



Bridge of Life
Education

Lab-D

Fast, Scalable Quantized Neural Network Inference on FPGAs, FINN

National Taiwan University

2021 Fall

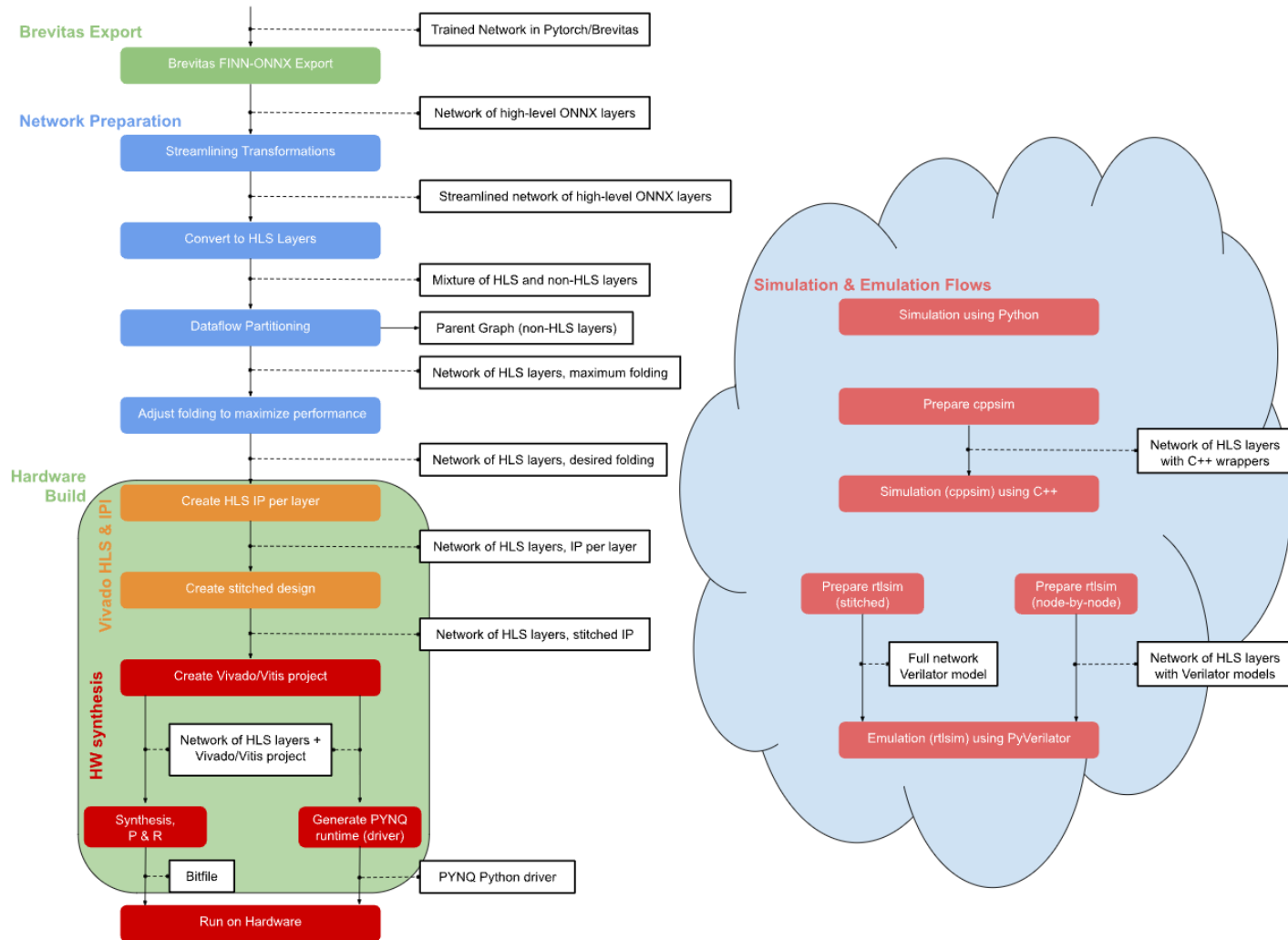
Outline

- Part1: End-to-End FINN Flow for a Simple Convolutional Net
- Part2 : VGG on CIFAR-100
- Part3: Performance Improvement
- Questions
- Report & Submission

Outline

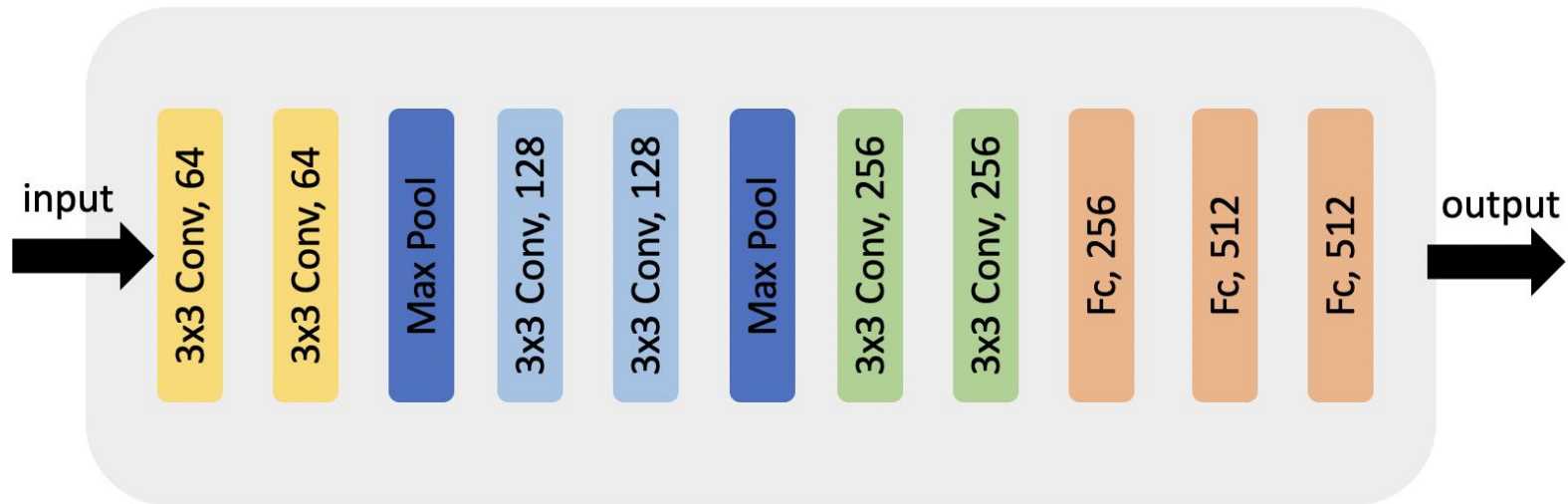
- Part1: End-to-End FINN Flow for a Simple Convolutional Net
- Part2 : VGG on CIFAR-100
- Part3: Performance improvement
- Report & Submission

Quick Recap of the End-to-End Flow



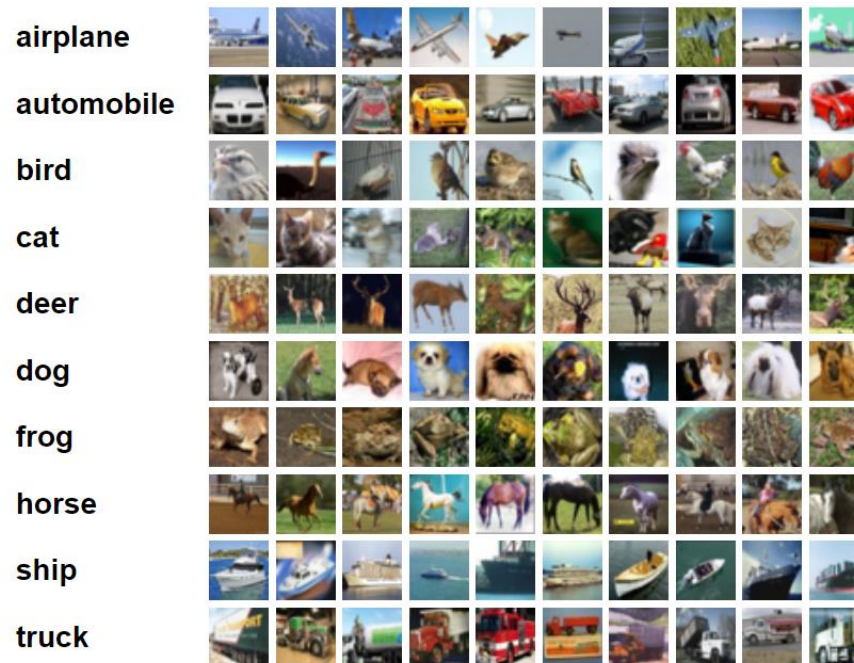
Quick Introduction to the CNV-w1a1 Network

- Input size: 32x32 8bits RGB Data
- Weights and Activations are binary values.
- Output is a number that represents the result predicted by our model.



