Lab 8: VGA

Brandon Chin

CSC343 - Instructor: Prof. Izidor Gertner

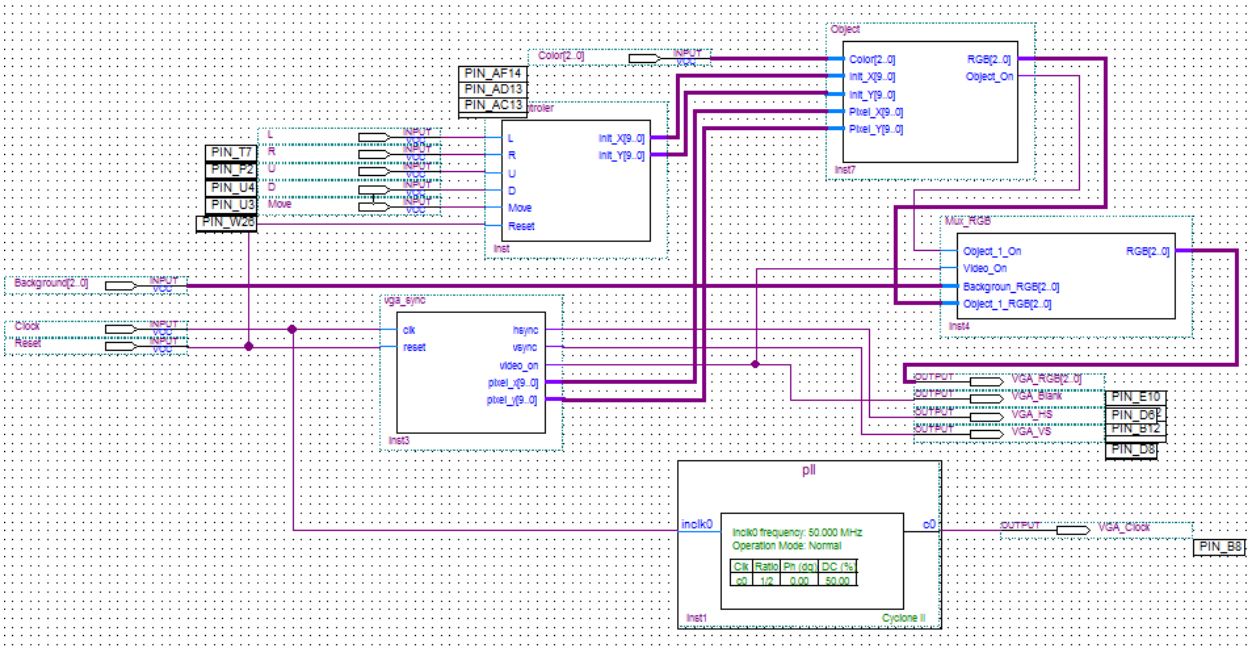
4/7/2015

Objective

In this lab we will gain an understanding on how video is displayed on a monitor using VGA connection. We will do this by implementing a circuit that displays an width x height rectangle and allows us to move the position of the rectangle and change the color of both the rectangle and the background. In the end, we will verify our design by testing it on a monitor with VGA connection.

Functionality and Specifications

Below is the top level circuit design of the VGA:



VGA_Sync:

- Main Component
- A counter
- Inputs: Clk, Reset
- Outputs: hsync, vsync, video_on, pixel_x[9], pixel_y[9]
- pixel_x and pixel_y indicate the location of the current pixel (range from (0,0) to (640,480) and changes every 2 clock cycles)
- The RGB color for every pixel has to be specified

```
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';
```

