Accelerating AI in Datacenters
Xilinx ML Suite
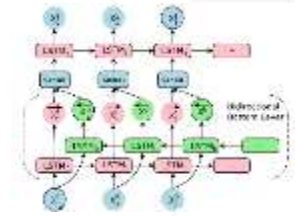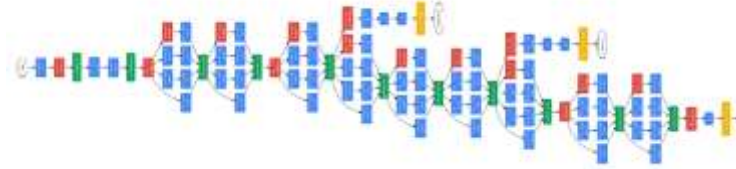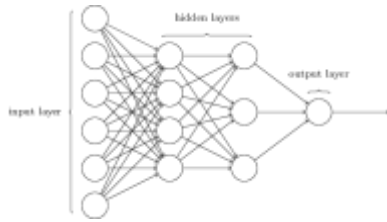
Rahul Nimaiyar
Director, Data Center IP Solutions

Kamran Khan
Product Marketing Manager, Lib and Software Acceleration

Oct 16th, 2018

≡ XILINX.

Deep Learning Models – A broad spectrum

Multi-Layer Perceptron

- Classification
- Universal Function Approximator
- Autoencoder

Convolutional Neural Network

- Feature Extraction
- Object Detection
- Image Segmentation
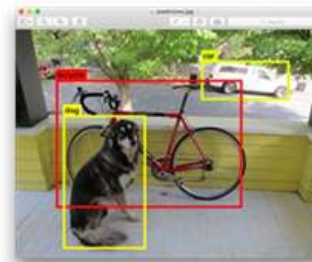
Recurrent Neural Network

- Sequence and Temporal Data
- Speech to Text
- Language Translation

Classification

"Dog"

Object Detection

Segmentation

XILINX

Xilinx ML Suite – DNN Processor + "Compiler, Quantizer, Runtime"

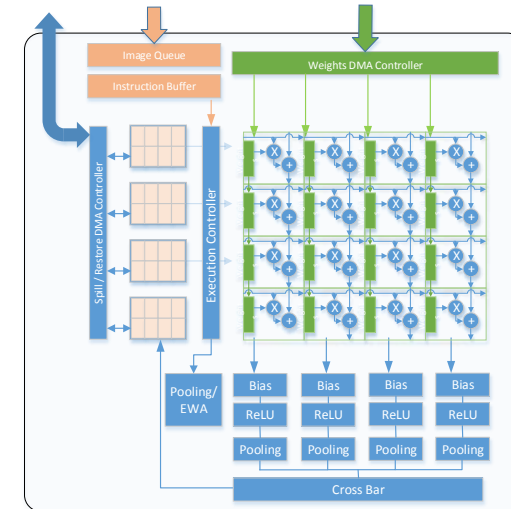**Trained Model**

Compiler + Runtime

"One Shot" Network Deployment
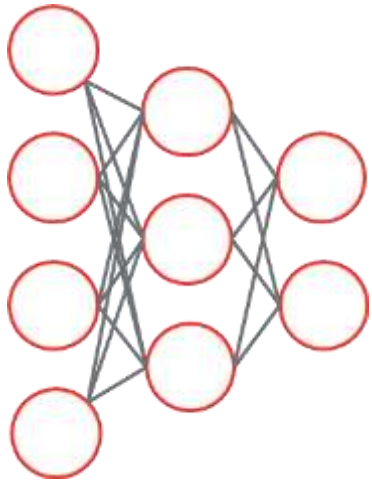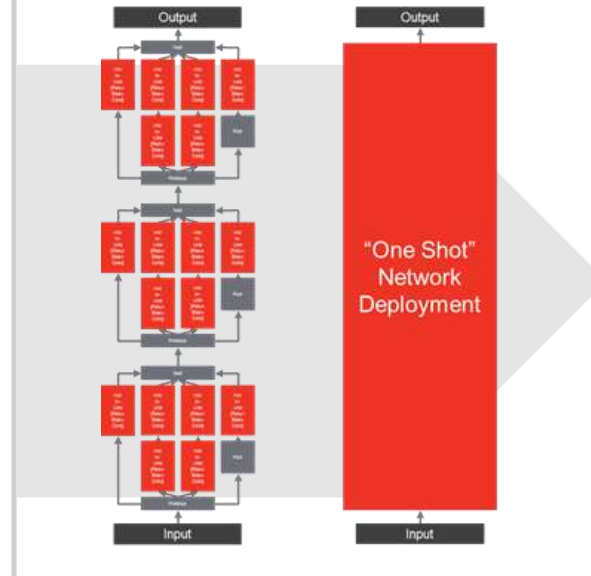
Xilinx DNN Processor

