VGA Controller using VHDL

By: Rotar David Rafael

Table of Contents

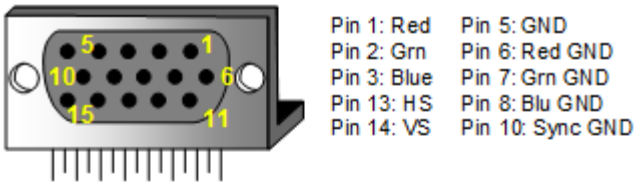
1. Design

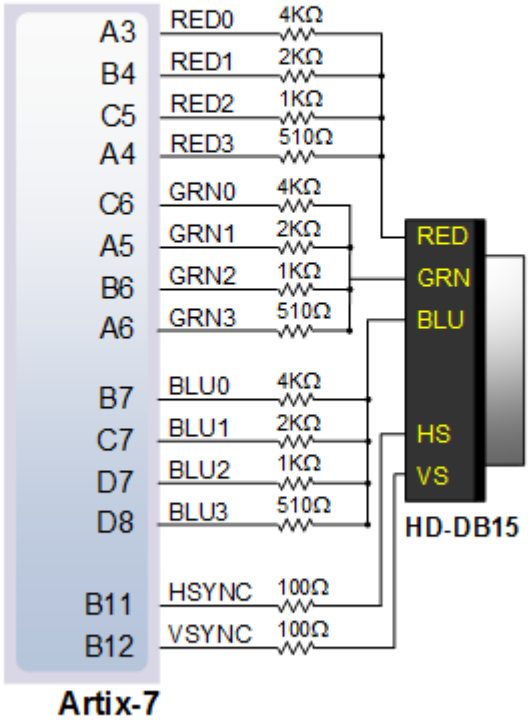
1.1 VGA Port

The VGA Controller works by the principle of sending through the cable a number of signals on different pins, those being:

- HSYNC: A sync signal active for the duration of blanking of horizontal rows
- VSYNC: Another sync signal active after displaying a full frame
- RGB signal: 2 pins for each color (signal and ground), being able to display 4 bit colors

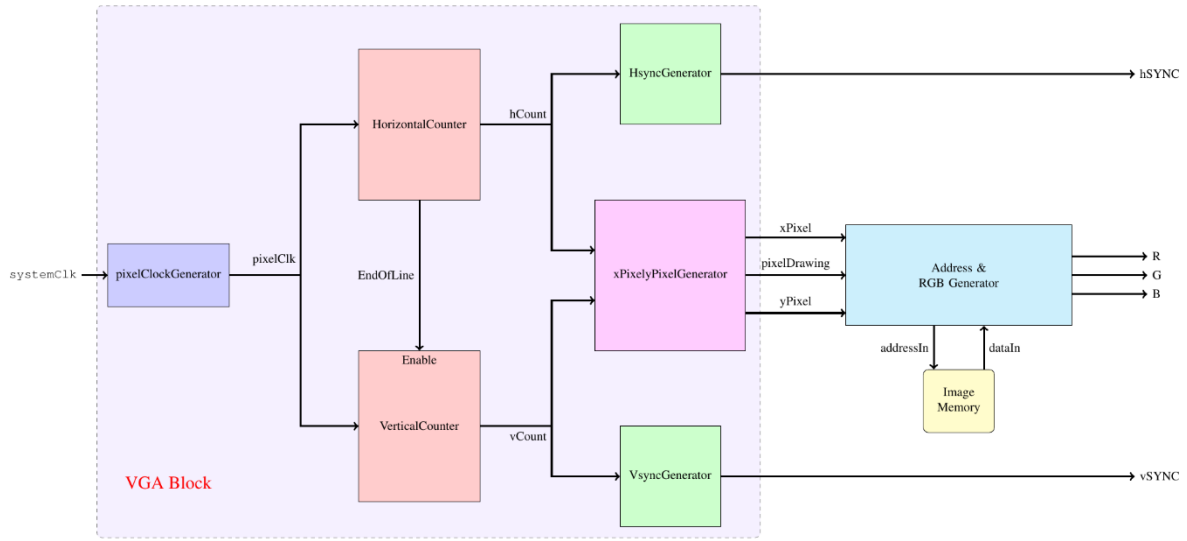
HSYNC and VSYNC signals are active on 0.



