Author Jasim Abbasi		Task Uppgift 8a		
Email engr.jasim@gmail.com	Approved		Version	File : VGA.VHD

TIES

VHDL Assignment

VGA Controller

Jasim Abbasi

The aim of this task is to improve, complete and validate the VGA design on VGA monitor and learn the working of Video Graphics Array (VGA)

CONTENTS

1 Introduction

The basic purpose of this task is to improve, complete and validate the VGA design on VGA monitor. The color of VGA monitor should be control by three buttons.

2 Specification

The client has an requirement to constract an embedded system that controls a screen color with three buttons. The requirement specification is to construct VGA controls, comment code, verify, test bench (before validation) and validate the code on the DE2 board.

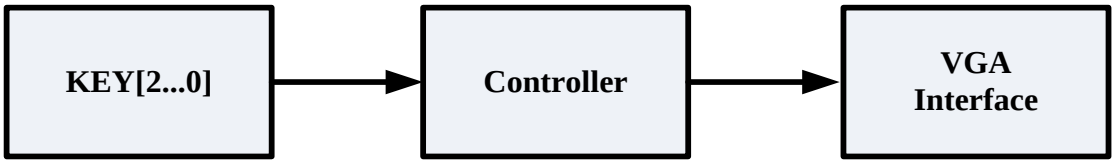


Figure 1: Schematic of the embedded system

2.1 Construction code requirements

The client has requirements that code should be written in VHDL and result should be simulate in modelsim and verify it on DE2-115 board and VGA monitor.

3 VGA (Video Graphics Array)

The VGA port has five active signals

- Three video signals for red, green and blue beams
- Hysnc signal for horizantal synchronization
- Vsync signal for vertical synchronization

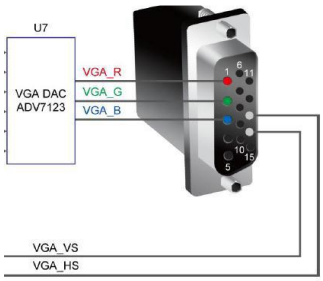


Figure 2: VGA port

The video signal is analog so that DAC(Digital to analog converter) is used to convert the digital output to the appropriate analog level. The digital output can be converted in to 2^N anlog level if the video is represented by an N-bit word.

Example: Three video signal can generate 2^{3N} different colors. Some of the color can be illustrated in the table 1

Table 1: Resulting color pattern

Red	Green	Blue	Color
0	0	0	Black
0	0	1	Blue
0	1	0	Green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White

3.1 VGA Timing

The standard VGA format monitor contains 640X480 resolutions of picture element as shown in the figure 3

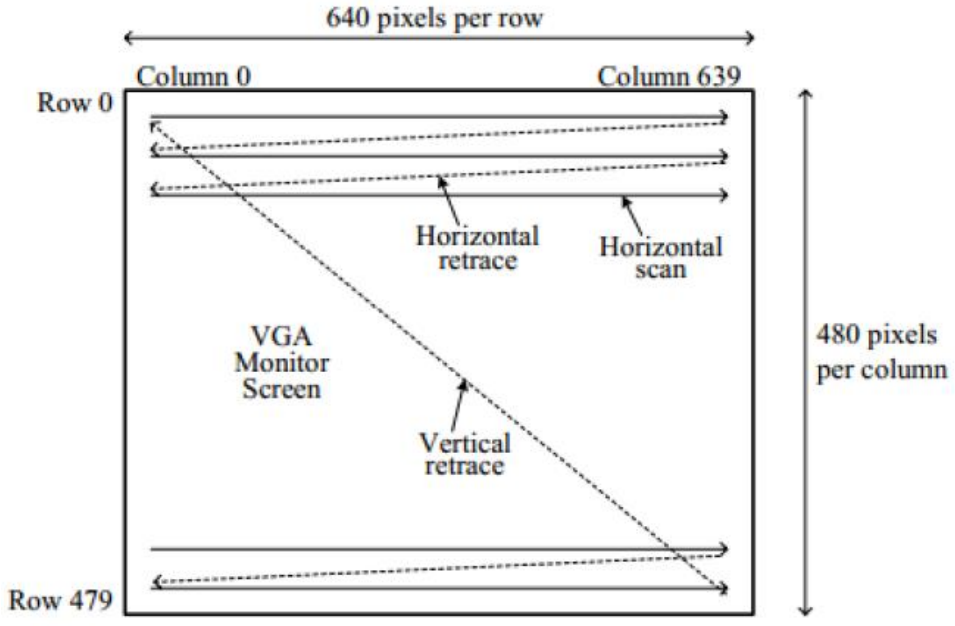


Figure 3: VGA monitor with 640X480 resolutions

The image can be display on the screen by turning **ON** and **OFF** of individual pixels. The monitor continuously scans through the entire screen at a very fast speed. The scan starts from

row 0, column 0 and moves to the right until it reaches the last column in the row. When the scan reaches the end of a row, it continues move to the the beginning of the next row. When the scan reaches the last pixel it goes back to the top left corner of the screen, and repeats the scanning process again. All the pixels are truned off during the horizontal and the vertical retraces[1].

