VGA Implementation

Last updated 5/20/20

VGA Basics

These slides describe the design and operation of a VGA display driver

Upon completion: You should be able to describe and operation of VGA display driver and use it in your own designs

VGA Driver

```
--------------------
--
-- VGA_drvr.vhdl
-- Created: 9/14/16
-- By: tj
-- For: EE3921
--
-- Rev 1 - modified for DE10 Lite - 7/24/17
-- Rev 2 - name changed
--         modified to conform to best practices - 7/15/18
--
--------------------
--  Overview
--
-- VGA sync driver
--
--
-- Creates the necessary v-sync and h-sync signals to drive a VGA display
-- Creates pixel X and Y coordinates to indicate the current raster location
-- Creates a "display on" signal to indicate data is being displayed  (optional usage)
-- Provides a buffer path for the RGB signal to ensure syncronization (optional usage)
--    NOTE: If using the RGB buffering, the RGB output is already blanked
--          by the display on signal
--------------------------
--- Details
--
-- Default is:
--             25MHz clock
--             640 x 480 display
--
-- Override default by using generics
--
-- Uses whatever the standard requires for a pixel clock
--     eg. default operation requires a 25MHz input (pixel) clock
--          which can be created via PLL of the DE10 Lite base 50MHz clk
--
--  Note that different display resolutions require:
--    Different pixel parameters
--    Different pixel clock frequencies
--    Different sync pulse polarities
--
--  This code is developed based on timing:
--     back porch -> display -> front porch -> sync pulse
--
```

VGA Driver

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;

entity VGA_drvr is
    generic(
        -- Default VGA 640-by-480 display parameters
        H_back_porch:  natural:=48;
        H_display:     natural:=640;
        H_front_porch: natural:=16;
        H_retrace:     natural:=96;
        V_back_porch:  natural:=33;
        V_display:     natural:=480;
        V_front_porch: natural:=10;
        V_retrace:     natural:=2;
        Color_bits:    natural:=4;
        H_sync_polarity: std_logic:= '0';    --  depends on standard (negative -> 0), (positive -> 1)
        V_sync_polarity: std_logic:= '0';    --  depends on standard (negative -> 0), (positive -> 1)
        -- calculated based on other generic parameters
        H_counter_size: natural:= 10;     --  depends on above generic values
        V_counter_size: natural:= 10      --  depends on above generic values
    );
```

VGA Driver

```
    port(
        -- clock and reset - vid_clk is the appropriate video clock
        -- vid_clk would be 25MHz for a 640 x 480 display
        i_vid_clk:      in    std_logic;
        i_rstb:         in    std_logic;
        -- standard video sync signals
        o_h_sync:       out   std_logic;
        o_v_sync:       out   std_logic;
        --  X and Y values for current pixel location being written to the screen
        --  Can be used for reference in upper levels of design
        o_pixel_x:      out   std_logic_vector (H_counter_size -1 downto 0);
        o_pixel_y:      out   std_logic_vector (V_counter_size - 1 downto 0);
        -- signal to indicate display is actively being written
        -- use this to set RGB values to 0 when not on an active part of the screen
        -- ** not used if using the RGB in/out synchronous signals
        o_vid_display: out   std_logic;
        -- convenience signals
        -- syncronize rgb outputs to vid_clk
        i_red_in:       in    std_logic_vector((Color_bits - 1) downto 0);
        i_green_in:     in    std_logic_vector((Color_bits - 1) downto 0);
        i_blue_in:      in    std_logic_vector((Color_bits - 1) downto 0);
        o_red_out:      out   std_logic_vector((Color_bits - 1) downto 0);
        o_green_out:    out   std_logic_vector((Color_bits - 1) downto 0);
        o_blue_out:     out   std_logic_vector((Color_bits - 1) downto 0)
    );
end;
```

VGA Driver

```
architecture behavioral of VGA_drvr is
    --
    -- Counter signals
    --
    signal h_count:         unsigned(H_counter_size - 1 downto 0);
    signal h_count_next:    unsigned(H_counter_size - 1 downto 0);
    signal v_count:         unsigned(V_counter_size - 1 downto 0);
    signal v_count_next:    unsigned(V_counter_size - 1 downto 0);
    --
    -- Display signals
    --
    signal v_display_on:    std_logic;
    signal h_display_on:    std_logic;
    signal display_on:      std_logic;
    --
    -- Convenience signals (RGB buffering)
    --
    signal red:    unsigned((Color_bits - 1) downto 0);
    signal green:  unsigned((Color_bits - 1) downto 0);
    signal blue:   unsigned((Color_bits - 1) downto 0);

