Παναγιώτης Κρεμμύδας 1435

10/27/2016

# Lab exercize 3:

# VGA controller

###### CE430: Digital Circuits Lab

###### Electrical& Computer Engineering

###### University of Thessaly

# Contents

0: Introduction

1: VRAM implementation: Initialization of display data into SPARTANS 3 onboard BRAM.

2: HSYNC signal & timing: 1st Pulse width modulator adhering to the 640x480 60hz monitor standards, control of the data flow to each horizontal pixel rotationally in the given framerate

3: VSYNC signal & timing: 2nd Pulse width modulator adhering to the 640x480 60hz monitor standards, control of the data flow to each vertical pixel rotationally in the given framerate

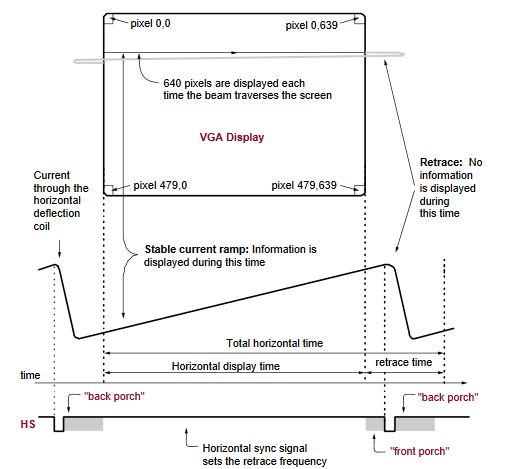
4: Conclusion: Challenges & Solutions in the designing, testing and implementing of the above

# Introduction:

Implementation of a Video Graphics Array Controller/Driver.

The goal is to successfully drive a typical monitor and display an image in it.

For the purpose of a continues representation through the VGA port, part of the internal RAM of the FPGA unit used, will be assigned as Video RAM (VRAM) of the driver. The suggested sample image for the testing of the aforementioned driver is the typical red, blue, green,black horizontal stripes separated repeatedly by white stripes. The black stripes part is also repeatedly vertically overlapped by a red,green,blue vertical stripe.



As shown in the above figure of the Xilinx Spartan3 manual , the VGA controller generates the horizontal sync (HS) and vertical sync (VS) pulse width modulator signals and coordinates the delivery of video data on each pixel clock.

Video data typically comes from a video refresh memory with one or more bytes assigned

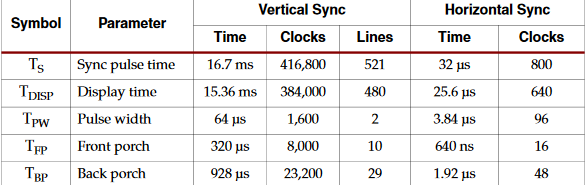
to each pixel location. The Spartan-3 Starter Kit board uses three bits per pixel, producing

one of the eight possible combinatorial colors. The controller indexes into the video

data buffer as the beams move across the display. The controller then retrieves and applies

video data to the display at precisely the time the electron beam is moving across a given

pixel



The signal timings are derived for a 640-pixel by 480-row display using a 25 MHz pixel clock and 60 Hz ± 1 refresh. The table above shows the relation between each of the timing symbols. The timing for the sync pulse width (TPW) and front and back porch intervals (TFPand TBP) are based on observations from various VGA displays. The front and back porch intervals are the pre- and post-sync pulse times. Information cannot be displayed during these times.

# Part 1 - VRAM implementation:

## Implementation

BRAM instances: Utilizing the bulk memory necessary for storing the image. 12288 out of the 16383 bits provided by a the 16Kx1 preconfigured BRAM block are used to store the pixel value for each one of the 3 colours (Red , Green, Blue). 3 BRAM modules contain the preinitialized memory representing the test image. The instances of those modules take as input the current address that the active pixel is corresponding to and feed the data directly to the Red, Green, Blue Fpga pins. The blocks utilizing the VRAM are permanently activated and their data is static

## <https://github.com/kmd178/Digital_Systems_lab3_VGA/commit/7b672c559dc86ec4af272ddadbd3d59523fd320c>

## Verification

## 

As indicated the corresponding output is assigned through the initialized memory. Output waveforms are performing as expected.

Observation: Enable signal should be true for the BRAM modules to work.

## Experiment/Resulting implementation

FPGA board testing was not necessary for this part of the assignment

# Part 2 - HSYNC signal & timing:

## Implementation

Pixel\_signal: Defines a clock that corresponds to the time available to display one pixel of information.

