Generative AI and AIoT (GenAIoT) Coding Skills Education

Lab Session 4.6

Building Own PYNQ-DPU and Model for FPGA Inference

Overview:

For many devices such as Ultra96v2, PYNQ-DPU github only provides older DPU and notebooks compatible with older Vitis-AI (v.2.5 and below). However, new DPUs and models have since been released. To use the latest compatible DPU, you need to rebuild the PYNQ-DPU with the latest DPU. In this lab, students will re-build the PYNQ-DPU for vitis-ai 3.5 (the latest) following the Vitis flow DPU development method. Secondly, we also quantize and compile models (Yolo3 new versions) for inference on the Ultra96v2 board.
Since the DPU building takes quite long, this manual only elaborates the method but the actual files were already generated for the students before the lab. Hence students do not need to repeat this method due to the limited time factor. Similarly, the quantized model (Yolov3_3.5) file has been provided, although we also show how to quantize the model. The model cpu quantization took over 2 hours on our computer. Hence, students should focus on the compilation of the model as well as the deployment on the FPGA. We will use the Yolov3_3.5 model provided in the vitis-ai model zoo.

Reference: https://xilinx.github.io/Vitis-AI/3.5/html/docs/workflow.html
https://github.com/Xilinx/DPU-PYNQ/blob/design_contest_3.5/README.md

Pre-Requisite:

1. It is assumed that PYNQ-3.0.1 has been installed on the SD card and is running successfully (Check previous lab).
2. Ubuntu 20.04 (Use VirtualBox to install Ubuntu 20.04)

Required Hardware:

1. Avnet Ultra96-v2 FPGA board and accessories

Required Software:

1. Windows/Linux/MAC Host PC
2. AMD PYNQ Framework
3. Vivado/Vitis 2023.1

4. Vitis AI Model Zoo
5. A browser (Mozilla/Chrome etc)

Part 1: Re-building the DPU Hardware Design (DPU Overlay)

To use the latest models such as Yolov7, Yolov8, you need to use Vitis-AI 3.5. However, for MPSoC boards such as Ultra96, the Vitis-AI 3.0 DPU also works with Vitis-AI 3.5 models. In addition, Vitis-AI 3.5 requires the use of Vivado/Vitis 2023.1 and Ubuntu 20.04.

Component	Requirement
ROCm GPU (GPU is optional but strongly recommended for quantization)	AMD ROCm GPUs supporting ROCm v5.5, requires Ubuntu 20.04
CUDA GPU (GPU is optional but strongly recommended for quantization)	NVIDIA GPUs supporting CUDA 11.8 or higher, (eg: NVIDIA P100, V100, A100)
CUDA Driver	NVIDIA-520.61.05 or higher for CUDA 11.8
Docker Version	19.03 or higher, nvidia-docker2
Operating System	Ubuntu 20.04
	CentOS 7.8, 7.9, 8.1
	RHEL 8.3, 8.4
CPU	Intel i3/i5/i7/i9/Xeon 64-bit CPU
	AMD EPYC 7F52 64-bit CPU

Steps:

1.1 Install Virtual Box and Ubuntu 20.04
1.2 Install Vivado/Vitis 2023.1
1.3 Clone or Download PYNQ-DPU_design_contest_3.5
1.4 Edit the DPU settings
1.5 Make the DPU
1.6 Copy the output files to PYNQ

1.1 Install Virtual Box and Ubuntu 20.04

- Download Ubuntu 20.04
  https://releases.ubuntu.com/focal/ubuntu-20.04.6-desktop-amd64.iso
- Install Ubuntu 20.04 in Virtual Box
  - ❖ Refer to the previous tutorial on installing Ubuntu in a Virtual Box
  - ❖ Alternatively: refer to this tutorial

https://ubuntu.com/tutorials/how-to-run-ubuntu-desktop-on-a-virtual-machine-using-virtualbox#1-overview

- ❖ Create a shared folder, enable drag and drop and clip board.
- ❖ Enable external USB access

  >> sudo adduser $USER dialout

- ❖ Install build-essential and curl

  >> sudo apt install curl

[**Note:** if you get Terminal not opening issue, try changing the language; Go to settings -> Region & Language-> Change English (United States) to English (Canada) then restart. If you also cannot do sudo, you need to add YOUR_USERNAME to the /etc/sudoers file (under sudo permissions). For example >> sudo gedit /etc/sudoers
Then add sanka ALL=(ALL:ALL) ALL]

1.2 Install Vivado/Vitis 2023.1

Since Vitis-AI 3.5 requires Vivado /Vitis / Petalinux 2023.1, let us install them

