PACMAN GAME

CSEE4840 Embedded System Final Project Report (May 2010)

Yunde Shi [ys2405@columbia.edu]
Department of Mechanical Engineering

PART I Basic Objective

(1) There is only one PacMan.
(2) There is only one "active" ghost.
(3) There are lots of small dot to eat (1 point / small dot); There are 4 large dots at the corners (5 points / large dot).
(4) The "score" at the bottom row should update whenever new dots are eaten.
(5) The "chasing" algorithm is the most naive ---- ghost simply goes towards the location of pacman.

The appearance of the game is shown in Figure 1 below.

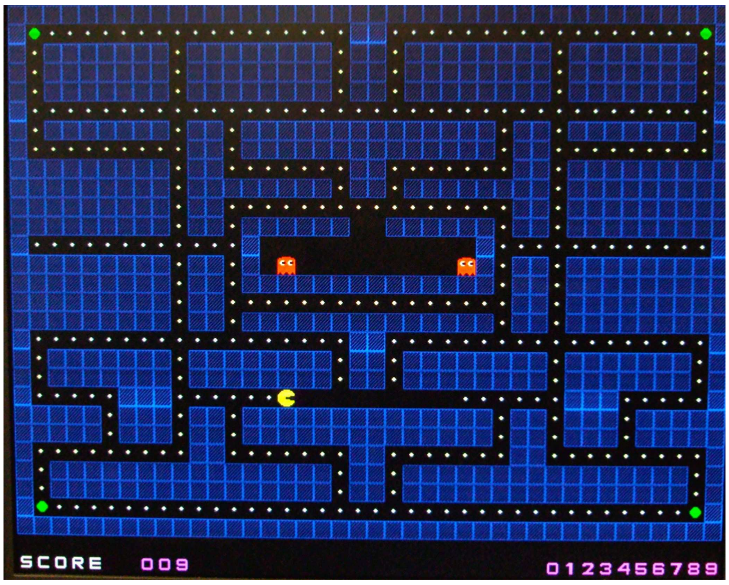


Figure 1. General Appearance of the PacMan Game

PART II Overall Design (hardware)

The NIOS-II system consists of the CPU, SRAM and two peripherals of PS2 and VGA.
% The VGA has only one mission ---- displaying the sprites based on the control array.
% The PS2 keyboard gets the key pressing from the player and sends out the specific key code.
% The NIOS processor reads the key code and decides the associated updates of the control array.
The communication between each module is through the Avalon Bus.
The block diagram of the system is shown in Figure 2 below.

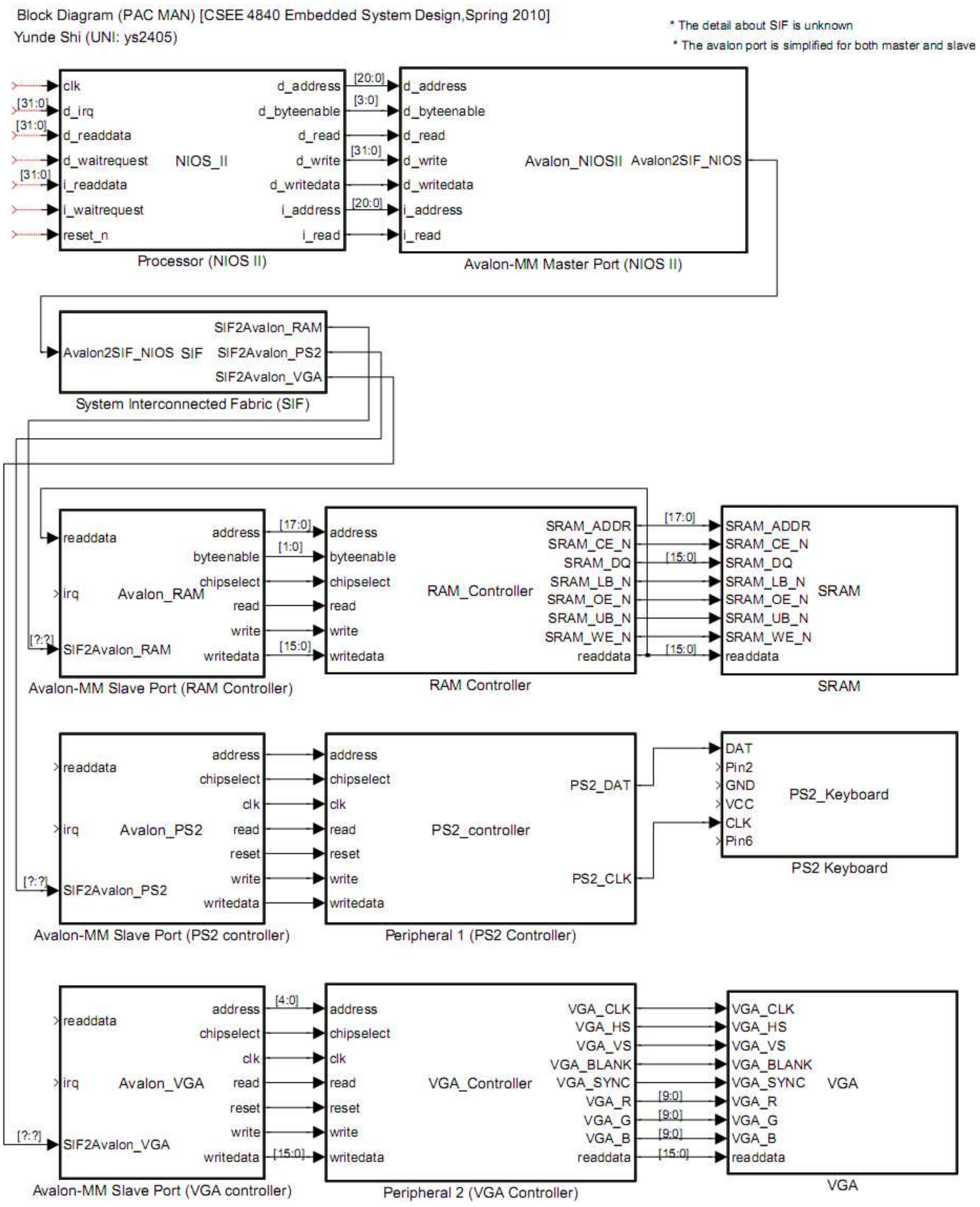
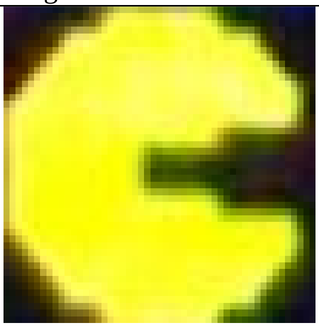

Block Diagram (PAC MAN) [CSEE 4840 Embedded System Design,Spring 2010]
Yunde Shi (UNI: ys2405)
* The detail about SIF is unknown
* The avalon port is simplified for both master and slave
Processor (NIOS II)
NIOS_II
Avalon-MM Master Port (NIOS II)
Avalon_NIOSII
Avalon2SIF_NIOS
System Interconnected Fabric (SIF)
SIF
SIF2Avalon_RAM
SIF2Avalon_PS2
SIF2Avalon_VGA
Avalon-MM Slave Port (RAM Controller)
Avalon_RAM
RAM Controller
RAM_Controller
SRAM
Avalon-MM Slave Port (PS2 controller)
Avalon_PS2
Peripheral 1 (PS2 Controller)
PS2_controller
PS2 Keyboard
PS2_Keyboard
Avalon-MM Slave Port (VGA controller)
Avalon_VGA
Peripheral 2 (VGA Controller)
VGA_Controller
VGA

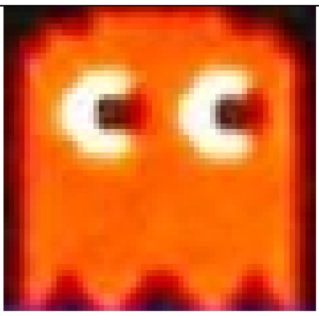
Figure 2. Block Diagram of the System

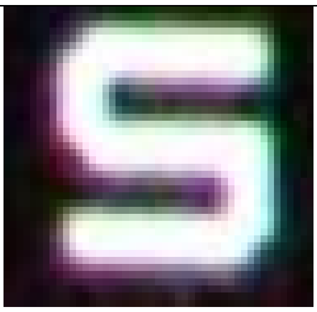
PART III Detailed Design ()

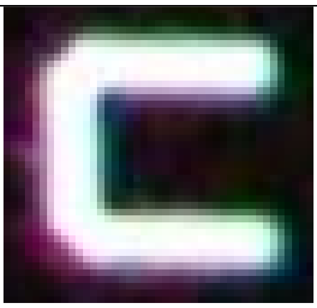
3.1 Sprites

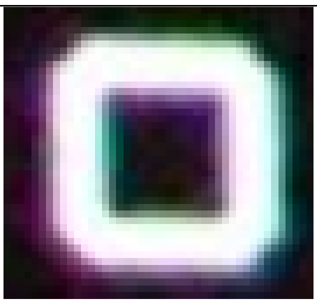
Sprite Name	Image Shot	VHDL sprite code

Pacman
(Mono-color: yellow)
		`sprite_pacman( 0) <= "0000000000000000";`
		`sprite_pacman( 1) <= "0000011111100000";`
		`sprite_pacman( 2) <= "0001111111111000";`
		`sprite_pacman( 3) <= "0011111111111100";`
		`sprite_pacman( 4) <= "0111111111111110";`
		`sprite_pacman( 5) <= "1111111111111110";`
		`sprite_pacman( 6) <= "1111111111100000";`
		`sprite_pacman( 7) <= "1111111000000000";`
		`sprite_pacman( 8) <= "1111111000000000";`
		`sprite_pacman( 9) <= "1111111111100000";`
		`sprite_pacman(10) <= "1111111111111110";`
		`sprite_pacman(11) <= "0111111111111110";`
		`sprite_pacman(12) <= "0011111111111100";`
		`sprite_pacman(13) <= "0001111111111000";`
		`sprite_pacman(14) <= "0000011111100000";`
		`sprite_pacman(15) <= "0000000000000000";`

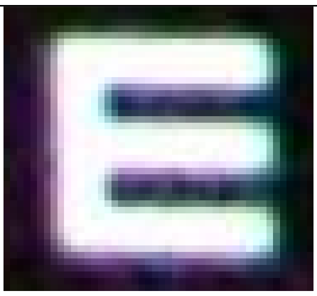
Ghost
(*the sprite code shown is only for the red channel.)
		`sprite_ghost_R( 0) <= "0000000000000000";`
		`sprite_ghost_R( 1) <= "0000111111110000";`
		`sprite_ghost_R( 2) <= "0111111111111110";`
		`sprite_ghost_R( 3) <= "1111111111111111";`
		`sprite_ghost_R( 4) <= "1111111111111111";`
		`sprite_ghost_R( 5) <= "1111000111100011";`
		`sprite_ghost_R( 6) <= "1111000111100011";`
		`sprite_ghost_R( 7) <= "1111111111111111";`
		`sprite_ghost_R( 8) <= "1111111111111111";`
		`sprite_ghost_R( 9) <= "1111111111111111";`
		`sprite_ghost_R(10) <= "1111111111111111";`
		`sprite_ghost_R(11) <= "1111111111111111";`
		`sprite_ghost_R(12) <= "1111111111111111";`
		`sprite_ghost_R(13) <= "1111111111111111";`
		`sprite_ghost_R(14) <= "1101111001111011";`
		`sprite_ghost_R(15) <= "1000110000110001";`