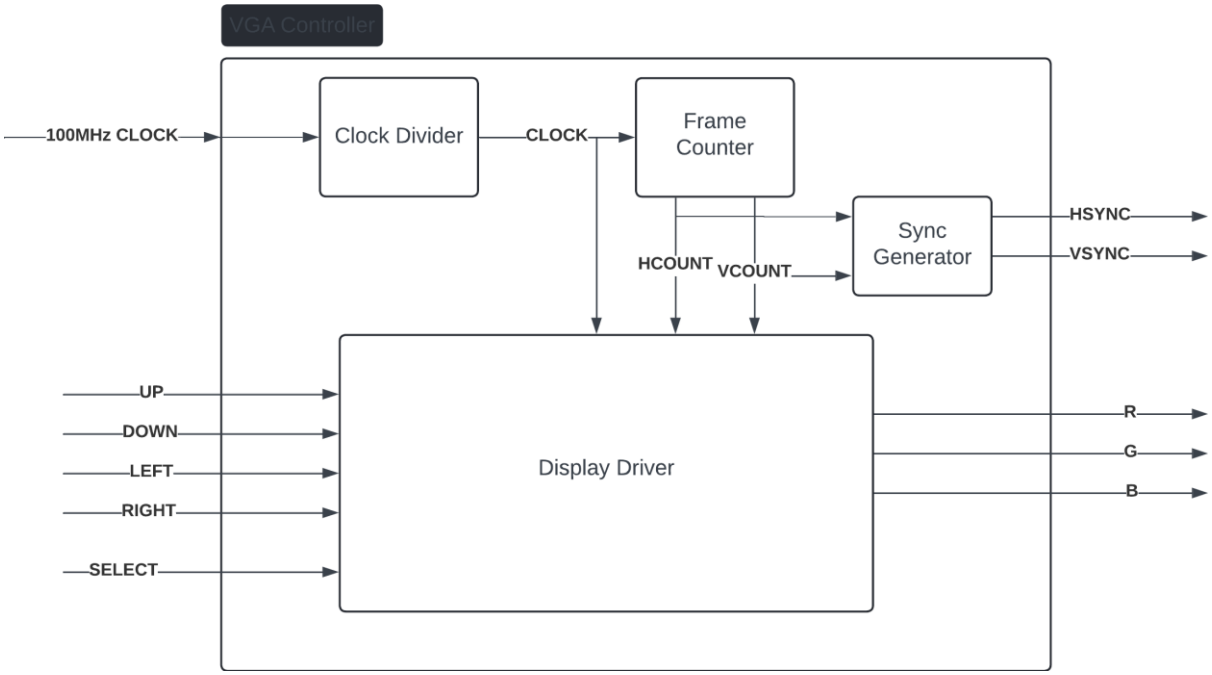


1.2 VGA Controller

The functioning of the controller works in principle by this diagram:



The controller takes in a Clock signal by the requirement of the specific resolution for which it is configured, and in turn generates the HSYNC and VSYNC signals that it sends sends to the monitor along with a Display Enable (D_Enable) signal and the X (VCOUNT) and Y (HCOUNT) coordinates of the current 'moment', which are sent to the Display Driver that handles the information that will be displayed on the screen.



1.3 Display Driver

As stated above, the Display Driver takes in as signals the coordinates of the current pixel, the D_Enable signal, alongside movement signals, a 2-bit SELECT (SEL[0:1]) signal and two clock signals, one for the memory and synchronization and one for movement.

The only output of the Display Driver is the values of the R, G and B signals of the current pixel of the controller.

If the Controller is outside of the “Active” phase, that being the Front Porch, the Back Porch or the Sync region, then D_Enable is 0 and the output of the RGB signals is 0. Otherwise, if the count of the controller is in the active region, then the controller will generate color values according to the mode selected.