Zynq™ Ultrascale+™

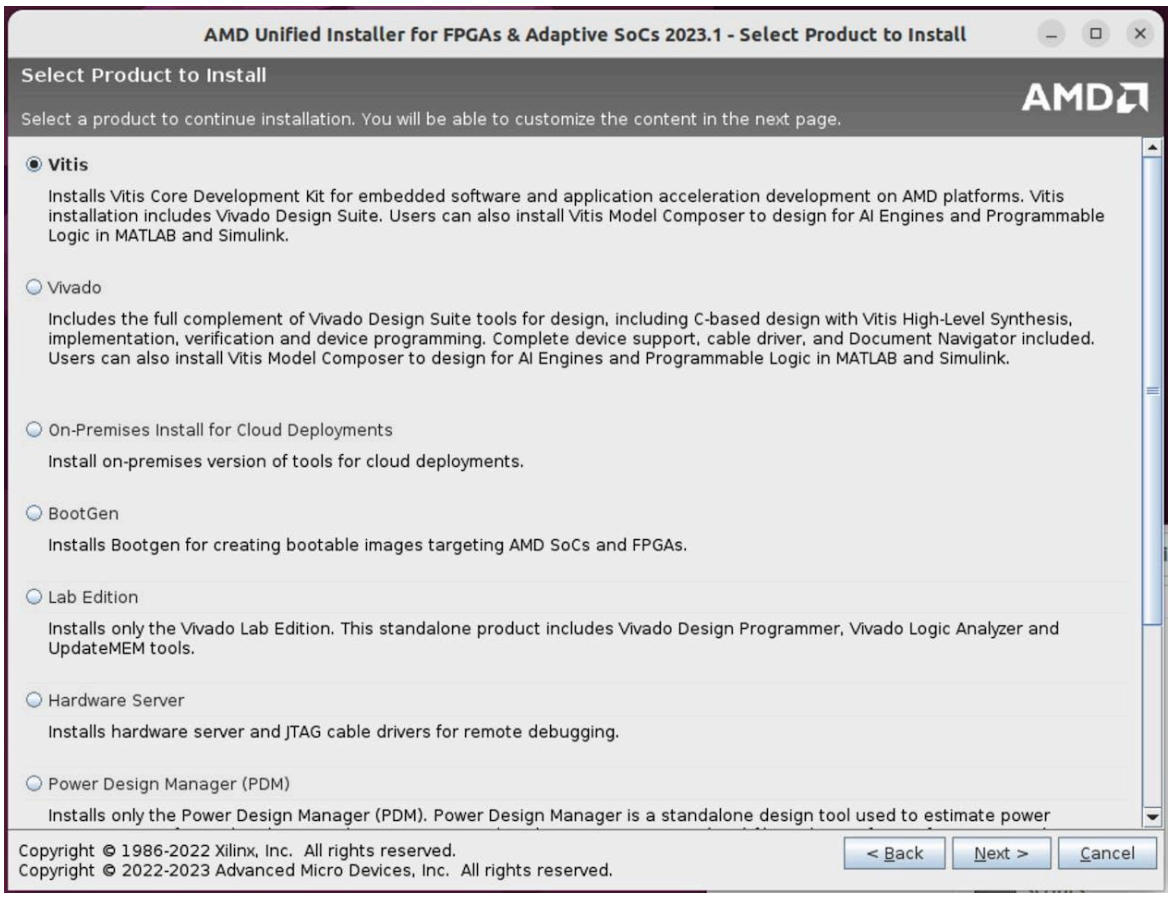
Vitis AI Release Version	DPUCZDX8G IP Version	Software Tools Version	Linux Kernel Version
v3.5	4.1 (not updated*)	Vivado / Vitis / PetaLinux 2023.1	6.1
v3.0	4.1	Vivado / Vitis / PetaLinux 2022.2	5.15

- Download AMD Unified Installer SFD 2023.1 (111GB)
  https://www.xilinx.com/member/forms/download/xef.html?filename=Xilinx_Unified_2023.1_0507_1903.tar.gz

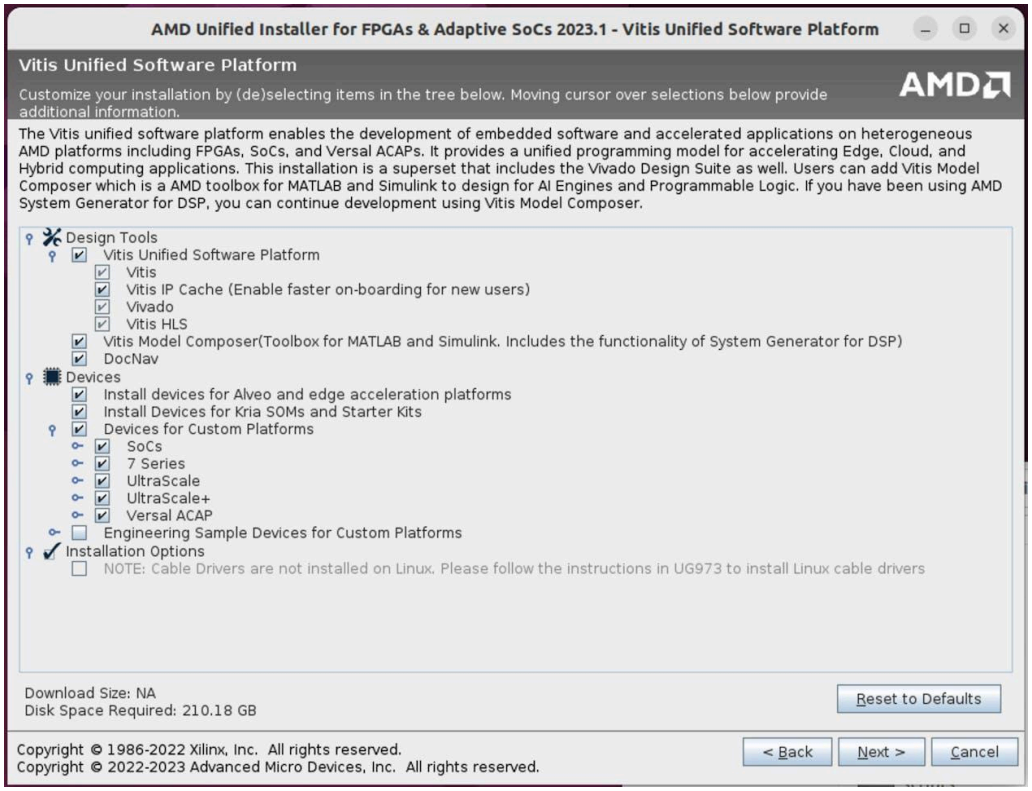
- Once downloaded, extract the installation folder and run the installer, `xsetup`, within it:

```
>>  cd ./Downloads/Xilinx_Unified_2023.1_0507_1903/
>> ~/Downloads/Xilinx_Unified_2023.1_0507_1903$ sudo ./xsetup
```

