1번.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.all;
entity image_processing is
port(
hsync_pir_n : in std_logic;
hactive_pir : in std_logic;
vsync_pir_n : in std_logic;
vactive_pir : in std_logic;
pir_rimg : in std_logic_vector (7 downto 0);
pir_gimg : in std_logic_vector (7 downto 0);
pir_bimg : in std_logic_vector (7 downto 0);
hsync_ip_n : out std_logic;
hactive_ip : out std_logic;
vsync_ip_n : out std_logic;
vactive_ip : out std_logic;
ip_rimg : out std_logic_vector (7 downto 0);
ip_gimg : out std_logic_vector (7 downto 0);
ip_bimg : out std_logic_vector (7 downto 0);
reset_disp_n : in std_logic;
dispclk : in std_logic);
end;
architecture simulation of image_processing is
constant H_HATCH_POS_S : integer := 40 ;
constant V_HATCH_POS_S : integer := 40 ;
constant H_HATCH_POS_E : integer := 140 ;
constant V_HATCH_POS_E : integer := 200 ;
constant H_HATCH_POS3_S : integer := 340 ;
constant V_HATCH_POS3_S : integer := 600 ;
constant H_HATCH_POS3_E : integer := 440 ;
constant V_HATCH_POS3_E : integer := 760 ;
constant H_HATCH_POS2_S : integer := 190 ;
constant V_HATCH_POS2_S : integer := 320 ;
constant H_HATCH_POS2_E : integer := 290 ;
constant V_HATCH_POS2_E : integer := 480 ;
constant WHITE_VALUE : std_logic_vector := CONV_STD_LOGIC_VECTOR(255,8);
constant BLACK_VALUE : std_logic_vector := CONV_STD_LOGIC_VECTOR(0,8);
constant BLACK : std_logic_vector := "00010000" ;
signal pcounter : integer range 0 to 511;
signal hcounter : integer range 0 to 1023;
signal hactive_1d : std_logic;
signal hactive_fpulse : std_logic;
begin
-- sync signal interface
sync_gen :
process(dispclk, reset_disp_n)
begin
```

```
if (reset_disp_n='0') then
hsync_ip_n <= '0';
hactive_1d <= '0'; -- hactive_1d : used for pulse generation
vsync_ip_n <= '0';
vactive_ip <= '0';
elsif (dispclk='1' and dispclk'event) then
hsync_ip_n <= hsync_pir_n;
hactive_1d <= hactive_pir; -- hactive_1d : used for pulse generation
vsync_ip_n <= vsync_pir_n;
vactive_ip <= vactive_pir;
end if;
end process;

hactive_ip <= hactive_1d;
-- falling edge of hactive signal.
hactive_fpulse <= (not hactive_pir) and hactive_1d;
-- image signal interface
image_gen :
process(dispclk)
variable flag, flag2 : std_logic;
begin
if (dispclk='1' and dispclk'event) then
if vactive_pir='1' and hactive_pir='1' then
pcounter <= (pcounter + 1) mod 512;
else
pcounter <= 0;
end if;
if vactive_pir='1' then
if hactive_fpulse = '1' then
hcounter <= (hcounter + 1) mod 1024;
end if;
else
hcounter <= 0;
end if;
if hactive_pir='1' and vactive_pir='1' then
if (hcounter >= V_HATCH_POS_S and hcounter < V_HATCH_POS_E
and pcounter >= H_HATCH_POS_S and pcounter < H_HATCH_POS_E)then
flag := '1';
elsif (hcounter >= V_HATCH_POS3_S and hcounter < V_HATCH_POS3_E
and pcounter >= H_HATCH_POS3_S and pcounter < H_HATCH_POS3_E) then
flag := '1';
elsif (hcounter >= V_HATCH_POS_S and hcounter < V_HATCH_POS_E
and pcounter >= H_HATCH_POS3_S and pcounter < H_HATCH_POS3_E)then
flag := '1';
elsif (hcounter >= V_HATCH_POS3_S and hcounter < V_HATCH_POS3_E
and pcounter >= H_HATCH_POS_S and pcounter < H_HATCH_POS_E) then
flag := '1';
else
flag := '0';
```

