

JC-SOPC-V

FPGA 实验系统平台

实验指导书

(EDA 部分)

北京杰创永恒科技有限公司

# 目录

前    言 .....	3
基础实验部分 .....	4
实验一  简单的 VIVADO 实例设计 .....	4
实验二  基于 VERILOG 格雷码编码器的设计 .....	27
实验三  含异步清零和同步使能的加法计数器 .....	29
实验四  八位七段数码管动态显示电路的设计 .....	33
实验五  数控分频器的设计 .....	43
实验六  VERILOG 层次结构电路设计 .....	45
实验七  步长可变的加减计数器的设计 .....	48
实验八  四位并行乘法器的设计 .....	50
实验九  设计四位全加器 .....	53
实验十  可控脉冲发生器的设计 .....	56
实验十一  基本触发器的设计 .....	58
应用实验部分 .....	60
实验十二  矩阵键盘显示电路的设计 .....	60
实验十三  16*16 点阵显示实验 .....	71
实验十四  直流电机的测速实验 .....	83
实验十五  步进电机驱动控制 .....	87
实验十六  PS2 接口键盘显示实验 .....	91
实验十七  VGA 彩条信号发生器的设计 .....	97
实验十八  用 VERILOG 设计七人表决器 .....	101
实验十九  用 VERILOG 设计四人抢答器 .....	103
实验二十  正负脉宽调制信号发生器设计 .....	105
综合设计实验 .....	107
实验二一  数字频率计的设计 .....	107
实验二二  多功能数字钟的设计 .....	112
实验二三  数字秒表的设计 .....	115
实验二四  出租车计费器的设计 .....	117
实验二五  基于 VERILOG 的数码锁的设计 .....	120
附录 1 管脚表格 .....	122

# 前 言

近十年由于超大规模集成电路和软件技术的快速发展,使数字系统集成到一片集成电路内成为可能,Altera、Xilinx、AMD 等公司都推出了非常好的 CPLD 和 FPGA 产品,并为这些产品的设计配备了设计、下载软件,这些软件除了支持图形方式设计数字系统外,还支持设计多种数字系统的设计语言,使数字系统设计起来更加容易。在小规模数字集成电路就要淘汰的今天,作为一个电子技术工程技术人员不懂 VERILOG 语言和 CPLD、FPGA 器件设计就象在计算机时代不会使用计算机一样可怕。

本实验指导书的目的就是帮助读者学会设计数字系统,并熟悉 Xilinx 公司产品和软件 Vivado 及其它相关软件的使用。

本实验指导书的实验内容从简单的组合电路的设计到复杂的数字系统的设计,详细的介绍了系统的设计方法和软件的各种操作。读者可以通过这本实验指导书设计自己的数字电路。

本实验指导书选编了有代表性的实验近三十个,实验内容从简单到复杂,使使用者能够很快的入手,同时本实验指导书还可以作为电子技术的加深课程或作为电子技术工程师参考用书。

本实验指导书配合 JC-SOPC-V FPGA 实验系统平台系列产品使用。如果用户有批评和建议可以和我们联系:

由于时间仓促,资料缺乏,有错误之处请读者原谅。

编者

# 基础实验部分

## 实验一 简单的 VIVADO 实例设计

### 一、 实验目的：

- 1. 通过一个简单的 3—8 译码器的设计，掌握组合逻辑电路的设计方法。
- 2. 初步了解 VIVADO 设计仿真，创建约束，在线下载的全过程。
- 3. 掌握组合逻辑电路的设计方法。

### 二、 实验原理：

3-8 译码器三输入，八输出。当输入信号按二进制方式的表示值为 N 时，输出端标号为 N 的输出端输出高电平表示有信号产生，而其它则为低电平表示无信号产生。因为三个输入端能产生的组合状态有八种，所以输出端在每种组合中仅有一位为高电平的情况下，能表示所有的输入组合。其真值表如表 1-1 所示

输入			输出							
A	B	C	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1		0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

表 1-1 三-八译码器真值表

译码器不需要像编码器那样用一个输出端指示输出是否有效。但可以在输入中加入一个输出使能端，用来指示是否将当前的输入进行有效的译码，当使能端指示输入信号无效或不用对当前信号进行译码时，输出端全为高电平，表示无任何信号。本例设计中没有考虑使能输入端，自己设计时可以考虑加入使能输入端时，程序如何设计。

三、 实验内容：

在本实验中，用三个拨动开关来表示三八译码器的三个输入（A、B、C）；用八个 LED 来表示三八译码器的八个输出（D0-D7）。通过输入不同的值来观察输入的结果与三八译码器的真值表（表 1-1）是否一致。实验箱中的拨动开关与 FPGA 的接口电路如下图 1-1 所示，当开关闭合（拨动开关的档位在下方）时其输出为低电平，反之输出高电平。其电路与 FPGA 的管脚接口电路如图 1-2 所示。

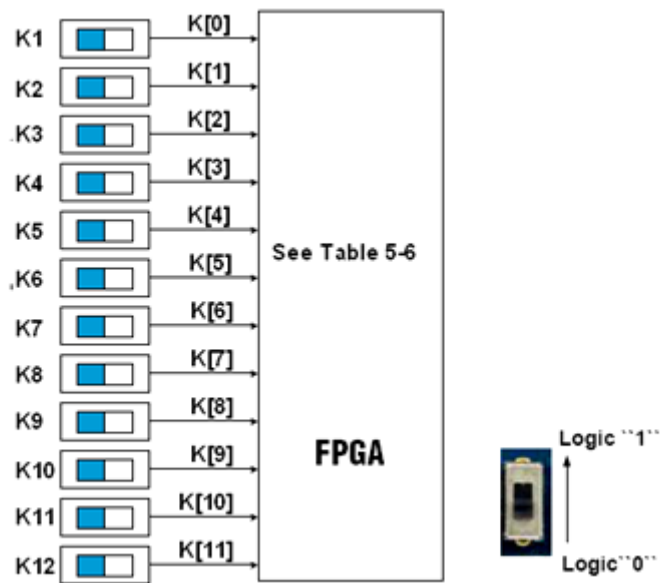


图 1-2 拨动开关与 FPGA 接口电路（1）

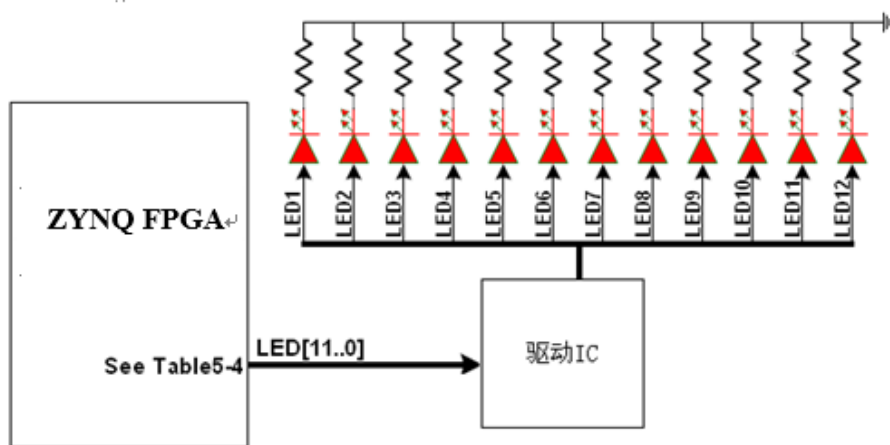


图 1-2 拨动开关与 FPGA 接口电路（2）

LED 灯与 FPGA 的接口电路如图 1-2 所示，当 FPGA 与其对应的端口为高电平时 LED 就会发光，反之 LED 灯灭。

四、 实验步骤：

下面将通过这个实验，向读者介绍 VIVADO 的项目文件的生成、综合、管脚分配以及

时序仿真等的操作过程。

## 1. 建立工程文件：

选择开始>程序>Xilinx Design Tools>Vivado 2015.4，运行 Vivado 软件。或者双击桌面上的 Vivado 2015.4 的图标运行 Vivado 软件，出现如图 1-3 所示界面。



图 1-3 Vivado 软件运行界面

选择软件中的菜单 File>New Project ...或者双击 Create New Project 图标新建一个工程。点击后如图 1-4 所示。点击图 1-4 中的 NEXT 进入工作目录。

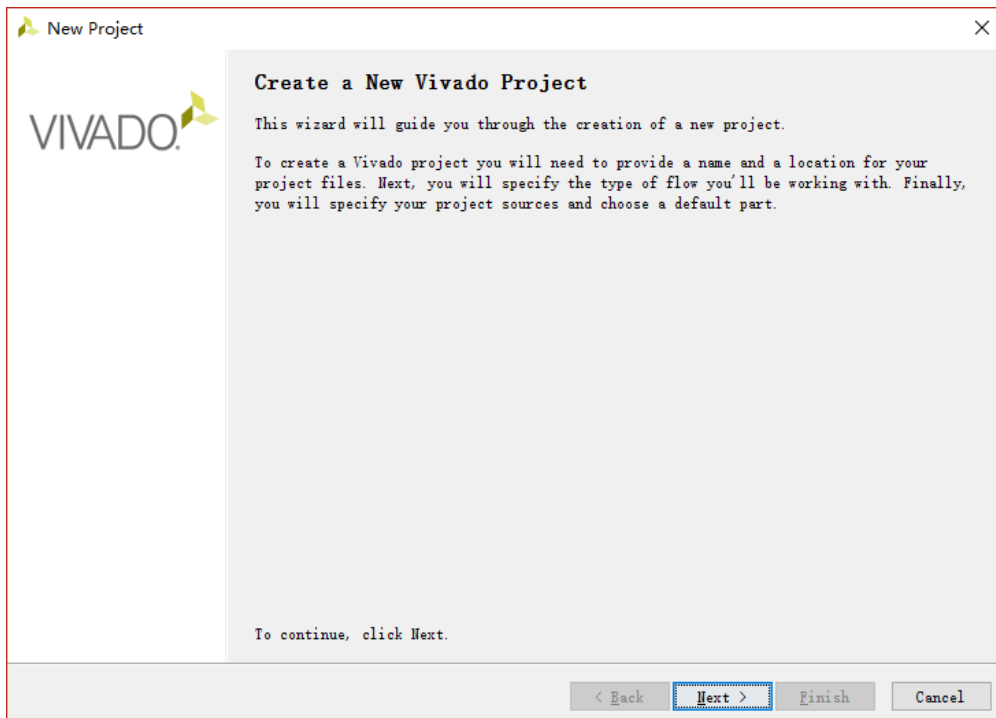


图 1-4 新建工程对话框

工程名的设定对话框如图 1-5 所示。第一个输入框 Project name 为工程名称输入框，用户可以自定义设定在此参考实验中工程名字命名为 EXP1，第二个输入框 project Location 为工程路径选择，用户可以通过点击此输入框后方的...选择工程路径如图 1-6 所示，选择工作路径之后单击 select。或者在第二个输入框 project Location 输入 D:/zynq\_vivado 工作路径来设定工程的目录，此路径不唯一，仅供参考，用户可以根据习惯自定义路径。选中路径之后，请勾选 create project subdirectory（创建工程子目录），勾选后软件会自动创建工程子目录，方便管理我们在此路径下创建的多工程。确认好工程名称和路径之后单击 NEXT。

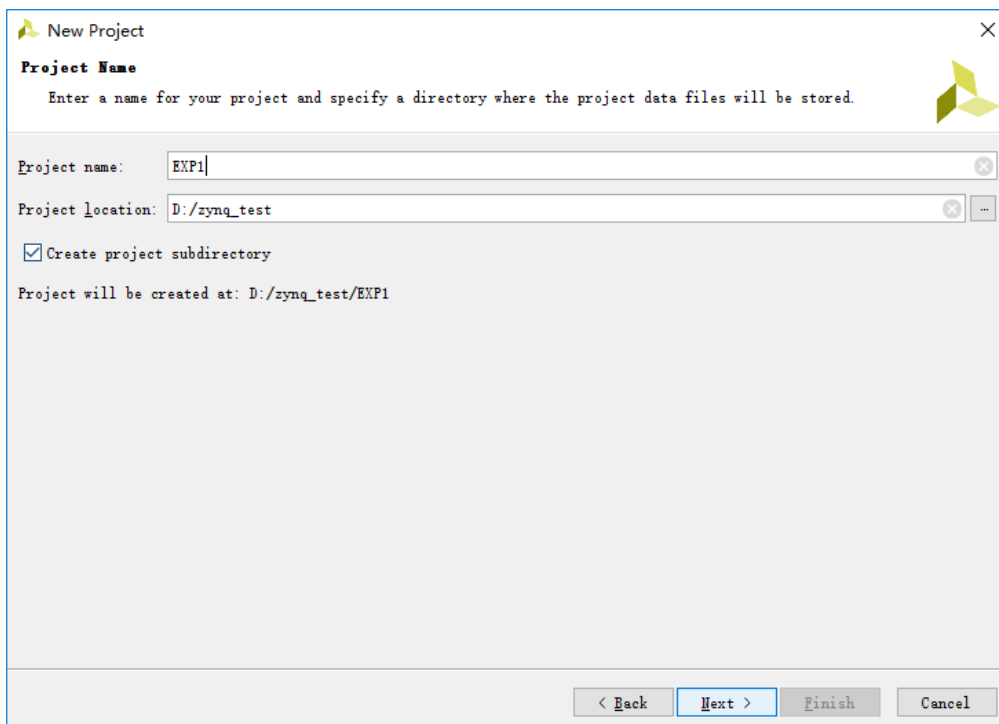


图 1-5 指定工程名称及工作目录

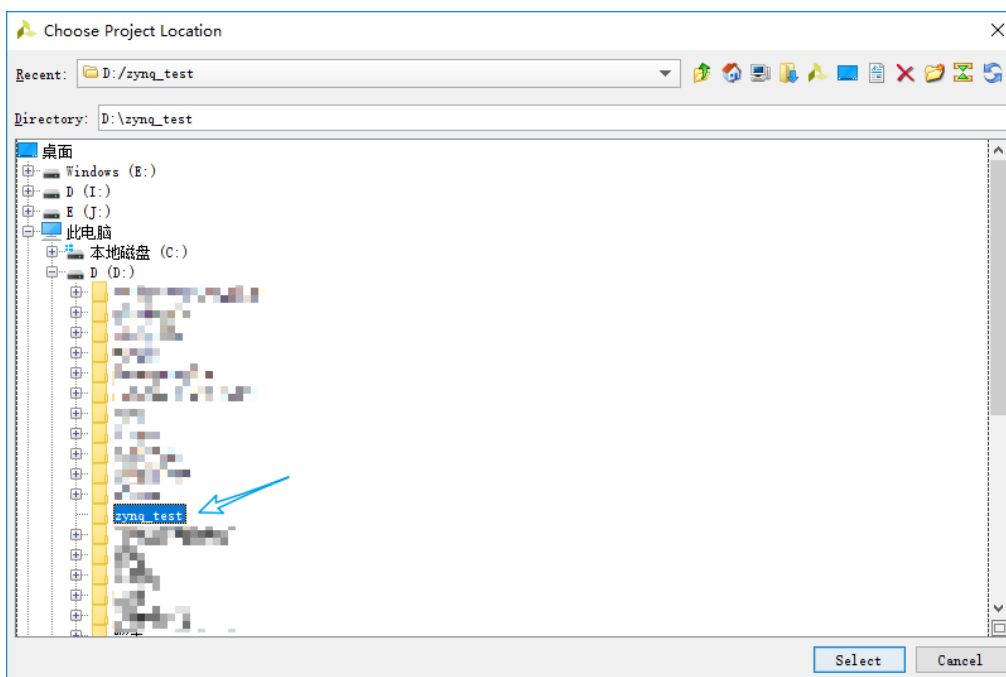


图 1-6 指定工作目录

单击 NEXT 后, 如图 1-7 所示, 这里按默认选项 RTL project 即可(务必勾选 Do not specify sources at this time)接下来点击 NEXT 进行器件选择对话框。



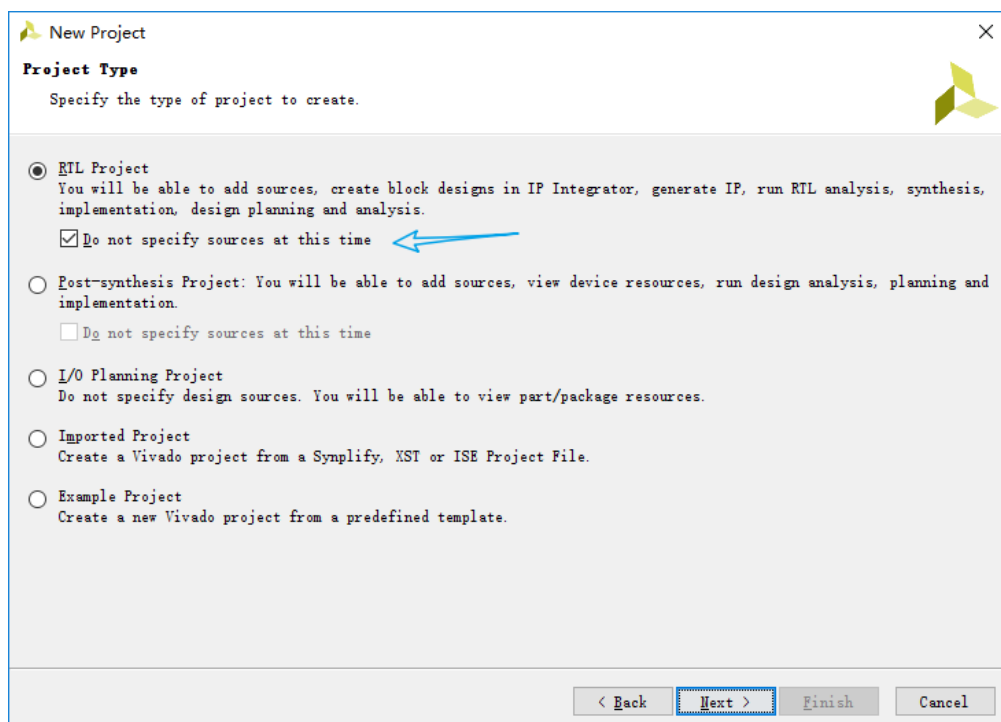


图 1-7 RTL Project

如图 1-8 所示。我们根据实验平台上的硬件选择 Zynq-7000 系列器件，芯片详细型号为 xc7z020clg484-2。用户可以根据不同的硬件设备型号进行选择。这里有两种快速锁定芯片方法：第一种在对话框的左上方的 Family 下拉菜单中选取 Zynq-7000，在中间右边的 Package 下拉菜单中选取 clg484，在左下方四个候选中选择 xc7z020clg484-2 如图 1-8 所示。

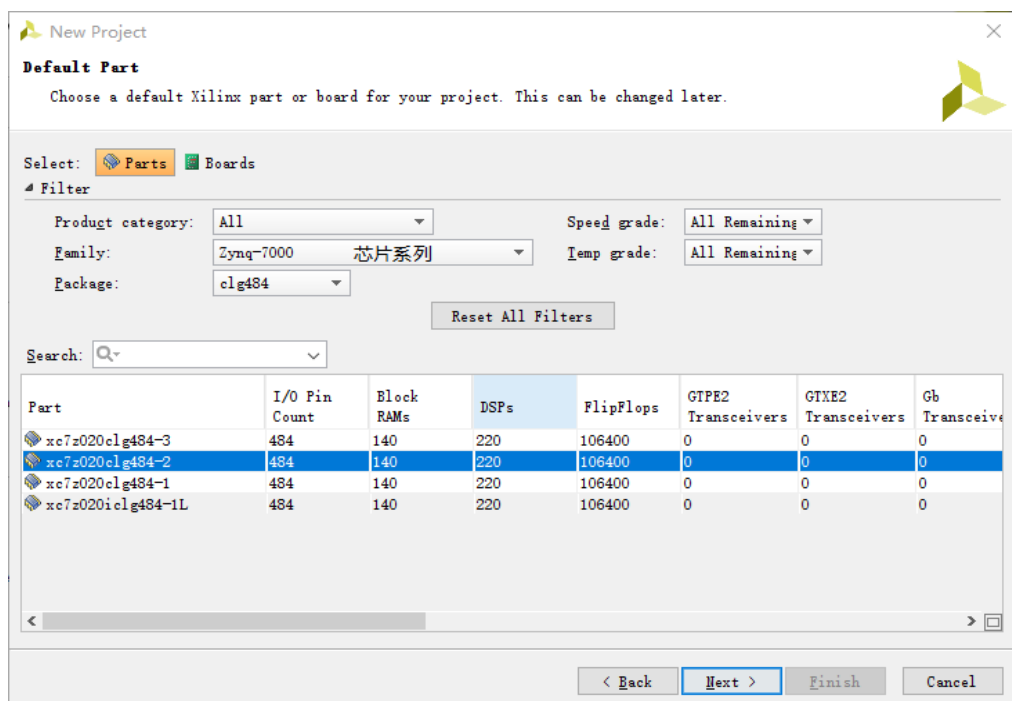


图 1-8 器件选择界面

第二种是在 search 一栏中输入 xc7z020clg484-2 如图 1-9 所示。选择正确的器件之后单击 NEXT。

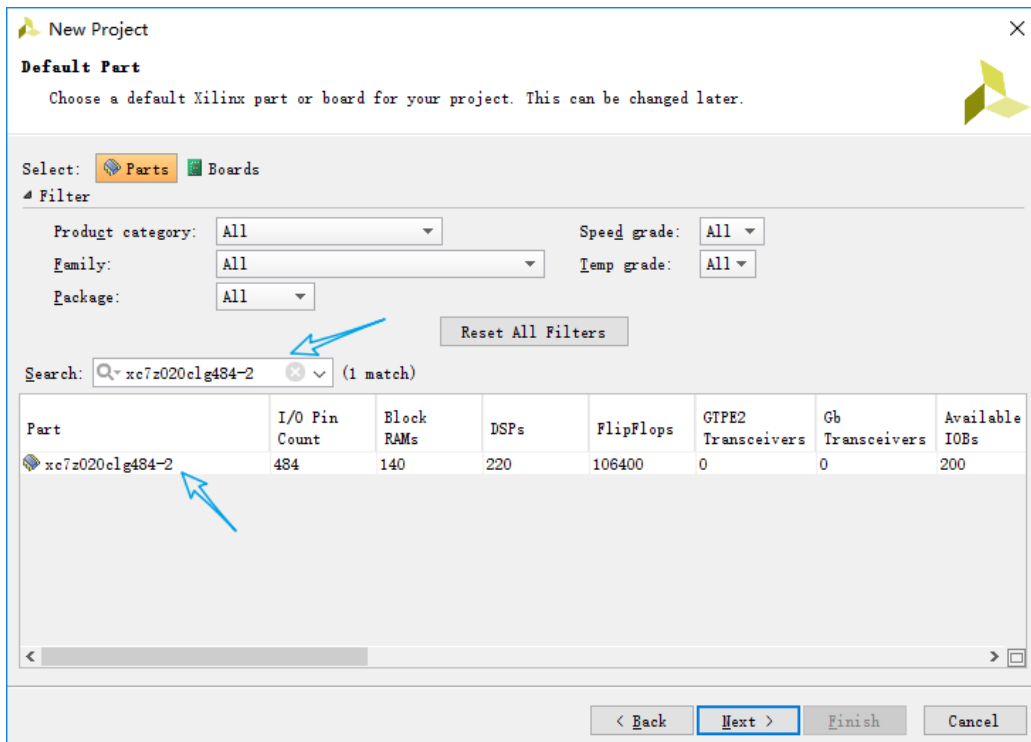


图 1-9 器件选择界面

如图 1-10 所示，是我们创建的工程信息概况信息，请仔细确认一遍信息无误之后，单击 FINISH 完成工程的创建。



图 1-10 工程信息概况

图 1-11 为，VIVADO 软件主界面各个功能选项的简单说明。详细使用技巧在后续工程中会以实操方式介绍。

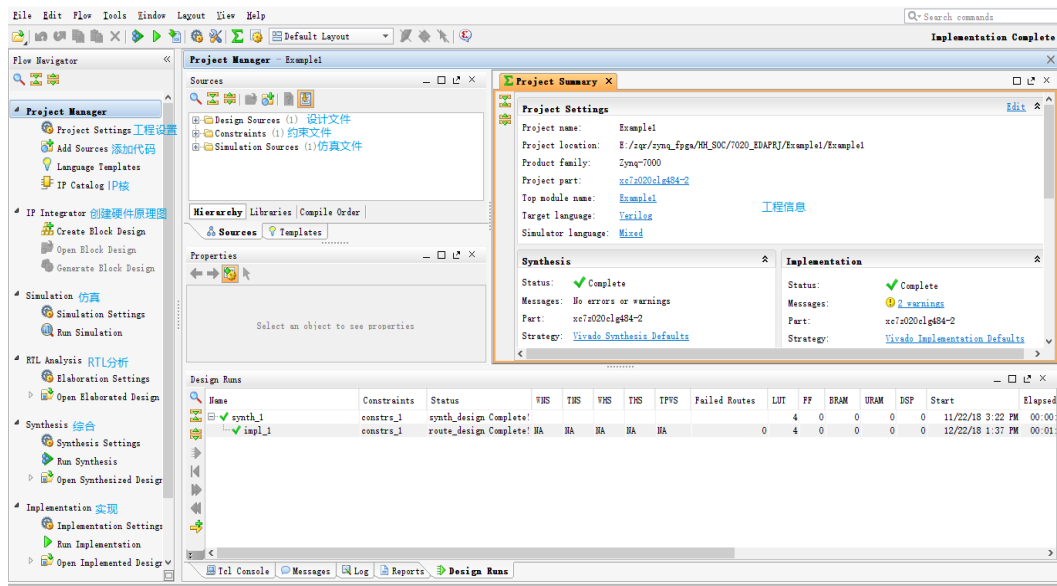


图 1-11 软件主页面图

## 2. 建立设计文件：

在设计工程中创建设计文件有两种方式，一是如图 1-12 所示单击左侧工具栏 Add Sources 进行创建。二是如图 1-13 所示单击 File 下拉菜单的 Add Sources...进行创建快捷键为 Ctrl+A。

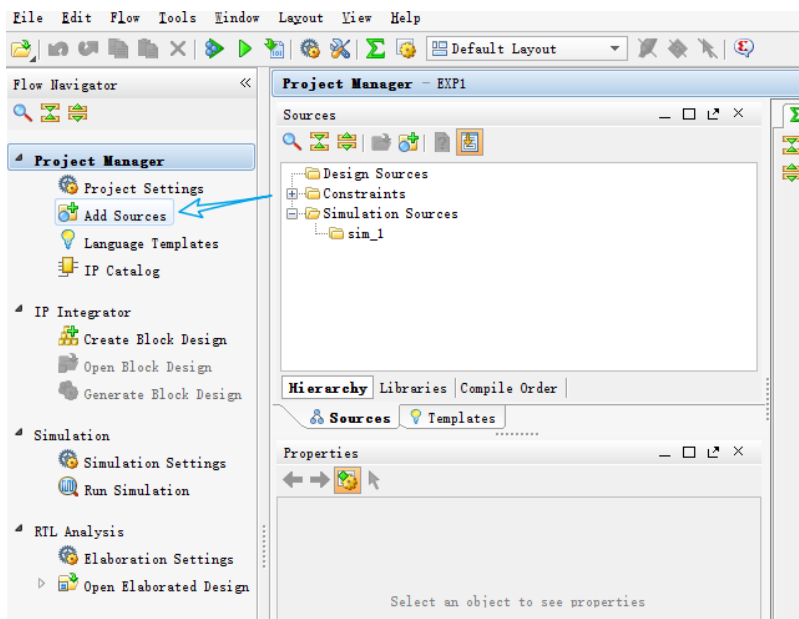


图 1-12

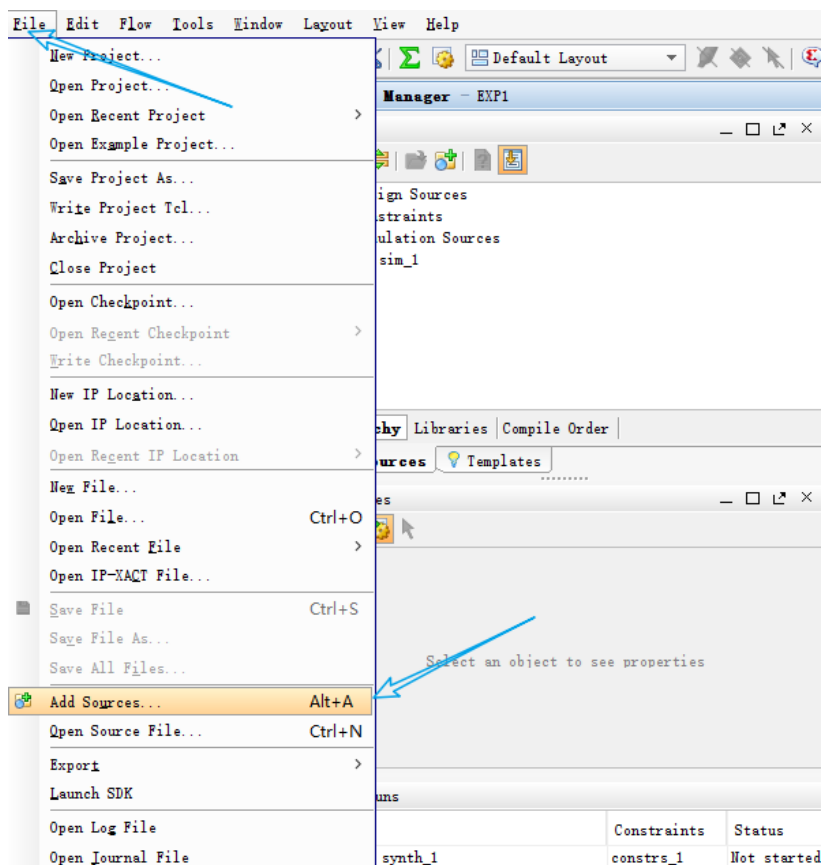


图 1-13

单击 Add Sources 出现图 1-14 所示的新建设计文件类型选择窗口。这里我们以建立 Verilog 设计文件为例进行说明，首先勾选 Add or create design sources 添加或创建设计文件-Verilog/VHDL等，单击 NEXT。创建约束文件、仿真文件等方法与之基本相同。

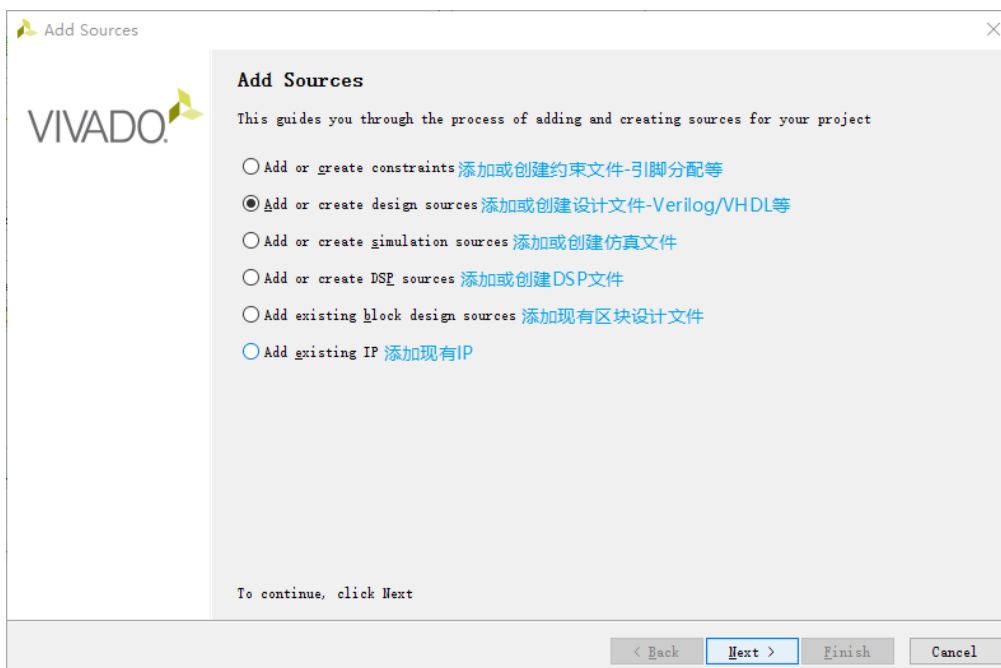


图 1-14 新建设计文件选择窗口

点击 Create File 创建一个三八译码器设计文件，如图 1-15 所示。



图 1-15 Vivado 创建设计文件对话框

点击 Create File 后会弹出如图 1-16 所示的窗口，在 File type 一栏中选择语言类型为 Verilog，在 File name 一栏中输入设计文件名称 EXP1（命名可以根据习惯自定义）。单击 OK。

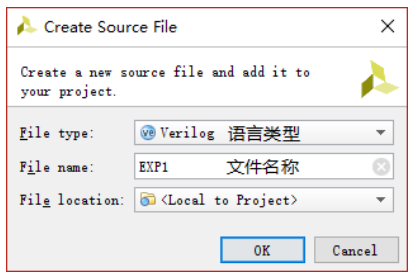


图 1-16 Create source file 对话框

单击 OK 后，如图 1-17 文件框内可以看见创建的 EXP1.v 文件，接下来点击 Finish。

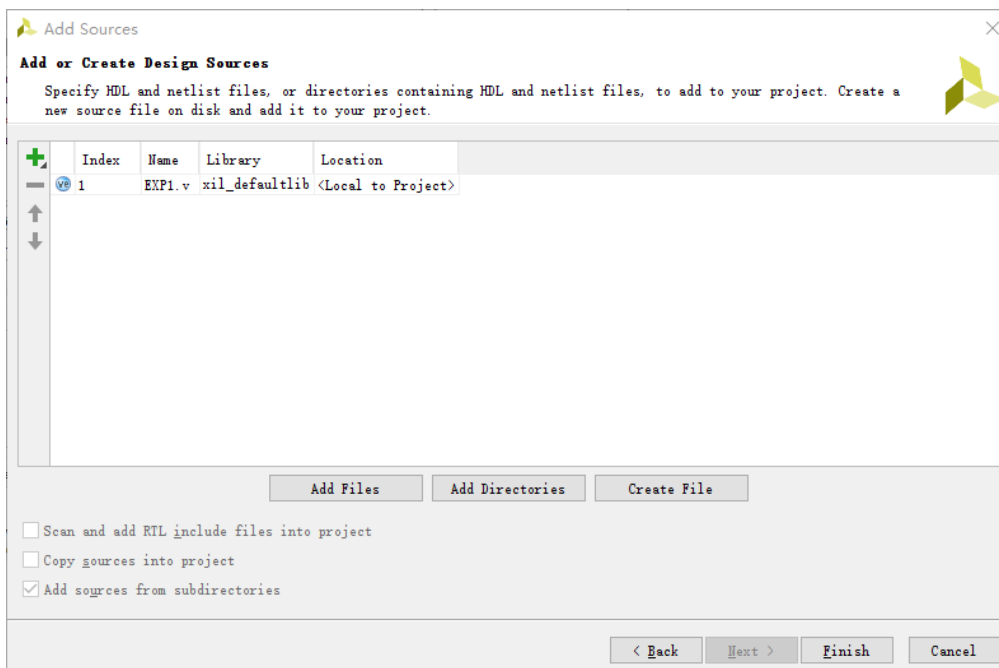


图 1-17 设计文件的端口定义

如图 1-18 所示，需要在此提示框内定义输入输出端口类型及其名称，左侧的“+”号表示添加端口“-”号表示删除。在 Port Name 一栏中输入端口名称，在 Direction 一栏中选择端口方向。在这个例子中，需要定义三个输入为 A、B、C，定义八个输出为 D0、D1、D2、D3、D4、D5、D6、D7。定义好端口之后单击 OK。

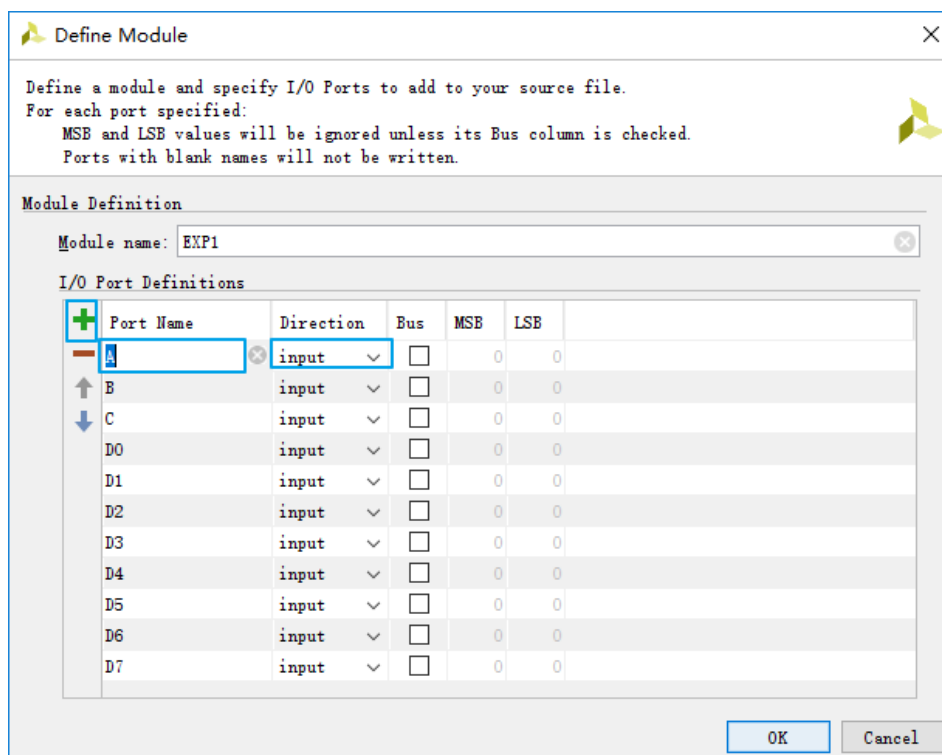


图 1-18 设计文件的端口定义

如图示 1-19，双击创建的设计文件 EXP1.v，可以在右侧预览到设计文件程序，可以看到 EXP1.v 文件中自动生成了我们刚刚添加的端口名称以及端口属性。接下来用户需根据 3-8 译码器的真值表在 EXP1.v 中补充 Verilog 设计语句。

