

**LS-CPU-EXB-001**

**体系结构与CPU设计教学实验系统**

**数字逻辑实验指导手册**

v 2.0

**目 录**

[实验一 基本门电路与FPGA环境熟悉 1](#_Toc444270885)

[实验二 组合逻辑电路实验 11](#_Toc444270886)

[实验三 锁存器、时钟与触发器电路实验 16](#_Toc444270887)

[实验四 时序逻辑电路实验 24](#_Toc444270888)

[实验五 存储器实验 31](#_Toc444270889)

[实验六 综合实验（数字时钟） 36](#_Toc444270890)

## 实验一 基本门电路与FPGA环境熟悉

1. **实验目的**

掌握FPGA编程入门知识、利用门级方法实现简单逻辑电路。

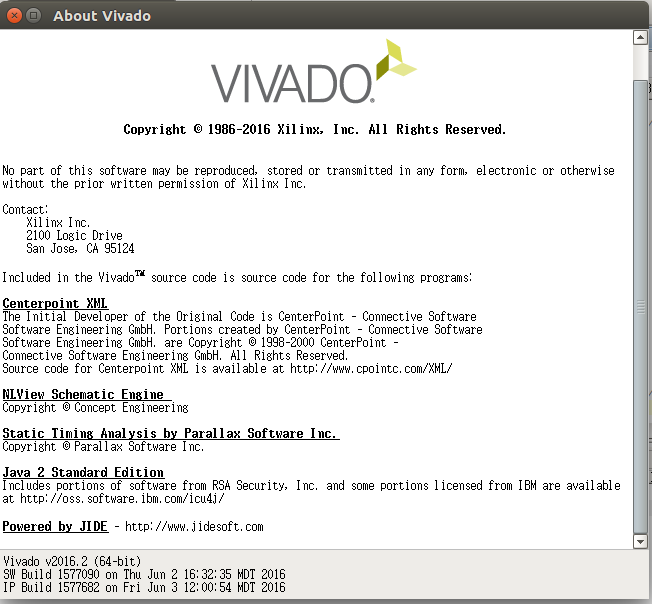
1. **实验内容**

（1）FPGA编程使用入门，掌握基本流程

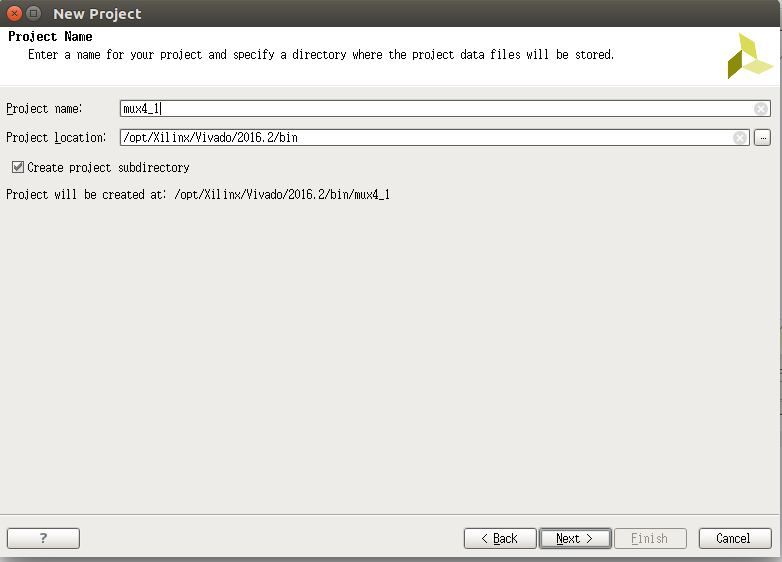
（2）Verilog基本逻辑电路实现选择器（4-1）、3-8译码器和8-3编码器（其中选择器采用assign表达式和基本的与或非等门逻辑）。

1. **实验步骤**
   * 1. **以4选1数据选择器为例，讲解FPGA编程环境的基本使用方法。**

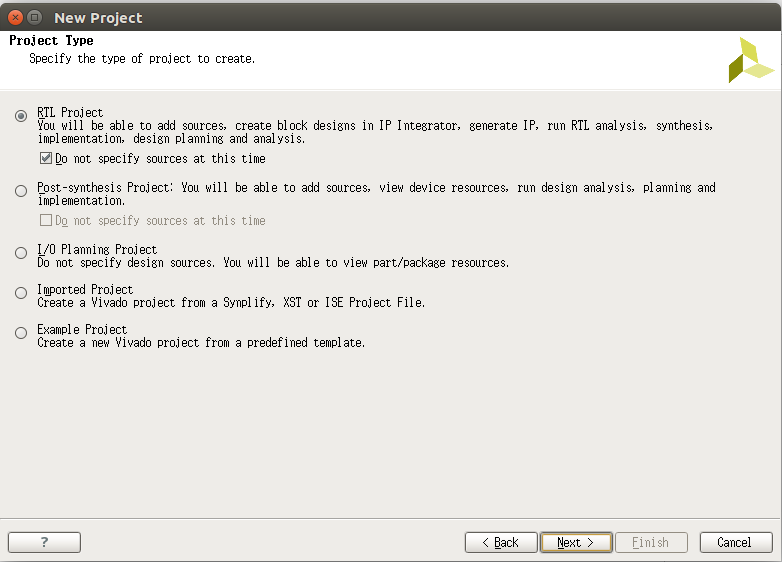
⑴ 开发环境采用Xilinx公司的Vivado，软件版本信息如下图所示：



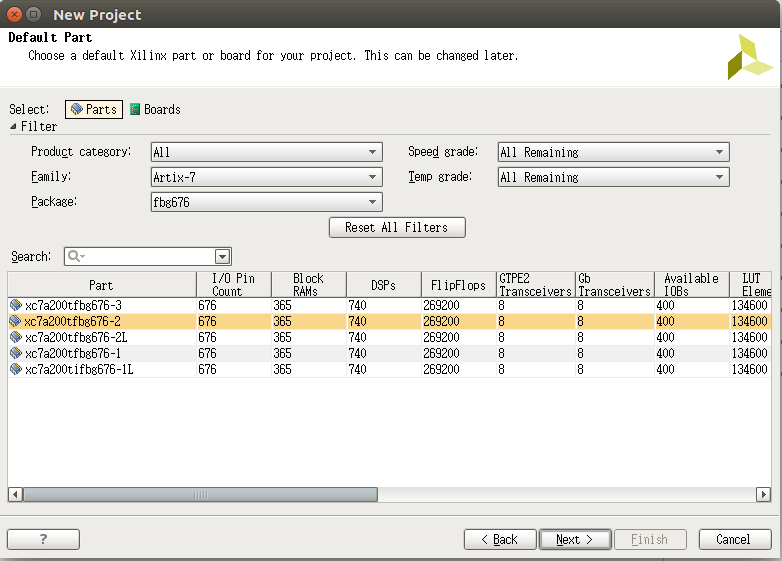
⑵ 启动Vivado软件，在菜单栏点击“File”->”New Project”，出现新建工程向导，选择“Next”，输入工程名称，选择工程的文件位置，然后选择“Next”：



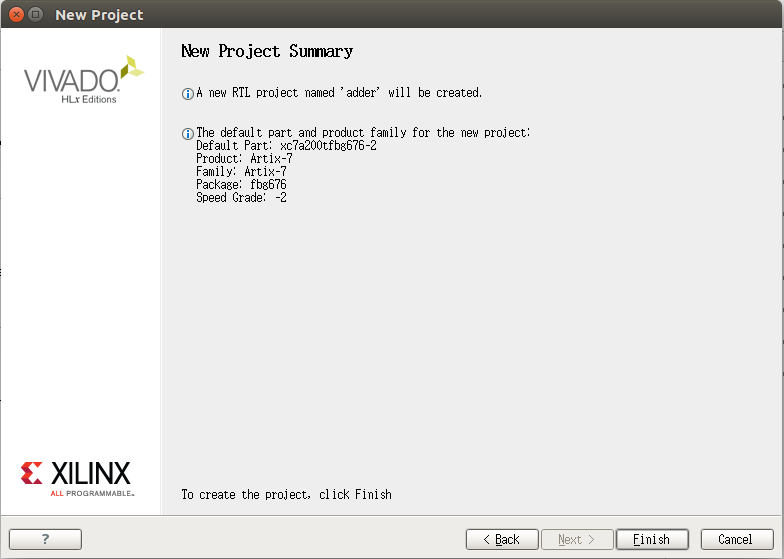
⑶ 输入工程名，选择”RTL Project”，勾选”Do not specify sources at this time”，点击”Next”：



