The UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Comp 541 Digital Logic and Computer Design
Fall 2014

Lab #5: Working with the boards!
Issued Wed 9/17/14; Due Wed 9/24/14 (submit by 11:59pm)

This lab introduces you the new hardware development kits that you will be using for the remainder of the semester. The lab assignment consists of three steps, each building upon the previous. This lab assignment will be explained in detail during the lab session. But, I am providing written instructions here, along with some screenshots, especially since this is the first assignment using the hardware kits.

You will learn the following:

- Safe handling of the boards, powering them up, connecting to your computers
- Designing encoders/decoders (to drive a 7-segment display)
- Designing timing references using counters
- Specifying pin mappings between the I/O ports of your design and the board pins
- Downloading circuit implementation onto the board, and running it

---

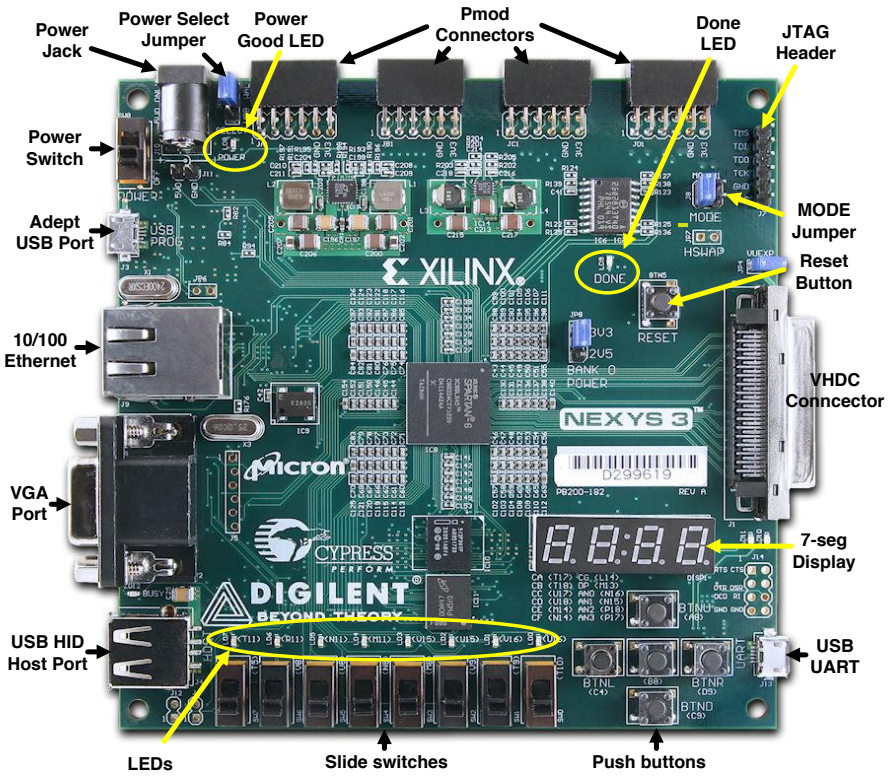
Part 0: Read the Manual!

Download the manual for the kits from the class website, and go through the following sections carefully:

- Overview
- Power Supplies
- Oscillators/Clocks
- Basic I/O (for Nexys 4, skip Tri-Color LEDs)

Follow these guidelines when using the kits:

- *Ground yourself before touching the kit!* And I don't mean that in a metaphysical sense. Please touch a metal faucet, an all-metal door (e.g., fire door), or the outside of a ground appliance (refrigerator, microwave oven). This step is all the more important during winter, especially if you wear rubber soles and walk on synthetic rugs or tiles. A static shock can send 2,000-4,000 volts to your circuit board and ruin it. And we don't have any extras.
- Set the power supply jumper on the board to derive power from the USB cable. (In the pictures on the next two pages, the jumper is on the top left of the board, near the "POWER" switch. It is labeled JP1 on Nexys 3 boards, and JP3 on Nexys 4 boards.) The jumper has a blue plastic cap. If the cap is not already on the "USB" setting, move it from the "Wall" setting to the "USB" setting.
- Now connect the board to your computer using the USB cable provided. The smaller end of the cable should go into the micro-USB programming socket near the power switch, labeled "USB PROG" on Nexys3 boards, and "PROG UART" on Nexys 4 boards). The larger end of the cable goes into a USB port on your computer.
- Turn the "POWER" switch (on the top left of the board) on. If you see the board light up and pass the self test, everything is good.



Nexys 3 Board

If you have a Nexys 3 board, study the above picture carefully. If you have a Nexys 4 board, study the picture on the next page.

