Hardware Engineering Lab

Team Bonsai

14th May 2022

Team members

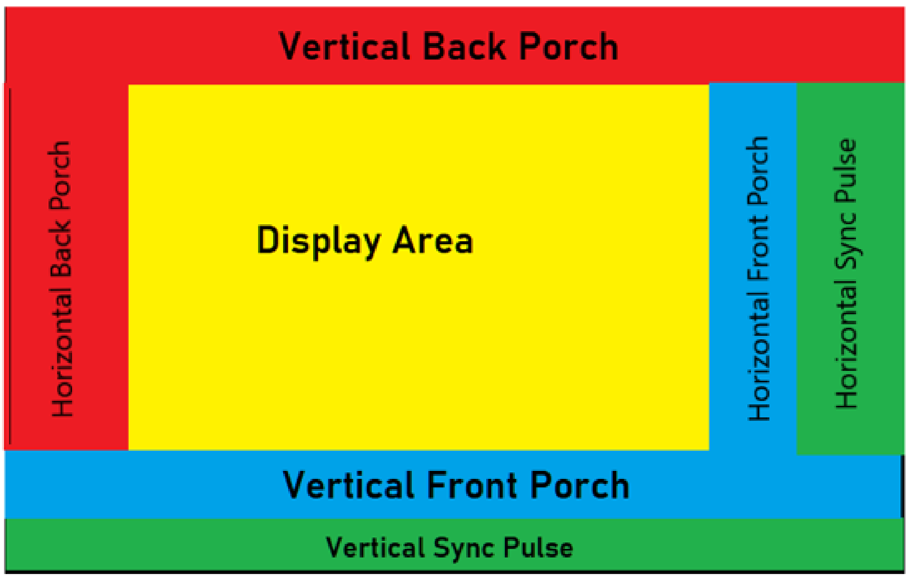
Nikolaos Karapoulatidis
Shehroz Bashir Malik
Christian Stratmann

Introduction

We will create a VGA Controller on an FPGA to drive a display with a resolution of 640*480.

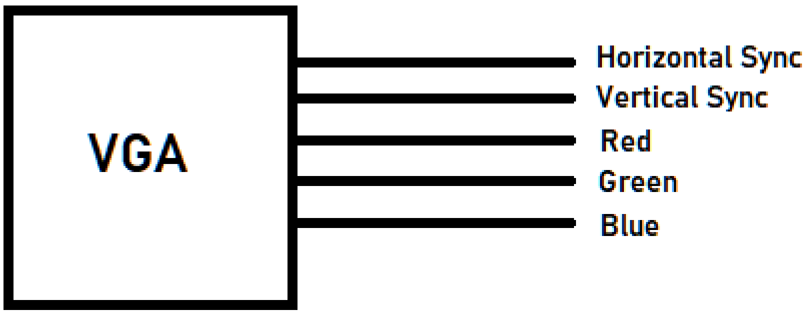
Concept description

Images in VGA are transmitted line by line with each individual pixel having a defined duration. When a line ends, a horizontal sync signal triggers which in return causes the monitor to change lines and reset to the beginning of the next line. After it has completed reading through all the lines, a horizontal synchronization signal is triggered which causes the monitor to reset to the initial position. These signals are called HSYNC and VSYNC respectively. They are always on until the retrace and then they go low again.



For simplicity's sake, we will shift the Horizontal Back Porch to the right of the Display area so it can be next to the Horizontal Front Porch and Horizontal Sync Pulse. Similarly we will shift the Vertical Back Porch to the bottom of the Display area so it can be next to the Vertical Front Porch and Vertical Sync Pulse.

