GND: Goalie, Ninja, Dodge

CSEE 4840 Final Project Report - Spring 2011

Jaiseung Bang, B.S. Computer Engineering, jb2861@columbia.edu,
Vincent Liao, B.S. Computer Engineering, vl2187@columbia.edu,
Arunagiri Venkatesan, B.S. Electrical Engineering, av2294@columbia.edu,
David Yang, B.S. Computer Engineering, qy2114@columbia.edu

May 12, 2011

Contents

Abstract

The basic goal of this project was to implement an object avoidance game that is controlled via video capture. Video from a camera is passed into the FPGA via a decoder chip. Compression and detection algorithms, implemented in hardware, will downscale and detect the presence of human bodies and other foreign objects in the view. Then the game software running on a NIOS processor will control the game mechanics using collision information that is provided by the hardware.

1 Introduction

1.1 Goals

In starting this project we wanted to better understand digital video signals, understand how to overcome the timing issues that can arise in timing critical applications, understand how to optimize the hardware to run efficiently, implement basic video processing, and creatively use technology to make a fun game.

1.2 Gameplay

Before playing the game, the background must be captured. This is done pressing button 0. There are three game modes, dodge, goalie, and ninja which are activated using the buttons 1, 2, and 3, respectively. In Dodge, players must avoid incoming red balls. Players get one point for every ball that is avoided. The game ends if the player hits a red ball. In Goalie, players must hit all incoming green balls. Players get one point for every ball that is hit. The game ends if the player misses a green ball. Ninja is a combination of the previous two. Players must dodge all red balls and hit all green balls. They get one point for each ball that is successfully handled. The game ends either when a red ball is hit or when a green ball is missed.

2 Algorithms

2.1 Downscaling

To make the game more manageable and to save hardware real estate, the video must be downscaled before any video processing is performed. To downscale the full VGA (640x480) resolution video feed to a smaller resolution, a nave averaging algorithm is used. The YCbCr values of blocks of pixels are averaged into one pixel. In other words, an eight times downscaling algorithm will take blocks that are 8x8 in size and average the values with the result describing a new, single pixel. In our implementation, the 640x480 resolution video is ultimately downscaled to 20x15.

2.2 Silhouette Generation

Silhouette Generation is the name we have given for the algorithm that processes video and subtracts the background. First, a snapshot of the background with no players present is taken. Then when the game is being played, the YCbCr values of a particular pixel in the background and the live feed are compared. If the difference of the values falls within an acceptable tolerance, then it means that there is no player present in that pixel. If the difference falls outside the tolerance, then the players body is present in that pixel.

3 Architecture

Figure 1 describes the hardware peripherals and components in the DE2 board that are used. Figure 2 is a overview of the hardware blocks and the signals between them.

Figure 1: High Level Diagram

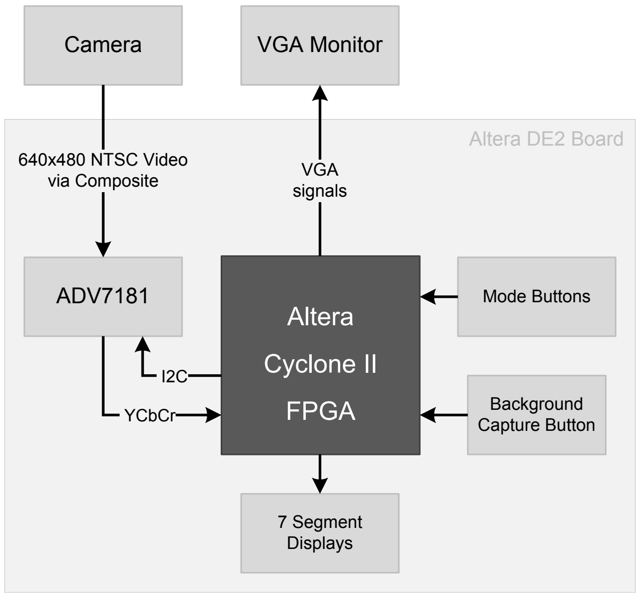
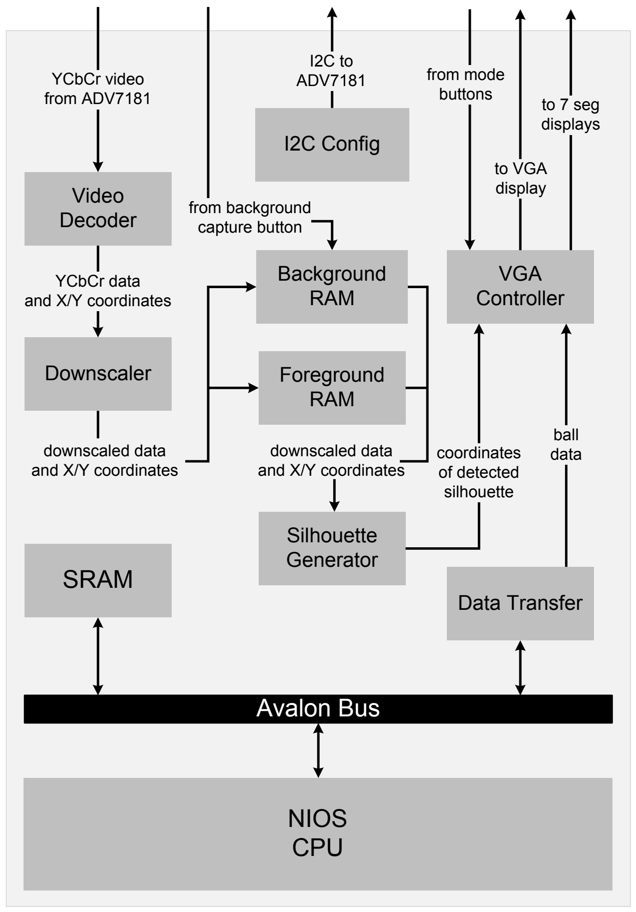


Figure 2: Block Diagram of Hardware Architecture



3.1 I2C Config

The operating mode of the ADV7181 video decoder is configured via a 2-wire serial, bidirectional port. The I2C Config component is an I2C master that configures the ADV7181 to convert from NTSC to 4:2:2 YCbCr digital video. This component is from the DE2_TV demo project from TerasIC.

3.2 Video Decoder

The video decoder receives 4:2:2 YCbCr from the ADV7181 video decoder. It outputs VGA sync signals, Y, Cb, Cr components, and X and Y coordinates for each pixel. This is a modified version of a component from the DE2_TV demo project.

3.3 Downscaler

The downscaler receives Y, Cb, Cr, X, and Y coordinates for each pixel in the 640x480 video feed. It downscales the video and outputs 80x60 Y, Cb and Cr color components and X and Y coordinates. The video feed comes in line by line. This poses a problem because squares of 8x8 pixels cannot be immediately averaged. To circumvent this problem, three intermediate line buffers store 80 sums of Y, Cb, and Cr values that correspond to 80 squares of 64 pixels in 8 lines. As each pixel at (X, Y) comes in, its YCbCr components are added to the $X/8$ index in the buffers. When the eighth line is being read in, the average of a square of 64 pixels is calculated and put into a final line buffer every eight pixels. This poses a timing issue. While the averages are being calculated only at the eighth line, we still want to output the averages at a uniform rate. To overcome this timing issue, an average from the final line buffer is outputted as every 64 pixels are read in. The index for this value is calculated using the following formula: $10(Y \bmod 8) + X/64$.

3.4 Background RAM and Foreground RAM

Both the Background RAM and Foreground RAM are instances of the same simple dual-port 4800x24-bit RAM component (specified in ram_img.vhd). This RAM component takes as input a write address (expressed as x and y coordinates), write data, a write enable bit, and a read address (expressed as x and y coordinates). The output is a single data signal (24-bit YCbCr value for the pixel at the read x and y coordinates). Both the Background RAM and Foreground RAM are used to store a 80x60 pixel YCbCr image. The total block memory usage of the Background RAM and Foreground RAM together is 230,400 out of 483,840 total RAM bits on the Cyclone II EP2C35. The purpose of the Background RAM is to store an initial background image before the game begins. It takes write data and write address input from the Downscaler, write enable input from a pushbutton, and read address input from the Silhouette Generator. It outputs read data to the Silhouette Generator. The Downscaler writes into the Background RAM when the appropriate pushbutton is pressed. Meanwhile, the Silhouette Generator continuously reads from the Background RAM, comparing it to a foreground image. The

Foreground RAM is essentially a frame buffer. It takes write data and write address input from the Downscaler and read address input from the Silhouette Generator. The write enable bit is always asserted to a high logic level. The Foreground RAM outputs read data to the Silhouette Generator. The Downscaler continuously updates the values in the Foreground RAM as the Silhouette Generator reads from the Foreground RAM.

3.5 Silhouette Generator

The Silhouette Generator takes in a pair of YCbBr data corresponding to the same X and Y coordinates from both the Background RAM and Foreground RAM. Background subtraction is performed on each pair of pixel data from an 80x60 image as they come in. If the difference between either the Y, Cb, or Cr component of the background and foreground pixels is greater than the background tolerance values, then the pixel is flagged as being part of the player. Otherwise, it is flagged as being part of the background. To filter for false positive and false negatives in this background subtraction, we downscale the image to 20x15. To accomplish this, an intermediate line buffer is used to store 20 counts of the flags that correspond to 20 squares of pixels in 4 lines. For each pixel at (X, Y), the count at the X/4 index in the intermediate line buffer is incremented if the pixel was flagged as being part of the background. When the fourth line is being read in, if the count of a square of 16 pixels is greater than 8, '1' is written to a final line buffer every four pixels. Otherwise, '0' is written. A value from the final line buffer is outputted as every 16 pixels are read in. The index for this value is calculated using the following formula: $5(y \bmod 8) + x/16$. This silhouette bit is outputted to the VGA Raster along with its corresponding X and Y coordinates in the 20x15 image.

3.6 VGA Raster

The VGA Raster generates the output signals that control the VGA Display. VGA Raster reads X and Y coordinates along with a silhouette bit from the Silhouette Generator and writes to an internal frame buffer. While printing a pixel on the screen, it checks if its corresponding silhouette bit is '1' or '0', and uses this information to determine what color to print.

The VGA Raster also interacts with NIOS to get various properties of the ball that is being displayed. Ch_in and cv_in signals, which are obtained from the SRAM, are used as the X and the Y coordinate of the center of the ball. R_in is used to determine the size of the ball that is being displayed and h_in determines whether the ball is meant to be hit or not. Depending on the mode of the game (controlled by the keys on the DE2_Board), VGA Raster checks for various end conditions when the ball grows to its maximum size (representing the depth position where the ball contacts the screen). Since the VGA Raster updates pixel by pixel, a collision can be detected when the silhouette and the ball are generated in the same pixel. In Dodge Mode, once there is a collision between the ball and the silhouette, the player loses. Otherwise, the score is incremented. Some special measures were taken in implementing the Goalie Mode. Unlike Dodge Mode, in which detecting any case where the ball and the silhouette coincide is enough to determine that the player lost, Goalie Mode does not end the game at the first instance when the ball and the silhouette do not coincide. For Goalie Mode, an internal bit is set to low at the start of drawing a frame. This bit is set to high if the player coincides with any point on the ball. At the end of drawing a frame, if the bit is still low, the game ends. Ninja Mode incorporates the two collision detection schemes outlined above and also uses the h_in signal to determine whether the ball is meant to be hit or dodged.

The software is responsible for generating a random pathway for the ball and determining the speed of the ball. A randomly generated integer from 0 to 17 determines the direction of the ball. From this value, the X and Y coordinates are incremented or decremented accordingly. The software also adjusts the rate at which the radius increases to simulate different speeds. Finally, the software generates a random bit to indicate whether the ball should be blocked or dodged (used in Ninja Mode). All this information is concatenated into a single 32-bit integer and written to SRAM.

4 Design Process

4.1 Design Decisions

- We chose to use block RAM since it is simpler to use and faster than off chip ram to use.
- We decided to use a 27Mhz clock since the input

YCbCr video signal is received at 27Mhz.

- The game was original to be "Human Tetris" where players must fit their bodies into specific shape. We decided to switch to a dodge ball style game since human tetris was too dependent on the distance of the players from the camera.

4.2 Issues

- The original prototype of the algorithm downscaled by a factor of 4 after background subtraction to filter out false positives and false negatives. We calculated the RAM blocks needed to store this result and realized that we wouldnt be able to fit the image within the block RAM. So, we decided to do a first pass of downscaling before background subtraction. We then tried to synthesize a RAM from a 2 dimensional array. This didnt create a valid RAM. It is here that we learned that 1 dimensional array must be used for describing RAMs.

- We assumed that the 27Mhz clock is already initialized. We then learned that it must be initialized by setting TD_RESET to '1'. Because we assumed that there was a clock when there wasnt, it led us to assume that the VGA display could not be clocked at 27Mhz and that the dual port RAM could not have different input and output clocks. Both problems were solved when the 27Mhz clock was initialized.

