桂林电子科技大学

电子信息学院

FPGA 实验指导书

指导老师：叶俊明

目录

实验一：ISE 软件的使用

一、实验目的

1、掌握 ISE 软件的基本使用

2、掌握内置门原语的使用

二、ISE 工程建立

1、双击桌面 ISE 图标，启动 ISE 软件如下图 1-1：

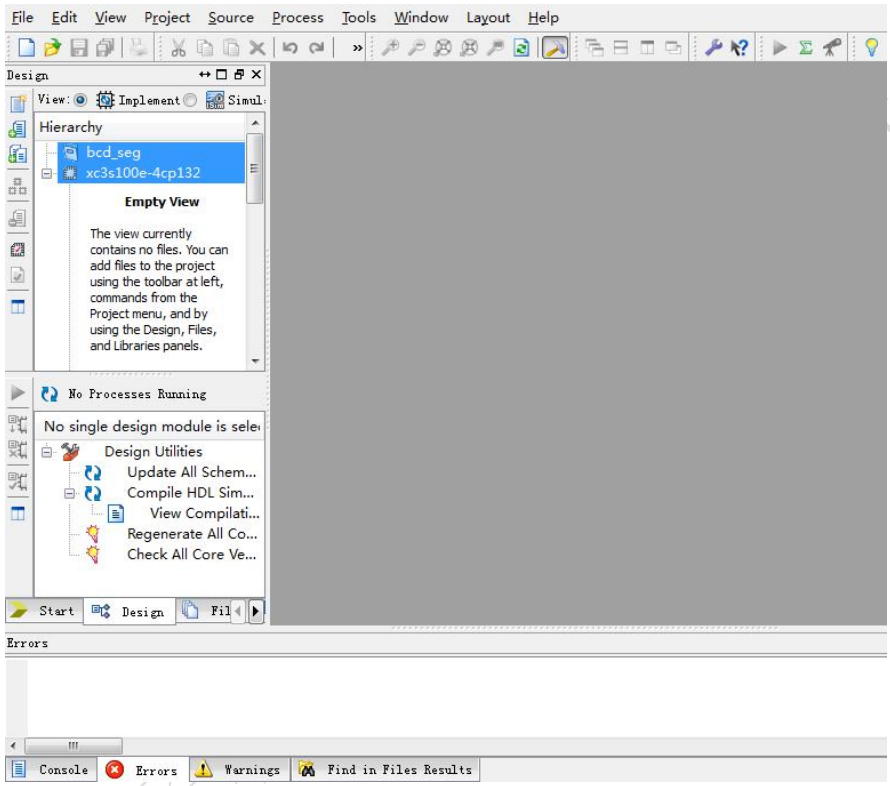


图 1-1

2、 点击菜单栏的文件（File）菜单中的命令（New Project），新建一个工程文件，并输入工程名、存储位置以及文件类型如下图 1-2，然后点击 Next。

注：

（1）工程名的首字符必须为字母或下划线。**存储位置必须为全英文的文件夹**。

（2）菜单中的 Discription 是对工程的注释说明，可以不作说明。

（3）文件类型可为硬件描述语言 HDL、图形描述 Schematic、EDIF（Synplify.Pro 软件生成的终端设计文件）、NGC/NGO（网表）。

（4）对话框下面有帮助信息（More Info），用户可以从中获得帮助。

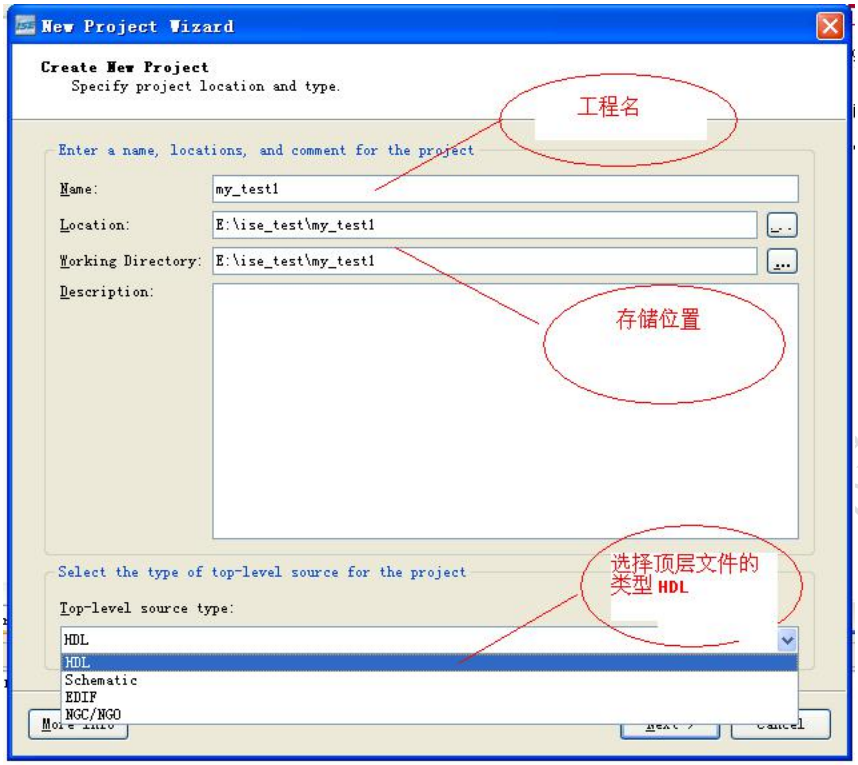


图 1-2

3、选择芯片类型、速度等级、综合工具、仿真工具等如图 1-3，点击 NEXT。

注：

（1） Family 以 Xilinx 的 **XC3S100E** 为例，封装形式为 CP132。

（2）仿真工具（Simulator）可以选择 ISE 自带的 Isim(VHDL/Verilog)，如果安装第三方仿真软件（如：Modelsim），可以选择仿真工具为 Modelsim-SE verilog 等。

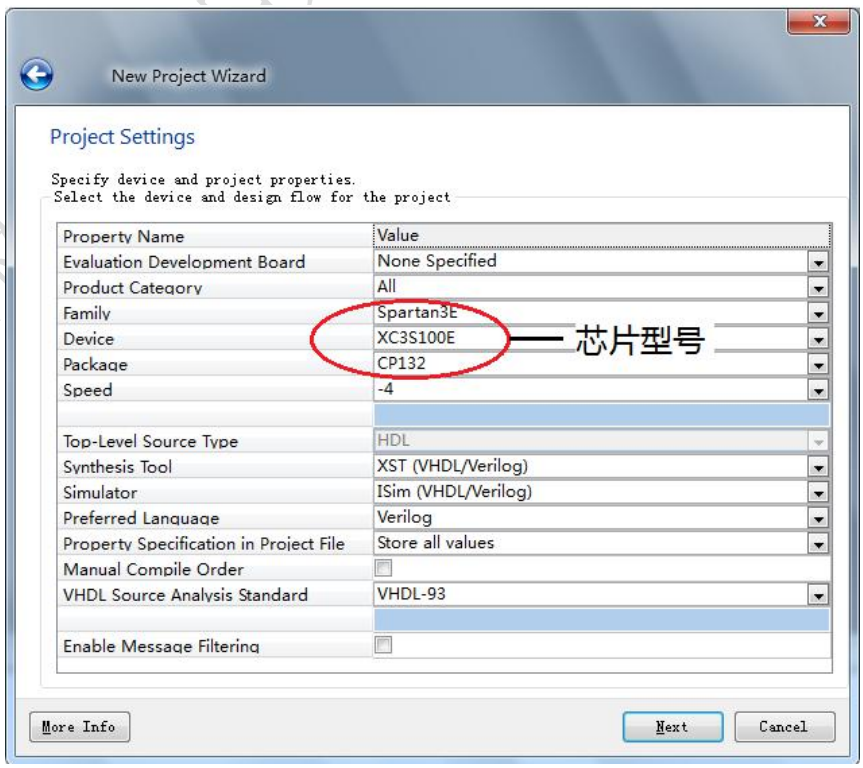


图 1-3

4、 出现下图 1-4 的工程摘要，点击 finish。

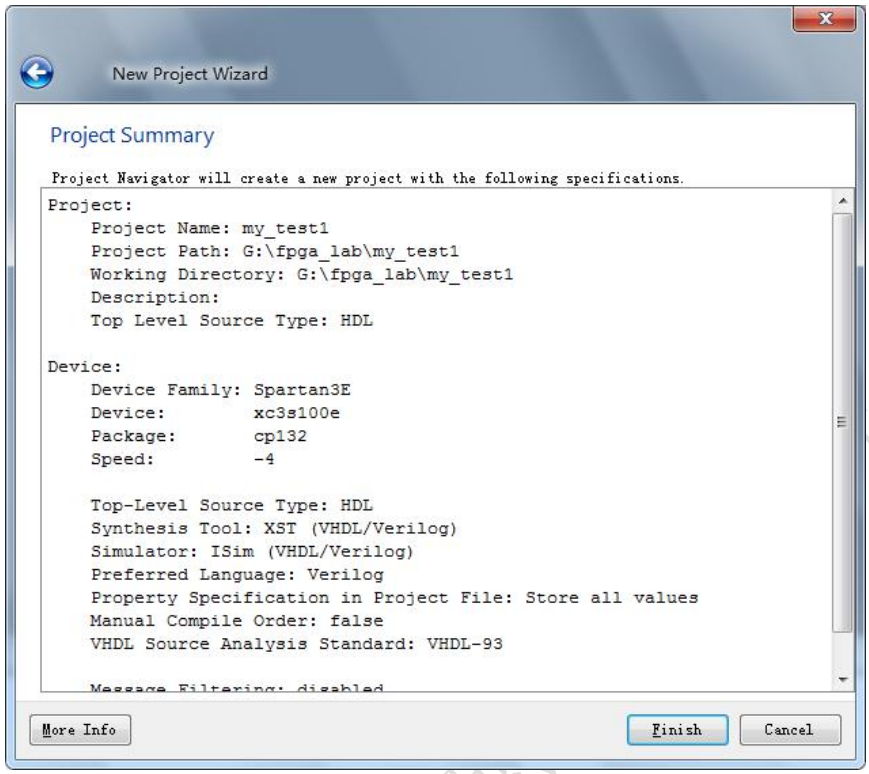


图 1-4

从 project 中选择添加资源文件 new source，并选择文件类型为 Verilog Module，输入文件名如下图 1-5。点击 NEXT 进行下一步操作。

