



香港中文大學
The Chinese University of Hong Kong

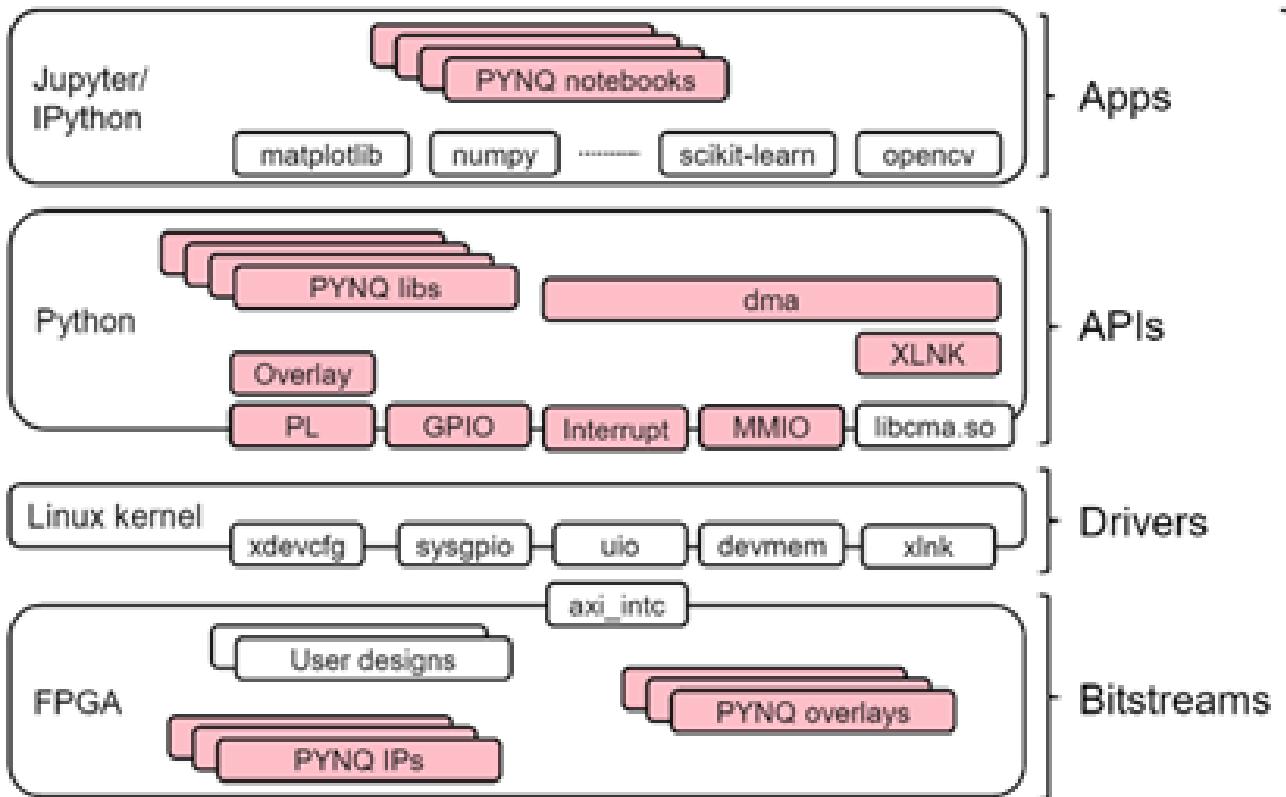
CENG3430 Rapid Prototyping of Digital Systems

Lab10: PYNQ Exercise





AXI IP Block Design Diagram





Lab10 Outline

- **PART 0: Hardware Setup**
- **PART 1: Flash Image File into SD Card**
- **PART 2: Write Python Code**
 - ① Example
 - ② Task
- **Appendix (Optional)**





Lab10 Outline

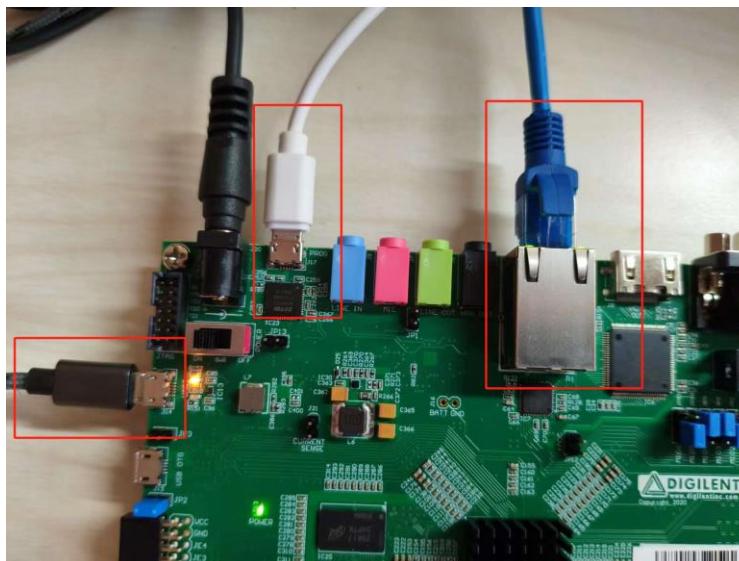
- **PART 0: Hardware Setup**
- **PART 1: Flash Image File into SD Card**
- **PART 2: Write Python Code**
 - ① Example
 - ② Task
- **Appendix (Optional)**





Hardware Setup

- In this lab, you need to connect two micro-b USB wires and the network cable to the Zedboard.
 - Connect the **JTAG port on the top**. This port is used to program the hardware bitstream.
 - Connect the **UART port under the power switch**. This port is used to program the software binary to the board and communicate with the software program.
 - Establish an **ethernet** between the computer and the Zedboard with network cable.





Lab10 Outline

- **PART 0: Hardware Setup**
- **PART 1: Flash Image File into SD Card**
- **PART 2: Write Python Code**
 - ① Example
 - ② Task
- **Appendix (Optional)**

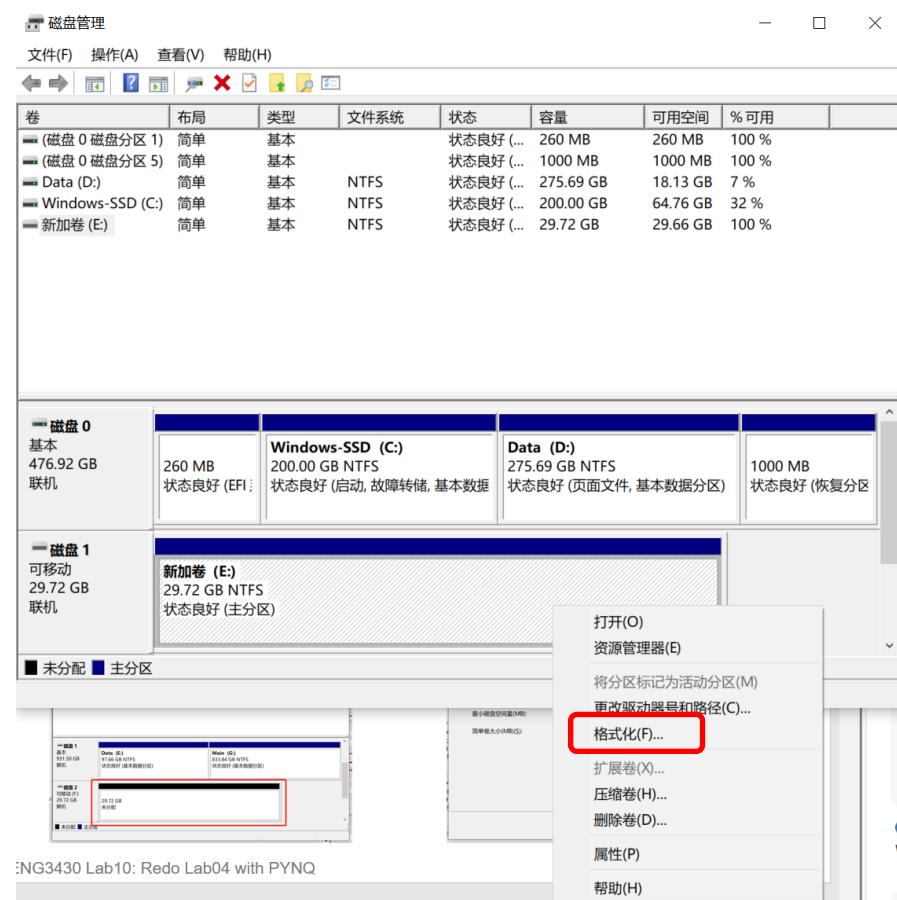


jupyter



Flash Image file into SD Card

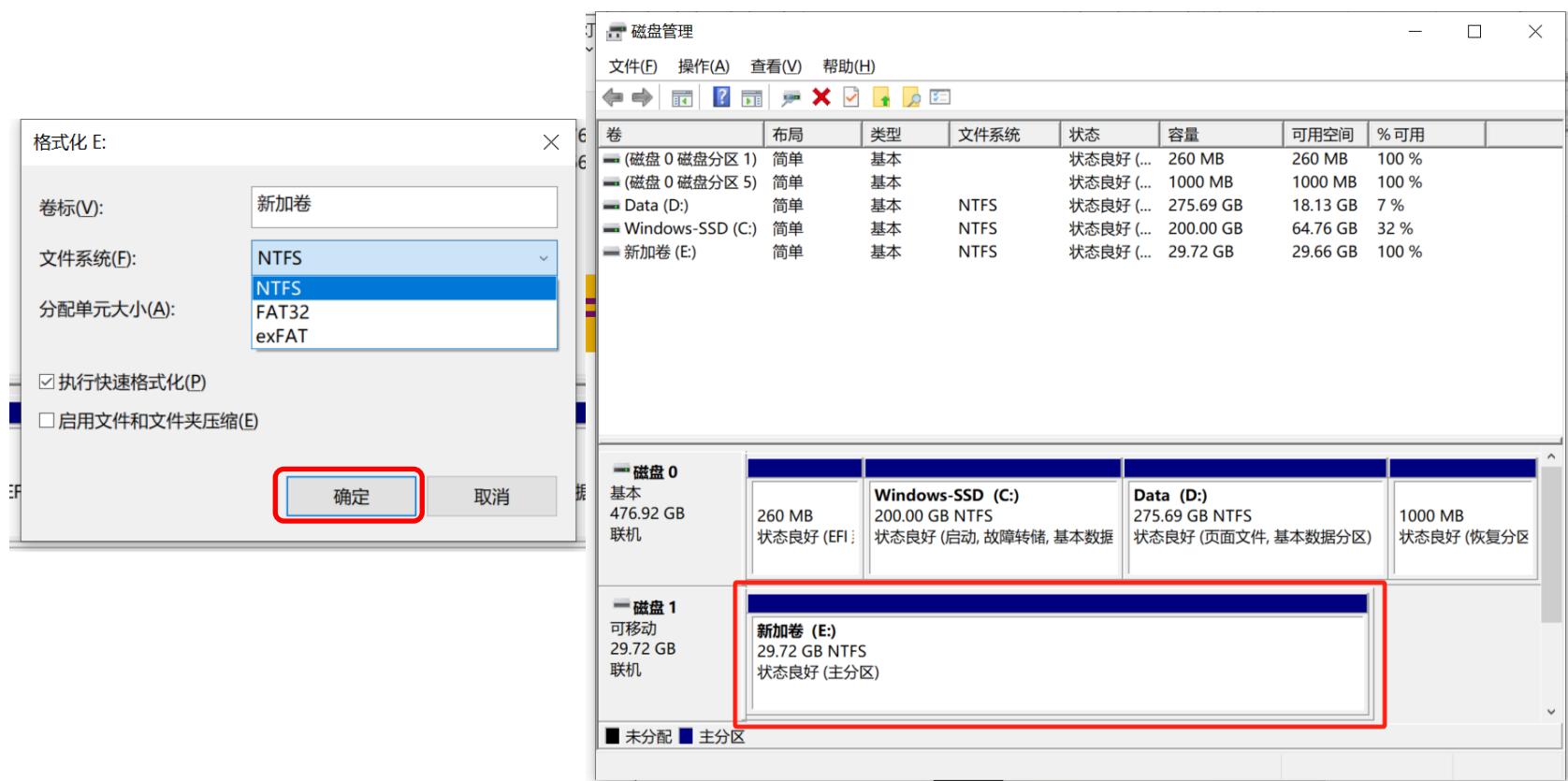
- Insert **SD card** into your computer
- Open **Disk Manager**
- Click **Format Hard Disk**





Flash Image file into SD Card

- Choose **NTFS** as file system
- Click **Confirm**
- You should find a **new volume** with almost 30GB





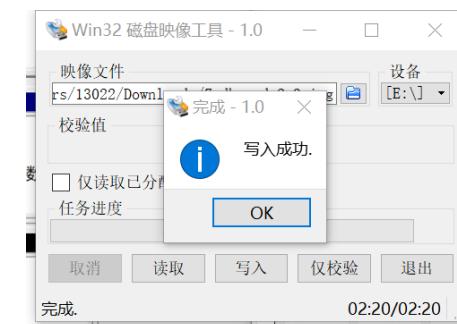
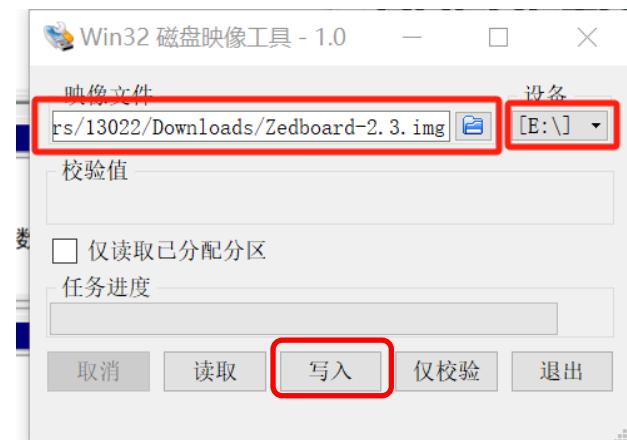
Flash Image file into SD Card

- Download and open **Win32DiskImager**
 - <https://win32diskimager.org>
- Select Zedboard-2.3.img as **image file**
- Select the new volume for SD card as **equipment**
- Click **Write** and wait for 3-5 mins

The screenshot shows a software interface for Movavi Video Editor. At the top, there's a decorative banner with a person's face and the word 'COOL!'. Below it, a large button says '立即下載' (Download Now). A black button at the bottom right says '下載' (Download). The main area has a table with the following data:

File Name	Win32DiskImager-1.0.0-install.exe
Version	1.0.0
Size	12.6 MB
Developer	Tobin Davis
Last Update	July 13, 2024

Below the table, the word 'Features' is visible.





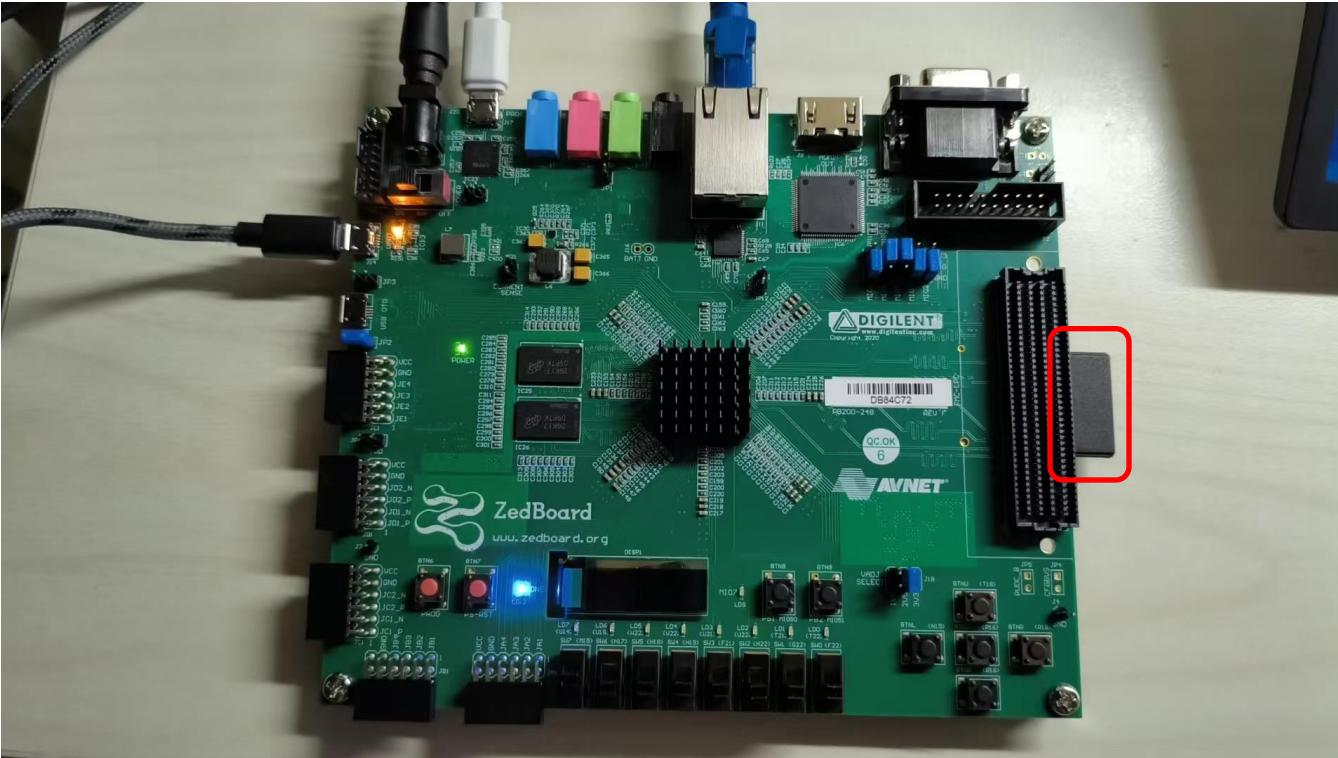
Lab10 Outline

- **PART 0: Hardware Setup**
- **PART 1: Flash Image File into SD Card**
- **PART 2: Write Python Code**
 - ① Example
 - ② Task
- **Appendix (Optional)**



Hardware Setting

- Insert the **SD card** into card slot in the Zedboard
- Off the power switch
- **LD11** and **LD12** will be turned on





