Using Python Overlays to Experiment with Neural Networks

SAN19-313

Tom Curran, Avnet

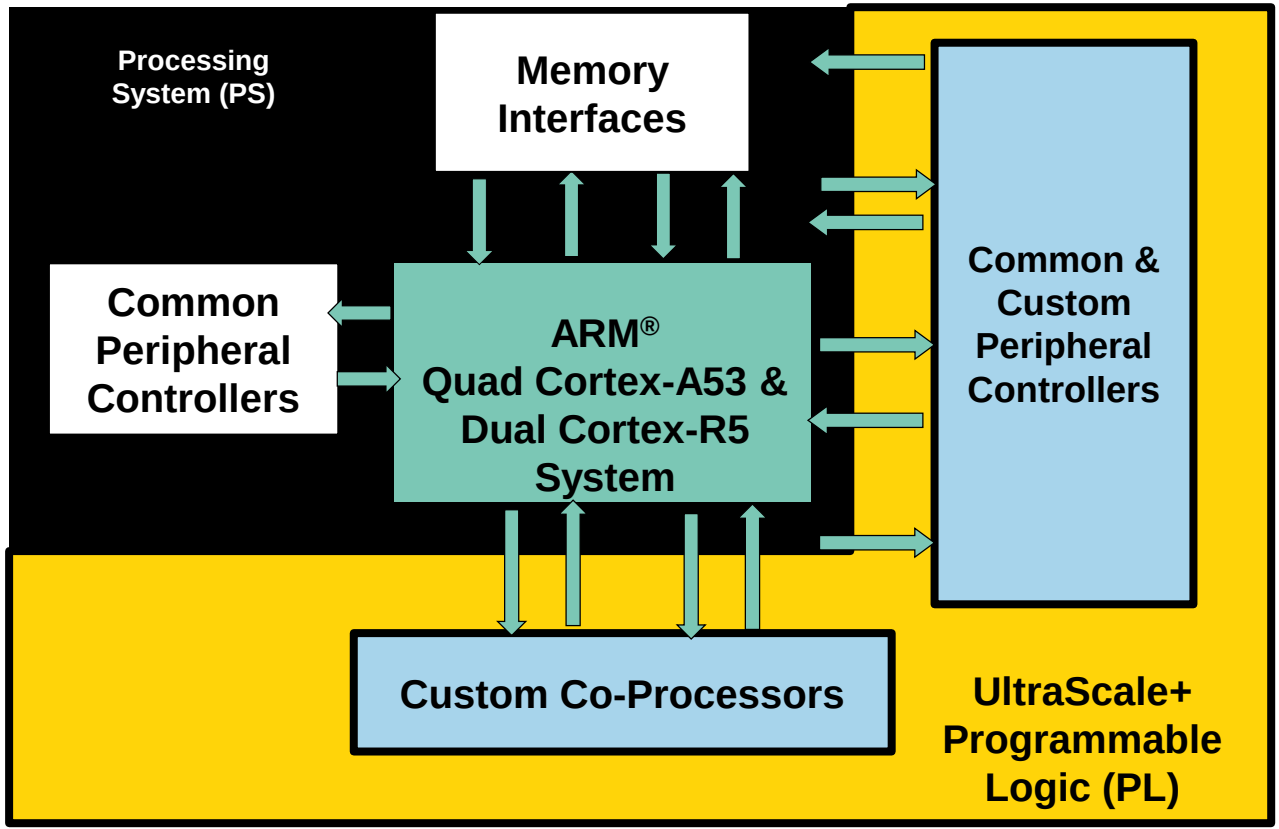
Linaro connect
San Diego 2019

Self Introduction

- Avnet – 12 years
- Global Products & Emerging Technologies
  - Technical training courses
  - Reference designs
  - PetaLinux BSPs
  - Dev board development
  - Embedded Linux & software development for Zynq, Zynq MPSoC, and MicroBlaze

