ECE 270 Lab Verification / Evaluation Form

Experiment 9

Evaluation:

IMPORTANT! You must complete this experiment during your scheduled lab period. All work for this experiment must be demonstrated to and verified by your lab instructor *before the end* of your scheduled lab period.

STEP	DESCRIPTION	MAX	SCORE
Step 1	Display "**GoPU**" on the 7-segment LEDs	2	
Step 2	Create two bounceless switches	2	
Step 3	Create a 4-bit binary UP counter	2	
Step 4	Demonstrate "bouncy" switch behavior	3	
Step 5	Convert the UP counter to an UP/DOWN counter	2	
Step 6	Utilize the on-chip oscillator and estimate its frequency	2	
Step 7	Divide the on-chip oscillator frequency in half	2	
Step 8	Display counter values in hexadecimal on 7-segment LEDs	3	
Step 9	Create a self-correcting 8-bit ring counter	3	
Step 10	Paste in state diagram code for light sequencer	2	
Step 11	Thought Questions	2	
	TOTAL	25	

Signature of Evaluator: _____

Academic Honesty Statement:

"In signing this statement, I hereby certify that the work on this experiment is my own and that I have not copied the work of any other student (past or present) while completing this experiment. I understand that if I fail to honor this agreement, I will receive a score of ZERO for this experiment and be subject to possible disciplinary action."

Printed Name:_____ Class No. __ __ __ __ - __

Signature:_____ Date: _____

Introduction to ispMACH 4256ZE Development Board

Instructional Objectives:

- To learn the basic features of a CPLD-based development board
- To learn how to realize state machines on a CPLD

Prelab Preparation:

- Read this document in its entirety
- Review the referenced Module 3 lecture material

See **Appendix A** of this document for instructions on how to program the CPLD board at your station using either **ispVM System** (integrated into ispLever) or **Diamond Programmer** (separate Lattice program).

Lecture/Demonstration:

Your lab instructor will give a brief presentation that includes the following:

- An overview of the ispMACH 4256ZE
- An overview of ispLEVER procedures for loading and testing compiled HDL programs

Experiment Description:

In this experiment, you will implement and test some basic Verilog modules that illustrate how the switches and LEDs on the ispMACH 4256ZE development board are interfaced to the CPLD. A skeleton file containing the I/O pin declarations ("top template") is provided on the course website.