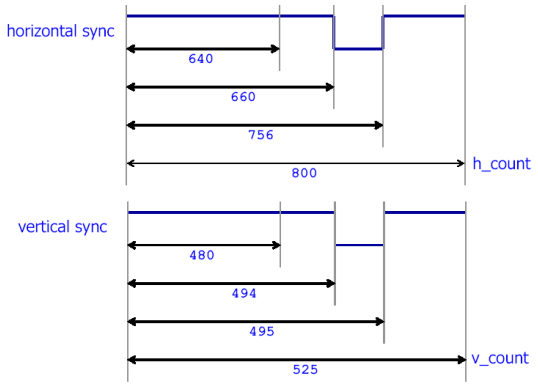


Figure 4: Horizantal Sync and Vertical sync

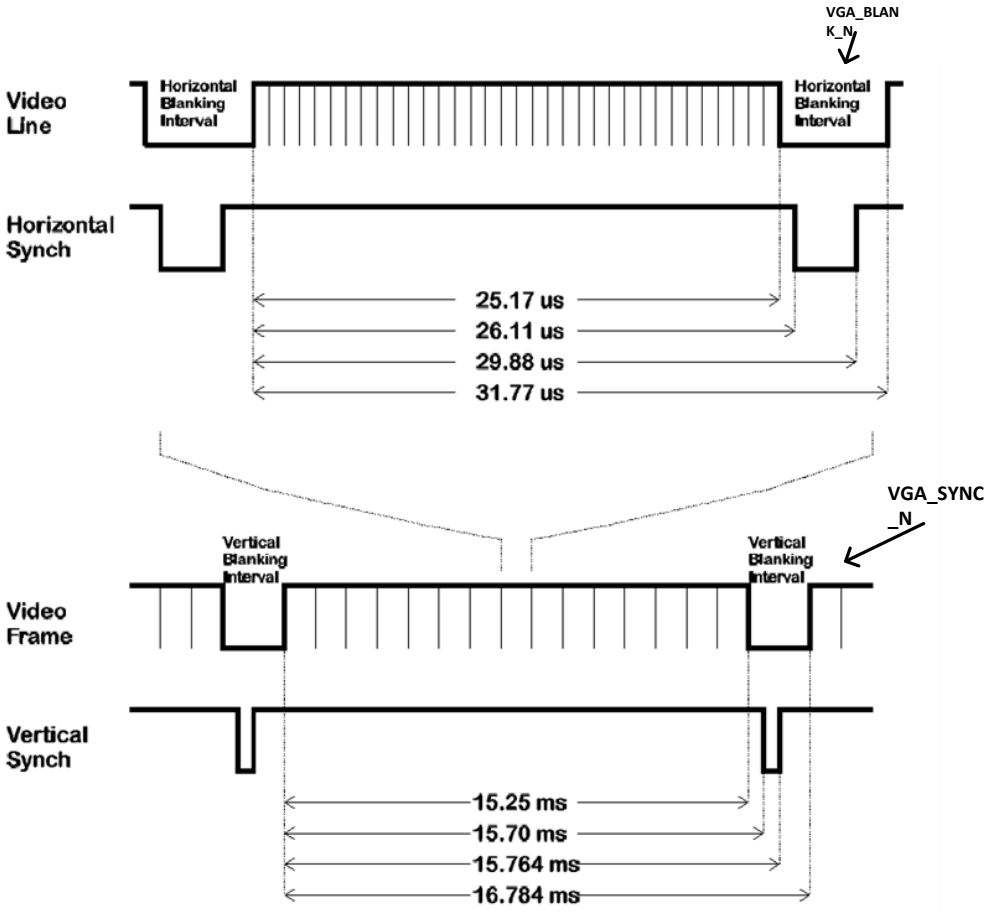


Figure 5: VGA Control Signal Timing (Protocol)

