Weifan Lin

Csc343

05/07/2015

VGA

Objective:

The goal of this assignment is to gain some understanding of how video is displayed on a monitor using VGA connection. With most of the VHDL files that have provided to us, we are able to change the color of the rectangle, and move the rectangle when we test on the DE2 board.

Specifications:

VGA DE2 Circuit Inputs/Ouputs

L,R,U,D: This four inputs determine the position of the rectangle.

Move: This input triggers the rectangle to move to the position defined by the LRUD inputs

Background[2..0]: This determines the color of the monitor background.

Color_in[2..0]: This is the RGB input that will change the color of the rectangle on the display.

Clock: This is an input that is triggering the entire circuit.

Reset: The is the reset input that reset everything back to default.

Vga_rgb[2..0]: This is an output for the RGB that takes inputs other multiplexer.

Vga_blank: This is an output that connects to the video_on of the vga_sync.

Vga_Hs: This is the output that is connected to the horizontal sync of vga_sync.

Vga_Vs: This is the output that is connected to the vertical sync of vga_sync

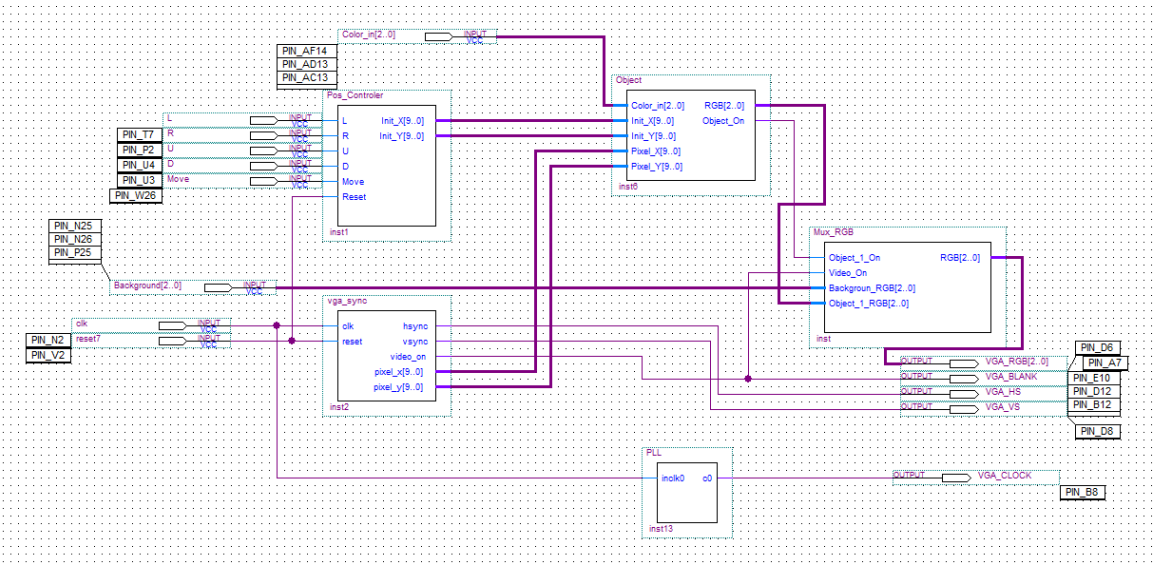
VGA_Clock: This is the output that is connected to the PLL's clock output.

Functionality:

VGA DE2 Circuit

There are five different components that are used to display VGA on the DE2 board which are the vga_sync, position controller, RGB multiplexer, object, and the phased locked loop. The main component is the vga_sync that used to define the location of x and y pixels locations, making it possible to display an image on the monitor. The position controllers determines the position of the object on the monitor using L,R,U,D inputs. The RGB multiplexer chooses whether to route the object's RGB to the RGB output or the background's RGB to the RGB output. The object displays a rectangle on the monitor, whose size can be altered. While the RGB output is capable of change the color of the object at that specific location. Finally component is the PLL which is responsible for generating a 25 MHz clock signal for the VGA outputs on the DE2 board.

Design:



Above figure demonstrated the final circuit composted of five components that is used to test on the DE2 board and monitor.