```
end if;
if hcounter >= V_HATCH_POS2_S and hcounter < V_HATCH_POS2_E
and pcounter >= H_HATCH_POS2_S and pcounter < H_HATCH_POS2_E then
flag2 := '1';
else
flag2 := '0';

end if;
end if;
if hactive_pir='1' and vactive_pir='1' then
if flag = '1' then
ip_rimg <= WHITE_VALUE;
ip_gimg <= WHITE_VALUE;
ip_bimg <= WHITE_VALUE;
elsif flag2 = '1' then
ip_rimg <= WHITE_VALUE;
ip_gimg <= WHITE_VALUE;
ip_bimg <= WHITE_VALUE;
else
ip_rimg <= BLACK_VALUE;
ip_gimg <= BLACK_VALUE;
ip_bimg <= BLACK_VALUE;
end if;
--else
-- ip_rimg <= BLACK;
-- ip_gimg <= BLACK;
-- ip_bimg <= BLACK;
end if;
end if;
end process;
end;
----------------
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.all;
use STD.TEXTIO.all;

entity ppm_image_write is
port(
hactive_ip : in std_logic;
vactive_ip : in std_logic;
ip_rimg : in std_logic_vector (7 downto 0);
ip_gimg : in std_logic_vector (7 downto 0);
ip_bimg : in std_logic_vector (7 downto 0);
reset_disp_n : in std_logic;
dispclk : in std_logic);
end ppm_image_write ;
architecture simulation of ppm_image_write is
```

```
file OUTFILE_IMG : text open write_mode is "White Window.ppm";
constant H_WIHTH : integer := 480;
constant V_HEIGHT : integer := 800;
------------------------------------------------------------------------
--procedure writeHeader (F: out text; Width, Height: in integer) is
procedure writeHeader (file F : text; Width, Height : in integer) is
variable L : line;
begin
-- ppm - portable pixmap file format
write(L, string'("P3")); -- magic number
writeline(F, L);
write(L, string'("# Creadted through VHDL simulation")); -- comment
writeline(F, L);
write(L, Width); -- width
write(L, string'(" "));
write(L, Height); -- height
writeline(F, L);
write(L, string'("255")); -- maximum color-component value
writeline(F, L);
end writeHeader;
------------------------------------------------------------------------
--procedure writeRGB (F: out text; R, G, B: in integer) is
procedure writeRGB (file F: text; R, G, B: in integer) is
variable L : line;
begin
write(L, R, RIGHT, 3); write(L, string'(" "));
write(L, G, RIGHT, 3); write(L, string'(" "));
write(L, B, RIGHT, 3);
writeline(F, L);
end writeRGB;

------------------------------------------------------------------------
begin
write_output : process(dispclk)
variable tmp_R, tmp_G, tmp_B : integer;
variable time_zero : std_logic := '0';
begin
if (time_zero = '0') then
writeHeader(OUTFILE_IMG, H_WIHTH, V_HEIGHT);
time_zero := '1';
elsif ( dispclk'event and dispclk = '1') then
if ( hactive_ip = '1' and vactive_ip = '1') then
tmp_R := conv_integer(unsigned(ip_rimg));
tmp_G := conv_integer(unsigned(ip_gimg));
tmp_B := conv_integer(unsigned(ip_bimg));
writeRGB(OUTFILE_IMG, tmp_R, tmp_G, tmp_B);
end if;
end if;
end process write_output;
```

```
end ;

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.all;
use STD.TEXTIO.all;
entity tb_EX14_03 is
end tb_EX14_03;
architecture simulation of tb_EX14_03 is
component clk_rst_gen
port(
reset_disp_n : out std_logic;
dispclk : out std_logic);
end component;
component sync_generator
port(
hsync_n : out std_logic;
hactive : out std_logic;
vsync_n : out std_logic;
vactive : out std_logic;

reset_disp_n : in std_logic;
dispclk : in std_logic);
end component;
component ppm_image_read
port(
hsync_n : in std_logic;
hactive : in std_logic;
vsync_n : in std_logic;
vactive : in std_logic;
hsync_pir_n : out std_logic;
hactive_pir : out std_logic;
vsync_pir_n : out std_logic;
vactive_pir : out std_logic;
pir_rimg : out std_logic_vector (7 downto 0);
pir_gimg : out std_logic_vector (7 downto 0);
pir_bimg : out std_logic_vector (7 downto 0);
reset_disp_n : in std_logic;
dispclk : in std_logic);
end component;
component image_processing
port(
hsync_pir_n : in std_logic;
hactive_pir : in std_logic;
vsync_pir_n : in std_logic;
vactive_pir : in std_logic;
pir_rimg : in std_logic_vector (7 downto 0);
pir_gimg : in std_logic_vector (7 downto 0);
```