⑷ 在筛选器的”family”选择”Artix 7”，”package”选择”fbg676”，在筛选得到的型号里面选择”xc7a200tfbg676-2”：

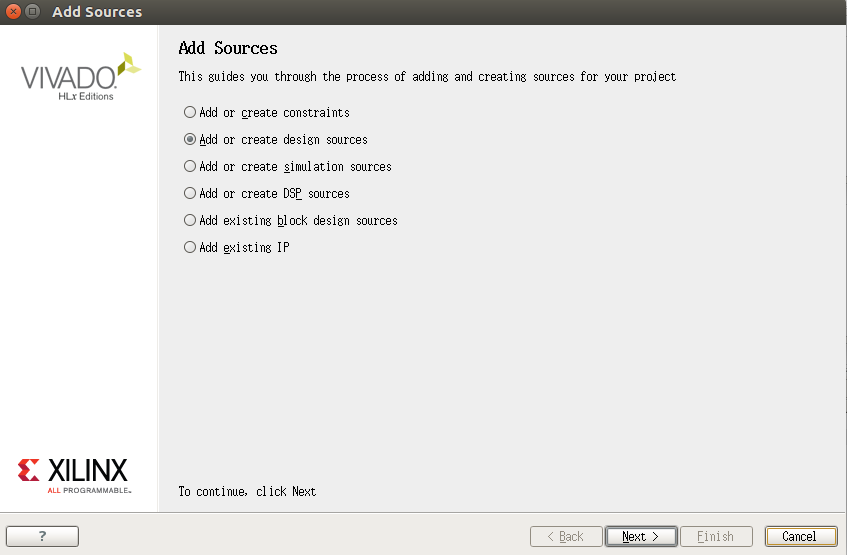


⑸ 选择“Next”，出现下图界面，点击“Finish”。

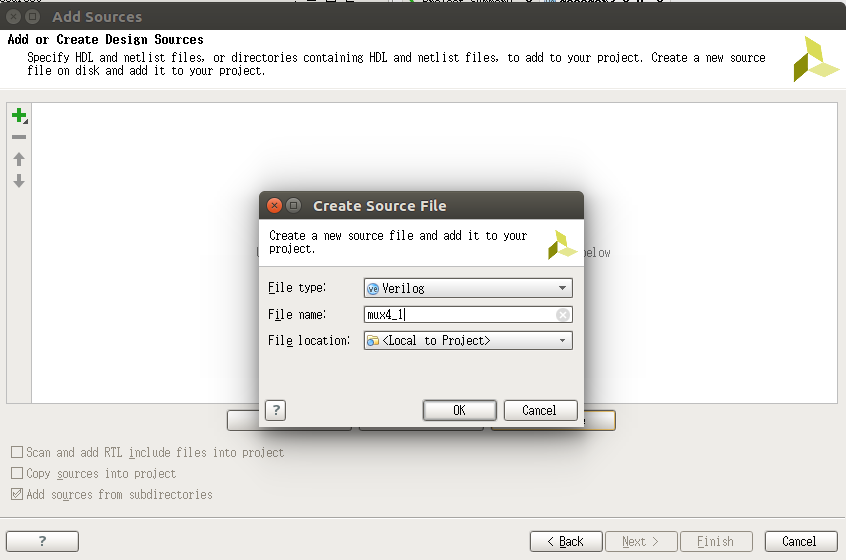


⑹ verilog代码都是以“.v”为后缀名的文件，可以在其他文件编辑器里写好，再添加到新建的工程中，也可以在工程中新建一个再编辑。

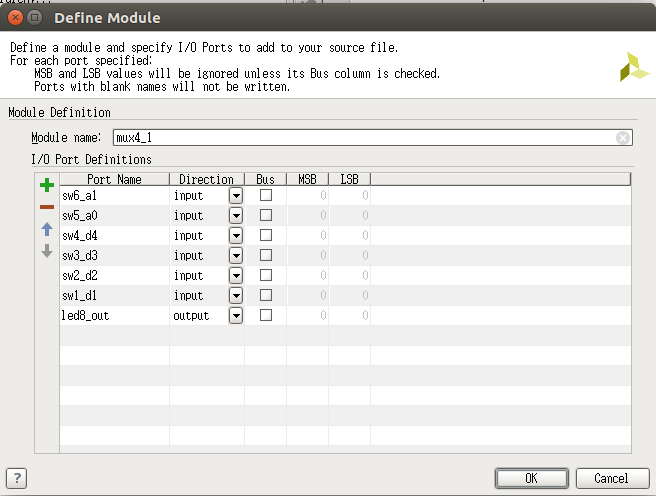
添加已有verilog文件的方法如下：在”Project Manager”下点击”Add sources”，选择”Add or create design sources”：



⑺ 点击”Next”，在出现的界面中点击”Create File”，输入文件名“mux4\_1”，点击“OK”，点击”Finish”：



⑻ 定义模块的输入输出。本例中使用开发板上U9单元的“拨码开关”作为输入，sw6\_a1和sw5\_a0为数据选择位，sw1\_d1、sw2\_d2、sw3\_d3、sw4\_d4为数据位。Led8\_out为数据输出。点击“OK”。

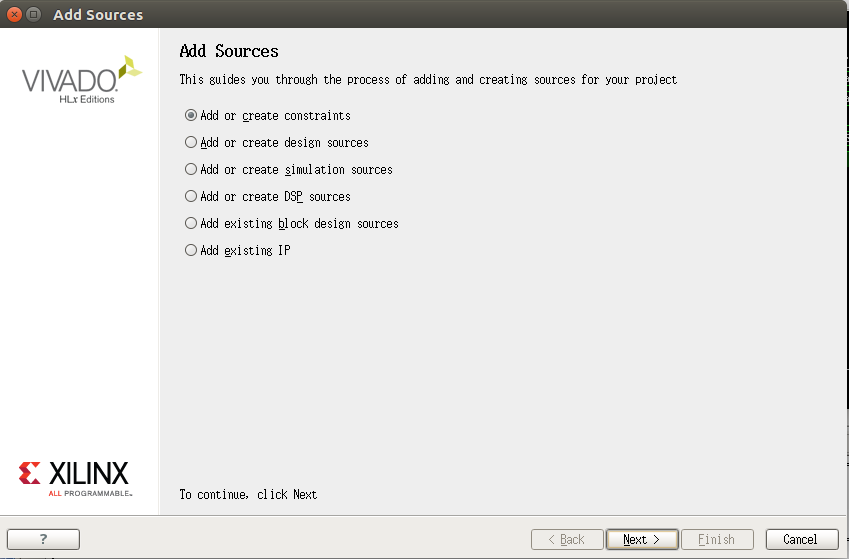


⑼ 根据数据选择器的逻辑表达式，如下。在代码编辑区，用assign实现该逻辑。led8\_out变量赋值语句为我们自己添加的代码。其他为自动生成的代码。

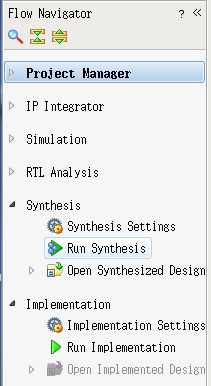
|  |
| --- |
| **module** mux4\_1**(**  **input** sw6\_a1**,**  **input** sw5\_a0**,**  **input** sw4\_d4**,**  **input** sw3\_d3**,**  **input** sw2\_d2**,**  **input** sw1\_d1**,**  **output** led8\_out  **);**  **assign** led8\_out **=** sw1\_d1**&(!**sw6\_a1**)&(!**sw5\_a0**)|**sw2\_d2**&(!**sw6\_a1**)&(** sw5\_a0**)|**  sw3\_d3**&(** sw6\_a1**)&(!**sw5\_a0**)|**sw4\_d4**&(** sw6\_a1**)&(** sw5\_a0**);**  **endmodule** |



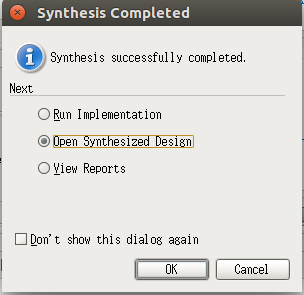
⑽ 添加模块中输入输出信号与开发板上芯片的引脚对应约束。在”Project Manager”下点击”Add sources”，选择”Add or create constraints”，类似前面的步骤创建mux4\_1.xdc。



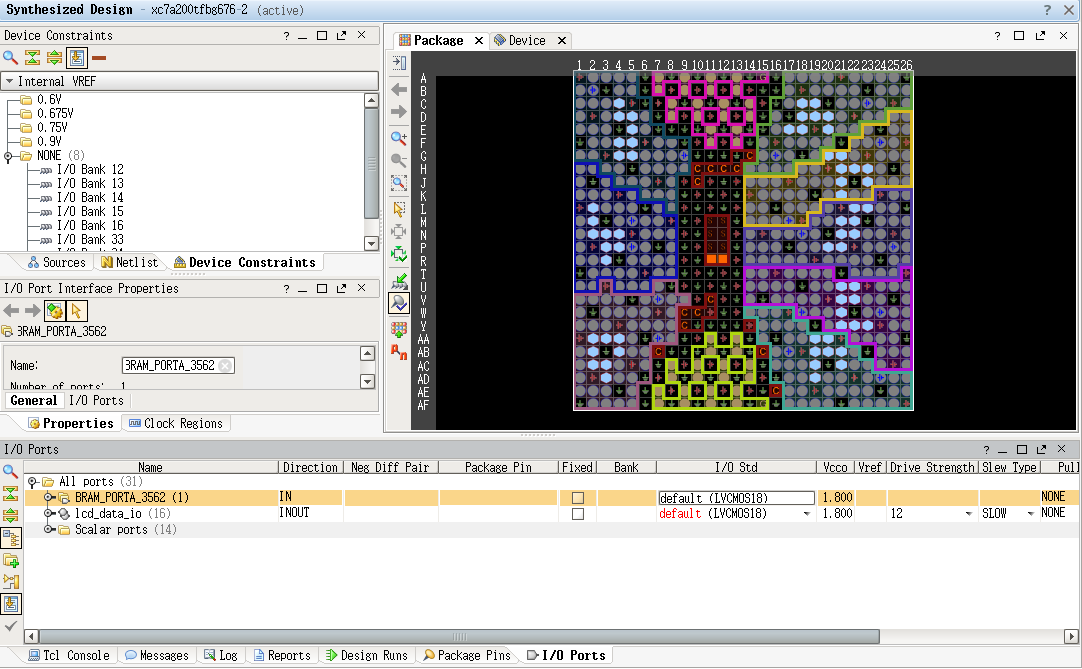
⑾ 点击Flow Navigator中Synthesis下Run Synthesis，进行综合：



综合完成后，选择”Open Syunthesized Design”，查看综合结果：

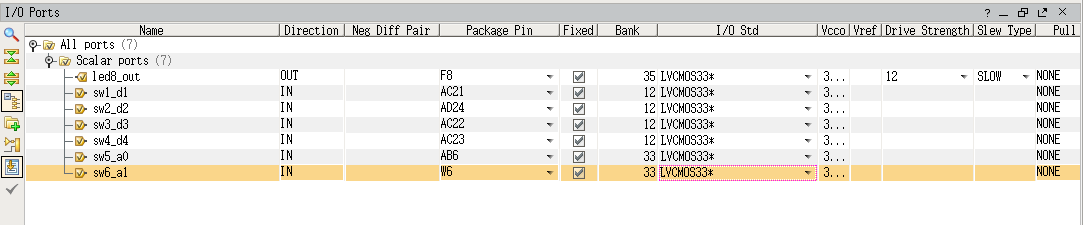


得到如下界面：



之后参照实验板原理图和引脚对应关系表，在I/O Ports栏下填入正确的Package Pin，针对本实验板，I/O Std要统一设置为LVCMOS33。

所有约束条件填写完成之后效果如下：

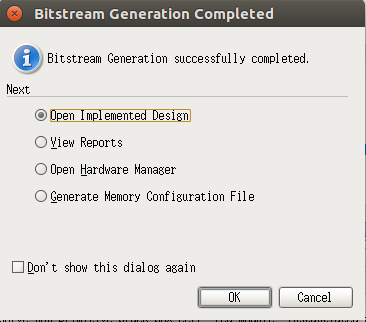


点击右上角的”保存”键即可保存为xdc文件。

保存后mux4\_1.xdc的内容如下：

|  |
| --- |
| set\_property PACKAGE\_PIN W6 [get\_ports sw6\_a1]  set\_property PACKAGE\_PIN AB6 [get\_ports sw5\_a0]  set\_property PACKAGE\_PIN AC21 [get\_ports sw1\_d1]  set\_property PACKAGE\_PIN AD24 [get\_ports sw2\_d2]  set\_property PACKAGE\_PIN AC22 [get\_ports sw3\_d3]  set\_property PACKAGE\_PIN AC23 [get\_ports sw4\_d4]  set\_property PACKAGE\_PIN F8 [get\_ports led8\_out]  set\_property IOSTANDARD LVCMOS33 [get\_ports sw1\_d1]  set\_property IOSTANDARD LVCMOS33 [get\_ports sw3\_d3]  set\_property IOSTANDARD LVCMOS33 [get\_ports led8\_out]  set\_property IOSTANDARD LVCMOS33 [get\_ports sw2\_d2]  set\_property IOSTANDARD LVCMOS33 [get\_ports sw4\_d4]  set\_property IOSTANDARD LVCMOS33 [get\_ports sw5\_a0]  set\_property IOSTANDARD LVCMOS33 [get\_ports sw6\_a1] |

