ECE3829 Lab 3: Light Sensor

A22

Required deliverables:

- Lab functionality demonstrated and signed off either in person or via zoom.
- An archived project and a single pdf of your Verilog modules submitted to canvas at time of sign-off.
- A Lab report submitted to canvas by the deadline.
- You must get signed-off to receive credit for the Verilog Code and Report .

Important Material:

This lab requires the use of a Digilent PMOD ALS PMOD. The Digilent web-site has reference material
https://reference.digilentinc.com/reference/pmod/pmodals/start

The ALS PMOD uses the ADC081S021 Analog-to-Digital converter to sample the Light sensor value. Read the device data sheet and interface example posted in Lab 3 on canvas.

- PmodALS interface example.pdf
- PmodALS_data.pdf

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. An exception this rule, you may change the XDC file names from JA[3:0] to JA0, JA1, JA2, J3 to allow the port directions to be different.
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 an MMCM to generate 10 MHz clock and synchronize the locked output to create an active low reset_n signal.
2. Create a light sensor interface module
    a. Implement a state machine to control the sampling of the light sensor.
    b. Implement an SPI interface to interface with the light sensor, use a shift register to read and store the sensor value.
3. Display the sensor value in hexadecimal on the seven segment display.

Global requirements

1. All sequential logic in this design must use the positive edge of the 10 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 synchronized locked output from the MMCM.
3. In the top-level module, the pin names in the Digilent Master XDC file must be used as your port names.
    a. In Lab 3 there is an exception this rule, you may change the XDC file names from JA[3:0] to JA0, JA1, JA2, J3 to allow the port directions to be different for each pin.

Part 1: Create a clock new clock module with a synchronized reset.

Use the Counter Tutorial to generate an MMCM clock module that outputs a 10 MHz clock and a locked output signal.

Create a new module named clock_gen that instantiates the MMCM clock module plus synchronization logic for the systems reset_n signal. Synchronize the locked output by putting it through two consecutive flip-flops without a reset. This will make the chances of metastability very small. The flip flops should use the 10 MHz clock . Instantiate your new clock module in the top level of your design.

All sequential logic should use the positive edge of the 10 MHz clock and have their active low reset signals driven by the locked output of the MMCM module.

btnC should be used as the reset input to the clock_gen module.

Part 2: Sensor Interface and Control

Create a light sensor module(s) that will capture a new light sensor value once per second (1 Hz).

- Declare a port parameter that controls the sensor sample rate in your port declaration so that the sample rate can be easily overridden by your testbench if you choose to simulate your design.
- The module should have at least one state machine for the overall control of the sensor interface.
- The module should have a sensor value output port that can connect to the seven segment display module used in Lab 2. The seven segment display module should be in your top level module not in your sensor module.
- The SCLK signal going to the light sensor is not an internal clock, it is just another signal and **SCLK should not be used in any sensitivity lists.**
- One way to generate SCLK, CS_N and SDO signals running of the 10 MHz input clock is to generate rising edge and falling edge enable signals.
    - These enables could then be used to
        - Generate an SPI SCLK signal between 1 MHz and 4 MHZ
        - Generate an CS_N output signal.
        - To enable reading the SDO input signal into an SIPO block.
- All signals must meet the timing requirements listed in the device specification.
- The notation SCLK and SCK are used interchangeably for the serial clock to the light sensor. The CS_N and CS# notations are used interchangeably for the chip select of the sensor. CS# is not a valid name in Verilog it is best to use the _N notation for active low signal naming.

An example controller state machine is described below but it is not a requirement to use it.

- An idle state to initialize the interface on power up.
- A wait state for 1 sec to control the sampling time
- A read light sensor state that triggers a read on the SPI interface
- A done state that waits for the SPI read operation to be completed before returning to idle.

Part 3: Display the sensor value on the 7-segment display

- Display the light sensor value on the 2 right most displays (display C and D).
- Display the last two digits of your WPI ID on displays A and B.

