Snappers

CSEE4840 Project

Yuhan Dai
Electrical Engineering Department
yd2233@columbia.edu

Lianyi Ding
Electrical Engineering Department
ld2504@columbia.edu

Dian Wang
Electrical Engineering Department
dw2504@columbia.edu

Chi Zhang
Electrical Engineering Department
cz2244@columbia.edu

Contents

1. Introduction
2. Design Architecture
3. Hardware
   3.1 Keyboard
   3.2 VGA
   3.3 Audio
4. Software
5. Work Division
6. Codes
   6.1 de2_vga_raster.vhd
   6.2 frog_controller.vhd
   6.3 ps_controller.vhd
   6.4 lab3_audio.vhd(top)
   6.5 de2_wm8731_audio.vhd
   6.6 hello_world.c

1. Introduction

“Snappers” is a popular puzzle game on iOS platform developed by Emerging Banking LLC in 2011. The goal of this game is to eliminate as many snappers (the red and green objects in the picture) as possible. The player needs to choose a snapper and turn it into four bullets in horizontal and vertical directions to shoot other snappers. Snappers of different colors can resist different times of shooting before it disappears. Once a snapper is shot, its color will change correspondingly. This game requires thinking, determination and trial-and-error. Our goal is to implement this game on the Altera Cyclone II FPGA board. We will use the FPGA board to display the graphics and the C program to control the movement and variation of the objects shown on the screen. We will add some sound effect to the game and we will implement some functional button to enable pause/resume and forward/backward. In addition, the players will learn how to play by following the tutorial instruction.

2. Design Architecture

In our project, there are mainly three parts of work. The block diagram below shows the fundamental architecture of Snappers.

First, we use the keyboard as the game controller. The keyboard controller should receive the input from the keyboard and send data into the Avalon Bus. Secondly, through the VGA control, we can edit the movement or variation of the figures which implemented by VGA raster. It will finally display on the LCD screen. Thirdly, as the procedure of VGA, we use the audio control to manage different sounds and then produce them through the sound box.

We can see from the diagram: Keyboard, LCD displayer and sound box are the peripherals of this design; all the three controllers pass the signal back and forward to CPU—NIOS processor through the Avalon Bus; SRAM can store all the data in the bus.

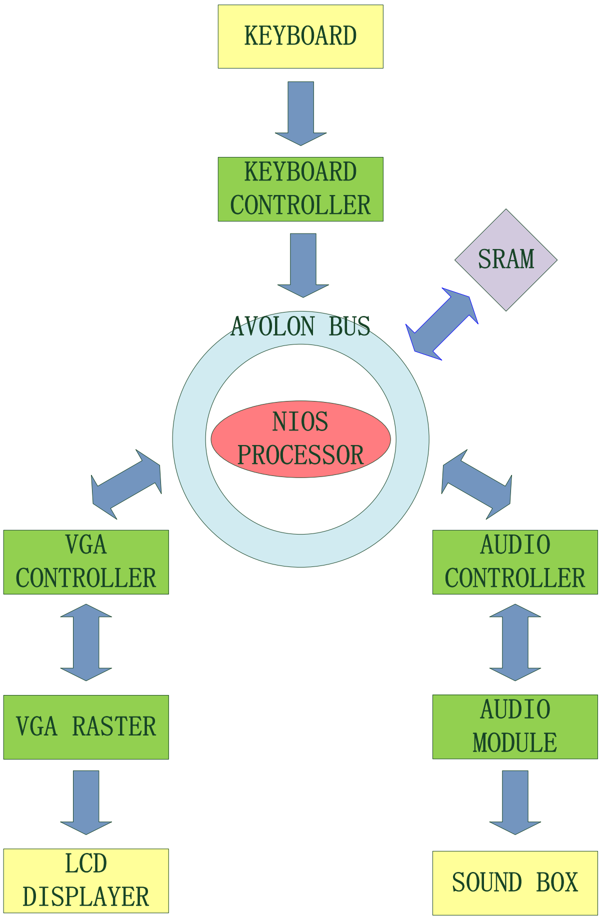


Figure 2.1 Architecture Diagram

3. Hardware

There are three fundamental implementations based on hardware, the keyboard, VGA and audio, which are explained separately below.

3.1 Keyboard Control

PS2 keyboard here are files from Lab 3. It is used to control both the video and audio part. We use VGA and audio function in our processor system to read the data from keyboard and then send the data to Avalon bus. The Nios2 system will connect the keyboard controller to the VGA and audio controller through the Avalon bus. The keyboard interface will be implemented in C program.

In this game, the key function table is shown in Table 3.1.

Table 3.1 Key Function

KEY	↑ ↓ ← →	N,B,P,S	Enter
FUNCTION	Change Snappers	For button: Nest, Back, Pause, Start	Select Snappers

3.2 VGA

3.2.1 Several elements' pictures

Snappers

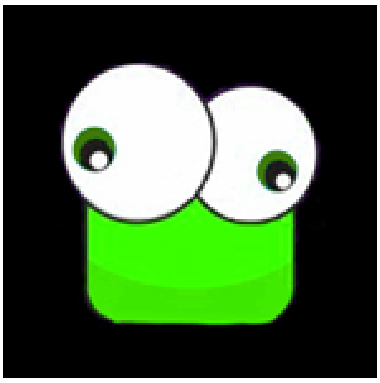


Figure 3.2.1-1 Green Snapper

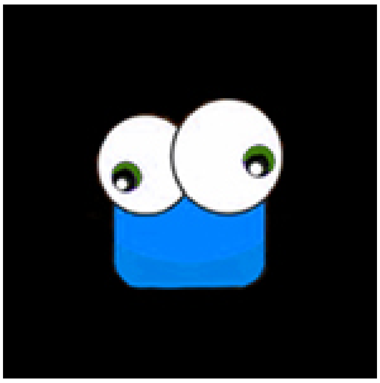


Figure 3.2.1-2 Blue Snapper

Bullets and explosion

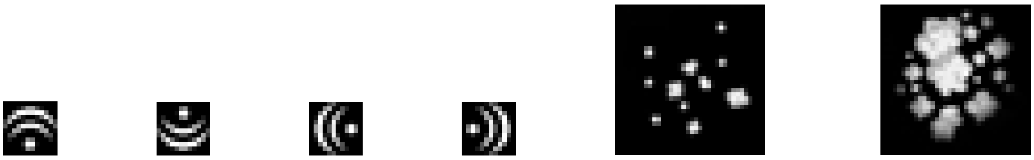


Figure 3.2.1-3 Bullets and explosion

Buttons

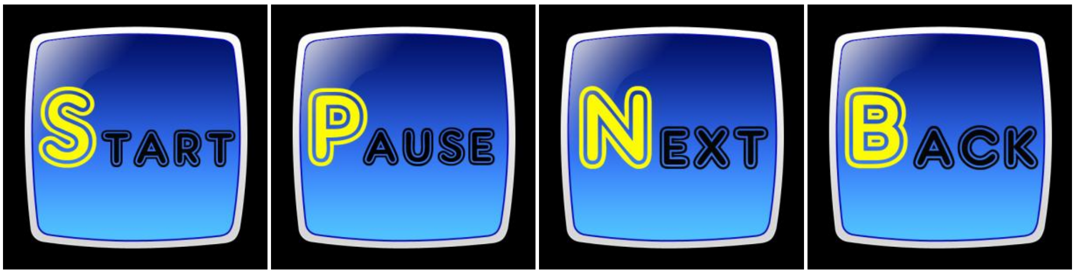


Figure 3.2.1-4 Buttons

Several texts

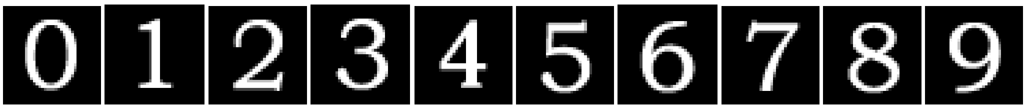


Figure 3.2.1-5 Numbers

Figure 3.2.1-6 Numbers

Figure 3.2.1-7 Numbers

3.2.2 VGA Design

8 different snappers

In our project, we have two types (big and small) snappers with four different colors. Since each snapper will have RGB values, there should be 48 figures for the snappers. In order to save the memory and logic element in FPGA, we use ROM to store the figure data and we set the same size for snappers with different colors so that we can change the color by simply switch the figure of RGB. This will reduce the number of figures as 8.

Snapper matrix

In the game snappers, we have to determine each snapper in each position in the hardware to draw that snapper correctly. In order to control different types of snappers we use a “for loop” so that we can store the information (location and type) into the array.

Snapper mirror

To make the snapper shown as blinking we use a mirror figure for each snapper. And use flag to control it switchover. Here is the original figure and the mirror figure:

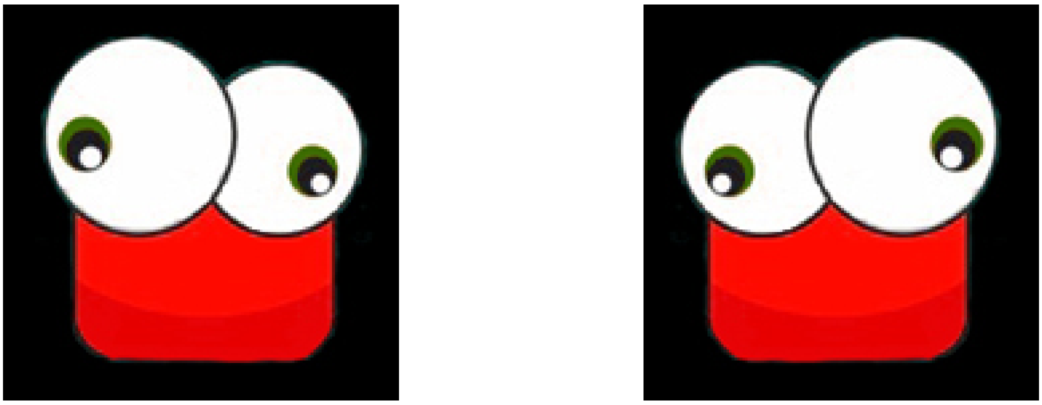


Figure 3.2.2-1 mirror snapper

Connect with software and show the picture

In NIOS2_ide, software will send the signal to hardware when the condition matched. We use CLK50 and CLK25 in video part of this project. We use slower clock to set parameter and flag control. On the other hand, we use faster clock to do the video output. Because we need the hardware finish all the parameter setting before the video output. In our project, we only receive the data of all the fast changed objects at the screen synchronization time, which is vga_vsync = “1‟ and vga_hsync = “1‟. This means we update the data when the screen point reset.

Color assigned

In this project, we use matlab to export the RGB pixels and store them into different ROMs. A vhdl file will be set as controller to connect the ROM into the system. In order to safe memory we ignore the last 5 bits of the RGB and use 1 bit matrix to indicate all the black and white figures.

Number and letter matrix:

For the implementation of number and letter, we use the display mechanic of Lab2. We use the hexadecimal matrix of ps2 for each letter and turn it into binary matrix for each character, and then use this for the R,G,B values.

Position distribution:

We use software to supply the vertical and horizontal coordinates to hardware so that the hardware will know the position of different tiles. We divide our screen into 2 parts: left part is used to display the game screen and the right part is used to put some function buttons. We will use 5X7 squares as the game screen and each of them is formed by 36X36 pixels.

3.3 Audio

Background music is the musical setting of ODE TO JOY by Beethoven. The sheet music is recorded in hardware using different period of pitches. After importing a normalized sin wave, we use different sample frequencies to generate different pitches and we use a constant frequency clock to make a timer to beat.

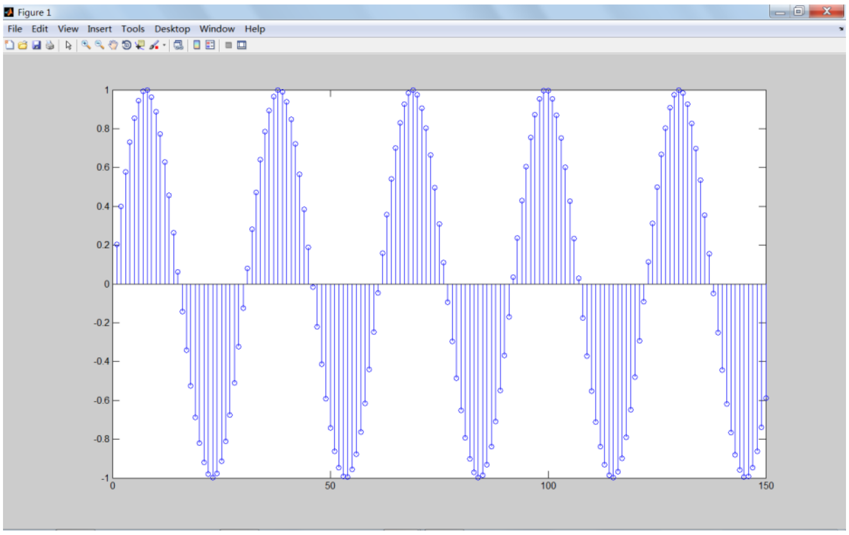
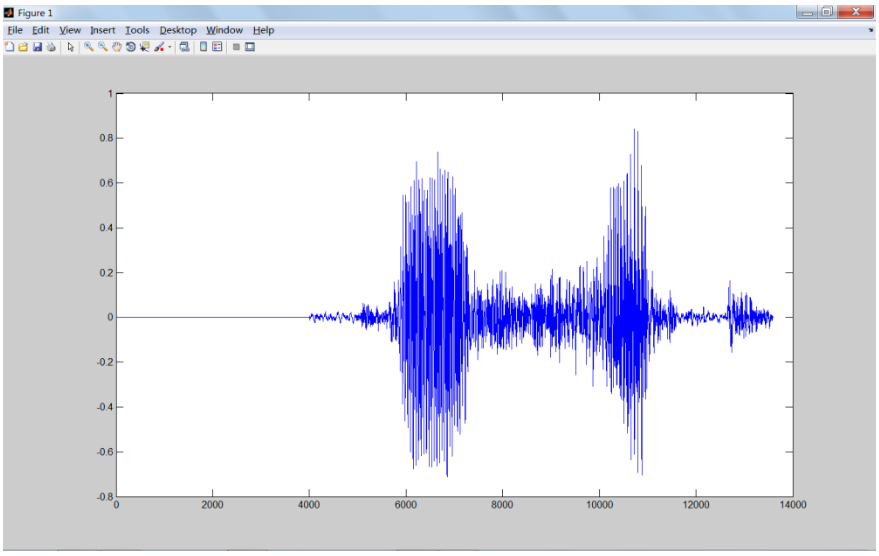


Figure 3.3.1

When the player chooses one snapper, it will scream because of fear. When the chosen snapper explodes, it will sound like a bomb. Either of them will last almost 1.5 seconds.

The original sample frequency of the recorder is 11025Hz. After I obtain the ".wav" file, we need to re-sample it as 8000Hz, because our sample frequency for audio part is designed as 8000Hz. The procedure fore creating these waveforms needs lots of analyses and calculations.



.

Figure 3.3.2

After normalizing this sound wave, I store it on chip, which fights against images because of the limitation of memory size. In order to decrease the size of this waveform, we can either decrease the sample frequency or the length of sound. To hear it clearly, we have to restrict the length down to 1.5 second.

Another difficulty to design the audio part is to make it synchronous with image changes. The delay of audio part will decrease the efficiency of software.

Additionally, The timing to superimpose the respond sound and background is also a problem because of different sample frequencies. To directly mix different sound signals together will lead to strange noise. There should be different timers to start and stop different audio signals in hardware. To make everything sounds smoothly during pressing the keyboard frequently needs a lot of time to debug.

4. Software

Defined structures

The software takes charge of both video and audio, and implements the mechanism and controls flow of the game.

The software needs to manage the status of each snapper and bullet displayed on the screen, decides how they should change their status from one to another, and when should a snapper generates a bullet.

For each snapper, we need to keep the information of its position(x ordinate and y ordinate), status, i.e. color, size and whether it is selected by the user and thus ready to be exploded or upgraded, and index number for VGA display.

There are 8 snapper types in total, 4 normal-sized of 4 color(blue, yellow, green, red) snapper with 4 enlarged-sized when they are selected.

```
struct snapper{
int x;      //x ordinate
int y;      //y ordinate
int num;    //index number for VGA display
int type;   //indicate the type of the snapper
int pointed; //indicate whether this particular snapper is currently selected
}
```

We implemented the first 10 stages of the game “Snappers” on the iOS platform, whose maximum snapper number that show in a single stage is 17, so we create a structure matrix of 17 elements of snapper types in the main function to store the information of all the snappers that appears in a single stage.

We store the initial status and position information of each stage in separate snapper matrices for convenience. In stage initialization, the corresponding information would be retrieved to set up the stage status.

Another important object in the game is the bullet. Bullet shares some attributes with snapper. Besides, the 4 types of bullet are up, down, left, right. Moreover, we need to record the initial position of the bullet to help telling whether a snapper is hit by a particular bullet.

```
struct bullet{
    int x;      //x ordinate
    int y;      //y ordinate
    int num;    //index number for VGA display
    int type;   //indicate the type of the bullet
    int init_x;   //x ordinate of the bullet when initially generated
    int init_y;   //y ordinate of the bullet when initially generated
};
```

Each snapper has 5 attributes and each bullet has 6, so it is quite a neat way to pack the attributes in structures and I think it’s more convenient than dealing with a bunch of integer matrix.

Function flow

We first initialize a snapper matrix, together with 4 bullet matrix, each of which corresponding to a direction among up/down/left/right. Then, level 0 is set to be the entrance stage and level_initialize() is called to set up the stage environment . Also show_level() and show_hit() are called to display the stage number and the hits left(once used up, you will lose). We also initiate a pointer for snapper selection and its position would be changed by the input of the keyboard.

Then the program enters an outer while loop to make the main process of the program run multiple times. After entering the while loop, the program examines which of the snappers is selected and enlarges it.

Then the program enters the inner while loop to continuously display the motion of the flying bullets (Here, we have a quite strong but still reasonable assumption that the player will wait until all the bullets finish flying). All of the 4 bullet matrices are scanned and if the type of a bullet is effective (not type_blank and within the flying range), its position would be updated (for example, right bullet moves rightward and thus right bullet would have its x ordinate plus 1). Meanwhile, the program checks if a bullet hits a snapper by calling hit() function, which traverses through the snapper matrix and the four bullet matrices. Once a bullet hits a snapper, there are two cases: case 1, the hit snapper is not a red one, then its type would be upgraded and color changed; case 2, the hit snapper is a red one, then it would be exploded (changing type to type_explode) and generate four new bullets and these new-born bullets would immediately begin flying across the screen. To make life easier, when a new bullet is generated, we just increment the corresponding pointer to the entry of the bullet structure matrix and write its information in the empty entries in the bullet matrices which we created at the beginning of main() function and their flying process would be implemented by the program scanning the matrix to update effective bullets' position. In the end of the inner while loop, the program examines if there are still some bullets in motion and set the corresponding indicating variables. If none of the current bullets is going to move, the program would not enter the inner while loop.

Also, the program calls stage_clear() to check if the play has successfully passed the current stage, that is, all the snappers of the current stage are exploded. If so, the screen is cleaned, and level got incremented. The program then initializes the status of the next stage, properly sets all the indicating variables to make the function working again. The program also checks if the play fails the stage (left hits less than zero).

The next part of the main() function is to respond to the input from the keyboard. Once one of the four arrow keys is pressed, the position of the pointer will change correspondingly. Once the player press ‘Enter’, the snapper would be upgraded or exploded.

5. Word division

Chi Zhang: VGA image display in hardware and the adjust VHDL code with C control to realize complicated figure generation.
Dian Wang: System integration, import figures into pixels array and display.

Lianyi Ding: Program C to realize software control of the whole game function.

Yuhan Dai: Take charge of the audio generation.

6. Codes

6.1 de2_vga_raster.vhd

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

entity de2_vga_raster is
   port (
      reset : in std_logic;
      clk    : in std_logic;                              -- Should be 25.125 MHz

      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 unsigned(9 downto 0); -- Blue[9:0]

      chipselect: in std_logic;
      write: in std_logic;
      address: in unsigned(15 downto 0);
      writedata: in unsigned(31 downto 0)

      );

end de2_vga_raster;

architecture rtl of de2_vga_raster is
```

```
    component frog_controller
  port(
clk      : in std_logic;
address: in unsigned (5 downto 0);
vertical: in integer;
horizontal: in integer;

pixel_R: out std_logic_vector(4 downto 0);
pixel_G: out std_logic_vector(4 downto 0);
pixel_B: out std_logic_vector(4 downto 0)
    );
end component;

component ps_controller
    port(
clk      : in std_logic;
address: in unsigned (5 downto 0);
vertical: in integer;
horizontal: in integer;


pixel_R: out std_logic_vector (4 downto 0);
pixel_G: out std_logic_vector (4 downto 0);
pixel_B: out std_logic_vector (4 downto 0)
            );

    end component;



    -- Video parameters
    signal Center_h:integer:=145;
    signal Center_v:integer:=75;
    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 lable_hstart1 : unsigned(9 downto 0);
    signal lable_vstart1 : unsigned(9 downto 0);
--------------------------------------------------
    signal buf_h_open, buf_v_open, buf_open : std_logic;    -- cursor area

    signal cursor_h, cursor_v, cursor_open : std_logic;    -- cursor area
```

```
   signal background_h, background_v, background_open,
ps_flag1,ps_flag2,ps_flag3,ps_flag4 : std_logic;    -- background area
   signal lable_v,lable_h, lable_open1, lable_flag1 : std_logic;    -- lable area
   signal Hscreen : integer;
   signal Vscreen : integer;
   signal clk25:std_logic:='0';--Clock with frequency 25MHz
   signal rectangle_h, rectangle_v, rectangle : std_logic;    -- rectangle area
   type snapper_row is array(0 to 35) of std_logic_vector(7 downto 0);      -- denote
tile content
   type snapper_column is array(0 to 35) of snapper_row;

   type bullet_row is array(0 to 11) of std_logic_vector(4 downto 0);      -- denote tile
content
   type bullet_column is array(0 to 11) of bullet_row;



   -----------------------------------------------------------------
   type background_row is array(16 downto 0) of std_logic_vector(0 downto 0);      --
denote ice position in map
   type background_column is array(12 downto 0) of background_row;

     signal map_level1 : background_column ;

------------------------------------------------------------------
   -- 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 v_buf1, h_buf1 : unsigned(9 downto 0);

signal v_buf, h_buf : unsigned(9 downto 0);
type sprite_hv_type is array(integer range 0 to 127) of unsigned(9 downto 0);
signal lable_hstart, lable_vstart,
frog_v,frog_h,
bullet_v, bullet_h : sprite_hv_type;
type hv_type is array(integer range 0 to 127) of unsigned(5 downto 0);
signal figure_type : hv_type;


   signal vga_hblank, vga_hsync,
      vga_vblank, vga_vsync : std_logic;    -- Sync. signals


   signal map_h, map_v : std_logic;
   signal h_count : integer := 0;
   signal v_count : integer := 0;

   signal frog_mode: std_logic_vector(63 downto 0) := (others => '0');
--   signal frog_mode: integer;
```

```
   signal h_frog_open, v_frog_open, frog_open :std_logic_vector(127 downto 0) :=
(others => '0');    -- rectangle area
   signal h_lable_open, v_lable_open, lable_open, lable_flag :std_logic_vector(127
downto 0) := (others => '0');    -- rectangle area

   signal h_bullet_open, v_bullet_open, bullet_open : std_logic_vector(127 downto
0) := (others => '0');    -- rectangle area

   signal explode_mode: integer :=0;

   signal frog_flag : integer := 30000000;
   signal frog_stage : integer := 0;
   signal frog_stage1 : integer := 0;
   signal frog_stage2 : integer := 0;
   signal frog_stage3 : integer := 0;

   signal explode_flag : integer := 0;
   signal explode_stage : integer := 0;
--timer
signal count : integer := 0;
signal timeflag : std_logic := '0';
signal timerange : integer := 500;

signal offset : integer := 0;
signal frog_R : std_logic_vector (4 downto 0);
signal frog_G : std_logic_vector (4 downto 0);
signal frog_B : std_logic_vector (4 downto 0);
signal ps_R : std_logic_vector (4 downto 0);
signal ps_G : std_logic_vector (4 downto 0);
signal ps_B : std_logic_vector (4 downto 0);



