

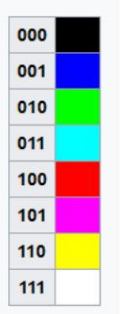
## WORK: VGA DOOR CONTROLLER

The aim of this work is to implement a VGA port controller (VGA controller). The auditor includes the synchronization circuit and a simple graphics output circuit that will be used to verify the implementation by driving the VGA port with suitable images test.

### Basic VGA port concepts

A VGA port has the following five active signals: one horizontal sync signal (*HSync*), one signal Vertical Sync (*VSyn*c) and three analog video signals *Red (R)*, *Green (G)* and *Blue (B)*. If the analog video signal represented by N-bits, can be converted to  $2^N$  voltage analog levels. Thus, the three video signals can create a palette of  $2^N$  different colors. For example, if we used 1-bit for each color, the possible color combinations for RGB components are shown in the table below.

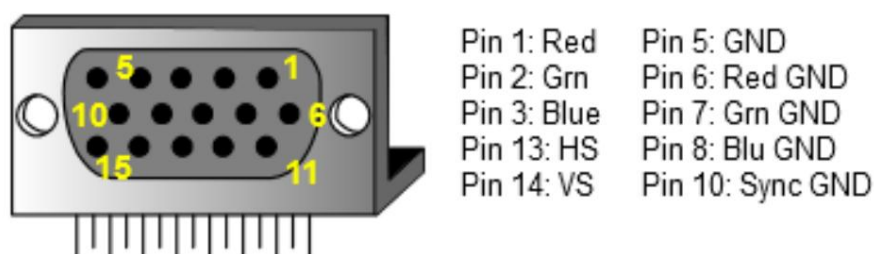
Red (R)	Green (G)	Blue (B)	Resulting color
0	0	0	black
0	0	1	blue
0	1	0	green
0	1	1	cyan
1	0	0	red
1	0	1	magenta
1	1	0	yellow
1	1	1	white



**Table 1. 3-bit VGA color palette**

The Zedboard development card uses 4-bits for each of the three analog signals (ie 16 different shades for each of R, G, B), ie 12-bit color, so in total can represent  $2^{12} = 4096$  different colors.

To convert from digital to analog for each of the three signals, the development card uses a resistor-ladder network of 4 FPGA terminals. The interface is done with a DB15F connector. The connector and connections to the FPGA terminals are shown in image below.