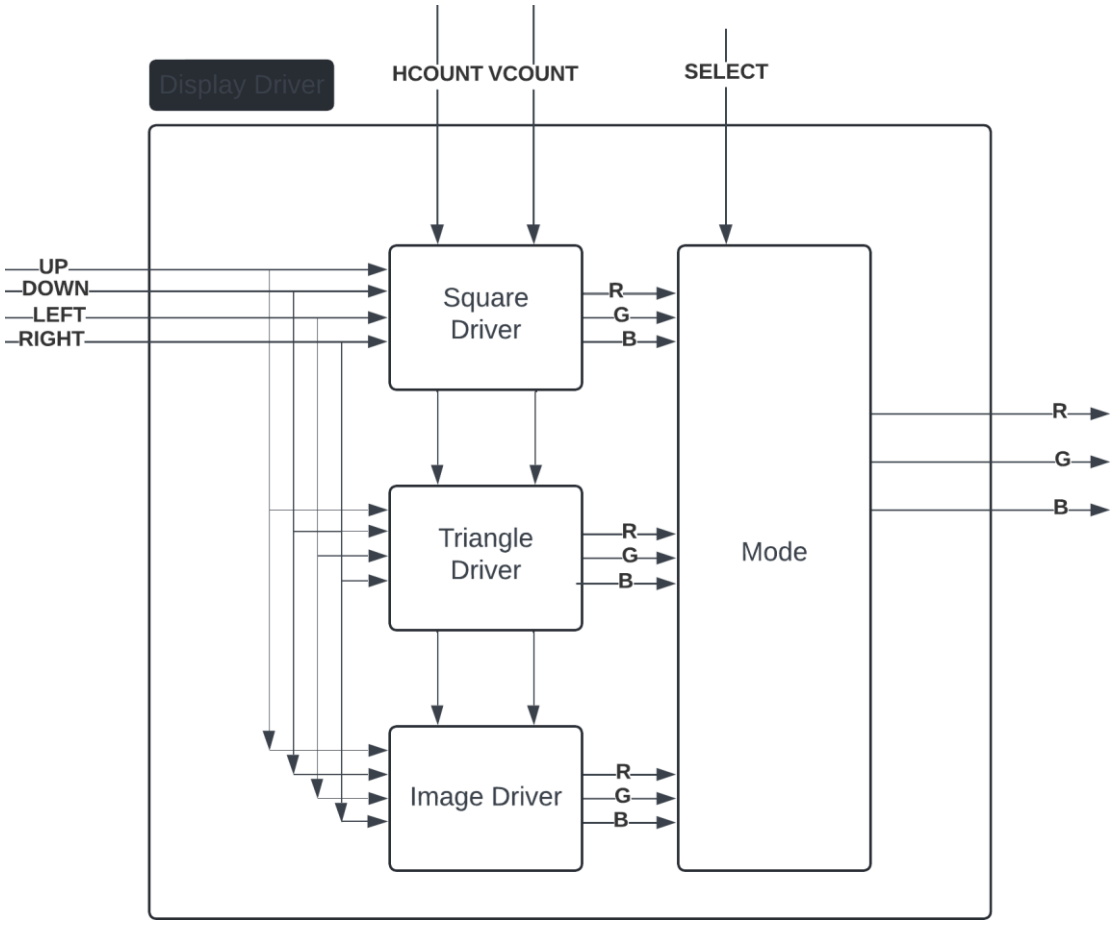
Driver modes:

SEL=00 – Square

SEL=01 – Image

SEL=10 – Triangle

SEL=11 – Screensaver Square



1.3.1 Square Driver

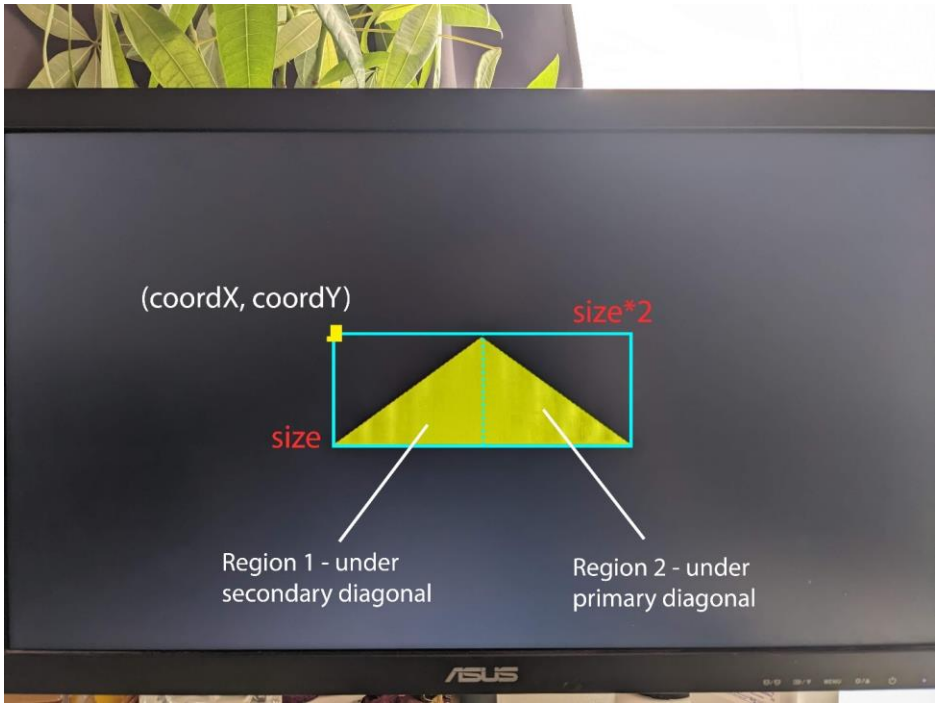
The Square Driver generates a cyan square. It remembers the center of the square, along with its size, and if the current pixel coordinates are in the region covered by the square, then the pixels will be assigned the value of the color cyan

Note: all the shapes are initially positioned in the middle of the screen, and, while moving, they cannot escape the borders of the screen because of restrictions implemented in the movement procedure.

1.3.2 Triangle Driver

The triangle driver works in a similar manner both to the square and the image driver. It remembers the coordinates of the upper left corner and the size of the triangle. The size represents the height and half the width of the triangle.

The driver splits the triangle into two symmetrical sides. The first region colors the pixels under the secondary diagonal of the square region and the second region colors the pixels under the main diagonal of that region.



1.3.3 Image Driver

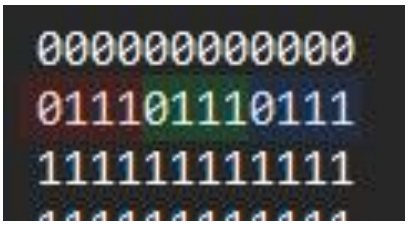
The Image Driver determines the rectangle in which it displays the image and if the region is valid, the Driver will get the specific address from the ROM memory in which the image is stored. Using the position of the upper left corner of the image and the current pixel coordinates, it calculates the relative position on the screen of the pixel and offsets the image, so the relative coordinates match the ones in the memory.

1.4 ROM Memory Image

As stated above, the image in mode “01” is saved in a ROM memory, where each pixel is represented as a row of 12 bits, each RGB value being stored as a 4-bit value. The whole image is stored in a linear manner, each 12-bit string being an element of an array of length imageHeight*imageWidth. The values in the memory are being accessed with the formula (pixelX*columns)+pixelY, where pixelX/Y is the position of the current pixel and “columns” is the number of columns of the image. This formula is used because of the linear nature of the data structure holding the information.

The Image is read from a file at compile time and then it is stored in the memory on the board.

The conversion from a picture file to a text file containing the information of every pixel in 12 bits, stored linearly from left to right, up to down was done using a Python script that went through every pixel in that order, read the values of the pixel, then mapped the 8 bit color value to a 4 bit one so that each color is displayed accurately, then wrote the pixel color values as a row in the text file.



