# Machine Learning for Embedded Demo

**Quenton Hall**
**Avnet Field Applications Engineer | ML Specialist**
**Boston | March 14**

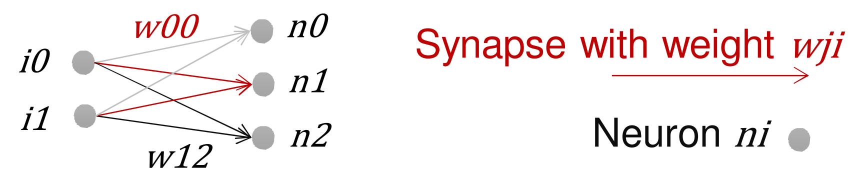
*Slide and diagram credits:*
*Kaiming He | Xiangyu Zhang | Shaoqing Ren | Jian Sun | Clayton Cameron | Michaela Blott | Andy Luo*

XILINX

# Neural Nets - A Nickel Tour

# Convolutional Neural Networks (CNNs)
## *from a computational point of view*

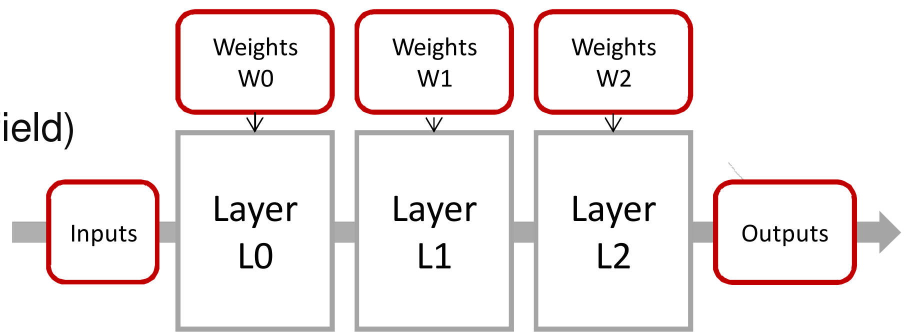
- **CNNs are usually feed forward* computational graphs constructed from one or more layers**
  - Up to 1000s of layers




- **Each layer consists of neurons *ni* which are interconnected with synapses, associated with weights *wij***

$$n0 = Act(w00*i0 + w10*i1)$$

- **Each neuron computes:**
  - Typically linear transform (dot-product of receptive field)
  - Followed by a non-linear “activation” function



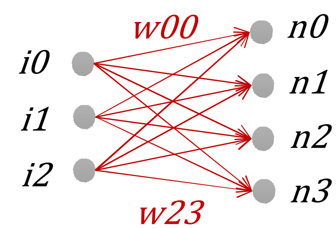

* With exception of RNNs



XILINX

# Fully Connected Layers

## *(aka inner product or dense layers)*




- **Each input activation is connected to every output activation**
  - Receptive field encompasses the full input

- **Can be written as a matrix-vector product with an element-wise non-linearity applied afterwards.**

$$\begin{pmatrix} i0 & i1 & i2 \end{pmatrix} \times \begin{pmatrix} W00 & W01 & W02 & W03 \\ W10 & W11 & W12 & W13 \\ W20 & W21 & W22 & W23 \end{pmatrix} = \begin{pmatrix} n0' & n1' & n2' & n3' \end{pmatrix}$$

$$(n0\ n1\ n2\ n3) = Act(\ n0'\ n1'\ n2'\ n3')$$

- **Implementation Challenges**
  - Connectivity
  - High weight memory requirement: #IN * #OUT * BITS
  - Low arithmetic intensity assuming weights off-chip
    2 * #IN* #OUT / #IN * #OUT * BITS/8

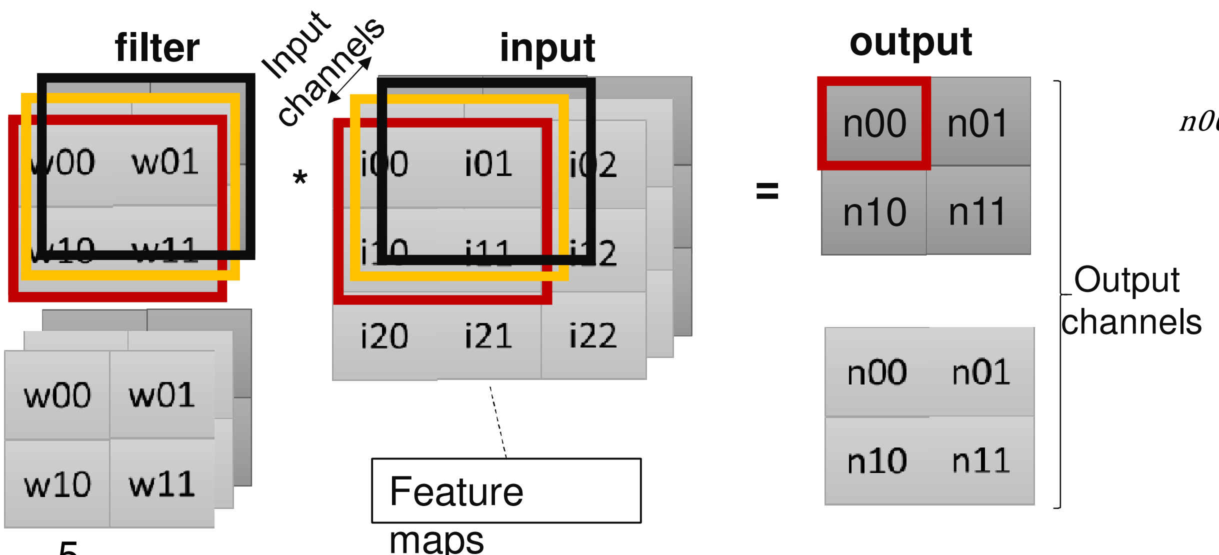
| MODEL | CONV WEIGHTS (M) | FC WEIGHTS (M) |
|---|---|---|
| ResNet50 | 23.454912 | 2.048 |
| AlexNet | 2.332704 | 58.621952 |
| VGG16 | 14.710464 | 123.633664 |

*IN:* *number of input channels*
*OUT:* *number of output channels*
*BITS:* *bit precision in data types*

# Convolutional Layers

## *Example 2D Convolution*

- **Convolutions capture some kind of locality, spatial or temporal, that we know exists in the domain**
- **Receptive field of each neuron reduced**
  - Applying convolution to all images in the previous layer
- **Weights represent the filters used for convolutions**

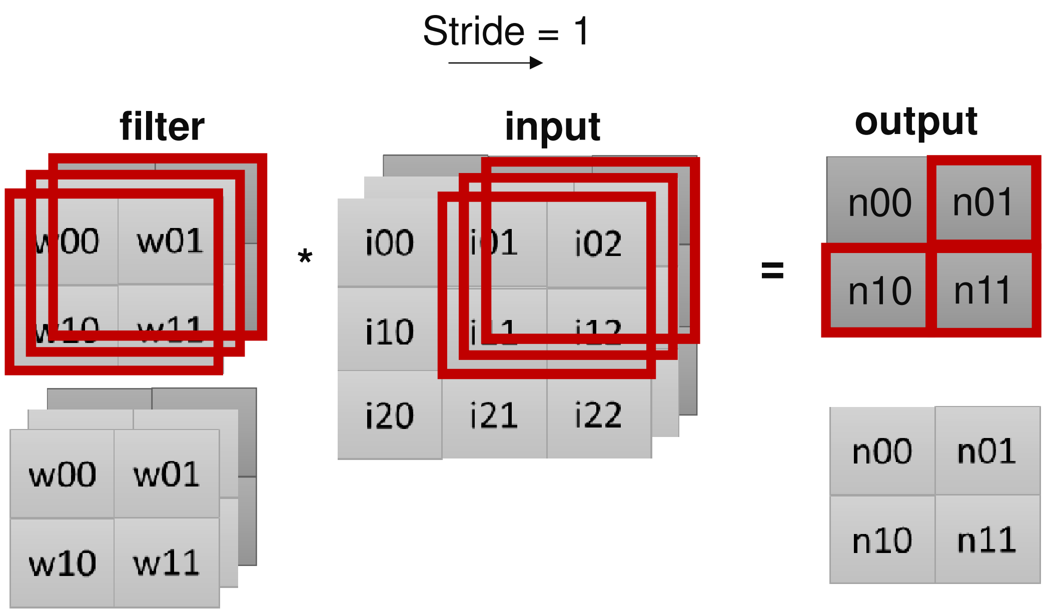



Input channel 0

$$n00 = Act(w00*i00 + w01*i01 + w10*i10 + w11*i11 + w00*i00 + w01*i01 + w10*i10 + w11*i11 + w00*i00 + w01*i01 + w10*i10 + w11*i11\,)$$