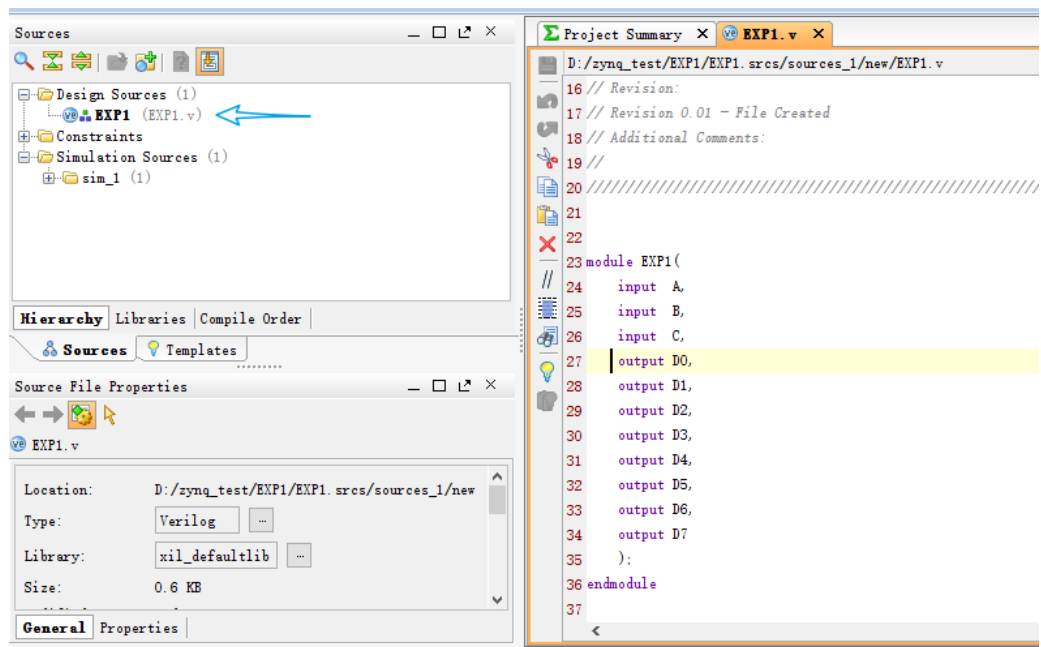


图 1-19

### 3. 对设计文件进行综合：

Vivado 综合，单击软件左侧工具栏 synthesis 下的 Run Synthesis。如图 1-20 所示。也可以单击软件上方工具栏相同的图标。

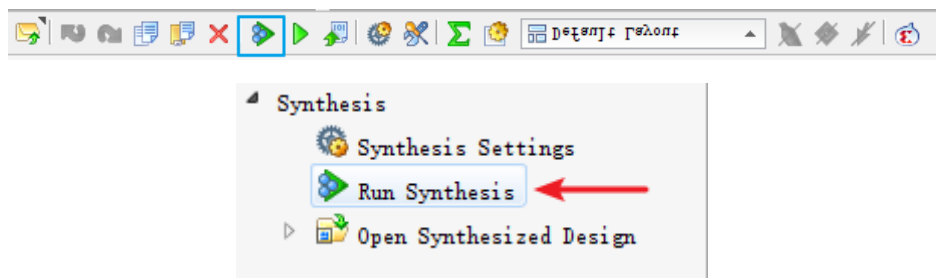


图 1-20 Vivado 综合

如图 1-21 在进行设计文件综合分析过程中，如果设计文件有错，在软件的下方信息提示栏中会提示错误的原因和位置，以便于使用者进行修改直到设计文件无错。

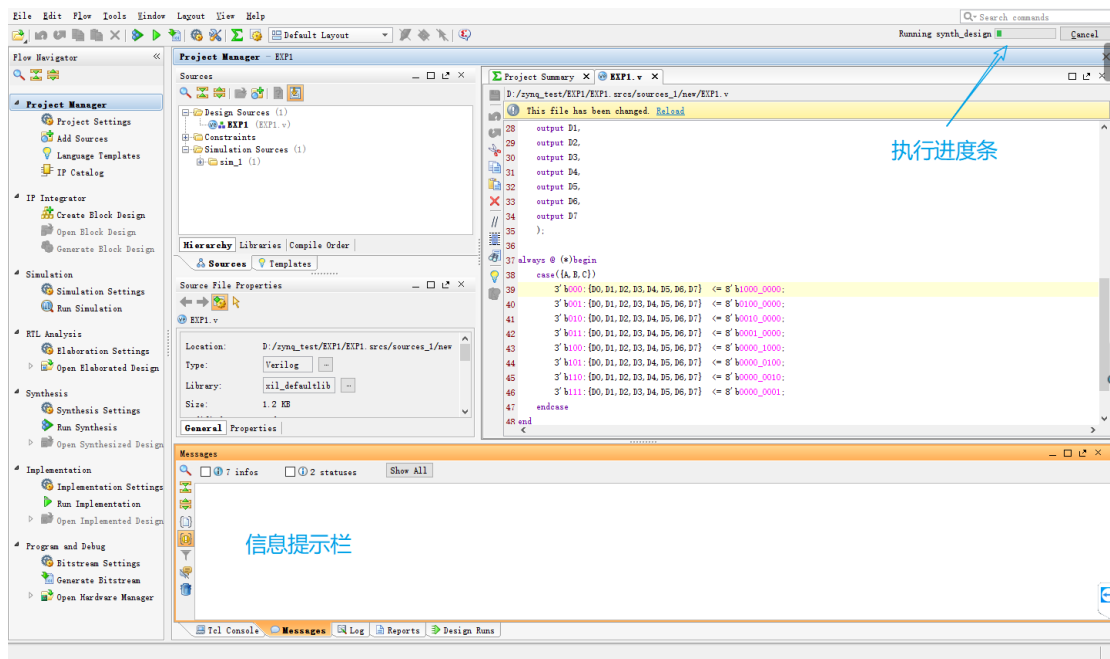


图 1-21 Vivado 综合

整个设计综合完成，软件会提示如图 1-22，选择是否 Run Implementation 或是 open synthesized design 或是 view reports。这里我们选择 view reports 单击 OK 即可。

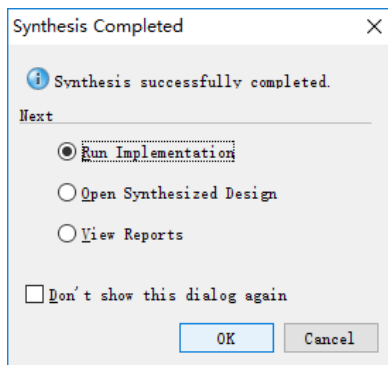


图 1-22 Vivado 综合结束

#### 4. 行为仿真：

点击 Add Sources（添加设计文件）-Add or create simulation sources（添加或创建仿真文件），如图 1-23 所示。



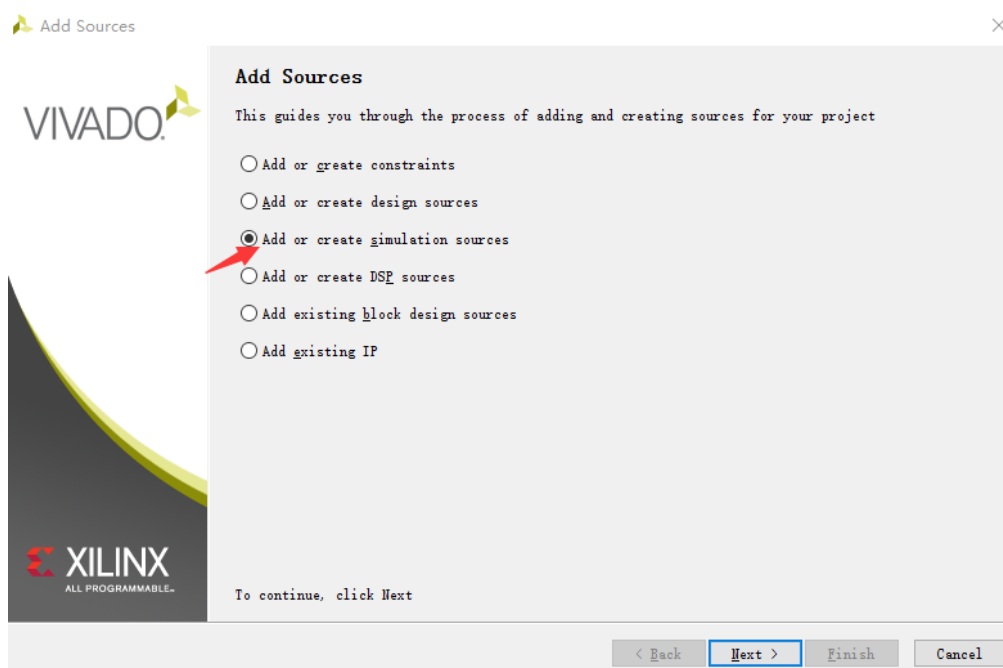


图 1-23 Vivado 界面

单击 Next 之后单击 Create File（生成文件）如图 1-24 所示。

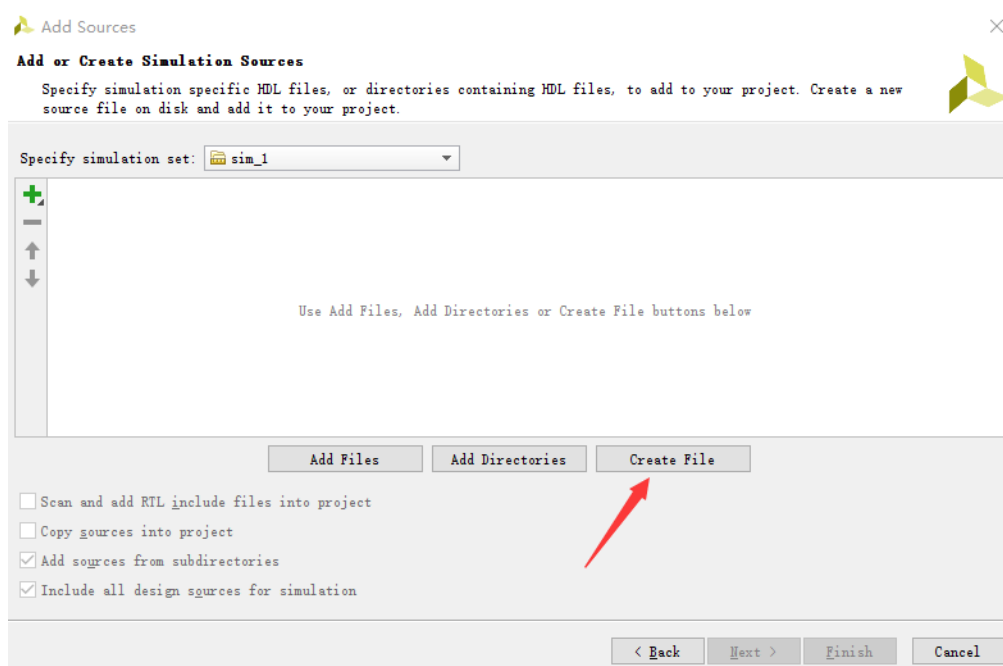


图 1-24 Vivado 界面

单击 Create File 之后会出现如图 1-25 所示的内容，这里需要给文件起一个名字，严谨的命名方式为“工程名\_TB/工程名\_tb”。文件类型选择 Verilog。之后单击 OK

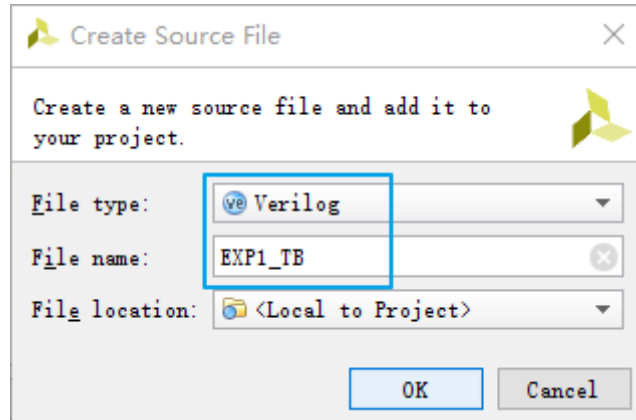


图 1-25 Vivado 界面

填写完毕后，单击 OK，单击 Finish，会提示添加端口信号，根据编写要求，我们不需要给测试文件头部添加端口，所以单击 Cancel 即可。如图 1-26。

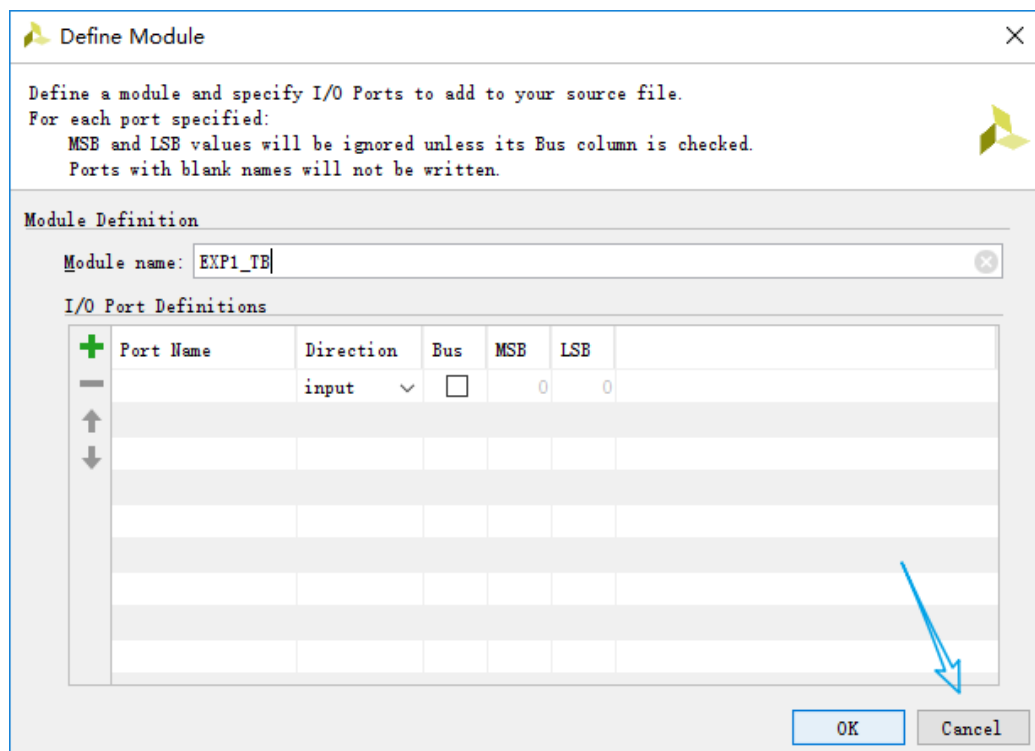


图 1-26 Vivado 界面

单击 Cancel 软件会提示如图 1-27 确认是否进行 Cancel 操作，点击 Yes 即可。

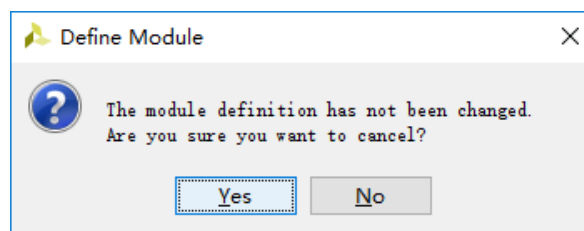


图 1-27 Vivado 界面

如图 1-28 此时在设计文件对话框内找到新建的测试 TB 文件，测试激励信号需要手动编写。按照标准编写方式对测试设计文件进行补充后，会出现 TB 设计文件对主工程文件 EXP.V 的一个调用如图 1-29，出现层级调用结构。如果没有出现如图所示的层级结构请检查 TB 文件中的例化部分是否正确。

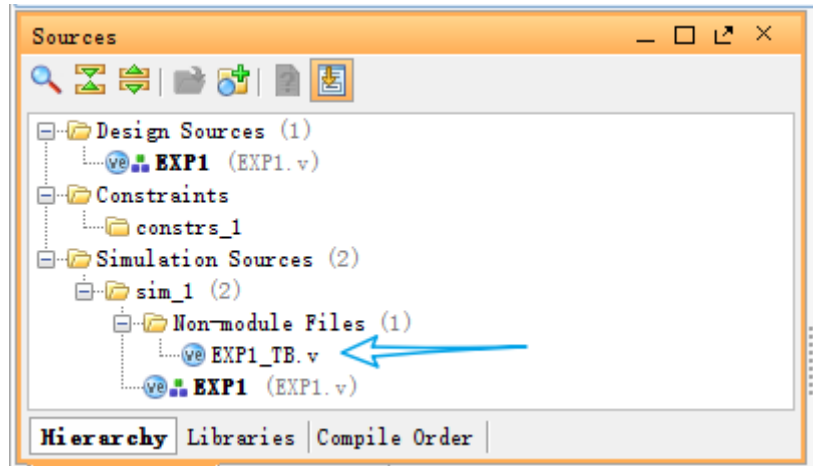


图 1-28 Vivado 界面

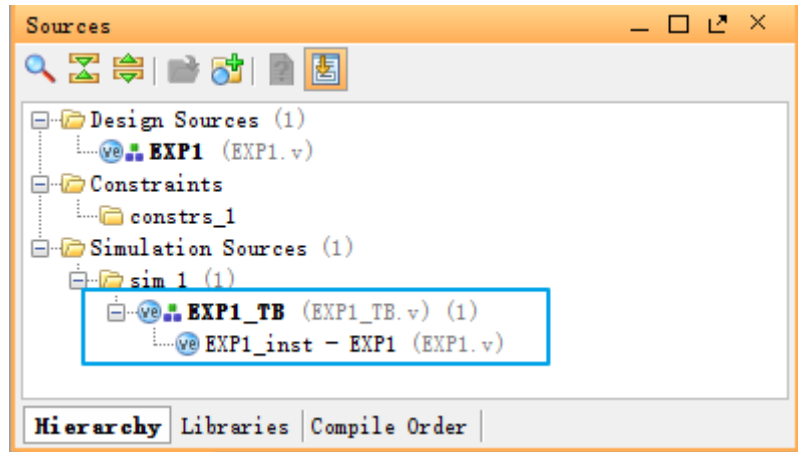


图 1-29 Vivado 界面

接下来点击的软件左侧操作栏中的 Run Simulation-Run Behavioral Simulation 如图 1-30

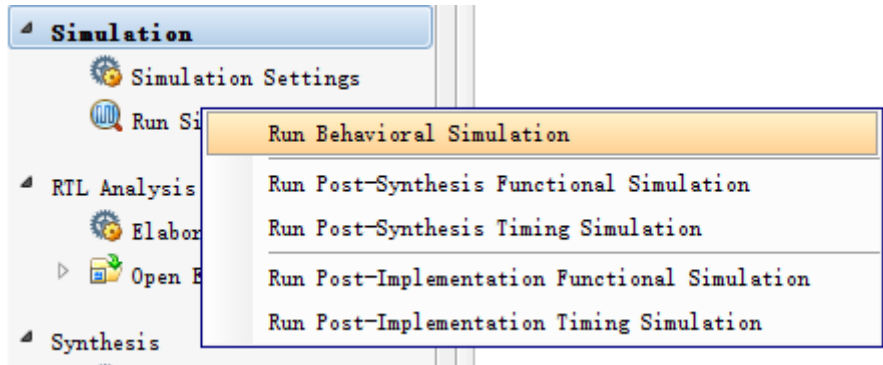


图 1-30 Vivado 界面

软件运行一会后即可才界面上观察仿真波形如图 1-31（波形出现初期请点击波形窗口栏左侧工具的箭头所致的 Zoom Fit 按钮进行全局观测，Zoom Fit 按钮上方的两个按钮分别是以图中黄线为基准的波形缩小和放大）。请对照 3-8 译码器设计的真值表观察输出波形信号是否达到预期。如果没有请检查设计。如果点击 zoom Fit 按钮后发现波形不完整，可能是仿真时间太短导致波形不完整。可以更改软件上方的仿真时间，更改仿真时间后请点击时间栏左侧的三角。如图 1-31。

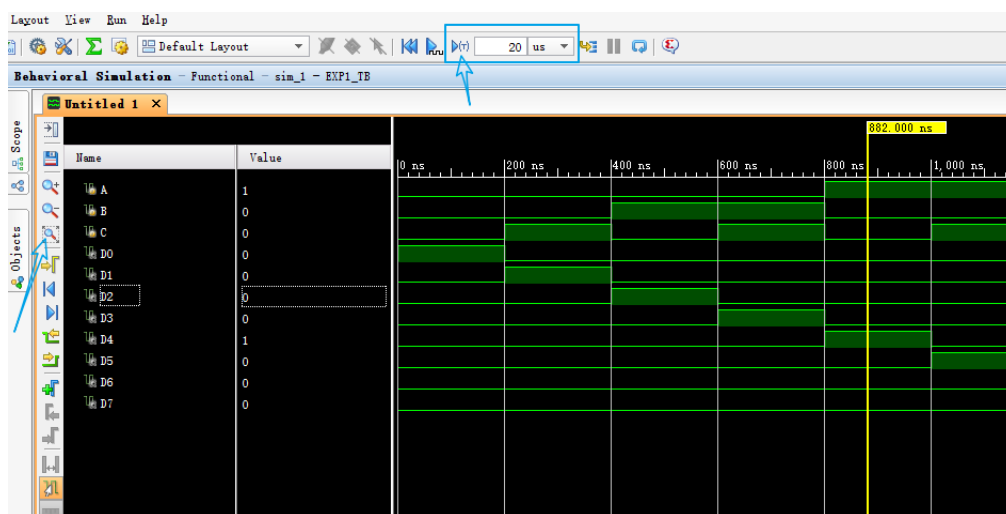


图 1-31 Vivado 仿真波形界面

## 5. 添加管脚约束：

管脚约束文件，即.XDC 文件，一般情况，生成后会在“工程名.srcs\constrs\_1\new”文件夹中。添加管脚约束有两种方式，一种直接加入写好的约束文件，另一种是综合后，添加管脚约束，推荐初学者使用第二种方式。

### 第一种：直接加入写好的约束文件：

将我们提供例程中 Example1\Example1.srcs\constrs\_1\new 文件夹复制到你所建立的工程中，Example1\Example1.srcs\constrs\_1\new 文件夹包含我们已经写好的.XDC 文件。

单击 Add Sourcess。

选择 Add or create constraints 然后单击 Next 如图 1-32



图 1-32 Vivado 界面

单击 Add Files 如图 1-33。

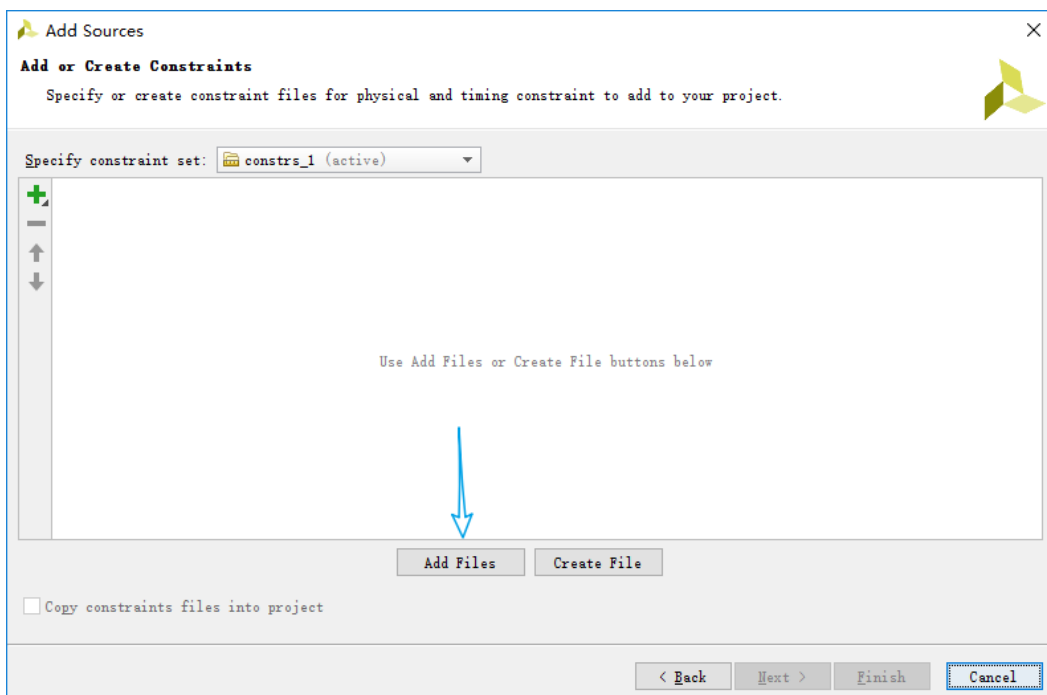


图 1-33 Vivado 界面

选中 Example1\Example1.srcs\constrs\_1\new 路径下的 Example1.xdc 文件, 然后点击 OK。再点击 Finish 完成约束文件的添加。

约束文件介绍与编写方法如下:

XDC 是 Xilinx Design Constraints 的简写, 其基础语法源自业界统一的约束规范 SDC。

XDC 的基本语法:

基本语法可以分为时钟约束、I/O 约束以及时序例外约束三大类, 这里我们只需要了解前两种约束即可。

以时钟端口为例约束编写格式如下：

首先设置时钟端口特性 IO 的电平标准为 1.8V

例：set\_property IOSTANDARD LVCMOS18 [get\_ports {Clk}]

解析：“set\_property IOSTANDARD”为设置端口特性的标准格式，无须改动。

“LVCMOS18”代表的是设置端口电平为 1.8V，这里的电平参数不唯一可以更具设计要求修改为 LVCMOS33 或 LVCMOS25。

“[get\_ports {Clk}]”指定被设置的端口，其中[get\_ports{ }]为标准格式，{ }内只需要填写被设置的端口名称即可，这里注意端口名称区分大写小，请和设计文件的端口名称一致。

接下来设置端口引脚

例：set\_property PACKAGE\_PIN Y8 [get\_ports {Clk}]

解析：“set\_property PACKAGE\_PIN Y8”前半部分 set\_property PACKAGE\_为标准格式，我们只需要更改\_后面的 PIN Y8 即可，根据我们的硬件手册或实验指导书尾页可以找的时钟端口在硬件中连接到核心芯片的哪个管脚上。我们根据管脚的不同进行修改即可。

“[get\_ports {Clk}]”指定被设置的端口，其中[get\_ports{ }]为标准格式，{ }内只需要填写被设置的端口名称即可，这里注意端口名称区分大写小，请和设计文件的端口名称一致。

上面的例子为单比特信号的设置电平标准和引脚，多比特信号设置大体相似只需要修改{}内容即可。

例{data[0]}，表示对 data[0]为进行设置。

其他约束需要视情况而定，如果在工程运行中出现错误，在软件的运行报告中会给出约束建议，我们只需要根据建议在 XDC 文件中添加对应约束即可。

下表是管教对照表，可以根据 Example1.xdc 文件对照一下。

端口名	使用模块信号	对应 FPGA 管脚	说 明
A	拨动开关 K1	V10	译码器的 三位输入
B	拨动开关 K2	AA9	
C	拨动开关 K3	W11	

D0	LED 灯 LED1	C15	译码器的 八位输出
D1	LED 灯 LED2	E15	
D2	LED 灯 LED3	B16	
D3	LED 灯 LED4	A16	
D4	LED 灯 LED5	G15	
D5	LED 灯 LED6	F16	
D6	LED 灯 LED7	C18	
D7	LED 灯 LED8	D16	

**第二种：综合后，添加管脚约束：**

综合工程，如图 1-34 所示。点击图示的绿色三角，也可以单击左侧菜单栏 Run Synthesis（快捷键 F11）。

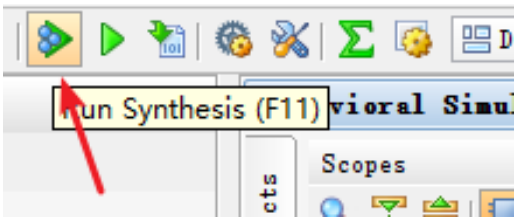


图 1-34 Vivado 综合界面

Run Synthesis 结束时软件提示如下图 1-35，请选择 Open Synthesized Design 点击 OK。

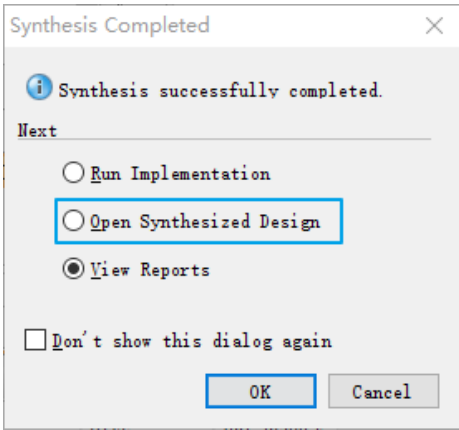


图 1-35 Vivado 综合界面

在 vivado 软件 Window 下拉列表找 I/O Ports 并单击，如图 1-36

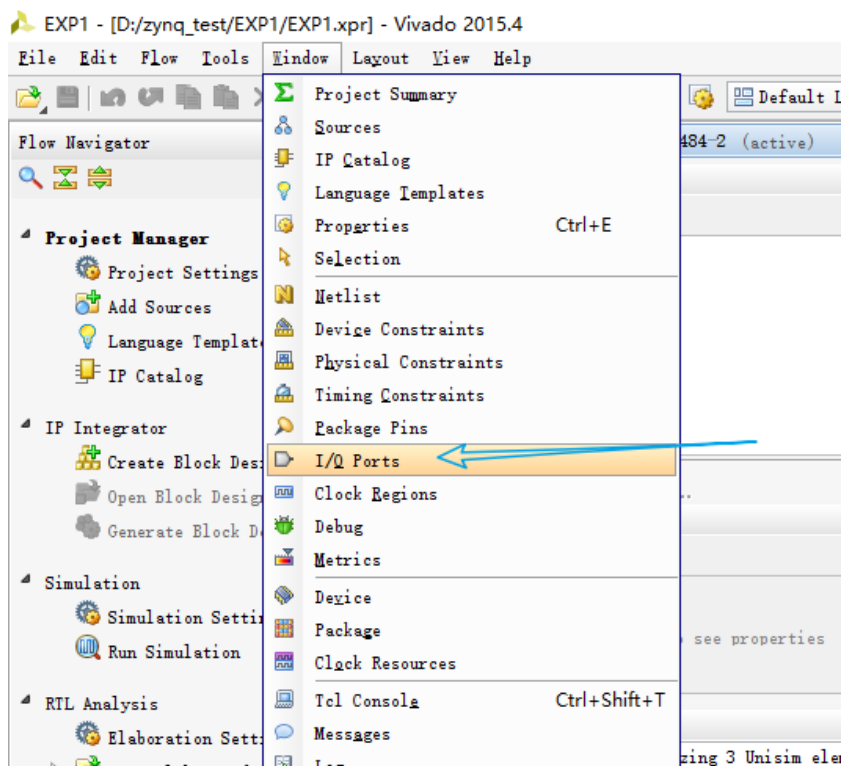


图 1-36 Vivado 综合界面

点击 I/O Ports，进行管脚分配以及电平设置（如图 1-37）在 site 栏内编辑硬件端口号，I/O Std 栏选择电压建议选择 LVC MOS18，CTRL+S 保存约束文件，给.XDC 文件命名，生成 XDC 文件，并保存到 Miz\_sys\Miz\_sys.srcs\constrs\_1。constrs\_1 文件夹是自己创建的。

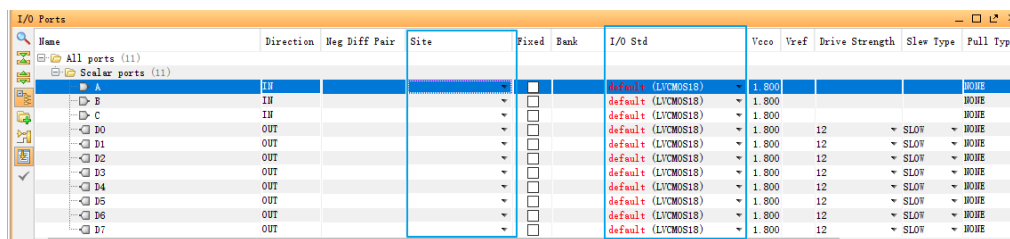


图 1-37 Vivado 综合界面

到此工程管脚约束文件创建完毕。

## 6. 综合并且产生 Bit 文件：

单击综合 > 单击执行 > 单击产生 Bit 文件(如图 1-38 所示)

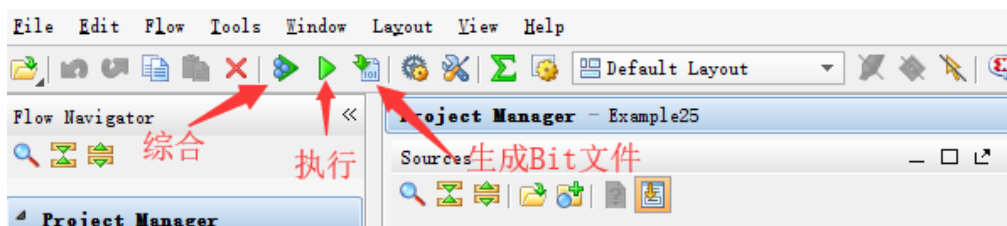




图 1-38 Vivado 综合界面

## 7. 下载程序:

给实验箱通电, 并且连接下载器(如图 1-39 请注意下载器线的方向和信号提示灯)



图 1-39 下载器和 JTAG 方向

单击左侧面板 OpenTarget 然后单击 Auto Connect。如图 1-40。

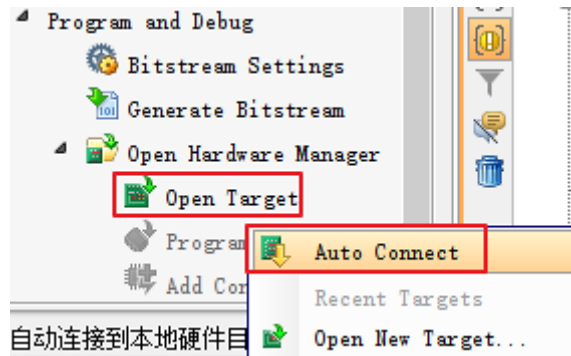


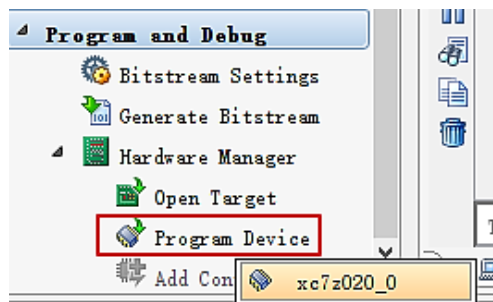
图 1-40 Vivado 界面

硬件连接成功后如图 1-41 所示

Name	Status
localhost (1)	Connect
xilinx_tcf/Digilent/210249854705 (1)	Open
xc7z020_0 (1)	Not pro
XADC (System Monitor)	

图 1-41 Vivado 界面

如图 1-42 单击 Program Device 然后选择 XC7Z020。也可以从顶部单击 Program device



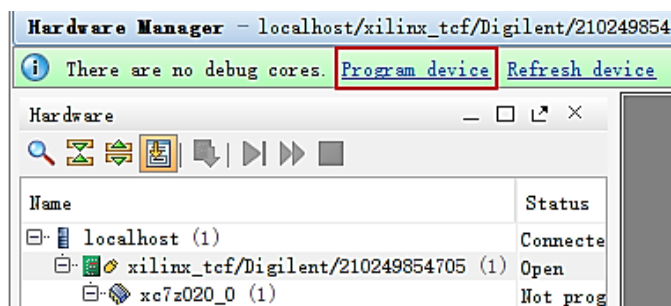


图 1-42 Vivado 界面

如图 1-43，弹出的对话框中有我们下载的 Bit 文件。

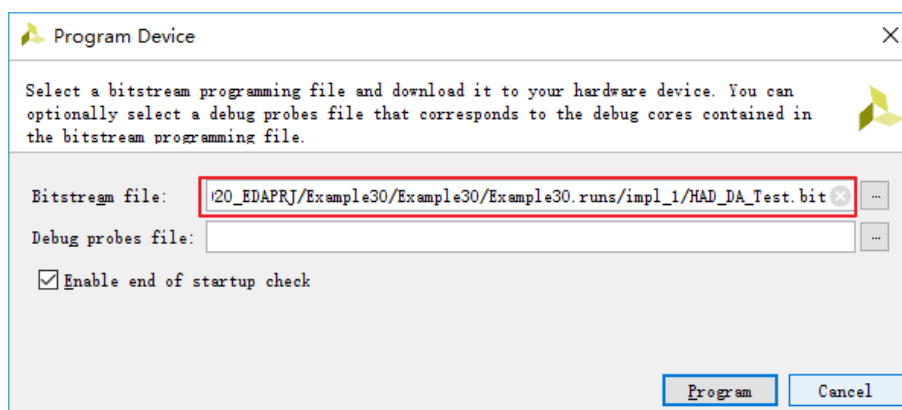
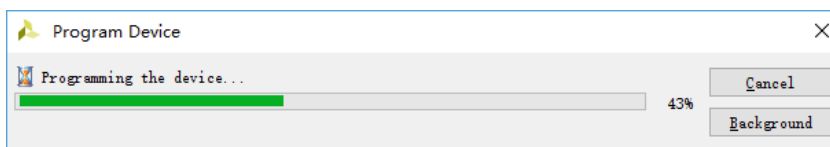


图 1-43 Vivado 界面

下载过程



## 五、 实验现象与结果：

文件加载到目标器件后，拨动拨动开关，LED 灯会按表 1-1 所示的真值表对应的点亮。因为 LED 灯模块的后四个灯 LED9-LED12 没有被使用，而 VIVADO 软件默认设置未使用的 IO 为高阻三态，所以后四个灯 LED9-LED12 一直常亮。

## 六、 实验报告：

1. 进一步熟悉和理解 VIVADO 软件的使用方法。
2. 绘出仿真波形，并作说明。
3. 将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。

## 实验二 基于 VERILOG 格雷码编码器的设计

### 一、 实验目的：

1. 了解格雷码变换的原理。
2. 进一步熟悉 VIVADO 软件的使用方法和 VERILOG 输入的全过程。
3. 进一步掌握实验系统的使用。

### 二、 实验原理：

格雷（Gray）码是一种可靠性编码，在数字系统中有着广泛的应用。其特点是任意两个相邻的代码中仅有一位二进制数不同，因而在数码的递增和递减运算过程中不易出现差错。但是格雷码是一种无权码，要想正确而简单的和二进制码进行转换，必须找出其规律。

根据组合逻辑电路的分析方法，先列出其真值表再通过卡诺图化简，可以很快的找出格雷码与二进制码之间的逻辑关系。其转换规律为：高位同，从高到低看异同，异出‘1’，同出‘0’。也就是将二进制码转换成格雷码时，高位是完全相同的，下一位格雷码是‘1’还是‘0’，完全是相邻两位二进制码的“异”还是“同”来决定。下面举一个简单的例子加以说明。

假如要把二进制码 10110110 转换成格雷码，则可以通过下面的方法来完成，方法如图 2-1。



图 2-1 格雷码变换示意图

因此，变换出来的格雷码为 11101101

### 三、 实验内容：

本实验要求完成的任务是变换 12 位二进制码到 12 位的格雷码。实验中用 12 位拨动开关模块的 K1~K12 表示 12 位二进制输入，用 LED 模块的 LED1~LED12 来表示转换的实验结果十二位格雷码。实验 LED 亮表示对应的位为‘1’，LED 灭表示对应的位为‘0’。通过输入不同的值来观察输入的结果与实验原理中的转换规则是否一致。实验箱中的拨动开关、与 FPGA 的接口电路，LED 灯与 FPGA 的接口电路以及拨动开关、LED 与 FPGA 的管脚连接

