

실험 3 Xilinx ZCU104 board 세팅 및 Vitis AI DPU API 이용한 MNIST 실습

작성자: 위두랑가

실험조교: 박천명, Hien

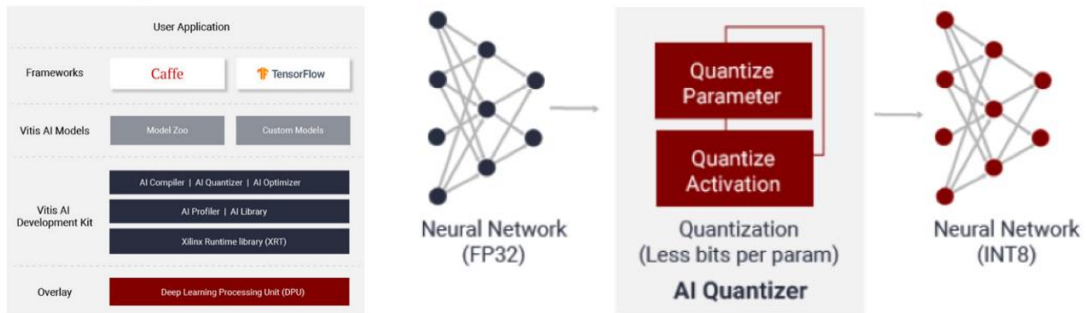
1. 실험 목적

이 실험에서는 Xilinx UltraScale+ MPSoC ZCU104 보드에서 MNIST quantization된 model를 다운받고 DPU 실행 파일을 만들어 MNIST test accuracy 및 보드에서 실행 시간을 확인해 본다.

- USB UART 통해 UltraScale+ MPSoC ZCU104 board와 PC 연결
- Xilinx zcu104 보드에 wifi dongle를 연결하여 인터넷 접속
- Vitis AI DPU API 실행하기 위한 초기화 세팅
- 실험 1에서 생성한 MNIST 모델을 업로드(xmodel)
- Xilinx zcu104 보드에서 MNIST classification test 및 evaluation 결과 확인

2. 실험 전에 준비해야할 내용

VITIS는 Xilinx UltraScale+ 및 Alveo board와 같은 Xilinx FPGA SoC (eg. Xilinx ZCU104) 플랫폼용 Vivado 및 기타 구성 요소를 사용하여 소프트웨어 및 하드웨어 개발을 위한 통합 소프트웨어 플랫폼이다. VITIS SDK의 핵심 구성 요소인 VART(VITIS AI 런타임)는 Edge 및 Cloud에서 최종 ML/AI 애플리케이션 배포를 위한 통합 인터페이스를 제공한다. Xilinx Vitis AI는 간단한 프로세스를 사용하여 Xilinx DPU(Deep Learning Processing Unit)에 딥 러닝 추론 애플리케이션을 배포하는 워크플로를 제공한다.



https://www.xilinx.com/support/documentation/sw_manuals/vitis_ai/1_0/ug1414-vitis-ai.pdf

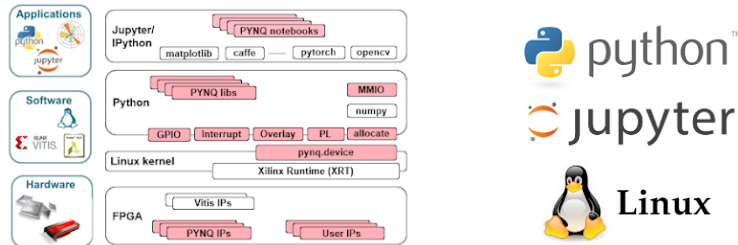
DPU(Deep Processing Unit)는 딥 러닝 추론 응용 프로그램을 위한 컨볼루션 신경망에 최적화되고 프로그래밍 가능한 논리(PL)에 배치된 구성 가능한 계산 엔진이다. DPU에는 다양한 애플리케이션의 요구 사항을 충족하도록 사용자 지정할 수 있는 효율적이고 확장 가능한 IP 코어가 포함되어 있다. DPU는 자체 명령어 세트를 정의하고 Vitis AI 컴파일러는 명령어를 생성한다.

<https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html>

Xilinx ZynqMP UltraScale+ SoC 플랫폼에서 AI 애플리케이션을 실행하는 일반적인 워크플로는 다음으로 구성된다.

- Model Quantization
- Model Compilation
- Model Optimization (Optional)
- Build DPU executable
- Build software application
- Integrate VITIS AI Unified APIs
- Compile and link the hybrid DPU application
- Deploy the hybrid DPU executable on FPGA

이 실험은 Xilinx ZCU104 보드에 PYNQ v2.6 image(OS) 파일을 올려 Xilinx Zynq 보드에서 편하게 python 및 jupyter notebook을 사용할 수 있는 플랫폼에서 진행한다.



<http://www.pynq.io/home.html>

3. 실습 실험을 진행하기 위한 환경 세팅

a) Xilinx UltraScale+ MPSoC ZCU104 board introduction

Xilinx Zynq® UltraScale+™ MPSoC ZCU104 평가 키트는 감시, ADAS(첨단 운전 보조 시스템), 머신 비전, AR(증강 현실), 드론 및 의료용 영상과 같은 임베디드 비전 애플리케이션 설계에 신속하게 착수할 수 있게 해준다. Zynq UltraScale+ MPSoC ZCU104 평가 키트는 비디오 코덱을 갖춘 Zynq UltraScale+ MPSoC 이 특징이며 임베디드 비전 용례를 위한 다양한 공통 주변 장치 및 인터페이스를 지원한다. ZCU104 평가 보드는 고속 DDR4 메모리 인터페이스, FMC 확장 포트, 멀티 기가 비트급 직렬 송수신기, 다양한 주변 장치 인터페이스, 맞춤 설계용 FPGA 패브릭을 갖춘 유연한 시제품 제작 플랫폼이다. ZCU104 reVISION 패키지는 OpenCV 라이브러리, 머신 러닝 프레임워크, 라이브 센서 지원을 갖춘 바로 활용 가능한 SDSoc™ 개발 환경 소프트웨어 플로우를 제공한다.