----------------------------------------ps----------------------------------------
  type timepattern1 is array(0 to 10) of unsigned(5 downto 0);
  type timepattern2 is array(integer range 0 to 3, integer range 0 to 16) of unsigned(5
downto 0);
  type timepattern3 is array(0 to 7) of unsigned(5 downto 0);



signal word_map1 : timepattern1 := (
"011001","011011","001110","011100","011100","111111","001110","010111","0111
01","001110","011011" );

signal word_map2 : timepattern2 :=
( -----------press N for
Next B for Back P for Pause S for Start
("011001","011011","001110","011100","011100","111111","010111","111111","0011
11","011000","011011","111111","010111","001110","100001", "011101","111111"),
("111111","111111","111111","111111","111111","111111","001011","111111","00111
1","011000","011011","111111","001011","001010","001011","010100","111111"),
("111111","111111","111111","111111","111111","111111","011001","111111","00111
1","011000","011011","111111","011001","001010","011110","011100","001110"),
("111111","111111","111111","111111","111111","111111","011100","111111","00111
1","011000","011011","111111","011100","011101","001010","011011","011101")
);

signal word_map3 : timepattern3 := (
  ("011100","010010","010110","001110", "111111", "111111","111111","111111")
);


  type background is array(integer range 0 to 12, integer range 0 to 16) of unsigned(5
downto 0);
  signal groundmap : background := (

("101100","101100","101100","101100","101100","101100","101100","101100","101
100","101100","101100","101100","101100","101100","101100","101100","101100")
,
("101100","101100","101100","101100","101100","101100","101100","101100","101
100","101100","101100","101100","101100","101100","101100","101100","101100")
,
("101100","101100","101100","101100","101100","101100","101100","101100","101
100","101100","101100","101100","101100","101100","101100","101100","101100")
,
("101100","101100","101100","101100","101100","101100","101100","101100","101
100","101100","101100","101100","101100","101100","101100","101100","101100")
,
("101100","101100","101100","101100","101100","101100","101100","101100","101
100","101100","101100","101100","101100","101100","101100","101100","101100")
,
("101100","101100","101100","101100","101100","101100","101100","101100","101
100","101100","101100","101100","101100","101100","101100","101100","101100")
,
("101100","101100","101100","101100","101100","101100","101100","101100","101
100","101100","101100","101100","101100","101100","101100","101100","101100")
,
("101100","101100","101100","101100","101100","101100","101100","101100","101
100","101100","101100","101100","101100","101100","101100","101100","101100")
,
("101100","101100","101100","101100","101100","101100","101100","101100","101
100","101100","101100","101100","101100","101100","101100","101100","101100")
,
("101100","101100","101100","101100","101100","101100","101100","101100","101
100","101100","101100","101100","101100","101100","101100","101100","101100")
,
("101100","101100","101100","101100","101100","101100","101100","101100","101
100","101100","101100","101100","101100","101100","101100","101100","101100")
,
("101100","101100","101100","101100","101100","101100","101100","101100","101
100","101100","101100","101100","101100","101100","101100","101100","101100")
```

```
,
("101100","101100","101100","101100","101100","101100","101100","101100","101
100","101100","101100","101100","101100","101100","101100","101100","101100")
);


    signal map1_hcount, map1_vcount: integer :=0;
    signal ps_address: unsigned (5 downto 0);
    signal ps_vertical,ps_horizontal: integer;
    signal Hnumber : integer;
    signal Vnumber : integer;
    signal TempHnumber : integer;

------------------------------------------------------------------------------------

signal test1: integer;
signal test2: integer;


signal image_address : unsigned(5 downto 0);
signal figure_num : integer;
signal figure_num1 : integer;
signal figure_num2 : integer;
signal figure_num3 : integer;
signal figure_num4 : integer;

signal h_cursor, v_cursor: unsigned(9 downto 0);

begin
h_buf <= writedata(9 downto 0);
v_buf <= writedata(19 downto 10);
figure_num <= to_integer(writedata(26 downto 20));

--


tile_rgb:                    frog_controller                    port                    map
(clk,image_address,test1,test2,frog_R,frog_G,frog_B);
ps_rgb:                      ps_controller                      port                    map
(clk,ps_address,ps_vertical,ps_horizontal,ps_R,ps_G,ps_B);





------------------------------------------------------------------------------------------
process (clk)
    begin
        if rising_edge(clk) then
            clk25 <= not clk25;
```

```
      end if;
   end process;
--------------------------------------------------------
DataProcess: process (clk)
   begin
      if rising_edge(clk) then

            if chipselect = '1' then
               if address(4 downto 0)    = "00000" then    --open frog
               if write = '1' then
               frog_h(    figure_num    ) <= h_buf;
               frog_v(    figure_num    ) <= v_buf;
--                 figure_type(    figure_num    ) <=    "001001";
               figure_type(    figure_num    ) <= address(10 downto 5);
               end if;
               elsif address(4 downto 0)    = "00001" then --open bullet
               if write = '1' then
               bullet_h(    figure_num    ) <= h_buf;
               bullet_v(    figure_num    ) <= v_buf;
--                 figure_type(    figure_num    ) <=    "010000";
               figure_type(    figure_num    ) <= address(10 downto 5);
               end if;
               elsif address(4 downto 0)    = "00100" then -- open lable
               if write = '1' then
----                    if address(10 downto 5) = "001010" or address(10 downto 5) = "001010" or address(10 downto 5) = "000010" or address(10 downto 5) = "000110" or address(10 downto 5) = "011011" then
               lable_flag1    <='1';
               lable_hstart1 <= h_buf;
               lable_vstart1 <= v_buf;
               end if;

               elsif address(4 downto 0)    = "00101" then --open press enter
               if write = '1' then
               ps_flag1 <= '1';
               end if;

               elsif address(4 downto 0)    = "00110" then --buttun explain
               if write = '1' then
               ps_flag2 <= '1';
               end if;

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


---------------------------------------------------------------------------
   -- Horizontal and vertical counters
```

```
HCounter : process (clk25)
begin
   if rising_edge(clk25) 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 (clk25)
begin
   if rising_edge(clk25) then
      if reset = '1' then
         Vcount <= (others => '0');
      elsif EndOfLine = '1' then
         if EndOfField = '1' then
            Vcount <= (others => '0');
         else
            Vcount <= Vcount + 1;
         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 (clk25)
begin
   if rising_edge(clk25) 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 (clk25)
begin
   if rising_edge(clk25) 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 (clk25)
   begin
      if rising_edge(clk25) 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 (clk25)
   begin
      if rising_edge(clk25) 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;
frogstaget: process (clk25)
begin
if rising_edge(clk25) then

            if frog_flag <= 30000000 then
            frog_stage <= 0;
            frog_flag <= frog_flag+1;
            elsif frog_flag <= 35000000 then
            frog_stage <= 1;
            frog_flag <= frog_flag+1;
            else frog_flag <= 0;
end if;
```

```
            if frog_flag <= 50000000 then
            frog_stage1 <= 0;
            frog_flag <= frog_flag+1;
            elsif frog_flag <= 58000000 then
            frog_stage1 <= 1;
            frog_flag <= frog_flag+1;
            else frog_flag <= 0;
end if;
            if frog_flag <= 70000000 then
            frog_stage2 <= 0;
            frog_flag <= frog_flag+1;
            elsif frog_flag <= 80000000 then
            frog_stage2 <= 1;
            frog_flag <= frog_flag+1;
            else frog_flag <= 0;
end if;
            if frog_flag <= 90000000 then
            frog_stage3 <= 0;
            frog_flag <= frog_flag+1;
            elsif frog_flag <= 120000000 then
            frog_stage3 <= 1;
            frog_flag <= frog_flag+1;
            else frog_flag <= 0;
end if;

            if explode_flag <= 40000000 then
            explode_stage <= 0;
            explode_flag <= explode_flag+1;
            elsif explode_flag <= 600000000 then
            explode_stage <= 1;
            explode_flag <= explode_flag+1;
            elsif explode_flag <= 80000000 then
            explode_stage <= 2;
            explode_flag <= explode_flag+1;


   end if;

end if;
end process;
----lable gen
LableHGen: process (clk)
begin
if rising_edge(clk) then
--for i in 0 to 60 loop
if reset = '1' or Hcount = HSYNC + HBACK_PORCH + lable_hstart1 +1 then
lable_h    <= '1';
end if;
if reset = '1' or  Hcount = HSYNC + HBACK_PORCH + lable_hstart1 + 330+1
then
```

```
lable_h    <= '0';
end if;
--end loop;
end if;
end process LableHGen;

LableVGen: process (clk)
begin
if rising_edge(clk) then
--for i in 0 to 60 loop
if reset = '1' then
lable_v <= '0';
elsif EndOfLine = '1' then
if Vcount = VSYNC + VBACK_PORCH + lable_vstart1    +6    then
lable_v <= '1';
end if;
if Vcount = VSYNC + VBACK_PORCH + lable_vstart1    + 72 +6 then
lable_v <= '0';
end if;
end if;
--end loop;
end if;
end process LableVGen;

lable_open1<= lable_v and lable_h ;



---------------frog generate
RectangleHGen: process (clk)
begin
if rising_edge(clk) then
for i in 0 to 60 loop
if reset = '1' or Hcount = HSYNC + HBACK_PORCH + frog_h(i)    then
h_frog_open(i)    <= '1';
end if;
if reset = '1' or    Hcount = HSYNC + HBACK_PORCH + frog_h(i) + 36    then
h_frog_open(i)    <= '0';
end if;
end loop;
end if;
end process RectangleHGen;

RectangleVGen: process (clk)
begin
if rising_edge(clk) then
for i in 0 to 60 loop
if reset = '1' then
v_frog_open(i) <= '0';
elsif EndOfLine = '1' then
if Vcount = VSYNC + VBACK_PORCH + frog_v(i)    +6    then
```

```
v_frog_open(i) <= '1';
end if;
if Vcount = VSYNC + VBACK_PORCH + frog_v(i)    + 36 +6 then
v_frog_open(i) <= '0';
end if;
end if;
end loop;
end if;
end process RectangleVGen;
------------------------------------------------------------
BulletHGen: process (clk)
begin
if rising_edge(clk) then
for i in 0 to 60 loop
if reset = '1' or Hcount = HSYNC + HBACK_PORCH + bullet_h(i) +1 then
h_bullet_open(i) <= '1';
end if;
if reset = '1' or    Hcount = HSYNC + HBACK_PORCH + bullet_h(i) + 12+1 then
h_bullet_open(i) <= '0';
end if;
end loop;
end if;
end process BulletHGen;

BulletVGen: process (clk)
begin
if rising_edge(clk) then
for i in 0 to 60 loop
if reset = '1' then
v_bullet_open(i) <= '0';
elsif EndOfLine = '1' then
if Vcount = VSYNC + VBACK_PORCH + bullet_v(i)    +6 - 1 then
v_bullet_open(i) <= '1';
end if;
if Vcount = VSYNC + VBACK_PORCH + bullet_v(i) + 12 +6 -1 then
v_bullet_open(i) <= '0';
end if;
end if;
end loop;
end if;
end process BulletVGen;
-------------------------------------------------------------

frog_open(1)<= v_frog_open(1) and h_frog_open(1) ;
frog_open(2)<= v_frog_open(2) and h_frog_open(2) ;
frog_open(3)<= v_frog_open(3) and h_frog_open(3) ;
frog_open(4)<= v_frog_open(4) and h_frog_open(4) ;
frog_open(5)<= v_frog_open(5) and h_frog_open(5) ;
frog_open(6)<= v_frog_open(6) and h_frog_open(6) ;
frog_open(7)<= v_frog_open(7) and h_frog_open(7) ;
frog_open(8)<= v_frog_open(8) and h_frog_open(8) ;
frog_open(9)<= v_frog_open(9) and h_frog_open(9) ;
frog_open(10)<= v_frog_open(10) and h_frog_open(10) ;

frog_open(11)<= v_frog_open(11) and h_frog_open(11) ;
frog_open(12)<= v_frog_open(12) and h_frog_open(12) ;
frog_open(13)<= v_frog_open(13) and h_frog_open(13) ;
frog_open(14)<= v_frog_open(14) and h_frog_open(14) ;
frog_open(15)<= v_frog_open(15) and h_frog_open(15) ;
frog_open(16)<= v_frog_open(16) and h_frog_open(16) ;
frog_open(17)<= v_frog_open(17) and h_frog_open(17) ;
frog_open(18)<= v_frog_open(18) and h_frog_open(18) ;
frog_open(19)<= v_frog_open(19) and h_frog_open(19) ;
frog_open(20)<= v_frog_open(20) and h_frog_open(20) ;

frog_open(21)<= v_frog_open(21) and h_frog_open(21) ;
frog_open(22)<= v_frog_open(22) and h_frog_open(22) ;
frog_open(23)<= v_frog_open(23) and h_frog_open(23) ;
frog_open(24)<= v_frog_open(24) and h_frog_open(24) ;
frog_open(25)<= v_frog_open(25) and h_frog_open(25) ;
frog_open(26)<= v_frog_open(26) and h_frog_open(26) ;
frog_open(27)<= v_frog_open(27) and h_frog_open(27) ;
frog_open(28)<= v_frog_open(28) and h_frog_open(28) ;
frog_open(29)<= v_frog_open(29) and h_frog_open(29) ;
frog_open(30)<= v_frog_open(30) and h_frog_open(30) ;

frog_open(31)<= v_frog_open(31) and h_frog_open(31) ;
frog_open(32)<= v_frog_open(32) and h_frog_open(32) ;
frog_open(33)<= v_frog_open(33) and h_frog_open(33) ;
frog_open(34)<= v_frog_open(34) and h_frog_open(34) ;
frog_open(35)<= v_frog_open(35) and h_frog_open(35) ;
frog_open(36)<= v_frog_open(36) and h_frog_open(36) ;
frog_open(37)<= v_frog_open(37) and h_frog_open(37) ;
frog_open(38)<= v_frog_open(38) and h_frog_open(38) ;
frog_open(39)<= v_frog_open(39) and h_frog_open(39) ;
frog_open(40)<= v_frog_open(40) and h_frog_open(40) ;

frog_open(41)<= v_frog_open(41) and h_frog_open(41) ;
frog_open(42)<= v_frog_open(42) and h_frog_open(42) ;
frog_open(43)<= v_frog_open(43) and h_frog_open(43) ;
frog_open(44)<= v_frog_open(44) and h_frog_open(44) ;
frog_open(45)<= v_frog_open(45) and h_frog_open(45) ;
bullet_open(1)<= v_bullet_open(1) and h_bullet_open(1) ;
bullet_open(2)<= v_bullet_open(2) and h_bullet_open(2) ;
bullet_open(3)<= v_bullet_open(3) and h_bullet_open(3) ;
bullet_open(4)<= v_bullet_open(4) and h_bullet_open(4) ;
bullet_open(5)<= v_bullet_open(5) and h_bullet_open(5) ;
bullet_open(6)<= v_bullet_open(6) and h_bullet_open(6) ;
bullet_open(7)<= v_bullet_open(7) and h_bullet_open(7) ;
bullet_open(8)<= v_bullet_open(8) and h_bullet_open(8) ;
bullet_open(9)<= v_bullet_open(9) and h_bullet_open(9) ;
bullet_open(10)<= v_bullet_open(10) and h_bullet_open(10) ;

bullet_open(11)<= v_bullet_open(11) and h_bullet_open(11) ;
bullet_open(12)<= v_bullet_open(12) and h_bullet_open(12) ;
bullet_open(13)<= v_bullet_open(13) and h_bullet_open(13) ;
bullet_open(14)<= v_bullet_open(14) and h_bullet_open(14) ;
bullet_open(15)<= v_bullet_open(15) and h_bullet_open(15) ;
bullet_open(16)<= v_bullet_open(16) and h_bullet_open(16) ;
bullet_open(17)<= v_bullet_open(17) and h_bullet_open(17) ;
bullet_open(18)<= v_bullet_open(18) and h_bullet_open(18) ;
bullet_open(19)<= v_bullet_open(19) and h_bullet_open(19) ;
bullet_open(20)<= v_bullet_open(20) and h_bullet_open(20) ;

bullet_open(21)<= v_bullet_open(21) and h_bullet_open(21) ;
bullet_open(22)<= v_bullet_open(22) and h_bullet_open(22) ;
bullet_open(23)<= v_bullet_open(23) and h_bullet_open(23) ;
bullet_open(24)<= v_bullet_open(24) and h_bullet_open(24) ;
bullet_open(25)<= v_bullet_open(25) and h_bullet_open(25) ;
bullet_open(26)<= v_bullet_open(26) and h_bullet_open(26) ;
bullet_open(27)<= v_bullet_open(27) and h_bullet_open(27) ;
bullet_open(28)<= v_bullet_open(28) and h_bullet_open(28) ;
bullet_open(29)<= v_bullet_open(29) and h_bullet_open(29) ;
bullet_open(30)<= v_bullet_open(30) and h_bullet_open(30) ;

bullet_open(31)<= v_bullet_open(31) and h_bullet_open(31) ;
bullet_open(32)<= v_bullet_open(32) and h_bullet_open(32) ;
bullet_open(33)<= v_bullet_open(33) and h_bullet_open(33) ;
bullet_open(34)<= v_bullet_open(34) and h_bullet_open(34) ;
bullet_open(35)<= v_bullet_open(35) and h_bullet_open(35) ;
bullet_open(36)<= v_bullet_open(36) and h_bullet_open(36) ;
bullet_open(37)<= v_bullet_open(37) and h_bullet_open(37) ;
bullet_open(38)<= v_bullet_open(38) and h_bullet_open(38) ;
bullet_open(39)<= v_bullet_open(39) and h_bullet_open(39) ;
bullet_open(40)<= v_bullet_open(40) and h_bullet_open(40) ;

bullet_open(41)<= v_bullet_open(41) and h_bullet_open(41) ;
bullet_open(42)<= v_bullet_open(42) and h_bullet_open(42) ;
bullet_open(43)<= v_bullet_open(43) and h_bullet_open(43) ;
bullet_open(44)<= v_bullet_open(44) and h_bullet_open(44) ;
bullet_open(45)<= v_bullet_open(45) and h_bullet_open(45) ;


  VideoOut: process (clk25, reset)
    begin
        if reset = '1' then
            VGA_R <= "0000000000";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";
elsif clk25'event and clk25 = '1' then
```

```
if vga_hblank = '0' and vga_vblank ='0' then

                VGA_R <= "0000000000";
                VGA_G <= "0000000000";
                VGA_B <= "0000000000";
            else
              VGA_R <= "0000000000";
              VGA_G <= "0000000000";
                VGA_B <= "0000000000";


end if;



for i in 0 to 60 loop
if frog_open(i) = '1' then
        VGA_R(4 downto 0) <= "11111";
        VGA_G(4 downto 0) <= "11111";
        VGA_B(4 downto 0) <= "11111";
test1 <=    to_integer(Vcount - frog_v(i)-7) -VSYNC - VBACK_PORCH;
test2 <=    to_integer(Hcount - frog_h(i)) -HSYNC - HBACK_PORCH;
image_address <= figure_type(i);
if                figure_type(i)                =               "001011"                then
--green_big
if frog_stage3 = 0 then


        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
else
            image_address <= "001100";
        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
end if;
elsif                figure_type(i)                =               "001001"                then
--green_small
--
  if frog_stage3 = 0 then
        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
else
        image_address <= "001010";
        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
```

```
end if;
  elsif figure_type(i) = "001111" then                                        --red_big

  if frog_stage2 = 0 then
      VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
      VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
      VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
else
        image_address <= "010000";
      VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
      VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
      VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
      end if;
elsif figure_type(i) = "001101" then                                      --red small

  if frog_stage2 = 0 then
      VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
      VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
      VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
else
      image_address <= "001110";
      VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
      VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
      VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
end if;
   elsif figure_type(i) = "000011" then                              --blue big
  if frog_stage1 = 0 then

      VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
      VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
      VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
else
        image_address <= "000100";
      VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
      VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
      VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
end if;
elsif figure_type(i) = "000001" then                               --blue small
if frog_stage1 = 0 then
      VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
      VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
      VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
else
        image_address <= "000010";
      VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
      VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
      VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
end if;
    elsif figure_type(i) = "000111" then                       --orange big
if frog_stage = 0 then
```

```
    VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
    VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
    VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
else
      image_address <= "001000";
    VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
    VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
    VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
end if;
   elsif figure_type(i) = "000101" then                    --orange small
if frog_stage = 0 then
    VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
    VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
    VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
else
      image_address <= "000110";
    VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
    VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
    VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
end if;
------------------------------------button-------------------------------------
elsif figure_type(i) = "011000" then                                --start
    VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
    VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
    VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
elsif figure_type(i) = "011001" then                                --pause
    VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
    VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
    VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
elsif figure_type(i) = "011010" then                                --next
    VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
    VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
    VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
elsif figure_type(i) = "011011" then                                --back
    VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
    VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
    VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
---------------------------------------explode------------------------------------
elsif figure_type(i) = "010101" then
if explode_stage = 0 then
    VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
    VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
    VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
 elsif   explode_stage = 1 then
      image_address <= "010110";
    VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
    VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
    VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
   elsif   explode_stage = 2 then
```

```
        VGA_R(9 downto 0) <= "0000000000";
        VGA_G(9 downto 0) <= "0000000000";
        VGA_B(9 downto 0) <= "0000000000";
--      VGA_R(9 downto 2) <= unsigned(map_black(to_integer(Vcount -frog_v(i) -6)
- (VSYNC + VBACK_PORCH +2))(to_integer(Hcount-frog_h(i) )- (HSYNC +
HBACK_PORCH )));
--      VGA_G(9 downto 2) <= unsigned(map_black(to_integer(Vcount -frog_v(i) -6)
- (VSYNC + VBACK_PORCH +2))(to_integer(Hcount-frog_h(i) )- (HSYNC +
HBACK_PORCH )));
--      VGA_B(9 downto 2) <= unsigned(map_black(to_integer(Vcount -frog_v(i) -6)
- (VSYNC + VBACK_PORCH +2))(to_integer(Hcount-frog_h(i) )- (HSYNC +
HBACK_PORCH )));
end if;
---------------------------------------------------arrow back 0-9-------------
elsif figure_type(i) = "011101" then ---lv

        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
elsif figure_type(i) = "011110" then ---hit
        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
elsif figure_type(i) = "011100" then ---arrow
        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
elsif figure_type(i) = "011111" then --0
        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
elsif figure_type(i) = "100000" then --1
        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
elsif figure_type(i) = "100001" then --2
        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
elsif figure_type(i) = "100010" then --3
        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
elsif figure_type(i) = "100011" then --4
        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
elsif figure_type(i) = "100100" then --5
        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
elsif figure_type(i) = "100101" then --6
        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
elsif figure_type(i) = "100110" then --7
        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
elsif figure_type(i) = "100111" then --8
        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
elsif figure_type(i) = "101000" then --9
        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));


elsif bullet_open(i) = '1' then
test1 <=    to_integer(Vcount - bullet_v(i)-6 - 1 ) -VSYNC - VBACK_PORCH;
test2 <=    to_integer(Hcount - bullet_h(i) +2 ) -HSYNC - HBACK_PORCH;
image_address <= figure_type(i);

if figure_type(i) = "010001" then                              --bullet up
        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));

elsif figure_type(i) = "010010" then                        --bullet down

        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));


elsif figure_type(i) = "010011" then                      --bullet left

        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));

elsif figure_type(i) = "010100" then                        --bullet right

        VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
        VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
        VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
```

```
end if;
end if;
          end loop;
   if lable_flag1 = '1' then
   if lable_open1 = '1' then      ---lable
      image_address <= address(10 downto 5);
      VGA_R(4 downto 0) <= "11111";
      VGA_G(4 downto 0) <= "11111";
      VGA_B(4 downto 0) <= "11111";
test1 <=    to_integer(Vcount - lable_vstart1-6) -VSYNC - VBACK_PORCH;
test2 <=    to_integer(Hcount - lable_hstart1+1) -HSYNC - HBACK_PORCH;
      VGA_R(9 downto 5) <= unsigned(frog_R(4 downto 0));
      VGA_G(9 downto 5) <= unsigned(frog_G(4 downto 0));
      VGA_B(9 downto 5) <= unsigned(frog_B(4 downto 0));
end if;
elsif ps_flag1 = '1' then    ----press enter
Vnumber    <= to_integer(Vcount) - VBACK_PORCH - VSYNC;
Hnumber <=    to_integer(Hcount) - HBACK_PORCH    - HSYNC;
if Vnumber >= 210 and Vnumber < 226 and Hnumber >= 210 and Hnumber < 298
then
ps_horizontal <= Hnumber mod 8;
ps_vertical <= (Vnumber - 210) mod 16;
ps_address <= word_map1((Hnumber-210)/8);
      VGA_R(9 downto 5) <= unsigned(ps_R(4 downto 0));
      VGA_G(9 downto 5) <= unsigned(ps_G(4 downto 0));
      VGA_B(9 downto 5) <= unsigned(ps_B(4 downto 0));
end if;
elsif ps_flag2 = '1' then    ---button explain
Vnumber    <= to_integer(Vcount) - VBACK_PORCH - VSYNC;
Hnumber <=    to_integer(Hcount) - HBACK_PORCH    - HSYNC;
if Vnumber >= 400 and Vnumber < 464 and Hnumber >= 400 and Hnumber < 536
then
ps_horizontal <= Hnumber mod 8;
ps_vertical <= (Vnumber - 400) mod 16;
ps_address <= word_map2((Vnumber - 400)/16, (Hnumber - 400)/8);
      VGA_R(9 downto 5) <= unsigned(ps_R(4 downto 0));
      VGA_G(9 downto 5) <= unsigned(ps_G(4 downto 0));
      VGA_B(9 downto 5) <= unsigned(ps_B(4 downto 0));
end if;

