Project Final Report
May 11, 2004

Group Members:
Charles Elliot Finkel (cef19)
Dagna Harasim (dh2106)
David Soofian (ds2133)

Table of Contents

1. Introduction 2
2. The Design 3
   2.1 The Game 3
   2.2 Software 3
   2.3 Hardware 4
3. Conclusion 7
4. Lessons Learned 8
   4.1 Dagna Harasim 8
   4.2 David Soofian 8
5. The Code 10
   5.1 Software 10
      5.1.2 drawing.c 10
      5.1.3 drawing.h 17
      5.1.4 game_engine.c 24
      5.1.5 isr.c 37
      5.1.6 main.c 38
      5.1.7 main.h 40
   5.2 Hardware 41
      5.2.1 display_prio.vhd 41
      5.2.2 font.vhd 43
      5.2.3 sprite.vhd 46
      5.2.4 sprite_loader.vhd 48
      5.2.5 sprite_video.vhd 51
      5.2.6 vga.vhd 55
      5.3.7 vga_timing.vhd 63
      5.2.8 system.mhs 70

1. Introduction

Our project has been designed to incorporate our knowledge of previous Embedded Systems labs as well as calls on us to implement new hardware, new features and integrated C-code for a fully working game. The project demonstrates our knowledge of the FPGA, its components, and VHDL and C.

We have designed a Pac-Man video game based on the original, well known game. Our Pac-Man has been implemented without any third party software. We have decided to use sprites to display our characters on the screen, giving them more fluid motion and multiple colors. The game engine has been implemented in C and the graphical portion has been implemented in VHDL. The project relies on the integration of both software and hardware. With both working, together we will be able to handle inputs from the keyboard, RAM, ROM, SRAM and output to the video display.

This design helps to mimic the original Pac-Man and the way it used hardware for most of its gaming console. With the hardware implemented we can virtually make any 1980's style arcade game. It is the basic function of sprite movements and character ROMs that made these games popular.

2. The Design

2.1 The Game

The game itself is a slightly modified version of the original Pac-Man game. Pac-man has gone through a slight make over and is now bigger and has an eye. The three ghosts are also bigger, re-designed and of different colors. The maze is different than the original version, allowing the bigger characters to move around smoothly. In our version, Pac-Man can eat two pallets at a time and the score he earns for the eaten pellets is displayed on the lower left corner of the screen. Just like in the original game, Pac-Man must avoid the ghost on his quest to eat all the pellets on the screen or face termination. We allow 3 lives for Pac-Man before the game is over.

[insert game picture here]

2.2 Software

The game engine has been implemented in C. It chooses in advance which graphic will be displayed for each individual sprite so that the appropriate graphic can be displayed at the next movement. We have four allocated spaces for sprites; this allows us to display four separate game characters on the screen at the same time. Every time a sprite moves the graphic is changed to make the characters look more realistic. Bitmaps of the game characters were created and stored as arrays in C. The graphic is sent from microblaze to the SRAM to be stored starting at memory location 0x4000. At this point the hardware takes over to display the graphics on the screen.

The vast majority of the code for the game is contained in two files: game_engine.c and drawing.c. Game_engine.c consists of all control and bookkeeping for the game, while drawing.c handles all drawing to the screen. That includes both the characters from ROM and the sprites. The characters from ROM are used for walls, pellets, and numbers and letters.

For each iteration of the main loop of the game engine, Pac-Man and the ghosts are allowed to move one pixel, but Pac-Man can only change directions every 8 pixels (or every one character length), and the ghosts can change direction every 4 pixels. There is a separate count for Pac-Man than there is for ghosts, and specifically, when Pac-Man hits a wall, the game characters will likely lose synchronization.

Since Pac-Man is two by two characters wide, and each pellet takes up one character, he can eat two pallets at a time. The game has wall detection for Pac-Man and the ghosts. There is hit detection if Pac-Man touches a ghost but for now instead of dying he only reappears at the top left of the screen. Ghost AI is limited to simple random movement.

2.3 Hardware

In our labs, we have learned to implement a live video display. We were able to store our character sets and graphics on the FPGA's RAM. This allowed us to save valuable time in drawing each character for the TV typewriter one by one by calling pre-existing mapped graphics. We used this part of our lab to display the maze and, the score and other background graphics on the screen. The characters used for the background are eight by eight pixels in size, allowing one bit per pixel.

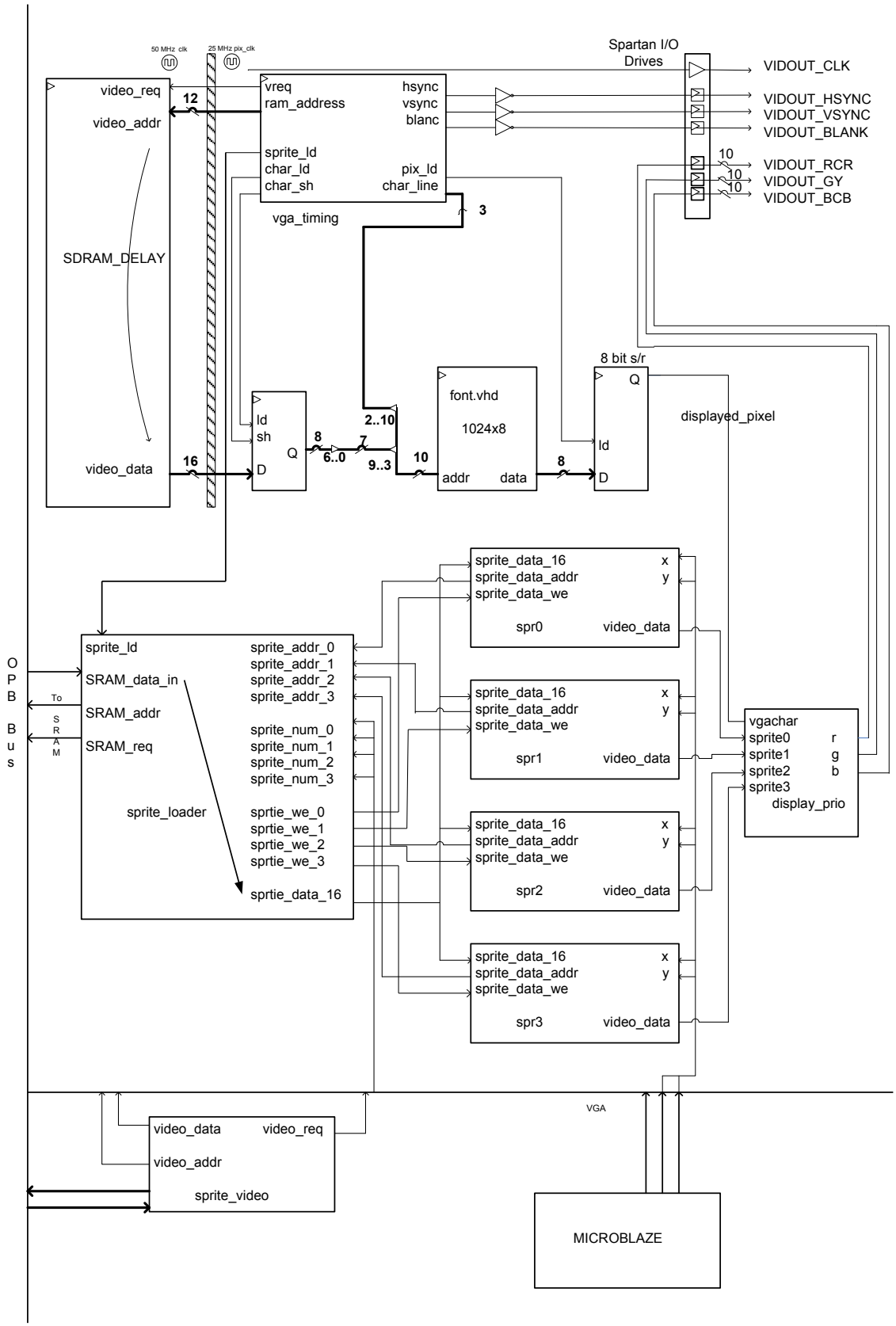
We are using sprites to display the moving characters on the screen. Each sprite moves one pixel at a time creating a smooth movement on the screen. The sprites are sixteen by sixteen pixels in size, allowing 4 bits per pixel. These four bits represent one of the sixteen colors we have chosen as our palette. This allows us to use multiple colors for Pac-Man and the ghosts. Another advantage to using sprites is their independent movements from each other.

Each sprite is composed of 16 lines, 64 bits each. Since SRAM allows for only 16 bits to be loaded at a time, each line of sprite can be visualized as a line divided into quarters. Loading and keeping track of the line quarters is taken care of in sprite_loader. Since each line of the sprite needs to be loaded ahead of time the loading is done during blanking. When sprite_loader receives a signal from vga_timing telling it when blanking is occurring, it sends a request to SRAM for a sprite graphic. As sprite_loader sends the graphic data to the appropriate instance of sprite (16 bits at a time) it also send the information telling the sprite which quarter is currently begin loaded. In return sprite sends back the number of the line currently loaded. This data, along with the sprite number and the number of the quarter in the line makes up the address in the SRAM where the appropriate sprite line is located.

The sprite component checks with vga_timing to see whether the display is in the location the sprite needs to be placed on the screen. If display is in the "active" region the lower 4 bits of the graphic data are sent to the display_prio for color decoding, and data is shifted by 4 bits. Display_prio contains two multiplexers that take care of deciding which graphic is sent to the screen and to decode the colors of the graphic. The first 5x1 MUX gives priority to the sprites over the data from the character ROM (font.vhd). Also, there is an order among the sprites, simply to create order during the displaying. The second MUX, 16x1, decodes the colors from each pixel of the sprite. The 16 colors were chosen based on our need.

Throughout the implementation, vga_timing tracks the time and the address currently displayed on the screen. It is important to keep tack of the horizontal and vertical lines on the screen in order to synchronize loading and displaying. This component sends signals to various other components within the VGA, to signal when appropriate action needs to be taken. For more information refer to the block diagram on page 6.

All of the components are connected in the VGA module, which is connected to OPB Bus by sprite_video. Sprite-video receives the information from microblaze (when microblaze is accessing an address 0xFEFD0000 to 0xFEFD1111) regarding the sprite number and it's location on the screen, and passes it to the VGA for processing.


50 MHz clk
25 MHz pix_clk
Spartan I/O Drives
VIDOUT_CLK
VIDOUT_HSYNC
VIDOUT_VSYNC
VIDOUT_BLANK
VIDOUT_RCR
VIDOUT_GY
VIDOUT_BCB
video_req
video_addr
SDRAM_DELAY
video_data
vreq
ram_address
sprite_ld
char_ld
char_sh
hsync
vsync
blanc
pix_ld
char_line
vga_timing
12
3
16
10
ld
sh
D
Q
8
6..0
7
9..3
2..10
font.vhd
1024x8
addr
data
8 bit s/r
displayed_pixel
sprite_data_16
sprite_data_addr
sprite_data_we
x
y
spr0
spr1
spr2
spr3
video_data
sprite_ld
SRAM_data_in
SRAM_addr
SRAM_req
sprite_loader
sprite_addr_0
sprite_addr_1
sprite_addr_2
sprite_addr_3
sprite_num_0
sprite_num_1
sprite_num_2
sprite_num_3
sprtie_we_0
sprtie_we_1
sprtie_we_2
sprtie_we_3
sprtie_data_16
OPB Bus
To SRAM
vgachar
sprite0
sprite1
sprite2
sprite3
r
g
b
display_prio
VGA
video_req
sprite_video
MICROBLAZE

3. Conclusion

We believe that the project was completed successfully. Although the AI is very simple and creates "dumb" ghosts, and the background has only one color, the game works well and has nice character graphics. We never imagined that creating Pac-Man would be so hard and time consuming. At first we didn't consider using sprites, as a matter of fact, we did not know what a sprite was until Professor Edwards mentioned it as a noteworthy change to our project. At that time we also learned of the option of using a BRAM, though after tireless nights and endless hours the idea of using SRAM sufficed..

We would like to give special thanks to our TA, Cristian Soviani. Without his help and dedication we would have not been able to complete the project. Cristian spent endless hours with our group, as well with others, explaining how the hardware works. He greatly contributed to our learning.

4. Lessons Learned

This project has taught us many things about the actual material, project design and implementation process, and about ourselves. Each of us learned something different and we would like to share the lessons we have learned during the creation of this project. Hopefully this section will act as a good advice to future teams.

4.1 Dagna Harasim

Throughout the production of the project I have learned that careful planning and a concrete design is a big part of a successful projects. It is important to research different methods that can be used to implement a project. Our design was changed three times until we finally decided on how to implement it. It was frustrating to start over and over, but in the end I have learned of the different methods of solving hardware problems. Coming into the class I had no knowledge of VHDL, but through the labs and mainly thanks to this project I gained better understanding of hardware and VHDL. I also learned how important timing is in the hardware design.

One thing our group failed to do was to designate a project manager who would oversee the progress. This created some chaos, which we fortunately overcame. Also we should have done a better job of dividing the work among each other, because in the end some people had to take on more responsibility than others. Although sometimes we don't have much choice in our project partners, it is important to choose members of the team wisely, so that the team consists of people who have skill in at least one area of the project.

My advice to future teams is to research, designate someone who will manage the project, create a timeline that will help you keep track of everyone's progress, and to not be afraid to ask for help. The professor and the TA are there to assist, and most likely have a lot more experience than you. And remember, start early!

4.2 David Soofian

In the production of this project I have learned many things. In constructing this project it seemed as if creating hardware was a simple task. VHDL has many restrictions from hardware which make that task a lot harder. Furthermore with each restriction we had come across, a new design had to be drawn up, sometimes changing the project minutely and sometimes making an overhaul of the complete work. Once all the hardware restrictions and specifications were known timing became the largest intricacy needed.

Coding in VHDL, like any coding language, just takes time to learn, but with each language comes its own issues. The actual coding was not really an issue; I learnt that timing and restrictions become your largest concerns in a project of this form. My focus was on hardware so the C code, although I supervised some of it because I have skills in

C, was not my concern. I did take time however to try and help with the function which C and VHDL would share.

In summation, I have learnt that projects like these take enormous amounts of time, calling on epiphany-like clarity to remember signal names and mappings between files and precision in project design. The project design involves research and trial and error but once that is done the process is much faster. Lastly, but most importantly, it is not the quantity of coding that makes the work worthwhile, it is the style of coding and the personalization of ideas and methods.

5. The Code

The following is a copy of the code used to implement this project. Charles Finkel was responsible for the software and Dagna Harasim and David Soofian created the hardware. Dagna Harasim also created the sprite graphics. All members of the team took part in testing.