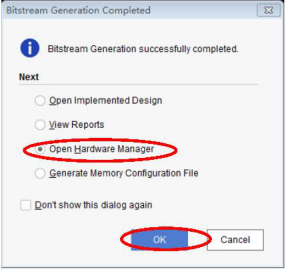
⑿ 后续的流程就是综合、布局布线和产生可烧写文件，可以依次双击运行，也可以只双击“Generate Bitsteam”会自动运行这三步，运行结果如下，可以选择Open Implemented Design来查看实现结果：



⒀ 在打开FPGA实验板，上电，并将下载线与电脑相连后，打开电源，FPGA板如下图：



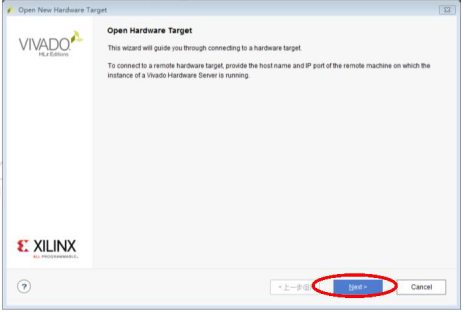
在比特流文件生成完成的窗口选择“Open Hardware Manager”，进入硬件管理界面。连接 FPGA开发板的电源线和与电脑的下载线，打开 FPGA电源。如下图：



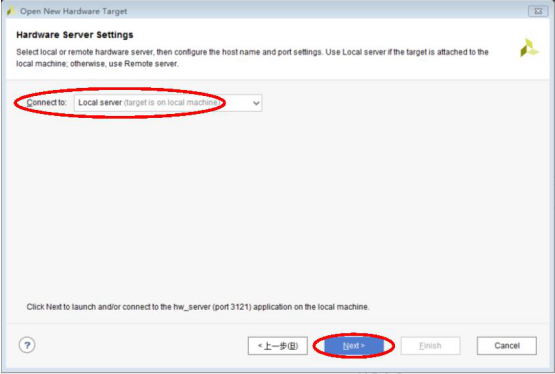
在“Hardware Manager”窗口的提示信息中，点击“ Open target”的下拉菜单的“ Open New Target”（或在“Flow Navigator”下“Program and Debug”中展开“Open Hardware Manager”，点击“Open Target”->“Open NewTarget”）。也可以选择“Auto Connect”自动连接器件。



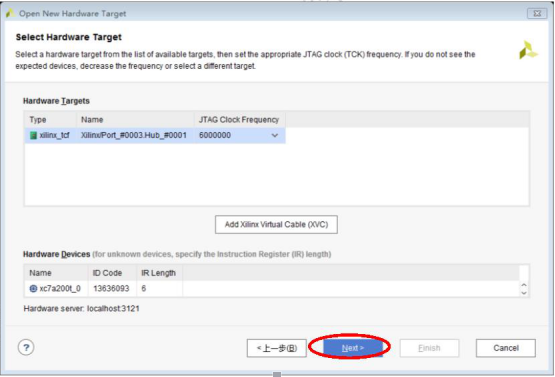
在“Open Hardware Target”向导中，先点击“Next”，进入 Server选择向导。如下图：



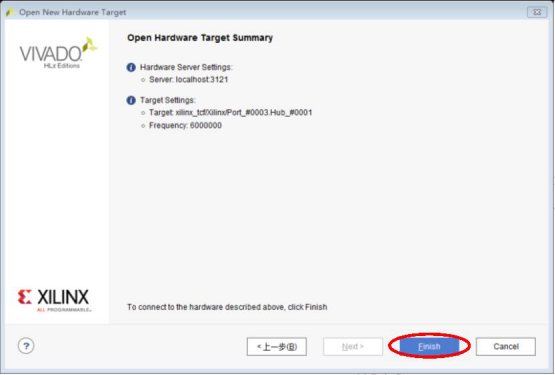
选择连接到“Local server”，点击“Next”。出现以下图：



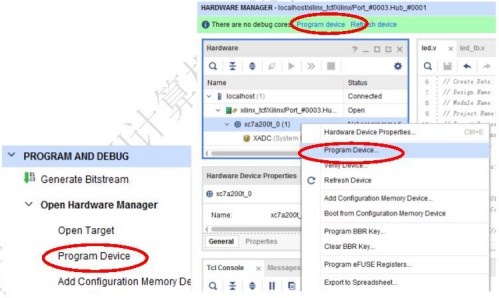
选择目标硬件，点击“Next”。



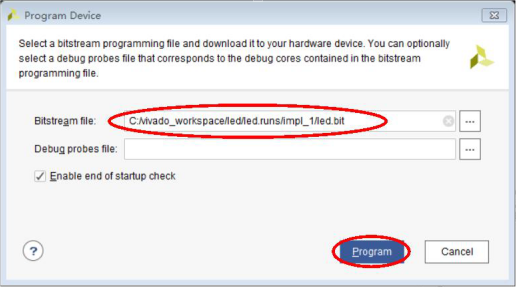
完成目标硬件打开。点击“Finish”。



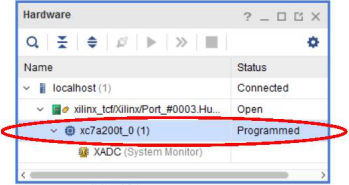
对目标硬件编程。在“ Hardware”窗口右键单击目标器件“ xc7a200t\_0”，选择“Program Device…”。或者“Flow Navigator”窗口中“Program and Debug”->“Hardware Manager”->“Program Device”。



选择下载的 bit流文件，点击“Program”。



完成下载后，“Hardware”窗口下的“xc7a200t\_0”的状态变成“Programmed”。

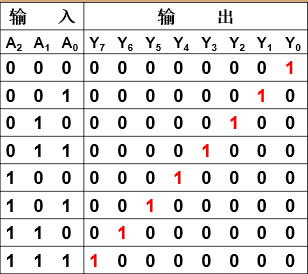


烧写bit文件，烧写完成后此时可以在FPGA实验板上进行实验操作。

⒅ 动拨码开关，观察LED8是否按照sw6和sw5，对sw1，sw2，sw3，sw4进行数据选择。

* + 1. **设计3-8译码器**

3-8译码器的真值表如下：



逻辑表达式为：

   ？

以U9单元的第3个拨码、第2个拨码、第1个拨码为输入，分别代表A\_2、A\_1、A\_0。

以开发板上从右至左的LED灯（LED1,LED2,LED3,LED4,LED5,LED6,LED7,LED8）依次表示Y0至Y7。

编写代码如下：

|  |
| --- |
| **module** decoder3\_8**(**  **input** Sw3\_A2**,**  **input** Sw2\_A1**,**  **input** Sw1\_A0**,**  **output** LED1\_Y0**,**  **output** LED2\_Y1**,**  **output** LED3\_Y2**,**  **output** LED4\_Y3**,**  **output** LED5\_Y4**,**  **output** LED6\_Y5**,**  **output** LED7\_Y6**,**  **output** LED8\_Y7  **);**  **assign** LED1\_Y0 **=** **!**Sw3\_A2**&!**Sw2\_A1**&!**Sw1\_A0**;**  **assign** LED2\_Y1 **=** **!**Sw3\_A2**&!**Sw2\_A1**&** Sw1\_A0**;**  **assign** LED3\_Y2 **=** **!**Sw3\_A2**&** Sw2\_A1**&!**Sw1\_A0**;**  **assign** LED4\_Y3 **=** **!**Sw3\_A2**&** Sw2\_A1**&** Sw1\_A0**;**  **assign** LED5\_Y4 **=** Sw3\_A2**&!**Sw2\_A1**&!**Sw1\_A0**;**  **assign** LED6\_Y5 **=** Sw3\_A2**&!**Sw2\_A1**&** Sw1\_A0**;**  **assign** LED7\_Y6 **=** Sw3\_A2**&** Sw2\_A1**&!**Sw1\_A0**;**  **assign** LED8\_Y7 **=** Sw3\_A2**&** Sw2\_A1**&** Sw1\_A0**;**  **endmodule** |

相应的引脚约束条件如下：

|  |
| --- |
| set\_property PACKAGE\_PIN AC22 [get\_ports Sw3\_A2]  set\_property PACKAGE\_PIN AD24 [get\_ports Sw2\_A1]  set\_property PACKAGE\_PIN AC21 [get\_ports Sw1\_A0]  set\_property PACKAGE\_PIN H7 [get\_ports LED1\_Y0]  set\_property PACKAGE\_PIN D5 [get\_ports LED2\_Y1]  set\_property PACKAGE\_PIN A3 [get\_ports LED3\_Y2]  set\_property PACKAGE\_PIN A5 [get\_ports LED4\_Y3]  set\_property PACKAGE\_PIN A4 [get\_ports LED5\_Y4]  set\_property PACKAGE\_PIN F7 [get\_ports LED6\_Y5]  set\_property PACKAGE\_PIN G8 [get\_ports LED7\_Y6]  set\_property PACKAGE\_PIN H8 [get\_ports LED8\_Y7]  set\_property IOSTANDARD LVCMOS33 [get\_ports LED1\_Y0]  set\_property IOSTANDARD LVCMOS33 [get\_ports LED2\_Y1]  set\_property IOSTANDARD LVCMOS33 [get\_ports LED3\_Y2]  set\_property IOSTANDARD LVCMOS33 [get\_ports LED4\_Y3]  set\_property IOSTANDARD LVCMOS33 [get\_ports LED5\_Y4]  set\_property IOSTANDARD LVCMOS33 [get\_ports LED6\_Y5]  set\_property IOSTANDARD LVCMOS33 [get\_ports LED7\_Y6]  set\_property IOSTANDARD LVCMOS33 [get\_ports LED8\_Y7]  set\_property IOSTANDARD LVCMOS33 [get\_ports Sw1\_A0]  set\_property IOSTANDARD LVCMOS33 [get\_ports Sw2\_A1]  set\_property IOSTANDARD LVCMOS33 [get\_ports Sw3\_A2] |

