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

Comp 541 Digital Logic and Computer Design
Prof. Montek Singh
Fall 2021

Lab #3B: Working with the boards!
Issued Wed 9/1/21; Due Wed 9/8/21 (11:55pm, via Sakai)

This lab introduces you to the 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!

There are two versions of our development kits: some are the "Nexys 4" model, and the rest are the two newer models, called "Nexys 4 DDR" or "Nexys A7". All models have identical capabilities for everything we will be doing this semester.

Download the appropriate manual from the class website (under *Reference Materials* in the right column) for the kit you have received, and go through the following sections carefully:

- Overview
- Power Supplies
- Oscillators/Clocks
- Basic I/O (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 grounded 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 picture on the next page, the jumper is on the top left of the board, near the "POWER" switch, labeled JP3) 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.
- Next, you will power up the board. Read the following important note first, and then use the instructions that follow.

- **IMPORTANT:** The power connector on the board is rather fragile. Repeated insertion and removal of the USB power cable into the side of the board can make the power connector socket loose, rendering the board unable to be powered up. This is the most common cause of board failure. You are strongly advised to **leave the USB cable plugged into the board whenever possible,** unplugging it only from the laptop side to disconnect the board. If you must occasionally remove the cable from the board (e.g., while transporting the kit), please exercise extreme care in doing so: hold the tiny socket with the fingers of one hand while you gently extract or insert the cable with the other hand. That is, please do not simply hold the board while you plug the cable in; that might put a lot of shear force between the socket and board, eventually causing the socket to come loose.

- Now carefully connect the board to your computer using the USB cable provided. Identify the micro-USB programming socket near the power switch, labeled "PROG UART". Hold this socket firmly with the fingers of one hand, and use the other hand to gently insert the smaller end of the USB cable provided with the kit. The larger end of the cable goes into a USB port on your computer.
- Make sure of one more jumper: JP1 (numbered 16 in the picture on the next page) should be sitting on the two leftmost pins, indicating that the FPGA chip upon power-up will be configured with a power-on-self-test configuration stored in a flash memory on the board. When you power up the board, you will see it go through a sequence that tests various color LEDs, the segmented display, and more (if you connect a monitor to the monitor port, you will see these tests, but you don't have to right now).
- Make sure the board is sitting on a level horizontal surface, and then 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. (If the board was not horizontal, its built-in accelerometer will likely cause the self-test error "Fail 2".)

The Boards.

Study the pictures below of the boards carefully. Some of our boards are the "Nexys 4" model, and the rest are the newer "Nexys 4 DDR" or "Nexys A7" models. All of the boards are nearly identical in functionality, except for the type of external memory on the board (which we don't use); the latter two types have DDR memory. However, connections between the FPGA chip to all the I/O (i.e., buttons, switches, LEDs, etc.) are numbered differently, so you should be sure to use the correct pin numbers. The course website will provide two versions of I/O configuration files with pin numbers: one for the Nexys 4 model, and another for the Nexys 4 DDR and Nexys A7 models (the latter two have identical pin numbering).