5 Conclusion

5.1 Distribution of Work

- Arun: researched and initially implemented displaying video from camera on VGA monitor; attempted ball generation code; helped with tweaking game; compiled design report, final report and presentation slides.

- David: developed initial prototype of algorithm; helped write all aspects of hardware; tried to use sram for storing downscaled frames; contributed to final report.

- Jason: developed ball generation in hardware; added sprites for on-screen score display; wrote software for control of game; wrote the majority of the VGA Raster; contributed to final report.

- Vincent: drafted original proposal; wrote VHDL code for downscaling, silhouette generation, and RAM; contributed to final report.

5.2 Lessons Learned

- To get the 27 MHz clock to appear on the input pin, TD_RESET must be set to '1'.

- Quartus only synthesizes 1 dimensional arrays as RAM.

- The VGA clock can be 27MHz.

5.3 Advice for Future Projects

- Remember to import pin assignments!

- Make reasonable milestones. We made the mistake of trying to implement the bulk of our project (the downscaling and silhouette generation algorithms) by the second milestone. Understanding the difficulty of the work and also your school schedules is integral in decided milestone goals.

- Understand exactly how each component works. Professor Edwards will warn you plenty of times. Hardware is not as forgiving as software. Since changing small bits of code and recompiling will take orders of magnitude longer for hardware, a hacker mentality will not work. Fully thinking through the design and understanding how each component interacts is more important that how quickly you start writing.

6 References

- `http://www.cs.columbia.edu/~sedwards/classes/2011/4840/Analog-Devices-ADV7181-video-decoder.pdf`

- `http://www.cs.columbia.edu/~sedwards/classes/2011/4840/ISSI-IS61LV25616-SRAM.pdf`

- `http://users.ece.gatech.edu/~hamblen/DE2/DE2_demo`

7 Source Code

7.1 data_transfer.vhd

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

entity data_transfer is

port (
clk50: in std_logic;

-- Avalon Bus Signals
    signal address : IN STD_LOGIC_VECTOR (15 DOWNTO 0);
    signal chipselect : IN STD_LOGIC;
    signal read : IN STD_LOGIC;
    signal reset_n : IN STD_LOGIC;
    signal write : IN STD_LOGIC;
    signal writedata : IN unsigned (31 DOWNTO 0);
    signal readdata : OUT unsigned (31 DOWNTO 0);

h : out std_logic;
ch      : out unsigned(11 downto 0);
cv      : out unsigned(11 downto 0);
r       : out unsigned(5 downto 0)
);
end data_transfer;

architecture rtl of data_transfer is
signal hit : std_logic;
signal circleH : unsigned(11 downto 0);
signal circleV : unsigned(11 downto 0);
signal radius  : unsigned(5 downto 0);


begin
SetPixel : process(clk50)
begin
if rising_edge(clk50) then
if chipselect = '1' then
if write = '1' then
--if address = "00000" then
--temp <= writedata;
hit <= writedata(30);
circleH <= writedata(23 downto 12);
circleV <= writedata(11 downto 0);
    radius <= writedata(29 downto 24);
```

```
--readdata <= "11111111111111111111111111111110";
--elsif address = "00001" then
--readdata <= "00000000000000000000000000000001";
--CIRCLE_HSTART <= writedata(31 downto 16);
--CIRCLE_VSTART <= writedata(15 downto 0);
--else if address(4 downto 0) = "00002" then

--end if;







--CIRCLE_HSTART <= writedata(31 downto 16);
--CIRCLE_VSTART <= writedata(15 downto 0);
--else if address(4 downto 0) = "00002" then

elsif read = '1' then
--if address = "00000" then
--readdata <= temp;
--end if;
end if;
end if;
end if;
end process SetPixel;

ch <= circleH;
cv <= circleV;
r  <= radius;
h  <= hit;
end rtl;
```

7.2 gnd.vhd

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

entity GND is

port (
-- Clocks
CLOCK_27,                                    -- 27 MHz
CLOCK_50 : in std_logic;                     -- 50 MHz

-- Buttons and switches
KEY : in std_logic_vector(3 downto 0);       -- Push buttons

-- LED displays
HEX0, HEX1, HEX2, HEX3                       -- 7-segment displays
: out std_logic_vector(6 downto 0);          -- (active low)

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

-- SRAM
SRAM_DQ : inout std_logic_vector(15 downto 0); -- Data bus 16 Bits
SRAM_ADDR : out std_logic_vector(17 downto 0); -- Address bus 18 Bits
SRAM_UB_N,                                     -- High-byte Data Mask
SRAM_LB_N,                                     -- Low-byte Data Mask
SRAM_WE_N,                                     -- Write Enable
SRAM_CE_N,                                     -- Chip Enable
SRAM_OE_N : out std_logic;                     -- Output Enable

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

-- Video Decoder
TD_DATA : in std_logic_vector(7 downto 0);   -- Data bus 8 bits
TD_HS,                                       -- H_SYNC
TD_VS : in std_logic;                        -- V_SYNC
TD_RESET : out std_logic                     -- Reset
);

end GND;

architecture arch of GND is

component I2C_AV_Config port(
iCLK : in std_logic;
iRST_N : in std_logic;
I2C_SCLK : out std_logic;
I2C_SDAT : inout std_logic
);
end component;

component TV_to_VGA port (
OSC_27 : in std_logic;
RESET : in std_logic;
VGA_BLANK : out std_logic;
VGA_SYNC : out std_logic;
VGA_CLOCK : out std_logic;
VGA_HS : out std_logic;
VGA_VS : out std_logic;
x640 : out unsigned(9 downto 0);
y480 : out unsigned(8 downto 0);
Y : out unsigned(7 downto 0);
Cb : out unsigned(7 downto 0);
Cr : out unsigned(7 downto 0);
TD_D : in std_logic_vector(7 downto 0);
TD_HS : in std_logic;
TD_VS : in std_logic
);
end component;

-- DECODER output signals
signal vga_blank_sig : std_logic;
signal vga_sync_sig : std_logic;
signal vga_clock_sig : std_logic;
signal vga_hs_sig : std_logic;
signal vga_vs_sig : std_logic;

signal x640 : unsigned(9 downto 0);
signal y480 : unsigned(8 downto 0);
signal Y : unsigned(7 downto 0);
signal Cb : unsigned(7 downto 0);
signal Cr : unsigned(7 downto 0);

-- DOWNSCALER output signals
```

```
signal x80 : unsigned(6 downto 0);
signal y60 : unsigned(5 downto 0);
signal data : unsigned(23 downto 0);

-- BACKGROUND_RAM output signal
signal bg : unsigned(23 downto 0);

-- FOREGROUND_RAM output signal
signal fg : unsigned(23 downto 0);

-- SILHOUETTE_GENERATOR output signals
signal rx : unsigned(6 downto 0);
signal ry : unsigned(5 downto 0);
signal sil_x : unsigned(4 downto 0);
signal sil_y : unsigned(3 downto 0);
signal sil : std_logic;

-- signals for SRAM
signal h  : std_logic;
signal ch : std_logic_vector(11 downto 0);
signal cv : std_logic_vector(11 downto 0);
signal r  : std_logic_vector(5 downto 0);

begin

TD_RESET <= '1'; -- necessary for CLOCK_27

I2C : I2C_AV_Config port map ( -- necessary for TV Decoder to get video data
iCLK => CLOCK_50,
iRST_N => '1',
I2C_SCLK => I2C_SCLK,
I2C_SDAT => I2C_SDAT
);

DECODER : TV_to_VGA port map (
OSC_27 => CLOCK_27, -- must be clocked at 27 MHz
RESET => '0',
VGA_BLANK => vga_blank_sig,
VGA_SYNC => vga_sync_sig,
VGA_CLOCK => vga_clock_sig,
VGA_HS => vga_hs_sig,
VGA_VS => vga_vs_sig,
x640 => x640,
y480 => y480,
Y => Y,
Cb => Cb,
Cr => Cr,
TD_D => TD_DATA,
TD_HS => TD_HS,
TD_VS => TD_VS
);

DOWNSCALER : entity work.downscaler port map (
clk => vga_clock_sig,
blank => vga_blank_sig,
sync => vga_sync_sig,
hs => vga_hs_sig,
vs => vga_vs_sig,
x640 => x640,
y480 => y480,
Y => Y,
Cb => Cb,
Cr => Cr,
x80 => x80,
y60 => y60,
YCbCr => data
);

BACKGROUND_RAM : entity work.ram_img port map (
wclk => CLOCK_50,
rclk => CLOCK_50,
we => not KEY(0),
wx => x80,
wy => y60,
di => data,
rx => rx,
ry => ry,
do => bg
);

FOREGROUND_RAM : entity work.ram_img port map (
wclk => CLOCK_50,
rclk => CLOCK_50,
we => '1',
wx => x80,
wy => y60,
di => data,
rx => rx,
ry => ry,
do => fg
);

SILHOUETTE_GENERATOR : entity work.silhouette_generator port map (
clk => CLOCK_27, -- must be clocked at 27 MHz to be in sync with TV_to_VGA and VGA_RASTER
```

```
fg => fg,
bg => bg,
rx => rx,
ry => ry,
xo => sil_x,
yo => sil_y,
do => sil
);

VGA_RASTER : entity work.vga_raster port map (
reset => '0',
clk => CLOCK_27, -- must be 27 MHz to be in sync with TV_to_VGA (25 MHz does not work well)
x => sil_x,
y => sil_y,
sil => sil,

ch_in => unsigned(ch),
cv_in => unsigned(cv),
h_in => h,
r_in => unsigned(r),

goalie => not KEY(2),
dodge => not KEY(1),
ninja => not KEY(3),
HEX0 => HEX0,
HEX1 => HEX1,
HEX2 => HEX2,
HEX3 => HEX3,

VGA_CLK => VGA_CLK,
VGA_HS => VGA_HS,
VGA_VS => VGA_VS,
VGA_BLANK => VGA_BLANK,
VGA_SYNC => VGA_SYNC,
VGA_R => VGA_R,
VGA_G => VGA_G,
VGA_B => VGA_B
);

nios: entity work.nios_system port map(
SRAM_ADDR_from_the_sram => SRAM_ADDR,
SRAM_CE_N_from_the_sram => SRAM_CE_N,
SRAM_DQ_to_and_from_the_sram => SRAM_DQ,
SRAM_LB_N_from_the_sram => SRAM_LB_N,
SRAM_OE_N_from_the_sram => SRAM_OE_N,
SRAM_UB_N_from_the_sram => SRAM_UB_N,
SRAM_WE_N_from_the_sram => SRAM_WE_N,
ch_from_the_transfer => ch,
cv_from_the_transfer => cv,
h_from_the_transfer => h,
r_from_the_transfer => r,
clk => CLOCK_50,
reset_n => '1'
);

end arch;
```

7.3 downscaler.vhd

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

entity downscaler is

port (
clk : in std_logic;
blank : in std_logic;
sync : in std_logic;
hs : in std_logic;
vs : in std_logic;
x640 : in unsigned(9 downto 0);
y480 : in unsigned(8 downto 0);
Y : in unsigned(7 downto 0);
Cb : in unsigned(7 downto 0);
Cr : in unsigned(7 downto 0);
x80 : out unsigned(6 downto 0);
y60 : out unsigned(5 downto 0);
YCbCr : out unsigned(23 downto 0)
);

end downscaler;

architecture arch of downscaler is

type buf_type14 is array(0 to 79) of unsigned(13 downto 0);
signal Y_buf : buf_type14 := (others => (others => '0'));
signal Cb_buf : buf_type14 := (others => (others => '0'));
signal Cr_buf : buf_type14 := (others => (others => '0'));

type buf_type24 is array(0 to 79) of unsigned(23 downto 0);
signal YCbCr_buf : buf_type24 := (others => (others => '0'));

begin

-- downscale a line
process (clk)
begin
if rising_edge(clk) then
--if hs = '0' and vs = '0' then
if x640(2 downto 0) = "000" and y480(2 downto 0) = "000" then
Y_buf(to_integer(x640(9 downto 3))) <= "000000" & Y;
Cb_buf(to_integer(x640(9 downto 3))) <= "000000" & Cb;
Cr_buf(to_integer(x640(9 downto 3))) <= "000000" & Cr;
elsif x640(2 downto 0) = "111" and y480(2 downto 0) = "111" then
```

