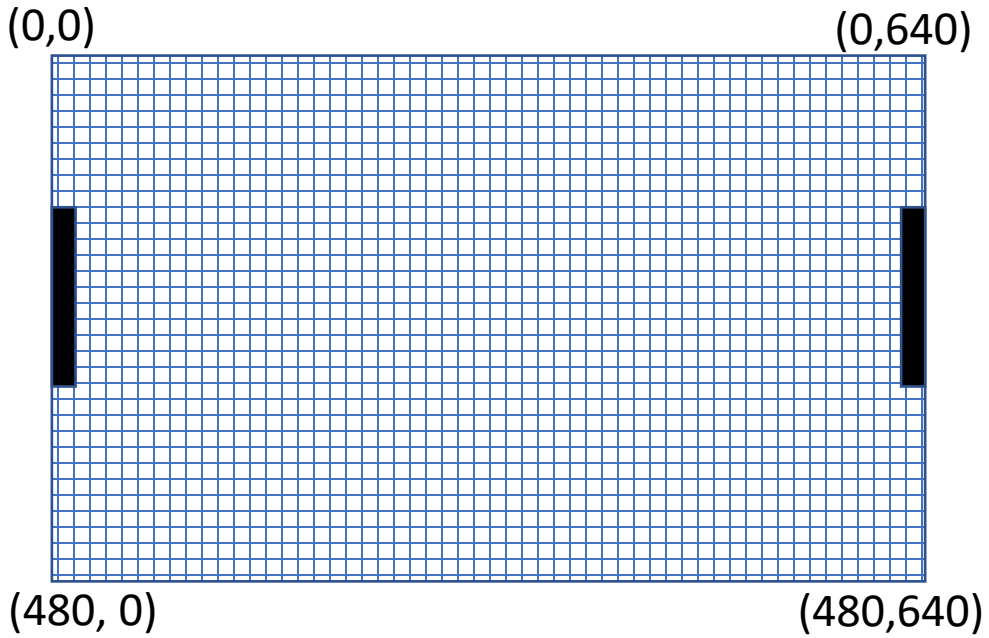
Pong

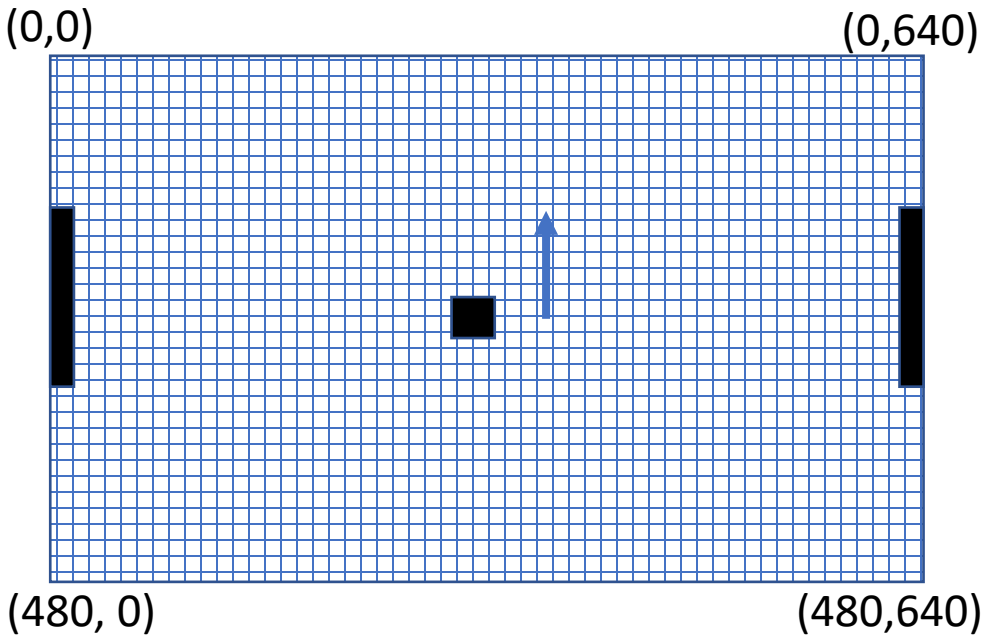


Pong

```
signal hpos : integer range 0 to 640:=0;
signal vpos : integer range 0 to 480:=0;
--Paddle Signals
signal paddle_h1 : integer range 586 to 2246:= 620;
signal paddle_v1 : integer range 47 to 1077:= 515;
signal paddle_h2 : integer range 586 to 2246:= 2197;
signal paddle_v2 : integer range 47 to 1077:= 512;

--Ball signals
signal ball_pos_h1      : integer range 586 to 2246:= 1500;
signal ball_pos_v1      : integer range 47 to 1077:= 515;
signal ball_up                  : std_logic:= '0';
signal ball_right               : std_logic:= '1';
signal ball_speed_h             : integer range 0 to 15:= default_ball_speed;
signal ball_speed_v             : integer range 0 to 15:= default_ball_speed;
```

