Smasher Video Game Design

CSEE 4840 Spring 2012 Project Design

Prepared for

Prof. Stephen Edwards

Prepared By

Jian Liu, jl3784

Chaoying Kang, ck2566

Tong Zhang, tz2176,

May 10th,2012

Smasher Video Game Design
Final Report

Jian Liu, Chaoying Kang, Tong Zhang

In this project, we designed a video game named “Smasher”. It’s a four board bouncing-ball video game controlled by two rotary switches with some extra effects, such as blinking of the cloud-like bricks, cute voice of the bricks when hit, etc. In the following we will introduce the key characteristics of our design, and discuss how we accomplish them.

i. Video controller

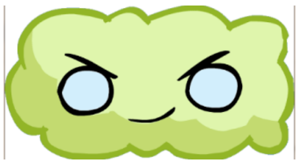
- *Game Graphics:*

We have learned the basic method to generate the VGA output through the fpga from the lab3 three. In our project , we use these method to continue our game graphics.
Our game graphics have some basic elements:

Each element has its own resolution and own colors. At the beginning of our design, we just use 24bit to store the image data of the game. However, it will cost lots of data area as the game graphics elements’ amount grow. So we decide to use the color maps indexed images instead of the 24bit RGB image, which can reduce the storage areas dramatically.

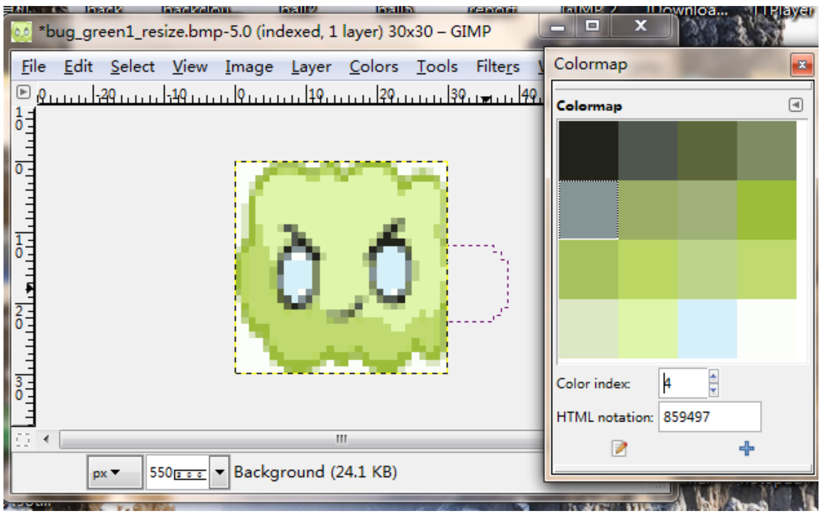
Take the bugs as an example. We use 30*30 16 colors indexed image to store the data.

First, we get the originally data from the internet:



Then we can use the Acdsee to converter the data into the resolution of 30*30.

The next step is to use the GIMP to convert the picture from 24 bits RGB to 16 colors color map indexed image.

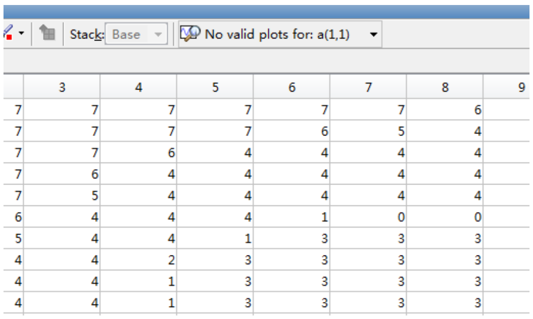


Color index: 4
HTML notation: 859497

Write down the color's RGB value to the VHDL file:

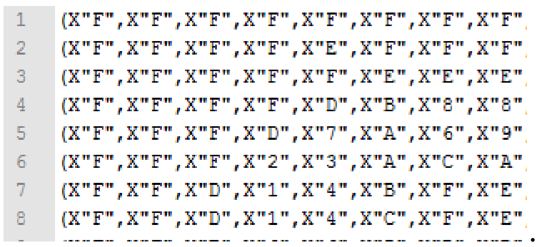
```
constant background: color :=(
X"ed2858",X"b93a96",X"8b5b3a",X"ed4639",
X"e85e5b",X"b3744e",X"eb6586",X"9f8472",
X"f4a79c",X"f0b0bc",X"d5c2c2",X"f5d1d8",
X"ddddb",X"f4e8ae",X"f7fad7",X"f7f8f6"
);
```

The next step is use matlab to read the data's indexed value of every pixel.


Stack: Base
No valid plots for: a(1,1)

	3	4	5	6	7	8	9
7	7	7	7	7	7	6	
7	7	7	7	6	5	4	
7	7	6	4	4	4	4	
7	6	4	4	4	4	4	
7	5	4	4	4	4	4	
6	4	4	4	1	0	0	
5	4	4	1	3	3	3	
4	4	2	3	3	3	3	
4	4	1	3	3	3	3	
4	4	1	3	3	3	3	

And convert the data in to the VHDL file:



```
(X"F",X"F",X"F",X"F",X"F",X"F",X"F",X"F"
(X"F",X"F",X"F",X"F",X"E",X"F",X"F",X"F"
(X"F",X"F",X"F",X"F",X"F",X"E",X"E",X"E"
(X"F",X"F",X"F",X"F",X"D",X"B",X"8",X"8"
(X"F",X"F",X"F",X"D",X"7",X"A",X"6",X"9"
(X"F",X"F",X"F",X"2",X"3",X"A",X"C",X"A"
(X"F",X"F",X"D",X"1",X"4",X"B",X"F",X"E"
(X"F",X"F",X"D",X"1",X"4",X"C",X"F",X"E"
```

We make the bugs for 30*30 pixel character. And they are put in order on the screen. So we decide using tile to realize the bug's display. We can have a map data to decide which tile on the screen should be displayed as a bug, and what color of the bug.For example, X"0000" means this tile should be displayed as an green bug.

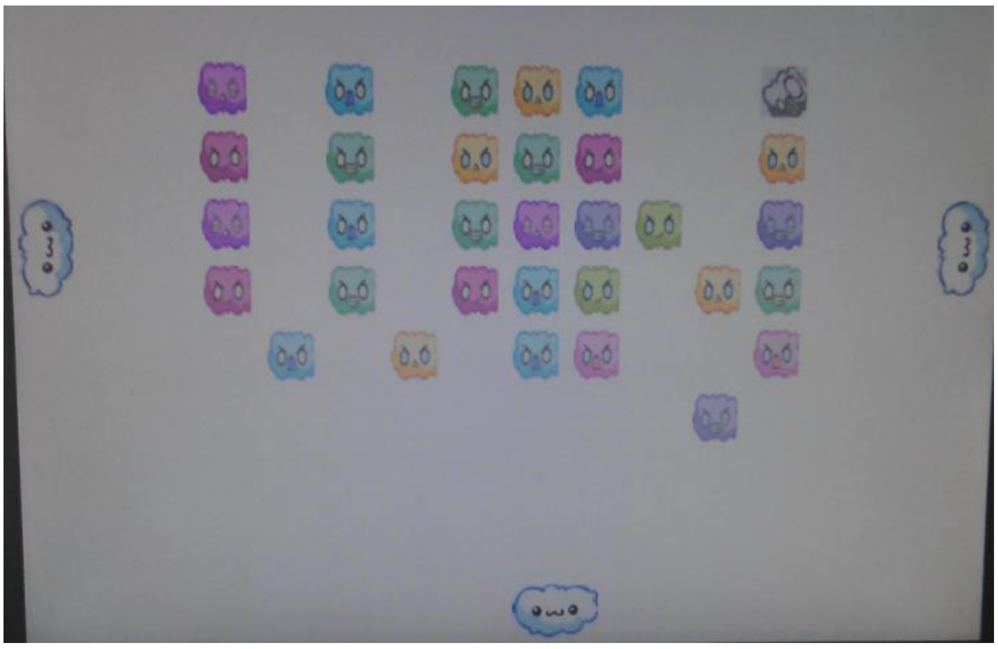
- *Other animation*

In our project,we use some little trick to realize some small animation. Such as the bug's blinking

eye.
We use two pictures

We use a VHDL internal timer to switch the picture, making it like the bug's blinking eye.
Another little animation is when the bug been hit, it will became grey and shacking. This is also realized by switching two pictures using an internal timer.

.



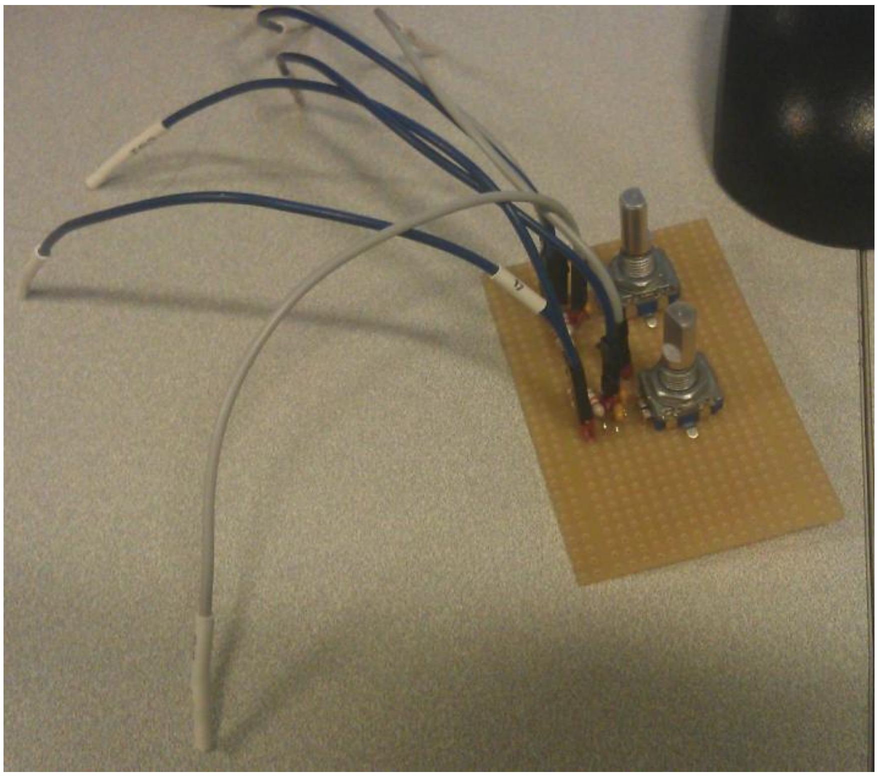
The display of the game when you when the game

Difficulty of the Video part:

Before the end of the project, we found that when we increase the number of sprite, the display will have many delay which will cause the display bugs. We try many ways to reduce the display delay but only receive little effect.

ii. Rotary Switch Design

We use two rotary switches to control the four cloud board (the upper one is bonded with the bottom one and the left one is bonded with the right one). There are three ports on the rotary switch. In our design, the middle one should beconnected to gnd; while the other two is connected to VDD through a resistor. PortA and B will give a group of output when we rotate the switch. The signal it gives through the oscilloscope is 11 01 00 10(clockwise) or 11 10 00 01(counterclockwise). Based on these output, we can design a Finite state machine which will take these outputs as inputs. We used the port 0,1,2,3 on the GPIO_1 as the inputs of two finite state machines. We have a component called rotary switch which will generate an interrupt to the main program every time we rotate the switch. During the interrupt, there is a signal transferred to the main program which indicates the rotating direction so that the main program can control the moving of the cloud.



iii. Audio Design

Audio design is based on the de2_wm8731_audio file given in the lab3. We first resampled the music to lower the data sampling rate so that we can have less communication between the cpu and the audio module. To obtain a better audio performance, we first want to write the data in the ROM in the VHDL, However, when we have about 8000 bytes of data, we find it takes a very long time to compile, it is not reasonable to store the huge amount of data in the ROM in VHDL.

We then tried to use the SDRAM to store the main program and SRAM to store the music. However, we found lots of bugs arises in the VGA display. We think this is mainly caused by the relatively low transfer speed of SDRAM and the high speed requirement of VGA display, interrupt detection and other control signal. Taking the advice from the TA, we stored the music in the SRAM as an alt_u16 array, which would only consume about 30K Bytes out of 512KB of the total memory. We build an audio module in the nios system, which would simultaneously generate a interrupt in a frequency of 1.34K Hz. Every time the main program detect the interrupt, 64 bytes of music data will be transferred from the SRAM, the audio module will store these data in a 64 bytes buffer and the data in this buffer will be read in a frequency of 43 K Hz. A new interrupt will be generated by the audio module when the data in the buffer reach its ends during the reading, a new bunch of 64 bytes data will be transferred to refresh the buffer.

For the sound we selected, we tried to increase the frequencies as much as possible to get a better sound performance. The final sampling frequency is 11K Hz. There are four kinds of sounds in total, the background music,the sound for hit of the brick, the hit of the vertical board and the sound for hit of the horizontal board.

iv. C Programming

The C programming part is used to control all the video patterns and audio signals and make them combine properly.

In the programming process, several key problems are handled respectively:

- Bouncing direction and speed after hitting the brick

To make the game more complex and fun, the bouncing directions and speed are treated differently according to different incident angles. The structure of a cloud-like brick is shown in the following figure. Basically, we separate every brick into eight parts: four edges and four corners. When the ball moves along facing directly to any of the corners, it will reflect straight with the same horizontal and vertical mobility vector magnitude. Otherwise if the ball hits any of the edges or hits the corner in a un-direct way, it will retain its parallel mobility vector sign, and reverse the other one. Besides, each time when a brick is hit, it got erased and the mobility magnitude for both horizontalvector and vertical vector will be reset to one. This is designed to cooperate with the reflection rule of the board, which will be discussed later.

Pattern of the cloud-like brick

- Bouncing direction and speed from the board

When bouncing from the board, speed and reflection angle will also vary. Basically, if the ball hits the first half of the board along its moving direction, both of the mobility vector sign will be reversed, while if the other part is hit, only the sign of vector paralleled with the board is reversed. Moreover, the speed of the reflected ball is based on the distance between the position it hits the board and the middle of the board. Therefore, the more it close to the edge, the more it faster the ball will be, and harder the game will be. The only way to slow down the ball is to hit bricks. Thus, if the player just bounces the ball between boards without hitting any bricks, the speed will get faster and faster. Pattern of the board is shown in following figure.

Pattern of the board

- End Frame design

If all the lives are cost, then the game end and goes into lose frame; if all the bricks are hit, the game also ends and goes into win frame. In either case, a pattern consists of bricks flashes. “WIN” for the win frame and “LOSE” for the lose frame. In each pattern, the bricks will flash one by one until the entire word is print. Then the word will keep valid for a moment then be erased off. This is accomplished by using a number of cycles for each of the bricks.

- Audio control

Each time the ball hits, either a brick or a board, a corresponding sound will be sent. In this case, there may a collision when multi bricks and boards are hit within close time and the sound could not be generated in time. To solve this problem, if any new audio elements need to be generated before the current one, the current one will stop immediately and the second one will begin to work.

v. Conclusion

The division of the duty is important. Code should be well commented if the meaning is not clear. Because we have thousands of lines, it helps other people to understand and can also help the one who wrote the code to pick up the idea after a long time. Finish the milestone on time so that you can have an easy life at the end. It was great experience working as a team solving all the problems. We did learn a lot from the project, and we all had a great time.