The horizontal sync is high when the current horizontal counter is between 656 and 751, which is the horizontal screen and the horizontal front porch. The vertical sync is high when the current vertical counter is between 490 and 491. Video is on when the current horizontal and vertical counters are within the bound of the horizontal and vertical screen.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity vga_sync is
6  port(
7      clk, reset: in std_logic;
8      hsync , vsync : out std_logic ;
9      video_on: out std_logic;
10     pixel_x , pixel_y : out std_logic_vector (9 downto 0)
11 );
12 end vga_sync;
13
14 architecture arch of vga_sync is
15 constant HD: integer := 640; --horizontal display area
16 constant HF: integer:= 16 ; --h. front porch
17 constant HB: integer:= 48 ; --h. back porch
18 constant HR: integer:= 96 ; --h. retrace
19 constant VD: integer := 480; --vertical display area
20 constant VF: integer:= 11; -- v. front porch
21 constant VB: integer := 31; -- v. back porch
22 constant VR: integer := 2; -- v. retrace
23
24 signal current_mod2, next_mod2 : std_logic;
25 signal current_v_count , next_v_count : unsigned(9 downto 0);
26 signal current_h_count , next_h_count : unsigned (9 downto 0);
27 signal current_v_sync , current_h_sync : std_logic;
28 signal next_v_sync , next_h_sync : std_logic;
29 signal h_end , v_end , pixel_tick: std_logic;
```

```
31 begin
32     process (clk,reset)
33     begin
34     if (reset = '1') then
35     current_mod2 <= '0';
36     current_v_count <= (others=>'0');
37     current_h_count <= (others=>'0');
38     current_v_sync <= '0';
39     current_h_sync <= '0';
40     elsif (clk'event and clk = '1') then
41     current_mod2 <= next_mod2 ;
42     current_v_count <= next_v_count;
43     current_h_count <= next_h_count;
44     current_v_sync <= next_v_sync ;
45     current_h_sync <= next_h_sync ;
46     end if ;
47 end process;
```

```
49   next_mod2 <= not current_mod2;
50   pixel_tick <= '1' when current_mod2='1' else '0';
51   h_end <= '1' when current_h_count=(HD+HF+HB+HR - 1) else --799
52   '0';
53   v_end <= '1' when current_v_count=(VD+VF+VB+VR - 1) else --524
54   '0';
55  process (current_h_count,h_end,pixel_tick)
56   begin
57  if pixel_tick = '1' then
58  if h_end='1' then
59   next_h_count <= (others=>'0');
60  else
61   next_h_count <= current_h_count + 1;
62   end if ;
63  else
64   next_h_count <= current_h_count;
65   end if ;
66   end process;
67  process (current_v_count,h_end,v_end,pixel_tick)
68   begin
69  if pixel_tick='1' and h_end='1' then
70  if (v_end='1') then
71   next_v_count <= (others=>'0');
72  else
73   next_v_count <= current_v_count + 1;
74   end if ;
75  else
76   next_v_count <= current_v_count;
77   end if ;
78   end process;
```

```
79   next_h_sync <= '1' when (current_h_count >=(HD+HF)) --656
80   and (current_h_count<=(HD+HF+HR-1)) else --751
81   '0';
82   video_on <= '1' when (current_h_count<HD) and
83   (current_v_count<VD) else
84   '0';
85   next_v_sync <= '1' when ( current_v_count >= ( VD+VF ) ) --490
86   and (current_v_count<=(VD+VF+VR-1)) else --491
87   '0';
88   hsync <= current_h_sync;
89   vsync <= current_v_sync;
90   pixel_x <= std_logic_vector(current_h_count);
91   pixel_y <= std_logic_vector(current_v_count);
92   end arch;
```

Mux_RGB:

- A multiplexer
- Inputs: Object_1_On, Video_On, Background_RGB[2], Object_1_RGB[2]
- Outputs: RGB[2]
- Chooses which object's RGB signal is to be round to the RGB output (*ie* object or background)

```
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";
```

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4  entity Mux_RGB is
5  port
6  (
7  Object_1_On, Video_On: in std_logic;
8  Backgroun_RGB, Object_1_RGB : in std_logic_vector(2 downto 0);
9  RGB : out std_logic_vector(2 downto 0)
10 );
11 end Mux_RGB;
12 architecture arch of Mux_RGB is
13 signal r : std_logic_vector(2 downto 0);
14 begin
15 --Enter your code here
16 process(Object_1_On, Video_On)
17 begin
18 if(Video_On = '1') then
19 if(Object_1_On = '1') then
20 RGB <= Object_1_RGB;
21 else
22 RGB <= Backgroun_RGB;
23 end if;
24 else
25 RGB <= "000";
26 end if;
27 end process;
28 end arch;
```

Object:

- Inputs: color[2], init_x[9], init_y[9], pixel_x[9], pixel_y[9]
- Outputs: RGB[2], object_on
- Displays a w x h rectangle
- When the current location is within the region of the object (initial position and size of the object) it sets the RGB output to the color of the object and also sets the object_on signal high.
- Every 2 cycles (when the pixel position from the vga_sync changes) it draws a pixel of the rectangle starting from the initial (x,y) to initial (x + w, y + h).

