# VGA Display Controller Project

**What is the project all about?**

The project that I chose is about designing a VGA controller to display something on a monitor. I am going to implement a circuit that will generate VGA signals to display an image on the monitor at a certain resolution. A VGA controller handles VGA signal timings and is written in VHDL when used on FPGAs. VGA controllers use a pixel clock at a frequency required of the VGA mode. The VGA controller has all the signal timings required to control the interface. These timings usually have to do with the horizontal and vertical signals and color magnitudes. The color magnitudes are analog signals sent over R, G, and B wires. Most VGA displays can use different resolutions, but a VGA controller will dictate the resolution by controlling the timing signals to control a raster pattern. A raster pattern is an area that is scanned in a rectangular pattern from top to bottom and side to side. This way it can create the image on the screen.

**How much you have implemented already in this project?**

So far, I have implemented a couple of ports and a couple of behavioral logics. This is a combination of different source codes that I have found. It compiles and can be programable on the FPGA, but the FPGA does not do anything. When I connect the VGA to a monitor nothing happens. It says that there is no image or anything.

```
entity vga_ctrl is
    Port ( CLK_I : in STD_LOGIC;
           VGA_HS_O : out STD_LOGIC;
           VGA_VS_O : out STD_LOGIC;
           VGA_RED_O : out STD_LOGIC_VECTOR (3 downto 0);
           VGA_BLUE_O : out STD_LOGIC_VECTOR (3 downto 0);
           VGA_GREEN_O : out STD_LOGIC_VECTOR (3 downto 0);
           PS2_CLK      : inout STD_LOGIC;
           PS2_DATA     : inout STD_LOGIC
           );
end vga_ctrl;
```

These are the ports that I have declared in my program. I have the horizontal sync and the vertical sync declared standard logic. These are outputs that are connected to the VGA connector, using the last two pins. I also have the clock declared, and it is a standard in. I have tried connecting to the clock ports, but it looks that with or without it does not do much. The clock ports are the first two pins under the first section of pins. I even have the RGB signals declared individually. Each signal is an output standard logic vector. Each is a vector that goes from 3 down to 0. The ports declared are all assigned to the pins on the VGA connector sections. The first bundle under the VGA pins are for the red signals. The second is for the green signal and lastly is the blue signal. Since the VGA signals are declared as vectors of 4 bits, they will need 4 pins each. Each signal will start from 0 to 3. I have also declared the height and width of my screen, which is 640x480, and trying for 60hz.

```
60
61      --***640x480@60Hz***--
62      constant WIDTH : natural := 640;
63      constant HEIGHT : natural := 480;
64
```

In most example codes that I have seen, they usually declare the dimension of the screen that is going to be used. Some even have multiple dimensions declared at the same time. This is all in case they have different screen sizes. I went with a 640x480 screen size because it is the smallest size, where the screen will come out in standard quality.

I also have a second set of code that I am testing doing the same thing. I have a clock that has been declared on this one. Also, horizontal and vertical sync, that have been declared as output standard logic. Lastly, the red, green, blue signals are declared as standard logic vectors, that go from 3 down to 0. The clock is also a standard logic vector going from 1 down to 0.

```
  Port (
    clk_24: in std_logic_vector(1 downto 0);
    vga_hs, vga_vs: out std_logic;
    vga_red, vga_green, vga_blue: out std_logic_vector(3 downto 0)
  );
end vga_2crtl;
```

In the same file, I have a second file called through the component. It has a clock that is standard logic. It also has horizontal and vertical sync in standard logic. Lastly, it has declared the red, green, blue signals as standard logic vectors that go from 3 down to 0.

```
component vga_sync is
    Port(
        clk: in std_logic;
        hsync, vsync: out std_logic;
        red, green, blue: out std_logic_vector (3 downto 0)
    );
end component vga_sync;
```

The second file called vga_sync has all the timings for a 640x480 display. I made horizontal and vertical positions that have a range. The horizontal covers from 0 to 800 and the vertical goes from 0 to 525. These positions are pixel positions. 800 is the coverage of the whole line because it covers the visible area, front porch, sync pulse, and the back porch. 525 is the coverage of the whole frame which also covers the visible area, front porch, sync pulse, and back porch.

