Super Frogger

CSEE 4840 Embedded System Design
Final Report

Department of Electrical Engineering
School of Engineering and Applied Science
Columbia University

May 2012

Group Member:

Ziyao Xu zx2127
Xin Zhang xz2270
Shengzhen Li sl3356
Chenxi Liu cl2985

1. Introduction

Frogger is an arcade game that was first developed by Konami in 1981. The game's object is to guide a frog safely to its home on the opposite side of the screen. To do this, you have to lead its way across a high way where cars and other vehicles are running. After this, there is also a dangerous river lying before the frog and it also needs to come through by stepping on the log and turtle floating on it. If all the five frogs safely come back to their home, you succeed in this level and have face next which has more speedy cars and logs.

2. Architecture

The following is our block diagrams of the whole design architecture, which shows the connection between different modules and the interaction between the CPU and hardware drivers. In this project, we mainly focus on VGA and Audio part.

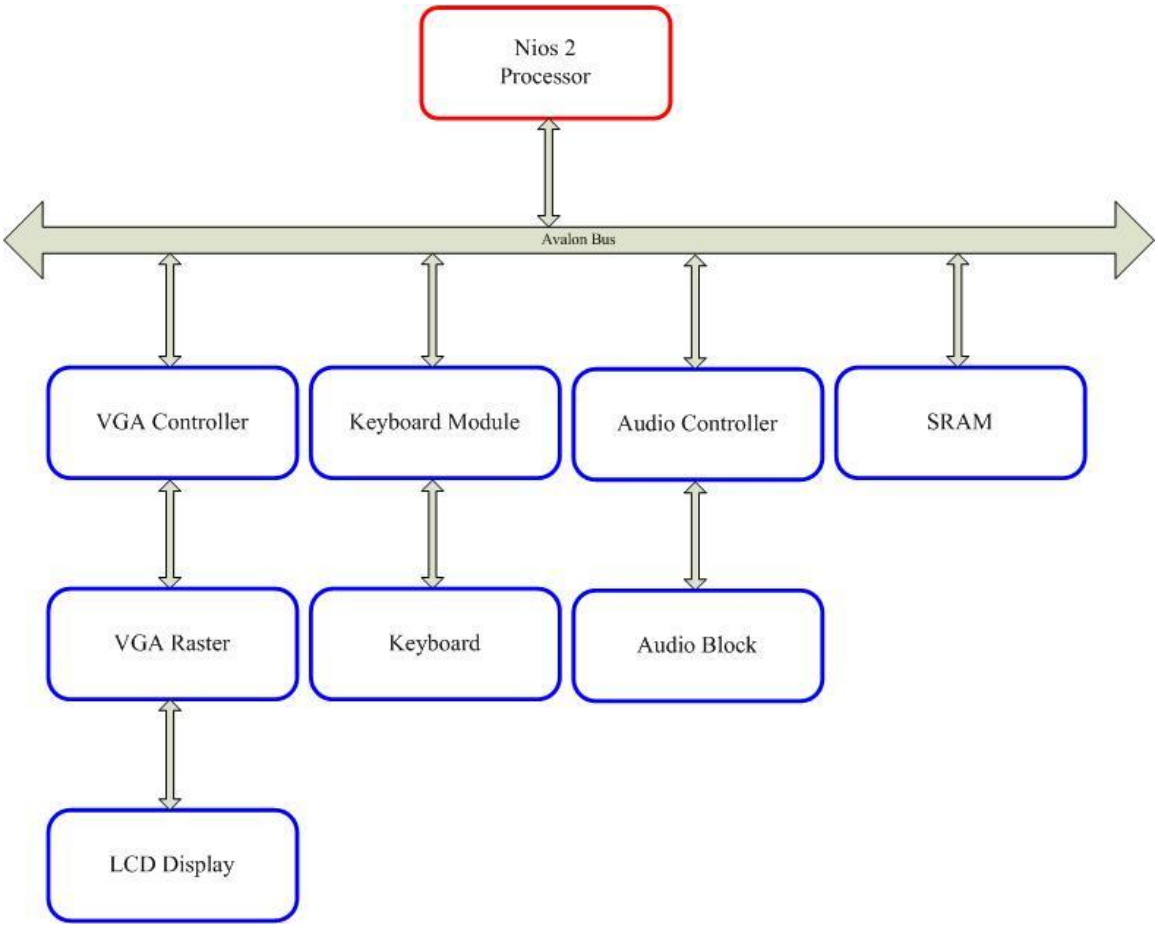


Figure 2.1 System Architecture

3. VGA PART

3.1 Video Display

Below is the block diagram for video part of our project. The DE2_VGA_RASTER takes use of five controllers each of which controls a separate part. For example, the frog controller deals with the display of all the issues with frog, like jumping and dying scenarios.

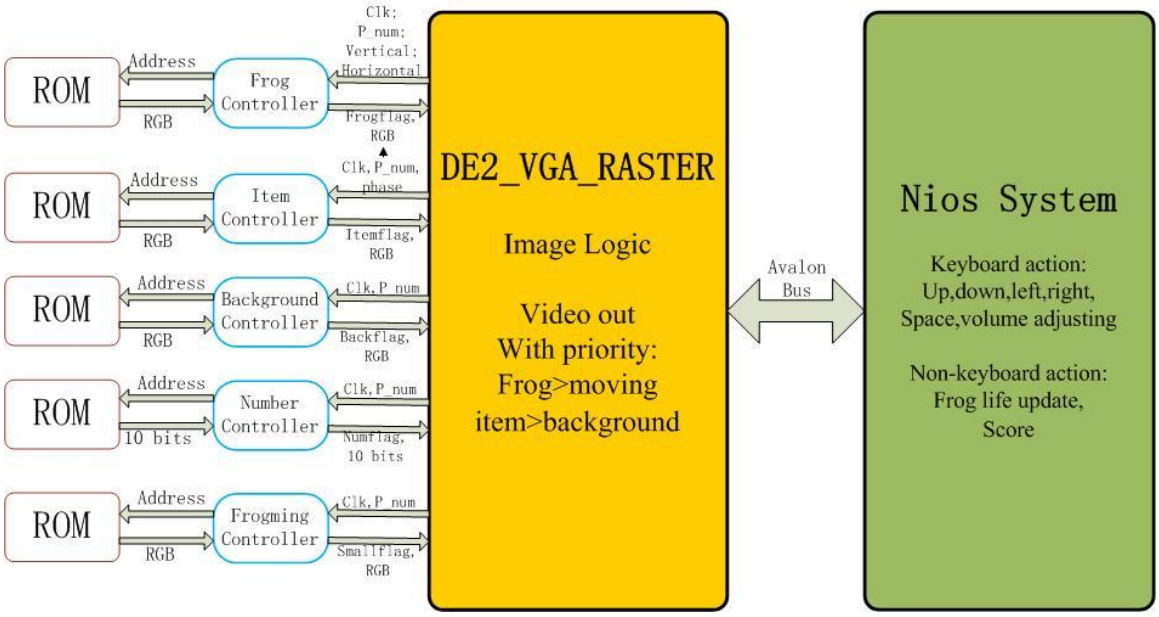


Figure 3.1 The Video Block Diagram

In order to imitate the origin game "Frogger", we need to display different part of articles. They include the frog, moving items on the road and in the river, background pictures, frog lives left, score, time, etc.

The main part of the VGA screen displays the game zone, which contains a frog, moving items and background pictures. They are displayed in three layers: the background pictures are on the bottom, the moving items in the middle and the frog on the very top. All articles here are basically in a tile with size of 32*32. Since we have three layers, each item should only be displayed within the outline of the "real" item instead of whole 32*32 pixels, deleting unnecessary pixels (to display pictures in the next layer) and setting the flag off. In addition, the frog requires the highest priority, following the moving item on the road in the river, and then the background pictures at last. Now we first introduce how each of these three layers display, followed with transparent and priority part.

3.2 Background

Background pictures include river, bush, road, frog home, ground and lawn. It's easy to display since most of them are stationary. The only concern here is that we could not make background composed of all different tiles because of the limit space of on-chip memory. Thus one feasible way to deal with the concern is to set background periodically utilizing only a small number of tiles. The main game zone is divided into 16*14= 224 tiles shown in the figure below. The tile labeled 0 and that labeled 1 is a same tile but displayed in an opposite direction. By doing this we could addressing the same ROM with only different indexes which save the space of ROM.

```
type pattern is array(integer range 0 to 14, integer range 0 to 15) of integer;
                constant river_pattern: pattern := (
    (  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0),
    (  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0),
    (  4,  3,  3,  3,  2,  4,  3,  3,  3,  3,  2,  4,  3,  3,  3,  2),
    (  0,  5,  5,  0,  0,  0,  5,  5,  5,  0,  0,  5,  5,  5,  0,  0),
    (  0,  0,  0,  0,  4,  3,  3,  2,  0,  0,  0,  4,  3,  3,  2,  0),
    (  0,  0,  0,  5,  5,  5,  0,  0,  0,  0,  0,  5,  5,  5,  0,  0),
    (  4,  2,  0,  0,  0,  4,  3,  2,  0,  0,  0,  4,  3,  2,  0,  0),
    (  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0),
    (  0,  0,  9,10,  0,  0,  0,  9, 10,  0,  0,  0,  9,10,  0,  0),
    (  0,  8,  0,  0,  0 ,8 ,0 ,0 ,0 ,0 ,0 ,8 ,0 ,0 ,0 ,  0),
    (  7,  0,  0,  0,  0,  0,  0,  0,  7,  0,  0,  0,  0,  0,  7,  0),
    (  0,  0,  6,  0,  0,  0,  0,  0,  0,  6,  0,  0,  0,  0,  6,  0),
    (  1,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  1,  0),
    (  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0),
    (  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0) );
```

Figure 3.2 Tile pattern that constructs the moving items

```
signal background: pattern := (
    ( 20,21,20,21,20,21,20,21,20,21,20,21,20,21,20,21),
    ( 12,14,16,12,14,16,12,14,16,12,14,16,12,14,16,12),
    ( 10,11,10,11,10,11,10,11,10,11,10,11,10,11,10,11),
    ( 10,11,10,11,10,11,10,11,10,11,10,11,10,11,10,11),
    ( 10,11,10,11,10,11,10,11,10,11,10,11,10,11,10,11),
    ( 10,11,10,11,10,11,10,11,10,11,10,11,10,11,10,11),
    ( 10,11,10,11,10,11,10,11,10,11,10,11,10,11,10,11),
    (  8,  9,  8,  9,  8,  9,  8,  9,  8,  9,  8,  9,  8,  9,  8,  9),
    (  6,  7,  6,  7,  6,  7,  6,  7,  6,  7,  6,  7,  6,  7,  6,  7),
    (  4,  5,  4,  5,  4,  5,  4,  5,  4,  5,  4,  5,  4,  5,  4,  5),
    (  4,  5,  4,  5,  4,  5,  4,  5,  4,  5,  4,  5,  4,  5,  4,  5),
    (  4,  5,  4,  5,  4,  5,  4,  5,  4,  5,  4,  5,  4,  5,  4,  5),
```

```
(  2,  3,  2,  3,  2,  3,  2,  3,  2,  3,  2,  3,  2,  3,  2,  3),
(  0,  1,  0,  1,  0,  1,  0,  1,  0,  1,  0,  1,  0,  1,  0,  1),
( 30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30));
```

Figure.3.3 The tile pattern that constructs the background

In figure.3.3, the even number shows the origin tile while the odd number represents the reverse image of the corresponding even tile. Using this method, the background is built with minimum ROM consumption while reserving a good display performance.

3.3 Moving items

Essentially, the display scheme of moving items is similar to the background part. One difference here is that we need to let these items move. Thus we set an offset to each of the tile rows respectively. By changing the value of the offset we realize the motion of objects including logs, cars, and turtles. However, we need the items to move smoothly but in different speed. Basically, moving smoothly means that the some items should move just one pixel in certain number of clock cycles, while others with different clock cycles.

Moreover we want some items, especially the turtles in the river move vividly just like they are "really" swimming. In order to do that, we set the turtles moving in three phases. The phase of the turtle changes to another picture in a certain order (like finite state machine) when the offset of corresponding row changes. Thus when moving towards, the tile sample of the turtles changes periodically according to the phase.

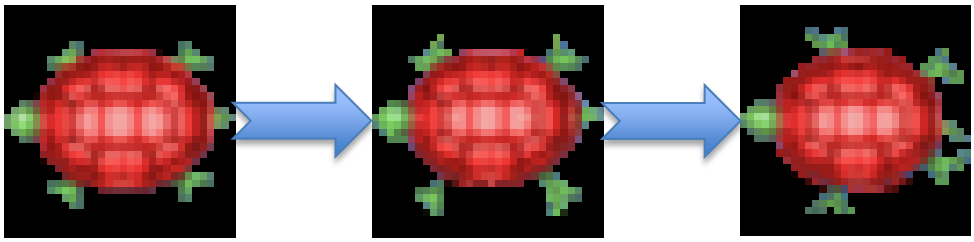


Figure 3.4 Three phases of a swimming turtle

3.4 Frog

The frog is on the very top layer and controlled by the keyboard. We changed the generate functions of index when reading from ROM in order to display frog in all four directions based on one sample, which saves us the trouble to store four pictures of the same frog. The code that realizes this operation is shown below.

```
case direction is
 when 2 =>   --left
        index <= to_unsigned ((horizontal * 32 + 31 - vertical), 10);
```

```
when 3 =>  --down
        index <= to_unsigned (((31-vertical) * 32 + 31 - horizontal), 10);
when 4 =>  --right
        index <= to_unsigned ((vertical + (31 - horizontal) * 32), 10);
when others =>  --up
        index <= to_unsigned ((vertical * 32 + horizontal), 10);
end case;
```

The index is used to get the precise pixel value of RGB in the one-dimension matrix of each ROM vhd files.

The "jump" action is controlled by the nios-system by generating two periods of delay when the frog is jumping in the air. After the first delay, the display of the frog changes to another picture in a coordinate 16 pixels away (depending on which direction the fro is jumping) and after the second delay, it turns back with the coordinate changed another 16 pixels.

Figure 3.5 Jumping movement of frog

One hard part here is that when frog jumps onto turtles or logs in the river, it needs to drift with the turtles/logs. The implement we take is that a vector of offset is set to the frog. Each element in the vector means the offset of frog in a certain row. When frog is in the river zone and does not fall into water, the offset of that row increases in the same rate of the item in that row, and the other elements in the offset vector are set to be zero.

When the frog collides with a car or jumps into river, it dies (which seems ridiculous but it is how the origin game is designed.) At this time, a continuous two frames of image is shown where the frog dies.

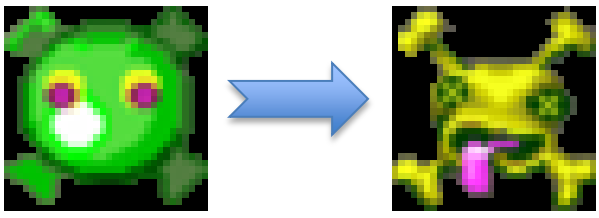


Figure 3.6 Two phases when the frog dies

3.5 Transparency & Priority

For each sample of tile, we set the part outside the real article a certain color, when the ROM-controller read that color, it will tell the VGA-raster not to display that part. For example:

```
        if frog_dead2_R = "0000" and frog_dead2_G = "0000" and frog_dead2_B =
"0000" then  --transparent the background
          frogflag <= '0';
        else   --display character
          frogflag <= '1';
          pixel_R <= frog_dead2_R;
          pixel_G <= frog_dead2_G;
          pixel_B <= frog_dead2_B;
  end if;
```

Priority is implemented in the aid of flag. We assign three different flags to the three layers, namely back-flag, item-flag and frog-flag. When the screen scans to some point, we first check whether the frog-flag's value is '1', if so, display the color provided by the frog-controller; if not, then check the item-flag and last back-flag.

3.6 Logic

The logic part of this project is mainly to determine whether the frog is crashed by the vehicles on the road or falls into the water. When the frog in the road zone, we just check the coordinates of the left and the right sides of the frog, comparing them to the corresponding elements of the item matrix. If the elements' values are zero, then no crash happen, otherwise the frog loses one life. When the frog is in the river zone, the scheme is similar. But this time we check the coordinates of the middle of frog instead.

3.7 Scoreboard

Number generation ROM:

We here adopt the method of number display in lab 2. Each number is displayed within a 16*8 pattern with only white color. This is because in our design, we are not mainly focusing on the picture and number quality. If we use three dimensional RGB to represent the number (even if we still use 4 bits to display numbers), there will still be taking much unnecessary memory space without leaving sufficient space for audio part. Actually at last the compilation summary indicates that the total memory space has reached up to 80% including video 65% and audio 15%.

For the scoreboard, we mainly build 6 matrixes to store the information to be displayed. Your score, high score, time left, life left, "press space" word and the level you

have been reaching up to are going to be displayed. Every time the content of these elements are changed in response to the control signal from either C or VHDL, the tile number stored in the matrix will be replaced by the desired value. All of the mapping from tile number to real RGB display on the screen is controlled by the number_controller, with the logic presented in the Display_number process. The controller signals are sent or received through Avalon bus in the process Key. On top of that, the small frog matrix is also constructed in scoreboard field as displaying colored number of remaining life, which is similar with other item RGB ROMs, but different from the number read and write operation. The time bar is achieved by counting down 15s then set the frog_alive to "01" which means the first stage of death (we have 4 states of death), and there will be a warning bar when the time left is less than 5s.

3.8 Software Design

The Nios2 C software programming mainly receives the keyboard action and detects the signal sent from the VHDL. Also the signals controlling the background music and special music are generated by C.

Video:

The whole architecture of C program is designed like this:

```
while(1){
        if (keyboard action(up, down, left, right))
            if (!pause)
            calculation & write to bus;
        else
            check frog life;
            write score or clear everything;
}
```

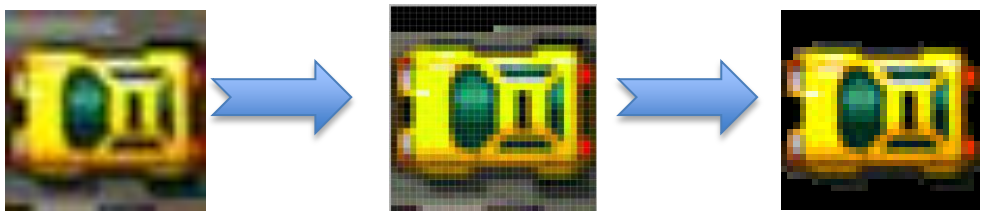
Therefore, one case is that there is keyboard action, and the C program reads the control signals like frog_alive and pause, then it will calculate the x and y axis of the frog to VHDL for display. Meanwhile there are constants recording the number of win and the current scores are updated and sent back.

The signals transmitted between software and hardware with their addresses (offset) is:

```
20  write & read forg_x
21  write & read frog_y
22  write frog_direction
23  write frog_action
24  read frog_alive
25  write & read pause
26  write score
27  read frog_life
28  write frog_x
29  write clear
```

3.9 Image Processing

We first use the 24 bits display method, while Red, Green, Blue each owns 8 bits. In implementing this, we first screenshot the whole picture of a Frogger game, and use software to cut each tile we need and shrink it into a 32 * 32 pixel image as shown in figure 3.7 (a). Then use photoshop to fill each pixel that we do not need (background) to a certain color (In this case, we use pure black, which is R:000,G:000,B:000, figure 3.7(b) and (c).) The third step is using Matlab to generate three one-dimension matrix vhd files which represents R,G,B respectively that meets the requirement of VHDL. However, when we implement it in our project, we find that the ROM we use for storing all the image tiles is not enough. In fact, it is consumed nearly two thirds while we only put half image tiles in it. So, it leaves us with no choice but to cut the origin 24 bits to half, that is 4 bits for R,G,B respectively. After doing this revision, we find that the image display quality is not decreasing tremendously. In fact, it remains at least seventy percent the origin performance, but saves half space.



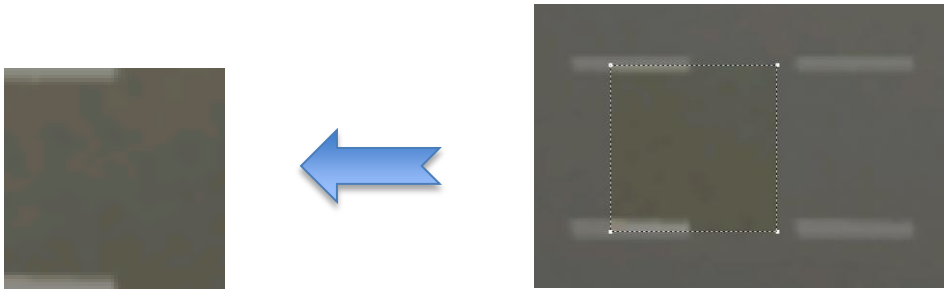
(a) (b) (c)

Figure 3.7 How the car tile is made

For the background part, first we want to use a whole 512 * 448 pixel picture. However, it later occurs to us that we could use the same tile idea as it is in the moving items. In our game display, there are a lot of parts, like the road and water, which are nearly the same. So we decide to use only one tile and construct the image using cyclic display of this tile and its reverse image, which is shown in figure 3.3. The code that realizes this part is simple and is shown below. At last, the whole background is consisted of only eight different tiles while keeping the most information of the origin 512 * 448 pixel picture.

```
if (address and "00001") = "00000" then  – judge whether the address is even or odd
index <= to_unsigned ((vertical*32 + horizontal), 10);
else
index <= to_unsigned ((vertical*32 + (31 - horizontal)), 10); -- reverse the image
```

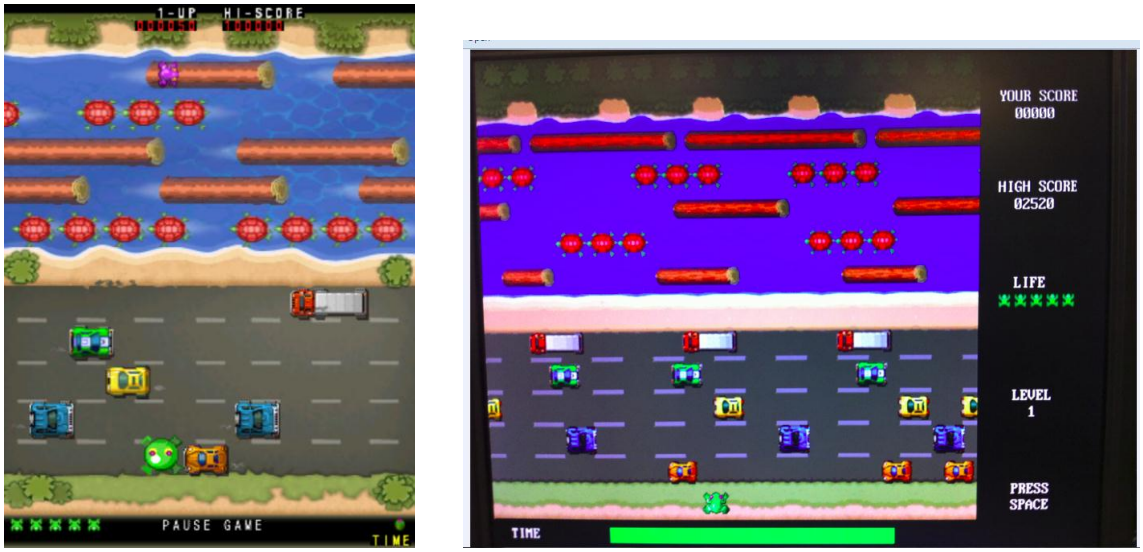
For example, all the road in the lower part is built using the single tile shown in figure 3.8(a). In fact, it is cut from the whole picture like the way it shows in figure 3.8 (b)



(a) (b)

Figure 3.8 Example of how a tile is cut

The final performance of our project is shown below. Actually, the real display quality is better than what the image shows because this photo is taken by cellphone and some details are not properly displayed (like the uppermost part and the water).