Simulation:

The simulation of the circuit on the DE2 board was successful. You were able to change the background color using first three buttons. Then you are able to change the RGB values of the rectangle. Next, we will try to change the color of the rectangle, while keeping the background color constant. After that, we will try to change the position of the rectangle denoted as Left, Right, UP, Down. The last part is to press reset button that change everything to default.

Conclusion:

The overall of the lab was very straight forward and simple. Since the majority of the components needed to test on the DE2 board were already given, there are only few things

that we need to modify: namely, adding an RGB inputs to the object component that would let us change the color of the displayed rectangle, and created a PLL vhdl file that use to divide the DE2 board's 50 MHz clock down to 25 MHz.

Appendix:

Pin Assignments:
To, Location
Clock, PIN_N2
Reset, PIN_V2
Background[0], PIN_N25
Background[1], PIN_N26
Background[2], PIN_P25
R, PIN_P2
L, PIN_T7
D, PIN_U3
U, PIN_U4
VGA_HS,PIN_A7
VGA_VS,PIN_D8
VGA_RGB[0],PIN_E10
VGA_RGB[1],PIN_D12
VGA_RGB[2],PIN_B12
VGA_Clock, PIN_B8
VGA_Blank, PIN_D6
Move,PIN_W26
Object_Color[0], PIN_AF14
Object_Color[1], PIN_AD13
Object_Color[2], PIN_AC13

VGA_SYNC VHDL:

```
library ieee;
use ieee.std_logic_1164.all ;
use ieee.numeric_std.all ;
entity vga_sync is
port(
clk, reset: in std_logic;
hsync , vsync : out std_logic ;
video_on: out std_logic;
pixel_x , pixel_y : out std_logic_vector (9 downto 0)
);
```

```
end vga_sync ;

architecture arch of vga_sync is
constant HD: integer :=640; --horizontal display area
constant HF: integer:=16 ; --h. front porch
constant HB: integer:=48 ; --h. back porch
constant HR: integer:=96 ; --h. retrace
constant VD: integer :=480; --vertical display area
constant VF: integer:= 11; -- v. front porch
constant VB: integer :=31; -- v. back porch
constant VR: integer :=2; -- v. retrace

signal current_mod2, next_mod2 : std_logic;
signal current_v_count , next_v_count : unsigned(9 downto 0) ;
signal current_h_count , next_h_count : unsigned (9 downto 0) ;
signal current_v_sync , current_h_sync : std_logic ;
signal next_v_sync , next_h_sync : std_logic;
signal h_end , v_end , pixel_tick: std_logic;

begin
process (clk,reset)
begin
if (reset = '1') then
current_mod2 <= '0';
current_v_count <= (others=>'0');
current_h_count <= (others=>'0');
current_v_sync <= '0';
current_h_sync <= '0';
elsif (clk'event and clk = '1') then
current_mod2 <= next_mod2 ;
current_v_count <= next_v_count;
current_h_count <= next_h_count;
current_v_sync <= next_v_sync ;
current_h_sync <= next_h_sync ;
end if ;
end process;

next_mod2 <= not current_mod2;
pixel_tick <= '1' when current_mod2='1' else '0';
h_end <= '1' when current_h_count=(HD+HF+HB+HR - 1) else --799
                '0';
v_end <= '1' when current_v_count=(VD+VF+VB+VR - 1) else --524
                '0';
process (current_h_count,h_end,pixel_tick)
```

```
begin

if pixel_tick = '1' then
if h_end='1' then
next_h_count <= (others=>'0');
else
next_h_count <= current_h_count + 1;
end if ;
else
next_h_count <= current_h_count;
end if ;
end process;
process (current_v_count,h_end,v_end,pixel_tick)
begin

if pixel_tick='1' and h_end='1' then
if (v_end='1') then
        next_v_count <= (others=>'0');
else
        next_v_count <= current_v_count + 1;
end if ;
else
        next_v_count <= current_v_count;
end if ;
end process;

next_h_sync <= '1' when (current_h_count >=(HD+HF)) --656
                        and (current_h_count<=(HD+HF+HR-1)) else --751
                        '0';

video_on <= '1' when (current_h_count<HD) and (current_v_count<VD) else
                        '0';
next_v_sync <= '1' when ( current_v_count >= ( VD+VF ) ) --490
                                and (current_v_count<=(VD+VF+VR-1)) else --491
                                '0';

hsync <= current_h_sync ;
vsync <= current_v_sync ;
pixel_x <= std_logic_vector(current_h_count);
pixel_y <= std_logic_vector(current_v_count);
end arch;
```

