Use a VGA monitor and the provided Ambient Light Sensor to create a light sensor monitor on the Basys3 (or Nexys3) board.

This project involves the design of a number of interfaces to peripheral devices. It drives the VGA display and reads the light sensor using an SPI interface. The lab involves the use of multiple sequential circuits (counters, shift registers, etc. but it is not necessary to use state machines), the Xilinx Core Generator (for the DCM or MMCM), and use of existing IP (the Digilent VGA controller). There are multiple parts to this project. To be successful will require a good design and debugging approach. Make simpler projects that you can test and debug separately and then combine them together.

This lab (and report) can be completed individually or with a lab partner (recommended) – it will be worth 35% of your course lab grade.

**Lab Signoff Deadline:** during a lab section (during the week of Feb 1 \(^\text{st}\)) – use the signoff sheet that describes what you have working AND bring along your Verilog listings for the TAs to check (they will return these to you to hand in with the report). The lab report is due in class on Friday February 5 \(^\text{th}\).

**Description**

Preliminary:
- 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 four seven segment displays. The input to the module should be a 16-bit wide bus, with four bits used to indicate the value to be displayed on each of the seven segments. You will also need a clock to cycle through the four digits.
- Make this a separate module – you will use this module in this and later projects.
- Test this out by using the slide-switches to enter various numbers.

Part 1: VGA display
- Create a VGA display using the VGA controller provided by Digilent (just the 640 by 480 version) – see information at end of this doc.
- Use the MMCM to create a 25MHz clock required for the VGA pixel clock.
  - (see the MMCM tutorial for how to add this IP to your design).
  - Note: only connect the 100MHz FPGA clock to the MMCM (nothing else)
  - Add a period constraint to your XDC (or UCF) to match the Basys3 board 100MHz clock frequency.
- Use the slide-switches to select and display the following patterns
- Complete red display
- Complete yellow display
- Complete green display
- A black screen with a white rectangle 200 pixels wide by 100 pixels high in the center of the screen (one pixel wide sides)

(These should be relatively easy once you start working with the VGA controller provided by Digilent – don’t forget to include the ‘blank’ signal)

Part 2: Light Sensor Interface

- Create an SPI interface to be able to read the 8-bits of light sensor information from the PmodALS module provided.
- Use the same MMCM to create the serial clock required for the ADC SCLK.
  - For Nexys3: Set the clock frequency to 4MHz - you will need to use an ODDR2 component to drive the SCLK output (see example at end of this doc)
  - For Basys3: Set the MMCM freq to 10MHz and drive the ADC SCLK at 1MHz
- Use a counter or shift register to create the ADC CS signal.
  - Verify the SCLK and CS signals are correct with an oscilloscope.
- Use a shift register to read in the 8-bits of ADC data
- Display the light sensor value on two of the seven-segment displays.
- Modify the color of the white rectangle from Part 1 so it changes color as the light sensor value changes
- Capture a new light sensor value every 1ms (1KHz)
- Use the other two seven-segment displays to display an incrementing count value (at a 2Hz rate) in decimal from 00 to 99 and repeating.
- Capture an SPI ADC 16-bit transfer using an oscilloscope (show the CS, SCLK, and SDO signals on the scope capture) and describe and include this in your report.
  - Note: For all ‘scope pictures, preferably take a screen capture with a USB flash drive rather than a camera picture. You should be able to clearly see all the signals and the timebase.

Extra credit
Up to 10% lab bonus points for any good improvements or enhancements to your design (must demo on board and describe in your report). For example make a system that can count objects passing through a light beam.
Reference Material

Read the *Seven Segment and VGA Port* section in the Nexys3 Reference Manual.

**For Nexys3: ODDR2 to drive the SCLK output to the Light Sensor**

--- ERROR:Place:1205 - This design contains a global buffer instance, `<sclk_buf>`, driving the net, `<sckk>`, that is driving the following (first 30) non-clock load pins off chip.
--- `< PIN: sclk.O; >`
--- This design practice, in Spartan-6, can lead to an unroutable situation due to limitations in the global routing. If the design does route there may be excessive delay or skew on this net. It is recommended to use a Clock Forwarding technique to create a reliable and repeatable low skew solution:
--- instantiate an ODDR2 component; tie the .D0 pin to Logic1; tie the .D1 pin to Logic0; tie the clock net to be forwarded to .C0; tie the inverted clock to .C1.