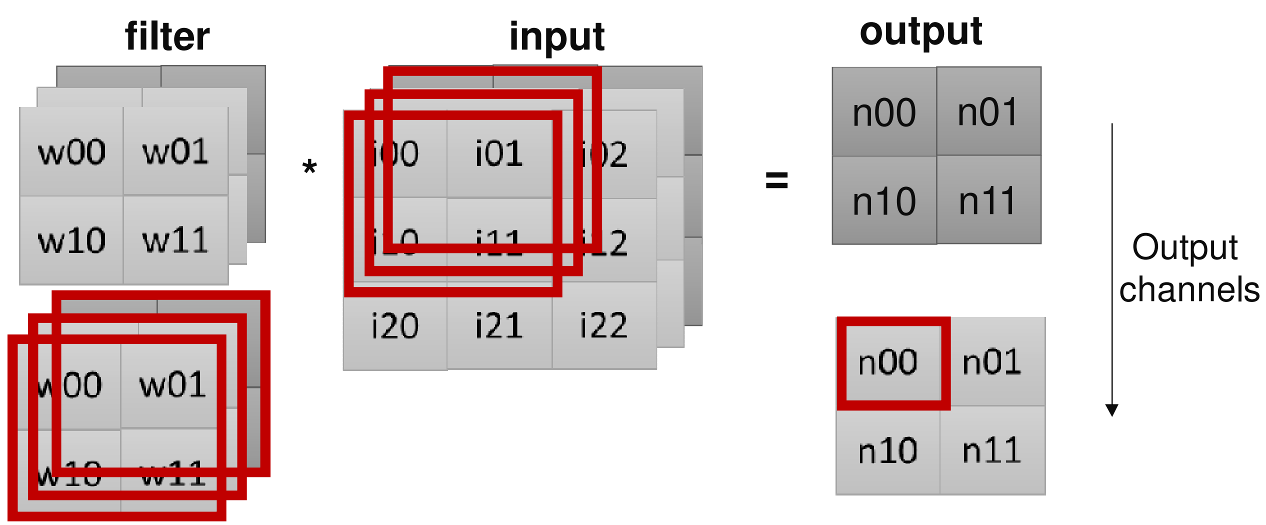
# 2D Convolutional Layers

- Slide the window till one feature map is complete
  - With a given stride size



# 2D Convolutional Layers

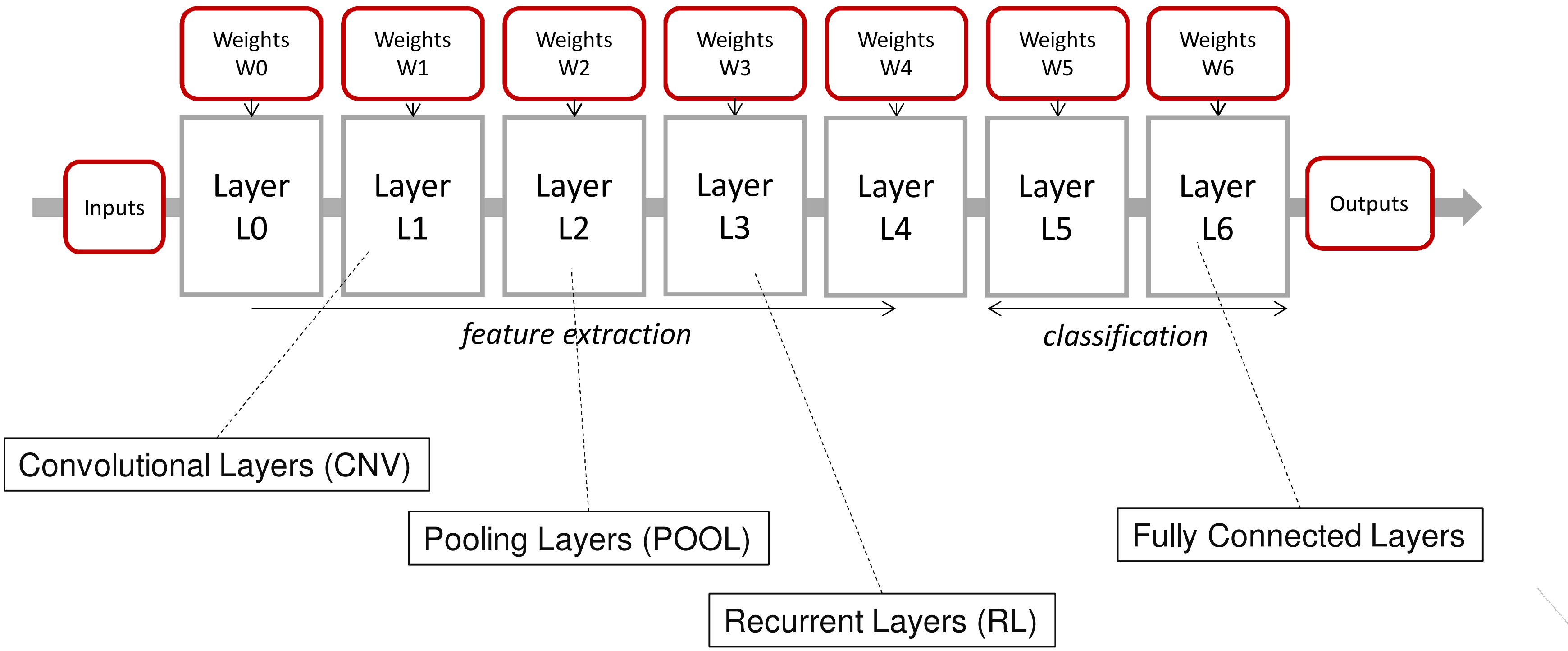
> **Compute next channel**






XILINX.

# NNs in More Detail

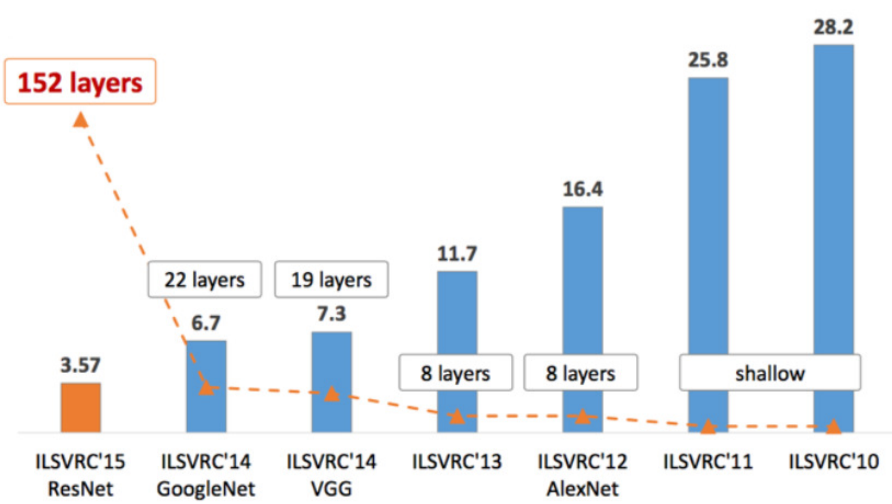



Activation & Batch Normalization

# ResNet – A brief history

# Image Classification - ImageNet

- In 2009, Fei-Fei Li introduced the ImageNet dataset
  - >14 Million images, 40000 object classes
- ImageNet Large Scale Visual Representation Challenge – ILSVRC
  - Subset of 1000 object classes, 1.2 Million images



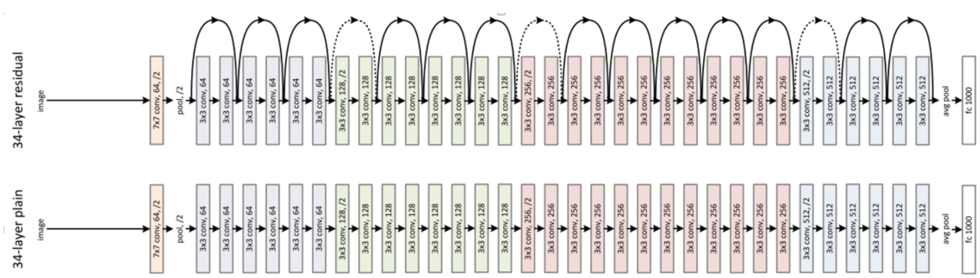

imagenet1000_clsid_to_human.txt

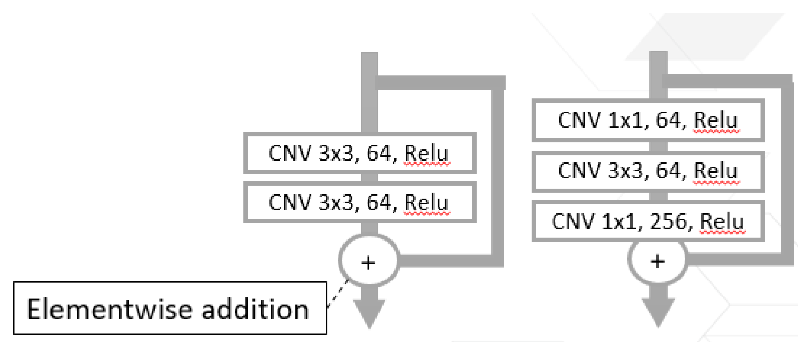
```
{0: 'tench, Tinca tinca',
 1: 'goldfish, Carassius auratus',
 2: 'great white shark, white shark, man-eater, man-eating shark, Carcharodon carcharias',
 3: 'tiger shark, Galeocerdo cuvieri',
 4: 'hammerhead, hammerhead shark',
 5: 'electric ray, crampfish, numbfish, torpedo',
 6: 'stingray',
 7: 'cock',
 8: 'hen',
 9: 'ostrich, Struthio camelus',
 10: 'brambling, Fringilla montifringilla',
 11: 'goldfinch, Carduelis carduelis',
 12: 'house finch, linnet, Carpodacus mexicanus',
 13: 'junco, snowbird',
 14: 'indigo bunting, indigo finch, indigo bird, Passerina cyanea',
 15: 'robin, American robin, Turdus migratorius',
```

*Image Credit: Kaiming He [http://kaiminghe.com/](http://kaiminghe.com/)*

# ResNets

- **Deep networks suffer from the “vanishing gradient” problem**
  - During back propagation, weight values in deep networks may not change significantly during the backward pass
    - Impacts our ability to train deep networks
- **ResNet was the first network architecture to employ “skip connections” which made it possible to train deeper networks with higher accuracy**
  - https://arxiv.org/abs/1512.03385
- **ResNet50 is so called because the architecture includes 50 convolution layers**

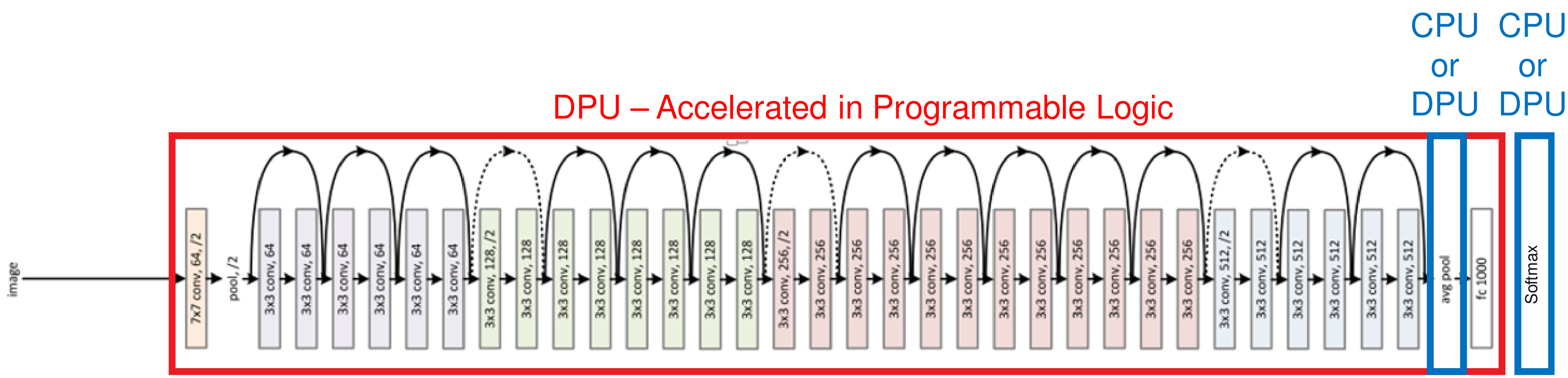





XILINX.