Low Latency, High Throughput – Batch 1

60-80% Efficiency

No FPGA Expertise Required

Xilinx DNN Processor (xDNN)
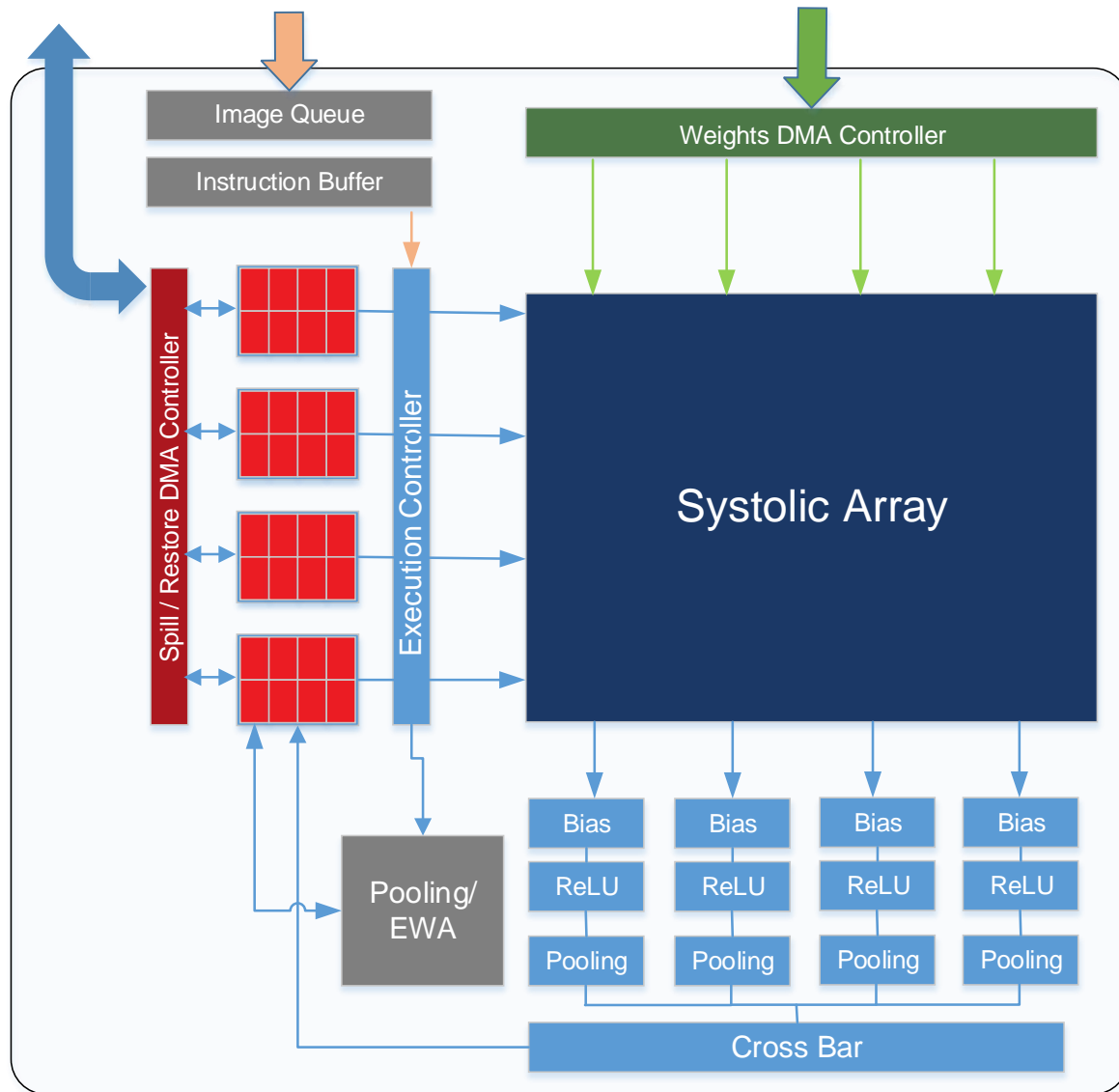
> Configurable Overlay Processor

> DNN Specific Instruction Set
  >> Convolution, Max Pool etc.