Nexys 4

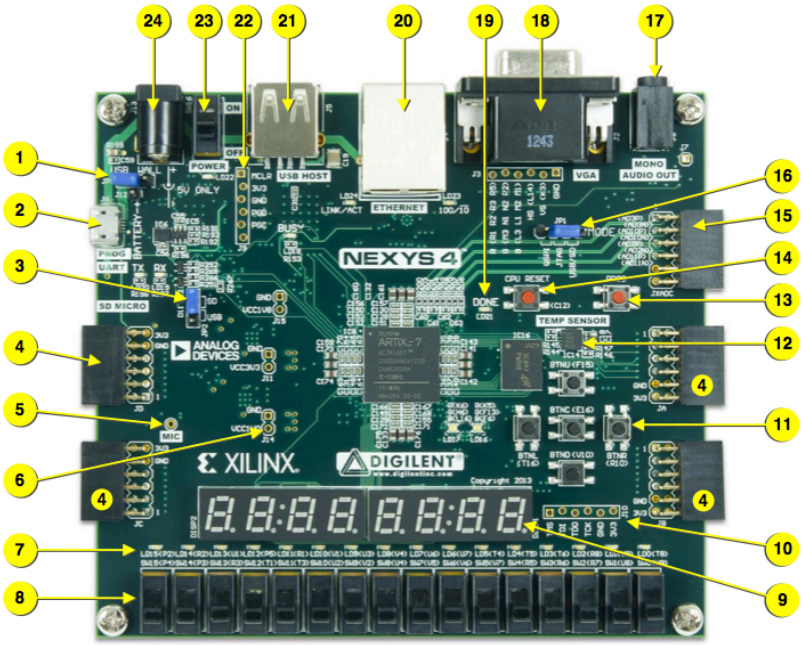


Figure 1. Nexys4 board features

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 DDR or Nexys A7

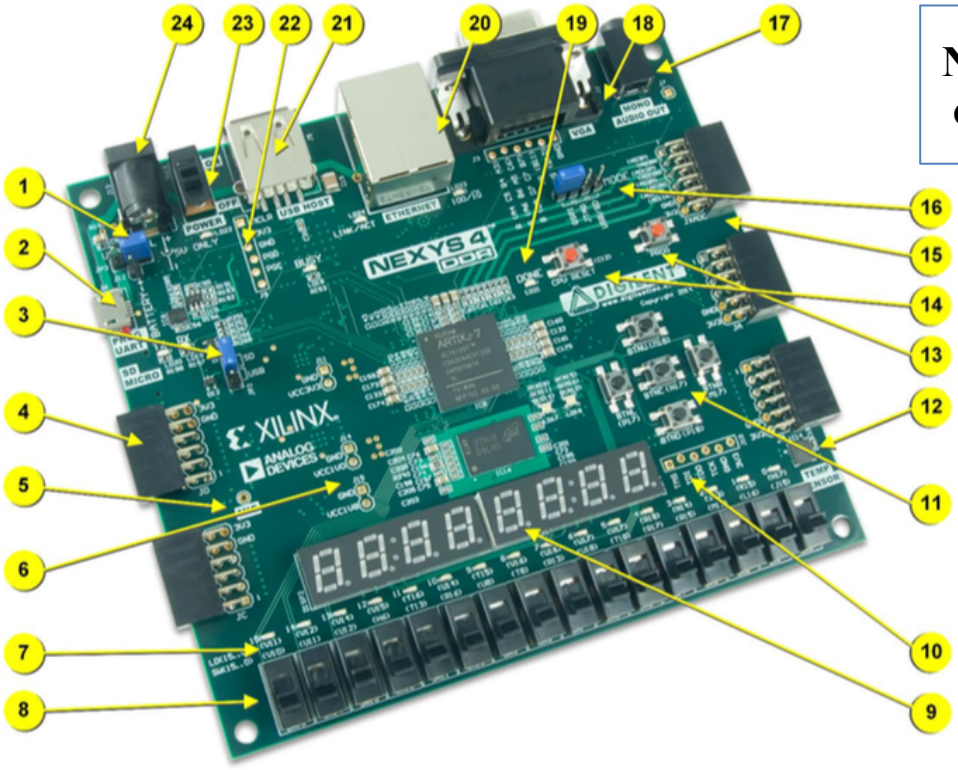


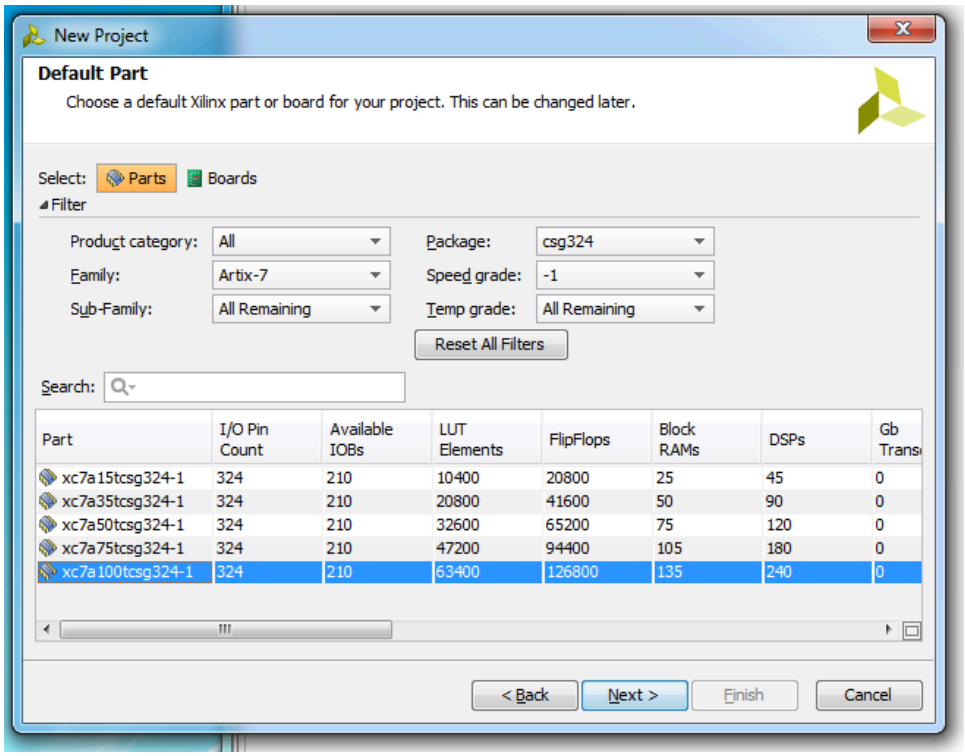
Figure 1. Nexys4 DDR board features.