**附:**

**实验1结果：**

U9单元开关状态：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | sw6 | sw5 | sw1 | LEDR点亮 |
| 上 | 上 | 下 |
| 2 | sw6 | sw5 | sw2 | LEDR点亮 |
| 上 | 下 | 下 |
| 3 | sw6 | sw5 | sw3 | LEDR点亮 |
| 下 | 上 | 下 |
| 4 | sw6 | sw5 | sw4 | LEDR点亮 |
| 下 | 下 | 下 |

**实验2结果：**

U9单元的开关状态：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | sw3 | sw2 | sw1 | 灯 |
| 1 | 下 | 下 | 下 | LED8灭 |
| 2 | 下 | 下 | 上 | LED7灭 |
| 3 | 下 | 上 | 下 | LED6灭 |
| 4 | 下 | 上 | 上 | LED5灭 |
| 5 | 上 | 下 | 下 | LED4灭 |
| 6 | 上 | 下 | 上 | LED3灭 |
| 7 | 上 | 上 | 下 | LED2灭 |
| 8 | 上 | 上 | 上 | LED1灭 |

## 实验二 组合逻辑电路实验

1. **实验目的**

（1）熟悉数码管原理。

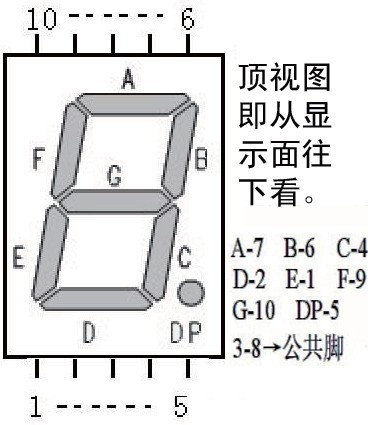
（2）掌握组合逻辑设计，熟悉数码管工作原理以及二进制与BCD码的转换。

1. **实验内容**

BCD码转换显示为10进制，即（0-9），从拨码开关输入二进制数，数码管显示输出的BCD码。

1. **实验步骤**
   * 1. **了解数码管原理。**

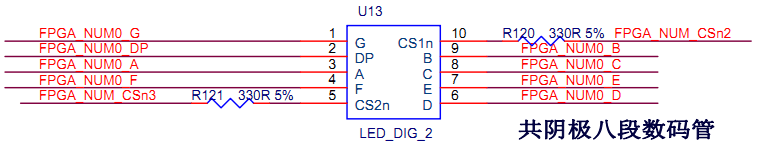
数码管分为共阴极和共阳极，共阴极数码管在控制端输入高电平，则数码管点亮；输入低电平，则数码管熄灭。1位数码管如下图所示。



对于双位数码管，如下图所示：



相应的控制原理图为：



管脚“1、2、3、4、6、7、8、9”分别控制点亮数码管的某段LED，高电平有效；管脚“5、10”分别控制显示哪个数字位，低电平有效。

本开发板所有的数码管使用相同的数据信号线，由FPGA\_NUM\_CSn0～7控制选择显示哪一位。

如果想让一个模块中的两个数码管都点亮，并且显示不同的数字，那么就需要利用人眼的暂留效应，对每个要显示的数码管进行动态扫描。即在极短得到时间（小于200ms）内，依次循环显示。

数值的显示可以采用如下逻辑：

|  |
| --- |
| **case** **(** num0\_scan\_data **)**  4'd0 **:** num0\_seg7 **<=** 7'b1111110**;** //0  4'd1 **:** num0\_seg7 **<=** 7'b0110000**;** //1  4'd2 **:** num0\_seg7 **<=** 7'b1101101**;** //2  4'd3 **:** num0\_seg7 **<=** 7'b1111001**;** //3  4'd4 **:** num0\_seg7 **<=** 7'b0110011**;** //4  4'd5 **:** num0\_seg7 **<=** 7'b1011011**;** //5  4'd6 **:** num0\_seg7 **<=** 7'b1011111**;** //6  4'd7 **:** num0\_seg7 **<=** 7'b1110000**;** //7  4'd8 **:** num0\_seg7 **<=** 7'b1111111**;** //8  4'd9 **:** num0\_seg7 **<=** 7'b1111011**;** //9  **default:**num0\_seg7**<=** 7'b0000000**;**  **endcase** |

* + 1. **下面介绍BCD码到10进制的转换**

8421码:又称BCD码，是最常用的十进制编码。其每位的权为8、4、2、1，按公式展开，即可得对应的十进制数，如(0101)2＝(8\*0)+(4\*1)+(2\*0)+(1\*1)＝(5)10。



实验时，要求从U9拨码开关单元的“4、3、2、1”位为输入，数码管U20单元的右侧数码管显示对应的十进制数值，左侧数码管关断。改变输入，使输出随之改变。

设计代码如下：

|  |
| --- |
| **module** BCD**(**  **input** **[**3**:**0**]** bcd\_num**,**  **output** **[**1**:**0**]** num1\_scan\_select**,**//选择FPGA\_NUM1 7段数码管的扫描位  **output** **[**7**:**0**]** num1\_seg7 //FPGA\_NUM1 7段数码管显示DP和a~g  **);**  **assign** num1\_scan\_select **=** 2'b10**;**  **assign** num1\_seg7**=**  **{**bcd\_num **==**4'b0000**}?**8'b01111110 **:**//0  **{**bcd\_num **==**4'b0001**}?**8'b00110000 **:**//1  **{**bcd\_num **==**4'b0010**}?**8'b01101101 **:**//2  **{**bcd\_num **==**4'b0011**}?**8'b01111001 **:**//3  **{**bcd\_num **==**4'b0100**}?**8'b00110011 **:**//4  **{**bcd\_num **==**4'b0101**}?**8'b01011011 **:**//5  **{**bcd\_num **==**4'b0110**}?**8'b01011111 **:**//6  **{**bcd\_num **==**4'b0111**}?**8'b01110000 **:**//7  **{**bcd\_num **==**4'b1000**}?**8'b01111111 **:**//8  **{**bcd\_num **==**4'b1001**}?**8'b01111011 **:**//9  8'b01111011**;**//9  **endmodule** |

介绍引脚约束代码如下：

|  |
| --- |
| set\_property PACKAGE\_PIN AC23 [get\_ports {bcd\_num[3]}]  set\_property PACKAGE\_PIN AC22 [get\_ports {bcd\_num[2]}]  set\_property PACKAGE\_PIN AD24 [get\_ports {bcd\_num[1]}]  set\_property PACKAGE\_PIN AC21 [get\_ports {bcd\_num[0]}]  set\_property PACKAGE\_PIN E26 [get\_ports {num1\_scan\_select[1]}]  set\_property PACKAGE\_PIN G25 [get\_ports {num1\_scan\_select[0]}]  set\_property PACKAGE\_PIN C4 [get\_ports {num1\_seg7[7]}]  set\_property PACKAGE\_PIN A2 [get\_ports {num1\_seg7[6]}]  set\_property PACKAGE\_PIN D4 [get\_ports {num1\_seg7[5]}]  set\_property PACKAGE\_PIN E5 [get\_ports {num1\_seg7[4]}]  set\_property PACKAGE\_PIN B4 [get\_ports {num1\_seg7[3]}]  set\_property PACKAGE\_PIN B2 [get\_ports {num1\_seg7[2]}]  set\_property PACKAGE\_PIN E6 [get\_ports {num1\_seg7[1]}]  set\_property PACKAGE\_PIN C3 [get\_ports {num1\_seg7[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {bcd\_num[3]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {bcd\_num[2]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {bcd\_num[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {bcd\_num[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_scan\_select[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_scan\_select[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[7]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[6]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[5]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[4]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[3]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[2]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[0]}] |

**附**

实验结果：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | sw3 | sw2 | sw1 | sw0 | 第六个数码管显示数字 |
| 1 | 上 | 上 | 上 | 上 | 0 |
| 2 | 上 | 上 | 上 | 下 | 1 |
| 3 | 上 | 上 | 下 | 上 | 2 |
| 4 | 上 | 上 | 下 | 下 | 3 |
| 5 | 上 | 下 | 上 | 上 | 4 |
| 6 | 上 | 下 | 上 | 下 | 5 |
| 7 | 上 | 下 | 下 | 上 | 6 |
| 8 | 上 | 下 | 下 | 下 | 7 |
| 9 | 下 | 上 | 上 | 上 | 8 |
| 10 | 下 | 上 | 上 | 下 | 9 |

其余情况也均显示“9”。

## 实验三 锁存器、时钟与触发器电路实验

1. **实验目的**

本实验的目的是复习锁存器和触发器的概念，并学会设计锁存器和触发器。

1. **实验内容**

（1）理解锁存器、触发器概念。

（2）设计RS、JK、D、T触发器。

1. **实验要求**

以拨码开关为输入，观察触发器的输出（LED灯为输出）。

1. **实验步骤**
   * 1. **RS锁存器实验**

下图为或非门组成的RS锁存器。其特性表如下表所示。



|  |  |  |  |
| --- | --- | --- | --- |
| S | R | Q | Q\* |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | S和R的1状态同时消失，次状态不定 |

特别注意：实验过程不能使S和R的1状态同时消失。

实验代码：

|  |
| --- |
| **module** rs\_latch**(**  **input** sw2\_R**,**  **input** sw1\_S**,**  **output** led8\_Q  **);**  **wire** temp**;**  **assign** led8\_Q **=** **!(**sw2\_R **|** temp**);**  **assign** temp **=** **!(**sw1\_S **|** led8\_Q**);**  **endmodule** |

约束文件：

|  |
| --- |
| set\_property PACKAGE\_PIN AD24 [get\_ports sw2\_R]  set\_property PACKAGE\_PIN AC21 [get\_ports sw1\_S]  set\_property PACKAGE\_PIN F8 [get\_ports led8\_Q]  set\_property IOSTANDARD LVCMOS33 [get\_ports led8\_Q]  set\_property IOSTANDARD LVCMOS33 [get\_ports sw1\_S]  set\_property IOSTANDARD LVCMOS33 [get\_ports sw2\_R] |

