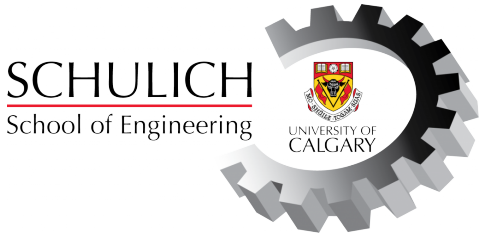


Department of Electrical and Computer Engineering

ENEL 453: Digital System Design

Fall 2017

Lab 3: VGA Display

1 Overview

In this lab project, you will reverse engineer a VGA display driver system implemented in VHDL on the Basys3 board. You will correct the provided reference code and add functionality. This lab is worth 10% of your final grade and is broken up as:

- Pre-lab: 2%
- In-lab simulation: 1%
- In-lab demonstration: 7%

Late penalties for the following week are: -1% Tuesday lecture, -2% Thursday lecture, -3% Friday tutorial, and zero grade beyond the Friday tutorial. All team mates can expect to answer questions regarding any aspect of the project. Work effectively as a team by splitting up tasks and supporting each other as needed.

Complete, working reference VHDL code and constraints file are provided in D2L. You will need to sign out a VGA monitor kit from our technicians.

The VGA display driver system presents two modes of operation:

- Vertical, multi-colored stripes, and
- A moving, bouncing box with switchable colors and adjustable size.

The functionality of the VGA display driver will be used in the Lab 5 project.

2 DELIVERABLES

2.1 PRE-LAB (2%)

1 Calculation of VGA signal levels produced from Basys3 board to VGA monitor.
2 VGA Monitor code top-level diagram.
3 Description of buttons and switches.
4 State-machine logic expressions.

2.2 SIMULATIONS (1%)

These simulations will be for the system after the modifications have been completed for the final design. These can be stored as screenshots for presentation to the TA.

1 Simulation for each VHDL design file, including
    a. Unchanged reference code VHDL files still used in the final design;
    b. Modified reference code VHDL files still used in the final design;
    c. New VHDL files used in the final design; and
    d. VGA Monitor code top-level simulation.

2.3 DEMONSTRATIONS (7%)

1 (2%) Verification that design is fully synchronous, including debounced inputs. Your TA will investigate the VHDL code and you will run synthesis and download the design to the Basys3 after it has been checked.

2 (5%) An additional mode of operation where the bouncing box can be switched to the first letters of the last names of the team members (e.g. for Drs. Onen and Yanushkevich: OY), and back again. Details of representation of the letters are left to the team, to make reasonable design decisions. The quality of the representation will be assessed by the TA according to:
    - 1% - recognizable letters, no movement of letters, adjustable letter color
    - 3% - good quality letters, bouncing letters, adjustable letter color
    - 5% - full behavior of the bouncing, adjustable color, adjustable size box is replicated with good quality letters

3 PRE-LAB - REVERSE ENGINEERING

The prelab work will be checked at the beginning of the lab period. One team member may present the prelab, but all team members are expected to be able to answer questions about the prelab, and marks may be deducted for weak answers.

The purpose of the pre-lab is to become familiar with the design that has been given to you by completing the exercises in the following subsections. Study the VHDL code, simulate it, view the RTL Schematic[1] in Vivado, download the design the Basys3 board, experiment with it, to make sure you understand how it works.

3.1. CALCULATION OF VGA SIGNAL LEVELS PRODUCED FROM BASYS3 BOARD TO VGA MONITOR

Read the entire Chapter 7 VGA Port in the Basys3 Reference Manual (starting on page 11). View the diagram on page 11, and note that there are 4 lines for each color, going through resistors (4kΩ, 2kΩ, 1kΩ, and 510Ω) to a common input for each color (Red, Green, Blue) on the HD-DB15 connector. These 4 lines create a binary weighting for the voltage that appears for each color, by using a voltage divider with an internal 75Ω termination resistor connected in parallel to ground at the color input of the monitor. According to the Reference Manual, VGA signals are 0V for black and 0.7V for maximum brightness.

RECORD: Fill in the voltage values in the table below. Is the full scale (i.e. "1111") voltage correct for VGA? If not, what issues can it cause?

