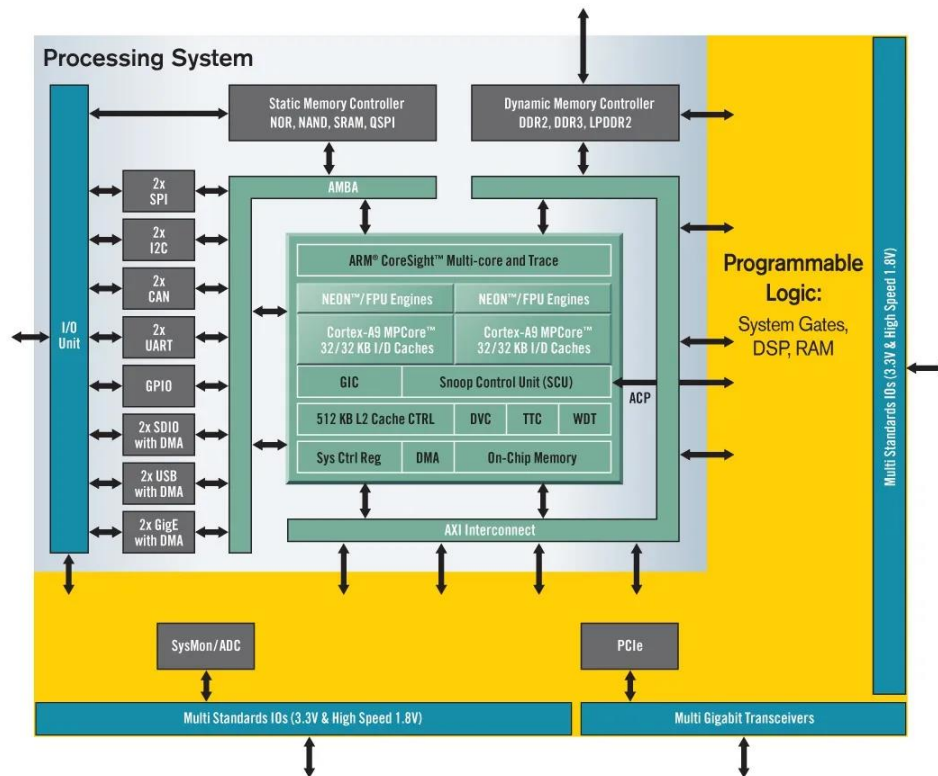


Lab 1: Setting Pynq-Z2 guide

1. Set up MicroSD Card:

- Since Pynq-Z2 is based on ZYNQ-7020 SoC that includes ARM cortex-A9 Processors, there is a processing system(PS).
- The second part of the Pynq-Z2 is the programmable logic (PL)
- By setting up the MicroSD card with Linux and into the ARM processor, to utilize the power of the arm processor while accelerates algorithms on the PL.
- This step will install an OS to the SD card using image file and help boot the ARM processors from the SD card.
- Source for the image file: <http://www.pynq.io/board.html>



- The software you can use is Win32 disk imager.