在实验一中都做了详细说明，这里不在赘述。

四、实验步骤：

- 1. 打开 VIVADO 软件，新建一个工程。
- 2. 建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。
- 3. 按照实验原理和自己的想法，在 VERILOG 编辑窗口编写 VERILOG 程序。
- 4. 编写完 VERILOG 程序后，保存起来。方法同实验一。
- 5. 对自己编写的 VERILOG 程序进行综合并仿真，对程序的错误进行修改。
- 6. 综合仿真无误后，依照拨动开关、LED 与 FPGA 的管脚连接表（表 2-1）或参照附录进行管脚分配，表 2-1 是示例程序的管脚分配表。分配完成后，再进行全综合一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
K1	拨动开关 K1	V10	格雷编码器的数据输入
K2	拨动开关 K2	AA9	
K3	拨动开关 K3	W11	
K4	拨动开关 K4	Y11	
K5	拨动开关 K5	AB10	
K6	拨动开关 K6	AA11	
K7	拨动开关 K7	V12	
K8	拨动开关 K8	U10	
K9	拨动开关 K9	J20	
K10	拨动开关 K10	J18	
K11	拨动开关 K11	K16	
K12	拨动开关 K12	J15	
D1	LED 灯 LED1	C15	格雷编码器的编码输出
D2	LED 灯 LED2	E15	
D3	LED 灯 LED3	B16	
D4	LED 灯 LED4	A16	
D5	LED 灯 LED5	G15	
D6	LED 灯 LED6	F16	
D7	LED 灯 LED7	C18	
D8	LED 灯 LED8	D16	
D9	LED 灯 LED9	G17	
D10	LED 灯 LED10	B19	
D11	LED 灯 LED11	A18	
D12	LED 灯 LED12	D18	

表 2-1 端口管脚分配表

- 7. 用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

## 五、 实验现象与结果：

以设计的参考示例为例，当设计文件加载到目标器件后，拨动拨动开关，LED 会按照实验原理中的格雷码输入一一对应的亮或者灭。

## 六、 实验报告：

1. 绘出仿真波形，并作说明。
2. 进一步熟悉 VIVADO 软件。
3. 将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。

# 实验三 含异步清零和同步使能的加法计数器

## 一、 实验目的：

1. 了解二进制计数器的工作原理。
2. 进一步熟悉 VIVADO 软件的使用方法和 VERILOG 输入。
3. 时钟在编程过程中的作用。

## 二、实验原理：

二进制计数器中应用最多、功能最全的计数器之一，含异步清零和同步使能的加法计数器的具体工作过程如下：

在时钟上升沿的情况下，检测使能端是否允许计数，如果允许计数（定义使能端高电平有效）则开始计数，否则一直检测使能端信号。在计数过程中再检测复位信号是否有效（低电平有效），当复位信号起作用时，使计数值清零，继续进行检测和计数。其工作时序如图 3-1 所示：

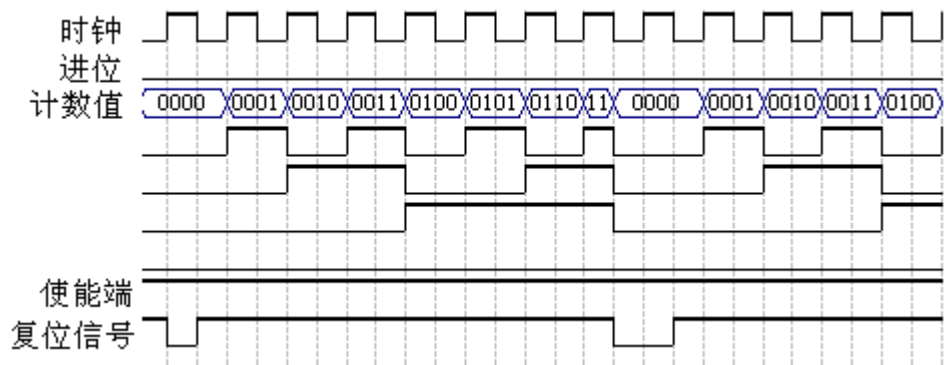


图 3-1 计数器的工作时序

## 三、实验内容：

本实验要求完成的任务是在时钟信号的作用下，通过使能端和复位信号来完成加法计数器的计数。实验中时钟信号使用数字时钟源模块的 1HZ 信号，用一位拨动开关 K1 表示使能端信号，用复位开关 S1 表示复位信号，用 LED 模块的 LED1~LED11 来表示计数的二进制结果。实验 LED 亮表示对应的位为 ‘1’，LED 灭表示对应的位为 ‘0’。通过输入不同的值模拟计数器的工作时序，观察计数的结果。实验箱中的拨动开关、与 FPGA 的接口电路，LED 灯与 FPGA 的接口电路以及拨动开关、LED 与 FPGA 的管脚连接在实验一中都做了详细说明，这里不在赘述。

数字时钟信号模块的电路原理如图 3-2 所示，表 3-1 是其时钟输出与 FPGA 的管脚连接表。



图 3-2 数字时钟信号模块电路原理

信号名称	对应 FPGA 管脚名	说明
DIGITAL-CLK	Y8	数字时钟信号送至 FPGA 的 Y8

表 3-1 数字时钟输出与 FPGA 的管脚连接表

按键开关模块的电路原理如图 3-3 所示，表 3-2 是按键开关的输出与 FPGA 的管脚连接表。

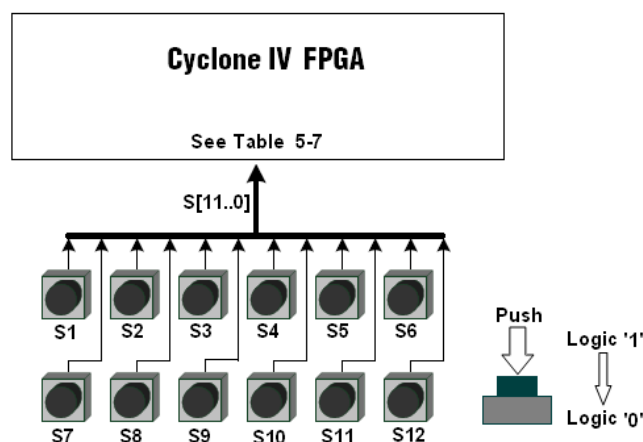


图 3-3 按键开关模块电路原理

信号名称	FPGA I/O 名称	功能说明
S[0]	P15	‘S1’ Switch
S[1]	N18	‘S2’ Switch
S[2]	P18	‘S3’ Switch
S[3]	R16	‘S4’ Switch
S[4]	T17	‘S5’ Switch
S[5]	M20	‘S6’ Switch
S[6]	K18	‘S7’ Switch
S[7]	K21	‘S8’ Switch
S[8]	K20	‘S9’ Switch
S[9]	L19	‘S10’ Switch
S[10]	M17	‘S11’ Switch
S[11]	M16	‘S12’ Switch

表 3-2 按键开关与 FPGA 的管脚连接表

#### 四、 实验步骤:

1. 打开 VIVADO 软件，新建一个工程。
2. 建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。
3. 按照实验原理和自己的想法，在 VERILOG 编辑窗口编写 VERILOG 程序。
4. 编写完 VERILOG 程序后，保存起来。方法同实验一。
5. 对自己编写的 VERILOG 程序进行综合并仿真，对程序的错误进行修改。
6. 综合无误后，依照时钟，拨动开关，按键，LED 与 FPGA 的管脚连接表参照表 3-3 分配。分配完成后，再进行全综合一次，以使管脚分配生效。
7. 用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

#### 五、 实验现象与结果:

以设计的参考示例为例，当设计文件加载到目标器件后，将数字信号源的时钟选择为 1HZ，使拨动开关 K1 置为高电平（使拨动开关向上），四位 LED 会按照实验原理中依次被点亮，当加法器加到 9 时，LED12（进位信号）被点亮。当复位键（按键开关的 S1 键）按下后，计数被清零。如果拨动开关 K1 置为低电平（拨动开关向下）则加法器不工作。

#### 六、 实验报告:

1. 绘出仿真波形，并作说明。
2. 写出在 VERILOG 编程过程中需要说明的规则。
3. 将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。
4. 改变时钟频率，看实验现象会有什么改变，试解释这一现象。



## 实验四 八位七段数码管动态显示电路的设计

### 一、 实验目的：

1. 了解数码管的工作原理。
2. 学习七段数码管显示译码器的设计。
3. 学习 VERILOG 的 CASE 语句及多层次设计方法。

### 二、 实验原理：

七段数码管是电子开发过程中常用的输出显示设备。在实验系统中使用的是两个四位一体、共阴极型七段数码管。其单个静态数码管如下图 4-1 所示。

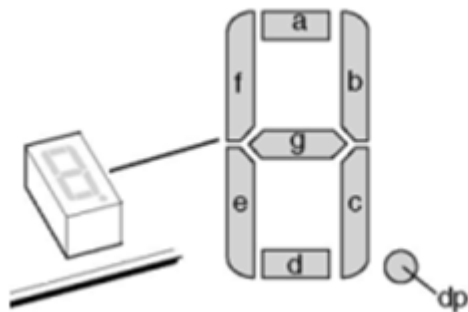


图 4-1 静态七段数码管

由于七段数码管公共端连接到 GND（共阴极型），当数码管中的那一个段被输入高电平，则相应的这一段被点亮。反之则不亮。四位一体的七段数码管在单个静态数码管的基础上加入了用于选择哪一位数码管的位选信号端口。八个数码管的 a、b、c、d、e、f、g、h、dp 都连在了一起，8 个数码管分别由各自的位选信号来控制，被选通的数码管显示数据，其余关闭。

### 三、 实验内容：

本实验要求完成的任务是在时钟信号的作用下，通过输入的键值在数码管上显示相应的键值。在实验中时，数字时钟选择 1KHZ 作为扫描时钟，用四个拨动开关做为输入，当四个拨动开关置为一个二进制数时，在数码管上显示其十六进制的值。实验箱中的拨动开关与 FPGA 的接口电路，以及拨动开关 FPGA 的管脚连接见下图表。

数码管显示模块的电路原理如图 4-2 所示，表 4-1 是其数码管的输入与 FPGA 的管脚连接表。

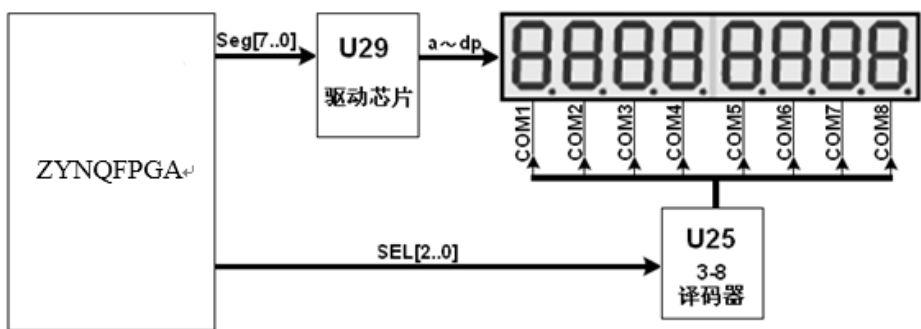


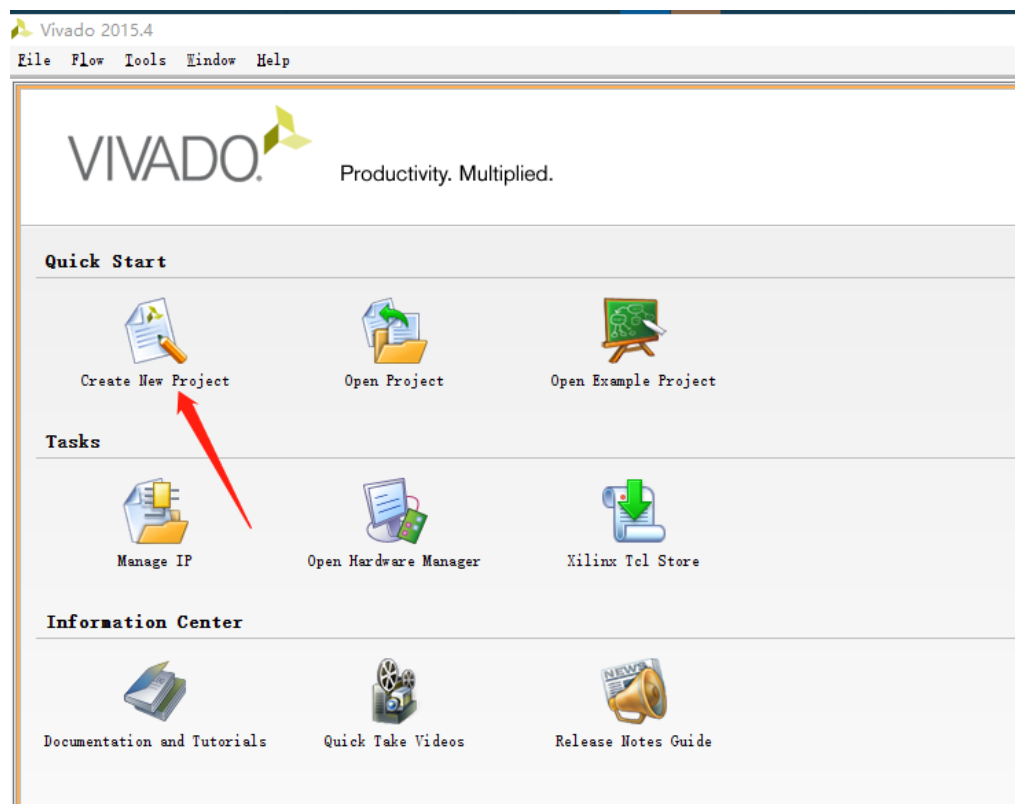
图 4-2 数字时钟信号模块电路原理

信号名称	FPGA I/O 名称	功能说明
Seg[0]	P16	7-Seg display “a”
Seg[1]	P17	7-Seg display “b”
Seg[2]	N17	7-Seg display “c”
Seg[3]	N15	7-Seg display “d”
Seg[4]	M15	7-Seg display “e”
Seg[5]	L17	7-Seg display “f”
Seg[6]	L18	7-Seg display “g”
Seg[7]	K19	7-Seg display “dp”
SEL[0]	L22	7-Seg COM port setcle
SEL[1]	P21	
SEL[2]	N20	

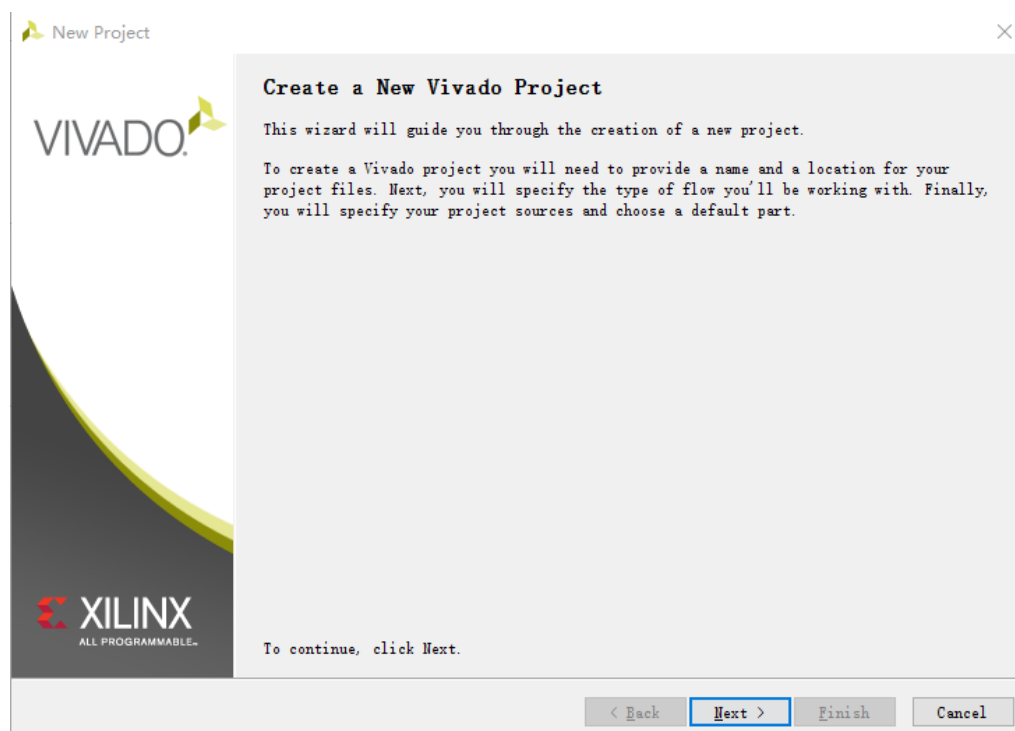
表 4-1 数码管与 FPGA 的管脚连接表

#### 四、 实验步骤：

1. 打开 VIVADO 软件，新建一个工程。双击 vivado 软件后点击 create new project

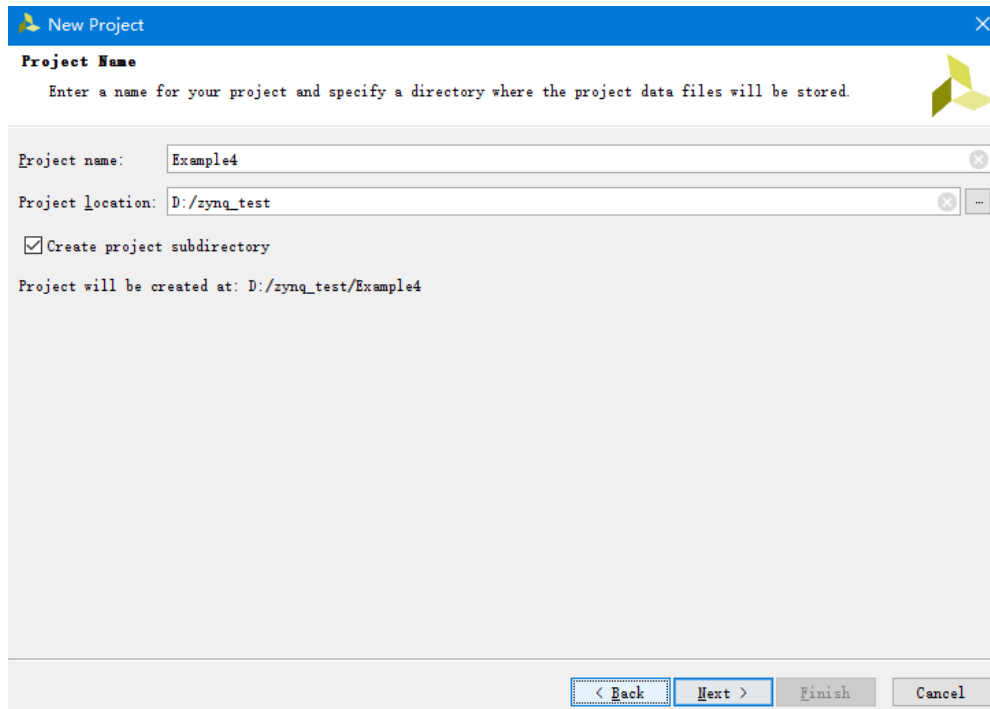


单击 next



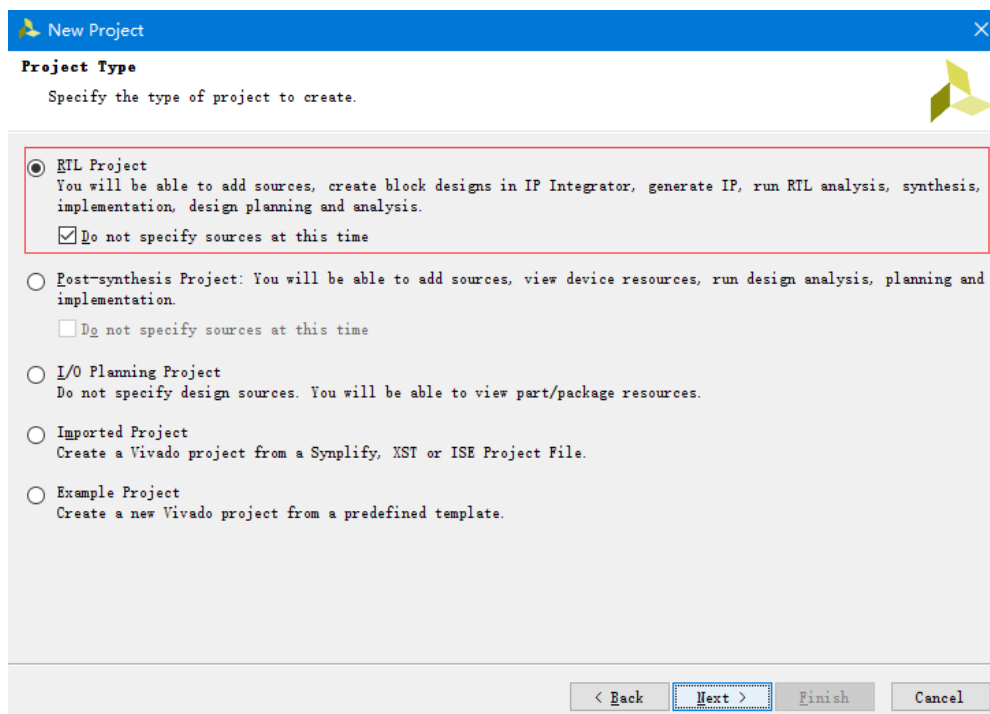
为工程命名，指定工程路径。在 project name 栏输入工程名称，project location 栏指定

工程路径，可以手动输入，也可以点击此栏尾部的...进行选择。操作完成后点击 next。



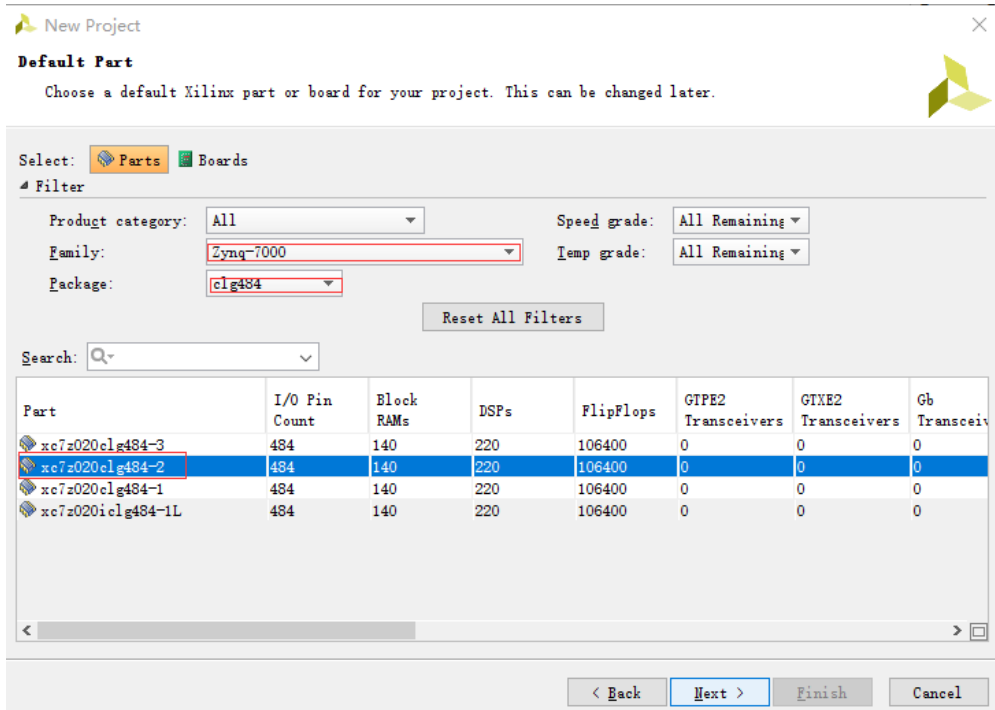
The screenshot shows the 'New Project' dialog box in Vivado, specifically the 'Project Name' step. The title bar is blue with the Vivado logo and 'New Project' text. Below the title bar, the text 'Project Name' is followed by a instruction: 'Enter a name for your project and specify a directory where the project data files will be stored.' The main area contains two text input fields: 'Project name:' with the value 'Example4' and 'Project location:' with the value 'D:/zynq\_test'. Below these fields is a checked checkbox labeled 'Create project subdirectory'. A line of text states 'Project will be created at: D:/zynq\_test/Example4'. At the bottom right, there are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'. The 'Next >' button is highlighted with a blue border.

选择工程类型 默认选择 RTL 工程即可。单击 next。

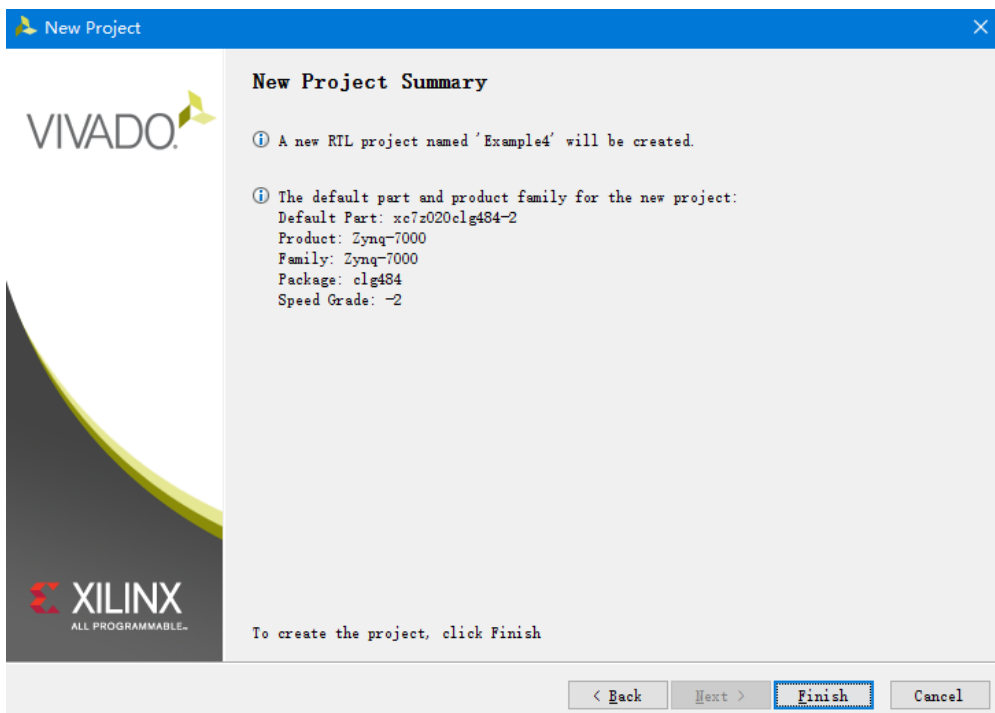


The screenshot shows the 'New Project' dialog box in Vivado, specifically the 'Project Type' step. The title bar is blue with the Vivado logo and 'New Project' text. Below the title bar, the text 'Project Type' is followed by a instruction: 'Specify the type of project to create.' The main area contains five radio button options, each with a description. The first option, 'RTL Project', is selected and highlighted with a red rectangular box. It includes a checked checkbox 'Do not specify sources at this time'. The other options are 'Post-synthesis Project', 'I/O Planning Project', 'Imported Project', and 'Example Project'. At the bottom right, there are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'. The 'Next >' button is highlighted with a blue border.

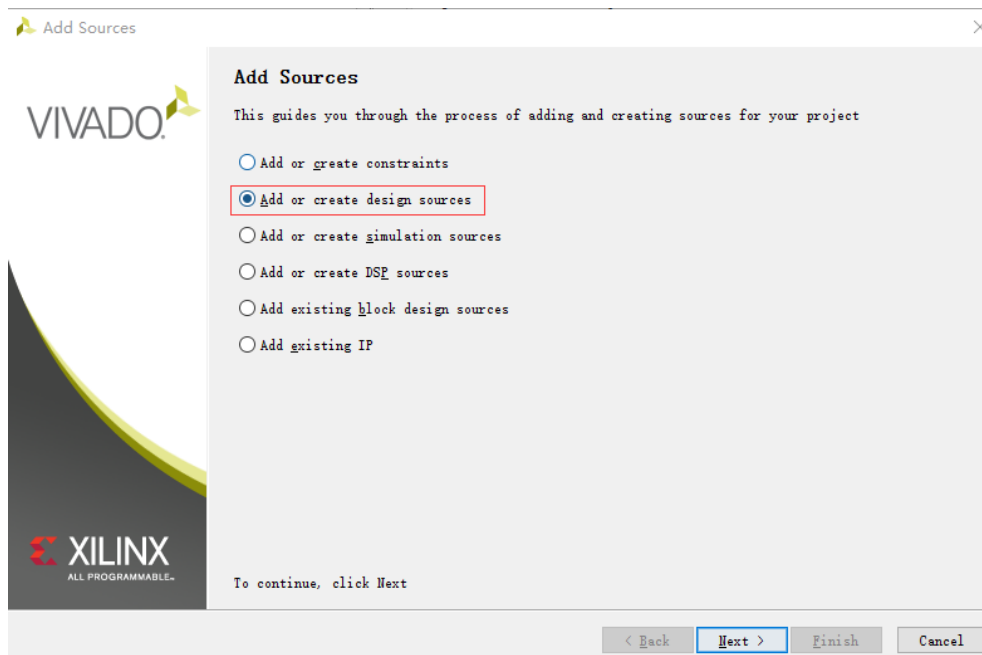
选择器件类型，快速锁定芯片方式如下，family 一栏中下拉菜单中选 中 zynq-7000，package 一栏选择 clg484，此刻在下方器件列表总可以看到已经被筛选过的器件剩下四个，我们选择 xc7z020clg484-2 即可。单击 next。



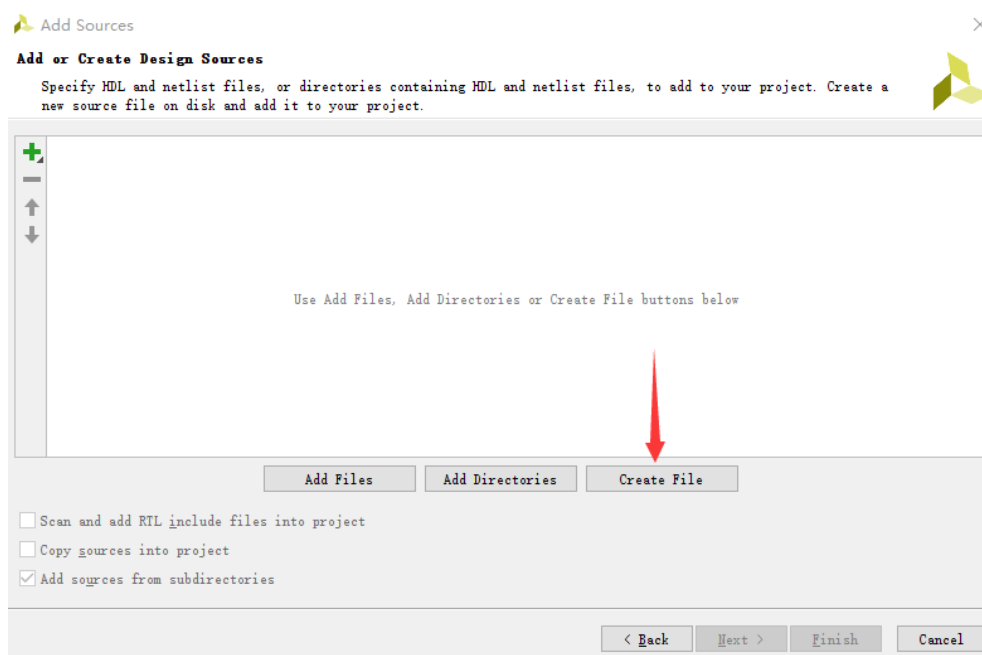
新建工程信息摘要，确认一遍器件信息是否正确，确认无误之后点击 finish。



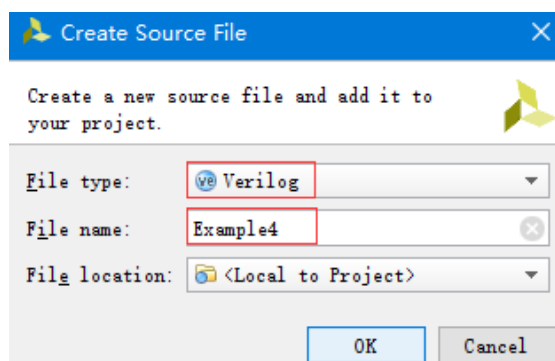
- 建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。操作步骤如下，点击 File-add sources 如下图，选中 add or create design sources，单击 next。



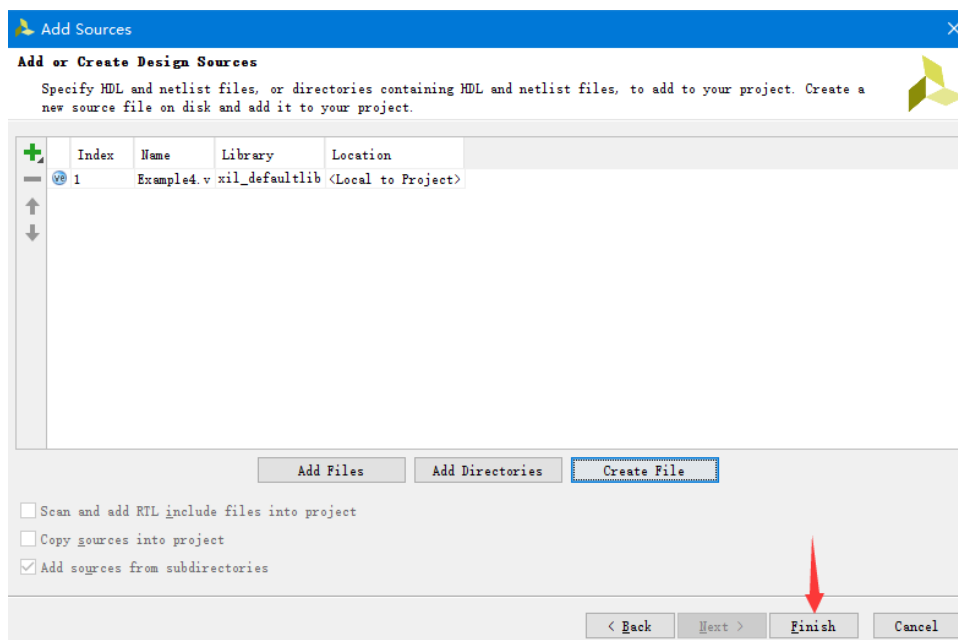
点击 create File 创建文件。



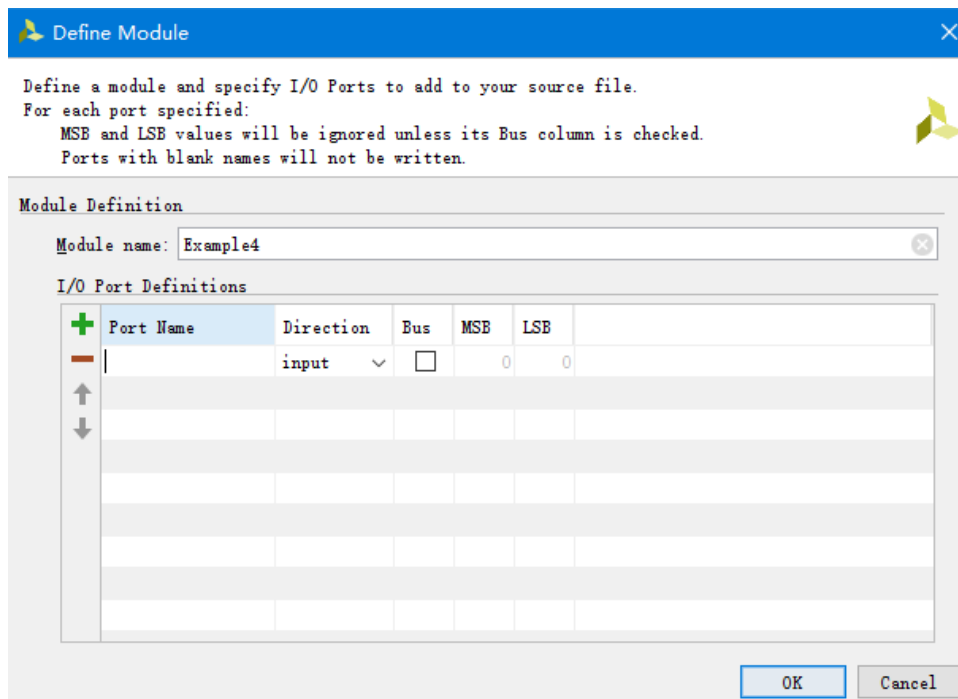
选择需要创建的文件类型 Verilog，输入文件名称，建议与工程名相同。点击 OK。



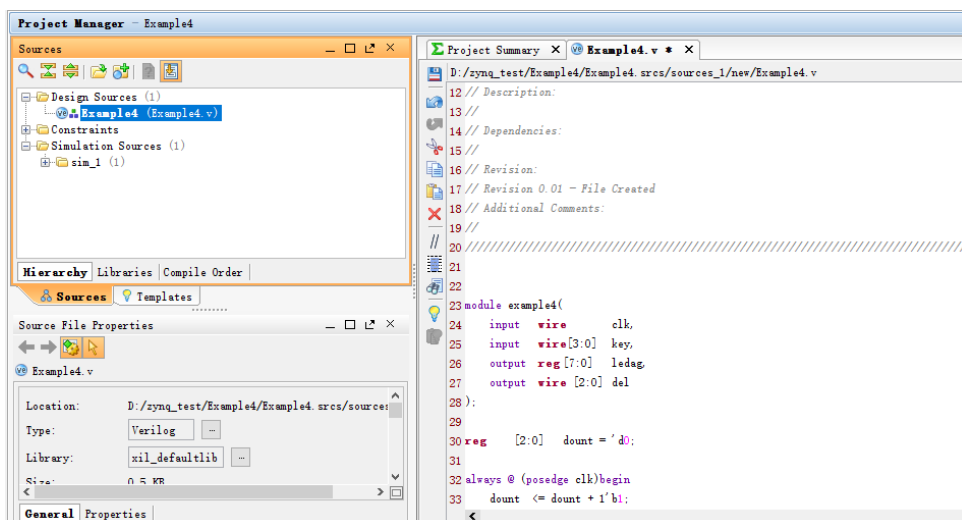
单击 finish。



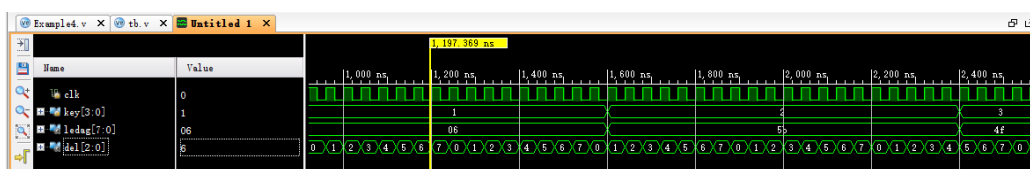
定义设计模块端口,通过+ -添加和删除端口,在 port name 一栏中输入端口名称,direction 一栏中选择端口类型,可以定义为 input、output、inout 三种类型。如果是多比特信号可以勾选 bus 后更爱 MSB 和 LSB 实现多比特信号定义。填写完毕后点击 OK。软件会自动为你生成带有端口的 Verilog 设计文件。



3. 按照实验原理和自己的想法,在 VERILOG 编辑窗口编写 VERILOG 程序,用户可参照光盘中提供的示例程序。



4. 编写完 VERILOG 程序后，保存起来。方法同实验一。
5. 对自己编写的 VERILOG 程序进行综合并仿真，对程序的错误进行修改。参考设计仿真波形如下，对比 key 和 ledag 的数字变化，是否与预期设计相符。



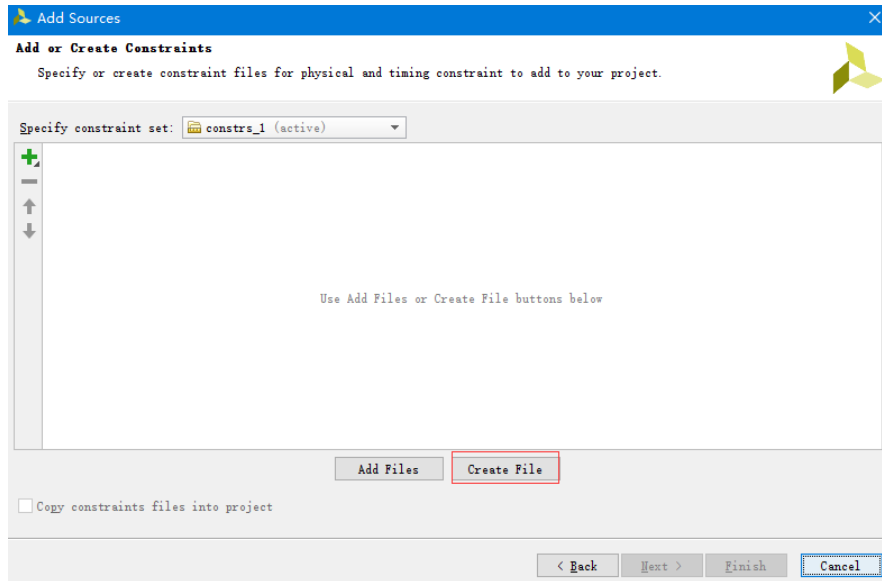
6. 综合仿真无误后，依照时钟、拨动开关、数码管与 FPGA 的管脚连接表 4-2 分配。分配管脚这里使用实验一中提到的添加约束文件。操作方法如下：

File-add sources, 选中 add or create constraints, 点击 next.