注意：要进行以下必要的设置，才能顺利地生成bit文件。

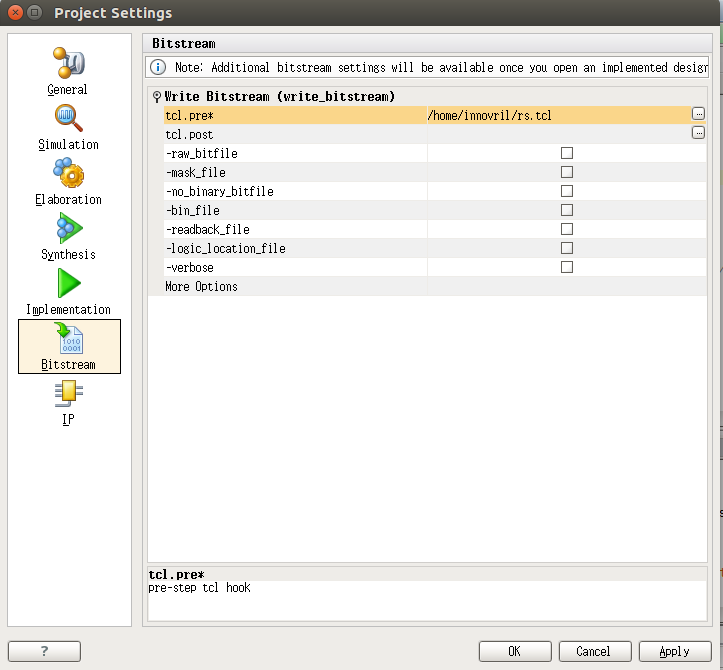
创建一个名为rs.tcl的文件，内容如下：

|  |
| --- |
| set\_property SEVERITY {Warning} [get\_drc\_checks LUTLP-1] |

之后点击”Flow Navigator”->”Program and Debug”->”Bitsteam Settings”：



在”tcl.pre”下加入之前创建的rs.tcl文件，点击”OK”：



* + 1. **时钟与触发器**

在某种条件触发的触发器中，除了置1、置0输入端外，又增加了一个触发信号输入端。只有触发信号发生某种变化后，触发器才能按照输入的置1，置0信号置成相应的状态。通常将这个触发信号称为时钟信号（CLOCK），记作CLK。

⑴ RS触发器

凡是在时钟信号作用下，逻辑功能符号下表所规定的逻辑功能者，无论触发方式如何，均称为SR触发器。

|  |  |  |  |
| --- | --- | --- | --- |
| S | R | Q | Q\* |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | S和R的1状态同时消失，次状态不定 |
| 1 | 1 | 1 | S和R的1状态同时消失，次状态不定 |

由上表得出RS触发器的特性方程：*Q\* = S + R’Q*

约束条件：SR = 0

下面以电平触发的RS触发器为例，进行FPGA实现。



设计代码如下：

|  |
| --- |
| **module** level\_rs\_latch**(**//电平触发的RS触发器  **input** sw3\_CLK**,**  **input** sw2\_R**,**  **input** sw1\_S**,**  **output** led8\_Q  **);**  **wire** s1**,**s2**;**  **assign** s1 **=** **!(**sw3\_CLK **&** sw1\_S**);**  **assign** s2 **=** **!(**sw3\_CLK **&** sw2\_R**);**  nand\_rs\_latch nand\_rs\_latch0**(.**s1**(**s1**),.**s2**(**s2**),.**Q**(**led8\_Q**));**  **endmodule**  //与非门实现的RS锁存器  **module** nand\_rs\_latch**(**  **input** s1**,**  **input** s2**,**  **output** Q  **);**  **wire** temp**;**  **assign** temp **=** **!(**Q**&**s2**);**  **assign** Q **=** **!(**s1**&**temp**);**    **endmodule** |

约束文件如下：

|  |
| --- |
| set\_property PACKAGE\_PIN AC22 [get\_ports sw3\_CLK]  set\_property PACKAGE\_PIN AD24 [get\_ports sw2\_R]  set\_property PACKAGE\_PIN AC21 [get\_ports sw1\_S]  set\_property PACKAGE\_PIN F8 [get\_ports led8\_Q]  set\_property IOSTANDARD LVCMOS33 [get\_ports led8\_Q]  set\_property IOSTANDARD LVCMOS33 [get\_ports sw1\_S]  set\_property IOSTANDARD LVCMOS33 [get\_ports sw2\_R]  set\_property IOSTANDARD LVCMOS33 [get\_ports sw3\_CLK] |

⑵ D触发器

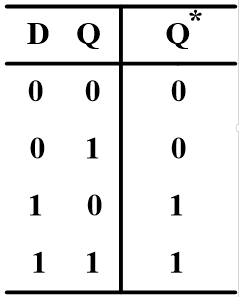
将SR触发器的S端，通过一级“非门”，连接到R端，重新命名S为D，那么这样得到的触发器为D触发器。如下图所示：



D触发器的特点是：当CLK为1的全部时间内，Q端的状态始终跟随D端的状态而改变。其特性方程为：

*Q\* = D*

其特性表如下：



对于D触发器，学生可以参考SR触发器，自己完成D触发器的实现。

⑶ JK触发器

本小节，介绍脉冲触发的JK触发器。为了提高触发器工作的可靠性，希望在每个时钟周期里输出端的状态只能改变一次。为此目的，在电平触发的触发器的基础上，又设计了脉冲触发的触发器。脉冲触发的JK触发器如下图所示：



特性方程为：

*Q\* = JQ’ + K’Q*

JK触发器的特性如下表所示：



脉冲触发的JK触发器的实现方式和电平触发的设计代码如下：

|  |
| --- |
| **module** pulse\_jk\_flipflop**(**  **input** sw3\_CLK**,**  **input** sw2\_J**,**  **input** sw1\_K**,**  **output** led8\_Q  **);**  **wire** s1\_m**,**s2\_m**,**s1\_s**,**s2\_s**,**temp\_out1**,**temp\_out2**,**temp\_q**;**//temp\_q即Q'    **assign** s1\_m **=** **!(**sw2\_J **&** sw3\_CLK **&** temp\_q**);**  **assign** s2\_m **=** **!(**sw1\_K **&** sw3\_CLK **&** led8\_Q**);**  **assign** s1\_s **=** **!(!**sw3\_CLK **&** temp\_out1**);**  **assign** s2\_s **=** **!(!**sw3\_CLK **&** temp\_out2**);**  nand\_rs\_latch nand\_rs\_latch\_m**(.**s1**(**s1\_m**),.**s2**(**s2\_m**),.**Q1**(**temp\_out1**),.**Q2**(**temp\_out2**));**  nand\_rs\_latch nand\_rs\_latch\_s**(.**s1**(**s1\_s**),.**s2**(**s2\_s**),.**Q1**(**led8\_Q**),.**Q2**(**temp\_q**));**    **endmodule**  //与非门实现的RS锁存器  **module** nand\_rs\_latch**(**  **input** s1**,**  **input** s2**,**  **output** Q1**,**  **output** Q2  **);**  **assign** Q1 **=** **!(**s1**&**Q2**);**  **assign** Q2 **=** **!(**Q1**&**s2**);**    **endmodule** |

约束文件如下：

|  |
| --- |
| set\_property PACKAGE\_PIN AC22 [get\_ports sw3\_CLK]  set\_property PACKAGE\_PIN AD24 [get\_ports sw2\_J]  set\_property PACKAGE\_PIN AC21 [get\_ports sw1\_K]  set\_property PACKAGE\_PIN F8 [get\_ports led8\_Q]  set\_property IOSTANDARD LVCMOS33 [get\_ports led8\_Q]  set\_property IOSTANDARD LVCMOS33 [get\_ports sw1\_K]  set\_property IOSTANDARD LVCMOS33 [get\_ports sw2\_J]  set\_property IOSTANDARD LVCMOS33 [get\_ports sw3\_CLK] |

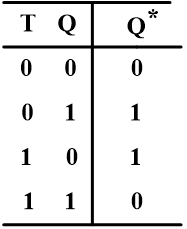
⑷ T触发器

在某些应用场合下，需要这样一种逻辑功能的触发器，当控制信号T=1时，每来一个时钟信号，它的状态就翻转一次；而当T=0时，时钟信号到达后它的状态保持不变。具备这种功能的触发器称为T触发器。

其特性方程为：

*Q\* = TQ’ + T’Q*

其特性表如下：



事实上，只要将JK触发器的两个输入端连接在一起作为T端，就可以构成T触发器。（补充：在需要SR触发器时，只要将JK触发器的J、K端当做S、R端使用，就可以实现SR触发器的功能，因此目前生产的触发器定型产品中只有JK触发器和D触发器两大类）。T触发器的电路结构如下图所示：



下面我们实现一种边沿触发的T触发器，并且利用开发板上的33MHz时钟作为真实的CLK信号，使得在T=0时，LED8保持不变，在T=1时，LED8以2Hz的频率进行闪烁。

设计代码如下：

|  |
| --- |
| **module** edge\_t\_flipflop**(**  **input** sw1\_t**,**  **input** clk**,**  //input sw3\_reset,  **output** **reg** led8\_Q  **);**  **reg** **[**24**:**0**]** count**;**  **parameter** COUNTER\_SUM **=** 25'd16500000**;** //33MHz  **always** **@(posedge** clk**)begin**  /\*if(sw3\_reset) begin  count <= 25'd0;  led8\_Q <= 1'b0;  end  else \*/**if(**count**<**COUNTER\_SUM**)begin**  count **<=** count **+** 25'd1**;**  **end**  **else** **begin**  led8\_Q **<=** sw1\_t**&!**led8\_Q **|** **!**sw1\_t**&**led8\_Q**;**  count **<=** 25'd0**;**  **end**  **end**    **endmodule** |

约束文件如下：

|  |
| --- |
| set\_property PACKAGE\_PIN AC19 [get\_ports clk]  set\_property PACKAGE\_PIN AC21 [get\_ports sw1\_t]  set\_property PACKAGE\_PIN F8 [get\_ports led8\_Q]  set\_property IOSTANDARD LVCMOS33 [get\_ports clk]  set\_property IOSTANDARD LVCMOS33 [get\_ports led8\_Q]  set\_property IOSTANDARD LVCMOS33 [get\_ports sw1\_t] |