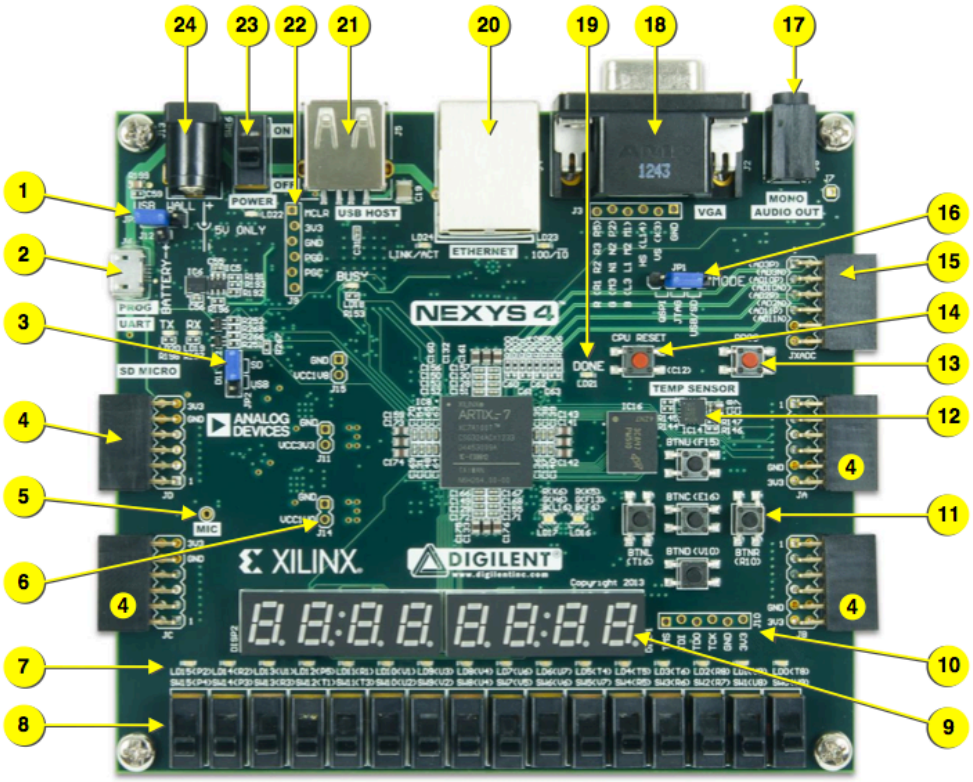


Figure 1. Nexys4 board features

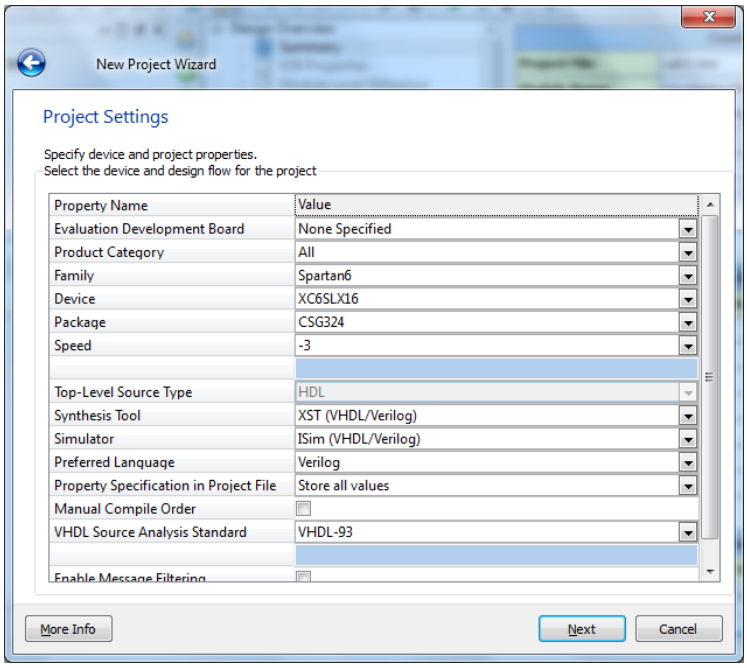
Callout	Component Description	Callout	Component Description
1	Power select jumper and battery header	13	FPGA configuration reset button
2	Shared UART/ JTAG USB port	14	CPU reset button (for soft cores)
3	External configuration jumper (SD / USB)	15	Analog signal Pmod connector (XADC)
4	Pmod connector(s)	16	Programming mode jumper
5	Microphone	17	Audio connector
6	Power supply test point(s)	18	VGA connector
7	LEDs (16)	19	FPGA programming done LED
8	Slide switches	20	Ethernet connector
9	Eight digit 7-seg display	21	USB host connector
10	JTAG port for (optional) external cable	22	PIC24 programming port (factory use)
11	Five pushbuttons	23	Power switch
12	Temperature sensor	24	Power jack

Nexys 4 Board

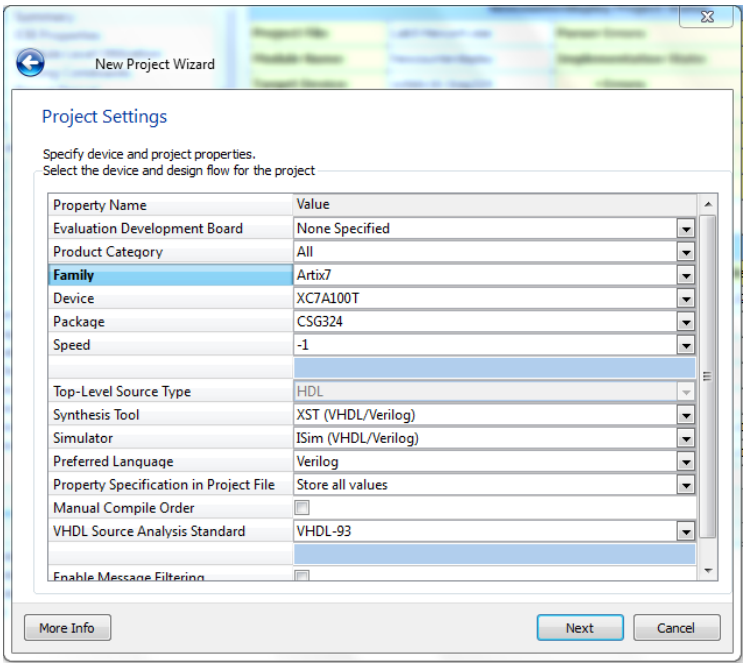
Part I: A 7-Segment Display Encoder (Decimal digit → Display character)

Let us begin by designing a simpler encoder to convert a single decimal digit (4-bit value) to a bit pattern suitable for driving one letter of a 7-segment display.