(a) The origin screenshot (b) Our Display Performance

Figure 3.9 The final display effect

4. Audio PART

4.1 Hardware Design

On Altera DE2 board, the audio implementation is provided by a 24 bits Wolfson Audio CODEC chip WM8731. This chip supports sampling frequency from 8KHz to 96KHz and four audio modes which is I2S, right justified, left justified and DSP mode. In our project, we only focus on the first mode with default configuration that both channels play the same soundtrack. The general block diagram of the audio design is shown as follows.

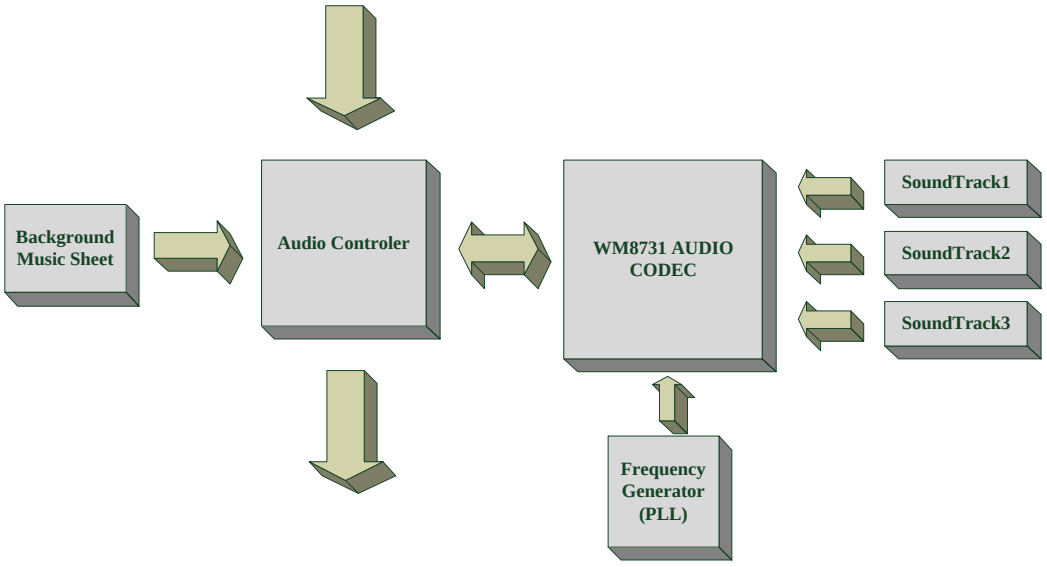


Figure 4.1 Audio Block Diagram

Audio Controller unit is responsible to control the audio codec to choose the correct music as expected by some certain input signals from the SRAM. In our design, there are two different kinds of music which are implemented in two different ways. One is the background music of the video game. Since the quite long length compared with other special sounds, it is implemented just like the MIDI music, that is, to store a certain period of wave first, then control the sampling frequency to get different pitches. The background music sheet is stored by a number of 16bits encoded symbols containing the information of the sampling frequency, delay, volume and sound No.

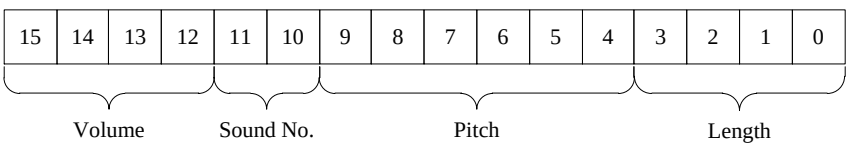


Figure 4.2 The structure of an audio symbol

Figure 1.2 illustrates the format of the encoded symbols. Each symbol has 16 bits as we described before: the leftmost 4 bits is to indicate the volume of the sound, bit 11 and bit 10 is related to the number of soundtrack, bit 9 to bit 4 select the pitch and the rightmost 4 bits give a certain length of this pitch. When a signal is read from the sheet, it will be sent to the decoder which is just inside the Audio Controller and send out the corresponding values of volume, soundtrack number, sampling frequency and length. This method can save more than 99% space of the ROM or SRAM. The other type is for the special music. We just store the sample values from the special waves and have a single frequency factor to control the sampling frequency of each special sound. Although the music is relatively short, i.e, 0.2 seconds, the number of the samples can be as much as 1600 if the sampling frequency is 8KHz.

When WM8731 Audio Codec receives the information from the controller, it will choose the soundtrack and output the sample values in order. Timing is significant in this unit. We design a PLL which has a frequency of 18.432MHz to provide the clock signal. 18.432MHz is calculated by the expression

$$18.432MHz = 48KHz \times 2 \times 16 \times 12$$

48KHz is the maximum sampling frequency for multiple channels, 2 is the number of the channels, 16 is the bit's number of each stored sample value. That is to say, both channels have the only output port which output sample values bit by bit.

Actually, in this game, the ports of background music and special music are separated with each other and provide signals to WM8731 simultaneously. Soundtrack 1 is for the background music, which is a normal sine wave with 144 samples. Soundtrack 2 and 3 are special music, each of which has 1600 samples by sampling frequency equals to 8KHz. To play the background music and special music simultaneously, just add them together and output the overlapped values bit by bit. The two special sounds are not allowed to play at the same time. To make sure the special music will not be interfered by the background one, the volume of the background music is only half of the volume of the special ones.

4.2 Software Design

In this game, all the control signals are from the Nios and stored in the SRAM including start signal, the sound number and the volume. The Audio Controller will read the data from this SRAM in each clock cycle.

	Background	Movement	Death
Control signal	None	Jump = 0	"alife" != 0
Total length	> 20s	0.2s	0.4s
Storage method	Sine wave	Entire wave	Entire wave
Sampling frequency	Controlled by pitches	8KHz	8KHz

Figure 4.3 Characters of audio

As we can see, the background music begins when the program is downloaded into the FPGA chip and plays in a cycle. One special sound called Movement is for the jumping movement of the frog. Although we can generate the start signal directly by the keyboard, since the image of jumping changes after a period of delay when holding the direction keys, this sound will be controlled by the same signal of jumping for video. The collision sound is a little different. Since The case "death" may happen both when pressing the key or not, it will be monitored before the judge of the keyboard. After all the design above, we add one more function, that is, we can control the volume of the music by the "+/-" keys on the keyboard. These are the only two keys available when the game is paused.

5. Contribution

Video part:

Chenxi Liu: architecture design, item and frog movement implementation, display adjustment, logic and timing, Nios C controller implementation.

Shengzhen Li: characters ROMs construction, picture processing and whole component controller interface construction.

Xin Zhang: design and implement scoreboard logic, debugging frog and item movement logic, Nios C keyboard controller construction and adjustment.

Audio part:

Ziyao Xu: responsible for the whole audio architecture design and implementation.

Finally all of us were working at putting the video and audio together and we finished our whole project at last. It is proved very efficient that all of the four members of this team work in parallel, avoiding time waste or idling of some members. As you can see, we indeed realize nearly all the design idea proposed in our design report.

6. Source Code

The image matrix vhds only contain a small part of numbers to keep this report from too verbose. The omission part is marked using ellipses.

main.c function:

```
#include <io.h>
#include <system.h>
#include <stdio.h>
#include <alt_types.h>
#include <sys/alt_irq.h>

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

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

unsigned char code;

int main()
{
  int count, i;
  int frog_x = 0;
  int frog_y = 0;
  int frog_alive=0;
  int pause = 0;
  int stopflag = 1;
  int frog_action = 0;
  int frog_direction=1;
  const int max=70000;
//  int frog_last_x;
  int frog_last_y = 416;
  int score = 0;
  int frog_life = 5;
  int dummy_life = 5;
  int offset;
  int clear = 1 ;

  // audio
  int VolRd = 0, VolWr = 0;
  int a, audioi;
```

```
while(1) {

  frog_alive = IORD_VGA_DATA(VGA_RASTER_BASE, 24);

  // audio collision
  if (frog_alive == 1) {
   IOWR_16DIRECT(AUDIO_BASE, 1*2, 0x0001);
   IOWR_16DIRECT(AUDIO_BASE, 3*2, 0x0001);
     audioi = 0;
     while (1) {
        audioi++;
        a = IORD_16DIRECT(AUDIO_BASE, audioi*2);
        printf("a = %x\n", a);
        if (audioi == 4) {
           break;
        }
     }
//       printf("stopflag = %x\n", stopflag);
//    IOWR_16DIRECT(AUDIO_BASE, 1*2, 0x0000);
  }
  else if (frog_alive == 2) {
   IOWR_16DIRECT(AUDIO_BASE, 1*2, 0x0001);
   IOWR_16DIRECT(AUDIO_BASE, 3*2, 0x0000);
     audioi = 0;
     while (1) {
        audioi++;
        a = IORD_16DIRECT(AUDIO_BASE, audioi*2);
        printf("a = %x\n", a);
        if (audioi == 4) {
           break;
        }
     }
//       printf("stopflag = %x\n", stopflag);
//    IOWR_16DIRECT(AUDIO_BASE, 1*2, 0x0000);
  }

 if (IORD_8DIRECT(PS2_BASE, 0) == 1){
//   while (!IORD_8DIRECT(PS2_BASE, 0)) ; /* Poll the status */
  code = IORD_8DIRECT(PS2_BASE, 4);
  printf("%x\n", code);
//   if (code == 0xf0){
//    initial = 0;}
  if (code == 0x1d || code == 0x1b || code == 0x1c || code == 0x23 || code == 0x29 ||
code == 0x55 || code == 0x4e){
   if (stopflag == 1)
```

```
    stopflag = 0;
 else
    stopflag = 1;
}

//pause function
pause = IORD_VGA_DATA(VGA_RASTER_BASE, 25);
if (code == 0x29 && stopflag == 0){
 pause = (pause + 1)%2;
 IOWR_VGA_DATA(VGA_RASTER_BASE, 25, pause);
}

// audio volume
if (code == 0x4e && stopflag == 0){
  VolRd = IORD_16DIRECT(AUDIO_BASE, 2*2);
  IOWR_16DIRECT(AUDIO_BASE, 1*2, 0x0002);
  if (VolRd < 3) VolWr = VolRd + 1;
  IOWR_16DIRECT(AUDIO_BASE, 2*2, VolWr);
}
else if (code == 0x55 && stopflag == 0){
  VolRd = IORD_16DIRECT(AUDIO_BASE, 2*2);
  IOWR_16DIRECT(AUDIO_BASE, 1*2, 0x0002);
  if (VolRd > 0) VolWr = VolRd - 1;
  IOWR_16DIRECT(AUDIO_BASE, 2*2, VolWr);
}



//send frog x and y to C
if (frog_alive == 0 && pause == 0 ) {
frog_x = IORD_VGA_DATA(VGA_RASTER_BASE, 20);
frog_y = IORD_VGA_DATA(VGA_RASTER_BASE, 21);

if (code == 0x1d && stopflag == 0)   //up
   {
    printf("frog_last_y : %d\n", frog_last_y);
    printf("frog_y: %d\n", frog_y);
    printf("frog_life : %d\n", frog_life);
    frog_direction=1;
    IOWR_VGA_DATA(VGA_RASTER_BASE, 22, frog_direction);
    // AUDIO
    IOWR_16DIRECT(AUDIO_BASE, 1*2, 0x0000);

 ////////////see whether we should add score or not/////////////
    if (frog_y == frog_last_y){
      score = score + 10;
```

```
      frog_last_y = frog_y - 32;
      IOWR_VGA_DATA(VGA_RASTER_BASE, 26, score);
    }

    frog_y -= 16;

    frog_action = 1;
    IOWR_VGA_DATA(VGA_RASTER_BASE, 23, frog_action);
    IOWR_VGA_DATA(VGA_RASTER_BASE, 21, frog_y);
    IOWR_VGA_DATA(VGA_RASTER_BASE, 20, frog_x);
    // audio
    IOWR_16DIRECT(AUDIO_BASE, 3*2, 0x0001);
    audioi = 0;
    while (1) {
       audioi++;
       a = IORD_16DIRECT(AUDIO_BASE, audioi*2);
       //printf("a = %x\n", a);
       if (audioi == 4) {
          break;
       }
    }
    for (count=0;count<max;count++){
    }
    if (frog_y  > 0) frog_y -= 16;

    //check whether arrive at destination
    if (frog_y == 32 ){
       IOWR_VGA_DATA(VGA_RASTER_BASE, 21, frog_y);
       IOWR_VGA_DATA(VGA_RASTER_BASE, 28, frog_x);
       printf("HIHIHIHIHIHI\n");
       frog_last_y = 416;
    }
    else{
    frog_action = 0;
    IOWR_VGA_DATA(VGA_RASTER_BASE, 21, frog_y);
    IOWR_VGA_DATA(VGA_RASTER_BASE, 20, frog_x);
    IOWR_VGA_DATA(VGA_RASTER_BASE, 23, frog_action);
    }
    // audio
    IOWR_16DIRECT(AUDIO_BASE, 3*2, 0x0000);
  }
else if (code == 0x1b && stopflag == 0)  //down
  {
    frog_direction=3;
    IOWR_VGA_DATA(VGA_RASTER_BASE, 22, frog_direction);
    // AUDIO
```

```
    IOWR_16DIRECT(AUDIO_BASE, 1*2, 0x0000);

    if (frog_y < 416) frog_y += 16;
    frog_action = 1;
    IOWR_VGA_DATA(VGA_RASTER_BASE, 23, frog_action);
    IOWR_VGA_DATA(VGA_RASTER_BASE, 21, frog_y);
    IOWR_VGA_DATA(VGA_RASTER_BASE, 20, frog_x);
    // audio
    IOWR_16DIRECT(AUDIO_BASE, 3*2, 0x0001);
    audioi = 0;
    while (1) {
       audioi++;
       a = IORD_16DIRECT(AUDIO_BASE, audioi*2);
       //printf("a = %x\n", a);
       if (audioi == 4) {
          break;
       }
    }
    for (count=0;count<max;count++){
    }
    if (frog_y < 416) frog_y += 16;
    frog_action = 0;
    IOWR_VGA_DATA(VGA_RASTER_BASE, 21, frog_y);
    IOWR_VGA_DATA(VGA_RASTER_BASE, 20, frog_x);
    IOWR_VGA_DATA(VGA_RASTER_BASE, 23, frog_action);
    // audio
    IOWR_16DIRECT(AUDIO_BASE, 3*2, 0x0000);
  }



else if (code == 0x1c && stopflag == 0)  //left
  {
   frog_direction=2;
   IOWR_VGA_DATA(VGA_RASTER_BASE, 22, frog_direction);
   // AUDIO
   IOWR_16DIRECT(AUDIO_BASE, 1*2, 0x0000);
   if (frog_x >= 16) frog_x -= 16;
   else if (frog_x >= 0) frog_x = 0;
   frog_action = 1;
   IOWR_VGA_DATA(VGA_RASTER_BASE, 23, frog_action);
   IOWR_VGA_DATA(VGA_RASTER_BASE, 21, frog_y);
   IOWR_VGA_DATA(VGA_RASTER_BASE, 20, frog_x);
   // audio
   IOWR_16DIRECT(AUDIO_BASE, 3*2, 0x0001);
```

```
    audioi = 0;
     while (1) {
        audioi++;
        a = IORD_16DIRECT(AUDIO_BASE, audioi*2);
        printf("a = %x\n", a);
        if (audioi == 4) {
           break;
        }
     }
     printf("stopflag = %x\n", stopflag);
    for (count=0;count<max;count++){
    }
    if (frog_x >= 16) frog_x -= 16;
    else if (frog_x >= 0) frog_x = 0;
    frog_action = 0;
    IOWR_VGA_DATA(VGA_RASTER_BASE, 21, frog_y);
    IOWR_VGA_DATA(VGA_RASTER_BASE, 20, frog_x);
    IOWR_VGA_DATA(VGA_RASTER_BASE, 23, frog_action);
    // audio
    IOWR_16DIRECT(AUDIO_BASE, 3*2, 0x0000);
    printf("stopflag = %x\n", stopflag);
   }


else if (code == 0x23 && stopflag == 0)  //right
   {
     frog_direction=4;
     IOWR_VGA_DATA(VGA_RASTER_BASE, 22, frog_direction);
     // AUDIO
     IOWR_16DIRECT(AUDIO_BASE, 1*2, 0x0000);
     if (frog_x <= 464) frog_x += 16;
     else if (frog_x <= 480) frog_x = 480;
     frog_action = 1;
     IOWR_VGA_DATA(VGA_RASTER_BASE, 23, frog_action);
     IOWR_VGA_DATA(VGA_RASTER_BASE, 21, frog_y);
     IOWR_VGA_DATA(VGA_RASTER_BASE, 20, frog_x);
     // audio
     IOWR_16DIRECT(AUDIO_BASE, 3*2, 0x0001);
     audioi = 0;
     while (1) {
        audioi++;
        a = IORD_16DIRECT(AUDIO_BASE, audioi*2);
        //printf("a = %x\n", a);
        if (audioi == 4) {
           break;
        }
```

```
      }
      for (count=0;count<max;count++){
      }
      if (frog_x <= 464) frog_x += 16;
      else if (frog_x <= 480) frog_x = 480;
      frog_action = 0;

      IOWR_VGA_DATA(VGA_RASTER_BASE, 21, frog_y);
      IOWR_VGA_DATA(VGA_RASTER_BASE, 20, frog_x);
      IOWR_VGA_DATA(VGA_RASTER_BASE, 23, frog_action);
      // audio
      IOWR_16DIRECT(AUDIO_BASE, 3*2, 0x0000);

    }
  }
  audioi = 0;
      while (1) {
         audioi++;
         a = IORD_16DIRECT(AUDIO_BASE, audioi*2);
         //printf("a = %x\n", a);
         if (audioi == 4) {
            break;
         }
      }
}
  else
  {
   IOWR_VGA_DATA(VGA_RASTER_BASE, 26, score); //keep writing score
   dummy_life = frog_life;
   frog_life = IORD_VGA_DATA(VGA_RASTER_BASE, 27); //keep reading frog life
 //  printf("original frog_life : %d  ", frog_life);
   if (dummy_life > frog_life){
      frog_last_y = 416;
   }
   if (frog_life == 0 && frog_alive  == 1){
      frog_life = 5;
      score = 0;
      IOWR_VGA_DATA(VGA_RASTER_BASE, 29, clear);
   }
  }
}
  printf("Goodbye!\n");

  return 0;
}
//20  write read forg_x
```

```
//21  write read frog_y
//22  write frog_direction
//23  write frog_action
//24  read frog_alive
//25  write read pause
//26  write score
//27  read frog_life
//28  write frog_x
//29  write clear
```

Lab3_vga.vhd:

```
--
-- DE2 top-level module that includes the simple VGA raster generator
--
-- 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_vga 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_vga;

architecture datapath of lab3_vga is

 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 counter : unsigned(15 downto 0);
 signal reset_n : std_logic;
 signal audio_clock : unsigned(1 downto 0) := "00";

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);

 process (CLOCK_50)
 begin
  if rising_edge(CLOCK_50) then
   if counter = x"ffff" then
    reset_n <= '1';
   else
    reset_n <= '0';
    counter <= counter + 1;
   end if;
  end if;
 end process;

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

```
 nios : entity work.nios_system port map (
   clk                    => CLOCK_50,
   reset_n                => reset_n,

   -- the_audio
   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,

   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,

   PS2_Data_to_the_ps2 => PS2_DAT,
   PS2_Clk_to_the_ps2 => PS2_CLK,

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

);

  HEX7    <= "0001110"; -- Leftmost F
  HEX6    <= "0001000";  -- R
  HEX5    <= "1000000";  -- O
  HEX4    <= "1000010";  -- G
  HEX3    <= "1000010";  -- G
  HEX2    <= "0000110";  -- E
  HEX1    <= "0001000";  -- R
  HEX0    <= (others => '1');        -- Rightmost
  LEDG    <= (others => '1');
  LEDR    <= (others => '1');
  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';

--  I2C_SCLK <= '1';

  -- Set all bidirectional ports to tri-state
  DRAM_DQ    <= (others => 'Z');
  FL_DQ      <= (others => 'Z');
  SRAM_DQ    <= (others => 'Z');
  OTG_DATA   <= (others => 'Z');
  LCD_DATA   <= (others => 'Z');
  SD_DAT     <= 'Z';
--  I2C_SDAT   <= 'Z';
  ENET_DATA  <= (others => 'Z');
  GPIO_0     <= (others => 'Z');
  GPIO_1     <= (others => 'Z');

end datapath;
```

vga_raster.vhd:

```
--Legal Notice: (C)2007 Altera Corporation. All rights reserved.  Your
--use of Altera Corporation's design tools, logic functions and other
--software and tools, and its AMPP partner logic functions, and any
--output files any of the foregoing (including device programming or
--simulation files), and any associated documentation or information are
--expressly subject to the terms and conditions of the Altera Program
--License Subscription Agreement or other applicable license agreement,
--including, without limitation, that your use is for the sole purpose
--of programming logic devices manufactured by Altera and sold by Altera
--or its authorized distributors.  Please refer to the applicable
--agreement for further details.