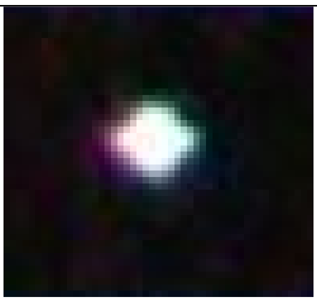
Letter 'S'
(white)
		`sprite_S( 0) <= "0000000000000000";`
		`sprite_S( 1) <= "0000000000000000";`
		`sprite_S( 2) <= "0000000000000000";`
		`sprite_S( 3) <= "0000111111111000";`
		`sprite_S( 4) <= "0001111111111000";`
		`sprite_S( 5) <= "0001100000000000";`
		`sprite_S( 6) <= "0001100000000000";`
		`sprite_S( 7) <= "0001111111111000";`
		`sprite_S( 8) <= "0000111111111000";`
		`sprite_S( 9) <= "0000000000011000";`
		`sprite_S(10) <= "0000000000011000";`
		`sprite_S(11) <= "0001111111111000";`
		`sprite_S(12) <= "0001111111110000";`
		`sprite_S(13) <= "0000000000000000";`
		`sprite_S(14) <= "0000000000000000";`
		`sprite_S(15) <= "0000000000000000";`

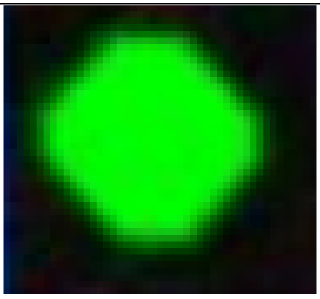
Letter 'C'
(white)
		`sprite_C( 0) <= "0000000000000000";`
		`sprite_C( 1) <= "0000000000000000";`
		`sprite_C( 2) <= "0000000000000000";`
		`sprite_C( 3) <= "0000111111111000";`
		`sprite_C( 4) <= "0001111111111000";`
		`sprite_C( 5) <= "0001100000000000";`
		`sprite_C( 6) <= "0001100000000000";`
		`sprite_C( 7) <= "0001100000000000";`
		`sprite_C( 8) <= "0001100000000000";`
		`sprite_C( 9) <= "0001100000000000";`
		`sprite_C(10) <= "0001100000000000";`
		`sprite_C(11) <= "0001111111111000";`
		`sprite_C(12) <= "0000111111111000";`
		`sprite_C(13) <= "0000000000000000";`
		`sprite_C(14) <= "0000000000000000";`
		`sprite_C(15) <= "0000000000000000";`

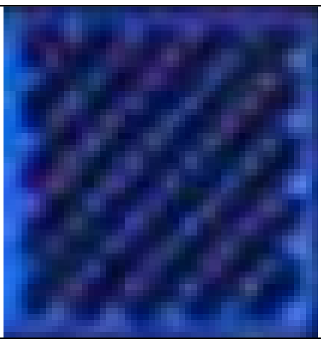
Letter 'O'
(white)
		`sprite_O( 0) <= "0000000000000000";`
		`sprite_O( 1) <= "0000000000000000";`
		`sprite_O( 2) <= "0000000000000000";`
		`sprite_O( 3) <= "0000111111110000";`
		`sprite_O( 4) <= "0001111111111000";`
		`sprite_O( 5) <= "0001100000011000";`
		`sprite_O( 6) <= "0001100000011000";`
		`sprite_O( 7) <= "0001100000011000";`
		`sprite_O( 8) <= "0001100000011000";`
		`sprite_O( 9) <= "0001100000011000";`
		`sprite_O(10) <= "0001100000011000";`
		`sprite_O(11) <= "0001111111111000";`
		`sprite_O(12) <= "0000111111110000";`
		`sprite_O(13) <= "0000000000000000";`
		`sprite_O(14) <= "0000000000000000";`
		`sprite_O(15) <= "0000000000000000";`

Letter 'R'
(white)
		`sprite_R( 0) <= "0000000000000000";`
		`sprite_R( 1) <= "0000000000000000";`
		`sprite_R( 2) <= "0000000000000000";`
		`sprite_R( 3) <= "0000111111110000";`
		`sprite_R( 4) <= "0001111111111000";`
		`sprite_R( 5) <= "0001100000011000";`
		`sprite_R( 6) <= "0001100000011000";`
		`sprite_R( 7) <= "0001111111111000";`
		`sprite_R( 8) <= "0001111111110000";`
		`sprite_R( 9) <= "0001100111000000";`
		`sprite_R(10) <= "0001100011100000";`
		`sprite_R(11) <= "0001100001110000";`
		`sprite_R(12) <= "0001100000111000";`
		`sprite_R(13) <= "0000000000000000";`
		`sprite_R(14) <= "0000000000000000";`
		`sprite_R(15) <= "0000000000000000";`

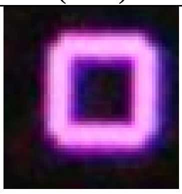
Letter 'E'
(white)
		`sprite_E( 0) <= "0000000000000000";`
		`sprite_E( 1) <= "0000000000000000";`
		`sprite_E( 2) <= "0000000000000000";`
		`sprite_E( 3) <= "0001111111111000";`
		`sprite_E( 4) <= "0001111111111000";`
		`sprite_E( 5) <= "0001100000000000";`
		`sprite_E( 6) <= "0001100000000000";`
		`sprite_E( 7) <= "0001111111111000";`
		`sprite_E( 8) <= "0001111111111000";`
		`sprite_E( 9) <= "0001100000000000";`
		`sprite_E(10) <= "0001100000000000";`
		`sprite_E(11) <= "0001111111111000";`
		`sprite_E(12) <= "0001111111111000";`
		`sprite_E(13) <= "0000000000000000";`
		`sprite_E(14) <= "0000000000000000";`
		`sprite_E(15) <= "0000000000000000";`

Small Dot (white)		`sprite_dot_small( 0) <= "0000000000000000";`
		`sprite_dot_small( 1) <= "0000000000000000";`
		`sprite_dot_small( 2) <= "0000000000000000";`
		`sprite_dot_small( 3) <= "0000000000000000";`
		`sprite_dot_small( 4) <= "0000000000000000";`
		`sprite_dot_small( 5) <= "0000000000000000";`
		`sprite_dot_small( 6) <= "0000000110000000";`
		`sprite_dot_small( 7) <= "0000001111000000";`
		`sprite_dot_small( 8) <= "0000000110000000";`
		`sprite_dot_small( 9) <= "0000000000000000";`
		`sprite_dot_small(10) <= "0000000000000000";`
		`sprite_dot_small(11) <= "0000000000000000";`
		`sprite_dot_small(12) <= "0000000000000000";`
		`sprite_dot_small(13) <= "0000000000000000";`
		`sprite_dot_small(14) <= "0000000000000000";`
		`sprite_dot_small(15) <= "0000000000000000";`

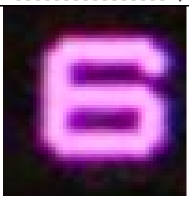
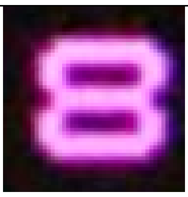
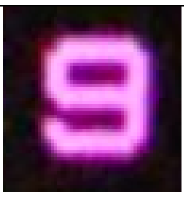
Large Dot (green)		`sprite_dot_large( 0) <= "0000000000000000";`
		`sprite_dot_large( 1) <= "0000000000000000";`
		`sprite_dot_large( 2) <= "0000000000000000";`
		`sprite_dot_large( 3) <= "0000001111000000";`
		`sprite_dot_large( 4) <= "0000011111100000";`
		`sprite_dot_large( 5) <= "0000111111110000";`
		`sprite_dot_large( 6) <= "0001111111111000";`
		`sprite_dot_large( 7) <= "0001111111111000";`
		`sprite_dot_large( 8) <= "0001111111111000";`
		`sprite_dot_large( 9) <= "0000111111110000";`
		`sprite_dot_large(10) <= "0000011111100000";`
		`sprite_dot_large(11) <= "0000001111000000";`
		`sprite_dot_large(12) <= "0000000000000000";`
		`sprite_dot_large(13) <= "0000000000000000";`
		`sprite_dot_large(14) <= "0000000000000000";`
		`sprite_dot_large(15) <= "0000000000000000";`

Brick (shaded)
color: blue
		`sprite_brick( 0) <= "1111111111111111";`
		`sprite_brick( 1) <= "1001001001001001";`
		`sprite_brick( 2) <= "1010010010010011";`
		`sprite_brick( 3) <= "1100100100100101";`
		`sprite_brick( 4) <= "1001001001001001";`
		`sprite_brick( 5) <= "1010010010010011";`
		`sprite_brick( 6) <= "1100100100100101";`
		`sprite_brick( 7) <= "1001001001001001";`
		`sprite_brick( 8) <= "1010010010010011";`
		`sprite_brick( 9) <= "1100100100100101";`
		`sprite_brick(10) <= "1001001001001001";`
		`sprite_brick(11) <= "1010010010010011";`
		`sprite_brick(12) <= "1100100100100101";`
		`sprite_brick(13) <= "1001001001001001";`
		`sprite_brick(14) <= "1010010010010011";`
		`sprite_brick(15) <= "1111111111111111";`

Decimal Numbers (Pink):

Sprite Image					

Sprite Code	"0000000000000000";
	"0000000000000000";
	"0000000000000000";
	"0000111111110000";
	"0001111111111000";
	"0001100000011000";
	"0001100000011000";
	"0001100000011000";
	"0001100000011000";
	"0001100000011000";
	"0001100000011000";
	"0001111111111000";
	"0000111111110000";
	"0000000000000000";
	"0000000000000000";
	"0000000000000000";
		"0000000000000000";
		"0000000000000000";
		"0000000000000000";
		"0000001111000000";
		"0000011111000000";
		"0000111111000000";
		"0000000111000000";
		"0000000111000000";
		"0000000111000000";
		"0000000111000000";
		"0000000111000000";
		"0000000111000000";
		"0000000111000000";
		"0000000000000000";
		"0000000000000000";
		"0000000000000000";
			"0000000000000000";
			"0000000000000000";
			"0000000000000000";
			"0000111111110000";
			"0001111111111000";
			"0001110000111000";
			"0000000000111000";
			"0000000001110000";
			"0000000011100000";
			"0000000111000000";
			"0000001110000000";
			"0000111111111000";
			"0001111111111000";
			"0000000000000000";
			"0000000000000000";
			"0000000000000000";
				"0000000000000000";
				"0000000000000000";
				"0000000000000000";
				"0000111111110000";
				"0001111111111000";
				"0001100000011000";
				"0000000000011000";
				"0000000111110000";
				"0000000111110000";
				"0000000000011000";
				"0001100000011000";
				"0001111111111000";
				"0000111111110000";
				"0000000000000000";
				"0000000000000000";
				"0000000000000000";
					"0000000000000000";
					"0000000000000000";
					"0000000000000000";
					"0000000011111000";
					"0000000111111000";
					"0000001110011000";
					"0000011100011000";
					"0000111000011000";
					"0001111111111100";
					"0001111111111100";
					"0000000000011000";
					"0000000000011000";
					"0000000000011000";
					"0000000000000000";
					"0000000000000000";
					"0000000000000000";

