* **Design Methodology**

I started my VGA controller project with counting the pixels on screen first horizontally and when a row is done, I passed new row in downwards until every row is done. While counting I determine the pixel locations according to screens resolution which means its horizontal and vertical porches and sync pulses, so pixel locations will be in the active (displayed) screen. After I mapped out the screen, only thing remained was the color of the screen and its attachment to switches, so I assign total 12 switches which goes like 4 switches to red, 4 switches to green, 4 switches to blue. I have also assigned a button for disabling the view, simply turning off the VGA port output.

- In my preliminary lab report I stated that I want to make the initial color of the screen white. At first, I thought because at the beginning switches are ‘0’ it made output conflicts but afterwards I got the idea that using switches as inverted way. After inverted them, it worked as I planned.

**-** As I worked on my project and learnt VGA more my modules and especially my design has been changed compared to my preliminary lab report.

* **Results**

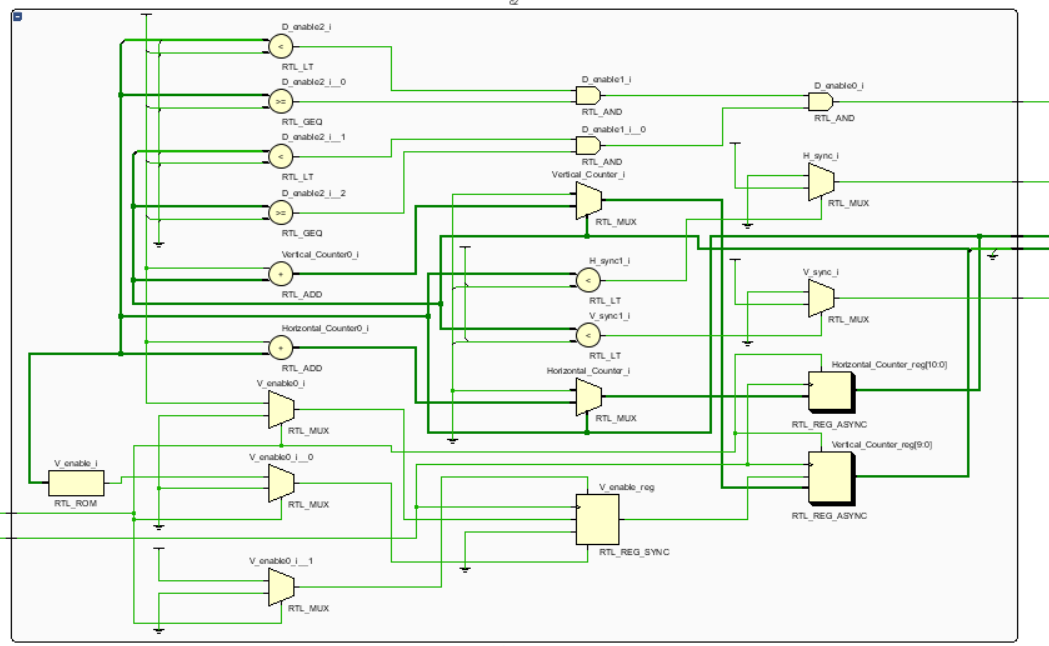
**RTL Schematic**

A screenshot of a cell phone

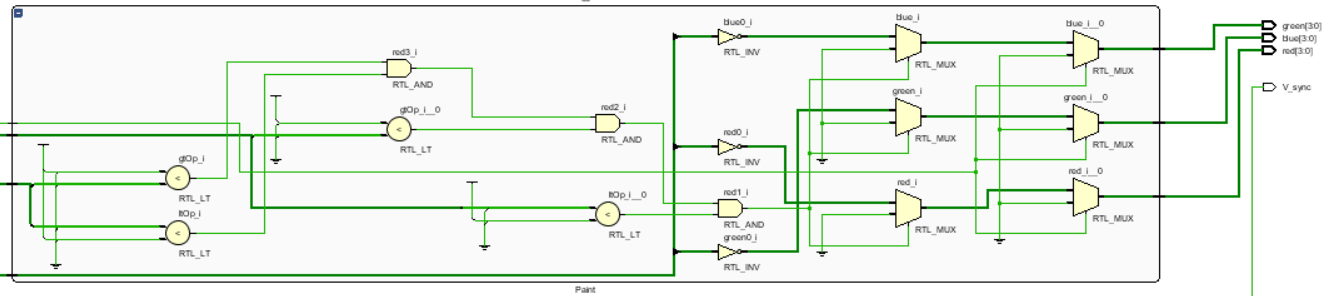
Description automatically generated

Like it is seen in the figure above my idea at the preliminary report was substantially right but I need to make some changes in the modules and their designs.

**Sync module**

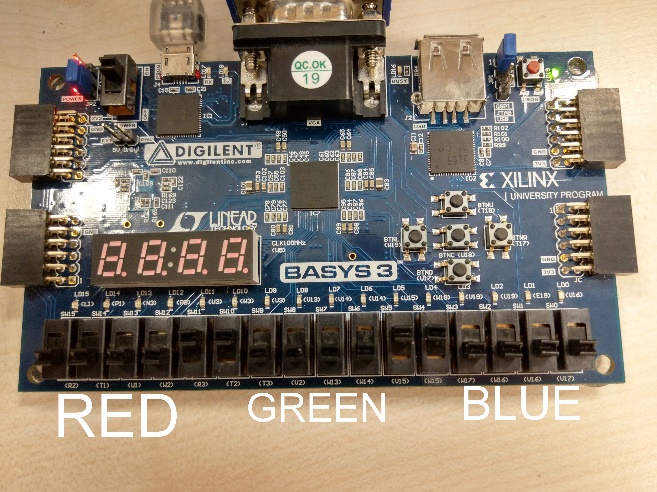


**Paint module**

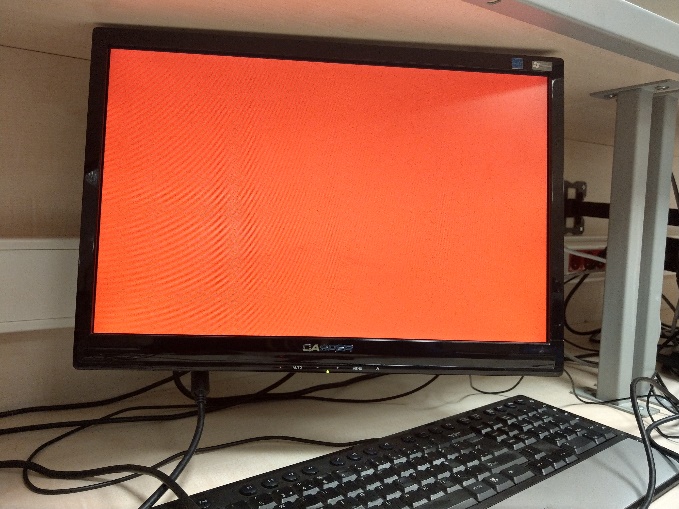
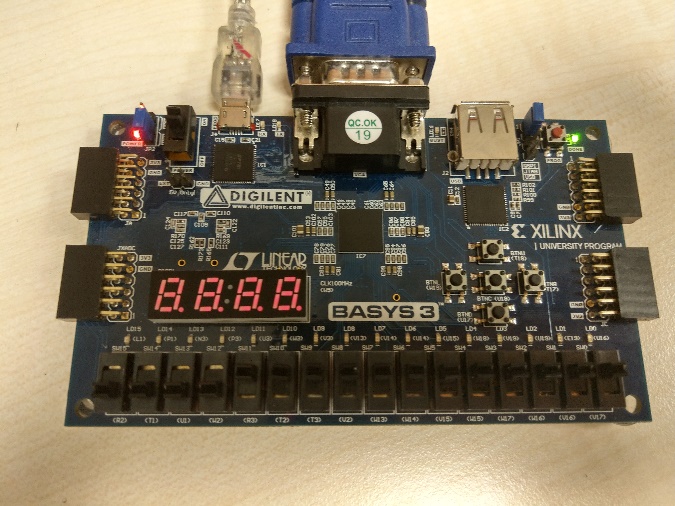