```
YCbCr_buf(to_integer(x640(9 downto 3))) <=
(Y_buf(to_integer(x640(9 downto 3)))(13 downto 6) + Y(7 downto 6))
& (Cb_buf(to_integer(x640(9 downto 3)))(13 downto 6) + Cb(7 downto 6))
& (Cr_buf(to_integer(x640(9 downto 3)))(13 downto 6) + Cr(7 downto 6));
-- averaging not perfect, but close
else
Y_buf(to_integer(x640(9 downto 3))) <= Y_buf(to_integer(x640(9 downto 3))) + Y;
Cb_buf(to_integer(x640(9 downto 3))) <= Cb_buf(to_integer(x640(9 downto 3))) + Cb;
Cr_buf(to_integer(x640(9 downto 3))) <= Cr_buf(to_integer(x640(9 downto 3))) + Cr;
end if;

-- outputs
x80 <= to_unsigned(to_integer(y480(2 downto 0))*10 + to_integer(x640(9 downto 6)), 7);
if y480 > 7 then
y60 <= y480(8 downto 3) - 1;
else
y60 <= "111011"; -- 59
end if;
YCbCr <= YCbCr_buf(to_integer(y480(2 downto 0))*10 + to_integer(x640(9 downto 6)));
--end if;
end if;
end process;

end arch;
```

7.4 hello_world.c

```
#include <io.h>
#include <system.h>
#include <stdio.h>
#include <time.h>
/*
#define IOWR_LED_DATA(base, offset, data) \
    IOWR_16DIRECT(base, (offset) * 2, data)
#define IORD_LED_DATA(base, offset) \
    IORD_16DIRECT(base, (offset) * 2)
#define IOWR_LED_SPEED(base, data) \
    IOWR_16DIRECT(base + 32, 0, data)
*/
#define IOWR_TRANSFER_DATA(base, offset, data) \
    IOWR_32DIRECT(base, (offset)*4, data);
#define IORD_TRANSFER_DATA(base, offset) \
    IORD_32DIRECT(base, (offset) * 4);

int main()
{
    short vertical;
    short horizontal;
    short radius;
    short hit; /*determine whether to hit or dodge the ball*/
    unsigned int message, count, count2, speed, reseed;
    int message2;
    int x,y;
    int direction;

    int unsigned temp, temp2;

    vertical = 240;
    horizontal = 320;
    srand((unsigned)time(NULL));
    count2 = 0;
    radius = 1;
    reseed = 0;

    direction = rand() %17;
    speed = rand()%4 + 4; //select from 20 to 24
    hit = rand()%2; //select 0 or 1


    if (direction == 0) {
        x = 0;
        y = 0;
    }
```

```
if (direction == 1) {
    x = 1;
    y = 0;
}
if (direction == 2) {
    x = -1;
    y = 0;
}
if (direction == 3) {
    x = 0;
    y = 1;
}
if (direction == 4) {
    x = -1;
    y = 1;
}
if (direction == 5) {
    x = 1;
    y = 1;
}
if (direction == 6) {
    x = -1;
    y = 1;
}
if (direction == 7) {
    x = -1;
    y = -1;
}
if (direction == 8) {
    x = 1;
    y = -1;
}
if (direction == 9) {
    x = 1;
    y = 2;
}
if (direction == 10) {
    x = 2;
    y = 1;
}
if (direction == 11) {
    x = -2;
    y = 1;
}
if (direction == 12) {
    x = -1;
    y = 2;
```

```
}
if (direction == 13) {
    x = 2;
    y = -1;
}
if (direction == 14) {
    x = 1;
    y = -2;
}
if (direction == 15) {
    x = -2;
    y = -1;
}
if (direction == 16) {
    x = -1;
    y = -2;
}



for(;;) {
    count = 0;
    if (reseed == 1) {
        direction = rand() %17;
        speed = rand()%4 + 4; //select from 20 to 24
        vertical = 240;
        horizontal = 320;

        if (direction == 0) {
            x = 0;
            y = 0;
        }
        if (direction == 1) {
            x = 1;
            y = 0;
        }
        if (direction == 2) {
            x = -1;
            y = 0;
        }
        if (direction == 3) {
            x = 0;
            y = 1;
        }
        if (direction == 4) {
            x = -1;
            y = 1;
```

```
}
if (direction == 5) {
    x = 1;
    y = 1;
}
if (direction == 6) {
    x = -1;
    y = 1;
}
if (direction == 7) {
    x = -1;
    y = -1;
}
if (direction == 8) {
    x = 1;
    y = -1;
}
if (direction == 9) {
    x = 1;
    y = 2;
}
if (direction == 10) {
    x = 2;
    y = 1;
}
if (direction == 11) {
    x = -2;
    y = 1;
}
if (direction == 12) {
    x = -1;
    y = 2;
}
if (direction == 13) {
    x = 2;
    y = -1;
}
if (direction == 14) {
    x = 1;
    y = -2;
}
if (direction == 15) {
     x = -2;
     y = -1;
}
if (direction == 16) {
    x = -1;
```

```
         y = -2;
     }
     reseed = 0;
 }


 while (count < 5000) {
     count++;
 }
 if (horizontal > 640 - radius) {
     x = -1;
 }
 if (horizontal < radius) {
     x = 1;
 }

 if (vertical > 480 - radius) {
     y = -1;
 }
 if (vertical < radius) {
     y = 1;
 }

if (count2 > speed) {
     if (radius > 50) {
         radius = 1;
         hit = rand() %2;
         reseed = 1;
     }
     else {
         if (radius < 10) {
             radius++;
         }
         else if (radius < 20) {
             radius = radius + 2;
         }
         else if (radius < 30) {
             radius = radius + 4;
         }
         else if (radius < 40) {
             radius = radius + 8;
         }
         else
             radius = radius + 10;
     }
     count2 = 0;
}
```

```
    count2++;
    vertical = vertical + y;
    horizontal = horizontal + x;
    int temp;
    unsigned int temp2;
    temp2 = hit << 30;
    message = 0;
    message = radius << 24;
    temp = horizontal << 12;
    message = message + temp + temp2;
    message = message + vertical;
    printf("This is %u\n", temp2);
    message2 = 100;

    //printf("%d\n", temp3);

   //message = rand() %10;
//message2 = radius;

int i;
//i = 0;
//for (i = 0; i < 100; i ++) {
//printf("HELLO");
  IOWR_TRANSFER_DATA(TRANSFER_BASE, 0, message);
  i = IORD_TRANSFER_DATA(TRANSFER_BASE, 0);
   // printf("%d", i);
    //printf("\n");
//}

//IOWR_VGA_DATA(VGA_BASE, 0, message2);


//temp = IORD_32DIRECT(VGA_BASE, 0);
//temp2 = IORD_32DIRECT(VGA_BASE, 4);

//printf("This is %d", temp);
//printf("Second is %d", temp2);
    //IOWR_32DIRECT(VGA_BASE, 0, message);
//IOWR_32DIRECT(VGA_BASE, 2, message2);

    //printf("This is %d\n", message);
    //printf("HELLO %d\n", hit);
    //printf("get back %d\n", i);
}


   // printf("Hello Michael\n");
```

```
    return 0;
}
```

7.5 ram_img.vhd

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

entity ram_img is

port (
wclk : in std_logic;
rclk : in std_logic;
we : in std_logic;
wx : in unsigned(6 downto 0);
wy : in unsigned(5 downto 0);
di : in unsigned(23 downto 0);
rx : in unsigned(6 downto 0);
ry : in unsigned(5 downto 0);
do : out unsigned(23 downto 0)
);

end ram_img;

architecture rtl of ram_img is
type ram_type is array(0 to 4799) of unsigned(23 downto 0);
signal RAM : ram_type := (others => (others => '0'));

begin

process (wclk)
begin
if rising_edge(wclk) then
if we = '1' then
RAM(to_integer(wy)*80 + to_integer(wx)) <= di;
end if;
end if;
end process;

process (rclk)
begin
if rising_edge(rclk) then
do <= RAM(to_integer(ry)*80 + to_integer(rx));
end if;
end process;

end rtl;
```

7.6 silhouette_generator.vhd

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

entity silhouette_generator is

port (
clk : in std_logic;
fg : in unsigned(23 downto 0);
bg : in unsigned(23 downto 0);
rx : out unsigned(6 downto 0);
ry : out unsigned(5 downto 0);
xo : out unsigned(4 downto 0);
yo : out unsigned(3 downto 0);
do : out std_logic
);

end silhouette_generator;

architecture arch of silhouette_generator is

constant Y_TOL : integer := 127;
constant Cb_TOL : integer := 2;
constant Cr_TOL : integer := 2;

signal x80 : unsigned(6 downto 0);
signal y60 : unsigned(5 downto 0);

type buf_type5 is array(0 to 19) of unsigned(4 downto 0);
signal count : buf_type5 := (others => (others => '0'));

signal sil_buf : std_logic_vector(0 to 19) := (others => '0');

