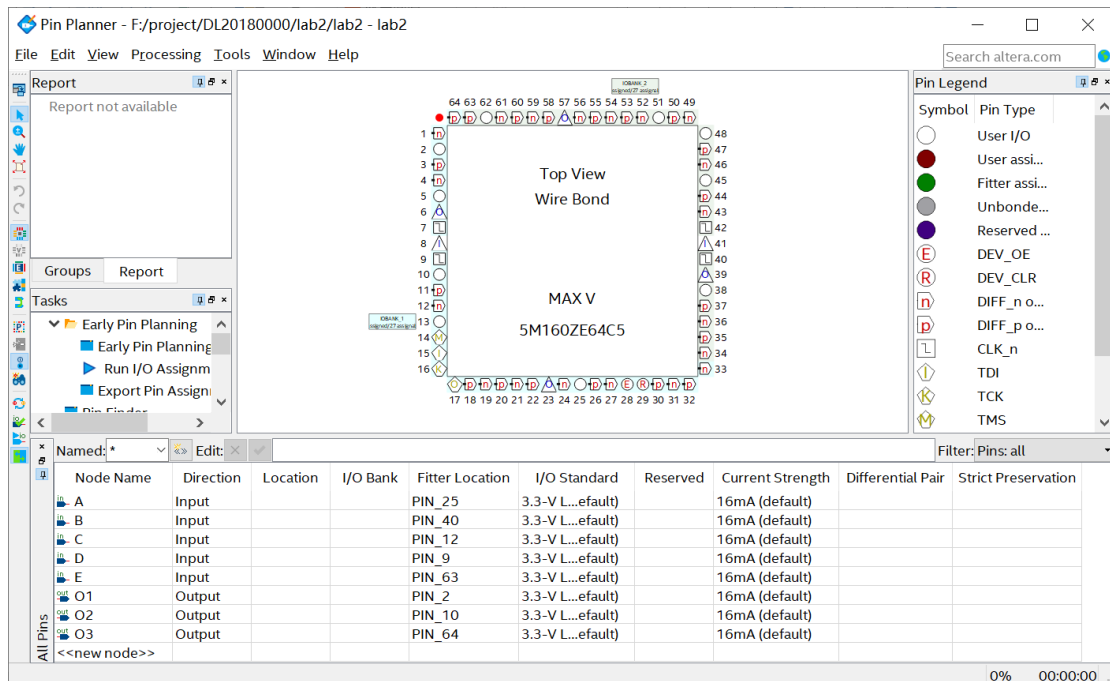
## 三 器件编程步骤

使用Quartus Prime18.1完成组合逻辑电路设计，对所设计的电路进行编译仿真，通过功能仿真，在逻辑上验证了所实现电路功能的正确性。

为了进一步验证电路运行过程中功能的正确性，就需要将所实现的功能电路下载到芯片中运行，在实际的硬件设备上验证。在硬件设备上验证时，需要为输入管脚提供实际的高、底电平作为信号输入，电路运行结束后产生的输出信号需要通过输出管脚连接外设以便指示输出电平的高、低。为了方便地为将实际电平信号接入管脚，同时将输出信号引出，就需要为输入输出管脚指定器件的引脚号，将电路的管脚与FPGA或CPLD芯片的管脚绑定，这一过程称作管脚分配或管脚锁定。

下面以最简单的与或非门电路为例描述从分配管脚到下载验证的全过程。

## 1、 选择Assignments->Pin Planner或者点击Pin Planner图标， 进入Pin Planner用户界面



Pin Planner 用户界面

## 2、拖拽All Pins窗口中的管脚至封装图对应的位置，或可以双击管脚所在行对应Location列的空白处，从出现的下拉菜单中选择对应的管脚编号。

将输入管脚分配至距拨位开关近的位置，输出管脚分配至距LED灯近的位置，由于实验箱上的芯片与其他电路结构的连接关系已经确定，所以实际分配管脚，应按照电路原理图的连接关系进行分配。

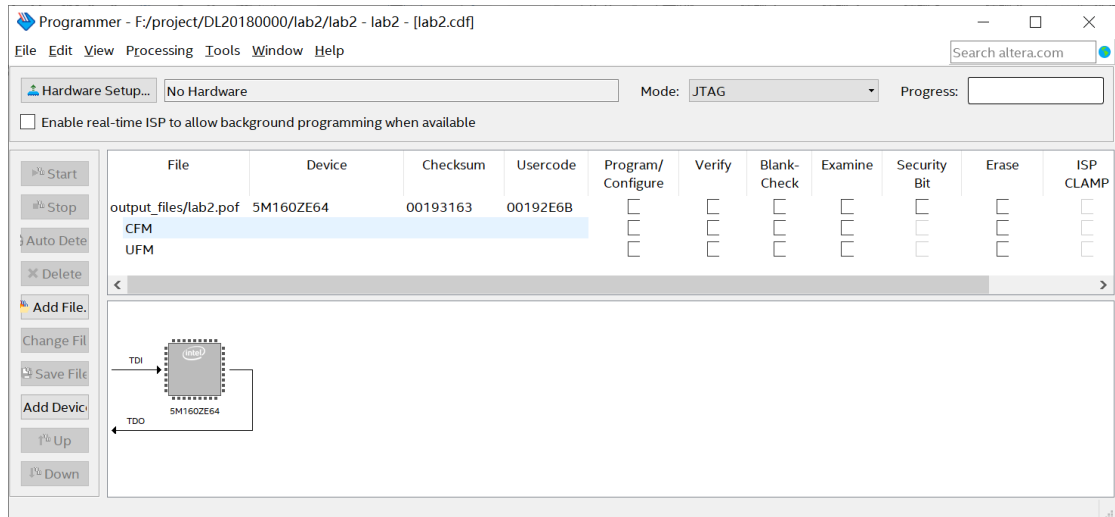
Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
A	Input	PIN_9	1	PIN_25	3.3-V LVTTTL (default)		16mA (default)
B	Input	PIN_11	1	PIN_40	3.3-V LVTTTL (default)		16mA (default)
C	Input	PIN_13	1	PIN_12	3.3-V LVTTTL (default)		16mA (default)
D	Input	PIN_19	1	PIN_9	3.3-V LVTTTL (default)		16mA (default)
E	Input	PIN_21	1	PIN_63	3.3-V LVTTTL (default)		16mA (default)
O1	Output	PIN_5	1	PIN_2	3.3-V LVTTTL (default)		16mA (default)
O2	Output	PIN_3	1	PIN_10	3.3-V LVTTTL (default)		16mA (default)
O3	Output	PIN_1	1	PIN_64	3.3-V LVTTTL (default)		16mA (default)

分配管脚后的All Pins界面

## 3、管脚分配完成后，对工程进行重新编译。

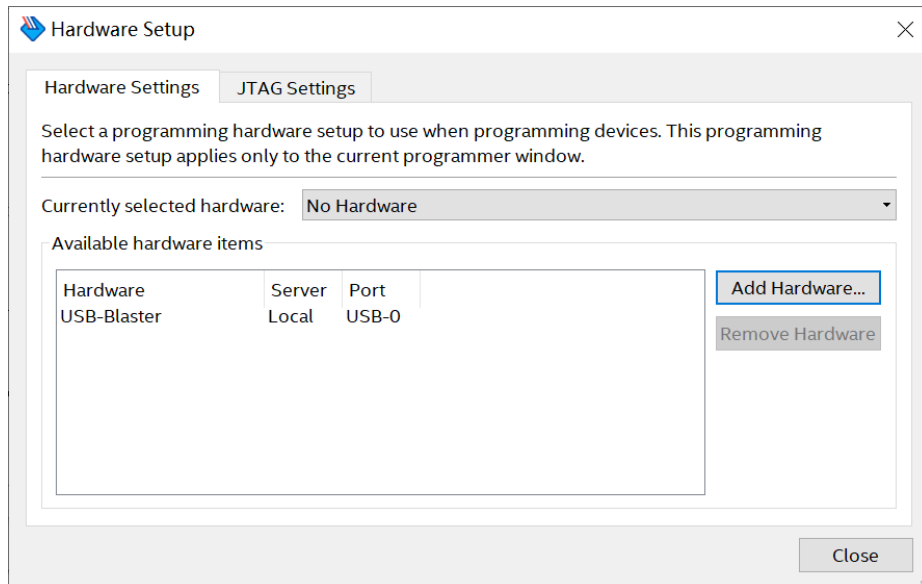
## 4、 点击 Programmer 图标，弹出 Programmer 界面。从界面左上角 Hardware Setup 右侧的文本框中若看到 No Hardware 字样，说明 Programmer 还未指定编程器。



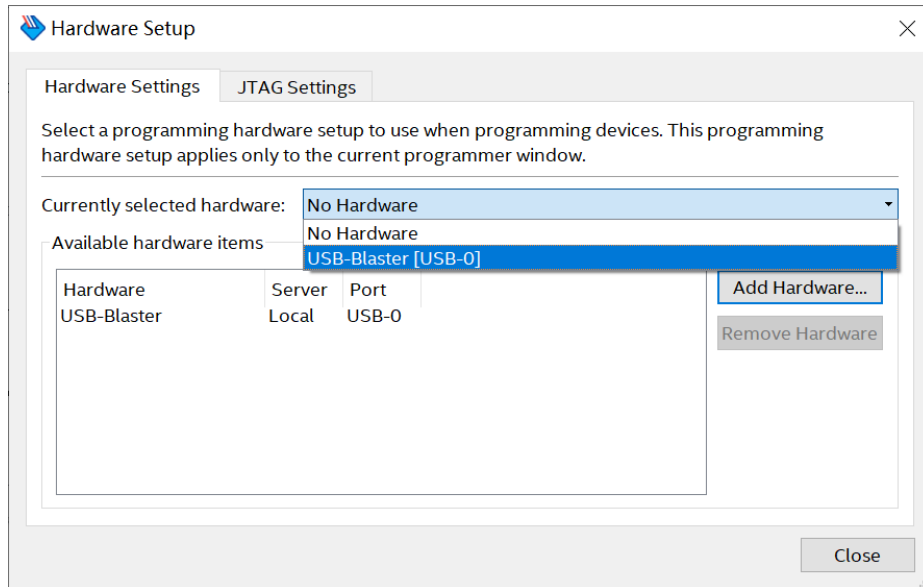


Programmer 界面

- 5、将编程器（本例为 USB Blaster）与计算机和目标板卡连接好，确保已经正确安装了 USB Blaster 的驱动程序后（若初次接上 USB Blaster 编程器，则需要为设备安装驱动程序，驱动程序位于 quartus 安装目录下的 drivers 文件夹下），单击 Hardware Setup 按钮，弹出 Hardware Setup 对话框，在 Currently selected hardware 右侧的下拉菜单中列出了当前可用的编程器，选择 USB-Blaster，然后点击 Close 关闭对话框，在 Programmer 界面中 Hardware Setup 按钮后将出现选择的编程器，此时硬件设置完成。

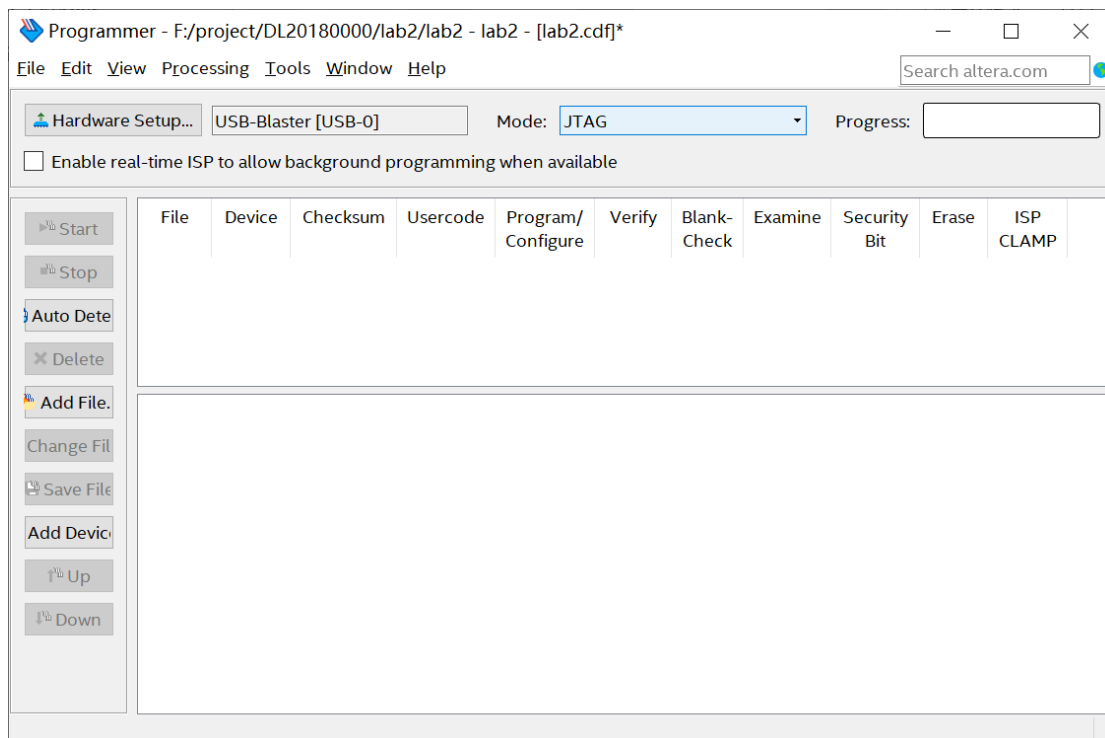


Hardware Setup 界面



选中编程器件 USB Blaster

- 6、硬件设置完成后，在 Mode 栏下选择对应的编程方式如 JTAG。选择待编程文件，单击左侧的 Add File 按钮添加待编程文件，注意：工程编译成功后将产生唯一名称与工程名相同的 .pof 文件用于下载。待编程文件添加完成后，勾选 Program/Configure 选项，点击左侧的 Start 按钮，开始下载，在 Progress 进度条中显示编程进度，编程完成后 Progress 的进度显示为 100% Successful，此时可以在目标板卡上进行程序验证。



未关联待下载文件的下载界面

## 四 实验报告

4.1 实验内容（描述所选题目）

4.2 实验步骤

4.3 实验结果

4.3.1 所设计的电路图；

4.3.2 软件仿真结果；

4.3.3 管脚分配情况；

4.3.4 下载验证结果；

4.4 思考题

## 实验 4：时序逻辑电路(1)—双稳态元件功能测试

### 一 实验准备

预习或复习数电教材上3.1节的双稳态元件部分的内容。掌握SR锁存器、D触发器、JK 触发器、 T触发器的电路结构和时序逻辑功能。

### 二 实验内容

1. 在Quartus Prime软件中实现S-R锁存器， /S-/R锁存器， JK触发器， D触发器及T触发器电路， 并下载到实验箱上进行测试验证。

分别测试S-R锁存、 /S-/R锁存器、 J-K触发器、 D触发器和T触发器的逻辑功能， 记录各自的逻辑功能， 记录各锁存器或触发器的次态真值表和仿真波形图， 借助实验更直观的理解如何从与或非逻辑门搭建巧妙的电路来实现数据锁存， 又如何从SR 锁存器一步步演变到D触发器、 JK 触发器以及T触发器的。

思考：

1.1、如何设置仿真波形的激励条件才能使波形仿真完备？

1.2、组合逻辑和时序逻辑中输入对输出的影响有什么差别？

2. 使用74系列芯片SN74HC00搭建/S-/R锁存器， 根据真值表测试/S-/R锁存器的功能， 思考/S-/R锁存器实现数据锁存的原因？

3. 使用74系列芯片SN74HC74测试D触发器的功能， 观察输入和输出之间的关系。根据测试结果回答：

a. 从时钟上升沿到Q输出经过多长时间？

b. 从时钟上升沿到/Q输出经过多长时间？

c. 从/CLR有效到Q输出0经过多长时间？， 到/Q输出1经过多长时间？

d. 有没有什么办法能验证一下不满足建立保持时间会怎样？

4. 使用SN74HC74完成/S-/R锁存器的设计， 测试/S-/R锁存器的功能， 思考/S-/R锁存器与D触发器的关系， /S-/R锁存器如何转变为D触发器？ D触发器又如何实现/S-/R锁存器？

### 三 实验报告

3.1 实验内容（描述所选题目）

3.2 实验步骤

### 3.3 实验结果

3.3.1 Quartus 18.1 中双稳态元件功能测试电路图;

3.3.2 Quartus 18.1 中双稳态元件功能仿真结果; 分析为什么得到该结果;

3.3.3 使用 74HC00 芯片实现/S-/R 锁存器的方法及测试结果;

3.3.4 74HC74 芯片结构及管脚功能;

3.3.5 使用 74HC74 测试并验证 D 触发器的原理、方法及结果;

3.3.6 使用 74HC74 测试并验证/S-/R 锁存器的原理、方法及结果;

### 3.4 思考题

# 实验 5：时序逻辑电路(2)—计数器的设计与应用

## 一 实验准备

1. 学习Verilog语法知识。可参考附录3.
2. 7段数码管显示原理
3. 学习同步计数器、异步计数器的工作原理

## 二 实验内容

### 2.1 Verilog 语法知识及使用 Verilog 实现简单电路

Verilog 语法请参考附录 3.