注：图 1-5 中资源文件的类型有多种，各项意义如下说明：

（1） IP（CORE Generator&Architecture Wizard）：使用 ISE 的 IP Core 工具快速生成可靠的源代码，目前比较流行。

（2）Schematic ：原理图输入。

（3）User Document ：用户文档类型。

（4）Verilog Module ： Verilog 模块类型，编写 Verilog hdl 代码。

（5）Verilog Test Fixture ： Verilog 测试模块类型，用于编写 Verilog 测试代码。

（6）VHDL Module ： VHDL 模块类型，用于编写 VHDL 代码。

（7）VHDL Library ： VHDL 库类型，用于制作 VHDL 库。

（8）VHDL Package ： VHDL 包类型，用于制作 VHDL 包。

（9）VHDL Test Fixture ： VHDL 测试模块类型，用于编写 VHDL 测试代码。

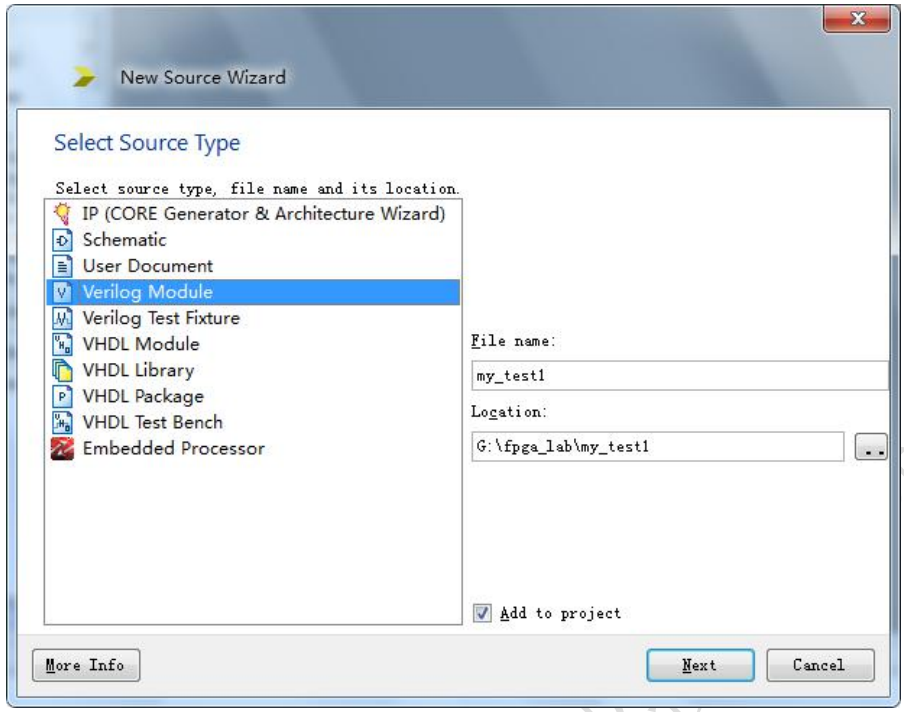


图 1-5

5、 此步骤要求用户设置管脚的方向及总线宽度类型，如图 1-6 所示。当然也可不填而选择在输入代码时人为输入。

注：

（1）以设计一个非门为例：a 为输入，方向为 input。
out 为输出，方向为 output。

（2）Bus 为端口总线选择，选择表示该端口为总线模式。

（3）MSB 和 LSB 为总线的最高位和最低位，以 8 位数据为例：最高（低）为可以为 0，也可以为 7。

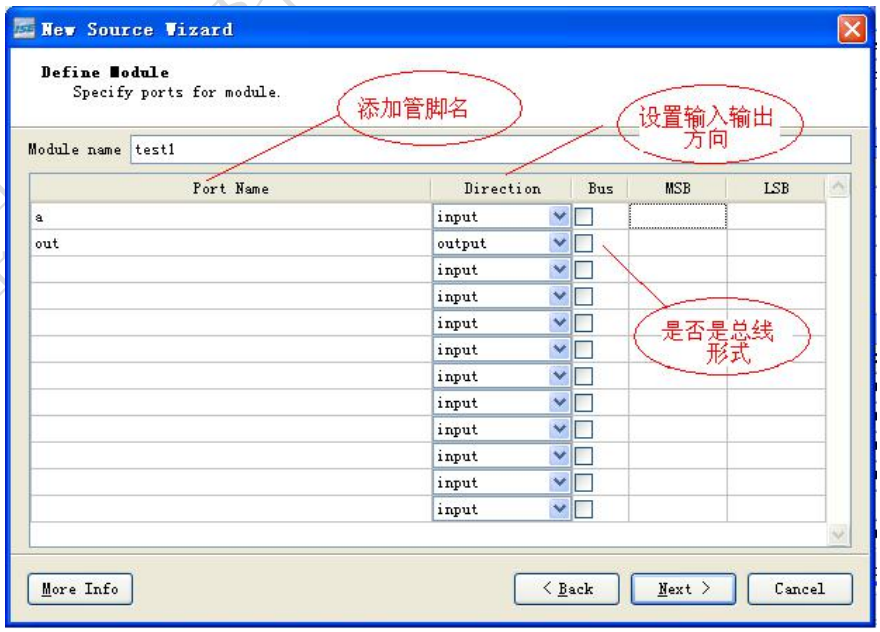


图 1-6

6、图 1-7 的 Summary 对话框总结出设计者新建资源文件的存储位置、类型，模块名、管脚等信息。如果用户想修改信息，可以返回修改（Back）。如果确定点击 Finish 完

成资源文件添加即可。

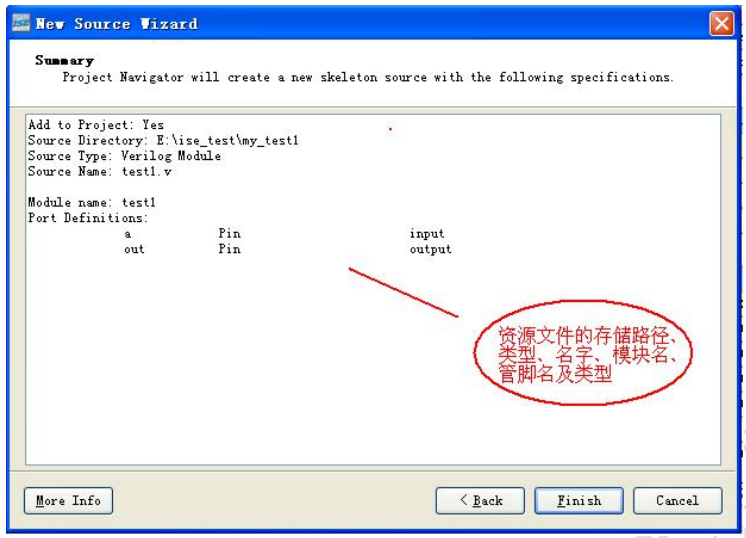


图 1-7

7、 在代码编辑区自动生成模块的基本框架（如：模块名、输入输出管脚名等）如图 1-8。

```
1 `timescale 1ns / 1ps
2 module lab1(a,out
3     );
4 input a;output out;
5  assign out=!a;
6 endmodule
```

图 1-8

本例为设计一个简单的非门，只需在模块中键入“**assign out=!a;**”保存即可。

10、点击 ISE 左边 Design 对话框中的 Implementation 的 xxx.v 的文件，在 Processes 中双击综合工具（Synthesize-XST）如图 1-9：

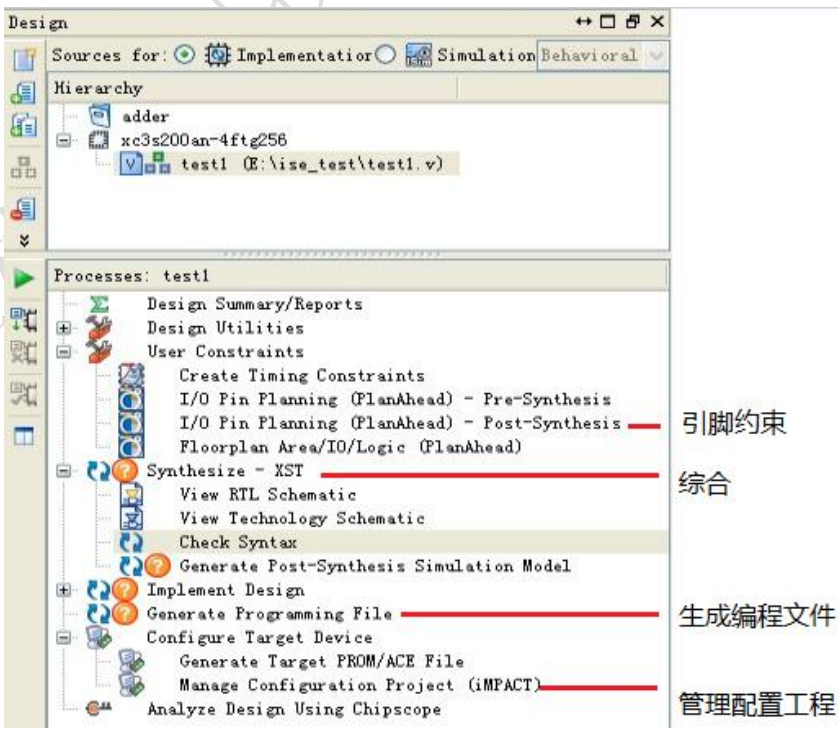


图 1-9

在软件下方 Console 对话框中显示 Process "Check Syntax" completed successfully（语法正确）。如有错误则双击 Console 对话框中粉红色的第一个 ERROR 找到语法错误之处加以修改，直至无错误为止，如图 1-10：