**附：**

**RS锁存器实验结果：**

当U9单元的sw2 sw1初始输入为“下上”，则led灭。

当U9单元的sw2 sw1变为“上上”，则led保持不变，仍灭。

当U9单元的sw2 sw1变为“上下”，则led点亮。

当U9单元的sw2 sw1变为“上上”，则led保持不变，仍亮。

附：学生可以自行尝试用与非门编写RS锁存器。

**RS触发器实验结果：**

当sw3为“下”，sw2和sw1的操作和RS锁存器一样。

当sw3为“上”，sw2和sw1的操作均不会引起输出的变化。

当sw2 sw1为“下上”，将sw3置“下”，LED灯灭。

然后，将sw3置“上”，LED灯仍灭。

接着，将sw2 sw1置为“上下”，此时LED灯仍灭。

然后，将sw3置“下”，触发条件到来，LED灯点亮。

学生可以尝试其他顺序的情况，但特别注意，在sw3为“上”时，sw2和sw1不能同时从1变到0。

**JK触发器实验结果：**

U9拨码单元，sw2和sw1分别代表J和K

当sw2 sw1为“下上”时，sw3往上拨动，再往下拨动后，LED点亮。

改变sw2 sw1为“上下”时，sw3往上，再往下拨动后，LED熄灭。

改变sw2 sw1为“上上”时，sw3往上，再往下拨动后，LED保持不变。

改变sw2 sw1为“下下”时，sw3往上，再往下拨动后，LED点亮。

保持sw2 sw1为“下下”，sw3往上，再往下拨动后，LED熄灭。

**T触发器实验结果：**

对于U9单元，

当sw1为“下”时，LED以2Hz进行闪烁。

当sw1为“上”时，LED状态保持不变。

## 实验四 时序逻辑电路实验

1. **实验目的**

掌握时序逻辑电路基本概念，掌握时序逻辑电路分析方法的基础上，进一步掌握时序逻辑电路的设计方法。了解矩阵键盘扫描工作的原理。

1. **实验内容**

设计一个带进位输出端的13进制计数器。

1. **实验步骤**
   * 1. **基本概念**

在时序逻辑电路中根据触发器的动作特点不同，分为同步时序电路和异步时序电路。在同步时序电路中，所有触发器状态的变化都是在同一时钟信号操作下同时发生的。而在异步时序电路中，触发器状态的变化不是同时发生的。有时还根据输入信号的特点将时序电路划分为米利型和摩尔型。在米利型电路中，输出信号不仅取决于存储电路的状态，而且还取决于输入变量；在摩尔型电路中，输出信号仅仅取决于存储电路的状态。

鉴于时序电路在工作时是在电路的有限个状态间按照一定的规律转换的，所以又将时序电路称为状态机。

* + 1. **设计一个带进位输出端的13进制计数器**

设计要求：以开发板上复位按钮sw17(reset button)键为模拟时钟，以U20单元的右侧数码管实时监测电路内部状态。每按一次按键，U20单元右侧数码管都能反应电路状态的变化（即，每按一次，数字加1），逢13进位，在LED8上显示（即，点亮LED8）。

设计思路：

1、进行逻辑抽象。

2、状态化简

3、状态分配

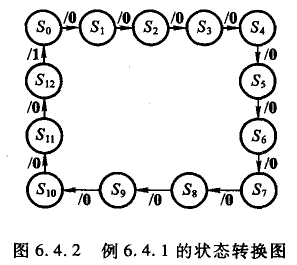
4、选择触发器类型，求出电路的状态方程、驱动方程和输出方程

5、根据得到的方程式画出逻辑图

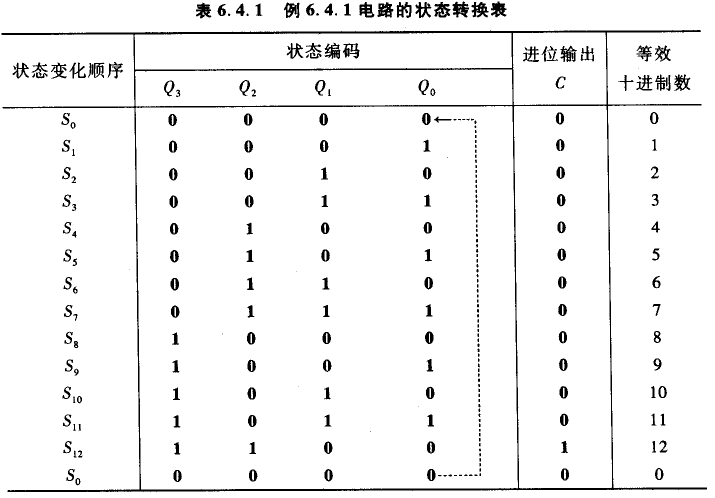
6、检查设计的电路是否能够自启动

请参考清华大学闫石主编的《数字电路技术基础》（第5版）第316页，例题6.4.1

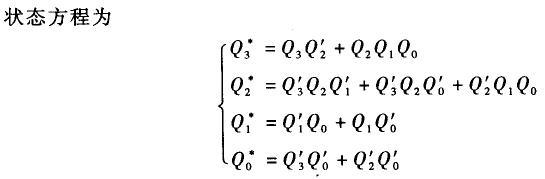
该13进制状态转换图如下：



状态转换表如下：



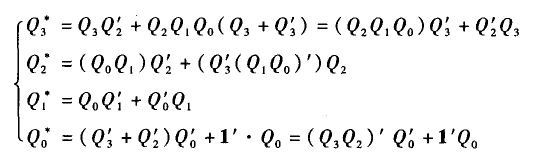
状态方程：



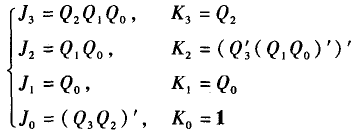
输出方程为：

*C = Q3Q2*

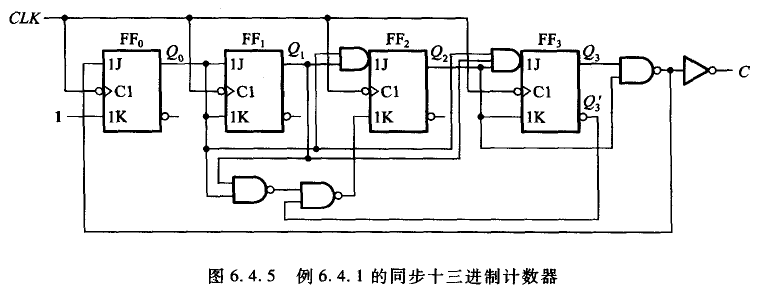
如果使用JK触发器组成这个电路，则将状态方程修改为如下形式：



各个驱动器的驱动方程为：



计数器逻辑电路图如下：



下面，我们在开发板上实现该13进制计数器。

|  |
| --- |
| **module** counter\_13**(**  **input** button\_clk**,**  **input** sw1\_reset**,**    **output** **reg** led8\_C**,**  **output** **reg[**1**:**0**]** num1\_scan\_select**,**//选择FPGA\_NUM1 7段数码管的扫描位  **output** **[**7**:**0**]** num1\_seg7 //FPGA\_NUM1 7段数码管显示DP和a~g  **);**  **reg** Q3**,**Q2**,**Q1**,**Q0**;**  **wire** **[**3**:**0**]** state\_num**;**  **always** **@(posedge** button\_clk **)** **begin**  **if(**sw1\_reset**)** **begin**  Q3 **<=** 1'b0**;**  Q2 **<=** 1'b0**;**  Q1 **<=** 1'b0**;**  Q0 **<=** 1'b0**;**  num1\_scan\_select **<=** 2'b11**;**//不选  led8\_C **<=** 1'b0**;**  **end**  **else** **begin**  Q3 **<=** Q3**&!**Q2 **|** Q2**&**Q1**&**Q0**;**  Q2 **<=** **!**Q3**&**Q2**&!**Q1 **|** **!**Q3**&**Q2**&!**Q0 **|** **!**Q2**&**Q1**&**Q0**;**  Q1 **<=** **!**Q1**&**Q0 **|** Q1**&!**Q0**;**  Q0 **<=** **!**Q3**&!**Q0 **|** **!**Q2**&!**Q0**;**  led8\_C **<=** **(**state\_num **==** 4'b1100**)?**1'b1**:**1'b0**;**  num1\_scan\_select **<=** 2'b10**;**//选择显示的数码管  **end**//else  **end**//always  //--------------display-------------------------  **assign** state\_num **=** **{**Q3**,**Q2**,**Q1**,**Q0**};** //采集电路内部状态  **assign** num1\_seg7**=**  **{**state\_num **==**4'b0000**}?**8'b01111110 **:**//0  **{**state\_num **==**4'b0001**}?**8'b00110000 **:**//1  **{**state\_num **==**4'b0010**}?**8'b01101101 **:**//2  **{**state\_num **==**4'b0011**}?**8'b01111001 **:**//3  **{**state\_num **==**4'b0100**}?**8'b00110011 **:**//4  **{**state\_num **==**4'b0101**}?**8'b01011011 **:**//5  **{**state\_num **==**4'b0110**}?**8'b01011111 **:**//6  **{**state\_num **==**4'b0111**}?**8'b01110000 **:**//7  **{**state\_num **==**4'b1000**}?**8'b01111111 **:**//8  **{**state\_num **==**4'b1001**}?**8'b01111011 **:**//9  **{**state\_num **==**4'b1010**}?**8'b01110111 **:**//10 A  **{**state\_num **==**4'b1011**}?**8'b00011111 **:**//11 b  **{**state\_num **==**4'b1100**}?**8'b01001110 **:**//12 C  8'b01001110**;**//12  **endmodule** |

约束文件：

|  |
| --- |
| set\_property PACKAGE\_PIN Y5 [get\_ports button\_clk]  set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets button\_clk\_IBUF]  set\_property PACKAGE\_PIN AC21 [get\_ports sw6\_reset]  set\_property PACKAGE\_PIN D6 [get\_ports led8\_C]  set\_property PACKAGE\_PIN E26 [get\_ports {num1\_scan\_select[1]}]  set\_property PACKAGE\_PIN G25 [get\_ports {num1\_scan\_select[0]}]  set\_property PACKAGE\_PIN C4 [get\_ports {num1\_seg7[7]}]  set\_property PACKAGE\_PIN A2 [get\_ports {num1\_seg7[6]}]  set\_property PACKAGE\_PIN D4 [get\_ports {num1\_seg7[5]}]  set\_property PACKAGE\_PIN E5 [get\_ports {num1\_seg7[4]}]  set\_property PACKAGE\_PIN B4 [get\_ports {num1\_seg7[3]}]  set\_property PACKAGE\_PIN B2 [get\_ports {num1\_seg7[2]}]  set\_property PACKAGE\_PIN E6 [get\_ports {num1\_seg7[1]}]  set\_property PACKAGE\_PIN C3 [get\_ports {num1\_seg7[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_scan\_select[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_scan\_select[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[7]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[6]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[5]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[4]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[3]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[2]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports button\_clk]  set\_property IOSTANDARD LVCMOS33 [get\_ports led8\_C]  set\_property IOSTANDARD LVCMOS33 [get\_ports sw6\_reset] |