-- turn off superfluous VHDL processor warnings
-- altera message_level Level1
-- altera message_off 10034 10035 10036 10037 10230 10240 10030

library altera;
use altera.altera_europa_support_lib.all;

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

```
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity vga_raster is
      port (
          -- inputs:
            signal address : IN STD_LOGIC_VECTOR (4 DOWNTO 0);
            signal chipselect : IN STD_LOGIC;
            signal clk : IN STD_LOGIC;
            signal read : IN STD_LOGIC;
            signal reset : IN STD_LOGIC;
            signal write : IN STD_LOGIC;
            signal writedata : IN STD_LOGIC_VECTOR (15 DOWNTO 0);

          -- outputs:
            signal VGA_B : OUT STD_LOGIC_VECTOR (9 DOWNTO 0);
            signal VGA_BLANK : OUT STD_LOGIC;
            signal VGA_CLK : OUT STD_LOGIC;
            signal VGA_G : OUT STD_LOGIC_VECTOR (9 DOWNTO 0);
            signal VGA_HS : OUT STD_LOGIC;
            signal VGA_R : OUT STD_LOGIC_VECTOR (9 DOWNTO 0);
            signal VGA_SYNC : OUT STD_LOGIC;
            signal VGA_VS : OUT STD_LOGIC;
            signal readdata : OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
          );
end entity vga_raster;


architecture europa of vga_raster is
component de2_vga_raster is
        port (
            -- inputs:
              signal address : IN STD_LOGIC_VECTOR (4 DOWNTO 0);
              signal chipselect : IN STD_LOGIC;
              signal clk : IN STD_LOGIC;
              signal read : IN STD_LOGIC;
              signal reset : IN STD_LOGIC;
              signal write : IN STD_LOGIC;
              signal writedata : IN STD_LOGIC_VECTOR (15 DOWNTO 0);

            -- outputs:
              signal VGA_B : OUT STD_LOGIC_VECTOR (9 DOWNTO 0);
              signal VGA_BLANK : OUT STD_LOGIC;
              signal VGA_CLK : OUT STD_LOGIC;
              signal VGA_G : OUT STD_LOGIC_VECTOR (9 DOWNTO 0);
              signal VGA_HS : OUT STD_LOGIC;
```

```
            signal VGA_R : OUT STD_LOGIC_VECTOR (9 DOWNTO 0);
            signal VGA_SYNC : OUT STD_LOGIC;
            signal VGA_VS : OUT STD_LOGIC;
            signal readdata : OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
          );
end component de2_vga_raster;

          signal internal_VGA_B :  STD_LOGIC_VECTOR (9 DOWNTO 0);
          signal internal_VGA_BLANK :  STD_LOGIC;
          signal internal_VGA_CLK :  STD_LOGIC;
          signal internal_VGA_G :  STD_LOGIC_VECTOR (9 DOWNTO 0);
          signal internal_VGA_HS :  STD_LOGIC;
          signal internal_VGA_R :  STD_LOGIC_VECTOR (9 DOWNTO 0);
          signal internal_VGA_SYNC :  STD_LOGIC;
          signal internal_VGA_VS :  STD_LOGIC;
          signal internal_readdata :  STD_LOGIC_VECTOR (15 DOWNTO 0);

begin

 --the_de2_vga_raster, which is an e_instance
 the_de2_vga_raster : de2_vga_raster
  port map(
    VGA_B => internal_VGA_B,
    VGA_BLANK => internal_VGA_BLANK,
    VGA_CLK => internal_VGA_CLK,
    VGA_G => internal_VGA_G,
    VGA_HS => internal_VGA_HS,
    VGA_R => internal_VGA_R,
    VGA_SYNC => internal_VGA_SYNC,
    VGA_VS => internal_VGA_VS,
    readdata => internal_readdata,
    address => address,
    chipselect => chipselect,
    clk => clk,
    read => read,
    reset => reset,
    write => write,
    writedata => writedata
  );


 --vhdl renameroo for output signals
 VGA_B <= internal_VGA_B;
 --vhdl renameroo for output signals
 VGA_BLANK <= internal_VGA_BLANK;
 --vhdl renameroo for output signals
```

```
 VGA_CLK <= internal_VGA_CLK;
 --vhdl renameroo for output signals
 VGA_G <= internal_VGA_G;
 --vhdl renameroo for output signals
 VGA_HS <= internal_VGA_HS;
 --vhdl renameroo for output signals
 VGA_R <= internal_VGA_R;
 --vhdl renameroo for output signals
 VGA_SYNC <= internal_VGA_SYNC;
 --vhdl renameroo for output signals
 VGA_VS <= internal_VGA_VS;
 --vhdl renameroo for output signals
 readdata <= internal_readdata;

end europa;
```

frog_controller.vhd:

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

entity frog_controller is
  port(
clk50   : in std_logic;
clk25   : in std_logic;
address: in unsigned (1 downto 0);
frog_alive : in unsigned (1 downto 0);
direction: in integer;
vertical: in integer;
horizontal: in integer;
frogflag : out std_logic;
pixel_R: out unsigned(3 downto 0);
pixel_G: out unsigned(3 downto 0);
pixel_B: out unsigned(3 downto 0)
  );
end frog_controller;

architecture rtl of frog_controller is

component frog_static_R_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned(3 downto 0)
  );
```

```
end component;

component frog_static_G_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned(3 downto 0)
  );
end component;

component frog_static_B_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned(3 downto 0)
  );
end component;

component frog_jump_R_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned(3 downto 0)
  );
end component;

component frog_jump_G_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned(3 downto 0)
  );
end component;

component frog_jump_B_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned(3 downto 0)
  );
end component;

component frog_dead1_R_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
```

```
data  : out unsigned(3 downto 0)
 );
end component;

component frog_dead1_G_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned(3 downto 0)
 );
end component;

component frog_dead1_B_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned(3 downto 0)
 );
end component;

component frog_dead2_R_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned(3 downto 0)
 );
end component;

component frog_dead2_G_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned(3 downto 0)
 );
end component;

component frog_dead2_B_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned(3 downto 0)
 );
end component;

signal index : unsigned ( 9 downto 0 );
signal index2 : unsigned ( 9 downto 0 );
```

```
signal frog_static_R,frog_static_G,frog_static_B : unsigned(3 downto 0);
signal frog_jump_R,frog_jump_G,frog_jump_B : unsigned(3 downto 0);
signal frog_dead1_R,frog_dead1_G,frog_dead1_B : unsigned(3 downto 0);
signal frog_dead2_R,frog_dead2_G,frog_dead2_B : unsigned(3 downto 0);

begin

frog_staticR: frog_static_R_ROM port map (clk50,index,frog_static_R);
frog_staticG: frog_static_G_ROM port map (clk50,index,frog_static_G);
frog_staticB: frog_static_B_ROM port map (clk50,index,frog_static_B);
frog_jumpR: frog_jump_R_ROM port map (clk50,index,frog_jump_R);
frog_jumpG: frog_jump_G_ROM port map (clk50,index,frog_jump_G);
frog_jumpB: frog_jump_B_ROM port map (clk50,index,frog_jump_B);
frog_dead1R: frog_dead1_R_ROM port map (clk50,index2,frog_dead1_R);
frog_dead1G: frog_dead1_G_ROM port map (clk50,index2,frog_dead1_G);
frog_dead1B: frog_dead1_B_ROM port map (clk50,index2,frog_dead1_B);
frog_dead2R: frog_dead2_R_ROM port map (clk50,index2,frog_dead2_R);
frog_dead2G: frog_dead2_G_ROM port map (clk50,index2,frog_dead2_G);
frog_dead2B: frog_dead2_B_ROM port map (clk50,index2,frog_dead2_B);

process(clk50)
begin
if rising_edge(clk50) then
case direction is
 when 2 =>   --left
        index <= to_unsigned ((horizontal * 32 + 31 - vertical), 10);
 when 3 =>   --down
        index <= to_unsigned (((31-vertical) * 32 + 31 - horizontal), 10);
 when 4 =>   --right
        index <= to_unsigned ((vertical + (31 - horizontal) * 32), 10);
 when others =>  --up
        index <= to_unsigned ((vertical * 32 + horizontal), 10);
end case;
   index2 <= to_unsigned ((vertical * 32 + horizontal), 10);
end if;
end process;

process(clk25)
begin
if rising_edge(clk25) then
if address = "00" then
frogflag <= '0';
else
case frog_alive is
 when "01" =>
```

```
          if frog_dead1_R = "0000" and frog_dead1_G = "0000" and frog_dead1_B =
"0000" then  --transparent the background
            frogflag <= '0';
          else   --display character
            frogflag <= '1';
            pixel_R <= frog_dead1_R;
            pixel_G <= frog_dead1_G;
            pixel_B <= frog_dead1_B;
   end if;
  when "10" =>
          if frog_dead2_R = "0000" and frog_dead2_G = "0000" and frog_dead2_B =
"0000" then  --transparent the background
            frogflag <= '0';
          else   --display character
            frogflag <= '1';
            pixel_R <= frog_dead2_R;
            pixel_G <= frog_dead2_G;
            pixel_B <= frog_dead2_B;
   end if;
  when others =>
case address is
  when "00" =>
          frogflag <= '0';
  when "01" =>   --when frog does not move
   if frog_static_R = "0000" and frog_static_G = "0000" and frog_static_B = "1111"
then  --transparent the background
            frogflag <= '0';
          else   --display character
            frogflag <= '1';
            pixel_R <= frog_static_R;
            pixel_G <= frog_static_G;
            pixel_B <= frog_static_B;
   end if;
  when "11" =>   --when frog jumps
   if frog_jump_R = "0000" and frog_jump_G = "0000" and frog_jump_B = "1111" then
--transparent the background
            frogflag <= '0';
          else   --display character
            frogflag <= '1';
            pixel_R <= frog_jump_R;
            pixel_G <= frog_jump_G;
            pixel_B <= frog_jump_B;
   end if;
  when others =>
          frogflag <= '0';
end case;
```

```
end case;
end if;
end if;
end process;

end rtl;
```

background_controller.vhd:

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

entity background_controller is
  port(
clk50   : in std_logic;
clk25   : in std_logic;
address: in unsigned (4 downto 0);
vertical: in integer;
horizontal: in integer;
backflag : out std_logic;
pixel_R: out unsigned (3 downto 0);
pixel_G: out unsigned (3 downto 0);
pixel_B: out unsigned (3 downto 0)
  );
end  background_controller;

architecture rtl of background_controller is


component startzone_R_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component startzone_G_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;
```

```
component startzone_B_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component roaddown_R_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component roaddown_G_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component roaddown_B_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component roadmain_R_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component roadmain_G_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
```

```
end component;

component roadmain_B_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component roadup_R_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component roadup_G_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component roadup_B_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component middlezone_R_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component middlezone_G_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
```

```
data  : out unsigned (3 downto 0)
 );
end component;

component middlezone_B_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component water_R_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component water_G_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component water_B_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component homeleft_R_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component homeleft_G_ROM
 port(
```

```
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component homeleft_B_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component homemiddle_R_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component homemiddle_G_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component homemiddle_B_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component homeright_R_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;
```

```
component homeright_G_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component homeright_B_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component homefrog_R_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component homefrog_G_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component homefrog_B_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component lawn_R_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
```

```
end component;

component lawn_G_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component lawn_B_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

signal index : unsigned ( 9 downto 0 );

signal startzone_R,startzone_G,startzone_B : unsigned (3 downto 0);
signal startzone_R_R,startzone_R_G,startzone_R_B : unsigned (3 downto 0);
signal roaddown_R,roaddown_G,roaddown_B : unsigned (3 downto 0);
signal roaddown_R_R,roaddown_R_G,roaddown_R_B : unsigned (3 downto 0);
signal roadmain_R,roadmain_G,roadmain_B : unsigned (3 downto 0);
signal roadmain_R_R,roadmain_R_G,roadmain_R_B : unsigned (3 downto 0);
signal roadup_R,roadup_G,roadup_B : unsigned (3 downto 0);
signal roadup_R_R,roadup_R_G,roadup_R_B : unsigned (3 downto 0);
signal middle_R,middle_G,middle_B : unsigned (3 downto 0);
signal middle_R_R,middle_R_G,middle_R_B : unsigned (3 downto 0);
signal water_R,water_G,water_B : unsigned (3 downto 0);
signal homeleft_R,homeleft_G,homeleft_B : unsigned (3 downto 0);
signal homemiddle_R,homemiddle_G,homemiddle_B : unsigned (3 downto 0);
signal homeright_R,homeright_G,homeright_B : unsigned (3 downto 0);
signal homefrog_R,homefrog_G,homefrog_B : unsigned (3 downto 0);
signal lawn_R,lawn_G,lawn_B : unsigned (3 downto 0);

begin

start_R: startzone_R_ROM port map (clk50,index,startzone_R);
start_G: startzone_G_ROM port map (clk50,index,startzone_G);
start_B: startzone_B_ROM port map (clk50,index,startzone_B);


roadd_R: roaddown_R_ROM port map (clk50,index,roaddown_R);
roadd_G: roaddown_G_ROM port map (clk50,index,roaddown_G);
```

```
roadd_B: roaddown_B_ROM port map (clk50,index,roaddown_B);


roadm_R: roadmain_R_ROM port map (clk50,index,roadmain_R);
roadm_G: roadmain_G_ROM port map (clk50,index,roadmain_G);
roadm_B: roadmain_B_ROM port map (clk50,index,roadmain_B);


roadu_R: roadup_R_ROM port map (clk50,index,roadup_R);
roadu_G: roadup_G_ROM port map (clk50,index,roadup_G);
roadu_B: roadup_B_ROM port map (clk50,index,roadup_B);


mid_R: middlezone_R_ROM port map (clk50,index,middle_R);
mid_G: middlezone_G_ROM port map (clk50,index,middle_G);
mid_B: middlezone_B_ROM port map (clk50,index,middle_B);

wat_R: water_R_ROM port map (clk50,index,water_R);
wat_G: water_G_ROM port map (clk50,index,water_G);
wat_B: water_B_ROM port map (clk50,index,water_B);

hleft_R: homeleft_R_ROM port map (clk50,index,homeleft_R);
hleft_G: homeleft_G_ROM port map (clk50,index,homeleft_G);
hleft_B: homeleft_B_ROM port map (clk50,index,homeleft_B);

hmiddle_R: homemiddle_R_ROM port map (clk50,index,homemiddle_R);
hmiddle_G: homemiddle_G_ROM port map (clk50,index,homemiddle_G);
hmiddle_B: homemiddle_B_ROM port map (clk50,index,homemiddle_B);

hright_R: homeright_R_ROM port map (clk50,index,homeright_R);
hright_G: homeright_G_ROM port map (clk50,index,homeright_G);
hright_B: homeright_B_ROM port map (clk50,index,homeright_B);

hfrog_R: homefrog_R_ROM port map (clk50,index,homefrog_R);
hfrog_G: homefrog_G_ROM port map (clk50,index,homefrog_G);
hfrog_B: homefrog_B_ROM port map (clk50,index,homefrog_B);

hlawn_R: lawn_R_ROM port map (clk50,index,lawn_R);
hlawn_G: lawn_G_ROM port map (clk50,index,lawn_G);
hlawn_B: lawn_B_ROM port map (clk50,index,lawn_B);


process(clk50)
begin
if rising_edge(clk50) then
if (address and "00001") = "00000" then
```

```
index <= to_unsigned ((vertical*32 + horizontal), 10);
else
index <= to_unsigned ((vertical*32 + (31 - horizontal)), 10);
end if;
end if;
end process;

process(clk25)
begin
if rising_edge(clk25) then


case address is
when "00000" =>
  backflag <= '1';
  pixel_R <= startzone_R ;
  pixel_G <= startzone_G ;
  pixel_B <= startzone_B ;

when "00001" =>
  backflag <='1';
  pixel_R <= startzone_R ;
  pixel_G <= startzone_G ;
  pixel_B <= startzone_B ;

when "00010" =>
  backflag <='1';
  pixel_R <= roaddown_R ;
  pixel_G <= roaddown_G ;
  pixel_B <= roaddown_B ;


when "00011" =>
  backflag <='1';
  pixel_R <= roaddown_R ;
  pixel_G <= roaddown_G ;
  pixel_B <= roaddown_B ;


when "00100" =>
  backflag <='1';
  pixel_R <= roadmain_R ;
  pixel_G <= roadmain_G ;
  pixel_B <= roadmain_B ;
```

```
when "00101" =>  --5
  backflag <='1';
  pixel_R <= roadmain_R ;
  pixel_G <= roadmain_G ;
  pixel_B <= roadmain_B ;


when "00110" =>  --6
  backflag <='1';
  pixel_R <= roadup_R ;
  pixel_G <= roadup_G ;
  pixel_B <= roadup_B ;


when "00111" =>  --7
  backflag <='1';
  pixel_R <= roadup_R ;
  pixel_G <= roadup_G ;
  pixel_B <= roadup_B ;


when "01000" =>  --8
  backflag <='1';
  pixel_R <= middle_R ;
  pixel_G <= middle_G ;
  pixel_B <= middle_B ;

when "01001" =>  --9
  backflag <='1';
  pixel_R <= middle_R ;
  pixel_G <= middle_G ;
  pixel_B <= middle_B ;

when "01010" =>  --10
  backflag <='1';
  pixel_R <= water_R ;
  pixel_G <= water_G ;
  pixel_B <= water_B ;

when "01011" =>  --11
  backflag <='1';
  pixel_R <= water_R ;
  pixel_G <= water_G ;
  pixel_B <= water_B ;

when "01100" =>  --12
```

```
  backflag <='1';
  pixel_R <= homeleft_R ;
  pixel_G <= homeleft_G ;
  pixel_B <= homeleft_B ;

when "01110" =>  --14
  backflag <='1';
  pixel_R <= homemiddle_R ;
  pixel_G <= homemiddle_G ;
  pixel_B <= homemiddle_B ;

when "10000" =>  --16
  backflag <='1';
  pixel_R <= homeright_R ;
  pixel_G <= homeright_G ;
  pixel_B <= homeright_B ;

when "10010" =>  --18
  backflag <='1';
  pixel_R <= homefrog_R ;
  pixel_G <= homefrog_G ;
  pixel_B <= homefrog_B ;

when "10100" =>  --20
  backflag <='1';
  pixel_R <= lawn_R ;
  pixel_G <= lawn_G ;
  pixel_B <= lawn_B ;

when "10101" =>  --21
  backflag <='1';
  pixel_R <= lawn_R ;
  pixel_G <= lawn_G ;
  pixel_B <= lawn_B ;

when others =>
  backflag <= '0';
 end case;
end if;
end process;


end rtl;
```

frogming_controller.vhd:

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

entity frogming_controller is
  port(
clk50   : in std_logic;
clk25   : in std_logic;
address: in unsigned (1 downto 0);
vertical: in integer;
horizontal: in integer;
smallflag : out std_logic;
pixel_R: out unsigned (3 downto 0);
pixel_G: out unsigned (3 downto 0);
pixel_B: out unsigned (3 downto 0)
  );
end  frogming_controller;

architecture rtl of frogming_controller is

component frogleft_R_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component frogleft_G_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component frogleft_B_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component frogright_R_ROM
```

```
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component frogright_G_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component frogright_B_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;


signal index : unsigned ( 9 downto 0 );

signal frogleft_R,frogleft_G,frogleft_B: unsigned (3 downto 0);
signal frogright_R,frogright_G,frogright_B : unsigned (3 downto 0);

begin

frogleftmap_R : frogleft_R_ROM port map (clk50, index, frogleft_R);
frogleftmap_G : frogleft_G_ROM port map (clk50, index, frogleft_G);
frogleftmap_B : frogleft_B_ROM port map (clk50, index, frogleft_B);

frogrightmap_R : frogright_R_ROM port map (clk50, index, frogright_R);
frogrightmap_G : frogright_G_ROM port map (clk50, index, frogright_G);
frogrightmap_B : frogright_B_ROM port map (clk50, index, frogright_B);


process(clk25)
begin
if rising_edge(clk25) then
index <= to_unsigned ((vertical * 8 + horizontal), 10);

case address is
```

```
when "00" =>
 smallflag <= '1';
  pixel_R <= "0000";
  pixel_G <= "0000";
  pixel_B <= "0000";
when "01" =>
  smallflag <= '1';
  pixel_R <= frogleft_R ;
  pixel_G <= frogleft_G ;
  pixel_B <= frogleft_B ;

when "10" =>
  smallflag <= '1';
  pixel_R <= frogright_R ;
  pixel_G <= frogright_G ;
  pixel_B <= frogright_B ;

when others =>
  smallflag <= '0';
 end case;
end if;
end process;

end rtl;
```

item_controller.vhd:

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

entity item_controller is
  port(
clk50   : in std_logic;
clk25   : in std_logic;
address: in unsigned (3 downto 0);
phase : in integer;
vertical: in integer;
horizontal: in integer;
itemflag : out std_logic;
pixel_R: out unsigned (3 downto 0);
pixel_G: out unsigned (3 downto 0);
pixel_B: out unsigned (3 downto 0)
  );
end item_controller;
```

```
architecture rtl of item_controller is

component car1_R_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component car1_G_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component car1_B_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component car2_R_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component car2_G_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component car2_B_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
```

```
data  : out unsigned (3 downto 0)
 );
end component;

component car3_R_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component car3_G_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component car3_B_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component car4_R_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component car4_G_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component car4_B_ROM
 port(
```

```
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component car5_1_R_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component car5_1_G_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component car5_1_B_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component car5_2_R_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component car5_2_G_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;
```

```
component car5_2_B_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component log_mid_R_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component log_mid_G_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component log_mid_B_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component log_head_R_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component log_head_G_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
```

```
end component;

component log_head_B_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component log_tail_R_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component log_tail_G_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component log_tail_B_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component turtle1_R_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end component;

component turtle1_G_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
```

```
data  : out unsigned (3 downto 0)
 );
end component;

component turtle1_B_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component turtle2_R_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component turtle2_G_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component turtle2_B_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component turtle3_R_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component turtle3_G_ROM
 port(
```

```
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

component turtle3_B_ROM
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end component;

--signal data_R:integer;
--signal data_G:integer;
--signal data_B:integer;
signal index : unsigned ( 9 downto 0 );
signal car1_R,car1_G,car1_B : unsigned (3 downto 0);
signal car2_R,car2_G,car2_B : unsigned (3 downto 0);
signal car3_R,car3_G,car3_B : unsigned (3 downto 0);
signal car4_R,car4_G,car4_B : unsigned (3 downto 0);
signal car5_1_R,car5_1_G,car5_1_B : unsigned (3 downto 0);
signal car5_2_R,car5_2_G,car5_2_B : unsigned (3 downto 0);
signal log_mid_R,log_mid_G,log_mid_B : unsigned (3 downto 0);
signal log_head_R,log_head_G,log_head_B : unsigned (3 downto 0);
signal log_tail_R,log_tail_G,log_tail_B : unsigned (3 downto 0);
signal turtle1_R,turtle1_G,turtle1_B : unsigned (3 downto 0);
signal turtle2_R,turtle2_G,turtle2_B : unsigned (3 downto 0);
signal turtle3_R,turtle3_G,turtle3_B : unsigned (3 downto 0);


begin

car1R: car1_R_ROM port map (clk50,index,car1_R);
car1G: car1_G_ROM port map (clk50,index,car1_G);
car1B: car1_B_ROM port map (clk50,index,car1_B);
car2R: car2_R_ROM port map (clk50,index,car2_R);
car2G: car2_G_ROM port map (clk50,index,car2_G);
car2B: car2_B_ROM port map (clk50,index,car2_B);
car3R: car3_R_ROM port map (clk50,index,car3_R);
car3G: car3_G_ROM port map (clk50,index,car3_G);
car3B: car3_B_ROM port map (clk50,index,car3_B);
car4R: car4_R_ROM port map (clk50,index,car4_R);
car4G: car4_G_ROM port map (clk50,index,car4_G);
car4B: car4_B_ROM port map (clk50,index,car4_B);
```

```
car5_1R: car5_1_R_ROM port map (clk50,index,car5_1_R);
car5_1G: car5_1_G_ROM port map (clk50,index,car5_1_G);
car5_1B: car5_1_B_ROM port map (clk50,index,car5_1_B);
car5_2R: car5_2_R_ROM port map (clk50,index,car5_2_R);
car5_2G: car5_2_G_ROM port map (clk50,index,car5_2_G);
car5_2B: car5_2_B_ROM port map (clk50,index,car5_2_B);
log_midR: log_mid_R_ROM port map (clk50,index,log_mid_R);
log_midG: log_mid_G_ROM port map (clk50,index,log_mid_G);
log_midB: log_mid_B_ROM port map (clk50,index,log_mid_B);
log_headR: log_head_R_ROM port map (clk50,index,log_head_R);
log_headG: log_head_G_ROM port map (clk50,index,log_head_G);
log_headB: log_head_B_ROM port map (clk50,index,log_head_B);
log_tailR: log_tail_R_ROM port map (clk50,index,log_tail_R);
log_tailG: log_tail_G_ROM port map (clk50,index,log_tail_G);
log_tailB: log_tail_B_ROM port map (clk50,index,log_tail_B);
turtle1R: turtle1_R_ROM port map (clk50,index,turtle1_R);
turtle1G: turtle1_G_ROM port map (clk50,index,turtle1_G);
turtle1B: turtle1_B_ROM port map (clk50,index,turtle1_B);
turtle2R: turtle2_R_ROM port map (clk50,index,turtle2_R);
turtle2G: turtle2_G_ROM port map (clk50,index,turtle2_G);
turtle2B: turtle2_B_ROM port map (clk50,index,turtle2_B);
turtle3R: turtle3_R_ROM port map (clk50,index,turtle3_R);
turtle3G: turtle3_G_ROM port map (clk50,index,turtle3_G);
turtle3B: turtle3_B_ROM port map (clk50,index,turtle3_B);

process(clk25)
begin
if rising_edge(clk25) then
index <= to_unsigned ((vertical*32 + horizontal), 10);
case address is
when "0000" =>
  itemflag <= '0';
when "0001" =>
if car1_R ="0000" and car1_G  ="0000" and car1_B  ="0000" then
  itemflag <= '0';
else
  itemflag <='1';
  pixel_R <= car1_R ;
  pixel_G <= car1_G ;
  pixel_B <= car1_B ;
end if;
when "0010" =>
if log_head_R ="0000" and log_head_G  = "0000" and log_head_B  = "0000" then
  itemflag <= '0';
else
  itemflag <='1';
```

```
  pixel_R <= log_head_R ;
  pixel_G <= log_head_G ;
  pixel_B <= log_head_B ;
end if;
when "0011" =>
if log_mid_R ="0000" and log_mid_G  = "0000" and log_mid_B  = "0000" then
  itemflag <= '0';
else
  itemflag <='1';
pixel_R <= log_mid_R ;
pixel_G <= log_mid_G ;
pixel_B <= log_mid_B ;
end if;
when "0100" =>
if log_tail_R ="0000" and log_tail_G  = "0000" and log_tail_B  = "0000" then
  itemflag <= '0';
else
  itemflag <='1';
pixel_R <= log_tail_R ;
pixel_G <= log_tail_G ;
pixel_B <= log_tail_B ;
end if;
when "0101" =>
case phase is
when 1 =>
if turtle1_R = "0000" and turtle1_G  = "0000" and turtle1_B  = "0000" then
  itemflag <= '0';
else
  itemflag <='1';
pixel_R <= turtle1_R ;
pixel_G <= turtle1_G ;
pixel_B <= turtle1_B ;
end if;
when 2 =>
if turtle2_R ="0000" and turtle2_G  = "0000" and turtle2_B  = "0000" then
  itemflag <= '0';
else
  itemflag <='1';
pixel_R <= turtle2_R ;
pixel_G <= turtle2_G ;
pixel_B <= turtle2_B ;
end if;
when 3 =>
if turtle3_R ="0000" and turtle3_G  ="0000" and turtle3_B  = "0000" then
  itemflag <= '0';
else
```

```
 itemflag <='1';
pixel_R <= turtle3_R ;
pixel_G <= turtle3_G ;
pixel_B <= turtle3_B ;
end if;
when others =>
 itemflag <= '0';
end case;
when "0110" =>
if car2_R ="0000" and car2_G  = "0000" and car2_B  = "0000" then
 itemflag <= '0';
else
 itemflag <='1';
 pixel_R <= car2_R ;
 pixel_G <= car2_G ;
 pixel_B <= car2_B ;
end if;
when "0111" =>
if car3_R ="0000" and car3_G  = "0000" and car3_B  = "0000" then
 itemflag <= '0';
else
 itemflag <='1';
 pixel_R <= car3_R ;
 pixel_G <= car3_G ;
 pixel_B <= car3_B ;
end if;
when "1000" =>
if car4_R ="0000" and car4_G  = "0000" and car4_B  = "0000" then
 itemflag <= '0';
else
 itemflag <='1';
 pixel_R <= car4_R ;
 pixel_G <= car4_G ;
 pixel_B <= car4_B ;
end if;
when "1001" =>
if car5_1_R ="0000" and car5_1_G  = "0000" and car5_1_B  = "0000" then
 itemflag <= '0';
else
 itemflag <='1';
 pixel_R <= car5_1_R ;
 pixel_G <= car5_1_G ;
 pixel_B <= car5_1_B ;
end if;
when "1010" =>
if car5_2_R ="0000" and car5_2_G  = "0000" and car5_2_B  = "0000" then
```

```
 itemflag <= '0';
else
 itemflag <='1';
 pixel_R <= car5_2_R ;
 pixel_G <= car5_2_G ;
 pixel_B <= car5_2_B ;
end if;
when others =>
 itemflag <= '0';
end case;
end if;
end process;


end rtl;
```

