**北京邮电大学数字电路与逻辑**

**设计实验**

****

学院：

班级：

作者：

学号：

题目：简易像素鸟的设计与实现

指导老师：

**目 录**

[**1.** **设计课题的任务要求** 4](#_Toc466318183)

[**2.** **系统设计** 5](#_Toc466318184)

[2.1 设计思路 5](#_Toc466318185)

[2.2 总体框图 6](#_Toc466318186)

[2.3 分块设计 8](#_Toc466318187)

[2.3.1 bird\_position 模块 8](#_Toc466318188)

[2.3.2 bird\_position2 模块 9](#_Toc466318189)

[2.3.3 cleanbtn 模块 10](#_Toc466318190)

[2.3.4 clk\_3 模块 12](#_Toc466318191)

[2.3.5 seg7\_1 模块 13](#_Toc466318192)

[2.3.6 random 模块 14](#_Toc466318193)

[2.3.7 pipe\_move 模块 15](#_Toc466318194)

[2.3.8 mode\_select 模块 16](#_Toc466318195)

[2.3.9 div\_clk3 模块 17](#_Toc466318196)

[2.3.10 div\_clk1 模块 18](#_Toc466318197)

[2.3.11 div\_clk 模块 19](#_Toc466318198)

[**3.** **仿真波形及波形分析** 20](#_Toc466318199)

[**3.1 bird\_position** 20](#_Toc466318200)

[**3.2 cleanbtn** 20](#_Toc466318201)

[**3.3 seg7\_1** 21](#_Toc466318202)

[**3.4 pipe\_move** 21](#_Toc466318203)

[**4.** **源程序** 22](#_Toc466318204)

[**5.** **功能说明及资源利用情况** 60](#_Toc466318205)

[**5.1 bird\_position** 61](#_Toc466318206)

[**5.2 bird\_position2** 62](#_Toc466318207)

[**5.3 cleanbtn** 62](#_Toc466318208)

[**5.4 clk\_3** 62](#_Toc466318209)

[**5.5 seg7\_1** 63](#_Toc466318210)

[**5.6 random** 63](#_Toc466318211)

[**5.7 pipe\_move** 64](#_Toc466318212)

[**5.8 mode\_select** 64](#_Toc466318213)

[**5.9 div\_clk3** 65](#_Toc466318214)

[**5.10 div\_clk1** 65](#_Toc466318215)

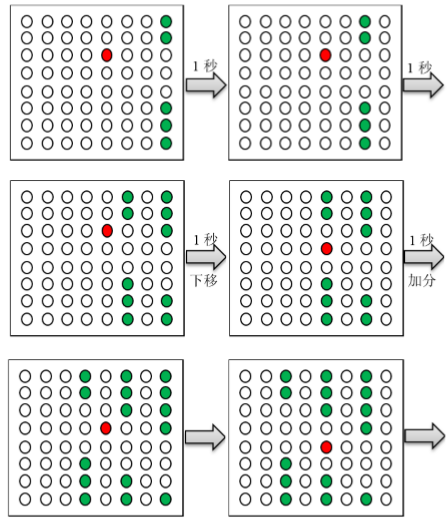
[**5.11 div\_clk** 66](#_Toc466318216)

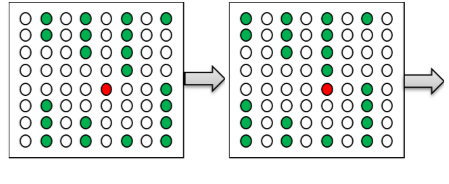
[**6.** **故障及问题分析** 66](#_Toc466318217)

[**7.** **总结和结论** 67](#_Toc466318218)

1. **设计课题的任务要求**
   * + 题目内容：简易像素鸟游戏的设计与实现。设计一个模仿像素鸟（Flappy Bird）的游戏。游戏中，玩家控制一只小鸟飞过一个个管 道，飞得越远分数越高。
     + 基本要求：

* 用 8×8 点阵进行游戏显示，游戏过程示意如下图。其中红色 LED 为小鸟，绿色 LED 表示水管。水管每秒左移 1 列，两个水管出现的时间间隔为 2 秒。水管中间有缝隙， 缝隙的位置随机，缝隙的高度固定为 3 个 LED。玩家通过 BTN0～BTN1 两个按键 控制小鸟进行上下移动，使小鸟能恰好通过水管缝隙。





* 用两个数码管显示游戏成绩。小鸟每穿过一个水管玩家得 1 分，当游戏成绩达到 12 分时游戏结束，点阵显示字符“V”。如果小鸟碰到水管则游戏结束，点阵显示字符 “X”。
* 按复位键重新开始游戏。
* 提高要求：

多种速度可以选择； 可以进行双人游戏，由两个人各自控制一只小鸟进行游戏； 增加其他游戏模式； 为游戏增加音效； 在 LCD1602 液晶屏上显示游戏成绩； 自拟其他功能。

1. **系统设计**

## 2.1 设计思路

整体采取模块化设计方案，共分成11个模块，包括防抖模块，分频模块，小鸟坐标模块，点阵输出模块，数码管显示模块，模式选择模块，随机发生器模块，读秒模块。

输入部分，btn按键首先接至防抖模块，防抖模块采用采样输出从而获得单位方波的输出，达到防抖的效果，拨码开关接各自对应信号。输出部分，点阵输出模块经一时钟沿便产生3组8位的信号输出，分别接入开发板的红显示、绿显示、点阵公共阴极3组向量引脚；同理数码管输出模块经一时钟沿便产生对应向量，用于接入开发板的数码管显示部分的引脚。

中间部分，分频器用于对50mhz主时钟的分频，输出各模块所需要的频率，小鸟坐标模块通过响应“上下左右”的按键输出小鸟的横纵坐标到点阵输出模块，模式选择模块通过响应拨码开关输出对应模式的信号量至点阵输出模块，随机发生器模块给点阵输出模块提供随机数，读秒模块用于画面的更新，同样接入点阵输出模块，实现了管道的移动。