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

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.

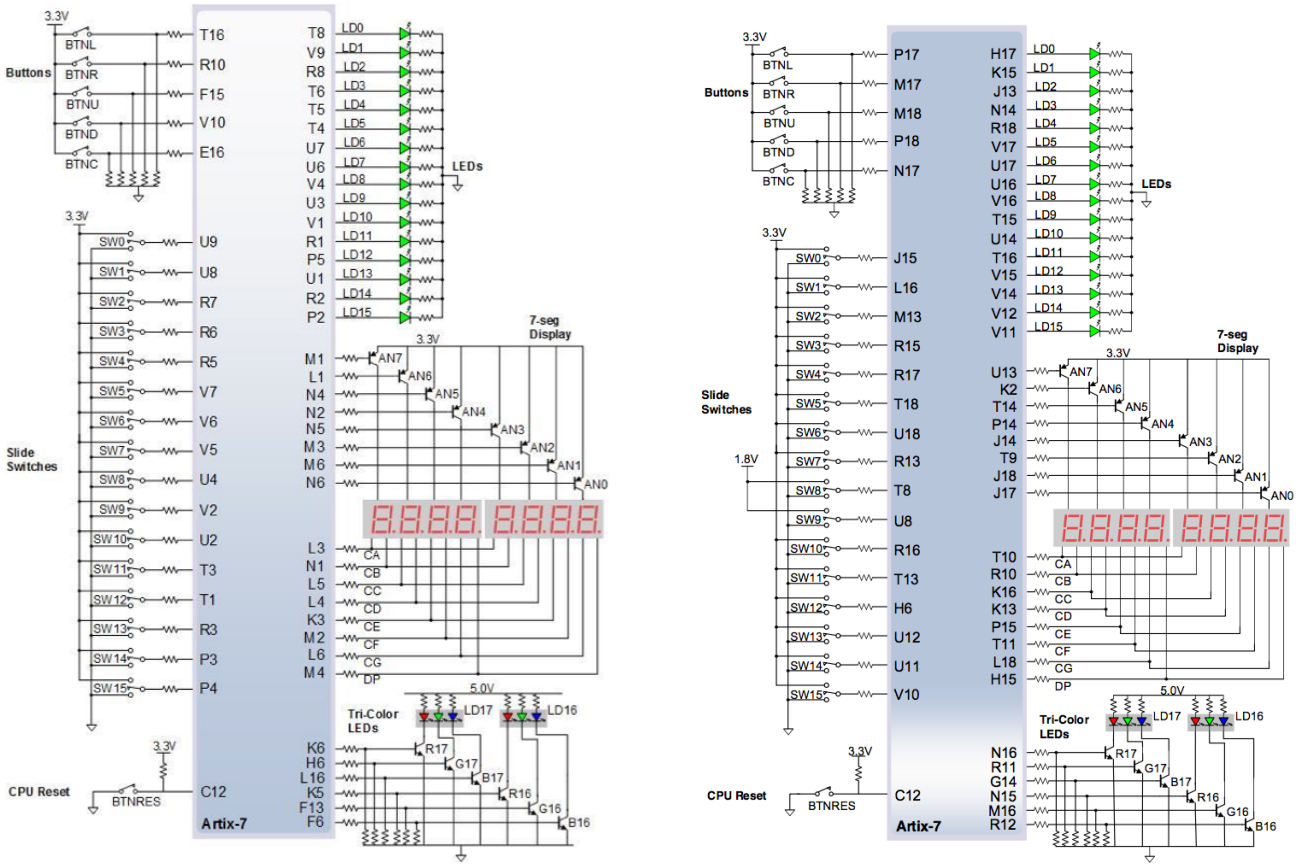
Create a new project called Lab3B. Since we are now using the actual hardware kits, be sure to use the correct part in the project settings, as shown in the picture below:



