



# Python Productivity for Zynq

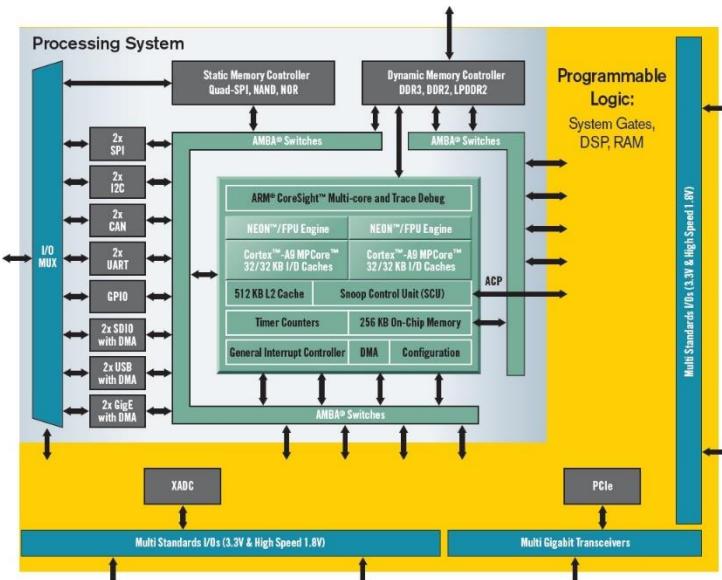


- > Zynq & Zynq Ultrascale+
- > PYNQ Framework
- > Technologies
- > Community

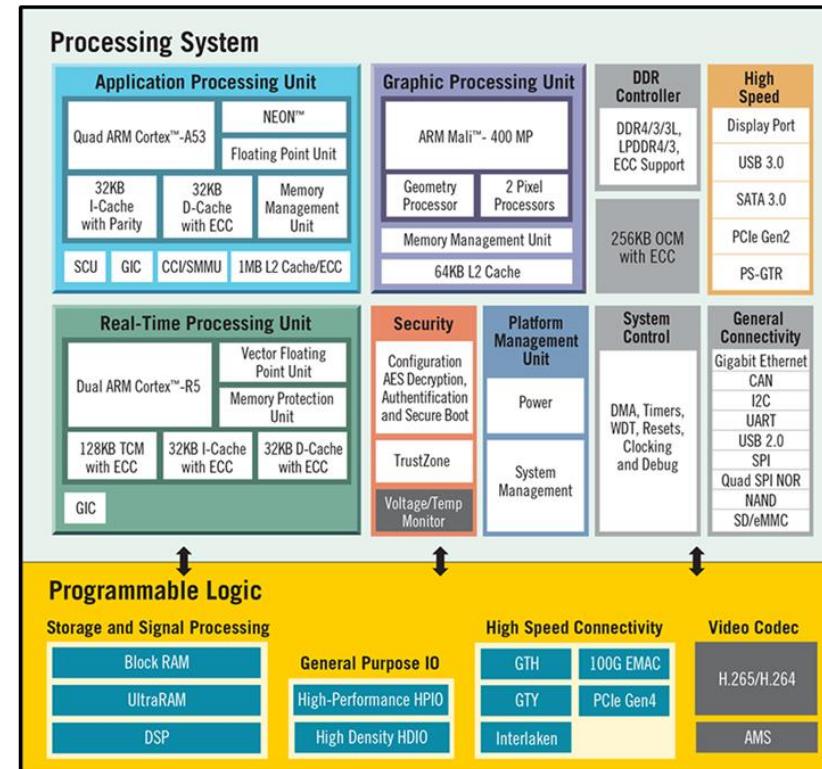
# ZYNQ and ZYNQ UltraSCALE+

Best-in-class, SoC and MPSoCs

ZYNQ 7000



ZYNQ UltraSCALE+



FPGAs and tightly-integrated CPUs enable entirely new opportunities

# Zynq applications

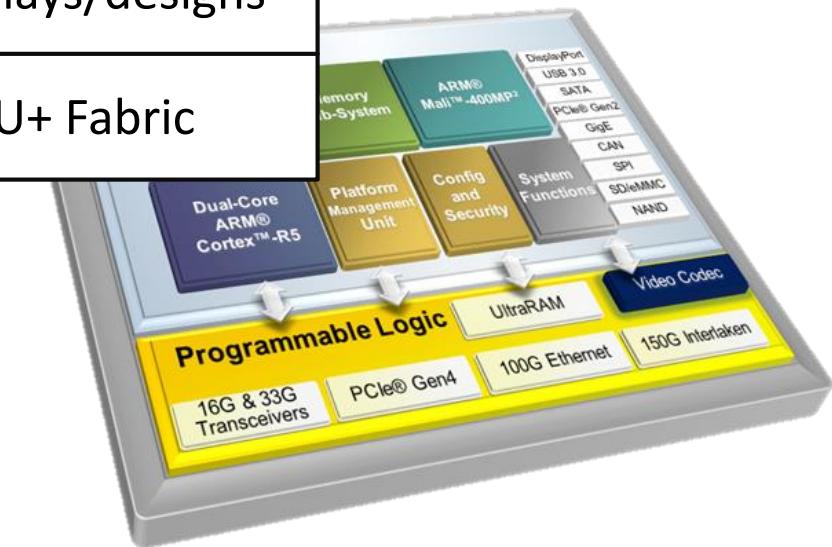
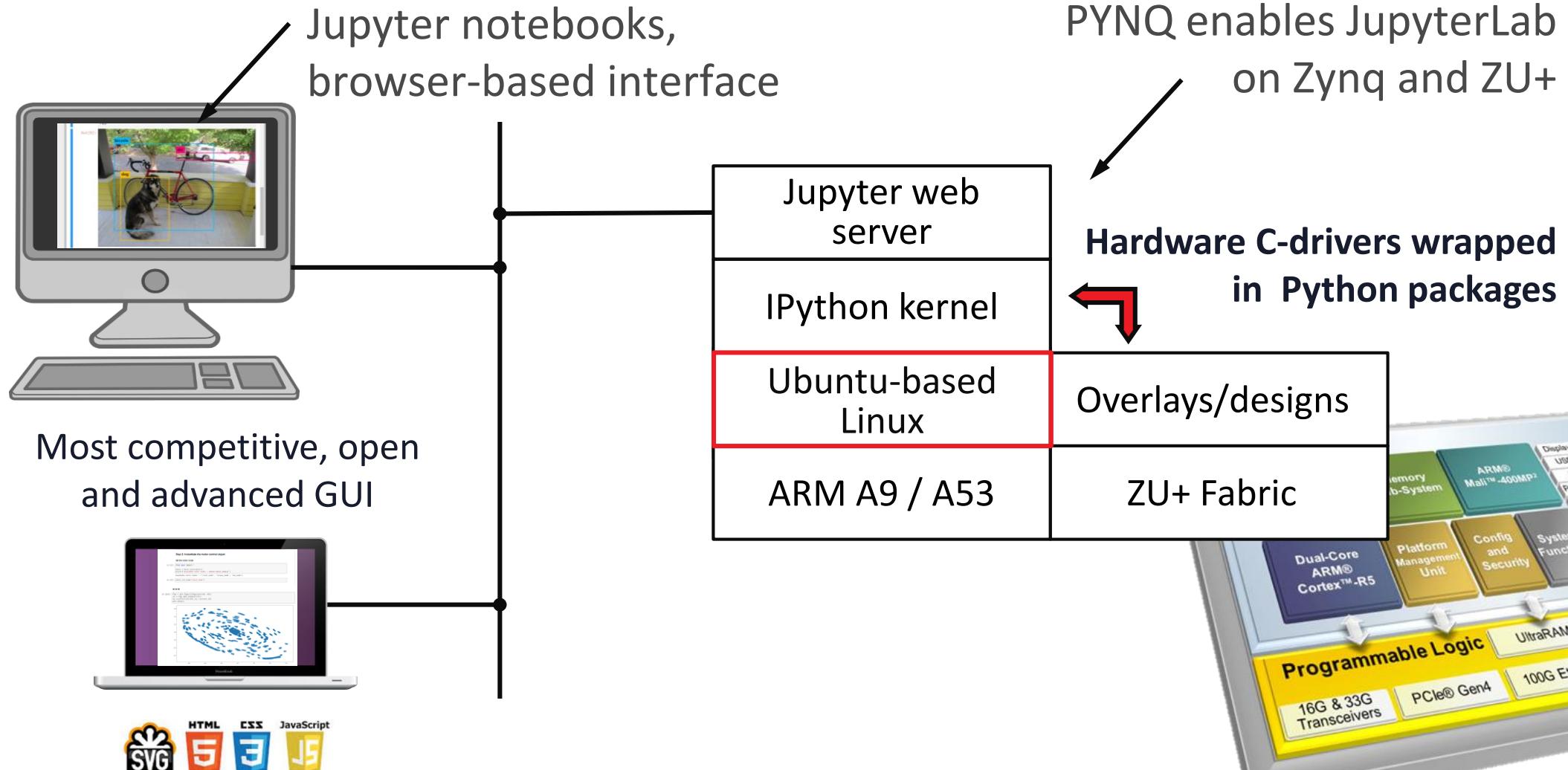