## 2.2 总体框图





（清晰图见图像文件flappybird.jpg）

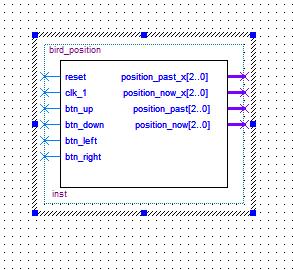
接口说明：





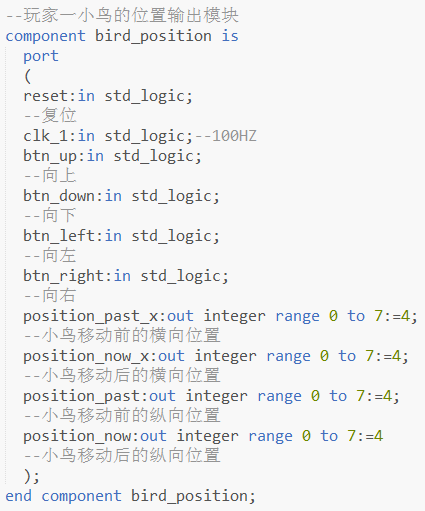
## 2.3 分块设计

### 2.3.1 bird\_position 模块

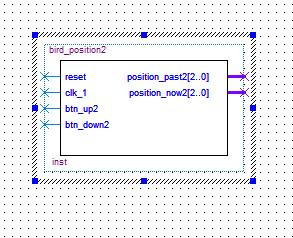


模块功能及接口说明：

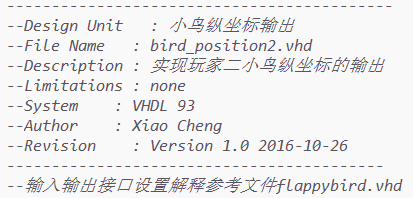


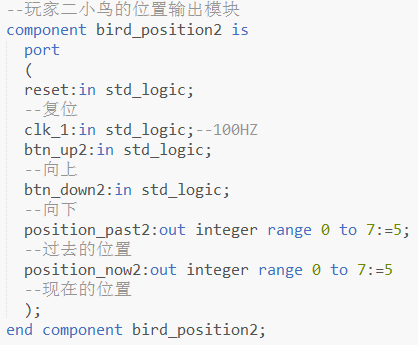


### 2.3.2 bird\_position2 模块

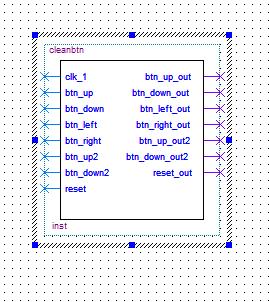


模块功能及接口说明：

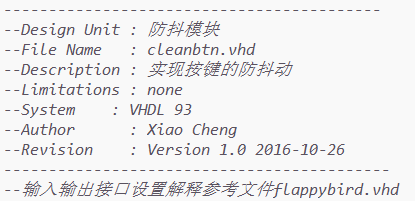




### 2.3.3 cleanbtn 模块

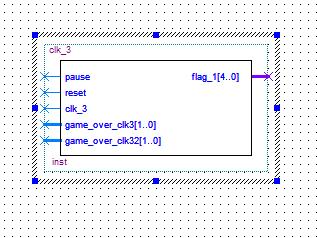


模块功能及接口说明：

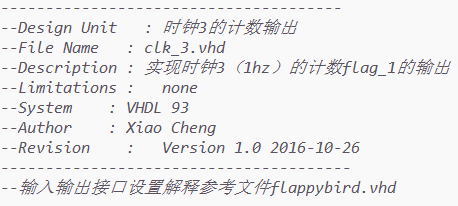


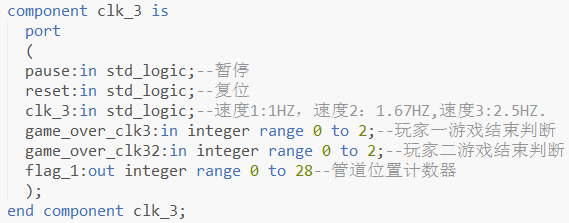


### 2.3.4 clk\_3 模块

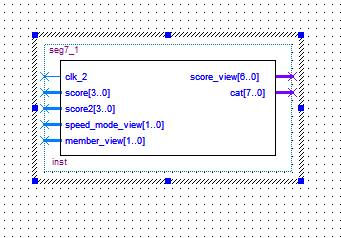


模块功能及接口说明：

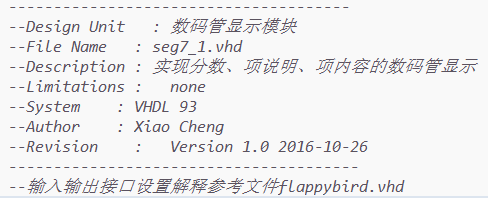


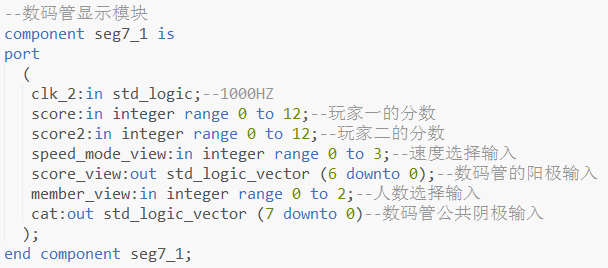


### 2.3.5 seg7\_1 模块

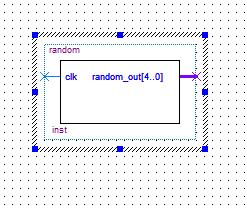


模块功能及接口说明：

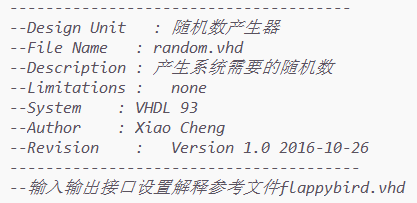


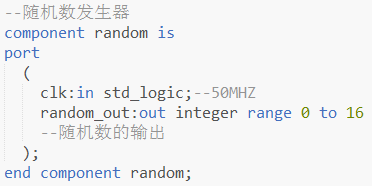


### 2.3.6 random 模块

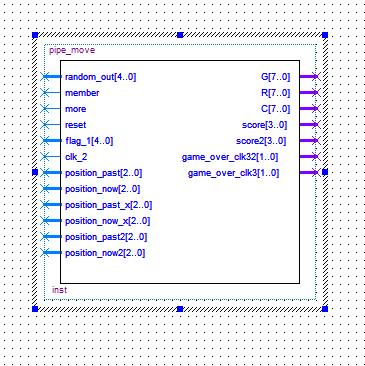


模块功能及接口说明：

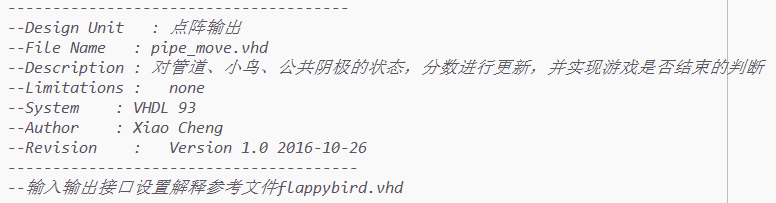




### 2.3.7 pipe\_move 模块

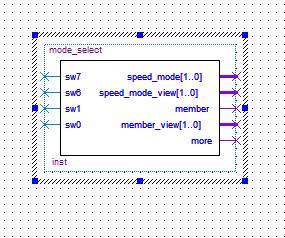


模块功能及接口说明：

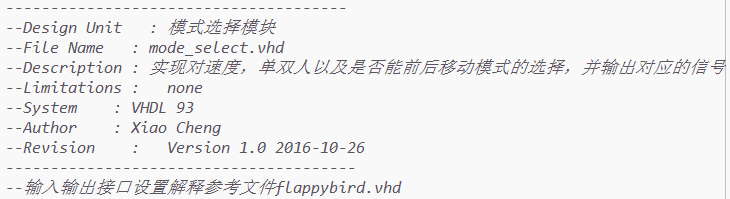
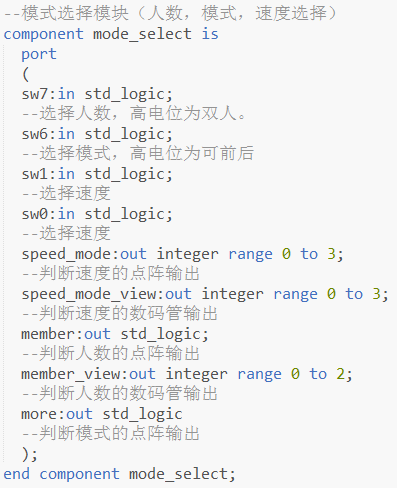




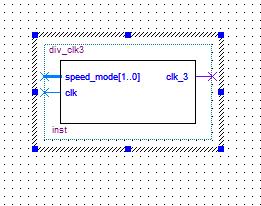
### 2.3.8 mode\_select 模块



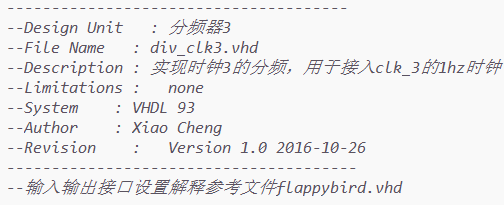
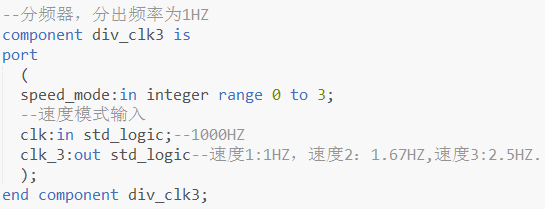
模块功能及接口说明：

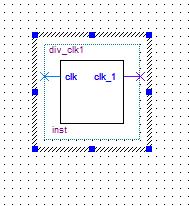
### 2.3.9 div\_clk3 模块



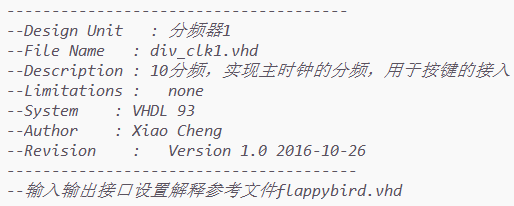
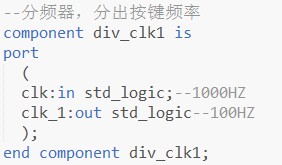
模块功能及接口说明：

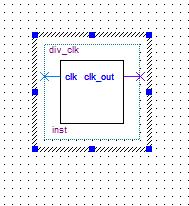
### 2.3.10 div\_clk1 模块



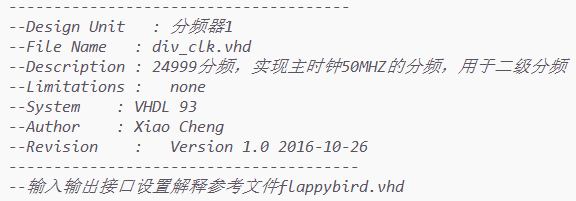
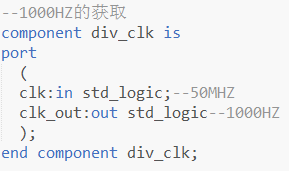
模块功能及接口说明：

### 2.3.11 div\_clk 模块



模块功能及接口说明：

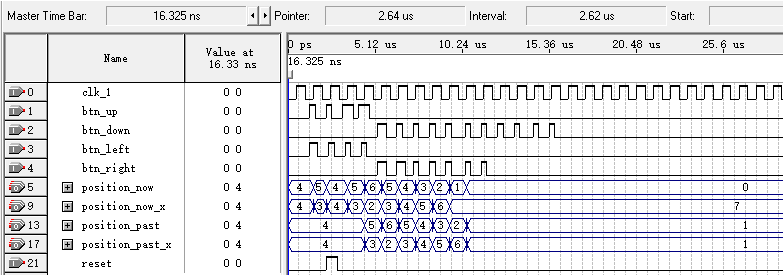
 

1. **仿真波形及波形分析**

由于主时钟频率过大，故采用分模块仿真的方式，这里对小鸟位置模块、点阵输出模块等主要模块进行了仿真，以下是各模块的仿真波形及波形分析。

**3.1 bird\_position**

* 仿真波形：

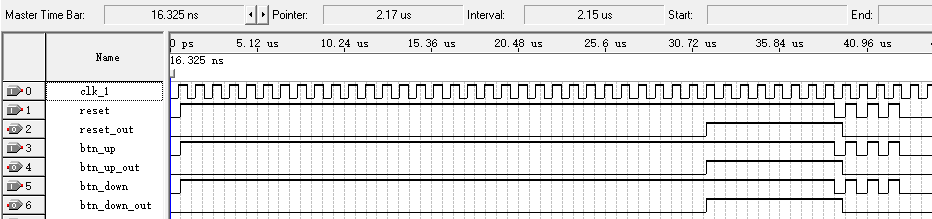


* 波形分析：

如图，当reset信号为1时，position\*全置为初始值4，之后当上升沿时检测到信号btn\*的高电位信号时position\*的位置会做出相应的变化，当position\*位于0或7时，若再输入上、左或下、右信号时position\*的位置不发生变化。

**3.2 cleanbtn**

* 仿真波形：



* 波形分析：

如图，reset、btn\*信号输入之后又连续的几个抖动方波，而对应的输出均为单位方波，从而达到了防抖的效果。

**3.3 seg7\_1**

* 仿真波形：

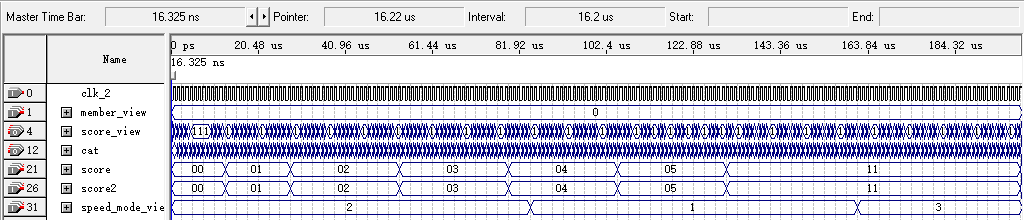


图3.3.1

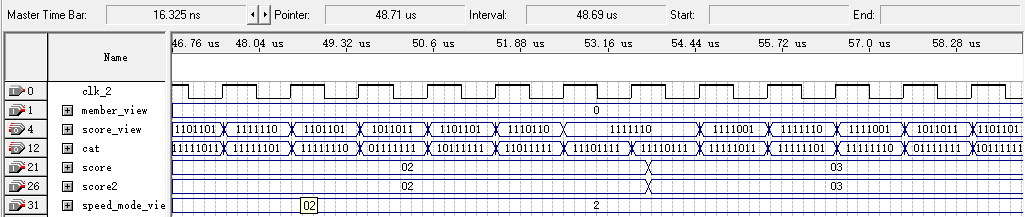


图3.3.2

* 波形分析：

以上，图3.3.1显示了较全面的分数显示仿真，图3.3.2 显示了分数为2和3时数码管的输入情况，如图二，当分数为2，速度为2时，当阴极cat为10111111和11110111时，数码管显示1111110（0），当阴极cat为11111101和01111111时，数码管显示1101101（2）.