Clock cycles necessary for a 50Mhz clock to move to the next pixel:

1/60(frames)= 521HYNC SIGNALS = 521\*800 pixel signals -> pixel period= 1/(521\*800\*60) = 1.99936020473clocks of a 50mhz clock

We are assumming a resync is taking place inside the monitor itself everytime the monitor needs to do a Horizontal retrace (x800 pixel signals). The resulting error 0.511836216=25.6% after x800 pixel cycles is not high enought to skew the monitors sampling from the middle of the incoming pixel signal's period to the wrong pixel.

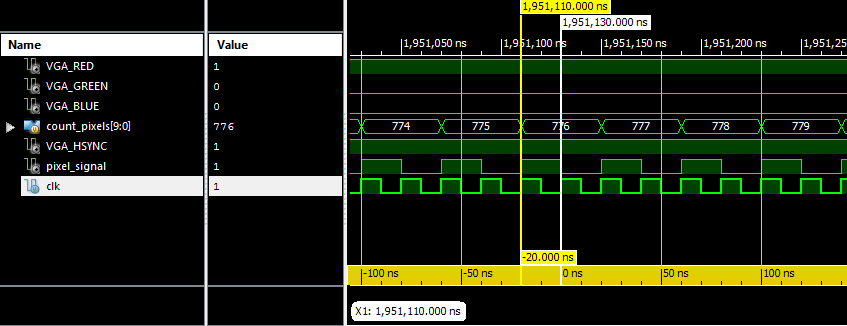
If necessary special clock instanciation modules can adjust the clock’s mhz accordingly to fit the desired period using onboard buffer routes. The result is an implementation with minimum sampling error.

Count\_pixels: Generally, a counter clocked by the pixel clock controls the horizontal timing. Decoded counter values generate the VGA\_HSYNC signal. This counter tracks the current pixel display location on a given row.

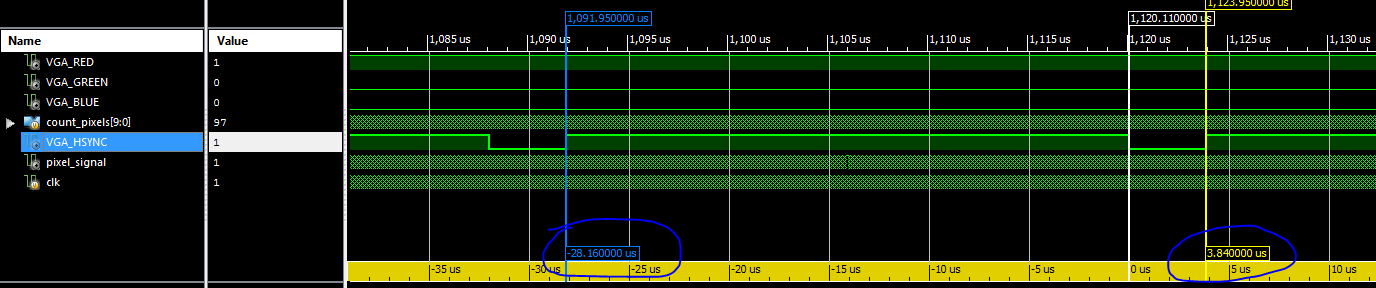
VGA\_HSYNC : Decoded count\_pixel values generate the VGA\_HSYNC signal which corresponds to a monitor’s line retracing time. This signals timing is predefined within manufacturers specifications and various timings correspond to different refresh rates and resolutions

https://github.com/kmd178/Digital\_Systems\_lab3\_VGA/commit/7b672c559dc86ec4af272ddadbd3d59523fd320c

## Verification



Pixel signal implementation and counter iteration.



Pulse width modulation of the HSYNC vga signal according to the specifications of the 640x480, 60hz monitor standard

Output waveforms are performing as expected.

## Experiment/Resulting implementation

FPGA board testing was not necessary for this part of the assignment

# Part 3 - VSYNC signal & timing:

## Implementation

Count\_lines: count\_pixels controls the horizontal timing. Decoded counter\_pixels values generate the VGA\_HSYNC signal. Count\_lines is a separate counter that tracks vertical timing. The count\_lines counter increments with each HS pulse and decoded values generate the VS signal. This counter tracks the current display row.

VGA\_VSYNC : Decoded count\_lines values generate the VGA\_VSYNC signal which corresponds to a monitor’s frame retracing time. This signals timing is predefined within manufacturers specifications and various timings correspond to different refresh rates and resolutions

Virtual\_out: Count\_lines and count\_pixels ,together with the logic that defines the states where pixels are rotated and displayed, are continuously running counters that form the address that is used by the video display buffer (the BRAM modules initialized in part 1).