5.1 Software

5.1.2 drawing.c

```
// code created by Charles Finkel

#include "drawing.h"

/* Places a sprite onto video. Variable sn represents one of the 4
   sprites that can be displayed at the same time.
   npic represents one of the 129 different pictures that can be displayed
   x can go from 0 to 640
   y can go from 0 to 480
 */
void place_sprite(int sn, int npic, int x, int y)
{
  XIo_Out16(SPRITE_XY + (sn * 16) + 0, x);
  XIo_Out16(SPRITE_XY + (sn * 16) + 4, y);
  XIo_Out16(SPRITE_XY + (sn * 16) + 8, npic);
}

/**
   Erases  trail left by pacman
   Updates mapparray, and video
*/
void erase_pacman_trail(int x, int y) {
  maparray[x][y] = SPACE;
  XIo_Out8(VGA_START + x + y*WC, SPACE);
  maparray[x+1][y] = SPACE;
  XIo_Out8(VGA_START + (x+1) + y*WC, SPACE);
  maparray[x][y+1] = SPACE;
  XIo_Out8(VGA_START + x + (y+1)*WC, SPACE);
  maparray[x+1][y+1] = SPACE;
  XIo_Out8(VGA_START + (x+1) + (y+1)*WC, SPACE);

}
```

```
/**
   Insert wall function
   This function will insert a wall into the maparray
   Orientation can be either horizontal or vertical
   Start_x and start_y are the starting positions of the wall
   Length determines how long the wall is starting from the start position and
   either downward or to the right, depending on orientation.  If the length
   is longer than the screen, the wall will just get cut off
   returns 1 for success, 0 for failure
**/
int insert_wall(int orientation, int length, int start_x, int start_y) {
  if ((start_x >= W_GAME) || (start_y >= H_GAME) || (orientation > 1))
    return 0;
  if (orientation == HORIZONTAL) {
    int x = start_x;
    while ((x < W_GAME) && (x - start_x < length)) {
      maparray[x][start_y] = WALL_H;
      x ++;
    }
  }
  else { // orientation == VERTICAL
    int y = start_y;
    while ((y < H_GAME) && (y - start_y < length)) {
      maparray[start_x][y] = WALL_V;
      y ++;
    }
  }
}

/**
   inserts wall with orientation vertical or horizontal,
   start point start, and end point end along axis specified in orientaiton
   other_coord is the point on the other axis
*/
int insert_wall2(int orientation, int start, int end, int other_coord) {
  switch (orientation) {
  case HORIZONTAL:
    insert_wall(orientation,end-start+1,start,other_coord);
    break;
  case VERTICAL:
    insert_wall(orientation,end-start+1,other_coord,start);
    break;
  }
    return 0;
```

```
}

void fill_with_pills() {
  int x,y;
  pills = 0;
  for (x = 0; x < WC; x++) {
    for (y = 0; y < HC; y++) {
      if ((maparray[x][y] != WALL_H) && (maparray[x][y] != WALL_V)) {
           maparray[x][y] = PILL;
           pills++;
      }
    }
  }
}

void draw_other_walls2(int x, int y) {
  insert_wall2(HORIZONTAL,3+x,8+x,6+y);
  insert_wall2(VERTICAL,9+y,21+y,6+x);
  insert_wall2(VERTICAL,10+y,18+y,6+x);
  insert_wall2(HORIZONTAL,3+x,13+x,21+y);
  insert_wall2(VERTICAL,4+y,15+y,11+x);

  insert_wall2(HORIZONTAL,15+x,27+x,3+y);
  insert_wall2(HORIZONTAL,14+x,27+x,6+y);

  insert_wall2(VERTICAL,12+y,17+y,16+x);
  insert_wall2(VERTICAL,10+y,14+y,19+x);
  insert_wall2(VERTICAL,10+y,17+y,22+x);
  insert_wall2(HORIZONTAL,16+x,22+x,17+y);

  insert_wall2(HORIZONTAL,16+x,33+x,21+y);

  insert_wall2(VERTICAL,1+y,6+y,35+x);

  insert_wall2(VERTICAL,3+y,14+y,30+x);
  insert_wall2(HORIZONTAL,25+x,30+x,14+y);
  insert_wall2(HORIZONTAL,25+x,30+x,17+y);
  insert_wall2(VERTICAL,9+y,17+y,33+x);

  insert_wall2(HORIZONTAL,36+x,40+x,9+y);
  insert_wall2(HORIZONTAL,36+x,38+x,12+y);
  insert_wall2(HORIZONTAL,36+x,38+x,15+y);
  insert_wall2(HORIZONTAL,36+x,38+x,18+y);
  insert_wall2(HORIZONTAL,36+x,38+x,21+y);
  insert_wall2(VERTICAL,21+y,23+y,36+x);
```

```
}

// orientation, length, x(80), y(60)
void draw_other_walls() {
  draw_other_walls2(0,0);
  draw_other_walls2(38,0);
  draw_other_walls2(0,23);
  draw_other_walls2(38,23);
  draw_other_walls2(0,46);
  draw_other_walls2(38,46);


  /*
  insert_wall(HORIZONTAL,47,5,4);
  insert_wall(VERTICAL, 20,10 ,20);
  insert_wall(VERTICAL, 6,55,1);
  insert_wall(VERTICAL, 30, 7,3);
  insert_wall(HORIZONTAL, 40, 30, 46);
  insert_wall(VERTICAL, 33, 72,7);
  */


}

/**
    No fail check.  Draws ghost # ghost_num, at coord x,y, and position pos
*/
void draw_ghost(int ghost_num, int x, int y, int pos) {
  /*
  print("draw ghost: ghost #",ghost_num);
  putnum(ghost_num);
  print("\n\r");
  */
  place_sprite(ghost_num+1,GHOST_1_POS_1+(ghost_num*2)+pos,x,y);
}

/**
   Draws pacman.  Assumes walls have already been checked.  If they haven't,
   and pacman is overlapping a wall, then he will take just be on top of that
   wall and in essence eat it up (shouldn't happen)
   if open is 1, then pacman's mouth is open. else, it's closed
*/
void draw_pacman(int x, int y, int orientation, char open) {

  //  char *pacman[4];
  int pacman_pic;
```

```
  switch (orientation) {
  case RIGHT :
    pacman_pic = PACMAN_OPEN_RIGHT; break;
  case LEFT :
    pacman_pic = PACMAN_OPEN_LEFT; break;
  case UP :
    pacman_pic = PACMAN_OPEN_UP; break;
  case DOWN :
    pacman_pic = PACMAN_OPEN_DOWN; break;
  default: pacman_pic = PACMAN_OPEN_RIGHT; break;
  }

  if(!open) pacman_pic += 4;

  place_sprite(PACMAN_SPRITE, pacman_pic, x, y);
  /*
  XIo_Out8(VGA_START + x + y*WC, (*pacman)[0]);
  XIo_Out8(VGA_START + (x+1) + y*WC, (*pacman)[1]);
  XIo_Out8(VGA_START + x + (y+1)*WC, (*pacman)[2]);
  XIo_Out8(VGA_START + (x+1) + (y+1)*WC, (*pacman)[3]);
  */
}


void create_bitmap(unsigned int *array,int location) {
  int x;
  for(x=0;x<32;x++) {
    XIo_Out32(VGA_START + 0x4000 + location*32*4 + x*4, array[x]);
  }

}

void create_bitmaps() {
  create_bitmap((unsigned int *)&pacman_open_right_array,PACMAN_OPEN_RIGHT);
  create_bitmap((unsigned int *)&pacman_open_left_array,PACMAN_OPEN_LEFT);
  create_bitmap((unsigned int *)&pacman_open_up_array,PACMAN_OPEN_UP);
  create_bitmap((unsigned int
*)&pacman_open_down_array,PACMAN_OPEN_DOWN);
  create_bitmap((unsigned int
*)&pacman_closed_right_array,PACMAN_CLOSED_RIGHT);
  create_bitmap((unsigned int
*)&pacman_closed_left_array,PACMAN_CLOSED_LEFT);
  create_bitmap((unsigned int *)&pacman_closed_up_array,PACMAN_CLOSED_UP);
  create_bitmap((unsigned int
*)&pacman_closed_down_array,PACMAN_CLOSED_DOWN);
```

```
  create_bitmap((unsigned int *)&ghost_1_pos_1,GHOST_1_POS_1);
  create_bitmap((unsigned int *)&ghost_1_pos_2,GHOST_1_POS_2);
  create_bitmap((unsigned int *)&ghost_2_pos_1,GHOST_2_POS_1);
  create_bitmap((unsigned int *)&ghost_2_pos_2,GHOST_2_POS_2);
  create_bitmap((unsigned int *)&ghost_3_pos_1,GHOST_3_POS_1);
  create_bitmap((unsigned int *)&ghost_3_pos_2,GHOST_3_POS_2);
  /*
  for(x=0;x<32;x++) {
    XIo_Out32(VGA_START + 0x4000 + x*4, pacman_open_right_array[x]);
  }
  */
}


void setup_screen() {
  int x,y;

  //  required outside walls:
  insert_wall(HORIZONTAL,W_GAME,0,0);
  insert_wall(HORIZONTAL,W_GAME,0,H_GAME-1);
  insert_wall(VERTICAL,H_GAME,0,0);
  insert_wall(VERTICAL,H_GAME,W_GAME-1,0);

  //other walls
  draw_other_walls();

  //fill rest of map with pills:
  fill_with_pills();

  // show score and time:
     XIo_Out8(VGA_START + 3 + 58*WC, 'S');
     XIo_Out8(VGA_START + 4 + 58*WC, 'C');
     XIo_Out8(VGA_START + 5 + 58*WC, 'O');
     XIo_Out8(VGA_START + 6 + 58*WC, 'R');
     XIo_Out8(VGA_START + 7 + 58*WC, 'E');
     XIo_Out8(VGA_START + 8 + 58*WC, ':');
     XIo_Out8(VGA_START + 12 + 58*WC, '0');
     XIo_Out8(VGA_START + 31 + 58*WC, 'R');
     XIo_Out8(VGA_START + 32 + 58*WC, 'E');
     XIo_Out8(VGA_START + 33 + 58*WC, 'A');
     XIo_Out8(VGA_START + 34 + 58*WC, 'D');
     XIo_Out8(VGA_START + 35 + 58*WC, 'Y');
     XIo_Out8(VGA_START + 37 + 58*WC, '3');

     XIo_Out8(VGA_START + 42 + 58*WC, 'L');
     XIo_Out8(VGA_START + 43 + 58*WC, 'I');
```

```
    XIo_Out8(VGA_START + 44 + 58*WC, 'V');
    XIo_Out8(VGA_START + 45 + 58*WC, 'E');
    XIo_Out8(VGA_START + 46 + 58*WC, 'S');
    XIo_Out8(VGA_START + 47 + 58*WC, ':');
    XIo_Out8(VGA_START + 48 + 58*WC, ' ');
    XIo_Out8(VGA_START + 49 + 58*WC, '3');


    XIo_Out8(VGA_START + 60 + 58*WC, 'T');
    XIo_Out8(VGA_START + 61 + 58*WC, 'I');
    XIo_Out8(VGA_START + 62 + 58*WC, 'M');
    XIo_Out8(VGA_START + 63 + 58*WC, 'E');
    XIo_Out8(VGA_START + 64 + 58*WC, ':');
    XIo_Out8(VGA_START + 68 + 58*WC, '0');

  // draw map:
  for(x=0; x<W_GAME; x++)
    for (y=0;y<H_GAME;y++)
      XIo_Out8(VGA_START + x + y*WC, maparray[x][y]);

  // For a check for walls to be as efficient as possible, change all wall characters to be the
same character:
  for(x=0; x<W_GAME; x++)
    for (y=0;y<H_GAME;y++)
      if (maparray[x][y] == WALL_V)
           maparray[x][y] = WALL;



  create_bitmaps();
  // not sure this is the best way of getting rid of the pacman at the top
  // left corner of the screen, but it works for now:

  place_sprite(1,1,639,479);
  place_sprite(2,1,639,479);
  place_sprite(3,1,639,479);
  place_sprite(4,1,639,479);


}
```