## Horizontal timing (line)

Polarity of horizontal sync pulse is negative.

| Scanline part | Pixels | Time [µs] |
|---|---|---|
| Visible area | 640 | 25.422045680238 |
| Front porch | 16 | 0.63555114200596 |
| Sync pulse | 96 | 3.8133068520357 |
| Back porch | 48 | 1.9066534260179 |
| Whole line | 800 | 31.777557100298 |

## Vertical timing (frame)

Polarity of vertical sync pulse is negative.

| Frame part | Lines | Time [ms] |
|---|---|---|
| Visible area | 480 | 15.253227408143 |
| Front porch | 10 | 0.31777557100298 |
| Sync pulse | 2 | 0.063555114200596 |
| Back porch | 33 | 1.0486593843098 |
| Whole frame | 525 | 16.683217477656 |

We move the horizontal positions in till we reach the end of the line. When you reach the end of the line you reset the horizontal position back to zero and increase the vertical position by one. You keep doing this in till you hit the last possible point which would be the 800x525 pixel. When you hit the pixel, you reset everything back to zero and start over at the 0x0 pixel. Next, you program the synchronization. The horizontal sync is low when it is on the front and back porch. Horizontal sync is high when it is between the front and back porch. The front porch ends at pixel 16 and the back porch ends at pixel 112. That means that in-between pixel 16 and 112 the horizontal sync should be high. It is the same thing for the vertical sync. The vertical sync is low between the 0 and 12th lines. It is high when it not between the 0 and 12th lines. The last thing to show is when the RGB signal is low and when it is high. The signal should be low when it is on the front and back porch. All 3 color channels will hold a zero value when it's between this and will hold a high value otherwise.

```
if (clk'event and clk='1')then
--Scanning the position
    if(hpos < 800)then
        hpos<=hpos+1;
    else
        hpos<=0;
        if(vpos < 525)then
                vpos<=vpos+1;
            else
                vpos<=0;
        end if;
    end if;
--Starting the synchronization
    if(hpos>16 and hpos<112)then
        hsync<='0';
        else
            hsync<='1';
    end if;
    if(vpos>0 and vpos<12)then
        vsync<='0';
        else
            vsync<='1';
    end if;
```

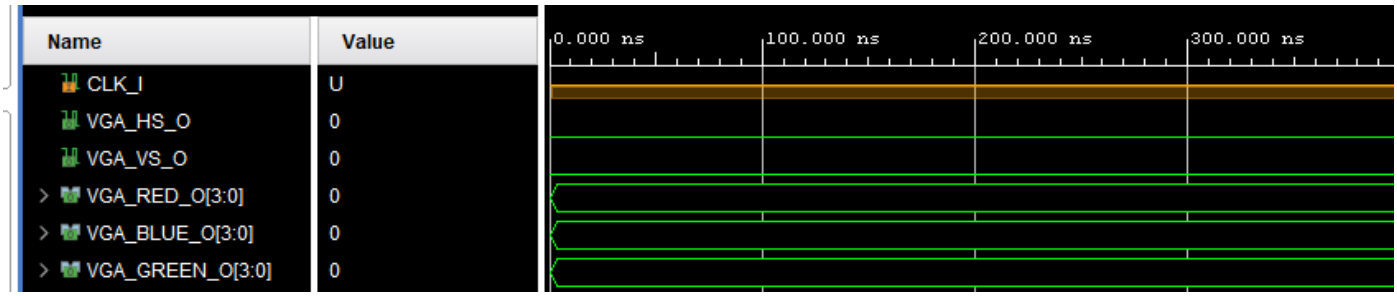
```
--show when rgb signals will be zero
if((hpos>0 and hpos<160) or (vpos>0 and vpos<45)) then
    red <= (others=>'0');
    green <= (others =>'0');
    blue <= (others => '0');
end if;
```