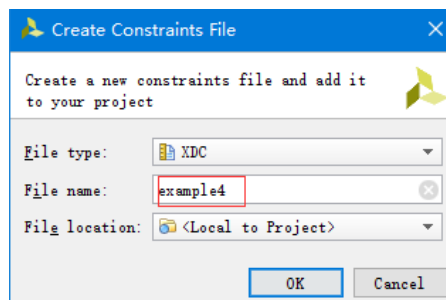


点击 create File

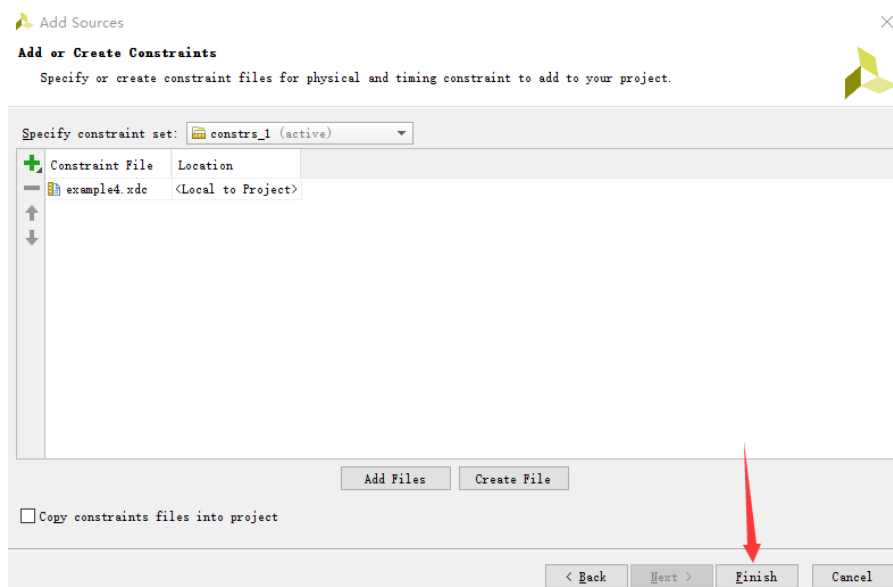




为新建的约束文件命名，建议与工程名字相同。方便文件管理。点击 OK



点击 finish



7. 请在创建的 example4.xdc 文件中编写管脚约束。具体编写方法请参照实验一中的编写说明。建议拷贝实验一中的约束文件，依照如下表格中的管脚进行更改。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
CLK	数字信号源	Y8	时钟为 1KHZ
KEY0	拨动开关 K1	V10	二进制数据输入
KEY1	拨动开关 K2	AA9	
KEY2	拨动开关 K3	W11	
KEY3	拨动开关 K4	Y11	
LEDAG0	数码管 A 段	P16	十六进制数据 输出显示
LEDAG1	数码管 B 段	P17	
LEDAG2	数码管 C 段	N17	
LEDAG3	数码管 D 段	N15	
LEDAG4	数码管 E 段	M15	
LEDAG5	数码管 F 段	L17	
LEDAG6	数码管 G 段	L18	
LEDAG7	数码管 DP 段	K19	
DEL0	位选 DEL0	L22	
DEL1	位选 DEL1	P21	
DEL2	位选 DEL2	N20	

表 4-2 端口管脚分配表

8. 用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

## 五、 实验现象与结果：

以设计的参考示例为例，当设计文件加载到目标器件后，将数字信号源模块的时钟选择为 1KHZ，拨动四位拨动开关，使其为一个固定二进制数值，则八个数码管均显示拨动开关所表示的十六进制的值。

## 六、 实验报告：

1. 绘出仿真波形，并作说明。
2. 说明扫描时钟是如何工作的，改变扫描时钟会有什么变化。
3. 实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。

## 实验五 数控分频器的设计

### 一、 实验目的：

1. 学习数控分频器的设计、分析和测试方法。
2. 了解和掌握分频电路实现的方法。
3. 掌握 EDA 技术的层级化设计方法。

### 二、 实验原理：

数控分频器的功能就是当输入端给定不同的输入数据时，将对输入的时钟信号有不同的分频比，数控分频器就是用计数值可并行预置的加法计数器来设计完成的，方法是将计数溢出位与预置数加载输入信号相接得到。

### 三、 实验内容：

本实验要求完成的任务是在时钟信号的作用下，通过输入八位的拨动开关输入不同的数据，改变分频比，使输出端口输出不同频率的时钟信号，得到数控分频的效果。在实验中时，数字时钟选择 1KHZ 作为输入的时钟信号（频率过高观察不到 LED 的闪烁快慢），用 12 个拨动开关做为数据的输入，当 12 个拨动开关置为一个固定二进制数时，在输出端口输出对应频率的时钟信号，用户可以用示波器接信号输出模块观察频率的变化。也可以使输出端口接 LED 灯来观察频率的变化。在此实验中我们把输入接入 LED 灯模块。实验箱中的拨动开关、LED 与 FPGA 的接口电路，以及拨动开关、LED 与 FPGA 的管脚连接方法在实验一中都做了详细说明，这里不在赘述。

### 四、 实验步骤：

1. 打开 VIVADO 软件，新建一个工程。
2. 建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。
3. 按照实验原理和自己的想法，在设计编辑窗口编写 VERILOG 程序。
4. 编写完 VERILOG 程序后，保存起来。方法同实验一。
5. 对自己编写的 VERILOG 程序进行综合并仿真，对程序的错误进行修改。
6. 综合仿真无误后，参照表 5-1 进行管脚分配拨动开关、LED 与 FPGA 管脚。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
INCLK	数字信号源	Y8	时钟为 1KHZ
DATA0	拨动开关 K1	V10	分频比数据
DATA 1	拨动开关 K2	AA9	

DATA 2	拨动开关 K3	W11	
DATA 3	拨动开关 K4	Y11	
DATA 4	拨动开关 K5	AB10	
DATA 5	拨动开关 K6	AA11	
DATA 6	拨动开关 K7	V12	
DATA 7	拨动开关 K8	U10	
DATA8	拨动开关 K9	J20	
DATA9	拨动开关 K10	J18	
DATA10	拨动开关 K11	K16	
DATA11	拨动开关 K12	J15	
FOUT	LED 灯 LED1	C15	分频输出

表 5-1 端口管脚分配表

7. 用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

## 五、 实验现象与结果：

以设计的参考示例为例，当设计文件加载到目标器件后，将数字信号源模块的时钟选择为 1KHZ，拨动八位拨动开关，使其为一个数值，则输入的时钟信号使 LED 灯开始闪烁，改变拨动开关，LED 的闪烁快慢会按一定的规则发生改变。

## 六、 实验报告：

1. 输入不同的 DATA 值绘出仿真波形，并作说明。
2. 在这个程序的基础上扩展成 16 位的分频器，写出 VERILOG 代码。
3. 将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。

## 实验六 VERILOG 层次结构电路设计

### 一、 实验目的：

1. 学习在 VIVADO 软件中多文件层级结构化调用。
2. 掌握单个设计转变成层级结构化设计的规则与方法。
3. 掌握从设计单个设计到层级结构化设计的创建过程。

### 二、 实验原理：

本实验的实验原理就是将前面设计的实验三、四、五通过 VIVADO 软件合并成一个设计文件。实现实验三、四、五中的所有功能。

### 三、 实验内容：

本实验要求完成的任务与实验三、四、五的实验内容基本一致。在实验中，时钟信号选取 1KHZ 做为数码管的扫描时钟，拨动开关输入一个预置的 12 位数据，经过数控分频电路（实验五）分频后得到一个较低的频率做为加法计数器（实验三）的时钟频率进行计数器的加法运算。得到的值给数码显示译码电路（实验四）在数码管上显示出来。实验箱中的数字时钟模块、拨动开关、按键开关、数码管、LED 与 FPGA 的接口电路，以及拨动开关、按键开关、数码管、LED 与 FPGA 的管脚连接在实验三、四、五中都做了详细说明，这里不在赘述。

### 四、 实验步骤：

1. 打开 VIVADO 软件，新建一个工程。
2. 将以前编写的实验三、四、五的源程序代码复制到当前工作目录下保存起来。
3. 单击 Add sources>Add files，如图 6-1 所示，打开复制到当前工作目录下和 sange 源程序代码， Example3.V， Example4.V， Example5.V 程序。

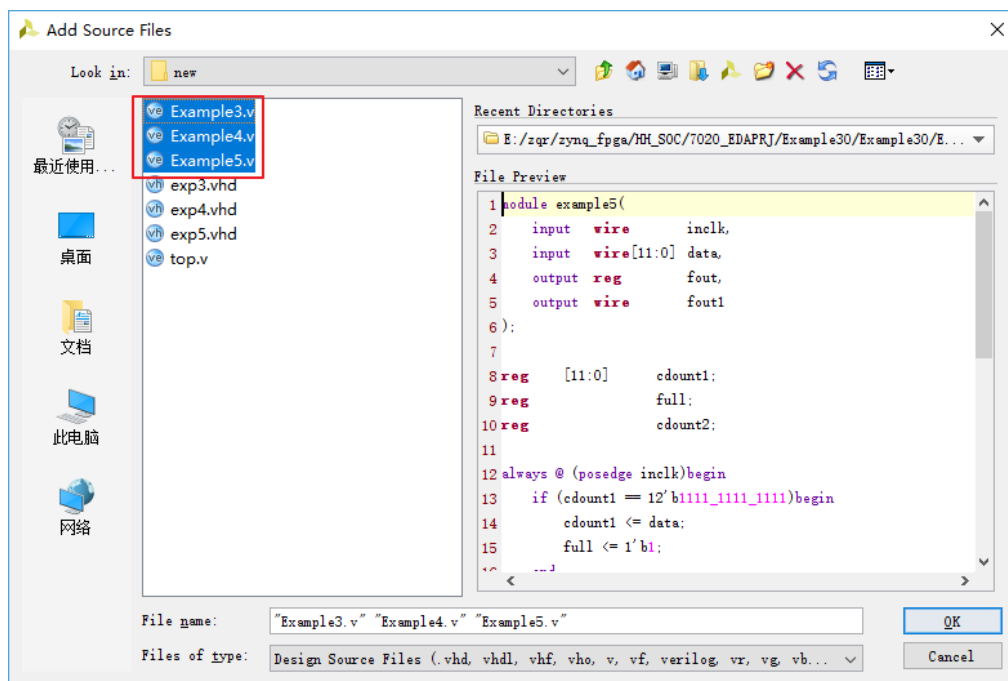
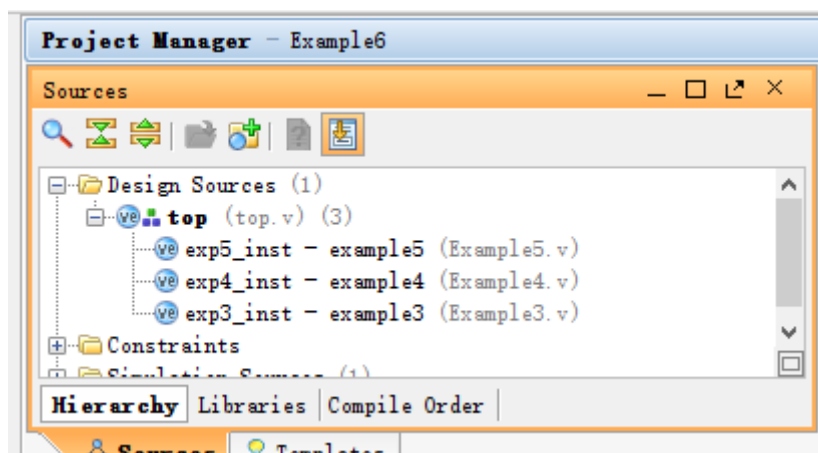


图 6-1 打开一个设计文件

4. 创建 Top.v 顶层文件，并把上一步添加的源代码，通过 Verilog 代码例化到顶层之下，软件会自动识别出顶层文件及层级结构。如下图所示



5. 对自己编写程序进行保存，然后综合并仿真，对程序的错误进行修改。
6. 综合仿真无误后，依照拨动开关、LED 与 FPGA 的管脚连接表 6-1 分配。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
CLK	数字信号源	Y8	显示时钟控控制
DATA0	拨动开关 K1	V10	
DATA 1	拨动开关 K2	AA9	
DATA 2	拨动开关 K3	W11	
DATA 3	拨动开关 K4	Y11	
DATA 4	拨动开关 K5	AB10	

DATA 5	拨动开关 K6	AA11	
DATA 6	拨动开关 K7	V12	
DATA 7	拨动开关 K8	U10	
DATA 8	拨动开关 K9	J20	
DATA 9	拨动开关 K10	J18	
DATA 10	拨动开关 K11	K16	
DATA 11	拨动开关 K12	J15	
RET	按键开关 S1	P15	复位信号
COUT	LED 灯 LED1	C15	进位标志位
LEDAG0	数码管 A 段	P16	数据显示
LEDAG1	数码管 B 段	P17	
LEDAG2	数码管 C 段	N17	
LEDAG3	数码管 D 段	N15	
LEDAG4	数码管 E 段	M15	
LEDAG5	数码管 F 段	L17	
LEDAG6	数码管 G 段	L18	
LEDAG7	数码管 DP 段	K19	
DEL0	位选 DEL0	L22	
DEL1	位选 DEL1	P21	
DEL2	位选 DEL2	N20	

表 6-1 端口管脚分配表

7. 用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

## 五、 实验现象与结果：

以设计的参考示例为例，当设计文件加载到目标器件后，将数字信号源模块的时钟选择为 1KHZ，拨动 12 位拨动开关，使其为一个数值，则八位数码管按一定的速率开始显示“0-F”，当数码管显示 A-F 时 LED 灯 LED1 开始被点亮，显示其它数值时熄灭。按 S1 键显示的数值又从 0 开始，拨动八位拨动开关，置于其它数据，数码管的显示速率会发生改变。

## 六、 实验报告：

1. 绘制出仿真波形，并作说明。
2. 自己设计文件，然后通过结构化层级设计连接到一起，设计自己的电路并在实验系统中验证。进一步掌握这种方法。
3. 写出在软件中通过其它方法单个设计到结构化层级设计的转换过程。
4. 将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。

## 实验七 步长可变的加减计数器的设计

### 一、 实验目的

- 1、 加深对计数器的认识。
- 2、 了解用 VERILOG 语言实验计数器的过程。

### 二、 实验原理

计数器的步长是指计数器每次的改变量，比如 74LS169 器件，它每次改变的时候要么加 1，要么减 1，因此，我们就说该计数器的步长为 1。

在很多应用领域，都希望计数器的步长可变。所谓步长可变，也就是计数器的步长是一个不定值，具体是多少要靠外部来干预。比如外部给定其步长为 5，那么该计数器每次要么加 5，要么减 5，也就是说计数器每次改变的量是 5。这种步长可变的计数器才具有一定的实际意义，比如在 DDSF 中的地址累加器就是一个步长可变的递增计数器。

### 三、 实验内容

本实验的任务是实现一个简单的 12 位计数器，步长的改变量要求从 0 至 15 可变，在设计中用拨动开关的 K1-K4 来作为步长改变量的输入，用 K12 来控制计数器的加减。具体要求为：当 K12 输入为高电平时，计数器为步长可变的加法计数器；当 K12 输入为低电平时，计数器为步长可变的减法计数器。计数器的输出用 12 位 LED 灯来表示其二进制代码。实验中的计数器的时钟频率为了使用观察用 1HZ 的时钟作为步进的频率。实验箱中的数字时钟模块、拨动开关、LED 与 FPGA 的接口电路，以及数字时钟源、拨动开关、LED 与 FPGA 的管脚连接在以前的实验中都做了详细说明，这里不在赘述。

### 四、 实验步骤

- 1、 打开 VIVADO 软件，新建一个工程。
- 2、 建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。
- 3、 按照实验原理和自己的想法，在 VERILOG 编辑窗口编写 VERILOG 程序。
- 4、 编写完 VERILOG 程序后，保存起来。方法同实验一。
- 5、 对自己编写的 VERILOG 程序进行综合并仿真，对程序的错误进行修改。
- 6、 综合仿真无误后，依照数字信号源、拨动开关、LED 灯与 FPGA 的管脚连接表 7-1 分配。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。



端口名	使用模块信号	对应 FPGA 管脚	说 明
CLK	数字信号源	Y8	时钟为 1KHZ
STEP0	拨动开关 K1	V10	步长的输入数据
STEP1	拨动开关 K2	AA9	
STEP2	拨动开关 K3	W11	
STEP3	拨动开关 K4	Y11	
UD	拨动开关 K12	J15	计数器加减控制
COUNT0	LED 灯 D1	C15	计数器输出显示
COUNT1	LED 灯 D2	E15	
COUNT2	LED 灯 D3	B16	
COUNT3	LED 灯 D4	A16	
COUNT4	LED 灯 D5	G15	
COUNT5	LED 灯 D6	F16	
COUNT6	LED 灯 D7	C18	
COUNT7	LED 灯 D8	D16	
COUNT8	LED 灯 D9	G17	
COUNT9	LED 灯 D10	B19	
COUNT10	LED 灯 D11	A18	
COUNT11	LED 灯 D12	D18	

表 7-1 端口管脚分配表

- 7、 用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

## 五、 实验现象与结果

以设计的参考示例为例，当设计文件加载到目标器件后，将数字信号源模块的时钟选择为 1HZ，拨动拨动开关 K1-K4，输入一个四位的数据做为一个可变的步长，观察 12 位 LED 灯的变化是不是按这个可变的步长在进行加减。拨动拨动开关 K12 观察 LED 是不是按设计的思想在进行加法和减法计数器的切换。

## 六、 实验报告

- 1、 给出不同的乘数和被乘数，绘仿真波形，并作说明。
- 2、 在这个程序的基础上设计一个八位的并行乘法器。
- 3、 在这个程序的基础上，用数码管来显示相乘结果的十进制值。
- 4、 实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。

## 实验八 四位并行乘法器的设计

### 一、 实验目的

- 1、 了解四位并行乘法器的原理。
- 2、 了解四位并行乘法器的设计思想。
- 3、 掌握用 VERILOG 语言实现基本二进制运算的方法。

### 二、 实验原理

实现并行乘法器的方法又很多种，但是归结起来基本上分为两类，一类是靠组合逻辑电路实现，另一类流水线实现。流水线结构的并行乘法器的最大优点就是速度快，尤其在连续输入的乘法器中，可以达到近乎但周期的运算速度，但是实现起来比组合逻辑电路要稍微复杂一些。下面就组合逻辑电路实现无符号数乘法的方法作详细介绍。

假如有被乘数 A 和乘数 B，首先用 A 与 B 的最低位相乘得到 S1，然后再把 A 左移 1 位与 B 的第 2 位相乘得到 S2，再将 A 左移 3 位与 B 的第三位相乘得到 S3，依此类推，直到把 B 的所有位都乘完为止，然后再把乘得的结果 S1、S2、S3……相加即得到相乘的结果。

需要注意的是，具体实现乘法器并不是真正的去乘，而是利用简单的判断去实现，举个简单的例子。假如 A 左移 n 位后与 B 的第 n 位相乘，如果 B 的这位为 ‘1’，那么相乘的中间结果就是 A 左移 n 位后的结果，否则如果 B 的这位为 ‘0’，那么就on直接让相乘的中间结果为 0 即可。带 B 的所有位相乘结束后，把所有的中间结果相加即得到 A 与 B 相乘的结果。

### 三、 实验内容

本实验的任务是实现一个简单的四位并行乘法器，被乘数 A 用拨挡开关模块的 K1~K4 来表示，乘数 B 用 K7~K10 来表示，相乘的结果用 LED 模块的 LED1~LED12 来表示，LED 亮表示对应的位为 ‘1’。时钟信号选取 1KHZ 做为扫描时钟，拨动开关输入一个四位的被乘数和一个四位的乘数，经过设计电路相乘后得到的数据在 LED 灯上显示出来。实验箱中的数字时钟模块、拨动开关、LED 与 FPGA 的接口电路，以及数字时钟源、拨动开关、LED 与 FPGA 的管脚连接在以前的实验中都做了详细说明，这里不在赘述。

### 四、 实验步骤

- 1、 打开 VIVADO 软件，新建一个工程。
- 2、 建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。
- 3、 按照实验原理和自己的想法，在 VERILOG 编辑窗口编写 VERILOG 程序。
- 4、 编写完 VERILOG 程序后，保存起来。方法同实验一。

- 5、 对自己编写的 VERILOG 程序进行综合并仿真，对程序的错误进行修改。
- 6、 综合仿真无误后，依照数字信号源、拨动开关、LED 灯与 FPGA 的管脚连接表 8-1。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
CLK	数字信号源	Y8	时钟为 1KHZ
A0	拨动开关 K4	Y11	被乘数数据
A1	拨动开关 K3	W11	
A2	拨动开关 K2	AA9	
A3	拨动开关 K1	V10	
B0	拨动开关 K10	J18	乘数数据
B1	拨动开关 K9	J20	
B2	拨动开关 K8	U10	
B3	拨动开关 K7	V12	
COUT0	LED 灯 LED12	C15	两数相乘结果 输出
COUT1	LED 灯 LED11	E15	
COUT2	LED 灯 LED10	B16	
COUT3	LED 灯 LED9	A16	
COUT4	LED 灯 LED8	G15	
COUT5	LED 灯 LED7	F16	
COUT6	LED 灯 LED6	C18	
COUT7	LED 灯 LED5	D16	
COUT8	LED 灯 LED4	G17	
COUT9	LED 灯 LED3	B19	
COUT10	LED 灯 LED2	A18	
COUT11	LED 灯 LED1	D18	

表 8-1 端口管脚分配表

- 7、 用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

### 五、 实验现象与结果

以设计的参考示例为例，当设计文件加载到目标器件后，将数字信号源模块的时钟选择为 1KHZ，拨动相应的拨动开关，输入一个四位的乘数和被乘数，则在 LED 灯上显示这两个数值相乘的结果的二进制数。

### 六、 实验报告

- 1、 给出不同的乘数和被乘数，绘仿真波形，并作说明。
- 2、 在这个程序的基础上设计一个八位的并行乘法器。
- 3、 在这个程序的基础上，用数码管来显示相乘结果的十进制值。

4、 实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。

## 实验九 设计四位全加器

### 一、实验目的

- 1、了解四位全加器的工作原理。
- 2、掌握基本组合逻辑电路的 FPGA 实现。
- 3、熟练应用 Vivado 进行 FPGA 开发。

### 二、实验原理

全加器是由两个加数  $X_i$  和  $Y_i$  以及低位来的进位  $C_{i-1}$  作为输入，产生本位和  $S_i$  以及向高位的进位  $C_i$  的逻辑电路。它不但要完成本位二进制码  $X_i$  和  $Y_i$  相加，而且还要考虑到低位进位  $C_{i-1}$  的逻辑。对于输入为  $X_i$ 、 $Y_i$  和  $C_{i-1}$ ，输出为  $S_i$  和  $C_i$  的情况，根据二进制加法法则可以得到全加器的真值表如下表 9-1 所示：

$X_i$	$Y_i$	$C_{i-1}$	$S_i$	$C_i$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

表 9-1 全加器真值表

由真值表得到  $S_i$  和  $C_i$  的逻辑表达式经化简后为：

$$S_i = X_i \oplus Y_i \oplus C_{i-1}$$
$$C_i = (X_i \oplus Y_i)C_{i-1} + X_i Y_i$$

这仅仅是一位的二进制全加器，要完成一个四位的二进制全加器，只需要把四个级联起来即可。

### 三、实验内容

本实验要完成的任务是设计一个四位二进制全加器。具体的实验过程就是利用实验系统上的拨动开关模块的 K1~K4 作为一个加数 X 输入，K7~K10 作为另一个加数 Y 输入，用 LED 模块的 LED1~LED12 来作为结果 S 输出，LED 亮表示输出 ‘1’，LED 灭表示输出 ‘0’。

实验箱中的拨动开关、LED 与 FPGA 的接口电路，以及拨动开关、LED 与 FPGA 的管脚连接在以前的实验中都做了详细说明，这里不在赘述。

#### 四、实验步骤

- 1、 打开 VIVADO 软件，新建一个工程。
- 2、 建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。
- 3、 按照实验原理和自己的想法，在 VERILOG 编辑窗口编写 VERILOG 程序。
- 4、 编写完 VERILOG 程序后，保存起来。方法同实验一。
- 5、 对自己编写的 VERILOG 程序进行综合并仿真，对程序的错误进行修改。
- 6、 综合仿真无误后，依照数字信号源、拨动开关、LED 灯与 FPGA 的管脚连接表 9-2 分配。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
X0	拨动开关 K4	Y11	加数数据
X1	拨动开关 K3	W11	
X2	拨动开关 K2	AA9	
X3	拨动开关 K1	V10	
Y0	拨动开关 K10	J18	加数数据
Y1	拨动开关 K9	J20	
Y2	拨动开关 K8	U10	
Y3	拨动开关 K7	V12	
m_Result0	LED 灯 LED12	C15	两个加数相加结果输出
m_Result1	LED 灯 LED11	E15	
m_Result2	LED 灯 LED10	B16	
m_Result3	LED 灯 LED9	A16	
m_Result4	LED 灯 LED8	G15	
m_Result5	LED 灯 LED7	F16	
m_Result6	LED 灯 LED6	C18	
m_Result7	LED 灯 LED5	D16	
m_Result8	LED 灯 LED4	G17	
m_Result9	LED 灯 LED3	B19	
m_Result10	LED 灯 LED2	A18	
m_Result11	LED 灯 LED1	D18	

表 9-2 端口管脚分配表

- 7、 用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

#### 五、实验现象与结果

以设计的参考示例为例，当设计文件加载到目标器件后，拨动相应的拨动开关，输入两

个四位的加数，则在 LED 灯上显示这两个数值相加的结果的二进制数。

## 六、 实验报告

- 1、 给出不同的加数，绘仿真波形，并作说明。
- 2、 在这个程序的基础上设计一个八位的全加器。
- 3、 在这个程序的基础上，用数码管来显示相乘结果的十进制值。
- 4、 将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。

## 实验十 可控脉冲发生器的设计

### 一、 实验目的

- 1、 了解可控脉冲发生器的实现机理。
- 2、 学会用示波器观察 FPGA 产生的信号。
- 3、 学习用 VERILOG 编写复杂功能的代码。

### 二、 实验原理

脉冲发生器就是要产生一个脉冲波形,而可控脉冲发生器则是要产生一个周期和占空比可变的脉冲波形。可控脉冲发生器的实现原理比较简单,可以简单的理解为一个计数器对输入的时钟信号进行分频的过程。通过改变计数器的上限值来达到改变周期的目的,通过改变电平翻转的阈值来达到改变占空比的目的。下面举个简单的例子来说明其工作原理。

假如有一个计数器  $T$  对时钟分频,其计数的范围是从  $0 \sim N$ ,另取一个  $M(0 \leq M \leq N)$ ,若输出为  $Q$ ,那么  $Q$  只要满足条件时,通过改变  $N$  值,即可改变输出的脉冲波的周期;改变  $M$  值,即可改变脉冲波的占空比。这样输出的脉冲波的周期和占空比分别为:

$$\begin{aligned} \text{周期} &= (N+1)T_{\text{CLOCK}} \\ \text{占空比} &= \frac{M}{N+1} \times 100\% \end{aligned} \quad Q = \begin{cases} 1 & 0 \leq T < M \\ 0 & M \leq T \leq N \end{cases}$$

### 三、 实验内容

本实验的任务就是要设计一个可控的脉冲发生器,要求输出的脉冲波的周期和占空比都可变。具体的实验过程中,时钟信号选用时钟模块中的 1MHz 时钟,然后再用按键模块的 S1 和 S7 来控制脉冲波的周期,每按下 S1,  $N$  会在慢速时钟作用下不断地递增 1,按下 S7,  $N$  会在慢速时钟作用下不断地递减 1;用 S2 和 S8 来控制脉冲波的占空比,每按下 S2,  $M$  会在慢速时钟作用下不断地递增 1,每按下 S8,  $M$  会在慢速时钟作用下不断地递减 1, S12 用作复位信号,当按下 S12 时,复位 FPGA 内部的脉冲发生器模块。脉冲波的输出直接输出到实验箱观测模块的探针,以使用示波器观察输出波形的改变。

### 四、 实验步骤

- 1、 打开 VIVADO 软件,新建一个工程。
- 2、 建完工程之后,再新建一个 VERILOG File,按照需求定义输入输出管脚。
- 3、 按照实验原理和自己的想法,在 VERILOG 编辑窗口编写 VERILOG 程序。
- 4、 编写完 VERILOG 程序后,保存起来。方法同实验一。



- 5、 对自己编写的 VERILOG 程序进行综合并仿真，对程序的错误进行修改。
- 6、 综合仿真无误后，依照拨动开关、LED 与 FPGA 的管脚连接表 10-1 分配。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
CLK	数字信号源	Y8	时钟为 1MHZ
NU	按键开关 S1	P15	频率控制/增加
ND	按键开关 S7	K18	频率控制/减少
MU	按键开关 S2	N18	占空比控制/增加
MD	按键开关 S8	K21	占空比控制/减少
RST	按键开关 S12	M16	复位控制
FOUT	输出观测模块	Y16	示波器观测点

表 10-1 端口管脚分配表

- 7、 用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

## 五、 实验结果与现象

以设计的参考示例为例，当设计文件加载到目标器件后，将数字信号源模块的时钟选择为 1MHZ，按下按键开关模块的 S12 按键，在输出观测模块通过示波器可能观测到一个频率约为 1KHZ、占空比为 50%的矩形波。按下 S1 键或者 S7 键，这个矩形波的频率会发生相应的增加或者减少。按下 S2 键或者 S8 键，这个矩形波的占空比会相应的增加或减少。

## 六、 实验报告

- 1、 绘出仿真波形，并作说明。
- 2、 在这个实验的基础上重新设计，使程序改变频率的时候不会影响占空比的改变。
- 3、 将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。

# 实验十一 基本触发器的设计

## 一、实验目的

- 1、了解基本触发器的工作原理。
- 2、进一步熟悉在 Vivado 中基于原理图设计的流程。

## 二、实验原理

基本触发器的电路如下图 11-1 所示。它可以由两个与非门交叉耦合组成,也可以由两个或非门交叉耦合组成。现在以两个与非门组成的基本触发器为例,来分析其工作原理。根据与非逻辑关系,可以得到基本触发器的状态转移真值表及简化的真值表,如下表 11-1 所示:

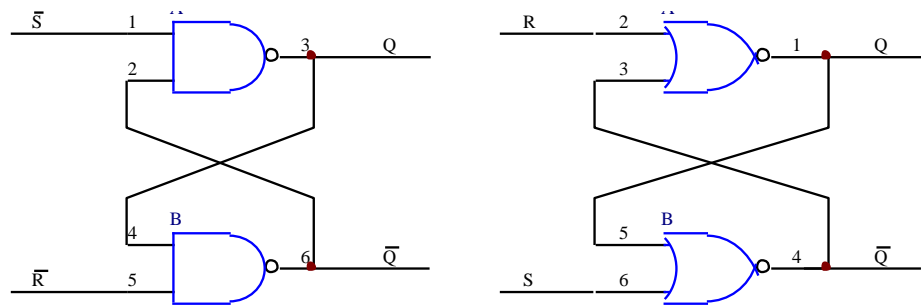


图 11-1 基本触发器电路

状态转移真值表				简化真值表		
$\bar{R}$	$\bar{S}$	$Q^n$	$Q^{n+1}$	$\bar{R}$	$\bar{S}$	$Q^{n+1}$
0	1	0	0	0	1	0
0	1	1	0	1	0	1
1	0	0	1	1	1	$Q^n$
1	0	1	1	0	0	不定
1	1	0	0			
1	1	1	1			
0	0	0	不定			
0	0	1	不定			

表 11-1 基本触发器状态转移真值表

根据真值表, 不难写出其特征方程:

其中式 (2) 为约束条件。

$$\begin{cases} Q^{n+1} = \bar{S} + \bar{R}Q^n & (1) \\ \bar{S} + \bar{R} = 1 & (2) \end{cases}$$

## 三、实验内容

本实验的任务就是利用 Vivado 软件产生一个基本触发器, 触发器的形式可以是与非门

结构的，也是可以或非门结构的。实验中用按键模块的用 K1 和 K3 来分别表示  $R$  和  $S$ ，用 LED 模块的 LED12 和 LED1 分别表示  $Q$  和  $\overline{Q}$ 。在  $R$  和  $S$  满足式 (2) 的情况下，观察  $Q$  和  $\overline{Q}$  的变化。

实验箱中的拨动开关、LED 与 FPGA 的接口电路，以及拨动开关、LED 与 FPGA 的管脚连接在以前的实验中都做了详细说明，这里不在赘述。

四、 实验步骤

- 1、 打开 VIVADO 软件，新建一个工程。
- 2、 建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。
- 3、 按照实验原理和自己的想法，在图形符号编辑窗口编写设计程序。
- 4、 设计好设计电路程序后，保存起来。方法同实验一。
- 5、 对自己编写的设计电路程序进行综合并仿真，对程序的错误进行修改。
- 6、 综合仿真无误后，依照拨动开关、LED 与 FPGA 的管脚连接表 11-2 分配。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚
NR	拨动开关 K1	V10
NS	拨动开关 K2	AA9
Q	LED 灯 LED12	D18
NQ	LED 灯 LED1	C15

表 11-2 端口管脚分配表

- 7、 用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

五、 实验现象与结果

以设计的参考示例为例，当设计文件加载到目标器件后，拨动相应的拨动开关（即  $R$ 、 $S$ ），则通过 LED 灯上的亮和灭来显示这个触发器的输入结果。将输入与输出和表 11-1 基本触发器状态转移真值表进行比较，看是否一致。

六、 实验报告

- 1、 绘出不同  $R$ 、 $S$  值的仿真波形，并作说明。
- 2、 试设计一个其它的功能触发器如 D 触发器、JK 触发器等
- 3、 将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。

# 应用实验部分

## 实验十二 矩阵键盘显示电路的设计

### 一、 实验目的

- 1、 了解普通  $4 \times 4$  键盘扫描的原理。
- 2、 进一步加深七段码管显示过程的理解。
- 3、 了解对输入/输出端口的定义方法。

### 二、 实验原理

实现键盘有两种方案：一是采用现有的一些芯片实现键盘扫描；再就是用软件实现键盘扫描。作为一个嵌入系统设计人员，总是会关心产品成本。目前有很多芯片可以用来实现键盘扫描，但是键盘扫描的软件实现方法有助于缩减一个系统的重复开发成本，且只需要很少的 CPU 开销。嵌入式控制器的功能强，可能充分利用这一资源，这里就介绍一下软键盘的实现方案。

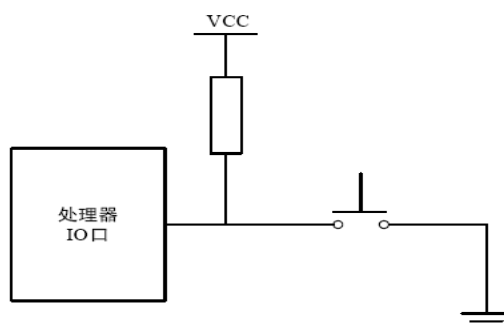


图 12-1 简单键盘电路

通常在一个键盘中使用了一个瞬时接触开关，并且用如图 12-1 所示的简单电路，微处理器可以容易地检测到闭合。当开关打开时，通过处理器的 I/O 口的一个上拉电阻提供逻辑 1；当开关闭合时，处理器的 I/O 口的输入将被拉低得到逻辑 0。可遗憾的是，开关并不完善，因为当它们被按下或者被释放时，并不能够产生一个明确的 1 或者 0。尽管触点可能看起来稳定而且很快地闭合，但与微处理器快速的运行速度相比，这种动作是比较慢的。当触点闭合时，其弹起就像一个球。弹起效果将产生如图 12-2 所示的好几个脉冲。弹起的持续时间通常将维持在  $5\text{ms} \sim 30\text{ms}$  之间。如果需要多个键，则可以将每个开关连接到微处理器上它自己的输入端口。然而，当开关的数目增加时，这种方法将很快使用完所有的输入端口。

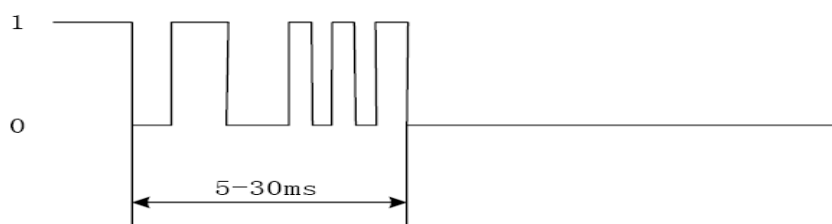


图 12-2 按键抖动

键盘上阵列这些开关最有效的方法（当需要 5 个以上的键时）就形成了一个如图 12-3 所示的二维矩阵。当行和列的数目一样多时，也就是方型的矩阵，将产生一个最优化的布列方式（I/O 端被连接的时候）。一个瞬时接触开关（按钮）放置在每一行与线一列的交叉点。矩阵所需的键的数目显然根据应用程序而不同。每一行由一个输出端口的一位驱动，而每一列供给输入端口一位。