> Make Zynq so easy-to-use that programmers can access the benefits without learning advanced digital design skills



## PYNQ framework





# Ubuntu-based Linux versus embedded Linux

Ubuntu-based Linux

➤ **Optimized for developer productivity**



- > All the Linux libraries and drivers you expect
  - > Pre-built SD card image
  - > Ubuntu/Debian ecosystem & community
- >>145,000,000 Google hits

3 orders of magnitude difference

Embedded Linux



➤ **Optimized for deployment efficiency**

- > Selective Linux libraries and drivers
  - > Commonly delivered in flash memory on board
  - > PetaLinux ecosystem:
- >> 143,000 Google hits

# PYNQ's Ubuntu-based Linux

PYNQ uses Ubuntu's:

- Root file system (RFS)
- Package manager (*apt-get*)
- Repositories

PYNQ bundles :

- Development tools
  - Cross-compilers
- Latest Python packages



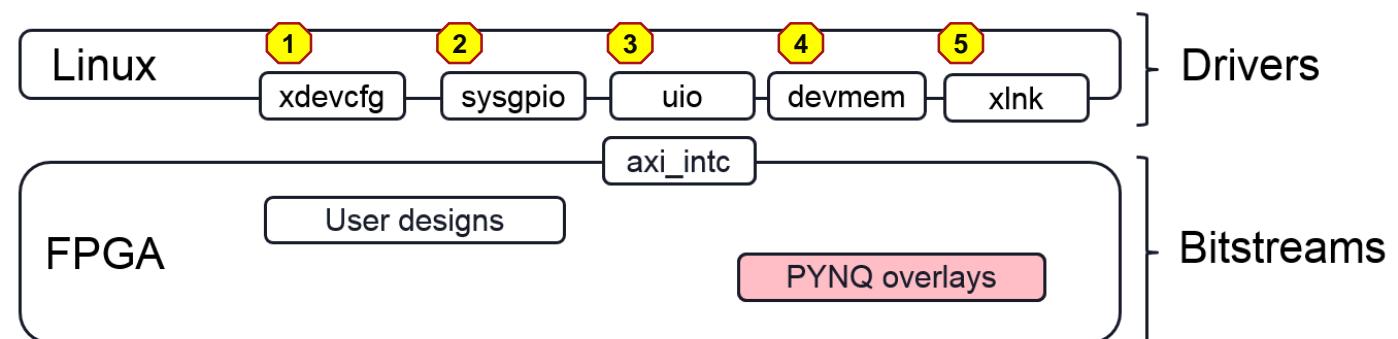
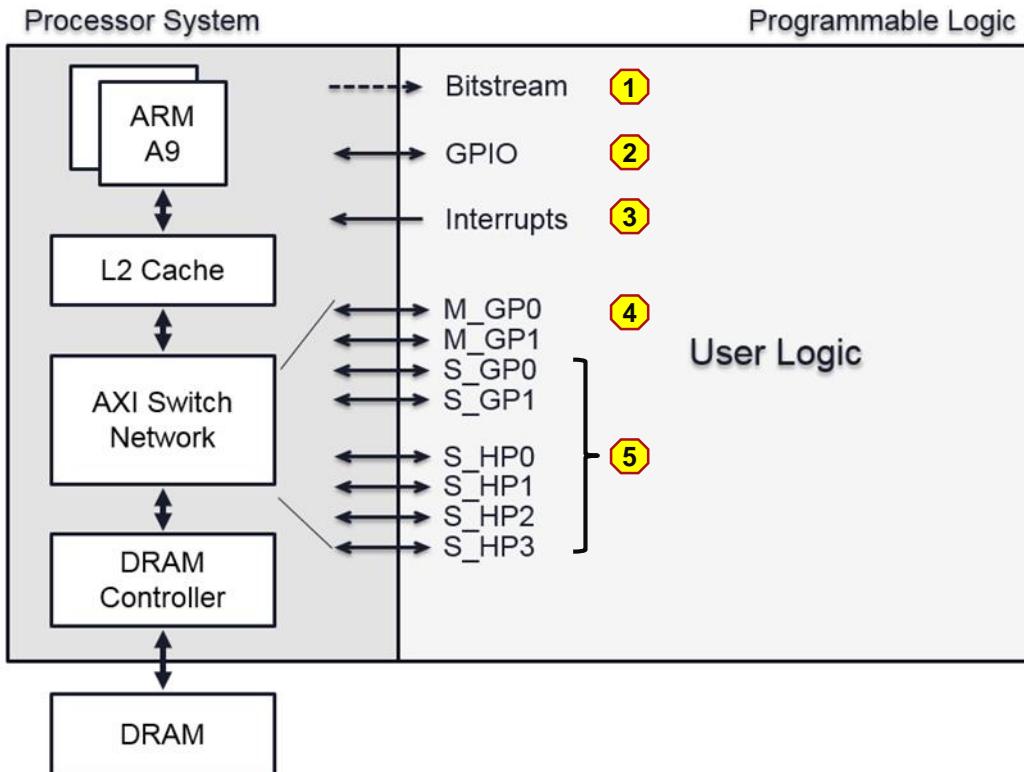
PYNQ uses the PetaLinux build flow and board support package:

- Access to all Xilinx kernel patches
- Works with any Xilinx supported board
- Configured with additional drivers for PS-PL interfaces

# PYNQ provides Linux Drivers for PS-PL Interfaces ...

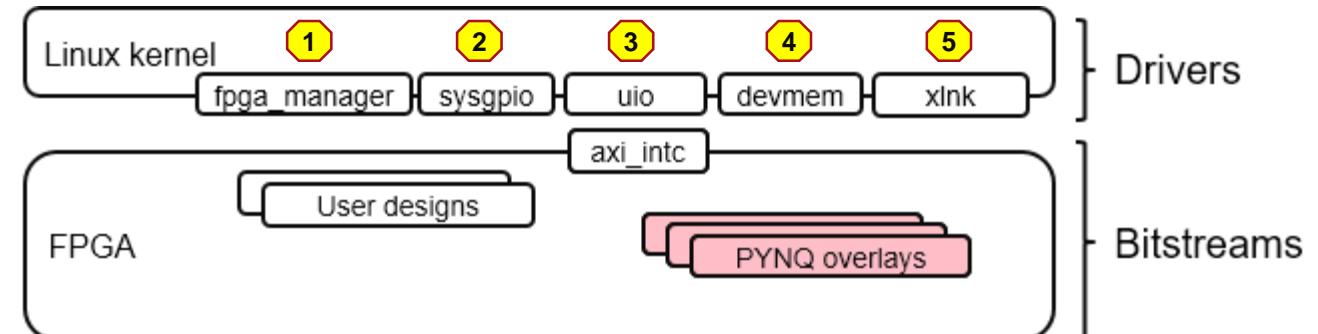
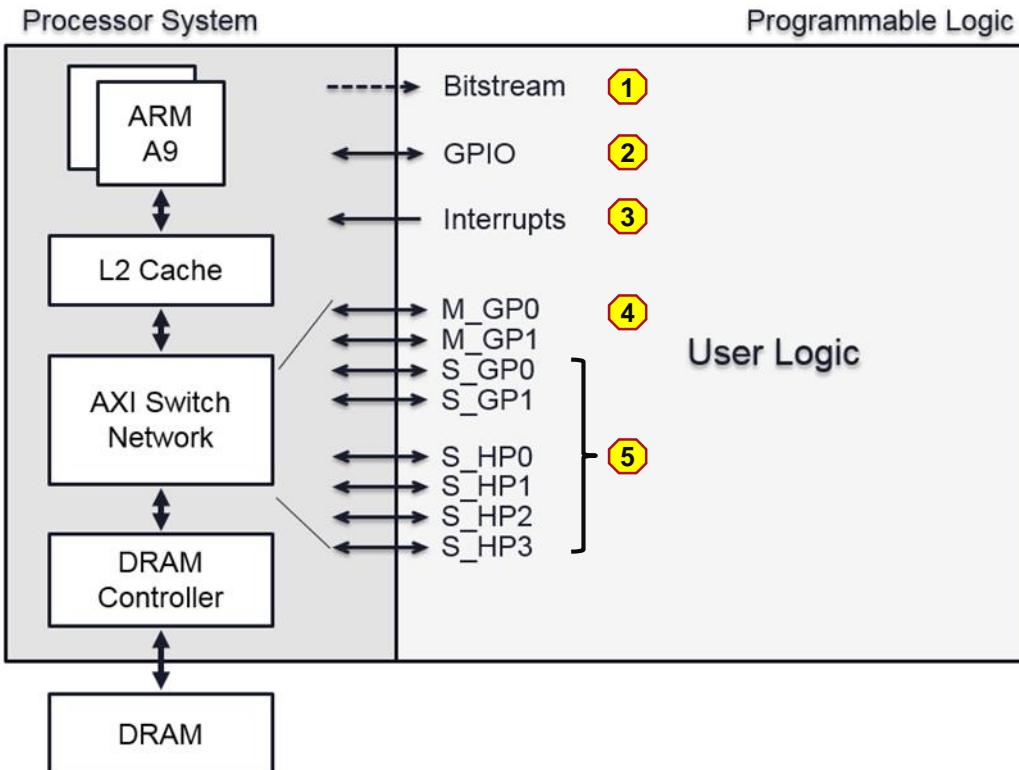
wrapped in Python Libraries

