Please complete these purchases and installations before the end of the 1st week of class

PURCHASES AND VIVADO INSTALLATION:
You need to open accounts at Xilinx:
    https://www.xilinx.com/
And Digilent:
    https://digilent.com/login.php

Be sure to indicate you are a student so you can get the academic discount (see email instructions on process).

You will need to buy the following FPGA at the academic discount.
    https://digilent.com/shop/zybo-z7-zynq-7000-arm-fpga-soc-development-board/

You will also need to buy:
- A microSD card. I recommend 8 GB but no larger than 16 GB.
- A microSD card reader/writer if you don't have one on your laptop already.
- A 1 to 3 foot microUSB cable.
- A short 1 to 3 foot twister-pair Ethernet cable.

INSTALLATION:
You'll need to download Vivado 2020.2 (newer versions should be okay).

Licensing is free for certain device classes, such as the Zynq 7010 FPGA that is on the FPGA board.

I have an installation video that is a bit dated but will work to step you through the process using linux as your development platform on your laptop.

Follow these steps:
- First, go to:
  https://www.xilinx.com/support/download.html
- Click on 'Vivado Archive' if 2020.2 is not listed.
- Click one of Web installer links depending on your OS

Once you have Vivado installed, you will need to download the following files and place them in the installation directory.

Under linux, install the board files if needed. First check if the zybo-z7-10 directory exists in the following directory. If not, perform the following:

```
cd <install_dir>/Vivado/<verson>/data/boards/board_files
mkdir zybo-z7-10
cd zybo-z7-10
mkdir A.0
cd A.0
cp <board_files>/board.xml .
```

```
   cp <board_files>/part0_pins.xml .
   cp <board_files>/preset.xml .
```

If you do this correctly, you will be able to select the Zybo-Z7-10 board under ‘boards’ when you create a Vivado project.

LINUX:
The Zybo-Z7-10 is a System-on-Chip FPGA, i.e., it has both a hardwired microprocessor (called the PS side) and a programmable logic (PL) side.

The processor enables you to run the Linux operating system so you can login directly to your board and run programs on the FPGA directly, exactly like you do on your laptop. You connect to the FPGA using either a serial (via USB) connection or the network (via twister-pair Ethernet). You always connect using the serial USB connection first and then this will enable you to configure the network on the FPGA so that you can logon through twister-pair connection instead (or in addition to). Remember, Linux is a multi-user, multi-tasking operating system so you can create multiple separate logins as needed.

You typically use a program called ‘minicom’ under Linux or ‘teraterm’ (or equivalent) under Windows to connect through the serial USB to the FPGA.

You need to first ensure the following:
- Connect the microUSB cable between your laptop and the FPGA board
- Power-on the FPGA with the switch
- Set the jumper to boot from micro-SD card (put jumper OVER the SD label)
- Have a micro-SD card plugged into the FPGA slot

Note: You must have the Linux OS already installed on the micro-SD card (see micro-SD CARD CREATION below).

There is NO windows version for the FPGA, you must use Linux.

Linux micro-SD CARD CREATION
I have provided pre-built OS images on my website for downloading and installation to the microSD card.

Once you have the microSD card AND a microSD card reader/writer, you'll need to recreate the partition table on the SD card. Typically, it has only one Windows-based filesystem partition. You need to create a 1 GB Fat32 partition (Boot partition) and a x GB ext4 (linux) partition for the linux filesystem.

Follow these steps to create a bootable micro-SD card if you are running Ubuntu Linux on your laptop (Windows users will do something similar but using Windows commands):
1) Mount the disk (plug it in) and note device name, e.g., /dev/sdc
    ***** WARNING: DO NOT PROCEED UNLESS YOU ARE CERTAIN OF THE DEVICE NAME for your SD card

    In the following, I assume the sd card is given the device names **/dev/sdc**

2) Unmount (run the following in a Linux xterm window)
```
umount /dev/sdc1
```

3) Create the boot and root partitions.
    WARNING: BE VERY CAREFUL WITH THIS COMMAND. BE SURE YOU CHOOSE THE SD CARD device in /dev, e.g., /dev/sdc
    NOTE: USE /dev/sdc and NOT /dev/sdc1