* **Screen shot of the display and Basys3**

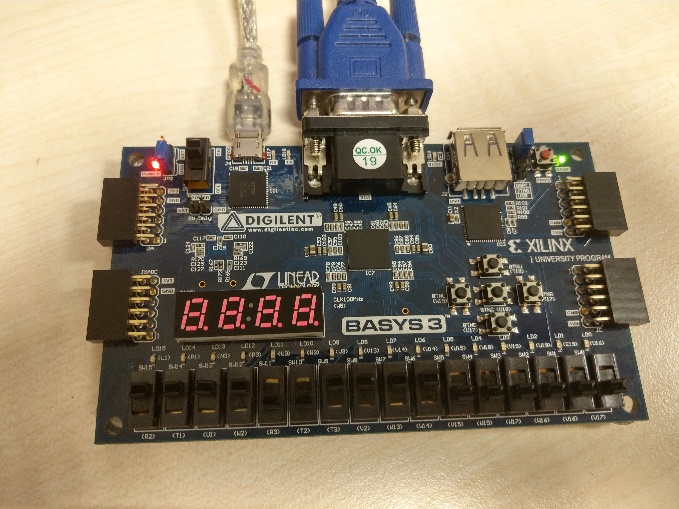
Because I want to obtain white screen at opening, I used switches as inverted. Most significant 4 switches assigned for red, 4 center switches for assigned green and least significant 4 switches for assigned blue.

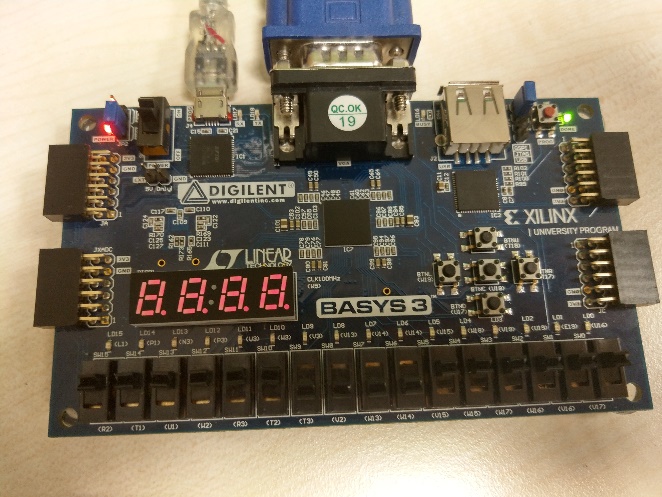
Red = “1111”, Green = “1111”, Blue = “1111”

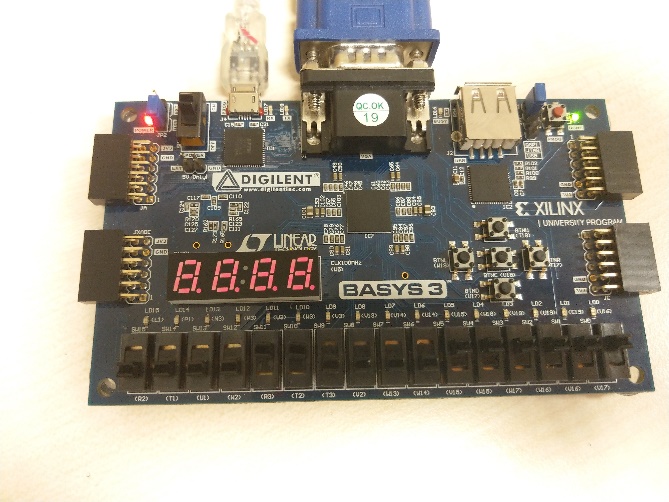
Red = “1111”, Green = “0000”, Blue = “0000”