**3.4 pipe\_move**

实际上一次管道更新的周期包含多个8行刷新周期，这里为了仿真方便，修改了管道更新的时间，现一个更新周期内只含10次行扫描。

* 仿真波形：

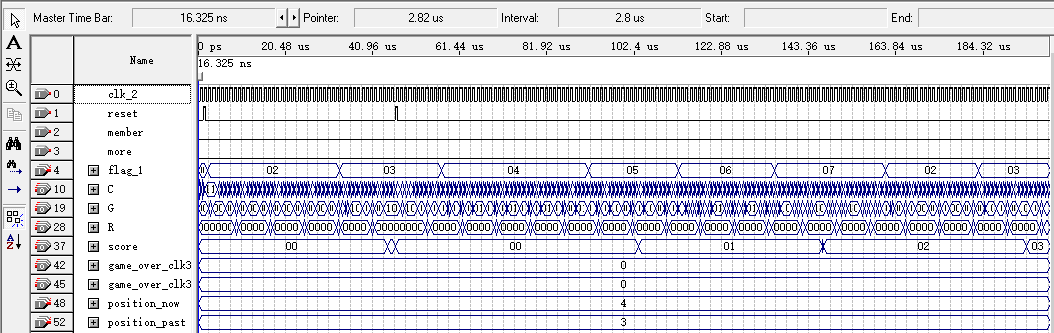


图 3.4.1

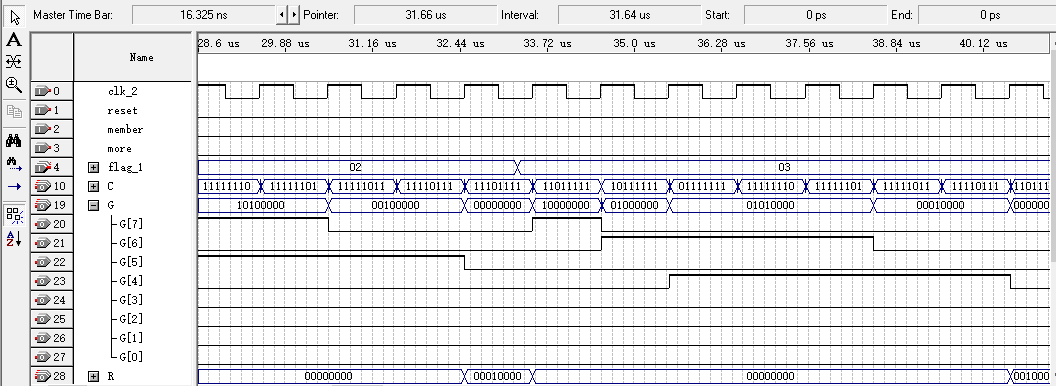


图 3.4.2

* 仿真分析：

以上，图 3.4.1 仿真了0分到3分的情况，图3.4.2 放大2分更新到3分的情形，可以看到，当flag\_1为2时，每个C信号（公共阴极），G信号（绿色阳极）有2个高电平，表明此时有两列管道，同理flag\_1为3时有两列管道，但位置相对之前的向左平移一列。

1. **源程序**

（由于直接复制粘贴源程序故存在一些排版上的问题，建议直接打开源程序批阅^-^）

**------------------------------------------**

**--Design Unit : 小鸟纵坐标输出**

**--File Name : bird\_position.vhd**

**--Description : 实现玩家一小鸟纵坐标的输出**

**--Limitations : none**

**--System : VHDL 93**

**--Author : Xiao Cheng**

**--Revision : Version 1.0 2016-10-26**

**-------------------------------------------**

--输入输出接口设置解释参考文件flappybird.vhd

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity bird\_position is

port

(

reset:in std\_logic;

clk\_1:in std\_logic;--100HZ

btn\_up:in std\_logic;

--向上

btn\_down:in std\_logic;

--向下

btn\_left:in std\_logic;

--向左

btn\_right:in std\_logic;

--向右

position\_past\_x:out integer range 0 to 7:=4;

position\_now\_x:out integer range 0 to 7:=4;

position\_past:out integer range 0 to 7:=4;

position\_now:out integer range 0 to 7:=4

);

end bird\_position;

architecture bird\_position\_arc of bird\_position is

signal position\_p:integer range 0 to 7:=4;

--鸟往纵坐标的信号量

signal position\_n:integer range 0 to 7:=4;

--鸟现纵坐标的信号量

signal position\_p\_x:integer range 0 to 7:=4;

--鸟往横坐标的信号量

signal position\_n\_x:integer range 0 to 7:=4;

--鸟现横坐标的信号量

begin

p:process(btn\_up,btn\_down)

begin

--复位

if reset = '1' then

position\_p\_x <= 4;

position\_n\_x <= 4;

position\_past\_x <= position\_p\_x;

position\_now\_x <= position\_n\_x;

position\_n <= 4;

position\_past <= position\_p;

position\_now <= position\_n;

else

--上升判断

if clk\_1'event and clk\_1 = '1' and btn\_up = '1' then

if position\_n = 7 then position\_n <= 7;position\_p <= 6;

else position\_p <= position\_n;position\_n <= position\_n+1;

end if;

end if;

--下降判断

if clk\_1'event and clk\_1 = '1' and btn\_down = '1' then

if position\_n = 0 then position\_n <= 0;position\_p <= 1;

else position\_p <= position\_n;position\_n <= position\_n-1;

end if;

end if;

--向左判断

if clk\_1'event and clk\_1 = '1' and btn\_left = '1' then

if position\_n\_x = 0 then position\_n\_x <= 0;position\_p\_x <= 1;

else position\_p\_x <= position\_n\_x;position\_n\_x <= position\_n\_x-1;

end if;

end if;

--向右判断

if clk\_1'event and clk\_1 = '1' and btn\_right = '1' then

if position\_n\_x = 7 then position\_n\_x <= 7;position\_p\_x <= 1;

else position\_p\_x <= position\_n\_x;position\_n\_x <= position\_n\_x+1;

end if;

end if;

position\_past <= position\_p;

position\_now <= position\_n;

position\_past\_x <= position\_p\_x;

position\_now\_x <= position\_n\_x;

end if;

end process p;

end bird\_position\_arc;

**-------------------------------------------**

**--Design Unit : 小鸟纵坐标输出**

**--File Name : bird\_position2.vhd**

**--Description : 实现玩家二小鸟纵坐标的输出**

**--Limitations : none**

**--System : VHDL 93**

**--Author : Xiao Cheng**

**--Revision : Version 1.0 2016-10-26**

**------------------------------------------**

--输入输出接口设置解释参考文件flappybird.vhd

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity bird\_position2 is

port

(

reset:in std\_logic;

clk\_1:in std\_logic;--100HZ

btn\_up2:in std\_logic;

btn\_down2:in std\_logic;

position\_past2:out integer range 0 to 7:=5;

position\_now2:out integer range 0 to 7:=5

);

end bird\_position2;

architecture bird\_position\_arc of bird\_position2 is

signal position\_p:integer range 0 to 7:=4;

signal position\_n:integer range 0 to 7:=4;

begin

p:process(btn\_up2,btn\_down2)

begin

--复位

if reset = '1' then

position\_n <= 4;

position\_past2 <= position\_p;

position\_now2 <= position\_n;

else

--向上

if clk\_1'event and clk\_1 = '1' and btn\_up2 = '1' then

if position\_n = 7 then position\_n <= 7;position\_p <= 6;

else position\_p <= position\_n;position\_n <= position\_n+1;

end if;

end if;

--向下

if clk\_1'event and clk\_1 = '1' and btn\_down2 = '1' then

if position\_n = 0 then position\_n <= 0;position\_p <= 1;

else position\_p <= position\_n;position\_n <= position\_n-1;

end if;

end if;

position\_past2 <= position\_p;

position\_now2 <= position\_n;

end if;

end process p;

end bird\_position\_arc;

**------------------------------------------**

**--Design Unit : 防抖模块**

**--File Name : cleanbtn.vhd**

**--Description : 实现按键的防抖动**

**--Limitations : none**

**--System : VHDL 93**

**--Author : Xiao Cheng**

**--Revision : Version 1.0 2016-10-26**

**-------------------------------------------**

--输入输出接口设置解释参考文件flappybird.vhd

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity cleanbtn is

port(

clk\_1: in std\_logic;

--1000HZ

btn\_up: in std\_logic;

btn\_up\_out: out std\_logic;

btn\_down: in std\_logic;

btn\_down\_out: out std\_logic;

btn\_left: in std\_logic;

btn\_left\_out: out std\_logic;

btn\_right: in std\_logic;

btn\_right\_out: out std\_logic;

btn\_up2: in std\_logic;

btn\_up\_out2: out std\_logic;

btn\_down2: in std\_logic;

btn\_down\_out2: out std\_logic;

reset: in std\_logic;

reset\_out: out std\_logic

);

end cleanbtn;

architecture cleanbtn\_arc of cleanbtn is

signal count1: integer range 0 to 40;

signal btn\_temp1: std\_logic;--btn\_up

signal count2: integer range 0 to 40;

signal btn\_temp2: std\_logic;--btn\_down

signal count12: integer range 0 to 40;

signal btn\_temp12: std\_logic;--btn\_up2

signal count22: integer range 0 to 40;

signal btn\_temp22: std\_logic;--btn\_down2

signal count1l: integer range 0 to 40;

signal btn\_temp1l: std\_logic;--btn\_left

signal count2r: integer range 0 to 40;

signal btn\_temp2r: std\_logic;--btn\_right

signal count3: integer range 0 to 40;

signal btn\_temp3: std\_logic;--reset

begin

--玩家一向上的防抖

p1:process(clk\_1)

begin

if clk\_1'event and clk\_1 = '1' then

if btn\_up = '1' then

--当计数器位于30至40时，若检测到信号则输出高电平

if count1 = 40 then

count1 <= count1;

btn\_temp1 <= '0';

elsif count1 >= 30 then

count1 <= count1 + 1;

btn\_temp1 <= '1';

else--其他部分输出低电平

count1 <= count1 + 1;

btn\_temp1 <= '0';

end if;

else

count1 <= 0;

btn\_temp1 <= '0';

end if;

end if;

end process p1;

--玩家一向下的防抖

p2:process(clk\_1)

begin

if clk\_1'event and clk\_1 = '1' then

if btn\_down = '1' then

if count2 = 40 then

count2 <= count2;

btn\_temp2 <= '0';

elsif count2 >= 30 then

count2 <= count2 + 1;

btn\_temp2 <= '1';

else

count2 <= count2 + 1;

btn\_temp2 <= '0';

end if;

else

count2 <= 0;

btn\_temp2 <= '0';

end if;

end if;

end process p2;

--复位的防抖

p3:process(clk\_1)

begin

if clk\_1'event and clk\_1 = '1' then

if reset = '1' then

if count3 = 40 then

count3 <= count3;

btn\_temp3 <= '0';

elsif count3 >= 30 then

count3 <= count3 + 1;

btn\_temp3 <= '1';

else

count3 <= count3 + 1;

btn\_temp3 <= '0';

end if;

else

count3 <= 0;

btn\_temp3 <= '0';

end if;

end if;

end process p3;