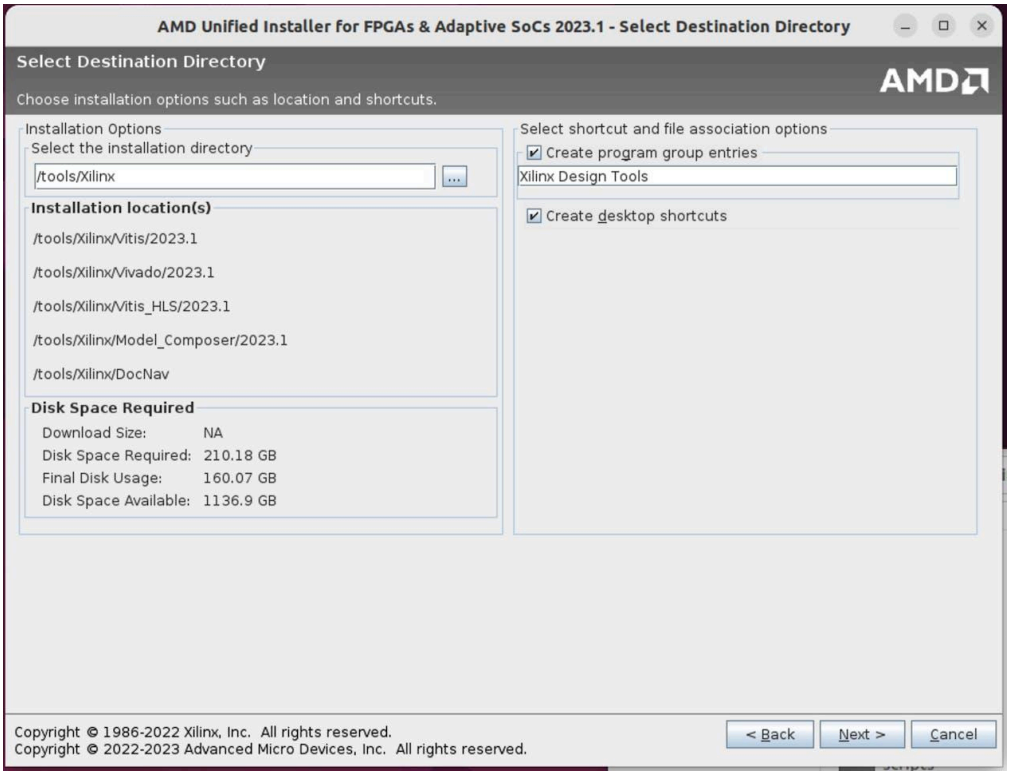
- Select Vitis as the product to install as it's the superset option that will include Vivado (but not PetaLinux, that's a separate install in a later step). Click Next



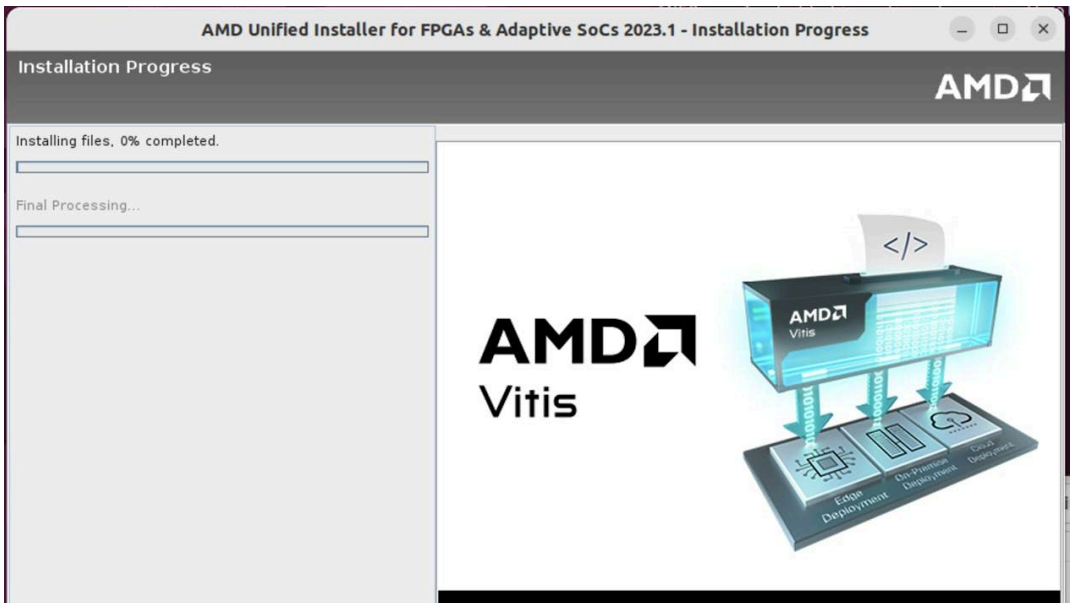
- Select the components you want, next **Agree** on All, you have no choice



- By default, the installer wants to install Vitis and Vivado to `/tools/Xilinx`. We also recommend the default (However you can choose elsewhere if you want).



- Click Next
- Check the Summary and Wait for the installation to complete (over 2hrs)



- At the end of the installation, a prompt will pop up to run an installation script to fill in any missing libraries for Versal ACAP tools. For the sake of completeness, run the installation script:

~/Downloads/Xilinx_Unified_2023.1_0507_1903 >> cd /tools/Xilinx/Vitis/2023.1/scripts/
/tools/Xilinx/Vitis/2023.1/scripts$ sudo ./installLibs.sh

- Install the various Xilinx programmer cable drivers next. Be sure to disconnect any that are currently connected to the host machine prior to running the installation script:

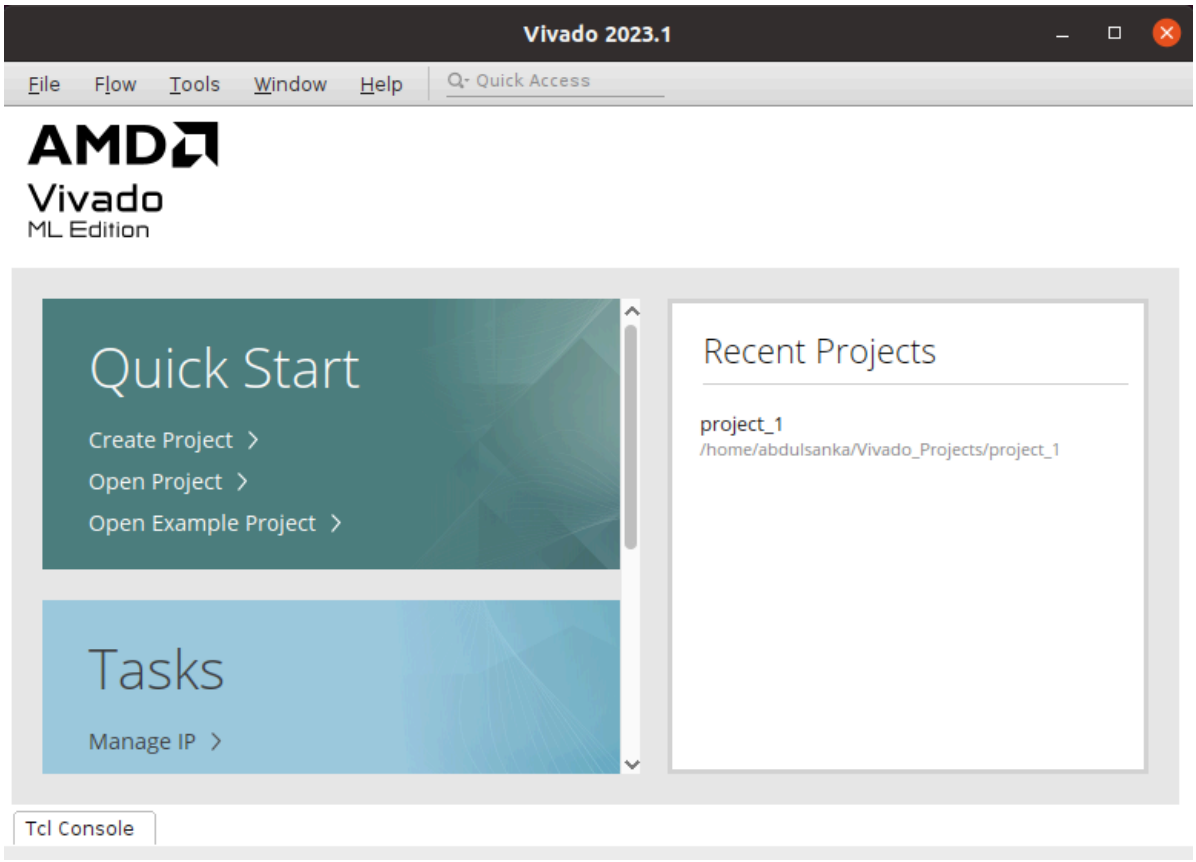
>> cd /tools/Xilinx/Vivado/2023.1/data/xicom/cable_drivers/lin64/install_script/install_drivers/
/tools/Xilinx/Vivado/2023.1/data/xicom/cable_drivers/lin64/install_script/install_drivers
>> sudo ./install_drivers

- Finally, to test the installation and launch Vivado and/or Vitis:

>> source /tools/Xilinx/Vivado/2023.1/settings64.sh
>> vivado

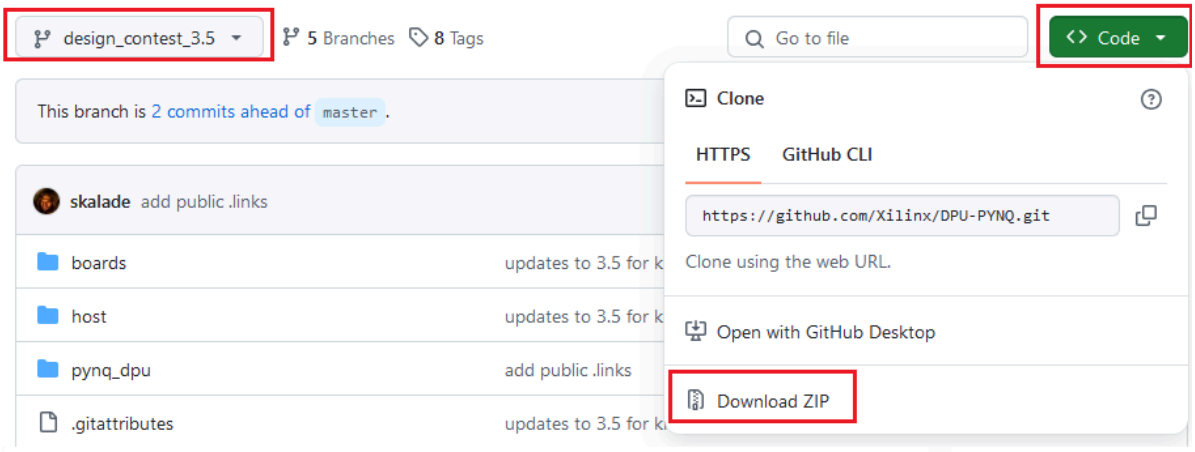
>> source /tools/Xilinx/Vitis/2023.1/settings64.sh
>> vitis

```
abdulsanka@Ubuntu-20-04:~$ source /tools/Xilinx/Vivado/2023.1/settings64.sh
abdulsanka@Ubuntu-20-04:~$ vivado

****** Vivado v2023.1 (64-bit)
  **** SW Build 3865809 on Sun May  7 15:04:56 MDT 2023
  **** IP Build 3864474 on Sun May  7 20:36:21 MDT 2023
  **** SharedData Build 3865790 on Sun May 07 13:33:03 MDT 2023
    ** Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.
    ** Copyright 2022-2023 Advanced Micro Devices, Inc. All Rights Reserved.

start_gui
```



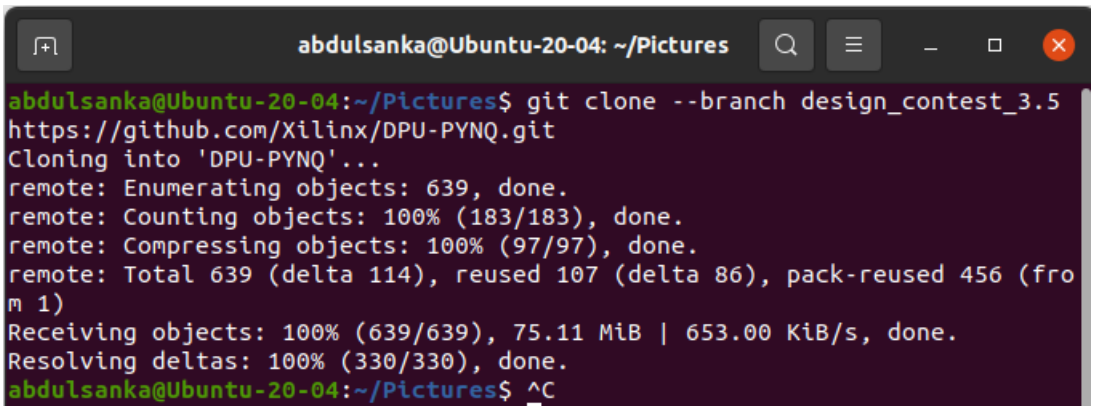
1.3 Clone or Download PYNQ-DPU_design_contest_3.5