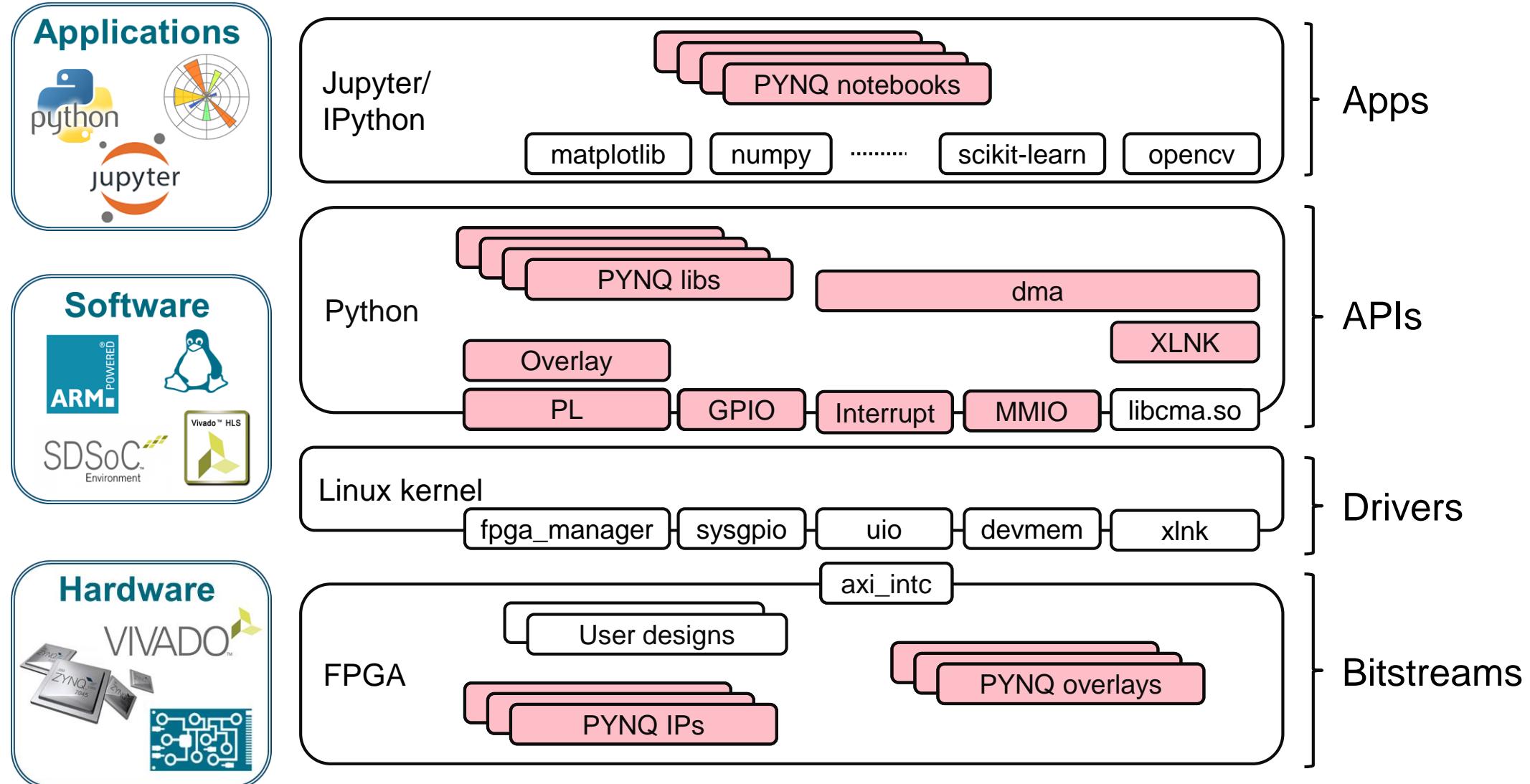
Zynq



# PYNQ provides Linux Drivers for PS-PL Interfaces ... wrapped in Python Libraries

Zynq





# Vivado Design Metadata available from Python

- > **PYNQ passes the Vivado metadata file to the target platform**
  - >> Initially Vivado TCL file
  - >> Now moving to Hardware Handoff (HHW) file
- > **It then parses the Vivado metadata file to:**
  - >> Set Zynq clock frequencies automatically
  - >> Assign memory-map attributes for every IP
  - >> Assign default MMIO drivers to IP, when no drivers are specified
- > **Creates a Python dictionary for the IP in the bitstream from the metadata file**
  - >> Enables bitstream metadata to be queried and modified in Python at runtime

# Hybrid Packages

- > New *hybrid packages* are created by extending Python packages with additional files:
  - » Design Bitstream
  - » Design metadata file
  - » C drivers
  - » Jupyter notebooks
- > Hybrid packages enable software-style packaging and distribution of designs
- > Use the Python package installer, PIP to install a hybrid package just like any regular Python (software only) package
  - » Delivers package's files to target board
- > Uses Python standard setup.py script for installation

# Software-style Packaging & Distribution of Designs

## Enabled by new *hybrid packages*

The figure displays four GitHub repository pages for Xilinx projects:

- Xilinx / QNN-MO-PYNQ**: A notebook titled "dorefanet-imagenet-samples.ipynb" showing code for image classification and a Beagleboard photo.
- Xilinx / IoT-SPYN**: A notebook titled "spyn.ipynb" demonstrating control of a 3-phase AC motor.
- Xilinx / PYNQ-DL**: A notebook titled "resize.ipynb" illustrating image resizing.
- Xilinx / PYNQ-ComputerVision**: A notebook titled "filter2d\_and\_dilate.ipynb" showing OpenCV overlay functionality.

Each screenshot shows the GitHub interface with code snippets, output cells, and explanatory text or images.

Download a design from GitHub with a single Python command:

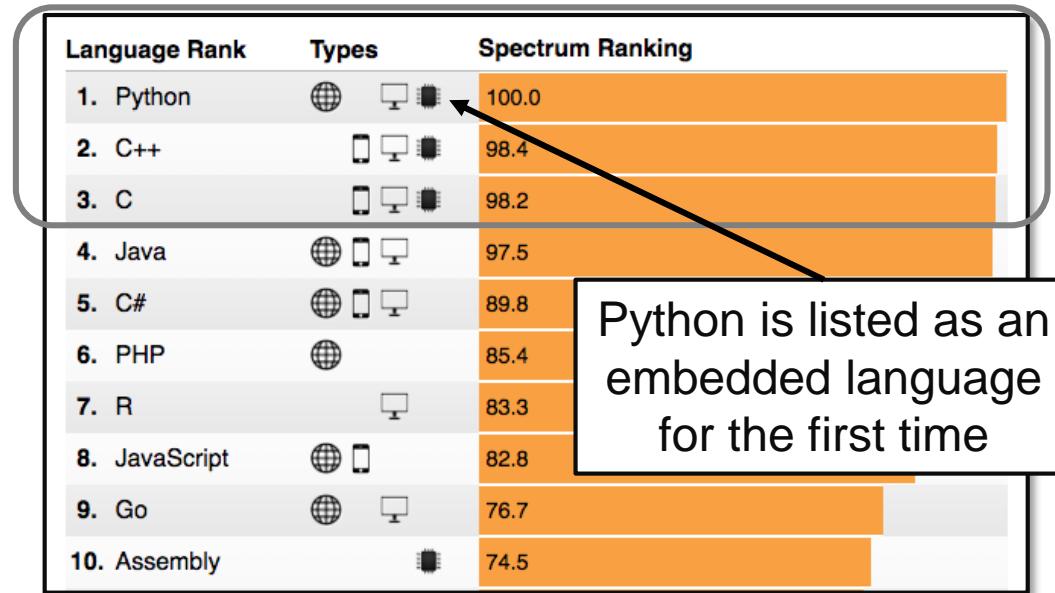
```
pip3.6 install git+https://github.com/Xilinx/pynqDL.git
```