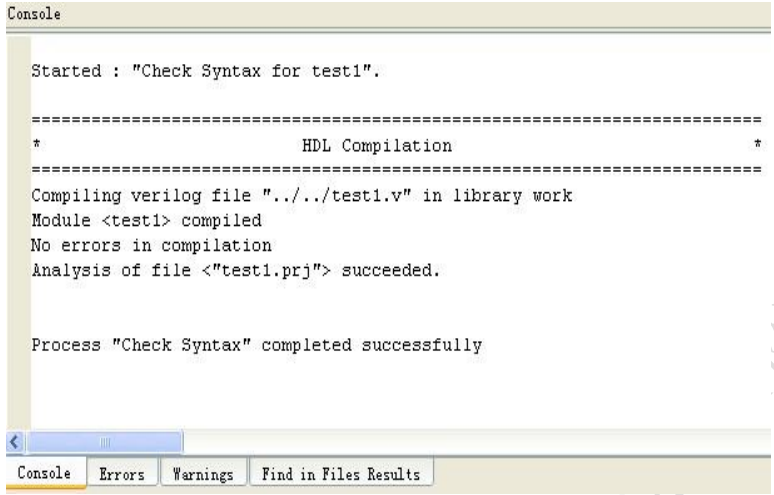


图 1-10

三、引脚约束

点击 Project 中的 NewSource 新建 Implementation Constraints File 文件，进行引脚约束如图 1-11：

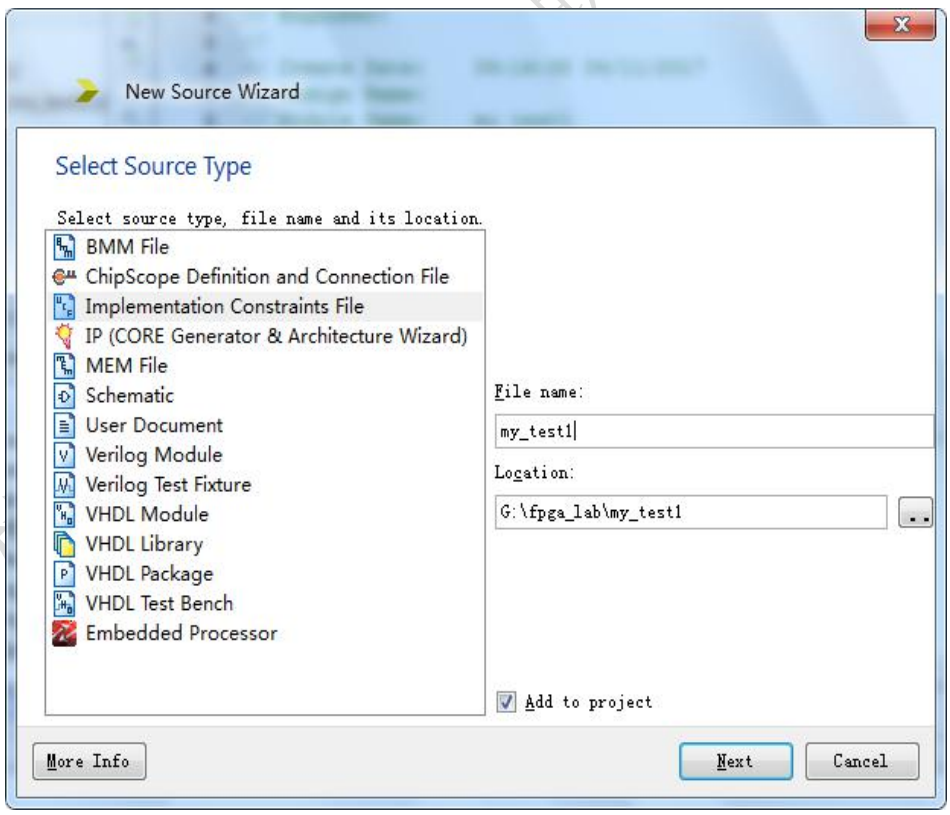


图 1-11

在 my_test1.ucf 文件中写入以下两条语句，用于约束输入输出引脚，也可以从表格 1 中选取：

NET "a" LOC = P11;
NET "out" LOC = M5;

Basys2 Spartan-3E pin definitions											
Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
C12	JD1	P11	SW0	N14	CC	B2	JA1	P8	MODE0	M7	GND
A13	JD2	M2	USB-DB1	N13	DP	C2	USB-WRITE	N7	MODE1	P5	GND
A12	NC	N2	USB-DB0	M13	AN2	C3	PS2D	N6	MODE2	P10	GND
B12	NC	M9	NC	M12	CG	D1	NC	N12	CCLK	P14	GND
B11	NC	N9	NC	L14	CA	D2	USB-WAIT	P13	DONE	A6	VDDO-3
C11	BTN1	M10	NC	L13	CF	L2	USB-DB4	A1	PROG	B10	VDDO-3
C6	JB1	N10	NC	F13	RED2	L1	USB-DB3	N8	DIN	E13	VDDO-3
B6	JB2	M11	LD1	F14	GRN0	M1	USB-DB2	N1	INIT	M14	VDDO-3
C5	JB3	N11	CD	D12	JD4	L3	SW1	P1	NC	P3	VDDO-3
B5	JA4	P12	CE	D13	RED1	E2	SW6	B3	GND	M8	VDDO-3
C4	NC	N3	SW7	C13	JD3	F3	SW5	A4	GND	E1	VDDO-3
B4	SW3	M6	UCLK	C14	RED0	F2	USB-ASTB	A8	GND	J2	VDDO-3
A3	JA2	P6	LD3	G12	BTN0	F1	USB-DSTB	C1	GND	A5	VDDO-2
A10	JC3	P7	LD2	K14	AN2	G1	LD7	C7	GND	E12	VDDO-2
C9	JC4	M4	BTN2	J12	AN1	G3	SW4	C10	GND	K1	VDDO-2
B9	JC2	N4	LD5	J13	BLU2	H1	USB-DB6	E3	GND	P9	VDDO-2
A9	JC1	M5	LD0	J14	HSYNC	H2	USB-DB5	E14	GND	A11	VDDO-1
B8	MCLK	N5	LD4	H13	BLU1	H3	USB-DB7	G2	GND	D3	VDDO-1
C8	RCCLK	G14	GRN2	H12	CB	B14	TMS	H14	GND	D14	VDDO-1
A7	BTN3	G13	GRN1	J3	JA3	B13	TCK-FPGA	J1	GND	K2	VDDO-1
B7	JB4	F12	AN0	K3	SW2	A2	TDO-USB	K12	GND	L12	VDDO-1
P4	LD6	K13	VSYNC	B1	PS2C	A14	TDO-S3	M3	GND	P2	VDDO-1

表 1

四、程序下载

1、生成下载文件后连接好硬件，并双击图 1-12 中的 Generate Programming File 生成 XXX.bit 文件。

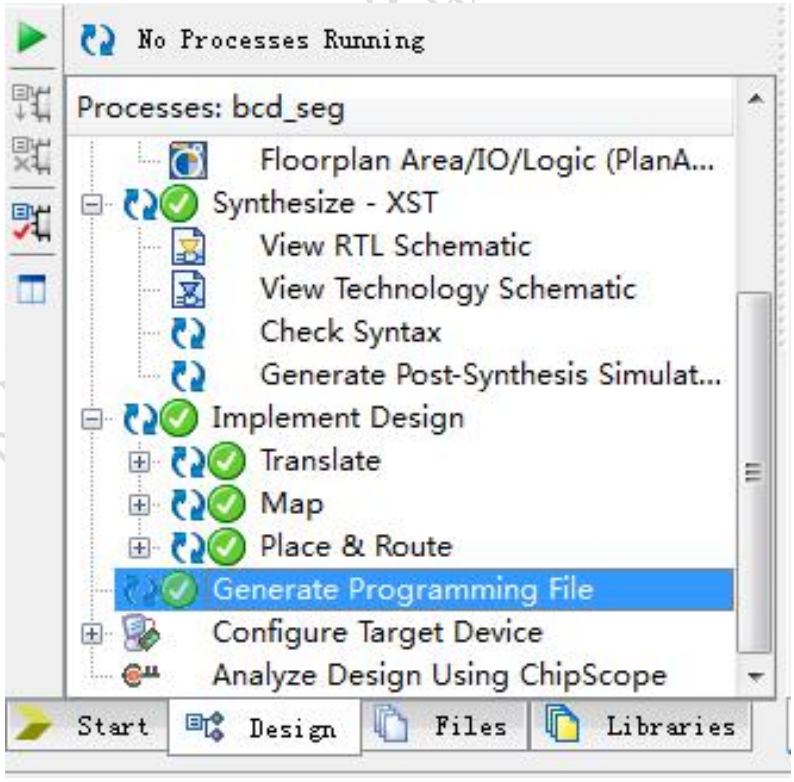


图 1-12

2、打开 Digilent Adept 软件进行下载 FPGA 即可如图 1-13，若需掉电存储则要依次下载 FPGA 和 PROM。

图 1-13

五、基于 ISE 的仿真

1、在 ISE 软件的 Project 菜单下，点击 New Project 如图 1-5。选择 Verilog Test Fixture 测试工具，并输入文件名（**xxx_T**）和存储路径。点击下一步，在总结（Summary）确定无误后点击 Finish 完成工程建立。

2、点击左上角 Design 对话框下的 Sources for：选择 Simulation，并在新建的工程中编写信号激励程序如下图 1-14：

```
1  `timescale 1ns / 1ps
2  module lab1_t;
3     reg a;
4     wire out;
5     lab1 uut (
6        .a(a),
7        .out(out)
8     );
9     initial begin
10       a=0;#100;
11       a=1;#100;
12       a=0;#100;
13    end
14 endmodule
```

图 1-14

程序编写完后，进行语法验证如下图 1-15：

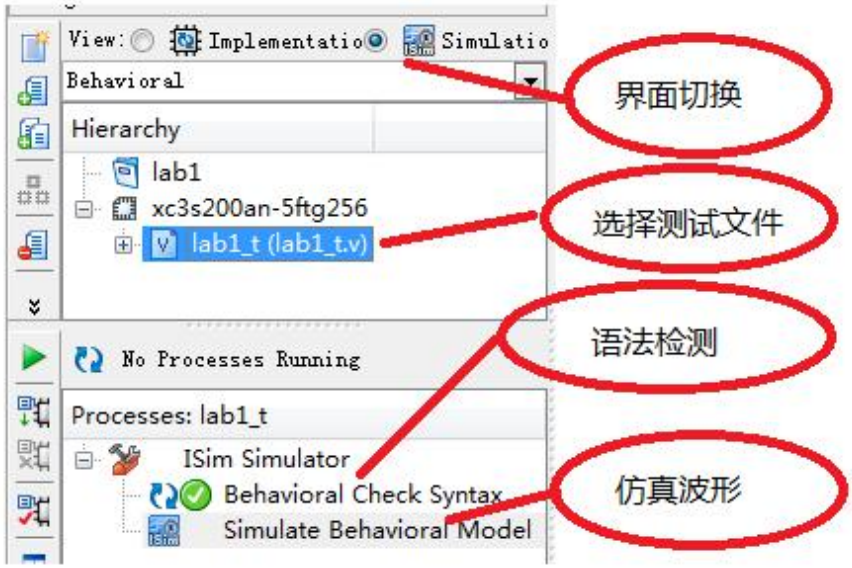


图 1-15

语法无误后，双击 Simulate Behavioral Model 进行仿真，得到下图 1-16：

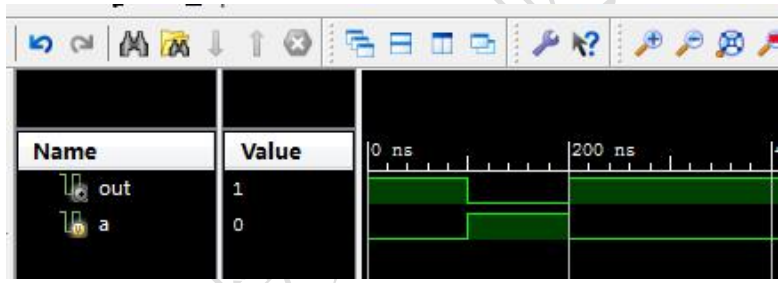


图 1-16

3、如果仿真时间内仍不能看清楚波形，可以在软件下方 Console 对话框中输入代码继续仿真如：run all ,run 1ms 等。

六、发挥部分

1、设计一个 3 输入与非门，并进行仿真。

实验二：加法器设计

一、实验目的

1、掌握简单组合逻辑电路设计思想
2、掌握内置门原语、用户定义的原语的描述方法
3、了解结构化描述的形式

二、实验原理说明

根据数电的基本知识，利用内置门原语设计一个半加器。使用用户定义的原语设计一个加器。真值表及电路原理图如下：

a_0	b_0	s_0	c_1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

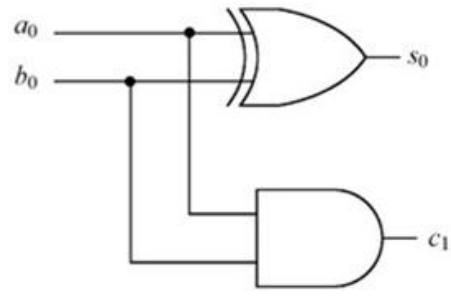


图 2-1

三、实验步骤

1、打开 ISE，新建工程：half_adder+学号.xise。
2、参考实验程序编写程序，实验加法器模块。
3、将程序综合后，生成 half_adder +学号.bit 可下载文件。
4、打开 ISE iMPACT 软件，将 half_adder +学号.bit 文件下载到 FPGA 中（Program FPGA only）。

