**Digital Systems Design**

VGA Controller

GIURGIU RAZVAN

CSATLOS ANDREI

GROUP : 30412

Content :

1. Project specifications
2. Timing
3. VGA Connector ( DE – 15 )
4. Block scheme
   1. Black Box
   2. Block scheme with main components
5. List of components and code
6. Inputs / Outputs
7. User manual – Active-HDL Simulation
8. Further development

1.Project specifications

The requirements for this project was to implement on a FPGA board, a VGA controller which is able to display 4 different images. The images must be selected from the FPGA board using switches and they also need to show the ability to change colors according to the user’s selection.

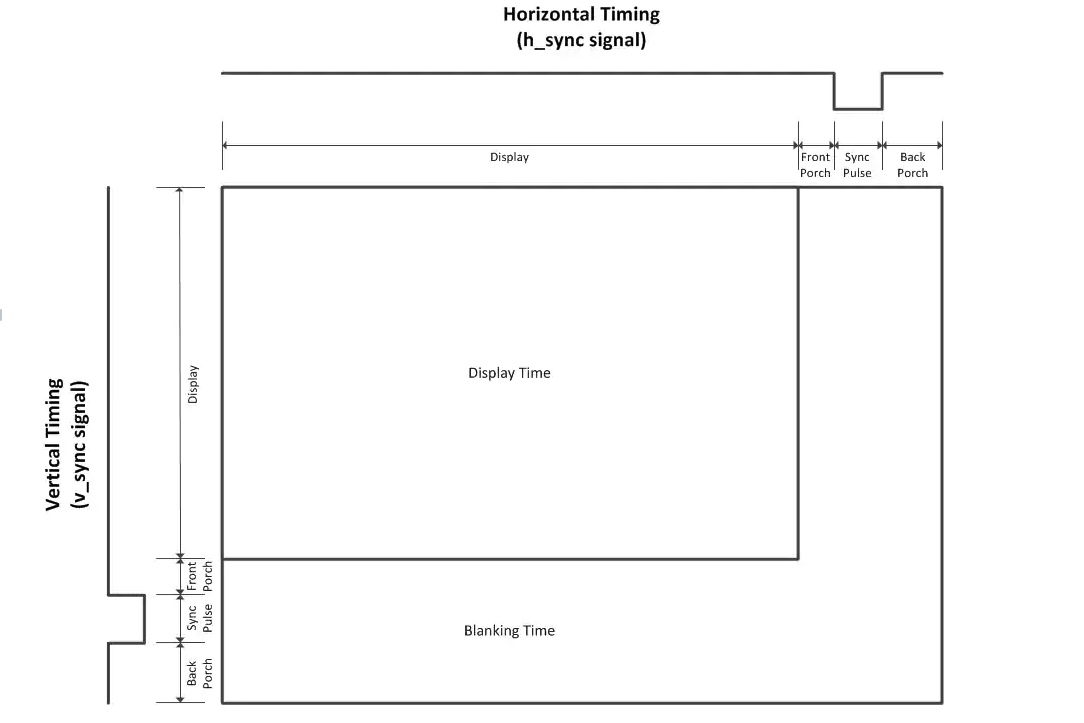
Images\Figures that are displayed can be moved on two axis, UP\DOWN and RIGHT\LEFT using buttons placed on the board.

2. Timing

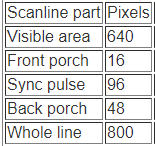
The figure illustrates the timing signals produced by the VGA controller. The controller contains two counters. One counter increments on pixel clocks and controls the timing of the h\_sync (horizontal sync) signal. By setting it up such that the display time starts at counter value 0, the counter value equals the pixel’s column coordinate during the display time. The horizontal display time is followed by a blanking time, which includes a horizontal front porch, the horizontal sync pulse itself, and the horizontal back porch, each of specified duration. At the end of the row, the counter resets to start the next row.