The inputs of the VGA display are as follows:



These signals range from 0 to 0.7V.
The pixel clock is 25 MHz. The specifications for the signal are as follows:

Horizontal Timing

Scanline part	Pixels
Visible area	640
Front porch	16
Sync pulse	96
Back porch	48
Whole line	800

Vertical Timing

Frame part	Lines
Visible area	480
Front porch	10
Sync pulse	2
Back porch	33
Whole frame	525

Project/Team management

We will be following the Agile Methodology. We will have weekly Agile Sprints. In the beginning of each sprint we will have sections dedicated to planning, design, coding and analysis. Initially, the team gathers together, physically or online, to have a brainstorming session. We discuss ideas to solve the problem, note down the requirements of the task ahead, adjust and incorporate our ideas to solve the aforementioned problem. An analysis is

carried out to determine if our work is in line with the requirements. This is an iterative process and it is quite normal for us to restart our work if we have no productive outcomes. The tasks are split evenly amongst 3 members.
Shehroz:

- VHDL Code
- Xilinx Implementation
- Documentation

Nikolaos:

- VHDL Code
- Test Bench
- Documentation

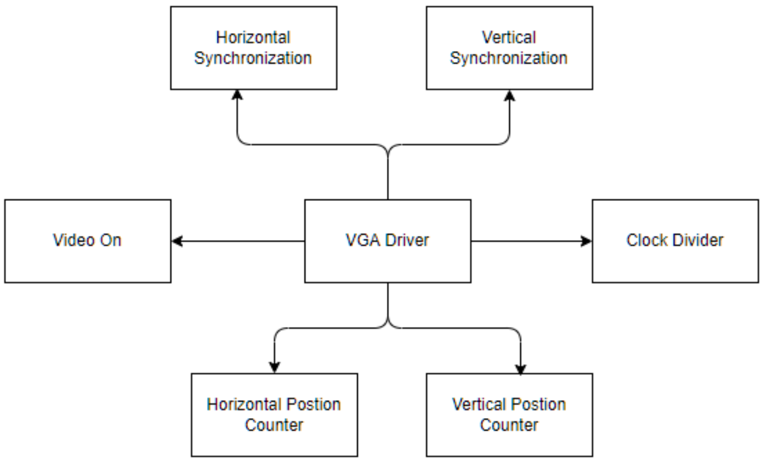
Christian:

- VHDL Code
- Documentation
- EAGLE PCB

Technologies

We will use ModelSim to program our VGA Controller in VHDL. It will be synthesized in the Xilinx ISE Software. This will then be implemented on a XIlinx Board. The Circuit design is carried out entirely on Eagle.

VHDL Implementation



Entity

In the entity section, the ports were defined as follows:
A clock and reset as input with type STD_logic. Horizontal Sync and Vertical Sync as output with type STD_logic. Red Green Blue as an output with STD_Logic_Vector. The values for this ranged from 0 to 2 as one bit is assigned to each color.

Architecture

In the architecture, 4 separate signals were initialized with values at 0. These are the Clock25, Horizontal Position, Vertical Position and VideoOn. A few constants were defined according to the signal specifications needed to run a resolution of 640*480 at 60Hz. Horizontal Display has an integer value of 639. Front Porch has an integer value of 16. The Horizontal Sync Pulse or the Retrace has an integer value of 96. Back Porch has an integer value of 48. The Vertical Display has an integer value of 479. Upper Porch has an integer value of 10. The Vertical Sync Pulse or the Retrace has an integer value of 2.

Processes

The system's functionality is divided into 6 different processes.

Clock Divider

This is a clocked process that depends on the rising edge and change of the main Clock. The default clock is 50mHz. According to the required pixel clock frequency for our VGA display we need 25mHz. So we use a clock divider which creates a clock of 25 mHz. This is achieved by inverting the default clock for every rising edge.

Horizontal Position Counter

During a rising edge, this checks if the horizontal position is equal to the total horizontal pixels. If this is true then it starts from the beginning which is the left side. If that's not the case then it increments the counter by 1 and heads to the next pixel on the right.

Vertical Position Counter

During a rising edge, this checks if the vertical position is equal to the total vertical pixels. If this is true then it starts from the beginning which is the top. If that's not the case then it increments the counter by 1 and heads to the next pixel.

Horizontal Synchronization

This process checks if the actual horizontal position is inside the horizontal front porch or back porch. If it is the case then HSYNC is set to 1. This means in turn if the horizontal position is inside the display area then HSYNC is set to 0.

Vertical Synchronization

This process checks if the actual vertical position is inside the vertical front porch or back porch. Same principle like horizontal synchronization.

Video On Check

This ensures that the video is only on when the actual position is not inside a porch. This is dependent on clock 25's rising edge.