Ethernet Configuration

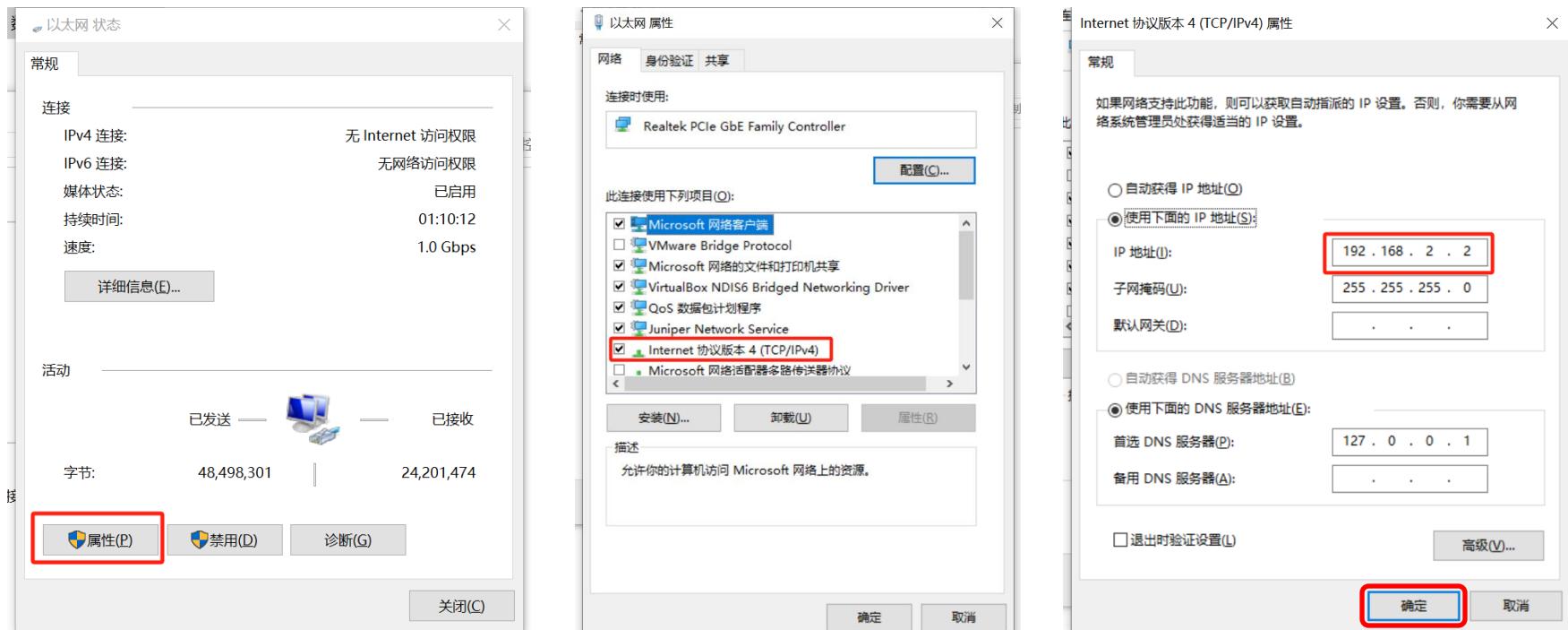
- Open Network and Sharing Center
- Select Ethernet





Ethernet Configuration

- Open Properties
- Select TCP/IPv4
- Set IP address to 192.168.2.2 and Confirm





Open Jupyter

- Go to **192.168.2.99** in your browser
- Password is **Xilinx**
- Now you can program in Python



jupyter

Logout

Files Running Clusters Nbextensions

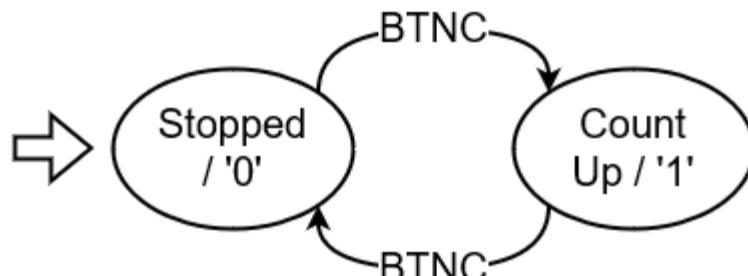
Select items to perform actions on them.

<input type="checkbox"/> 0	<input type="checkbox"/>	Name	Last Modified
<input type="checkbox"/>		23Project	6年前
<input type="checkbox"/>		base	6年前
<input type="checkbox"/>		common	6年前
<input type="checkbox"/>		getting_started	6年前
<input type="checkbox"/>		lab10test	6年前
<input type="checkbox"/>		logictools	6年前
<input type="checkbox"/>		NewProject	6年前
<input type="checkbox"/>		Welcome to Pynq.ipynb	6年前



Stopwatch Behavior

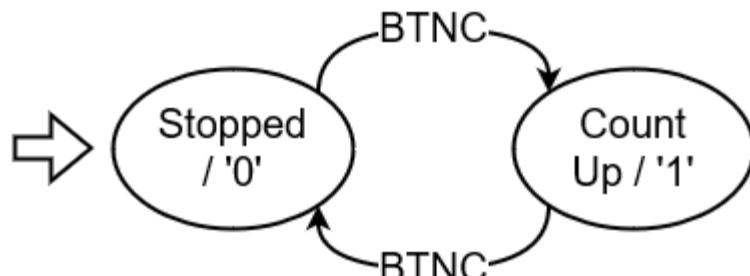
- Behavior
 - Two states
 - **Stopped State** (Initial State)
 - Select the value using **SW0 ~ 3**
 - Display the value in binary format through **LED0 ~ 3**
 - **Count-Up State**





Stopwatch Behavior

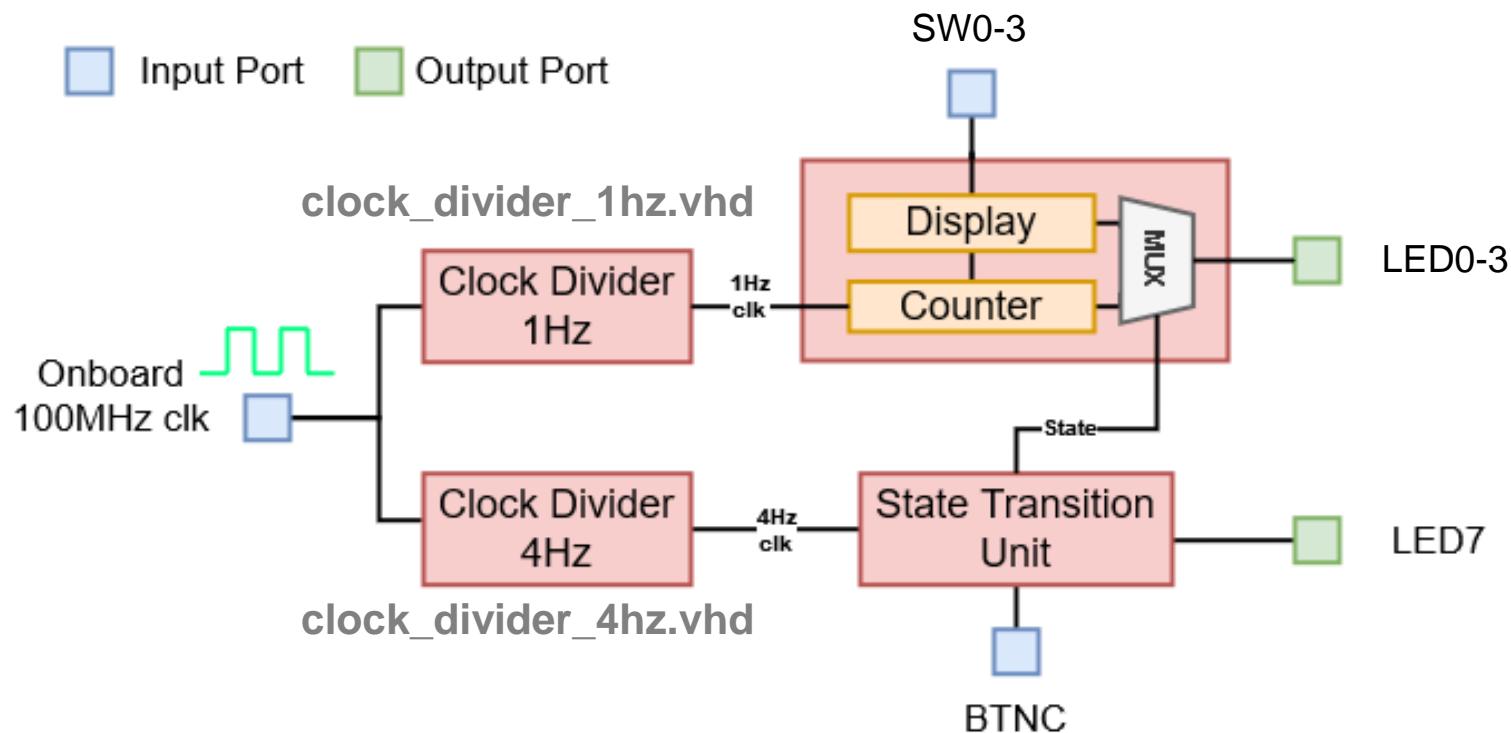
- Behavior
 - Two states
 - **Stopped State**
 - **Count-Up State**
 - Count up from the value selected in Stopped State in **1Hz**
 - If the value reaches “**1111**”, jump to “**0000**” and keep count up





Stopwatch Behavior

- Logic Schematic



lab04.vhd

Upload .bit, .hwh Files, and Ex. Code



- Download **Zedboard_AXI GPIO.bit**, **Example.ipynb**, and **Zedboard_AXI GPIO.hwh** from Blackboard
- Create a new folder and upload them
- Open **Example.ipynb**

The screenshot shows the Jupyter Notebook interface. At the top left is the logo and name "jupyter". On the right is a "Logout" button. Below the header are tabs for "Files", "Running", "Clusters", and "Nbextensions". The "Files" tab is selected. A message "Select items to perform actions on them." is displayed. In the center, there is a file list with a red box around the first three files: "Example.ipynb", "Zedboard_AXI GPIO.bit", and "Zedboard_AXI GPIO.hwh". To the right of the file list are buttons for "Upload" (which has a red box around it), "New", and a refresh icon. Below the file list, there are columns for "Name" (with a dropdown arrow) and "Last Modified" (with a dropdown arrow). The files listed are "zhanshiyong" (modified "几秒前"), "Example.ipynb" (modified "6年前"), "Zedboard_AXI GPIO.bit" (modified "6年前"), and "Zedboard_AXI GPIO.hwh" (modified "6年前").



Example Code (Part 1)

```
from pynq import Overlay
import time
class FSMController:
    def __init__(self, bitfile_path):
        self.ol = Overlay(bitfile_path)
        self.leds = self.ol.led.channel1
        self.switches = self.ol.sw.channel1
        self.btns = self.ol.btn.channel1
        self.state = "0"
        self.counter = 0
        self.last_1hz = time.time()
        self.last_4hz = time.time()
        self.last_btn_press = time.time()
        self.btn_pressed = False
        self.update_leds()

    def update_leds(self):
        state_val = int(self.state, 2) << 7 #State Position
        count_val = self.counter & 0x0F #0-3 led
        led_value = state_val | count_val
        self.leds.write(led_value, 0xFF)
        print(f"{bin(led_value)[2:]}")

    def read_inputs(self):
        current_time = time.time()
        self.last_btn_press = current_time
        btn_val = self.btns.read()
        return {
            'din': self.switches.read() & 0x0F,
            'btn_c': (btn_val & 0x01) == 0x01, # bit0
        }
```

Declare Overlay

Set I/O Signals

Set Time and State-transition Variables

Just as Embedded System
😊 Programmer Friendly

Update led output with xxx.write()

Read sw and btn input with xxx.read()



Example Code (Part 2)

```
def run(self):
    while True:
        current_time = time.time()
        inputs = self.read_inputs()
        if current_time - self.last_4hz >= 0.25:
            self.last_4hz = current_time
            btn_c = inputs['btn_c']
            if btn_c and not self.btn_pressed:
                if self.state == "0":
                    self.state = "1"
                elif self.state == "1":
                    self.state = "0"
                self.btn_pressed = True
            elif not btn_c:
                self.btn_pressed = False
        if self.state == "0":
            self.counter = inputs['din']
        elif self.state == "1":
            if current_time - self.last_1hz >= 1.0:
                self.last_1hz = current_time
                if self.counter < 15:
                    self.counter += 1
                else:
                    self.counter = 0
            self.update_leds()
            time.sleep(0.5)

if __name__ == "__main__":
    controller = FSMController("./design_1.bit")
    controller.run()
```

Check BtnC for every 0.25 sec.

State Transition

Avoid Oscillation

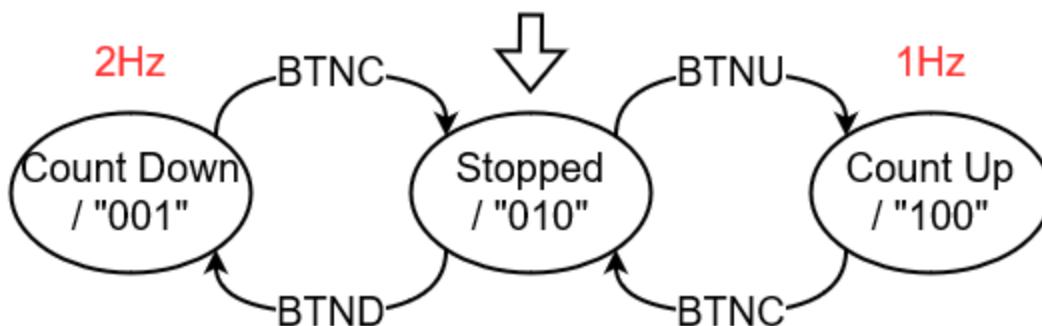
Behavior

Declare .bit file



Three-state Stopwatch

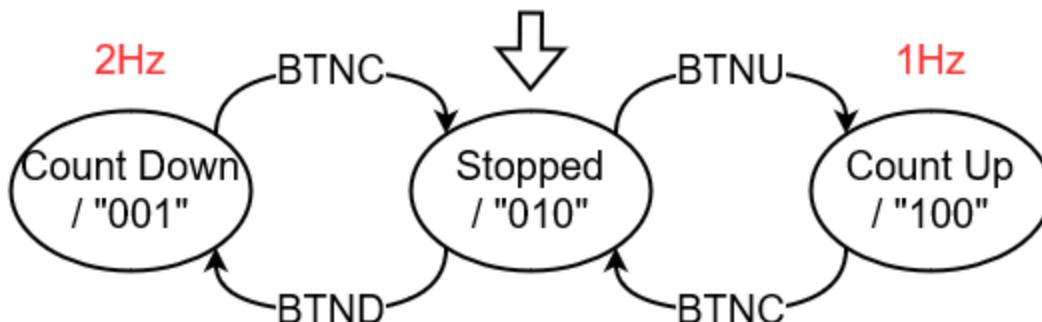
- Compare to the two-state stopwatch in example
 - **Stopped state (“010”)** is the same
 - **Count Up state (“100”)** is no longer loop back
 - If the counter reaches “1111”, stops at “1111”
 - **Count Down state (“001”)**
 - Count down the value selected by the stopped state in 2Hz
 - If the counter reaches “0000”, stops at “0000”





Three-state Stopwatch

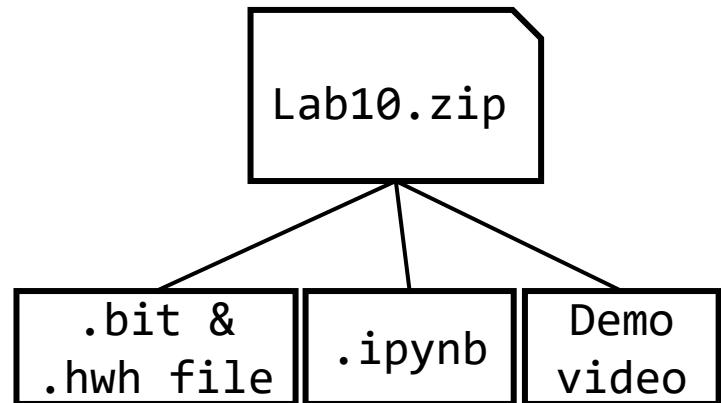
- Compare to the two-state stopwatch in example
 - LED 5-7 for showing the new state values ("001", "010", "100")
 - BTNU and BTND are added for state transition
 - BTNC will no longer switch back from Count Up state to Stopped state
 - Non-specified transition will create no effects
 - E.g., Pressing BTND in Count Up state has no effect





Submission Guide

- Following the question specification to implement the **Three-State Stopwatch**
- Demo video instruction
 1. Select 1010b and press **BTNU**
 2. Wait until 1111b and press **BTND**
 3. Then press **BTNC**
 4. Select 1000b and press **BTND**
 5. Wait until 0000b and press **BTNU**
 6. Finally press **BTNC**
- Submit the zip file according to the figure
 - The video should **clearly demonstrate LEDs, SWs and BTNs at the same time**
 - Deadline: **12:30 on 9 Apr. 2025**
 - Late submission is NOT acceptable.





香港中文大學
The Chinese University of Hong Kong

Thank You!





Lab10 Outline

- **PART 0: Hardware Setup**
- **PART 1: Flash Image File into SD Card**
- **PART 2: Write Python Code**
 - ① Example
 - ② Task
- **Appendix (Optional)**





Lab10 Appendix Outline

- **PART 0: VM Setting**
- **PART 1: Create PYNQ Image**



- ① SDSoc Configuration
- ② Petalinux Configuration
- ③ Generate Image

- **PART 2: Create PYNQ Overlay**



- ⑤ IP Integration
- ⑥ HDL Wrapper & Generate Bitstream



VM Setting

- If you already have the Linux 16.04 environment on your computer, ignore this section.

