龙芯中科
LOONGSON TECHNOLOGY

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 1.0

龙芯中科技术有限公司
Loongson Technology Corporation Limited

目　录

实验一 基本门电路与 FPGA 环境熟悉

1.1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。

1.2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）。

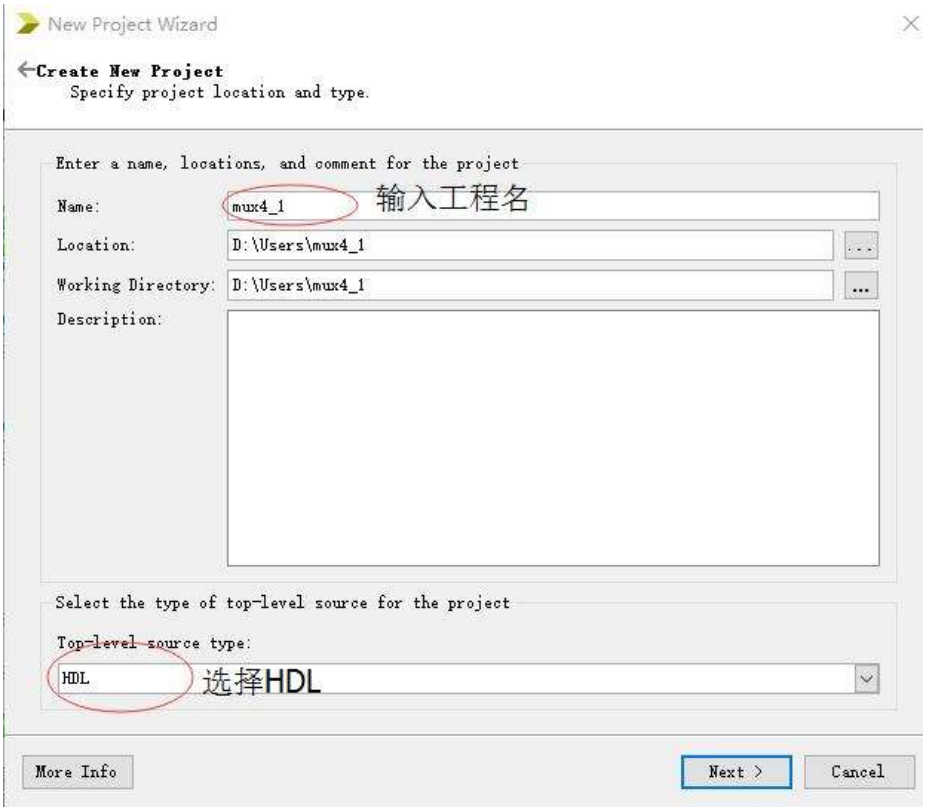
1.3实验步骤

1.3.1 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。

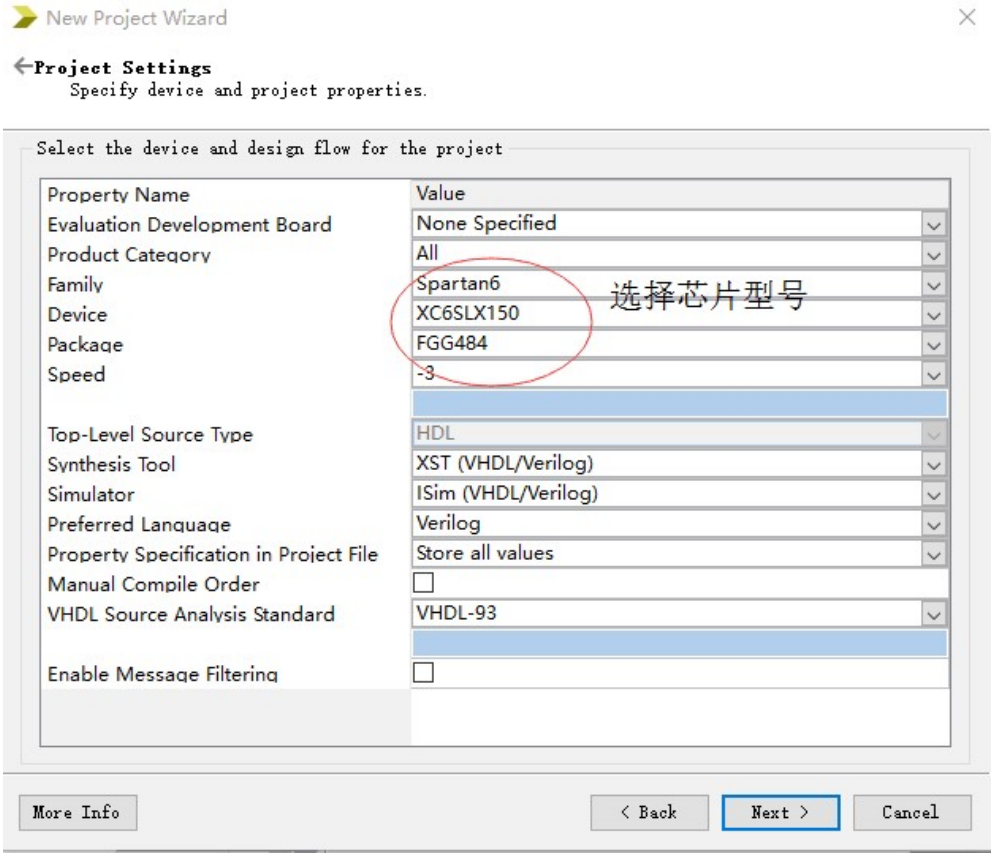
⑴ 开发环境采用 Xilinx 公司的 ISE Design Suite 14.3，软件图标如下图所示：