begin

XYCounter : process (clk)
begin
if rising_edge(clk) then
if x80 < 79 then
x80 <= x80 + 1;
else
x80 <= (others => '0');
if y60 < 59 then
y60 <= y60 + 1;
else
y60 <= (others => '0');
```

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

process (clk)
begin
if rising_edge(clk) then
----------------------------------------------------------------------------------------------------
if x80(1 downto 0) = "00" and y60(1 downto 0) = "00" then

if fg(23 downto 16) > bg(23 downto 16) then
if fg(23 downto 16) - bg(23 downto 16) > Y_TOL then
count(to_integer(x80(6 downto 2))) <= "00001";
else
if fg(15 downto 8) > bg(15 downto 8) then
if fg(15 downto 8) - bg(15 downto 8) > Cb_TOL then
count(to_integer(x80(6 downto 2))) <= "00001";
else
if fg(7 downto 0) > bg(7 downto 0) then
if fg(7 downto 0) - bg(7 downto 0) > Cr_TOL then
count(to_integer(x80(6 downto 2))) <= "00001";
else
count(to_integer(x80(6 downto 2))) <= (others => '0');
end if;
else
if bg(7 downto 0) - fg(7 downto 0) > Cr_TOL then
count(to_integer(x80(6 downto 2))) <= "00001";
else
count(to_integer(x80(6 downto 2))) <= (others => '0');
end if;
end if;
end if;
else
if bg(15 downto 8) - fg(15 downto 8) > Cb_TOL then
count(to_integer(x80(6 downto 2))) <= "00001";
else
if fg(7 downto 0) > bg(7 downto 0) then
if fg(7 downto 0) - bg(7 downto 0) > Cr_TOL then
count(to_integer(x80(6 downto 2))) <= "00001";
else
count(to_integer(x80(6 downto 2))) <= (others => '0');
end if;
else
if bg(7 downto 0) - fg(7 downto 0) > Cr_TOL then
count(to_integer(x80(6 downto 2))) <= "00001";
else
```

```
count(to_integer(x80(6 downto 2))) <= (others => ’0’);
end if;
end if;
end if;
end if;
end if;
else
if bg(23 downto 16) - fg(23 downto 16) > Y_TOL then
count(to_integer(x80(6 downto 2))) <= "00001";
else
if fg(15 downto 8) > bg(15 downto 8) then
if fg(15 downto 8) - bg(15 downto 8) > Cb_TOL then
count(to_integer(x80(6 downto 2))) <= "00001";
else
if fg(7 downto 0) > bg(7 downto 0) then
if fg(7 downto 0) - bg(7 downto 0) > Cr_TOL then
count(to_integer(x80(6 downto 2))) <= "00001";
else
count(to_integer(x80(6 downto 2))) <= (others => ’0’);
end if;
else
if bg(7 downto 0) - fg(7 downto 0) > Cr_TOL then
count(to_integer(x80(6 downto 2))) <= "00001";
else
count(to_integer(x80(6 downto 2))) <= (others => ’0’);
end if;
end if;
end if;
else
if bg(15 downto 8) - fg(15 downto 8) > Cb_TOL then
count(to_integer(x80(6 downto 2))) <= "00001";
else
if fg(7 downto 0) > bg(7 downto 0) then
if fg(7 downto 0) - bg(7 downto 0) > Cr_TOL then
count(to_integer(x80(6 downto 2))) <= "00001";
else
count(to_integer(x80(6 downto 2))) <= (others => ’0’);
end if;
else
if bg(7 downto 0) - fg(7 downto 0) > Cr_TOL then
count(to_integer(x80(6 downto 2))) <= "00001";
else
count(to_integer(x80(6 downto 2))) <= (others => ’0’);
end if;
end if;
end if;
end if;
end if;
end if;
------------------------------------------------------------------------------------------------------------
elsif x80(1 downto 0) = "11" and y60(1 downto 0) = "11" then

if count(to_integer(x80(6 downto 2))) < 8 then
sil_buf(to_integer(x80(6 downto 2))) <= '0';
elsif count(to_integer(x80(6 downto 2))) > 8 then
sil_buf(to_integer(x80(6 downto 2))) <= '1';
-- borderline case; must evaluate 16th square
elsif fg(23 downto 16) > bg(23 downto 16) then
if fg(23 downto 16) - bg(23 downto 16) > Y_TOL then
sil_buf(to_integer(x80(6 downto 2))) <= '1';
else
if fg(15 downto 8) > bg(15 downto 8) then
if fg(15 downto 8) - bg(15 downto 8) > Cb_TOL then
sil_buf(to_integer(x80(6 downto 2))) <= '1';
else
if fg(7 downto 0) > bg(7 downto 0) then
if fg(7 downto 0) - bg(7 downto 0) > Cr_TOL then
sil_buf(to_integer(x80(6 downto 2))) <= '1';
else
sil_buf(to_integer(x80(6 downto 2))) <= '0';
end if;
else
if bg(7 downto 0) - fg(7 downto 0) > Cr_TOL then
sil_buf(to_integer(x80(6 downto 2))) <= '1';
else
sil_buf(to_integer(x80(6 downto 2))) <= '0';
end if;
end if;
end if;
else
if bg(15 downto 8) - fg(15 downto 8) > Cb_TOL then
sil_buf(to_integer(x80(6 downto 2))) <= '1';
else
if fg(7 downto 0) > bg(7 downto 0) then
if fg(7 downto 0) - bg(7 downto 0) > Cr_TOL then
sil_buf(to_integer(x80(6 downto 2))) <= '1';
else
sil_buf(to_integer(x80(6 downto 2))) <= '0';
end if;
else
if bg(7 downto 0) - fg(7 downto 0) > Cr_TOL then
sil_buf(to_integer(x80(6 downto 2))) <= '1';
else
sil_buf(to_integer(x80(6 downto 2))) <= '0';
```

```
end if;
end if;
end if;
end if;
end if;
else
if bg(23 downto 16) - fg(23 downto 16) > Y_TOL then
sil_buf(to_integer(x80(6 downto 2))) <= '1';
else
if fg(15 downto 8) > bg(15 downto 8) then
if fg(15 downto 8) - bg(15 downto 8) > Cb_TOL then
sil_buf(to_integer(x80(6 downto 2))) <= '1';
else
if fg(7 downto 0) > bg(7 downto 0) then
if fg(7 downto 0) - bg(7 downto 0) > Cr_TOL then
sil_buf(to_integer(x80(6 downto 2))) <= '1';
else
sil_buf(to_integer(x80(6 downto 2))) <= '0';
end if;
else
if bg(7 downto 0) - fg(7 downto 0) > Cr_TOL then
sil_buf(to_integer(x80(6 downto 2))) <= '1';
else
sil_buf(to_integer(x80(6 downto 2))) <= '0';
end if;
end if;
end if;
else
if bg(15 downto 8) - fg(15 downto 8) > Cb_TOL then
sil_buf(to_integer(x80(6 downto 2))) <= '1';
else
if fg(7 downto 0) > bg(7 downto 0) then
if fg(7 downto 0) - bg(7 downto 0) > Cr_TOL then
sil_buf(to_integer(x80(6 downto 2))) <= '1';
else
sil_buf(to_integer(x80(6 downto 2))) <= '0';
end if;
else
if bg(7 downto 0) - fg(7 downto 0) > Cr_TOL then
sil_buf(to_integer(x80(6 downto 2))) <= '1';
else
sil_buf(to_integer(x80(6 downto 2))) <= '0';
end if;
end if;
end if;
end if;
end if;
end if;

----------------------------------------------------------------------------------------------------
else

if fg(23 downto 16) > bg(23 downto 16) then
if fg(23 downto 16) - bg(23 downto 16) > Y_TOL then
count(to_integer(x80(6 downto 2))) <= count(to_integer(x80(6 downto 2))) + 1;
else
if fg(15 downto 8) > bg(15 downto 8) then
if fg(15 downto 8) - bg(15 downto 8) > Cb_TOL then
count(to_integer(x80(6 downto 2))) <= count(to_integer(x80(6 downto 2))) + 1;
else
if fg(7 downto 0) > bg(7 downto 0) then
if fg(7 downto 0) - bg(7 downto 0) > Cr_TOL then
count(to_integer(x80(6 downto 2))) <= count(to_integer(x80(6 downto 2))) + 1;
end if;
else
if bg(7 downto 0) - fg(7 downto 0) > Cr_TOL then
count(to_integer(x80(6 downto 2))) <= count(to_integer(x80(6 downto 2))) + 1;
end if;
end if;
end if;
else
if bg(15 downto 8) - fg(15 downto 8) > Cb_TOL then
count(to_integer(x80(6 downto 2))) <= count(to_integer(x80(6 downto 2))) + 1;
else
if fg(7 downto 0) > bg(7 downto 0) then
if fg(7 downto 0) - bg(7 downto 0) > Cr_TOL then
count(to_integer(x80(6 downto 2))) <= count(to_integer(x80(6 downto 2))) + 1;
end if;
else
if bg(7 downto 0) - fg(7 downto 0) > Cr_TOL then
count(to_integer(x80(6 downto 2))) <= count(to_integer(x80(6 downto 2))) + 1;
end if;
end if;
end if;
end if;
end if;
else
if bg(23 downto 16) - fg(23 downto 16) > Y_TOL then
count(to_integer(x80(6 downto 2))) <= count(to_integer(x80(6 downto 2))) + 1;
else
if fg(15 downto 8) > bg(15 downto 8) then
if fg(15 downto 8) - bg(15 downto 8) > Cb_TOL then
count(to_integer(x80(6 downto 2))) <= count(to_integer(x80(6 downto 2))) + 1;
else
```

```
if fg(7 downto 0) > bg(7 downto 0) then
if fg(7 downto 0) - bg(7 downto 0) > Cr_TOL then
count(to_integer(x80(6 downto 2))) <= count(to_integer(x80(6 downto 2))) + 1;
end if;
else
if bg(7 downto 0) - fg(7 downto 0) > Cr_TOL then
count(to_integer(x80(6 downto 2))) <= count(to_integer(x80(6 downto 2))) + 1;
end if;
end if;
end if;
else
if bg(15 downto 8) - fg(15 downto 8) > Cb_TOL then
count(to_integer(x80(6 downto 2))) <= count(to_integer(x80(6 downto 2))) + 1;
else
if fg(7 downto 0) > bg(7 downto 0) then
if fg(7 downto 0) - bg(7 downto 0) > Cr_TOL then
count(to_integer(x80(6 downto 2))) <= count(to_integer(x80(6 downto 2))) + 1;
end if;
else
if bg(7 downto 0) - fg(7 downto 0) > Cr_TOL then
count(to_integer(x80(6 downto 2))) <= count(to_integer(x80(6 downto 2))) + 1;
end if;
end if;
end if;
end if;
end if;
end if;

end if;

-- outputs
xo <= to_unsigned(to_integer(y60(1 downto 0))*5 + to_integer(x80(6 downto 4)), 5);
if y60 > 3 then
yo <= y60(5 downto 2) - 1;
else
yo <= "1110"; -- 14
end if;
do <= sil_buf(to_integer(y60(1 downto 0))*5 + to_integer(x80(6 downto 4)));

end if;
end process;

rx <= x80; -- not sure why this works
ry <= y60; -- thought that rx and ry needed to be 1 cycle ahead of x80 and y60

end arch;
```

7.7 TV_to_VGA.v

```
module TV_to_VGA (
OSC_27,
RESET,
  VGA_BLANK,
VGA_SYNC,
VGA_CLOCK,
VGA_HS,
  VGA_VS,
x640,// Modified by VL
y480,// Modified by VL
// VGA_R,// Modified by VL
//  VGA_G,// Modified by VL
//  VGA_B,// Modified by VL
Y,// Modified by VL
Cb,// Modified by VL
Cr,// Modified by VL
  TD_D,
  TD_HS,
  TD_VS
);
input     OSC_27;
input RESET;
output  VGA_BLANK;
output VGA_SYNC;
output VGA_CLOCK;
output VGA_HS;
output  VGA_VS;
output  [9:0]x640; // Modified by VL
output  [8:0]y480; // Modified by VL
//output [9:0]VGA_R; // Modified by VL
//output  [9:0]VGA_G; // Modified by VL
//output  [9:0]VGA_B; // Modified by VL
input  [7:0]TD_D;
input  TD_HS;
input  TD_VS;


output [7:0] Y;       //4:4:4 Y // Modified by VL
output [7:0] Cb;      //4:4:4 Cb // Modified by VL
output [7:0] Cr;      //4:4:4 Cr // Modified by VL
wire mTD_HSx2;

itu_r656_decoder U1
(
.CLOCK(OSC_27),//system clock
.TD_D(TD_D[7:0]),//4:2:2 video data stream
```

```
  .TD_HS(TD_HS),//Decoder_hs
.TD_VS(TD_VS),//Decoder_vs
.Y(Y[7:0]),        //4:4:4 Y
.Cb(Cb[7:0]),       //4:4:4 Cb
.Cr(Cr[7:0]),       //4:4:4 Cr
    .HSx2(mTD_HSx2),
.blank(VGA_BLANK)
);

//YCbCr2RGB U2( // Modified by VL
// .Red(VGA_R[9:0]),// Modified by VL
// .Green(VGA_G[9:0]),// Modified by VL
// .Blue(VGA_B[9:0]),// Modified by VL
// .iY(Y[7:0]),// Modified by VL
// .iCb(Cb[7:0]),// Modified by VL
// .iCr(Cr[7:0]),// Modified by VL
// .iRESET(!RESET),// Modified by VL
// .iCLK(OSC_27) // Modified by VL
// ); // Modified by VL

‘include "VGA_Param.h"
reg  [10:0]L_COUNTER;//<<
reg  [10:0]RL_COUNTER;//<<
wire  sync_reset=(RL_COUNTER==9)?1:0;//<<
reg   sync_en;//<<
reg   [7:0]delay;//<<

reg [9:0] H_Cont;
reg [9:0] V_Cont;
reg [9:0] X_Count; // Modified by VL
reg [8:0] Y_Count; // Modified by VL
reg oVGA_H_SYNC;
reg oVGA_V_SYNC;
reg Pre_HS;
reg Pre_VS;
reg mACT_HS;
reg mACT_VS;

always@(posedge OSC_27 or negedge sync_en)//<<
begin
if(!sync_en)//<<
begin
Pre_HS <= 0;
mACT_HS <= 0;
H_Cont <= 0;
oVGA_H_SYNC <= 0;
X_Count <= 0; // Modified by VL
```

```
end
else
begin
Pre_HS <= mTD_HSx2;
if({Pre_HS,mTD_HSx2}==2'b10)
mACT_HS <= 1;
if(mACT_HS)
begin
// H_Sync Counter
if( H_Cont < 852 )
H_Cont <= H_Cont+1;
else
begin
H_Cont <= 0;
mACT_HS <= 0;
end
// H_Sync Generator
if( H_Cont < H_SYNC_CYC )
oVGA_H_SYNC <= 0;
else
oVGA_H_SYNC <= 1;
// X_Count Generator // Modified by VL
if( H_Cont > X_START && X_Count < 639 ) // Modified by VL
X_Count <= X_Count+1; // Modified by VL
else // Modified by VL
X_Count <= 0; // Modified by VL
end
else
begin
oVGA_H_SYNC <= 0;
H_Cont <= 0;
end
end
end

always@(posedge OSC_27 or negedge sync_en)//<<
begin
if(!sync_en)//<<
begin
Pre_VS <= 1;
mACT_VS <= 0;
V_Cont <= 0;
oVGA_V_SYNC <= 0;
Y_Count <= 0; // Modified by VL
end
else
begin
```

```
Pre_VS <= TD_VS;
if({Pre_VS,TD_VS}==2'b01)
mACT_VS <= 1;
if( (H_Cont==1) && mACT_VS)
begin
// V_Sync Counter
if( V_Cont < 524 )
V_Cont <= V_Cont+1;
else
V_Cont <= 0;
// V_Sync Generator
if( V_Cont < V_SYNC_CYC )
oVGA_V_SYNC <= 0;
else
oVGA_V_SYNC <= 1;
// Y_Count Generator // Modified by VL
if( V_Cont > Y_START && Y_Count < 479 ) // Modified by VL
Y_Count <= Y_Count+1; // Modified by VL
else // Modified by VL
Y_Count <= 0; // Modified by VL
end
end
end

assign x640 = X_Count; // Modified by VL
assign y480 = Y_Count; // Modified by VL
assign VGA_HS = oVGA_H_SYNC;
assign VGA_VS = oVGA_V_SYNC;
assign VGA_SYNC = 1'b0;
assign VGA_CLOCK = OSC_27;


//>>lock detector
always @(posedge TD_HS) begin
if (TD_VS) L_COUNTER=0;
else L_COUNTER=L_COUNTER+1;
end

always @(posedge TD_VS) begin
RL_COUNTER=L_COUNTER;//1714
end
always@(negedge sync_reset or posedge TD_VS) begin
if (!sync_reset)
delay=0;
    else if (delay < 250)
 delay=delay+1;
end
```

```
always@(negedge sync_reset or negedge TD_VS) begin
if (!sync_reset)
sync_en=0;
else if (delay < 100)
    sync_en=0;
    else
sync_en=1;
end
//<<

endmodule
```