> Any Network, Any Image Size

> High Frequency & High Compute Efficiency

> Compile and run new networks

Image Queue

Instruction Buffer

Weights DMA Controller

Spill / Restore DMA Controller

Execution Controller

Systolic Array

Pooling/EWA

Bias | Bias | Bias | Bias

ReLU | ReLU | ReLU | ReLU

Pooling | Pooling | Pooling | Pooling

Cross Bar

XILINX.

Rapid Feature and Performance Improvement

xDNN-v1
Q4CY17

- Array of Accumulator
- Int16 (Batch=1) and Int8 (Batch=2) support
- Instructions: Convolution, ReLU, Pool, Elementwise
- Flexible kernel size(square) and strides
- 500 MHz

xDNN-v2
Q2CY18

- All xDNN-v1 Features
- DDR Caching: Larger Image size
- New Instructions: Depth-wise Convolution, De-convolution, Up-sampling
- Rectangular Kernels
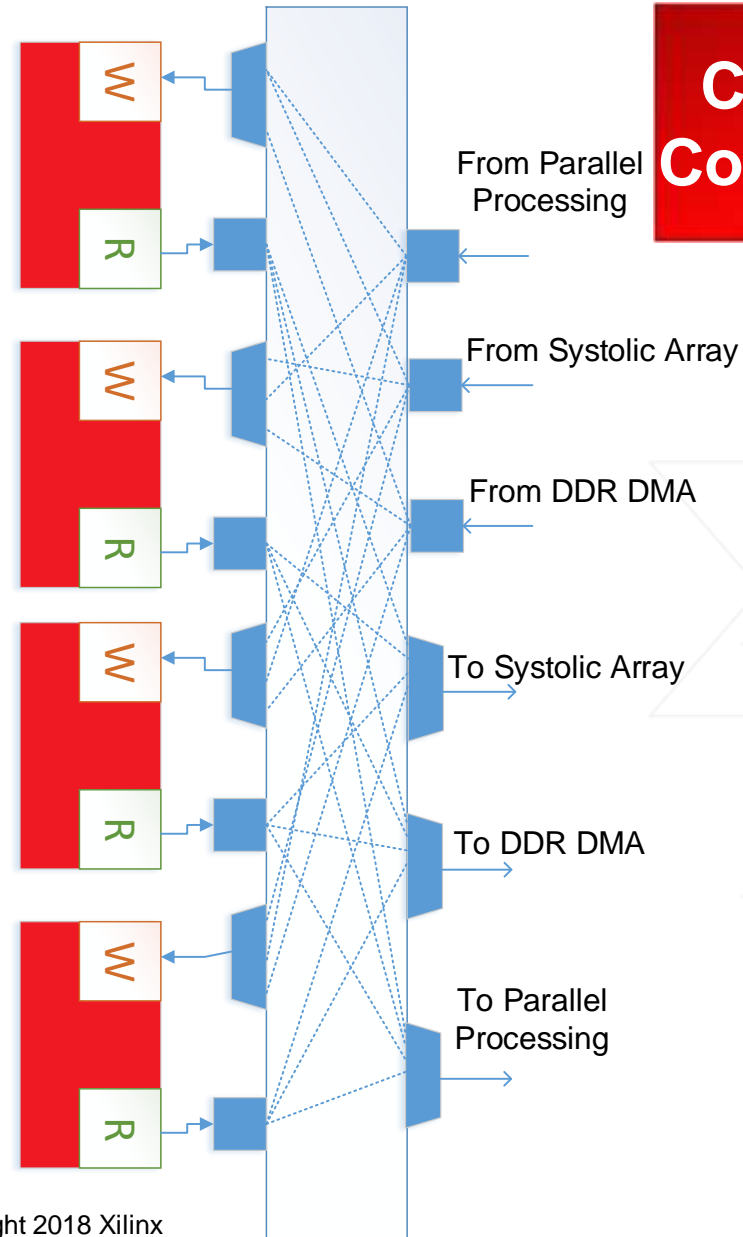- 500 MHz

xDNN-v3
Q4CY18

- New Systolic Array Implementation: 2.2x lower latency
- Instruction Level Parallelism – non-blocking data movement
- Batch=1 for Int8 – lower latency
- Feature compatible with xDNN-v2
- 720+ MHz