You can find a Verilog template (example code) for the clock forwarding circuit from within ISE. Click on the `<lightbulb>` icon to bring up the Language Templates pane. From within this pane, select:

**Verilog > Synthesis Constructs > Coding Examples > Misc > Output Clock Forwarding Using DDR**

```verilog
// Clock forwarding circuit using the double data-rate register
// Spartan-3E/3A/6
// Xilinx HDL Language Template, version 14.7
ODDR2 #(  
  .DDR_ALIGNMENT("NONE"), // Sets output alignment to "NONE", "C0" or "C1"
  .INIT(1'b0), // Sets initial state of the Q output to 1'b0 or 1'b1
  .SRTYPE("SYNC")  // Specifies "SYNC" or "ASYNC" set/reset
) clock_forward_inst (  
  .Q(sclk), // 1-bit DDR output clk
  .C0(clk_4M), // 1-bit clock input
  .C1(~clk_4M), // 1-bit clock input inverted
  .CE(1'b1), // 1-bit clock enable input
  .D0(1'b0), // 1-bit data input (associated with C0)
  .D1(1'b1), // 1-bit data input (associated with C1)
  .R(1'b0), // 1-bit reset input
  .S(1'b0) // 1-bit set input
);
```

---

From Xilinx UG382 “Spartan-6 FPGA Clocking Resources”

**Figure 3-13: Zero Delay Buffer for Single-Ended Clocks**

---
Notes:

Your final design should combine parts 1 and 2.

Prepare a sign-off sheet and demo your system during one of your lab sessions before the deadline. Have your printed Verilog source files ready so they can be checked during the demo (don’t forget to include names, description, and comments).

Write a report including: an introduction, a description of your design including good block diagrams showing how you implemented the design, a section describing how many flip-flops your design used and why? Include part of the synthesis file that displays warnings – explain these. Include a conclusion describing any problems or issues you had and any lessons learned. Include your signoff sheet and your source files in an appendix.
Grading Guidelines

• [50 pts] Implementation
  - [50 pts] Design works on board and meets requirements

• [20 pts] Source Code – Verilog in Appendix
  - Code style and comments (well-commented and tab-indented code!)
  - Use of *case* vs. *if*, spaghetti code vs. structured, etc.
  - Recognizable implementation of "standard" elements (state machines, counters, shift registers, clock dividers, decoders)
  - Good modular design
  - No latches or other synthesis problems

• [30 pts] Lab Report
  - [5 pts] Brief Introduction / Problem Statement
  - [15 pts] General overview of approach to solution and description (include Block and State Diagrams with descriptions) and oscilloscope pictures
  - [5 pts] FPGA resource usage (# flip-flops with explanation) and listing and explanation of warning messages (don’t copy all the Xilinx reports – just the relevant sections)
  - [5 pts] Conclusions
    - Problems faced in implementation
    - Solutions used to solve problems
    - Lessons learned from the project
    - Suggestions for further improvements and extensions

• [10 pts] Extra points
  - Possible extra points for good additional features or capabilities (need to demo on board and include description in report)
ECE 3829: Lab 2 sign-off sheet

Name: ____________________ ECE Box #: _____________________

Name: ____________________ ECE Box #: _____________________

Preliminary
The seven segments display works (0000 to FFFF) ________________

Part 1 (VGA)
The VGA displays different colors ___________________
The VGA displays a white rectangle ___________________

Part 2 (Light Sensor)
The light sensor displays the correct value ________________
on the seven segment displays (dark to light)
An incrementing count (00 to 99) is displayed ________________
The rectangle changes color ____________________________

All combined
All parts are combined into one project _________________

Extra Credit (describe)
Example: count objects _______________________________
_________________________________________________________________