3.2 VGA Display

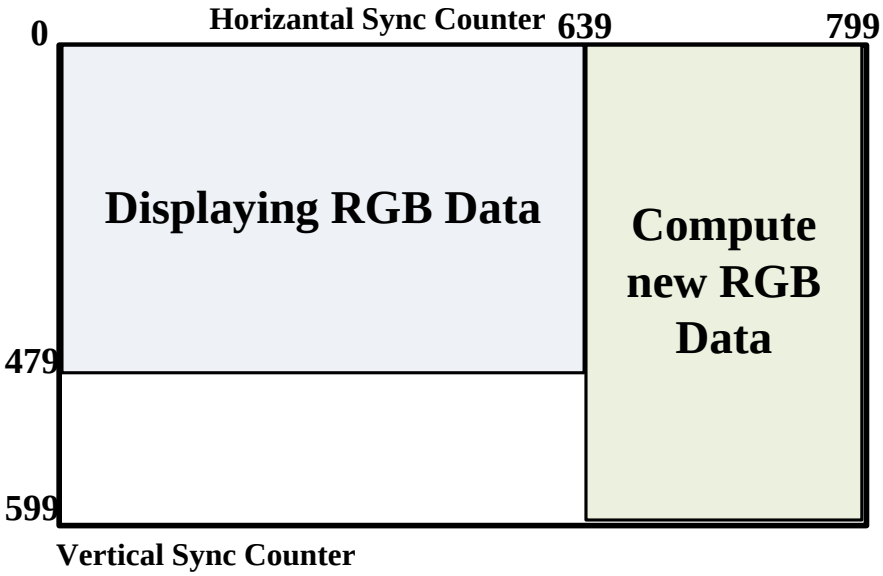
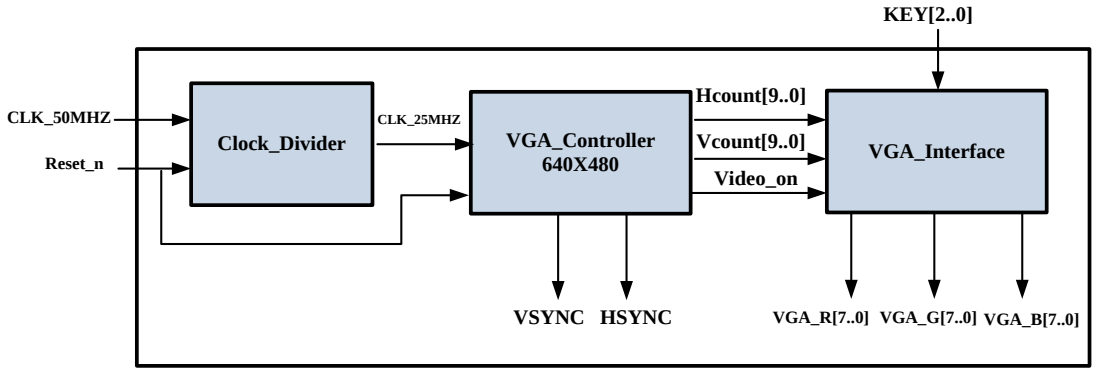


Figure 6: VGA display

3.3 VGA Component

The system has following module in it

i- Clock divider
ii- VGA Controller
iii- VGA Interface



3.3.1 Clock Divider

On the DE2-board a clock source of 50MHZ is available whereas the VGA controller requires 25 MHZ clock frequency.

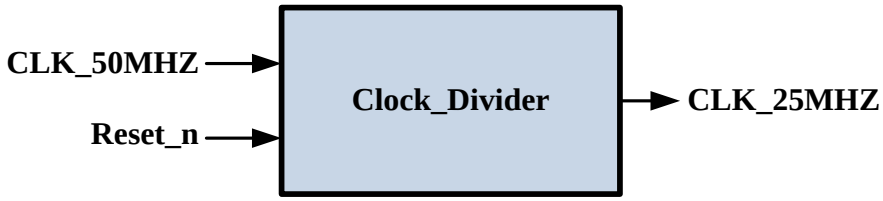


Figure: Clock Divider Component

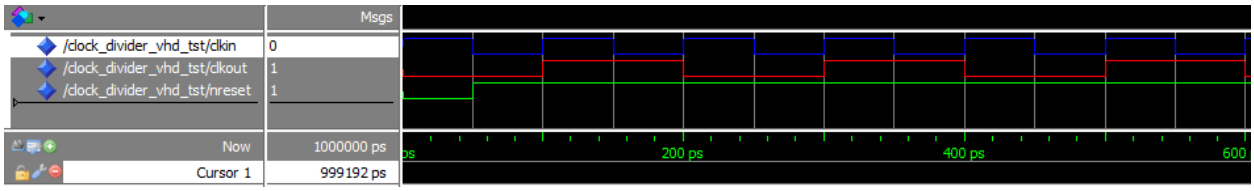


Figure: Clock Divider Result (Clkin=50MHZ(Blue signal) and Clkout=25MHZ(red signal))

3.3.2 VGA Controller

This module will generate the horizontal and vertical synchronization signals by counting the pixel clocks.

- The HSYNC should get low in the beginning of each horizontal row (96 pixel clocks) and high for the rest of time.
- The VSYNC should get low in the beginning of each frame and high for the rest of time.
- HCount is 10 bit horizontal counter that counts from 0 to 799(for one horizontal row) and then reset itself to count for next row.
- VCount is 10 bit counter that counts horizontal lines. It goes from 0 to 599 and reset itself to count the next frame.
- Video_on signal is high when video is written on the VGA display.