XILINX

xDNN – Channel Parallel Systolic Array
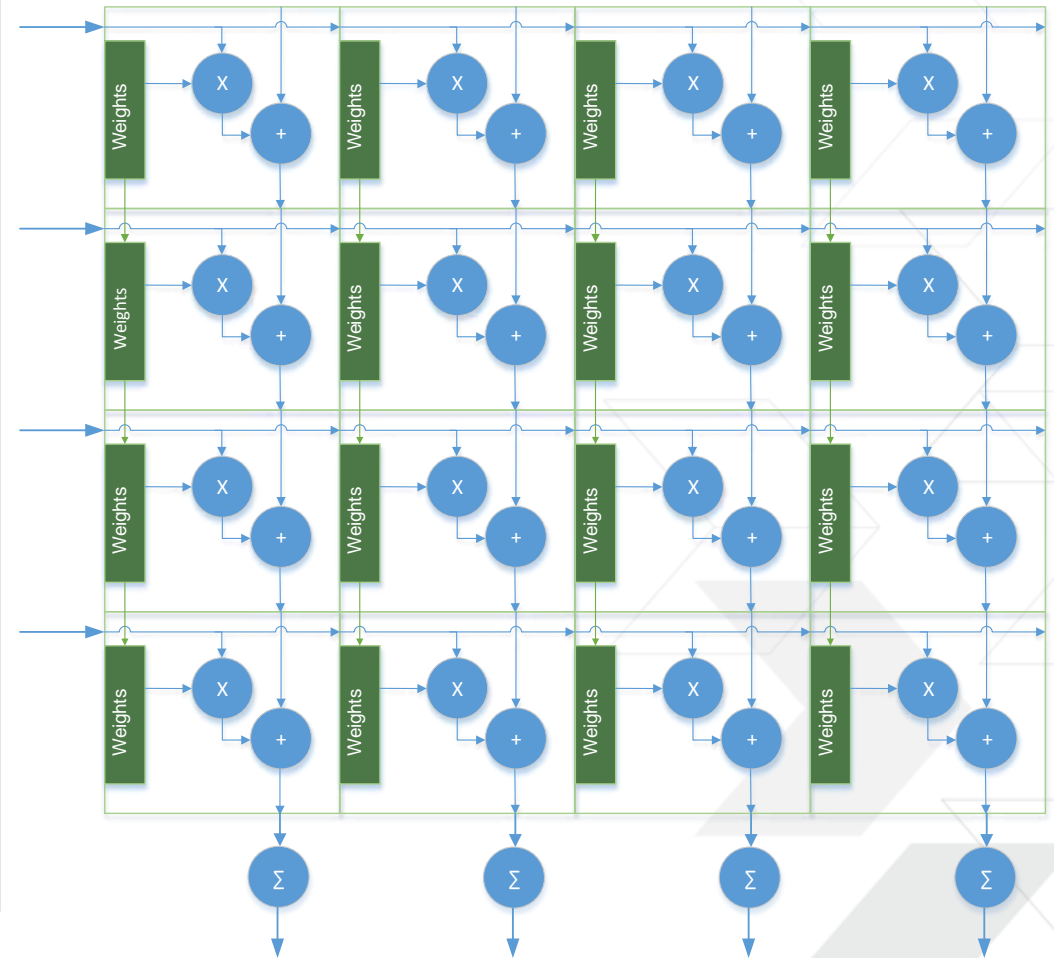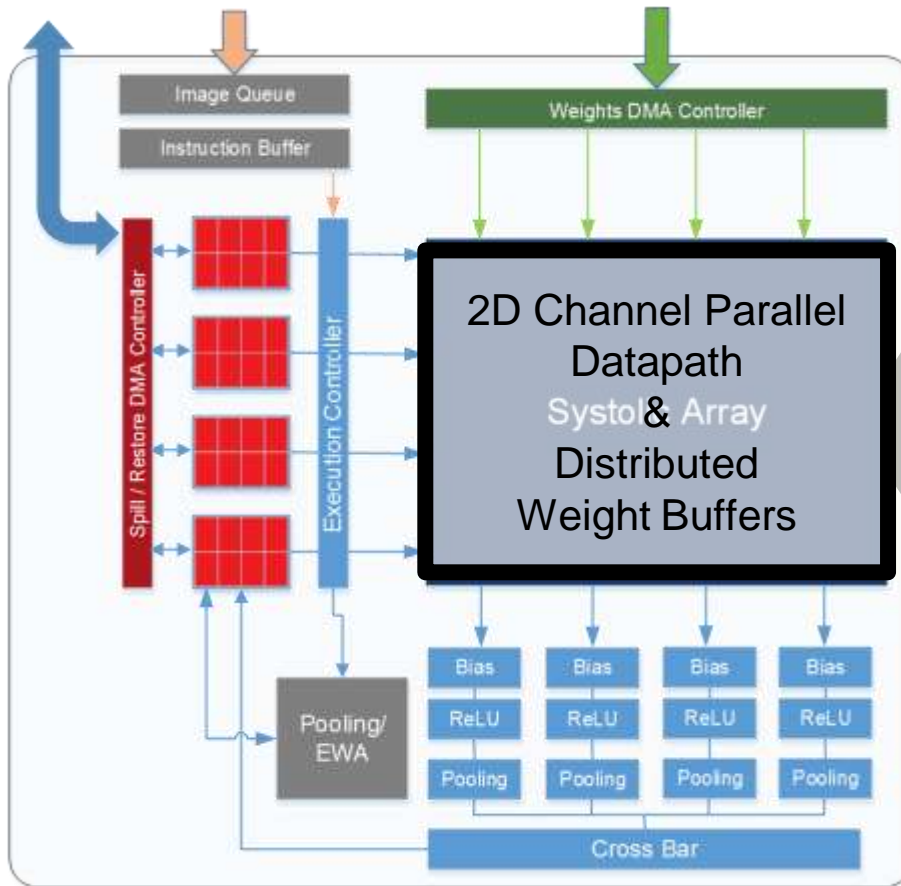
Micro-Architecture Optimized for underlying Ultrascale+ FPGA Fabric