# DNNDK ResNet Inference




For ResNet50:

- 70 Layers
- 7.7 Billion operations
- 25.5 MBytes of weight storage*
- 10.1 MBytes for activations*

**Assuming int8*

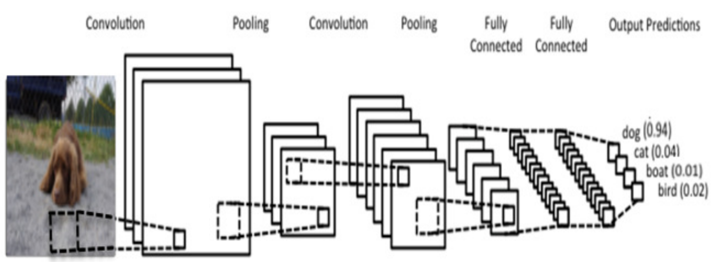


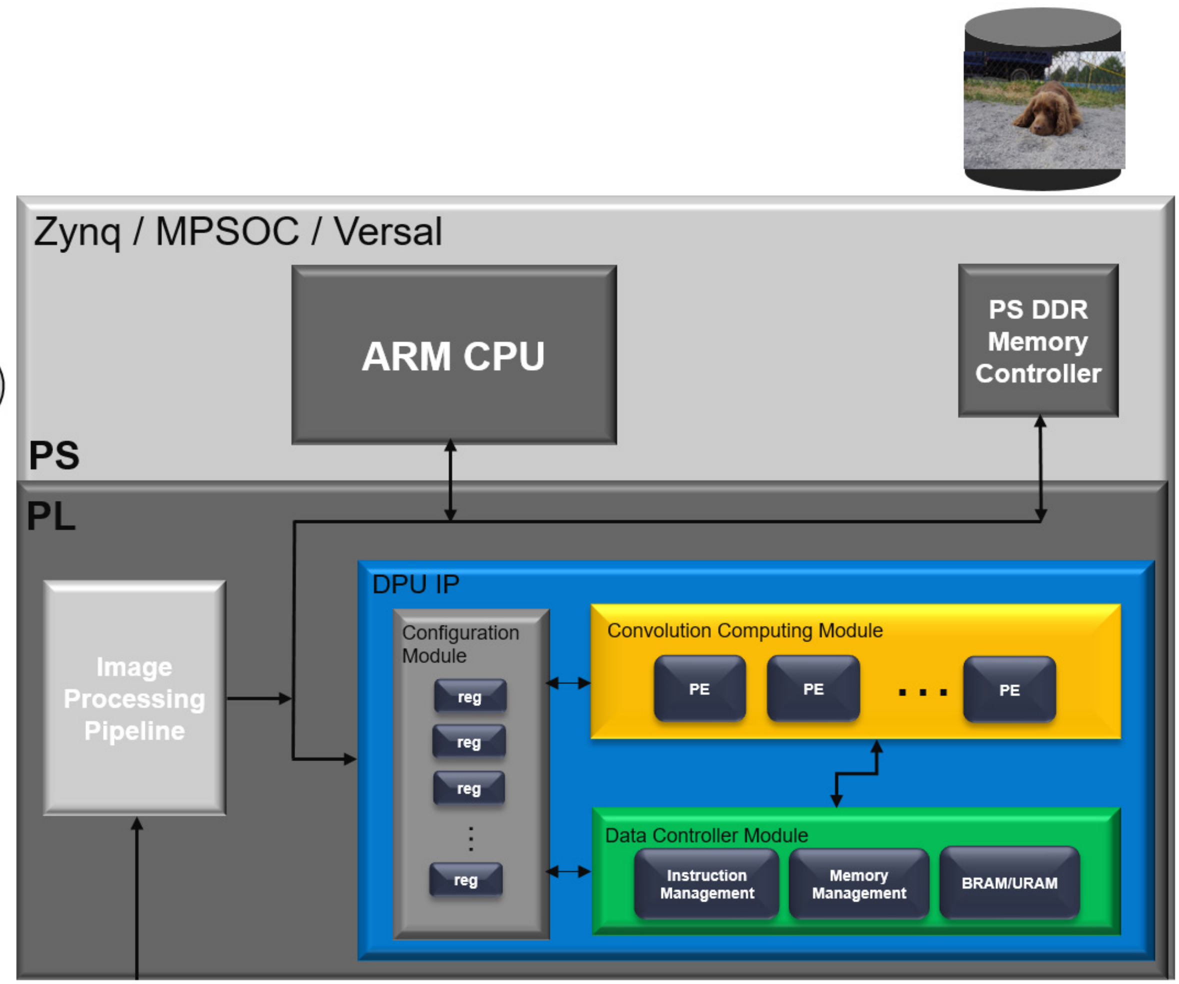
XILINX.

# Network Inference with DNNDK

# DPU Data Flow

ARM reads 960x540 video frames
from SD into PS DDR

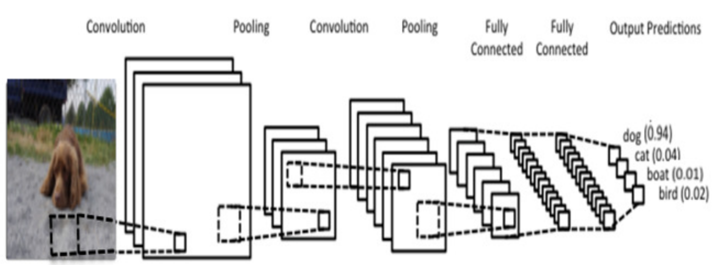


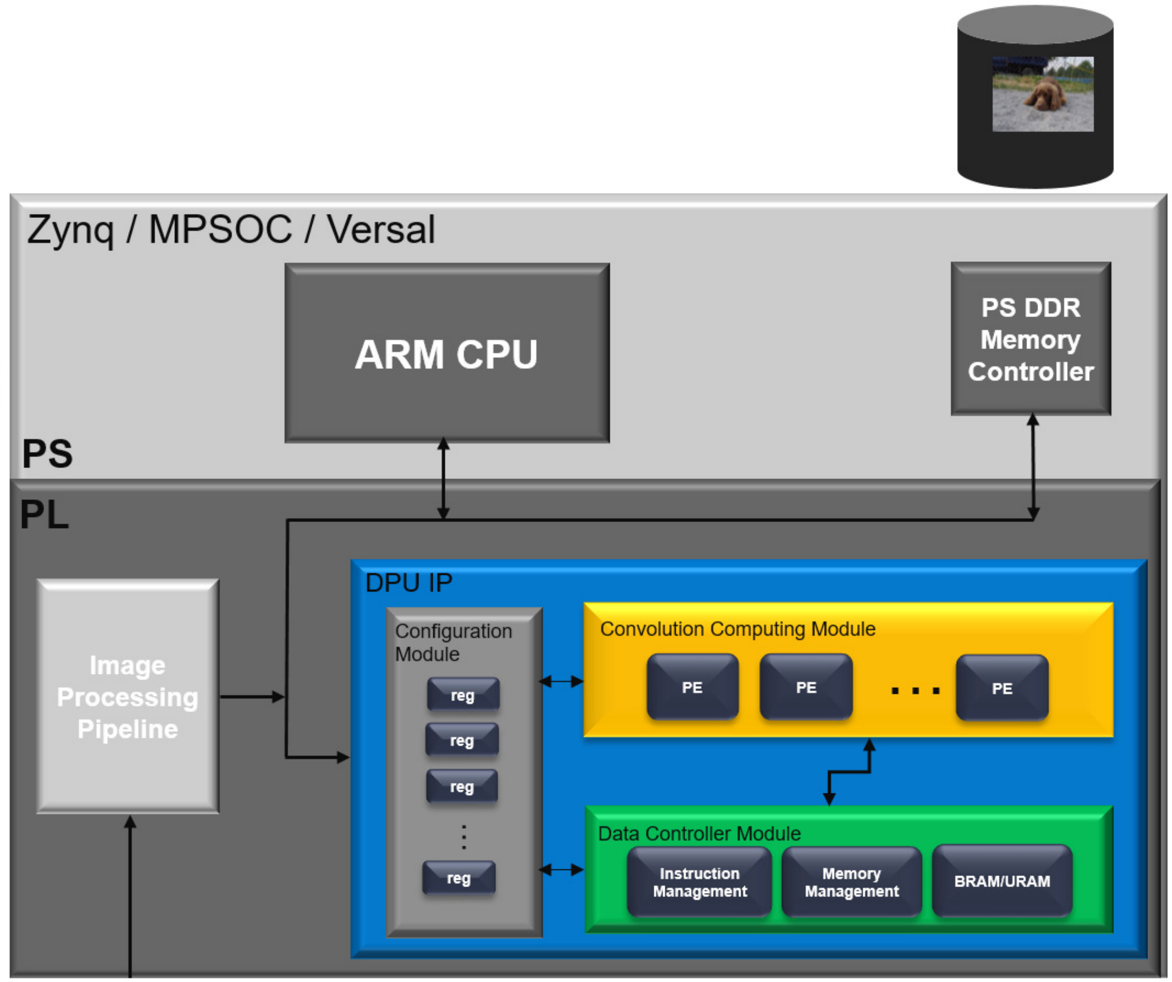




*Slide and animation credit – Clayton Cameron and family*

# DPU Data Flow

ARM or PL scales each frame to the the required input dims of the network (ie 227x227x3)

