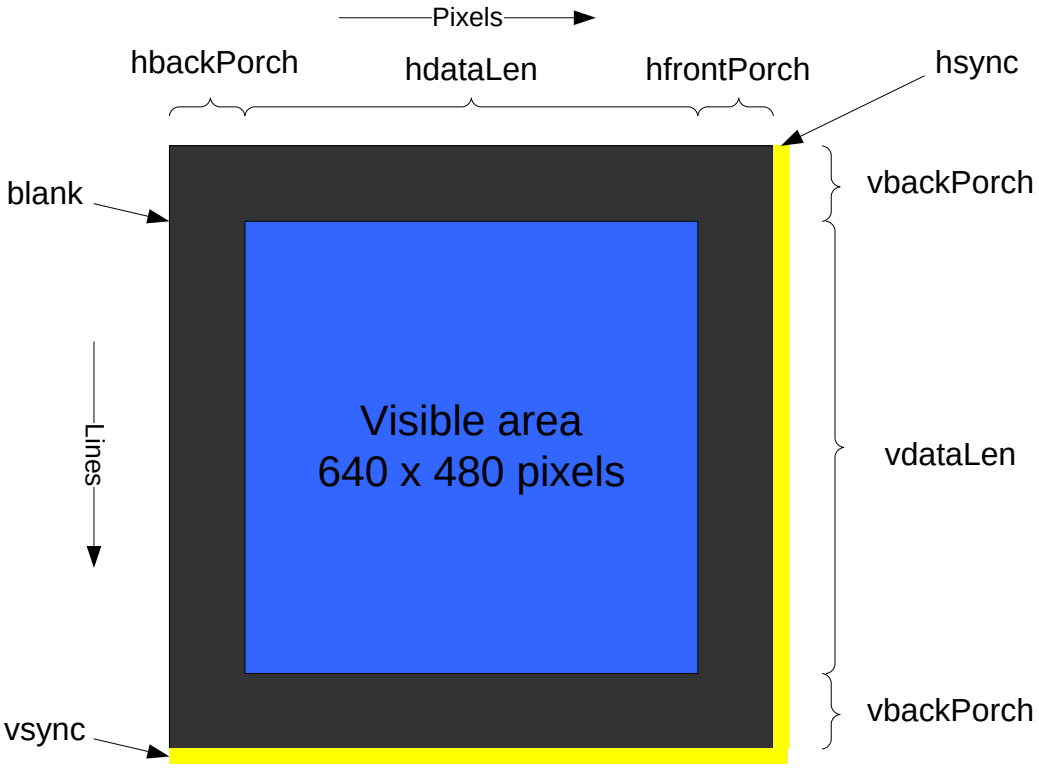
DSD Exercise



Procedures and Functions in VHDL

(The VGA Exercise)

18-11-2015/PHM

GOALS

VHDL allows you to create functions and procedures in addition to design entities (= components), which allows you to simplify reuse and provide encapsulation. In this exercise we'll explore this subject and apply its usage to a simple VGA controller.

The goals for this exercise are:

- To get experience with VHDL functions, procedures and packages
- To get some basic knowledge about VGA

PREREQUISITES

- Have Quartus II up and running
- That you have read THE DSD EXERCISE GUIDELINES!!!

THE EXERCISE ITSELF

To begin with, you must implement some basic functions and procedures and apply them to a VGA design afterwards.

1) THE LOGIC-PACKAGE

To get a bit of hands-on with functions and procedures, you must implement a small library with a few logic functions.

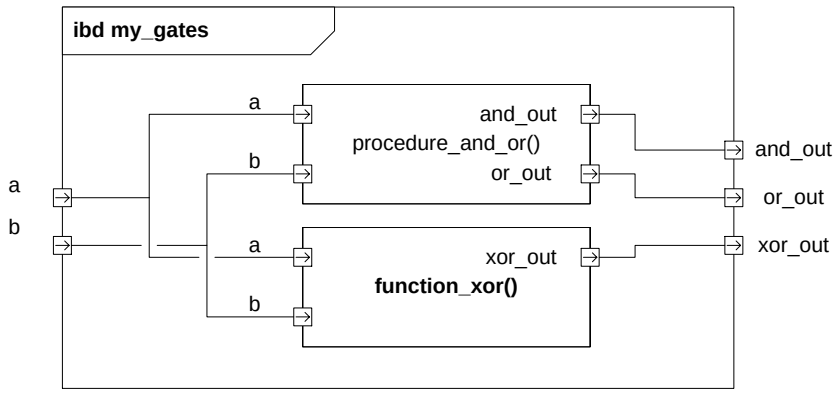


FIGURE 1: MY_GATES IBD

1) Implement the design entity: "my_gates". The design must consist of a function: "my_xor" and a procedure: "my_and_or". Use the interface described in figure 1. Compile your design in Quartus II and view the output in the Netlist Viewers->RTL Viewer. *How does it compare with your design?*