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

   end process VideoOut;

   VGA_CLK <= clk25;
   VGA_HS <= not vga_hsync;
   VGA_VS <= not vga_vsync;
   VGA_SYNC <= '0';
   VGA_BLANK <= not (vga_hsync or vga_vsync);

end rtl;
```

6.2 frog_controller.vhd

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

          entity frog_controller is
             port(
          clk     : in std_logic;
          address: in unsigned (5 downto 0);
          vertical: in integer;
          horizontal: in integer;

          pixel_R: out std_logic_vector (4 downto 0);
          pixel_G: out std_logic_vector (4 downto 0);
          pixel_B: out std_logic_vector (4 downto 0)
             );

          end frog_controller;

          architecture rtl of frog_controller is

          -------------------------------------------------------frog_red_big----------------------
    ----------
          component frog_red_big_r
             port(
          clk     : in std_logic;
          addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
             );
          end component;

          component frog_red_big_g
             port(
          clk     : in std_logic;
          addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
             );
          end component;

          component frog_red_big_b
             port(
          clk     : in std_logic;
          addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
```

```
        );
    end component;
    -------------------------------------------------------------frog_red_big_mirror-------------------------
    component frog_red_big_mirror_r
        port(
    clk      : in std_logic;
    addr    : in integer;
    data    : out std_logic_vector (4 downto 0)
        );
    end component;

    component frog_red_big_mirror_g
        port(
    clk      : in std_logic;
    addr    : in integer;
    data    : out std_logic_vector (4 downto 0)
        );
    end component;

    component frog_red_big_mirror_b
        port(
    clk      : in std_logic;
    addr    : in integer;
    data    : out std_logic_vector (4 downto 0)
        );
    end component;


    -----------------------------------------------------------------frog_blue_big------------------------
    component frog_blue_big_r
        port(
    clk      : in std_logic;
    addr    : in integer;
    data    : out std_logic_vector (4 downto 0)
        );
    end component;

    component frog_blue_big_g
        port(
    clk      : in std_logic;
    addr    : in integer;
    data    : out std_logic_vector (4 downto 0)
        );
    end component;

    component frog_blue_big_b
        port(
    clk      : in std_logic;
```

```
     addr    : in integer;
     data    : out std_logic_vector (4 downto 0)
        );
     end component;
     --
     ------------------------------------------------------------frog_blue_big_mirror------
-----------------------
     component frog_blue_big_mirror_r
        port(
     clk      : in std_logic;
     addr    : in integer;
     data    : out std_logic_vector (4 downto 0)
        );
     end component;

     component frog_blue_big_mirror_g
        port(
     clk      : in std_logic;
     addr    : in integer;
     data    : out std_logic_vector (4 downto 0)
        );
     end component;

     component frog_blue_big_mirror_b
        port(
     clk      : in std_logic;
     addr    : in integer;
     data    : out std_logic_vector (4 downto 0)
        );
     end component;

     ----------------------------------------------------------start-------------------------
     component map_start_r
        port(
     clk      : in std_logic;
     addr    : in integer;
     data    : out std_logic_vector (4 downto 0)
        );
     end component;

     component map_start_g
        port(
     clk      : in std_logic;
     addr    : in integer;
     data    : out std_logic_vector (4 downto 0)
        );
     end component;

     component map_start_b
        port(
     clk      : in std_logic;
     addr    : in integer;
     data    : out std_logic_vector (4 downto 0)
        );
     end component;
     ------------------------------------------------------------back------------------------
     component map_back_r
        port(
     clk      : in std_logic;
     addr    : in integer;
     data    : out std_logic_vector (4 downto 0)
        );
     end component;

     component map_back_g
        port(
     clk      : in std_logic;
     addr    : in integer;
     data    : out std_logic_vector (4 downto 0)
        );
     end component;

     component map_back_b
        port(
     clk      : in std_logic;
     addr    : in integer;
     data    : out std_logic_vector (4 downto 0)
        );
     end component;
     ----------------------------------------------------------next------------------------
     component map_next_r
        port(
     clk      : in std_logic;
     addr    : in integer;
     data    : out std_logic_vector (4 downto 0)
        );
     end component;

     component map_next_g
        port(
     clk      : in std_logic;
     addr    : in integer;
     data    : out std_logic_vector (4 downto 0)
        );
     end component;

     component map_next_b
        port(
     clk      : in std_logic;
```

```
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        ------------------------------------------------------------------pause---------------------------
        component map_pause_r
           port(
        clk      : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;

        component map_pause_g
           port(
        clk      : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;

        component map_pause_b
           port(
        clk      : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        ------------------------------------------------------------------bulletup------------------------
---
        component map_bulletup_r
           port(
        clk      : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        ------------------------------------------------------------------bulletdown--------------------
-------
        component map_bulletdown_r
           port(
        clk      : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        ----------------------------------------------------------------bulletleft-------------------------
--
        component map_bulletleft_r
           port(
        clk     : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        ------------------------------------------------------------bulletright-------------------
------
        component map_bulletright_r
           port(
        clk     : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        ----------------------------------------------------------explode1----------------------
--
        component map_explode1_r
           port(
        clk     : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        ------------------------------------------------------------explode2--------------------
----
        component map_explode2_r
           port(
        clk     : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        --------------------------------------------------------------map_lv--------------------
----
        component map_lv
           port(
        clk     : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        --------------------------------------------------------------map_hit-------------------
-----
        component map_hit
           port(
        clk     : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        ------------------------------------------------------------map_0---------------------
---
        component map_0
           port(
        clk      : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        ------------------------------------------------------------map_1---------------------
---
        component map_1
           port(
        clk      : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        ------------------------------------------------------------map_2---------------------
---
        component map_2
           port(
        clk      : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        ------------------------------------------------------------map_3---------------------
---
        component map_3
           port(
        clk      : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        ------------------------------------------------------------map_4---------------------
---
        component map_4
           port(
        clk      : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        ------------------------------------------------------------map_5---------------------
---
        component map_5
           port(
        clk      : in std_logic;
```

```
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        ------------------------------------------------------------map_6---------------------
---
        component map_6
           port(
        clk      : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        ------------------------------------------------------------map_7---------------------
---
        component map_7
           port(
        clk      : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        ------------------------------------------------------------map_8---------------------
---
        component map_8
           port(
        clk      : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        ------------------------------------------------------------map_9---------------------
---
        component map_9
           port(
        clk      : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        ------------------------------------------------------------map_Good_Job-----------
-------------
        component map_Good_Job
           port(
        clk      : in std_logic;
        addr    : in integer;
        data    : out std_logic_vector (4 downto 0)
           );
        end component;
        ------------------------------------------------------------map_Snappers------------
```

```
-------------
      component map_Snappers
         port(
      clk      : in std_logic;
      addr    : in integer;
      data    : out std_logic_vector (4 downto 0)
         );
      end component;
      ---------------------------------------------------------------map_Try_Again----------
---------------
      component map_Try_Again
         port(
      clk      : in std_logic;
      addr    : in integer;
      data    : out std_logic_vector (4 downto 0)
         );
      end component;
      --------------------------------------------------------arrow------------------------------
      component map_arrow_r
         port(
      clk      : in std_logic;
      addr    : in integer;
      data    : out std_logic_vector (4 downto 0)
         );
      end component;

      component map_background_r
         port(
      clk      : in std_logic;
      addr    : in integer;
      data    : out std_logic_vector (4 downto 0)
         );
      end component;

      component map_background_g
         port(
      clk      : in std_logic;
      addr    : in integer;
      data    : out std_logic_vector (4 downto 0)
         );
      end component;

      component map_background_b
         port(
      clk      : in std_logic;
      addr    : in integer;
      data    : out std_logic_vector (4 downto 0)
         );
      end component;
```

```
        signal ver : integer;
        signal hor : integer;


        signal frog_g_b_R,frog_g_b_G,frog_g_b_B : std_logic_vector(4 downto 0);
        signal frog_g_b_m_R,frog_g_b_m_G,frog_g_b_m_B : std_logic_vector(4
downto 0);
        signal frog_g_s_R,frog_g_s_G,frog_g_s_B : std_logic_vector(4 downto 0);
        signal frog_g_s_m_R,frog_g_s_m_G,frog_g_s_m_B : std_logic_vector(4
downto 0);

        signal frog_r_b_R,frog_r_b_G,frog_r_b_B : std_logic_vector(4 downto 0);
        signal frog_r_b_m_R,frog_r_b_m_G,frog_r_b_m_B : std_logic_vector(4
downto 0);
        signal frog_r_s_R,frog_r_s_G,frog_r_s_B : std_logic_vector(4 downto 0);
        signal frog_r_s_m_R,frog_r_s_m_G,frog_r_s_m_B : std_logic_vector(4
downto 0);


        signal frog_b_b_R,frog_b_b_G,frog_b_b_B : std_logic_vector(4 downto 0);
        signal frog_b_b_m_R,frog_b_b_m_G,frog_b_b_m_B : std_logic_vector(4
downto 0);
        signal frog_b_s_R,frog_b_s_G,frog_b_s_B : std_logic_vector(4 downto 0);
        signal frog_b_s_m_R,frog_b_s_m_G,frog_b_s_m_B : std_logic_vector(4
downto 0);

        signal frog_o_b_R,frog_o_b_G,frog_o_b_B : std_logic_vector(4 downto 0);
        signal frog_o_b_m_R,frog_o_b_m_G,frog_o_b_m_B : std_logic_vector(4
downto 0);
        signal frog_o_s_R,frog_o_s_G,frog_o_s_B : std_logic_vector(4 downto 0);
        signal frog_o_s_m_R,frog_o_s_m_G,frog_o_s_m_B : std_logic_vector(4
downto 0);

        signal start_R,start_G,start_B : std_logic_vector(4 downto 0);
        signal next_R,next_G,next_B : std_logic_vector(4 downto 0);
        signal back_R,back_G,back_B : std_logic_vector(4 downto 0);
        signal pause_R,pause_G,pause_B : std_logic_vector(4 downto 0);

        signal bulletdown_R,bulletdown_G,bulletdown_B : std_logic_vector(4
downto 0);
        signal bulletup_R,bulletup_G,bulletup_B : std_logic_vector(4 downto 0);
        signal bulletleft_R,bulletleft_G,bulletleft_B : std_logic_vector(4 downto 0);
        signal bulletright_R,bulletright_G,bulletright_B : std_logic_vector(4 downto
0);
        signal black_R,black_G,black_B : std_logic_vector(4 downto 0);
        signal explode1_R,explode1_G,explode1_B : std_logic_vector(4 downto 0);
        signal explode2_R,explode2_G,explode2_B : std_logic_vector(4 downto 0);
        -----------------------------------------
        signal map_lv_R,map_lv_G,map_lv_B : std_logic_vector(4 downto 0);
        signal map_hit_R,map_hit_G,map_hit_B : std_logic_vector(4 downto 0);
```

```
	signal map_0_R,map_0_G,map_0_B : std_logic_vector(4 downto 0);
	signal map_1_R,map_1_G,map_1_B : std_logic_vector(4 downto 0);
	signal map_2_R,map_2_G,map_2_B : std_logic_vector(4 downto 0);
	signal map_3_R,map_3_G,map_3_B : std_logic_vector(4 downto 0);
	signal map_4_R,map_4_G,map_4_B : std_logic_vector(4 downto 0);
	signal map_5_R,map_5_G,map_5_B : std_logic_vector(4 downto 0);
	signal map_6_R,map_6_G,map_6_B : std_logic_vector(4 downto 0);
	signal map_7_R,map_7_G,map_7_B : std_logic_vector(4 downto 0);
	signal map_8_R,map_8_G,map_8_B : std_logic_vector(4 downto 0);
	signal map_9_R,map_9_G,map_9_B : std_logic_vector(4 downto 0);
	signal map_Snappers_R,map_Snappers_G,map_Snappers_B :
std_logic_vector(4 downto 0);
	signal map_Try_Again_R,map_Try_Again_G,map_Try_Again_B :
std_logic_vector(4 downto 0);
	signal map_Good_Job_R,map_Good_Job_G,map_Good_Job_B :
std_logic_vector(4 downto 0);
	---------------------------------------
	signal arrow_R,arrow_G,arrow_B:std_logic_vector(4 downto 0);
------------arrow
	signal background_R,background_G,background_B : std_logic_vector(4
downto 0);


	begin
	---------------------------------------green ----------------------------------
	frog_green_big_red: frog_red_big_g port map (clk,ver*36+hor,frog_g_b_R);
	frog_green_big_green: frog_red_big_r port map
(clk,ver*36+hor,frog_g_b_G);
	frog_green_big_blue: frog_red_big_g port map
(clk,ver*36+hor,frog_g_b_B);

	frog_green_big_mirror_red: frog_red_big_mirror_g port map
(clk,ver*36+hor,frog_g_b_m_R);
	frog_green_big_mirror_green: frog_red_big_mirror_r port map
(clk,ver*36+hor,frog_g_b_m_G);
	frog_green_big_mirror_blue: frog_red_big_mirror_g port map
(clk,ver*36+hor,frog_g_b_m_B);

	frog_green_small_red: frog_blue_big_r port map
(clk,ver*36+hor,frog_g_s_R);
	frog_green_small_green: frog_blue_big_b port map
(clk,ver*36+hor,frog_g_s_G);
	frog_green_small_blue: frog_blue_big_r port map
(clk,ver*36+hor,frog_g_s_B);

	frog_green_small_mirror_red: frog_blue_big_mirror_r port map
(clk,ver*36+hor,frog_g_s_m_R);
	frog_green_small_mirror_green: frog_blue_big_mirror_b port map
(clk,ver*36+hor,frog_g_s_m_G);
	frog_green_small_mirror_blue: frog_blue_big_mirror_r port map
```

```
(clk,ver*36+hor,frog_g_s_m_B);

        ---------------------------------------red ----------------------------------
        frog_red_big_red: frog_red_big_r port map (clk,ver*36+hor,frog_r_b_R);
        frog_red_big_green: frog_red_big_g port map (clk,ver*36+hor,frog_r_b_G);
        frog_red_big_blue: frog_red_big_g port map (clk,ver*36+hor,frog_r_b_B);

        frog_red_big_mirror_red: frog_red_big_mirror_r port map
(clk,ver*36+hor,frog_r_b_m_R);
        frog_red_big_mirror_green: frog_red_big_mirror_g port map
(clk,ver*36+hor,frog_r_b_m_G);
        frog_red_big_mirror_blue: frog_red_big_mirror_b port map
(clk,ver*36+hor,frog_r_b_m_B);

        frog_red_small_red: frog_blue_big_b port map (clk,ver*36+hor,frog_r_s_R);
        frog_red_small_green: frog_blue_big_r port map
(clk,ver*36+hor,frog_r_s_G);
        frog_red_small_blue: frog_blue_big_r port map
(clk,ver*36+hor,frog_r_s_B);

        frog_red_small_mirror_red: frog_blue_big_mirror_b port map
(clk,ver*36+hor,frog_r_s_m_R);
        frog_red_small_mirror_green: frog_blue_big_mirror_r port map
(clk,ver*36+hor,frog_r_s_m_G);
        frog_red_small_mirror_blue: frog_blue_big_mirror_r port map
(clk,ver*36+hor,frog_r_s_m_B);

        ---------------------------------------blue ----------------------------------
        frog_blue_big_red: frog_red_big_g port map (clk,ver*36+hor,frog_b_b_R);
        frog_blue_big_green: frog_red_big_g port map
(clk,ver*36+hor,frog_b_b_G);
        frog_blue_big_blue: frog_red_big_r port map (clk,ver*36+hor,frog_b_b_B);

        frog_blue_big_mirror_red: frog_red_big_mirror_g port map
(clk,ver*36+hor,frog_b_b_m_R);
        frog_blue_big_mirror_green: frog_red_big_mirror_g port map
(clk,ver*36+hor,frog_b_b_m_G);
        frog_blue_big_mirror_blue: frog_red_big_mirror_r port map
(clk,ver*36+hor,frog_b_b_m_B);

        frog_blue_small_red: frog_blue_big_r port map
(clk,ver*36+hor,frog_b_s_R);
        frog_blue_small_green: frog_blue_big_r port map
(clk,ver*36+hor,frog_b_s_G);
        frog_blue_small_blue: frog_blue_big_b port map
(clk,ver*36+hor,frog_b_s_B);

        frog_blue_small_mirror_red: frog_blue_big_mirror_r port map
(clk,ver*36+hor,frog_b_s_m_R);
        frog_blue_small_mirror_green: frog_blue_big_mirror_r port map
```

```
(clk,ver*36+hor,frog_b_s_m_G);
	frog_blue_small_mirror_blue: frog_blue_big_mirror_b port map
(clk,ver*36+hor,frog_b_s_m_B);

	---------------------------------------orange ----------------------------------
	frog_orange_big_red: frog_red_big_r port map
(clk,ver*36+hor,frog_o_b_R);
	frog_orange_big_green: frog_red_big_r port map
(clk,ver*36+hor,frog_o_b_G);
	frog_orange_big_blue: frog_red_big_g port map
(clk,ver*36+hor,frog_o_b_B);

	frog_orange_big_mirror_red: frog_red_big_mirror_r port map
(clk,ver*36+hor,frog_o_b_m_R);
	frog_orange_big_mirror_green: frog_red_big_mirror_r port map
(clk,ver*36+hor,frog_o_b_m_G);
	frog_orange_big_mirror_blue: frog_red_big_mirror_g port map
(clk,ver*36+hor,frog_o_b_m_B);

	frog_orange_small_red: frog_blue_big_b port map
(clk,ver*36+hor,frog_o_s_R);
	frog_orange_small_green: frog_blue_big_b port map
(clk,ver*36+hor,frog_o_s_G);
	frog_orange_small_blue: frog_blue_big_r port map
(clk,ver*36+hor,frog_o_s_B);

	frog_orange_small_mirror_red: frog_blue_big_mirror_b port map
(clk,ver*36+hor,frog_o_s_m_R);
	frog_orange_small_mirror_green: frog_blue_big_mirror_b port map
(clk,ver*36+hor,frog_o_s_m_G);
	frog_orange_small_mirror_blue: frog_blue_big_mirror_r port map
(clk,ver*36+hor,frog_o_s_m_B);

	-----------------------------------bullet ----------------------------------
	bulletup_red: map_bulletup_r port map (clk,ver*12+hor,bulletup_R);
	bulletup_green: map_bulletup_r port map (clk,ver*12+hor,bulletup_G);
	bulletup_blue: map_bulletup_r port map (clk,ver*12+hor,bulletup_B);

	bulletdown_red: map_bulletdown_r port map
(clk,ver*12+hor,bulletdown_R);
	bulletdown_green: map_bulletdown_r port map
(clk,ver*12+hor,bulletdown_G);
	bulletdown_blue: map_bulletdown_r port map
(clk,ver*12+hor,bulletdown_B);

	bulletleft_red: map_bulletleft_r port map (clk,ver*12+hor,bulletleft_R);
	bulletleft_green: map_bulletleft_r port map (clk,ver*12+hor,bulletleft_G);
	bulletleft_blue: map_bulletleft_r port map (clk,ver*12+hor,bulletleft_B);

	bulletright_red: map_bulletright_r port map (clk,ver*12+hor,bulletright_R);
```

```
bulletright_green: map_bulletright_r port map (clk,ver*12+hor,bulletright_G);
bulletright_blue: map_bulletright_r port map (clk,ver*12+hor,bulletright_B);


--------------------------------------------------------arrow-----------------------------------------------
arrow_red: map_arrow_r port map (clk,ver*36+hor,arrow_R);
arrow_green: map_arrow_r port map (clk,ver*36+hor,arrow_G);
arrow_blue: map_arrow_r port map (clk,ver*36+hor,arrow_B);
--------------------------------------------------------lv-----------------------------------------------
lv_red: map_lv port map (clk,ver*36+hor,map_lv_R);
lv_green: map_lv port map (clk,ver*36+hor,map_lv_G);
lv_blue: map_lv port map (clk,ver*36+hor,map_lv_B);
--------------------------------------------------------hit-----------------------------------------------
hit_red: map_hit port map (clk,ver*36+hor,map_hit_R);
hit_green: map_hit port map (clk,ver*36+hor,map_hit_G);
hit_blue: map_hit port map (clk,ver*36+hor,map_hit_B);
--------------------------------------------------------0-----------------------------------------------
map_0_red: map_0 port map (clk,ver*36+hor,map_0_R);
map_0_green: map_0 port map (clk,ver*36+hor,map_0_G);
map_0_blue: map_0 port map (clk,ver*36+hor,map_0_B);
--------------------------------------------------------1-----------------------------------------------
map_1_red: map_1 port map (clk,ver*36+hor,map_1_R);
map_1_green: map_1 port map (clk,ver*36+hor,map_1_G);
map_1_blue: map_1 port map (clk,ver*36+hor,map_1_B);
--------------------------------------------------------2-----------------------------------------------
map_2_red: map_2 port map (clk,ver*36+hor,map_2_R);
map_2_green: map_2 port map (clk,ver*36+hor,map_2_G);
map_2_blue: map_2 port map (clk,ver*36+hor,map_2_B);
--------------------------------------------------------3-----------------------------------------------
map_3_red: map_3 port map (clk,ver*36+hor,map_3_R);
map_3_green: map_3 port map (clk,ver*36+hor,map_3_G);
map_3_blue: map_3 port map (clk,ver*36+hor,map_3_B);
--------------------------------------------------------4-----------------------------------------------
map_4_red: map_4 port map (clk,ver*36+hor,map_4_R);
map_4_green: map_4 port map (clk,ver*36+hor,map_4_G);
map_4_blue: map_4 port map (clk,ver*36+hor,map_4_B);
--------------------------------------------------------5-----------------------------------------------
map_5_red: map_5 port map (clk,ver*36+hor,map_5_R);
map_5_green: map_5 port map (clk,ver*36+hor,map_5_G);
map_5_blue: map_5 port map (clk,ver*36+hor,map_5_B);
--------------------------------------------------------6-----------------------------------------------
map_6_red: map_6 port map (clk,ver*36+hor,map_6_R);
map_6_green: map_6 port map (clk,ver*36+hor,map_6_G);
map_6_blue: map_6 port map (clk,ver*36+hor,map_6_B);
--------------------------------------------------------7-----------------------------------------------
map_7_red: map_7 port map (clk,ver*36+hor,map_7_R);
map_7_green: map_7 port map (clk,ver*36+hor,map_7_G);
map_7_blue: map_7 port map (clk,ver*36+hor,map_7_B);
--------------------------------------------------------8-----------------------------------------------
```

```
        map_8_red: map_8 port map (clk,ver*36+hor,map_8_R);
        map_8_green: map_8 port map (clk,ver*36+hor,map_8_G);
        map_8_blue: map_8 port map (clk,ver*36+hor,map_8_B);
        ------------------------------------------------9----------------------------------------
        map_9_red: map_9 port map (clk,ver*36+hor,map_9_R);
        map_9_green: map_9 port map (clk,ver*36+hor,map_9_G);
        map_9_blue: map_9 port map (clk,ver*36+hor,map_9_B);
        ------------------------------------------------Snappers----------------------------------------
-
        map_Snappers_red: map_Snappers port map
(clk,ver*330+hor,map_Snappers_R);
        map_Snappers_green: map_Snappers port map
(clk,ver*330+hor,map_Snappers_G);
        map_Snappers_blue: map_Snappers port map
(clk,ver*330+hor,map_Snappers_B);
        ------------------------------------------------Good_Job---------------------------------------
--
        map_Good_Job_red: map_Good_Job port map
(clk,ver*330+hor,map_Good_Job_R);
        map_Good_Job_green: map_Good_Job port map
(clk,ver*330+hor,map_Good_Job_G);
        map_Good_Job_blue: map_Good_Job port map
(clk,ver*330+hor,map_Good_Job_B);
        -----------------------------------------------Try_Again---------------------------------------
--
        map_Try_Again_red: map_Try_Again port map
(clk,ver*330+hor,map_Try_Again_R);
        map_Try_Again_green: map_Try_Again port map
(clk,ver*330+hor,map_Try_Again_G);
        map_Try_Again_blue: map_Try_Again port map
(clk,ver*330+hor,map_Try_Again_B);
        -------------------------------------------explode-------------------------------------
        explode1_red: map_explode1_r port map (clk,ver*36+hor,explode1_R);
        explode1_green: map_explode1_r port map (clk,ver*36+hor,explode1_G);
        explode1_blue: map_explode1_r port map (clk,ver*36+hor,explode1_B);

        explode2_red: map_explode2_r port map (clk,ver*36+hor,explode2_R);
        explode2_green: map_explode2_r port map (clk,ver*36+hor,explode2_G);
        explode2_blue: map_explode2_r port map (clk,ver*36+hor,explode2_B);

        --black_red: map_black_r port map (clk,ver*36+hor,black_R);
        --black_green: map_black_r port map (clk,ver*36+hor,black_G);
        --black_blue: map_black_r port map (clk,ver*36+hor,black_B);
        -------------------------------------buttom -------------------------------------
        start_red: map_start_r port map (clk,ver*36+hor,start_R);
        start_green: map_start_g port map (clk,ver*36+hor,start_G);
        start_blue: map_start_b port map (clk,ver*36+hor,start_B);

        pause_red: map_pause_r port map (clk,ver*36+hor,pause_R);
        pause_green: map_pause_g port map (clk,ver*36+hor,pause_G);
```

```
    pause_blue: map_pause_b port map (clk,ver*36+hor,pause_B);

    next_red: map_next_r port map (clk,ver*36+hor,next_R);
    next_green: map_next_g port map (clk,ver*36+hor,next_G);
    next_blue: map_next_b port map (clk,ver*36+hor,next_B);

    back_red: map_back_r port map (clk,ver*36+hor,back_R);
    back_green: map_back_g port map (clk,ver*36+hor,back_G);
    back_blue: map_back_b port map (clk,ver*36+hor,back_B);
    ------------------------------------------------------------
    background_red:          map_background_r          port          map
(clk,ver*36+hor,background_R);
    background_green:          map_background_g          port          map
(clk,ver*36+hor,background_G);
    background_blue:          map_background_b          port          map
(clk,ver*36+hor,background_B);



    process(clk)
    begin
    if rising_edge(clk) then
       hor <= horizontal;
       ver <= vertical;
    end if;
    end process;

    process(clk)

    begin
    if rising_edge(clk) then
    case address is
    ---------------------------------------green--------------------------------------
      when "001011" =>                                    --gb
        if frog_g_b_R /= "00000" then
        pixel_R <= frog_g_b_R;
        else
        pixel_R <= "00000";
        end if;

        if frog_g_b_G /= "00000" then
        pixel_G <= frog_g_b_G;
        else
        pixel_G <= "00000";
        end if;

        if frog_g_b_B /= "00000" then
        pixel_B <= frog_g_b_B;
        else
        pixel_B <= "00000";
```

```
   end if;

 when "001100" =>                           -- gbm
   if frog_g_b_m_R /= "00000" then
   pixel_R <= frog_g_b_m_R;
   else
   pixel_R <= "00000";
   end if;

   if frog_g_b_m_G /= "00000" then
   pixel_G <= frog_g_b_m_G;
   else
   pixel_G <= "00000";
   end if;

   if frog_g_b_m_B /= "00000" then
   pixel_B <= frog_g_b_m_B;
   else
   pixel_B <= "00000";
   end if;

when "001001" =>                                 -- gs
    if frog_g_s_R /= "00000" then
   pixel_R <= frog_g_s_R;
   else
   pixel_R <= "00000";
   end if;

   if frog_g_s_G /= "00000" then
   pixel_G <= frog_g_s_G;
   else
   pixel_G <= "00000";
   end if;

   if frog_g_s_B /= "00000" then

   pixel_B <= frog_g_s_B;
   else
   pixel_B <= "00000";
   end if;

when "001010" =>                                 -- gsm
    if frog_g_s_m_R /= "00000" then
   pixel_R <= frog_g_s_m_R;
   else
   pixel_R <= "00000";
   end if;

   if frog_g_s_m_G /= "00000" then
   pixel_G <= frog_g_s_m_G;
```

```
    else
    pixel_G <= "00000";
    end if;

    if frog_g_s_m_B /= "00000" then
    pixel_B <= frog_g_s_m_B;
    else
    pixel_B <= "00000";
    end if;
--------------------------------------------------------------red--------------------------------
 when "001111" =>                                              --rb
    if frog_r_b_R /= "00000" then
    pixel_R <= frog_r_b_R;
    else
    pixel_R <= "00000";
    end if;

    if frog_r_b_R /= "00000" then
    pixel_G <= frog_r_b_G;
    else
    pixel_G <= "00000";
    end if;

    if frog_r_b_B /= "00000" then
    pixel_B <= frog_r_b_B;
    else
    pixel_B <= "00000";
    end if;

  when "010000" =>                                             -- rbm
    if frog_r_b_m_R /= "00000" then
    pixel_R <= frog_r_b_m_R;
    else
    pixel_R <= "00000";
    end if;

    if frog_r_b_m_G /= "00000" then
    pixel_G <= frog_r_b_m_G;
    else
    pixel_G <= "00000";
    end if;

    if frog_r_b_m_B /= "00000" then
    pixel_B <= frog_r_b_m_B;
    else
    pixel_B <= "00000";
    end if;

 when "001101" =>                                                     -- rs
     if frog_r_s_R /= "00000" then
```

```
             pixel_R <= frog_r_s_R;
             else
             pixel_R <= "00000";
             end if;

             if frog_r_s_G /= "00000" then
             pixel_G <= frog_r_s_G;
             else
             pixel_G <= "00000";
             end if;

             if frog_r_s_B /= "00000" then
             pixel_B <= frog_r_s_B;
             else
             pixel_B <= "00000";
             end if;

        when "001110" =>                                                  -- rsm
               if frog_r_s_m_R /= "00000" then
             pixel_R <= frog_r_s_m_R;
             else
             pixel_R <= "00000";
             end if;

             if frog_r_s_m_G /= "00000" then
             pixel_G <= frog_r_s_m_G;
             else
             pixel_G <= "00000";
             end if;

             if frog_r_s_m_B /= "00000" then
             pixel_B <= frog_r_s_m_B;
             else
             pixel_B <= "00000";
             end if;

      ------------------------------------------------------------blue---------------------------
-
        when "000011" =>                                          --bb
             if frog_b_b_R /= "00000" then
             pixel_R <= frog_b_b_R;
             else
             pixel_R <= "00000";
             end if;

             if frog_b_b_R /= "00000" then
             pixel_G <= frog_b_b_G;
             else
             pixel_G <= "00000";
             end if;
             if frog_b_b_B /= "00000" then
             pixel_B <= frog_b_b_B;
             else
             pixel_B <= "00000";
             end if;

         when "000100" =>                                    -- bbm
             if frog_b_b_m_R /= "00000" then
             pixel_R <= frog_b_b_m_R;
             else
             pixel_R <= "00000";
             end if;

             if frog_b_b_m_G /= "00000" then
             pixel_G <= frog_b_b_m_G;
             else
             pixel_G <= "00000";
             end if;

             if frog_b_b_m_B /= "00000" then
             pixel_B <= frog_b_b_m_B;
             else
             pixel_B <= "00000";
             end if;

        when "000001" =>                                           -- bs
               if frog_b_s_R /= "00000" then
             pixel_R <= frog_b_s_R;
             else
             pixel_R <= "00000";
             end if;

             if frog_b_s_G /= "00000" then
             pixel_G <= frog_b_s_G;
             else
             pixel_G <= "00000";
             end if;

             if frog_b_s_B /= "00000" then
             pixel_B <= frog_b_s_B;
             else
             pixel_B <= "00000";
             end if;

        when "000010" =>                                           -- bsm
               if frog_b_s_m_R /= "00000" then
             pixel_R <= frog_b_s_m_R;
             else
             pixel_R <= "00000";
```

```
            end if;

            if frog_b_s_m_G /= "00000" then
            pixel_G <= frog_b_s_m_G;
            else
            pixel_G <= "00000";
            end if;

            if frog_b_s_m_B /= "00000" then
            pixel_B <= frog_b_s_m_B;
            else
            pixel_B <= "00000";
            end if;

        ------------------------------------------------------------orange-----------------------
----
          when "000111" =>                                              --ob
            if frog_o_b_R /= "00000" then
            pixel_R <= frog_o_b_R;
            else
            pixel_R <= "00000";
            end if;

            if frog_o_b_R /= "00000" then
            pixel_G <= frog_o_b_G;
            else
            pixel_G <= "00000";
            end if;

            if frog_o_b_B /= "00000" then
            pixel_B <= frog_o_b_B;
            else
            pixel_B <= "00000";
            end if;

           when "001000" =>                                             -- obm
            if frog_o_b_m_R /= "00000" then
            pixel_R <= frog_o_b_m_R;
            else
            pixel_R <= "00000";
            end if;

            if frog_o_b_m_G /= "00000" then
            pixel_G <= frog_o_b_m_G;
            else
            pixel_G <= "00000";
            end if;

            if frog_o_b_m_B /= "00000" then
            pixel_B <= frog_o_b_m_B;
```

```
      else
      pixel_B <= "00000";
      end if;

  when "000101" =>                                    -- os
        if frog_o_s_R /= "00000" then
      pixel_R <= frog_o_s_R;
      else
      pixel_R <= "00000";
      end if;

      if frog_o_s_G /= "00000" then
      pixel_G <= frog_o_s_G;
      else
      pixel_G <= "00000";
      end if;

      if frog_o_s_B /= "00000" then
      pixel_B <= frog_o_s_B;
      else
      pixel_B <= "00000";
      end if;

  when "000110" =>                                    -- osm
        if frog_o_s_m_R /= "00000" then
      pixel_R <= frog_o_s_m_R;
      else
      pixel_R <= "00000";
      end if;

      if frog_o_s_m_G /= "00000" then
      pixel_G <= frog_o_s_m_G;
      else
      pixel_G <= "00000";
      end if;

      if frog_o_s_m_B /= "00000" then
      pixel_B <= frog_o_s_m_B;
      else
      pixel_B <= "00000";
      end if;
 ---------------------------------------------------------bullet--------------------------
-
  when "010001" =>                                --up
      if bulletup_R /= "00000" then
      pixel_R <= bulletup_R;
      else
      pixel_R <= "00000";
      end if;
```

```
    if bulletup_G /= "00000" then
    pixel_G <= bulletup_G;
    else
    pixel_G <= "00000";
    end if;

    if bulletup_B /= "00000" then
    pixel_B <= bulletup_B;
    else
    pixel_B <= "00000";
    end if;

  when "010010" =>                                    -- down
    if bulletdown_R /= "00000" then
    pixel_R <= bulletdown_R;
    else
    pixel_R <= "00000";
    end if;

    if bulletdown_G /= "00000" then
    pixel_G <= bulletdown_G;
    else
    pixel_G <= "00000";
    end if;

    if bulletdown_B /= "00000" then
    pixel_B <= bulletdown_B;
    else
    pixel_B <= "00000";
    end if;

when "010011" =>                                            -- left
     if bulletleft_R /= "00000" then
    pixel_R <= bulletleft_R;
    else
    pixel_R <= "00000";
    end if;

    if bulletleft_G /= "00000" then
    pixel_G <= bulletleft_G;
    else
    pixel_G <= "00000";
    end if;

    if bulletleft_B /= "00000" then
    pixel_B <= bulletleft_B;
    else
    pixel_B <= "00000";
    end if;
```

```
          when "010100" =>                                        -- right
               if bulletright_R /= "00000" then
             pixel_R <= bulletright_R;
             else
             pixel_R <= "00000";
             end if;

             if bulletright_G /= "00000" then
             pixel_G <= bulletright_G;
             else
             pixel_G <= "00000";
             end if;

             if bulletright_B /= "00000" then
             pixel_B <= bulletright_B;
             else
             pixel_B <= "00000";
             end if;
         ------------------------------------------------------------explode----------------------
------
          when "010101" =>
             if explode1_R /= "00000" then
             pixel_R <= explode1_R;
             else
             pixel_R <= "00000";
             end if;

             if explode1_G /= "00000" then
             pixel_G <= explode1_G;
             else
             pixel_G <= "00000";
             end if;

             if explode1_B /= "00000" then
             pixel_B <= explode1_B;
             else
             pixel_B <= "00000";
             end if;

           when "010110" =>
             if explode2_R /= "00000" then
             pixel_R <= explode2_R;
             else
             pixel_R <= "00000";
             end if;

             if explode2_G /= "00000" then
             pixel_G <= explode2_G;
             else
             pixel_G <= "00000";
```

```
        end if;

        if explode2_B /= "00000" then
        pixel_B <= explode2_B;
        else
        pixel_B <= "00000";
        end if;

-- when "010110" =>
--          if black_R /= "00000" then
--      pixel_R <= black_R;
--      else
--      pixel_R <= "00000";
--      end if;
--
--      if black_G /= "00000" then
--      pixel_G <= black_G;
--      else
--      pixel_G <= "00000";
--      end if;
--
--      if black_B /= "00000" then
--      pixel_B <= black_B;
--      else
--      pixel_B <= "00000";
--      end if;
----------------------------------------------button----------------------
when "011000" =>
        if start_R /= "00000" then
        pixel_R <= start_R;
        else
        pixel_R <= "00000";
        end if;

        if start_G /= "00000" then
        pixel_G <= start_G;
        else
        pixel_G <= "00000";
        end if;

        if start_B /= "00000" then
        pixel_B <= start_B;
        else
        pixel_B <= "00000";
        end if;
when "011001" =>
        if pause_R /= "00000" then
        pixel_R <= pause_R;
        else
        pixel_R <= "00000";
        end if;

        if pause_G /= "00000" then
        pixel_G <= pause_G;
        else
        pixel_G <= "00000";
        end if;

        if pause_B /= "00000" then
        pixel_B <= pause_B;
        else
        pixel_B <= "00000";
        end if;
when "011010" =>
        if next_R /= "00000" then
        pixel_R <= next_R;
        else
        pixel_R <= "00000";
        end if;

        if next_G /= "00000" then
        pixel_G <= next_G;
        else
        pixel_G <= "00000";
        end if;

        if next_B /= "00000" then
        pixel_B <= next_B;
        else
        pixel_B <= "00000";
        end if;
when "011011" =>
        if back_R /= "00000" then
        pixel_R <= back_R;
        else
        pixel_R <= "00000";
        end if;

        if back_G /= "00000" then
        pixel_G <= back_G;
        else
        pixel_G <= "00000";
        end if;

        if back_B /= "00000" then
        pixel_B <= back_B;
        else
        pixel_B <= "00000";
        end if;
```

```
--------------------------------------------------------------------------------
 when "011100" =>                                          --arrow
     if arrow_R /= "00000" then
     pixel_R <= arrow_R;
     else
     pixel_R <= "00000";
     end if;

     if arrow_G /= "00000" then
     pixel_G <= arrow_G;
     else
     pixel_G <= "00000";
     end if;

     if arrow_B /= "00000" then
     pixel_B <= arrow_B;
     else
     pixel_B <= "00000";
     end if;
--------------------------------------------------------------------------------
 when "011101" =>                                          --lv
     if map_lv_R /= "00000" then
     pixel_R <= map_lv_R;
     else
     pixel_R <= "00000";
     end if;

     if map_lv_G /= "00000" then
     pixel_G <= map_lv_G;
     else
     pixel_G <= "00000";
     end if;

     if map_lv_B /= "00000" then
     pixel_B <= map_lv_B;
     else
     pixel_B <= "00000";
     end if;

--------------------------------------------------------------------------------
 when "011110" =>                                          --hit
     if map_hit_R /= "00000" then
     pixel_R <= map_hit_R;
     else
     pixel_R <= "00000";
     end if;

     if map_hit_G /= "00000" then
     pixel_G <= map_hit_G;
     else
```

```
      pixel_G <= "00000";
      end if;

      if map_hit_B /= "00000" then
      pixel_B <= map_hit_B;
      else
      pixel_B <= "00000";
      end if;
-------------------------------------------------------------------------------
  when "011111" =>                                              --0
      if map_0_R /= "00000" then
      pixel_R <= map_0_R;
      else
      pixel_R <= "00000";
      end if;

      if map_0_G /= "00000" then
      pixel_G <= map_0_G;
      else
      pixel_G <= "00000";
      end if;

      if map_0_B /= "00000" then
      pixel_B <= map_0_B;
      else
      pixel_B <= "00000";
      end if;
-------------------------------------------------------------------------------
  when "100000" =>                                              --1
      if map_1_R /= "00000" then
      pixel_R <= map_1_R;
      else
      pixel_R <= "00000";
      end if;

      if map_1_G /= "00000" then
      pixel_G <= map_1_G;
      else
      pixel_G <= "00000";
      end if;

      if map_1_B /= "00000" then
      pixel_B <= map_1_B;
      else
      pixel_B <= "00000";
      end if;
-------------------------------------------------------------------------------
  when "100001" =>                                              --2
      if map_2_R /= "00000" then
      pixel_R <= map_2_R;
```

```
    else
    pixel_R <= "00000";
    end if;

    if map_2_G /= "00000" then
    pixel_G <= map_2_G;
    else
    pixel_G <= "00000";
    end if;

    if map_2_B /= "00000" then
    pixel_B <= map_2_B;
    else
    pixel_B <= "00000";
    end if;
------------------------------------------------------------------------------
 when "100010" =>                                              --3
    if map_3_R /= "00000" then
    pixel_R <= map_3_R;
    else
    pixel_R <= "00000";
    end if;

    if map_3_G /= "00000" then
    pixel_G <= map_3_G;
    else
    pixel_G <= "00000";
    end if;

    if map_3_B /= "00000" then
    pixel_B <= map_3_B;
    else
    pixel_B <= "00000";
    end if;
------------------------------------------------------------------------------
 when "100011" =>                                              --4
    if map_4_R /= "00000" then
    pixel_R <= map_4_R;
    else
    pixel_R <= "00000";
    end if;

    if map_4_G /= "00000" then
    pixel_G <= map_4_G;
    else
    pixel_G <= "00000";
    end if;

    if map_4_B /= "00000" then
    pixel_B <= map_4_B;
```

```
      else
      pixel_B <= "00000";
      end if;
-----------------------------------------------------------------------------
 when "100100" =>                                             --5
      if map_5_R /= "00000" then
      pixel_R <= map_5_R;
      else
      pixel_R <= "00000";
      end if;

      if map_5_G /= "00000" then
      pixel_G <= map_5_G;
      else
      pixel_G <= "00000";
      end if;

      if map_5_B /= "00000" then
      pixel_B <= map_5_B;
      else
      pixel_B <= "00000";
      end if;
-----------------------------------------------------------------------------
 when "100101" =>                                             --6
      if map_6_R /= "00000" then
      pixel_R <= map_6_R;
      else
      pixel_R <= "00000";
      end if;

      if map_6_G /= "00000" then
      pixel_G <= map_6_G;
      else
      pixel_G <= "00000";
      end if;

      if map_6_B /= "00000" then
      pixel_B <= map_6_B;
      else
      pixel_B <= "00000";
      end if;
-----------------------------------------------------------------------------
 when "100110" =>                                             --7
      if map_7_R /= "00000" then
      pixel_R <= map_7_R;
      else
      pixel_R <= "00000";
      end if;

      if map_7_G /= "00000" then
      pixel_G <= map_7_G;
      else
      pixel_G <= "00000";
      end if;

      if map_7_B /= "00000" then
      pixel_B <= map_7_B;
      else
      pixel_B <= "00000";
      end if;
-------------------------------------------------------------------------------
 when "100111" =>                                    --8
      if map_8_R /= "00000" then
      pixel_R <= map_8_R;
      else
      pixel_R <= "00000";
      end if;

      if map_8_G /= "00000" then
      pixel_G <= map_8_G;
      else
      pixel_G <= "00000";
      end if;

      if map_8_B /= "00000" then
      pixel_B <= map_8_B;
      else
      pixel_B <= "00000";
      end if;
-------------------------------------------------------------------------------
 when "101000" =>                                    --9
      if map_9_R /= "00000" then
      pixel_R <= map_9_R;
      else
      pixel_R <= "00000";
      end if;

      if map_9_G /= "00000" then
      pixel_G <= map_9_G;
      else
      pixel_G <= "00000";
      end if;

      if map_9_B /= "00000" then
      pixel_B <= map_9_B;
      else
      pixel_B <= "00000";
      end if;
-------------------------------------------------------------------------------
 when "101001" =>                                    --Snappers
```

```
     if map_Snappers_R /= "00000" then
     pixel_R <= map_Snappers_R;
     else
     pixel_R <= "00000";
     end if;

     if map_Snappers_G /= "00000" then
     pixel_G <= map_Snappers_G;
     else
     pixel_G <= "00000";
     end if;

     if map_Snappers_B /= "00000" then
     pixel_B <= map_Snappers_B;
     else
     pixel_B <= "00000";
     end if;
--------------------------------------------------------------------------------
  when "101010" =>                                          --Try_Again
     if map_Try_Again_R /= "00000" then
     pixel_R <= map_Try_Again_R;
     else
     pixel_R <= "00000";
     end if;

     if map_Try_Again_G /= "00000" then
     pixel_G <= map_Try_Again_G;
     else
     pixel_G <= "00000";
     end if;

     if map_Try_Again_B /= "00000" then
     pixel_B <= map_Try_Again_B;
     else
     pixel_B <= "00000";
     end if;
--------------------------------------------------------------------------------
  when "101011" =>                                          --Good_Job
     if map_Good_Job_R /= "00000" then
     pixel_R <= map_Good_Job_R;
     else
     pixel_R <= "00000";
     end if;

     if map_Good_Job_G /= "00000" then
     pixel_G <= map_Good_Job_G;
     else
     pixel_G <= "00000";
     end if;
```

```
        if map_Good_Job_B /= "00000" then
        pixel_B <= map_Good_Job_B;
        else
        pixel_B <= "00000";
        end if;
    --------------------------------------------------------------------------
     when "101100" =>                                        --background
        if background_R /= "00000" then
        pixel_R <= background_R;
        else
        pixel_R <= "00000";
        end if;

        if background_G /= "00000" then
        pixel_G <= background_G;
        else
        pixel_G <= "00000";
        end if;

        if background_B /= "00000" then
        pixel_B <= background_B;
        else
        pixel_B <= "00000";
        end if;




      when others =>
      pixel_R <= "00000";
      pixel_G <= "00000";
      pixel_B <= "00000";
    end case;

    end if;
    end process;

end rtl;
```