    Run the following in a Linux xterm window
```
fdisk /dev/sdc
      p (prints the partition table -- probably none at this
point)
      n, p, 1, 2048, +1G (create a 1 GB FAT32 filesystem)
      t, b (change 1G partition to be W95 FAT32)
      a, 1 (make the FAT partition bootable)
      n, p, 2, <use default>, <use default, rest of disk>
      p (verify it is linux)
```
The 'p' command (print partition table) should show 'something' like the following. You out may be different depending on which disk you mounted and the size of the SD card you are using.
```
Disk /dev/sdc: 15.9 GB, 15931539456 bytes, 31116288 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x00000000

   Device Boot      Start         End      Blocks   Id  System
/dev/sdd1   *          63     2120579     1060258+   b  W95 FAT32
/dev/sdd2         2120580    31101839    14490630   83  Linux
```

Do NOT forget to write the changes to the actual disk with the following command.

```
        w (write out the updates)
```

4) Format the partitions (run the following in a Linux xterm window)

```
mkfs.vfat -n BOOT /dev/sdc1
```

5) Mount the boot partition (run the following in a Linux xterm window)

```
mount /dev/sdc1 <mount_point>/boot
```

6) Copy the BOOT files from my weblink to the FAT32 boot partition (run the following in a Linux xterm window)

```
cp BOOT.bin <mount_point>/boot
cp image.ub <mount_point>/boot
```

7) Unmount the boot partition (run the following in a Linux xterm window)

```
umount <mount_point>/boot
```

8) Create the root filesystem on the root partition

WARNING: BE VERY CAREFUL WITH THIS COMMAND. BE SURE YOU CHOOSE THE SD CARD device in /dev, e.g., /dev/sdc2

Run the following in a Linux xterm window:

```
dd if=rootfs.ext4 of=/dev/sdc2
resize2fs /dev/sdc2
sync
```

The SD card should now be in a bootable state and will have a persistent filesystem.

9) Check that there are no errors once this command finishes. Double check it succeeded by mounting the root partition (run the following in a Linux xterm window)

```
mount /dev/sdc2 /mnt
```

10) Check that the root filesystem exists (run the following in a Linux xterm window)

```
cd /mnt
ls
```

You should see a list of directories, such as root, etc

11) Unmount the root partition (run the following in a Linux xterm window)

```
umount /mnt
```

12) Plug the SD card into you Zybo board and power on.

13) Check to see if two USB devices have been created in the /dev directory

```
ls -ltra /dev/ttyUSB*
```

Typically you will see ‘ttyUSB0’ and ‘ttyUSB1’

14) Use minicom to logon (run the following in a Linux xterm window), and then type the single letter commands, e.g., ‘o’, without the single quotes

```
   minicom -s
   ‘o’, select Serial Port setup
   ‘A’, /dev/ttyUSB1
   ‘F’, Hardware Flow Control MUST BE OFF
   <cr>, To return to main menu
   Select Exit, and press <cr> a couple times, you should see the
   login prompt.
```

To bring up minicom menu again while running minicom, type

```
   <Ctrl>A, Shift-Z
```

You should see the following prompt once you log in.

```
   root@Zybo-Z7-10:~#
```

If it says anything else, type ‘boot’ and hit enter.

NOTE: PLEASE BE SURE THAT you set the minicom parameters with ‘Hardware Flow Control’ OFF. ALSO, PLEASE BE SURE THAT the serial port parameters are set to 115200, N, 8, 1 (see minicom discussion).

Linux Embedded Within Windows or Native Linux Installation on your LAPTOP:
I included here some useful Linux commands in case you run into problems with, e.g., connecting to your FPGA using mincom or programming the FPGA from your laptop.

Note: Do not confuse these instructions with the Linux that is running on the Zybo Board. This is for students who either have Linux natively installed on their laptops or decide to embed Linux within Windows as described by instructions elsewhere.

