Galaxian

CSEE 4840 Embedded System Design

Final Report

*Department of Computer Science
†Department of Electrical Engineering
†Department of Computer Engineering
School of Engineering and Applied Science,
Columbia University in the City of New York

May 2011

Xiaotian Huo
Computer Engineering Department
xh2144@columbia.edu

Yaolong Gao
Electrical Engineering Department
yg2258@columbia.edu

Qi Ding
Computer Science Department
qd2110@columbia.edu

Feng Ding
Electrical Engineering Department
fd2266@columbia.edu

CONTENT

ABSTRACT:

The main goal of this project is to implement a classic video game Galaxian based on the Altera Cyclone II FPGA board. The project involves both hardware set up and software programming. Moreover, this project is focused on the video signal processing and image display with the FPGA board. This design report will provide the general idea of the project and the details of the implementation.

1 INTRODUCTION:

Galaxian is a 1979 fixed shooter arcade game by Namco and released by Midway Mfg. in the US. The game is a great success on the first generation of Family Computer platform (also known as FCs). This game introduced several firsts to the game industry at that time. Although true color (as opposed to a color overlay for a game that was otherwise black and white) began appearing as early as 1975, Galaxian took graphics a step further with multi-colored animated sprites and explosions, a crude theme song, different colored fonts for the score and high score, more prominent background "music" and the scrolling starfield, and graphic icons that showed the number of ships left and how many rounds the player had completed. Due to these features, rebuilding this game becomes a perfect project to exploit the FPGA board with its powerful image processing functionality.

To implement the game, the project involves both hardware set up and software programming. Especially, due to the complex display of the game, the hardware set up will take the most of the work, including the display of images like “spaceship” and “enemy”, the arrangement of enemies in the screen and the set up of PS2 keyboard.

For the software, the difficulty lies in how to manage the objects on the screen, especially the flying path of the enemy and how they track the spaceship. The algorithm will have to handle the path of the fly, the position that avoids collision among enemies themselves and how to aim at the player.

The game starts after pressing “enter” key. After that the enemy and the spaceship will both show up. A galaxy background always moves from the top to the bottom. On the right side of the screen, game information is displayed: High score and current score of the player, number of life, and the current level. In the gameplay, player controls the ship using PS2 keyboard. Game interface is shown on the VGA. The position of swarm would fluctuate back and forth during the game. The swarm would fall out of wave one by one to attack the ship, if it is not destroyed, it would fly back to the wave. The flying path of the bullet and the swarm are depended on the current position of the ship. In other words, both swarm and the bullet rejected by them has tracing function.In the beginning, the player has 3 lives for the ship, which moves left and right at the bottom of the wraparound screen. Several rows of enemies are on the screen (formed a matrix), they would jump out of row and randomly projecting bullest to the sapceship. After one wave of swarm is destroyed, the player will face the next wave. Scores are calculated during the game, and are updated every time an enemy is destroyed. Different kind of enemy receives different level of scores. The high score is set to 5000 at first. If the player exceeds the score, the new high score will be recorded.

2 ARCHITECTURE

In this project, there are three major hardware devices, i.e. the FPGA board, the display, keyboard and sound box. To make these devices work properly, they should be connected together and set up the FPGA board. The figure below is the block diagram of the basic hardware architecture and thus makes the FPGA board programmable.

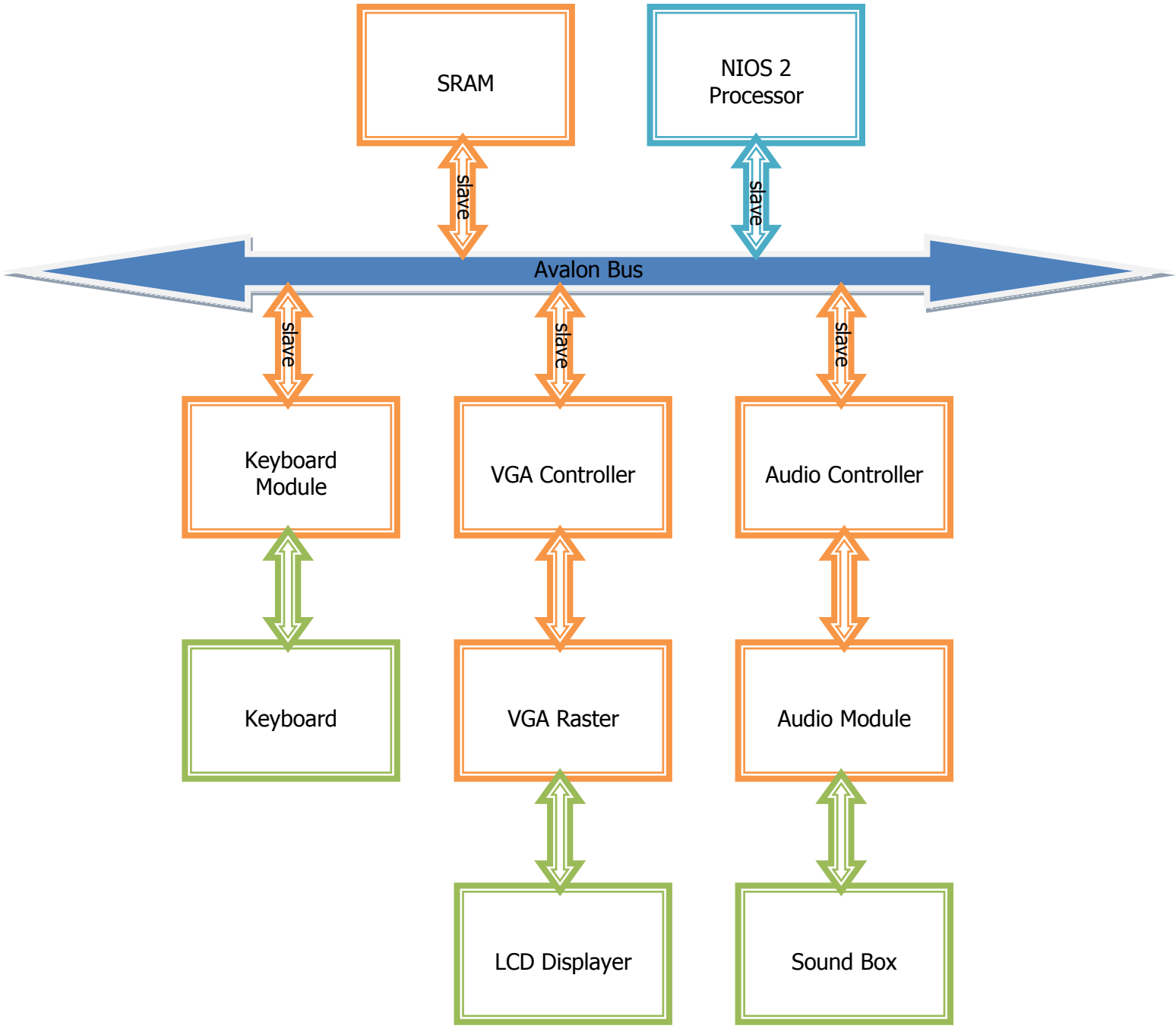


Figure 2-1 Basic Architecture

The green block in fig 2-1 indicates the real hardware devices and all the other blocks together with the Avalon Bus is the architecture of the FPGA board. Here, orange blocks stands for Avalon slaves and the blue block is the CPU. The keyboard module, VGA controller and the Audio controller are interfaces which make the block can communicate through the Avalon bus. Here, the keyboard module uses the given code which provided in Lab 3 hence its interface and control logic written in VHDL code is in a same file. The VGA controller and Audio controller are written in our project. All these three blocks will be bind to the Avalon bus with SOPC builder in the Quartus. The VGA raster is the block which actually communicates with the LCD displayer through VGA port and displays the game graphics. In the VGA block, the basic graphic patterns will be pre-set and the software will send the coordinates of positions to the VGA raster hence realize the control of the game graphics. The audio controller and audio module works in a similar way to the VGA controller and VGA raster. Software will tell the audio block when to play which kind of sound. The details of the implementation will be provided in the upcoming sections.

3 DESIGN IMPLEMENTATION

3.1 Keyboard

Our project uses keyboard as the primary input device for the user to control and play the game. To build the keyboard as a peripheral device, we take reference to the audio part of Lab 3. By using SOPC builder and include the .vhd file provided in Lab 3, we successfully bind the keyboard to the Avalon bus on the DE2 board. We only implement the read interface for the keyboard since for our project, the system do not need to write data to the keyboard such as control the LED light to indicate Caps Lock is on or off. The data length for keyboard interface is kept the same as the VGA block, which is 32-bit long.

Once the keyboard is connected to the Avalon bus, we can read the data from keyboard in the Nios2 which tells us which button(s) user presses. The following list is the buttons we set for the game.

Button	Usage
a	Control the spaceship to move left
d	Control the spaceship to move right
j	Restart the game after game over
enter	Start the game, pause/resume the game
space	Shoot bullet

Here is the detail information when we implementing these keys.

1. When implement the plane movement keys, we do not use the data read from the keyboard directly. This will cause a subtle stuck for the plane to move when user keep pressing the button without release. Instead, we put some logic condition to see if the user keeps pressing the movement button without release it. This feature improves the user experience to the game and makes the control easier.

2. We implement the feature that the user can shoot bullet while the plane is moving. There is no confliction or latency on the movement keys and space key. This is realized in a similar way as we eliminate the subtle stuck when the user keeps pressing a button as described above.

3.2 VGA

3.2.1 Several elements' pictures

Enemies

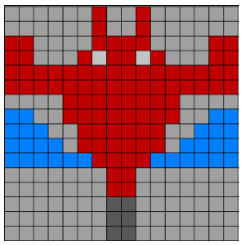


Figure 3.2.1-1 Red bee

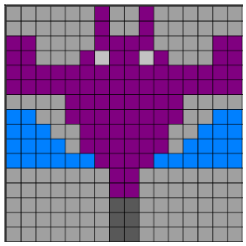


Figure 3.2.1-2 Purple bee

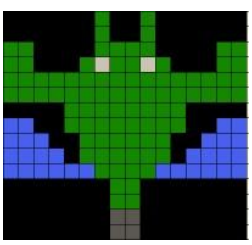


Figure 3.2.1-3 Green bee

Spaceship

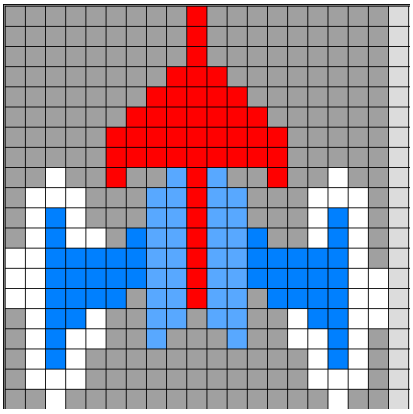


Figure 3.2.1-4 User's spaceship

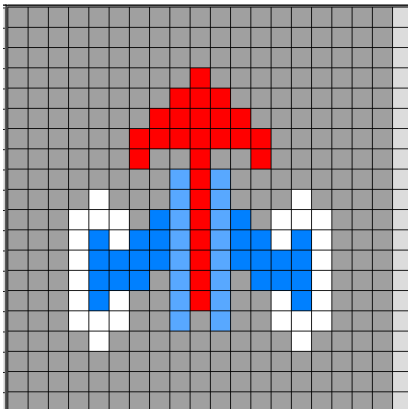


Figure 3.2.1-5 Small spaceship

Start picture

Figure 3.2.1-6 Start image

Several texts

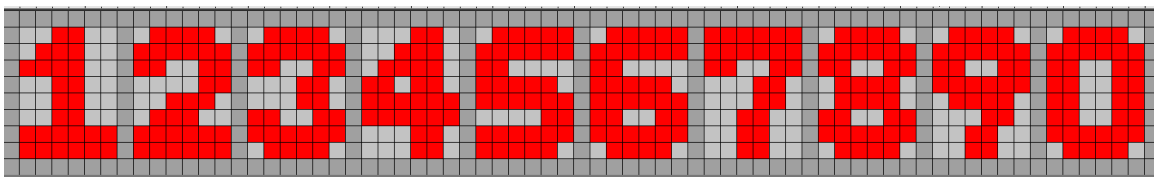


Figure 3.2.1-7 Numbers

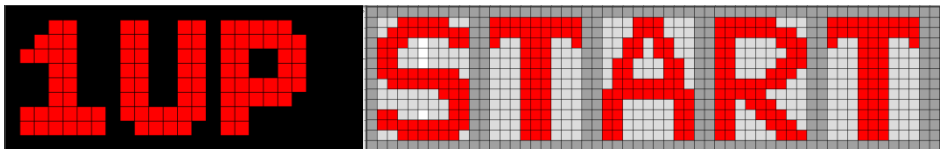


Figure 3.2.1-8 Others texts

Explosion

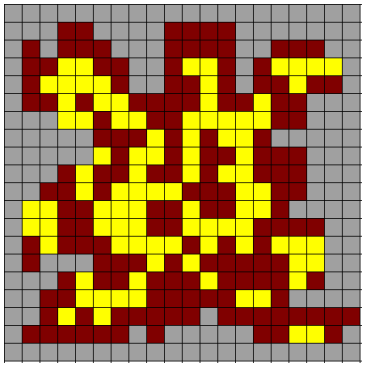


Figure 3.2.1-9 Big explosion pic

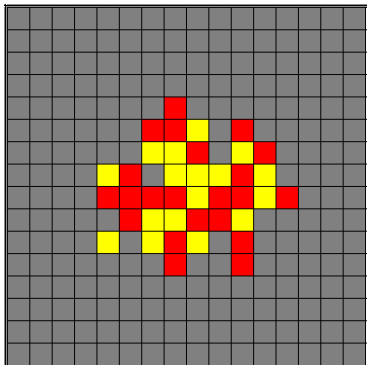


Figure 3.2.1-9 Small explosion pic

3.2.2 VGA Design

3 different bees and 1 command

In our project, we have 5 lines bees in the bee matrix. The shapes of the last 4 lines' bees are equal. However, the color of bees is different with each other. In order to save the memory and logic element in FPGA, we just used one picture for 3 different kinds of bees. In our hardware, we first receive the signal that represents the type of the bee from software. Then we draw the bee's color according to that signal.

Bee matrix

In the game Galaxian, we have to determine each bee in each position in the hardware to draw that bee correctly. In order to reduce the number of for loops in the hardware, we separated each bee in column/row just the same size of the bee (16 pixels). That means we can simply shift right 4 of the bee coordinate. The result is the number of the column/row of that bee in the matrix.

360 degrees bees

The bee, which is flying down, can rotate 360 degrees. We only used 3 pictures to show 16 different pictures in our game. In the hardware, we also receive the signal about the angle of the flying bee. We maybe reversal or switch the x coordinate with y coordinate according to the signal of degree.

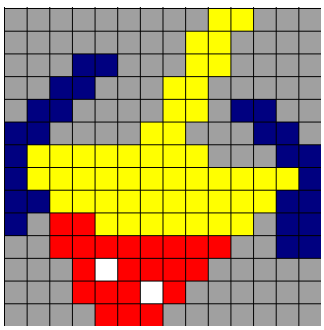


Figure 3.2.2-1 30 degrees

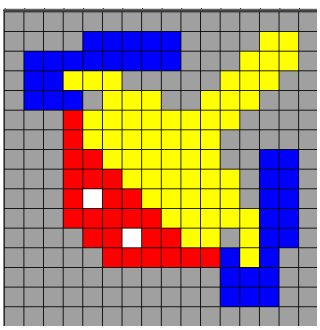


Figure 3.2.2-2 60 degrees

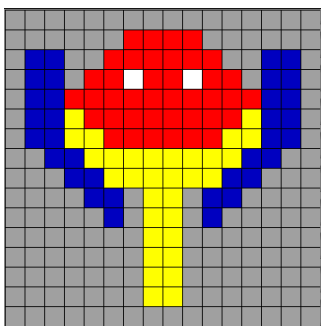


Figure 3.2.2-3 Normal pic

Connect with software

✓ Show the picture

In NIOS_ide, software will send the signal to hardware when the condition matched. However, the hardware maybe receives the signal during drawing the picture. In that case, the position of the object is changed in the hardware. Hardware will draw that picture according to the new position message. That will cause the flipping on the screen. In our project, we only receive the data of all the fast changed objects at the screen synchronization time, which is vga_vsync = '1' and vga_hsync = '1'. This means we update the data when the screen point reset.

✓ Synchronize the time

During the design of our project, we realized there were problem about the time synchronization problem between software and hardware. When there are huge numbers of calculations in the software, the hardware will be delayed by the software because the data signal is delayed by software. So, in each loop in software, the time will be different according to the number of conditions in software.

In order to solve that problem, we transform the synchronization signal at the beginning of whole while loop in software to hardware. That signal tells hardware start count times. In the hardware, time count is fixed. When hardware counts to a constant number, it will send back a signal to software. If the software finishes its job before that time, software just wait until the signal comes.

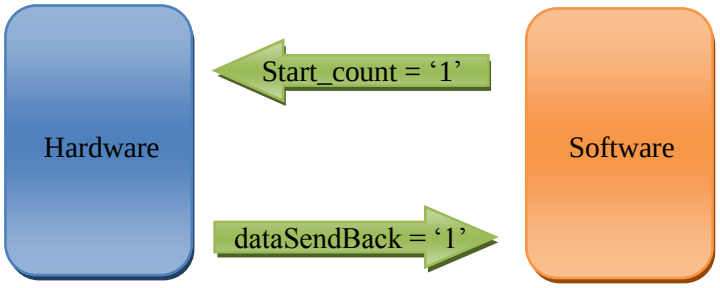


Figure 3.2.2-4 SW and HW synchronization

3.2.3 Star background:

We implemented a background with 28 stars in 4 phases. In each phase, 7 stars shine and disappear at the same time over and over again in cycles. Each star has 2 pixels in y-axis and 1 pixel in x-axis. Stars are moving from the top of the screen to the bottom again and again in iterations. To achieve a more verisimilar effect, 7 stars in the same phase are distributed "randomly" in vertical dimension and evenly in horizontal dimension. Moreover, the color of stars is changing each time they shine. When 4 phases of stars shine at the different time slot, they form a virtual galaxy.

Phase bias:

We divided 28 stars in 4 phases, each phase shine and dim at the same time. The percentage of shining time takes 50%. So whenever we see the screen, half of the stars are bright and the other half is dim.

To implement this, we set up a counter. This counter keep counting between 0 and 40000, whenever it reaches 1/4 of 40000, the first phase of stars toggles its brightness. Similarly, when the counter reaches 2/4, 3/4, 4/4 of 4000, the other three phases of stars toggles their brightness respectively.

Color changing

We didn't assign color for each star. It's easy to implement but consumes a lot of resources. Instead we assign color for each phase of stars (7 stars). But the color is keeping changing through the process. We assigned a "shine" to store the color information, start with "001". When every a star goes from dim to bright, the color information change step to step from "001" to "111". So the color appears to each phase is actually a sequence:

Purple, white, red, yellow, brown, green, blue.

Position distribution:

In order to save system resources, we don't assign position for each star – that would be 28 star vertical addresses and 28 horizontal addresses. Instead, we only assign position for the first star, and assign constant relative addresses for the rest of stars. In this way, the relative motions between stars remain zero, and every time we want to acquire all star positions, we only need to calculate them based on the first star (say this "first star" has position of (base-coordinate-x, base-coordinate-y)).

28 stars are distributed evenly in x-axis, and 4 phases of stars are alternatively distributed. To be more specific, four phases of stars' horizontal position are:

base-coordinate-x + 88*i (i = 0~6)

base-coordinate-x + 88*j **+22** (j = 0~6)

base-coordinate-x + 88*m **+44** (m = 0~6)

base-coordinate-x + 88*n **+66** (n = 0~6)

That means for each big slot of 88 pixels, 4 phases alternatively take a small slot of 22 pixels.

Vertical position bias for each star is stored in an array.

For each phase (7 stars), they always shine at the same time, so we want them more scatter.

We divided the vertical 480 pixels into 7 parts. For each phase, every star has a unique part, and for every adjacent pair of stars, we want the distance between them is as far as possible. The distribution is as follows:

y-axis	star 1	star 2	star 3	star 4	star 5	star 6	star 7
phase 1	3	6	1	7	4	2	5
phase 2	6	1	7	4	2	5	3
phase 3	1	7	4	2	5	3	6
phase 4	5	3	6	1	7	4	2

We normalize these parts, multiply by 480 pixels. Then we added a fine tuning to each phase's part, we get the vertical position bias distribution:

y-axis	star 1	star 2	star 3	star 4	star 5	star 6	star 7
phase 1	189	393	53	461	257	121	325
phase 2	423	83	491	287	151	355	219
phase 3	38	446	242	106	310	174	378
phase 4	295	159	363	23	431	227	91

Every star's vertical position can be expressed by **base-Y-coordinate + vertical bias**.

Base-Y-coordinate is the Y-coordinate of the first star.

After these steps, then the galaxy really appears to be "randomly" distributed at "random" phase with "random" colors.

3.3 Audio

In our original plan, we tried to implement the SD card interface for the audio part of the game such that we can play the music files stored in the SD card. However, we did not find enough materials to support us on doing this. Instead, we build several ROMs on the DE2 board using the tool wizard in the Quartus and store the audio files in these ROMs. The sound was first recorded from the original game. Then we use music edit software to lower the sample rate for the recorded sound. After this, we transform the wmv file into mif file using the software on the internet and finally store these mif files in the ROMs we built. To play these mif files, we did some modifications based on the audio part of Lab 3. The infrastructure of the audio implementation involves some discussion with the Battle City team.

The structure of the main functional audio blocks are shown in the below figure.

The audio_driver blocks is connected to the Avalon bus and contains the wm8371 block, which is provided in the audio part of Lab 3. The wm8371 is instantiated inside the audio_driver block. It is in the wm8371 block does the main function of audio

play. We reconfigure the structure of the wm8371 by adding a finite state machine (FSM) to control the play of each piece of music. When the play finishes, a signal will be generated and tell the software that the play finishes.

The interface of the audio_driver is also 32-bit wide. By sending command to the audio block, the software can control which piece of music to play. In this project we implement 3 kinds of music. One is the sound of the plane firing, one is the sound for explosion and the last one is for the enemy attacking. If there is a conflict to choose which sound to play, our plan is that always play the sound that happens last. This is to say, if during the enemy attacking period we fire a bullet, it will then play the sound of firing the bullet. This is also the case in the original game.

The main constrain in this method of implementation audio block is that it will consume too much memory on the DE2 board, which is the main reason for only three kind of sound. If the memory is large enough, it is easy to perform some modification and add more kinds of sound in our project.

The structure of the main functional audio blocks are shown in the below figure.

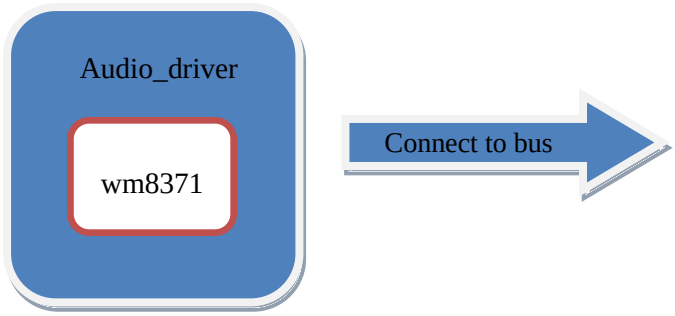


Figure 3.3-1 Audio structure

4 Software Design

4.1 Objects on the Screen

We implemented all the game logic in the software. The main moving objects in the game are 36 alien enemies (20 green, 8 purple, 6 red, and 2 command enemies) lining up in formation or flying downing the screen, one player spaceship, one bullet from the spaceship, and bullets from the enemies. Other information objects include explosion, the start picture, high score and current score, level and player life indicator, and the "game over" picture. Each type of enemy has different score and different moving speed.

Our biggest challenge is how to control the positions of these objects on the screen, especially the flying enemies. The enemy will first circle down from the formation and then fly towards the spaceship and drop bullets at the spaceship at the same time. At first, we used "$x = a + r \cdot \cos\theta, y = b + r \cdot \sin\theta$" to calculate the position of the enemy in the circle and "$y = kx + b$" for the path towards the spaceship. But the trigonometric functions and floating point computation greatly slows down the calculation and leaves the movement at an inconsistent speed. Our solution is to precompute the change of the position in the circle, store it in an array and use it for all the enemies regardless of their original positions in the formation since their relative change is all the same except for the difference of left and right direction. We keep a count for the step the enemy has taken when circling to access the element in the array. A smaller 1/4 circle array is used for turning.

In order to make the effect that the enemy is tracing the spaceship, we simplified the route by making all the enemies fly at a $45°$ angle and check their relative position to the spaceship twice during the flying down, the first check at the time when they finish the circle and the second check at the time when their vertical movement reaches 150, and then fly to the side where the spaceship is.

The similar solution is used for the bullets from the enemy. Rather than flying towards exactly where the spaceship is, the bullet moves at either $0°$ or $\arctan(0.5)$ angle depending on how far the enemy is from the spaceship at the time the bullet is dropping down. Actually, this makes the game even harder.

The maximum number of enemies flying together on the screen is 8. All the information about a flying enemy is stored in an eight-element array of type bee which is defined by us:

```
typedef struct {
   int flying;          // 1 flying, 0 in formation
   int angle;           // which direction the enemy is facing
   int flyingH;         // horizontal position
   int flyingV;         // vertical position
   int row;             // the original row in the formation
   int column;          // the original column in the formation
   int flySide;         // 1 fly from left side, -1 fly from right side
   int angleCount;      // counter used for circling step counting
   int circleCount;     // counter used for circling speed control
   int smoothCount;     // counter used for turning speed control
```

```
   int flyCount;            // counter used for flying speed control
   int flyCountToBe;        // different speed value for flyCount based on bee type
   int bulletLeftCount;     // counter used for projecting bullet
   int done;                // 1 fly back to formation after flying out, 0 keey flying
   int k;                   // 1 fly to right, -1 fly to left
   int turn;                // whether the bee is turning
   int track;               // whether the bee needs to check spaceship position
   int type;                // the type of the bee
   int bulletLeft;          // how many bullet to be projected
} bee;
```

The information of each bullet dropped by the enemy is store in a linked list of type bullet. Maximum number is 30.

```
typedef struct bullet{
   int h;                   // horizontal position
   int v;                   // vertical position
   int k;                   // 1 move right, -1 move left, 0 move down
   int number;              // the number of the bullet on the screen
   int beeBulletMoveDown;   // counter for the movement speed
   struct bullet* prevBullet;   // pointer to the previous bullet
   struct bullet* nextBullet;   // pointer to the next bullet
} bullet;
```

4.2 Animation

In order to make the flying enemy look alive rather than just a picture moving down, we need the images of the enemy heading towards 16 different directions. Here we only drew 3 images for each type of enemy. And by changing its color and convert the angle, we achieve 16 images for each type of the enemy, which saves us a lot of resource. When the flying enemy is in the circle, its head is towards the direction it's moving to. We count each step here and convert the count to the direction.

After the circle, it's always facing the spaceship. In this way, the enemy looks intelligent and agile. To find the right direction the enemy should face, we have to use floating point to calculate the slope of the line where the enemy and spaceship are on and then choose the right angle based on the value of the slope.

For the enemies in the formation, we use a five-element integer array,

```
int alive[5] = {8320, 43680, 174760, 699050, 699050};
00000010000010000000
00001010101010100000
10101010101010101010
10101010101010101010
10101010101010101010
```

each element representing a line and the binary representation of the integer indicating the enemy at the according bit is in the formation (1) or not (0). This design allows us to reform the formation easily. The entire formation will move left and right together intermittently.

When an enemy is shot by the spaceship or the spaceship is hit by a flying enemy or enemy's bullet, there is an explosion. The explosion of the enemy and spaceship goes through two phases. The explosion of the spaceship is bigger. With the explosion image getting bigger in each phase, we created the effect of a dynamic explosion.

We randomly choose the leftmost or rightmost enemies to fly down at different time interval depending on the current level of the game and the enemies still alive. Notice that the command enemy is always companied by at most two red enemies below it following certain pattern. So we add additional test when choosing an enemy to make them fly down together.

