```
----------------------------------------------------------------------------------
-- Engineer: Taggart Bonham and Will Chisholm
--
-- Create Date: 05/23/2017 09:05:13 PM
-- Design Name:
-- Module Name: VGA_TEST_TOP - Behavioral
-- Project Name: Etch-a-Sketch final project
-- Target Devices: Digilent Basys3 Board
-- Tool Versions: Vivado 2016.1
-- Description: Top file of final project.
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;
library UNISIM;
use UNISIM.VComponents.all;

entity EtchaSketch_TOP is
    Port ( mclk : in STD_LOGIC;
           r,g,b : in STD_LOGIC;
           XA, XB, YA, YB : in STD_LOGIC;
           rst : in STD_LOGIC;
           H_sync : out STD_LOGIC;
           V_sync : out STD_LOGIC;
           vgaBlue, vgaGreen, vgaRed : out STD_LOGIC_VECTOR(3 downto 0));
end EtchaSketch_TOP;

architecture Behavioral of EtchaSketch_TOP is

--timing driver for VGA connection
Component VGA_SYNC is
Port (  clk     :   in  STD_LOGIC;
        V_sync  :   out STD_LOGIC;
        H_sync  :   out STD_LOGIC;
        video_on:   out STD_LOGIC;
        pixel_x :   out STD_LOGIC_vector(9 downto 0);
        pixel_y :   out STD_LOGIC_VECTOR(8 downto 0) );
end component;

--logic regarding current location on screen
Component CurPixel is
Port (    clk : in STD_LOGIC;
          UP : in STD_LOGIC;
          DOWN : in STD_LOGIC;
          LEFT : in STD_LOGIC;
          RIGHT : in STD_LOGIC;
          curLoc : out STD_LOGIC_VECTOR(16 downto 0));
end component;

Component ColorDecoder is
    Port ( r : in STD_LOGIC;
           g : in STD_LOGIC;
           b : in STD_LOGIC;
           COLOR : out STD_LOGIC_VECTOR(7 downto 0));
end component;


--Component blk_mem_gen_1 is
--  PORT (
--  clka : IN STD_LOGIC;
--  wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
--  addra : IN STD_LOGIC_VECTOR(16 DOWNTO 0);
```

```
--  dina : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
--  clkb : IN STD_LOGIC;
--  enb : IN STD_LOGIC;
--  addrb : IN STD_LOGIC_VECTOR(16 DOWNTO 0);
--  doutb : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
--);
--  end component;

 --BRAM storage for VGA
--stores current color at screen locations
 Component BRAM IS
    PORT (
      clka : IN STD_LOGIC;
      wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
      addra : IN STD_LOGIC_VECTOR(16 DOWNTO 0);
      dina : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
      clkb : IN STD_LOGIC;
      addrb : IN STD_LOGIC_VECTOR(16 DOWNTO 0);
      doutb : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
    );
  END component;
--controller for design
--inputs from rotary encoders turned into control signals
--responsible for moving the current location and clearing
Component cwController is
PORT (     clk : in STD_LOGIC;
           a1 : in STD_LOGIC;
           a2 : in STD_LOGIC;
           b1 : in STD_LOGIC;
           b2 : in STD_LOGIC;
           rst : in STD_LOGIC;
           UP : out STD_LOGIC;
           DOWN : out STD_LOGIC;
           LEFT : out STD_LOGIC;
           RIGHT : out STD_LOGIC;
           CLR : out STD_LOGIC);
  end component;

 Component Mux2x1 is
   Port ( A : in STD_LOGIC_VECTOR (16 downto 0);
          B : in STD_LOGIC_VECTOR (16 downto 0);
          sel : in STD_LOGIC;
          y: out STD_LOGIC_VECTOR (16 downto 0));
end component;

-- SIGNAL DECLARATIONS


-- Signals for the serial clock divider, which divides the 100 MHz clock down to 1 MHz
constant SCLK_DIVIDER_VALUE: integer := 2;--100 / 2;
signal sclkdiv: unsigned(1 downto 0) := (others => '0');  -- clock divider counter
signal sclk_unbuf: std_logic := '0';    -- unbuffered serial clock