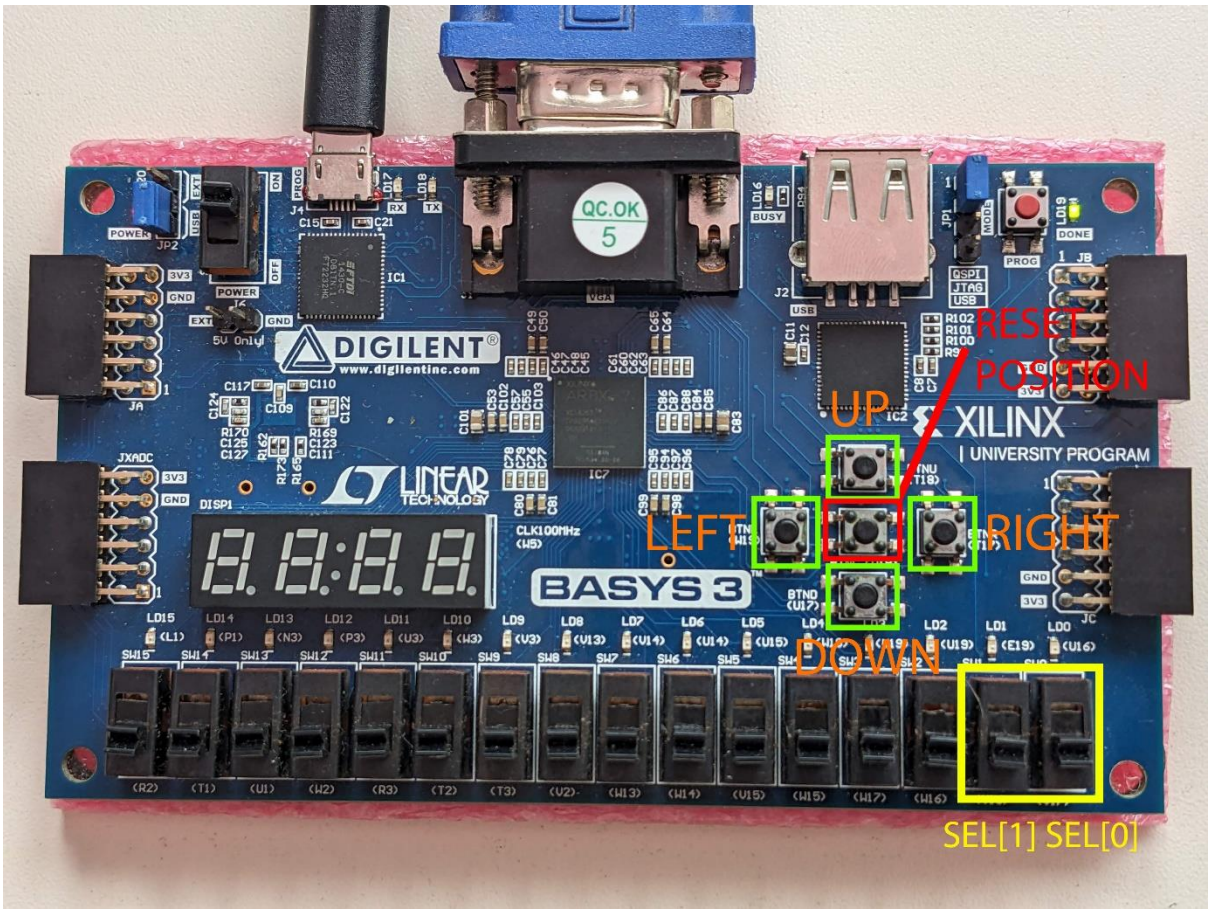
2. Interface

The method of interacting with the Controller and the Display Driver is with the buttons on the board and two switches for the mode.

The SEL[0] and SEL[1] switches select the mode of the Display Driver, as stated above in section 1.3. the down-position is considered as ‘0’ and the up is considered ‘1’.

The position of the image is controlled using the 5 buttons on the board. The up, down, left and right buttons move the shape/image in their respective direction, while the middle button resets the position of the shapes back to the middle of the screen.

To be noted that moving one specific shape does not affect the position of the other shapes, while the reset button re-centers all the displays.


RESET
POSITION
UP
LEFT
RIGHT
DOWN
SEL[1] SEL[0]
DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
BASYS 3
LINEAR TECHNOLOGY
QC.OK
5