Position Controller VHDL:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity Pos_Controler is
	port
	(
		L, R, U, D, Move, Reset: in std_logic;
		Init_X, Init_Y: out std_logic_vector(9 downto 0)
	);
end Pos_Controler;

architecture arch of Pos_Controler is

signal x, y: unsigned(9 downto 0):= "0011001000";

begin

Init_X <= std_logic_vector(x);
Init_Y <= std_logic_vector(Y);


Process(Move, reset)
begin
if(reset = '1') then
	x <= "0011001000";
	y <="0011001000";
	elsif(Falling_Edge(Move)) then
		if(U = '1' and (y>10)) then
			y <= y - "0000001010";
		end if;
		if(D = '1' and (y < (416 - 10) )) then
			y <= y + "0000001010";
		end if;
		if(L = '1' and (x>10)) then
			x <= x - "0000001010";
		end if;
		if(R = '1' and (x < (432 - 10))) then
			x <= x + "0000001010";
		end if;
	end if;
end process;
end arch;
```

RGB Multiplexer VHDL:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity Mux_RGB is
	port
	(
		Object_1_On, Video_On: in std_logic;
		Backgroun_RGB, Object_1_RGB : in std_logic_vector(2 downto 0);
		RGB : out std_logic_vector(2 downto 0)
	);
end Mux_RGB;

architecture arch of Mux_RGB is

signal r : std_logic_vector(2 downto 0);

begin

RGB <= Backgroun_RGB when Object_1_On = '0' and Video_On = '1' else
	Object_1_RGB when Object_1_On = '1' and Video_On = '1' else
	"000";

end arch;
```

Object VHDL:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity Object is
	port
	(
		Color_in : in std_logic_vector(2 downto 0);
		Init_X, Init_Y: in std_logic_vector(9 downto 0);
		Pixel_X, Pixel_Y: in std_logic_vector(9 downto 0);
		RGB : out std_logic_vector(2 downto 0);
		Object_On: out std_logic
	);
end Object;

architecture arch of Object is

Constant W : unsigned(9 downto 0):= "0011010000"; --208
Constant H : unsigned(9 downto 0):= "0001000000"; --64
```

```
signal Color : std_logic_vector(2 downto 0):= "010"; --Green (You can change this if you like)

begin

process(Pixel_X, Pixel_Y)
begin
	Color <= Color_in;
	RGB <= Color;
	if((Pixel_X >= Init_X) and (Pixel_Y >= Init_Y) and( unsigned(Pixel_X) < (unsigned(Init_X) + W)) and (unsigned(Pixel_Y) < (unsigned(Init_Y) + H)) ) then
		Object_On <= '1';
	else
		Object_On <= '0';
	end if;
end process;
end arch;
```

PLL VHDL:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
LIBRARY altera_mf;
USE altera_mf.all;
ENTITY PLL IS
PORT
(
inclk0 : IN STD_LOGIC := '0';
c0 : OUT STD_LOGIC
);
END PLL;
ARCHITECTURE SYN OF pll IS
SIGNAL sub_wire0 : STD_LOGIC_VECTOR (5 DOWNTO 0);
SIGNAL sub_wire1 : STD_LOGIC ;
SIGNAL sub_wire2 : STD_LOGIC ;
SIGNAL sub_wire3 : STD_LOGIC_VECTOR (1 DOWNTO 0);
SIGNAL sub_wire4_bv : BIT_VECTOR (0 DOWNTO 0);
SIGNAL sub_wire4 : STD_LOGIC_VECTOR (0 DOWNTO 0);
COMPONENT altpll
GENERIC (
clk0_divide_by : NATURAL;
clk0_duty_cycle : NATURAL;
clk0_multiply_by : NATURAL;
clk0_phase_shift : STRING;
compensate_clock : STRING;
```