5.1.3 drawing.h

```
// the following code was created by Charles Finkel

#ifndef _DRAW
#define _DRAW

#define W_GAME 80 // width (of the game map)
#define H_GAME 56 // height (of the game map
#define HORIZONTAL 0
#define VERTICAL 1

char maparray[W_GAME][H_GAME];
int pills = 0;

// character maps:
#define WALL_H 1 //wall horizontal
#define WALL_V 2 // wall vertical
#define WALL 1   // default wall
#define SPACE 0    // blank space (pacman trail)
#define PILL 3

// sprite 0 is allocated  to only pacman
#define PACMAN_SPRITE 0

// npic location of pacman faced right, left, up, and down, and ghosts
#define PACMAN_OPEN_RIGHT 0
#define PACMAN_OPEN_LEFT 1
#define PACMAN_OPEN_UP 2
#define PACMAN_OPEN_DOWN 3
#define PACMAN_CLOSED_RIGHT 4
#define PACMAN_CLOSED_LEFT 5
#define PACMAN_CLOSED_UP 6
#define PACMAN_CLOSED_DOWN 7
#define GHOST_1_POS_1 8
#define GHOST_1_POS_2 9
#define GHOST_2_POS_1 10
#define GHOST_2_POS_2 11
#define GHOST_3_POS_1 12
#define GHOST_3_POS_2 13


int insert_wall(int orientation, int length, int start_x, int start_y);
void fill_with_pills();
void setup_screen();
```

```
void erase_pacman_trail(int x, int y);
void place_sprite(int sn, int npic, int x, int y);
void draw_ghost(int ghost_num, int x, int y, int pos);

const coord ready_pos = {37,58};


// the following code was created by Dagna Harasim

const unsigned int pacman_open_right_array [32] = {
  0x00000044,0x44000000,
  0x00004444,0x44440000,
  0x00044444,0x44444000,
  0x00444444,0x00444400,
  0x04444444,0x00444440,
  0x44444444,0x44444400,
  0x44444444,0x44444000,
  0x44444444,0x44440000,
  0x44444444,0x44400000,
  0x44444444,0x44440000,
  0x44444444,0x44444000,
  0x04444444,0x44444400,
  0x00444444,0x44444440,
  0x00044444,0x44444400,
  0x00004444,0x44444000,
  0x00000044,0x44400000
};

const unsigned int pacman_open_left_array [32] = {
  0x00000044,0x44000000,
  0x00004444,0x44440000,
  0x00044444,0x44444000,
  0x00444400,0x44444400,
  0x04444400,0x44444440,
  0x00444444,0x44444444,
  0x00044444,0x44444444,
  0x00004444,0x44444444,
  0x00000444,0x44444444,
  0x00004444,0x44444444,
  0x00044444,0x44444444,
  0x00444444,0x44444440,
  0x04444444,0x44444400,
  0x00444444,0x44444000,
  0x00044444,0x44400000,
  0x00000444,0x44000000
};
```

```
const unsigned int pacman_open_up_array [32] = {
  0x00000000,0x00000000,
  0x00004000,0x00004000,
  0x00044400,0x00044400,
  0x00444440,0x00444440,
  0x04444444,0x04444440,
  0x04444444,0x44444444,
  0x44400444,0x44444444,
  0x44400444,0x44444444,
  0x44444444,0x44444444,
  0x44444444,0x44444444,
  0x04444444,0x44444440,
  0x04444444,0x44444440,
  0x00444444,0x44444400,
  0x00044444,0x44444000,
  0x00004444,0x44440000,
  0x00000444,0x44400000
};
const unsigned int pacman_open_down_array [32] = {
  0x00000444,0x44400000,
  0x00004444,0x44440000,
  0x00044444,0x44444000,
  0x00444444,0x44444400,
  0x04444444,0x44444440,
  0x04444444,0x44444440,
  0x44444444,0x44444444,
  0x44444444,0x44444444,
  0x44444444,0x44400444,
  0x44444444,0x44400444,
  0x44444444,0x44444440,
  0x04444440,0x44444440,
  0x04444400,0x04444400,
  0x00444000,0x00444000,
  0x00040000,0x00040000,
  0x00000000,0x00000000
};
const unsigned int pacman_closed_right_array [32] = {

  0x00000044,0x44000000,
  0x00004444,0x44440000,
  0x00044444,0x44444000,
  0x00444444,0x00444400,
  0x04444444,0x00444440,
  0x44444444,0x44444444,
  0x44444444,0x44444444,
  0x44444444,0x44444444,
```

```
  0x44444444,0x44400000,
  0x44444444,0x44444444,
  0x44444444,0x44444444,
  0x04444444,0x44444440,
  0x00444444,0x44444400,
  0x00044444,0x44444000,
  0x00004444,0x44440000,
  0x00000044,0x44000000
 };
 const unsigned int pacman_closed_left_array [32] = {
  0x00000044,0x44000000,
  0x00004444,0x44440000,
  0x00044444,0x44444000,
  0x00444400,0x44444400,
  0x04444400,0x44444440,
  0x44444444,0x44444444,
  0x44444444,0x44444444,
  0x44444444,0x44444444,
  0x00000444,0x44444444,
  0x44444444,0x44444444,
  0x44444444,0x44444444,
  0x04444444,0x44444440,
  0x00444444,0x44444400,
  0x00044444,0x44444000,
  0x00004444,0x44440000,
  0x00000044,0x44000000
 };
 const unsigned int pacman_closed_up_array [32] = {
  0x00000444,0x04400000,
  0x00004444,0x04440000,
  0x00044444,0x04444000,
  0x00444444,0x04444400,
  0x04444444,0x04444440,
  0x04444444,0x44444440,
  0x44400444,0x44444444,
  0x44400444,0x44444444,
  0x44444444,0x44444444,
  0x44444444,0x44444444,
  0x04444444,0x44444440,
  0x04444444,0x44444440,
  0x00444444,0x44444400,
  0x00044444,0x44444000,
  0x00004444,0x44440000,
  0x00000444,0x44400000

 };
```

```
const unsigned int pacman_closed_down_array [32] = {
  0x00000444,0x44400000,
  0x00004444,0x44440000,
  0x00044444,0x44444000,
  0x00444444,0x44444400,
  0x04444444,0x44444440,
  0x04444444,0x44444440,
  0x44444444,0x44444444,
  0x44444444,0x44444444,
  0x44444444,0x44400444,
  0x44444444,0x44400444,
  0x44444444,0x44444440,
  0x04444440,0x44444440,
  0x00444440,0x44444400,
  0x00044440,0x44444000,
  0x00004440,0x44440000,
  0x00000440,0x44400000
};


const unsigned int ghost_1_pos_1 [32] = {
  0x00000FFF,0xF0000000,
  0x0000FFFF,0xFF000000,
  0x000FF1FF,0x1FF00000,
  0x00FFF1FF,0x1FFF0000,
  0x00FFFFFF,0xFFFF0000,
  0x00FFF0F0,0xF0FF0000,
  0x000FFF0F,0x0FFFF000,
  0x000FFFFF,0xFFFFF000,
  0x000FFFFF,0xFFFFF000,
  0x0000FFFF,0xFFFFFF00,
  0x0000FFFF,0xFFFFFF00,
  0x0000FFFF,0xFFFFFF00,
  0x00000FFF,0xFFFFFFF0,
  0x00000FFF,0xFFFFFFF0,
  0x000000FF,0x0FFF0FFF,
  0x000000F0,0x00F000F0
};
const unsigned int ghost_1_pos_2 [32] = {
  0x00000FFF,0xF0000000,
  0x0000FFFF,0xFF000000,
  0x000FF1FF,0x1FF00000,
  0x00FFF1FF,0x1FFF0000,
  0x00FFFF00,0x0FFF0000,
  0x00FFF000,0x00FF0000,
  0x000FFF00,0x0FFFF000,
```

```
  0x000FFFFF,0xFFFFF000,
  0x000FFFFF,0xFFFFF000,
  0x0000FFFF,0xFFFFFF00,
  0x0000FFFF,0xFFFFFF00,
  0x0000FFFF,0xFFFFFF00,
  0x00000FFF,0xFFFFFFF0,
  0x00000FFF,0xFFFFFFF0,
  0x000000FF,0x0FFF0FFF,
  0x0000000F,0x0F0F000F
 };

 const unsigned int ghost_2_pos_1 [32] = {
  0x000000CC,0xC0000000,
  0x00000CCC,0xCC000000,
  0x0000CCCC,0xCCC00000,
  0x00CCCC33,0x3CCCC000,
  0x00CCC333,0x3CCC000,
  0x0CCCCC33,0x3CCCCC00,
  0x0CCCCCCC,0xCCCCCC00,
  0xCCCCCCCC,0xCCCCCCC0,
  0xCCCCCCCC,0xCCCCCCC0,
  0xCCCCCCCC,0xCCCCCCC0,
  0xCCCC0CC0,0x0CCC0CC0,
  0xCCC00CC0,0x0CCC0CC0,
  0xCCC0CCC0,0x0CC00CCC,
  0xCC00CCC0,0x0CC000CC,
  0xCC00CCC0,0x0CC000CC,
  0xC0000C00,0x00C000CC
 };
 const unsigned int ghost_2_pos_2 [32] = {
  0x000000CC,0xC0000000,
  0x00000CCC,0xCC000000,
  0x0000CCCC,0xCCC00000,
  0x00CCCC33,0x3CCCC000,
  0x00CCC333,0x33CCC000,
  0x0CCCCC33,0x3CCCCC00,
  0x0CCCCCCC,0xCCCCCC00,
  0xCCCCCCCC,0xCCCCCCC0,
  0xCCCCCCCC,0xCCCCCCC0,
  0xCCCCCCCC,0xCCCCCCC0,
  0x0CCCC0CC,0x00CCC0CC,
  0x0CCC00CC,0x00CCC0CC,
  0x0CCC0CCC,0x00CC00CC,
  0x0CC00CCC,0x00CC000C,
  0x0CC00CCC,0x00CC000C,
  0x0C0000C0,0x000C000C
```

```
};
const unsigned int ghost_3_pos_1 [32] = {
  0x000000DD,0xD0000000,
  0x00000DDD,0xDD000000,
  0x0000DD8D,0x8DD00000,
  0x0000DD8D,0x8DD00000,
  0x00000DDD,0xDD000000,
  0x000000DD,0xDDD00000,
  0x000000DD,0xDDD00000,
  0x00000DDD,0xDDD00000,
  0x00000DDD,0xDDD00000,
  0x0000DDDD,0xDDD00000,
  0x0000DDDD,0xDDDD0000,
  0x0000DDDD,0xDDDD0000,
  0x000DDDDD,0xDDDDD000,
  0x0000DDDD,0xDDDDDD00,
  0x000000DD,0xDDDDDDD0,
  0x00000000,0x0DDDDDD0,
};
const unsigned int ghost_3_pos_2 [32] = {
  0x000000DD,0xD0000000,
  0x00000DDD,0xDD000000,
  0x0000DD8D,0x8DD00000,
  0x0000DD8D,0x8DD00000,
  0x00000DDD,0xDD000000,
  0x000000DD,0xDDD00000,
  0x000000DD,0xDDD00000,
  0x00000DDD,0xDDD00000,
  0x00000DDD,0xDDD00000,
  0x0000DDDD,0xDDD00000,
  0x0000DDDD,0xDDDD0000,
  0x0000DDDD,0xDDDD0000,
  0x000DDDDD,0xDDDDD000,
  0x000DDDDD,0xDDDDDD00,
  0x0000DDDD,0x0000DDD0,
  0x0000DDD0,0x00000DD0,

};

#endif
```