6.3 ps_controller.vhd

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

entity ps_controller is
  port(
clk      : in std_logic;
address: in unsigned (5 downto 0);
vertical: in integer;
horizontal: in integer;
pixel_R: out std_logic_vector (4 downto 0);
pixel_G: out std_logic_vector (4 downto 0);
pixel_B: out std_logic_vector (4 downto 0)
   );

end ps_controller;

architecture rtl of ps_controller is

----------------NUMBER COMPONENT-------------0-9
component ps_0_rom
   port(
           clk      : in std_logic;
           addr     : in integer;
           data     : out std_logic_vector (4 downto 0)
         );
end component;

component ps_1_rom
   port(
           clk      : in std_logic;
           addr     : in integer;
           data     : out std_logic_vector (4 downto 0)
         );
end component;

component ps_2_rom
   port(
           clk      : in std_logic;
           addr     : in integer;
           data     : out std_logic_vector (4 downto 0)
         );
end component;

component ps_3_rom
   port(
           clk      : in std_logic;
           addr     : in integer;
           data     : out std_logic_vector (4 downto 0)
         );
end component;

component ps_4_rom
   port(
           clk      : in std_logic;
           addr     : in integer;
           data     : out std_logic_vector (4 downto 0)
         );
end component;
```

```
component ps_5_rom
   port(
          clk      : in std_logic;
          addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
         );
end component;

component ps_6_rom
   port(
          clk      : in std_logic;
          addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
         );
end component;

component ps_7_rom
   port(
          clk      : in std_logic;
          addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
         );
end component;

component ps_8_rom
   port(
          clk      : in std_logic;
          addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
         );
end component;

component ps_9_rom
   port(
          clk      : in std_logic;
          addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
         );
end component;

--------------------LETTER COMPONENT-------------------A-X
component ps_A_rom
   port(
          clk      : in std_logic;
          addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
         );
end component;
```