The other counter increments as each row completes, therefore controlling the timing of the v\_sync (vertical sync) signal. Again, this is set up such that the display time starts at counter value 0, so the counter value equals the pixel’s row coordinate during the display time. As before, the vertical display time is followed by a blanking time, with its corresponding front porch, sync pulse, and back porch. Once the vertical blanking time completes, the counter resets to begin the next screen refresh.

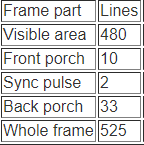
Using these counters, the VGA controller outputs the horizontal sync, vertical sync, display enable, and pixel coordinate signals. The sync pulses are specified as positive or negative polarity for each VGA mode.



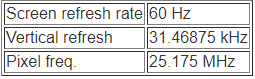
**Horizontal timing (line)**



## Vertical timing (frame)



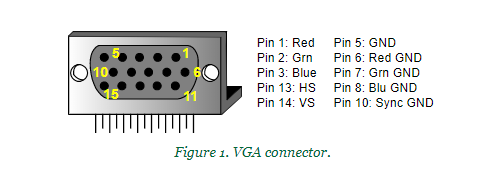
## General timing



3. VGA Connector ( DE – 15 )

VGA stands for Video Graphics Array. Initially, it refers specifically to the display hardware first introduced with IBM® PS/2 computer in 1987. With the widespread adoption, it now usually refers to the analog computer display standards the DE-15 Connector (commonly known as VGA connector).

We will only concentrate on the 5 signals out of 15 pins in this project. These signals are Red, Grn, Blue, HS, and VS. Red, Grn, and Blue are three analog signals that specify the color of a point on the screen



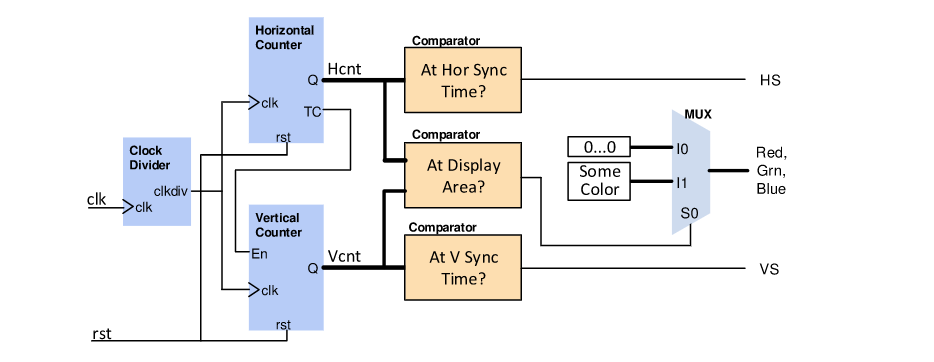
4. Block scheme

4.1 Black box

Using Xilinix ISE Design Suite after the synthesis has been done we can see the Black Box that is going to be implemented :



4.2 Block scheme



The block scheme may seem complicate but it is very simple. First we are are using a clock divider because we are havig a 50 mHz Clock signal and we need to work with a 25 mHz Clock signal. We start the sync process to generate our horizontal sync ( HSYNC ) and our vertical sync ( VSYNC ) as described above and we use a signal ( videoOn ) to keep track of the moments when we can draw something on the screen or not.

In the drawing process we are moving our initial points from which the drawing of our figure started up/down or left/rigt in a procedure name „Position”. At any particular moment we can press the reset button and our screen will become black.

Next, if we are allowed to draw, based on our selection we will draw one of 4 predefined figures. We can choose from a sqaure, lines, a triangle, and a spehere

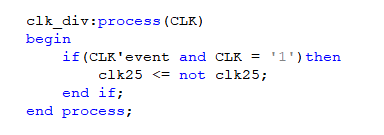
5. List of components and code

For this project we used the following components :