注：因为Vivado 默认不支持用按键做always语句的触发条件，所以要加上橘色的语句。

**附：**

**实验结果：**

当把U9单元sw1置为“下”时，按一下Reset button键，电路进入复位状态。

当将U9单元sw1置为“上”时，按动Reset button键，每按一下，数码管数字加1，该数码管显示了内部电路各个触发器的翻转状态。

当数码管数值为“C”，也即12时，再按一下Reset button键，数码管数值变为0，同时LED灯点亮，表示进位输出。

拓展：学生可以在次基础上，尝试使用实例化4个JK触发器实现，也可以使用行为级描述实现该功能。

## 实验五 存储器实验

1. **实验目的**

（1）掌握如何使用Verilog定义多维数组存储器。

（2）掌握存储器的读写方法。

（3）理解存储器使能、读写使能、地址总线和数据总线的意义。

1. **实验内容**

（1）实现RAM并实现RAM的读写操作。

（2）使RAM的读写过程可视化。

读写过程可视化是指在写数据过程，通过拨码开关设置好数据，同时将该数据显示在数码管上。按一下按键后，数值写入RAM中；然后设置新地址和新数据，再按一次按键，又将数据写入新地址。

读数据时，设置好使能位及地址后，每按一次按键，读出一个n位2进制数，显示和数码管和LED灯上。

1. **实验步骤**
   * 1. **背景知识**

在Verilog代码中，可以用多维数组定义存储器。一个32字节的8位存储器块，可以定义为32\*8的数组，在Verilog语言中可以用下面的语句来定义：reg[7:0] memory\_array [31:0];

* + 1. **实验内容**

请用上述方法实现一个4\*4的RAM，编译工程，并将工程下载至开发板上，用拨码开关作为地址、数据和控制信号的输入端；用LED和数码管作为输出显示端，先向每一个RAM输入特定数据，然后逐个读出显示，观察运行结果，是否满足设计要求。

实验代码如下：

|  |
| --- |
| **module** rw\_ram**(**  **input** row1\_clk**,**    **input** u9sw6\_ram\_en**,** //RAM使能  **input** u9sw5\_r\_en**,** //读使能  **input** **[**1**:**0**]** u10\_addr**,** //4个地址,U10单元的{sw2,sw1}  **input** **[**3**:**0**]** data\_in**,** //4位数据输入U10单元的{sw6,sw5,sw4,sw3}    **output** **[**3**:**0**]** data\_out\_led**,** //最右边4个led：{led13,led1,led2,led3}  **output** **[**3**:**0**]** col1\_4**,**  **output** **[**1**:**0**]** num1\_scan\_select**,**//选择FPGA\_NUM1 7段数码管的扫描位  **output** **[**7**:**0**]** num1\_seg7 //FPGA\_NUM1 7段数码管显示DP和a~g  **);**  **reg[**3**:**0**]** ram**[**3**:**0**];**//4行，每行4位  **reg[**3**:**0**]** data\_out**;**    **always** **@(posedge** row1\_clk**)**  **begin**  **if(**u9sw6\_ram\_en**)**//RAM使能  **begin**  **if(**u9sw5\_r\_en**)**//读数据  **begin**  data\_out **<=**ram**[**u10\_addr**];**  **end**//if  **else** **begin** //写数据  ram**[**u10\_addr**]<=**data\_in**;**  **end**  **end**//if  **end**//always  //----------check the key:col1row1----------  **assign** col1\_4 **=** 4'b0111**;**  **assign** data\_out\_led **=** **~**data\_out**;**//灯亮为1，灯灭为0  **assign** num1\_scan\_select **=** 2'b10**;**//选择显示的数码管    **assign** num1\_seg7**=**  **{(**data\_in **==**4'b0000**)&!**u9sw5\_r\_en**|(**data\_out **==**4'b0000**)&**u9sw5\_r\_en**}?**8'b01111110 **:**//0  **{(**data\_in **==**4'b0001**)&!**u9sw5\_r\_en**|(**data\_out **==**4'b0001**)&**u9sw5\_r\_en**}?**8'b00110000 **:**//1  **{(**data\_in **==**4'b0010**)&!**u9sw5\_r\_en**|(**data\_out **==**4'b0010**)&**u9sw5\_r\_en**}?**8'b01101101 **:**//2  **{(**data\_in **==**4'b0011**)&!**u9sw5\_r\_en**|(**data\_out **==**4'b0011**)&**u9sw5\_r\_en**}?**8'b01111001 **:**//3  **{(**data\_in **==**4'b0100**)&!**u9sw5\_r\_en**|(**data\_out **==**4'b0100**)&**u9sw5\_r\_en**}?**8'b00110011 **:**//4  **{(**data\_in **==**4'b0101**)&!**u9sw5\_r\_en**|(**data\_out **==**4'b0101**)&**u9sw5\_r\_en**}?**8'b01011011 **:**//5  **{(**data\_in **==**4'b0110**)&!**u9sw5\_r\_en**|(**data\_out **==**4'b0110**)&**u9sw5\_r\_en**}?**8'b01011111 **:**//6  **{(**data\_in **==**4'b0111**)&!**u9sw5\_r\_en**|(**data\_out **==**4'b0111**)&**u9sw5\_r\_en**}?**8'b01110000 **:**//7  **{(**data\_in **==**4'b1000**)&!**u9sw5\_r\_en**|(**data\_out **==**4'b1000**)&**u9sw5\_r\_en**}?**8'b01111111 **:**//8  **{(**data\_in **==**4'b1001**)&!**u9sw5\_r\_en**|(**data\_out **==**4'b1001**)&**u9sw5\_r\_en**}?**8'b01111011 **:**//9  **{(**data\_in **==**4'b1010**)&!**u9sw5\_r\_en**|(**data\_out **==**4'b1010**)&**u9sw5\_r\_en**}?**8'b01110111 **:**//10 A  **{(**data\_in **==**4'b1011**)&!**u9sw5\_r\_en**|(**data\_out **==**4'b1011**)&**u9sw5\_r\_en**}?**8'b00011111 **:**//11 b  **{(**data\_in **==**4'b1100**)&!**u9sw5\_r\_en**|(**data\_out **==**4'b1100**)&**u9sw5\_r\_en**}?**8'b01001110 **:**//12 C  **{(**data\_in **==**4'b1101**)&!**u9sw5\_r\_en**|(**data\_out **==**4'b1101**)&**u9sw5\_r\_en**}?**8'b00111101 **:**//13 d  **{(**data\_in **==**4'b1110**)&!**u9sw5\_r\_en**|(**data\_out **==**4'b1110**)&**u9sw5\_r\_en**}?**8'b01001111 **:**//14 E  **{(**data\_in **==**4'b1111**)&!**u9sw5\_r\_en**|(**data\_out **==**4'b1111**)&**u9sw5\_r\_en**}?**8'b01000111 **:**//15 F  8'b01000111**;**  **endmodule** |

约束文件如下：

|  |
| --- |
| set\_property PACKAGE\_PIN U7 [get\_ports row1\_clk]  set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets row1\_clk\_IBUF]  set\_property PACKAGE\_PIN AC22 [get\_ports u9sw6\_ram\_en]  set\_property PACKAGE\_PIN AC23 [get\_ports u9sw5\_r\_en]  set\_property PACKAGE\_PIN AD24 [get\_ports {u10\_addr[1]}]  set\_property PACKAGE\_PIN AC21 [get\_ports {u10\_addr[0]}]  set\_property PACKAGE\_PIN Y6 [get\_ports {data\_in[3]}]  set\_property PACKAGE\_PIN AA7 [get\_ports {data\_in[2]}]  set\_property PACKAGE\_PIN W6 [get\_ports {data\_in[1]}]  set\_property PACKAGE\_PIN AB6 [get\_ports {data\_in[0]}]  set\_property PACKAGE\_PIN A5 [get\_ports {data\_out\_led[3]}]  set\_property PACKAGE\_PIN A3 [get\_ports {data\_out\_led[2]}]  set\_property PACKAGE\_PIN D5 [get\_ports {data\_out\_led[1]}]  set\_property PACKAGE\_PIN H7 [get\_ports {data\_out\_led[0]}]  set\_property PACKAGE\_PIN V8 [get\_ports {col1\_4[3]}]  set\_property PACKAGE\_PIN V9 [get\_ports {col1\_4[2]}]  set\_property PACKAGE\_PIN Y8 [get\_ports {col1\_4[1]}]  set\_property PACKAGE\_PIN V7 [get\_ports {col1\_4[0]}]  set\_property PACKAGE\_PIN E26 [get\_ports {num1\_scan\_select[1]}]  set\_property PACKAGE\_PIN G25 [get\_ports {num1\_scan\_select[0]}]  set\_property PACKAGE\_PIN C4 [get\_ports {num1\_seg7[7]}]  set\_property PACKAGE\_PIN A2 [get\_ports {num1\_seg7[6]}]  set\_property PACKAGE\_PIN D4 [get\_ports {num1\_seg7[5]}]  set\_property PACKAGE\_PIN E5 [get\_ports {num1\_seg7[4]}]  set\_property PACKAGE\_PIN B4 [get\_ports {num1\_seg7[3]}]  set\_property PACKAGE\_PIN B2 [get\_ports {num1\_seg7[2]}]  set\_property PACKAGE\_PIN E6 [get\_ports {num1\_seg7[1]}]  set\_property PACKAGE\_PIN C3 [get\_ports {num1\_seg7[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {col1\_4[3]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {col1\_4[2]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {col1\_4[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {col1\_4[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {data\_in[2]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {data\_in[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {data\_in[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {data\_in[3]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {data\_out\_led[2]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {data\_out\_led[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {data\_out\_led[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {data\_out\_led[3]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_scan  \_select[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_scan  \_select[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[7]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[6]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[5]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[4]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[3]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[2]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {num1\_seg7[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {u10\_addr[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {u10\_addr[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports row1\_clk]  set\_property IOSTANDARD LVCMOS33 [get\_ports u9sw5\_r\_en]  set\_property IOSTANDARD LVCMOS33 [get\_ports u9sw6\_ram\_en] |