2) Modify your design entity into a package with an appropriate header. The procedure + function implementations must be in the body section.

3) Create a tester and import your new library: work.my_gates.all Map inputs to switches and outputs to LEDs, and test it on the DE2-board.

2) VGA!

The purpose of this exercise is to implement the clocking circuit for a VGA controller. The VGA interface was originally designed for Cathode Ray Tube (CRT) monitors. This technology uses a cathode beam to highlight a single pixel on the screen at a time, see figure 2. The beam travels in horizontally, until it meets the end of the line, then it is turned off and moved to the beginning of the next line. At the end of the last line, the beam is again switched off and moved to the first pixel in the first line again. The darkened border regions gave time for the beam to travel to the beginning of the next line or frame. In graphics controllers, the upper and lower dark areas (vertical back- and fron porch) are typically used to load the next frame from memory. The dark areas in the sides are known as horizontal back- and front porch.

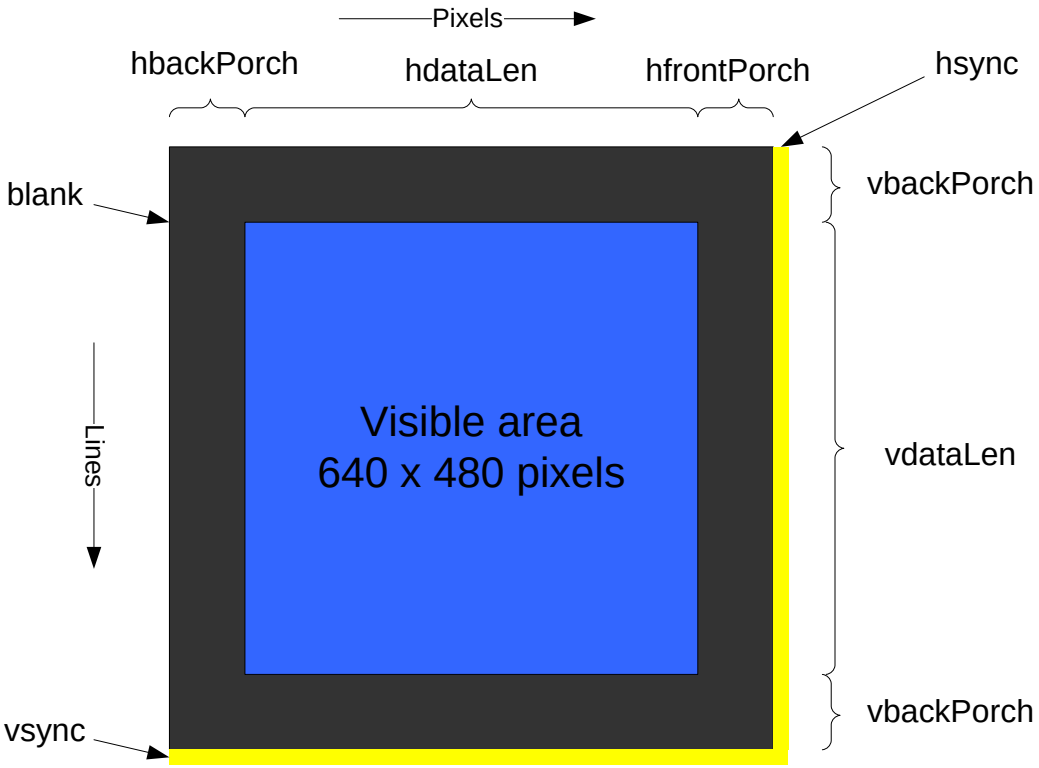


FIGURE 2 VGA PICTURE LAYOUT

A graphics controller uses a clock signal as a time reference, so the back- and front porch times are always converted into a number of clock cycles. Figure 3 shows a timing diagram of the interface. The diagram is valid both vertically and horizontally. When we are in the dark regions, the “blankOut” signal is active to blank- or shut off the pixels. At the end of each line, the “syncOut” signal goes low to indicate that a new horizontal line (or vertical frame) on its rising edge. Valid pixel data (RGB) can be output in the data interval, also shown in the figure.

