Display Logic Curve with ROM VHDL Code

Logic Systems And Processors (České Vysoké Učení Technické v Praze)

```
1
2
3   library ieee;
4   use ieee.std_logic_1164.all;
5   use ieee.numeric_std.all;       -- type integer and unsigned
6
7   entity DisplayLogicWithRom is
8   port(
9          yrow, xcolumn : in unsigned(9 downto 0); -- row and  column indexes of VGA video
10         VGA_CLK : in std_logic;
11         VGA_R, VGA_G, VGA_B: out std_logic_vector(9 downto 0)--  color information
12      );
13  end;
14
15  architecture behavioral of DisplayLogicWithRom is
16  -- Intensity of 10bit color in percent
17  constant C100 : std_logic_vector(9 downto 0) := (others=>'1');
18  constant C75 : std_logic_vector(9 downto 0) := (9=>'1', 8=>'0', others=>'1');
19  constant C50 : std_logic_vector(9 downto 0) := (9=>'0', others=>'1');
20  constant C25 : std_logic_vector(9 downto 0) := (9=>'0', 8=>'0', others=>'1');
21  constant C0 : std_logic_vector(9 downto 0) := (others=>'0');
22  constant G588 : std_logic_vector(9 downto 0) := "1001001100"; --G 147
23  constant B884 : std_logic_vector(9 downto 0) := "1101110100";-- B 221
24  constant R40 : std_logic_vector(9 downto 0) := "0010100000"; --R 40
25  constant B22 : std_logic_vector(9 downto 0) := "0001011000";-- B 22
26  constant G111 : std_logic_vector(9 downto 0) := "0110111100"; --G 111
27
28  -- records are VHDL equivalents of structures
29  type RGB_type is
30     record
31        R : std_logic_vector(9 downto 0);
32        G : std_logic_vector(9 downto 0);
33        B : std_logic_vector(9 downto 0);
34     end record;
35
36  -- Used colors - we defined them by the way that will allow their OR merging in future
37  constant BLUE : RGB_type := (C0,C0,C50);
38  constant GREEN : RGB_type := (C0,C50,C0);
39  constant RED : RGB_type := (C50,C0,C0);
40  constant YELLOW : RGB_type := (C75,C75,C0);
41  constant BLACK : RGB_type := (C0,C0,C0);
42  constant SKY_BLUE: RGB_type := (C0,G588,B884); --Light blue Color
43  constant DARK_BLUE: RGB_type := (R40,B22,G111); -- Darker Blue Color
44  constant WHITE: RGB_type := (C100,C100,C100);
45
46  constant YSIZE : integer := 240;
47  constant XSIZE : integer := 320;
48  constant EMBORGX : integer := 220; -- positions of picture in the flag
49  constant EMBORGY : integer := 32;
50  constant MEMROWSIZE : integer := 64; -- memory organization
51  constant MEMROWCOUNT : integer := 64;
52  constant MEM_END_ADDRESS : integer := 4095;
53
54  constant SORGX : integer := 16; -- positions of picture in the flag
55  constant SORGY : integer := 15;
56  constant SROW : integer := 128; -- memory organization
57  constant SROWCOUNT : integer := 56;
58
59
60
61  component Shark
62     PORT
63     (
64        address     : IN STD_LOGIC_VECTOR (12 DOWNTO 0);
65        clock     : IN STD_LOGIC  := '1';
66        q     : OUT STD_LOGIC_VECTOR (1 DOWNTO 0)
```

```
67      );
68  end component;
69
70
71  signal shark_address_s : STD_LOGIC_VECTOR(12 DOWNTO 0);
72  signal shark_q_s : STD_LOGIC_VECTOR (1 DOWNTO 0);
73
74
75  begin
76
77
78  shark_inst : Shark
79     port map(
80        address => shark_address_s,
81        clock   => VGA_CLK,
82        q       => shark_q_s
83     );
84
85
86      LSPflag : process(xcolumn, yrow, shark_q_s) -- output of process depends on xcolumn
    and yrow
87      variable RGB : RGB_type; -- output colors
88      variable x, y : integer; -- renamed xcolumn and yrow
89     variable  isShark01, isShark02, isShark03:boolean;
90      begin
91        x:=to_integer(xcolumn); y:=to_integer(yrow); -- convert to integer
92
93        isShark01:= x>=SORGX and x<SORGX+SROW
94                  and y>=SORGY and y<SORGY+SROWCOUNT;
95        isShark02:= x>=(SORGX + 150) and x<(SORGX + 150)+SROW
96                  and y>=(SORGY) and y<SORGY+SROWCOUNT;
97        isShark03:= x>=(SORGX + 90) and x<(SORGX + 90)+SROW
98                  and y>=(SORGY + 160) and y<(SORGY + 160)+SROWCOUNT;
99
100
101     if (isShark01 or isShark02 or isShark03) and shark_q_s/="01" then
102          case shark_q_s is
103             when "10" => RGB:=YELLOW;
104             when "00" => RGB:=BLACK;
105             when "11" => RGB:=BLUE;
106             when others => RGB:=BLACK;
107          end case;
108
109     elsif(x<0) or (x>=XSIZE) or (y<0) or (y>=YSIZE) then
110          RGB:=BLACK;                          -- Black outside the flag pixels
111       elsif (y >= 101) and (y <= 111) then
112         RGB:=BLACK;                        -- Strip 1
113       elsif (y >= 131) and (y <= 141) then
114         RGB:=BLACK;                   -- Strip 2
115       elsif (y >= 81) and (y <= 161) then
116          RGB:=WHITE;
117
118       else
119          RGB:=SKY_BLUE;
120       end if;
121
122       if isShark01 then shark_address_s <= std_logic_vector(to_unsigned((y-SORGY)*SROW +
    (SORGX+SROW-x),
123                                                                    shark_address_s'
    LENGTH));
124       elsif isShark02 then shark_address_s <= std_logic_vector(to_unsigned((y-SORGY)*SROW
     + (SORGX+SROW + 150 - x),
125                                                                    shark_address_s'
    LENGTH));
126       elsif isShark03 then shark_address_s <= std_logic_vector(to_unsigned((y-(SORGY+160
    ))*SROW + (x-(SORGX+90)),
```

```
127                                                                    shark_address_s'
    LENGTH));
128
129         else shark_address_s <=(others=>'0'); end if;
130
131      -- Copy results in RGB to outputs of entity
132         VGA_R<=RGB.R; VGA_G<=RGB.G; VGA_B<=RGB.B;
133   ---------------------------------------------------------------------------
134      end process;
135
136
137   end architecture behavioral;
```