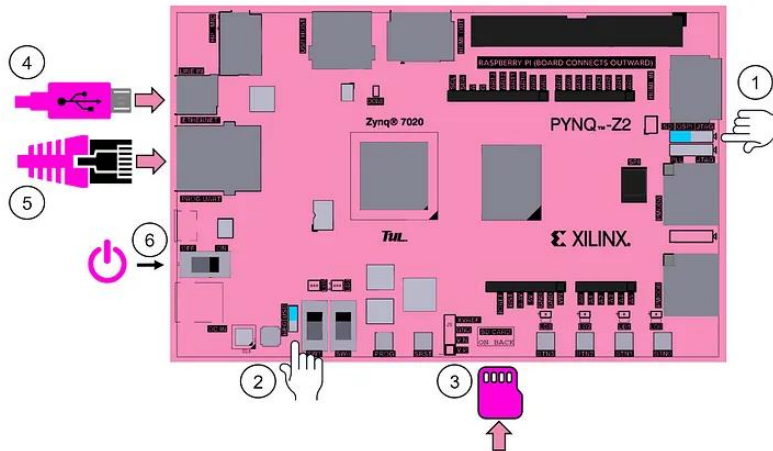
2. Board Setup

- Is a jumper to give you two options: QSPI and JTAG.
- Is where you specify the power source for the board.
- Is the slot for your micro SD card.
- Is the USB connection from your board to your PC.
- Is where you would connect to the Ethernet port using a cable.
- What do you do if you don't have internet?
 - You would need to assign static IP Address.
 - First connect your FPGA with your Laptop/PC.
 - Need to config IP address as 192.168.2.1 and subnet mask as 255.255.255.0. The rest of the field should be blank.

d. The Board has a static default IP as 192.168.2.99

e. Refer to this link:

<https://pynq.readthedocs.io/en/v2.5/appendix.html#assign-your-computer-a-static-ip>



7. Once you boot the board, the red LED will turn on first.

8. Then DONE led will turn on once SD card has been flashed properly.

9. After some time, the blue and green LEDs will start to flash.

10. The blue LEDs will turn off, while the yellow LEDs will remain on. This indicates that the board is ready for use.

3. If There is Ethernet connected to your Router

a. Whenever your device is connected to router, an IP address is assigned via DHCP Server.

b. To talk to FPGA, you need to find the IP address for your FPGA.

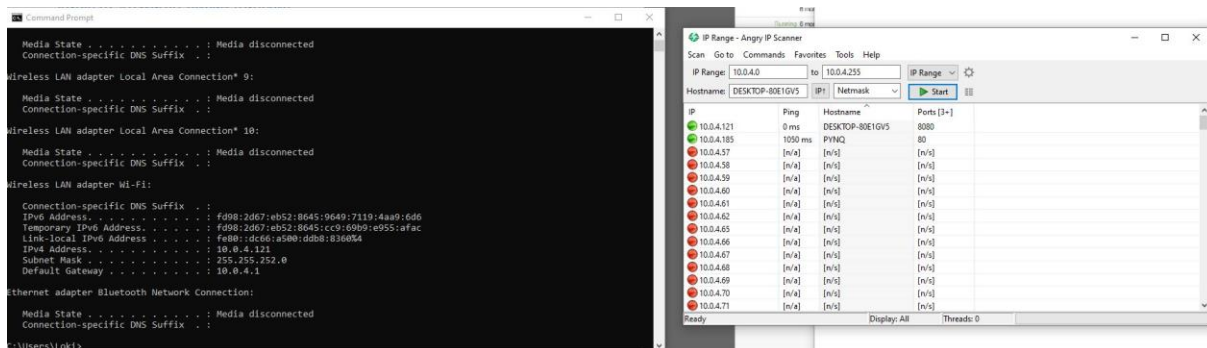
c. Using IP config, I used my IPv4 IP as my starting point of the IP range.

i. When you run IP Range, use the IP 10.0.4.12 to help you.

ii. I typed 10.0.4.0 as my starting point.

iii. Set the end point to 10.0.4.255.

iv. Click Start to find your IP addresses.



d. From the IP range, the IP for my FPGA board was 10.0.4.185.