Create a new project called Lab5. Be sure to use the proper settings for the kits we are now using, by setting the *Family, Device*, *Package* and *Speed* parameters exactly as in the picture below:



Nexys 3 Project Settings



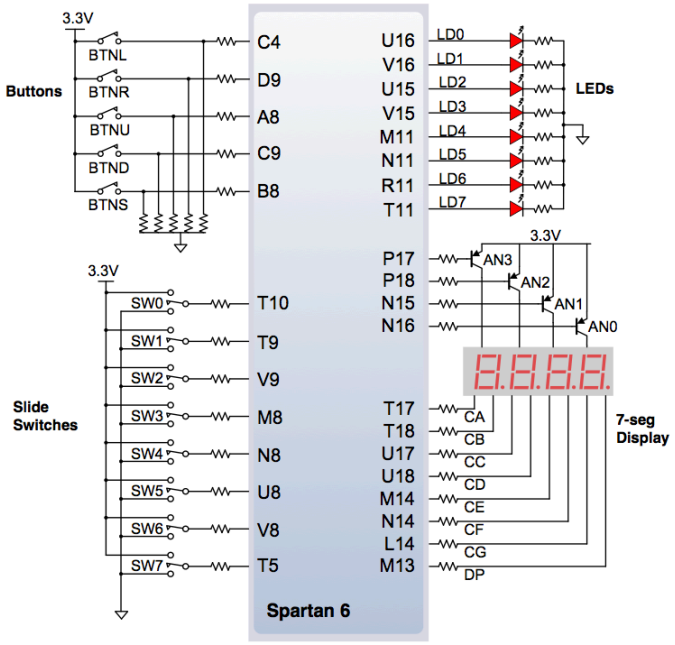
Nexys 4 Project Settings

<u>Important:</u> The *Family, Device,* and *Speed* parameters depend upon whether you are using a Nexys 3 board or a Nexys 4 board.

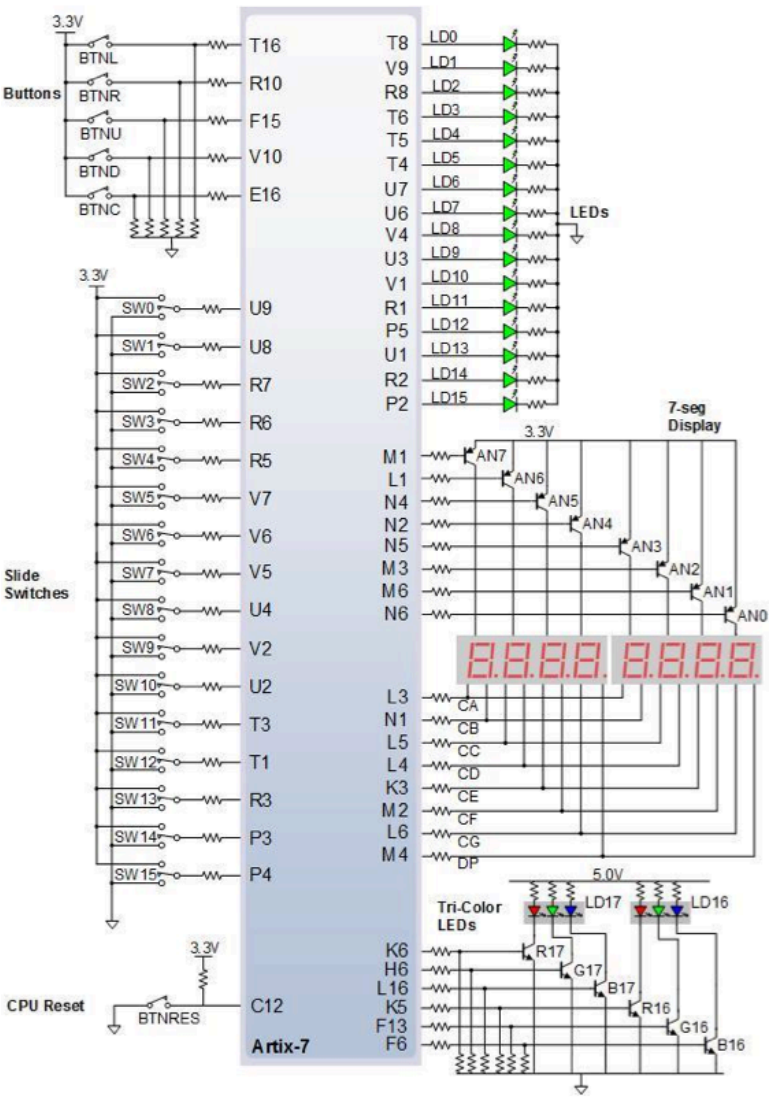
Download the Verilog file **dec7seg.v** from the class website. Add it to your project by clicking the "Add Copy of Source" button near the top left (see picture on the right) and then selecting the file you downloaded. Understand that this button makes a *copy* of the file you select, and places it into the project folder. If you instead use "Add Source", that leaves the file where it is, and adds it to the project. In my opinion, it is always better to use "Add Copy of Source", especially if you are copying a file from an earlier lab; otherwise, any changes you make to the file will be reflected in the earlier lab as well.