Pong – Ball Control

```
--Moves the ball and detects collisions with the edges and the paddles
move_ball : process (vga_clk)
begin

	if (rising_edge(vga_clk) and new_frame = '1') then

		if (reset = '1') then
			left_player_score <= 0;
			right_player_score <= 0;
			ball_pos_v1 <= 515;
			ball_pos_h1 <= 1500;
			ball_speed_h <= default_ball_speed;
			ball_speed_v <= default_ball_speed;
		else

			--If ball travelling up, and not at top
			if (ball_pos_v1 < 1062 and ball_up = '1') then
				ball_pos_v1 <= ball_pos_v1 + ball_speed_v;
			--If ball travelling up and at top
			elsif (ball_up = '1') then
				ball_up <= '0';
			--Ball travelling down and not at bottom
			elsif (ball_pos_v1 >47 and ball_up = '0') then
				ball_pos_v1 <= ball_pos_v1 - ball_speed_v;
			--Ball travelling down and at bottom
			elsif (ball_up = '0') then
				ball_up <= '1';
			end if;
```




```
--If ball travelling right, and not far right
if (ball_pos_h1 < 2231 and ball_right = '1') then
	ball_pos_h1 <= ball_pos_h1 + ball_speed_h;
--If ball travelling right and at far right
elsif (ball_right = '1') then
	ball_right	<= '0';

	if (left_player_score  < 9) then
		left_player_score <= left_player_score + 1;
		--Reset ball position
		ball_pos_v1 <= 515;
		ball_pos_h1 <= 1500;
	else
		--Force a reset by stopping the ball
		ball_speed_h			<= 0;
		ball_speed_v			<= 0;
	end if;
```

Pong – Ball Control

```
        --Ball travelling left and not at far left
        elsif (ball_pos_h1 >586 and ball_right = '0') then
                ball_pos_h1 <= ball_pos_h1 - ball_speed_h;
        --Ball travelling left and at far left
        elsif (ball_right = '0') then
                ball_right <= '1';

                if (right_player_score  < 9) then
                        right_player_score <= right_player_score + 1;
                        --Reset ball position
                        ball_pos_v1 <= 515;
                        ball_pos_h1 <= 1500;
                else
                        --Force a reset by stopping the ball
                        ball_speed_h                    <= 0;
                        ball_speed_v                    <= 0;
                end if;
        end if;
end if;
```

Pong – Ball Control

```
        --Very simple collision detection
        elsif rising_edge(vga_clk) then
                --Since only the ball is blue and only the paddles are red then if they occur together a collision has happend!
                if (set_blue = X"F" and set_red = X"F") then
                        ball_right <= ball_right XOR '1'; --Toggle horizontal ball direction on collision
                end if;

        end if;

end process;
```

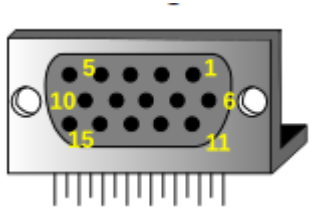
Pong – Draw Paddle

```
--Draws the left and right paddles
draw_paddle : process (vga_clk)
begin

	if (rising_edge(vga_clk)) then
		--Paddles are 15 x 80 pixels
		if ( (hpos >= paddle_h1 and hpos < paddle_h1 + 15) and (vpos >= paddle_v1 and vpos < paddle_v1 + 80) ) then
			set_red <= X"F";
		elsif ( (hpos >= paddle_h2 and hpos < paddle_h2 + 15) and (vpos >= paddle_v2 and vpos < paddle_v2 + 80) ) then
			set_red <= X"F";
		else
			set_red <= X"0";
		end if;

	end if;

end process;
```