# PYNQ enabling technologies

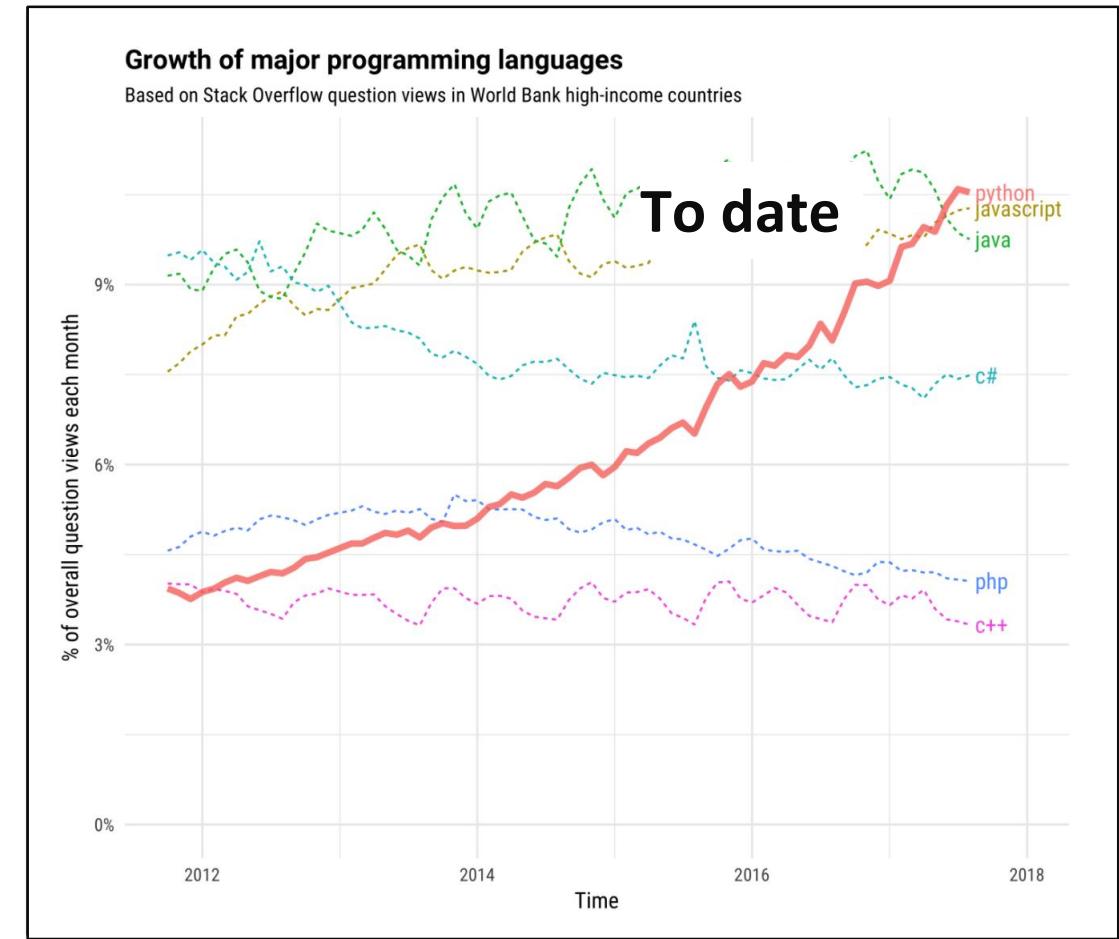


# Python is increasingly the Language of Choice

Top Programming Languages,  
IEEE Spectrum, July'18



<https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages>

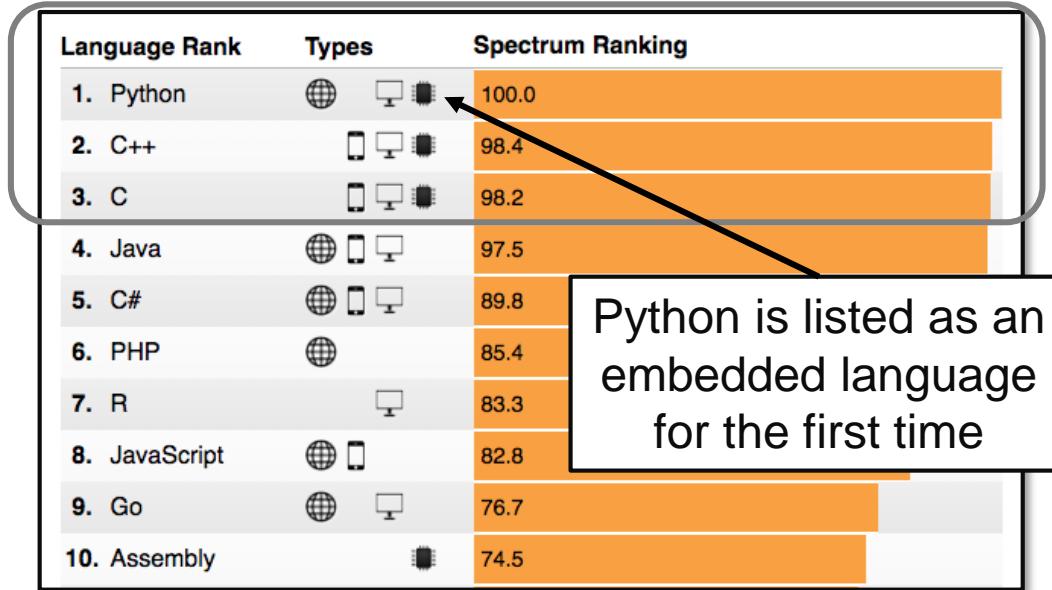


<https://stackoverflow.blog/2017/09/06/incredible-growth-python/>

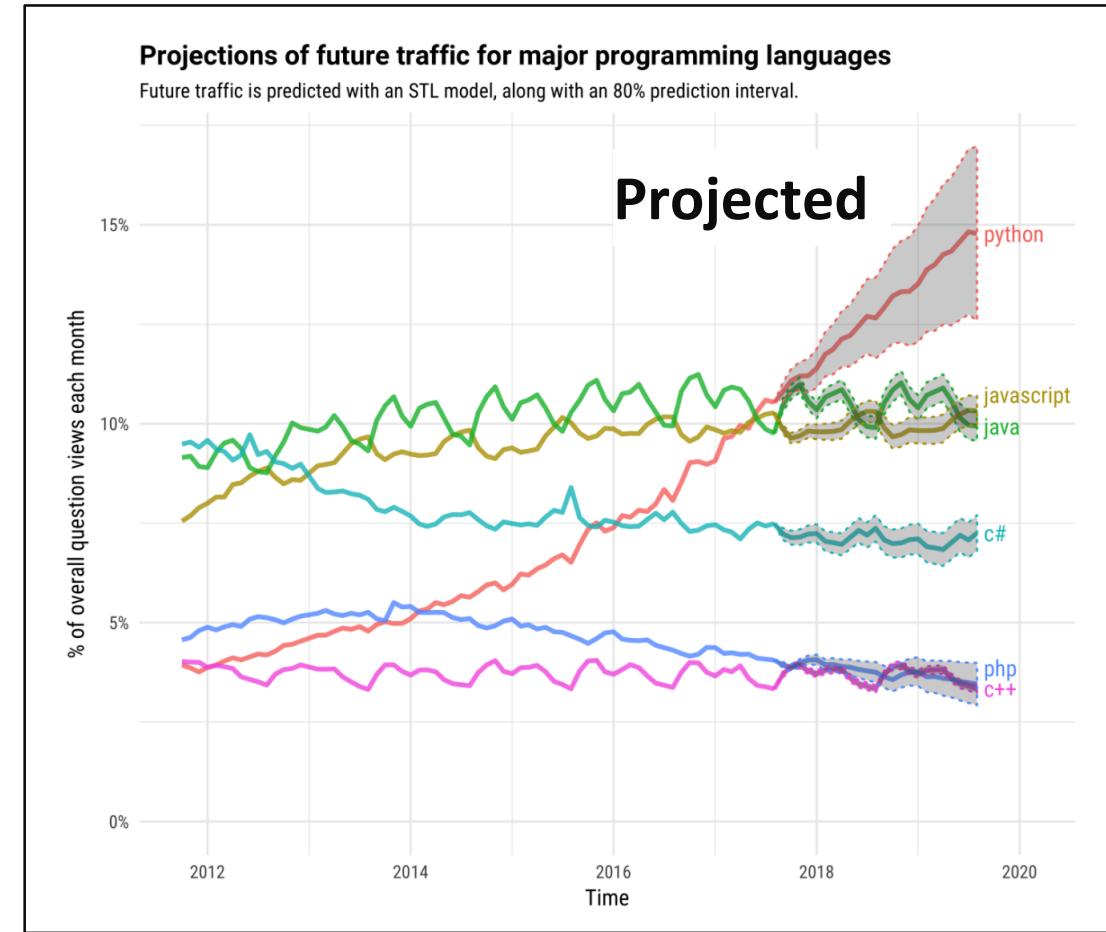
Python is the fastest growing language: driven by data science, AI, ML and academia