function.vhd:

```
--
-- Copyright (c)2002 Flarion Technologies. All Rights Reserved.
--
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
--use ieee.std_logic_arith.all;
use ieee.std_logic_misc.all;

package functions is
-- signed / unsigned shifts
function s_shr (v : std_logic_vector; s : integer) return std_logic_vector;

end functions;

package body functions is
-- sign shift right (arithmetic)
function s_shr (v : std_logic_vector; s : integer) return std_logic_vector is
variable r : std_logic_vector (v'range);
begin
case s is
when 0 => r := v;
when 1 => r (v'left - 1 downto 0)                    := v (v'left downto 1);
 for i in v'left downto v'left - 1 + 1 loop
r (i) := v (v'left);
 end loop;
```

```
when 2 => r (v'left - 2 downto 0)                    := v (v'left downto 2);
  for i in v'left downto v'left - 2 + 1 loop
r (i) := v (v'left);
  end loop;
when 3 => r (v'left - 3 downto 0)                    := v (v'left downto 3);
  for i in v'left downto v'left - 3 + 1 loop
r (i) := v (v'left);
  end loop;
when others => null;
end case;
return r;
end;
end functions;
```

number_controller.vhd:

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

entity number_controller is
  port(
clk50   : in std_logic;
clk25   : in std_logic;
address: in unsigned (5 downto 0);
vertical: in integer;
horizontal: in integer;
numflag : out std_logic;
charc : out unsigned (9 downto 0)
  );
end number_controller;

architecture rtl of number_controller is

component number_all_ROM
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (7 downto 0)
  );
end component;

signal index : unsigned (9 downto 0);
signal pilenum : integer;
signal data : unsigned (7 downto 0);
signal data_tmp: unsigned (7 downto 0);
```

```
begin
 data_out: number_all_ROM port map (clk50,index,data);

process(clk25)
begin

if rising_edge(clk25) then
  pilenum <= to_integer(address);
  index <= to_unsigned (vertical + pilenum * 16,10);
  data_tmp <= (data sll (horizontal - 1)) and x"80";
 if pilenum > 36 then
   numflag <= '0';
 else
   if data_tmp = "00000000" then
                    numflag <= '0';
   else
                    numflag <= '1';
                    charc <= "1111111111";
          end if;
 end if;
end if;
end process;


end rtl;
```

sram.vhd:

```
--Legal Notice: (C)2007 Altera Corporation. All rights reserved.  Your
--use of Altera Corporation's design tools, logic functions and other
--software and tools, and its AMPP partner logic functions, and any
--output files any of the foregoing (including device programming or
--simulation files), and any associated documentation or information are
--expressly subject to the terms and conditions of the Altera Program
--License Subscription Agreement or other applicable license agreement,
--including, without limitation, that your use is for the sole purpose
--of programming logic devices manufactured by Altera and sold by Altera
--or its authorized distributors.  Please refer to the applicable
--agreement for further details.


-- turn off superfluous VHDL processor warnings
```

```
-- altera message_level Level1
-- altera message_off 10034 10035 10036 10037 10230 10240 10030

library altera;
use altera.altera_europa_support_lib.all;

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity sram is
      port (
          -- inputs:
            signal address : IN STD_LOGIC_VECTOR (17 DOWNTO 0);
            signal byteenable : IN STD_LOGIC_VECTOR (1 DOWNTO 0);
            signal chipselect : IN STD_LOGIC;
            signal read : IN STD_LOGIC;
            signal write : IN STD_LOGIC;
            signal writedata : IN STD_LOGIC_VECTOR (15 DOWNTO 0);

          -- outputs:
            signal SRAM_ADDR : OUT STD_LOGIC_VECTOR (17 DOWNTO 0);
            signal SRAM_CE_N : OUT STD_LOGIC;
            signal SRAM_DQ : INOUT STD_LOGIC_VECTOR (15 DOWNTO 0);
            signal SRAM_LB_N : OUT STD_LOGIC;
            signal SRAM_OE_N : OUT STD_LOGIC;
            signal SRAM_UB_N : OUT STD_LOGIC;
            signal SRAM_WE_N : OUT STD_LOGIC;
            signal readdata : OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
          );
end entity sram;


architecture europa of sram is
component de2_sram_controller is
        port (
            -- inputs:
              signal address : IN STD_LOGIC_VECTOR (17 DOWNTO 0);
              signal byteenable : IN STD_LOGIC_VECTOR (1 DOWNTO 0);
              signal chipselect : IN STD_LOGIC;
              signal read : IN STD_LOGIC;
              signal write : IN STD_LOGIC;
              signal writedata : IN STD_LOGIC_VECTOR (15 DOWNTO 0);

            -- outputs:
```

```
            signal SRAM_ADDR : OUT STD_LOGIC_VECTOR (17 DOWNTO 0);
            signal SRAM_CE_N : OUT STD_LOGIC;
            signal SRAM_DQ : INOUT STD_LOGIC_VECTOR (15 DOWNTO 0);
            signal SRAM_LB_N : OUT STD_LOGIC;
            signal SRAM_OE_N : OUT STD_LOGIC;
            signal SRAM_UB_N : OUT STD_LOGIC;
            signal SRAM_WE_N : OUT STD_LOGIC;
            signal readdata : OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
          );
end component de2_sram_controller;

          signal internal_SRAM_ADDR :  STD_LOGIC_VECTOR (17 DOWNTO 0);
          signal internal_SRAM_CE_N :  STD_LOGIC;
          signal internal_SRAM_LB_N :  STD_LOGIC;
          signal internal_SRAM_OE_N :  STD_LOGIC;
          signal internal_SRAM_UB_N :  STD_LOGIC;
          signal internal_SRAM_WE_N :  STD_LOGIC;
          signal internal_readdata :  STD_LOGIC_VECTOR (15 DOWNTO 0);

begin

 --the_de2_sram_controller, which is an e_instance
 the_de2_sram_controller : de2_sram_controller
  port map(
    SRAM_ADDR => internal_SRAM_ADDR,
    SRAM_CE_N => internal_SRAM_CE_N,
    SRAM_DQ => SRAM_DQ,
    SRAM_LB_N => internal_SRAM_LB_N,
    SRAM_OE_N => internal_SRAM_OE_N,
    SRAM_UB_N => internal_SRAM_UB_N,
    SRAM_WE_N => internal_SRAM_WE_N,
    readdata => internal_readdata,
    address => address,
    byteenable => byteenable,
    chipselect => chipselect,
    read => read,
    write => write,
    writedata => writedata
  );


 --vhdl renameroo for output signals
 SRAM_ADDR <= internal_SRAM_ADDR;
 --vhdl renameroo for output signals
 SRAM_CE_N <= internal_SRAM_CE_N;
 --vhdl renameroo for output signals
```

```
 SRAM_LB_N <= internal_SRAM_LB_N;
 --vhdl renameroo for output signals
 SRAM_OE_N <= internal_SRAM_OE_N;
 --vhdl renameroo for output signals
 SRAM_UB_N <= internal_SRAM_UB_N;
 --vhdl renameroo for output signals
 SRAM_WE_N <= internal_SRAM_WE_N;
 --vhdl renameroo for output signals
 readdata <= internal_readdata;

end europa;
```

number_all_ROM.vhd:

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

entity number_all_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned(7 downto 0)
 );
end number_all_ROM;

architecture rtl of number_all_ROM is
type rom_type is array(0 to 591) of unsigned(7 downto 0);
 constant ROM: rom_type := (
x"00", x"00", x"7c", x"c6", x"ce", x"ce", x"d6", x"d6", x"e6", x"e6", x"c6", x"7c", x"00",
x"00", x"00", x"00",--0
x"00", x"00", x"18", x"38", x"78", x"18", x"18", x"18", x"18", x"18",x"18", x"7e", x"00",
x"00", x"00", x"00",--1
x"00", x"00", x"7c", x"c6", x"06", x"0c", x"18", x"30", x"60", x"c0", x"c6", x"fe", x"00",
x"00", x"00", x"00",--2
x"00", x"00", x"7c", x"c6", x"06", x"06", x"3c", x"06", x"06", x"06", x"c6", x"7c", x"00",
x"00", x"00", x"00", --3
x"00", x"00", x"0c", x"1c", x"3c", x"6c", x"cc", x"fe", x"0c", x"0c", x"0c", x"1e", x"00",
x"00", x"00", x"00", --4
x"00", x"00", x"fe", x"c0", x"c0", x"c0", x"fc", x"06", x"06", x"06", x"c6", x"7c", x"00",
x"00", x"00", x"00",--5
x"00", x"00", x"38", x"60", x"c0", x"c0", x"fc", x"c6", x"c6", x"c6", x"c6", x"7c", x"00",
x"00", x"00", x"00",--6
x"00", x"00", x"fe", x"c6", x"06", x"06", x"0c", x"18", x"30", x"30", x"30", x"30", x"00",
x"00", x"00", x"00",--7
```

x"00", x"00", x"7c", x"c6", x"c6", x"c6", x"7c", x"c6", x"c6", x"c6", x"c6", x"7c", x"00",
x"00", x"00", x"00",--8
x"00", x"00", x"7c", x"c6", x"c6", x"c6", x"7e", x"06", x"06", x"06", x"0c", x"78", x"00",
x"00", x"00", x"00",--9

x"00", x"00", x"10", x"38", x"6c", x"c6", x"c6", x"fe", x"c6", x"c6", x"c6", x"c6", x"00",
x"00", x"00", x"00", --A 10
x"00", x"00", x"fc", x"66", x"66", x"66", x"7c", x"66", x"66", x"66", x"66", x"fc", x"00",
x"00", x"00", x"00",--B 11
x"00", x"00", x"3c", x"66", x"c2", x"c0", x"c0", x"c0", x"c0", x"c2", x"66", x"3c", x"00",
x"00", x"00", x"00",--C 12
x"00", x"00", x"f8", x"6c", x"66", x"66", x"66", x"66", x"66", x"66", x"6c", x"f8", x"00",
x"00", x"00", x"00",--D 13
x"00", x"00", x"fe", x"66", x"62", x"68", x"78", x"68", x"60", x"62", x"66", x"fe", x"00",
x"00", x"00", x"00",--E 14
x"00", x"00", x"fe", x"66", x"62", x"68", x"78", x"68", x"60", x"60", x"60", x"f0", x"00",
x"00", x"00", x"00",--F 15
x"00", x"00", x"3c", x"66", x"c2", x"c0", x"c0", x"de", x"c6", x"c6", x"66", x"3a", x"00",
x"00", x"00", x"00",--G 16
x"00", x"00", x"c6", x"c6", x"c6", x"c6", x"fe", x"c6", x"c6", x"c6", x"c6", x"c6", x"00",
x"00", x"00", x"00",--H 17
x"00", x"00", x"3c", x"18", x"18", x"18", x"18", x"18", x"18", x"18", x"18", x"3c", x"00",
x"00", x"00", x"00",--I 18
x"00", x"00", x"1e", x"0c", x"0c", x"0c", x"0c", x"0c", x"cc", x"cc", x"cc", x"78", x"00",
x"00", x"00", x"00",--J 19
x"00", x"00", x"e6", x"66", x"66", x"6c", x"78", x"78", x"6c", x"66", x"66", x"e6", x"00",
x"00", x"00", x"00",--K 20
x"00", x"00", x"f0", x"60", x"60", x"60", x"60", x"60", x"60", x"62", x"66", x"fe", x"00",
x"00", x"00", x"00",--L 21
x"00", x"00", x"c6", x"ee", x"fe", x"fe", x"d6", x"c6", x"c6", x"c6", x"c6", x"c6", x"00",
x"00", x"00", x"00",--M 22
x"00", x"00", x"c6", x"e6", x"f6", x"fe", x"de", x"ce", x"c6", x"c6", x"c6", x"c6", x"00",
x"00", x"00", x"00",--N 23
x"00", x"00", x"7c", x"c6", x"c6", x"c6", x"c6", x"c6", x"c6", x"c6", x"c6", x"7c", x"00",
x"00", x"00", x"00",--O 24
x"00", x"00", x"fc", x"66", x"66", x"66", x"7c", x"60", x"60", x"60", x"60", x"f0", x"00",
x"00", x"00", x"00",--P 25
x"00", x"00", x"7c", x"c6", x"c6", x"c6", x"c6", x"c6", x"c6", x"d6", x"de", x"7c", x"00",
x"00", x"00", x"00",--Q 26
x"00", x"00", x"fc", x"66", x"66", x"66", x"7c", x"6c", x"66", x"66", x"66", x"e6", x"00",
x"00", x"00", x"00",--R 27
x"00", x"00", x"7c", x"c6", x"c6", x"60", x"38", x"0c", x"06", x"c6", x"c6", x"7c", x"00",
x"00", x"00", x"00",--S 28
x"00", x"00", x"7e", x"7e", x"5a", x"18", x"18", x"18", x"18", x"18", x"18", x"3c", x"00",
x"00", x"00", x"00",--T 29

```
x"00", x"00", x"c6", x"c6", x"c6", x"c6", x"c6", x"c6", x"c6", x"c6", x"c6", x"7c", x"00",
x"00", x"00", x"00",--U 30
x"00", x"00", x"c6", x"c6", x"c6", x"c6", x"c6", x"c6", x"c6", x"6c", x"38", x"10", x"00",
x"00", x"00", x"00",--V 31
x"00", x"00", x"c6", x"c6", x"c6", x"c6", x"d6", x"d6", x"d6", x"fe", x"ee", x"6c", x"00",
x"00", x"00", x"00",--W 32
x"00", x"00", x"c6", x"c6", x"6c", x"7c", x"38", x"38", x"7c", x"6c", x"c6", x"c6", x"00",
x"00", x"00", x"00",--X 33
x"00", x"00", x"66", x"66", x"66", x"66", x"3c", x"18", x"18", x"18", x"18", x"3c", x"00",
x"00", x"00", x"00",--Y 34
x"00", x"00", x"fe", x"c6", x"86", x"0c", x"18", x"30", x"60", x"c2", x"c6", x"fe", x"00",
x"00", x"00", x"00",--Z 35
x"00", x"00", x"00", x"00", x"00", x"00", x"00", x"00", x"00", x"00", x"00", x"00",
x"00", x"00", x"00", x"00" --nothing displayed 36
);

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

de2_vga_raster.vhd:

```
--------------------------------------------------------------------------------
--
-- VGA display main code
-- Author: Xin Zhang xz2270@columbia.edu
--                    Chenxi Liu cl2985@columbia.edu
--          Shengzhen Li sl3356@columbia.edu
-- Acknowledgement: Stephen A. Edwards sedwards@cs.columbia.edu
--
--------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.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 std_logic_vector(9 downto 0); -- Blue[9:0]

   read      : in  std_logic;
   write     : in  std_logic;
   chipselect : in  std_logic;
   address    : in  unsigned(4 downto 0);
   readdata   : out signed(15 downto 0);
   writedata  : in  signed(15 downto 0)
);

end de2_vga_raster;

architecture rtl of de2_vga_raster is

-- components declaration for frog, item, number, background, frogming
component item_controller
  port(
clk50   : in std_logic;
clk25   : in std_logic;
address: in unsigned (3 downto 0);
phase : in integer;
vertical: in integer;
horizontal: in integer;
itemflag : out std_logic;
pixel_R: out unsigned(3 downto 0);
pixel_G: out unsigned(3 downto 0);
pixel_B: out unsigned(3 downto 0)
  );
end component;

component frog_controller
  port(
clk50   : in std_logic;
clk25   : in std_logic;
address: in unsigned (1 downto 0);
frog_alive : in unsigned (1 downto 0);
direction: in integer;
```

```
vertical: in integer;
horizontal: in integer;
frogflag : out std_logic;
pixel_R: out unsigned(3 downto 0);
pixel_G: out unsigned(3 downto 0);
pixel_B: out unsigned(3 downto 0)
 );
end component;

component number_controller is
 port(
clk50   : in std_logic;
clk25   : in std_logic;
address: in unsigned (5 downto 0);
vertical: in integer;
horizontal: in integer;
numflag : out std_logic;
charc : out unsigned (9 downto 0)
 );
end component;

component background_controller is
 port(
clk50   : in std_logic;
clk25   : in std_logic;
address: in unsigned (4 downto 0);
vertical: in integer;
horizontal: in integer;
backflag : out std_logic;
pixel_R: out unsigned (3 downto 0);
pixel_G: out unsigned (3 downto 0);
pixel_B: out unsigned (3 downto 0)
 );
end  component;

component frogming_controller is
 port(
clk50   : in std_logic;
clk25   : in std_logic;
address: in unsigned (1 downto 0);
vertical: in integer;
horizontal: in integer;
smallflag : out std_logic;
pixel_R: out unsigned (3 downto 0);
pixel_G: out unsigned (3 downto 0);
pixel_B: out unsigned (3 downto 0)
```

```
  );
end  component;

  -- Video parameters

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

  constant VTOTAL      : integer := 525;
  constant VSYNC       : integer := 2;
  constant VBACK_PORCH  : integer := 33;
  constant VACTIVE     : integer := 480;
  constant VFRONT_PORCH : integer := 10;


  -- Horizontal and vertical 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 Hscreen : integer;
  signal Vscreen : integer;
  signal time_vertical : integer;
  signal time_horizontal : integer;
  signal Htime : integer;
  signal Vtime : integer;
  signal hpile : integer := 0;
  signal vpile : integer := 0;
  signal num_horizontal : integer;
  signal num_vertical : integer;
  signal horizontal : integer;
  signal vertical : integer;
  signal frog_horizontal : integer;
  signal frog_vertical : integer;
  signal frog_x : integer := 224;
  signal frog_y : integer := 416;
  signal Hnumber : integer;
  signal Vnumber : integer;
  signal Hback : integer;
  signal vga_hblank, vga_hsync,
    vga_vblank, vga_vsync : std_logic;  -- Sync. signals

  -- Signals for display items
  -- different flags for display
```

```
signal frogflag : std_logic ;
signal itemflag : std_logic ;
signal backflag : std_logic ;
signal numflag : std_logic;
signal deadflag : std_logic := '0';
signal smallflag : std_logic := '0';
signal timeflag : std_logic := '0';
signal jiamingflag : std_logic := '0';

-- different color parameters as outputs from different ROMs
signal color_R : unsigned(3 downto 0);
signal color_G : unsigned(3 downto 0);
signal color_B : unsigned(3 downto 0);

signal frog_R : unsigned(3 downto 0);
signal frog_G : unsigned(3 downto 0);
signal frog_B : unsigned(3 downto 0);

signal icolor_R : integer;
signal icolor_G : integer;
signal icolor_B : integer;

signal back_R : unsigned(3 downto 0);
signal back_G : unsigned(3 downto 0);
signal back_B : unsigned(3 downto 0);

signal small_R : unsigned(3 downto 0);
signal small_G : unsigned(3 downto 0);
signal small_B : unsigned(3 downto 0);

signal charc : unsigned(9 downto 0);

-- pattern number of different matrices
signal patternNum : integer := 0;
signal back_num : integer;
signal small_num : integer;

-- corresponding input addresses to controllers
signal sel_addr : unsigned (3 downto 0);
signal sel_frog : unsigned (1 downto 0);
signal sel_back : unsigned (4 downto 0);
signal sel_small : unsigned (1 downto 0);
signal sel_num_addr : unsigned (5 downto 0);

-- declaration of different tile matrices
type pattern_offset is array (0 to 14) of integer;
```

```
 type pattern is array(integer range 0 to 14, integer range 0 to 15) of integer;
 type scorepattern is array(integer range 0 to 1, integer range 0 to 10) of integer;
 type timepattern is array(0 to 7) of integer;

 signal offset : pattern_offset := (0,32,0,0,0,0,0,0,0,0,0,0,0,0,0);
 signal frog_offset : pattern_offset := (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
 signal speed : pattern_offset := (1,1,20,-4,60,-3,3,1,-3,4,-3,3,-4,1,1);

 constant river_pattern: pattern := (
( 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
( 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
( 4,3,3,3,2,4,3,3,3,3,2,4,3,3,3,2),
( 0,5,5,0,0,0,5,5,5,0,0,5,5,5,0,0),
( 0,0,0,0,4,3,3,2,0,0,0,4,3,3,2,0),
( 0,0,0,5,5,5,0,0,0,0,0,5,5,5,0,0),
( 4,2,0,0,0,4,3,2,0,0,0,4,3,2,0,0),
( 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
( 0,0,9,10,0,0,0,9,10,0,0,0,9,10,0,0),
( 0,8,0,0,0,8,0,0,0,0,0,8,0,0,0,0),
( 7,0,0,0,0,0,0,0,7,0,0,0,0,0,7,0),
( 0,0,6,0,0,0,0,0,0,6,0,0,0,0,6,0),
( 1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0),
( 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
( 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
);

 signal background: pattern := (
( 20,21,20,21,20,21,20,21,20,21,20,21,20,21,20,21),
( 12,14,16,12,14,16,12,14,16,12,14,16,12,14,16,12),
( 10,11,10,11,10,11,10,11,10,11,10,11,10,11,10,11),
( 10,11,10,11,10,11,10,11,10,11,10,11,10,11,10,11),
( 10,11,10,11,10,11,10,11,10,11,10,11,10,11,10,11),
( 10,11,10,11,10,11,10,11,10,11,10,11,10,11,10,11),
( 10,11,10,11,10,11,10,11,10,11,10,11,10,11,10,11),
( 8,9,8,9,8,9,8,9,8,9,8,9,8,9,8,9),
( 6,7,6,7,6,7,6,7,6,7,6,7,6,7,6,7),
( 4,5,4,5,4,5,4,5,4,5,4,5,4,5,4,5),
( 4,5,4,5,4,5,4,5,4,5,4,5,4,5,4,5),
( 4,5,4,5,4,5,4,5,4,5,4,5,4,5,4,5),
( 2,3,2,3,2,3,2,3,2,3,2,3,2,3,2,3),
( 0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1),
( 30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30)
);