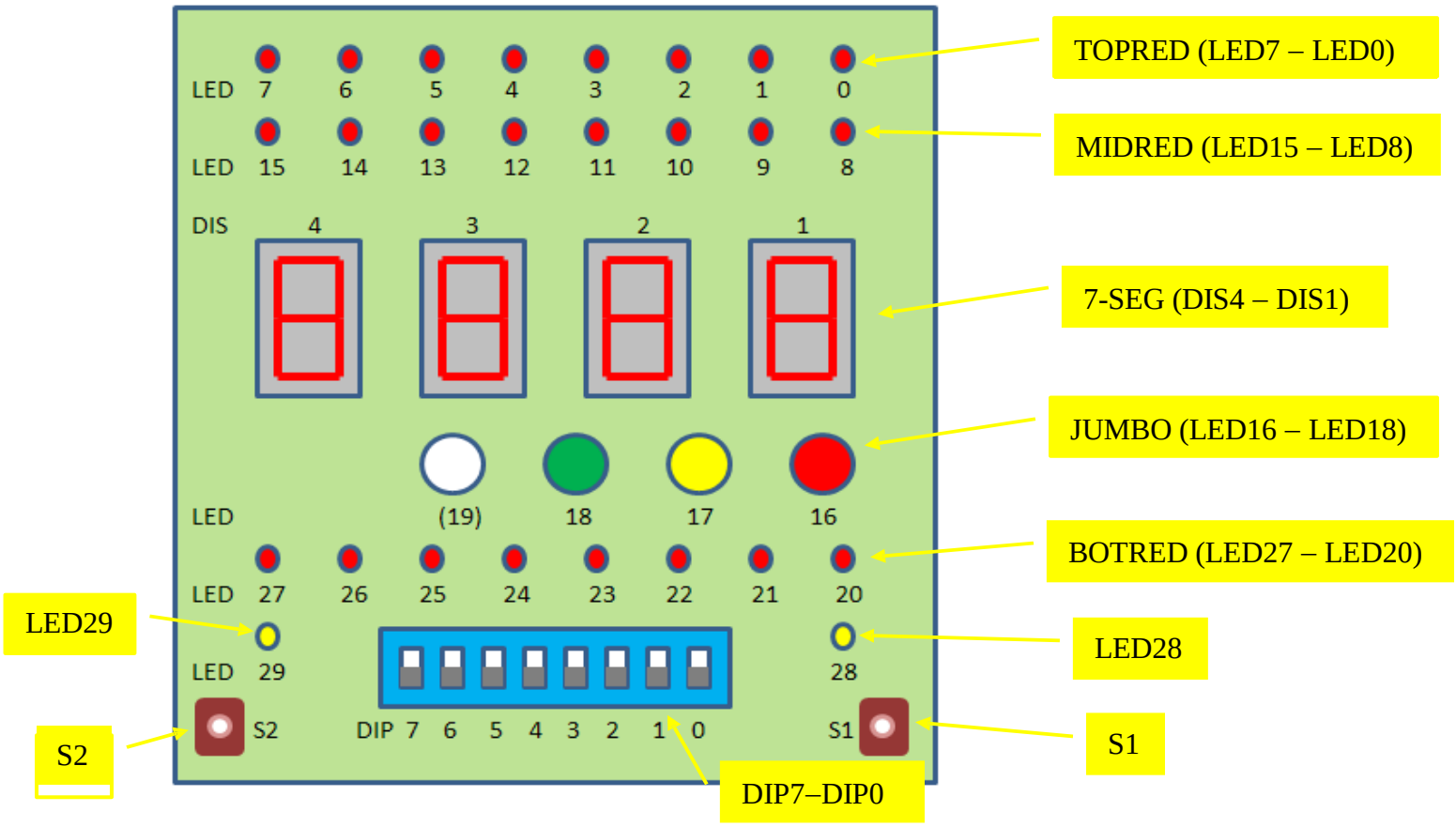


Figure 1. Lattice 4256ZE CPLD Development Board.

Step (1):

Route the DIP switches to the corresponding bottom row of red LEDs (BOTRED). When the leftmost DIP switch (DIP7) is "up", display "**GoPU**" on the four 7-segment displays. Note the alphanumeric display code definitions provided in the skeleton file.

Step (2):

Create two bounceless switches using S1 and S2 (the two SPDT momentary pushbutton switches), as outlined on page 13 of the Module 3 Lecture Summary notes. Route the output of each bounceless switch to the corresponding yellow LED next to it (LEDs 28 and 29, respectively). Each LED should be ON when the corresponding pushbutton is pressed.

Checkpoint: Demonstrate Steps 1 and 2 to your Lab Instructor.

Step (3):

Create a 4-bit binary UP counter and route its outputs to LEDs 0-3 (top row of red LEDs, right-hand side). Use the bounceless switch output from S1 (created in Step 2) to clock this counter and DIP0 to asynchronously reset this counter.

Step (4):

Create a second 4-bit binary UP counter and route its outputs to LEDs 4-7 (top row of red LEDs, left-hand side). Use the normally open ("NO") S1 switch contact (i.e., the one that is grounded when the button is pressed) to clock this counter and DIP0 to asynchronously reset this counter. After confirming the need for the switch contacts to be debounced (in order to provide a "clean" clocking signal), route the bounceless switch output from S1 to clock this counter. Confirm that both binary counters increment "in sync". Also, note what happens if the asynchronous reset (DIP0) is asserted while the counters are being clocked.

Checkpoint: Demonstrate Steps 3 and 4 to your Lab Instructor, including the "bouncy" switch behavior compared with the "debounced" behavior.