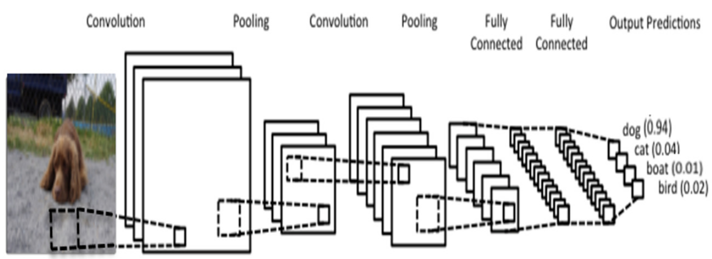


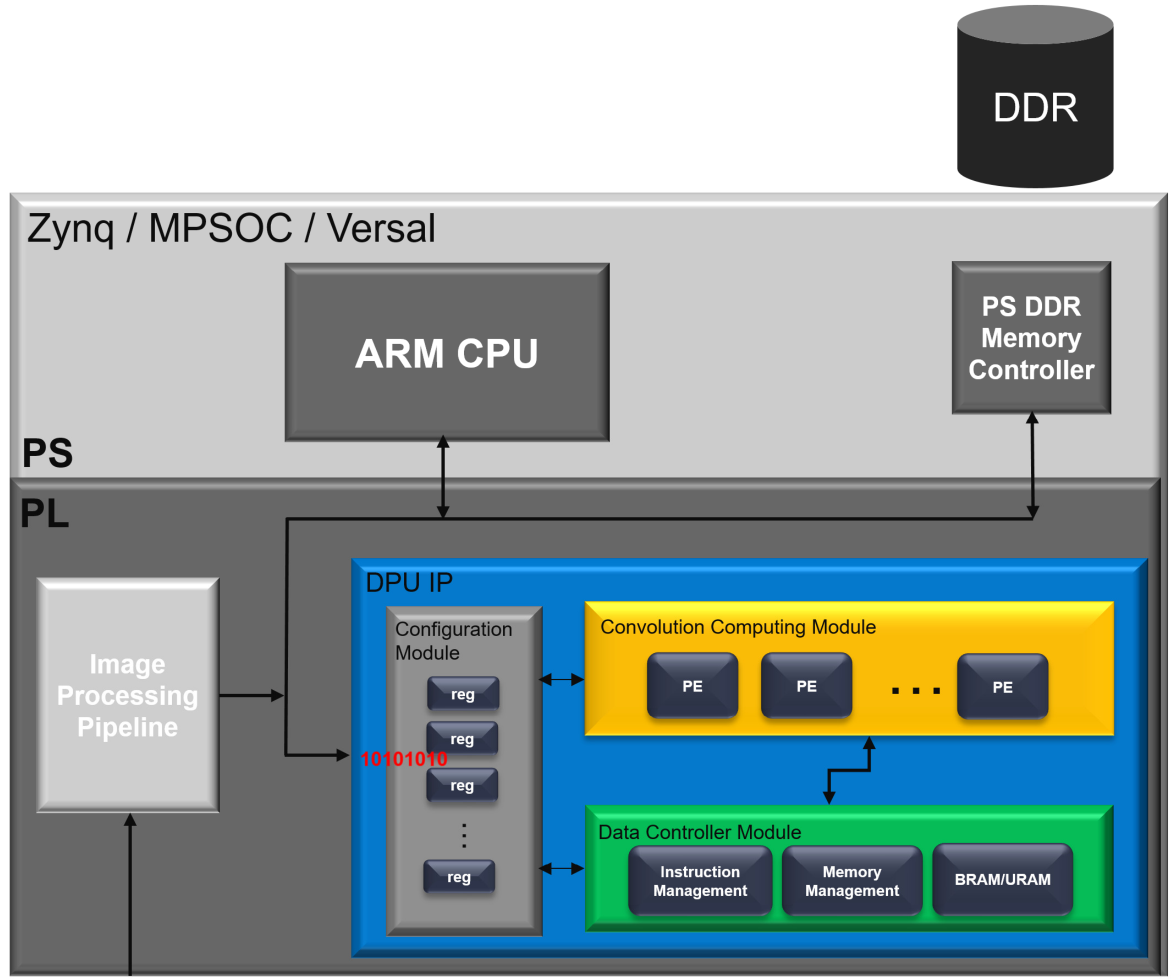




*Slide and animation credit – Clayton Cameron and family*

# DPU Data Flow

ARM writes the Configuration Module to setup DPU for image processing and starts execution



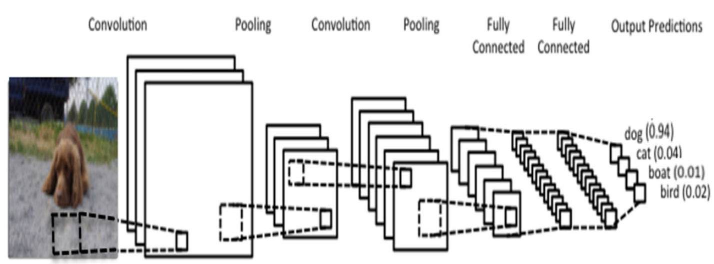



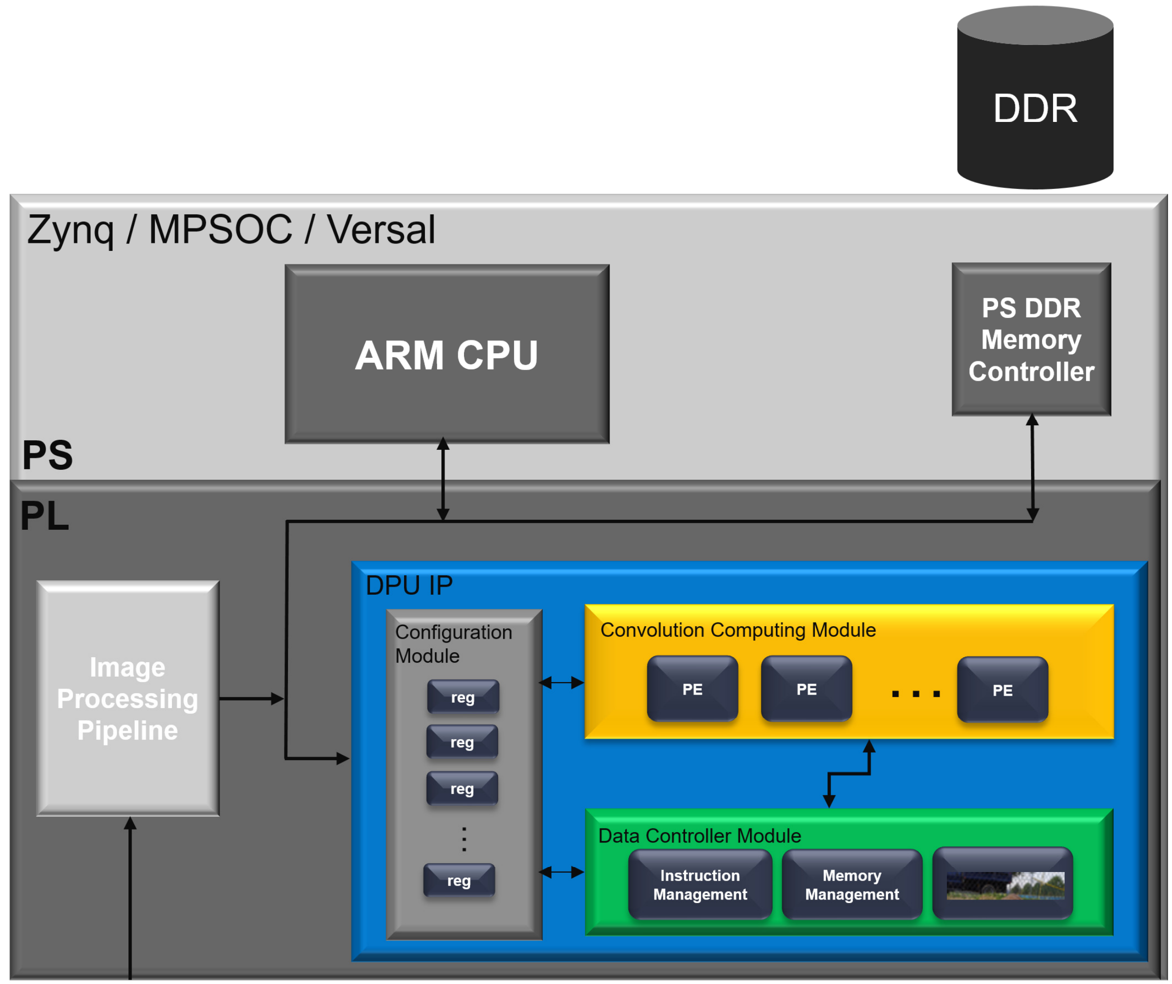

*Slide and animation credit – Clayton Cameron and family*

XILINX.