# Python is increasingly the Language of Choice

Top Programming Languages,  
IEEE Spectrum, July'18



<https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages>



<https://stackoverflow.blog/2017/09/06/incredible-growth-python/>

Python is the fastest growing language: driven by data science, AI, ML and academia

# Ecosystem advantage: there's a Python library for that...

176,635 projects    1,291,618 releases    1,847,881 files    322,040 users

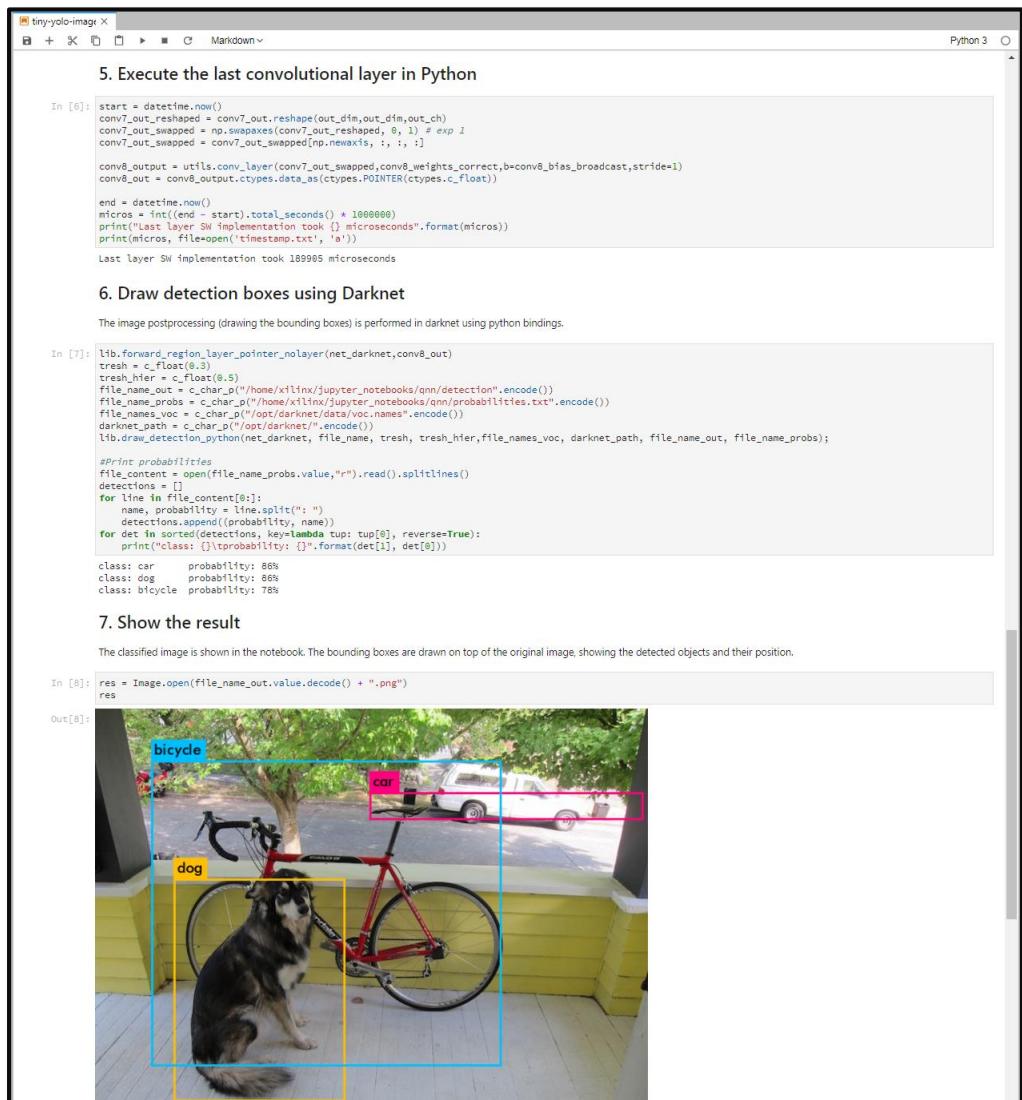


The Python Package Index (PyPI) is a repository of software for the Python programming language. PyPI helps you find and install software developed and shared by the Python community. Learn about installing packages. Package authors use PyPI to distribute their software. Learn how to package your Python code for PyPI.

<https://pypi.org> retrieved 19 Apr 2019

C<sub>1</sub>Python is written in C ... and most popular C/C++ frameworks have Python libraries

# Jupyter Notebooks ... the engine of data science



The screenshot shows a Jupyter Notebook interface with several code cells and their outputs.

**In [6]:**

```
5. Execute the last convolutional layer in Python
In [6]: start = datetime.now()
conv7_out_reshaped = conv7_out.reshape(out_dim,out_dim,out_ch)
conv7_out_swapped = np.swapaxes(conv7_out_reshaped, 0, 1) # exp 1
conv7_out_swapped = conv7_out_swapped[np.newaxis, :, :, :]
conv8_output = utils.conv_layer(conv7_out_swapped,conv8_weights_correct,b=conv8_bias_broadcast,stride=1)
conv8_out = conv8_output.ctypes.data_as(ctypes.POINTER(ctypes.c_float))

end = datetime.now()
micros = int((end - start).total_seconds()) * 1000000
print("last layer SW implementation took {} microseconds".format(micros))
print(micros, file=open('timestamp.txt', 'a'))

Last layer SW implementation took 189965 microseconds
```

**In [7]:**

```
6. Draw detection boxes using Darknet
The image postprocessing (drawing the bounding boxes) is performed in darknet using python bindings.

In [7]: lib.forward_region_layer_pointer_nolayer(net_darknet,conv8_out)
tresh = c_float(0.3)
tresh_hier = c_float(0.8)
file_name_out = c_char_p("/home/xilinx/jupyter_notebooks/qnn/detection".encode())
file_name_probs = c_char_p("/home/xilinx/jupyter_notebooks/qnn/probabilities.txt".encode())
file_name_voc = c_char_p("/opt/darknet/data/voc.names".encode())
darknet_path = c_char_p("/opt/darknet/.encode()")
lib.draw_detection_python(net_darknet, file_name, tresh, tresh_hier,file_names_voc, darknet_path, file_name_out, file_name_probs);

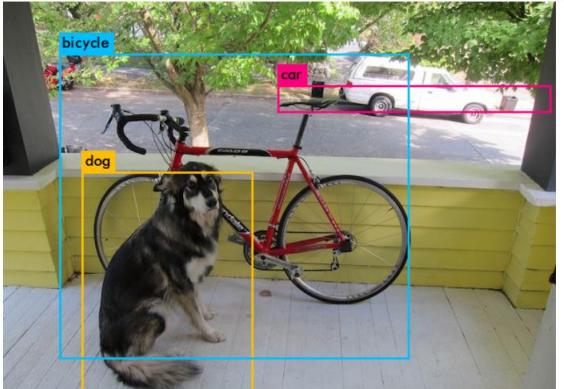
#print(probabilities
file_content = open(file_name_probs.value.decode(),"r").readlines()
detections = []
for line in file_content[0:]:
    name, probability = line.split(": ")
    detections.append((probability, name))
for det in sorted(detections, key=lambda tup: tup[0], reverse=True):
    print("class: {} \tprobability: {}".format(det[1], det[0]))
```

**In [8]:**

```
7. Show the result
The classified image is shown in the notebook. The bounding boxes are drawn on top of the original image, showing the detected objects and their position.

In [8]: res = Image.open(file_name_out.value.decode() + ".png")
res
```

**Out[8]:**



Open source browser-based, executable documents

Live code, text, multimedia, graphics, equations, widgets ...

1.7 million notebooks on GitHub

Taught to 1,000+ Berkeley data science students

# JupyterLab: web-based IDE incl. Notebooks

This screenshot shows the JupyterLab interface. On the left is a sidebar with various file operations like 'New Julia 0.4.5 console', 'New Python 2 console', etc. The main area has a terminal window showing help documentation for Python's 'object'. Below it is a Python notebook cell (In [1]) containing code to plot histograms of beta bands from an EEG dataset. The resulting histogram is displayed as a multi-colored plot. Another cell (In [2]) runs a script to load an MRI image and plot its density across time.

Jupyter Notebook is now one of many plug-ins within the JupyterLab integrated development environment

This screenshot shows the JupyterLab interface with a file browser on the left listing various Jupyter-related files and folders. The main area contains a Python notebook cell (In [1]) demonstrating a polar plot of MRI intensity. The plot shows concentric rings of color representing intensity levels. Another cell (In [2]) shows code for generating a similar histogram to the one in the first screenshot. A terminal window at the bottom shows system statistics like CPU usage and memory usage.

