```vhdl


library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;       -- type integer and unsigned

entity DisplayLogicCurve is
port(
       yrow, xcolumn : in unsigned(9 downto 0); -- row and  column number of VGA video
      VGA_CLK : in std_logic;
       VGA_R, VGA_G, VGA_B: out std_logic_vector(9 downto 0)--  color information
    );
end;

architecture behavioral of DisplayLogicCurve is
-- Intensity of 10bit color in percent
constant C100 : std_logic_vector(9 downto 0) := (others=>'1');
constant C75 : std_logic_vector(9 downto 0) := (9=>'1', 8=>'0', others=>'1');
constant C50 : std_logic_vector(9 downto 0) := (9=>'0', others=>'1');
constant C25 : std_logic_vector(9 downto 0) := (9=>'0', 8=>'0', others=>'1');
constant C0 : std_logic_vector(9 downto 0) := (others=>'0');
constant G588 : std_logic_vector(9 downto 0) := "1001001100"; --G 147
constant B884 : std_logic_vector(9 downto 0) := "1101110100";-- B 221
constant R40 : std_logic_vector(9 downto 0) := "0010100000"; --R 40
constant B22 : std_logic_vector(9 downto 0) := "0001011000";-- B 22
constant G111 : std_logic_vector(9 downto 0) := "0110111100"; --G 111

-- records are VHDL equivalents of structures
type RGB_type is
   record
      R : std_logic_vector(9 downto 0);
      G : std_logic_vector(9 downto 0);
      B : std_logic_vector(9 downto 0);
   end record;
-- Used colors - we defined them by the way allowing good overlapping
constant BLUE : RGB_type := (C0,C0,C50);
constant GREEN : RGB_type := (C0,C50,C0);
constant RED : RGB_type := (C50,C0,C0);
constant YELLOW : RGB_type := (C75,C75,C0);
constant BLACK : RGB_type := (C0,C0,C0);
constant SKY_BLUE: RGB_type := (C0,G588,B884);
constant NAVY_BLUE: RGB_type := (R40,B22,G111);
constant WHITE: RGB_type := (C100,C100,C100);

constant YSIZE : integer := 240;
constant XSIZE : integer := 320;
constant EMBORGX : integer := 220;
constant EMBORGY : integer := 32;
constant MEMROWSIZE : integer := 64;
constant MEMROWCOUNT : integer := 64;
constant MEM_END_ADDRESS : integer := 4095;

constant SORGX : integer := 16; -- positions for picture in the flag
constant SORGY : integer := 15;
constant SROW : integer := 128; -- memory organization
constant SROWCOUNT : integer := 56;


component Shark
   PORT
   (
      address     : IN STD_LOGIC_VECTOR (12 DOWNTO 0);
      clock    : IN STD_LOGIC  := '1';
      q     : OUT STD_LOGIC_VECTOR (1 DOWNTO 0)
   );
end component;
```

```vhdl

COMPONENT BlackCurve is
PORT(
   xcolumn : in unsigned(9 downto 0);
   q  : out unsigned(6 downto 0)
);
END COMPONENT;



signal shark_address_s : STD_LOGIC_VECTOR(12 DOWNTO 0);
signal shark_q_s : STD_LOGIC_VECTOR (1 DOWNTO 0);

signal blackCurve_q_s : unsigned(6 downto 0);

begin



shark_inst : Shark
   port map(
      address => shark_address_s,
      clock   => VGA_CLK,
      q       => shark_q_s
   );


blackC_inst : BlackCurve
PORT MAP(xcolumn => xcolumn,
         q => blackCurve_q_s);

    LSPflag : process(xcolumn, yrow, blackCurve_q_s) -- output of process depends on
xcolumn and yrow
    variable RGB : RGB_type; -- output colors
    variable x, y : integer; -- renamed xcolumn and yrow
    variable isPicture,isShark01, isShark02, isShark03:boolean;
    constant GREEN_LINE_ORG:integer:=100; -- TODO
    begin
      x:=to_integer(xcolumn); y:=to_integer(yrow); -- convert to integer

      isShark01:= x>=SORGX and x<SORGX+SROW
                  and y>=SORGY and y<SORGY+SROWCOUNT;

      isShark02:= x>=(SORGX + 150) and x<(SORGX + 150)+SROW
                  and y>=(SORGY) and y<SORGY+SROWCOUNT;

      isShark03:= x>=(SORGX + 90) and x<(SORGX + 90)+SROW  -- 3rd Inverted Shark
                  and y>=(SORGY + 160) and y<(SORGY + 160)+SROWCOUNT;

      if(x<0) or (x>=XSIZE) or (y<0) or (y>=YSIZE) then
         RGB:=BLACK;  -- black if there is no flag

      elsif (isShark01 or isShark02 or isShark03) and shark_q_s/="01" then
         case shark_q_s is
            when "10" => RGB:=YELLOW;
            when "00" => RGB:=BLACK;
            when "11" => RGB:=BLUE;
            when others => RGB:=BLACK;
         end case;

      elsif (y > to_integer(blackCurve_q_s) + 86) and (y < to_integer(blackCurve_q_s)+
104) then
         RGB:=BLACK;
      elsif (y > to_integer(blackCurve_q_s) + 117) and (y < to_integer(blackCurve_q_s)+
135) then
         RGB:=BLACK;
```

```
        elsif (y >= 81) and (y <= 161) then
            RGB:=WHITE;

        else
           RGB:=SKY_BLUE;
        end if;


        if isShark01 then shark_address_s  <= std_logic_vector(to_unsigned((y-SORGY)*SROW +
(SORGX+SROW-x),
                                                                              shark_address_s'
LENGTH));
        elsif isShark02 then shark_address_s  <= std_logic_vector(to_unsigned((y-SORGY)*SROW
 + (SORGX+SROW + 150 - x),
                                                                              shark_address_s'
LENGTH));
        elsif isShark03 then shark_address_s  <= std_logic_vector(to_unsigned((y-(SORGY+160
))*SROW + (x-(SORGX+90)),
                                                                              shark_address_s'
LENGTH));

        else shark_address_s  <=(others=>'0');
        end if;



    -- Copy results in RGB to outputs of entity
        VGA_R<=RGB.R; VGA_G<=RGB.G; VGA_B<=RGB.B;
-----------------------------------------------------------------------------
    end process;


end architecture behavioral;
```