# DPU Data Flow

DPU DMA engine fetches input activation and weights from PS DDR

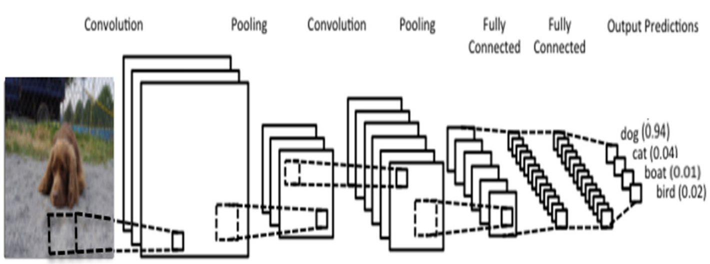


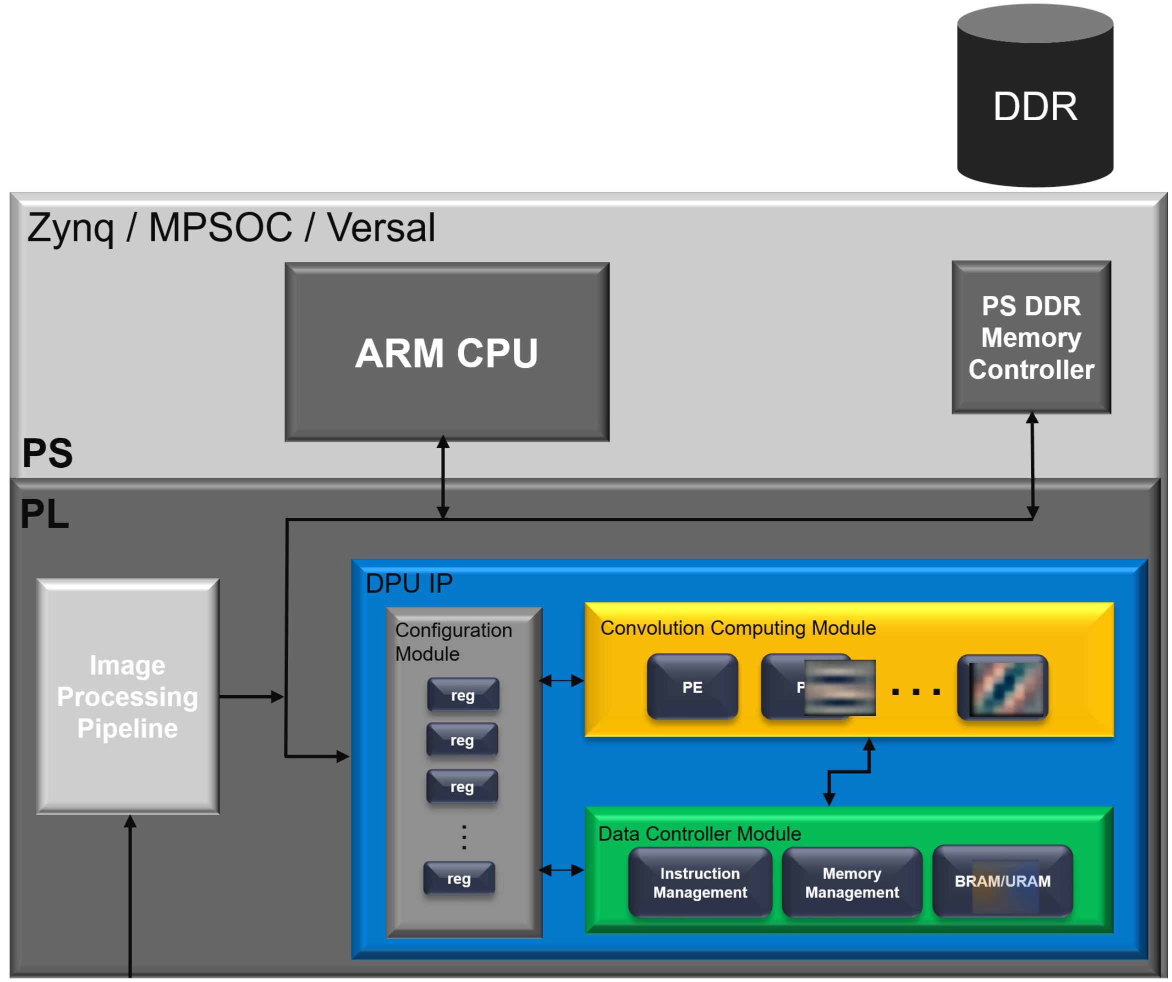




*Slide and animation credit – Clayton Cameron and family*

# DPU Data Flow

The DPU calculates intermediate output activations

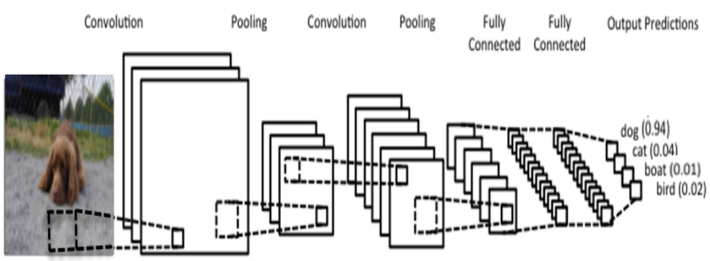


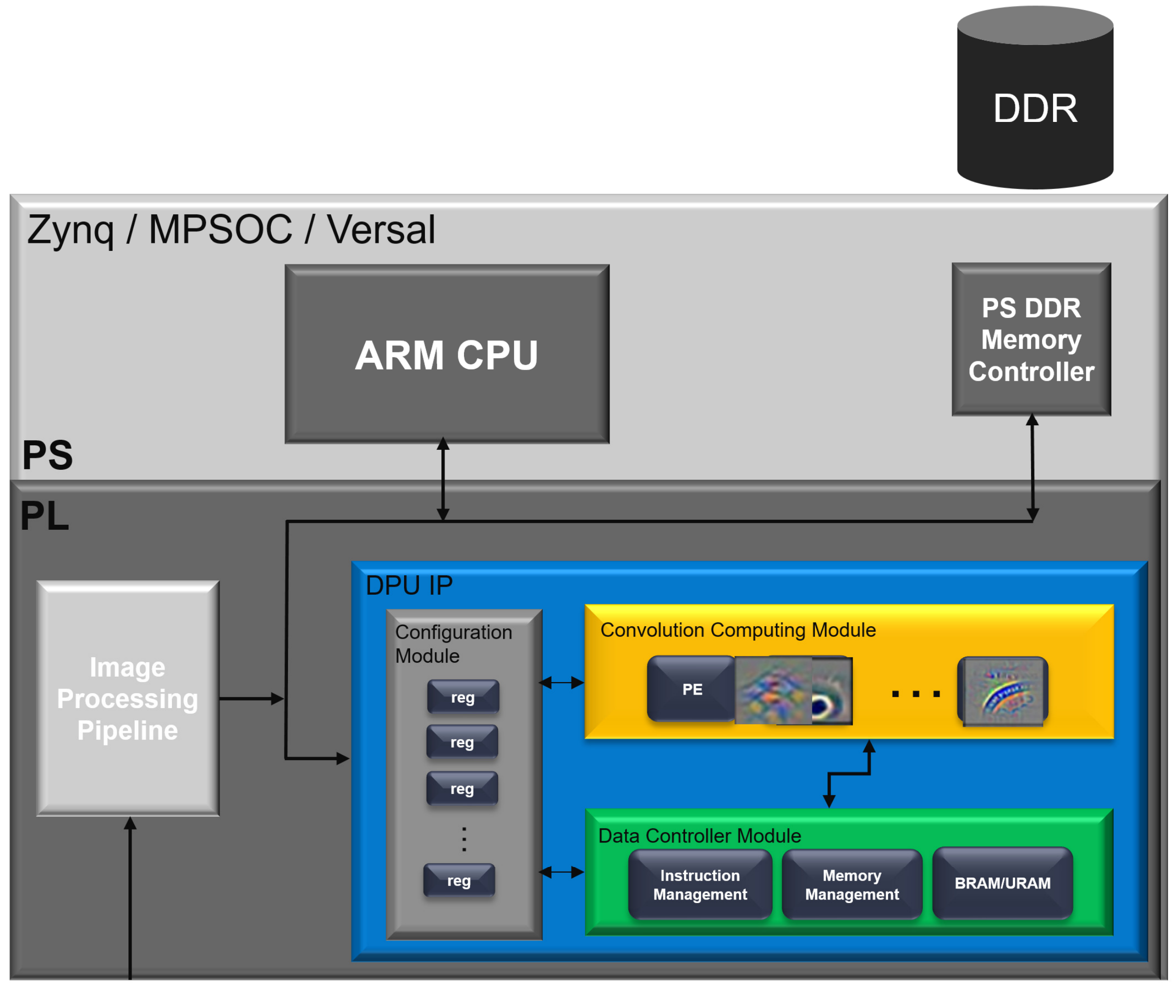




*Slide and animation credit – Clayton Cameron and family*

# DPU Data Flow

The DPU continues this process until all the output activations are complete

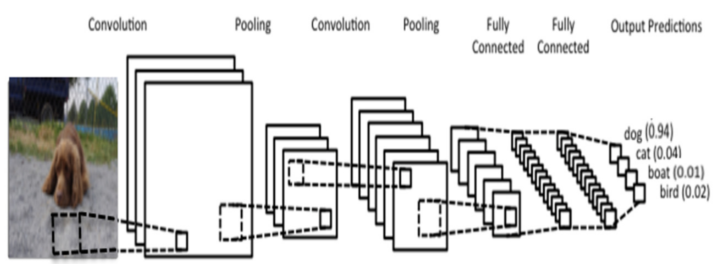


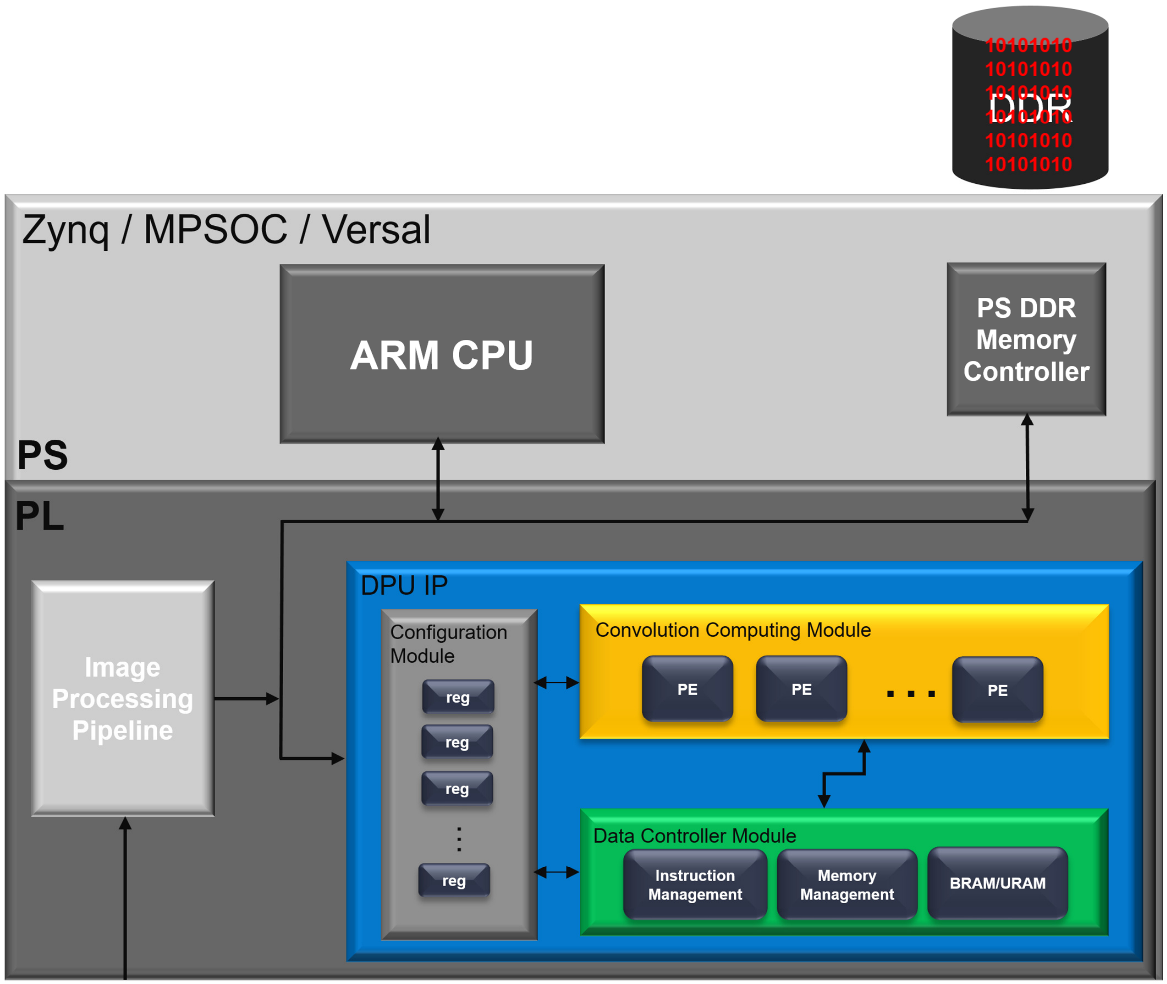




*Slide and animation credit – Clayton Cameron and family*

# DPU Data Flow

DPU DMAs output activations back to PS DDR



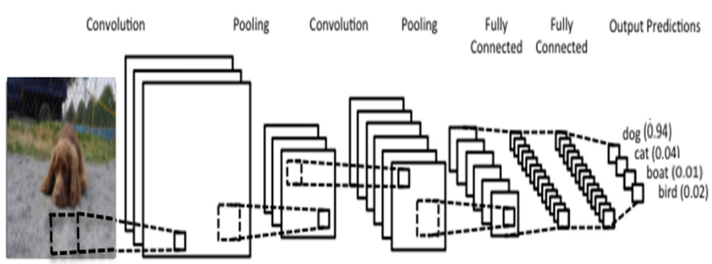




*Slide and animation credit – Clayton Cameron and family*

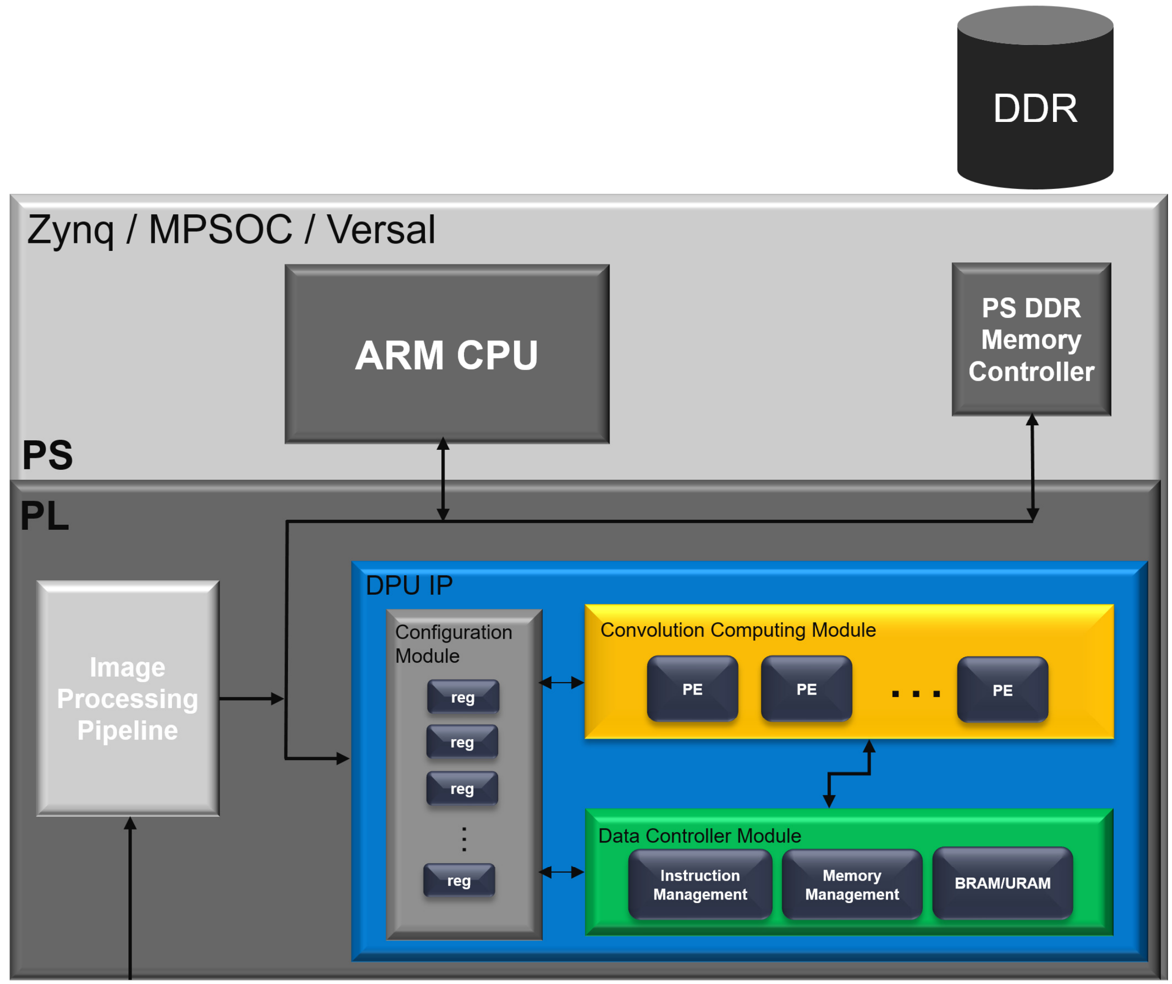
XILINX

# DPU Data Flow

ARM performs post processing
on output activations




```
top[0] prob = 0.999485  name = Sussex spaniel
top[1] prob = 0.000335  name = cocker spaniel, English cocker spaniel, cocker
top[2] prob = 0.000123  name = Irish setter, red setter
top[3] prob = 0.000045  name = clumber, clumber spaniel
top[4] prob = 0.000005  name = Irish water spaniel
```




*Slide and animation credit – Clayton Cameron and family*

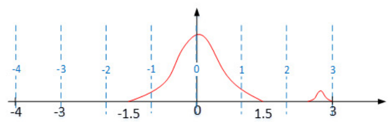
XILINX.

# DNNDK Highlights

# Quantization

## Quantization Strategy

- **Our Quantization Strategy**
  - Uniform Symmetric Quantization → 8Bit for Our DPU
  - Scale = 2^N

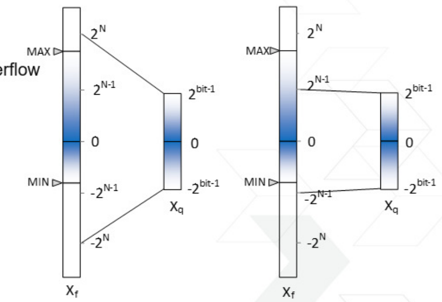



c) Uniform Symmetric Quantization

- **Advantages**
  - Hardware friendly
  - High efficiency: all fix-point calculation
  - Make use of redundancy of CNN models(especially with BatchNorm Layers)

## Two quantization methods

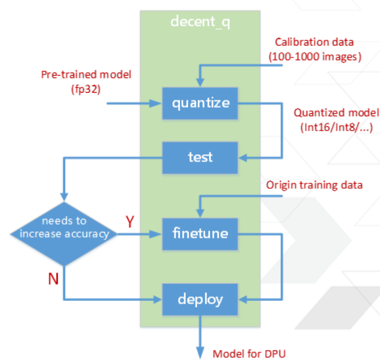
- **Non-Overflow Method:**