- Goto the PYNQ-DPU github, https://github.com/Xilinx/DPU-PYNQ/tree/design_contest_3.5
- click Code and Download the ZIP
- Unzip the file



- Alternatively, you can run the following commands to clone the PYNQ-DPU 3.5 repo. However, I prefer to download and unzip the zip file of the repo

  >> git clone --branch design_contest_3.5 https://github.com/Xilinx/DPU-PYNQ.git



1.4 Edit the DPU settings

The processing system ip version in Vivado 2023.1 is version 3.5 not 3.4 as in the previous Vivado i.e., 2022.2, hence we need to modify the gen_platform tcl file

- Open the gen_platform.tcl file in the /home/abdulsanka/Pictures/DPU-PYNQ/boards/Ultra96v2 folder
- Search 3.4,
- Change the zynq_ultra_ps_e:3.4 to zynq_ultra_ps_e:3.5

```
   ET_BOARD_INTERFACE {Custom} CONFIG.EDK_IPTYPE {PERIPHERAL} ]
42 cell -type ip -vlnv xilinx.com:ip:zynq_ultra_ps_e:3.4 ps_e
```

- Save and close the file

1.5 Build the DPU using the provided MakeFile

After making necessary modifications, it is time to build the DPU design. A MakeFile is provided for the whole process.

- Open the DPU-PYNQ/boards folder

  >> cd DPU-PYNQ/boards

  >> make Board=Ultra96v2

  The make process generates the required files in the boards directory:

  a. Dpu.bit
  b. Dpu.hwh
  c. dpu.xclbin

1.6 Copy the output files to PYNQ-DPU-3.5 Parent directory

- If you have rebuilt the DPU hardware design by yourself, you can see arch.json file inside folder DPU-PYNQ/Boards/<Board>/binary_container_1/link/vivado/vpl/prj/prj.gen/sources_1/bd/dpu/ip/design_1_DPUCZDX8G_1_0/arch.json.
- You can also use the find . -name arch.json command to locate this file.
- Copy the arch.json, dpu.bit, dpu.hwh, and dpu.xclbin to the PYNQ-DPU-3.5 folder

Part 2: Quantization and Compilation of the Model

Steps:

2.1 Install docker
2.2 Prepare docker and Project files
2.3 Pull and start vitis-ai docker tensorflow docker image
2.4 Evaluate and Quantize the model (optional)
2.5 Compile the quantized model

2.1 Install docker

To install the latest stable versions of Docker CLI, Docker Engine, and their dependencies **(Steps 2 and 3 not necessary, pls skip for this lab only)** :

1. download the script

```
>> curl -fsSL https://get.docker.com -o install-docker.sh
```

2. verify the script's content

```
>>  cat install-docker.sh
```

3. run the script with --dry-run to verify the steps it executes

```
>> sh install-docker.sh --dry-run
```

4. run the script either as root, or using sudo to perform the installation.

```
>> sudo sh install-docker.sh
```

5. Test docker by running the hello from docker images

```
>> sudo docker run hello-world
```

2.2 Prepare docker and Project files

- Download and unzip the **gef_lab4_6.zip** in the shared folder folder.
- Copy the **gef_lab4_6** folder to your Ubuntu home directory.
- Open a terminal and run t

```
>> cd gef_lab_6

>> bash prepare_docker.sh
```

- Download the model from vitis-ai model zoo, run one of the following. The python code will download the model from the vitis-ai 3.5 model zoo

>> ./getModel_vai3.5.py --name tf_yolov3_3.5 or

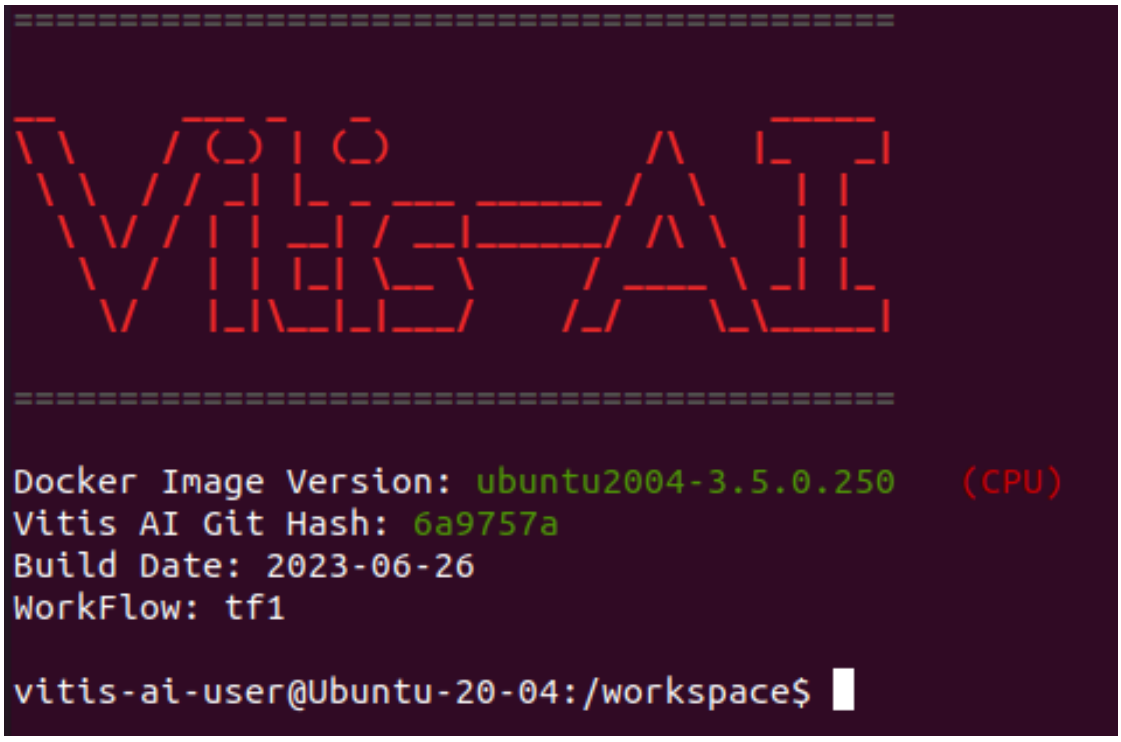
>> python3 getModel_vai3.5.py --name tf_yolov3_3.5