CIFAR-10 dataset

- The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class.
- There are 50000 training images and 10000 test images.

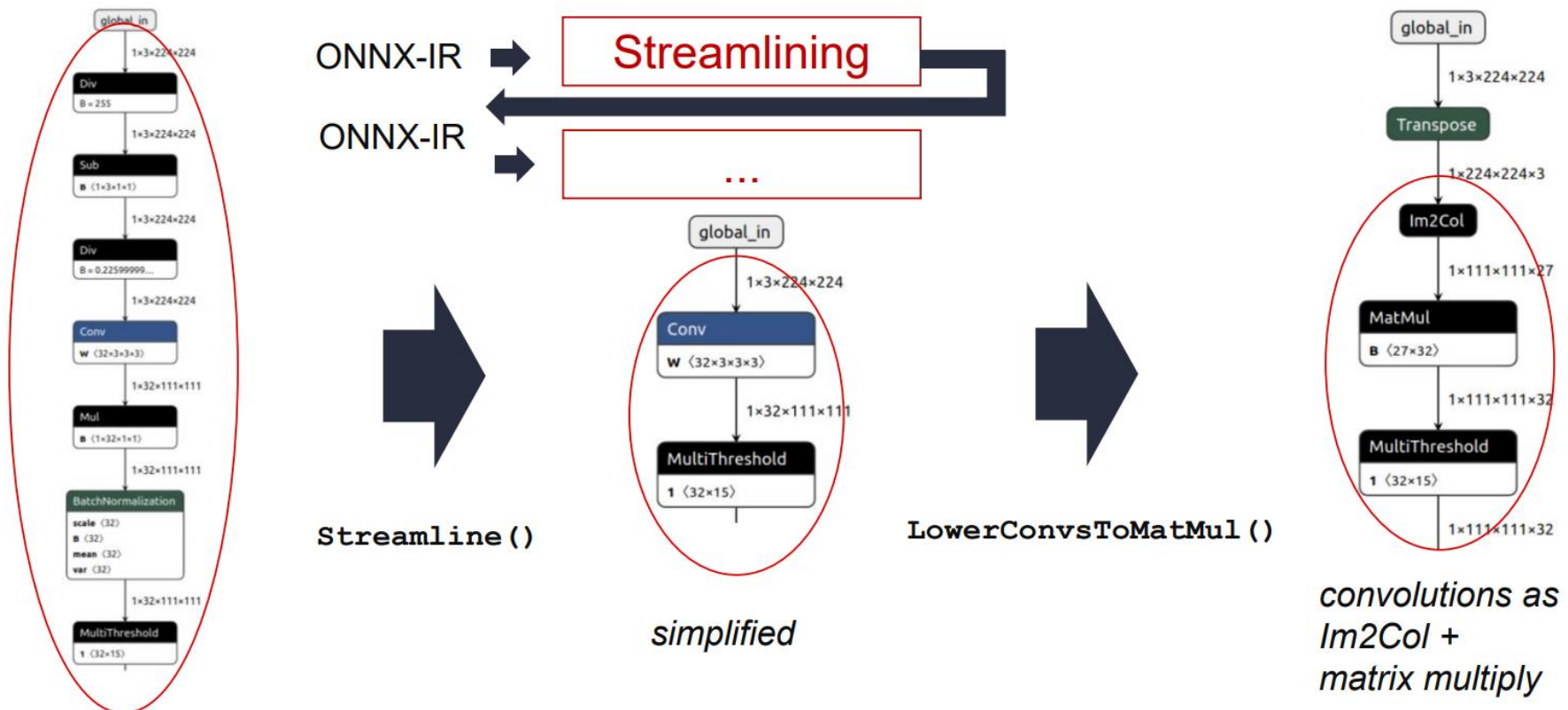


Brevitas Export, FINN Import and Tidy-Up

- We will start by exporting the pretrained CNV-w1a1 network to ONNX.
- Run the "tidy-up" transformations to have a first look at the topology.

```
1 import onnx
2 from finn.util.test import get_test_model_trained
3 import brevitas.onnx as bo
4 from finn.core.modelwrapper import ModelWrapper
5 from finn.transformation.infer_shapes import InferShapes
6 from finn.transformation.fold_constants import FoldConstants
7 from finn.transformation.general import GiveReadableTensorNames, GiveUniqueNodeNames, RemoveStaticGraphInputs
8
9 cnv = get_test_model_trained("CNV", 1, 1)
10 bo.export_finn_onnx(cnv, (1, 3, 32, 32), build_dir + "/end2end_cnv_w1a1_export.onnx")
11 model = ModelWrapper(build_dir + "/end2end_cnv_w1a1_export.onnx")
12 model = model.transform(InferShapes())
13 model = model.transform(FoldConstants())
14 model = model.transform(GiveUniqueNodeNames())
15 model = model.transform(GiveReadableTensorNames())
16 model = model.transform(RemoveStaticGraphInputs())
17 model.save(build_dir + "/end2end_cnv_w1a1_tidy.onnx")
```

Transformation



Transformation

- Pre-processing
 - Divides the input uint8 data by 255 so the inputs to the CNV-w1a1 network are bounded between [0, 1]
- Post-processing
 - Takes the output of the network and returns the index (0-9) of the image category with the highest probability (top-1).

Dataflow Partitioning

- We'll first convert the layers that we can put into the FPGA into their HLS equivalents and separate them out into a dataflow partition.

```
model = ModelWrapper(build_dir + "/end2end_cnv_w1a1_streamlined.onnx")
model = model.transform(to_hls.InferBinaryStreamingFCLayer(mem_mode))
model = model.transform(to_hls.InferQuantizedStreamingFCLayer(mem_mode))
# TopK to LabelSelect
model = model.transform(to_hls.InferLabelSelectLayer())
# input quantization (if any) to standalone thresholding
model = model.transform(to_hls.InferThresholdingLayer())
model = model.transform(to_hls.InferConvInpGen())
model = model.transform(to_hls.InferStreamingMaxPool())
# get rid of Reshape(-1, 1) operation between hlslib nodes
model = model.transform(RemoveCNVtoFCFlatten())
# get rid of Tranpose -> Tranpose identity seq
model = model.transform(absorb.AbsorbConsecutiveTransposes())
# infer tensor data layouts
model = model.transform(InferDataLayouts())
parent_model = model.transform(CreateDataflowPartition())
parent_model.save(build_dir + "/end2end_cnv_w1a1_dataflow_parent.onnx")
sdp_node = parent_model.get_nodes_by_op_type("StreamingDataflowPartition")[0]
sdp_node = getCustomOp(sdp_node)
dataflow_model_filename = sdp_node.get_nodeattr("model")
# save the dataflow partition with a different name for easier access
dataflow_model = ModelWrapper(dataflow_model_filename)
dataflow_model.save(build_dir + "/end2end_cnv_w1a1_dataflow_model.onnx")
```

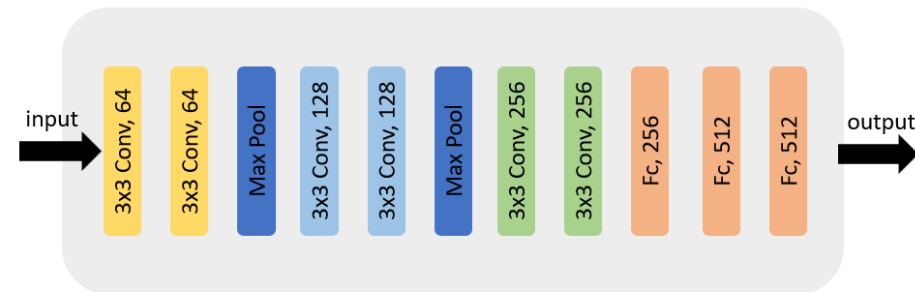
Adjust folding factor

- We have to set the folding factors for certain layers to adjust the performance of our accelerator.