  - Choose quantize pos -> all values does not overflow
  - No saturation
  - Sensitive to large values
- **Min-Diff Method:**
  - Pos = Minimize$\sum(X_{qi}-X_{fi})^2$
  - Need saturated truncation




A) Non-Overflow Method    B) Min-Diff Method

## Quantization Tool – decent_q

- **4 steps in decent_q**
  - quantize – quantize network
  - test – test network accuracy/mAP
  - finetune – finetune quantized network
  - deploy – generate model for DPU
- **Data**
  - Calibration data – quantize activation
  - Training data – further increase accuracy






XILINX.

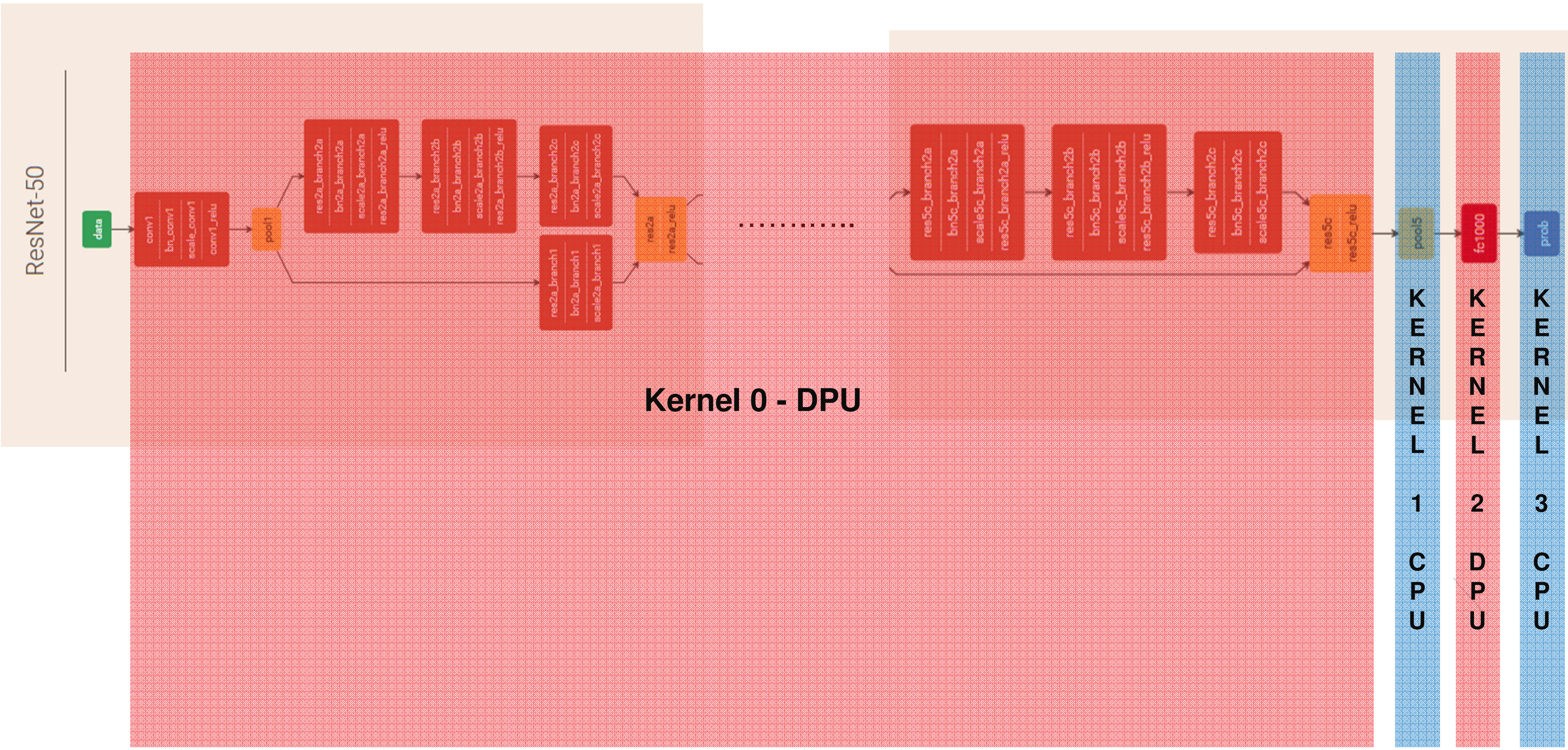
# Compilation – ResNet50 Example

```
[DNNC][Warning] Only max pooling is supported, but [pool5] layer has average pooling type.
[DNNC][Warning] layer [pool5] is not supported in DPU, deploy it in CPU instead.
[DNNC][Warning] layer [prob] is not supported in DPU, deploy it in CPU instead.
[DNNC][Warning] Fail to convert gv file to jpg because 'dot' is not installed in current system. Try to install it using 'sudo apt-get instal
l graphviz'. The original gv file is saved in 'resnet50_kernel_graph.gv'.

DNNC Kernel Information

1. Overview
kernel numbers  : 4
kernel topology : resnet50_kernel_graph.jpg

2. Kernel Description in Detail
kernel id       : 0
kernel name     : resnet50_0
type            : DPUKernel
nodes           : NA
input node(s)   : conv1(0)
output node(s)  : res5c_branch2c(0)

kernel id       : 1
kernel name     : resnet50_1
type            : CPUKernel
nodes           : NA
input node(s)   : pool5
output node(s)  : pool5

kernel id       : 2
kernel name     : resnet50_2
type            : DPUKernel
nodes           : NA
input node(s)   : fc1000(0)
output node(s)  : fc1000(0)

kernel id       : 3
kernel name     : resnet50_3
type            : CPUKernel
nodes           : NA
input node(s)   : prob
output node(s)  : prob
```