Sprite Image					

Sprite Code	"0000000000000000";
	"0000000000000000";
	"0000000000000000";
	"0001111111111000";
	"0001111111111000";
	"0001100000000000";
	"0001100000000000";
	"0001111111110000";
	"0001111111111000";
	"0000000000011000";
	"0001100000011000";
	"0001111111111000";
	"0000111111110000";
	"0000000000000000";
	"0000000000000000";
	"0000000000000000";
		"0000000000000000";
		"0000000000000000";
		"0000000000000000";
		"0000111111110000";
		"0001111111111000";
		"0001100000011000";
		"0001100000000000";
		"0001111111110000";
		"0001111111111000";
		"0001100000011000";
		"0001100000011000";
		"0001111111111000";
		"0000111111110000";
		"0000000000000000";
		"0000000000000000";
		"0000000000000000";
			"0000000000000000";
			"0000000000000000";
			"0000000000000000";
			"0001111111111000";
			"0001111111111000";
			"0000000001110000";
			"0000000001110000";
			"0000000011100000";
			"0000000011100000";
			"0000000111000000";
			"0000000111000000";
			"0000001110000000";
			"0000001110000000";
			"0000000000000000";
			"0000000000000000";
			"0000000000000000";
				"0000000000000000";
				"0000000000000000";
				"0000000000000000";
				"0000111111110000";
				"0001111111111000";
				"0001100000011000";
				"0001100000011000";
				"0000111111110000";
				"0000111111110000";
				"0001100000011000";
				"0001100000011000";
				"0001111111111000";
				"0000111111110000";
				"0000000000000000";
				"0000000000000000";
				"0000000000000000";
					"0000000000000000";
					"0000000000000000";
					"0000000000000000";
					"0000111111110000";
					"0001111111111000";
					"0001100000011000";
					"0001100000011000";
					"0001111111111000";
					"0000111111111000";
					"0000000000011000";
					"0001100000011000";
					"0001111111111000";
					"0000111111110000";
					"0000000000000000";
					"0000000000000000";
					"0000000000000000";

3.2 Control Array (vhd version shown below, 30x(40*4) binary array)

```
     0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39
     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
("1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111",
 "1111001100100010001000100010001000100010001000100010001000100010001000100010001011111111001000100010001000100010001000100010001000100010001000100010001000111111",
 "1111001011111111111111111111111111110010111111111111111111111111111111110010111111110010111111111111111111111111111111110010111111111111111111111111111100101111",
 "1111001011111111111111111111111111110010111111111111111111111111111111110010111111110010111111111111111111111111111111110010111111111111111111111111111100101111",
 "1111001011111111111111111111111111110010111111111111111111111111111111110010111111110010111111111111111111111111111111110010111111111111111111111111111100101111",
 "1111001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000101111",
 "1111001011111111111111111111111111110010111111110010111111111111111111111111111111111111111111111111111111110010111111110010111111111111111111111111111100101111",
 "1111001000100010001000100010001000100010111111110010111111111111111111111111111111111111111111111111111111110010111111110010001000100010001000100010001000101111",
 "1111111111111111111111111111111111110010111111110010001000100010001000100010001011111111001000100010001000100010001000101111111100101111111111111111111111111111",
 "1111111111111111111111111111111111110010111111111111111111111111111111110010111111110010111111111111111111111111111111110010111111111111111111111111111111111111",
 "1111111111111111111111111111111111110010111111110010001000100010001000100010001000100010001000100010001000100010001000101111111100101111111111111111111111111111",
 "1111111111111111111111111111111111110010111111110010111111111111111111111111000000001111111111111111111111110010111111110010111111111111111111111111111111111111",
 "1111001000100010001000100010001000100010001000100010111100000000000000000000000000000000000000000000000011110010001000100010001000100010001000100010001000101111",
 "1111111111111111111111111111111111110010111111110010111101000000000000000000000000000000000000000000010011110010111111110010111111111111111111111111111111111111",
 "1111111111111111111111111111111111110010111111110010111111111111111111111111111111111111111111111111111111110010111111110010111111111111111111111111111111111111",
 "1111111111111111111111111111111111110010111111110010001000100010001000100010001000100010001000100010001000100010111111110010111111111111111111111111111111111111",
 "1111111111111111111111111111111111110010111111110010111111111111111111111111111111111111111111111111111111110010111111110010111111111111111111111111111111111111",
 "1111001000100010001000100010001000100010001000100010001000100010001000100010001011111111001000100010001000100010001000100010001000100010001000100010001000101111",
 "1111001011111111111111111111111111110010111111111111111111111111111111110010111111110010111111111111111111111111111111110010111111111111111111111111111100101111",
 "1111001011111111111111111111111111110010111111111111111111111111111111110010111111110010111111111111111111111111111111110010111111111111111111111111111100101111",
 "1111001000100010001000100010111111111111001000100010001000100010001000100010001000100001001000100010001000100010001000100010001011111111111100100010001000100010001011111",
 "1111111111111111111100101111111111110010111111110010111111111111111111111111111111111111111111111111111111110010111111110010111111111111001011111111111111111111",
 "1111111111111111111100101111111111110010111111110010111111111111111111111111111111111111111111111111111111110010111111110010111111111111001011111111111111111111",
 "1111001000100010001000100010001000100010111111110010001000100010001000100010001011111111001000100010001000100010001000101111111100100010001000100010001000100010001000101111",
 "1111001011111111111111111111111111111111111111111111111111111111111111110010111111110010111111111111111111111111111111111111111111111111111111111111111100101111",
 "1111001011111111111111111111111111111111111111111111111111111111111111110010111111110010111111111111111111111111111111111111111111111111111111111111111100101111",
 "1111001100100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000111111",
 "1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111",
 "0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",
 "0001001000110100010100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001100111100010011010101111001101111011111" );
     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
     0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39
```

The control Array is 30x40 (since the sprite is 16x16 pixel and the screen is 480x640), and we use 4 bits to represent a sprite type (e.g. pacman, ghost, etc.)
There could be at least 16 sprite types because by adding additional constraints of row numbers, we can have more sprite types using the 4 bits representation. (see the following.)

Note: different sprite type can have the same control array code (e.g. 'S' is 0001 and pacman is also 0001, but since their row location would never be the same, they are distinguishable.)

```
=========================================
  -- for control_array_V <= 27:
  -- pacman        : 0001
  -- brick         : 1111
  -- background    : 0000
  -- dot_small     : 0010
  -- dot_large     : 0011
  -- ghost         : 0100

  -- for control_array_V = 29:
  -- 'S'           : 0001
  -- 'C'           : 0010
  -- 'O'           : 0011
  -- 'R'           : 0100
  -- 'E'           : 0101
  -- '0'           : 0110
  -- '1'           : 0111
  -- '2'           : 1000
  -- '3'           : 1001
  -- '4'           : 1010
  -- '5'           : 1011
  -- '6'           : 1100
  -- '7'           : 1101
  -- '8'           : 1110
  -- '9'           : 1111
=========================================
```

3.3 How to update the control array declared in VHDL code

Since we didn't let the VHDL code read the control array through the SRAM (as the on-block RAM (60kB) is large enough for our purpose (control array in de2_vga_raster.vhd is just 30x160 of zeros and ones)), we keep the control array *in both the C code and de2_vga_raster.vhd code*.
And by using IOWR_16DIRECT(VGA_BASE, adr_offset, data), we can update the associated entry of the control array in the vhd code whenever such a change first occurs in the C code, so that the control array in the C and vhd are updated in a almost synchronized manner (there may be a bit delay for the vhd code, but that delay is never observable ☺)

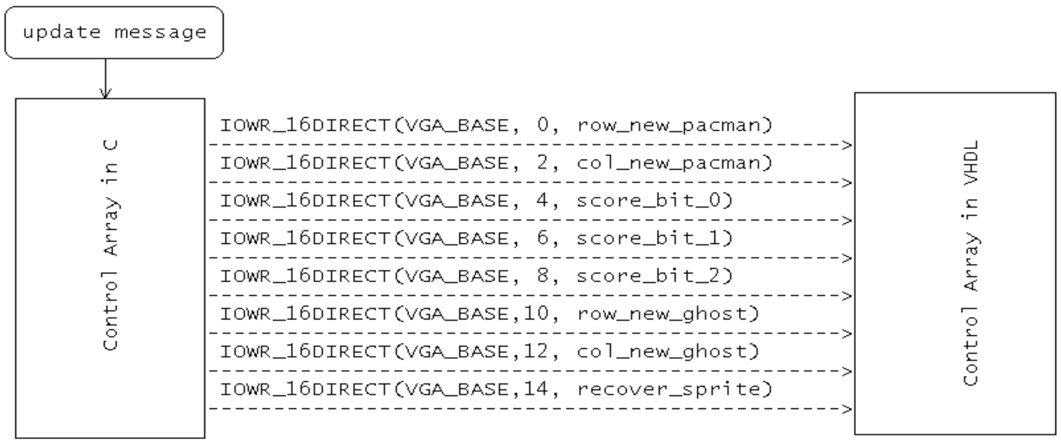

update message
Control Array in C
IOWR_16DIRECT(VGA_BASE, 0, row_new_pacman)
IOWR_16DIRECT(VGA_BASE, 2, col_new_pacman)
IOWR_16DIRECT(VGA_BASE, 4, score_bit_0)
IOWR_16DIRECT(VGA_BASE, 6, score_bit_1)
IOWR_16DIRECT(VGA_BASE, 8, score_bit_2)
IOWR_16DIRECT(VGA_BASE,10, row_new_ghost)
IOWR_16DIRECT(VGA_BASE,12, col_new_ghost)
IOWR_16DIRECT(VGA_BASE,14, recover_sprite)
Control Array in VHDL

Figure 3. Update the control array in vhd code through C

PART III Difficulties and Lesson Learned

(1) *PS2 keyboard reading never succeeds*
Solution: The signal name DATA and DAT should not be confused.
Note: The "DE2_pin_assignments.csv" has a line of

```
------------------------------
PS2_DAT     PIN_C24
------------------------------
```

(2) *The sprite display on the screen is not "certain" when the key board is pressed to change the location of pacman*
Solution: Keep sending the control array updates instructions even in the while loop for the key board reading, because sometimes sending the updates once is not "enough" / "reliable" enough.

