All Invaders Are Belong To Us
CSEE 4840 Spring 2011
Project Report

Sanat Apte*, Mashooq Muhaimen†, Rahul Parikh* , Yibin Sun*
*Department of Electrical Engineering
†Department of Computer Science
School of Engineering and Applied Science, Columbia University in the City of New York
{ sa2832, mm3858, rtp2119, ys2522}@columbia.edu

Abstract

What follows is the design and implementation of an embedded system that mimics the classic arcade game *Space Invaders*. This project utilizes both hardware and software capabilities of the Altera DE2 board. The implementation involves a combination of C and VHDL. We use a PS/2 keyboard, a VGA monitor and a Wolfson WM8371 audio codec as peripheral devices.

1 Game Overview:

All Invaders Are Belong to Us! is a two-dimensional fixed shooter game in which the player controls a home-ship by moving it horizontally across the bottom of the screen and firing at descending aliens. The aim is to defeat five rows of twelve aliens that move horizontally back and forth across the screen as they advance towards the bottom of the screen. The player defeats an alien, and earns points, by shooting it with laser missiles. Defeating the aliens brings another wave that is more difficult, a loop which can continue indefinitely.

The aliens attempt to destroy the home-ship by shooting at it while they approach the bottom of the screen. If they reach the bottom, the alien invasion is successful and the game ends. The home-ship is partially protected by four stationary defense barricades that can be destroyed by the aliens and the home-ship.

Our rendition of the game is divided into 3 phases:The Game Start Phase: This phase contains the introductory screen which asks the player whether he/she is ready to “Save the Planet”.

The Game Play Phase: This is the main phase of the game. The player plays the game by shooting at the alien waves and hence sequentially clearing levels. Each player begins with 3 lives. He/ She can lose the game in two ways: if he/she gets shot by alien missiles 3 times, or if the alien wave proceeds so far down that they reach the barricade level (this indicates a successful alien invasion!).

The Game Over Phase: On losing the game, the game-state is shifted to the game-over screen. The user can restart the game by pressing the 'enter' key and resume saving the planet.

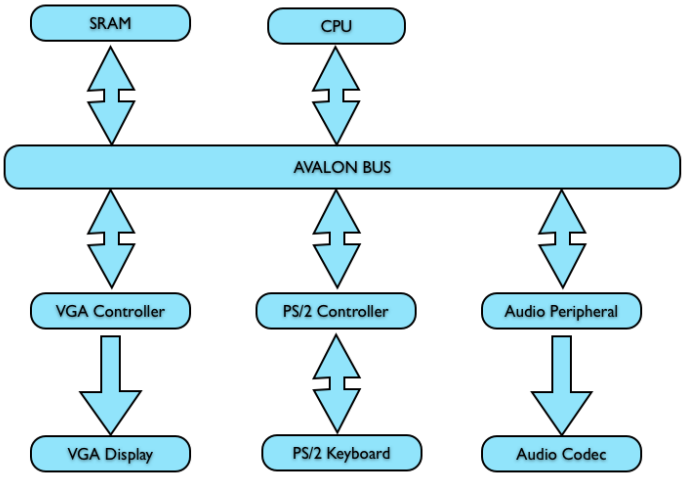


Figure 1. System Components

2 System Overview

Figure 1 gives an overview of the hardware components of the system. The system is implemented on an Altera DE2 board. We use the NIOSII processor as our CPU. The CPU uses the on-board SRAM as its memory device. We use a PS/2 keyboard as our game controller and the on-board Wolfson WM8371 audio codec to play game sound. A VGA monitor is used as the display. The CPU is setup as an Avalon master. It communicates to other hardware components of the system through the Avalon bus. The system contains the following Avalon slave peripherals: an SRAM controller, a PS/2 controller, an audio peripheral and a VGA controller. The SRAM controller facilitates communication between the CPU and the SRAM by translating the protocol spoken by the Avalon bus to that for the SRAM [1]. The PS/2 controller reads data from the PS/2 keyboard and passes the information to the software. The audio controller talks to the audio codec and tells it what frequency to produce and controls the on/off state interfaces with audio codec. The VGA controller, in addition to interfacing with the VGA display, contains the bulk of hardware logic for displaying game sprites.

Majority of the game flow logic (i.e. collision detection, keeping track of score, level changes etc.) are in the software. The software sends periodic updates to the hardware to change the video display and play sound. Two events cause the software to change game state: a time tick (implemented using a software counter delay) and keyboard inputs coming from the PS/2 controller.

3 Hardware Description

Description of the main hardware components that we worked on follows:

3.1 VGA Raster (includes VGA controller logic)

Most of the VGA Raster modules listed below operate at approximately 25 MHz, which is half the frequency of our main system clock. An exception is the module that listens for SW updates coming through Avalon.

3.1.1 Core VGA Raster Modules

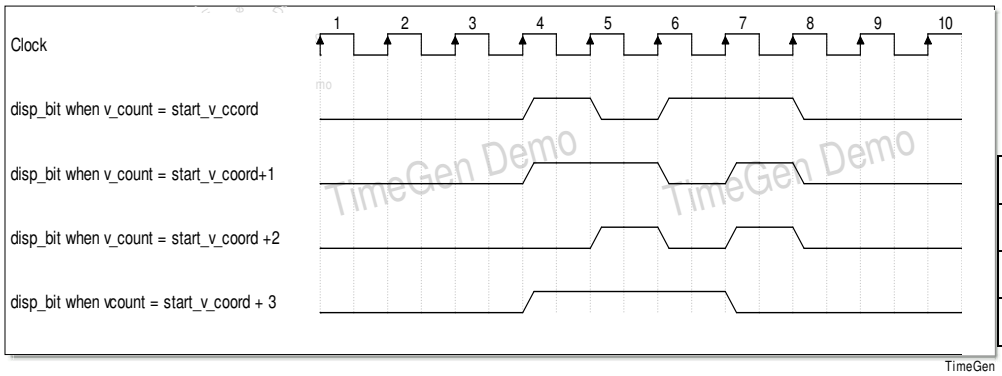
The heart of the video display hardware is a raster scanner [2]. Fortunately, we had access to a fully functional raster scanner that was provided to us as part of one of our labs. We used it almost without modification.

3.1.2 Sprite Display Modules

We use several sprite modules to display the sprite images (e.g. home ship sprite, alien sprites etc.) Each sprite module has a corresponding display bit. The role of each sprite module is to decide whether or not the display bit should be set to 1. The hardware finds the starting location of the sprite image in signal variables that are updated through the software (we will talk about the updating mechanism shortly). Figure 2 shows how a sprite drawing module's display bit is set using a small 4x4 sprite example. This display bit is decoded into a color (depending on what sprite module the bit belongs to) and the raster sets the VGA pixel to that color. It is possible however for a display bit for a particular sprite display module to be overridden by the display bit by a higher priority sprite module. Table 1 shows the priorities of different sprite modules.

Priority in descending order	***Sprite***
Home_bit	Home Ship
Wave_bit	The aliens and their phalanx
Home_missile	Home missile
Aliemissile_bits	Aliens' missiles
Score_bit	Score number
Intro_bit	Welcoming screen
Game_over_bit	Game over screen
Lives_bit_1 and 2 and 3	Display life image
Barricade1_bit 2 (x4)	Barricade sprite
Star#_bit (x18)	Blinking stars
Vga_hblank; vga_vblank; other	Blank

Table 1. Priorities (in order of decreasing priority) for various Sprites.



1	0	1	1
1	1	0	1
0	1	0	1
1	1	1	0

Figure 2. Example of how a 4x4 sprite image is drawn. The hardware module responsible for drawing the sprite starts setting a display bit when the raster scanner's horizontal position matches the starting horizontal and vertical position (v_count) designated to the sprite (in this figure, the starting horizontal coordinate of the sprite is set to the 4th rising clock edge) . At each rising clock edge, the hardware sets the display bit to a sprite bit and increments an index so that in the next rising edge it looks at a bit one position to the right. It does this until it exhausts a sprite row. Then the display module waits for the raster to return to the starting horizontal position again (this will happen when v_count = start_v_coord + 1) before it starts reading the second sprite row.

A brief discussion of the main sprite modules follow. We do not discuss Scores/levels and lives modules as they are too similar to other modules discussed.

Figure 3. VGA Display during game play

3.1.2.1 Home ship module:

The home ship module uses the method described in Figure 2 to display a 32x32 sprite image. The starting horizontal and vertical coordinate of the home ship is updated by the software.

3.1.2.2 Alien Wave Module:

We draw the alien wave (see Figure 3) by repeatedly drawing the same 16x16 alien sprite. Even though we draw up to 5x12 = 60 aliens, we just have software update one set of horizontal and vertical starting coordinate (the starting horizontal and vertical coordinate of the top left-most alien in the alien wave). We can do this because each alien's relative distance to the top left-most alien position always remains the same. Given the alien id (row and column index in the 5x12 matrix), it is straightforward to calculate the starting x and y coordinate of an alien :

For the alien in the ith row and jth column:

the starting x coordinate of the alien = j * (alien_sprite_width + horizontal gap between aliens)

the starting y coordinate of the alien = i * (alien_sprite_length + vertical gap between aliens)

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	0	1	1	0	1	1	1
0	0	1	0	1	0	0	0	0	0	1	1

Figure 4. An alien wave display and the corresponding bitmap containing alive/kill status for each of the aliens where alive=1, kill = 0. The hardware does not display an alien sprite if its corresponding bit in the bitmap is set to 0. The bitmap is updated by the software every time an alien is killed. It is initialized to all 1's at the beginning of each level.

The hardware maintains a bitmap that contains the current status (1=alive, 0=dead) of each alien. This bitmap is updated by software whenever an alien is killed. The hardware does not display an alien if its status is set to "killed". This situation is illustrated in Figure 4.

3.1.2.3 Barricade Module

The barricade module (shown in Figure 3) is displayed using the method described in Figure 2, with one twist. The hardware keeps a 16x4 bitmap for each barricade. The barricade sprite can be seen as composed of 64 2x8 smaller sprites images/blocks. Each bit in the bitmap contains intact/destroyed (1=intact, 0=destroyed) status for each such block. As the raster scanner scans the barricade pixels, the barricade module figures out the what block the current pixel belongs to and checks the intact/destroyed status for that block. The block is not displayed if its status is "destroyed".

This module actually has a good deal of similarity with the alien wave module. Just think of 2x8 barricade block as an alien sprite with the gap between the sprites set to 0. The remaining logic is exactly as it is in the alien wave module.

3.1.2.4 Home and Alien Missile Display

The home missile display module is straightforward. We keep a couple of signal variables in the hardware that indicate starting horizontal and vertical coordinates for the home missile. We use these coordinates and draw the home missile using the method described in Figure 2. The coordinates are updated by the software to make the missile move. Note that we allow only one home missile at a time (this is consistent with the original *Space Invaders*). So just keeping one set of starting coordinates is enough. A Starting y coordinate of 0 indicates there is no active home missile.

The alien missile display is slightly more complicated. We allow a maximum of 10 missiles to be simultaneously displayed on the screen. As such, there are 10 sets of starting coordinates that the software updates. The software sets the coordinates of a missile to 0 if it doesn't want the hardware to display the missile.

3.1.2.5 Software Listener Module

Given that the CPU operates at 50 MHz, it is necessary for the listener module to operate at 50 MHz too. This sets this module apart from most other modules in the VGA controller, which all operate at close to 25 MHz.

As has been alluded to in the previous sections, the software updates different state vectors in the hardware to change the video display. These state vectors include the location of the home ship, the starting location of the alien wave, the alien alive/killed bitmap etc. Software updates these state vectors through the Avalon bus.

The Listener module receives information through the Avalon interface, decodes the instructions and sets the appropriate state variables. The decoding is done according to the memory map described in Section 4.

We now describe the software updating mechanism in a little more detail. To be precise, the software does not directly update majority of the state vectors that the hardware display modules use. It updates temporary versions of these state variables. The state variable versions the hardware looks at are updated when the VGA raster is in the vertical blanking region. The justification for this is that the sprite modules make the assumption that the starting coordinates of a sprite image do not change when the module is in the middle of drawing the image. No sprites are drawn when the raster is in the vertical blanking region so it is safe to update the state vectors. Figure 6 illustrates one possible scenario that can happen if we update the starting coordinate for a sprite while the sprite is being drawn. To be clear, the

problem shown in Figure 6 is due to the way the VHDL logic is written, but not having to worry about this type of problem simplifies the logic.

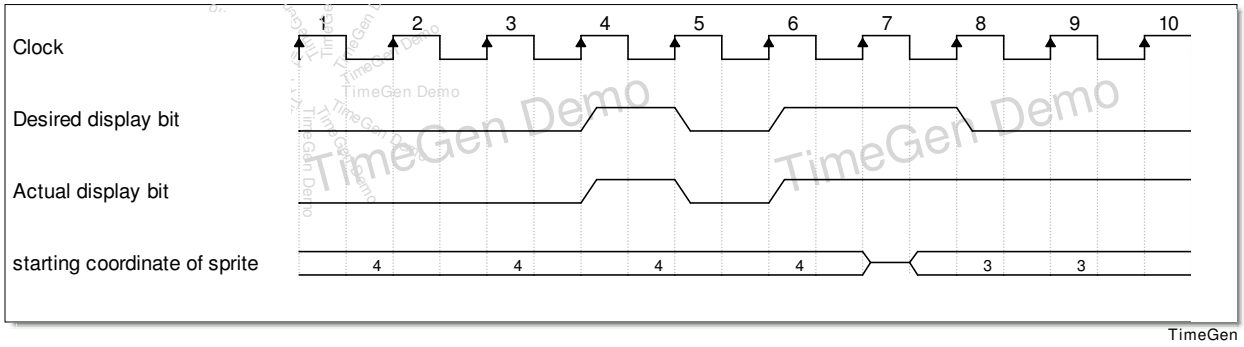


Figure 6. Illustration of a possible scenario when starting coordinate of a 1D sprite image ([1011]) is changed. Here, when hardware starts drawing the sprite, the starting coordinate is set to 4. It is desired that hardware sets the display bit signal to low at the rising edge of clock cycle 8 (starting coordinate of 4 + sprite width of 4). However, at the rising edge of clock cycle 7, the starting coordinate is changed by the software to 3. When the hardware reaches clock cycle 8, it is no longer the case that (starting coordinate + sprite width) = 8, and HW never sets the bit to low. During testing, a variant of this problem caused our video display to show momentary flickers on the screen. While the flickering was rare and momentary, it was undesirable and was noticeable to the keen observer.

3.1.2.6 Star/Background module

As the background , we displayed a number of stars which “twinkle”. Each star is a 16 x 16 sprite and is drawn using an individual process. The “twinkle” effect is produced with the help of the counters which count up to different values for each star process. This counter assists in toggling between the two separate sprite images for stars, which gives us the “twinkle”.

3.1.2.7 Intro and Game Over Modules

As discussed before, the game can be in one of three states: the “intro” state, the “playing” state and the “game over” state. When in the “intro” state, the intro text bitmap and the stars are displayed. For the “playing” state the alien wave, the home ship, the barricades, the score, lives etc are displayed and for the “game over” state the stars and the game ending text are displayed. Figure 5 shows the Intro and Game Over screens.

3.1.2.8 Animation (incorporated into other modules)

There are three main animations. They are the blinking stars, animated alien and the explosion of alien. The animations can be categorized into two types depending on their behavior. First is the repeating type. Both blinking star and animated alien belong to it. The animated alien consists of two

sprites of different postures. The sprite switches between two images every 25M clock cycles (or one second). The blinking stars adopt the same concept and different switching times gives the impression of outer space. The explosion of alien is a little different, it only occurs once and there is no switching back and forth. So the timing is harder to determine. The animation starts when an alien is announced to be killed. The animation lasts for 1250 counts. The trick is that the hardware only checks the states of an alien when displaying it. Hence, the count is updated only when the VGA raster scans to the exploding alien's position. Therefore, the time for the explosion is not following the 25MHz clock directly but it is incremented for 16x16 times per frame. The animation ends after the pre-defined duration for explosion expires.

3.2 PS2 Keyboard

The PS2 keyboard implementation borrows heavily from an existing implementation by Prof. Stephen Edwards and Yingjiang Gu. The implementation is for receiving keyboard data only. The least significant bit is the status bit that indicates the arrival of a scan code. The least significant byte is the data. Reading the data clears it from the input register. The timing diagram is shown in Figure 7..

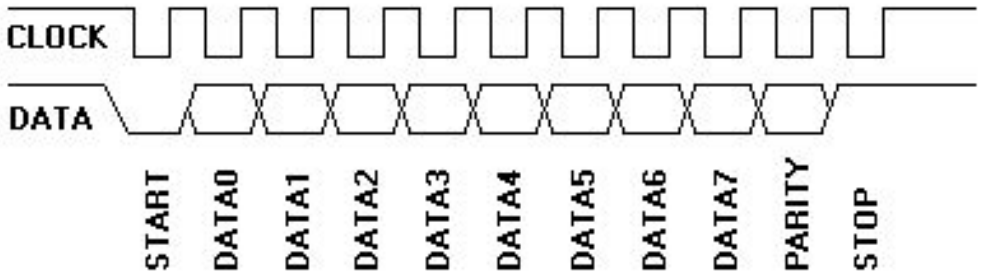


Figure 7. PS2 Keyboard Timing Diagram

3.3 Audio

The game produces two different sounds: a sound when an alien dies and a sound when there is a hit on the barricade. Although there was room for more, we decided to limit the sound to just two types as the game was becoming too noisy otherwise. For this implementation we used a simplified version of the lab 3 audio part. The audio peripheral receives signals from the software which tells it when to turn the sound on or off, and a signal which tells it what kind of sound to produce, this communicates with the audio codec. The codec contains a look up table for a pure sine tone, the signal sent by the peripheral basically decides how fast the sine wave is traversed. i.e. if a small value of counter is given, the counter expires quickly and is reset, meaning the sine wave is traversed at a high speed, a high frequency sound. Conversely, if we give a large counter value, a low frequency sound is produced. We use combinations of these sounds to produce sounds of our interest in the game.

4 Memory Map:

Memory mapped I/O was central to our project. It is how hardware and software communicated with each other. Table x gives the memory map of our system.

BASE ADDRESS	OFFSET	DESCRIPTION
VGA_BASE	0x0000	Home ship upper right coordinate
VGA_BASE	0x0001	Alien wave upper right coordinate (horizontal)
VGA_BASE	0x0002	Alien wave upper right coordinate (vertical)
VGA_BASE	0x0003 to 0x0007	Value of each five digits of the score
VGA_BASE	0x0008	Remaining Lives
VGA_BASE	0x0009	Home ship missile upper right coordinate (horizontal)
VGA_BASE	0x000A	Home ship missile upper right coordinate (vertical)
VGA_BASE	0x0010 to 0x0019	Alien missiles upper right coordinate (horizontal)
VGA_BASE	0x00A0 to 0x00A9	Alien missiles upper right coordinate (vertical)
VGA_BASE	0x00B0	Game state
VGA_BASE	0x8000 to 0x80311	Alien_statues_1 to Alien_statues_60
VGA_BASE	0xC000 to 0xC03F	Barricade 0, block 0 to 63
VGA_BASE	0xC040 to 0xC07F	Barricade 1, block 0 to 63
VGA_BASE	0xC080 to 0xC0BF	Barricade 2, block 0 to 63
VGA_BASE	0xC0B0 to 0xC0FF	Barricade 3, block 0 to 63
VGA_BASE	0xFFFF	Reset the state vectors
AUDIO_BASE	0x0000	Audio on off (4 MSB) and frequency (2 LSB)
PS2_BASE	0x0000	Control bit indicating incoming scan code
PS2_BASE	0x0004	Scan code to be read

Table 2. Memory Map of the System

5 Software Description

5.1 Software Architecture

Pseudo code for the SW Architecture is given in Figure 8. The software runs in an infinite loop, it reads inputs from keyboard and updates the keyboard state (this is described in detail in section 5.2).

However, the software does not immediately change the game state. It waits for a “time tick” that is implemented using a software counter variable. The reason for this is that otherwise SW changes the game state too frequently for the hardware to handle or the game to be enjoyable. Note that SW changes the game state at each time tick regardless of keyboard inputs. Among other things, the alien wave movement does not depend on player’s keyboard inputs.

```
while ( true )
  {
        read keyboard, update state
        counter = 0
        if ( counter == delay )
              respond to keyboard: move home ship/ spawn home missile

              collision detection for home missile and aliens
              collision detection for home missile and barricades
              collision detection for alien missiles and barricade
              send state updates to hardware

              if ( num_active_missiles < num_missiles_for_current level )
                    spawn new alien missile
              if all aliens killed
                    setup parameters to start new level
              if all 3 lives exhausted
                    transition to “game over” state
              counter = 0
        else
```

Figure 8. SW Architecture Pseudo Code

5.2 Intro and Game Over Logic

The game over and Intro logic are based on a state machine which checks for keyboard events, time outs and how the game progresses. This state machine is shown in Figure 9.

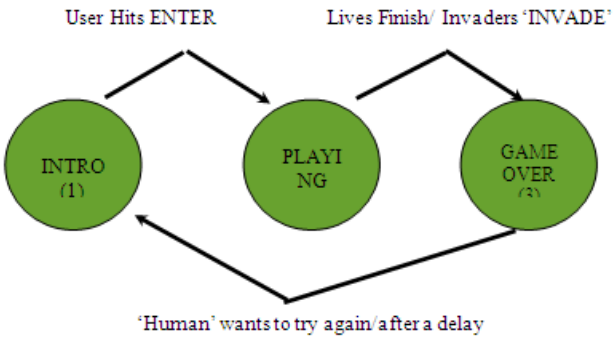


Figure 9. SW State Machine for game state

The game state is set to “INTRO” by default, when we start checking for keyboard inputs, we wait for the first time a user hits the ‘enter’ key this sets the game state to “PLAYING”, the and the game begins. The user goes about trying to stop the invaders, if the invaders take over by crossing the barricades or when the user runs out of lives, the game state changes to “game over state” and a message appears on the screen which tells the user he has failed in his mission to save the planet. But in our world the user can try again! The game resets. At each change of game state, the software sends the new state of the game to the hardware.

5.3 PS2 Keyboard

The software program associated with the PS/2 keyboard consists of two parts. One is the state machine implementation and another is the response instruction sets that correspond to individual states.

The state machine is shown in Figure 10. There are six states. Idle state is the initial state and as the name suggests, the home ship is at rest in this state. Left and Right state indicates the current movement of the home ship. Fire state means that the home ship is firing in stationary position. Left, Right and Fire states are used so that the rate of these actions of the home ship will not be constrained by the auto repeat feature of the PS/2 keyboard. The last two states are Left and Fire, and Right and Fire. These two states are created to support fire on the move. In another word, home ship does not require to quit movement in either direction to fires.

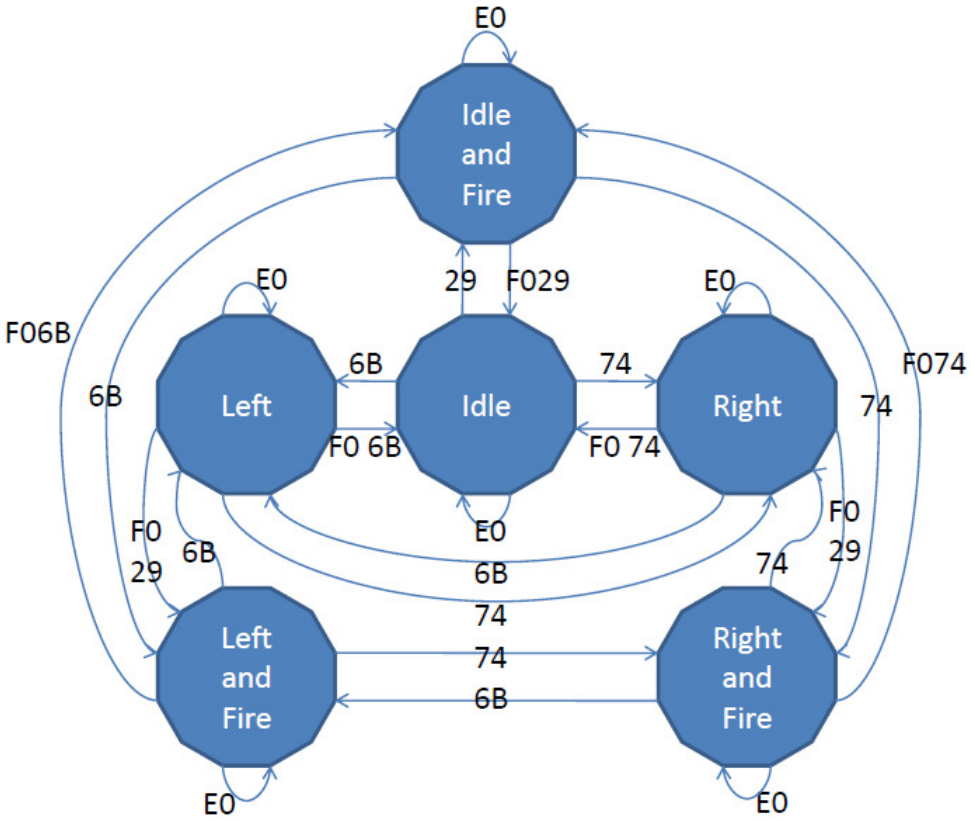


Figure 10. PS2 Keyboard State Machine

The change of state is triggered by the scan code read. For example, 0x29 will cause the state to change from Idle to Fire, once it is received. The corresponding break code will cause the state to change back to Idle. However, 0xF0 should not immediately trigger a change in the state. One reason is the dual actions taken by the home ship when it fires on the move. As a result, 0xF0 could mean stop movement or cease fire. To avoid the confusion, the following scan code will be important. If it is 0x29, then it will be cease fire and the state change will be switching to the corresponding Left or Right state. The second reason is that 0xF0 can come from irrelevant key inputs such as the down-arrow key between the right and left arrow keys. Accidental contact can send unneeded 0xF0 to the system.