5.1.4 game_engine.c

```
// code created by Charles Finkel

#include "game_engine.h"
//#include "drawing.h"

int keys = 0,open = 0;;

int score,time_left,pacman_eight_count, ghost_eight_count;
extern char buf;

/**
  Sets Pac Man's position.  Assumes position is correct (pac-man is not
  out-of-bounds
*/
void set_pacman_position(int x, int y) {
 pacman_stats.pos.x = x;
 pacman_stats.pos.y = y;
}

/** Get number of pills at position x and y,
   2 X 2 character square (this is on the 80X60 scale) */
int get_number_of_pills(int x, int y) {
 return ((maparray[x][y] == PILL) +
         (maparray[x+1][y] == PILL) +
         (maparray[x][y+1] == PILL) +
         (maparray[x+1][y+1] == PILL));
}

/**   Places pac-man on screen (most top-left area available)
     Returns 1 if successful, else 0
 */
int find_room_for_pacman() {
 int x, y;
 for(x=1; x<W_GAME-2; x++)
   for (y=1;y<H_GAME-2;y++) {
     if ((maparray[x][y] != WALL) && (maparray[x+1][y] != WALL) &&
          (maparray[x][y+1] != WALL) && (maparray[x+1][y+1] != WALL)) {
         draw_pacman(x*8,y*8,RIGHT,OPEN);
         put_pacman(x,y,RIGHT);
         pacman_stats.direction = RIGHT;
         return 1;
     }
   }
```

```
  return 0;
}

void set_pacman_orientation(int orientation) {
  pacman_stats.direction = orientation;
}

/**
   Draws pacman, and updates pac-man's coordinates
   Returns 0 if pacman can move successfully (no ghost), else return 1
*/
void put_pacman(int x,int y,int orientation) {
  set_pacman_orientation(orientation);
  //  pacman_stats.direction = orientation;
  // draw_pacman(x*8,y*8,orientation);
  set_pacman_position(x,y);
}

/**
   Increases score by amount specified in 'amount'.  Assumes amount is a positive integer
*/
void increase_score(int amount) {
  score += amount;
  display_score();
}

/**
   Refreshes score - incomplete
*/
void display_score() {
  int s2 = score/10;
  XIo_Out8(VGA_START + 12 + 58*WC, score%10 + 48);
  if (s2 != 0) {
    int s3 = s2/10;
    XIo_Out8(VGA_START + 11 + 58*WC, s2%10 + 48);
    if (s3 != 0) {
      int s4 = s3/10;
      XIo_Out8(VGA_START + 10 + 58*WC, (s3%10 + 48));
      if (s4 != 0) {
           XIo_Out8(VGA_START + 10 + 58*WC, (s4%10 + 48));
      }
    }
  }
}
```

```
void ready_set_go() {
  int j,a;
  for (j = 0; j < 30000000; j++) {a=j-34;}
  XIo_Out8(VGA_START + 37 + 58*WC, '2');
  for (j = 0; j < 30000000; j++) {a=j-34;}
  XIo_Out8(VGA_START + 37 + 58*WC, '1');
  for (j = 0; j < 30000000; j++) {a=j-34;}
  XIo_Out8(VGA_START + 37 + 58*WC, 'G');
  XIo_Out8(VGA_START + 38 + 58*WC, 'O');


}

/*
  //test:
void test()
{
  int i,a,j;

  while(1) {
    for (i = 0; i < 10; i ++) {
      for (j = 0; j < 500000; j++) {a=i-34;}
      move_pacman(RIGHT);
    }
    for (i = 0; i < 10; i ++) {
      for (j = 0; j < 500000; j++) {a=i-34;}
      move_pacman(DOWN);
    }
    for (i = 0; i < 10; i ++) {
      for (j = 0; j < 500000; j++) {a=i-34;}
      move_pacman(LEFT);
    }
    for (i = 0; i < 10; i ++) {
      for (j = 0; j < 500000; j++) {a=i-34;}
      move_pacman(UP);
    }
  }
}
*/

/* This function should only be called when the value of eight count is not 0.
   If it is we have a problem.
   This funtion does not need to test if there is a wall (cuz there won't be)
   position of pacman at this point should be set to _______*/
void move_pacman(int direction, int offset) {
  int x = pacman_stats.pos.x;
```

```
  int y = pacman_stats.pos.y;
  switch (direction) {
  case RIGHT:
    draw_pacman(x*8-8+offset,y*8,direction,open);
     break;
  case LEFT:
    draw_pacman(x*8+8-offset,y*8,direction,open);
    break;
  case UP:
    draw_pacman(x*8,y*8+8-offset,direction,open);
    break;
  case DOWN:
    draw_pacman(x*8,y*8-8+offset,direction,open);
    break;
  default:
    break;
  }

}

/* This function should only be called when the value of eight count is 0.
   If it's not we have a problem.
   This function will not necessarily 'turn' pacman, but if the user requests a turn
   he will.  Else, he'll just stay moving forward.
   But this function has to check if a wall is present.  If it is, don't move pacman,
   and keep eight count at zero.
   Returns 1 if pacman is facing a wall, else 0.
   Also, eats a pill if it's in the previous spot.
   position of pacman at this point should be set to _______*/
int turn_pacman(int direction) {
  int x = pacman_stats.pos.x;
  int y = pacman_stats.pos.y;
  open = !open;
  //  print("turn_pacman function called\n\r");
  // increase score by amount of pills pacman is on top of:
  increase_score(get_number_of_pills(x,y));
  // remove pills from that area:
  erase_pacman_trail(x,y);

  // check if there is a wall where pacman wants to move:
  switch (direction) {
  case RIGHT:
    if ((maparray[x + 2][y] == WALL) ||
          (maparray[x + 2][y + 1] == WALL))
      break;
    else {
```

```
draw_pacman(x*8,y*8,direction,OPEN);
    put_pacman(x+1,y,RIGHT);
    return 0;
  }
  break;
case LEFT:
  if ((maparray[x - 1][y] == WALL) ||
        (maparray[x - 1][y + 1] == WALL))
    break;
  else {
draw_pacman(x*8,y*8,direction,OPEN);
    put_pacman(x-1,y,LEFT);
    return 0;
  }
  break;
case UP:
  if ((maparray[x][y - 1] == WALL) ||
        (maparray[x + 1][y - 1] == WALL))
    break;
  else {
draw_pacman(x*8,y*8,direction,OPEN);
    put_pacman(x,y-1,UP);
    return 0;
  }
  break;
case DOWN:
  if ((maparray[x][y + 2] == WALL) ||
        (maparray[x + 1][y + 2] == WALL))
    break;
  else {
draw_pacman(x*8,y*8,direction,OPEN);
    put_pacman(x,y+1,DOWN);
    return 0;
  }
  break;
}
// pacman trying to move to a wall:
if (pacman_stats.direction != direction) {
  switch (pacman_stats.direction) {
  case RIGHT:
  if ((maparray[x + 2][y] == WALL) ||
        (maparray[x + 2][y + 1] == WALL))
    return 1;
  else {
draw_pacman(x*8,y*8,RIGHT,OPEN);
    put_pacman(x+1,y,RIGHT);
```

```
      return 0;
    }
      break;
    case LEFT:
    if ((maparray[x - 1][y] == WALL) ||
          (maparray[x - 1][y + 1] == WALL))
      break;
    else {
  draw_pacman(x*8,y*8,LEFT,OPEN);
      put_pacman(x-1,y,LEFT);
      return 0;
    }
    break;
    case UP:
    if ((maparray[x][y - 1] == WALL) ||
          (maparray[x + 1][y - 1] == WALL))
      break;
    else {
  draw_pacman(x*8,y*8,UP,OPEN);
      put_pacman(x,y-1,UP);
      return 0;
    }
      break;
    case DOWN:
    if ((maparray[x][y + 2] == WALL) ||
          (maparray[x + 1][y + 2] == WALL))
      break;
    else {
  draw_pacman(x*8,y*8,DOWN,OPEN);
      put_pacman(x,y+1,DOWN);
      return 0;
    }
      break;
    }
  }
  else {
  draw_pacman(x*8,y*8,pacman_stats.direction,OPEN);
    return 1;
  }

}

void move_ghosts(int offset) {
  int i;
  for (i = 0; i < 3; i ++) {
    //  for (i = 0; i < 3; i ++) {
```

```
    int x = ghost_stats[i].g_coord.x;
    int y = ghost_stats[i].g_coord.y;
    int direction = ghost_stats[i].direction;
    int open = (offset%8) < 4;
    switch (direction) {
    case RIGHT:
      draw_ghost(i,x*8-8+offset,y*8,open);
      break;
    case LEFT:
      draw_ghost(i,x*8+8-offset,y*8,open);
      break;
    case UP:
      draw_ghost(i,x*8,y*8+8-offset,open);
      break;
    case DOWN:
      draw_ghost(i,x*8,y*8-8+offset,open);
      break;
    default:

    }
  }
  /*
  int x = pacman_stats.pos.x;
  int y = pacman_stats.pos.y;
  char open = (offset%8) < 4;
  switch (direction) {
  case RIGHT:
    draw_pacman(x*8-8+offset,y*8,direction,open);
     break;
  case LEFT:
    draw_pacman(x*8+8-offset,y*8,direction,open);
    break;
  case UP:
    draw_pacman(x*8,y*8+8-offset,direction,open);
    break;
  case DOWN:
    draw_pacman(x*8,y*8-8+offset,direction,open);
    break;
  default:
  */
}

int put_ghost(int ghost_num, int x, int y, int dir) {
  int gx,gy,px,py,i,j,k,l;
  ghost_stats[ghost_num].g_coord.x = x;
  ghost_stats[ghost_num].g_coord.y = y;
```

```
  ghost_stats[ghost_num].direction = dir;
  gx = x;
  gy = y;
  px = pacman_stats.pos.x;
  py = pacman_stats.pos.y;
  for (i = gx; i < gx+2; i ++)
    for (j = gy; j < gy + 2; j ++)
      for (k = px; k < px + 2; k ++)
          for (l = py; l < py + 2; l ++) {
            if ((i == k) && (j == l)) {
              print("Pacman got eaten\n\r");
              return 1;
            }
          }
  return 0;
}

/**
Returns 1 if pacman got eaten, else 0
**/
int turn_ghosts() {
  int moved = 0;
  int pos;
  int i,x,y;
  int killed=0;;
  //  for (i = 0; i < 3; i ++) {
  for (i = 0; i < 3; i ++) {
    moved = 0;
    x = ghost_stats[i].g_coord.x;
    y = ghost_stats[i].g_coord.y;
    draw_ghost(i,x*8,y*8,0);
    pos = rand() % 4;
    while (!moved) {
      switch (pos) {
      case RIGHT:
          if (!((maparray[x + 2][y] == WALL) ||
               (maparray[x + 2][y + 1] == WALL))) {
            if (put_ghost(i,x+1,y,RIGHT)) {killed = 1; }
            moved = 1;
          }
          else {
            pos = LEFT;
          }
          break;
      case LEFT:
          if (!((maparray[x - 1][y] == WALL) ||
```

```
            (maparray[x - 1][y + 1] == WALL))) {
          if (put_ghost(i,x-1,y,LEFT)) {killed = 1; }
          moved = 1;
         }
         else {
          pos = UP;
         }
         break;
    case UP:
         if (!((maparray[x][y - 1] == WALL) ||
            (maparray[x + 1][y - 1] == WALL))) {
          if (put_ghost(i,x,y-1,UP)) {killed = 1;}
          moved = 1;
         }
         else {
          pos = DOWN;
         }
         break;
    case DOWN:
         if (!((maparray[x][y + 2] == WALL) ||
            (maparray[x + 1][y + 2] == WALL)))
          {
           if (put_ghost(i,x,y+1,DOWN)) {killed = 1;}
          moved = 1;
         }
         else {
          pos = RIGHT;
         }
         break;
    default:
         print("You shouldn't be here\n\r");
         break;
    }
   }
  }
  return (killed);
}

int lose_life() {
  lives--;
  XIo_Out8(VGA_START + 49 + 58*WC, 48 + lives);
  return lives;
}

void game_loop() {
  int j,a,ghost_go = 0;
```

```
  char pacman_is_facing_a_wall = 0;
  pacman_eight_count = 0;
  ghost_eight_count = 0;
  print("Game running\n\r");
  while(1) {
    keys += buf;
    for (j = 0; j < 200000; j++) {a=j-34;}
    if ((pacman_eight_count % 8) == 0) {
      pacman_eight_count = 0;
      /*
      print("character: ");
      putnum(buf);
      print("\n\r");
      */
      switch (buf) {
      case LEFT_ARROW : pacman_is_facing_a_wall = turn_pacman(LEFT); break;
      case RIGHT_ARROW : pacman_is_facing_a_wall = turn_pacman(RIGHT); break;
      case DOWN_ARROW : pacman_is_facing_a_wall = turn_pacman(DOWN); break;
      case UP_ARROW : pacman_is_facing_a_wall = turn_pacman(UP); break;
      default :
          pacman_is_facing_a_wall = 1;
//print("You shouldn't be here\n\r");
          break;
      }
    }
    else {
      move_pacman(pacman_stats.direction,pacman_eight_count);
    }
    // if pacman is facing a wall, we don't want to increase the eight count,
    // so that we can continue to call the turn_pacman function instead of
    // move_pacman

    if (!pacman_is_facing_a_wall) pacman_eight_count ++;

    ghost_go = !(ghost_go);
    if (ghost_go) {
      if ((ghost_eight_count % 8) == 0) {
          if (score == pills) // game is over
            break;
          ghost_eight_count = 0;
          // if pacman, gets killed, replace him at the beginning.
          if (turn_ghosts()) {
            pacman_eight_count = 0;
            if (lose_life() == 0) {
              break;
            }
```

```
            find_room_for_pacman();
          }
      }
      else {
          move_ghosts(ghost_eight_count);
      }

      ghost_eight_count++;
    }
  }
  print("Game over");

  XIo_Out8(VGA_START + 31 + 58*WC, 'G');
  XIo_Out8(VGA_START + 32 + 58*WC, 'A');
  XIo_Out8(VGA_START + 33 + 58*WC, 'M');
  XIo_Out8(VGA_START + 34 + 58*WC, 'E');
  XIo_Out8(VGA_START + 35 + 58*WC, ' ');
  XIo_Out8(VGA_START + 36 + 58*WC, 'O');
  XIo_Out8(VGA_START + 37 + 58*WC, 'V');
  XIo_Out8(VGA_START + 38 + 58*WC, 'E');
  XIo_Out8(VGA_START + 39 + 58*WC, 'R');

}

void place_ghosts_randomly() {
  int i=0;
  int placed=0;
  int x,y;

  // not random at all:
          put_ghost(0,36,42,RIGHT);
          draw_ghost(0,56*8,42*8,0);
          put_ghost(1,5,47,RIGHT);
          draw_ghost(1,45*8,47*8,0);
          put_ghost(2,42,23,0);
          draw_ghost(2,42*8,23*8,0);

          /*
  for (i=0;i<3;i++) {
    placed = 0;
    while(!placed) {
      x = (rand() % (W_GAME-3)) + 1;
      y = (rand() % (W_GAME-3)) + 1;
      if ((maparray[x][y] != WALL) && (maparray[x+1][y] != WALL) &&
            (maparray[x][y+1] != WALL) && (maparray[x+1][y+1] != WALL)) {
          put_ghost(i,x,y);
```

```
            draw_ghost(i,x*8,y*8,0);
            placed = 1;
        }
      }
    }
            */


}

int get_keys() {
  return keys;
}

/**
   Starts the game
*/
void start_game() {

  int j = 0,a;
  // set score to 0:
  score = 0;
  time_left = 0;
  //  while (buf == 0) {     for (j = 0; j < 200000; j++) {a=j-34;}}
  //  srand( (unsigned int) time( NULL ));
  srand(get_keys());

  if(find_room_for_pacman() == 0) {
    print("Game will not run! (No room for pac-man)\n\r");
    return;
  }

  place_ghosts_randomly();

  ready_set_go();

  //  test();
  game_loop();
}
```

Game_engine.h

```
// code created by Charles Finkel

#ifndef _GAME
#define _GAME
```

```
void start_game();
int place_pacman();
//int move_pacman(int direction);
int move_pacman_right();
int move_pacman_left();
int move_pacman_up();
int move_pacman_down();
void set_pacman_position(int x, int y);
void increase_score(int amount);
void display_score();
void put_pacman(int x, int y, int orientation);

struct pacman_stats{
  coord pos;
  int direction;
} pacman_stats;

int lives = 3;

struct ghost_stats{
  coord g_coord;
  int direction;
} ghost_stats[3];

/*
  ASCII values of right, left, up and down arrows
 */
#define RIGHT_ARROW 0x43
#define LEFT_ARROW 0x44
#define UP_ARROW 0x41
#define DOWN_ARROW 0x42

#define OPEN 1
#define CLOSED 0

#endif
```

5.1.5 isr.c

```
#include "xbasic_types.h"
#include "xio.h"
#include "xparameters.h"
#include "xuartlite_l.h"

#define BUF_SIZE 256

char buf;

/*
 * Interrupt service routine for the UART
 */
void uart_handler(void *callback)
{
  Xuint32 IsrStatus;

  Xuint8 incoming_character;

  /* Check the ISR status register so we can identify the interrupt source */
  IsrStatus = XIo_In32(XPAR_MYUART_BASEADDR +
XUL_STATUS_REG_OFFSET);

  if ((IsrStatus & (XUL_SR_RX_FIFO_FULL | XUL_SR_RX_FIFO_VALID_DATA))
!= 0) {
    /* The input FIFO contains data: read it */
    incoming_character =
      (Xuint8) XIo_In32( XPAR_MYUART_BASEADDR + XUL_RX_FIFO_OFFSET );

    buf = incoming_character;
  }

  if ((IsrStatus & XUL_SR_TX_FIFO_EMPTY) != 0) {
    /* The output FIFO is empty: we can send another character */
  }

}
```

5.1.6 main.c

```
/*Modified by Dagna Harasim, David Soofian, Charles Finkel*/

#include "main.h"
#include "drawing.c"
#include "game_engine.c"

#include "xbasic_types.h"
#include "xio.h"

extern void uart_handler(void *callback);
extern int uart_interrupt_count;
extern char buf;

void setup_interrupts()
{
  /*
   * Reset the interrupt controller peripheral
   */

  /* Disable the interrupt signal */
  XIntc_mMasterDisable(XPAR_INTC_SINGLE_BASEADDR);

  /* Disable all interrupt sources */
  XIntc_mEnableIntr(XPAR_INTC_SINGLE_BASEADDR,0);

  /* Acknowledge all possible interrupt sources
     to make sure none are pending */
  XIntc_mAckIntr(XPAR_INTC_SINGLE_BASEADDR, 0xffffffff);

  /*
   * Install the UART interrupt handler
   */

  XIntc_InterruptVectorTable[XPAR_INTC_MYUART_INTERRUPT_INTR].Handler =
    uart_handler;

  /*
   * Enable interrupt sources
   */

  /* Enable CPU interrupts */
  microblaze_enable_interrupts();
```

```
  /* Enable interrupts from the interrupt controller */
  XIntc_mMasterEnable(XPAR_INTC_SINGLE_BASEADDR);

  /* Tell the interrupt controller to accept interrupts from the UART */
  XIntc_mEnableIntr(XPAR_INTC_SINGLE_BASEADDR,
XPAR_MYUART_INTERRUPT_MASK);

  /* Enable UART interrupt generation */
  XUartLite_mEnableIntr(XPAR_MYUART_BASEADDR);
}

// code created by Charles Finkel

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

int main()
{
  int x,y;

  // Enable the instruction cache: makes the code run 6 times faster
  microblaze_enable_icache();

  // erasing screen
  for(x=0; x<H*W; x+=4)
    XIo_Out32(VGA_START + x, 0);

  setup_interrupts();
  setup_screen();
  start_game();

  return 0;
}
```

5.1.7 main.h

```
#ifndef _MAIN
#define _MAIN

#include "xbasic_types.h"
#include "xio.h"

#include "xintc_l.h"
#include "xuartlite_l.h"
//#include "pthread.h"


//int pthread_create(pthread_t *tid, const pthread_attr_t *tattr,
//void*(*start_routine)(void *), void *arg);

//void start_routine

#define W 640
#define H 480
#define WC 80 // width (in characters)
#define HC 60 // height (in characters)
#define VGA_START 0x00800000
#define SPRITE_XY 0xFEFD0000

#define LEFT 0
#define RIGHT 1
#define UP 2
#define DOWN 3

typedef struct{
  int x;
  int y;
}coord;

#endif
```