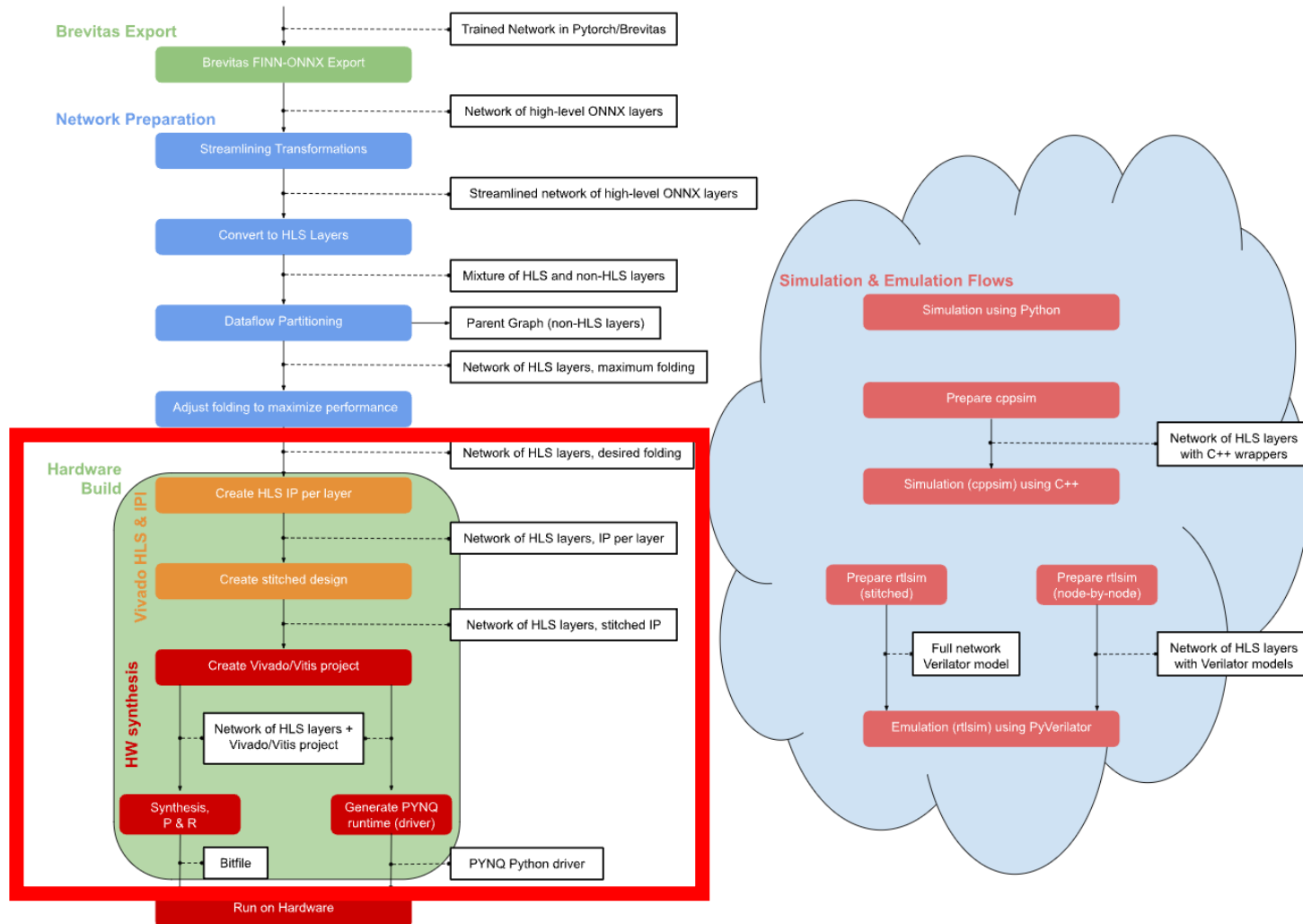
```
model = ModelWrapper(build_dir + "/end2end_cnv_w1a1_dataflow_model.onnx")
fc_layers = model.get_nodes_by_op_type("StreamingFCLayer_Batch")
# each tuple is (PE, SIMD, in_fifo_depth) for a Layer
folding = [
    (16, 3, 128),
    (32, 32, 128),
    (16, 32, 128),
    (16, 32, 128),
    (4, 32, 81),
    (1, 32, 2),
    (1, 4, 2),
    (1, 8, 128),
    (5, 1, 3),
]
for fcl, (pe, simd, ififodepth) in zip(fc_layers, folding):
    fcl_inst = getCustomOp(fcl)
    fcl_inst.set_nodeattr("PE", pe)
    fcl_inst.set_nodeattr("SIMD", simd)
    fcl_inst.set_nodeattr("inFIFODepth", ififodepth)

# use same SIMD values for the sliding window operators
swg_layers = model.get_nodes_by_op_type("ConvolutionInputGenerator")
for i in range(len(swg_layers)):
    swg_inst = getCustomOp(swg_layers[i])
    simd = folding[i][1]
    swg_inst.set_nodeattr("SIMD", simd)

model = model.transform(GiveUniqueNodeNames())
model.save(build_dir + "/end2end_cnv_w1a1_folded.onnx")
```



Hardware Build



Hardware Generation

- Specify the target board and clock period
- ZynqBuild is the hardware generation function
- This step may take about 120 minutes depending on your host computer.

```
1 test_pynq_board = "Pynq-Z2"
2 target_clk_ns = 10
3
4 from finn.transformation.fpgadataflow.make_zynq_proj import ZynqBuild
5 model = ModelWrapper(build_dir+"/end2end_cnv_w1a1_folded.onnx")
6 model = model.transform(ZynqBuild(platform = test_pynq_board, period_ns = target_clk_ns))
7 model.save(build_dir + "/end2end_cnv_w1a1_synth.onnx")
```

Deployment and Remote Execution

- Set up our pynq-board's IP and PORT

```
import os
```

```
# set up the following values according to your own environment
# FINN will use ssh to deploy and run the generated accelerator
ip = os.getenv("PYNQ_IP", "192.168.2.99")
username = os.getenv("PYNQ_USERNAME", "xilinx")
password = os.getenv("PYNQ_PASSWORD", "xilinx")
port = os.getenv("PYNQ_PORT", 22)
target_dir = os.getenv("PYNQ_TARGET_DIR", "/home/xilinx/finn_cnv_end2end_example")
# set up ssh options to only allow publickey authentication
options = "-o PreferredAuthentications=publickey -o PasswordAuthentication=no"

# test access to PYNQ board
! ssh {options} {username}@{ip} -p {port} cat /var/run/motd.dynamic
```

PYNQ-Z2 SSH

- We have set up some PYNQ-Z2 boards in the lab.
- IP: 140.112.207.155
 - SSH port: 2xx00
 - xx = PYNQ number (ex: for PYNQ03, xx = 03)
 - Username: xilinx Password: xilinx

Group	PYNQ number
Group1	01
Group2	02
Group3	03
Group4	04

Download the dataset

- Connect to PYNQ-board and download the dataset

```
! ssh {options} -t {username}@{ip} -p {port}  
'echo {password} | sudo -S pip3 install git+https://github.com/fbcotter/dataset\_loading.git@0.0.4#egg=dataset\_loading'
```