5.4 Alien Missile Spawns

Each level has a set number of active missiles associated with it. At any instance, SW makes sure that exactly that number of active missiles is active. The number of active missiles is increased by one at each level until it reaches the maximum number of missiles the hardware supports:10 (more on this later).

When an alien missile perishes in a collision, a live alien is chosen in random, and a missile is spawned from that alien. Because the set of aliens to choose from is limited to the live aliens, the concentration of fire coming from each alien increases as more and more aliens are killed.

5.5 Home missile collision with aliens

The SW checks if the current position of the home missile has a collision with any of the alien (dead or alive) positions in the alien wave. It then checks if the alien that it has a collision with has already perished. If so, it lets the home missile move up. If there is a collision, then the SW sends update to the hardware indicating the home missile is not longer active. It also lets the HW know the id of the perished alien. SW then updates its own data structure for keeping track of the alive aliens. This data structure is slightly different than the structure that the HW maintains (shown in Figure x). The data structure that the SW maintains is shown in Figure 11.

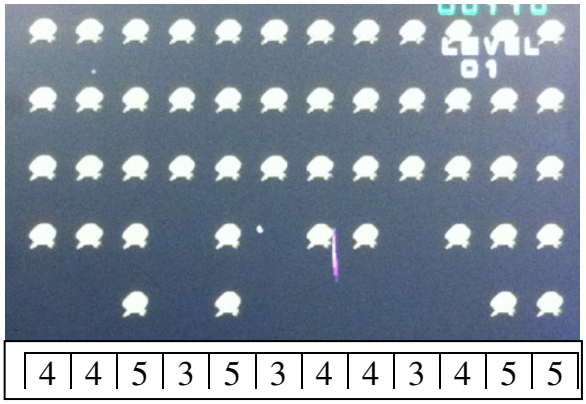


Figure 11. SW maintains a map containing the id of the bottom-most alien for each column. If SW detects a home-missile collision with an alien position dead or alive, it figures out the row and column index of the alien. SW then checks this SW map to figure out if the colliding alien is dead or alive. Note that collision can only happen with the bottom-most alien. This is why it is sufficient to just hold the id of the bottom most alien in each column.

5.6 Alien missile collision with home-ship

This is simply a matter of looping over all the active missile positions and calculating if any of them collide with position of the home ship.

5.7 Barricade collision

The barricade blocks can be destroyed from the top (by the alien) and the bottom (by the home-ship). It is not enough for the SW just to detect a barricade collision; it has to check if the colliding block has already been destructed. The SW keeps two maps to help it make its decision: an array of block id's of the top-most blocks that are still intact in each column, and an array of block ids of the bottom-most intact blocks in each column. If a collision is detected from the top, the top-most block in the

corresponding column is incremented. If a collision is detected from the bottom, the bottom-most block in the corresponding column is decremented. If the top-most and bottom-most block of a column matches, that column is declared destroyed by setting the corresponding entry in both arrays to 0. Any home and alien missile goes through a barricade column if its entry in the top or bottom array is set to 0.

5.8 Levels and scores:

When all the aliens in a wave perish, a new alien wave is ushered in. This is the beginning of a new level. The levels get increasingly more difficult. The starting level starts with 4 active missiles. For each level that follows, the number of active missiles is increased by 1, until we reach 10, which is the maximum number of missiles the hardware allows. Once we reach 10, we take down the number of missiles to 6, but increase the velocity at which the missiles approach. This process is repeated over and over until the velocity of the alien missiles equal that of the home-missile. At that point we take mercy on the player and don't increase the level difficulty. Given that none of the testers could get past 10 active missiles even at the lowest velocity, we are not too concerned about this limitation.

The scores are increased by 10 after shooting down an alien. The score is retained and further accumulated after the player successfully goes past a level. The score is reset to 0 if the player loses three lives.

5.9 Sound

The sound updates from the software are communicated to the hardware in order to decide whether to produce a sound or not and what sound to produce. In the software we check for different events that are occurring in the game and produce a sound accordingly. We introduce a delay between an 'on' signal and an 'off' signal for the sound to become discernible. The sounds produced are dependent on the value we write to the audio base register. A combination of sounds can be produced by following one on-off cycle with another.

6 Who Did What

Mashooq worked on the HW display of the alien wave, home ship, barricades, home and alien missiles. He also wrote the SW collision detection logic for the barricades, home ship and alien waves, and the home and alien missile spawning. He worked on the software architecture and the level setting logic (jointly with YiBin).

YiBin worked on the PS/2 controller, the alien and home ship hardware animation and creating the home ship sprite bitmap. For the software, he worked on the PS/2 keyboard input handler/state

machine, movement of the home ship and the alien wave, collision detection of alien missiles and the home-ship, and SW architecture and level setting logic.

Sanat handled all HW and SW logic for the audio peripheral. He also worked on both HW and SW portions of Intro and Game Over screens and the state transitions from “Intro” to “Playing” to “Game Over”. He drew the initial HW barricades and score display (including the sprite bitmaps).

Rahul worked on the VHDL modules and sprite images for the “twinkling”/animating background stars, levels, lives and scores. He was also responsible for updating these sprites from the software. Rahul is the creator of the alien sprite bitmaps used in the alien wave animation.

All four were heavily involved in testing the game and tweaking parameters for difficulty levels.

7 Lessons Learned and Advice for Future Projects

- Draw state machine and timing diagrams.
- Work thoroughly on the design, spending time on it early saves time later.
- The hardware compilation time can be huge. To speed up compilation, if possible, test VHDL modules in isolation and temporarily comment out all the modules that you don’t need. This will save a lot of compilation time. While the module you are testing may have issues that only become apparent when other modules are active, in general, testing in isolation can root out a lot of bugs.
- Given the long VHDL compilation time, if modifying VHDL code, review any code changes and convince yourself that the code is logically correct before hitting the compile button. It will save you time.
- To future project students, we suggest starting with getting the hardware and software to talk to each other. You can try hard-coding in the SW just to make sure SW can update the signals in the HW that it wants to update. After you get the HW and SW to talk, proceed to get as much as the hardware done as possible. This advice is especially applicable to video game projects, or projects where visual information on the screen can be used to easily test and debug the software.

References:

[1] Edwards, Stephen. CSEE W4840 Embedded System Design Lab 3. Retrieved on May 13, 2011 from http://www.cs.columbia.edu/~sedwards/classes/2011/4840/lab3.pdf

[2] Edwards, Stephen. CSEE W4840 Embedded System Design Spring 2011 Lecture Slides:Sprite Graphics. Retrieved on May 13, 2011 from http://www.cs.columbia.edu/~sedwards/classes/2011/4840/sprites.pdf

[2] Edwards, Stephen. CSEE W4840 Embedded System Design Spring 2011 Lecture Slides:Video. Retrieved on May 13, 2011 from http://www.cs.columbia.edu/~sedwards/classes/2011/4840/video.pdf

8 Code Listing

8.1 invaders.c

```
////*
// * All Invaders Are Belong To Us
// *
// */

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

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

#define INTERVAL 10
#define HOME_INCREMENT 4
#define ALIEN_INCREMENT 5
#define LEFT_BOUNDARY 0
#define RIGHT_BOUNDARY 595
#define HOME_V_START 440
#define HOME_MISSILE_BOUNDARY 30
#define ALIEN_MISSILE_BOUNDARY 440
#define HOME_MISSILE_INC 15
#define ALIEN_MISSILE_INC 5
#define MAX_MISSILES 10


#define WAVE_RIGHT_BOUNDARY 264
#define WAVE_FINAL_ROW 300

#define ALIEN_LENGTH 16
#define ALIEN_WIDTH 16
#define BARRICADE_LENGTH 32
#define BARRICADE_WIDTH 32
#define ALIEN_GAP 16
#define NUM_ALIENS_PER_ROW 12
#define NUM_ROWS 5

#define BARRICADE1_H_COORD 125
#define BARRICADE1_V_COORD 375
#define BARRICADE2_H_COORD 225
#define BARRICADE2_V_COORD 375
#define BARRICADE3_H_COORD 325
#define BARRICADE3_V_COORD 375
#define BARRICADE4_H_COORD 425
#define BARRICADE4_V_COORD 375

#define NUM_H_BLOCKS_BARRICADE 4
#define BARRICADE_BLOCK_LENGTH 2
#define BARRICADE_BLOCK_WIDTH 8

#define INIT_WAVE_DELAY 3000
#define WAVE_DELAY_DEC 0
#define INIT_ACTIVE_MISSILES 4
```

```
#define ALIEN_CLOCK_FACTOR 1

//global variables for difficulty
int wave_delay = INIT_WAVE_DELAY;
int alien_missile_inc = ALIEN_MISSILE_INC;

//global variables
unsigned char code;
int val = 0;
int codeFlag = 0;
int direction = 0;
int wave_h = 60;
int wave_v = 30;
int home_missile_x = 0;
int home_missile_y = 0;
int rowId = 0;
int max_active_missiles = INIT_ACTIVE_MISSILES;
int randCol = 0;
int randRow = 0;

int rowIdx = 0;
int colIdx = 0;
int alienIdx = 0;
int m_idx = 0;

int barricade_collision = 0;
int barricade_col_id = 0;
int barricade_block_idx = 0;

int counter = 10000;
int alien_clock_factor = 1;
int alien_delay_counter = 0;

int alien_status[4];

int current_score = 0;
int display_score;
int save_score;
int lives1 = 3;
int save_lives;
int level = 0;

int cdelay=0;
int ddelay=0;

int first_alien_incol[NUM_ALIENS_PER_ROW];

int barricade_v_coords[4];
int barricade_h_coords[4];


int first_barricade_block[4][NUM_H_BLOCKS_BARRICADE];
int last_barricade_block[4][NUM_H_BLOCKS_BARRICADE];

int active_missiles[MAX_MISSILES];
```

```
int alien_missile_x[MAX_MISSILES];
int alien_missile_y[MAX_MISSILES];

int num_active_missiles = 0;

int restart = 0;
int key_delay=0;

//game state values//
int game_state=1;
int ent_count=2;
char state = 'l';
int state_reset = 0;
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
void restart_game()
{
    val = 0;
    IOWR_16DIRECT(VGA_BASE, 0,val);
    IOWR_32DIRECT(AUDIO_BASE,0,0x00000069);
    wave_delay = INIT_WAVE_DELAY;
    max_active_missiles = INIT_ACTIVE_MISSILES;
    counter = 10000;
    alien_delay_counter = 0;
    alien_clock_factor = ALIEN_CLOCK_FACTOR;
    level = 0;

    alien_missile_inc = ALIEN_MISSILE_INC;

    cdelay = 0;
    ddelay = 0;


    codeFlag = 0;
    direction = 0;
    wave_h = 60;
    wave_v = 30;
    int kk = 0;
    for ( ; kk < 4 ; kk++ ) {
        alien_status[kk] = 0xFFFF;
    }

    home_missile_x = 0;
    home_missile_y = 0;
    IOWR_VGA_DATA(VGA_BASE, 9, home_missile_x);
    IOWR_VGA_DATA(VGA_BASE, 10, home_missile_y);

    current_score = 0;
    display_score = current_score;
    IOWR_VGA_DATA(VGA_BASE, 3, display_score);
    IOWR_VGA_DATA(VGA_BASE, 4, display_score);
    IOWR_VGA_DATA(VGA_BASE, 5, display_score);
    IOWR_VGA_DATA(VGA_BASE, 6, display_score);
    IOWR_VGA_DATA(VGA_BASE, 7, display_score);
    //for lives
    lives1 = 3;
```

```
    IOWR_VGA_DATA(VGA_BASE, 8, lives1);

    int ii = 0; int jj = 0;

    for ( ii = 0 ; ii < NUM_ALIENS_PER_ROW; ii++ )
    {
       first_alien_incol[ii] = NUM_ROWS;
    }

    for ( ii = 0; ii < 4 ; ii++ )
    {
       for ( jj = 0 ; jj < NUM_H_BLOCKS_BARRICADE; jj++ ) //TODO: #define these constants
       {
          first_barricade_block[ii][jj] = 1;
          last_barricade_block[ii][jj] = 32/BARRICADE_BLOCK_LENGTH;
       }


    }

    barricade_v_coords[0] = BARRICADE1_V_COORD;
    barricade_v_coords[1] = BARRICADE2_V_COORD;
    barricade_v_coords[2] = BARRICADE3_V_COORD;
    barricade_v_coords[3] = BARRICADE4_V_COORD;

    barricade_h_coords[0] = BARRICADE1_H_COORD;
    barricade_h_coords[1] = BARRICADE2_H_COORD;
    barricade_h_coords[2] = BARRICADE3_H_COORD;
    barricade_h_coords[3] = BARRICADE4_H_COORD;


    for (ii = 0  ; ii < MAX_MISSILES; ii++ )
    {
       active_missiles[ii] = 0;
       alien_missile_x[ii] = 0;
       alien_missile_y[ii] = 0;
    }

    num_active_missiles = 0;

    IOWR_16DIRECT(VGA_BASE, 0,0);
    restart = 0;
    key_delay=0;
    state_reset=0;
}


void new_level()
{
    level++;

    cdelay = 0;
    ddelay = 0;

    IOWR_VGA_DATA(VGA_BASE,48,(level - level%10)/10);
    IOWR_VGA_DATA(VGA_BASE,49,level%10);
```

```
IOWR_32DIRECT(AUDIO_BASE,0,0x00000069);

val = 0;
IOWR_16DIRECT(VGA_BASE, 0,val);
codeFlag = 0;
direction = 0;
wave_h = 60;
wave_v = 30;

alien_delay_counter = 0;

if ( alien_clock_factor > 1 )
   alien_clock_factor--;

int kk = 0;
for ( ; kk < 4 ; kk++ ) {
   alien_status[kk] = 0xFFFF;
}

home_missile_x = 0;
home_missile_y = 0;
IOWR_VGA_DATA(VGA_BASE, 9, home_missile_x);
IOWR_VGA_DATA(VGA_BASE, 10, home_missile_y);


if (wave_delay > 1000 )
    wave_delay -= WAVE_DELAY_DEC;
if(max_active_missiles<MAX_MISSILES)
{
     max_active_missiles++;
}else {
   if ( alien_missile_inc < 12 )
   {
      alien_missile_inc += 2;
      max_active_missiles = 6;
   }


}
//for lives


int ii = 0; int jj = 0;

for ( ii = 0 ; ii < NUM_ALIENS_PER_ROW; ii++ )
{
   first_alien_incol[ii] = NUM_ROWS;
}

for ( ii = 0; ii < 4 ; ii++ )
{
   for ( jj = 0 ; jj < NUM_H_BLOCKS_BARRICADE; jj++ ) //TODO: #define these constants
   {
      first_barricade_block[ii][jj] = 1;
      last_barricade_block[ii][jj] = 32/BARRICADE_BLOCK_LENGTH;
   }
```

```
    }

    for (ii = 0  ; ii < MAX_MISSILES; ii++ )
    {
        active_missiles[ii] = 0;
        alien_missile_x[ii] = 0;
        alien_missile_y[ii] = 0;
    }

    num_active_missiles = 0;

    IOWR_16DIRECT(VGA_BASE, 0,0);
    restart = 0;
    key_delay=0;
    state_reset=0;
    state = 'I';


}

int alien_phalanx(direction)
{
    switch (direction)
    {
        case 0: //Moving Right
            IOWR_VGA_DATA(VGA_BASE,1,wave_h);
            IOWR_VGA_DATA(VGA_BASE,2,wave_v);

            if(wave_h < WAVE_RIGHT_BOUNDARY){
                wave_h++ ;
                IOWR_16DIRECT(VGA_BASE,2,wave_h);
            }
            else{
                direction = 1; //Turn around
                if (wave_v < WAVE_FINAL_ROW)//advance forward
                {
                    wave_v+=5;
                    IOWR_16DIRECT(VGA_BASE,4,wave_v);
                }
            }
            break;
        case 1: //Moving Left
            if(wave_h > 0){
                wave_h--;
                IOWR_16DIRECT(VGA_BASE,2,wave_h);
            }
            else{
                direction = 0; //Turn around
                if (wave_v < WAVE_FINAL_ROW)//advance forward
                {
                    wave_v+=5;
                    IOWR_16DIRECT(VGA_BASE,4,wave_v);
                }
            }
            break;
    }
```

```
    return direction;
}



char read_key()
{
    code=IORD_8DIRECT(PS2_BASE,4);
    printf("scan code is %x\n", code);
    switch (code)
    {
        case 0x29:
            //printf("space ! \n");

            if (codeFlag == 0)
            {
                if (state == 'I')
                    state = 'F';

                if (state == 'L')
                    state = 'G';

                if (state == 'R')
                    state = 'H';

                //codeFlag = 1;
            }
            else
            {
                if(codeFlag == 2)
                {
                    if (state == 'G')
                        state = 'L';
                    else if (state == 'H')
                        state = 'R';
                    else if (state == 'F')
                       state = 'I';
                    //codeFlag = 1;
                }
                codeFlag = 0;
            }
            break;

        case 0x5a:

                if ( game_state == 3 )
                {
                    if ( codeFlag == 0 ) {
                       game_state=1;
                       IOWR_VGA_DATA(VGA_BASE,52 , game_state);
                       //codeFlag = 1;
                    }
                    else
                    {
                     codeFlag = 0;
```

```
            }

        }
        else if((ent_count%2)==0)
        {
            //printf("\n1:ent count %d",ent_count);
            //toggle enter button state
            ent_count++;
            //change game state to 'playing'
            game_state=2;
            IOWR_VGA_DATA(VGA_BASE,52 , game_state);
             //printf("\n2:ent count %d",ent_count);
             //codeFlag = 0;
        }
        else if((ent_count%2)!=1)
        {
             printf("\n3:ent count %d",ent_count);
            ent_count++;
            IOWR_VGA_DATA(VGA_BASE, 0xFFFF, 0);
            restart_game();
            new_level();
             printf("\n4:ent count %d",ent_count);
             //codeFlag = 0;
        }
        codeFlag = 0;
    break;

case 0x6B:
  //printf("left ! \n");

    if ( codeFlag == 1 ) {
        if (state == 'I')
            state = 'L';

        if (state == 'F')
            state = 'G';

        if (state == 'R')
            state = 'L';

        if (state == 'H')
            state = 'G';
        codeFlag = 0;
    }
    else {
        if(codeFlag == 2)
        {
            if (state == 'G')
                state = 'F';
            else if (state == 'L')
                state = 'I';
            //codeFlag = 1;
        }
        codeFlag = 0;
    }
    break;
```

```
        case 0x74:
            //printf("right ! \n");

            if  ( codeFlag == 1 ) {

                if (state == 'I')
                    state = 'R';

                if (state == 'F')
                    state = 'H';

                if (state == 'L')
                    state = 'R';

                if (state == 'G')
                    state = 'H';
                codeFlag = 0;
            }
            else {
                if(codeFlag == 2)
                {
                    if (state == 'H')
                      state = 'F';
                    else if (state == 'R')
                      state = 'I';
                    //codeFlag = 1;
                }
                codeFlag = 0;
            }
            break;
        case 0xE0:
            //printf("E0 ! \n");
            codeFlag = 1;
            break;
        case 0xF0:
            codeFlag = 2;
            break;
        default:
            codeFlag = 0;
            break;
    }
    printf("state change to %c\n", state);
    printf("codeFlag is %i\n\n",codeFlag);
    return state;
}

void response(state)
{
    switch (state)
    {
        case 'I':
            //do nothing
            break;
        case 'L':
            val -= HOME_INCREMENT;
```

```
            if ( val < LEFT_BOUNDARY ) val += HOME_INCREMENT;
               IOWR_16DIRECT(VGA_BASE, 0,val);
            break;
         case 'R':
            val += HOME_INCREMENT;
            if  ( val > RIGHT_BOUNDARY ) val -= HOME_INCREMENT;
               IOWR_16DIRECT(VGA_BASE, 0,val);
            break;
         case 'F':
            if ( home_missile_y == 0 ) // means there is no active home missile
            {
                home_missile_x = val+16; // make this a constant
                home_missile_y = HOME_V_START;
                //cdelay=20000;
            }
            break;
         case 'H':
            val += HOME_INCREMENT;
            if  ( val > RIGHT_BOUNDARY ) val -= HOME_INCREMENT;
               IOWR_16DIRECT(VGA_BASE, 0,val);
            if ( home_missile_y == 0 ) // means there is no active home missile
            {
                home_missile_x = val+16; // make this a constant
                home_missile_y = HOME_V_START;
                //cdelay=20000;
            }
            break;
         case 'G':
            val -= HOME_INCREMENT;
            if ( val < LEFT_BOUNDARY ) val += HOME_INCREMENT;
               IOWR_16DIRECT(VGA_BASE, 0,val);
            if ( home_missile_y == 0 ) // means there is no active home missile
            {
                home_missile_x = val+16; // make this a constant
                home_missile_y = HOME_V_START;
               // cdelay=20000;
            }
            break;
         default:
            break;
      }
}

void delay(int time)
{
   while(time--);
}

int main()
{

   int num_active_cols = NUM_ALIENS_PER_ROW;
   int end_of_level_flag = 1;
   do
   {
```

```
restart_game();
IOWR_VGA_DATA(VGA_BASE, 0xFFFF, 0);
////////////////////////////////////////

//change game state to 'intro screen'//
game_state=1;
IOWR_VGA_DATA(VGA_BASE,52 , game_state);
////////////////////////////////////////




end_of_level_flag = 1;

while (lives1)
{



   if (end_of_level_flag)
   {
      printf("End of level flag set \n");
      new_level();
      IOWR_VGA_DATA(VGA_BASE, 0xFFFF, 0);
      num_active_cols = NUM_ALIENS_PER_ROW;
      end_of_level_flag = 0;
   }

   //Commanding the alien wave phalanx
   if (counter >= wave_delay)
   {

      int nn = 0;
      int breachFlag = 0;
      for ( nn = 0; nn < NUM_ALIENS_PER_ROW; nn++ )
      {

       if ((first_alien_incol[nn] > 0)
       && ( wave_v + first_alien_incol[nn] *  ( ALIEN_LENGTH )
       +   (first_alien_incol[nn] - 1) *  ( ALIEN_GAP ) > BARRICADE1_V_COORD-
BARRICADE_LENGTH))
         {

          breachFlag = 1;
          break;
         }
      }
      if ( breachFlag ) {
         printf("breached\n");
         lives1 = 0;
         continue;
      }


      // home missile display
```

```
            if ( home_missile_y > 0 )
            {
               if ( home_missile_y < HOME_MISSILE_BOUNDARY  )
               {
                  home_missile_y = 0;
                  home_missile_x = 0;
               }
               else
               {
                  home_missile_y -= HOME_MISSILE_INC;

                  barricade_collision = 0;
                  // figure out if the home missile collided with a barricade block
                  int kk =0;
                  for ( kk =0; kk < 4; kk++ )
                  {
                     if ( (home_missile_y >= barricade_v_coords[kk]) &&
                     (home_missile_x >= barricade_h_coords[kk]) &&
                     (home_missile_x < barricade_h_coords[kk]+BARRICADE_WIDTH) &&
                     (home_missile_y <= 400))
                     {
                        // change
                        barricade_col_id = (int)((home_missile_x -
barricade_h_coords[kk])/BARRICADE_BLOCK_WIDTH);
                        if ( last_barricade_block[kk][barricade_col_id] > 0 )
                        {
                           barricade_block_idx = (last_barricade_block[kk][barricade_col_id]-1) *
NUM_H_BLOCKS_BARRICADE + barricade_col_id;


                           IOWR_VGA_DATA(VGA_BASE,(3<<14)|(kk<<6)|barricade_block_idx,0);

                           if (( last_barricade_block[kk][barricade_col_id] == 1 )
                           || ( last_barricade_block[kk][barricade_col_id] ==
first_barricade_block[kk][barricade_col_id] ) )
                           {
                              first_barricade_block[kk][barricade_col_id] = 0;
                              last_barricade_block[kk][barricade_col_id] = 0;
                           }
                           else {
                              last_barricade_block[kk][barricade_col_id] =
last_barricade_block[kk][barricade_col_id]-1;
                           }



                           home_missile_y = 0;
                           home_missile_x = 0;
                           barricade_collision = 1;
                        }
                     }
                  }



                  // home missile barricade collision
```

```
if(barricade_collision==1 && game_state==2)
{
   ////////////produce sound////////////////
IOWR_32DIRECT(AUDIO_BASE,0,0x111100FF);
ddelay = 6;

}



if ( !barricade_collision ) {
   rowId = NUM_ROWS;

   if (( home_missile_x >= wave_h ) &&
    ( ( (home_missile_x - wave_h) % (ALIEN_WIDTH + ALIEN_GAP)) < ALIEN_WIDTH ))
   {
      // find the col index of the alien
      colIdx =    (int) ((home_missile_x - wave_h) / (ALIEN_WIDTH + ALIEN_GAP));

      if ((colIdx < NUM_ALIENS_PER_ROW) && ( first_alien_incol[colIdx] > 0) &&
       ( home_missile_y < (wave_v + ALIEN_LENGTH +
       ( first_alien_incol[colIdx] - 1) *  (ALIEN_LENGTH + ALIEN_GAP))))
      {
         // at least one invader belongs to us
         rowIdx = first_alien_incol[colIdx];
         first_alien_incol[colIdx] = first_alien_incol[colIdx] - 1;

         if ( !first_alien_incol[colIdx] )
         {
          num_active_cols--;
         }

         // convert 2D index to 1D
         alienIdx = (rowIdx - 1) * NUM_ALIENS_PER_ROW + colIdx;


         // deliver the good news to the hardware
         IOWR_VGA_DATA(VGA_BASE,(1<<15)|alienIdx,wave_v);

         // The 3 Billion Dollar home missile perished in the collision.
         // Cease its existence.
         home_missile_y = 0;
         home_missile_x = 0;
         // update score and send to the hardware
         current_score+=10;
         display_score = current_score;
         //IOWR_VGA_DATA(VGA_BASE, 7, current_score%10)
         display_score/=10;
         IOWR_VGA_DATA(VGA_BASE, 6, display_score%10);
         display_score/=10;
         IOWR_VGA_DATA(VGA_BASE, 5, display_score%10);
         display_score/=10;
         IOWR_VGA_DATA(VGA_BASE, 4, display_score%10);
         display_score/=10;
         IOWR_VGA_DATA(VGA_BASE, 3, display_score%10);
```

```
                    ////////////produce sound/////////////////
                    IOWR_32DIRECT(AUDIO_BASE,0,0x111100AA);

                    cdelay=6;




                    ////////////////////////////////////////
                }
            }
        }


    }
    IOWR_VGA_DATA(VGA_BASE, 9, home_missile_x);
    IOWR_VGA_DATA(VGA_BASE, 10, home_missile_y);
}

  if ( cdelay > 1 )
       {
          cdelay--;
       }
  else if ( cdelay == 1 )
       {
        cdelay = 0;
        IOWR_32DIRECT(AUDIO_BASE,0,0x00000069);

       }

  if ( ddelay > 1 )
       {
          ddelay--;
       }
  else if ( ddelay == 1 )
       {
        ddelay = 0;
        IOWR_32DIRECT(AUDIO_BASE,0,0x00000069);

       }




    alien_delay_counter = 0;
     // logic for alien waves
    direction = alien_phalanx(direction);

    m_idx = 0;
    while ( m_idx < max_active_missiles ) //alien attacks
    {
       if ( active_missiles[m_idx] )
       {
          alien_missile_y[m_idx] =  alien_missile_y[m_idx] + alien_missile_inc;
          if ( alien_missile_y[m_idx] > ALIEN_MISSILE_BOUNDARY )
```

```
                    {
                     active_missiles[m_idx] = 0;
                     alien_missile_x[m_idx] = 0;
                     alien_missile_y[m_idx] = 0;
                     num_active_missiles--;
                     IOWR_VGA_DATA(VGA_BASE, (1<<4) | m_idx , 0);
                     IOWR_VGA_DATA(VGA_BASE, (1<<5) | m_idx , 0);
                    }
                    else
                    {
                      IOWR_VGA_DATA(VGA_BASE, (1<<5) | m_idx , alien_missile_y[m_idx]);
                    }

                    barricade_collision = 0;


                    int kk = 0;
                    for ( kk =0; kk < 4 ; kk++ )
                    {
                        //figure out if the alien missile collided with a barricade block
                       if ( (alien_missile_y[m_idx] >= barricade_v_coords[kk]-16) &&
                       (alien_missile_x[m_idx] >= barricade_h_coords[kk]) &&
                       (alien_missile_x[m_idx] < barricade_h_coords[kk]+BARRICADE_WIDTH))
                       {

                           barricade_col_id = (int)((alien_missile_x[m_idx] -
barricade_h_coords[kk])/BARRICADE_BLOCK_WIDTH);
                           if ( first_barricade_block[kk][barricade_col_id] > 0 )
                           {
                              barricade_block_idx = (first_barricade_block[kk][barricade_col_id]-1) *
NUM_H_BLOCKS_BARRICADE + barricade_col_id;

                              IOWR_VGA_DATA(VGA_BASE,(3<<14)|(kk<<6)|barricade_block_idx,0);
                              if (( first_barricade_block[kk][barricade_col_id] == 16 )
                              || ( first_barricade_block[kk][barricade_col_id] ==
last_barricade_block[kk][barricade_col_id] ) )
                              {
                                 first_barricade_block[kk][barricade_col_id] = 0;
                                 last_barricade_block[kk][barricade_col_id] = 0;
                              }
                              else {
                                 first_barricade_block[kk][barricade_col_id] =
first_barricade_block[kk][barricade_col_id]+1;
                              }
                              alien_missile_y[m_idx] = 0;
                              alien_missile_x[m_idx] = 0;
                              active_missiles[m_idx] = 0;
                              num_active_missiles--;
                              barricade_collision = 1;
                              }



                       }
                    }
```

```
            //  missile barricade collision
              int edelay=20000;


            if(!barricade_collision){
            // killing our home ship
             if  (alien_missile_y[m_idx] >= ALIEN_MISSILE_BOUNDARY - alien_missile_inc &&
                   alien_missile_x[m_idx] >= val && alien_missile_x[m_idx] <=val+32)
              {
                IOWR_VGA_DATA(VGA_BASE, 0x000B, 0);
                lives1--;
                IOWR_VGA_DATA(VGA_BASE, 8, lives1);



                int i;
                for (i = 0  ; i < MAX_MISSILES; i++ )
                {
                   active_missiles[i] = 0;
                   alien_missile_x[i] = 0;
                   alien_missile_y[i] = 0;
                }
                num_active_missiles=0;
                state_reset = 1;
                IOWR_8DIRECT(PS2_BASE,0,0);
                delay(1000000);
                IOWR_VGA_DATA(VGA_BASE, 0x000B, 1 );
               }
            }
          }
          m_idx++;
}


if ( num_active_missiles < max_active_missiles ) //
{
          if ( !num_active_cols )
          {
           printf("End of level flag set 2\n");
          end_of_level_flag = 1;


            delay(900000);

          }
          else {
             // find the free slot
              m_idx = 0;
             while ( active_missiles[m_idx] ) {
                  m_idx++;
              }
             active_missiles[m_idx] = 1;
              num_active_missiles++;

             // pick an alien column in random
```

```
                    do {
                      randCol = ( rand() % NUM_ALIENS_PER_ROW ) ;
                    } while ( ! first_alien_incol[randCol] ) ;

                    // pick an alive alien in the column in random
                    randRow = rand() % first_alien_incol[randCol];

                     // figure out the missile origin
                     alien_missile_x[m_idx] = wave_h + (randCol) * (ALIEN_WIDTH + ALIEN_GAP) +
(ALIEN_WIDTH /2);
                     alien_missile_y[m_idx] = wave_v + (randRow) * (ALIEN_LENGTH + ALIEN_GAP) +
ALIEN_LENGTH;

                    IOWR_VGA_DATA(VGA_BASE, (1<<4) | m_idx , alien_missile_x[m_idx]);
                    IOWR_VGA_DATA(VGA_BASE, (1<<5) | m_idx , alien_missile_y[m_idx]);
                  }
              }

          counter = 0; // reset counter


        }
        else{
            counter++;
        }

        //Reading Keyboard Inputs
        if (IORD_8DIRECT(PS2_BASE,0))
        {
            state=read_key();
        }

        if(!key_delay)
        {
            response(state);
            key_delay=5000;
        }
        else
            key_delay--;

        if (state_reset)
        {
            state='I';
            state_reset = 0;
        }
      }
        //change game state to 'game over screen'//
        game_state=3;
        IOWR_VGA_DATA(VGA_BASE,52 , game_state);
        ////////////////////////////////////////

        //wait for another round of game
          while (!IORD_8DIRECT(PS2_BASE,0));
            code=IORD_8DIRECT(PS2_BASE,4);
          while (!IORD_8DIRECT(PS2_BASE,0));
            code= IORD_8DIRECT(PS2_BASE,4);
```

```
        while (!IORD_8DIRECT(PS2_BASE,0));
           code= IORD_8DIRECT(PS2_BASE,4);

       ////////////////////////////////////

       IOWR_VGA_DATA(VGA_BASE, 0xFFFF, 0);
            ent_count++;
            restart = 1;

   }while(1);
   printf("should never exit\n");

   return 0;
}
```