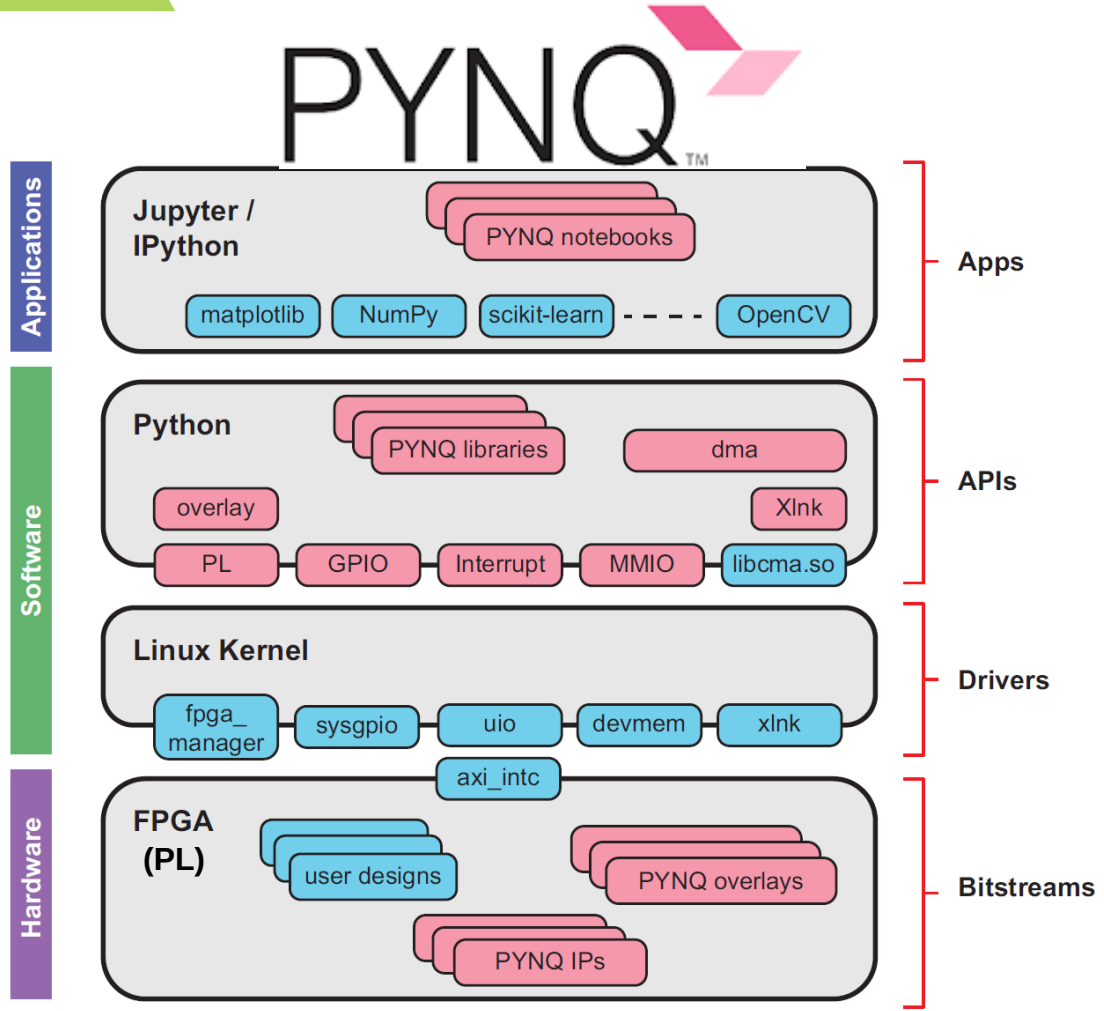
Accelerating a Processor

Zynq MPSoC System-on-Chip PS & PL



What is PYNQ?

- Open-source project from Xilinx
- Makes it easy to design embedded systems with Zynq
- Python language and libraries
  - Exploits both PS & PL
- Create high performance embedded applications
  - Parallel hardware execution
  - High frame-rate video processing
  - Hardware accelerated algorithms
  - Real-time signal processing
  - High bandwidth IO
  - Low latency control