(3) *The naïve chasing algorithm of the pacman by the ghost does not work well whenever there's a wall between the pacman and the ghost as the ghost always tries to go towards the pacman.*
Solution: (not implemented due to time issue; inspired by BaoLin Shao) Using "decision tree" should be able to solve this problem where the tree has some certain depth based on the trade off between the effectiveness of chasing and the decision time consumed ---- the tree is generated based on all the possible moving choice of the ghost and some decision law is applied to pick the best "path" inside the tree.

Acknowledgement

I sincerely thank Baolin Shao and Scott Schuff during my entire project ---- they are extremely helpful and they always encourage me to see the "bright side" whenever I was clogged in the hardship no matter how stupid my game is.Their encouragement spurs the pacman come out.

APPENDIX (Useful Files)

file_1: 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;
    read       : in  std_logic;
    write      : in  std_logic;
    chipselect : in  std_logic;
    address    : in  unsigned(4 downto 0);
    readdata   : out unsigned(15 downto 0);
    writedata  : in  unsigned(15 downto 0);

    VGA_CLK,                         -- Clock -- Should be 25.125 MHz
    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;

  constant RECTANGLE_HSTART : integer := 100;
  constant RECTANGLE_HEND   : integer := 540;
  constant RECTANGLE_VSTART : integer := 100;
  constant RECTANGLE_VEND   : integer := 380;



  -- Signals for the video controller
  signal Hcount : unsigned(9 downto 0);  -- Horizontal position (0-800)
  signal Vcount : unsigned(9 downto 0);  -- Vertical position (0-524)
  signal EndOfLine, EndOfField : std_logic;

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

  signal rectangle_h, rectangle_v, rectangle : std_logic;  -- rectangle area
  signal clk_counter : std_logic := '0';
  signal circle : std_logic;
  signal circle_center_x : unsigned (9 downto 0) := "0100000000";   -- the x_pos of the circle
  signal circle_center_y : unsigned (9 downto 0) := "0100000000";   -- the y_pos of the circle
  signal delay_time : unsigned (7 downto 0) := "00000000";
  signal circle_radius   : unsigned (9 downto 0) := "0000010100";    -- the radius of the circle

  signal color : unsigned (1 downto 0):="00";  -- fatal! if i don't initialize my color, the ball
would not show on the screen (THANKS to Baolin)

  type array_type_16x16 is array (15 downto 0) of unsigned (15 downto 0);
  signal sprite_brick  : array_type_16x16;
  signal sprite_pacman : array_type_16x16;
  signal sprite_dot_small: array_type_16x16;
  signal sprite_dot_large: array_type_16x16;
  signal sprite_ghost_R: array_type_16x16;
  signal sprite_ghost_G: array_type_16x16;
  signal sprite_ghost_B: array_type_16x16;

  signal sprite_S: array_type_16x16;
  signal sprite_C: array_type_16x16;
  signal sprite_O: array_type_16x16;
  signal sprite_R: array_type_16x16;
  signal sprite_E: array_type_16x16;

  signal sprite_0: array_type_16x16;
  signal sprite_1: array_type_16x16;
  signal sprite_2: array_type_16x16;
  signal sprite_3: array_type_16x16;
  signal sprite_4: array_type_16x16;
  signal sprite_5: array_type_16x16;
  signal sprite_6: array_type_16x16;
  signal sprite_7: array_type_16x16;
  signal sprite_8: array_type_16x16;
  signal sprite_9: array_type_16x16;




  --type array_type_30x160 is array (29 downto 0) of unsigned (159 downto 0);
  type array_type_30x160 is array (29 downto 0, 159 downto 0) of std_logic;
  --signal control_array : array_type_30x160;


  -- for control_array_V <= 27:
  -- pacman        : 0001
  -- brick         : 1111
  -- background    : 0000
  -- dot_small     : 0010
  -- dot_large     : 0011
  -- ghost         : 0100

  -- for control_array_V = 29:
  -- 'S'           : 0001
  -- 'C'           : 0010
  -- 'O'           : 0011
  -- 'R'           : 0100
  -- 'E'           : 0101
  -- '0'           : 0110
  -- '1'           : 0111
  -- '2'           : 1000
  -- '3'           : 1001
  -- '4'           : 1010
  -- '5'           : 1011
  -- '6'           : 1100
  -- '7'           : 1101
  -- '8'           : 1110
  -- '9'           : 1111

  -- pacman initial_pos_row = 20, initial_pos_col = 20
  --                                                 0   1   2   3   4   5   6   7   8   9  10
11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
35  36  37  38  39
```

```
  --                                                    |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |
  signal control_array : array_type_30x160 :=
("11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111",

"111100110010001000100010001000100010001000100010001000100010001000100010001011111111001000100010
0010001000100010001000100010001000100010001000100010001000111111",

"111100101111111111111111111111111111001011111111111111111111111111111110010111111110010111111111
1111111111111111111111110010111111111111111111111111111100101111",

"111100101111111111111111111111111111001011111111111111111111111111111110010111111110010111111111
1111111111111111111111110010111111111111111111111111111100101111",

"111100101111111111111111111111111111001011111111111111111111111111111110010111111110010111111111
1111111111111111111111110010111111111111111111111111111100101111",

"111100100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010
0010001000100010001000100010001000100010001000100010001000101111",

"111100101111111111111111111111111111001011111111001011111111111111111111111111111111111111111111
1111111111110010111111110010111111111111111111111111111100101111",

"111100100010001000100010001000100010001011111111001011111111111111111111111111111111111111111111
1111111111110010111111110010001000100010001000100010001000101111",

"111111111111111111111111111111111111001011111111001000100010001000100010001011111111001000100010
0010001000100010111111110010111111111111111111111111111111111111",

"111111111111111111111111111111111111001011111111111111111111111111111110010111111110010111111111
1111111111111111111111110010111111111111111111111111111111111111",

"111111111111111111111111111111111111001011111111001000100010001000100010001000100010001000100010
0010001000100010111111110010111111111111111111111111111111111111",

"111111111111111111111111111111111111001011111111001011111111111111111111111100000000111111111111
1111111111110010111111110010111111111111111111111111111111111111",

"111100100010001000100010001000100010001000101111000000000000000000000000000000000000000000000000
0000000011110010001000100010001000100010001000100010001000101111",

"111111111111111111111111111111111111001011111111001011110100000000000000000000000000000000000000
0000010011110010111111110010111111111111111111111111111111111111",

"111111111111111111111111111111111111001011111111001011111111111111111111111111111111111111111111
1111111111110010111111110010111111111111111111111111111111111111",

"111111111111111111111111111111111111001011111111001000100010001000100010001000100010001000100010
0010001000100010111111110010111111111111111111111111111111111111",

"111111111111111111111111111111111111001011111111001011111111111111111111111111111111111111111111
1111111111110010111111110010111111111111111111111111111111111111",

"111100100010001000100010001000100010001000100010001000100010001000100010001011111111001000100010
0010001000100010001000100010001000100010001000100010001000101111",

"111100101111111111111111111111111111001011111111111111111111111111111110010111111110010111111111
1111111111111111111111110010111111111111111111111111111100101111",

"111100101111111111111111111111111111001011111111111111111111111111111110010111111110010111111111
1111111111111111111111110010111111111111111111111111111100101111",

"111100100010001000100010111111111111001000100010001000100010001000100010001000100010001000100010
0010001000100010001000100010001000101111111111110010001000100010001000101111",

"111111111111111111110010111111111111001011111111001011111111111111111111111111111111111111111111
1111111111110010111111110010111111111111001011111111111111111111",

"111111111111111111110010111111111111001011111111001011111111111111111111111111111111111111111111
1111111111110010111111110010111111111111001011111111111111111111",
"111100100010001000100010001000100010001011111111001000100010001000100010001011111111001000100010
0010001000100010111111110010001000100010001000100010001000101111",

"111100101111111111111111111111111111111111111111111111111111111111111111001011111111001011111111
1111111111111111111111111111111111111111111111111111111100101111",

"111100101111111111111111111111111111111111111111111111111111111111111111001011111111001011111111
1111111111111111111111111111111111111111111111111111111100101111",

"111100110010001000100010001000100010001000100010001000100010001000100010001000100010001000100010
0010001000100010001000100010001000100010001000100010001000111111",

"111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111",

"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000",

"000100100011010001010000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000001100111100010011010101111001101111101111" );
  --                                                    |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |
  --                                                    0   1   2   3   4   5   6   7   8   9  10
11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
35  36  37  38  39

  -- NOTE: the indexing of control array ! (the top left element corresponds to
control_array(29,159))
  signal sprite_pos_V : unsigned (9 downto 0) := "0000000000";
  signal sprite_pos_H : unsigned (9 downto 0) := "0000000000";

  signal pacman_pos_row_old : unsigned (9 downto 0) := "0000010100";  -- i.e. D'20'
  signal pacman_pos_col_old : unsigned (9 downto 0) := "0000010100";  -- i.e. D'20'
  signal pacman_pos_row_new : unsigned (9 downto 0) := "0000010100";
  signal pacman_pos_col_new : unsigned (9 downto 0) := "0000010100";

  signal score_bit_0 : unsigned (9 downto 0) := "0000000000";
  signal score_bit_1 : unsigned (9 downto 0) := "0000000000";
  signal score_bit_2 : unsigned (9 downto 0) := "0000000000";

  signal ghost_pos_row_old : unsigned (9 downto 0) := "0000001101";  -- i.e. D'13'
  signal ghost_pos_col_old : unsigned (9 downto 0) := "0000001110";  -- i.e. D'14'
  signal ghost_pos_row_new : unsigned (9 downto 0) := "0000001101";
  signal ghost_pos_col_new : unsigned (9 downto 0) := "0000001110";
  signal recover_sprite : unsigned (3 downto 0) := "0000";


  signal step : std_logic := '0';

begin



  sprite_brick( 0) <= "1111111111111111";
  sprite_brick( 1) <= "1001001001001001";
  sprite_brick( 2) <= "1010010010010011";
  sprite_brick( 3) <= "1100100100100101";
  sprite_brick( 4) <= "1001001001001001";
  sprite_brick( 5) <= "1010010010010011";
  sprite_brick( 6) <= "1100100100100101";
  sprite_brick( 7) <= "1001001001001001";
  sprite_brick( 8) <= "1010010010010011";
  sprite_brick( 9) <= "1100100100100101";
  sprite_brick(10) <= "1001001001001001";
  sprite_brick(11) <= "1010010010010011";
  sprite_brick(12) <= "1100100100100101";
  sprite_brick(13) <= "1001001001001001";
  sprite_brick(14) <= "1010010010010011";
  sprite_brick(15) <= "1111111111111111";

  sprite_pacman( 0) <= "0000000000000000";
  sprite_pacman( 1) <= "0000011111100000";
  sprite_pacman( 2) <= "0001111111111000";
  sprite_pacman( 3) <= "0011111111111100";
  sprite_pacman( 4) <= "0111111111111110";
  sprite_pacman( 5) <= "1111111111111110";
  sprite_pacman( 6) <= "1111111111100000";
  sprite_pacman( 7) <= "1111111000000000";
  sprite_pacman( 8) <= "1111111000000000";
  sprite_pacman( 9) <= "1111111111100000";
  sprite_pacman(10) <= "1111111111111110";
  sprite_pacman(11) <= "0111111111111110";
  sprite_pacman(12) <= "0011111111111100";
  sprite_pacman(13) <= "0001111111111000";
  sprite_pacman(14) <= "0000011111100000";
  sprite_pacman(15) <= "0000000000000000";