```
component ps_B_rom
   port(
            clk      : in std_logic;
            addr    : in integer;
            data    : out std_logic_vector (4 downto 0)
          );
end component;

component ps_C_rom
   port(
            clk      : in std_logic;
            addr    : in integer;
            data    : out std_logic_vector (4 downto 0)
          );
end component;

component ps_D_rom
   port(
            clk      : in std_logic;
            addr    : in integer;
            data    : out std_logic_vector (4 downto 0)
          );
end component;

component ps_E_rom
   port(
            clk      : in std_logic;
            addr    : in integer;
            data    : out std_logic_vector (4 downto 0)
          );
end component;

component ps_F_rom
   port(
            clk      : in std_logic;
            addr    : in integer;
            data    : out std_logic_vector (4 downto 0)
          );
end component;

component ps_G_rom
   port(
            clk      : in std_logic;
            addr    : in integer;
            data    : out std_logic_vector (4 downto 0)
          );
end component;

component ps_H_rom
   port(
            clk     : in std_logic;
            addr    : in integer;
            data    : out std_logic_vector (4 downto 0)
          );
end component;

component ps_I_rom
   port(
            clk     : in std_logic;
            addr    : in integer;
            data    : out std_logic_vector (4 downto 0)
          );
end component;

component ps_J_rom
   port(
            clk     : in std_logic;
            addr    : in integer;
            data    : out std_logic_vector (4 downto 0)
          );
end component;

component ps_K_rom
   port(
            clk     : in std_logic;
            addr    : in integer;
            data    : out std_logic_vector (4 downto 0)
          );
end component;

component ps_L_rom
   port(
            clk     : in std_logic;
            addr    : in integer;
            data    : out std_logic_vector (4 downto 0)
          );
end component;

component ps_M_rom
   port(
            clk     : in std_logic;
            addr    : in integer;
            data    : out std_logic_vector (4 downto 0)
          );
end component;

component ps_N_rom
   port(
            clk     : in std_logic;
            addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
        );
end component;

component ps_O_rom
   port(
          clk      : in std_logic;
          addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
        );
end component;

component ps_P_rom
   port(
          clk      : in std_logic;
          addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
        );
end component;

component ps_Q_rom
   port(
          clk      : in std_logic;
          addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
        );
end component;

component ps_R_rom
   port(
          clk      : in std_logic;
          addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
        );
end component;

component ps_S_rom
   port(
          clk      : in std_logic;
          addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
        );
end component;

component ps_T_rom
   port(
          clk      : in std_logic;
          addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
        );
```

```
end component;

component ps_U_rom
   port(
          clk      : in std_logic;
          addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
        );
end component;

component ps_V_rom
   port(
          clk      : in std_logic;
          addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
        );
end component;

component ps_W_rom
   port(
          clk      : in std_logic;
          addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
        );
end component;

component ps_X_rom
   port(
          clk      : in std_logic;
          addr    : in integer;
          data    : out std_logic_vector (4 downto 0)
        );
end component;
---------------------------------------------------------------


signal ver : integer;
signal hor : integer;

signal ps_0_R,ps_0_G,ps_0_B : std_logic_vector(4 downto 0);
signal ps_1_R,ps_1_G,ps_1_B : std_logic_vector(4 downto 0);
signal ps_2_R,ps_2_G,ps_2_B : std_logic_vector(4 downto 0);
signal ps_3_R,ps_3_G,ps_3_B : std_logic_vector(4 downto 0);
signal ps_4_R,ps_4_G,ps_4_B : std_logic_vector(4 downto 0);
signal ps_5_R,ps_5_G,ps_5_B : std_logic_vector(4 downto 0);
signal ps_6_R,ps_6_G,ps_6_B : std_logic_vector(4 downto 0);
signal ps_7_R,ps_7_G,ps_7_B : std_logic_vector(4 downto 0);
signal ps_8_R,ps_8_G,ps_8_B : std_logic_vector(4 downto 0);
signal ps_9_R,ps_9_G,ps_9_B : std_logic_vector(4 downto 0);
signal ps_A_R,ps_A_G,ps_A_B : std_logic_vector(4 downto 0);
signal ps_B_R,ps_B_G,ps_B_B : std_logic_vector(4 downto 0);
signal ps_C_R,ps_C_G,ps_C_B : std_logic_vector(4 downto 0);
signal ps_D_R,ps_D_G,ps_D_B : std_logic_vector(4 downto 0);
signal ps_E_R,ps_E_G,ps_E_B : std_logic_vector(4 downto 0);
signal ps_F_R,ps_F_G,ps_F_B : std_logic_vector(4 downto 0);
signal ps_G_R,ps_G_G,ps_G_B : std_logic_vector(4 downto 0);
signal ps_H_R,ps_H_G,ps_H_B : std_logic_vector(4 downto 0);
signal ps_I_R,ps_I_G,ps_I_B : std_logic_vector(4 downto 0);
signal ps_J_R,ps_J_G,ps_J_B : std_logic_vector(4 downto 0);
signal ps_K_R,ps_K_G,ps_K_B : std_logic_vector(4 downto 0);
signal ps_L_R,ps_L_G,ps_L_B : std_logic_vector(4 downto 0);
signal ps_M_R,ps_M_G,ps_M_B : std_logic_vector(4 downto 0);
signal ps_N_R,ps_N_G,ps_N_B : std_logic_vector(4 downto 0);
signal ps_O_R,ps_O_G,ps_O_B : std_logic_vector(4 downto 0);
signal ps_P_R,ps_P_G,ps_P_B : std_logic_vector(4 downto 0);
signal ps_Q_R,ps_Q_G,ps_Q_B : std_logic_vector(4 downto 0);
signal ps_R_R,ps_R_G,ps_R_B : std_logic_vector(4 downto 0);
signal ps_S_R,ps_S_G,ps_S_B : std_logic_vector(4 downto 0);
signal ps_T_R,ps_T_G,ps_T_B : std_logic_vector(4 downto 0);
signal ps_U_R,ps_U_G,ps_U_B : std_logic_vector(4 downto 0);
signal ps_V_R,ps_V_G,ps_V_B : std_logic_vector(4 downto 0);
signal ps_W_R,ps_W_G,ps_W_B : std_logic_vector(4 downto 0);
signal ps_X_R,ps_X_G,ps_X_B : std_logic_vector(4 downto 0);



begin

ps_0_red: ps_0_rom port map (clk,ver*8+hor,ps_0_R);
ps_0_green: ps_0_rom port map (clk,ver*8+hor,ps_0_G);
ps_0_blue: ps_0_rom port map (clk,ver*8+hor,ps_0_B);

ps_1_red: ps_1_rom port map (clk,ver*8+hor,ps_1_R);
ps_1_green: ps_1_rom port map (clk,ver*8+hor,ps_1_G);
ps_1_blue: ps_1_rom port map (clk,ver*8+hor,ps_1_B);

ps_2_red: ps_2_rom port map (clk,ver*8+hor,ps_2_R);
ps_2_green: ps_2_rom port map (clk,ver*8+hor,ps_2_G);
ps_2_blue: ps_2_rom port map (clk,ver*8+hor,ps_2_B);

ps_3_red: ps_3_rom port map (clk,ver*8+hor,ps_3_R);
ps_3_green: ps_3_rom port map (clk,ver*8+hor,ps_3_G);
ps_3_blue: ps_3_rom port map (clk,ver*8+hor,ps_3_B);

ps_4_red: ps_4_rom port map (clk,ver*8+hor,ps_4_R);
ps_4_green: ps_4_rom port map (clk,ver*8+hor,ps_4_G);
ps_4_blue: ps_4_rom port map (clk,ver*8+hor,ps_4_B);
```

```
ps_5_red: ps_5_rom port map (clk,ver*8+hor,ps_5_R);
ps_5_green: ps_5_rom port map (clk,ver*8+hor,ps_5_G);
ps_5_blue: ps_5_rom port map (clk,ver*8+hor,ps_5_B);

ps_6_red: ps_6_rom port map (clk,ver*8+hor,ps_6_R);
ps_6_green: ps_6_rom port map (clk,ver*8+hor,ps_6_G);
ps_6_blue: ps_6_rom port map (clk,ver*8+hor,ps_6_B);

ps_7_red: ps_7_rom port map (clk,ver*8+hor,ps_7_R);
ps_7_green: ps_7_rom port map (clk,ver*8+hor,ps_7_G);
ps_7_blue: ps_7_rom port map (clk,ver*8+hor,ps_7_B);

ps_8_red: ps_8_rom port map (clk,ver*8+hor,ps_8_R);
ps_8_green: ps_8_rom port map (clk,ver*8+hor,ps_8_G);
ps_8_blue: ps_8_rom port map (clk,ver*8+hor,ps_8_B);

ps_9_red: ps_9_rom port map (clk,ver*8+hor,ps_9_R);
ps_9_green: ps_9_rom port map (clk,ver*8+hor,ps_9_G);
ps_9_blue: ps_9_rom port map (clk,ver*8+hor,ps_9_B);

ps_A_red: ps_A_rom port map (clk,ver*8+hor,ps_A_R);
ps_A_green: ps_A_rom port map (clk,ver*8+hor,ps_A_G);
ps_A_blue: ps_A_rom port map (clk,ver*8+hor,ps_A_B);

ps_B_red: ps_B_rom port map (clk,ver*8+hor,ps_B_R);
ps_B_green: ps_B_rom port map (clk,ver*8+hor,ps_B_G);
ps_B_blue: ps_B_rom port map (clk,ver*8+hor,ps_B_B);

ps_C_red: ps_C_rom port map (clk,ver*8+hor,ps_C_R);
ps_C_green: ps_C_rom port map (clk,ver*8+hor,ps_C_G);
ps_C_blue: ps_C_rom port map (clk,ver*8+hor,ps_C_B);

ps_D_red: ps_D_rom port map (clk,ver*8+hor,ps_D_R);
ps_D_green: ps_D_rom port map (clk,ver*8+hor,ps_D_G);
ps_D_blue: ps_D_rom port map (clk,ver*8+hor,ps_D_B);

ps_E_red: ps_E_rom port map (clk,ver*8+hor,ps_E_R);
ps_E_green: ps_E_rom port map (clk,ver*8+hor,ps_E_G);
ps_E_blue: ps_E_rom port map (clk,ver*8+hor,ps_E_B);

ps_F_red: ps_F_rom port map (clk,ver*8+hor,ps_F_R);
ps_F_green: ps_F_rom port map (clk,ver*8+hor,ps_F_G);
ps_F_blue: ps_F_rom port map (clk,ver*8+hor,ps_F_B);

ps_G_red: ps_G_rom port map (clk,ver*8+hor,ps_G_R);
ps_G_green: ps_G_rom port map (clk,ver*8+hor,ps_G_G);
ps_G_blue: ps_G_rom port map (clk,ver*8+hor,ps_G_B);

ps_H_red: ps_H_rom port map (clk,ver*8+hor,ps_H_R);
ps_H_green: ps_H_rom port map (clk,ver*8+hor,ps_H_G);
```

```
ps_H_blue: ps_H_rom port map (clk,ver*8+hor,ps_H_B);

ps_I_red: ps_I_rom port map (clk,ver*8+hor,ps_I_R);
ps_I_green: ps_I_rom port map (clk,ver*8+hor,ps_I_G);
ps_I_blue: ps_I_rom port map (clk,ver*8+hor,ps_I_B);

ps_J_red: ps_J_rom port map (clk,ver*8+hor,ps_J_R);
ps_J_green: ps_J_rom port map (clk,ver*8+hor,ps_J_G);
ps_J_blue: ps_J_rom port map (clk,ver*8+hor,ps_J_B);

ps_K_red: ps_K_rom port map (clk,ver*8+hor,ps_K_R);
ps_K_green: ps_K_rom port map (clk,ver*8+hor,ps_K_G);
ps_K_blue: ps_K_rom port map (clk,ver*8+hor,ps_K_B);

ps_L_red: ps_L_rom port map (clk,ver*8+hor,ps_L_R);
ps_L_green: ps_L_rom port map (clk,ver*8+hor,ps_L_G);
ps_L_blue: ps_L_rom port map (clk,ver*8+hor,ps_L_B);

ps_M_red: ps_M_rom port map (clk,ver*8+hor,ps_M_R);
ps_M_green: ps_M_rom port map (clk,ver*8+hor,ps_M_G);
ps_M_blue: ps_M_rom port map (clk,ver*8+hor,ps_M_B);

ps_N_red: ps_N_rom port map (clk,ver*8+hor,ps_N_R);
ps_N_green: ps_N_rom port map (clk,ver*8+hor,ps_N_G);
ps_N_blue: ps_N_rom port map (clk,ver*8+hor,ps_N_B);

ps_O_red: ps_O_rom port map (clk,ver*8+hor,ps_O_R);
ps_O_green: ps_O_rom port map (clk,ver*8+hor,ps_O_G);
ps_O_blue: ps_O_rom port map (clk,ver*8+hor,ps_O_B);

ps_P_red: ps_P_rom port map (clk,ver*8+hor,ps_P_R);
ps_P_green: ps_P_rom port map (clk,ver*8+hor,ps_P_G);
ps_P_blue: ps_P_rom port map (clk,ver*8+hor,ps_P_B);

ps_Q_red: ps_Q_rom port map (clk,ver*8+hor,ps_Q_R);
ps_Q_green: ps_Q_rom port map (clk,ver*8+hor,ps_Q_G);
ps_Q_blue: ps_Q_rom port map (clk,ver*8+hor,ps_Q_B);

ps_R_red: ps_R_rom port map (clk,ver*8+hor,ps_R_R);
ps_R_green: ps_R_rom port map (clk,ver*8+hor,ps_R_G);
ps_R_blue: ps_R_rom port map (clk,ver*8+hor,ps_R_B);

ps_S_red: ps_S_rom port map (clk,ver*8+hor,ps_S_R);
ps_S_green: ps_S_rom port map (clk,ver*8+hor,ps_S_G);
ps_S_blue: ps_S_rom port map (clk,ver*8+hor,ps_S_B);

ps_T_red: ps_T_rom port map (clk,ver*8+hor,ps_T_R);
ps_T_green: ps_T_rom port map (clk,ver*8+hor,ps_T_G);
ps_T_blue: ps_T_rom port map (clk,ver*8+hor,ps_T_B);
```

```
ps_U_red: ps_U_rom port map (clk,ver*8+hor,ps_U_R);
ps_U_green: ps_U_rom port map (clk,ver*8+hor,ps_U_G);
ps_U_blue: ps_U_rom port map (clk,ver*8+hor,ps_U_B);

ps_V_red: ps_V_rom port map (clk,ver*8+hor,ps_V_R);
ps_V_green: ps_V_rom port map (clk,ver*8+hor,ps_V_G);
ps_V_blue: ps_V_rom port map (clk,ver*8+hor,ps_V_B);

ps_W_red: ps_W_rom port map (clk,ver*8+hor,ps_W_R);
ps_W_green: ps_W_rom port map (clk,ver*8+hor,ps_W_G);
ps_W_blue: ps_W_rom port map (clk,ver*8+hor,ps_W_B);

ps_X_red: ps_X_rom port map (clk,ver*8+hor,ps_X_R);
ps_X_green: ps_X_rom port map (clk,ver*8+hor,ps_X_G);
ps_X_blue: ps_X_rom port map (clk,ver*8+hor,ps_X_B);





process(clk)
begin
if rising_edge(clk) then
    hor    <= horizontal;
    ver    <= vertical;
end if;
end process;


process(clk)

begin
if rising_edge(clk) then
case address is

   when "000000" =>

      if ps_0_R /= "00000" then
      pixel_R <= ps_0_R;
      else
      pixel_R <= "00000";
      end if;

      if ps_0_G /= "00000" then
      pixel_G <= ps_0_G;
      else
      pixel_G <= "00000";
      end if;

      if ps_0_B /= "00000" then
      pixel_B <= ps_0_B;
      else
      pixel_B <= "00000";
      end if;

   when "000001" =>

      if ps_1_R /= "00000" then
      pixel_R <= ps_1_R;
      else
      pixel_R <= "00000";
      end if;

      if ps_1_G /= "00000" then
      pixel_G <= ps_1_G;
      else
      pixel_G <= "00000";
      end if;

      if ps_1_B /= "00000" then
      pixel_B <= ps_1_B;
      else
      pixel_B <= "00000";
      end if;

   when "000010" =>

      if ps_2_R /= "00000" then
      pixel_R <= ps_2_R;
      else
      pixel_R <= "00000";
      end if;

      if ps_2_G /= "00000" then
      pixel_G <= ps_2_G;
      else
      pixel_G <= "00000";
      end if;

      if ps_2_B /= "00000" then
      pixel_B <= ps_2_B;
      else
      pixel_B <= "00000";
      end if;

   when "000011" =>

      if ps_3_R /= "00000" then
      pixel_R <= ps_3_R;
      else
```

```
   pixel_R <= "00000";
   end if;

   if ps_3_G /= "00000" then
   pixel_G <= ps_3_G;
   else
   pixel_G <= "00000";
   end if;

   if ps_3_B /= "00000" then
   pixel_B <= ps_3_B;
   else
   pixel_B <= "00000";
   end if;

when "000100" =>

   if ps_4_R /= "00000" then
   pixel_R <= ps_4_R;
   else
   pixel_R <= "00000";
   end if;

   if ps_4_G /= "00000" then
   pixel_G <= ps_4_G;
   else
   pixel_G <= "00000";
   end if;

   if ps_4_B /= "00000" then
   pixel_B <= ps_4_R;
   else
   pixel_B <= "00000";
   end if;

when "000101" =>

   if ps_5_R /= "00000" then
   pixel_R <= ps_5_R;
   else
   pixel_R <= "00000";
   end if;

   if ps_5_G /= "00000" then
   pixel_G <= ps_5_G;
   else
   pixel_G <= "00000";
   end if;

   if ps_5_B /= "00000" then
```

```
    pixel_B <= ps_5_B;
    else
    pixel_B <= "00000";
    end if;

when "000110" =>

    if ps_6_R /= "00000" then
    pixel_R <= ps_6_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_6_G /= "00000" then
    pixel_G <= ps_6_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_6_B /= "00000" then
    pixel_B <= ps_6_B;
    else
    pixel_B <= "00000";
    end if;

when "000111" =>

    if ps_7_R /= "00000" then
    pixel_R <= ps_7_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_7_G /= "00000" then
    pixel_G <= ps_7_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_7_B /= "00000" then
    pixel_B <= ps_7_B;
    else
    pixel_B <= "00000";
    end if;

when "001000" =>

    if ps_8_R /= "00000" then
    pixel_R <= ps_8_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_8_G /= "00000" then
    pixel_G <= ps_8_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_8_B /= "00000" then
    pixel_B <= ps_8_B;
    else
    pixel_B <= "00000";
    end if;

when "001001" =>

    if ps_9_R /= "00000" then
    pixel_R <= ps_9_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_9_G /= "00000" then
    pixel_G <= ps_9_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_9_B /= "00000" then
    pixel_B <= ps_9_B;
    else
    pixel_B <= "00000";
    end if;

when "001010" =>

    if ps_A_R /= "00000" then
    pixel_R <= ps_A_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_A_G /= "00000" then
    pixel_G <= ps_A_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_A_B /= "00000" then
    pixel_B <= ps_A_B;
    else
    pixel_B <= "00000";
    end if;

when "001011" =>

    if ps_B_R /= "00000" then
    pixel_R <= ps_B_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_B_G /= "00000" then
    pixel_G <= ps_B_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_B_B /= "00000" then
    pixel_B <= ps_B_B;
    else
    pixel_B <= "00000";
    end if;

when "001100" =>

    if ps_C_R /= "00000" then
    pixel_R <= ps_C_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_C_G /= "00000" then
    pixel_G <= ps_C_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_C_B /= "00000" then
    pixel_B <= ps_C_B;
    else
    pixel_B <= "00000";
    end if;

when "001101" =>

    if ps_D_R /= "00000" then
    pixel_R <= ps_D_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_D_G /= "00000" then
    pixel_G <= ps_D_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_D_B /= "00000" then
    pixel_B <= ps_D_B;
    else
    pixel_B <= "00000";
    end if;

when "001110" =>

    if ps_E_R /= "00000" then
    pixel_R <= ps_E_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_E_G /= "00000" then
    pixel_G <= ps_E_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_E_B /= "00000" then
    pixel_B <= ps_E_B;
    else
    pixel_B <= "00000";
    end if;

when "001111" =>

    if ps_F_R /= "00000" then
    pixel_R <= ps_F_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_F_G /= "00000" then
    pixel_G <= ps_F_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_F_B /= "00000" then
```

```
   pixel_B <= ps_F_B;
   else
   pixel_B <= "00000";
   end if;

when "010000" =>

   if ps_G_R /= "00000" then
   pixel_R <= ps_G_R;
   else
   pixel_R <= "00000";
   end if;

   if ps_G_G /= "00000" then
   pixel_G <= ps_G_G;
   else
   pixel_G <= "00000";
   end if;

   if ps_G_B /= "00000" then
   pixel_B <= ps_G_B;
   else
   pixel_B <= "00000";
   end if;

when "010001" =>

   if ps_H_R /= "00000" then
   pixel_R <= ps_H_R;
   else
   pixel_R <= "00000";
   end if;

   if ps_H_G /= "00000" then
   pixel_G <= ps_H_G;
   else
   pixel_G <= "00000";
   end if;

   if ps_H_B /= "00000" then
   pixel_B <= ps_H_B;
   else
   pixel_B <= "00000";
   end if;

when "010010" =>

   if ps_I_R /= "00000" then
   pixel_R <= ps_I_R;
   else
```

```
    pixel_R <= "00000";
    end if;

    if ps_I_G /= "00000" then
    pixel_G <= ps_I_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_I_B /= "00000" then
    pixel_B <= ps_I_B;
    else
    pixel_B <= "00000";
    end if;

when "010011" =>

    if ps_J_R /= "00000" then
    pixel_R <= ps_J_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_J_G /= "00000" then
    pixel_G <= ps_J_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_J_B /= "00000" then
    pixel_B <= ps_J_B;
    else
    pixel_B <= "00000";
    end if;

when "010100" =>

    if ps_K_R /= "00000" then
    pixel_R <= ps_K_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_K_G /= "00000" then
    pixel_G <= ps_K_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_K_B /= "00000" then
    pixel_B <= ps_K_B;
    else
    pixel_B <= "00000";
    end if;

when "010101" =>

    if ps_L_R /= "00000" then
    pixel_R <= ps_L_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_L_G /= "00000" then
    pixel_G <= ps_L_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_L_B /= "00000" then
    pixel_B <= ps_L_B;
    else
    pixel_B <= "00000";
    end if;

when "010110" =>

    if ps_M_R /= "00000" then
    pixel_R <= ps_M_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_M_G /= "00000" then
    pixel_G <= ps_M_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_M_B /= "00000" then
    pixel_B <= ps_M_B;
    else
    pixel_B <= "00000";
    end if;

when "010111" =>

    if ps_N_R /= "00000" then
    pixel_R <= ps_N_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_N_G /= "00000" then
    pixel_G <= ps_N_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_N_B /= "00000" then
    pixel_B <= ps_N_B;
    else
    pixel_B <= "00000";
    end if;

when "011000" =>

    if ps_O_R /= "00000" then
    pixel_R <= ps_O_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_O_G /= "00000" then
    pixel_G <= ps_O_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_O_B /= "00000" then
    pixel_B <= ps_O_B;
    else
    pixel_B <= "00000";
    end if;

when "011001" =>

    if ps_P_R /= "00000" then
    pixel_R <= ps_P_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_P_G /= "00000" then
    pixel_G <= ps_P_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_P_B /= "00000" then
    pixel_B <= ps_P_B;
    else
    pixel_B <= "00000";
    end if;

when "011010" =>

    if ps_Q_R /= "00000" then
    pixel_R <= ps_Q_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_Q_G /= "00000" then
    pixel_G <= ps_Q_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_Q_B /= "00000" then
    pixel_B <= ps_Q_B;
    else
    pixel_B <= "00000";
    end if;

when "011011" =>

    if ps_R_R /= "00000" then
    pixel_R <= ps_R_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_R_G /= "00000" then
    pixel_G <= ps_R_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_R_B /= "00000" then
    pixel_B <= ps_R_B;
    else
    pixel_B <= "00000";
    end if;

when "011100" =>

    if ps_S_R /= "00000" then
    pixel_R <= ps_S_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_S_G /= "00000" then
    pixel_G <= ps_S_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_S_B /= "00000" then
    pixel_B <= ps_S_B;
    else
    pixel_B <= "00000";
    end if;

when "011101" =>

    if ps_T_R /= "00000" then
    pixel_R <= ps_T_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_T_G /= "00000" then
    pixel_G <= ps_T_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_T_B /= "00000" then
    pixel_B <= ps_T_B;
    else
    pixel_B <= "00000";
    end if;

when "011110" =>

    if ps_U_R /= "00000" then
    pixel_R <= ps_U_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_U_G /= "00000" then
    pixel_G <= ps_U_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_U_B /= "00000" then
    pixel_B <= ps_U_B;
    else
    pixel_B <= "00000";
    end if;

when "011111" =>

    if ps_V_R /= "00000" then
    pixel_R <= ps_V_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_V_G /= "00000" then
    pixel_G <= ps_V_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_V_B /= "00000" then
    pixel_B <= ps_V_B;
    else
    pixel_B <= "00000";
    end if;

when "100000" =>

    if ps_W_R /= "00000" then
    pixel_R <= ps_W_R;
    else
    pixel_R <= "00000";
    end if;

    if ps_W_G /= "00000" then
    pixel_G <= ps_W_G;
    else
    pixel_G <= "00000";
    end if;

    if ps_W_B /= "00000" then
    pixel_B <= ps_W_B;
    else
    pixel_B <= "00000";
    end if;

when "100001" =>

    if ps_X_R /= "00000" then
    pixel_R <= ps_X_R;
    else
```

```
            pixel_R <= "00000";
            end if;

            if ps_X_G /= "00000" then
            pixel_G <= ps_X_G;
            else
            pixel_G <= "00000";
            end if;

            if ps_X_B /= "00000" then
            pixel_B <= ps_X_B;
            else
            pixel_B <= "00000";
            end if;


          when others =>
          pixel_R <= "00000";
          pixel_G <= "00000";
          pixel_B <= "00000";

        end case;

        end if;
        end process;

    end rtl;
```

6.4 lab3_audio.vhd(top)