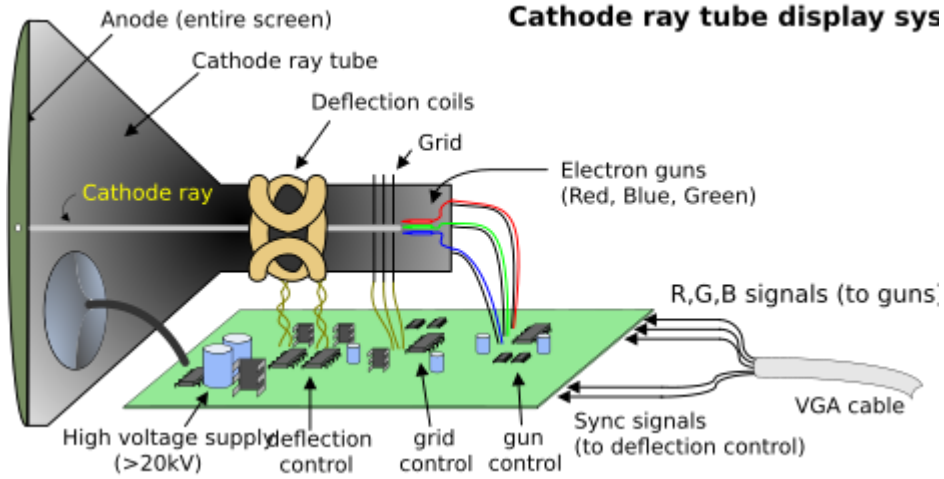
Pong – Draw Ball

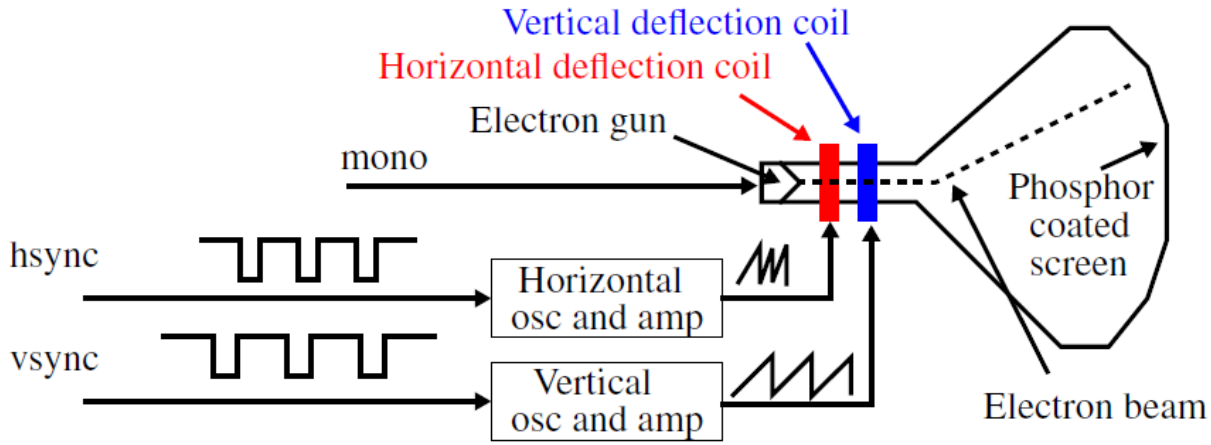
```
--Draws the ball
draw_ball : process (vga_clk)
begin

	if (rising_edge(vga_clk)) then
		--The ball is 15 x 15 pixels
		if ( (hpos >= ball_pos_h1 and hpos < ball_pos_h1 + 15) and (vpos >= ball_pos_v1 and vpos < ball_pos_v1 + 15) ) then
			set_blue <= X"F";
		else
			set_blue <= X"0";
		end if;

	end if;

end process;
```