4.3 Interface Talking to the Hardware

With so many objects showing on the screen, it is very important for us to make full use of both the 32-bit data field and the 5-bit address field efficiently. To write data to vga, we use IOWR_32DIRECT(VGA_BASE, address, data). Reading data IORD_32DIRECT(VGA_BASE, 0) is only used to receive the synchronization signal from the hardware in order to control the tempo of the entire while loop. To write data to audio, we use IOWR_32DIRECT(AUDIO_BASE, 0, data). After sending a sound signal to audio, we also send a stop signal so that the sound won't be played repeatedly. We use IORD_8DIRECT(PS2_BASE, offset) to read data from the keyboard.

The detailed interface for VGA is as following:

<table>
<tr><td colspan="2">address</td><td colspan="3">data</td><td rowspan="2">utility</td></tr>
<tr><td>hardware</td><td>software</td><td>flags (31-20)</td><td>19-10</td><td>9-0</td></tr>
<tr><td>01101</td><td>52</td><td></td><td></td><td></td><td>Synchronization</td></tr>
<tr><td>01100</td><td>48</td><td>angle(30-27) + type(24-23) + number(22-20)</td><td>flyingH</td><td>flyingV</td><td>Flying Enemy</td></tr>
<tr><td>01010</td><td>40</td><td>Number (25-20)</td><td>h</td><td>v</td><td>Enemy Bullet</td></tr>
<tr><td>01000</td><td>32</td><td></td><td>beeMaxH</td><td>beeMaxV</td><td>Formation</td></tr>
<tr><td>00111</td><td>28</td><td></td><td>planeH</td><td>planeV</td><td>Spaceship</td></tr>
<tr><td>00110</td><td>24</td><td></td><td>bullet</td><td>bullet</td><td>Player Bullet</td></tr>
<tr><td>00101</td><td>20</td><td>The 5th line (24), the 4th line (23), the 3rd line (22), the 2nd line (21), the 1st line (20)</td><td colspan="2">alive</td><td>Enemy in Formation</td></tr>
<tr><td rowspan="3">00100</td><td rowspan="3">16</td><td>1</td><td colspan="2">startPicV</td><td>Start Screen</td></tr>
<tr><td>2</td><td colspan="2">level (5-3) + player life (2-0)</td><td>Level & Player Life</td></tr>
<tr><td>3</td><td colspan="2"></td><td>Clear Screen</td></tr>
</table>

		4	1 (show) , 0 (hide)		Ready
		5	1 (show) , 0 (hide)		Pause
		6	h	v	G
		7	h	v	A
		8	h	v	M
		9	h	v	E
		10	h	v	O
		11	h	v	V
		12	h	v	E
		13	h	v	R
00011	12	small explosion pic 1; otherwise, 0	expH	expV	Spaceship Explosion
00010	8	small explosion pic 1; otherwise, 0	expH	expV	Enemy Explosion
00001	4		1s(19-16) + 10s(15-12) + 100s(11-8) + 1000s(7-4) + 10000s(3-0)		High Score
00000	0		1s(19-16) + 10s(15-12) + 100s(11-8) + 1000s(7-4) + 10000s(3-0)		Current Score

The direction of the enemy is as below.

Angle	Decimal representation
350 – 360, 0 – 10 (straight up)	0
10 – 30 (left up 30)	4
30 – 60 (left up 45)	8
60 – 80 (left up 60)	12
80 – 100 (left)	1
100 – 120 (left down 30)	14
120 – 150 (left down 45)	10
150 – 170 (left down 60)	6
170 – 190 (down)	2
190 – 210 (right down 60)	7
210 – 240 (right down 45)	11
240 – 260 (right down 30)	15
260 – 280 (right)	3
280 – 300 (right up 60)	13
300 – 330 (right up 45)	9
330 – 350 (right up 30)	5

The type of the enemy is as below:

Type	Decimal representation
Green Enemy	0
Purple Enemy	1
Red Enemy	2
Command Enemy	3

5 Work Division and Lessons Learned

Xiaotian: vga architecture, image, synchronization

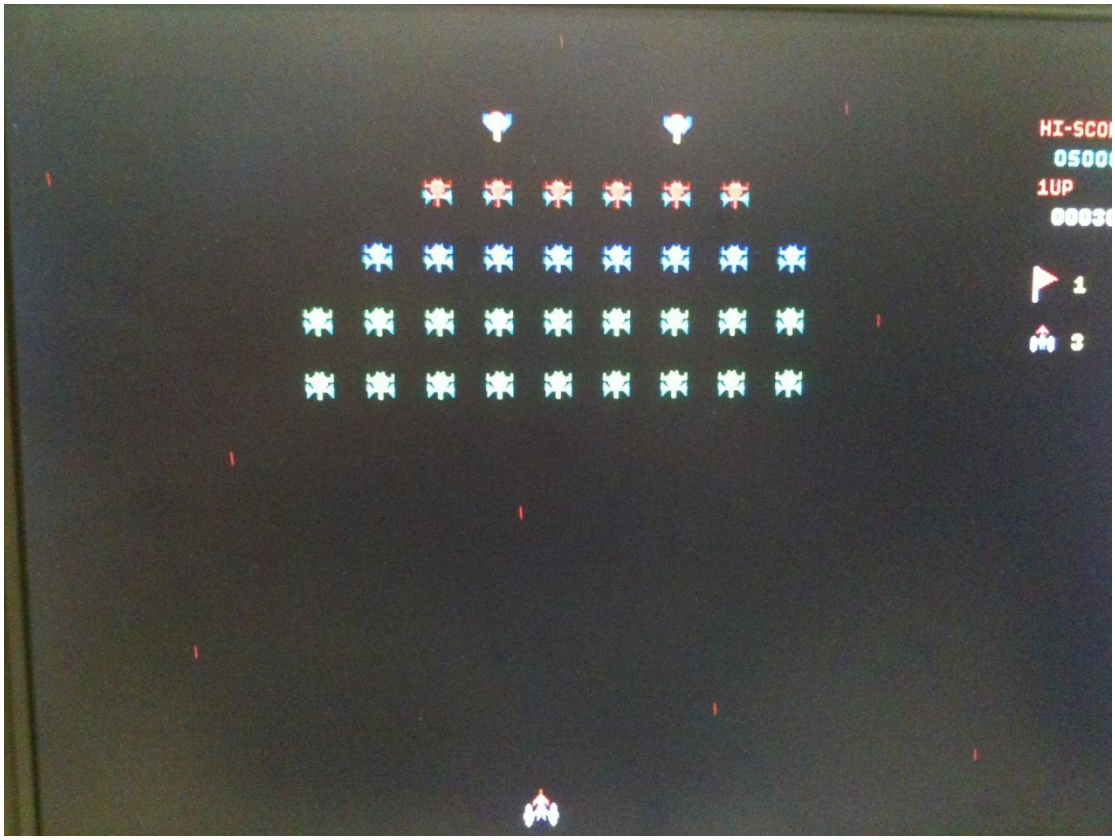
Feng: star background design, image processing

Yaolong: ps2 keyboard, audio implementation

Qi: software, communication with hardware

The division of the duty is important. When the team member knows what he/she needs to do clearly, the development can be very efficient. Code should be well commented if the meaning is not clear. Because we have thousands of lines, it helps other people to understand and can also help the one who wrote the code to pick up the idea after a long time. Finish the milestone on time so that you can have an easy life at the end. Use the most simple way to solve problems, even it looks naïve, because the simplest is likely to be the most efficient. Have fun with the project!

Thanks for Prof. Edwards and our TA Sungjun Kim for all the help and suggestion!