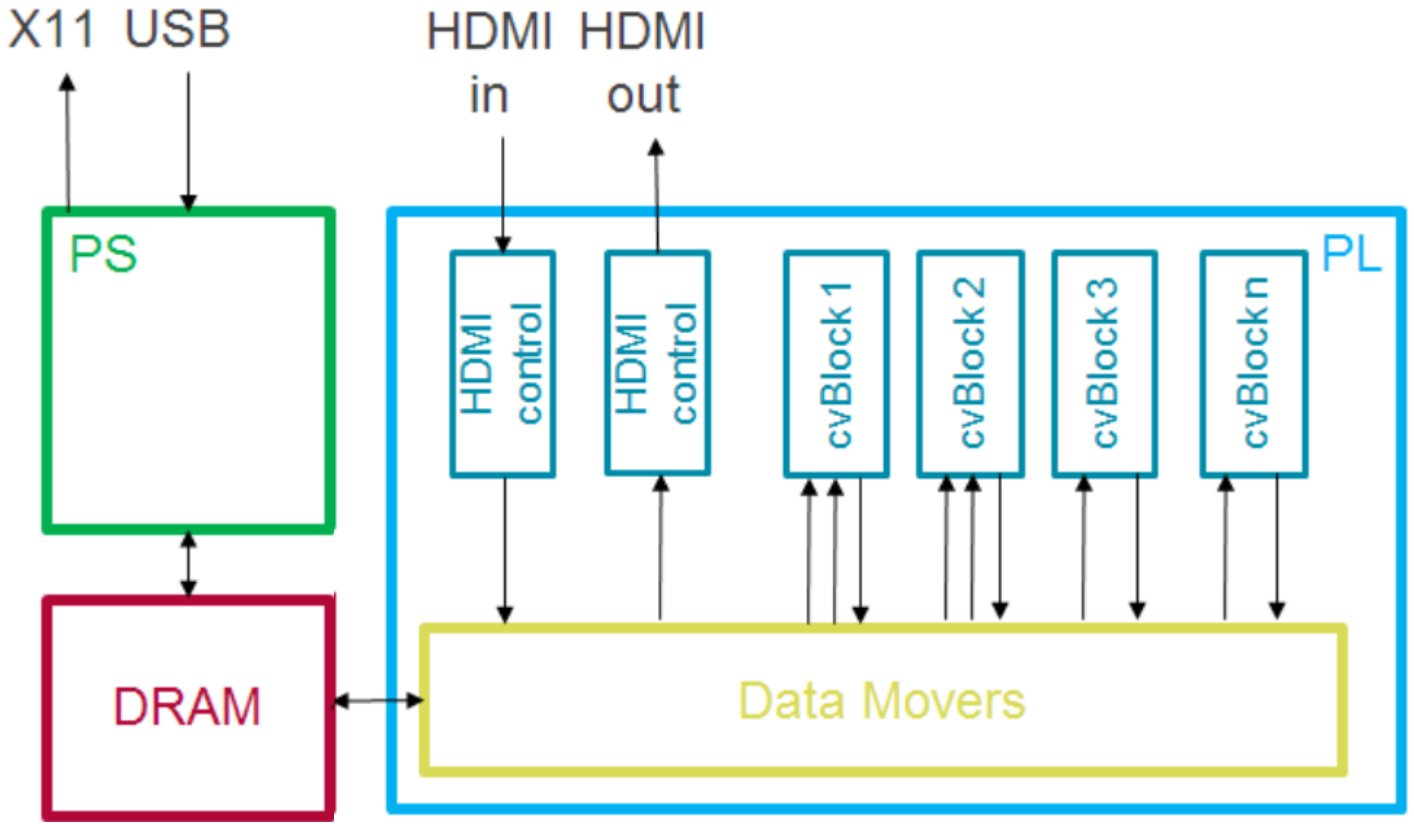
http://avnet.me/pynq_info

What is an Overlay?

- Python wrapper around an underlying PL Hardware Design
- Access hardware co-processors and peripherals as function calls
- Analogous to highly-accelerated and customized software libraries
- Image processing example using PL for hardware acceleration
  - SW programmer uses a library to run image processing functions on the co-processors
    - edge detect, thresholding, etc
  - Hardware co-processors are loaded to the PL dynamically, as required, just like a software library
  - Using Pynq, separate image processing functions could be implemented in different overlays and loaded from Python on demand.

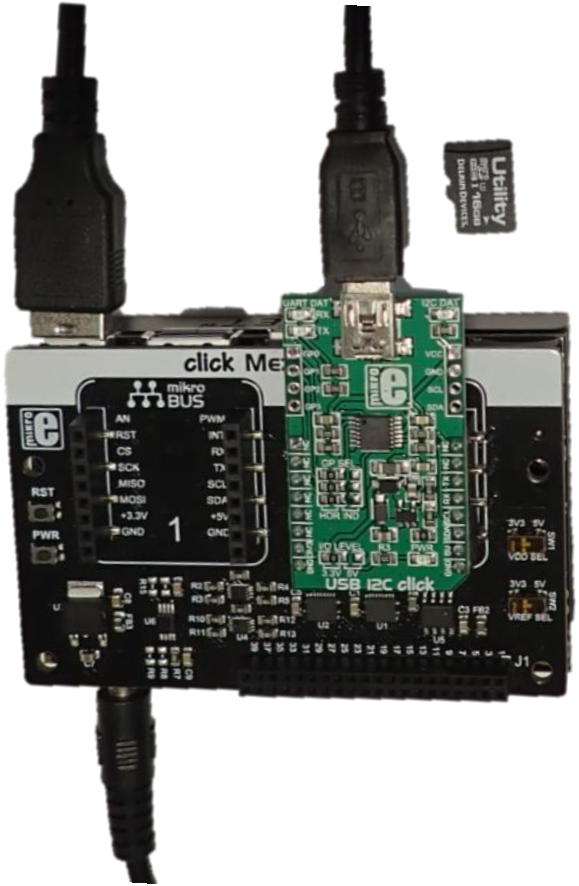
https://github.com/Xilinx/PYNQ-ComputerVision

Image Processing Example



Set up for PYNQ

- Download PYNQ image and write to microSD card
  - Ultra96: http://avnet.me/ultra96-pynq
- Connect all the hardware
  - Ultra96-V1 or Ultra96-V2
  - 4A Power Supply
  - Internet
    - Wired USB-to-Ethernet
    - WiFi
    - USB Gadget Ethernet also available
  - Terminal through Click Mezzanine USB-UART in mikroBUS #2
    - Not required