四、实验程序

```
`timescale 1ns / 1ps
module half_adder(c1,s0, a0,b0);
input a0,b0;
output s0,c1;
……
endmodule
```

其余程序自己写。

五、引脚配置

```
NET "cout" LOC = G1;//LED 高电平亮
NET "sum" LOC = P4;
NET "a" LOC = N3;
NET "b" LOC = E2;
NET "cin" LOC = F3;做全加器时用。
```

六、发挥问题

1、用 3 种方法实验半加器。

2、设计一个 4 位全加器程序。

3、设计一个表决器程序：ABC 三人：A 为 1 时 F 为 1，或者 B 和 C 都为 1 时 F 为 1。其它情况时 F 为 0。

七、思考问题及需掌握要点

1、掌握结构化描述的形式。

2、掌握基本的门级原语的使用。

3、思考：用调用文件的方式完成本实验。

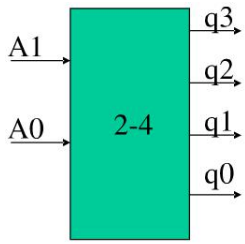
实验三：2-4 译码器设计

一、实验目的

1、 掌握 ISE 软件的使用
2、 掌握 Verilog HDL 的程序结构
3、 掌握 always 语句的用法

二、实验原理说明

本实验为基础实验，设计实验平台上的 K3（G12）、K4（J10）为输入端 A，LED6-LED9（L16、L14、J13、J12）为输出端 Q。按键的输入以 2-4 编码器的编码格式反应到 LED 上。如图 2-1 为译码器真值表，例：输入 A 全（00）$_2$时输出 Q 为（0001）$_2$。



A0	A1	Q3	Q2	Q1	Q0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

图 3-1

三、实验步骤

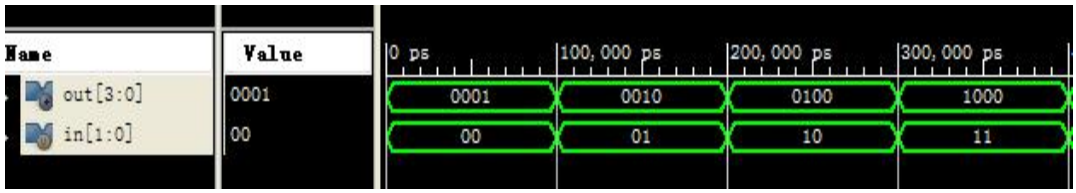
1、 打开 ISE，新建工程：decoder2_4+学号.xise。
2、 参考实验程序编写程序。
3、 将程序综合后，生成 decoder2_4+学号.bit 可下载文件。
4、 打开 ISE iMPACT 软件，将 decoder2_4+学号.bit 文件下载到 FPGA 中（Program FPGA only）。其中下载步骤在实验 1 中有介绍。

四、实验程序

```
`timescale 1ns / 1ps
module decoder2_4(Q,A );
input [1:0]A;
output reg [3:0]Q;
always @(A) begin
使用if语句设计//1<<a;
end
endmodule
```

五、程序仿真

```
module decode2_4_Test;
reg [1:0] A;     wire [3:0] Q;
decode2_4 uut (.Q(Q), .A(A));
initial begin    A = 0;#100;
                 forever begin A=A+1;#100;end
        end
endmodule
```

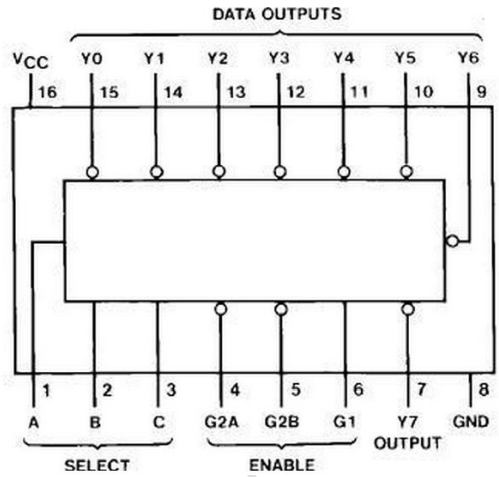


六、引脚配置

NET "Q[3]" LOC = G1;//LED 高电平亮
NET "Q[2]" LOC = P4;
NET "Q[1]" LOC = N4;
NET "Q[0]" LOC = N5;
NET "A[1]" LOC = N3;
NET "A[0]" LOC = E2;

七、发挥部分

1、用“case”和“if”语句分别用两种方法设计 4_2 编码器。
2、设计一个 3_8 译码器。



真值表

输入						输出							
G_1	G_{2A}	G_{2B}	C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
×	H	×	×	×	×	H	H	H	H	H	H	H	H
×	×	H	×	×	×	H	H	H	H	H	H	H	H
L	×	×	×	×	×	H	H	H	H	H	H	H	H
H	L	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	L	H	H	H	H	H	L	H	H	H	H
H	L	L	H	L	L	H	H	H	H	L	H	H	H
H	L	L	H	L	H	H	H	H	H	H	L	H	H
H	L	L	H	H	L	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L

八、思考问题及需掌握要点

1、能否将编码器和译码器各做成一个模块，并将其连接起来（表面现象：两个按键控制分别控制两个 LED）。
2、思考如何用调用用户自定义模块的方法来完成上诉思考题。
3、掌握 Verilog HDL 模块的格式。

实验四：分频器的设计

一、实验目的

1、掌握 ISE 软件的使用
2、掌握 Verilog HDL 的程序结构
3、掌握 always 语句的用法

二、实验原理说明

本实验为基础实验，将实验平台上有源晶振分频，并控制 LED 闪烁。

三、实验步骤

1、打开 ISE，新建工程：fdivision+学号.xise。
设计一个 20 分频器，再将其改成 50M 分频器。
2、将程序综合后，生成 fdivision+学号.bit 可下载文件。
3、打开 ISE iMPACT 软件，将 fdivision+学号.bit 文件下载到 FPGA 中（Program FPGA only）。其中下载步骤在第二章中有介绍。

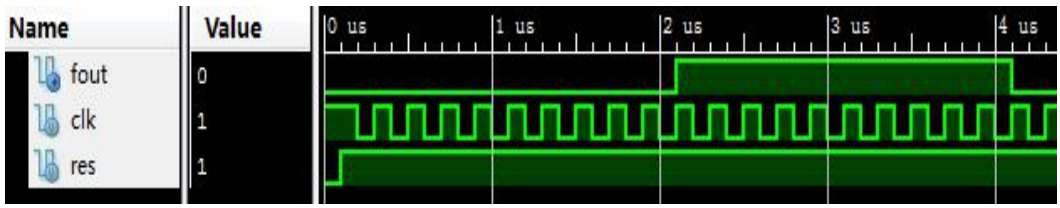
四、实验程序

```
module fdivision(fout,clk,res);
input clk;//定义时钟输入端口；
input   res;
output  ____  fout;//输出端口；
reg [29:0]j;
always@(_____)//时钟上升沿，同步。
begin
      if(res)begin  当前 always 中等式左边的变量复位  end
       else //不复位时，模块正常工作；
          begin
                if(_____)//先做 20 分频，再改成 50M 分频。
                  begin j<=0;fout<=~fout;end
                else j<=j+1;
          end
 end
endmodule
```

五、实验仿真

```
module lab4_t;
      reg clk;
      reg res;
      wire fout;
      fdivision uut (.fout(fout), .clk(clk), .res(res));
      initial
            begin
                  clk=0;res=1;//复位；
                  clk=1;#100;
```

```
        res=0;//不复位；
        forever    #10 clk=!clk;
    end
endmodule
```

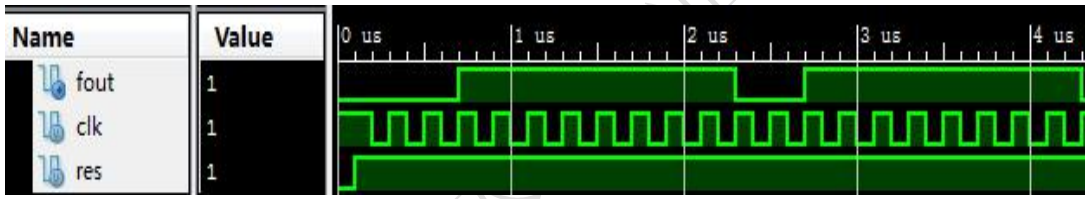


六、引脚配置

NET "res" LOC = N3;//拨码开关，确保为高电平时正常工作。
NET "fout" LOC = M5;//LED0
NET "clk" LOC = B8;//MCLK 时钟默认为 50M。

七、发挥部分

1、将程序改成异步复位程序。
2、实验程序为占空比 50%的方波信号，将程序修改成占空比 80%的 PWM 信号。



3、设计程序，通过按键调节上诉实验中信号的占空比。

八、思考问题及需掌握要点

1、掌握 Verilog HDL 模块的格式。
2、了解阻塞语句与非阻塞语句的异同。

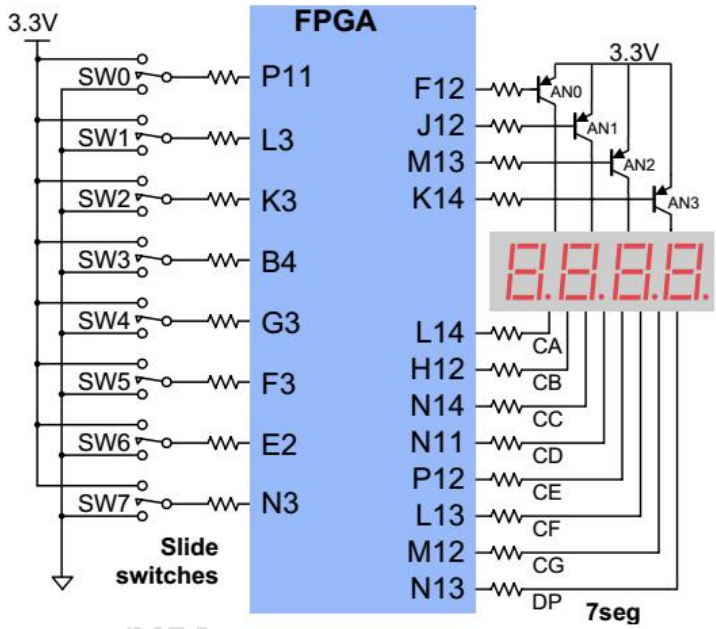
实验五：BCD 译码器

一、实验目的

1、 掌握 ISE 软件的使用
2、 掌握 Verilog HDL 的程序结构
3、 掌握 case 语句的用法

二、实验原理说明

对输入的四位 BCD 码进行译码，把码值显示在数码管上。输入端为拔码开关，四位数码管为共阳数码管。



三、实验步骤

1、打开 ISE，新建工程：bcd_seg+学号.xise。
2、参考实验程序编写程序，实验 bcd 译码模块。
3、将程序综合后，生成 bcd_seg+学号.bit 可下载文件。
4、 打开 ISE iMPACT 软件，将 bcd_seg+学号.bit 文件下载到 FPGA 中（Program FPGA only）。
5、修改程序，让所有数码管都显示 BCD 值。
6、 修改程序，用 case 语句使按键控制 LED 亮灭。