signal sclk: std_logic := '0';          -- internal serial clock


--wiring
--outputs from VGA_SYNC
signal vid_on : STD_LOGIC;
signal x : STD_LOGIC_VECTOR(9 downto 0);
signal y: STD_LOGIC_VECTOR(8 downto 0);

--inputs to CurrentPixel
signal PIXELUP : STD_LOGIC;
signal PIXELDOWN : STD_LOGIC;
```

```
signal PIXELLEFT : STD_LOGIC;
signal PIXELRIGHT : STD_LOGIC;


--inputs to ColorDecoder
signal RGBin : STD_LOGIC_VECTOR(7 downto 0);

--inputs to 2x1 Mux
signal CLR : std_logic;
signal currentPixel, screenLoc : STD_LOGIC_VECTOR(16 downto 0);


--inputs to BRAM
signal decodedColor : std_logic_vector(7 downto 0):= (others => '0');
signal wr_add_in : std_logic_vector(16 downto 0);
signal memEN : std_logic_vector(0 downto 0);
--output from BRAM
signal color : STD_LOGIC_VECTOR(7 downto 0);
-----------------------------------------------
begin
-- Clock buffer for sclk
-- The BUFG component puts the signal onto the FPGA clocking network

Slow_clock_buffer: BUFG
    port map (I => sclk_unbuf,
              O => sclk );

-- Divide the 100 MHz clock down to 25 MHz, then toggling a flip flop gives the final
-- 1 MHz system clock
Serial_clock_divider: process(mclk)
begin
    if rising_edge(mclk) then
        if sclkdiv = 1 then
            sclkdiv <= (others => '0');
            sclk_unbuf <= NOT(sclk_unbuf);
        else
            sclkdiv <= sclkdiv + 1;
        end if;
    end if;
end process Serial_clock_divider;

--generates the signals that drive the VGA, syncroniczes output from BRAM to display
sync: VGA_sync port map(
        clk => sclk, --25 MHz clock
        V_sync  => V_sync,
        H_sync  => H_sync,
        video_on => vid_on,
        pixel_x =>  x,
        pixel_y =>  y);


--switch to BRAM color storage converter
--turns 3 discreet RGB switches into RGB(8)
colorDec: ColorDecoder port map (
    r => r,
    g =>g,
    b =>b,
    COLOR => decodedColor);

 --driven by controller, logic for current location
 --of etch-a-sketch pixel on the screen
pixel: CurPixel port map(
        clk => sclk,
        UP => PIXELUP,
```

```
        DOWN => PIXELDOWN,
        LEFT => PIXELLEFT,
        RIGHT => PIXELRIGHT,
        curLoc => currentPixel);

--ties x and y together from VGA_sync
        screenLoc <= x(9 downto 1) & y(8 downto 1);

--driven by clear, inputs either the current location on screen
--or the VGA screen sync signal to enable the BRAM to be written over
mux: Mux2x1 port map(
    A => currentPixel,
    B => screenLoc,
    sel => CLR,
    y => wr_add_in
);


--BRAM stores the RGB(8) colors of the current location
--read-write enabled during vid_on to ensure only legitimate positions on screen stored
--to protect against porches being stored
memEn(0) <= vid_on; --need to feed into bram as a std_logic_vector
--mem: blk_mem_gen_1 port map(
--    clka => sclk,
--    wea => memEn,
--    addra => wr_add_in,
--    dina => decodedColor,
--    enb => '1',
--    clkb => sclk,
--    addrb => screenLoc,
--    doutb => color);

mem: bram port map(
    clka => sclk,
    wea => memEn,
    addra => wr_add_in,
    dina => decodedColor,
    clkb => sclk,
    addrb => screenLoc,
    doutb => color);

--controller converts rotary signals into Left right up down signals
--additionally drives the clearing of the screen
controls: cwController port map(
    clk => sclk,
    a1 => XA,
    a2 => XB,
    b1 => YA,
    b2 => YB,
    rst => rst,
    UP => PIXELUP,
    DOWN => PIXELDOWN,
    LEFT => PIXELLEFT,
    RIGHT => PIXELRIGHT,
    CLR => CLR);


--splits color into red green and blue for vga
vgaRed <= color(7 downto 5) & '0';
vgaGreen <= color(4 downto 2) & '0';
vgaBlue <= color(1 downto 0) & "00";

end Behavioral;
```