5.2 Hardware

5.2.1 display_prio.vhd

```
--------------------------------------------------------------------------
--
--Color Decider and Prioritizer
--
--This component is part of the VGA module.  It decodes the color of
--each pixel and based on predefined priority sends the new pixel
--information to vga.vhd to be displayed on the screen.
--
--Created by Dagna Harasim
--
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity display_prio is
  port(
    vgachar     : in std_logic_vector(3 downto 0);
    sprite0     : in std_logic_vector(3 downto 0);
    sprite1     : in std_logic_vector(3 downto 0);
    sprite2     : in std_logic_vector(3 downto 0);
    sprite3     : in std_logic_vector(3 downto 0);

    r           : out std_logic_vector(7 downto 0);
    g           : out std_logic_vector(7 downto 0);
    b           : out std_logic_vector(7 downto 0)

  );
end display_prio;

architecture xxx of display_prio is

signal data4    : std_logic_vector(3 downto 0);
signal rgb      : std_logic_vector(23 downto 0);

begin
  --output sprite0 first
  data4 <= sprite0 when not (sprite0 = X"0") else
           sprite1 when not (sprite1 = X"0") else
           sprite2 when not (sprite2 = X"0") else
           sprite3 when not (sprite3 = X"0") else
           vgachar;

--pixel color
  rgb <= X"000000" when data4 = X"0" else  -- transp/black
         X"FF0000" when data4 = X"1" else  -- red
         X"00FF00" when data4 = X"2" else  -- green
         X"0000FF" when data4 = X"3" else  -- blue
```

```
          X"FFFF00" when data4 = X"4" else  -- yellow
          X"DBD704" when data4 = X"5" else  -- yellow darker
          X"F7FC51" when data4 = X"6" else  -- yellow lighter
          X"FFFDB5" when data4 = X"7" else  -- yellow lighter more

          X"FFA500" when data4 = X"8" else  -- orange
          X"00F6FF" when data4 = X"9" else  -- light blue
          X"E8D122" when data4 = X"A" else  -- dark yellow
          X"FCF5A9" when data4 = X"B" else  -- light yellow

          X"f79204" when data4 = X"C" else  -- orange
          X"48007C" when data4 = X"D" else  -- purple
          X"20752D" when data4 = X"E" else  -- dark green
          X"FFFFFF" when data4 = X"F";      -- white

  r <= rgb(23 downto 16);
  g <= rgb(15 downto 8);
  b <= rgb(7 downto 0);

end xxx;
```

5.2.2 font.vhd

```
--------------------------------------------------------------------------------
-- Character ROM that stores all all 128 ACSII characters
-- Written by Dagna Harasim, David Soofian and Charles Finkel
--------------------------------------------------------------------------------

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

entity rom_1024_8 is
  port (
    Clk  : in std_logic;
    en   : in std_logic; -- Read enable
    addr : in std_logic_vector(9 downto 0);
    data : out std_logic_vector(7 downto 0)
 );
end rom_1024_8;

architecture imp of rom_1024_8 is
  type rom_type is array (0 to 1023) of std_logic_vector(7 downto 0);
  constant ROM : rom_type :=
    (
      X"00", X"00", X"00", X"00", X"00", X"00", X"00", X"00",
      X"ff", X"ff", X"ff", X"ff", X"ff", X"ff", X"ff", X"ff",
      X"ff", X"ff", X"ff", X"ff", X"ff", X"ff", X"ff", X"ff",
      X"00", X"00", X"00", X"18", X"18", X"00", X"00", X"00",
      X"10", X"38", X"7c", X"fe", X"7c", X"38", X"10", X"00",
      X"38", X"7c", X"38", X"fe", X"fe", X"d6", X"10", X"38",
      X"10", X"38", X"7c", X"fe", X"fe", X"7c", X"10", X"38",
      X"00", X"00", X"18", X"3c", X"3c", X"18", X"00", X"00",
      X"ff", X"ff", X"e7", X"c3", X"c3", X"e7", X"ff", X"ff",
      X"00", X"3c", X"66", X"42", X"42", X"66", X"3c", X"00",
      X"ff", X"c3", X"99", X"bd", X"bd", X"99", X"c3", X"ff",
      X"0f", X"07", X"0f", X"7d", X"cc", X"cc", X"cc", X"78",
      X"3c", X"66", X"66", X"66", X"3c", X"18", X"7e", X"18",
      X"3f", X"33", X"3f", X"30", X"30", X"70", X"f0", X"e0",
      X"7f", X"63", X"7f", X"63", X"63", X"67", X"e6", X"c0",
      X"18", X"db", X"3c", X"e7", X"e7", X"3c", X"db", X"18",
      X"80", X"e0", X"f8", X"fe", X"f8", X"e0", X"80", X"00",
      X"02", X"0e", X"3e", X"fe", X"3e", X"0e", X"02", X"00",
      X"18", X"3c", X"7e", X"18", X"18", X"7e", X"3c", X"18",
      X"66", X"66", X"66", X"66", X"66", X"00", X"66", X"00",
      X"7f", X"db", X"db", X"7b", X"1b", X"1b", X"1b", X"00",
      X"3e", X"61", X"3c", X"66", X"66", X"3c", X"86", X"7c",
      X"00", X"00", X"00", X"00", X"7e", X"7e", X"7e", X"00",
      X"18", X"3c", X"7e", X"18", X"7e", X"3c", X"18", X"ff",
      X"18", X"3c", X"7e", X"18", X"18", X"18", X"18", X"00",
      X"18", X"18", X"18", X"18", X"7e", X"3c", X"18", X"00",
      X"00", X"18", X"0c", X"fe", X"0c", X"18", X"00", X"00",
      X"00", X"30", X"60", X"fe", X"60", X"30", X"00", X"00",
      X"00", X"00", X"c0", X"c0", X"c0", X"fe", X"00", X"00",
      X"00", X"24", X"66", X"ff", X"66", X"24", X"00", X"00",
      X"00", X"18", X"3c", X"7e", X"ff", X"ff", X"00", X"00",
      X"00", X"ff", X"ff", X"7e", X"3c", X"18", X"00", X"00",
```

```
X"00", X"00", X"00", X"00", X"00", X"00", X"00", X"00",
X"18", X"3c", X"3c", X"18", X"18", X"00", X"18", X"00",
X"66", X"66", X"24", X"00", X"00", X"00", X"00", X"00",
X"6c", X"6c", X"fe", X"6c", X"fe", X"6c", X"6c", X"00",
X"18", X"3e", X"60", X"3c", X"06", X"7c", X"18", X"00",
X"00", X"c6", X"cc", X"18", X"30", X"66", X"c6", X"00",
X"38", X"6c", X"38", X"76", X"dc", X"cc", X"76", X"00",
X"18", X"18", X"30", X"00", X"00", X"00", X"00", X"00",
X"0c", X"18", X"30", X"30", X"30", X"18", X"0c", X"00",
X"30", X"18", X"0c", X"0c", X"0c", X"18", X"30", X"00",
X"00", X"66", X"3c", X"ff", X"3c", X"66", X"00", X"00",
X"00", X"18", X"18", X"7e", X"18", X"18", X"00", X"00",
X"00", X"00", X"00", X"00", X"00", X"18", X"18", X"30",
X"00", X"00", X"00", X"7e", X"00", X"00", X"00", X"00",
X"00", X"00", X"00", X"00", X"00", X"18", X"18", X"00",
X"06", X"0c", X"18", X"30", X"60", X"c0", X"80", X"00",
X"38", X"6c", X"c6", X"d6", X"c6", X"6c", X"38", X"00",
X"18", X"38", X"18", X"18", X"18", X"18", X"7e", X"00",
X"7c", X"c6", X"06", X"1c", X"30", X"66", X"fe", X"00",
X"7c", X"c6", X"06", X"3c", X"06", X"c6", X"7c", X"00",
X"1c", X"3c", X"6c", X"cc", X"fe", X"0c", X"1e", X"00",
X"fe", X"c0", X"c0", X"fc", X"06", X"c6", X"7c", X"00",
X"38", X"60", X"c0", X"fc", X"c6", X"c6", X"7c", X"00",
X"fe", X"c6", X"0c", X"18", X"30", X"30", X"30", X"00",
X"7c", X"c6", X"c6", X"7c", X"c6", X"c6", X"7c", X"00",
X"7c", X"c6", X"c6", X"7e", X"06", X"0c", X"78", X"00",
X"00", X"18", X"18", X"00", X"00", X"18", X"18", X"00",
X"00", X"18", X"18", X"00", X"00", X"18", X"18", X"30",
X"06", X"0c", X"18", X"30", X"18", X"0c", X"06", X"00",
X"00", X"00", X"7e", X"00", X"00", X"7e", X"00", X"00",
X"60", X"30", X"18", X"0c", X"18", X"30", X"60", X"00",
X"7c", X"c6", X"0c", X"18", X"18", X"00", X"18", X"00",
X"7c", X"c6", X"de", X"de", X"de", X"c0", X"78", X"00",
X"38", X"6c", X"c6", X"fe", X"c6", X"c6", X"c6", X"00",
X"fc", X"66", X"66", X"7c", X"66", X"66", X"fc", X"00",
X"3c", X"66", X"c0", X"c0", X"c0", X"66", X"3c", X"00",
X"f8", X"6c", X"66", X"66", X"66", X"6c", X"f8", X"00",
X"fe", X"62", X"68", X"78", X"68", X"62", X"fe", X"00",
X"fe", X"62", X"68", X"78", X"68", X"60", X"f0", X"00",
X"3c", X"66", X"c0", X"c0", X"ce", X"66", X"3a", X"00",
X"c6", X"c6", X"c6", X"fe", X"c6", X"c6", X"c6", X"00",
X"3c", X"18", X"18", X"18", X"18", X"18", X"3c", X"00",
X"1e", X"0c", X"0c", X"0c", X"cc", X"cc", X"78", X"00",
X"e6", X"66", X"6c", X"78", X"6c", X"66", X"e6", X"00",
X"f0", X"60", X"60", X"60", X"62", X"66", X"fe", X"00",
X"c6", X"ee", X"fe", X"fe", X"d6", X"c6", X"c6", X"00",
X"c6", X"e6", X"f6", X"de", X"ce", X"c6", X"c6", X"00",
X"7c", X"c6", X"c6", X"c6", X"c6", X"c6", X"7c", X"00",
X"fc", X"66", X"66", X"7c", X"60", X"60", X"f0", X"00",
X"7c", X"c6", X"c6", X"c6", X"c6", X"ce", X"7c", X"0e",
X"fc", X"66", X"66", X"7c", X"6c", X"66", X"e6", X"00",
X"3c", X"66", X"30", X"18", X"0c", X"66", X"3c", X"00",
X"7e", X"7e", X"5a", X"18", X"18", X"18", X"3c", X"00",
X"c6", X"c6", X"c6", X"c6", X"c6", X"c6", X"7c", X"00",
X"c6", X"c6", X"c6", X"c6", X"c6", X"6c", X"38", X"00",
X"c6", X"c6", X"c6", X"d6", X"d6", X"fe", X"6c", X"00",
X"c6", X"c6", X"6c", X"38", X"6c", X"c6", X"c6", X"00",
```

```
      X"66", X"66", X"66", X"3c", X"18", X"18", X"3c", X"00",
      X"fe", X"c6", X"8c", X"18", X"32", X"66", X"fe", X"00",
      X"3c", X"30", X"30", X"30", X"30", X"30", X"3c", X"00",
      X"c0", X"60", X"30", X"18", X"0c", X"06", X"02", X"00",
      X"3c", X"0c", X"0c", X"0c", X"0c", X"0c", X"3c", X"00",
      X"10", X"38", X"6c", X"c6", X"00", X"00", X"00", X"00",
      X"00", X"00", X"00", X"00", X"00", X"00", X"00", X"ff",
      X"30", X"18", X"0c", X"00", X"00", X"00", X"00", X"00",
      X"00", X"00", X"78", X"0c", X"7c", X"cc", X"76", X"00",
      X"e0", X"60", X"7c", X"66", X"66", X"66", X"dc", X"00",
      X"00", X"00", X"7c", X"c6", X"c0", X"c6", X"7c", X"00",
      X"1c", X"0c", X"7c", X"cc", X"cc", X"cc", X"76", X"00",
      X"00", X"00", X"7c", X"c6", X"fe", X"c0", X"7c", X"00",
      X"3c", X"66", X"60", X"f8", X"60", X"60", X"f0", X"00",
      X"00", X"00", X"76", X"cc", X"cc", X"7c", X"0c", X"f8",
      X"e0", X"60", X"6c", X"76", X"66", X"66", X"e6", X"00",
      X"18", X"00", X"38", X"18", X"18", X"18", X"3c", X"00",
      X"06", X"00", X"06", X"06", X"06", X"66", X"66", X"3c",
      X"e0", X"60", X"66", X"6c", X"78", X"6c", X"e6", X"00",
      X"38", X"18", X"18", X"18", X"18", X"18", X"3c", X"00",
      X"00", X"00", X"ec", X"fe", X"d6", X"d6", X"d6", X"00",
      X"00", X"00", X"dc", X"66", X"66", X"66", X"66", X"00",
      X"00", X"00", X"7c", X"c6", X"c6", X"c6", X"7c", X"00",
      X"00", X"00", X"dc", X"66", X"66", X"7c", X"60", X"f0",
      X"00", X"00", X"76", X"cc", X"cc", X"7c", X"0c", X"1e",
      X"00", X"00", X"dc", X"76", X"60", X"60", X"f0", X"00",
      X"00", X"00", X"7e", X"c0", X"7c", X"06", X"fc", X"00",
      X"30", X"30", X"fc", X"30", X"30", X"36", X"1c", X"00",
      X"00", X"00", X"cc", X"cc", X"cc", X"cc", X"76", X"00",
      X"00", X"00", X"c6", X"c6", X"c6", X"6c", X"38", X"00",
      X"00", X"00", X"c6", X"d6", X"d6", X"fe", X"6c", X"00",
      X"00", X"00", X"c6", X"6c", X"38", X"6c", X"c6", X"00",
      X"00", X"00", X"c6", X"c6", X"c6", X"7e", X"06", X"fc",
      X"00", X"00", X"7e", X"4c", X"18", X"32", X"7e", X"00",
      X"0e", X"18", X"18", X"70", X"18", X"18", X"0e", X"00",
      X"18", X"18", X"18", X"18", X"18", X"18", X"18", X"00",
      X"70", X"18", X"18", X"0e", X"18", X"18", X"70", X"00",
      X"76", X"dc", X"00", X"00", X"00", X"00", X"00", X"00",
      X"00", X"10", X"38", X"6c", X"c6", X"c6", X"fe", X"00"
    );

begin
  process (Clk)
  begin
    if (Clk'event and Clk =  '1' ) then
      if (en =  '1' ) then
        data <= ROM(conv_integer(addr));
      end
      if;
    end if;
  end process;
end imp;
```