--玩家二向上的防抖

p4:process(clk\_1)

begin

if clk\_1'event and clk\_1 = '1' then

if btn\_up2 = '1' then

if count12 = 40 then

count12 <= count12;

btn\_temp12 <= '0';

elsif count12 >= 30 then

count12 <= count12 + 1;

btn\_temp12 <= '1';

else

count12 <= count12 + 1;

btn\_temp12 <= '0';

end if;

else

count12 <= 0;

btn\_temp12 <= '0';

end if;

end if;

end process p4;

--玩家二向下的防抖

p5:process(clk\_1)

begin

if clk\_1'event and clk\_1 = '1' then

if btn\_down2 = '1' then

if count22 = 40 then

count22 <= count22;

btn\_temp22 <= '0';

elsif count22 >= 30 then

count22 <= count22 + 1;

btn\_temp22 <= '1';

else

count22 <= count22 + 1;

btn\_temp22 <= '0';

end if;

else

count22 <= 0;

btn\_temp22 <= '0';

end if;

end if;

end process p5;

--玩家一向左的防抖

p6:process(clk\_1)

begin

if clk\_1'event and clk\_1 = '1' then

if btn\_left = '1' then

if count1l = 40 then

count1l <= count1l;

btn\_temp1l <= '0';

elsif count1l >= 30 then

count1l <= count1l + 1;

btn\_temp1l <= '1';

else

count1l <= count1l + 1;

btn\_temp1l <= '0';

end if;

else

count1l <= 0;

btn\_temp1l <= '0';

end if;

end if;

end process p6;

--玩家一向右的防抖

p7:process(clk\_1)

begin

if clk\_1'event and clk\_1 = '1' then

if btn\_right = '1' then

if count2r = 40 then

count2r <= count2r;

btn\_temp2r <= '0';

elsif count2r >= 30 then

count2r <= count2r + 1;

btn\_temp2r <= '1';

else

count2r <= count2r + 1;

btn\_temp2r <= '0';

end if;

else

count2r <= 0;

btn\_temp2r <= '0';

end if;

end if;

end process p7;

btn\_up\_out <= btn\_temp1;

btn\_down\_out <= btn\_temp2;

btn\_up\_out2 <= btn\_temp12;

btn\_down\_out2 <= btn\_temp22;

btn\_left\_out <= btn\_temp1l;

btn\_right\_out <= btn\_temp2r;

reset\_out <= btn\_temp3;

end cleanbtn\_arc;

**--------------------------------------**

**--Design Unit : 时钟3的计数输出**

**--File Name : clk\_3.vhd**

**--Description : 实现时钟3（1hz）的计数flag\_1的输出**

**--Limitations : none**

**--System : VHDL 93**

**--Author : Xiao Cheng**

**--Revision : Version 1.0 2016-10-26**

**---------------------------------------**

--输入输出接口设置解释参考文件flappybird.vhd

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity clk\_3 is

port

(

pause: in std\_logic;

reset: in std\_logic;

clk\_3:in std\_logic;--速度1:1HZ，速度2：1.67HZ,速度3:2.5HZ.

game\_over\_clk3:in integer range 0 to 2;

game\_over\_clk32:in integer range 0 to 2;

flag\_1:out integer range 0 to 28

);

end clk\_3;

architecture clk\_3\_arc of clk\_3 is

signal flag\_1\_sig:integer range 0 to 28:=0;

--flag\_1的信号量

begin

p:process(clk\_3)

begin

--复位清零

if reset = '1' then

flag\_1\_sig <= 0;

flag\_1 <= 0;

else

if clk\_3'event and clk\_3 = '1' then

if game\_over\_clk3 /= 0 and game\_over\_clk32 /= 0 then flag\_1\_sig <= 0; --游戏结束的复位

elsif pause = '1' then

flag\_1\_sig <= flag\_1\_sig;

else

flag\_1\_sig <= flag\_1\_sig +1;

end if;

end if;

flag\_1 <= flag\_1\_sig;

end if;

end process p;

end clk\_3\_arc;

**--------------------------------------**

**--Design Unit : 分频器1**

**--File Name : div\_clk.vhd**

**--Description : 24999分频，实现主时钟50MHZ的分频，用于二级分频**

**--Limitations : none**

**--System : VHDL 93**

**--Author : Xiao Cheng**

**--Revision : Version 1.0 2016-10-26**

**---------------------------------------**

--输入输出接口设置解释参考文件flappybird.vhd

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity div\_clk is

port

(

clk:in std\_logic;--50MHZ

clk\_out:out std\_logic--1000HZ

);

end div\_clk;

architecture div\_arc of div\_clk is

signal count:integer range 0 to 24999:=0;

signal clk\_1\_sig:std\_logic;

begin

p:process(clk)

begin

if clk'event and clk = '1' then

if count = 24999 then count <=0 ;clk\_1\_sig <= not clk\_1\_sig;

else count <= count+1;

end if;

end if;

clk\_out <= clk\_1\_sig;

end process p;

end div\_arc;

**--------------------------------------**

**--Design Unit : 分频器1**

**--File Name : div\_clk1.vhd**

**--Description : 10分频，实现主时钟的分频，用于按键的接入**

**--Limitations : none**

**--System : VHDL 93**

**--Author : Xiao Cheng**

**--Revision : Version 1.0 2016-10-26**

**---------------------------------------**

--输入输出接口设置解释参考文件flappybird.vhd

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity div\_clk1 is

port

(

clk:in std\_logic;--1000HZ

clk\_1:out std\_logic --100HZ

);

end div\_clk1;

architecture div\_arc of div\_clk1 is

signal count:integer range 0 to 4:=0;

signal clk\_1\_sig:std\_logic;

begin

p:process(clk)

begin

if clk'event and clk = '1' then

if count = 4 then count <=0 ;clk\_1\_sig <= not clk\_1\_sig;

else count <= count+1;

end if;

end if;

clk\_1 <= clk\_1\_sig;

end process p;

end div\_arc;

**--------------------------------------**

**--Design Unit : 分频器3**

**--File Name : div\_clk3.vhd**

**--Description : 实现时钟3的分频，用于接入clk\_3的1hz时钟**

**--Limitations : none**

**--System : VHDL 93**

**--Author : Xiao Cheng**

**--Revision : Version 1.0 2016-10-26**

**---------------------------------------**

--输入输出接口设置解释参考文件flappybird.vhd

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity div\_clk3 is

port

(

speed\_mode:in integer range 0 to 3;

clk:in std\_logic;--1000HZ

clk\_3:out std\_logic

);

end div\_clk3;

architecture div\_arc of div\_clk3 is

signal count:integer range 0 to 499:=0;

signal clk\_3\_sig:std\_logic;

begin

p:process(clk)

begin

if clk'event and clk = '1' then

--不同速度对应不同分频

case speed\_mode is

when 0 =>--1/INF

if count = 899 then count <=0 ;clk\_3\_sig <= not clk\_3\_sig;

else count <= count+1;

end if;

when 1 =>--1HZ

if count = 499 then count <=0 ;clk\_3\_sig <= not clk\_3\_sig;

else count <= count+1;

end if;

when 2 =>--1.67HZ

if count = 299 then count <=0 ;clk\_3\_sig <= not clk\_3\_sig;

else count <= count+1;

end if;

when 3 =>--2.5HZ

if count = 199 then count <=0 ;clk\_3\_sig <= not clk\_3\_sig;

else count <= count+1;

end if;

end case;

end if;

clk\_3 <= clk\_3\_sig;

end process p;

end div\_arc;

**--------------------------------------**

**--Design Unit : 模式选择模块**

**--File Name : mode\_select.vhd**

**--Description : 实现对速度，单双人以及是否能前后移动模式的选择，并输出对应的信号**

**--Limitations : none**

**--System : VHDL 93**

**--Author : Xiao Cheng**

**--Revision : Version 1.0 2016-10-26**

**---------------------------------------**

--输入输出接口设置解释参考文件flappybird.vhd

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity mode\_select is

port

(

sw7:in std\_logic;

sw6:in std\_logic;

sw1:in std\_logic;

sw0:in std\_logic;

speed\_mode:out integer range 0 to 3;

speed\_mode\_view:out integer range 0 to 3;

member:out std\_logic;

--人数

member\_view:out integer range 0 to 2;

more:out std\_logic

--是否能前后

);

end mode\_select;

architecture mode\_select\_arc of mode\_select is

begin

p:process(sw1,sw0)

variable speed\_sig:std\_logic\_vector (0 to 1);

variable sw\_sig:std\_logic\_vector (0 to 1);

begin

speed\_sig := sw1&sw0;

sw\_sig := sw7&sw6;

--速度的判断

case speed\_sig is

when "00" => speed\_mode <= 0;speed\_mode\_view <= 0;

when "01" => speed\_mode <= 1;speed\_mode\_view <= 1;

when "10" => speed\_mode <= 2;speed\_mode\_view <= 2;

when "11" => speed\_mode <= 3;speed\_mode\_view <= 3;

end case;

--人数以及是否前后的判断

case sw\_sig is

when "00" => member <= '0'; more <= '0'; member\_view <= 1;

when "01" => member <= '0'; more <= '1'; member\_view <= 1;

when "10" => member <= '1'; more <= '0'; member\_view <= 2;

when "11" => member <= '1'; more <= '1'; member\_view <= 2;

end case;

end process p;

end mode\_select\_arc;

**--------------------------------------**

**--Design Unit : 点阵输出**

**--File Name : pipe\_move.vhd**

**--Description : 对管道、小鸟、公共阴极的状态，分数进行更新，并实现游戏是否结束的判断**

**--Limitations : none**

**--System : VHDL 93**

**--Author : Xiao Cheng**

**--Revision : Version 1.0 2016-10-26**

**---------------------------------------**

--输入输出接口设置解释参考文件flappybird.vhd

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity pipe\_move is

port

(

random\_out:in integer range 0 to 16;

member:in std\_logic;

more:in std\_logic;

reset:in std\_logic;

flag\_1:in integer range 0 to 28;

clk\_2:in std\_logic;--1000HZ

position\_past:in integer range 0 to 7:=4;

position\_now:in integer range 0 to 7:=4;

position\_past\_x:in integer range 0 to 7:=4;

position\_now\_x:in integer range 0 to 7:=4;

position\_past2:in integer range 0 to 7:=4;

position\_now2:in integer range 0 to 7:=4;

G:out std\_logic\_vector (7 downto 0);

R:out std\_logic\_vector (7 downto 0);

C:out std\_logic\_vector (7 downto 0);

score:out integer range 0 to 12;

score2:out integer range 0 to 12;

game\_over\_clk32:out integer range 0 to 2;

game\_over\_clk3:out integer range 0 to 2

);

end pipe\_move;

architecture pipe\_move\_arc of pipe\_move is

type matrix\_index is array (16 downto 0) of integer range 0 to 5;

signal random1:matrix\_index:=(1,2,4,3,2,0,2,3,0,2,4,5,1,3,4,2,1);