4 Test Protocol

Simple test bench have written to verify the working of the VGA. In this project push button is uses as a key.

Table 1: Test protocol of VGA

Case	Description	OK
Test 1	Reset_n=0	No operation
Test 2	KEY=001(DE2-Board) KEY=110(Modelsim)	Red Screen
Test 3	KEY=010((DE2-Board) KEY=101(Modelsim)	Blue Screen
Test 4	KEY=100(DE2-Board) KEY=011(Modelsim)	Green

Note: Push button work on inverted logic because of those different keys for simulation and validation are used.

Table 2: Supplementary table

KEY(Simulation)	KEY(DE2 Board)	Output
KEY=001	KEY=110	Red Screen
KEY=010	KEY=101	Green Screen
KEY=100	KEY=011	Blue Screen

5 Verifications and Validation

Result from simulation

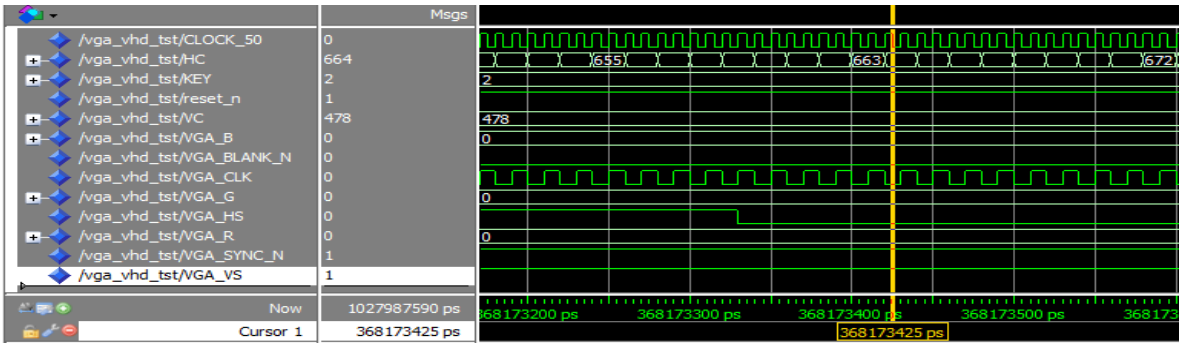


Figure : Horizontal blanking (**VGA_BLANK_N**) falling edge

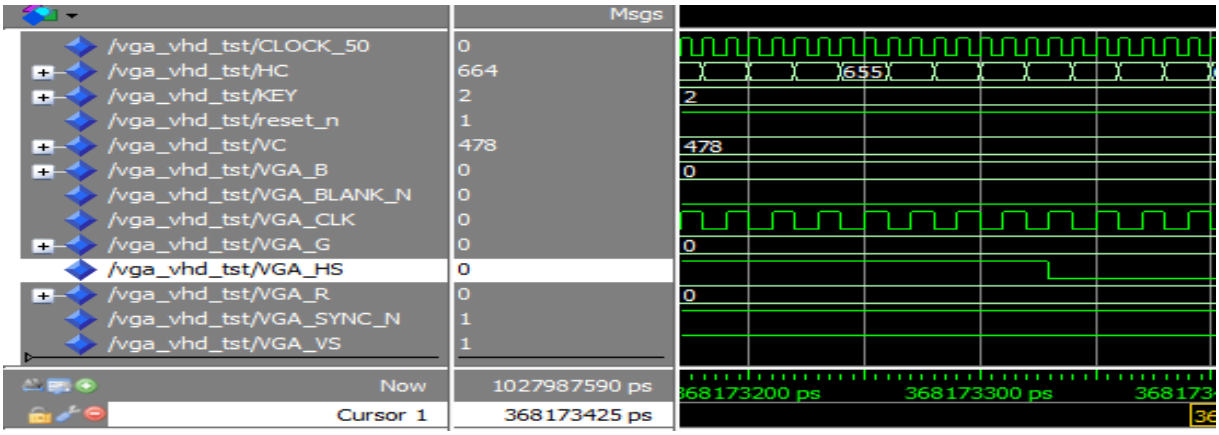


Figure: Horizontal sync (**VGA_HS)** falling edge

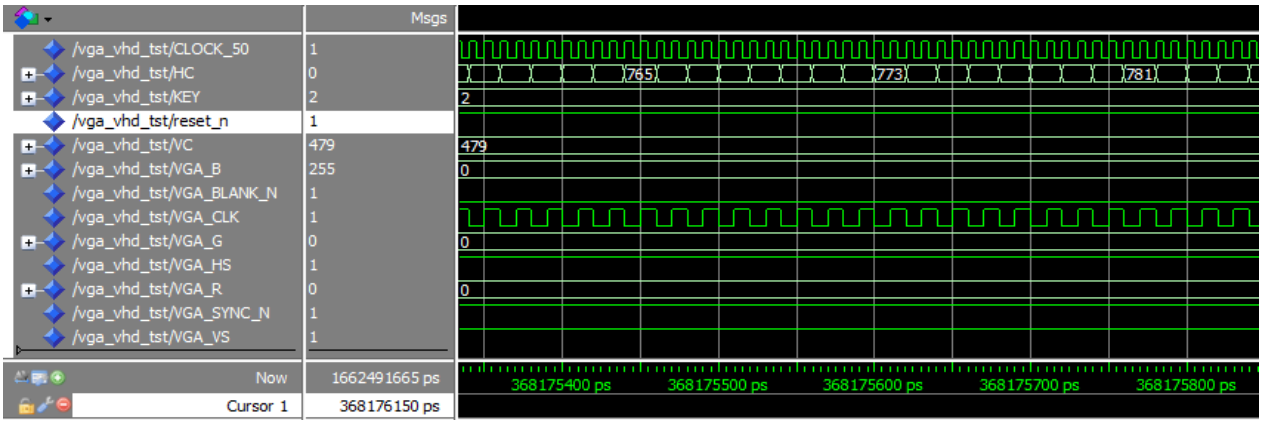
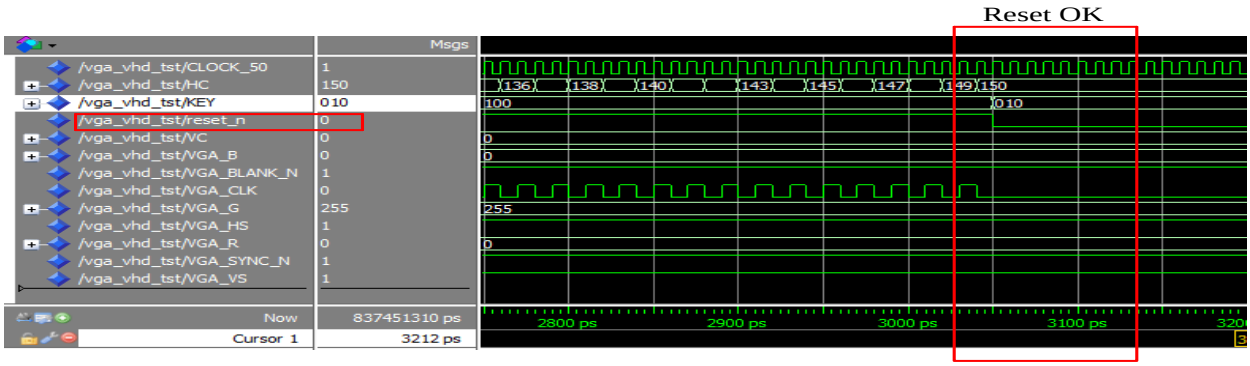
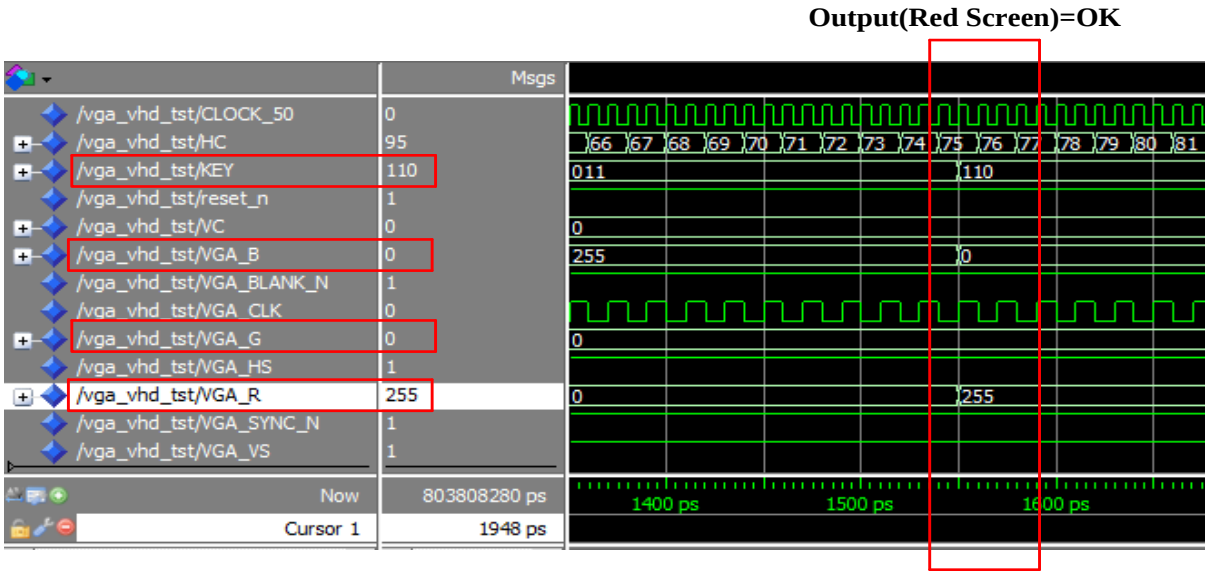