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity Object is
6  port
7  (
8  Color: in std_logic_vector(2 downto 0);
9  Init_X, Init_Y: in std_logic_vector(9 downto 0);
10 Pixel_X, Pixel_Y: in std_logic_vector(9 downto 0);
11 RGB : out std_logic_vector(2 downto 0);
12 Object_On: out std_logic
13 );
14 end Object;
15
16 architecture arch of Object is
17 Constant W : unsigned(9 downto 0):= "0011010000"; --208
18 Constant H : unsigned(9 downto 0):= "0001000000"; --64
19
20 begin
21 process(Pixel_X, Pixel_Y)
22 begin
23 RGB <= Color;
24 if(
25      (Pixel_X >=init_X) and
26      (pixel_Y>=init_Y) and
27      (unsigned (Pixel_X)<(unsigned(init_x)+w)) and
28      (unsigned(pixel_y)<(unsigned (init_Y)+H)) )then
29      Object_On <= '1';
30 else
31      object_On <='0';
32 end if;
33 end process;
34 end arch;
```

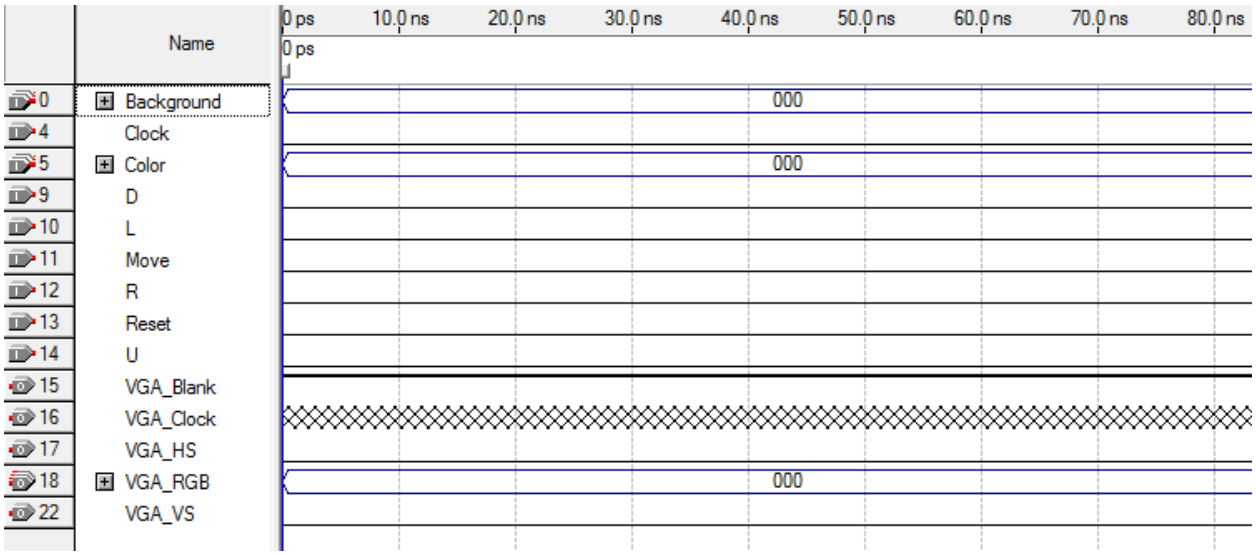
Position_Controller:

- Inputs: L, R, U, D, Move, Reset
- Outputs: init_x[9], init_y[9]
- Defines the position of the object (when the move pushbutton is pressed, depending on the configuration of the switches, left, right, up, down, the object should move in one of the 8 possible directions).
- The object should always stay inside the monitor's frame.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity Pos_Controler is
6  port
7  (
8      L, R, U, D, Move, Reset: in std_logic;
9      Init_X, Init_Y: out std_logic_vector(9 downto 0)
10 );
11 end Pos_Controler;
12
13 architecture arch of Pos_Controler is
14 signal x, y: unsigned(9 downto 0):= "0011001000";
15 begin
16 Init_X <= std_logic_vector(x);
17 Init_Y <= std_logic_vector(Y);
18 Process(Move, reset)
19
20 begin
21
22 if (reset='1') then
23      x<="0011001000";
24      y<="0011001000";
```

```
26  elsif(Falling_edge(move)) then
27      if (U='1' and (y>10)) then
28      y<=y-"0000001000";
29      end if;
30
31      if (D='1' and (y < (416-10))) then
32          y<=y+"0000001000";
33      end if;
34
35      if (L='1' and (x>10)) then
36          x<=x-"0000001000";
37      end if;
38
39      if (R='1' and (x < (432-10))) then
40          x<=x+"0000001000";
41      end if;
42  end if;
43  end process;
44  end arch;
```