begin
```

VGA Driver

```
--
--  counter logic
--
process (h_count, v_count)
begin
   --
   --    Horizontal counter
   --
   if (h_count >= (H_back_porch + H_display + H_front_porch + H_retrace - 1)) then
      h_count_next <= (others => '0');
   else
      h_count_next <= h_count + 1;
   end if;
   --
   --    Horizontal Sync
   --
   if ((h_count >= (H_back_porch + H_display + H_front_porch)) and
                   (h_count <= (H_back_porch + H_display + H_front_porch + H_retrace))) then
      o_h_sync <= H_sync_polarity;
   else
      o_h_sync <= not(H_sync_polarity);
   end if;
   --
   --    Horizontal display on
   --
   if ((h_count >= (H_back_porch)) and (h_count <= (H_back_porch + H_display - 1))) then
      h_display_on <= '1';
   else
      h_display_on <= '0';
   end if;
   --
```

VGA Driver

```
    --
    --    Vertical counter
    --
    --    Must also wait for the end of the horizontal counter
    --    to get all the way to the lower right
    --
    if ((v_count >= (V_back_porch + V_display + V_front_porch + V_retrace - 1)) and
                    (h_count >= (H_back_porch + H_display + H_front_porch + H_retrace -1))) then
       v_count_next <= (others => '0');
    elsif (h_count >= (H_back_porch + H_display + H_front_porch + H_retrace -1)) then
       v_count_next <= v_count + 1;
    else
       v_count_next <= v_count;
    end if;
    --
    --    Vertical Sync
    --
    if ((v_count >= (V_back_porch + V_display + V_front_porch)) and
                    (v_count <= (V_back_porch + V_display + V_front_porch + V_retrace))) then
       o_v_sync <= V_sync_polarity;
    else
       o_v_sync <= not(V_sync_polarity);
    end if;
    --
    --    Vertical display on
    --
    if ((v_count >= (V_back_porch)) and (v_count <= (V_back_porch + V_display - 1))) then
       v_display_on <= '1';
    else
       v_display_on <= '0';
    end if;
end process;
--
--    Combined display on
--
display_on <= h_display_on AND v_display_on;
```

VGA Driver

```
----------------
--  Synchronous update section
----------------
process (i_vid_clk, i_rstb)
begin
   if i_rstb='0' then
      v_count  <= (others=>'0');
      h_count  <= (others=>'0');
      red      <= (others=>'0');
      green    <= (others=>'0');
      blue     <= (others=>'0');
   elsif (rising_edge(i_vid_clk)) then
      v_count <= v_count_next;
      h_count <= h_count_next;
      --RGB syncronizer
      red   <= unsigned(i_red_in);
      green <= unsigned(i_green_in);
      blue  <= unsigned(i_blue_in);
   end if;
end process;
```

VGA Driver

```
    -----------------
    --  Output section
    -----------------
    --
    -- alternate signal to control RGB values externally
    -- not used if the RGB syncronizer is used
    --
    o_vid_display <= display_on;
    --
    -- pixel values range from 0 to ((display size) -1)
    -- if display is off, pixel values are set to (display size)
    -- eg x might range from 0 to 799 with x = 800 when the display is off
    --
    process(all)
    begin
        if(h_display_on = '1') then
            o_pixel_x <= std_logic_vector(h_count - H_back_porch);
        else
            o_pixel_x <= std_logic_vector(to_unsigned(H_display, H_counter_size));
        end if;
        if(v_display_on = '1') then
            o_pixel_y <= std_logic_vector(v_count - V_back_porch);
        else
            o_pixel_y <= std_logic_vector(to_unsigned(V_display, V_counter_size));
        end if;
    end process;
    --
    -- RGB helper to turn off RGB when display is not in the active area of the screen
    --
    process(all)
    begin
        if(display_on = '1') then
            o_red_out <= std_logic_vector(red);
            o_green_out <= std_logic_vector(green);
            o_blue_out <= std_logic_vector(blue);
        else
            o_red_out <= (others => '0');
            o_green_out <= (others => '0');
            o_blue_out <= (others => '0');
        end if;
    end process;


end architecture;
```

VGA Driver - Implementation

```
--------------------
--
-- vga_1280x1024_test_de10.vhdl
-- Created: 7/16/18
-- By: johnsontimoj
-- For: EE3921
--
--------------------
--  Overview
--
-- Test for VGA_drvr in 1280 x 1024 mode
--
--
-- Instantiates the VGA_drvr module and drives RGB
--    1280 x 1024 at 108MHz
--
--------------------------
--- Details
--
--  uses switches as RGB input
--------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity vga_1280x1024_test_de10 is
   port(
      CLOCK_50:    in std_logic;  -- 50MHz
      SW:          in std_logic_vector(9 downto 0);
      VGA_HS:      out std_logic;
      VGA_VS:      out std_logic;
      VGA_R:       out std_logic_vector(3 downto 0);
      VGA_G:       out std_logic_vector(3 downto 0);
      VGA_B:       out std_logic_vector(3 downto 0)
    );
end;
```

VGA Driver - Implementation

