Lab 21. Displaying on a VGA screen

In this lab, we will learn how to control the output of a VGA screen, with a resolution of 1280x1024 and a refresh rate of 60 MHz.

Go to `~/labs/lab21` and open the `lab21.xpr` project with Vivado.

```
vivado lab21.xpr &
```

The top module of the project is `src/top_vga.vhd`, as you can see the module imports the package `vga_pkg`, available in the `src/vga_pkg.vhd`. Have a look at the content of the package, you should see there all the required constants for our display.

Let's go back now to the `top_vga` file. In the port declaration, we see all the interface to control the VGA screen and a `RST` signal, which is connected to the central push button of the Basys-3 board.

Exercise 1. Generating the pixel clock

Our screen should work with a refresh rate of 60 Hz. It means that each pixel should be updated once every 0.1666 s. Considering the total amount of pixels in the screen, calculate the clock frequency required to drive the screen.

Now we need our FPGA to generate a clock of the frequency you just calculated. Create the clocking wizard IP, as in Lab 18, and set the frequency of the output clock accordingly. We don't need any control signal, so untick the `locked` box.

Generate the IP, and instantiate in the `top_vga` file as usual. The output clock should be mapped to the `clk_pix` signal already defined in the module.

Exercise 2. Control the screen

To control the screen we need two counters that will monitor the location of the current driven pixel. Create the new counter signals `x` and `y` in the declarative part of the module.

Tip: The width of these signals is already provided in the `display_pkg.vhd` file.

Now create a clocked process, sensitive to `clk_pix`, which should increase `x` and `y`, in the following way.

`x` is increased at each clock cycle. When `x` is equal to the total screen width in pixels is reset and `y` is increased by 1. When `y` is equal to the total screen height in pixel, is reset.

In the same process, set the output `VGA_HS` high, whenever the `x` coordinate is in the horizontal sync region of the screen, otherwise keep it low. Similarly, set `VGA_VS` high when `y` is in the vertical sync region, else is low.

Finally, still in the process, we need to drive the `VGA_RED`, `VGA_BLUE` and `VGA_GREEN` signals in order to print images on the screen. This should be done, when `x` and `y` are in the active region of the screen. Outside of this region, all three signals should be set to 0 (BLACK).

As first test, let's just try set the color values, to print a blue screen.

The 12b hexadecimal color mapping can be found at this address.

Finally, add a synchronous reset mechanism, that set the counters `x` and `y` to 0, as well as all the VGA control signals.

Once you are done, run the implementation and generate the bitstream.

Load it to the board, and connect it to one of the provided VGA screen, and check the expected outputs and reset mechanism.

Exercise 3. Print the German Flag

Modify the code above to print the German Flag.

(Optional) Exercise 4. Movements

Screens create the illusion of movement, by changing the frame frequently.

Try to implement in the design a white horizontal bar of 400 pixels that shift to the right, at every refresh of the screen, on a blue background.