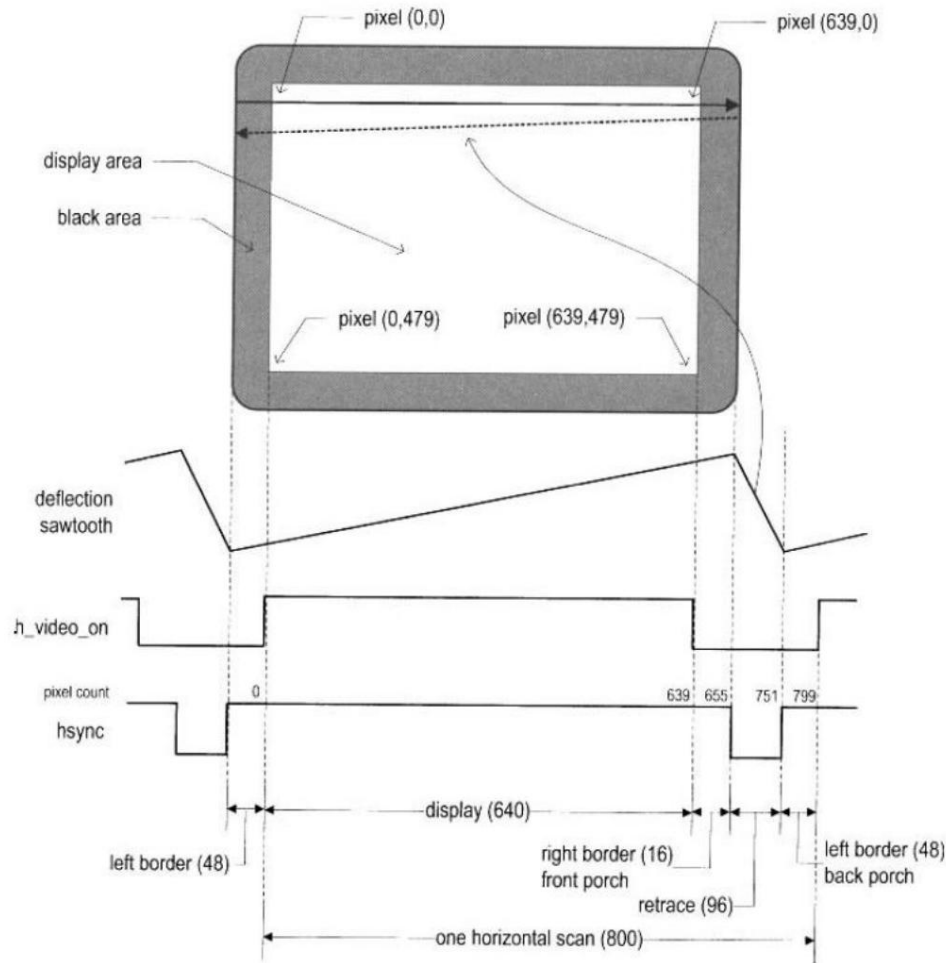
VGA Pin	Signal	Description	Zynq Pin
1	RED	Red video	V20, U20, V19, V18
2	GREEN	Green video	AB22, AA22, AB21, AA21
3	BLUE	Blue video	Y21, Y20, AB20, AB19
4	ID2/RES	formerly Monitor ID bit 2	NC
5	GND	Ground (HSync)	NC
6	RED_RTN	Red return	NC
7	GREEN_RTN	Green return	NC
8	BLUE_RTN	Blue return	NC
9	KEY/PWR	formerly key	NC
10	GND	Ground (VSync)	NC
11	ID0/RES	formerly Monitor ID bit 0	NC
12	ID1/SDA	formerly Monitor ID bit 1	NC
13	HSync	Horizontal sync	AA19
14	VSync	Vertical sync	Y19
15	ID3/SCL	formerly Monitor ID bit 3	NC

**Figure 1. DB15F connector and Zedboard VGA connectors**

## VGA synchronization

A VGA controller contains a synchronization circuit for the output of *HSync* and *VSync* signals connected to the VGA port for horizontal and vertical scan control, respectively. These two signals are decoded by internal horizontal and vertical scan meters that determine the position of the current pixel on the screen. The *HSync* signal specifies the time required to scan a horizontal bar and the *VSync* signal specifies the time required to scan the entire screen. The following paragraphs present in detail the waveforms of the timing signals for the horizontal and vertical scanning of a 640x480 VGA image.