8.2 de2_vga_raster.vhd

```
--------------------------------------------------------------------------------
--
--
-- Simple VGA raster display
--
-- Stephen A. Edwards
-- sedwards@cs.columbia.edu
--
--------------------------------------------------------------------------------
--
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity de2_vga_raster is

  port (
    reset         : in std_logic;
    clk           : in std_logic;                    -- Should be 25.125 MHz
    read          : in  std_logic;
    write         : in  std_logic;
    chipselect    : in  std_logic;
    address       : in  unsigned(15 downto 0);
    readdata      : out unsigned(15 downto 0);
    writedata     : in  unsigned(15 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 unsigned(9 downto 0) -- Blue[9:0]
    );

end de2_vga_raster;
```

```
architecture rtl of de2_vga_raster is

  -- Video parameters

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

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

  -- Space Invader Game Parameter

  constant HOME_VSTART        : integer := 440; -- vertical starting position
of home ship
  constant SHIP_LENGTH        : integer := 32;
  constant SHIP_WIDTH         : integer := 32;
  constant ALIEN_LENGTH       : integer := 16;
  constant ALIEN_WIDTH        : integer := 16;
  constant GAP_LENGTH         : integer := 32;
  constant GAP_WIDTH          : integer := 16;
  constant HOME_MISSILE_LENGTH : integer := 16;
  constant HOME_MISSILE_WIDTH : integer := 2;

  constant STAR_LENGTH      : integer := 16;
  constant STAR_WIDTH         : integer := 16;
  constant STAR1_H_COORD    : integer := 70;
  constant STAR1_V_COORD    : integer := 90;
  constant STAR2_H_COORD    : integer := 156;
  constant STAR2_V_COORD    : integer := 300;
  constant STAR3_H_COORD    : integer := 228;
  constant STAR3_V_COORD    : integer := 450;
  constant STAR4_H_COORD    : integer := 280;
  constant STAR4_V_COORD    : integer := 50;
  constant STAR5_H_COORD    : integer := 390;
  constant STAR5_V_COORD    : integer := 200;
  constant STAR6_H_COORD    : integer := 400;
  constant STAR6_V_COORD    : integer := 400;
  constant STAR7_H_COORD    : integer := 500;
  constant STAR7_V_COORD    : integer := 33;
  constant STAR8_H_COORD    : integer := 460;
  constant STAR8_V_COORD    : integer := 103;
  constant STAR9_H_COORD    : integer := 580;
  constant STAR9_V_COORD    : integer := 300;
  constant STAR10_H_COORD    : integer := 600;
  constant STAR10_V_COORD    : integer := 420;
  constant STAR11_H_COORD    : integer := 170;
  constant STAR11_V_COORD    : integer := 190;
  constant STAR12_H_COORD    : integer := 20;
  constant STAR12_V_COORD    : integer := 500;
  constant STAR13_H_COORD    : integer := 128;
```

```
constant STAR13_V_COORD    : integer := 450;
constant STAR14_H_COORD    : integer := 180;
constant STAR14_V_COORD    : integer := 50;
constant STAR15_H_COORD    : integer := 290;
constant STAR15_V_COORD    : integer := 200;
constant STAR16_H_COORD    : integer := 275;
constant STAR16_V_COORD    : integer := 90;
constant STAR17_H_COORD    : integer := 345;
constant STAR17_V_COORD    : integer := 300;

constant NUM_ALIENS_PER_ROW: integer := 12;
constant NUM_ALIEN_ROWS     : integer := 5;

constant TEXT_LENGTH        : integer := 16;
constant TEXT_WIDTH         : integer := 80;
constant NUM_LENGTH         : integer := 16;
constant NUM_WIDTH          : integer := 16;

constant OFFSET_X           : integer := 32;
constant OFFSET_Y           : integer := 32;

constant SCORE_X            : integer :=100;
constant SCORE_Y            : integer :=20;

constant TIMER_H_COORD    : integer := 520;
constant TIMER_V_COORD    : integer := 64;
constant TIMER_H_COORD_1  : integer := 536;
constant TIMER_V_COORD_1  : integer := 80;
constant TIMER_H_COORD_2  : integer := 552;
constant TIMER_V_COORD_2  : integer := 80;

constant BARRICADE1_H_COORD: integer := 125;
constant BARRICADE1_V_COORD: integer := 375;
constant BARRICADE2_H_COORD: integer := 225;
constant BARRICADE2_V_COORD: integer := 375;
constant BARRICADE3_H_COORD: integer := 325;
constant BARRICADE3_V_COORD: integer := 375;
constant BARRICADE4_H_COORD: integer := 425;
constant BARRICADE4_V_COORD: integer := 375;
constant BARRICADE_LENGTH: integer := 32;
constant BARRICADE_WIDTH: integer := 32;
constant NUM_H_BLOCKS_BARRICADE : integer := 4;
constant BARRICADE_BLOCK_LENGTH : integer := 2;
constant BARRICADE_BLOCK_WIDTH : integer := 8;


constant SCORE_H_COORD      : integer := 520;
constant SCORE_V_COORD      : integer := 20;
constant NUM_DIGITS : integer := 5;
constant SCORE_H_COORD_1    : integer := 520;
constant SCORE_V_COORD_1    : integer := 36;
constant SCORE_LENGTH : integer := 16;
constant SCORE_WIDTH : integer := 16;
constant SCORE_GAP : integer := 0;

constant LIVES_H_COORD      : integer := 10;
constant LIVES_V_COORD      : integer := 20;
```

```
  constant LIVES_H_COORD_1    : integer := 18;
  constant LIVES_V_COORD_1    : integer := 36;
  constant LIVES_H_COORD_2    : integer := 42;
  constant LIVES_V_COORD_2    : integer := 36;
  constant LIVES_H_COORD_3    : integer := 66;
  constant LIVES_V_COORD_3    : integer := 36;

  -- Signals for the video controller

  signal clk25 : std_logic := '0';
  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  pixel_set, home_bit, wave_bit, score_bit,
score_bit_1, barricade1_bit,barricade2_bit,
barricade3_bit,barricade4_bit,star1_bit,star2_bit,star3_bit,star4_bit,
star5_bit,star6_bit,star7_bit,star8_bit,star9_bit,star10_bit,star11_bit,star1
2_bit,star13_bit,star14_bit,star15_bit,
star16_bit,star17_bit,homemissile_bit,timer_bit,timer_bit_1,timer_bit_2,timer
_bit_3,timer_bit_4,intro_bit,game_over_bit,
lives_bit,lives_bit_1,lives_bit_2,lives_bit_3 : std_logic;

      type alien_missile_array is array(9 downto 0) of std_logic;

    signal alienmissile_bits  : alien_missile_array;

  --type ram_type is array(15 downto 0) of unsigned(15 downto 0);
  --signal state_vector : ram_type ;
  signal home_status    : std_logic;
  signal ram_address    : unsigned(15 downto 0);
  signal home_h_coord   : unsigned(15 downto 0);
  signal home_h_coord_temp    : unsigned(15 downto 0);

  --signal state : states;
  signal wave_h_coord   : unsigned(15 downto 0);
  signal wave_v_coord   : unsigned(15 downto 0);

  signal wave_h_coord_temp    : unsigned(15 downto 0);
  signal wave_v_coord_temp    : unsigned(15 downto 0);

  signal wave_h_counter : integer := 0;
  signal wave_v_counter : integer := 0;

  signal alien_h_coord  : unsigned(15 downto 0);
  signal alien_v_coord  : unsigned(15 downto 0);


  signal score : unsigned(31 downto 0);
  type score_type is array(4 downto 0) of integer;
  signal score_array    : score_type;
  signal lives1_num     : integer;
  signal lives2_num     : integer;
  signal lives3_num     : integer;
  signal timer1_num     : integer;
```

```
  signal timer2_num     : integer;
  signal timer3_num     : integer;
  signal timer4_num     : integer;


  signal star_h_coord   : unsigned(15 downto 0);
  signal star_v_coord   : unsigned(15 downto 0);

  signal home_missile_x : integer;
  signal home_missile_y : integer;

  signal home_missile_x_temp : integer;
  signal home_missile_y_temp : integer;


  type alien_missiles is array(9 downto 0) of integer;
  signal alien_missile_x : alien_missiles;
  signal alien_missile_y : alien_missiles;
  signal alien_missile_x_temp : alien_missiles;
  signal alien_missile_y_temp : alien_missiles;

  --type bit_status is array(59 downto 0) of std_logic;
  --signal alien_status : bit_status;
  signal alien_status_1 : unsigned(31 downto 0);
  signal alien_status_2 : unsigned(31 downto 0);

--  type score_disp is array(15 downto 0) of unsigned(79 downto 0);
--  signal score_text : score_disp;
--  type lives_disp is array(15 downto 0) of unsigned(79 downto 0);
--  signal lives_text : lives_disp;

  type disp is array(15 downto 0) of unsigned(79 downto 0);
  signal score_text : disp;
  signal timer_text : disp;
  signal lives_text : disp;

  signal score_row : unsigned (79 downto 0);
  signal score_col : unsigned (79 downto 0);
  signal lives_row : unsigned (79 downto 0);
  signal lives_col : unsigned (79 downto 0);
  signal timer_row : unsigned (79 downto 0);
  signal counter:integer:=0;

  signal lives_row_1 : unsigned (15 downto 0);
  signal lives_row_2 : unsigned (15 downto 0);
  signal lives_row_3 : unsigned (15 downto 0);
  signal timer_row_1 : unsigned (15 downto 0);
  signal timer_row_2 : unsigned (15 downto 0);
  signal timer_row_3 : unsigned (15 downto 0);
  signal timer_row_4 : unsigned (15 downto 0);
  ----------------------------------

  type arr is array (29 downto 0) of integer;
  type matrix is array (15 downto 0,15 downto 0) of integer;

  type image_32_32 is array(31 downto 0) of unsigned(31 downto 0);
  signal home_ship : image_32_32;
```

```
  type image_16_16 is array(15 downto 0) of unsigned(15 downto 0);
  signal alien_ship : image_16_16;
  signal alien_ship2: image_16_16;
  signal alien_explode:image_16_16;
  signal star_sprite: image_16_16;
  signal star_sprite2:image_16_16;
  signal lives_sprite:image_16_16;

  signal sprite_0, sprite_1, sprite_2, sprite_3, sprite_4, sprite_5,
sprite_6, sprite_7, sprite_8, sprite_9 :image_16_16;
  signal home_ship_barricade : image_32_32;

  type per_barricade_bits is array(63 downto 0) of std_logic;
  type barr_status_bits is array(3 downto 0) of per_barricade_bits;
  signal barricade_status : barr_status_bits;

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

-------------------------------------------------------------------------
--game state = 1:hello screen
--game state = 2:game in progress
--game state = 3:game over
  signal game_state : integer:=1;

 type intro_array is array(86 downto 0) of unsigned(299 downto 0);
 signal intro: intro_array;
 signal intro_row:unsigned(299 downto 0);
 constant INTRO_H_COORD : integer := 170;
 constant INTRO_V_COORD : integer := 120;



 type game_over_arr is array(39 downto 0) of unsigned(299 downto 0);
 signal game_over : game_over_arr;
 signal game_over_row:unsigned(299 downto 0);
 constant GAME_OVER_H_COORD: integer :=180;
 constant GAME_OVER_V_COORD: integer :=120;

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

begin

  -- Horizontal and vertical counters

 --home_h_coord <= to_unsigned(300,16); -- home ship location
 home_ship(0)  <= "00000000000000011000000000000000" ;
 home_ship(1)  <= "00000000000000111100000000000000" ;
 home_ship(2)  <= "00000000000001111110000000000000" ;
 home_ship(3)  <= "00000000000011111111000000000000" ;
 home_ship(4)  <= "00000000000111111111100000000000" ;
 home_ship(5)  <= "00000000001110111101110000000000" ;
 home_ship(6)  <= "00000000011111000011111000000000" ;
 home_ship(7)  <= "00000000111111000011111100000000" ;
 home_ship(8)  <= "00000001111111100111111110000000" ;
 home_ship(9)  <= "00000011111111111111111111000000" ;
 home_ship(10) <= "00000111111011111111011111100000" ;
```

```
 home_ship(11) <= "00001111110011111111001111110000" ;
 home_ship(12) <= "00011111100111111111100111111000" ;
 home_ship(13) <= "00111111100111111111100111111100" ;
 home_ship(14) <= "01111111110011111111001111111110" ;
 home_ship(15) <= "11111111111111111111111111111111" ;
 home_ship(16) <= "11111111111011111111011111111111" ;
 home_ship(17) <= "01111111110001111111000111111110" ;
 home_ship(18) <= "00111111100000111110000011111100" ;
 home_ship(19) <= "00011111000000011000000011111000" ;
 home_ship(20) <= "00001110000000000000000001110000" ;
 home_ship(21) <= "00000100000000000000000000100000" ;
 home_ship(22) <= "00000000000000000000000000000000" ;
 home_ship(23) <= "00000000000000000000000000000000" ;
 home_ship(24) <= "00000000000000000000000000000000" ;
 home_ship(25) <= "00000000000000000000000000000000" ;
 home_ship(26) <= "00000000000000000000000000000000" ;
 home_ship(27) <= "00000000000000000000000000000000" ;
 home_ship(28) <= "00000000000000000000000000000000" ;
 home_ship(29) <= "00000000000000000000000000000000" ;
 home_ship(30) <= "00000000000000000000000000000000" ;
 home_ship(31) <= "00000000000000000000000000000000" ;
-------------------------------------------------------------

 alien_ship(0) <=  "0000011111100000" ;
 alien_ship(1) <=  "0000111111110000" ;
 alien_ship(2) <=  "0001111111111000" ;
 alien_ship(3) <=  "0011101111011100" ;
 alien_ship(4) <=  "0111000110001110" ;
 alien_ship(5) <=  "1111101111011111" ;
 alien_ship(6) <=  "1111111111111111" ;
 alien_ship(7) <=  "1111111111111111" ;
 alien_ship(8) <=  "0011000000001100" ;
 alien_ship(9) <=  "0011000000001100" ;
 alien_ship(10) <= "0000111111110000" ;
 alien_ship(11) <= "0000111111110000" ;
 alien_ship(12) <= "0011000000001100" ;
 alien_ship(13) <= "0011000000001100" ;
 alien_ship(14) <= "1100000000000011" ;
 alien_ship(15) <= "1100000000000011" ;

 alien_ship2(0) <=  "0000011111100000" ;
 alien_ship2(1) <=  "0000111111110000" ;
 alien_ship2(2) <=  "0001111111111000" ;
 alien_ship2(3) <=  "0011101111011100" ;
 alien_ship2(4) <=  "0111000110001110" ;
 alien_ship2(5) <=  "1111101111011111" ;
 alien_ship2(6) <=  "1111111111111111" ;
 alien_ship2(7) <=  "1111111111111111" ;
 alien_ship2(8) <=  "1111111111111111" ;
 alien_ship2(9) <=  "1100000000000011" ;
 alien_ship2(10) <= "1100000000000011" ;
 alien_ship2(11) <= "1100000000000011" ;
 alien_ship2(12) <= "0110000000000110" ;
 alien_ship2(13) <= "0011000000001100" ;
 alien_ship2(14) <= "0001100000011000" ;
 alien_ship2(15) <= "0000110000110000" ;
```

```
 alien_explode(0) <=  "1000000100000001" ;
 alien_explode(1) <=  "0100000010000010" ;
 alien_explode(2) <=  "0010000100000100" ;
 alien_explode(3) <=  "0001000010001000" ;
 alien_explode(4) <=  "0000100100010000" ;
 alien_explode(5) <=  "0000010010100000" ;
 alien_explode(6) <=  "0000000100000000" ;
 alien_explode(7) <=  "1111110000111111" ;
 alien_explode(8) <=  "0000000000000000" ;
 alien_explode(9) <=  "0000010100100000" ;
 alien_explode(10) <= "0000100010010000" ;
 alien_explode(11) <= "0001000100001000" ;
 alien_explode(12) <= "0010000010000100" ;
 alien_explode(13) <= "0100000100000010" ;
 alien_explode(14) <= "1000000010000001" ;
 alien_explode(15) <= "0000000100000000" ;

 star_sprite(0) <=  "0000000010000000" ;
 star_sprite(1) <=  "0000000111000000" ;
 star_sprite(2) <=  "0000000010000000" ;
 star_sprite(3) <=  "0000000000000000" ;
 star_sprite(4) <=  "0000000000000000" ;
 star_sprite(5) <=  "0000000000000000" ;
 star_sprite(6) <=  "0000000000000000" ;
 star_sprite(7) <=  "0000000000000000" ;
 star_sprite(8) <=  "0000000000000000" ;
 star_sprite(9) <=  "0000000000000000" ;
 star_sprite(10) <= "0000000000000000" ;
 star_sprite(11) <= "0000000000000000" ;
 star_sprite(12) <= "0000000000000000" ;
 star_sprite(13) <= "0000000000000000" ;
 star_sprite(14) <= "0000000000000000" ;
 star_sprite(15) <= "0000000000000000" ;

 star_sprite2(0) <=  "0000000000000000" ;
 star_sprite2(1) <=  "0000000000000000" ;
 star_sprite2(2) <=  "0000000000000000" ;
 star_sprite2(3) <=  "0000000000000000" ;
 star_sprite2(4) <=  "0000000000000000" ;
 star_sprite2(5) <=  "0000000000000000" ;
 star_sprite2(6) <=  "0000000000000000" ;
 star_sprite2(7) <=  "0000000000000000" ;
 star_sprite2(8) <=  "0000000000000000" ;
 star_sprite2(9) <=  "0000000000000000" ;
 star_sprite2(10) <= "0000000000000000" ;
 star_sprite2(11) <= "0000000000000000" ;
 star_sprite2(12) <= "0000000000000000" ;
 star_sprite2(13) <= "0000000000000000" ;
 star_sprite2(14) <= "0000000000000000" ;
 star_sprite2(15) <= "0000000000000000" ;
-------------------------------------------------------------------------------
-----------------------
 lives_sprite(0) <=  "0000000110000000" ;
 lives_sprite(1) <=  "0000001111000000" ;
 lives_sprite(2) <=  "0000011111100000" ;
 lives_sprite(3) <=  "0000110110110000" ;
 lives_sprite(4) <=  "0001111001111000" ;
```

```
 lives_sprite(5) <=  "0010011111001100" ;
 lives_sprite(6) <=  "0110111111101110" ;
 lives_sprite(7) <=  "1110011111001111" ;
 lives_sprite(8) <=  "1111111111111111" ;
 lives_sprite(9) <=  "1111101111011111" ;
 lives_sprite(10) <= "0111000110001110" ;
 lives_sprite(11) <= "0010000000000100" ;
 lives_sprite(12) <= "0000000000000000" ;
 lives_sprite(13) <= "0000000000000000" ;
 lives_sprite(14) <= "0000000000000000" ;
 lives_sprite(15) <= "0000000000000000" ;
--------------------------------------------------------------------------------
-----------------------
score_text( 0) <=
"0000000000000000000000000000000000000000000000000000000000000000000000000000
0000";
score_text( 1) <=
"0000000000000000000000000000000000000000000000000000000000000000000000000000
0000";
score_text( 2) <=
"0000000000000000000000000000000000000000000000000000000000000000000000000000
0000";
score_text( 3) <=
"0000111111111000000011111111100000001111111100000000111111110000000111111111
1000";
score_text( 4) <=
"0001111111111000000111111111100000011111111110000001111111111000000111111111
1000";
score_text( 5) <=
"0001100000000000000110000000000000011000001100000011000000110000001100000000
0000";
score_text( 6) <=
"0001100000000000000110000000000000011000001100000011000000110000001100000000
0000";
score_text( 7) <=
"0001111111111000000110000000000000011000001100000011111111110000001111111111
1000";
score_text( 8) <=
"0000111111111000000110000000000000011000001100000011111111110000001111111111
1000";
score_text( 9) <=
"0000000000011000000110000000000000011000001100000011001110000000001100000000
0000";
score_text(10) <=
"0000000000011000000110000000000000011000001100000011000111000000001100000000
0000";
score_text(11) <=
"0001111111111000000111111111100000011111111100000011000011100000001111111111
1000";
score_text(12) <=
"0001111111111000000001111111110000000111111110000000110000011100000001111111111
1000";
score_text(13) <=
"0000000000000000000000000000000000000000000000000000000000000000000000000000
0000";
```

```
score_text(14) <=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000";
score_text(15) <=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000";
--------------------------------------------------------------------------------
-------------------------
lives_text( 0) <=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000";
lives_text( 1) <=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000";
lives_text( 2) <=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000";
lives_text( 3) <=
"00000011000000000001111111111000000110000011000001111111110000000111111111000
0000";
lives_text( 4) <=
"00000011000000000001111111111000000110000011000001111111110000000111111111000
0000";
lives_text( 5) <=
"00000011000000000000000110000000000110000011000001100000000000011000000000000
0000";
lives_text( 6) <=
"00000011000000000000000110000000000110001100000011000000000000011000000000000
0000";
lives_text( 7) <=
"00000011000000000000000110000000000110001100000011111111110000001111111110000
0000";
lives_text( 8) <=
"00000011000000000000000110000000000110001100000011111111110000001111111110000
0000";
lives_text( 9) <=
"00000011000000000000000110000000000011011000000011000000000000000000000110000
0000";
lives_text(10) <=
"00000011000000000000000110000000000011011000000011000000000000000000000110000
0000";
lives_text(11) <=
"00000011111111110000111111111100000000110110000000111111111000000111111111000
0000";
lives_text(12) <=
"00000011111111110000111111111100000000011100000000111111111000000111111111000
0000";
lives_text(13) <=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000";
lives_text(14) <=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000";
lives_text(15) <=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000";
```

```
------------------------------------------------------------------------------
------------------------
timer_text( 0) <=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000";
timer_text( 1) <=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000";
timer_text( 2) <=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000";
timer_text( 3) <=
"00000011000000000000111111111110000011000001100000111111111000001100000000000000
0000";
timer_text( 4) <=
"00000011000000000000111111111110000011000001100000111111111000001100000000000000
0000";
timer_text( 5) <=
"00000011000000000000110000000000000011000001100000110000000000001100000000000000
0000";
timer_text( 6) <=
"00000011000000000000110000000000000011000110000001100000000000001100000000000000
0000";
timer_text( 7) <=
"00000011000000000000111111111110000001100011000000111111111000001100000000000000
0000";
timer_text( 8) <=
"00000011000000000000111111111110000001100011000000111111111000001100000000000000
0000";
timer_text( 9) <=
"00000011000000000000110000000000000001101100000001100000000000001100000000000000
0000";
timer_text(10) <=
"00000011000000000000110000000000000001101100000001100000000000001100000000000000
0000";
timer_text(11) <=
"00000011111111110000111111111110000000110110000000111111111000001111111110000000
0000";
timer_text(12) <=
"00000011111111110000111111111110000000011100000000111111111000001111111110000000
0000";
timer_text(13) <=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000";
timer_text(14) <=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000";
timer_text(15) <=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000";



------------------------------------------------------------------------------
------------------------
sprite_0( 0) <= "0000000000000000";
```

```
sprite_0( 1) <= "0000000000000000";
sprite_0( 2) <= "0000000000000000";
sprite_0( 3) <= "0000111111110000";
sprite_0( 4) <= "0001111111111000";
sprite_0( 5) <= "0001100000011000";
sprite_0( 6) <= "0001100000011000";
sprite_0( 7) <= "0001100000011000";
sprite_0( 8) <= "0001100000011000";
sprite_0( 9) <= "0001100000011000";
sprite_0(10) <= "0001100000011000";
sprite_0(11) <= "0001111111111000";
sprite_0(12) <= "0000111111110000";
sprite_0(13) <= "0000000000000000";
sprite_0(14) <= "0000000000000000";
sprite_0(15) <= "0000000000000000";

sprite_1( 0) <= "0000000000000000";
sprite_1( 1) <= "0000000000000000";
sprite_1( 2) <= "0000000000000000";
sprite_1( 3) <= "0000001111000000";
sprite_1( 4) <= "0000011111000000";
sprite_1( 5) <= "0000111111000000";
sprite_1( 6) <= "0000000111000000";
sprite_1( 7) <= "0000000111000000";
sprite_1( 8) <= "0000000111000000";
sprite_1( 9) <= "0000000111000000";
sprite_1(10) <= "0000000111000000";
sprite_1(11) <= "0000000111000000";
sprite_1(12) <= "0000000111000000";
sprite_1(13) <= "0000000000000000";
sprite_1(14) <= "0000000000000000";
sprite_1(15) <= "0000000000000000";

sprite_2( 0) <= "0000000000000000";
sprite_2( 1) <= "0000000000000000";
sprite_2( 2) <= "0000000000000000";
sprite_2( 3) <= "0000111111110000";
sprite_2( 4) <= "0001111111111000";
sprite_2( 5) <= "0001110000111000";
sprite_2( 6) <= "0000000000111000";
sprite_2( 7) <= "0000000001110000";
sprite_2( 8) <= "0000000011100000";
sprite_2( 9) <= "0000000111000000";
sprite_2(10) <= "0000001110000000";
sprite_2(11) <= "0000111111111000";
sprite_2(12) <= "0001111111111000";
sprite_2(13) <= "0000000000000000";
sprite_2(14) <= "0000000000000000";
sprite_2(15) <= "0000000000000000";

sprite_3( 0) <= "0000000000000000";
sprite_3( 1) <= "0000000000000000";
sprite_3( 2) <= "0000000000000000";
sprite_3( 3) <= "0000111111110000";
sprite_3( 4) <= "0001111111111000";
sprite_3( 5) <= "0001100000011000";
sprite_3( 6) <= "0000000000011000";
```

```
sprite_3( 7) <= "0000000111110000";
sprite_3( 8) <= "0000000111110000";
sprite_3( 9) <= "0000000000011000";
sprite_3(10) <= "0001100000011000";
sprite_3(11) <= "0001111111111000";
sprite_3(12) <= "0000111111110000";
sprite_3(13) <= "0000000000000000";
sprite_3(14) <= "0000000000000000";
sprite_3(15) <= "0000000000000000";

sprite_4( 0) <= "0000000000000000";
sprite_4( 1) <= "0000000000000000";
sprite_4( 2) <= "0000000000000000";
sprite_4( 3) <= "0000000011111000";
sprite_4( 4) <= "0000000111111000";
sprite_4( 5) <= "0000001110011000";
sprite_4( 6) <= "0000011100011000";
sprite_4( 7) <= "0000111000011000";
sprite_4( 8) <= "0001111111111100";
sprite_4( 9) <= "0001111111111100";
sprite_4(10) <= "0000000000011000";
sprite_4(11) <= "0000000000011000";
sprite_4(12) <= "0000000000011000";
sprite_4(13) <= "0000000000000000";
sprite_4(14) <= "0000000000000000";
sprite_4(15) <= "0000000000000000";

sprite_5( 0) <= "0000000000000000";
sprite_5( 1) <= "0000000000000000";
sprite_5( 2) <= "0000000000000000";
sprite_5( 3) <= "0001111111111000";
sprite_5( 4) <= "0001111111111000";
sprite_5( 5) <= "0001100000000000";
sprite_5( 6) <= "0001100000000000";
sprite_5( 7) <= "0001111111110000";
sprite_5( 8) <= "0001111111111000";
sprite_5( 9) <= "0000000000011000";
sprite_5(10) <= "0001100000011000";
sprite_5(11) <= "0001111111111000";
sprite_5(12) <= "0000111111110000";
sprite_5(13) <= "0000000000000000";
sprite_5(14) <= "0000000000000000";
sprite_5(15) <= "0000000000000000";

sprite_6( 0) <= "0000000000000000";
sprite_6( 1) <= "0000000000000000";
sprite_6( 2) <= "0000000000000000";
sprite_6( 3) <= "0000111111110000";
sprite_6( 4) <= "0001111111111000";
sprite_6( 5) <= "0001100000011000";
sprite_6( 6) <= "0001100000000000";
sprite_6( 7) <= "0001111111110000";
sprite_6( 8) <= "0001111111111000";
sprite_6( 9) <= "0001100000011000";
sprite_6(10) <= "0001100000011000";
sprite_6(11) <= "0001111111111000";
sprite_6(12) <= "0000111111110000";
```

```
sprite_6(13) <= "0000000000000000";
sprite_6(14) <= "0000000000000000";
sprite_6(15) <= "0000000000000000";

sprite_7( 0) <= "0000000000000000";
sprite_7( 1) <= "0000000000000000";
sprite_7( 2) <= "0000000000000000";
sprite_7( 3) <= "0001111111111000";
sprite_7( 4) <= "0001111111111000";
sprite_7( 5) <= "0000000001110000";
sprite_7( 6) <= "0000000001110000";
sprite_7( 7) <= "0000000011100000";
sprite_7( 8) <= "0000000011100000";
sprite_7( 9) <= "0000000111000000";
sprite_7(10) <= "0000000111000000";
sprite_7(11) <= "0000001110000000";
sprite_7(12) <= "0000001110000000";
sprite_7(13) <= "0000000000000000";
sprite_7(14) <= "0000000000000000";
sprite_7(15) <= "0000000000000000";

sprite_8( 0) <= "0000000000000000";
sprite_8( 1) <= "0000000000000000";
sprite_8( 2) <= "0000000000000000";
sprite_8( 3) <= "0000111111110000";
sprite_8( 4) <= "0001111111111000";
sprite_8( 5) <= "0001100000011000";
sprite_8( 6) <= "0001100000011000";
sprite_8( 7) <= "0000111111110000";
sprite_8( 8) <= "0000111111110000";
sprite_8( 9) <= "0001100000011000";
sprite_8(10) <= "0001100000011000";
sprite_8(11) <= "0001111111111000";
sprite_8(12) <= "0000111111110000";
sprite_8(13) <= "0000000000000000";
sprite_8(14) <= "0000000000000000";
sprite_8(15) <= "0000000000000000";

sprite_9( 0) <= "0000000000000000";
sprite_9( 1) <= "0000000000000000";
sprite_9( 2) <= "0000000000000000";
sprite_9( 3) <= "0000111111110000";
sprite_9( 4) <= "0001111111111000";
sprite_9( 5) <= "0001100000011000";
sprite_9( 6) <= "0001100000011000";
sprite_9( 7) <= "0001111111111000";
sprite_9( 8) <= "0000111111111000";
sprite_9( 9) <= "0000000000011000";
sprite_9(10) <= "0001100000011000";
sprite_9(11) <= "0001111111111000";
sprite_9(12) <= "0000111111110000";
sprite_9(13) <= "0000000000000000";
sprite_9(14) <= "0000000000000000";
sprite_9(15) <= "0000000000000000";
-----------------------------------------------------------------
 home_ship_barricade(0) <=  "11111111000001111100000111111111" ;
 home_ship_barricade(1) <=  "11111111000001111100000111111111" ;
```

```
 home_ship_barricade(2) <=  "11111111000001111100000111111111" ;
 home_ship_barricade(3) <=  "11111111000001111100000111111111" ;