The most common problem is connecting to the FPGA using the serial port (via minicom). Another method to connect to the FPGA is through the network using a twisted-pair ethernet cable, which is covered later. The current configuration of linux on the Zybo board boots up with the IP set to 192.168.0.10

The Zybo board has only one micro-USB cable that serves three purposes: 1) It enables a serial connection between the host laptop and the FPGA, 2) It allows the FPGA to be programmed using the Hardware Manager within Vivado (NOTE: THIS DOES NOT WORK WITH THE CURRENT kernel, you must program using the PCAP interface described below) and 3) It supplies power to the board.

When you plug in the micro-USB into your laptop and power-on, several ports are automatically created by the Linux OS running on your laptop. In Linux, all communication ports, serial, USB, network, hard drive, etc. are represented as files, usually in the directory:

```
/dev
```

The serial ports are easily recognized because they have a 'tty' prefix. I usually issue the following Linux command from within an xterm window to see what names Linux has assigned to the serial ports just created:

```
ls -ltra /dev/ttyUSB*
```

This command displays a long listing (with dates, sizes, etc.) of all files in /dev and orders them reverse chronologically with the newest dated files at the very bottom of the listing. You will see something similar to the following:

```
crw-rw----.  1 root dialout 188,   1 Aug  5 21:24 ttyUSB0
crw-rw----.  1 root dialout 188,   3 Aug  5 21:24 ttyUSB1
```

The most important aspect of this listing are the names on the right and the permissions on the left. The Linux OS permissions are given as 'x rwx rwx rwx', where the right three triplets indicate the permissions associated with the 'owner', the 'group' and 'everyone else', respectively. The listing for both of these ports indicate that the owner and group have read/write permission. But the owner is 'root' and the group is 'dialout', in which case, you, as a user, are excluded from these and instead fall into the 'everyone else' category, which has no permissions at all.

I typically just issue the following sequence of commands to make sure I have access to these devices. First I become superuser by typing the following and then enter the superuser password:

```
su
```

Note that under Ubuntu, you need to become superuser using the following:

```
sudo su
```

And then typing the superuser password.

Once I'm superuser (the prompt changes to a '#'), I type:

```
   chmod a+rwx /dev/ttyUSB*
```

This changes the permissions to allow anyone to access these devices.

Exit superuser mode by typing <Ctrl-D> BEFORE RUNNING minicom.

Now you can run:

```
   minicom -s
```

And then scroll down to 'Serial port setup', and press Enter. You should see a menu of A through G. To change the Serial Device, press 'A' and change the entry to point to one of the two devices shown earlier, e.g.,

```
   /dev/ttyUSB1
```

Also, PLEASE BE SURE THAT 'Hardware Flow Control' is No. If it is Yes, press F to toggle it. ALSO, PLEASE BE SURE THAT the serial port parameters are set to 115200, N, 8, 1.

Pressing Enter again returns you to the main menu. You can save the 'setup' by selecting 'Save setup as.." and entering a filename. The file should appear in your home directory. If you are successful, you can just type 'minicom' next time and it will be configured using your default configuration.

Note that only one of these devices is the serial port, either USB0 or USB1. Try both. If you are successful, you will see the a Linux prompt with a '#' sign, indicating you are root. NOTE: THIS IS THE Linux OS RUNNING ON THE FPGA -- not the Linux running on your laptop -- be careful, it is easy to get confused. Note that you may be required to logon. Logon by typing 'root' for the username and 'root' for the password.

NOTE: THE FOLLOWING WORKS ONLY IF YOU HAVE NOT BOOTED Linux (you set the jumper to QPSI). YOU MUST USE PCAP TO PROGRAM THE BOARD WHEN Linux IS RUNNING:

As indicated above, a USB programming port is also created when you power on the board. You will need to change the permission of that port as well to enable Vivado to program the FPGA. The programming port is an actual USB port. Use the following command to list all USB devices plugged into your laptop:

```
   lsusb
```

You should see a listing of 10-15 items. The FPGA device will look similar to the following:

```
   Bus 001 Device 004: ID 0403:6010 Future Technology Devices
   International, Ltd FT2232C/D/H Dual UART/FIFO IC
```

The string 'Future Technology Devices International' is the USB port created when you power on your FPGA.