Step (5):

Route the bounceless switch output from S2 to the binary counter created in Step 4. Then, convert the 4-bit binary UP counter created in Step 3 into an UP/DOWN counter; use DIP6 to control the UP/DOWN mode (0 $\rightarrow$ DOWN, 1 $\rightarrow$ UP). Confirm that the counters created in Steps 3 and 4 can be clocked independently using the S1 and S2 bounceless switches, respectively (continue to use a common asynchronous reset supplied by DIP0).

Step (6):

Route the on-chip oscillator (`tmr_out`) to clock the UP/DOWN counter (replaces the bounceless switch S1 as the counter clocking signal), and route !DIP1 to the oscillator disable (`osc_dis`). Confirm that DIP1 enables/disables the on-chip oscillator (i.e., starts/stops the counter). Then, estimate the frequency of the `tmr_out` clocking signal by observing the counter output (note that this is the lowest possible frequency at which the on-chip oscillator can run). The on-chip oscillator setup is described on page 12 of the Module 3 Lecture Summary notes.

Checkpoint: Demonstrate Steps 5 and 6 to your Lab Instructor.

Step (7):

Divide the output of the on-chip oscillator in half using a T flip-flop (i.e., one-bit counter), as shown on page 12 of the Module 3 Lecture Summary notes. Use the output of this flip-flop in place of `tmr_out` as the clocking signal for the UP/DOWN counter. Estimate the frequency of this "divided" clocking signal by observing the counter output.

Step (8):

Route the S1 bounceless switch output (previously disconnected) back to the UP/DOWN counter clocking source. Display the UP/DOWN counter value as a (single) hexadecimal digit on 7-segment display DIS1 (the rightmost 7-segment LED), and display the second (4-bit binary) UP counter value as a (single) hexadecimal digit on 7-segment display DIS4 (the leftmost 7-segment LED). Use DIP7 to select whether "**GoPU**" is displayed on DIS4..DIS1 (DIP7=1) or the hexadecimal counter values are displayed on DIS1 and DIS4 (DIP7=0).

Checkpoint: Demonstrate Steps 7 and 8 to your Lab Instructor.

Step (9):

Create a self-correcting 8-bit ring counter and route its outputs to the middle row of red LEDs (MIDRED), as shown on page 20 of the Module 3 Lecture Summary notes. Use the "divided" oscillator output to clock the ring counter.

Step (10):

Include the Verilog module for the "Light Sequencer Using State Diagram" (`moorelsa_sd.v`) 4-mode light sequencer (provided on the course website) and route the output variables to the "jumbo" (green/yellow/red) LEDs (*note that some modifications to the declarations will be required*). Use the "divided" clocking signal (from Step 7) as the clocking source, and use DIP0 to provide an asynchronous reset to the state machine. Use DIP3 and DIP2 to provide the "mode control" inputs M1 and M0. Confirm that the light sequencer works as anticipated, and that all four state machines (UP/DOWN counter clocked using S1, UP counter clocked using S2, ring counter clocked using internal oscillator, and light sequencer clocked using internal oscillator) coexist "peacefully" and operate independently.

Checkpoint: Demonstrate Steps 9 and 10 to your Lab Instructor.

Step (11): Write your answers to the following Thought Questions in the space provided**.**

1. Why does the asynchronous reset signal *not* need to be debounced?

   _____

   _____

2. What change would you make if you wanted to clock your state machine(s) even slower than the "divided" clock (derived from the internal oscillator) utilized in this experiment?