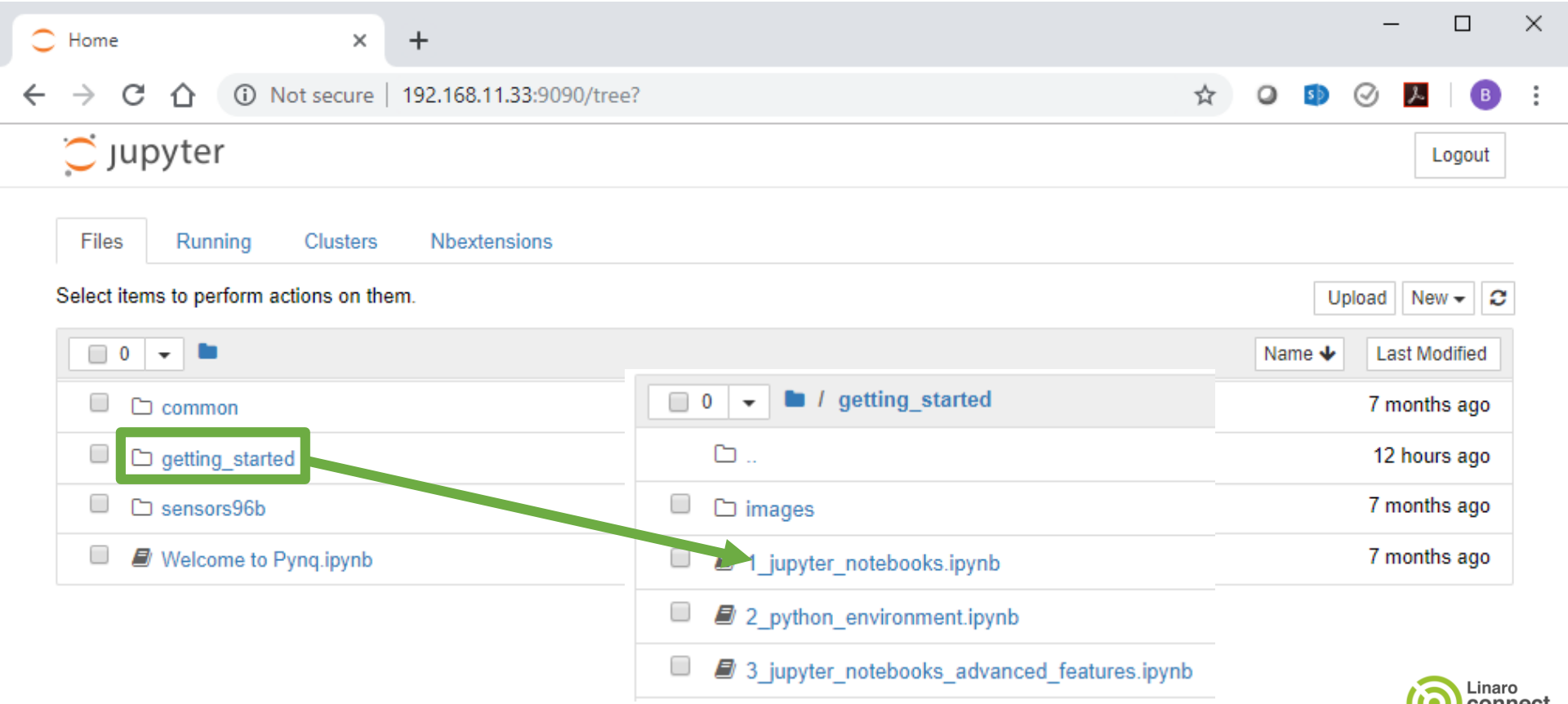
Getting Started with PYNQ

http://avnet.me/ultra96-pynq-setup

- Boot the board

- Open browser to IP Address of the board (turn off any VPN)
  - Wi-Fi: 192.168.2.1
  - USB Gadget: 192.168.3.1
  - USB-to-Ethernet: Depends on what router assigned.
    - If Terminal connected, use `ifconfig` command to determine IP address