Thanks for Prof. Edwards and our TA Shangru Li for all the help and suggestion

VHDL Codes:

de2_vga_raster.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity de2_vga_raster is

   port (

      clk_50 : in std_logic;                         -- Should be 25.125 MHz
     writedata : in unsigned (15 downto 0);
     readdata : out unsigned(15 downto 0);
      address : in std_logic_vector(17 downto 0);
     irq : out std_logic;
      chipselect,
     reset : in std_logic;
      read,
      write : in std_logic;
      VGA_CLK,                                      -- Clock
      VGA_HS,                                      -- H_SYNC
      VGA_VS,                                      -- V_SYNC
      VGA_BLANK,                                     -- BLANK
      VGA_SYNC : out std_logic;               -- SYNC
      VGA_R,                                       -- Red[9:0]
      VGA_G,                                       -- Green[9:0]
      VGA_B : out std_logic_vector(9 downto 0) -- Blue[9:0]
      );

end de2_vga_raster;

architecture rtl of de2_vga_raster is

   -- Video parameters

   constant HTOTAL           : integer := 800;
   constant HSYNC            : integer := 96;
   constant HBACK_PORCH    : integer := 48;
   constant HACTIVE          : integer := 640;
   constant HFRONT_PORCH : integer := 16;

   constant VTOTAL           : integer := 525;
constant VSYNC              : integer := 2;
constant VBACK_PORCH    : integer := 33;
constant VACTIVE          : integer := 480;
constant VFRONT_PORCH : integer := 10;
signal bugdie_sta : unsigned(1 downto 0);
signal HPOS : integer ;
signal VPOS : integer ;
signal irq1 : std_logic;
signal irq2 : std_logic;
signal RECTANGLE_HSTART : integer := 100;
signal RECTANGLE_VSTART : integer := 100;
signal BALL_POS_H : integer := 10;
signal BALL_POS_V : integer := 10;
signal BUG_H : integer := 0;
signal BUG_V : integer := 0;
signal BUG_STA : integer := 0;
signal YUN_POS : integer := 10;
signal YUN_POS1 : integer := 10;

signal SPEEDUP_POS_H : integer ;
signal SPEEDUP_POS_V : integer ;
signal LOSELIFE_POS_H : integer ;
signal LOSELIFE_POS_V : integer ;
signal blackhole_POS_H : integer ;
signal blackhole_POS_V : integer ;
signal heart_POS_H : integer := 0;
signal heart_POS_V : integer := 0;

signal RECTANGLE_HSTART_yun : integer := 100;
signal RECTANGLE_VSTART_yun : integer := 100;
signal YUN_STA : integer := 0;
signal YUN_STA11 : integer := 0;
signal YUN_STA21 : integer := 0;
signal YUN_STA31 : integer := 0;
signal YUN_STA41 : integer := 0;
signal BUG_STA1 : integer := 0;
signal BUG_STA2 : integer := 0;
signal BUG_STA3 : integer := 0;
signal BUG_STA4 : integer := 0;
signal YUN_FLAG : integer := 30000000;
signal BUG_FLAG1 : integer := 50000000;
signal BUG_FLAG2 : integer := 50000000;
signal BUG_FLAG3 : integer := 50000000;
signal BUG_FLAG4 : integer := 50000000;
signal bugdie2_RGB : unsigned(1 downto 0);
signal bugdie3_RGB : unsigned(1 downto 0);

-- Signals for the video controller
signal Hcount : unsigned(9 downto 0);     -- Horizontal position (0-800)
signal Vcount : unsigned(9 downto 0);     -- Vertical position (0-524)
signal EndOfLine, EndOfField : std_logic;
signal numv_bug : integer := 0;
signal numh_bug : integer := 0;
signal numv_bug_flag : integer := 0;
signal numh_bug_flag : integer := 0;
signal numh_back : integer := 0;
signal numv_back : integer := 0;
signal numh_back_1 : integer := 0;
signal numv_back_1 : integer := 0;
signal numh_back_2 : integer := 0;
signal numv_back_2 : integer := 0;
signal numh_back_3 : integer := 0;
signal numv_back_3 : integer := 0;
signal numv_yun : integer := 0;
signal numh_yun : integer := 0;
signal numv_yun1 : integer := 0;
signal numh_yun1 : integer := 0;
signal numv_yun2 : integer := 0;
signal numh_yun2 : integer := 0;
signal numv_yun3 : integer := 0;
signal numh_yun3 : integer := 0;
signal numv_ball : integer := 0;
signal numh_ball : integer := 0;


signal numv_numb : integer := 0;
signal numh_numb : integer := 0;
signal numv_numb1 : integer := 0;
signal numh_numb1 : integer := 0;
signal numv_numb2 : integer := 0;
signal numh_numb2 : integer := 0;
signal numv_heart : integer := 0;
signal numh_heart : integer := 0;
signal numv_heart1 : integer := 0;
signal numh_heart1 : integer := 0;
signal numv_heart2 : integer := 0;
signal numh_heart2 : integer := 0;
signal numv_blackhole1 : integer := 0;
signal numh_blackhole1 : integer := 0;
signal numv_blackhole2 : integer := 0;
signal numh_blackhole2 : integer := 0;
signal numv_blackhole3 : integer := 0;
signal numh_blackhole3 : integer := 0;
signal numh_3 : integer := 0;
signal numv_loselife : integer := 0;
signal numh_loselife : integer := 0;
signal numv_speedup : integer := 0;
signal numh_speedup : integer := 0;

signal OUT_R : std_logic_vector(7 downto 0);
signal OUT_G : std_logic_vector(7 downto 0);
signal OUT_B : std_logic_vector(7 downto 0);

signal bug_pink1_RGB : unsigned(3 downto 0);
signal bug_pink2_RGB : unsigned(3 downto 0);
signal bug_purple1_RGB : unsigned(3 downto 0);
signal bug_purple2_RGB : unsigned(3 downto 0);
signal bug_green1_RGB : unsigned(3 downto 0);
signal bug_green2_RGB : unsigned(3 downto 0);
signal bug_yellow1_RGB : unsigned(3 downto 0);
signal bug_yellow2_RGB : unsigned(3 downto 0);
signal yun1_RGB : unsigned(3 downto 0);
signal yun2_RGB : unsigned(3 downto 0);
signal yun3_RGB : unsigned(3 downto 0);
signal yun11_RGB : unsigned(3 downto 0);
signal yun12_RGB : unsigned(3 downto 0);
signal yun13_RGB : unsigned(3 downto 0);
signal yun21_RGB : unsigned(3 downto 0);
signal yun22_RGB : unsigned(3 downto 0);
signal yun23_RGB : unsigned(3 downto 0);
signal yun31_RGB : unsigned(3 downto 0);
signal yun32_RGB : unsigned(3 downto 0);
signal yun33_RGB : unsigned(3 downto 0);
signal blackhole1_RGB : unsigned(3 downto 0);
signal blackhole2_RGB : unsigned(3 downto 0);
signal blackhole3_RGB : unsigned(3 downto 0);
signal loselife_RGB : unsigned(3 downto 0);
signal speedup_RGB : unsigned(1 downto 0);
signal heart_RGB : unsigned(2 downto 0);
signal numberofheart : integer;
signal ball_RGB : unsigned(3 downto 0);
signal back_RGB : std_logic;
signal ball_sta : unsigned(2 downto 0);
signal one_RGB : unsigned(3 downto 0);
signal two_RGB : unsigned(3 downto 0);
signal three_RGB : unsigned(3 downto 0);
signal four_RGB : unsigned(3 downto 0);
signal five_RGB : unsigned(3 downto 0);
signal six_RGB : unsigned(3 downto 0);
signal seven_RGB : unsigned(3 downto 0);
signal eight_RGB : unsigned(3 downto 0);
signal nine_RGB : unsigned(3 downto 0);
signal zero_RGB : unsigned(3 downto 0);
signal numbsco: unsigned(15 downto 0);




    signal VGA_R_BUFFER : std_logic_vector(9 downto 0);
    signal VGA_G_BUFFER : std_logic_vector(9 downto 0);
    signal VGA_B_BUFFER : std_logic_vector(9 downto 0);
    signal VGA_R_yun1 : std_logic_vector(9 downto 0);
    signal VGA_G_yun1 : std_logic_vector(9 downto 0);
    signal VGA_B_yun1 : std_logic_vector(9 downto 0);
    signal VGA_R_yun2 : std_logic_vector(9 downto 0);
    signal VGA_G_yun2 : std_logic_vector(9 downto 0);
    signal VGA_B_yun2 : std_logic_vector(9 downto 0);
    signal VGA_R_yun3 : std_logic_vector(9 downto 0);
    signal VGA_G_yun3 : std_logic_vector(9 downto 0);
    signal VGA_B_yun3 : std_logic_vector(9 downto 0);
    signal VGA_R_yun4 : std_logic_vector(9 downto 0);
    signal VGA_G_yun4 : std_logic_vector(9 downto 0);
    signal VGA_B_yun4 : std_logic_vector(9 downto 0);
    signal VGA_R_ball : std_logic_vector(9 downto 0);
    signal VGA_G_ball : std_logic_vector(9 downto 0);
    signal VGA_B_ball : std_logic_vector(9 downto 0);
    signal VGA_R_bug : std_logic_vector(7 downto 0);
    signal VGA_G_bug : std_logic_vector(7 downto 0);
    signal VGA_B_bug : std_logic_vector(7 downto 0);
    signal VGA_R_num : std_logic_vector(7 downto 0);
    signal VGA_G_num : std_logic_vector(7 downto 0);
    signal VGA_B_num : std_logic_vector(7 downto 0);
```

```
        signal yun_vga1,yun_vga2,yun_vga3,yun_vga4,bug_vga,ball_vga : std_logic;
        signal   reset1,reset2,reset3,reset4,reset11,reset22,reset33,reset44,virb_sta2,virb_sta3,virb_sta4   :
std_logic;
        signal h_virb1,v_virb1,h_virb2,v_virb2,h_virb3,v_virb3,h_virb4,v_virb4 : integer;
        signal virb_sta1 : unsigned(0 downto 0)    :="1";

        signal vga_hblank, vga_hsync, vga_vblank, vga_vsync : std_logic;     -- Sync. signals
         signal rectangle_h, rectangle_v, rectangle : std_logic;     -- rectangle area
        signal rectangle_h_yun, rectangle_v_yun, rectangle_yun : std_logic;
        signal rectangle_h_yun1, rectangle_v_yun1, rectangle_yun1 : std_logic;
        signal rectangle_h_yun2, rectangle_v_yun2, rectangle_yun2 : std_logic;
        signal rectangle_h_yun3, rectangle_v_yun3, rectangle_yun3 : std_logic;
        signal rectangle_h_ball, rectangle_v_ball, rectangle_ball : std_logic;
        signal rectangle_h_heart, rectangle_v_heart, rectangle_heart : std_logic;

        signal rectangle_h_loselife, rectangle_v_loselife, rectangle_loselife : std_logic;
        signal rectangle_h_blackhole1, rectangle_v_blackhole1, rectangle_blackhole1 : std_logic;
        signal rectangle_h_blackhole2, rectangle_v_blackhole2, rectangle_blackhole2 : std_logic;
        signal rectangle_h_blackhole3, rectangle_v_blackhole3, rectangle_blackhole3 : std_logic;
        signal rectangle_h_speedup, rectangle_v_speedup, rectangle_speedup : std_logic;   signal
rectangle_h_bug_purple, rectangle_v_bug_purple, rectangle_bug_purple : std_logic;
        signal rectangle_h_bug_green, rectangle_v_bug_green, rectangle_bug_green : std_logic;
        signal rectangle_h_back, rectangle_v_back, rectangle_back : std_logic;
        signal rectangle_h_numb, rectangle_v_numb, rectangle_numb : std_logic;
        signal number : unsigned(15 downto 0);
        signal clk : std_logic;
        type bug_row is array(0 to 29) of std_logic_vector(7 downto 0);
        type bug_column is array(0 to 29) of bug_row;
        type yun_row is array(0 to 33) of std_logic_vector(7 downto 0);
        type yun_column is array(0 to 57) of yun_row;
        type ball_row is array(0 to 14) of std_logic_vector(7 downto 0);
        type ball_column is array(0 to 14) of ball_row;
        type    background_r is array (0 to 11) of std_logic_vector(3 downto 0);
        type    background_c is array (0 to 15) of background_r;
        type    heartnumb is array (0 to 3) of std_logic;
        type    numin is array (0 to 3) of unsigned(3 downto 0);
        type    color is array (0 to 15) of std_logic_vector(23 downto 0);
        type    color1 is array (0 to 3) of std_logic_vector(7 downto 0);
        type    color2 is array (0 to 3) of std_logic_vector(23 downto 0);
        type    color3 is array (0 to 7) of std_logic_vector(23 downto 0);
        signal numinput : numin := ("0000","0000","0000","0000");
signal background : background_c := (
("0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000"),
("0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000"),
("0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000"),
("0000","0000","0000","1110","1111","1110","1111","1110","1001","0000","0000","0000"),
("0000","0000","0000","1000","1001","1010","1011","1100","1101","0000","0000","0000"),
("0000","0000","0000","1100","1101","1100","1101","1110","1011","0000","0000","0000"),
("0000","0000","0000","1011","1100","1011","1011","1001","1101","0000","0000","0000"),
("0000","0000","0000","1101","1001","1101","1111","1110","1001","0000","0000","0000"),
("0000","0000","0000","1001","1101","1110","1100","1100","1101","0000","0000","0000"),
("0000","0000","0000","1100","1111","1011","1010","1000","1010","0000","0000","0000"),
("0000","0000","0000","1011","1001","1010","1001","1101","1101","0000","0000","0000"),
("0000","0000","0000","1010","1101","1110","1001","1001","1011","0000","0000","0000"),
("0000","0000","0000","1000","1001","1011","1101","1000","1100","0000","0000","0000"),
("0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000"),
("0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000"),
("0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000")
);
signal heartnum : heartnumb := ('1','1','1','1');
constant zero : color :=(
X"b11d6d",X"da4850",X"9c6e73",X"897e8d",
X"d86793",X"cf8449",X"9a967e",X"dd9ab6",
X"faa459",X"f2b02c",X"c0bca7",X"d8bc6f",
X"c5c2c9",X"dedadd",X"f6e1bc",X"fcfdfb"
);

constant one : color :=(
X"77358f",X"9164a7",X"c05a8a",X"e65943",
X"ea843a",X"969a8a",X"abaab4",X"daa755",
X"b1b093",X"b7b1d9",X"c5b9d7",X"ddd6e4",
X"efded7",X"faf2e8",X"fafbff",X"fdfefb"
);

constant two : color :=(
X"a91436",X"b33357",X"b83b35",X"d26828",
X"d26788",X"cc7c96",X"f094ae",X"f5b322",
X"fcab7e",X"f2b0c4",X"f5be62",X"e5c0cc",
X"f3dca4",X"f7dce5",X"faf1d4",X"fdfdfb"
);

constant three : color :=(
X"db3168",X"d4437a",X"d46e94",X"e786a6",
X"d58cb0",X"e094a5",X"afaca7",X"c7a3c1",
X"f1a0bc",X"ebb4c9",X"cfc5dd",X"ecced8",
X"f8e0ea",X"faf0f6",X"fffbfd",X"fcfefc"
);
constant four : color :=(
X"6c3189",X"db2050",X"dc2c63",X"9f63a1",
X"e75c41",X"e05c89",X"e390b2",X"e4a63d",
X"bba7ce",X"dfba8e",X"e3b9cd",X"f1c3d5",
X"e8d9e6",X"fde9a7",X"faeff6",X"fdfefc"
);
constant five : color :=(
X"dc0c55",X"d92262",X"e44746",X"db457a",
X"de618e",X"e78244",X"db7da1",X"d1a2bf",
X"eeb3c7",X"f6c03e",X"e3cddb",X"f2dde6",
X"f3e8ef",X"f8efd4",X"fefdf4",X"fcfdfc"
);
constant six : color :=(
X"c8254f",X"83449f",X"c65a79",X"c1648e",
X"b98099",X"a189c0",X"e47f67",X"ae9576",
X"e295ae",X"e6a85c",X"ccb1ca",X"bec4db",
X"fbd875",X"d8dde9",X"fbfcfb",X"fffdee"
);
constant seven : color :=(
X"c54b56",X"8a55a2",X"cb4e7b",X"b46d5f",
X"b87391",X"da7431",X"e498b9",X"baae8f",
X"f9a482",X"ecb12d",X"eab262",X"c2b5cb",
X"eecbab",X"e3d8e5",X"f8f0ce",X"fdfdfb"
);
constant eight : color :=(
X"d80649",X"d82160",X"d63d78",X"c75d92",
X"c57eaf",X"df8cab",X"c196bd",X"b7a9cf",
X"c6afd4",X"e2adc0",X"bbb7d9",X"cdc8e2",
X"d5cfe1",X"e8e3ec",X"f8f9fd",X"fdfefb"
);
constant nine : color :=(
X"db0247",X"d84078",X"d15c86",X"986dab",
X"c77789",X"e07f67",X"e88527",X"b29ac5",
X"f9b719",X"e7b587",X"f2c760",X"e8c3d2",
X"f3d994",X"f4dee7",X"f4e8c6",X"fcfefc"
);
constant ball : color :=(
X"62a6f1",X"7ab3f2",X"97bff3",X"fec11b",
X"fec645",X"fbc56a",X"b4d0f5",X"f9ce92",
X"fedc1d",X"f4d6b7",X"d6dff6",X"ffe666",
X"f4e4da",X"fef086",X"f5f3fb",X"fcfbf9"
);

constant bug_green1 : color :=(
X"21231b",X"4e564d",X"5d663a",X"7e8b63",
X"859497",X"9bad65",X"9fb078",X"9cbd3a",
X"a7c25c",X"bcd764",X"bed38a",X"c0da71",
X"dce7c3",X"dff5aa",X"d6f0f9",X"fcfefa"
);
constant bug_green2 : color :=(
X"5e6746",X"889664",X"8cb01b",X"9bbc37",
X"abc94b",x"b1cb5d",x"b6c688",x"b7cc70",
x"c0da6e",x"c6df7a",x"d0e39e",x"d7e2b1",
x"d4ec97",x"ececcc",x"e0f5ab",x"fcfefa"
);

constant bug_pink1 :color := (
x"2b262a",x"675a65",x"885a6d",x"df61ba",
x"88909a",x"ab899f",x"c58f87",x"e384c9",
x"f4a1e4",x"fbb7ab",x"e9b9df",x"bed8e1",
x"f9d2f4",x"d3f0fa",x"faeaf5",x"fdfefd"
);

constant bug_pink2 :color := (
x"513b41",x"896d82",x"997070",x"dc56b2",
x"e36ec1",x"c79088",x"e387c9",x"c3a5c0",
x"f69ee4",x"f2a3e3",x"e2b6d8",x"fbb5aa",
x"f3b6e5",x"f9d1f3",x"fadaf3",x"fdfcfc"
);

constant bug_purple1 :color := (
x"131315",x"35343d",x"4b4a56",x"7f5cca",
x"6c6b7d",x"8e70d1",x"937fb9",x"848895",
x"9e85d7",x"b09be0",x"a7aabd",x"d1c4ea",
x"b9d2db",x"dfd3f3",x"d8edf8",x"fbfefd"
);

constant bug_purple2 :color := (
x"5f5578",x"61626d",x"815ecc",x"9072d2",
x"8e7db3",x"838593",x"a48bd9",x"af9be1",
x"c3b2e7",x"aac0c7",x"c7bcda",x"ccbfea",
x"dcd0f1",x"e5dbf8",x"d7e2f2",x"fafdfd"
);

constant bug_yellow1 :color := (
x"20211c",x"564e34",x"4e5454",x"818577",
x"9a8640",x"9b9671",x"8699a4",x"eb8315",
x"eec441",x"d4cd9d",x"f9d96f",x"f5df8c",
x"d2edf6",x"faf5bd",x"fbf6ca",x"fdfffa"
```