VGA Display



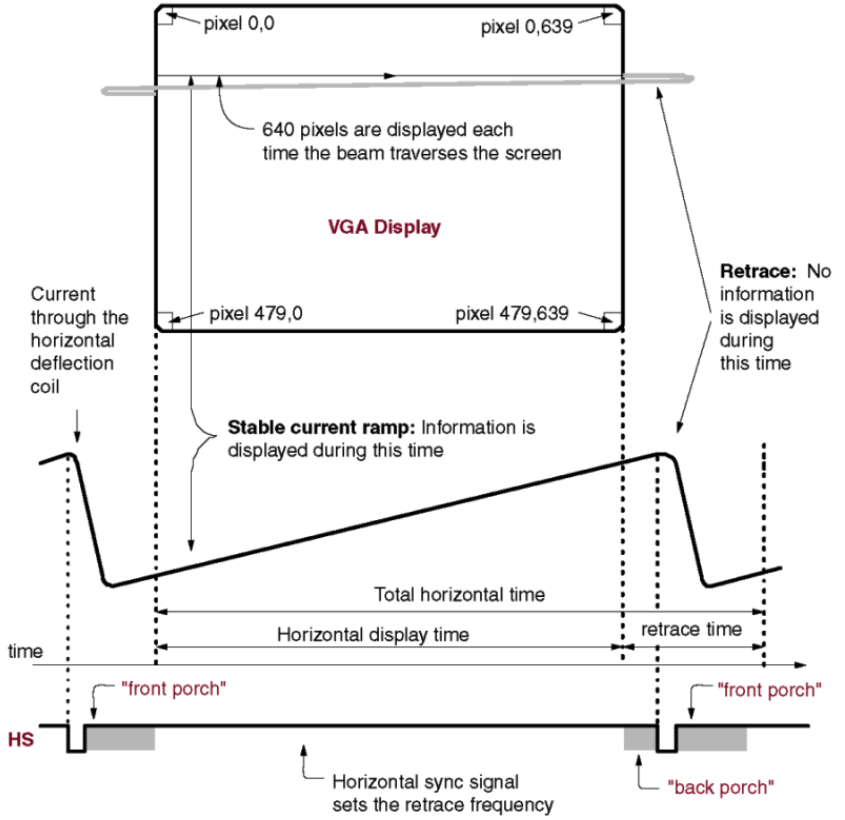
Pin 1: Red	Pin 5: GND
Pin 2: Grn	Pin 6: Red GND
Pin 3: Blue	Pin 7: Grn GND
Pin 13: HS	Pin 8: Blu GND
Pin 14: VS	Pin 10: Sync GND

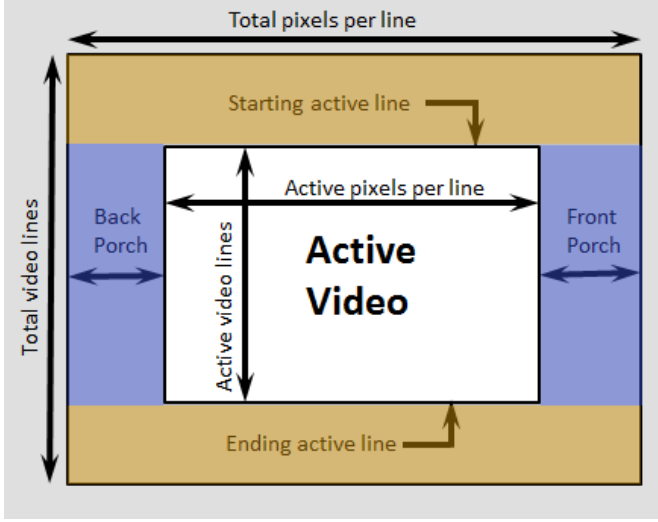




VGA Display

- Video Graphics Array
- Working Principle
  - Scanning the screen and using two signals *hsync* and *vsync* to synchronize the exact location on the screen where cursor is ready to draw