Download the SystemVerilog file **dec7seg.sv** from the class website. Add it to your project by clicking *Add Sources* → *Add or create design sources,* and be sure that the *Copy sources into project* box is checked. Understand that checking this box makes a *copy* of the file you select, and places it into the project folder. If this box is unchecked, the downloaded file is left where it is but still added to the project. In my opinion, it is always better to use the copy option, 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.

Your board has an 8-digit segmented display. In this lab, we will be using only a single digit (the rightmost one); the remaining seven will be kept unlit.

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.



Switches and LEDs: Nexys 4 (left); and Nexys 4 DDR / Nexys A7 (right)

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]. For Nexys 4, these are pins {R6, R7, U8, U9} (see figure above, and also find these labels next to the switches on the board). In this mapping, R6 is the MSB and U9 is the LSB. For Nexys 4 DDR / Nexys A7, these are pins {R15, M13, L16, J15}.
- We want to connect the 8-bit display output of the Verilog module, `segments[7:0]` to the pins L3-M4 for Nexys 4, or pins T10-H15 for Nexys 4 DDR / Nexys A7.
- Finally, we want to connect the digit select output of the Verilog module, `digitselect[]`, to the corresponding selection outputs on the board ("anode selects"), i.e. pins M1-N6 for Nexys 4, or pins U13-J17 for Nexys 4 DDR / Nexys A7.

These connections are specified in one or more "Xilinx Design Constraints" (XDC file). Download the files **segdisplay.xdc** OR **segdisplay_DDR.xdc** (depending on your board model), and **switches.xdc** OR **switches_DDR.xdc** from the course website, and do the following: Click on *Add Sources* → *Add or create constraints* → *Add Files,* pick the files you just downloaded, and be sure to check the *Copy constraints files into project* box.

The figures below (from the reference manual) illustrate the pin assignment to the segmented display.