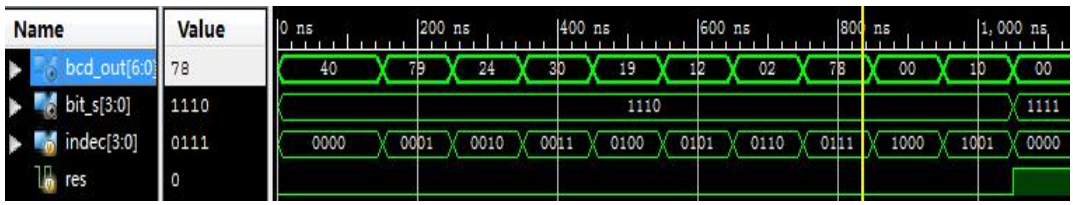
四、实验程序

```
`timescale 1ns / 1ps
module bcd_seg(bcd_out,indec,res,bit_s);
input    [3:0]indec;input res;
output reg [3:0]bit_s;
output reg [6:0]bcd_out;
always@(indec,res)
begin
  if(res)begin bcd_out=7'Bx;bit_s=4'b1111;end
```

```
 else begin
        bit_s=4'b1110;//低电平对应的数码管选通。
case(indec)
0:bcd_out=7'hc0;//共阳数码管；
1:bcd_out=7'hf9;
2:bcd_out=7'ha4;
3:bcd_out=7'hb0;
4:bcd_out=7'h99;
5:bcd_out=7'h92;
6:bcd_out=7'h82;
7:bcd_out=7'hf8;
8:bcd_out=7'h80;
9:bcd_out=7'h90;
default:bcd_out=7'hx;
endcase
end
end
endmodule
```

五、实验仿真

```
module bcd_tf;
      reg [3:0] indec;reg res;
      wire [6:0] bcd_out;
      wire [3:0]bit_s;
      bcd_seg uut (
            .bcd_out(bcd_out), .indec(indec),
            .bit_s(bit_s), .res(res)
      );
      initial begin//过程语句；
         indec = 0;res = 1;#50;
         repeat (10)begin #100 indec=indec+1; end
         res = 0;indec = 0;
         repeat (10)begin #100 indec=indec+1; end
       end
endmodule
```



六、引脚配置

```
NET "bcd_out[0]" LOC = L14;
NET "bcd_out[1]" LOC = H12;NET "bcd_out[2]" LOC = N14;
NET "bcd_out[3]" LOC = N11;NET "bcd_out[4]" LOC = P12;
```

```
NET "bcd_out[5]" LOC = L13;NET "bcd_out[6]" LOC = M12;
NET "bit_s[0]" LOC = F12;//低电平亮
NET "bit_s[1]" LOC = J12;
NET "bit_s[2]" LOC = M13;
NET "bit_s[3]" LOC = K14;
NET "res" LOC = G12;//低电平复位；
NET "indec[0]" LOC = P11;
NET "indec[1]" LOC = L3;
NET "indec[2]" LOC = K3;
NET "indec[3]" LOC = B4;
```

七、发挥部分

1、用函数的方法设计本实验。

2、将程序修改成一个秒表。

八、思考问题及需掌握要点

1、提示：控制四位数码管同时亮，可以修改位选的逻辑属性。

2、掌握 case 的语法格式。

3、掌握静态显示的原理。

实验六：流水灯

一、实验目的

1、掌握 ISE 软件的使用
2、掌握 case 语句的用法
3、掌握简单时序逻辑电路的设计思想

二、实验原理说明

对时钟信号（50M）进行分频得到 1Hz 的信号，用于控制灯的移动速度。使用 case 或左右移语句完成对 LED 的控制。

三、实验步骤

1、打开 ISE，新建工程：ledwater+学号.xise。
2、参考实验程序编写程序，实验 bcd 译码模块。
3、将程序综合后，生成 ledwater+学号.bit 可下载文件。
4、打开 ISE iMPACT 软件，将 ledwater+学号.bit 文件下载到 FPGA 中（Program FPGA only）。
5、 修改程序，设计自己想要的闪烁方式。

四、实验程序

实验程序 1：

```
`timescale 1ns / 1ps
module Flash_LED(clk,led,res);//模块名及端口参数
output reg [7:0]led; //输出端口定义
input clk; //输入端口定义，50M 时钟
input res;
reg[29:0]counter; //中间变量 counter 定义为寄存器型
reg[7:0]location;
always @(posedge clk )
begin
if(res)begin counter<=0; location<=0;led<=8'b0000_0000;end
else
begin
        if(counter>=25000_000)begin
            counter<=0;
            if(location>=8)location<=0;
            else location<=location+1;
                end
      else counter<=counter+1;
        case (location)
            0:led<=8'b0000_0001;
            1:led<=8'b0000_0010;
            ……//其它程序学生自己补全。
```

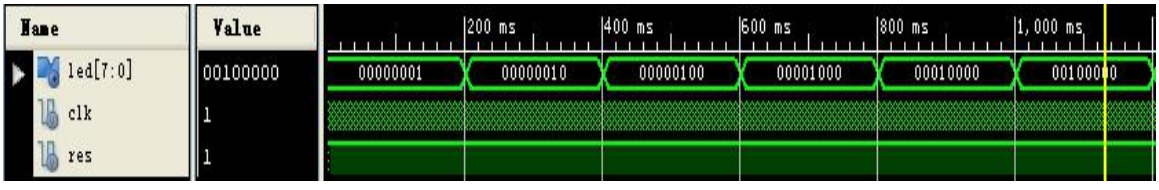
```
                default    :led<=8'b0000_0000;
            endcase
        end
    end
    endmodule
```

实验程序 2：

```
module Flash_LED(clk,led,res);//模块名及端口参数
output reg [7:0]led; //输出端口定义
input clk; //输入端口定义，50M 时钟
input res; //reset
reg[29:0]counter; //中间变量 counter 定义为寄存器型
always @(posedge clk )
begin
if(res)begin counter<=0; led<=8'b0000_0001;end
else
begin
      if(led==0)led<=8'b0000_0001;
        if(counter>=25_000_000)
            begin counter<=0;
              led<=led<<1;
          end
      else counter<=counter+1;
end
end//end always
endmodule
```

五、实验仿真

```
`timescale 1ns / 1ps
module Flash_LED_tf;
    reg clk;reg res;
    wire [7:0] led;
    Flash_LED uut (.clk(clk), .led(led), .res(res));
    initial begin
        clk = 0;res = 1; clk=1;#100;
        res=0;
        forever begin #25 clk=~clk;end
    end
endmodule
```



六、引脚配置

```
NET "led[0]" LOC = G1;
NET "led[1]" LOC = P4;
NET "led[2]" LOC = N4;
NET "led[3]" LOC = N5;
NET "led[4]" LOC = P6;
NET "led[5]" LOC = P7;
NET "led[6]" LOC = M11;
NET "led[7]" LOC = M5;
NET "res" LOC = A7;//确保开关为高电平；
NET "clk" LOC = B8;
```

七、发挥部分

1、编写程序实现多样花样流水灯模式自动切换。

2、使用按键控制流水灯的花样选择。

八、思考问题及需掌握要点

1、掌握 case 的语法格式及左右移指令的用法。

2、了解时序电路程序的编写方法。

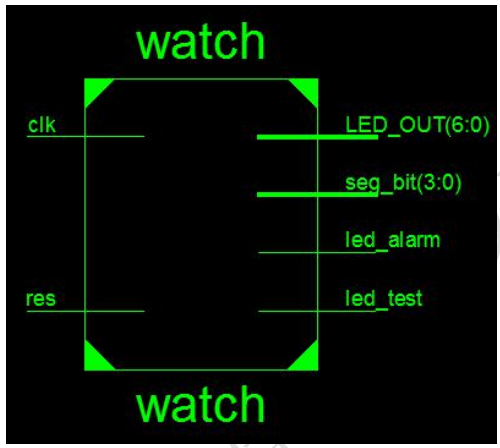
实验七：数字钟

一、实验目的

1、掌握简单状态机的设计思想
2、掌握 case 语句的用法
3、掌握数字钟的工作原理

二、实验原理说明

对时钟信号（50M）进行分频得到动态扫描的位选信号与及秒脉冲信号，用于控制 4 位数码管的扫描速度（速度必须合适）及时钟秒信号。使用 case 语句控制输出数据并显示。



三、实验步骤

1、 打开 ISE，新建工程：watch+学号.xise。
2、 参考实验程序编写程序，实验 bcd 译码模块。
3、将程序综合后，生成 watch +学号.bit 可下载文件。
打开 ISE iMPACT 软件，将 watch +学号.bit 文件下载到 FPGA 中（Program FPGA only）。

四、实验程序

```
`timescale 1ns / 1ps
module watch(clk,LED_OUT,seg_bit,led_test,res,led_alarm );
input clk,res;
output reg led_alarm;
reg clk_1hz;
reg clk_1khz;
//wire clk_1kHz_G;
output reg led_test;
output reg [3:0]seg_bit;//定义 seg_bit 为 4 位的输出口；
output reg [6:0]LED_OUT;
reg [25:0]counter;
reg [25:0]counter1;//分频寄存器；
reg [7:0]display_tab;
reg [7:0]second_L=5,second_H=5;
reg [7:0]miniute_L=9,miniute_H=5;
```

```
//reg [7:0]hour_L,hour_H;
parameter s0=4'b0001,s1=4'b0010,s2=4'b0100,s3=4'b1000;//独热码
reg [3:0]state;
////////////////////////////////////////////////分频器//////////////////////////////////////////////////
always@(posedge clk)
begin
    if(res)begin counter1<=0;clk_1hz<=0;counter<=0;clk_1khz<=0;end
     else begin
            if(counter1>=25_000_000)//分频，产生秒脉冲信号；
              begin counter1<=0;clk_1hz<=~clk_1hz;end
          else counter1<=counter1+1;
            if(counter>=25_000)//分频，产生动态扫描控制脉冲信号
             begin counter<=0;clk_1khz<=~clk_1khz;end
             //state=state+1;if(state==4)state=0;
             //非状态机方法提示
           else begin counter<=counter+1;end
    end
end
////////////////////////////////////////////时钟主体//////////////////////////////////////////////////
always@(posedge clk_1hz,posedge res)//异步复位；
begin
    if(res)begin
        second_L<=5;second_H<=5;
        miniute_L<=9;miniute_H<=5;
        led_alarm<=0;
        end
    else
    begin
        if(second_L>=9)
        begin
        second_L<=0;
          if(second_H>=5)
            begin second_H<=0;
                if(miniute_L>=9)
                  begin
                    miniute_L<=0;
                    if(miniute_H>=5)miniute_H<=0;
                    else miniute_H<=miniute_H+1;
                  end
                else miniute_L<=miniute_L+1;
            end
          else second_H<=second_H+1;
        end
      else second_L<=second_L+1;
```