<https://www.xilinx.com/products/boards-and-kits/zcu104.html#overview>

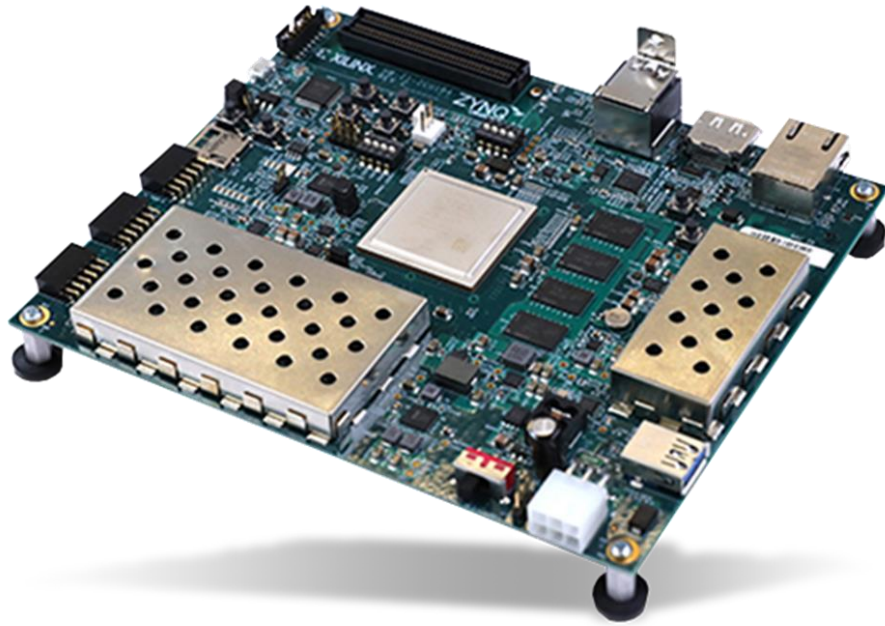


Figure 3.1. Xilinx UltraScale+ MPSoC ZCU104 Board.

b) Board Setup

- i.) Boot Dip Switches (SW6)을 다음 position으로 세팅한다: (이것은 Micro-SD 카드로부터 보드를 부팅하도록 한다.)
 - Dip switch 1 (Mode 0): On (down position in diagram)
 - Dip switch 2 (Mode 1): Off (up position in diagram)
 - Dip switch 3 (Mode 2): Off (up)
 - Dip switch 4 (Mode 3): Off (up)
- ii.) 12V 파워 케이블을 연결한다.
- iii.) Micro SD card를 삽입하고 적절한 PYNQ 이미지를 로드한다.
- iv.) USB cable을 PC나 노트북과 보드의 USB JTAG UART MicroUSB 포트에 연결한다.
- v.) 보드를 켜다. (power 스위치를 ON 위치로 민다.)

빨간 LED와 보드의 노란 LED가 추가로 켜지며 보드가 켜졌음을 보여준다. 몇 초후 빨간 LED가 노란색으로 변한다. 이것은 bitstream이 다운되었고

시스템이 부팅 중임을 보여준다.

보드가 셋업되었다면 주피터 노트북을 시작하기 위해 연결해야한다.

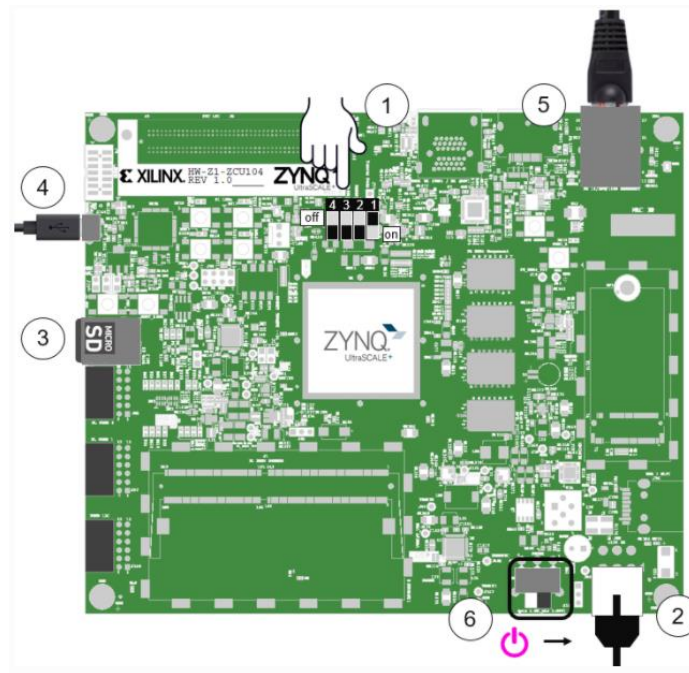


Figure 3.2. Xilinx UltraScale+ MPSoC ZCU104 Board 실행 방법.

사용할때 불이 계속 들어와있음을 확인해라.

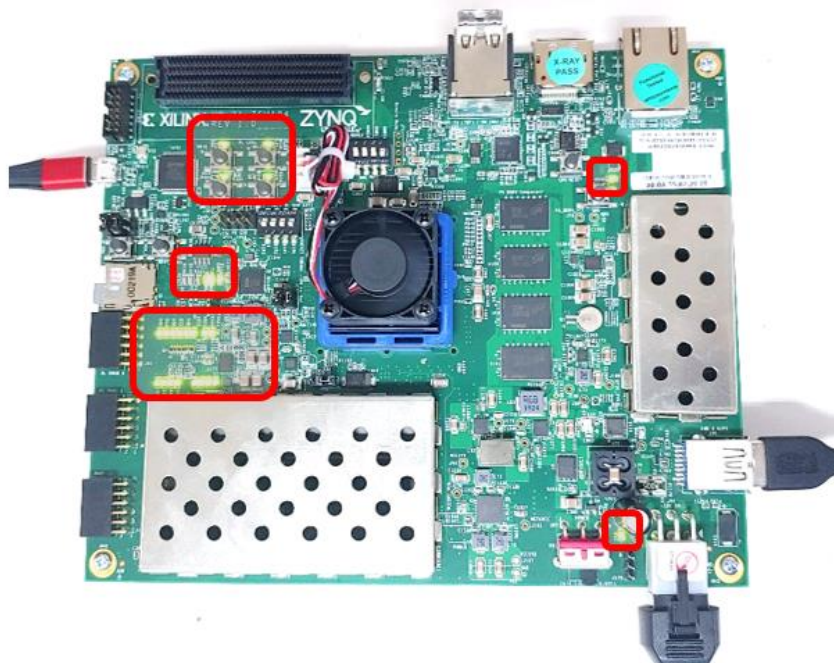
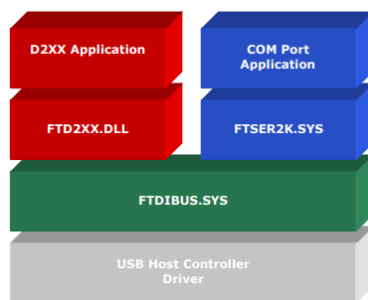


Figure 3.3. The leds of Xilinx UltraScale+ MPSoC ZCU104 board.

c) Connecting UltraScale+ MPSoC ZCU104 board with PC:

PC와 UltraScale+ MPSoC ZCU104 보드 Serial 통신을 통해 연결하기 위해 PC에 Virtual COM port (VCP) drivers를 설치해야 된다. ZCU104 보드는 FDDI chipset를 가지고 있기 때문에 Windows PC에 FT-D2XX driver 설치한다. 해당 driver가 ftdichip 홈페이지에서 다운 받을 수 있으며 아래와 같이 setup executable 파일을 다운 받아 CDM212364_Setup.exe 실행한다.