**What is left to be implemented that you could not do?**

One of the things left to do is to finish the last part of my second test code. I need to figure out how to declare the clocks at a certain Mhz. I know that to get a 60hz refresh rate I need a pixel frequency of 25MHz. When the clock frequency is set 25MHz you put it in a port map so that you can map out to the pins. I also need to declare a PLL component along with my reset. The last would be to just test it. On the first set of code, I must debug it. It compiles and programs over to the FPGA board, but nothing happens.

**What are the difficulties and challenges you faced for this project?**

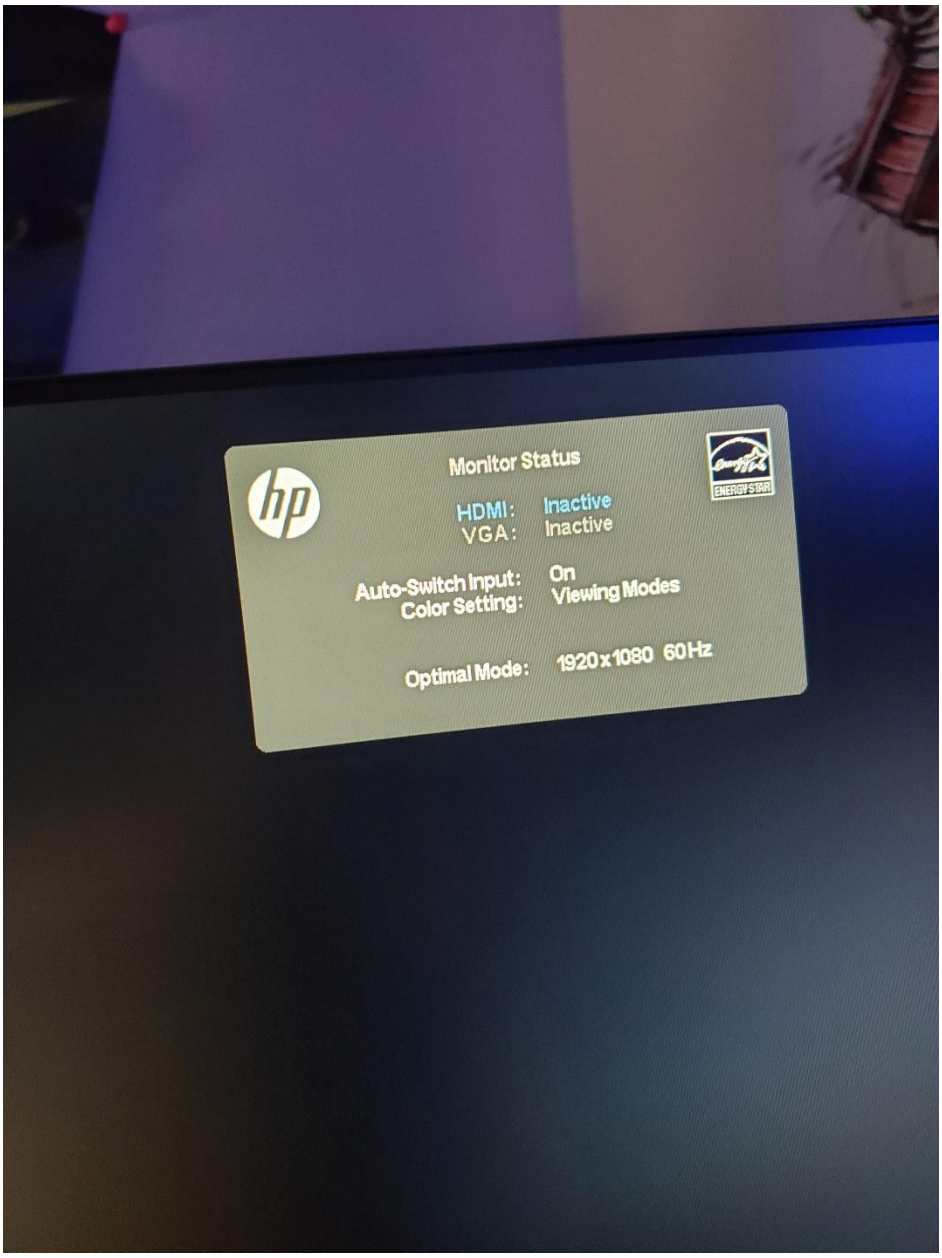
A couple of difficulties I faced while working on this project was looking for examples of VHDL code and translating it to something that I could use. The reason being I have never formally coded in VHDL before this class. We used VHDL in digital logic but I ever had to change one or two lines in each project. Granted this was also about 3 years ago, so I am very rusty. In the first set of codes, I need to figure out why nothing is outputting to the screen. I get the code to compile with no errors and it also shows no issues when it is running the simulation. But I noticed that the clock doesn't change. So that might have something to do with it.