Validating the Accuracy on a PYNQ Board

- Connect to PYNQ-board
- Validating the Accuracy

```
! ssh {options} -t {username}@{ip} -p {port}  
'cd {target_dir_pynq}; echo {password} | sudo -S python3.6 validate.py --dataset cifar10 --batchsize 1000'
```

```
[sudo] password for xilinx: Tar File found in dest_dir. Not Downloading again  
Extracting Python CIFAR10 data.  
Files extracted  
batch 1 / 10 : total OK 851 NOK 149  
batch 2 / 10 : total OK 1683 NOK 317  
batch 3 / 10 : total OK 2522 NOK 478  
batch 4 / 10 : total OK 3370 NOK 630  
batch 5 / 10 : total OK 4207 NOK 793  
batch 6 / 10 : total OK 5044 NOK 956  
batch 7 / 10 : total OK 5887 NOK 1113  
batch 8 / 10 : total OK 6728 NOK 1272  
batch 9 / 10 : total OK 7570 NOK 1430  
batch 10 / 10 : total OK 8419 NOK 1581  
Final accuracy: 84.190000
```

Screen Dump this result

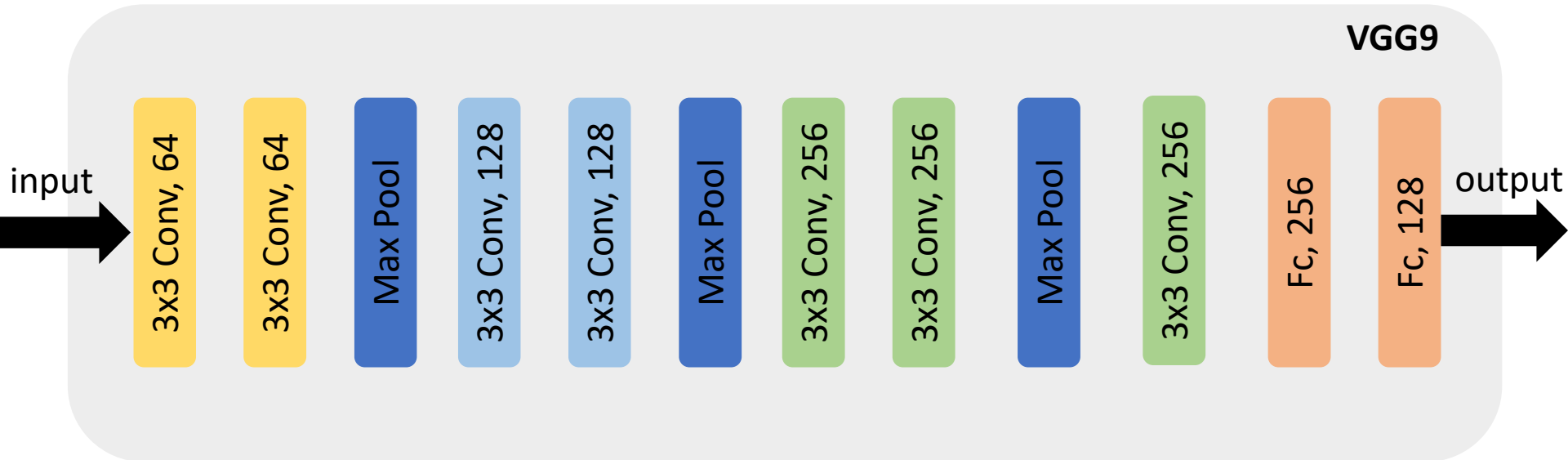
Part1 Requirements

- Screen Dump
 - Each ONNX Graph
 - Model Accuracy
- Describe your observations and understanding of each transformation

Outline

- Part1: End-to-End FINN Flow for a Simple Convolutional Net
- **Part2 : VGG on CIFAR-100**
- Part3: Performance improvement
- Report & Submission

VGG9



CIFAR100

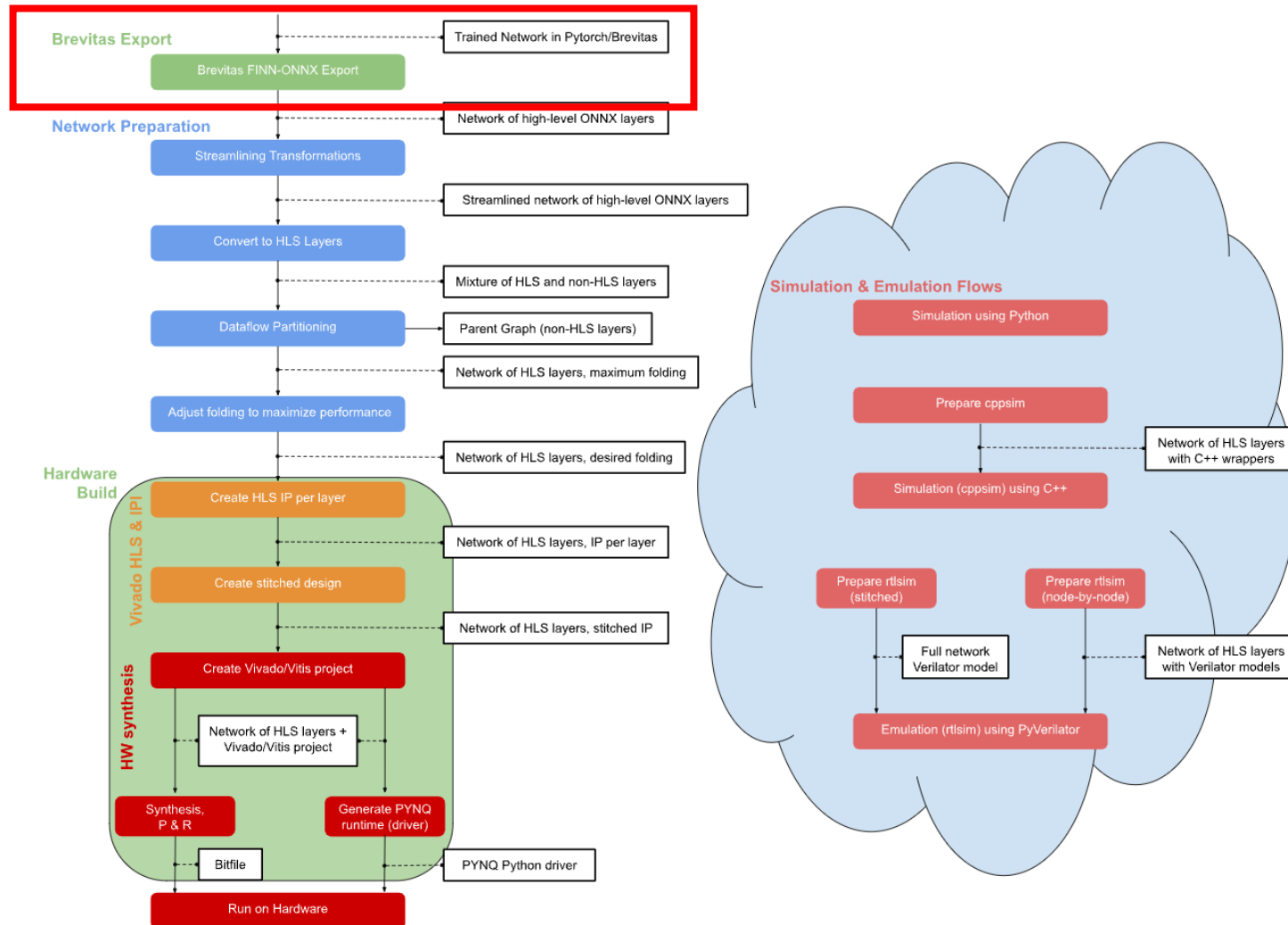
- It has 100 classes containing 600 images each.
- There are 500 training images and 100 testing images per class.