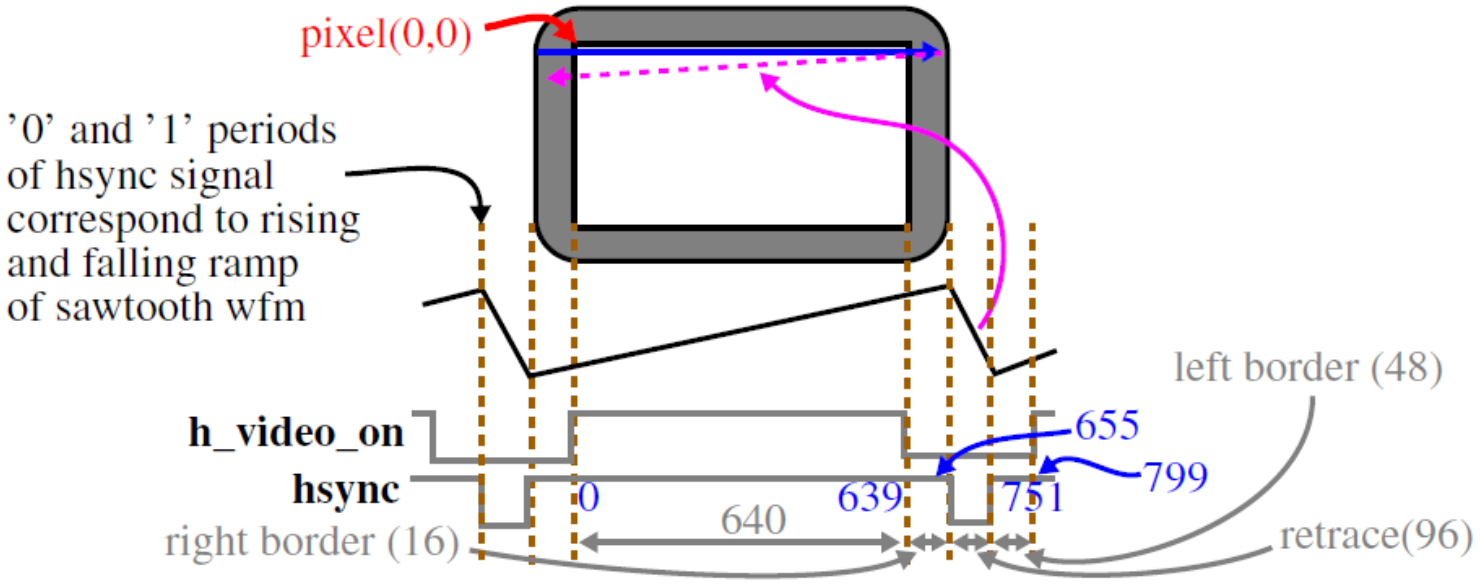


VGA - Timing

- **vga_sync** module generates the timing and synchronization signals
  - The *hsync* and *vsync* are connected directly to the VGA port
  - These signals drive internal counters that in turn drive *pixel_x* and *pixel_y*
  - The *video_on* signal is used to enable and disable the display

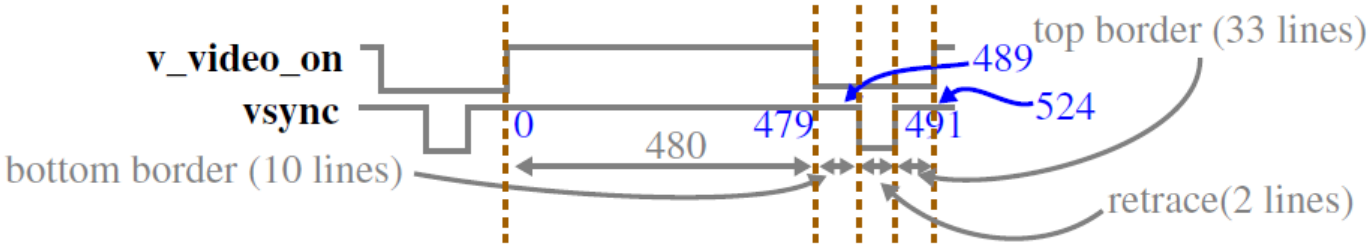
- **pixel generator circuit** logic generates three video signals -- the **rgb** signal
  - The color value is derived from the external control and data signals
  - The vga_sync circuit generates the *hsync* signal, which specifies the time to traverse (scan) a row, while the *vsync* signal specifies the time to traverse the entire screen
  - 640x480 VGA screen with a 25-MHz *pixel rate (*VGA mode)
  - The screen usually includes a small black border around the visible portion
  - The top-left is coordinate (0, 0) while the bottom right is coordinate (639,479)

VGA Display



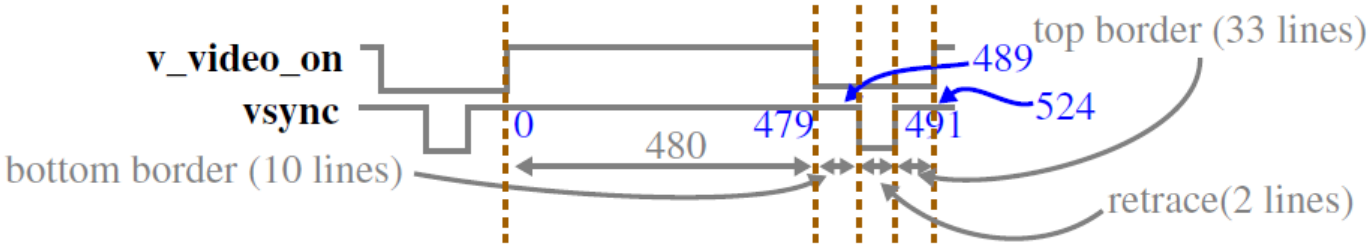
- **Display**: Visible region of screen -- 640 pixels
- **Retrace**: Region in which the electron beam returns to left edge. Video signal is disabled and its length is 96 pixels
- **Right border**: *front porch* (porch before retrace). Video signal is disabled and its length is 16 pixels
- **Left border**: *back porch (*48 pixels): Video signal is disabled and its length is (may differ depending on monitor).