  sprite_dot_small( 0) <= "0000000000000000";
  sprite_dot_small( 1) <= "0000000000000000";
  sprite_dot_small( 2) <= "0000000000000000";
  sprite_dot_small( 3) <= "0000000000000000";
  sprite_dot_small( 4) <= "0000000000000000";
  sprite_dot_small( 5) <= "0000000000000000";
  sprite_dot_small( 6) <= "0000000110000000";
  sprite_dot_small( 7) <= "0000001111000000";
  sprite_dot_small( 8) <= "0000000110000000";
  sprite_dot_small( 9) <= "0000000000000000";
  sprite_dot_small(10) <= "0000000000000000";
  sprite_dot_small(11) <= "0000000000000000";
  sprite_dot_small(12) <= "0000000000000000";
  sprite_dot_small(13) <= "0000000000000000";
  sprite_dot_small(14) <= "0000000000000000";
  sprite_dot_small(15) <= "0000000000000000";

  sprite_dot_large( 0) <= "0000000000000000";
  sprite_dot_large( 1) <= "0000000000000000";
  sprite_dot_large( 2) <= "0000000000000000";
  sprite_dot_large( 3) <= "0000001111000000";
  sprite_dot_large( 4) <= "0000011111100000";
  sprite_dot_large( 5) <= "0000111111110000";
  sprite_dot_large( 6) <= "0001111111111000";
  sprite_dot_large( 7) <= "0001111111111000";
  sprite_dot_large( 8) <= "0001111111111000";
  sprite_dot_large( 9) <= "0000111111110000";
  sprite_dot_large(10) <= "0000011111100000";
  sprite_dot_large(11) <= "0000001111000000";
  sprite_dot_large(12) <= "0000000000000000";
  sprite_dot_large(13) <= "0000000000000000";
  sprite_dot_large(14) <= "0000000000000000";
  sprite_dot_large(15) <= "0000000000000000";

  sprite_ghost_R( 0) <= "0000000000000000";
  sprite_ghost_R( 1) <= "0000111111110000";
  sprite_ghost_R( 2) <= "0111111111111110";
  sprite_ghost_R( 3) <= "1111111111111111";
  sprite_ghost_R( 4) <= "1111111111111111";
  sprite_ghost_R( 5) <= "1111000111100011";
  sprite_ghost_R( 6) <= "1111000111100011";
  sprite_ghost_R( 7) <= "1111111111111111";
  sprite_ghost_R( 8) <= "1111111111111111";
  sprite_ghost_R( 9) <= "1111111111111111";
  sprite_ghost_R(10) <= "1111111111111111";
  sprite_ghost_R(11) <= "1111111111111111";
  sprite_ghost_R(12) <= "1111111111111111";
  sprite_ghost_R(13) <= "1111111111111111";
  sprite_ghost_R(14) <= "1101111001111011";
  sprite_ghost_R(15) <= "1000110000110001";

  sprite_ghost_G( 0) <= "0000000000000000";
  sprite_ghost_G( 1) <= "0000000000000000";
  sprite_ghost_G( 2) <= "0000000000000000";
  sprite_ghost_G( 3) <= "0000000000000000";
  sprite_ghost_G( 4) <= "0001110000111000";
  sprite_ghost_G( 5) <= "0011000001100000";
  sprite_ghost_G( 6) <= "0011000001100000";
  sprite_ghost_G( 7) <= "0001110000111000";
  sprite_ghost_G( 8) <= "0000000000000000";
  sprite_ghost_G( 9) <= "0000000000000000";
  sprite_ghost_G(10) <= "0000000000000000";
  sprite_ghost_G(11) <= "0000000000000000";
  sprite_ghost_G(12) <= "0000000000000000";
  sprite_ghost_G(13) <= "0000000000000000";
  sprite_ghost_G(14) <= "0000000000000000";
  sprite_ghost_G(15) <= "0000000000000000";

  sprite_ghost_B( 0) <= "0000000000000000";
  sprite_ghost_B( 1) <= "0000000000000000";
  sprite_ghost_B( 2) <= "0000000000000000";
  sprite_ghost_B( 3) <= "0000000000000000";
  sprite_ghost_B( 4) <= "0001110000111000";
  sprite_ghost_B( 5) <= "0011000001100000";
  sprite_ghost_B( 6) <= "0011000001100000";
  sprite_ghost_B( 7) <= "0001110000111000";
  sprite_ghost_B( 8) <= "0000000000000000";
  sprite_ghost_B( 9) <= "0000000000000000";
  sprite_ghost_B(10) <= "0000000000000000";
  sprite_ghost_B(11) <= "0000000000000000";
  sprite_ghost_B(12) <= "0000000000000000";
  sprite_ghost_B(13) <= "0000000000000000";
  sprite_ghost_B(14) <= "0000000000000000";
  sprite_ghost_B(15) <= "0000000000000000";

  sprite_S( 0) <= "0000000000000000";
  sprite_S( 1) <= "0000000000000000";
  sprite_S( 2) <= "0000000000000000";
  sprite_S( 3) <= "0000111111111000";
  sprite_S( 4) <= "0001111111111000";
  sprite_S( 5) <= "0001100000000000";
  sprite_S( 6) <= "0001100000000000";
  sprite_S( 7) <= "0001111111111000";
  sprite_S( 8) <= "0000111111111000";
  sprite_S( 9) <= "0000000000011000";
  sprite_S(10) <= "0000000000011000";
  sprite_S(11) <= "0001111111111000";
  sprite_S(12) <= "0001111111110000";
  sprite_S(13) <= "0000000000000000";
  sprite_S(14) <= "0000000000000000";
  sprite_S(15) <= "0000000000000000";

  sprite_C( 0) <= "0000000000000000";
  sprite_C( 1) <= "0000000000000000";
  sprite_C( 2) <= "0000000000000000";
  sprite_C( 3) <= "0000111111111000";
  sprite_C( 4) <= "0001111111111000";
  sprite_C( 5) <= "0001100000000000";
  sprite_C( 6) <= "0001100000000000";
  sprite_C( 7) <= "0001100000000000";
  sprite_C( 8) <= "0001100000000000";
  sprite_C( 9) <= "0001100000000000";
  sprite_C(10) <= "0001100000000000";
  sprite_C(11) <= "0001111111111000";
  sprite_C(12) <= "0000111111111000";
  sprite_C(13) <= "0000000000000000";
  sprite_C(14) <= "0000000000000000";
  sprite_C(15) <= "0000000000000000";

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


  sprite_R( 0) <= "0000000000000000";
  sprite_R( 1) <= "0000000000000000";
  sprite_R( 2) <= "0000000000000000";
  sprite_R( 3) <= "0000111111110000";
  sprite_R( 4) <= "0001111111111000";
  sprite_R( 5) <= "0001100000011000";
  sprite_R( 6) <= "0001100000011000";
  sprite_R( 7) <= "0001111111111000";
  sprite_R( 8) <= "0001111111110000";
  sprite_R( 9) <= "0001100111000000";
  sprite_R(10) <= "0001100011100000";
  sprite_R(11) <= "0001100001110000";
  sprite_R(12) <= "0001100000111000";
  sprite_R(13) <= "0000000000000000";
  sprite_R(14) <= "0000000000000000";
  sprite_R(15) <= "0000000000000000";


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


  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";




  -- Horizontal and vertical s

  HCounter : process (clk_counter)
  begin
    if rising_edge(clk_counter) then
      if reset = '1' then
        Hcount <= (others => '0');
      elsif EndOfLine = '1' then
```

```
        Hcount <= (others => '0');
      else
        Hcount <= Hcount + 1;
      end if;
    end if;
  end process HCounter;

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

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

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

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

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

  -- Rectangle generator

  RectangleHGen : process (clk_counter)
  begin
    if rising_edge(clk_counter) then
      if reset = '1' or Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART then
        rectangle_h <= '1';
      elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND then
        rectangle_h <= '0';
      end if;
    end if;
  end process RectangleHGen;

  RectangleVGen : process (clk_counter)
  begin
    if rising_edge(clk_counter) then
      if reset = '1' then
        rectangle_v <= '0';
      elsif EndOfLine = '1' then
        if Vcount = VSYNC + VBACK_PORCH - 1 + RECTANGLE_VSTART then
          rectangle_v <= '1';
        elsif Vcount = VSYNC + VBACK_PORCH - 1 + RECTANGLE_VEND then
          rectangle_v <= '0';
        end if;
      end if;
    end if;
  end process RectangleVGen;

  rectangle <= rectangle_h and rectangle_v;




  -- Registered video signals going to the video DAC

  VideoOut: process (clk_counter, reset)
  variable control_array_row : unsigned (159 downto 0);
  variable sprite_brick_row  : unsigned (15 downto 0);
  variable sprite_pacman_row : unsigned (15 downto 0);
  variable sprite_dot_small_row: unsigned (15 downto 0);
  variable sprite_dot_large_row: unsigned (15 downto 0);
  variable sprite_ghost_R_row: unsigned (15 downto 0);
  variable sprite_ghost_G_row: unsigned (15 downto 0);
  variable sprite_ghost_B_row: unsigned (15 downto 0);
  variable sprite_S_row: unsigned (15 downto 0);
  variable sprite_C_row: unsigned (15 downto 0);
  variable sprite_O_row: unsigned (15 downto 0);
  variable sprite_R_row: unsigned (15 downto 0);
  variable sprite_E_row: unsigned (15 downto 0);
  variable sprite_0_row: unsigned (15 downto 0);
  variable sprite_1_row: unsigned (15 downto 0);
  variable sprite_2_row: unsigned (15 downto 0);
  variable sprite_3_row: unsigned (15 downto 0);
  variable sprite_4_row: unsigned (15 downto 0);
  variable sprite_5_row: unsigned (15 downto 0);
  variable sprite_6_row: unsigned (15 downto 0);
  variable sprite_7_row: unsigned (15 downto 0);
  variable sprite_8_row: unsigned (15 downto 0);
  variable sprite_9_row: unsigned (15 downto 0);
  variable control_array_V : unsigned (9 downto 0);
  variable control_array_H : unsigned (9 downto 0);
  variable sprite_array_V : unsigned (9 downto 0);
  variable sprite_array_H : unsigned (9 downto 0);

  begin
    if reset = '1' then
```

```
      VGA_R <= "0000000000";
      VGA_G <= "0000000000";
      VGA_B <= "0000000000";
    elsif clk_counter'event and clk_counter = '1' then  -- indicate rising edge

      -- read updated pacman location
      if (chipselect = '1' and write = '1' and address = "00000") then
        pacman_pos_row_new <= writedata(9 downto 0);
      end if;
      if (chipselect = '1' and write = '1' and address = "00001") then
        pacman_pos_col_new <= writedata(9 downto 0);
      end if;

