Lab # 9

VGA Controller

Introduction

VGA Controller is used to control a monitor (PC monitor) and has a simple protocol as we will see in this lab.

Kit parts for this lab
A closer look

VGA Basics

The term VGA really means one of two things depending on how you use the acronym. It’s either a standard 15-pin connector used to drive video devices (e.g. a VGA cable) or it’s the protocol used to drive information out on that cable (e.g. a VGA interface spec.). The interface defines how information is sent across the wires from your board to the VGA device. The cable defines which pins you use on the standard connector for those signals.

The most basic thing to know about VGA is that it is a protocol designed to be used with analog CRT (cathode ray tube) output devices. On these devices the electron beam moves across the screen from left to right as you’re looking at the screen at a fixed rate (the refresh rate defines how fast the beam moves), and also moves down the screen from top to bottom at a fixed rate. While it’s moving across and down the screen, you can modify the Red, Green, and Blue values on the VGA interface to control what color is being painted to the screen at the current location.
So, painting a certain color on the screen is as easy as keeping track of where the beam is, and making sure the R, G, and B signals are at the right values when the beam is over the point on the screen where you want that color.

If you don’t do anything to stop it, the beam will move to the right and bottom of the screen and get stuck there. You can force the beam to move back to the left by asserting an active-low signal called hSync (horizontal sync). You can force the beam to move back to the top of the screen by asserting an active-low signal called vSync (vertical sync).

Because the beam moves at a fixed rate (defined by the monitor’s refresh rate), you can keep track of where the beam is on the screen by counting clock ticks after the hSync and vSync signals.

So, the basics of the VGA control/timer circuit are just a pair of counters to count horizontal ticks and vertical ticks of the VGA clock. How many ticks are there? That depends on how fast your clock is, and how many pixels you want to paint during the time the beam moves across the screen.

The basic (ancient) standard for “plain” VGA is 640 pixels on each line, and 480 lines down the screen. This is “640x480” mode. Figure 1 shows a 640x480 screen, and the horizontal sync (hSync) timing required to make it work.

After the hSync pulse, you must wait for a certain number of ticks before painting pixels to the screen. This gives the beam time to get back to the left and start moving forward again. This time is called the “back porch” because it’s in back of the hSync timing pulse. Then you count 640 pixels as the beam moves. After the 640th pixel, you wait for some amount of time (this is the “front porch” because it’s in front of hSync), then assert the hSync signal (asserted low) for a certain amount of time.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Vertical Sync</th>
<th>Horizontal Sync</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>Clocks</td>
</tr>
<tr>
<td>(T_S)</td>
<td>Sync pulse time</td>
<td>16.7 ms</td>
<td>416,800</td>
</tr>
<tr>
<td>(T_{DISP})</td>
<td>Display time</td>
<td>15.36 ms</td>
<td>384,000</td>
</tr>
<tr>
<td>(T_{PW})</td>
<td>Pulse width</td>
<td>64 (\mu s)</td>
<td>1,600</td>
</tr>
<tr>
<td>(T_{FP})</td>
<td>Front porch</td>
<td>320 (\mu s)</td>
<td>8,000</td>
</tr>
<tr>
<td>(T_{BP})</td>
<td>Back porch</td>
<td>928 (\mu s)</td>
<td>23,200</td>
</tr>
</tbody>
</table>
VHDL Lab

Nour El-Deen I. Jaber

---

Diagram showing time intervals and pixel coordinates.

- $T_S$
- $T_{disp}$
- $T_{fp}$
- $T_{dp}$

---

VGA Display

- Pixel 0,0
- Pixel 0,639
- Pixel 479,0
- Pixel 479,639

- 640 pixels are displayed each time the beam traverses the screen.

- Retrace: No information is displayed during this time.

- Stable current ramp: Information is displayed during this time.

- Total horizontal time
- Horizontal display time
- Retrace time

- "back porch"
- "front porch"

---

Horizontal sync signal sets the retrace frequency.
RGB Colors

The Spartan3e board has a VGA interface with one wire connected to the Xilinx part for each of the R, G, and B signals. This means you can make a generous eight colors on the screen by turning on combinations of the R, G, and B. Figure below shows the colors you can get with this simple interface.

<table>
<thead>
<tr>
<th>VGA_RED</th>
<th>VGA_GREEN</th>
<th>VGA_BLUE</th>
<th>Resulting Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Black</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Blue</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Green</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Cyan</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Red</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Magenta</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Yellow</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>White</td>
</tr>
</tbody>
</table>

Hardware Connection
Entity Declaration – see kit pins –

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use ieee.numeric_std.all;

entity vga is
  Port ( 
    CLK50_in : in STD_LOGIC;
    HS : out STD_LOGIC;
    VS : out STD_LOGIC;
    r,g,b : out std_logic
  );
end vga;
```

Architecture Processes and signals

```vhdl
signal clk25: std_logic :='0';
signal H_counter_value : integer :=0;
signal V_counter_value : integer:=0;
signal x : integer:=0;
signal y : integer:=0;

-- Process to generate 25MHz clock out of 50 MHz clock
```

```vhdl
begin
  if clk50_in'event and clk50_in='1' then
    clk25 := not clk25;
  end if;
end process;
```
Process to generate Hsync and Vsync

HS_VS_generator :process (clk25)
begin
  if clk25'event and clk25='1' then
    H_counter_value <= H_counter_value + 1;
    if (H_counter_value = 800) then
      H_counter_value <= 0;
      V_counter_value <= V_counter_value + 1;
    end if;
    if (V_counter_value = 521) then
      V_counter_value <= 0;
    end if;
    x <= H_counter_value-143;
    y <= V_counter_value-31;
    x_ball<= x - ball_x_l+1;
    y_ball<= y - ball_y_u+1;
    if (H_counter_value < 96) then
      HS <= '0';
    else
      HS <= '1';
    end if;
    if (V_counter_value < 2) then
      VS <= '0';
    else
      VS <= '1';
    end if;
  end if;
end process;
**Process to assign colors**

```vhdl
process(clk25,H_counter_value,V_counter_value,ball_on,c)
begin
  if (H_counter_value >=144
      and H_counter_value < 783
      and V_counter_value >=31
      and V_counter_value <510)
  then

    if ball_on='1' then

      if c='1' then
        r<=ball_rgb(2);
        b<=ball_rgb(1);
        g<=ball_rgb(0);
      else
        r<='1';
        g<='0';
        b<='0';
      end if;
    else
      r<='1';
      g<='0';
      b<='0';
    end if;

  end if;
end process;
```

You will need letters stored in a ROM I’ll attach the complete file on my web page.
Sample output

Now, let’s connect the circuit