Depending on the board, you either have a 4-digit display or an 8-digit one. In this lab, we will be using only a single digit (the rightmost one). But the remaining ones (three or seven) need to be kept unlit. Therefore, one line in the Verilog description will be different depending on which board you are using. Please follow the comments in the file and use the appropriate line.

Go through the Basic I/O section of the manual very carefully to understand how the slide switches and the 7-segment display work. The schematic is repeated here for convenience.



Nexys 3: Switches and LEDs



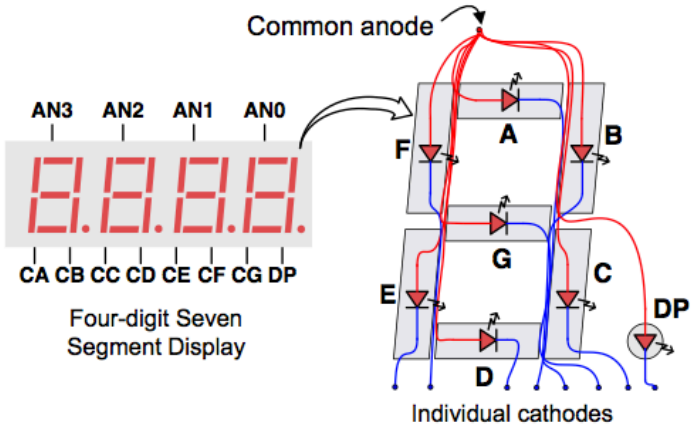
Nexys 4: Switches and LEDs

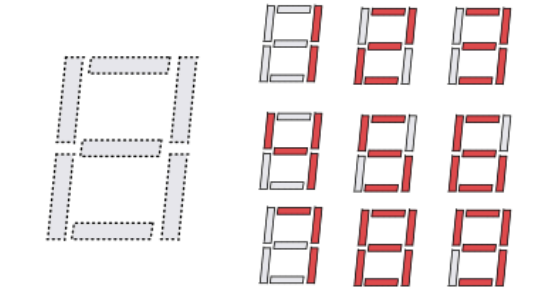
Now we need to specify how the inputs/outputs of your Verilog module map to the actual I/O pins on the board.

- We want to use the 4 rightmost slider switches for the 4-bit input to your module, A[3:0]. These would be pins {M8, V9, T9, T10} on the Nexys 3 board and pins {R6, R7, U8, U9} on the Nexys 4 boards (see the relevant figure above, and also find these labels next to the switches on the board). In this mapping, M8/R6 is the MSB and T10/U9 is the LSB.
- We want to connect the 8-bit display output of the Verilog module, `segments[7:0]`, to the following eight pins: T17-M13 on Nexys 3 and L3-M4 on Nexys 4.
- Finally, we want to connect the digit select output of the Verilog module, `digitselect[]`, to the corresponding selection outputs on the board ("anode selects"). For Nexys 3, which has a 4-digit display, connect `digitselect[3:0]` to P17-N16. For Nexys 4, which has an 8-digit display, connect `digitselect[7:0]` to M1-N6.

These connections are specified in a "User Constraints File" (sometimes called an "Implementation Constraints File" in some dialog boxes). Do the following: Click on "New Source" → select "Implementation

Constraints File", and give it the name `segdisplay.ucf`. An empty file will be created and added to the project. Cut and paste the contents of the file by the same name provided on the class website. (Alternatively, you could have downloaded the file from the website, and used the "Add Copy of Source" command.)





Carefully go through the Verilog source and the reference manual, and be sure you understand all of the following:

- What does `digitselect[]` do? Why is it 4 bits long for Nexys 3 but 8 bits for Nexys 4?
- Why is there a negation in front of the pattern assigned to `digitselect[]`?
- Look at the patterns used for lighting up a decimal digit in the picture above, and verify that the Verilog module does the same.
- Why is there a negation in front of the pattern assigned to `segments[7:0]`?