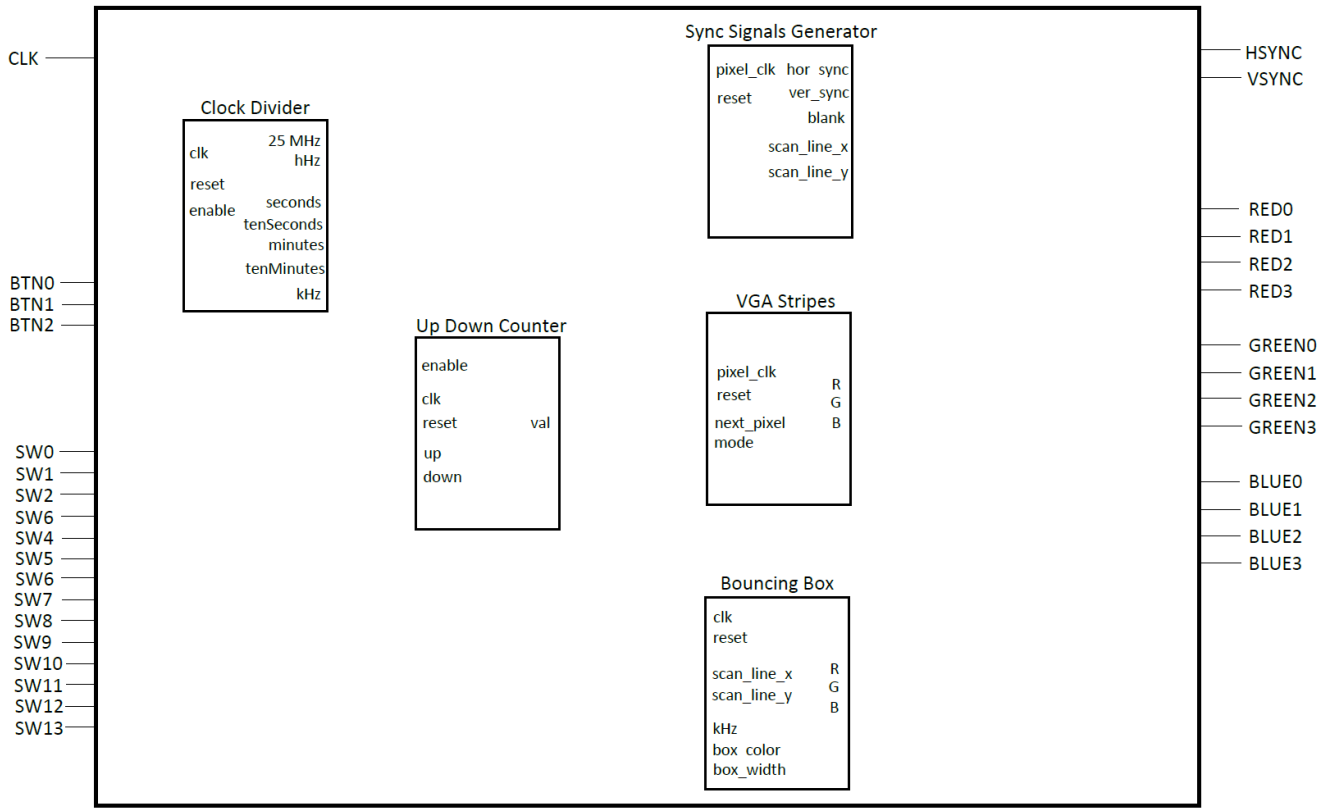
RED3	RED2	RED1	RED0	RED input voltage to HD-DB15 (voltage divider value) (V)
0	0	0	0	0
0	0	0	1	0.0484
0	0	1	0	0.0968
0	0	1	1	0.1452
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

[1] Be sure to view the RTL Schematic in Vivado, it is a very useful feature

3.2. VGA MONITOR CODE TOP-LEVEL DIAGRAM

Review the VHDL code provided in D2L. Determine what is the top-level entity.

RECORD: Complete the internal connections in the diagram below.



3.3. DESCRIPTION OF BUTTONS AND SWITCHES

RECORD: By reviewing the code (and running simulations, downloading, experimenting, etc., if necessary), determine the function of each switch and button.