XILINX.

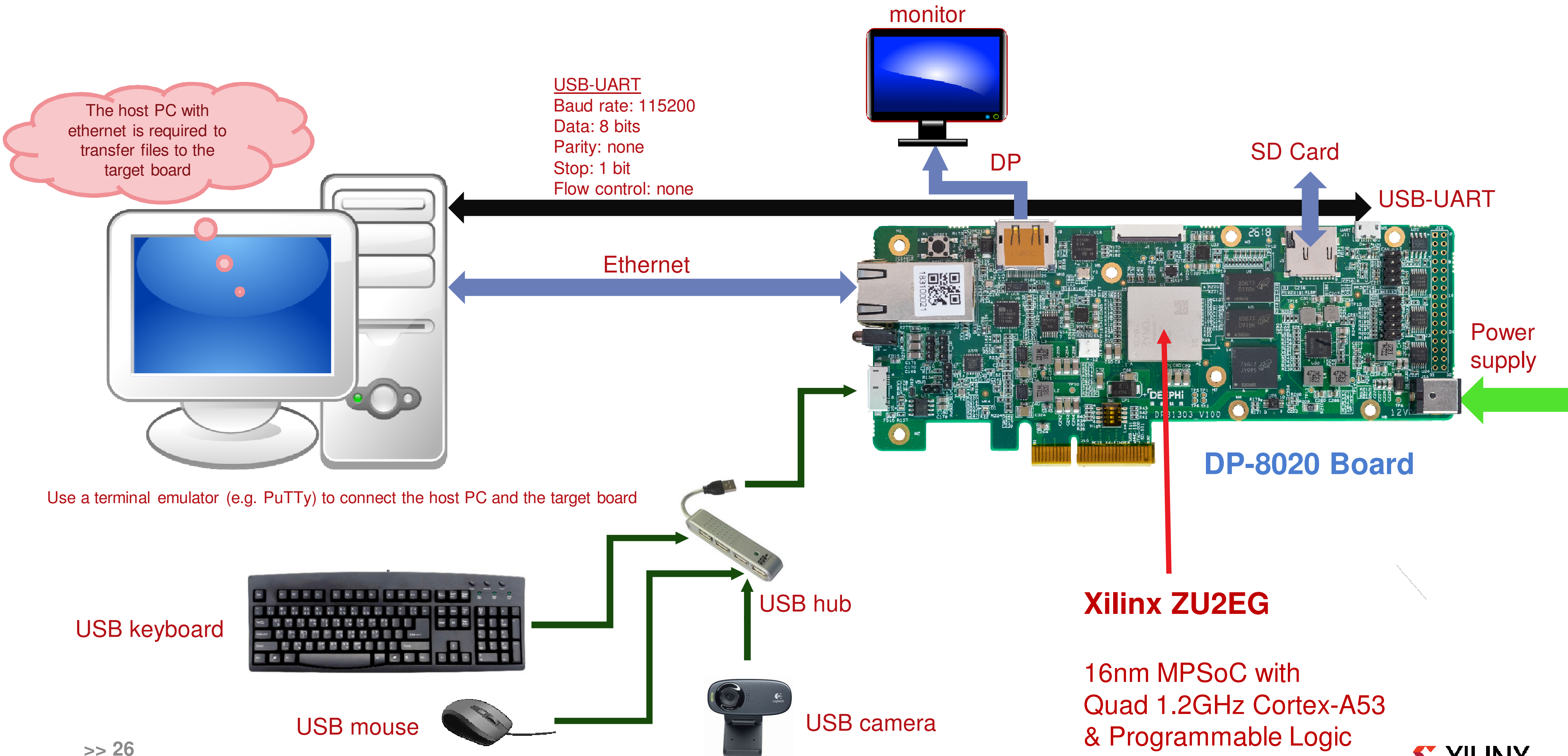
# B4096 ResNet Deployment in DNNDK v2.08






XILINX.

# Typical Evaluation / Development Environment

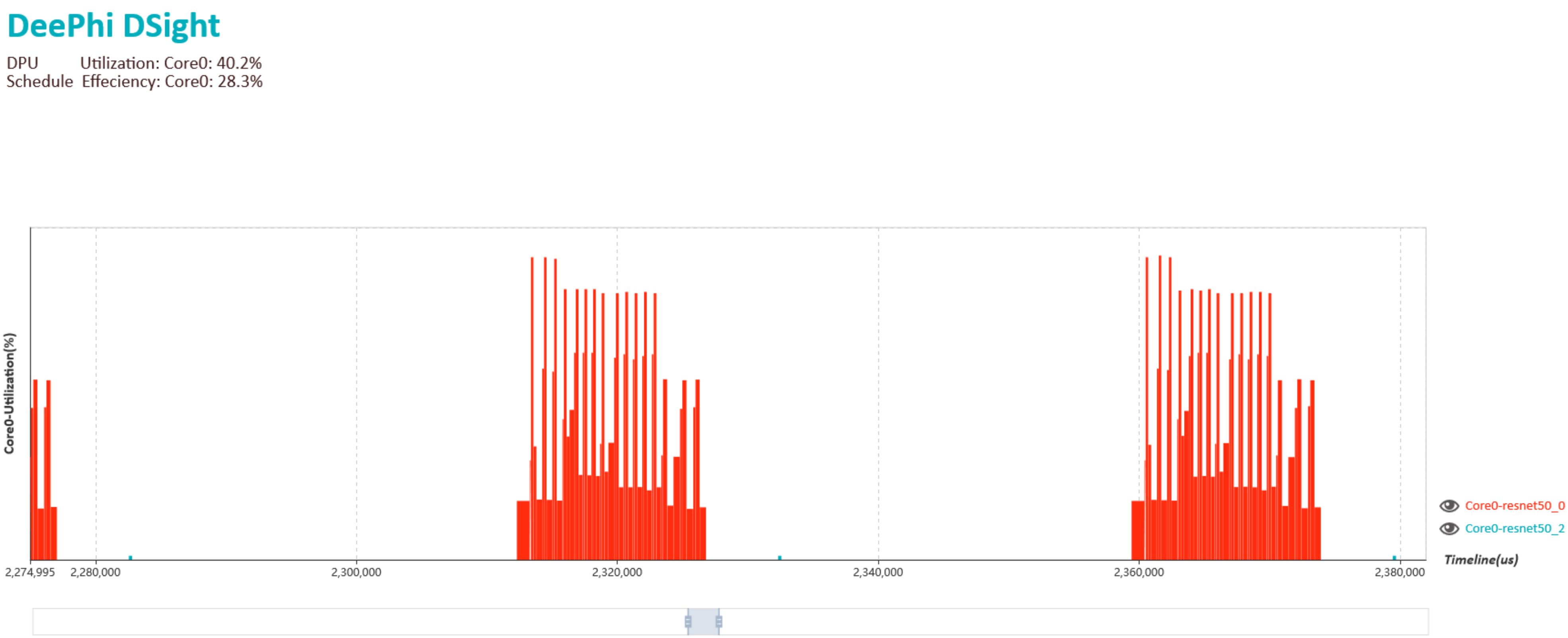





XILINX.

# Live Demo

# DSight Profiler






XILINX.

# What have we accomplished

> Demonstrated DECENT model quantization flow
> Demonstrated DNNC model compilation flow
> Demonstrated ResNet50 model deployment on the ZCU102
> Demonstrated Dsight profiling flow

# Resources

## Edge AI Resources

The following resources are available to help you start developing with the Edge AI Platform.