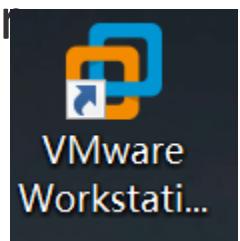
Objective: Create Linux environment for **Part 2**

Output: **Linux 16.04** Environment



VM Setting

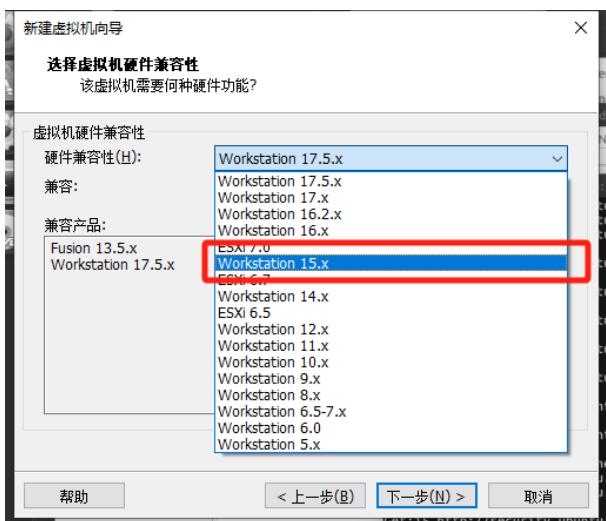
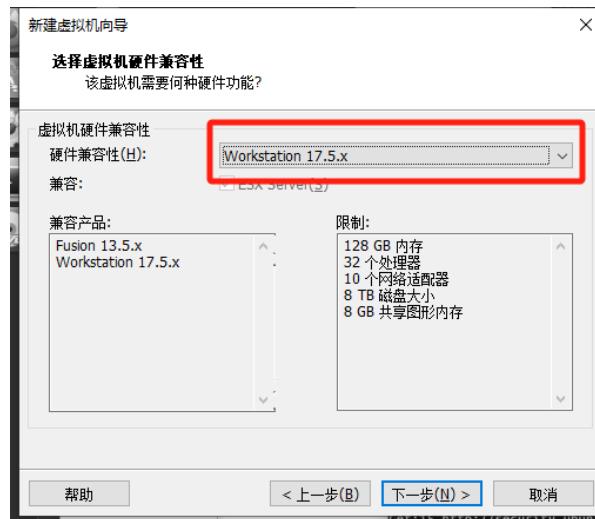
- Download **VMWare Workstation** and **ubuntu-16.04.7-desktop-amd64.iso**
 - www.vmware.com/products/desktop-hypervisor/workstation-and-fusion
 - <https://releases.ubuntu.com/16.04/>
- Open **VMWare** and Create a New Virtual Machine
 - Before starting this step, you need to make sure your computer has 80GB of free storage.





VM Setting

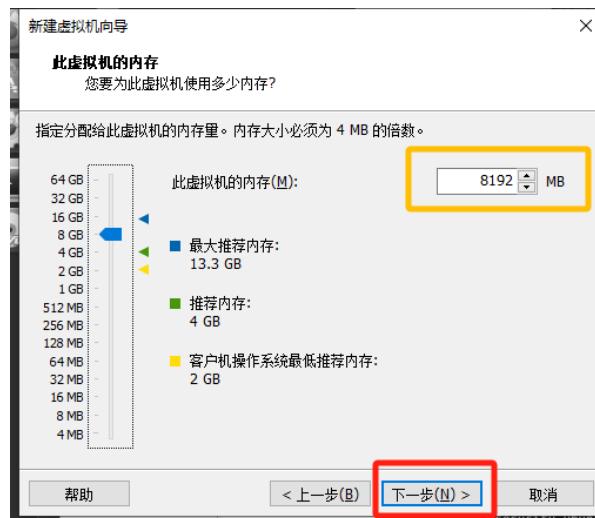
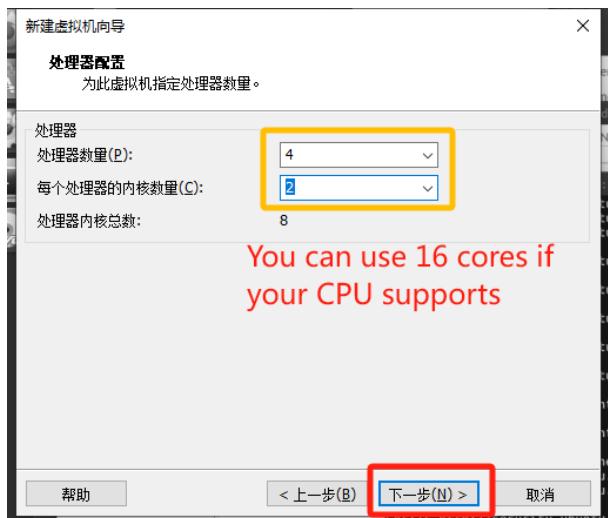
- Follow the Instruction in Following Slides





VM Setting

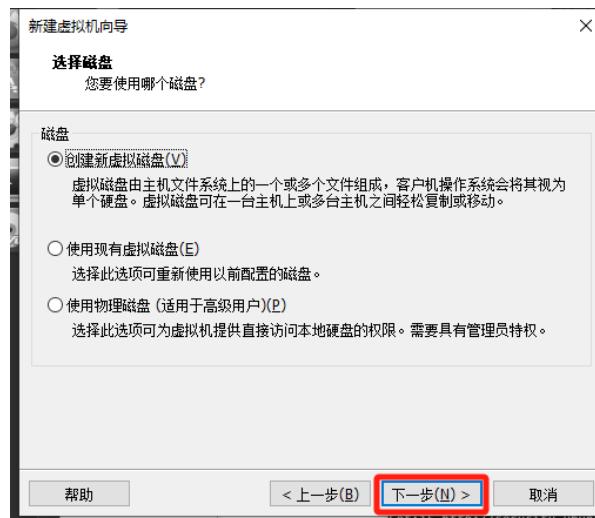
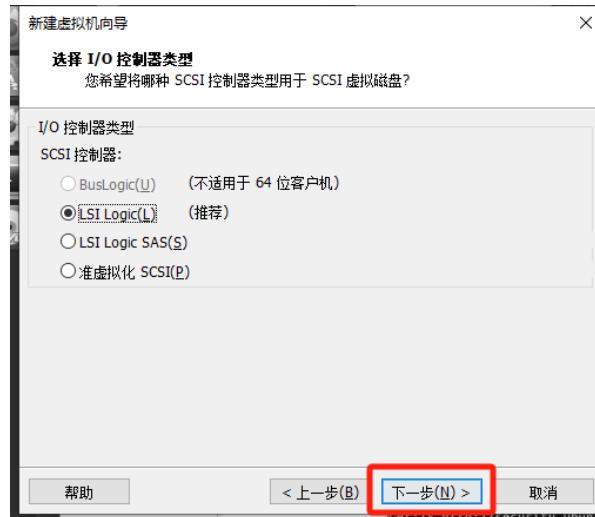
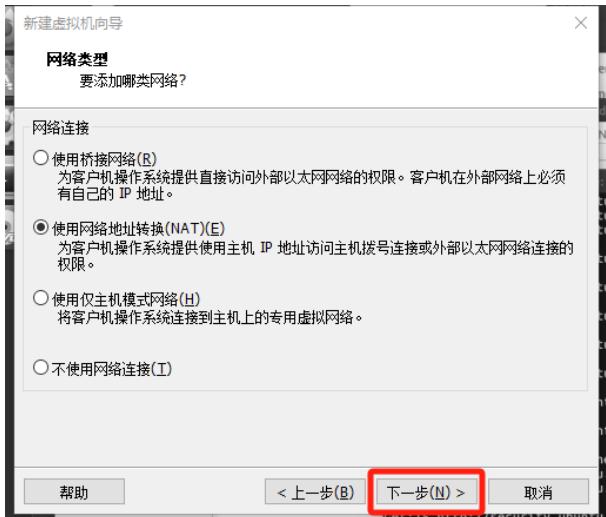
- Follow the Instruction in Following Slides





VM Setting

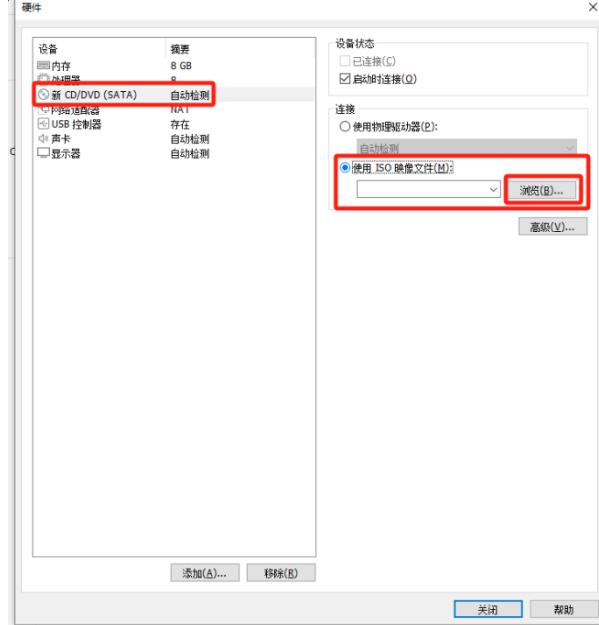
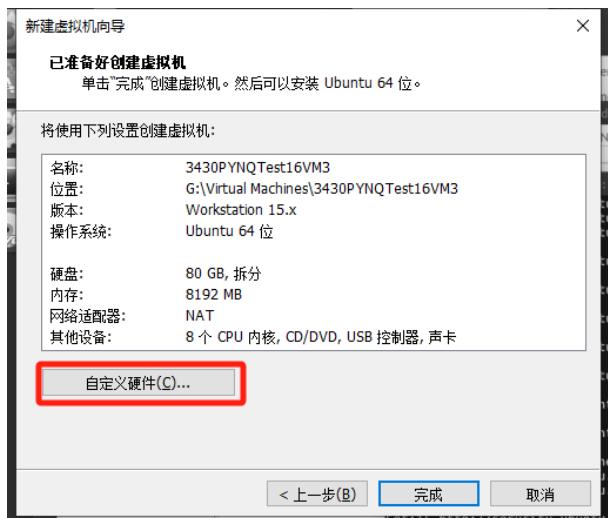
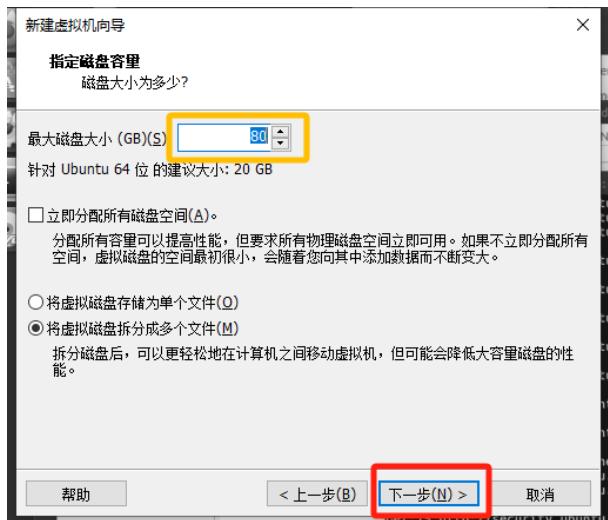
- Follow the Instruction in Following Slides





VM Setting

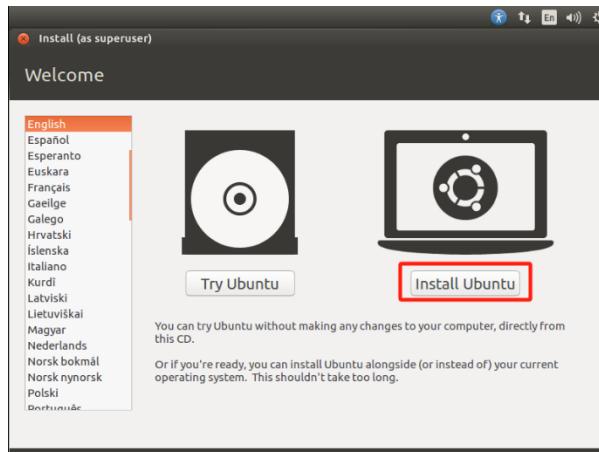
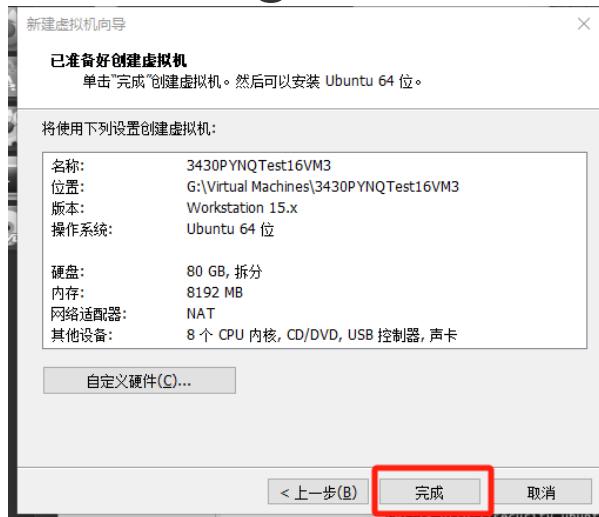
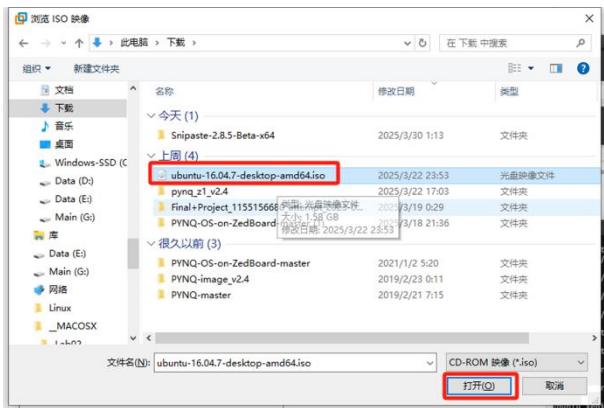
- Follow the Instruction in Following Slides





VM Setting

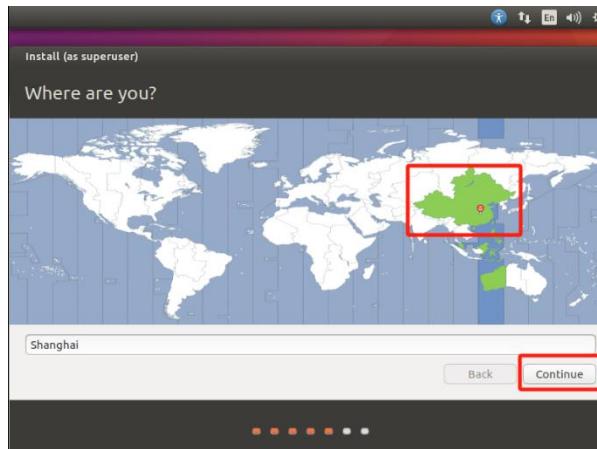
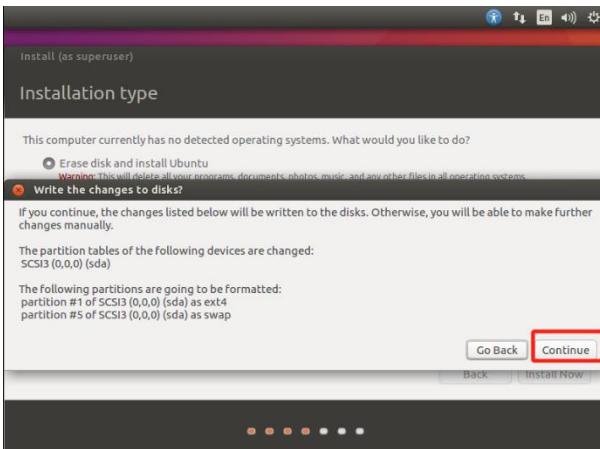
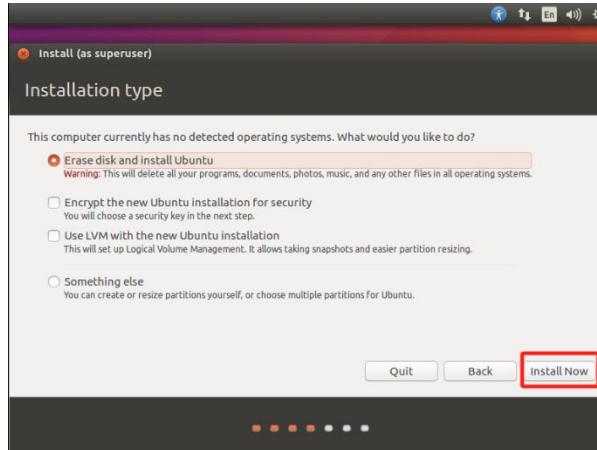
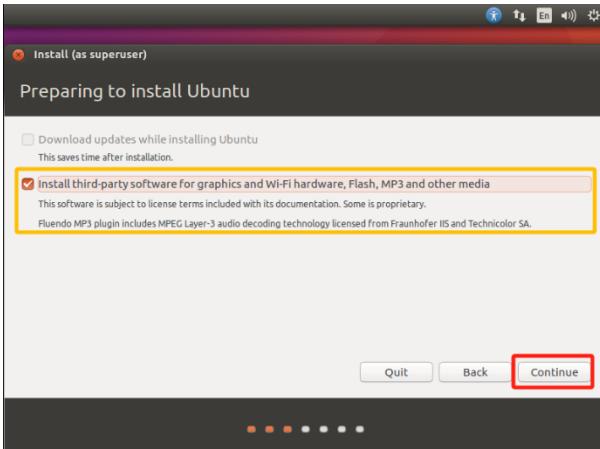
- Follow the Instruction in Following Slides