<https://ftdichip.com/drivers/vcp-drivers/>



FTDI Chip

HOME PRODUCTS APPLICATIONS DRIVERS SUPPORT ABOUT US

Drivers

Home / Drivers / VCP Drivers

Device Overview VCP Drivers D2XX Drivers D3XX Drivers

Virtual COM Port Drivers

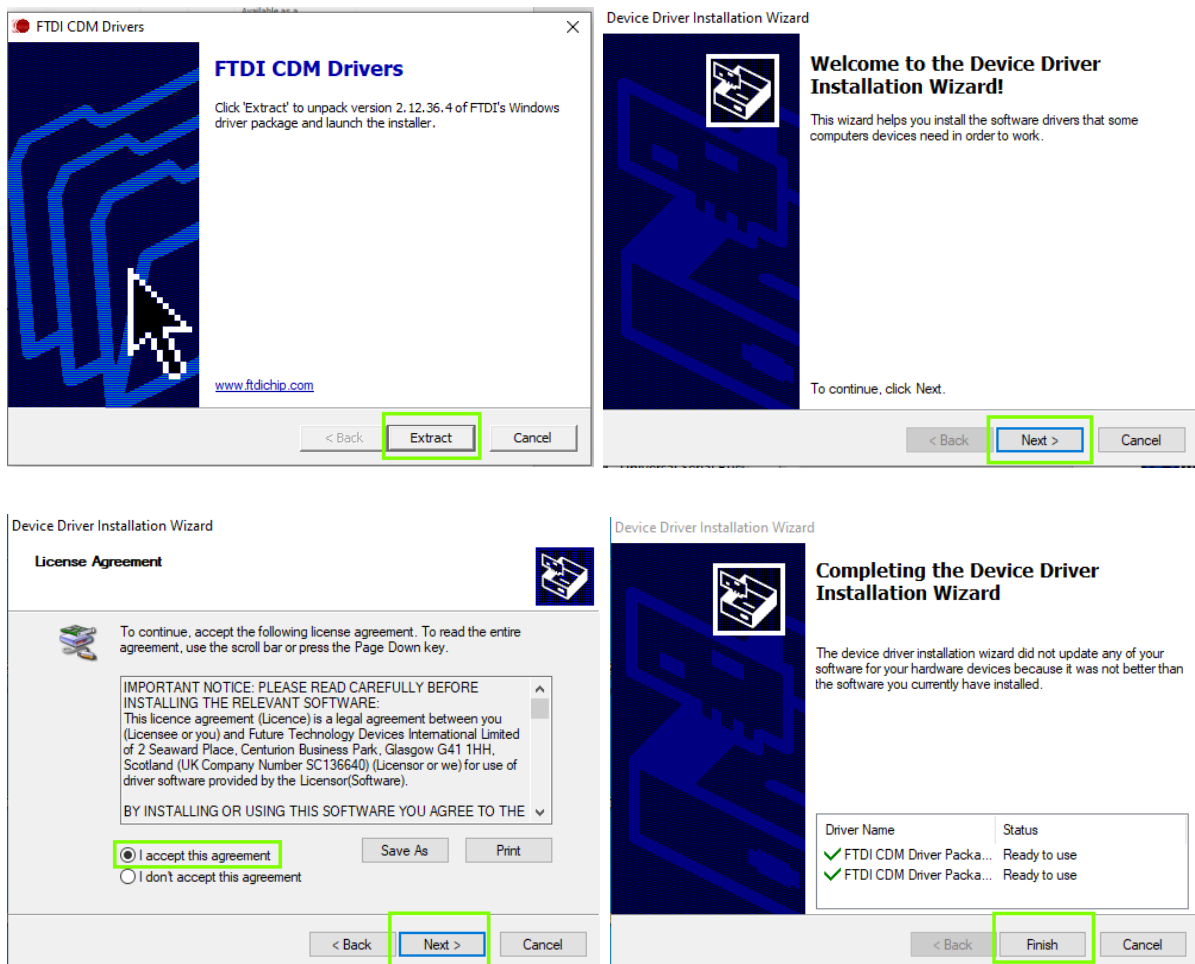
Virtual COM port (VCP) drivers cause the USB device to appear as an additional COM port available to the PC. Application software can access the USB device in the This page contains the VCP drivers currently available for FTDI devices.

For D2XX Direct drivers, please [click here](#).

Installation guides are available from the [Installation Guides page](#) of the [Documents](#) section of this site for selected operating systems.

Operating System	Release Date	Processor Architecture							Comments
		X86 (32-Bit)	X64 (64-Bit)	PPC	ARM	MIPSII	MIPSIV	SH4	
Windows*	2021-07-15	2.12.36.4	2.12.36.4	-	-	-	-	-	WHQL Certified. Includes VCP and D2XX. Available as a setup executable . Please read the Release Notes and Installation Guides .
Linux	-	-	1.5.0	-	-	-	-	-	All FTDI devices now supported in Ubuntu 11.10, kernel 3.0.0-19. Refer to TN-101 if you need a custom VCP VID/PID in Linux. VCP drivers are integrated into the kernel .
Mac OS X 10.3 to 10.8	2012-08-10	2.2.18	2.2.18	2.2.18	-	-	-	-	Refer to TN-105 if you need a custom VCP VID/PID in MAC OS
Mac OS X 10.9 to 10.14	2019-12-24	-	2.4.4	-	-	-	-	-	This driver is signed by Apple

Combined Driver Model (CDM) package 설치 (CDM212364_Setup.exe 파일 실행)



먼저 장치관리자를 통해 COM 포트 넘버를 확인해야한다. PC에 따라 다를 수 있다.