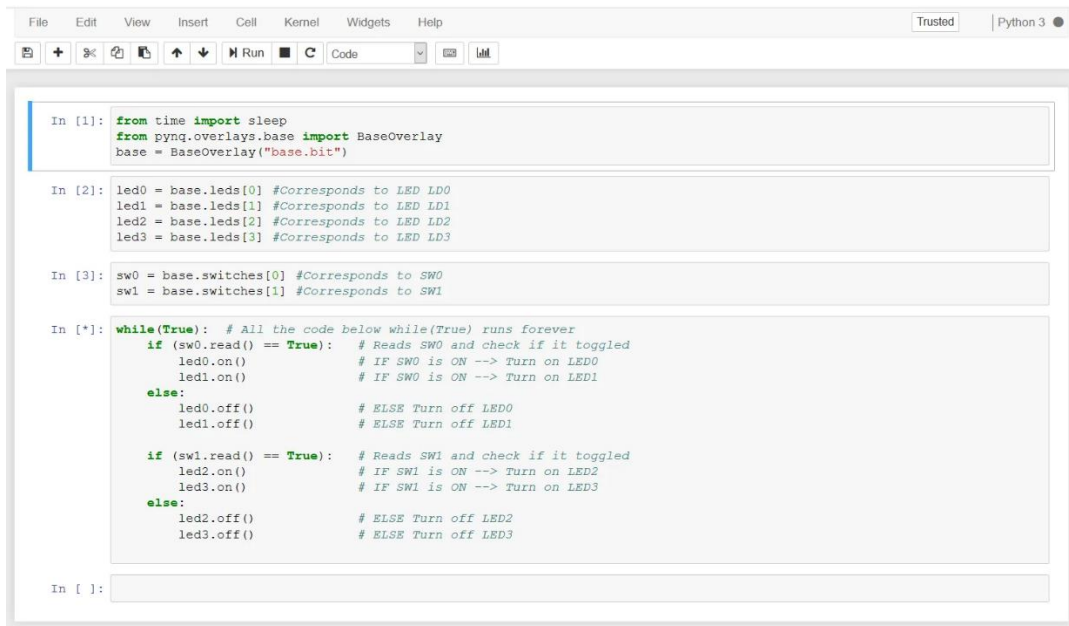
e. This is the IP address that I need to communicate with.

4. Talking to your PYNQ board

- In the browser, enter the IP address of the FPGA.
- If you have no internet, remember to refer to the previous step of setting up a static IP address.
- You will get into Jupyter-Notebook
- Jupyter is a server that runs on Linux to leverage power of ARM processors.
- Log in using password : xilinx\

5. Create first notebook

- New > Folder
- Name the folder "Examples"
- Create new Python file
- For a blinking led lab, below is the code.



```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [1]: from time import sleep
        from pynq.overlays.base import BaseOverlay
        base = BaseOverlay("base.bit")

In [2]: led0 = base.leds[0] #Corresponds to LED LD0
        led1 = base.leds[1] #Corresponds to LED LD1
        led2 = base.leds[2] #Corresponds to LED LD2
        led3 = base.leds[3] #Corresponds to LED LD3

In [3]: sw0 = base.switches[0] #Corresponds to SW0
        sw1 = base.switches[1] #Corresponds to SW1

In [*]: while(True): # All the code below while(True) runs forever
        if (sw0.read() == True): # Reads SW0 and check if it toggled
            led0.on() # IF SW0 is ON --> Turn on LED0
            led1.on() # IF SW0 is ON --> Turn on LED1
        else:
            led0.off() # ELSE Turn off LED0
            led1.off() # ELSE Turn off LED1

        if (sw1.read() == True): # Reads SW1 and check if it toggled
            led2.on() # IF SW1 is ON --> Turn on LED2
            led3.on() # IF SW1 is ON --> Turn on LED3
        else:
            led2.off() # ELSE Turn off LED2
            led3.off() # ELSE Turn off LED3

In [ ]:

```

Code:

```

from time import sleep
from pynq.overlays.base import BaseOverlay
base = BaseOverlay("base.bit")

```

Explanation:

- We first import the sleep command which is used to generate a delay if you want to experiment.

You the sleep(timeinseconds) command for experimentation. We then import the BaseOverlay

which is used to use some underlying hardware. I don't want to confuse you here by talking about Overlays for now but if you remember about Programming Logic or PL, Overlay files are generated by VIVADO in the form of ".bit". These are essentially hardware libraries that will leverage the power of FPGA. The hardware defines all the connection, inputs and the outputs. So by importing these bit files, you can share data to those inputs using APIs that we will explore next.