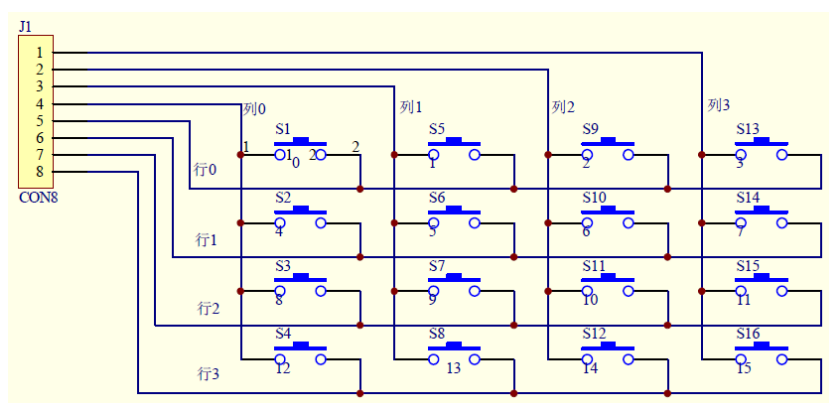


图 12-3 矩阵键盘

键盘扫描的实现过程如下：对于 4×4 键盘，通常连接为 4 行、4 列，因此要识别按键，只需要知道是哪一行和哪一列即可，为了完成这一识别过程，我们的思想是，首先固定输出 4 行为高电平，然后输出 4 列为低电平，在读入输出的 4 行的值，通常高电平会被低电平拉低，如果读入的 4 行均为高电平，那么肯定没有按键按下，否则，如果读入的 4 行有一位为低电平，那么对应的该行肯定有一个按键按下，这样便可以获取到按键的行值。同理，获取列值也是如此，先输出 4 列为高电平，然后在输出 4 行为低电平，再读入列值，如果其中有哪一位为低电平，那么肯定对应的那一列有按键按下。

获取到行值和列值以后，组合成一个 8 位的数据，根据实现不同的编码在对每个按键进行匹配，找到键值后在 7 段码管显示。

### 三、 实验内容

本实验要求完成的任务是通过编程实现对 4X4 矩阵键盘按下键的键值的读取，并在数码管上完成一定功能（如移动等）的显示。按键盘的定义，按下 “\*” 键则在数码管是显示

“E”键值。按下“#”键在数码管上显示“F”键值。其它的键则按键盘上的标识进行显示。

在此实验中数码管与 FPGA 的连接电路和管脚连接在以前的实验中都做了详细说明，这里不在赘述。本实验箱上的 4X4 矩阵键盘的电路原理如图 12-4 所示。与 FPGA 的管脚连接如表 12-1 所示。

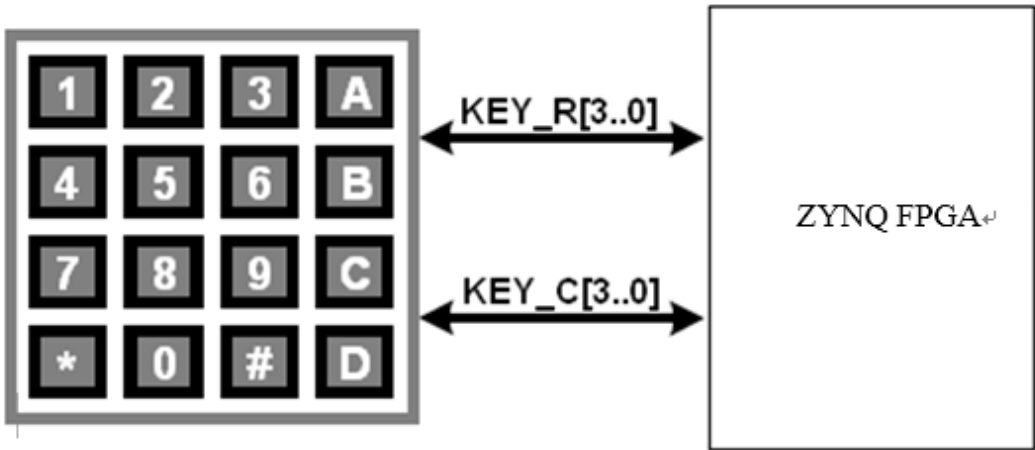


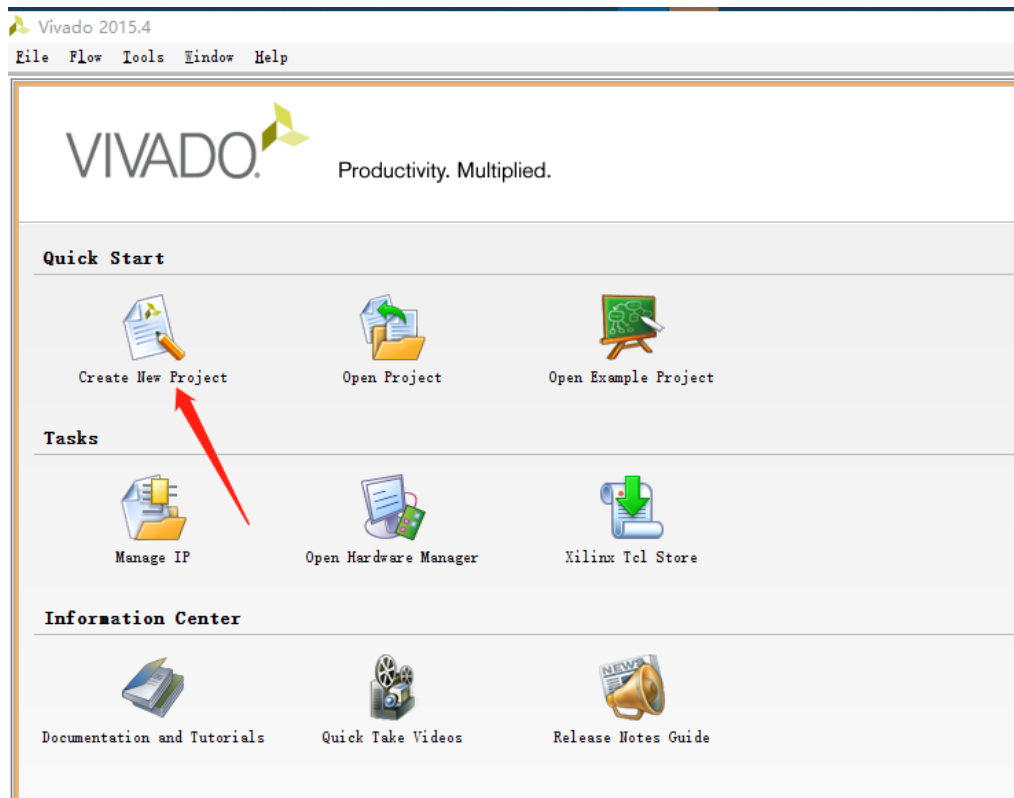
图 12-4 4X4 矩阵键盘电路原理图

信号名称	FPGA I/O 名称	功能说明
KEY_R[0]	D17	Keypad row[0]
KEY_R[1]	C17	Keypad row[1]
KEY_R[2]	E16	Keypad row[2]
KEY_R[3]	H17	Keypad row[3]
KEY_C[0]	G21	Keypad col[0]
KEY_C[1]	G16	Keypad col[1]
KEY_C[2]	A17	Keypad col[2]
KEY_C[3]	B17	Keypad col[3]

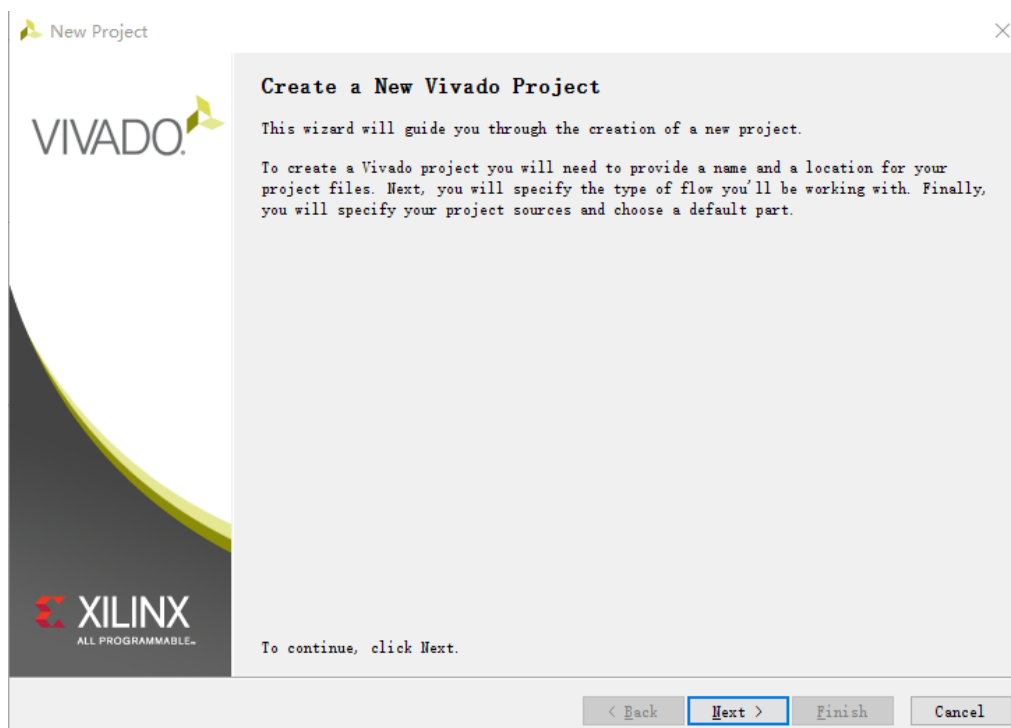
表 12-1 4X4 矩阵键与 FPGA 的管脚连接表

四、 实验步骤：

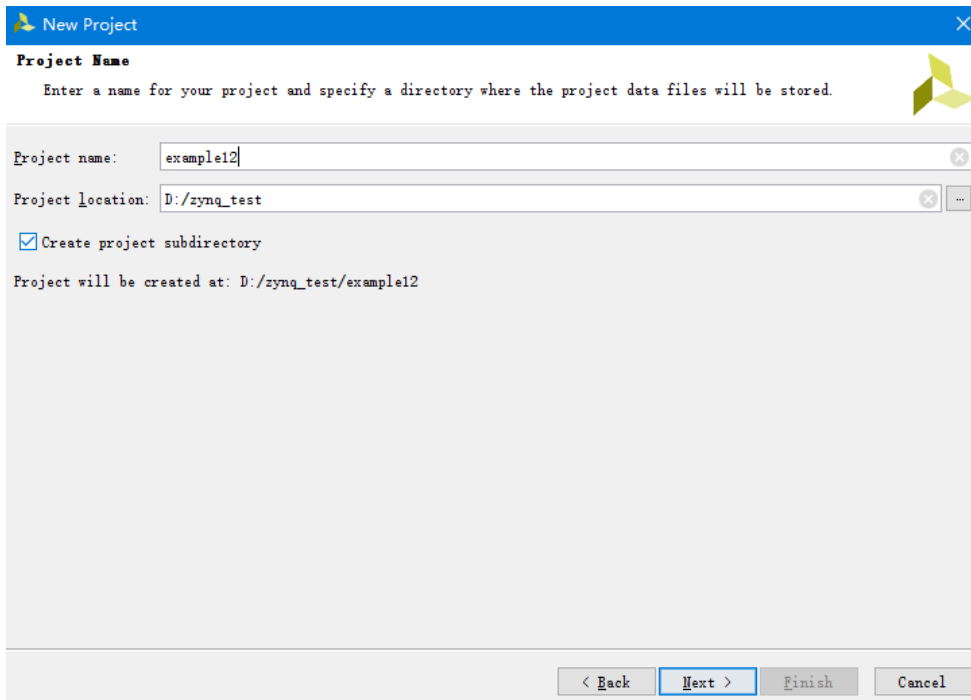
1. 打开 VIVADO 软件，新建一个工程。双击 vivado 软件后点击 create new project



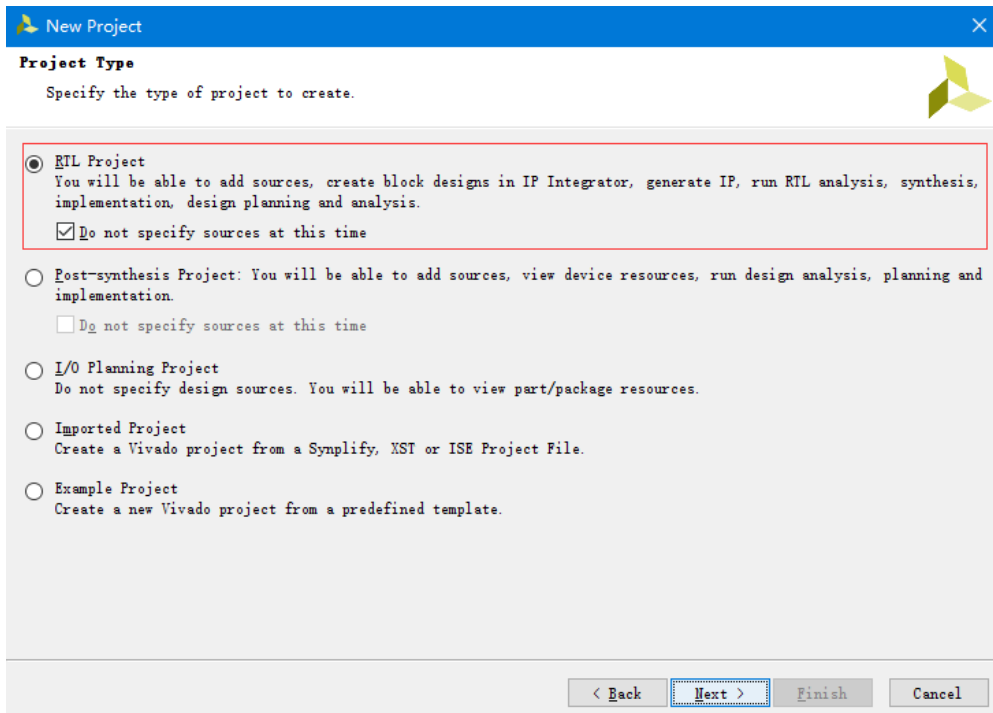
单击 next



为工程命名，指定工程路径。在 project name 栏输入工程名称，project location 栏指定工程路径，可以手动输入，也可以点击此栏尾部的...进行选择。操作完成后点击 next。

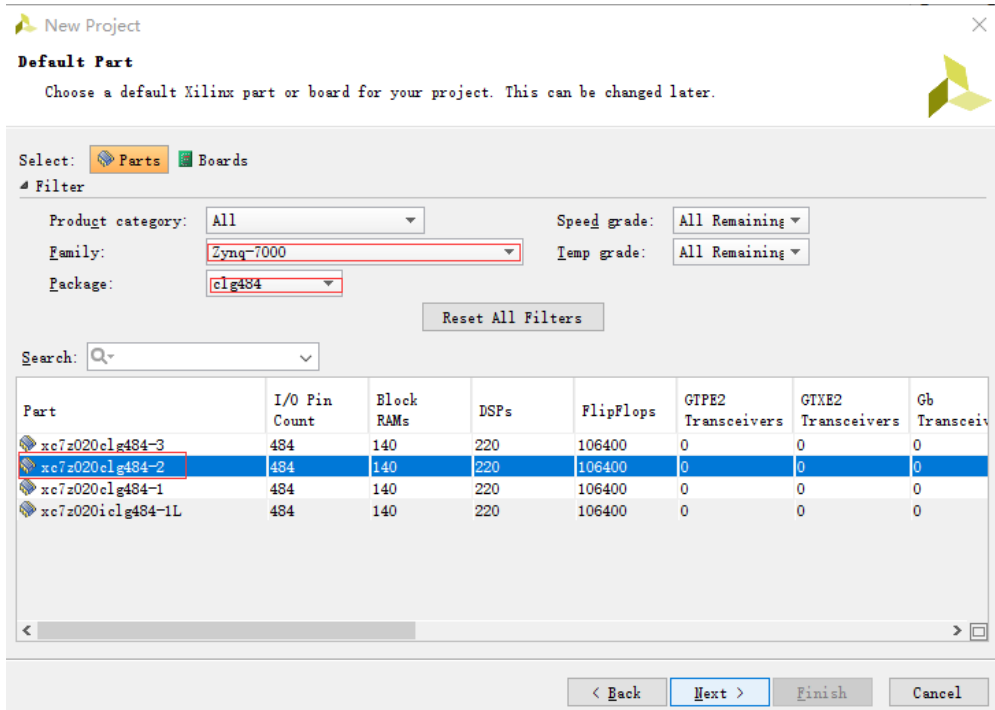


选择工程类型 默认选择 RTL 工程即可。单击 next。

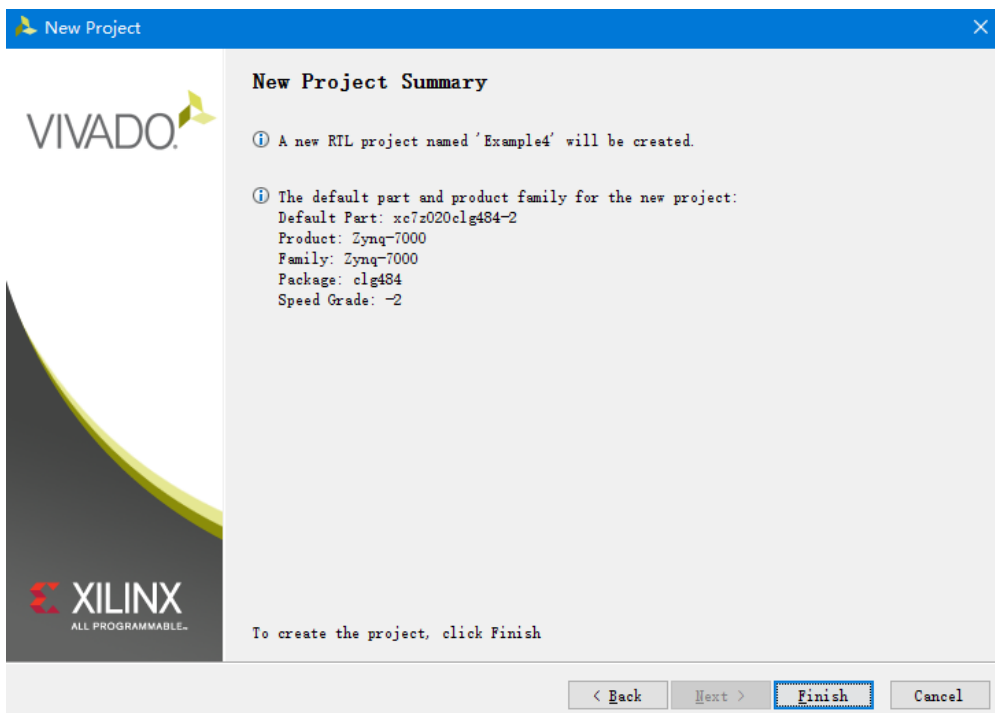


选择器件类型，快速锁定芯片方式如下，family 一栏中下拉菜单中选中 zynq-7000，package 一栏选择 clg484，此刻在下方器件列表总可以看到已经被筛选过的器件剩下四个，我们选择 xc7z020clg484-2 即可。单击 next。

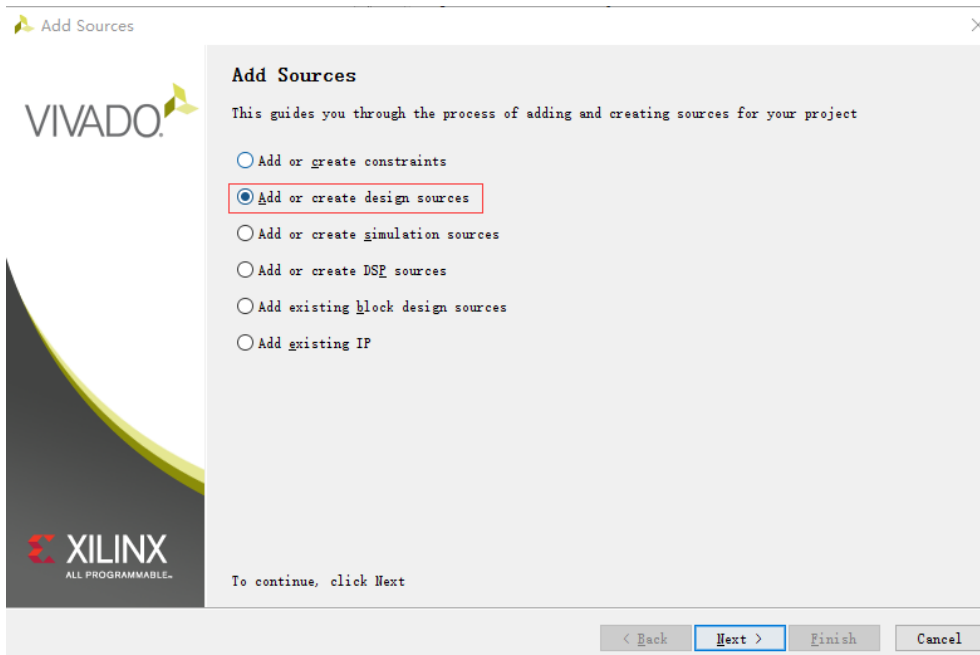




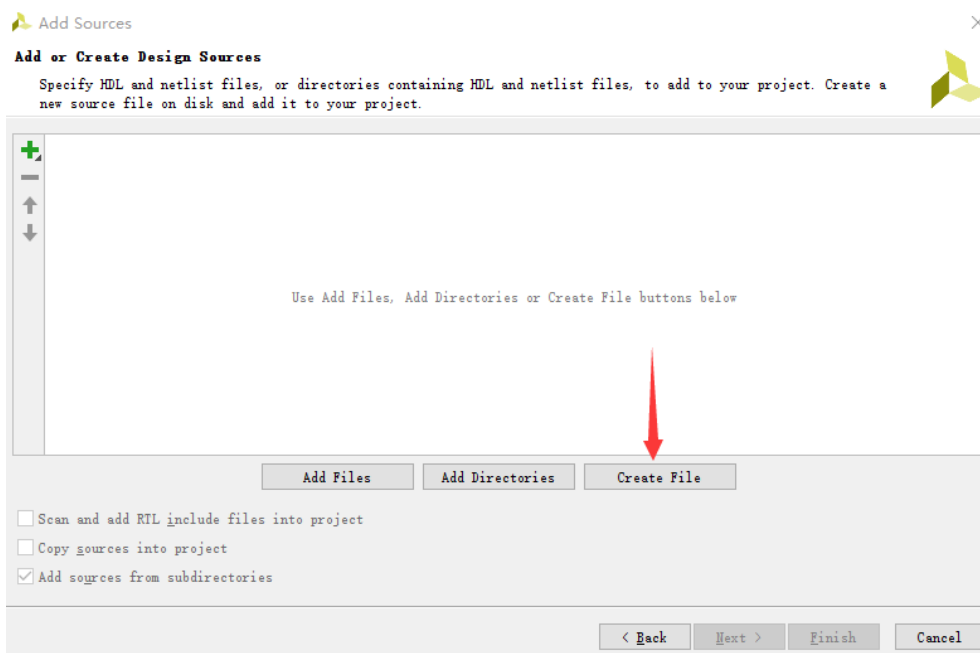
新建工程信息摘要，确认一遍器件信息是否正确，确认无误之后点击 finish。



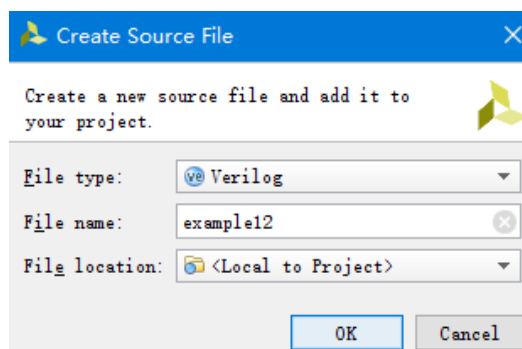
- 建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。操作步骤如下，点击 File-add sources 如下图，选中 add or create design sources，单击 next。



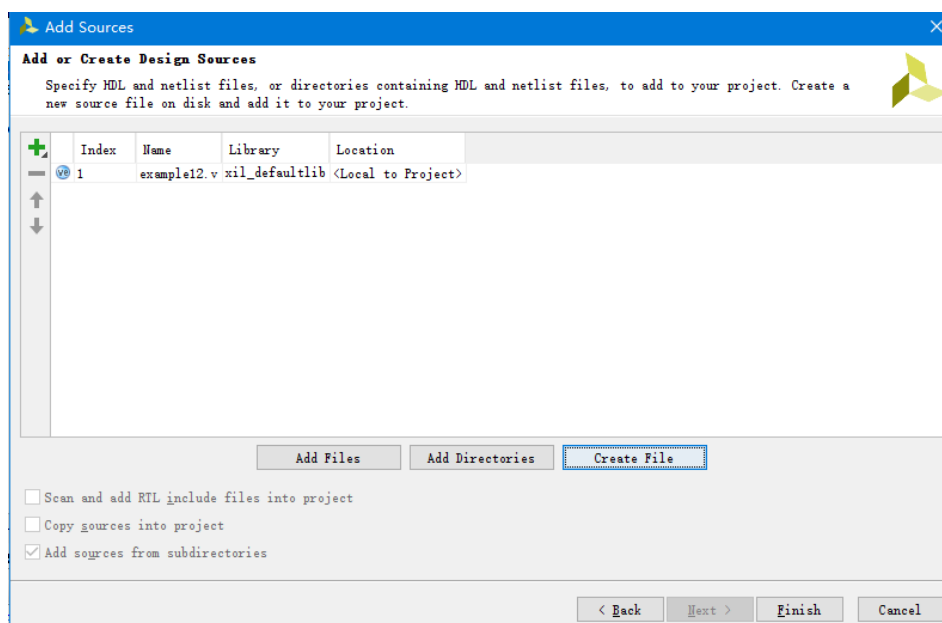
点击 create File 创建文件。



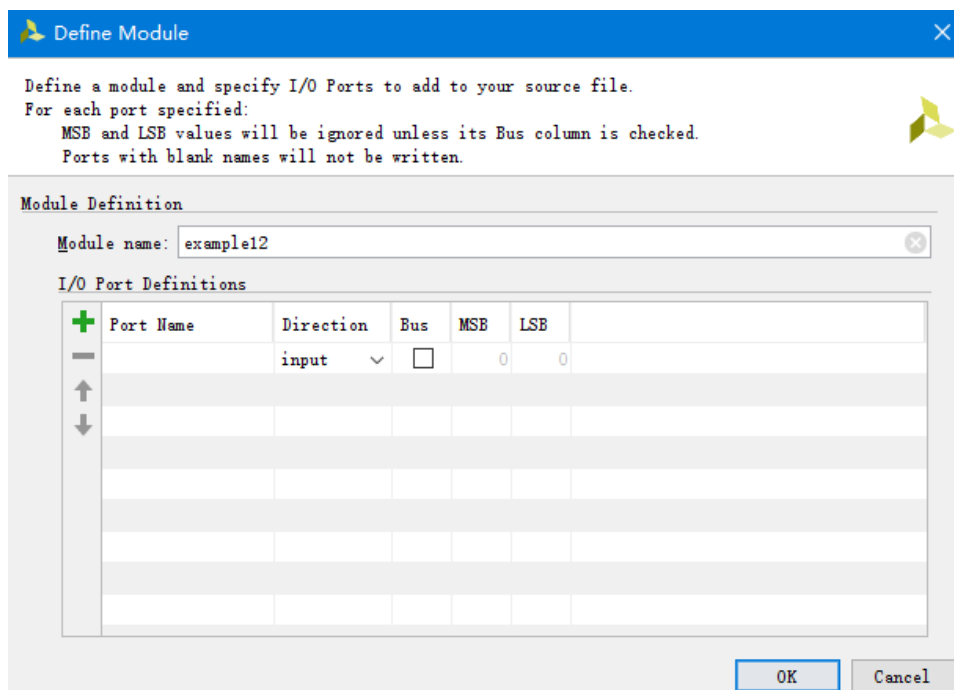
选择需要创建的文件类型 Verilog，输入文件名称，建议与工程名相同。点击 OK。



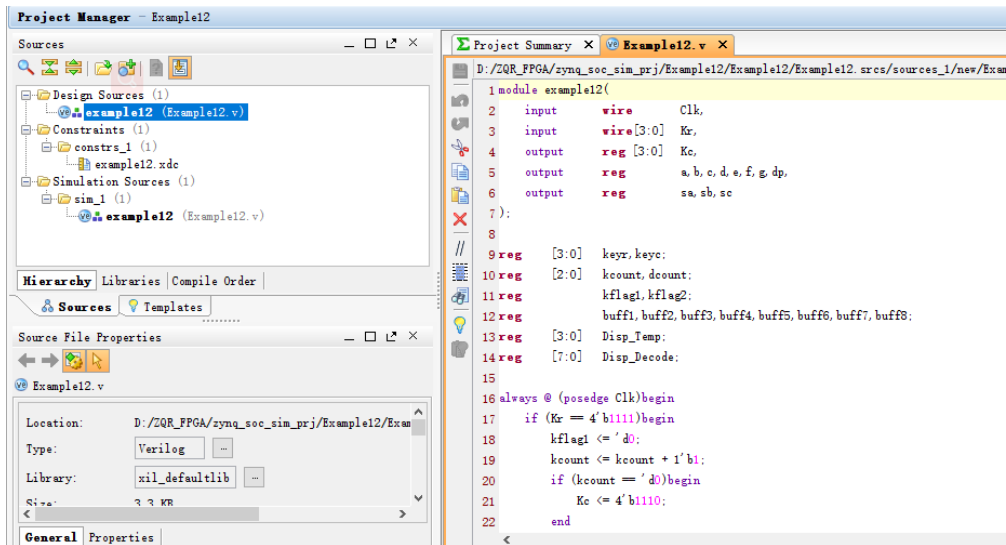
单击 finish。



定义设计模块端口,通过+ -添加和删除端口,在 port name 一栏中输入端口名称,direction 一栏中选择端口类型,可以定义为 input、output、inout 三种类型。如果是多比特信号可以勾选 bus 后更爱 MSB 和 LSB 实现多比特信号定义。填写完毕后点击 OK。软件会自动为你生成带有端口的 Verilog 设计文件。



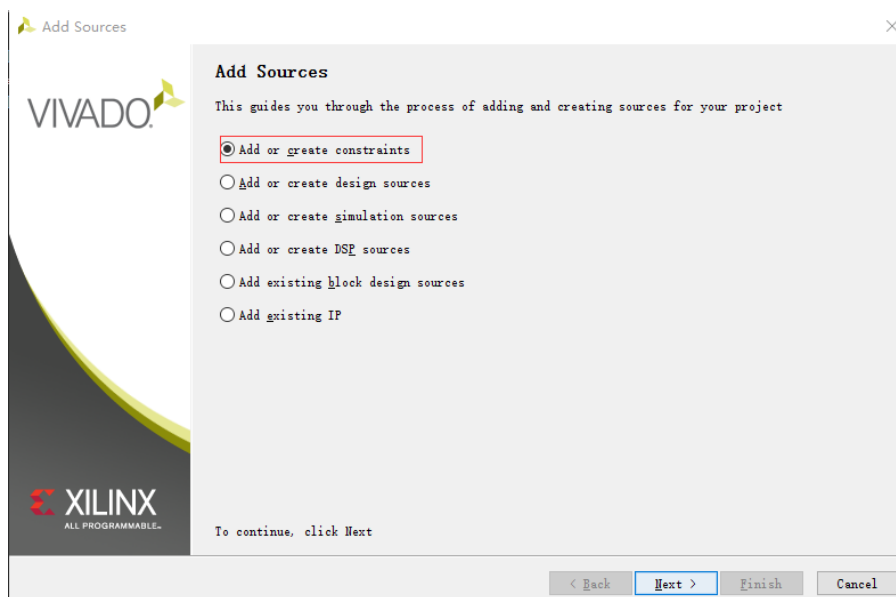
3. 按照实验原理和自己的想法,在 VERILOG 编辑窗口编写 VERILOG 程序,用户可参照光盘中提供的示例程序。



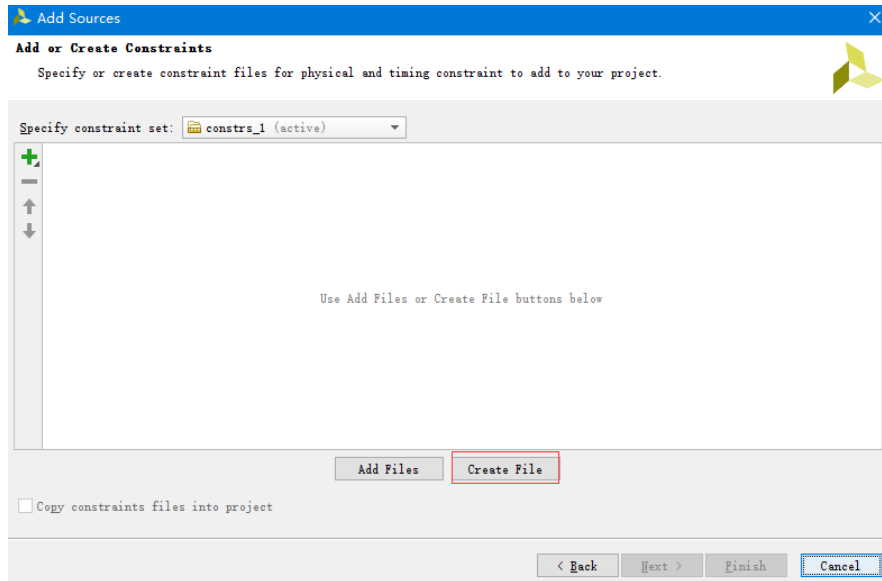
4. 编写完 VERILOG 程序后，保存起来。
5. 对自己编写的 VERILOG 程序进行综合并仿真，对程序的错误进行修改。
6. 综合仿真无误后，依照时钟，矩阵键盘、数码管与 FPGA 的管脚连接表 12-2 分配。

分配管脚这里使用实验一中提到的添加约束文件。操作方法如下：

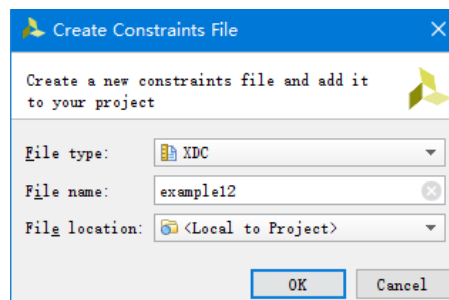
File-add sources，选中 add or create constraints，点击 next。



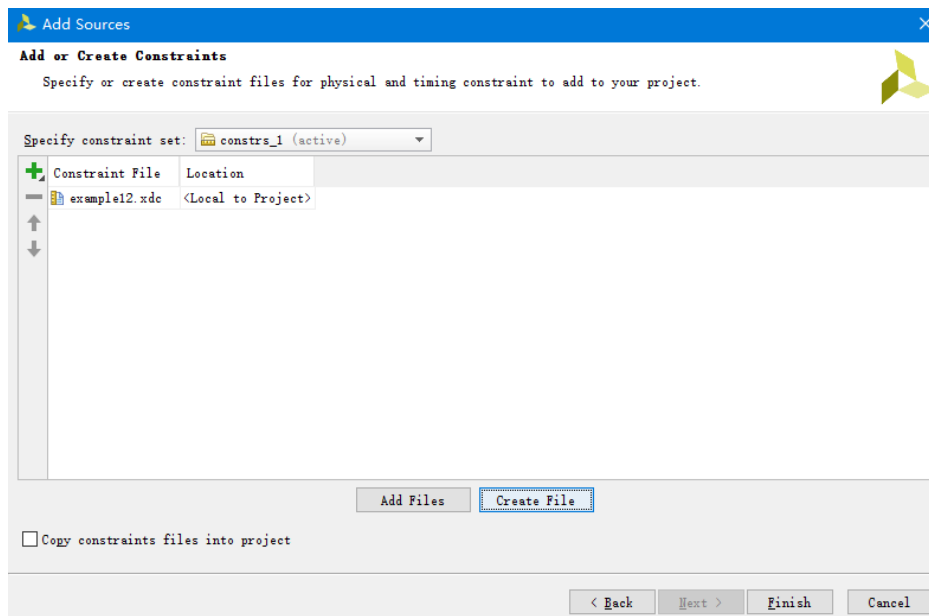
点击 create File



为新建的约束文件命名，建议与工程名字相同。方便文件管理。点击 OK



点击 finish



- 请在创建的 example12.xdc 文件中编写管脚约束。具体编写方法请参照实验一中的编写说明。建议拷贝实验一中的约束文件，依照如下表格中的管脚进行更改。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
CLK	数字信号源	Y8	时钟为 1KHZ
KR0	4*4 矩阵键盘 R0	D17	矩阵键盘行信号
KR1	4*4 矩阵键盘 R1	C17	
KR2	4*4 矩阵键盘 R2	E16	
KR3	4*4 矩阵键盘 R3	H17	
KC0	4*4 矩阵键盘 C0	G21	矩阵键盘列信号
KC1	4*4 矩阵键盘 C1	G16	
KC2	4*4 矩阵键盘 C2	A17	
KC3	4*4 矩阵键盘 C3	B17	
A	数码管模块 A 段	P16	键值显示
B	数码管模块 B 段	P17	
C	数码管模块 C 段	N17	
D	数码管模块 D 段	N15	
E	数码管模块 E 段	M15	
F	数码管模块 F 段	L17	
G	数码管模块 G 段	L18	
DP	数码管模块 DP 段	K19	
SA	数码管模块 SEL0	L22	
SB	数码管模块 SEL1	P21	
SC	数码管模块 SEL2	N20	

表 12-2 端口管脚分配表

- 8、 用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

## 五、 实验结果与现象

以设计的参考示例为例，当设计文件加载到目标器件后，将数字信号源模块的时钟选择为 1KHZ，按下矩阵键盘的某一个键，则在数码管上显示对应的这个键标识的键值，当再按下第二个键的时候前一个键的键值在数码管上左移一位。按下“\*”键则在数码管是显示“E”键值。按下“#”键在数码管上显示“F”键值。

## 六、 实验报告

- 1、 绘出不同的键值时的数码管的仿真波形，并作说明。
- 2、 根据自己的思路，找一找还有没有其它方法进行键盘的扫描显示。并画出流程图。
- 3、 将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。

# 实验十三 16\*16 点阵显示实验

## 一、 实验目的

- 1、 了解点阵字符产生和显示原理和系统的 16\*16 点阵的工作机理。
- 2、 加强对总线产生、地址定位的 FPGA 实现方法的理解。
- 3、 掌握在 FPGA 中调用 ROM 的使用方法。

## 二、 实验原理

本实验主要完成汉字字符在 LED 上的显示，16\*16 扫描 LED 点阵的工作原理与 8 位扫描数码管类似，只是显示的方式与结果不一样而已。下面就本实验系统的 16\*16 点阵的工件原理做一些简单的说明。