JupyterLab - an open-source, extensible IDE in a browser

# PYNQ enabled boards



# PYNQ-enabled boards

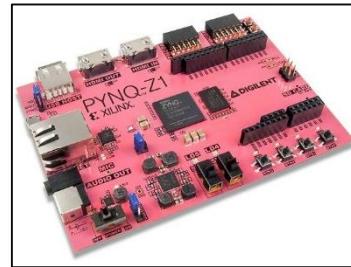
ZYNQ™

ZYNQ®  
MPSoC

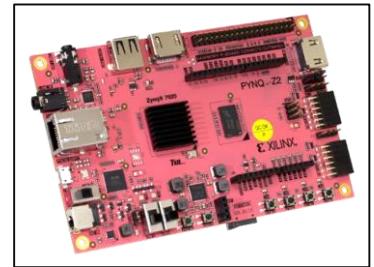
ZYNQ®  
RFSoC

## > Python productivity for Zynq

- >> Open source
- >> Build image for other Zynq boards



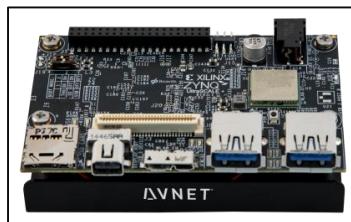
PYNQ-Z1



PYNQ-Z2

## > Downloadable SD card image

- >> Zynq 7000
  - PYNQ-Z1 (Digilent)
  - PYNQ-Z2 (TUL)
- >> Zynq MPSoC
  - Ultra96 (Avnet)
  - ZCU104 (Xilinx)
- >> Zynq RFSoC
  - ZCU111 RFSoC (Xilinx)



Ultra96

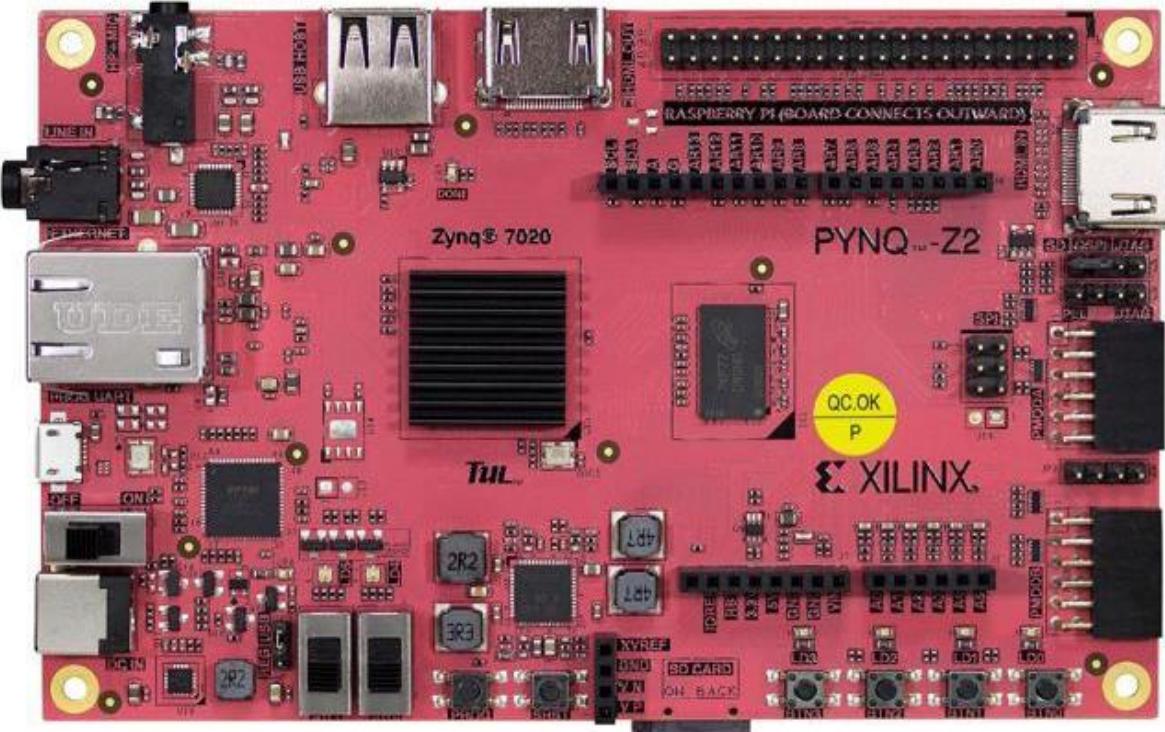


ZCU104



ZCU111

# New PYNQ-Z2 Board



\$119/€119 or equivalent

- New PYNQ reference platform
- New stereo audio with on-board codec
- New Raspberry Pi connector
- Open source design
- Z2 manufactured in Taiwan by TUL
- Distributed globally by Premier Farnell
- Also Newegg in US
- Academic discounts & donations available

# Benefits of PYNQ



# Start using PYNQ out-of-the-box

```
Starting /etc/rc.local Compatibility...
[ OK ] Started System Logging Service.
[ OK ] Started Permit User Sessions.
[ OK ] Started Enable support for additional executable binary formats.
[ OK ] Started LSB: Set the CPU Frequency Scaling governor to "ondemand".
[ OK ] Started LSB: Load kernel modules needed to enable cpufreq scaling.
[ OK ] Started LSB: starts/stops the 2ping listener.
[ OK ] Started LSB: Start NTP daemon.
Stopping LSB: Start NTP daemon...
Starting LSB: set CPUFreq kernel parameters...
[ OK ] Started Login Service.
[ OK ] Started LSB: set CPUFreq kernel parameters.
[ OK ] Stopped LSB: Start NTP daemon.
[ OK ] Created slice user-0.slice.
Starting User Manager for UID 0...
[ OK ] Started Session c1 of user root.
Starting LSB: Start NTP daemon...
[ OK ] Started User Manager for UID 0.
[ OK ] Started LSB: Start NTP daemon.
rc.local[1449]: /root/2_jupyter_server.sh: Jupyter server started
[ OK ] Started Session c2 of user root.
rc.local[1449]: /root/3_pl_server.sh: Programmable Logic server started
[ OK ] Started Session c3 of user root.
[ OK ] Started /etc/rc.local Compatibility.
[ OK ] Started Serial Getty on ttyPS0.
[ OK ] Started Getty on tty1.
[ OK ] Reached target Login Prompts.
[ OK ] Started LSB: start Samba SMB/CIFS daemon (smbd).
[ OK ] Reached target Multi-User System.
[ OK ] Reached target Graphical Interface.
Starting Update UTMP about System Runlevel Changes...
[ OK ] Started Update UTMP about System Runlevel Changes.

Ubuntu 15.10 pynq ttyPS0

pynq login: xilinx (automatic login)

Last login: Thu Jan  1 00:00:12 UTC 1970 on ttyPS0
xilinx@pynq:~$
```

> **PYNQ delivered as downloadable SD card image**

  >> Linux preconfigured

> **Additional packages and drivers pre-installed**

  >> USB peripheral drivers: webcams, wifi modules ...

> **PYNQ is for Zynq**

  >> PYNQ image is portable to other Zynq boards

Start using Zynq out of the box ✓

# Desktop Linux

## > Network/Internet access

- >> “apt-get” to install packages from Ubuntu universe
- >> Samba(Network drive)
- >> Web services

## > Git directly on board

## > Compilers and other development tools

- >> Gcc., MicroBlaze, RISC-V ....

## > Python packages

- >> “pip install”
- >> PYNQ Community examples

A selection of projects from the PYNQ community is shown below. Note that some examples are built on different versions of the PYNQ image.

**Binary Neural Network**  
Xilinx Labs, NTNU,  
University Sydney



**SPynq:**  
NTUA Greece



**Processing noisy filters**  
PYNQ Japan user group



**Soft GPU for ZC706**  
Ruhr University Bochum



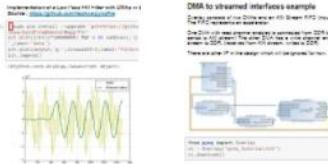
**Video Processing**  
VectorBlox



**FIR filter example**  
CU Boulder



**DMA and stream**  
Tutorial



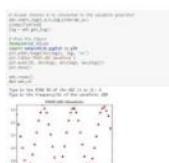
**CNN Example**  
Imperial College London



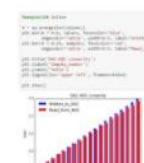
## Example Notebooks

A selection of notebook examples are shown below that are included in the PYNQ image. The notebooks contain live code, and generated output from the code can be saved in the notebook. Notebooks can be viewed as webpages, or opened on a Pynq enabled board where the code cells in a notebook can be executed.