Draw Box

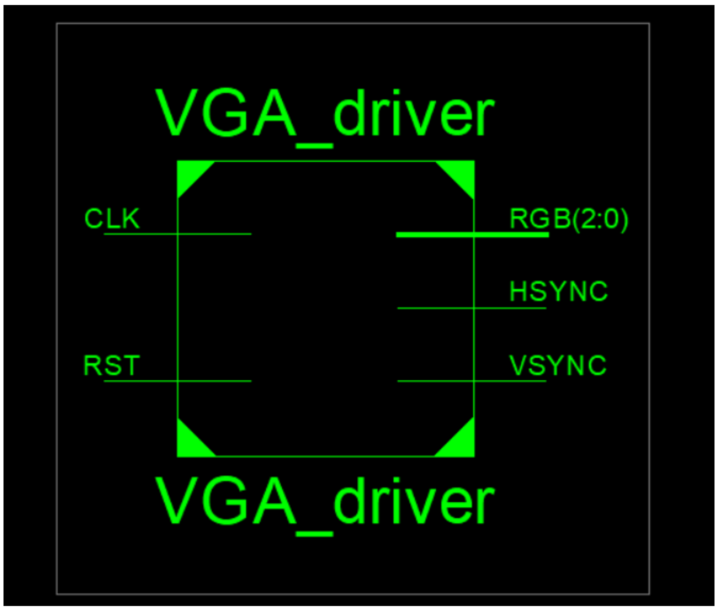
With this process we set the output RGB to 111 if we are between 10 and 60 for the horizontal position and vertical position. If we are outside those boundaries then we set RGB to 000. Thus a white square appears on the VGA monitor since red+green+blue=white and vertical length = horizontal length = 50.

Test Bench

Since everything is counting based on the clock for our VGA driver code, we only need a clock signal as an input in the test bench. In addition the reset has to be set to 0 as well. The testbench is basically only toggling the clock every 5ns and the reset every 100ns.

Synthesis

The VHDL code was synthesized and ran on the Xilinx ISE. Upon successful synthesis the following RTL schematic was generated:



Upon exploring the RTL schematic, the lower level diagram was generated. Due to its large size, this diagram is available in the appendix.

FPGA Implementation

The User Constraints file was created with the following code:

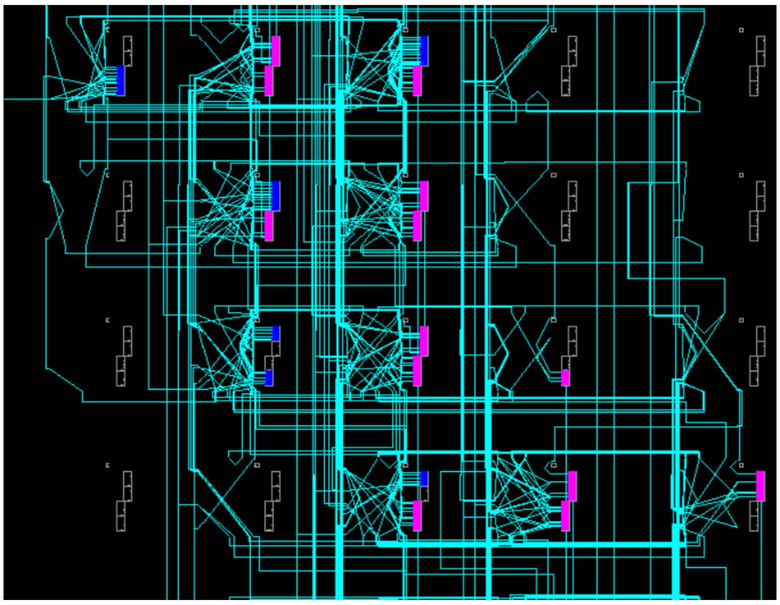
```
NET "CLK" LOC = "C9" | IOSTANDARD = LVCMOS33 ;
NET "RST" LOC = "P14" ;

NET "RGB<0>" LOC = "H14" | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST ;
NET "RGB<1>" LOC = "H15" | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST ;
NET "RGB<2>" LOC = "G15" | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST ;
NET "HSYNC" LOC = "F15" | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST ;
NET "VSYNC" LOC = "F14" | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST ;
```

The Spartan 3E’s user manual was taken as reference for these User Constraints.