XILINX

xDNN Processor – Tensor Memory
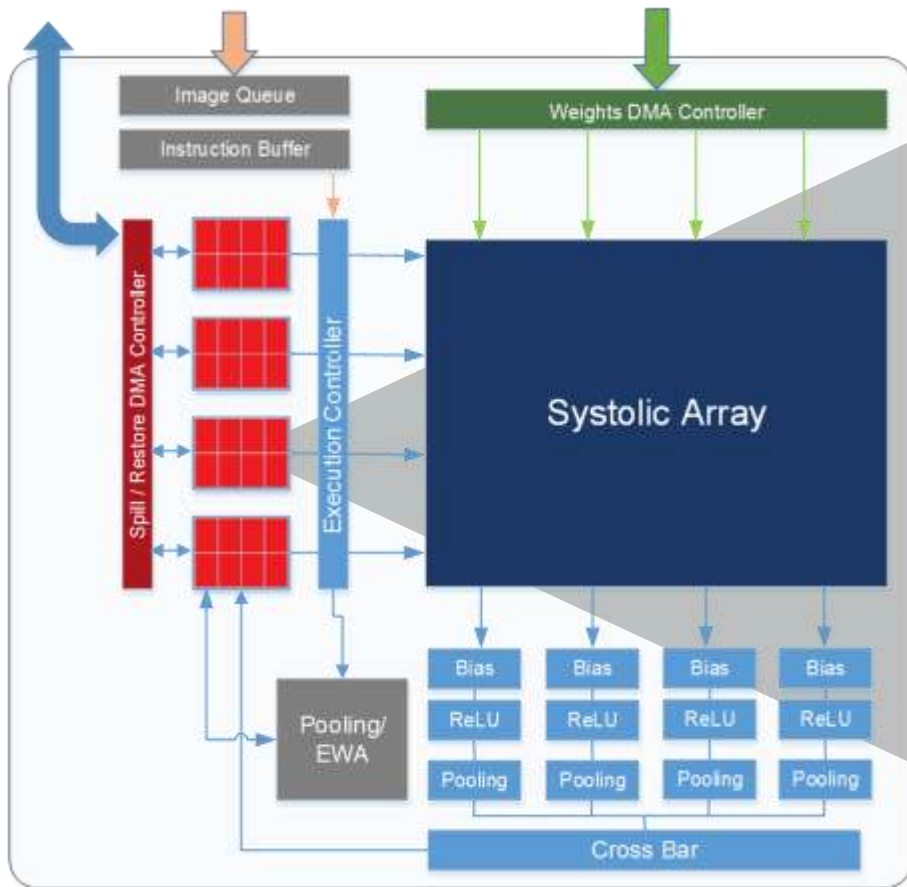
Channel Parallel Concurrent Access