6 Codes

6.1 de2_vga_raster.vhd

```
--------------------------------------------------------------------------------
-- Simple VGA raster display
--
-- VGA Design for Galaxian
--
--------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;

entity de2_vga_raster 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(4 downto 0);
		readdata	: out unsigned(31 downto 0);
		writedata	: in  unsigned(31 downto 0);

		VGA_CLK,  		-- Clock
		VGA_HS,  		-- H_SYNC
		VGA_VS,  		-- V_SYNC
		VGA_BLANK,  		-- BLANK
		VGA_SYNC : out std_logic;	-- SYNC
		VGA_R,  		-- Red[9:0]
		VGA_G,  		-- Green[9:0]
		VGA_B : out std_logic_vector(9 downto 0) -- Blue[9:0]
	);

end de2_vga_raster;

architecture rtl of de2_vga_raster is
	constant CoorGlaH : integer := 145;
	constant CoorGlaV : integer := 100;
	constant GLA_LONG: integer := 250;
	constant GLA_HEIGHT: integer := 300;
	signal glaH, glaV, glaG, glaColorG : std_logic;
	signal galaxianColor : unsigned (2 downto 0);

	-- 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;

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


signal b1V, b1H, b2V, b2H, b3V, b3H :unsigned (9 downto 0);
signal Rb1V, Rb1H, Rb2V, Rb2H, Rb3V, Rb3H :unsigned (9 downto 0);
signal b1Hshow, b1Vshow, b2Hshow, b2Vshow, b3Hshow, b3Vshow : std_logic;
signal b1, b2, b3 :std_logic;

constant BULLET_LONG : integer := 1;
constant BULLET_HEIGHT   : integer := 3;

signal clk25 : std_logic := '0';

signal TransColorSignal : unsigned (2 downto 0);

----------signal for the plane----------
signal planeH, planeV, planeG :std_logic;
signal CoorPlaneH, TCoorPlaneH : unsigned (9 downto 0)          := "0000011001";
signal CoorPlaneV, TCoorPlaneV : unsigned (9 downto 0)          := "0110010000";
constant PLANE_SIZE : integer := 20;

----------signal for the bee----------
signal bee_GreenG, bee_PurpleG, bee_RedG, big_beeG :std_logic;
signal CoorBeeH : unsigned (9 downto 0)          := "0000011001";
signal CoorBeeV : unsigned (9 downto 0)          := "0001100100";
constant BEE_SIZE : integer := 16;

----------signal for the flying bee----------
type CoorBeeSet          is array (0 to 7) of unsigned (9 downto 0);
type BeeAngleSet         is array (0 to 7) of unsigned (3 downto 0);
type BeeTypeSet          is array (0 to 7) of unsigned (1 downto 0);
signal FlybeeH, FlybeeV, FlybeeGback, FlybeeG : unsigned (0 to 7);
signal FlyBeeType: BeeTypeSet:= (
("00"),("00"),("00"),("00"),("00"),("00"),("00"),("00"));
signal TCoorFlyBeeH, CoorFlyBeeH : CoorBeeSet:= (
("0000011001"),("0000011001"),("0000011001"),("0000011001"),
("0000011001"),("0000011001"),("0000011001"),("0000011001"));
signal TCoorFlyBeeV, CoorFlyBeeV : CoorBeeSet:= (
("0001100100"),("0001100100"),("0001100100"),("0001100100"),
("0001100100"),("0001100100"),("0001100100"),("0001100100"));
signal FlyBeeAngle : BeeAngleSet:= (
("0000"),("0000"),("0000"),("0000"),("0000"),("0000"),("0000"),("0000"));

----------signal for the Explode----------
signal ExplodeH, ExplodeV, ExplodeG :std_logic;
signal BigExplodeH, BigExplodeV, BigExplodeG :std_logic;
```

```
signal CoorExplodeH, planeExplodeH : unsigned (9 downto 0)        := "0000000000";
signal CoorExplodeV, planeExplodeV : unsigned (9 downto 0)        := "0000000000";
signal TCoorExplodeH, TplaneExplodeH: unsigned (9 downto 0) := "1001011000";
signal TCoorExplodeV, TplaneExplodeV: unsigned (9 downto 0) := "0000000000";
signal Small: std_logic := '0';
signal planeSmall: std_logic:= '0';

----------signal for the bee matrix----------
signal beeMaxH, beeMaxV, beeMaxG :std_logic;
signal CoorBeeMaxH : unsigned (9 downto 0)                  := "0000110010";
signal CoorBeeMaxV : unsigned (9 downto 0)                  := "0000110010";
constant BEEMAX_LONG : integer := 304;
constant BEEMAX_HEIGHT : integer := 144;

----------signal for the box----------
constant BOX_LONG : integer := 100;
constant BOX_HEIGHT: integer := 480;
signal boxG : std_logic;

----------signal for the data send back----------
signal dataSendBack: unsigned (31 downto 0);

----------signal for time delay count----------
signal timeDelayCount: integer := 0;
signal windFlipCount: integer := 0;
signal startCount: std_logic := '0';
constant synctime: integer := 60000;
signal windFlip: std_logic := '0';

----------alive matrix----------
type alive_max_type is array (0 to 8) of unsigned (0 to 19);
signal AliveMax: alive_max_type := (
("10101010101010101000"),("00000000000000000000"),
("10101010101010100010"),("00000000000000000000"),
("10101010101010001010"),("00000000000000000000"),
("10101010101010001010"),("00000000000000000000"),
("10101010101000101010"));

signal tmpM1, tmpM2, tmpM3, tmpM4, tmpM5 : unsigned (19 downto 0) :=
"10101010101010101010";

----------bee type matrix----------
type bee_type_max is array (0 to 8) of unsigned (0 to 1);
constant BeeTypeMax : bee_type_max := (
("11"),("00"),("10"),("00"),("01"),("00"),("00"),("00"),("00"));

----------score data----------
type score_type is array (0 to 4) of unsigned (3 downto 0);
signal hiScoreData, scoreData : score_type := (
("0000"),("0101"),("0000"),("0000"),("0000"));

----------bee bullet----------
type bee_bullet_type is array (0 to 29) of unsigned (9 downto 0);
signal CoorBeeBulletH, CoorBeeBulletV :bee_bullet_type;
signal TCoorBeeBulletH, TCoorBeeBulletV :bee_bullet_type;
signal beeBulletH, beeBulletV, beeBulletG: unsigned (0 to 29);
```

```
        signal getBullet: std_logic;

        ----------------signal for the star-------------------
        signal flipstate : std_logic :='0';
        signal flipcount : integer := 0;
        signal flipstate1 : std_logic := '0';
        signal flipstate2 : std_logic := '1';
        signal flipstate3 : std_logic := '1';

        type starposition is array(integer range 0 to 27) of integer;
        signal stararrayV :starposition:= (189,393,53,461,257,121,325,423,
        83,491,287,151,355,219,38,446,242,106,310,174,378,295,159,363,23,431,227,91);
        signal roll :starposition:= (189,393,53,461,257,121,325,423,
        83,491,287,151,355,219,38,446,242,106,310,174,378,295,159,363,23,431,227,91);
        signal coorstarH : unsigned (9 downto 0) := "1001101010";
        signal coorstarV : unsigned (9 downto 0) := "0000011000";
        signal starH, starV, starG :std_logic;
        signal shinecount : integer :=1;
        signal shine : unsigned (2 downto 0):= "001";
        ----------------signal for the star1-------------------
        signal starH1, starV1, starG1 :std_logic;
        signal shinecount1 : integer :=1;
        signal shine1 : unsigned (2 downto 0):= "011";
        ----------------signal for the star2-------------------
        signal starH2, starV2, starG2 :std_logic;
        signal shinecount2 : integer :=1;
        signal shine2 : unsigned (2 downto 0):= "101";
        ----------------signal for the star3-------------------
        signal starH3, starV3, starG3 :std_logic;
        signal shinecount3 : integer :=1;
        signal shine3 : unsigned (2 downto 0):= "111";

        ----------------------color signal-------------------
        type textMatrix1 is array(0 to 9) of unsigned (0 to 54);
        type textMatrix2 is array(0 to 9) of unsigned (0 to 21);
        type normalMatrix is array(integer range 0 to 15, integer range 0 to 15) of unsigned(1
downto 0);
        type matrix is array(integer range 0 to 15, integer range 0 to 15) of unsigned(2 downto 0);
        type matrix24 is array(integer range 0 to 19, integer range 0 to 19) of unsigned(2 downto
0);

        ------------------------information--------------------
        signal mainPic: std_logic := '1';
        signal planeLife : unsigned (2 downto 0);
        signal level : unsigned (2 downto 0);
        signal mainPicV: unsigned (9 downto 0) := "0001100100";
        signal TmpClearScr, clearScr : std_logic := '0';
        signal readySignal, pauseSignal : std_logic := '0';
        signal readyH, readyV, readyG : std_logic;
        signal pauseH, pauseV, pauseG : std_logic;
        signal readyColor, pauseColor: unsigned (2 downto 0);
        constant CoorReadyH: integer:= 250;
        constant CoorReadyV: integer:= 350;
        constant READY_LONG: integer:= 50;
        constant READY_HEIGHT: integer:= 14;
```

```
--------------------------the matrix for text-------------------------------------
signal hiScore: textMatrix1 := (
("00000000000000000000000000000000000000000000000000000000"),
("01100110111111000000011110001111000111001111001111110"),
("01100110111111000000111111011111101111110111111101111110"),
("01100110001100000000110000011001101100110110011011000000"),
("01111110001100011110111110011000001100110110011011111110"),
("01111110001100011110011111011000001100110111110011111110"),
("01100110001100000000000011011001101100110111110011000000"),
("01100110111111000000111111011111101111110110111011111110"),
("01100110111111000000011110001111000111001101101111110"),
("01111110011111001100000"),("00000000000000000000000"));

constant CoorTextH : integer := 550;
constant CoorTextV : integer := 50;
constant textMatrixLong : integer := 55;
constant textMatrixHeight : integer := 55;
signal TextG, TextH, TextV :std_logic;

---------------------------number---------------------------
signal hiScoreColorSignal, scoreColorSignal : unsigned (2 downto 0);
type numberMatrix is array(0 to 9) of unsigned (0 to 5);
signal one: numberMatrix := (
("000000"),("001100"),("011100"),("111100"),("001100"),
("001100"),("001100"),("111111"),("111111"),("000000"));
signal two: numberMatrix := (
("000000"),("011110"),("111111"),("110011"),("000011"),
("001110"),("011100"),("111111"),("111111"),("000000"));
signal three: numberMatrix := (
("000000"),("011110"),("111111"),("110011"),("000110"),
("000110"),("110011"),("111111"),("011110"),("000000"));
signal four: numberMatrix := (
("000000"),("000110"),("001110"),("011110"),("110110"),
("111111"),("111111"),("000110"),("000110"),("000000"));
signal five: numberMatrix := (
("000000"),("011111"),("111111"),("110000"),("111110"),
("111111"),("000011"),("111111"),("111110"),("000000"));
signal six: numberMatrix := (
("000000"),("011111"),("111111"),("110000"),("111110"),
("111111"),("110011"),("111111"),("011110"),("000000"));
signal seven: numberMatrix := (
("000000"),("111111"),("111111"),("110011"),("000110"),
("000110"),("001100"),("001100"),("001100"),("000000"));
signal eight: numberMatrix := (
("000000"),("011110"),("111111"),("110011"),("011110"),
("011110"),("110011"),("111111"),("011110"),("000000"));
signal nine: numberMatrix := (
("000000"),("011110"),("111111"),("110011"),("111111"),
("011110"),("000110"),("001100"),("001100"),("000000"));
signal zero: numberMatrix := (
```

```
("000000"),("011110"),("111111"),("110011"),("110011"),
("110011"),("110011"),("111111"),("011110"),("000000"));

-------------------------start---------------------------------
type startType is array(0 to 13) of unsigned (0 to 49);
signal start:startType := (
("00000000000000000000000000000000000000000000000000"),
("00011111000111111110000111100001111111000111111110"),
("00111111100111111111000111111000111111110011111110"),
("01110001100100110010001100110001100001100100110010"),
("01100000000000110000001100110001100001100000110000"),
("01100000000000110000001100110001100011100000110000"),
("01111111000000110000001111111001111111000000110000"),
("00111111100000110000011111111001111110000000110000"),
("00000001100000110000011000011001111100000000110000"),
("00000001100000110000011000011001101110000000110000"),
("01100011100000110000011000011001100111000000110000"),
("01111111000000110000011000011001100011100000110000"),
("00111110000000110000011000011001100001100000110000"),
("00000000000000000000000000000000000000000000000000"));

signal ready:startType := (
("00000000000000000000000000000000000000000000000000"),
("01111111000111111110000111100001111110000110000110"),
("01111111100111111111000111111000111111110000110000110"),
("01100001100110000000001100110001100011100110000110"),
("01100001100110000000001100110001100001100110000110"),
("01100011100110000000001100110001100001100011001100"),
("01111111000111111110001111111000110000110001111100"),
("01111110000111111110011111111001100001100001111000"),
("01111100000110000000011100111001100001100000110000"),
("01101110000110000000011000011001100001100000110000"),
("01100111000110000000011000011001100011100000110000"),
("01100011100111111111001100001100111111100000110000"),
("01100001100111111111001100001100111111100000110000"),
("00000000000000000000000000000000000000000000000000"));

signal pause:startType := (
("00000000000000000000000000000000000000000000000000"),
("01111111000011110000110000110000111110001111111110"),
("01111111100011111110001100001100011111110011111110"),
("01100001100011001100011000011001110001100110000000"),
("01100001100011001100011000011001100000000110000000"),
("01100001100011001100011000011001100000000110000000"),
("01111111100011111110001100001100111111100011111110"),
("01111111000111111110011000011000111111100111111110"),
("01100000000110000110011000011000000001100110000000"),
("01100000000110000110011000011000000001100110000000"),
("01100000000110000110011000011001100011100110000000"),
("01100000000110000110011111111001111111000111111110"),
("01100000000110000110001111110000111110000111111110"),
("00000000000000000000000000000000000000000000000000"));

------------------------gameover-----------------------------
type alphaMatrix is array(0 to 13) of unsigned (0 to 9);
signal g: alphaMatrix := (
```

```
("0000000000"),
("0001111000"),
("0011111100"),
("0110000110"),
("0110000110"),
("0110000000"),
("0110000000"),
("0110111100"),
("0110111110"),
("0110000110"),
("0110000110"),
("0011111110"),
("0001111100"),
("0000000000"));
signal a: alphaMatrix := (
("0000000000"),
("0001111000"),
("0011111100"),
("0011001100"),
("0011001100"),
("0011001100"),
("0011111100"),
("0111111110"),
("0110000110"),
("0110000110"),
("0110000110"),
("0110000110"),
("0110000110"),
("0000000000"));
signal m: alphaMatrix := (
("0000000000"),
("0110000110"),
("0110000110"),
("0111001110"),
("0111001110"),
("0111111110"),
("0110110110"),
("0110110110"),
("0110000110"),
("0110000110"),
("0110000110"),
("0110000110"),
("0110000110"),
("0000000000"));
signal e: alphaMatrix := (
("0000000000"),
("0111111110"),
("0111111110"),
("0110000000"),
("0110000000"),
("0110000000"),
("0111111110"),
("0111111110"),
("0110000000"),
("0110000000"),
("0110000000"),
```

```
("0111111110"),
("0111111110"),
("0000000000"));
signal o: alphaMatrix := (
("0000000000"),
("0001111000"),
("0011111100"),
("0011001100"),
("0110000110"),
("0110000110"),
("0110000110"),
("0110000110"),
("0110000110"),
("0011001100"),
("0011111100"),
("0001111000"),
("0000000000"));
signal v: alphaMatrix := (
("0000000000"),
("0110000110"),
("0110000110"),
("0110000110"),
("0110000110"),
("0110000110"),
("0011001100"),
("0011001100"),
("0011001100"),
("0011001100"),
("0000110000"),
("0000110000"),
("0000110000"),
("0000000000"));
signal r: alphaMatrix := (
("0000000000"),
("0111111100"),
("0111111110"),
("0110000110"),
("0110000110"),
("0110001110"),
("0111111100"),
("0111111000"),
("0111110000"),
("0110111000"),
("0110011100"),
("0110001110"),
("0110000110"),
("0000000000"));

signal gH, tmpgH, aH, tmpaH, mH, tmpmH, e1H, tmpe1H,
        oH, tmpoH, vH, tmpvH, e2H, tmpe2H, rH, tmprH: unsigned (9 downto 0);
signal gV, tmpgV, aV, tmpaV, mV, tmpmV, e1V, tmpe1V,
        oV, tmpoV, vV, tmpvV, e2V, tmpe2V, rV, tmprV: unsigned (9 downto 0);
signal gHG, aHG, mHG, e1HG, oHG, vHG, e2HG, rHG: std_logic;
signal gVG, aVG, mVG, e1VG, oVG, vVG, e2VG, rVG: std_logic;
signal gG, aG, mG, e1G, oG, vG, e2G, rG: std_logic;
```

```
	constant ALPHALONG : integer := 10;
	constant ALPHAHEIGHT: integer := 14;
	signal gameoverSignal : std_logic;
	signal gameoverColorSignal : unsigned (2 downto 0);

	type xiaotianType is array(0 to 13) of unsigned (0 to 139);
	signal xiaotian: xiaotianType:= (
	("00000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000"),
	("01100001100011111100000111100000111100001111111100011111100000111100
011100011000000000000000000000000000000000000000000001111000001111110"),
	("01100001100011111100001111110000111111000111111110001111110000111111100
011100011000000000000000000000000000000000000000000011111100001111110"),
	("01100001100000110000001100110000110011000100110010000011000000110000110
011110011000000000000000000000000000000000000000000011000011000001100"),
	("00110011000000110000001100110001100001100000110000000011000000110011000
011110011000000000000000000000000000000000000000000011000011000001100"),
	("00111111000000110000001100110001100001100000110000000011000000110011000
011111011000000000000000000000000000000000000000000011000011000001100"),
	("00001100000000110000001111111001100001100000110000000011000000111111000
011011011000000000000000000000000000000000000000000011000011000001100"),
	("00001100000000110000011111111001100001100000110000000011000000111111110
011011011000000000000000000000000000000000000000000011011011000001100"),
	("00111111000000110000011000011001100001100000110000000011000000110000110
011001111000000000000000000000000000000000000000000011011011000001100"),
	("00110011000000110000011000011001100001100000110000000011000000110000110
011001111000000000000000000000000000000000000000000011001111000001100"),
	("01100001100000110000011000011001100000011000000011000000110000110
011001111000000000000000000000000000000000000000000011101111000001100"),
	("01100001100011111100011000011000111111000000110000001111111000110000110
011000111000000000000000000000000000000000000000000011111111000111100"),
	("01100001100011111100011000011000111111000000110000001111111000110000110
011000111000000000000000000000000000000000000000000011101100011111100"),
	("00000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000"));

	type yaolongType is array(0 to 13) of unsigned (0 to 159);
	signal yaolong: yaolongType:= (
	("00000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000"),
	("01100001100001111000000111100001100000000001111000011100011000011110000
000000000000000000000000000000000000000000000000000011111111001111111100111000110
00001111000"),
	("01100001100011111100001111110001100000000011111000111000110000111110000
000000000000000000000000000000000000000000000000000011111111001111111100111000110
00011111100"),
	("01100001100011001100001100110001100000000011001100011110011001100001100
000000000000000000000000000000000000000000000000000011000000011000000001111001100
00110000110"),
	("01100001100011001100001100110001100000000011000011001111001100110000110
000000000000000000000000000000000000000000000000000011000000011000000001111001100
00110000110"),
	("00110011000011001100011000011001100000000011000011001111101100110000000
000000000000000000000000000000000000000000000000000011000000011000000001111101100
00110000000"),
```

```
        ("00111111000011111100011000011001100000000110000110011011011001100000000
00000000000000000000000000000000000000000000000000000001111111000111111100110110110
00110000000"),
        ("00011110000111111110011000011001100000000110000110011011011001100111100
00000000000000000000000000000000000000000000000000000001111111000111111100110110110
00110111100"),
        ("00001100000110000110011000011001100000000110000110011001111001100111110
00000000000000000000000000000000000000000000000000000001100000000110000000011001111
00110111110"),
        ("00001100000110000110011000011001100000000110000110011001111001100000110
00000000000000000000000000000000000000000000000000000001100000000110000000011001111
00110000110"),
        ("00001100000110000110001100110001100000000110011000110011111001100000110
00000000000000000000000000000000000000000000000000000001100000000110000000011001111
00110000110"),
        ("00001100000110000110001111110001111111100011111100011000111000111111110
00000000000000000000000000000000000000000000000000000001100000000111111100110001111
00011111110"),
        ("00001100000110000110000111100001111111100001111000011000111000011111100
00000000000000000000000000000000000000000000000000000001100000000111111100110001111
00001111100"),
        ("00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000"));

        ----------trademarkMatrix----------
        type trademarkMatrix is array(0 to 14) of unsigned (0 to 127);
        signal trademark: trademarkMatrix := (
        ("00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000"),
        ("00000000000000000000000000000000000000000000000000000011111000001111100001
11111100111111110000011110000111111000000111000011111100"),
        ("00001111100000000000000000000000000000000000000000001111111000111111100
11111100111111110000011110000111111100000111000111111110"),
        ("00010000010000000000000000000000000000000000000000011100011001110001100
10000000110000000000011011000110000110000110110001100000110"),
        ("00100111001000001111000111100001100000110000000001100000000110000000011
10000000110000000000011011000110000110000110110001100000110"),
        ("01001000100100011111101111110011100001110000000001100000000110000000011
10000000110000000001100110001100001100011001100011000000110"),
        ("01010000000100011001101100110111100011110000000001100000000111111100011
11111100111111110001100110000111111000011001100011000000110"),
        ("01010000000100000011011001100011000001100000000011000000000111111100011
11111100111111110011000110000111111000110001100011000000110"),
        ("01010000000100000111001100110001100000110000000001100000000000001100111
10000000110000000011111111001100001100111111110011000000110"),
        ("01001000100100001110001100110001100000110000000001100000000000001100111
10000000110000000011111111001100001100111111110011000000110"),
        ("00100111001000011111101111110111111101111110000000111000110011000111001
10000000110000000000001100011000011000000011000110000110"),
        ("00010000010000011111100111100111111101111110000000011111110011111110001
11111100111111110000000110000111111000000001100011111110"),
        ("00001111100000000000000000000000000000000000000000011111000011111000001
11111100111111110000000110000111111000000001100001111110"),
        ("00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000"),
```

```
	("0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000"));

	---------------------------the matrix for the plane and bees------------------------
	signal plane :matrix24:= (
	("000","000","000","000","000","000","000","000","000","011","000","000","000","000"
,"000","000","000","000","000","000"),
	("000","000","000","000","000","000","000","000","000","011","000","000","000","000"
,"000","000","000","000","000","000"),
	("000","000","000","000","000","000","000","000","000","011","000","000","000","000"
,"000","000","000","000","000","000"),
	("000","000","000","000","000","000","000","000","011","011","011","000","000","000"
,"000","000","000","000","000","000"),
	("000","000","000","000","000","000","000","011","011","011","011","011","000","000"
,"000","000","000","000","000","000"),
	("000","000","000","000","000","000","011","011","011","011","011","011","011","000"
,"000","000","000","000","000","000"),
	("000","000","000","000","000","011","011","011","011","011","011","011","011","011"
,"000","000","000","000","000","000"),
	("000","000","000","000","000","011","011","011","011","011","011","011","011","011"
,"000","000","000","000","000","000"),
	("000","000","010","000","000","011","000","000","001","011","001","000","000","011"
,"000","000","010","000","000","000"),
	("000","010","010","010","000","000","000","001","001","011","001","001","000","000"
,"000","010","010","010","000","000"),
	("000","010","001","010","000","000","000","001","001","011","001","001","000","000"
,"000","010","001","010","000","000"),
	("000","010","001","010","000","000","001","001","001","011","001","001","001","000"
,"000","010","001","010","000","000"),
	("000","010","001","001","001","001","001","001","001","011","001","001","001","001"
,"001","001","001","010","000","000"),
	("010","010","001","001","001","001","001","001","001","011","001","001","001","001"
,"001","001","001","010","010","000"),
	("010","010","001","001","001","001","000","001","001","011","001","001","000","001"
,"001","001","001","010","010","000"),
	("010","010","001","001","010","000","000","001","001","000","001","001","000","000"
,"010","001","001","010","010","000"),
	("000","010","001","010","010","000","000","001","000","000","000","001","000","000"
,"010","010","001","010","000","000"),
	("000","010","001","010","000","000","000","000","000","000","000","000","000","000"
,"000","010","001","010","000","000"),
	("000","010","010","010","000","000","000","000","000","000","000","000","000","000"
,"000","010","010","010","000","000"),
	("000","000","010","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","010","000","000","000")
	);

	signal small_explode :matrix:= (
	("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000"),
	("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000"),
	("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000"),
	("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000"),
```

```
        ("000","000","000","000","000","000","000","011","000","000","000","000","000","000"
,"000","000"),
        ("000","000","000","000","000","000","011","011","100","000","011","000","000","000"
,"000","000"),
        ("000","000","000","000","000","000","100","100","011","000","100","011","000","000"
,"000","000"),
        ("000","000","000","000","100","011","000","100","100","100","011","100","000","000"
,"000","000"),
        ("000","000","000","000","011","011","011","011","100","011","011","100","011","000"
,"000","000"),
        ("000","000","000","000","000","011","100","100","011","011","100","000","000","000"
,"000","000"),
        ("000","000","000","000","100","000","100","011","100","000","011","000","000","000"
,"000","000"),
        ("000","000","000","000","000","000","000","011","000","000","011","000","000","000"
,"000","000"),
        ("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000"),
        ("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000"),
        ("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000"),
        ("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000")
        );

        signal big_bee :matrix:= (
        ("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000"),
        ("000","000","000","000","000","000","011","011","011","011","000","000","000","000"
,"000","000"),
        ("000","001","001","000","000","011","011","011","011","011","011","000","000","001"
,"001","000"),
        ("000","001","001","000","011","011","010","011","011","010","011","011","000","001"
,"001","000"),
        ("000","001","001","011","011","011","011","011","011","011","011","011","011","001"
,"001","000"),
        ("000","001","001","100","011","011","011","011","011","011","011","011","100","001"
,"001","000"),
        ("000","001","001","100","100","011","011","011","011","011","011","100","100","001"
,"001","000"),
        ("000","000","001","001","100","100","100","100","100","100","100","100","001","001"
,"000","000"),
        ("000","000","000","001","001","100","100","100","100","100","100","001","001","000"
,"000","000"),
        ("000","000","000","000","001","001","000","100","100","000","001","001","000","000"
,"000","000"),
        ("000","000","000","000","000","001","000","100","100","000","001","000","000","000"
,"000","000"),
        ("000","000","000","000","000","000","000","100","100","000","000","000","000","000"
,"000","000"),
        ("000","000","000","000","000","000","000","100","100","000","000","000","000","000"
,"000","000"),
        ("000","000","000","000","000","000","000","100","100","000","000","000","000","000"
,"000","000"),
```

```
        ("000","000","000","000","000","000","000","100","100","000","000","000","000","000"
,"000","000"),
        ("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000"));

        signal explode :matrix:= (
        ("000","000","000","000","000","000","000","000","000","101","000","000","000","000"
,"000","000"),
        ("000","000","000","000","000","101","100","100","100","101","101","101","101","000"
,"000","000"),
        ("100","101","000","000","000","101","101","101","101","100","101","101","101","000"
,"000","000"),
        ("100","100","100","100","101","101","101","100","101","100","101","100","101","101"
,"101","000"),
        ("101","100","101","100","101","101","100","100","100","101","101","100","100","100"
,"101","000"),
        ("000","101","101","100","101","101","100","100","100","101","100","100","100","101"
,"101","000"),
        ("000","000","000","100","100","101","101","101","101","100","100","101","100","101"
,"101","000"),
        ("000","000","000","000","100","100","101","101","101","101","100","100","101","101"
,"101","000"),
        ("000","000","101","101","100","101","101","100","101","100","100","101","101","000"
,"000","000"),
        ("000","000","101","101","101","101","101","100","101","100","100","101","101","000"
,"000","000"),
        ("101","101","101","100","100","100","100","101","101","101","100","100","101","000"
,"000","000"),
        ("101","101","100","100","100","101","100","100","101","100","100","100","100","101"
,"101","000"),
        ("101","100","101","101","101","101","101","101","000","101","101","101","100","100"
,"101","000"),
        ("101","100","101","101","000","000","100","101","000","000","000","101","101","100"
,"100","000"),
        ("101","101","000","000","000","000","100","100","000","000","000","101","101","101"
,"101","000"),
        ("000","000","000","000","000","000","101","101","000","000","000","000","000","000"
,"000","000"));

        signal bee:normalMatrix:= (
        ("00","00","00","00","00","00","00","00","00","00","00","00","00","00","00","00"),
        ("00","00","00","00","00","00","01","00","00","01","00","00","00","00","00","00"),
        ("00","01","00","00","00","00","01","01","01","01","00","00","00","00","01","00"),
        ("00","01","00","00","00","01","11","01","01","11","01","00","00","00","01","00"),
        ("00","01","01","01","01","01","01","01","01","01","01","01","01","01","01","00"),
        ("00","01","01","01","01","01","01","01","01","01","01","01","01","01","01","00"),
        ("00","00","00","00","01","01","01","01","01","01","01","01","00","00","00","00"),
        ("00","00","00","00","01","01","01","01","01","01","01","01","00","00","00","00"),
        ("00","00","00","00","00","01","01","01","01","01","01","00","00","00","00","00"),
        ("00","10","10","00","00","01","01","01","01","01","01","00","00","10","10","00"),
        ("00","10","10","10","10","10","01","01","01","01","10","10","10","10","10","00"),
        ("00","10","10","10","00","00","00","01","01","00","00","00","10","10","10","00"),
        ("00","10","10","00","00","00","00","01","01","00","00","00","00","10","10","00"),
        ("00","10","00","00","00","00","00","01","01","00","00","00","00","00","10","00"),
        ("00","00","00","00","00","00","00","01","01","00","00","00","00","00","00","00"),
        ("00","00","00","00","00","00","00","00","00","00","00","00","00","00","00","00")
```

```
);

signal beef:normalMatrix:= (
("00","00","00","00","00","00","00","00","00","00","00","00","00","00","00","00"),
("00","00","00","00","00","00","01","00","00","01","00","00","00","00","00","00"),
("00","01","00","00","00","00","01","01","01","01","00","00","00","00","01","00"),
("00","01","00","00","00","01","11","01","01","11","01","00","00","00","01","00"),
("00","01","01","01","01","01","01","01","01","01","01","01","01","01","01","00"),
("00","01","01","01","01","01","01","01","01","01","01","01","01","01","01","00"),
("00","00","00","00","01","01","01","01","01","01","01","01","00","00","00","00"),
("00","10","00","00","01","01","01","01","01","01","01","01","00","00","10","00"),
("00","10","10","00","00","01","01","01","01","01","01","00","00","10","10","00"),
("00","10","10","10","00","01","01","01","01","01","01","00","10","10","10","00"),
("00","10","10","10","10","10","01","01","01","01","10","10","10","10","10","00"),
("00","00","00","00","00","00","00","01","01","00","00","00","00","00","00","00"),
("00","00","00","00","00","00","00","01","01","00","00","00","00","00","00","00"),
("00","00","00","00","00","00","00","01","01","00","00","00","00","00","00","00"),
("00","00","00","00","00","00","00","01","01","00","00","00","00","00","00","00"),
("00","00","00","00","00","00","00","00","00","00","00","00","00","00","00","00")
);

signal bee30:normalMatrix:= (
("00","00","00","00","00","00","00","00","00","00","00","00","00","00","00","00"),
("00","00","00","00","00","00","00","00","00","00","00","00","00","00","00","00"),
("00","00","10","10","00","00","00","00","00","00","00","00","00","00","00","00"),
("00","00","10","10","00","00","00","00","00","00","00","00","00","00","00","00"),
("00","00","10","10","00","00","00","00","01","01","00","00","00","00","10","00"),
("00","00","10","10","10","10","01","01","01","01","01","00","00","10","10","00"),
("00","00","00","10","10","10","01","01","01","01","01","00","00","10","10","00"),
("00","00","00","00","00","01","01","01","01","01","01","10","10","10","10","00"),
("00","01","01","01","01","01","01","01","01","01","01","10","10","10","00","00"),
("00","01","01","00","01","01","01","01","01","01","00","00","00","00","00","00"),
("00","01","01","00","01","11","01","01","01","01","00","00","00","00","00","00"),
("00","00","00","00","01","01","01","11","01","01","01","01","00","00","00","00"),
("00","00","00","00","01","01","01","01","01","00","00","01","01","00","00","00"),
("00","00","00","01","00","00","00","01","00","00","00","01","01","00","00","00"),
("00","00","00","00","00","00","01","00","00","00","00","01","00","00","00","00"),
("00","00","00","00","00","00","00","00","00","00","00","00","00","00","00","00")
);

signal bee45:normalMatrix:= (
("00","00","00","00","00","00","00","00","00","00","00","00","00","00","00","00"),
("00","00","00","00","00","00","10","00","00","00","00","00","00","00","00","00"),
("00","00","00","00","00","10","10","00","00","00","00","00","00","00","00","00"),
("00","00","00","00","10","10","10","00","00","00","00","00","00","00","00","00"),
("00","00","00","00","10","10","10","00","00","01","01","01","00","00","00","00"),
("00","01","01","00","00","10","10","01","01","01","01","01","00","00","00","00"),
("00","01","01","01","00","01","01","10","01","01","01","01","00","00","00","00"),
("00","01","00","01","01","01","01","01","01","01","01","00","00","00","00","00"),
("00","00","00","01","01","01","01","01","01","10","01","00","00","00","00","00"),
("00","00","00","01","11","01","01","01","01","01","10","10","10","10","10","00"),
("00","00","01","01","01","01","01","01","01","01","10","10","10","10","00","00"),
("00","01","00","01","01","01","11","01","01","00","00","10","10","00","00","00"),
("00","00","00","00","01","01","01","01","01","01","00","00","00","00","00","00"),
("00","00","00","00","00","01","00","00","00","01","01","00","00","00","00","00"),
("00","00","00","00","01","00","00","00","01","01","01","00","00","00","00","00"),
```

```
        ("00","00","00","00","00","00","00","00","00","00","00","00","00","00","00","00")
    );

        signal small_plane:matrix:= (
        ("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000"),
        ("000","000","000","000","000","000","000","011","000","000","000","000","000","000"
,"000","000"),
        ("000","000","000","000","000","000","011","011","011","000","000","000","000","000"
,"000","000"),
        ("000","000","000","000","000","011","011","011","011","011","000","000","000","000"
,"000","000"),
        ("000","000","000","000","011","011","011","011","011","011","011","000","000","000"
,"000","000"),
        ("000","000","000","000","011","000","000","011","000","000","011","000","000","000"
,"000","000"),
        ("000","000","000","000","000","000","001","011","001","000","000","000","000","000"
,"000","000"),
        ("000","000","010","000","000","000","001","011","001","000","000","000","010","000"
,"000","000"),
        ("000","010","010","010","000","001","001","011","001","001","000","010","010","010"
,"000","000"),
        ("000","010","001","010","001","001","001","011","001","001","001","010","001","010"
,"000","000"),
        ("000","010","001","001","001","001","001","011","001","001","001","001","001","010"
,"000","000"),
        ("000","010","001","001","001","000","001","011","001","000","001","001","001","010"
,"000","000"),
        ("000","010","001","010","000","000","001","011","001","000","000","010","001","010"
,"000","000"),
        ("000","010","010","010","000","000","001","000","001","000","000","010","010","010"
,"000","000"),
        ("000","000","010","000","000","000","000","000","000","000","000","000","010","000"
,"000","000"),
        ("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000"));

        signal bigbee30:matrix:= (
        ("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000"),
        ("000","000","000","000","000","000","000","000","000","000","100","100","000","000"
,"000","000"),
        ("000","000","000","000","000","000","000","000","000","100","100","000","000","000"
,"000","000"),
        ("000","000","000","000","001","001","000","000","000","100","100","000","000","000"
,"000","000"),
        ("000","000","000","001","001","000","000","000","100","100","000","000","000","000"
,"000","000"),
        ("000","000","001","001","000","000","000","000","100","100","000","001","001","000"
,"000","000"),
        ("000","001","001","000","000","000","000","100","100","000","000","000","001","001"
,"000","000"),
        ("000","001","100","100","100","100","100","100","100","100","000","000","000","001"
,"001","000"),
        ("000","001","100","100","100","100","100","100","100","100","100","100","100","100"
,"001","000"),
```

```
        ("000","001","001","100","100","100","100","100","100","100","100","100","100","001"
,"001","000"),
        ("000","001","000","011","011","100","100","100","100","100","100","100","000","001"
,"001","000"),
        ("000","000","000","011","011","011","011","011","011","011","011","000","000","001"
,"001","000"),
        ("000","000","000","000","011","010","011","011","011","011","000","000","000","000"
,"000","000"),
        ("000","000","000","000","011","011","011","010","011","000","000","000","000","000"
,"000","000"),
        ("000","000","000","000","000","011","011","011","000","000","000","000","000","000"
,"000","000"),
        ("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000"));

        signal bigbee45:matrix:= (
        ("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000"),
        ("000","000","000","000","001","001","001","001","001","000","000","000","000","100"
,"100","000"),
        ("000","001","001","001","001","001","001","001","001","000","000","000","100","100"
,"100","000"),
        ("000","001","001","100","100","100","000","000","000","000","000","100","100","100"
,"000","000"),
        ("000","001","001","001","000","100","100","100","000","000","100","100","100","000"
,"000","000"),
        ("000","000","000","011","100","100","100","100","100","100","100","100","000","000"
,"000","000"),
        ("000","000","000","011","100","100","100","100","100","100","100","000","000","000"
,"000","000"),
        ("000","000","000","011","011","100","100","100","100","100","100","000","000","001"
,"001","000"),
        ("000","000","000","011","011","011","100","100","100","100","100","100","000","001"
,"001","000"),
        ("000","000","000","011","010","011","011","100","100","100","100","100","000","001"
,"001","000"),
        ("000","000","000","011","011","011","011","011","100","100","100","100","100","001"
,"001","000"),
        ("000","000","000","000","011","011","010","011","011","100","100","000","100","001"
,"001","000"),
        ("000","000","000","000","000","011","011","011","011","011","011","001","100","001"
,"000","000"),
        ("000","000","000","000","000","000","000","000","000","000","000","001","001","001"
,"000","000"),
        ("000","000","000","000","000","000","000","000","000","000","000","001","001","001"
,"000","000"),
        ("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000"));

        signal red_flag:matrix24:= (
        ("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000"),
        ("000","010","010","011","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000"),
        ("000","010","010","011","011","011","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000"),
```

```
	("000","010","010","011","011","011","011","011","000","000","000","000","000","000"
,"000","000","000","000","000","000"),
	("000","010","010","011","011","011","011","011","011","011","000","000","000","000"
,"000","000","000","000","000","000"),
	("000","010","010","011","011","011","011","011","011","011","011","011","000","000"
,"000","000","000","000","000","000"),
	("000","010","010","011","011","011","011","011","011","011","011","011","011","011"
,"000","000","000","000","000","000"),
	("000","010","010","011","011","011","011","011","011","011","011","011","011","011"
,"011","000","000","000","000","000"),
	("000","010","010","011","011","011","011","011","011","011","011","011","000","000"
,"000","000","000","000","000","000"),
	("000","010","010","011","011","011","011","011","011","011","000","000","000","000"
,"000","000","000","000","000","000"),
	("000","010","010","011","011","011","011","011","000","000","000","000","000","000"
,"000","000","000","000","000","000"),
	("000","010","010","011","011","011","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000"),
	("000","010","010","011","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000"),
	("000","010","010","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000"),
	("000","010","010","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000"),
	("000","010","010","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000"),
	("000","010","010","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000"),
	("000","010","010","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000"),
	("000","010","010","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000"),
	("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000"));

	constant INFO_H: integer := 550;
	constant INFO_V: integer := 125;
	constant info_size: integer := 50;
	signal infoH, infoV, infoG : std_logic;
	signal infoColor :unsigned(2 downto 0);

	signal big_explode: matrix24:= (
	("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000"),
	("000","000","000","101","101","000","000","000","000","101","101","101","101","000"
,"000","000","000","000","000","000"),
	("000","101","000","101","101","101","000","000","000","101","101","101","101","000"
,"000","101","101","101","000","000"),
	("000","101","101","100","100","101","101","000","000","101","100","100","101","000"
,"101","100","100","100","100","000"),
	("000","101","100","100","100","100","101","000","000","101","101","100","101","000"
,"101","101","100","101","101","000"),
	("000","101","101","100","101","100","100","101","101","101","101","100","101","101"
,"100","101","101","000","000","000"),
	("000","000","000","100","100","101","100","100","101","101","100","100","100","100"
,"100","101","101","000","000","000"),
```

```
("000","000","000","000","000","101","101","101","100","101","100","101","100","100","101","000","000","000","000","000"),
("000","000","000","000","000","100","101","100","100","101","100","101","101","100","101","101","101","000","000","000"),
("000","000","000","101","100","101","101","100","101","101","100","101","100","100","101","101","101","000","000","000"),
("000","000","101","101","100","101","100","100","100","100","101","101","101","100","100","101","101","000","000","000"),
("000","100","100","101","101","100","100","100","101","101","100","101","101","100","100","101","000","000","000","000"),
("000","100","100","101","101","100","100","100","101","101","100","100","100","100","101","101","101","101","000","000"),
("000","101","100","101","101","101","100","100","100","100","101","100","101","100","101","100","100","100","000","000"),
("000","101","000","000","000","101","101","101","100","101","100","101","101","101","101","101","100","100","000","000"),
("000","000","000","101","100","101","101","100","101","101","101","100","101","101","101","101","100","101","000","000"),
("000","000","101","101","100","100","100","100","101","101","101","101","101","100","100","101","000","000","000","000"),
("000","000","101","100","101","100","101","101","101","101","101","000","101","101","101","101","101","101","101","101"),
("000","101","101","101","101","101","101","000","101","000","000","000","000","000","101","101","100","100","101","000"),
("000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000"));

type cursorType is array(0 to 15) of unsigned (0 to 15);
signal cursor: cursorType:= (
("0000000000000000"),("0000000000000000"),("0000000000000000"),("0011000000000000"),("0011110000000000"),
("0011111100000000"),("0011111111000000"),("0011111111110000"),("0011111111111100"),("0011111111110000"),
("0011111111000000"),("0011111100000000"),("0011110000000000"),("0011000000000000"),("0000000000000000"),
("0000000000000000"));

type picMatrix is array(integer range 0 to 99, integer range 0 to 49) of unsigned(2 downto 0);
signal galaxian1: picMatrix := (
("111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","111"),
```

```
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
```

```
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","100","100","100","100"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","100","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","100","100","100","100","100","10
0","100","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","100","000","000","000","000","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","100","100","100","000","000","000","000","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","100","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","100","100","100","100","000","000","000","000","000","000","000","000","00
0","000","000","000","100","100"),
```

("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","100","000","000","000","000","000","000","000","000","000","100","100","10
0","100","100","100","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","100","000","000","000","000","000","000","000","000","000","100","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
100","100","100","000","000","000","000","000","000","000","000","000","100","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","100","
000","000","000","000","000","000","000","000","000","000","100","100","100","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","100","100","100","100","
000","000","000","000","000","000","000","000","000","000","100","100","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","
000","000","000","000","000","000","000","000","000","100","100","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","
000","000","000","000","000","000","000","000","000","100","100","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","
000","000","000","000","000","000","000","000","000","100","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","100","100","100","100","000","
000","000","000","000","000","000","000","000","000","100","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","
000","000","000","000","000","000","000","000","000","100","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","
000","000","000","000","000","000","000","000","000","100","111","111","111","111","111","11
1","111","111","111","111","100"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","
000","000","000","000","000","000","000","000","000","100","111","111","111","111","111","11
1","111","111","111","111","100"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","
000","000","000","000","000","000","000","000","000","100","100","111","111","111","111","11
1","111","111","111","100","100"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","100","100","000","000","000","000","000","000","000","000","000","000","
000","000","000","100","100","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","100","000","000","000","000","000","000","000","000","000","000","000","000","
000","100","100","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),

("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","100","000","000","000","000","000","000","000","000","000","000","000","
000","100","100","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","100","000","000","000","000","000","000","000","000","000","000","000","
000","100","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","100","100","000","000","000","000","000","000","000","000","000","000","000","
000","100","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","100"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"100","000","000","000","000","000","000","000","000","000","000","000","000","000","
100","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","100","000"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"100","000","000","000","000","000","000","000","000","000","000","000","000","000","
100","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","100","100","100","100","000"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","100"
,"100","000","000","000","000","000","000","000","000","000","000","000","000","000","
100","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","100","100","100","000"),
("111","111","111","111","111","111","111","111","111","111","111","111","100","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","
100","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","100","000"),
("111","111","111","111","111","111","111","111","111","111","111","111","100","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","100","100","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","100"),
("111","111","111","111","111","111","111","111","111","111","111","111","100","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","100","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","100","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","100","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","100","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","100","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","100","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","100","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","100"),
("111","111","111","111","111","111","111","111","111","111","111","100","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","100","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","100","000"),
("111","111","111","111","111","111","111","111","111","111","111","100","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","100","100","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","100","100","100","000"),

```
	("111","111","111","111","111","111","111","111","111","111","111","100","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
100","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","100","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","111","100","100","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
100","100","100","111","111","111","111","111","111","111","111","111","111","111","100","10
0","100","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","100","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","100","111","111","111","111","111","111","111","111","111","100","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","100","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","100","100","100","100","100","100","100","100","100","100","100","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","100","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","100","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","100","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","100","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","100","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","100","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","100","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","100","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","100","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","100","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
```

("111","111","111","111","111","111","111","111","111","100","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("111","111","111","111","111","111","111","111","111","111","100","100","100","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("111","111","111","111","111","111","111","111","111","111","111","111","100","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","100"
,"100","100","100","100","000","000","000","000","000","000","000","100","100","100","
100","100","100","100","100","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","100"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","100","000","000","000","000","000","000","000","100","111","111","
111","111","111","111","111","100","000","000","000","000","000","000","000","000","00
0","000","000","000","000","100"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","100","100","100","100","100","100","100","100","111","111","
111","111","111","111","111","100","000","000","000","000","000","000","000","000","10
0","100","100","100","100","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","100","100","000","000","000","000","000","000","000","000","000","10
0","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","100","000","000","000","100","100","100","100","100","100","100","100","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","100","100","000","000","000","100","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
100","100","100","100","100","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","100","
100","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","100","100","
100","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),

```
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"));
	signal galaxian2: picMatrix := (
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
```

```
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
```

111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","100"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","100","000"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","100","100","100","10
0","100","100","100","100","000"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","00
0","000","000","000","000","000"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","100","100","100","100","100","100","100","100","000","000","000","000","00
0","000","000","000","000","000"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","100","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","100","100","100","100","
100","100","100","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","100","10
0","100","100","100","100","100"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","100","100","100","100","100","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","100","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","100","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","100","100","100","100","100","100","100","111","11
1","111","111","111","111","111"),
("100","100","100","100","100","100","100","111","111","111","111","111","111","111"
,"111","100","100","100","100","100","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","100","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("000","000","000","000","000","000","000","100","111","111","111","111","111","111"
,"100","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","100","100","100","100","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("000","000","000","000","000","000","000","100","100","111","111","100","100","100"
,"100","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","100","111","111","111","111","111","111","111","111","111","111","111","111","11
1","100","100","100","100","100"),
("000","000","000","000","000","000","000","000","000","100","100","000","000","000"
,"000","000","000","000","000","000","000","000","000","100","100","100","100","
100","100","111","111","111","111","111","111","111","111","111","111","111","111","111","10
0","000","000","000","000","000"),
("000","000","000","000","000","000","000","000","000","100","100","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","100","111","111","111","

