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

Comp 541 Digital Logic and Computer Design
Prof. Montek Singh
Spring 2020

Lab #5: Input/Output Devices
Issued Fri 2/7/20; Due Fri 2/14/20 (11:59 pm)

You will learn the following in this lab:

- Driving a multi-digit 7-segment display
- Working with a USB keyboard
- Creating sound waveforms using pulse-width modulation
- Using the built-in accelerometer
- Driving output LEDs
- Integrating all the I/O devices together into a demo

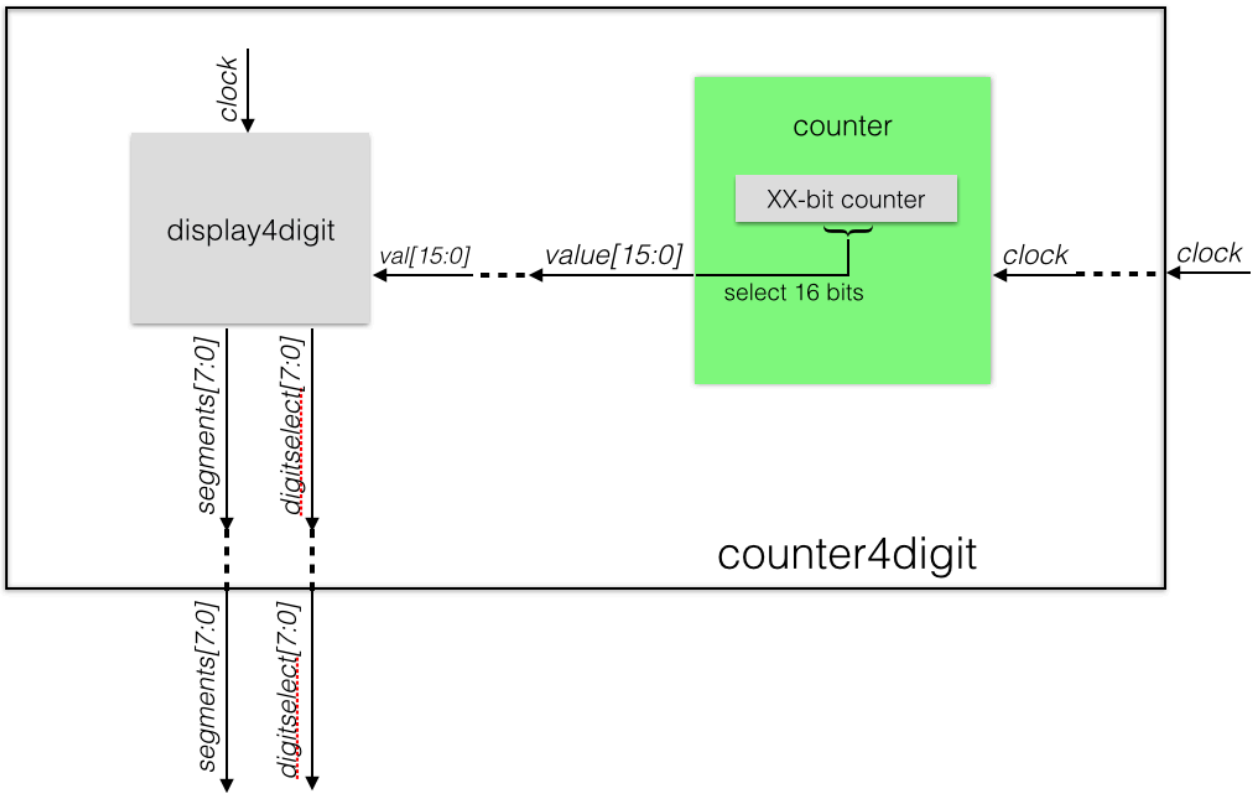
Part 0: Handling a 4-digit seven-segment display

Create a new project, `Lab5`. Copy your `hexto7seg.sv` from Lab 3B to the new project (you may use "Copy Source"). Do the following:

- Create a new top-level module called `counter4digit`. A block diagram appears on the next page.
- Download the Verilog file `display4digit.sv` from the website, and add it to the project
  - This file contains a module called `display4digit()`. This is a higher-level module that drives all four digits of the display in rapid succession, each time generating the appropriate segment values using a *single hex encoder* (i.e., single instance of `hexto7seg`).
  - Pay careful attention to the following: How each of the 4 digits of the display is selected in a round-robin fashion; how different hex values are selected for generating the output, again in a round-robin fashion.
- Feed 16 bits (instead of 4 bits) into this display, from a counter module. This module is labeled "counter" in the picture below (shaded green). You will have to change the width of the counter to be more than 32 bits in order to be able to select 16 bits that are not changing too fast. Choose these 16 bits such that the least-significant hexit is changing about 16 times a second; the next hexit will then change once a second; the one after that will change once every 16 seconds; and the leftmost one will change once every 256 seconds. All these times are accurate to within a factor of two.
- Copy the XDC file from the last part of Lab 3B.

Question 1: Calculate the refresh frequency for each digit of the display. How many times per second is one particular digit of the 4-digit display visited per second?