#### 2.1.1 带使能的二选一数控开关。

输入信号 a,b,sel,en, 输出信号 y;

当en为1, 同时sel为1时, 选择a路信号输出给y;

当en为1, 同时sel为0时, 选择b路信号输出给y。

当en为0时, y恒输出0。

```
always @(*) begin
    if(en == 1'b1) begin
        if(sel == 1'b1) y <= a;
        else
            y <= b;
        end
    else begin
        y <= 0;
    end
end
```

#### 2.1.2 数据分配器

输入信号为a,b,i,输出信号为y0,y1,y2,y3;

当ab为00时, i输出给y0,y1y2y3输出为0;

当ab为01时, i输出给y1,y0y2y3输出为0;

当ab为10时, i输出给y2,y0y1y3输出为0;

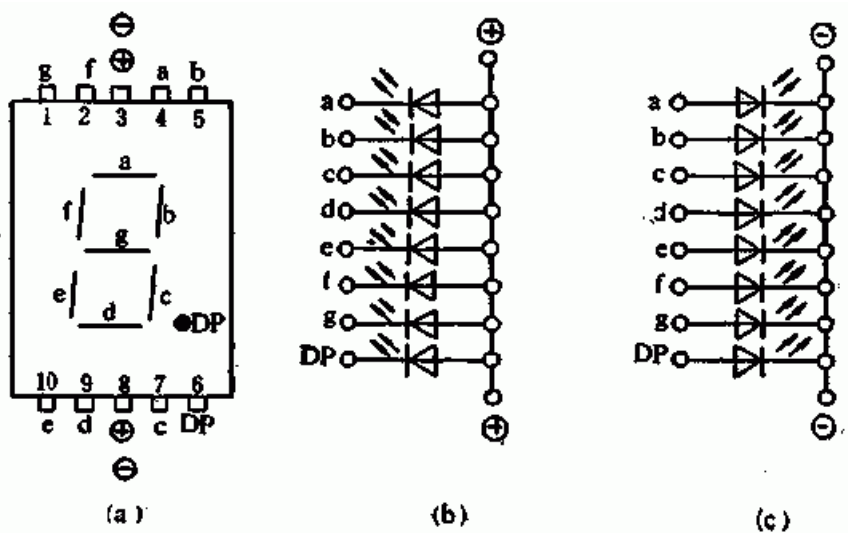
当ab为11时, i输出给y3,y0y1y2输出为0;

```
always @(*) begin
    case({a,b})
        2'b00: {y3,y2,y1,y0} <= {3'b000,i};
        2'b01: {y3,y2,y1,y0} <= {2'b00,i,1'b0};
        2'b10: {y3,y2,y1,y0} <= {1'b0,i,2'b00};
        2'b11: {y3,y2,y1,y0} <= {i,3'b000};
        default: ;
    endcase
end
```

## 2.2 七段数码管的工作原理及电路实现



七段数码管外观图



(a)管脚排列(俯视图) (b)共阳极结构 (c)共阴极结构

7段数码管由a b c d e f g dp八段发光二极管组成，靠每一段的亮灭状态的不同来显示不同的字符，当对应的段二极管上有电流流过时就会发光。8个段二极管的连接方式分为共阴极和共阳极两种，共阳极接法是指数码管的八段发光二极管的阳极(正极)都连在一起，而阴极对应的各段可分别控制，如图(b)所示，此时控制各段的信号为低时该段点亮；共阴极接法是指数码管的八段发光二极管的阴极(负极)都连在一起，而阳极对应的各段可分别控制，如图(c)所示，此时控制各段的信号为高时该段点亮。例：要让共阴极数码管显示0，需使 $abcdefg=7'b1111110$ ，而要让共阳极数码管显示0，则需使 $abcdefg=7'b0000001$ 。

理解了7段数码管的工作原理，在Quartus Prime18.1中使用Verilog语句或者原理图完成7段数码管显示程序的设计、仿真，并下载到开发板上使用实验箱上的数码管进行验证。完成从设计、编译、仿真、分配管脚、下载验证的全过程。

## 2.3 计数器的设计与实现

使用双稳态元件（如D触发器、JK触发器、T触发器）实现四位计数器设计（同步计数器、异步计数器皆可），并下载到实验箱上验证。

1) 根据数电教材上计数器章节，分析该部分所设计的电路是加1还是减1计

数器；

2) 将设计好的电路使用Quartus Prime18.1实现并进行功能和时序仿真，分析两种仿真结果的异同；

3) 使用7段数码管作为四位计数器的输出显示，同时请考虑计数器的时钟信号如何给，将设计完整的工程下载到实验箱上验证；

#### 思考如下问题：

1. 所设计的四位计数器是加1计数还是减1技术？如果是加1（减1）计数器，思考减1（加1）计数器如何实际？
2. 所设计的四位计数器是同步计数器还是异步计数器？思考同步计数器和异步计数器的区别？
3. 所用的双稳态元件是什么？如果用其他触发器替换，电路图又该如何设计？
4. 时钟输入用什么表示？

### 三 实验报告

#### 3.1 实验原理

#### 3.2 实验内容、步骤及结果

##### 3.2.1 实验内容 1：

Verilog 基本语法

实验设计思路

用 verilog 设计的步骤和仿真结果

##### 3.2.2 实验内容 2：

七段数码管的原理、真值表

实验设计思路

用 verilog 设计的关键代码

##### 3.2.3 实验内容 3：

所设计计数器的原理图

计数器的仿真结果

#### 3.3 思考题



## 实验 6：时序逻辑电路(3)—时序逻辑电路设计

### 一 实验准备

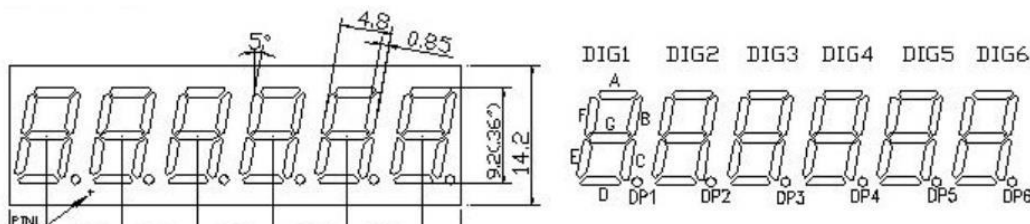
1. 复习数电教材中时序逻辑电路设计部分

### 二 实验内容

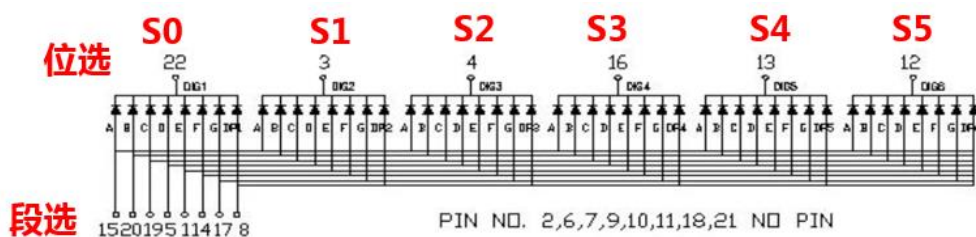
自选题目进行设计，使用Quartus Prime软件和实验箱进行电路设计并验证。

#### 参考题目一：扫描式数码管驱动

实验5中大家完成了7段数码管的驱动电路，验证时使用的是独立工作的数码管。在实验箱上，除了两个独立的共阳极数码管外，还有6个共阴极的数码管。这6个数码管采用的扫描式显示方式，使用14个信号驱动6个数码管，8个段选信号a b c d e f g dp和6个位选信号s1 s2 s3 s4 s5 s6。



扫描式数码管外观显示



扫描式数码管信号线的连接关系

如上所示，6个数码管的段选信号A B C D E F G接到一起，由DIG1、DIG2、DIG3、DIG4、DIG5、DIG6（实验箱上标记为S1 S2 S3 S4 S5 S6）位选信号控制每个管子的亮灭。当DIG1为低时，数码管1按照段选信号A B C D E F G的规则被点亮，DIG1为低的同时，DIG2、DIG3、DIG4、DIG5、DIG6必须为高，这样段选信号A B C D E F G即使也送到了其它数码管，但是因为DIG2、DIG3、DIG4、DIG5、DIG6为高，导致这5个数码管的二极管因为没有正向压降所以没有电流流过就不会被点亮。利用人眼的视觉暂留特点，可以使6个管子轮流点亮，也就是使DIG1、DIG2、DIG3、DIG4、DIG5、DIG6轮流为低，同时段选信号A B C D E F G根据DIG的变化同步更新，就可以让6个管子看起来同

时是亮的了，并且各自显示各自的数字。

本次实验中，可以先设计6个数码管显示相同的数字，然后再设计数码管显示不同的数字。自己查阅资料，完成设计。

### 参考题目二：串并转换

实际工程中，数据收发是经常会碰到的，数据的收发和存储过程中，为了数据速率的匹配，经常会用到串并转换。例如USB接口转换芯片FT245，就是将USB的串行信号转换为8位总线的并行接口。串转并可以降低并口边的时钟速率。本实验中，大家可以选取实际中的应用，将数据从串转到并、或者从并转到串，思考转换过程中的数据速率如何做到匹配。

### 参考题目三：双钮电子锁

在日常生活中，电子锁已经越来越普遍。双钮电子锁是一种简易的密码锁。双钮电子锁的工作原理如下：

双钮电子锁系统中有A、B两个按键，A为密码输入键，A被按压的次数为密码的值；B键为确认键同时也是密码位数的统计按键。如预设密码“1536”，在开门时，先按一次A，再按下B则密码第一位输入完成，代表输入的为1；输入第二位密码时连续按五次A键，再次按下B键则密码第二位输入完成，代表输入值为5；以此类推当B键按压四次后代表四位密码完全输入完成，系统将开始判断输入的四位密码是否与预设的密码相同，如果两者相同则表示开门的LED灯被点亮，表示密码输入正确-开门；若两者不相同则表示密码输入错误的LED灯被点亮，同时电路发出报警声。

根据双钮电子锁的功能描述，分析双钮电子锁功能的具体实现，使用Quartus Prime软件完成电路设计，并进行仿真验证。仿真无误后，下载到实验箱上进行验证。

**思考：**双钮电子锁中的按键用开关表示，开关的抖动是否会对密码设置和开锁造成影响？如果是，如何解决？

## 三 实验报告

### 3.1 实验内容（所选题目描述）

### 3.2 实验过程

#### 3.2.1 建模、分析和具体的电路设计思路；

#### 3.2.2 具体的电路实现和仿真结果

#### 3.2.3 在实验箱上的验证方法和验证结果

### 3.3 思考题

### 3.4 对本次设计性实验的思考

# 实验 7-实验 8：数字系统设计

## 一 实验准备

复习数电教材上的理论内容，特别是数字系统设计章节。

## 二 实验内容

自主选题自主设计自主实现

### 参考题目一：数字钟（可参考教材257-260页）

在前面的实验内容中，大家已经掌握了任意模值计数器的设计、动态扫描数码管的驱动电路设计。数字钟是由两个模60的计数器（表示分钟和秒）、一个模24的计数器（表示小时）和数码管显示组成。模60的计数器和模24的计数器要考虑分别使用两个数码管显示，一个显示个位、一个显示十位，所以设计上要注意这个模60的计数器其实是由一个模10的计数器和模6的计数器组成的，这里要注意置位信号的设计。同样的，模24的计数器是由一个模10的计数器和一个模3的计数器组成，同时这个模24的计数器要在23的时候回到0重新计数，这里是一个难点，大家要注意。

具体实现步骤：

1. 设计一个由模10和模6组成的模60的计数器。在Quartus Prime中实现并仿真。注意，一定要对照仿真结果来帮助自己检查设计。如果仿真结果不对，根据不对的地方一步步往前推，进而分析电路设计不对的地方，改正后重新仿真。如此反复直到设计满足原本的设计目标。
2. 按照上面所述，设计一个模24的计数器。自己设计、自己验证、自己查错。
3. 设计动态扫描数码管驱动电路，注意是6个数码管显示独立的数字。
4. 根据自底向上的设计方法，将以上3个设计封装成符号文件，然后新建一个顶层bdf文件，调用2个模60的计数器和一个模24的计数器，构建数字钟，再加上动态扫描数码管驱动电路，作为显示。
5. 整个工程编译后，分配管脚再编译，最后下载到板子上验证。

### 参考题目二：交通灯（可参考教材211页）

交通红绿灯是人们出行必会遇到的。交通灯设计的好坏会直接影响通行效率。在西安市的不少路口，红绿灯就设计的差强人意。如果让你来设计一个交通控制灯，你的策略是怎样的了？你能否将你认为的好的策略使用实验室的环

境搭建出来？

数电课堂上交通灯是一个非常生动的例题。大家可以从这个例题着手，开动脑筋争取设计出更好的电路。实验箱上的拨位开关可以作为有行人或者有车辆的输入，led灯可以作为南北向/东西向的黄/红/绿灯。

### 参考题目三：串口通信

该内容是运用数字电路的知识来实现与计算机的串口通信，在实际工程 and 生活中应用的较多。如：打印机正是使用的串口与计算机进行通信的。“通信”是信息工程专业接触比较多的一个知识面，从并行到串行、同步到异步、从有线到无线、低频到射频、信源编解码、信道编解码等涵盖的知识面非常广。在实际的工程中，经常也需要电路系统与PC机进行信息交互以完成PC机对下位机的配置或者PC机对于下位机的数据读取，而串口通信是一种简便易行的可选方案之一。

串口通信的相关知识：

#### 2.3.1 系统结构



#### 2.3.2 UART串口通信原理

通用异步收发传输器(Universal Asynchronous Receiver/Transmitter)，通常称作UART，是一种异步收发传输器。将数据由串行通信与并行通信间作传输转，作为并行输入成为串行输出的芯片UART 是一种通用串行数据总线，用于异步通信。该总线双向通信，可以实现全双工传输和接收。

##### 1) UART 通信协议

UART 作为异步串口通信协议的一种，工作原理是将传输数据的每个字符一位接一位地传输。

其中每一位(Bit)的意义如下：

起始位：先发出一个逻辑“0”的信号，表示传输字符的开始。

数据位：紧接着起始位之后。数据位的个数可以是4、5、6、7、8等，构成

一个字符。通常采用ASCII 码。从最低位开始传送，靠时钟定位。

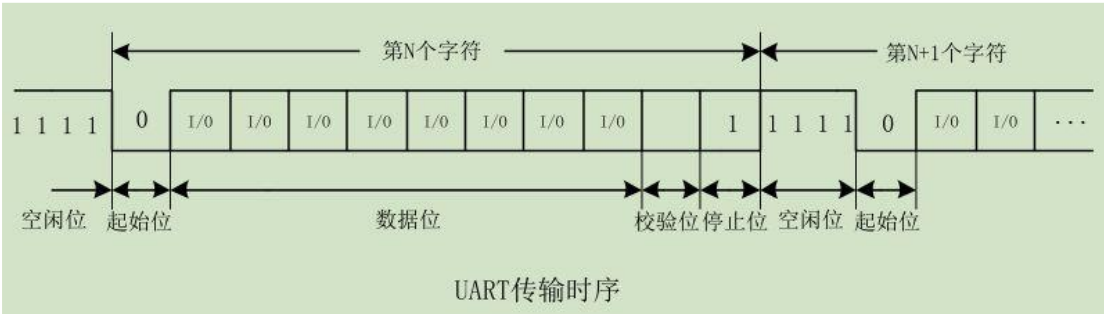
**奇偶校验位：**数据位加上这一位后，使得“1”的位数应为偶数(偶校验)或奇数(奇校验)，以此来校验数据传送的正确性。

**停止位：**它是一个字符数据的结束标志。可以是1位、1.5位、2位的高电平。

由于数据是在传输线上定时的，并且每一个设备有其自己的时钟，很可能在通信中两台设备间出现了小小的不同步。因此停止位不仅仅是表示传输的结束，并且提供计算机校正时钟同步的机会。适用于停止位的位数越多，不同时钟同步的容忍程度越大，但是数据传输率同时也越慢。

**空闲位：**处于逻辑“1”状态，表示当前线路上没有数据传送。

UART 协议传输时序如下图所示：



## 2) UART工作原理

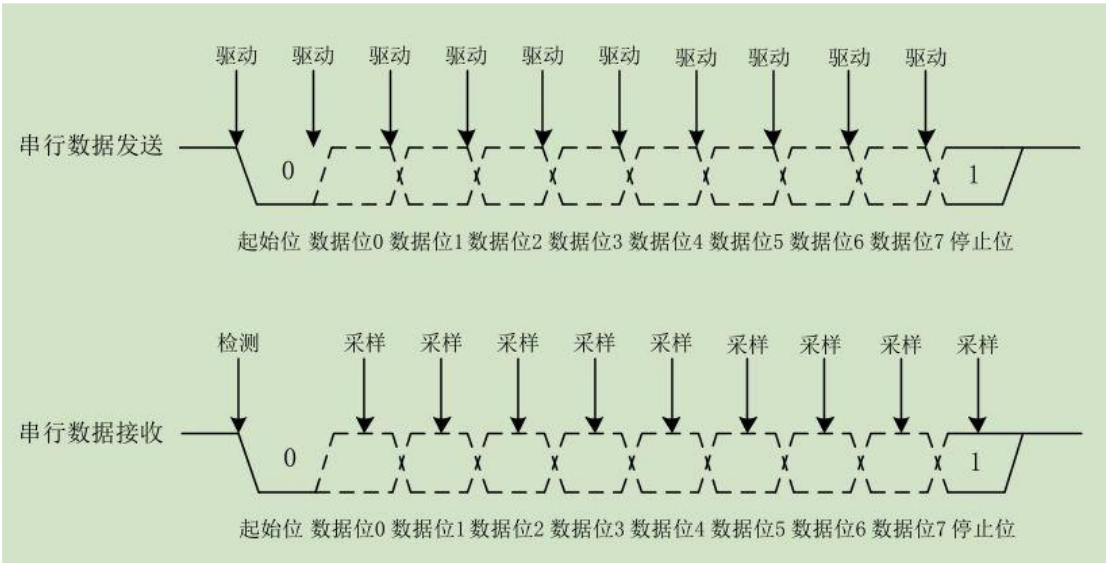
**发送数据过程：**空闲状态，线路处于高电位；当收到发送数据指令后，拉低线路一个数据位的时间T，接着数据按低位到高位依次发送，数据发送完毕后，接着发送奇偶校验位和停止位（停止位为高电位），一帧数据发送结束。

**接收数据过程：**空闲状态，线路处于高电位；当检测到线路的下降沿（线路电位由高电位变为低电位）时说明线路有数据传输，按照约定的波特率从低位到高位接收数据，数据接收完毕后，接着接收并比较奇偶校验位是否正确，如果正确则通知后续设备准备接收数据或存入缓存。