111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","10
0","000","000","000","000","000"),
("000","000","000","000","000","000","000","000","000","100","100","000","000","000"
,"000","000","000","000","000","000","000","100","100","100","100","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","100","100","100","10
0","000","000","000","000","000"),
("100","100","100","100","100","100","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","100","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("111","111","111","111","111","100","000","000","000","000","000","000","000","000"
,"000","000","100","100","100","100","100","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","100","000","000","000","00
0","000","000","000","000","000"),
("111","111","111","111","100","000","000","000","000","000","000","000","000","000"
,"000","000","100","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","100","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("111","111","111","111","100","000","000","000","000","000","000","000","000","000"
,"000","000","100","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","100","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("111","111","111","111","100","000","000","000","000","000","000","000","000","100"
,"100","100","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","100","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("111","111","111","100","100","000","000","000","000","000","000","000","000","100"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","100","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("111","111","100","000","000","000","000","000","100","100","100","100","100","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","100","100","100","000","000","000","000","00
0","000","000","000","000","000"),
("111","100","100","000","000","000","000","000","100","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("100","000","000","000","000","100","100","100","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("100","000","000","000","000","100","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("100","100","100","100","100","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("100","100","100","100","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("100","111","111","111","111","111","111","111","111","111","111","111","111","111","100"
,"100","100","100","100","111","111","111","111","111","111","111","111","111","111","111","

111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("111","111","111","111","111","111","111","111","111","111","111","100","100","100"
,"100","100","100","100","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("111","111","111","111","111","111","111","111","111","111","111","100","000","000","000"
,"000","100","100","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("111","111","111","111","111","100","100","100","100","100","100","100","000","000","000"
,"000","100","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("111","111","111","111","100","000","000","000","000","000","000","000","000","000","000"
,"000","100","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("100","100","100","100","000","000","000","000","000","000","000","000","000","100","100"
,"100","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","000","000","000","000","000","100","100","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","000","000","000","000","000","100","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","000","100","100","100","100","111","111","111","111"
,"111","111","100","100","100","100","100","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","000","100","111","111","111","111","111","111","111"
,"111","100","000","000","000","000","000","100","100","100","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("100","000","000","000","000","000","000","100","100","100","100","100","100","100","100"
,"100","100","000","000","000","000","000","000","000","000","100","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("100","000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","100","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("100","000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","100","100","100","111","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("100","000","000","000","000","000","000","000","000","000","000","000","000","100","100"
,"100","100","100","100","100","000","000","000","000","000","000","000","000","000","100","
111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("100","000","000","000","000","000","000","100","100","111","111","111","111","111","111"
,"111","111","111","100","000","000","000","000","000","000","000","000","000","000","100","

100","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","100","100","111","111","111","111","111","111"
,"111","111","111","100","000","000","000","000","000","000","000","000","000","000","
000","100","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","100","111","111","111","111","111","111","111"
,"111","111","100","100","000","000","000","000","000","000","000","000","000","000","
000","100","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","100","111","111","111","111","111","111","111"
,"111","100","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","100","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","100","111","111","111","111","111","111","100"
,"100","100","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","100","100","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","100","111","111","111","111","111","100","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","100","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","100","111","111","111","111","111","100","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","100","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","100","111","111","111","111","111","100","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","100","100","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","100","111","111","111","100","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","100","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","100","111","111","100","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","100","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","100","111","111","100","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","100","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","000","100","111","100","000","000","000","000"
,"000","000","000","000","000","000","100","111","100","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","100","111","100","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","000","100","100","100","000","000","000","000"
,"000","000","000","000","000","000","100","111","100","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","100","100","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","100","111","100","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","100","100","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","100","111","111","111","100","000","000","000","000","000","000","

```
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("000","000","000","000","100","100","000","000","000","000","000","000","000","000"
,"000","000","000","000","100","111","111","111","100","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("000","000","000","000","100","100","000","000","000","000","000","000","000","000"
,"000","000","000","000","100","111","111","111","100","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("000","000","000","000","100","100","000","000","000","000","000","000","000","000"
,"000","000","000","000","100","111","111","111","100","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","100","111","111","111","100","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","100","111","100","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("100","100","100","100","100","100","100","100","000","000","000","000","000","000"
,"000","000","000","000","000","100","100","100","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","100","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","111","100","100","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","100","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","111","100","100","100","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","100","
100","100","100","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","111","111","111","100","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","100","
111","111","100","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","100"
,"100","100","100","100","000","000","000","000","100","100","100","100","100","100","111","
111","111","111","100","100","100","100","100","100","100","100","100","100","100","100","10
0","100","100","100","100","100"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","100","000","000","000","000","100","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","100","100","100","100","111","111","111","111","111","111","111","
```

111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","

```
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111")
	);
	signal galaxian3: picMatrix := (
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
```

```
("111","111","111","111","111","111","111","111","111","100","100","100","100","100"
,"100","100","100","100","100","100","100","100","100","100","100","100","100","100","100","
100","100","100","100","100","100","100","100","100","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","100","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","100","111","111","111","111","111","11
1","111","111","111","111","111"),
("100","100","100","100","100","100","100","100","100","100","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","100","100","100","100","100","100","10
0","100","100","100","100","100"),
("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","100","100","100","100","100","100","100","100","100","100","100","100","100","
100","100","100","100","100","100","100","100","100","100","100","100","100","100","100","10
0","100","100","100","100","100"),
("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","100","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("000","100","100","100","100","100","100","100","100","100","100","100","100","100"
,"100","100","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("000","100","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
```

```
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
```

```
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("100","100","100","100","100","100","100","100","100","100","100","100","100","100"
,"100","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","100","100","100","100","100","100","100","100","111","111","111","11
1","111","111","111","111","111"),
	("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","100","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","100","000","000","000","000","000","000","000","000","100","111","111","11
1","111","111","111","111","111"),
	("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","100","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","100","000","000","000","000","000","000","000","000","000","100","111","111","11
1","111","111","111","111","111"),
```

("100","100","100","100","100","100","100","100","100","000","000","000","000","000"
,"000","000","100","100","100","100","100","111","111","111","111","111","100","100","100","
100","100","100","000","000","000","000","000","000","000","000","000","100","100","100","10
0","111","111","111","111","111"),
("100","111","111","111","111","111","111","111","100","000","000","000","000","000"
,"000","000","000","000","000","000","000","100","111","111","111","100","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","100","111","111","111","111"),
("100","111","111","111","111","111","111","111","100","000","000","000","000","000"
,"000","000","000","000","000","000","000","100","100","100","100","100","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","100","111","111","111","111"),
("100","111","111","111","111","111","111","111","111","100","100","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","100","100","100","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","100","111","111","100"),
("100","111","111","111","111","111","111","111","111","100","100","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","100","111","100","
000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","100","100","100","000"),
("100","111","111","111","111","111","111","111","100","000","000","000","000","000"
,"000","000","000","000","000","000","000","100","100","111","111","111","
100","100","100","100","100","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("100","111","111","111","111","111","111","100","100","000","000","000","000","000"
,"000","000","000","000","000","000","000","100","100","111","111","111","
111","111","111","111","100","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("100","111","111","111","111","111","100","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","100","111","111","
111","111","111","111","111","100","100","100","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("100","111","111","111","111","111","100","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","100","111","111","
111","111","111","111","111","111","111","100","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("100","111","111","111","111","100","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","100","111","111","
111","111","111","111","111","111","111","111","100","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("100","111","111","111","100","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","100","100","
111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","00
0","000","000","000","000","000"),
("100","111","111","111","100","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
100","111","111","111","111","111","111","111","111","111","111","100","100","000","000","00
0","000","000","000","000","000"),
("100","111","111","111","100","000","000","000","000","000","000","000","000","000"
,"100","100","100","000","000","000","000","000","000","000","000","000","000","000","000","
100","111","111","111","111","111","111","111","111","111","111","111","100","000","000","00
0","000","000","000","000","000"),
("100","111","111","100","000","000","000","000","000","000","000","000","000","000"
,"100","111","100","000","000","000","000","000","000","000","000","000","000","000","000","
100","111","111","111","111","111","111","111","111","111","111","111","100","000","000","00
0","000","000","000","000","000"),

("100","111","111","100","000","000","000","000","000","000","000","000","000","000"
,"100","111","100","000","000","000","000","000","000","000","000","000","000","000","
100","100","111","111","111","111","111","111","111","100","100","100","000","000","000","00
0","000","000","000","000","000"),
("100","111","111","100","000","000","000","000","000","000","000","000","000","000"
,"100","111","100","000","000","000","000","000","000","000","000","000","000","000","
000","000","100","111","111","111","111","111","100","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("100","111","111","100","000","000","000","000","000","000","000","000","000","100"
,"111","111","111","100","000","000","000","000","000","000","000","000","000","000","
000","000","100","111","111","111","111","100","100","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("100","111","111","100","000","000","000","000","000","000","000","000","000","100"
,"111","111","111","100","000","000","000","000","000","000","000","000","000","000","
000","000","000","100","111","111","100","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("100","111","111","100","000","000","000","000","000","000","000","000","000","100"
,"111","111","111","100","000","000","000","000","000","000","000","000","000","000","
000","000","000","100","111","111","100","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("100","111","100","100","000","000","000","000","000","000","000","000","000","100"
,"111","111","111","100","000","000","000","000","000","000","000","000","000","000","
000","000","000","100","100","100","100","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("100","100","000","000","000","000","000","000","000","000","000","000","000","100"
,"111","111","111","100","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","100","10
0","100","000","000","000","000"),
("100","100","000","000","000","000","000","000","000","000","000","000","000","000"
,"100","100","100","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","000","100","11
1","100","000","000","000","000"),
("100","100","000","000","000","000","000","000","000","000","000","000","000","000"
,"100","100","100","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","100","100","111","11
1","111","100","100","000","000"),
("100","100","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","100","111","111","11
1","111","111","100","000","000"),
("100","100","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","100","100","100","111","111","111","11
1","111","111","111","100","000"),
("100","100","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","100","100","111","111","111","111","11
1","111","111","111","111","100"),
("100","111","100","100","100","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","100","100","100","000","000","000","
000","000","000","000","100","100","100","100","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("100","111","111","111","111","100","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","100","111","111","100","000","000","000","
000","000","000","000","100","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),

```
("111","111","111","111","111","111","100","100","100","100","100","100","100","100"
,"100","100","100","100","100","100","100","100","111","111","111","111","100","100","100","
100","100","100","100","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
```

```
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"));
	signal galaxian4: picMatrix := (
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
	("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
```

```
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("100","100","100","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("000","000","000","100","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("000","000","000","100","100","100","100","100","100","100","100","100","100","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("100","000","000","000","000","000","000","000","000","000","000","000","000","100"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("100","000","000","000","000","000","000","000","000","000","000","000","000","100"
,"100","100","100","100","100","100","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","100","100","100","100","100","100","100","100","100","100","100","000","000"
,"000","000","000","000","000","000","100","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","100","000","000"
,"000","000","000","000","000","000","100","100","100","100","100","100","100","100","100","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","100","100"
,"100","100","100","100","100","000","000","000","000","000","000","000","000","000","000","
100","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","100","000","000","000","000","000","000","000","000","000","000","
100","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","100","100","100","100","100","100","100","100","000","000","
```

100","100","100","100","100","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","100","000","000","
000","000","000","000","000","100","100","100","100","100","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","100","000","
000","000","000","000","000","000","000","000","000","100","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","100","
100","100","100","000","000","000","000","000","000","000","100","100","100","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","100","100","100","100","100","000","000","000","000","000","100","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","000","100","100","10
0","100","100","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","100","100","100","100","000","000","000","00
0","000","000","100","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","100","100","000","000","00
0","000","000","000","100","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","100","100","10
0","000","000","000","000","100"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","10
0","100","000","000","000","100"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","100","000","000","000","100"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","100","100","100","000"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","100","000"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","100"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","

111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","100","100","100","100","100","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
100","000","000","000","000","000","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","100","
000","000","000","000","000","000","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","100","
000","000","000","000","000","000","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","100","000","
000","000","000","000","000","000","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","100","000","000","
000","000","000","000","000","000","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","100","000","000","
000","000","000","000","000","000","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","100","000","000","
000","000","000","000","000","000","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","100","000","000","
000","000","000","000","000","100","111","111","111","111","111","111","111","111","111","11
1","111","111","100","100","100"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","100","000","
000","000","000","000","100","111","111","111","111","111","111","111","111","111","111","11
1","111","100","000","000","000"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","100","
000","000","000","100","111","111","111","111","111","111","111","111","111","111","111","11
1","111","100","000","000","000"),
("111","111","111","111","111","111","100","100","100","100","100","100","100","100"
,"100","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
100","100","100","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","100","000","100"),
("111","111","111","111","111","100","000","000","000","000","000","000","000","000"
,"000","100","111","111","111","111","111","111","111","111","111","111","111","111","111","

111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","100","100"),
("111","111","111","100","100","100","000","000","000","000","000","000","000","000"
,"000","100","111","111","111","111","111","111","111","111","111","100","100","100","100","
100","100","100","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","100","000","000","000","000","000","000","000","000","000","000","000"
,"000","100","111","111","111","111","111","111","111","111","100","000","000","000","000","
000","000","000","100","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","100","100","000","000","000","000","000","000","000","000","000","000","000"
,"000","100","111","111","111","111","111","111","111","111","100","000","000","000","000","
000","000","000","100","111","100","100","100","100","100","100","100","100","111","111","11
1","111","111","111","111","111"),
("100","000","000","000","000","000","000","000","000","000","000","000","000","100"
,"100","111","111","111","111","111","111","111","111","100","000","000","000","000","000","
000","000","000","100","100","000","000","000","000","000","000","000","000","100","111","11
1","111","111","111","111","111"),
("100","000","000","000","000","000","000","000","000","000","000","000","000","100"
,"111","111","111","111","111","111","111","111","100","100","000","000","000","000","000","
000","000","000","100","100","000","000","000","000","000","000","000","000","100","100","10
0","100","111","111","111","111"),
("000","000","000","000","000","000","000","000","000","000","000","000","000","100"
,"111","111","111","111","111","111","111","111","100","000","000","000","000","000","000","000","
000","100","100","100","111","100","100","100","000","000","000","000","000","000","000","00
0","000","100","111","111","111"),
("000","000","000","000","000","000","000","000","000","000","000","000","000","100"
,"111","111","111","111","111","111","111","100","100","000","000","000","000","000","000","000","
000","100","111","111","111","111","111","111","100","000","000","000","000","000","000","000","00
0","000","100","100","100","111"),
("000","000","000","000","000","000","000","000","000","000","100","100","100","111"
,"111","111","111","111","111","111","100","000","000","000","000","000","000","000","000","000","
000","100","111","111","111","111","111","111","100","000","000","000","000","000","000","000","00
0","000","000","000","000","100"),
("000","000","000","000","000","000","000","000","000","000","100","111","111","111"
,"111","111","111","111","111","111","100","000","000","000","000","000","000","000","000","000","
000","100","111","111","111","111","111","111","100","000","000","000","000","000","000","000","00
0","000","000","000","000","100"),
("000","000","000","000","000","000","000","000","100","100","111","111","111","111"
,"111","111","111","111","111","111","100","000","000","000","000","000","000","000","000","000","
000","100","111","111","111","111","100","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","000","000","000","100","111","111","111","111","111"
,"111","111","111","111","111","111","100","000","000","000","000","000","000","000","000","000","
000","100","111","100","100","100","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","100","100","100","111","111","111","111","111","111"
,"111","111","111","111","111","100","000","000","000","000","000","000","000","000","000","000","
100","100","100","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","000","000","100","111","111","111","111","111","111","111","111"
,"111","111","111","111","100","000","000","000","000","000","000","000","000","000","000","
100","100","100","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("000","000","000","100","100","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","100","000","000","000","000","000","000","000","000","000","000","000","

```
100","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
            ("000","000","000","100","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","100","000","000","000","000","000","000","000","000","000","000","000","
100","000","000","000","000","000","000","000","000","000","000","000","100","100","000","00
0","000","000","000","000","000"),
            ("000","000","000","000","100","111","111","111","111","111","111","111","111","111"
,"111","111","100","000","000","000","000","000","000","000","000","000","000","000","000","
100","000","000","000","000","000","000","000","000","000","000","000","100","100","000","00
0","000","000","000","000","000"),
            ("000","000","000","000","000","100","111","111","111","111","111","111","111","111"
,"111","111","100","000","000","000","000","000","000","000","000","000","000","000","000","
100","000","000","000","000","000","000","000","000","000","000","000","100","100","000","00
0","000","000","000","000","000"),
            ("000","000","000","000","000","000","100","111","111","111","111","111","111","111"
,"111","111","100","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","000","000","000","100","111","100","00
0","000","000","000","000","000"),
            ("000","000","000","000","000","000","100","100","100","100","111","111","111","111"
,"111","111","100","000","000","000","000","000","000","000","000","000","000","100","100","
000","000","000","000","000","000","000","000","000","000","000","100","111","111","111","10
0","000","000","000","000","000"),
            ("000","000","000","000","000","000","000","000","000","000","100","111","111","111"
,"111","111","100","000","000","000","000","000","000","000","000","000","000","100","100","
000","000","000","000","000","000","000","000","000","000","000","100","111","111","111","10
0","000","000","000","000","000"),
            ("000","000","000","000","000","000","000","000","000","000","100","111","111","111"
,"111","111","100","000","000","000","000","000","000","000","000","000","000","100","100","
000","000","000","000","000","000","000","000","000","000","000","100","111","111","111","10
0","000","000","000","000","000"),
            ("000","000","000","000","000","000","000","000","000","000","100","100","100","111"
,"111","100","100","000","000","000","000","000","000","000","000","000","000","100","100","
000","000","000","000","000","000","000","000","000","000","000","100","111","111","111","10
0","000","000","000","000","000"),
            ("000","000","000","000","000","000","000","000","000","000","000","000","000","100"
,"100","000","000","000","000","000","000","000","000","000","000","000","000","100","100","
000","000","000","000","000","000","000","000","000","000","000","100","111","111","100","10
0","000","000","000","000","000"),
            ("000","000","000","000","000","000","000","000","000","000","000","000","000","100"
,"100","000","000","000","000","000","000","000","000","000","000","000","000","100","100","
000","000","000","000","000","000","000","000","000","000","000","000","100","100","000","00
0","000","000","000","000","000"),
            ("000","000","000","000","000","000","000","000","000","000","000","000","000","100"
,"100","000","000","000","000","000","000","000","000","000","000","100","100","111","111","
100","000","000","000","000","000","000","000","000","000","000","000","100","100","000","00
0","000","000","000","000","000"),
            ("000","000","000","000","000","000","000","000","000","000","000","000","000","100"
,"100","000","000","000","000","000","000","000","000","000","000","100","111","111","111","
100","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
            ("000","000","000","000","000","000","000","000","000","000","000","000","000","100"
,"100","000","000","000","000","000","000","000","000","000","000","100","111","111","111","
100","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
            ("000","000","000","000","000","000","000","000","000","000","000","000","000","100"
,"100","000","000","000","000","000","000","000","000","000","000","100","111","111","111","
```

100","000","000","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("100","000","000","000","000","000","000","000","000","000","000","000","000","100"
,"100","000","000","000","000","000","000","000","000","100","100","111","111","111","111","
111","100","100","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("111","100","000","000","000","000","000","000","000","000","000","000","000","100"
,"100","000","000","000","000","000","000","000","000","100","111","111","111","111","111","
111","111","100","000","000","000","000","000","000","000","000","000","000","000","000","00
0","000","000","000","000","000"),
("111","111","100","100","100","100","100","100","100","100","100","100","100","000"
,"000","000","000","100","100","100","100","100","100","111","111","111","111","111","111","
111","111","111","100","100","000","000","000","000","000","000","000","000","000","000","00
0","000","100","100","100","100"),
("111","111","111","111","111","111","111","111","111","111","111","111","100","000"
,"000","000","000","100","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","100","000","000","000","000","000","000","000","000","000","000","00
0","000","100","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","100"
,"100","100","100","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","100","100","100","100","100","100","100","100","100","100","10
0","100","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","

```
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"));
        signal galaxian5: picMatrix := (
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
```

```
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
```

```
		("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","100","100","100","100","100","100","100","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","100","000","000","000","000","000","000","000","100","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("111","111","111","111","111","111","111","111","111","111","111","111","111","100"
,"100","100","100","100","100","000","000","000","000","000","000","000","100","100","100","
100","100","100","100","100","100","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("111","111","111","111","111","111","111","111","111","111","111","111","100","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("111","111","111","111","111","111","100","100","100","100","100","100","100","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("111","111","111","111","111","100","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("111","100","100","100","100","100","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","100","000","000","000","000","000","
000","000","000","000","000","000","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("100","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","100","000","000","000","000","000","
000","000","000","000","000","000","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("100","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","100","100","100","100","100","100","100","100","100","100","100","100","000","
000","000","000","000","000","000","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("100","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","100","111","111","111","111","111","111","111","111","111","100","100","000","
000","000","000","000","000","000","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("000","000","000","000","000","000","000","000","000","000","000","000","000","100"
,"100","100","111","111","111","111","111","111","111","111","111","100","000","000","000","
000","000","000","000","100","100","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
```