VGA Display



- *hsync* signal is obtained by a special **mod-800 counter** and a decoding circuit.
  - The *counter* starts from the beginning of the display region.
  - This allows the counter's output to be used as the x-axis coordinate or *pixel_x* signal.
  - *hsync* is low for the counter interval 656 (640+16) to 751 (640+16+96-1).
- The *h_video_on* signal is used to ensure that the monitor is black in the border regions and during retrace. It is asserted when the counter is smaller than 640.
- The time unit of the movement is in terms of the horizontal scan lines.
- One period of the *vsync* signal is 525 lines, and has a corresponding set of four regions

VGA Display



- A counter is also used here with the output defining the *pixel_y* coordinate.
- *vsync* goes low when line count is 490 or 491.
- *v_video_on* is asserted only when line count is less than 480.
- We assumed the pixel rate was 25 MHz -- this allows 60 screen refreshes/second (anything less results in flicker).
- s = 60 screens/second * 525 lines/screen * 800 pixels/line = 25.2 Mpixels/sec

VGA Display

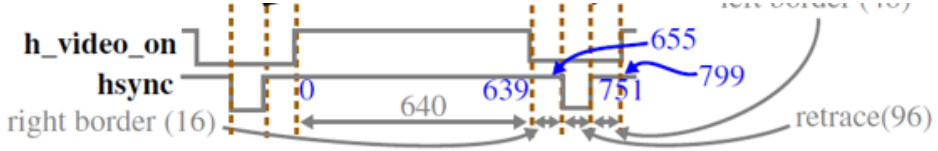
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use work.vga_lib.all;

entity vga is
   port (clk            : in  std_logic;
         rst            : in  std_logic;
         switch         : in  std_logic_vector(9 downto 0);
         img_pos        : in  std_logic_vector(2 downto 0);
         red, green, blue : out std_logic_vector(3 downto 0);
         h_sync, v_sync   : out std_logic;
         video_on       : out std_logic);
end vga;
```

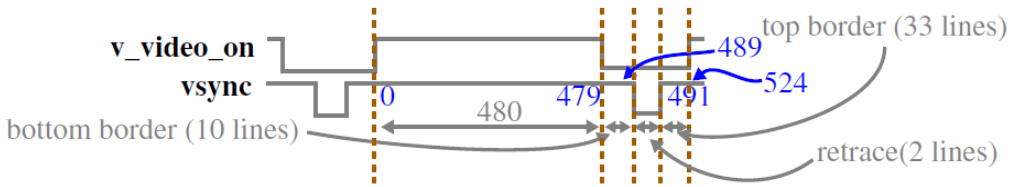
VGA Display

```
architecture default_arch of vga is

  signal v_count : unsigned(COUNT_RANGE);
  signal h_count : unsigned(COUNT_RANGE);

   signal v_en : std_logic;
   signal h_en : std_logic;

-- VGA_SYNC_GEN Signals

  signal v_count_r : natural;
  signal h_count_r : natural;

begin

-- VGA MAIN BEGINS

     process(v_count, h_count)

          begin
              v_en <= ‘0’;
              h_en <= ‘0’;
             row_address <= (OTHERS => '0’);

-- Check the Left-most Y          and the Right-most Y

               if(v_count > to_unsigned(0, 10 )) and (v_count <= to_unsigned(479, 10 )) then

                 -- If within bounds, display
                    v_en <= '1’;

               else
                    v_en <= ‘0’;
               end if;
```

```
 -- Check the Left-most X  and the Right-most X

  if(h_count > to_unsigned(0, 10 )) and (h_count <= to_unsigned(639, 10 ))  then
                    H_en <= '1';
 else
                    H_en <= '0';
 end if;
 end process;
```