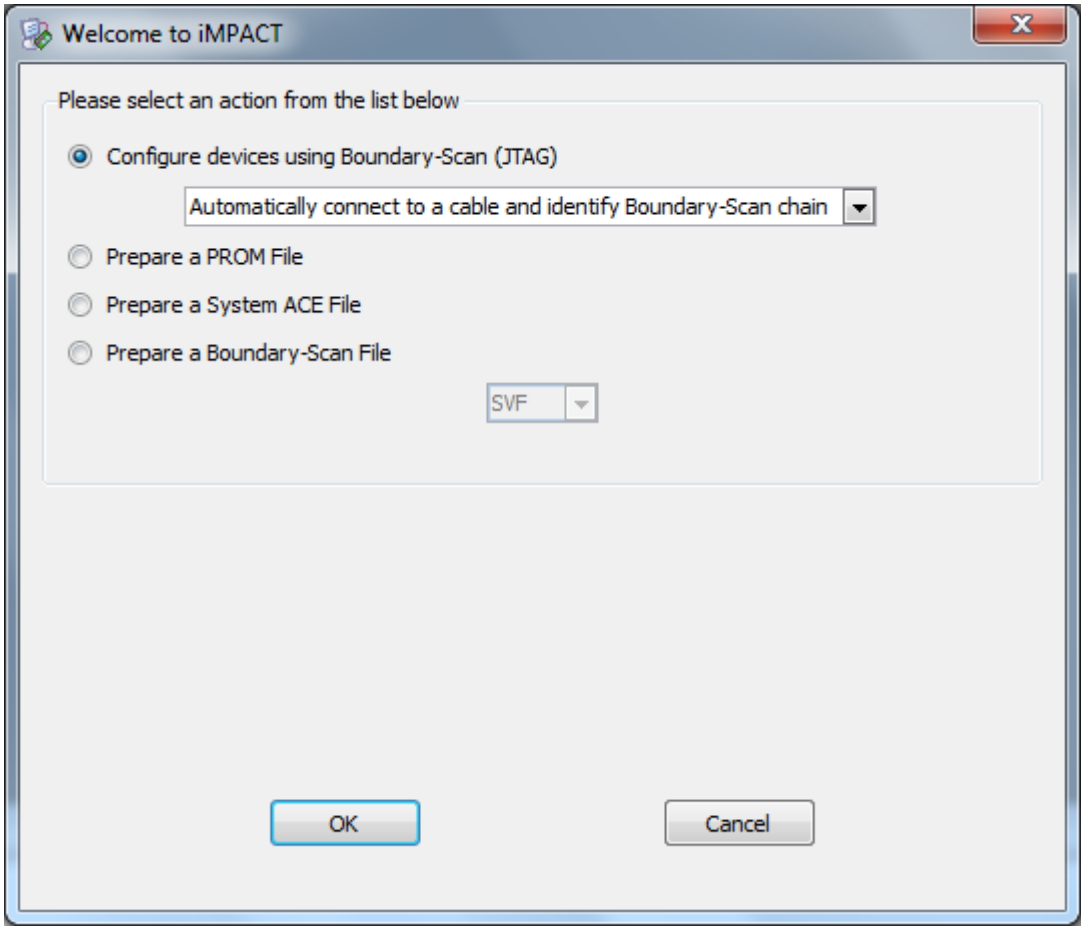
(As you must have gathered, there is only one pattern that can be output at a given time. To show a multi-digit number, each of the digits is displayed on the corresponding position on the display for a brief time, before moving on to the next digit. By cycling through all the digits are a high speed, we see the illusion of an multi-digit display! This is the task for the next lab!)

Now let us compile and get it ready for the board. Be sure Implementation view is selected at the top. Click *Synthesize-XST, Implement Design* and *Generate Programming File* one after the other in sequence. Or, simply click *Generate Programming File,* and the tool will automatically perform all the steps in sequence. Once successfully completed, this step generates a ".bit" file in your project folder (**decto7seg.bit**). Look for it.

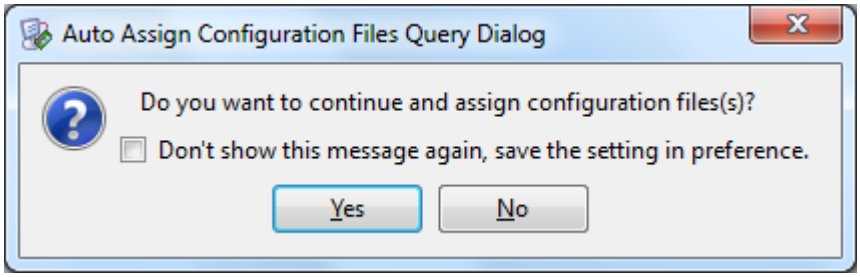
TIP: Sometimes, when things go terribly wrong, you may have to run each of these three steps separately. Sometimes, you may have to force a complete rebuild by right-clicking and selecting *ReRun All.*

Now let us download the implementation onto the board. This step is also called "programming the board." First, connect the board to your computer according to instructions provided in Part 0. The tool we will be using for this step is called *iMPACT.* You can launch it by clicking Tools→ iMPACT, or by clicking Configure Target Device. The tool ISE iMPACT opens in a new window. Run the wizard by clicking Edit → Launch Wizard.