("000","000","000","000","000","000","000","000","000","000","000","000","000","100"
,"111","111","111","111","111","111","111","111","111","111","100","100","000","000","000","
000","000","000","000","100","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("100","100","000","000","000","000","000","000","000","100","100","100","100","111"
,"111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","000","
000","000","100","100","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","100","000","000","000","000","000","000","100","100","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","000","
000","000","100","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","100","100","100","000","000","000","000","000","100","111","111"
,"111","111","111","111","111","111","111","111","111","111","100","000","000","000","000","000","000","
000","100","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","100","000","000","000","000","000","100","100","100"
,"111","111","111","111","111","111","111","111","100","100","000","000","000","000","000","000","
000","100","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","100","100","100","000","000","000","000","000"
,"100","111","111","111","111","111","100","000","000","000","000","000","000","000","100","
100","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","100","000","000","000","000","000"
,"100","100","111","111","111","100","100","000","000","000","000","000","000","000","100","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","100","100","100","000","000"
,"000","000","100","111","100","000","000","000","000","000","000","000","100","100","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","100","000"
,"000","000","100","100","100","000","000","000","000","000","000","000","100","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111","100"
,"100","100","000","000","000","000","000","000","000","100","100","100","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","100","000","000","000","000","000","000","000","100","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("111","111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"100","000","000","000","000","000","100","100","100","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("100","100","100","100","100","111","111","111","111","111","111","111","111","111","100"
,"000","000","000","000","000","100","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("000","000","000","000","000","100","111","111","111","111","111","111","100","000"
,"000","000","000","000","100","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),

```
		("000","000","000","000","000","000","100","111","111","111","111","111","100","000"
,"000","000","000","000","100","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("000","000","000","000","000","000","100","100","100","100","100","100","000","000"
,"000","000","000","100","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("100","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"100","100","100","100","100","100","100","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("111","100","100","100","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","100","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("111","111","111","100","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","100","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("111","111","111","100","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","100","100","100","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("111","111","111","111","100","100","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","100","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("111","111","111","111","111","100","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","100","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("111","111","111","111","111","100","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","100","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("111","111","111","111","111","100","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","000","000","000","000","000","000","000","100","100","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("111","111","111","111","111","100","000","000","000","000","000","000","000","000"
,"100","100","100","100","000","000","000","000","000","000","000","000","000","000","100","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("100","111","111","111","111","100","000","000","000","000","000","000","000","000"
,"100","111","111","100","000","000","000","000","000","000","000","000","000","000","100","
100","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("000","100","111","111","111","100","000","000","000","000","000","000","000","000"
,"100","111","111","100","000","000","000","000","000","000","000","000","000","000","000","
000","100","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
		("000","100","111","111","111","100","000","000","000","000","000","000","000","000"
,"100","111","111","100","000","000","000","000","000","000","000","000","000","000","000","
000","100","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
```

```
("000","000","100","111","111","100","100","000","000","000","000","000","000","000"
,"100","111","111","111","100","000","000","000","000","000","000","000","000","000","000","
000","000","100","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("000","000","000","100","111","111","100","000","000","000","000","000","000","000"
,"000","100","111","111","100","100","000","000","000","000","000","000","000","000","000","
000","000","100","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("000","000","000","000","100","111","100","000","000","000","000","000","000","000"
,"000","000","100","111","111","100","000","000","000","000","000","000","000","000","000","
000","000","100","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("000","000","000","000","100","100","100","000","000","000","000","000","000","000"
,"000","000","100","111","111","100","000","000","000","000","000","000","000","000","000","
000","000","100","100","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("000","000","000","000","000","100","100","000","000","000","000","000","000","000"
,"000","000","100","111","111","111","100","000","000","000","000","000","000","000","000","
000","000","000","000","100","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("000","000","000","000","000","000","100","000","000","000","000","000","000","000"
,"000","000","100","111","111","111","100","000","000","000","000","000","000","000","000","
000","000","000","000","100","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("000","000","000","000","000","000","000","100","100","000","000","000","000","000"
,"000","000","000","100","111","111","100","000","000","000","000","000","000","000","000","
000","000","000","000","100","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("000","000","000","000","000","000","000","100","100","000","000","000","000","000"
,"000","000","000","000","100","111","100","000","000","000","000","000","000","000","000","
000","000","000","000","100","100","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("000","000","000","000","000","000","000","100","100","000","000","000","000","000"
,"000","000","000","000","100","111","100","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("000","000","000","000","000","000","000","100","100","000","000","000","000","000"
,"000","000","000","000","100","111","100","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("000","000","000","000","000","000","000","100","100","000","000","000","000","000"
,"000","000","000","000","100","111","100","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","100","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("000","000","000","000","000","000","000","100","100","000","000","000","000","000"
,"000","000","000","000","100","111","111","100","100","000","000","000","000","000","000","
000","000","000","000","000","000","000","100","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("000","000","000","000","000","000","000","100","100","000","000","000","000","000"
,"000","000","000","000","100","111","111","111","100","000","000","000","000","000","000","
000","000","000","000","000","000","000","100","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","100","111","111","100","000","000","000","000","000","000","
000","000","000","000","000","000","000","100","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
```

```
        ("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","100","111","111","100","000","000","000","000","000","000","
000","000","000","000","000","000","000","100","100","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","100","111","111","111","100","000","000","000","000","000","
000","000","000","000","000","000","000","000","000","100","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","100","111","111","111","111","100","000","000","000","000","
000","000","000","000","000","000","000","000","000","100","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","100","111","111","111","111","100","000","000","000","000","
000","000","000","000","000","000","000","000","000","100","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("000","000","000","000","000","000","000","000","000","000","000","000","000","000"
,"000","000","000","000","000","100","111","111","111","111","100","000","000","000","000","
000","000","000","000","000","000","000","000","000","100","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("100","100","100","000","000","000","100","100","100","100","100","000","000","000"
,"000","000","000","000","000","100","111","111","111","111","111","100","100","100","000","
000","000","000","000","000","000","000","000","000","100","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","100","000","000","000","100","111","111","111","100","000","000","000"
,"000","000","000","000","000","100","111","111","111","111","111","111","111","100","000","
000","000","000","000","000","000","000","000","000","100","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","100","100","100","111","111","111","111","111","100","100","100"
,"100","100","100","100","100","111","111","111","111","111","111","111","111","111","100","
100","100","100","100","100","100","100","100","100","100","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
```

```
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"),
        ("111","111","111","111","111","111","111","111","111","111","111","111","111","111"
,"111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","
111","111","111","111","111","111","111","111","111","111","111","111","111","111","111","11
1","111","111","111","111","111"));

begin

Delay : process (clk)
begin
        if rising_edge(clk) then
                clk25 <= not clk25;
        end if;
end process Delay;

DataProcess : process (clk)
variable flag :unsigned (11 downto 0);
variable beeNum: integer;
variable bulletNum : integer;
variable chipAndWrite : std_logic;
variable convertFlag: integer;
begin
        if rising_edge(clk) then
                if reset_n = '0' then
                        readdata <= (others => '0');
                else
                        chipAndWrite := chipselect and write;
                        if chipselect = '1' and read = '1' then
                                readdata <= dataSendBack;
```

```
elsif chipAndWrite = '1' and address = "01101" then
        startCount <= '1';
else
        startCount <= '0';
end if;

flag := writedata(31 downto 20);
if chipAndWrite = '1' and address = "01100" then            -- flying bee
        beeNum := to_integer(flag(2 downto 0));
        TCoorFlyBeeH(beeNum) <= writedata(19 downto 10);
        TCoorFlyBeeV(beeNum) <= writedata(9 downto 0);
        FlyBeeAngle(beeNum) <= flag(10 downto 7);
        FlyBeeType(beeNum) <= flag(4 downto 3);
elsif chipAndWrite = '1' and address = "01010" then                 --
bee bullet
        bulletNum := to_integer(flag(5 downto 0));

        TCoorBeeBulletH(bulletNum) <= writedata(19 downto 10);
        TCoorBeeBulletV(bulletNum) <= writedata(9 downto 0);
elsif chipAndWrite = '1' and address = "01000" then                 --
bee matrix
        CoorBeeMaxH <= writedata(19 downto 10);
        CoorBeeMaxV <= writedata(9 downto 0);
elsif chipAndWrite = '1' and address = "00111" then                 --
plane
        TCoorPlaneH <= writedata(19 downto 10);
        TCoorPlaneV <= writedata(9 downto 0);
elsif chipAndWrite = '1' and address = "00110" then                 --
plane bullet
        Rb1H <= writedata(19 downto 10);
        Rb1V <= writedata(9 downto 0);
elsif chipAndWrite = '1' and address = "00101" then                 --
alive matrix
        if (flag(0) = '1') then                 -- alive 1
                tmpM1 <= writedata(19 downto 0);
        elsif (flag(1) = '1') then    -- alive 2
                tmpM2 <= writedata(19 downto 0);
        elsif (flag(2) = '1') then    -- alive 3
                tmpM3 <= writedata(19 downto 0);
        elsif (flag(3) = '1') then    -- alive 4
                tmpM4 <= writedata(19 downto 0);
        elsif (flag(4) = '1') then    -- alive 5
                tmpM5 <= writedata(19 downto 0);
        end if;
elsif chipAndWrite = '1' and address = "00100" then                 --
information
        convertFlag := to_integer(flag);
        if convertFlag = 1 then                 -- main pic
                mainPic <= writedata(0);
                mainPicV <= writedata(19 downto 10);
        elsif convertFlag = 2 then-- other information
                planeLife <= writedata(2 downto 0);
                level <= writedata(5 downto 3);
        elsif convertFlag = 3 then
                TmpClearScr <= '1';
        elsif convertFlag = 4 then-- ready
```

```
                        readySignal     <= writedata(0);
                    elsif convertFlag = 5 then-- pause
                        pauseSignal <= writedata(0);
                    elsif convertFlag = 6 then-- g
                        tmpgH <= writedata(19 downto 10);
                        tmpgV <= writedata(9 downto 0);
                    elsif convertFlag = 7 then-- a
                        tmpaH <= writedata(19 downto 10);
                        tmpaV <= writedata(9 downto 0);
                    elsif convertFlag = 8 then-- m
                        tmpmH <= writedata(19 downto 10);
                        tmpmV <= writedata(9 downto 0);
                    elsif convertFlag = 9 then-- e
                        tmpe1H <= writedata(19 downto 10);
                        tmpe1V <= writedata(9 downto 0);
                    elsif convertFlag = 10 then         -- o
                        tmpoH <= writedata(19 downto 10);
                        tmpoV <= writedata(9 downto 0);
                    elsif convertFlag = 11 then         -- v
                        tmpvH <= writedata(19 downto 10);
                        tmpvV <= writedata(9 downto 0);
                    elsif convertFlag = 12 then         -- e
                        tmpe2H <= writedata(19 downto 10);
                        tmpe2V <= writedata(9 downto 0);
                    elsif convertFlag = 13 then         -- r
                        tmprH <= writedata(19 downto 10);
                        tmprV <= writedata(9 downto 0);
                    end if;
                elsif chipAndWrite = '1' and address = "00011" then                 --
plane explode
                    if (flag(0) = '1') then
                        planeSmall <= '1';
                    else
                        planeSmall <= '0';
                    end if;
                    TplaneExplodeH <= writedata(19 downto 10);
                    TplaneExplodeV <= writedata(9 downto 0);
                elsif chipAndWrite = '1' and address = "00010" then                 --
bee explode
                    if (flag(0) = '1') then     -- small explode
                        Small <= '1';
                    else
                        Small <= '0';
                    end if;
                    TCoorExplodeH <= writedata(19 downto 10);
                    TCoorExplodeV <= writedata(9 downto 0);
                elsif chipAndWrite = '1' and address = "00001" then                 --
high score
                    hiScoreData(0) <= writedata(3 downto 0);
                    hiScoreData(1) <= writedata(7 downto 4);
                    hiScoreData(2) <= writedata(11 downto 8);
                    hiScoreData(3) <= writedata(15 downto 12);
                    hiScoreData(4) <= writedata(19 downto 16);
                elsif chipAndWrite = '1' and address = "00000" then                 --
score
                    scoreData(0) <= writedata(3 downto 0);
```

```
                                scoreData(1) <= writedata(7 downto 4);
                                scoreData(2) <= writedata(11 downto 8);
                                scoreData(3) <= writedata(15 downto 12);
                                scoreData(4) <= writedata(19 downto 16);
                        end if;
                end if;
        end if;
end process DataProcess;

SyncProcess: process (clk25)
begin
   if rising_edge(clk25) then
      if(vga_vsync = '1' and vga_hsync = '1') then
                        b1V                                     <= Rb1V;
                        b1H                                             <= Rb1H;
                        AliveMax(0)                     <= tmpM1;
                        AliveMax(2)                     <= tmpM2;
                        AliveMax(4)                     <= tmpM3;
                        AliveMax(6)                     <= tmpM4;
                        AliveMax(8)                     <= tmpM5;
                        CoorExplodeH                    <= TCoorExplodeH;
                        CoorExplodeV                    <= TCoorExplodeV;
                        planeExplodeH                   <= TplaneExplodeH;
                        planeExplodeV                   <= TplaneExplodeV;
                        CoorPlaneH                              <= TCoorPlaneH;
                        CoorPlaneV                              <= TCoorPlaneV;
                        clearScr                        <= TmpClearScr;
                        for i in 0 to 6 loop
                                CoorFlyBeeH(i)          <= TCoorFlyBeeH(i);
                                CoorFlyBeeV(i)          <= TCoorFlyBeeV(i);
                        end loop;
                        for j in 0 to 29 loop
                                CoorBeeBulletV(j)       <= TCoorBeeBulletV(j);
                                CoorBeeBulletH(j)       <= TCoorBeeBulletH(j);
                        end loop;
                        gH <= tmpgH; aH <= tmpaH; mH <= tmpmH; e1H <= tmpe1H;
                        oH <= tmpoH; vH <= tmpvH; e2H <= tmpe2H; rH <= tmprH;
                        gV <= tmpgV; aV <= tmpaV; mV <= tmpmV; e1V <= tmpe1V;
                        oV <= tmpoV; vV <= tmpvV; e2V <= tmpe2V; rV <= tmprV;
      end if;
   end if;
end process SyncProcess;

  -- Horizontal and vertical counters
HCounter : process (clk25)
begin
        if rising_edge(clk25) then
                if reset_n = '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_n = '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_n = '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_n = '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_n = '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_n = '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;

------------------------plane Bullet-------------------------
Bullet1HGen : process (clk)
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			b1Hshow <= '0';
		elsif Hcount = HSYNC + HBACK_PORCH + b1H then
			b1Hshow <= '1';
		elsif Hcount = HSYNC + HBACK_PORCH + BULLET_LONG + b1H then
			b1Hshow <= '0';
		end if;
	end if;
end process Bullet1HGen;

Bullet1VGen : process (clk)
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			b1Vshow <= '0';
		elsif Vcount = VSYNC + VBACK_PORCH - 1 + b1V then
			b1Vshow <= '1';
		elsif Vcount = VSYNC + VBACK_PORCH - 1 + b1V + BULLET_HEIGHT
then
			b1Vshow <= '0';
		end if;
	end if;
end process Bullet1VGen;

b1 <= b1Hshow and b1Vshow;

--------------------------plane--------------------------------
PlaneHGen: process (clk)
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			planeH <= '0';
		elsif Hcount = HSYNC + HBACK_PORCH + CoorPlaneH then
```

```
                        planeH <= '1';
                elsif Hcount = HSYNC + HBACK_PORCH + CoorPlaneH + PLANE_SIZE then
                        planeH <= '0';
                end if;
        end if;
end process PlaneHGen;

PlaneVGen: process (clk)
begin
        if rising_edge(clk) then
                if reset_n = '0' then
                        planeV <= '0';
                elsif Vcount = VSYNC + VBACK_PORCH - 1 + CoorPlaneV then
                        planeV <= '1';
                elsif Vcount = VSYNC + VBACK_PORCH - 1 + CoorPlaneV + PLANE_SIZE
then
                        planeV <= '0';
                end if;
        end if;
end process PlaneVGen;

planeG <= planeV and planeH;

-------------------------bee matrix-----------------------
BeeMaxHGen: process (clk)
begin
        if rising_edge(clk) then
                if reset_n = '0' then
                        beeMaxH <= '0';
                elsif Hcount = HSYNC + HBACK_PORCH + CoorBeeMaxH then
                        beeMaxH <= '1';
                elsif Hcount = HSYNC + HBACK_PORCH + CoorBeeMaxH +
BEEMAX_LONG then
                        beeMaxH <= '0';
                end if;
        end if;
end process BeeMaxHGen;

BeeMaxVGen: process (clk)
begin
        if rising_edge(clk) then
                if reset_n = '0' then
                        beeMaxV <= '0';
                elsif Vcount = VSYNC + VBACK_PORCH + CoorBeeMaxV - 1 then
                        beeMaxV <= '1';
                elsif Vcount = VSYNC + VBACK_PORCH + CoorBeeMaxV - 1 +
BEEMAX_HEIGHT then
                        beeMaxV <= '0';
                end if;
        end if;
end process BeeMaxVGen;
beeMaxG <= beeMaxV and beeMaxH;

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

```
                if reset_n = '0' then
                        ExplodeH <= '0';
                elsif Hcount = HSYNC + HBACK_PORCH + CoorExplodeH then
                        ExplodeH <= '1';
                elsif Hcount = HSYNC + HBACK_PORCH + CoorExplodeH + BEE_SIZE then
                        ExplodeH <= '0';
                end if;
        end if;
end process ExplodeHGen;

ExplodeVGen: process (clk)
begin
        if rising_edge(clk) then
                if reset_n = '0' then
                        ExplodeV <= '0';
                elsif Vcount = VSYNC + VBACK_PORCH + CoorExplodeV - 1 then
                        ExplodeV <= '1';
                elsif Vcount = VSYNC + VBACK_PORCH + CoorExplodeV - 1 + BEE_SIZE
then
                        ExplodeV <= '0';
                end if;
        end if;
end process ExplodeVGen;

ExplodeG <= ExplodeV and ExplodeH;

BigExplodeHGen: process (clk)
begin
        if rising_edge(clk) then
                if reset_n = '0' then
                        BigExplodeH <= '0';
                elsif Hcount = HSYNC + HBACK_PORCH + planeExplodeH then
                        BigExplodeH <= '1';
                elsif Hcount = HSYNC + HBACK_PORCH + planeExplodeH + PLANE_SIZE
then
                        BigExplodeH <= '0';
                end if;
        end if;
end process BigExplodeHGen;

BigExplodeVGen: process (clk)
begin
        if rising_edge(clk) then
                if reset_n = '0' then
                        BigExplodeV <= '0';
                elsif Vcount = VSYNC + VBACK_PORCH + planeExplodeV - 1 then
                        BigExplodeV <= '1';
                elsif Vcount = VSYNC + VBACK_PORCH + planeExplodeV - 1 +
PLANE_SIZE then
                        BigExplodeV <= '0';
                end if;
        end if;
end process bigExplodeVGen;

BigExplodeG <= BigExplodeV and BigExplodeH;
```

```
TextHGen: process (clk)
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			TextH <= '0';
		elsif Hcount = HSYNC + HBACK_PORCH + CoorTextH then
			TextH <= '1';
		elsif Hcount = HSYNC + HBACK_PORCH + CoorTextH + textMatrixLong
then
			TextH <= '0';
		end if;
	end if;
end process TextHGen;

TextVGen: process (clk)
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			TextV <= '0';
		elsif Vcount = VSYNC + VBACK_PORCH + CoorTextV - 1 then
			TextV <= '1';
		elsif Vcount = VSYNC + VBACK_PORCH + CoorTextV - 1 +
textMatrixHeight then
			TextV <= '0';
		end if;
	end if;
end process TextVGen;

TextG <= TextV and TextH;

BeeGen: process (clk)
variable resultTmpH :unsigned (9 downto 0);
variable resultTmpV :unsigned (9 downto 0);
variable resultH :integer;
variable resultV :integer;
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			big_beeG	<= '0';
			bee_PurpleG <= '0';
			bee_GreenG	<= '0';
			bee_RedG	<= '0';
		elsif BeeMaxG = '1' then
			resultTmpH := to_integer(Hcount) - HSYNC - HBACK_PORCH -
CoorBeeMaxH;
			resultH := to_integer(resultTmpH(9 downto 4));
			resultTmpV := to_integer(Vcount) - VSYNC - VBACK_PORCH -
CoorBeeMaxV + 1;
			resultV := to_integer(resultTmpV(9 downto 4));
			if AliveMax (resultV)(resultH) = '1' then
				if BeeTypeMax(resultV) = "11" then
					big_beeG	<= '1';
				elsif BeeTypeMax(resultV) = "10" then
					bee_RedG	<= '1';
				elsif BeeTypeMax(resultV) = "01" then
					bee_PurpleG	<= '1';
```

```
                                elsif BeeTypeMax(resultV) = "00" then
                                        bee_GreenG      <= '1';
                                end if;
                        else
                                big_beeG                <= '0';
                                bee_RedG                <= '0';
                                bee_PurpleG     <= '0';
                                bee_GreenG              <= '0';
                        end if;
                end if;
        end if;
end process BeeGen;

BoxProcess: process(clk)
begin
        if rising_edge(clk) then
                if reset_n = '0' then
                        boxG <= '0';
                elsif Hcount >= HSYNC + HBACK_PORCH + HACTIVE - BOX_LONG and
Vcount >= VSYNC + VBACK_PORCH - 1 then
                        boxG <= '1';
                elsif Hcount <= HSYNC + HBACK_PORCH + HACTIVE
                or Vcount <= VSYNC + VBACK_PORCH - 1 + BOX_HEIGHT then
                        boxG <= '0';
                end if;
        end if;
end process BoxProcess;

----------------------------flying bees------------------------------
FlyBeeHGen: process (clk)
begin
        if rising_edge(clk) then
                for i in 0 to 6 loop
                        if reset_n = '0' then
                                FlybeeH(i) <= '0';
                        elsif Hcount = HSYNC + HBACK_PORCH + CoorFlyBeeH(i) then
                                FlybeeH(i) <= '1';
                        elsif Hcount = HSYNC + HBACK_PORCH + CoorFlyBeeH(i) +
BEE_SIZE then
                                FlybeeH(i) <= '0';
                        end if;
                end loop;
        end if;
end process FlyBeeHGen;

FlyBeeVGen: process (clk)
begin
        if rising_edge(clk) then
                for i in 0 to 6 loop
                        if reset_n = '0' then
                                FlybeeV(i) <= '0';
                        elsif Vcount = VSYNC + VBACK_PORCH + CoorFlyBeeV(i) - 1 then
                                FlybeeV(i) <= '1';
                        elsif Vcount = VSYNC + VBACK_PORCH + CoorFlyBeeV(i) - 1 +
BEE_SIZE then
                                FlybeeV(i) <= '0';
```

```
                        end if;
                end loop;
        end if;
end process FlyBeeVGen;

FlybeeGBack(0) <= FlybeeV(0) and FlybeeH(0);
FlybeeGBack(1) <= FlybeeV(1) and FlybeeH(1);
FlybeeGBack(2) <= FlybeeV(2) and FlybeeH(2);
FlybeeGBack(3) <= FlybeeV(3) and FlybeeH(3);
FlybeeGBack(4) <= FlybeeV(4) and FlybeeH(4);
FlybeeGBack(5) <= FlybeeV(5) and FlybeeH(5);
FlybeeGBack(6) <= FlybeeV(6) and FlybeeH(6);


FlybeeGGen : process (clk)
variable colorSignal : unsigned (2 downto 0);
variable tmpSignal : unsigned (1 downto 0);
variable flyV : integer;
variable flyH : integer;
begin
        if rising_edge(clk) then
                for i in 0 to 6 loop
                        if FlybeeGBack(i) = '1' then
                                flyV := to_integer(Vcount) - VSYNC - VBACK_PORCH + 1 -
to_integer(CoorFlyBeeV(i));
                                flyH := to_integer(Hcount) - HSYNC - HBACK_PORCH - 1 -
to_integer(CoorFlyBeeH(i));
                                if FlyBeeType(i) = "11" then
                                        if FlyBeeAngle(i) = "0000" then
                                                colorSignal := big_bee(flyV, flyH);
                                        elsif FlyBeeAngle(i) = "0100" then
                                                colorSignal := bigbee30(16 - flyV, flyH);
                                        elsif FlyBeeAngle(i) = "1000" then
                                                colorSignal := bigbee45(16 - flyV, flyH);
                                        elsif FlyBeeAngle(i) = "1100" then
                                                colorSignal := bigbee30(16 - flyH - 1, flyV);
                                        elsif FlyBeeAngle(i) = "0001" then
                                                colorSignal := big_bee(flyH, flyV);
                                        elsif FlyBeeAngle(i) = "1110" then
                                                colorSignal := bigbee30(16 - flyH - 1, 16 -
flyV);
                                        elsif FlyBeeAngle(i) = "1010" then
                                                colorSignal := bigbee45(flyV, flyH);
                                        elsif FlyBeeAngle(i) = "0110" then
                                                colorSignal := bigbee30(flyV, flyH);
                                        elsif FlyBeeAngle(i) = "0010" then
                                                colorSignal := big_bee(16 - flyV, flyH);
                                        elsif FlyBeeAngle(i) = "0111" then
                                                colorSignal := bigbee30(flyV, 16 - flyH - 1);
                                        elsif FlyBeeAngle(i) = "1011" then
                                                colorSignal := bigbee45(flyV, 16 - flyH - 1);
                                        elsif FlyBeeAngle(i) = "1111" then
                                                colorSignal := bigbee30(flyH, 16 - flyV);
                                        elsif FlyBeeAngle(i) = "0011" then
                                                colorSignal := big_bee(16 - flyH - 1, flyV);
                                        elsif FlyBeeAngle(i) = "1101" then
```

```
                        colorSignal := bigbee30(flyH, flyV);
                    elsif FlyBeeAngle(i) = "1001" then
                        colorSignal := bigbee45(flyH, flyV);
                    elsif FlyBeeAngle(i) = "0101" then
                        colorSignal := bigbee30(16 - flyV, 16 - flyH -
1);
                    end if;
                    if colorSignal = "000" then
                        FlybeeG(i) <= '0';
                    else
                        FlybeeG(i) <= '1';
                    end if;
                else
                    if FlyBeeAngle(i) = "0000" then
                        tmpSignal := bee(flyV, flyH);
                    elsif FlyBeeAngle(i) = "0100" then
                        tmpSignal := bee30(16 - flyV, flyH);
                    elsif FlyBeeAngle(i) = "1000" then
                        tmpSignal := bee45(16 - flyV, flyH);
                    elsif FlyBeeAngle(i) = "1100" then
                        tmpSignal := bee30(16 - flyH, flyV);
                    elsif FlyBeeAngle(i) = "0001" then
                        tmpSignal := bee(flyH, flyV);
                    elsif FlyBeeAngle(i) = "1110" then
                        tmpSignal := bee30(16 - flyH, 16 - flyV);
                    elsif FlyBeeAngle(i) = "1010" then
                        tmpSignal := bee45(flyV, flyH);
                    elsif FlyBeeAngle(i) = "0110" then
                        tmpSignal := bee30(flyV, flyH);
                    elsif FlyBeeAngle(i) = "0010" then
                        tmpSignal := bee(16 - flyV, flyH);
                    elsif FlyBeeAngle(i) = "0111" then
                        tmpSignal := bee30(flyV, 16 - flyH - 1);
                    elsif FlyBeeAngle(i) = "1011" then
                        tmpSignal := bee45(flyV, 16 - flyH - 1);
                    elsif FlyBeeAngle(i) = "1111" then
                        tmpSignal := bee30(flyH, 16 - flyV);
                    elsif FlyBeeAngle(i) = "0011" then
                        tmpSignal := bee(16 - flyH - 1, flyV);
                    elsif FlyBeeAngle(i) = "1101" then
                        tmpSignal := bee30(flyH, flyV);
                    elsif FlyBeeAngle(i) = "1001" then
                        tmpSignal := bee45(flyH, flyV);
                    elsif FlyBeeAngle(i) = "0101" then
                        tmpSignal := bee30(16 - flyV, 16 - flyH - 1);
                    end if;

                    if tmpSignal = "10" then
                        colorSignal := "111";
                    elsif tmpSignal = "11" then
                        colorSignal := "100";
                    elsif FlyBeeType(i) = "10" and tmpSignal = "01" then
                        colorSignal := "011";
                    elsif FlyBeeType(i) = "00" and tmpSignal = "01" then
                        colorSignal := "110";
                    elsif FlyBeeType(i) = "01" and tmpSignal = "01" then
```

```
                                        colorSignal := "001";
                                    end if;

                                    if tmpSignal = "00" then
                                        FlybeeG(i) <= '0';
                                    else
                                        FlybeeG(i) <= '1';
                                    end if;
                                end if;
                                TransColorSignal <= colorSignal;
                            end if;
                        end loop;
                end if;
end process FlybeeGGen;

----------------------------bee bullet---------------------------------
BeeBulletHGen : process (clk)
begin
        if rising_edge(clk) then
                for i in 0 to 29 loop
                        if reset_n = '0' then
                                beeBulletH(i) <= '0';
                        elsif Hcount = HSYNC + HBACK_PORCH + coorBeeBulletH(i) then
                                beeBulletH(i) <= '1';
                        elsif Hcount = HSYNC + HBACK_PORCH + BULLET_LONG +
coorBeeBulletH(i) then
                                beeBulletH(i) <= '0';
                        end if;
                end loop;
        end if;
end process BeeBulletHGen;

BeeBulletVGen : process (clk)
begin
        if rising_edge(clk) then
                for i in 0 to 29 loop
                        if reset_n = '0' then
                                beeBulletV(i) <= '0';
                        elsif Vcount = VSYNC + VBACK_PORCH - 1 + coorBeeBulletV(i)
then
                                beeBulletV(i) <= '1';
                        elsif Vcount = VSYNC + VBACK_PORCH - 1 + BULLET_HEIGHT +
coorBeeBulletV(i) then
                                beeBulletV(i) <= '0';
                        end if;
                end loop;
        end if;
end process BeeBulletVGen;

BeeBulletGGen: process (clk)
variable tmp: std_logic;
begin
        if rising_edge(clk) then
                tmp := '0';
                for i in 0 to 29 loop
                        beeBulletG(i) <= beeBulletH(i) and beeBulletV(i);
```

```
                        if (beeBulletG(i) = '1' and tmp = '0') then
                                tmp := '1';
                        end if;
                end loop;
                getBullet <= tmp;
        end if;
end process BeeBulletGGen;

----------------------------Score process----------------------------
ScoreProcess : process (clk)
variable H : unsigned (9 downto 0);
variable V : unsigned (9 downto 0);
variable num : integer;
variable color : std_logic;
begin
        if rising_edge(clk) then
                H := Hcount - HSYNC - HBACK_PORCH - 1 - CoorTextH - 10;
                V := Vcount - VSYNC - VBACK_PORCH + 1 - 65;

                if to_integer(H) >= 0 and to_integer(H) <= 39 and
                        to_integer(V) >= 0 and to_integer(V) <= 9 then
                        num := to_integer(hiScoreData(to_integer(H(9 downto 3))));
                        if num = 1 then
                                color := one(to_integer(V))(to_integer(H(2 downto 0)));
                        elsif num = 2 then
                                color := two(to_integer(V))(to_integer(H(2 downto 0)));
                        elsif num = 3 then
                                color := three(to_integer(V))(to_integer(H(2 downto 0)));
                        elsif num = 4 then
                                color := four(to_integer(V))(to_integer(H(2 downto 0)));
                        elsif num = 5 then
                                color := five(to_integer(V))(to_integer(H(2 downto 0)));
                        elsif num = 6 then
                                color := six(to_integer(V))(to_integer(H(2 downto 0)));
                        elsif num = 7 then
                                color := seven(to_integer(V))(to_integer(H(2 downto 0)));
                        elsif num = 8 then
                                color := eight(to_integer(V))(to_integer(H(2 downto 0)));
                        elsif num = 9 then
                                color := nine(to_integer(V))(to_integer(H(2 downto 0)));
                        elsif num = 0 then
                                color := zero(to_integer(V))(to_integer(H(2 downto 0)));
                        end if;
                        if color = '0' then
                                hiScoreColorSignal <= "000";
                        elsif color = '1' then
                                hiScoreColorSignal <= "111";
                        end if;
                end if;
        end if;
end process ScoreProcess;

ScoreProcess2 : process (clk)
variable H2 : unsigned (9 downto 0);
variable V2 : unsigned (9 downto 0);
variable num : integer;
```

```
variable color : std_logic;
begin
	if rising_edge(clk) then

		H2 := Hcount - HSYNC - HBACK_PORCH - 1 - CoorTextH - 10;
		V2 := Vcount - VSYNC - VBACK_PORCH + 1 - 95;

		if to_integer(H2) >= 0 and to_integer(H2) <= 39 and
			to_integer(V2) >= 0 and to_integer(V2) <= 9 then
			num := to_integer(scoreData(to_integer(H2(9 downto 3))));
			if num = 1 then
				color := one(to_integer(V2))(to_integer(H2(2 downto 0)));
			elsif num = 2 then
				color := two(to_integer(V2))(to_integer(H2(2 downto 0)));
			elsif num = 3 then
				color := three(to_integer(V2))(to_integer(H2(2 downto 0)));
			elsif num = 4 then
				color := four(to_integer(V2))(to_integer(H2(2 downto 0)));
			elsif num = 5 then
				color := five(to_integer(V2))(to_integer(H2(2 downto 0)));
			elsif num = 6 then
				color := six(to_integer(V2))(to_integer(H2(2 downto 0)));
			elsif num = 7 then
				color := seven(to_integer(V2))(to_integer(H2(2 downto 0)));
			elsif num = 8 then
				color := eight(to_integer(V2))(to_integer(H2(2 downto 0)));
			elsif num = 9 then
				color := nine(to_integer(V2))(to_integer(H2(2 downto 0)));
			elsif num = 0 then
				color := zero(to_integer(V2))(to_integer(H2(2 downto 0)));
			end if;
			if color = '0' then
				scoreColorSignal <= "000";
			elsif color = '1' then
				scoreColorSignal <= "010";
			end if;
		end if;
	end if;
end process ScoreProcess2;

------------------------------stars background--------------------------
starGen: process (clk)
begin
	if rising_edge(clk) then
		starH <= '0';
		starH1 <= '0';
		starH2 <= '0';
		starH3 <= '0';
		for i in 0 to 6 loop
			if
			(to_integer(Hcount(9 downto 0))-HSYNC-HBACK_PORCH-
to_integer(coorstarH)+88*i =0) and
			(to_integer(Vcount(9 downto 0))-VSYNC-VBACK_PORCH-
roll(i) >=0)and
			(to_integer(Vcount(9 downto 0))-VSYNC-VBACK_PORCH-roll(i)
<=1)then
```