* Frequency divider
* 4 Counters
* Comparators
* Multiplexers

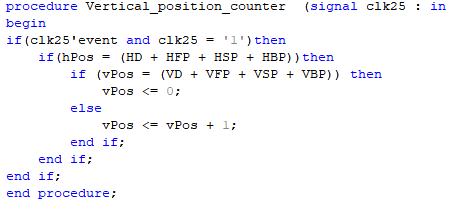
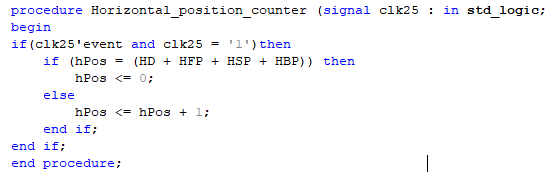
• Frequency divider :

used to transform a 50 mHz Clock to a 25 mHz Clock



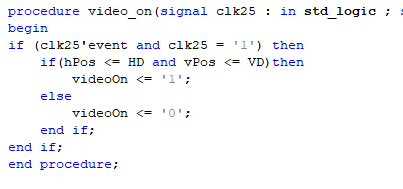
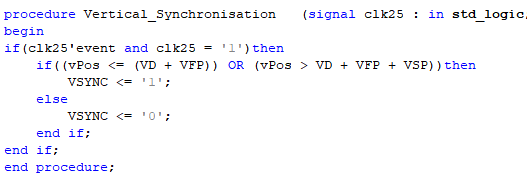
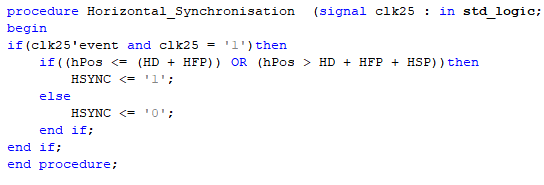
• Counters used for HSYNC and VSYNC

Used to generate the lines and the frames

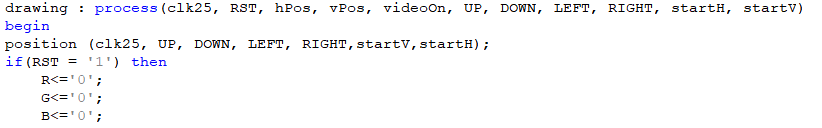


• Comparators

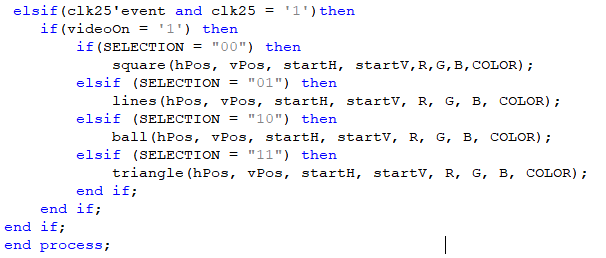
Used to check if the screen is in blinking time or we can draw



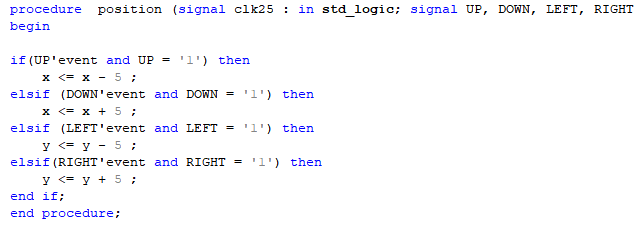
Next, the drawing process begins, we are changing the position of the figure acording or reset it according to the user input



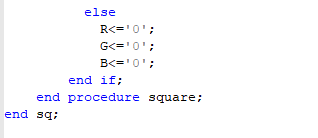
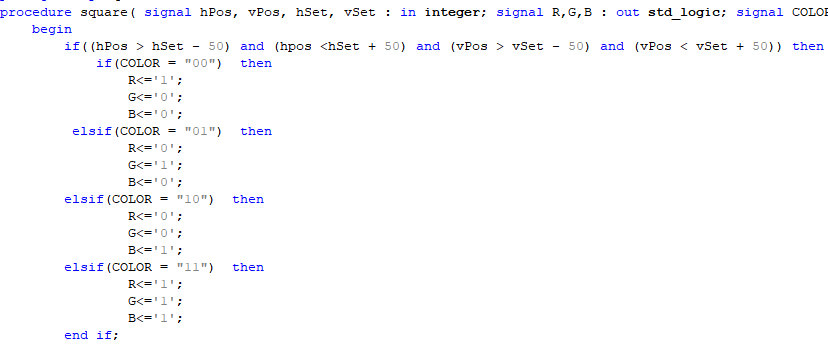
If the reset is not on we start drawing the figure that we want based on SELECTION