16\*16 点阵由此 256 个 LED 通过排列组合而形成 16 行\*16 列的一个矩阵式的 LED 阵列，俗称 16\*16 点阵。单个的 LED 的电路如下图 13-1 所示：



图 13-1 单个 LED 电路图

由上图可知，对于单个 LED 的电路图当 Rn 输入一个高电平，同时 Cn 输入一个低电平时，电路形成一个回路，LED 发光。也就是 LED 点阵对应的这个点被点亮。16\*16 点阵也就是由 16 行和 16 列的 LED 组成，其中每一行的所有 16 个 LED 的 Rn 端并联在一起，每一列的所有 16 个 LED 的 Cn 端并联在一起。通过给 Rn 输入一个高电平，也就相当于给这一列所有 LED 输入了一个高电平，这时只要某个 LED 的 Cn 端输入一个低电平时，对应的 LED 就会被点亮。具体的电路如下图 13-2 所示：

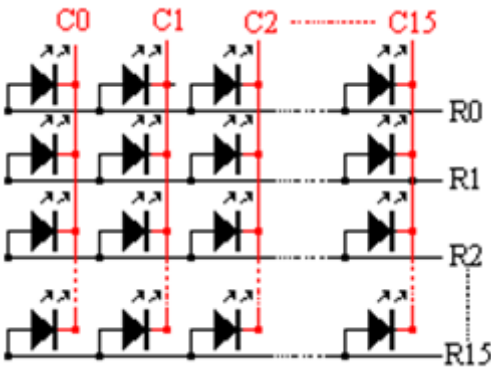


图 13-2 16\*16 点阵电路原理图

在点阵上显示字符是根据其字符在点阵上显示点的亮灭来表示的如下图 13-3 所示：



图 13-3 字符在点阵上的显示

在上图中，显示的是一个“汉”字，只要将被“汉”字所覆盖的区域的点点亮，则在点阵中就会显示一个“汉”字。根据前面我们所介绍的点阵显示的原理。当我们选中第一列后，根据要显示汉字的第一列中所需被点亮的点对应的  $R_n$  置为高电平，则在第一列中需要被点亮的点就会被点亮。依此类推，显示第二列、第三列……第  $N$  列中需要被点亮的点。然后根据人眼的视觉原理，将每一列显示的点的间隔时间设为一定的值，那么我们会感觉显示一个完整的不闪烁的汉字。同时也可以按照这个原理来显示其它的汉字。下图 11-4 是一个汉字显示所需要的时序图：

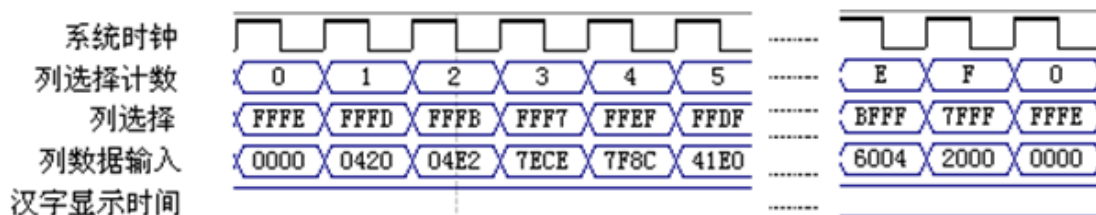


图 13-4 显示时序图

在上图中，在系统时钟的作用下，首先选取其中的一列，将数据输入让这列的 LED 显示其数据（当为高电平时 LED 发光，否则不发光）。然后选取下一列来显示下一列的数据。当完成一个  $16 \times 16$  点阵的数据输入时，即列选择计数到最后一列后，再从第一列开始输入相同的数据。这样只要第一次显示第一列的数据和第二次显示第一列的数据的时间足够短，那么人的眼睛就会看到第一列的数据总是显示的，而没有停顿现象。同样的道理其它列也是这样，直到显示下一个汉字。

在实际的运用当中，一个汉字是由多个八位的数据来构成的，那么要显示多个汉字的时候，这些数据可以根据一定的规则存放到存储器中，当要显示这个汉字的时候只要将存储器中对应的数据取出显示即可。本实验的示例程序依次显示的是“欢迎使用嵌入式开发系统”。数据量不大，所以没有放入存储器中，而在程序中直接输入对应的一个 16 位的数据。示例程序的字库数据的格式如下图 13-5 所示：



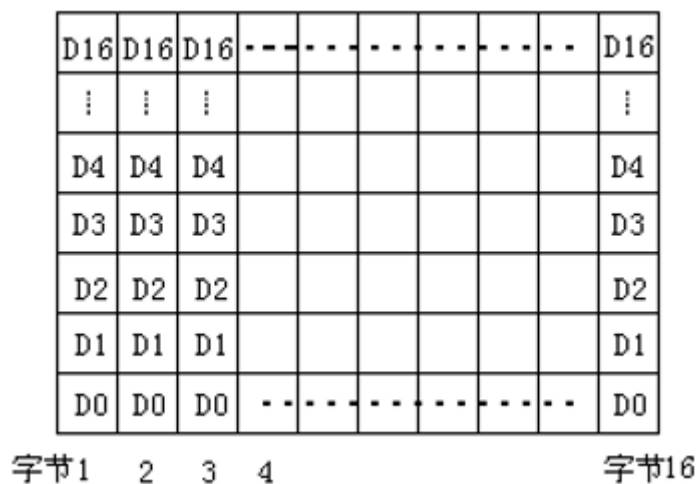


图 13-5 字库格式

### 三、 实验内容

本实验要求完成的任务是通过编程实现对 16X16 点阵的控制。在点阵的循环显示“欢迎使用嵌入式开发系统”这几个汉字和字符。

16\*16 点阵的电路原理在前面已经做了详尽的说明，在此实验中，16\*16 点阵由 4 个 8\*8 点阵组成，考虑到 LED 电流功耗与 FPGA 电流功耗的关系，在实验的电路中加入驱动电路。

具体电路如下图 13-6 所示。与 FPGA 的管脚连接如表 13-1 所示。

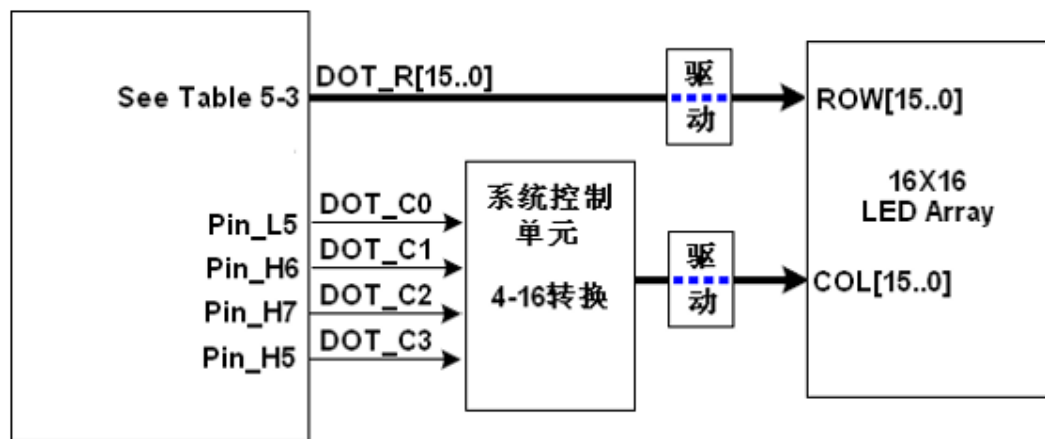


图 13-6 16\*16 点阵电路图

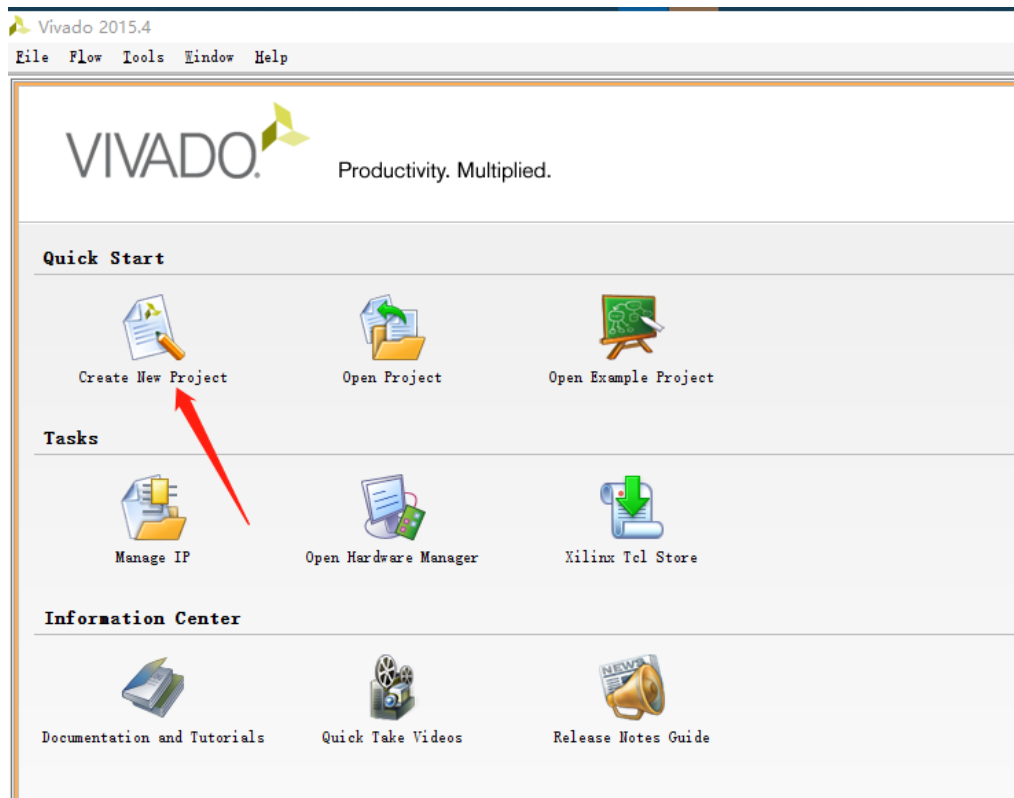
信号名称	FPGA I/O 名称	功能说明
DOT_R[0]	AB14	Dot array Row  Data
DOT_R[1]	W16	
DOT_R[2]	U17	
DOT_R[3]	U15	

DOT_R[4]	W15	
DOT_R[5]	Y14	
DOT_R[6]	V14	
DOT_R[7]	V13	
DOT_R[8]	T16	
DOT_R[9]	M19	
DOT_R[10]	N19	
DOT_R[11]	P20	
DOT_R[12]	L21	
DOT_R[13]	M21	
DOT_R[14]	W17	
DOT_R[15]	AA16	
DOT_C0	AB15	Select Col
DOT_C1	AB16	
DOT_C2	W18	
DOT_C3	AB17	

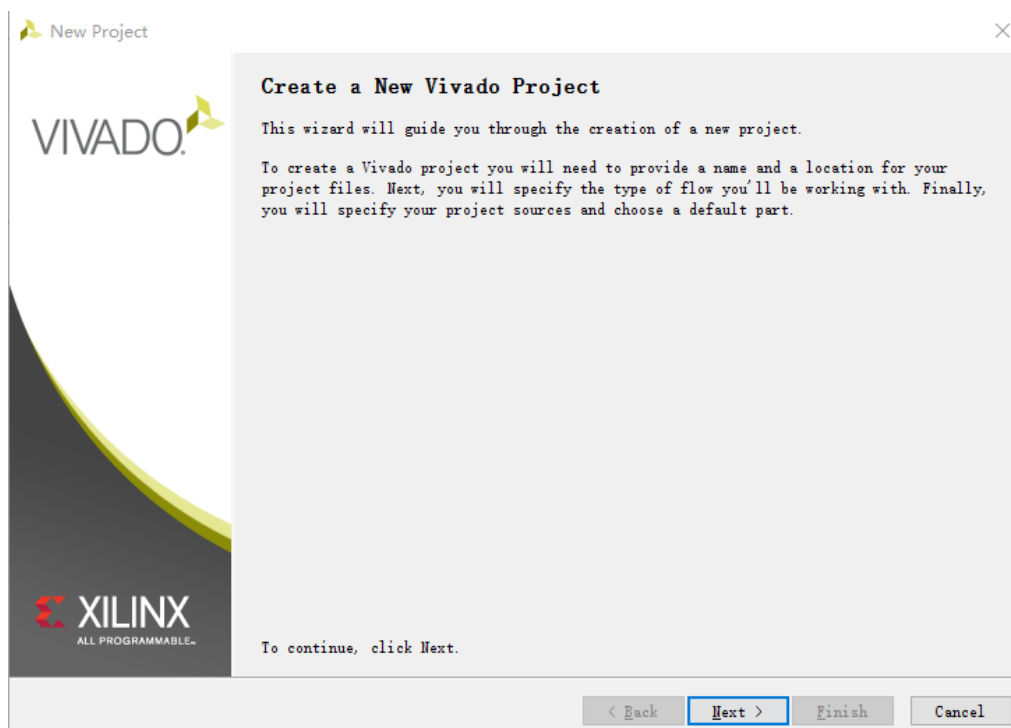
表 13-1 16X16 点阵与 FPGA 的管脚连接表

#### 四、 实验步骤：

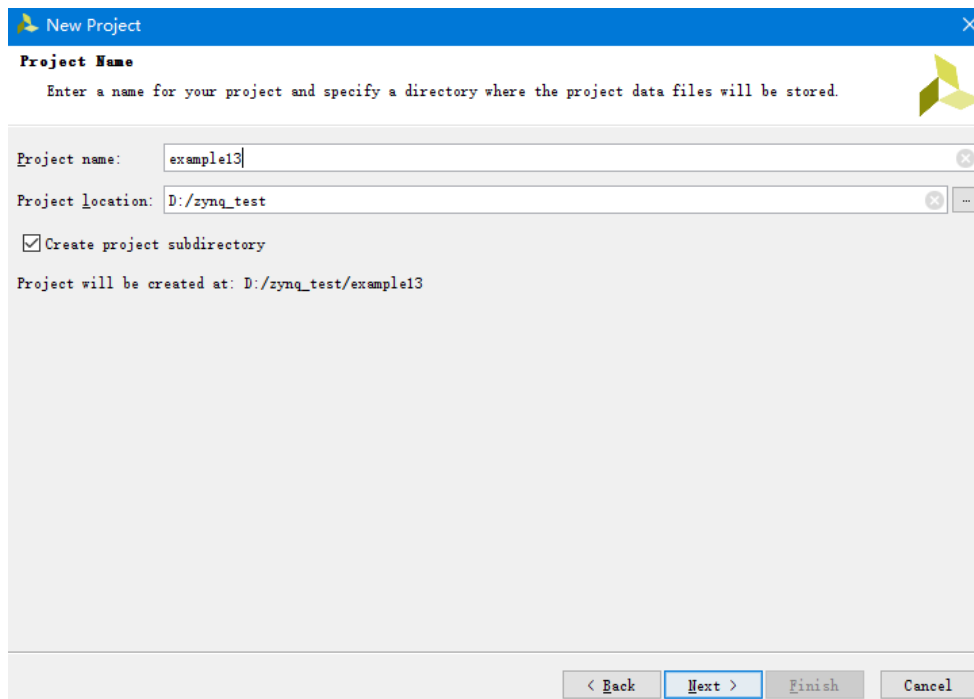
1. 打开 VIVADO 软件，新建一个工程。双击 vivado 软件后点击 create new project



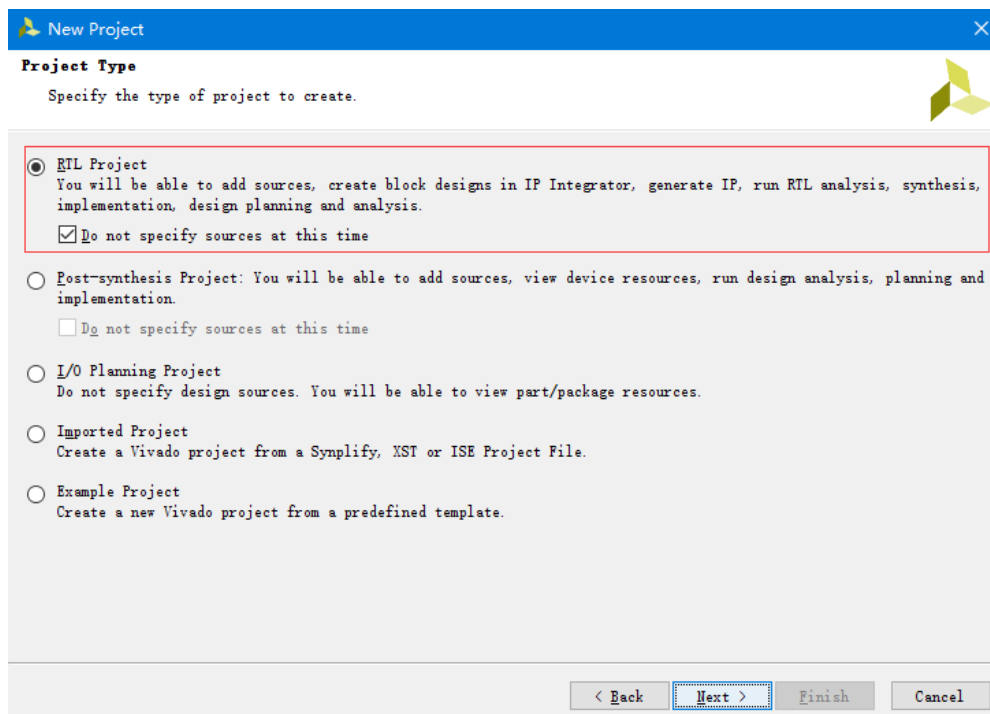
单击 next



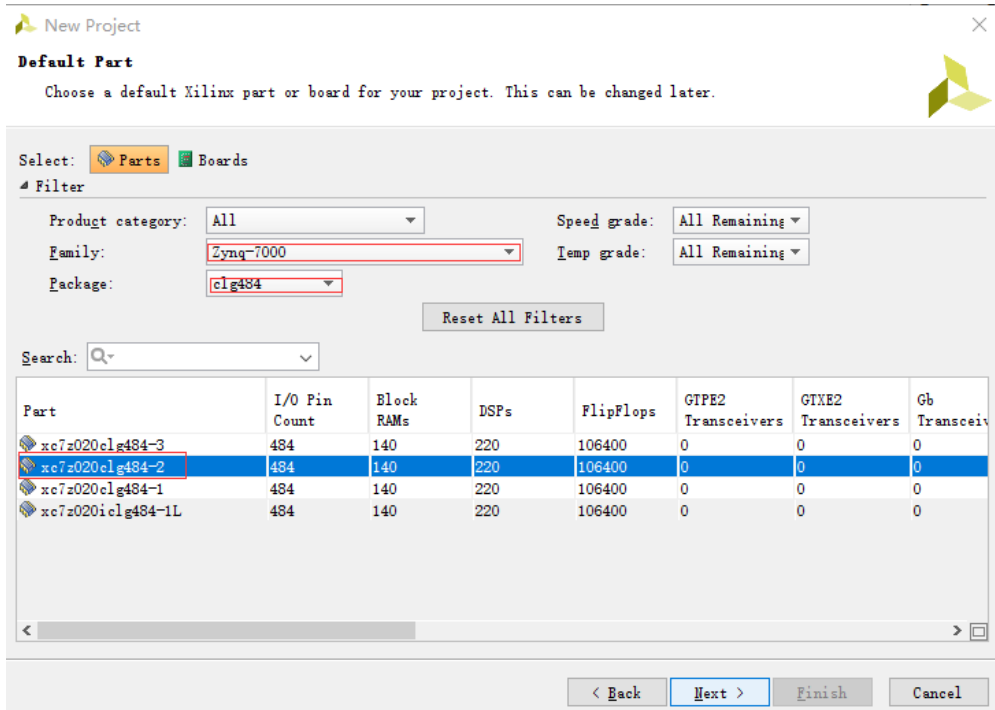
为工程命名，指定工程路径。在 project name 栏输入工程名称，project location 栏指定工程路径，可以手动输入，也可以点击此栏尾部的...进行选择。操作完成后点击 next。



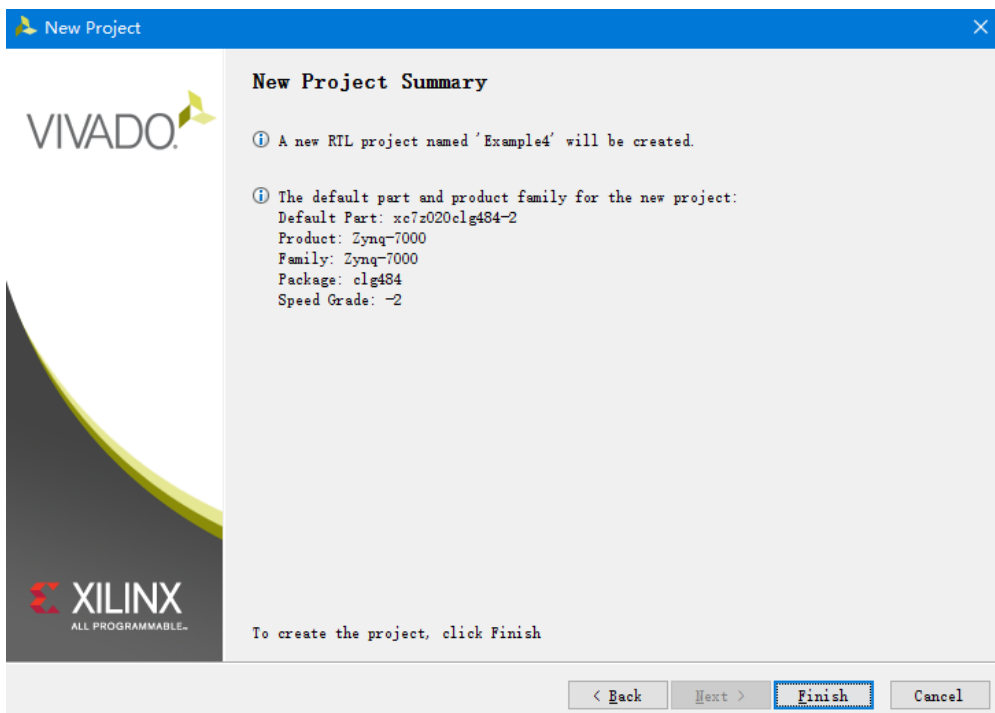
选择工程类型 默认选择 RTL 工程即可。单击 next。



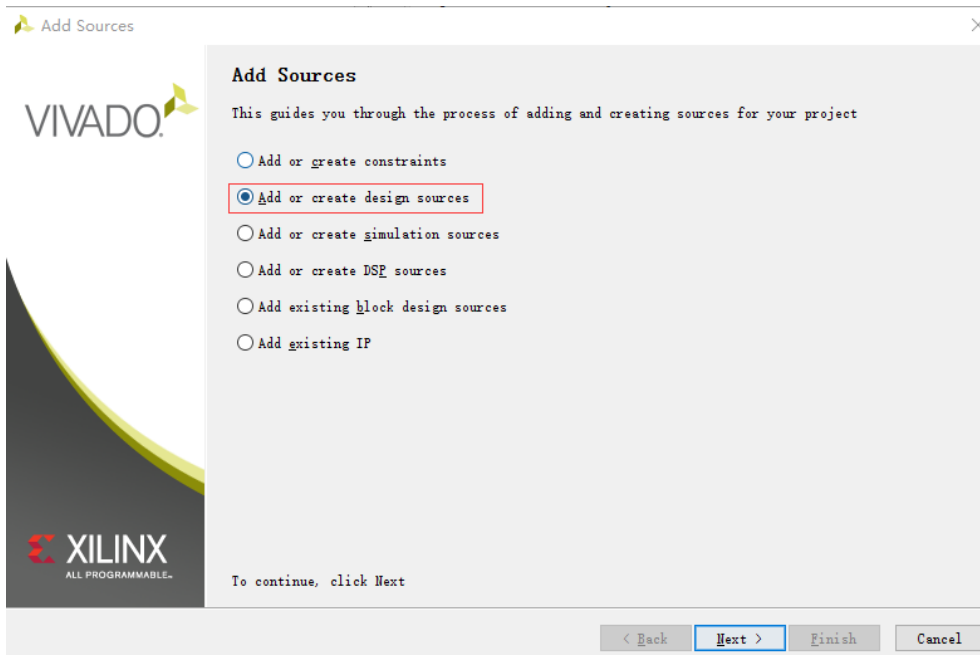
选择器件类型，快速锁定芯片方式如下，family 一栏中下拉菜单中选中 zynq-7000，package 一栏选择 clg484，此刻在下方器件列表总可以看到已经被筛选过的器件剩下四个，我们选择 xc7z020clg484-2 即可。单击 next。



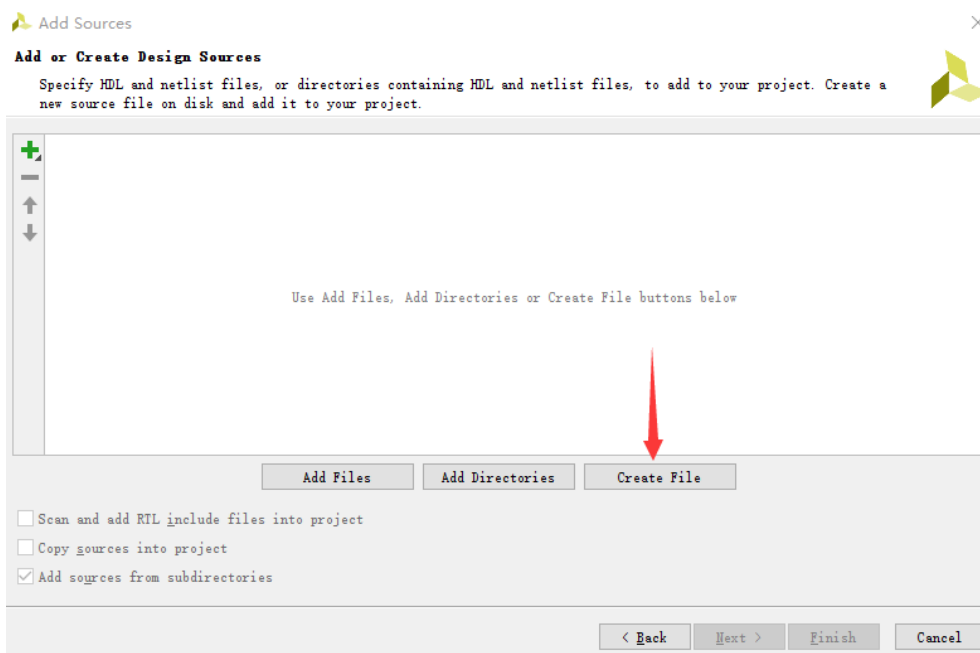
新建工程信息摘要，确认一遍器件信息是否正确，确认无误之后点击 finish。



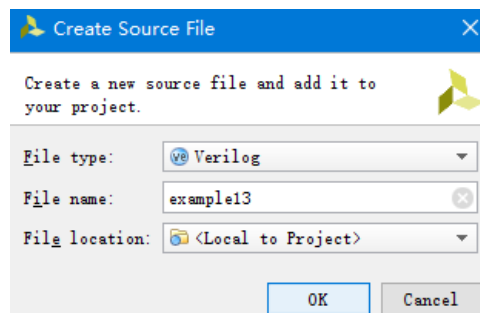
- 建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。操作步骤如下，点击 File-add sources 如下图，选中 add or create design sources，单击 next。



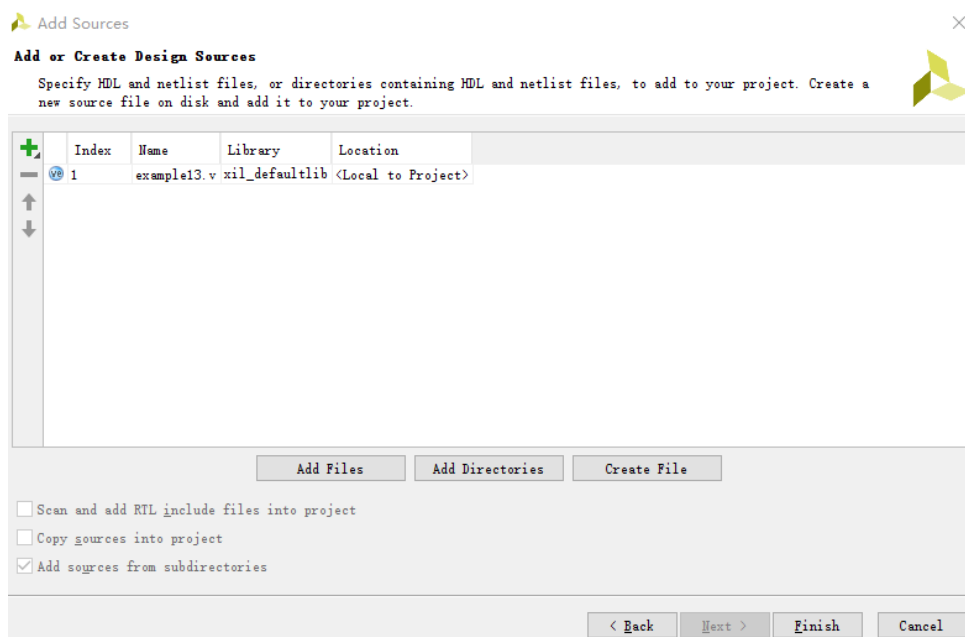
点击 create File 创建文件。



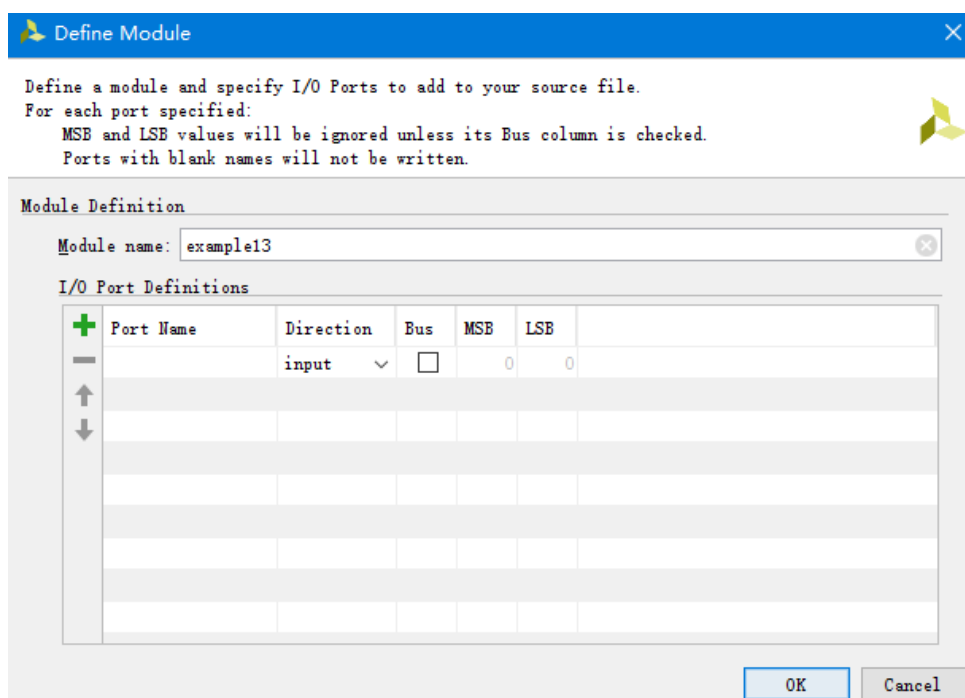
选择需要创建的文件类型 Verilog，输入文件名称，建议与工程名相同。点击 OK。



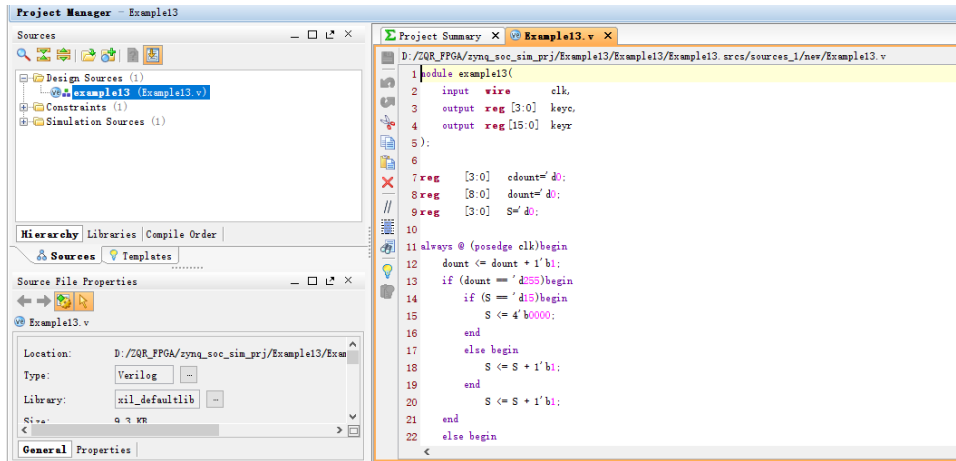
单击 finish。



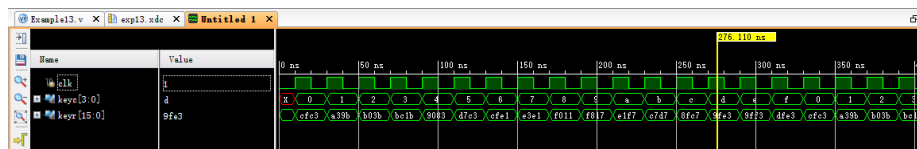
定义设计模块端口,通过+ -添加和删除端口,在 port name 一栏中输入端口名称,direction 一栏中选择端口类型,可以定义为 input、output、inout 三种类型。如果是多比特信号可以勾选 bus 后更爱 MSB 和 LSB 实现多比特信号定义。填写完毕后点击 OK。软件会自动为你生成带有端口的 Verilog 设计文件。



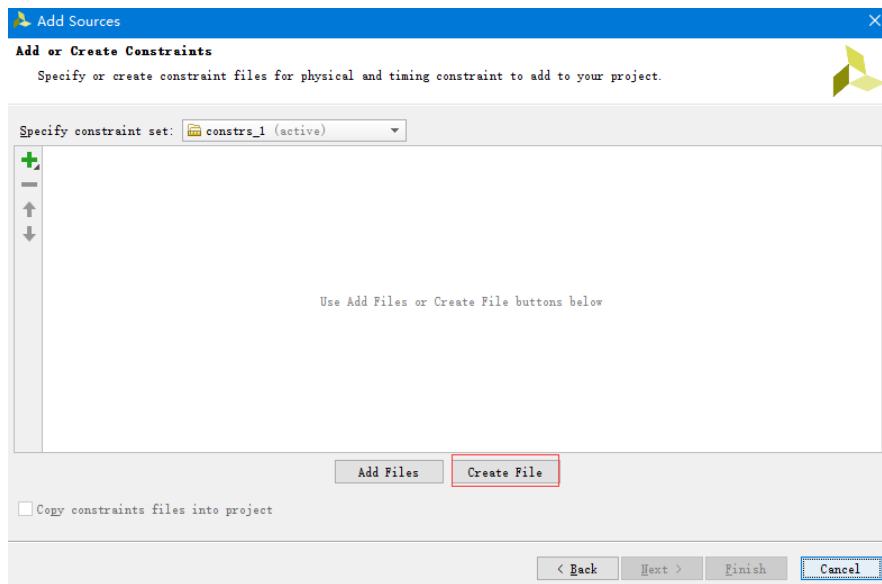
3. 按照实验原理和自己的想法,在 VERILOG 编辑窗口编写 VERILOG 程序。



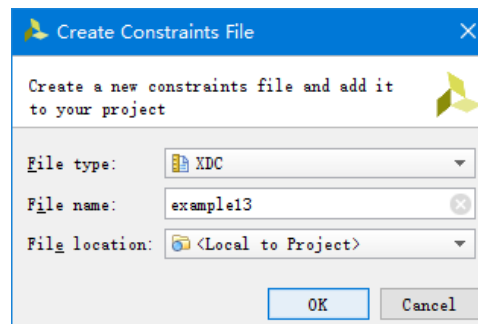
4. 编写完 VERILOG 程序后，保存起来。
5. 对自己编写的 VERILOG 程序进行综合并仿真，对程序的错误进行修改。



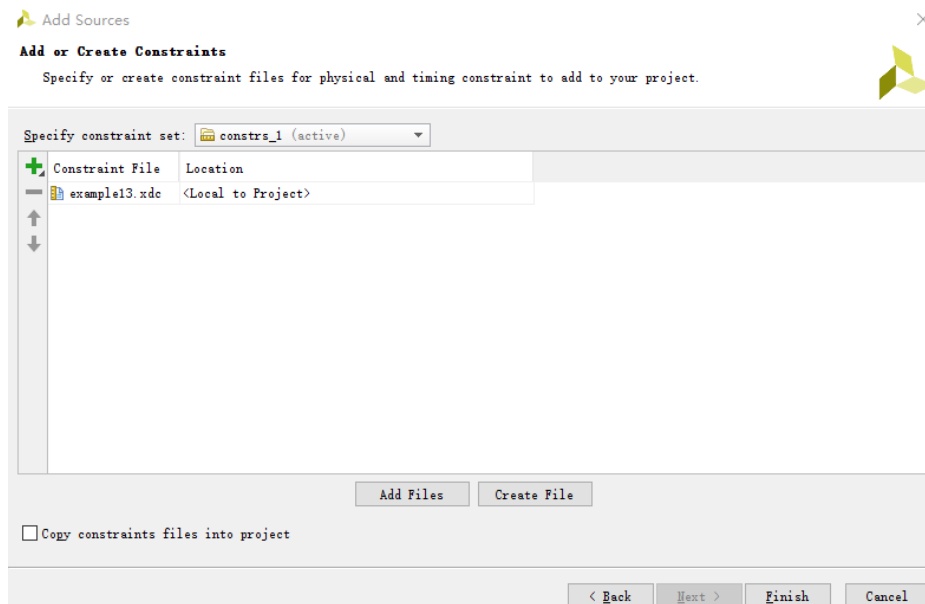




为新建的约束文件命名，建议与工程名字相同。方便文件管理。点击 OK



点击 finish



7. 请在创建的 example13.xdc 文件中编写管脚约束。具体编写方法请参照实验一中的编写说明。建议拷贝实验一中的约束文件，依照如下表格中的管脚进行更改。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
CLK	数字信号源	Y8	时钟为 1KHZ
KEYC0-C3	16*16 点阵	见表 13-1	点阵 C0-C4 端口
KEYR0-R15	16*16 点阵	见表 13-1	点阵 R0-R15 端口

表 13-2 端口管脚分配表

- 8、用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

## 五、 实验结果与现象

以设计的参考示例为例，当设计文件加载到目标器件后，将数字信号源模块的时钟选择为 1KHZ，在点阵模块循环依次显示“欢迎使用嵌入式开发系统”每个字符的显示约为 0.5 秒。最后显示的时间约为 2 秒，然后再次显示“欢迎使用嵌入式开发系统”。