 home_ship_barricade(4) <=  "11111111000001111100000111111111" ;
 home_ship_barricade(5) <=  "11111111000001111100000111111111" ;
 home_ship_barricade(6) <=  "11111111111111111111111111111111" ;
 home_ship_barricade(7) <=  "11111111111111111111111111111111" ;
 home_ship_barricade(8) <=  "11111111111111111111111111111111" ;
 home_ship_barricade(9) <=  "11111111111110000011111111111111" ;
 home_ship_barricade(10) <= "11111111111110000011111111111111" ;
 home_ship_barricade(11) <= "11111111111110000011111111111111" ;
 home_ship_barricade(12) <= "11111111111110000011111111111111" ;
 home_ship_barricade(13) <= "11111111111111111111111111111111" ;
 home_ship_barricade(14) <= "11111111111111111111111111111111" ;
 home_ship_barricade(15) <= "11111111111111111111111111111111" ;
 home_ship_barricade(16) <= "11111111111111111111111111111111" ;
 home_ship_barricade(17) <= "11111111111110000011111111111111" ;
 home_ship_barricade(18) <= "11111111111110000011111111111111" ;
 home_ship_barricade(19) <= "11111111111110000011111111111111" ;
 home_ship_barricade(20) <= "11111111111110000011111111111111" ;
 home_ship_barricade(21) <= "11111111111111111111111111111111" ;
 home_ship_barricade(22) <= "11111111111111111111111111111111" ;
 home_ship_barricade(23) <= "11111111111111111111111111111111" ;
 home_ship_barricade(24) <= "11111111111111111111111111111111" ;
 home_ship_barricade(25) <= "11111111111111111111111111111111" ;
 home_ship_barricade(26) <= "11111111111111111111111111111111" ;
 home_ship_barricade(27) <= "11111111111111111111111111111111" ;
 home_ship_barricade(28) <= "11111111111111111111111111111111" ;
 home_ship_barricade(29) <= "11111111111111111111111111111111" ;
 home_ship_barricade(30) <= "11111111111111111111111111111111" ;
 home_ship_barricade(31) <= "11111111111111111111111111111111" ;


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

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

intro(0)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(1)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(2)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(3)<=
"00000000001111111111111111000000111100000000000000000111100000000000000000
00000000011111111111111111100000011111000000000111100000011110000000000001
11100000011111111111111111000000111111111111111110000000000111111111111111
00000011111111111111111000000011111111111111111000000000000000000000000000";
```

```
intro(4)<=
"000000000011111111111111110000011110000000000000000011110000000000000000
000000000111111111111111111100000111111000000001111000000111100000000000011
110000000111111111111111100000011111111111111111100000000011111111111111111
000000111111111111111110000000111111111111111110000000000000000000000000";
intro(5)<=
"000000000011100000000001110000011110000000000000000011110000000000000000
000000000000000011111000000000000111111100000011110000000011110000000000111
100000000111000000000011100000011100000000000011100000000111000000000000000
000001110000000000011100000011100000000000000000000000000000000000000000";
intro(6)<=
"000000000011100000000001110000011110000000000000000011110000000000000000
000000000000000011111000000000000111100111000011110000000011110000000011111
000000000111000000000011100000011100000000000011100000000111000000000000000
000001110000000000011100000011100000000000000000000000000000000000000000";
intro(7)<=
"000000000011111111111111110000011110000000000000000011110000000000000000
000000000000000011111000000000000111100011100011110000000000111100000011110
000000000111111111111111100000011100000000000011100000001111111111111100000
000001111111111111111110000000111111111111111110000000000000000000000000";
intro(8)<=
"000000000011111111111111110000011110000000000000000011110000000000000000
000000000000000011111000000000000111100011100111100000000000111100001111100
000000000111111111111111100000011100000000000011100000001111111111111100000
000001111111111111111110000000111111111111111110000000000000000000000000";
intro(9)<=
"000000000011100000000001110000011110000000000000000011110000000000000000
000000000000000011111000000000000111100001110111100000000000111111111111000
000000000111000000000011100000011100000000000011100000000111000000000000000
000001110000001110000000000000000000000000001110000000000000000000000000";
intro(10)<=
"000000000011100000000001110000011110000000000000000011110000000000000000
000000000000000011111000000000000111100000111111100000000000011111111100000
000000000111000000000011100000011100000000000011100000000111000000000000000
000001110000000111000000000000000000000000001110000000000000000000000000";
intro(11)<=
"000000000011100000000001110000011111111111111110000001111111111111111000000
000000000111111111111111111100000111100000011111100000000000000011111100000
000000000111000000000011100000011111111111111111100000000011111111111111111
000001110000000011100000000001111111111111111100000000000000000000000000";
intro(12)<=
"000000000011100000000001110000011111111111111110000001111111111111111000000
000000000111111111111111111100000111100000000111100000000000000011111100000
000000000111000000000011100000011111111111111111100000000011111111111111111
000000111000000000111100000001111111111111111100000000000000000000000000";
intro(13)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(14)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(15)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000
```

```
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(16)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(17)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(18)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000001111111111111111110000001111111111111111110000001111
111111111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(19)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000001111111111111111110000001111111111111111110000001111
111111111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(20)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000001110000000000001110000001110000000000011100000011
100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(21)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000001110000000000001110000001110000000000011100000011
100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(22)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000001111111111111111110000001111111111111111110000001111
111111111000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(23)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000001111111111111111110000001111111111111111110000001111
111111111000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(24)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000001110000000000001110000001110000000111000000000011
100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(25)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000001110000000000001110000001110000000111000000000011
100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(26)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000001110000000000001110000001110000000001110000000011
```

```
11111111111111110000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(27)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000011100000000000111000000111000000000011110000001 1
11111111111111110000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(28)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(29)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(30)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(31)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(32)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(33)<=
"00000000000000000000011111111111111110000001111111111111110000001111000000
00000000000111111111111111100000011111100000001111100000011111111111111111000
00000000000011111111111111111110000001111111111111111000000000000000111000000
00000011100000011111111111111110000000000000000000000000000000000000000000000";
intro(34)<=
"00000000000000000000011111111111111110000001111111111111110000001111000000
00000000000111111111111111100000011111100000001111100000011111111111111111000
00000000000011111111111111111110000001111111111111111000000000000000111000000
00000011100000011111111111111110000000000000000000000000000000000000000000000";
intro(35)<=
"00000000000000000000011100000000000111000000111000000000000000000000011110000000
00000000000111000000000111000000111110111000011111000000011100000000000000000
00000000000000000000111111000000000000011100000000011100000000000000111000000
00000011100000011100000000000000000000000000000000000000000000000000000000000";
intro(36)<=
"00000000000000000000011100000000000111000000111000000000000000000000011110000000
00000000000111000000000111000000111110011100011111000000011100000000000000000
00000000000000000000111111000000000000011100000000011100000000000000111000000
00000011100000011100000000000000000000000000000000000000000000000000000000000";
intro(37)<=
"00000000000000000000011111111111111110000001111111111110000000000011110000000
00000000000111000000000111000000111110001110011111000000011100000001111111000
00000000000000000000111111000000000000011100000000011100000000000000111000000
00000011100000011111111111111110000000000000000000000000000000000000000000000";
```

```
intro(38)<=
"00000000000000000000011111111111111110000011111111111100000000011110000000
00000000001110000000000111000001111100001110111110000001110000001111111000
00000000000000000001111110000000000001110000000001110000000000000111000000
00000011100000011111111111111111000000000000000000000000000000000000000";
intro(39)<=
"00000000000000000000011100000000001110000011100000000000000000000011110000000
00000000001110000000000111000001111100000111111100000011100000000000111000
00000000000000000001111110000000000001110000000001110000000000000111000000
00000011100000000000000000000111000000000000000000000000000000000000000";
intro(40)<=
"00000000000000000000011100000000001110000011100000000000000000000011110000000
00000000001110000000000111000001111100000111111100000011100000000000111000
00000000000000000001111110000000000001110000000001110000000000000111000000
00000011100000000000000000000111000000000000000000000000000000000000000";
intro(41)<=
"00000000000000000000011111111111111110000011111111111111100000011111111111
11111000001111111111111111000000111110000000111111000000111111111111111000
00000000000000000001111110000000000001111111111111111000000000000111111111
11111111100000011111111111111111000000000000000000000000000000000000000";
intro(42)<=
"00000000000000000000011111111111111110000011111111111111100000011111111111
11111000001111111111111111000000111110000000111111000000111111111111111000
00000000000000000001111110000000000001111111111111111000000000000111111111
11111111100000011111111111111111000000000000000000000000000000000000000";
intro(43)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000";
intro(44)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000";
intro(45)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000";
intro(46)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000";
intro(47)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000";
intro(48)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000";
intro(49)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000
```

```
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000";
intro(50)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000";
intro(51)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000";
intro(52)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000";
intro(53)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000";
intro(54)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000";
intro(55)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000";
intro(56)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000";
intro(57)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000";
intro(58)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000";
intro(59)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000";
intro(60)<=
"000000000000000000000000000000000000000000000000000011111111111111111000000111
111111111111100000011110000000000000111100000011111111111111110000000000000000
```

```
0001111111111111111110000000011100000000000011100000000111111111111111000000
0000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(61)<=
"0000000000000000000000000000000000000000000000000011111111111111111000000111
1111111111110000000111100000000000011110000000111111111111111100000000000000
0001111111111111111110000000011100000000000011100000000111111111111111000000
0000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(62)<=
"0000000000000000000000000000000000000000000000001110000000000000000000000111
0000000000111000000001111000000000011110000000111000000000000000000000000000
0000000011111100000000000000111000000000001110000000011100000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(63)<=
"0000000000000000000000000000000000000000000000001110000000000000000000000111
0000000000111000000001111000000000011110000000111000000000000000000000000000
0000000011111100000000000000111000000000001110000000011100000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(64)<=
"0000000000000000000000000000000000000000000000001111111111111111111000000111
1111111111110000000000111100000111100000000000111111111111000000000000000000
0000000011111100000000000000111111111111111111100000000111111111100000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(65)<=
"0000000000000000000000000000000000000000000000001111111111111111111000000111
1111111111110000000000111100001111000000000001111111111110000000000000000000
0000000011111100000000000000111111111111111111100000000111111111100000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(66)<=
"0000000000000000000000000000000000000000000000000000000000000000000111000000111
0000000000111000000000000111000011100000000000011100000000000000000000000000
0000000011111100000000000000111000000000000111000000001110000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(67)<=
"0000000000000000000000000000000000000000000000000000000000000000000111000000111
0000000000111000000000000111111110000000000000111000000000000000000000000000
0000000011111100000000000000111000000000000111000000001110000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(68)<=
"0000000000000000000000000000000000000000000000001111111111111111111000000111
0000000000111000000000000111111100000000000001111111111111111100000000000000
0000000011111100000000000000111000000000000111000000001111111111111110000000
0000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(69)<=
"0000000000000000000000000000000000000000000000001111111111111111111000000111
0000000000111000000000000111111000000000000011111111111111111000000000000000
0000000011111100000000000000111000000000000111000000001111111111111110000000
0000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(70)<=
"0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(71)<=
"0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000";
```

```
intro(72)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(73)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(74)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(75)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(76)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001
111111111111111100000011110000000000000000001111111111111111100000011111100000
001111100000011111111111111111000000111111111111111111100000000111111111111111
110000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(77)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001
111111111111111100000011110000000000000000001111111111111111100000011111110000
001111100000011111111111111111000000111111111111111111100000000111111111111111
110000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(78)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001
110000000000011100000011110000000000000000001110000000000111000001111101110 0
001111100000011100000000000000000000000011111000000000000000000000000000000001
110000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(79)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001
110000000000011100000011110000000000000000001110000000000111000001111100111 0
001111100000011100000000000000000000000011111000000000000000000000000000000001
110000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(80)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001
111111111111111100000011110000000000000000001111111111111111100000011111000111
001111100000011111111111000000000000000000011111000000000000000000000011111111111
110000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(81)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001
111111111111111100000011110000000000000000001111111111111111100000011111000011
101111100000011111111111000000000000000000011111000000000000000000000011111111111
110000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(82)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001
110000000000000000000011110000000000000000001110000000000111000001111100000 1
111111100000011100000000000000000000000011111000000000000000000000011100000000
000000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(83)<=
"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001
```

```
11000000000000000000001111000000000000000000111000000000011100000011111000000
11111110000001110000000000000000000000000111110000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(84)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000001
11000000000000000000001111111111111111000001110000000000111000001111100000000
01111110000001111111111111111100000000000011111000000000000000000011100000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(85)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000001
11000000000000000000011111111111111110000011100000000001110000011111000000000
00011110000001111111111111111000000000000111110000000000000000000011100000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
intro(86)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";



game_over(0)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(1)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(2)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(3)<=
"00000000000111111111111111111000000011111111111111111000000111111111111111
00000011111111111111111100000000000000000111111111111111100000011110000000000
00000111100000011111111111111111000000111111111111111110000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(4)<=
"00000000000111111111111111111000000011111111111111111000000111111111111111
00000011111111111111111100000000000000000111111111111111100000011110000000000
00001111000000011111111111111111000000111111111111111110000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(5)<=
"00000000000111000000000000000000000011100000000000111000000111000011100001111
00000011100000000000000000000000000000111000000000001110000000011110000000000
00011110000000011100000000000000000001110000000000011100000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(6)<=
"00000000000111000000000000000000000011100000000000111000000111000011100001111
00000011100000000000000000000000000000111000000000001110000000011110000000000
00111100000000011100000000000000000001110000000000011100000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
```

```
game_over(7)<=
"00000000000111000000001111111000000011111111111111110000001110000111000011
00000011111111111000000000000000000000000111000000000001110000000000111100000
01111000000000011111111110000000000000111111111111111111000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(8)<=
"00000000000111000000001111111000000011111111111111110000001110000111000011
00000011111111111000000000000000000000000111000000000001110000000000111100000
11110000000000011111111110000000000000111111111111111111000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(9)<=
"00000000000111000000000000111000000011100000000000111000000111000000000000111
00000011100000000000000000000000000000000111000000000001110000000000011110011
11100000000000011100000000000000000000111000000111000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(10)<=
"00000000000111000000000000111000000011100000000000111000000111000000000000111
00000011100000000000000000000000000000000111000000000001110000000000001111111
11000000000000011100000000000000000000111000000011100000000000011110000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(11)<=
"00000000000111111111111111111000000011100000000000111000000111000000000000111
00000011111111111111111100000000000000000111111111111111110000000000000011111
11000000000000011111111111111111100000011100000000001110000000000111100000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(12)<=
"00000000000111111111111111111000000011100000000000111000000111000000000000111
00000011111111111111111100000000000000000111111111111111110000000000000011111
10000000000000011111111111111111100000011100000000001110000000000111100000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(13)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000111000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(14)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000110000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(15)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(16)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(17)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(18)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000
```

```
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(19)<=
"00000000000000000000000000000000000000000000000000011100000000000000011100000011
10000000000000111000000111111111111111111100000011111111111111111100000011111
10000000000111000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(20)<=
"00000000000000000000000000000000000000000000000000011100000000000000011100000011
10000000000000111000000111111111111111111100000011111111111111111100000011111
10000000000111000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(21)<=
"00000000000000000000000000000000000000000000000000011100000000000000011100000011
10000000000000111000000111000001110000011100000011100000000000011100000011101
11000000000111000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(22)<=
"00000000000000000000000000000000000000000000000000011100000000000000011100000011
10000000000000111000000111000001110000011100000011100000000000011100000011100
11100000000111000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(23)<=
"00000000000000000000000000000000000000000000000000011111111111111111100000011
10000000000000111000000111000001110000011100000011111111111111111100000011100
01110000000111000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(24)<=
"00000000000000000000000000000000000000000000000000011111111111111111100000011
10000000000000111000000111000001110000011100000011111111111111111100000011100
00111000000111000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(25)<=
"00000000000000000000000000000000000000000000000000011100000000000000011100000011
10000000000000111000000111000000000000111000000111000000000000111000000011100
00011100000111000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(26)<=
"00000000000000000000000000000000000000000000000000011100000000000000011100000011
10000000000000111000000111000000000000111000000111000000000000111000000011100
00001110000111000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(27)<=
"00000000000000000000000000000000000000000000000000011100000000000000011100000011
10000000000000111000000111000000000000111000000111000000000000111000000011100
00000111000111000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(28)<=
"00000000000000000000000000000000000000000000000000011100000000000000011100000011
11111111111111111000000111000000000000111000000111000000000000111000000011100
00000011111111000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(29)<=
"00000000000000000000000000000000000000000000000000011100000000000000011100000011
11111111111111111000000111000000000000111000000111000000000000111000000011100
```

```
00000001111111000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(30)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(31)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(32)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(33)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(34)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(35)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(36)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(37)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(38)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
game_over(39)<=
"00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000";
```

```
-------------------------------------------------------------------------------
-----------------------

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

  HCounter : process (clk25)
  begin
    if rising_edge(clk25) then
      if reset = '1' then
        Hcount <= (others => '0');
        elsif EndOfLine = '1' then
        Hcount <= (others => '0');
      else
        Hcount <= Hcount + 1;
      end if;
    end if;
  end process HCounter;

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

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

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

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

  HSyncGen : process (clk25)
  begin
    if rising_edge(clk25) then
      if reset = '1' or EndOfLine = '1' then
        vga_hsync <= '1';
      elsif Hcount = HSYNC - 1 then
        vga_hsync <= '0';
      end if;
    end if;
  end process HSyncGen;

  HBlankGen : process (clk25)
  begin
```

```
  if rising_edge(clk25) then
    if reset = '1' then
      vga_hblank <= '1';
    elsif Hcount = HSYNC + HBACK_PORCH then
      vga_hblank <= '0';
    elsif Hcount = HSYNC + HBACK_PORCH + HACTIVE then
      vga_hblank <= '1';
          wave_h_coord <= wave_h_coord_temp;
          home_h_coord <= home_h_coord_temp;

    end if;
  end if;
end process HBlankGen;

VSyncGen : process (clk25)
begin
  if rising_edge(clk25) then
    if reset = '1' then
      vga_vsync <= '1';
    elsif EndOfLine ='1' then
      if EndOfField = '1' then
        vga_vsync <= '1';
      elsif Vcount = VSYNC - 1 then
        vga_vsync <= '0';
      end if;
    end if;
  end if;
end process VSyncGen;

VBlankGen : process (clk25)
begin
  if rising_edge(clk25) then
    if reset = '1' then
      vga_vblank <= '1';
    elsif EndOfLine = '1' then
      if Vcount = VSYNC + VBACK_PORCH - 1 then
        vga_vblank <= '0';
      elsif Vcount = VSYNC + VBACK_PORCH + VACTIVE - 1 then
        vga_vblank <= '1';
        wave_v_coord <= wave_v_coord_temp;
        home_missile_x <= home_missile_x_temp;
        home_missile_y <= home_missile_y_temp;
            alien_missile_x(0) <= alien_missile_x_temp(0) ;
            alien_missile_y(0) <= alien_missile_y_temp(0) ;
            alien_missile_x(1) <= alien_missile_x_temp(1) ;
            alien_missile_y(1) <= alien_missile_y_temp(1) ;
            alien_missile_x(2) <= alien_missile_x_temp(2) ;
            alien_missile_y(2) <= alien_missile_y_temp(2) ;
            alien_missile_x(3) <= alien_missile_x_temp(3) ;
            alien_missile_y(3) <= alien_missile_y_temp(3) ;
            alien_missile_x(4) <= alien_missile_x_temp(4) ;