由于UART 是异步传输，没有传输同步时钟。为了能保证数据传输的正确性，UART 采用16 倍数据波特率的时钟进行采样。每个数据有16 个时钟采样，取中间的采样值，以保证采样不会滑码或误码。一般UART 一帧的数据位数为8，这样即使每个数据有一个时钟的误差，接收端也能正确地采样到数据。

**UART 的接收数据时序为：**当检测到数据的下降沿时，表明线路上有数据进行传输，这时计数器CNT 开始计数，当计数器为24=16+8 时，采样的值为第0位数据；当计数器的值为40 时，采样的值为第1 位数据，依此类推，进行后面6 个数据的采样。如果需要进行奇偶校验，则当计数器的值为152 时，采样的值即为奇偶位；当计数器的值为168 时，采样的值为“1”表示停止位，一帧数据接收完成。

一个标准的10位异步串行通信协议(包含1个起始位、1个停止位和8个数据位)收发时序,如图2所示



10位标准串行通信协议收发时序图

3) UART波特率发生器

波特率是衡量数据传输速率的指标,表示每秒传送数据的字符数,单位为Baud。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地采样,以提取出异步的串行数据。根据给定的晶振时钟和要求的波特率,可以算出波特率分频计数值。

2.3.3 具体实现步骤

- 1. 查阅计算机串口通信的相关知识,包括信号名称和信号时序波形图;
- 2. 编写适配于计算机串口通信的CPLD程序;
- 3. 在核心板上下载该程序并测试,完成与计算机的串口通信。

注意:

1) 实现串并转换时,需根据MAX V CPLD核心板的原理图上的USB转串口芯片来选择PC机上的USB串口驱动程序,可以去CH340官网下载该驱动;自行下载并查阅CH340的器件手册。

2) 搭建测试环境并完成测试。测试时需要借助串口调试助手,在PC机上安装串口调试助手,用USB连接线连接核心板和PC机,打开串口调试助手测试。测试过程中可以通过示波器或者实验箱上的LED灯的亮灭观察核心板上主要信号的波形,一来可以排查错误,二来可以通过观测实际波形同时对照串口通信的时序图以加深对串口通信的理解。若通信过程中出现错误,根据现象分析可能出错的地方并排查和纠错,直到问题解决。

参考题目四: 自动售货机控制器设计



自动售货机在日常生活中非常常见，本实验中通过数字电路系统设计自动售货机的主控制器，模拟自动售货机的工作过程。

#### 2.4.1 自动售货机控制器的工作原理：

与实际生活中使用的自动售货机控制器相对应，自动售货机控制器应包括用户操作功能和系统维护功能。

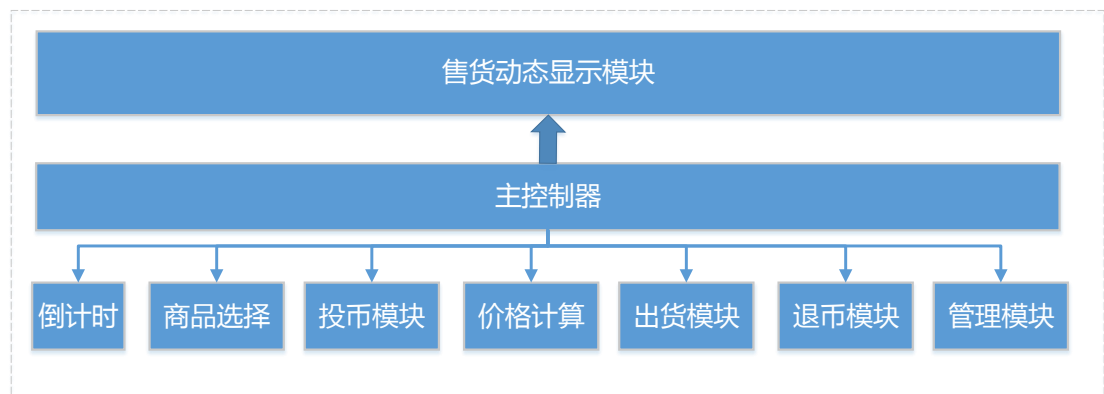
1) 用户操作：用户可以选择商品种类、选择商品数量；用户投币购买，输入投币金额、投币次数；投币确认后比较投币金额是否足以支付商品价值，确定是否出货、退币。

2) 系统维护功能是指维护人员可以根据情况按需调整商品售价。

3) 显示功能：在用户操作过程中应全程动态显示自动售货过程中的重要信息；在系统维护过程中，应显示系统的维护信息，如商品调价。

可以采用自顶向下或自底向上的方法进行设计。设计时，若采用自顶向下则先完成主控制器设计，然后由主控制器调用各功能模块实现整体的控制器设计。若采用自底向上，则先完成各功能模块，在售货逻辑下分别实现各功能模块的衔接。

若采用自顶向下的方法，自动售货机控制器框图如下：



##### 商品选择模块的功能：

购物重置信号有效，模块清零重置

购物重置信号无效，商品选择使能无效，购物选择按键无效，模块不工作

购物重置信号无效，商品选择使能有效，在时间触发下，系统根据按键输入信号确定商品种类、商品数量，实时输出商品价格。在规定时间内按下确认购买按键后，确认商品种类、商品数量、输出商品最终价格，否则模块清零重置，重新选择商品。

##### 投币管理模块的功能：

投币重置信号有效，投币完成

投币重置信号无效，投币使能无效，投币按键开关无效，无法投币

投币重置信号无效，投币使能有效时，在时间触发下根据投币的面值和投

币的次数计算投币的金额，并实时显示投币金额。在规定时间内按下投币确认键，确认投币金额，如果没有按下投币确认键，在投币规定时间到后系统会自动确认。

**价格计算模块功能：**

对选择的商品总价和投币金额进行比较，以确定后续是否出货、是否退币

**出货模块功能：**

根据价格计算结果，完成商品出货模块设计

**退币模块功能：**

根据价格计算结果，完成退币模块设计

**管理模块功能：**

管理面自动售货机控制器的功能主要是系统维护人员根据需要对商品价格进行设定

**2.4.2 实现步骤**

- 1、分析自动售货机控制器工作原理，确定实现方案
- 2、使用Quartus Prime软件依次实现各功能模块和主控制器模块（如果采用自顶向下的方法）
- 3、将各功能模块连接成为完整的自动售货机控制器系统
- 4、仿真
- 5、下载验证

### **三 实验报告**

**3.1 实验内容（所选题目描述）**

**3.2 实验过程**

3.2.1 建模、分析和具体的电路设计思路；

3.2.2 具体的电路实现和仿真结果

3.2.3 在实验箱上的验证方法和验证结果

**3.3 实验总结**

**3.4 对课程的意见和建议**



# 附录 1：Quartus Prime18.1 软件设计流程

## 一、 打开软件

在开始菜单中，双击 Quartus Prime 18.1 打开软件（如图 2- 1 所示）或者双击桌面上的 Quartus Prime18.1 的图标打开软件。软件界面如图 2- 2 所示。

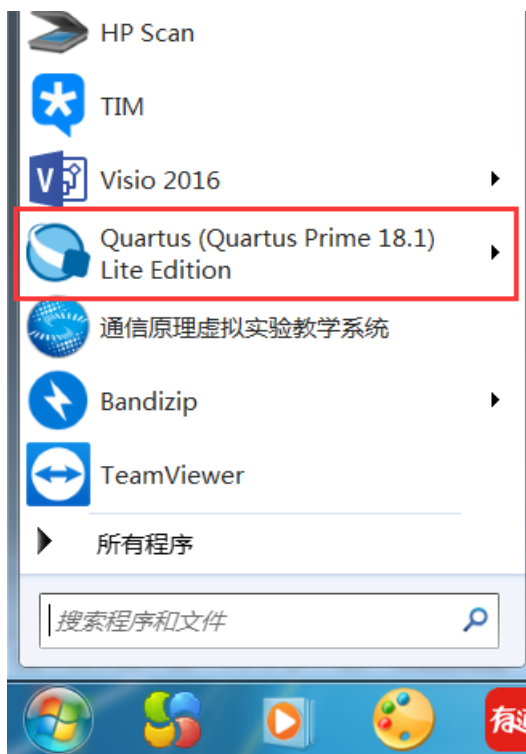


图 2- 1 开始菜单

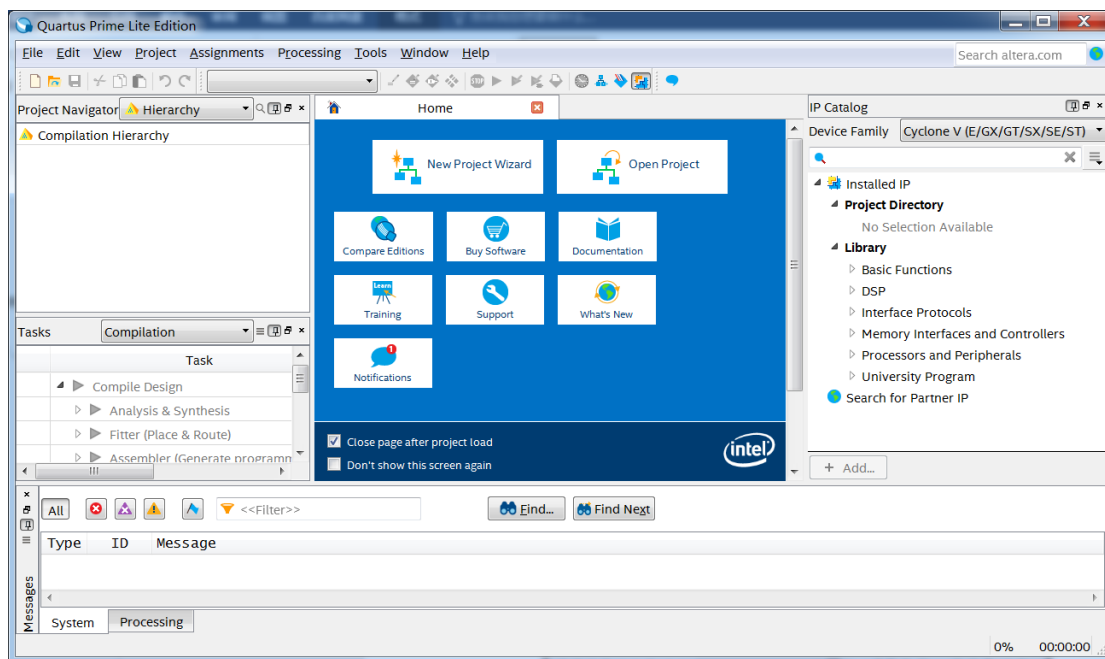


图 2- 2: Quartus Prime 18.1 初始界面

关掉 Home 窗口后，Quartus Prime 的工程工作区显示在界面上，Quartus Prime 界面上的各个工作窗口如图 2- 3 所示。

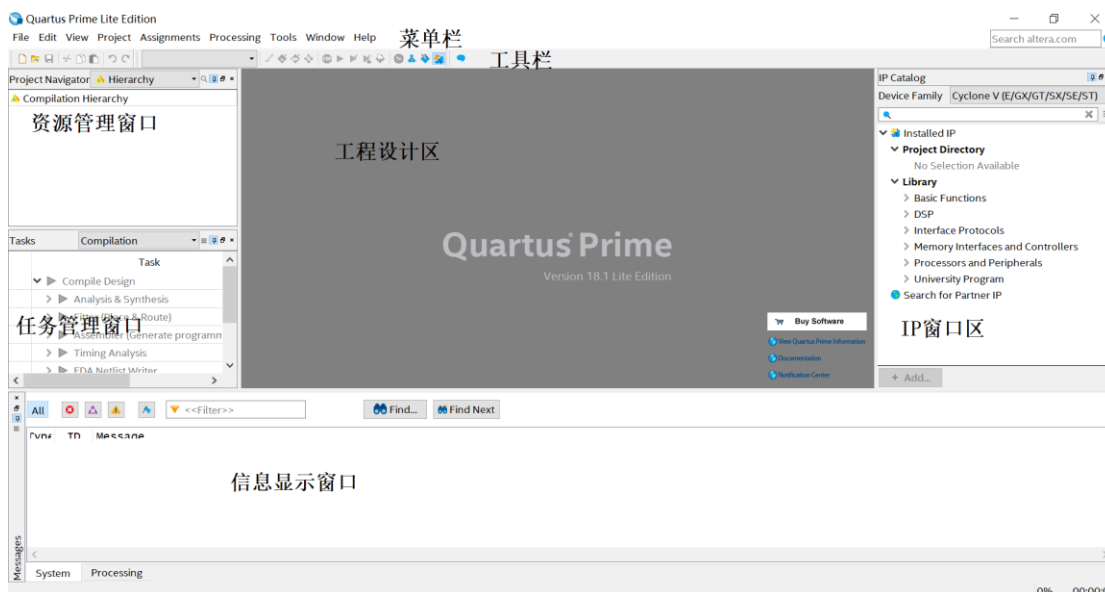


图 2- 3 Quartus Prime 软件窗口介绍

## 二、创建新工程

- 1) 选择 File 菜单下 “New Project Wizard..”, 创建新工程, 如图 2- 4 所示。

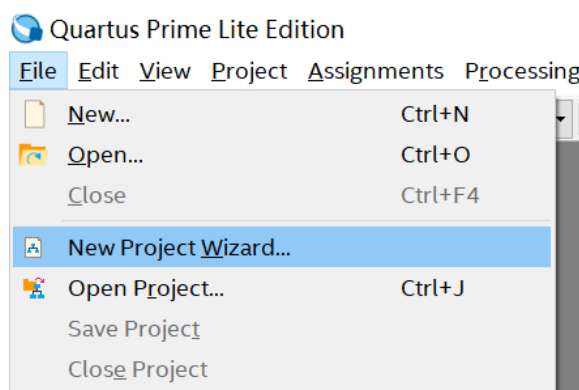


图 2- 4 创建新工程 (1)

- 2) 查看 Introduction 内容, 点击 next

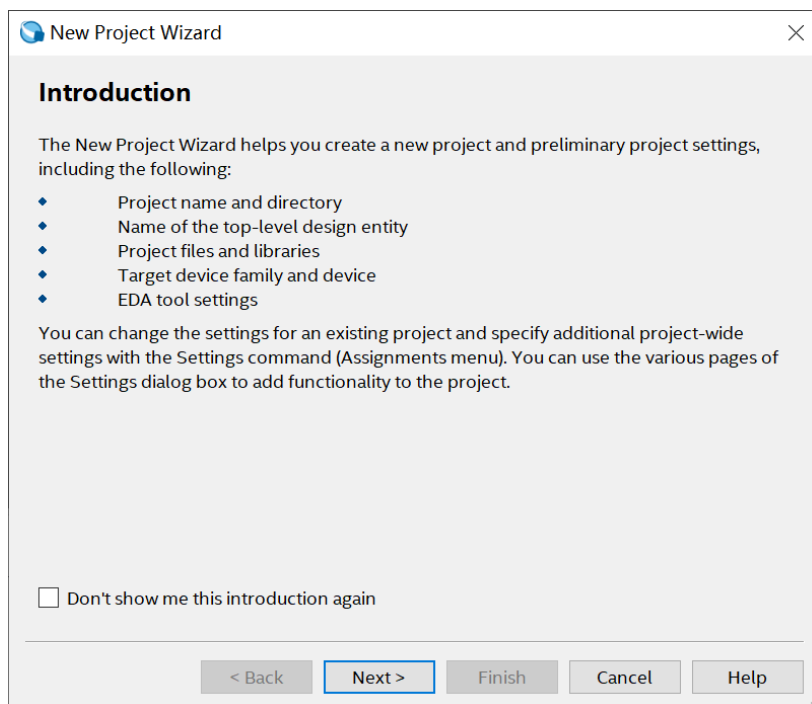


图 2- 5 创建新工程 (2)

- 3) 在出现的工程路径和工程名称设置话框中输入工程存放路径、工程名和顶层文件名, 工程存放路径建议不要放在安装路径下, 便于工程文件与安装文件区别开, 另外也有助于工程的统一管理, 命名的工程名应便于区分, 如存放路径为 F:/project/DL20180000/lab2, 其中 DL 为数字逻辑英文 digital logic 的缩写, 20180000 为学生学号, 以后该同学的工程都存放在 DL20180000 中, lab2 是本次实验的工程目录。注意: 工程名根据个人喜好设置, 需以字母开头, 中间无特殊符号。然后点击 Next。

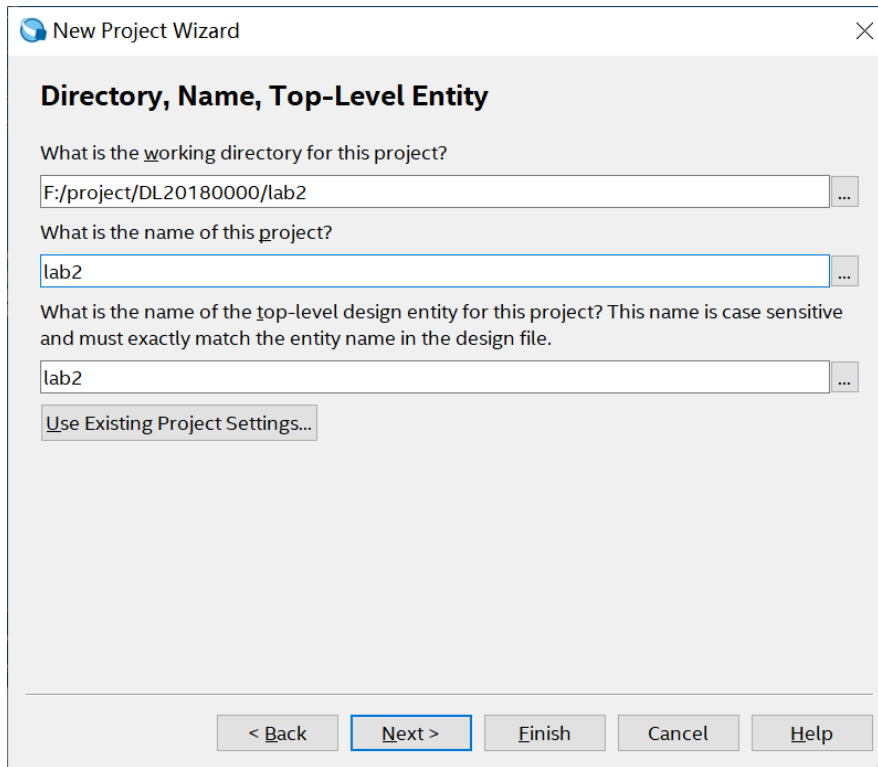


图 2- 6 创建新工程（3）

4) 第一次新建工程时因为没有工程模板，所以选择工程类型为 “Empty Type”，随后会弹出指定工程文件、目标器件、EDA 工具等的选项。如果有合适自己的工程模板，则可以在此步选择 “Project template”。