The RGB values on the VGA interface are actually represented by three analogue voltages: Red, Blue and Green. The DE-2 board has a built-in VGA-Digital to Analog Converter. It uses a 10-bit input per channel to generate the RGB voltages. You can manipulate the colors by manipulating the three 10-bit color values.

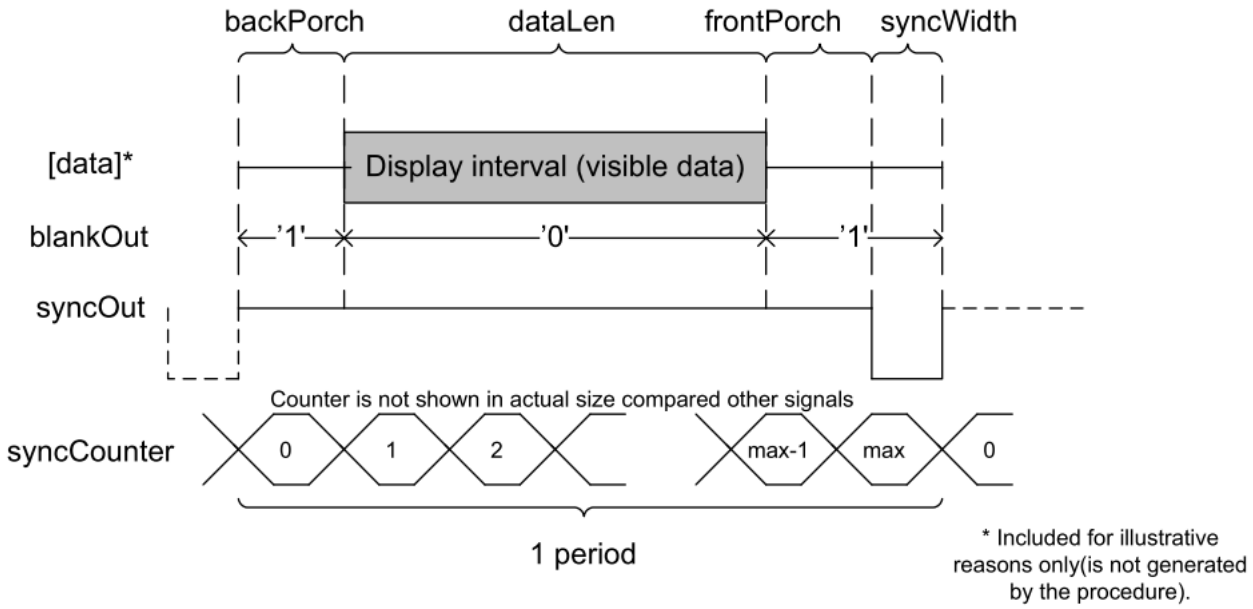


FIGURE 3 VGA TIMING

1) Download and extract the VGA.zip project located with this exercise in Black Board. Open it in Quartus and investigate its design. Write a brief description in your documentation. You can read about VGA in the DE2 user manual page 38-39.
2) Implement the missing sync generator. Use this interface:

```
procedure  syncGenerator( [class] syncCounter : [mode] [type];
                          [class] syncOut     : [mode] [type];
                          [class] blankOut    : [mode] [type];
                          [class] frontPorch  : [mode] [type];
                          [class] backPorch   : [mode] [type];
                          [class] dataLen     : [mode] [type];
                          [class] syncWidth   : [mode] [type]) is
```

   Find the class(constant, signal), mode (in, out or inout) and type(integer, natural,std_logic) for each parameter in the procedure above. This information can be retrieved by investigating how the function is used in the template. Create the empty procedure with the right modes, classes, and types in the declarative part of the architecture.
3) Implement the functionality as described in Figure 3. The procedure must update the SyncCounter each time it is called, and the BlankOut and SyncOut signals must be updated based on the current SynCounter count.
4) Create a functional simulation of the procedure.
5) Download and test on the DE2-board
6) (Optional) Can you implement the Danish flag? Moving objects??! Bar diagram??!