From Parallel Processing

From Systolic Array

From DDR DMA

To Systolic Array

To DDR DMA

To Parallel Processing

© Copyright 2018 Xilinx

 XILINX

xDNN Compiler + Runtime

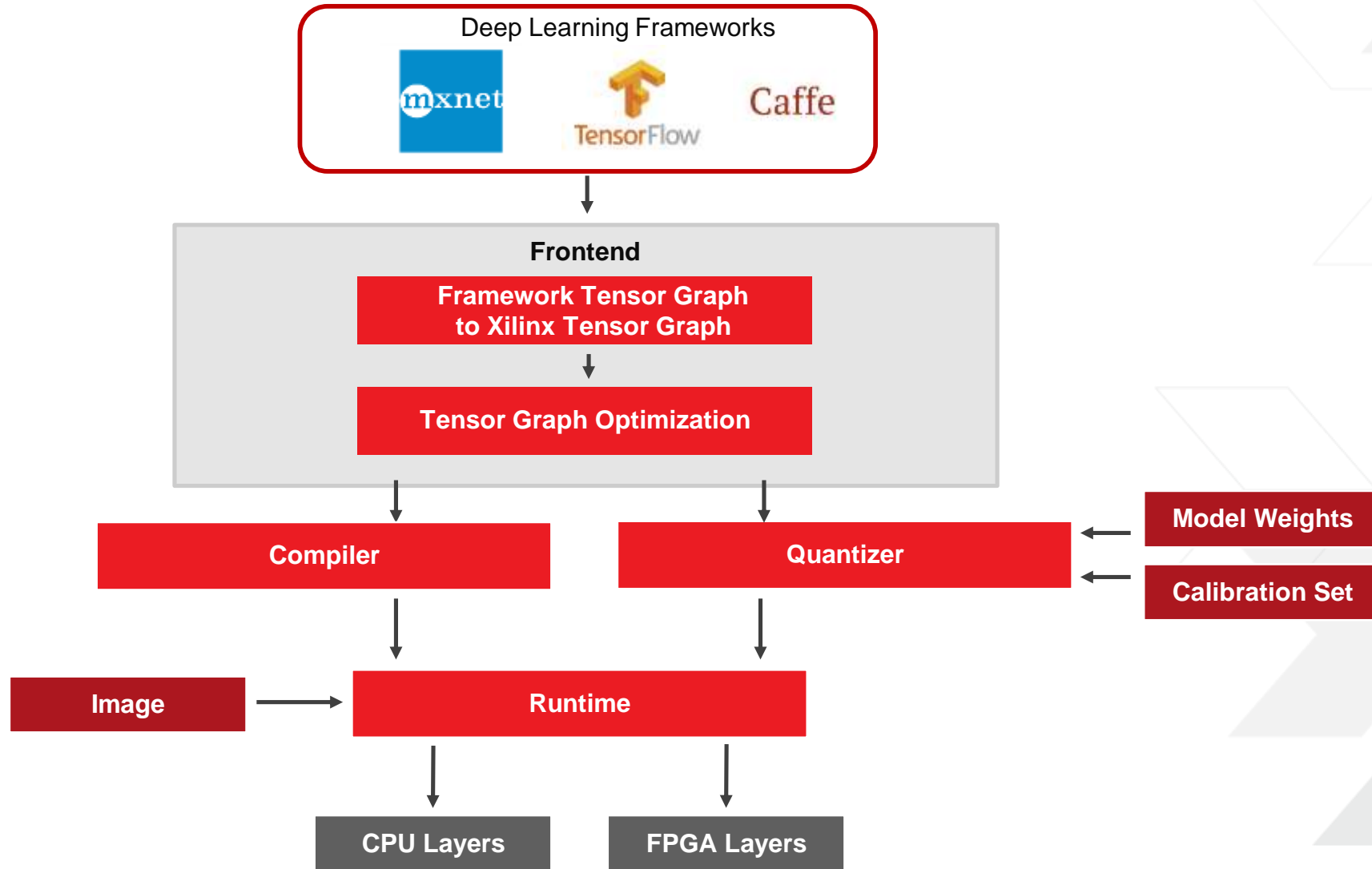*https://github.com/Xilinx/ml-suite*

© Copyright 2018 Xilinx

Graph Partitioning

> One time loading of sub-graph instructions

> Data Flow Execution