## 六、 实验报告

- 1、在这个程序的基础上试写出其它汉字的字库并在点阵上显示出来。
- 2、思考怎样让汉字旋转和左右移动。
- 3、试利用 FPGA 的 ROM 将字库存入 ROM，然后再调用的形式编写程序。
- 4、绘出仿真波形图，并加以说明。

## 实验十四 直流电机的测速实验

### 一、 实验目的

- 1、 掌握直流电机的工作原理。
- 2、 了解开关型霍尔传感器的工作原理和使用方法。
- 3、 掌握电机测速的原理。

### 二、 实验原理

直流电机是我们生活当中常用的一种电子设备。其内部结构如下图 14-1 所示：



图 14-1 直流电机结构图

下面就上图来说明直流电机的工作原理。将直流电源通过电刷接通电枢绕组，使电枢导体有电流流过，由于电磁作用，这样电枢导体将会产生磁场。同时产生的磁场与主磁极的的磁场产生电磁力，这个电磁力作用于转子，使转子以一定的速度开始旋转。这样电机就开始工作。

为了能够测定出电机在单位时间内转子旋转了多少个周期，我们在电机的外部电路中加入了一个开关型的霍尔原件（44E），同时在电子转子上的转盘上加入了一个能够使霍尔原件产生输出的带有磁场的磁钢片。当电机旋转时，带动转盘是的磁钢片一起旋转，当磁钢片旋转到霍尔器件的上方时，可以导致霍尔器件的输出端高电平变为低电平。当磁钢片转过霍尔器件上方后，霍尔器件的输出端又恢复高电平输出。这样电机每旋转一周，则会使霍尔器件的输出端产生一个低脉冲，我们就可以通过检测单位时间内霍尔器件输出端低脉冲的个数来推算出直流电机在单位时间内的转速。直流电机和开关型霍尔器件的电路原理图如下图 14-2 所示：

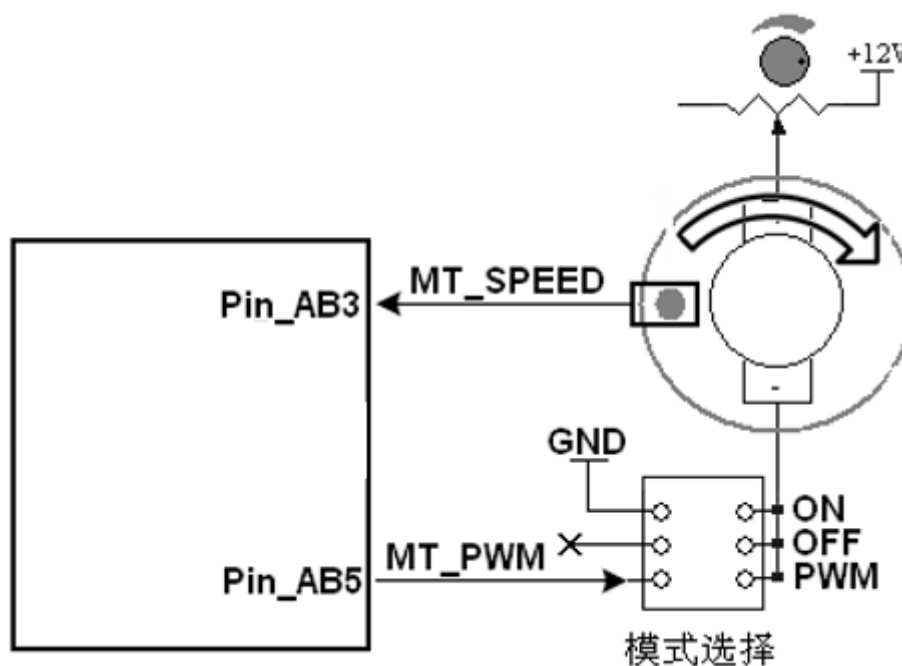


图 14-2 直流电机、霍尔器件电路图

电机的转速通常是指每分钟电机的转速，也就是单位为 rpm，实际测量过程中，为了减少转速刷新的时间，通常都是 5~10 秒刷新一次。如果每 6 秒钟刷新一次，那么相当于只记录了 6 秒钟内的电机转数，把记录的数据乘 10 即得到一分钟的转速。最后将这个数据在数码管上显示出来。

最后显示的数据因为是将数据乘以 10，也就是将个位数据的后面加上一位来做个位即可，这一位将一直为 0。如：45\*10 变为 450，即为在“45”个位后加了一位“0”。由此可知，这个电机的转速的误差将是 20 以内。为了使显示的数据能够在数码管是显示稳定，在这个数据的输出时加入了一个 16 位的锁存器，把锁存的数据送给数码管显示，这样就会因为在计数过程中，数据的变化而使数码管显示不断变化。

### 三、 实验内容

本实验要求完成的任务是通过编程实现电机转数读取，并在数码管上显示。其读取数据和显示数据的时序关系如下图 14-3 所示：

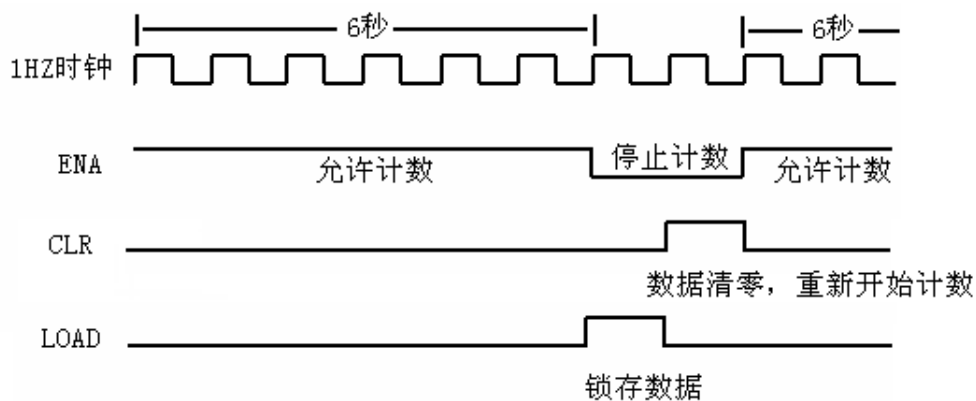


图 14-3 实验控制信号时序图

在此实验中数码管与 FPGA 的连接电路和管脚连接在以前的实验中都做了详细说明，这里不在赘述。直流电机和霍尔器件与 FPGA 的管脚连接如表 14-1 所示。

信号名称	对应 FPGA 管脚名	说明
MT-PWM	R21	PWM 信号输入至直流电机
MT-SPEED	V5	霍尔器尔器件输出至 FPGA

表 14-1 直流电机、霍尔器件与 FPGA 的管脚连接表

#### 四、 实验步骤

- 1、 打开 VIVADO 软件，新建一个工程。
- 2、 建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。
- 3、 按照实验原理和自己的想法，在 VERILOG 编辑窗口编写 VERILOG 程序。示例程序共提供 4 个 VERILOG 源程序。每一个源程序完成一定的功能。其具体的功能如下表 14-2:

文件名称	完成功能
TELTCL.V	在时钟的作用下生成测频的控制信号。
CNT10.V	十进制计数器。在实验中使用 4 个来进行计数
SEG32B.V	16 位的锁存器，在锁存控制信号的作用下，将计数的值锁存
DISPLAY.V	显示译码，将锁存的数据显示出来。

表 14-2 示例程序功能表

- 4、 编写完 VERILOG 程序后，保存起来。方法同实验一。
- 5、 将自己编辑好的的程序进行综合仿真，并对程序的错误进行修改，最终通过综合。
- 6、 综合仿真无误后，依照直流电机、霍尔器件、数码管与 FPGA 的管脚连接表 14-3 分配。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
CLK	数字信号源	Y8	时钟为 1MHZ

MOTOR	直流电机模块	V5	44E 脉冲输出
LEDAG0	数码管 A 段	P16	电机转速显示
LEDAG1	数码管 B 段	P17	
LEDAG2	数码管 C 段	N17	
LEDAG3	数码管 D 段	N15	
LEDAG4	数码管 E 段	M15	
LEDAG5	数码管 F 段	L17	
LEDAG6	数码管 G 段	L18	
LEDAG7	数码管 DP 段	K19	
SEL0	位选 DEL0	L22	
SEL1	位选 DEL1	P21	
SEL2	位选 DEL2	N20	

表 14-3 端口管脚分配表

- 7、用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致

## 五、 实验结果与现象

以设计的参考示例为例，当设计文件加载到目标器件后，将数字信号源模块的时钟选择为 1MHZ，将直流电机模块的模式选择到 GND 模式，旋转改变转速的电位器，使直流电机开始旋转，此时在一定的时间内，数码管上将显示此时直流电机的每分钟转速。通过电位器慢慢增加或者减少直流电机的转动速率，此时数码管上的数值也会相应的增加或者减少。

## 六、 实验报告

- 1、绘出仿真波形，并作说明。
- 2、试编写程序将实验的结果精确到个位。
- 3、将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。

## 实验十五 步进电机驱动控制

### 一、 实验目的

- 1、 了解步进电机的工作原理。
- 2、 掌握用 FPGA 产生驱动步进电机的时序。
- 3、 掌握用 FPGA 来控制步进电机转动的整个过程。

### 二、 实验原理

步进电机是工业过程控制及仪表中常用的控制元件之一，例如在机械装置中可以用丝杆把角度变为直线位移，也可以用步进电机带动螺旋电位器，调节电压或电源，从而实现对执行机械的控制。步进电机可以直接用数字信号驱动，使用非常方便。步进电机还具有快速启停、精确步进和定位等特点，因而在数控机床、绘图仪、打印机以及光学仪器中得到广泛的应用。

步进电机是工业控制及仪表中常用的控制元件之一，例如在机械装置中可以精确控制机械装置的旋转角度、移动距离等。步进电机可以直接用数字信号来驱动，使用非常方便。另外步进电机还具有快速起停、精确步进和定位的特点。

步进电机实际上是一个数据/角度转换器，三相步进电机的结构原理如下图 15-1 所示：

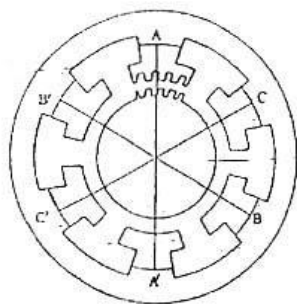


图 15-1 三相步进电机的结构示意图

从图中可以看出，电机的定子有六个等分的磁极，A、A'、B、B'、C、C'，相邻的两个磁极之间夹角为  $60^\circ$ ，相对的两个磁极组成一组（A—A'，B—B'，C—C'），当某一绕组有电流通过时，该绕组相应的两个磁极形成 N 极和 S 极，每个磁极上各有五个均分布的矩形小齿，电机的转子上有 40 个矩形小齿均匀地分布在圆周上，相邻两个齿之间夹角为  $9^\circ$ 。

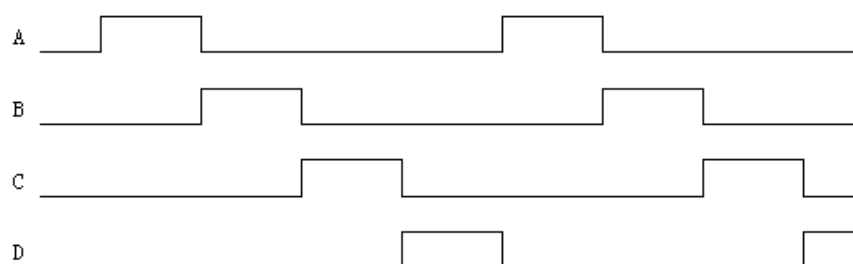
(1)当某一相绕组通电时，对应的磁极就产生磁场，并与转子转动一定的角度，使转子和定子的齿相互对齐。由此可见，错齿是促使步进电机旋转的原因。

例如在三相三拍控制方式中，若 A 相通电，B、C 相都不通电，在磁场作用下使转子齿和 A 相的定子齿对齐，我们以此作为初始状态。设与 A 相磁极中心线对齐的转子的齿为 0 号齿，由于 B 相磁极与 A 相磁极相差  $120^\circ$  不是  $9^\circ$  的整数倍( $120 \div 9 = 13 \frac{2}{3}$ )，所以此时转子齿没有与 B 相定子的齿对应，只是第 13 号小齿靠近 B 相磁极的中心线，与中心线相差  $3^\circ$ ，如果此时突然变为 B 相通电，A、C 相不通电，则 B 相磁极迫使 13 号转子齿与之对齐，转子就转动  $3^\circ$ ，这样使电机转子一步。如果按照 A—AB—B—BC—C—CA—A 次序通电则为正转。通常用三相六拍环形脉冲分配器产生步进脉冲。

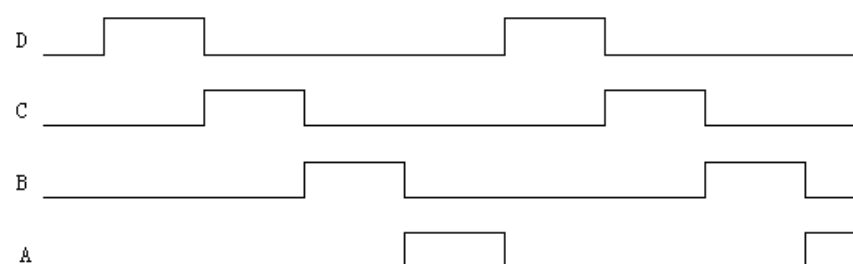
(2)运转速度的控制。若改变 ABC 三相绕组高低电平的宽度，就会导致通电和断电的变化速率变化，使电机转速改变，所以调节脉冲的周期就可以控制步进电机的运转速度。

(3)旋转的角度控制。因为输入一个 CP 脉冲使步进电机三相绕组状态变化一次，并相应地旋转一个角度，所以步进电机旋转的角度由输入的 CP 脉冲数确定。

本实验箱所使用步进电机为 4 相步进电机，最小旋转角度为  $1.8^\circ$ ，其正向转动控制时序如下所示，每一个脉冲控制其转过  $1.8^\circ$ 。



反向转动控制时序如下：



### 三、 实验内容

本实验要完成的任务就是设计步进电机的控制电路。通过一个拨动开关 K1 来控制步进电机的顺时针和逆时针旋转；通过八个按键开关 S1-S8 来控制步进电机旋转的角度。

在本实验中，用到的模块有数字信号源模块、拨动开关模块、按键开关、步进电机模块等。其中数字信号源、拨动、按键开关与 FPGA 的连接电路和管脚连接在以前的实验中都做了详细说明，这里不在赘述。步进电机模块位于实验平台的左上方，其工作状态通过一个跳



线来选择。其控制电路如下图 15-2 所示。其与 FPGA 的管脚连接如表 15-1 所示。

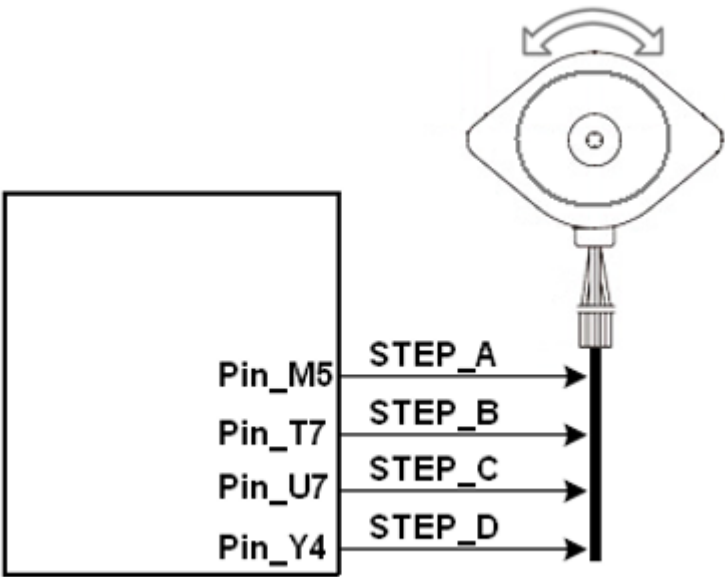


图 15-2 步进电机驱动电路图

信号名称	FPGA I/O 名称
STEP_A	N22
STEP_B	R20
STEP_C	M22
STEP_D	P22

表 15-1 步进电机模块接口与 FPGA 的管脚连接表

四、 实验步骤

- 1、 打开 VIVADO 软件，新建一个工程。
- 2、 建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。
- 3、 按照实验原理和自己的想法，在 VERILOG 编辑窗口编写 VERILOG 程序。
- 4、 编写完 VERILOG 程序后，保存起来。方法同实验一。
- 5、 将自己编辑好的的程序进行综合仿真，并对程序的错误进行修改，最终通过综合。
- 6、 综合仿真无误后，依照步进电机、按键、拨动开关与 FPGA 的管脚连接表 15-2 分配。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
CLK	数字时钟模块	Y8	接 1KHZ 时钟
KEYORDER	拨动开关 K1	V10	顺/逆时针控制
ASTEP	步进电机 A 相	N22	步进电机相位

BSTEP	步进电机 B 相	R20	控制信号
CSTEP	步进电机 C 相	M22	
DSTEP	步进电机 D 相	P22	
Key7_5	按键开关 S1	P15	旋转角度控制
Key15	按键开关 S2	N18	
Key30	按键开关 S3	P18	
Key45	按键开关 S4	R16	
Key90	按键开关 S5	T17	
Key180	按键开关 S6	M20	
Key360	按键开关 S7	K18	
Key8	按键开关 S8	K21	

表 15-2 端口管脚分配表

- 7、用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

## 五、 实验结果与现象

以设计的参考示例为例，当设计文件加载到目标器件后，数字时钟选择 1KHZ，按动按键开关 S1-S8，步进电机将会按照程序设计的相应的步进角度进行旋转。拨动拨动开关的 K1，步进电机旋转的方向将会发生改变。

## 六、 实验报告

- 1、绘出仿真波形，并作说明。
- 2、将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。

## 实验十六 PS2 接口键盘显示实验

### 一、 实验目的

- 1、 学习用 FPGA 设计简单通信协议的方法。
- 2、 学习 PS2 的工作原理，扫描码的 ASCII 码的转换。
- 3、 掌握 VERILOG 编写中的一些小技巧。

### 二、 实验原理

PS2 通信协议是一种双向同步串行通讯协议。通讯的两端通过 CLOCK(时钟信号端)同步,并通过 DATA(数据端口)交换数据。任何一方如果想要抑制另外一方的通讯时,只需要把 CLOCK 拉到低电平。

PS2 标准,规范每笔数据传输包含起始位(start bit)、扫描码(scan code)、奇同位检查(odd parity)、以及终止位(stop bit)共计 11 位,并以双向串行数据传输的方式,达到通信的目的。且当主机端(host)或从机端(slave)并无传送或接收数据时,数据传输端口及频率均将升为高电位。图 16-1 所示为每一笔数据传输所包含之内容如下:

起始位(“0”)

8 位数据宽度的扫描码( scan code )。

奇同位检查,使扫描码与奇同位加起来 1 的数字为奇数个。

终止位(“1”)

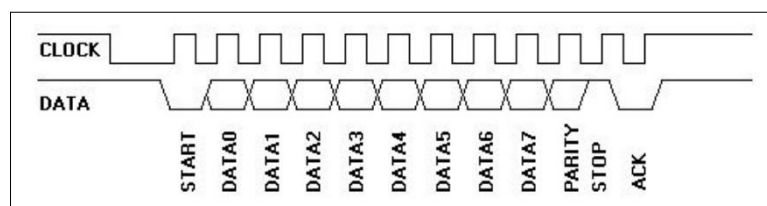


图 16-1 PS2 串行传输标准

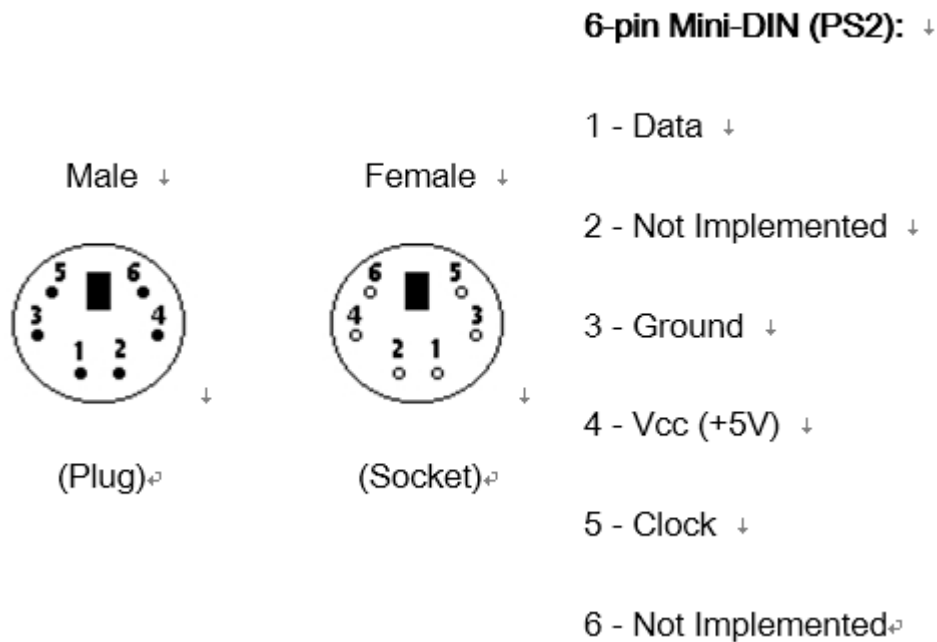


图 16-2 PS2 端口脚位定义

PS2 控制接口仅使用到两条传输端口，一为频率端口，另一则为数据端口如图 16-2 所示，且此传输埠必为三态(Tri-State)并具有双向(bidirectional)特性。PS2 传输产品上，常见为鼠标与键盘，两者的驱动原理均相同，仅扫描码(scan code)不同。因此我们以 PS2 键盘为例进行说明。

键盘其实就是一个大型的按键矩阵，它们由安装在电路板上的处理器（叫做“键盘编码器”）来监视着。虽然不同的键盘可能采用不同的处理器，但是它们完成的任务都是一样的，即监视哪些按键被按下，哪些按键被释放了，并将这些信息传送到主机。如果有必要，处理器处理所有的去抖动，并在它的 16 字节的缓冲区里缓冲数据。主机端包含了一个“键盘控制器”与键盘处理器进行通讯，并解码来自键盘处理器的信息，然后高速系统当前按键对应的处理事情。主机与键盘之间的通讯仍旧采用 IBM 的协议。

键盘处理器花费很多时间来扫描或监视按键矩阵。如果发现有按键按下、释放或长按，键盘就发送“扫描码”的信息到主机。扫描码有两种不同的类型：“通码”和“断码”。当一个键被按下去或长按的时候，键盘就发送通码；当一个键被释放的时候，键盘就发送断码。每个键盘被分配了唯一的通码和断码，这样主机通过查找唯一的扫描码就可以确定是哪个按键被按下或释放。每个键一整套的通断码组成了“扫描码集”，现在所有的键盘都采用第二套扫描码。由于没有一个简单的公式可以计算扫描码，所以要知道某个特定按键的通码和断码，只能采用查表的方法来获得。需要特别注意的是，按键的通码值表示键盘上的一个按键，

并不表示印刷在按键上的那个字符，这就意味着通码和 ASCII 码之间没有任何关联。

另外，第二套通码都只有一个字节宽，但也有少数“扩展按键”的通码是两字节或四字节宽，这类码的第一个字节总是 0xE0。与通码一样，每个按键在释放的时候，键盘就会发送一个断码。每个键也都有它自己的唯一的断码，不过庆幸的是，断码与断码之间存在着必然的联系。多数第二套断码有两个字长，它们的第一个字节是 0xF0，第二个字节就是对应按键的通码。扩展按键的断码通常有三个字节，前两个字节 0xE0 和 0xF0，最后一个字节是这个按键通码的最后一个字节。表 16-1 列出了键盘按键的通码和断码。

键值	通码	断码	键值	通码	断码	键值	通码	断码
A	1C	F0,1C	9	46	F0,46	[	54	F0,54
B	32	F0,32	`	0E	F0,0E	INSERT	67	F0,67
C	21	F0,21	-	4E	F0,4E	HOME	6E	F0,6E
D	23	F0,23	=	55	F0,55	PG UP	6F	F0,6F
E	24	F0,24	\	5C	F0,5C	DELETE	64	F0,64
F	2B	F0,2B	BKSP	66	F0,66	END	65	F0,65
G	34	F0,34	SPACE	29	F0,29	PG DN	6D	F0,6D
H	33	F0,33	TAB	0D	F0,0D	U ARROW	63	F0,63
I	43	F0,48	CAPS	14	F0,14	L ARROW	61	F0,61
J	3B	F0,3B	L SHFT	12	F0,12	D ARROW	60	F0,60
K	42	F0,42	L CTRL	11	F0,11	R ARROW	6A	F0,6A
L	4B	F0,4B	L WIN	8B	F0,8B	NUM	76	F0,76
M	3A	F0,3A	L ALT	19	F0,19	KP /	4A	F0,4A
N	31	F0,31	R SHFT	59	F0,59	KP *	7E	F0,7E
O	44	F0,44	R CTRL	58	F0,58	KP -	4E	F0,4E
P	4D	F0,4D	R WIN	8C	F0,8C	KP +	7C	F0,7C
Q	15	F0,15	R ALT	39	F0,39	KP EN	79	F0,79
R	2D	F0,2D	APPS	8D	F0,8D	KP .	71	F0,71
S	1B	F0,1B	ENTER	5A	F0,5A	KP 0	70	F0,70
T	2C	F0,2C	ESC	08	F0,08	KP 1	69	F0,69
U	3C	F0,3C	F1	07	F0,07	KP 2	72	F0,72
V	2A	F0,2A	F2	0F	F0,0F	KP 3	7A	F0,7A
W	1D	F0,1D	F3	17	F0,17	KP 4	6B	F0,6B
X	22	F0,22	F4	1F	F0,1F	KP 5	73	F0,73

Y	35	F0,35	F5	27	F0,27	KP 6	74	F0,74
Z	1A	F0,1A	F6	2F	F0,2F	KP 7	6C	F0,6C
0	45	F0,45	F7	37	F0,37	KP 8	75	F0,75
1	16	F0,16	F8	3F	F0,3F	KP 9	7D	F0,7D
2	1E	F0,1E	F9	47	F0,47	]	5B	F0,5B
3	26	F0,26	F10	4F	F0,4F	;	4C	F0,4C
4	25	F0,25	F11	56	F0,56	'	52	F0,52
5	2E	F0,2E	F12	5E	F0,5E	,	41	F0,41
6	36	F0,36	PRNT SCRN	57	F0,57	.	49	F0,49
7	3D	F0,3D	SCROLL	5F	F0,5F	/	4A	F0,4A
8	3E	F0,3E	PAUSE	62	F0,62			

表 16-1 PS2 键盘扫描码

### 三、 实验内容

本实验的任务就是利用 PS2 接口将键盘按键的通码在数码管上显示出来。实验箱中用到 PS2 键盘接口与 FPGA 的接口电路如图 16-3 所示。其与 FPGA 的管脚连接如表 16-2 所示。

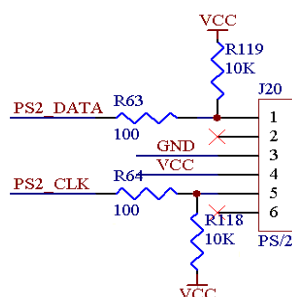


图 16-3 PS2 键盘接口电路图

信号名称	FPGA I/O 名称	功能说明
KB_DAT	V8	KeyBoard data
KB_CLK	AB7	KeyBoard clock

表 16-2 PS2 键盘接口与 FPGA 管脚连接表

### 四、 实验步骤

- 1、打开 VIVADO 软件，新建一个工程。
- 2、建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。
- 3、按照实验原理和自己的想法，在 VERILOG 编辑窗口编写 VERILOG 程序。示例程序共提供 2 个 VERILOG 源程序。每一个源程序完成一定的功能。其具体的功能如下表 16-

3 所示:

文件名称	完成功能
keyboard.V	PS2 键盘控制器电路设计。
DISPLAY.V	七段显示器译码电路设计。

表 16-3 示例程序功能表

- 4、编写完 VERILOG 程序后，保存起来。方法同实验一。
- 6、将自己编辑好的的程序进行综合仿真，并对程序的错误进行修改，最终通过综合。
- 7、综合仿真无误后，依照拨动开关、LED 与 FPGA 的管脚连接表 16-4 分配。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
CLK	数字信号源	Y8	时钟为 24MHZ
RESET	按键开关 S1	P15	复位信号
KYCOCLK	PS2 模块时钟端	V8	PS2 同步时钟
DATA	PS2 模块数据端	AB7	PS2 数据
A	数码管 A 段	P16	扫描码显示
B	数码管 B 段	P17	
C	数码管 C 段	N17	
D	数码管 D 段	N15	
E	数码管 E 段	M15	
F	数码管 F 段	L17	
G	数码管 G 段	L18	
DP	数码管 DP 段	K19	
SA	位选 DEL0	L22	
SB	位选 DEL1	P21	
SC	位选 DEL2	N20	

表 16-4 端口管脚分配表

- 9、用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

五、 实验结果与现象

以设计的参考示例为例，将 PS2 接口的键盘接入 PS2 接口内。当设计文件加载到目标器件后，将数字信号源模块的时钟选择为 24MHZ，按下 PS2 键盘上的键，则在实验平台的八位数码管上的中间两位将显示被按键的扫描码。观察其按下的键值所对应的扫描码是否与表 21-1 一一对应。按下按键开关 S1 则停止对键盘的扫描，数码管上的扫描码不会发生改变。

六、 实验报告

- 1、绘出仿真波形，并作说明。

2、将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。



## 实验十七 VGA 彩条信号发生器的设计

### 一、 实验目的

- 1、 了解普通显示器正确显示的时序。
- 2、 了解 VERILOG 产生 VGA 显示时序的方法。
- 3、 进一步加强对 FPGA 的认识。

### 二、 实验原理

尽管显示器的新品层出不穷，但 CRT（Cathode Ray Tube，阴极射线管）的基本工作原理一直沿用了几十年，直到今天也没有太大的变化。显示器是一种复杂的设备，其扩展性和可靠性也十分惊人，在这一方面，电子控制起了很大的作用，任何机械都会有磨损，唯有用电子才能延长寿命，甚至能适应数千小时的工作。电子枪是显示像管的核心，它发出的电子束击中光敏材料（荧光屏），刺激荧光粉就能产生图像。实际上，电子枪和大体积的、功率强劲的二极管没有什么区别，其原理也适用于电视机和示波器。

CRT 分为几个部分：Deflection Coil（偏转线圈）用于电子枪发射器的定位，它能够产生一个强磁场，通过改变强度来移动电子枪。线圈偏转的角度有限，当电子束传播到一个平坦的表面时，能量会轻微的偏移目标，仅有部分荧光粉被击中，四边的图像会产生弯曲现象。为了解决这个问题，显示器生产厂把显像管做成球形，让荧光粉充分地接受能量，缺点是屏幕将变得弯曲，电子束射击由左至右，由上至下的过程称为刷新，不断重复的刷新能保持图像的持续性。

显示器屏幕的色彩是由 RGB（红、绿、蓝）三色光所合成的，我们可通过调整这三个基色调出其它的颜色，在许多图像处理软件里都有提供色彩调配功能，你可输入三基色的数值来调配颜色，也可直接根据软件提供的调色板来选择颜色。在这一部分的功能上实验系统采用专用的编解码芯片来完成。其具体实现、原理我们将在以后的实验中做详细的说明。在本实验中只用到了 RGB 三基色来组成八种颜色构成彩条信号。

VGA 显示器在显示过程中主要由五个信号来控制，分别是 R、G、B、HS 和 VS。其中 R、G、B 分别用来驱动显示器三个基色的显示，即红、绿和蓝，HS 是行同步信号，VS 是场同步信号。在做本实验时，由于没有任何显示器驱动，所以显示器工作在默认状态，分辨率：640×480，刷新率：60Hz。在此状态下，当 VS 和 HS 都为低电平时，VGA 显示器显示亮的状态，其正向扫描过程约为 26us。当一行扫描结束后，行同步信号 HS 置高电平，持续

约 6us 后，变成低电平，在 HS 为高电平期间，显示器产生消隐信号，这就是显示器回扫的过程。当扫描完一场后，也就是扫描完 480 行以后，场同步信号 VS 置高电平，产生场同步，此同步信号可以使扫描线回到显示器的第一行第一列位置。显示器显示的时序图如下图 17-1 所示：

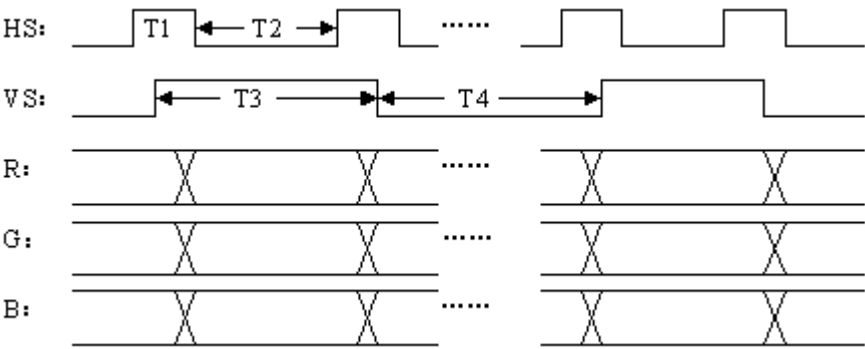


图 17-1 CRT 显示器时序

上图中 T1 为同步消隐信号，约为 6us 脉宽，T2 为行显示过程，约为 26us，T3 为行同步信号，宽度为两个行同步周期，T4 为显示时间，约为 480 行周期。

三、 实验内容

本实验要完成的任务就是通过 FPGA 在显示器上显示一些条纹或图案,要求 CRT 显示器上能够显示横条纹、竖条纹以及棋盘格子图案。实验中系统时钟选择时钟模块的 12MHz，用一个按键模块的 S1 来控制显示模式，每按下一次，屏幕上的图案改变一次，依次为横条纹、竖条纹以及棋盘格子图案。实验的输出就直接输出到 VGA 接口，通过 CRT 显示器显示出来。

实验箱中用到的数字时钟模块、按键开关与 FPGA 的接口电路，以及数字时钟源、按键开关与 FPGA 的管脚连接在以前的实验中都做了详细说明，这里不在赘述。VGA 接口在实验系统的视频输入输出模块。我们可以通模块上的一个三位的跳线来选择 VGA 的三基色信号是通过编解码芯片输出还是直接从 FPGA 输出。其电路图如下图 17-2 所示：

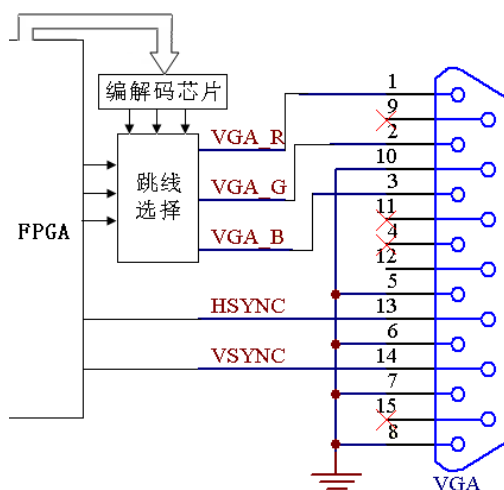


图 17-2 VGA 与 FPGA 的电路连接图

信号名称	对应 FPGA 管脚名	说明
VGA-R	V17	VGA 接口的 R 信号
VGA-G	Y16	VGA 接口的 G 信号
VGA-B	U16	VGA 接口的 B 信号
HSYNC	AA14	VGA 行同步信号
VSYNC	Y15	VGA 场同步信号

表 17-1 VGA 接口与 FPGA 直接连接后的管脚连接表

#### 四、实验步骤

- 1、打开 VIVADO 软件，新建一个工程。
- 2、建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。
- 3、按照实验原理和自己的想法，在 VERILOG 编辑窗口编写 VERILOG 程序。
- 4、编写完 VERILOG 程序后，保存起来。方法同实验一。
- 5、对自己编写的 VERILOG 程序进行综合并仿真，对程序的错误进行修改。
- 6、综合仿真无误后，依照数字信号源模块、VGA 模块、按键开关模块与 FPGA 的管脚连接表 17-2 分配。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
CLK	数字信号源	Y8	时钟为 12MHZ
KEY	按键开关 S1	P15	显示模式选择
R	VGA 模块 R 信号	V17	VGA 接口信号
G	VGA 模块 G 信号	Y16	
B	VGA 模块 B 信号	U16	
HS	VGA 模块行同步	AA14	
VS	VGA 模块场同步	Y15	

表 17-2 端口管脚分配表

7、用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

## 五、 实验结果与现象

以设计的参考示例为例，当设计文件加载到目标器件后，将显示器与实验系统视频输入输出模块的 VGA 接口连接起来，将其模块上的三位跳线全部选择 FPGA（跳至下端），数字信号源的时钟选择为 12MHZ。此时连接的 VGA 显示屏上将会出现纵向的彩条信号。按下按键开关模块的 S1 键将会改变为横彩条和方格彩条。

## 六、 实验报告

- 1、绘出仿真波形，并作说明。
- 2、将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。
- 3、试编写其它图形的 VGA 显示的程序。

## 实验十八 用 VERILOG 设计七人表决器

### 一、 实验目的

- 1、 熟悉 VERILOG 的编程。
- 2、 熟悉七人表决器的工作原理。
- 3、 进一步了解实验系统的硬件结构。

### 二、 实验原理

所谓表决器就是对于一个行为，由多个人投票，如果同意的票数过半，就认为此行为可行；否则如果否决的票数过半，则认为此行为无效。

七人表决器顾名思义就是由七个人来投票，当同意的票数大于或者等于 4 时，则认为同意；反之，当否决的票数大于或者等于 4 时，则认为不同意。实验中用 7 个拨动开关来表示七个人，当对应的拨动开关输入为 ‘1’ 时，表示此人同意；否则若拨动开关输入为 ‘0’，则表示此人反对。表决的结果用一个 LED 表示，若表决的结果为同意，则 LED 被点亮；否则，如果表决的结果为反对，则 LED 不会被点亮。同时，数码管上显示通过的票数。

### 三、 实验内容

本实验就是利用实验系统中的拨动开关模块和 LED 模块以及数码管模块来实现一个简单的七人表决器的功能。拨动开关模块中的 K1~K7 表示七个人，当拨动开关输入为 ‘1’ 时，表示对应的人投同意票，否则当拨动开关输入为 ‘0’ 时，表示对应的人投反对票；LED 模块中 LED1 表示七人表决的结果，当 LED1 点亮时，表示此行为通过表决；否则当 LED1 熄灭时，表示此行为未通过表决。同时通过的票数在数码管上显示出来。

在此实验中数码管、LED、拨动开关与 FPGA 的连接电路和管脚连接在以前的实验中都做了详细说明，这里不在赘述。

### 四、 实验步骤

- 1、 打开 VIVADO 软件，新建一个工程。
- 2、 建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。
- 3、 按照实验原理和自己的想法，在 VERILOG 编辑窗口编写 VERILOG 程序。
- 4、 编写完 VERILOG 程序后，保存起来。方法同实验一。
- 5、 对自己编写的 VERILOG 程序进行综合并仿真，对程序的错误进行修改。
- 6、 综合仿真无误后，依照拨动开关、LED、数码管与 FPGA 的管脚连接表 18-1 分配。

