ECE3829 Lab 2: VGA Display Design

A22

Required deliverables:

- Functionality demonstrated and signed off.
- Archived project and a single pdf of your Verilog modules submitted to canvas at time of sign-off.
- Lab report submitted to canvas by the deadline.

Getting Started and Counter Tutorial:

Before starting this lab, you may wish to complete the counter tutorial.
It walks you through to following processes.

- How to generate and instantiate the MMCM clock modules needed to generate the 25 MHz for the lab.
- How to generate a counter enable signal.
- How to analyze your implementation results.
- How to setup a simple simulation to help you debug .

Important Material:

This lab requires the use of a monitor with a VGA input. In addition to the Basys3 board, you will need access to a **Monitor with a VGA input** and a **cable** to connect your Basys3 board to the monitor.

Read the VGA display section in the reference manual to understand how the interface works.
https://reference.digilentinc.com/reference/programmable-logic/basys-3/reference-manual

Use the **vga_controller_640_60.vhd** file provided on canvas. It is written in VHDL but the comments at the beginning for the document are useful and the port declarations are very similar to Verilog.

Use the Master XDC file from Digilent for your pin constraints. Only include the pins you are using and the and CONFIG_VOLTAGE and CFGBVS properties in your constraints file. **Do not change the port names in the master XDC file**, change your port names to match those in the XDC file.
XDC File Link: https://github.com/Digilent/digilent-xdc/blob/master/Basys-3-Master.xdc

Reminder:

- Remember to update your file header comment in each file and follow recommended coding guide line presented in class.
- The specific implementation details defined in the lab and commenting guidelines listed in the grading rubric at the end of the lab will factor into credit received. Follow the instructions.
- You must get your lab signed-off in order to receive credit for the Verilog code and report portions of the lab. If you run out of time, you can still sign-off with what you have working for partial credit. The Verilog you submit to canvas must be the code used during your sign-off.

All code should be written in Verilog for this lab.

Late Submissions:

Please check Canvas for the day and time of each lab submission. Any assignment submitted after the deadline in canvas, will receive a 20% late deduction and can be submitted up to a week late. Unless special arrangements have been made in advance, no submissions after one week late will receive credit. Labs must be signed-off to receive credit for any part of the lab.

Overview

This lab is split into the following steps.

1. Generate a MMCM module using the Vivado IP wizard (see tutorial for details).
2. Updating your seven_seg module to show all four characters at once.
3. Displaying information to the VGA monitor both stationary and moving.
4. In a single top-level module , bring all the functionality together.
    a. All modules that require a clock should be driven by the 25 MHz MMCM output clock and lock output from the MMCM should drive the resets of your logic. All your logic should have a reset condition with a synchronous active low reset.
    b. It is OK to have a few assign statements if something needs inversion or a simple one-line assignment. Other than those special cases, all your logic should go in modules, these modules should be instantiated in your top-level design with wires connecting them.
5. Designing a debounce circuit for use with your slider and push button inputs and instantiate one debounce for each specified input in your top-level module.

Global requirements

1. All sequential logic in this design must use the positive edge of the 25 MHz clock output from the MMCM in their sensitivity list.
2. All sequential logic in this design must use a synchronous active low reset (reset_n) driven by the lock output from the MMCM. Note that the vga controller VHDL code has an active high reset, so you will need to invert reset_n signal to create a reset signal for the controller.
3. In the top-level module, the pin names in the Digilent Master XDC file must be used as your port names. Do not change the Master XDC file names, change your names to match theirs.

Part 1: Generate the Clock Module

Use the Counter Tutorial to generate an MMCM clock module that outputs a 25 MHz clock and a lock signal. All sequential logic should use the positive edge of the 25 MHz clock and have their active low reset signals driven by the lock output of the MMCM module.

Top level connections to the MMCM include

- A push button connects to the reset input (no debouncer on this push button)
- 100 MHz clock pin connects to its clock input
- 25 MHz clock output pin drives rest of logic
- lock output drives active low reset_n signal and inverted lock drives active high reset signal in the vendor provided VHDL code.

Part 2: Seven Segment Display

Modify the simple seven segment display from lab 1 to create a seven_seg module that can display a value from “0000” to “FFFF” on the all four seven segment displays at once. Use a clock to cycle through the four digits. The update rate should be such that all four displays are not reasonably bright and do not flicker. During reset, the displays should be off. Do not include other functionality in this module as you will need to reuse it for later labs. The updated seven_seg module will have the same ports as in lab 1 plus two more listed below. The 4-bit select signal is no longer be needed as all displays will be shown at once.

- 25 MHz clock input
- Active low reset input

In the top-level module, the four display input ports should be assigned to your WPI ID numbers.

Part 3A: VGA Display

Create a VGA display select module to interface with the VGA controller provided on canvas and the VGA pins on the FPGA.