            alien_missile_y(4) <= alien_missile_y_temp(4) ;
            alien_missile_x(5) <= alien_missile_x_temp(5) ;
            alien_missile_y(5) <= alien_missile_y_temp(5) ;
            alien_missile_x(6) <= alien_missile_x_temp(6) ;
            alien_missile_y(6) <= alien_missile_y_temp(6) ;
            alien_missile_x(7) <= alien_missile_x_temp(7) ;
```

```
              alien_missile_y(7) <= alien_missile_y_temp(7) ;
              alien_missile_x(8) <= alien_missile_x_temp(8) ;
              alien_missile_y(8) <= alien_missile_y_temp(8) ;
              alien_missile_x(9) <= alien_missile_x_temp(9) ;
              alien_missile_y(9) <= alien_missile_y_temp(9) ;
        end if;
      end if;
    end if;
  end process VBlankGen;

-------------------------------------------------------------------------------
----------------------
DrawHomeShip : process (clk25 )

variable home_h_counter: integer := 0;
variable home_v_counter : integer := 0;
variable home_row : unsigned (31 downto 0);

begin
      if rising_edge(clk25) and game_state=2 then
        if home_status = '1' then
            if EndOfLine = '1' then
                  if ( home_v_counter > 0 ) then
                        home_v_counter := home_v_counter + 1;
                  end if;
                  if Vcount = VSYNC + VBACK_PORCH - 1 + HOME_VSTART then
                        home_v_counter := 1;
                  elsif Vcount = VSYNC + VBACK_PORCH - 1 + HOME_VSTART +
SHIP_LENGTH-1 then
                        home_v_counter := 0;
                  end if;

            end if;

            if ( home_h_counter > 0 ) then
                  home_h_counter := home_h_counter + 1;
            end if;

            if Hcount = HSYNC + HBACK_PORCH - 1 + home_h_coord then
                  home_h_counter := 1;
            elsif Hcount = HSYNC + HBACK_PORCH - 1 + home_h_coord +
SHIP_WIDTH - 1 then
                  home_h_counter := 0;
            end if;

            if (( home_h_counter > 0 ) and (home_v_counter > 0)) then
                  home_row := home_ship(home_v_counter-1);
                  home_bit <= home_row(SHIP_WIDTH-home_h_counter);
            else
                  home_bit <= '0';
            end if;
        end if;
      end if;
end process DrawHomeShip;

-------------------------------------------------------------------------------
----------------------
```

```
--------------------------------------------------------------------------------
----------------------
DrawAlien1 : process (clk25)
--variable count:integer:=0;
variable alien_h_counter: integer := 0;
variable alien_v_counter : integer := 0;
variable alien_row : unsigned (15 downto 0);
variable gap_h_counter: integer := 0;
variable gap_v_counter: integer := 0;
variable alien_id : integer := 0;
variable first_alien_inrow : integer := 0;
variable alive_bit : std_logic := '1';
variable anime_counter : integer := 0;
variable toggle : std_logic := '0';
variable exploded1 : std_logic_vector(31 downto 0);
variable exploded2 : std_logic_vector(31 downto 0);
variable explode_bit : std_logic := '0';
variable explode_counter : integer := 0;

begin
      if rising_edge(clk25) and game_state=2 then

            if ((Hcount = HSYNC + HBACK_PORCH - 1 + wave_h_coord)
                  and (Vcount = VSYNC + VBACK_PORCH - 1 + wave_v_coord)) then

                  alien_id := 0;
                  first_alien_inrow := 0;

            elsif ( (Hcount = HSYNC + HBACK_PORCH - 1 + wave_h_coord
                  + (NUM_ALIENS_PER_ROW*ALIEN_WIDTH) + ((NUM_ALIENS_PER_ROW-
1)*GAP_WIDTH))
                  and (Vcount = VSYNC + VBACK_PORCH - 1 + wave_v_coord
                  + ((first_alien_inrow+1)*ALIEN_LENGTH) +
((first_alien_inrow)*GAP_LENGTH) ))then

                  first_alien_inrow := first_alien_inrow + 1;
            end if;

            if EndOfLine = '1' then

                  if ( alien_v_counter > 0 ) then
                        alien_v_counter := alien_v_counter + 1;
                  end if;

                  if Vcount = VSYNC + VBACK_PORCH - 1 + wave_v_coord +
gap_v_counter then
                        alien_v_counter := 1;

                  elsif Vcount = VSYNC + VBACK_PORCH - 1 + wave_v_coord +
ALIEN_LENGTH-1+gap_v_counter then
                        if ( gap_v_counter < (ALIEN_LENGTH +
GAP_LENGTH)*(NUM_ALIEN_ROWS-1) ) then
                              gap_v_counter := gap_v_counter + ALIEN_LENGTH +
GAP_LENGTH;
                        else
```

```
                              gap_v_counter := 0;
                        end if;

                        alien_v_counter := 0;
                  end if;

                  alien_id := first_alien_inrow*NUM_ALIENS_PER_ROW;

            end if;

            if ( alien_h_counter > 0 ) then
                  alien_h_counter := alien_h_counter + 1;
            end if;

            if Hcount = HSYNC + HBACK_PORCH - 1 + wave_h_coord +
gap_h_counter then
                        alien_h_counter := 1;
            elsif Hcount = HSYNC + HBACK_PORCH - 1 + wave_h_coord +
ALIEN_WIDTH - 1 + gap_h_counter then

                  if ( gap_h_counter < (ALIEN_WIDTH +
GAP_WIDTH)*(NUM_ALIENS_PER_ROW-1) ) then
                        gap_h_counter := gap_h_counter + ALIEN_WIDTH +
GAP_WIDTH;
                  else
                        gap_h_counter := 0;
                  end if;
                  alien_h_counter := 0;

                  alien_id := alien_id + 1;
            end if;

            if ( alien_id < 32 ) then
                        alive_bit := alien_status_1(alien_id);
                        if alive_bit = '1' then
                              exploded1(alien_id) := '0';--reset
                        else
                              explode_bit := exploded1(alien_id);
                        end if;
            else
                        alive_bit := alien_status_2(alien_id - 31);
                        if alive_bit = '1' then
                              exploded2(alien_id) := '0';--reset
                        else
                              explode_bit := exploded2(alien_id);
                        end if;
            end if;

            if (( alien_h_counter > 0 ) and (alien_v_counter > 0)) then
                  if alive_bit = '1' then
                        if toggle = '1' then
                              alien_row := alien_ship(alien_v_counter-1);
                        else
                              alien_row := alien_ship2(alien_v_counter-1);
                        end if;
                        wave_bit <= alien_row(ALIEN_WIDTH-alien_h_counter);
```

```
                  elsif (not (alive_bit = '1')) and explode_bit = '0' then
                        if explode_counter < 1250 then
                              alien_row := alien_explode(alien_v_counter-1);
                              wave_bit <= alien_row(ALIEN_WIDTH-
alien_h_counter);
                              explode_counter := explode_counter+1;
                        else
                              if ( alien_id < 32 ) then
                                    exploded1(alien_id) := '1';--exploded
already
                              else
                                    exploded2(alien_id) := '1';
                              end if;
                              explode_counter := 0;--reset counter
                        end if;
                  else
                        wave_bit <= '0';
                  end if;
            else
                  wave_bit <= '0';
            end if;

            if anime_counter = 25000000 then
                  anime_counter := 0;
                  toggle := not toggle;
            else
                  anime_counter := anime_counter + 1;
            end if;
      end if;
end process DrawAlien1;

-------------------------------------------------------------------------------
------------------------------
DrawScoreText : process (clk25)
variable score_x:integer;
variable score_y:integer;
variable score_v_counter : integer := 0;
variable score_h_counter : integer := 0;
begin
if rising_edge(clk25) and game_state/=1 then

      if EndOfLine = '1' then
            if ( score_v_counter > 0 ) then
                  score_v_counter := score_v_counter + 1;
            end if;

            if Vcount = VSYNC + VBACK_PORCH - 1 + SCORE_V_COORD then
                  score_v_counter := 1;
            elsif Vcount = VSYNC + VBACK_PORCH - 1 + SCORE_V_COORD  +
TEXT_LENGTH-1 then
                  score_v_counter := 0;
            end if;

      end if;

      if ( score_h_counter > 0 ) then
            score_h_counter := score_h_counter + 1;
```

```
      end if;

      if Hcount = HSYNC + HBACK_PORCH - 1 + SCORE_H_COORD then
            score_h_counter := 1;
      elsif Hcount = HSYNC + HBACK_PORCH - 1 + SCORE_H_COORD + TEXT_WIDTH - 1
then
                  score_h_counter := 0;
      end if;

      if (( score_h_counter > 0 ) and (score_v_counter > 0)) then
            score_row <= score_text(score_v_counter-1);
            score_bit <= score_row(80-score_h_counter);
      else
            score_bit <= '0';
      end if;

end if;
end process DrawScoreText;
--------------------------------------------------------------------------------
------------------------------
DrawLivesText : process (clk25)
variable lives_x:integer;
variable lives_y:integer;
variable lives_v_counter : integer := 0;
variable lives_h_counter : integer := 0;

begin
if rising_edge(clk25) and game_state=2 then

      if EndOfLine = '1' then
            if ( lives_v_counter > 0 ) then
                  lives_v_counter := lives_v_counter + 1;
            end if;

            if Vcount = VSYNC + VBACK_PORCH - 1 + LIVES_V_COORD then
                  lives_v_counter := 1;
            elsif Vcount = VSYNC + VBACK_PORCH - 1 + LIVES_V_COORD  +
TEXT_LENGTH-1 then
                  lives_v_counter := 0;
            end if;

            end if;

            if ( lives_h_counter > 0 ) then
                  lives_h_counter := lives_h_counter + 1;
            end if;

            if Hcount = HSYNC + HBACK_PORCH - 1 + LIVES_H_COORD then
                  lives_h_counter := 1;
            elsif Hcount = HSYNC + HBACK_PORCH - 1 + LIVES_H_COORD +
TEXT_WIDTH - 1 then
                  lives_h_counter := 0;
            end if;

            if (( lives_h_counter > 0 ) and (lives_v_counter > 0)) then
                  lives_row <= lives_text(lives_v_counter-1);
                  lives_bit <= lives_row(80-lives_h_counter);
```

```
            else
                  lives_bit <= '0';
            end if;

      end if;
end process DrawLivesText;
--------------------------------------------------------------------------------
DrawTimerText : process (clk25)
variable timer_x:integer;
variable timer_y:integer;
variable timer_v_counter : integer := 0;
variable timer_h_counter : integer := 0;
begin
if rising_edge(clk25) and game_state/=1 then

      if EndOfLine = '1' then
                  if ( timer_v_counter > 0 ) then
                        timer_v_counter := timer_v_counter + 1;
                  end if;

                  if Vcount = VSYNC + VBACK_PORCH - 1 + TIMER_V_COORD then
                        timer_v_counter := 1;
                  elsif Vcount = VSYNC + VBACK_PORCH - 1 + TIMER_V_COORD  +
TEXT_LENGTH-1 then
                        timer_v_counter := 0;
                  end if;

      end if;

      if ( timer_h_counter > 0 ) then
                  timer_h_counter := timer_h_counter + 1;
      end if;

                  if Hcount = HSYNC + HBACK_PORCH - 1 + TIMER_H_COORD then
                        timer_h_counter := 1;
                  elsif Hcount = HSYNC + HBACK_PORCH - 1 + TIMER_H_COORD +
TEXT_WIDTH - 1 then
                        timer_h_counter := 0;
                  end if;

      if (( timer_h_counter > 0 ) and (timer_v_counter > 0)) then

                  timer_row <= timer_text(timer_v_counter-1);
                  timer_bit <= timer_row(80-timer_h_counter);
      else
                  timer_bit <= '0';
      end if;

end if;
end process DrawTimerText;
--------------------------------------------------------------------------------
----
DrawTimerVal_1 : process (clk25)
variable timer_x:integer;
variable timer_y:integer;
variable num:integer;
```

```
variable timer_h_counter_1:integer := 0;
variable timer_v_counter_1:integer := 0;
begin
if rising_edge(clk25) and  game_state/=1 then
      num:=timer1_num;


      if EndOfLine = '1' then
            if ( timer_v_counter_1 > 0 ) then
                  timer_v_counter_1 := timer_v_counter_1 + 1;
            end if;
            if Vcount = VSYNC + VBACK_PORCH - 1 + TIMER_V_COORD_1  then
                  timer_v_counter_1 := 1;
            elsif Vcount = VSYNC + VBACK_PORCH - 1 + TIMER_V_COORD_1  +
NUM_LENGTH-1 then
                  timer_v_counter_1 := 0;
            end if;

      end if;

    if ( timer_h_counter_1 > 0 ) then
                  timer_h_counter_1 := timer_h_counter_1 + 1;
      end if;

      if Hcount = HSYNC + HBACK_PORCH - 1 + TIMER_H_COORD_1 then
                  timer_h_counter_1 := 1;
    elsif Hcount = HSYNC + HBACK_PORCH - 1 + TIMER_H_COORD_1 + NUM_WIDTH - 1
then
                  timer_h_counter_1 := 0;
      end if;

      if (( timer_h_counter_1 > 0 ) and (timer_v_counter_1 > 0)) then
                  if(num = 0) then
                  timer_row_1 <= sprite_0(timer_v_counter_1-1);
                  timer_bit_1 <= timer_row_1(80-timer_h_counter_1);
                  elsif(num = 1) then
                  timer_row_1 <= sprite_1(timer_v_counter_1-1);
                  timer_bit_1 <= timer_row_1(80-timer_h_counter_1);
                  elsif(num = 2) then
                  timer_row_1 <= sprite_2(timer_v_counter_1-1);
                  timer_bit_1 <= timer_row_1(80-timer_h_counter_1);
                  elsif(num = 3) then
                  timer_row_1 <= sprite_3(timer_v_counter_1-1);
                  timer_bit_1 <= timer_row_1(80-timer_h_counter_1);
                  elsif(num = 4) then
                  timer_row_1 <= sprite_4(timer_v_counter_1-1);
                  timer_bit_1 <= timer_row_1(80-timer_h_counter_1);
                  elsif(num = 5) then
                  timer_row_1 <= sprite_5(timer_v_counter_1-1);
                  timer_bit_1 <= timer_row_1(80-timer_h_counter_1);
                  elsif(num = 6) then
                  timer_row_1 <= sprite_6(timer_v_counter_1-1);
                  timer_bit_1 <= timer_row_1(80-timer_h_counter_1);
                  elsif(num = 7) then
                  timer_row_1 <= sprite_7(timer_v_counter_1-1);
                  timer_bit_1 <= timer_row_1(80-timer_h_counter_1);
                  elsif(num = 8) then
```

```
                  timer_row_1 <= sprite_8(timer_v_counter_1-1);
                  timer_bit_1 <= timer_row_1(80-timer_h_counter_1);
                  elsif(num = 9) then
                  timer_row_1 <= sprite_9(timer_v_counter_1-1);
                  timer_bit_1 <= timer_row_1(80-timer_h_counter_1);
                  end if;
      else
                  timer_bit_1 <= '0';
      end if;

end if;
end process DrawTimerVal_1;
--------------------------------------------------------------------------------
------
DrawTimerVal_2 : process (clk25)
variable timer_x:integer;
variable timer_y:integer;
variable num:integer;
variable timer_h_counter_2:integer := 0;
variable timer_v_counter_2:integer := 0;
begin
      if rising_edge(clk25) and  game_state/=1 then
            num:=timer2_num;


            if EndOfLine = '1' then
                  if ( timer_v_counter_2 > 0 ) then
                        timer_v_counter_2 := timer_v_counter_2 + 1;
                  end if;
                  if Vcount = VSYNC + VBACK_PORCH - 1 + TIMER_V_COORD_2  then
                        timer_v_counter_2 := 1;
                  elsif Vcount = VSYNC + VBACK_PORCH - 1 + TIMER_V_COORD_2  +
NUM_LENGTH-1 then
                        timer_v_counter_2 := 0;
                  end if;

            end if;

            if ( timer_h_counter_2 > 0 ) then
                        timer_h_counter_2 := timer_h_counter_2 + 1;
            end if;

            if Hcount = HSYNC + HBACK_PORCH - 1 + TIMER_H_COORD_2 then
                        timer_h_counter_2 := 1;
            elsif Hcount = HSYNC + HBACK_PORCH - 1 + TIMER_H_COORD_2 +
NUM_WIDTH - 1 then
                        timer_h_counter_2 := 0;
            end if;

      if (( timer_h_counter_2 > 0 ) and (timer_v_counter_2 > 0)) then

            if(num = 0) then
                        timer_row_2 <= sprite_0(timer_v_counter_2-1);
                        timer_bit_2 <= timer_row_2(80-timer_h_counter_2);
            elsif(num = 1) then
                        timer_row_2 <= sprite_1(timer_v_counter_2-1);
                        timer_bit_2 <= timer_row_2(80-timer_h_counter_2);
```

```
            elsif(num = 2) then
                        timer_row_2 <= sprite_2(timer_v_counter_2-1);
                        timer_bit_2 <= timer_row_2(80-timer_h_counter_2);
            elsif(num = 3) then
                        timer_row_2 <= sprite_3(timer_v_counter_2-1);
                        timer_bit_2 <= timer_row_2(80-timer_h_counter_2);
            elsif(num = 4) then
                        timer_row_2 <= sprite_4(timer_v_counter_2-1);
                        timer_bit_2 <= timer_row_2(80-timer_h_counter_2);
            elsif(num = 5) then
                        timer_row_2 <= sprite_5(timer_v_counter_2-1);
                        timer_bit_2 <= timer_row_2(80-timer_h_counter_2);
            elsif(num = 6) then
                        timer_row_2 <= sprite_6(timer_v_counter_2-1);
                        timer_bit_2 <= timer_row_2(80-timer_h_counter_2);
            elsif(num = 7) then
                        timer_row_2 <= sprite_7(timer_v_counter_2-1);
                        timer_bit_2 <= timer_row_2(80-timer_h_counter_2);
            elsif(num = 8) then
                        timer_row_2 <= sprite_8(timer_v_counter_2-1);
                        timer_bit_2 <= timer_row_2(80-timer_h_counter_2);
            elsif(num = 9) then
                        timer_row_2 <= sprite_9(timer_v_counter_2-1);
                        timer_bit_2 <= timer_row_2(80-timer_h_counter_2);
            end if;

            else
                        timer_bit_2 <= '0';
      end if;

end if;
end process DrawTimerVal_2;

--------------------------------------------------------------------------
------
DrawLivesVal_1 : process (clk25)
variable lives_x:integer;
variable lives_y:integer;
variable num:integer;
variable lives_h_counter_1:integer := 0;
variable lives_v_counter_1:integer := 0;
begin
if rising_edge(clk25) and game_state=2 then
            num:=lives1_num;

            if ( lives1_num > 0) then
                  if EndOfLine = '1' then
                        if ( lives_v_counter_1 > 0 ) then
                              lives_v_counter_1 := lives_v_counter_1 + 1;
                        end if;
                        if Vcount = VSYNC + VBACK_PORCH - 1 + LIVES_V_COORD_1
then
                              lives_v_counter_1 := 1;
                        elsif Vcount = VSYNC + VBACK_PORCH - 1 +
LIVES_V_COORD_1  + NUM_LENGTH-1 then
                              lives_v_counter_1 := 0;
                        end if;
```

```
                  end if;

                        if ( lives_h_counter_1 > 0 ) then
                              lives_h_counter_1 := lives_h_counter_1 + 1;
                        end if;

                        if Hcount = HSYNC + HBACK_PORCH - 1 + LIVES_H_COORD_1
then
                              lives_h_counter_1 := 1;
                        elsif Hcount = HSYNC + HBACK_PORCH - 1 +
LIVES_H_COORD_1 + NUM_WIDTH - 1 then
                              lives_h_counter_1 := 0;
                        end if;

                  if (( lives_h_counter_1 > 0 ) and (lives_v_counter_1 > 0))
then
                              lives_row_1 <= lives_sprite(lives_v_counter_1-
1);
                              lives_bit_1 <= lives_row_1(80 -
lives_h_counter_1);
                  else
                              lives_bit_1 <= '0';
                  end if;
            end if;

end if;
end process DrawLivesVal_1;
--------------------------------------------------------------------------
------
DrawLivesVal_2 : process (clk25)
variable lives_x:integer;
variable lives_y:integer;
variable num:integer;
variable lives_h_counter_2:integer := 0;
variable lives_v_counter_2:integer := 0;
begin
if rising_edge(clk25) and game_state=2 then
      num:=lives2_num;

      if ( lives1_num > 1) then
            if EndOfLine = '1' then
                  if ( lives_v_counter_2 > 0 ) then
                        lives_v_counter_2 := lives_v_counter_2 + 1;
                  end if;
                  if Vcount = VSYNC + VBACK_PORCH - 1 + LIVES_V_COORD_2  then
                        lives_v_counter_2 := 1;
                  elsif Vcount = VSYNC + VBACK_PORCH - 1 + LIVES_V_COORD_2  +
NUM_LENGTH-1 then
                        lives_v_counter_2 := 0;
                  end if;

            end if;

                  if ( lives_h_counter_2 > 0 ) then
                        lives_h_counter_2 := lives_h_counter_2 + 1;
                  end if;
```

```
            if Hcount = HSYNC + HBACK_PORCH - 1 + LIVES_H_COORD_2 then
                        lives_h_counter_2 := 1;
            elsif Hcount = HSYNC + HBACK_PORCH - 1 + LIVES_H_COORD_2 +
NUM_WIDTH - 1 then
                        lives_h_counter_2 := 0;
            end if;

            if (( lives_h_counter_2 > 0 ) and (lives_v_counter_2 > 0)) then
                  lives_row_2 <= lives_sprite(lives_v_counter_2-1);
                  lives_bit_2 <= lives_row_2(80 - lives_h_counter_2);
            else
                  lives_bit_2 <= '0';
            end if;
      end if;

end if;
end process DrawLivesVal_2;
--------------------------------------------------------------------------
------
DrawLivesVal_3 : process (clk25)
variable lives_x:integer;
variable lives_y:integer;
variable num:integer;
variable lives_h_counter_3:integer := 0;
variable lives_v_counter_3:integer := 0;
begin
if rising_edge(clk25) and game_state=2 then
                  num:=lives3_num;

                  if ( lives1_num > 2 ) then
                        if EndOfLine = '1' then
                              if ( lives_v_counter_3 > 0 ) then
                                    lives_v_counter_3 := lives_v_counter_3 +
1;
                              end if;
                              if Vcount = VSYNC + VBACK_PORCH - 1 +
LIVES_V_COORD_3  then
                                    lives_v_counter_3 := 1;
                              elsif Vcount = VSYNC + VBACK_PORCH - 1 +
LIVES_V_COORD_3  + NUM_LENGTH-1 then
                                    lives_v_counter_3 := 0;
                              end if;

                        end if;

                        if ( lives_h_counter_3 > 0 ) then
                              lives_h_counter_3 := lives_h_counter_3 + 1;
                        end if;

                        if Hcount = HSYNC + HBACK_PORCH - 1 + LIVES_H_COORD_3
then
                              lives_h_counter_3 := 1;
                        elsif Hcount = HSYNC + HBACK_PORCH - 1 +
LIVES_H_COORD_3 + NUM_WIDTH - 1 then
                              lives_h_counter_3 := 0;
                        end if;
```

```
                        if (( lives_h_counter_3 > 0 ) and (lives_v_counter_3
> 0)) then
                              lives_row_3 <= lives_sprite(lives_v_counter_3-
1);
                              lives_bit_3 <= lives_row_3(80 -
lives_h_counter_3);
                        else
                              lives_bit_3 <= '0';
                        end if;
                  end if;

end if;
end process DrawLivesVal_3;


--------------------------------------------------------------------------
----
DrawBarricade1 : process (clk25 )
variable barricade_x:integer;
variable barricade_y:integer;
variable barricade1_v_counter : integer := 0;
variable barricade1_h_counter : integer := 0;
variable barricade1_row : unsigned(31 downto 0);
variable vert_block : integer := 0;
variable barricade_block_id : integer := 0;
variable active_bit : std_logic := '1';
begin
      if rising_edge(clk25) and game_state=2 then

            if EndOfLine = '1' then
                  if ( barricade1_v_counter > 0 ) then
                        barricade1_v_counter := barricade1_v_counter + 1;
                  end if;
                  if Vcount = VSYNC + VBACK_PORCH - 1 + BARRICADE1_V_COORD
then
                        barricade1_v_counter := 1;
                  elsif Vcount = VSYNC + VBACK_PORCH - 1 + BARRICADE1_V_COORD
+ BARRICADE_LENGTH-1 then
                        barricade1_v_counter := 0;
                        vert_block := 0;
                  end if;
            end if;

            if ( barricade1_h_counter > 0 ) then
                  barricade1_h_counter := barricade1_h_counter + 1;
            end if;

            if Hcount = HSYNC + HBACK_PORCH - 1 + BARRICADE1_H_COORD then
                        barricade1_h_counter := 1;
            elsif Hcount = HSYNC + HBACK_PORCH - 1 + BARRICADE1_H_COORD +
BARRICADE_WIDTH - 1 then
                        barricade1_h_counter := 0;

                              if ( Vcount = VSYNC + VBACK_PORCH - 1 +
BARRICADE1_V_COORD + (vert_block + 1) * BARRICADE_BLOCK_LENGTH ) then
                                    vert_block := vert_block + 1;
```

```
                              end if;
                              barricade_block_id := vert_block *
NUM_H_BLOCKS_BARRICADE;
            end if;

            if Hcount = HSYNC + HBACK_PORCH - 1 + BARRICADE1_H_COORD +
(barricade_block_id + 1 -
(vert_block*NUM_H_BLOCKS_BARRICADE))*BARRICADE_BLOCK_WIDTH then
                  barricade_block_id := barricade_block_id + 1;
            end if;

            active_bit := barricade_status(0)(barricade_block_id);

            if (( barricade1_h_counter > 0 ) and (barricade1_v_counter > 0)
and active_bit = '1') then

                  barricade1_row := home_ship_barricade(barricade1_v_counter-
1);
                  barricade1_bit <= barricade1_row(32-barricade1_h_counter);

            else
                  barricade1_bit <= '0';
            end if;

      end if;
end process DrawBarricade1;

---------------------------------
-------------------------------------------------------------------------------
---------------
DrawBarricade2 : process (clk25 )
variable barricade_x:integer;
variable barricade_y:integer;
variable barricade2_v_counter : integer := 0;
variable barricade2_h_counter : integer := 0;
variable barricade2_row : unsigned(31 downto 0);
variable vert_block : integer := 0;
variable barricade_block_id : integer := 0;
variable active_bit : std_logic := '1';
begin
      if rising_edge(clk25) and game_state=2 then
            if EndOfLine = '1' then
                  if ( barricade2_v_counter > 0 ) then
                        barricade2_v_counter := barricade2_v_counter + 1;
                  end if;
                  if Vcount = VSYNC + VBACK_PORCH - 1 + BARRICADE2_V_COORD
then
                        barricade2_v_counter := 1;
                  elsif Vcount = VSYNC + VBACK_PORCH - 1 + BARRICADE2_V_COORD
+ BARRICADE_LENGTH-1 then
                        barricade2_v_counter := 0;
                        vert_block := 0;
                  end if;
            end if;

            if ( barricade2_h_counter > 0 ) then