**附：**

**实验结果：**

将U9单元sw6 sw5 置为“下上”，进行写RAM操作。这种情况下：

置U10 “1000 00”，表示在00地址处写入8，按一下col1row1键，即完成对地址00写入8。

置U10 “0100 01”，表示在01地址处写入4，按一下col1row1键，即完成对地址01写入4。

置U10 “0010 10”，表示在10地址处写入2，按一下col1row1键，即完成对地址10写入2。

置U10 “0001 11”，表示在11地址处写入1，按一下col1row1键，即完成对地址11写入1。

将U9单元sw6 sw5 置为“下下”，进行读RAM操作。这种情况下：

置U10的sw2 sw1为“00”，按一下col1row1键，数码管显示“8”，同时右侧第4个数码管点亮。

置U10的sw2 sw1为“01”，按一下col1row1键，数码管显示“4”，同时右侧第3个数码管点亮。

置U10的sw2 sw1为“10”，按一下col1row1键，数码管显示“2”，同时右侧第2个数码管点亮。

置U10的sw2 sw1为“11”，按一下col1row1键，数码管显示“1”，同时右侧第1个数码管点亮。

## 实验六 综合实验（数字时钟）

1. **实验目的**

（1）熟悉数字逻辑电路。

（2）掌握数字电路系统的设计方法。

（3）熟悉矩阵键盘使用方法。

1. **实验内容**

实现数字时钟。

1. **实验要求**

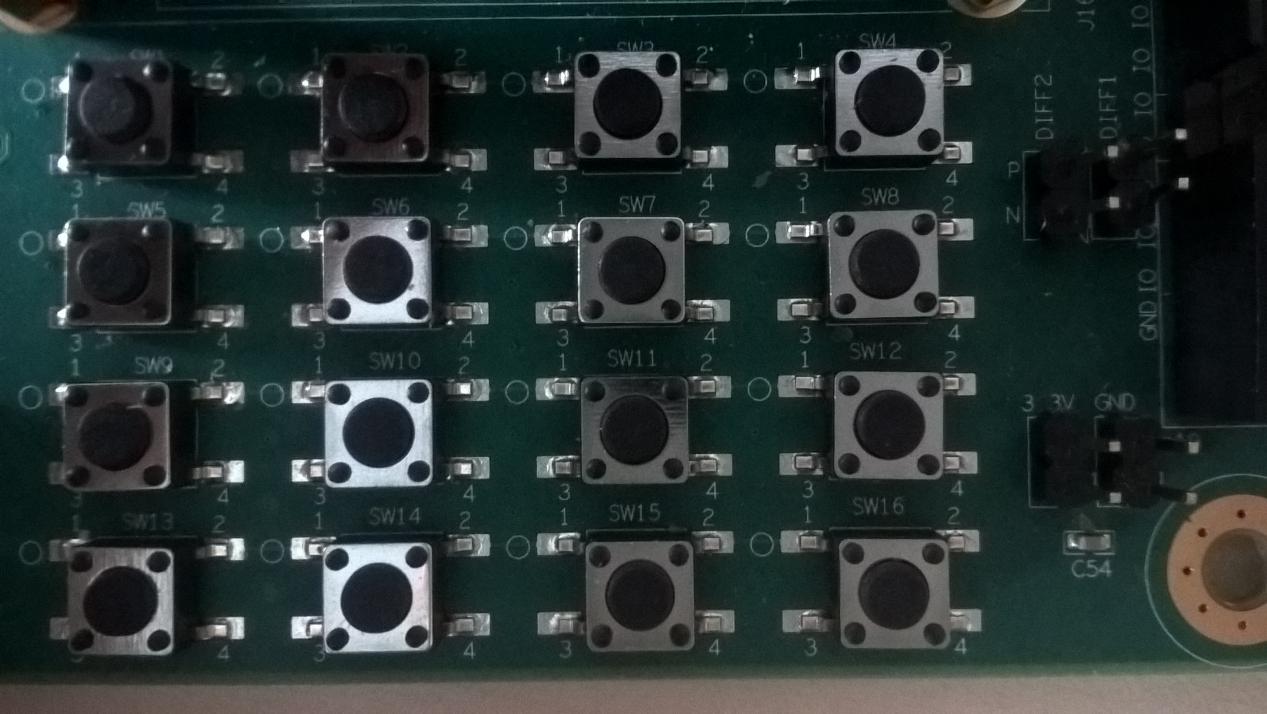
（1）准确计时，以数字形式显示时、分、秒。

（2）小时的计时要求为“12翻1”，分和秒的计时要求为60进位。

（3）能校正时间

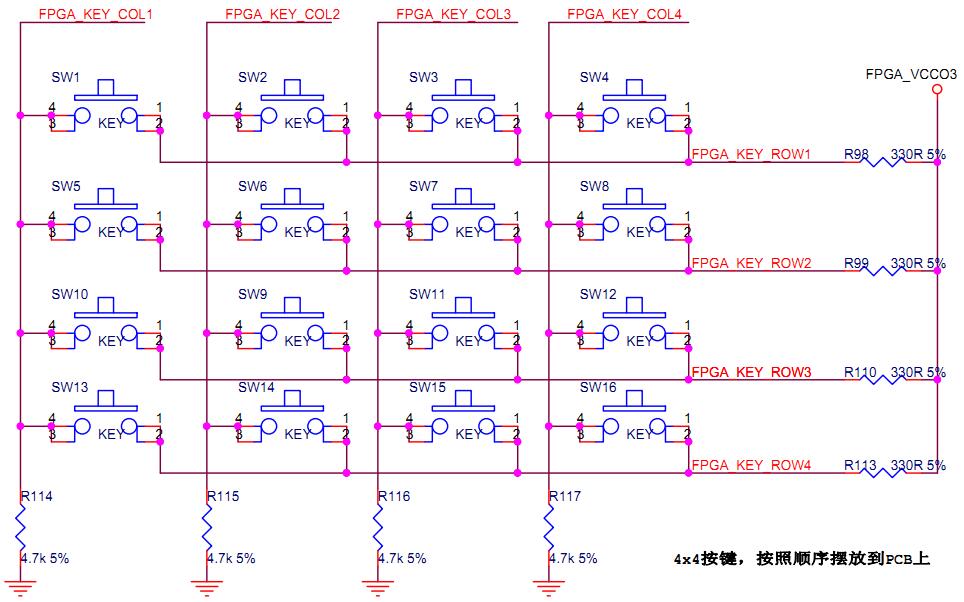
说明：数字钟是能够准确的显示时、分、秒时间，显示时间有误差可以校时。本节综合实验需要用到数码管显示时间，用到按键调整时间，用到时钟进行精确计时。数码管、时钟的使用方法均可参考前面实验章节。

下面结合开发板，介绍矩阵键盘的原理，实物图和原理图均见下图。



开发板上有一个4X4的矩阵键盘，从左到右有4列：col1、col2、col3、col4，从上到下有4行：row 1，row 2，row 3，row 4。

下面结合矩阵键盘原理图，介绍矩阵键盘原理以及对按下的1个键进行检测的方法。



在矩阵键盘中检测按下的按键是哪一个或哪些的检测方法如下：

首先，在（col1，col2，col3，col4）输入0111的情况下，在row1, row2, row3, row4处检测是否有0输出，有0输出的行表示该行的第1列的按键被按下。然后，在（col1，col2，col3，col4）输入1011的情况下，在row1, row2, row3, row4处检测是否有0输出，有0输出的行表示该行的第2列的按键被按下。接着，在（col1，col2，col3，col4）输入1101的情况下，在row1, row2, row3, row4处检测是否有0输出，有0输出的行表示该行的第3列的按键被按下。最后，在（col1，col2，col3，col4）输入1110的情况下，在row1, row2, row3, row4处检测是否有0输出，有0输出的行表示该行的第4列的按键被按下。

例如：当按钮（col1， row1）被按下时，给该按钮所在列col1输入0，则会在所在行row1检测到一个0输出。

当按键（col1,row1）(col1,row3)被同时按下时，则在（col1，col2，col3，col4）输入0111的情况下，检测到row1, row2, row3, row4的情况为0101。

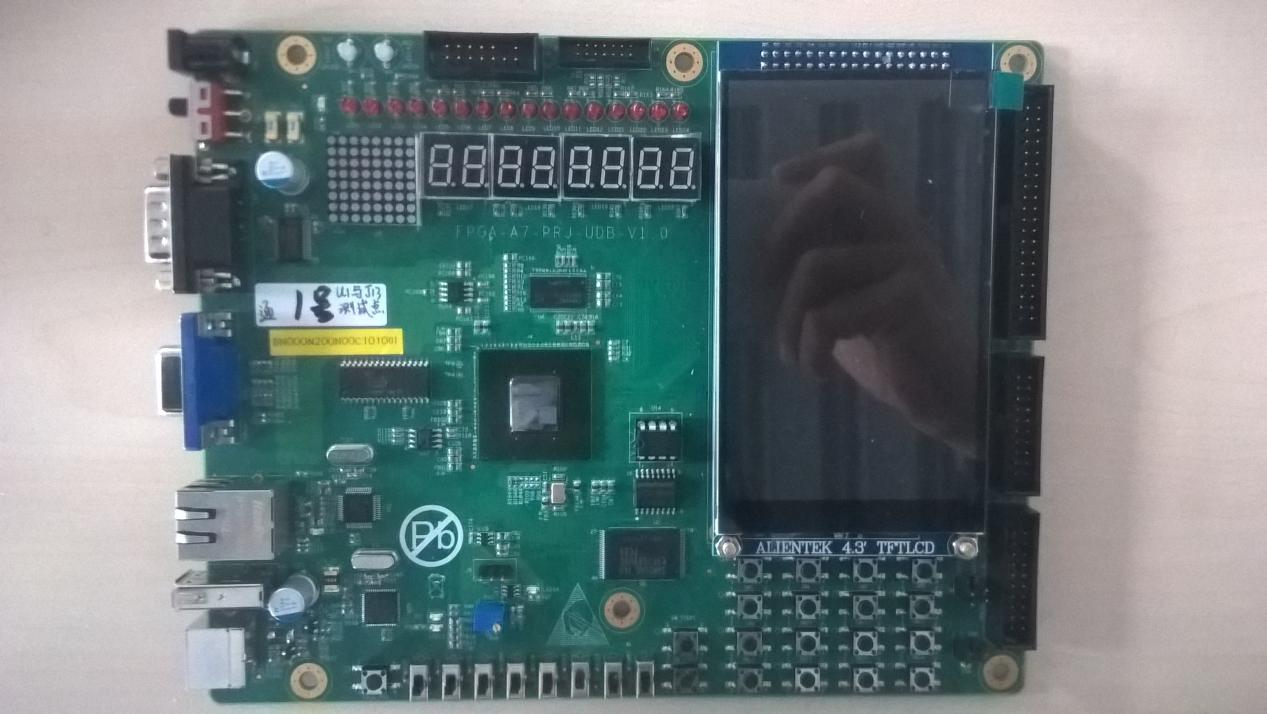
基于以上原理，可设定一个机制扫描键盘以确定按下了哪些键。

1. **实验步骤**

设计思路：

以U11单元的数码管显示小时，以U13单元的数码管显示分钟，以U20单元的数码管显示秒。

当U9单元最右侧的拨码开关为置位时，按绿色框中的按键，可以对每一位的数据进行调节，以校对时钟。当U9单元最右侧的拨码开关为复位时，时钟开始正常计时。



设置键

复位和设置

显示时间

说明：学生不必拘泥参考设计，可以自行设计更节约按键的其他设计方式。