- Use sw[1:0] slide switches to select what to display on the monitor.
- They are used to select which one of the following options are displayed.
    - Completely yellow display.
    - Vertical bars of alternating red and white. Each bar is 16-pixels wide in the horizontal direction.
    - A black screen with a green block 128-pixels wide by 128-pixels high in the top right-hand corner.
    - A single horizontal blue stripe 32-pixels wide at the bottom of the screen. Rest of the screen is black.

Color values are 12-bit values with 4 red bits, 4 blue bits and 4 green bits. All colors should be assigned to a 12-bit parameter. To break the color signal down into its red, blue, green values use part selects.

You should not need any complicated math to do this, use bit locations and binary logic when you can. For example, if something needs to repeat every 8-bits, bit[3] of a count value would be high for 8 bits and low 8 bits.

Part 3B: Moving Block

When a push button is pressed show a blue screen with a red block 32-pixels wide and 32-pixels high with a starting location of at the top of the screen and in the middle horizontally. Shift the block one block width down at a 2 Hz rate while the button is pushed. The move button takes priority over the sw[1:0] controls.

There are many ways you can do this. One way to create a counter that counts from 0 to 14 at a 2 Hz rate update rate to control the vertical position of the block. This could be done by creating a 2 Hz enable signal from a counter similar to what was done in the counter tutorial. Remember all sequential blocks should use the posedge of the 25 MHz clock in their sensitivity list. At the bottom of the screen you can wrap around, stop or reverse direction.

Part 4: Put it all together in top_lab2.v

In the top_lab2.v module you will instantiate and connect the following items

- MMCM module
- New synchronous Seven Segment Display module
- VGA controller provided in canvas, vga_controller_640_60.vhd
- Your VGA display select module(s)
- A debouncer on sw[1:0] and the button used for move mode. (see part 5)

It may be necessary to make some simple assign statements in the top level, to invert the reset signal for the VGA controller, to set your WPI ID, concatenate busses, etc… but more complicated logic must all go into a module.

ALL port names to the FPGA pins must match the port names in the XDC file. The XDC file port names should not be changed. Include only the ports you need, comment out or removed unused ports from your XDC file.

Part 5: Debouncing Circuit

For each slider and push-button input in your design except for the reset to the MMCM, you need a debouncing circuit. The debouncing circuit removes the glitching that occurs when the switches/buttons change state.

Your debounce circuit should have the following ports

- One 1-bit input.
- One 1-bit output.
- 25 Mhz clock input
- Active low reset input

Your debounce circuit should do the following

- Create a counter that takes 10 msec to reach its terminal count.
- If the input changes state the counter is cleared / set to zero count.
- When the counter reaches its terminal count, it assigns the output to the input value and resets to its count to zero.
- During reset the count value should be zero and the output should be assigned to the input by passing the debounce logic.

Extra Credit

5 points lab bonus points for any good improvements or enhancements to your design. You must demo on board and describe the additional functionality well in your report. For example, print ECE3829 on the VGA monitor or make a Pong game! If you do something really extraordinary additional points can be granted beyond the 5 points at the discretion of Prof. Stander.

Sign-Off Procedure

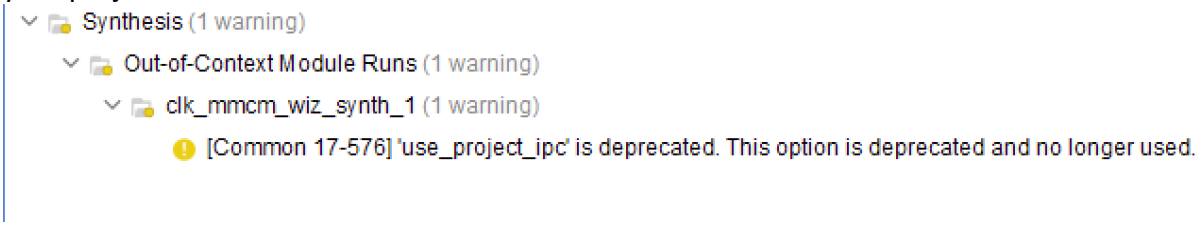
Signoff can be done in-person or remotely via zoom. If you are signing-off remotely, you will need to have the ability to share your screen with the TA and also have video capability to display the Basys-3 board functions and the vga monitor clearly.

You may repeat steps 1-4 as many times as needed. But you may only submit your archived project and pdf of your Verilog code once during your final demonstration. No changes to these submissions are allowed after your final demonstration.

1. Have project fully Generated with Bitstream and be ready to show to the TA.
2. Have your project archived and all your Verilog code in a single pdf file ready to submit.
3. In Vivado show your messages window with Warnings and Errors displayed to the TA. TA will verify that only expected warnings occurred. Some warnings are expected. However, points will be deducted if warnings are generated by your code that are not expected.
4. Download your bit stream on to your board with TA present.
5. Demonstrate the functionality in the Functionality Check List. TA will record results and assign points for all functionality that is correct.
    a. You can demo as many times as needed.
    b. You will always be rewarded points for the functions that are working.
6. Once you are happy with your demonstration results.
    a. You will submit your archived project in canvas immediately.
    b. You will submit the single pdf version in canvas immediately.
    c. TA will verify they are submitted and complete the sign-off process.

