|  |  |
| --- | --- |
| *Title:* | ***Lab #10: FPGA PONG – VGA Controller*** |
| *Name:* |  |

# INTRODUCTION

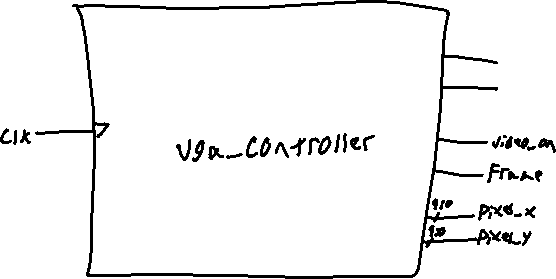
You need to use Xilinx Vivado.

In this lab, we will output the signals required to display images on a VGA monitor. To keep the lab simple, you will only need to build the component and instantiate it. There is a new modified test bench that you can use to actually test your display.

Start by reading Section 7 of the BASYS Reference Manual to gain an understanding of how the seven segment display works.

Also look at the lecture notes on how VGA signal is produced.

Here is a quick overview of what we are trying to accomplish:



One deviation from the lecture that you may notice is that we are omitting the RGB values. This is because our top module will determine the color depending on whether the ball or paddle are supposed to be drawn. Instead, we have the VGA controller send a signal whenever it is drawing the frame itself, and a pulse whenever a frame is finished. Additionally, we now will output the current pixel being displayed, which is used in the ball and paddle modules later to determine if they should be drawn or not.

Another small note: the VGA signal for a 640x480 display requires a timing of 25.125MHz to properly function. We will use clock\_divider’s 25MHz output as a rough approximation of the required timing.

# Procedure

1. In your project, delete the current test bench file. Import the new one and set it as top.
2. Create a new component called “vga\_controller.vhd”. We will not be directly giving you the signal names this time; it is your responsibility to make sure that they will declare and instantiate correctly in your own code, since this will not be directly seen by the test bench.
   * You will need all of the ports shown in the above diagram.
3. VGA signals require very precise timing. Even a single pixel can stop the entire display from working as intended. As such, you will be given all of the constants needed to accurately display the screen.

constant H\_Visible : integer := 640;

constant H\_FPorch : integer := 16;

constant H\_Sync : integer := 96;

constant H\_BPorch : integer := 48;

constant H\_Total : integer := 800;

constant V\_Visible : integer := 480;

constant V\_FPorch : integer := 10;

constant V\_Sync : integer := 2;

constant V\_BPorch : integer := 33;

constant V\_Total : integer := 525;

For a reminder of what these constants actually represent, look to the lecture notes, specifically on scanning on the screen. For a quick refresher, Visible is the actual screen space (640x480). The front porch follows on the right and bottom sides, and is used as a buffer before the sync (sometimes called retrace) signal. Then the Hsync on the right and the Vsync on the bottom occur. Before LED screens, this was the time when the electron beam would shift back across the screen to get back to the left or top side. New monitors keep this as a convention for timing, with standards like HDMI using more sophisticated timing sequences. Lastly, the back porch is the top and left side before the visible area, and is used as more buffer to stabilize the inputs.

We will not be adhering to this perfectly. Instead of having the back porches be on the left and top, for the purposes of our component we will pretend they are on the right and bottom. This means our first pixel at (0,0) will be in the visible space, and the last pixel will be at (639, 479). This makes the logic for the displays significantly easier.

# Design the Component

This component can get very confusing very quickly if you do not keep in mind that all of our counters start at 0 and count up to N-1, not N (You could also technically start it at one, and count up to N, but this is not common industry practice).

1. You will need two signals here to act as the actual counters. Both need to be big enough to fit the resolution of the entire screen (they need to fit H\_Total and V\_Total). Name these something like H\_Count and V\_Count, and initialize them to zero.
2. We will have one PROCESS block, triggered on the rising edge of the clock. In this statement, we will first increment the horizontal counter. If it has crossed the entire horizontal resolution, we will reset it to zero.
3. Still in the process block, whenever the horizontal counter resets (so it has finished scanning one line) we will increment the vertical counter. When this counter crosses the entire vertical resolution, we will reset it as well.
4. Now, we are done with the process blocks! Now hop into the continuous assignments. pixel\_x and pixel\_y can just be what the counters have in them already, since it already is tracking one pixel at a time.
5. Frame is a pulsed signal, that updates once per frame. You want this signal to be on when both the horizontal and vertical counters last scan the visible area:



1. video\_on is used to inform the ball and paddle modules that it is currently displaying the visible area of the monitor. This is used in calculations later when those modules are deciding if they want to draw their icon or not. This needs to be on whenever the current pixel count is within the visible area.
2. Hsync must be turned on whenever the counter is between the horizontal front and back porch signals. Vsync must be turned on whenever the counter is between the vertical front and back porch signals. These tend to be the part where students forget zero-indexing, so be very careful when choosing (>, >=, <, <=, etc!)
   1. It helps to think through “when is the first time this signal would be ‘1’? when does it turn to ‘0’
3. Lastly. Now that the VGA component should be finished, head back to top.vhd. declare and instantiate it, remembering that the clk on the vga\_controller should be the 25MHz clock from the clock divider. In our case, just route pixel\_x, pixel\_y, frame, and video\_on to signals, and route the h and v sync to the ports.

# Compile and Test it!

You should now be able to generate a bitstream and program your FPGA. If your code works, you will be able to [Unsure, Something along the lines of typing on screen?].

Show your TA and have them sign the check off sheet.

**Troubleshooting**

If you are having trouble, run through the following troubleshooting steps:

* Does your display show any output at all? (black screen, flickering, etc?)
  1. You are almost there! This is most likely due to incorrect counting on your sync, video\_on, or frame counters. Look carefully at them, try to see when they turn on and when they turn off. (keep in mind that they need to go from start to N-1! Counting up 10 times would be a 0 -> 9 counter!)
* Are you getting no display?
  1. Things get a bit difficult here, any number of things can be wrong. To start, check your signal names and port names carefully. Ensure that they are routed to the correct places on top.vhd
  2. Next, check the constants on your vga controller. This website has a list of all of the pixel counts needed.
     1. <http://www.tinyvga.com/vga-timing/640x480@60Hz>
  3. Check how you’re counting your horizontal and vertical counters carefully. Make sure that the vertical counter only increments after the entire horizontal row is drawn. Make sure they’re both going to the proper values (640x480 means h counter goes to 639, v counter goes to 479).
  4. Follow this process for all of your counters, make sure that you pay very close attention to how they increment.
  5. Check your conditional assignments. make sure video\_on is only on when the pixel count is in the visible area. Make sure frame *only* pulses once per frame.