```
);

constant bug_yellow2 :color := (
x"6a6957",x"9e9975",x"9b9a96",x"e8b315",
x"c4bc8c",x"dcc066",x"eec23b",x"f9d86b",
x"f8dd7d",x"ebe2af",x"fae89d",x"f7ecca",
x"fcefad",x"fbf5b8",x"fbf7c0",x"fcfefa"
);

constant xiaoyun1 :color := (
x"000200",x"272927",x"4d4f4c",x"4e84ea",
x"6796ec",x"969896",x"7ca4ee",x"96b6f0",
x"acc4cf",x"c0c1be",x"b3ccf5",x"cee8fb",
x"d3eef9",x"f5f4fd",x"f4fcff",x"fdfffc"
);

constant xiaoyun2 :color := (
x"0a0d09",x"232321",x"2f302e",x"51524f",
x"4d84e9",x"848786",x"6696ec",x"7ca5ef",
x"a5a9a8",x"94b7f0",x"c6c7c4",x"b3cdf5",
x"cde8fb",x"d5eef9",x"f6fcff",x"fdfffc"
);

constant xiaoyun3 :color := (
x"151412",x"645859",x"4e85ea",x"919290",
x"6b99ed",x"cb8b9b",x"8cb1f1",x"fab6c7",
x"c9cbc8",x"bad2f7",x"cadce8",x"cee7fb",
x"f0e0ea",x"d5eef9",x"f6fcff",x"fdfffb"
);
constant bugdie1 : color1 :=(
X"2a",X"72",X"a3",X"e5"
);
constant bugdie2 : color1 :=(
X"1f",X"6d",X"a0",X"ef"
);
constant bugdie3 : color1 :=(
X"2e",X"75",X"a5",X"f0"
);
constant speedup: color2 :=(
X"7ab3f2",X"9ab9db",X"bfd9f5",X"f4f9fa"
);
constant blackhole: color :=(
X"000200",X"0c101a",X"151b2a",X"212638",
X"2c3246",X"384158",X"434e68",X"4f5e7c",
X"5d6b8c",X"6e7c9e",X"747f96",X"8d868b",
X"8491af",X"9fa8be",X"c6b39c",X"edddb9"
);
constant loselife: color :=(
X"504932",X"867557",X"87794f",X"ae8075",
X"b2a170",X"f3a3ab",X"cec07f",X"fed42b",
X"ffda49",X"fddf61",X"fce171",X"fee787",
X"fded9f",X"faedb3",X"fff4d4",X"fffefb"
);
constant heart: color3 :=(
X"c36969",X"b17a88",X"a088a6",X"f77a5c",
X"60abf6",X"8ebbee",X"d8eafa",X"fdfffc"
);
constant backgroundco: color :=(
X"cee6ea",X"ffffff",X"d5e5ec",X"ffffff",
X"e8e9cb",X"eaecc4",X"ebedde",X"e5f3ed",
X"ecf1f4",X"e6f5f5",X"ffffff",X"ffffff",
X"ffffff",X"ffffff",X"ffffff",X"ffffff"
);
begin
   -- Horizontal and vertical counters
process (clk_50)
begin
       if rising_edge(clk_50) then
              clk<= not clk;
       end if;
end process;

   HCounter : process (clk)
   begin
       if rising_edge(clk) then
           if reset = '1' then
               Hcount <= (others => '0');
           elsif EndOfLine = '1' then
               Hcount <= (others => '0');
           else
               Hcount <= Hcount + 1;
           end if;
       end if;
   end process HCounter;

   EndOfLine <= '1' when Hcount = HTOTAL - 1 else '0';

   VCounter: process (clk)
```

```
begin
   if rising_edge(clk) then
      if reset = '1' then
         Vcount <= (others => '0');
        irq <='0';
      elsif EndOfLine = '1' then
         if EndOfField = '1' then
        irq <='1';
            Vcount <= (others => '0');
         else
            Vcount <= Vcount + 1;
        irq <='0';
         end if;
      end if;
   end if;
end process VCounter;

EndOfField <= '1' when Vcount = VTOTAL - 1 else '0';

-- State machines to generate HSYNC, VSYNC, HBLANK, and VBLANK

HSyncGen : process (clk)
begin
   if rising_edge(clk) then
      if reset = '1' or EndOfLine = '1' then
         vga_hsync <= '1';
      elsif Hcount = HSYNC - 1 then
         vga_hsync <= '0';
      end if;
   end if;
end process HSyncGen;

HBlankGen : process (clk)
begin
   if rising_edge(clk) then
      if reset = '1' then
         vga_hblank <= '1';
      elsif Hcount = HSYNC + HBACK_PORCH then
         vga_hblank <= '0';
      elsif Hcount = HSYNC + HBACK_PORCH + HACTIVE then
         vga_hblank <= '1';
      end if;
   end if;
end process HBlankGen;
```

```
    VSyncGen : process (clk)
    begin
        if rising_edge(clk) then
            if reset = '1' then
                vga_vsync <= '1';
            elsif EndOfLine ='1' then
                if EndOfField = '1' then
                    vga_vsync <= '1';
                elsif Vcount = VSYNC - 1 then
                    vga_vsync <= '0';
                end if;
            end if;
        end if;
    end process VSyncGen;

    VBlankGen : process (clk)
    begin
        if rising_edge(clk) then
            if reset = '1' then
                vga_vblank <= '1';
            elsif EndOfLine = '1' then
                if Vcount = VSYNC + VBACK_PORCH - 1 then
                    vga_vblank <= '0';
                elsif Vcount = VSYNC + VBACK_PORCH + VACTIVE - 1 then
                    vga_vblank <= '1';
                end if;
            end if;
        end if;
    end process VBlankGen;

YUNSTA: process (clk)
begin
if rising_edge(clk) then

                if YUN_FLAG <= 70000000 then
                YUN_STA <= 0;
                YUN_FLAG <= YUN_FLAG+1;
                elsif YUN_FLAG <= 73000000 then
                YUN_STA <= 1;
                YUN_FLAG <= YUN_FLAG+1;
                else YUN_FLAG <= 0;
end if;
end if;
end process;

BUG_PINKSTA1: process (clk)
begin
if rising_edge(clk) then

		if BUG_FLAG1 <= 50000000 then
		BUG_STA1 <= 0;
		BUG_FLAG1 <= BUG_FLAG1+1;
		elsif BUG_FLAG1 <= 53000000 then
		BUG_STA1 <= 1;
		BUG_FLAG1 <= BUG_FLAG1+1;
		else BUG_FLAG1 <= 0;
end if;
end if;
end process;



BUG_PINKSTA2: process (clk)
begin
if rising_edge(clk) then

		if BUG_FLAG2 <= 100000000 then
		    BUG_STA2 <= 0;
		    BUG_FLAG2 <= BUG_FLAG2+1;
		elsif BUG_FLAG2 <= 103000000 then
		    BUG_STA2 <= 1;
		    BUG_FLAG2 <= BUG_FLAG2+1;
		else BUG_FLAG2 <= 0;
end if;
end if;
end process;



BUG_PINKSTA3: process (clk)
begin
if rising_edge(clk) then

		if BUG_FLAG3 <= 130000000 then
		    BUG_STA3 <= 0;
		    BUG_FLAG3 <= BUG_FLAG3+1;
		elsif BUG_FLAG3 <= 133000000 then
```

```
                    BUG_STA3 <= 1;
                    BUG_FLAG3 <= BUG_FLAG3+1;
               else BUG_FLAG3 <= 0;
end if;
end if;
end process;

BUG_PINKSTA4: process (clk)
begin
if rising_edge(clk) then

               if BUG_FLAG4 <= 80000000 then
                    BUG_STA4 <= 0;
                    BUG_FLAG4 <= BUG_FLAG4+1;
               elsif BUG_FLAG4 <= 83000000 then
                    BUG_STA4 <= 1;
                    BUG_FLAG4 <= BUG_FLAG4+1;
               else BUG_FLAG4 <= 0;
end if;
end if;
end process;
YUNSTA11: process (clk_50)
begin
       if rising_edge(clk_50) then
              if chipselect = '1' then
                     if address(7 downto 0)="001100" then
                                   YUN_STA11 <= to_integer(writedata);
                     end if;
              end if;
       end if;
end process;


YUNSTA12: process (clk_50)
begin
       if rising_edge(clk_50) then
              if chipselect = '1' then
                     if address(7 downto 0)="001101" then
                                   YUN_STA21 <= to_integer(writedata);
                     end if;
              end if;
       end if;
end process;
```

```
YUNSTA13: process (clk_50)
begin
        if rising_edge(clk_50) then
                if chipselect = '1' then
                        if address(7 downto 0)="001110" then
                                        YUN_STA31 <= to_integer(writedata);
                        end if;
                end if;
        end if;
end process;

YUNSTA14: process (clk_50)
begin
        if rising_edge(clk_50) then
                if chipselect = '1' then
                        if address(7 downto 0)="001111" then
                                        YUN_STA41 <= to_integer(writedata);
                        end if;
                end if;
        end if;
end process;

BUGSTA : process (clk_50)
begin
        if rising_edge(clk_50) then
                if chipselect = '1' then
                        if address(7 downto 0)="00001000" then
                                        h_virb1 <= to_integer(writedata(7 downto 4));
                                        v_virb1 <= to_integer (writedata(3 downto 0));
                                        background(h_virb1)(v_virb1)(3) <= std_logic(writedata(8));

                        end if;
                        end if;
                end if;
end process;

RES1 : process (clk)
                begin
                        if rising_edge(clk) then
                                reset11<= reset1;
                        end if;
                end process;

RES2 : process (clk)
```

```
                begin
                        if rising_edge(clk) then
                                reset22<=reset2;
                        end if;
                end process;

RES3 : process (clk)
                begin
                        if rising_edge(clk) then
                                reset33<=reset3;
                        end if;
                end process;

RES4 : process (clk)
                begin
                        if rising_edge(clk) then
                                reset44<=reset4;
                        end if;
                end process;

numinput1: process (clk_50)
begin
if rising_edge(clk_50) then
        if reset='1' then
        numinput(0) <= "0000";
        else
                if chipselect = '1' then
                        if address(7 downto 0)="00010001" then
                                numinput(0) <= (writedata(3 downto 0));
                        end if;
                end if;
        end if;
end if;
end process;

numinput2: process (clk_50)
begin
if rising_edge(clk_50) then
        if reset='1' then
        numinput(1) <= "0000";
        else
                if chipselect = '1' then
                        if address(7 downto 0)="00010001" then
                                numinput(1) <= (writedata(7 downto 4));
                        end if;
                end if;
        end if;
end if;
end process;

numinput3: process (clk_50)
begin
if rising_edge(clk_50) then
        if reset='1' then
        numinput(2) <= "0000";
        else
                if chipselect = '1' then
                        if address(7 downto 0)="00010001" then
                                numinput(2) <= (writedata(11 downto 8));
                        end if;
                end if;
        end if;
end if;
end process;

numinput4: process (clk_50)
begin
if rising_edge(clk_50) then
        if reset='1' then
                numinput(3) <= "0000";
        else
                if chipselect = '1' then
                        if address(7 downto 0)="00010001" then
                                numinput(3) <= (writedata(15 downto 12));
                        end if;
                end if;
        end if;
end if;
end process;


BALLPOS_H: process (clk_50)
begin
if rising_edge(clk_50) then
        if reset='1' then
                BALL_POS_H <= 0;
        else
                if chipselect = '1' then
```

```
                        if address(7 downto 0)="00000101" then
                                BALL_POS_H <= to_integer(writedata(15 downto 0));
                        end if;
                end if;
        end if;
end if;
end process;


BALLPOS_V: process (clk_50)
begin
if rising_edge(clk_50) then
        if reset='1' then
        BALL_POS_V <= 0;
        else
                if chipselect = '1' then
                        if address(7 downto 0)="00000110" then
                                BALL_POS_V <= to_integer(writedata(15 downto 0));
                        end if;
                end if;
        end if;
end if;
end process;


NOH: process (clk_50)
begin
if rising_edge(clk_50) then
        if reset='1' then
                numberofheart <= 0;
        else
                if chipselect = '1' then
                        if address(7 downto 0)="00010000" then
                                numberofheart <= to_integer(writedata(15 downto 0));
                        end if;
                end if;
        end if;
end if;
end process;


NOH2 : process (clk)
begin
        if rising_edge(clk) then
```

```
            case numberofheart is
                  when 4 =>
                        heartnum(3) <='1';
                        heartnum(2) <='1';
                        heartnum(1) <='1';
                        heartnum(0) <='1';
                  when 3 =>
                        heartnum(3) <='0';
                        heartnum(2) <='1';
                        heartnum(1) <='1';
                        heartnum(0) <='1';
                  when 2 =>
                        heartnum(3) <='0';
                        heartnum(2) <='0';
                        heartnum(1) <='1';
                        heartnum(0) <='1';
                  when 1 =>
                        heartnum(3) <='0';
                        heartnum(2) <='0';
                        heartnum(1) <='0';
                        heartnum(0) <='1';
                  when 0 =>
                        heartnum(3) <='0';
                        heartnum(2) <='0';
                        heartnum(1) <='0';
                        heartnum(0) <='0';
                  when others =>
                  end case;
      end if;
end process;


YUNPOS: process (clk_50)
begin
if rising_edge(clk_50) then
      if reset='1' then
      YUN_POS <= 0;
      else
            if chipselect = '1' then
                  if address(7 downto 0)="00000111" then
                        YUN_POS <= to_integer(writedata(15 downto 0));
                  end if;
            end if;
      end if;
end if;
end process;


lifePOSh: process (clk_50)
begin
if rising_edge(clk_50) then
	if reset='1' then
	loselife_POS_h <= 0;
	else
		if chipselect = '1' then
			if address(7 downto 0)="00000010" then
				loselife_POS_h <= to_integer(writedata(15 downto 0));
			end if;
		end if;
	end if;
end if;
end process;


lifePOSv: process (clk_50)
begin
if rising_edge(clk_50) then
	if reset='1' then
	loselife_POS_v <= 0;
	else
		if chipselect = '1' then
			if address(7 downto 0)="00000011" then
				loselife_POS_v <= to_integer(writedata(15 downto 0));
			end if;
		end if;
	end if;
end if;
end process;


speeduph: process (clk_50)
begin
if rising_edge(clk_50) then
	if reset='1' then
	speedup_POS_h <= 0;
	else
		if chipselect = '1' then
			if address(7 downto 0)="00001010" then
```

```
                        speedup_POS_h <= to_integer(writedata(15 downto 0));
                end if;
            end if;
        end if;
end if;
end process;


speedupv: process (clk_50)
begin
if rising_edge(clk_50) then
        if reset='1' then
        speedup_POS_v <= 0;
        else
            if chipselect = '1' then
                if address(7 downto 0)="00001011" then
                        speedup_POS_v <= to_integer(writedata(15 downto 0));
                end if;
            end if;
        end if;
end if;
end process;


YUNPOS1: process (clk_50)
begin
if rising_edge(clk_50) then
        if reset='1' then
        YUN_POS1 <= 0;
        else
            if chipselect = '1' then
                if address(7 downto 0)="00001001" then
                        YUN_POS1 <= to_integer(writedata(15 downto 0));
                end if;
            end if;
        end if;
end if;
end process;





    begin
```

```
        if rising_edge(clk) then
          RECTANGLE_HSTART <= to_integer(Hcount) - HSYNC - HBACK_PORCH;
            if reset = '1' then
              rectangle_h <= '1';
              elsif RECTANGLE_HSTART >= 0 and RECTANGLE_HSTART <= 639 then
                rectangle_h <= '1';
            else
                rectangle_h <= '0';
              end if;
        end if;
    end process RectangleHGen;

    RectangleVGen : process (clk)
    begin
        if rising_edge(clk) then
       RECTANGLE_VSTART <= to_integer(Vcount) - VSYNC - VBACK_PORCH;
            if reset = '1' then
                rectangle_v <= '0';
              elsif EndOfLine = '1' then
                if RECTANGLE_VSTART >=0 and RECTANGLE_VSTART <=479 then
                    rectangle_v <= '1';
                else
                    rectangle_v <= '0';
                end if;
            end if;
        end if;
    end process RectangleVGen;