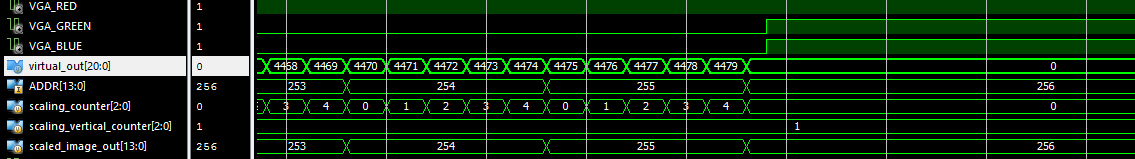
scaled\_image\_out: Becase the internal memory of the fPGA is insufficient to support the full 640x480 resolution, it is necessary to slow down the address rotation by repeating the same addresses through the BRAM

scaling\_horizontal\_counter->4 :Controls horizontal scalling. The same horizontal pixel inside the VRAM is displayed 4 consequetive times

scaling\_vertical\_counter->4: Controls vertical scalling. The same vertical pixel line inside the VRAM is displayed 4 consequetive times

<https://github.com/kmd178/Digital_Systems_lab3_VGA/commit/7b672c559dc86ec4af272ddadbd3d59523fd320c>

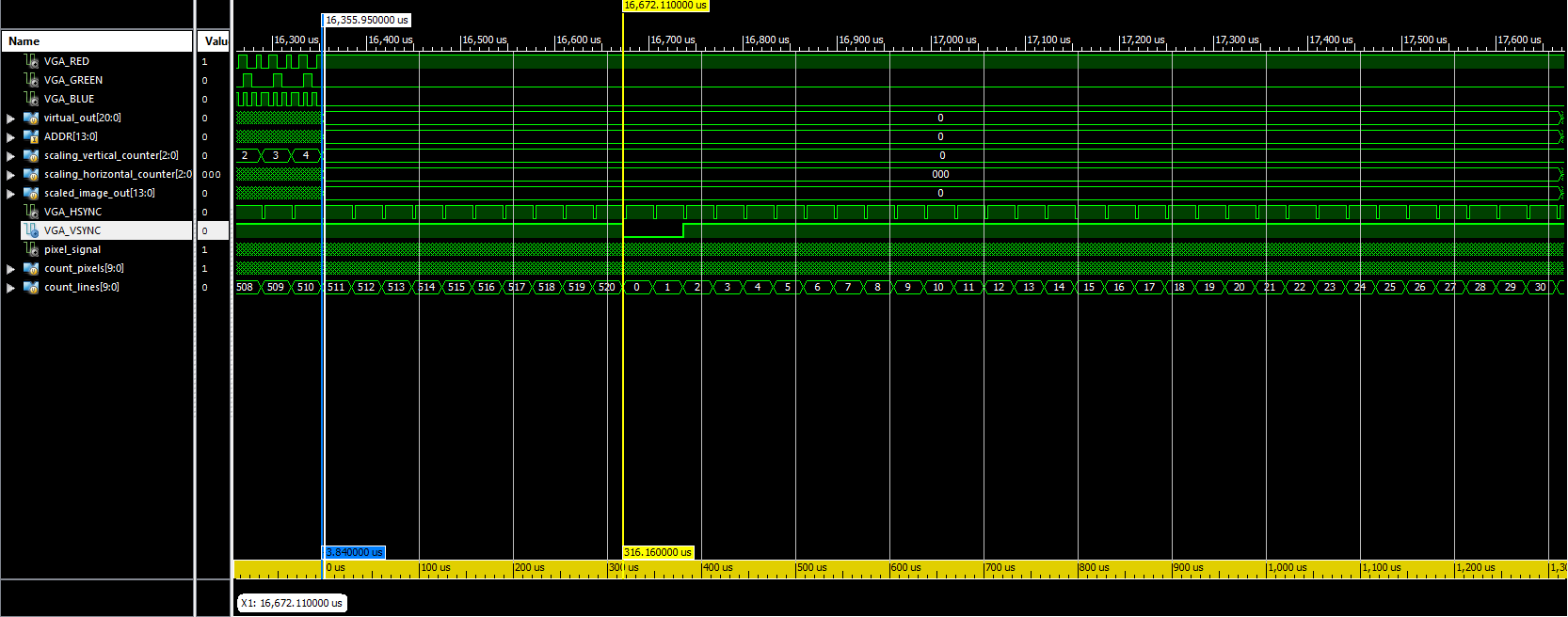
## Verification



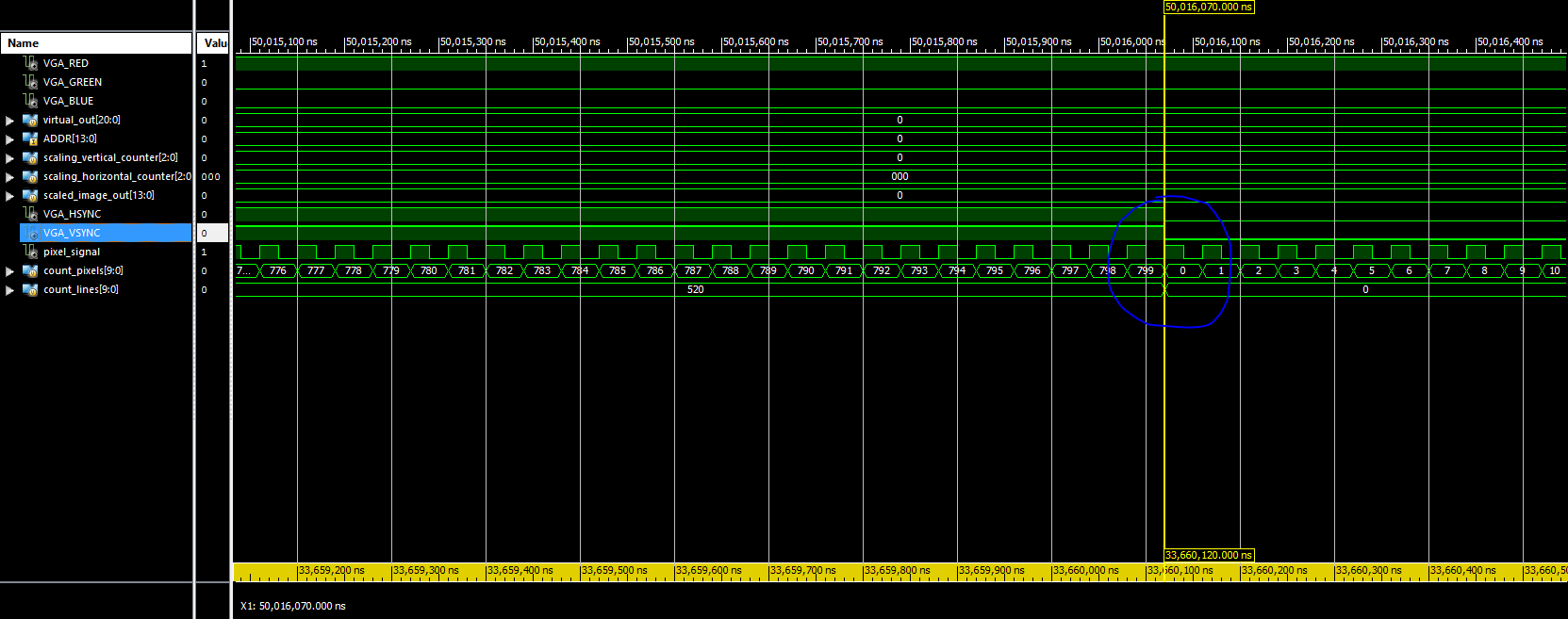
Scaling function example: Pixel adresses are iterated every 4 virtual pixels. Line adresses are iterated every 4 virtual lines



Pulse width modulation of the VSYNC vga signal according to the specifications of the 640x480, 60hz monitor standard



VSYNC every 520lines.



Count\_pixels, count\_lines loops

Output waveforms are performing as expected.

## Experiment/Resulting implementation

First trial: VSYNC signal incorrect pulse width modulation. No monitor output is displayed

Second trial: Memory initialization is in binary form not hexademical as implemented. Resulting image contains stripes of vertical deactivated pixels due to the value 1 initialized in memory instead of F.

Third trial: Test image displayed through the VGA port to the monitor as expected. The FPGA board performs as expected.

# Conclusions:

Insufficient hardware capabilities like a limited VRAM may require upscaling of the contained image to a higher resolution. The upscaling is implemented through the multiplication of the time needed to increment rotationally each given stored data pixel or stored data pixel line (upscaling horizontal and vertical factors)

The hardware of the monitor itself supports sample recalibration on every horizontal retrace making it easier for different system clocks to implement a video controller that can support their various resolutions and refresh rates.