```
                                starH <= '1';
                        end if;
                end loop;
                for j in 0 to 6 loop
                        if
                        (to_integer(Hcount(9 downto 0))-HSYNC-HBACK_PORCH-
to_integer(coorstarH)+88*j+22 =0) and
                        (to_integer(Vcount(9 downto 0))-VSYNC-VBACK_PORCH-
roll(j+7) >=0)and
                        (to_integer(Vcount(9 downto 0))-VSYNC-VBACK_PORCH-roll(j+7)
<=1)then
                                starH1 <= '1';
                        end if;
                end loop;
                for m in 0 to 6 loop
                        if
                        (to_integer(Hcount(9 downto 0))-HSYNC-HBACK_PORCH-
to_integer(coorstarH)+88*m+44 =0) and
                        (to_integer(Vcount(9 downto 0))-VSYNC-VBACK_PORCH-
roll(m+14) >=0)and
                       (to_integer(Vcount(9  downto  0))-VSYNC-VBACK_PORCH-roll(m+14)
<=1)then
                                starH2 <= '1';
                        end if;
                end loop;

                for n in 0 to 6 loop
                        if
                        (to_integer(Hcount(9 downto 0))-HSYNC-HBACK_PORCH-
to_integer(coorstarH)+88*n+66 =0) and
                        (to_integer(Vcount(9 downto 0))-VSYNC-VBACK_PORCH-
roll(n+21) >=0)and
                        (to_integer(Vcount(9 downto 0))-VSYNC-VBACK_PORCH-roll(n+21)
<=1)
                         then
                        starH3 <= '1';
                        end if;
                end loop;
        end if;
end process starGen;
-----------------------------------------------

starG <= starH and flipstate;
starG1 <= starH1 and flipstate1;
starG2 <= starH2 and flipstate2;
starG3 <= starH3 and flipstate3;
-----------------------------------------------
flipGen: process (clk)
begin
        if rising_edge(clk) then
                if EndOfField = '1' then
                        for l in 0 to 27 loop
                                if to_integer(coorstarV)+stararrayV(l)>= 480 then
                                        roll(l)<=to_integer(coorstarV)+stararrayV(l)-480;
                                else
                                        roll(l)<=to_integer(coorstarV)+stararrayV(l);
```

```
		end if;
	end loop;
	if shinecount =1600 then
		shinecount <= 0;
		if coorstarV = "0111011110" then
			coorstarV <= "0000000000";
		else
			coorstarV <= coorstarV + "0000000001";
		end if;
	else
		shinecount <= shinecount +1;
	end if;

	if flipcount = 40000 then
		flipcount <= 0;
		if flipstate ='0' then
			flipstate <= '1';
		elsif flipstate ='1' then
			flipstate <= '0';
		end if;

		if shine = "111" then
			shine <= "001";
		else
			shine <= shine + "001";
		end if;

		if flipstate2 ='0' then
			flipstate2 <= '1';
		elsif flipstate2 ='1' then
			flipstate2 <= '0';
		end if;

		if shine2 = "001" then
			shine2 <= "111";
		else
			shine2 <= shine2 - "001";
		end if;
	elsif flipcount = 20000 then
		flipcount <= flipcount +1;
		if flipstate1 ='0' then
			flipstate1 <= '1';
		elsif flipstate1 ='1' then
			flipstate1 <= '0';
		end if;
		if shine1 = "111" then
			shine1 <= "001";
		else
			shine1 <= shine1 + "001";
		end if;
		if flipstate3 ='0' then
			flipstate3 <= '1';
		elsif flipstate3 ='1' then
			flipstate3 <= '0';
		end if;
		if shine3 = "111" then
```

```
                                        shine3 <= "001";
                                    else
                                        shine3 <= shine3 + "001";
                                    end if;
                                else
                                    flipcount <= flipcount +1;
                                end if;
                            end if;
                        end if;
end process flipGen;

galaxianHGen : process (clk)
begin
        if rising_edge(clk) then
                if reset_n = '0' then
                        glaH <= '0';
                elsif Hcount = HSYNC + HBACK_PORCH + CoorGlaH then
                        glaH <= '1';
                elsif Hcount = HSYNC + HBACK_PORCH + GLA_LONG + CoorGlaH then
                        glaH <= '0';
                end if;
        end if;
end process galaxianHGen;

galaxianVGen : process (clk)
begin
        if rising_edge(clk) then
                if reset_n = '0' then
                        glaV <= '0';
                elsif Vcount = VSYNC + VBACK_PORCH - 1 + mainPicV then
                        glaV <= '1';
                elsif (Vcount = VSYNC + VBACK_PORCH - 1 + GLA_HEIGHT + mainPicV)
or
                                        Vcount = VSYNC + VBACK_PORCH - 1 + VACTIVE then
                        glaV <= '0';
                end if;
        end if;
end process galaxianVGen;

glaG <= glaH and glaV;

galaxianGen : process (clk)
variable h,v : integer;
begin
        if rising_edge(clk) then
                if glaG = '1' and mainPic = '1' then
                        h := to_integer(Hcount) - (HSYNC + HBACK_PORCH + CoorGlaH );
                        v := to_integer(Vcount) - (VSYNC + VBACK_PORCH - 1 +
to_integer(mainPicV));
                        if v < 100 then
                                --------- big picture
                                if h < 50 then
                                        galaxianColor <= galaxian1(v,h);
                                elsif h >= 50 and h < 100 then
                                        galaxianColor <= galaxian2(v,h - 50);
                                elsif h >= 100 and h < 150 then
```

```
                                        galaxianColor <= galaxian3(v,h - 100);
                                elsif h >= 150 and h < 200 then
                                        galaxianColor <= galaxian4(v,h - 150);
                                elsif h >= 200 and h < 250 then
                                        galaxianColor <= galaxian5(v,h - 200);
                                end if;
                        elsif v >= 205 and v < 220 and h >= 65 and h < 193 then      ---------
trademark
                                if trademark(v - 205)(h - 65) = '1' then
                                        galaxianColor <= "010";
                                else
                                        galaxianColor <= "000";
                                end if;
                        elsif v >= 119 and v < 137  and h >= 87 and h < 103 then --------- cursor
                                if cursor(v - 119)(h - 87) = '1' then
                                        galaxianColor <= "100";
                                else
                                        galaxianColor <= "000";
                                end if;
                        elsif v >= 120 and v < 134 and h >= 108 and h < 158 then --------- start
                                if start(v - 120)(h - 108) = '1' then
                                        galaxianColor <= "001";
                                else
                                        galaxianColor <= "000";
                                end if;
                        elsif v >= 160 and v < 174  and h >= 45 and h < 185 then ---------
xiaotian qi
                                if xiaotian(v - 160)(h - 45) = '1' then
                                        galaxianColor <= "010";
                                else
                                        galaxianColor <= "000";
                                end if;
                        elsif v >= 180 and v < 194  and h >= 45 and h < 205 then ---------
yaolong feng
                                if yaolong(v - 180)(h - 45) = '1' then
                                        galaxianColor <= "010";
                                else
                                        galaxianColor <= "000";
                                end if;
                        else
                                galaxianColor <= "000";
                        end if;
                end if;
        end if;
end process galaxianGen;

-------------------------sync with software-----------------------
timeCountDelay: process (clk)
begin
        if rising_edge(clk) then
                if startCount = '1' then
                        timeDelayCount <= 0;
                end if;
                if timeDelayCount = synctime then
                        dataSendBack <= x"0000000F";
                else
```

```
                        timeDelayCount <= timeDelayCount + 1;
                        dataSendBack <= (others => '1');
                end if;
        end if;
end process timeCountDelay;

-------------------------- wind flip -----------------------------
timeCountDelay2: process (clk)
begin
        if rising_edge(clk) then
                if timeDelayCount = synctime then
                        if windFlipCount = 20000 then
                                windFlip <= not(windFlip);
                                windFlipCount <= 0;
                        else
                                windFlipCount <= windFlipCount + 1;
                        end if;
                end if;
        end if;
end process timeCountDelay2;

------------------------ready---------------------------
readyHGen : process (clk)
begin
        if rising_edge(clk) then
                if reset_n = '0' or readySignal = '0' then
                        readyH <= '0';
                elsif Hcount = HSYNC + HBACK_PORCH + CoorReadyH then
                        readyH <= '1';
                elsif Hcount = HSYNC + HBACK_PORCH + READY_LONG + CoorReadyH
then
                        readyH <= '0';
                end if;
        end if;
end process readyHGen;

readyVGen : process (clk)
begin
        if rising_edge(clk) then
                if reset_n = '0' or readySignal = '0' then
                        readyV <= '0';
                elsif Vcount = VSYNC + VBACK_PORCH - 1 + CoorReadyV then
                        readyV <= '1';
                elsif Vcount = VSYNC + VBACK_PORCH - 1 + READY_HEIGHT +
CoorReadyV then
                        readyV <= '0';
                end if;
        end if;
end process readyVGen;

readyG <= readyH and readyV;

readyGGen: process (clk)
begin
        if rising_edge(clk) then
                if readyG = '1' then
```

```
                        if (ready(to_integer(Vcount) - (VSYNC + VBACK_PORCH - 1 +
CoorReadyV))
                        (to_integer(Hcount) - (HSYNC + HBACK_PORCH + CoorReadyH)) =
'1') then
                                readyColor <= "011";
                        else
                                readyColor <= "000";
                        end if;
                end if;
        end if;
end process readyGGen;

----------------------pause----------------------
pauseHGen : process (clk)
begin
        if rising_edge(clk) then
                if reset_n = '0' or pauseSignal = '0' then
                        pauseH <= '0';
                elsif Hcount = HSYNC + HBACK_PORCH + CoorReadyH then
                        pauseH <= '1';
                elsif Hcount = HSYNC + HBACK_PORCH + READY_LONG + CoorReadyH
then
                        pauseH <= '0';
                end if;
        end if;
end process pauseHGen;

pauseVGen : process (clk)
begin
        if rising_edge(clk) then
                if reset_n = '0' or pauseSignal = '0' then
                        pauseV <= '0';
                elsif Vcount = VSYNC + VBACK_PORCH - 1 + CoorReadyV then
                        pauseV <= '1';
                elsif Vcount = VSYNC + VBACK_PORCH - 1 + READY_HEIGHT +
CoorReadyV then
                        pauseV <= '0';
                end if;
        end if;
end process pauseVGen;

pauseG <= pauseH and pauseV;

pauseGGen: process (clk)
begin
        if rising_edge(clk) then
                if pauseG = '1' then
                        if (pause(to_integer(Vcount) - (VSYNC + VBACK_PORCH - 1 +
CoorReadyV))
                        (to_integer(Hcount) - (HSYNC + HBACK_PORCH + CoorReadyH)) =
'1') then
                                pauseColor <= "011";
                        else
                                pauseColor <= "000";
                        end if;
                end if;
```

```
		end if;
end process pauseGGen;

-------------------------gameover----------------------
gameover_g_HGen : process (clk)
begin
		if rising_edge(clk) then
			if reset_n = '0' then
				gHG <= '0';
			elsif Hcount = HSYNC + HBACK_PORCH + gH then
				gHG <= '1';
			elsif Hcount = HSYNC + HBACK_PORCH + ALPHALONG + gH then
				gHG <= '0';
			end if;
		end if;
end process gameover_g_HGen;

gameover_g_VGen : process (clk)
begin
		if rising_edge(clk) then
			if reset_n = '0' then
				gVG <= '0';
			elsif Vcount = VSYNC + VBACK_PORCH - 1 + gV then
				gVG <= '1';
			elsif Vcount = VSYNC + VBACK_PORCH - 1 + ALPHAHEIGHT + gV then
				gVG <= '0';
			end if;
		end if;
end process gameover_g_VGen;

gG <= gHG and gVG;

gameover_a_HGen : process (clk)
begin
		if rising_edge(clk) then
			if reset_n = '0' then
				aHG <= '0';
			elsif Hcount = HSYNC + HBACK_PORCH + aH then
				aHG <= '1';
			elsif Hcount = HSYNC + HBACK_PORCH + ALPHALONG + aH then
				aHG <= '0';
			end if;
		end if;
end process gameover_a_HGen;

gameover_a_VGen : process (clk)
begin
		if rising_edge(clk) then
			if reset_n = '0' then
				aVG <= '0';
			elsif Vcount = VSYNC + VBACK_PORCH - 1 + aV then
				aVG <= '1';
			elsif Vcount = VSYNC + VBACK_PORCH - 1 + ALPHAHEIGHT + aV then
				aVG <= '0';
			end if;
		end if;
```

```
end process gameover_a_VGen;

aG <= aHG and aVG;

gameover_m_HGen : process (clk)
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			mHG <= '0';
		elsif Hcount = HSYNC + HBACK_PORCH + mH then
			mHG <= '1';
		elsif Hcount = HSYNC + HBACK_PORCH + ALPHALONG + mH then
			mHG <= '0';
		end if;
	end if;
end process gameover_m_HGen;

gameover_m_VGen : process (clk)
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			mVG <= '0';
		elsif Vcount = VSYNC + VBACK_PORCH - 1 + mV then
			mVG <= '1';
		elsif Vcount = VSYNC + VBACK_PORCH - 1 + ALPHAHEIGHT + mV then
			mVG <= '0';
		end if;
	end if;
end process gameover_m_VGen;

mG <= mHG and mVG;

gameover_e1_HGen : process (clk)
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			e1HG <= '0';
		elsif Hcount = HSYNC + HBACK_PORCH + e1H then
			e1HG <= '1';
		elsif Hcount = HSYNC + HBACK_PORCH + ALPHALONG + e1H then
			e1HG <= '0';
		end if;
	end if;
end process gameover_e1_HGen;

gameover_e1_VGen : process (clk)
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			e1VG <= '0';
		elsif Vcount = VSYNC + VBACK_PORCH - 1 + e1V then
			e1VG <= '1';
		elsif Vcount = VSYNC + VBACK_PORCH - 1 + ALPHAHEIGHT + e1V then
			e1VG <= '0';
		end if;
	end if;
```

```
end process gameover_e1_VGen;

e1G <= e1HG and e1VG;

gameover_o_HGen : process (clk)
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			oHG <= '0';
		elsif Hcount = HSYNC + HBACK_PORCH + oH then
			oHG <= '1';
		elsif Hcount = HSYNC + HBACK_PORCH + ALPHALONG + oH then
			oHG <= '0';
		end if;
	end if;
end process gameover_o_HGen;

gameover_o_VGen : process (clk)
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			oVG <= '0';
		elsif Vcount = VSYNC + VBACK_PORCH - 1 + oV then
			oVG <= '1';
		elsif Vcount = VSYNC + VBACK_PORCH - 1 + ALPHAHEIGHT + oV then
			oVG <= '0';
		end if;
	end if;
end process gameover_o_VGen;

oG <= oVG and oHG;

gameover_v_HGen : process (clk)
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			vHG <= '0';
		elsif Hcount = HSYNC + HBACK_PORCH + vH then
			vHG <= '1';
		elsif Hcount = HSYNC + HBACK_PORCH + ALPHALONG + vH then
			vHG <= '0';
		end if;
	end if;
end process gameover_v_HGen;

gameover_v_VGen : process (clk)
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			vVG <= '0';
		elsif Vcount = VSYNC + VBACK_PORCH - 1 + vV then
			vVG <= '1';
		elsif Vcount = VSYNC + VBACK_PORCH - 1 + ALPHAHEIGHT + vV then
			vVG <= '0';
		end if;
	end if;
```

```
end process gameover_v_VGen;

vG <= vHG and vVG;

gameover_e2_HGen : process (clk)
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			e2HG <= '0';
		elsif Hcount = HSYNC + HBACK_PORCH + e2H then
			e2HG <= '1';
		elsif Hcount = HSYNC + HBACK_PORCH + ALPHALONG + e2H then
			e2HG <= '0';
		end if;
	end if;
end process gameover_e2_HGen;

gameover_e2_VGen : process (clk)
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			e2VG <= '0';
		elsif Vcount = VSYNC + VBACK_PORCH - 1 + e2V then
			e2VG <= '1';
		elsif Vcount = VSYNC + VBACK_PORCH - 1 + ALPHAHEIGHT + e2V then
			e2VG <= '0';
		end if;
	end if;
end process gameover_e2_VGen;

e2G <= e2HG and e2VG;

gameover_r_HGen : process (clk)
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			rHG <= '0';
		elsif Hcount = HSYNC + HBACK_PORCH + rH then
			rHG <= '1';
		elsif Hcount = HSYNC + HBACK_PORCH + ALPHALONG + rH then
			rHG <= '0';
		end if;
	end if;
end process gameover_r_HGen;

gameover_r_VGen : process (clk)
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			rVG <= '0';
		elsif Vcount = VSYNC + VBACK_PORCH - 1 + rV then
			rVG <= '1';
		elsif Vcount = VSYNC + VBACK_PORCH - 1 + ALPHAHEIGHT + rV then
			rVG <= '0';
		end if;
	end if;
```

```
end process gameover_r_VGen;

rG <= rHG and rVG;

gameoverGen : process (clk)
variable ch, cv: integer;
begin
	if rising_edge(clk) then
		if gG = '1' then
			ch := to_integer(Hcount) - (HSYNC + HBACK_PORCH +
to_integer(gH));
			cv := to_integer(Vcount) - (VSYNC + VBACK_PORCH - 1 +
to_integer(gV));
			gameoverSignal <= g(cv)(ch);
		elsif aG = '1' then
			ch := to_integer(Hcount) - (HSYNC + HBACK_PORCH +
to_integer(aH));
			cv := to_integer(Vcount) - (VSYNC + VBACK_PORCH - 1 +
to_integer(aV));
			gameoverSignal <= a(cv)(ch);
		elsif mG = '1' then
			ch := to_integer(Hcount) - (HSYNC + HBACK_PORCH +
to_integer(mH));
			cv := to_integer(Vcount) - (VSYNC + VBACK_PORCH - 1 +
to_integer(mV));
			gameoverSignal <= m(cv)(ch);
		elsif e1G = '1' then
			ch := to_integer(Hcount) - (HSYNC + HBACK_PORCH +
to_integer(e1H));
			cv := to_integer(Vcount) - (VSYNC + VBACK_PORCH - 1 +
to_integer(e1V));
			gameoverSignal <= e(cv)(ch);
		elsif oG = '1' then
			ch := to_integer(Hcount) - (HSYNC + HBACK_PORCH +
to_integer(oH));
			cv := to_integer(Vcount) - (VSYNC + VBACK_PORCH - 1 +
to_integer(oV));
			gameoverSignal <= o(cv)(ch);
		elsif vG = '1' then
			ch := to_integer(Hcount) - (HSYNC + HBACK_PORCH +
to_integer(vH));
			cv := to_integer(Vcount) - (VSYNC + VBACK_PORCH - 1 +
to_integer(vV));
			gameoverSignal <= v(cv)(ch);
		elsif e2G = '1' then
			ch := to_integer(Hcount) - (HSYNC + HBACK_PORCH +
to_integer(e2H));
			cv := to_integer(Vcount) - (VSYNC + VBACK_PORCH - 1 +
to_integer(e2V));
			gameoverSignal <= e(cv)(ch);
		elsif rG = '1' then
			ch := to_integer(Hcount) - (HSYNC + HBACK_PORCH +
to_integer(rH));
			cv := to_integer(Vcount) - (VSYNC + VBACK_PORCH - 1 +
to_integer(rV));
			gameoverSignal <= r(cv)(ch);
```

```
                end if;

                if gameoverSignal = '1' then
                        gameoverColorSignal <= "010";
                else
                        gameoverColorSignal <= "000";
                end if;
        end if;
end process gameoverGen;

----------------------red flag-----------------------
infoHGen : process(clk)
begin
        if rising_edge(clk) then
                if reset_n = '0' then
                        infoH <= '0';
                elsif Hcount = HSYNC + HBACK_PORCH + INFO_H then
                        infoH <= '1';
                elsif Hcount = HSYNC + HBACK_PORCH + info_size + INFO_H then
                        infoH <= '0';
                end if;
        end if;
end process infoHGen;

infoVGen : process(clk)
begin
        if rising_edge(clk) then
                if reset_n = '0' then
                        infoV <= '0';
                elsif Vcount = VSYNC + VBACK_PORCH - 1 + INFO_V then
                        infoV <= '1';
                elsif Vcount = VSYNC + VBACK_PORCH - 1 + info_size + INFO_V then
                        infoV <= '0';
                end if;
        end if;
end process infoVGen;

infoG <= infoH and infoV;

infoGen :process(clk)
variable H : integer;
variable V : integer;
variable tmpV : integer;
variable num : integer;
variable color : std_logic;
begin
        if rising_edge(clk) then
                if infoG = '1' then
                        H := to_integer(Hcount) - HSYNC - HBACK_PORCH - 1 - INFO_H;
                        V := to_integer(Vcount) - VSYNC - VBACK_PORCH + 1 - INFO_V;
                        if H >= 0 and H < 20 and V >= 0 and V < 20 then
                                infoColor <= red_flag(V, H);
                        elsif H >= 0 and H < 16 and V >= 30 and V < 46 then
                                infoColor <= small_plane(V - 30, H);
                        elsif (H >= 25 and H < 31 and V >= 5 and V < 15) then
                                tmpV := V - 5;
```

```
		num := to_integer(planeLife);
		if num = 1 then
			color := one(tmpV)(H - 25);
		elsif num = 2 then
			color := two(tmpV)(H - 25);
		elsif num = 3 then
			color := three(tmpV)(H - 25);
		elsif num = 4 then
			color := four(tmpV)(H - 25);
		elsif num = 5 then
			color := five(tmpV)(H - 25);
		elsif num = 6 then
			color := six(tmpV)(H - 25);
		elsif num = 7 then
			color := seven(tmpV)(H - 25);
		elsif num = 8 then
			color := eight(tmpV)(H - 25);
		elsif num = 9 then
			color := nine(tmpV)(H - 25);
		elsif num = 0 then
			color := zero(tmpV)(H - 25);
		end if;
		if color = '0' then
			infoColor <= "000";
		elsif color = '1' then
			infoColor <= "100";
		end if;
	elsif (H >= 25 and H < 31 and V >= 35 and V < 45) then
		tmpV := V - 35;
		num := to_integer(level);
		if num = 1 then
			color := one(tmpV)(H - 25);
		elsif num = 2 then
			color := two(tmpV)(H - 25);
		elsif num = 3 then
			color := three(tmpV)(H - 25);
		elsif num = 4 then
			color := four(tmpV)(H - 25);
		elsif num = 5 then
			color := five(tmpV)(H - 25);
		elsif num = 6 then
			color := six(tmpV)(H - 25);
		elsif num = 7 then
			color := seven(tmpV)(H - 25);
		elsif num = 8 then
			color := eight(tmpV)(H - 25);
		elsif num = 9 then
			color := nine(tmpV)(H - 25);
		elsif num = 0 then
			color := zero(tmpV)(H - 25);
		end if;
		if color = '0' then
			infoColor <= "000";
		elsif color = '1' then
			infoColor <= "100";
		end if;
```

```
                        else
                                infoColor <= "000";
                        end if;
                end if;
        end if;
end process infoGen;

VideoOut: process (clk, reset_n)
variable colorSignal : unsigned (2 downto 0);
variable flyV : integer;
variable flyH : integer;
variable tmpTextH: integer;
variable tmpTextV: integer;
variable tmpSignal : unsigned (1 downto 0);
variable tmpExplodeH, tmpExplodeV : integer;
begin
        if reset_n = '0' then
                VGA_R <= "0000000000";
                VGA_G <= "0000000000";
                VGA_B <= "0000000000";
        elsif clk'event and clk = '1' then
                if glaG = '1' and galaxianColor /= "000" and mainPic = '1' then
                        colorSignal := galaxianColor;
                elsif TextG = '1' then
                        tmptextH := to_integer(Hcount) - HSYNC - HBACK_PORCH - 1 -
CoorTextH;
                        tmptextV := to_integer(Vcount) - VSYNC - VBACK_PORCH + 1 -
CoorTextV;
                        if tmptextH >= 0 and tmptextH <= 54 and tmptextV >= 0 and tmptextV
<= 9 then
                                if hiScore (tmptextV)(tmptextH) = '1' then
                                        colorSignal := "011";
                                else
                                        colorSignal := "000";
                                end if;
                        elsif tmptextH >= 10 and tmptextH <= 49 and tmptextV >= 15 and
tmptextV <= 24 then
                                colorSignal := hiScoreColorSignal;
                        elsif tmptextH >= 0 and tmptextH <= 21 and tmptextV >= 30 and
tmptextV <= 39 then
                                if oneUP (tmpTextV - 30)(tmpTextH) = '1' then
                                        colorSignal := "011";
                                else
                                        colorSignal := "000";
                                end if;
                        elsif tmptextH >= 10 and tmptextH <= 49 and tmptextV >= 45 and
tmptextV <= 54 then
                                colorSignal := scoreColorSignal;
                        end if;
                elsif infoG = '1' then
                        colorSignal := infoColor;
                elsif boxG = '1' then
                        colorSignal := "000";
                elsif (FlybeeG(0) = '1' or FlybeeG(1) = '1' or FlybeeG(2) = '1' or FlybeeG(3) = '1'
or
```

```
                                FlybeeG(4) = '1' or FlybeeG(5) = '1' or FlybeeG(6) = '1') and
mainPic = '0' then
                                colorSignal := TransColorSignal;
                        elsif readyG = '1' then
                                colorSignal := readyColor;
                        elsif pauseG = '1' then
                                colorSignal := pauseColor;
                        elsif gG = '1' or aG = '1' or mG = '1' or e1G = '1' or
                                oG = '1' or vG = '1' or e2G = '1' or rG = '1' then
                                colorSignal := gameoverColorSignal;
                        elsif ExplodeG = '1' and mainPic = '0' then
                                if Small = '1' then
                                colorSignal := small_explode(
                                to_integer(Vcount) - VSYNC - VBACK_PORCH + 1 -
to_integer(CoorExplodeV),
                                to_integer(Hcount) - HSYNC - HBACK_PORCH - 1 -
to_integer(CoorExplodeH));
                                elsif Small = '0' then
                                colorSignal := explode(
                                to_integer(Vcount) - VSYNC - VBACK_PORCH + 1 -
to_integer(CoorExplodeV),
                                to_integer(Hcount) - HSYNC - HBACK_PORCH - 1 -
to_integer(CoorExplodeH));
                                end if;
                        elsif BigExplodeG = '1' and mainPic = '0' then                  -- plane
Explode
                                tmpExplodeH := to_integer(Hcount) - HSYNC - HBACK_PORCH - 1 -
to_integer(planeExplodeH);
                                tmpExplodeV := to_integer(Vcount) - VSYNC - VBACK_PORCH + 1 -
to_integer(planeExplodeV);
                                if planeSmall = '1' then
                                        if tmpExplodeH < 16 and tmpExplodeV < 16 then
                                                colorSignal := explode(tmpExplodeV, tmpExplodeH);
                                        end if;
                                elsif planeSmall = '0' then
                                        colorSignal := big_explode(tmpExplodeV, tmpExplodeH);
                                end if;
                        elsif planeG = '1' and mainPic = '0' then
                                colorSignal := plane(
                                to_integer(Vcount) - VSYNC - VBACK_PORCH + 1 -
to_integer(CoorPlaneV),
                                to_integer(Hcount) - HSYNC - HBACK_PORCH - 1 -
to_integer(CoorPlaneH));
                        elsif big_beeG = '1' and mainPic = '0' then
                                colorSignal := big_bee(
                                to_integer(Vcount) - VSYNC - VBACK_PORCH + 1 -
to_integer(CoorBeeMaxV),
                                to_integer(Hcount) - HSYNC - HBACK_PORCH - 1 -
to_integer(CoorBeeMaxH));
                        elsif (b1 = '1' or getBullet = '1') and mainPic = '0' then
                                colorSignal := "010";
                        elsif BeeMaxG = '1' and mainPic = '0' then
                                if windFlip = '1' then
                                tmpSignal := bee(
                                to_integer(Vcount) - VSYNC - VBACK_PORCH + 1 -
to_integer(CoorBeeMaxV),
```