## Horizontal Sync



**Figure 2. Horizontal scan time for VGA 640x480 screen**

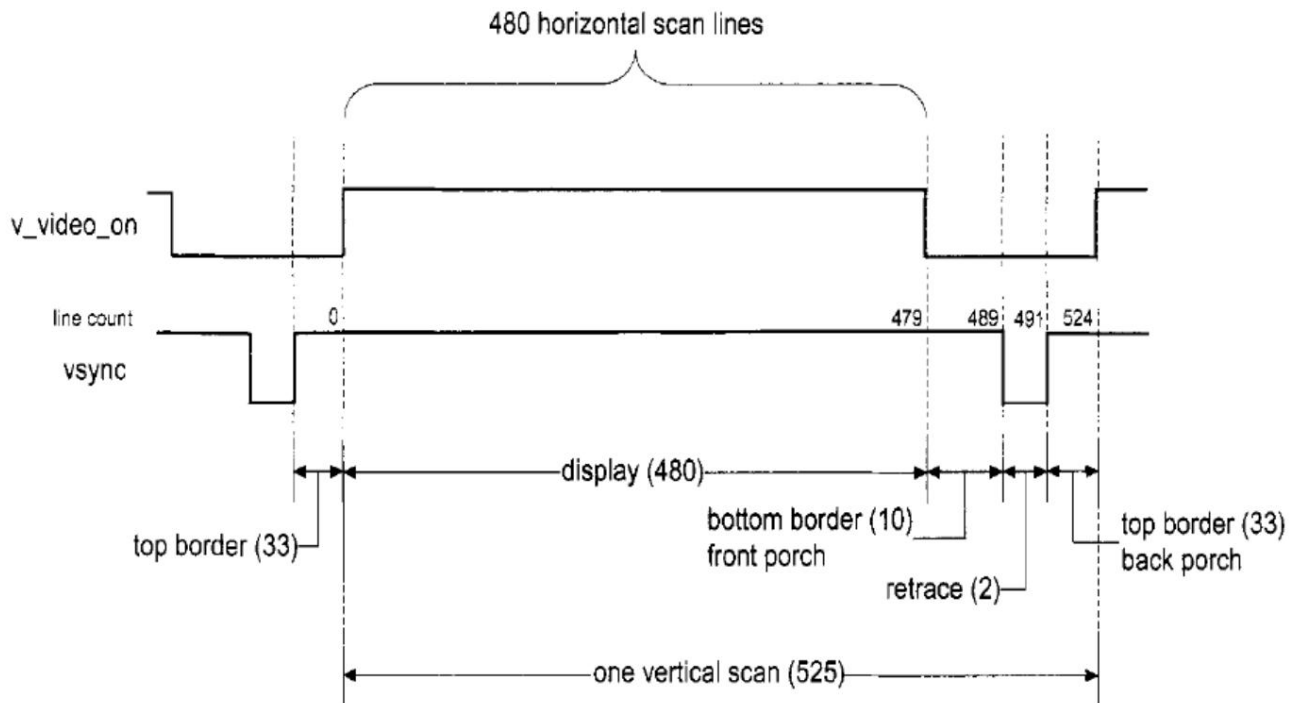
A period of the *HSync* signal contains 800 pixels and is divided into four regions:

- Display: refers to the area where the pixels appear on the screen. The resolution is **640** pixels (0-639).
- Retrace: refers to the area where the pixels return to the left edge. The control signal (*video\_on*) must be turned off (ie black). The range is **96** pixels (656-751)
- Right edge: Also known as *front porch* (ie front porch before reset). The control signal (*video\_on*) must be off. The range is **16** pixels (640-655).
- Left end: Also known as back porch. The control signal (*video\_on*) must be off. The range is **48** pixels (752-799).

The *HSync* synchronization signal can be generated by a modulo-800 meter and suitable decoding. Starting the measurement from the beginning of the active area allows the meter to be used as a horizontal coordinate. The *HSync* signal is active-low and is activated when the meter value is  $656 > HSync \leq 751$ .

The control signal (*h\_video\_on*) is activated when the horizontal coordinate is in the active area (display), ie when the value of the relevant meter is less than 640.

## Vertical Synchronization



**Figure 3. Vertical scan timing for VGA 640x480 screen**

A period of the *VSync* signal contains 525 lines and is divided into four regions:

- **Display:** refers to the area where the horizontal bars appear on the screen. The range is **480** lines (0-479).
- **Retrace:** refers to the area where the lines return to the top of the screen. The video signal must be left off (ie black). The range is **2** lines (490-491).
- **Bottom edge:** Also known as *front porch* (ie front porch before reset). The signal video must be turned off. The range is **10** lines (480-489).
- **Upper end:** Also known as back porch (ie front porch after reset). The video signal must be off. The range is **33** pixels (492 -524).