7.8 vga_raster.vhd

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

entity vga_raster is

  port (
    reset : in std_logic;
    clk : in std_logic;
x : in unsigned(4 downto 0);
y : in unsigned(3 downto 0);
    sil : in std_logic;
ch_in :in unsigned(11 downto 0);
cv_in :in unsigned(11 downto 0);
h_in :std_logic;
r_in :in unsigned(5 downto 0);
dodge  :in std_logic;
goalie :in std_logic;
ninja  :in std_logic;

HEX3,
HEX2,
HEX1,
HEX0 : out std_logic_vector( 6 downto 0);
    VGA_CLK,                         -- Clock
    VGA_HS,                          -- H_SYNC
    VGA_VS,                          -- V_SYNC
    VGA_BLANK,                       -- BLANK
    VGA_SYNC : out std_logic;        -- SYNC
    VGA_R,                           -- Red[9:0]
    VGA_G,                           -- Green[9:0]
    VGA_B : out unsigned(9 downto 0) -- Blue[9:0]
    );

end vga_raster;

architecture rtl of vga_raster is
```

```
  -- Video parameters

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

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

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

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

signal Xcount : unsigned(9 downto 0);
signal Ycount : unsigned(8 downto 0);
signal silhouette : std_logic_vector(0 to 299) := (others => '0'); -- silhouette

type ram_type is array(15 downto 0) of unsigned(15 downto 0);
type array_t is array (15 downto 0) of unsigned(79 downto 0);
type array_t2 is array (15 downto 0) of unsigned(127 downto 0);
type array_t3 is array (15 downto 0) of unsigned(95 downto 0);
type array_temp is array (15 downto 0) of unsigned(15 downto 0);

signal CIRCLE_HSTART : unsigned(11 downto 0):= "000000010000";
signal CIRCLE_VSTART : unsigned(11 downto 0):= "000000010000";
signal CIRCLE_RADIUS : unsigned(5 downto 0) := "010000";
signal CIRCLE_HIT: std_logic := '1';
signal SPRITE_SCORE: array_t;
signal SPRITE_LOSE: array_t2;
signal SPRITE_HSCORE: array_t3;
signal SPRITE0: array_temp;
signal SPRITE1: array_temp;
signal SPRITE2: array_temp;
signal SPRITE3: array_temp;
signal SPRITE4: array_temp;
signal SPRITE5: array_temp;
signal SPRITE6: array_temp;
signal SPRITE7: array_temp;
signal SPRITE8: array_temp;
signal SPRITE9: array_temp;

signal SCORE1 : unsigned(3 downto 0) := "0000"; --keeps track of the score of the current game
signal SCORE2 : unsigned(3 downto 0) := "0000"; --1 is most significant and 4 is the least significar
signal SCORE3 : unsigned(3 downto 0) := "0000";
signal SCORE4 : unsigned(3 downto 0) := "0000";
signal HSCORE1 : unsigned(3 downto 0) := "0000"; --keeps track of the highest score for all games.
signal HSCORE2 : unsigned(3 downto 0) := "0000";
signal HSCORE3 : unsigned(3 downto 0) := "0000";
signal HSCORE4 : unsigned(3 downto 0) := "0000";

signal increment: std_logic;
signal nlose: std_logic := '1';
signal restart: std_logic := '0';
signal hit: std_logic := '0';

signal touched: std_logic := '0';
signal tempLose: std_logic := '0';
signal ninja_hit: std_logic := '0';

signal temp: unsigned(31 downto 0);
signal circle : std_logic;
signal circleEND : std_logic;



begin
  --Sprites for various messages in the game
  SPRITE_LOSE(0)  <= "0000000000000000000000000000000000000000000000000000000000000000000000000000000
  SPRITE_LOSE(1)  <= "0000000000000000000000000000000000000000000000000000000000000000000000000000000
  SPRITE_LOSE(2)  <= "0000000000000000000000000000000000000000000000000000000000000000000000000000000
  SPRITE_LOSE(3)  <= "0011110000111100000111111111100000011000000110000000000000000000000000111000000000
  SPRITE_LOSE(4)  <= "0001110000111000000111111111100000011000000110000000000000000000000000111000000000
  SPRITE_LOSE(5)  <= "0000111001110000000111000001100000011000000110000000000000000000000000111000000000
  SPRITE_LOSE(6)  <= "0000011111100000000111000001100000011000000110000000000000000000000000111000000000
  SPRITE_LOSE(7)  <= "0000001110000000000111000001100000011000000110000000000000000000000000111000000000
  SPRITE_LOSE(8)  <= "0000001110000000000111000001100000011000000110000000000000000000000000111000000000
  SPRITE_LOSE(9)  <= "0000001110000000000111000001100000011000000110000000000000000000000000111000000000
  SPRITE_LOSE(10) <= "0000001110000000000111000001100000011000000110000000000000000000000000111000000000
  SPRITE_LOSE(11) <= "0000001110000000000111111111100000011111111110000000000000000000000001111111111000
  SPRITE_LOSE(12) <= "0000001110000000000111111111100000011111111110000000000000000000000001111111111000
  SPRITE_LOSE(13) <= "0000000000000000000000000000000000000000000000000000000000000000000000000000000
  SPRITE_LOSE(14) <= "0000000000000000000000000000000000000000000000000000000000000000000000000000000
  SPRITE_LOSE(15) <= "0000000000000000000000000000000000000000000000000000000000000000000000000000000

  SPRITE_SCORE(0)  <= "000000000000000000000000000000000000000000000000000000000000000000000000000000
  SPRITE_SCORE(1)  <= "000000000000000000000000000000000000000000000000000000000000000000000000000000
  SPRITE_SCORE(2)  <= "000000000000000000000000000000000000000000000000000000000000000000000000000000
  SPRITE_SCORE(3)  <= "000011111111100000001111111110000000111111110000000111111111000000011111111110
  SPRITE_SCORE(4)  <= "000111111111100000011111111110000001111111111000000111111111100000011111111110
  SPRITE_SCORE(5)  <= "000110000000000000011100000000000001100000011000000110000001100000011000000000
  SPRITE_SCORE(6)  <= "000110000000000000011100000000000001100000011000000110000001100000011000000000
  SPRITE_SCORE(7)  <= "000111111111000000011100000000000001100000011000000111111111100000011111111110
  SPRITE_SCORE(8)  <= "000011111111100000011100000000000001100000011000000111111111000000011111111110
  SPRITE_SCORE(9)  <= "000000000001100000011100000000000001100000011000000110011000000000011000000000
  SPRITE_SCORE(10) <= "000000000001100000011100000000000001100000011000000110001100000000011000000000
  SPRITE_SCORE(11) <= "000111111111100000011111111110000001111111111000000110000110000000011111111110
  SPRITE_SCORE(12) <= "000111111111000000001111111110000000111111110000000110000011000000011111111110
  SPRITE_SCORE(13) <= "000000000000000000000000000000000000000000000000000000000000000000000000000000
  SPRITE_SCORE(14) <= "000000000000000000000000000000000000000000000000000000000000000000000000000000
  SPRITE_SCORE(15) <= "000000000000000000000000000000000000000000000000000000000000000000000000000000

  SPRITE_HSCORE(0)  <= "00000000000000000000000000000000000000000000000000000000000000000000000000000
  SPRITE_HSCORE(1)  <= "00000000000000000000000000000000000000000000000000000000000000000000000000000
  SPRITE_HSCORE(2)  <= "00000000000000000000000000000000000000000000000000000000000000000000000000000
  SPRITE_HSCORE(3)  <= "00011000000110000000111111111000000011111111100000001111111100000001111111110
  SPRITE_HSCORE(4)  <= "00011000000110000001111111111000000111111111100000011111111110000001111111111
  SPRITE_HSCORE(5)  <= "00011000000110000001100000000000000111000000000000011000000110000001100000011
  SPRITE_HSCORE(6)  <= "00011000000110000001100000000000000111000000000000011000000110000001100000011
  SPRITE_HSCORE(7)  <= "00011111111110000001111111110000000111000000000000011000000110000001111111111
  SPRITE_HSCORE(8)  <= "00011111111110000000111111111000000111000000000000011000000110000001111111110
  SPRITE_HSCORE(9)  <= "00011000000110000000000000011000000111000000000000011000000110000001100110000
  SPRITE_HSCORE(10) <= "00011000000110000000000000011000000111000000000000011000000110000001100011000
  SPRITE_HSCORE(11) <= "00011000000110000001111111111000000111111111100000011111111110000001100001100
  SPRITE_HSCORE(12) <= "00011000000110000001111111110000000011111111100000001111111100000001100000110
  SPRITE_HSCORE(13) <= "00000000000000000000000000000000000000000000000000000000000000000000000000000
  SPRITE_HSCORE(14) <= "00000000000000000000000000000000000000000000000000000000000000000000000000000
  SPRITE_HSCORE(15) <= "00000000000000000000000000000000000000000000000000000000000000000000000000000

  SPRITE0(0)  <= "0000000000000000";
  SPRITE0(1)  <= "0000000000000000";
  SPRITE0(2)  <= "0000000000000000";
  SPRITE0(3)  <= "0000111111110000";
  SPRITE0(4)  <= "0001111111111000";
  SPRITE0(5)  <= "0001100000011000";
  SPRITE0(6)  <= "0001100000011000";
  SPRITE0(7)  <= "0001100000011000";
  SPRITE0(8)  <= "0001100000011000";
  SPRITE0(9)  <= "0001100000011000";
  SPRITE0(10) <= "0001100000011000";
  SPRITE0(11) <= "0001111111111000";
  SPRITE0(12) <= "0000111111110000";
  SPRITE0(13) <= "0000000000000000";
```

```
SPRITE0(14) <= "0000000000000000";
SPRITE0(15) <= "0000000000000000";

SPRITE1(0)  <= "0000000000000000";
SPRITE1(1)  <= "0000000000000000";
SPRITE1(2)  <= "0000000000000000";
SPRITE1(3)  <= "0000000110000000";
SPRITE1(4)  <= "0000000110000000";
SPRITE1(5)  <= "0000000110000000";
SPRITE1(6)  <= "0000000110000000";
SPRITE1(7)  <= "0000000110000000";
SPRITE1(8)  <= "0000000110000000";
SPRITE1(9)  <= "0000000110000000";
SPRITE1(10) <= "0000000110000000";
SPRITE1(11) <= "0000000110000000";
SPRITE1(12) <= "0000000110000000";
SPRITE1(13) <= "0000000000000000";
SPRITE1(14) <= "0000000000000000";
SPRITE1(15) <= "0000000000000000";

SPRITE2(0)  <= "0000000000000000";
SPRITE2(1)  <= "0000000000000000";
SPRITE2(2)  <= "0000000000000000";
SPRITE2(3)  <= "0000111111110000";
SPRITE2(4)  <= "0001111111111000";
SPRITE2(5)  <= "0001110000111000";
SPRITE2(6)  <= "0000000000111000";
SPRITE2(7)  <= "0000000001110000";
SPRITE2(8)  <= "0000000011100000";
SPRITE2(9)  <= "0000000111000000";
SPRITE2(10) <= "0000001110000000";
SPRITE2(11) <= "0000111111111000";
SPRITE2(12) <= "0001111111111000";
SPRITE2(13) <= "0000000000000000";
SPRITE2(14) <= "0000000000000000";
SPRITE2(15) <= "0000000000000000";

SPRITE3(0)  <= "0000000000000000";
SPRITE3(1)  <= "0000000000000000";
SPRITE3(2)  <= "0000000000000000";
SPRITE3(3)  <= "0000111111110000";
SPRITE3(4)  <= "0001111111111000";
SPRITE3(5)  <= "0001100000011000";
SPRITE3(6)  <= "0000000000011000";
SPRITE3(7)  <= "0000000111110000";
SPRITE3(8)  <= "0000000111110000";
SPRITE3(9)  <= "0000000000011000";
```

```
SPRITE3(10) <= "0001100000011000";
SPRITE3(11) <= "0001111111111000";
SPRITE3(12) <= "0000111111110000";
SPRITE3(13) <= "0000000000000000";
SPRITE3(14) <= "0000000000000000";
SPRITE3(15) <= "0000000000000000";

SPRITE4(0)  <= "0000000000000000";
SPRITE4(1)  <= "0000000000000000";
SPRITE4(2)  <= "0000000000000000";
SPRITE4(3)  <= "0000000011111000";
SPRITE4(4)  <= "0000000111111000";
SPRITE4(5)  <= "0000001110011000";
SPRITE4(6)  <= "0000011100011000";
SPRITE4(7)  <= "0000111000011000";
SPRITE4(8)  <= "0001111111111100";
SPRITE4(9)  <= "0001111111111100";
SPRITE4(10) <= "0000000000011000";
SPRITE4(11) <= "0000000000011000";
SPRITE4(12) <= "0000000000011000";
SPRITE4(13) <= "0000000000000000";
SPRITE4(14) <= "0000000000000000";
SPRITE4(15) <= "0000000000000000";