As shown on top the clock signal never changes on the first set of code. I thought if I could declare the clock signal as it would automatically change by itself, but it looks like it needs to manually change by using a test bench. So, I have put a pause on this part of the code for now. On the second part of the code, I just need to figure out how to declare a PLL component. Using a PLL component will help with assign and generating a 25MHz clock on the FPGA board. Once that is implemented you can technically just use the port map then assign all the pins correctly and it should work just fine. Another thing is assigning all the pins correctly. I do not know if using the clock signal will be like in the labs where I just replace what is written in constraints with what I need. The VGA port of the constraints also has things that need to be figured out. I know for the most part where to apply the horizontal sync and the vertical sync. The RGB signals seem straight forward also, so it just needs to implement.

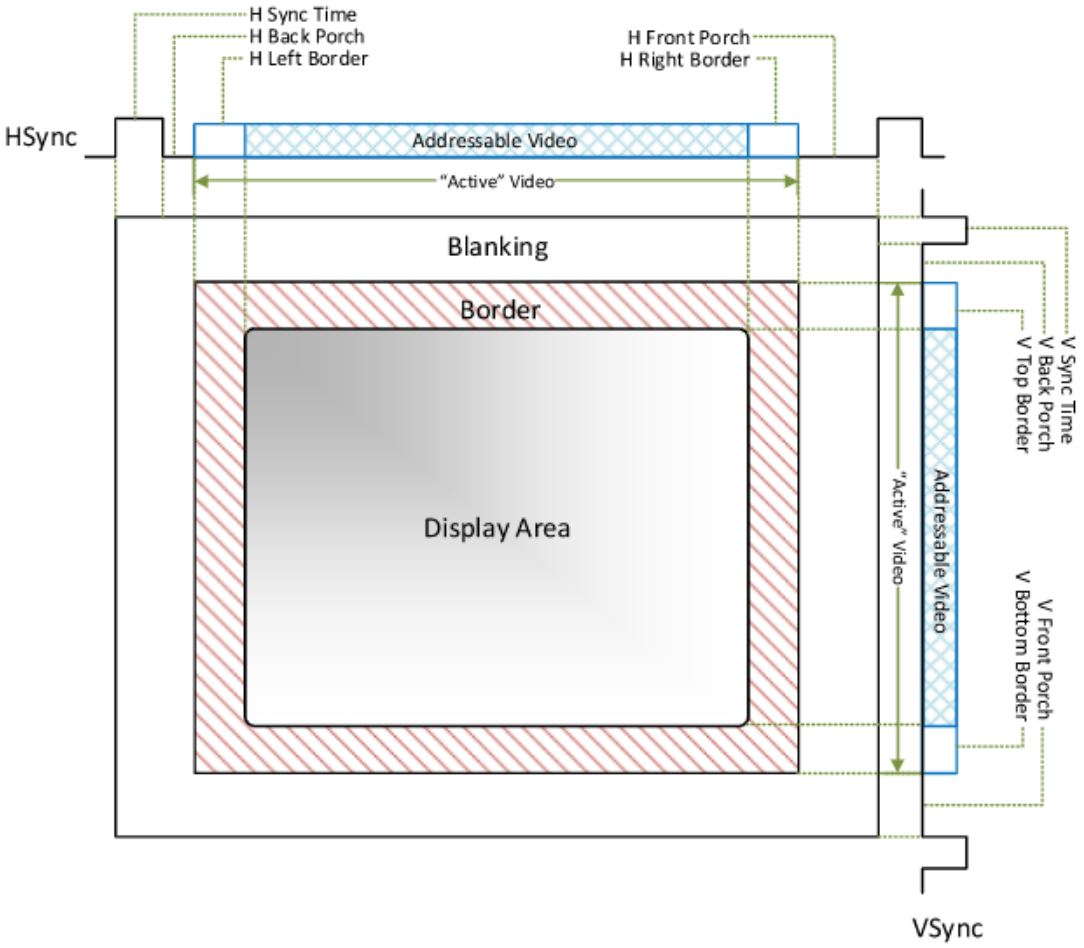
**Design and simulation figures and tables:**