                  barricade2_h_counter := barricade2_h_counter + 1;
```

```
            end if;

            if Hcount = HSYNC + HBACK_PORCH - 1 + BARRICADE2_H_COORD then
                  barricade2_h_counter := 1;
            elsif Hcount = HSYNC + HBACK_PORCH - 1 + BARRICADE2_H_COORD +
BARRICADE_WIDTH - 1 then
                  barricade2_h_counter := 0;
                  if ( Vcount = VSYNC + VBACK_PORCH - 1 + BARRICADE2_V_COORD
+ (vert_block + 1) * BARRICADE_BLOCK_LENGTH ) then
                                    vert_block := vert_block + 1;
                  end if;
                  barricade_block_id := vert_block * NUM_H_BLOCKS_BARRICADE;
            end if;


            if Hcount = HSYNC + HBACK_PORCH - 1 + BARRICADE2_H_COORD +
(barricade_block_id + 1 -
(vert_block*NUM_H_BLOCKS_BARRICADE))*BARRICADE_BLOCK_WIDTH then
                  barricade_block_id := barricade_block_id + 1;
            end if;

            active_bit := barricade_status(1)(barricade_block_id);

            if (( barricade2_h_counter > 0 ) and (barricade2_v_counter > 0)
and active_bit = '1') then
                  barricade2_row := home_ship_barricade(barricade2_v_counter-
1);
                  barricade2_bit <= barricade2_row(32-barricade2_h_counter);
            else
                  barricade2_bit <= '0';
            end if;

      end if;
end process DrawBarricade2;
--------------------------------------------------------------------------
--------------------
 DrawBarricade3 : process (clk25 )

variable barricade_x:integer;
variable barricade_y:integer;
variable barricade3_v_counter : integer := 0;
variable barricade3_h_counter : integer := 0;
variable barricade3_row : unsigned(31 downto 0);
variable vert_block : integer := 0;
variable barricade_block_id : integer := 0;
variable active_bit : std_logic := '1';
begin
      if rising_edge(clk25) and game_state=2 then
            if EndOfLine = '1' then
                  if ( barricade3_v_counter > 0 ) then
                        barricade3_v_counter := barricade3_v_counter + 1;
                  end if;
                  if Vcount = VSYNC + VBACK_PORCH - 1 + BARRICADE3_V_COORD
then
                        barricade3_v_counter := 1;
                  elsif Vcount = VSYNC + VBACK_PORCH - 1 + BARRICADE3_V_COORD
+ BARRICADE_LENGTH-1 then
```

```
                        barricade3_v_counter := 0;
                        vert_block := 0;
                  end if;
            end if;

            if ( barricade3_h_counter > 0 ) then
                  barricade3_h_counter := barricade3_h_counter + 1;
            end if;

            if Hcount = HSYNC + HBACK_PORCH - 1 + BARRICADE3_H_COORD then
                        barricade3_h_counter := 1;
            elsif Hcount = HSYNC + HBACK_PORCH - 1 + BARRICADE3_H_COORD +
BARRICADE_WIDTH - 1 then
                        barricade3_h_counter := 0;
                        if ( Vcount = VSYNC + VBACK_PORCH - 1 +
BARRICADE3_V_COORD + (vert_block + 1) * BARRICADE_BLOCK_LENGTH ) then
                                    vert_block := vert_block + 1;
                        end if;
                        barricade_block_id := vert_block *
NUM_H_BLOCKS_BARRICADE;
            end if;

            if Hcount = HSYNC + HBACK_PORCH - 1 + BARRICADE3_H_COORD +
(barricade_block_id + 1 -
(vert_block*NUM_H_BLOCKS_BARRICADE))*BARRICADE_BLOCK_WIDTH then
                  barricade_block_id := barricade_block_id + 1;
            end if;

            active_bit := barricade_status(2)(barricade_block_id);

            if (( barricade3_h_counter > 0 ) and (barricade3_v_counter > 0)
and active_bit = '1') then
                  barricade3_row := home_ship_barricade(barricade3_v_counter-
1);
                  barricade3_bit <= barricade3_row(32-barricade3_h_counter);
            else
                  barricade3_bit <= '0';
            end if;

      end if;
end process DrawBarricade3;
--------------------------------------------------------------------------
---------------
 DrawBarricade4 : process (clk25 )

variable barricade_x:integer;
variable barricade_y:integer;
variable barricade4_v_counter : integer := 0;
variable barricade4_h_counter : integer := 0;
variable barricade4_row : unsigned(31 downto 0);
variable vert_block : integer := 0;
variable barricade_block_id : integer := 0;
variable active_bit : std_logic := '1';
begin
      if rising_edge(clk25) and game_state=2 then
            if EndOfLine = '1' then
                  if ( barricade4_v_counter > 0 ) then
```

```
                        barricade4_v_counter := barricade4_v_counter + 1;
                  end if;
                  if Vcount = VSYNC + VBACK_PORCH - 1 + BARRICADE4_V_COORD
then
                        barricade4_v_counter := 1;
                  elsif Vcount = VSYNC + VBACK_PORCH - 1 + BARRICADE4_V_COORD
+ BARRICADE_LENGTH-1 then
                        barricade4_v_counter := 0;
                        vert_block := 0;
                  end if;
            end if;

            if ( barricade4_h_counter > 0 ) then
                  barricade4_h_counter := barricade4_h_counter + 1;
            end if;

            if Hcount = HSYNC + HBACK_PORCH - 1 + BARRICADE4_H_COORD then
                  barricade4_h_counter := 1;
            elsif Hcount = HSYNC + HBACK_PORCH - 1 + BARRICADE4_H_COORD +
BARRICADE_WIDTH - 1 then
                  barricade4_h_counter := 0;
                  if ( Vcount = VSYNC + VBACK_PORCH - 1 + BARRICADE4_V_COORD
+ (vert_block + 1) * BARRICADE_BLOCK_LENGTH ) then
                                    vert_block := vert_block + 1;
                  end if;
                  barricade_block_id := vert_block * NUM_H_BLOCKS_BARRICADE;
            end if;

            if Hcount = HSYNC + HBACK_PORCH - 1 + BARRICADE4_H_COORD +
(barricade_block_id + 1 -
(vert_block*NUM_H_BLOCKS_BARRICADE))*BARRICADE_BLOCK_WIDTH then
                  barricade_block_id := barricade_block_id + 1;
            end if;

            active_bit := barricade_status(3)(barricade_block_id);

            if (( barricade4_h_counter > 0 ) and (barricade4_v_counter > 0)
and active_bit = '1') then
                  barricade4_row := home_ship_barricade(barricade4_v_counter-
1);
                  barricade4_bit <= barricade4_row(32-barricade4_h_counter);
            else
                  barricade4_bit <= '0';
            end if;

      end if;
end process DrawBarricade4;
-------------------------------------

--------------------------------------------------------------------------------
----------------------
DrawIntro : process (clk25)
--variable intro:integer;
variable intro_y:integer;
variable intro_v_counter : integer := 0;
variable intro_h_counter : integer := 0;
begin
```

```
            if rising_edge(clk25) and game_state=1 then

                  if EndOfLine = '1' then
                        if ( intro_v_counter > 0 ) then
                              intro_v_counter := intro_v_counter + 1;
                        end if;

                        if Vcount = VSYNC + VBACK_PORCH - 1 + INTRO_V_COORD
then
                              intro_v_counter := 1;
                        elsif Vcount = VSYNC + VBACK_PORCH - 1 +
INTRO_V_COORD  + 86 then
                              intro_v_counter := 0;
                        end if;

                  end if;

                        if ( intro_h_counter > 0 ) then
                              intro_h_counter := intro_h_counter + 1;
                        end if;

                        if Hcount = HSYNC + HBACK_PORCH - 1 + INTRO_H_COORD
then
                              intro_h_counter := 1;
                        elsif Hcount = HSYNC + HBACK_PORCH - 1 +
INTRO_H_COORD + 299 then
                              intro_h_counter := 0;
                        end if;

                        if (( intro_h_counter > 0 ) and (intro_v_counter >
0)) then
                              intro_row <= intro(intro_v_counter-1);
                              intro_bit <= intro_row(300-intro_h_counter);
                        else
                              intro_bit <= '0';
                        end if;

            end if;

end process DrawIntro;

--------------------------------------------------------------------------
-----------------------
DrawGameOver : process (clk25)

variable game_over_v_counter : integer := 0;
variable game_over_h_counter : integer := 0;
begin
      if rising_edge(clk25) and game_state=3 then

            if EndOfLine = '1' then
                  if ( game_over_v_counter > 0 ) then
                        game_over_v_counter := game_over_v_counter + 1;
                  end if;

            if Vcount = VSYNC + VBACK_PORCH - 1 + game_over_V_COORD then
                  game_over_v_counter := 1;
```

```
            elsif Vcount = VSYNC + VBACK_PORCH - 1 + game_over_V_COORD  + 39
then
                  game_over_v_counter := 0;
            end if;

            end if;

            if ( game_over_h_counter > 0 ) then
                  game_over_h_counter := game_over_h_counter + 1;
            end if;

            if Hcount = HSYNC + HBACK_PORCH - 1 + game_over_H_COORD then
                  game_over_h_counter := 1;
            elsif Hcount = HSYNC + HBACK_PORCH - 1 + game_over_H_COORD + 299
then
                  game_over_h_counter := 0;
            end if;

            if (( game_over_h_counter > 0 ) and (game_over_v_counter > 0))
then
                  game_over_row <= game_over(game_over_v_counter-1);
                  game_over_bit <= game_over_row(300-game_over_h_counter);
            else
                  game_over_bit <= '0';
            end if;

      end if;
end process DrawGameOver;


--------------------------------------------------------------------------------
----------------------
DrawScore1 : process (clk25)
variable score_h_counter: integer := 0;
variable score_v_counter : integer := 0;
variable score_row_1 : unsigned (15 downto 0);
variable gap_h_counter: integer := 0;
variable score_id : integer := 0;
variable num : integer := 0;
begin
      if rising_edge(clk25) and game_state/=1 then

            if EndOfLine = '1' then

                  if ( score_v_counter > 0 ) then
                        score_v_counter := score_v_counter + 1;
                  end if;

                  -----
                  if Vcount = VSYNC + VBACK_PORCH - 1 + SCORE_V_COORD_1  then
                        score_v_counter := 1;
                  elsif  Vcount = VSYNC + VBACK_PORCH - 1 + SCORE_V_COORD_1 +
SCORE_LENGTH then
                        score_v_counter := 0;
                  end if;

                  score_id := 0;
```

```
            end if;

            if ( score_h_counter > 0 ) then
                  score_h_counter := score_h_counter + 1;
            end if;

            if Hcount = HSYNC + HBACK_PORCH - 1 + SCORE_H_COORD_1 +
gap_h_counter then
                        score_h_counter := 1;
            elsif Hcount = HSYNC + HBACK_PORCH - 1 + SCORE_H_COORD_1 +
SCORE_WIDTH - 1 + gap_h_counter then

                  ----------------------------
                  if ( gap_h_counter < (SCORE_WIDTH + SCORE_GAP)*(NUM_DIGITS-
1) ) then
                        gap_h_counter := gap_h_counter + SCORE_WIDTH +
SCORE_GAP;
                  else
                        gap_h_counter := 0;
                  end if;
                  score_h_counter := 0;

                  score_id := score_id + 1;
            end if;

            num := score_array(score_id);

            if (( score_h_counter > 0 ) and (score_v_counter > 0)) then
            ------------------------
                  if(num = 0) then
                        score_row_1 := sprite_0(score_v_counter-1);

                  elsif(num = 1) then
                        score_row_1 := sprite_1(score_v_counter-1);

                  elsif(num = 2) then
                        score_row_1 := sprite_2(score_v_counter-1);

                  elsif(num = 3) then
                        score_row_1 := sprite_3(score_v_counter-1);

                  elsif(num = 4) then
                        score_row_1 := sprite_4(score_v_counter-1);

                  elsif(num = 5) then
                        score_row_1 := sprite_5(score_v_counter-1);

                  elsif(num = 6) then
                        score_row_1 := sprite_6(score_v_counter-1);

                  elsif(num = 7) then
                        score_row_1 := sprite_7(score_v_counter-1);

                  elsif(num = 8) then
                        score_row_1 := sprite_8(score_v_counter-1);
```

```
                  elsif(num = 9) then
                        score_row_1 := sprite_9(score_v_counter-1);

                  end if;

                  score_bit_1 <= score_row_1(SCORE_WIDTH-score_h_counter);

            else
                  score_bit_1 <= '0';
            end if;
      end if;
end process DrawScore1;

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


DrawHomeMissile : process (clk25 )


      variable homemissile_v_counter : integer := 0;
      variable homemissile_h_counter : integer := 0;

      begin
            if rising_edge(clk25) and game_state=2 then

                  if ( home_missile_y > 0 ) then

                        if EndOfLine = '1' then
                              if ( homemissile_v_counter > 0 ) then
                                    homemissile_v_counter :=
homemissile_v_counter + 1;
                              end if;
                              if Vcount = VSYNC + VBACK_PORCH - 1 +
home_missile_y then
                                    homemissile_v_counter := 1;
                              elsif Vcount > VSYNC + VBACK_PORCH - 1 +
home_missile_y + HOME_MISSILE_LENGTH-1 then
                                    homemissile_v_counter := 0;
                              end if;
                        end if;

                        if ( homemissile_h_counter > 0 ) then
                              homemissile_h_counter := homemissile_h_counter
+ 1;
                        end if;

                        if Hcount = HSYNC + HBACK_PORCH - 1 + home_missile_x
then
                              homemissile_h_counter := 1;
                        elsif Hcount > HSYNC + HBACK_PORCH - 1 +
home_missile_x + HOME_MISSILE_WIDTH - 1 then
                              homemissile_h_counter := 0;
                        end if;

                        if ((homemissile_h_counter > 0 ) and
(homemissile_v_counter > 0)) then
```

```
                                    homemissile_bit <= '1';
                        else
                                    homemissile_bit <= '0';
                        end if;
            else
                        homemissile_bit <= '0';
            end if; -- home_missile_y > 0
      end if;
end process DrawHomeMissile;
-------------------------------------------------


process(clk25)
      variable alienmissile_v_counter : alien_missiles;
      variable alienmissile_h_counter : alien_missiles;

      begin
            if rising_edge(clk25) and game_state=2 then
                  for i in 0 to 9 loop
                  if ( alien_missile_y(i) > 0 ) then

                        if EndOfLine = '1' then
                              if ( alienmissile_v_counter(i) > 0 ) then
                                    alienmissile_v_counter(i) :=
alienmissile_v_counter(i) + 1;
                              end if;
                              if Vcount = VSYNC + VBACK_PORCH - 1 +
alien_missile_y(i) then
                                    alienmissile_v_counter(i) := 1;
                              elsif Vcount > VSYNC + VBACK_PORCH - 1 +
alien_missile_y(i) + HOME_MISSILE_LENGTH-1 then
                                    alienmissile_v_counter(i) := 0;
                              end if;
                        end if;

                        if ( alienmissile_h_counter(i) > 0 ) then
                              alienmissile_h_counter(i) :=
alienmissile_h_counter(i) + 1;
                        end if;

                        if Hcount = HSYNC + HBACK_PORCH - 1 +
alien_missile_x(i) then
                              alienmissile_h_counter(i) := 1;
                        elsif Hcount > HSYNC + HBACK_PORCH - 1 +
alien_missile_x(i) + HOME_MISSILE_WIDTH - 1 then
                              alienmissile_h_counter(i) := 0;
                        end if;

                        if ((alienmissile_h_counter(i) > 0 ) and
(alienmissile_v_counter(i) > 0)) then
                                    alienmissile_bits(i) <= '1';
                        else
                                    alienmissile_bits(i) <= '0';
                        end if;
            else
                        alienmissile_bits(i) <= '0';
            end if; -- home_missile_y > 0
```

```
             end loop;

      end if;

end process;
-------------------------------------------
--------------------------------------------------------------------------------
-------------------- .
DrawStar1 : process (clk25 )
variable star_x:integer;
variable star_y:integer;
variable star1_v_counter : integer := 0;
variable star1_h_counter : integer := 0;
variable star1_row : unsigned(15 downto 0);
variable star_toggle1: std_logic := '0';
variable star_counter1: integer := 0;

begin
if rising_edge(clk25) then
      if EndOfLine = '1' then
            if ( star1_v_counter > 0 ) then
                  star1_v_counter := star1_v_counter + 1;
            end if;

            if Vcount = VSYNC + VBACK_PORCH - 1 + STAR1_V_COORD then
                  star1_v_counter := 1;
            elsif Vcount = VSYNC + VBACK_PORCH - 1 + STAR1_V_COORD +
STAR_LENGTH-1 then
                  star1_v_counter := 0;
            end if;
      end if;

    if ( star1_h_counter > 0 ) then
            star1_h_counter := star1_h_counter + 1;
      end if;

      if Hcount = HSYNC + HBACK_PORCH - 1 + STAR1_H_COORD then
          star1_h_counter := 1;
    elsif Hcount = HSYNC + HBACK_PORCH - 1 + STAR1_H_COORD + STAR_WIDTH - 1
then
          star1_h_counter := 0;
      end if;

      if (( star1_h_counter > 0 ) and (star1_v_counter > 0)) then
            if star_toggle1 = '1' then
                  star1_row := star_sprite(star1_v_counter-1);
            else
                  star1_row := star_sprite2(star1_v_counter-1);
      end if;
star1_bit <= star1_row(16-star1_h_counter);

else
      star1_bit <= '0';
end if;
      if star_counter1 = 10000000 then
            star_counter1 := 0;
            star_toggle1 := not star_toggle1;
```

```
      else
            star_counter1 := star_counter1 + 1;
      end if;

end if;
end process DrawStar1;
-------------------------------------------------------------------------------
-----------------------
Drawstar2 : process (clk25 )
variable star_x:integer;
variable star_y:integer;
variable star2_v_counter : integer := 0;
variable star2_h_counter : integer := 0;
variable star2_row : unsigned(15 downto 0);
variable star_toggle2: std_logic := '0';
variable star_counter2: integer := 0;

begin
if rising_edge(clk25) then
      if EndOfLine = '1' then
            if ( star2_v_counter > 0 ) then
                  star2_v_counter := star2_v_counter + 1;
            end if;
            if Vcount = VSYNC + VBACK_PORCH - 1 + STAR2_V_COORD then
                  star2_v_counter := 1;
            elsif Vcount = VSYNC + VBACK_PORCH - 1 + STAR2_V_COORD +
STAR_LENGTH-1 then
            star2_v_counter := 0;
      end if;
end if;

if ( star2_h_counter > 0 ) then
      star2_h_counter := star2_h_counter + 1;
end if;

if Hcount = HSYNC + HBACK_PORCH - 1 + STAR2_H_COORD then
    star2_h_counter := 1;
elsif Hcount = HSYNC + HBACK_PORCH - 1 + STAR2_H_COORD + STAR_WIDTH - 1 then
    star2_h_counter := 0;
end if;

if (( star2_h_counter > 0 ) and (star2_v_counter > 0)) then
      if star_toggle2 = '1' then
            star2_row := star_sprite(star2_v_counter-1);
      else
star2_row := star_sprite2(star2_v_counter-1);
end if;

star2_bit <= star2_row(16-star2_h_counter);
else
      star2_bit <= '0';
end if;
if star_counter2 = 13000000 then
      star_counter2 := 0;
      star_toggle2 := not star_toggle2;
else
      star_counter2 := star_counter2 + 1;
```

```
end if;

end if;
end process Drawstar2;
-------------------------------------------------------------------------------
-----------------------
Drawstar3 : process (clk25 )
variable star_x:integer;
variable star_y:integer;
variable star3_v_counter : integer := 0;
variable star3_h_counter : integer := 0;
variable star3_row : unsigned(15 downto 0);
variable star_toggle3: std_logic := '0';
variable star_counter3: integer := 0;
begin
if rising_edge(clk25) then



if EndOfLine = '1' then
      if ( star3_v_counter > 0 ) then
            star3_v_counter := star3_v_counter + 1;
      end if;
      if Vcount = VSYNC + VBACK_PORCH - 1 + STAR3_V_COORD then
            star3_v_counter := 1;
      elsif Vcount = VSYNC + VBACK_PORCH - 1 + STAR3_V_COORD + STAR_LENGTH-1
then
            star3_v_counter := 0;
      end if;
end if;

if ( star3_h_counter > 0 ) then
      star3_h_counter := star3_h_counter + 1;
end if;

if Hcount = HSYNC + HBACK_PORCH - 1 + STAR3_H_COORD then
    star3_h_counter := 1;
elsif Hcount = HSYNC + HBACK_PORCH - 1 + STAR3_H_COORD + STAR_WIDTH - 1 then
    star3_h_counter := 0;
end if;

if (( star3_h_counter > 0 ) and (star3_v_counter > 0)) then
      if star_toggle3 = '1' then
            star3_row := star_sprite(star3_v_counter-1);
      else
      star3_row := star_sprite2(star3_v_counter-1);
end if;

star3_bit <= star3_row(16-star3_h_counter);
else
star3_bit <= '0';
end if;
if star_counter3 = 16000000 then
      star_counter3 := 0;
      star_toggle3 := not star_toggle3;
else
      star_counter3 := star_counter3 + 1;
```

```
end if;

end if;
end process Drawstar3;
-------------------------------------------------------------------------------
-----------------------
DrawStar4 : process (clk25 )
variable star_x:integer;
variable star_y:integer;
variable star4_v_counter : integer := 0;
variable star4_h_counter : integer := 0;
variable star4_row : unsigned(15 downto 0);
variable star_toggle4: std_logic := '0';
variable star_counter4: integer := 0;
begin
if rising_edge(clk25) then



if EndOfLine = '1' then
if ( star4_v_counter > 0 ) then
star4_v_counter := star4_v_counter + 1;
end if;
if Vcount = VSYNC + VBACK_PORCH - 1 + STAR4_V_COORD then
star4_v_counter := 1;
elsif Vcount = VSYNC + VBACK_PORCH - 1 + STAR4_V_COORD + STAR_LENGTH-1 then
star4_v_counter := 0;
end if;
end if;

    if ( star4_h_counter > 0 ) then
star4_h_counter := star4_h_counter + 1;
end if;

if Hcount = HSYNC + HBACK_PORCH - 1 + STAR4_H_COORD then
          star4_h_counter := 1;
    elsif Hcount = HSYNC + HBACK_PORCH - 1 + STAR4_H_COORD + STAR_WIDTH - 1
then
          star4_h_counter := 0;
end if;

if (( star4_h_counter > 0 ) and (star4_v_counter > 0)) then
 if star_toggle4 = '1' then
star4_row := star_sprite(star4_v_counter-1);
else
star4_row := star_sprite2(star4_v_counter-1);
end if;
star4_bit <= star4_row(16-star4_h_counter);
else
star4_bit <= '0';
end if;
if star_counter4 = 19000000 then
star_counter4 := 0;
star_toggle4 := not star_toggle4;
else
star_counter4 := star_counter4 + 1;
end if;
```

```
end if;
end process DrawStar4;
-------------------------------------------------------------------------------
-----------------------
Drawstar5 : process (clk25 )
variable star_x:integer;
variable star_y:integer;
variable star5_v_counter : integer := 0;
variable star5_h_counter : integer := 0;
variable star5_row : unsigned(15 downto 0);
variable star_toggle5: std_logic := '0';
variable star_counter5: integer := 0;
begin
if rising_edge(clk25) then



if EndOfLine = '1' then
if ( star5_v_counter > 0 ) then
star5_v_counter := star5_v_counter + 1;
end if;
if Vcount = VSYNC + VBACK_PORCH - 1 + STAR5_V_COORD then
star5_v_counter := 1;
elsif Vcount = VSYNC + VBACK_PORCH - 1 + STAR5_V_COORD + STAR_LENGTH-1 then
star5_v_counter := 0;
end if;
end if;

    if ( star5_h_counter > 0 ) then
star5_h_counter := star5_h_counter + 1;
end if;

if Hcount = HSYNC + HBACK_PORCH - 1 + STAR5_H_COORD then
          star5_h_counter := 1;
    elsif Hcount = HSYNC + HBACK_PORCH - 1 + STAR5_H_COORD + STAR_WIDTH - 1
then
          star5_h_counter := 0;
end if;

if (( star5_h_counter > 0 ) and (star5_v_counter > 0)) then
 if star_toggle5 = '1' then
star5_row := star_sprite(star5_v_counter-1);
else
star5_row := star_sprite2(star5_v_counter-1);
end if;
star5_bit <= star5_row(16-star5_h_counter);

else
star5_bit <= '0';
end if;
if star_counter5 = 22000000 then
star_counter5 := 0;
star_toggle5 := not star_toggle5;
else
star_counter5 := star_counter5 + 1;
end if;
```

```
end if;
end process Drawstar5;
-------------------------------------------------------------------------------
------------------------
Drawstar6 : process (clk25 )
variable star_x:integer;
variable star_y:integer;
variable star6_v_counter : integer := 0;
variable star6_h_counter : integer := 0;
variable star6_row : unsigned(15 downto 0);
variable star_toggle6: std_logic := '0';
variable star_counter6: integer := 0;
begin
if rising_edge(clk25) then



if EndOfLine = '1' then
if ( star6_v_counter > 0 ) then
star6_v_counter := star6_v_counter + 1;
end if;
if Vcount = VSYNC + VBACK_PORCH - 1 + STAR6_V_COORD then
star6_v_counter := 1;
elsif Vcount = VSYNC + VBACK_PORCH - 1 + STAR6_V_COORD + STAR_LENGTH-1 then
star6_v_counter := 0;
end if;
end if;

    if ( star6_h_counter > 0 ) then
star6_h_counter := star6_h_counter + 1;
end if;

if Hcount = HSYNC + HBACK_PORCH - 1 + STAR6_H_COORD then
          star6_h_counter := 1;
    elsif Hcount = HSYNC + HBACK_PORCH - 1 + STAR6_H_COORD + STAR_WIDTH - 1
then
          star6_h_counter := 0;
end if;

if (( star6_h_counter > 0 ) and (star6_v_counter > 0)) then
 if star_toggle6 = '1' then
star6_row := star_sprite(star6_v_counter-1);
else
star6_row := star_sprite2(star6_v_counter-1);
end if;
star6_bit <= star6_row(16-star6_h_counter);

else
star6_bit <= '0';
end if;
if star_counter6 = 25000000 then
star_counter6 := 0;
star_toggle6 := not star_toggle6;
else
star_counter6 := star_counter6 + 1;
end if;
```

```
end if;
end process Drawstar6;
-------------------------------------------------------------------------------
------------------------
Drawstar7 : process (clk25 )
variable star_x:integer;
variable star_y:integer;
variable star7_v_counter : integer := 0;
variable star7_h_counter : integer := 0;
variable star7_row : unsigned(15 downto 0);
variable star_toggle7: std_logic := '0';
variable star_counter7: integer := 0;
begin
if rising_edge(clk25) then



if EndOfLine = '1' then
if ( star7_v_counter > 0 ) then
star7_v_counter := star7_v_counter + 1;
end if;
if Vcount = VSYNC + VBACK_PORCH - 1 + STAR7_V_COORD then
star7_v_counter := 1;
elsif Vcount = VSYNC + VBACK_PORCH - 1 + STAR7_V_COORD + STAR_LENGTH-1 then
star7_v_counter := 0;
end if;
end if;

    if ( star7_h_counter > 0 ) then
star7_h_counter := star7_h_counter + 1;
end if;

if Hcount = HSYNC + HBACK_PORCH - 1 + STAR7_H_COORD then
          star7_h_counter := 1;
    elsif Hcount = HSYNC + HBACK_PORCH - 1 + STAR7_H_COORD + STAR_WIDTH - 1
then
          star7_h_counter := 0;
end if;

if (( star7_h_counter > 0 ) and (star7_v_counter > 0)) then
 if star_toggle7 = '1' then
star7_row := star_sprite(star7_v_counter-1);
else
star7_row := star_sprite2(star7_v_counter-1);
end if;
star7_bit <= star7_row(16-star7_h_counter);

else
star7_bit <= '0';
end if;
if star_counter7 = 8000000 then
star_counter7 := 0;