```
        --
        -- DE2 top-level module that includes the simple audio component
        --
        -- Stephen A. Edwards, Columbia University, sedwards@cs.columbia.edu
        --
        -- From an original by Terasic Technology, Inc.
        -- (DE2_TOP.v, part of the DE2 system board CD supplied by Altera)
        --

        library IEEE;
        use IEEE.std_logic_1164.all;
        use IEEE.numeric_std.all;

        entity lab3_audio is

          port (
            -- Clocks

            CLOCK_27,                                               -- 27 MHz
            CLOCK_50,                                               -- 50 MHz
            EXT_CLOCK  :  in  std_logic;                             --  External
    Clock
```

```
          -- Buttons and switches

          KEY : in std_logic_vector(3 downto 0);              -- Push buttons
          SW : in std_logic_vector(17 downto 0);              -- DPDT switches

          -- LED displays

          HEX0,  HEX1,  HEX2,  HEX3,  HEX4,  HEX5,  HEX6,  HEX7  --
7-segment displays
              : out std_logic_vector(6 downto 0);
          LEDG : out std_logic_vector(8 downto 0);            -- Green LEDs
          LEDR : out std_logic_vector(17 downto 0);           -- Red LEDs

          -- RS-232 interface

          UART_TXD  :  out  std_logic;                                  --  UART
transmitter
          UART_RXD  :  in  std_logic;                                   --  UART
receiver

          -- IRDA interface

     --       IRDA_TXD  :  out  std_logic;                              --  IRDA
Transmitter
          IRDA_RXD  :  in  std_logic;                                   --  IRDA
Receiver

          -- SDRAM

          DRAM_DQ : inout std_logic_vector(15 downto 0); -- Data Bus
          DRAM_ADDR : out std_logic_vector(11 downto 0); -- Address Bus
          DRAM_LDQM,                                                         --
Low-byte Data Mask
          DRAM_UDQM,                                                         --
High-byte Data Mask
          DRAM_WE_N,                                                  --  Write
Enable
          DRAM_CAS_N,                                               -- Column
Address Strobe
          DRAM_RAS_N,                                                   --  Row
Address Strobe
          DRAM_CS_N,                                                   --  Chip
Select
          DRAM_BA_0,                                                   --  Bank
Address 0
          DRAM_BA_1,                                                   --  Bank
Address 0
          DRAM_CLK,                                                 -- Clock
          DRAM_CKE : out std_logic;                            -- Clock Enable
```

```
        -- FLASH

        FL_DQ : inout std_logic_vector(7 downto 0);            -- Data bus
        FL_ADDR : out std_logic_vector(21 downto 0);    -- Address bus
        FL_WE_N,                                                          --  Write
Enable
        FL_RST_N,                                                        -- Reset
        FL_OE_N,                                                          --  Output
Enable
        FL_CE_N : out std_logic;                                  -- Chip Enable

        -- SRAM

        SRAM_DQ : inout std_logic_vector(15 downto 0); -- Data bus 16 Bits
        SRAM_ADDR : out std_logic_vector(17 downto 0); -- Address bus 18
Bits
        SRAM_UB_N,                                                              --
High-byte Data Mask
        SRAM_LB_N,                                                              --
Low-byte Data Mask
        SRAM_WE_N,                                                       --  Write
Enable
        SRAM_CE_N,                                                        --  Chip
Enable
        SRAM_OE_N  :  out  std_logic;                           --  Output
Enable

        -- USB controller

        OTG_DATA : inout std_logic_vector(15 downto 0); -- Data bus
        OTG_ADDR : out std_logic_vector(1 downto 0);        -- Address
        OTG_CS_N,                                                          --  Chip
Select
        OTG_RD_N,                                                        -- Write
        OTG_WR_N,                                                        -- Read
        OTG_RST_N,                                                      -- Reset
        OTG_FSPEED,                              --  USB  Full  Speed,  0  =
Enable, Z = Disable
        OTG_LSPEED : out std_logic;         -- USB Low Speed, 0 = Enable, Z
= Disable
        OTG_INT0,                                                     --  Interrupt
0
        OTG_INT1,                                                     --  Interrupt
1
        OTG_DREQ0,                                                        --  DMA
Request 0
        OTG_DREQ1  :  in  std_logic;                                --  DMA
Request 1
        OTG_DACK0_N,                                                      --  DMA
```

```
Acknowledge 0
        OTG_DACK1_N  :  out  std_logic;                           --  DMA
Acknowledge 1

        -- 16 X 2 LCD Module

        LCD_ON,                         -- Power ON/OFF
        LCD_BLON,                        -- Back Light ON/OFF
        LCD_RW,                         -- Read/Write Select, 0 = Write, 1 =
Read
        LCD_EN,                         -- Enable
        LCD_RS : out std_logic;         -- Command/Data Select, 0 = Command,
1 = Data
        LCD_DATA : inout std_logic_vector(7 downto 0); -- Data bus 8 bits

        -- SD card interface

        SD_DAT,                         -- SD Card Data
        SD_DAT3,                        -- SD Card Data 3
        SD_CMD : inout std_logic;       -- SD Card Command Signal
        SD_CLK : out std_logic;         -- SD Card Clock

        -- USB JTAG link

        TDI,                            -- CPLD -> FPGA (data in)
        TCK,                             -- CPLD -> FPGA (clk)
        TCS : in std_logic;             -- CPLD -> FPGA (CS)
        TDO : out std_logic;            -- FPGA -> CPLD (data out)

        -- I2C bus

        I2C_SDAT : inout std_logic; -- I2C Data
        I2C_SCLK : out std_logic;       -- I2C Clock

        -- PS/2 port

        PS2_DAT,                        -- Data
        PS2_CLK : in std_logic;         -- Clock

        -- VGA output

        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]

        --   Ethernet Interface

        ENET_DATA : inout std_logic_vector(15 downto 0);     -- DATA bus
16Bits
        ENET_CMD,             -- Command/Data Select, 0 = Command, 1 =
Data
        ENET_CS_N,                                                  --
Chip Select
        ENET_WR_N,                                                  --
Write
        ENET_RD_N,                                                  --
Read
        ENET_RST_N,                                                 --
Reset
        ENET_CLK : out std_logic;                              -- Clock
25 MHz
        ENET_INT : in std_logic;                              -- Interrupt

        -- Audio CODEC

        AUD_ADCLRCK : inout std_logic;                           -- ADC
LR Clock
        AUD_ADCDAT : in std_logic;                               -- ADC
Data
        AUD_DACLRCK : inout std_logic;                           -- DAC
LR Clock
        AUD_DACDAT : out std_logic;                              -- DAC
Data
        AUD_BCLK : inout std_logic;                                 --
Bit-Stream Clock
        AUD_XCK : out std_logic;                                -- Chip
Clock

        -- Video Decoder

        TD_DATA : in std_logic_vector(7 downto 0);   -- Data bus 8 bits
        TD_HS,                                           -- H_SYNC
        TD_VS : in std_logic;                         -- V_SYNC
        TD_RESET : out std_logic;                      -- Reset

        -- General-purpose I/O

        GPIO_0,                                                 -- GPIO
Connection 0
```

```
        GPIO_1 : inout std_logic_vector(35 downto 0) -- GPIO Connection 1
        );

    end lab3_audio;

    architecture datapath of lab3_audio is

      component de2_wm8731_audio is
       port (
        clk  :  in  std_logic;                      --      Audio  CODEC  Chip
Clock AUD_XCK
        reset_n : 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 std_logic_vector(15 downto 0);
        sw: in std_logic_vector(17 downto 0);
        pitch       : in unsigned(7 downto 0);
        tone         : in unsigned(1 downto 0)
        -- 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 component;
      component nios_system is
            port (
                  -- 1) global signals:
                     clk : IN STD_LOGIC;
                     reset_n : IN STD_LOGIC;


                     AUD_ADCDAT_to_the_audio : IN STD_LOGIC;
                     AUD_ADCLRCK_from_the_audio         :         OUT
STD_LOGIC;
                     AUD_BCLK_to_and_from_the_audio      :      INOUT
STD_LOGIC;
                     AUD_DACDAT_from_the_audio : OUT STD_LOGIC;
                     AUD_DACLRCK_from_the_audio         :         OUT
STD_LOGIC;

                  -- the_ps2
```

```
                     PS2_Clk_to_the_ps2 : IN STD_LOGIC;
                     PS2_Data_to_the_ps2 : IN STD_LOGIC;

                  -- the_sram
                     SRAM_ADDR_from_the_sram          :          OUT
STD_LOGIC_VECTOR (17 DOWNTO 0);
                     SRAM_CE_N_from_the_sram : OUT STD_LOGIC;
                     SRAM_DQ_to_and_from_the_sram      :      INOUT
STD_LOGIC_VECTOR (15 DOWNTO 0);
                     SRAM_LB_N_from_the_sram : OUT STD_LOGIC;
                     SRAM_OE_N_from_the_sram : OUT STD_LOGIC;
                     SRAM_UB_N_from_the_sram : OUT STD_LOGIC;
                     SRAM_WE_N_from_the_sram : OUT STD_LOGIC;

                   VGA_BLANK_from_the_vga_raster        :        OUT
STD_LOGIC;
                   VGA_B_from_the_vga_raster          :          OUT
STD_LOGIC_VECTOR (9 DOWNTO 0);
                     VGA_CLK_from_the_vga_raster : OUT STD_LOGIC;
                     VGA_G_from_the_vga_raster          :        OUT
STD_LOGIC_VECTOR (9 DOWNTO 0);
                     VGA_HS_from_the_vga_raster : OUT STD_LOGIC;
                     VGA_R_from_the_vga_raster          :        OUT
STD_LOGIC_VECTOR (9 DOWNTO 0);
                     VGA_SYNC_from_the_vga_raster        :       OUT
STD_LOGIC;
                     VGA_VS_from_the_vga_raster : OUT STD_LOGIC
                  );
    end component;

      component de2_i2c_av_config is
      port (
        iCLK : in std_logic;
        iRST_N : in std_logic;
        I2C_SCLK : out std_logic;
        I2C_SDAT : inout std_logic
      );
      end component;

      signal audio_clock : unsigned(1 downto 0) := "00";
      signal audio_request : std_logic;

    begin

      process (CLOCK_50)
      begin
        if rising_edge(CLOCK_50) then
          audio_clock <= audio_clock + "1";
        end if;
      end process;
      AUD_XCK <= audio_clock(1);

      i2c : de2_i2c_av_config port map (
        iCLK         => CLOCK_50,
        iRST_n      => '1',
        I2C_SCLK => I2C_SCLK,
        I2C_SDAT => I2C_SDAT
      );

      V1: de2_wm8731_audio port map (
        clk => audio_clock(1),
        reset_n => '1',
        test_mode => '1',
        audio_request => audio_request,
        data => "0000000000000000",
        sw => sw,
        pitch => "00000000",
        tone => "00"

      );

     V2: nios_system port map (

                     clk => CLOCK_50,
                     reset_n => '1',

                     AUD_ADCDAT_to_the_audio => AUD_ADCDAT,
                     AUD_ADCLRCK_from_the_audio => AUD_ADCLRCK,
                     AUD_BCLK_to_and_from_the_audio => AUD_BCLK,
                     AUD_DACDAT_from_the_audio => AUD_DACDAT,
                     AUD_DACLRCK_from_the_audio => AUD_DACLRCK,

                  -- the_ps2
                     PS2_Clk_to_the_ps2 => PS2_CLK,
                     PS2_Data_to_the_ps2 => PS2_DAT,


                  VGA_BLANK_from_the_vga_raster    => VGA_BLANK,
                  VGA_B_from_the_vga_raster          => VGA_B,
                  VGA_CLK_from_the_vga_raster        => VGA_CLK,
                  VGA_G_from_the_vga_raster          => VGA_G,
                  VGA_HS_from_the_vga_raster         => VGA_HS,
                  VGA_R_from_the_vga_raster          => VGA_R,
                  VGA_SYNC_from_the_vga_raster      => VGA_SYNC,
                  VGA_VS_from_the_vga_raster         => VGA_VS,

                  -- the_sram
```

```
                    SRAM_ADDR_from_the_sram => SRAM_ADDR,
                    SRAM_CE_N_from_the_sram => SRAM_CE_N,
                    SRAM_DQ_to_and_from_the_sram => SRAM_DQ,
                    SRAM_LB_N_from_the_sram => SRAM_LB_N,
                    SRAM_OE_N_from_the_sram => SRAM_OE_N,
                    SRAM_UB_N_from_the_sram => SRAM_UB_N,
                    SRAM_WE_N_from_the_sram => SRAM_WE_N
                );

HEX7        <= "0001001"; -- Leftmost
HEX6        <= "0000110";
HEX5        <= "1000111";
HEX4        <= "1000111";
HEX3        <= "1000000";
HEX2        <= "1000000";
HEX1        <= (others => '1');
HEX0        <= (others => '1');                -- Rightmost
LEDG        <= (others => '0');
LEDR        <= (others => '0');
LCD_ON       <= '1';
LCD_BLON <= '1';
LCD_RW <= '1';
LCD_EN <= '0';
LCD_RS <= '0';

SD_DAT3 <= '1';
SD_CMD <= '1';
SD_CLK <= '1';


UART_TXD <= '0';
DRAM_ADDR <= (others => '0');
DRAM_LDQM <= '0';
DRAM_UDQM <= '0';
DRAM_WE_N <= '1';
DRAM_CAS_N <= '1';
DRAM_RAS_N <= '1';
DRAM_CS_N <= '1';
DRAM_BA_0 <= '0';
DRAM_BA_1 <= '0';
DRAM_CLK <= '0';
DRAM_CKE <= '0';
FL_ADDR <= (others => '0');
FL_WE_N <= '1';
FL_RST_N <= '0';
FL_OE_N <= '1';
FL_CE_N <= '1';
OTG_ADDR <= (others => '0');
OTG_CS_N <= '1';
OTG_RD_N <= '1';
```

```
        OTG_RD_N <= '1';
        OTG_WR_N <= '1';
        OTG_RST_N <= '1';
        OTG_FSPEED <= '1';
        OTG_LSPEED <= '1';
        OTG_DACK0_N <= '1';
        OTG_DACK1_N <= '1';

        TDO <= '0';

        ENET_CMD <= '0';
        ENET_CS_N <= '1';
        ENET_WR_N <= '1';
        ENET_RD_N <= '1';
        ENET_RST_N <= '1';
        ENET_CLK <= '0';

        TD_RESET <= '0';

        -- Set all bidirectional ports to tri-state
        DRAM_DQ      <= (others => 'Z');
        FL_DQ        <= (others => 'Z');
        OTG_DATA     <= (others => 'Z');
        LCD_DATA     <= (others => 'Z');
        SD_DAT       <= 'Z';
        ENET_DATA    <= (others => 'Z');
        GPIO_0       <= (others => 'Z');
        GPIO_1       <= (others => 'Z');

end datapath;
```

6.5 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;
        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);
        sw: in std_logic_vector(17 downto 0);
        pitch     : in unsigned(7 downto 0);
        tone      : in unsigned(1 downto 0);

        -- 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 flag : std_logic;

            signal lrck_divider : unsigned(7 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_out1         : unsigned(15 downto 0);
            signal sin_out2         : unsigned(15 downto 0);
            signal sin_out3         : unsigned(15 downto 0);
            signal sin_counter    : unsigned(7 downto 0);
            signal ff             : std_logic :='0';

           signal divider1 : unsigned(7 downto 0);
           signal one_step : unsigned(11 downto 0):=X"000";
           signal flg           : unsigned(3 downto 0);
           signal clk1          : std_logic :='0';
            signal clk1_div : unsigned(23 downto 0);
      begin


            -- LRCK divider
            -- Audio chip main clock is 18.432MHz / Sample rate 48KHz
            -- Divider is 18.432 MHz / 48KHz /2= 192 (X"C0")
            -- Left justify mode set by I2C controller
            process (clk)
            begin
                if rising_edge(clk) then
                   if pitch = "00001111" then                              --pitch=15
                        divider1 <= X"30";--30
                        if reset_n = '0' then
                            lrck_divider <= (others => '0');
                        elsif lrck_divider = divider1 then --
25MHz/48/0x4A0=440Hz
                            lrck_divider <= X"00";
                        else
                        lrck_divider <= lrck_divider + 1;
                        end if;
                    elsif pitch = "00001110" or flg = X"5" then
--pitch=14
                      divider1 <= X"36"; --36
                        if reset_n = '0' then
                            lrck_divider <= (others => '0');
                        elsif lrck_divider = divider1 then --
25MHz/48/0x4A0=440Hz
                            lrck_divider <= X"00";
                        else
                        lrck_divider <= lrck_divider + 1;
                        end if;
                    elsif pitch = "00001101" or flg = X"4" then
--pitch=13
                      divider1 <= X"3C";--3D-------------------------------
                        if reset_n = '0' then
                            lrck_divider <= (others => '0');
                        elsif lrck_divider = divider1 then --
25MHz/48/0x4A0=440Hz
                            lrck_divider <= X"00";
                        else
                            lrck_divider <= lrck_divider + 1;
                        end if;
                    elsif pitch = "00001100" or flg = X"3" then
--pitch=12
                      divider1 <= X"40"; --41
                        if reset_n = '0' then
                            lrck_divider <= (others => '0');
                        elsif lrck_divider = divider1 then --
25MHz/48/0x4A0=440Hz
                            lrck_divider <= X"00";
                        else
                            lrck_divider <= lrck_divider + 1;
                        end if;
                    elsif pitch = "00001011" or flg = X"2" then
--pitch=11
                      divider1 <= X"48";--49
                        if reset_n = '0' then
```

```
              lrck_divider <= (others => '0');
            elsif lrck_divider = divider1 then --
25MHz/48/0x4A0=440Hz
              lrck_divider <= X"00";
            else
              lrck_divider <= lrck_divider + 1;
            end if;
          elsif pitch = "00001010" or flg = X"1" then
--pitch=10
           divider1 <= X"52"; --52
            if reset_n = '0' then
              lrck_divider <= (others => '0');
            elsif lrck_divider = divider1 then --
25MHz/48/0x4A0=440Hz
              lrck_divider <= X"00";
            else
              lrck_divider <= lrck_divider + 1;
            end if;
          elsif pitch = "00001001" then                --pitch=9
           divider1 <= X"5C";   --56             --57
            if reset_n = '0' then
              lrck_divider <= (others => '0');
            elsif lrck_divider = divider1 then --
25MHz/48/0x4A0=440Hz
              lrck_divider <= X"00";
            else
              lrck_divider <= lrck_divider + 1;
            end if;
          elsif pitch = "00001000" then                --pitch=8
           divider1 <= X"62";--62
            if reset_n = '0' then
              lrck_divider <= (others => '0');
            elsif lrck_divider = divider1 then --
25MHz/48/0x4A0=440Hz
              lrck_divider <= X"00";
            else
              lrck_divider <= lrck_divider + 1;
            end if;
          elsif pitch = "00000111" or flg = X"0" then
--pitch=7
           divider1 <= X"6E"; --6E
            if reset_n = '0' then
              lrck_divider <= (others => '0');
            elsif lrck_divider = divider1 then --
25MHz/48/0x4A0=440Hz
              lrck_divider <= X"00";
            else
              lrck_divider <= lrck_divider + 1;
            end if;
          elsif pitch = "00000110" then                --pitch=6
```

```
                       divider1 <= X"7C";    --7B
                         if reset_n = '0' then
                             lrck_divider <= (others => '0');
                         elsif lrck_divider = divider1 then --
25MHz/48/0x4A0=440Hz
                             lrck_divider <= X"00";
                         else
                             lrck_divider <= lrck_divider + 1;
                         end if;
                     elsif pitch = "00000101" then
                       divider1 <= X"80";    --83
                         if reset_n = '0' then
                             lrck_divider <= (others => '0');
                         elsif lrck_divider = divider1 then --
25MHz/48/0x4A0=440Hz
                             lrck_divider <= X"00";
                         else
                             lrck_divider <= lrck_divider + 1;
                         end if;
                    elsif pitch = "00000100" then
                       divider1 <= X"92"; --93
                         if reset_n = '0' then
                             lrck_divider <= (others => '0');
                         elsif lrck_divider = divider1 then --
25MHz/48/0x4A0=440Hz
                             lrck_divider <= X"00";
                         else
                             lrck_divider <= lrck_divider + 1;
                         end if;
                    elsif pitch = "00000011" then
                       divider1 <= X"A4";      --A5
                         if reset_n = '0' then
                             lrck_divider <= (others => '0');
                         elsif lrck_divider = divider1 then --
25MHz/48/0x4A0=440Hz
                             lrck_divider <= X"00";
                         else
                             lrck_divider <= lrck_divider + 1;
                         end if;
                    elsif pitch = "00000010" then
                       divider1 <= X"B8";    --B0
                         if reset_n = '0' then
                             lrck_divider <= (others => '0');
                         elsif lrck_divider = divider1 then --
25MHz/48/0x4A0=440Hz
                             lrck_divider <= X"00";
                         else
                             lrck_divider <= lrck_divider + 1;
                         end if;
                    elsif pitch = "00000001" then
```

```
            divider1 <= X"C4";            --C4
             if reset_n = '0' then
               lrck_divider <= (others => '0');
             elsif   lrck_divider   =   divider1      then                  --
25MHz/48/0x4A0=440Hz
               lrck_divider <= X"00";
             else
               lrck_divider <= lrck_divider + 1;
             end if;
    --       if pitch = "00000001" then
    --      divider1 <= X"30";
    --         if reset_n = '0' then
    --           lrck_divider <= (others => '0');
    --               elsif   lrck_divider   =   divider1     then                 --
25MHz/48/0x4A0=440Hz
    --           lrck_divider <= X"00";
    --         else
    --         lrck_divider <= lrck_divider + 1;
    --         end if;
    --       elsif pitch = "00000010" then
    --      divider1 <= X"C4";
    --         if reset_n = '0' then
    --           lrck_divider <= (others => '0');
    --               elsif   lrck_divider   =   divider1     then                 --
25MHz/48/0x4A0=440Hz
    --           lrck_divider <= X"00";
    --         else
    --         lrck_divider <= lrck_divider + 1;
    --         end if;
          end if;
        end if;
      end process;

      process (clk)
      begin
        if rising_edge(clk) then
          if reset_n = '0' then
            bclk_divider <= (others => '0');
          elsif bclk_divider = ((divider1+1) srl 4) or set_lrck = '1'    then
            bclk_divider <= X"0";
          else
            bclk_divider <= bclk_divider + 1;
          end if;
        end if;
      end process;


      process (clk1)
      begin
        if rising_edge(clk1) then
        if tone="11" then
--chapter1
            if one_step = X"001"    then
              flg <= X"3";
              one_step <= one_step + 1;
            elsif one_step = X"005"    then
              flg <= X"F";
              one_step <= one_step + 1;
            elsif one_step = X"006"    then
              flg <= X"3";
              one_step <= one_step + 1;
            elsif one_step = X"00A"    then
              flg <= X"F";
              one_step <= one_step + 1;
            elsif one_step = X"00B"    then
              flg <= X"4";
              one_step <= one_step + 1;
            elsif one_step = X"00F"    then
              flg <= X"F";
              one_step <= one_step + 1;
            elsif one_step = X"010"    then
              flg <= X"5";
              one_step <= one_step + 1;
            elsif one_step = X"014"    then
              flg <= X"F";
              one_step <= one_step + 1;
            elsif one_step = X"016"    then
              flg <= X"5";
              one_step <= one_step + 1;
            elsif one_step = X"01A"    then
              flg <= X"F";
              one_step <= one_step + 1;
            elsif one_step = X"01B"    then
              flg <= X"4";
              one_step <= one_step + 1;
            elsif one_step = X"01D"    then
              flg <= X"F";
              one_step <= one_step + 1;
            elsif one_step = X"01E"    then
              flg <= X"3";
              one_step <= one_step + 1;
            elsif one_step = X"023"    then
              flg <= X"F";
              one_step <= one_step + 1;
            elsif one_step = X"024"    then
              flg <= X"2";
              one_step <= one_step + 1;
            elsif one_step = X"028"    then
              flg <= X"F";
              one_step <= one_step + 1;
        elsif one_step = X"029"    then
          flg <= X"1";
          one_step <= one_step + 1;
        elsif one_step = X"02C"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"02D"    then
          flg <= X"1";
          one_step <= one_step + 1;
        elsif one_step = X"032"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"033"    then
          flg <= X"2";
          one_step <= one_step + 1;
        elsif one_step = X"037"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"038"    then
          flg <= X"3";
          one_step <= one_step + 1;
        elsif one_step = X"03C"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"03E"    then
          flg <= X"3";
          one_step <= one_step + 1;
        elsif one_step = X"042"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"043"    then
          flg <= X"2";
          one_step <= one_step + 1;
        elsif one_step = X"045"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"046"    then
          flg <= X"2";
          one_step <= one_step + 1;
        elsif one_step = X"04D"    then
          flg <= X"F";
          one_step <= one_step + 1;
--chapter2
        elsif one_step = X"04E"    then
          flg <= X"3";
          one_step <= one_step + 1;
        elsif one_step = X"052"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"053"    then
```

```
   flg <= X"3";
   one_step <= one_step + 1;
elsif one_step = X"057"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"058"    then
   flg <= X"4";
   one_step <= one_step + 1;
elsif one_step = X"05C"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"05E"    then
   flg <= X"5";
   one_step <= one_step + 1;
elsif one_step = X"062"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"063"    then
   flg <= X"5";
   one_step <= one_step + 1;
elsif one_step = X"067"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"068"    then
   flg <= X"4";
   one_step <= one_step + 1;
elsif one_step = X"06C"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"06D"    then
   flg <= X"3";
   one_step <= one_step + 1;
elsif one_step = X"071"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"072"    then
   flg <= X"2";
   one_step <= one_step + 1;
elsif one_step = X"076"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"077"    then
   flg <= X"1";
   one_step <= one_step + 1;
elsif one_step = X"07B"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"07C"    then
   flg <= X"1";
   one_step <= one_step + 1;
```

```
        elsif one_step = X"080"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"081"    then
          flg <= X"2";
          one_step <= one_step + 1;
        elsif one_step = X"085"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"086"    then
          flg <= X"3";
          one_step <= one_step + 1;
        elsif one_step = X"08A"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"08C"    then
          flg <= X"3";
          one_step <= one_step + 1;
        elsif one_step = X"090"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"091"    then
          flg <= X"2";
          one_step <= one_step + 1;
        elsif one_step = X"093"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"094"    then
          flg <= X"2";
          one_step <= one_step + 1;
        elsif one_step = X"09B"    then
          flg <= X"F";
          one_step <= one_step + 1;
--chapter3
        elsif one_step = X"09C"    then
          flg <= X"2";
          one_step <= one_step + 1;
        elsif one_step = X"0A0"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"0A1"    then
          flg <= X"2";
          one_step <= one_step + 1;
        elsif one_step = X"0A5"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"0A6"    then
          flg <= X"3";
          one_step <= one_step + 1;
        elsif one_step = X"0AA"    then
```

```
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"0AB"    then
   flg <= X"1";
   one_step <= one_step + 1;
elsif one_step = X"0AF"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"0B0"    then
   flg <= X"2";
   one_step <= one_step + 1;
elsif one_step = X"0B4"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"0B5"    then
   flg <= X"3";
   one_step <= one_step + 1;
elsif one_step = X"0B7"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"0B8"    then
   flg <= X"4";
   one_step <= one_step + 1;
elsif one_step = X"0BA"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"0BB"    then
   flg <= X"3";
   one_step <= one_step + 1;
elsif one_step = X"0BF"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"0C0"    then
   flg <= X"1";
   one_step <= one_step + 1;
elsif one_step = X"0C4"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"0C5"    then
   flg <= X"2";
   one_step <= one_step + 1;
elsif one_step = X"0C9"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"0CA"    then
   flg <= X"3";
   one_step <= one_step + 1;
elsif one_step = X"0CC"    then
   flg <= X"F";
   one_step <= one_step + 1;
```

```
        elsif one_step = X"0CD"    then
          flg <= X"4";
          one_step <= one_step + 1;
        elsif one_step = X"0CF"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"0D0"    then
          flg <= X"3";
          one_step <= one_step + 1;
        elsif one_step = X"0D4"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"0D5"    then
          flg <= X"2";
          one_step <= one_step + 1;
        elsif one_step = X"0D9"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"0DA"    then
          flg <= X"1";
          one_step <= one_step + 1;
        elsif one_step = X"0DE"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"0DF"    then
          flg <= X"2";
          one_step <= one_step + 1;
        elsif one_step = X"0E3"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"0E4"    then
          flg <= X"0";
          one_step <= one_step + 1;
        elsif one_step = X"0EC"    then
          flg <= X"F";
          one_step <= one_step + 1;
--chapter4
        elsif one_step = X"0ED"    then
          flg <= X"3";
          one_step <= one_step + 1;
        elsif one_step = X"0F1"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"0F2"    then
          flg <= X"3";
          one_step <= one_step + 1;
        elsif one_step = X"0F6"    then
          flg <= X"F";
          one_step <= one_step + 1;
        elsif one_step = X"0F7"    then
```

```
   flg <= X"4";
   one_step <= one_step + 1;
elsif one_step = X"0FB"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"0FC"    then
   flg <= X"5";
   one_step <= one_step + 1;
elsif one_step = X"100"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"102"    then
   flg <= X"5";
   one_step <= one_step + 1;
elsif one_step = X"106"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"107"    then
   flg <= X"4";
   one_step <= one_step + 1;
elsif one_step = X"109"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"10A"    then
   flg <= X"3";
   one_step <= one_step + 1;
elsif one_step = X"10E"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"10F"    then
   flg <= X"2";
   one_step <= one_step + 1;
elsif one_step = X"113"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"114"    then
   flg <= X"1";
   one_step <= one_step + 1;
elsif one_step = X"118"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"119"    then
   flg <= X"1";
   one_step <= one_step + 1;
elsif one_step = X"11D"    then
   flg <= X"F";
   one_step <= one_step + 1;
elsif one_step = X"11E"    then
   flg <= X"2";
   one_step <= one_step + 1;
```

```
                elsif one_step = X"122"    then
                   flg <= X"F";
                   one_step <= one_step + 1;
                elsif one_step = X"123"    then
                   flg <= X"3";
                   one_step <= one_step + 1;
                elsif one_step = X"127"    then
                   flg <= X"F";
                   one_step <= one_step + 1;
                elsif one_step = X"129"    then
                   flg <= X"2";
                   one_step <= one_step + 1;
                elsif one_step = X"12D"    then
                   flg <= X"F";
                   one_step <= one_step + 1;
                elsif one_step = X"12E"    then
                   flg <= X"1";
                   one_step <= one_step + 1;
                elsif one_step = X"130"    then
                   flg <= X"F";
                   one_step <= one_step + 1;
                elsif one_step = X"131"    then
                   flg <= X"1";
                   one_step <= one_step + 1;
                elsif one_step = X"139"    then
                   flg <= X"F";
                   one_step <= one_step + 1;




                elsif one_step = X"13F" then
                   flg <= X"F";
                   one_step <= X"000";
                else
                   one_step <= one_step + 1;
                end if;
            else
                flg <=X"E";
              end if;
        end if;
    end process;

process (clk)
    begin
        if rising_edge(clk) then
            if clk1_div = X"2FFFFF"    then
```

```
                        clk1_div <= X"000000";
                        clk1 <= not clk1;
                    else
                      clk1_div <= clk1_div + 1;
                    end if;
                end if;
            end process;



            set_lrck <= '1' when lrck_divider = divider1 else '0';




            process (clk)
            begin
                if rising_edge(clk) 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) = (divider1+1) srl 5 else '0';
--5      (6)
            clr_bclk <= '1' when bclk_divider(3 downto 0) = (divider1+1) srl 4 else '0';
--11       (12)

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

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