Experiment with Getting Started

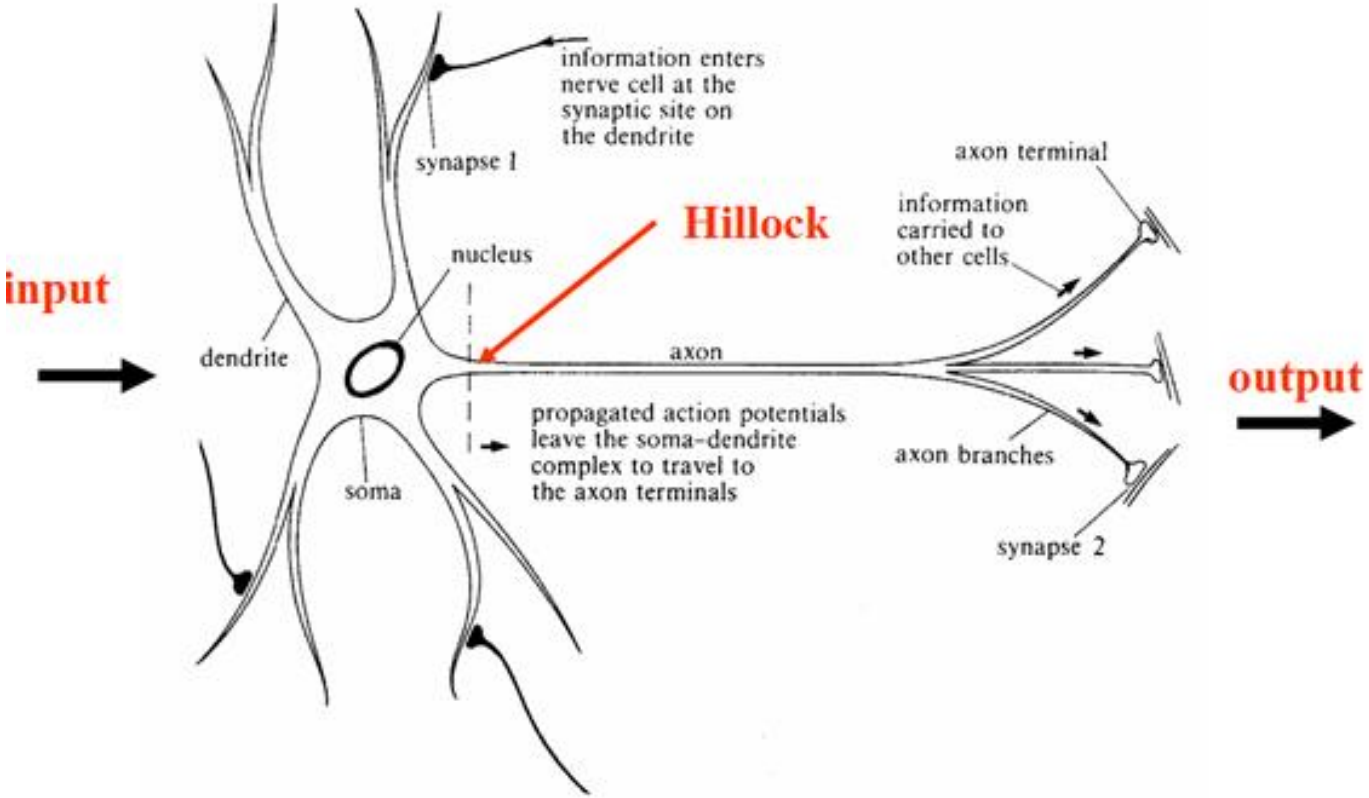


Available Overlays

- Most are on the Xilinx GitHub
  https://github.com/Xilinx
  - Sort by Language: Jupyter Notebook
- Support for Ultra96:
  - Quantized Neural Networks
    https://github.com/Xilinx/QNN-MO-PYNQ
  - Binarized Neural Networks
    https://github.com/Xilinx/BNN-PYNQ
  - OpenCV vision pipelines
    https://github.com/Xilinx/PYNQ-ComputerVision
  - Extended Kalman Filter on GPS data
    https://github.com/sfox14/pynq-ekf
  - PYNQ Helloworld image resizer
    https://github.com/Xilinx/PYNQ-HelloWorld

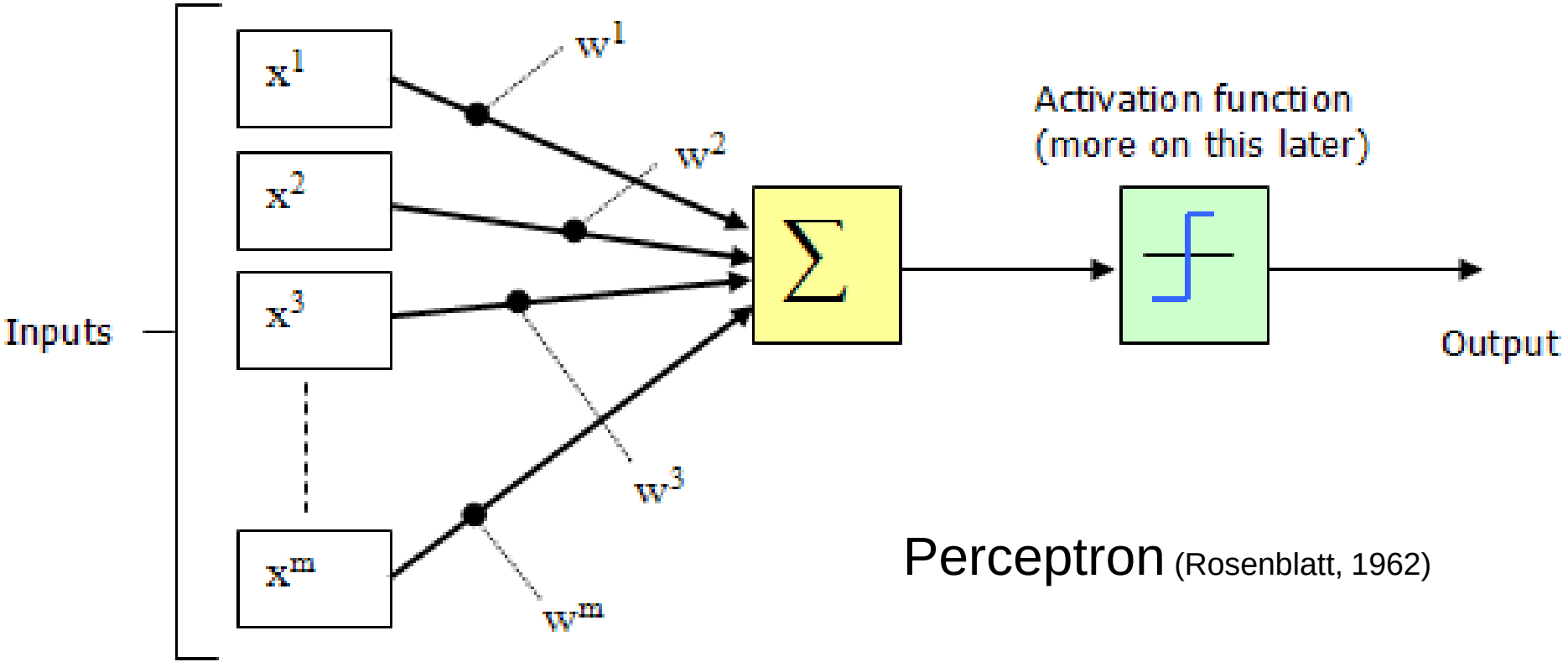
What Are Neural Networks?