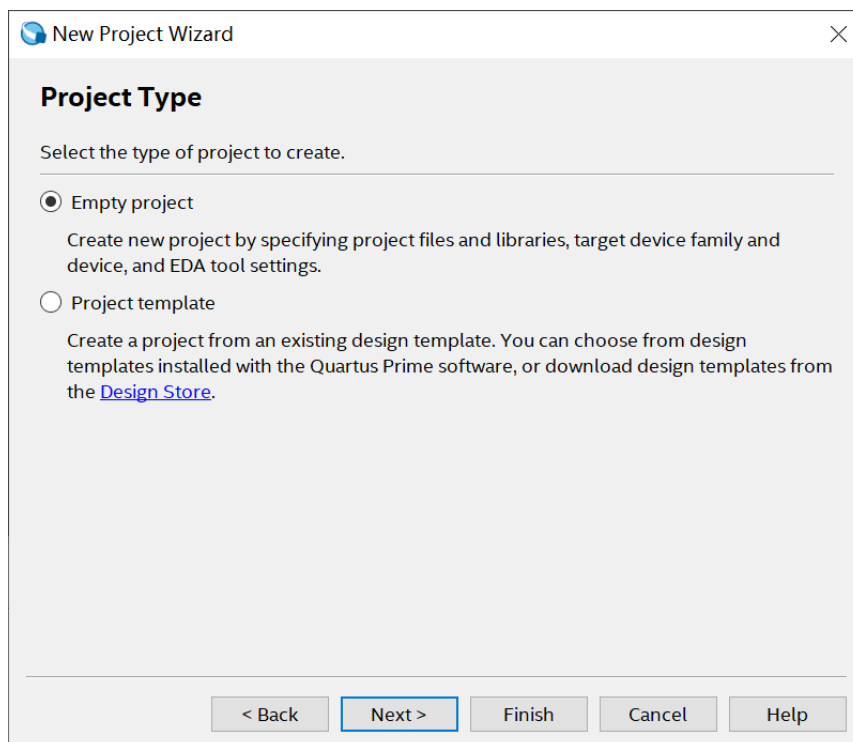


图 2- 7 创建新工程（4）

5) 在弹出的对话框中将已经设计好的文件添加到工程中，这些文件可以为 Verilog、VHDL 源文件、利用 Quartus Prime 设计出的图形设计文件或第三方综合工具综合后的网表文件，然后点击 Next。如本次暂时无需添加文件，则直接点击 Next。

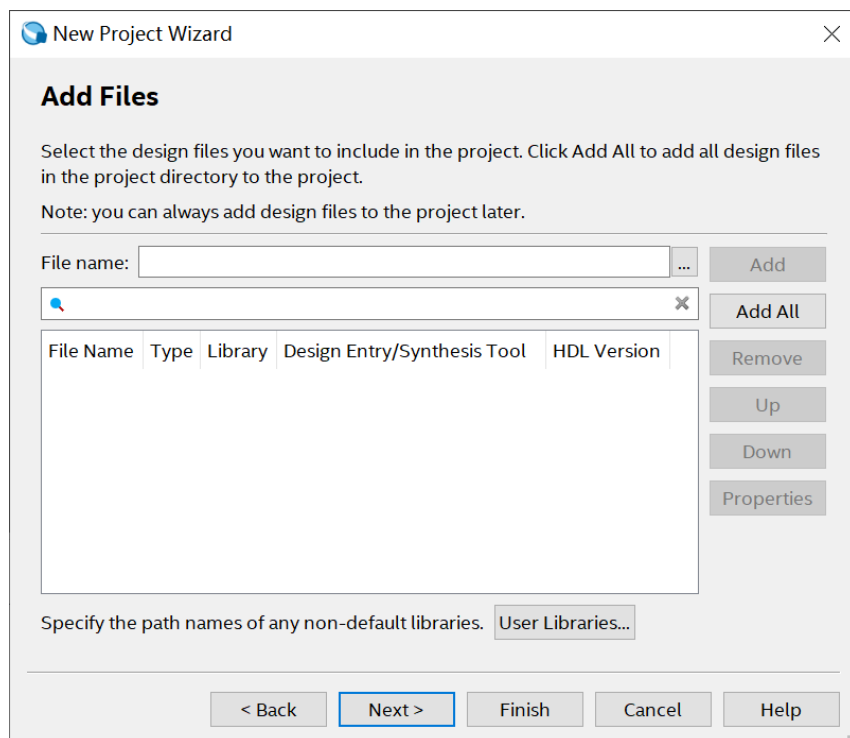


图 2- 8 创建新工程（5）

6) 在弹出的对话框中指定目标芯片。在 Family 下拉列表框中选择器件系列，相应地在 Available devices 列表框中会列出该系列器件的型号。如果非常清楚器件的特征，如封装、引脚数、速度等级等，可以从右侧 Package 处选择封装，从 Pin count 处选择引脚数，从 Speed grade 处选择速度等级从而快速定位所需要的器件型号。如图示中选择的 Family 系列为 MAX V，芯片型号为 5M160ZE64C5。这个型号表示该芯片属于 Intel FPGA 的 MAX V 器件系列，具有 160 个逻辑单元，封装为扁平封装，有 64 个管脚，温度等级为商业档，速度等级为 5。请读者注意的是，Intel FPGA 的芯片速度等级数字越大延迟越大，Xilinx 的芯片速度等级数字越大延迟越小。点击 Next。

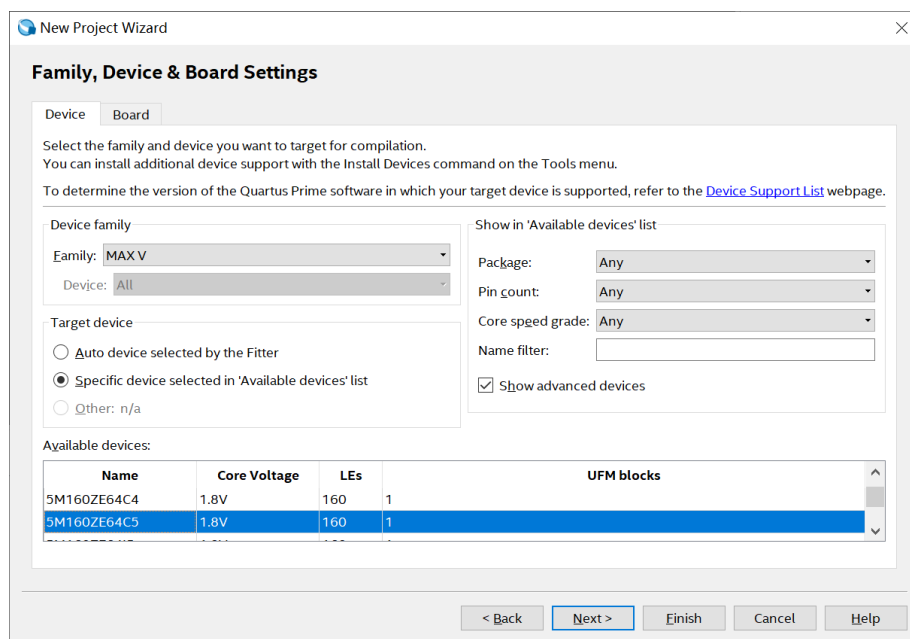


图 2- 9 创建新工程（6）

7) 在弹出的 EDA 工具设置界面中，可以指定第三方 EDA 综合、仿真与时序分析工具，在 Design Entry/Synthesis 后指定第三方综合工具，在 Simulation 后指定仿真工具。如选择使用 Quartus Prime 自带的综合、仿真与时序分析工具，则选择 ModelSim-Altera，点击 Next。

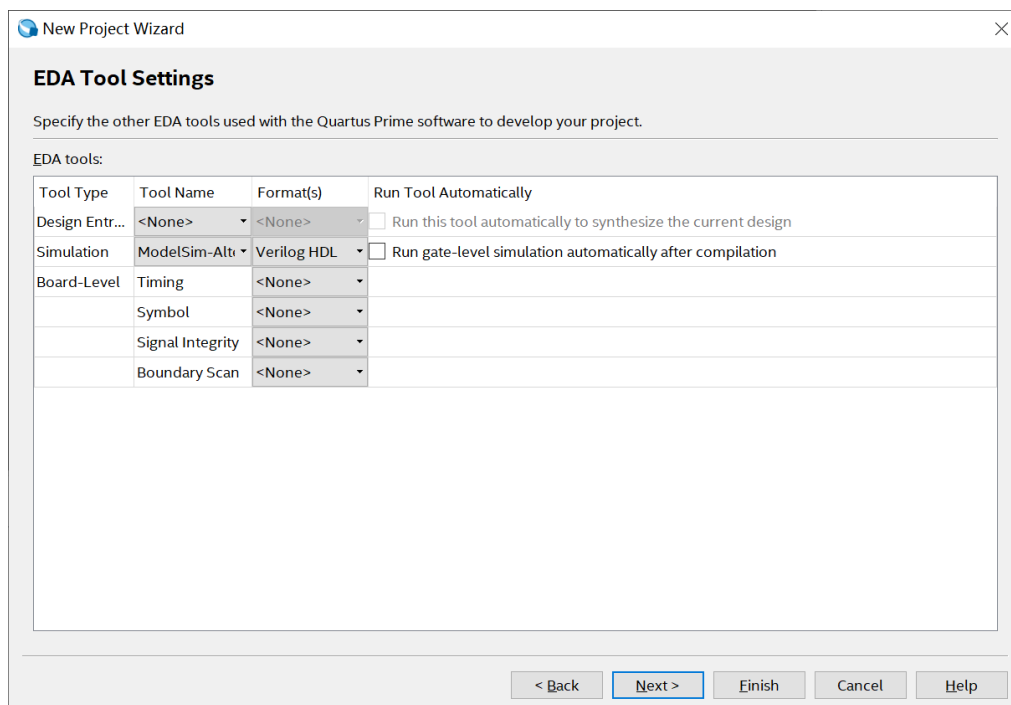


图 2- 10 创建新工程（7）

8) 在弹出的对话框中列出了工程设置信息，依次为工程路径、工程名、顶层实体名、加入文件数、添加的库数、选择的器件信息、EDA 工具信息及器件操作条件信息，点击 Finish 完成工程创建。

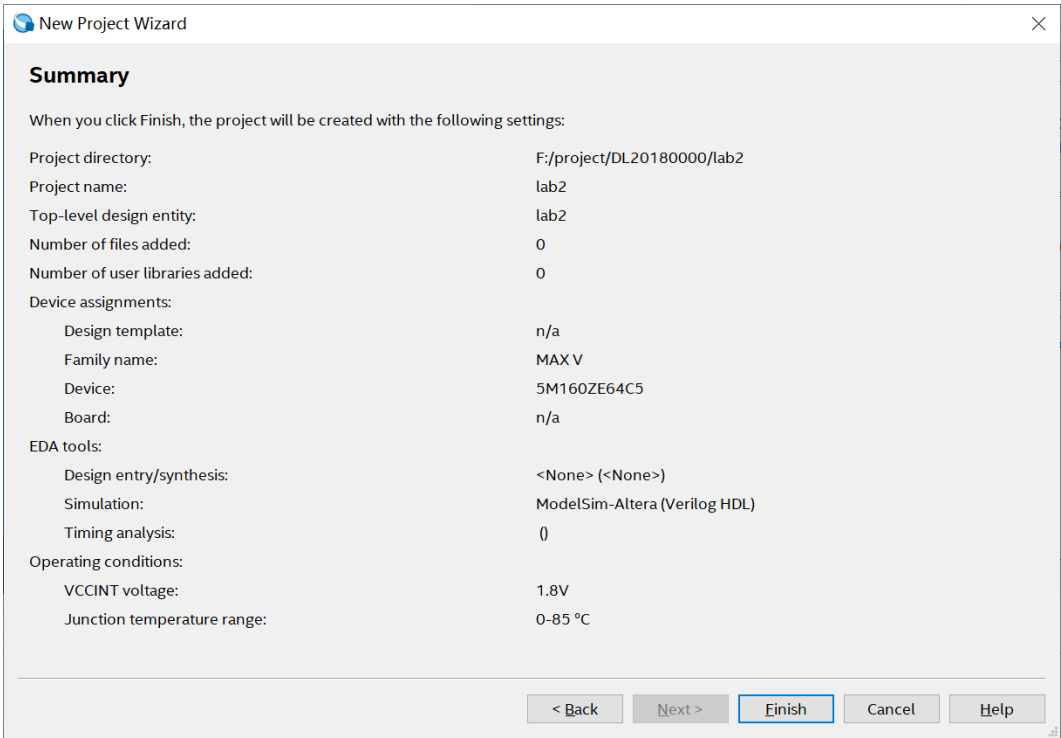


图 2- 11 创建新工程（8）

9) 工程创建后 Quartus Prime 主界面如图 2- 12 所示，芯片型号显示在工程导航栏（Project Navigator）里分层结构（Hierarchy）处，顶层实体文件名称 lab2 显示在器件名称下方。

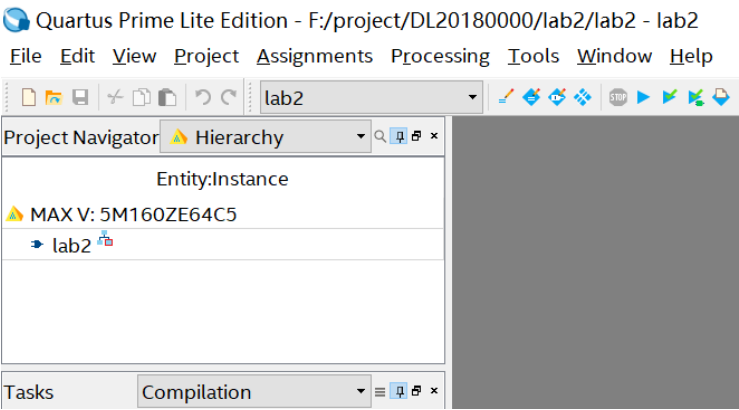


图 2- 12 创建新工程（9）

10) 工程创建后，工程存放路径 F:\project\DL20180000\lab2 下有创建工程产生的文件，包括一个名为 db 的文件夹和两个文件，其中 lab2.qpf 是工程文件，它以建立工程时设置的工程名 lab2 为名（如图 2- 13 所示）、以 qpf 为后缀。一个工程对应一个 qpf 文件，如果在操作过程中误关了工程，可以进入工程存放路径双击 qpf 文件重新打开该工程。qsf 是工程设置文件。此时的工程是一个空工程，即没有任何设计文件。



图 2- 13 创建新工程（10）

### 三、设计输入

新建工程完成后，接下来就可以进行电路设计了。在 Quartus Prime 中提供了多种设计输入的方式，如：原理图、硬件描述语言、IP 核调用等，本例中以原理图为例进行介绍。原理图设计输入是一种传统的设计方法，以符号引入方式将需要的逻辑函数引入。