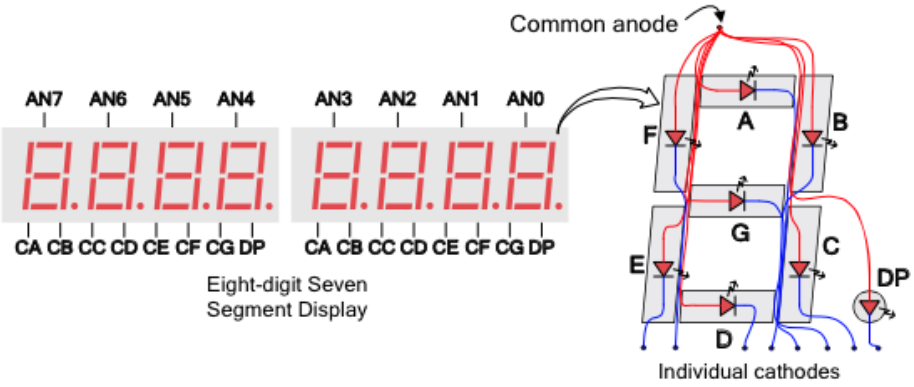


Figure 18. Common anode circuit node

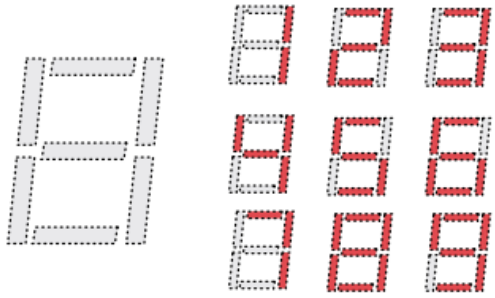


Figure 17. An un-illuminated seven-segment display, and nine illumination patterns corresponding to decimal digits

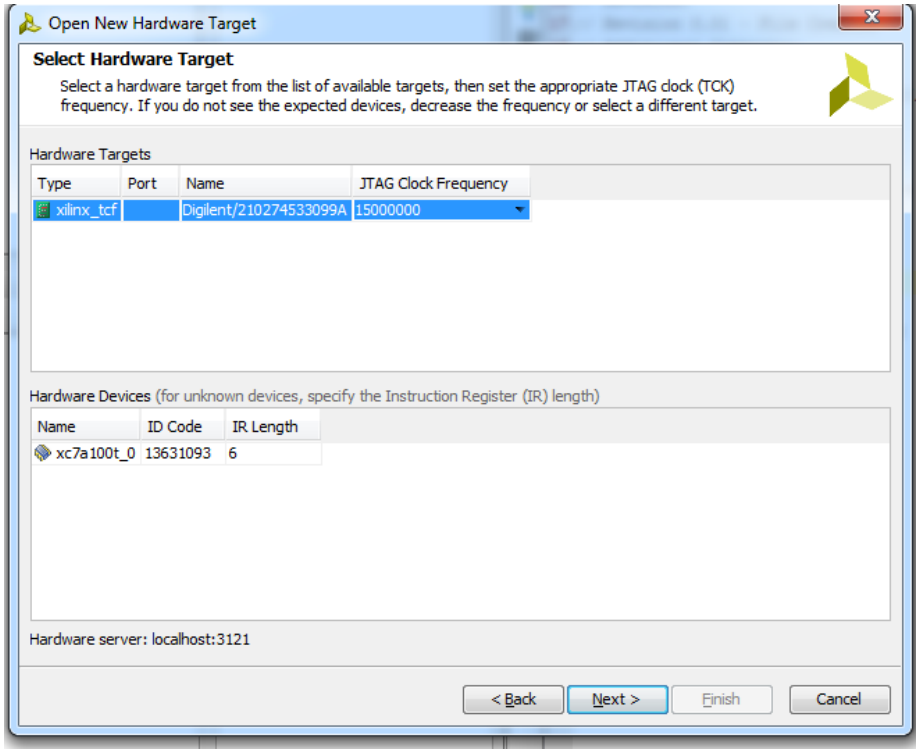
Carefully go through the Verilog source and the reference manual, and be sure you understand all of the following:

- What does `digitselect[]` do?
- Why is 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 at a high speed, we see the illusion of a multi-digit display! This is the task for the next lab!)

Now let us compile and get it ready for the board. First click *Synthesis* → *Run Synthesis.* Once it completes successfully, click *Run Implementation,* and then *Generate Bitstream.* Or, simply click *Generate Bitstream,* and the tool will automatically perform the earlier steps in sequence. Once successfully completed, this step generates a ".bit" file in your project folder (**decto7seg.bit**). Look for it. (You may click *Cancel* on the pop-up that is prompting you to run the *Hardware Manager* at this time.)