--自定义一个17位的数组，填放管子的空缺位置初始值

signal random:matrix\_index;

signal G\_sig:std\_logic\_vector (7 downto 0);

signal R\_sig:std\_logic\_vector (7 downto 0);

signal C\_sig:std\_logic\_vector (7 downto 0);

signal count:integer range -1 to 7:=-1;

--计数器，用来实现行扫描

signal count\_p:integer range -1 to 7:=-1;

--存放count计数器前一状态的值

signal flag\_past:integer range 0 to 28;

--存放flag\_1前一状态的值

signal game\_over\_sig:integer range 0 to 2;

signal game\_over\_sig2:integer range 0 to 2;

signal score\_sig:integer range 0 to 13:=0;

signal score\_sig2:integer range 0 to 13:=0;

signal count\_s1:integer range 0 to 20:=0;

--微型计数器，辅助实现分数判断

signal position\_px\_sig:integer range 0 to 7;

signal position\_nx\_sig:integer range 0 to 7;

signal count\_r:integer range 0 to 17;

-- signal random\_count:integer range 0 to 113;

begin

p1:process(clk\_2,reset)

begin

--复位回初始状态

if reset = '1' then

C <= "11111111";

count <= -1;

game\_over\_sig <= 0;

game\_over\_sig2 <= 0;

count\_s1 <= 0;

score\_sig <= 0;

score\_sig2 <= 0;

count\_p <= -1;

flag\_past <= 0;

game\_over\_clk3 <= game\_over\_sig;

game\_over\_clk32 <= game\_over\_sig2;

score2 <= score\_sig2;

score <= score\_sig;

count\_r <= 0;

else

if clk\_2'event and clk\_2 = '1' then

------------------------------------更新随机数-----------------------------------------

if count\_r = 17 then count\_r <= 17;

else count\_r <= count\_r+1;

end if;

if count\_r /= 17 then

random (count\_r) <= random1(random\_out);

end if;

------------------------------点阵阴极状态的更新----------------------------------

--管子的每一次移动便对计数器count进行重新计数

if flag\_1 /= flag\_past then count <= -1;flag\_past <= flag\_1;

end if;

--计数器状态更新

if count = 7 then count <= 0;

else count <= count+1;

end if;

--点阵阴极状态的更新

for i in 0 to 7 loop

if i = count then C\_sig(i) <= '0';

else C\_sig(i) <= '1';

end if;

end loop;

C <= C\_sig;

------------------------更新管道的位置（较难理解）------------------------------|

--每次刷新一行时先对管道进行清空初始化

if count /= count\_p then

for i in 0 to 7 loop

G\_sig(i) <= '0';

end loop;

count\_p <= count;

end if;

-----------------------打印正在刷新的这一行所对应的管道---------------------

for i in 0 to 16 loop

if 7-flag\_1+2\*i >= 0 and 7-flag\_1+2\*i <= 7 then

if count /= random(i) and count /= random(i)+1 and count /= random(i)+2 then

G\_sig(7-flag\_1+2\*i) <= '1';

else

G\_sig(7-flag\_1+2\*i) <= '0';

end if;

end if;

end loop;

G <= G\_sig;

--------------------------------点阵红点的更新及游戏状态判断---------------------------------|

if member = '0' then--当只有一个玩家时

if more = '1' then--当可以前后移动时，引入鸟的横坐标

position\_px\_sig <= position\_past\_x;

position\_nx\_sig <= position\_now\_x;

else

position\_px\_sig <= 4;

position\_nx\_sig <= 4;

end if;

if game\_over\_sig = 2 then--游戏失败

case count is

when 7 => R\_sig <= "10000001";G\_sig <= "00000000";

when 6 => R\_sig <= "01000010";G\_sig <= "00000000";

when 5 => R\_sig <= "00100100";G\_sig <= "00000000";

when 4 => R\_sig <= "00011000";G\_sig <= "00000000";

when 3 => R\_sig <= "00011000";G\_sig <= "00000000";

when 2 => R\_sig <= "00100100";G\_sig <= "00000000";

when 1 => R\_sig <= "01000010";G\_sig <= "00000000";

when 0 => R\_sig <= "10000001";G\_sig <= "00000000";

when others => R\_sig <= "00000000";G\_sig <= "00000000";

end case;

R <= R\_sig;

G <= G\_sig;

elsif game\_over\_sig = 1 then--游戏通关

case count is

when 7 => R\_sig <= "00000000";G\_sig <= "10000001";

when 6 => R\_sig <= "00000000";G\_sig <= "10000001";

when 5 => R\_sig <= "00000000";G\_sig <= "10000001";

when 4 => R\_sig <= "00000000";G\_sig <= "10000001";

when 3 => R\_sig <= "00000000";G\_sig <= "10000001";

when 2 => R\_sig <= "00000000";G\_sig <= "01000010";

when 1 => R\_sig <= "00000000";G\_sig <= "00100100";

when 0 => R\_sig <= "00000000";G\_sig <= "00011000";

when others => R\_sig <= "00000000";G\_sig <= "00000000";

end case;

R <= R\_sig;

G <= G\_sig;

else--游戏继续进行时

-----------------------------------小鸟位置的更新------------

if count = position\_past then R\_sig(position\_px\_sig) <= '0';

end if;

if count = position\_now then R\_sig(position\_nx\_sig) <= '1';

else

R\_sig <= "00000000";

end if;

R <= R\_sig;

---------------------------（较难）游戏分数以及游戏状态的更新--------------------------

--微型计数器count\_s1的更新，从1计数到20，到20作停留

if count\_s1 = 20 then count\_s1 <= 20;

else count\_s1 <= count\_s1 + 1;

end if;

if score\_sig = 12 then game\_over\_sig <= 1;

else

if game\_over\_sig = 2 then score\_sig <= score\_sig;

else

if flag\_1 >= 3-(position\_nx\_sig-4) and flag\_1 rem 2 = (1-(position\_nx\_sig rem 2)) then

if flag\_1 /= flag\_past then count\_s1 <= 0;flag\_past <= flag\_1;

end if;

if position\_now = 7 then

if count = position\_now then

if G\_sig(position\_nx\_sig) = '1' then

--判断微型计数器是否记录完了一个周期的完整的行扫描（从第一行扫到第八行）

if count\_s1 >= 8 then score\_sig <= score\_sig+1;game\_over\_sig <= 2;

else score\_sig <= score\_sig;game\_over\_sig <= 2;

end if;

end if;

end if;

else

if count = position\_now+1 then

if G\_sig(position\_nx\_sig) = '1' then

--判断微型计数器是否记录完了一个周期的完整的行扫描（从第一行扫到第八行）

if count\_s1 >= 8 then score\_sig <= score\_sig+1;game\_over\_sig <= 2;

else score\_sig <= score\_sig;game\_over\_sig <= 2;

end if;

end if;

end if;

end if;

if game\_over\_sig = 0 then

--判断微型计数器是否记录完了一个周期的完整的行扫描（从第一行扫到第八行），在扫下一行时进行分数判断

if count\_s1 = 9 then

score\_sig <= score\_sig+1;game\_over\_sig <= 0;

end if;

end if;

end if;

end if;

end if;

game\_over\_clk3 <= game\_over\_sig;

score <= score\_sig;

end if;

end if;

if member = '1' then--当有两个玩家同时进行游戏时

if game\_over\_sig /= 0 then--玩家一游戏结束时

if game\_over\_sig2 /=0 then

--玩家一和玩家二同时游戏结束

--比较两个玩家的分数，进行胜负判断

if score\_sig > score\_sig2 then

case count is

when 7 => R\_sig <= "00000000";G\_sig <= "01001110";

when 6 => R\_sig <= "00000000";G\_sig <= "01101010";

when 5 => R\_sig <= "00000000";G\_sig <= "01001110";

when 4 => R\_sig <= "00000000";G\_sig <= "01000010";

when 3 => R\_sig <= "00000000";G\_sig <= "11100010";

when 2 => R\_sig <= "00000000";G\_sig <= "01000010";

when 1 => R\_sig <= "00000000";G\_sig <= "00100100";

when 0 => R\_sig <= "00000000";G\_sig <= "00011000";

when others => R\_sig <= "00000000";G\_sig <= "00000000";

end case;

R <= R\_sig;

G <= G\_sig;

elsif score\_sig < score\_sig2 then

case count is

when 7 => R\_sig <= "00000000";G\_sig <= "01111110";

when 6 => R\_sig <= "00000000";G\_sig <= "01001010";

when 5 => R\_sig <= "00000000";G\_sig <= "01111110";

when 4 => R\_sig <= "00000000";G\_sig <= "00010010";

when 3 => R\_sig <= "00000000";G\_sig <= "01110010";

when 2 => R\_sig <= "00000000";G\_sig <= "01000010";

when 1 => R\_sig <= "00000000";G\_sig <= "00100100";

when 0 => R\_sig <= "00000000";G\_sig <= "00011000";

when others => R\_sig <= "00000000";G\_sig <= "00000000";

end case;

R <= R\_sig;

G <= G\_sig;

else

case count is

when 7 => R\_sig <= "00000000";G\_sig <= "11111111";

when 6 => R\_sig <= "00000000";G\_sig <= "11111111";

when 5 => R\_sig <= "00000000";G\_sig <= "00011000";

when 4 => R\_sig <= "00000000";G\_sig <= "00011000";

when 3 => R\_sig <= "00000000";G\_sig <= "00011000";

when 2 => R\_sig <= "00000000";G\_sig <= "00011000";

when 1 => R\_sig <= "00000000";G\_sig <= "00011000";

when 0 => R\_sig <= "00000000";G\_sig <= "00011000";

when others => R\_sig <= "00000000";G\_sig <= "00000000";

end case;

R <= R\_sig;

G <= G\_sig;

end if;

elsif game\_over\_sig2 = 0 then--只有玩家二游戏进行

--更新小鸟

if count = position\_past2 then R\_sig(4) <= '0';

end if;

if count = position\_now2 then R\_sig(4) <= '1';

else

for i in 0 to 7 loop

R\_sig(i) <= '0';

end loop;

end if;

R <= R\_sig;

--分数判断

if count\_s1 = 20 then count\_s1 <= 20;

else count\_s1 <= count\_s1 + 1;

end if;

if score\_sig2 = 12 then game\_over\_sig2 <= 1;

else

if game\_over\_sig2 = 2 then score\_sig2 <= score\_sig2;

else

if flag\_1 >= 3 and flag\_1 rem 2 = 1 then

if flag\_1 /= flag\_past then count\_s1 <= 0;flag\_past <= flag\_1;

end if;

if count = position\_now2+1 or (count = 7 and position\_now2 = 7)then

if G\_sig(4) = '1' then

if count\_s1 >= 8 then score\_sig2 <= score\_sig2+1;game\_over\_sig2 <= 2;

else score\_sig2 <= score\_sig2;game\_over\_sig2 <= 2;

end if;

end if;

end if;

if game\_over\_sig2 = 0 then

if count\_s1 = 9 then

score\_sig2 <= score\_sig2+1;game\_over\_sig2 <= 0;

end if;

end if;

end if;

end if;

end if;

game\_over\_clk32 <= game\_over\_sig2;

score2 <= score\_sig2;

game\_over\_clk3 <= game\_over\_sig;

score <= score\_sig;

end if;

elsif game\_over\_sig = 0 then--玩家一游戏进行

if game\_over\_sig2 /= 0 then--玩家二游戏结束，只有玩家一进行游戏

--更新小鸟

if count = position\_past then R\_sig(4) <= '0';

end if;

if count = position\_now then R\_sig(4) <= '1';

else

for i in 0 to 7 loop

R\_sig(i) <= '0';

end loop;

end if;