Team Bonsai

The Design Summary was generated with zero errors. All signals were completely routed and all constraints were met. The report is available on our Github. A snapshot of the Routed Design is provided below but this does not show the entire figure.



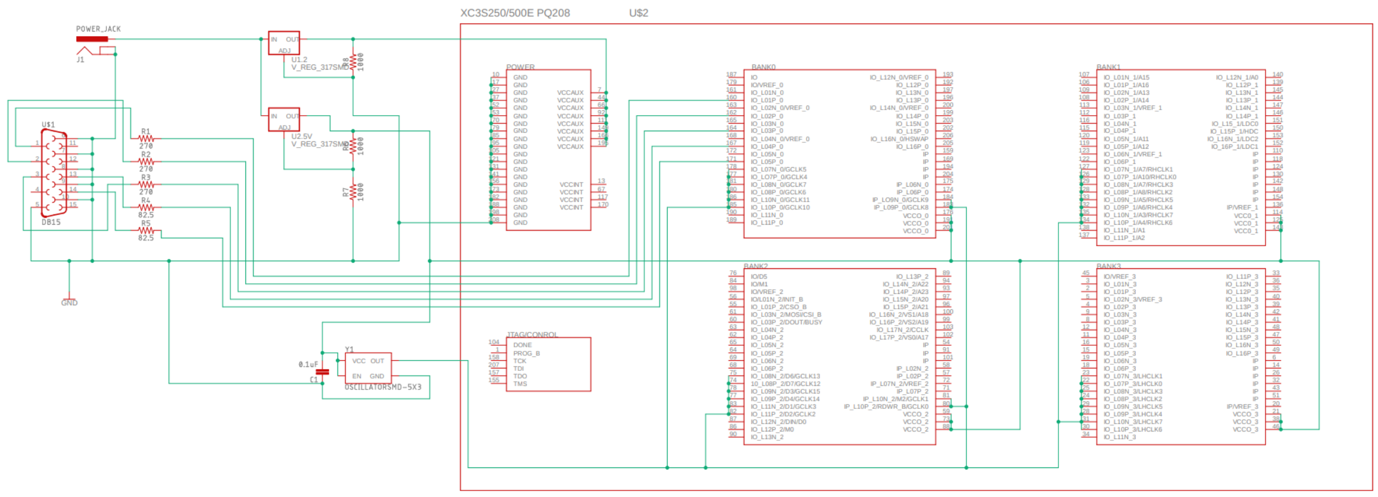
Lastly, the bitstream file was successfully generated and will be implemented on a Spartan 3 board.

PCB Design

In the following, we are describing the design of our evaluation board.