star_toggle7 := not star_toggle7;
else
star_counter7 := star_counter7 + 1;
end if;
```

```
end if;
end process Drawstar7;
-------------------------------------------------------------------------------
------------------------
Drawstar9 : process (clk25 )
variable star_x:integer;
variable star_y:integer;
variable star9_v_counter : integer := 0;
variable star9_h_counter : integer := 0;
variable star9_row : unsigned(15 downto 0);
variable star_toggle9: std_logic := '0';
variable star_counter9: integer := 0;
begin
if rising_edge(clk25) then



if EndOfLine = '1' then
if ( star9_v_counter > 0 ) then
star9_v_counter := star9_v_counter + 1;
end if;
if Vcount = VSYNC + VBACK_PORCH - 1 + STAR9_V_COORD then
star9_v_counter := 1;
elsif Vcount = VSYNC + VBACK_PORCH - 1 + STAR9_V_COORD + STAR_LENGTH-1 then
star9_v_counter := 0;
end if;
end if;

    if ( star9_h_counter > 0 ) then
star9_h_counter := star9_h_counter + 1;
end if;

if Hcount = HSYNC + HBACK_PORCH - 1 + STAR9_H_COORD then
          star9_h_counter := 1;
    elsif Hcount = HSYNC + HBACK_PORCH - 1 + STAR9_H_COORD + STAR_WIDTH - 1
then
          star9_h_counter := 0;
end if;

if (( star9_h_counter > 0 ) and (star9_v_counter > 0)) then
 if star_toggle9 = '1' then
star9_row := star_sprite(star9_v_counter-1);
else
star9_row := star_sprite2(star9_v_counter-1);
end if;
star9_bit <= star9_row(16-star9_h_counter);

else
star9_bit <= '0';
end if;
if star_counter9 = 19000000 then
star_counter9 := 0;
star_toggle9 := not star_toggle9;
else
star_counter9 := star_counter9 + 1;
end if;
```

```
end if;
end process Drawstar9;
-------------------------------------------------------------------------------
------------------------
Drawstar8 : process (clk25 )
variable star_x:integer;
variable star_y:integer;
variable star8_v_counter : integer := 0;
variable star8_h_counter : integer := 0;
variable star8_row : unsigned(15 downto 0);
variable star_toggle8: std_logic := '0';
variable star_counter8: integer := 0;
begin
if rising_edge(clk25) then



if EndOfLine = '1' then
if ( star8_v_counter > 0 ) then
star8_v_counter := star8_v_counter + 1;
end if;
if Vcount = VSYNC + VBACK_PORCH - 1 + STAR8_V_COORD then
star8_v_counter := 1;
elsif Vcount = VSYNC + VBACK_PORCH - 1 + STAR8_V_COORD + STAR_LENGTH-1 then
star8_v_counter := 0;
end if;
end if;

    if ( star8_h_counter > 0 ) then
star8_h_counter := star8_h_counter + 1;
end if;

if Hcount = HSYNC + HBACK_PORCH - 1 + STAR8_H_COORD then
          star8_h_counter := 1;
    elsif Hcount = HSYNC + HBACK_PORCH - 1 + STAR8_H_COORD + STAR_WIDTH - 1
then
          star8_h_counter := 0;
end if;

if (( star8_h_counter > 0 ) and (star8_v_counter > 0)) then
 if star_toggle8 = '1' then
star8_row := star_sprite(star8_v_counter-1);
else
star8_row := star_sprite2(star8_v_counter-1);
end if;

star8_bit <= star8_row(16-star8_h_counter);

else
star8_bit <= '0';
end if;
if star_counter8 = 17000000 then
star_counter8 := 0;
star_toggle8 := not star_toggle8;
else
star_counter8 := star_counter8 + 1;
```

```
end if;

end if;
end process Drawstar8;
-------------------------------------------------------------------------------
-------------------------
Drawstar10 : process (clk25 )
variable star_x:integer;
variable star_y:integer;
variable star10_v_counter : integer := 0;
variable star10_h_counter : integer := 0;
variable star10_row : unsigned(15 downto 0);
variable star_toggle10: std_logic := '0';
variable star_counter10: integer := 0;
begin
if rising_edge(clk25) then



if EndOfLine = '1' then
if ( star10_v_counter > 0 ) then
star10_v_counter := star10_v_counter + 1;
end if;
if Vcount = VSYNC + VBACK_PORCH - 1 + STAR10_V_COORD then
star10_v_counter := 1;
elsif Vcount = VSYNC + VBACK_PORCH - 1 + STAR10_V_COORD + STAR_LENGTH-1 then
star10_v_counter := 0;
end if;
end if;

    if ( star10_h_counter > 0 ) then
star10_h_counter := star10_h_counter + 1;
end if;

if Hcount = HSYNC + HBACK_PORCH - 1 + STAR10_H_COORD then
          star10_h_counter := 1;
    elsif Hcount = HSYNC + HBACK_PORCH - 1 + STAR10_H_COORD + STAR_WIDTH - 1
then
          star10_h_counter := 0;
end if;

if (( star10_h_counter > 0 ) and (star10_v_counter > 0)) then
 if star_toggle10 = '1' then
star10_row := star_sprite(star10_v_counter-1);
else
star10_row := star_sprite2(star10_v_counter-1);
end if;

star10_bit <= star10_row(16-star10_h_counter);

else
star10_bit <= '0';
end if;
if star_counter10 = 10000000 then
star_counter10 := 0;
star_toggle10 := not star_toggle10;
else
```

```
star_counter10 := star_counter10 + 1;
end if;

end if;
end process Drawstar10;
-------------------------------------------------------------------------------
------------------------
DrawStar11 : process (clk25 )
variable star_x:integer;
variable star_y:integer;
variable star11_v_counter : integer := 0;
variable star11_h_counter : integer := 0;
variable star11_row : unsigned(15 downto 0);
variable star_toggle11: std_logic := '0';
variable star_counter11: integer := 0;
begin
if rising_edge(clk25) then



if EndOfLine = '1' then
if ( star11_v_counter > 0 ) then
star11_v_counter := star11_v_counter + 1;
end if;
if Vcount = VSYNC + VBACK_PORCH - 1 + STAR11_V_COORD then
star11_v_counter := 1;
elsif Vcount = VSYNC + VBACK_PORCH - 1 + STAR11_V_COORD + STAR_LENGTH-1 then
star11_v_counter := 0;
end if;
end if;

    if ( star11_h_counter > 0 ) then
star11_h_counter := star11_h_counter + 1;
end if;

if Hcount = HSYNC + HBACK_PORCH - 1 + STAR11_H_COORD then
          star11_h_counter := 1;
    elsif Hcount = HSYNC + HBACK_PORCH - 1 + STAR11_H_COORD + STAR_WIDTH - 1
then
          star11_h_counter := 0;
end if;

if (( star11_h_counter > 0 ) and (star11_v_counter > 0)) then
 if star_toggle11 = '1' then
star11_row := star_sprite(star11_v_counter-1);
else
star11_row := star_sprite2(star11_v_counter-1);
end if;
star11_bit <= star11_row(16-star11_h_counter);

else
star11_bit <= '0';
end if;
if star_counter11 = 7000000 then
star_counter11 := 0;
star_toggle11 := not star_toggle11;
else
```

```
star_counter11 := star_counter11 + 1;
end if;

end if;
end process DrawStar11;
-------------------------------------------------------------------------------
-------------------------
DrawStar12 : process (clk25 )
variable star_x:integer;
variable star_y:integer;
variable star12_v_counter : integer := 0;
variable star12_h_counter : integer := 0;
variable star12_row : unsigned(15 downto 0);
variable star_toggle12: std_logic := '0';
variable star_counter12: integer := 0;
begin
if rising_edge(clk25) then



if EndOfLine = '1' then
if ( star12_v_counter > 0 ) then
star12_v_counter := star12_v_counter + 1;
end if;
if Vcount = VSYNC + VBACK_PORCH - 1 + STAR12_V_COORD then
star12_v_counter := 1;
elsif Vcount = VSYNC + VBACK_PORCH - 1 + STAR12_V_COORD + STAR_LENGTH-1 then
star12_v_counter := 0;
end if;
end if;

    if ( star12_h_counter > 0 ) then
star12_h_counter := star12_h_counter + 1;
end if;

if Hcount = HSYNC + HBACK_PORCH - 1 + STAR12_H_COORD then
          star12_h_counter := 1;
    elsif Hcount = HSYNC + HBACK_PORCH - 1 + STAR12_H_COORD + STAR_WIDTH - 1
then
          star12_h_counter := 0;
end if;

if (( star12_h_counter > 0 ) and (star12_v_counter > 0)) then
 if star_toggle12 = '1' then
star12_row := star_sprite(star12_v_counter-1);
else
star12_row := star_sprite2(star12_v_counter-1);
end if;
star12_bit <= star12_row(16-star12_h_counter);

else
star12_bit <= '0';
end if;
if star_counter12 = 5000000 then
star_counter12 := 0;
star_toggle12 := not star_toggle12;
else
```

```
star_counter12 := star_counter12 + 1;
end if;

end if;
end process DrawStar12;
-------------------------------------------------------------------------------
-------------------------
DrawStar13 : process (clk25 )
variable star_x:integer;
variable star_y:integer;
variable star13_v_counter : integer := 0;
variable star13_h_counter : integer := 0;
variable star13_row : unsigned(15 downto 0);
variable star_toggle13: std_logic := '0';
variable star_counter13: integer := 0;
begin
if rising_edge(clk25) then



if EndOfLine = '1' then
if ( star13_v_counter > 0 ) then
star13_v_counter := star13_v_counter + 1;
end if;
if Vcount = VSYNC + VBACK_PORCH - 1 + STAR13_V_COORD then
star13_v_counter := 1;
elsif Vcount = VSYNC + VBACK_PORCH - 1 + STAR13_V_COORD + STAR_LENGTH-1 then
star13_v_counter := 0;
end if;
end if;

    if ( star13_h_counter > 0 ) then
star13_h_counter := star13_h_counter + 1;
end if;

if Hcount = HSYNC + HBACK_PORCH - 1 + STAR13_H_COORD then
          star13_h_counter := 1;
    elsif Hcount = HSYNC + HBACK_PORCH - 1 + STAR13_H_COORD + STAR_WIDTH - 1
then
          star13_h_counter := 0;
end if;

if (( star13_h_counter > 0 ) and (star13_v_counter > 0)) then
 if star_toggle13 = '1' then
star13_row := star_sprite(star13_v_counter-1);
else
star13_row := star_sprite2(star13_v_counter-1);
end if;
star13_bit <= star13_row(16-star13_h_counter);

else
star13_bit <= '0';
end if;
if star_counter13 = 6000000 then
star_counter13 := 0;
star_toggle13 := not star_toggle13;
else
```

```
star_counter13 := star_counter13 + 1;
end if;

end if;
end process DrawStar13;
-------------------------------------------------------------------------------
-------------------------
DrawStar14 : process (clk25 )
variable star_x:integer;
variable star_y:integer;
variable star14_v_counter : integer := 0;
variable star14_h_counter : integer := 0;
variable star14_row : unsigned(15 downto 0);
variable star_toggle14: std_logic := '0';
variable star_counter14: integer := 0;
begin
if rising_edge(clk25) then



if EndOfLine = '1' then
if ( star14_v_counter > 0 ) then
star14_v_counter := star14_v_counter + 1;
end if;
if Vcount = VSYNC + VBACK_PORCH - 1 + STAR14_V_COORD then
star14_v_counter := 1;
elsif Vcount = VSYNC + VBACK_PORCH - 1 + STAR14_V_COORD + STAR_LENGTH-1 then
star14_v_counter := 0;
end if;
end if;

    if ( star14_h_counter > 0 ) then
star14_h_counter := star14_h_counter + 1;
end if;

if Hcount = HSYNC + HBACK_PORCH - 1 + STAR14_H_COORD then
          star14_h_counter := 1;
    elsif Hcount = HSYNC + HBACK_PORCH - 1 + STAR14_H_COORD + STAR_WIDTH - 1
then
          star14_h_counter := 0;
end if;

if (( star14_h_counter > 0 ) and (star14_v_counter > 0)) then
 if star_toggle14 = '1' then
star14_row := star_sprite(star14_v_counter-1);
else
star14_row := star_sprite2(star14_v_counter-1);
end if;
star14_bit <= star14_row(16-star14_h_counter);

else
star14_bit <= '0';
end if;
if star_counter14 = 8000000 then
star_counter14 := 0;
star_toggle14 := not star_toggle14;
else
```

```
star_counter14 := star_counter14 + 1;
end if;

end if;
end process DrawStar14;
-------------------------------------------------------------------------------
--------------------------
DrawStar15 : process (clk25 )
variable star_x:integer;
variable star_y:integer;
variable star15_v_counter : integer := 0;
variable star15_h_counter : integer := 0;
variable star15_row : unsigned(15 downto 0);
variable star_toggle15: std_logic := '0';
variable star_counter15: integer := 0;
begin
if rising_edge(clk25) then



if EndOfLine = '1' then
if ( star15_v_counter > 0 ) then
star15_v_counter := star15_v_counter + 1;
end if;
if Vcount = VSYNC + VBACK_PORCH - 1 + STAR15_V_COORD then
star15_v_counter := 1;
elsif Vcount = VSYNC + VBACK_PORCH - 1 + STAR15_V_COORD + STAR_LENGTH-1 then
star15_v_counter := 0;
end if;
end if;

    if ( star15_h_counter > 0 ) then
star15_h_counter := star15_h_counter + 1;
end if;

if Hcount = HSYNC + HBACK_PORCH - 1 + STAR15_H_COORD then
          star15_h_counter := 1;
    elsif Hcount = HSYNC + HBACK_PORCH - 1 + STAR15_H_COORD + STAR_WIDTH - 1
then
          star15_h_counter := 0;
end if;

if (( star15_h_counter > 0 ) and (star15_v_counter > 0)) then
 if star_toggle15 = '1' then
star15_row := star_sprite(star15_v_counter-1);
else
star15_row := star_sprite2(star15_v_counter-1);
end if;
star15_bit <= star15_row(16-star15_h_counter);

else
star15_bit <= '0';
end if;
if star_counter15 = 10000000 then
star_counter15 := 0;
star_toggle15 := not star_toggle15;
else
```

```
star_counter15 := star_counter15 + 1;
end if;

end if;
end process DrawStar15;
-------------------------------------------------------------------------------
---------------------------
DrawStar16 : process (clk25 )
variable star_x:integer;
variable star_y:integer;
variable star16_v_counter : integer := 0;
variable star16_h_counter : integer := 0;
variable star16_row : unsigned(15 downto 0);
variable star_toggle16: std_logic := '0';
variable star_counter16: integer := 0;
begin
if rising_edge(clk25) then



if EndOfLine = '1' then
if ( star16_v_counter > 0 ) then
star16_v_counter := star16_v_counter + 1;
end if;
if Vcount = VSYNC + VBACK_PORCH - 1 + STAR16_V_COORD then
star16_v_counter := 1;
elsif Vcount = VSYNC + VBACK_PORCH - 1 + STAR16_V_COORD + STAR_LENGTH-1 then
star16_v_counter := 0;
end if;
end if;

    if ( star16_h_counter > 0 ) then
star16_h_counter := star16_h_counter + 1;
end if;

if Hcount = HSYNC + HBACK_PORCH - 1 + STAR16_H_COORD then
          star16_h_counter := 1;
    elsif Hcount = HSYNC + HBACK_PORCH - 1 + STAR16_H_COORD + STAR_WIDTH - 1
then
          star16_h_counter := 0;
end if;

if (( star16_h_counter > 0 ) and (star16_v_counter > 0)) then
 if star_toggle16 = '1' then
star16_row := star_sprite(star16_v_counter-1);
else
star16_row := star_sprite2(star16_v_counter-1);
end if;
star16_bit <= star16_row(16-star16_h_counter);

else
star16_bit <= '0';
end if;
if star_counter16 = 10000000 then
star_counter16 := 0;
star_toggle16 := not star_toggle16;
else
```

```
star_counter16 := star_counter16 + 1;
end if;

end if;
end process DrawStar16;
-------------------------------------------------------------------------------
----------------------------
DrawStar17 : process (clk25 )
variable star_x:integer;
variable star_y:integer;
variable star17_v_counter : integer := 0;
variable star17_h_counter : integer := 0;
variable star17_row : unsigned(15 downto 0);
variable star_toggle17: std_logic := '0';
variable star_counter17: integer := 0;
begin
if rising_edge(clk25) then



if EndOfLine = '1' then
if ( star17_v_counter > 0 ) then
star17_v_counter := star17_v_counter + 1;
end if;
if Vcount = VSYNC + VBACK_PORCH - 1 + STAR17_V_COORD then
star17_v_counter := 1;
elsif Vcount = VSYNC + VBACK_PORCH - 1 + STAR17_V_COORD + STAR_LENGTH-1 then
star17_v_counter := 0;
end if;
end if;

    if ( star17_h_counter > 0 ) then
star17_h_counter := star17_h_counter + 1;
end if;

if Hcount = HSYNC + HBACK_PORCH - 1 + STAR17_H_COORD then
          star17_h_counter := 1;
    elsif Hcount = HSYNC + HBACK_PORCH - 1 + STAR17_H_COORD + STAR_WIDTH - 1
then
          star17_h_counter := 0;
end if;

if (( star17_h_counter > 0 ) and (star17_v_counter > 0)) then
 if star_toggle17 = '1' then
star17_row := star_sprite(star17_v_counter-1);
else
star17_row := star_sprite2(star17_v_counter-1);
end if;
star17_bit <= star17_row(16-star17_h_counter);

else
star17_bit <= '0';
end if;
if star_counter17 = 7500000 then
star_counter17 := 0;
star_toggle17 := not star_toggle17;
else
```

```
star_counter17 := star_counter17 + 1;
end if;

end if;
end process DrawStar17;

--Listening to Software Updates
ram_address <= address(15 downto 0);
 ListenForUpdates: process( clk )
variable kill_info : unsigned(6 downto 0);
variable barricade_info : unsigned(2 downto 0);
begin
      if rising_edge(clk) then
            if reset = '1' then
                  home_h_coord_temp <= (others => '0');
                  home_status <= '1';
                  wave_h_coord_temp <= (others => '0');
                  wave_v_coord_temp <= (others => '0');
                  score_array(0) <= 0;
                  score_array(1) <= 0;
                  score_array(2) <= 0;
                  score_array(3) <= 0;
                  score_array(4) <= 0;
                  timer1_num <= 0;
                  timer2_num <= 0;
                  timer3_num <= 0;
                  timer4_num <= 0;
                  lives1_num <= 0;
                  home_missile_x_temp <= 0;
                  home_missile_y_temp <= 0;
                  alien_missile_x_temp(0) <= 0;
                  alien_missile_y_temp(0) <= 0;
                  alien_missile_x_temp(1) <= 0;
                  alien_missile_y_temp(1) <= 0;
                  alien_missile_x_temp(2) <= 0;
                  alien_missile_y_temp(2) <= 0;
                  alien_missile_x_temp(3) <= 0;
                  alien_missile_y_temp(3) <= 0;
                  alien_missile_x_temp(4) <= 0;
                  alien_missile_y_temp(4) <= 0;
                  alien_missile_x_temp(5) <= 0;
                  alien_missile_y_temp(5) <= 0;
                  alien_missile_x_temp(6) <= 0;
                  alien_missile_y_temp(6) <= 0;
                  alien_missile_x_temp(7) <= 0;
                  alien_missile_y_temp(7) <= 0;
                  alien_missile_x_temp(8) <= 0;
                  alien_missile_y_temp(8) <= 0;
                  alien_missile_x_temp(9) <= 0;
                  alien_missile_y_temp(9) <= 0;


                  alien_status_1 <= x"FFFFFFFF";
                  alien_status_2 <= x"FFFFFFFF";
                  barricade_status(0) <= x"FFFFFFFFFFFFFFFF";
                  barricade_status(1) <= x"FFFFFFFFFFFFFFFF";
                  barricade_status(2) <= x"FFFFFFFFFFFFFFFF";
```

```
                barricade_status(3) <= x"FFFFFFFFFFFFFFFF";


            else
                  if ( chipselect = '1' ) then
                        if ( write = '1' ) then
                              if  ram_address = x"FFFF" then
                                          -- reset from software
                                          alien_status_1 <= x"FFFFFFFF";
                                          alien_status_2 <= x"FFFFFFFF";

                                          home_status <= '1';
                                          --------------------------------
                                          barricade_status(0) <=
x"FFFFFFFFFFFFFFFF";
                                          barricade_status(1) <=
x"FFFFFFFFFFFFFFFF";
                                          barricade_status(2) <=
x"FFFFFFFFFFFFFFFF";
                                          barricade_status(3) <=
x"FFFFFFFFFFFFFFFF";

                                          alien_missile_x_temp(0) <= 0;
                                          alien_missile_y_temp(0) <= 0;
                                          alien_missile_x_temp(1) <= 0;
                                          alien_missile_y_temp(1) <= 0;
                                          alien_missile_x_temp(2) <= 0;
                                          alien_missile_y_temp(2) <= 0;
                                          alien_missile_x_temp(3) <= 0;
                                          alien_missile_y_temp(3) <= 0;
                                          alien_missile_x_temp(4) <= 0;
                                          alien_missile_y_temp(4) <= 0;
                                          alien_missile_x_temp(5) <= 0;
                                          alien_missile_y_temp(5) <= 0;
                                          alien_missile_x_temp(6) <= 0;
                                          alien_missile_y_temp(6) <= 0;
                                          alien_missile_x_temp(7) <= 0;
                                          alien_missile_y_temp(7) <= 0;
                                          alien_missile_x_temp(8) <= 0;
                                          alien_missile_y_temp(8) <= 0;
                                          alien_missile_x_temp(9) <= 0;
                                          alien_missile_y_temp(9) <= 0;


                              elsif ram_address = x"0000" then
                                    home_h_coord_temp <= writedata(15 downto
0); -- home ship x coordinate
                              elsif ram_address = x"0001" then
                                    wave_h_coord_temp <= writedata(15 downto
0);
                              elsif ram_address = x"0002" then
                                    wave_v_coord_temp <= writedata(15 downto
0);
                              ----------------------------------------
                              elsif ram_address = x"0003" then
                                    score_array(0) <= to_integer(writedata(15
downto 0));
```

```
                              elsif ram_address = x"0004" then
                                    score_array(1) <= to_integer(writedata(15
downto 0));
                              elsif ram_address = x"0005" then
                                    score_array(2) <= to_integer(writedata(15
downto 0));
                              elsif ram_address = x"0006" then
                                    score_array(3) <= to_integer(writedata(15
downto 0));
                              elsif ram_address = x"0007" then
                                    score_array(4) <= to_integer(writedata(15
downto 0));
                              elsif ram_address = x"0008" then
                                    lives1_num <= to_integer(writedata(15
downto 0));
                              elsif ram_address = x"0009" then
                                    home_missile_x_temp <=
to_integer(writedata(15 downto 0));
                              elsif ram_address = x"000a" then
                                    home_missile_y_temp <=
to_integer(writedata(15 downto 0));
                              elsif ram_address = x"000b" then
                                    home_status <= writedata(0);
                              elsif ram_address >= x"0010" and ram_address <=
x"0019" then

      alien_missile_x_temp(to_integer(ram_address(3 downto 0))) <=
to_integer(writedata(15 downto 0));
                              elsif ram_address >= x"0020" and ram_address <=
x"0029" then

      alien_missile_y_temp(to_integer(ram_address(3 downto 0))) <=
to_integer(writedata(15 downto 0));
                              elsif ram_address  = x"0030" then
                                  timer1_num <= to_integer(writedata(15
downto 0));
                              elsif ram_address  = x"0031" then
                                  timer2_num <= to_integer(writedata(15
downto 0));
                              elsif ram_address  = x"0032" then
                                  timer3_num <= to_integer(writedata(15
downto 0));
                              elsif ram_address  = x"0033" then
                                  timer4_num <= to_integer(writedata(15
downto 0));


                              elsif ram_address(15) = '1' then
                                    if ( ram_address(14) = '0' ) then
                                          kill_info := ram_address(6 downto
0);
                                          if ( kill_info <= 31 ) then

      alien_status_1(to_integer(kill_info)) <= '0';
                                          else

      alien_status_2(to_integer(kill_info-31)) <= '0';
```

```
                                        end if;
                                    elsif ( ram_address(14) = '1' ) then

      barricade_status(to_integer(ram_address(7 downto
6)))(to_integer(ram_address(5 downto 0))) <= '0';
                                    end if;


                                    ----------------------------------------
----------
                                    elsif ram_address=x"0034" then
                                          game_state<=to_integer(writedata(15
downto 0));
                                          --game_state<=3;
                                 ----------------------------------------
-------
                              end if ;
                        end if;

                        if ( read = '1' ) then
                              if ram_address = "00000000" then
                                    readdata <= home_h_coord;
                              elsif ram_address = "00000001" then
                                    readdata <= wave_h_coord;
                              elsif ram_address = "00000010" then
                                    readdata <= wave_v_coord;
                              end if ;
                        end if; -- read
                  end if; -- chipselect
            end if; -- reset
      end if; -- rising edge

end process ListenForUpdates;

  -- Registered video signals going to the video DAC

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

    elsif clk25'event and clk25 = '1' then
            if home_bit = '1' then
                  VGA_R <= "1111111111";
                  VGA_G <= "0000000000";
                  VGA_B <= "0000000000";

            elsif wave_bit = '1' then
                  VGA_R <= "1111111111";
                  VGA_G <= "1111111111";
                  VGA_B <= "0000000000";

            elsif homemissile_bit = '1' then
                  VGA_R <= "1111111111";
                  VGA_G <= "1111111111";
```

```
                VGA_B <= "1111111111";

            elsif alienmissile_bits(0) = '1' or alienmissile_bits(1) = '1' or
alienmissile_bits(2) = '1'
            or alienmissile_bits(3) = '1' or alienmissile_bits(4) = '1' or
alienmissile_bits(5) = '1'
            or alienmissile_bits(6) = '1' or alienmissile_bits(7) = '1' or
alienmissile_bits(8)='1' or alienmissile_bits(9)='1'
            then
                VGA_R <= "1111111111";
                VGA_G <= "0000000000";
                VGA_B <= "1111111111";

            elsif score_bit = '1' then
                VGA_R <= "1111111111";
                VGA_G <= "1111111111";
                VGA_B <= "1111111111";

            elsif score_bit_1='1' then
                VGA_R <= "0000000000";
                VGA_G <= "1111111111";
                VGA_B <= "0000000000";

            elsif intro_bit = '1' then
                VGA_R <= "0000000000";
                VGA_G <= "1111111111";
                VGA_B <= "0000000000";

            elsif game_over_bit = '1' then
                VGA_R <= "1111111111";
                VGA_G <= "0000000000";
                VGA_B <= "0000000000";

            elsif lives_bit = '1' then
                VGA_R <= "1111111111";
                VGA_G <= "1111111111";
                VGA_B <= "1111111111";

            elsif lives_bit_1 = '1' or lives_bit_2 = '1' or lives_bit_3 = '1'
then
                VGA_R <= "1111111111";
                VGA_G <= "0000000000";
                VGA_B <= "0000000000";

            elsif timer_bit = '1' or timer_bit_1 = '1' or timer_bit_2 = '1'
            or timer_bit_3 = '1' or timer_bit_4 = '1'  then
                VGA_R <= "0000000000";
                VGA_G <= "0000000000";
                VGA_B <= "1111111111";

            elsif barricade1_bit = '1' or barricade2_bit = '1' or
barricade3_bit = '1' or barricade4_bit = '1' then
                VGA_R <= "0000000000";
                VGA_G <= "1111111111";
                VGA_B <= "0000000000";
```

```
            elsif star1_bit = '1' or star2_bit = '1' or star3_bit = '1' or
star4_bit = '1' or star5_bit = '1'
        or star6_bit = '1' or star7_bit = '1' or star8_bit = '1' or star9_bit
= '1' or star10_bit = '1'
        or star11_bit = '1' or star12_bit = '1' or star13_bit = '1' or
star14_bit = '1' or star15_bit = '1'
        or star16_bit  = '1' or star17_bit = '1' then
                  VGA_R <= "1111111111";
                  VGA_G <= "1111111111";
                  VGA_B <= "1111111111";

            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;
```