⑵ 双击“ISE Design Suit 14.3”，打开软件后，“File”->”New Project”。

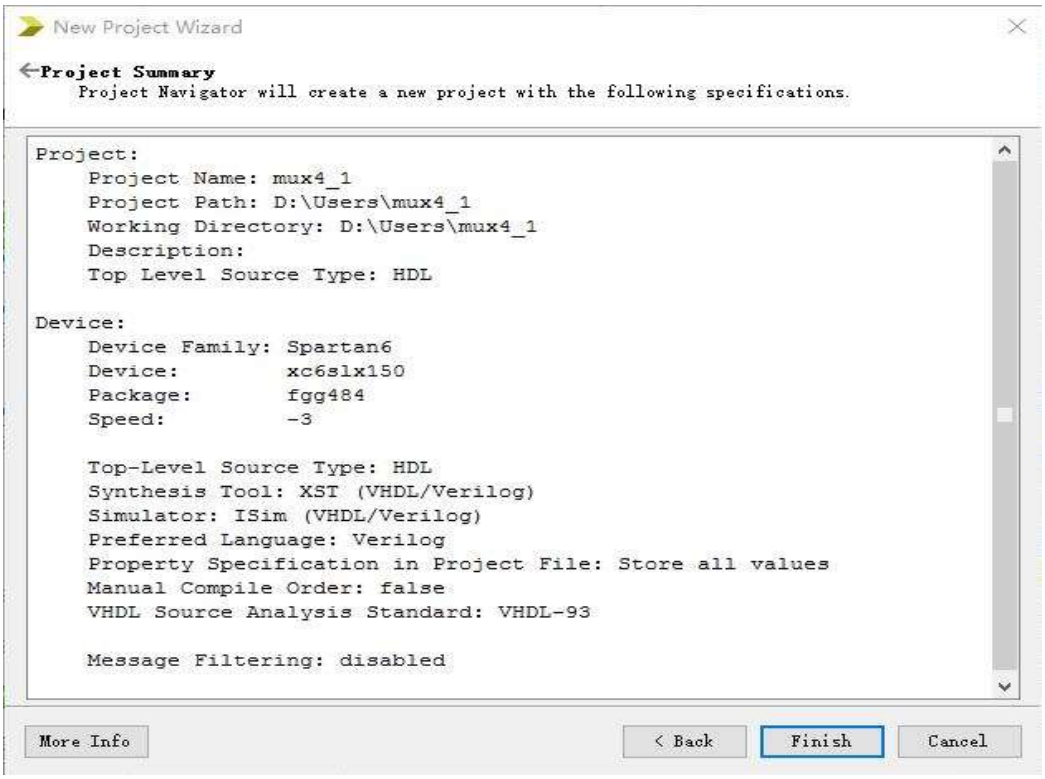
⑶ 输入工程名，选择 HDL，点击“下一步”。



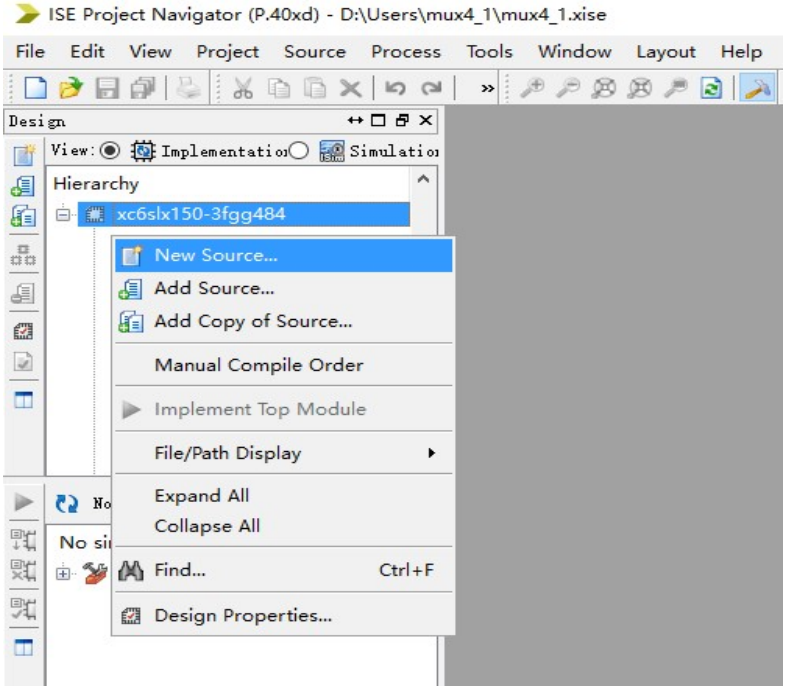
(4) 选择芯片型号，语言选择 Verilog，点击“Next”。



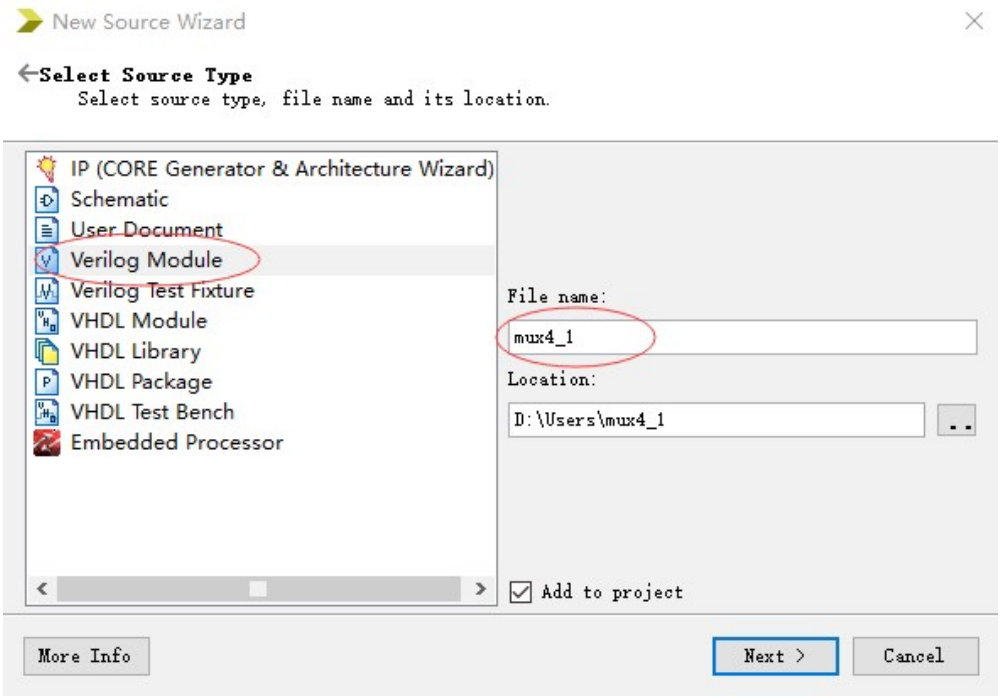
(5) 点击“Finish”。



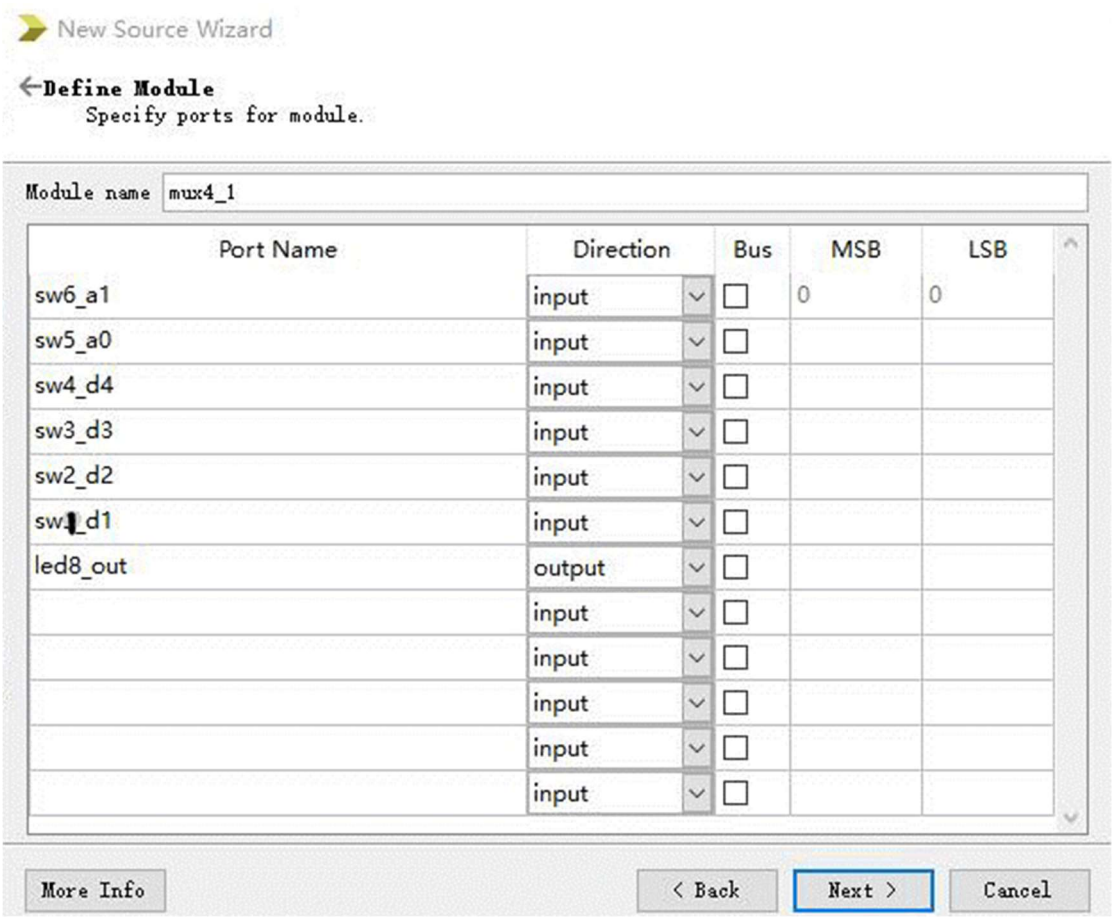
⑹ 右键单击“xc6slx150-3fgg484”，选择“NewSource…”。



⑺ 选择 Verilog Module，输入文件名“mux4_1”，点击“Next”。



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“Next”再点“Finish”