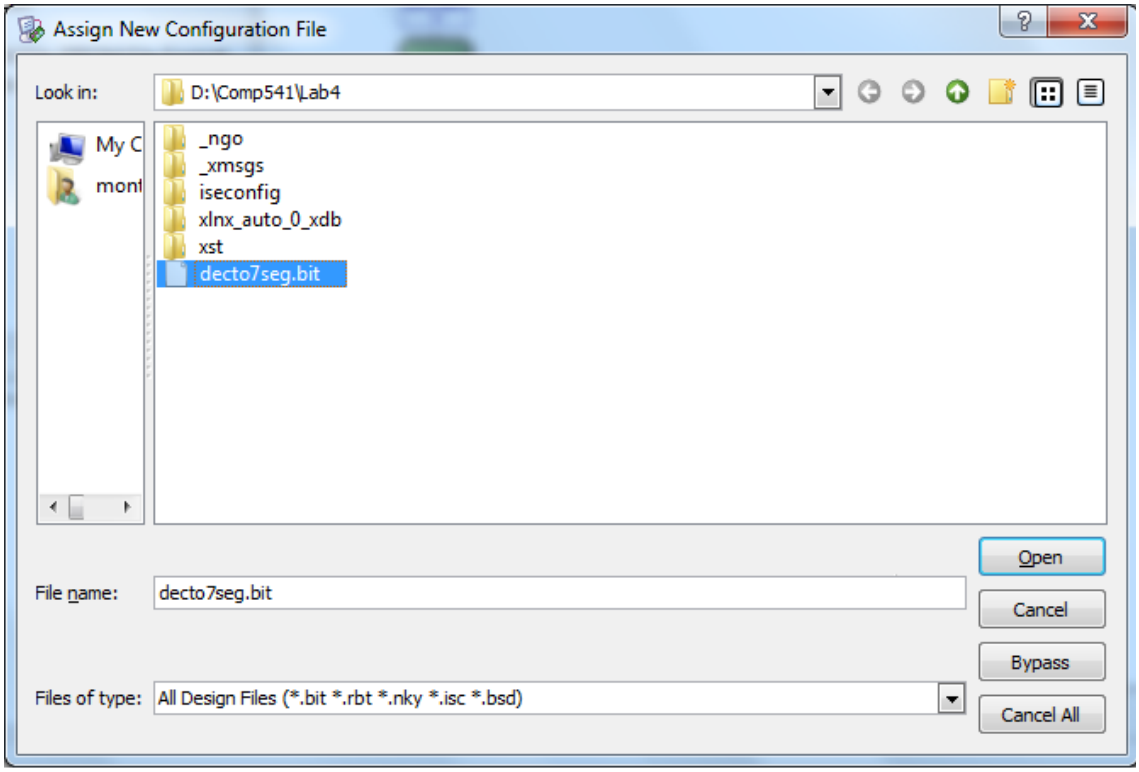
The following screenshots show you the sequence of dialog boxes that pop up. Make the selections exactly as shown.



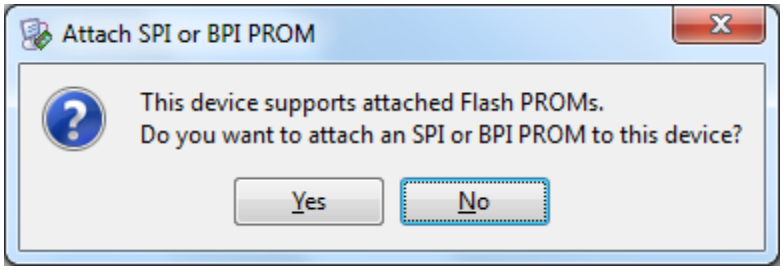
Click OK.



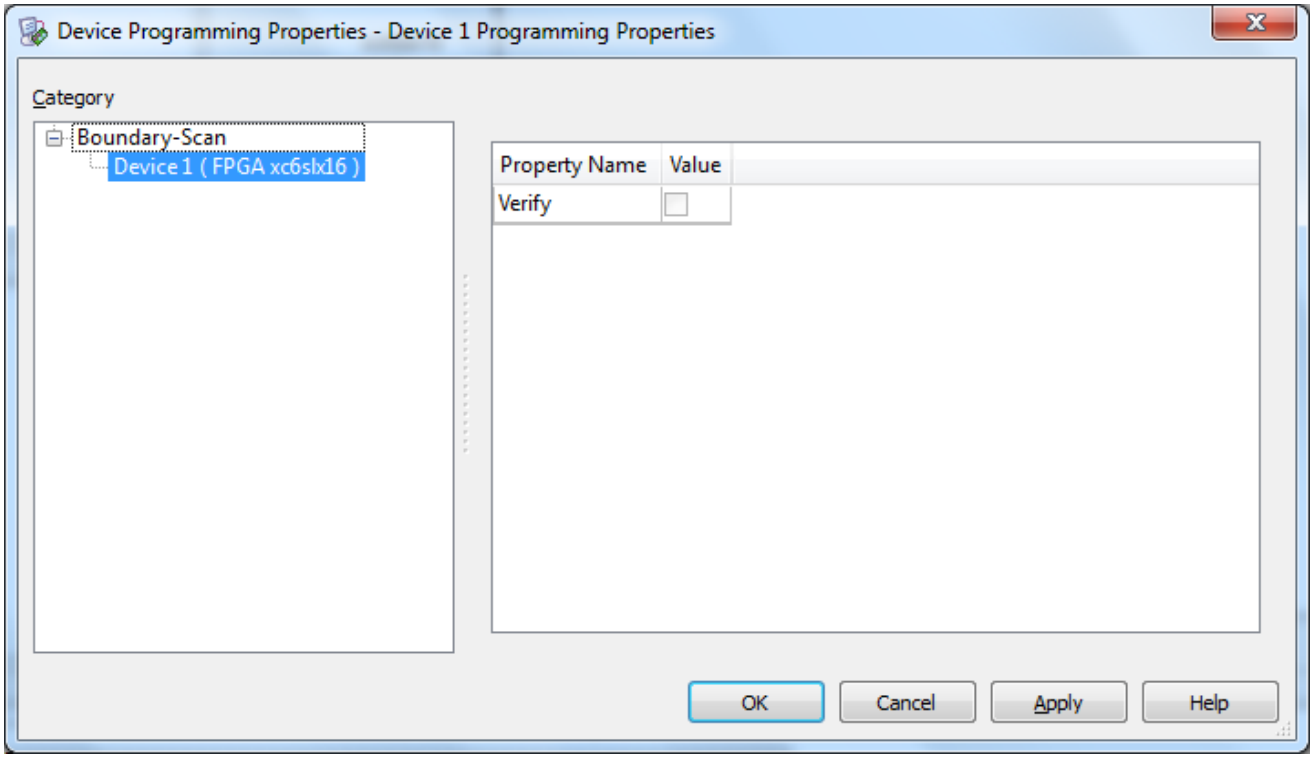
Click Yes.