5.2.3 sprite.vhd

```
----------------------------------------------------------------------
--
--Sprite
--
--Component of the VGA module.  Reads appropriate sprite graphic from
--BRAM and sends for display at a desired location.
--
-- Created by Dagna Harasim and David Soofian
--
----------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity sprite is
  port (
    pixclk              : in std_logic;
    sprreset            : in std_logic;
    x                   : in std_logic_vector (10 downto 0);
    y                   : in std_logic_vector (9 downto 0);
    line_cnt            : in std_logic_vector (9 downto 0);
    pix_cnt             : in std_logic_vector (10 downto 0);
    sprite_data_16      : in std_logic_vector(15 downto 0);
    sprite_data_we      : in std_logic_vector(3 downto 0);
    sprite_data_addr    : out std_logic_vector(3 downto 0);
    video_data          : out std_logic_vector(3 downto 0));
end sprite;

architecture Behavioral of sprite is

signal spritesr : std_logic_vector(63 downto 0);
signal xcnt     : std_logic_vector(4 downto 0);
signal ycnt     : std_logic_vector(4 downto 0);
signal spritesh : std_logic;

signal x_start : std_logic;

begin

--check if in active region
  process(pixclk)
    begin
      if pixclk'event and pixclk='1' then

        if x = pix_cnt then
          x_start <= '1';
        else
          x_start <= '0';
        end if;

        if x_start = '1' then
          xcnt <= "00000";
```

```
        elsif xcnt(4)='0' then
          xcnt <= xcnt + 1;
        end if;

        if pix_cnt = 0 and y = line_cnt then
            ycnt <=  "00000";
        elsif pix_cnt = 641 and ycnt(4)='0' then
            ycnt <= ycnt + 1;
        end if;

      end if;
    end process;

    sprite_data_addr <= ycnt(3 downto 0);  --current vertical lien of
sprite

    spritesh <= not xcnt(4);            --set to 1 when in active
region

 --load from BRAM and shift if loading to screen
  process(pixclk)
    begin
      if pixclk'event and pixclk='1' then

        if sprite_data_we(3)='1' then
          spritesr(15 downto 0) <= sprite_data_16;
        elsif spritesh ='1' then
          spritesr(15 downto 0) <= spritesr(11 downto 0) & "0000";
        end if;

        if sprite_data_we(2)='1' then
          spritesr(31 downto 16) <= sprite_data_16;
        elsif spritesh ='1' then
          spritesr(31 downto 16) <= spritesr(27 downto 12);
        end if;

        if sprite_data_we(1)='1' then
          spritesr(47 downto 32) <= sprite_data_16;
        elsif spritesh ='1' then
          spritesr(47 downto 32) <= spritesr(43 downto 28);
        end if;

        if sprite_data_we(0)='1' then
          spritesr(63 downto 48) <= sprite_data_16;
        elsif spritesh ='1' then
          spritesr(63 downto 48) <= spritesr(59 downto 44);
        end if;

      end if;

    end process;

--send pixel out to video
video_data <= spritesr(63 downto 60) when xcnt(4) = '0' and ycnt(4) =
'0' else "0000";

  end Behavioral;
```

5.2.4 sprite_loader.vhd

```
----------------------------------------------------------------------
--
-- sprite loader
-- Loads sprites from SRAM during blanking
--
-- Created by Dagna Harasim and David Soofian
--
----------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity sprite_loader is
  port (
    pixclk              : in std_logic;
    reset               : in std_logic;
    SRAM_data_in        : in std_logic_vector (15 downto 0);
    sprite_ld           : in std_logic;
    sprite_addr_0, sprite_addr_1, sprite_addr_2, sprite_addr_3  : in
std_logic_vector (3 downto 0);
    sprite_num_0, sprite_num_1, sprite_num_2, sprite_num_3      : in
std_logic_vector (6 downto 0);
    sprite_we_0, sprite_we_1, sprite_we_2, sprite_we_3          : out
std_logic_vector (3 downto 0);
    sprite_data_16      : out std_logic_vector(15 downto 0);
    SRAM_addr           : out std_logic_vector(19 downto 0);
    SRAM_req            : out std_logic
    );
end sprite_loader;

architecture Behavioral of sprite_loader is

signal SRAM_sm, sprite_sm        : std_logic_vector(4 downto 0);
signal sprite_ld_delay           : std_logic_vector(1 downto 0);
signal current_sprite_addr       : std_logic_vector(3 downto 0);
signal current_sprite_num        : std_logic_vector(6 downto 0);

begin

 process(reset,pixclk)
  begin
    if reset = '1' then
      sprite_ld_delay <= "00";
    elsif pixclk'event and pixclk='1' then
      sprite_ld_delay <= sprite_ld & sprite_ld_delay(1);
    end if;
  end process;

  process(reset,pixclk)
  begin
```

```
    if reset = '1' then
      SRAM_sm <= "10000";
    elsif pixclk'event and pixclk='1' then
      if sprite_ld ='1' then
        SRAM_sm <= "00000";
      elsif SRAM_sm(4) = '0' then
        SRAM_sm <= SRAM_sm + 1;
      end if;
    end if;
  end process;

 process(reset,pixclk)
  begin
    if reset = '1' then
      sprite_sm <= "10000";
    elsif pixclk'event and pixclk='1' then
      if sprite_ld_delay(0) ='1' then
        sprite_sm <= "00000";
      elsif sprite_sm(4) = '0' then
        sprite_sm <= sprite_sm + 1;
      end if;
    end if;
  end process;

  --sprite_we_y(x) is the xth quarter of curent line of sprite #y

  sprite_we_0(0) <= '1' when sprite_sm = "00000" else '0';
  sprite_we_0(1) <= '1' when sprite_sm = "00001" else '0';
  sprite_we_0(2) <= '1' when sprite_sm = "00010" else '0';
  sprite_we_0(3) <= '1' when sprite_sm = "00011" else '0';
  sprite_we_1(0) <= '1' when sprite_sm = "00100" else '0';
  sprite_we_1(1) <= '1' when sprite_sm = "00101" else '0';
  sprite_we_1(2) <= '1' when sprite_sm = "00110" else '0';
  sprite_we_1(3) <= '1' when sprite_sm = "00111" else '0';
  sprite_we_2(0) <= '1' when sprite_sm = "01000" else '0';
  sprite_we_2(1) <= '1' when sprite_sm = "01001" else '0';
  sprite_we_2(2) <= '1' when sprite_sm = "01010" else '0';
  sprite_we_2(3) <= '1' when sprite_sm = "01011" else '0';
  sprite_we_3(0) <= '1' when sprite_sm = "01100" else '0';
  sprite_we_3(1) <= '1' when sprite_sm = "01101" else '0';
  sprite_we_3(2) <= '1' when sprite_sm = "01110" else '0';
  sprite_we_3(3) <= '1' when sprite_sm = "01111" else '0';


  sprite_data_16 <= SRAM_data_in;

  current_sprite_num <= sprite_num_0 when  SRAM_sm(3 downto 2) = "00"
else
                        sprite_num_1 when  SRAM_sm(3 downto 2) = "01"
else
                        sprite_num_2 when  SRAM_sm(3 downto 2) = "10"
else
                        sprite_num_3 when  SRAM_sm(3 downto 2) = "11";

  current_sprite_addr <= sprite_addr_0 when SRAM_sm(3 downto 2) = "00"
else
```

```
                          sprite_addr_1 when SRAM_sm(3 downto 2) = "01"
else
                          sprite_addr_2 when SRAM_sm(3 downto 2) = "10"
else
                          sprite_addr_3 when SRAM_sm(3 downto 2) = "11";


  SRAM_addr <= "0000001" & current_sprite_num & current_sprite_addr &
SRAM_sm(1 downto 0);
  SRAM_req <= not SRAM_sm(4);

end Behavioral;
```

5.2.5 sprite_video.vhd

```
------------------------------------------------------------------------
--
-- Sprite Video
-- Connects the VGA module to the OPB Bus
-- Created by Dagna Harasim
--
------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity sprite_video is

  generic (
    C_OPB_AWIDTH    : integer := 32;
    C_OPB_DWIDTH    : integer := 32;
    C_BASEADDR      : std_logic_vector := X"FEFD_0000";
    C_HIGHADDR      : std_logic_vector := X"FEFD_00FF"
    );

  port (

    OPB_Clk : in std_logic;
    OPB_Rst : in std_logic;
    OPB_ABus : in std_logic_vector (31 downto 0);
    OPB_BE : in std_logic_vector (3 downto 0);
    OPB_DBus : in std_logic_vector (31 downto 0);
    OPB_RNW : in std_logic;
    OPB_select : in std_logic;
    OPB_seqAddr : in std_logic;

    UIO_DBus : out std_logic_vector (31 downto 0);
    UIO_errAck : out std_logic;
    UIO_retry : out std_logic;
    UIO_toutSup : out std_logic;
    UIO_xferAck : out std_logic;

    pixel_clock        : in std_logic;

    video_data     : in std_logic_vector(15 downto 0);
    video_addr     : out std_logic_vector(19 downto 0);
    video_req      : out std_logic;

    VIDOUT_CLK     : out std_logic;
    VIDOUT_RCR     : out std_logic_vector(9 downto 0);
    VIDOUT_GY      : out std_logic_vector(9 downto 0);
    VIDOUT_BCB     : out std_logic_vector(9 downto 0);
    VIDOUT_BLANK_N : out std_logic;
    VIDOUT_HSYNC_N : out std_logic;
    VIDOUT_VSYNC_N : out std_logic);
end sprite_video;
```

```
architecture xxx of sprite_video is

component vga
  port (
    clk            : in std_logic;
    pix_clk        : in std_logic;
    rst            : in std_logic;
    video_data     : in std_logic_vector(15 downto 0);
    video_addr     : out std_logic_vector(19 downto 0);
    video_req      : out std_logic;
    x0, x1, x2, x3  : in std_logic_vector (10 downto 0);
    y0, y1, y2, y3  : in std_logic_vector (9 downto 0);
    sp0, sp1, sp2, sp3  : in std_logic_vector (6 downto 0);
    VIDOUT_CLK     : out std_logic;
    VIDOUT_RCR     : out std_logic_vector(9 downto 0);
    VIDOUT_GY      : out std_logic_vector(9 downto 0);
    VIDOUT_BCB     : out std_logic_vector(9 downto 0);
    VIDOUT_BLANK_N : out std_logic;
    VIDOUT_HSYNC_N : out std_logic;
    VIDOUT_VSYNC_N : out std_logic);
end component;

signal x0, y0, x1, y1, x2, y2, x3, y3, sp0, sp1, sp2, sp3:
std_logic_vector(15 downto 0);

signal addr, wdata : std_logic_vector(15 downto 0);
signal rnw : std_logic;
signal cs, q0, q1, q2, we: std_logic;

begin

vga_i : vga port map (
    clk => OPB_Clk,
    pix_clk => pixel_clock,
    rst => OPB_Rst,
    video_data => video_data,
    video_addr => video_addr,
    video_req => video_req,
    x0 => x0(10 downto 0),
    x1 => x1(10 downto 0),
    x2 => x2(10 downto 0),
    x3 => x3(10 downto 0),

    y0 => y0(9 downto 0),
    y1 => y1(9 downto 0),
    y2 => y2(9 downto 0),
    y3 => y3(9 downto 0),

    sp0 => sp0(6 downto 0),
    sp1 => sp1(6 downto 0),
    sp2 => sp2(6 downto 0),
    sp3 => sp3(6 downto 0),

    VIDOUT_CLK => VIDOUT_CLK,
    VIDOUT_RCR => VIDOUT_RCR,
    VIDOUT_GY => VIDOUT_GY,
```

```
    VIDOUT_BCB => VIDOUT_BCB,
    VIDOUT_BLANK_N => VIDOUT_BLANK_N,
    VIDOUT_HSYNC_N => VIDOUT_HSYNC_N,
    VIDOUT_VSYNC_N => VIDOUT_VSYNC_N
);

-- OPB interface

cs <= OPB_select when OPB_ABus(31 downto 16) = X"FEFD" else '0';

process (OPB_RST, OPB_Clk)
begin
  if OPB_Clk'event and OPB_Clk='1' then
    addr <= OPB_ABus(15 downto 0);
    wdata <= OPB_DBus(31 downto 16);
    rnw <= OPB_RNW;
  end if;
end process;

-- State machine for OPB interface
process (OPB_Clk)
begin
  if OPB_Clk'event and OPB_Clk='1' then
    q2 <= (not q2 and q1) or (q2 and not q1);
    q1 <= (cs and not q2 and not q1) or (q2 and not q1);
    q0 <= q2 and not q1;
  end if;
end process;

we <= q2 and not q1 and not rnw;


process (OPB_Clk)
begin
  if OPB_Clk'event and OPB_Clk='1' and we='1' then
    case addr(5 downto 2) is
      when  "0000" => x0 <= wdata;
      when  "0001" => y0 <= wdata;
      when  "0010" => sp0 <= wdata;

      when  "0100" => x1 <= wdata;
      when  "0101" => y1 <= wdata;
      when  "0110" => sp1 <= wdata;

      when  "1000" => x2 <= wdata;
      when  "1001" => y2 <= wdata;
      when  "1010" => sp2 <= wdata;

      when  "1100" => x3 <= wdata;
      when  "1101" => y3 <= wdata;
      when  "1110" => sp3 <= wdata;

      when others => null;
    end case;
  end if;
end process;
```

```
UIO_xferAck <= q0;

UIO_DBus <= X"00000000";
UIO_errAck <= '0';
UIO_retry <= '0';
UIO_toutSup <= '0';

end xxx;
```