```
         if reset_n = '0' then
            shift_out <= (others => '0');
         elsif set_lrck = '1' then
--            if sw(0) = '0' then
--               shift_out <= sin_out1;
--         elsif sw(1) = '0' then
--            shift_out <= sin_out2;
--        elsif sw(2) = '0' then
--            shift_out <= sin_out3;
          if tone = "11"    then                               --background
               shift_out <= sin_out1;
             --ff <= '1';
          elsif tone = "01"    then                            --scream
               shift_out <= sin_out1;
               --ff <= '0';
           elsif tone = "00" or flg = X"F" then              --non
               shift_out <= data;
            else
               shift_out <= data;     --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(clk)
      begin
         if rising_edge(clk) then
            if reset_n = '0' then
                  sin_counter <= (others => '0');
            elsif lrck_lat = '1' and lrck = '0'    then
               if sin_counter = "10001111" then          --48
                  sin_counter <= "00000000";
               else
                  sin_counter <= sin_counter + 1;
               end if;
            end if;
         end if;
      end process;
```

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

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

with sin_counter select sin_out1 <=
    X"0000" when "00000000",
    X"10b4" when "00000001",
    X"2120" when "00000010",
    X"30fb" when "00000011",
    X"3fff" when "00000100",
    X"4deb" when "00000101",
    X"5a81" when "00000110",
    X"658b" when "00000111",
    X"6ed9" when "00001000",
    X"7640" when "00001001",
    X"7ba2" when "00001010",
    X"7ee6" when "00001011",
    X"7fff" when "00001100",
    X"7ee6" when "00001101",
    X"7ba2" when "00001110",
    X"7640" when "00001111",
    X"6ed9" when "00010000",
    X"658b" when "00010001",
    X"5a81" when "00010010",
    X"4deb" when "00010011",
    X"3fff" when "00010100",
    X"30fb" when "00010101",
    X"2120" when "00010110",
    X"10b4" when "00010111",
    X"0000" when "00011000",
    X"ef4b" when "00011001",
    X"dee0" when "00011010",
    X"cf05" when "00011011",
X"c001" when "00011100",
X"b215" when "00011101",
X"a57e" when "00011110",
X"9a74" when "00011111",
X"9127" when "00100000",
X"89bf" when "00100001",
X"845d" when "00100010",
X"8119" when "00100011",
X"8000" when "00100100",
X"8119" when "00100101",
X"845d" when "00100110",
X"89bf" when "00100111",
X"9127" when "00101000",
X"9a74" when "00101001",
X"a57e" when "00101010",
X"b215" when "00101011",
X"c000" when "00101100",
X"cf05" when "00101101",
X"dee0" when "00101110",
X"ef4b" when "00101111",


X"0000" when "00110000",
X"10b4" when "00110001",
X"2120" when "00110010",
X"30fb" when "00110011",
X"3fff" when "00110100",
X"4deb" when "00110101",
X"5a81" when "00110110",
X"658b" when "00110111",
X"6ed9" when "00111000",
X"7640" when "00111001",
X"7ba2" when "00111010",
X"7ee6" when "00111011",
X"7fff" when "00111100",
X"7ee6" when "00111101",
X"7ba2" when "00111110",
X"7640" when "00111111",
X"6ed9" when "01000000",
X"658b" when "01000001",
X"5a81" when "01000010",
X"4deb" when "01000011",
X"3fff" when "01000100",
X"30fb" when "01000101",
X"2120" when "01000110",
X"10b4" when "01000111",
X"0000" when "01001000",
X"ef4b" when "01001001",
X"dee0" when "01001010",
X"cf05" when "01001011",
X"c001" when "01001100",
X"b215" when "01001101",
X"a57e" when "01001110",
X"9a74" when "01001111",
X"9127" when "01010000",
X"89bf" when "01010001",
X"845d" when "01010010",
X"8119" when "01010011",
X"8000" when "01010100",
X"8119" when "01010101",
X"845d" when "01010110",
X"89bf" when "01010111",
X"9127" when "01011000",
X"9a74" when "01011001",
X"a57e" when "01011010",
X"b215" when "01011011",
X"c000" when "01011100",
X"cf05" when "01011101",
X"dee0" when "01011110",
X"ef4b" when "01011111",


X"0000" when "01100000",
X"10b4" when "01100001",
X"2120" when "01100010",
X"30fb" when "01100011",
X"3fff" when "01100100",
X"4deb" when "01100101",
X"5a81" when "01100110",
X"658b" when "01100111",
X"6ed9" when "01101000",
X"7640" when "01101001",
X"7ba2" when "01101010",
X"7ee6" when "01101011",
X"7fff" when "01101100",
X"7ee6" when "01101101",
X"7ba2" when "01101110",
X"7640" when "01101111",
X"6ed9" when "01110000",
X"658b" when "01110001",
X"5a81" when "01110010",
X"4deb" when "01110011",
X"3fff" when "01110100",
X"30fb" when "01110101",
X"2120" when "01110110",
X"10b4" when "01110111",
X"0000" when "01111000",
X"ef4b" when "01111001",
X"dee0" when "01111010",
X"cf05" when "01111011",
                X"c001" when "01111100",
                X"b215" when "01111101",
                X"a57e" when "01111110",
                X"9a74" when "01111111",
                X"9127" when "10000000",
                X"89bf" when "10000001",
                X"845d" when "10000010",
                X"8119" when "10000011",
                X"8000" when "10000100",
                X"8119" when "10000101",
                X"845d" when "10000110",
                X"89bf" when "10000111",
                X"9127" when "10001000",
                X"9a74" when "10001001",
                X"a57e" when "10001010",
                X"b215" when "10001011",
                X"c000" when "10001100",
                X"cf05" when "10001101",
                X"dee0" when "10001110",
                X"ef4b" when "10001111",


                X"0000" when others;

        end architecture;
```

6.6 hello_world.c

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

#define IOWR_VGA_DATA(base, offset, data) \
        IOWR_32DIRECT(base, (offset) * 4, data)

#define speed 1

#define origin_x 18
#define origin_y 36

#define op 4

#define type_sred 13
#define type_lred 15
#define type_sgreen 9
#define type_lgreen 11
#define type_sblue 1
#define type_lblue 3
#define type_sorange 5
#define type_lorange 7
```

```
#define type_explode 21
#define type_start 24
#define type_pause 25
#define type_next 26
#define type_back 27
#define type_begin 41
#define type_fail 42
#define type_win 43
#define type_blank 45

#define type_up 17
#define type_down 18
#define type_left 19
#define type_right 20

#define type_lv 29
#define type_hit 30

#define type_0 31
#define type_1 32
#define type_2 33
#define type_3 34
#define type_4 35
#define type_5 36
#define type_6 37
#define type_7 38
#define type_8 39
#define type_9 40

#define snum1 1
#define snum2 9
#define snum3 1
#define snum4 13
#define snum5 13
#define snum6 17
#define snum7 14
#define snum8 15
#define snum9 14
#define snum10 9

#define tap_left1 1
#define tap_left2 1
#define tap_left3 2
#define tap_left4 3
#define tap_left5 3
#define tap_left6 4
#define tap_left7 4
#define tap_left8 3
#define tap_left9 3
#define tap_left10 2
```

```
#define x_offset 18

//=========================structure def=========================
struct snapper{
    int x;
    int y;
    int num;
    int type;
    int pointed;
};

struct bullet{
      int x;
      int y;
      int num;
      int type;
      int init_x;
      int init_y;
};

struct pointer{
          int x;
          int y;
  };

//=============================================================
====

struct snapper s1[1]={{162,180,1,type_sred,0}};

struct snapper s2[9]={{162,36,1,type_sred,0},
                                        {306,36,2,type_sred,0},
                                        {90,108,3,type_sred,0},
                                        {234,108,4,type_sred,0},
                                        {162,252,5,type_sred,0},
                                        {18,324,6,type_sred,0},
                                        {234,324,7,type_sred,0},
                                        {90,396,8,type_sred,0},
                                        {306,396,9,type_sred,0}};

struct snapper s3[1]={{162,180,1,type_sgreen,0}};

struct snapper s4[13]={{18,36,1,type_sred,0},
                                          {306,36,2,type_sred,0},
                                          {18,108,3,type_sgreen,0},
                                          {90,108,4,type_sgreen,0},
                                          {162,108,5,type_sgreen,0},
                                          {234,108,6,type_sgreen,0},
                                          {306,108,7,type_sgreen,0},
                                          {18,180,8,type_sred,0},
                                          {90,180,9,type_sred,0},
                                          {162,180,10,type_sorange,0},
                                          {234,180,11,type_sred,0},
                                          {306,180,12,type_sred,0},
                                          {162,252,13,type_sred,0}};

struct snapper s5[13]={{90,36,1,type_sred,0},
                                          {162,36,2,type_sred,0},
                                          {234,36,3,type_sred,0},
                                          {18,108,4,type_sred,0},
                                          {90,108,5,type_sred,0},
                                          {234,108,6,type_sred,0},
                                          {306,108,7,type_sred,0},
                                          {18,180,8,type_sblue,0},
                                          {90,180,9,type_sred,0},
                                          {234,180,10,type_sred,0},
                                          {306,180,11,type_sblue,0},
                                          {90,252,12,type_sred,0},
                                          {234,252,13,type_sred,0}};

struct snapper s6[17]={{18,36,1,type_sred,0},
                                          {90,36,2,type_sgreen,0},
                                          {162,36,3,type_sgreen,0},
                                          {234,36,4,type_sred,0},
                                          {306,36,5,type_sred,0},
                                          {162,108,6,type_sgreen,0},
                                          {234,108,7,type_sgreen,0},
                                          {306,108,8,type_sgreen,0},
                                          {18,180,9,type_sred,0},
                                          {162,180,10,type_sred,0},
                                          {234,180,11,type_sred,0},
                                          {306,180,12,type_sgreen,0},
                                          {162,252,13,type_sgreen,0},
                                          {234,252,14,type_sgreen,0},
                                          {90,324,15,type_sblue,0},
                                          {162,324,16,type_sgreen,0},
                                          {234,324,17,type_sgreen,0}};

struct snapper s7[14]={{18,36,1,type_sgreen,0},
                                          {90,36,2,type_sred,0},
                                          {234,36,3,type_sred,0},
                                          {306,36,4,type_sgreen,0},
                                          {90,108,5,type_sgreen,0},
                                          {234,108,6,type_sgreen,0},
                                          {90,180,7,type_sred,0},
                                          {162,180,8,type_sgreen,0},
                                          {234,180,9,type_sred,0},
                                          {90,252,10,type_sblue,0},
                                          {162,252,11,type_sred,0},
                                          {234,252,12,type_sblue,0},
                                          {18,324,13,type_sred,0},
                                          {306,324,14,type_sred,0}};

struct snapper s8[15]={{162,36,1,type_sred,0},
                                          {18,108,2,type_sgreen,0},
                                          {162,108,3,type_sred,0},
                                          {306,108,4,type_sgreen,0},
                                          {18,252,5,type_sgreen,0},
                                          {90,252,6,type_sorange,0},
                                          {162,252,7,type_sred,0},
                                          {234,252,8,type_sorange,0},
                                          {306,252,9,type_sgreen,0},
                                          {18,324,10,type_sgreen,0},
                                          {90,324,11,type_sgreen,0},
                                          {162,324,121,type_sgreen,0},
                                          {234,324,13,type_sgreen,0},
                                          {306,324,14,type_sgreen,0},
                                          {162,396,15,type_sgreen,0}};

struct snapper s9[14]={{18,36,1,type_sgreen,0},
                                          {162,36,2,type_sred,0},
                                          {306,36,3,type_sgreen,0},
                                          {90,108,4,type_sred,0},
                                          {162,108,5,type_sgreen,0},
                                          {234,108,6,type_sred,0},
                                          {18,252,7,type_sred,0},
                                          {162,252,8,type_sblue,0},
                                          {306,252,9,type_sred,0},
                                          {18,324,10,type_sgreen,0},
                                          {306,324,11,type_sgreen,0},
                                          {90,396,12,type_sred,0},
                                          {162,396,13,type_sorange,0},
                                          {234,396,14,type_sred,0}};

struct snapper s10[9]={{162,36,1,type_sgreen,0},
                                          {18,108,2,type_sred,0},
                                          {90,108,3,type_sgreen,0},
                                          {306,108,4,type_sred,0},
                                          {18,180,5,type_sred,0},
                                          {90,180,6,type_sgreen,0},
                                          {162,180,7,type_sred,0},
                                          {306,180,8,type_sred,0},
                                          {90,252,9,type_sred,0}};

struct snapper *pointed;
struct bullet *upcnt;
struct bullet *downcnt;
struct bullet *leftcnt;
struct bullet *rightcnt;
```

```
int bcnt;
int snum;
int level;
int tap_left;
int upstop=0;
int downstop=0;
int leftstop=0;
int rightstop=0;
int flag2=0;

//assemble the data and send, for snappers
void swd(struct snapper *s){
        int wd;
        int ad;
        wd=(s->num<<20)+(s->y<<10)+s->x;
        ad=(s->type<<5)+0;
        IOWR_VGA_DATA(VGA_RASTER_BASE, ad, wd);
}

//assemble the data and send, for bullets
void bwd(struct bullet *b){
        int wd;
        int ad;
        wd=(b->num<<20)+(b->y<<10)+b->x;
        ad=(b->type<<5)+1;
        IOWR_VGA_DATA(VGA_RASTER_BASE, ad, wd);
}


void show_level(){
        int wd;
        int ad;
        int type=type_lv;
        int num_level=1;
        int num_nlevel=2;
        int type_nlevel;
        int lx=400;
        int ly=10;
        int lnx=435;
        int lny=10;

        switch(level){
                case 0:
                      type_nlevel=type_0;
                      break;
                case 1:
                      type_nlevel=type_1;
                      break;
                case 2:
```

```
            type_nlevel=type_2;
            break;
        case 3:
            type_nlevel=type_3;
            break;
        case 4:
            type_nlevel=type_4;
            break;
        case 5:
            type_nlevel=type_5;
            break;
        case 6:
            type_nlevel=type_6;
            break;
        case 7:
            type_nlevel=type_7;
            break;
        case 8:
            type_nlevel=type_8;
            break;
        case 9:
            type_nlevel=type_9;
            break;
    }
    wd=(num_level<<20)+(ly<<10)+lx;
    ad=type<<5;
    IOWR_VGA_DATA(VGA_RASTER_BASE, ad, wd);
    wd=(num_nlevel<<20)+(lny<<10)+lnx;
    ad=type_nlevel<<5;
    IOWR_VGA_DATA(VGA_RASTER_BASE, ad, wd);
}

void show_hit(){
    int wd;
    int ad;
    int type=type_hit;
    int num_hit=3;
    int num_nhit=4;
    int type_nhit;
    int hx=510;
    int hy=10;
    int hnx=545;
    int hny=10;

    switch(tap_left){
        case 0:
            type_nhit=type_0;
            break;
        case 1:
            type_nhit=type_1;
            break;
        case 2:
            type_nhit=type_2;
            break;
        case 3:
            type_nhit=type_3;
            break;
        case 4:
            type_nhit=type_4;
            break;
        case 5:
            type_nhit=type_5;
            break;
        case 6:
            type_nhit=type_6;
            break;
        case 7:
            type_nhit=type_7;
            break;
        case 8:
            type_nhit=type_8;
            break;
        case 9:
            type_nhit=type_9;
            break;
    }
    wd=(num_hit<<20)+(hy<<10)+hx;
    ad=type<<5;
    IOWR_VGA_DATA(VGA_RASTER_BASE, ad, wd);
    wd=(num_nhit<<20)+(hny<<10)+hnx;
    ad=type_nhit<<5;
    IOWR_VGA_DATA(VGA_RASTER_BASE, ad, wd);
}

void show_pointer(struct pointer *p){
    int wd;
    int ad=2;
    wd=((p->y+28)<<10)+(p->x+20);
    IOWR_VGA_DATA(VGA_RASTER_BASE, ad, wd);
}

int begin_x=200;
int begin_y=200;
int fail_x=300;
int fail_y=300;
int win_x=400;
int win_y=400;

void show_label(int type){
    int wd;
```

```
	int ad;
	int x;
	int y;
	switch(type){
		case type_begin:
			x=begin_x;
			y=begin_y;

		case type_fail:
			x=fail_x;
			y=fail_y;
		case type_win:
			x=win_x;
			y=win_y;
	}
	wd=(y<<10)+x;
	ad=(type<<5)+4;
	IOWR_VGA_DATA(VGA_RASTER_BASE, ad, wd);
}

void show_press_enter(){
	int wd=0;
	int ad=5;
	IOWR_VGA_DATA(VGA_RASTER_BASE, ad, wd);
}

void show_button_explain(){
	int wd=0;
	int ad=6;
	IOWR_VGA_DATA(VGA_RASTER_BASE, ad, wd);
}


void show_buttons(){
	int wd;
	int ad;
	int x=435;
	int y=50;
	int type=type_start;
	int num=1;
	wd=(num<<20)+(y<<10)+x;
	ad=(type<<5)+0;
	IOWR_VGA_DATA(VGA_RASTER_BASE, ad, wd);
	num++;
	x=x+45;
	type=type_pause;
	wd=(num<<20)+(y<<10)+x;
	ad=(type<<5)+0;
	IOWR_VGA_DATA(VGA_RASTER_BASE, ad, wd);
	num++;
```