选择 File->New..或者点击工具栏中的 New 快捷图标或者使用快捷键 Ctrl+N 开始新建输入文件，如图 2- 14 所示。

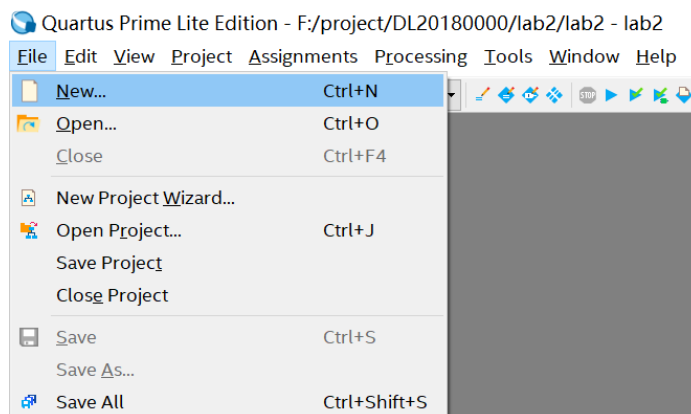


图 2- 14 建立输入入口

在新建对话框中选择 Design Files 下的 “Block Diagram/Schematic File”，如图 2- 15 所示。



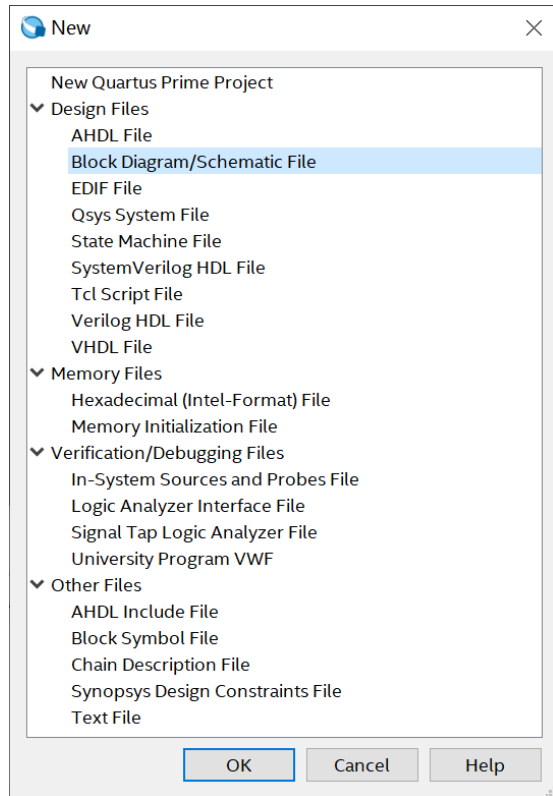


图 2-15 新建对话框

选中“Block Diagram/Schematic File”后点击“OK”按钮，出现图 2-16 的原理图编辑窗口。图形编辑窗口上侧为绘图工具栏。

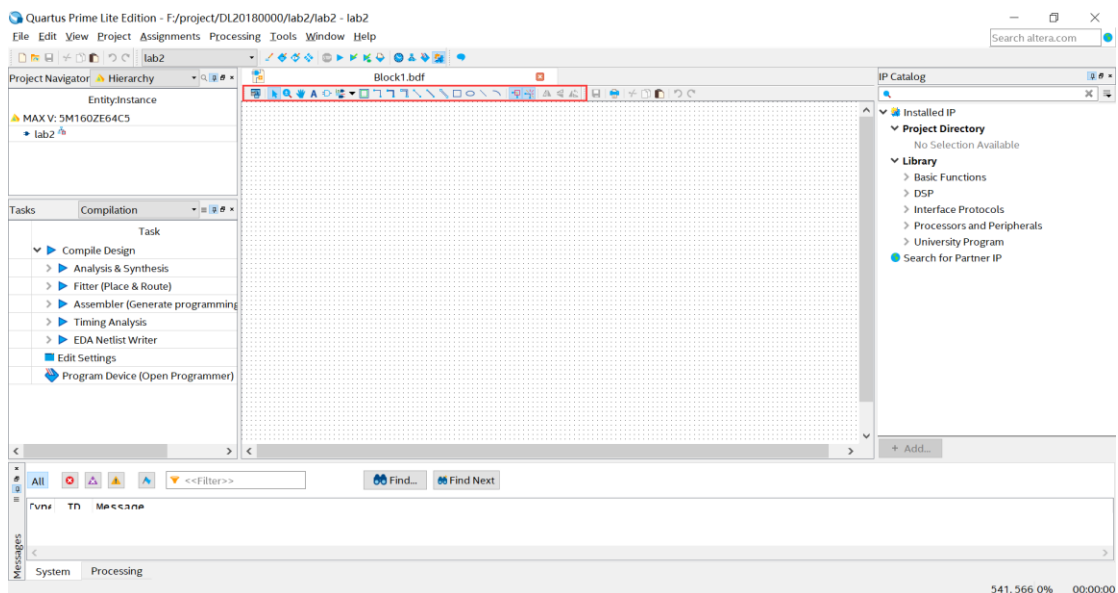



图 2-16 原理图编辑窗口

进行原理图绘制时首先要添加原理图符号，点击工具栏按钮中的  Symbol Tool 图标或者在原理图编辑窗口中双击鼠标左键，打开 Symbol 对话框，如图 2-17 所示。

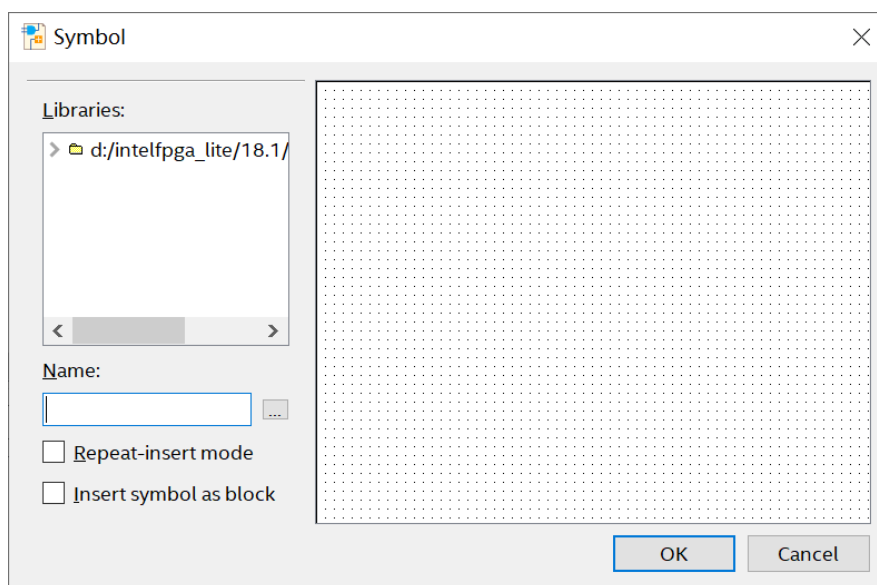


图 2- 17 电路符号选择窗口

Symbol 对话框左上侧 Libraries 部分，列出了当前可以调用的库资源，库资源包含安装 Quartus Prime 软件时自带的资源库和用户自己创建的工程资源库，**错误!未找到引用源。**中只显示了安装时自带的资源库，如果用户创建了自己的工程资源库，则会显示在自带资源库的下方。

在 Symbol 窗口的 Name 处输入 and2，同时输入与门的符号就出现在预览区域中，如图 2- 18 所示。

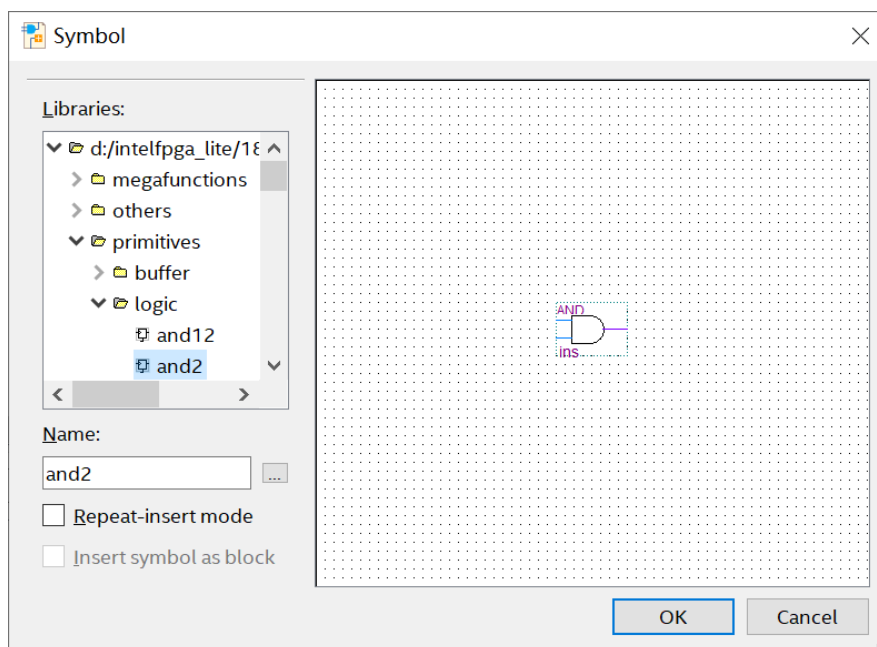


图 2- 18 Symbol 对话框右侧预览符号

在选中需要的原理图符号后, 点击 Symbol 窗口中的 OK 按钮或者直接按回车键, Symbol 窗口消失，返回到原理图编辑窗口，选中的原理图符号会黏在鼠标上随光标移动，在原理图

编辑窗口选择合适的位置点击鼠标左键即可将该符号放到原理图编辑窗口，如图 2- 19 所示。

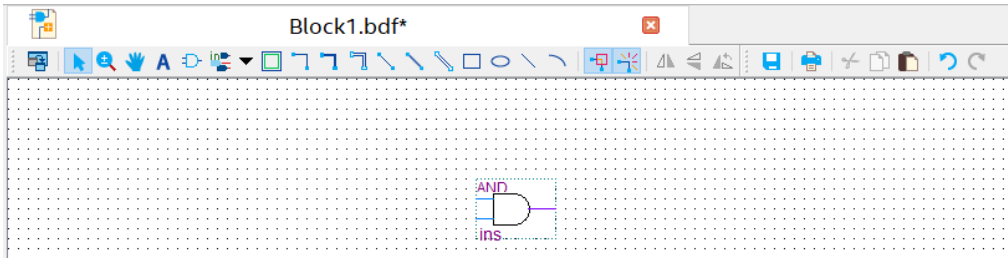


图 2- 19 在原理图文件中添加原理图符号

重复刚才添加二输入与门的方式，将本次实验中所需的原理图符号二输入与门 and2、二输入或门 or2、非门 not、输入端 input、输出端 output 都添加到原理图窗口中，如图 2- 20 所示。

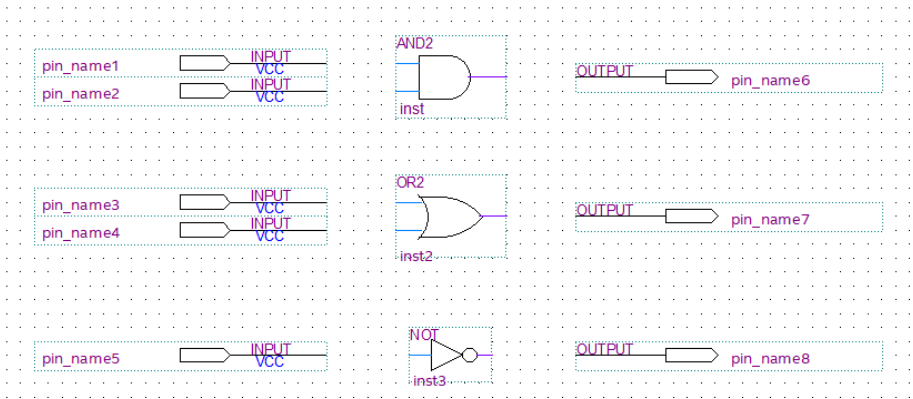


图 2- 20 在原理图文件中添加实现与、或、非电路所有所需符号

使用绘图工具栏中的连线工具或者鼠标拖动方式将输入端口接入对应门的输入端，输出端口与对应门的输出端相连，得到如图 2- 22 所示的与、或、非门电路。连线的时候请注意线不能乱连、错连，如果连线位置不正确，则会出现红色的 x 符号，在这种情况下，将该处的连线删除，重新连接，直到所有的线都正确连接。

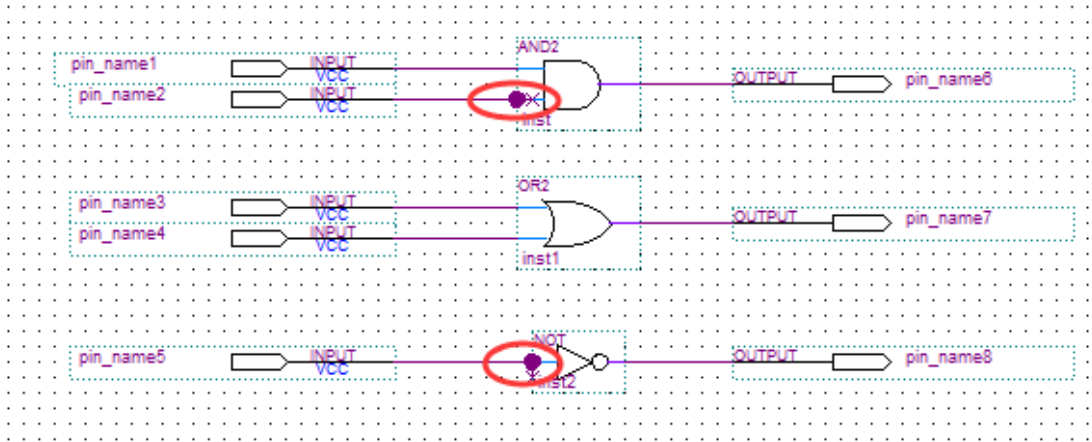


图 2- 21 线未正确连接时出现红 X

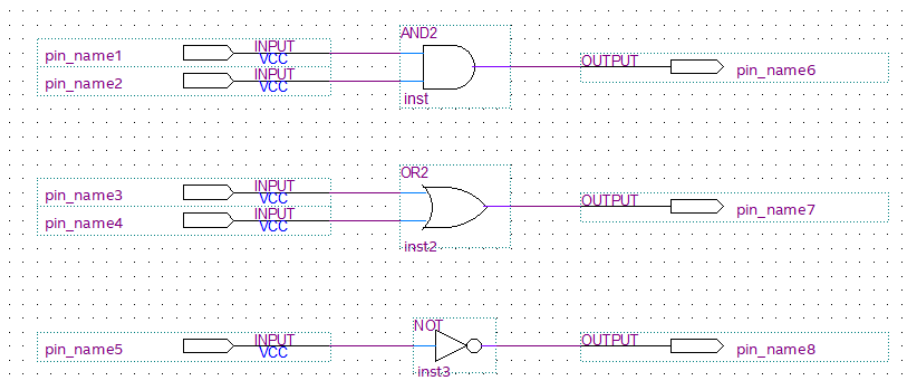


图 2- 22 在原理图文件中实现的与、或、非电路

从原理图添加的输入输出端口均有一个默认的 pin name，为了使这些输入输出的名字看起来更直观，需要为这些端口重命名，重命名的方法为，选中需要被重命名的端口，单击右键，选择 Properties，在 Pin name 处输入合适的端口名，单击 OK，如图 2- 23 示。

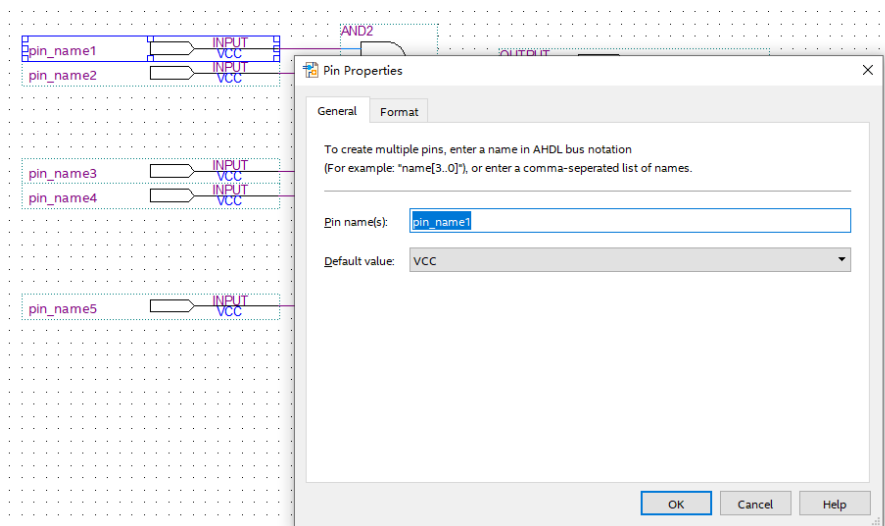


图 2- 23 修改端口名

本例中要求实现二输入与门、二输入或门和非门电路，二输入与门的输入输出分别为 A、B、O1，二输入或门的输入分别为 C、D、O2，非门的输入输出分别为 E 和 O3，所以将 5 个输入依次重命名为 A、B、C、D、E，将输出依次重命名为 O1、O2、O3，如图 2- 24 所示。

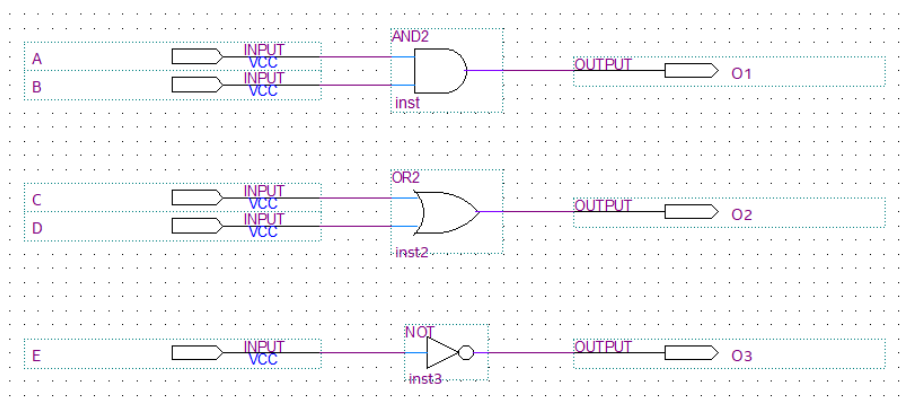


图 2- 24 重命名端口后的电路图

点击保存按钮，保存原理图文件并为设计的原理图文件命名（如 lab2test），注意保存的文件名可以由用户自定义，但是应以字母或下划线开头，中间不能包含特殊字符。如果保存的文件名 lab2test 和新建工程时设置的顶层实体名称 lab2 不一致，需要重新设置顶层文件，方法为在工程导航窗口中将工程导航选项从下拉菜单中由树形 Hierarchy 切换到文件 Files，选中需要被设为顶层的文件，单击右键，选择 Set as Top-Level Entity。