RectangleHGen_loselife : process (clk)
    begin
        if rising_edge(clk) then
            if reset = '1' then
              rectangle_h_loselife <= '1';
              elsif Hcount >= HSYNC + HBACK_PORCH + loselife_POS_H and Hcount <= HSYNC +
HBACK_PORCH + loselife_POS_H + 14 then
                          rectangle_h_loselife <= '1';
                          numh_loselife    <=    to_integer(Hcount)    -    HSYNC    -    HBACK_PORCH    -
loselife_POS_H ;
                        elsif Hcount >= HSYNC + HBACK_PORCH + loselife_POS_H + 15    then
                          rectangle_h_loselife <= '0';
                          numh_loselife<= 0;
            end if;
        end if;
    end process RectangleHGen_loselife;

    RectangleVGen_loselife : process (clk)
    begin
        if rising_edge(clk) then
            if reset = '1' then
                    rectangle_v_loselife <= '0';
                    elsif EndOfLine = '1' then
              if Vcount >= VSYNC + VBACK_PORCH + loselife_POS_V and Vcount <= VSYNC +
VBACK_PORCH + loselife_POS_V+20 then
                    rectangle_v_loselife <= '1';
                    numv_loselife <= to_integer(Vcount) - VSYNC - VBACK_PORCH - loselife_POS_V ;
              elsif Vcount >= VSYNC + VBACK_PORCH + loselife_POS_V+21 then
                    rectangle_v_loselife <= '0';
                    numv_loselife <= 0;
              end if;
            end if;
        end if;
    end process RectangleVGen_loselife;

RectangleHGen_blackhole1 : process (clk)
    begin
        if rising_edge(clk) then
            if reset = '1' then
                    rectangle_h_blackhole1 <= '1';
                    elsif Hcount >= HSYNC + HBACK_PORCH + 49 and Hcount <= HSYNC +
HBACK_PORCH + 49 + 30 then
                            rectangle_h_blackhole1 <= '1';
                            numh_blackhole1 <= to_integer(Hcount) - HSYNC - HBACK_PORCH - 49 ;
                    elsif Hcount >= HSYNC + HBACK_PORCH + 50+ 30    then
                            rectangle_h_blackhole1 <= '0';
                            numh_blackhole1<= 0;
            end if;
        end if;
    end process RectangleHGen_blackhole1;

    RectangleVGen_blackhole1 : process (clk)
    begin
        if rising_edge(clk) then
            if reset = '1' then
              rectangle_v_blackhole1 <= '0';
                  elsif EndOfLine = '1' then
                            if Vcount >= VSYNC + VBACK_PORCH + 49 and Vcount <= VSYNC +
VBACK_PORCH + 49+29 then
```

```
                        rectangle_v_blackhole1 <= '1';
                        numv_blackhole1 <= to_integer(Vcount) - VSYNC - VBACK_PORCH - 49 ;
                     elsif Vcount >= VSYNC + VBACK_PORCH + 50+30 then
                        rectangle_v_blackhole1 <= '0';
                        numv_blackhole1 <= 0;
                end if;
             end if;
          end if;
       end process RectangleVGen_blackhole1;


    RectangleHGen_blackhole2 : process (clk)
       begin
          if rising_edge(clk) then
             if reset = '1' then
                     rectangle_h_blackhole2 <= '1';
                            elsif Hcount >= HSYNC + HBACK_PORCH + 550 and Hcount <=
    HSYNC + HBACK_PORCH + 550 + 29 then
                                rectangle_h_blackhole2 <= '1';
                                numh_blackhole2 <= to_integer(Hcount) - HSYNC -
    HBACK_PORCH - 550 ;
                            elsif Hcount >= HSYNC + HBACK_PORCH + 550 + 30    then
                     rectangle_h_blackhole2 <= '0';
                 numh_blackhole2<= 0;
             end if;
          end if;
       end process RectangleHGen_blackhole2;


       RectangleVGen_blackhole2 : process (clk)
       begin
          if rising_edge(clk) then
             if reset = '1' then
               rectangle_v_blackhole2 <= '0';
                elsif EndOfLine = '1' then
                     if Vcount >= VSYNC + VBACK_PORCH + 49 and Vcount <= VSYNC +
    VBACK_PORCH + 49+29 then
                        rectangle_v_blackhole2 <= '1';
                        numv_blackhole2 <= to_integer(Vcount) - VSYNC - VBACK_PORCH - 49 ;
                     elsif Vcount >= VSYNC + VBACK_PORCH + 50+30 then
                        rectangle_v_blackhole2 <= '0';
                        numv_blackhole2 <= 0;
                     end if;
             end if;
```

```
        end if;
    end process RectangleVGen_blackhole2;



RectangleHGen_blackhole3 : process (clk)
    begin
        if rising_edge(clk) then
            if reset = '1' then
                    rectangle_h_blackhole3 <= '1';
                        elsif Hcount >= HSYNC + HBACK_PORCH + 420 and Hcount <= HSYNC +
HBACK_PORCH + 420 + 29 then
                            rectangle_h_blackhole3 <= '1';
                            numh_blackhole3 <= to_integer(Hcount) - HSYNC - HBACK_PORCH - 420 ;
                        elsif Hcount >= HSYNC + HBACK_PORCH + 421 + 30    then
                    rectangle_h_blackhole3 <= '0';
                    numh_blackhole3<= 0;
            end if;
        end if;
    end process RectangleHGen_blackhole3;


  RectangleVGen_blackhole3 : process (clk)
    begin
        if rising_edge(clk) then
            if reset = '1' then
              rectangle_v_blackhole3 <= '0';
    elsif EndOfLine = '1' then
                    if Vcount >= VSYNC + VBACK_PORCH + 320 and Vcount <= VSYNC +
VBACK_PORCH + 320+29 then
                        rectangle_v_blackhole3 <= '1';
                        numv_blackhole3 <= to_integer(Vcount) - VSYNC - VBACK_PORCH - 320 ;
                    elsif Vcount >= VSYNC + VBACK_PORCH + 321+ 29 then
                        rectangle_v_blackhole3 <= '0';
                        numv_blackhole3 <= 0;
                end if;
            end if;
        end if;
    end process RectangleVGen_blackhole3;
```

```
RectangleHGen_speedup : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '1' then
           rectangle_h_speedup <= '1';
        elsif Hcount >= HSYNC + HBACK_PORCH + speedup_POS_H and Hcount <= HSYNC +
HBACK_PORCH + speedup_POS_H + 14 then
             rectangle_h_speedup <= '1';
           numh_speedup <= to_integer(Hcount) - HSYNC - HBACK_PORCH - speedup_POS_H ;
         elsif Hcount = HSYNC + HBACK_PORCH + speedup_POS_H + 15    then
             rectangle_h_speedup <= '0';
           numh_speedup<= 0;
         end if;
      end if;
   end process RectangleHGen_speedup;


   RectangleVGen_speedup : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '1' then
           rectangle_v_speedup <= '0';
        elsif EndOfLine = '1' then
           if Vcount >= VSYNC + VBACK_PORCH + speedup_POS_V and Vcount <= VSYNC +
VBACK_PORCH + speedup_POS_V+14 then
           rectangle_v_speedup <= '1';
           numv_speedup <= to_integer(Vcount) - VSYNC - VBACK_PORCH - speedup_POS_V ;
           elsif Vcount >= VSYNC + VBACK_PORCH + speedup_POS_V+15 then
             rectangle_v_speedup <= '0';
           numv_speedup <= 0;
           end if;
         end if;
      end if;
   end process RectangleVGen_speedup;


RectangleHGen_heart : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '1' then
           rectangle_h_heart <= '1';
           elsif Hcount >= HSYNC + HBACK_PORCH and Hcount <= HSYNC + HBACK_PORCH
+ 79 then
             rectangle_h_heart <= '1';
```

```
            numh_heart1 <= to_integer(Hcount) - HSYNC - HBACK_PORCH ;
            elsif Hcount >= HSYNC + HBACK_PORCH    + 80    then
              rectangle_h_heart <= '0';
            numh_heart1<= 0;
          end if;
      end if;
   end process RectangleHGen_heart;


numh_heart<=numh_heart1 mod 20;
numh_heart2<=numh_heart1/20;






   RectangleVGen_heart : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '1' then
           rectangle_v_heart <= '0';
           elsif EndOfLine = '1' then
           if  Vcount  >=  VSYNC  +  VBACK_PORCH  +  32  and  Vcount  <=  VSYNC  +
VBACK_PORCH + 32+19 then
           rectangle_v_heart <= '1';
           numv_heart <= to_integer(Vcount) - VSYNC - VBACK_PORCH - 32;
           elsif Vcount >= VSYNC + VBACK_PORCH + 33 + 20 then
             rectangle_v_heart <= '0';
           numv_heart <= 0;
end if;
         end if;
      end if;
   end process RectangleVGen_heart;




RectangleHGen_yun1 : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '1' then
           rectangle_h_yun1 <= '1';
           elsif  Hcount  >=  HSYNC  +  HBACK_PORCH  +  YUN_POS  and  Hcount  <=  HSYNC  +
HBACK_PORCH + YUN_POS + 57 then
             rectangle_h_yun1 <= '1';
            numh_yun1 <= to_integer(Hcount) - HSYNC - HBACK_PORCH - YUN_POS ;
         elsif Hcount >= HSYNC + HBACK_PORCH + YUN_POS +58    then
            rectangle_h_yun1 <= '0';
           numh_yun1<= 0;
         end if;
      end if;
   end process RectangleHGen_yun1;


   RectangleVGen_yun1 : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '1' then
                  rectangle_v_yun1 <= '0';
              elsif EndOfLine = '1' then
                if Vcount >= VSYNC + VBACK_PORCH + 5 and Vcount <= VSYNC +
VBACK_PORCH + 5+33 then
                  rectangle_v_yun1 <= '1';
                  numv_yun1 <= to_integer(Vcount) - VSYNC - VBACK_PORCH - 5 ;
                elsif Vcount >= VSYNC + VBACK_PORCH + 5+34 then
                  rectangle_v_yun1 <= '0';
                  numv_yun1 <= 0;
                end if;
         end if;
      end if;
   end process RectangleVGen_yun1;



RectangleHGen_back : process (clk)
   begin
           if reset = '1' then
                rectangle_h_back <= '1';
           elsif Hcount >= HSYNC + HBACK_PORCH + 0 and Hcount <= HSYNC +
HBACK_PORCH + YUN_POS + 639 then
                rectangle_h_back <= '1';
                numh_back <= ((to_integer(Hcount) - HSYNC - HBACK_PORCH))mod 160 ;
           elsif Hcount >= HSYNC + HBACK_PORCH + YUN_POS + 640    then
                rectangle_h_back <= '0';
                numh_back<= 0;
         end if;
   end process RectangleHGen_back;
```

```
RectangleVGen_back : process (clk)
    begin
              if reset = '1' then
                     rectangle_v_back <= '1';
              elsif Vcount >= VSYNC + VBACK_PORCH and Vcount <= VSYNC + VBACK_PORCH
+ 479 then
                     rectangle_v_back <= '1';
                     numv_back <= ((to_integer(Vcount) - VSYNC - VBACK_PORCH))mod 160;
              elsif Vcount >= VSYNC + VBACK_PORCH + 480 then
                     rectangle_v_back <= '0';
                     numv_back<= 0;
            end if;
    end process RectangleVGen_back;
numh_back_3<=numh_back_1 mod 160;

RectangleHGen_yun : process (clk)
    begin
        if rising_edge(clk) then
            if reset = '1' then
              rectangle_h_yun <= '1';
              elsif Hcount >= HSYNC + HBACK_PORCH + YUN_POS and Hcount <= HSYNC +
HBACK_PORCH + YUN_POS + 57 then
                rectangle_h_yun <= '1';
                numh_yun <= to_integer(Hcount) - HSYNC - HBACK_PORCH - YUN_POS ;
            elsif Hcount >= HSYNC + HBACK_PORCH + YUN_POS +58    then
                rectangle_h_yun <= '0';
              numh_yun<= 0;
            end if;
        end if;
    end process RectangleHGen_yun;


    RectangleVGen_yun : process (clk)
    begin
        if rising_edge(clk) then
            if reset = '1' then
                     rectangle_v_yun <= '0';
                  elsif EndOfLine = '1' then
                     if Vcount >= VSYNC + VBACK_PORCH + 440 and Vcount <= VSYNC +
VBACK_PORCH + 440+33 then
                         rectangle_v_yun <= '1';
                         numv_yun <= to_integer(Vcount) - VSYNC - VBACK_PORCH - 440 ;
                     elsif Vcount >= VSYNC + VBACK_PORCH + 440+34 then
                         rectangle_v_yun <= '0';
```

```
                    numv_yun <= 0;
                end if;
           end if;
        end if;
    end process RectangleVGen_yun;



RectangleHGen_yun2 : process (clk)
    begin
        if rising_edge(clk) then
            if reset = '1' then
              rectangle_h_yun2 <= '1';
              elsif  Hcount  >=  HSYNC  +  HBACK_PORCH  +5  and  Hcount  <=  HSYNC  +
HBACK_PORCH +5+33then
                rectangle_h_yun2 <= '1';
                numv_yun2 <= to_integer(Hcount) - HSYNC - HBACK_PORCH-5 ;
            elsif Hcount >= HSYNC + HBACK_PORCH + 5+34 then
                rectangle_h_yun2 <= '0';
              numv_yun2<= 0;
            end if;
        end if;
    end process RectangleHGen_yun2;


    RectangleVGen_yun2 : process (clk)
    begin
        if rising_edge(clk) then
            if reset = '1' then
                    rectangle_v_yun2 <= '0';
                 elsif EndOfLine = '1' then
                    if Vcount >= VSYNC + VBACK_PORCH + YUN_POS1+2 and Vcount <= VSYNC +
VBACK_PORCH    + YUN_POS1 +2+55 then
                        rectangle_v_yun2 <= '1';
                        numh_yun2 <= to_integer(Vcount) - VSYNC - VBACK_PORCH - YUN_POS1 ;
                    elsif Vcount >= VSYNC + VBACK_PORCH    + YUN_POS1 +2 + 56 then
                        rectangle_v_yun2 <= '0';
                           numh_yun2 <= 0;
                    end if;
            end if;
        end if;
    end process RectangleVGen_yun2;
```

```
RectangleHGen_yun3 : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '1' then
           rectangle_h_yun3 <= '1';
           elsif  Hcount  >=  HSYNC  +  HBACK_PORCH  +600  and  Hcount  <=  HSYNC  +
HBACK_PORCH +600+33then
             rectangle_h_yun3 <= '1';
             numv_yun3 <= to_integer(Hcount) - HSYNC - HBACK_PORCH-600;
         elsif Hcount >= HSYNC + HBACK_PORCH + 600+34 then
             rectangle_h_yun3 <= '0';
           numv_yun3<= 0;
         end if;
      end if;
   end process RectangleHGen_yun3;


   RectangleVGen_yun3 : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '1' then
                rectangle_v_yun3 <= '0';
                elsif EndOfLine = '1' then
                if Vcount >= VSYNC + VBACK_PORCH + YUN_POS1+2 and Vcount <= VSYNC +
VBACK_PORCH    + YUN_POS1+2+55 then
                        rectangle_v_yun3 <= '1';
                        numh_yun3   <=   to_integer(Vcount)   -   VSYNC   -   VBACK_PORCH   -
YUN_POS1 ;
                elsif Vcount >= VSYNC + VBACK_PORCH    + YUN_POS1+2 + 56 then
                     rectangle_v_yun3 <= '0';
                     numh_yun3 <= 0;
                end if;
         end if;
      end if;
   end process RectangleVGen_yun3;




RectangleHGen_ball : process (clk)
   begin
      if rising_edge(clk) then
         if reset = '1' then
```

```
              rectangle_h_ball <= '1';
              elsif Hcount >= HSYNC + HBACK_PORCH + BALL_POS_H and Hcount <= HSYNC +
HBACK_PORCH + BALL_POS_H + 15 then
                rectangle_h_ball <= '1';
                numh_ball <= to_integer(Hcount) - HSYNC - HBACK_PORCH - BALL_POS_H;
            elsif Hcount >= HSYNC + HBACK_PORCH + BALL_POS_H + 16 then
                rectangle_h_ball <= '0';
              numh_ball<= 0;
            end if;
        end if;
    end process RectangleHGen_ball;