```
architecture hardware of vga_1280x1024_test_de10 is
   -- intermediate signals
   signal   clk_108:       std_logic;

   COMPONENT VGA_drvr
      GENERIC(
         -- Default VGA 640-by-480 display parameters
         H_back_porch:  natural:=48;
         H_display:     natural:=640;
         H_front_porch: natural:=16;
         H_retrace:     natural:=96;
         V_back_porch:  natural:=33;
         V_display:     natural:=480;
         V_front_porch: natural:=10;
         V_retrace:     natural:=2;
         Color_bits:    natural:=4;
         H_sync_polarity: std_logic:= '0';   --  depends on standard (negative -> 0), (positive -> 1)
         V_sync_polarity: std_logic:= '0';   --  depends on standard (negative -> 0), (positive -> 1)
         -- calculated based on other generic parameters
         H_counter_size: natural:= 10;    --  depends on above generic values
         V_counter_size: natural:= 10     --  depends on above generic values
      );
      PORT(
         i_vid_clk      :  IN STD_LOGIC;
         i_rstb         :  IN STD_LOGIC;
         o_h_sync       :  OUT STD_LOGIC;
         o_v_sync       :  OUT STD_LOGIC;
         o_pixel_x      :  OUT STD_LOGIC_VECTOR(h_counter_size-1 DOWNTO 0);
         o_pixel_y      :  OUT STD_LOGIC_VECTOR(v_counter_size-1 DOWNTO 0);
         o_vid_display  :  OUT STD_LOGIC;
         i_red_in       :  IN STD_LOGIC_VECTOR(color_bits-1 DOWNTO 0);
         i_green_in     :  IN STD_LOGIC_VECTOR(color_bits-1 DOWNTO 0);
         i_blue_in      :  IN STD_LOGIC_VECTOR(color_bits-1 DOWNTO 0);
         o_red_out      :  OUT STD_LOGIC_VECTOR(color_bits-1 DOWNTO 0);
         o_green_out    :  OUT STD_LOGIC_VECTOR(color_bits-1 DOWNTO 0);
         o_blue_out     :  OUT STD_LOGIC_VECTOR(color_bits-1 DOWNTO 0)
      );
   END COMPONENT;
```

```
   component pll_108MHz
      PORT
      (
         inclk0: IN STD_LOGIC  := '0';
         c0    : OUT STD_LOGIC
      );
   end component;


Begin
```

VGA Driver - Implementation

```
Begin
   ------------------
   -- VGA_drvr with 1280x1024 configuration
   ------------------
   vga: VGA_drvr
      generic map(H_back_porch => 248,
                  H_display => 1280,
                  H_front_porch => 48,
                  H_retrace => 112,
                  V_back_porch => 38,
                  V_display => 1024,
                  V_front_porch => 1,
                  V_retrace => 3,
                  Color_bits => 4,
                  H_counter_size => 11,
                  V_counter_size => 11,
                  H_sync_polarity => '1',
                  V_sync_polarity => '1'
      )
      PORT MAP(
               i_vid_clk      => clk_108,
               i_rstb         => SW(0) ,
               o_h_sync       => VGA_HS,
               o_v_sync       => VGA_VS,
               --o_pixel_x       => PIXEL_X,
               --o_pixel_y       => PIXEL_Y,
               --o_vid_display   => VID_DISPLAY,
               i_red_in(0)    => SW(7),
               i_red_in(1)    => SW(7),
               i_red_in(2)    => SW(8),
               i_red_in(3)    => SW(9),
               i_green_in(0)  => SW(4),
               i_green_in(1)  => SW(4),
               i_green_in(2)  => SW(5),
               i_green_in(3)  => SW(6),
               i_blue_in(0)   => SW(1),
               i_blue_in(1)   => SW(1),
               i_blue_in(2)   => SW(2),
               i_blue_in(3)   => SW(3),
               o_red_out      => VGA_R,
               o_green_out    => VGA_G,
               o_blue_out     => VGA_B
      );
```

```
   ------------------
   --  Clock divider PLL
   ------------------
   pll : pll_108MHz
      PORT MAP (
               inclk0    => CLOCK_50,
               c0  => clk_108
      );

end architecture;
```

VGA Driver - Implementation