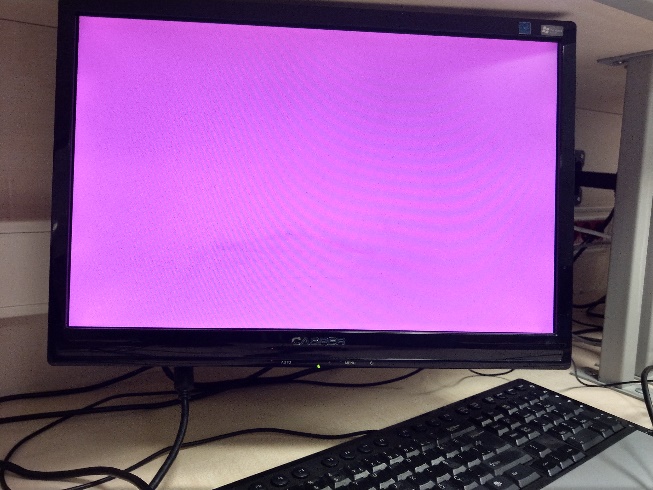
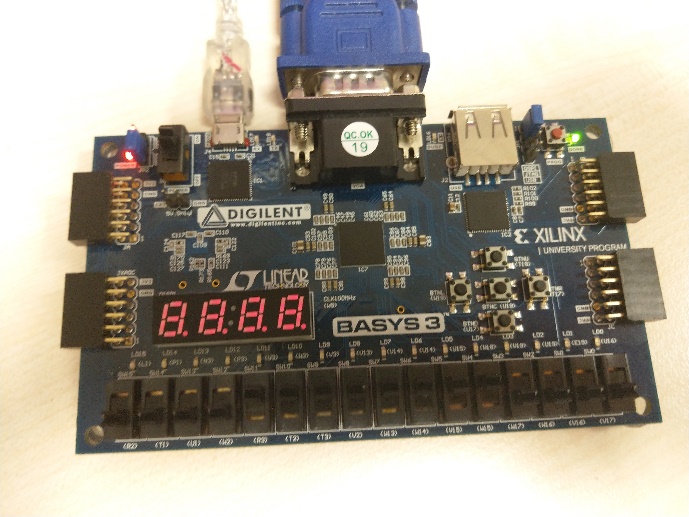
Red = “0000”, Green = “0000”, Blue = “1111”

Red = “0000”, Green = “1111”, Blue = “0000”

Red = “1111”, Green = “1111”, Blue = “0000”

Red = “1111”, Green = “0000”, Blue = “1111”

* **Conculusion**

At first I was thinking take the resolution as 1280x1024 and I did, but I did not obtain what I expected for display, it was like 2/3 part of the screen was active and rest was black and I could not change it no matter what I did with those spects, I searched on internet about spects of the screen I used and I found that it was 1440x900. It is a rare resoulution actually. I change my code respect to those spects and afterward I finally got what I wanted from the display.

* **Appendices**

**Top Module:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Top\_Module is

Port ( clk : in std\_logic;

clear\_btn : in std\_logic;

Switches : in std\_logic\_vector(11 downto 0);

H\_sync : out std\_logic;

V\_sync : out std\_logic;

red : out std\_logic\_vector(3 downto 0);

green : out std\_logic\_vector(3 downto 0);

blue : out std\_logic\_vector(3 downto 0));

end Top\_Module;

architecture Behavioral of Top\_Module is

component Synch is

Port ( clk100Mhz : in std\_logic;

clear : in std\_logic;

H\_sync : out std\_logic;

V\_sync : out std\_logic;

D\_enable : out std\_logic;

Location\_H : out std\_logic\_vector(10 downto 0);

Location\_V : out std\_logic\_vector(10 downto 0));

end component;

component Paint is

Port (D\_enable : in std\_logic;

Location\_H : in std\_logic\_vector(10 downto 0);

Location\_V : in std\_logic\_vector(10 downto 0);