VM Setting

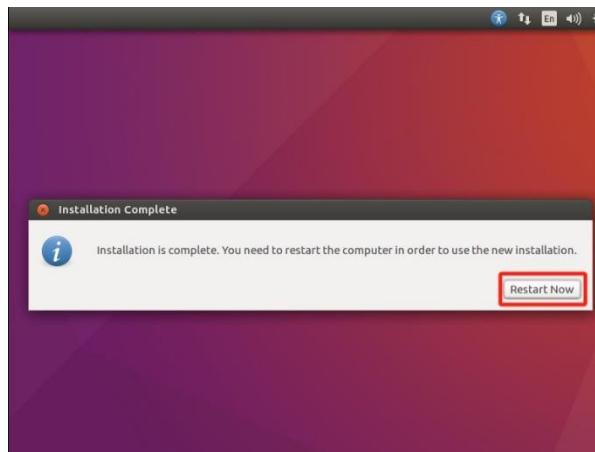
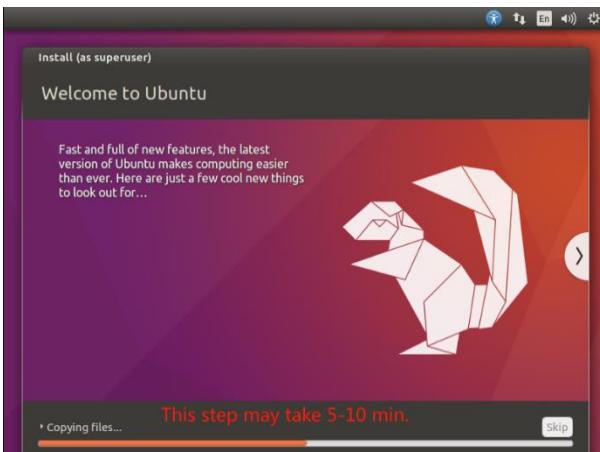
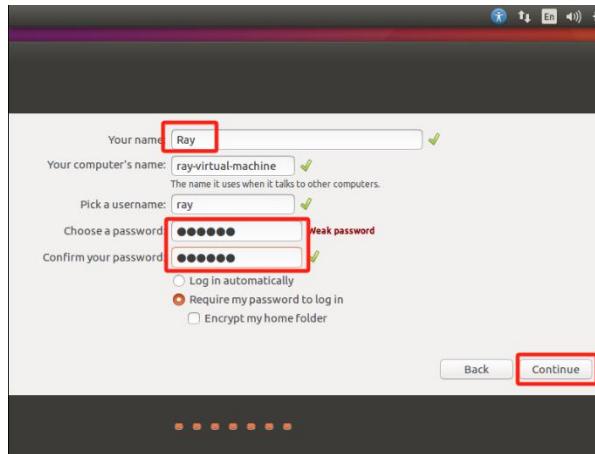
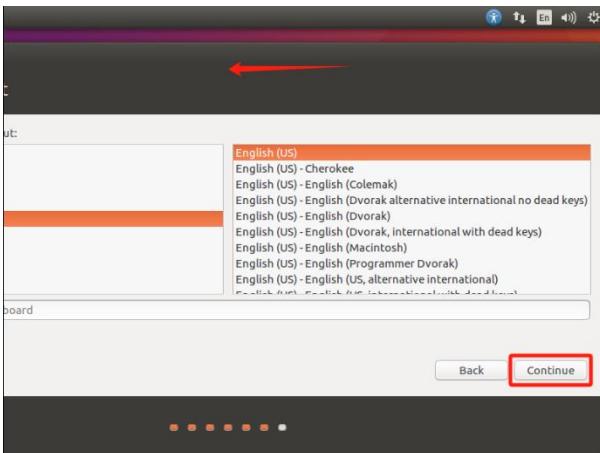
- Follow the Instruction in Following Slides





VM Setting

- Follow the Instruction in Following Slides





Create PYNQ Image

Objective: Support APIs, Drivers, and Apps

Output: **Zedboard_v2.3.img**



Download Required Documents

- Download **SDSoC 2018.2 for Linux** from AMD
 - www.xilinx.com/support/download.html/content/xilinx/en/downloadNav/vitis/archive-sdsoc.html

The screenshot shows the Xilinx support website's download menu. The top navigation bar includes 'Licensing Help', 'NIC Software & Drivers' (selected), and 'Alveo Packages'. The main menu items are 'Vivado (HW Developer)', 'Vitis (SW Developer)' (highlighted with a red box), 'Vitis Embedded Platforms', 'PetaLinux', 'Power Design Manager', 'Device Models', and 'Documentation Navigator'. Below this, a 'Version' dropdown shows options: '2024.2', '2024.1', '2023.2', 'Vitis Archive' (highlighted with a yellow box), 'SDAccel Archive', 'SDK/PetaLinux Archive' (highlighted with a yellow box). The 'Vitis Archive' section lists versions: '2019.1', '2018' (highlighted with a red box), '2018.3', '2018.2' (highlighted with a red box), '2018.1', '2017', and '2017.4'. A 'Feedback' button is located at the bottom right.

The screenshot shows the 'SDSoC - 2018.2 Full Product Installation' page. It features an 'IMPORTANT' section with a note about using web installers for better performance. It also notes that download verification is supported only with Google Chrome and Microsoft Internet Explorer. Two download links are provided: 'SDSoC 2018.2 web installer for Windows 64 (EXE - 50.58 MB)' and 'SDSoC 2018.2 web installer for Linux 64 (BIN - 99.48 MB)' (highlighted with a red box). A 'MD5 SUM Value' for each is listed. A 'Download Verification' section includes 'Digests', 'Signature', and 'Public Key' buttons. At the bottom, there's a link to 'SDx 2018.2 SFD (TAR/GZIP - 20.35 GB)' with its own MD5 sum value.



Download Required Documents

- Download **Petalinux 2018.2** and **.bsp** file for **Zedboard** from AMD
 - www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vitis/archive-sdk.html

Licensing Help

NIC Software & Drivers

Alveo Packages

Vivado (HW Developer)

Vitis (SW Developer)

Vitis Embedded Platforms

PetaLinux

Power Design Manager

Device Models

Documentation Navigator

Version

2024.2 Starting 2019.2, Xilinx SDK development environments are unified into an all-in-one Vitis™ unified software platform. There will be no 2019.2 or future releases of Xilinx SDK.

2024.1

2023.2

Vitis Archive

SDSoC Archive

SDAccel Archive

SDK/PetaLinux Archive

2019.1

2018

2018.3

2018.2

2018.1

2017

2017.4

Feedback

PetaLinux 2018.2 Open Components Source Code (TAR/GZIP - 6.09GB)
MD5 SUM Value : ae55e25d33967c61aeb5485b40f90650

PetaLinux 2018.2 Installer (TAR/GZIP - 6.15GB)
MD5 SUM Value : 686edec30123bacf94102f2bc6ed70ff

ZC702 BSP (BSP - 105.1 MB)
MD5 SUM Value : 9e763f30c038a2b798b89381439578ea

ZC706 BSP (BSP - 106.72 MB)
MD5 SUM Value : ae8a6911c2cee537cb376b9eab8c08ca

ZED BSP (BSP - 106.97 MB)
MD5 SUM Value : 4c549acdd7308b4a157fe47f1b9c51a0

AC701 BSP (BSP - 570.06 MB)
MD5 SUM Value : 9ba946f82219f73d49fe5afffc4c3327d

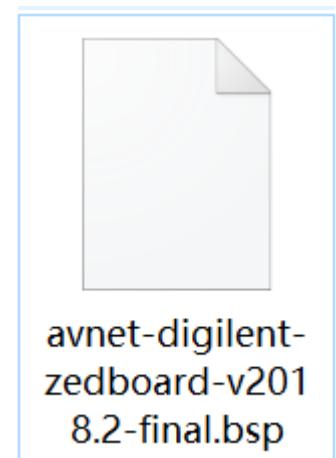
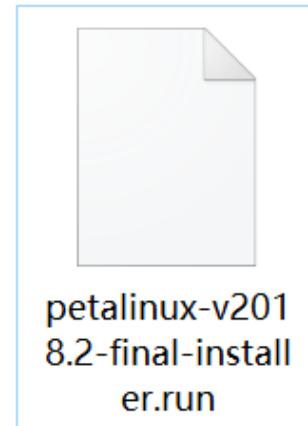
Feedback



Download Required Documents

- Download **PYNQ image source code v2.3**
 - https://github.com/Xilinx/PYNQ/tree/image_v2.3

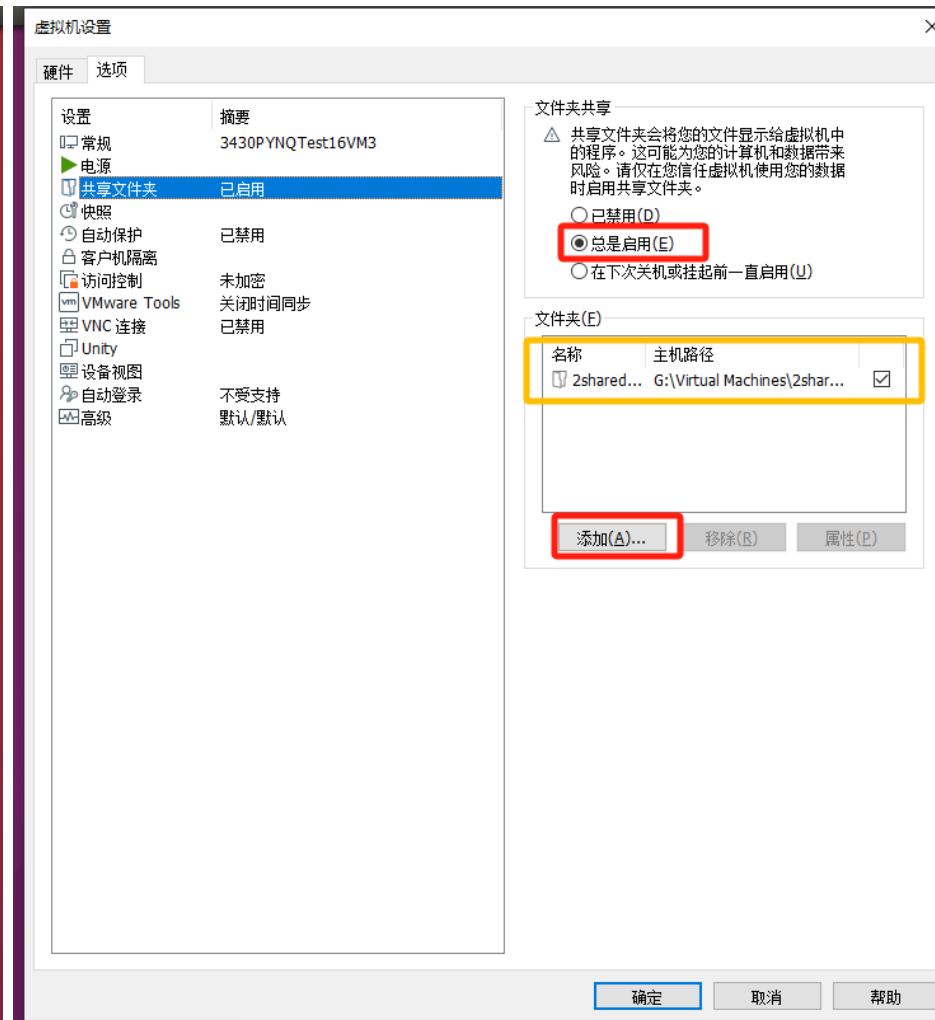
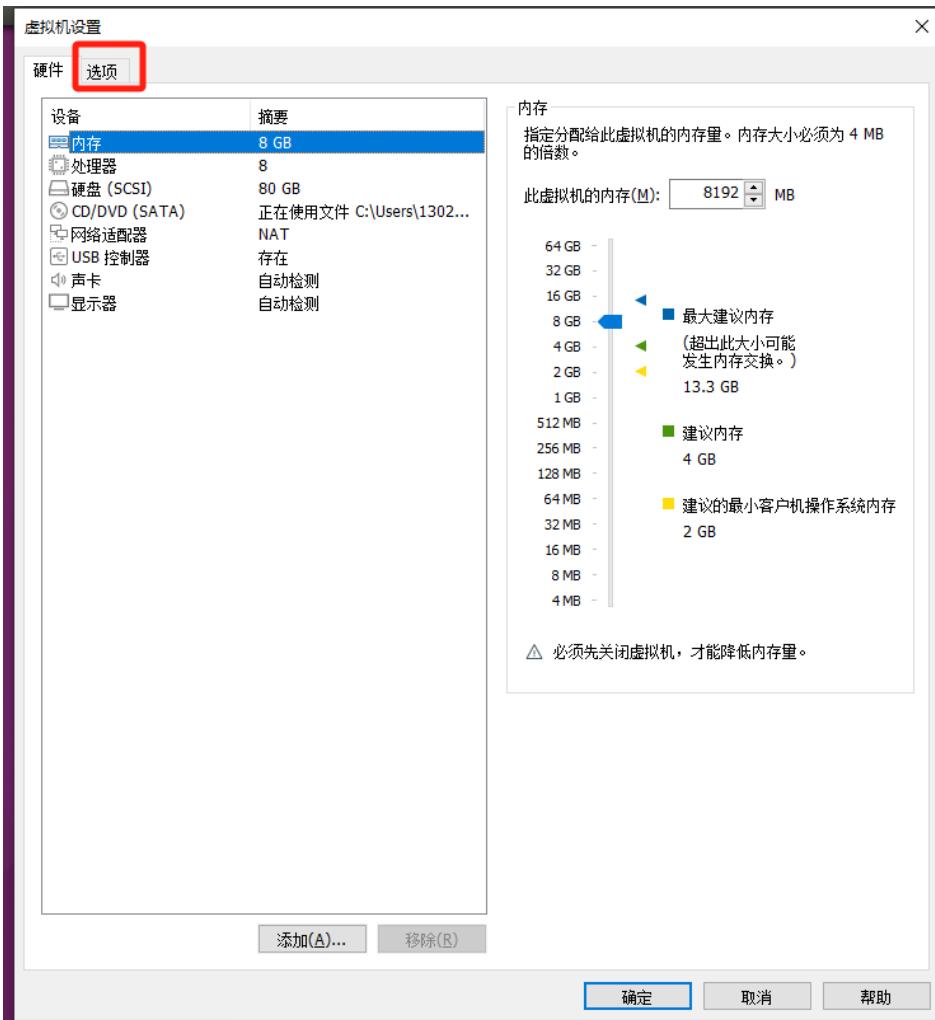
The screenshot shows the GitHub repository page for 'PYNQ' under the 'Xilinx' organization. The 'image_v2.3' branch is selected. On the left, there's a file tree showing various files like '.gitattributes', '.gitignore', 'CONTRIBUTING.md', 'LICENSE', 'README.md', etc. On the right, there's an 'About' section with links to 'www.pynq.io/' and 'pynq'. Below that are sections for 'Readme', 'BSD-3-Clause license', 'Activity', 'Custom properties', '2.1k stars', '132 watching', '832 forks', and 'Report repository'. Under 'Releases', there's a link to 'Belfast - bugfix release (Latest) on Oct 27, 2022'. The 'Packages' section indicates 'No packages published'. The 'Contributors' section shows 66 contributors with small profile icons.





Install and Configure SDSoC

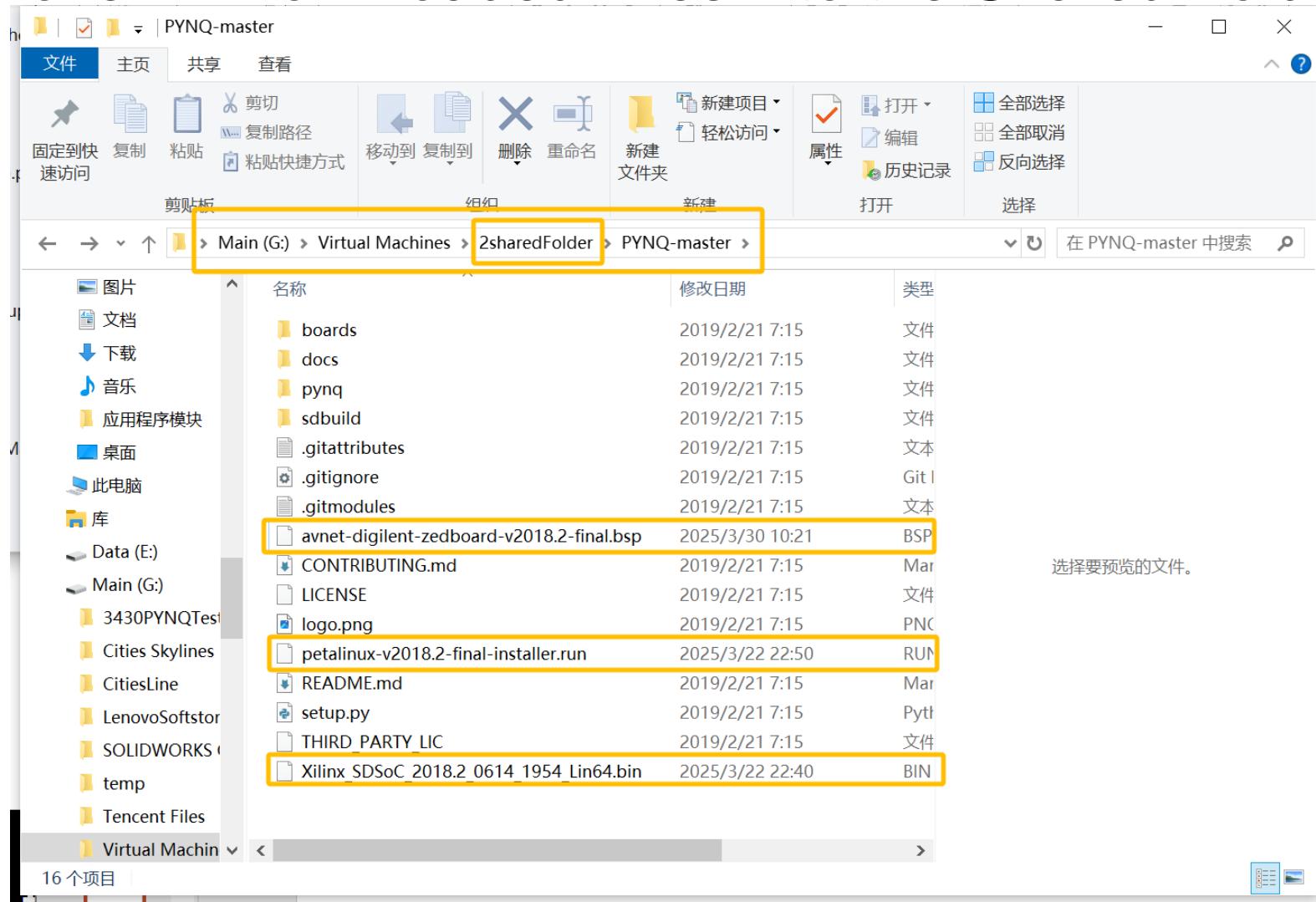
- Create a Shared Folder in VMware setting