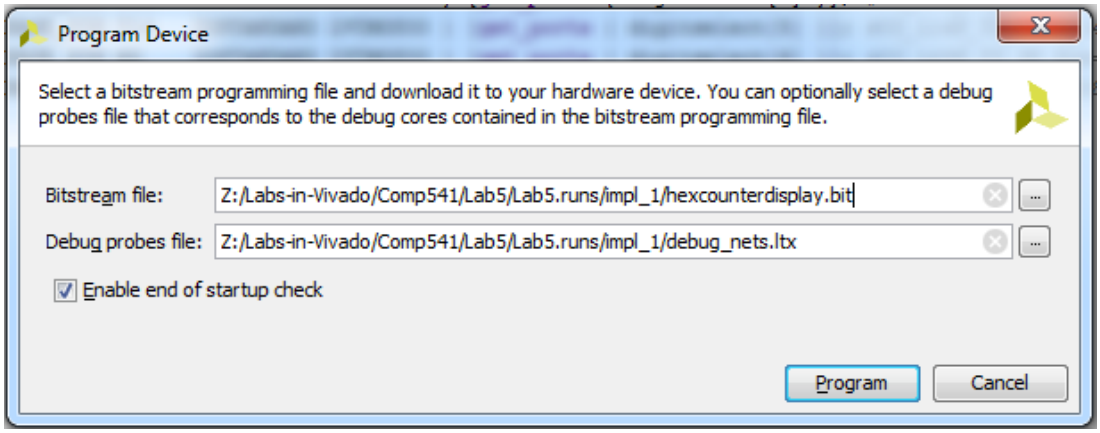
Now let us download the implementation onto the board. This step is also called "programming the board" or "configuring the board." First, connect the board to your computer according to instructions provided in Part 0, and be sure it is turned on and has passed the self-test. Now click *Hardware Manager* → *Open Target* → *Open New Target,* which launches the hardware target wizard. Choose *Connect to: Local server,* and click *Next.* You should see the following:



NOTE: If you are running Vivado inside a virtual machine, you must ensure that the Nexys board connects to the USB port of the *virtual machine* instead of the host machine! This is typically controlled by the VM software. For example, if you are using VirtualBox, from its main menu, select Devices, then USB, and underneath, check "Digilent USB device…", which will then connect your board to the virtual OS. Also, for VirtualBox, make sure you installed the Extension Pack as described in Lab 1 instructions, and the USB setting is 2.0 or 3.0 (try both if needed).

Click *Next* and then *Finish* to complete the connection to the Nexys 4 board.

Next, click *Program Device,* and then click on the device name, *xc7a100t_0,* under it. You should see a dialog box like the following (the file path will be different for you, and the debug probes entry may be empty):



Finally, click *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)