  RectangleVGen_ball : process (clk)
    begin
        if rising_edge(clk) then
            if reset = '1' then
                rectangle_v_ball <= '0';
            elsif EndOfLine = '1' then
                if Vcount >= VSYNC + VBACK_PORCH + BALL_POS_V and Vcount <= VSYNC +
VBACK_PORCH + BALL_POS_V+14 then
                    rectangle_v_ball <= '1';
                  numv_ball <= to_integer(Vcount) - VSYNC - VBACK_PORCH - BALL_POS_V ;
                elsif Vcount >= VSYNC + VBACK_PORCH + BALL_POS_V+15 then
                    rectangle_v_ball <= '0';
                     numv_ball <= 0;
                end if;
            end if;
        end if;
    end process RectangleVGen_ball;

rectangle_ball <= rectangle_h_ball and rectangle_v_ball;
rectangle_yun <= rectangle_h_yun and rectangle_v_yun;
rectangle_yun1 <= rectangle_h_yun1 and rectangle_v_yun1;
rectangle_yun2 <= rectangle_h_yun2 and rectangle_v_yun2;
rectangle_yun3 <= rectangle_h_yun3 and rectangle_v_yun3;
rectangle <= rectangle_h and rectangle_v;
rectangle_back <= rectangle_v_back and rectangle_h_back;
rectangle_blackhole1 <= rectangle_v_blackhole1 and rectangle_h_blackhole1;
rectangle_blackhole2 <= rectangle_v_blackhole2 and rectangle_h_blackhole2;
rectangle_blackhole3 <= rectangle_v_blackhole3 and rectangle_h_blackhole3;
rectangle_loselife <= rectangle_v_loselife and rectangle_h_loselife;
rectangle_heart <= rectangle_v_heart and rectangle_h_heart;
rectangle_speedup <= rectangle_v_speedup and rectangle_h_speedup;
rectangle_numb <= rectangle_h_numb and rectangle_v_numb;
```

```
            HPOS <= to_integer(Hcount) - HSYNC - HBACK_PORCH;
            VPOS <= to_integer(Vcount) - VSYNC - VBACK_PORCH;
            numh_bug <= HPOS mod 40;
            numv_bug <= VPOS mod 40;
            numh_bug_flag <= HPOS/40;
            numv_bug_flag <= VPOS/40;

RGB2 : process (clk_50)
begin
  if rising_edge(clk_50) then
                        if virb_sta1="1" and h_virb1=numh_bug_flag and v_virb1=numv_bug_flag then

                        case bugdie_sta is
                                    when "00" =>
                                          OUT_R<=bugdie2(to_integer(bugdie2_RGB)) ;
                                          OUT_G<=bugdie2(to_integer(bugdie2_RGB)) ;
                                          OUT_B<=bugdie2(to_integer(bugdie2_RGB)) ;
                                    when "01" =>
                                          OUT_R<=bugdie3(to_integer(bugdie3_RGB)) ;
                                          OUT_G<=bugdie3(to_integer(bugdie3_RGB)) ;
                                          OUT_B<=bugdie3(to_integer(bugdie3_RGB)) ;
                                    when others =>

                                    end case;

                        else


                        case background(numh_bug_flag)(numv_bug_flag) is
                        when "1000" =>
                                    if BUG_STA1 = 0 then
                                          OUT_R<=bug_pink1(to_integer(bug_pink1_RGB))(23     downto
16) ;
                                          OUT_G<=bug_pink1(to_integer(bug_pink1_RGB))(15     downto
8) ;
                                          OUT_B<=bug_pink1(to_integer(bug_pink1_RGB))(7 downto 0) ;
                                    else
                                          OUT_R<=bug_pink2(to_integer(bug_pink2_RGB))(23     downto
16) ;
                                          OUT_G<=bug_pink2(to_integer(bug_pink2_RGB))(15     downto
8) ;
                                          OUT_B<=bug_pink2(to_integer(bug_pink2_RGB))(7 downto 0) ;
                                    end if;
```

```
                      when "1001"    =>

                                 if BUG_STA2 = 0 then
                                      OUT_R<=bug_yellow1(to_integer(bug_yellow1_RGB))(23
downto 16) ;
                                      OUT_G<=bug_yellow1(to_integer(bug_yellow1_RGB))(15
downto 8) ;
                                      OUT_B<=bug_yellow1(to_integer(bug_yellow1_RGB))(7
downto 0) ;
                                 else
                                      OUT_R<=bug_yellow2(to_integer(bug_yellow2_RGB))(23
downto 16) ;
                                      OUT_G<=bug_yellow2(to_integer(bug_yellow2_RGB))(15
downto 8) ;
                                      OUT_B<=bug_yellow2(to_integer(bug_yellow2_RGB))(7
downto 0) ;
                                 end if;

                      when "1010" =>
                                 if BUG_STA3 = 0 then
                                      OUT_R<=bug_green1(to_integer(bug_green1_RGB))(23  downto
16) ;
                                      OUT_G<=bug_green1(to_integer(bug_green1_RGB))(15  downto
8) ;
                                      OUT_B<=bug_green1(to_integer(bug_green1_RGB))(7    downto
0) ;
                                 else
                                      OUT_R<=bug_green2(to_integer(bug_green2_RGB))(23  downto
16) ;
                                      OUT_G<=bug_green2(to_integer(bug_green2_RGB))(15  downto
8) ;
                                      OUT_B<=bug_green2(to_integer(bug_green2_RGB))(7    downto
0) ;
                                 end if;



                      when "1011" =>
                                 if BUG_STA4 = 0 then
                                      OUT_R<=bug_purple1(to_integer(bug_purple1_RGB))(23
downto 16) ;
                                      OUT_G<=bug_purple1(to_integer(bug_purple1_RGB))(15
downto 8) ;
```

```
                                                OUT_B<=bug_purple1(to_integer(bug_purple1_RGB))(7 downto
0) ;
                                        else
                                                OUT_R<=bug_purple2(to_integer(bug_purple2_RGB))(23
downto 16) ;
                                                OUT_G<=bug_purple2(to_integer(bug_purple2_RGB))(15
downto 8) ;
                                                OUT_B<=bug_purple2(to_integer(bug_purple2_RGB))(7 downto
0) ;
                                        end if;



                        when "1100" =>

                                        if BUG_STA1 = 0 then
                                                OUT_R<=bug_pink1(to_integer(bug_pink1_RGB))(15     downto
8) ;
                                                OUT_G<=bug_pink1(to_integer(bug_pink1_RGB))(7 downto 0);
                                                OUT_B<=bug_pink1(to_integer(bug_pink1_RGB))(23     downto
16) ;
                                        else
                                                OUT_R<=bug_pink2(to_integer(bug_pink2_RGB))(15     downto
8) ;
                                                OUT_G<=bug_pink2(to_integer(bug_pink2_RGB))(7 downto 0);
                                                OUT_B<=bug_pink2(to_integer(bug_pink2_RGB))(23     downto
16) ;
                                        end if;



                        when "1101" =>

                                        if BUG_STA3 = 0 then
                                                OUT_R<=bug_purple1(to_integer(bug_purple1_RGB))(15
downto 8) ;
                                                OUT_G<=bug_purple1(to_integer(bug_purple1_RGB))(7 downto
0);
                                                OUT_B<=bug_purple1(to_integer(bug_purple1_RGB))(23
downto 16) ;
                                        else
                                        OUT_R<=bug_purple2(to_integer(bug_purple2_RGB))(15
downto 8) ;
                                        OUT_G<=bug_purple2(to_integer(bug_purple2_RGB))(7 downto
0);
                                        OUT_B<=bug_purple2(to_integer(bug_purple2_RGB))(23
downto 16) ;
                                    end if;




                        when "1110" =>

                                    if BUG_STA4 = 0 then
                                        OUT_R<=bug_yellow1(to_integer(bug_yellow1_RGB))(15
downto 8) ;
                                        OUT_G<=bug_yellow1(to_integer(bug_yellow1_RGB))(7
downto 0);
                                        OUT_B<=bug_yellow1(to_integer(bug_yellow1_RGB))(23
downto 16) ;
                                    else
                                        OUT_R<=bug_yellow2(to_integer(bug_yellow2_RGB))(15
downto 8) ;
                                        OUT_G<=bug_yellow2(to_integer(bug_yellow2_RGB))(7
downto 0);
                                        OUT_B<=bug_yellow2(to_integer(bug_yellow2_RGB))(23
downto 16) ;
                                    end if;




                        when "1111" =>

                                    if BUG_STA2 = 0 then
                                        OUT_R<=bug_green1(to_integer(bug_green1_RGB))(15  downto
8) ;
                                        OUT_G<=bug_green1(to_integer(bug_green1_RGB))(7    downto
0);
                                        OUT_B<=bug_green1(to_integer(bug_green1_RGB))(23  downto
16) ;
                                    else
```

```
                                        OUT_R<=bug_green2(to_integer(bug_green2_RGB))(15  downto
8) ;
                                        OUT_G<=bug_green2(to_integer(bug_green2_RGB))(7    downto
0);
                                        OUT_B<=bug_green2(to_integer(bug_green2_RGB))(23  downto
16) ;
                                    end if;
                        when others =>
                          end case;
                          end if;
                     end if;

end process RGB2;

Rpink1 : work.bug_pink1_rom
port map(
       clk => clk,
    h_addr => numh_bug,
    v_addr => numv_bug,
    dataout => bug_pink1_RGB
);

Rpink2 : work.bug_pink2_rom
port map(
       clk => clk,
    h_addr => numh_bug,
    v_addr => numv_bug,
    dataout => bug_pink2_RGB
);


Ryellow1 : work.bug_yellow1_rom
port map(
       clk => clk,
    h_addr => numh_bug,
    v_addr => numv_bug,
    dataout => bug_yellow1_RGB
);

Ryellow2 : work.bug_yellow2_rom
port map(
       clk => clk,
    h_addr => numh_bug,
    v_addr => numv_bug,
    dataout => bug_yellow2_RGB
);


Rpurple1 : work.bug_purple1_rom
port map(
       clk => clk,
    h_addr => numh_bug,
    v_addr => numv_bug,
    dataout => bug_purple1_RGB
);

Rpurple2 : work.bug_purple2_rom
port map(
       clk => clk,
    h_addr => numh_bug,
    v_addr => numv_bug,
    dataout => bug_purple2_RGB
);

Rgreen1 : work.bug_green1_rom
port map(
       clk => clk,
    h_addr => numh_bug,
    v_addr => numv_bug,
    dataout => bug_green1_RGB
);

Rgreen2 : work.bug_green2_rom
port map(
       clk => clk,
    h_addr => numh_bug,
    v_addr => numv_bug,
    dataout => bug_green2_RGB
);


pyun1 : work.xiaoyun1_rom
port map(
       clk => clk,
    h_addr => numh_yun,
    v_addr => numv_yun,
    dataout => yun1_RGB
);
```

```
pyun2 : work.xiaoyun2_rom
port map(
      clk => clk,
   h_addr => numh_yun,
   v_addr => numv_yun,
   dataout => yun2_RGB
);

pyun3 : work.xiaoyun3_rom
port map(
      clk => clk,
   h_addr => numh_yun,
   v_addr => numv_yun,
   dataout => yun3_RGB
);

pyun11 : work.xiaoyun1_rom
port map(
      clk => clk,
   h_addr => numh_yun1,
   v_addr => numv_yun1,
   dataout => yun11_RGB
);

pyun12 : work.xiaoyun2_rom
port map(
      clk => clk,
   h_addr => numh_yun1,
   v_addr => numv_yun1,
   dataout => yun12_RGB
);

pyun13 : work.xiaoyun3_rom
port map(
      clk => clk,
   h_addr => numh_yun1,
   v_addr => numv_yun1,
   dataout => yun13_RGB
);

pyun21 : work.xiaoyun1_rom
port map(
      clk => clk,
    h_addr => numh_yun2,
    v_addr => numv_yun2,
    dataout => yun21_RGB
);

pyun22 : work.xiaoyun3_rom
port map(
       clk => clk,
    h_addr => numh_yun2,
    v_addr => numv_yun2,
    dataout => yun22_RGB
);

pyun23 : work.xiaoyun3_rom
port map(
       clk => clk,
    h_addr => numh_yun2,
    v_addr => numv_yun2,
    dataout => yun23_RGB
);

pyun31 : work.xiaoyun1_rom
port map(
       clk => clk,
    h_addr => numh_yun3,
    v_addr => numv_yun3,
    dataout => yun31_RGB
);

pyun32 : work.xiaoyun2_rom
port map(
       clk => clk,
    h_addr => numh_yun3,
    v_addr => numv_yun3,
    dataout => yun32_RGB
);

pyun33 : work.xiaoyun3_rom
port map(
       clk => clk,
    h_addr => numh_yun3,
    v_addr => numv_yun3,
    dataout => yun33_RGB
);
```

```
pball : work.ball_rom
port map(
	clk => clk,
    h_addr => numh_ball,
    v_addr => numv_ball,
    dataout => ball_RGB
);

pbd2 : work.bugdie2
port map(
	clk => clk,
    h_addr => numh_bug,
    v_addr => numv_bug,
    dataout => bugdie2_RGB
);

pbd3 : work.bugdie3
port map(
    clk => clk,
    h_addr => numh_bug,
    v_addr => numv_bug,
    dataout => bugdie3_RGB
);

bdtm : work.timer_bugdie
port map(
    clk => clk,
    output => bugdie_sta
);

--tmd1 : work.timerfordie
--port map(
--clk=>clk,
--reset=>reset1 and (not reset11),
--output=>virb_sta1
--);

tmd2 : work.timerfordie
port map(
clk=>clk,
reset=>reset2 and (not reset22),
output=>virb_sta2
);
```

```
tmd3 : work.timerfordie
port map(
clk=>clk,
reset=>reset3 and (not reset33),
output=>virb_sta3
);

tmd4 : work.timerfordie
port map(
clk=>clk,
reset=>reset4 and (not reset44),
output=>virb_sta4
);

tmd5 : work.timer_ball
port map(
clk=>clk,
output=>ball_sta
);

backg : work.backround_rom
port map(
clk => clk,
    h_addr => numh_back,
    v_addr => numv_back,
    dataout => back_RGB
);

hert : work.heart_rom
port map(
clk => clk,
    h_addr => numh_heart,
    v_addr => numv_back,
    dataout => heart_RGB
);

speedu : work.speedup_rom
port map(
clk => clk,
    h_addr => numh_speedup,
    v_addr => numv_speedup,
    dataout => speedup_RGB
);
```

```
losel : work.loselife_rom
port map(
clk => clk,
   h_addr => numh_loselife,
   v_addr => numv_loselife,
   dataout => loselife_RGB
);





   VideoOut: process (clk, reset)
   begin
      if reset = '1' then
              VGA_R <= "0000000000";
              VGA_G <= "0000000000";
              VGA_B <= "0000000000";
      elsif clk'event and clk = '1' then

if rectangle = '1'    and (
((numh_bug<=30 and numv_bug <=29 and numh_bug >=1)
  and background(numh_bug_flag)(numv_bug_flag)(3)='1'
and (OUT_R <=X"F5" or OUT_G <=X"F5" or OUT_B <=X"F5"))
or
(
(numh_bug<=36 and numv_bug <=34 and numh_bug >=2)
and
(
(virb_sta1="1" and h_virb1 = numh_bug_flag and v_virb1 = numv_bug_flag)

)

and(OUT_G <=X"EE")
)
)
then
            VGA_R(9 downto 2) <= OUT_R ;
            VGA_R(1 downto 0) <= "11";
            VGA_G(9 downto 2) <= OUT_G ;
            VGA_G(1 downto 0) <= "11";
            VGA_B(9 downto 2) <= OUT_B ;
            VGA_B(1 downto 0) <= "11";
elsif rectangle_ball = '1' and (ball(to_integer(ball_RGB))(23 downto 16) <=X"F0" or
ball(to_integer(ball_RGB))(15 downto 8) <=X"F0" or ball(to_integer(ball_RGB))(7 downto 0)
<=X"F0") then

if Vcount >= VSYNC + VBACK_PORCH + BALL_POS_V and Vcount <= VSYNC +
VBACK_PORCH + BALL_POS_V+14 then


            VGA_R(9 downto 2) <= ball(to_integer(ball_RGB))(23 downto 16) ;
             VGA_R(1 downto 0) <= "11";
             VGA_G(9 downto 2) <= ball(to_integer(ball_RGB))(15 downto 8) ;
             VGA_G(1 downto 0) <= "11";
             VGA_B(9 downto 2) <= ball(to_integer(ball_RGB))(7 downto 0) ;
             VGA_B(1 downto 0) <= "11";
end if;
elsif rectangle_yun3 = '1' and (xiaoyun1(to_integer(yun32_RGB))(23 downto 16)<=X"F0" or
xiaoyun1(to_integer(yun32_RGB))(15 downto 8)<=X"F0" or xiaoyun1(to_integer(yun32_RGB))(7
downto 0)<=X"F0") then
if YUN_STA41 =1 then
             VGA_R(9 downto 2) <= xiaoyun3(to_integer(yun33_RGB))(23 downto 16) ;
             VGA_R(1 downto 0) <= "11";
             VGA_G(9 downto 2) <= xiaoyun3(to_integer(yun33_RGB))(15 downto 8) ;
             VGA_G(1 downto 0) <= "11";
             VGA_B(9 downto 2) <= xiaoyun3(to_integer(yun33_RGB))(7 downto 0) ;
             VGA_B(1 downto 0) <= "11";
else