R <= R\_sig;

--玩家一分数判断

if count\_s1 = 20 then count\_s1 <= 20;

else count\_s1 <= count\_s1 + 1;

end if;

if score\_sig = 12 then game\_over\_sig <= 1;

else

if game\_over\_sig = 2 then score\_sig <= score\_sig;

else

if flag\_1 >= 3 and flag\_1 rem 2 = 1 then

if flag\_1 /= flag\_past then count\_s1 <= 0;flag\_past <= flag\_1;

end if;

if count = position\_now+1 or (count = 7 and position\_now = 7)then

if G\_sig(4) = '1' then

if count\_s1 >= 8 then score\_sig <= score\_sig+1;game\_over\_sig <= 2;

else score\_sig <= score\_sig;game\_over\_sig <= 2;

end if;

end if;

end if;

if game\_over\_sig = 0 then

if count\_s1 = 9 then

score\_sig <= score\_sig+1;game\_over\_sig <= 0;

end if;

end if;

end if;

end if;

end if;

game\_over\_clk3 <= game\_over\_sig;

score <= score\_sig;

game\_over\_clk32 <= game\_over\_sig2;

score2 <= score\_sig2;

elsif game\_over\_sig2 = 0 then--玩家一和玩家二同时进行游戏

--更新小鸟

if count = position\_past then R\_sig(4) <= '0';

end if;

if count = position\_past2 then R\_sig(4) <= '0';

end if;

if count = position\_now or count = position\_now2 then R\_sig(4) <= '1';

else

R\_sig <= "00000000";

end if;

R <= R\_sig;

--更新玩家一分数

if count\_s1 = 20 then count\_s1 <= 20;

else count\_s1 <= count\_s1 + 1;

end if;

if score\_sig = 12 then game\_over\_sig <= 1;

else

if game\_over\_sig = 2 then score\_sig <= score\_sig;

else

if flag\_1 >= 3 and flag\_1 rem 2 = 1 then

if flag\_1 /= flag\_past then count\_s1 <= 0;flag\_past <= flag\_1;

end if;

if count = position\_now+1 or (count = 7 and position\_now = 7)then

if G\_sig(4) = '1' then

if count\_s1 >= 8 then score\_sig <= score\_sig+1;game\_over\_sig <= 2;

else score\_sig <= score\_sig;game\_over\_sig <= 2;

end if;

end if;

end if;

if game\_over\_sig = 0 then

if count\_s1 = 9 then

score\_sig <= score\_sig+1;game\_over\_sig <= 0;

end if;

end if;

end if;

end if;

end if;

game\_over\_clk3 <= game\_over\_sig;

score <= score\_sig;

--更新玩家二分数

if score\_sig2 = 12 then game\_over\_sig2 <= 1;

else

if game\_over\_sig2 = 2 then score\_sig2 <= score\_sig2;

else

if flag\_1 >= 3 and flag\_1 rem 2 = 1 then

if flag\_1 >= 3 and flag\_1 rem 2 = 1 then

end if;

if count = position\_now2+1 or (count = 7 and position\_now2 = 7) then

if G\_sig(4) = '1' then

if count\_s1 >= 8 then score\_sig2 <= score\_sig2+1;game\_over\_sig2 <= 2;

else score\_sig2 <= score\_sig2;game\_over\_sig2 <= 2;

end if;

end if;

end if;

if game\_over\_sig2 = 0 then

if count\_s1 = 9 then

score\_sig2 <= score\_sig2+1;game\_over\_sig2 <= 0;

end if;

end if;

end if;

end if;

end if;

game\_over\_clk32 <= game\_over\_sig2;

score2 <= score\_sig2;

end if;

end if;

end if;

end if;

end if;

end process p1;

--p2:process(score\_sig,score\_sig2,clk\_2)

--begin

-- if score\_sig'event or score\_sig2'event then

-- beep <= clk\_f;

--else beep <= '0';

--end if;

--end process p2;

end pipe\_move\_arc;

**--------------------------------------**

**--Design Unit : 随机数产生器**

**--File Name : random.vhd**

**--Description : 产生系统需要的随机数**

**--Limitations : none**

**--System : VHDL 93**

**--Author : Xiao Cheng**

**--Revision : Version 1.0 2016-10-26**

**---------------------------------------**

--输入输出接口设置解释参考文件flappybird.vhd

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity random is

port

(

clk:in std\_logic;--50MHZ

random\_out:out integer range 0 to 16

);

end random;

architecture random\_arc of random is

signal random\_sig:integer range 0 to 16;

signal count:integer range 0 to 22;

begin

p:process(clk)

begin

if clk'event and clk = '1' then

if count = 22 then count <= 0;

else count <= count+1;

end if;

if count = 22 then

if random\_sig = 16 then random\_sig <= 0;

else random\_sig <= random\_sig +1;

end if;

end if;

end if;

random\_out <= random\_sig;

end process p;

end random\_arc;

**--------------------------------------**

**--Design Unit : 数码管显示模块**

**--File Name : seg7\_1.vhd**

**--Description : 实现分数、项说明、项内容的数码管显示**

**--Limitations : none**

**--System : VHDL 93**

**--Author : Xiao Cheng**

**--Revision : Version 1.0 2016-10-26**

**---------------------------------------**

--输入输出接口设置解释参考文件flappybird.vhd

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity seg7\_1 is

port

(

clk\_2:in std\_logic;--1000HZ

score:in integer range 0 to 12;

score2:in integer range 0 to 12;

speed\_mode\_view:in integer range 0 to 3;

score\_view:out std\_logic\_vector (6 downto 0);

member\_view:in integer range 0 to 2;

cat:out std\_logic\_vector (7 downto 0)

);

end seg7\_1;

architecture seg7\_1\_arc of seg7\_1 is

signal count:integer range 0 to 7:=0;

--计数器

signal cat\_sig:std\_logic\_vector (7 downto 0);

--公共阴极的信号量

begin

p:process (clk\_2,score)

begin

--计数状态的更新，每次只有一个阴极有效

if clk\_2'event and clk\_2 = '1' then

if count = 7 then count <= 0;

else count <= count +1;

end if;

for i in 0 to 7 loop

if i = 7-count then cat\_sig(i) <= '0';

else cat\_sig(i) <= '1';

end if;

end loop;

for i in 0 to 7 loop

cat(i) <= cat\_sig(i);

end loop;

--判断计数状态，从而使得对应1~8数码管显示对应内容

case count is

--第7个管子显示玩家一分数的十位

when 7 =>

case score is

when 0 => score\_view <= "1111110";

when 1 => score\_view <= "1111110";

when 2 => score\_view <= "1111110";

when 3 => score\_view <= "1111110";

when 4 => score\_view <= "1111110";

when 5 => score\_view <= "1111110";

when 6 => score\_view <= "1111110";

when 7 => score\_view <= "1111110";

when 8 => score\_view <= "1111110";

when 9 => score\_view <= "1111110";

when 10 => score\_view <= "0110000";

when 11 => score\_view <= "0110000";

when 12 => score\_view <= "0110000";

end case;

--第八个管子显示玩家一分数的各位

when 0 =>

case score is

when 0 => score\_view <= "1111110";

when 1 => score\_view <= "0110000";

when 2 => score\_view <= "1101101";

when 3 => score\_view <= "1111001";

when 4 => score\_view <= "0110011";

when 5 => score\_view <= "1011011";

when 6 => score\_view <= "1011111";

when 7 => score\_view <= "1110000";

when 8 => score\_view <= "1111111";

when 9 => score\_view <= "1111011";

when 10 => score\_view <= "1111110";

when 11 => score\_view <= "0110000";

when 12 => score\_view <= "1101101";

end case;

--第二个管子显示游戏速度

when 2 =>

case speed\_mode\_view is

when 0 => score\_view <= "1111110";

when 1 => score\_view <= "0110000";

when 2 => score\_view <= "1101101";

when 3 => score\_view <= "1111001";

end case;

--第五个管子显示玩家二分数的十位

when 5 =>

case score2 is

when 0 => score\_view <= "1111110";

when 1 => score\_view <= "1111110";

when 2 => score\_view <= "1111110";

when 3 => score\_view <= "1111110";

when 4 => score\_view <= "1111110";

when 5 => score\_view <= "1111110";

when 6 => score\_view <= "1111110";

when 7 => score\_view <= "1111110";

when 8 => score\_view <= "1111110";

when 9 => score\_view <= "1111110";

when 10 => score\_view <= "0110000";

when 11 => score\_view <= "0110000";

when 12 => score\_view <= "0110000";

end case;

--第六个管子显示玩家二分数的个位

when 6 =>

case score2 is

when 0 => score\_view <= "1111110";

when 1 => score\_view <= "0110000";

when 2 => score\_view <= "1101101";

when 3 => score\_view <= "1111001";

when 4 => score\_view <= "0110011";

when 5 => score\_view <= "1011011";

when 6 => score\_view <= "1011111";

when 7 => score\_view <= "1110000";

when 8 => score\_view <= "1111111";

when 9 => score\_view <= "1111011";

when 10 => score\_view <= "1111110";

when 11 => score\_view <= "0110000";

when 12 => score\_view <= "1101101";

end case;

--第一个管子，显示形状“S”，表示是速度标志

when 1 =>

score\_view <= "1011011";

--第三个管子显示形状“n”，表示是人数标志

when 3 =>

score\_view <= "1110110";

--第四个管子显示游戏人数

when 4 =>

if member\_view = 1 then score\_view <= "0110000";

elsif member\_view = 2 then score\_view <= "1101101";

else score\_view <= "1111110";

end if;

end case;

end if;

end process p;

end seg7\_1\_arc;

**--------------------------------------**

**--Design Unit : 各模块的链接**

**--File Name : flappybird.vhd**

**--Description : 实现各模块的链接操作**

**--Limitations : none**

**--System : VHDL 93**

**--Author : Xiao Cheng**

**--Revision : Version 1.0 2016-10-26**

**---------------------------------------**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity flappybird is

port

(

pause:in std\_logic;

--sw4 暂停开关接入（sw拨码开关）

sw7:in std\_logic;

--sw7 游戏人数开关接入

sw6:in std\_logic;