2.3 Pull and start vitis-ai docker tensorflow docker image

- Run the docker_run.sh script to pull and start the vitis-ai docker image. The workspace directory of the docker is the same as your **gef_lab_6** directory

>> sudo ./docker_run.sh xilinx/vitis-ai-tensorflow-cpu:latest

```
abdulsanka@Ubuntu-20-04:~/DPU-PYNQ-3.5/host$ sudo ./docker_run.sh xilinx/vitis-ai-tensorflow-cpu:latest
[sudo] password for abdulsanka:
latest: Pulling from xilinx/vitis-ai-tensorflow-cpu
eaead16dc43b: Pull complete
a567ea3a455d: Pull complete
2f3680aba1ae: Pull complete
a715f4037a7c: Pull complete
682553482ae9: Pull complete
4f4fb700ef54: Pull complete
62410ba017f6: Pull complete
9a63e9b0f03d: Pull complete
b6ae25c9f860: Pull complete
c401a115001f: Pull complete
6ed3c246054a: Pull complete
24872cd27811: Pull complete
Digest: sha256:eb15920932017c0f8d4d356e5e9608de83a8081250bf4af1f5d45d02e8bee274
Status: Downloaded newer image for xilinx/vitis-ai-tensorflow-cpu:latest
docker.io/xilinx/vitis-ai-tensorflow-cpu:latest
```



- Check available conda environment

  >> conda env list

```
vitis-ai-user@Ubuntu-20-04:/workspace$ conda env list
# conda environments:
#
base                  *  /opt/vitis_ai/conda
vitis-ai-tensorflow      /opt/vitis_ai/conda/envs/vitis-ai-tensorflow

vitis-ai-user@Ubuntu-20-04:/workspace$
```

- Activate tensorflow conda environment

  >> conda activate vitis-ai-tensorflow

```
vitis-ai-user@Ubuntu-20-04:/workspace$ conda activate vitis-ai-tensorflow
(vitis-ai-tensorflow) vitis-ai-user@Ubuntu-20-04:/workspace$
```

2.4 Evaluate and Quantize the model ***(not necessary, pls skip for the sake of time)***

- Evaluate the float model

  >> bash code/test/run_eval.sh

```
(vitis-ai-tensorflow) vitis-ai-user@Ubuntu-20-04:/workspace/tf_yolov3_3.5$ bash code/test/run_eval.sh
2024-10-22 22:30:41.707771: W tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could not load dynamic library 'libcuda.so.1'; dler
ile or directory; LD_LIBRARY_PATH: /opt/xilinx/xrt/lib:/usr/lib:/usr/lib/x86_64-linux-gnu
2024-10-22 22:30:41.707818: E tensorflow/stream_executor/cuda/cuda_driver.cc:318] failed call to cuInit: UNKNOWN ERROR (303)
2024-10-22 22:30:41.707852: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not appear to be running on this host
2024-10-22 22:30:41.708314: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was
2024-10-22 22:30:41.714749: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency: 2904000000 Hz
2024-10-22 22:30:41.715187: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x56173e075b70 initialized for platform Host (this do
2024-10-22 22:30:41.715200: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Host, Default Version
WARNING:tensorflow:From /opt/vitis_ai/conda/envs/vitis-ai-tensorflow/lib/python3.6/site-packages/tensorflow_core/python/ops/array_ops.py:1475:
nd will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
  0%|
2024-10-22 22:30:44.691572: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 22151168 exceeds 10% of system memory.
2024-10-22 22:30:44.710966: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 22257792 exceeds 10% of system memory.
  0%|
2024-10-22 22:30:44.887881: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 22151168 exceeds 10% of system memory.
2024-10-22 22:30:44.894059: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 22257792 exceeds 10% of system memory.
  0%|
2024-10-22 22:30:45.061569: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 22151168 exceeds 10% of system memory.
 84%|
```

```
2024-10-22 22:30:45.061569: W tensorflow/core/framework/cpu_allocator_impl.cc:8
100%|
evaluate 4952 images
dog AP: 0.8528021565800722
person AP: 0.8227158470394319
train AP: 0.8563845094226997
sofa AP: 0.7832549753323914
chair AP: 0.6176287587676694
car AP: 0.8644736906371318
pottedplant AP: 0.5167998707874021
diningtable AP: 0.7139159881446634
horse AP: 0.8671286713357269
cat AP: 0.8790726915226089
cow AP: 0.8166664306266627
bus AP: 0.8533804484664665
bicycle AP: 0.8647011954901694
motorbike AP: 0.8485215488672923
bird AP: 0.783689016324824
tvmonitor AP: 0.7717309546891318
sheep AP: 0.8060322471492734
aeroplane AP: 0.8251643774656249
boat AP: 0.7181729441408141
bottle AP: 0.6300390844916984
mAP: 0.7846137703640876
(vitis-ai-tensorflow) vitis-ai-user@Ubuntu-20-04:/workspace/tf_yolov3_3.5$
```

- Quantize the model, go to the quantize folder

  >> cd tf_yolov3_3.5/code/quantize

```
(vitis-ai-tensorflow) vitis-ai-user@Ubuntu-20-04:/workspace/tf_yolov3_3.5$ cd code/quantize
(vitis-ai-tensorflow) vitis-ai-user@Ubuntu-20-04:/workspace/tf_yolov3_3.5/code/quantize$
```