The two numbers on the far left are components of the name that Linux assigns to the device. You can inspect the default permissions of this device by the following command:

```
   ls -ltra /dev/bus/usb/001/004
```

Your numbers are likely to be different so replace here as needed. My default listing is as follows:

```
   crw-rw-r--. 1 root root 189, 3 Aug 14 14:28 /dev/bus/usb/001/
   004
```

Although everyone can read from this device, only ‘root’ can write to it. Open up permissions by becoming superuser again (see above) and then using the following command:

```
   chmod a+rwx /dev/bus/usb/001/004
```

Exit superuser by pressing <Ctrl-D> BEFORE RUNNING Vivado. You should be able to run vivado by typing:

```
   vivado
```

And then the Hardware Manager should show your device as described in the video.

NOTE: Those of you who embed Linux within Windows will usually see serial devices as follows:

```
   /dev/ttyS0
   /dev/ttyS1
   ...
```

Remember Linux is running inside of Windows, so Windows ‘passes’ control to serial devices to Ubuntu. Use the device manager in Windows to see which COM ports are created when you power on your FPGA. Plug the micro-USB cable in and out to see what devices appear. Once you know the COM port number, e.g., COM5, the corresponding Linux port is /dev/ttyS5.

Network Instructions:

NOTE: The current kernel boots with the IP set to 192.168.0.10 so you do NOT need to do the following to configure the network.

For those of you who would like to logon to your FPGA using the network, you can use these instructions to setup the network.

You connect a regular twisted-pair ethernet cable between the port on your FPGA and the port on your laptop (if you don’t have one on your laptop, you’ll need to buy an USB-ethernet adaptor).

As indicated above, you MUST first logon using the serial port. Once you are logged on to the FPGA running Linux, type the following to view the default network configuration:

```
   ifconfig
```

You should see something similar to the following:

```
   eth0      Link encap:Ethernet  HWaddr 00:0A:35:00:1E:51
             inet addr:192.168.1.10  Bcast:192.168.1.255  Mask:255.255.255.0
             inet6 addr: fe80::20a:35ff:fe00:1e51/64 Scope:Link
             UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
             RX packets:72614779 errors:0 dropped:883 overruns:0 frame:0
             TX packets:58228458 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:1000
             RX bytes:1672147321 (1.5 GiB)  TX bytes:3866989628 (3.6 GiB)
             Interrupt:27 Base address:0xb000
```

I have already configured the ethernet connection ‘eth0’ in this listing because the ‘inet addr: 192.168.1.10’ is already defined. You will likely see that nothing is assigned to inet addr.

Type the following to define the first part of a local area network:

```
   ifconfig eth0 192.168.1.10
```

Typing ifconfig again should show a listing similar to mine. I usually also type the following to make sure a default route is setup:

```
   route
```

You should see something like the following:

```
   192.168.1.0     *               255.255.255.0   U     0      0        0 eth0
```

The command ‘ifconfig’ usually also sets up a default route for you. If it doesn’t, you’ll need to add one, typically with a command as follows:

```
   route add default gw 192.168.1.0 eth0
```

Now you need to setup the network on your laptop in a similar fashion. In a different xterm (NOT the one that you used to connect to the FPGA via mincom), type the following command:

```
   ifconfig
```

And then

```
   ifconfig eth0 192.168.1.20
```

NOTE: I assign the laptop using ‘.20’ while the board is assigned ‘.10’. THEY MUST BE DIFFERENT and on the same subnet, ‘192.168.1’

You can also type

```
   route
```

And see that a route exists similar to the route shown when doing this on the FPGA above.

Windows Subsystem for Linux (WSL):
Those of you running Linux within Windows (the Windows Subsystem for Linux or WSL) can get some tips from articles such as the following:

https://seanthegeek.net/234/graphical-linux-applications-bash-ubuntu-windows/

Please post on the Discussion board if you find any errors with these instructions, or can offer additional guidance, especially for those of you who embed Linux within Windows.