### Edge AI Tools

| Product | Documentation | Tool Download | File Size | MD5 Checksum |
|---|---|---|---|---|
| DNNDK | DNNDK User Guide (UG1327) | xlnx_dnndk_v2.08_1902.tar.gz | 1007 MB | cf4dade1b3af14437ae97c09691ba381 |
| DNNDK for SDSoC | DNNDK User Guide for SDSoC (UG1331) | xilinx_dnndk_v2.08_for_sdsoc_190214.tar.gz | 667 MB | 7f165aff5062497e4bb69b70773c49b1 |

### Edge AI Evaluation Boards

| Product | Documentation | Image Download | File Size | MD5 Checksum |
|---|---|---|---|---|
| ZCU102 Kit | ZCU102 User Guide (UG1182) | 2018-12-04-zcu102-desktop-stretch.img.zip | 571 MB | d0d5faf8ece80b96f5591d09756d5a5d |
| ZCU104 Kit | ZCU104 User Guide (UG1267) | 2018-12-04-zcu104-desktop-stretch.img.zip | 571 MB | ada2420c4afbd89efdeea741e0917e26 |
| Avnet Ultra 96 | Ultra 96 User Guide | xilinx-ultra96-desktop-stretch-2018-12-10.img.zip | 566 MB | c5d2422063213b4bc4c18a3223c6adc8 |

### Edge AI Targeted Reference Designs (TRD)

| Product | Image Download & Docs | File Size | MD5 Checksum |
|---|---|---|---|
| DPU TRD | zcu102-dpu-trd-2018-2-1903.zip | 459 MB | 872170d1038d0c824cb2c808743930e4 |

### Platform Downloads

| Product | Download | File Size | MD5 Checksum |
|---|---|---|---|
| ZCU102 SDSoC 2018.3 Platform for DNNDK | zcu102-rv-ss-2018-3-dnndk.tar.gz | 1.3 GB | 7102c6942eb65d8b9d258914f69c6eaa |
| ZCU104 SDSoC 2018.3 Platform for DNNDK | zcu104-rv-ss-2018-3-dnndk.tar.gz | 1.3 GB | d5bc80aa8135a719e273e2ff6ca85762 |

https://www.xilinx.com/products/design-tools/ai-inference/ai-developer-hub.html#edge https://forums.xilinx.com/t5/Deephi-DNNDK/bd-p/Deephi

Community Forums › Forums › Applications › Deephi DNNDK

## Announcements

Welcome to the Deephi DNNDK Community Forum. This community should serve as a resource to ask and learn about using Deephi DNNDK on all supported platforms, new feature announcements and troubleshooting AI applications.

Most Recent Threads
Before you post, please read our Community Forums Guidelines or to get started see our Community Forum Help.

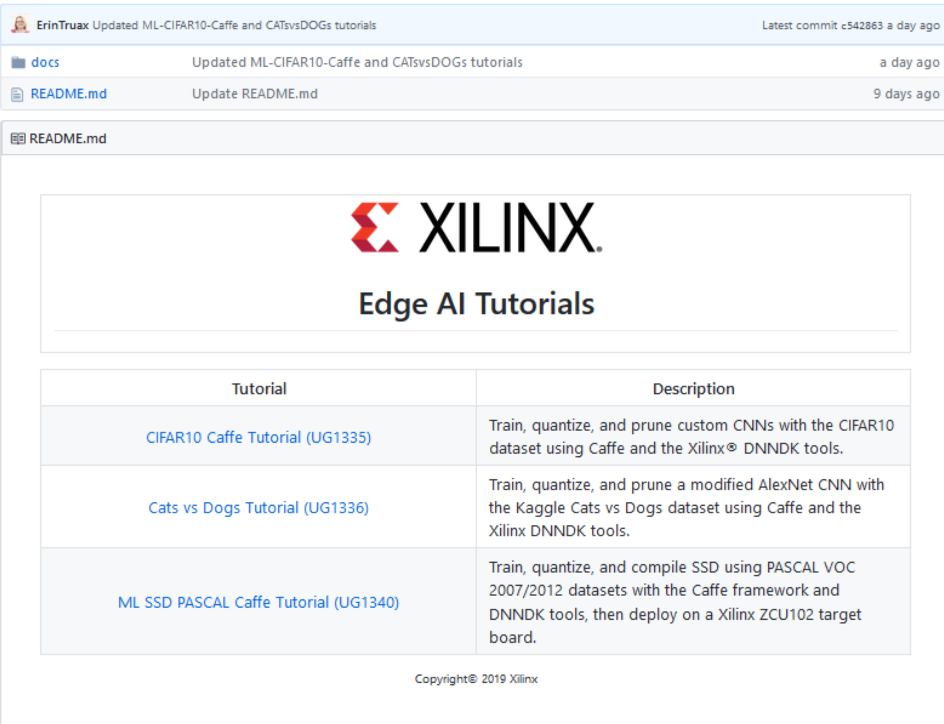
## Discussions

Post a Question

- Where is correct img file for ZCU104
  by tacbook on 03-06-2019 02:28 AM • Latest post on 03-07-2019 01:41 PM by qhall — 0 likes, 3 replies
- DPU Targeted Reference Design Released to Xilinx.c...
  by qhall on 03-05-2019 03:24 PM — 2 likes, 0 replies
- zcu104 boot
  by @xx on 03-05-2019 05:43 AM • Latest post on 03-05-2019 09:21 AM by meherp — 0 likes, 2 replies
- os image zcu104
  by @xx on 03-04-2019 11:27 PM • Latest post on 03-05-2019 09:29 AM by meherp — 0 likes, 2 replies
- Monitor flickers while runing dnndk example
  by deepg799 on 03-04-2019 08:16 PM • Latest post on 03-07-2019 01:16 PM by terryo — 0 likes, 3 replies
- **Is there simple tutorial for K7 Custom board?**
  by trustfarm on 02-28-2019 11:38 PM • Latest post on 03-03-2019 11:44 PM by trustfarm — 0 likes, 2 replies

https://forums.xilinx.com/t5/Deephi-DNNDK/bd-p/Deephi

# Resources



ErinTruax Updated ML-CIFAR10-Caffe and CATsvsDOGs tutorials — Latest commit c542863 a day ago

| | | |
|---|---|---|
| docs | Updated ML-CIFAR10-Caffe and CATsvsDOGs tutorials | a day ago |
| README.md | Update README.md | 9 days ago |

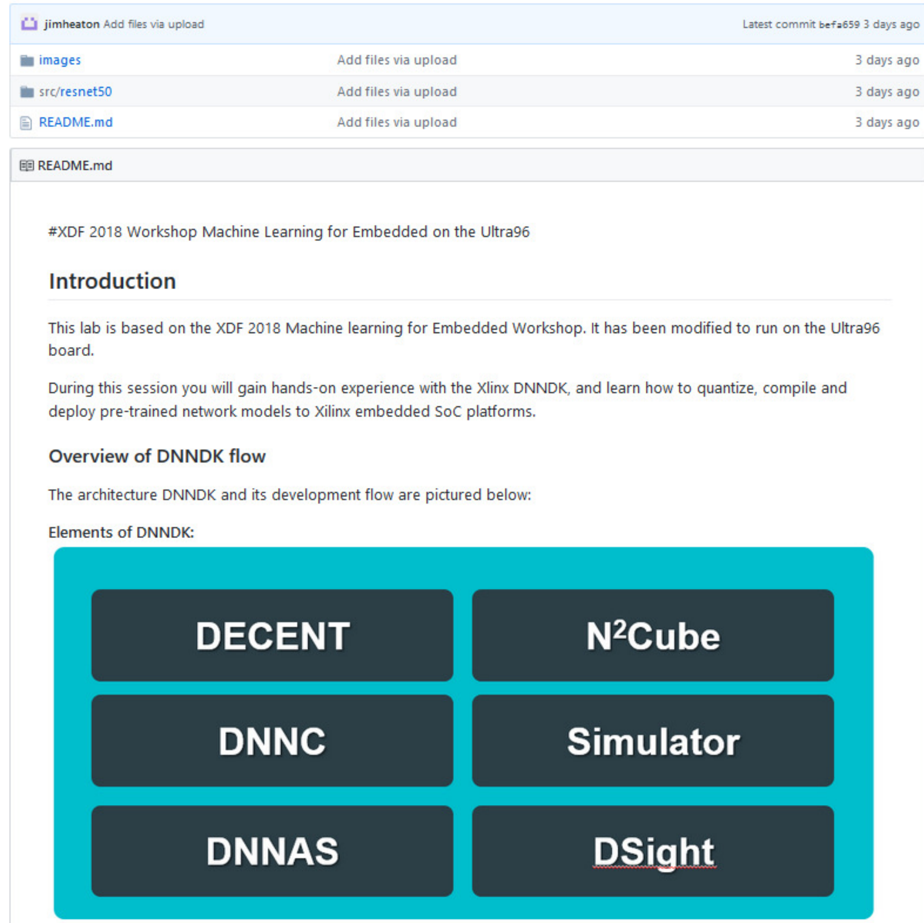
README.md

XILINX

## Edge AI Tutorials

| Tutorial | Description |
|---|---|
| CIFAR10 Caffe Tutorial (UG1335) | Train, quantize, and prune custom CNNs with the CIFAR10 dataset using Caffe and the Xilinx® DNNDK tools. |
| Cats vs Dogs Tutorial (UG1336) | Train, quantize, and prune a modified AlexNet CNN with the Kaggle Cats vs Dogs dataset using Caffe and the Xilinx DNNDK tools. |
| ML SSD PASCAL Caffe Tutorial (UG1340) | Train, quantize, and compile SSD using PASCAL VOC 2007/2012 datasets with the Caffe framework and DNNDK tools, then deploy on a Xilinx ZCU102 target board. |





jimheaton Add files via upload — Latest commit befa659 3 days ago

| | | |
|---|---|---|
| images | Add files via upload | 3 days ago |
| src/resnet50 | Add files via upload | 3 days ago |
| README.md | Add files via upload | 3 days ago |

README.md

#XDF 2018 Workshop Machine Learning for Embedded on the Ultra96

## Introduction

This lab is based on the XDF 2018 Machine learning for Embedded Workshop. It has been modified to run on the Ultra96 board.

During this session you will gain hands-on experience with the Xlinx DNNDK, and learn how to quantize, compile and deploy pre-trained network models to Xilinx embedded SoC platforms.

### Overview of DNNDK flow

The architecture DNNDK and its development flow are pictured below:

Elements of DNNDK:



https://github.com/Xilinx/Edge-AI-Platform-Tutorials

https://github.com/jimheaton/Ultra96_ML_Embedded_Workshop

# Getting Started

1. Purchase a supported Xilinx evaluation board (eg ZCU102, ZCU104, Ultra96)
2. Configure a suitable build environment
3. Experience and modify Xilinx DNN examples
3. Evaluate quantization and compilation of Xilinx examples or custom models

# Key Takeaways

1. DNNDK is able to deploy pre-trained DNN models to Xilinx SoC easily & quickly without writing any RTL

2. DNNDK supports both local and AWS build environments

3. DNNDK supports deployment of DN models with no FPGA experience