5.2.6 vga.vhd

```
----------------------------------------------------------------------
--
-- VGA video generator
-- Uses the vga_timing module to generate hsync etc.
-- Cristian Soviani, Dennis Lim, and Stephen A. Edwards
-- Modified by Dagna Harasim and David Soofian
--
----------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity vga is
  port (
    clk            : in std_logic;
    pix_clk        : in std_logic;
    rst            : in std_logic;
    video_data     : in std_logic_vector(15 downto 0);
    video_addr     : out std_logic_vector(19 downto 0);
    video_req      : out std_logic;

    x0, x1, x2, x3      : in std_logic_vector (10 downto 0);
    y0, y1, y2, y3      : in std_logic_vector (9 downto 0);
    sp0, sp1, sp2, sp3  : in std_logic_vector (6 downto 0);

    VIDOUT_CLK     : out std_logic;
    VIDOUT_RCR     : out std_logic_vector(9 downto 0);
    VIDOUT_GY      : out std_logic_vector(9 downto 0);
    VIDOUT_BCB     : out std_logic_vector(9 downto 0);
    VIDOUT_BLANK_N : out std_logic;
    VIDOUT_HSYNC_N : out std_logic;
    VIDOUT_VSYNC_N : out std_logic);
end vga;

architecture Behavioral of vga is

component rom_1024_8 is
  port (
    Clk  : in std_logic;
    en   : in std_logic; -- Read enable
    addr : in std_logic_vector(9 downto 0);
    data : out std_logic_vector(7 downto 0));
  end component;

  -- Fast low-voltage TTL-level I/O pad with 12 mA drive

  component OBUF_F_12
    port (
      O : out STD_ULOGIC;
      I : in STD_ULOGIC);
  end component;
```

```
  -- Basic edge-sensitive flip-flop

  component FD
    port (
      C : in std_logic;
      D : in std_logic;
      Q : out std_logic);
  end component;

  -- Force instances of FD into pads for speed

  attribute iob : string;
  attribute iob of FD : component is "true";

  component vga_timing
    port (
      pixel_clock          : in std_logic;
      reset                : in std_logic;
      h_sync_delay         : out std_logic;
      v_sync_delay         : out std_logic;
      blank                : out std_logic;
      vga_ram_read_address : out std_logic_vector (19 downto 0);
      char_ld              : out std_logic;
      char_sh              : out std_logic;
      pix_ld               : out std_logic;
      char_line            : out std_logic_vector(2 downto 0);
      sprite_en            : out std_logic;
      line_cnt             : out std_logic_vector (9 downto 0);   -- Y
coordinate
      pixel_cnt            : out std_logic_vector (10 downto 0)   -- X
coordinate

      );
  end component;

 component sprite
  port (
    pixclk              : in std_logic;
    sprreset            : in std_logic;
    x                   : in std_logic_vector (10 downto 0);
    y                   : in std_logic_vector (9 downto 0);
    line_cnt            : in std_logic_vector (9 downto 0);
    pix_cnt             : in std_logic_vector (10 downto 0);
    sprite_data_16      : in std_logic_vector(15 downto 0);
    sprite_data_we      : in std_logic_vector(3 downto 0);
    sprite_data_addr    : out std_logic_vector(3 downto 0);
    video_data          : out std_logic_vector(3 downto 0)
  );
 end component;

component display_prio
  port(
    vgachar     : in std_logic_vector(3 downto 0);
    sprite0     : in std_logic_vector(3 downto 0);
    sprite1     : in std_logic_vector(3 downto 0);
    sprite2     : in std_logic_vector(3 downto 0);
    sprite3     : in std_logic_vector(3 downto 0);
```

```
    r           : out std_logic_vector(7 downto 0);
    g           : out std_logic_vector(7 downto 0);
    b           : out std_logic_vector(7 downto 0)
  );
end component;

component sprite_loader
  port (
    pixclk              : in std_logic;
    reset               : in std_logic;
    SRAM_data_in        : in std_logic_vector (15 downto 0);
    sprite_ld           : in std_logic;

    sprite_addr_0, sprite_addr_1, sprite_addr_2, sprite_addr_3  : in
std_logic_vector (3 downto 0);
    sprite_num_0, sprite_num_1, sprite_num_2, sprite_num_3      : in
std_logic_vector (6 downto 0);
    sprite_we_0, sprite_we_1, sprite_we_2, sprite_we_3          : out
std_logic_vector (3 downto 0);

    sprite_data_16      : out std_logic_vector(15 downto 0);
    SRAM_addr           : out std_logic_vector(19 downto 0);
    SRAM_req            : out std_logic
    );
end component;


signal line_cnt                 : std_logic_vector (9 downto 0);   -- Y
coordinate
signal pixel_cnt                : std_logic_vector (10 downto 0);  -- X
coordinate

signal r, g, b                  : std_logic_vector (9 downto 0);
signal r_t, g_t, b_t            : std_logic_vector (7 downto 0);

signal blank, blank_1, blank_2  : std_logic;
signal hsync, hsync_1, hsync_2  : std_logic;
signal vsync, vsync_1, vsync_2  : std_logic;
signal vga_ram_read_address     : std_logic_vector(19 downto 0);
signal vreq                     : std_logic;
signal vga_shreg                : std_logic_vector(15 downto 0);
signal char_out                 : std_logic_vector(7 downto 0);
signal charld                   : std_logic;
signal charsh                   : std_logic;
signal pixld                    : std_logic;
signal charline                 : std_logic_vector(2 downto 0);
signal displayed_pixel          : std_logic;
signal vga_color                : std_logic_vector(3 downto 0);
signal rom_data                 : std_logic_vector(7 downto 0);
signal rom_addr                 : std_logic_vector(9 downto 0);
signal bram_video_data          : std_logic_vector(15 downto 0);
signal pixshreg                 : std_logic_vector(7 downto 0);
signal sprite_en                : std_logic;
signal sprite_data_16           : std_logic_vector(15 downto 0);
signal sprite_SRAM_addr         : std_logic_vector(19 downto 0);
signal sprite_SRAM_req          : std_logic;
```

```
signal sprite0_vdata, sprite1_vdata, sprite2_vdata, sprite3_vdata
: std_logic_vector(3 downto 0);
signal sprite0_we, sprite1_we, sprite2_we, sprite3_we
: std_logic_vector(3 downto 0);
signal sprite_addr0, sprite_addr1, sprite_addr2, sprite_addr3
: std_logic_vector(3 downto 0);

begin

  st : vga_timing
    port map (
    pixel_clock => pix_clk,
    reset => rst,
    h_sync_delay => hsync,
    v_sync_delay => vsync,
    blank => blank,
    vga_ram_read_address => vga_ram_read_address,
    char_ld => charld,
    char_sh => charsh,
    pix_ld => pixld,
    char_line => charline,
    sprite_en => sprite_en,
    line_cnt => line_cnt,
    pixel_cnt => pixel_cnt
    );


  spr0 : sprite
    port map (
    pixclk => pix_clk,
    sprreset => rst,
    x => x0, -- 11 bits
    y => y0,
    line_cnt => line_cnt,
    pix_cnt => pixel_cnt,
    sprite_data_16 => sprite_data_16,
    sprite_data_we => sprite0_we,
    sprite_data_addr => sprite_addr0,
    video_data => sprite0_vdata
    );

  spr1 : sprite
    port map (
    pixclk => pix_clk,
    sprreset => rst,
    x => x1,
    y => y1,
    line_cnt => line_cnt,
    pix_cnt => pixel_cnt,
    sprite_data_16 => sprite_data_16,
    sprite_data_we => sprite1_we,
    sprite_data_addr => sprite_addr1,
    video_data => sprite1_vdata
    );

  spr2 : sprite
    port map (
```

```
  pixclk => pix_clk,
  sprreset => rst,
  x => x2,
  y => y2,
  line_cnt => line_cnt,
  pix_cnt => pixel_cnt,
  sprite_data_16 => sprite_data_16,
  sprite_data_we => sprite2_we,
  sprite_data_addr => sprite_addr2,
  video_data => sprite2_vdata
  );

spr3 : sprite
  port map (
  pixclk => pix_clk,
  sprreset => rst,
  x => x3,
  y => y3,
  line_cnt => line_cnt,
  pix_cnt => pixel_cnt,
  sprite_data_16 => sprite_data_16,
  sprite_data_we => sprite3_we,
  sprite_data_addr => sprite_addr3,
  video_data => sprite3_vdata
  );

sp_ld : sprite_loader
port map(
  pixclk => pix_clk,
  reset => rst,
  SRAM_data_in => video_data,
  sprite_ld => sprite_en,
  sprite_addr_0 => sprite_addr0,
  sprite_addr_1 => sprite_addr1,
  sprite_addr_2 => sprite_addr2,
  sprite_addr_3 => sprite_addr3,
  sprite_num_0 => sp0,
  sprite_num_1 => sp1,
  sprite_num_2 => sp2,
  sprite_num_3 => sp3,
  sprite_we_0 => sprite0_we,
  sprite_we_1 => sprite1_we,
  sprite_we_2 => sprite2_we,
  sprite_we_3 => sprite3_we,
  sprite_data_16 => sprite_data_16,
  SRAM_addr => sprite_SRAM_addr,
  SRAM_req => sprite_SRAM_req
 );

rom_map : rom_1024_8
  port map (
    Clk => clk,
    en => '1',
    addr => rom_addr,
    data => rom_data
    );
```

```
  -- Video request is true when the RAM address is even

  -- FIXME: This should be disabled during blanking to reduce memory
traffic

  vreq <= '1';

  -- Generate video_req (to the RAM controller) by delaying vreq by
  -- a cycle synchronized with the pixel clock

  process (clk)
  begin
    if clk'event and clk='1' then
      video_req <= pix_clk and vreq;
    end if;
  end process;

  -- The video address is the upper 19 bits from the VGA timing
generator
  -- because we are using two pixels per word and the RAM address
counts words

  video_addr <= '0' & vga_ram_read_address(19 downto 1) when
sprite_SRAM_req = '0' else
                sprite_SRAM_addr;



  -- The video shift register: either load it from RAM or shift it up a
byte

  process (pix_clk)
  begin
    if pix_clk'event and pix_clk='1' then
      if charld = '1' then
        vga_shreg <= video_data;
      elsif charsh = '1' then
        vga_shreg <= vga_shreg(7 downto 0) & "00000000";
      end if;
  end if;
  end process;

  char_out <= vga_shreg(15 downto 8);
  rom_addr <= char_out(6 downto 0) & charline (2 downto 0);

  process (pix_clk)
  begin
    if pix_clk'event and pix_clk='1' then
      if (pixld ='1') then
        pixshreg <= rom_data;
      else
        pixshreg <= pixshreg(6 downto 0) & '0';     --shift
      end if;
    end if;
  end process;

  displayed_pixel <= pixshreg(7);        --display 1 bit at a time
```

```
  vga_color <= "0011" when displayed_pixel = '1' else "0000";

--assign appropiate color to each pixel and choose which one to display
 dprio : display_prio
  port map (
    vgachar => vga_color,
    sprite0 => sprite0_vdata,
    sprite1 => sprite1_vdata,
    sprite2 => sprite2_vdata,
    sprite3 => sprite3_vdata,

    r => r_t,
    g => g_t,
    b => b_t
  );

r <= r_t & "00";
g <= g_t & "00";
b <= b_t & "00";


process (pix_clk)
  begin
    if pix_clk'event and pix_clk='1' then
      hsync_1 <= hsync;
      hsync_2 <= hsync_1;
      vsync_1 <= vsync;
      vsync_2 <= vsync_1;
      blank_1 <= blank;
      blank_2 <= blank_1;
    end if;
end process;

  -- Video clock I/O pad to the DAC

  vidclk : OBUF_F_12 port map (
    O => VIDOUT_clk,
    I => pix_clk);

  -- Control signals: hsync, vsync, and blank
  hsync_ff : FD port map (
    C => pix_clk,
    D => not hsync_2,
    Q => VIDOUT_HSYNC_N );

  vsync_ff : FD port map (
    C => pix_clk,
    D => not vsync_2,
    Q => VIDOUT_VSYNC_N );

  blank_ff :  FD port map (
    C => pix_clk,
    D => not blank_2,
    Q => VIDOUT_BLANK_N );
```

```
-- Three digital color signals

  rgb_ff : for i in 0 to 9 generate

    r_ff : FD port map (
      C => pix_clk,
      D => r(i),
      Q => VIDOUT_RCR(i) );

    g_ff : FD port map (
      C => pix_clk,
      D => g(i),
      Q => VIDOUT_GY(i) );

    b_ff : FD port map (
      C => pix_clk,
      D => b(i),
      Q => VIDOUT_BCB(i) );


  end generate;
end Behavioral;
```