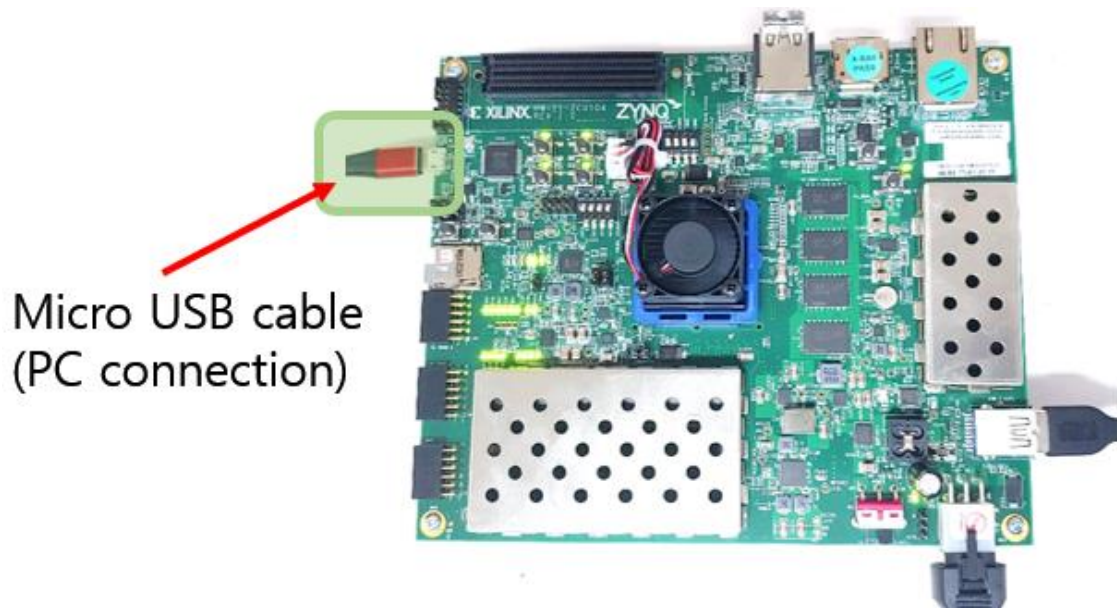


Figure 3.4. Xilinx UltraScale+ MPSoC ZCU104 Board와 PC 연결.

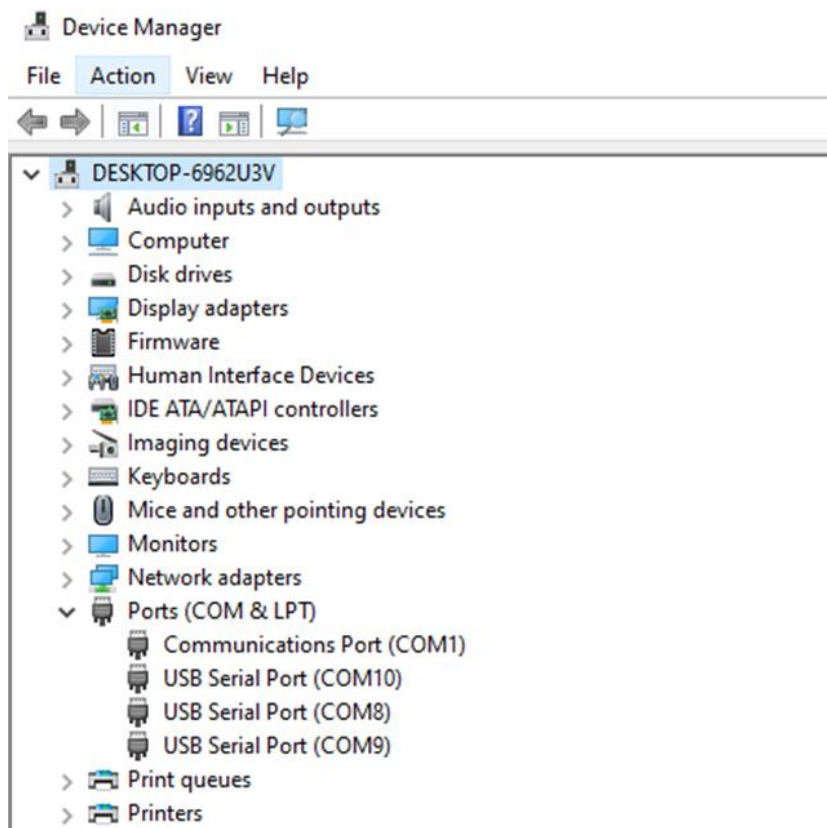
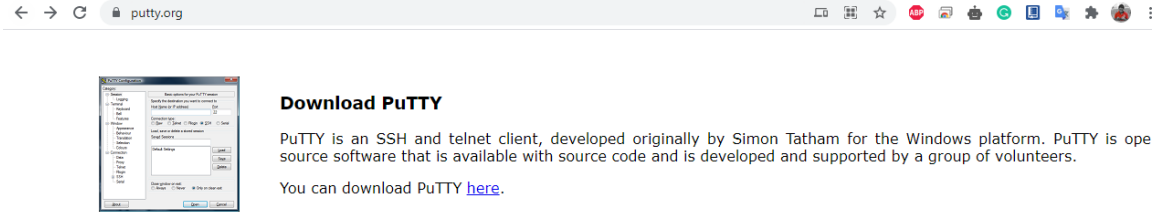


Figure 3. 5. For example, the COM port number is COM8.

PuTTY program을 설치하여 Windows PC와 Xilinx ZCU104 연결한다.

<https://www.putty.org/>

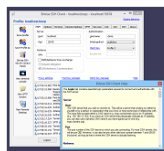


Download PuTTY

PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform. PuTTY is open source software that is available with source code and is developed and supported by a group of volunteers.

You can download PuTTY [here](#).

Below suggestions are independent of the authors of PuTTY. They are *not* to be seen as endorsements by the PuTTY project.

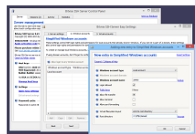


Bitvise SSH Client

Bitvise SSH Client is an SSH and SFTP client for Windows. It is developed and supported professionally by Bitvise. The SSH Client is robust, easy to install, easy to use, and supports all features supported by PuTTY, as well as the following:

- graphical SFTP file transfer;
- single-click Remote Desktop tunneling;
- auto-reconnecting capability;
- dynamic port forwarding through an integrated proxy;
- an FTP-to-SFTP protocol bridge.