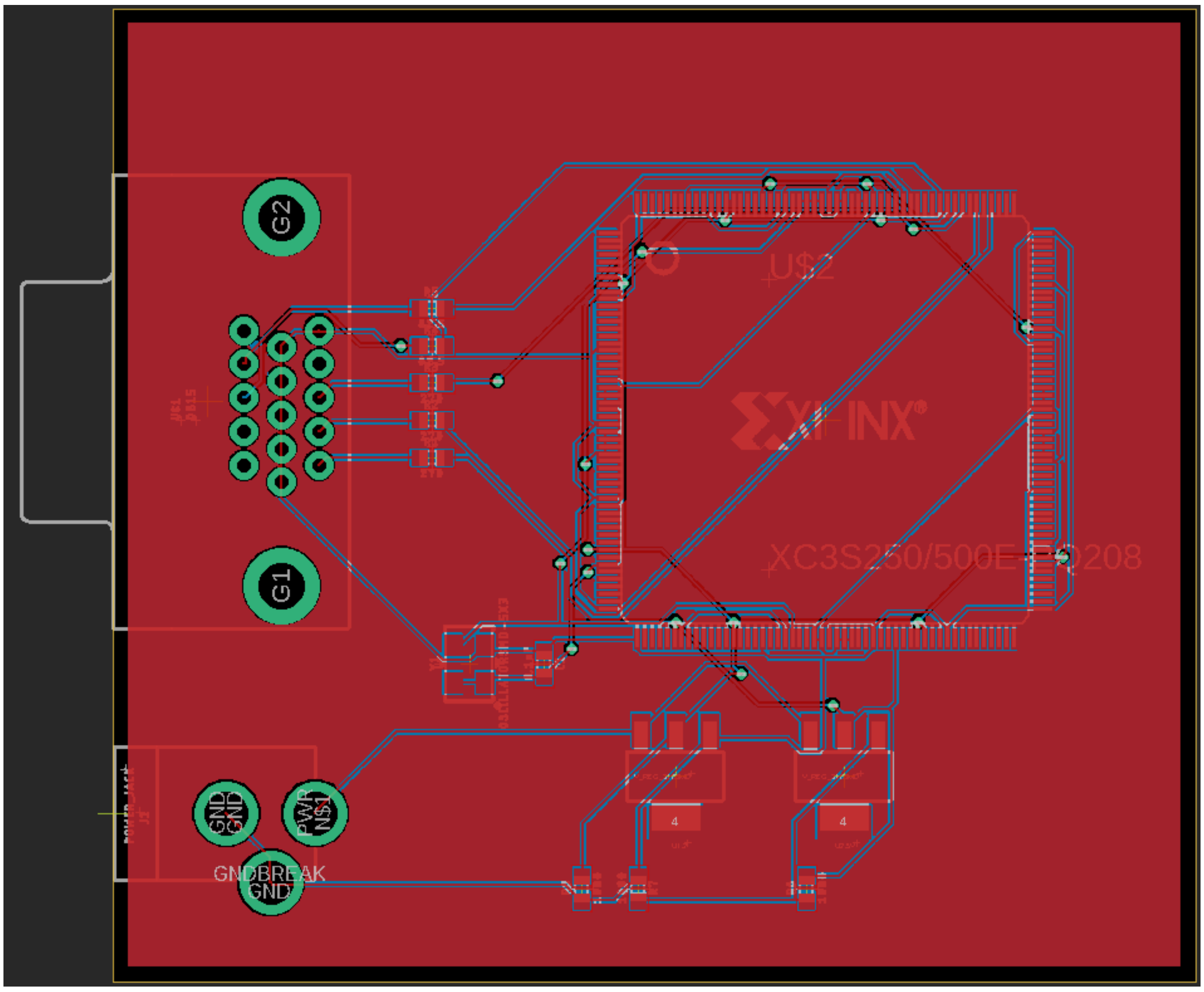
Schematic

We decided on the Spartan 3E from Xilinx for its cheap price and sufficient specifications. We powered it with two voltage regulators that supply 2.5 and 1.2 volts.
The oscillator is providing a 25 MHz clock signal to the FPGA.