      -- read updated score bits
      if (chipselect = '1' and write = '1' and address = "00010") then
        score_bit_0 <= writedata(9 downto 0);
      end if;
      if (chipselect = '1' and write = '1' and address = "00011") then
        score_bit_1 <= writedata(9 downto 0);
      end if;
      if (chipselect = '1' and write = '1' and address = "00100") then
        score_bit_2 <= writedata(9 downto 0);
      end if;

      -- read updated ghost location
      if (chipselect = '1' and write = '1' and address = "00101") then
        ghost_pos_row_new <= writedata(9 downto 0);
      end if;
      if (chipselect = '1' and write = '1' and address = "00110") then
        ghost_pos_col_new <= writedata(9 downto 0);
      end if;
      if (chipselect = '1' and write = '1' and address = "00111") then
        recover_sprite <= writedata(3 downto 0);
      end if;


      control_array(to_integer(unsigned(29 - pacman_pos_row_old)),to_integer(unsigned(159 -
(pacman_pos_col_old sll 2) - 0))) <= '0';
      control_array(to_integer(unsigned(29 - pacman_pos_row_old)),to_integer(unsigned(159 -
(pacman_pos_col_old sll 2) - 1))) <= '0';
      control_array(to_integer(unsigned(29 - pacman_pos_row_old)),to_integer(unsigned(159 -
(pacman_pos_col_old sll 2) - 2))) <= '0';
      control_array(to_integer(unsigned(29 - pacman_pos_row_old)),to_integer(unsigned(159 -
(pacman_pos_col_old sll 2) - 3))) <= '0';
      control_array(to_integer(unsigned(29 - pacman_pos_row_new)),to_integer(unsigned(159 -
(pacman_pos_col_new sll 2) - 0))) <= '0';
      control_array(to_integer(unsigned(29 - pacman_pos_row_new)),to_integer(unsigned(159 -
(pacman_pos_col_new sll 2) - 1))) <= '0';
      control_array(to_integer(unsigned(29 - pacman_pos_row_new)),to_integer(unsigned(159 -
(pacman_pos_col_new sll 2) - 2))) <= '0';
      control_array(to_integer(unsigned(29 - pacman_pos_row_new)),to_integer(unsigned(159 -
(pacman_pos_col_new sll 2) - 3))) <= '1';

      if (ghost_pos_row_old /= ghost_pos_row_new OR ghost_pos_col_old /= ghost_pos_col_new) then
        control_array(to_integer(unsigned(29 - ghost_pos_row_old)),to_integer(unsigned(159 -
(ghost_pos_col_old sll 2) - 0)))<= recover_sprite(3);
        control_array(to_integer(unsigned(29 - ghost_pos_row_old)),to_integer(unsigned(159 -
(ghost_pos_col_old sll 2) - 1)))<= recover_sprite(2);
        control_array(to_integer(unsigned(29 - ghost_pos_row_old)),to_integer(unsigned(159 -
(ghost_pos_col_old sll 2) - 2)))<= recover_sprite(1);
        control_array(to_integer(unsigned(29 - ghost_pos_row_old)),to_integer(unsigned(159 -
(ghost_pos_col_old sll 2) - 3)))<= recover_sprite(0);

        control_array(to_integer(unsigned(29 - ghost_pos_row_new)),to_integer(unsigned(159 -
(ghost_pos_col_new sll 2) - 0))) <= '0';
        control_array(to_integer(unsigned(29 - ghost_pos_row_new)),to_integer(unsigned(159 -
(ghost_pos_col_new sll 2) - 1))) <= '1';
        control_array(to_integer(unsigned(29 - ghost_pos_row_new)),to_integer(unsigned(159 -
(ghost_pos_col_new sll 2) - 2))) <= '0';
        control_array(to_integer(unsigned(29 - ghost_pos_row_new)),to_integer(unsigned(159 -
(ghost_pos_col_new sll 2) - 3))) <= '0';
      end if;

      -- update score_bit_0
      if (score_bit_0 = 0) then
        control_array(0,(159 - 36 - 0)) <= '0';
        control_array(0,(159 - 36 - 1)) <= '1';
        control_array(0,(159 - 36 - 2)) <= '1';
        control_array(0,(159 - 36 - 3)) <= '0';
      end if;
      if (score_bit_0 = 1) then
        control_array(0,(159 - 36 - 0)) <= '0';
        control_array(0,(159 - 36 - 1)) <= '1';
        control_array(0,(159 - 36 - 2)) <= '1';
        control_array(0,(159 - 36 - 3)) <= '1';
      end if;
      if (score_bit_0 = 2) then
        control_array(0,(159 - 36 - 0)) <= '1';
        control_array(0,(159 - 36 - 1)) <= '0';
        control_array(0,(159 - 36 - 2)) <= '0';
        control_array(0,(159 - 36 - 3)) <= '0';
      end if;
      if (score_bit_0 = 3) then
        control_array(0,(159 - 36 - 0)) <= '1';
        control_array(0,(159 - 36 - 1)) <= '0';
        control_array(0,(159 - 36 - 2)) <= '0';
        control_array(0,(159 - 36 - 3)) <= '1';
      end if;
      if (score_bit_0 = 4) then
        control_array(0,(159 - 36 - 0)) <= '1';
        control_array(0,(159 - 36 - 1)) <= '0';
        control_array(0,(159 - 36 - 2)) <= '1';
        control_array(0,(159 - 36 - 3)) <= '0';
      end if;
      if (score_bit_0 = 5) then
        control_array(0,(159 - 36 - 0)) <= '1';
        control_array(0,(159 - 36 - 1)) <= '0';
        control_array(0,(159 - 36 - 2)) <= '1';
        control_array(0,(159 - 36 - 3)) <= '1';
      end if;
      if (score_bit_0 = 6) then
        control_array(0,(159 - 36 - 0)) <= '1';
        control_array(0,(159 - 36 - 1)) <= '1';
        control_array(0,(159 - 36 - 2)) <= '0';
        control_array(0,(159 - 36 - 3)) <= '0';
      end if;
      if (score_bit_0 = 7) then
        control_array(0,(159 - 36 - 0)) <= '1';
        control_array(0,(159 - 36 - 1)) <= '1';
        control_array(0,(159 - 36 - 2)) <= '0';
        control_array(0,(159 - 36 - 3)) <= '1';
      end if;
      if (score_bit_0 = 8) then
        control_array(0,(159 - 36 - 0)) <= '1';
        control_array(0,(159 - 36 - 1)) <= '1';
        control_array(0,(159 - 36 - 2)) <= '1';
        control_array(0,(159 - 36 - 3)) <= '0';
      end if;
      if (score_bit_0 = 9) then
        control_array(0,(159 - 36 - 0)) <= '1';
        control_array(0,(159 - 36 - 1)) <= '1';
        control_array(0,(159 - 36 - 2)) <= '1';
        control_array(0,(159 - 36 - 3)) <= '1';
      end if;

      -- update score_bit_1
      if (score_bit_1 = 0) then
        control_array(0,(159 - 32 - 0)) <= '0';
        control_array(0,(159 - 32 - 1)) <= '1';
        control_array(0,(159 - 32 - 2)) <= '1';
        control_array(0,(159 - 32 - 3)) <= '0';
      end if;
      if (score_bit_1 = 1) then
        control_array(0,(159 - 32 - 0)) <= '0';
        control_array(0,(159 - 32 - 1)) <= '1';
        control_array(0,(159 - 32 - 2)) <= '1';
        control_array(0,(159 - 32 - 3)) <= '1';
      end if;
      if (score_bit_1 = 2) then
        control_array(0,(159 - 32 - 0)) <= '1';
        control_array(0,(159 - 32 - 1)) <= '0';
        control_array(0,(159 - 32 - 2)) <= '0';
        control_array(0,(159 - 32 - 3)) <= '0';
      end if;
      if (score_bit_1 = 3) then
        control_array(0,(159 - 32 - 0)) <= '1';
        control_array(0,(159 - 32 - 1)) <= '0';
        control_array(0,(159 - 32 - 2)) <= '0';
        control_array(0,(159 - 32 - 3)) <= '1';
      end if;
      if (score_bit_1 = 4) then
        control_array(0,(159 - 32 - 0)) <= '1';
        control_array(0,(159 - 32 - 1)) <= '0';
        control_array(0,(159 - 32 - 2)) <= '1';
        control_array(0,(159 - 32 - 3)) <= '0';
      end if;
      if (score_bit_1 = 5) then
        control_array(0,(159 - 32 - 0)) <= '1';
        control_array(0,(159 - 32 - 1)) <= '0';
        control_array(0,(159 - 32 - 2)) <= '1';
        control_array(0,(159 - 32 - 3)) <= '1';
      end if;
      if (score_bit_1 = 6) then
        control_array(0,(159 - 32 - 0)) <= '1';
        control_array(0,(159 - 32 - 1)) <= '1';
        control_array(0,(159 - 32 - 2)) <= '0';
        control_array(0,(159 - 32 - 3)) <= '0';
      end if;
      if (score_bit_1 = 7) then
        control_array(0,(159 - 32 - 0)) <= '1';
        control_array(0,(159 - 32 - 1)) <= '1';
        control_array(0,(159 - 32 - 2)) <= '0';
        control_array(0,(159 - 32 - 3)) <= '1';
      end if;
      if (score_bit_1 = 8) then
        control_array(0,(159 - 32 - 0)) <= '1';
        control_array(0,(159 - 32 - 1)) <= '1';
        control_array(0,(159 - 32 - 2)) <= '1';
        control_array(0,(159 - 32 - 3)) <= '0';
      end if;
      if (score_bit_1 = 9) then
        control_array(0,(159 - 32 - 0)) <= '1';
        control_array(0,(159 - 32 - 1)) <= '1';
        control_array(0,(159 - 32 - 2)) <= '1';
        control_array(0,(159 - 32 - 3)) <= '1';
      end if;