Bitvise SSH Client is **free to use**. You can [download it here](#).



Bitvise SSH Server

Bitvise SSH Server is an SSH, SFTP and SCP server for Windows. It is robust, easy to install, easy to use, and works well with a variety of SSH clients, including Bitvise SSH Client, OpenSSH, and PuTTY. The SSH Server is developed and supported professionally by Bitvise.

You can [download Bitvise SSH Server here](#).

Package files

You probably want one of these. They include versions of all the PuTTY utilities.

(Not sure whether you want the 32-bit or the 64-bit version? Read the [FAQ entry](#).)

MSI ('Windows Installer')

64-bit x86: [putty-64bit-0.76-installer.msi](#) ([or by FTP](#)) ([signature](#))

64-bit ARM: [putty-arm64-0.76-installer.msi](#) ([or by FTP](#)) ([signature](#))

32-bit x86: [putty-0.76-installer.msi](#) ([or by FTP](#)) ([signature](#))

Unix source archive

.tar.gz: [putty-0.76.tar.gz](#) ([or by FTP](#)) ([signature](#))

- Serial 선택하고 확인해라
- COM port number 입력해라
- Speed: 115200 input
- Open button 을 눌러라

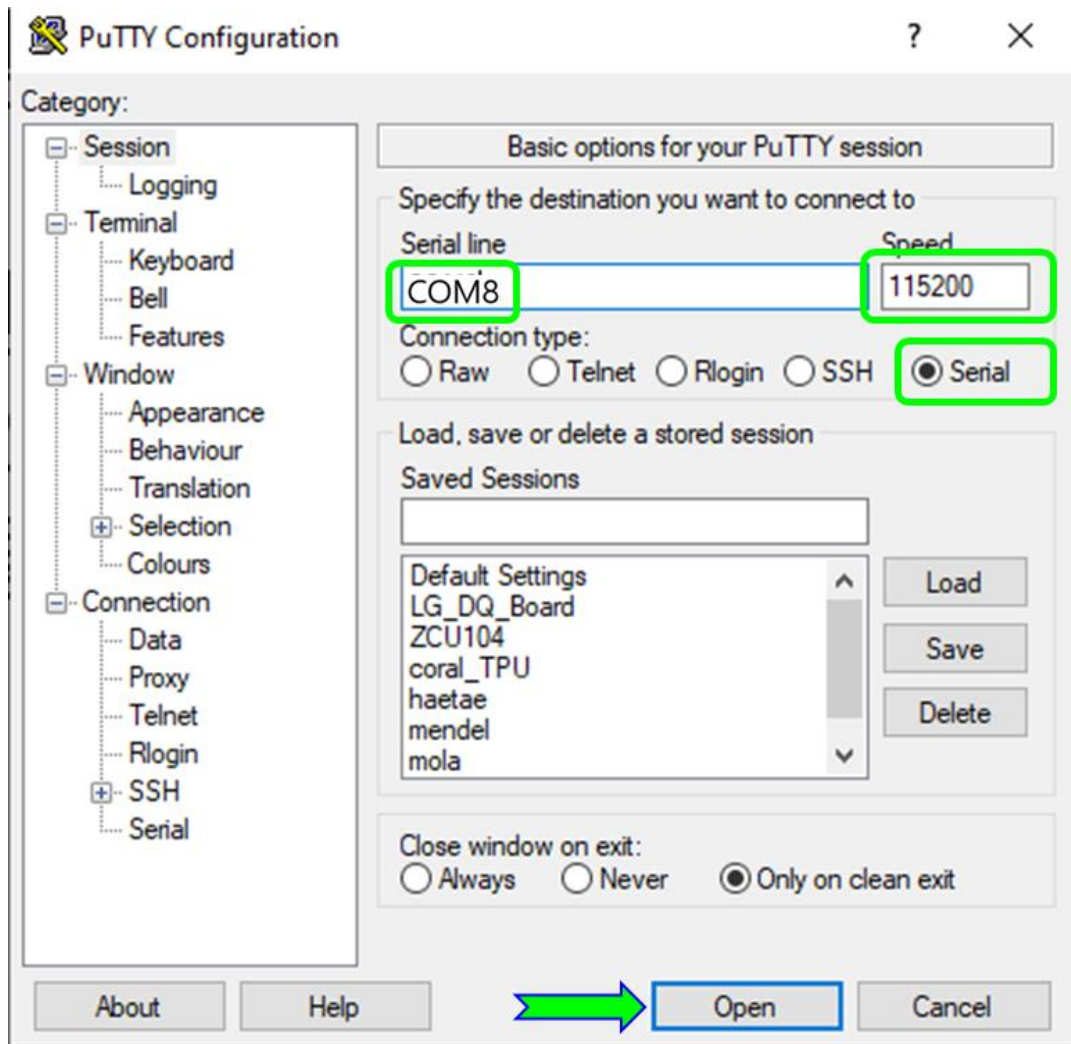


Figure 3.6. Putty Program 실행

그리고 로그인하면 선택할 수 있다:

- 자동 로그인
- 직접 입력

username : xilinx

password : xilinx

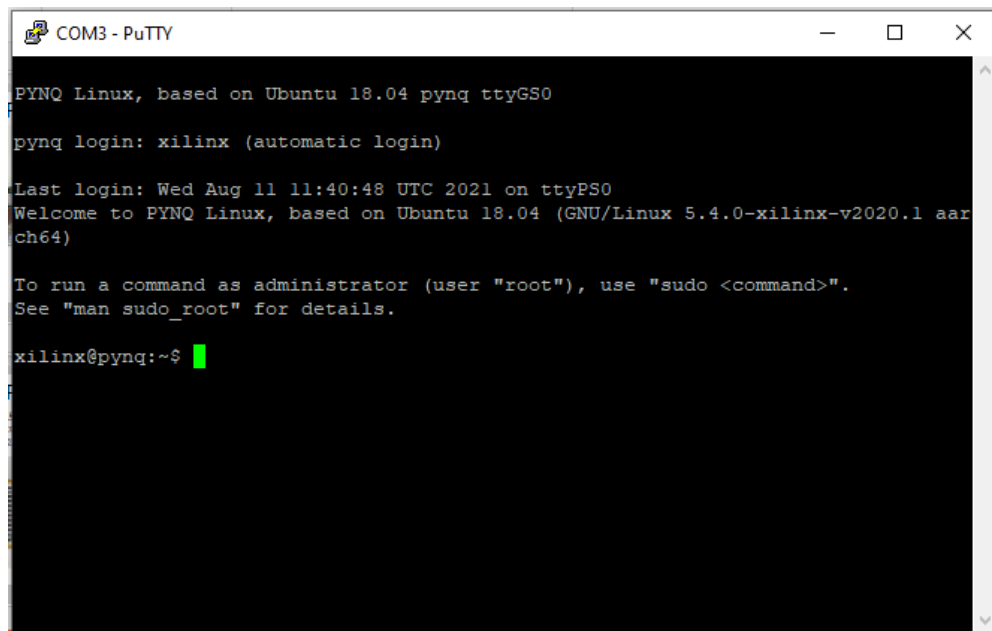


Figure 3.7. Putty Terminal

d) Xilinx zcu104 보드에 wifi dongle를 연결하여 인터넷 접속

보드와 PC를 같은 네트워크에 연결해야 한다. 그러나 zcu104 보드가 자동으로 ip 주소를 얻게된다.

- 보드의 usb 포트에 wifi dongle를 연결하여 라우터/스위치에 연결
- 라우터/스위치의 이더넷 또는 WiFi에 컴퓨터를 연결
- PC에서 <http://<보드 IP 주소>>로 이동

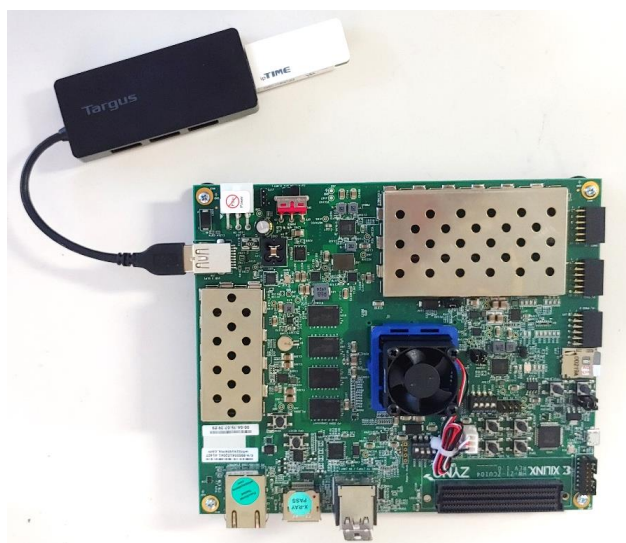
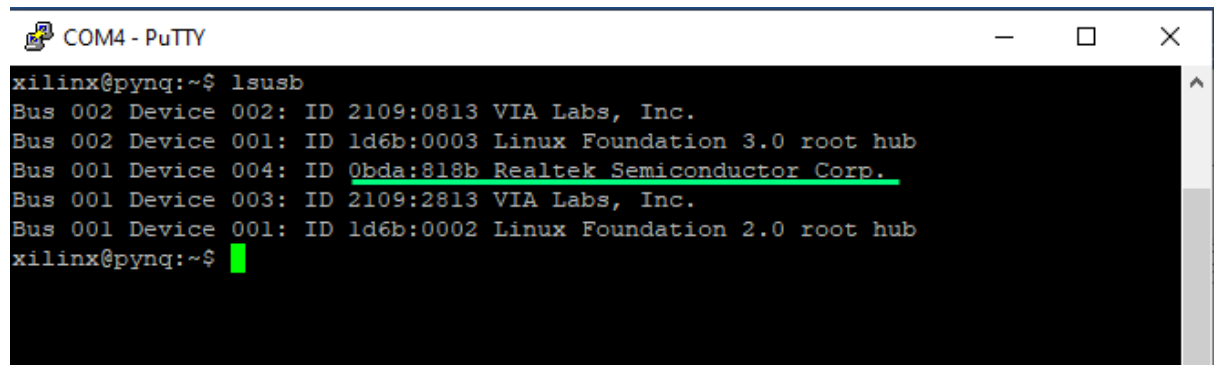


Figure 3.7. 보드의 usb 포트에 wifi dongle를 연결

lsusb 명령어 통해 wifi dongle를 제대로 연결되었는지 확인할 수 있다.

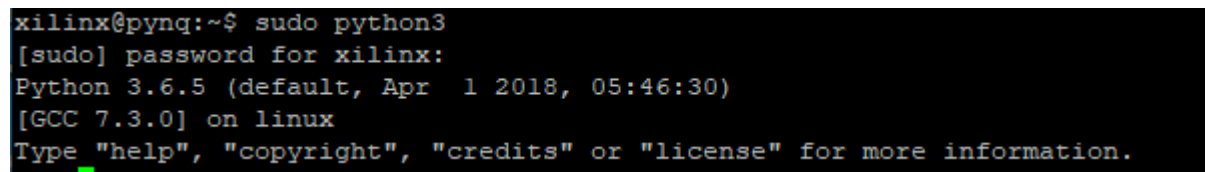
```
$ lsusb
```



```
COM4 - PuTTY
xilinx@pynq:~$ lsusb
Bus 002 Device 002: ID 2109:0813 VIA Labs, Inc.
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 004: ID 0bda:818b Realtek Semiconductor Corp.
Bus 001 Device 003: ID 2109:2813 VIA Labs, Inc.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
xilinx@pynq:~$
```