Simulation



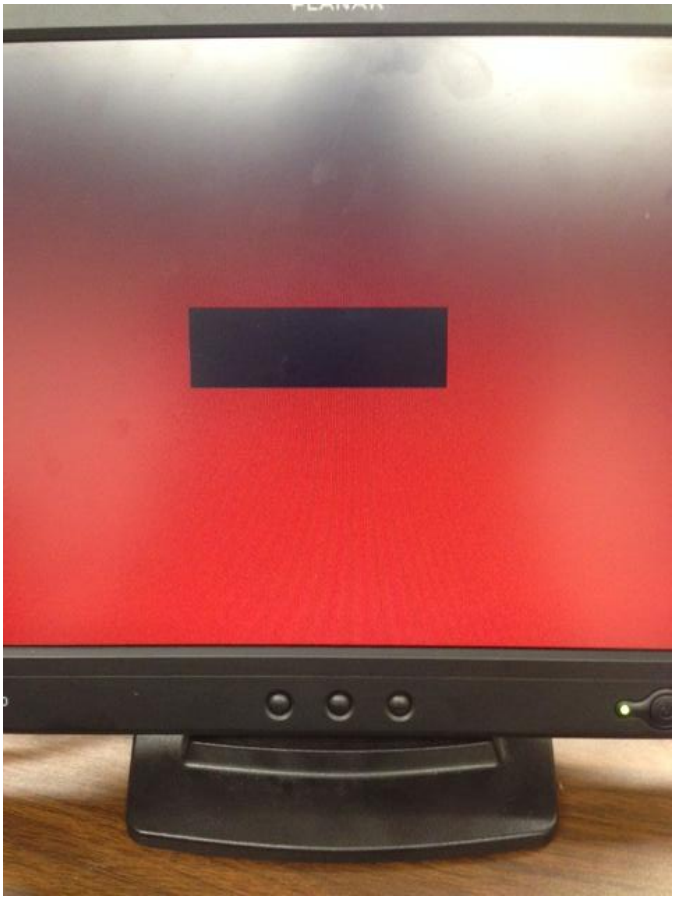
DE2 Circuit Board and Monitor Test

Before connecting to the DE2 board, we must first assign the correct pins to each component of the circuit.

Pin assignments text file:

```
To, Location
Clock, PIN_N2
Reset, PIN_V2
Background[0],PIN_N25
Background[1],PIN_N26
Background[2],PIN_P25
Color[0],PIN_AF14
Color[1],PIN_AD13
Color[2],PIN_AC13
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
```

Now we can begin board testing:



Conclusion

We designed a circuit which connects to a monitor via VGA and displays a rectangle that the user can move or change its color. The user can set the movement direction through the DE2 Circuit Board switches, and presses the pushbutton to execute the movement in the specified direction. The user can also change the color switches to change the rgb values of each pixel to change the color of either the rectangle or the background. this was all verified through testing and waveform simulations.