----characters mapping to pattern numbers
```

```
----A  B C D E F G H I J K L M N O P Q R S T U V W  X Y Z
----10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35

signal score_matrix : scorepattern := (
(36, 34, 24, 30, 27, 36, 28, 12, 24, 27, 14),
(36, 36, 36,  0,  0,  0,  0,  0, 36, 36, 36)
);

signal high_matrix : scorepattern := (
(36, 17, 18, 16, 17, 36, 28, 12, 24, 27, 14),
(36, 36, 36,  0,  0,  0,  0,  0, 36, 36, 36)
);

signal life_left : timepattern := (
(36, 36, 36, 21, 18, 15, 14, 36)
);
----------------------------------------
signal frogming : scorepattern := (
(0, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2),
(0, 0, 0,  0,  0,  0,  0,  0, 0, 0, 0)
);
----------------------------------------
signal time_left : timepattern := (
36, 36, 29, 18, 22, 14, 36, 36
);

signal start_matrix : scorepattern := (
(36, 36, 36, 25, 27, 14, 28, 28, 36, 36, 36),
(36, 36, 36, 28, 25, 10, 12, 14, 36, 36, 36)
);

signal level_matrix : scorepattern :=(
(36, 36, 36, 21, 14, 31, 14, 21, 36, 36, 36),
(36, 36, 36, 36, 36, 0, 36, 36, 36, 36, 36)
);



  -- clk signals and count parameters
  signal clk25 : std_logic;
  signal clk_move : std_logic;
  signal counter : integer := 0;
  signal countermax : integer := 125000;
  signal count : integer := 0;
  signal i : integer;
  signal j : integer :=2;
```

```
  signal k : integer :=1;
  signal q : integer :=1;
  signal phase : integer := 1;
  signal pause : integer := 1;

  -- signals for frog actions
  signal frog_direction : integer := 1;
  signal frog_alive : unsigned (1 downto 0) := "00";
  signal frog_action : integer := 0;

  -- signals for game status
  signal timerange : integer := 300;
  signal frog_life : integer := 5;
  signal score_number : integer;
  signal score : integer := 0;
  signal winnum : integer := 0;
  signal highscore : integer := 0;

begin

  -- controller mapping
  ROM : item_controller port map (clk,clk25,
sel_addr,phase,vertical,horizontal,itemflag,color_R,color_G,color_B);
  frog_ROM : frog_controller port map (clk,clk25,
sel_frog,frog_alive,frog_direction,frog_vertical,frog_horizontal,frogflag,frog_R,frog_G,f
rog_B);
  number_ROM : number_controller port map (clk, clk25, sel_num_addr,
num_vertical, num_horizontal, numflag,charc);
  back : background_controller port map (clk, clk25, sel_back, vertical, Hback,
backflag, back_R, back_G, back_B);
  ming : frogming_controller port map (clk, clk25, sel_small, num_vertical,
num_horizontal, smallflag, small_R, small_G, small_B);

  -- 25MHz generation
  process (clk)
  begin
    if rising_edge(clk) then
      clk25 <= not clk25;
    end if;
  end process;

 -- counter for clk-move generation in order to control item movement
  process (clk25)
  begin
          if rising_edge (clk25) then
            if counter  >= countermax - 30000 * (winnum/5) then
```

```
                counter <= 0;
                clk_move <= not clk_move;
            else
                counter <= counter + 1;
            end if;
          end if;
 end process;

-- bottom time bar count down
Timecount: process (clk)
        begin
                if rising_edge (clk) then
                        if pause = 0 then
                   if frog_alive = "11" then
                                count <= 1;
                                timeflag <= '0';
                        elsif count = 2500000 then
                                count <= 1;
                                timeflag <= '1';
                        else
                                timeflag <= '0';
                                count <= count + 1;
                        end if;
                        end if;
                end if;
end process Timecount;

-- communication between hardware and software via Avalon bus
  Key: process (clk)
   begin
        if rising_edge(clk) then
                if chipselect = '1'  then
                        if write = '1' then
                                -- send current position, direction, action of the frog
                                if to_integer(address) = 20  then
                                        frog_x <= to_integer(writedata) -
frog_offset(frog_y/32);
                                elsif to_integer(address) = 21  then
                                        frog_y <= to_integer(writedata);
                                elsif to_integer(address) = 22  then
                                        frog_direction <= to_integer(writedata);
                                elsif to_integer(address) = 23  then
                                        frog_action <= to_integer(writedata);
                           -- receive pause signal
                                elsif to_integer(address) = 25  then
                                        pause <= to_integer(writedata);
```

```
                -- display the score
                elsif to_integer(address) = 26 then
                        score <= to_integer(writedata);
                        score_matrix(1, 7) <= score mod 10;
                        score_matrix(1, 6) <= (score mod 100)/10;
                        score_matrix(1, 5) <= (score mod 1000)/100;
                        score_matrix(1, 4) <= (score mod 10000)/1000;
                        if score >= highscore then
                                highscore <= score;
                                high_matrix(1, 7) <= score_matrix(1, 7);
                                high_matrix(1, 6) <= score_matrix(1, 6);
                                high_matrix(1, 5) <= score_matrix(1, 5);
                                high_matrix(1, 4) <= score_matrix(1, 4);

                        end if;
                --detect whether the frog goes home, and the
background matrix may change
                elsif to_integer(address) = 28  then
                        if background(1, (to_integer(writedata)+16)/32)
= 14 then
                                background(1,
(to_integer(writedata)+16)/32) <= 18;
                                winnum <= winnum + 1;
                                timerange <= 300;
                                if ((winnum + 1) mod 5) = 0 then
                                        background(1, 1) <= 14;
                                        background(1, 4) <= 14;
                                        background(1, 7) <= 14;
                                        background(1, 10) <= 14;
                                        background(1, 13) <= 14;
                                        jiamingflag <= '1';
                                end if;
                                frog_x <= 224;
                                frog_y <= 416;
                                frog_direction <= 1;
                                frog_action <= 0;
                        else
                                deadflag <= '1';
                                frog_action <= 0;
                        end if;
                elsif to_integer(address) = 29  then
                        if to_integer(writedata) = 1 then
                                background(1, 1) <= 14;
                                background(1, 4) <= 14;
                                background(1, 7) <= 14;
                                background(1, 10) <= 14;
```

```
                                        background(1, 13) <= 14;
                                        winnum <= 0;
                                        timerange <= 300;

                                end if;
                        end if;
                elsif read = '1' then
                        if to_integer(address) = 20 then
                                readdata <= to_signed((frog_x +
frog_offset(frog_y/32)), 16);
                        elsif to_integer(address) = 21 then
                                readdata <= to_signed(frog_y, 16);
                        elsif to_integer(address) = 24 then
                               readdata <= to_signed(to_integer(frog_alive), 16);
                        elsif to_integer(address) = 25 then
                                readdata <= to_signed(pause, 16);
                        elsif to_integer(address) = 27 then
                                readdata <= to_signed(frog_life, 16);
                        end if;
                end if;
        elsif frog_alive = "11" then
                frog_x <= 224;
                frog_y <= 416;
                frog_direction <= 1;
                deadflag <= '0';
                timerange <= 300;
                if frog_life = 0 then
                        pause <= 1;
                end if;
        elsif frog_alive = "00" then
                if timeflag = '1' then
                        if timerange > 0 then
                                timerange <= timerange - 1;
                        end if;
                end if;
                if jiamingflag = '1' then
                        jiamingflag <= '0';
                end if;
        end if;
    end if;
end process Key;

-- Horizontal and vertical counters

HCounter : process (clk25)
begin
```

```
  if rising_edge(clk25) then
    if reset = '0' 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 = '0' 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 = '0' 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 = '0' 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 = '0' 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 = '0' 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;
-- items generator

-- item movement logic
Move_item : process (clk_move)
begin
        if rising_edge(clk_move) then
```

```
        if pause=0 then
         if k = 30 then
                if phase = 3 then phase <= 1;
          else phase <= phase + 1;
          end if;
          k <= 1;
         else k<= k+1;
         end if;
         if j = 5 then   j <= 2;
         else j <= j+1;
         end if;
         for i in 0 to 14 loop
          if (abs(speed(i)) mod j) = 0 then
                 if speed(i) > 0 then
                        offset(i) <= (offset(i)+1) mod 512;
                 else
                        offset(i) <= (offset(i)-1) mod 512;
                 end if;
                end if;
         end loop;

                if frog_y mod 32 = 0 then
                 for i in 0 to 14 loop
                        if i = frog_y/32 then
                          if (abs(speed(i)) mod j) = 0 then
                                 if frog_x + frog_offset(i) + speed(i)/(abs(speed(i))) >= 0
and frog_x + frog_offset(i) + speed(i)/(abs(speed(i))) < 480 and frog_y >= 0 and
frog_y < 224 then
                                        if speed(i) > 0 then
                                          frog_offset(i) <= frog_offset(i) + 1;
                                        else
                                          frog_offset(i) <= frog_offset(i) - 1;
                                        end if;
                                 end if;
                     end if;
                        else
                                 frog_offset(i) <= 0;
                        end if;
                 end loop;
                end if;
        end if;
        end if;
 end process Move_item;

 -- frog action logic
 Logic : process (clk)
```

```
 begin
        if rising_edge (clk) then
        level_matrix(1, 5) <= winnum/5 + 1;
        if pause = 0 then
                start_matrix <= ((36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36),
                                                (36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36)
                                                );
        if frog_action = 0 then
        case frog_alive is
        when "00" => --alive
          if jiamingflag = '1' then
                frog_life <= frog_life + 1;
                if frog_life + 1 > 5 then
                        frogming(1, 2 * (frog_life - 4)) <= 2;
                        frogming(1, 2 * (frog_life - 4)- 1) <= 1;
                else
                        frogming(0, 2 * (frog_life + 1)) <= 2;
                        frogming(0, 2 * (frog_life + 1)- 1) <= 1;
                end if;
          elsif deadflag = '1' or timerange <= 0 then
            frog_alive <= "01";
                frog_life <= frog_life - 1;
                if frog_life > 5 then
                        frogming(1, 2 * (frog_life - 5)) <= 0;
                        frogming(1, 2 * (frog_life - 5)- 1) <= 0;
                else
                        frogming(0, 2 * frog_life) <= 0;
                        frogming(0, 2 * frog_life - 1) <= 0;
                end if;
          elsif frog_y >= 224 then
            if river_pattern (frog_y/32,((frog_x + 4 + frog_offset(frog_y/32) + 1 -
offset(frog_y/32)) mod 512)/32) = 0 and river_pattern (frog_y/32,((frog_x + 28 +
frog_offset(frog_y/32) + 1 - offset(frog_y/32)) mod 512)/32) = 0 then
                frog_alive <= "00";
            else
                frog_alive <= "01";
                frog_life <= frog_life - 1;
                if frog_life > 5 then
                        frogming(1, 2 * (frog_life - 5)) <= 0;
                        frogming(1, 2 * (frog_life - 5)- 1) <= 0;
                else
                        frogming(0, 2 * frog_life) <= 0;
                        frogming(0, 2 * frog_life - 1) <= 0;
             end if;
            end if;
          elsif frog_y >=64 then
```

```
                if river_pattern (frog_y/32,((frog_x + 16 + frog_offset(frog_y/32) + 1 -
offset(frog_y/32)) mod 512)/32) > 0 then
                  frog_alive <= "00";
                else
                  frog_alive <= "01";
                  frog_life <= frog_life - 1;
                  if frog_life > 5 then
                          frogming(1, 2 * (frog_life - 5)) <= 0;
                          frogming(1, 2 * (frog_life - 5)- 1) <= 0;
                  else
                          frogming(0, 2 * frog_life) <= 0;
                          frogming(0, 2 * frog_life - 1) <= 0;
                  end if;
                end if;
          end if;

        when "01" => --dead
            if q = 15000000 then
              q <= 1;
              frog_alive <= "10";
            else
                        q <= q + 1;
                  end if;
        when "10" =>
                  if q = 15000000 then
              q <= 1;
              frog_alive <= "11";
            else
                        q <= q + 1;
                  end if;
        when "11" =>
                  if frog_life = 0 then
                          frog_life <= 5;
                          frogming(0,1) <= 1;
                          frogming(0,3) <= 1;
                          frogming(0,5) <= 1;
                          frogming(0,7) <= 1;
                          frogming(0,9) <= 1;
                          frogming(0,2) <= 2;
                          frogming(0,4) <= 2;
                          frogming(0,6) <= 2;
                          frogming(0,8) <= 2;
                          frogming(0,10) <= 2;
                  end if;
            frog_alive <= "00";
        end case;
```

```
        end if;
        else
                start_matrix <= ((36, 36, 36, 25, 27, 14, 28, 28, 36, 36, 36),
                                                (36, 36, 36, 28, 25, 10, 12, 14, 36, 36, 36)
                                                );
        end if;
        end if;
 end process Logic;

-- bakcground, item, frog display
 Display_item : process (clk25)
 begin
  if rising_edge(clk25) then
          Hscreen <= to_integer(Hcount) - HBACK_PORCH  - HSYNC + 1 ;
          Vscreen <= to_integer(Vcount) - VBACK_PORCH - VSYNC  ;

          if Hscreen >= 0 and Hscreen < 512 and Vscreen >=0 and Vscreen < 480 then
     patternNum <= river_pattern (Vscreen/32,((Hscreen-offset(Vscreen/32)) mod
512)/32);
                if Vscreen >= 64 then
                        back_num <= background(Vscreen/32,(Hscreen + 1)/32);
                else
                        back_num <= background(Vscreen/32,(Hscreen)/32);
                end if;

                horizontal <= (Hscreen - offset(Vscreen/32)) mod 32;
                vertical <= Vscreen mod 32;
           Hback <= Hscreen mod 32;
           sel_addr <= to_unsigned(patternNum,4);
                sel_back <= to_unsigned(back_num,5);

                --draw frog
                if  Hscreen >= (frog_x + frog_offset(frog_y/32)) and Hscreen < ((frog_x
+ frog_offset(frog_y/32)) + 32)  and Vscreen >= frog_y and Vscreen < frog_y + 32
then
                  frog_horizontal <= Hscreen - (frog_x + frog_offset(frog_y/32)) ;
                  frog_vertical   <= Vscreen - frog_y;

                case frog_action is
                  when 0 =>
                    sel_frog <= "01"; --static
                  when 1 =>
                        sel_frog <= "11"; --jump
                  when others =>
                        sel_frog <= "00"; --frogflag = '0'
                end case;
```

```
                else
                  sel_frog <= "00";
                end if;
            else
              sel_addr <= "0000";
              sel_frog <= "00";
                sel_back <= "11111";
            end if;
          end if;
 end process Display_item;

-- scoreboard pattern setting
Display_number: process(clk25)
begin
 if rising_edge(clk25) then
        Hnumber <= to_integer(Hcount) - HBACK_PORCH  - HSYNC - 512 - 8;
        Vnumber <= to_integer(Vcount) - VBACK_PORCH - VSYNC;
        --display score on the screen
        if Hnumber >= 0 and Hnumber < 88 and Vnumber >= 36 and Vnumber < 492
then
                num_horizontal <= Hnumber mod 8;
                if Vnumber >= 36 and Vnumber < 68 then
                        score_number <= score_matrix((Vnumber - 36)/16, (Hnumber
+ 2)/8);
                        num_vertical <= (Vnumber - 36) mod 16;
                elsif Vnumber >= 120 and Vnumber < 152 then
                        score_number <= high_matrix((Vnumber - 120)/16, (Hnumber
+ 2)/8);
                        num_vertical <= (Vnumber - 120) mod 16;
                elsif Vnumber >= 210 and Vnumber < 226 then
                        score_number <= life_left((Hnumber + 2)/8);
                        num_vertical <= (Vnumber - 210) mod 16;
                elsif Vnumber >= 226 and Vnumber < 258 then
                        small_num <= frogming((Vnumber - 226)/16, (Hnumber)/8);
                        num_vertical <= (Vnumber - 226) mod 16;
                elsif Vnumber >= 318 and Vnumber < 350 then
                        score_number <= level_matrix((Vnumber - 318)/16, (Hnumber
+ 2)/8);
                        num_vertical <= (Vnumber - 318) mod 16;
                elsif Vnumber >= 410 and Vnumber < 442 then
                        score_number <= start_matrix((Vnumber - 410)/16, (Hnumber
+ 2)/8);
                        num_vertical <= (Vnumber - 410) mod 16;
                end if;
        elsif Hscreen >= 0 and Hscreen < 64 and Vnumber >= 456 and Vnumber <
472 then
```

```
                score_number <= time_left((Hscreen + 2)/8);
                num_vertical <= (Vnumber - 456) mod 16;
                num_horizontal <= Hscreen mod 8;
        else
                score_number <= 36;          --set score_number to display nothing
                small_num <= 3;
        end if;
        case small_num is
                when 0 => sel_small <= "00";
                when 1 => sel_small <= "01";
                when 2 => sel_small <= "10";
                when others => sel_small <= "11";
        end case;
        sel_num_addr <= to_unsigned(score_number,6);
 end if;
end process Display_number;

-- video out rules
 VideoOut : process(clk25,reset)
 begin
        if reset = '0' then
    VGA_R <= "0000000000";
    VGA_G <= "0000000000";
    VGA_B <= "0000000000";
  elsif clk25'event and clk25 = '1' then
     if frogflag = '1' then
       VGA_R <= std_logic_vector(frog_R & "000000");
       VGA_G <= std_logic_vector(frog_G & "000000");
       VGA_B <= std_logic_vector(frog_B & "000000");
            elsif itemflag = '1' then
       VGA_R <= std_logic_vector(color_R & "000000");
       VGA_G <= std_logic_vector(color_G & "000000");
       VGA_B <= std_logic_vector(color_B & "000000");
                elsif backflag = '1' then
       VGA_R <= std_logic_vector(back_R & "000000");
       VGA_G <= std_logic_vector(back_G & "000000");
       VGA_B <= std_logic_vector(back_B & "000000");
                elsif numflag = '1' and Hnumber <= 89 then
                  VGA_R <= std_logic_vector(charc);
       VGA_G <= std_logic_vector(charc);
       VGA_B <= std_logic_vector(charc);
                elsif smallflag = '1' then
       VGA_R <= std_logic_vector(small_R & "000000");
       VGA_G <= std_logic_vector(small_G & "000000");
       VGA_B <= std_logic_vector(small_B & "000000");
```

```
                elsif (Hcount - HSYNC - HBACK_PORCH >= 128) and (Hcount - HSYNC
- HBACK_PORCH <= 128 + timerange) and (Vcount - VSYNC - VBACK_PORCH >= 456)
and (Vcount - VSYNC - VBACK_PORCH < 472) then
                        if timerange > 75 then
                          VGA_R <= "0000000000";
                          VGA_G <= "1111111111";
                          VGA_B <= "0000000000";
                        else
                                -- times up warning
                                if (timerange mod 5) = 0 then
                                        VGA_R <= "0000000000";
                                        VGA_G <= "1111111111";
                                        VGA_B <= "0000000000";
                                else
                                        VGA_R <= "1111111111";
                                        VGA_G <= "0000000000";
                                        VGA_B <= "0000000000";
                                end if;
                        end if;
                elsif 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;
   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;
```

number_E_ROM.vhd:

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

entity number_E_ROM is
```

```
  port(
clk   : in std_logic;
addr  : in unsigned (7 downto 0);
data  : out std_logic
  );
end number_E_ROM;

architecture rtl of number_E_ROM is
type rom_type is array(0 to 215) of std_logic;
  constant ROM: rom_type := (
'0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0',
'1','1','1','1','1','1','1','0','0','0','0','0',
'0','1','1','0','0','0','1','1','0','0','0','0',
'0','1','1','0','0','0','0','0','0','0','0','0',
'0','1','1','0','0','0','0','0','0','0','0','0',
'0','1','1','0','0','0','0','0','0','0','0','0',
'0','1','1','1','1','1','1','0','0','0','0','0',
'0','1','1','0','0','0','1','0','0','0','0','0',
'0','1','1','0','0','0','0','0','0','0','0','0',
'0','1','1','0','0','0','0','0','0','0','0','0',
'0','1','1','0','0','0','0','0','0','0','0','0',
'0','1','1','0','0','0','1','1','0','0','0','0',
'1','1','1','1','1','1','1','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0');
begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

de2_audio.vhd:

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

```
entity de2_audio is

 port (
  clk      : in  std_logic;
  reset_n   : in  std_logic;
  read     : in  std_logic;
  write    : in  std_logic;
  chipselect : in  std_logic;
  address   : in  unsigned(1 downto 0);
        writedata  : in  unsigned(15 downto 0);
        readdata   : out unsigned(15 downto 0);

  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_audio;

architecture rtl of de2_audio is
--component 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);
--
--   -- 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 de2_audio_select is
 port (
  clk      : in  std_logic;
  reset_n   : in  std_logic;
        audio_ini  : in  std_logic;
```

```
        audio_sel  : in  unsigned (1 downto 0);
        audio_vol  : 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;

 type ram_type is array(3 downto 0) of unsigned(15 downto 0);
 signal RAM : ram_type := (X"0000", X"0000", X"0000", X"0000");
 signal ram_address   : unsigned(1 downto 0);
 signal audio_request : std_logic;
 signal audio_sel     : unsigned(15 downto 0) := X"0000";
 signal audio_ini     : unsigned(15 downto 0) := X"0000";
 signal audio_tmp        : unsigned(15 downto 0) := X"0000";
 signal audio_vol     : unsigned(15 downto 0) := X"0000";

begin
 ram_address <= address(1 downto 0);

 process (clk)
 begin
        if rising_edge(clk) then
          if reset_n = '0' then
                audio_sel <= (others => '0');
                audio_ini <= (others => '0');
    else
     if chipselect = '1' then
                        if read = '1' then
                                readdata <= RAM(to_integer(ram_address));
                                audio_sel  <= RAM(to_integer("01"));
                                audio_ini  <= RAM(to_integer("11"));
                                audio_vol  <= RAM(to_integer("10"));
                        elsif write = '1' then
                                RAM(to_integer(ram_address)) <= writedata;
                        end if;
                end if;
          end if;
        end if;
 end process;

--V1: de2_wm8731_audio
```

```
--port map(
--    clk => clk,      --  Audio CODEC Chip Clock AUD_XCK (18.43 MHz)
--    reset_n => reset_n,
--    test_mode => '1',      --    Audio CODEC controller test mode
--    audio_request => audio_request,  --    Audio controller request new data
--    data => "0000000000000000",
--
--    -- Audio interface signals
--    AUD_ADCLRCK  => AUD_ADCLRCK,   --    Audio CODEC ADC LR Clock
--    AUD_ADCDAT   => AUD_ADCDAT,   --    Audio CODEC ADC Data
--    AUD_DACLRCK  => AUD_DACLRCK,   --    Audio CODEC DAC LR Clock
--    AUD_DACDAT   => AUD_DACDAT,   --    Audio CODEC DAC Data
--    AUD_BCLK     => AUD_BCLK  --    Audio CODEC Bit-Stream Clock
--  );

  V1: de2_audio_select port map (
          clk       => clk,
          reset_n    => reset_n,
          audio_ini  => audio_ini (0),
          audio_sel  => audio_sel (1 downto 0),
          audio_vol  => audio_vol (1 downto 0),

          AUD_ADCLRCK  => AUD_ADCLRCK,
          AUD_ADCDAT   => AUD_ADCDAT,
          AUD_DACLRCK  => AUD_DACLRCK,
          AUD_DACDAT   => AUD_DACDAT,
          AUD_BCLK     => AUD_BCLK
  );


end rtl;
```

de2_audio_pll.vhd:

```
-- megafunction wizard: %ALTPLL%
-- GENERATION: STANDARD
-- VERSION: WM1.0
-- MODULE: altpll

-- ============================================================
-- File Name: de2_audio_pll.vhd
-- Megafunction Name(s):
--                          altpll
--
-- Simulation Library Files(s):
```

```
--                      altera_mf
-- ============================================================
-- ************************************************************
-- THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
--
-- 7.2 Build 151 09/26/2007 SJ Full Version
-- ************************************************************


--Copyright (C) 1991-2007 Altera Corporation
--Your use of Altera Corporation's design tools, logic functions
--and other software and tools, and its AMPP partner logic
--functions, and any output files from any of the foregoing
--(including device programming or simulation files), and any
--associated documentation or information are expressly subject
--to the terms and conditions of the Altera Program License
--Subscription Agreement, Altera MegaCore Function License
--Agreement, or other applicable license agreement, including,
--without limitation, that your use is for the sole purpose of
--programming logic devices manufactured by Altera and sold by
--Altera or its authorized distributors.  Please refer to the
--applicable agreement for further details.


LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY altera_mf;
USE altera_mf.all;

ENTITY de2_audio_pll IS
PORT
(
inclk0          : IN STD_LOGIC  := '0';
c0              : OUT STD_LOGIC
);
END de2_audio_pll;


ARCHITECTURE SYN OF de2_audio_pll IS

SIGNAL sub_wire0    : STD_LOGIC_VECTOR (5 DOWNTO 0);
SIGNAL sub_wire1    : STD_LOGIC ;
SIGNAL sub_wire2    : STD_LOGIC ;
SIGNAL sub_wire3    : STD_LOGIC_VECTOR (1 DOWNTO 0);
SIGNAL sub_wire4_bv         : BIT_VECTOR (0 DOWNTO 0);
```

```
SIGNAL sub_wire4    : STD_LOGIC_VECTOR (0 DOWNTO 0);



COMPONENT altpll
GENERIC (
clk0_divide_by              : NATURAL;
clk0_duty_cycle             : NATURAL;
clk0_multiply_by            : NATURAL;
clk0_phase_shift            : STRING;
compensate_clock            : STRING;
inclk0_input_frequency              : NATURAL;
intended_device_family              : STRING;
lpm_hint            : STRING;
lpm_type            : STRING;
operation_mode              : STRING;
port_activeclock            : STRING;
port_areset             : STRING;
port_clkbad0            : STRING;
port_clkbad1            : STRING;
port_clkloss            : STRING;
port_clkswitch              : STRING;
port_configupdate           : STRING;
port_fbin               : STRING;
port_inclk0             : STRING;
port_inclk1             : STRING;
port_locked             : STRING;
port_pfdena             : STRING;
port_phasecounterselect             : STRING;
port_phasedone              : STRING;
port_phasestep              : STRING;
port_phaseupdown            : STRING;
port_pllena             : STRING;
port_scanaclr           : STRING;
port_scanclk            : STRING;
port_scanclkena             : STRING;
port_scandata           : STRING;
port_scandataout            : STRING;
port_scandone               : STRING;
port_scanread               : STRING;
port_scanwrite              : STRING;
port_clk0               : STRING;
port_clk1               : STRING;
port_clk2               : STRING;
port_clk3               : STRING;
port_clk4               : STRING;
```

```
port_clk5               : STRING;
port_clkena0            : STRING;
port_clkena1            : STRING;
port_clkena2            : STRING;
port_clkena3            : STRING;
port_clkena4            : STRING;
port_clkena5            : STRING;
port_extclk0            : STRING;
port_extclk1            : STRING;
port_extclk2            : STRING;
port_extclk3            : STRING
);
PORT (
inclk    : IN STD_LOGIC_VECTOR (1 DOWNTO 0);
clk      : OUT STD_LOGIC_VECTOR (5 DOWNTO 0)
);
END COMPONENT;

BEGIN
sub_wire4_bv(0 DOWNTO 0) <= "0";
sub_wire4    <= To_stdlogicvector(sub_wire4_bv);
sub_wire1    <= sub_wire0(0);
c0    <= sub_wire1;
sub_wire2    <= inclk0;
sub_wire3    <= sub_wire4(0 DOWNTO 0) & sub_wire2;

altpll_component : altpll
GENERIC MAP (
clk0_divide_by => 50000000,
clk0_duty_cycle => 50,
clk0_multiply_by => 18431999,
clk0_phase_shift => "0",
compensate_clock => "CLK0",
inclk0_input_frequency => 20000,
intended_device_family => "Cyclone II",
lpm_hint => "CBX_MODULE_PREFIX=de2_audio_pll",
lpm_type => "altpll",
operation_mode => "NORMAL",
port_activeclock => "PORT_UNUSED",
port_areset => "PORT_UNUSED",
port_clkbad0 => "PORT_UNUSED",
port_clkbad1 => "PORT_UNUSED",
port_clkloss => "PORT_UNUSED",
port_clkswitch => "PORT_UNUSED",
port_configupdate => "PORT_UNUSED",
port_fbin => "PORT_UNUSED",
```

```
port_inclk0 => "PORT_USED",
port_inclk1 => "PORT_UNUSED",
port_locked => "PORT_UNUSED",
port_pfdena => "PORT_UNUSED",
port_phasecounterselect => "PORT_UNUSED",
port_phasedone => "PORT_UNUSED",
port_phasestep => "PORT_UNUSED",
port_phaseupdown => "PORT_UNUSED",
port_pllena => "PORT_UNUSED",
port_scanaclr => "PORT_UNUSED",
port_scanclk => "PORT_UNUSED",
port_scanclkena => "PORT_UNUSED",
port_scandata => "PORT_UNUSED",
port_scandataout => "PORT_UNUSED",
port_scandone => "PORT_UNUSED",
port_scanread => "PORT_UNUSED",
port_scanwrite => "PORT_UNUSED",
port_clk0 => "PORT_USED",
port_clk1 => "PORT_UNUSED",
port_clk2 => "PORT_UNUSED",
port_clk3 => "PORT_UNUSED",
port_clk4 => "PORT_UNUSED",
port_clk5 => "PORT_UNUSED",
port_clkena0 => "PORT_UNUSED",
port_clkena1 => "PORT_UNUSED",
port_clkena2 => "PORT_UNUSED",
port_clkena3 => "PORT_UNUSED",
port_clkena4 => "PORT_UNUSED",
port_clkena5 => "PORT_UNUSED",
port_extclk0 => "PORT_UNUSED",
port_extclk1 => "PORT_UNUSED",
port_extclk2 => "PORT_UNUSED",
port_extclk3 => "PORT_UNUSED"
)
PORT MAP (
inclk => sub_wire3,
clk => sub_wire0
);



END SYN;

-- ============================================================
-- CNX file retrieval info
-- ============================================================
```

```
-- Retrieval info: PRIVATE: ACTIVECLK_CHECK STRING "0"
-- Retrieval info: PRIVATE: BANDWIDTH STRING "1.000"
-- Retrieval info: PRIVATE: BANDWIDTH_FEATURE_ENABLED STRING "0"
-- Retrieval info: PRIVATE: BANDWIDTH_FREQ_UNIT STRING "MHz"
-- Retrieval info: PRIVATE: BANDWIDTH_PRESET STRING "Low"
-- Retrieval info: PRIVATE: BANDWIDTH_USE_AUTO STRING "1"
-- Retrieval info: PRIVATE: BANDWIDTH_USE_CUSTOM STRING "0"
-- Retrieval info: PRIVATE: BANDWIDTH_USE_PRESET STRING "0"
-- Retrieval info: PRIVATE: CLKBAD_SWITCHOVER_CHECK STRING "0"
-- Retrieval info: PRIVATE: CLKLOSS_CHECK STRING "0"
-- Retrieval info: PRIVATE: CLKSWITCH_CHECK STRING "1"
-- Retrieval info: PRIVATE: CNX_NO_COMPENSATE_RADIO STRING "0"
-- Retrieval info: PRIVATE: CREATE_CLKBAD_CHECK STRING "0"
-- Retrieval info: PRIVATE: CREATE_INCLK1_CHECK STRING "0"
-- Retrieval info: PRIVATE: CUR_DEDICATED_CLK STRING "c0"
-- Retrieval info: PRIVATE: CUR_FBIN_CLK STRING "e0"
-- Retrieval info: PRIVATE: DEVICE_SPEED_GRADE STRING "6"
-- Retrieval info: PRIVATE: DIV_FACTOR0 NUMERIC "1"
-- Retrieval info: PRIVATE: DUTY_CYCLE0 STRING "50.00000000"
-- Retrieval info: PRIVATE: EXPLICIT_SWITCHOVER_COUNTER STRING "0"
-- Retrieval info: PRIVATE: EXT_FEEDBACK_RADIO STRING "0"
-- Retrieval info: PRIVATE: GLOCKED_COUNTER_EDIT_CHANGED STRING "1"
-- Retrieval info: PRIVATE: GLOCKED_FEATURE_ENABLED STRING "1"
-- Retrieval info: PRIVATE: GLOCKED_MODE_CHECK STRING "0"
-- Retrieval info: PRIVATE: GLOCK_COUNTER_EDIT NUMERIC "1048575"
-- Retrieval info: PRIVATE: HAS_MANUAL_SWITCHOVER STRING "1"
-- Retrieval info: PRIVATE: INCLK0_FREQ_EDIT STRING "50.000"
-- Retrieval info: PRIVATE: INCLK0_FREQ_UNIT_COMBO STRING "MHz"
-- Retrieval info: PRIVATE: INCLK1_FREQ_EDIT STRING "100.000"
-- Retrieval info: PRIVATE: INCLK1_FREQ_EDIT_CHANGED STRING "1"
-- Retrieval info: PRIVATE: INCLK1_FREQ_UNIT_CHANGED STRING "1"
-- Retrieval info: PRIVATE: INCLK1_FREQ_UNIT_COMBO STRING "MHz"
-- Retrieval info: PRIVATE: INTENDED_DEVICE_FAMILY STRING "Cyclone II"
-- Retrieval info: PRIVATE: INT_FEEDBACK__MODE_RADIO STRING "1"
-- Retrieval info: PRIVATE: LOCKED_OUTPUT_CHECK STRING "0"
-- Retrieval info: PRIVATE: LONG_SCAN_RADIO STRING "1"
-- Retrieval info: PRIVATE: LVDS_MODE_DATA_RATE STRING "Not Available"
-- Retrieval info: PRIVATE: LVDS_MODE_DATA_RATE_DIRTY NUMERIC "0"
-- Retrieval info: PRIVATE: LVDS_PHASE_SHIFT_UNIT0 STRING "deg"
-- Retrieval info: PRIVATE: MIRROR_CLK0 STRING "0"
-- Retrieval info: PRIVATE: MULT_FACTOR0 NUMERIC "1"
-- Retrieval info: PRIVATE: NORMAL_MODE_RADIO STRING "1"
-- Retrieval info: PRIVATE: OUTPUT_FREQ0 STRING "18.43200000"
-- Retrieval info: PRIVATE: OUTPUT_FREQ_MODE0 STRING "1"
-- Retrieval info: PRIVATE: OUTPUT_FREQ_UNIT0 STRING "MHz"
-- Retrieval info: PRIVATE: PHASE_RECONFIG_FEATURE_ENABLED STRING "0"
```

```
-- Retrieval info: PRIVATE: PHASE_RECONFIG_INPUTS_CHECK STRING "0"
-- Retrieval info: PRIVATE: PHASE_SHIFT0 STRING "0.00000000"
-- Retrieval info: PRIVATE: PHASE_SHIFT_STEP_ENABLED_CHECK STRING "0"
-- Retrieval info: PRIVATE: PHASE_SHIFT_UNIT0 STRING "deg"
-- Retrieval info: PRIVATE: PLL_ADVANCED_PARAM_CHECK STRING "0"
-- Retrieval info: PRIVATE: PLL_ARESET_CHECK STRING "0"
-- Retrieval info: PRIVATE: PLL_AUTOPLL_CHECK NUMERIC "1"
-- Retrieval info: PRIVATE: PLL_ENA_CHECK STRING "0"
-- Retrieval info: PRIVATE: PLL_ENHPLL_CHECK NUMERIC "0"
-- Retrieval info: PRIVATE: PLL_FASTPLL_CHECK NUMERIC "0"
-- Retrieval info: PRIVATE: PLL_FBMIMIC_CHECK STRING "0"
-- Retrieval info: PRIVATE: PLL_LVDS_PLL_CHECK NUMERIC "0"
-- Retrieval info: PRIVATE: PLL_PFDENA_CHECK STRING "0"
-- Retrieval info: PRIVATE: PLL_TARGET_HARCOPY_CHECK NUMERIC "0"
-- Retrieval info: PRIVATE: PRIMARY_CLK_COMBO STRING "inclk0"
-- Retrieval info: PRIVATE: RECONFIG_FILE STRING "de2_audio_pll.mif"
-- Retrieval info: PRIVATE: SACN_INPUTS_CHECK STRING "0"
-- Retrieval info: PRIVATE: SCAN_FEATURE_ENABLED STRING "0"
-- Retrieval info: PRIVATE: SELF_RESET_LOCK_LOSS STRING "0"
-- Retrieval info: PRIVATE: SHORT_SCAN_RADIO STRING "0"
-- Retrieval info: PRIVATE: SPREAD_FEATURE_ENABLED STRING "0"
-- Retrieval info: PRIVATE: SPREAD_FREQ STRING "50.000"
-- Retrieval info: PRIVATE: SPREAD_FREQ_UNIT STRING "KHz"
-- Retrieval info: PRIVATE: SPREAD_PERCENT STRING "0.500"
-- Retrieval info: PRIVATE: SPREAD_USE STRING "0"
-- Retrieval info: PRIVATE: SRC_SYNCH_COMP_RADIO STRING "0"
-- Retrieval info: PRIVATE: STICKY_CLK0 STRING "1"
-- Retrieval info: PRIVATE: SWITCHOVER_COUNT_EDIT NUMERIC "1"
-- Retrieval info: PRIVATE: SWITCHOVER_FEATURE_ENABLED STRING "1"
-- Retrieval info: PRIVATE: SYNTH_WRAPPER_GEN_POSTFIX STRING "0"
-- Retrieval info: PRIVATE: USE_CLK0 STRING "1"
-- Retrieval info: PRIVATE: USE_CLKENA0 STRING "0"
-- Retrieval info: PRIVATE: USE_MIL_SPEED_GRADE NUMERIC "0"
-- Retrieval info: PRIVATE: ZERO_DELAY_RADIO STRING "0"
-- Retrieval info: LIBRARY: altera_mf altera_mf.altera_mf_components.all
-- Retrieval info: CONSTANT: CLK0_DIVIDE_BY NUMERIC "50000000"
-- Retrieval info: CONSTANT: CLK0_DUTY_CYCLE NUMERIC "50"
-- Retrieval info: CONSTANT: CLK0_MULTIPLY_BY NUMERIC "18431999"
-- Retrieval info: CONSTANT: CLK0_PHASE_SHIFT STRING "0"
-- Retrieval info: CONSTANT: COMPENSATE_CLOCK STRING "CLK0"
-- Retrieval info: CONSTANT: INCLK0_INPUT_FREQUENCY NUMERIC "20000"
-- Retrieval info: CONSTANT: INTENDED_DEVICE_FAMILY STRING "Cyclone II"
-- Retrieval info: CONSTANT: LPM_TYPE STRING "altpll"
-- Retrieval info: CONSTANT: OPERATION_MODE STRING "NORMAL"
-- Retrieval info: CONSTANT: PORT_ACTIVECLOCK STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_ARESET STRING "PORT_UNUSED"
```

```
-- Retrieval info: CONSTANT: PORT_CLKBAD0 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_CLKBAD1 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_CLKLOSS STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_CLKSWITCH STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_CONFIGUPDATE STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_FBIN STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_INCLK0 STRING "PORT_USED"
-- Retrieval info: CONSTANT: PORT_INCLK1 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_LOCKED STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_PFDENA STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_PHASECOUNTERSELECT STRING
"PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_PHASEDONE STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_PHASESTEP STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_PHASEUPDOWN STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_PLLENA STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_SCANACLR STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_SCANCLK STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_SCANCLKENA STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_SCANDATA STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_SCANDATAOUT STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_SCANDONE STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_SCANREAD STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_SCANWRITE STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clk0 STRING "PORT_USED"
-- Retrieval info: CONSTANT: PORT_clk1 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clk2 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clk3 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clk4 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clk5 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clkena0 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clkena1 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clkena2 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clkena3 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clkena4 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clkena5 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_extclk0 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_extclk1 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_extclk2 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_extclk3 STRING "PORT_UNUSED"
-- Retrieval info: USED_PORT: @clk 0 0 6 0 OUTPUT_CLK_EXT VCC "@clk[5..0]"
-- Retrieval info: USED_PORT: @extclk 0 0 4 0 OUTPUT_CLK_EXT VCC
"@extclk[3..0]"
-- Retrieval info: USED_PORT: @inclk 0 0 2 0 INPUT_CLK_EXT VCC "@inclk[1..0]"
-- Retrieval info: USED_PORT: c0 0 0 0 0 OUTPUT_CLK_EXT VCC "c0"
-- Retrieval info: USED_PORT: inclk0 0 0 0 0 INPUT_CLK_EXT GND "inclk0"
```

```
-- Retrieval info: CONNECT: @inclk 0 0 1 0 inclk0 0 0 0 0
-- Retrieval info: CONNECT: c0 0 0 0 0 @clk 0 0 1 0
-- Retrieval info: CONNECT: @inclk 0 0 1 1 GND 0 0 0 0
-- Retrieval info: GEN_FILE: TYPE_NORMAL de2_audio_pll.vhd TRUE FALSE
-- Retrieval info: GEN_FILE: TYPE_NORMAL de2_audio_pll.ppf TRUE FALSE
-- Retrieval info: GEN_FILE: TYPE_NORMAL de2_audio_pll.inc FALSE FALSE
-- Retrieval info: GEN_FILE: TYPE_NORMAL de2_audio_pll.cmp TRUE FALSE
-- Retrieval info: GEN_FILE: TYPE_NORMAL de2_audio_pll.bsf FALSE FALSE
-- Retrieval info: GEN_FILE: TYPE_NORMAL de2_audio_pll_inst.vhd FALSE FALSE
-- Retrieval info: GEN_FILE: TYPE_NORMAL de2_audio_pll_waveforms.html TRUE
FALSE
-- Retrieval info: GEN_FILE: TYPE_NORMAL de2_audio_pll_wave*.jpg FALSE FALSE
-- Retrieval info: LIB_FILE: altera_mf
-- Retrieval info: CBX_MODULE_PREFIX: ON
```

De2_audio_delect.vhd:

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

entity de2_audio_select is

 port (
  clk      : in  std_logic;
  reset_n   : in  std_logic;
        audio_ini  : in  std_logic;
        audio_sel  : in  unsigned (1 downto 0);
        audio_vol  : in  unsigned (1 downto 0);

        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_audio_select;

architecture rtl of de2_audio_select 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);
         frq_divider1 : in unsigned(15 downto 0);
         frq_divider2 : in unsigned(15 downto 0);
         audio_volume1: in unsigned(1 downto 0);
         audio_volume2: in unsigned(1 downto 0);
         audio_track1 : in unsigned(1 downto 0);
         audio_track2 : in unsigned(1 downto 0);
         audio_vol        : in unsigned(1 downto 0);
--       press_delay  : integer;

--       cycle_divider : in unsigned(9 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;

  type ram1_type is array(integer range 0 to 150 - 1) of unsigned(15 downto 0);
  type ram2_type is array(integer range 0 to 2,integer range 0 to 1) of unsigned(15
downto 0);

  signal RAM1 : ram1_type :=   (X"2015", X"2203", X"2065", X"2203", X"2065",
X"2203", X"2064", X"2202",
                                                        X"2084", X"2202", X"2095",
X"2203", X"2065", X"2203", X"2065", X"2203",
                                                        X"2084", X"2202", X"2094",
X"2202", X"2085", X"2203", X"2045", X"2203",
                                                        X"2045", X"2203", X"2204",
X"2202", X"2084", X"2202", X"2095", X"2203",
                                                        X"2065", X"2203", X"2065",
X"2203", X"2205", X"2203",
                                                        X"2015", X"2203", X"2065",
X"2203", X"2065", X"2203", X"2064", X"2202",
                                                        X"2084", X"2202", X"2095",
X"2203", X"2065", X"2203", X"2065", X"2203",
                                                        X"2064", X"2202", X"2094",
X"2202", X"20D4", X"2202", X"20D4", X"2202",
                                                        X"20D4", X"2202", X"20B4",
X"2202", X"2095", X"2203", X"2085", X"2203",
```

```
                                                    X"2095", X"2203", X"2065",
X"2203", X"2065", X"2203", X"2064", X"2202",
                                                    X"2094", X"2202",
                                                    X"20D5", X"2203", X"20D5",
X"2203", X"20B5", X"2203", X"2095", X"2203",
                                                    X"2085", X"2203", X"2045",
X"2203", X"2045", X"2203", X"2044", X"2202",
                                                    X"2084", X"2202", X"20B5",
X"2203", X"20B5", X"2203", X"2095", X"2203",
                                                    X"2085", X"2203", X"2095",
X"2203", X"2065", X"2203", X"2065", X"2203",
                                                    X"2064", X"2202", X"2094",
X"2202",
                                                    X"20D5", X"2203", X"20D5",
X"2203", X"20B5", X"2203", X"2095", X"2203",
                                                    X"2085", X"2203", X"2045",
X"2203", X"2045", X"2203", X"2044", X"2202",
                                                    X"2084", X"2202", X"20B5",
X"2203", X"20B5", X"2203", X"2095", X"2203",
                                                    X"2085", X"2203", X"2095",
X"2203", X"2065", X"2203", X"2065", X"2203",
                                                    X"2205", X"2203");

 signal RAM2 : ram2_type := ((X"11A5", X"1201"),
                                                    (X"15B6", X"1201"),
                                                    (X"1201", X"1201"));

 signal audio_ctrl              : unsigned (15 downto 0);
 signal audio_ctrl1    : unsigned (15 downto 0);
 signal audio_ctrl2    : unsigned (15 downto 0);
 signal audio_set               : std_logic;
 signal audio_request           : std_logic;
 signal audio_lat               : std_logic;
 signal nrow                    : integer       := 0;
 signal ncol                    : integer       := 3;
 signal ram_address : integer       := 0;
 signal delay_cnt1              : integer       := 0;
 signal delay_cnt2              : integer       := 0;

 signal freq_divider1: unsigned (15 downto 0);
 signal audio_track1 : unsigned (1 downto 0);
 signal audio_pitch1 : unsigned (5 downto 0);
 signal audio_length1           : unsigned (3 downto 0);
 signal audio_volume1           : unsigned (1 downto 0);
 signal DELAY1                          : integer := 0;
```

```
 signal freq_divider2: unsigned (15 downto 0);
 signal audio_track2 : unsigned (1 downto 0);
 signal audio_pitch2 : unsigned (5 downto 0);
 signal audio_length2        : unsigned (3 downto 0);
 signal audio_volume2        : unsigned (1 downto 0);
 signal DELAY2                       : integer := 0;

 signal press                : std_logic;
 signal press_delay   : integer;
 constant PRESS_DELAY_MAX       : integer := 8000000;

--
begin

 audio_volume1               <= audio_ctrl1(13 downto 12);
 audio_track1          <= audio_ctrl1(11 downto 10);
 audio_pitch1          <= audio_ctrl1(9 downto 4);
 audio_length1               <= audio_ctrl1(3 downto 0);

 audio_volume2               <= audio_ctrl2(13 downto 12);
 audio_track2          <= audio_ctrl2(11 downto 10);
 audio_pitch2          <= audio_ctrl2(9 downto 4);
 audio_length2               <= audio_ctrl2(3 downto 0);

 process (clk)
 begin
        if rising_edge(clk) then
   if reset_n = '0' then
      delay_cnt1 <= 0;
   elsif delay_cnt1 = DELAY1 - 1 then
      delay_cnt1 <= 0;
   else
      delay_cnt1 <= delay_cnt1 + 1;
   end if;
  end if;
 end process;

 process (clk)
 begin
        if rising_edge(clk) then
   if reset_n = '0' then
     delay_cnt2 <= 0;
          elsif audio_lat = '0' and audio_ini = '1' then
                delay_cnt2 <= 0;
--        elsif nrow = 0 and press_delay > PRESS_DELAY_MAX - 1 then
--                delay_cnt2 <= 0;
```

```
    elsif delay_cnt2 = DELAY2 - 1 then
                delay_cnt2 <= 0;
    else
      delay_cnt2 <= delay_cnt2 + 1;
    end if;
   end if;
 end process;

 process (clk)
 begin
        if rising_edge(clk) then
                audio_lat <= audio_ini;
        end if;
 end process;

 process (clk)
 begin
  if rising_edge(clk) then
          if reset_n = '0' then
                ram_address <= 0;
          else
                audio_ctrl1 <= RAM1(ram_address);
                if delay_cnt1 = 0 then
                        if ram_address = 150 - 1 then
                                ram_address <= 0;
                        else
                                ram_address <= ram_address + 1;
                        end if;
                end if;
          end if;
        end if;
 end process;

--  process (clk)
--  begin
--        if rising_edge(clk) then
--                if audio_lat = '0' and audio_ini = '1' then
--                        press <= '1';
--                elsif audio_lat = '1' and audio_ini = '0' then
--                        press <= '0';
--                end if;
--        end if;
--  end process;

--  process (clk)
--  begin
```

```
--      if rising_edge(clk) then
--              if press = '0' then
--                      press_delay <= 0;
--              elsif press_delay > PRESS_DELAY_MAX - 1 then
--                      press_delay <= 0;
--              else
--                      press_delay <= press_delay + 1;
--              end if;
--      end if;
-- end process;

 process (clk)
 begin
  if rising_edge(clk) then
              if reset_n = '0' then
                      nrow <= 0;
                      ncol <= 0;
              else
                      if audio_lat = '0' and audio_ini = '1' then
                              nrow            <= to_integer(audio_sel);
                              ncol            <= 0;
                      end if;
                      audio_ctrl2 <= RAM2(nrow, ncol);
                      if delay_cnt2 = DELAY2 - 1 then
                              if ncol < 1 then
                                      ncol <= ncol + 1;
                              end if;
                      end if;

              end if;
      end if;
 end process;

 V1: de2_wm8731_audio port map (
      clk => clk,
      reset_n => reset_n,
      test_mode => '1',               -- Output a sine wave
      audio_request => audio_request,
      data => "0000000000000000",
      frq_divider1 => freq_divider1,
      frq_divider2 => freq_divider2,
      audio_volume1       => audio_volume1,
      audio_volume2       => audio_volume2,
      audio_track1  => audio_track1,
      audio_track2  => audio_track2,
      audio_vol           => audio_vol,
```

```
--          press_delay             => press_delay,
            --          cycle_divider  => cycle_divider,

            -- Audio interface signals
            AUD_ADCLRCK  => AUD_ADCLRCK,
            AUD_ADCDAT   => AUD_ADCDAT,
            AUD_DACLRCK  => AUD_DACLRCK,
            AUD_DACDAT   => AUD_DACDAT,
            AUD_BCLK     => AUD_BCLK
             );

  with audio_pitch1 select freq_divider1 <=
            X"0245" when "000001",
            X"0225" when "000010",
            X"0206" when "000011",
            X"01E9" when "000100",
            X"01CD" when "000101",
            X"01B3" when "000110",
            X"019B" when "000111",
            X"0184" when "001000",
            X"016E" when "001001",
            X"0159" when "001010",
            X"0146" when "001011",
            X"0134" when "001100",
            X"0122" when "001101",
            X"0112" when "001110",
            X"0103" when "001111",
            X"00F4" when "010000",
            X"00E6" when "010001",
            X"00D9" when "010010",
            X"00CD" when "010011",
            X"00C2" when "010100",
            X"00B7" when "010101",
            X"00AC" when "010110",
            X"00A3" when "010111",
            X"009A" when "011000",
            X"0091" when "011001",
            X"0000" when others;

  with audio_pitch2 select freq_divider2 <=
            X"0900" when "011010",
            X"1200" when "011011",
            X"0000" when others;

  with audio_length1 select DELAY1 <=
   100000 when X"0",
```

```
  625000 when X"1",
 1250000 when X"2",
 2500000 when X"3",
 5000000 when X"4",
10000000 when X"5",
20000000 when X"6",
40000000 when X"7",
   10000 when others;

 with audio_length2 select DELAY2 <=
  100000 when X"0",
  625000 when X"1",
 1250000 when X"2",
 2500000 when X"3",
 5000000 when X"4",
10000000 when X"5",
20000000 when X"6",
40000000 when X"7",
   10000 when others;

end rtl;
```