The procedure that is used for movement, check at each step if we need to move the figure on one of the 2 axis



We are using procedure to draw each figure, an example of how the drawing is done is showd here :



6. Inputs / Outputs

* Inputs :
* CLK
* UP
* DOWN
* LEFT
* RIGHT
* RST
* SELECTION
* COLOR
* Outputs :
* R
* G
* B
* HSYNC
* VSYNC

Description of each input and output :

CLK - a clock signal based on which all the components work

UP/DOWN – we move the figure by pushing the buttons assigned on an x axis up and down

RIGHT/ LEFT – we move the figure by pushing the buttons assigned on an y axis left and right

SELECTION – we have 4 figures from which we can select, encoded as

* SQUARE (”00”)
* LINES (”01”)
* SPHERE (”10”)
* TRIANGLE (”11”)

COLOR – we can select from 4 different colors encoded as

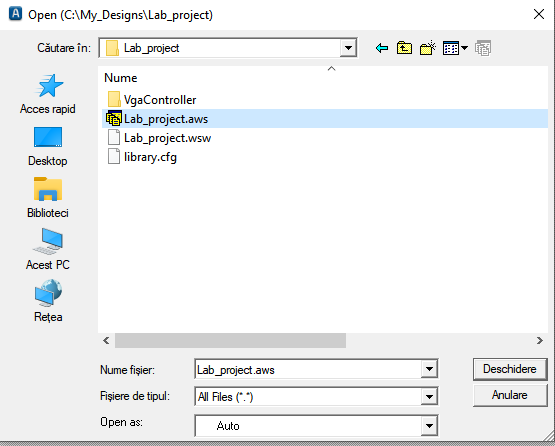
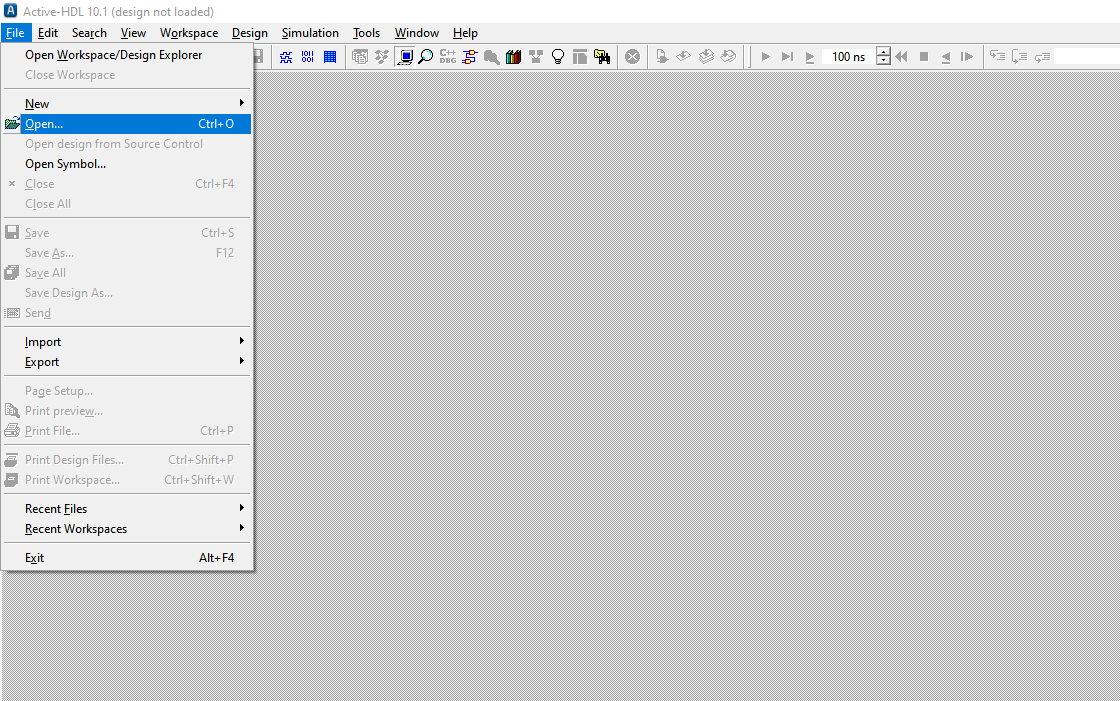
* RED (”00”)
* GREEN (”01”)
* BLUE (”10”)
* WHITE (”11”)