Switch/ button	Function
BTN0	Reset. Places system in reset.
BTN1	
BTN2	
SW0	
SW1	
SW2	
SW3	
SW4	
SW5	
SW6	
SW7	
SW8	
SW9	
SW10	
SW11	
SW12	
SW13	

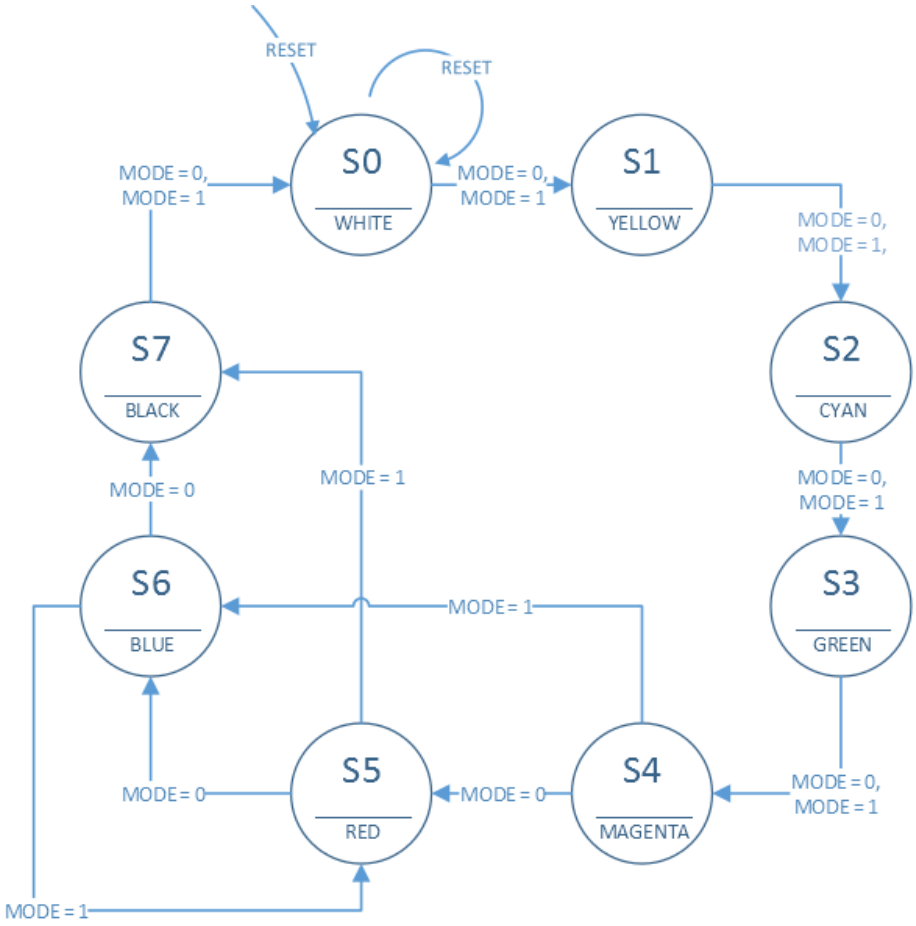
3.4. STATE-MACHINE LOGIC EXPRESSIONS

The **vga_stripes_dff.vhd** file controls a mode of operation of the VGA display driver whereby colored stripes are displayed. There is a state machine in this file that controls the progression of the colors, based on the input signal **MODE**.

When MODE = 0, the colors progress WHITE > YELLOW > CYAN > GREEN > MAGENTA > RED > BLUE > BLACK.

When MODE = 1 the chain becomes WHITE > YELLOW > CYAN > GREEN > MAGENTA > BLUE > RED > BLACK.

When the reset signal is held high, the color is WHITE. The state diagram is shown below.



The file **vga_stripes_dff.vhd** uses a state machine design based on logic expressions. You will need to be familiar with how to design a state machine using this method. The table below describes the logic expressions.

Record: Complete the table for the state machine.

Color	State#	Description	D-Input Expression
WHITE	0	Always transition from BLACK	d(0) = q(7)
YELLOW			
RED	5	Transition from MAGENTA if mode=0 otherwise transition from BLUE	d(5) = (q(4) AND NOT *mode*) OR (q(6) AND *mode*)

An alternative VHDL file containing the state machine is **vga_stripes_dff2.vhd.** The state machine has been described in different style which maps the state diagram more directly to the code. This style of writing a state machine is typically easier to code and more maintainable. Compare the two versions and select which one is a better design to use.