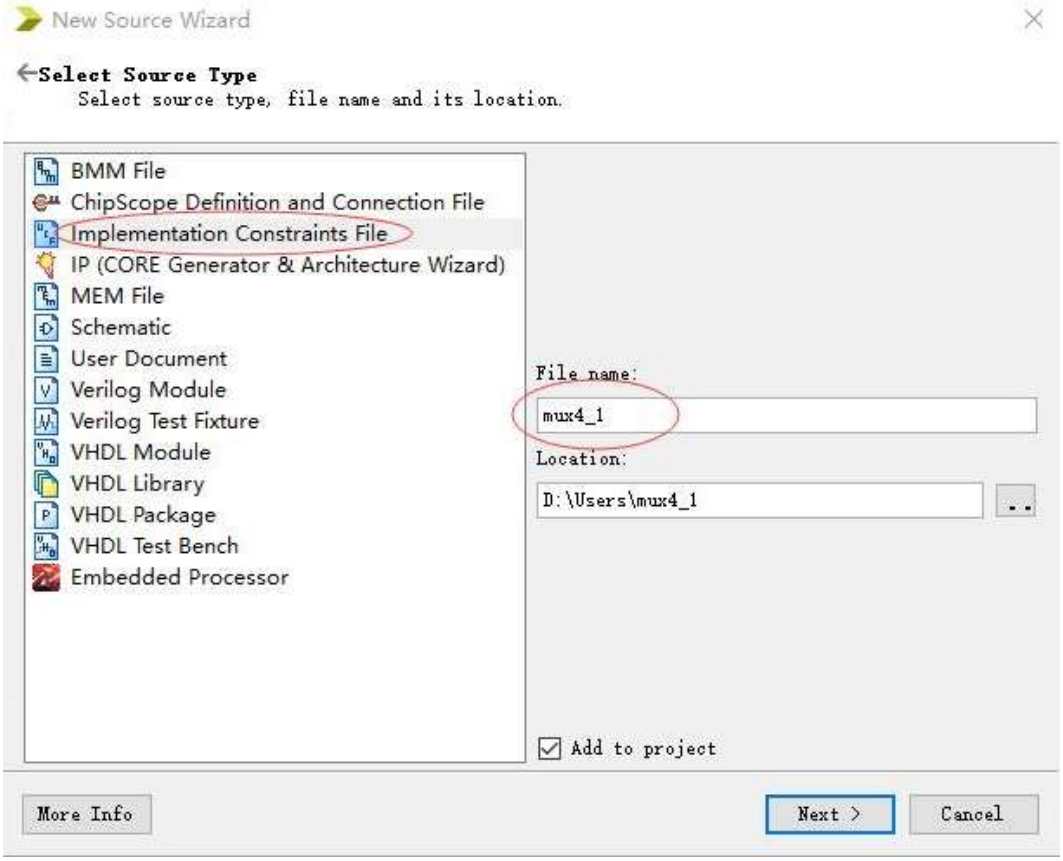


(9) 根据数据选择器的逻辑表达式，如下。在代码编辑区，用 assign 实现该逻辑。led8_out 变量赋值语句为我们自己添加的代码。其他为自动生成的代码。

$$Y_1 = D_0(A_1'A_0') + D_1(A_1'A_0) + D_2(A_1A_0') + D_3(A_1A_0)$$

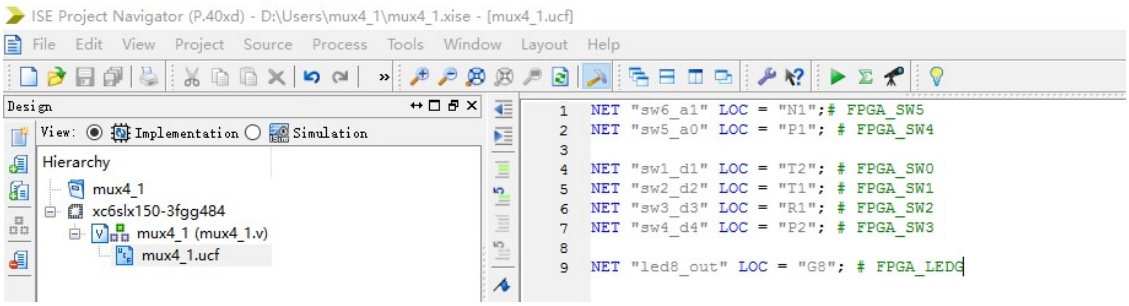
```
module mux4_1(
    input sw6_a1,
    input sw5_a0,
    input sw4_d4,
    input sw3_d3,
    input sw2_d2,
    input sw1_d1,
    output led8_out
    );
assign led8_out = sw1_d1&(!sw6_a1)&(!sw5_a0)|sw2_d2&(!sw6_a1)&( sw5_a0)|
                  sw3_d3&( sw6_a1)&(!sw5_a0)|sw4_d4&( sw6_a1)&( sw5_a0);

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。右键单击“xc6slx150-3fgg484”，选择“NewSource…”，选择“Implementation Constraints File”，输入约束文件名“mux4_1”，点击“Next”。

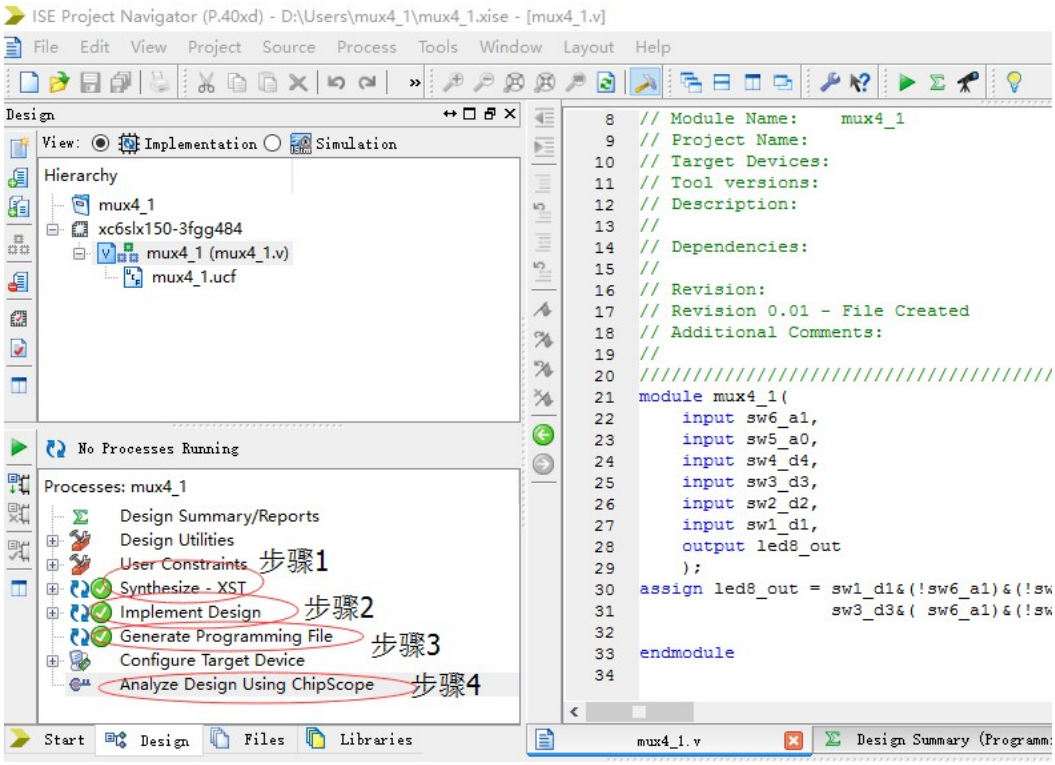


⑾ 打开“mux4_1.ucf”文件，结合板子原理图，进行对应引脚和模块接口的映射。

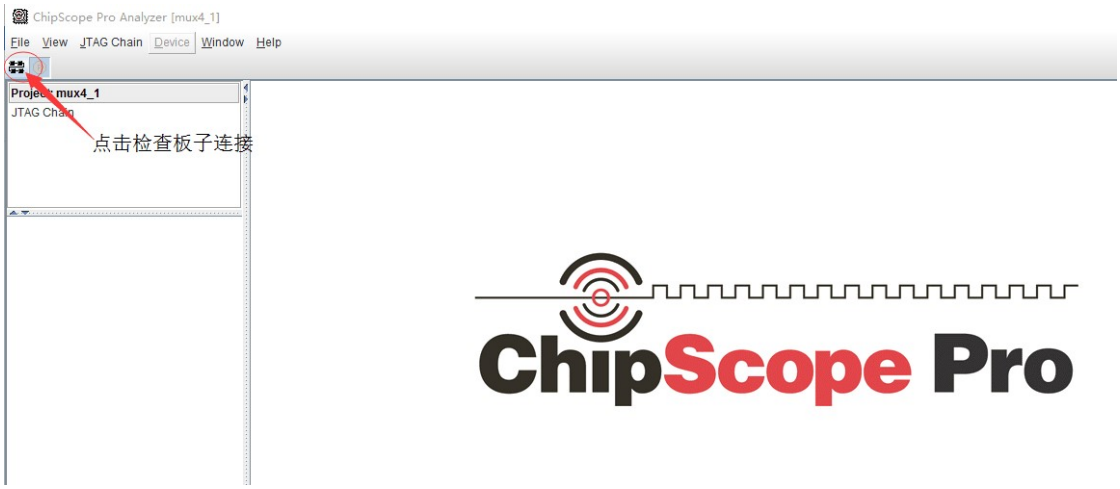
```
NET "sw6_a1" LOC = "N1";# FPGA_SW5
NET "sw5_a0" LOC = "P1"; # FPGA_SW4