Code:

```
led0 = base.leds[0] #Corresponds to LED LD0
led1 = base.leds[1] #Corresponds to LED LD1
led2 = base.leds[2] #Corresponds to LED LD2
led3 = base.leds[3] #Corresponds to LED LD3
```

Explanation:

- Now we need a reference to the LEDs somehow. These reference are provided by the bit file. We can get access to the four leds using base.leds[LEDNo] and then we store them in a variable. Essentially, you are just getting a reference to the INPUTS.

Code:

```
sw0 = base.switches[0] #Corresponds to SW0
sw1 = base.switches[1] #Corresponds to SW1
```

Explanation: Similarly we can get reference to the switches by using base.switches[SWITCHNo].

Code:

```
while(True): # All the code below while(True) runs forever
    if (sw0.read() == True): # Reads SW0 and check if it toggled
        led0.on()           # IF SW0 is ON --> Turn on LED0
        led1.on()           # IF SW0 is ON --> Turn on LED1
    else:
        led0.off()          # ELSE Turn off LED0
        led1.off()          # ELSE Turn off LED1

    if (sw1.read() == True): # Reads SW1 and check if it toggled
```

```

    led2.on()      # IF SW1 is ON --> Turn on LED2
    led3.on()      # IF SW1 is ON --> Turn on LED3
else:
    led2.off()     # ELSE Turn off LED2
    led3.off()     # ELSE Turn off LED3

```

Explanation:

- If you have no experience in python, then this can be a bit troublesome. But let me try to explain what we are doing. Also, please in mind that indentation is necessary. Blocks are identified using indentation. There are no brackets in Python that can define the block. Indentation is used for this purpose. We want our program to run forever polling the switches and make decisions. The indentation is defined below.

```

while(True):
    Belongs to while loop
    Belongs to while loop
Doesn't belong to while loop

```

We can use on() or off() function to power on or power off the led. Since we have a reference to all four

LEDs i.e led0, led1, led2, led3 we can power these LEDs by using

```
led0.on() # Turns on LED0.
```

You can also refer to LEDs without storing them in the variable. But its generally better to use variables. Here is how you can directly access LED0:

```
base.led[0].on()
```

Reading Switches is extremely easy. You need to call read() function. They either return TRUE or FALSE depending upon the state of the variable. So if the SW0 reads TRUE, we turn on LED0 and LED1. If SW0

reads FALSE, we turn them off.

```

if (sw0.read() == True):    # Reads SW0 and check if it toggled
    led0.on()               # IF SW0 is ON --> Turn on LED0
    led1.on()               # IF SW0 is ON --> Turn on LED1
else:
    led0.off()              # ELSE Turn off LED0
    led1.off()              # ELSE Turn off LED1

```

Same goes for SW1.

```
while(True): # All the code below while(True) runs forever
    if (sw0.read() == True): # Reads SW0 and check if it toggled
        led0.on()           # IF SW0 is ON --> Turn on LED0
        led1.on()           # IF SW0 is ON --> Turn on LED1
    else:
        led0.off()          # ELSE Turn off LED0
        led1.off()          # ELSE Turn off LED1

    if (sw1.read() == True): # Reads SW1 and check if it toggled
        led2.on()           # IF SW1 is ON --> Turn on LED2
        led3.on()           # IF SW1 is ON --> Turn on LED3
    else:
        led2.off()          # ELSE Turn off LED2
        led3.off()          # ELSE Turn off LED3
```

- Now Run all the statements from top to bottom by either keep pressing **SHIFT + ENTER** or
- by going to Cell → Run All.

Challenge:

- Use timer and toggle function.
- Use command: `help(base)`, this should give you hints to solve the challenge
- Make sure to import the libraries and assign the LEDs first.

```
#Turn off all the LEDs
led0.off()
led1.off()
led2.off()
led3.off()

while(True):
    led0.toggle()
    led1.toggle()
    led2.toggle()
    led3.toggle()
    sleep(1)
```