Install and Configure SDSoC

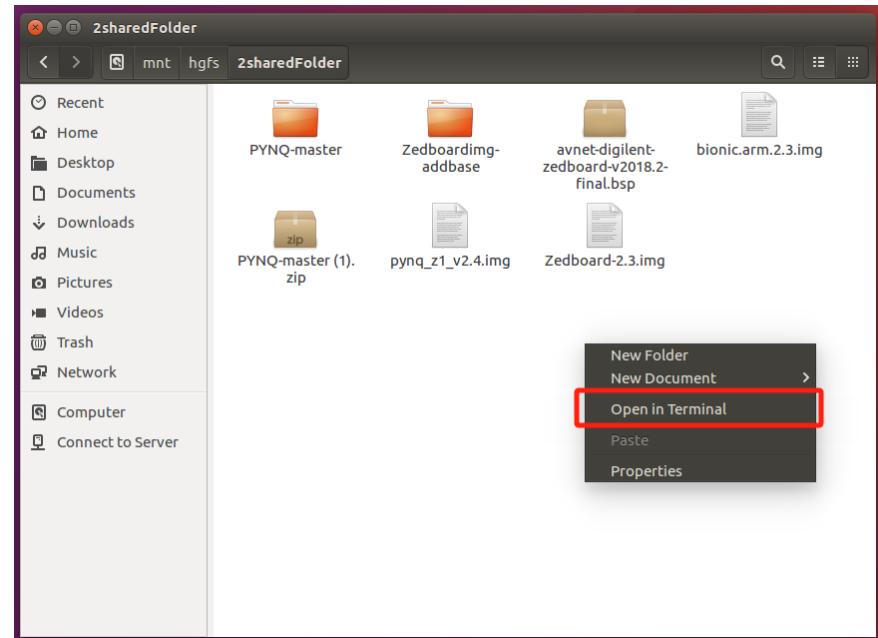
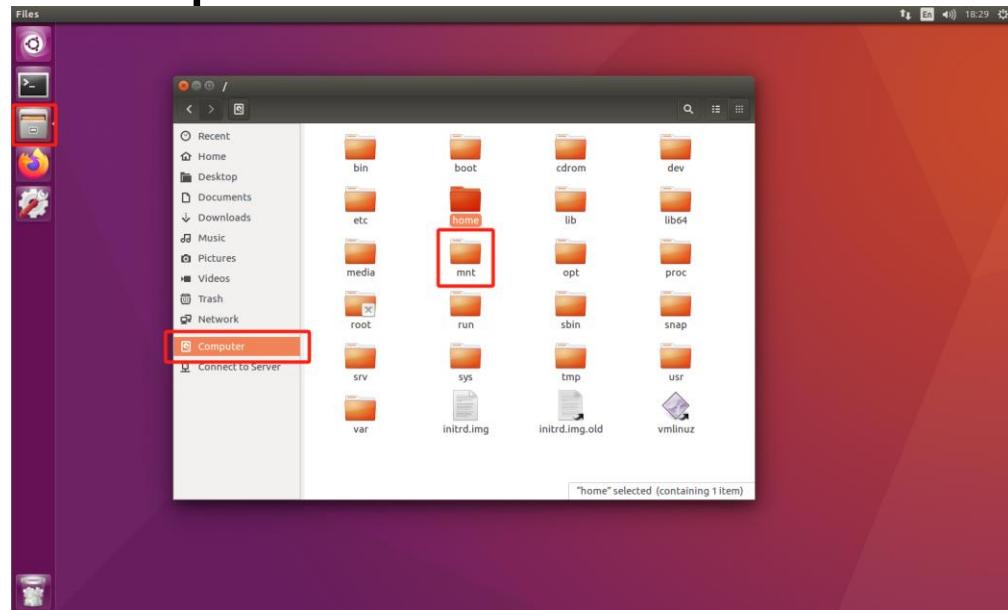
- Move All Downloaded Files into the Shared Folder





Install and Configure SDSoC

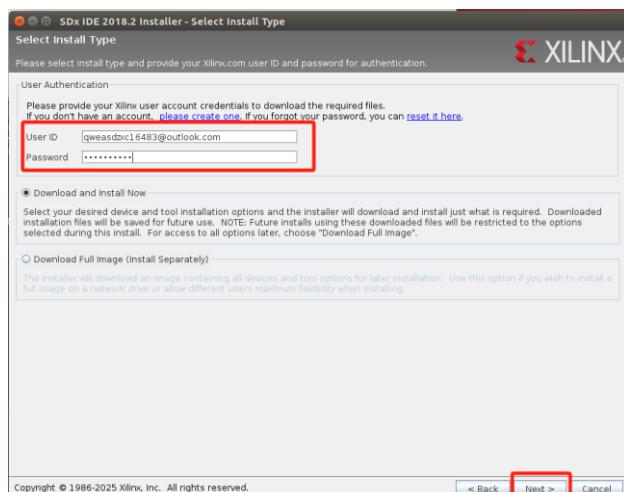
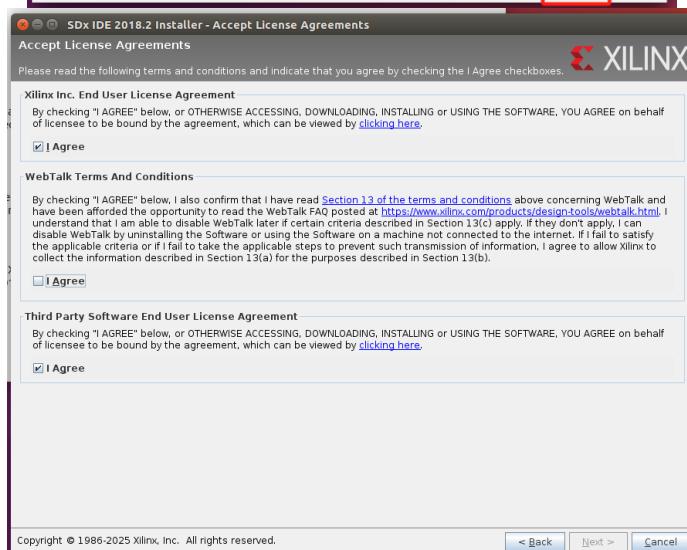
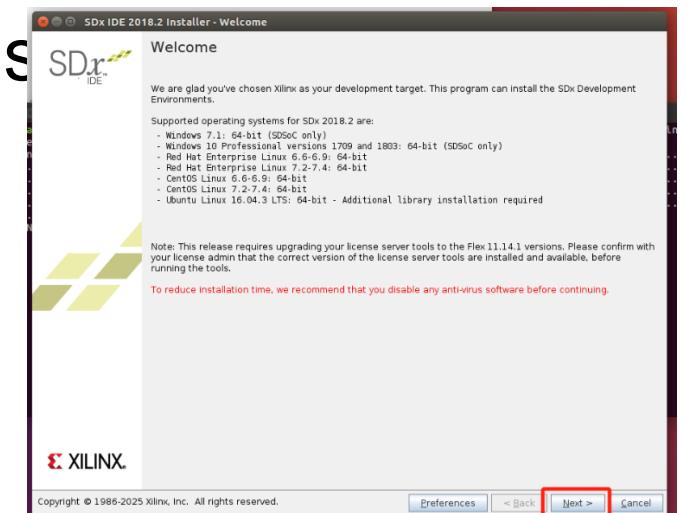
- Open the Shared Folder and Terminal in the VM





Install and Configure SDSoc

- Input ./Xilinx_SDSoc_2018.2_0614_1954_Lin64.bin
- Follow the default instructions





Install and Configure SDSoC

- Follow the default instructions

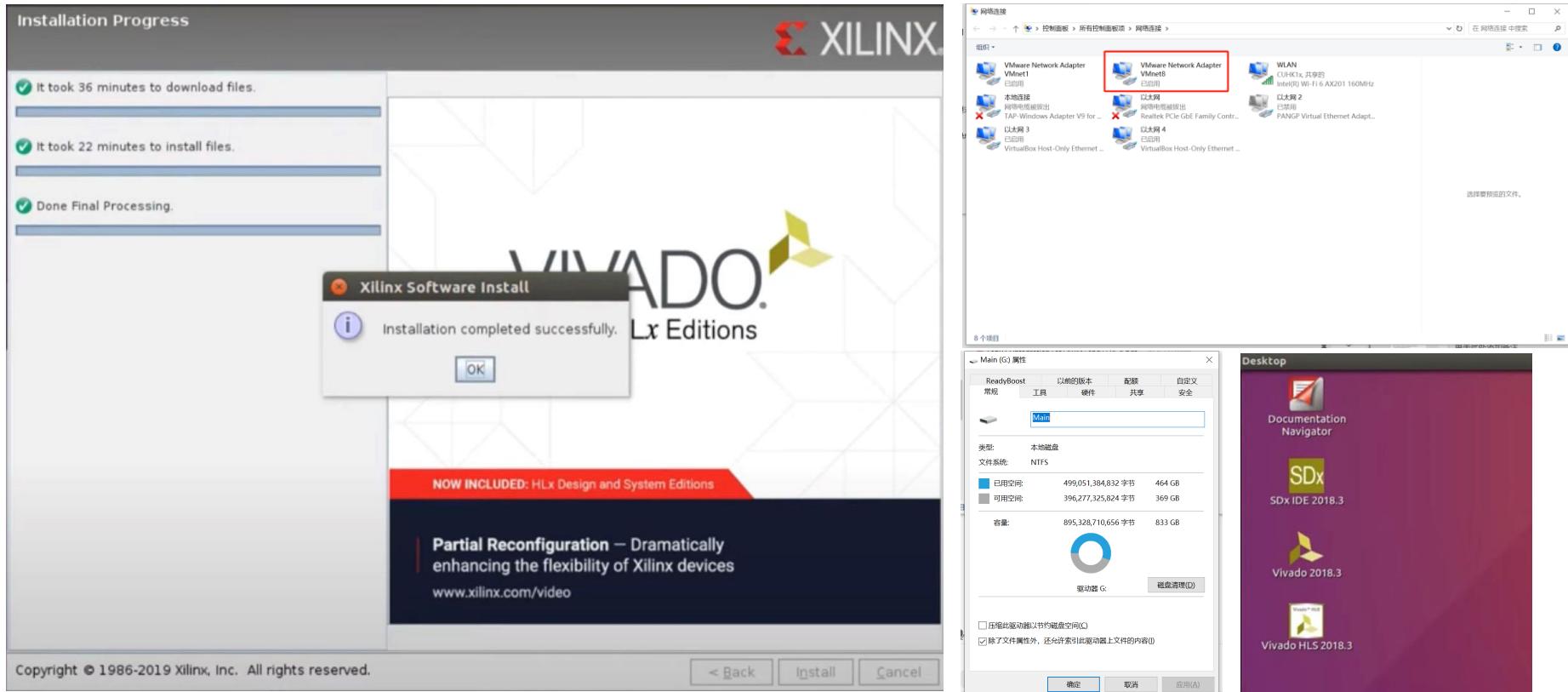
The screenshots illustrate the steps of the SDx IDE 2018.2 Installer:

- SDx Development Environments**: Shows the main configuration screen with various options like Design Tools, Devices, and Installation Options. A note at the bottom says "Cannot write to /opt/Xilinx. Check the read/write permissions." The "Next >" button is highlighted.
- Select Destination Directory**: Shows the destination directory set to "/opt/Xilinx". It includes options for creating program group entries and desktop shortcuts. A note at the bottom says "Initialize a new folder in PYNQ-image folder". The "Next >" button is highlighted.
- Select Destination Directory**: Shows the destination directory set to "/mnt/hgfs/2sharedFolder/PYNQ-master/sdbuild". It includes options for creating program group entries and desktop shortcuts. A note at the bottom says "Cannot write to /opt/Xilinx. Check the read/write permissions." The "Next >" button is highlighted.
- Installation Summary**: Summarizes the selected configurations:
 - Devices**: Built-in Platforms and associated devices for SDSoC; Devices for Custom Platforms (SoCs)
 - Design Tools**: Software Defined Development Environment (SDx) IDE for SDSoC and SDAccel (Vivado, Software Development, DocNav)
 - Installation Options**: Enable WebTalk for SDK to send usage statistics to Xilinx; Enable WebTalk for Vivado to send usage statistics to Xilinx (Always enabled for WebPACK license); Acquire or Manage a License Key
 - Installation location**: /mnt/hgfs/2sharedFolder/PYNQ-master/sdbuild/SDx_2018.2
 - Download location**: /mnt/hgfs/2sharedFolder/PYNQ-master/sdbuild/Downloads/SDx_2018.2
 - Disk Space Required**: Download Size: 10.29 GB, Disk Space Required: 50.33 GB, Disk Space Available: 374.84 GBThe "Install" button is highlighted.



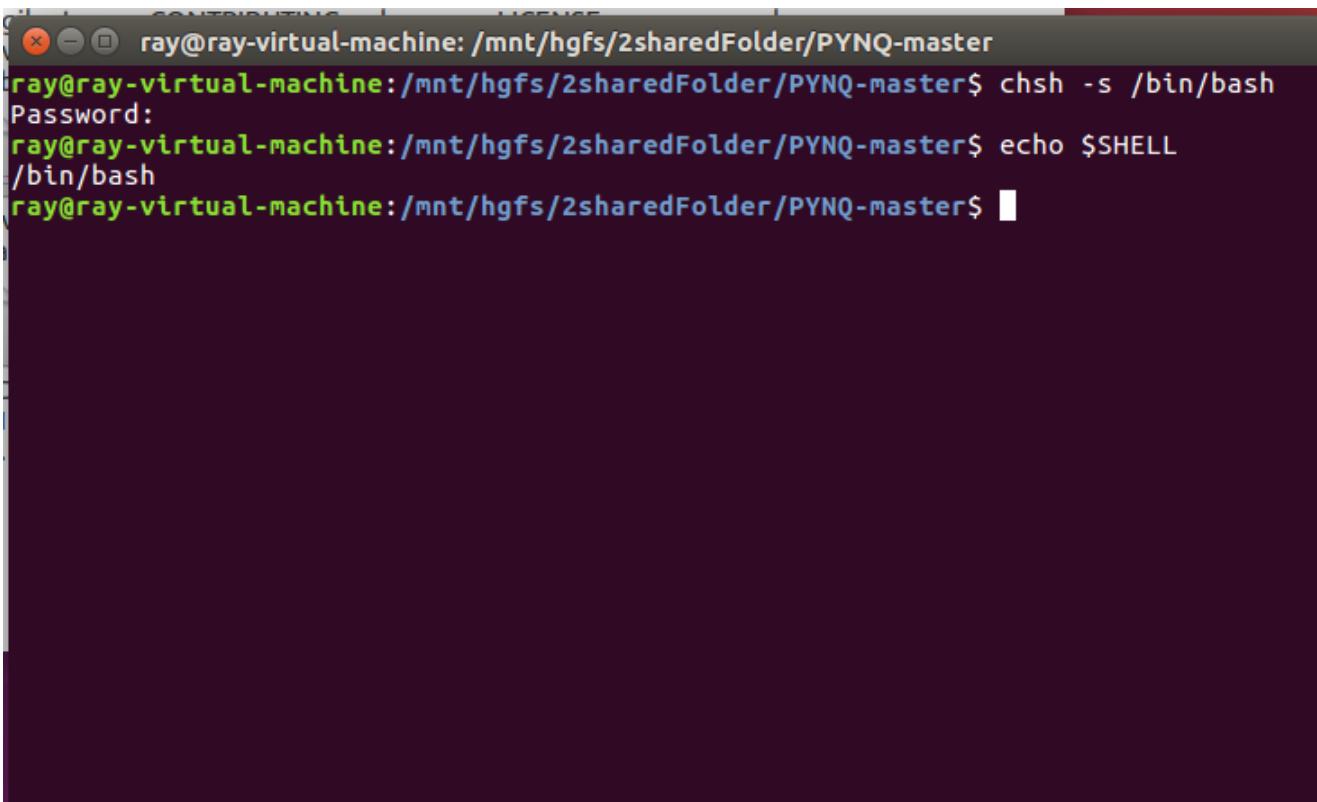
Install and Configure SDSoC

- The installation may take more than 15 min.
 - If you encountered unknown errors, you can check
 - Ethernet connection for NMVet8
 - Rest disk space in both host and virtual machine



Install Dependencies for Petalinux

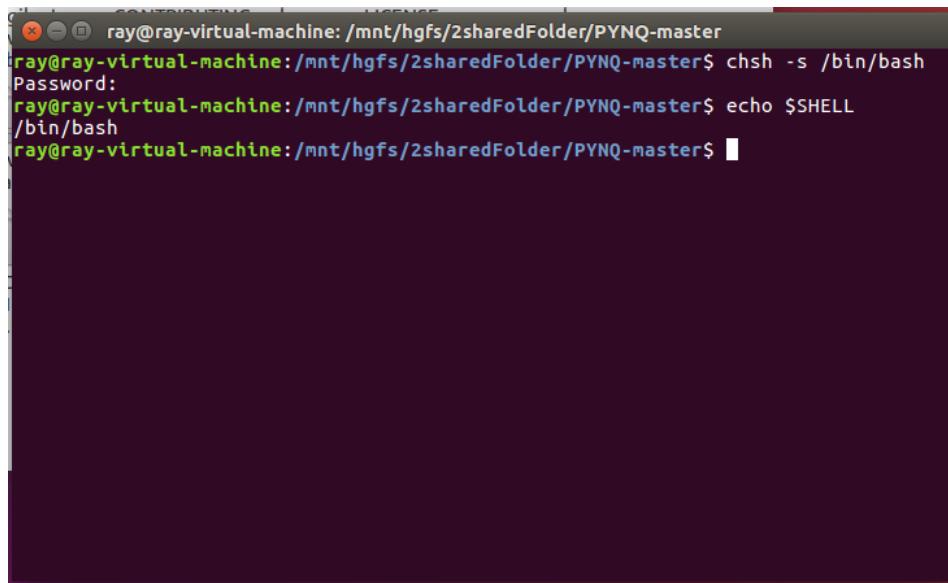
- Run **chsh -s /bin/bash** and restart the VM
 - **echo \$SHELL** can be used to check if Shell has been changed to bash
 - The output should be */bin/bash*



```
ray@ray-virtual-machine: /mnt/hgfs/2sharedFolder/PYNQ-master
ray@ray-virtual-machine:/mnt/hgfs/2sharedFolder/PYNQ-master$ chsh -s /bin/bash
Password:
ray@ray-virtual-machine:/mnt/hgfs/2sharedFolder/PYNQ-master$ echo $SHELL
/bin/bash
ray@ray-virtual-machine:/mnt/hgfs/2sharedFolder/PYNQ-master$ █
```

Install Dependencies for Petalinux

- Run **chsh -s /bin/bash** and restart the VM
 - **echo \$SHELL** can be used to check if Shell has been changed to bash
 - The output should be */bin/bash*



```
ray@ray-virtual-machine:~/mnt/hgfs/2sharedFolder/PYNQ-master$ chsh -s /bin/bash
Password:
ray@ray-virtual-machine:~/mnt/hgfs/2sharedFolder/PYNQ-master$ echo $SHELL
/bin/bash
ray@ray-virtual-machine:~/mnt/hgfs/2sharedFolder/PYNQ-master$
```

- This step ensures that the system is using **Bash** as the shell, which is commonly used in development environments due to its compatibility and features.