             VGA_R(9 downto 2) <= xiaoyun1(to_integer(yun32_RGB))(23 downto 16) ;
             VGA_R(1 downto 0) <= "11";
             VGA_G(9 downto 2) <= xiaoyun1(to_integer(yun32_RGB))(15 downto 8) ;
             VGA_G(1 downto 0) <= "11";
             VGA_B(9 downto 2) <= xiaoyun1(to_integer(yun32_RGB))(7 downto 0) ;
             VGA_B(1 downto 0) <= "11";
end if;
elsif rectangle_yun2 = '1'and (xiaoyun1(to_integer(yun22_RGB))(23 downto 16)<=X"F0" or
xiaoyun1(to_integer(yun22_RGB))(15 downto 8)<=X"F0" or xiaoyun1(to_integer(yun22_RGB))(7
downto 0)<=X"F0" ) then
if YUN_STA31 =1 then
             VGA_R(9 downto 2) <= xiaoyun3(to_integer(yun23_RGB))(23 downto 16) ;
             VGA_R(1 downto 0) <= "11";
             VGA_G(9 downto 2) <= xiaoyun3(to_integer(yun23_RGB))(15 downto 8) ;
             VGA_G(1 downto 0) <= "11";
             VGA_B(9 downto 2) <= xiaoyun3(to_integer(yun23_RGB))(7 downto 0) ;
             VGA_B(1 downto 0) <= "11";
else
```

```
case YUN_STA is
when 0 =>

            VGA_R(9 downto 2) <= xiaoyun1(to_integer(yun22_RGB))(23 downto 16) ;
            VGA_R(1 downto 0) <= "11";
            VGA_G(9 downto 2) <= xiaoyun1(to_integer(yun22_RGB))(15 downto 8) ;
            VGA_G(1 downto 0) <= "11";
            VGA_B(9 downto 2) <= xiaoyun1(to_integer(yun22_RGB))(7 downto 0) ;
            VGA_B(1 downto 0) <= "11";
end if;
elsif rectangle_yun1 = '1' and (xiaoyun1(to_integer(yun11_RGB))(23 downto 16)<=X"F0" or
xiaoyun1(to_integer(yun11_RGB))(15 downto 8)<=X"F0" or xiaoyun1(to_integer(yun11_RGB))(7
downto 0)<=X"F0")    then
if YUN_STA21 =1 then
            VGA_R(9 downto 2) <= xiaoyun3(to_integer(yun13_RGB))(23 downto 16) ;
            VGA_R(1 downto 0) <= "11";
            VGA_G(9 downto 2) <= xiaoyun3(to_integer(yun13_RGB))(15 downto 8) ;
            VGA_G(1 downto 0) <= "11";
            VGA_B(9 downto 2) <= xiaoyun3(to_integer(yun13_RGB))(7 downto 0) ;
            VGA_B(1 downto 0) <= "11";
else


            VGA_R(9 downto 2) <= xiaoyun1(to_integer(yun11_RGB))(23 downto 16) ;
            VGA_R(1 downto 0) <= "11";
            VGA_G(9 downto 2) <= xiaoyun1(to_integer(yun11_RGB))(15 downto 8) ;
            VGA_G(1 downto 0) <= "11";
            VGA_B(9 downto 2) <= xiaoyun1(to_integer(yun11_RGB))(7 downto 0) ;
            VGA_B(1 downto 0) <= "11";
end if;

elsif rectangle_yun = '1'and (xiaoyun1(to_integer(yun1_RGB))(23 downto 16)<=X"F0" or
xiaoyun1(to_integer(yun1_RGB))(15 downto 8)<=X"F0" or xiaoyun1(to_integer(yun1_RGB))(7
downto 0)<=X"F0" ) then
if YUN_STA11 =1 then
VGA_R(9 downto 2) <= xiaoyun3(to_integer(yun3_RGB))(23 downto 16) ;
VGA_R(1 downto 0) <= "11";
VGA_G(9 downto 2) <= xiaoyun3(to_integer(yun3_RGB))(15 downto 8) ;
VGA_G(1 downto 0) <= "11";
VGA_B(9 downto 2) <= xiaoyun3(to_integer(yun3_RGB))(7 downto 0) ;
VGA_B(1 downto 0) <= "11";
else

VGA_R(9 downto 2) <= xiaoyun1(to_integer(yun1_RGB))(23 downto 16) ;
VGA_R(1 downto 0) <= "11";
VGA_G(9 downto 2) <= xiaoyun1(to_integer(yun1_RGB))(15 downto 8) ;
VGA_G(1 downto 0) <= "11";
VGA_B(9 downto 2) <= xiaoyun1(to_integer(yun1_RGB))(7 downto 0) ;
VGA_B(1 downto 0) <= "11";
end if;


elsif rectangle_heart = '1' and heartnum(numh_heart2)='1'and (heart(to_integer(heart_RGB))(23
downto 16)<=X"F0" or heart(to_integer(heart_RGB))(15 downto 8)<=X"F0" or
heart(to_integer(heart_RGB))(7 downto 0)<=X"F0")    then

            VGA_R(9 downto 2) <= heart(to_integer(heart_RGB))(23 downto 16) ;
            VGA_R(1 downto 0) <= "11";
            VGA_G(9 downto 2) <= heart(to_integer(heart_RGB))(15 downto 8) ;
            VGA_G(1 downto 0) <= "11";
            VGA_B(9 downto 2) <= heart(to_integer(heart_RGB))(7 downto 0) ;
            VGA_B(1 downto 0) <= "11";

elsif rectangle_loselife = '1'and (loselife(to_integer(loselife_RGB))(23 downto 16)<=X"F0" or
loselife(to_integer(loselife_RGB))(15 downto 8)<=X"F0" or loselife(to_integer(loselife_RGB))(7
downto 0)<=X"F0" ) then
            VGA_R(9 downto 2) <= loselife(to_integer(loselife_RGB))(23 downto 16) ;
            VGA_R(1 downto 0) <= "11";
            VGA_G(9 downto 2) <= loselife(to_integer(loselife_RGB))(15 downto 8) ;
            VGA_G(1 downto 0) <= "11";
            VGA_B(9 downto 2) <= loselife(to_integer(loselife_RGB))(7 downto 0) ;
            VGA_B(1 downto 0) <= "11";
elsif vga_hblank = '0' and vga_vblank ='0' then
                VGA_R <= "1111111111";
                VGA_G <= "1111111111";
                VGA_B <= "1111111111";
     else
                VGA_R <= "0000000000";
                VGA_G <= "0000000000";
                VGA_B <= "0000000000";
end if;
end if;
end process;


   VGA_CLK <= clk;
   VGA_HS <= not vga_hsync;
   VGA_VS <= not vga_vsync;
   VGA_SYNC <= '0';
   VGA_BLANK <= not (vga_hsync or vga_vsync);
end rtl;

de2_wm8731_audio.vhd
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity de2_wm8731_audio is
port (
        clk : in std_logic;              --    Audio CODEC Chip Clock AUD_XCK (18.43 MHz)
       reset_n : in std_logic;
        chipselect : in std_logic;
        --test_mode : in std_logic;              --       Audio CODEC controller test mode
        --audio_request : out std_logic;    --       Audio controller request new data
        --data : in unsigned(15 downto 0);

        writedata : in unsigned (15 downto 0);
        readdata : out unsigned(15 downto 0);
       irq : out std_logic;
       address : in std_logic_vector(15 downto 0);
       read, write: in std_logic;

        -- Audio interface signals
        AUD_ADCLRCK    : out    std_logic;      --       Audio CODEC ADC LR Clock
        AUD_ADCDAT     : in     std_logic;      --       Audio CODEC ADC Data
        AUD_DACLRCK    : out    std_logic;      --       Audio CODEC DAC LR Clock
        AUD_DACDAT     : out    std_logic;      --       Audio CODEC DAC Data
        AUD_BCLK          : inout std_logic    --       Audio CODEC Bit-Stream Clock
   );
end    de2_wm8731_audio;

architecture rtl of de2_wm8731_audio is

        signal lrck : std_logic;
        signal bclk : std_logic;
        signal xck    : std_logic;

        signal lrck_divider : unsigned(11 downto 0);
        signal bclk_divider : unsigned(3 downto 0);

        signal set_bclk : std_logic;
        signal set_lrck : std_logic;
        signal clr_bclk : std_logic;
        signal lrck_lat : std_logic;

        signal shift_out : unsigned(15 downto 0);

        signal sin_out          : unsigned(15 downto 0);
        signal sin_counter : unsigned(4 downto 0);
       signal buf_add : unsigned(4 downto 0) := "00000";
       signal input : unsigned(15 downto 0);
       type RAM_type is array (0 to 31) of unsigned(15 downto 0);
        signal buffer1 : RAM_type :=
  (

X"0000",
X"0fff",
X"0000",
X"ffff",
X"0000",
X"0fff",
X"0000",
X"ffff",
X"0000",
X"0fff",
X"0000",
X"ffff",
X"0000",
X"0fff",
X"0000",
X"ffff",
X"0000",
X"0fff",
X"0000",
X"ffff",
X"0000",
X"0fff",
X"0000",
X"ffff",
X"0000",
X"0fff",
X"0000",
X"ffff",
X"0000",
X"0fff",
X"0000",
X"ffff"
);

signal audio_clock : unsigned(1 downto 0) := "00";
signal clk25: std_logic;
signal clk6: std_logic;
signal audio_clock2 : unsigned(3 downto 0) := "0000";

begin

    process (CLk)
    begin
        if rising_edge(CLk) then
            audio_clock <= audio_clock + "1";
        end if;
    end process;
clk25<=audio_clock(1);

    process (CLk)
    begin
        if rising_edge(CLk) then
            audio_clock2 <= audio_clock2 + "1";
        end if;
    end process;
clk6<=audio_clock2(1);

    process (clk25)
    begin
        if rising_edge(clk25) then
            if reset_n = '0' then
                lrck_divider <= (others => '0');
       ----lrck_divider  =  X"1EB"  clk=25M,  sample  rate  6.3K,  actual  rate  25M/1EB  =  54K      irq
rate=32k/64 = 500
            elsif lrck_divider = X"320"    then                -- "C0" minus 1
                lrck_divider <= X"000";
            else
                lrck_divider <= lrck_divider + 1;
            end if;
        end if;
    end process;

    process (clk25)
    begin
        if rising_edge(clk25) then
            if reset_n = '0' then
                bclk_divider <= (others => '0');
            elsif bclk_divider = X"B" or set_lrck = '1'    then
                bclk_divider <= X"0";
            else
                bclk_divider <= bclk_divider + 1;
            end if;
        end if;
    end process;

    set_lrck <= '1' when lrck_divider = X"320" else '0';

    process (clk25)
    begin
        if rising_edge(clk25) then
            if reset_n = '0' then
                lrck <= '0';
            elsif set_lrck = '1' then
                lrck <= not lrck;
            end if;
        end if;
    end process;

    -- BCLK divider
    set_bclk <= '1' when bclk_divider(3 downto 0) = "0101" else '0';
    clr_bclk <= '1' when bclk_divider(3 downto 0) = "1011" else '0';

    process (clk25)
    begin
        if rising_edge(clk25) then
            if reset_n = '0' then
                bclk <= '0';
            elsif set_lrck = '1' or clr_bclk = '1' then
                bclk <= '0';
            elsif set_bclk = '1' then
                bclk <= '1';
            end if;
        end if;
    end process;

    -- Audio data shift output
    process (clk25)
    begin
        if rising_edge(clk25) then
```

```
        if reset_n = '0' then
            shift_out <= (others => '0');
        elsif set_lrck = '1' then
            --if test_mode = '1' then
                shift_out <= sin_out;
            --else
                --shift_out <= data;
            --end if;
        elsif clr_bclk = '1' then
            shift_out <= shift_out (14 downto 0) & '0';
        end if;
    end if;
end process;

    -- Audio outputs

    AUD_ADCLRCK    <= lrck;
    AUD_DACLRCK    <= lrck;
    AUD_DACDAT      <= shift_out(15);
    AUD_BCLK          <= bclk;

    -- Self test with Sin wave

    process(clk25)
    begin
        if rising_edge(clk25) then
            if reset_n = '0' then
                    sin_counter <= (others => '0');
            elsif lrck_lat = '1' and lrck = '0'    then
                if sin_counter = "11111" then
                    sin_counter <= "00000";
                else
                    sin_counter <= sin_counter + 1;
                end if;
            end if;
        end if;
    end process;

    process(clk25)
    begin
        if rising_edge(clk25) then
            lrck_lat <= lrck;
        end if;
    end process;
```

```
        process (clk25)
        begin
            if rising_edge(clk25) then
                if lrck_lat = '1' and lrck = '0' then
                    --audio_request <= '1';
                else
                    --audio_request <= '0';
                end if;
            end if;
        end process;

audio_input: process (clk)
begin
if rising_edge(clk) then
       --if reset_n='0' then
               --buf_add <= "000000";
       --else
              if chipselect = '1' then
                     if write ='1' then


                                                         --if address(7 downto 0)="0001110" then
                                buffer1(to_integer(unsigned(address))) <=     writedata(15 downto 0);
--                          if buf_add = "101111" then
--                                 buf_add <= "000000";
--                          else
--                                 buf_add <= buf_add + 1;
                     end if;

                     if read ='1' then

                                readdata(15 downto 0) <= buffer1(to_integer(unsigned(address)));
unsigned(address); --buffer1(



                     end if;
              end if;
       --end if;
end if;
end process;


process(clk25)
```

```
begin
    if rising_edge(clk25) then
       if reset_n='0' then
              irq <= '0';
       else
               if write = '1' and chipselect ='1' then
                     irq <= '0';
              elsif(sin_counter = "11111") then
                     irq <= '1';
              elsif not(sin_counter = "11111") then
                     irq <= '0';
              end if;
       end if;
end if;
end process;

  sin_out <= buffer1(to_integer(sin_counter));
--buffer1(to_integer(sin_counter))<= input;

end architecture;
```

C Code:

```
#include <io.h>
#include <system.h>
#include <stdio.h>
#include <stddef.h>

#define win_delay1 25000
#define win_delay2 80000
#define lose_delay1 3
#define lose_delay2 25000
#define lose_delay3 80000




 #define IOWR_VGA_DATA(base, offset, data) \
  IOWR_16DIRECT(base, (offset) * 2, data)

  #define IORD_IRCOMP_INST_DATA(base, offset) \
  IORD_16DIRECT(base, (offset) * 2)

  #define IOWR_SRAM(base, offset, data) \
  IOWR_16DIRECT(base, (offset) * 2, data)
#define IOWR_DE2_WM8731_AUDIO_INST_DATA(base, offset, data) \
  IOWR_16DIRECT(base, (offset) * 2, data)


  int i;
  int j;
  int q ;
  int m,pa,v;
  int aaa = 0; // control siganl of aaaa voice
  int aaa_flag=0;
  //int aa[7];
  int aa_mark1=0;
  int aa_mark2=0;
  int yun_sound1=0;
  int yun_sound2=0;

  int i,j,h,k,t,cx;
  int h=290;
  int v=380;
```

```
float sh=1;
float sv=1;
int c0=0; int c1=0;int c2=0;int c3=0;int c4=0;int c5=0;int c6=0;int c7=0;
int a=0;
int hb=0;
int vb=0;
int key;
int hy=300;
int vy=200;
unsigned char code;

float ball_v;
float ball_h;
float up_v,up_h,down_v,down_h,left_v,left_h,right_v,right_h;
int y_mid;
int count=60;
//y_mid=hy+29;

int life=4;
int final_score=0;
int yun_count=0;    //number of yun that have been hit
int hit_count=0;    //number that board has bounced
//int

int lose_flag=0;
int win_flag=0;

int slow_speed=350;
int fast_speed=500;

int easy=1;
int hard=0;

int lifeball_v;
int lifeball_h;
int lv;
int lh;

float llh=640;//lose life
float llv=-480;
float ll_h=0.01;
float ll_v=0.01;
int ll_flag=0;
```

```
    float speedh=320;
    float speedv=240;
    float speed_h;
    float speed_v;
    int speed_flag=0;
    int speed_count=0;

    int bak[16][12] ={

        {0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,1,1,1,1,1,1,0,0,0},
        {0,0,0,1,1,1,1,1,1,0,0,0},
        {0,0,0,1,1,1,1,1,1,0,0,0},
        {0,0,0,1,1,1,1,1,1,0,0,0},
        {0,0,0,1,1,1,1,1,1,0,0,0},
        {0,0,0,1,1,1,1,1,1,0,0,0},
        {0,0,0,1,1,1,1,1,1,0,0,0},
        {0,0,0,1,1,1,1,1,1,0,0,0},
        {0,0,0,1,1,1,1,1,1,0,0,0},
        {0,0,0,1,1,1,1,1,1,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0}

    };