      -- update score_bit_2
      if (score_bit_2 = 0) then
        control_array(0,(159 - 28 - 0)) <= '0';
        control_array(0,(159 - 28 - 1)) <= '1';
        control_array(0,(159 - 28 - 2)) <= '1';
        control_array(0,(159 - 28 - 3)) <= '0';
      end if;
      if (score_bit_2 = 1) then
        control_array(0,(159 - 28 - 0)) <= '0';
        control_array(0,(159 - 28 - 1)) <= '1';
        control_array(0,(159 - 28 - 2)) <= '1';
        control_array(0,(159 - 28 - 3)) <= '1';
      end if;
      if (score_bit_2 = 2) then
        control_array(0,(159 - 28 - 0)) <= '1';
        control_array(0,(159 - 28 - 1)) <= '0';
        control_array(0,(159 - 28 - 2)) <= '0';
        control_array(0,(159 - 28 - 3)) <= '0';
      end if;
      if (score_bit_2 = 3) then
        control_array(0,(159 - 28 - 0)) <= '1';
        control_array(0,(159 - 28 - 1)) <= '0';
        control_array(0,(159 - 28 - 2)) <= '0';
        control_array(0,(159 - 28 - 3)) <= '1';
      end if;
      if (score_bit_2 = 4) then
        control_array(0,(159 - 28 - 0)) <= '1';
        control_array(0,(159 - 28 - 1)) <= '0';
        control_array(0,(159 - 28 - 2)) <= '1';
        control_array(0,(159 - 28 - 3)) <= '0';
      end if;
      if (score_bit_2 = 5) then
        control_array(0,(159 - 28 - 0)) <= '1';
        control_array(0,(159 - 28 - 1)) <= '0';
        control_array(0,(159 - 28 - 2)) <= '1';
        control_array(0,(159 - 28 - 3)) <= '1';
      end if;
      if (score_bit_2 = 6) then
        control_array(0,(159 - 28 - 0)) <= '1';
        control_array(0,(159 - 28 - 1)) <= '1';
        control_array(0,(159 - 28 - 2)) <= '0';
        control_array(0,(159 - 28 - 3)) <= '0';
      end if;
      if (score_bit_2 = 7) then
        control_array(0,(159 - 28 - 0)) <= '1';
        control_array(0,(159 - 28 - 1)) <= '1';
        control_array(0,(159 - 28 - 2)) <= '0';
        control_array(0,(159 - 28 - 3)) <= '1';
      end if;
      if (score_bit_2 = 8) then
        control_array(0,(159 - 28 - 0)) <= '1';
        control_array(0,(159 - 28 - 1)) <= '1';
        control_array(0,(159 - 28 - 2)) <= '1';
        control_array(0,(159 - 28 - 3)) <= '0';
      end if;
      if (score_bit_2 = 9) then
        control_array(0,(159 - 28 - 0)) <= '1';
        control_array(0,(159 - 28 - 1)) <= '1';
        control_array(0,(159 - 28 - 2)) <= '1';
        control_array(0,(159 - 28 - 3)) <= '1';
      end if;


      pacman_pos_row_old <= pacman_pos_row_new;    -- update pacman_pos_row_new
      pacman_pos_col_old <= pacman_pos_col_new;
      ghost_pos_row_old <= ghost_pos_row_new;    -- update ghost_pos_row_new
      ghost_pos_col_old <= ghost_pos_col_new;
      control_array_V := (Vcount - (VSYNC + VBACK_PORCH)) srl 4;
      control_array_H := (Hcount - (HSYNC + HBACK_PORCH)) srl 4;
      sprite_array_V := (Vcount - (VSYNC + VBACK_PORCH)) AND "0000001111";
      sprite_array_H := (Hcount - (HSYNC + HBACK_PORCH)) AND "0000001111";


      -- brick (code: 1111)
      if (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 0))) = '1' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '1' and
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 2))) = '1' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '1' and
control_array_V <= 27) then
       sprite_brick_row := sprite_brick(to_integer(unsigned(sprite_array_V)));
       if (sprite_brick_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
          VGA_R <= "0000000000";
          VGA_G <= "0000000000";
          VGA_B <= "1111111111";
       else
          VGA_R <= "0000000000";
          VGA_G <= "0000000000";
          VGA_B <= "0000000000";
       end if;

      -- pacman (code: 0001)
      elsif (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 0))) = '0' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '0' and
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 2))) = '0' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '1' and
control_array_V <= 27) then
       sprite_pacman_row := sprite_pacman(to_integer(unsigned(sprite_array_V)));
       if (sprite_pacman_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
          VGA_R <= "1111111111";
          VGA_G <= "1111111111";
          VGA_B <= "0000000000";
       else
          VGA_R <= "0000000000";
          VGA_G <= "0000000000";
          VGA_B <= "0000000000";
       end if;

      -- dot_small (code: 0010)
      elsif (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 0))) = '0' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '0' and
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 2))) = '1' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '0' and
control_array_V <= 27) then
       sprite_dot_small_row := sprite_dot_small(to_integer(unsigned(sprite_array_V)));
       if (sprite_dot_small_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
          VGA_R <= "1111111111";
          VGA_G <= "1111111111";
          VGA_B <= "1111111111";
        else
          VGA_R <= "0000000000";
          VGA_G <= "0000000000";
          VGA_B <= "0000000000";
        end if;

      -- dot_large (code: 0011)
      elsif (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 0))) = '0' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '0' and
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 2))) = '1' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '1' and
control_array_V <= 27) then
       sprite_dot_large_row := sprite_dot_large(to_integer(unsigned(sprite_array_V)));
       if (sprite_dot_large_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
          VGA_R <= "0000000000";
          VGA_G <= "1111111111";
          VGA_B <= "0000000000";
       else
          VGA_R <= "0000000000";
          VGA_G <= "0000000000";
          VGA_B <= "0000000000";
       end if;

      -- ghost (code: 0100)
      elsif (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 0))) = '0' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '1' and
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 2))) = '0' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '0' and
control_array_V <= 27) then
       sprite_ghost_R_row := sprite_ghost_R(to_integer(unsigned(sprite_array_V)));
       sprite_ghost_G_row := sprite_ghost_G(to_integer(unsigned(sprite_array_V)));
       sprite_ghost_B_row := sprite_ghost_B(to_integer(unsigned(sprite_array_V)));
       if (sprite_ghost_R_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
          VGA_R <= "1111111111";
       else
          VGA_R <= "0000000000";
       end if;
       if (sprite_ghost_G_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
          VGA_G <= "1111111111";
       else
          VGA_G <= "0000000000";
       end if;
       if (sprite_ghost_B_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
          VGA_B <= "1111111111";
        else
           VGA_B <= "0000000000";
        end if;

      -- letter 'S'
      elsif (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 0))) = '0' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '0' and
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 2))) = '0' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '1' and
control_array_V = 29) then
          sprite_S_row := sprite_S(to_integer(unsigned(sprite_array_V)));
        if (sprite_S_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
           VGA_R <= "1111111111";
           VGA_G <= "1111111111";
           VGA_B <= "1111111111";
        else
           VGA_R <= "0000000000";
           VGA_G <= "0000000000";
           VGA_B <= "0000000000";
        end if;

      -- letter 'C'
      elsif (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 0))) = '0' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '0' and
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 2))) = '1' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '0' and
control_array_V = 29) then
          sprite_C_row := sprite_C(to_integer(unsigned(sprite_array_V)));
        if (sprite_C_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
           VGA_R <= "1111111111";
           VGA_G <= "1111111111";
           VGA_B <= "1111111111";
        else
           VGA_R <= "0000000000";
           VGA_G <= "0000000000";
           VGA_B <= "0000000000";
        end if;

      -- letter 'O'
      elsif (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 0))) = '0' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '0' and
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 2))) = '1' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '1' and
control_array_V = 29) then
          sprite_O_row := sprite_O(to_integer(unsigned(sprite_array_V)));
        if (sprite_O_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
           VGA_R <= "1111111111";
           VGA_G <= "1111111111";
           VGA_B <= "1111111111";
        else
           VGA_R <= "0000000000";
           VGA_G <= "0000000000";
           VGA_B <= "0000000000";
        end if;

      -- letter 'R'
      elsif (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 0))) = '0' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '1' and
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 2))) = '0' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '0' and
control_array_V = 29) then
          sprite_R_row := sprite_R(to_integer(unsigned(sprite_array_V)));
        if (sprite_R_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
           VGA_R <= "1111111111";
           VGA_G <= "1111111111";
           VGA_B <= "1111111111";
```

```
        else
           VGA_R <= "0000000000";
           VGA_G <= "0000000000";
           VGA_B <= "0000000000";
        end if;


      -- letter 'E'
      elsif (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 - 
(control_array_H sll 2) - 0))) = '0' and control_array(to_integer(unsigned(29 - 
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '1' and 
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 - 
(control_array_H sll 2) - 2))) = '0' and control_array(to_integer(unsigned(29 - 
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '1' and 
control_array_V = 29) then
         sprite_E_row := sprite_E(to_integer(unsigned(sprite_array_V)));
        if (sprite_E_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
           VGA_R <= "1111111111";
           VGA_G <= "1111111111";
           VGA_B <= "1111111111";
        else
           VGA_R <= "0000000000";
           VGA_G <= "0000000000";
           VGA_B <= "0000000000";
        end if;

      -- letter '0'
      elsif (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 - 
(control_array_H sll 2) - 0))) = '0' and control_array(to_integer(unsigned(29 - 
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '1' and 
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 - 
(control_array_H sll 2) - 2))) = '1' and control_array(to_integer(unsigned(29 - 
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '0' and 
control_array_V = 29) then
         sprite_0_row := sprite_0(to_integer(unsigned(sprite_array_V)));
        if (sprite_0_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
           VGA_R <= "1111111111";
           VGA_G <= "0000000000";
           VGA_B <= "1111111111";
        else
           VGA_R <= "0000000000";
           VGA_G <= "0000000000";
           VGA_B <= "0000000000";
        end if;

      -- letter '1'
      elsif (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 - 
(control_array_H sll 2) - 0))) = '0' and control_array(to_integer(unsigned(29 - 
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '1' and 
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 - 
(control_array_H sll 2) - 2))) = '1' and control_array(to_integer(unsigned(29 - 
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '1' and 
control_array_V = 29) then
         sprite_1_row := sprite_1(to_integer(unsigned(sprite_array_V)));
        if (sprite_1_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
           VGA_R <= "1111111111";
           VGA_G <= "0000000000";
           VGA_B <= "1111111111";
        else
           VGA_R <= "0000000000";
           VGA_G <= "0000000000";
           VGA_B <= "0000000000";
        end if;

      -- letter '2'
      elsif (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 - 
(control_array_H sll 2) - 0))) = '1' and control_array(to_integer(unsigned(29 - 
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '0' and 
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 - 
(control_array_H sll 2) - 2))) = '0' and control_array(to_integer(unsigned(29 - 
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '0' and 
control_array_V = 29) then
         sprite_2_row := sprite_2(to_integer(unsigned(sprite_array_V)));
        if (sprite_2_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
```

```
           VGA_R <= "1111111111";
           VGA_G <= "0000000000";
           VGA_B <= "1111111111";
        else
           VGA_R <= "0000000000";
           VGA_G <= "0000000000";
           VGA_B <= "0000000000";
        end if;

      -- letter '3'
      elsif (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 0))) = '1' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '0' and
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 2))) = '0' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '1' and
control_array_V = 29) then
        sprite_3_row := sprite_3(to_integer(unsigned(sprite_array_V)));
        if (sprite_3_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
           VGA_R <= "1111111111";
           VGA_G <= "0000000000";
           VGA_B <= "1111111111";
        else
           VGA_R <= "0000000000";
           VGA_G <= "0000000000";
           VGA_B <= "0000000000";
        end if;

      -- letter '4'
      elsif (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 0))) = '1' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '0' and
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 2))) = '1' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '0' and
control_array_V = 29) then
        sprite_4_row := sprite_4(to_integer(unsigned(sprite_array_V)));
        if (sprite_4_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
           VGA_R <= "1111111111";
           VGA_G <= "0000000000";
           VGA_B <= "1111111111";
        else
           VGA_R <= "0000000000";
           VGA_G <= "0000000000";
           VGA_B <= "0000000000";
        end if;