  >> bash quantize.sh (almost 3hrs on CPU)

```
(vitis-ai-tensorflow) vitis-ai-user@Ubuntu-20-04:/workspace$ cd tf_yolov3_3.5/code/quantize
(vitis-ai-tensorflow) vitis-ai-user@Ubuntu-20-04:/workspace/tf_yolov3_3.5/code/quantize$ bash quantize.sh
[INFO]Start Quantize ...
[Config] FLOAT_MODEL: ../../float/yolov3_voc.pb
[Config] INPUT_HEIGHT: 416
[Config] INPUT_WIDTH: 416
/graph_quantizer.cc:1166] [DECENT_WARNING] replace_relu6: 1
2024-10-23 10:56:28.690459: W /home/xbuild/conda-bld/vai_q_tensorflow_1687316851653/work/vai_q_t
/graph_quantizer.cc:1166] [DECENT_WARNING] replace_sigmoid: 0
2024-10-23 10:56:28.690462: W /home/xbuild/conda-bld/vai_q_tensorflow_1687316851653/work/vai_q_t
/graph_quantizer.cc:1166] [DECENT_WARNING] fold_bn_only: 0
2024-10-23 10:56:28.690466: W /home/xbuild/conda-bld/vai_q_tensorflow_1687316851653/work/vai_q_t
/graph_quantizer.cc:1166] [DECENT_WARNING] fold_reshape: 0
2024-10-23 10:56:28.690469: W /home/xbuild/conda-bld/vai_q_tensorflow_1687316851653/work/vai_q_t
/graph_quantizer.cc:1166] [DECENT_WARNING] replace_softmax: 0
******************** Quantization Summary ********************
INFO: Output:
  quantize_eval_model: ../../quantized/quantize_eval_model.pb
[INFO]Quantization finished, results are in ../../quantized
(vitis-ai-tensorflow) vitis-ai-user@Ubuntu-20-04:/workspace/tf_yolov3_3.5/code/quantize$
```

2.5 Compile the quantized model

- Compile the model

>> mkdir tf_yolov3_3.5/compiled

```
(vitis-ai-tensorflow) vitis-ai-user@Ubuntu-20-04:/workspace$ mkdir tf_yolov3_3.5/compiled
(vitis-ai-tensorflow) vitis-ai-user@Ubuntu-20-04:/workspace$
```

>> vai_c_tensorflow --frozen_pb \

tf_yolov3_3.5/quantized/quantize_eval_model.pb --arch arch_ultra96v2.json \

--output_dir tf_yolov3_3.5/compiled --net_name tf_yolov3_3.5 \

--options '{"input_shape": "1,416,416,3"}'

```
(vitis-ai-tensorflow) vitis-ai-user@Ubuntu-20-04:/workspace$ vai_c_tensorflow --frozen_pb \
> tf_yolov3_3.5/quantized/quantize_eval_model.pb  --arch arch_ultra96v2.json \
> --output_dir tf_yolov3_3.5/compiled  --net_name tf_yolov3_3.5 \
> --options  '{"input_shape": "1,416,416,3"}'
**********************************************
* VITIS_AI Compilation - Xilinx Inc.
**********************************************
[INFO] Namespace(batchsize=1, inputs_shape=['1,416,416,3'], layout='NHWC', model_files=['tf_y
v3_3.5/quantized/quantize_eval_model.pb'], model_type='tensorflow', named_inputs_shape=None,
_filename='/tmp/tf_yolov3_3.5_0x101000017010404_org.xmodel', proto=None)
in_shapes: [[1, 416, 416, 3]]
[INFO] tensorflow model: /workspace/tf_yolov3_3.5/quantized/quantize_eval_model.pb
[INFO] parse raw model     :100%|                  | 294/294 [00:00<00:00, 19444.10it/s]
[INFO] infer shape (NHWC)  :100%|                  | 459/459 [00:00<00:00, 1577.40it/s]
[INFO] perform level-0 opt :100%|                  | 3/3 [00:00<00:00, 222.63it/s]
[INFO] perform level-1 opt :100%|                  | 8/8 [00:00<00:00, 271.46it/s]
[INFO] generate xmodel     :100%|                  | 290/290 [00:00<00:00, 396.33it/s]
[INFO] dump xmodel: /tmp/tf_yolov3_3.5_0x101000017010404_org.xmodel
[UNILOG][INFO] Compile mode: dpu
[UNILOG][INFO] Debug mode: null
[UNILOG][INFO] Target architecture: DPUCZDX8G_ISA1_B1600_0101000017010404
[UNILOG][INFO] Graph name: quantize_eval_model, with op num: 590
[UNILOG][INFO] Begin to compile...
[UNILOG][INFO] Total device subgraph number 5, DPU subgraph number 1
[UNILOG][INFO] Compile done.
[UNILOG][INFO] The meta json is saved to "/workspace/tf_yolov3_3.5/compiled/meta.json"
[UNILOG][INFO] The compiled xmodel is saved to "/workspace/tf_yolov3_3.5/compiled/tf_yolov3_3
xmodel"
[UNILOG][INFO] The compiled xmodel's md5sum is 7de5e164b99b98223dfae3d661c81cc2, and has been
ved to "/workspace/tf_yolov3_3.5/compiled/md5sum.txt"
(vitis-ai-tensorflow) vitis-ai-user@Ubuntu-20-04:/workspace$
```

Copy the compiled x-model to the PYNQ-DPU-3.5 (same folder as the .bit and .hwh files)

Part 3: Inference on the Target Board (Deployment)

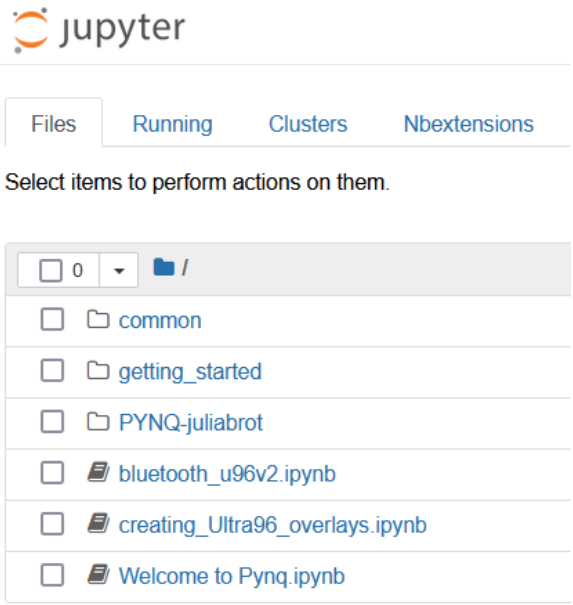
Steps:

3.1 Access board via Jupyter lab and Putty
3.2 Copy the compiled x-model to the target (Ultra96 board) using Samba
3.3 Install pynq_dpu on the target board
3.4 Pull and start vitis-ai docker tensorflow docker image
3.5 Evaluate and Quantize the model (optional)
3.6 Compile the quantized model

3.1 Access board via Jupyter lab/Putty and Connect to Wifi

In this part, it is assumed that the board has been set up with PYNQ. If the board has not been set up, refer to our previous lab to download and burn the PYNQ 3.0.1 image on the SD card using Balena Etcher. Insert the SD card into the board, connect the cables and power on the board.