```
process(H_en, V_en)
   begin
          if(H_en AND V_en) = '1' then
              red(3 downto 1)   <= switch(2 downto 0);
              green(3 downto 1)  <= switch(5 downto 3);
              blue(3 downto 1) <= switch(8 downto 6);
           else
              red   <= "0000" ;
              blue  <= "0000" ;
              green <= "0000" ;
           end if;
 end process;
```





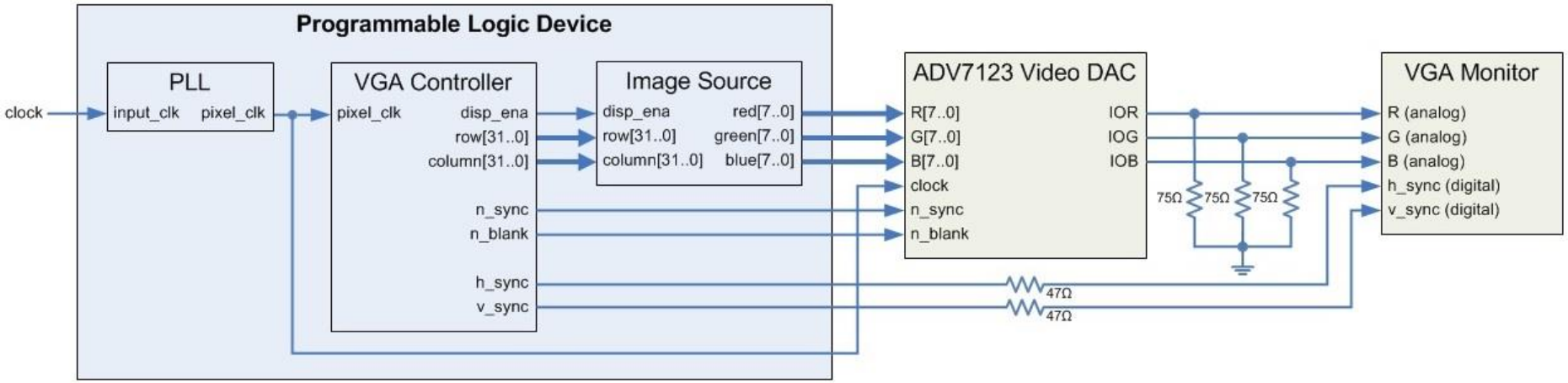
VGA Display

```
-- VGA_SYNC_GEN BEGINS

      process(clk, rst)
          begi

         if(rst = '1') then

            _count_r <= 0;
            _count_r <= 0;

          elsif (rising_edge(clk)) then

              h_count_r <= h_count_r + 1;  --(increment by 1)

          if(h_count_r >= H_MAX+1) then--799 value found in vga_lib file under "H_MAX"

               h_count_r <= 0;

          end if;

         if(h_count_r = H_VERT_INC) then--699 value found in vga_lib file under "H_VERT_INC“

               v_count_r <= v_count_r + 1; -- (increment by 1)
         end if;

         if(v_count_r >= V_MAX+1) then--524 value found in vga_lib file under "V_MAX"

              v_count_r <= 0;

         end if;

       end if;

end process;
```

```
h_count <= to_unsigned(h_count_r, 10);

 v_count <= to_unsigned(v_count_r, 10);

process(h_count_r, v_count_r)

        begin
              if (h_count_r > HSYNC_BEGIN and h_count_r < HSYNC_END) then

                    h_sync <= '0’;

               else

                    h_sync <= '1’;

                end if;

              if (v_count_r > VSYNC_BEGIN-2 and v_count_r < VSYNC_END) then

                    v_sync <= '0’;

               else
                    v_sync <= '1’;

                end if;

               if (h_count_r <= (H_DISPLAY_END ) and v_count_r <= (V_DISPLAY_END )) then

                     video_on <= '1’;

               else
                     ideo_on <= ‘0’;
                end if;

        end process;