Figure: Horizontal sync (**VGA_HS)** rising edge

Case 1:

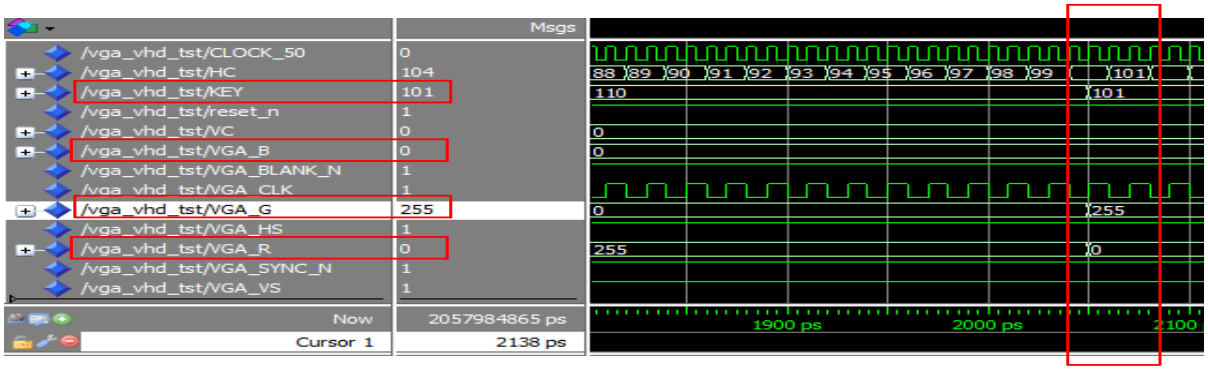


Case 2:



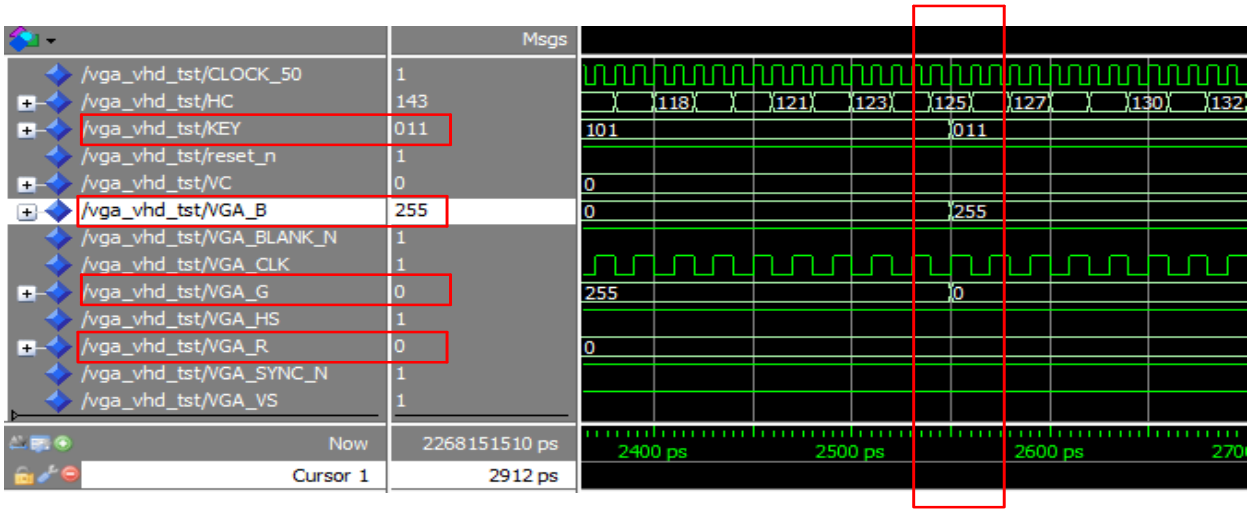
Case 3:

Output(Green Screen)=OK



Case 4:

Output(Blue Screen)=OK



Validation

Table 6: Verification and validation of VGA

Case	Description	OK	Verification Modelsim	Validation DE2-115	Validation LCD
Test 1	Reset_n=0	Black screen	OK	OK	OK
Test 2	Key=110	Red screen	OK	OK	OK
Test 3	Key=101	Green Screen	OK	OK	OK
Test 4	Key=011	Blue Screen	OK	OK	OK

6 REFERENCES

[1] http://ece.gmu.edu/coursewebpages/ECE/ECE448/S13/viewgraphs/ECE448_lecture7_VGA_1.pdf