SPRITE5( 0) <= "0000000000000000";
SPRITE5( 1) <= "0000000000000000";
SPRITE5( 2) <= "0000000000000000";
SPRITE5( 3) <= "0001111111111000";
SPRITE5( 4) <= "0001111111111000";
SPRITE5( 5) <= "0001100000000000";
SPRITE5( 6) <= "0001100000000000";
SPRITE5( 7) <= "0001111111110000";
SPRITE5( 8) <= "0001111111111000";
SPRITE5( 9) <= "0000000000011000";
SPRITE5(10) <= "0001100000011000";
SPRITE5(11) <= "0001111111111000";
SPRITE5(12) <= "0000111111110000";
SPRITE5(13) <= "0000000000000000";
SPRITE5(14) <= "0000000000000000";
SPRITE5(15) <= "0000000000000000";

SPRITE6( 0) <= "0000000000000000";
SPRITE6( 1) <= "0000000000000000";
SPRITE6( 2) <= "0000000000000000";
SPRITE6( 3) <= "0000111111110000";
SPRITE6( 4) <= "0001111111111000";
SPRITE6( 5) <= "0001100000011000";
```

```
SPRITE6( 6) <= "0001100000000000";
SPRITE6( 7) <= "0001111111110000";
SPRITE6( 8) <= "0001111111111000";
SPRITE6( 9) <= "0001100000011000";
SPRITE6(10) <= "0001100000011000";
SPRITE6(11) <= "0001111111111000";
SPRITE6(12) <= "0000111111110000";
SPRITE6(13) <= "0000000000000000";
SPRITE6(14) <= "0000000000000000";
SPRITE6(15) <= "0000000000000000";

SPRITE7( 0) <= "0000000000000000";
SPRITE7( 1) <= "0000000000000000";
SPRITE7( 2) <= "0000000000000000";
SPRITE7( 3) <= "0001111111111000";
SPRITE7( 4) <= "0001111111111000";
SPRITE7( 5) <= "0000000001110000";
SPRITE7( 6) <= "0000000001110000";
SPRITE7( 7) <= "0000000011100000";
SPRITE7( 8) <= "0000000011100000";
SPRITE7( 9) <= "0000000111000000";
SPRITE7(10) <= "0000000111000000";
SPRITE7(11) <= "0000001110000000";
SPRITE7(12) <= "0000001110000000";
SPRITE7(13) <= "0000000000000000";
SPRITE7(14) <= "0000000000000000";
SPRITE7(15) <= "0000000000000000";

SPRITE8( 0) <= "0000000000000000";
SPRITE8( 1) <= "0000000000000000";
SPRITE8( 2) <= "0000000000000000";
SPRITE8( 3) <= "0000111111110000";
SPRITE8( 4) <= "0001111111111000";
SPRITE8( 5) <= "0001100000011000";
SPRITE8( 6) <= "0001100000011000";
SPRITE8( 7) <= "0000111111110000";
SPRITE8( 8) <= "0000111111110000";
SPRITE8( 9) <= "0001100000011000";
SPRITE8(10) <= "0001100000011000";
SPRITE8(11) <= "0001111111111000";
SPRITE8(12) <= "0000111111110000";
SPRITE8(13) <= "0000000000000000";
SPRITE8(14) <= "0000000000000000";
SPRITE8(15) <= "0000000000000000";

SPRITE9(0)  <= "0000000000000000";
SPRITE9(1)  <= "0000000000000000";
```

```
SPRITE9(2)  <= "0000000000000000";
SPRITE9(3)  <= "0000111111110000";
SPRITE9(4)  <= "0001111111111000";
SPRITE9(5)  <= "0001100000011000";
SPRITE9(6)  <= "0001100000011000";
SPRITE9(7)  <= "0001111111111000";
SPRITE9(8)  <= "0000111111111000";
SPRITE9(9)  <= "0000000000111000";
SPRITE9(10) <= "0000000000111000";
SPRITE9(11) <= "0000000000111000";
SPRITE9(12) <= "0000000000111000";
SPRITE9(13) <= "0000000000000000";
SPRITE9(14) <= "0000000000000000";
SPRITE9(15) <= "0000000000000000";

-- Horizontal and vertical counters

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

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

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

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

-- X and Y Counters
XCounter: process (clk)
begin
if rising_edge(clk) then
if (Hcount > HSYNC + HBACK_PORCH and Xcount < HACTIVE - 1) then
Xcount <= Xcount + 1;
else
Xcount <= (others => '0');
end if;
end if;
end process XCounter;

YCounter: process (clk)
begin
if rising_edge(clk) then
if (Vcount > VSYNC + VBACK_PORCH and Ycount < VACTIVE) then
if Hcount = 1 then
Ycount <= Ycount + 1;
end if;
else
Ycount <= (others => '0');
end if;
end if;
end process YCounter;

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

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

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

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

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

-- Silhouette Updater
SilUpdate : process (clk, y, x)
begin
if rising_edge(clk) then
-- workaround begin
if x = 0 then
silhouette(to_integer(y)*20) <= sil;
else
silhouette(to_integer(y)*20 + 20 - to_integer(x)) <= sil;
end if;
-- workaround end
-- line below should be the correct code, but does not work for some reason
--silhouette(to_integer(y)*20 + 19 - to_integer(x)) <= sil;
end if;
end process SilUpdate;

--Generates the balls on screen based on the vertical, horizontal coordinates and the size of the rad
CircleGen : process (clk)
begin
  if rising_edge(clk) then
CIRCLE_HSTART <= ch_in;
CIRCLE_VSTART <= cv_in;
CIRCLE_RADIUS <= r_in;
if ((to_integer(Hcount)-HSYNC - HBACK_PORCH-to_integer(CIRCLE_HSTART))*(to_integer(Hcount)-HSYNC - HB
  circle <= '1';
else
  circle <= '0';
end if;
  end if;
end process CircleGen;

  -- Registered video signals going to the video DAC

  VideoOut: process (clk, reset)
  variable SCORE_V : unsigned (9 downto 0);
  variable SCORE_H : unsigned (9 downto 0);
  variable SCORE_H2: unsigned (9 downto 0);
  variable LOSE_H3: unsigned (9 downto 0);
  variable HIGH_H4: unsigned (9 downto 0);
  variable SCORE_TEMP: unsigned(79 downto 0);
  variable LOSE_TEMP: unsigned(127 downto 0);
  variable HIGH_TEMP: unsigned(95 downto 0);
  variable SCORE1s: unsigned(15 downto 0);
  variable SCORE2s: unsigned(15 downto 0);
  variable SCORE3s: unsigned(15 downto 0);
  variable SCORE4s: unsigned(15 downto 0);
  variable HSCORE1s: unsigned(15 downto 0);
  variable HSCORE2s: unsigned(15 downto 0);
  variable HSCORE3s: unsigned(15 downto 0);
  variable HSCORE4s: unsigned(15 downto 0);

  begin
    if reset = '1' then
      VGA_R <= "0000000000";
      VGA_G <= "0000000000";
      VGA_B <= "0000000000";
elsif clk'event and clk = '1' then
  SCORE_V := (Vcount - (VSYNC + VBACK_PORCH)) AND "0000001111";
  SCORE_H := (Hcount - (HSYNC + HBACK_PORCH)) AND "0001111111";
  SCORE_H2:= (Hcount - (HSYNC + HBACK_PORCH)) AND "0000001111";
  LOSE_H3 := (Hcount - (HSYNC + HBACK_PORCH));
  HIGH_H4 := (Hcount - (HSYNC + HBACK_PORCH));

  --Updates the score when increment is flagged
  if to_integer(CIRCLE_RADIUS) < 45 and increment = '1' then
if (to_integer(SCORE4) < 9) then
  SCORE4 <= SCORE4 + 1;
else
  SCORE4 <= "0000";
  if (to_integer(SCORE3) < 9) then
SCORE3 <= SCORE3 + 1;
  else
SCORE3 <= "0000";
if(to_integer(SCORE2) < 9) then
  SCORE2 <= SCORE2 + 1;
else
  SCORE2 <= "0000";
  if(to_integer(SCORE1) < 9) then
SCORE1 <= SCORE1 + 1;
  else
SCORE1 <= "0000";
  end if;
end if;
  end if;
end if;
increment <= '0';
  end if;
  --Display the Score in LED
  if to_integer(SCORE4) = 9 then
HEX0 <= "0010000";
  elsif to_integer(SCORE4) = 8 then
HEX0 <= "0000000";
  elsif to_integer(SCORE4) = 7 then
HEX0 <= "1111000";
  elsif to_integer(SCORE4) = 6 then
HEX0 <= "0000010";
  elsif to_integer(SCORE4) = 5 then
HEX0 <= "0010010";
  elsif to_integer(SCORE4) = 4 then
HEX0 <= "0011001";
  elsif to_integer(SCORE4) = 3 then
HEX0 <= "0110000";
  elsif to_integer(SCORE4) = 2 then
HEX0 <= "0100100";
  elsif to_integer(SCORE4) = 1 then
```

```
HEX0 <= "1111001";
  elsif to_integer(SCORE4) = 0 then
HEX0 <= "1000000";
  end if;
  if to_integer(SCORE3) = 9 then
HEX1 <= "0010000";
  elsif to_integer(SCORE3) = 8 then
HEX1 <= "0000000";
  elsif to_integer(SCORE3) = 7 then
HEX1 <= "1111000";
  elsif to_integer(SCORE3) = 6 then
HEX1 <= "0000010";
  elsif to_integer(SCORE3) = 5 then
HEX1 <= "0010010";
  elsif to_integer(SCORE3) = 4 then
HEX1 <= "0011001";
  elsif to_integer(SCORE3) = 3 then
HEX1 <= "0110000";
  elsif to_integer(SCORE3) = 2 then
HEX1 <= "0100100";
  elsif to_integer(SCORE3) = 1 then
HEX1 <= "1111001";
  elsif to_integer(SCORE3) = 0 then
HEX1 <= "1000000";
  end if;
  if to_integer(SCORE2) = 9 then
HEX2 <= "0010000";
  elsif to_integer(SCORE2) = 8 then
HEX2 <= "0000000";
  elsif to_integer(SCORE2) = 7 then
HEX2 <= "1111000";
  elsif to_integer(SCORE2) = 6 then
HEX2 <= "0000010";
  elsif to_integer(SCORE2) = 5 then
HEX2 <= "0010010";
  elsif to_integer(SCORE2) = 4 then
HEX2 <= "0011001";
  elsif to_integer(SCORE2) = 3 then
HEX2 <= "0110000";
  elsif to_integer(SCORE2) = 2 then
HEX2 <= "0100100";
  elsif to_integer(SCORE2) = 1 then
HEX2 <= "1111001";
  elsif to_integer(SCORE2) = 0 then
HEX2 <= "1000000";
  end if;
  if to_integer(SCORE1) = 9 then
```

```
HEX3 <= "0010000";
  elsif to_integer(SCORE1) = 8 then
HEX3 <= "0000000";
  elsif to_integer(SCORE1) = 7 then
HEX3 <= "1111000";
  elsif to_integer(SCORE1) = 6 then
HEX3 <= "0000010";
  elsif to_integer(SCORE1) = 5 then
HEX3 <= "0010010";
  elsif to_integer(SCORE1) = 4 then
HEX3 <= "0011001";
  elsif to_integer(SCORE1) = 3 then
HEX3 <= "0110000";
  elsif to_integer(SCORE1) = 2 then
HEX3 <= "0100100";
  elsif to_integer(SCORE1) = 1 then
HEX3 <= "1111001";
  elsif to_integer(SCORE1) = 0 then
HEX3 <= "1000000";
  end if;
  --Sets the different modes of the game
  if goalie = '1' then
restart <= '1';
hit <= '1';
ninja_hit <= '0';
  elsif dodge = '1' then
restart <= '1';
hit <= '0';
ninja_hit <= '0';
  elsif ninja = '1' then
restart <= '1';
ninja_hit <= '1';
  end if;
  if ninja_hit = '1' then
hit <= h_in;
  end if;
  --restarts the game when the corresponding key is pressed
  if nlose = '0' and restart = '1' and to_integer(CIRCLE_RADIUS) < 5  then
SCORE4 <= "0000";
SCORE3 <= "0000";
SCORE2 <= "0000";
SCORE1 <= "0000";
nlose <= '1';
tempLose <= '0';
restart <= '0';
  end if;
      if vga_hblank = '1' or vga_vblank = '1' then
```