NET "sw1_d1" LOC = "T2"; # FPGA_SW0
NET "sw2_d2" LOC = "T1"; # FPGA_SW1
NET "sw3_d3" LOC = "R1"; # FPGA_SW2
NET "sw4_d4" LOC = "P2"; # FPGA_SW3

NET "led8_out" LOC = "G8"; # FPGA_LEDG
```



⑿ 分别按照下图步骤 1、步骤 2、步骤 3，进行综合、布局布线、生成“.bit”文件。若不成功，则检查错误。如果成功，则双击，进行步骤 4。

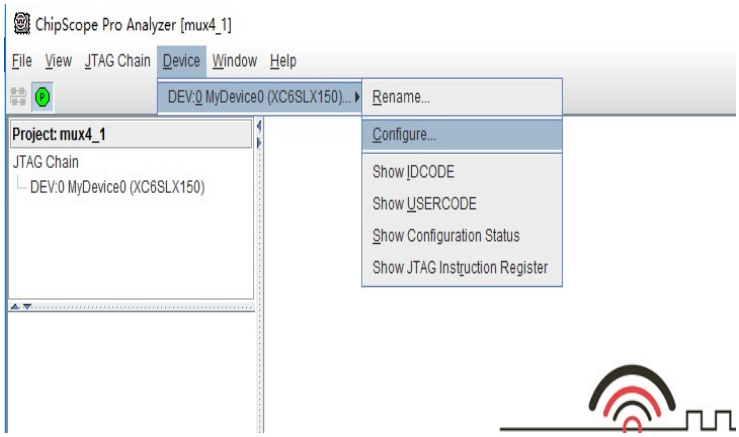


⒀ 双击步骤 4 后，会打开如下界面，将开发板通电，连接好 JTAG 下载线。点击下图红色箭头所指的图标，打开线缆，检查连接性。

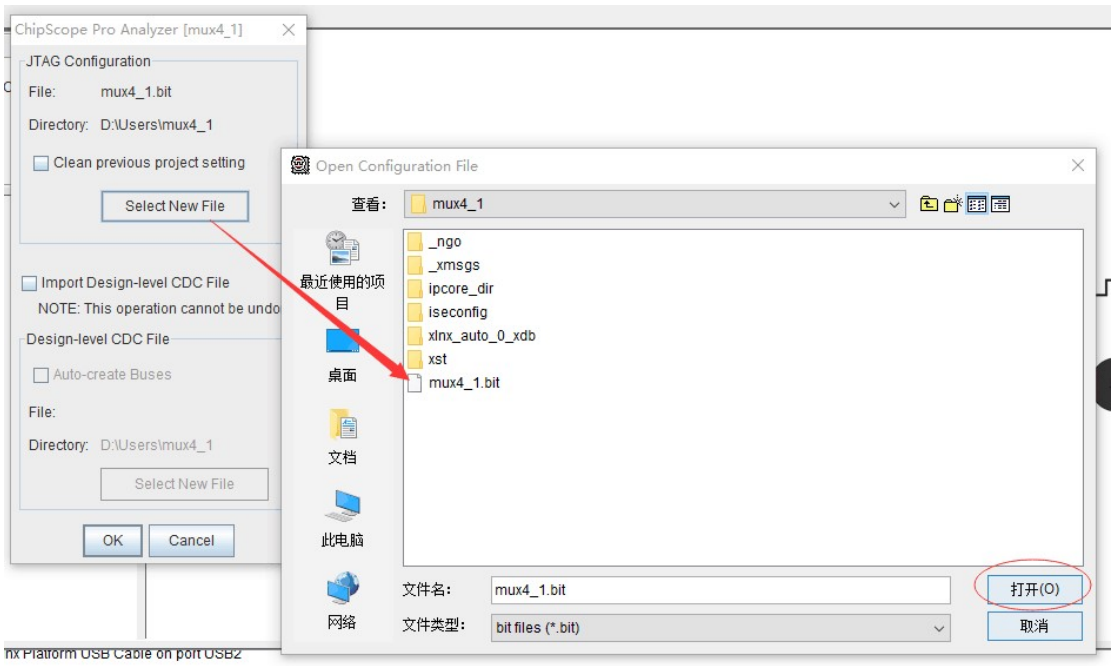


⑭ 如果连接正常，会出现如下界面。点击“OK”。

⑮ 点击“Device”→“DEV:0 My Device0(XC6SLX150)” →“Configure”



⑯ 出现如下界面，点击“Select New File”，选择综合生成的 mux4_1.bit 文件。



⑰ 待右下角进度条为 100%。程序即下载成功。

⑱ 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。

1.3.2 设计 3-8 译码器

3-8 译码器的真值表如下：

输 入			输 出							
A_2	A_1	A_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

逻辑表达式为：

$$Y_0 = m_0 = A_2'A_1'A_0' \quad Y_1 = m_1 = A_2'A_1'A_0 \quad Y_2 = m_2 = A_2'A_1A_0' \quad Y_3 = m_3 = A_2'A_1A_0 ?$$

$$Y_4 = m_4 = A_2A_1'A_0' \quad Y_5 = m_5 = A_2A_1'A_0 \quad Y_6 = m_6 = A_2A_1A_0' \quad Y_7 = m_7 = A_2A_1A_0$$

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（LED3,LED2,LED1,LED13,LED12,LED11,LED10,LED9）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 LED3_Y0,
    output LED2_Y1,
    output LED1_Y2,
    output LED13_Y3,
    output LED12_Y4,
    output LED11_Y5,
    output LED10_Y6,
    output LED9_Y7
    );
    assign LED3_Y0  = !Sw3_A2&!Sw2_A1&!Sw1_A0;
    assign LED2_Y1  = !Sw3_A2&!Sw2_A1& Sw1_A0;
    assign LED1_Y2  = !Sw3_A2& Sw2_A1&!Sw1_A0;
    assign LED13_Y3 = !Sw3_A2& Sw2_A1& Sw1_A0;
    assign LED12_Y4 =  Sw3_A2&!Sw2_A1&!Sw1_A0;
    assign LED11_Y5 =  Sw3_A2&!Sw2_A1& Sw1_A0;
    assign LED10_Y6 =  Sw3_A2& Sw2_A1&!Sw1_A0;
    assign LED9_Y7  =  Sw3_A2& Sw2_A1& Sw1_A0;
endmodule
```