    #define IOWR_SRAM(base, offset, data) \
    IOWR_16DIRECT(base, (offset) * 2, data)
#define IOWR_DE2_WM8731_AUDIO_INST_DATA(base, offset, data) \
    IOWR_16DIRECT(base, (offset) * 2, data)

static void irqhandler_rot (void * context, alt_u32 id)
{

        if (IORD_16DIRECT (IRCOMP_INST_BASE,0) == 2 && hy<555) {hy=hy+25;}
        else if ( IORD_16DIRECT (IRCOMP_INST_BASE,0)    == 3 && hy>20) {hy=hy-25;}
        IOWR_16DIRECT(IRCOMP_INST_BASE, 0, 0);    // reset the interrupt request
        for (i=0; i<3000; i++){}    // introduce delay to avoid multi-interruput per rotation
}

static void irqhandler_rot2 (void * context, alt_u32 id)
{
        if (IORD_16DIRECT (IRCOMP2_INST_BASE,0) == 2 && vy<395) {vy=vy+25;}
        else if ( IORD_16DIRECT (IRCOMP2_INST_BASE,0)    == 3 && vy>25) {vy=vy-25;}
        IOWR_16DIRECT(IRCOMP2_INST_BASE, 0, 0);    // reset the interrupt request
        for (i=0; i<3000; i++){}    // introduce delay to avoid multi-interruput per rotation
}

int qa,ma,ra,sa;


static void irqhandler_audio (void * context, alt_u32 id)
{

      if(aaa==0 && yun_sound1==0 && yun_sound2==0)
(qa !=1000)
      {
              for (i=0;i<32;i++)
            {
                ma = music[i+qa*32];
                IOWR_DE2_WM8731_AUDIO_INST_DATA(DE2_WM8731_AUDIO_INST_BASE, i, ma) ;

            }

            if (qa< 2626)
                qa=qa+1;
            else
                qa=0;

    }

    else if(aaa)
    {               //aaaa
        for (i=0;i<32;i++)
        {
                ma = music[84084+i+pa*32];
                IOWR_DE2_WM8731_AUDIO_INST_DATA(DE2_WM8731_AUDIO_INST_BASE, i, ma) ;
        }
                if (aaa>1)
                {
                        aaa--;
                        pa=0;
                        //i=0;
                }
                else if (yun_sound1==1 || yun_sound2==1)
            {
                //i=32;
                aaa--;
            }


        if(pa<252)    pa = pa+1;    //max = 169
        else {
                pa = 0;
                aaa = 0;
              }
        }

    else if (aaa==0 && yun_sound1==1) //aaaa
    {
        for (i=0;i<32;i++)
        {
            ma = music[84084+8095+i+ra*32];
            IOWR_DE2_WM8731_AUDIO_INST_DATA(DE2_WM8731_AUDIO_INST_BASE, i, ma) ;

        }

        if(ra<98)    ra = ra+1;    //max = 169
        else {
                ra = 0;
                yun_sound1 = 0;
                //qa = 1001;
              }
    }

    else if (aaa==0 && yun_sound2==1) //aaaa
    {
        for (i=0;i<32;i++)
        {
            ma = music[84084+8095+3143+i+sa*32];
            IOWR_DE2_WM8731_AUDIO_INST_DATA(DE2_WM8731_AUDIO_INST_BASE, i, ma) ;

        }

        if(sa<119)    sa = sa+1;    //max = 169
        else {
                sa = 0;
                yun_sound2 = 0;
                //qa = 1001;
                }
    }
}





static void irqhandler (void * context, alt_u32 id)
{



//    printf ("interrupt occurred\n");
    IOWR_VGA_DATA(VGA_BASE, 5, h);
    IOWR_VGA_DATA(VGA_BASE, 6, v);
    IOWR_VGA_DATA(VGA_BASE, 7, hy);
  // IOWR_16DIRECT(VGA_BASE, 0, 0);    // reset the interrupt request
}


int main()
{


    alt_irq_register( IRCOMP_INST_IRQ, NULL,(void*)irqhandler_rot );
    alt_irq_register( IRCOMP2_INST_IRQ, NULL,(void*)irqhandler_rot2 );

    alt_irq_register (VGA_IRQ, NULL, irqhandler );

      alt_irq_register (DE2_WM8731_AUDIO_INST_IRQ, NULL, irqhandler_audio );


    for (;;)
    {
        IOWR_VGA_DATA(VGA_BASE, 9, vy);
        IOWR_VGA_DATA(VGA_BASE, 2, 89);
        IOWR_VGA_DATA(VGA_BASE, 3, 78);
        IOWR_VGA_DATA(VGA_BASE, 16, life);

    up_v=v;
    up_h=h+7.5;

    down_v=v+15;
    down_h=h+7.5;

    left_v=v+7.5;
    left_h=h;

    right_v=v+7.5;
    right_h=h+15;

    ball_v=v+7.5;
    ball_h=h+7.5;
    hb=ball_h/40;
    vb=ball_v/40;




    if (sh<0 && sv<0 && ball_h>hb*40+30-3 && ball_h<hb*40+40
&& ball_v>=vb*40+35-3 && ball_v<vb*40+40+3 && bak[hb][vb]==1)
    {
            a=0x1000;
            c4=a+vb+16*hb;
            sh=1;
            sv=1;
            bak[hb][vb]=0;
               count--;
               yun_count++;
               aaa++;

    else if (sh<0 && sv>0 && ball_h>=hb*40+30-3 && ball_h<hb*40+40
&& ball_v>=vb*40+30-3 && ball_v<vb*40+40+3 && bak[hb][vb+1]==1)
    {
            a=0x1000;
            c5=a+vb+1+16*hb;
            sh=1;
            sv=-1;
            bak[hb][vb+1]=0;
               count--;
               yun_count++;
               aaa++;


    }
    //left & bottom corner
    else if (sh>0 && sv<0 && ball_h>=hb*40+30-3 && ball_h<hb*40+35+3
&& ball_v>=vb*40+35-3 && ball_v<vb*40+37.5+3 && bak[hb+1][vb]==1)
```

```
    {
            a=0x1000;
            c6=a+vb+16*(hb+1);
            sh=-1;
            sv=1;
            bak[hb+1][vb]=0;
                count--;
                yun_count++;
                aaa++;

    }

    else if (sh>0 && sv>0 && ball_h>=hb*40+30-3 && ball_h<hb*40+35+3
&& ball_v>=vb*40+30-3 && ball_v<vb*40+35+3 && bak[hb+1][vb+1]==1)
    {
            a=0x1000;
            c7=a+vb+1+16*(hb+1);
            sh=-1;
            sv=-1;
            bak[hb+1][vb+1]=0;
                count--;
                yun_count++;
                aaa++;

    }

    //top edge
  else if (((sh>=0 && sv>0)||(sh<=0 && sv>0)) && ball_v==(vb*40+32.5)
        //down_v>=((vb+1)*40) && down_v<=((vb+1)*40+5)
            //&& down_h>(hb*40-5) && down_h<(hb*40+35) && bak[hb][vb+1]==1)
&& ball_h>=(hb*40-5) && ball_h<=(hb*40+30+5) && bak[hb][vb+1]==1)
//    if (ball_h==hb*40 && ball_v>vb*40 && ball_v<(vb+1)*40 && bak[hb][vb]==1)
    {

                a=0x1000;
                c0=a+vb+1+16*hb;
                sv=-sv;
                bak[hb][vb+1]=0;
                count--;
                yun_count++;
                aaa++;

    }
    //bottom edge
    else if (((sh>=0 && sv<0)||(sh<=0 && sv<0)) && ball_v==vb*40+37.5
                        //up_v>=((vb*40)-20) && up_v<=((vb*40)-7)
                        //&& up_h>(hb*40-5) && up_h<(hb*40+35) && bak[hb][vb-1]==1
&& ball_h>=(hb*40-5) && ball_h<=(hb*40+30+5) && bak[hb][vb]==1)
    {
                        a=0x1000;
                    c1=a+vb+16*hb;
                    sv=-sv;
                    bak[hb][vb]=0;
                    count--;
                    yun_count++;
                    aaa++;

    }
    //left edge
    else if (((sh>0 && sv>=0)||(sh>0 && sv<=0)) && ball_h==(hb*40+32.5)
                        //right_h<=(hb*40+45) && right_h>=(hb*40+40)
                        //&& right_v>(vb*40-5) && right_v<(vb*40+30+5)
&& ball_v>=(vb*40-3) && ball_v<=(vb*40+30+3)
&& bak[hb+1][vb]==1)
    {
                        a=0x1000;
                    c2=a+vb+16*(hb+1);
                    sh=-sh;
                    bak[hb+1][vb]=0;
                    count--;
                    yun_count++;
                    aaa++;
                    //aa[6]=1;
                    //printf("count is %d\n",count);
                   // printf("yun_count is %d\n",yun_count);
//                  for (i=0; i<300; i++)
//                  {
//                      IOWR_VGA_DATA(VGA_BASE, 8, c2);
//                  }
    }
    // right edge
    else if (((sh<0 && sv>0)||(sh<0 && sv<0)) && ball_h==(hb*40+37.5)
                //left_h<=(hb*40) && left_h>=((hb*40)-15)
                        //&& left_v>(vb*40-5) && left_v<(vb*40+35)
&& ball_v>=(vb*40-5) && ball_v<=(vb*40+30+5) && bak[hb][vb]==1 )
    {
                        a=0x1000;
                    c3=a+vb+16*hb;
                    sh=-sh;
                    bak[hb][vb]=0;
                    count--;
                    yun_count++;
                    aaa++;
                    //aa[7]=1;
                    //printf("count is %d\n",count);
                  // printf("yun_count is %d\n",yun_count);
//                  for (i=0; i<300; i++)
//                  {
//                        IOWR_VGA_DATA(VGA_BASE, 8, c3);
//                  }
    }
      if (easy==1)
     {
                    for (i=0; i<slow_speed; i++)
           {
                IOWR_VGA_DATA(VGA_BASE, 8, c0);
                IOWR_VGA_DATA(VGA_BASE, 8, c1);
                IOWR_VGA_DATA(VGA_BASE, 8, c2);
                IOWR_VGA_DATA(VGA_BASE, 8, c3);
                IOWR_VGA_DATA(VGA_BASE, 8, c4);
                IOWR_VGA_DATA(VGA_BASE, 8, c5);
                IOWR_VGA_DATA(VGA_BASE, 8, c6);
                IOWR_VGA_DATA(VGA_BASE, 8, c7);
           }
     }
     else if (hard==1)
     {
                    for (i=0; i<fast_speed; i++)
           {
                IOWR_VGA_DATA(VGA_BASE, 8, c0);
                IOWR_VGA_DATA(VGA_BASE, 8, c1);
                IOWR_VGA_DATA(VGA_BASE, 8, c2);
                IOWR_VGA_DATA(VGA_BASE, 8, c3);
                IOWR_VGA_DATA(VGA_BASE, 8, c4);
                IOWR_VGA_DATA(VGA_BASE, 8, c5);
                IOWR_VGA_DATA(VGA_BASE, 8, c6);
                IOWR_VGA_DATA(VGA_BASE, 8, c7);
           }
     }

//   for (i=0; i<3000; i++)
//   {
```

```
//    }

//    if (ball_v>49 && ball_v<79 && ball_h>49 && ball_h<79 && !(sh>0 && sv>0))
//    {
//        if (hit_count%2==0)
//        {
//                h=550;
//                v=49;
//                sh=-1;
//                sv=1;
//        }
//        else
//        {
//                h=420;
//                v=320;
//                sh=-1;
//                sv=1;
//        }
//    }
//    else if (ball_v>49 && ball_v<79 && ball_h>550 && ball_h<580 && !(sh<0 &&sv>0))
//    {
//        if (hit_count%2==0)
//        {
//                h=49;
//                v=49;
//                sh=1;
//                sv=1;
//        }
//        else
//        {
//                h=420;
//                v=320;
//                sh=-1;
//                sv=1;
//        }
//    }
//    else if (ball_v>320 && ball_v<350 && ball_h>420 && ball_h<450 && !(sh<0&&sv>0))
//    {
//        if (hit_count%2==0)
//        {
//                h=49;
//                v=49;
//                sh=1;
//                sv=1;
//        }
//        else
//        {
//               h=550;
//               v=49;
//               sh=-1;
//               sv=1;
//        }
//    }


    if (hit_count%15==0 && ll_flag==0)
    {
          llh=320;
          llv=240;
          ll_h=sh;
          ll_v=sv;
          ll_flag=1;
        //IOWR_VGA_DATA(VGA_BASE, 2, ll_h);
        //IOWR_VGA_DATA(VGA_BASE, 3, ll_v);
    }

    if (ll_flag==1)
    {
            //if ((llv+22)>=440 && (llv+22)<460 && (llh+7.5)>=(hy-2) && (llh+7.5)<=(hy+60))
            if ((llv+22)==442 && (llh+7.5)>=(hy-2) && (llh+7.5)<=(hy+60))
            {
                llh=640;
                llv=480;
                life--;
                ll_flag=0;
            }
            //else if (llv>=18 && llv<=38 && (llh+7.5)>=(hy-2) && (llh+7.5)<=(hy+60))
            else if (llv==38 && (llh+7.5)>=(hy-2) && (llh+7.5)<=(hy+60))
            {
                llh=640;
                llv=480;
                life--;
                ll_flag=0;
            }
            //else if (llh>=18 && llh<=38 && (llv+11)>=(vy-2) && (llv+11)<=(vy+60))
            else if (llh==38 && (llv+11)>=(vy-2) && (llv+11)<=(vy+60))
            {
                llh=640;
                llv=480;
                life--;
                ll_flag=0;
            }
            //else if ((llh+15)>=602 && (llh+15)<=622 && (llv+11)>=(vy-2) && (llv+11)<=(vy+60))
            else if ((llh+15)==602 && (llv+11)>=(vy-2) && (llv+11)<=(vy+60))
            {
                llh=640;
                llv=480;
                life--;
                ll_flag=0;
            }
            else if (llh>480 || llh<0 || llv>640 || llv<0)
            {
                llh=640;
                llv=480;
                ll_flag=0;
            }
            else
            {
                llh+=ll_h;
                llv+=ll_v;
            }
            //IOWR_VGA_DATA(VGA_BASE, 2, ll_h);
            //IOWR_VGA_DATA(VGA_BASE, 3, ll_v);
            //printf("life is %d\n",life);
    }

          if (h<=0)
          {
                for (i=0;i<2000;i++)
                {
                        h=90;
                        v=200;
                }
                h=90;
                v=200;
                sh=-1;
                sv=1;
                life--;
                //sh=1;
          }
          if (h>=624)
          {
```

```
    for (i=0;i<2000;i++)
    {
        h=550;
        v=200;
    }
    h=550;
    v=200;
    sh=1;
    sv=1;
    life--;
    //sh=-1;
}

if (v>=464)
{
    for (i=0;i<2000;i++)
    {
        h=290;
        v=380;
    }
    h=290;
    v=380;
    sh=1;
    sv=1;
    life--;
    //    sv=-1;
}

if (v<=0)
{
    for (i=0;i<2000;i++)
    {
        h=290;
        v=80;
    }
    h=290;
    v=80;
    sh=-1;
    sv=-1;
    life--;
      // sv=1;
}
```

```
        //bottom board
         if (down_v>=440 && down_v <460 && down_h>=(hy-2) && down_h<=(hy+60))
         {
//              for (h=0;h<50;h++)
//              {
//                      IOWR_VGA_DATA(VGA_BASE, 1, 1);
//              }
                hit_count++;
                //printf("hit_count is %d\n",hit_count);
                if (sh>0 && sv>0)
                {

                        if (hy+29>down_h)
                        {
                                sh=-sh;
                                sv=-sv;
                                int tmp=hy+29-down_h;
                                if (tmp>=20 && sh<=3 && sv<=3)
                                {
                                        sh*=1.6;
                                        sv*=1.6;
                                }
                                else if (tmp>=9 && tmp<20 && sh<=3 && sv<=3)
                                {
                                        sh*=1.3;
                                        sv*=1.3;
                                }

                        }
                        else
                        {
                                sv=-sv;
                                int tmp=down_h-hy-29;
                                if (tmp>=20 && sh<=3 && sv<=3)
                                {
                                        sh*=1.6;
                                        sv*=1.6;
                                }
                                else if (tmp>=9 && tmp<20 && sh<=3 && sv<=3)
                                {
                                        sh*=1.3;
                                        sv*=1.3;
                                }
                        }
        }
        else if (sh<0 && sv>0)
        {
                if (hy+29>=down_h)
                {
                        sv=-sv;
                        int tmp=hy+29-down_h;
                        if (tmp>=20 && sh<=3 && sv<=3)
                        {
                                sh*=1.6;
                                sv*=1.6;
                        }
                        else if (tmp>=9 && tmp<20 && sh<=3 && sv<=3)
                        {
                                sh*=1.3;
                                sv*=1.3;
                        }

                }
                else
                {
                        sv=-sv;
                        sh=-sh;
                        int tmp=down_h-hy-29;
                        if (tmp>=20 && sh<=3 && sv<=3)
                        {
                                sh*=1.6;
                                sv*=1.6;
                        }
                        else if (tmp>=9 && tmp<20 && sh<=3 && sv<=3)
                        {
                                sh*=1.3;
                                sv*=1.3;
                        }
                }
        }
        yun_sound1=1;
        //for (k=0;k<5000;k++)
        //IOWR_VGA_DATA(VGA_BASE, 12, 1);
  }