   _____

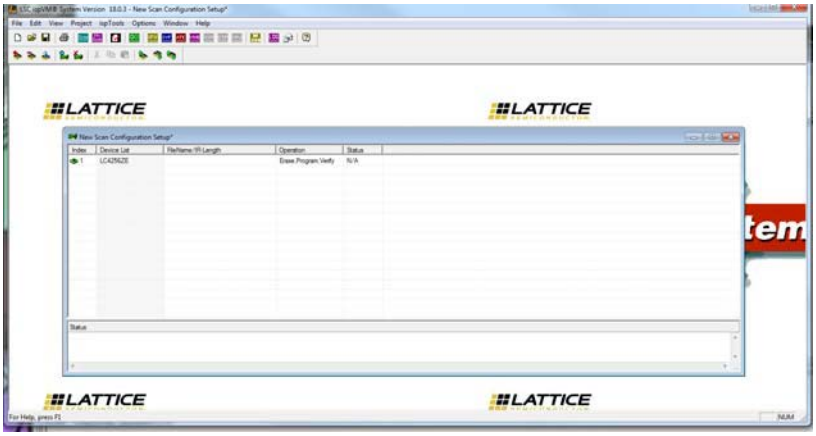
   _____

Appendix A: Programming the CPLD Board

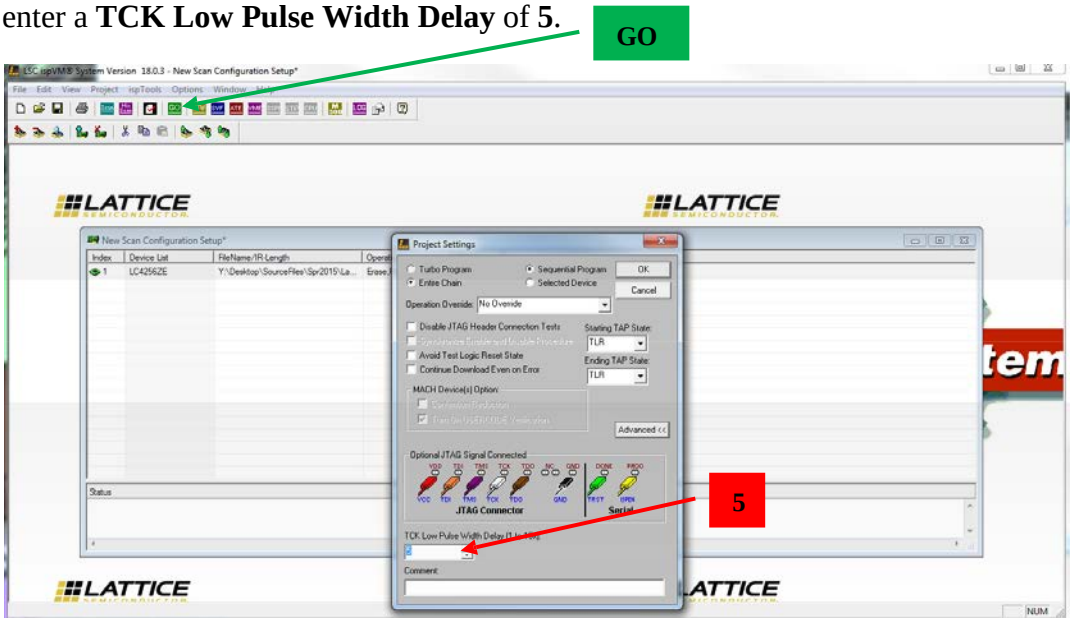
There are two different tools that can be used to program the CPLD board: **ispVM System** (tool within ispLever) and **Diamond Programmer** (separate Lattice program).

A. Using ispVM System

1. From the **ispLever** toolbar, select **Tools → ispVM System**.
2. Click on the **Scan** icon; **ispVM System** should find the CPLD.