FPGA or CPU **FPGA** **CPU** **FPGA** **CPU**

Pre-Processing   **Subgraph 1**   **Parallel Subgraphs**   **Post-Processing**
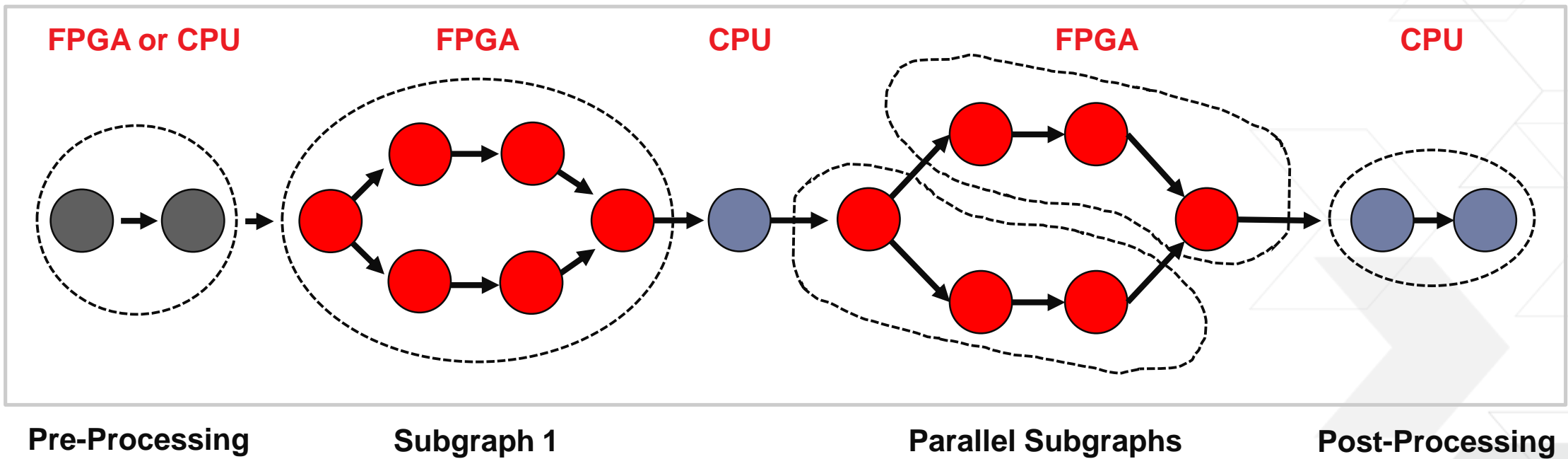
1 Core -> Multi-Core -> Multi-Chip

XILINX

Fusing of Operations
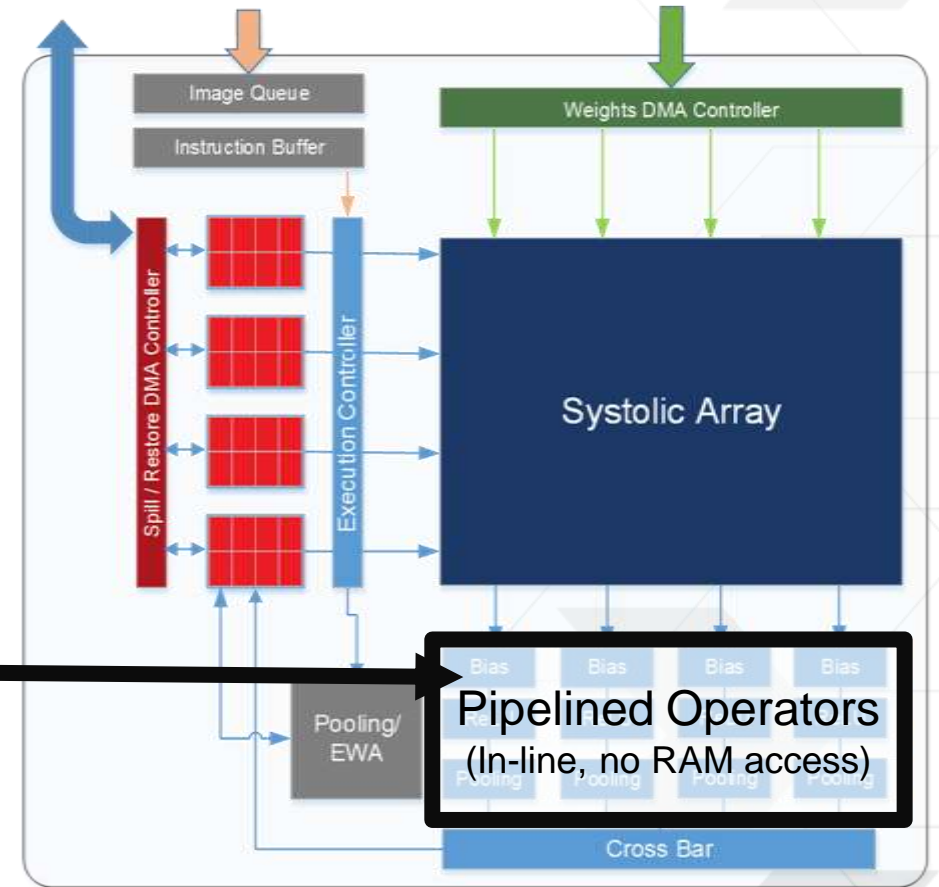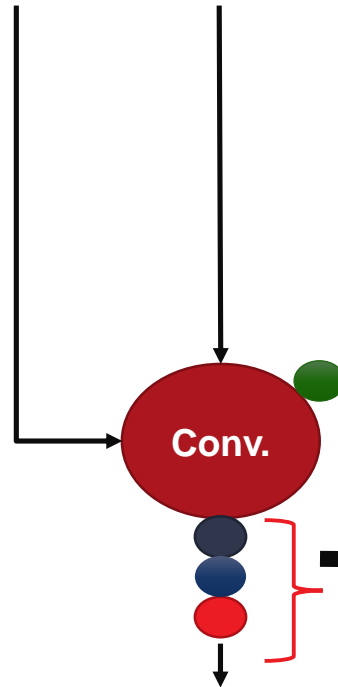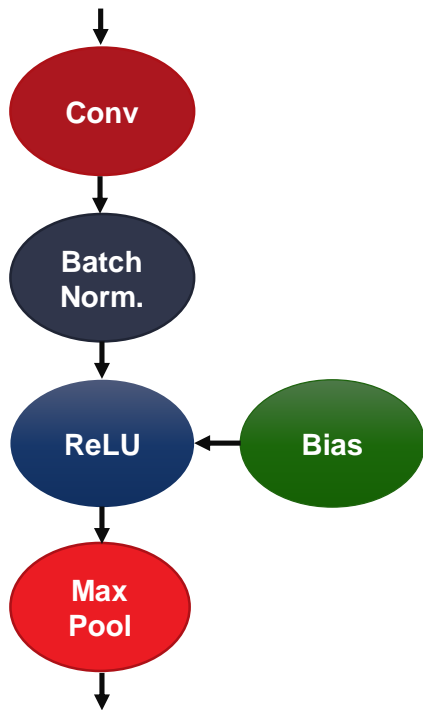
> Fuse operations as pipe-lined stages