--sw6 游戏模式开关接入

sw1:in std\_logic;

--sw1&sw0 控制速度开关接入

sw0:in std\_logic;

reset:in std\_logic;

--btn7 重置复位按键接入（btn按键）

clk:in std\_logic;

--时钟接入50MHZ

btn\_up:in std\_logic;

--btn0 玩家一向上按键接入

btn\_down:in std\_logic;

--btn1 玩家一向下按键接入

btn\_left:in std\_logic;

--btn3 玩家一向左按键接入

btn\_right:in std\_logic;

--btn2 玩家一向右按键接入

btn\_up2:in std\_logic;

--btn4 玩家二向上按键接入

btn\_down2:in std\_logic;

--btn5 玩家二向下按键接入

G:out std\_logic\_vector (7 downto 0);

--管道点阵显示向量输出

R:out std\_logic\_vector (7 downto 0);

--小鸟显示向量输出

C:out std\_logic\_vector (7 downto 0);

--公共阴极向量输出

score\_view:out std\_logic\_vector (6 downto 0);

--数码管阳极向量输出

cat:out std\_logic\_vector (7 downto 0)

--数码管公共阴极向量输出

);

end flappybird;

architecture flappy\_bird\_arc of flappybird is

--按键防抖模块

component cleanbtn is

port(

clk\_1: in std\_logic;--1000HZ

btn\_up: in std\_logic;

--玩家一向上按键输入

btn\_up\_out: out std\_logic;

--玩家一向上按键输出

btn\_down: in std\_logic;

--玩家一向下按键输入

btn\_down\_out: out std\_logic;

--玩家一向下按键输出

btn\_left: in std\_logic;

--玩家一向左按键输入

btn\_left\_out: out std\_logic;

--玩家一向左按键输出

btn\_right: in std\_logic;

--玩家一向右按键输入

btn\_right\_out: out std\_logic;

--玩家一向右按键输出

btn\_up2: in std\_logic;

--玩家二向上按键输入

btn\_up\_out2: out std\_logic;

--玩家二向上按键输出

btn\_down2: in std\_logic;

--玩家二向下按键输入

btn\_down\_out2: out std\_logic;

--玩家二向下按键输出

reset: in std\_logic;

--复位按键输入

reset\_out: out std\_logic

--复位按键输出

);

end component cleanbtn;

--玩家一小鸟的位置输出模块

component bird\_position is

port

(

reset:in std\_logic;

clk\_1:in std\_logic;--100HZ

btn\_up:in std\_logic;

btn\_down:in std\_logic;

btn\_left:in std\_logic;

btn\_right:in std\_logic;

position\_past\_x:out integer range 0 to 7:=4;

--小鸟移动前的横向位置

position\_now\_x:out integer range 0 to 7:=4;

--小鸟移动后的横向位置

position\_past:out integer range 0 to 7:=4;

--小鸟移动前的纵向位置

position\_now:out integer range 0 to 7:=4

--小鸟移动后的纵向位置

);

end component bird\_position;

--玩家二小鸟的位置输出模块

component bird\_position2 is

port

(

reset:in std\_logic;

clk\_1:in std\_logic;--100HZ

btn\_up2:in std\_logic;

btn\_down2:in std\_logic;

position\_past2:out integer range 0 to 7:=5;

position\_now2:out integer range 0 to 7:=5

);

end component bird\_position2;

--模式选择模块（人数，模式，速度选择）

component mode\_select is

port

(

sw7:in std\_logic;

sw6:in std\_logic;

sw1:in std\_logic;

sw0:in std\_logic;

speed\_mode:out integer range 0 to 3;

--判断速度的点阵输出

speed\_mode\_view:out integer range 0 to 3;

--判断速度的数码管输出

member:out std\_logic;

--判断人数的点阵输出

member\_view:out integer range 0 to 2;

--判断人数的数码管输出

more:out std\_logic

--判断模式的点阵输出

);

end component mode\_select;

--管道，小鸟，点阵公共阴极向量的更新以及碰撞判断和分数更新

component pipe\_move is

port

(

random\_out:in integer range 0 to 16;

member:in std\_logic;

more:in std\_logic;

reset:in std\_logic;

flag\_1:in integer range 0 to 28;

--记录时钟clk3的计数器

clk\_2:in std\_logic;--1000HZ

position\_past:in integer range 0 to 7:=4;

position\_now:in integer range 0 to 7:=4;

position\_past\_x:in integer range 0 to 7:=4;

position\_now\_x:in integer range 0 to 7:=4;

position\_past2:in integer range 0 to 7:=4;

position\_now2:in integer range 0 to 7:=4;

G:out std\_logic\_vector (7 downto 0);

R:out std\_logic\_vector (7 downto 0);

C:out std\_logic\_vector (7 downto 0);

score:out integer range 0 to 12;

--玩家一的分数输出

score2:out integer range 0 to 12;

--玩家二的分数输出

game\_over\_clk32:out integer range 0 to 2;

--玩家二的游戏结束判断输出

game\_over\_clk3:out integer range 0 to 2

--玩家一的游戏结束判断输出

);

end component pipe\_move;

--1000HZ的获取

component div\_clk is

port

(

clk:in std\_logic;--50MHZ

clk\_out:out std\_logic--1000HZ

);

end component div\_clk;

--随机数发生器

component random is

port

(

clk:in std\_logic;--50MHZ

random\_out:out integer range 0 to 16

--随机数的输出

);

end component random;

--分频器，分出频率为1HZ

component div\_clk3 is

port

(

speed\_mode:in integer range 0 to 3;

clk:in std\_logic;--1000HZ

clk\_3:out std\_logic--速度1:1HZ，速度2：1.67HZ,速度3:2.5HZ.

);

end component div\_clk3;

--分频器，分出按键频率

component div\_clk1 is

port

(

clk:in std\_logic;--1000HZ

clk\_1:out std\_logic--100HZ

);

end component div\_clk1;

component clk\_3 is

port

(

pause:in std\_logic;

reset:in std\_logic;

clk\_3:in std\_logic;--速度1:1HZ，速度2：1.67HZ,速度3:2.5HZ.

game\_over\_clk3:in integer range 0 to 2;

game\_over\_clk32:in integer range 0 to 2;

flag\_1:out integer range 0 to 28

);

end component clk\_3;

--数码管显示模块

component seg7\_1 is

port

(

clk\_2:in std\_logic;--1000HZ

score:in integer range 0 to 12;

score2:in integer range 0 to 12;

speed\_mode\_view:in integer range 0 to 3;

score\_view:out std\_logic\_vector (6 downto 0);

member\_view:in integer range 0 to 2;

cat:out std\_logic\_vector (7 downto 0)

);

end component seg7\_1;

--\*\_sig\* 表示各自的信号量

signal clk\_out\_sig:std\_logic;

signal clk\_3\_sig1:std\_logic;

signal flag\_1\_sig2:integer range 0 to 28;

signal game\_over\_clk3\_sig3:integer range 0 to 2;

signal game\_over\_clk32\_sig3:integer range 0 to 2;

signal clk\_1\_sig7:std\_logic;

signal position\_past\_sig8:integer range 0 to 7;

signal position\_past2\_sig8:integer range 0 to 7;

signal postion\_past\_x\_sig:integer range 0 to 7;

signal postion\_now\_x\_sig:integer range 0 to 7;

signal position\_now\_sig10:integer range 0 to 7;

signal position\_now2\_sig10:integer range 0 to 7;

signal score\_sig13:integer range 0 to 12;

signal score2\_sig13:integer range 0 to 12;

signal clk\_4\_sig15:std\_logic;

signal speed\_mode\_sig:integer range 0 to 3;

signal speed\_mode\_view\_sig:integer range 0 to 3;

signal member\_sig:std\_logic;

signal more\_sig:std\_logic;

signal memberv\_sig:integer range 0 to 2;

signal clk4\_sig:std\_logic;

signal random\_out\_sig:integer range 0 to 16;

signal btn\_up\_sig:std\_logic;

signal btn\_down\_sig:std\_logic;

signal btn\_up\_sig2:std\_logic;

signal btn\_down\_sig2:std\_logic;

signal btn\_left\_sig:std\_logic;

signal btn\_right\_sig:std\_logic;

signal reset\_sig:std\_logic;

begin

--模块间的连接

u0:div\_clk port map (clk,clk\_out\_sig);

u1:div\_clk3 port map(speed\_mode\_sig,clk\_out\_sig,clk\_3\_sig1);

u2:clk\_3 port map(pause,reset\_sig,clk\_3\_sig1,game\_over\_clk3\_sig3,game\_over\_clk32\_sig3,flag\_1\_sig2);

u3:pipe\_move port map(random\_out\_sig,member\_sig,more\_sig,reset\_sig,flag\_1\_sig2,clk\_out\_sig,position\_past\_sig8,position\_now\_sig10,postion\_past\_x\_sig,postion\_now\_x\_sig,position\_past2\_sig8,position\_now2\_sig10,G,R,C,score\_sig13,score2\_sig13,game\_over\_clk32\_sig3,game\_over\_clk3\_sig3);

u4:div\_clk1 port map(clk\_out\_sig,clk\_1\_sig7);

u5:bird\_position port map(reset\_sig,clk\_1\_sig7,btn\_up\_sig,btn\_down\_sig, btn\_left\_sig, btn\_right\_sig,postion\_past\_x\_sig,postion\_now\_x\_sig,position\_past\_sig8, position\_now\_sig10);

u6:seg7\_1 port map(clk\_out\_sig,score\_sig13,score2\_sig13,speed\_mode\_view\_sig,score\_view,memberv\_sig,cat);

u7:mode\_select port map(sw7,sw6,sw1,sw0,speed\_mode\_sig,speed\_mode\_view\_sig,member\_sig,memberv\_sig,more\_sig);

u8:bird\_position2 port map(reset\_sig,clk\_1\_sig7,btn\_up\_sig2,btn\_down\_sig2, position\_past2\_sig8, position\_now2\_sig10);

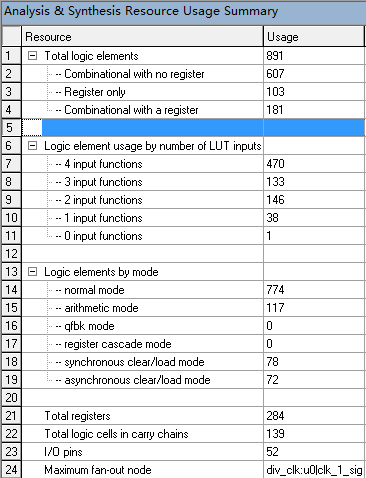
u9:random port map(clk,random\_out\_sig);

u10:cleanbtn port map(clk\_out\_sig,btn\_up,btn\_up\_sig,btn\_down,btn\_down\_sig,btn\_left,btn\_left\_sig,btn\_right,btn\_right\_sig,btn\_up2,btn\_up\_sig2,btn\_down2,btn\_down\_sig2,reset,reset\_sig);

end flappy\_bird\_arc;