//top board
if (up_v>=18 && up_v <=38 && up_h>=(hy-2) && up_h<=(hy+60))
```

```
        {
//              for (h=0;h<50;h++)
//              {
//                      IOWR_VGA_DATA(VGA_BASE, 1, 2);
//              }
            hit_count++;
            //printf("hit_count is %d\n",hit_count);
            if (sh>0 && sv<0)
            {
                if (hy+29>up_h)
                {
                    sh=-sh;
                    sv=-sv;
                    int tmp=hy+29-up_h;
                    if (tmp>=20 && sh<=3 && sv<=3)
                    {
                        sh*=1.6;
                        sv*=1.6;
                    }
                    else if (tmp>=9 && tmp<20 && sh<=3 && sv<=3)
                    {
                        sh*=1.3;
                        sv*=1.3;
                    }

                }
                else
                {
                    sv=-sv;
                    int tmp=up_h-hy-29;
                    if (tmp>=20 && sh<=3 && sv<=3)
                    {
                        sh*=1.6;
                        sv*=1.6;
                    }
                    else if (tmp>=9 && tmp<20 && sh<=3 && sv<=3)
                    {
                        sh*=1.3;
                        sv*=1.3;
                    }
                }
            }
            else if (sh<0 && sv<0)
            {
```

```
                        if (hy+29>=up_h)
                        {
                                sv=-sv;
                                int tmp=hy+29-up_h;
                                if (tmp>=20 && sh<=3 && sv<=3)
                                {
                                        sh*=1.6;
                                        sv*=1.6;
                                }
                                else if (tmp>=9 && tmp<20 && sh<=3 && sv<=3)
                                {
                                        sh*=1.3;
                                        sv*=1.3;
                                }

                        }
                        else
                        {
                                sv=-sv;
                                sh=-sh;
                                int tmp=up_h-hy-29;
                                if (tmp>=20 && sh<=3 && sv<=3)
                                {
                                        sh*=1.6;
                                        sv*=1.6;
                                }
                                else if (tmp>=9 && tmp<20 && sh<=3 && sv<=3)
                                {
                                        sh*=1.3;
                                        sv*=1.3;
                                }
                        }
                }
                yun_sound1=1;
          }

        //left board
        if (left_h>=18 && left_h<=38 && left_v>=(vy-2) && left_v<=(vy+60))
          {
//                for (h=0;h<50;h++)
//                {
//                        IOWR_VGA_DATA(VGA_BASE, 1, 8);
//                }
                hit_count++;
```

```
//printf("hit_count is %d\n",hit_count);
if (sh<0 && sv<0)
{
        if (vy+29<left_v)
        {
                sh=-sh;
                sv=-sv;
                int tmp=left_v-vy-29;
                if (tmp>=20 && sh<=3 && sv<=3)
                {
                        sh*=1.4;
                        sv*=1.4;
                }
                else if (tmp>=9 && tmp<20 && sh<=3 && sv<=3)
                {
                        sh*=1.2;
                        sv*=1.2;
                }

        }
        else
        {
                sh=-sh;
                int tmp=left_v-vy-29;
                if (tmp>=20 && sh<=3 && sv<=3)
                {
                        sh*=1.4;
                        sv*=1.4;
                }
                else if (tmp>=9 && tmp<20 && sh<=3 && sv<=3)
                {
                        sh*=1.2;
                        sv*=1.2;
                }
        }
}
else if (sh<0 && sv>0)
{
        if (vy+29<=left_v)
        {
                sh=-sh;
                int tmp=left_v-vy-29;
                if (tmp>=20 && sh<=3 && sv<=3)
                {
                        sh*=1.4;
                        sv*=1.4;
                    }
                    else if (tmp>=9 && tmp<20 && sh<=3 && sv<=3)
                    {
                        sh*=1.2;
                        sv*=1.2;
                    }

                }
                else
                {
                    sv=-sv;
                    sh=-sh;
                    int tmp=left_v-vy-29;
                    if (tmp>=20 && sh<=3 && sv<=3)
                    {
                        sh*=1.4;
                        sv*=1.4;
                    }
                    else if (tmp>=9 && tmp<20 && sh<=3 && sv<=3)
                    {
                        //sh*=1    IOWR_VGA_DATA(VGA_BASE, 2, llh);
   //IOWR_VGA_DATA(VGA_BASE, 3, llv);.2;
                        sv*=1.2;
                    }
                }
            }
            yun_sound2=1;
        }



        //right board
        if (right_h>=602 && right_h<=622 && right_v>=(vy-2) && right_v<=(vy+60))
        {
//            for (h=0;h<50;h++)
//            {
//                IOWR_VGA_DATA(VGA_BASE, 1, 4);
//            }
            hit_count++;
            //printf("hit_count is %d\n",hit_count);
            if (sh>0 && sv<0)
            {
                if (vy+29<right_v)
        {
                sh=-sh;
                sv=-sv;
                int tmp=right_v-vy-29;
                if (tmp>=20 && sh<=3 && sv<=3)
                {
                        sh*=1.4;
                        sv*=1.4;
                }
                else if (tmp>=9 && tmp<20 && sh<=3 && sv<=3)
                {
                        sh*=1.2;
                        sv*=1.2;
                }

        }
        else
        {
                sh=-sh;
                int tmp=right_v-vy-29;
                if (tmp>=20 && sh<=3 && sv<=3)
                {
                        sh*=1.4;
                        sv*=1.4;
                }
                else if (tmp>=9 && tmp<20 && sh<=3 && sv<=3)
                {
                        sh*=1.2;
                        sv*=1.2;
                }
        }
}
else if (sh>0 && sv>0)
{
        if (vy+29<=right_v)
        {
                sh=-sh;
                int tmp=right_v-vy-29;
                if (tmp>=20 && sh<=3 && sv<=3)
                {
                        sh*=1.4;
                        sv*=1.4;
                }
                else if (tmp>=9 && tmp<20 && sh<=3 && sv<=3)
```

```
                    {
                        sh*=1.2;
                        sv*=1.2;
                    }

                }
                else
                {
                    sv=-sv;
                    sh=-sh;
                    int tmp=right_v-vy-29;
                    if (tmp>=20 && sh<=3 && sv<=3)
                    {
                        sh*=1.4;
                        sv*=1.4;
                    }
                    else if (tmp>=9 && tmp<20 && sh<=3 && sv<=3)
                    {
                        sh*=1.2;
                        sv*=1.2;
                    }
                }
            }
            yun_sound2=1;
        }

        //if (hy==0)
        h=h+sh;
        v=v+sv;









      if (count==0)
      {
            win_flag=1;
            //final_score=yun_count*100;
            final_score=6000+(60-hit_count)*50+life*500;
            //printf("life is %d\n",life);
              //a=0x1000;
          int c;
```

```
while(count==0)
    {

            a=0x1000;
            t=0x1000;
            for (h=0; h<win_delay1; h++)
            {
                    c=a+7+16*12;
                    IOWR_VGA_DATA(VGA_BASE, 8, c);
            }
            for (h=0; h<win_delay1; h++)
            {
                    c=a+8+16*11;
                    IOWR_VGA_DATA(VGA_BASE, 8, c);
            }
            a=0x1100;
            for (h=0; h<win_delay1; h++)
            {
                    c=a+3+16*3;
                    IOWR_VGA_DATA(VGA_BASE, 8, c);
                    //cx=t+8+16*12;
                    //IOWR_VGA_DATA(VGA_BASE, 8, cx);
            }
            for (h=0; h<win_delay1; h++)
            {
                    c=a+4+16*3;
                    IOWR_VGA_DATA(VGA_BASE, 8, c);
                    //cx=t+8+16*12;
                    //IOWR_VGA_DATA(VGA_BASE, 8, cx);
            }
            for (h=0; h<win_delay1; h++)
            {
                    c=a+5+16*3;
                    IOWR_VGA_DATA(VGA_BASE, 8, c);
                    //cx=t+8+16*12;
                    //IOWR_VGA_DATA(VGA_BASE, 8, cx);
            }
            for (h=0; h<win_delay1; h++)
            {
                    c=a+6+16*3;
                    IOWR_VGA_DATA(VGA_BASE, 8, c);
                    //cx=t+8+16*12;
                    //IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
        c=a+7+16*4;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
        //cx=t+8+16*12;
        //IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
        c=a+6+16*5;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
        //cx=t+8+16*12;
        //IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
        c=a+5+16*5;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
      // cx=t+8+16*12;
        //IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
        c=a+4+16*5;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
        //cx=t+8+16*12;
        //IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
        c=a+3+16*5;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
        //cx=t+8+16*12;
        //IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
        c=a+7+16*6;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
        //cx=t+8+16*12;
        //IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
```

```
{
        c=a+6+16*7;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
      // cx=t+8+16*12;
        //IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
        c=a+5+16*7;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
        //cx=t+8+16*12;
        //IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
        c=a+4+16*7;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
        //cx=t+8+16*12;
        //IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
        c=a+3+16*7;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
        //cx=t+8+16*12;
        //IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
        c=a+3+16*8;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
        //cx=t+8+16*12;
        //IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
        c=a+4+16*8;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
      // cx=t+8+16*12;
        //IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
        c=a+5+16*8;
```

```
          IOWR_VGA_DATA(VGA_BASE, 8, c);
          //cx=t+8+16*12;
          //IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
          c=a+6+16*8;
          IOWR_VGA_DATA(VGA_BASE, 8, c);
          //cx=t+8+16*12;
          //IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
          c=a+7+16*8;
          IOWR_VGA_DATA(VGA_BASE, 8, c);
        // cx=t+8+16*12;
        // IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
          c=a+7+16*9;
          IOWR_VGA_DATA(VGA_BASE, 8, c);
          //cx=t+8+16*12;
          //IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
          c=a+6+16*9;
          IOWR_VGA_DATA(VGA_BASE, 8, c);
        // cx=t+8+16*12;
        // IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
          c=a+5+16*9;
          IOWR_VGA_DATA(VGA_BASE, 8, c);
        // cx=t+8+16*12;
        // IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
          c=a+4+16*9;
          IOWR_VGA_DATA(VGA_BASE, 8, c);
        // cx=t+8+16*12;
        //IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
        c=a+3+16*9;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
    //    cx=t+8+16*12;
        //IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
        c=a+5+16*10;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
      // cx=t+8+16*12;
      // IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
        c=a+6+16*11;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
    //    cx=t+8+16*12;
      // IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
        c=a+7+16*12;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
    //    cx=t+8+16*12;
    //    IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
        c=a+6+16*12;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
    //    cx=t+8+16*12;
    //    IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
        c=a+5+16*12;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
   //     cx=t+8+16*12;
    //    IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
```

```
for (h=0; h<win_delay1; h++)
{
          c=a+4+16*12;
          IOWR_VGA_DATA(VGA_BASE, 8, c);
      //    cx=t+8+16*12;
      //    IOWR_VGA_DATA(VGA_BASE, 8, cx);
}
for (h=0; h<win_delay1; h++)
{
          c=a+3+16*12;
          IOWR_VGA_DATA(VGA_BASE, 8, c);
      //    cx=t+8+16*12;
  //        IOWR_VGA_DATA(VGA_BASE, 8, cx);
}

for (h=0; h<win_delay2; h++)
{
}

for (h=0; h<win_delay1; h++)
{
        a=0x1000;
        c=a+7+16*3;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
        c=a+8+16*3;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
        c=a+3+16*4;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
        c=a+4+16*4;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
        c=a+5+16*4;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
        c=a+6+16*4;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
        c=a+8+16*4;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
        c=a+7+16*5;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
        c=a+8+16*5;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
        c=a+3+16*6;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
        c=a+4+16*6;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
```

```
c=a+5+16*6;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+6+16*6;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+8+16*6;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+7+16*7;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+8+16*7;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+8+16*8;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+8+16*9;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+8+16*12;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+3+16*10;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+4+16*10;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+6+16*10;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+7+16*10;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+8+16*10;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+3+16*11;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+4+16*11;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+5+16*11;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+7+16*11;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+7+16*12;
IOWR_VGA_DATA(VGA_BASE, 8, c);

 //a=0x1000;
 c=a+3+16*3;
 IOWR_VGA_DATA(VGA_BASE, 8, c);
 c=a+4+16*3;
 IOWR_VGA_DATA(VGA_BASE, 8, c);
 c=a+5+16*3;
 IOWR_VGA_DATA(VGA_BASE, 8, c);
```

```
c=a+6+16*3;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+7+16*4;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+6+16*5;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+5+16*5;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+4+16*5;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+3+16*5;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+7+16*6;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+6+16*7;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+5+16*7;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+4+16*7;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+3+16*7;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+6+16*8;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+5+16*8;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+4+16*8;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+3+16*8;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+7+16*8;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+6+16*9;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+5+16*9;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+4+16*9;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+3+16*9;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+7+16*9;
IOWR_VGA_DATA(VGA_BASE, 8, c);
c=a+5+16*10;
IOWR_VGA_DATA(VGA_BASE, 8, c);
```

```
                        c=a+6+16*11;
                        IOWR_VGA_DATA(VGA_BASE, 8, c);
                        c=a+6+16*12;
                        IOWR_VGA_DATA(VGA_BASE, 8, c);
                        c=a+5+16*12;
                        IOWR_VGA_DATA(VGA_BASE, 8, c);
                        c=a+4+16*12;
                        IOWR_VGA_DATA(VGA_BASE, 8, c);
                        c=a+3+16*12;
                        IOWR_VGA_DATA(VGA_BASE, 8, c);
                        c=a+8+16*11;
                        IOWR_VGA_DATA(VGA_BASE, 8, c);
                    }


            }
    }
    if (life==0)
    {
          lose_flag=1;
          //final_score=yun_count*100;
          //final_score=6000+(60-hit_count)*50+life*500;
        // printf("final_score is %d\n",final_score);

        int c;



        while(life==0)
        {


                //a=0x1000;
                for (i=0;i<16;i++)
                {
                    for (j=0;j<12;j++)
                    {
                       for (h=0; h<lose_delay1; h++)
                       {
                         c=4096+j+16*i;
                         IOWR_VGA_DATA(VGA_BASE, 8, c);
                       }
                    }
                }

               a=0x1100;
```

```
for (h=0; h<lose_delay2; h++)
{
        c=a+3+16*2;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+4+16*2;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+5+16*2;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+6+16*2;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+7+16*2;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+7+16*3;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+7+16*4;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+3+16*5;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+4+16*5;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
```

```
}
for (h=0; h<lose_delay2; h++)
{
        c=a+5+16*5;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+6+16*5;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+7+16*5;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+7+16*6;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+7+16*7;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+6+16*7;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+5+16*7;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+4+16*7;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+3+16*7;
```

```
		IOWR_VGA_DATA(VGA_BASE, 8, c);
	}
	for (h=0; h<lose_delay2; h++)
	{
		c=a+3+16*6;
		IOWR_VGA_DATA(VGA_BASE, 8, c);
	}
	for (h=0; h<lose_delay2; h++)
	{
		c=a+3+16*10;
		IOWR_VGA_DATA(VGA_BASE, 8, c);
	}
	for (h=0; h<lose_delay2; h++)
	{
		c=a+3+16*9;
		IOWR_VGA_DATA(VGA_BASE, 8, c);
	}
	for (h=0; h<lose_delay2; h++)
	{
		c=a+3+16*8;
		IOWR_VGA_DATA(VGA_BASE, 8, c);
	}
	for (h=0; h<lose_delay2; h++)
	{
		c=a+4+16*8;
		IOWR_VGA_DATA(VGA_BASE, 8, c);
	}
	for (h=0; h<lose_delay2; h++)
	{
		c=a+5+16*8;
		IOWR_VGA_DATA(VGA_BASE, 8, c);
	}
	for (h=0; h<lose_delay2; h++)
	{
		c=a+5+16*9;
		IOWR_VGA_DATA(VGA_BASE, 8, c);
	}
	for (h=0; h<lose_delay2; h++)
	{
		c=a+5+16*10;
		IOWR_VGA_DATA(VGA_BASE, 8, c);
	}
	for (h=0; h<lose_delay2; h++)
	{
        c=a+6+16*10;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+7+16*10;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+7+16*9;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+7+16*8;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+3+16*13;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+3+16*12;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+3+16*11;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+4+16*11;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
{
        c=a+5+16*11;
        IOWR_VGA_DATA(VGA_BASE, 8, c);
}
for (h=0; h<lose_delay2; h++)
```

```
                        {
                                c=a+5+16*12;
                                IOWR_VGA_DATA(VGA_BASE, 8, c);
                        }
                        for (h=0; h<lose_delay2; h++)
                        {
                                c=a+5+16*13;
                                IOWR_VGA_DATA(VGA_BASE, 8, c);
}
                        for (h=0; h<lose_delay2; h++)
{
                                c=a+6+16*11;
IOWR_VGA_DATA(VGA_BASE, 8, c);
                        }
                        for (h=0; h<lose_delay2; h++)
                        {
                                c=a+7+16*11;
                                IOWR_VGA_DATA(VGA_BASE, 8, c);
                        }
                        for (h=0; h<lose_delay2; h++)
                        {
                                c=a+7+16*12;
                                IOWR_VGA_DATA(VGA_BASE, 8, c);
                        }
                        for (h=0; h<lose_delay2; h++)
                        {
                                c=a+7+16*13;
                                IOWR_VGA_DATA(VGA_BASE, 8, c);
                        }

                        for (h=0; h<lose_delay3; h++)
{
                        }


                }
     }



   }
   return 0;
}
```