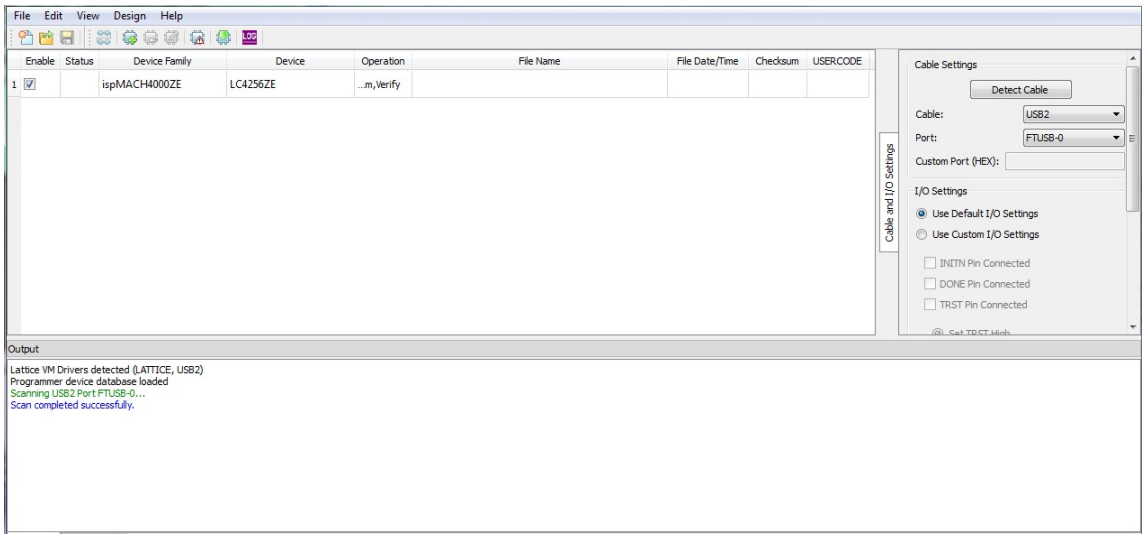
3. Click on the first line of the **New Scan Configuration Setup** box (that lists the LC4256ZE) to bring up the **Device Information** dialog box; browse for the JEDEC **(.jed)** file created by **ispLever** when you synthesized your Verilog module. Upon clicking OK, the filename will be filled in.
4. On the **ispVM System** toolbar, select **Project → Project Settings**; in the **Project Settings** dialog box, click on **Advanced**. In the extension to this dialog box that appears, enter a **TCK Low Pulse Width Delay** of **5**.



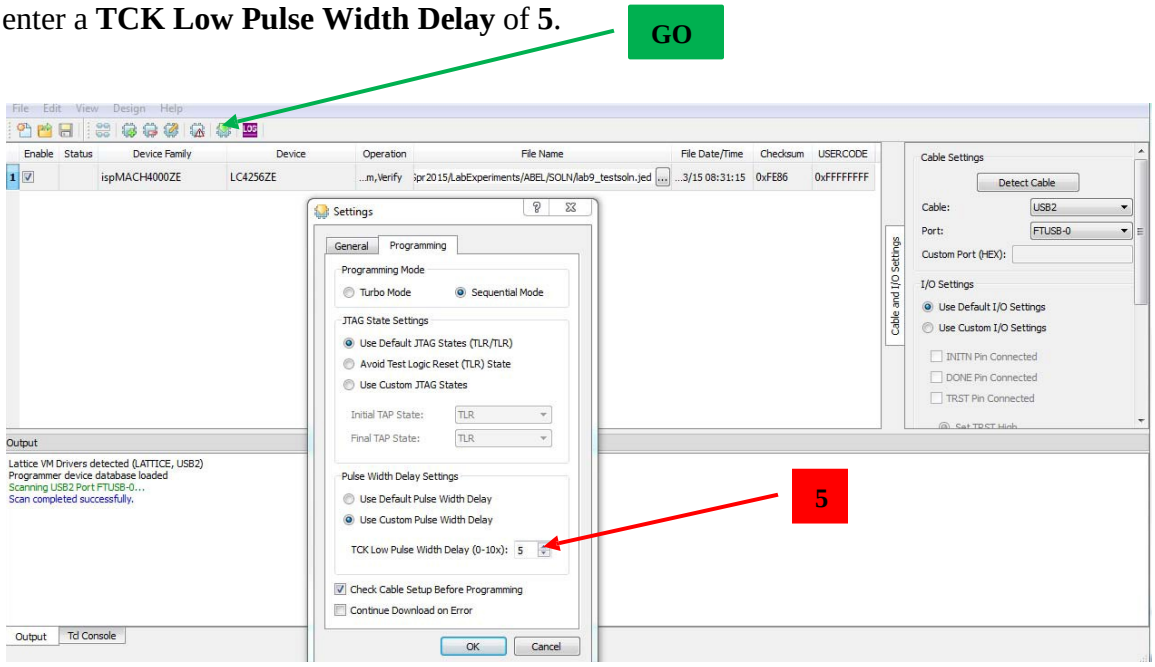
5. After clicking OK, you are now ready to program the CPLD, which can be accomplished by clicking on the **GO** icon (shown above).