```
        x=x+45;
        type=type_next;
        wd=(num<<20)+(y<<10)+x;
        ad=(type<<5)+0;
        IOWR_VGA_DATA(VGA_RASTER_BASE, ad, wd);
        num++;
        x=x+45;
        type=type_back;
        wd=(num<<20)+(y<<10)+x;
        ad=(type<<5)+0;
        IOWR_VGA_DATA(VGA_RASTER_BASE, ad, wd);
}

void expand(struct snapper *s, struct pointer *p){
                int i;
                for(i=0; i<snum; i++){

if((p->x==(s+i)->x)&&(p->y==(s+i)->y)&&((s+i)->pointed==0)&&(s+i)->type!=type_explode){
                      printf("s.type was %d\n",(s+i)->type);
                      (s+i)->type=(s+i)->type+2;
                      (s+i)->pointed=1;

                      printf("s%d expanded, s.type=%d\n", i+1, (s+i)->type);
                      swd(s+i);
                      pointed=s+i;
                      break;
                      }
                }
        }
void contract(struct snapper *p){
        if(p->pointed==1){
                p->type=p->type-2;
                p->pointed=0;
                printf("p.type=%d\n",p->type);
                swd(p);
                printf("snapper is contracted.\n");
        }
}


void clear_screen(struct snapper *s){

        int num;
        int i;
        switch(level){
                case 0:
                        snum=snum1;

                        break;
                case 1:
                        snum=snum2;

                        break;
                case 2:
                        snum=snum3;

                        break;
                case 3:
                        snum=snum4;

                        break;
                case 4:
                        snum=snum5;

                        break;
                case 5:
                        snum=snum6;

                        break;
                case 6:
                        snum=snum7;

                        break;
                case 7:
                        snum=snum8;

                        break;
                case 8:
                        snum=snum9;

                        break;
                case 9:
                        snum=snum10;

                        break;
        }
        //printf("snum=%d",snum);
        for(i=0; i<num; i++){

              s->type=type_blank;
              //printf("s.num=%d    ", s->num);
              //printf("cleaning s.type=%d    ", s->type);
              swd(s);
              s++;
          }
}


void print_btype(struct bullet *b){
        switch(b->type){
                case type_up: printf("(up)");
                        break;
                case type_down: printf("(down)");
                        break;
                case type_left: printf("(left)");
                        break;
                case type_right: printf("(right)");
                        break;
        }
}


void explode_init(struct snapper *s, struct bullet *b1, struct bullet *b2, struct bullet
*b3, struct bullet *b4){

                        b1->init_x=s->x+12;
                        b1->init_y=s->y-12;
                        b1->x=s->x+12;
                        b1->y=s->y-12;
                        b1->type=type_up;
                        bcnt++;
                        b1->num=bcnt;

                        printf("generate b%d.x=%d, ", b1->num-snum-1, b1->x);
                        printf("b%d.y=%d, ", b1->num-snum-1, b1->y);
                        print_btype(b1);
                        printf("\n");


                        b2->init_x=s->x+12;
                        b2->init_y=s->y+38;
                        b2->x=s->x+12;
                        b2->y=s->y+38;
                        b2->type=type_down;
                        bcnt++;
                        b2->num=bcnt;

                        printf("generate b%d.x=%d, ", b2->num-snum-1, b2->x);
                        printf("b%d.y=%d, ", b2->num-snum-1, b2->y);
                        print_btype(b2);
                        printf("\n");

                        b3->init_x=s->x+12;
                        b3->init_y=s->y+15;
                        b3->x=s->x-12;
                        b3->y=s->y+15;
                        b3->type=type_left;
                        bcnt++;
                        b3->num=bcnt;
```

```
                printf("generate b%d.x=%d, ", b3->num-snum-1, b3->x);
                printf("b%d.y=%d, ", b3->num-snum-1, b3->y);
                print_btype(b3);
                printf("\n");

                b4->init_x=s->x+34;
                b4->init_y=s->y+15;
                b4->x=s->x+34;
                b4->y=s->y+15;
                b4->type=type_right;
                bcnt++;
                b4->num=bcnt;

                printf("generate b%d.x=%d, ", b4->num-snum-1, b4->x);
                printf("b%d.y=%d, ", b4->num-snum-1, b4->y);
                print_btype(b4);
                printf("\n");
                //printf("four bullets initial position received\n");

}

void hit(struct snapper *s, struct bullet *b){
    if(s->type!=type_explode&&b->type!=type_blank&&s->type!=type_blank){
    switch(b->type){
        case type_up: // up bullet

            if((s->y)>b->y&&s->x==((b->x)-12)&&b->init_y>s->y&&s!=pointed){

                if(s->type!=type_sred&&s->type<16){
                    s->type=s->type+4;
                    printf("changed type to %d\n", s->type);
                    swd(s);
                    b->type=type_blank;
                    printf("kill b%d",b->num-snum-1);
                    print_btype(b);
                    printf("\n");
                    bwd(b);
                }
                else if(s->type==type_sred)// red explodes
                    {
                     b->type=type_blank;
                     bwd(b);
                     s->type=type_explode;
                     swd(s);
                     //for(i=0; i<1000; i++);

//s->type=type_btype_bnnlanktype_bnnlanktype_bnnlanktype_bnnlanktype_bnnlankn
nlank;
                     //swd(s);
                     printf("explode by b%d ",b->num-snum-1);
```

```
         print_btype(b);
         printf("\n");
         upcnt++;
         downcnt++;
         leftcnt++;
         rightcnt++;
         explode_init(s,upcnt,downcnt,leftcnt,rightcnt);
        }
  }
  break;
case type_down: //down bullet
 if(s->y<b->y&&s->x==((b->x)-12)&&b->init_y<s->y&&s!=pointed){
     if(s->type!=type_sred&&s->type<16){
        s->type=s->type+4;
        swd(s);
        b->type=type_blank;
        bwd(b);
     }
     else if(s->type==type_sred)// red explodes
        {
         b->type=type_blank;
         bwd(b);
         s->type=type_explode;
         swd(s);
         upcnt++;
         downcnt++;
         leftcnt++;
         rightcnt++;
         explode_init(s,upcnt,downcnt,leftcnt,rightcnt);
        }
  }

  break;
case type_left:    //left bullet
   if(s->y==b->y-15&&s->x>(b->x)&&b->init_x>s->x&&s!=pointed){

     printf("s%d hit by b%d ", s->num, b->num-snum-1);
     print_btype(b);

     if(s->type!=type_sred&&s->type<16){
        s->type=s->type+4;
        printf("changed type to %d\n", s->type);

        swd(s);
        b->type=type_blank;
        printf("kill b%d",b->num-snum-1);
        print_btype(b);
        printf("\n");
        bwd(b);
     }
```

```
                    else if(s->type==type_sred)// red explodes
                          {
                           b->type=type_blank;
                           bwd(b);
                           s->type=type_explode;
                           swd(s);
                           upcnt++;
                           downcnt++;
                           leftcnt++;
                           rightcnt++;
                           explode_init(s,upcnt,downcnt,leftcnt,rightcnt);

                          }
                }
                break;
            case type_right:    //right bullet

if(s->y==(b->y-15)&&s->x==(b->x+10)&&b->init_x<s->x&&s!=pointed){

                    if(s->type!=type_sred&&s->type<16){
                          s->type=s->type+4;
                          swd(s);
                          b->type=type_blank;
                          bwd(b);
                    }
                    else if(s->type==type_sred)// red explodes
                          {
                           b->type=type_blank;
                           s->type=type_explode;
                           swd(s);
                           bwd(b);

                           upcnt++;
                           downcnt++;
                           leftcnt++;
                           rightcnt++;
                           explode_init(s,upcnt,downcnt,leftcnt,rightcnt);

                          }
                }
                break;

        }
      }
}


int stage_clear(struct snapper *s){
        int i;
        for(i=0; i<snum; i++){
            if((s+i)->type!=type_explode)
                return 0;
        }
        return 1;
}

void game_over(struct snapper *s){
    clear_screen(s);
  //***********************add game over picture here
}

void level_initialize(struct snapper *s){
      int i;
      struct snapper *sl;
      switch(level){

            case 0:
                    sl=s1;
                    snum=snum1;
                    tap_left=tap_left1;
                    break;
            case 1:
                    sl=s2;
                    snum=snum2;
                    tap_left=tap_left2;
                    break;

            case 2:
                    sl=s3;
                    snum=snum3;
                    tap_left=tap_left3;
                    break;
            case 3:
                    sl=s4;
                    snum=snum4;
                    tap_left=tap_left4;
                    break;
            case 4:
                    sl=s5;
                    snum=snum5;
                    tap_left=tap_left5;
                    break;
            case 5:
                    sl=s6;
                    snum=snum6;
                    tap_left=tap_left6;
                    break;
            case 6:
                    sl=s7;
                    snum=snum7;
```

```
               tap_left=tap_left7;
               break;

        case 7:
               sl=s8;
               snum=snum8;
               tap_left=tap_left8;
               break;
        case 8:
               sl=s9;
               snum=snum9;
               tap_left=tap_left9;
               break;
        case 9:
               sl=s10;
               snum=snum10;
               tap_left=tap_left10;
               break;

     }
     //printf("snum=%d\n", snum);
    for(i=0; i<snum ;i++){

        s->x=(sl+i)->x;
        s->y=(sl+i)->y;
        s->num=(sl+i)->num+op;
        //printf("%d ",s->num);
        s->type=(sl+i)->type;
        s->pointed=0;
        swd(s);
        s++;

    }
    bcnt=op+snum+1;

}


//*******************MAIN***************************
int main(){
  /*
  int upstop=0;
  int downstop=0;
  int leftstop=0;
  int rightstop=0;
  */
  int cc=0;
  int i=0;
  int flg=1;
  int tone=3;
```

```
    int delay=1;
    int release=0;
    printf("Hello :D\n");
    unsigned char code,read;
    IOWR_16DIRECT(AUDIO_BASE,0,(tone*256));

    //***********struct snapper(x,y,num,type,pointed)************

    //***********struct bullet(x,y,num,type, init_x, init_y)********************

  //=================snappers initialization
    struct snapper s[17]={{-1,-1,-1,-1,0},
                            {-1,-1,-1,-1,0},
                            {-1,-1,-1,-1,0},
                            {-1,-1,-1,-1,0},
                            {-1,-1,-1,-1,0},
                            {-1,-1,-1,-1,0},
                            {-1,-1,-1,-1,0},
                            {-1,-1,-1,-1,0},
                            {-1,-1,-1,-1,0},
                            {-1,-1,-1,-1,0},
                            {-1,-1,-1,-1,0},
                            {-1,-1,-1,-1,0},
                            {-1,-1,-1,-1,0},
                            {-1,-1,-1,-1,0},
                            {-1,-1,-1,-1,0},
                            {-1,-1,-1,-1,0},
                            {-1,-1,-1,-1,0}};


    //================bullets initialization
    struct bullet up[10]={{-1,-1,0,type_up,-1,-1},
                            {-1,-1,0,type_up,-1,-1},
                            {-1,-1,0,type_up,-1,-1},
                            {-1,-1,0,type_up,-1,-1},
                            {-1,-1,0,type_up,-1,-1},
                            {-1,-1,0,type_up,-1,-1},
                            {-1,-1,0,type_up,-1,-1},
                            {-1,-1,0,type_up,-1,-1},
                            {-1,-1,0,type_up,-1,-1},
                            {-1,-1,0,type_up,-1,-1}};

    upcnt=up-1;

    struct bullet down[10]={{-1,-1,0,type_down,-1,-1},
                              {-1,-1,0,type_down,-1,-1},
                              {-1,-1,0,type_down,-1,-1},
                              {-1,-1,0,type_down,-1,-1},
                              {-1,-1,0,type_down,-1,-1},
                              {-1,-1,0,type_down,-1,-1},
                              {-1,-1,0,type_down,-1,-1},
                              {-1,-1,0,type_down,-1,-1},
                              {-1,-1,0,type_down,-1,-1},
                              {-1,-1,0,type_down,-1,-1}};


    downcnt=down-1;

    struct bullet left[10]={{-1,-1,0,type_left,-1,-1},
                              {-1,-1,0,type_left,-1,-1},
                              {-1,-1,0,type_left,-1,-1},
                              {-1,-1,0,type_left,-1,-1},
                              {-1,-1,0,type_left,-1,-1},
                              {-1,-1,0,type_left,-1,-1},
                              {-1,-1,0,type_left,-1,-1},
                              {-1,-1,0,type_left,-1,-1},
                              {-1,-1,0,type_left,-1,-1},
                              {-1,-1,0,type_left,-1,-1}};
    leftcnt=left-1;

    struct bullet right[10]={{-1,-1,0,type_right,-1,-1},
                             {-1,-1,0,type_right,-1,-1},
                             {-1,-1,0,type_right,-1,-1},
                             {-1,-1,0,type_right,-1,-1},
                             {-1,-1,0,type_right,-1,-1},
                             {-1,-1,0,type_right,-1,-1},
                             {-1,-1,0,type_right,-1,-1},
                             {-1,-1,0,type_right,-1,-1},
                             {-1,-1,0,type_right,-1,-1},
                             {-1,-1,0,type_right,-1,-1}};
    rightcnt=right-1;
    //show_label(type_begin);
    //show_press_enter();
    //show_button_explain();
    //show_buttons();
    level=6q           ` ;
    level_initialize(s);
    show_level();
    show_hit();

    //clear_screen();

    struct bullet *bit;
    //=====================================

    struct pointer p1={origin_x, origin_y};
    printf("pointer x=%d y=%d\n",p1.x, p1.y);
    show_pointer(&p1);

    while(1){
        expand(s,&p1);
        printf("upstop=%d, ", upstop);
        printf("downstop=%d, ", downstop);
        printf("leftstop=%d, ", leftstop);
        printf("rightstop=%d\n", rightstop);
        //printf("upcnt=%d ", upcnt);
    // printf("up=%d ", up);
        while(upcnt>up-1&&(upstop==0||downstop==0||leftstop==0||rightstop==0)){
                //printf("cc=%d",cc);
                //printf("There are bullets.\n");
                for(bit=upcnt; bit>up-1; bit--){
                      printf("in!!    ");
                      if(bit->type!=type_blank){

                            printf("b%d.x=%d, ", bit->num-snum-1, bit->x);
                            printf("b%d.y=%d, ", bit->num-snum-1, bit->y);
                            print_btype(bit);
                            printf("\n");

                            bit->y=bit->y-speed;
                            if(bit->y<20){
                                  printf("b%d hits wall",bit->num-snum-1);
                                  print_btype(bit);
                                  printf("\n");
                                  bit->type=type_blank;
                                  bwd(bit);
                            }
                            else{

                                  bwd(bit);
                                  for(i=0; i<snum; i++){

                                      hit((s+i),bit);
                            }

                      }
                }
                      //for(i=0; i<1000; i++);
                }

              for(bit=downcnt; bit>down-1; bit--){
                      if(bit->type!=type_blank){

                            printf("b%d.x=%d, ", bit->num-snum-1, bit->x);
                            printf("b%d.y=%d, ", bit->num-snum-1, bit->y);
                            print_btype(bit);
                            printf("\n");

                            bit->y=bit->y+speed;
                            if(bit->y>460){
```

```
                printf("b%d hits wall",bit->num-snum-1);
                print_btype(bit);
                printf("\n");
                bit->type=type_blank;
                bwd(bit);
            }
            else{
                bwd(bit);
                for(i=0; i<snum; i++){

                    hit((s+i),bit);
            }

        }
}
        //for(i=0; i<1000; i++);
}


  for(bit=leftcnt; bit>left-1; bit--){
        if(bit->type!=type_blank){

            printf("b%d.x=%d, ", bit->num-snum-1, bit->x);
            printf("b%d.y=%d, ", bit->num-snum-1, bit->y);
            print_btype(bit);
            printf("\n");

            bit->x=bit->x-speed;
            if(bit->x<10){
                printf("b%d hits wall",bit->num-snum-1);
                print_btype(bit);
                printf("\n");
                bit->type=type_blank;
                bwd(bit);
            }
            else{
                bwd(bit);
                for(i=0; i<snum; i++){

                    hit((s+i),bit);
            }

        }
}
        //for(i=0; i<1000; i++);
}

  for(bit=rightcnt; bit>right-1; bit--){
        if(bit->type!=type_blank){
            printf("b%d.x=%d, ", bit->num-snum-1, bit->x);
            printf("b%d.y=%d, ", bit->num-snum-1, bit->y);
            print_btype(bit);
            printf("\n");

            bit->x=bit->x+speed;
            if(bit->x>340){
                printf("b%d hits wall",bit->num-snum-1);
                print_btype(bit);
                printf("\n");
                bit->type=type_blank;
                bwd(bit);
            }
            else{

                bwd(bit);
                for(i=0; i<snum; i++){

                    hit((s+i),bit);
            }

        }
  }
        //for(i=0; i<1000; i++);
  }

  //////////////////////////
for(bit=upcnt; bit>up-1; bit--){

        if(bit->type!=type_blank){
            //printf("up_offset=%d, ",(bit-up));
            upstop=0;
            //printf("******break\n");
            break;
        }
}

if(bit==up-1){
    printf("bit= up -1")    ;
    //printf("check**********up_offset=%d\n",(bit-up));
    upstop=1;
}
/////////////////
for(bit=downcnt; bit>down-1; bit--){
        if(bit->type!=type_blank){
            downstop=0;
            break;
        }
}
```

```
      if(bit==down-1){
         downstop=1;
      }
      /////////////////
      for(bit=leftcnt; bit>left-1; bit--){
           if(bit->type!=type_blank){
              leftstop=0;
              break;
           }
      }
      if(bit==left-1)
         leftstop=1;
      ////////////////
      for(bit=rightcnt; bit>right-1; bit--){
           if(bit->type!=type_blank){
              rightstop=0;
              break;
           }
      }
      if(bit==right-1)
         rightstop=1;
      /*
      printf("upstop=%d, ", upstop);
      printf("downstop=%d, ", downstop);
      printf("leftstop=%d, ", leftstop);
      printf("rightstop=%d\n", rightstop);
     */
      for(i=0; i<7000; i++);

   }
 //struct snapper ss1={origin_x, origin_y, 0 , type_blank,0};
if(stage_clear(s)){
      //printf("yuhan");
      clear_screen(s);
      level++;
      printf("***********************level up\n");
      level_initialize(s);
      //pointed=&ss1;
      show_level();
      show_hit();
      upstop=1;
      downstop=1;
      leftstop=1;
      rightstop=1;
      upcnt=up-1;
      downcnt=down-1;
      leftcnt=left-1;
      rightcnt=right-1;
      //printf("enter clear\n");
   }
```

```
if(tap_left==0&&stage_clear(s)==0)
     game_over(s);


while (!IORD_8DIRECT(PS2_BASE, 0)) ; /* Poll the status */
code = IORD_8DIRECT(PS2_BASE, 4);
//printf("code=%d\n",code);

switch(code){
   case 240:
      release=1;
      break;

   //up
   case 117:
        if(release==1){
              tone=1;

           for (delay=1; delay<10000; delay++)
              IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
           for (delay=1; delay<10000; delay++)
              IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+14);
           for (delay=1; delay<10000; delay++)
              IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
              for (delay=1; delay<10000; delay++)
              IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
           for (delay=1; delay<10000; delay++)
              IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+14);
           for (delay=1; delay<10000; delay++)
              IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
           tone=3;
           IOWR_16DIRECT(AUDIO_BASE,0,(tone*256));
            //printf("up released\n");
            release=0;
            contract(pointed);

            if(p1.y>35){
                 p1.y=p1.y-72;
                 show_pointer(&p1);
            }
            printf("pointer: %d, %d\n", p1.x, p1.y);
        }
        break;

  //down
  case 114:
          if(release==1){
              tone=1;

           for (delay=1; delay<10000; delay++)
        IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
      for (delay=1; delay<10000; delay++)
        IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+14);
      for (delay=1; delay<10000; delay++)
        IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
        for (delay=1; delay<10000; delay++)
        IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
      for (delay=1; delay<10000; delay++)
        IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+14);
      for (delay=1; delay<10000; delay++)
        IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
      tone=3;
      IOWR_16DIRECT(AUDIO_BASE,0,(tone*256));
       //printf("down released\n");
       release=0;
       contract(pointed);

       if(p1.y<480-36){
          p1.y=p1.y+72;
          show_pointer(&p1);
       }
       printf("pointer: %d, %d\n", p1.x, p1.y);
    }
     break;




//left
case 107:
    if(release==1){
        tone=1;

      for (delay=1; delay<10000; delay++)
        IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
      for (delay=1; delay<10000; delay++)
        IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+14);
      for (delay=1; delay<10000; delay++)
        IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
        for (delay=1; delay<10000; delay++)
        IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
      for (delay=1; delay<10000; delay++)
        IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+14);
      for (delay=1; delay<10000; delay++)
        IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
      tone=3;
      IOWR_16DIRECT(AUDIO_BASE,0,(tone*256));
       //printf("left released\n");
       release=0;
       contract(pointed);
```

```
        if(p1.x>36){
           p1.x=p1.x-72;
           show_pointer(&p1);
        }
        printf("pointer: %d, %d\n", p1.x, p1.y);
     }
    break;

 //right
 case 116:
      if(release==1){
         tone=1;

       for (delay=1; delay<10000; delay++)
         IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
       for (delay=1; delay<10000; delay++)
         IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+14);
       for (delay=1; delay<10000; delay++)
         IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
         for (delay=1; delay<10000; delay++)
         IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
       for (delay=1; delay<10000; delay++)
         IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+14);
       for (delay=1; delay<10000; delay++)
         IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
       tone=3;
       IOWR_16DIRECT(AUDIO_BASE,0,(tone*256));
        //printf("right released\n");
        release=0;
        contract(pointed);

        if(p1.y<324){
           p1.x=p1.x+72;
           show_pointer(&p1);
        }
        printf("pointer: %d, %d\n", p1.x, p1.y);
     }
     break;
//enter
case 90:
   if(release==1){
     tone=1;
       for (delay=1; delay<10000; delay++)
         IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
       for (delay=1; delay<10000; delay++)
         IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+14);
       for (delay=1; delay<10000; delay++)
         IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
         for (delay=1; delay<10000; delay++)
```

```
                        IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
                    for (delay=1; delay<10000; delay++)
                        IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+14);
                    for (delay=1; delay<10000; delay++)
                        IOWR_16DIRECT(AUDIO_BASE,0,(tone*256)+15);
                    tone=3;
                    IOWR_16DIRECT(AUDIO_BASE,0,(tone*256));
                //printf("enter released\n");
                for(i=0; i<snum; i++){
                        if((s+i)->pointed==1){
                              tap_left--;              //***********************minus tap here
                              show_hit();
                              printf("taps left:%d\n", tap_left);
                              if((s+i)->type!=type_lred){ //not big red
                                    (s+i)->type=(s+i)->type+4;
                                    printf("s%d changes type to %d\n",i+1, (s+i)->type);
                                    swd(s+i);
                                    break;
                              }
                              else{
                                      if((s+i)->type!=type_explode){    // red snapper explodes
                                      printf("s%d wants to explode\n", i+1) ;
                                      (s+i)->type=type_explode;
                                      swd(s+i);
                                      upcnt++;
                                      //printf("after enter, upcnt=%d\n", upcnt);
                                      downcnt++;
                                      leftcnt++;
                                      rightcnt++;
                                      upstop=0;
                                      downstop=0;
                                      leftstop=0;
                                      rightstop=0;
                                      explode_init(s+i, upcnt, downcnt, leftcnt, rightcnt);
                                      break;
                                         }
                              }
                        }

                 }
            }

            break;

        }
    }

    return 0;
}
```