4 SIMULATIONS

As part of good design practice, testbenches are written to test and verify that each entity works as expected. Complex designs are composed of lower-level units that are integrated into higher-level units (e.g. in Lab 2, **downcounter** was integrated into **clock_divider**, which was then integrated into **digital_clock**).

Unit testing is when each of the lower-level units is tested individually with a testbench. A design file with generics, such as **downcounter**, only needs to be unit tested once, with reasonable values for the generics. *Integration testing* is when the lower-level units are combined to become a higher-level unit, which is then tested. At the highest level of integration, we perform *system testing* of the model, which would be on our top-level entity. After the design has been downloaded to the FPGA, we can perform *board-level testing*, which depending on the context, can also be referred to as system testing. The testing methodology where lower-level units are tested first, then higher-level, is called *bottom-up testing*.

Show your TA the testbench simulations of each design file as listed in 2.2, including your top-level, following the bottom-up testing methodology. These simulations must be of the completed design (after it is fully synchronous and letters are displayed). All team members must be able to answer questions regarding the simulations or marks may be deducted.

5 DEMONSTRATION

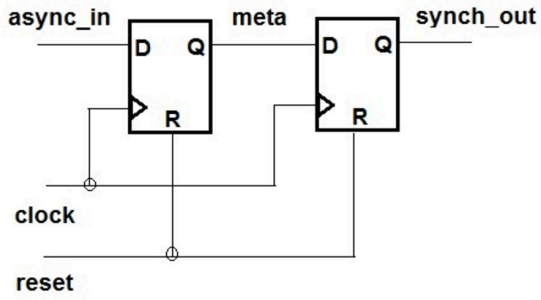
The VHDL source code must be reverse-engineered to add the following functionality.

5.1. FULLY SYNCHRONOUS DESIGN

In digital systems, good design practice is synchronous design, which means that all flip-flops are clocked from the same clock source[2] and the flip-flop timing is not violated. In modern, very large designs, this design guidance is modified, but this topic falls well outside the scope of this course.

The VHDL source code features a violation of the principle of synchronous design, because the signal `pixel_clk` is used as a clock source within the design (i.e. as the clock input to flip-flops). Redesign the system so that only the `clk` signal is used as a clock source, throughout the design.

An additional concern is that the inputs from the switches and push-buttons are also asynchronous. The classic method is to use a synchronizer circuit to bring the asynchronous signals to the synchronous domain. The synchronizer is 2 flip-flops connected serially together, as shown in the circuit below.



In the circuit, async_in would be connected to your switch or push-button, and synch_out would be connected to your digital circuit. Implement this circuit for every switch and push-button you are using.

A further complication to switches (and push-buttons) is "switch bounce," whereby multiple transitions can occur after a switch has been switched, due to mechanical contact bouncing of the switch, until it stabilizes. This bounce can occur for several milliseconds. Design and implement a debouncing circuit that will connect to your synchronizer circuits and provide debounced signals to your system. One crude method would be to sample the synchronized signals every 50ms or so. A better example is in the following link:

https://eewiki.net/pages/viewpage.action?pageId=4980758

However, you must understand the circuit in order to use it.

[2] e.g. only `rising_edge(clk)` is used

5.2. LETTERS DISPLAY

One of the modes of operation of the VGA display driver is to generate a box that bounces from the edges of the monitor. This box' size can be adjusted to be larger or smaller, and its color can be adjusted.

Study the Basys3 board reference manual Chapter 7 VGA (starting page 11). Study the VHDL code, simulate it, view the RTL Schematic in Vivado, download the design the Basys3 board, experiment with it, to make sure you understand how it is drawing graphics on the VGA monitor.

Then, modify the code so that you can display letters that correspond to the first initials of the last names of each team member. The letters must replace the bouncing box. A way to switch the display between the box and the letters, and back again, must be developed and implemented.

The particular implementation and features of the letters are up to the team to decide. Section 2.3 describes the grading criteria for the letters display.