分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
K1	拨动开关 K1	V10	七位投票人的表决器
K2	拨动开关 K2	AA9	
K3	拨动开关 K3	W11	
K4	拨动开关 K4	Y11	
K5	拨动开关 K5	AB10	
K6	拨动开关 K6	AA11	
K7	拨动开关 K7	V12	
m_Result	LED 模块 LED1	C15	表决结果
LEDAG0	数码管模块 A 段	P16	表决通过的票数
LEDAG1	数码管模块 B 段	P17	
LEDAG2	数码管模块 C 段	N17	
LEDAG3	数码管模块 D 段	N15	
LEDAG4	数码管模块 E 段	M15	
LEDAG5	数码管模块 F 段	L17	
LEDAG6	数码管模块 G 段	L18	
LEDAG7	数码管模块 DP 段	K19	

表 18-1 端口管脚分配表

- 7、用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

## 五、 实验结果与现象

以设计的参考示例为例，当设计文件加载到目标器件后，拨动实验系统中拨动开关模块的 K0-K7 七位拨动开关，如果拨动开关的值为“1”（即拨动开关的开关置于上端，表示此人通过表决）的个数大于或等于四时 LED 模块的 LED1 被点亮，否则 LED1 不被点亮。同时数码管上显示通过表决的人数。

## 六、 实验报告

- 1、绘出仿真波形，并作说明。
- 2、将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。
- 3、试在此实验的基础上增加一个表决的时间，只的在这一时间内的表决结果有效。

## 实验十九 用 VERILOG 设计四人抢答器

### 一、 实验目的

- 1、熟悉四人抢答器的工作原理。
- 2、加深对 VERILOG 语言的理解。
- 3、掌握 EDA 开发的基本流程。

### 二、 实验原理

抢答器在各类竞赛性质的场合得到了广泛的应用，它的出现，消除了原来由于人眼的误差而未能正确判断最先抢答的人的情况。

抢答器的原理比较简单，首先必须设置一个抢答允许标志位，目的就是为了允许或者禁止抢答者按按钮；如果抢答允许位有效，那么第一个抢答者按下的按钮就将其清除，同时记录按钮的序号，也就是对应的按按钮的人，这样做的目的是为了禁止后面再有人按下按钮的情况。总的说来，抢答器的实现就是在抢答允许位有效后，第一个按下按钮的人将其清除以禁止再有按钮按下，同时记录清楚抢答允许位的按钮的序号并显示出来，这就是抢答器的实现原理。

### 三、 实验内容

本实验的任务是设计一个四人抢答器，用按键模块的 S5 来作抢答允许按钮，用 S1~S4 来表示 1 号抢答者~4 号抢答者，同时用 LED 模块的 LED1~LED4 分别表示于抢答者对应的位子。具体要求为：按下 S12 一次，允许一次抢答，这时 S1~S4 中第一个按下的按键将抢答允许位清除，同时将对应的 LED 点亮，用来表示对应的按键抢答成功。数码管显示对应抢答成功者的号码。

在此实验中数码管、LED、按键开关与 FPGA 的连接电路和管脚连接在以前的实验中都做了详细说明，这里不在赘述。

### 四、 实验步骤

- 1、 打开 VIVADO 软件，新建一个工程。
- 2、 建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。
- 3、 按照实验原理和自己的想法，在 VERILOG 编辑窗口编写 VERILOG 程序。
- 4、 编写完 VERILOG 程序后，保存起来。方法同实验一。
- 5、 对自己编写的 VERILOG 程序进行综合并仿真，对程序的错误进行修改。

6、 综合仿真无误后，依照按键开关、LED、数码管与 FPGA 的管脚连接表 19-1 分配。

分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
S1	按键开关 S1	P15	表示 1 号抢答者
S2	按键开关 S2	N18	表示 2 号抢答者
S3	按键开关 S3	P18	表示 3 号抢答者
S4	按键开关 S4	R16	表示 4 号抢答者
START	按键开关 S12	M16	开始抢答按键
DOUT0	LED 模块 LED1	C15	1 号抢答者灯
DOUT1	LED 模块 LED2	E15	2 号抢答者灯
DOUT2	LED 模块 LED3	B16	3 号抢答者灯
DOUT3	LED 模块 LED4	A16	4 号抢答者灯
LEDAG0	数码管模块 A 段	P16	抢答成功者 号码显示
LEDAG1	数码管模块 B 段	P17	
LEDAG2	数码管模块 C 段	N17	
LEDAG3	数码管模块 D 段	N15	
LEDAG4	数码管模块 E 段	M15	
LEDAG5	数码管模块 F 段	L17	
LEDAG6	数码管模块 G 段	L18	
LEDAG7	数码管模块 DP 段	K19	

表 19-1 端口管脚分配表

7、 用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

五、 实验结果与现象

以设计的参考示例为例，当设计文件加载到目标器件后，按下按键开关的 S12 按键，表示开始抢答。然后，同时按下 S1-S4，首先按下的键的键值被数码管显示出来，对应的 LED 灯被点亮。与此同时，其它按键失去抢答作用。

六、 实验报告

- 1、 绘出仿真波形，并作说明。
- 2、 将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。



## 实验二十 正负脉宽调制信号发生器设计

### 一、 实验目的

- 1、 掌握了解正负脉宽数控调制信号发生的原理。
- 2、 熟练的运用示波器观察实验箱上的探测点波形。
- 3、 掌握时序电路设计的基本思想。

### 二、 实验原理

首先详细说明一下正负脉宽数控的原理。所以正负脉宽数控就是直接直接输入脉冲信号的正脉宽数和负脉宽数，当然，正负脉宽数一旦定下来，脉冲波的周期也就确定下来了。其次是调制信号，调制信号有很多种，有频率调制、相位调制、幅度调制等等，本实验中仅对输出的波形进行最简单的数字调制，另外为了 EDA 设计的灵活性，实验中要求可以输出非调制波形、正脉冲调制和负脉冲调制。非调制波形就是原始的脉冲波形；正脉冲调制就是在脉冲波输出‘1’的期间用输出另一个频率的方波，而在脉冲波为‘0’器件还是原始波形；负脉冲调制正好与正脉冲调制相反，要求在脉冲波输出为‘0’期间输出另外一个频率的方波，而在‘1’期间则输出原始波形。为了简化实验，此处的调制波形（另外一个频率的方波）就用原始的时钟信号。其具体的波形如下图 20-1 所示：

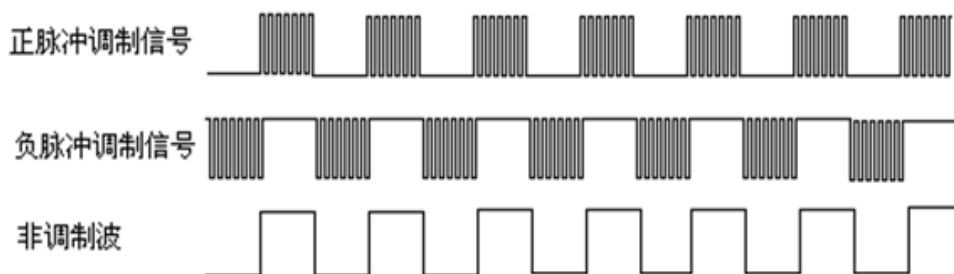


图 20-1 调制波形

### 三、 实验内容

本实验的任务是设计一个正负脉宽数控调制信发生器。要求能够输出正负脉宽数控的脉冲波、正脉冲调制的脉冲波和负脉冲调制的脉冲波形。实验中的时钟信号选择时钟模块的 1MHz 信号，用拨挡开关模块的 K1~K4 作为正脉冲脉宽的输入，用 K7~K10 作为负脉冲脉宽的输入，用按键开关模块中的 S1 做为模式选择键，每按下一次，输出的脉冲波形改变一次，依次为原始脉冲波、正脉冲调制波和负脉冲调制波形。波形输出至实验箱观测模块的探针，以便示波器观察。

在此实验中拨动开关、按键开关、输出观测端与 FPGA 的连接电路和管脚连接在以前的实验中都做了详细说明，这里不在赘述。

#### 四、实验步骤

- 1、 打开 VIVADO 软件，新建一个工程。
- 2、 建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。
- 3、 按照实验原理和自己的想法，在 VERILOG 编辑窗口编写 VERILOG 程序。
- 4、 编写完 VERILOG 程序后，保存起来。方法同实验一。
- 5、 对自己编写的 VERILOG 程序进行综合并仿真，对程序的错误进行修改。
- 6、 综合仿真无误后，依照拨动开关、按键开关、输出观测点与 FPGA 的管脚连接表 19-1 分配。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
CLK	数字信号源	Y8	时钟为 1MHZ
P0	拨动开关 K1	V10	正脉宽度输入
P1	拨动开关 K2	AA9	
P2	拨动开关 K3	W11	
P3	拨动开关 K4	Y11	
N0	拨动开关 K7	V12	负脉宽度输入
N1	拨动开关 K8	U10	
N2	拨动开关 K9	J20	
N3	拨动开关 K10	J18	
MODE	按键开关 S1	P15	输出模式选择
FOUT	观测模块输出	Y16	调制信号输出

表 19-1 端口管脚分配表

- 7、 用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

#### 五、实验结果及现象

以设计的参考示例为例，当设计文件加载到目标器件后，将数字信号源模块的时钟选择为 1MHZ，拨动八位拨动开关，使 K1-K4 中至少有一个为高电平，K5-K8 至少有一个为高电平，此时输出观测模块用示波器可以观测到一个矩形波，其高低电平的占空比为 K1-K4 高电平的个数与 K7-K10 高电平个数的比。按下 S1 按键后，矩形波发生改变。

#### 六、实验报告

- 1、 绘出仿真波形，并作说明。
- 2、 将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。

# 综合设计实验

## 实验二一 数字频率计的设计

### 一、 实验目的

- 1、 了解等精度测频的方法和原理。
- 2、 掌握如何在 FPGA 内部设计多种功能模块。
- 3、 掌握 VERILOG 在测量模块设计方面的技巧。

### 二、 实验原理

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

$$f=N/T$$

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

等精度频率测频的实现方法，可以用图 21-1 所示的框图来实现。

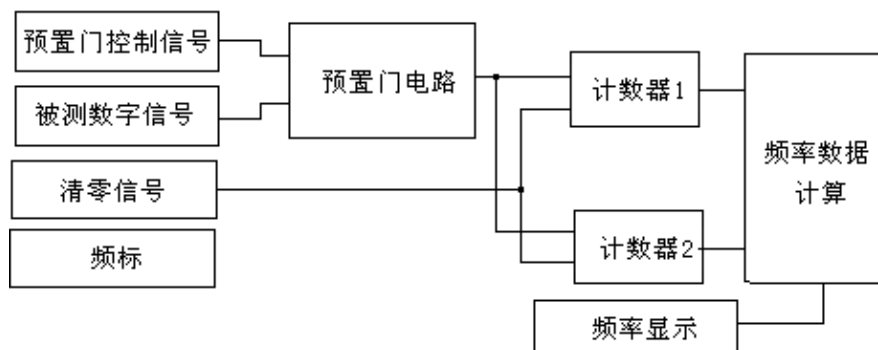


图 21-1 等精度测频实现框图

所谓等精度是指该频率计在所测量的整个频段内部，均可实现相同精度的测量，即测量精度与频率无关。图中预置门信号通常为 1s。其内部包括一个同步门电路，用来实现被测频标与被测频率的同步，提高测量精度，减少基本误差。该部分与清零脉冲协调工作用来控

制两个计数器的启动脉冲。计数器 1 和计数器 2 分别用来给频标和被测数字脉冲计数，设在同步门控制结束时计数器 1 计数 N1，计数器 2 计数 N2，假设频标频率为 F1，被测频率位 Fx，则可写出公式：

$$F_x/N_2=F_1/N_1; \cdots\cdots\cdots (1)$$

$$F_x=(F_1/N_1)*N_2\cdots\cdots\cdots (2)$$

由公式可以看出，测量精度与预置门时间无关，主要由 F1 的频率稳定度来确定，所以为了提高测量精度，主要是提高频标的频率稳定度，换句话说，测量精度基本上近似于频标的稳定度，若频标的稳定度位  $10^{-6}$ ，则测量误差边可达到  $10^{-6}$ 。在该电路中，为了确保频标计数与被测频率完全同步（即被测频率的上升沿开始计数，1s 以后，被测频率的下跳沿停止计数），同步门必须由被测信号来控制，设计方法多种多样，可由学生独立完成。

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

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

在设计频率计的时候，八个七段码管最多可以显示 99,999,999Hz，因此在设计时候用八个 4 位二进制码（BCD 码）来表示，另外还必须同样的八个 4 位二进制码来对输入的频率进行计数，在闸门下降沿的时候，将后者的值锁存到寄存器中。其信号的时序关系如下图 21-2 所示：

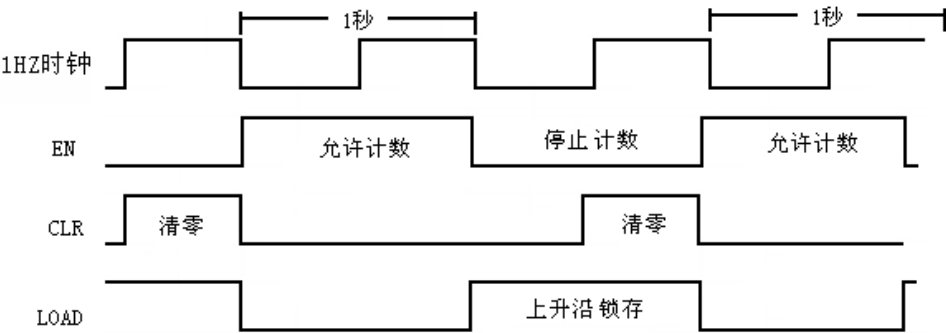


图 21-2 控制信号时序关系

三、 实验内容

本实验要完成的任务就是设计一个频率计，系统时钟选择底板上的 24M 的时钟，闸门时间为 1s(通过对系统时钟进行分频得到)，在闸门为高电平期间，对输入的频率进行计数，当闸门变低的时候，记录当前的频率值，并将频率计数器清零，频率的显示每过 2 秒刷新一次。被测频率通过一个拨动开关来选择是使用系统中的数字时钟源模块的时钟信号还是从外部通过系统的输入输出模块的输入端输入一个数字信号进行频率测量。当拨动开关为高电平时，测量从外部输入的数字信号，否则测量系统数字时钟信号模块的数字信号。其实现框图如下图 21-3 所示：

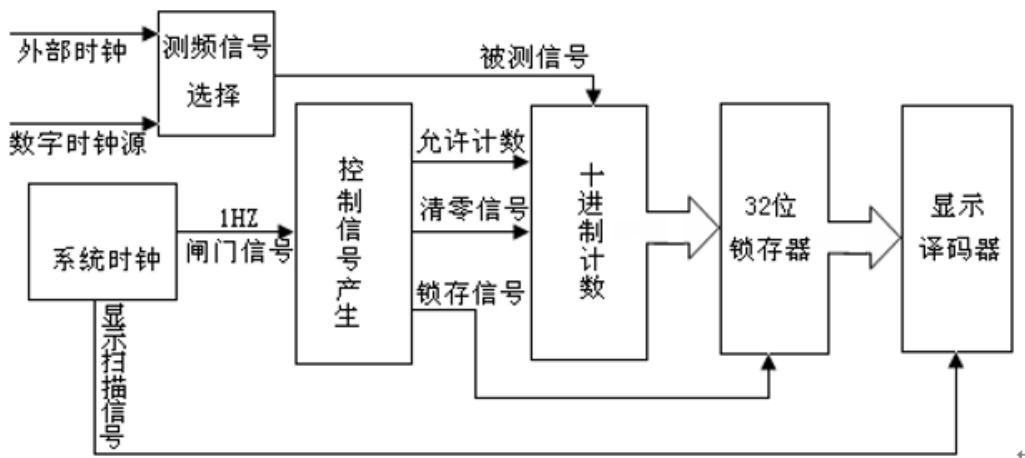


图 21-3 测频实现框图

在本实验中，用到的模块有数字信号源模块、拨动开关模块、24M 系统时钟源模块、数码管显示模块等。其中数码管、数字信号源、拨动开关与 FPGA 的连接电路和管脚连接在以前的实验中都做了详细说明，这里不在赘述。详细说明请参阅用户使用手册。其与 FPGA 的管脚连接如表 21-4 所示。

信号名称	对应 FPGA 管脚名	说明
系统时钟源	Y8	24MHZ 系统时钟

表 21-4 24M 系统时钟与 FPGA 的管脚连接表

四、 实验步骤

- 1、 打开 VIVADO 软件，新建一个工程。
- 2、 建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。
- 3、 按照实验原理和自己的想法，在 VERILOG 编辑窗口编写 VERILOG 程序。示例程序共提供 6 个 VERILOG 源程序。每一个源程序完成一定的功能。其具体的功能如下表 21-2:

文件名称	完成功能
CLKOUT.V	产生 1HZ 的闸门信号和 1KHZ 的显示扫描信号
MUX.V	被测信号源选择模块
TELTCL.V	在时钟的作用下生成测频的控制信号。
CNT10.V	十进制计数器。在实验中使用 8 个来进行计数
SEG32B.V	32 位的锁存器，在锁存控制信号的作用下，将计数的值锁存
DISPLAY.V	显示译码，将锁存的数据显示出来。

表 21-2 示例程序功能表

- 4、编写完 VERILOG 程序后，保存起来。方法同实验一。
- 5、新建一个顶层设计文件，将其他设计文件例化到顶层。
- 6、将自己编辑好的程序进行综合仿真，并对程序的错误进行修改，最终通过综合。
- 7、综合仿真无误后，依照直流电机、霍尔器件、数码管与 FPGA 的管脚连接表 21-2 分配。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
CLK50M	24M 系统时钟	Y8	底板可调时钟
CLKIN1	输入输出观测模块	V17	外部被测时钟输入
CLKIN2	数字信号源模块	Y8	内部被测时钟输入
KEY	拨动开关 K1	V10	外部/内部被测时钟选择
LEDAG0	数码管 A 段	P16	被测信号频率显示
LEDAG1	数码管 B 段	P17	
LEDAG2	数码管 C 段	N17	
LEDAG3	数码管 D 段	N15	
LEDAG4	数码管 E 段	M15	
LEDAG5	数码管 F 段	L17	
LEDAG6	数码管 G 段	L18	
LEDAG7	数码管 DP 段	K19	
SEL0	位选 DEL0	L22	
SEL1	位选 DEL1	P21	
SEL2	位选 DEL2	N20	

表 21-2 端口管脚分配表

- 8、用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

## 五、实验结果与现象

以设计的参考示例为例，当设计文件加载到目标器件后，拨动拨动开关的 K1，使其置为高电平，从输入输出观测模块的输入端输入一个频率大于 1HZ 的时钟信号，这时在数码管上显示这个时钟信号的频率值。如果使拨动开关置为低电平，数码管上显示的值 of 系统上的数字信号源的时钟的频率值。改变数字信号源的时钟，看显示的值是否与标值一致。

## 六、 实验报告

- 1、 绘出仿真波形，并作说明。
- 2、 根据前面介绍的等精度频率计的实现方法，写出等精度频率计的 VERILOG 代码。
- 3、 将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。

## 实验二二 多功能数字钟的设计

### 一、 实验目的

- 1、 了解数字钟的工作原理。
- 2、 进一步熟悉用 VERILOG 语言编写驱动七段码管显示的代码。
- 3、 掌握 VERILOG 编写中的一些小技巧。

### 二、 实验原理

多功能数字钟应该具有的功能有：显示时一分一秒、整点报时、小时和分钟可调等基本功能。首先要知道钟表的工作机理，整个钟表的工作应该是在 1Hz 信号的作用下进行，这样每来一个时钟信号，秒增加 1 秒，当秒从 59 秒跳转到 00 秒时，分钟增加 1 分，同时当分钟从 59 分跳转到 00 分时，小时增加 1 小时，但是需要注意的是，小时的范围是从 0~23 时。

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

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

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

### 三、 实验内容

本实验的任务就是设计一个多功能数字钟，要求显示格式为 小时—分钟—秒钟，整点报时，报时时间为 5 秒，即从整点前 5 秒钟开始进行报时提示，LED 开始闪烁，过整点后，停止闪烁。系统时钟选择时钟模块的 10KHz，要得到 1Hz 时钟信号，必须对系统时钟进行 10,000 次分频。调整时间的的按键用按键模块的 S1 和 S2，S1 调节小时，每按下一次，小时增加一个小时，S2 调整分钟，每按下一次，分钟增加一分钟。另外用 S12 按键作为系统时钟复位，复位后全部显示 00—00—00。

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



不在赘述。

四、实验步骤

- 1、 打开 VIVADO 软件，新建一个工程。
- 2、 建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。
- 3、 按照实验原理和自己的想法，在 VERILOG 编辑窗口编写 VERILOG 程序。
- 4、 编写完 VERILOG 程序后，保存起来。方法同实验一。
- 5、 对自己编写的 VERILOG 程序进行综合并仿真，对程序的错误进行修改，直到完全通过综合和仿真。
- 6、 综合仿真无误后，依照按键开关、数码管、LED 灯与 FPGA 的管脚连接表 22-1 分配。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
CLK	数字信号源	Y8	时钟为 10KHZ
S1	按键开关 S1	P15	调整小时
S2	按键开关 S2	N18	调整分钟
RESET	按键开关 S12	M16	复位
LED0	LED 灯模块 LED1	C15	整点倒计时
LED1	LED 灯模块 LED2	E15	
LED2	LED 灯模块 LED3	B16	
LED3	LED 灯模块 LED4	A16	
DISPLAY0	数码管 A 段	P16	时间显示
DISPLAY 1	数码管 B 段	P17	
DISPLAY 2	数码管 C 段	N17	
DISPLAY 3	数码管 D 段	N15	
DISPLAY 4	数码管 E 段	M15	
DISPLAY 5	数码管 F 段	L17	
DISPLAY 6	数码管 G 段	L18	
DISPLAY7	数码管 DP 段	K19	
SEG-SEL0	位选 DEL0	L22	
SEG-SEL1	位选 DEL1	P21	
SEG-SEL2	位选 DEL2	N20	

表 22-1 端口管脚分配表

- 7、 用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

五、实验结果与现象

以设计的参考示例为例，当设计文件加载到目标器件后，将数字信号源模块的时钟选择为 1MHZ，数码管开始显示时间，从 00-00-00 开始。在整点的前 5 秒的时候，LED 灯模块

的 LED1-LED5 开始闪烁。一旦超过整点，LED 停止显示。按动按键开关的 S1、S2 小时和分钟开始步进，进行时间的调整。按下按键开关的 S12，显示恢复到 00-00-00 重新开始显示时间。

## 六、 实验报告

- 1、 绘出仿真波形，并作说明。
- 2、 将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。
- 3、 在此实验的基础上试用其它的方法来实现数字钟的功能，并增加其它功能。

## 实验二三 数字秒表的设计

### 一、 实验目的

- 1、了解数字秒表的工作原理。
- 2、进一步熟悉用 VERILOG 语言编写驱动七段码管显示的代码。
- 3、掌握 VERILOG 编写中的一些小技巧。

### 二、 实验原理

秒表由于其计时精确，分辨率高（0.01 秒），在各种竞技场所得到了广泛的应用。秒表的工作原理与实验十五的多功能时钟基本相同，唯一不同的是，由于秒表的计时时钟信号，由于其分辨率为 0.01 秒，所以整个秒表的工作时钟是在 100Hz 的时钟信号下完成。当秒表的计时小于 1 个小时时，显示的格式是 mm-ss-xx（mm 表示分钟：0~59；ss 表示秒：0~59；xx 表示百分之一秒：0~99），当秒表的计时大于或等于一个小时时，显示的和多功能时钟是一样的，就是 hh-mm-ss（hh 表示小时：0~99），由于秒表的功能和钟表有所不同，所以秒表的 hh 表示的范围不是 0~23，而是 0~99，这也是和多功能时钟不一样的地方。

在设计秒表的时候，时钟的选择为 100Hz。变量的选择：因为 xx（0.01 秒）和 hh（小时）表示的范围都是 0~99，所以用两个 4 为二进制码（BCD 码）表示；而 ss（秒钟）和 mm（分钟）表示的范围是 0~59，所以用一个 3 位的二进制码和一个 4 位的二进制码（BCD）码表示。显示的时候要注意的问题就是小时的判断，如果小时是 00，则显示格式为 mm-ss-xx，如果小时不为 00，则显示 hh-mm-ss。

### 三、 实验内容

本实验的任务就是设计一个秒表，系统时钟选择时钟模块的 1KHz，由于计时时钟信号为 100Hz，因此需要对系统时钟进行 10 分频才能得到，之所以选择 1KHz 的时钟是因为七段码管需要扫描显示，所以选择 1KHz。另外为了控制方便，需要一个复位按键、启动计时按键和停止计时按键，分别选用实验箱按键模块的 S1、S2 和 S3，按下 S1，系统复位，所有寄存器全部清零；按下 S2，秒表启动计时；按下 S3，秒表停止计时，并且七段码管显示当前计时时间，如果再次按下 S2，秒表继续计时，除非按下 S1，系统才能复位，显示全部为 00—00—00。

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

#### 四、 实验步骤

- 1、打开 VIVADO 软件，新建一个工程。
- 2、建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。
- 3、按照实验原理和自己的想法，在 VERILOG 编辑窗口编写 VERILOG 程序。
- 4、编写完 VERILOG 程序后，保存起来。方法同实验一。
- 5、对自己编写的 VERILOG 程序进行综合并仿真，对程序的错误进行修改。
- 6、综合仿真无误后，依照拨动开关、LED 与 FPGA 的管脚连接表 23-1 分配。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
CLK	数字信号源	Y8	时钟为 1KHZ
START	按键开关 S1	P15	秒表开始计数
OVER	按键开关 S2	N18	秒表停止计数
RESET	按键开关 S3	P18	复位信号
LEDAG0	数码管 A 段	P16	秒表计数结果 输出
LEDAG1	数码管 B 段	P17	
LEDAG2	数码管 C 段	N17	
LEDAG3	数码管 D 段	N15	
LEDAG4	数码管 E 段	M15	
LEDAG5	数码管 F 段	L17	
LEDAG6	数码管 G 段	L18	
LEDAG7	数码管 DP 段	K19	
SEL0	位选 DEL0	L22	
SEL1	位选 DEL1	P21	
SEL2	位选 DEL2	N20	

表 23-1 端口管脚分配表

- 7、用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

#### 五、 实验结果与现象

以设计的参考示例为例，当设计文件加载到目标器件后，将数字信号源模块的时钟选择为 1KHZ，设计的数字秒表从 00-00-00 开始计秒。，直到按下停止按键（按键开关 S2）。数码管停止计秒。按下开始按键（按键开关 S1），数码管继续进行计秒。按下复位按键（按键开关 S3）秒表从 00-00-00 重新开始计秒。

#### 六、 实验报告

- 1、绘出仿真波形，并作说明。
- 2、将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。

## 实验二四 出租车计费器的设计

### 一、 实验目的

- 1、了解出租车计费器的工作原理。
- 2、学会用 VERILOG 语言编写正确的七段码管显示程序。
- 3、数量掌握用 VERILOG 编写复杂功能模块。
- 4、进一步数量状态积在系统设计中的应用。

### 二、 实验原理

出租车计费器一般都是按公里计费，通常是起步价 xx 元（xx 元可以行走 x 公里），然后再是 xx 元/公里。所以要完成一个出租车计费器，就要有两个计数单位，一个用来计公里，另外一个用来计费用。通常在出租车的轮子上都有传感器，用来记录车轮转动的圈数，而车轮子的周长是固定的，所以知道了圈数自然也就知道了里程。在这个实验中，就要模拟出租车计费器的工作过程，用直流电机模拟出租车轮子，通过传感器，可以得到电机每转一周输出一个脉冲波形。结果的显示用 8 个七段码管，前四个显示里程，后四个显示费用。

在设计 VERILOG 程序时，首先在复位信号的作用下将所有用到的寄存器进行清零，然后开始设定到起步价记录状态，在此状态时，在起步价规定的里程里都一直显示起步价，直到路程超过起步价规定的里程时，系统转移到每公里计费状态，此时每增加一公里，计费器增加相应的费用。

另外讲一讲编写过程中的的一些小技巧。为了便于显示，在编写过程中的数据用 BCD 码来显示，这样就不存在数据格式转换的问题。比如表示一个三位数，那么就分别用四位二进制码来表示，当个位数字累加大于 9 时，将其清零，同时十位数字加 1，依此类推。

### 三、 实验内容

本实验要完成的任务就是设计一个简单的出租车计费器，要求是起步价 3 元，准行 1 公里，以后 1 元/公里。显示部分的七段码管扫描时钟选择时钟模块的 1KHz，电机模块的跳线选择 GND 端，这样通过旋钮电机模块的电位器，即可达到控制电机转速的目的。另外用按键模块的 S1 来作为整个系统的复位按钮，每复位一次，计费器从头开始计费。直流电机用来模拟出租车的车轮子，没转动一圈认为是行走 1 米，所以每旋转 1000 圈，认为车子前进 1 公里。系统设计是需要检测电机的转动情况，每转一周，计米计数器增加 1。七段码管显示要求为前 4 个显示里程，后 3 个显示费用。

实验箱中用到的数字时钟模块、按键开关、直流电机模块、数码管与 FPGA 的接口电路，以及数字时钟源、按键开关、LED、数码管与 FPGA 的管脚连接在以前的实验中都做了详细说明，这里不在赘述。

#### 四、 实验步骤

- 1、打开 VIVADO 软件，新建一个工程。
- 2、建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。
- 3、按照实验原理和自己的想法，在 VERILOG 编辑窗口编写 VERILOG 程序。
- 4、编写完 VERILOG 程序后，保存起来。方法同实验一。
- 5、对自己编写的 VERILOG 程序进行综合并仿真，对程序的错误进行修改。
- 6、综合仿真无误后，依照拨动开关、LED 与 FPGA 的管脚连接表 24-1 分配。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。

端口名	使用模块信号	对应 FPGA 管脚	说 明
CLK	数字信号源	Y8	时钟为 1KHZ
MOTOR	直流电机模块	V5	44E 脉冲输出
RST	按键开关 S1	P15	复位信号
DISPLAY0	数码管 A 段	P16	计价器费用显示
DISPLAY 1	数码管 B 段	P17	
DISPLAY 2	数码管 C 段	N17	
DISPLAY 3	数码管 D 段	N15	
DISPLAY 4	数码管 E 段	M15	
DISPLAY 5	数码管 F 段	L17	
DISPLAY 6	数码管 G 段	L18	
DISPLAY 7	数码管 DP 段	K19	
SEG-SEL0	位选 DEL0	L22	
SEG-SEL1	位选 DEL1	P21	
SEG-SEL2	位选 DEL2	N20	

表 24-1 端口管脚分配表

- 7、用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

#### 五、 实验结果与现象

以设计的参考示例为例，当设计文件加载到目标器件后，将数字信号源模块的时钟选择为 1KHZ，将直流电机模块选择为“ON”，旋动直流电机边上的速度调节旋钮，使直流电机开始旋转，观察数码管显示的值与设计的是否一致。

#### 六、 实验报告

1、绘出仿真波形，并作说明。

2、将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。

## 实验二五 基于 VERILOG 的数码锁的设计

### 一、实验目的

- 1、了解数码锁的工作原理。
- 2、了解数码锁的实现方法。
- 3、进一步掌握 4×4 键盘的扫描的实现过程

### 二、实验原理

数码锁又称密码锁，它只需要主人记住自己的开锁密码，开门时只需要将密码输入，就可以开门，所以密码锁的核心问题就是密码的比对问题。

假如密码锁有六位，那么在系统复位后，用户按键 6 次，输入一个完整的密码串，输入完 6 次后，系统进行比对，如果发现密码吻合，则开门，否则要求用户继续输入，如果连续 3 次输入的密码串都是错误的，则系统报警，直到输入正确的密码，报警声停止。

实验中还要用到 4×4 键盘，用于用户输入密码。系统需要完成 4×4 键盘的扫描，确定有键按下后需要获取其键值，并对其进行编码，从而进行按键的识别，并将相应的按键值进行显示。键盘扫描的实现过程如下：对于 4×4 键盘，通常连接为 4 行、4 列，因此要识别按键，只需要知道是哪一行和哪一列即可，为了完成这一识别过程，首先输出 4 列中的第一列为低电平，其它列为高电平，然后读取行值；然后再输出 4 列中的第二列为低电平，读取行值，依此类推，不断循环。系统在读取行值的时候会自动判断，如果读进来的行值全部为高电平，则说明没有按键按下，否则如果读进来的行值发现不全为高电平，则说明键盘整列中必定有至少一个按键按下，读取此时的行值和当前的列值，即可判断到当前的按键位置。获取到行值和列值以后，组合成一个 8 位的数据，根据实现不同的编码在对每个按键进行匹配，找到键值后在 7 段码管显示。

### 三、实验内容

本实验需要完成的任务就是一个密码锁，考虑到系统中有键盘扫描、七段码管显示和报警，系统时钟选择时钟模块的 10KHz 时钟。键盘扫描和显示均是用 1KHz（对系统时钟进行 10 分频），输入密码时，七段码管从右至左显示按键对应的数值，每按键(0~9)一次，显示左移一次，6 次密码输入结束后系统开始校验，校验结束后，七段码管全灭。也就是显示部分维持的时间就是按键 6 次的时间和校验的时间。密码输入连续三次错误开始报警。实验中要求用 LED 模块的 LED1 指示键盘状态，如果有按键按下，LED1 亮起，直到松开该按键；



用 LED2 指示门的状态，也就是密码校验结果，如果密码校验正确，LED2 亮起，否则如果密码校验错误 LED2 闪烁 4 次，然后熄灭，表明密码错误。系统的复位用核心板上的复位按钮 RESET，复位时，七段码管全部熄灭。

实验箱中用到的数字时钟模块、按键开关、LED、数码管、键盘阵列与 FPGA 的接口电路，以及数字时钟源、按键开关、喇叭接口、LED、数码管、键盘阵列与 FPGA 的管脚连接在以前的实验中都做了详细说明，这里不在赘述。核心板上的复位按钮 RESET 请参照用户使用手册上的说明。

#### 四、实验步骤

- 1、打开 VIVADO 软件，新建一个工程。
- 2、建完工程之后，再新建一个 VERILOG File，按照需求定义输入输出管脚。
- 3、按照实验原理和自己的想法，在 VERILOG 编辑窗口编写 VERILOG 程序。
- 4、编写完 VERILOG 程序后，保存起来。方法同实验一。
- 5、对自己编写的 VERILOG 程序进行综合并仿真，对程序的错误进行修改，直到完全通过综合和仿真。
- 6、综合仿真无误后，依照所用各模块与 FPGA 的管脚参照附录进行管脚分配。分配完成后，再进行综合-执行-生成 Bit 文件一次，以使管脚分配生效。
- 7、用下载电缆通过 JTAG 口将对应的 Bit 文件加载到 FPGA 中。观察实验结果是否与自己的编程思想一致。

#### 五、实验结果与现象

以设计的参考示例为例，当设计文件加载到目标器件后，将数字信号源模块的时钟选择为 10KHZ，按下矩阵键盘的数字键，在数码管上将依次显示按下的键值。每按下一个键，LED 灯 LED1 就闪一次。如果输入的数据与程序设定的数据相同，则 LED 灯 D12 被点亮。如果输入错误，则 LED 灯 D12 闪烁。连续输入错误 3 次 12 个 LED 开始闪烁报警。

#### 六、实验报告

- 1、绘出仿真波形，并作说明。
- 2、将实验原理、设计过程、综合仿真波形和分析结果、硬件测试结果记录下来。
- 3、在此实验的基础上试用其它的方法来实现数字钟的功能，并增加其它功能。

# 附录 1 管脚表格

管脚描述	FPGA 管脚
LED	
LED1	C15
LED2	E15
LED3	B16
LED4	A16
LED5	G15
LED6	F16
LED7	C18
LED8	D16
LED9	G17
LED10	B19
LED11	A18
LED12	D18
SW_KEY	
SW_K1	V10
SW_K2	AA9
SW_K3	W11
SW_K4	Y11
SW_K5	AB10
SW_K6	AA11
SW_K7	V12
SW_K8	U10
SW_K9	J20
SW_K10	J18
SW_K11	K16
SW_K12	J15
CLK	
Clk	Y8
33.333M	F7
按键	
S1	P15
S2	N18
S3	P18
S4	R16
S5	T17
S6	M20
S7	K18

S8	K21
S9	K20
S10	L19
S11	M17
S12	M16
数码管	
7SEG_A	P16
7SEG_B	P17
7SEG_C	N17
7SEG_D	N15
7SEG_E	M15
7SEG_F	L17
7SEG_G	L18
7SEG_DP	K19
7SEL_0	L22
7SEL_1	P21
7SEL_2	N20
交通灯	
R1	AA4
R2	U12
Y1	W5
Y2	AA12
G1	V4
G2	Y13
矩阵键盘	
KB_C0	D17
KB_C1	C17
KB_C2	E16
KB_C3	H17
KB_R0	G21
KB_R1	G16
KB_R2	A17
KB_R3	B17
点阵	
DOT_R[0]	AB14
DOT_R[1]	W16
DOT_R[2]	U17
DOT_R[3]	U15
DOT_R[4]	W15

DOT_R[5]	Y14
DOT_R[6]	V14
DOT_R[7]	V13
DOT_R[8]	T16
DOT_R[9]	M19
DOT_R[10]	N19
DOT_R[11]	P20
DOT_R[12]	L21
DOT_R[13]	M21
DOT_R[14]	W17
DOT_R[15]	AA16
DOT_C0	AB15
DOT_C1	AB16
DOT_C2	W18
DOT_C3	AB17
DC MOTOR	
MT_PWM	R21
MT_SPEED	V5
MOTOR	
STEP_A	N22
STEP_B	R20
STEP_C	M22
STEP_D	P22
PS2	
KB_DATA	V8
KB_CLK	AB7
VGA	
VGA-R	V17
VGA-G	Y16
VGA-B	U16
HSYNC	AA14
VSYNC	Y15
观测点	
Input-J3	V17
Output-J4	Y16
ADC	
ADC_D0	R18
ADC_D1	Y20

ADC_D2	U14
ADC_D3	AA21
ADC_D4	T21
ADC_D5	AA22
ADC_D6	T22
ADC_D7	V18
ADC_D8	V22
ADC_D9	U20
ADC_CLK	R19
ADC_CH	W20
DAC	
DAC_D0	AB21
DAC_D1	AB22
DAC_D2	V19
DAC_D3	V20
DAC_D4	W21
DAC_D5	W22
DAC_D6	U22
DAC_D7	U21
DAC_D8	U19
DAC_D9	T18
DAC_CLK	T19
DAC_CH	AB20
DAC_WR	Y21
adc	
SADC_CLK	Y18
SADC_CS	Y19
SADC_DIN	AB19
SDAC_CLK	AA17
SDAC_CS	AA19
SDAC_DOUT	AA18