B. Using Lattice Diamond Programmer

1. From the **Windows Program** menu, select **Lattice Diamond Programmer.**
2. On the **Getting Started** dialog box, select **Create a new Project from a Scan** then click on the **Detect Cable** button; after scanning, Diamond Programmer should find the CPLD board at your station.



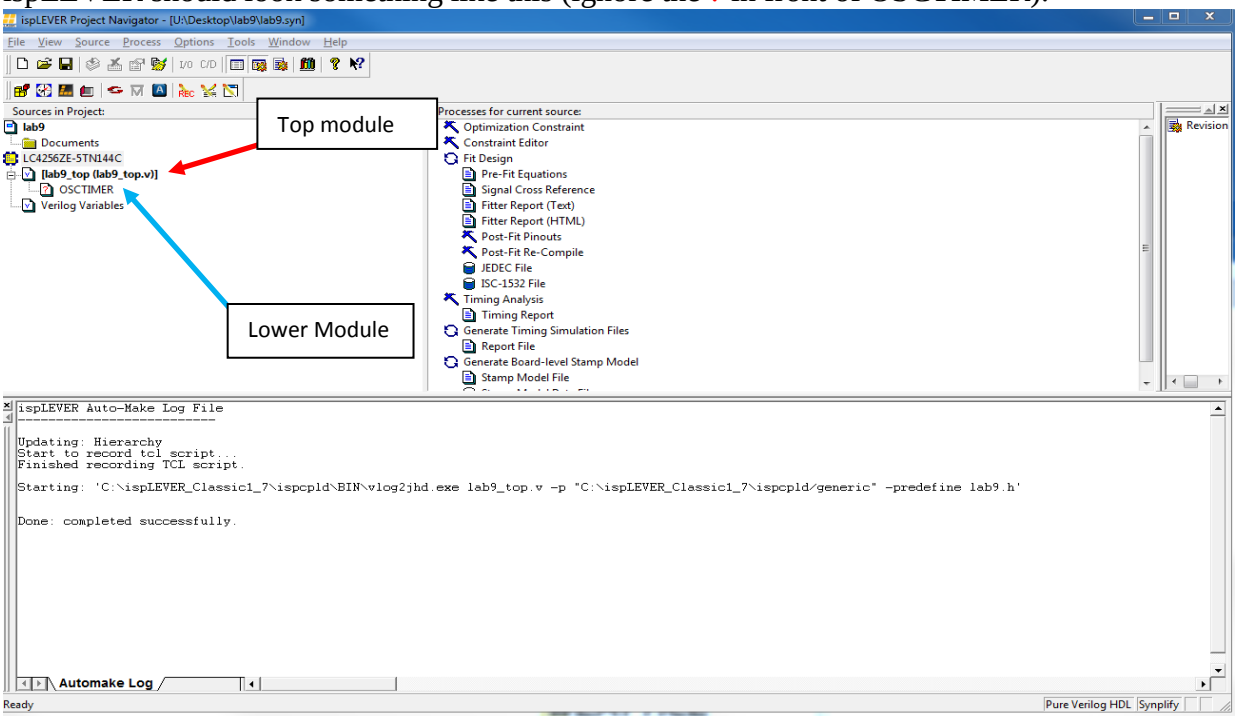
3. Next, click on the box under the **File Name** heading and browse for the JEDEC **(.jed)** file created by **isp_Lever** when you synthesized your Verilog module.
4. Next, go to **Edit → Settings** and select the **Programming** tab; under the heading **Pulse Width Delay Settings**, click on the **Use Custom Pulse Width Delay** radio button and enter a **TCK Low Pulse Width Delay** of **5**.