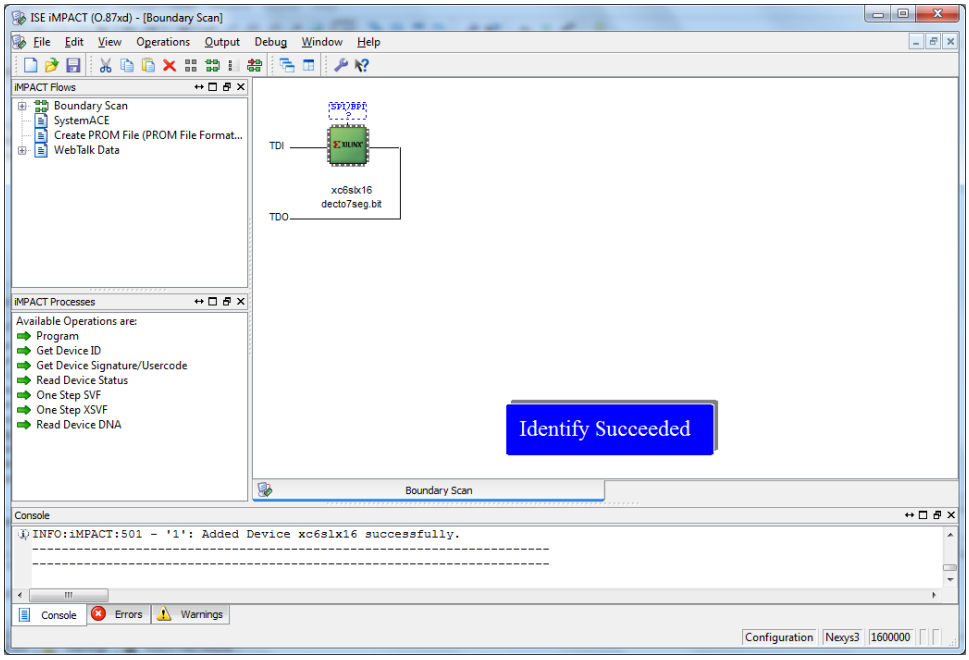
Select the ".bit" file that you generated (**decto7seg.bit**).



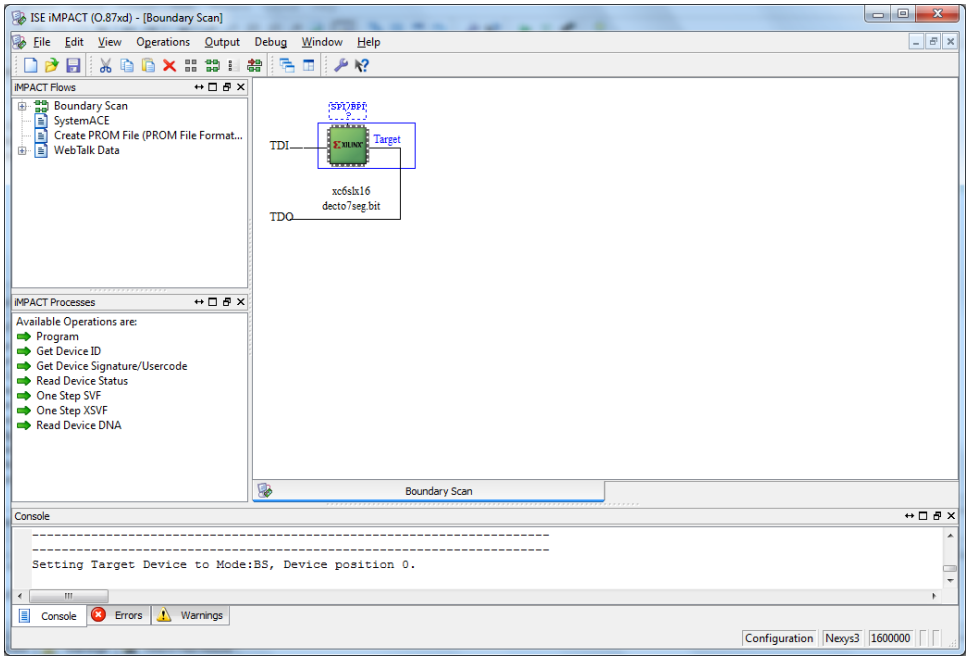
Select No (we are not working with flash memories etc., at least not yet).