Install Dependencies for Petalinux

- Run sudo apt-get update and sudo apt-get upgrade

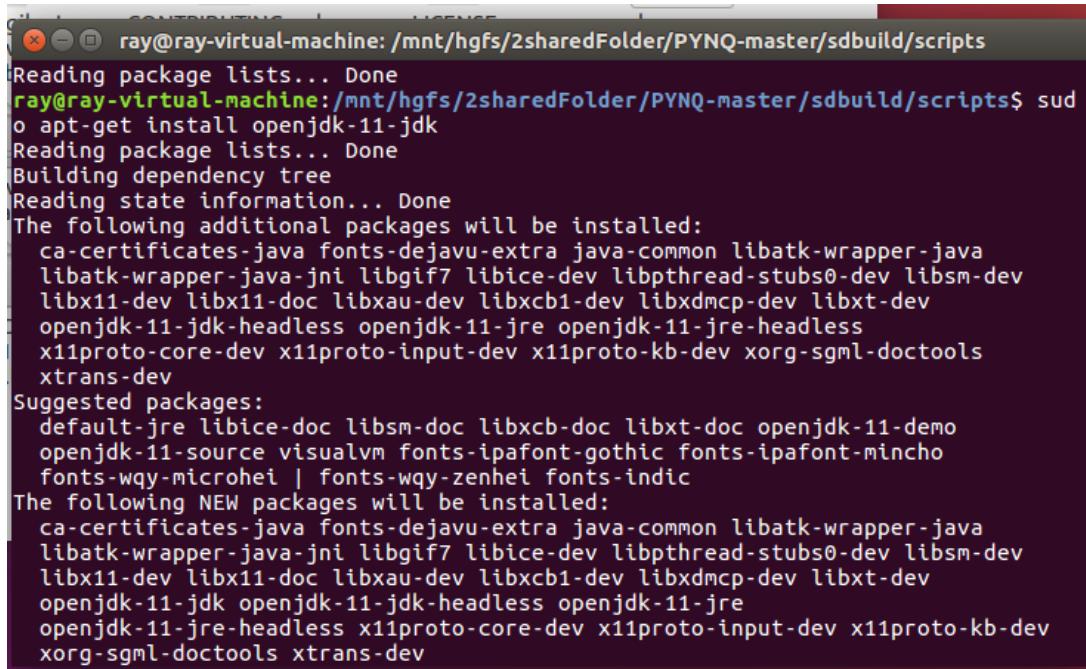
```
ray@ray-virtual-machine: /mnt/hgfs/2sharedFolder/PYNQ-master/sdbuild/scripts
ray@ray-virtual-machine: /mnt/hgfs/2sharedFolder/PYNQ-master/sdbuild/scripts$ sudo apt-get update
Get:1 http://cn.archive.ubuntu.com/ubuntu xenial InRelease [247 kB]
Get:2 http://cn.archive.ubuntu.com/ubuntu xenial-updates InRelease [106 kB]
Get:3 http://cn.archive.ubuntu.com/ubuntu xenial-backports InRelease [106 kB]
Get:4 http://security.ubuntu.com/ubuntu xenial-security InRelease [106 kB]
Get:5 http://cn.archive.ubuntu.com/ubuntu xenial/main amd64 Packages [1,201 kB]
Get:6 http://cn.archive.ubuntu.com/ubuntu xenial/main i386 Packages [1,196 kB]
Get:7 http://security.ubuntu.com/ubuntu xenial-security/main amd64 Packages [913 kB]
Get:8 http://cn.archive.ubuntu.com/ubuntu xenial/main Translation-en [568 kB]
13% [6 Packages store 0 B] [8 Translation-en 55.6 kB/568 kB 10%] [7 Packages 72 kB]
Get:9 http://cn.archive.ubuntu.com/ubuntu xenial/main amd64 DEP-11 Metadata [733 kB]
Get:10 http://cn.archive.ubuntu.com/ubuntu xenial/main 64x64 Icons [409 kB]
Get:11 http://cn.archive.ubuntu.com/ubuntu xenial/restricted amd64 Packages [8,344 kB]
Get:12 http://cn.archive.ubuntu.com/ubuntu xenial/restricted i386 Packages [8,684 kB]
Get:13 http://cn.archive.ubuntu.com/ubuntu xenial/restricted Translation-en [2,908 kB]
Get:14 http://cn.archive.ubuntu.com/ubuntu xenial/restricted amd64 DEP-11 Metadata [186 kB]
```

```
ray@ray-virtual-machine: /mnt/hgfs/2sharedFolder/PYNQ-master/sdbuild/scripts
ray@ray-virtual-machine: /mnt/hgfs/2sharedFolder/PYNQ-master/sdbuild/scripts$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages have been kept back:
  apt apt-utils dpkg libapt-pkg5.0 linux-generic-hwe-16.04
  linux-headers-generic-hwe-16.04 linux-image-generic-hwe-16.04
  python3-software-properties python3-update-manager
  software-properties-common software-properties-gtk ubuntu-adantage-tools
  update-manager update-manager-core update-notifier update-notifier-common
The following packages will be upgraded:
  accountsservice apparmor apport apport-gtk apt-transport-https aptdaemon
  aptdaemon-data base-files bind9-host ca-certificates distro-info-data
  dnsmasq-base dnutils dpkg-dev file-roller firefox firefox-locale-en
  ghostscript ghostscript-x gir1.2-packagekitglib-1.0 grub-common grub-pc
  grub-pc-bin grub2-common gstreamer1.0-plugins-good gstreamer1.0-pulseaudio
  imagemagick imagemagick-6.q16 imagemagick-common initramfs-tools
  initramfs-tools-bin initramfs-tools-core intel-microcode krb5-locales
  libaccountsservice0 libapparmor-perl libapparmor1 libapt-inst2.0
  libbind9-140 libc-bin libc-dev-bin libc6 libc6-dbg libicu-dev libcaca0
  libcurl3 libcurl3-gnutls libdns-export162 libdns162 libdpkg-perl libexif12
  libfreetype6 libglib2.0-0 libglib2.0-bin libgnutls-openssl27
```

- sudo apt-get update: Updates the **local package index** with the latest information from the repositories, ensuring that your system is aware of the latest available software versions.
- sudo apt-get upgrade: Upgrades all **installed packages** on your system to their latest versions based on the updated package list.

Install Dependencies for Petalinux

- Run **sudo add-apt-repository ppa:openjdk-r/ppa**
- Run **sudo apt-get update**

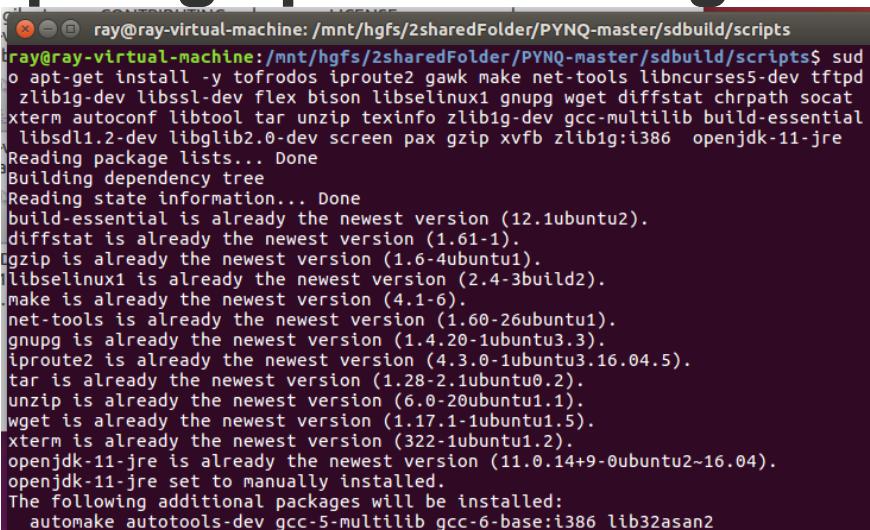


```
ray@ray-virtual-machine: /mnt/hgfs/2sharedFolder/PYNQ-master/sdbuild/scripts$ sudo apt-get update
Reading package lists... Done
ray@ray-virtual-machine: /mnt/hgfs/2sharedFolder/PYNQ-master/sdbuild/scripts$ sudo apt-get install openjdk-11-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ca-certificates-java fonts-dejavu-extra java-common libatk-wrapper-java
  libatk-wrapper-java-jni libgif7 libice-dev libpthread-stubs0-dev libsm-dev
  libx11-dev libx11-doc libxau-dev libxcb1-dev libxdmcp-dev libxt-dev
  openjdk-11-jdk-headless openjdk-11-jre openjdk-11-jre-headless
  x11proto-core-dev x11proto-input-dev x11proto-kb-dev xorg-sgml-doctools
  xtrans-dev
Suggested packages:
  default-jre libice-doc libsm-doc libxcb-doc libxt-doc openjdk-11-demo
  openjdk-11-source visualvm fonts-ipafont-gothic fonts-ipafont-mincho
  fonts-wqy-microhei | fonts-wqy-zenhei fonts-indic
The following NEW packages will be installed:
  ca-certificates-java fonts-dejavu-extra java-common libatk-wrapper-java
  libatk-wrapper-java-jni libgif7 libice-dev libpthread-stubs0-dev libsm-dev
  libx11-dev libx11-doc libxau-dev libxcb1-dev libxdmcp-dev libxt-dev
  openjdk-11-jdk openjdk-11-jdk-headless openjdk-11-jre
  openjdk-11-jre-headless x11proto-core-dev x11proto-input-dev x11proto-kb-dev
  xorg-sgml-doctools xtrans-dev
```

- Ubuntu 16.04's default repositories only include **OpenJDK 8** (the package `openjdk-8-jre`), and OpenJDK 11 was not included in those repositories. So, we need to add a repository that includes it

Install Dependencies for Petalinux

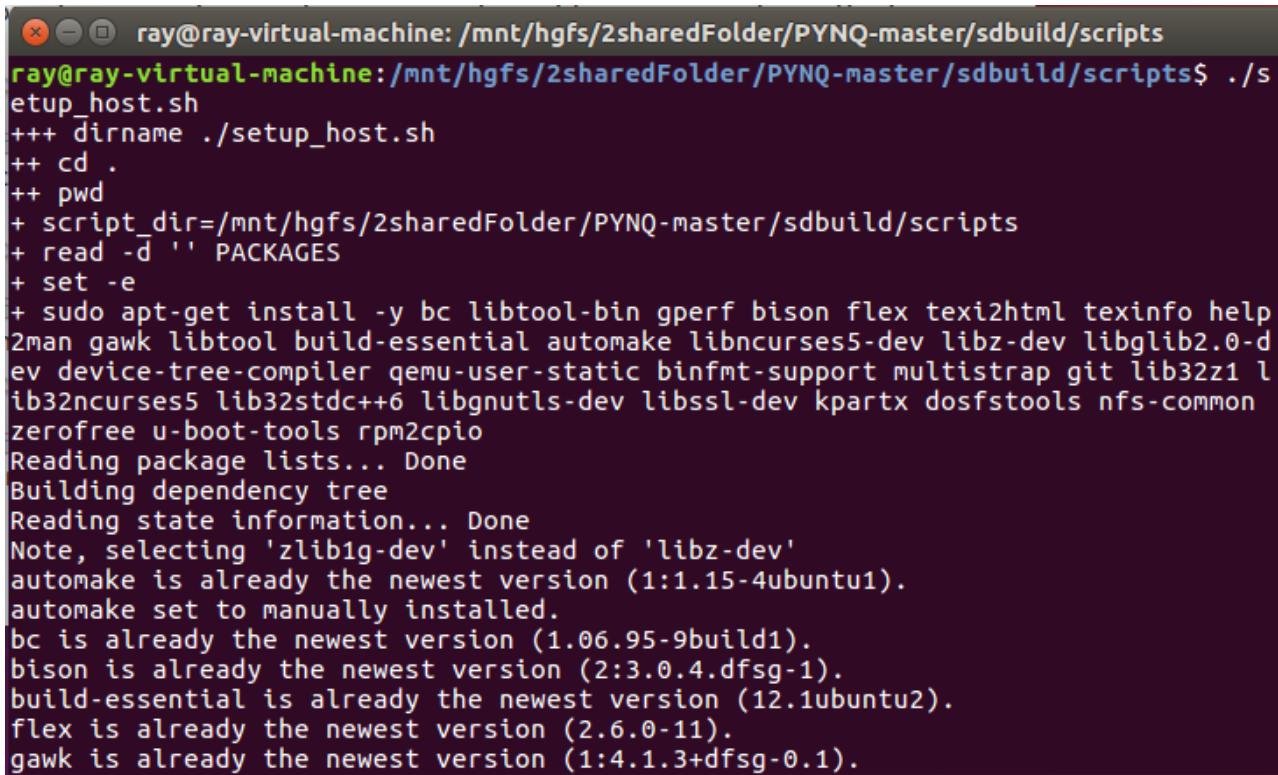
- Run **sudo apt-get install -y tofrodos iproute2 gawk make net-tools libncurses5-dev tftpd zlib1g-dev libssl-dev flex bison libselinux1 gnupg wget diffstat chrpath socat xterm autoconf libtool tar unzip texinfo zlib1g-dev gcc-multilib build-essential libsdl1.2-dev libglib2.0-dev screen pax gzip xvfb zlib1g:i386 openjdk-11-jre**



```
ray@ray-virtual-machine:/mnt/hgfs/2sharedFolder/PYNQ-master/sdbuild/scripts$ sudo apt-get install -y tofrodos iproute2 gawk make net-tools libncurses5-dev tftpd zlib1g-dev libssl-dev flex bison libselinux1 gnupg wget diffstat chrpath socat xterm autoconf libtool tar unzip texinfo zlib1g-dev gcc-multilib build-essential libsdl1.2-dev libglib2.0-dev screen pax gzip xvfb zlib1g:i386 openjdk-11-jre
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.1ubuntu2).
diffstat is already the newest version (1.61-1).
gzip is already the newest version (1.6-4ubuntu1).
libselinux1 is already the newest version (2.4-3build2).
make is already the newest version (4.1-6).
net-tools is already the newest version (1.60-26ubuntu1).
gnupg is already the newest version (1.4.20-1ubuntu3.3).
iproute2 is already the newest version (4.3.0-1ubuntu3.16.04.5).
tar is already the newest version (1.28-2.1ubuntu0.2).
unzip is already the newest version (6.0-20ubuntu1.1).
wget is already the newest version (1.17.1-1ubuntu1.5).
xterm is already the newest version (322-1ubuntu1.2).
openjdk-11-jre is already the newest version (11.0.14+9-0ubuntu2~16.04).
openjdk-11-jre set to manually installed.
The following additional packages will be installed:
  automake autotools-dev gcc-5-multilib gcc-6-base:i386 lib32asan2
```

Install Dependencies for Petalinux

- Run `sdbuild/scripts/setup_hosts.sh`



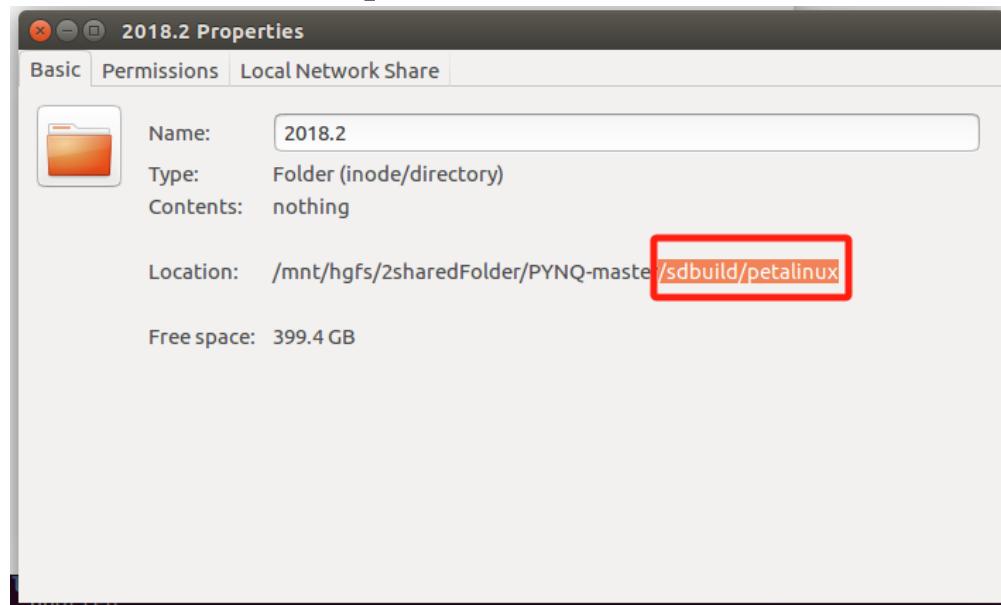
```
ray@ray-virtual-machine: /mnt/hgfs/2sharedFolder/PYNQ-master/sdbuild/scripts$ ./setup_host.sh
++ dirname ./setup_host.sh
++ cd .
++ pwd
+ script_dir=/mnt/hgfs/2sharedFolder/PYNQ-master/sdbuild/scripts
+ read -d '' PACKAGES
+ set -e
+ sudo apt-get install -y bc libtool-bin gperf bison flex texi2html texinfo help2man gawk libtool build-essential automake libncurses5-dev libbz-dev libglib2.0-dev device-tree-compiler qemu-user-static binfmt-support multistrap git lib32z1 lib32ncurses5 lib32stdc++6 libgnutls-dev libssl-dev kpartx dosfstools nfs-common zerofree u-boot-tools rpm2cpio
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'zlib1g-dev' instead of 'libbz-dev'
automake is already the newest version (1:1.15-4ubuntu1).
automake set to manually installed.
bc is already the newest version (1.06.95-9build1).
bison is already the newest version (2:3.0.4.dfsg-1).
build-essential is already the newest version (12.1ubuntu2).
flex is already the newest version (2.6.0-11).
gawk is already the newest version (1:4.1.3+dfsg-0.1).
```

- If you encounter error such as *Cannot create symlink to ‘COPYING’*, you need to create a new folder in your VM and copy **all files** in the shared folder into it.