- Modeled after nerve cells in the brain, called neurons
- Neuron consists of:
  - **Inputs and weights** (synapses and dendrites)
  - **Summation and Activation** (cell body)
  - **Output** (hillock -- firing across a membrane through the axon)



*Credit: Sacha Barber, *Neural Network for beginners,* https://www.codeproject.com/Articles/16419/AI-Neural-Network-for-beginners-Part-of

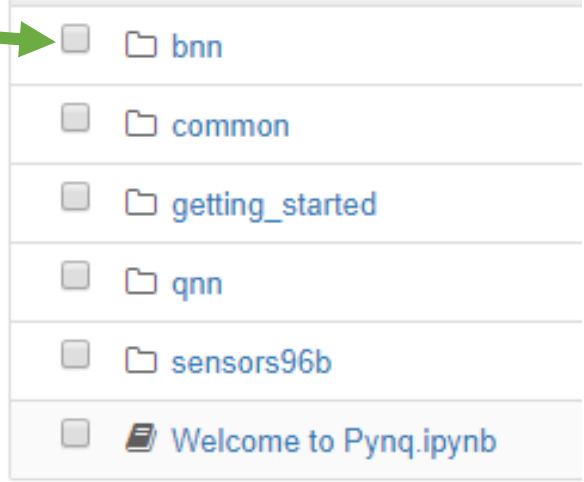
What Are Artificial Neural Networks?



Perceptron (Rosenblatt, 1962)

Install Neural Networks Notebook Into PYNQ

- Binarized Neural Network (BNN)
  - **sudo pip3 install git+https://github.com/Xilinx/BNN-PYNQ.git**

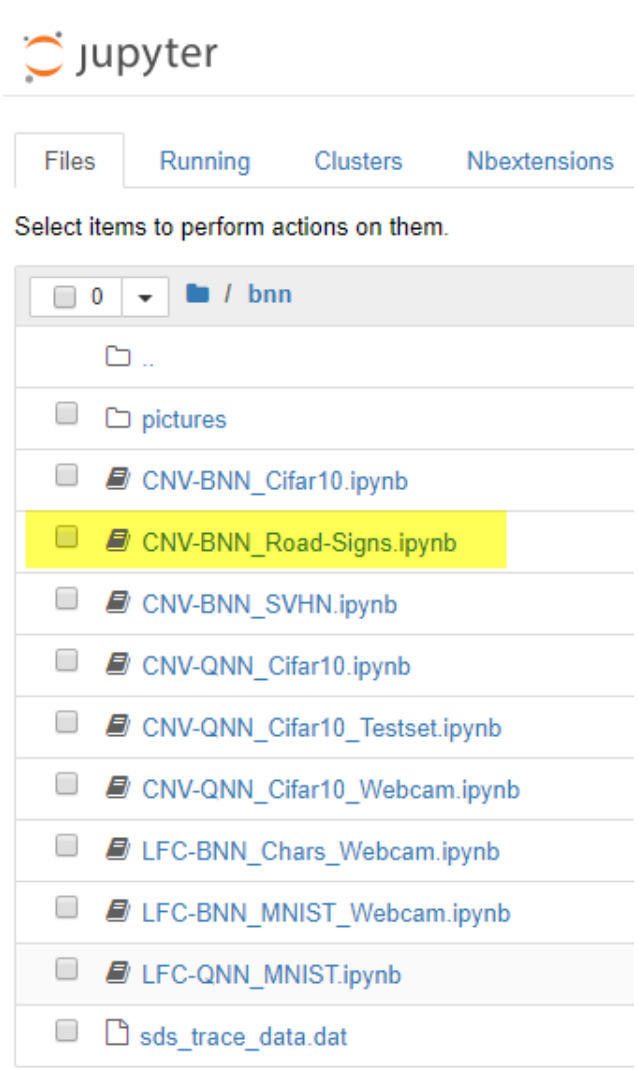


BNN Road Signs Example

- Binary Neural Network on Pynq
- Image recognition with a binarized neural network inspired by the VGG-16 model
  - 6 convolutional layers
  - 3 max pool layers
  - 3 fully connected layers

https://neurohive.io/en/popular-networks/vgg16/

Recorded Example (Sahaj Sarup):
https://youtu.be/ptzrg9dPI3w



Instantiate a Classifier

1. Instantiate a Classifier

Creating a classifier will automatically download the correct bitstream onto the device and load the weights trained on the specified dataset. By default there are three sets of weights available for the BNN version of the CNV network using 1 bit weights and 1 activation (W1A1) - this example uses the German Road Sign dataset.