```
        VGA_R <= "0000000000";
        VGA_G <= "0000000000";
        VGA_B <= "0000000000";
  elsif nlose = '0' then
--Screen displayed when the player loses, and saves high score if the current score is higher than sa
if to_integer(Hcount- (HSYNC + HBACK_PORCH)) < 384 and to_integer(Hcount- (HSYNC + HBACK_PORCH)) > 25
LOSE_TEMP := SPRITE_LOSE(to_integer(unsigned(SCORE_V)));
if (LOSE_TEMP(to_integer(unsigned(127 - LOSE_H3))) = '1') then
VGA_R <= "1111111111";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
else
VGA_R <= "0000000000";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
end if;
end if;
if (to_integer(HSCORE1) < to_integer(SCORE1)) then
HSCORE1 <= SCORE1;
HSCORE2 <= SCORE2;
HSCORE3 <= SCORE3;
HSCORE4 <= SCORE4;
elsif (to_integer(HSCORE1) = to_integer(SCORE1)) then
if (to_integer(HSCORE2) < to_integer(SCORE2)) then
HSCORE1 <= SCORE1;
HSCORE2 <= SCORE2;
HSCORE3 <= SCORE3;
HSCORE4 <= SCORE4;
elsif (to_integer(HSCORE2) = to_integer(SCORE2)) then
if (to_integer(HSCORE3) < to_integer(SCORE3)) then
HSCORE1 <= SCORE1;
HSCORE2 <= SCORE2;
HSCORE3 <= SCORE3;
HSCORE4 <= SCORE4;
elsif (to_integer(HSCORE3) = to_integer(SCORE3)) then
if (to_integer(HSCORE4) < to_integer(SCORE4)) then
HSCORE1 <= SCORE1;
HSCORE2 <= SCORE2;
HSCORE3 <= SCORE3;
HSCORE4 <= SCORE4;
end if;
end if;
end if;
end if;
restart <= '0';
  --Display the HSCORE text
  elsif to_integer(Hcount - (HSYNC + HBACK_PORCH)) > 0 and to_integer(Hcount - (HSYNC + HBACK_PORCH))
```

```
HIGH_TEMP := SPRITE_HSCORE(to_integer(unsigned(SCORE_V)));
if (HIGH_TEMP(to_integer(unsigned(96 - HIGH_H4))) = '1') then
  VGA_R <= "1111111111";
  VGA_G <= "0000000000";
  VGA_B <= "0000000000";
    else
  if circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)))
VGA_R <= "0000000000";
VGA_G <= "1000000000";
VGA_B <= "0000000000";
    if CIRCLE_RADIUS > 46 and touched = '0' then
  tempLose <= '1';
    end if;
  elsif circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)
VGA_R <= "1000000000";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "0000000000";
VGA_G <= "1111111111";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 then
touched <= '1';
tempLose <= '0';
end if;
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "1111111111";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 then
nlose <= '0';
end if;
  elsif silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))) = '1' then
VGA_R <= "1111111111";
VGA_G <= "1111111111";
VGA_B <= "1111111111";
  else
VGA_R <= "0000000000";
VGA_G <= "1111111111";
VGA_B <= "1111111111";
  end if;
end if;
  --Display the SCORE text
  elsif to_integer(Hcount- (HSYNC + HBACK_PORCH)) > 559 and to_integer(Vcount - (VSYNC + VBACK_PORCH)
SCORE_TEMP := SPRITE_SCORE(to_integer(unsigned(SCORE_V)));
    if (SCORE_TEMP(to_integer(unsigned(127 - SCORE_H))) = '1') then
  VGA_R <= "1111111111";
  VGA_G <= "0000000000";
  VGA_B <= "0000000000";
    else
  if circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)))
VGA_R <= "0000000000";
VGA_G <= "1000000000";
VGA_B <= "0000000000";
    if CIRCLE_RADIUS > 46 and touched = '0' then
  tempLose <= '1';
    end if;
  elsif circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)
VGA_R <= "1000000000";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "0000000000";
VGA_G <= "1111111111";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 then
touched <= '1';
tempLose <= '0';
end if;
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "1111111111";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 then
nlose <= '0';
end if;
  elsif silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))) = '1' then
VGA_R <= "1111111111";
VGA_G <= "1111111111";
VGA_B <= "1111111111";
  else
VGA_R <= "0000000000";
VGA_G <= "1111111111";
VGA_B <= "1111111111";
  end if;
    end if;
  --determine whether the player gets points or loses
  elsif to_integer(Vcount - (VSYNC + VBACK_PORCH)) > 478 and (to_integer(Hcount)-HSYNC - HBACK_PORCH)
if CIRCLE_RADIUS > 46 and hit = '0' then
increment <= '1';
elsif hit = '1'and tempLose = '1'  then
nlose <= '0';
elsif hit = '1' and CIRCLE_RADIUS > 46 then
increment <= '1';
```

```
touched <= '0';
end if;

  elsif to_integer(Hcount - (HSYNC+ HBACK_PORCH)) > 0 and to_integer(Hcount - (HSYNC+ HBACK_PORCH)) <
if to_integer(HSCORE1) = 9 then
  HSCORE1s := SPRITE9(to_integer(unsigned(SCORE_V)));
elsif to_integer(HSCORE1) = 8 then
  HSCORE1s := SPRITE8(to_integer(unsigned(SCORE_V)));
elsif to_integer(HSCORE1) = 7 then
  HSCORE1s := SPRITE7(to_integer(unsigned(SCORE_V)));
elsif to_integer(HSCORE1) = 6 then
  HSCORE1s := SPRITE6(to_integer(unsigned(SCORE_V)));
elsif to_integer(HSCORE1) = 5 then
  HSCORE1s := SPRITE5(to_integer(unsigned(SCORE_V)));
elsif to_integer(HSCORE1) = 4 then
  HSCORE1s := SPRITE4(to_integer(unsigned(SCORE_V)));
elsif to_integer(HSCORE1) = 3 then
  HSCORE1s := SPRITE3(to_integer(unsigned(SCORE_V)));
elsif to_integer(HSCORE1) = 2 then
  HSCORE1s := SPRITE2(to_integer(unsigned(SCORE_V)));
elsif to_integer(HSCORE1) = 1 then
  HSCORE1s := SPRITE1(to_integer(unsigned(SCORE_V)));
elsif to_integer(HSCORE1) = 0 then
  HSCORE1s := SPRITE0(to_integer(unsigned(SCORE_V)));
end if;
if (HSCORE1s(to_integer(unsigned(15- SCORE_H2))) = '1') then
  VGA_R <= "1111111111";
  VGA_G <= "0000000000";
  VGA_B <= "0000000000";
else
  if circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)))
VGA_R <= "0000000000";
VGA_G <= "1000000000";
VGA_B <= "0000000000";
    if CIRCLE_RADIUS > 46 and touched = '0' then
  tempLose <= '1';
    end if;
  elsif circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)
VGA_R <= "1000000000";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "0000000000";
VGA_G <= "1111111111";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 then
touched <= '1';
```

```
tempLose <= '0';
end if;
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "1111111111";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 then
nlose <= '0';
end if;
  elsif silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))) = '1' then
VGA_R <= "1111111111";
VGA_G <= "1111111111";
VGA_B <= "1111111111";
  else
VGA_R <= "0000000000";
VGA_G <= "1111111111";
VGA_B <= "1111111111";
  end if;
end if;
  elsif to_integer(Hcount - (HSYNC+ HBACK_PORCH)) > 15 and to_integer(Hcount - (HSYNC+ HBACK_PORCH))
if to_integer(HSCORE2) = 9 then
  HSCORE2s := SPRITE9(to_integer(unsigned(SCORE_V)));
elsif to_integer(HSCORE2) = 8 then
  HSCORE2s := SPRITE8(to_integer(unsigned(SCORE_V)));
elsif to_integer(HSCORE2) = 7 then
  HSCORE2s := SPRITE7(to_integer(unsigned(SCORE_V)));
elsif to_integer(HSCORE2) = 6 then
  HSCORE2s := SPRITE6(to_integer(unsigned(SCORE_V)));
elsif to_integer(HSCORE2) = 5 then
  HSCORE2s := SPRITE5(to_integer(unsigned(SCORE_V)));
elsif to_integer(HSCORE2) = 4 then
  HSCORE2s := SPRITE4(to_integer(unsigned(SCORE_V)));
elsif to_integer(HSCORE2) = 3 then
  HSCORE2s := SPRITE3(to_integer(unsigned(SCORE_V)));
elsif to_integer(HSCORE2) = 2 then
  HSCORE2s := SPRITE2(to_integer(unsigned(SCORE_V)));
elsif to_integer(HSCORE2) = 1 then
  HSCORE2s := SPRITE1(to_integer(unsigned(SCORE_V)));
elsif to_integer(HSCORE2) = 0 then
  HSCORE2s := SPRITE0(to_integer(unsigned(SCORE_V)));
end if;
if (HSCORE2s(to_integer(unsigned(15- SCORE_H2))) = '1') then
VGA_R <= "1111111111";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
else
  if circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)))
```

```
VGA_R <= "0000000000";
VGA_G <= "1000000000";
VGA_B <= "0000000000";
    if CIRCLE_RADIUS > 46 and touched = '0' then
  tempLose <= '1';
    end if;
  elsif circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)
VGA_R <= "1000000000";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "0000000000";
VGA_G <= "1111111111";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 then
touched <= '1';
tempLose <= '0';
end if;
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "1111111111";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 then
nlose <= '0';
end if;
  elsif silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))) = '1' then
VGA_R <= "1111111111";
VGA_G <= "1111111111";
VGA_B <= "1111111111";
  else
VGA_R <= "0000000000";
VGA_G <= "1111111111";
VGA_B <= "1111111111";
  end if;
end if;
elsif to_integer(Hcount - (HSYNC+ HBACK_PORCH)) > 31 and to_integer(Hcount - (HSYNC+ HBACK_PORCH)) <
  if to_integer(HSCORE3) = 9 then
HSCORE3s := SPRITE9(to_integer(unsigned(SCORE_V)));
  elsif to_integer(HSCORE3) = 8 then
HSCORE3s := SPRITE8(to_integer(unsigned(SCORE_V)));
  elsif to_integer(HSCORE3) = 7 then
HSCORE3s := SPRITE7(to_integer(unsigned(SCORE_V)));
  elsif to_integer(HSCORE3) = 6 then
HSCORE3s := SPRITE6(to_integer(unsigned(SCORE_V)));
  elsif to_integer(HSCORE3) = 5 then
HSCORE3s := SPRITE5(to_integer(unsigned(SCORE_V)));
  elsif to_integer(HSCORE3) = 4 then
```

```
HSCORE3s := SPRITE4(to_integer(unsigned(SCORE_V)));
  elsif to_integer(HSCORE3) = 3 then
HSCORE3s := SPRITE3(to_integer(unsigned(SCORE_V)));
  elsif to_integer(HSCORE3) = 2 then
HSCORE3s := SPRITE2(to_integer(unsigned(SCORE_V)));
  elsif to_integer(HSCORE3) = 1 then
HSCORE3s := SPRITE1(to_integer(unsigned(SCORE_V)));
  elsif to_integer(HSCORE3) = 0 then
HSCORE3s := SPRITE0(to_integer(unsigned(SCORE_V)));
  end if;
  if (HSCORE3s(to_integer(unsigned(15- SCORE_H2))) = '1') then
VGA_R <= "1111111111";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
  else
  if circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)))
VGA_R <= "0000000000";
VGA_G <= "1000000000";
VGA_B <= "0000000000";
    if CIRCLE_RADIUS > 46 and touched = '0' then
  tempLose <= '1';
    end if;
  elsif circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)
VGA_R <= "1000000000";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "0000000000";
VGA_G <= "1111111111";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 then
touched <= '1';
tempLose <= '0';
end if;
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "1111111111";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 then
nlose <= '0';
end if;
  elsif silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))) = '1' then
VGA_R <= "1111111111";
VGA_G <= "1111111111";
VGA_B <= "1111111111";
  else
VGA_R <= "0000000000";
```