Classes

beaver, dolphin, otter, seal, whale
aquarium fish, flatfish, ray, shark, trout
orchids, poppies, roses, sunflowers, tulips
bottles, bowls, cans, cups, plates
apples, mushrooms, oranges, pears, sweet peppers
clock, computer keyboard, lamp, telephone, television
bed, chair, couch, table, wardrobe
bee, beetle, butterfly, caterpillar, cockroach
bear, leopard, lion, tiger, wolf
bridge, castle, house, road, skyscraper
cloud, forest, mountain, plain, sea
camel, cattle, chimpanzee, elephant, kangaroo
fox, porcupine, possum, raccoon, skunk
crab, lobster, snail, spider, worm
baby, boy, girl, man, woman
crocodile, dinosaur, lizard, snake, turtle
hamster, mouse, rabbit, shrew, squirrel
maple, oak, palm, pine, willow
bicycle, bus, motorcycle, pickup truck, train
lawn-mower, rocket, streetcar, tank, tractor

<https://www.cs.toronto.edu/~kriz/cifar.html>

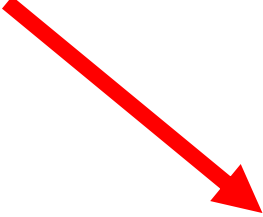
Model Prepare



Define Network Architecture

- We need to define our network architecture in CNV.py

Change the following simple Network example to VGG9



```
32  CNV_OUT_CH_POOL = [(64, False), (64, True), (128, False), (128, True), (256, False), (256, False)]
33  INTERMEDIATE_FC_FEATURES = [(256, 512), (512, 512)]
34  LAST_FC_IN_FEATURES = 512
35  LAST_FC_PER_OUT_CH_SCALING = False
36  POOL_SIZE = 2
37  KERNEL_SIZE = 3
```

CNV.py

Specify Dataset

- Change Dataset from CIFAR-10 to CIFAR-100

```
108     if dataset == 'CIFAR10':
109         train_transforms_list = [transforms.RandomCrop(32, padding=4),
110                                 transforms.RandomHorizontalFlip(),
111                                 transforms.ToTensor()]
112         transform_train = transforms.Compose(train_transforms_list)
113         builder = CIFAR10
114
115     elif dataset == 'MNIST':
116         transform_train = transform_to_tensor
117         builder = MirrorMNIST
118     else:
119         raise Exception("Dataset not supported: {}".format(args.dataset))
```

trainer.py

Configuration File

- Create a configuration file for VGG9
- Bit width of weigh and activation is 1. (Binary Network)

```
1  [MODEL]
2  ARCH: CNV
3  PRETRAINED_URL: https://github.com/Xilinx/brevitas/releases/download/bnn\_pynq-r0/cnv\_1w1a-758c8fef.pth
4  EVAL_LOG: https://github.com/Xilinx/brevitas/releases/download/cnv\_test\_ref-r0/cnv\_1w1a\_eval-ea2f0427.txt
5  DATASET: CIFAR10
6  IN_CHANNELS: 3
7  NUM_CLASSES: 10
8
9  [QUANT]
10 WEIGHT_BIT_WIDTH: 1
11 ACT_BIT_WIDTH: 1
12 IN_BIT_WIDTH: 8
13
```

Command

- Training
 - `python3 bnn_pynq_train.py --network CNV_1W1A --experiments ./experiments`
- Testing
 - `python3 bnn_pynq_train.py --evaluate --network CNV_1W1A --resume ./ experiments /CNV_1W1A_1W1A_xxxxxx/checkpoints/best.tar`

Export Network

- Add the following two lines of code in train.py

```
if args.resume:
    print('Loading model checkpoint at: {}'.format(args.resume))
    package = torch.load(args.resume, map_location='cpu')
    model_state_dict = package['state_dict']
    model.load_state_dict(model_state_dict, strict=args.strict)
    vgg = model
    bo.export_finn_onnx(vgg, (1, 3, 32, 32), "./vgg_w1a1.onnx")
```

trainer.py

Part2 Requirements

- File
 - vgg_w1a1.onnx
- Screen Dump
 - Accuracy of this model on both Server and FPGA

Outline

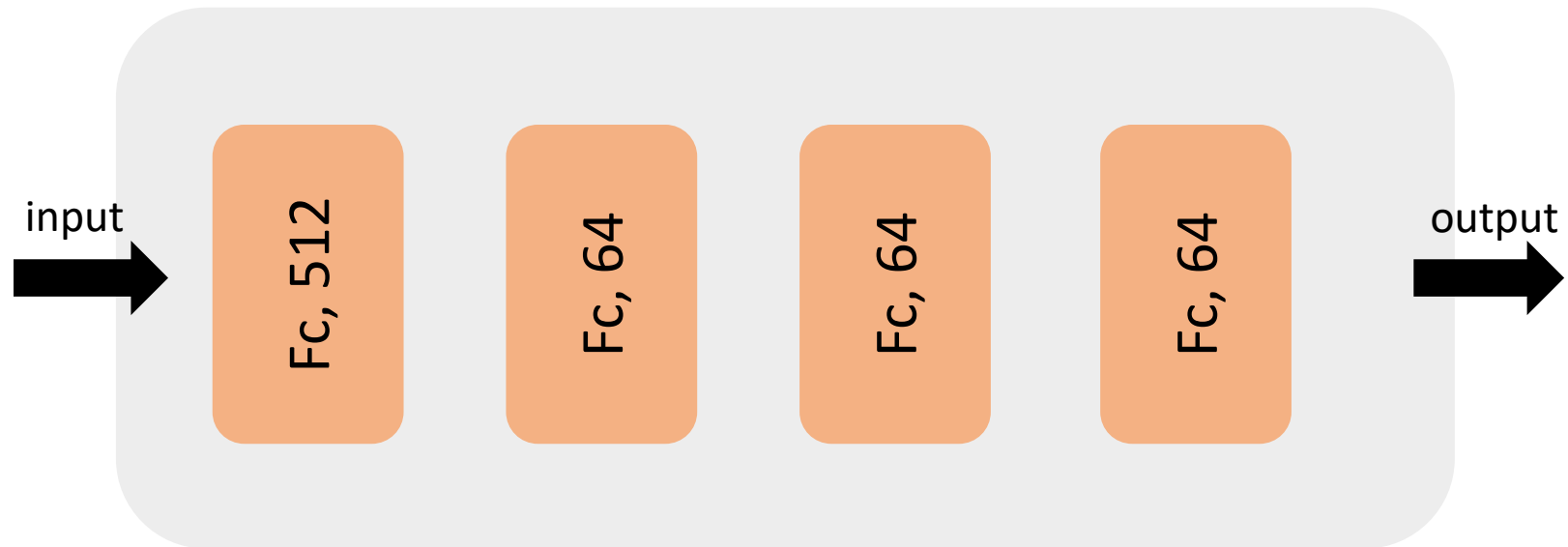
- Part1: End-to-End FINN Flow for a Simple Convolutional Net
- Part2 : VGG on CIFAR-100
- **Part3: Performance improvement**
- Report & Submission

Handwritten Digit Classification

- Fully-connected network trained on the MNIST data set



Network Architecture



Folding: Adjusting the Parallelism

- We can set folding factors for each layer, controlled by the **PE** (parallelization over outputs) and **SIMD** (parallelization over inputs) parameters.

```
fc_layers = model.get_nodes_by_op_type("StreamingFCLayer_Batch")
# (PE, SIMD, in_fifo_depth, out_fifo_depth, ramstyle) for each layer
config = [
    (16, 49, 16, 64, "block"),
    (8, 8, 64, 64, "auto"),
    (8, 8, 64, 64, "auto"),
    (10, 8, 64, 10, "distributed"),
]
for fcl, (pe, simd, ififo, ofifo, ramstyle) in zip(fc_layers, config):
    fcl_inst = getCustomOp(fcl)
    fcl_inst.set_nodeattr("PE", pe)
    fcl_inst.set_nodeattr("SIMD", simd)
    fcl_inst.set_nodeattr("inFIFOdepth", ififo)
    fcl_inst.set_nodeattr("outFIFOdepth", ofifo)
    fcl_inst.set_nodeattr("ram_style", ramstyle)

# set parallelism for input quantizer to be same as first layer's SIMD
inp_qnt_node = model.get_nodes_by_op_type("Thresholding_Batch")[0]
inp_qnt = getCustomOp(inp_qnt_node)
inp_qnt.set_nodeattr("PE", 49)
```