8.3 audio_peripheral.vhd

```
--audio peripheral
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity audio_peripheral is

port(

      avs_s1_clk : in std_logic;
```

```
      avs_s1_reset_n : in std_logic;
      avs_s1_read : in std_logic;
      avs_s1_write: in std_logic;
      avs_s1_chipselect: in std_logic;
      avs_s1_address: in std_logic;
      avs_s1_readdata: out unsigned(31 downto 0);
      avs_s1_writedata: in unsigned(31 downto 0);
      freq : out std_logic_vector(15 downto 0);
      mod_depth: out std_logic_vector(3 downto 0);
      --on_off: out std_logic
      on_off:out std_logic_vector(15 downto 0)
);

end audio_peripheral;


architecture rtl of audio_peripheral is

      signal frq_reg: unsigned(31 downto 0);
      signal mod_depth_reg:unsigned(3 downto 0);
      signal cnt:unsigned(27 downto 0):=x"0000000";
      signal x: std_logic:='0';
      signal y: std_logic:='0';
      signal flag: unsigned(2 downto 0):="011";
      signal clk : std_logic:='0';

begin


clk<=avs_s1_clk;

 peripheral:process(clk)
      begin
            if rising_edge(clk) then
                  if avs_s1_reset_n = '0' then

                              avs_s1_readdata<=(others=>'0');
                  else
                      if avs_s1_chipselect = '1' then
                  if avs_s1_write = '1' then
                              if avs_s1_address = '0' then
                                    frq_reg<=avs_s1_writedata(31 downto 0);
                              end if;
                        end if;
                      end if;
                  end if;
            end if;
end process peripheral;


      process(clk)
      variable time_counter : integer := 0;
            begin
             if rising_edge(clk) then

                        freq<= std_logic_vector(frq_reg(15 downto 0));
                        --freq<=X"00AA";
```

```
                        mod_depth<=x"A";


                        on_off<=std_logic_vector(frq_reg(31 downto 16));
                        --on_off<=x"1111";
            end if;

      end process;
end rtl;
```