Expected Warnings

These are expected the IP provided by Vivado is using unsupported options they will not affect your project.



Sign-Off Check List (50 points)

Project Status (10 points)

1. Expected files present in project (3 points)
   - Top level, MMCM, seven segment display, vga_controller_640_60.vhd, vga display select module(s), debouncers, and constraints file present in project.
2. Vivado warnings (5 points)
   - Full credit no unexpected warnings.
3. Project programs part successfully(2 points)

Functionality Check List (36 points) / 6 points each item

1. WPI ID shown on 7 segment displays:
   - Last four digits of your WPI ID should appear on the 4 7 segments display at the same time with good brightness and no flickering.
2. VGA Displays Yellow Screen:
   - The entire display is yellow.
3. VGA Displays vertical bars:
   - Displays vertical bars alternating red and white. 16-pixels wide in the horizontal direction.
4. VGA Displays green block on black background:
   - A black screen with a green block 128-pixels wide by 128-pixels high in the top right-hand corner.
5. VGA Displays single horizontal stripe :
   - A single horizontal blue stripe 32-pixels wide at the bottom of the screen. Rest of screen black.
6. Moving Block when button pushed
   - Blue screen with a red block 32-pixels wide and 32-pixels high with a starting location of in the middle of the top of the screen that moves down one block width at a 2 Hz rate while a button is pushed.

Extra Credit Demonstrated (up to 5 points)

- Extra credit must be demonstrated at time of sign-off and documented in report to receive credit.

Canvas Submissions Completed (4 points)

1. Archived project submitted in canvas at time of sign-off
2. PDF of Verilog code submitted in canvas at time of sign-off

Rubric for Verilog Code in PDF (20 points)

1. Code complies with specific instructions detailed in the lab description. (7 points)

- Top level port connections use the names in the original Digilent master XDC file.
- Top level block instantiates MMCM module, seven segment display module, vga_controller_640_60 module, vga display module(s) and debouncers on sw[1:0] and move push button.
- Clock modules inputs and outputs are hooked up as specified.
- Debouncer module is written to spec: has required ports, is implemented using a 10 msec counter, count is cleared when input toggles, reset conditions are met.
- Seven segment display retained ports from Lab 1 + clock and reset_n inputs – select signals. WPI ID is assigned in the top-level module.
- VGA display logic uses sw[1:0] to select which of the four images to display and a push button to enable the moving block.
- VGA display logic uses parameters for color definitions.

2. Code is well commented (5 points)

- Each file should start with a commented header. Files generated with Vivado will create a template for you, fill in the values for your lab information. If not generated by Vivado refer to the decoder project example for format.
- Each Input and Output should have an in-line comment as to what it does.
- Similar to input and outputs, wire and reg statements should be commented
- If using Parameters, a comment describing how the parameters are used and their values should be added.
- Each set of concurrent statements should have a comment describing what the block of code does.

3. Code is well formatted, organized and written (8 points)

- Code is well organized with separate assignments and always blocks for each logical function.
- Top level block contains lower level modules with only a few assign statements.
- Signal names are descriptive but not to long.
- Code is appropriately indented with only one statement per line.
- No complicated math (* and /) not needed everything can be done using part selects, shifts, bit and logical operations.
- All logic driven with the posedge of the 25 MHz clock.
- All student generated logic has a reset state driven by a synchronous active low reset.
- Best Practices outlined in class were followed.

Rubric for Report (30 points)

1. Brief Introduction / Problem Statement (5 points)

2. General Overview of solution (15 points)

- Include a block diagram of your top-level block (see block diagram in Lab 1 for example).
    - Label your module names and signal names on the block diagram.
    - Show connections to the FPGA pins
- Write a brief description of what each module does and how your modules connect together.
- If you implemented extra credit, include a description of the extra features.
- Write a detailed description of how each module was implemented.
    - For more complicated modules where multiple always blocks are used draw a block diagram of the module if it helps describe what is going in. I picture is worth a 1000 words.

3. Implementation Summary (5 points)

- Generate a Utilization report in Vivado and analyze the results(example shown in counter tutorial). The report shows how many LUTs and Registers are used by each module.
    - Summarize the results of the utilization report.
    - Explain how the LUTs and Registers were used in each module you wrote and comment on whether this met your expectations or if it was different from expectation.
- If unexpected warnings occurred in your project. Explain the warnings and how they should be resolved.

4. Conclusion (5 points)

- What problems or challenges were faced in implementation?
- What solutions were used to solve them?
- What lessons were learned from the project?
- Suggestions for further improvements and extensions?
- Where there particular things you liked about the lab?