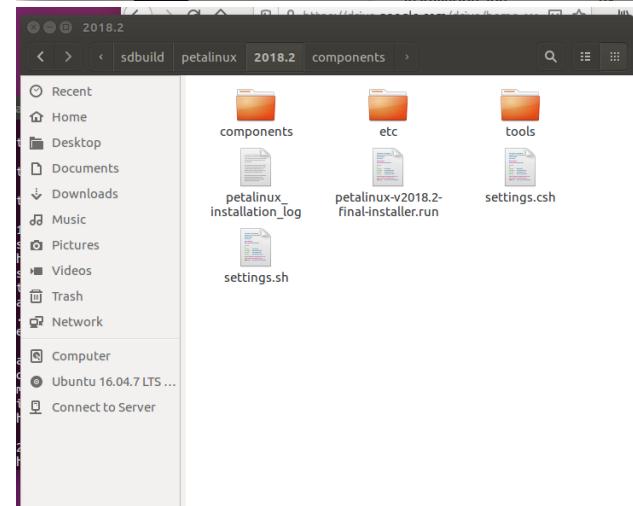


Install Petalinux (SUPER HARD)

- Initialize a new folder under sdbuild name as petalinux/2018.2
- Run **./petalinux-v2018.2-final-installer.run**
/sdbuild/petalinux/2018.2



```
ray@ray-virtual-machine: ~/Desktop/ceshi/PYNQ-master/sdbuild/petalinux/2018.2
bionic.arm.2.3.img
ray@ray-virtual-machine:~/Desktop/ceshi$ cd PYNQ-master/
ray@ray-virtual-machine:~/Desktop/ceshi/PYNQ-master$ ls
boards          pynq
CONTRIBUTING.md README.md
docs            sdbuild
LICENSE         setup.py
logo.png        THIRD_PARTY_LIC
petalinux_installation_log Xilinx_SDSoC_2018.2_0614_1954_Lin64.bin
ray@ray-virtual-machine:~/Desktop/ceshi/PYNQ-master$ cd pynq/
ray@ray-virtual-machine:~/Desktop/ceshi/PYNQ-master/pynq$ ls
gpio.py    interrupt.py  mlio.py  overlay.py  pl.py  ps.py  uio.py
__init__.py  halconfig.py  mraa.py  osd.py     pmbus.py  xlnk.py
ray@ray-virtual-machine:~/Desktop/ceshi/PYNQ-master/pynq$ cd ..
ray@ray-virtual-machine:~/Desktop/ceshi/PYNQ-master$ cd sdbuild/
ray@ray-virtual-machine:~/Desktop/ceshi/PYNQ-master/sdbuild$ ls
boot  build  ccache  Makefile  packages  petalinux  README.md  scripts  util
ray@ray-virtual-machine:~/Desktop/ceshi/PYNQ-master/sdbuild$ cd petalinux/
ray@ray-virtual-machine:~/Desktop/ceshi/PYNQ-master/sdbuild/petalinux$ cd 2018.2
ray@ray-virtual-machine:~/Desktop/ceshi/PYNQ-master/sdbuild/petalinux/2018.2$ ./petalinux-v2018.2-final-installer.run
INFO: Checking installer checksum...
```

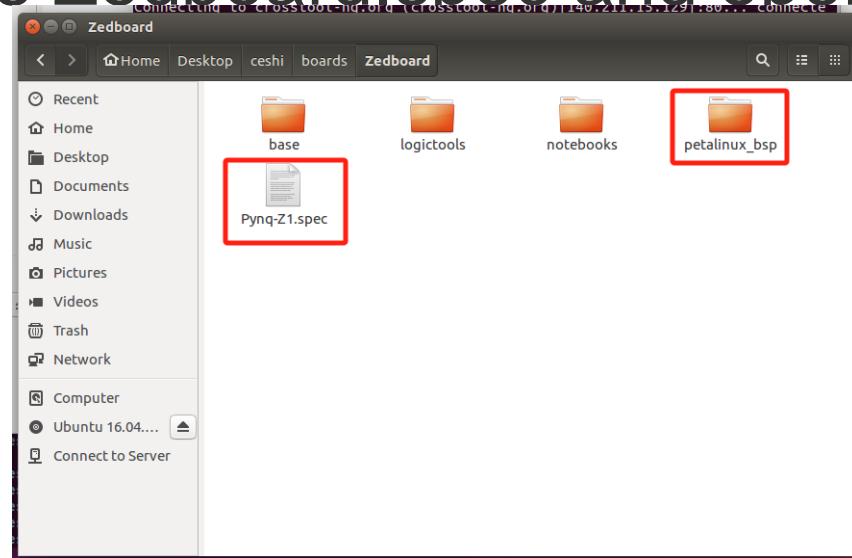
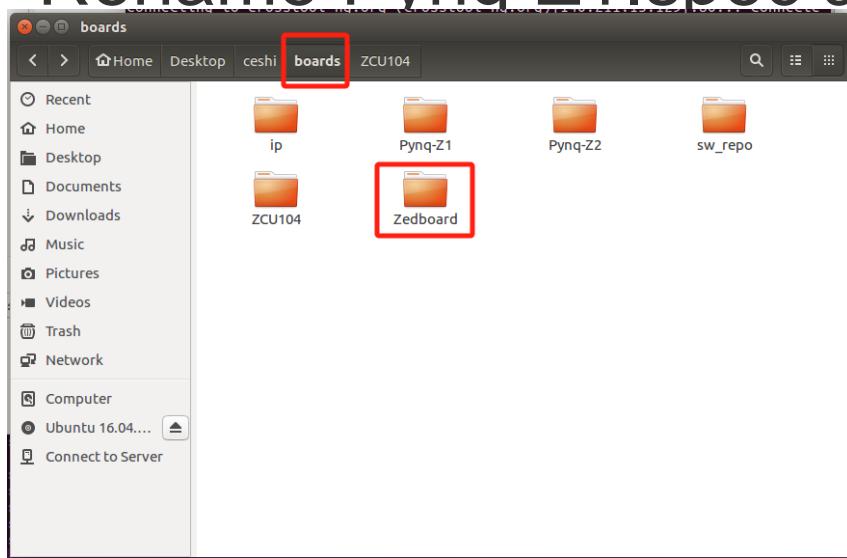


- After installation, the folder used for installation should be the same as the right picture



Add .bsp file

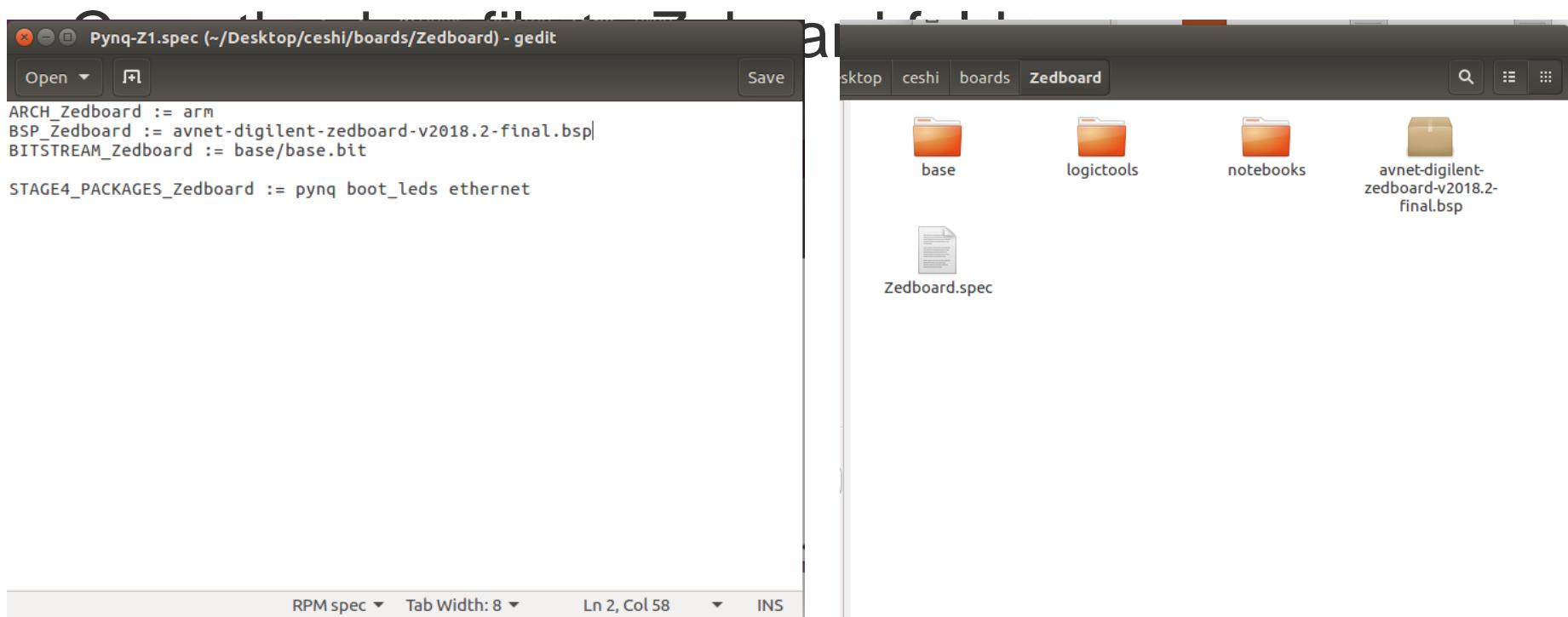
- Open boards under PYNQ-master folder
- Duplicate Pynq-Z1 and rename the folder as Zedboard
- Delete *petalinux_bsp* folder and add *avnet-digilent-zedboard-2018.2.bsp* to this folder
- Rename Pynq-Z1.spec as Zedboard.spec and open it





Add .bsp file

- Change all “PYNQ-Z1” to “zed” and fill BSP filename
- Attach the file name of .bsp file after
BSP_Zedboard :=





Prepare Environment

- Run **source sdbuild/Xilinx/Vivado/2018.2/setting64.sh**
- Run **source sdbuild/petalinux/2018.2/setting.sh**
- Run **source sdbuild/SDSoC/2018.2/setting64.sh**
 - echo \$PATH can be used to check source variables
 - The output should be like

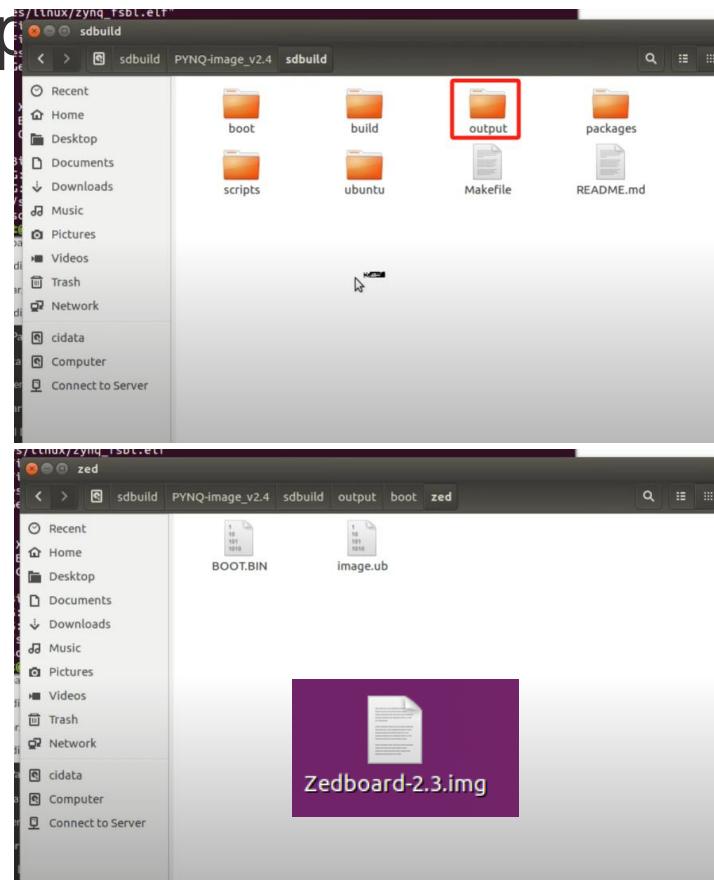
```
vagrant@ubuntu-xenial:~$ INFO: Checking network and other services
WARNING: No tftp server found - please refer to "PetaLinux SDK Installation Guide" for its impact and solution
vagrant@ubuntu-xenial:~$ echo $PATH
/sdbuild/petalinux/2018.3/tools/linux-i386/petalinux/bin:/sdbuild/petalinux/2018.3/tools/common/petalinux/bin:/sdbuild/petalinux/2018.3/tools/xscf/SDK/2018.3/bin:/sdbuild/petalinux/2018.3/tools/linux-i386/gcc-arm-none-eabi-r5/bin:/sdbuild/petalinux/2018.3/tools/linux-i386/microblaze-xilinx-elf/bin:/sdbuild/petalinux/2018.3/tools/linux-i386/microblazeel-xilinx-linux-gnu/bin:/sdbuild/petalinux/2018.3/tools/linux-i386/gcc-arm-none-eabi/bin:/sdbuild/petalinux/2018.3/tools/linux-i386/gcc-arm-linux-gnueabi/bin:/sdbuild/petalinux/2018.3/tools/linux-i386/aarch64-none-elf/bin:/sdbuild/petalinux/2018.3/tools/linux-i386/aarch64-linux-gnu/bin:/sdbuild/Xilinx/DocNav:/sdbuild/Xilinx/SDx/2018.3/bin:/sdbuild/Xilinx/SDK/2018.3/bin:/sdbuild/Xilinx/SDK/2018.3/gnu/microblaze/lin/bin:/sdbuild/Xilinx/SDK/2018.3/gnu/arm/lin/bin:/sdbuild/Xilinx/SDK/2018.3/gnu/microblaze/linux_toolchain/lin64_le/bin:/sdbuild/Xilinx/SDK/2018.3/gnu/aarch32/lin/gcc-arm-linux-gnueabi/bin:/sdbuild/Xilinx/SDK/2018.3/gnu/aarch32/lin/gcc-arm-none-eabi/bin:/sdbuild/Xilinx/SDK/2018.3/gnu/aarch64/lin/aarch64-linux/bin:/sdbuild/Xilinx/SDK/2018.3/gnu/aarch64/lin/aarch64-none/bin:/sdbuild/Xilinx/SDK/2018.3/gnu/armr5/lin/gcc-arm-none-eabi/bin:/sdbuild/Xilinx/SDK/2018.3/tps/lnx64/cmake-3.3.2/bin:/sdbuild/Xilinx/Vivado/2018.3/bin:/opt/qemu/bin:/opt/cross tool-ng/bin:/home/vagrant/bin:/home/vagrant/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
vagrant@ubuntu-xenial:~$
```



Build the Image

- Run **make boot_file BOARDS=zed** under sdbuild folder
 - It may take 4-5 hours
- The output are 1)a folder named output with image.ub inside and 2).img file at desktop

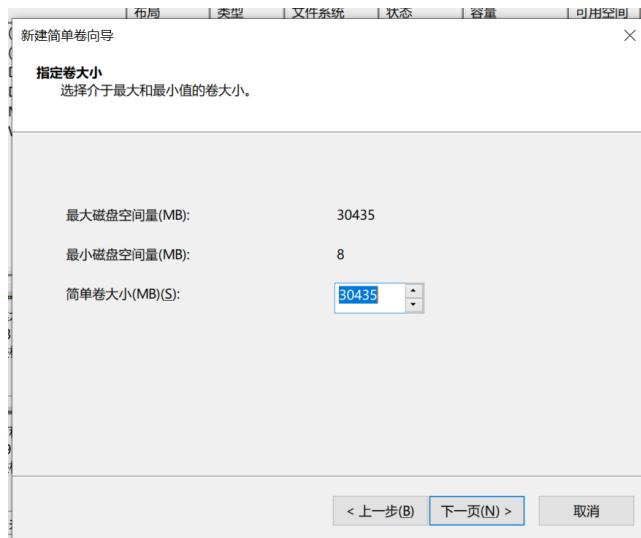
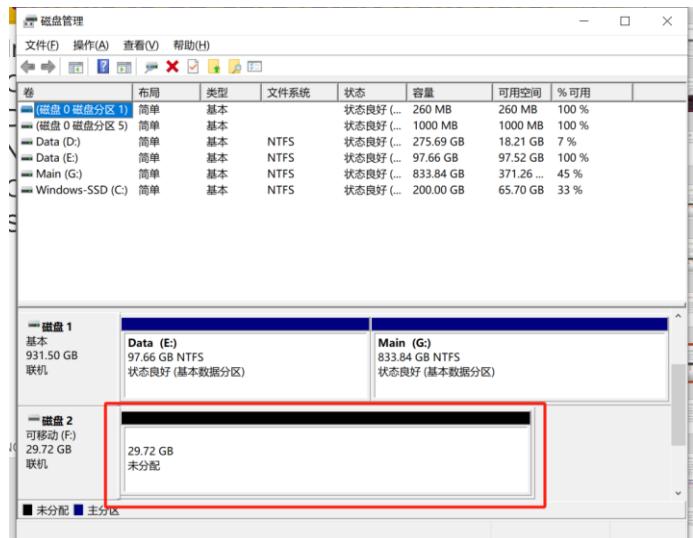
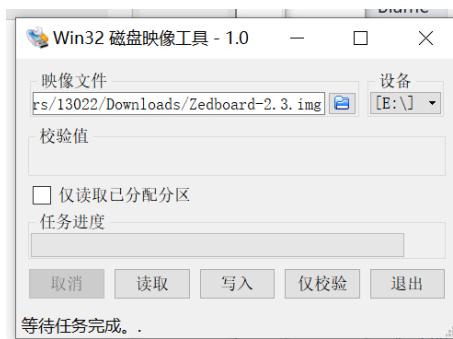
```
vagrant@ubuntu-xenial:/sdbuild/PYNQ-image_v2.4/sdbuild$ make boot_files BOARDS=zed
fatal: Not a git repository (or any parent up to mount point /sdbuild)
Stopping at filesystem boundary (GIT_DISCOVERY_ACROSS_FILESYSTEM not set).
/opt/qemu/bin/qemu-arm-static -version | fgrep 2.8.0
qemu-arm version 2.8.0
which vivado | fgrep 2018.3
/sdbuild/Xilinx/Vivado/2018.3/bin/vivado
which sdx | fgrep 2018.3
/sdbuild/Xilinx/SDX/2018.3/bin/sdx
which petalinux-config | fgrep 2018.3
/sdbuild/petalinux/2018.3/tools/common/petalinux/bin/petalinux-config
which arm-linux-gnueabihf-gcc
/sdbuild/petalinux/2018.3/tools/linux-i386/gcc-arm-linux-gnueabi/bin/arm-linux-gnueabihf-gcc
which microblaze-xilinx-elf-gcc
/sdbuild/petalinux/2018.3/tools/linux-i386/microblaze-xilinx-elf/bin/microblaze-xilinx-elf-gcc
which ct-ng
/opt/crosstool-ng/bin/ct-ng
which python | fgrep /usr/bin/python
/usr/bin/python
sudo -n mount > /dev/null
bash /sdbuild/PYNQ-image_v2.4/sdbuild/scripts/check_Env.sh
```