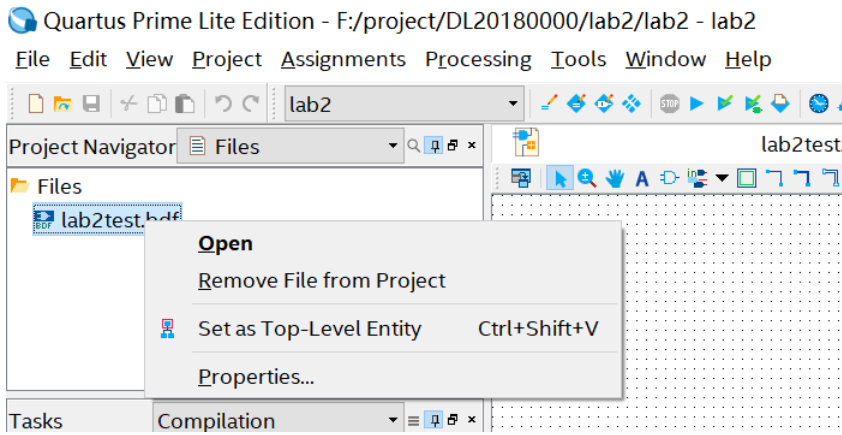


图 2- 25 设置顶层文件

设置顶层文件后，重新切换至工程导航栏的树形结构处，将看到顶层实体已经变为被设置为顶层实体的文件名，如图 2- 26 所示。

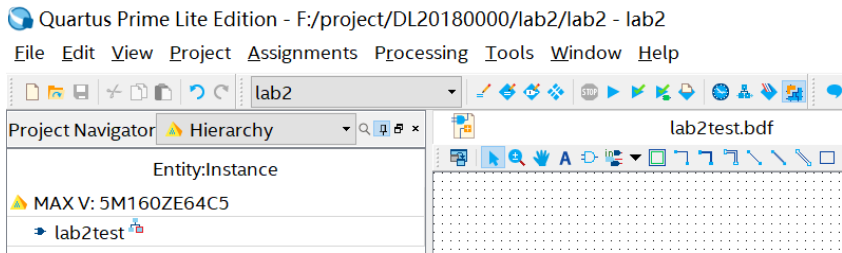



图 2- 26 重置顶层实体后的树形结构图

## 四、编译

点击 Processing->Start Compilation 或者点击快捷菜单键中的编译按钮  进行全编译，全编译包括语法检查、逻辑综合、布局布线、生成编程配置文件、EDA 网表写入等，全编译开始后，Task 任务栏中的每一个小步骤依次执行，并显示编译到哪一个具体的步骤，如图 2-27 所示。同时在 Messages 信息栏中显示相关编译信息，如图 2-28 所示。全编译完成后会产生编译报告，编译报告根据当前执行的编译命令显示当前工程的信息和占用的资源情况。若编译不成功，信息栏中将以红色字体呈现错误信息提示，可根据提示修改设计文档后，重新编译，直到编译成功。全编译成功后在工程目录下将产生一个名为 output\_files 的文件夹，该文件夹中存放有用于下载到目标器件中的配置文件。

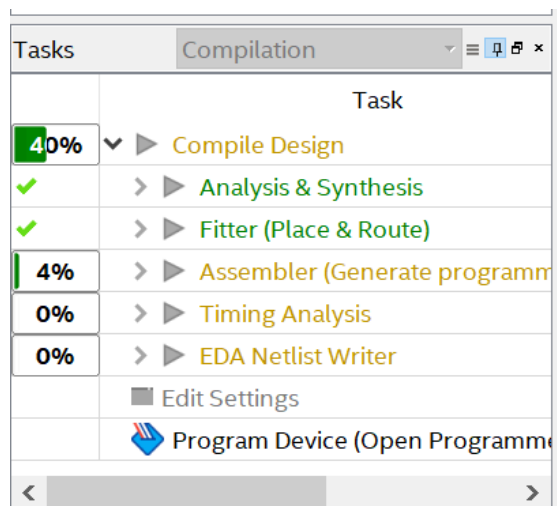


图 2- 27 执行全编译时 Task 任务栏过程显示

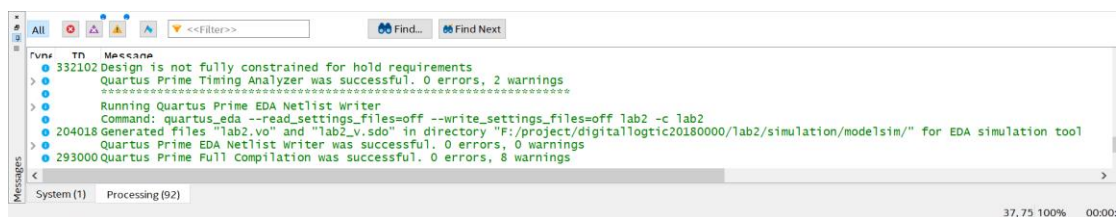


图 2- 28 Message 信息窗口

注意：以上全编译中的各个步骤可以全部执行，也可以单独执行，单独执行时，在 Task 任务栏对应的小步骤上右键点击选择 Start 开始执行，如图 2- 29 所示。

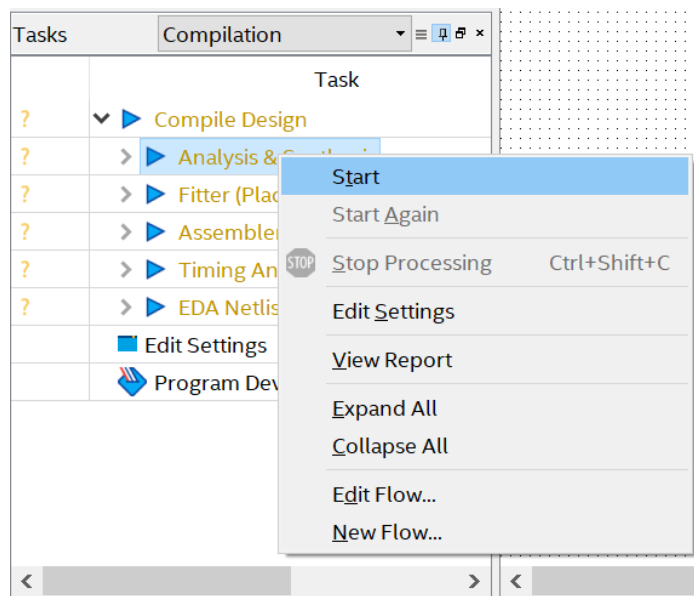


图 2- 29 单独执行 Analysis&Synthesis

## 五、仿真

Quartus Prime 集成了可以进行功能仿真和时序仿真的仿真器，在进行仿真前首先要建立矢量波形文件。

1、 点击 File->New， 在新建窗口中选择位于 Verification/Debugging Files 下的 University Program VWF 文件， 如图 2- 30 所示， 点击 OK 按钮， 出现矢量波形文件窗口， 该窗口包含工具栏、 信号栏和波形栏三部分， 如图 2- 31 所示。

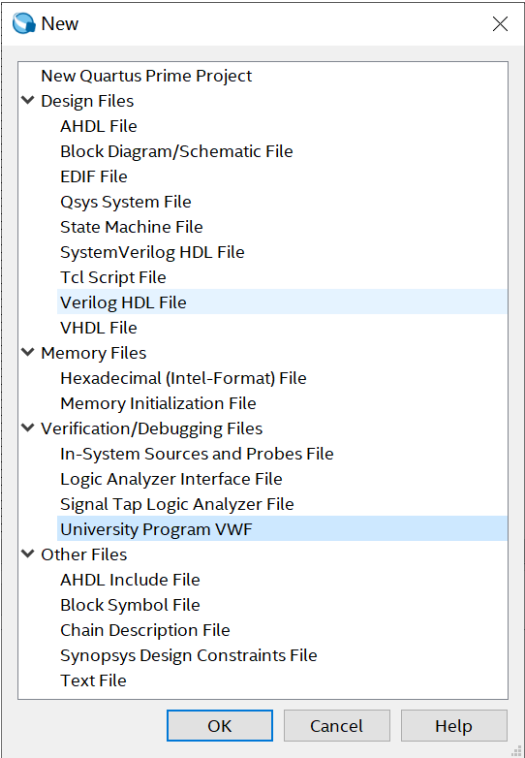


图 2- 30 新建矢量波形文件窗口

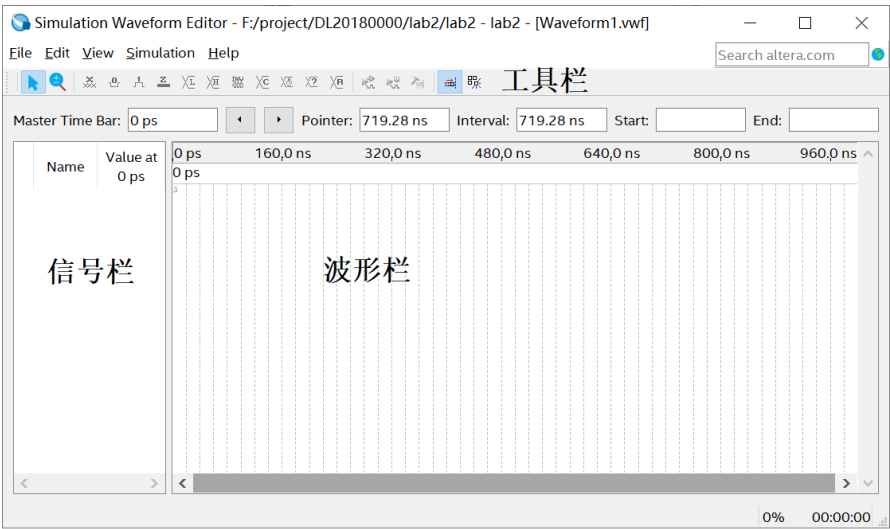


图 2- 31 波形编辑窗口概览

2、 仿真前要添加需要被仿真的信号。在信号栏空白处双击鼠标左键或者在 Edit 菜单

栏下选择 Insert->Insert Node or Bus，弹出如图 2- 32 所示的插入节点或总线对话框。

3、单击图 2- 32 右侧的 Node Finder 按钮，弹出如图 2- 33 所示的 Node finder 对话框，在该对话框中将右上侧 Filter 处选择为 Design Entry(all names)，点击 Filter 下方的 List 按钮，设计中的信号不限于输入输出信号都将在 Nodes Found 栏中显示出来，如图 2- 33 所示，选择所需要仿真的信号（如本例中选择输入结点 A、B、C、D、E 和输出结点 O1、O2、O3）点击 ‘>’ 添加到右侧 Selected Nodes 栏中，如图 2- 34 所示。如果要选中所有列出节点，则直接点击 ‘>>’ 按钮即可将所有节点添加到 Selected Nodes 栏中。完成选择后点击 OK 返回 Insert Node or Bus 窗口，再点击 OK 即可将信号添加到矢量波形编辑窗口，如图 2- 35 所示，此时所有的输入信号均默认为 0。

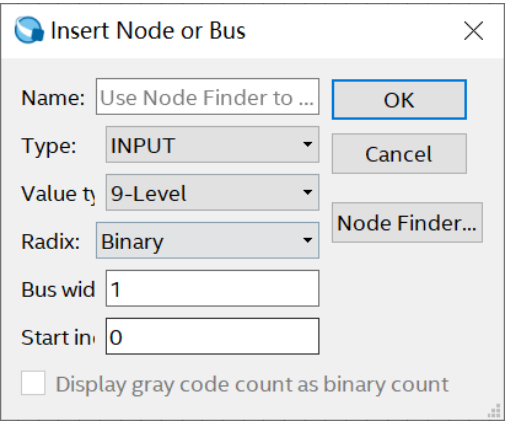


图 2- 32 插入节点或总线对话框

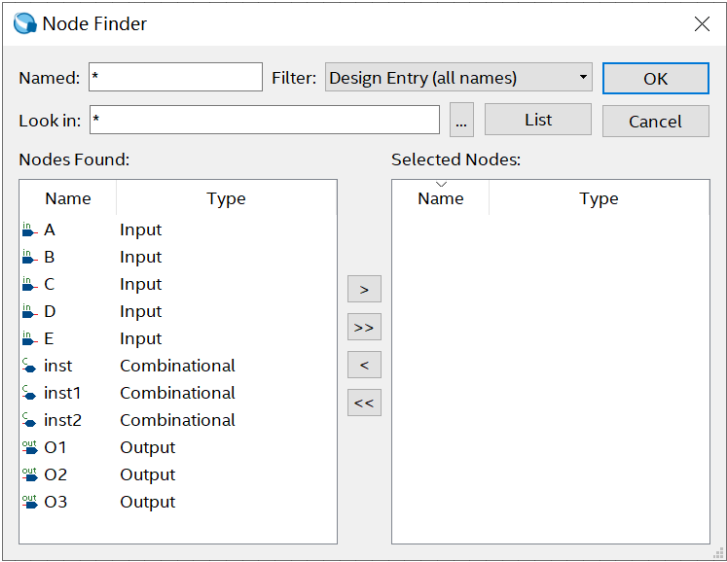


图 2- 33 Nodes Finder 窗口



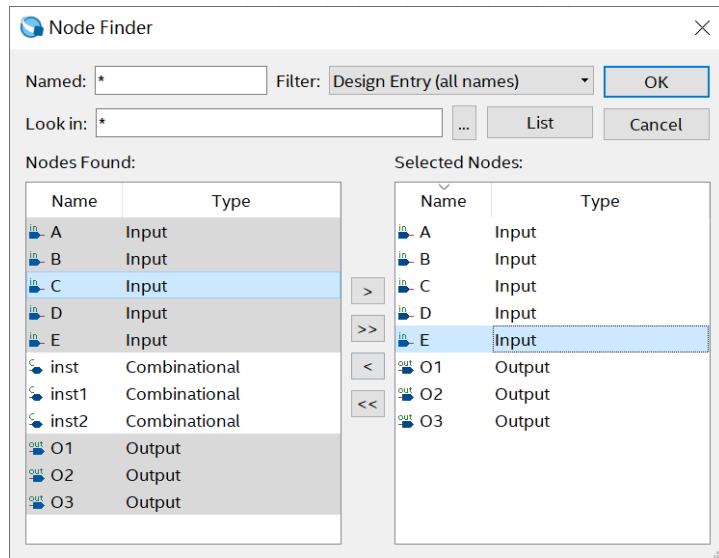


图 2- 34 添加所需仿真的节点

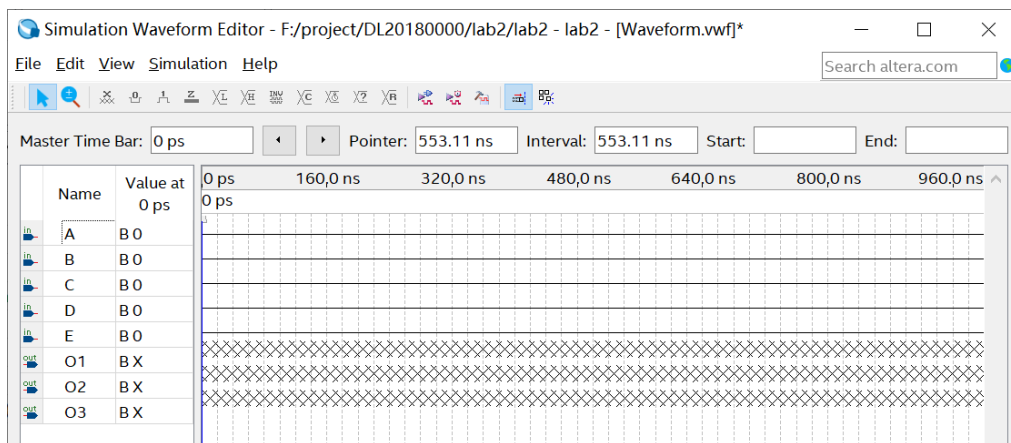




图 2- 35 添加信号后的矢量波形编辑窗口

4、在波形编辑窗口中，选中信号栏中的某个输入信号，通过工具栏或者 Edit 菜单栏下 Value 子菜单来编辑输入信号，如选中信号的某一段点击工具栏中  将信号强制为 0，点击  将信号强制为 1。

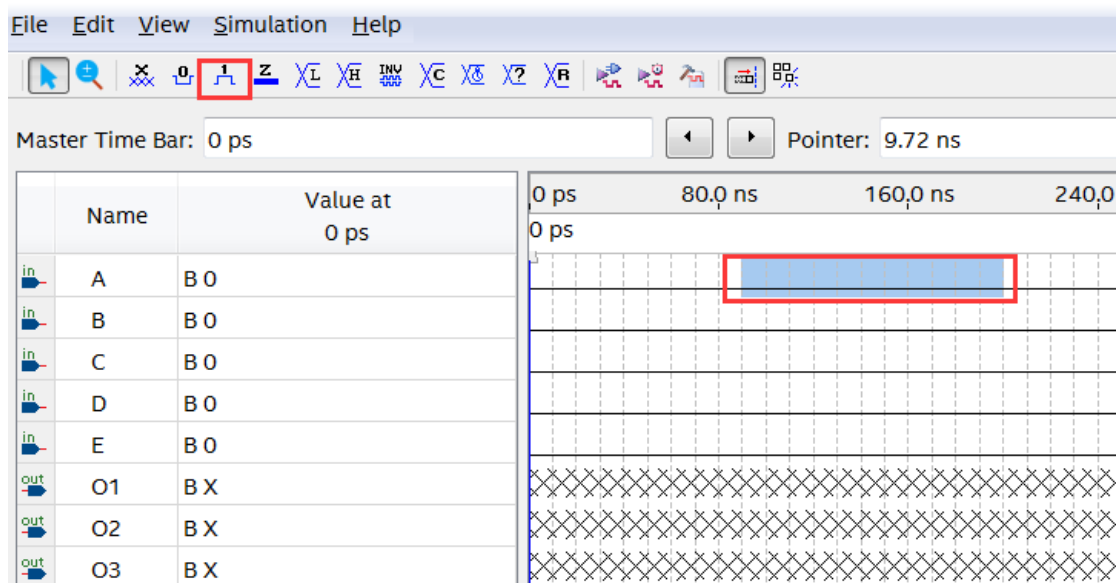


图 2- 36 设置激励波形信号（1）

设置完成的激励波形信号如所示。

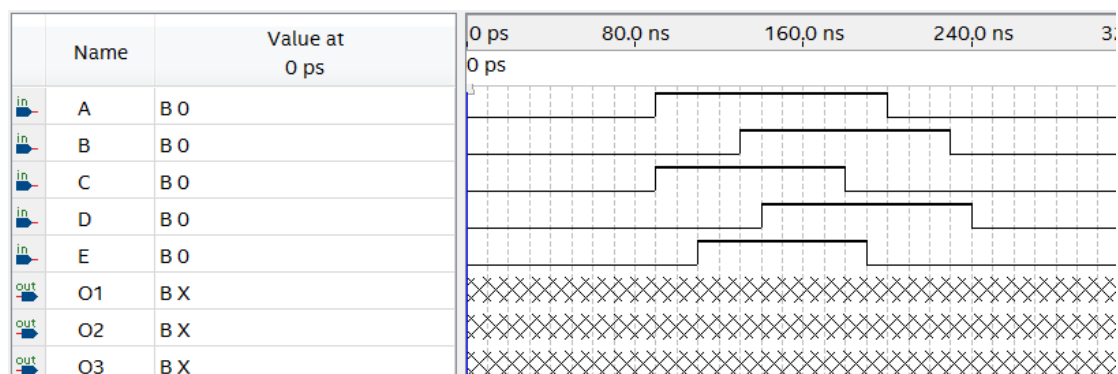
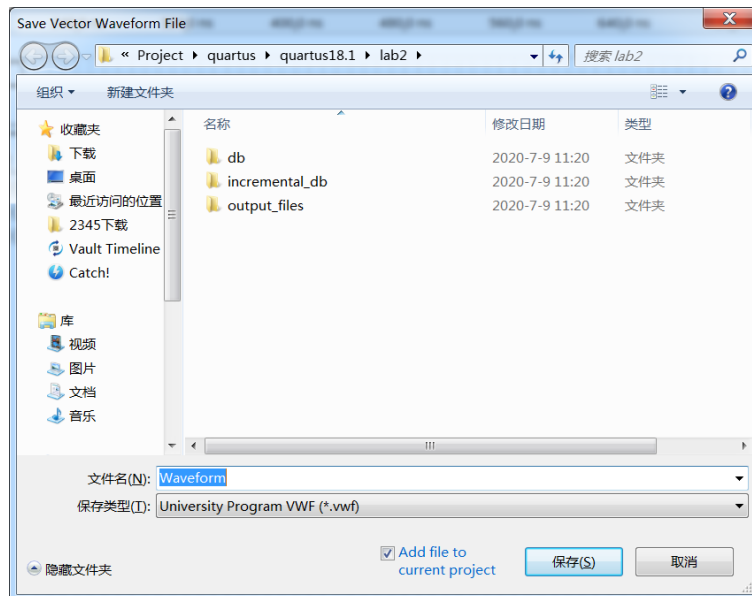