APPENDIX

VHDL code

```
-- Company: TEIS AB

-- Engineer: Jasim Abbasi

-- Create Date: 2015 Febrary 2

-- Design Name: VGA

-- Target Devices: ALTERA Cyclone IV EP4CE115F29C7

-- Tool versions: Quartus v11 and ModelSim

-- I/O Pin Description

-- VGA_CLK: PIN_Y2

-- KEY[2..0]: N21,M21,M23

-- reset_n: Y23

-- Detail discription of the other pin assignment in the manual

library ieee;

use ieee.std_logic_1164.all;

use ieee.std_logic_unsigned.all;

USE ieee.numeric_std.all ;

entity VGA is

port( reset_n, CLOCK_50 : in std_logic;

-- till VGA enheten
```

```
 HC : out std_logic_vector(10-1 downto 0);
  VC : out std_logic_vector(10-1 downto 0);
          ----------------VGA_ Signals------------------
  VGA_HS, VGA_VS, VGA_CLK : out std_logic; -- Vertical, horizantal sync and clk signal
   VGA_BLANK_N, VGA_SYNC_N : out std_logic;
   VGA_B, VGA_G, VGA_R : out std_logic_vector(7 downto 0);
KEY : IN std_logic_vector(2 downto 0)); ---Button to control the screen color
end VGA;
architecture VGA_ARCH of VGA is

signal  HCOUNTER: integer:=0; --horizintal counter
signal  VCOUNTER: integer :=0; -- Vertical counter
Signal  H : std_logic_vector(10-1 downto 0):= "0000000000";
Signal V : std_logic_vector(10-1 downto 0) := "0000000000";
signal clk_25mhz : std_logic; -- 25 MHZ clock signal
begin
VGA_CLK <= clk_25mhz;
process_clock_25mhz:
---------  Process to generate the 25 MHZ clock-----------
process (clock_50)
    begin
                if reset_n='0' then
          clk_25mhz<= '0';
         elsif rising_edge(clock_50) then
```

```
        clk_25mhz <= not clk_25mhz;  -- 25MHZ clock for VGA
        end if;
 end process;
 ---------- End of 25 MHZ clock process----------
process_sync_screen :
process (clk_25mhz,KEY) -- Sync Process
  begin
     HC<=H;
                    VC<=V;
    if rising_edge(clk_25mhz) then
      if reset_n = '0' then  -- Async reset
                              H <= (others => '0');
        V <= (others => '0');
                             VCOUNTER<=0;
                             HCOUNTER<=0;
       Else
-- count h up
       if H >= 799 then
       H <= (others => '0');
                    HCOUNTER<=0;
       else
       H <= H + 1;   -- horizantal counter increment
                    HCOUNTER<=HCOUNTER+1;
       end if;
```

```
if (v >= 524) and (h >= 707) then
   v <= (others => '0');
elsif h = 707 then
    v <= v + 1;
                                VCOUNTER<=VCOUNTER+1; -- Vertical counter
end if;
if (H>= 0 AND H <= 639) then
   VGA_BLANK_N <= '1';
else
   VGA_BLANK_N <= '0';
end if;
if (V >= 0 AND V <= 479) then
    VGA_SYNC_N <= '1';
else
     VGA_SYNC_N <= '0';
end if;
                if (H >= 659 AND H <= 755) then
   VGA_HS<= '0';
else
    VGA_HS<= '1';
end if;
if (V>= 493 AND V<= 494) then
    VGA_VS<= '0';
else
```

```
            VGA_VS <= '1';
        end if;
  ---------- Button 1: For red screen color----------
   if (H < 640 and KEY(0)= '0') then
                VGA_R <= "11111111";
                VGA_G <= "00000000";
                        VGA_B <= "00000000";
    end if;
           ---------- Button 2: For Blue screen color----------
    if (H < 640 and KEY(1) = '0') then
         VGA_R <= "00000000";
                        VGA_G <= "11111111";
                        VGA_B <= "00000000";
    end if;
           ---------- Button 3: For Green screen color----------
   if (H < 640 and KEY(2) = '0') then
         VGA_R <= "00000000";
                VGA_G <= "00000000";
                VGA_B <= "11111111";
    end if;
  end if;
  end if;
end process process_sync_screen;
end VGA_ARCH;
```