```
VGA_G <= "1111111111";
VGA_B <= "1111111111";
  end if;
end if;
elsif to_integer(Hcount - (HSYNC+ HBACK_PORCH)) > 47 and to_integer(Hcount - (HSYNC+ HBACK_PORCH)) <
  if to_integer(HSCORE4) = 9 then
HSCORE4s := SPRITE9(to_integer(unsigned(SCORE_V)));
  elsif to_integer(HSCORE4) = 8 then
HSCORE4s := SPRITE8(to_integer(unsigned(SCORE_V)));
  elsif to_integer(HSCORE4) = 7 then
HSCORE4s := SPRITE7(to_integer(unsigned(SCORE_V)));
  elsif to_integer(HSCORE4) = 6 then
HSCORE4s := SPRITE6(to_integer(unsigned(SCORE_V)));
  elsif to_integer(HSCORE4) = 5 then
HSCORE4s := SPRITE5(to_integer(unsigned(SCORE_V)));
  elsif to_integer(HSCORE4) = 4 then
HSCORE4s := SPRITE4(to_integer(unsigned(SCORE_V)));
  elsif to_integer(HSCORE4) = 3 then
HSCORE4s := SPRITE3(to_integer(unsigned(SCORE_V)));
  elsif to_integer(HSCORE4) = 2 then
HSCORE4s := SPRITE2(to_integer(unsigned(SCORE_V)));
  elsif to_integer(HSCORE4) = 1 then
HSCORE4s := SPRITE1(to_integer(unsigned(SCORE_V)));
  elsif to_integer(HSCORE4) = 0 then
HSCORE4s := SPRITE0(to_integer(unsigned(SCORE_V)));
  end if;
if (HSCORE4s(to_integer(unsigned(15- SCORE_H2))) = '1') then
VGA_R <= "1111111111";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
else
  if circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)))
VGA_R <= "0000000000";
VGA_G <= "1000000000";
VGA_B <= "0000000000";
    if CIRCLE_RADIUS > 46 and touched = '0' then
  tempLose <= '1';
    end if;
  elsif circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)
VGA_R <= "1000000000";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "0000000000";
VGA_G <= "1111111111";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 then
```

```
touched <= '1';
tempLose <= '0';
end if;
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "1111111111";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 then
nlose <= '0';
end if;
  elsif silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))) = '1' then
VGA_R <= "1111111111";
VGA_G <= "1111111111";
VGA_B <= "1111111111";
  else
VGA_R <= "0000000000";
VGA_G <= "1111111111";
VGA_B <= "1111111111";
  end if;
end if;
  elsif to_integer(Hcount - (HSYNC+ HBACK_PORCH)) > 576 and to_integer(Hcount - (HSYNC+ HBACK_PORCH))
if to_integer(SCORE1) = 9 then
  SCORE1s := SPRITE9(to_integer(unsigned(SCORE_V)));
elsif to_integer(SCORE1) = 8 then
  SCORE1s := SPRITE8(to_integer(unsigned(SCORE_V)));
elsif to_integer(SCORE1) = 7 then
  SCORE1s := SPRITE7(to_integer(unsigned(SCORE_V)));
elsif to_integer(SCORE1) = 6 then
  SCORE1s := SPRITE6(to_integer(unsigned(SCORE_V)));
elsif to_integer(SCORE1) = 5 then
  SCORE1s := SPRITE5(to_integer(unsigned(SCORE_V)));
elsif to_integer(SCORE1) = 4 then
  SCORE1s := SPRITE4(to_integer(unsigned(SCORE_V)));
elsif to_integer(SCORE1) = 3 then
  SCORE1s := SPRITE3(to_integer(unsigned(SCORE_V)));
elsif to_integer(SCORE1) = 2 then
  SCORE1s := SPRITE2(to_integer(unsigned(SCORE_V)));
elsif to_integer(SCORE1) = 1 then
  SCORE1s := SPRITE1(to_integer(unsigned(SCORE_V)));
elsif to_integer(SCORE1) = 0 then
  SCORE1s := SPRITE0(to_integer(unsigned(SCORE_V)));
end if;
if (SCORE1s(to_integer(unsigned(15- SCORE_H2))) = '1') then
  VGA_R <= "1111111111";
  VGA_G <= "0000000000";
  VGA_B <= "0000000000";
else
```

```
  if circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)))
VGA_R <= "0000000000";
VGA_G <= "1000000000";
VGA_B <= "0000000000";
    if CIRCLE_RADIUS > 46 and touched = '0' then
  tempLose <= '1';
    end if;
  elsif circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)
VGA_R <= "1000000000";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "0000000000";
VGA_G <= "1111111111";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 then
touched <= '1';
tempLose <= '0';
end if;
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "1111111111";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 then
nlose <= '0';
end if;
  elsif silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))) = '1' then
VGA_R <= "1111111111";
VGA_G <= "1111111111";
VGA_B <= "1111111111";
  else
VGA_R <= "0000000000";
VGA_G <= "1111111111";
VGA_B <= "1111111111";
  end if;
end if;
  elsif to_integer(Hcount - (HSYNC+ HBACK_PORCH)) > 592 and to_integer(Hcount - (HSYNC+ HBACK_PORCH))
if to_integer(SCORE2) = 9 then
  SCORE2s := SPRITE9(to_integer(unsigned(SCORE_V)));
elsif to_integer(SCORE2) = 8 then
  SCORE2s := SPRITE8(to_integer(unsigned(SCORE_V)));
elsif to_integer(SCORE2) = 7 then
  SCORE2s := SPRITE7(to_integer(unsigned(SCORE_V)));
elsif to_integer(SCORE2) = 6 then
  SCORE2s := SPRITE6(to_integer(unsigned(SCORE_V)));
elsif to_integer(SCORE2) = 5 then
  SCORE2s := SPRITE5(to_integer(unsigned(SCORE_V)));
```

```
elsif to_integer(SCORE2) = 4 then
  SCORE2s := SPRITE4(to_integer(unsigned(SCORE_V)));
elsif to_integer(SCORE2) = 3 then
  SCORE2s := SPRITE3(to_integer(unsigned(SCORE_V)));
elsif to_integer(SCORE2) = 2 then
  SCORE2s := SPRITE2(to_integer(unsigned(SCORE_V)));
elsif to_integer(SCORE2) = 1 then
  SCORE2s := SPRITE1(to_integer(unsigned(SCORE_V)));
elsif to_integer(SCORE2) = 0 then
  SCORE2s := SPRITE0(to_integer(unsigned(SCORE_V)));
end if;
if (SCORE2s(to_integer(unsigned(15- SCORE_H2))) = '1') then
VGA_R <= "1111111111";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
else
  if circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)))
VGA_R <= "0000000000";
VGA_G <= "1000000000";
VGA_B <= "0000000000";
    if CIRCLE_RADIUS > 46 and touched = '0' then
  tempLose <= '1';
    end if;
  elsif circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)
VGA_R <= "1000000000";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "0000000000";
VGA_G <= "1111111111";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 then
touched <= '1';
tempLose <= '0';
end if;
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "1111111111";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 then
nlose <= '0';
end if;
  elsif silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))) = '1' then
VGA_R <= "1111111111";
VGA_G <= "1111111111";
VGA_B <= "1111111111";
  else
```

```
VGA_R <= "0000000000";
VGA_G <= "1111111111";
VGA_B <= "1111111111";
  end if;
end if;
elsif to_integer(Hcount - (HSYNC+ HBACK_PORCH)) > 607 and to_integer(Hcount - (HSYNC+ HBACK_PORCH)) <
  if to_integer(SCORE3) = 9 then
SCORE3s := SPRITE9(to_integer(unsigned(SCORE_V)));
  elsif to_integer(SCORE3) = 8 then
SCORE3s := SPRITE8(to_integer(unsigned(SCORE_V)));
  elsif to_integer(SCORE3) = 7 then
SCORE3s := SPRITE7(to_integer(unsigned(SCORE_V)));
  elsif to_integer(SCORE3) = 6 then
SCORE3s := SPRITE6(to_integer(unsigned(SCORE_V)));
  elsif to_integer(SCORE3) = 5 then
SCORE3s := SPRITE5(to_integer(unsigned(SCORE_V)));
  elsif to_integer(SCORE3) = 4 then
SCORE3s := SPRITE4(to_integer(unsigned(SCORE_V)));
  elsif to_integer(SCORE3) = 3 then
SCORE3s := SPRITE3(to_integer(unsigned(SCORE_V)));
  elsif to_integer(SCORE3) = 2 then
SCORE3s := SPRITE2(to_integer(unsigned(SCORE_V)));
  elsif to_integer(SCORE3) = 1 then
SCORE3s := SPRITE1(to_integer(unsigned(SCORE_V)));
  elsif to_integer(SCORE3) = 0 then
SCORE3s := SPRITE0(to_integer(unsigned(SCORE_V)));
  end if;
  if (SCORE3s(to_integer(unsigned(15- SCORE_H2))) = '1') then
VGA_R <= "1111111111";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
  else
  if circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)))
VGA_R <= "0000000000";
VGA_G <= "1000000000";
VGA_B <= "0000000000";
    if CIRCLE_RADIUS > 46 and touched = '0' then
  tempLose <= '1';
    end if;
  elsif circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)
VGA_R <= "1000000000";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "0000000000";
VGA_G <= "1111111111";
VGA_B <= "0000000000";
```

```
if CIRCLE_RADIUS > 46 then
touched <= '1';
tempLose <= '0';
end if;
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "1111111111";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 then
nlose <= '0';
end if;
  elsif silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))) = '1' then
VGA_R <= "1111111111";
VGA_G <= "1111111111";
VGA_B <= "1111111111";
  else
VGA_R <= "0000000000";
VGA_G <= "1111111111";
VGA_B <= "1111111111";
  end if;
end if;
elsif to_integer(Hcount - (HSYNC+ HBACK_PORCH)) > 623 and to_integer(Hcount - (HSYNC+ HBACK_PORCH)) <
  if to_integer(SCORE4) = 9 then
SCORE4s := SPRITE9(to_integer(unsigned(SCORE_V)));
  elsif to_integer(SCORE4) = 8 then
SCORE4s := SPRITE8(to_integer(unsigned(SCORE_V)));
  elsif to_integer(SCORE4) = 7 then
SCORE4s := SPRITE7(to_integer(unsigned(SCORE_V)));
  elsif to_integer(SCORE4) = 6 then
SCORE4s := SPRITE6(to_integer(unsigned(SCORE_V)));
  elsif to_integer(SCORE4) = 5 then
SCORE4s := SPRITE5(to_integer(unsigned(SCORE_V)));
  elsif to_integer(SCORE4) = 4 then
SCORE4s := SPRITE4(to_integer(unsigned(SCORE_V)));
  elsif to_integer(SCORE4) = 3 then
SCORE4s := SPRITE3(to_integer(unsigned(SCORE_V)));
  elsif to_integer(SCORE4) = 2 then
SCORE4s := SPRITE2(to_integer(unsigned(SCORE_V)));
  elsif to_integer(SCORE4) = 1 then
SCORE4s := SPRITE1(to_integer(unsigned(SCORE_V)));
  elsif to_integer(SCORE4) = 0 then
SCORE4s := SPRITE0(to_integer(unsigned(SCORE_V)));
  end if;
  if (SCORE4s(to_integer(unsigned(15- SCORE_H2))) = '1') then
VGA_R <= "1111111111";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
```

```
  else
if circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))) =
  VGA_R <= "0000000000";
  VGA_G <= "1000000000";
  VGA_B <= "0000000000";
  if CIRCLE_RADIUS > 46 and touched = '0' then
    tempLose <= '1';
      end if;
elsif circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)))
  VGA_R <= "1000000000";
  VGA_G <= "0000000000";
  VGA_B <= "0000000000";
elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)))
  VGA_R <= "0000000000";
  VGA_G <= "1111111111";
  VGA_B <= "0000000000";
  if CIRCLE_RADIUS > 46 then
touched <= '1';
tempLose <= '0';
  end if;
elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)))
  VGA_R <= "1111111111";
  VGA_G <= "0000000000";
  VGA_B <= "0000000000";
  if CIRCLE_RADIUS > 46 then
nlose <= '0';
  end if;
elsif silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))) = '1' then
  VGA_R <= "1111111111";
  VGA_G <= "1111111111";
  VGA_B <= "1111111111";
    else
  VGA_R <= "0000000000";
  VGA_G <= "1111111111";
  VGA_B <= "1111111111";
    end if;
  end if;
  elsif circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)
VGA_R <= "0000000000";
VGA_G <= "1000000000";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 and touched = '0' then
tempLose <= '1';
end if;
  elsif circle = '1' and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5)
VGA_R <= "1000000000";
VGA_G <= "0000000000";
```

```
VGA_B <= "0000000000";
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "0000000000";
VGA_G <= "1111111111";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 then
touched <= '1';
tempLose <= '0';
end if;
  elsif circle = '1'and silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))
VGA_R <= "1111111111";
VGA_G <= "0000000000";
VGA_B <= "0000000000";
if CIRCLE_RADIUS > 46 then
nlose <= '0';
end if;
  elsif silhouette(to_integer(Ycount(8 downto 5))*20 + to_integer(Xcount(9 downto 5))) = '1' then
VGA_R <= "1111111111";
        VGA_G <= "1111111111";
        VGA_B <= "1111111111";

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

  VGA_CLK <= clk;
  VGA_HS <= not vga_hsync;
  VGA_VS <= not vga_vsync;
  VGA_SYNC <= '0';
  VGA_BLANK <= not (vga_hsync or vga_vsync);

end rtl;
```