Throughput Test on PYNQ Board

- FINN provides the `throughput_test_remote` function for the throughput test.

```
from finn.core.throughput_test import throughput_test_remote

model = ModelWrapper(build_dir + "/tfc_w1_a1_pynq_deploy.onnx")
res = throughput_test_remote(model, 10000)
print("Network metrics:")
for key in res:
    print(str(key) + ": " + str(res[key]))
```

```
Network metrics:
runtime[ms]: 10.43391227722168
throughput[images/s]: 958413.2714850444
DRAM_in_bandwidth[Mb/s]: 751.3960048442748
DRAM_out_bandwidth[Mb/s]: 0.9584132714850445
fclk[mhz]: 100.0
N: 10000
```

Adjusting the Parallelism

- Try to set Layer2 PE and SIMD number to 1 (Exp1)
- Try your best to achieve the best performance by adjusting the PE and SIMD. (Exp2)
 - Try 3 sets at least.(List them in a table)

Part3 Requirements

- Screen Dump
 - Model Accuracy (At least 90%)
 - All of your performance results
- List all your results in a table
- Write down your observations and explain possible reasons for this result (Exp1)
- Explain your methodology for adjusting the folding factors and analyze the results. (Exp2)

Outline

- Part1: End-to-End FINN Flow for a Simple Convolutional Net
- Part2 : VGG on CIFAR-100
- Part3: Performance Improvement
- Questions
- Report & Submission

Question1

- Please draw the circuit diagrams for the following two PE and SIMD configuration.
 - PE: 4 SIMD: 1
 - PE: 8 SIMD: 8

Question2

- Following the previous question, suppose we need to calculate a 8×8 matrix multiplication; please calculate the latency for the two configurations.
- You have to draw some pictures to explain how the calculations are allocated to the hardware.

Question3

- For a $H \times W \times C_i$ feature map, and given C_0 stride 1 filters of size $K \times K$, what is the shape of the filter matrix and image matrix?

Question4

- In this section, the image feature map in the figure is of size **2 x 3**. However the real application the image might be of size **224 x 224**. In this case, can we deal with the whole image? Are there any practical solutions in the perspective of hardware design?

Grading

- Part1: 20%
- Part2: 30%
- Part3: 30%
- Questions : 20%

Submission (1/2)

- Hierarchy:
 - GroupID_labD/
 - Part1/
 - All of the Screen Dumps
 - Part2/
 - All of the Screen Dumps
 - vgg9_w1a1.onnx
 - Part3/
 - All of the Screen Dumps
 - Question.pdf
 - Report.pdf

The file name of the screen dump should be comprehensible.

Submission (2/2)

- Compress all above files in a single zip file named GroupID_labD.zip
- Submit to NTU COOL
- Deadline: 12/05 (Sun.) 23:59
 - 30% off for the late submission penalty

AnyDesk

[Why AnyDesk](#) ▾[Solutions](#) ▾[Pricing](#)[Services](#) ▾[Company](#) ▾[Start Free Trial](#)[Downloads](#)

Access. Now.

Access any device at any time. From anywhere. Always secure and fast.

[Download Now](#)

Windows (3.8 MB)

AnyDesk is free for personal use.

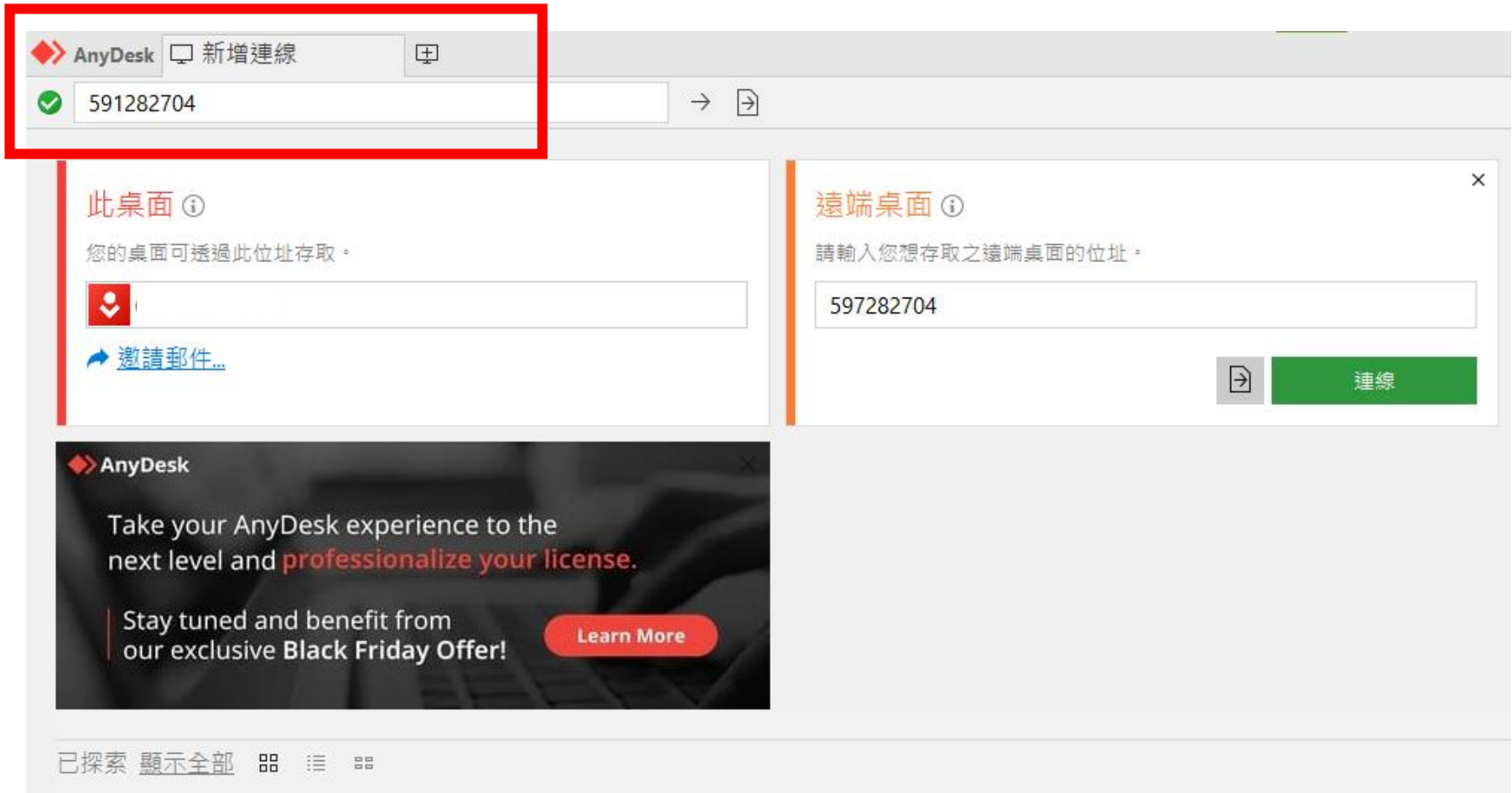
Professional users need a license. Get started with a free 14 day trial now.