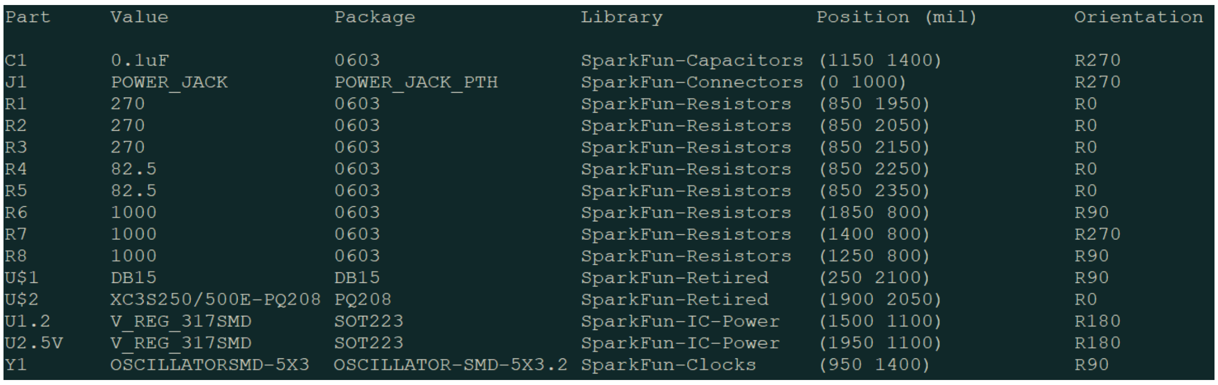


PCB

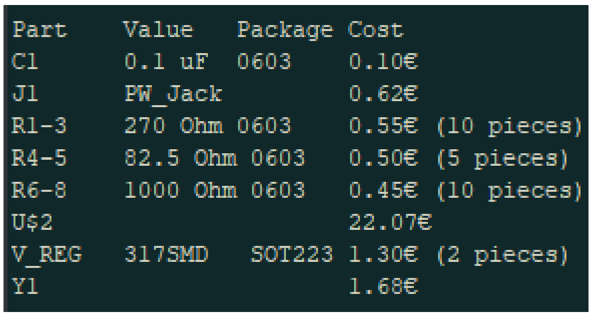
The PCB has a width of 80.1 mm and a length of 67.5 mm.
All connectors are on the left to reduce the designing workload for a case.



Bill of materials



Part	Value	Package	Library	Position (mil)	Orientation
C1	0.1uF	0603	SparkFun-Capacitors	(1150 1400)	R270
J1	POWER_JACK	POWER_JACK_PTH	SparkFun-Connectors	(0 1000)	R270
R1	270	0603	SparkFun-Resistors	(850 1950)	R0
R2	270	0603	SparkFun-Resistors	(850 2050)	R0
R3	270	0603	SparkFun-Resistors	(850 2150)	R0
R4	82.5	0603	SparkFun-Resistors	(850 2250)	R0
R5	82.5	0603	SparkFun-Resistors	(850 2350)	R0
R6	1000	0603	SparkFun-Resistors	(1850 800)	R90
R7	1000	0603	SparkFun-Resistors	(1400 800)	R270
R8	1000	0603	SparkFun-Resistors	(1250 800)	R90
U$1	DB15	DB15	SparkFun-Retired	(250 2100)	R90
U$2	XC3S250/500E-PQ208	PQ208	SparkFun-Retired	(1900 2050)	R0
U1.2	V_REG_317SMD	SOT223	SparkFun-IC-Power	(1500 1100)	R180
U2.5V	V_REG_317SMD	SOT223	SparkFun-IC-Power	(1950 1100)	R180
Y1	OSCILLATORSMD-5X3	OSCILLATOR-SMD-5X3.2	SparkFun-Clocks	(950 1400)	R90



Part	Value	Package	Cost
C1	0.1 uF	0603	0.10€
J1	PW_Jack		0.62€
R1-3	270 Ohm	0603	0.55€ (10 pieces)
R4-5	82.5 Ohm	0603	0.50€ (5 pieces)
R6-8	1000 Ohm	0603	0.45€ (10 pieces)
U$2			22.07€
V_REG	317SMD	SOT223	1.30€ (2 pieces)
Y1			1.68€

This results in a total cost of €27.27 for the components. Some of them are bought in redundancy, as small resistors get easily lost and are cheaper in a larger package.