Similar to horizontal scanning, the *VSync* synchronization signal can be generated by a mod-525 meter and appropriate decoding. Again, starting the measurement from the beginning of the active area allows the meter to be used as a vertical coordinate. The *VSync* signal is active-low and is activated when the meter value is between 490 and 491.

The control signal (`v_video_on`) is activated when the vertical coordinate is in the active area (display), ie when the value of the relevant meter is less than 480.

## Pixel Rate calculation

To calculate the pixel rate we need the following:

p: number of pixels in a horizontal scan bar. For 640x480 resolution,  $p = 800$  pixels / line.

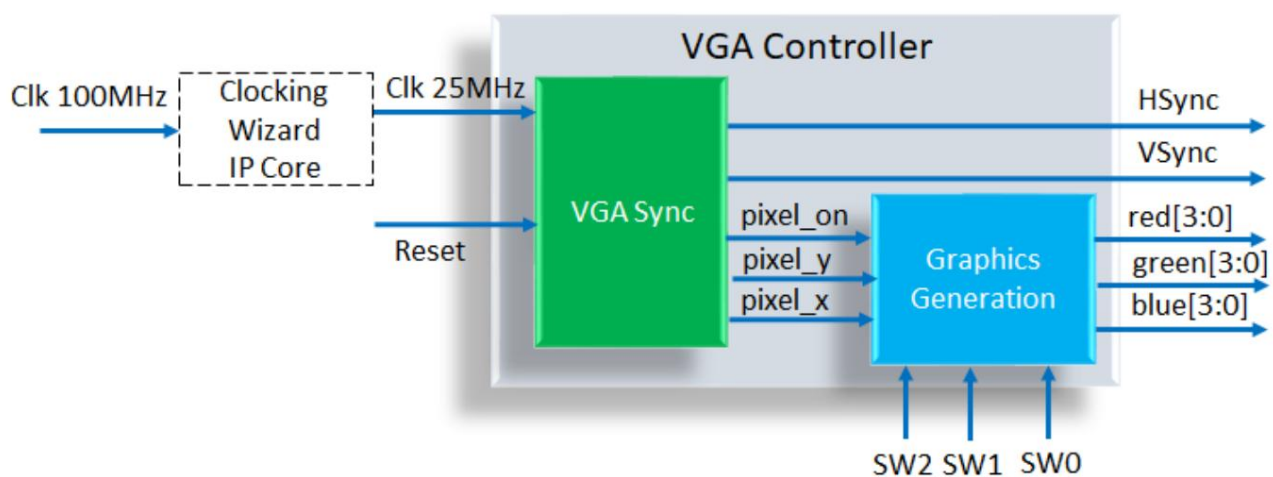
l: number of lines on the screen (vertical scan). For resolution 640x480,  $l = 525$  lines / frame.

f: screen refresh rate. To avoid flickering, set it to  $f = 60$  frames / sec. The human eye requires a frequency of at least 30 frames / sec.

Therefore, the pixel rate for a 640x480, 60Hz screen is  $= p \times l \times f \approx 25$  Mpixels / sec (screens allow a tolerance of 5%), ie we need a 25MHz clock.

## Architecture of a VGA controller

The proposed architecture for the VGA controller is shown in the image below.



**Figure 4. Proposed VGA controller architecture**

The *VGA Sync* module accepts a 25 MHz clock as input. When the VGA controller is implemented on the Zedboard development card you will need to add the Clocking Wizard IP Core to your design Xilinx which will accept as input the 100 MHz Zedboard crystal clock and will produce the 25 MHz clock.

The *VGA Sync* module produces the necessary synchronization signals (*HSync*, *VSync*) as well as the signals that specify the horizontal (*pixel\_x*) and vertical coordinates (*pixel\_y*) and the *pixel\_on* signal that is enabled in the active area of the image (display).