You are encouraged to play with the refresh frequency by changing which bits of the internal variable `c` are used to perform the round-robin selection of display digit. For example, using higher numbered bits (`c[23:22]` instead of `c[18:17]`) will slow the display refresh down enough for you to see the round-robin selection. (Here, `c` refers to a variable within the `display4digit` module used for round-robin digit selection on the display.) You may try other slower/faster values also.



Part 1: Extend to an 8-digit seven-segment display

Extend the 4-digit display from Part 0 to an 8-digit display module called `display8digit`, in file `display8digit.sv`. Do the following:

- Create a new top-level module called `counter8digit`, similar to the counter4digit you created in Part 0. That is, extend the 4-way round-robin digit selection to 8-way.
- Select the speed of round-robin selection so each digit is refreshed with the same frequency as in Part 0. That is, now you will need to spend only half as much time to display each digit so that, once all 8 digits have been displayed, the same digit is re-visited at the same frequency as in the 4-digit display.
- Feed 32 bits (instead of 16 bits) into this display, from the counter module. Choose these 32 bits such that the *least-significant hexit is changing **approximately** 256 times a second*; the next hexit will then change 16 times a second; the one after that will change once a second; then next every 16 seconds; and so on. All these times should be accurate to within a factor of two.

Question 2: For how much time is each digit displayed before moving on to the next digit?

Part 2a: Understand the keyboard module

Copy the files for the keyboard module from the website. The main keyboard module description is in `keyboard.sv`. Take a quick look and see if you understand it. Refer to the Nexys board manual section on *HID Controller* (Sec. 7 for Nexys 4, and Sec. 8 for Nexys 4 DDR boards). It is okay if not all of it makes sense at this time; we will discuss in detail in class later.

A demo is provided in file `keyboard_demo.sv`, along with an XDC file. Use your Verilog for the 8-digit segmented display from Part 1, in file `display8digit.sv`. Mark the demo as your top-level module, and implement on the board. Connect a USB keyboard (available in the classroom closet) to the USB connector (labeled "USB HOST", near the power switch). As you press and release keys on the keyboard, a special code for each event (called a *scancode*) is received from the keyboard, and displayed on the segmented display.

Part 2b: Understand the sound module

Next, copy the files for the sound module into the same project (leave the keyboard files as is). The main sound module description is in `sound.sv`. Take a quick look and see if you understand it. It simply generates a square waveform, as an approximation of a sine wave, with a period that is provided to it as an input. Again, it is okay if not all of it makes sense at this time; we will discuss in detail in class later.

A demo is provided in file `sound_demo.sv`, along with an XDC file. Now mark the sound demo as your top-level module, and implement on the board. Connect a pair of headphones or speakers to the audio output port (labeled "MONO AUDIO OUT", at the top right of the board). You should hear a sound that will cycle from note C4 to C5, over and over, with each note being played for approximately 1 second.

Part 2c: Understand the accelerometer module

Finally, copy the files for the accelerometer module into the same project (leave the keyboard and sound files as is). There are nine source files zipped together; download and unzip them, and add all of them to your project via the *Add Design Sources* dialog. The main accelerometer module description is in `accelerometer.sv`. Take a quick look at this file; we will discuss more in class later. There are several helper files that have the bulk of the hardware description of the accelerometer module, include a sophisticated communication interface. These helper files (provided by the board manufacturer) are written in VHDL instead of Verilog; you do not have to understand what's in these! You will simply use the top-level Verilog, which puts a simple wrapper around the implementation details.

A demo is provided in file `accel_sound_demo.sv`, along with XDC files (different files for the two board versions). It will re-use the 8-digit display module and the sound module described above. Now mark the demo module (`SoundAccel`) as your top-level module, and implement on the board. Connect a pair of headphones or speakers to the audio output port. Pick up the board, and tilt it left and right. As you change the angle of the tilt, you will see the components of gravity along the *X* and *Y* directions displayed on the segmented display. You will also hear a sound whose tone will change, depending on the tilt along the *Y* axis. Specifically, the period of the sound will be proportional to the *Y* tilt.

Part 3: Integrate the I/O devices into a new demo

Your task is now to create a new demo that integrates all of these I/O devices in a new manner. Create a new top-level module called `io_demo`, and put it in file `io_demo.sv`. Do the following:

- Integrate the keyboard and the sound so that the tone produced by the sound module is a function of the scancode coming out of the keyboard. Just select 8 keys of your choice on the keyboard, generating 8 different scancodes, and select a tone from C4 to C5 based on the scancode (simply using a nested conditional statement, effectively implementing a large multiplexer).

- Integrate the accelerometer with the 16 LED lights just above the slider switches. As you tilt the board left or right, one of the LEDs should light up, somewhat in proportion to the amount of the tilt. See the Nexys 4 manual for information on the LED lights, and see your board for the pin numbers for the lights.

What to submit:

- **The top-level SystemVerilog sources: io_demo.sv and display8digit.sv**
- **Your answers to Questions 1 and 2.**
- **Show a working demo during the class/lab session on Fri Feb 14 if completed by that time. If you complete it after the lab session, please attach a video (phone camera is fine) to your submission.**

How to submit: **Please submit your work by email by 11:59 pm on Fri Feb 14 as follows:**

- **Send email to: `comp541-submit-s20@cs.unc.edu`**
- **Use subject line: Lab 5**
- **Include the Verilog sources as specified above, and your answers to Questions 1 and 2 within the email body.**