相应的引脚约束条件如下：

```
NET "Sw3_A2"   LOC = "R1";# FPGA_SW2
NET "Sw2_A1"   LOC = "T1"; # FPGA_SW1
NET "Sw1_A0"   LOC = "T2"; # FPGA_SW0

NET "LED3_Y0"  LOC = "A8"; # FPGA_LED3
NET "LED2_Y1"  LOC = "B8"; # FPGA_LED2
NET "LED1_Y2"  LOC = "F9"; # FPGA_LED1
NET "LED13_Y3" LOC = "C10"; # FPGA_LED12
NET "LED12_Y4" LOC = "D10"; # FPGA_LED11
NET "LED11_Y5" LOC = "D11"; # FPGA_LED10
NET "LED10_Y6" LOC = "H10"; # FPGA_LED9
NET "LED9_Y7"  LOC = "G9"; # FPGA_LED8
```

附:

实验 1 结果:

U9 单元开关状态:

1	sw6	sw5	sw1	LED8 点亮
	下	下	上	
2	sw6	sw5	sw2	LED8 点亮
	下	上	上	
3	sw6	sw5	sw3	LED8 点亮
	上	下	上	
4	sw6	sw5	sw4	LED8 点亮
	上	上	上	

实验 2 结果:

U9 单元的开关状态:

	sw3	sw2	sw1	灯
1	下	下	下	右侧第 1 个灯灭
2	下	下	上	右侧第 2 个灯灭
3	下	上	下	右侧第 3 个灯灭
4	下	上	上	右侧第 4 个灯灭
5	上	下	下	右侧第 5 个灯灭
6	上	下	上	右侧第 6 个灯灭
7	上	上	下	右侧第 7 个灯灭
8	上	上	上	右侧第 8 个灯灭

实验二 组合逻辑电路实验

2.1 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。

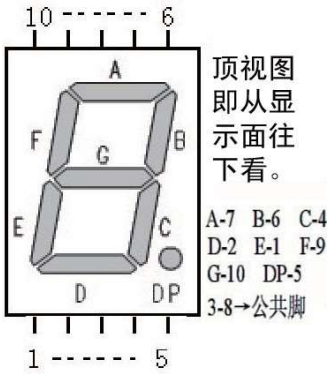
2.2实验内容

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。

2.3实验步骤

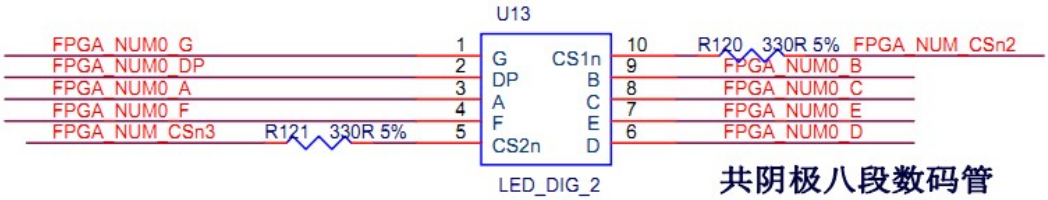
2.3.1 了解数码管原理。

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。



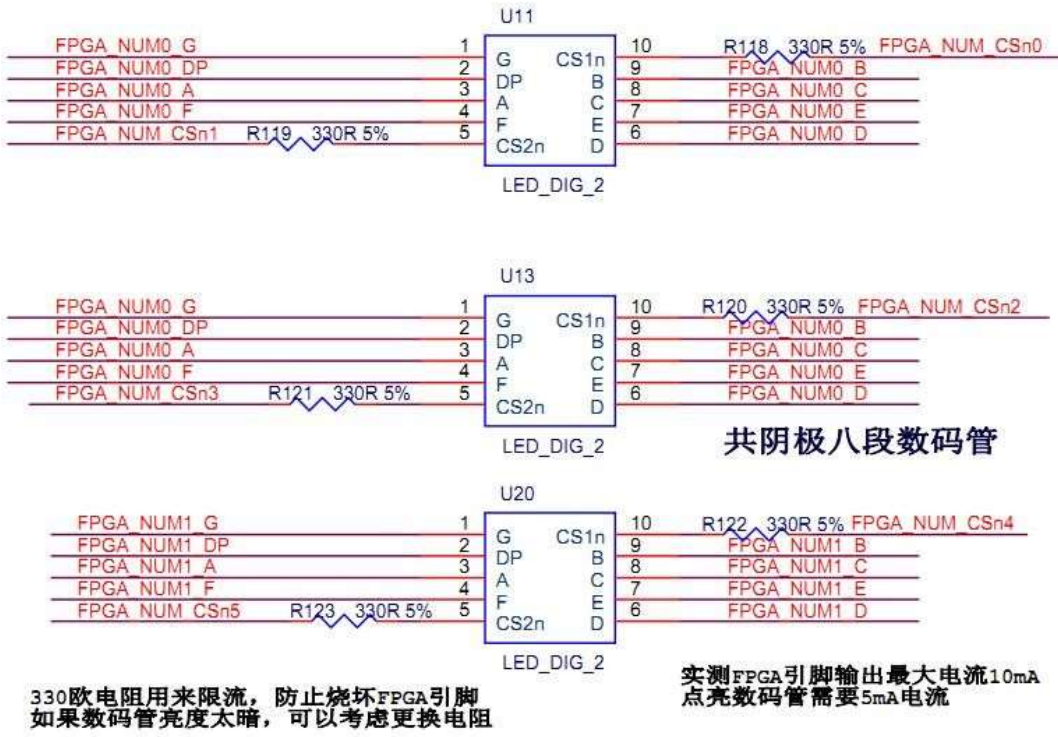
对于双位数码管，如下图所示：

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。

对于本开发板，U11 和 U13 使用相同的数据信号线，由 FPGA_NUM_CSn0、FPGA_NUM_CSn1、FPGA_NUM_CSn2、FPGA_NUM_CSn3 选择显示哪一位。U20 使用独立的数据信号线，由 FPGA_NUM_CSn4、FPGA_NUM_CSn5 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

短得到时间内，依次循环显示。

例如，对于 U11 和 U13 中的 4 位数码管，可以用如下逻辑控制轮流显示：

```
num0_scan_select <= 4'b0111;
num0_scan_select <= 4'b1011;
num0_scan_select <= 4'b1101;
num0_scan_select <= 4'b1110;
```

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
```

2.3.2 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)=$(5)^{10}$。

编码种类 / 十进制数	8421码（BCD代码）	余3码	2421码	5211码	余3循环码
0	0000	0011	0000	0000	0010
1	0001	0100	0001	0001	0110
2	0010	0101	0010	0100	0111
3	0011	0110	0011	0101	0101
4	0100	0111	0100	0111	0100
5	0101	1000	1011	1000	1100
6	0110	1001	1100	1001	1101
7	0111	1010	1101	1100	1111
8	1000	1011	1110	1101	1110
9	1001	1100	1111	1111	1010
权	8421		2421	5211	

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：

```
//U9
NET "bcd_num<3>" LOC = "P2";//FPGA_SW3
NET "bcd_num<2>" LOC = "R1";//FPGA_SW2
NET "bcd_num<1>" LOC = "T1";//FPGA_SW1
NET "bcd_num<0>" LOC = "T2";//FPGA_SW0
//U20 选择 FPGA_NUM1 7 段数码管的扫描位
NET "num1_scan_select<1>" LOC = "F5";//FPGA_NUM_CSn4
NET "num1_scan_select<0>" LOC = "C4";//FPGA_NUM_CSn5