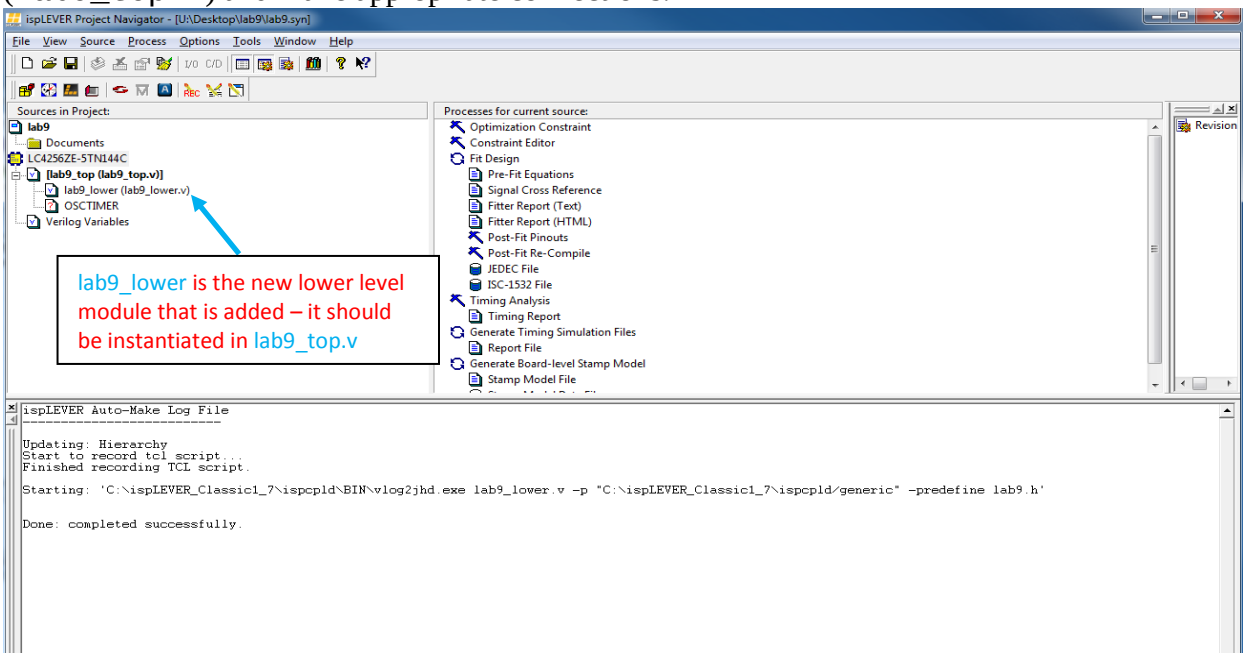
5. After clicking OK, you are now ready to program the CPLD, which can be accomplished by clicking on the **GO** icon (shown above).

C. Hierarchical Design and Adding Multiple Verilog Modules

1. The best way to describe hardware is by building your TOP level module with smaller lower level modules. This is referred to as *hierarchical design*.
2. Your Verilog description can have multiple lower level modules but should have only one TOP level module. All lower modules should be instantiated in the TOP module.
3. Modules and use of hierarchical design help in keeping your code organized and concise.
4. When you add a new (TOP) module and paste in the provided skeleton file, your ispLEVER should look something like this (ignore the ? in front of OSCTIMER).



5. Next, to add a lower level module, right click on **LC4256ZE-5TN144C → New → Verilog Module → OK.** Then, you should instantiate this module in the top module (`lab9_top.v`) and make appropriate connections.



6. You can add multiple lower level modules following the above steps.