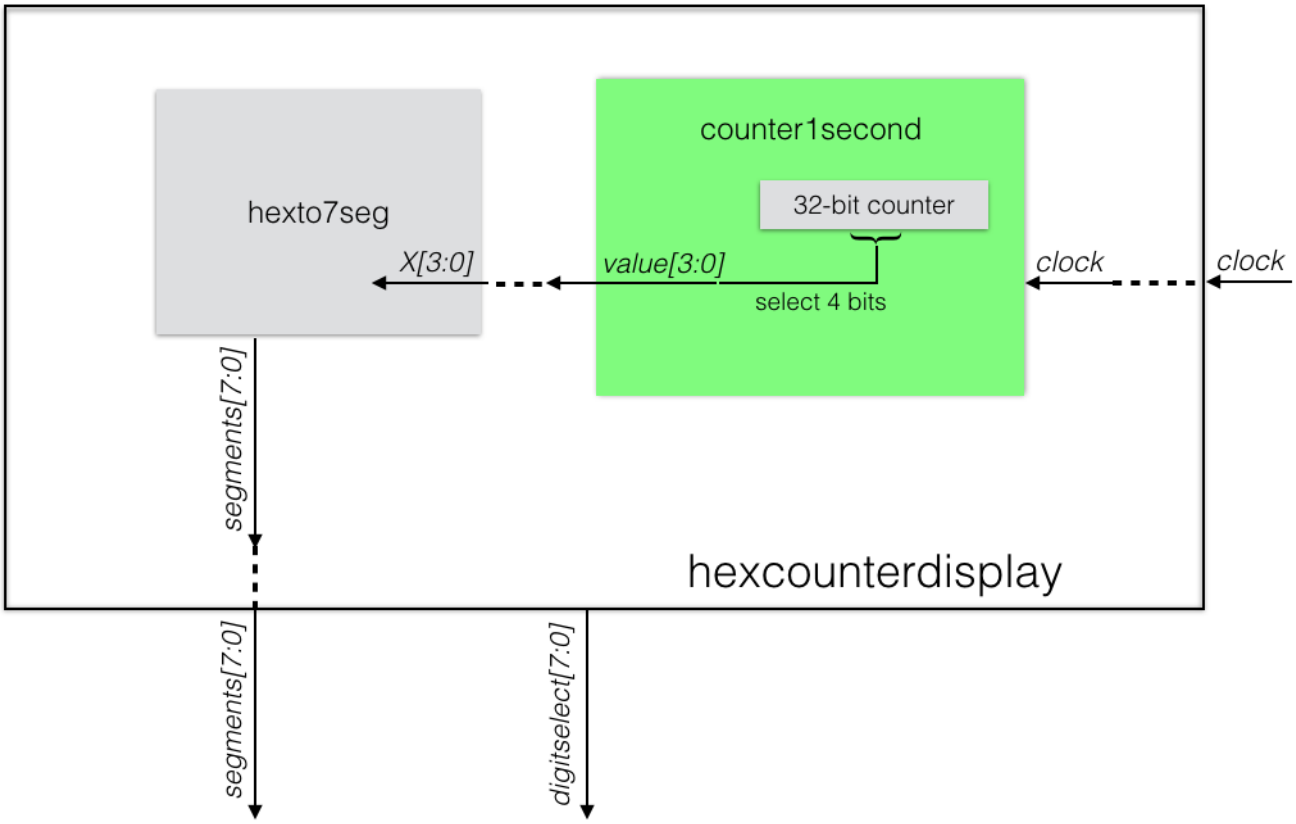
Make a copy of **dec7seg.sv** from Part I and give it the name **hexto7seg.sv.** Right-click this new file and select *Set as Top.* Modify the display encoder to handle a hexadecimal digit (i.e., "0" to "F"). Also, change the module's name to `hexto7seg()`. You will simply need to add six additional lines of Verilog to the 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 bitstream 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 E3 for Nexys 4/Nexys 4 DDR/Nexys A7 boards (see Oscillators/Clocks in the manual). Copy the **clock.xdc** file from the class website and add it to your project under the *Constraints* group via the *Add Sources* → *Add or create constraints* menu.
- 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 this number 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.
- *Constraints/XDC files:* You will need to remove the slider switches from the design since they are no longer used. While you can do so by removing the file **switches.xdc** from the project, a better way is to right-click that file and click *Disable.* Remember to re-enable it later if you need to use the slider switches.
- 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.sv** (your hex to 7-segment display encoder from Part II). Modify it so it *no longer produces* `digitselect[]`, which should instead be produced by the top-level module. Simply comment out the appropriate lines with `digitselect[]`.
    2. **counter1second.sv** (your counter module with a 4-bit output that changes roughly once per second)
    3. **hexcounterdisplay.sv** (the top-level module), which should contain one instance of the module `hexto7seg`, and one instance of the module `counter1second`. This top-level module should now produce 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.**



Note: In the figure above, the 4-bit input to the module `hexto7seg` is called *X[3:0],* while the 4-bit output from the `counter1second` module is called *value[3:0]*. These are names *internal* to those modules' definitions. In the parent module `hexcounterdisplay`, which encloses the other two modules, you will declare a 4-bit wire, with a name of your choice, which will connect the two modules together. This is similar to function/procedure declarations in software languages such as C: the names of variables inside a function need not be the same as the names used by the calling function.

What to submit:

- **The three Verilog sources (hexto7seg.sv, counter1second.sv, and hexcounterdisplay.sv).**

How to submit:

- **Submit via Sakai ("Lab 3B" under Assignments).**
- **Include the attachments as described above.**
- **Submit your work by 11:55pm on Wednesday, September 8.**

Demo: **Show a working demo of your design for Part III during the lab session on Friday, Sep 10.**