Flash file to SD card (Same as Before)



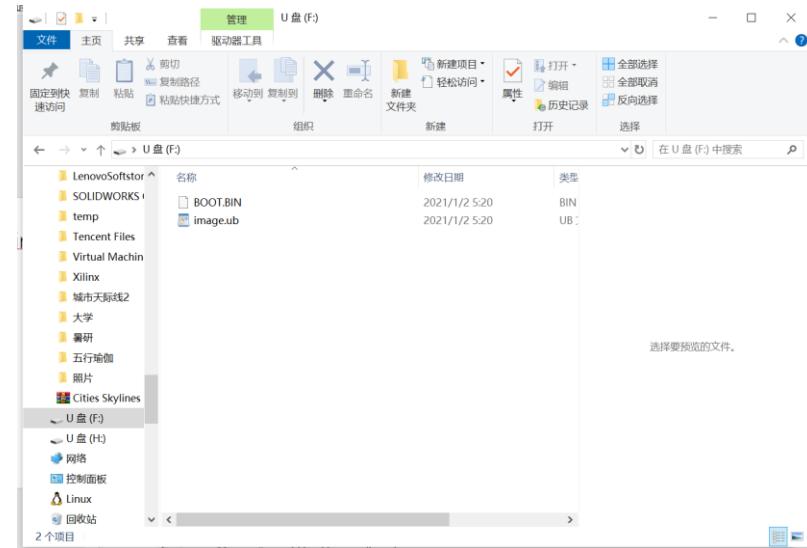
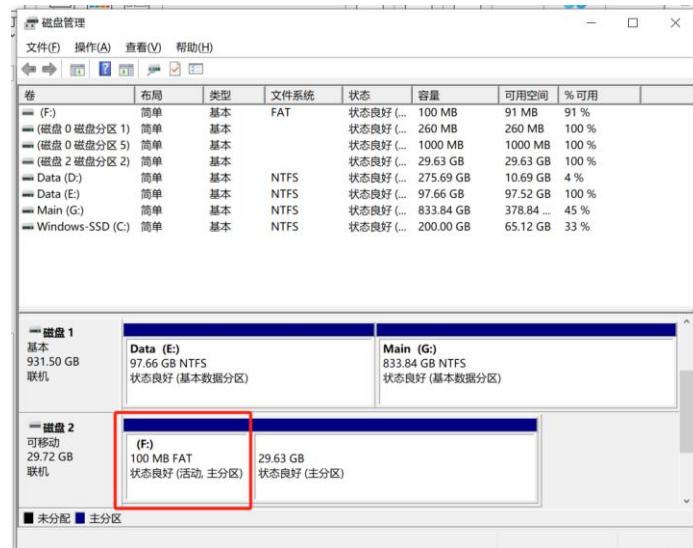
- Insert **SD card** into your computer
- Open **Disk Manager**
- Delete all **volumes** on the disk
- New a single volume with **the biggest accessible volume**
- Open **Win32DiskImager**
- Select **.img** file and flash it into SD card





Change image.ub

- After burning, 2 volumes will be created.
- F: (smaller one) is accessible with **BOOT.BIN** and **image.ub** inside.
- If you face any **unknown error**, changing them with files inside output folder is one possible solution





Create PYNQ Overlay

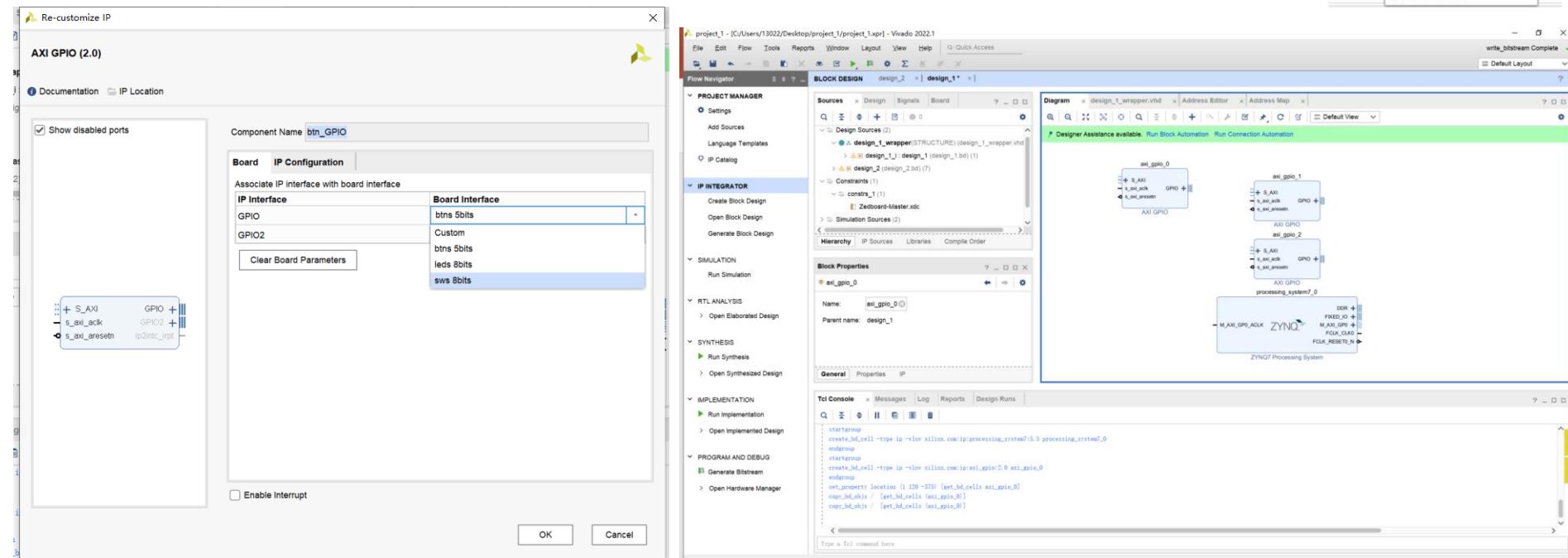
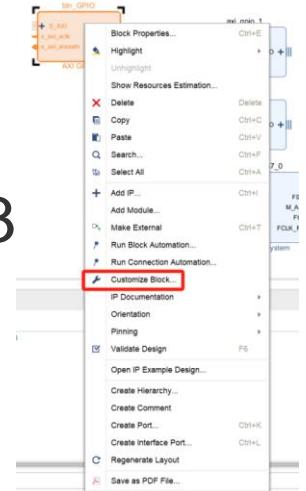
Objective: Support Bitstreams

Output: **.bit** and **.hwh** files with Your Personal PYNQ Overlay



IP Integration

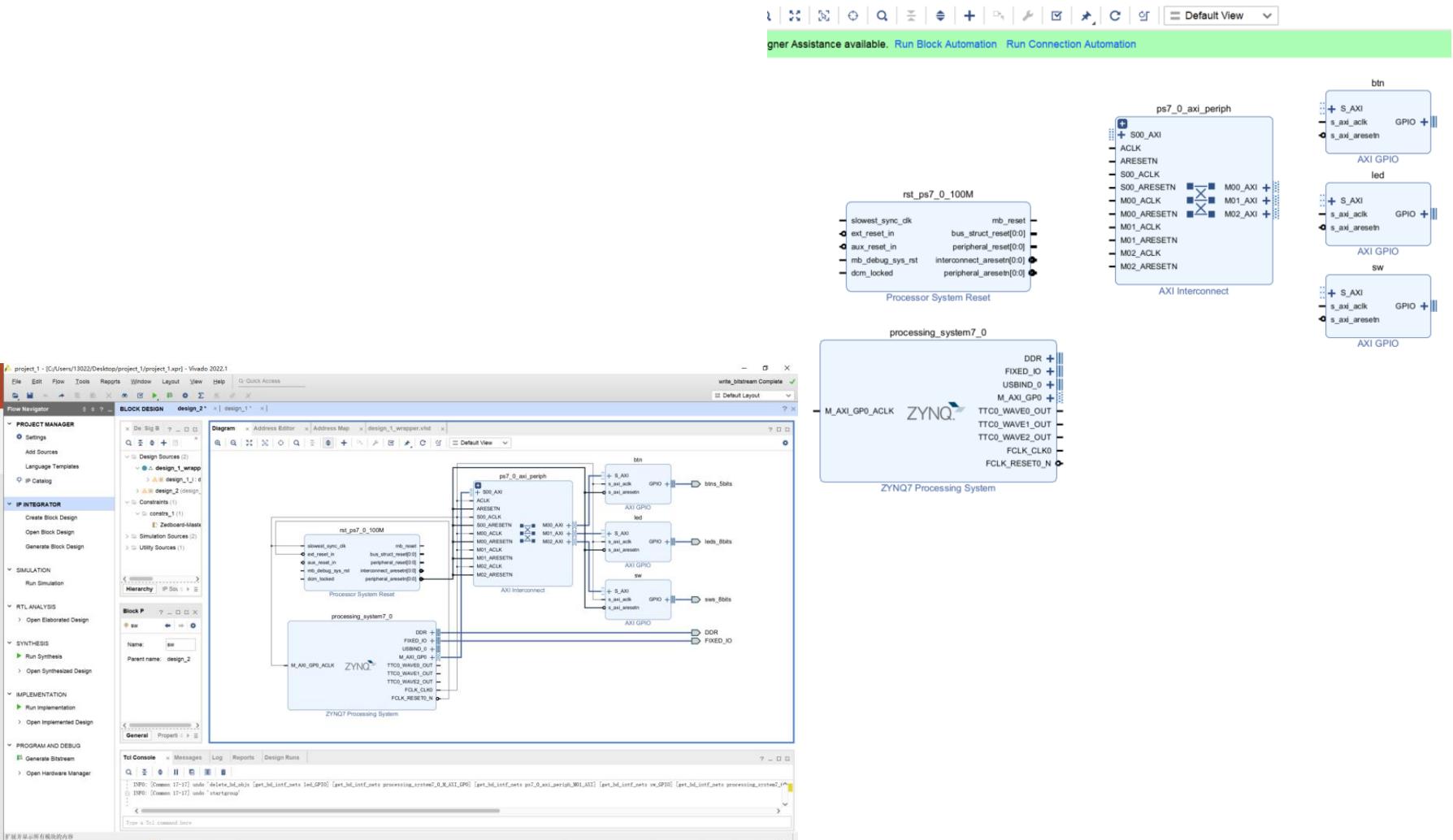
- Open Vivado and **Create a Block Design** as Lab07
 - Only in this time, no need to create new IP block
- Add **Zynq Processing System x1** and **AXI GPIO x3**
- Custom Blocks to **leds(8b)**, **sws(5b)**, and **bt�s(8b)**
- Rename the IP block as **btn**, **led**, and **sw**



IP Integration

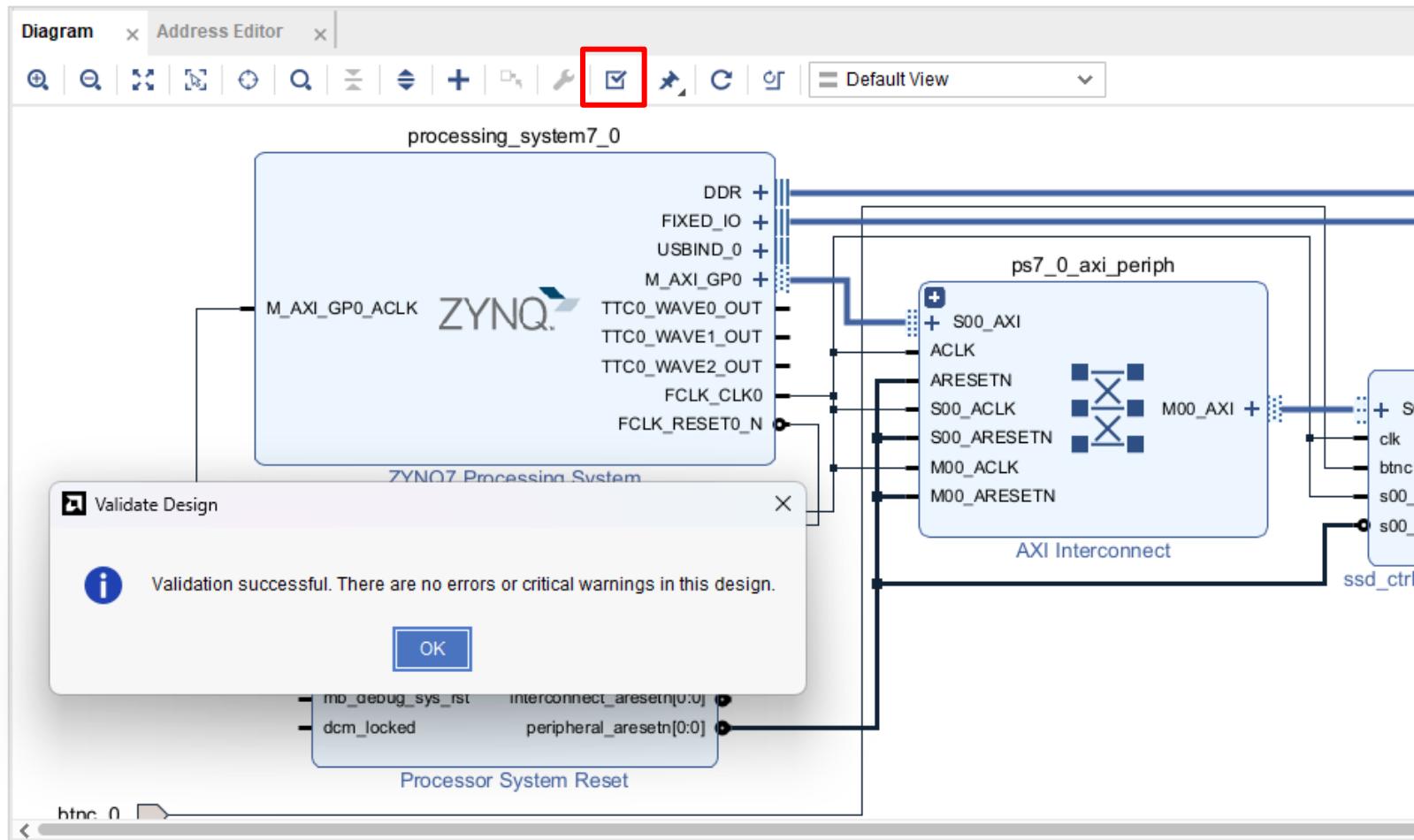


- Run Block Automation and Run Connection Automation



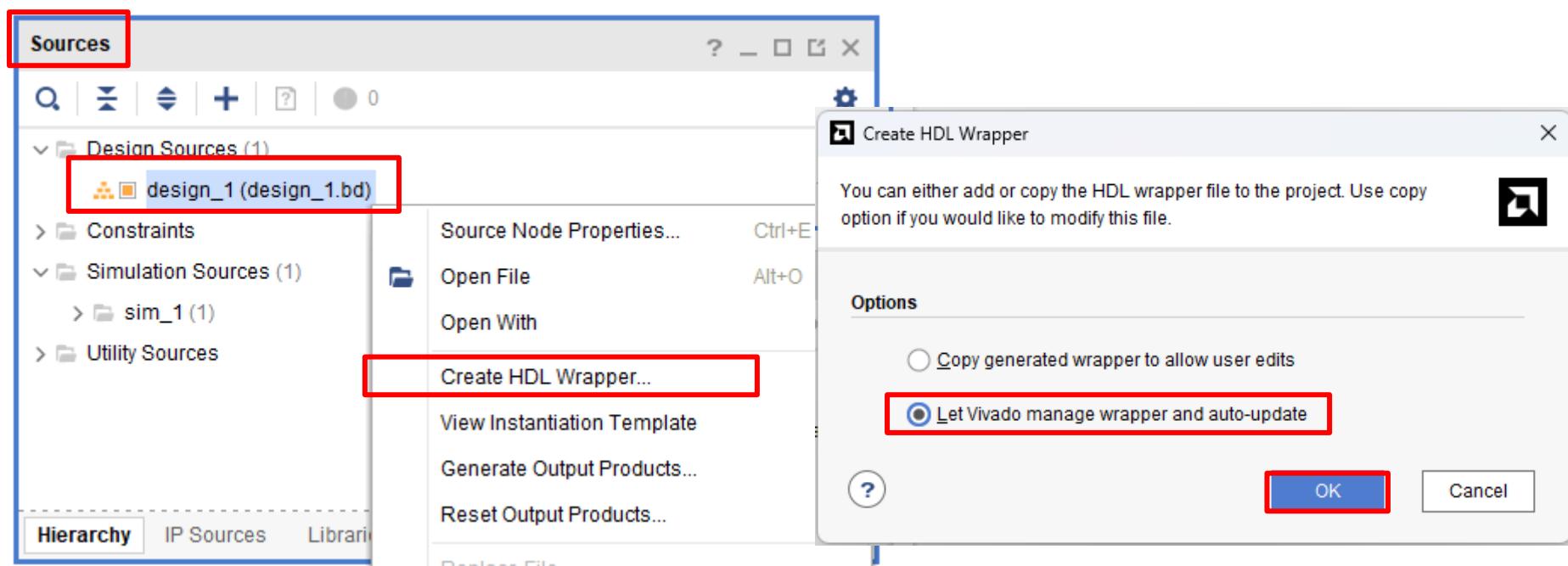
IP Integration

- Click Validate Design
- The result should have no errors



HDL Wrapper & Generate Bitstream

- Right click the **design_1** in Design Source
- Select **Create HDL Wrapper**





Export PYNQ Overlay

- **Export Hardware (.xsa with .hwh file) and Export Bitstream File (.bit file)**
- Change file name to be the same
- Upload them to **jupyter** under the same folder with .ipynb code