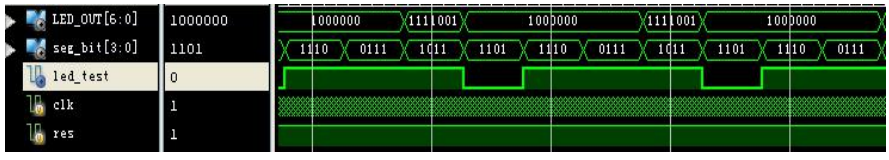
```
if(miniute_L==0&&miniute_H==0&&second_H<=1&&second_L<9)//到点报时；
      led_alarm<=~led_alarm;
      else led_alarm<=0;
    end
end

//BUFG AA(clk_1kHz_G,clk_1khz);//全局时钟，避免时钟倾斜。
////////////////////////////////////////////动态扫描块//////////////////////////////////////////////
always@(posedge clk_1kHz_G,posedge res)//异步复位；
begin
      if(res)begin state<=0;display_tab<=0;seg_bit<=4'b1111;led_test<=0;end
        else begin
        case(state)
        s0://第一步：让数码管的最低位亮，并显示秒的个位值。
        begin
              seg_bit<=4'b0111;
              display_tab<=second_L;
              led_test<=1;
              state<=s1;
        end
        s1: //第二步：让数码管的第 2 位亮，并显示秒的十位值。
        begin
              seg_bit<=4'b1011;
              display_tab<=second_H;
              led_test<=1;//led_test=~clk_1hz;
              state<=s2;
        end
        s2: //第三步：让数码管的第 3 位亮，并显示分的个位值。
        begin
             seg_bit<=4'b1101;
             display_tab<=miniute_L;
             led_test<=~clk_1hz;
             state<=s3;
        end
        s3: //第四步：让数码管的最高位亮，并显示分的十位值。
        begin
              seg_bit<=4'b1110;
              display_tab<=miniute_H;
              led_test<=1;
              state<=s0;
        end
        default: begin    state<=s0; end
        endcase
    end
```

```
end
//////////////////////////////////////////////////BCD 译码器//////////////////////////////////////////
always@(display_tab,res)
begin
    if(res)begin LED_OUT<=7'hff; end
     else
                case (display_tab)//数码管显示表；
                  0:LED_OUT <=7'b100_0000;//0
                  1:LED_OUT <=7'b111_1001;//1
                  2:LED_OUT <=7'b010_0100;//2
                  3:LED_OUT <=7'b011_0000;//3
                  4:LED_OUT <=7'b001_1001;//4
                  5:LED_OUT <=7'b001_0010;//5
                  6:LED_OUT <=7'b000_0010;//6
                  7:LED_OUT <=7'b111_1000;//7
                  8:LED_OUT <=7'b000_0000;//8
                  9:LED_OUT <=7'b001_0000;//9
                default:LED_OUT <=7'bz;
                endcase
end
endmodule
```

五、实验仿真

```
module clock_tf;
      reg clk;
      wire [6:0] LED_OUT;
      wire [3:0] seg_bit;
      wire led_test;
      reg res ;
      watch uut (
          .clk(clk), .LED_OUT(LED_OUT),
          .seg_bit(seg_bit), .led_test(led_test),.res(res));
      initial begin
          clk = 0;res=0;
          #100;
          res=0;
        forever begin #25 clk=~clk;end
      end
endmodule
```



六、引脚配置

```
NET "LED_OUT[0]" LOC = L14;
NET "LED_OUT[1]" LOC = H12;
NET "LED_OUT[2]" LOC = N14;
NET "LED_OUT[3]" LOC = N11;
NET "LED_OUT[4]" LOC = P12;
NET "LED_OUT[5]" LOC = L13;
NET "LED_OUT[6]" LOC = M12;
NET "seg_bit[3]" LOC = F12;//低电平亮
NET "seg_bit[2]" LOC = J12;
NET "seg_bit[1]" LOC = M13;
NET "seg_bit[0]" LOC = K14;
NET "res" LOC = A7;//低电平复位；
NET "led_alarm" LOC = M5;
NET "led_test" LOC = N13;
NET "clk" LOC = B8;
```

七、发挥部分

1、添加到点报时功能或闹钟功能。

2、用状态机的方法编写 15 进制加法计数器。

八、思考问题及需掌握要点

1、掌握 case 的语法格式。

2、掌握状态机的程序编写思想。

3、思考：用状态机的方法编写多人抢答器。

实验八：ASK 调制

一、实验目的

1、掌握 ASK 调制的理论

2、掌握 ASK 调制的 verilog hdl 实现

二、实验原理说明

本实验采用键控法，典型实现方法是用一个电子开关（与门）来控制载波振荡器的输出而获得，如图 8-1。

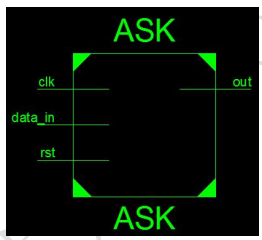


图 8-1

三、实验步骤

1、打开 ISE，新建工程：ask_coder+学号.xise。

2、参考实验程序编写程序，实验 ask_coder 模块。

3、将程序综合后，生成 ask_coder +学号.bit 可下载文件。

4、打开 ISE iMPACT 软件，将 ask_coder +学号.bit 文件下载到 FPGA 中（Program FPGA only）。

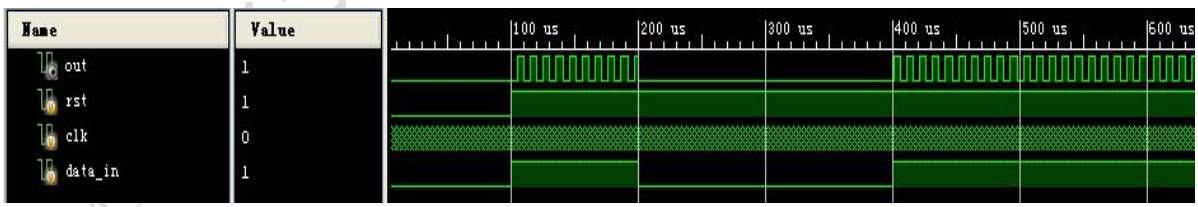
四、实验程序

```
`timescale 1ns / 1ps
module ASK(out,data_in,rst,clk);
output out;
input data_in;input rst;input clk;
reg [9:0]count;reg carry;reg flag;
assign out=flag&carry;
always@(posedge clk)
begin
    if(rst)begin count<=0;carry<=0;flag<=0;end
    else
      begin
```

```
            if(count>=50)
                  begin count<=0;carry<=~carry;   //载波生成
                 end
            else count<=count+1;
            if(data_in)flag<=1;else flag<=0;//基带信号
            end
end
```

五、实验仿真

```
module ask_tf;
      reg rst;
      reg clk;
      reg data_in;
      wire out;
      ASK uut (
            .out(out), .data_in(data_in),
            .rst(rst), .clk(clk)
      );
      initial begin
            clk = 0;rst = 1;
            clk = 1;
            #100;
            rst=0;
            data_in = 1;#100000;
            data_in = 0;#200000;
            data_in = 1; #200000;
      end
     always     begin #50    clk=!clk;end
endmodule
```



六、引脚配置

```
NET "clk" LOC = P9;       NET "data_in" LOC = K11;
NET "out" LOC = D13;      NET "rst" LOC = K12;
```

七、发挥部分

1、修改程序中，调整调制信号频率。

2、修改程序中，改变基带信号。

八、思考问题及需掌握要点

1、掌握 ASK 的实现方法。

2、思考：ASK 的解调方法。

附录

3 输入与非门

```
module vote(a,b,c,f);
input a,b,c;
output f;
wire a,b,c,f;
assign f=(a&b)|(a&c)|(b&c);
endmodule
```

半加器 1：

```
module half_adder(a,b,sum,carry);
input a,b;
output sum,carry;
assign sum=a^b;
assign carry=a&b;
endmodule
```

半加器 2：

```
module half_adder(carry,sum,a,b);
output carry,sum;
input a,b;
assign{carry,sum}=a+b;
endmodule
```

全加器:

```
module falf_adder(carry,sum,a,b,cin);
output carry,sum;
input a ,b,cin;
assign{carry,sum}=a+b+cin;
endmodule;
```

2－4 4－2 编译码器及拼接

```
`timescale 1ns / 1ps
module coder2_4(out,in);
output reg [3:0]out;
input [1:0]in;
```

```
always@(in)
begin
      if(in==2'b00)out=4'b0001;
      else if(in==2'b01)out=4'b0010;
      else if(in==2'b10)out=4'b0100;
      else if(in==2'b11)out=4'b1000;
      else out=4'bx;
end
endmodule

module decoder4_2(out,in);
output reg [1:0]out;
input [3:0]in;
always@(in)
begin
if(in==4'b0001)out=2'b00;
else if(in==4'b0010)out=2'b01;
else if(in==4'b0100)out=2'b10;
else if(in==4'b1000)out=2'b11;
else out=2'bx;
end
endmodule

module aa(out,in);
input [1:0]in;
output [1:0]out;
wire [3:0]w;
coder2_4 aaa (.out(w), .in(in));
decoder4_2 bbb (.out(out), .in(w));
endmodule
```

3_8 编码器

```
module decode38(y,a,g1,g2a,g2b);
output reg [7:0]y;
input [2:0]a;
input g1,g2a,g2b;
always@(a,g1,g2a,g2b)
begin
  if(g1&&!g2a&&!g2b)//正常工作。
    begin
      case(a)
          3'b000:y=8'b1111_1110;
            1:y=8'b1111_1101;
            2:y=8'b1111_1011;
```

```
            3:y=8'b1111_0111;
            4:y=8'b1110_1111;
            5:y=8'b1101_1111;
            6:y=8'b1011_1111;
            3'b111:y=8'b0111_1111;
          default:y=8'b1111_1111;
      endcase
      end
   else y=8'b1111_1111;
end
endmodule
```

占空比 80% 波形设计

```
module f_div10(f,clk,res);
output reg f;
input clk,res;
reg [3:0]count;
parameter PWM=8;//修改可以改占空比。
always @(posedge clk,posedge res)begin //异步复位
      if(res)begin count<=0;f<=0;end//re 变量
      else begin
            if(count<PWM) begin f<=1;end
            else if(count>=PWM)f<=0;
            if(count>=99)count<=0;
              else    count<=count+1;
      end
end
endmodule
```