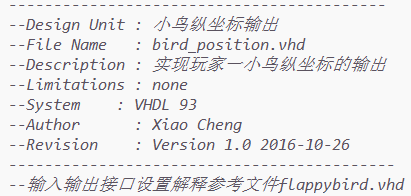
1. **功能说明及资源利用情况**

总体资源利用情况：



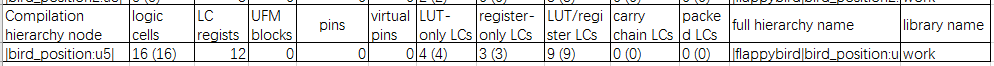


**5.1 bird\_position**

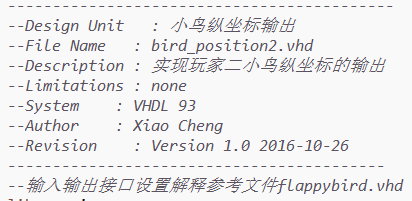


功能说明：实现玩家一小鸟的坐标输出。在时钟上升沿时若检测到按键的信号，对小鸟的横纵坐标进行算术加减。

资源利用情况：



**5.2 bird\_position2**



功能说明：实现玩家二小鸟的坐标输出。在时钟上升沿时若检测到按键的信号，对小鸟的横纵坐标进行算术加减。

资源利用情况：



**5.3 cleanbtn**

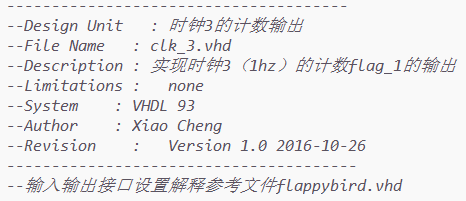


功能说明：实现按键的防抖动功能，将板子上的btn按键引脚与输入相接，通过采样防抖法，每按下一次按键，对应输出高电平持续时间为1/100 s 的单位方波，从而达到不会被多次检测的功能。

资源利用情况：



**5.4 clk\_3**

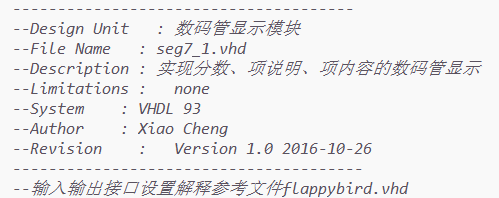


功能说明：实现时钟3的计数输出，例如，当速度选择为一档时，没过1s，实现计数器flag\_1的算术加一，且当检测到复位信号或是游戏结束时将计数器置零，当读取到暂停信号时停止flag\_1的算术加法，最后计数器作为点阵输出模块的输入，以达到管道移动的效果。

资源利用情况：



**5.5 seg7\_1**

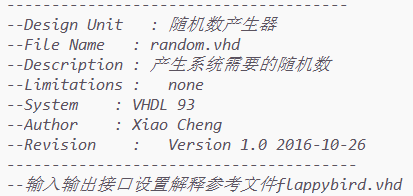


功能说明：实现数码管的显示功能，输入有玩家一、玩家二的分数，速度，玩家人数，输出7段显示以及公共阴极的向量，显示在4到1组数码管组。

资源利用情况：



**5.6 random**

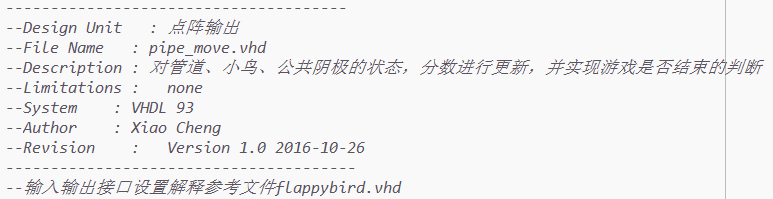


功能说明:接入时钟为50MHZ的时钟，每个上升沿都会被检测并实现一个摸为17的计数器的计数，输出给点阵输出模块。

资源利用情况：



**5.7 pipe\_move**

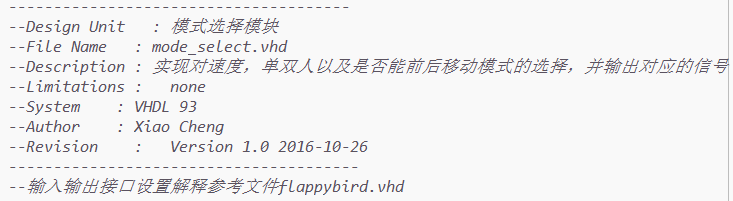


功能说明：实现点阵的输出，每1/1000s输出3组8位的信号，分别为绿、红阳极和公共阴极，其中公共阴极的输出方式为行扫描式的输出，即输出为11111110🡪11111101🡪11111011🡪11110111🡪11101111🡪11011111🡪10111111🡪01111111🡪11111110……其中每次转换的周期为1/1000s。同时，实现分数的更新，判断游戏是否结束，这里采用的方法是如若同时坚测到红色阳极和绿色阳极的输出为‘1’的高电平，则确定游戏结束，如果分数到达12分则判断游戏胜利。

资源利用情况:



**5.8 mode\_select**

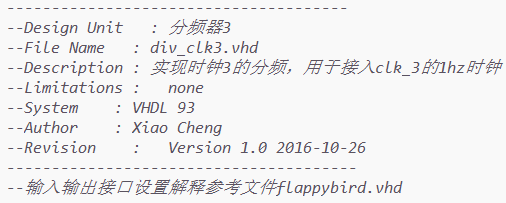


功能说明：实现模式的选择，通过读取拨码开关的信号判断对应的模式，sw7判断人数，sw6判断是否能自由移动，sw1&sw0判断速度。

资源利用情况：

不额外占用逻辑资源，需要使用4个输入管脚。

**5.9 div\_clk3**

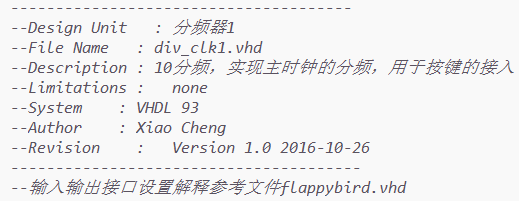


功能说明：实现时钟3的分频，对于3档不同的速度进行不同的分频，分别为1hz，1.67hz和2.5hz。

资源利用情况：



**5.10 div\_clk1**

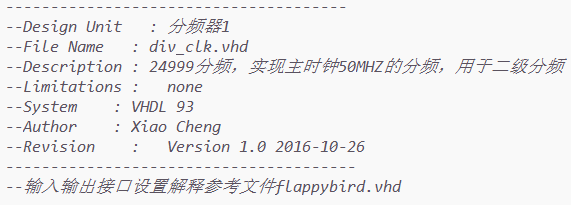


功能说明：实现时钟1的分频，用于检测按键。实现100hz的输出。

资源利用情况：



**5.11 div\_clk**



功能说明：实现1000hz时钟的分频。

资源利用情况：



1. **故障及问题分析**

在本次实验中，从编译到仿真再到下载到板子上做测试，整个过程中我遇到过很多问题，也出现过故障，回头分析故障修改代码，主要的故障和问题如下：

* 首先是编译，刨除分号没加，end if没写等等小错误，其一，vhdl语言不同于其他高级语言，在写for循环时，循环次数必须为常数，而我的代码中需要对循环范围进行判断，通过在循环体内加入if判断条件解决；其二，vhdl语言中没有真正意义的除法，只有整除和取余数，所以不能希望除的结果是小数；其三，对于一个进程，只允许有一个时钟输入，不能检测两个不同时钟的上升沿，甚至不能同时检测一个时钟的上升沿和下降沿，只能通过多进程来描述。
* 接着仿真方面，编译通过后，做仿真时发现对时钟上升沿没有感应，后检查代码，发现敏感量中没有包括时钟。
* 最后是在板子上做测试时出现的情况，这里也包括代码上存在的逻辑错误。首先是正确使用板子的问题，板上点阵出现闪烁，原因是频率未调到50MHZ，频率太慢就会出现闪烁的问题；然后出现数码管的显示问题，靠左侧两个数码管过亮，右边三组数码管则略暗，且左侧两个数码管无法显示对应的数字而其他的数码管只是亮度的问题，原因是拨码开关sw10的第三位和第四位未拨至上边，这会导致这两个数码管的阴极不受控制，且会对其他数码管的亮度造成影响。以上是使用板子的问题，接下来是代码上的逻辑错误。点阵显示管道移动的部分让人煞费苦心，第一次做测试时整列的管道都会亮起，且管道向左移动时会将路径上所有绿色灯管点亮，通过整理代码逻辑，发现在对每一横排管道做更新时，没有预先做清零处理，导致过去的状态会一直延续；然后是判断胜负的问题，之前无法对小鸟在进入管道前死亡的情况做判断，小鸟进入管道会呈现黄色然后才会显示游戏结束，且只进入一个管道后就显示游戏胜利，后引入计数器count\_s1（详见代码），实现每进入一个管道加一分，完美地解决了问题。在做双人模式时，刚开始测试时，出现当两个玩家同时进行游戏时无法显示的问题，后检查代码发现其中一个end if添加的位置出现错误，导致两个玩家共存的程序根本不会执行的情况，修正后解决了问题。

1. **总结和结论**

通过两周的时间，我完成了所有基本功能的展现并实现了很多拓展功能，基本功能包括管道按要求地进行移动，每一组管道的缺口出现需要呈现随机的状态，小鸟要响应按键，且按键都要进行防抖动处理，若小鸟进入了管道，则分数加一，如果小鸟在停留在管道的过程中碰撞到了管子则显示游戏结束，当玩家通过12个管道时显示胜利的‘V’形字样，当玩家失败时显示‘×’字样，玩家可以通过复位按键进行游戏的重置。

关于附加功能，首先，我实现了管道移动速度的可调，设置了3档移速，其中1档为1单位/s，其次，我添加了暂停按钮，玩家可以随时中断游戏的进行，再下来，我设置了可自由移动的模式，即小鸟可以前后移动，自由穿梭，最后，我增加了双人模式，两个玩家可以进行游戏对战，若一方胜利则显示哪方胜利，平局显示‘T’字样。

关于不足，唯一的不足便是还能实现更多的功能，没有加入音效，没有使用LCD进行分数显示，没有设置开机动画，不过总体有很好的可玩性，也有很多的人性化的设计。

感谢这次实验给我带来的体验，虽然过程中修改代码、反复调试非常费时费力，不过当解决了问题之后剩下的全是收获的喜悦，同时，这是真正意义上的工程实践，从头到尾全靠个人的努力设计实现，没有参考其他的类似实验，我对于今后项目实现的流程有了一个更清晰的认识，希望自己再接再厉，精益求精，将“工匠精神”应用到日常的学习和日后的工作中去。