图 2- 37 设置激励波形信号（2）

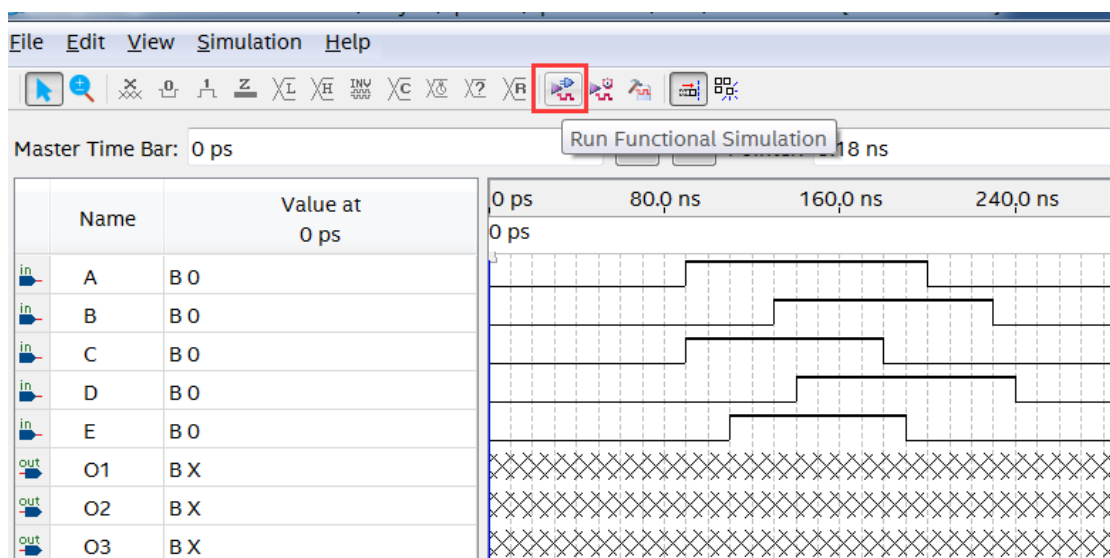
注意：在进行激励信号设置时，要保证输入信号的完备性，即要满足信号所有的输入情况。

思考：如何才能满足输入信号的完备性？

5、 点击 File->Save 或者 Ctrl+s，将设置好的激励波形文件保存到工程中。保存时可以为波形文件进行命名。



6、 然后点击工具栏上的功能仿真按钮 ，运行功能仿真。



7、 仿真结束后，弹出图 2- 38 所示的仿真结果图。

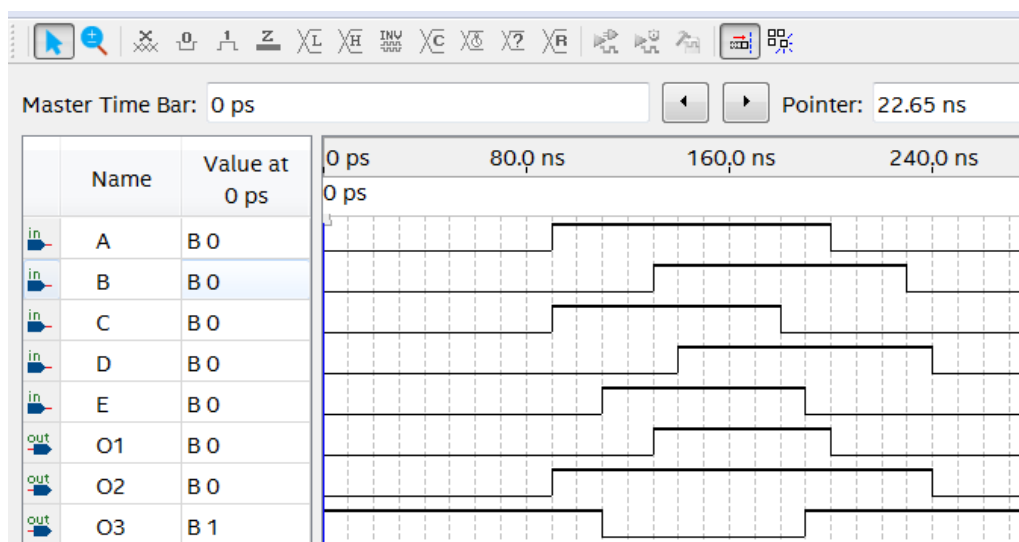


图 2- 38 仿真结果图

仿真结果出现后，应根据仿真结果图，判断所设计电路逻辑是否正确。注意：波形图中的信号位置是可以进行拖动的，方法为：鼠标左击选中待移动的信号，然后拖拽至合适的位置，如本例中 A、B 通过二输入与门得到输出信号 O1，可以将 O1 拖拽至 B 信号的下方，方便观察结果是否正确，同理将 O2 拖拽至 D 信号的下方，拖动之后的运行结果图如图 2- 39 所示。

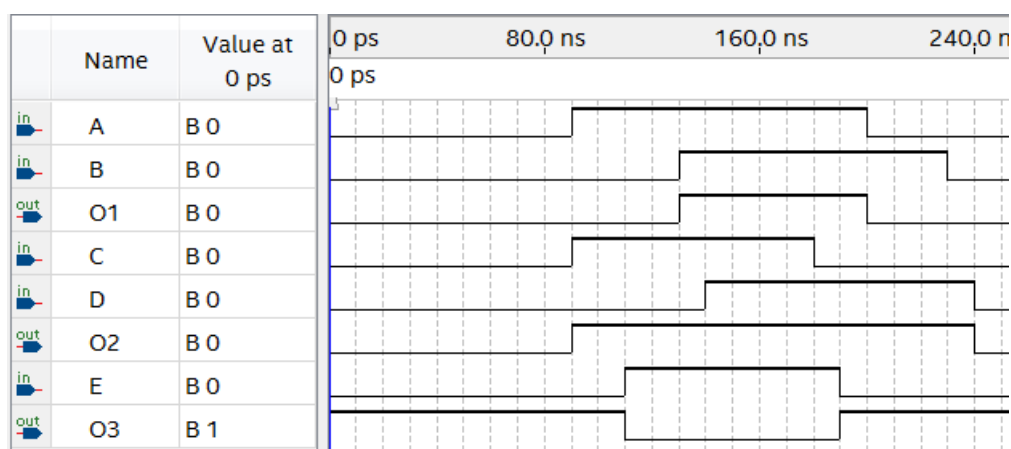


图 2- 39 拖动输出信号位置后的仿真结果图

## 六、管脚分配

略。本次实验不做

## 七、器件编程

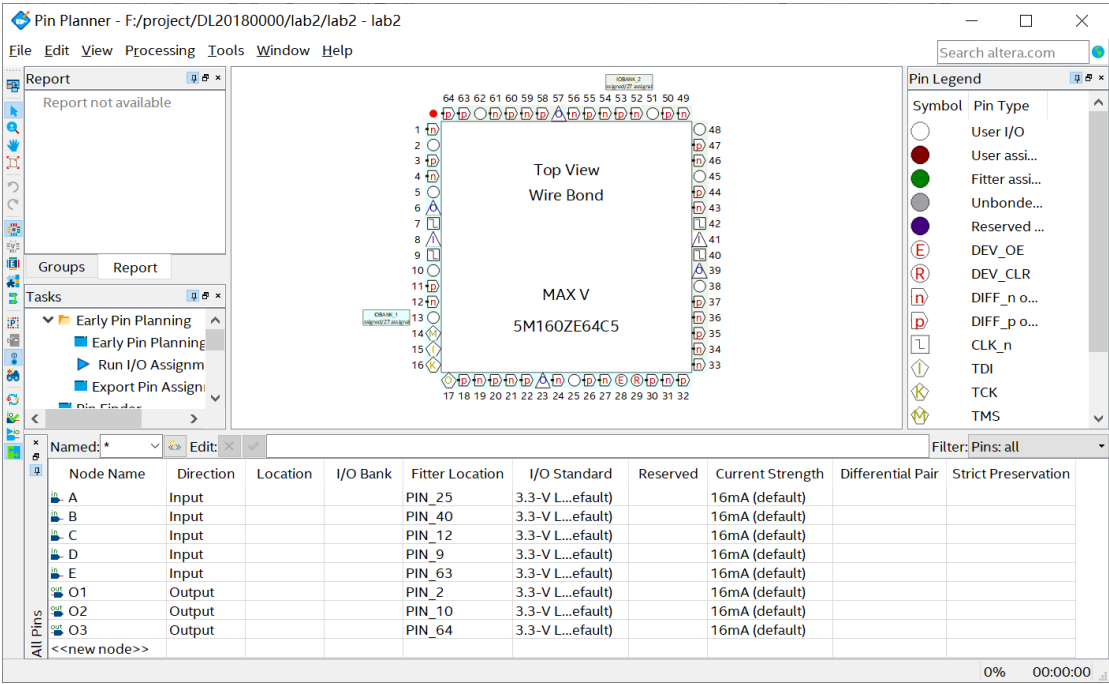
略。本次实验不做。

# 附录 2 : Quartus Prime18.1 器件编程流程

接附录 1，补充管脚分配和器件编程部分。新建工程、设计输入、编译、功能仿真部分见附录 1。

- 1、打开软件
- 2、新建工程
- 3、设计输入
- 4、编译
- 5、功能仿真
- 6、管脚分配

1) 选择 Assignments->Pin Planner 或者点击 Pin Planner 图标，进入 Pin Planner 用户界面



Pin Planner 用户界面

2) 拖拽All Pins窗口中的管脚至封装图对应的位置，或可以双击管脚所在行对应Location列的空白处，从出现的下拉菜单中选择对应的管脚编号。


将输入管脚分配至距拨位开关近的位置，输出管脚分配至距LED灯近的位置，由于实验箱上的芯片与其他电路结构的连接关系已经确定，所以实际分配管脚，应按照电路原理图的连接关系进行分配。

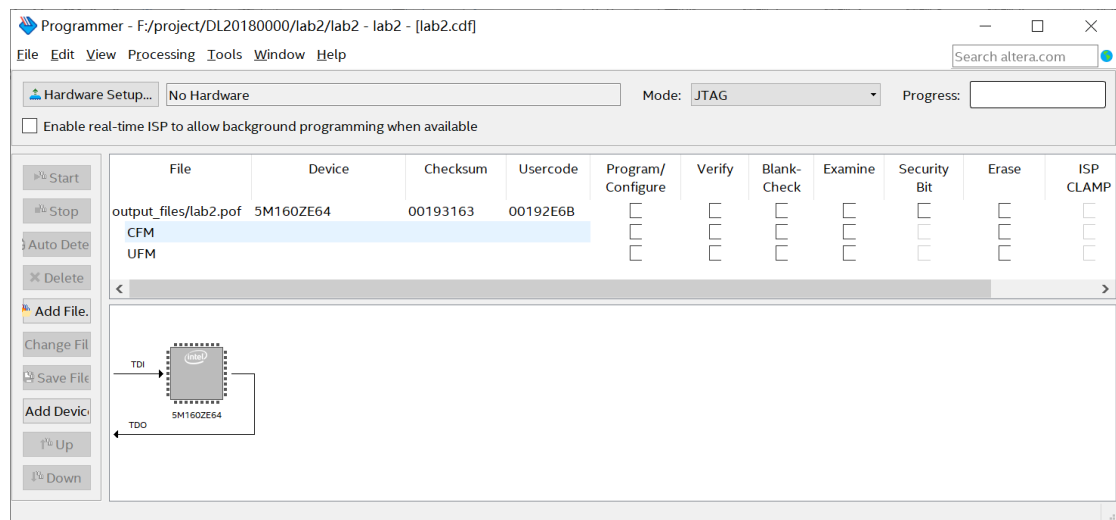
Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
A	Input	PIN_9	1	PIN_25	3.3-V LVTTTL (default)		16mA (default)
B	Input	PIN_11	1	PIN_40	3.3-V LVTTTL (default)		16mA (default)
C	Input	PIN_13	1	PIN_12	3.3-V LVTTTL (default)		16mA (default)
D	Input	PIN_19	1	PIN_9	3.3-V LVTTTL (default)		16mA (default)
E	Input	PIN_21	1	PIN_63	3.3-V LVTTTL (default)		16mA (default)
O1	Output	PIN_5	1	PIN_2	3.3-V LVTTTL (default)		16mA (default)
O2	Output	PIN_3	1	PIN_10	3.3-V LVTTTL (default)		16mA (default)
O3	Output	PIN_1	1	PIN_64	3.3-V LVTTTL (default)		16mA (default)
<<new node>>							

分配管脚后的All Pins界面

3) 管脚分配完成后，对工程进行重新编译。

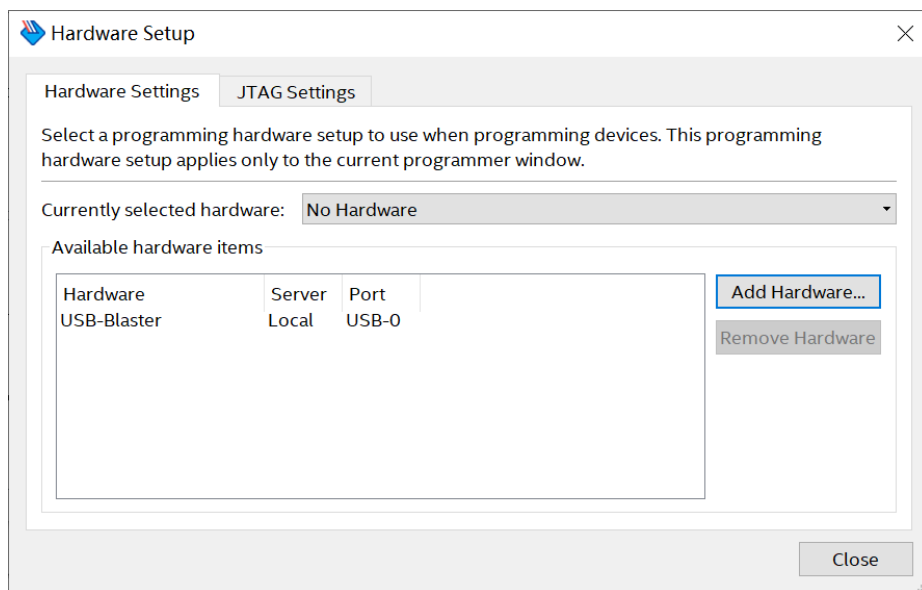
## 7、器件编程

1) 点击 Programmer 图标 ，弹出 Programmer 界面。从界面左上角 Hardware Setup 右侧的文本框中若看到 No Hardware 字样，说明 Programmer 还未指定编程器。

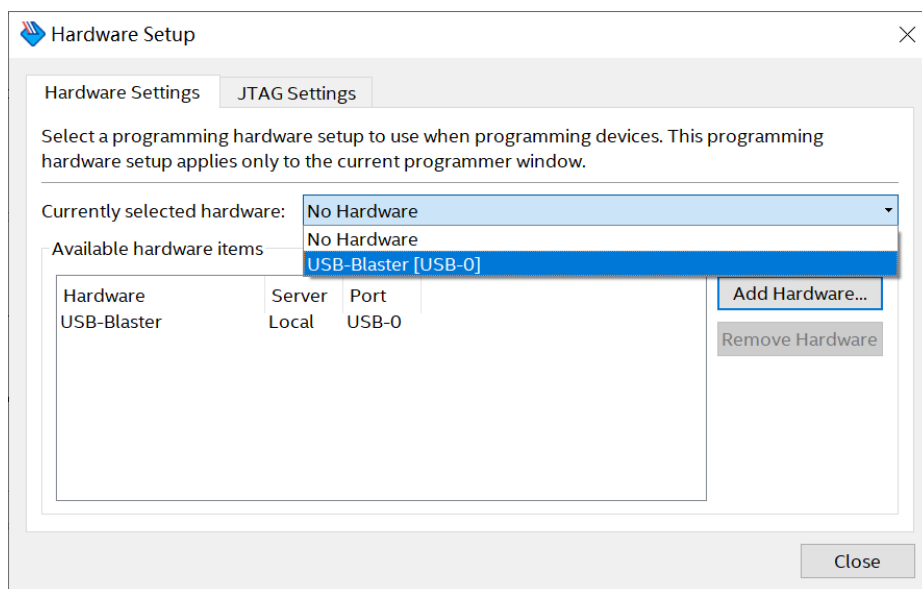


Programmer 界面

2) 将编程器（本例为 USB Blaster）与计算机和目标板卡连接好，确保已经正确安装了 USB Blaster 的驱动程序后（若初次接上 USB Blaster 编程器，则需要为设备安装驱动程序，驱动程序位于 quartus 安装目录下的 drivers 文件夹下），单击 Hardware Setup 按钮，弹出 Hardware Setup 对话框，在 Currently selected hardware 右侧的下拉菜单中列出了当前可用的编程器，选择 USB-Blaster，然后点击 Close 关闭对话框，在 Programmer 界面中 Hardware Setup 按钮后将出现选择的编程器，此时硬件设置完成。