Part 4: Put it all together in top_lab3.v

In the top_lab3.v module you will instantiate and connect the following items

- Clock generation module with its input reset signal connected to btnC, its 10 MHz output for all sequential logic and a synchronous active low signal, reset_n or all sequential logic.
- Seven Segment Display module from Lab 2.
- Light sensor module(s) connected to PMOD JA pins.

It may be necessary to make some simple assign statements at the top level; this is OK but should be kept to a minimum. If you have implemented your design with multiple top level modules that is permitted but not required.

ALL port names to the FPGA pins must match the port names in the XDC file. An exception is that, you may change the XDC file names from JA[3:0] to JA0, JA1, JA2, J3 to allow the port directions for each pin to be different. Include only the ports you need, comment out or removed unused ports from your XDC file.

Extra Credit

Up to 5 lab bonus points for any good improvements, enhancements to your design. You must demo on board and describe the additional functionality well in your report. For example, make a system that can count objects passing through a light beam. If you do something really extraordinary additional points can be granted beyond the 5 points at the discretion of Prof. Stander.

Sign-Off Procedure

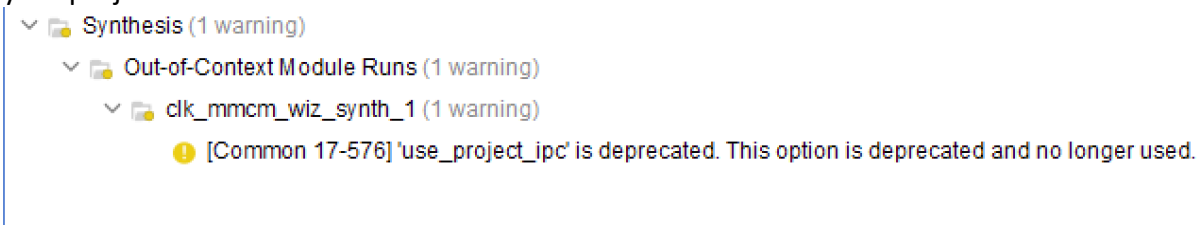
Signoff can be done remotely via zoom or in-person during lab hours. If via zoom, you will need to have the ability to share your screen with the TA and also have video capability to display the Basys-3 board functions clearly.

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 (both RTL and Simulation files) 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 (needs update)

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



Sign-Off Check List (40 points)

Project Status (5 points)

1. Expected files present in project (1 points)
   - Top level, clock generation, light sensor, seven segment display.
2. Vivado warnings (3 points)
   - Full credit no unexpected warnings.
3. Project programs part successfully(1 points)

Functionality Check List (32 points) / 8 points each item

1. WPI ID and Light Value displayed on seven segment display:
   - Last two digits of your WPI ID should appear on Display A and B.
   - Light sensor values should appear on Display C and D
2. Shine a bright light on sensor
   - Sensor should read near 8'hff
3. Cover Sensor with your finger:
   - Sensor should read near 8'h0.
4. Sensor reads intermediate value for different levels of coverage
   - Block light to various degrees to get in between values.

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 (3 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. (6 points)

- Clock generation module is implemented to specification.
- All sequential logic is driven by the 10 MHz clock signal and the synchronous active low reset_n signal.
- A new senor value is captured every second (1 Hz).
- A state machine is present to control sampling functions.
- SPI signals running at a 1 to 4 MHz rate.
- Sample rate parameter is declared in the sensor module port list.

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 (9 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 sequential logic driven with the posedge of the 10 MHz clock.
- All student generated logic has a reset state driven by a synchronous active low reset.
- State machine is coded in recommended coding style.
- There should be no latches in the design.
- Best Practices outlined in class were followed.

Rubric for Report (20 points)

Brief Introduction / Problem Statement (2 points)

General Overview of solution (10 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 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. A picture is worth a 1000 words.
    - Include a State Diagram and a detailed description of your control state machine(s).

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 and comment on whether this met your expectations or if it was different from expectation.
    - Explain your timing results.
        - Include your worst case negative slack and worst case total slack numbers.
        - What signal paths took the most time?
        - How might you improve the timing on these nets?
- If unexpected warnings occurred in your project. Explain the warnings and how they should be resolved.

Conclusion (3 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?