5.3.7 vga_timing.vhd

```
----------------------------------------------------------------------
--
-- VGA timing and address generator
--
-- Fixed-resolution address generator.  Generates h-sync, v-sync, and
-- blanking signals along with a 20-bit RAM address.  H-sync and v-sync
-- signals are delayed two cycles to compensate for the DAC pipeline.
--
-- Cristian Soviani, Dennis Lim, and Stephen A. Edwards
-- Modified by Dagna Harasim and David Soofian
--
----------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity vga_timing is
  port (
    pixel_clock                : in std_logic;
    reset                      : in std_logic;
    h_sync_delay               : out std_logic;
    v_sync_delay               : out std_logic;
    blank                      : out std_logic;
    vga_ram_read_address       : out std_logic_vector(19 downto 0);
    --bram_read_address          : out std_logic_vector(19 downto 0);
    char_ld                    : out std_logic;
    char_sh                    : out std_logic;
    pix_ld                     : out std_logic;
    char_line                  : out std_logic_vector(2 downto 0);
    sprite_en                  : out std_logic;
    line_cnt                   : out std_logic_vector (9 downto 0);
-- Y coordinate
    pixel_cnt                  : out std_logic_vector (10 downto 0)
-- X coordinate
    );
end vga_timing;

architecture Behavioral of vga_timing is

  constant SRAM_DELAY : integer := 5;
  -- 640 X 480 @ 60Hz with a 25.175 MHz pixel clock
  constant H_ACTIVE      : integer := 640;
  constant H_FRONT_PORCH : integer := 16;
  constant H_BACK_PORCH  : integer := 48;
  constant H_TOTAL       : integer := 800;
  constant V_ACTIVE      : integer := 480;
  constant V_FRONT_PORCH : integer := 11;
  constant V_BACK_PORCH  : integer := 31;
  constant V_TOTAL       : integer := 524;
```

```
  signal line_count      : std_logic_vector (9 downto 0);   -- Y
coordinate
  signal pixel_count     : std_logic_vector (10 downto 0);  -- X
coordinate
  signal cnt40           : std_logic_vector (19 downto 0);  -- vertical
character

  signal h_sync          : std_logic;                       --
horizontal sync
  signal v_sync          : std_logic;                       -- vertical
sync
  signal h_sync_delay0   : std_logic;                       -- h_sync
delayed 1 clock
  signal v_sync_delay0   : std_logic;                -- v_sync delayed
1 clock
  signal h_blank         : std_logic;                -- horizontal
blanking
  signal v_blank         : std_logic;                       -- vertical

  -- flag to reset the ram address during vertical blanking
  signal reset_vga_ram_read_address     : std_logic;
  -- flag to hold the address during horizontal blanking
  signal hold_vga_ram_read_address      : std_logic;
  signal ram_address_counter            : std_logic_vector (19 downto
0);
  signal bram_address_counter           : std_logic_vector (19 downto
0);

begin

  -- Pixel counter

  process ( pixel_clock, reset )
  begin
    if reset = '1' then
      pixel_count <= "00000000000";
    elsif pixel_clock'event and pixel_clock = '1' then
      if pixel_count = (H_TOTAL - 1) then
        pixel_count <= "00000000000";
      else
        pixel_count <= pixel_count + 1;
      end if;
    end if;
  end process;

  --set signals


  -- Horizontal sync

  process ( pixel_clock, reset )
  begin
    if reset = '1' then
      h_sync <= '0';
    elsif pixel_clock'event and pixel_clock = '1' then
      if pixel_count = (H_ACTIVE + H_FRONT_PORCH - 1) then
        h_sync <= '1';
```

```
      elsif pixel_count = (H_TOTAL - H_BACK_PORCH - 1) then
        h_sync <= '0';
      end if;
    end if;
  end process;

  -- Line counter

  process ( pixel_clock, reset )
  begin
    if reset = '1' then
      line_count <= "0000000000";
    elsif pixel_clock'event and pixel_clock = '1' then
      if ((line_count = V_TOTAL - 1) and (pixel_count = H_TOTAL - 1))
then
        line_count <= "0000000000";
      elsif pixel_count = (H_TOTAL - 1) then
        line_count <= line_count + 1;
      end if;
    end if;
  end process;

  char_line <= line_count(2 downto 0);

  -- Vertical sync

  process ( pixel_clock, reset )
  begin
    if reset = '1' then
      v_sync <= '0';
    elsif pixel_clock'event and pixel_clock = '1' then
      if line_count = (V_ACTIVE + V_FRONT_PORCH -1) and
         pixel_count = (H_TOTAL - 1) then
        v_sync <= '1';
      elsif line_count = (V_TOTAL - V_BACK_PORCH - 1) and
            pixel_count = (H_TOTAL - 1) then
        v_sync <= '0';
      end if;
    end if;
  end process;

  -- Add two-cycle delays to h/v_sync to compensate for the DAC
pipeline

  process ( pixel_clock, reset )
  begin
    if reset = '1' then
      h_sync_delay0 <= '0';
      v_sync_delay0 <= '0';
      h_sync_delay  <= '0';
      v_sync_delay  <= '0';
    elsif pixel_clock'event and pixel_clock = '1' then
      h_sync_delay0 <= h_sync;
      v_sync_delay0 <= v_sync;
      h_sync_delay  <= h_sync_delay0;
      v_sync_delay  <= v_sync_delay0;
    end if;
```

```
  end process;

  -- Horizontal blanking

  -- The constants are offset by two to compensate for the delay
  -- in the composite blanking signal

  process ( pixel_clock, reset )
  begin
    if reset = '1' then
      h_blank <= '0';
    elsif pixel_clock'event and pixel_clock = '1' then
      if pixel_count = (H_ACTIVE - 2) then
        h_blank <= '1';
      elsif pixel_count = (H_TOTAL - 2) then
        h_blank <= '0';
      end if;
    end if;
  end process;

  -- Vertical Blanking

  -- The constants are offset by two to compensate for the delay
  -- in the composite blanking signal

  process ( pixel_clock, reset )
  begin
    if reset = '1' then
      v_blank <= '0';
    elsif pixel_clock'event and pixel_clock = '1' then
      if line_count = (V_ACTIVE - 1) and pixel_count = (H_TOTAL - 2)
then
        v_blank <= '1';
      elsif line_count = (V_TOTAL - 1) and pixel_count = (H_TOTAL - 2)
then
        v_blank <= '0';
      end if;
    end if;
  end process;

  -- Composite blanking

  process ( pixel_clock, reset )
  begin
    if reset = '1' then
      blank <= '0';
    elsif pixel_clock'event and pixel_clock = '1' then
      if (h_blank or v_blank) = '1' then
        blank <= '1';
      else
        blank <= '0';
      end if;
    end if;
  end process;

  -- RAM address counter
```

```
  -- Two control signals:

  -- reset_ram_read_address is active from the end of each field until
the
  -- beginning of the next

  -- hold_vga_ram_read_address is active from the end of each line to
the
  -- start of the next

  process ( pixel_clock, reset )
  begin
    if reset = '1' then
      reset_vga_ram_read_address <= '0';
    elsif pixel_clock'event and pixel_clock = '1' then
      if line_count = V_ACTIVE - 1 and
         pixel_count = ( (H_TOTAL - 1 ) - SRAM_DELAY ) then
        -- reset the address counter at the end of active video
        reset_vga_ram_read_address <= '1';
      elsif line_count = V_TOTAL - 1 and
            pixel_count =((H_TOTAL -1) - SRAM_DELAY) then
        -- re-enable the address counter at the start of active video
        reset_vga_ram_read_address <= '0';
      end if;
    end if;
  end process;

  process ( pixel_clock, reset )
  begin
    if reset = '1' then
      hold_vga_ram_read_address <= '0';
    elsif pixel_clock'event and pixel_clock = '1' then
      if pixel_count = ((H_ACTIVE - 1) - SRAM_DELAY) then
        -- hold the address counter at the end of active video
        hold_vga_ram_read_address <= '1';
      elsif pixel_count = ((H_TOTAL - 1) - SRAM_DELAY) then
        -- re-enable the address counter at the start of active video
        hold_vga_ram_read_address <= '0';
      end if;
    end if;
  end process;

 --repeat same set of addresses 8 times
  process ( pixel_clock )
  begin
   if pixel_clock'event and pixel_clock = '1' then
   if (pixel_count = 0) then
      if (line_count = V_TOTAL - 1) then
        cnt40 <= "00000000000000000000";
        elsif (line_count(2 downto 0) = "111") then
      cnt40 <= cnt40 + 80;
      end if;
    end if;
  end if;
  end process;

  -- advance one byte every 8 pixels
```

```
  process ( pixel_clock )
  begin
    if pixel_clock'event and pixel_clock = '1' then
      if (pixel_count = H_TOTAL-1-SRAM_DELAY) then
        ram_address_counter <= cnt40;
      elsif (pixel_count(3 downto 0) = "1010" ) then
        ram_address_counter <= ram_address_counter + 2;
      end if;
    end if;
  end process;

  --set pix_ld
  process ( pixel_clock )
  begin
    if pixel_clock'event and pixel_clock = '1' then
      if (pixel_count(3 downto 0)="0000" or pixel_count(3 downto
0)="1000") then
        pix_ld <= '1';
      else
        pix_ld <= '0';
      end if;
    end if;
  end process;

  --set char_ld
  process ( pixel_clock )
  begin
    if pixel_clock'event and pixel_clock = '1' then
      if (pixel_count(3 downto 0) = "1110") then
        char_ld <= '1';
      else
        char_ld <= '0';
      end if;
    end if;
  end process;

  --set char_sh
  process ( pixel_clock )
  begin
    if pixel_clock'event and pixel_clock = '1' then
      if (pixel_count(3 downto 0) = "0110") then
        char_sh <= '1';
      else
        char_sh <= '0';
      end if;
    end if;
  end process;

--set sprite_en
  process ( pixel_clock )
  begin
    if pixel_clock'event and pixel_clock = '1' then
      if pixel_count =  H_ACTIVE + 4
          then
            sprite_en <= '1';
          else
            sprite_en <= '0';
```

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


  vga_ram_read_address <= ram_address_counter;

  pixel_cnt <= pixel_count;
  line_cnt <= line_count;

end Behavioral;
```

5.2.8 system.mhs

```
--------------------------------------------------------------------------
-- Modified by Dagna Harasim
--------------------------------------------------------------------------

# Parameters
PARAMETER VERSION = 2.0.0

# Global Ports

PORT PB_A = PB_A, DIR = OUT, VEC = [19:0]
PORT PB_D = PB_D, DIR = INOUT, VEC = [15:0]
PORT PB_LB_N = PB_LB_N, DIR = OUT
PORT PB_UB_N = PB_UB_N, DIR = OUT
PORT PB_WE_N = PB_WE_N, DIR = OUT
PORT PB_OE_N = PB_OE_N, DIR = OUT
PORT RAM_CE_N = RAM_CE_N, DIR = OUT
PORT VIDOUT_CLK = VIDOUT_CLK, DIR = OUT
PORT VIDOUT_HSYNC_N = VIDOUT_HSYNC_N, DIR = OUT
PORT VIDOUT_VSYNC_N = VIDOUT_VSYNC_N, DIR = OUT
PORT VIDOUT_BLANK_N = VIDOUT_BLANK_N, DIR = OUT
PORT VIDOUT_RCR = VIDOUT_RCR, DIR = OUT, VEC = [9:0]
PORT VIDOUT_GY = VIDOUT_GY, DIR = OUT, VEC = [9:0]
PORT VIDOUT_BCB = VIDOUT_BCB, DIR = OUT, VEC = [9:0]
PORT FPGA_CLK1 = FPGA_CLK1, DIR = IN
PORT RS232_TD = RS232_TD, DIR=OUT
PORT RS232_RD = RS232_RD, DIR=IN
PORT AU_CSN_N =  AU_CSN_N, DIR=OUT
PORT AU_BCLK =  AU_BCLK, DIR=OUT
PORT AU_MCLK = AU_MCLK, DIR=OUT
PORT AU_LRCK = AU_LRCK, DIR=OUT
PORT AU_SDTI = AU_SDTI, DIR=OUT
PORT AU_SDTO0 = AU_SDTO0, DIR=IN

# Sub Components


BEGIN microblaze
 PARAMETER INSTANCE = mymicroblaze
 PARAMETER HW_VER = 2.00.a
 PARAMETER C_USE_BARREL = 1
 PARAMETER C_USE_ICACHE = 1
 PARAMETER C_ADDR_TAG_BITS = 6
 PARAMETER C_CACHE_BYTE_SIZE = 2048
 PARAMETER C_ICACHE_BASEADDR = 0x00860000
 PARAMETER C_ICACHE_HIGHADDR = 0x0087FFFF
```

```
 PORT Clk = sys_clk
 PORT Reset = fpga_reset
 PORT Interrupt = intr
 BUS_INTERFACE DLMB = d_lmb
 BUS_INTERFACE ILMB = i_lmb
 BUS_INTERFACE DOPB = myopb_bus
 BUS_INTERFACE IOPB = myopb_bus
END

BEGIN opb_intc
 PARAMETER INSTANCE = intc
 PARAMETER HW_VER = 1.00.c
 PARAMETER C_BASEADDR = 0xFFFF0000
 PARAMETER C_HIGHADDR = 0xFFFF00FF
 PORT OPB_Clk = sys_clk
 PORT Intr =  uart_intr
 PORT Irq =   intr
 BUS_INTERFACE SOPB = myopb_bus
END

BEGIN bram_block
 PARAMETER INSTANCE = bram
 PARAMETER HW_VER = 1.00.a
 BUS_INTERFACE PORTA = conn_0
 BUS_INTERFACE PORTB = conn_1
END

BEGIN opb_xsb300
 PARAMETER INSTANCE = xsb300
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_BASEADDR = 0x00800000
 PARAMETER C_HIGHADDR = 0x00FFFFFF
 PORT PB_A = PB_A
 PORT PB_D = PB_D
 PORT PB_LB_N = PB_LB_N
 PORT PB_UB_N = PB_UB_N
 PORT PB_WE_N = PB_WE_N
 PORT PB_OE_N = PB_OE_N
 PORT RAM_CE_N = RAM_CE_N
 PORT OPB_Clk = sys_clk

 BUS_INTERFACE SOPB = myopb_bus

 PORT video_req = video_req
 PORT video_addr = video_addr
 PORT video_data = video_data
```

```
END

BEGIN sprite_video

 PARAMETER INSTANCE = sprites
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_BASEADDR = 0xFEFD0000
 PARAMETER C_HIGHADDR = 0xFEFDFFFF

 PORT OPB_Clk = sys_clk
 PORT pixel_clock = pixel_clock
 PORT VIDOUT_CLK = VIDOUT_CLK
 PORT VIDOUT_HSYNC_N = VIDOUT_HSYNC_N
 PORT VIDOUT_VSYNC_N = VIDOUT_VSYNC_N
 PORT VIDOUT_BLANK_N = VIDOUT_BLANK_N
 PORT VIDOUT_RCR = VIDOUT_RCR
 PORT VIDOUT_GY = VIDOUT_GY
 PORT VIDOUT_BCB = VIDOUT_BCB
 BUS_INTERFACE SOPB = myopb_bus

 PORT video_req = video_req
 PORT video_addr = video_addr
 PORT video_data = video_data

END


BEGIN clkgen
 PARAMETER INSTANCE = clkgen_0
 PARAMETER HW_VER = 1.00.a
 PORT FPGA_CLK1 = FPGA_CLK1
 PORT sys_clk = sys_clk
 PORT pixel_clock = pixel_clock
 PORT fpga_reset = fpga_reset
END

BEGIN lmb_lmb_bram_if_cntlr
 PARAMETER INSTANCE = lmb_lmb_bram_if_cntlr_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_BASEADDR = 0x00000000
 PARAMETER C_HIGHADDR = 0x00000FFF
 BUS_INTERFACE DLMB = d_lmb
 BUS_INTERFACE ILMB = i_lmb
 BUS_INTERFACE PORTA = conn_0
 BUS_INTERFACE PORTB = conn_1
```

```
END

BEGIN opb_uartlite
 PARAMETER INSTANCE = myuart
 PARAMETER HW_VER = 1.00.b
 PARAMETER C_CLK_FREQ = 50_000_000
 PARAMETER C_USE_PARITY = 0
 PARAMETER C_BASEADDR = 0xFEFF0100
 PARAMETER C_HIGHADDR = 0xFEFF01FF
 PORT OPB_Clk = sys_clk
 PORT Interrupt = uart_intr
 BUS_INTERFACE SOPB = myopb_bus
 PORT RX=RS232_RD
 PORT TX=RS232_TD
END

BEGIN opb_v20
 PARAMETER INSTANCE = myopb_bus
 PARAMETER HW_VER = 1.10.a
 PARAMETER C_DYNAM_PRIORITY = 0
 PARAMETER C_REG_GRANTS = 0
 PARAMETER C_PARK = 0
 PARAMETER C_PROC_INTRFCE = 0
 PARAMETER C_DEV_BLK_ID = 0
 PARAMETER C_DEV_MIR_ENABLE = 0
 PARAMETER C_BASEADDR = 0x0fff1000
 PARAMETER C_HIGHADDR = 0x0fff10ff
 PORT SYS_Rst = fpga_reset
 PORT OPB_Clk = sys_clk
END

BEGIN lmb_v10
 PARAMETER INSTANCE = d_lmb
 PARAMETER HW_VER = 1.00.a
 PORT LMB_Clk = sys_clk
 PORT SYS_Rst = fpga_reset
END

BEGIN lmb_v10
 PARAMETER INSTANCE = i_lmb
 PARAMETER HW_VER = 1.00.a
 PORT LMB_Clk = sys_clk
 PORT SYS_Rst = fpga_reset
END
```