**ADC waveforms**



**DAC ADC example**



**Downloading overlays**



**Grove ADC**



# Simplify downloading bitstreams to PL

- > **PYNQ ‘Overlay’ class**
  - » Simplifies downloading bitstream
  - » two lines of code
  - » No Xilinx tools required
- > **Maintain many bitstreams on the SD card**
  - » E.g. multiple different demos
- > **Can execute Python in browser, or from command line**

```
from pynq import Overlay  
  
ol = Overlay('gray.bit')
```

Simply and fast way to configure Programmable Logic ✓

# Simplify IP debug and prototyping

> Debug of IP typically uses C/C++

> SDK tools used to:

- >> Compile test application
- >> Download application to board
- >> Step through code

> PYNQ MMIO class allows peek/poke of IP registers from Python

- >> Python executes directly on the board
- >> No offline compilation, download loop
- >> No SDK tools

C code – compile, debug from host

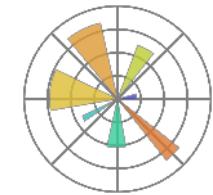
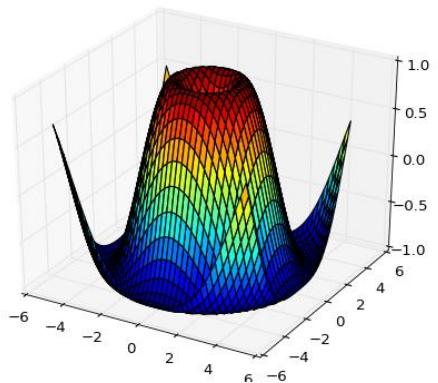
```
/**************************************************************************/  
* This function does a selftest on the IIC device and XIic driver as an  
* example.  
* @param DeviceId is the XPAR_<IIC_instance>_DEVICE_ID value from  
* xparameters.h.  
*****/  
int IicSelfTestExample(u16 DeviceId)  
{  
    Status = XIic_CfgInitialize(&Iic, ConfigPtr, ConfigPtr->BaseAddress);  
    if (Status != XST_SUCCESS) {  
        return XST_FAILURE;  
    }  
  
    /* Perform a self-test to ensure that the hardware was built */  
    Status = XIic_SelfTest(&Iic);  
    if (Status != XST_SUCCESS) {  
        return XST_FAILURE;  
    }  
}
```

Python executes directly on target

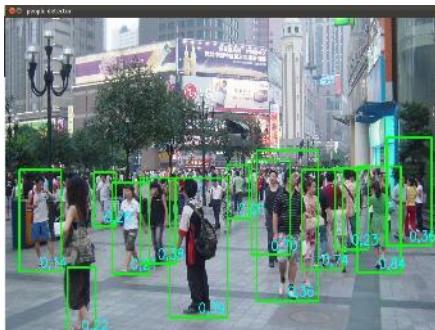
```
from pynq import MMIO  
  
# Map registers to MMIO instance  
my_ip = MMIO(my_ip_addr, LENGTH)  
  
# Write 0x1 to start IP  
my_ip.write(CONTROL_REGISTER, 0x1)  
# Check status register  
my_ip.read(STATUS_REGISTER)
```

Rapid testing and prototyping ✓

# Python packages for data analysis and visualisation



Matplotlib



> **Take advantage of Python for data analysis and processing**

- >> NumPy
  - Scientific computing package for Python
- >> Matplotlib
  - Python 2D plotting library
- >> Pandas
  - Data analysis tools for Python
- >> OpenCV
  - Computer Vision and machine learning software

Optimized open-source software libraries ✓

# Example

## 6. Detailed Classification Information

In addition to highest ranked class, it is possible to get the rank of a couple of images of a car, an airplane, and a bird and place them in a plot.

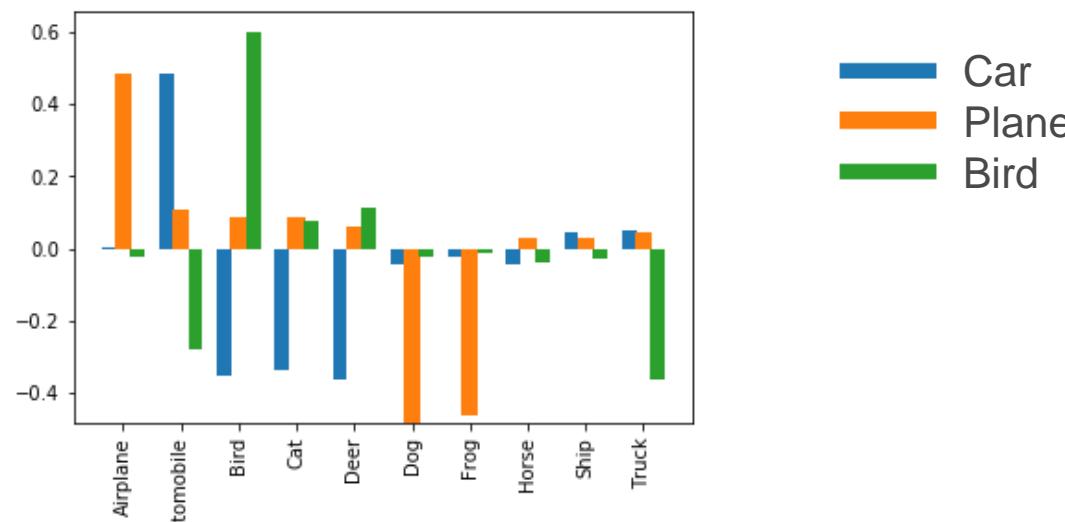


### Probability object belongs to class

```
In [8]: from IPython.display import display
```

```
%matplotlib inline
import matplotlib.pyplot as plt

x_pos = np.arange(len(car_class))
fig, ax = plt.subplots()
ax.bar(x_pos - 0.25, (car_class/255)-1, 0.25)
ax.bar(x_pos, (air_class/255)-1, 0.3)
ax.bar(x_pos + 0.25, (bird_class/255)-1, 0.25)
ax.set_xticklabels(classifier.bnn.classes, rotation='vertical')
ax.set_xticks(x_pos)
ax.set
plt.show()
```



oseconds

images per second

69 163 244 249 244 267 268]

oseconds

images per second

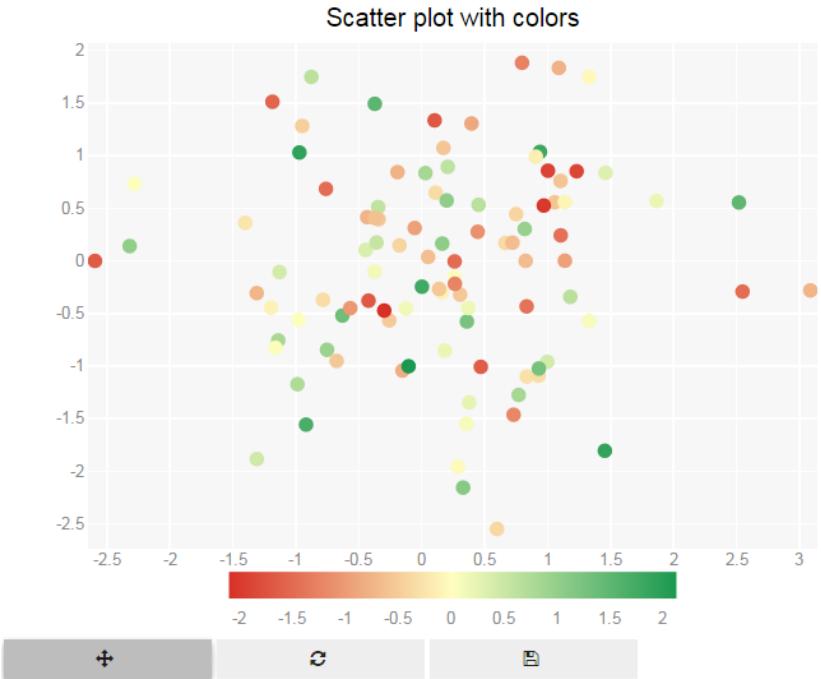
277 277 271 132 137 262 263 266]

oseconds

images per second

274 284 249 252 245 248 163]

# Take advantage of interactive widgets

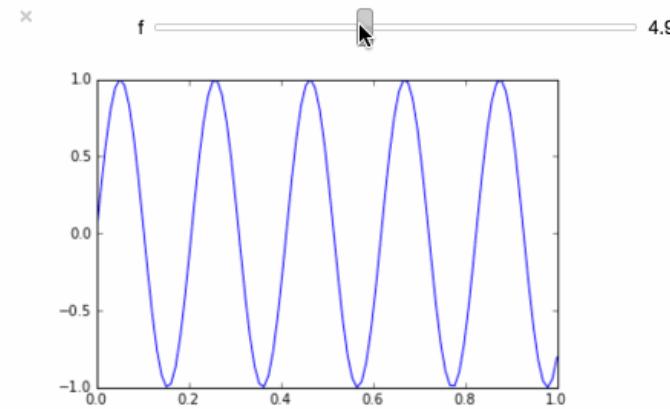


<http://jupyter.org/widgets.html>

```
In [22]: from IPython.html.widgets import *
t = arange(0.0, 1.0, 0.01)
```

```
def pltsin(f):
    plt.plot(x,sin(2*pi*t*f))
    plt.show()

interact(pltsin, f=(1,10,0.1))
```