Switches : in std\_logic\_vector(11 downto 0);

red : out std\_logic\_vector(3 downto 0);

blue : out std\_logic\_vector(3 downto 0);

green : out std\_logic\_vector(3 downto 0));

end component;

signal D\_enable : std\_logic;

signal clr : std\_logic;

signal Location\_H : std\_logic\_vector(10 downto 0);

signal Location\_V : std\_logic\_vector(10 downto 0);

begin

clr <= clear\_btn;

c2 : Synch port map(clk, clr, H\_sync, V\_sync, D\_enable, Location\_H, Location\_V);

c3 : Paint port map( D\_enable, Location\_H, Location\_V,Switches, red, green, blue);

end Behavioral;

**Sync:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

entity Synch is

Port ( clk100Mhz : in std\_logic;

clear : in std\_logic;

D\_enable : out std\_logic;

Location\_H : out std\_logic\_vector(10 downto 0);

Location\_V : out std\_logic\_vector(10 downto 0);

H\_sync : out std\_logic;

V\_sync : out std\_logic);

end Synch;

architecture Behavioral of Synch is

constant Pixels\_H : integer := 1904;

constant Pixels\_V : integer := 932;

constant hf\_porch : integer := 1824;

constant vf\_porch : integer := 931;

constant hb\_porch : integer := 232;

constant vb\_porch : integer := 28;

constant hs\_pulse : integer := 152;

constant vs\_pulse : integer := 3;

signal V\_enable : std\_logic;

signal Horizontal\_Counter : integer range 0 to 1904;

signal Vertical\_Counter : integer range 0 to 932;

begin

process(clk100Mhz, clear)

begin

if clear = '1' then

Horizontal\_Counter <= 0;

elsif(clk100Mhz'event and clk100Mhz = '1') then

if Horizontal\_Counter = Pixels\_H - 1 then

Horizontal\_Counter <= 0;

V\_enable <= '1';

else

Horizontal\_Counter <= Horizontal\_Counter + 1;

V\_enable <= '0';

end if;

end if;

end process;

H\_sync <= '0' when Horizontal\_Counter < 112 else '1';

process(clk100Mhz, clear)

begin

if clear = '1' then

Vertical\_Counter <= 0;

elsif(clk100Mhz'event and clk100Mhz = '1' and V\_enable = '1') then

if Vertical\_Counter = Pixels\_V - 1 then

Vertical\_Counter <= 0;

else

Vertical\_Counter <= Vertical\_Counter + 1;

end if;

end if;

end process;

V\_sync <= '0' when Vertical\_Counter < 3 else '1';

D\_enable <= '1' when (((Horizontal\_Counter < hf\_porch) and (Horizontal\_Counter >= (hb\_porch + hs\_pulse))) and ((Vertical\_Counter < vf\_porch) and (Vertical\_Counter >= (vb\_porch + vs\_pulse)))) else '0';

Location\_H <= conv\_std\_logic\_vector(Horizontal\_Counter, 11);

Location\_V <= conv\_std\_logic\_vector(Vertical\_Counter, 11);

end Behavioral;

**Paint**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity Paint is

Port (D\_enable : in std\_logic;

Location\_H : in std\_logic\_vector(10 downto 0);

Location\_V : in std\_logic\_vector(10 downto 0);

Switches : in std\_logic\_vector(11 downto 0);

red : out std\_logic\_vector(3 downto 0);

blue : out std\_logic\_vector(3 downto 0);

green : out std\_logic\_vector(3 downto 0));

end Paint;

architecture Behavioral of Paint is

begin

process(D\_enable,Location\_H,Location\_V)

begin

red <= "0000";

blue <= "0000";

green <= "0000";

if D\_enable = '1' then

if Location\_V > 30 and Location\_V < 932 and Location\_H > 384 and Location\_H < 1904 then

red <= not Switches(11 downto 8);

green <= not Switches(7 downto 4);

blue <= not Switches(3 downto 0);

end if;

end if;

end process;

end Behavioral;