Explanation

```
import bnn
print(bnn.available_params(bnn.NETWORK_CNVW1A1))

classifier = bnn.CnvClassifier(bnn.NETWORK_CNVW1A1, 'road-signs', bnn.RUNTIME_HW)
```

```
['cifar10', 'road-signs', 'streetview']
```

Operation

List the Available Classes

2. List the available classes

The selected dataset can classify images in 42 classes, the names of which are accessible through the classifier.

```
print(classifier.classes)
```

```
['20 Km/h', '30 Km/h', '50 Km/h', '60 Km/h', '70 Km/h', '80 Km/h', 'End 80 Km/h', '100 Km/h', '120 Km/h', 'No overtaking', 'No
overtaking for large trucks', 'Priority crossroad', 'Priority road', 'Give way', 'Stop', 'No vehicles', 'Prohibited for vehicle
s with a permitted gross weight over 3.5t including their trailers, and for tractors except passenger cars and buses', 'No entr
y for vehicular traffic', 'Danger Ahead', 'Bend to left', 'Bend to right', 'Double bend (first to left)', 'Uneven road', 'Road
slippery when wet or dirty', 'Road narrows (right)', 'Road works', 'Traffic signals', 'Pedestrians in road ahead', 'Children cr
ossing ahead', 'Bicycles prohibited', 'Risk of snow or ice', 'Wild animals', 'End of all speed and overtaking restrictions', 'T
urn right ahead', 'Turn left ahead', 'Ahead only', 'Ahead or right only', 'Ahead or left only', 'Pass by on right', 'Pass by on
left', 'Roundabout', 'End of no-overtaking zone', 'End of no-overtaking zone for vehicles with a permitted gross weight over 3.
5t including their trailers, and for tractors except passenger cars and buses', 'Not a roadsign']
```

Open Images To Be Classified

3. Open images to be classified

The images that we want to classify are loaded and shown to the user

```
from PIL import Image
import numpy as np
from os import listdir
from os.path import isfile, join
from IPython.display import display

imgList = [f for f in listdir("/home/xilinx/jupyter_notebooks/bnn/pictures/road_signs/") if isfile(join("/home/xilinx/jupyter_not

images = []

for imgFile in imgList:
    img = Image.open("/home/xilinx/jupyter_notebooks/bnn/pictures/road_signs/" + imgFile)
    images.append(img)
    img.thumbnail((64, 64), Image.ANTIALIAS)
    display(img)
```


Launch BNN on Co-processor (PL)

4. Launching BNN in hardware

The images are passed in the PL and the inference is performed. The images will be automatically formatted to the required format that is processed by CNV network (Cifar-10 format).

```
results = classifier.classify_images(images)
print("Identified classes: {0}".format(results))
for index in results:
    print("Identified class name: {0}".format((classifier.class_name(index))))
```

```
Inference took 915.00 microseconds, 305.00 usec per image
Classification rate: 3278.69 images per second
Identified classes: [14 27 41]
Identified class name: Stop
Identified class name: Pedestrians in road ahead
Identified class name: End of no-overtaking zone
```

Launch Same Thing in Software (PS)

5. Launching BNN in software

The inference on the same image is performed in sofware on the ARM core by passing the RUNTIME_SW attribute to the Image-Classifier

```
sw_class = bnn.CnvClassifier(bnn.NETWORK_CNVW1A1,"road-signs", bnn.RUNTIME_SW)

results = sw_class.classify_images(images)
print("Identified classes: {0}".format(results))
for index in results:
    print("Identified class name: {0}".format((classifier.class_name(index))))
```

```
Inference took 1691285.06 microseconds, 563761.69 usec per image
Classification rate: 1.77 images per second
Identified classes: [14 27 41]
Identified class name: Stop
Identified class name: Pedestrians in road ahead
Identified class name: End of no-overtaking zone
```

- Processed at rate of 1.77 images per second
- Co-processor is 1850x faster


VET
P
STOP
C415


VET
STOP
C415


VET
STOP
C415

How Can You Get Started?

- PYNQ Experience not required
- Xilinx design experience not required
- You need hardware and an internet connection
- Learn more about PYNQ at http://avnet.me/pynq_info
- Get the Ultra96 image at http://avnet.me/ultra96-pynq
  - Also contains information about overlays
- Watch the Getting Started Video on YouTube http://avnet.me/ultra96-pynq-setup
- Watch the BNN Example Video on YouTube https://youtu.be/ptzrg9dPI3w
- Run a Neural Network PYNQ example yourself