仿真程序、波形与实验四相同。

让数码管从 0~9 循环变化。

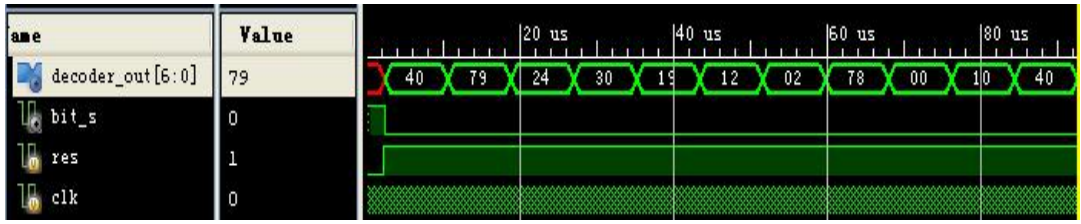
```
module test(
      input res,
      input clk,
      output reg [6:0] decoder_out,//7 段数码管
        output reg bit_s//4 位共阳数码管，用其中一位；
      );
        reg [3:0]indec;
        reg[29:0]j;
always@(posedge clk)//时钟上升沿，同步复位；
begin
  if(res)
    begin
      decoder_out<=7'hx;indec<=0;bit_s<=1;j<=0;
```

```
  end
 else
       begin
              if(j>=40)//改成 50000000 ， 50M 分频方便观察。
                   begin j<=0;if(indec>=9)indec<=0;
                                 else indec<=indec+1;
                   end
              else    j<=j+1;
          bit_s<=0;
         case(indec)
          0:begin decoder_out<=7'hc0; end
          1:begin decoder_out<=7'hf9; end
          2:begin decoder_out<=7'ha4; end
          3:begin decoder_out<=7'hb0; end
          4:begin decoder_out<=7'h99; end
          5:begin decoder_out<=7'h92; end
          6:begin decoder_out<=7'h82; end
          7:begin decoder_out<=7'hf8; end
          8:begin decoder_out<=7'h80; end
          9:begin decoder_out<=7'h90; end
          default:decoder_out<=7'h0;
          endcase
     end
end
endmodule
```

仿真程序

```
module bcd_tf;
     reg res;
     reg clk;
     wire [6:0] decoder_out;
     wire bit_s;
     test uut (
          .res(res),
          .clk(clk),
          .decoder_out(decoder_out),
          .bit_s(bit_s)
     );
     initial begin
          clk = 0;res = 1;clk = 1;
          #100;
         repeat(20)begin #100 clk=!clk;end
          res=0;#100;
          forever begin #100 clk=!clk;end
     end
```

endmodule



按键控制花样流水灯

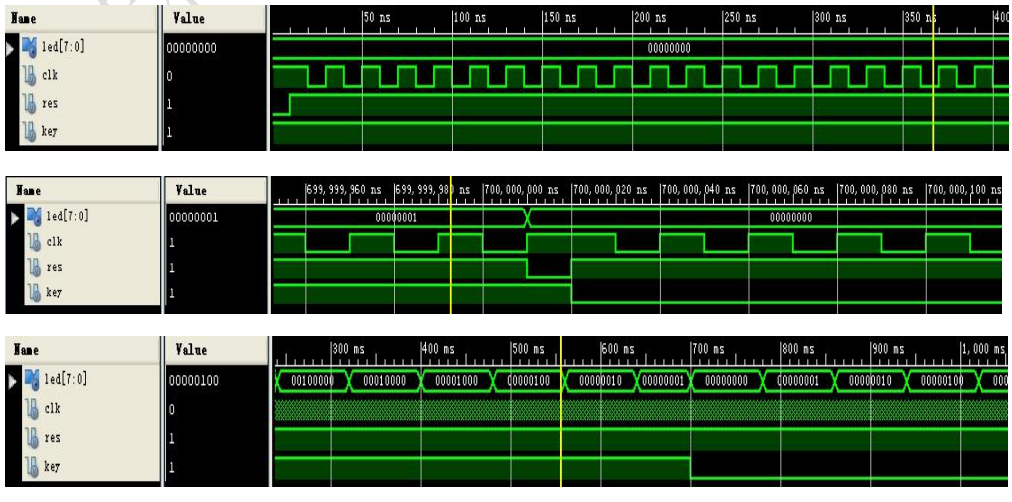
```
module ledwater(clk,led,res,key);//模块名及端口参数
output reg [7:0]led; //输出端口定义
input clk; //输入端口定义，50M 时钟
input res; //reset
input key;//按键切换方向。
reg[29:0]counter,key_count; //中间变量 counter 用于分频，
//key_count 用于防抖
reg bit_flag;//等于 0 左移，等于 1 右移。
////////////////////////////////////////////////
always @(posedge clk)
begin
            if(!res)begin counter<=0; key_count<=0;
                        led<=8'b0000_0000;bit_flag<=0;end
            else
              begin
                 if(counter>=25_000_000)
                      begin counter<=0;
                              if(bit_flag)
                                  begin
                                  led<=led<<1;
                                  if(led==0)led<=8'b0000_0001;
                                  end
                               else
                                  begin
                                  led<=led>>1;
                                  if(led==0)led<=8'b1000_0000;
                                  end
                        end
              else counter<=counter+1;
            end
          if(!key)
              begin
                  key_count<=key_count+1;
                   if(key_count>=100000)
                   begin
                   key_count<=0;
```

```
                    if(!key)
                      begin bit_flag<=!bit_flag;end
                    end
                end       else key_count<=0;
end
endmodule
```

仿真程序

```
`timescale 1ns / 1ps
module con_dir_ledwater_tf;
      reg clk;
      reg res;
      reg key;
      wire [7:0] led;
      ledwater uut (.clk(clk), .led(led), .res(res), .key(key));
      initial begin
            clk = 0;
            res = 1;
            key = 1;
            clk=1;
            #10;
         res=0;
         repeat(70000000)begin #10 clk=!clk;end
            clk=0;
            res=0;
            clk=1;
            #10; //同步复位；
            res=1;key=0;
         repeat(70000000)begin #10 clk=!clk;end
      end
endmodule
```

下图 1 是初始化波形，下图 2 是按键按下时的波形，下图 3 是整体波形。

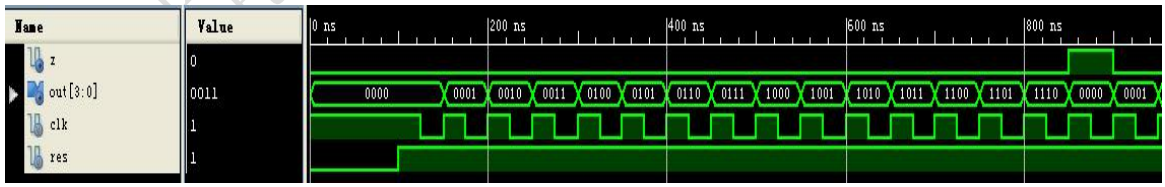


用多种方法设计模 15 的加法计数器

```
`timescale 1ns / 1ps
module count15(input clk, input res,output reg z,output reg [3:0] out );
always @(posedge clk)
begin
   if(res)begin out<=0;z<=0; end
   else
    begin
      if(out>=14)    begin out<=0;z<=1;end
      else begin out<=out+1;z<=0;end
    end
end
endmodule
```

模 15 的加法计数器仿真程序

```
`timescale 1ns / 1ps
module count15_tf;
      reg clk;
      reg res;
      wire z;
      wire [3:0] out;
     count15 uut (.clk(clk), .res(res), .z(z),.out(out));
      initial begin
            clk = 0;res = 1;clk=1;
            #100;
            res=0;
          forever begin #25 clk=~clk;end
      end
endmodule
```



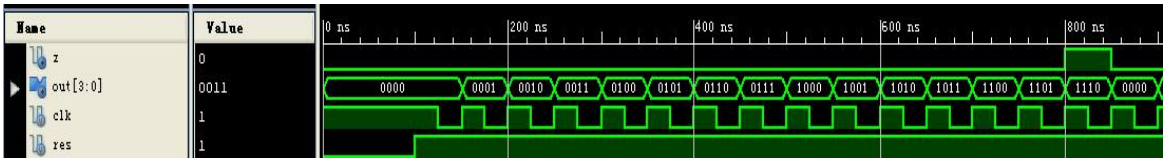
使用状态机的模 15 计数器，采用单个 always 语句，输出与输入是同步的。

```
`timescale 1ns / 1ps
module count15(input clk,input res,output reg z,output reg [3:0] out);
parameter s0=0,s1=1,s2=2,s3=3,s4=4,s5=5,
            s6=6,s7=7,s8=8,s9=9,s10=10,s11=11,
            s12=12,s13=13,s14=14;
always @(posedge clk)
begin
   if(res)begin out<=s0;z<=0; end
```

```
   else
    begin
      case (out)
        s0:begin out<=s1;z<=0;end
        s1:begin out<=s2;z<=0;end
        s2:begin out<=s3;z<=0;end
        s3:begin out<=s4;z<=0;end
        s4:begin out<=s5;z<=0;end
        s5:begin out<=s6;z<=0;end
        s6:begin out<=s7;z<=0;end
        s7:begin out<=s8;z<=0;end
        s8:begin out<=s9;z<=0;end
        s9:begin out<=s10;z<=0;end
        s10:begin out<=s11;z<=0;end
        s11:begin out<=s12;z<=0;end
        s12:begin out<=s13;z<=0;end
        s13:begin out<=s14;z<=0;end
        s14:begin out<=s0;z<=1;end
        default:out<=s0;
      endcase
    end
end
endmodule
```

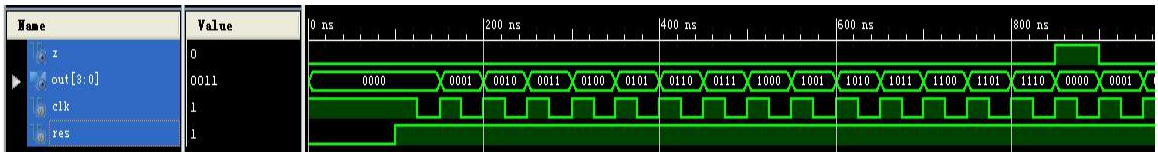
模 15 计数器仿真程序

```
`timescale 1ns / 1ps
module count15_tf;
      reg clk;
      reg res;
      wire z;
      wire [3:0] out;
    count15 uut (.clk(clk), .res(res), .z(z),.out(out));
      initial begin
          clk = 0;
          res = 1;clk=1;
          #100;
          res=0;
        forever begin #25 clk=~clk;end
      end
endmodule
```