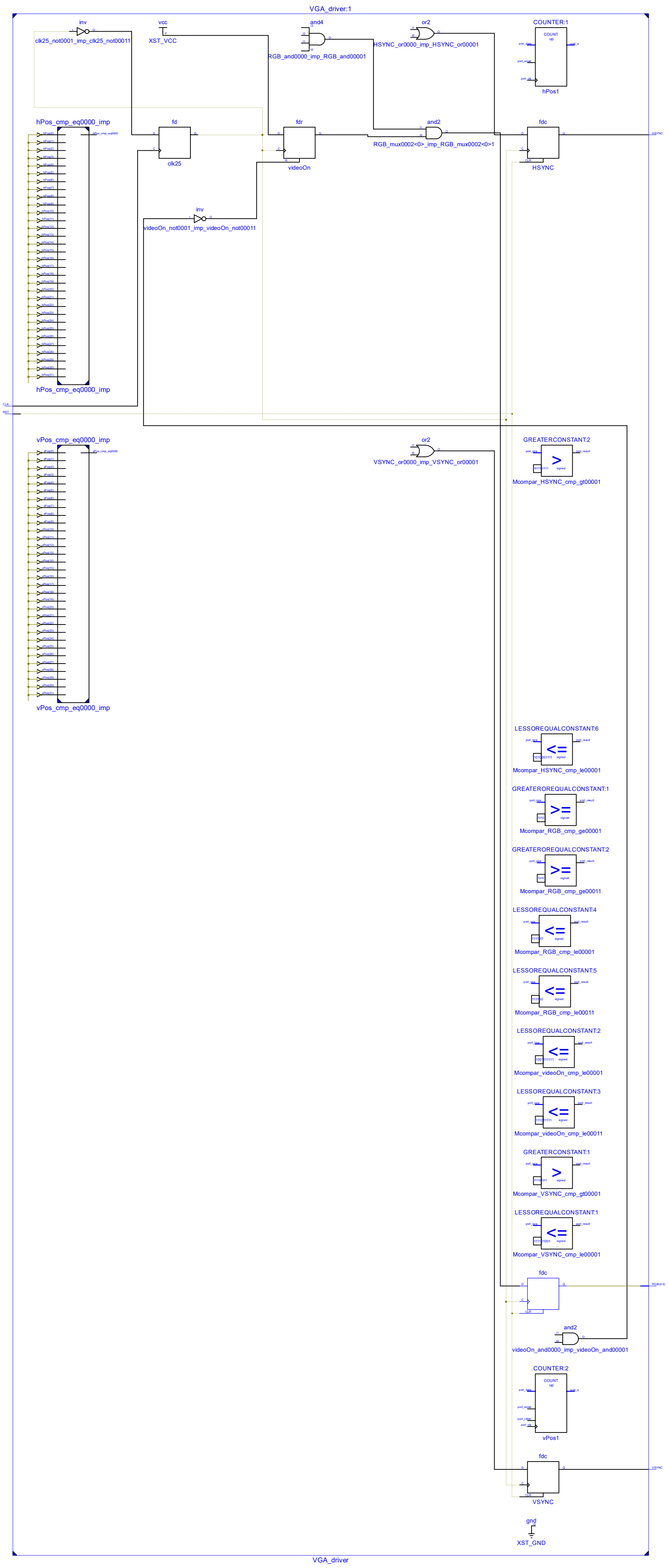
The PCB will approximately cost €4.74. But the shipping cost could drive it up to around €25.

Sources/References

https://github.com/NikolaosKarapoulatidis/Hardware-Embedded-Bonsai
https://digilent.com/reference/_media/reference/programmable-logic/spartan-3e/s3estarter_ug.pdf

Appendix

The following page will contain the detailed RTL schematics.


VGA_driver:1
inv
clk25_not0001_imp_clk25_not00011
vcc
XST_VCC
and4
RGB_and0000_imp_RGB_and00001
or2
HSYNC_or0000_imp_HSYNC_or00001
COUNTER:1
COUNT up
hPos1
hPos_cmp_eq0000_imp
fd
clk25
fdr
videoOn
and2
RGB_mux0002<0>_imp_RGB_mux0002<0>1
fdc
HSYNC
inv
videoOn_not0001_imp_videoOn_not00011
vPos_cmp_eq0000_imp
or2
VSYNC_or0000_imp_VSYNC_or00001
GREATERCONSTANT:2
Mcompar_HSYNC_cmp_gt00001
LESSOREQUALCONSTANT:6
Mcompar_HSYNC_cmp_le00001
GREATEROREQUALCONSTANT:1
Mcompar_RGB_cmp_ge00001
GREATEROREQUALCONSTANT:2
Mcompar_RGB_cmp_ge00011
LESSOREQUALCONSTANT:4
Mcompar_RGB_cmp_le00001
LESSOREQUALCONSTANT:5
Mcompar_RGB_cmp_le00011
LESSOREQUALCONSTANT:2
Mcompar_videoOn_cmp_le00001
LESSOREQUALCONSTANT:3
Mcompar_videoOn_cmp_le00011
GREATERCONSTANT:1
Mcompar_VSYNC_cmp_gt00001
LESSOREQUALCONSTANT:1
Mcompar_VSYNC_cmp_le00001
fdc
and2
videoOn_and0000_imp_videoOn_and00001
COUNTER:2
COUNT up
vPos1
fdc
VSYNC
gnd
XST_GND
VGA_driver