Deeper Dive Learning

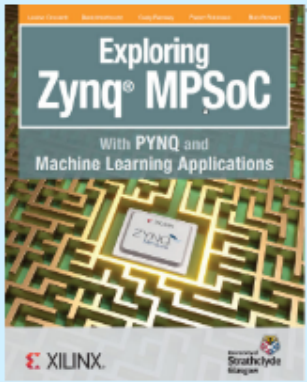
- Avnet Technical Training Courses On Demand for Ultra96
  - **http://avnet.me/TTC_on_Demand**
  - **Software Development (Bare Metal), Hardware Development, PetaLinux, SDSoC, Artificial Intelligence, and PYNQ (2-day course)**

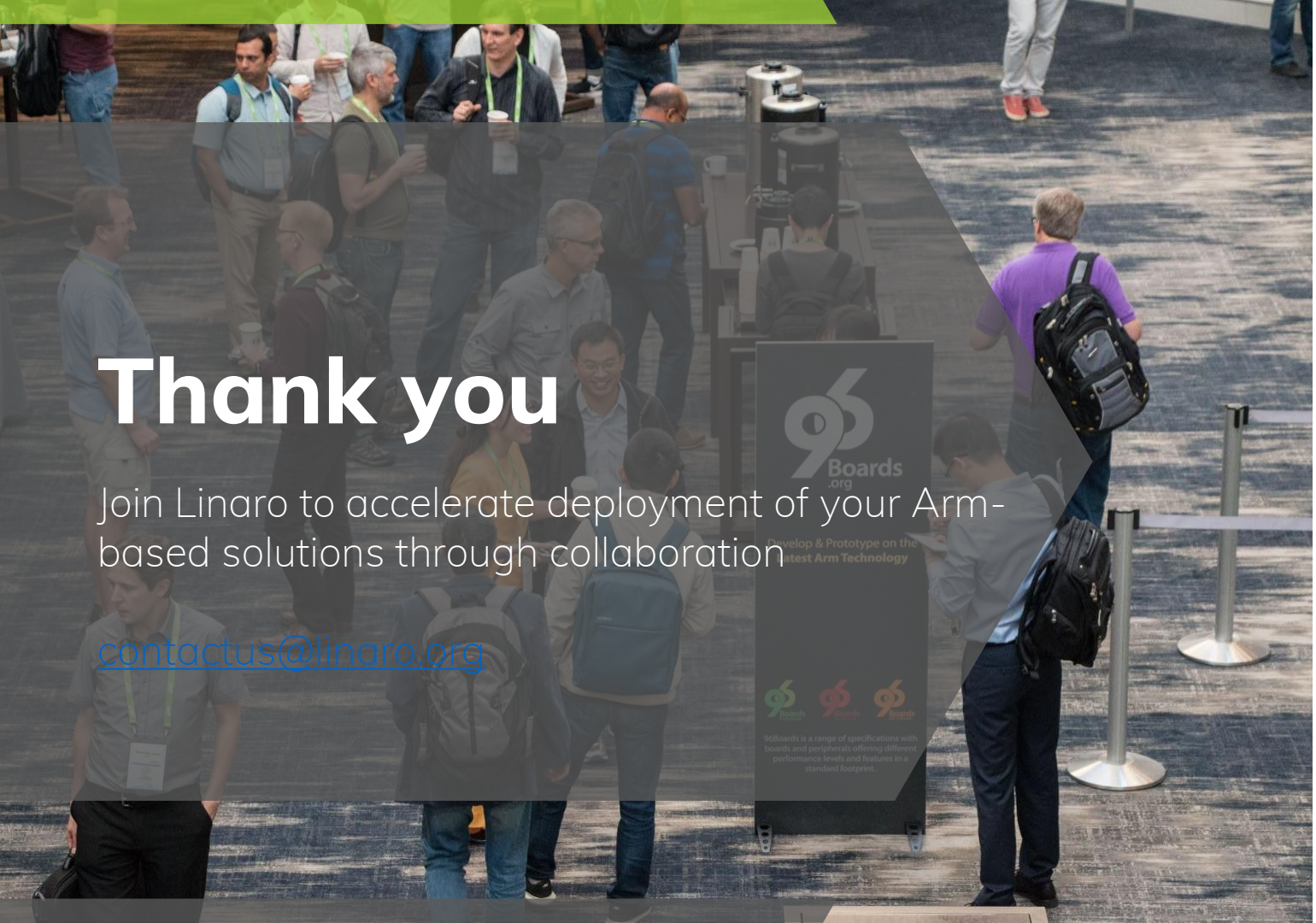
Free eBook: Exploring Zynq MPSoC with PYNQ and Machine learning applications

The book introduces **Zynq MPSoC** (Multi-Processor System-on-Chip) from Xilinx that combines an **Arm Cortex-A53** based processing systems, **Arm Cortex-R5** real-time processors, and **FPGA** programmable logic.

Topics include Zynq MPSoC **architecture**, **design tools**, hardware/software **co-design**, and **software-defined** methodologies. It features sepcial sections on **PYNQ** (Python-based framework for Zynq) and **machine learning**.

Download **free** from www.zynq-mpsoc-book.com or get a hard-copy on Amazon and other booksellers.





Thank you

Join Linaro to accelerate deployment of your Arm-based solutions through collaboration

contactus@linaro.org

Linaro connect
San Diego 2019