아래와 같이 sudo 권한으로 python3 실행하여 와이파이 연결한다

```
$ sudo python3
```

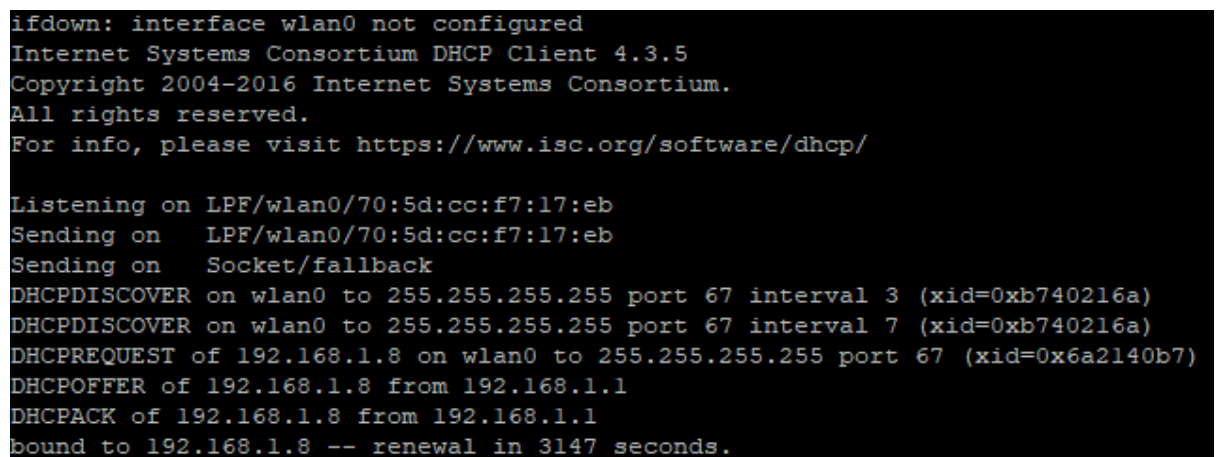


```
xilinx@pynq:~$ sudo python3
[sudo] password for xilinx:
Python 3.6.5 (default, Apr 1 2018, 05:46:30)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> from pynq.lib import Wifi
```

```
>>> port = Wifi()
```

```
>>> port.connect('wifi_name', 'wifi_password')
```



```
ifdown: interface wlan0 not configured
Internet Systems Consortium DHCP Client 4.3.5
Copyright 2004-2016 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/wlan0/70:5d:cc:f7:17:eb
Sending on   LPF/wlan0/70:5d:cc:f7:17:eb
Sending on   Socket/fallback
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 3 (xid=0xb740216a)
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 7 (xid=0xb740216a)
DHCPREQUEST of 192.168.1.8 on wlan0 to 255.255.255.255 port 67 (xid=0x6a2140b7)
DHCPOFFER of 192.168.1.8 from 192.168.1.1
DHCPACK of 192.168.1.8 from 192.168.1.1
bound to 192.168.1.8 -- renewal in 3147 seconds.
```

```
>>> exit()
```

제대로 인터넷 연결되었는지 확인하는 방법 :

```
$ sudo ping -I wlan0 www.google.com -c 10
```

```
xilinx@pynq:~$ sudo ping -I wlan0 www.google.com -c 10
[sudo] password for xilinx:
PING www.google.com (142.250.204.68) from 192.168.0.86 wlan0: 56(84) bytes of data.
64 bytes from hkg07s39-in-f4.1e100.net (142.250.204.68): icmp_seq=1 ttl=55 time=46.1 ms
64 bytes from hkg07s39-in-f4.1e100.net (142.250.204.68): icmp_seq=2 ttl=55 time=44.3 ms
64 bytes from hkg07s39-in-f4.1e100.net (142.250.204.68): icmp_seq=3 ttl=55 time=39.2 ms
64 bytes from hkg07s39-in-f4.1e100.net (142.250.204.68): icmp_seq=4 ttl=55 time=
```

e) Vitis AI DPU API 실행하기 위한 초기화 세팅.

초기화 세팅 단계가 시간이 오래걸릴 수 있기 때문에 담당 조교가 미리 세팅해 놓은 상태에서 실습을 진행한다.

For creating PYNQ DPU overlay on UltraScale+ ZCU104 Board (Setting up assistant):

```
> git clone --recursive --shallow-submodules https://github.com/Xilinx/DPU-PYNQ.git
```

```
> cd DPU-PYNQ/upgrade
```

```
> sudo make
```

```
> sudo pip3 install pynq-dpu
```

```
> cd $PYNQ_JUPYTER_NOTEBOOKS
```

```
> sudo pynq get-notebooks pynq-dpu -p .
```

(40-50분 정도 걸린다. 이 부분은 미리 세팅하였으니 진행하지 말 것)

```
> pip install --upgrade pynq-dpu
```


f) Running Jupyter notebook on UltraScale+ ZCU104 Board

> ifconfig

And we get this:

```
xilinx@pynq:~$ ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 166 bytes 18228 (18.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 166 bytes 18228 (18.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

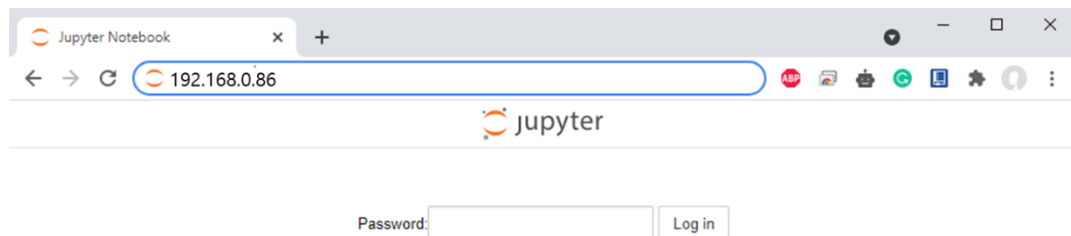
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.86 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::725d:ccff:fe7:17eb prefixlen 64 scopeid 0x20<link>
    ether 70:5d:cc:f7:17:eb txqueuelen 1000 (Ethernet)
    RX packets 299 bytes 208872 (208.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 355 bytes 39463 (39.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Ip_address

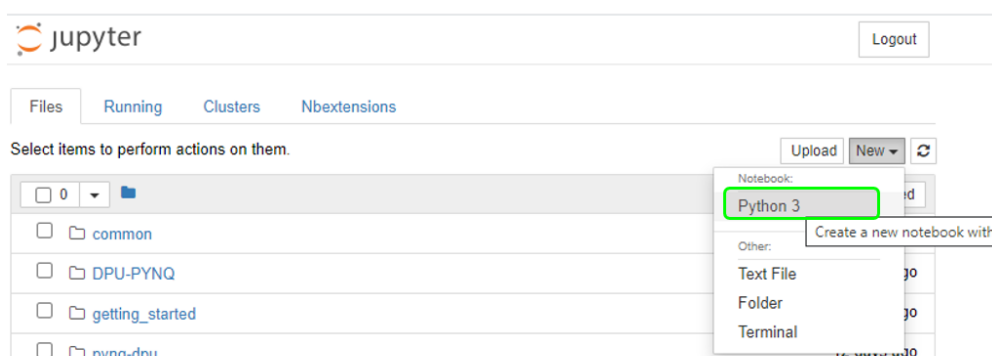
ZCU104 Board와 PC는 동일한 네트워크에 연결되어 있어야 한다.

인터넷 브라우저를 열고 보드 IP(192.168.0.86)를 연결한다.

비밀번호는 xilinx이다.



주피터 노트북에 새로운 python3 파일을 생성해라.

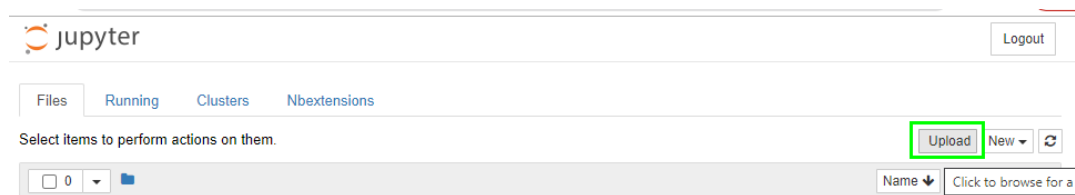


4. Jupyter Notebook을 통해 실습 실험

실험 1에서 만든 MNIST quantized model를 다운 받아 Xilinx UltraScale+ ZCU104 board에 올려서 model inference accuracy 및 실행 시간을 계산한다.

a) Quantized MNIST model upload

UltraScale+ ZCU104 board를 위한 MNIST 모델을 업로드해야한다. PC로 다운로드한 xmodel을 업로드할 수 있다.



For preparing the overlay(FPGA configuration) 및 model upload:

```
from pynq_dpu import DpuOverlay

overlay = DpuOverlay("dpu.bit")

overlay.load_model("customcnn.xmodel")
```

b) Library 추가

```
from time import time
import numpy as np
import mnist
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
from six.moves import urllib

opener = urllib.request.build_opener()
opener.addheaders = [('User-agent', 'Mozilla/5.0')]
urllib.request.install_opener(opener)
```

c) MNIST testset download 및 normalization

```

raw_data = mnist.test_images()
normalized_data = np.asarray(raw_data/255, dtype=np.float32)
test_data = np.expand_dims(normalized_data, axis=3)
test_label = mnist.test_labels()

print("Total number of test images: {}".format(test_data.shape[0]))
print(" Dimension of each picture: {}x{}".format(test_data.shape[1],
                                                    test_data.shape[2]))

```

```

Total number of test images: 10000
Dimension of each picture: 28x28

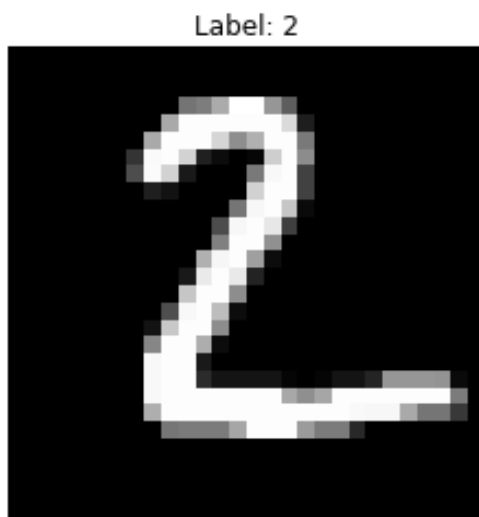
```

d) Data Visualization

```

plt.imshow(test_data[1,:,:,0], 'gray')
plt.title('Label: {}'.format(test_label[1]))
plt.axis('off')
plt.show()

```



e) Vitis DPU API

```

dpu = overlay.runner

inputTensors = dpu.get_input_tensors()
outputTensors = dpu.get_output_tensors()

shapeIn = tuple(inputTensors[0].dims)
shapeOut = tuple(outputTensors[0].dims)
outputSize = int(outputTensors[0].get_data_size() / shapeIn[0])

softmax = np.empty(outputSize)

```

```
output_data = [np.empty(shapeOut, dtype=np.float32, order="C")]
input_data = [np.empty(shapeIn, dtype=np.float32, order="C")]
image = input_data[0]
```

```
def calculate_softmax(data):
    result = np.exp(data)
    return result
```

f) Execute MNIST classification test on board:

```
num_pics = 10
fig, ax = plt.subplots(1, num_pics, figsize=(12,12))
plt.tight_layout()
for i in range(num_pics):
    image[0,...] = test_data[i]
    job_id = dpu.execute_async(input_data, output_data)
    dpu.wait(job_id)
    temp = [j.reshape(1, outputSize) for j in output_data]
    softmax = calculate_softmax(temp[0][0])
    prediction = softmax.argmax()

    ax[i].set_title('Pred: {}'.format(prediction))
    ax[i].axis('off')
    ax[i].imshow(test_data[i,:,:,:], 'gray')
```



g) 이제 evaluation results를 얻었다:

```
# We can also evaluate on the entire test dataset.

total = test_data.shape[0]
predictions = np.empty_like(test_label)
print("Classifying {} digit pictures ...".format(total))

start = time()
for i in range(total):
    image[0,...] = test_data[i]
    job_id = dpu.execute_async(input_data, output_data)
    dpu.wait(job_id)
    temp = [j.reshape(1, outputSize) for j in output_data]
    softmax = calculate_softmax(temp[0][0])
    predictions[i] = softmax.argmax()

stop = time()
correct = np.sum(predictions==test_label)
execution_time = stop-start
print("Overall accuracy: {}".format(correct/total))
print(" Execution time: {:.4f}s".format(execution_time))
print(" Throughput: {:.4f}FPS".format(total/execution_time))
```

```

Classifying 10000 digit pictures ...
Overall accuracy: 0.9855
Execution time: 5.4886s
Throughput: 1821.9470FPS

```

h) 끝나면 메모리를 지우고 clean up할 수 있다:

5. Clean up

We will need to remove references to `vart.Runner` and let Python garbage-collect the unused graph objects. This will make sure we can run other notebooks without any issue.

```

del overlay
del dpu

```

5. 실험 후 보고서에 포함될 내용

- i. Xilinx ZCU104 보드에서 Quantization을 된 CNN model의 Testset의 대한 Accuracy, Execution time와 Throughput 결과를 작성하시오
- ii. 실험 1에서 학습한 Fully Connected model를 Quantization 가정을 진행하여 생성된 FC xmodel의 Testset의 대한 Execution time 결과와 CNN xmodel의 Testset의 대한 Execution time 결과를 비교하시오
- iii. Xilinx ZCU104 보드에서 Quantization을 된 model를 10,000 Test Image를 classify시 걸리는 시간과 서버에서 Quantization을 된 model를 동작시 걸리는 시간을 비교하시오

Quantization Model	서버에서 classify시 걸리는 시간	ZCU104 보드에서 classify시 걸리는 시간
CNN		
FC		