This is an image from the VGA port to the VGA cable. As you can see the done light has also been turned on indicating that the first set of code has been programmed on the device correctly.




Here you can see that even though the program has been sent to the device with no issues, nothing populates on the display. The display just shows that the VGA is inactive.

Just showing that the VGA cable was plugged incorrectly.




| Description | Notation | Time | Width/Freq |
|---|---|---|---|
| Pixel Clock | $t_{clk}$ | 39.7 ns (± 0.5%) | 25.175MHz |
| Hor Sync Time | $t_{hs}$ | 3.813 µs | 96 Pixels |
| Hor Back Porch | $t_{hbp}$ | 1.907 µs | 48 Pixels |
| Hor Front Porch | $t_{hfp}$ | 0.636 µs | 16 Pixels |
| Hor Addr Video Time | $t_{haddr}$ | 25.422 µs | 640 Pixels |
| Hor L/R Border | $t_{hbd}$ | 0 µs | 0 Pixels |
| V Sync Time | $t_{vs}$ | 0.064 ms | 2 Lines |
| V Back Porch | $t_{vbp}$ | 1.048 ms | 33 Lines |
| V Front Porch | $t_{vfp}$ | 0.318 ms | 10 Lines |
| V Addr Video Time | $t_{vaddr}$ | 15.253 ms | 480 Lines |
| V T/B Border | $t_{vbd}$ | 0 ms | 0 Lines |

Both images show how the front, back porch look like and where the display area lands in between both of them. It also shows how it affects the horizontal and vertical syncs along with the table.

**Resources:**

- http://www.ece.ualberta.ca/~elliott/ee552/studentAppNotes/1998_w/Altera_UP1_Board_Map/vga.html
- https://www-inst.eecs.berkeley.edu/~cs150/sp99/sp99/project/compvideo.htm
- https://reference.digilentinc.com/learn/programmable-logic/tutorials/nexys-4-ddr-vga-test-pattern-with-mouse-overlay/start
- https://www.youtube.com/watch?v=WK5FT5RD1sU&t=896s&ab_channel=Toni
- https://learn.digilentinc.com/Documents/269
- https://www.digikey.com/eewiki/pages/viewpage.action?pageId=15925278#:~:text=It%20requires%20the%20user%20to,image%20at%20the%20proper%20time
- https://www.instructables.com/Design-of-a-Simple-VGA-Controller-in-VHDL/#:~:text=VGA%20Controller%20is%20the%20digital,sync%20signals%20for%20display%20purpose
- http://tinyvga.com/vga-timing/640x480@60Hz