NET "num1_seg7<7>" LOC = "E3"; # FPGA_NUM1_DP
NET "num1_seg7<6>" LOC = "D3"; # FPGA_NUM1_A
NET "num1_seg7<5>" LOC = "E4"; # FPGA_NUM1_B
NET "num1_seg7<4>" LOC = "E5"; # FPGA_NUM1_C
NET "num1_seg7<3>" LOC = "D5"; # FPGA_NUM1_D
NET "num1_seg7<2>" LOC = "E6"; # FPGA_NUM1_E
NET "num1_seg7<1>" LOC = "C3"; # FPGA_NUM1_F
NET "num1_seg7<0>" LOC = "F3"; # FPGA_NUM1_G
```

附

实验结果：

	sw3	sw2	sw1	sw0	最右侧数码管显示数字
1	下	下	下	下	0
2	下	下	下	上	1
3	下	下	上	下	2
4	下	下	上	上	3
5	下	上	下	下	4
6	下	上	下	上	5
7	下	上	上	下	6
8	下	上	上	上	7
9	上	下	下	下	8
10	上	下	下	上	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

3.1 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。

3.2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。

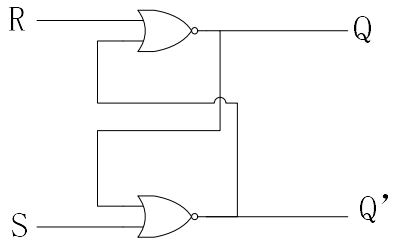
3.3实验要求

以拨码开关为输入，观察触发器的输出（LED 灯为输出）。

3.4实验步骤

3.4.1 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：

```
NET "sw2_R" LOC = "T1";#FPGA_SW1
NET "sw1_S" LOC = "T2";#FPGA_SW0
NET "led8_Q"LOC = "G8";#FPGA_LEDG
```

3.4.2 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。

(1) RS 触发器

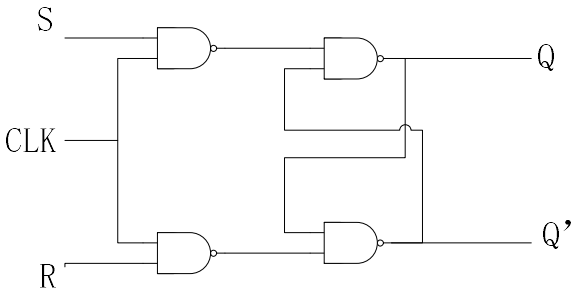
凡是在时钟信号作用下，逻辑功能符号下表所规定的逻辑功能者，无论触发方式如何，均称为 SR 触发器。

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：

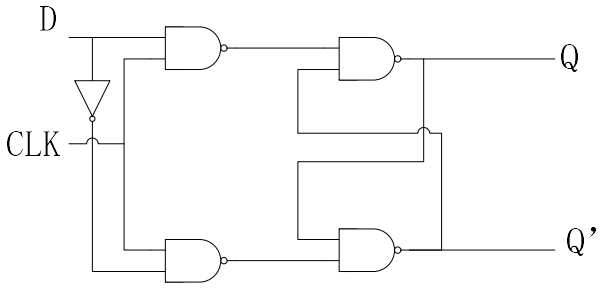
```
module level_rs_latch(//电平触发的 RS 触发器
    input sw3_CLK,
    input sw2_R,
    input sw1_S,
    output led8_Q
    );
     wire s1,s2;
     assign s1 = !(sw3_CLK & sw1_S);
    assign s2 = !(sw3_CLK & sw2_R);
     nand_rs_latch nand_rs_latch0(.s1(s1),.s2(s2),.Q(led8_Q));

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：

```
NET "sw3_CLK" LOC = "R1"; # FPGA_SW2
NET "sw2_R" LOC = "T1"; # FPGA_SW1
NET "sw1_S" LOC = "T2"; # FPGA_SW0

NET "led8_Q" LOC = "G8"; # FPGA_LEDG
```

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：

$$Q^* = D$$

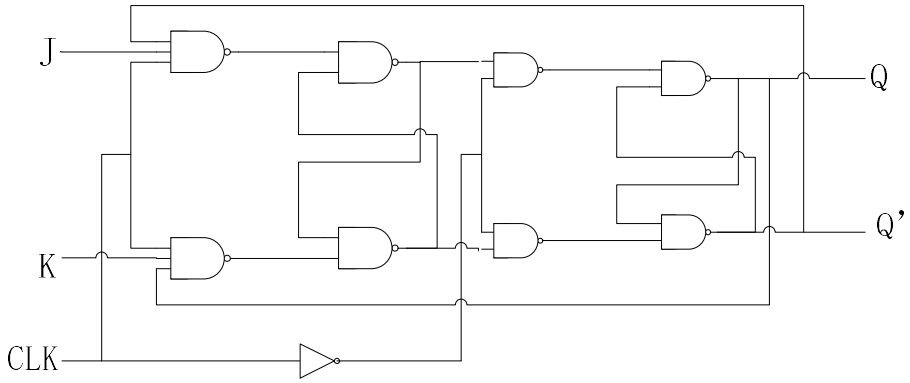
其特性表如下：

D	Q	Q^*
0	0	0
0	1	0
1	0	1
1	1	1

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：

CLK	J	K	Q	Q^*	说明
×	×	×	×	Q	保持原态
↓	0	0	0	0	储存
↓	0	0	1	1	
↓	0	1	0	0	置0(复位)
↓	0	1	1	0	
↓	1	0	0	1	置1(置位)
↓	1	0	1	1	
↓	1	1	0	1	Q′ 计数
↓	1	1	1	0	

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：

```
NET "sw3_CLK" LOC = "R1"; # FPGA_SW2
NET "sw2_J" LOC = "T1"; # FPGA_SW1
NET "sw1_K" LOC = "T2"; # FPGA_SW0

NET "led8_Q" LOC = "G8"; # FPGA_LEDG
```

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。

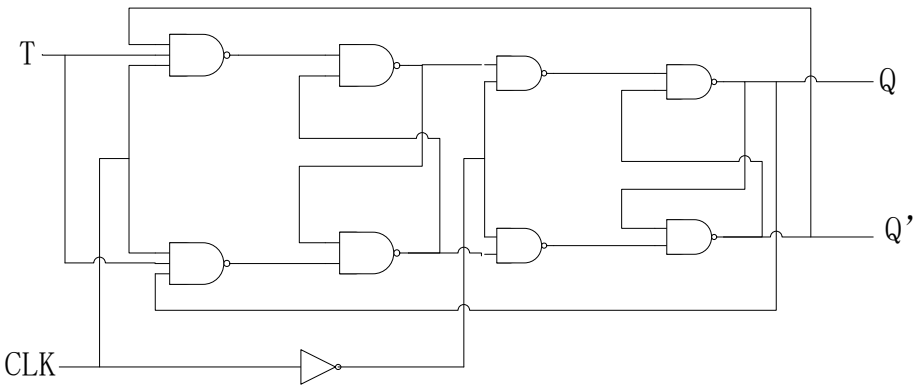
其特性方程为：

$$Q^* = TQ' + T'Q$$

其特性表如下：

T	Q	Q^*
0	0	0
0	1	1
1	0	1
1	1	0

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：

```
NET "clk"       LOC = "A11";#FPGA_CLK_IN2, 33MHz
NET "clk" CLOCK_DEDICATED_ROUTE=FALSE;

NET "sw1_t"     LOC = "T2"; # FPGA_SW0
#NET "sw3_reset" LOC = "T1"; # FPGA_SW1
NET "led8_Q"    LOC = "G8"; # FPGA_LEDG
```

附：

RS 锁存器实验结果：

当 U9 单元的 sw2 sw1 初始输入为“上下”，则 led8 灭。

当 U9 单元的 sw2 sw1 变为“下下”，则 led8 保持不变，仍灭。

当 U9 单元的 sw2 sw1 变为“下上”，则 led8 点亮。

当 U9 单元的 sw2 sw1 变为“下下”，则 led8 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“上”，sw2 和 sw1 的操作和 RS 锁存器一样。

当 sw3 为“下”，sw2 和 sw1 的操作均不会引起输出的变化。

当 sw2 sw1 为“上下”，将 sw3 置“上”，LED 灯灭。

然后，将 sw3 置“下”，LED 灯仍灭。

接着，将 sw2 sw1 置为“下上”，此时 LED 灯仍灭。

然后，将 sw3 置“上”，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“上下”时，sw3 往上拨动，再往下拨动后，LED8 点亮。