Test Bench

```
LIBRARY ieee;

USE ieee.std_logic_1164.all;


ENTITY VGA_vhd_tst IS

END VGA_vhd_tst;

ARCHITECTURE VGA_arch OF VGA_vhd_tst IS

-- constants

-- signals

SIGNAL CLOCK_50 : STD_LOGIC;

SIGNAL HC : STD_LOGIC_VECTOR(9 DOWNTO 0);

SIGNAL KEY : STD_LOGIC_VECTOR(2 DOWNTO 0);

SIGNAL reset_n : STD_LOGIC;

SIGNAL VC : STD_LOGIC_VECTOR(9 DOWNTO 0);

SIGNAL VGA_B : STD_LOGIC_VECTOR(7 DOWNTO 0);

SIGNAL VGA_BLANK_N : STD_LOGIC;

SIGNAL VGA_CLK : STD_LOGIC;

SIGNAL VGA_G : STD_LOGIC_VECTOR(7 DOWNTO 0);

SIGNAL VGA_HS : STD_LOGIC;

SIGNAL VGA_R : STD_LOGIC_VECTOR(7 DOWNTO 0);

SIGNAL VGA_SYNC_N : STD_LOGIC;

SIGNAL VGA_VS : STD_LOGIC;

COMPONENT VGA

        PORT (
```

```
        CLOCK_50 : IN STD_LOGIC;
        HC : OUT STD_LOGIC_VECTOR(9 DOWNTO 0);
        KEY : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
        reset_n : IN STD_LOGIC;
        VC : OUT STD_LOGIC_VECTOR(9 DOWNTO 0);
        VGA_B : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
        VGA_BLANK_N : OUT STD_LOGIC;
        VGA_CLK : OUT STD_LOGIC;
        VGA_G : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
        VGA_HS : OUT STD_LOGIC;
        VGA_R : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
        VGA_SYNC_N : OUT STD_LOGIC;
        VGA_VS : OUT STD_LOGIC
        );
END COMPONENT;
BEGIN
        i1 : VGA
        PORT MAP (
-- list connections between master ports and signals
        CLOCK_50 => CLOCK_50,
        HC => HC,
        KEY => KEY,
        reset_n => reset_n,
        VC => VC,
```

```
            VGA_B => VGA_B,
            VGA_BLANK_N => VGA_BLANK_N,
            VGA_CLK => VGA_CLK,
            VGA_G => VGA_G,
            VGA_HS => VGA_HS,
            VGA_R => VGA_R,
            VGA_SYNC_N => VGA_SYNC_N,
            VGA_VS => VGA_VS
            );
clk_signal: process
   begin
            --------clOCK SIGNAL---------
        CLOCK_50<='1';
                        wait for 5 ps;
                       --wait on clk, reset;
        CLOCK_50<='0';
        wait for 5 ps;
                       --wait on clk, reset;
 end process;
init : PROCESS
-- variable declarations
BEGIN
        -- code that executes only once
          reset_n<='0';
```

```
        wait for 50 ps;
            --- Test Case 1-------
            reset_n<='1';
             key<="110";
             wait for 500 ps;
             --- Test Case 2-------
            key<="101";
             wait for 500 ps;
             --- Test Case 3-------
            key<="011";
              wait for 500 ps;
            key<="110";
             wait for 500 ps;
            key<="101";
wait for 500 ps;
            key<="011";
             wait for 500 ps;
            key<="010";
             reset_n<='1';
             wait for 250 ps;
            key<="100";
      wait for 250 ps;
            key<="001";
```

```
WAIT;

END PROCESS init;

always : PROCESS

-- optional sensitivity list

-- (        )

-- variable declarations

BEGIN

        -- code executes for every event on sensitivity list

WAIT;

END PROCESS always;

END VGA_arch;
```

Do file

```
onerror {resume}

quietly WaveActivateNextPane {} 0

add wave -noupdate /vga_vhd_tst/CLOCK_50

add wave -noupdate -radix unsigned /vga_vhd_tst/HC

add wave -noupdate /vga_vhd_tst/KEY

add wave -noupdate /vga_vhd_tst/reset_n

add wave -noupdate -radix unsigned /vga_vhd_tst/VC

add wave -noupdate -radix unsigned /vga_vhd_tst/VGA_B

add wave -noupdate /vga_vhd_tst/VGA_BLANK_N

add wave -noupdate /vga_vhd_tst/VGA_CLK

add wave -noupdate -radix unsigned /vga_vhd_tst/VGA_G

add wave -noupdate /vga_vhd_tst/VGA_HS
```

```
add wave -noupdate -radix unsigned /vga_vhd_tst/VGA_R
add wave -noupdate /vga_vhd_tst/VGA_SYNC_N
add wave -noupdate /vga_vhd_tst/VGA_VS
TreeUpdate [SetDefaultTree]
WaveRestoreCursors {{Cursor 1} {2912 ps} 0}
quietly wave cursor active 1
configure wave -namecolwidth 205
configure wave -valuecolwidth 100
configure wave -justifyvalue left
configure wave -signalnamewidth 0
configure wave -snapdistance 10
configure wave -datasetprefix 0
configure wave -rowmargin 4
configure wave -childrowmargin 2
configure wave -gridoffset 0
configure wave -gridperiod 1
configure wave -griddelta 40
configure wave -timeline 0
configure wave -timelineunits ps
update
WaveRestoreZoom {2363 ps} {3312 ps}
```