[Start Free Trial](#)[Order Now](#)

<https://anydesk.com/en>

Server Login(1/)

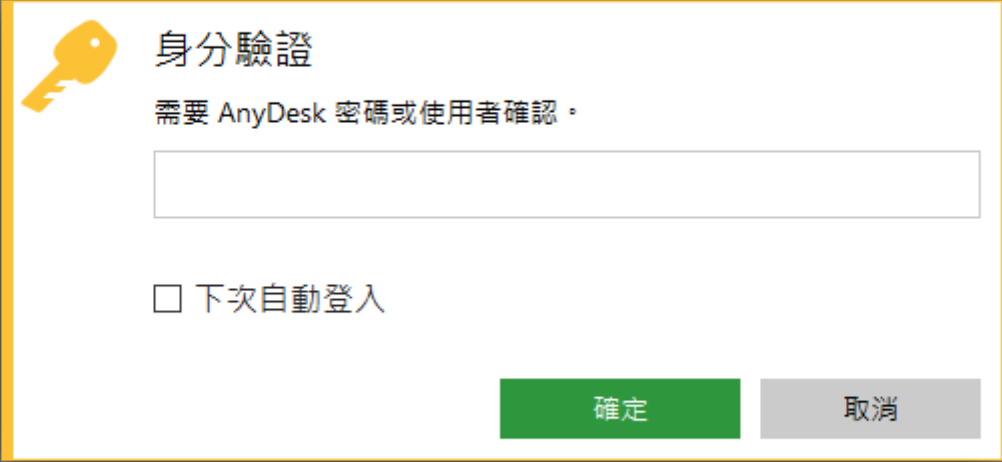
Server ID : 591 282 704



The screenshot shows the AnyDesk web interface. At the top, there is a header bar with the AnyDesk logo, a button labeled "新增連線" (Add Connection), and a search bar containing the ID "591282704". Below the header, the main content area is divided into two panels. The left panel, titled "此桌面" (This Desktop), shows a download icon and a link to "邀請郵件..." (Invite Email...). The right panel, titled "遠端桌面" (Remote Desktop), shows a text input field with the ID "597282704" and a green "連線" (Connect) button. At the bottom, there is a promotional banner for AnyDesk's Black Friday Offer with a "Learn More" button. The footer contains navigation links: "已探索" (Explored), "顯示全部" (Show All), and icons for grid, list, and search views.

Server Login(2/)

Password : boledu421



A screenshot of an AnyDesk login dialog box. The dialog has a yellow border and a yellow key icon on the left. The title is '身分驗證' (Identity Verification). Below the title is the text '需要 AnyDesk 密碼或使用者確認。' (Need AnyDesk password or user confirmation.). There is a text input field. Below the input field is a checkbox labeled '下次自動登入' (Auto-login next time). At the bottom right are two buttons: '確定' (OK) in green and '取消' (Cancel) in gray.

身分驗證

需要 AnyDesk 密碼或使用者確認。

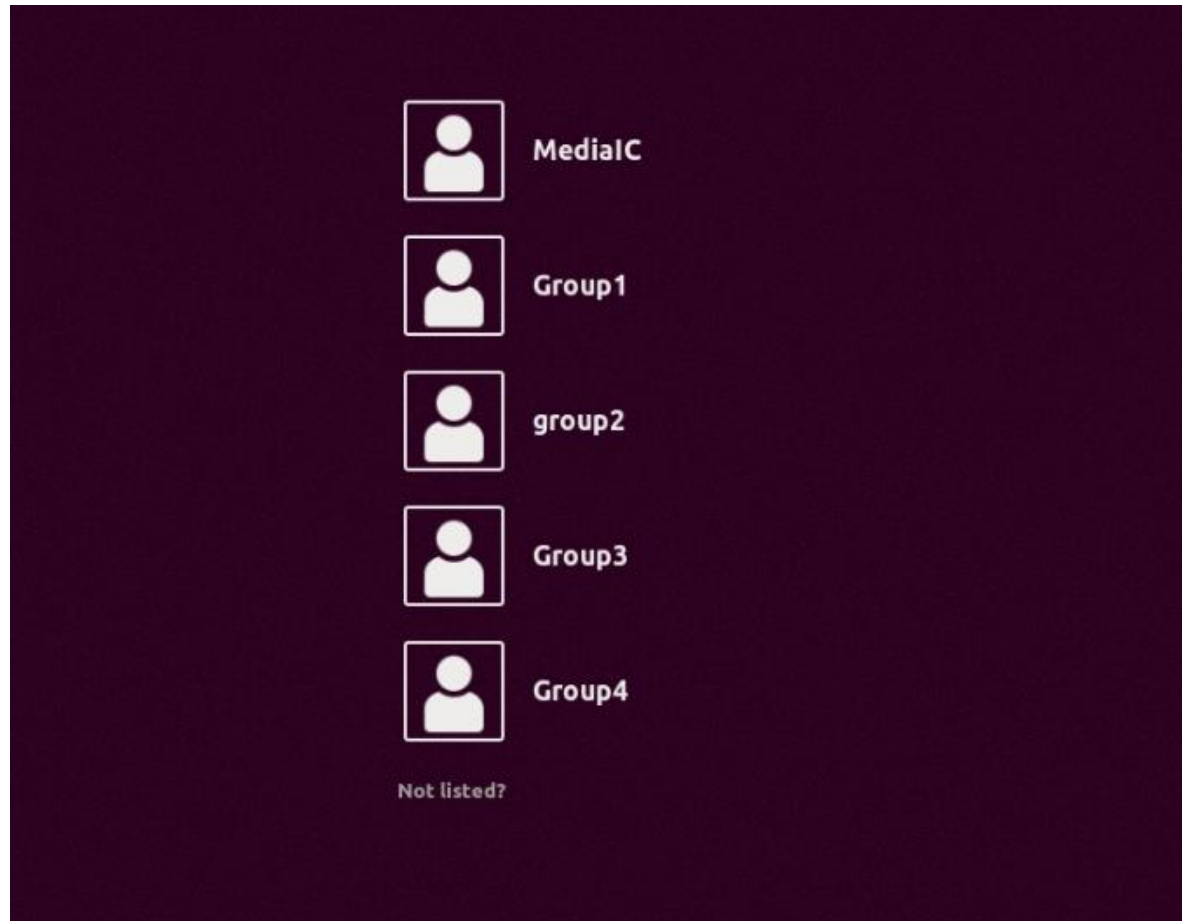
☐ 下次自動登入

確定 取消

Server Login(3/)