改变 sw2 sw1 为“下上”时，sw3 往上，再往下拨动后，LED8 熄灭。

改变 sw2 sw1 为“下下”时，sw3 往上，再往下拨动后，LED8 保持不变。

改变 sw2 sw1 为“上上”时，sw3 往上，再往下拨动后，LED8 点亮。

保持 sw2 sw1 为“上上”，sw3 往上，再往下拨动后，LED8 熄灭。

T 触发器实验结果：

对于 U9 单元，

当 sw1 为“上”时，LED8 以 2Hz 进行闪烁。

当 sw1 为“下”时，LED8 状态保持不变。

实验四 时序逻辑电路实验

4.1 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。

4.2实验内容

设计一个带进位输出端的 13 进制计数器。

4.3实验步骤

4.3.1 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。

4.3.2 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

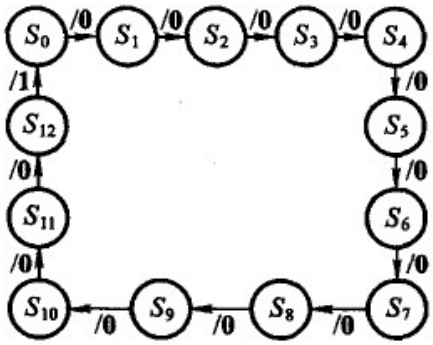
4、选择触发器类型，求出电路的状态方程、驱动方程和输出方程

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：

状态变化顺序	状态编码				进位输出	等效
	Q_3	Q_2	Q_1	Q_0	C	十进制数
S_0	0	0	0	0	0	0
S_1	0	0	0	1	0	1
S_2	0	0	1	0	0	2
S_3	0	0	1	1	0	3
S_4	0	1	0	0	0	4
S_5	0	1	0	1	0	5
S_6	0	1	1	0	0	6
S_7	0	1	1	1	0	7
S_8	1	0	0	0	0	8
S_9	1	0	0	1	0	9
S_{10}	1	0	1	0	0	10
S_{11}	1	0	1	1	0	11
S_{12}	1	1	0	0	1	12
S_0	0	0	0	0	0	0

状态方程：

$$\begin{cases} Q_3^* = Q_3Q_2' + Q_2Q_1Q_0 \\ Q_2^* = Q_3'Q_2Q_1' + Q_3'Q_2Q_0' + Q_2'Q_1Q_0 \\ Q_1^* = Q_1'Q_0 + Q_1Q_0' \\ Q_0^* = Q_3'Q_0' + Q_2'Q_0' \end{cases}$$

输出方程为：

$$C = Q_3Q_2$$

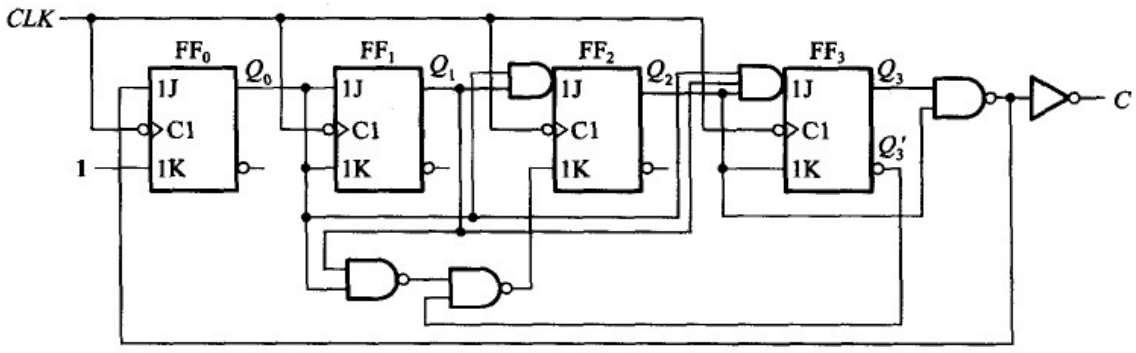
如果使用 JK 触发器组成这个电路，则将状态方程修改为如下形式：

$$\begin{cases} Q_3^* = Q_3Q_2' + Q_2Q_1Q_0(Q_3 + Q_3') = (Q_2Q_1Q_0)Q_3' + Q_2'Q_3 \\ Q_2^* = (Q_0Q_1)Q_2' + (Q_3'(Q_1Q_0)')Q_2 \\ Q_1^* = Q_0Q_1' + Q_0'Q_1 \\ Q_0^* = (Q_3' + Q_2')Q_0' + 1' \cdot Q_0 = (Q_3Q_2)'\,Q_0' + 1'Q_0 \end{cases}$$

各个驱动器的驱动方程为：

$$\begin{cases} J_3 = Q_2Q_1Q_0, & K_3 = Q_2 \\ J_2 = Q_1Q_0, & K_2 = (Q_3'(Q_1Q_0)')' \\ J_1 = Q_0, & K_1 = Q_0 \\ J_0 = (Q_3Q_2)', & K_0 = 1 \end{cases}$$

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            sw6_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(sw6_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：

```
NET "button_clk"  LOC = "U3";#FPGA_RESET_IN
NET "button_clk" CLOCK_DEDICATED_ROUTE = FALSE;
NET "sw6_reset" LOC = "N1";# FPGA_SW5

NET "led8_C"    LOC = "G8";# FPGA_LEDG
//U20 选择 FPGA_NUM1  7 段数码管的扫描位
NET "num1_scan_select<1>" LOC = "F5";//FPGA_NUM_CSn4
NET "num1_scan_select<0>" LOC = "C4";//FPGA_NUM_CSn5

NET "num1_seg7<7>" LOC = "E3"; # FPGA_NUM1_DP
NET "num1_seg7<6>" LOC = "D3"; # FPGA_NUM1_A
NET "num1_seg7<5>" LOC = "E4"; # FPGA_NUM1_B
NET "num1_seg7<4>" LOC = "E5"; # FPGA_NUM1_C
NET "num1_seg7<3>" LOC = "D5"; # FPGA_NUM1_D
NET "num1_seg7<2>" LOC = "E6"; # FPGA_NUM1_E
NET "num1_seg7<1>" LOC = "C3"; # FPGA_NUM1_F
NET "num1_seg7<0>" LOC = "F3"; # FPGA_NUM1_G
```

附：

实验结果：

当把 U9 单元 sw6 置为“上”时，按一下 Reset button 键，电路进入复位状态。

当将 U9 单元 sw6 置为“下”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 LED8 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

5.1 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。

5.2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。

5.3实验步骤

5.3.1 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];

5.3.2 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：

```
NET "row1_clk"     LOC = "Y2";#FPGA_KEY_ROW1
NET "row1_clk" CLOCK_DEDICATED_ROUTE=FALSE;
NET "u9sw6_ram_en" LOC = "N1";# FPGA_SW5
NET "u9sw5_r_en"   LOC = "P1"; # FPGA_SW4
#4 个地址,U10 单元的{sw2,sw1}{8,7}
NET "u10_addr<1>"  LOC = "M1"; # FPGA_SW7
NET "u10_addr<0>"  LOC = "M2"; # FPGA_SW6
#4 位数据输入 U10 单元的{sw6,sw5,sw4,sw3}{12,11,10,9}
NET "data_in<3>"   LOC = "K2"; # FPGA_SW11
NET "data_in<2>"   LOC = "K1"; # FPGA_SW10
NET "data_in<1>"   LOC = "L1"; # FPGA_SW9
NET "data_in<0>"   LOC = "L3"; # FPGA_SW8
#最右边 4 个 led: {led13,led1,led2,led3}
NET "data_out_led<3>"  LOC = "C10"; # FPGA_LED12,led13
NET "data_out_led<2>"  LOC = "F9"; # FPGA_LED1,led1
NET "data_out_led<1>"  LOC = "B8"; # FPGA_LED2,led2
NET "data_out_led<0>"  LOC = "A8"; # FPGA_LED3,led3

NET "col1_4<3>" LOC = "T4";#FPGA_KEY_COL1
NET "col1_4<2>" LOC = "V3";#FPGA_KEY_COL2
NET "col1_4<1>" LOC = "U4";#FPGA_KEY_COL3
NET "col1_4<0>" LOC = "W3";#FPGA_KEY_COL4

//U20 选择 FPGA_NUM1  7 段数码管的扫描位
NET "num1_scan_select<1>" LOC = "F5";//FPGA_NUM_CSn4
NET "num1_scan_select<0>" LOC = "C4";//FPGA_NUM_CSn5

NET "num1_seg7<7>" LOC = "E3"; # FPGA_NUM1_DP
NET "num1_seg7<6>" LOC = "D3"; # FPGA_NUM1_A
NET "num1_seg7<5>" LOC = "E4"; # FPGA_NUM1_B
NET "num1_seg7<4>" LOC = "E5"; # FPGA_NUM1_C
NET "num1_seg7<3>" LOC = "D5"; # FPGA_NUM1_D
NET "num1_seg7<2>" LOC = "E6"; # FPGA_NUM1_E
NET "num1_seg7<1>" LOC = "C3"; # FPGA_NUM1_F
NET "num1_seg7<0>" LOC = "F3"; # FPGA_NUM1_G
```