R/G/B – outputs that decide the color of the pixel at each step (RED, GREEN, BLUE)

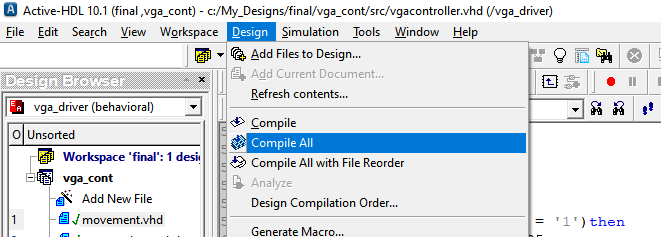
1. User manual – Active-HDL Simulation

In order to simulate our VGA Controller in Active-HDL we need to follow some steps after running the program

First step : After the aplication has started running we will open our project from **File -> Open**, or we can use a shortcut **CTRL + 0 .**



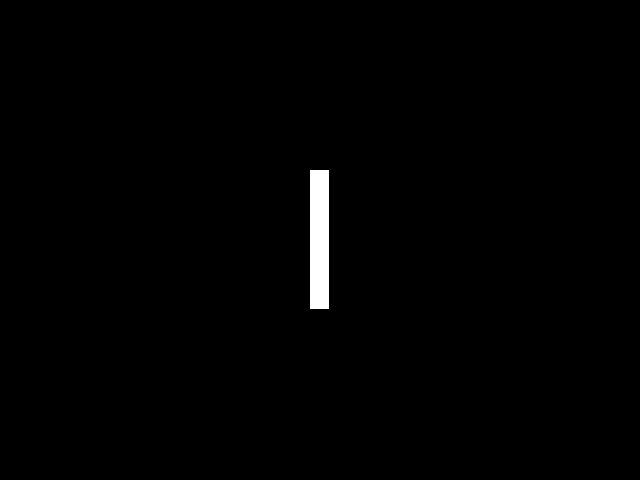
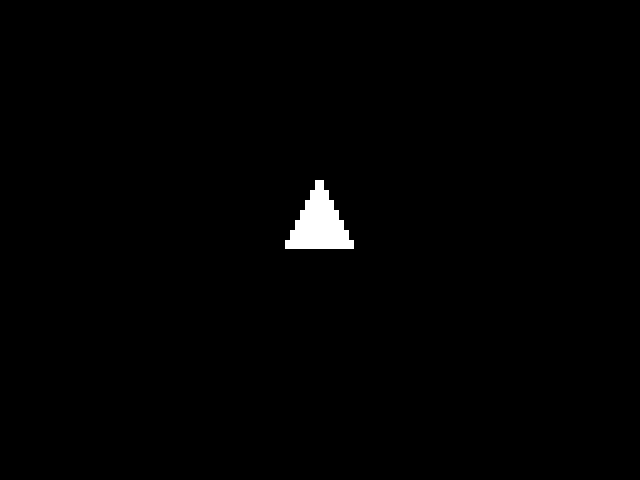
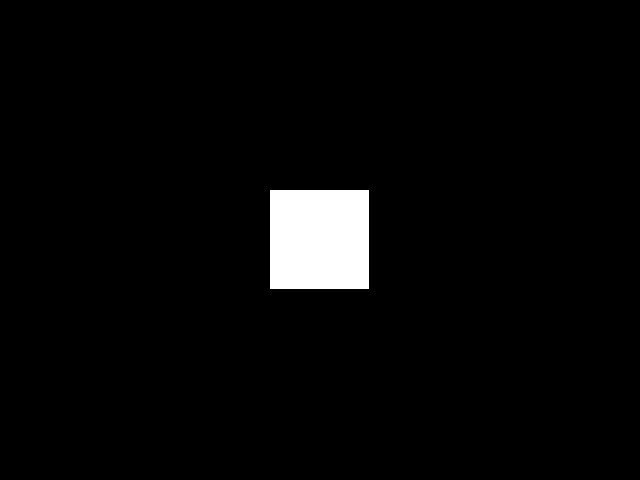
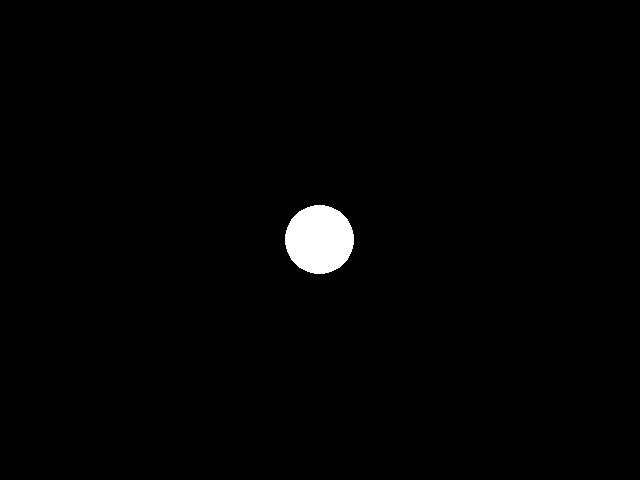
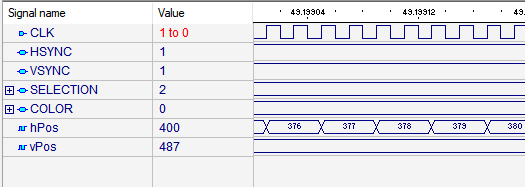
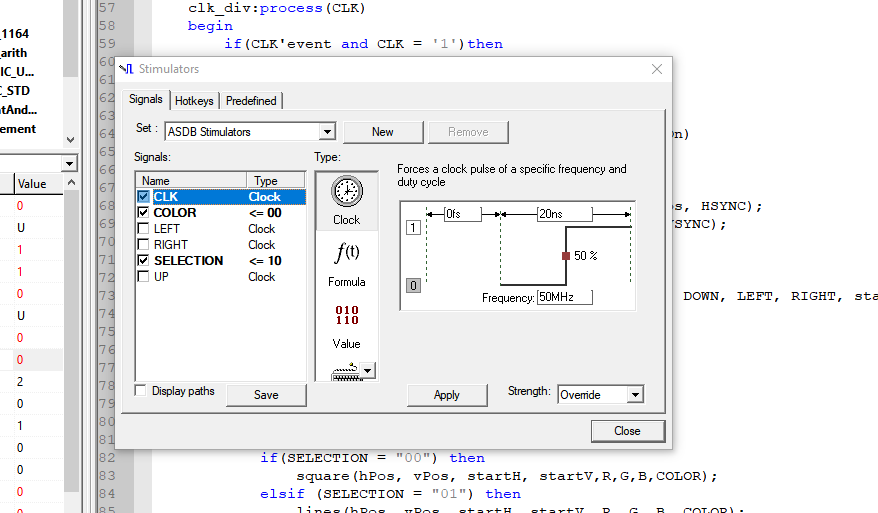
Second step : The code wil be displayed on the screen. To initialize the simulation we must first compile our project **Design -> Compile all**