<https://blog.dominodatalab.com/interactive-dashboards-in-jupyter/>

```
In [7]: p2 = IntProgress(max=56)
p2.value += 10
p2.description = 'Running'
display(p2)
```

ThingA  ThingA (modified)  
ThingB  ThingB  
ThingC  ThingC (modified)  
ThingD  ThingD

show

group\_by

values

stereo   fullscreen

0

Add intuitive graphical interfaces ✓

# Why PYNQ is a Game-changer!

- > **PYNQ makes Zynq/ZynqU+ accessible to non-traditional customers**
- > **PYNQ delivers open source benefits**
  - >> Huge ecosystem
  - >> Extensive knowledge base
  - >> Amazing community support
- > **PYNQ enables highly-productive ...**
  - >> Prototyping
  - >> Debug
  - >> Verification
  - >> Evaluation
- > **PYNQ powers awesome demonstrators**
- > **PYNQ documentation flows are amazing**
  - >> Capture your own work
  - >> Capture work you want to re-use
- > **PYNQ designs can be ... just like software**
  - >> Packaged, published and distributed
- > **“PYNQ makes FPGAs FUN again!”, J. Gray, Xilinx Power User**

# Community

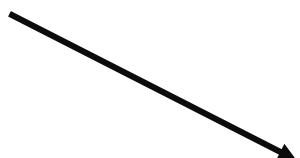




Home Get Started PYNQ-Z1 Bo

## Community Projects

Selection of projects  
and notebooks



A selection of projects from the PYNQ community is shown below. Note that some examples are built on different versions of the PYNQ image.

### Binary Neural Network

Xilinx Labs, NTNU,  
University Sydney

[BINonPynq](#)  
The project uses the Xilinx Zynq SoC to implement a Binary Neural Network. It is a neural network that uses binary weights and activations. It is trained using a sparse gradient descent algorithm.



### SPyng:

NTUA Greece

[SPyng: Python interface](#)  
In this project we use Python to interface the Zynq SoC with external cameras and run them from the Jupyter Notebook. The camera feeds are then processed by the SPyng library to detect objects in real-time.



### Processing noisy filters

PYNQ Japan user group

[ではじめる機械学習](#)  
Gardine Maruya  
This notebook shows how to process noisy filters. It includes a plot of a noisy signal and a plot of the filtered signal.

### Soft GPU for ZC706

Ruhr University Bochum

[Soft GPU for ZC706](#)  
Hans-Joachim Kunkel  
This notebook shows how to use a soft GPU on the ZC706. It includes a plot of a sine wave and a photograph of a person's face.

### Video Processing

Vectorblobx

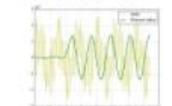
[VectorBlox Video Processing](#)  
In this module, several filters can be applied to YUV420 input images. These filters include motion blur, crop, rotate, and blur. The filters are implemented using Vectorblox's SIMD-based architecture.



### FIR filter example

CU Boulder

[Implementation of a Linear Phase FIR Filter with DMA via IP](#)  
Detailed description of the FIR filter implementation. It shows how to implement a linear phase FIR filter using DMA. The filter is implemented using a convolutional neural network (CNN) architecture.



### DMA and stream

Tutorial

[DMA to streamified interface example](#)  
Detailed description of the DMA and streamified interface example. It shows how to implement a DMA to streamified interface. The DMA is used to transfer data from memory to the streamified interface. The streamified interface is then used to transfer data to a host computer.



### CNN Example

Imperial College London

[FPGA Deep learning CNN example](#)  
Detailed description of the FPGA Deep learning CNN example. It shows how to implement a CNN on an FPGA. The CNN is implemented using a convolutional neural network (CNN) architecture.



## Example Notebooks

A selection of notebook examples are shown below that are included in the PYNQ image. The notebooks contain live code, and generated output from the code can be saved in the notebook. Notebooks can be viewed as webpages, or opened on a Pynq enabled board where the code cells in a notebook can be executed.

### ADC waveforms

Yannick Gobet

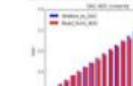
[ADC waveforms](#)  
Detailed description of the ADC waveforms example. It shows how to capture ADC waveforms. The code includes a plot of the captured waveforms.



### DAC ADC example

Yannick Gobet

[DAC ADC example](#)  
Detailed description of the DAC ADC example. It shows how to interface a DAC and ADC. The code includes a plot of the generated DAC waveform and a plot of the captured ADC waveform.



### Downloading overlays

Yannick Gobet

[Downloading Overlays](#)  
Detailed description of the Downloading Overlays example. It shows how to download an FPGA overlay and update it. The code includes a plot of the captured waveform.



### Grove ADC

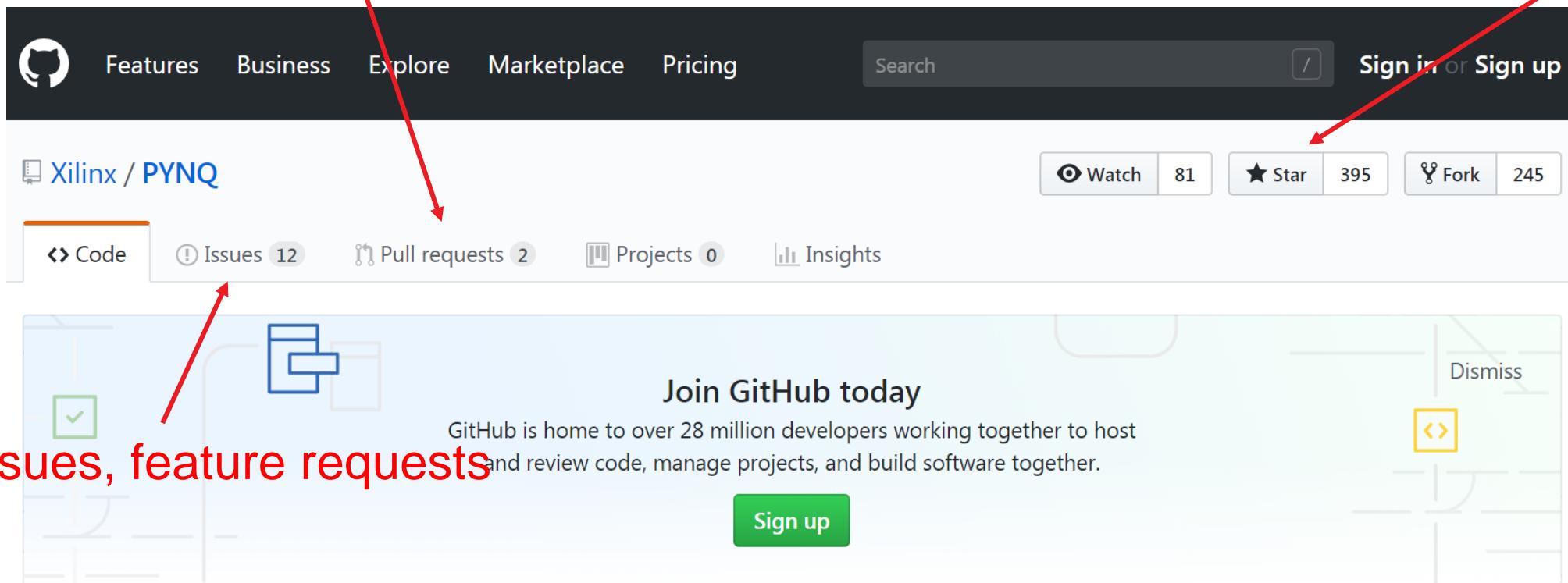
Yannick Gobet

[Grove ADC](#)  
Detailed description of the Grove ADC example. It shows how to interface a Grove ADC. The code includes a plot of the captured waveform.



# All Feedback helps

Contribute



Issues, feature requests

Python Productivity for ZYNQ <http://www.pynq.io/>

pynq

# Summary



- > PYNQ is Python productivity for Zynq
- > Everything runs on Zynq, access via a browser
- > Support for Zynq Ultrascale+
- > Overlays are hardware libraries and enable software developers to use Zynq
- > Provides a rapid prototyping framework for hardware developers



[pynq.io](https://pynq.io)

[pynq.readthedocs.org](https://pynq.readthedocs.org)

[github.com/Xilinx/PYNQ](https://github.com/Xilinx/PYNQ)

[tul.com.tw/ProductsPYNQ-Z2.html](https://tul.com.tw/ProductsPYNQ-Z2.html)

[pynq.io/support](https://pynq.io/support)

# Adaptable. Intelligent.