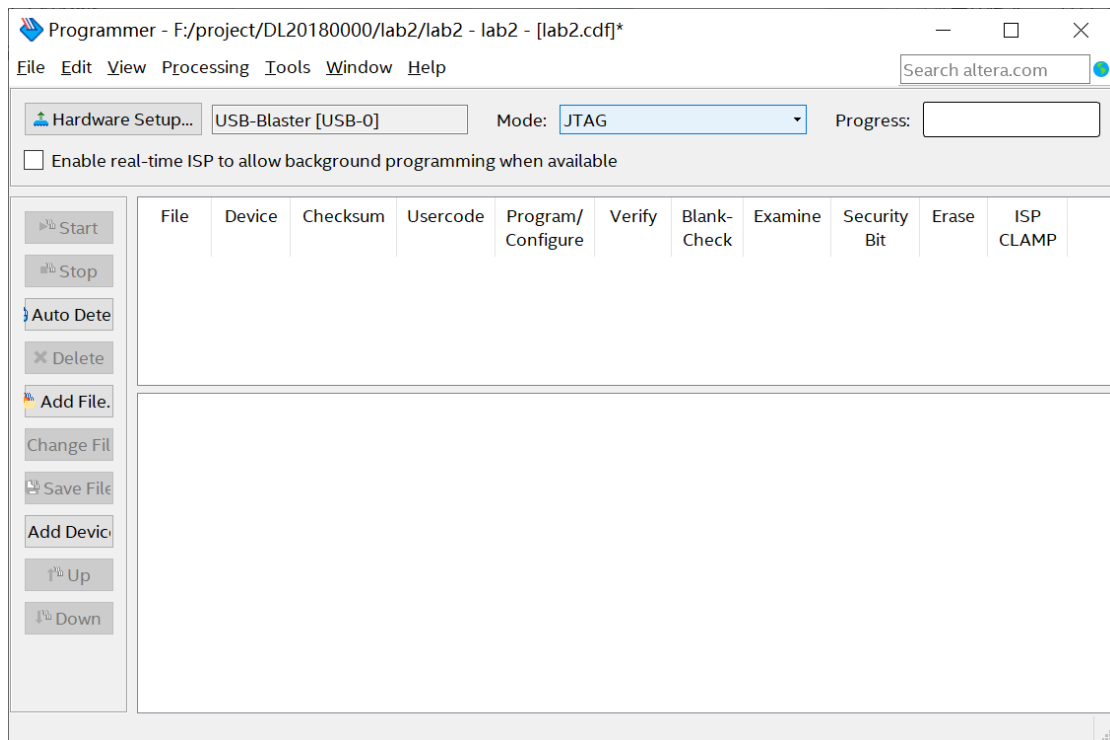


Hardware Setup 界面



选中编程器件 USB Blaster

3) 硬件设置完成后，在 Mode 栏下选择对应的编程方式如 JTAG。选择待编程文件，单击左侧的 Add File 按钮添加待编程文件，注意：工程编译成功后将产生唯一一个名称与工程名相同的.pof文件用于下载。待编程文件添加完成后，勾选 Program/Configure 选项，点击左侧的 Start 按钮，开始下载，在 Progress 进度条中显示编程进度，编程完成后 Progress 的进度显示为 100% Successful，此时可以在目标板卡上进行程序验证。



未关联待下载文件的下载界面



## 附录 3 : Verilog 语法初步

### 1. module(模块)

#### a. module 的基本概念

module 是层次化设计的基本构件，逻辑描述放在 module 内部。示例如下：

```
module test(  
input a, b, c, d, e,  
output o1, o2, o3  
);  
    assign o1 = a&b;  
    assign o2 = c|d;  
    assign o3 = ~e;  
endmodule
```

module 能够表示物理块，如 IC 或 ASIC 单元；或逻辑块，如一个 CPU 设计的 ALU 部分；

也可以表示整个系统。每一个模块的描述从关键词 module 开始，有一个名称（如上例中的 test），由关键词 endmodule 结束。

#### b. 模块端口

端口定义了该模块的输入输出引脚，模块通过端口与外部通信。端口有三种类型 input、output、inout。上例中的端口信号有 a、b、c、d、e 和 o1、o2、o3。

#### c. 模块实例化

一个模块中可以包含其它模块，在一个模块中通过模块实例化来调用另一个模块。每个实例都有自己的名字(u1, u2, u3)。实例名是每个对象唯一的标记，通过这个标记可以查看每个实例的内部。实例中端口的次序与模块定义的次序相同。模块实例化与调用程序不同。每个实例都是模块的一个完全的拷贝，相互独立、并行。

```
module test(  
input a, b, c, d, e,  
output o1, o2, o3  
);  
    and u1(o1,a,b);  
    or u2(o2,c,d);  
    not u3(o3,e);
```

### 2. 整数常量和实数常量

Verilog 语言中，整数的表示方式为：<size>'<base><value>，其中

size：表示占用的二进制位宽 bit。缺省为 32 位

base：数基，可为 2 进制(b)、8 进制(o)、10 进制(d)、16 进制(h)。缺省为 10 进制 value：是所选数基内任意有效数字，包括 X、Z。

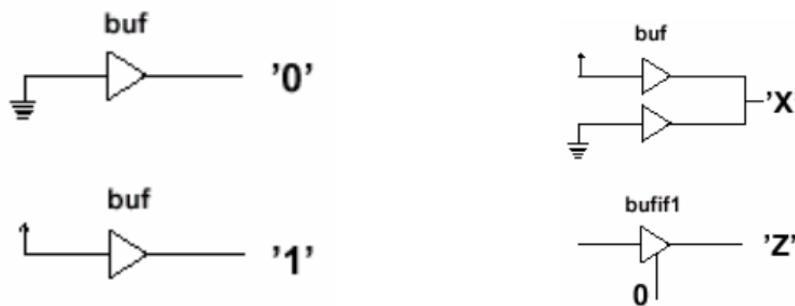
8'd12	8 表示 8bit, d 表示是十进制。该例表示值 12
12'H83A	12 表示 12bit, H 表示是十六进制。该例表示值为 16 进制的 83A
4'b0111	4 表示 4bit, b 表示二进制。该例表示二进制 0111，即 7。

整数的大小可以定义也可以不定义。没有定义大小(size)的整数缺省为 32bit, 缺省数基为十进制。数基(base)和数字(16 进制)中的字母无大小写之分。当数值 value 大于指定的大小时，截去高位。如 2'b1101 表示的是 2'b01。

实数常量可以用十进制或科学表示法表示。科学表示法表示方式如下：

<尾数><e 或 E><指数>，表示：尾数 $\times 10$  指数。例：32E-4 表示 0.0032；4.1e3 表示 4100。

### 3. Verilog 四值逻辑系统



Verilog 中采用四值逻辑系统：0 表示逻辑假或低电平，1 表示逻辑真或高电平，‘X’表示不确定的值，‘Z’表示高阻态。

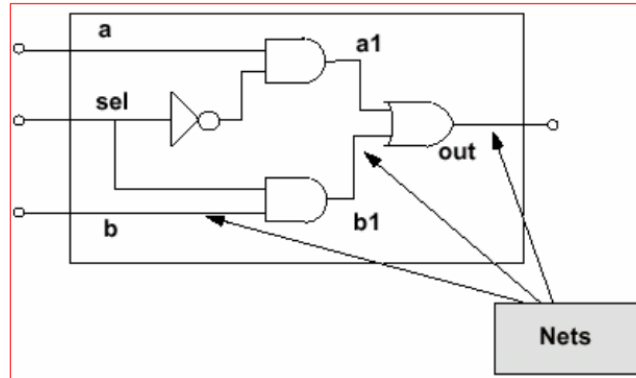
### 4. 主要数据类型

Verilog 主要有三类(class)数据类型：net(线网)，表示器件之间的物理连接；register(寄存器)，表示抽象存储元件；parameters(参数)：运行时的常数。

#### a. net 线网

net 需要被持续的驱动，驱动它的可以是门和模块。当 net 驱动器的值发生变化时，Verilog 自动的将新值传送到 net 上。

在下图中，线网 out 由 or 门驱动。当 or 门的输入信号置位时将传输到线网 net 上。



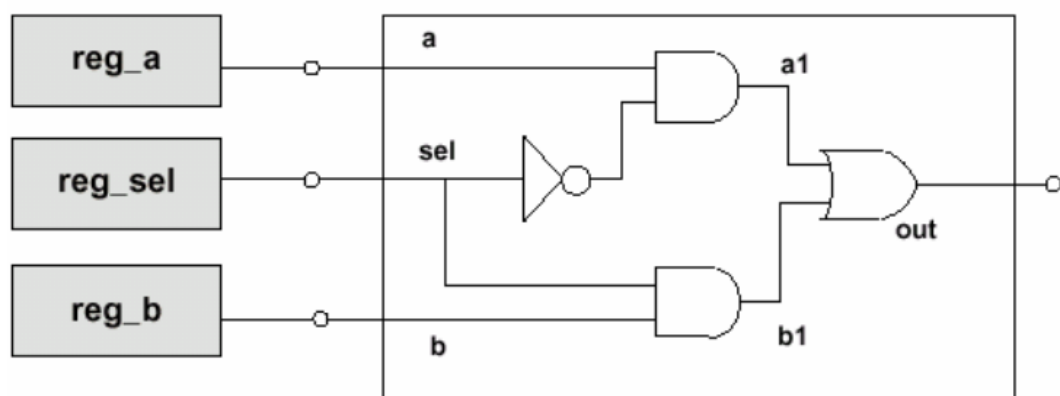
net 的分类如下表所示。wire 类型是最常用的类型，只有连接功能。没有声明的 net 的缺省类型为 1 位(标量)wire 类型。但这个缺省类型可由编译指导改变：'default\_nettype <nettype>。

net类型	功 能
wire, tri	标准内部连接线(缺省)
supply1, supply0	电源和地
wor, <b>*trior</b>	多驱动源线或
wand, <b>*triand</b>	多驱动源线与
<b>*triereg</b>	能保存电荷的net
<b>*tri1, tri0</b>	无驱动时上拉/下拉



## b. register 寄存器

寄存器类型在赋新值以前保持原值，寄存器类型大量应用于行为模型描述及激励描述。 在下面的例子中，reg\_a、 reg\_b、 reg\_sel 用于施加激励给 2:1 多路器。用行为描述结构给寄存器类型赋值。给 reg 类型赋值是在过程块中。



reg 的类型如下表所示：

寄存器类型	功能
<b>reg</b>	可定义的无符号整数变量，可以是标量(1位)或矢量，是最常用的寄存器类型
<b>integer</b>	32位有符号整数变量，算术操作产生二进制补码形式的结果。通常用作不会由硬件实现的的数据处理。
<b>real</b>	双精度的带符号浮点变量，用法与 <b>integer</b> 相同。
<b>time</b>	64位无符号整数变量，用于仿真时间的保存与处理
<b>realtime</b>	与 <b>real</b> 内容一致，但可以用作实数仿真时间的保存与处理

### c. Verilog 中 net 和 register 声明语法

net 声明: `<net_type> [range] [delay] <net_name>[, net_name];`

net\_type: net 类型

range: 矢量范围，以[MSB: LSB]格式

delay: 定义与 net 相关的延时

net\_name: net 名称，一次可定义多个 net, 用逗号分开。

寄存器声明: `<reg_type> [range] <reg_name>[, reg_name];`

reg\_type: 寄存器类型

range: 矢量范围，以[MSB: LSB]格式。只对 reg 类型有效 reg\_name :

寄存器名称，一次可定义多个寄存器，用逗号分开。

举例: `reg a; // 一个标量寄存器`

`wand w; // 一个标量 wand 类型 net`

`reg [3: 0] v; // 从 MSB 到 LSB 的 4 位寄存器向量` `reg [7: 0] m, n; // 两个 8 位寄存器`

`tri [15: 0] busa; // 16 位三态总线`

`wire [0: 31] w1, w2; // 两个 32 位 wire, MSB 为 bit0`

信号可以分为端口信号和内部信号。出现在端口列表中的信号是端口信号，其它的信号为内部信号。对于端口信号，输入端口只能是 net 类型。输出端口可以是 net 类型，也可以是 register 类型。若输出端口在过程块中赋值则为 register 类型；若在过程块外赋值(包括实例化语句)，则为 net 类型。内部信号类型与输出端口相同，可以是 net 或 register 类型。判断方法也与输出端口相同。若在过程块中赋值，则为 register 类型；若在过程块外赋值，则为 net 类型。

### d. parameter 参数

用参数声明一个可变常量，常用于定义延时及宽度变量。

参数定义的语法: `parameter <list_of_assignment>;`

可一次定义多个参数，用逗号隔开。参数的定义是局部的，只在当前模块

中有效。参数定义可使用以前定义的整数和实数参数。

```
module mod1( out, in1, in2); ...  
    Parameter IDLE = 4'b0000;    endmodule
```

5. Verilog 操作符

操作符类型	符号	<div>最高</div> <div>↑</div> <div>优先级</div> <div>↓</div> <div>最低</div>
连接及复制操作符	{ } { }	
一元操作符	! ~ &   ^	
算术操作符	* / % + -	
逻辑移位操作符	<< >>	
关系操作符	> < >= <=	
相等操作符	== === != !==	
按位操作符	& ^ ~^ 	
逻辑操作符	&& 	
条件操作符	? :	

6. 过程块和过程赋值

过程块是行为模型的基础。过程块有两种：initial 块，只能执行一次，不能综合，用在仿真中；always 块，循环执行。

过程块中有下列部件：

- a 过程赋值语句：在描述过程块中的数据流
- b 高级结构（循环，条件语句）：描述块的功能
- c 时序控制：控制块的执行及块中的语句。

在过程块中的赋值称为过程赋值。在过程赋值语句中表达式左边的信号必须是寄存器类

型（如 reg 类型）。在过程赋值语句等式右边可以是任何有效的表达式，数据类型也没有限制。如果一个信号没有声明则缺省为 wire 类型。使用过程赋值语句给 wire 赋值会产生错误。如下例中的 half\_carry 在过程块 always 中赋值，应该是 reg 型变量。但前面并没有对这个变量声明，所以缺省为 wire 类型。这里就出现了错误。应该将其声明为 reg 型，就解决了问题。

```

module adder (out, a, b, cin);
    input a, b, cin;
    output [1:0] out;
    wire a, b, cin;
    reg half_sum;
    reg [1: 0] out;
    always @( a or b or cin)
    begin
        half_sum = a ^ b ^ cin ; // OK
        half_carry = a & b | a & !b & cin | !a & b & cin ; //
ERROR!
        out = {half_carry, half_sum} ;
    end
endmodule

```

half\_carry  
没有声明

时序控制@可以用在 RTL 级或行为级组合逻辑或时序逻辑描述中。可以用关键字 posedge 和 negedge 限定信号敏感边沿。敏感表中可以有多个信号，用关键字 or 连接。

## 7. 持续赋值

可以用持续赋值语句描述组合逻辑，代替用门及其连接描述方式。持续赋值在过程块外部使用。持续赋值用于 net 驱动。持续赋值只能在等式左边有一个简单延时说明。只限于在表达式左边用#delay 形式。持续赋值可以是显式或隐含的。

语法：<assign> [#delay] [strength] <net\_name> = <expressions>;

```

wire out;

assign out = a & b; // 显式

wire inv = ~in; // 隐含

```

## 8. 基本语句

### a. 条件语句（if 分支语句）

可以多层嵌套。在嵌套 if 序列中，else 和前面最近的 if 相关。为提高可读性及确保正确关联，使用 begin...end 块语句指定其作用域。例，一个带使能的二选一数控开关可用以下语句实现。

```

always @(*) begin
    if(en == 1'b1) begin
        if(sel == 1'b1) y <= a;
        else            y <= b;
    end
    else begin
        y <= 0;
    end
end

```

#### 描述方式:

if(表达式)

begin

.....

end

else

begin

.....

end

#### b. 条件语句 (case 语句)

除了 if 语句，verilog 中同样的还有 case 语句。case 语句的一般写法如下所示，这同时也是一个数据分配器的具体实现。

```

always @(*) begin
    case({a,b})
        2'b00: {y3,y2,y1,y0} <= {3'b000,i};
        2'b01: {y3,y2,y1,y0} <= {2'b00,i,1'b0};
        2'b10: {y3,y2,y1,y0} <= {1'b0,i,2'b00};
        2'b11: {y3,y2,y1,y0} <= {i,3'b000};
        default: ;
    endcase
end

```

#### c. 循环语句

Verilog 中，有四种循环语句：

repeat，将一块语句循环执行确定次数；

repeat (次数表达式) <语句>

while：在条件表达式为真时一直循环执行；

while (条件表达式) <语句> forever：重复执行直到仿真结束；

forever <语句>

for：在执行过程中对变量进行计算和判断，在条件满足时执行；

for(赋初值；条件表达式；计算) <语句>

循环语句在 verilog 中不常用，因为只有 for 语句是可以被综合的，其它三种仅在仿真中使用。