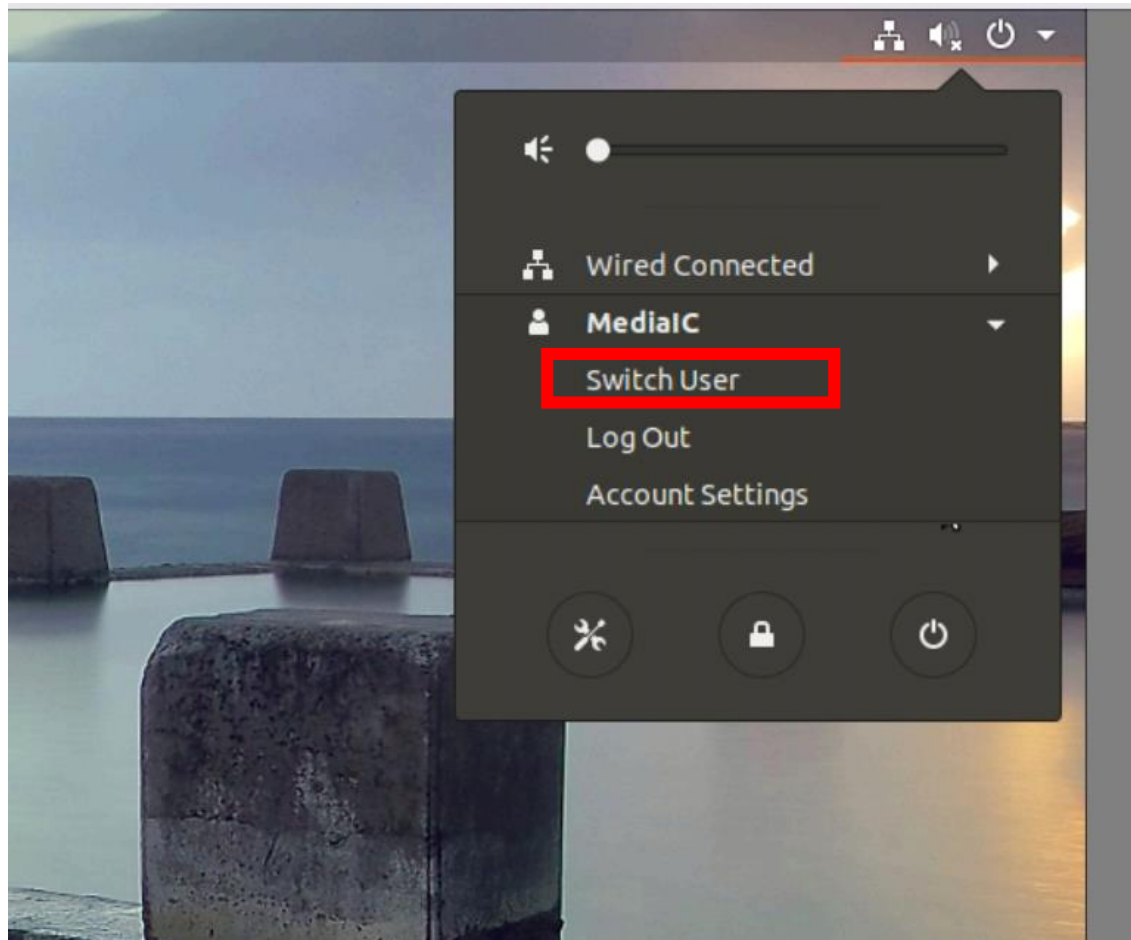
Choose your account

Password : boledu421



Switch User

After using this server, be sure to click the switch user button!!!



Server Login

- You can connect them through SSH:
 - ID: **591 282 704**
 - password: **boledu421**
- Due to the limited resources of the servers, you have to make an appointment to use them.
 - Students who use the server without making an appointment will get some penalties. This may cause the server out of memory and terminate the building process unexpectedly
 - Each group can only book 4 hours per day
 - <https://reurl.cc/ox3k73>

Bashrc File Configuration

- Only needs to be set for the first time log in.
- Create build directory in your home
 - `mkdir ~/build`
- Open bashrc file
 - `vi ~/.bashrc`
- Add the following codes at the end of bashrc file
 - `export FINN_HOST_BUILD_DIR=/home/groupX/build`
 - `export VIVADO_PATH=/tools/Xilinx/Vivado/2020.1`
 - `export BOARD=Pynq-Z2`
 - `PYNQ_IP=140.112.207.155`
 - `source /tools/Xilinx/Vivado/2020.1/settings64.sh`

Open Jupyter Notebook

- Command
 - `bash run-docker.sh notebook`

```

Successfully installed PyVirtator-0.0.1
oh-my-xilinx @ 1bae737669901e762f581af73348332b5c4b2ada
Obtaining file:///workspace/finn
Requirement already satisfied: finn-base@ git+https://github.com/Xilinx/finn-base#egg=finn-base from git+https://github.com/Xilinx/finn-base#egg=finn-base (0.0.post0.dev108+gac0b86a)
Requirement already satisfied: bitstring>=3.1.7 in /opt/conda/lib/python3.8/site-packages (from finn-base@ git+https://github.com/Xilinx/finn-base#egg=finn-base)
Requirement already satisfied: numpy in /opt/conda/lib/python3.8/site-packages (from finn-base@ git+https://github.com/Xilinx/finn-base#egg=finn-base)
Installing collected packages: finn
  Running setup.py develop for finn
Successfully installed finn
/tools/Xilinx/Vivado/2020.1/settings64.sh: line 5: /tools/Xilinx/DocNav/.settings64-DocNav.sh: No such file or directory
[I 16:48:47.899 NotebookApp] Writing notebook server cookie secret to /workspace/.local/share/jupyter/runtime/notebook_cookie_secret
[I 16:48:48.127 NotebookApp] Serving notebooks from local directory: /workspace/finn/notebooks
[I 16:48:48.127 NotebookApp] Jupyter Notebook 6.4.4 is running at:
[I 16:48:48.127 NotebookApp] http://finn_dev_mediaic:8888/?token=3b7a3720713d91bdb74b4b174b526b9dde0f918f308f83ca
[I 16:48:48.127 NotebookApp] or http://127.0.0.1:8888/?token=3b7a3720713d91bdb74b4b174b526b9dde0f918f308f83ca
[I 16:48:48.127 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 16:48:48.129 NotebookApp]

To access the notebook, open this file in a browser:
  file:///workspace/.local/share/jupyter/runtime/nbserver-7-open.html
Or copy and paste one of these URLs:
  http://finn_dev_mediaic:8888/?token=3b7a3720713d91bdb74b4b174b526b9dde0f918f308f83ca
  or http://127.0.0.1:8888/?token=3b7a3720713d91bdb74b4b174b526b9dde0f918f308f83ca
  
```

Open this link

PYNQ board first-time setup

PYNQ board first-time setup

We use *host* to refer to the PC running the FINN Docker environment, which will build the accelerator+driver and package it up, and *target* to refer to the PYNQ board. To be able to access the target from the host, you'll need to set up SSH public key authentication:

Start on the target side:

1. Note down the IP address of your PYNQ board. This IP address must be accessible from the host.
2. Ensure the `bitstring` package is installed: `sudo pip3 install bitstring`

Continue on the host side (replace the `<PYNQ_IP>` and `<PYNQ_USERNAME>` with the IP address and username of your board from the first step):

1. Launch the Docker container from where you cloned finn with `./run-docker.sh`
2. Go into the `ssh_keys` directory (e.g. `cd /workspace/finn/ssh_keys`)
3. Run `ssh-keygen` to create a key pair e.g. `id_rsa` private and `id_rsa.pub` public key
4. Run `ssh-copy-id -i id_rsa.pub <PYNQ_USERNAME>@<PYNQ_IP>` to install the keys on the remote system
5. Test that you can `ssh <PYNQ_USERNAME>@<PYNQ_IP>` without having to enter the password. Pass the `-v` flag to the ssh command if it doesn't work to help you debug.

https://finn.readthedocs.io/en/latest/getting_started.html

Change Dataset

- In part2, before we validate our model on PYNQ-Z2, we must specify our dataset to cifar100 first.

```
61     if dataset == "mnist":
62         from dataset_loading import mnist
63
64         trainx, trainy, testx, testy, valx, valy = mnist.load_mnist_data(
65             dataset_root, download=True, one_hot=False
66         )
67     elif dataset == "cifar10":
68         from dataset_loading import cifar
69
70         trainx, trainy, testx, testy, valx, valy = cifar.load_cifar_data(
71             dataset_root, download=True, one_hot=False, cifar10=False
72         )
73     else:
74         raise Exception("Unrecognized dataset")
```












File Location



Files Running Clusters

Select items to perform actions on them.

0 / end2end_example / bnn-pynq

- ..
- ☐  cnv_end2end_example.ipynb
- ☐  tfc_end2end_example.ipynb
- ☐  tfc_end2end_verification.ipynb
- ☐  cnv-mp-fc.png
- ☐  finn-design-flow-example.svg
- ☐  finn-hw-arch.png
- ☐  pynq_shell_project.png
- ☐  stitched_ip.png
- ☐  StreamingDataflowPartition_1.pdf
- ☐  top.pdf
- ☐  verification.png

Part1 (points to cnv_end2end_example.ipynb)

Part3 (points to tfc_end2end_example.ipynb)

Lab#D Material

- LabD/
 - Brevitas/
 - Vgg9/
 - finn.sh
 - Git clone FINN from github

TA Information

- Hua-Yang Weng
 - E-mail: huayang@media.ee.ntu.edu.tw
 - TA hours: Wed. 21:00 – 23:00
 - Location: <https://meet.google.com/ifb-twqn-xci>
- Ke-Han Li
 - E-mail: khlee@media.ee.ntu.edu.tw
 - TA hours: Wed. 21:00 – 23:00
 - Location: BL421