```
pir_bimg : in std_logic_vector (7 downto 0);
hsync_ip_n : out std_logic;
hactive_ip : out std_logic;
vsync_ip_n : out std_logic;
vactive_ip : out std_logic;
ip_rimg : out std_logic_vector (7 downto 0);
ip_gimg : out std_logic_vector (7 downto 0);

ip_bimg : out std_logic_vector (7 downto 0);
reset_disp_n : in std_logic;
dispclk : in std_logic);
end component;
component ppm_image_write
port(
hactive_ip : in std_logic;
vactive_ip : in std_logic;
ip_rimg : in std_logic_vector (7 downto 0);
ip_gimg : in std_logic_vector (7 downto 0);
ip_bimg : in std_logic_vector (7 downto 0);
reset_disp_n : in std_logic;
dispclk : in std_logic);
end component;
signal reset_disp_n : std_logic;
signal dispclk : std_logic;
signal hsync_n : std_logic;
signal hactive : std_logic;
signal vsync_n : std_logic;
signal vactive : std_logic;
signal hsync_pir_n : std_logic;
signal hactive_pir : std_logic;
signal vsync_pir_n : std_logic;
signal vactive_pir : std_logic;
signal pir_rimg : std_logic_vector (7 downto 0);
signal pir_gimg : std_logic_vector (7 downto 0);
signal pir_bimg : std_logic_vector (7 downto 0);
signal hsync_ip_n : std_logic;
signal hactive_ip : std_logic;
signal vsync_ip_n : std_logic;
signal vactive_ip : std_logic;
signal ip_rimg : std_logic_vector (7 downto 0);
signal ip_gimg : std_logic_vector (7 downto 0);
signal ip_bimg : std_logic_vector (7 downto 0);
begin
u_clkgen: clk_rst_gen
port map (
reset_disp_n => reset_disp_n,
dispclk => dispclk);
u_sync_gen: sync_generator
port map (
```

```
hsync_n => hsync_n,
hactive => hactive,
vsync_n => vsync_n,
vactive => vactive,
reset_disp_n => reset_disp_n,
dispclk => dispclk);
u_image_read: ppm_image_read
port map (
hsync_n => hsync_n,
hactive => hactive,
vsync_n => vsync_n,
vactive => vactive,
hsync_pir_n => hsync_pir_n,
hactive_pir => hactive_pir,
vsync_pir_n => vsync_pir_n,
vactive_pir => vactive_pir,
pir_rimg => pir_rimg,
pir_gimg => pir_gimg,
pir_bimg => pir_bimg,
reset_disp_n => reset_disp_n,
dispclk => dispclk);
u_image_processing : image_processing
port map (
hsync_pir_n => hsync_pir_n,
hactive_pir => hactive_pir,
vsync_pir_n => vsync_pir_n,
vactive_pir => vactive_pir,
pir_rimg => pir_rimg,
pir_gimg => pir_gimg,
pir_bimg => pir_bimg,
hsync_ip_n => hsync_ip_n,
hactive_ip => hactive_ip,
vsync_ip_n => vsync_ip_n,
vactive_ip => vactive_ip,
ip_rimg => ip_rimg,
ip_gimg => ip_gimg,
ip_bimg => ip_bimg,
reset_disp_n => reset_disp_n,
dispclk => dispclk);
u_image_write : ppm_image_write
port map (
hactive_ip => hactive_ip,
vactive_ip => vactive_ip,
ip_rimg => ip_rimg,
ip_gimg => ip_gimg,
ip_bimg => ip_bimg,
reset_disp_n => reset_disp_n,
dispclk => dispclk);
end ;
```