- Open Device manager to find the COM PORT number
- Access the board using Putty
- Open a browser, use **192.168.3.1:9090** to access the Jupyter lab on the board. Use password = **xilinx**

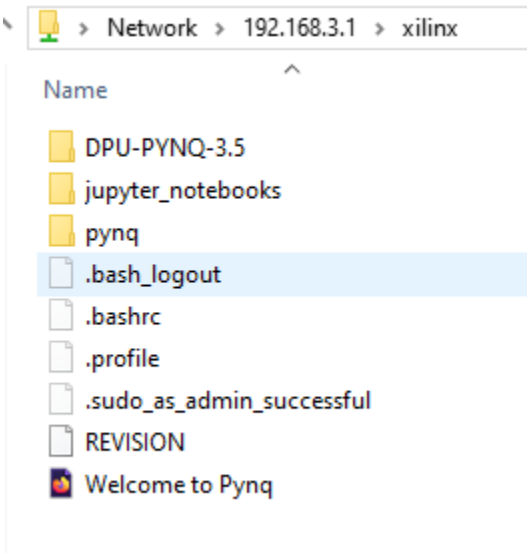


- Open and run the wifi.ipynb notebook in the common folder to connect the board to the internet

3.2 Copy the compiled x-model and pyn_dpu notebooks to the target (Ultra96 board) using Samba

- On Ubuntu: open a file browser, click **other locations** and type **smb://192.168.3.1/xilinx** in the Connect to Server box: (if connected via USB Ethernet)
  username is **xilinx** and the password is **xilinx**
- On Windows: Open File explorer and type **\\192.168.3.1\xilinx,** [If you have issues, refer to Appendix A at the end of this file]

  Use, username = **xilinx** also, password = **xilinx**
- Copy and paste PYNQ-DPU-3.5 folder (containing the hardware files and the pynq setup files) to the /home/xilinx directory on the board
- Copy and paste the **pynq_dpu** in the notebooks folder to **jupyter_notebooks** folder on the board



3.3 Install pynq_dpu on the target board

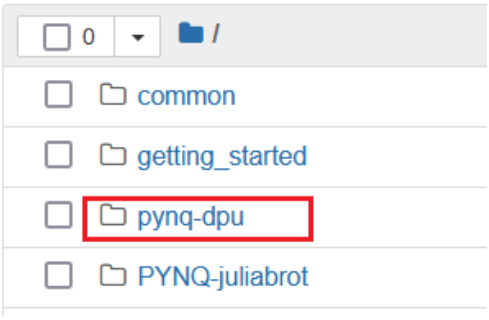
- Open a terminal in Jupyter lab, and run

  >> pip3 install pynq-dpu --no-build-isolation

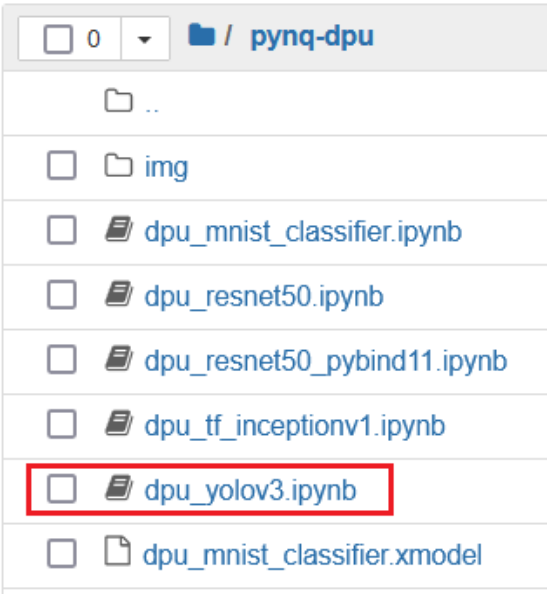
  >> pip3 install /home/xilinx/DPU-PYNQ-3.5 --no-build-isolation

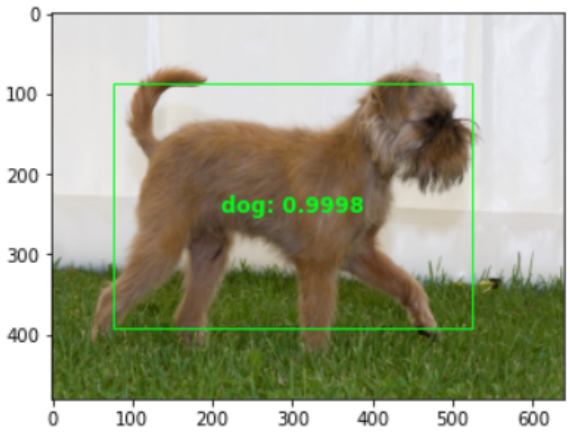
3.4 Perform inference with Yolov3

- From the Jupyter home, open **pynq-dpu** folder



- Open and run **dpu_yolov3.ipynb** step by step





- Change the image number and repeat, remember do not need to execute the first two cells of the notebook

Exercise: Try the object detection using a camera

Appendix A: Troubleshoot Samba Access Denied Samba Issues

If you have access denied issue while opening Samba. Try these steps to clear the problem

a. Change password

    >> sudo smbpasswd -a xilinx

b. Open the Samba configuration file (`smb.conf`) to check the share settings, scroll to the end of the file and ensure that the shared directory is correctly defined as below, and permissions are set appropriately as below, for example (add public=yes):

    >> sudo nano /etc/samba/smb.conf

```
[xilinx]
path = /home/xilinx
browsable =yes
writable = yes
guest ok = yes
read only = no
valid users = samba xilinx
follow symlinks = yes
wide links = yes
create mask = 664
directory mask = 755
force user = xilinx
public = yes
```

Type Ctrl+O to save and Ctrl+x to quit

c. Set directory permission

    >> sudo chmod 777 /home/xilinx

d. Restart Samba

    >> sudo systemctl restart smbd nmbd

e. Clear Cached Credentials on Windows: Open command prompt and run the command

net use \\192.168.3.1\xilinx /delete

This command removes any stored credentials for that share.

f. Re-open Samba by typing **\\192.168.3.1\xilinx** in a file explorer