附：

实验结果：

将 U9 单元 sw6 sw5 置为“上下”，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“上上”，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）

6.1 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。

6.2实验内容

实现数字时钟。

6.3实验要求

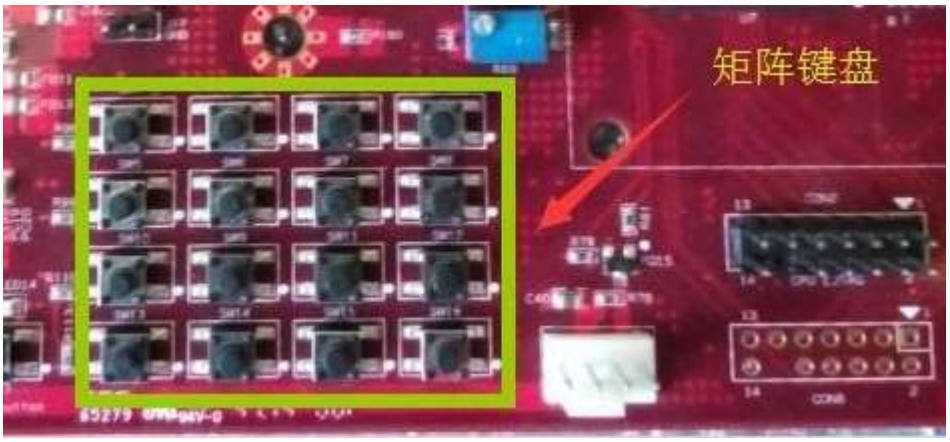
（1）准确计时，以数字形式显示时、分、秒。

（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

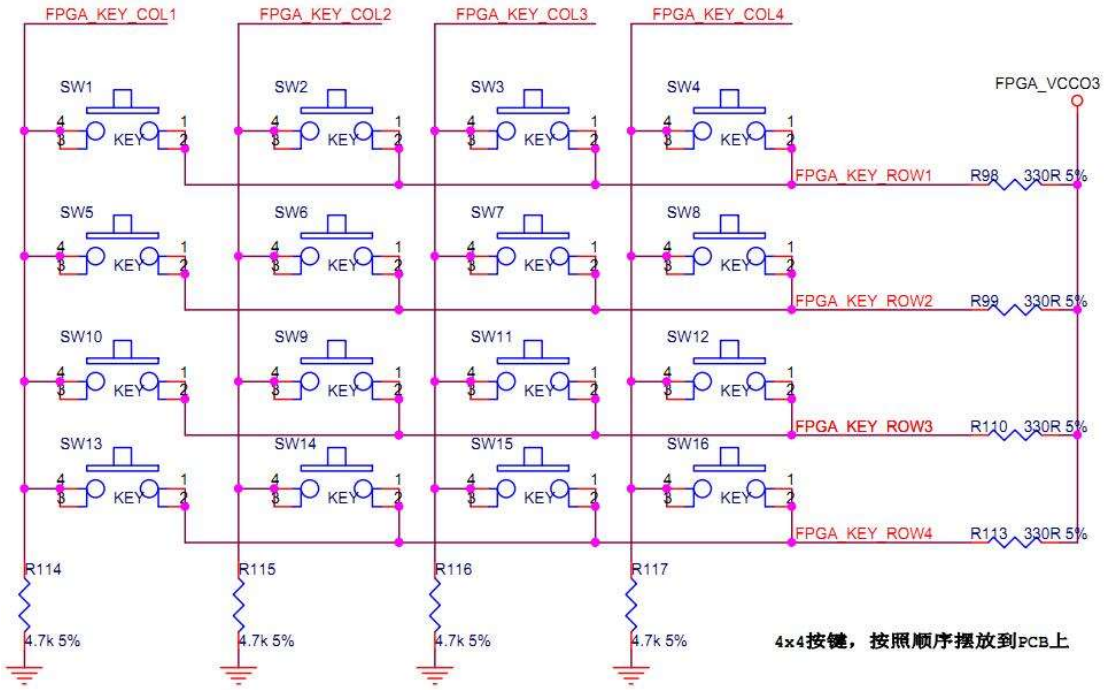
说明：数字钟是能够准确的显示时、分、秒时间，显示时间有误差可以校时。本节综合实验需要用到数码管显示时间，用到按键调整时间，用到时钟进行精确计时。数码管、时钟的使用方法均可参考前面实验章节。

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。

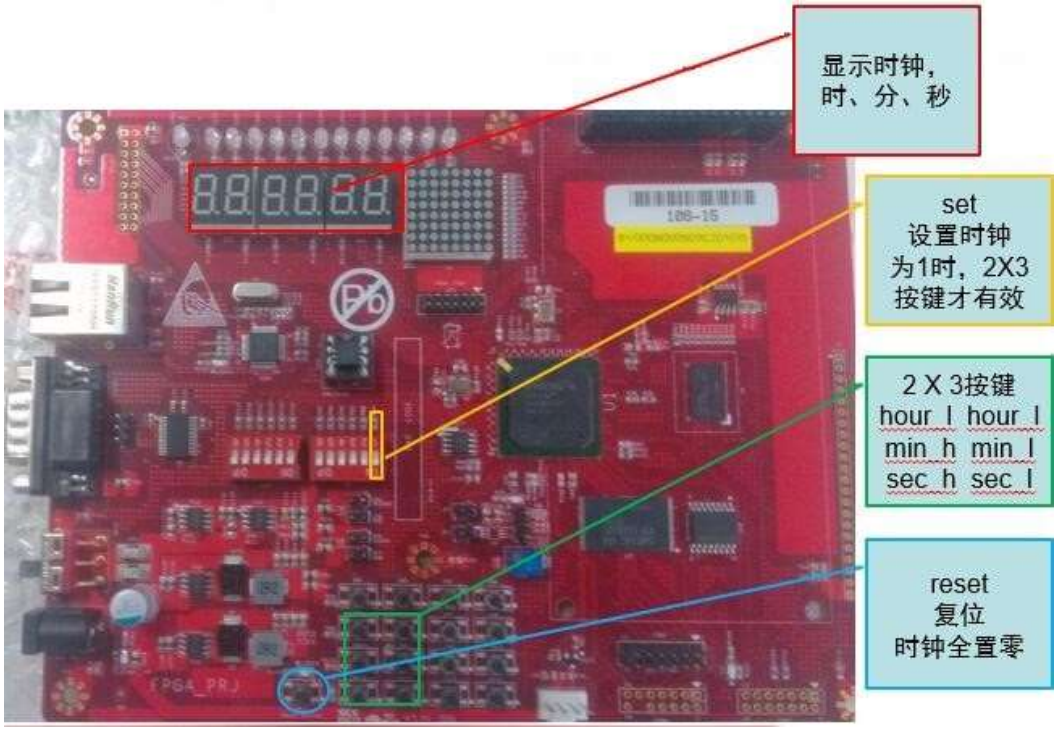
基于以上原理，可设定一个机制扫描键盘以确定按下了哪些键。

6.4 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。



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。