```
                                to_integer(Hcount) - HSYNC - HBACK_PORCH - 1 -
to_integer(CoorBeeMaxH));
                        else
                                tmpSignal := beef(
                                to_integer(Vcount) - VSYNC - VBACK_PORCH + 1 -
to_integer(CoorBeeMaxV),
                                to_integer(Hcount) - HSYNC - HBACK_PORCH - 1 -
to_integer(CoorBeeMaxH));
                        end if;
                        if bee_RedG = '1' or bee_GreenG = '1' or bee_PurpleG = '1' then
                                if tmpSignal = "10" then
                                        colorSignal := "111";
                                elsif tmpSignal = "11" then
                                        colorSignal := "100";
                                elsif bee_RedG = '1' and tmpSignal = "01" then
                                        colorSignal := "101";
                                elsif bee_GreenG = '1' and tmpSignal = "01" then
                                        colorSignal := "110";
                                elsif bee_PurpleG = '1' and tmpSignal = "01" then
                                        colorSignal := "001";
                                elsif tmpSignal = "00" then
                                        colorSignal := "000";
                                end if;
                        else
                                colorSignal := "000";
                        end if;
                elsif starG = '1' then
                        colorSignal := shine;
                elsif starG1 = '1' then
                        colorSignal := shine1;
                elsif starG2 = '1' then
                        colorSignal := shine2;
                elsif starG3 = '1' then
                        colorSignal := shine3;
                else
                        colorSignal := "000";
                end if;

                if clearScr = '1' then
                        VGA_R <= "0000000000";
                        VGA_G <= "0000000000";
                        VGA_B <= "0000000000";

                elsif colorSignal = "001" then   -- purple
                        VGA_R <= "0101111011";
                        VGA_G <= "0011001111";
                        VGA_B <= "1111111111";

                elsif colorSignal = "011" then   -- red
                        VGA_R <= "1111111111";
                        VGA_G <= "0000000000";
                        VGA_B <= "0000000000";

                elsif colorSignal = "100" then   -- yellow
                        VGA_R <= "1111111111";
                        VGA_G <= "1111111111";
```

```
                VGA_B <= "0000000000";

            elsif colorSignal = "101" then   -- brown
                VGA_R <= "1100100000";
                VGA_G <= "0000011110";
                VGA_B <= "0000011110";

            elsif colorSignal = "110" then   -- green
                VGA_R <= "0000010101";
                VGA_G <= "1101111000";
                VGA_B <= "0000000111";

            elsif colorSignal = "111" then   -- light blue
                VGA_R <= "0000010000";
                VGA_G <= "1101010100";
                VGA_B <= "1101010111";

            elsif clearScr = '1' then
                VGA_R <= "0000000000";
                VGA_G <= "0000000000";
                VGA_B <= "0000000000";

            elsif colorSignal = "010" then   -- white
                VGA_R <= "1111111111";
                VGA_G <= "1111111111";
                VGA_B <= "1111111111";

            -------------------------this is background----------------------
            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;
```

6.2 lab3.vhd (top model file)

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

entity lab3 is

  port (
    signal CLOCK_50 : in std_logic;              -- 50 MHz clock
    signal LEDR : out std_logic_vector(17 downto 0); -- Red LEDs

          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]

          PS2_CLK,
    PS2_DAT         : in std_logic;

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

    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
    );

end lab3;

architecture rtl of lab3 is

  signal counter : unsigned(15 downto 0);
  signal reset_n : std_logic := '1';

  signal audio_request : std_logic;
  signal audio_clock_18 : std_logic;
  signal audio_counter : unsigned(31 downto 0);
  signal temp : std_logic_vector(31 downto 0);
-------------------------------------------------------
```

```
component audio_driver is
port(
        clock_50 : in std_logic;
        clock_18 : in std_logic;
        cpu_cmd : in std_logic_vector(31 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_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;

component audio_pll is
port (
        inclk0              : in std_logic  := '0';
        c0                          : out std_logic
);
end component;

begin

pll : audio_pll port map(
        inclk0 => CLOCK_50,
        c0 => audio_clock_18
);

AUD_XCK <= audio_clock_18;

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


v1: audio_driver port map (
        clock_50 =>CLOCK_50,
        clock_18 => audio_clock_18,
        cpu_cmd => temp,

        --Audio interface signals
        AUD_ADCLRCK  => AUD_ADCLRCK,
        AUD_ADCDAT   => AUD_ADCDAT,
```

```
        AUD_DACLRCK  => AUD_DACLRCK,
        AUD_DACDAT   => AUD_DACDAT,
        AUD_BCLK     => AUD_BCLK
);

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;

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

    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,

        VGA_BLANK_from_the_vga              => VGA_BLANK,
        VGA_B_from_the_vga                  => VGA_B,
        VGA_CLK_from_the_vga                => VGA_CLK,
    VGA_G_from_the_vga                      => VGA_G,
        VGA_HS_from_the_vga                 => VGA_HS,
        VGA_R_from_the_vga                  => VGA_R,
        VGA_SYNC_from_the_vga               => VGA_SYNC,
        VGA_VS_from_the_vga                 => VGA_VS,

        PS2_Clk_to_the_ps2                  => PS2_CLK,
    PS2_Data_to_the_ps2                 => PS2_DAT,
        data_from_the_audio             => temp
);

end rtl;
```

6.3 de2_wm8731_audio.vhd

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

entity de2_wm8731_audio is
port (
   clk : in std_logic;       --  Audio CODEC Chip Clock AUD_XCK (18.43 MHz)
   reset_n : in std_logic;
   clk_50 : in std_logic;
   disable : in std_logic; --when '1', no output from wm8731
   sound : in std_logic_vector(3 downto 0); -- select which sound will be played
   sound_finish1 : out std_logic;-- exp
   sound_finish2 : out std_logic;-- fire
   sound_finish3 : out std_logic;-- fall
   -- Audio interface signals

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

architecture rtl of de2_wm8731_audio is

   signal lrck : std_logic;
   signal bclk : std_logic;
   signal xck  : std_logic;
   signal lrck_divider : unsigned(7 downto 0);
   signal bclk_divider : unsigned(3 downto 0);
   signal set_bclk : std_logic;
   signal set_lrck : std_logic;
   signal clr_bclk : std_logic;
   signal lrck_lat : std_logic;
   signal shift_out : unsigned(15 downto 0);

   signal rom_data_bullet : unsigned(15 downto 0);-- from "bullet" rom to mux
   signal rom_data_explo : unsigned(15 downto 0); -- from "exploration" rom to mux
   signal rom_data_fall : unsigned(15 downto 0);-- from "fall" rom to mux
--signal rom_data_begin : unsigned(15 downto 0);
   signal mem_addr_bullet : unsigned(12 downto 0);
   signal mem_addr_explo : unsigned(12 downto 0);
signal mem_addr_fall : unsigned(13 downto 0);
--signal mem_addr_begin : unsigned(13 downto 0);
   signal counter1 : unsigned(2 downto 0);
   signal counter2 : unsigned(2 downto 0);
   signal counter3 : unsigned(3 downto 0);
--signal counter4 : unsigned(3 downto 0);
   signal data_from_mux : unsigned(15 downto 0);


   signal temp : std_logic; -- control the output from wm8731
```

```
component beebullet is
port
(
address		: IN STD_LOGIC_VECTOR (12 DOWNTO 0);
clock		: IN STD_LOGIC ;
q		: OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end component;

component beeexplo is
port
(
address		: IN STD_LOGIC_VECTOR (12 DOWNTO 0);
clock		: IN STD_LOGIC ;
q		: OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end component;

component beefall is
port
(
address		: IN STD_LOGIC_VECTOR (13 DOWNTO 0);
clock		: IN STD_LOGIC ;
q		: OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end component;

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_bullet : beebullet port map(
          address => std_logic_vector(mem_addr_bullet),
          clock => clk_50,
          unsigned(q) => rom_data_bullet
 );

audio_explo : beeexplo port map (
          address  => std_logic_vector(mem_addr_explo),
          clock => clk_50,
          unsigned(q) => rom_data_explo
);


audio_fall : beefall port map(
          address  => std_logic_vector(mem_addr_fall),
          clock => clk_50,
          unsigned(q) => rom_data_fall
);
```

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

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

process (clk)
begin
	if rising_edge(clk) 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 (clk)
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			shift_out <= (others => '0');
		elsif set_lrck = '1' then
			shift_out <= data_from_mux;
		elsif clr_bclk = '1' then
			shift_out <= shift_out (14 downto 0) & '0';
		end if;
		-- when disable = 1, no audio data output, which means mute.
		if disable = '1' then
			temp <= '0';
		else
			temp <= shift_out(15);
		end if;
	end if;
end process;

   -- Audio outputs

   AUD_ADCLRCK  <= lrck;
   AUD_DACLRCK  <= lrck;
   AUD_DACDAT   <= temp;
   AUD_BCLK     <= bclk;

   -- read data from ROM

   -- mux to select which sound to be played

data_from_mux <= rom_data_bullet  when sound = "0001" else
                                    rom_data_explo   when sound = "0010" else
                                    rom_data_fall    when sound = "0100" else
                                    x"0000";


   -- counter 1 for bullet
process(clk)
begin
	if rising_edge(clk) then
		if reset_n = '0' then
			mem_addr_bullet <= (others => '0');
			sound_finish1 <= '0';
			counter1<="000";
		elsif lrck_lat = '1' and lrck = '0' then
			if counter1 = "101" then
				counter1 <= "000";
				if mem_addr_bullet = x"0dff" then
					mem_addr_bullet <= (others => '0');
					sound_finish1 <='1';
				else
					mem_addr_bullet <= mem_addr_bullet + 1;
				end if;
			else
				counter1 <= counter1 + 1;
```

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

   -- counter 2 for explo
process(clk)
begin
        if rising_edge(clk) then
                if reset_n = '0' then
                        mem_addr_explo <= "0000000000000";
                        sound_finish2 <= '0';
                        counter2<="000";
                elsif lrck_lat = '1' and lrck = '0' then
                        if counter2 = "101" then
                                counter2 <= "000";
                                if mem_addr_explo = x"1049" then
                                        mem_addr_explo <= "0000000000000";
                                        sound_finish2 <='1';
                                else
                                        mem_addr_explo <= mem_addr_explo + 1;
                                end if;
                        else
                                counter2 <= counter2 + 1;
                        end if;
                end if;
        end if;
end process;

   -- counter 3 for fall
process(clk)
begin
        if rising_edge(clk) then
                if reset_n = '0' then
                        mem_addr_fall <= "00000000000000";
                        sound_finish3 <= '0';
                        counter3<="0000";
                elsif lrck_lat = '1' and lrck = '0' then
                        if counter3 = "1010" then
                                counter3 <= "0000";
                                if mem_addr_fall = x"2845" then
                                        mem_addr_fall <= "00000000000000";
                                        sound_finish3 <='1';
                                else
                                        mem_addr_fall <= mem_addr_fall + 1;
                                end if;
                        else
                                counter3 <= counter3 + 1;
                        end if;
                end if;
        end if;
end process;

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

process(clk)
```

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

end architecture;
```

6.4 audio_controller.vhd

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

entity audio_bus is

  port (
    clk        : in  std_logic;
    reset_n    : in  std_logic;
    write      : in  std_logic;
    chipselect : in  std_logic;
    writedata  : in  unsigned(15 downto 0);
    cpu_cmd : in std_logic_vector(31 downto 0)
    );
end audio_bus;

architecture rtl of audio_bus is

begin
  process (clk)
  begin
    if rising_edge(clk) then
      if reset_n = '0' then
        tone <= x"1000";
      else
        if chipselect = '1' then
                        if write = '1' then
                                tone <= writedata;
          end if;
        end if;
      end if;
    end if;
  end process;

end rtl;
```

6.5 audio_driver.vhd

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

entity audio_driver is
port(
```

```
        clock_50 : in std_logic;
        clock_18 : in std_logic;

        clk       : in  std_logic;
        reset_n1  : in  std_logic;
        read      : in  std_logic;
        write     : in  std_logic;
        chipselect : in  std_logic;
        address   : in  unsigned(4 downto 0);
        readdata  : out unsigned(31 downto 0);
        --writedata  : in  unsigned(31 downto 0);
        cpu_cmd : in std_logic_vector(31 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 audio_driver;


architecture behavior of audio_driver is

signal disable : std_logic; -- when '1' disable audio module
signal reset_n : std_logic; -- when '0' reset audio module
signal sound_sel : std_logic_vector(3 downto 0); -- when "0001", play "fire", when "0010",
play "explosion"
signal play_finish1 : std_logic;-- exp
signal play_finish2 : std_logic;-- fire
signal play_finish3 : std_logic;-- falling down
signal reset_sm : std_logic; -- when '1' reset state machine

component de2_wm8731_audio is
port (
   clk : in std_logic;       --  Audio CODEC Chip Clock AUD_XCK (18.43 MHz)
   reset_n : in std_logic;
   clk_50 : in std_logic;
   disable : in std_logic;
   sound : in std_logic_vector(3 downto 0); -- select which sound will be played
   sound_finish1 : out std_logic;-- exp
   sound_finish2 : out std_logic;-- fire
   sound_finish3 : out std_logic;-- fall
        --sound_finish4 : out std_logic;-- begin

   -- 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;
```

```
signal audio_request : std_logic;

type state is (s0, bullet, explode, falling);
signal fsm_state : state;

begin

reset_sm <= cpu_cmd(28);

-- state machine of player
process(clock_50,reset_sm, cpu_cmd, fsm_state)
begin
	if rising_edge(clock_50) then
		if(reset_sm = '1') then
			fsm_state <= s0;
		elsif fsm_state = s0 then
			if cpu_cmd = x"00000061" then
				fsm_state <= bullet;
			elsif cpu_cmd = x"00000062" then
				fsm_state <= explode;
			elsif cpu_cmd = x"00000063" then
				fsm_state <= falling;
			else
				fsm_state <= s0;
			end if;
		elsif fsm_state = bullet then
			if cpu_cmd = x"00000062" then
				fsm_state <= explode;
			elsif cpu_cmd = x"00000063" then
				fsm_state <= falling;
			elsif play_finish2 = '1' then
				fsm_state <= s0;
			else
				fsm_state <= bullet;
			end if;
		elsif fsm_state = explode then
			if cpu_cmd = x"00000061" then
				fsm_state <= bullet;
			elsif cpu_cmd = x"00000063" then
				fsm_state <= falling;
			elsif play_finish1 = '1' then
				fsm_state <= s0;
			else
				fsm_state <= explode;
			end if;
		elsif fsm_state = falling then
			if cpu_cmd = x"00000061" then
				fsm_state <= bullet;
			elsif cpu_cmd = x"00000062" then
				fsm_state <= explode;
			elsif play_finish3 = '1' then
				fsm_state <= s0;
			else
```

```
                                    fsm_state <= falling;
                                end if;
                            end if;
                    end if;
end process;

reset_n <= '0' when fsm_state = s0 else
                    '1' when fsm_state = bullet else
                    '1' when fsm_state = explode else
                    '1' when fsm_state = falling else
                    '1';

disable <= '1' when fsm_state = s0 else
            '0' when fsm_state = bullet else
                    '0' when fsm_state = explode else
                    '0' when fsm_state = falling else
                    '0';

sound_sel <= "0001" when fsm_state = bullet else
                                "0010" when fsm_state = explode else
                                "0100" when fsm_state = falling else
                                "0000";

--port map to the wm8731 module
audio: de2_wm8731_audio port map(

   clk => clock_18,
   reset_n => reset_n,
   clk_50 =>clock_50,
   disable => disable,
   sound => sound_sel,
   sound_finish1 => play_finish1,
   sound_finish2 => play_finish2,
   sound_finish3 => play_finish3,

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

end architecture;
```