The *Graphics Generation* unit will accept the pixel coordinates, data inputs / controls that determine the contents of the screen and will produce the 3 RGB color components that will drive the VGA port and always as long as the coordinate is enabled (*pixel\_on*).

In general, a *Graphics Generation* unit produces content with three different approaches:

a) The *bit-mapped* approach uses Video Memory (VRAM) or frame buffer to store the data that will be displayed on the screen. Each pixel corresponds to a word in memory that is properly addressed by the `pixel_x` and `pixel_y` signals that specify the coordinates. VRAM contents are constantly updated by a Graphics module

Processing Unit) which writes the memory with the appropriate data. A circuit continuously reads the contents of the memory and routes its contents, ie the RGB signals to the output. This architecture is very common but has high memory requirements. For example, 640x480 resolution requires 307200 pixels. For monochrome display (1-bit color) need 307200 bits, for 3-bit color need 921600 bits and for 12-bit color (supported by Zedboard) need 3686400 bits.

b) To reduce memory requirements, an alternative is to use a *tile mapped* approach where multiple bits are grouped together to define a *tile*. For example, we can define a group of 8x8 pixels as a tile and the 640x480 screen becomes a screen with 80x60 tiles. So only 4800 words are needed for *tile memory*. The number of bits in the word depends on the number of tile patterns. For example, if you set 32 patterns for each tile, then 5 bits per tile are enough

the size of the memory for the 4800 tiles becomes  $5 * 4800 = 24000$  bits. Usually for storing tiles patterns a ROM is used as *pattern memory*. For the  $8 \times 8 = 64$  pixels of a monochrome (1-bit) tile 64 bits are needed, while for all the patterns of a tile  $32 \times 64 = 2048$  bits. That is, a total of  $24000 + 2048 = 26048$  bits are required which is much less than the 307200 bits of the bit-mapped approach. A typical example of using this architecture is for text display (display).

c) Some applications need to display "simple" graphics with few and "simple" objects. In this case, to avoid wasting memory resources on storing a screen that is largely empty, we can produce these simple objects dynamically with appropriate

Graphics Generation circuits. This approach is called *object-mapped*.

According to this approach, individual object production units are used that keep the coordinates for the object they describe, while the use of a collision detection and handling unit and logic for the appearance of activated objects is often required.

Very often, the above three approaches are used in combination. For example, a *bit-mapped* approach can be used to generate the background (eg from ROM), an approach *object-mapped* to produce basic (simple) graphics and a *tile-mapped* approach to display text on the screen at the same time.

In the work you will use approach (c) by implementing a simple graphics production unit (Graphics Generation) for the display of simple test images.

The *Graphics Generation* unit of the work will produce dynamically, following the *object mapped approach*, the 3 RGB color components depending on the images we want to send to the VGA port and always if the coordinate is enabled (pixel\_on).

Drive the four bits of each color component with the same bit (eg Red = "0000" or Red = "" 1111 ").

The unit produces 8 different test images depending on the position of the external SW2 switches, SW1, SW0 of the card according to the table below.

SW2	SW1	SW0	Picture
0	0	0	Color Checkerboard
0	0	1	Color Vertical Bars
0	1	0	Color Horizontal Bars
0	1	1	Red Vertical Bards
1	0	0	Red Horizontal Bards
1	0	1	Red
1	0	1	Green
1	1	1	Red Contour

Table 2. Test images



Figure 6. *Color Checkerboard* test image (each square has dimensions 64x64) for "000" input