      -- letter '5'
      elsif (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 0))) = '1' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '0' and
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 2))) = '1' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '1' and
control_array_V = 29) then
        sprite_5_row := sprite_5(to_integer(unsigned(sprite_array_V)));
        if (sprite_5_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
           VGA_R <= "1111111111";
           VGA_G <= "0000000000";
           VGA_B <= "1111111111";
        else
           VGA_R <= "0000000000";
           VGA_G <= "0000000000";
           VGA_B <= "0000000000";
        end if;

      -- letter '6'
      elsif (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 0))) = '1' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '1' and
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 2))) = '0' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '0' and
control_array_V = 29) then
```

```
         sprite_6_row := sprite_6(to_integer(unsigned(sprite_array_V)));
        if (sprite_6_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
           VGA_R <= "1111111111";
           VGA_G <= "0000000000";
           VGA_B <= "1111111111";
        else
           VGA_R <= "0000000000";
           VGA_G <= "0000000000";
           VGA_B <= "0000000000";
        end if;


      -- letter '7'
      elsif (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 0))) = '1' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '1' and
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 2))) = '0' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '1' and
control_array_V = 29) then
         sprite_7_row := sprite_7(to_integer(unsigned(sprite_array_V)));
        if (sprite_7_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
           VGA_R <= "1111111111";
           VGA_G <= "0000000000";
           VGA_B <= "1111111111";
        else
           VGA_R <= "0000000000";
           VGA_G <= "0000000000";
           VGA_B <= "0000000000";
        end if;

      -- letter '8'
      elsif (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 0))) = '1' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '1' and
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 2))) = '1' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '0' and
control_array_V = 29) then
         sprite_8_row := sprite_8(to_integer(unsigned(sprite_array_V)));
        if (sprite_8_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
           VGA_R <= "1111111111";
           VGA_G <= "0000000000";
           VGA_B <= "1111111111";
        else
           VGA_R <= "0000000000";
           VGA_G <= "0000000000";
           VGA_B <= "0000000000";
        end if;

      -- letter '9'
      elsif (control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 0))) = '1' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 1))) = '1' and
control_array(to_integer(unsigned(29 - control_array_V)),to_integer(unsigned(159 -
(control_array_H sll 2) - 2))) = '1' and control_array(to_integer(unsigned(29 -
control_array_V)),to_integer(unsigned(159 - (control_array_H sll 2) - 3))) = '1' and
control_array_V = 29) then
         sprite_9_row := sprite_9(to_integer(unsigned(sprite_array_V)));
        if (sprite_9_row(to_integer(unsigned(15 - sprite_array_H))) = '1') then
           VGA_R <= "1111111111";
           VGA_G <= "0000000000";
           VGA_B <= "1111111111";
        else
           VGA_R <= "0000000000";
           VGA_G <= "0000000000";
           VGA_B <= "0000000000";
        end if;


      elsif vga_hblank = '0' and vga_vblank ='0' then
         VGA_R <= "0000000000";
         VGA_G <= "0000000000";
         VGA_B <= "0000000000";
      else
        VGA_R <= "0000000000";
        VGA_G <= "0000000000";
        VGA_B <= "0000000000";
      end if;
    end if;
  end process VideoOut;

  VGACLK_GEN: process(clk)
begin
  if rising_edge(clk) then
       clk_counter <= not(clk_counter);
  end if;
end process VGACLK_GEN;

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

File_2: hello_world.c

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

int main()
{
  int row_old_pacman = 20;
  int col_old_pacman = 20;
  int row_new_pacman = 20;
  int col_new_pacman = 20;
  int row_old_ghost = 13;
  int col_old_ghost = 14;
  int row_new_ghost = 13;
  int col_new_ghost = 14;
  int recover_sprite = 0;
  unsigned char code;
  int score = 0;
  int score_bit_0;
  int score_bit_1;
  int score_bit_2;
  int count = 0;

  // pacman        : 0001 (0x1)
  // brick         : 1111 (0xf)
  // background    : 0000 (0x0)
  // dot_small     : 0010 (0x2)
  // dot_large     : 0011 (0x3)
  // ghost         : 0100 (0x4)
  //                          0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39
  //                          |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
  int control_array[30][40] =
{0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,
0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,   // 0

0xf,0x3,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0xf,0xf,0x2,0x2,0x2,0
x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x3,0xf,   // 1

0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0
xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,   // 2

0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0
xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,   // 3
```

```
0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0
xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,   // 4

0xf,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0
x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0xf,   // 5

0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0
xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,   // 6

0xf,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0
xf,0xf,0xf,0x2,0xf,0xf,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0xf,   // 7

0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0xf,0xf,0x2,0x2,0x2,0
x2,0x2,0x2,0x2,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,   // 8

0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0
xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,   // 9

0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0
x2,0x2,0x2,0x2,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,   //10

0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0x0,0x0,0xf,0xf,0xf,0
xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,   //11

0xf,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0xf,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0
x0,0x0,0xf,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0xf,   //12

0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0x4,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0
x0,0x4,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,   //13

0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0
xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,   //14

0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0
x2,0x2,0x2,0x2,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,   //15

0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0
xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,   //16

0xf,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0xf,0xf,0x2,0x2,0x2,0
x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0xf,   //17

0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0
xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,   //18

0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0
xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,   //19

0xf,0x2,0x2,0x2,0x2,0x2,0xf,0xf,0xf,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x1,0x2,0x2,0x2,0
x2,0x2,0x2,0x2,0x2,0x2,0x2,0xf,0xf,0xf,0x2,0x2,0x2,0x2,0x2,0xf,   //20

0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0
xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,   //21

0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0
xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0xf,0x2,0xf,0xf,0xf,0xf,0xf,   //22

0xf,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0xf,0xf,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0xf,0xf,0x2,0x2,0x2,0
x2,0x2,0x2,0x2,0xf,0xf,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0xf,   //23

0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0
xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,   //24

0xf,0x2,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,0xf,0x2,0xf,0xf,0
xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0x2,0xf,   //25

0xf,0x3,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0
x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x2,0x3,0xf,   //26

0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0
xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,0xf,   //27
```

```
0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0
x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,   //28

0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0
x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0 }; //29

  printf("Ready to read the PS2 keyboard!\n");

  for (;;)
  {
      while (!IORD_8DIRECT(PS2_BASE,0))
      {
        IOWR_16DIRECT(VGA_BASE, 0, row_new_pacman);
        IOWR_16DIRECT(VGA_BASE, 2, col_new_pacman);
        IOWR_16DIRECT(VGA_BASE, 4, score_bit_0);
        IOWR_16DIRECT(VGA_BASE, 6, score_bit_1);
        IOWR_16DIRECT(VGA_BASE, 8, score_bit_2);
        IOWR_16DIRECT(VGA_BASE, 10, row_new_ghost);
        IOWR_16DIRECT(VGA_BASE, 12, col_new_ghost);
        IOWR_16DIRECT(VGA_BASE, 14, recover_sprite);

        if (count == 0)
        {
            if (row_new_pacman > row_new_ghost)
                if (control_array[row_new_ghost+1][col_new_ghost] != 0xf)  // check if wall is
hit
                    row_new_ghost++;
            if (row_new_pacman < row_new_ghost)
                if (control_array[row_new_ghost-1][col_new_ghost] != 0xf)  // check if wall is
hit
                    row_new_ghost--;
            if (col_new_pacman > col_new_ghost)
                if (control_array[row_new_ghost][col_new_ghost+1] != 0xf)  // check if wall is
hit
                    col_new_ghost++;
            if (col_new_pacman < col_new_ghost)
                if (control_array[row_new_ghost][col_new_ghost-1] != 0xf)  // check if wall is
hit
                    col_new_ghost--;
        }
        count++;
        if (count == 30000)
            count = 0;

      }
      code = IORD_8DIRECT(PS2_BASE,4);

      // LEFT  ARROW: 224 107 224 240 107
      // RIGHT ARROW: 224 116 224 240 116
      // UP    ARROW: 224 117 224 240 117
      // DOWN  ARROW: 224 114 224 240 114

      if (code == 224)
      {
        while (!IORD_8DIRECT(PS2_BASE,0))
        {
          IOWR_16DIRECT(VGA_BASE, 0, row_new_pacman);
          IOWR_16DIRECT(VGA_BASE, 2, col_new_pacman);
          IOWR_16DIRECT(VGA_BASE, 4, score_bit_0);
          IOWR_16DIRECT(VGA_BASE, 6, score_bit_1);
          IOWR_16DIRECT(VGA_BASE, 8, score_bit_2);
          IOWR_16DIRECT(VGA_BASE, 10, row_new_ghost);
          IOWR_16DIRECT(VGA_BASE, 12, col_new_ghost);
          IOWR_16DIRECT(VGA_BASE, 14, recover_sprite);
        }
        code = IORD_8DIRECT(PS2_BASE,4);
        switch (code)
        {
            case 107:
                if (control_array[row_new_pacman][col_new_pacman-1] != 0xf)  // check if wall is
hit
                {
                    col_new_pacman--;
                }
                break;

            case 116:
                if (control_array[row_new_pacman][col_new_pacman+1] != 0xf)  // check if wall is
hit
                {
                    col_new_pacman++;
                }
                break;

            case 117:
                if (control_array[row_new_pacman-1][col_new_pacman] != 0xf)  // check if wall is
hit
                {
                    row_new_pacman--;
                }
                break;

            case 114:
                if (control_array[row_new_pacman+1][col_new_pacman] != 0xf)  // check if wall is
hit
                {
                    row_new_pacman++;
                }
                break;

            default :
                break;
        }
      }
      IOWR_16DIRECT(VGA_BASE, 0, row_new_pacman);
      IOWR_16DIRECT(VGA_BASE, 2, col_new_pacman);

      if (control_array[row_new_pacman][col_new_pacman] == 0x2)  // eat a small dot
      {
        score = score + 1;
      }
      if (control_array[row_new_pacman][col_new_pacman] == 0x3)  // eat a large dot
      {
        score = score + 5;
      }

      score_bit_0 = score%10;
      score_bit_1 = (score/10)%10;
      score_bit_2 = score/100;

      IOWR_16DIRECT(VGA_BASE, 4, score_bit_0);
      IOWR_16DIRECT(VGA_BASE, 6, score_bit_1);
      IOWR_16DIRECT(VGA_BASE, 8, score_bit_2);
      IOWR_16DIRECT(VGA_BASE, 10, row_new_ghost);
      IOWR_16DIRECT(VGA_BASE, 12, col_new_ghost);
      printf("******* score = %d; %d,%d,%d\n", score, score_bit_2, score_bit_1, score_bit_0);

      control_array[row_old_pacman][col_old_pacman] = 0x0;
      control_array[row_new_pacman][col_new_pacman] = 0x1;

      row_old_pacman = row_new_pacman;
      col_old_pacman = col_new_pacman;

      if (row_old_ghost != row_new_ghost || col_old_ghost != col_new_ghost)
      {
          IOWR_16DIRECT(VGA_BASE, 14, recover_sprite);
          control_array[row_old_ghost][col_old_ghost] = recover_sprite;
          recover_sprite = control_array[row_new_ghost][col_new_ghost];
          control_array[row_new_ghost][col_new_ghost] = 0x4;
          row_old_ghost = row_new_ghost;
          col_old_ghost = col_new_ghost;
      }
  }

  return 0;
}
```