Third step : We need now to initialize our simulation from the menu **Simulation -> Initialize Simulation.** In order to display our simulation window we need no press the icon **New Waveform**



Step four : We will select all signals that we are using in the project and we will go to **Stimulators** Tab. From the stimulator tab we can select values for our simulation. We must first assign a 50 MHz Clock to signal **CLK** and select which figure we want to display **SELECTION.** We can also change the color by assigning a value to **COLOR.**



1. Further Development

For furher development we can use a ROM memory to store more complex images/figures and we can transform the whole project in a game that can be controlled from the board or maybe from a keyboard. We can tweak the signals to get a better resolutin and a higher refresh rate so things can look more real.

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

use IEEE.NUMERIC\_STD.ALL;

use ieee.std\_logic\_textio.all;

use std.textio.all;

use work.sq.all;

use work.line.all;

use work.b.all;

use work.tr.all;

use work.countAndSync.all;

use work.movement.all;

entity vga\_driver is

Port ( CLK : in STD\_LOGIC;

RST : in STD\_LOGIC;

HSYNC : inout STD\_LOGIC;

VSYNC : inout STD\_LOGIC;

UP : in STD\_LOGIC;

DOWN : in STD\_LOGIC;

LEFT : in STD\_LOGIC;

RIGHT : in STD\_LOGIC;

SELECTION : inout STD\_LOGIC\_VECTOR ( 1 downto 0);

COLOR : inout STD\_LOGIC\_VECTOR ( 1 downto 0);

R : inout STD\_LOGIC ;

G : inout STD\_LOGIC ;

B : inout STD\_LOGIC);

end vga\_driver;

architecture Behavioral of vga\_driver is

signal clk25 : std\_logic := '0';

constant HD : integer := 639; -- 639 Horizontal Display (640)

constant HFP : integer := 16; -- 16 Right border (front porch)

constant HSP : integer := 96; -- 96 Sync pulse (Retrace)

constant HBP : integer := 48; -- 48 Left boarder (back porch)

constant VD : integer := 479; -- 479 Vertical Display (480)

constant VFP : integer := 10; -- 10 Right border (front porch)

constant VSP : integer := 2; -- 2 Sync pulse

constant VBP : integer := 33; -- 33 Left boarder (back porch)

signal hPos : integer := 0;

signal vPos : integer := 0;

signal videoOn : std\_logic := '0';

signal startH : integer := HD/2;

signal startV : integer := VD/2;

begin

clk\_div:process(CLK)

begin

if(CLK'event and CLK = '1')then

clk25 <= not clk25;

end if;

end process;

sync : process(clk25,RST,hPos,vPos,HSYNC,VSYNC, videoOn)

begin

Horizontal\_position\_counter (clk25,RST,hPos,vPos);

Vertical\_position\_counter(clk25, RST, hPos, vPos);

Horizontal\_Synchronisation (clk25, RST, hPos, vPos, HSYNC);

Vertical\_Synchronisation(clk25, RST, hPos, vPos, VSYNC);

video\_on(clk25, RST, hPos, vPos, videoOn);

end process;

drawing : process(clk25, RST, hPos, vPos, videoOn, UP, DOWN, LEFT, RIGHT, startH, startV)

begin

position (clk25, UP, DOWN, LEFT, RIGHT,startV,startH);

if(RST = '1') then

R<='0';

G<='0';

B<='0';

elsif(clk25'event and clk25 = '1')then

if(videoOn = '1') then

if(SELECTION = "00") then

square(hPos, vPos, startH, startV,R,G,B,COLOR);

elsif (SELECTION = "01") then

lines(hPos, vPos, startH, startV, R, G, B, COLOR);

elsif (SELECTION = "10") then

ball(hPos, vPos, startH, startV, R, G, B, COLOR);

elsif (SELECTION = "11") then

triangle(hPos, vPos, startH, startV, R, G, B, COLOR);

end if;

end if;

end if;

end process;

end Behavioral;