Click OK. If all went well so far, you will see an "Identify Succeeded" message:



Next, right-click on the green Xilinx box (this is the chip in the middle of the board onto which you are "burning" your compiled design), and click *Select Target Device.* You should see this:



Finally, click Program (either in the middle left of the window, or right-click Xilinx and select Program). If all went well, your design has now been implemented onto the circuit board, and is now running! Play with the slider switches and verify the output is as expected.

Part II: A Hex Display Encoder (Hex digit → Display character)

Make a copy of `dec7seg.v` from Part I and give it the name `hexto7seg.v`. Remove `dec7seg.v` from your project (right-click → *Remove*); this will not delete the file, only remove it from your project. Use *Add Source* to add the new `hexto7seg.v` file into your project. Modify the display encoder to handle a hexadecimal digit (i.e., "0" to "F"). Also, change the module's name to `hexto7seg()`. You will simply need to add six additional lines of code to your Verilog module to handle the cases "A" through "F". It does not matter whether you choose to display certain letters in lowercase or uppercase (e.g., "a" vs. "A"). Rerun the tool chain, generate the programming file (`hexto7seg.bit`), and download it to the board and confirm that you can now display '0' through 'F'.

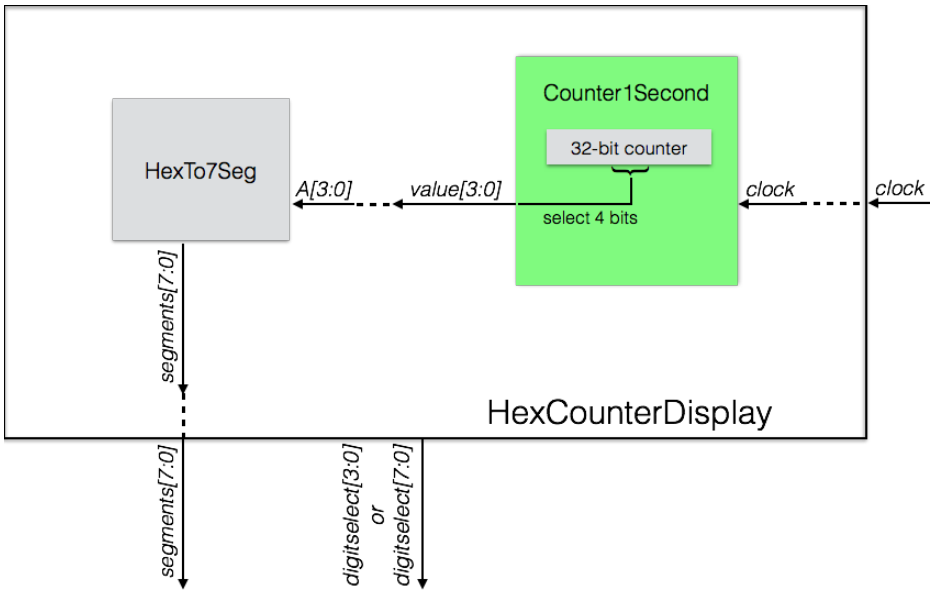
Part III: Displaying a single-digit hex counter

Modify the design of Part II so that the value to be displayed is not input via the slide switches. Instead, create a counter that cycles through the values "0"—"F" repeatedly, and feed that value to your hex display encoder from Part II. For you to be able to observe if this design is working properly, the counter must be running at a reasonable speed! Make it count at a rate of close to once per second. Note the following:

- Your counter will need a clock. The board provides a 100 MHz clock on pin V10 for Nexys 3 boards, and pin E3 for Nexys 4 boards (see Oscillators/Clocks in the manual). Add this to your .ucf file (simply uncomment the relevant line).
- Make a counter that is 32 bits long, even though we are displaying only four bits. The lowermost bits are clearly changing too rapidly. (The LSB flips 100 million times a second!) The uppermost bits are changing too slowly. You need to find 4 consecutive bits somewhere in the middle so that the least significant of those is changing approximately once per second. These 4 bits form a hexadecimal number (from 0 to 15) that is to be displayed. TIP: You will not get a frequency of exactly once per second, but anything between half second and two seconds is okay.

- You will need to modify the .ucf file to remove the inputs from the slide switches, since they are no longer used (simply comment them out).
- Be sure your design is modular. That is, the top-level module should internally have two modules (a counter, and a hex display encoder). Accordingly, there should be three distinct Verilog files:
    1. **hexto7seg.v** (your hex to 7-segment display encoder from Part II). Observe that it should *no longer produce* `digitselect[]`, which should instead be produced by the top-level module. Simply comment out the appropriate line from the module's header.
    2. **counter1second.v** (your counter module with a 4-bit output that changes roughly once per second)
    3. **hexcounterdisplay.v** (the top-level module), which should contain one instance of the module from hexto7seg.v, and one instance of the module from counter1second.v. This top-level module should now have the output `digitselect[]`.

    Please use the filenames specified.
- The following block diagram shows the hierarchy that your design must follow. **You will not receive full credit if your design does not follow the modular construction specified in this figure.**



What to submit:

- **The three Verilog sources (hexto7seg.v, counter1second.v, and hexcounterdisplay.v).**
- **Show a working demo of your design for Part III during the next lab hours.**

How to submit: **Please submit your work by email as follows:**

- **Send email to:** `comp541submit-cs@cs.unc.edu`
- **Use subject line: Lab 5**
- **Include the three attachments as specified above**