inclk0_input_frequency : NATURAL;
intended_device_family : STRING;
lpm_hint : STRING;
lpm_type : STRING;
operation_mode : STRING;
port_activeclock : STRING;
port_areset : STRING;
port_clkbad0 : STRING;
port_clkbad1 : STRING;
port_clkloss : STRING;
port_clkswitch : STRING;
port_configupdate : STRING;
port_fbin : STRING;
port_inclk0 : STRING;
port_inclk1 : STRING;
port_locked : STRING;
port_pfdena : STRING;
port_phasecounterselect : STRING;
port_phasedone : STRING;
port_phasestep : STRING;
port_phaseupdown : STRING;
port_pllena : STRING;
port_scanaclr : STRING;
port_scanclk : STRING;
port_scanclkena : STRING;
port_scandata : STRING;
port_scandataout : STRING;
port_scandone : STRING;
port_scanread : STRING;
port_scanwrite : STRING;
port_clk0 : STRING;
port_clk1 : STRING;
port_clk2 : STRING;
port_clk3 : STRING;
port_clk4 : STRING;
port_clk5 : STRING;
port_clkena0 : STRING;
port_clkena1 : STRING;
port_clkena2 : STRING;
port_clkena3 : STRING;
port_clkena4 : STRING;
port_clkena5 : STRING;
port_extclk0 : STRING;
port_extclk1 : STRING;

```
port_extclk2 : STRING;
port_extclk3 : STRING
);
PORT (
inclk : IN STD_LOGIC_VECTOR (1 DOWNTO 0);
clk : OUT STD_LOGIC_VECTOR (5 DOWNTO 0)
);
END COMPONENT;
BEGIN
sub_wire4_bv(0 DOWNTO 0) <= "0";
sub_wire4 <= To_stdlogicvector(sub_wire4_bv);
sub_wire1 <= sub_wire0(0);
c0 <= sub_wire1;
sub_wire2 <= inclk0;
sub_wire3 <= sub_wire4(0 DOWNTO 0) & sub_wire2;
altpll_component : altpll
GENERIC MAP (
clk0_divide_by => 2,
clk0_duty_cycle => 50,
clk0_multiply_by => 1,
clk0_phase_shift => "0",
compensate_clock => "CLK0",
inclk0_input_frequency => 20000,
intended_device_family => "Cyclone II",
lpm_hint => "CBX_MODULE_PREFIX=PLL",
lpm_type => "altpll",
operation_mode => "NORMAL",
port_activeclock => "PORT_UNUSED",
port_areset => "PORT_UNUSED",
port_clkbad0 => "PORT_UNUSED",
port_clkbad1 => "PORT_UNUSED",
port_clkloss => "PORT_UNUSED",
port_clkswitch => "PORT_UNUSED",
port_configupdate => "PORT_UNUSED",
port_fbin => "PORT_UNUSED",
port_inclk0 => "PORT_USED",
port_inclk1 => "PORT_UNUSED",
port_locked => "PORT_UNUSED",
port_pfdena => "PORT_UNUSED",
port_phasecounterselect => "PORT_UNUSED",
port_phasedone => "PORT_UNUSED",
port_phasestep => "PORT_UNUSED",
port_phaseupdown => "PORT_UNUSED",
port_pllena => "PORT_UNUSED",
```

```
port_scanaclr => "PORT_UNUSED",
port_scanclk => "PORT_UNUSED",
port_scanclkena => "PORT_UNUSED",
port_scandata => "PORT_UNUSED",
port_scandataout => "PORT_UNUSED",
port_scandone => "PORT_UNUSED",
port_scanread => "PORT_UNUSED",
port_scanwrite => "PORT_UNUSED",
port_clk0 => "PORT_USED",
port_clk1 => "PORT_UNUSED",
port_clk2 => "PORT_UNUSED",
port_clk3 => "PORT_UNUSED",
port_clk4 => "PORT_UNUSED",
port_clk5 => "PORT_UNUSED",
port_clkena0 => "PORT_UNUSED",
port_clkena1 => "PORT_UNUSED",
port_clkena2 => "PORT_UNUSED",
port_clkena3 => "PORT_UNUSED",
port_clkena4 => "PORT_UNUSED",
port_clkena5 => "PORT_UNUSED",
port_extclk0 => "PORT_UNUSED",
port_extclk1 => "PORT_UNUSED",
port_extclk2 => "PORT_UNUSED",
port_extclk3 => "PORT_UNUSED"
)
PORT MAP (
inclk => sub_wire3,
clk => sub_wire0
);
END SYN;
```