-- VGA_SYNC_GEN ENDS

end default_arch;
```

VGA Display

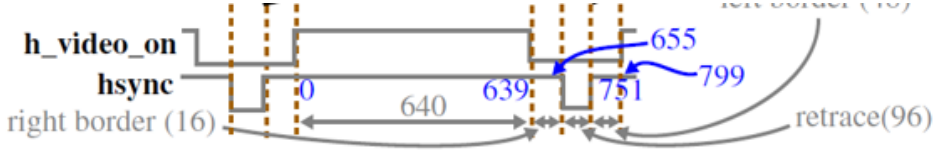
- https://forum.digikey.com/t/vga-controller-vhdl/12794

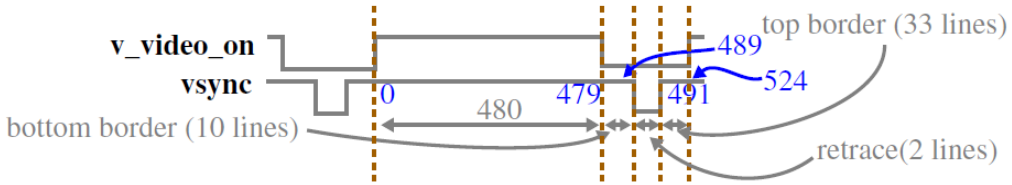


VGA Display

https://projectf.io/posts/video-timings-vga-720p-1080p/

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY vga_controller IS

  GENERIC(
    h_pulse  : INTEGER := 208;    --horiztonal sync pulse width in pixels
    h_bp     : INTEGER := 336;    --horiztonal back porch width in pixels
    h_pixels : INTEGER := 1920;   --horiztonal display width in pixels
    h_fp     : INTEGER := 128;    --horiztonal front porch width in pixels
    h_pol    : STD_LOGIC := '0';  --horizontal sync pulse polarity (1 = positive, 0 = negative)
    v_pulse  : INTEGER := 3;      --vertical sync pulse width in rows
    v_bp     : INTEGER := 38;     --vertical back porch width in rows
    v_pixels : INTEGER := 1200;   --vertical display width in rows
    v_fp     : INTEGER := 1;      --vertical front porch width in rows
    v_pol    : STD_LOGIC := '1'); --vertical sync pulse polarity (1 = positive, 0 = negative)

  PORT(
     pixel_clk : IN  STD_LOGIC;  --pixel clock at frequency of VGA mode being used
     reset_n   : IN  STD_LOGIC;  --active low asycnchronous reset
     h_sync    : OUT STD_LOGIC;  --horiztonal sync pulse
     v_sync    : OUT STD_LOGIC;  --vertical sync pulse
     disp_ena  : OUT STD_LOGIC;  --display enable ('1' = display time, '0' = blanking time)
     column    : OUT INTEGER;    --horizontal pixel coordinate
     row       : OUT INTEGER;    --vertical pixel coordinate
     n_blank   : OUT STD_LOGIC;  --direct blacking output to DAC
     n_sync    : OUT STD_LOGIC); --sync-on-green output to DAC

END vga_controller;
```





VGA Display

```
ARCHITECTURE behavior OF vga_controller IS

 CONSTANT h_period : INTEGER := h_pulse + h_bp + h_pixels + h_fp; --total number of pixel clocks in a row
 CONSTANT v_period : INTEGER := v_pulse + v_bp + v_pixels + v_fp; --total number of rows in column

BEGIN

 n_blank <= '1';  --no direct blanking
 n_sync <= '0';  --no sync on green

  PROCESS(pixel_clk, reset_n)
    VARIABLE h_count : INTEGER RANGE 0 TO h_period - 1 := 0;  --horizontal counter (counts the columns)
    VARIABLE v_count : INTEGER RANGE 0 TO v_period - 1 := 0;  --vertical counter (counts the rows)

  BEGIN

     IF(reset_n = '0') THEN    --reset asserted
          h_count := 0;           --reset horizontal counter
          v_count := 0;           --reset vertical counter
          h_sync <= NOT h_pol;      --deassert horizontal sync
          v_sync <= NOT v_pol;      --deassert vertical sync
          disp_ena <= '0';         --disable display

          column <= 0;            --reset column pixel coordinate
          row <= 0;               --reset row pixel coordinate

      ELSIF(pixel_clk'EVENT AND pixel_clk = '1') THEN

     --counters
          IF(h_count < h_period - 1) THEN    --horizontal counter (pixels)
               h_count := h_count + 1;
          ELSE
               h_count := 0;

      IF(v_count < v_period - 1) THEN  --veritcal counter (rows)
           v_count := v_count + 1;
      ELSE
          v_count := 0;
      END IF;
   END IF;
```

```
--horizontal sync signal
    IF(h_count < h_pixels + h_fp OR h_count >= h_pixels + h_fp + h_pulse) THEN
     h_sync <= NOT h_pol;    --deassert horiztonal sync pulse
    ELSE
     h_sync <= h_pol;        --assert horiztonal sync pulse
    END IF;


    --vertical sync signal
    IF(v_count < v_pixels + v_fp OR v_count >= v_pixels + v_fp + v_pulse) THEN
     v_sync <= NOT v_pol;    --deassert vertical sync pulse
    ELSE
     v_sync <= v_pol;        --assert vertical sync pulse
    END IF;
```

VGA Display

```
--set pixel coordinates
      IF(h_count < h_pixels) THEN  --horiztonal display time
        column <= h_count;          --set horiztonal pixel coordinate
      END IF;
      IF(v_count < v_pixels) THEN  --vertical display time
        row <= v_count;             --set vertical pixel coordinate
      END IF;
      --set display enable output
      IF(h_count < h_pixels AND v_count < v_pixels) THEN  --display time
        disp_ena <= '1';                                  --enable display
      ELSE                                          --blanking time
        disp_ena <= '0';                                  --disable display
      END IF;
    END IF;
  END PROCESS;
END behavior;
```