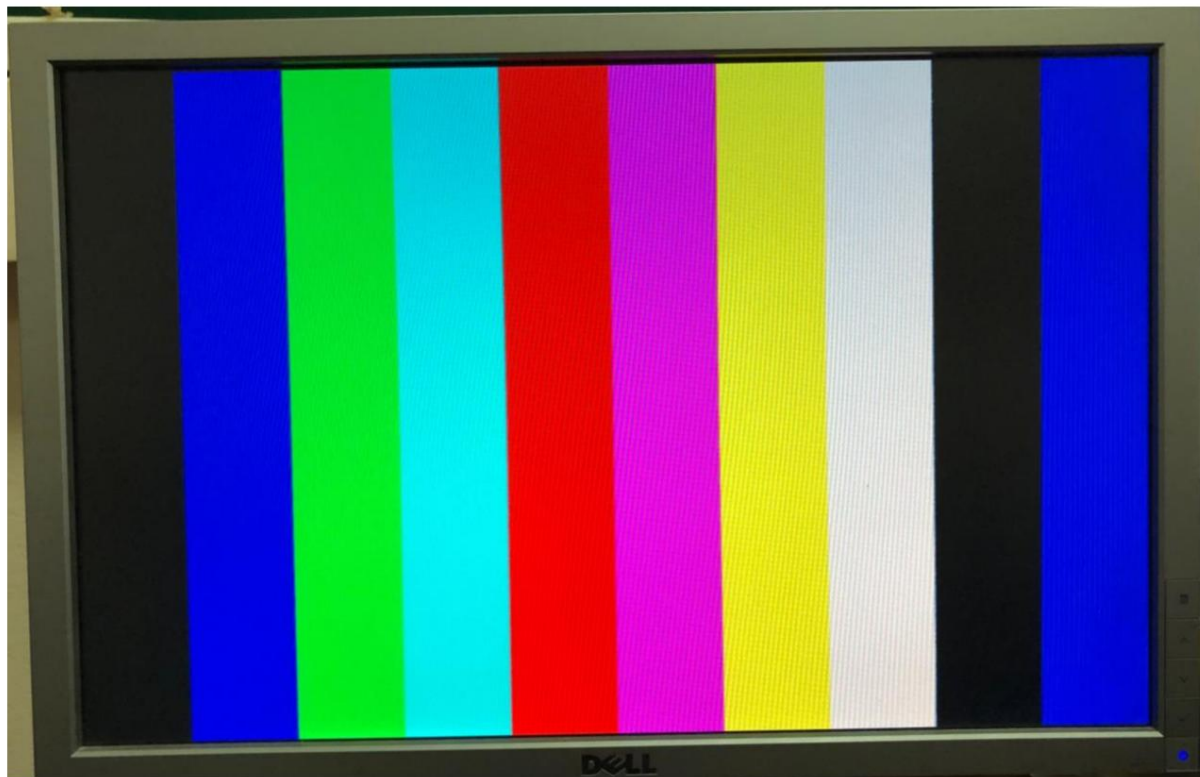


Figure 7. Vertical Bars test image (the width of each bar is 64 pixels) for “001” input



Figure 8. *Horizontal Bars* test image (the height of each bar is 64 lines) for “010” input