de2_wm8731_audio.vhd:

```
library ieee;
use ieee.std_logic_1164.all;
--use ieee.std_logic_arith.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;

library work;
use work.functions.all;
use work.soundtrack.all;

entity de2_wm8731_audio is
port (
        clk                             : in std_logic;      --  Audio CODEC Chip Clock
AUD_XCK (18.432 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);
        frq_divider1    : in unsigned(15 downto 0);
        frq_divider2    : in unsigned(15 downto 0);
        audio_volume1           : in unsigned(1 downto 0);
```

```
        audio_volume2       : in unsigned(1 downto 0);
        audio_track1  : in unsigned(1 downto 0);
        audio_track2  : in unsigned(1 downto 0);
        audio_vol           : in unsigned(1 downto 0);
--      press_delay         : in integer;
--      tone                : in unsigned(1 downto 0);

--      mul_channel         : in std_logic;
--      num_channel         : in std_logic;
--      cycle_divider  : in unsigned(9 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

component de2_audio_pll is
port (
inclk0          : in std_logic;
c0                      : out std_logic
);
end component;

signal lrck             : std_logic;
signal bclk             : std_logic;
signal xck              : std_logic;
signal scale            : std_logic;
signal audio_clock      : std_logic;

signal lrck_divider  : unsigned(7 downto 0);
signal bclk_divider  : unsigned(3 downto 0);
signal scale_divider1: unsigned(15 downto 0);
signal scale_divider2: unsigned(15 downto 0);

signal set_bclk : std_logic;
signal set_lrck : std_logic;
signal set_scale: std_logic;
signal clr_bclk : std_logic;
signal scale_lat: std_logic;
```

```
signal sin_out1    : std_logic_vector(15 downto 0);
signal sin_out2    : std_logic_vector(15 downto 0);
signal sin_counter1: unsigned(11 downto 0);
signal sin_counter2: unsigned(11 downto 0);
signal sin_out3: std_logic_vector(15 downto 0);
signal sin_out4: std_logic_vector(15 downto 0);
signal freq_lat    : unsigned(15 downto 0);
signal shift_out   : unsigned(15 downto 0);

--constant PRESS_DELAY_MAX : integer := 8000000;

begin

   -- LRCK divider
   -- Audio chip main clock is 18.432MHz / Sample rate 48KHz
   -- Divider is 18.432 MHz / 48KHz = 192 (X"C0")
   -- Left justify mode set by I2C controller

 audio_pll: de2_audio_pll port map (
          inclk0  => clk,
          c0                => audio_clock
 );

 process (audio_clock)
 begin
  if rising_edge(audio_clock) then
    if reset_n = '0' then
      lrck_divider <= (others => '0');
    elsif lrck_divider = X"BF"  then       -- "C0" minus 1
      lrck_divider <= X"00";
    else
      lrck_divider <= lrck_divider + 1;
    end if;
  end if;
 end process;

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

```
  end if;
 end process;

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

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

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

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

 -- Audio data shift output
 process (audio_clock)
 begin
  if rising_edge(audio_clock) then
   if reset_n = '0' then
    shift_out <= (others => '0');
   elsif set_lrck = '1' then          -------
    if test_mode = '1' then
                    -- soundtrack selection
                    if audio_track1 = "00" then
                            sin_out1 <=
s_shr(std_logic_vector(default(to_integer(sin_counter1))),
to_integer(audio_volume1));
```

```
                    end if;
                        sin_out3 <=
s_shr(std_logic_vector(jump(to_integer(sin_counter2))),
to_integer(audio_volume2));
                        sin_out4 <=
s_shr(std_logic_vector(collision(to_integer(sin_counter2))),
to_integer(audio_volume2));

                    if audio_track2 = "00" then
                        sin_out2 <= sin_out3;
                    elsif audio_track2 = "01" then
                        sin_out2 <= sin_out4;
                    end if;
                    shift_out <= unsigned(s_shr(sin_out1 + sin_out2,
to_integer(audio_vol)));
      else
        shift_out <= data;
      end if;
    elsif clr_bclk = '1' then
      shift_out <= shift_out (14 downto 0) & '0';
    end if;
  end if;
 end process;

  -- Audio outputs

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

  -- Self test with Sin wave
  process (audio_clock)
        begin
                if rising_edge(audio_clock) then
                        if reset_n = '0' then
                                scale_divider1 <= (others => '0');
                        elsif scale_divider1 = frq_divider1 then       -- "C0" minus 1
                                scale_divider1 <= X"0000";
                        else
                                scale_divider1 <= scale_divider1 + 1;
                        end if;
                end if;
  end process;

  process (audio_clock)
```

```
          begin
            if rising_edge(audio_clock) then
      if reset_n = '0' then
       scale_divider2 <= (others => '0');
      elsif scale_divider2 = frq_divider2 then       -- "C0" minus 1
       scale_divider2 <= X"0000";
      else
       scale_divider2 <= scale_divider2 + 1;
      end if;
     end if;
   end process;

   process(audio_clock)
   begin
    if rising_edge(audio_clock) then
      if reset_n = '0' then
         sin_counter1 <= (others => '0');
      elsif scale_divider1 = frq_divider1 - 1  then
       if sin_counter1 = "000010001111" then
         sin_counter1 <= "000000000000";
       else
         sin_counter1 <= sin_counter1 + 1;
       end if;
      end if;
     end if;
   end process;

          process(audio_clock)
          begin
            if rising_edge(audio_clock) then
            freq_lat         <= frq_divider2;
            end if;
          end process;

   process(audio_clock)
   begin
    if rising_edge(audio_clock) then
      if reset_n = '0' then
         sin_counter2 <= (others => '0');
                  elsif freq_lat /= frq_divider2 then
                           sin_counter2 <= "000000000000";
--                  elsif audio_track2 = "00" and press_delay > PRESS_DELAY_MAX - 1
then
--                           sin_counter2 <= "000000000000";
      elsif scale_divider2 = frq_divider2 - 2  then
       if sin_counter2 = "011000111111" then
```

```
        sin_counter2 <= "000000000000";
       else
        sin_counter2 <= sin_counter2 + 1;
       end if;
      end if;
    end if;
  end process;

end architecture;
```

car_B_ROM.vhd:

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

entity car1_B_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end car1_B_ROM;

architecture rtl of car1_B_ROM is
--type rom_type is array (0 to 1023) of integer;
--constant ROM : rom_type :=
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
 constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000");


begin

 process(clk)
        begin
          if rising_edge(clk) then
                data <= ROM(to_integer(addr));
          end if;
```

```
 end process;

end rtl;
```

car1_G_ROM.vhd:

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

entity car1_G_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned(3 downto 0)
 );
end car1_G_ROM;

architecture rtl of car1_G_ROM is
--type rom_type is array (0 to 1023) of integer;
--constant ROM : rom_type :=
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
 constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000");


begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

car_R_ROM.vhd:

```
library ieee;
```

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

entity car1_R_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned(3 downto 0)
  );
end car1_R_ROM;

architecture rtl of car1_R_ROM is
--type rom_type is array (0 to 1023) of integer;
--constant ROM : rom_type :=
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
  constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

car2_B_ROM.vhd:

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

entity car2_B_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
```

```
 );
end car2_B_ROM;

architecture rtl of car2_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
 constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000",
……
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000");



begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

car2_G_ROM.vhd:

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

entity car2_G_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end car2_G_ROM;

architecture rtl of car2_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
 constant ROM: rom_type := (
```

```
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");



begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

car2_R_ROM.vhd:

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

entity car2_R_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end car2_R_ROM;

architecture rtl of car2_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
 constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000");
```

```
begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

car3_B_ROM.vhd:

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

entity car3_B_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end car3_B_ROM;

architecture rtl of car3_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");


begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;
```

```
end rtl;
```

car3_G_ROM.vhd:

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

entity car3_G_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end car3_G_ROM;

architecture rtl of car3_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000",
……
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000");


begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

car3_R_ROM.vhd:

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

```
use ieee.numeric_std.all;

entity car3_R_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end car3_R_ROM;

architecture rtl of car3_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
  constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000");


begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

car4_B_ROM.vhd:

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

entity car4_B_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end car4_B_ROM;
```

```
architecture rtl of car4_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

car4_G_ROM.vhd:

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

entity car4_G_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end car4_G_ROM;

architecture rtl of car4_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
```

```
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

  process(clk)
          begin
           if rising_edge(clk) then
                   data <= ROM(to_integer(addr));
           end if;
  end process;

end rtl;
```

car4_R_ROM.vhd:

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

entity car4_R_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end car4_R_ROM;

architecture rtl of car4_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");



begin

  process(clk)
          begin
```

```
        if rising_edge(clk) then
                data <= ROM(to_integer(addr));
        end if;
 end process;

end rtl;
```

car5_1_B_ROM.vhd:

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

entity car5_1_B_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end car5_1_B_ROM;

architecture rtl of car5_1_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

  process(clk)
          begin
           if rising_edge(clk) then
                   data <= ROM(to_integer(addr));
           end if;
  end process;

end rtl;
```

car5_1_G_ROM.vhd:

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

entity car5_1_G_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
);
end car5_1_G_ROM;

architecture rtl of car5_1_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000",
……….
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

car5_1_R_ROM.vhd:

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

entity car5_1_R_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
```

```
data  : out unsigned (3 downto 0)
 );
end car5_1_R_ROM;

architecture rtl of car5_1_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

car5_2_B_ROM.vhd:

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

entity car5_2_B_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end car5_2_B_ROM;

architecture rtl of car5_2_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
```

```
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

  process(clk)
          begin
           if rising_edge(clk) then
                   data <= ROM(to_integer(addr));
           end if;
  end process;

end rtl;
```

car5_2_G_ROM.vhd:

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

entity car5_2_G_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end car5_2_G_ROM;

architecture rtl of car5_2_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");
```

```
begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

car5_2_R_ROM.vhd:

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

entity car5_2_R_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end car5_2_R_ROM;

architecture rtl of car5_2_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;
```

```
end rtl;
```

frog_dead1_B_ROM.vhd:

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

entity frog_dead1_B_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end frog_dead1_B_ROM;

architecture rtl of frog_dead1_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0100","0011","0011","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0010","0001","0010","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

frog_dead1_G_ROM.vhd:

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

entity frog_dead1_G_ROM is
  port(
```

```
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end frog_dead1_G_ROM;

architecture rtl of frog_dead1_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0110","1000","0111","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0101","0110","0101","0000","0000","0000","0000",
……….
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

frog_dead1_R_ROM.vhd:

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

entity frog_dead1_R_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end frog_dead1_R_ROM;

architecture rtl of frog_dead1_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
```

```
"0000","0000","0000","0000","0101","0100","0100","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0010","0001","0010","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

frog_dead2_B_ROM.vhd:

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

entity frog_dead2_B_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end frog_dead2_B_ROM;

architecture rtl of frog_dead2_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0100","0100","0100","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0100","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0010","0000","0000","0010","0000","0000","0000");

begin
```

```
 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

frog_dead2_G_ROM.vhd:

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

entity frog_dead2_G_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end frog_dead2_G_ROM;

architecture rtl of frog_dead2_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0110","0110","0110","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0101","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0100","0101","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0100","0101","0101","0100","0000","0000","0000");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

frog_dead2_R_ROM.vhd:

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

entity frog_dead2_R_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end frog_dead2_R_ROM;

architecture rtl of frog_dead2_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0110","0110","0110","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0101","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0001","0010","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0011","0011","0010","0011","0000","0000","0000");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

frog_jump_B_ROM.vhd:

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

entity frog_jump_B_ROM is
  port(
```

```
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end frog_jump_B_ROM;

architecture rtl of frog_jump_B_ROM is

type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"1111","1111","1111","1111","1111","1111","1111","0000","1111","1111","1111","
1111","1111","1111","1111","1111","1111","1111","1111","1111","1111","1111","1
111","1111","1111","0000","1111","1111","1111","1111","1111","1111",
……..
"1111","1111","1111","1111","1111","1111","1111","1111","1111","1111","1111","
1111","1111","1111","1111","1111","1111","1111","1111","1111","1111","1111","1
111","1111","1111","1111","1111","1111","1111","1111","1111","1111");


begin

 process(clk)
        begin
          if rising_edge(clk) then
                data <= ROM(to_integer(addr));
          end if;
 end process;

end rtl;
```

frog_jump_G_ROM.vhd:

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

entity frog_jump_G_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end frog_jump_G_ROM;

architecture rtl of frog_jump_G_ROM is
```

```
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0110","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0011","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000");


begin

  process(clk)
          begin
           if rising_edge(clk) then
                   data <= ROM(to_integer(addr));
           end if;
  end process;

end rtl;
```

frog_jump_R_ROM.vhd:

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

entity frog_jump_R_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end frog_jump_R_ROM;

architecture rtl of frog_jump_R_ROM is

type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
```

```
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000");


begin

  process(clk)
          begin
           if rising_edge(clk) then
                   data <= ROM(to_integer(addr));
           end if;
  end process;

end rtl;
```

frog_static_B_ROM.vhd:

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

entity frog_static_B_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out  unsigned (3 downto 0)
  );
end frog_static_B_ROM;

architecture rtl of frog_static_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"1111","1111","1111","1111","1111","1111","1111","1111","1111","1111","1111","
1111","1111","1111","1111","1111","1111","1111","1111","1111","1111","1111","1
111","1111","1111","1111","1111","1111","1111","1111","1111","1111",
……..
"1111","1111","1111","1111","1111","1111","1111","1111","1111","1111","1111","
1111","1111","1111","1111","1111","1111","1111","1111","1111","1111","1111","1
111","1111","1111","1111","1111","1111","1111","1111","1111","1111");


begin

  process(clk)
          begin
```

```
        if rising_edge(clk) then
                data <= ROM(to_integer(addr));
        end if;
 end process;

end rtl;
```

frog_static_G_ROM.vhd:

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

entity frog_static_G_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end frog_static_G_ROM;

architecture rtl of frog_static_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

  process(clk)
        begin
          if rising_edge(clk) then
                data <= ROM(to_integer(addr));
          end if;
  end process;

end rtl;
```

frog_static_R_ROM.vhd:

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

entity frog_static_R_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned(3 downto 0)
  );
end frog_static_R_ROM;

architecture rtl of frog_static_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000",
……
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

frogleft_B_ROM.vhd:

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

entity frogleft_B_ROM is
  port(
clk   : in std_logic;
```

```
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end frogleft_B_ROM;

architecture rtl of frogleft_B_ROM is
type rom_type is array(0 to 127) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000",
"0000","0000","0000","0000","0000","0000","0000","0000",
"0000","0000","0000","0001","0000","0000","0000","0000",
"0000","0000","0001","0010","0000","0000","0001","0010",
"0000","0000","0001","0011","0001","0010","0110","1001",
"0000","0000","0001","0011","0011","1000","1001","0101",
"0000","0000","0000","0010","0100","0111","0101","0000",
"0000","0000","0000","0000","0010","0010","0000","0000",
"0000","0000","0000","0000","0000","0010","0001","0000",
"0000","0000","0001","0001","0000","0010","0010","0000",
"0000","0000","0001","0011","0010","0010","0010","0001",
"0000","0000","0001","0011","0011","0001","0001","0011",
"0000","0000","0001","0011","0011","0001","0000","0000",
"0000","0000","0001","0011","0001","0001","0000","0000",
"0000","0000","0010","0010","0001","0000","0000","0000",
"0000","0000","0000","0000","0000","0000","0000","0000");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

frogleft_G_ROM.vhd:

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

entity frogleft_G_ROM is
```

```
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end frogleft_G_ROM;

architecture rtl of frogleft_G_ROM is
type rom_type is array(0 to 127) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000",
"0000","0000","0000","0010","0000","0000","0000","0011",
"0000","0000","0101","0111","0011","0000","0101","1001",
"0000","0000","1000","1010","0110","0000","1010","1111",
"0000","0000","1001","1011","0111","1000","1100","1111",
"0000","0000","1001","1100","1010","0101","1000","1111",
"0000","0000","0110","1010","1100","1000","1001","1111",
"0000","0000","0000","0111","1011","1101","1111","1111",
"0000","0000","0000","0000","0000","1010","1110","1111",
"0000","0000","0110","1000","0000","1001","1101","1111",
"0000","0000","1000","1010","1000","1011","1101","1110",
"0000","0000","1000","1011","1011","1100","1100","1010",
"0000","0000","0111","1010","1100","1001","0111","0110",
"0000","0000","0111","1001","1000","0101","0011","0000",
"0000","0000","0101","0110","0101","0000","0000","0000",
"0000","0000","0000","0000","0000","0000","0000","0000");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

frogleft_R_ROM.vhd:

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

```
entity frogleft_R_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end frogleft_R_ROM;

architecture rtl of frogleft_R_ROM is
type rom_type is array(0 to 127) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000",
"0000","0000","0000","0000","0000","0000","0000","0000",
"0000","0000","0000","0001","0000","0000","0000","0000",
"0000","0000","0010","0011","0000","0000","0001","0010",
"0000","0000","0010","0100","0001","0010","0110","1001",
"0000","0000","0010","0100","0100","1001","1001","0101",
"0000","0000","0001","0011","0101","1000","0101","0000",
"0000","0000","0000","0000","0011","0010","0000","0000",
"0000","0000","0000","0000","0000","0011","0010","0000",
"0000","0000","0001","0010","0000","0011","0011","0000",
"0000","0000","0010","0100","0010","0010","0010","0010",
"0000","0000","0010","0101","0100","0001","0001","0011",
"0000","0000","0010","0101","0100","0001","0000","0001",
"0000","0000","0010","0100","0010","0001","0001","0000",
"0000","0000","0010","0011","0001","0000","0000","0000",
"0000","0000","0000","0000","0000","0000","0000","0000");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

frogright_B_ROM.vhd:

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

```
entity frogright_B_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end frogright_B_ROM;

architecture rtl of frogright_B_ROM is
type rom_type is array(0 to 127) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000",
"0000","0000","0000","0000","0000","0000","0000","0000",
"0000","0000","0000","0000","0001","0000","0000","0000",
"0010","0001","0000","0000","0010","0001","0000","0000",
"1001","0110","0010","0001","0011","0001","0000","0000",
"0101","1001","1000","0011","0011","0001","0000","0000",
"0000","0110","0111","0100","0010","0000","0000","0000",
"0000","0000","0010","0010","0000","0000","0000","0000",
"0000","0001","0010","0000","0000","0000","0000","0000",
"0000","0010","0010","0000","0001","0001","0000","0000",
"0001","0010","0010","0010","0011","0001","0000","0000",
"0010","0001","0001","0011","0011","0001","0000","0000",
"0000","0000","0001","0011","0011","0001","0000","0000",
"0000","0000","0000","0001","0011","0001","0000","0000",
"0000","0000","0000","0001","0010","0001","0000","0000",
"0000","0000","0000","0000","0000","0000","0000","0000");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

frogright_G_ROM.vhd:

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

```
use ieee.numeric_std.all;

entity frogright_G_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end frogright_G_ROM;

architecture rtl of frogright_G_ROM is
type rom_type is array(0 to 127) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000",
"0011","0000","0000","0000","0010","0001","0000","0000",
"1001","0101","0000","0011","0111","0100","0000","0000",
"1111","1010","0000","0111","1010","1000","0000","0000",
"1111","1100","1000","0111","1100","1001","0000","0000",
"1111","1000","0101","1001","1100","1001","0000","0000",
"1111","1001","1000","1100","1010","0110","0000","0000",
"1111","1111","1101","1011","0110","0000","0000","0000",
"1111","1110","1010","0000","0000","0000","0000","0000",
"1111","1101","1000","0000","1001","0110","0000","0000",
"1110","1101","1010","1000","1011","1001","0000","0000",
"1010","1100","1100","1011","1011","1000","0000","0000",
"0110","0111","1001","1100","1011","1000","0000","0000",
"0000","0011","0101","1001","1010","0111","0000","0000",
"0000","0000","0000","0101","0110","0101","0000","0000",
"0000","0000","0000","0000","0000","0000","0000","0000");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

frogright_R_ROM.vhd:

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

entity frogright_R_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end frogright_R_ROM;

architecture rtl of frogright_R_ROM is
type rom_type is array(0 to 127) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000",
"0000","0000","0000","0000","0000","0000","0000","0000",
"0000","0000","0000","0000","0001","0000","0000","0000",
"0010","0001","0000","0001","0011","0010","0000","0000",
"1001","0110","0010","0001","0100","0010","0000","0000",
"0101","1001","1000","0100","0100","0010","0000","0000",
"0000","0110","1000","0101","0011","0001","0000","0000",
"0000","0000","0010","0011","0000","0000","0000","0000",
"0000","0010","0010","0000","0000","0000","0000","0000",
"0000","0011","0011","0000","0010","0001","0000","0000",
"0010","0010","0010","0010","0100","0010","0000","0000",
"0011","0001","0010","0101","0101","0001","0000","0000",
"0001","0000","0010","0101","0101","0001","0000","0000",
"0000","0001","0001","0010","0100","0001","0000","0000",
"0000","0000","0000","0010","0011","0010","0000","0000",
"0000","0000","0000","0000","0000","0000","0000","0000");
begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

homefrog_B_ROM.vhd:

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

entity homefrog_B_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end homefrog_B_ROM;

architecture rtl of homefrog_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","
0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0
010","0010","0010","0010","0010","0010","0010","0010","0010",
……................
"1100","1100","1100","1100","1100","1100","1100","1101","1101","1101","1101","
1101","1101","1101","1101","1101","1101","1101","1101","1101","1101","1
101","1101","1101","1101","1100","1100","1100","1100","1100");
begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

homefrog_G_ROM.vhd:

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

entity homefrog_G_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
```

```
 );
end homefrog_G_ROM;

architecture rtl of homefrog_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0111","0111","0111","0110","0110","0110","0110","0110","0110","0110","0110","
0110","0110","0110","0110","0110","0110","0110","0110","0110","0111","0110","0
110","0110","0110","0110","0110","0111","0111","0111","0111","0111",
……..
"0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","1000","
1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1
000","1000","1000","1000","0111","0111","0111","0111","0111","0110");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

homefrog_R_ROM.vhd:

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

entity homefrog_R_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end homefrog_R_ROM;

architecture rtl of homefrog_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
```

```
"0110","0110","0101","0101","0101","0101","0101","0101","0101","0101","0101","
0101","0110","0101","0101","0101","0101","0101","0101","0101","0110","0101","0
101","0101","0101","0101","0101","0110","0110","0110","0110",
……..
"0100","0100","0100","0101","0101","0101","0101","0101","0101","0101","0110","
0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0
110","0110","0110","0110","0101","0101","0101","0101","0100","0100");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

homeleft_B_ROM.vhd:

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

entity homeleft_B_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end homeleft_B_ROM;

architecture rtl of homeleft_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","
0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0
010","0010","0010","0010","0010","0010","0010","0010","0010",
……..
"1100","1100","1101","1100","1100","1100","1100","1100","1100","1100","1100","
1100","1100","1100","1100","1100","1100","1100","1100","1100","1100","1100","1
100","1100","1100","1100","1100","1100","1100","1100","1100");
```

```
begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

homeleft_G_ROM.vhd:

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

entity homeleft_G_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end homeleft_G_ROM;

architecture rtl of homeleft_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","
0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0
111","0110","0110","0101","0110","0111","0111","0110","0101","0101",
……..
"0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","
0111","0111","0111","0110","0110","0110","0110","0110","0110","0110","0110","0
110","0110","0110","0110","0110","0110","0110","0111","0111","0111");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;
```

```
end rtl;
```

homeleft_R_ROM.vhd:

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

entity homeleft_R_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end homeleft_R_ROM;

architecture rtl of homeleft_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0101","0101","0101","0100","0101","0101","0101","0101","0101","0101",
……..
"0101","0101","0101","0101","0101","0101","0101","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

homemiddle_B_ROM.vhd:

```
library ieee;
```

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

entity homemiddle_B_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end homemiddle_B_ROM;

architecture rtl of homemiddle_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010",
……..
"1100","1100","1100","1100","1100","1100","1100","1101","1101","1101","1101","1101","1101","1101","1101","1101","1101","1101","1101","1101","1101","1101","1101","1101","1101","1101","1101","1101","1100","1100");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

homemiddle_G_ROM.vhd:

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

entity homemiddle_G_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
```

```
end homemiddle_G_ROM;

architecture rtl of homemiddle_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0111","0111","0111","0111","0111","0111","0111","0111","0111","0110","0110","
0110","0110","0110","0110","0110","0110","0110","0110","0110","0111","0111","0
110","0110","0110","0111","0111","0111","0111","0111","0111","0111",
……..
"0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","1000","
1000","1000","1000","1000","0111","0111","0111","0111","0111","0111","0111","0
111","0111","0111","0111","0111","0111","0111","0111","0111","0110");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

homemiddle_R_ROM.vhd:

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

entity homemiddle_R_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end homemiddle_R_ROM;

architecture rtl of homemiddle_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0110","0110","0110","0110","0110","0101","0110","0110","0110","0101","0101","
0101","0101","0101","0101","0101","0101","0101","0101","0101","0110","0110","0
101","0101","0101","0110","0101","0110","0110","0110","0110","0110",
```

```
……
"0100","0100","0100","0101","0101","0101","0101","0101","0101","0101","0110","
0110","0110","0110","0110","0101","0101","0101","0101","0101","0101","0101","0
101","0101","0101","0101","0101","0101","0101","0101","0100");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

homeright_B_ROM.vhd:

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

entity homeright_B_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end homeright_B_ROM;

architecture rtl of homeright_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","
0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0
010","0010","0010","0010","0010","0010","0010","0010","0010",
……
"1100","1100","1100","1100","1100","1100","1100","1100","1100","1100","1100","
1100","1100","1100","1100","1100","1100","1100","1100","1100","1100","1100","1
100","1101","1101","1101","1101","1101","1101","1101","1101","1101");

begin

 process(clk)
        begin
```

```
        if rising_edge(clk) then
                data <= ROM(to_integer(addr));
        end if;
 end process;

end rtl;
```

homeright_G_ROM.vhd:

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

entity homeright_G_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end homeright_G_ROM;

architecture rtl of homeright_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","
0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0
111","0111","0111","0111","0111","0111","0111","0111","0111","0111",
……..
"0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","
0110","0110","0110","0110","0110","0110","0110","0110","0110","0111","0111","0
111","1000","1000","1000","0111","0111","1000","1000","1000","1000");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

homeright_R_ROM.vhd:

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

entity homeright_R_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end homeright_R_ROM;

architecture rtl of homeright_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","
0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0
110","0110","0110","0110","0110","0110","0110","0110","0110","0110",
……
"0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","
0100","0100","0100","0100","0100","0100","0100","0100","0100","0101","0101","
101","0110","0110","0110","0110","0101","0110","0110","0110","0110");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

lawn_B_ROM.vhd:

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

entity lawn_B_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
```

```
 );
end lawn_B_ROM;

architecture rtl of lawn_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","
0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0
010","0010","0010","0010","0010","0010","0010","0010","0010","0010",
……..
"0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","
0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0010","0
010","0010","0010","0010","0010","0010","0010","0010","0010","0010");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

lawn_G_ROM.vhd:

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

entity lawn_G_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end lawn_G_ROM;

architecture rtl of lawn_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","
0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0
111","0111","0111","0111","0111","0111","0111","0111","0111","0111",
```

```
……..
"0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","
0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0
111","0111","0111","0111","0111","0111","0111","0111","0111","0111");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

lawn_R_ROM.vhd:

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

entity lawn_R_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end lawn_R_ROM;

architecture rtl of lawn_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","
0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0
110","0110","0110","0110","0110","0110","0110","0110","0110",
……..
"0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","
0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0
110","0110","0110","0110","0110","0110","0110","0110","0110");

begin

 process(clk)
```

```
        begin
          if rising_edge(clk) then
                data <= ROM(to_integer(addr));
          end if;
  end process;

end rtl;
```

log_head_B_ROM.vhd:

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

entity

log_head_B_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto

0);
data  : out unsigned (3 downto 0)
  );
end log_head_B_ROM;

architecture rtl of

log_head_B_ROM is
type rom_type is array(0

to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
00

00","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","000
0"

,"0000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
00
```

```
00","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","000
0"

,"0000","0000","0000","0000","0000","0000","0000","0000","0000");

begin



process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM

(to_integer(addr));
          end if;
 end process;

end rtl;
```

log_head_G_ROM.vhd:

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

entity log_head_G_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end log_head_G_ROM;

architecture rtl of log_head_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");

begin
```

```
 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

log_head_R_ROM.vhd:

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

entity log_head_R_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end log_head_R_ROM;

architecture rtl of log_head_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");


begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;
```

```
end rtl;
```

log_mid_B_ROM.vhd:

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

entity log_mid_B_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end log_mid_B_ROM;

architecture rtl of log_mid_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

log_mid_G_ROM.vhd:

```
library ieee;
```

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

entity log_mid_G_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end log_mid_G_ROM;

architecture rtl of log_mid_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

log_mid_R_ROM.vhd:

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

entity log_mid_R_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned(3 downto 0)
  );
```

```
end log_mid_R_ROM;

architecture rtl of log_mid_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

log_tail_B_ROM.vhd:

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

entity log_tail_B_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end log_tail_B_ROM;

architecture rtl of log_tail_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
```

```
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

log_ tail _G_ROM.vhd:

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

entity log_tail_G_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end log_tail_G_ROM;

architecture rtl of log_tail_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

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

```
                data <= ROM(to_integer(addr));
        end if;
 end process;

end rtl;
```

log_ tail _R_ROM.vhd:

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

entity log_tail_R_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end log_tail_R_ROM;

architecture rtl of log_tail_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

middlezone_B_ROM.vhd:

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

```
use ieee.numeric_std.all;

entity middlezone_B_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end middlezone_B_ROM;

architecture rtl of middlezone_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"1101","1100","1100","1100","1100","1100","1100","1100","1100","1100","1100","
1101","1101","1101","1101","1101","1101","1101","1101","1101","1101","1101","1
101","1101","1101","1101","1101","1101","1101","1100","1100","1100",
……..
"0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","
0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0
111","0111","0111","0111","0111","0111","0111","0111","0111","0111");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

middlezone_G_ROM.vhd:

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

entity middlezone_G_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end middlezone_G_ROM;
```

```
architecture rtl of middlezone_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","
0111","0111","0111","0111","0111","0111","0111","0111","1000","1000","1000","1
000","1000","1000","1000","0111","0111","0111","0111","0111","0111",
………………..
"1011","1011","1011","1011","1011","1011","1011","1011","1011","1011","1011","
1011","1011","1011","1011","1011","1011","1011","1011","1011","1011","1011","1
011","1011","1011","1011","1011","1011","1011","1011","1011","1011");

begin

  process(clk)
        begin
          if rising_edge(clk) then
                data <= ROM(to_integer(addr));
          end if;
  end process;

end rtl;
```

middlezone_R_ROM.vhd:

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

entity middlezone_R_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end middlezone_R_ROM;

architecture rtl of middlezone_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","
0101","0101","0101","0101","0101","0110","0101","0101","0110","0110","0110","0
110","0110","0110","0101","0101","0101","0101","0101","0101","0101",
………………..
```

```
"1110","1110","1110","1110","1110","1110","1110","1110","1110","1110","1110","
1110","1110","1110","1110","1110","1110","1110","1110","1110","1110","1110","1
110","1110","1110","1110","1110","1110","1110","1110","1110","1110");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

roaddown_B_ROM.vhd:

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

entity roaddown_B_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end roaddown_B_ROM;

architecture rtl of roaddown_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","
1000","1000","1000","1000","0111","0101","0100","0100","0100","0100","0100","0
100","0100","0100","0100","0101","0101","0100","0100","0100","0100",
……..
"0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","
0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0
101","0101","0101","0101","0101","0101","0101","0101","0101","0101");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
```

```
            end if;
  end process;

end rtl;
```

roaddown_G_ROM.vhd:

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

entity roaddown_G_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end roaddown_G_ROM;

architecture rtl of roaddown_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","
1000","1000","1000","1000","1000","0110","0101","0101","0101","0101","0101","0
101","0101","0101","0101","0101","0101","0101","0101","0101",
……..
"1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","
1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1
010","1010","1010","1010","1010","1010","1010","1010","1010","1010");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

roaddown_R_ROM.vhd:

```
library ieee;
```

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

entity roaddown_R_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end roaddown_R_ROM;

architecture rtl of roaddown_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","0110","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101",
……..
"1001","1001","1001","1001","1001","1001","1001","1001","1001","1001","1001","1001","1001","1001","1001","1001","1001","1001","1001","1001","1001","1001","1001","1000","1000","1001","1001","1001","1001","1001","1001","1001");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

roadmain_B_ROM.vhd:

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

entity roadmain_B_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
```

```
end roadmain_B_ROM;

architecture rtl of roadmain_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100",
……..
"1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

roadmain_G_ROM.vhd:

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

entity roadmain_G_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end roadmain_G_ROM;

architecture rtl of roadmain_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101",
```

```
……..
"1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","
1000","1000","1000","1000","1000","0101","0101","0101","0101","0101","0101","0
101","0101","0101","0101","0101","0101","0101","0101","0101");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

roadmain_R_ROM.vhd:

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

entity roadmain_R_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end roadmain_R_ROM;

architecture rtl of roadmain_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","
1000","1000","1000","1000","1000","0101","0101","0101","0101","0101","0101","0
101","0101","0101","0101","0101","0101","0101","0101","0101",
……..
"1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","
1000","1000","1000","1000","1000","0101","0101","0101","0101","0101","0101","0
101","0101","0101","0101","0101","0101","0101","0101","0101");

begin

 process(clk)
```

```
        begin
          if rising_edge(clk) then
                data <= ROM(to_integer(addr));
          end if;
  end process;

end rtl;
```

roadup_B_ROM.vhd:

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

entity roadup_B_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end roadup_B_ROM;

architecture rtl of roadup_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0111","0111","0111","0111","0111","0111","1000","0111","0111","0111","0111","
0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0
111","0111","0111","0111","0111","0111","1000","1000","0111","0111",
……..
"1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","
1000","1000","1000","1000","1000","0101","0100","0100","0100","0100","0100","0
100","0100","0100","0100","0100","0100","0100","0100","0100","0100");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                  data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

roadup_G_ROM.vhd:

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

entity roadup_G_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end roadup_G_ROM;

architecture rtl of roadup_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"1011","1011","1011","1011","1011","1011","1011","1011","1011","1011","1011","
1011","1011","1011","1011","1011","1011","1011","1011","1011","1011","1011","1
011","1011","1011","1011","1011","1011","1011","1011","1011","1011",
……..
"1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","
1000","1000","1000","1000","1000","0110","0101","0101","0101","0101","0101","0
101","0101","0101","0101","0101","0101","0101","0101","0101");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

roadup_R_ROM.vhd:

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

```
entity roadup_R_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end roadup_R_ROM;

architecture rtl of roadup_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"1110","1110","1110","1110","1110","1110","1110","1110","1110","1110","1110","
1110","1110","1110","1110","1110","1110","1110","1110","1110","1110","1110","1
110","1110","1110","1110","1110","1110","1110","1110","1110",
……..
"1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","
1000","1000","1000","1000","1000","0110","0110","0101","0101","0101","0101","0
101","0101","0101","0101","0101","0101","0101","0101","0101","0101");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

startzone_B_ROM.vhd:

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

entity startzone_B_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end startzone_B_ROM;

architecture rtl of startzone_B_ROM is
```

```
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101",
……..
"0101","0101","0100","0100","0100","0100","0100","0100","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0101","0100","0101");

begin

  process(clk)
          begin
           if rising_edge(clk) then
                   data <= ROM(to_integer(addr));
           end if;
  end process;

end rtl;
```

startzone_G_ROM.vhd:

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

entity startzone_G_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end startzone_G_ROM;

architecture rtl of startzone_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010","1010",
……..
```

```
"1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","1000","
1000","1000","1001","1000","1000","1000","1000","1000","1000","1000","1000","1
000","1000","1001","1001","1000","1000","1001","1000","1000","1000");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

startzone_R_ROM.vhd:

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

entity startzone_R_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end startzone_R_ROM;

architecture rtl of startzone_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"1001","1001","1001","1001","1001","1001","1001","1001","1001","1001","1001","
1001","1001","1001","1001","1001","1001","1001","1001","1001","1001","1001","1
001","1001","1001","1001","1001","1001","1001","1001","1001","1001",
……..
"1100","1011","1011","1011","1011","1011","1011","1011","1100","1011","1011","
1011","1100","1100","1011","1011","1011","1100","1011","1100","1100","1011","1
011","1100","1100","1100","1100","1100","1100","1011","1011","1011");

begin

 process(clk)
        begin
```

```
        if rising_edge(clk) then
                data <= ROM(to_integer(addr));
        end if;
 end process;

end rtl;
```

turtle1_B_ROM.vhd:

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

entity turtle1_B_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end turtle1_B_ROM;

architecture rtl of turtle1_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
  constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

  process(clk)
          begin
           if rising_edge(clk) then
                   data <= ROM(to_integer(addr));
           end if;
  end process;

end rtl;
```

turtle1_G_ROM.vhd:

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

entity turtle1_G_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end turtle1_G_ROM;

architecture rtl of turtle1_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
  constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000");


begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

turtle1_R_ROM.vhd:

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

entity turtle1_R_ROM is
  port(
```

```
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end turtle1_R_ROM;

architecture rtl of turtle1_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
 constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

turtle2_B_ROM.vhd:

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

entity turtle2_B_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end turtle2_B_ROM;

architecture rtl of turtle2_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
 constant ROM: rom_type := (
```

```
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");


begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

turtle2_G_ROM.vhd:

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

entity turtle2_G_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end turtle2_G_ROM;

architecture rtl of turtle2_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
 constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");
```

```
begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

turtle2_R_ROM.vhd:

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

entity turtle2_R_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end turtle2_R_ROM;

architecture rtl of turtle2_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
 constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;
```

```
end rtl;
```

turtle3_B_ROM.vhd:

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

entity turtle3_B_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end turtle3_B_ROM;

architecture rtl of turtle3_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
  constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

turtle3_G_ROM.vhd:

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

```
use ieee.numeric_std.all;

entity turtle3_G_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end turtle3_G_ROM;

architecture rtl of turtle3_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
  constant ROM: rom_type := (

"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");


begin

  process(clk)
          begin
            if rising_edge(clk) then
                    data <= ROM(to_integer(addr));
            end if;
  end process;

end rtl;
```

turtle3_R_ROM.vhd:

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

entity turtle3_R_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
```

```
 );
end turtle3_R_ROM;

architecture rtl of turtle3_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
 constant ROM: rom_type := (
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000",
……..
"0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","
0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0000","0
000","0000","0000","0000","0000","0000","0000","0000","0000");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

water_B_ROM.vhd:

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

entity water_B_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end water_B_ROM;

architecture rtl of water_B_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"1100","1100","1100","1100","1100","1100","1100","1100","1100","1100","1100","
1100","1100","1100","1100","1100","1100","1100","1100","1100","1100","1100","1
100","1100","1100","1100","1100","1100","1100","1100","1100",
```

```
……..
"1100","1100","1100","1100","1100","1100","1100","1100","1100","1100","1100","
1100","1100","1100","1100","1100","1100","1100","1100","1100","1100","1100","1
100","1100","1100","1100","1100","1100","1100","1100","1100");

begin

 process(clk)
        begin
         if rising_edge(clk) then
                data <= ROM(to_integer(addr));
         end if;
 end process;

end rtl;
```

water_G_ROM.vhd:

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

entity water_G_ROM is
 port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
 );
end water_G_ROM;

architecture rtl of water_G_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","
0110","0111","0111","0111","0111","0111","0111","0111","0111","0111","0111","0
110","0110","0110","0110","0110","0110","0110","0110","0110","0110",
……..
"0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","0110","
0110","0111","0111","0111","0111","0111","0111","0111","0110","0110","0110","0
110","0110","0110","0110","0110","0110","0110","0110","0110","0110");

begin

 process(clk)
        begin
```

```
        if rising_edge(clk) then
                data <= ROM(to_integer(addr));
        end if;
 end process;

end rtl;
```

water_R_ROM.vhd:

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

entity water_R_ROM is
  port(
clk   : in std_logic;
addr  : in unsigned (9 downto 0);
data  : out unsigned (3 downto 0)
  );
end water_R_ROM;

architecture rtl of water_R_ROM is
type rom_type is array(0 to 1023) of unsigned(3 downto 0);
constant ROM: rom_type := (
"0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","
0100","0100","0011","0011","0011","0011","0011","0011","0011","0100","0100","0
100","0100","0100","0100","0100","0100","0100","0100","0100","0100",
……….....
"0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","0100","
0100","0100","0011","0011","0011","0011","0011","0011","0100","0100","0100","0
100","0100","0100","0100","0100","0100","0100","0100","0100","0100");

begin

  process(clk)
          begin
           if rising_edge(clk) then
                   data <= ROM(to_integer(addr));
           end if;
  end process;

end rtl;
```

soundtrack.vhd:

```
library ieee;
```

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

package soundtrack is

type SOUND1 is array(integer range 0 to 143) of unsigned(15 downto 0);
type SOUND2 is array(integer range 0 to 1599) of unsigned(15 downto 0);

constant default: SOUND1 := (
   X"0000",
   X"30F8",
   X"5A72",
   X"7624",
   X"7FFE",
   X"76C1",
   X"5C1C",
   X"3450",
   X"0568",
   X"D635",
   X"AD3C",
   X"8FCB",
   X"8148",
   X"82E6",
   X"93AB",
   X"B0CE",
   X"D646",
   X"FF71",
   X"27BC",
   X"4B28",
   X"66AF",
   X"7874",
   X"7FC5",
   X"7CFE",
   X"714F",
   X"5E78",
   X"4685",
   X"2B8D",
   X"0F81",
   X"F409",
   X"DA72
……..
X"fd73",
X"ff69",
X"ff1d",
X"ff1c",
```

```
X"feb4",
X"012a",
X"009d",
X"fe0b",
X"0061",
X"ff86",
X"ff92",
X"fe63",
X"ffc5",
X"fe12",
X"0094",
X"fe19",
X"ffb5",
X"fe7c",
X"ff68",
X"ffc7",
X"fe02",
X"00c1",
X"fd3d",
X"fe3e",
X"007f",
X"ff05",
X"ff04",
X"ff17");


end soundtrack;
```