采用 2 个 always 语句设计模 15 计数器，输出滞后 1 个时钟（摩尔机）。

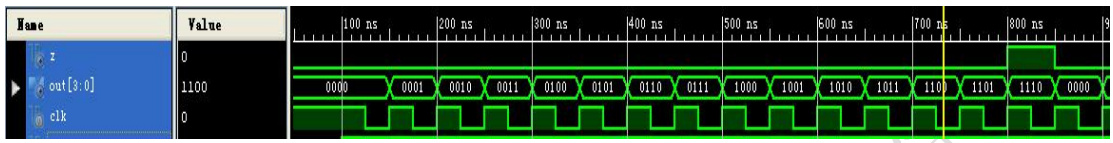
```
`timescale 1ns / 1ps
module count15(input clk, input res,output reg z,output reg [3:0] out);
parameter s0=0,s1=1,s2=2,s3=3,s4=4,s5=5,
                s6=6,s7=7,s8=8,s9=9,s10=10,s11=11,
                 s12=12,s13=13,s14=14;
always @(posedge clk)//第一个 always 用于状态的转换。
begin
   if(res)begin out<=s0; end //采用同步复位。
   else
    begin
       case (out)
       s0:out<=s1;
         s1:out<=s2;
         s2:out<=s3;
         s3:out<=s4;
         s4:out<=s5;
         s5:out<=s6;
         s6:out<=s7;
         s7:out<=s8;
         s8:out<=s9;
         s9:out<=s10;
         s10:out<=s11;
         s11:out<=s12;
         s12:out<=s13;
         s13:out<=s14;
         s14:out<=s0;
         default:out<=s0;
      endcase
    end
end
 always @(out)//第二个 always  语句。输出逻辑 Out logic
 begin
   if(out==s14)z=1;else z=0;
 end
endmodule
```



采用 3 个 always 语句设计模 15 计数器，输出 z 在 out 等于 4’B1110 时马上为 1，这与采用 2 个 always 语句的方法是不同的。

```
`timescale 1ns / 1ps
module count15(input clk,input res,output reg z,output reg [3:0] out);
reg [3:0]next_state,state;
parameter s0=0,s1=1,s2=2,s3=3,s4=4,s5=5,
          s6=6,s7=7,s8=8,s9=9,s10=10,s11=11,
          s12=12,s13=13,s14=14;
always @(posedge clk)//定义当前状态 cs
begin
  if(res)begin state<=s0; end //同步复位
  else   state<=next_state;
end
always @(state)//定义下一状态 ns
 begin
  case(state)
      s0:begin next_state<=s1;end
        s1:begin next_state<=s2;end
        s2:begin next_state<=s3;end
        s3:begin next_state<=s4;end
        s4:begin next_state<=s5;end
        s5:begin next_state<=s6;end
        s6:begin next_state<=s7;end
        s7:begin next_state<=s8;end
        s8:begin next_state<=s9;end
        s9:begin next_state<=s10;end
        s10:begin next_state<=s11;end
        s11:begin next_state<=s12;end
        s12:begin next_state<=s13;end
        s13:begin next_state<=s14;end
        s14:begin next_state<=s0;end
        default:next_state<=s0;
  endcase
 end
always @(state)//定义输出逻辑关系 ol
begin
      case (state)
        s0:begin out=0;z=0;end
        s1:begin out=1;z=0;end
        s2:begin out=2;z=0;end
        s3:begin out=3;z=0;end
        s4:begin out=4;z=0;end
        s5:begin out=5;z=0;end
        s6:begin out=6;z=0;end
        s7:begin out=7;z=0;end
        s8:begin out=8;z=0;end
```

```
        s9:begin out=9;z=0;end
        s10:begin out=10;z=0;end
        s11:begin out=11;z=0;end
        s12:begin out=12;z=0;end
        s13:begin out=13;z=0;end
        s14:begin out=14;z=1;end
        default:begin out=0;z=0;end
      endcase
    end
endmodule
```



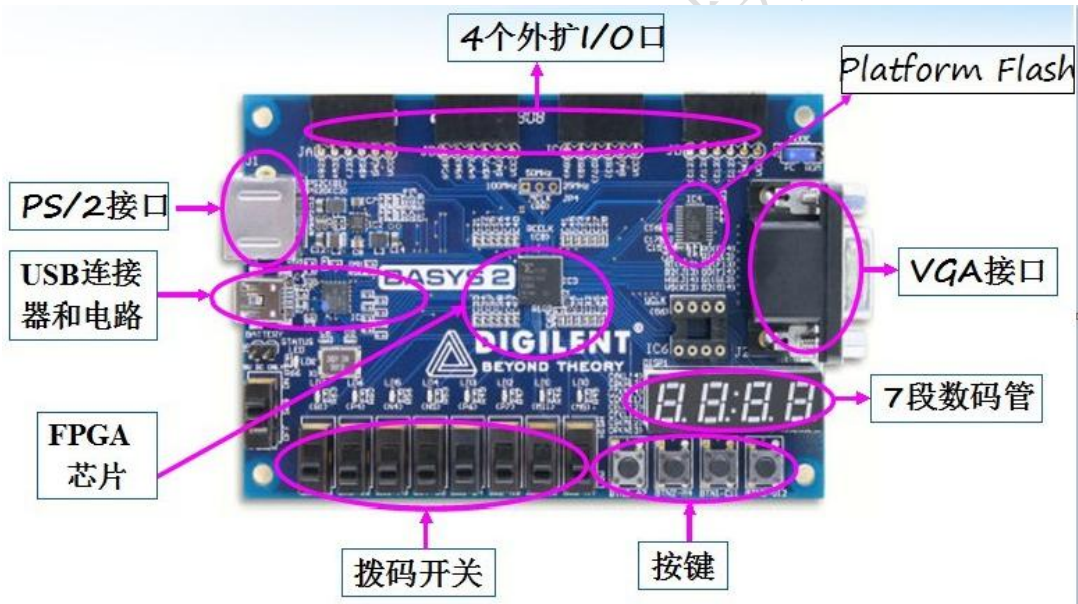
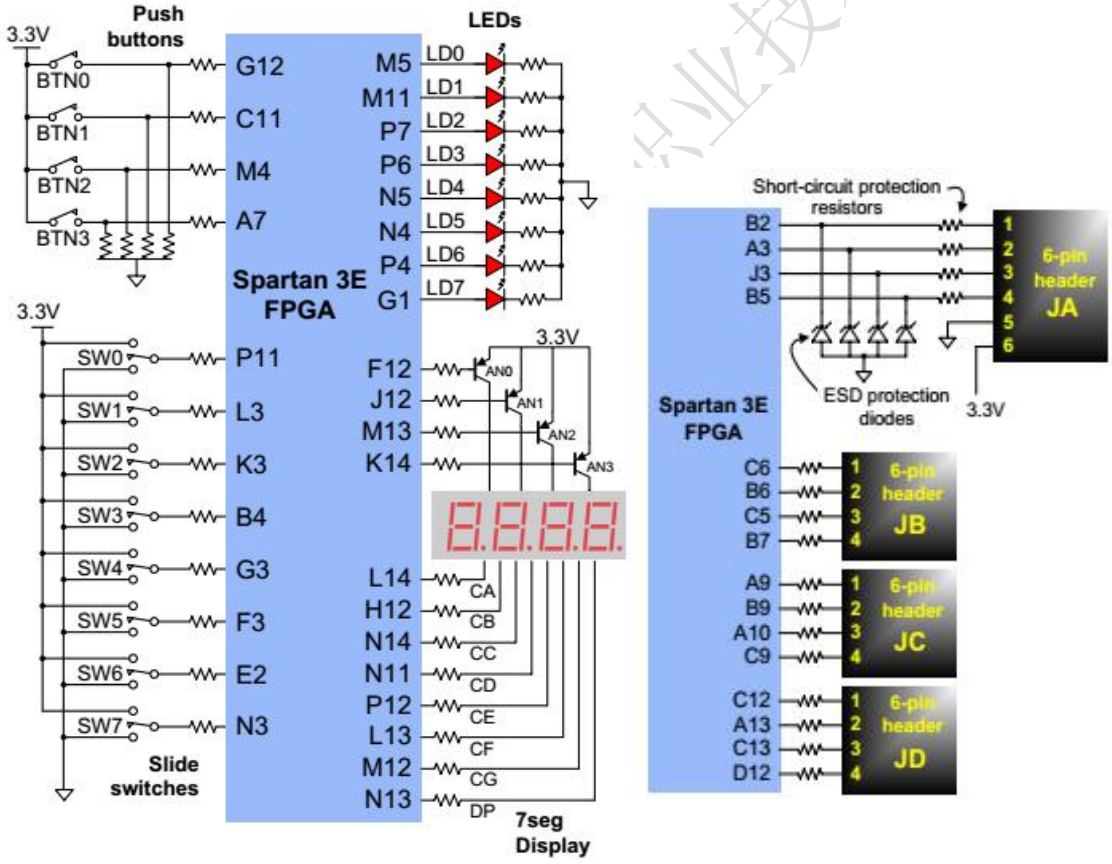
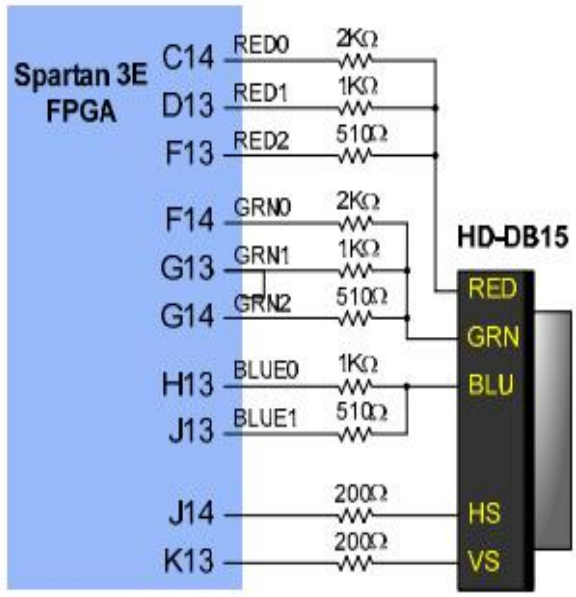
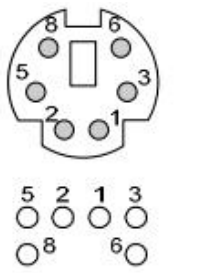


表4.3.1　Basys2各IO管脚定义

发光二极管		时钟		拨码开关		按键		数码管	
LD0	M5	MCLK	B8	SW0	P11	BTN0	G12	AN0	F12
LD1	M11	RCCLK	C8	SW1	L3	BTN1	C11	AN1	J12
LD2	P6	CCLK	N12	SW2	K3	BTN2	M4	AN2	M13
LD3	P7	UCLK	M6	SW3	B4	BTN3	A7	AN3	K14
LD4	N5			SW4	G3	BTN4		CA	L14
LD5	N4			SW5	F3			CB	H12
LD6	P4			SW6	E2			CC	N14
LD7	G1			SW7	N3			CD	N11
								CE	P12
								CF	L13
								CG	M12
								DP	N13







Pin1: Data
Pin2: Data
Pin3: GND
Pin5: Vdd
Pin6: Clock
Pin8: Clock