6.6 galaxian.c

```
#include <io.h>
#include <system.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <math.h>
#include <string.h>
```

```
#define maxFlyingBeeNum 7
#define maxPlaneLife 3
#define maxBeeLife 36

int flags = 0;
volatile unsigned int data;

typedef struct {
   int flying;
   int angle;
   int flyingH;
   int flyingV;
   int row;
   int column;
   int flySide;
   int angleCount;
   int circleCount;
   int smoothCount;
   int flyCount;
   int flyCountToBe;
   int bulletLeftCount;
   int done;
   int k;
   int turn;
   int track;
   int type;
   int bulletLeft;
} bee;

typedef struct bullet{
   int h;
   int v;
   int k;
   int number;
   int beeBulletMoveDown;
   struct bullet* prevBullet;
   struct bullet* nextBullet;
} bullet;

static inline void resetFlyingBee (bee* thisBee)
{
   thisBee->flying = 0;
   thisBee->angle = 0;
   thisBee->flyingH = 600;
   thisBee->flyingV = 440;
   thisBee->row = -1;
   thisBee->column = -1;
   thisBee->flySide = 0;
   thisBee->angleCount = 0;
   thisBee->circleCount = 0;
   thisBee->smoothCount = 0;
   thisBee->flyCount = 0;
   thisBee->flyCountToBe = 0;
   thisBee->bulletLeftCount = 0;
   thisBee->done = 0;
```

```
    thisBee->k = -1;
    thisBee->turn = 0;
    thisBee->track = 0;
    thisBee->type = 0;
    thisBee->bulletLeft = 0;
}

static inline void showStart (int startPicV)
{
    flags = 1;
    data = (flags << 20) + (startPicV << 10) + 1;
    IOWR_32DIRECT(VGA_BASE, 16, data);
}

static inline void hideStart (int startPicV)
{
    flags = 1;
    data = (flags << 20) + (startPicV << 10);
    IOWR_32DIRECT(VGA_BASE, 16, data);
}

static inline void showInfo (int level, int life)
{
    flags = 2;
    data = (flags << 20) + (life << 3) + (level+1);
    IOWR_32DIRECT(VGA_BASE, 16, data);
}

static inline void clearScreen()
{
    flags = 3;
    data = (flags << 20);
    IOWR_32DIRECT(VGA_BASE, 16, data);
}

static inline void showReady()
{
    flags = 4;
    data = (flags << 20) + 1;
    IOWR_32DIRECT(VGA_BASE, 16, data);
}

static inline void hideReady()
{
    flags = 4;
    data = (flags << 20);
    IOWR_32DIRECT(VGA_BASE, 16, data);
}

static inline void showPause()
{
    flags = 5;
    data = (flags << 20) + 1;
    IOWR_32DIRECT(VGA_BASE, 16, data);
}
```

```
static inline void hidePause()
{
   flags = 5;
   data = (flags << 20);
   IOWR_32DIRECT(VGA_BASE, 16, data);
}

static inline void showGameOver ()
{
   int i, j = 0;
   int gameover[8][2] = {{100, 50}, {310, 50}, {120, 450}, {330, 450}, {150, 450}, {360, 50},
{170, 50}, {380, 450}};
   int path[8][2] = { {1, 2}, {-1, 2}, {1, -2}, {-1, -2}, {1, -2}, {-1, 2}, {1, 2}, {-1, -2}};

   flags = 6;
   while (j++ < 100) {
      IOWR_32DIRECT(VGA_BASE, 52, 0);
      for (i = 0; i < 8; i++) {
         gameover[i][0] += path[i][0];
         gameover[i][1] += path[i][1];
         data = ((flags + i) << 20) + (gameover[i][0] << 10) + gameover[i][1];
         IOWR_32DIRECT(VGA_BASE, 16, data);
      }
      usleep(10000);
      while (IORD_32DIRECT(VGA_BASE, 0) != 0x0F);
   }

   while (1) {
      if(IORD_8DIRECT(PS2_BASE, 0)) {
         if (IORD_8DIRECT(PS2_BASE, 4) == 0x3B) {
            break;
         }
      }
   }
   for (i = 0; i < 8; i++) {
      data = ((flags + i) << 20) + (600 << 10) + 400;
      IOWR_32DIRECT(VGA_BASE, 16, data);
   }

}

static inline void showPlane (int planeH, int planeV)
{
   data = (planeH << 10) + planeV;
   IOWR_32DIRECT(VGA_BASE, 28, data);
}

static inline void showBullet(int bulletH, int bulletV)
{
   data = (bulletH << 10) + bulletV;
   IOWR_32DIRECT(VGA_BASE, 24, data);
}

static inline void showBeeBullet(bullet* thisBullet)
{
   flags = thisBullet->number;
```

```
    data = (flags << 20) + (thisBullet->h << 10) + thisBullet->v;
    IOWR_32DIRECT(VGA_BASE, 40, data);
}

static inline void addBullet(int planeH, int planeV, int flyingH, int flyingV, int number, bullet**
head)
{
    bullet* newBullet = malloc(sizeof(bullet));
    if (newBullet == NULL)
        return;

    if (planeH < flyingH - 10)
        newBullet->k = -1;
    else if (planeH > flyingH + 10)
        newBullet->k = 1;
    else
        newBullet->k = 0;

    newBullet->h = flyingH;
    newBullet->v = flyingV;
    newBullet->number = number;
    newBullet->beeBulletMoveDown = 0;

    if (head == NULL) {
        newBullet->prevBullet = NULL;
        newBullet->nextBullet = NULL;
        *head = newBullet;
    } else {
        newBullet->prevBullet = NULL;
        newBullet->nextBullet = *head;
        (*head)->prevBullet = newBullet;
        *head = newBullet;
    }
}

static inline void delBullet(bullet* thisBullet)
{
    thisBullet->h = 600;
    thisBullet->v = 400;
    showBeeBullet(thisBullet);
    if (thisBullet->prevBullet != NULL)
        thisBullet->prevBullet->nextBullet = thisBullet->nextBullet;
    if (thisBullet->nextBullet != NULL)
        thisBullet->nextBullet->prevBullet = thisBullet->prevBullet;
    free(thisBullet);
    thisBullet = NULL;
}

static inline void clearBulletList(bullet* head)
{
    bullet* curr;
    while (head != NULL) {
        curr = head;
        head = curr->nextBullet;
        delBullet(curr);
    }
```

```
}

static inline void showBeeMax(int beeMaxH, int beeMaxV)
{
   data = (beeMaxH << 10) + beeMaxV;
   IOWR_32DIRECT(VGA_BASE, 32, data);
}

static inline void showAlive(int alive[])
{
   int i = 0;
   for (i = 0; i < 5; i++) {
      data = (1 << (i + 20)) + alive[i];
      IOWR_32DIRECT(VGA_BASE, 20, data);
   }
}

static inline void showFlyingBee(bee thisBee, int i)
{
   flags = (thisBee.angle << 7) + (thisBee.type << 3) + i;
   data = (flags << 20) + (thisBee.flyingH << 10) + thisBee.flyingV;
   IOWR_32DIRECT(VGA_BASE, 48, data);
}

static inline void showExplosion (int expH, int expV, int small)
{
   data = (small << 20) + (expH << 10) + expV;
   IOWR_32DIRECT(VGA_BASE, 8, data);
}

static inline void showPlaneExplosion (int expH, int expV, int small)
{
   data = (small << 20) + (expH << 10) + expV;
   IOWR_32DIRECT(VGA_BASE, 12, data);
}

static inline void showScore (int *hiScore, int score)
{
   int first, second, third, fourth, fifth;

   fifth = score / 10000;
   fourth = score / 1000 - fifth*10;
   third = score / 100 - fifth*100 - fourth*10;
   second = score / 10 - fifth*1000 - fourth*100 - third*10;
   first = score - fifth*10000 - fourth*1000 - third*100 - second*10;

   if (*hiScore < score) {
      *hiScore = score;
      data = (first << 16) + (second << 12) + (third << 8) + (fourth << 4) + fifth;
      IOWR_32DIRECT(VGA_BASE, 4, data);
   }

   data = (first << 16) + (second << 12) + (third << 8) + (fourth << 4) + fifth;
   IOWR_32DIRECT(VGA_BASE, 0, data);
}
```

```
static inline int chooseBeeToFly(bee* thisBee, int alive[])
{
   int i, j;
   int side = rand()%10;
   if (side < 5) {
      for (i = 0; i < 19; i+=2) {
         for (j = 0; j < 5; j++) {
            if ((alive[j] & (1 << (19 - i))) >> (19 - i) == 1) {
               thisBee->row = i;
               thisBee->column = j<<1;
               thisBee->flySide = 1;
               alive[j] ^= 1 << (19 -i);
               return 1;
            }
         }
      }
   } else {
      for (i = 18; i >= 0; i-=2) {
         for (j = 0; j < 5; j++) {
            if ((alive[j] & (1 << (19 - i))) >> (19 - i) == 1) {
               thisBee->row = i;
               thisBee->column = j<<1;
               thisBee->flySide = -1;
               alive[j] ^= 1 << (19 -i);
               return 1;
            }
         }
      }
   }

   return 0;
}

static inline int convertAngle (int original, int invert)
{
   int transAngle;
   float angle = original * 180 / 106;
   if (invert < 0)
      angle = 360 - angle;

   if ((angle >= 0 && angle < 10) || (angle >= 350 && angle < 360)) {
      transAngle = 0;                  // 0 0 0 0 = 0
   } else if (angle >= 10 && angle < 30) {
      transAngle = 4;                  // 0 1 0 0 = 4
   } else if (angle >= 30 && angle < 60) {
      transAngle = 8;                  // 1 0 0 0 = 8
   } else if (angle >= 60 && angle < 80) {
      transAngle = 12;                  // 1 1 0 0 = 12
   } else if (angle >= 80 && angle < 100) {
      transAngle = 1;                  // 0 0 0 1 = 1
   } else if (angle >= 100 && angle < 120) {
      transAngle = 14;                  // 1 1 1 0 = 14
   } else if (angle >= 120 && angle < 150) {
      transAngle = 10;                  // 1 0 1 0 = 10
   } else if (angle >= 150 && angle < 170) {
      transAngle = 6;                  // 0 1 1 0 = 6
```

```
    } else if (angle >= 170 && angle < 190) {
        transAngle = 2;                    // 0 0 1 0 = 2
    } else if (angle >= 190 && angle < 210) {
        transAngle = 7;                    // 0 1 1 1 = 7
    } else if (angle >= 210 && angle < 240) {
        transAngle = 11;                    // 1 0 1 1 = 11
    } else if (angle >= 240 && angle < 260) {
        transAngle = 15;                    // 1 1 1 1 = 15
    } else if (angle >= 260 && angle < 280) {
        transAngle = 3;                    // 0 0 1 1 = 3
    } else if (angle >= 280 && angle < 300) {
        transAngle = 13;                    // 1 1 0 1 = 13
    } else if (angle >= 300 && angle < 330) {
        transAngle = 9;                    // 1 0 0 1 = 9
    } else if (angle >= 330 && angle < 350) {
        transAngle = 5;                    // 0 1 0 1 = 5
    }

    return transAngle;
}

static inline void facePlane (bee* thisBee, int planeH, int planeV)
{
    if (planeH == thisBee->flyingH) {
        if (planeV >= thisBee->flyingV)
            thisBee->angle = 2;
        else
            thisBee->angle = 0;
        return;
    }

    float k = (float) (planeV - thisBee->flyingV) / (planeH - thisBee->flyingH);

    if (planeV >= thisBee->flyingV) {
        if (k >= 5.67128) {
            thisBee->angle = 2;
        } else if (k >= 1.73205) {
            thisBee->angle = 7;
        } else if (k >= 0.57735) {
            thisBee->angle = 11;
        } else if (k > 0) {
            thisBee->angle = 15;
        } else if (k == 0) {
            if (planeH < thisBee->flyingH)
                thisBee->angle = 1;
            else
                thisBee->angle = 3;
        } else if (k >= -0.01) {
            thisBee->angle = 1;
        } else if (k >= -0.57735) {
            thisBee->angle = 14;
        } else if (k >= -1.73205) {
            thisBee->angle = 10;
        } else if (k >= -5.67128) {
            thisBee->angle = 6;
        } else {
```

```
            thisBee->angle = 2;
        }
    }
    else {
        if (k >= 5.67128) {
            thisBee->angle = 0;
        } else if (k >= 1.73205) {
            thisBee->angle = 4;
        } else if (k >= 0.57735) {
            thisBee->angle = 8;
        } else if (k > 0) {
            thisBee->angle = 12;
        } else if (k == 0) {
            if (planeH < thisBee->flyingH)
                thisBee->angle = 1;
            else
                thisBee->angle = 3;
        } else if (k >= -0.01) {
            thisBee->angle = 3;
        } else if (k >= -0.57735) {
            thisBee->angle = 13;
        } else if (k >= -1.73205) {
            thisBee->angle = 9;
        } else if (k >= -5.67128) {
            thisBee->angle = 5;
        } else {
            thisBee->angle = 0;
        }
    }
}

int main() {

    int i;
    int fire = 0;
    int bulletAllowed = 1;
    int beeMaxDirection = 1;
    int beeMaxMoveWait = 0;
    int beeMaxMoveHold = 0;
    int breakcode = 0;
    int leftMove = 0;
    int rightMove = 0;
    int pause = 0;
    unsigned char code;

    int planeMoveCount = 0;
    int beeMaxMoveCount = 0;
    int bulletCount = 0;
    int beeBulletCount = 0;
    int explodeCount = 0;
    int outExplodeCount = 0;
    int planeExplodeCount = 0;
    int rebornCount = 0;
    int waitCount = 0;

    int bulletNum = 0;
```

```
int flyingBeeNum = 0;
int planeLife = maxPlaneLife;
int beeLife = maxBeeLife;

int bulletH;
int bulletV = 400;
int planeH = 267;
int planeV = 400;
int beeMaxH = 100;
int beeMaxV = 50;
int outExpH;
int outExpV;
int startPicV = 480;

const int BEEMAX_LONG = 304;
const int BEEMAX_HEIGHT = 144;

int level = 0;
int score = 0;
int hiScore = 5000;

bullet* head = NULL;
bee flyingBee[maxFlyingBeeNum];
int alive[5] = {8320, 43680, 174760, 699050, 699050};
const int initAlive[5] = {8320, 43680, 174760, 699050, 699050};
int scoreArray[5] = {60, 50, 40, 30, 30};
int backAngle[16] = {0, 0, 5, 4, 9, 8, 13, 12, 3, 1, 15, 14, 11, 10, 7, 6};
int beeBullet[8] = {1, 2, 2, 3, 3, 3, 4, 4};
int waitTime = 2000;
int beeLifeThreshold[8][4] = {{20, 10, 5, 1},
                              {22, 12, 6, 2},
                              {24, 14, 7, 3},
                              {26, 16, 8, 4},
                              {28, 18, 9, 5},
                              {30, 20, 10, 6},
                              {32, 22, 11, 7},
                              {34, 24, 12, 8}};

int waitTimeThreshold[8][5] = {{7600, 7100,  6600,  6100, 5600},
                               {7400, 6900,  6400,  5900, 5400},
                               {7200, 6700,  6200,  5700, 5200},
                               {7000, 6500,  6000,  5500, 5000},
                               {6800, 6300,  5800,  5300, 4800},
                               {6600, 6100,  5600,  5100, 4600},
                               {6400, 5900,  5400,  4900, 4400},
                               {6200, 5700,  5200,  4700, 4200}};

int circleChangeH[132] = { 0, 0, 0, 0, 0, 0,-1, 0, 0, 0,
                          -1, 0, 0,-1, 0, 0,-1, 0,-1,-1,
                           0,-1,-1,-1,-1,-1,-1,-1,-1,-1,
                          -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
                          -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
                          -1,-1,-1,
                          -1,-1,-1,
                          -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
                          -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
```

```
                        0,-1,-1,-1,-1,-1,-1,-1,-1,-1,
                       -1, 0, 0,-1, 0, 0,-1, 0,-1,-1,
                        0, 0, 0, 0, 0, 0,-1, 0, 0, 0,

                        0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
                        1, 1, 0, 1, 0, 0, 1, 0, 0, 1,
                        1, 1, 1, 1, 1, 1};

   int circleChangeV[132] = {-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
                       -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
                       -1,-1,-1,-1,-1,-1,-1,-1,-1, 0,
                       -1,-1, 0,-1,-1, 0,-1, 0, 0,-1,
                        0, 0,-1, 0, 0, 0,-1, 0, 0, 0,
                        0, 0, 0,
                        0, 0, 0,
                        0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
                        1, 1, 0, 1, 1, 0, 1, 0, 0, 1,
                        1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
                        1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                        1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

                        1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                        1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                        0, 1, 1, 1, 1, 1};

   int turnChangeH[20] = { 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
                           0, 0, 0,-1, 0, 0,-1,-1, 0,-1};

start:
   // clear any flying bee left on the screen
   for (i = 0; i < maxFlyingBeeNum; i++) {
      resetFlyingBee(&flyingBee[i]);
      showFlyingBee(flyingBee[i], i);
   }

   // clear gameover if left on the screen
   for (i = 0; i < 8; i++) {
      data = ((flags + i) << 20) + (600 << 10) + 400;
      IOWR_32DIRECT(VGA_BASE, 16, data);
   }
   hideReady();
   hidePause();
   showScore(&hiScore, score);
   showInfo(level, planeLife);

   /*-----game start screen-----*/
   for (i = 480; i > 100; i--) {
      startPicV = i;
      showStart(startPicV);
      usleep(10000);
      while(IORD_8DIRECT(PS2_BASE, 0) == 1) {
         code = IORD_8DIRECT(PS2_BASE, 4);
      }
   }

   while (1) {
```

```
    if(IORD_8DIRECT(PS2_BASE, 0)) {
        if (IORD_8DIRECT(PS2_BASE, 4) == 0x5A) {
            hideStart(startPicV);
            break;
        }
    }
}

while (1) {
    IOWR_32DIRECT(VGA_BASE, 52, 0);
    showInfo(level, planeLife);

    /*-----keyboard control plane coordinate-----*/
    if(IORD_8DIRECT(PS2_BASE, 0)) {

        code = IORD_8DIRECT(PS2_BASE, 4);

        if (breakcode == 1) {
            breakcode = 0;
            switch (code) {
            case 0x1C:
                leftMove = 0;
                break;
            case 0x23:
                rightMove = 0;
                break;
            default:
                break;
            }
        } else {
            switch (code) {
            case 0x29:
                if (fire == 0 && rebornCount == 0) {
                    IOWR_32DIRECT(AUDIO_BASE, 0, 0x00000061);
                    IOWR_32DIRECT(AUDIO_BASE, 0, 0x00000008);
                    fire = 1;
                    bulletH = planeH + 10;
                }
                break;
            case 0x1C:
                leftMove = -1;
                break;
            case 0x23:
                rightMove = 1;
                break;
            case 0x5A:
                pause = 1;
                leftMove = 0;
                rightMove = 0;
                showPause();
            case 0xF0:
                breakcode = 1;
            }
        }
    }
```

```
    while (pause == 1) {

      if(IORD_8DIRECT(PS2_BASE, 0)) {
        code = IORD_8DIRECT(PS2_BASE, 4);

        if (breakcode == 1) {
          breakcode = 0;
          if (code == 0x5A) {
            pause = 0;
            hidePause();
            break;
          }
        } else {
          if (code == 0xF0)
            breakcode = 1;
        }
      }
    }


    if (planeLife == 0 && planeExplodeCount == 0 && outExplodeCount == 0 &&
explodeCount == 0) {
      showGameOver();

      fire = 0;
      bulletV = 400;
      level = 0;
      score = 0;
      showScore(&hiScore, score);
      planeLife = maxPlaneLife;
      memcpy(alive, initAlive, sizeof(alive));
      beeLife = maxBeeLife;
      waitCount = 0;
      rebornCount = 0;

      clearBulletList(head);
      leftMove = 0;
      rightMove = 0;
      goto start;
    }

    if (rebornCount == 0 && planeLife > 0) {
      if (planeMoveCount == 10) {
        planeMoveCount = 0;
        if ((planeH == 50 && leftMove == -1) || (planeH == 480 && rightMove == 1)) {
        }
        else
          planeH += leftMove + rightMove;
      } else
        planeMoveCount++;

      showPlane(planeH, planeV);
      showBullet(bulletH, bulletV);

    } else {
      if (rebornCount == 1000 && planeLife > 0)
```

```
        showReady();
    rebornCount--;
    if (rebornCount == 0) {
        bulletAllowed = 1;
        hideReady();
    }
}

/*-----bullet coordinate-----*/
if (bulletCount == 1) {
    bulletCount = 0;

    if (fire == 1) {
        bulletV--;
        if (bulletV == 0) {
            bulletV = 400;
            fire = 0;
            showBullet(600, 400);
            if (rebornCount > 0)
                bulletAllowed = 0;
        }

    } else {
        bulletH = planeH + 10;
    }

    if (bulletAllowed == 1 && bulletV < 400)
        showBullet(bulletH, bulletV);

} else
    bulletCount++;

/*-----bee alive matrix-----*/

if (beeLife == 0 && explodeCount == 0 && outExplodeCount == 0 &&
    planeExplodeCount == 0 && fire == 0 && head == NULL) {

    if (level < 7)
        level++;

    beeLife = maxBeeLife;
    planeH = 267;
    memcpy(alive, initAlive, sizeof(alive));
    usleep(1000000);
    waitCount = 0;
    // clear plane moving direction
    leftMove = 0;
    rightMove = 0;
    while(IORD_8DIRECT(PS2_BASE, 0) == 1) {
        code = IORD_8DIRECT(PS2_BASE, 4);
    }

}

showAlive(alive);
```

```
/*-----bee matrix coordinate-----*/
if (beeMaxMoveCount == 20) {
   beeMaxMoveCount = 0;

   if (beeMaxMoveWait == 0) {
      beeMaxH += beeMaxDirection;
      if (beeMaxH <= 100)
         beeMaxDirection = 1;
      else if (beeMaxH >= 150)
         beeMaxDirection = -1;
      beeMaxMoveHold++;
      if (beeMaxMoveHold == 10)
         beeMaxMoveWait = 1;
      showBeeMax(beeMaxH, beeMaxV);
   } else {
      beeMaxMoveHold--;
      if (beeMaxMoveHold == 0)
         beeMaxMoveWait = 0;
   }

} else
   beeMaxMoveCount++;

/*-----bullet hits bee in matrix-----*/
if (bulletV >= beeMaxV && bulletV <= beeMaxV + BEEMAX_HEIGHT &&
   bulletH >= beeMaxH && bulletH <= beeMaxH + BEEMAX_LONG) {

   int col = (bulletV - beeMaxV) / 16;
   int row = (bulletH - beeMaxH) / 16;

   if (col%2 == 0 && (alive[col>>1] & (1 << (19 - row))) >> (19 - row) == 1) {
      alive[col>>1] &= ~ (1 << (19 - row));
      beeLife--;
      score += scoreArray[col>>1];
      showScore(&hiScore, score);
      fire = 0;
      bulletV = 400;
      outExpH = beeMaxH + 16*row;
      outExpV = beeMaxV + 16*col;
      showExplosion(outExpH, outExpV, 1);
      IOWR_32DIRECT(AUDIO_BASE, 0, 0x00000062);
      IOWR_32DIRECT(AUDIO_BASE, 0, 0x00000008);
      showBullet(600, 400);
      explodeCount = 1;
   }
}


/*----------explode count----------*/
if (explodeCount != 0) {
   if (explodeCount != 300) {
      explodeCount++;
      if (explodeCount == 150) {
         showExplosion(outExpH, outExpV, 0);
      }
   } else {
```

```
        explodeCount = 0;
        showExplosion(600, 400, 0);
    }
}

/*----------out explode count----------*/
if (outExplodeCount != 0) {
    if (outExplodeCount != 300) {
        outExplodeCount++;
        if (outExplodeCount == 150) {
            showExplosion(outExpH, outExpV, 0);
        }
    } else {
        outExplodeCount = 0;
        showExplosion(600, 400, 0);
    }
}

/*----------plane explode count----------*/
if (planeExplodeCount != 0) {
    if (planeExplodeCount != 300) {
        planeExplodeCount++;
        if (planeExplodeCount == 150) {
            showPlaneExplosion(planeH, planeV, 0);
        }
    } else {
        planeExplodeCount = 0;
        showPlaneExplosion(600, 400, 0);
        planeH = 267;
    }
}


/*-----flying bee coordinate-----*/
if (beeLife > beeLifeThreshold[level][0])
    waitTime = waitTimeThreshold[level][0];
else if (beeLife > beeLifeThreshold[level][1])
    waitTime = waitTimeThreshold[level][1];
else if (beeLife > beeLifeThreshold[level][2])
    waitTime = waitTimeThreshold[level][2];
else if (beeLife > beeLifeThreshold[level][3])
    waitTime = waitTimeThreshold[level][3];
else
    waitTime = waitTimeThreshold[level][4];

if (waitCount < waitTime)
    waitCount++;
else {
    if (flyingBeeNum < maxFlyingBeeNum){
        int chosenRow[2];
        int chosenCol[2];
        int chosenSide;
        int chosenNum = 0;
        for (i = 0; i < maxFlyingBeeNum; i++) {
            if (flyingBee[i].flying == 0) {
                if (chosenNum > 0) {
```

```
    chosenNum--;
    flyingBee[i].row = chosenRow[chosenNum];
    flyingBee[i].column = chosenCol[chosenNum];
    flyingBee[i].flySide = chosenSide;
    alive[flyingBee[i].column>>1] ^= 1 << (19 - flyingBee[i].row);
    flyingBeeNum++;
    flyingBee[i].flying = 1;
    flyingBee[i].flyingH = beeMaxH + 16*flyingBee[i].row;
    flyingBee[i].flyingV = beeMaxV + 16*flyingBee[i].column;
    flyingBee[i].bulletLeft = beeBullet[level];
    switch (flyingBee[i].column) {
       case 0:
       flyingBee[i].type = 3;
       flyingBee[i].flyCountToBe = 7;
       break;
       case 2:
       flyingBee[i].type = 2;
       flyingBee[i].flyCountToBe = 7;
       break;
       case 4:
       flyingBee[i].type = 1;
       flyingBee[i].flyCountToBe = 6;
       break;
       default:
       flyingBee[i].type = 0;
       flyingBee[i].flyCountToBe = 8;
       break;
    }
    waitCount = 0;
  } else if (chooseBeeToFly(&flyingBee[i], alive) == 1) {
    IOWR_32DIRECT(AUDIO_BASE, 0, 0x00000063);
    IOWR_32DIRECT(AUDIO_BASE, 0, 0x00000008);
    flyingBeeNum++;
    flyingBee[i].flying = 1;
    flyingBee[i].flyingH = beeMaxH + 16*flyingBee[i].row;
    flyingBee[i].flyingV = beeMaxV + 16*flyingBee[i].column;
    flyingBee[i].bulletLeft = beeBullet[level];

    switch (flyingBee[i].column) {
       case 0:
       flyingBee[i].type = 3;
       flyingBee[i].flyCountToBe = 7;
       break;
       case 2:
       flyingBee[i].type = 2;
       flyingBee[i].flyCountToBe = 7;
       break;
       case 4:
       flyingBee[i].type = 1;
       flyingBee[i].flyCountToBe = 6;
       break;
       default:
       flyingBee[i].type = 0;
       flyingBee[i].flyCountToBe = 8;
       break;
    }
```

```
                waitCount = 0;

                chosenSide = flyingBee[i].flySide;
                if (flyingBee[i].column == 0) {
                   if ((alive[1] & (1 << (19 - flyingBee[i].row))) >> (19 - flyingBee[i].row) == 1)
{
                      chosenRow[chosenNum] = flyingBee[i].row;
                      chosenCol[chosenNum] = 2;
                      chosenNum++;
                   }

                   if (flyingBee[i].row < 10) {

                      if ((alive[1] & (1 << (19 - flyingBee[i].row + 2))) >> (19 - flyingBee[i].row
+ 2) == 1) {
                         chosenRow[chosenNum] = flyingBee[i].row - 2;
                         chosenCol[chosenNum] = 2;
                         chosenNum++;
                      }
                      if (chosenNum == 2)
                         continue;
                      if ((alive[1] & (1 << (19 - flyingBee[i].row - 2))) >> (19 - flyingBee[i].row
- 2) == 1) {
                         chosenRow[chosenNum] = flyingBee[i].row + 2;
                         chosenCol[chosenNum] = 2;
                         chosenNum++;
                      }

                   } else {

                      if ((alive[1] & (1 << (19 - flyingBee[i].row - 2))) >> (19 - flyingBee[i].row
- 2) == 1) {
                         chosenRow[chosenNum] = flyingBee[i].row + 2;
                         chosenCol[chosenNum] = 2;
                         chosenNum++;
                      }
                      if (chosenNum == 2)
                         continue;
                      if ((alive[1] & (1 << (19 - flyingBee[i].row + 2))) >> (19 - flyingBee[i].row
+ 2) == 1) {
                         chosenRow[chosenNum] = flyingBee[i].row - 2;
                         chosenCol[chosenNum] = 2;
                         chosenNum++;
                      }
                   }

                } else if (flyingBee[i].column == 2) {

                   if (flyingBee[i].row < 10) {
                      if ((alive[0] & (1 << (19 - 6))) >> (19 - 6) == 1) {
                         chosenRow[chosenNum] = 6;
                         chosenCol[chosenNum] = 0;
                         chosenNum++;
                      } else {
                         break;
```

```
    }

    switch (flyingBee[i].row) {
      case 4:
        if ((alive[1] & (1 << (19 - 6))) >> (19 - 6) == 1) {
          chosenRow[chosenNum] = 6;
          chosenCol[chosenNum] = 2;
          chosenNum++;
        }
        if (chosenNum == 2)
          break;
        if ((alive[1] & (1 << (19 - 8))) >> (19 - 8) == 1) {
          chosenRow[chosenNum] = 8;
          chosenCol[chosenNum] = 2;
          chosenNum++;
        }
        break;
      case 8:
        if ((alive[1] & (1 << (19 - 6))) >> (19 - 6) == 1) {
          chosenRow[chosenNum] = 6;
          chosenCol[chosenNum] = 2;
          chosenNum++;
        }
        if (chosenNum == 2)
          break;
        if ((alive[1] & (1 << (19 - 4))) >> (19 - 4) == 1) {
          chosenRow[chosenNum] = 4;
          chosenCol[chosenNum] = 2;
          chosenNum++;
        }
        break;
      default:
        break;
    }

  } else {
    if ((alive[0] & (1 << (19 - 12))) >> (19 - 12) == 1) {
      chosenRow[chosenNum] = 12;
      chosenCol[chosenNum] = 0;
      chosenNum++;
    } else {
      break;
    }

    switch (flyingBee[i].row) {
      case 14:
        if ((alive[1] & (1 << (19 - 12))) >> (19 - 12) == 1) {
          chosenRow[chosenNum] = 12;
          chosenCol[chosenNum] = 2;
          chosenNum++;
        }
        if (chosenNum == 2)
          break;
        if ((alive[1] & (1 << (19 - 10))) >> (19 - 10) == 1) {
          chosenRow[chosenNum] = 10;
          chosenCol[chosenNum] = 2;
```

```
                        chosenNum++;
                    }
                    break;
                case 10:
                    if ((alive[1] & (1 << (19 - 12))) >> (19 - 12) == 1) {
                        chosenRow[chosenNum] = 12;
                        chosenCol[chosenNum] = 2;
                        chosenNum++;
                    }
                    if (chosenNum == 2)
                        break;
                    if ((alive[1] & (1 << (19 - 14))) >> (19 - 14) == 1) {
                        chosenRow[chosenNum] = 14;
                        chosenCol[chosenNum] = 2;
                        chosenNum++;
                    }
                    break;
                default:
                    break;
                }
            }
        }
    }

    if (chosenNum == 0)
        break;
    }
  }
}

if (flyingBeeNum > 0) {
  for (i = 0; i < maxFlyingBeeNum; i++) {
    if (flyingBee[i].flying == 1) {
      if (flyingBee[i].angleCount < 132) {
        if (flyingBee[i].circleCount == flyingBee[i].flyCountToBe) {

          flyingBee[i].circleCount = 0;
          flyingBee[i].flyingH += flyingBee[i].flySide *
circleChangeH[flyingBee[i].angleCount];
          flyingBee[i].flyingV += circleChangeV[flyingBee[i].angleCount];
          flyingBee[i].angle = convertAngle(flyingBee[i].angleCount,
flyingBee[i].flySide);
          flyingBee[i].angleCount++;

          if (flyingBee[i].angleCount == 106 &&
            ((flyingBee[i].flySide > 0 && flyingBee[i].flyingH > planeH) ||
             (flyingBee[i].flySide < 0 && flyingBee[i].flyingH < planeH)))
            flyingBee[i].angleCount = 132;

          if (flyingBee[i].angleCount == 132) {
            flyingBee[i].k = flyingBee[i].flySide;
            flyingBee[i].track = flyingBee[i].flyingV;
          }
          showFlyingBee(flyingBee[i], i);
```

```
            } else
               flyingBee[i].circleCount++;

         } else {

            if (flyingBee[i].flyCount == flyingBee[i].flyCountToBe) {
               flyingBee[i].flyCount = 0;

               switch (flyingBee[i].done) {
                  case 1:
                  flyingBee[i].flyingH = beeMaxH + 16*flyingBee[i].row;
                  if (beeMaxV + 16*flyingBee[i].column - flyingBee[i].flyingV > 0 &&
                     beeMaxV + 16*flyingBee[i].column - flyingBee[i].flyingV < 33 &&
                     (beeMaxV + 16*flyingBee[i].column - flyingBee[i].flyingV)%4 == 0) {

                     flyingBee[i].angle = backAngle[((beeMaxV + 16*flyingBee[i].column -
flyingBee[i].flyingV) >> 1) - flyingBee[i].row/10 - 1];

                  } else if (flyingBee[i].flyingV == beeMaxV + 16*flyingBee[i].column) {

                     flyingBeeNum--;
                     alive[flyingBee[i].column>>1] ^= 1 << (19 -flyingBee[i].row);
                     resetFlyingBee(&flyingBee[i]);
                  }
                  break;

                  case 0:
                  if (rebornCount == 0)
                     facePlane(&flyingBee[i], planeH, planeV);

                  if (flyingBee[i].flyingV == flyingBee[i].track) {
                     if ((flyingBee[i].flyingH+8) - (planeH+10) < 0) {
                        if (flyingBee[i].k != 1) {
                           flyingBee[i].k = 1;
                           flyingBee[i].turn = 1;
                        }
                     } else {
                        if (flyingBee[i].k != -1) {
                           flyingBee[i].k = -1;
                           flyingBee[i].turn = -1;
                        }
                     }

                     if (flyingBee[i].flyingH >= 20 && flyingBee[i].flyingH <= 510) {
                        addBullet(planeH+10, planeV+10, flyingBee[i].flyingH+8,
flyingBee[i].flyingV+8, bulletNum++, &head);
                        flyingBee[i].bulletLeft--;
                     }
                  }

                  // smoothly turn
                  if (flyingBee[i].turn != 0) {
                     if (flyingBee[i].smoothCount < 20) {
                        flyingBee[i].flyingH -=
flyingBee[i].turn*turnChangeH[flyingBee[i].smoothCount++];
                     } else if (flyingBee[i].smoothCount == 20) {
```

```
                        flyingBee[i].smoothCount = 0;
                        flyingBee[i].turn = 0;
                        flyingBee[i].flyingH += flyingBee[i].k;
                    }
                } else {
                    flyingBee[i].flyingH += flyingBee[i].k;

                    // change direction before flying out of the screen
                    if (flyingBee[i].flyingV < 420) {
                        if (flyingBee[i].flyingH > 500 && flyingBee[i].k == 1) {
                            flyingBee[i].k = -1;
                            flyingBee[i].turn = -1;
                        } else if (flyingBee[i].flyingH < 20 && flyingBee[i].k == -1) {
                            flyingBee[i].k = 1;
                            flyingBee[i].turn = 1;
                        }
                    }

                    if (flyingBee[i].flyingV == beeMaxV + 16*flyingBee[i].column + 150) {
                        if (rebornCount == 0) {

                            if ((flyingBee[i].flyingH+8) - (planeH+10) < 0) {
                                if (flyingBee[i].k != 1) {
                                    flyingBee[i].k = 1;
                                    flyingBee[i].turn = 1;
                                }
                            } else {
                                if (flyingBee[i].k != -1) {
                                    flyingBee[i].k = -1;
                                    flyingBee[i].turn = -1;
                                }
                            }
                            if (flyingBee[i].type == 1)
                                addBullet(planeH+10, planeV+10, flyingBee[i].flyingH+8,
flyingBee[i].flyingV+8, bulletNum++, &head);
                        }
                    }
                }

                break;

                default:
                break;
            }
            flyingBee[i].flyingV++;
            showFlyingBee(flyingBee[i], i);

            // fly out the screen
            if (flyingBee[i].flyingV == 470) {
                flyingBee[i].flyingV = 0;
                if (beeLife > level) {
                    flyingBee[i].angle = 2;
                    flyingBee[i].done = 1;
                } else {
                    flyingBee[i].bulletLeft = beeBullet[level];
                }
```

```
                }

                if (bulletNum >= 30)
                    bulletNum = 0;

                // shoot bullet to plane
                if (flyingBee[i].bulletLeft > 0 && flyingBee[i].bulletLeft < beeBullet[level]) {
                   if (flyingBee[i].bulletLeftCount == 30) {
                      flyingBee[i].bulletLeftCount = 0;
                      addBullet(planeH+10, planeV+10, flyingBee[i].flyingH+8,
flyingBee[i].flyingV+8, bulletNum++, &head);
                      flyingBee[i].bulletLeft--;
                   } else
                      flyingBee[i].bulletLeftCount++;
                }


                // flying bee hits the plane
                if (flyingBee[i].flyingV >= 388 && flyingBee[i].flyingV <= 416 &&
                   fabs(flyingBee[i].flyingH - planeH -2) <= 16 &&
                   rebornCount ==0 && planeLife > 0) {
                   showPlaneExplosion(planeH, planeV, 1);
                   planeExplodeCount = 1;
                   score += scoreArray[flyingBee[i].column>>1]>>1;
                   showScore(&hiScore, score);
                   resetFlyingBee(&flyingBee[i]);
                   showFlyingBee(flyingBee[i], i);
                   showPlane(600, 400);
                   showBullet(600, 400);
                   flyingBeeNum--;
                   beeLife--;
                   planeLife--;
                   rebornCount = 2000;
                   // clear plane moving direction
                   leftMove = 0;
                   rightMove = 0;
                   while(IORD_8DIRECT(PS2_BASE, 0) == 1) {
                      code = IORD_8DIRECT(PS2_BASE, 4);
                   }
                }

             }
             flyingBee[i].flyCount++;
          }

          // bullet hits flying bee
          if (bulletV <= flyingBee[i].flyingV+16 && bulletV >= flyingBee[i].flyingV &&
          bulletH <= flyingBee[i].flyingH+14 && bulletH >= flyingBee[i].flyingH+2 &&
             fire == 1) {
             outExpH = flyingBee[i].flyingH;
             outExpV = flyingBee[i].flyingV;
             IOWR_32DIRECT(AUDIO_BASE, 0, 0x00000062);
             IOWR_32DIRECT(AUDIO_BASE, 0, 0x00000008);
             showExplosion(outExpH, outExpV, 1);
             outExplodeCount = 1;
             score += scoreArray[flyingBee[i].column>>1]<<1;
```

```
                showScore(&hiScore, score);
                fire = 0;
                bulletV = 400;
                showBullet(600, 400);
                resetFlyingBee(&flyingBee[i]);
                showFlyingBee(flyingBee[i], i);
                flyingBeeNum--;
                beeLife--;
            }
        }
    }
}

/*-----bee bullet coordinate-----*/
if (beeBulletCount == 3) {
    beeBulletCount = 0;
    bullet* curr;
    curr = head;
    while (curr != NULL) {
        bullet* next;
        next = curr->nextBullet;

        // bee bullet flying out of screen
        if (curr->v >= 477 || curr->h >= 500 || curr->h <= 0) {
            if (head == curr)
                head = next;

            delBullet(curr);
            curr = next;
            continue;
        }

        // bee bullet hits the plane
        if (curr->v >= 400 && curr->v <= 420 &&
            curr->h - planeH <= 20 && curr->h >= planeH &&
            rebornCount == 0 && planeLife > 0) {
            showPlaneExplosion(planeH, planeV, 1);
            planeExplodeCount = 1;
            showPlane(600, 400);
            leftMove = 0;
            rightMove = 0;
            showBullet(600, 400);
            planeLife--;
            rebornCount = 2000;
            if (head == curr)
                head = next;

            delBullet(curr);
            curr = next;
            continue;
        }

        if (curr->beeBulletMoveDown == 1) {
            curr->h += curr->k;
            curr->beeBulletMoveDown = 0;
        } else {
```

```
                curr->beeBulletMoveDown = 1;
            }
            curr->v++;
            showBeeBullet(curr);
            curr = next;
          }
        } else
          beeBulletCount++;

        while (IORD_32DIRECT(VGA_BASE, 0) != 0x0F);

    }

    return 0;
}
```