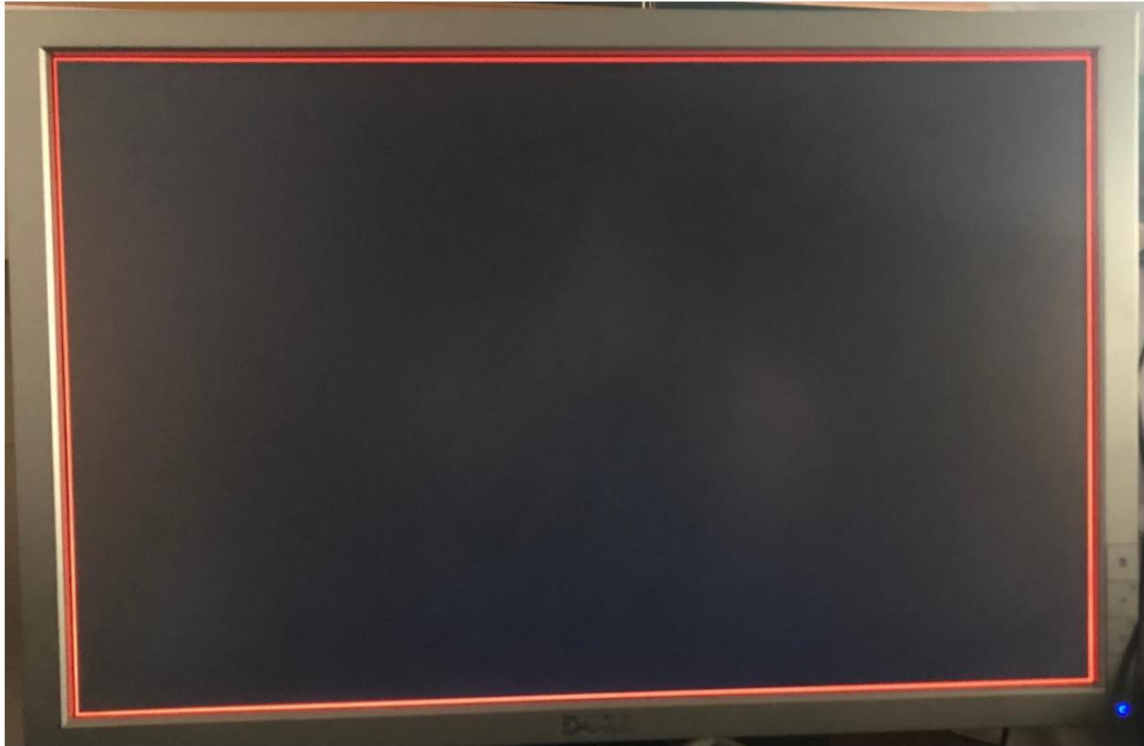




Figure 9. *Vertical Red Bars* test image (the width of each red bar is 64 lines) for "011" input



Figure 10. *Horizontal Red Bars* test image (the height of each red bar is 64 lines) for "100" input



**Figure 11. Horizontal *Red Contour* test image for “111” input**

- Use the template for the top level entity of the VGA controller (entity vga\_ctrl) from the e-class.
- **The controller outputs leading to the VGA port are registered, but taking care at the same time be synchronized with respect to cash prices.**
- Set the timing parameters for the dimension 640x480 as constants.
- You will be given a suitable VHDL testbench to verify the correct design by simulation (entity vga\_test) which you will need to modify to test them all  
Option input combinations (SW2, SW1, SW0) starting the tests from the «Red Contour »and« Vertical Red Bars »and« Horizontal Red Bars »as they will facilitate you in debugging.
- For pin constraints, consult the card manual.
- Test the operation of the circuit first by simulation. Check "with the eye" by observing waveforms that the timing for *HSync* and *VSync* meets the specifications for 640x480 resolution. Also check that the timing for the RGB component signals is correct and that they are off (0) outside the display area.

- Next, use the free VGA simulator tool given to you in the e-class for to check the result of your design by uploading the **write.txt** file that is located in the sim\_1 \ behav \ xsim subfolder of your project that contains the simulation outputs. At parameters of VGA Simulator to set Back Porch X = 48, Back Porch Y = 33 and Pixel Clock Rate = 25MHz. All test images should be presented in the technical reference on frames of the demonstration of proper operation.

## VGA Simulator

**Need Help?** [Here is a blog post that will get you acquainted with the tool](#)

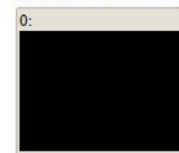
Log File:  write.txt [Download Example Log File](#)  
Resolution X:  Resolution Y:   
Pixel Clock Rate (MHz):

Fine tuning:

Back Porch X:  48  
Back Porch Y:  33

Working...  
\$12000 / \$12000

Click on frames to review:



[Download current frame](#)



The delivery of the work will be done **exclusively** through the e-class platform within the specified one deadline and will include:

- a) Technical report according to the template found in the e-class
- b) The VIVADO project is compressed