> Access activations only once

Instruction Level Parallelism
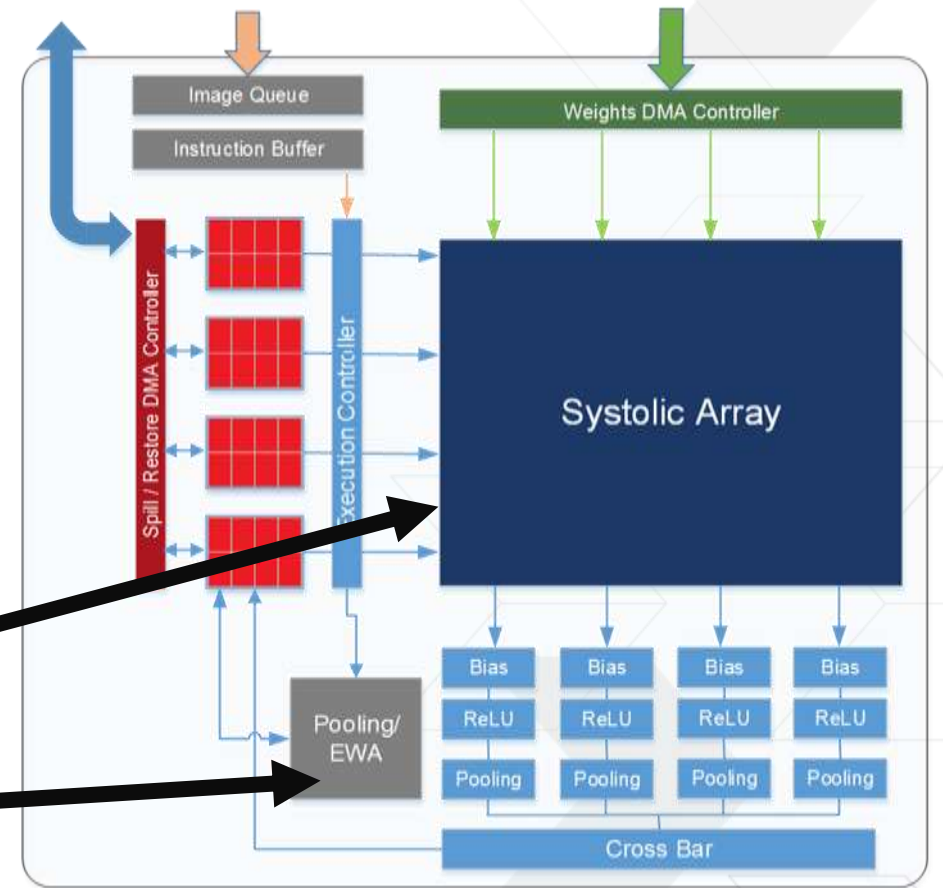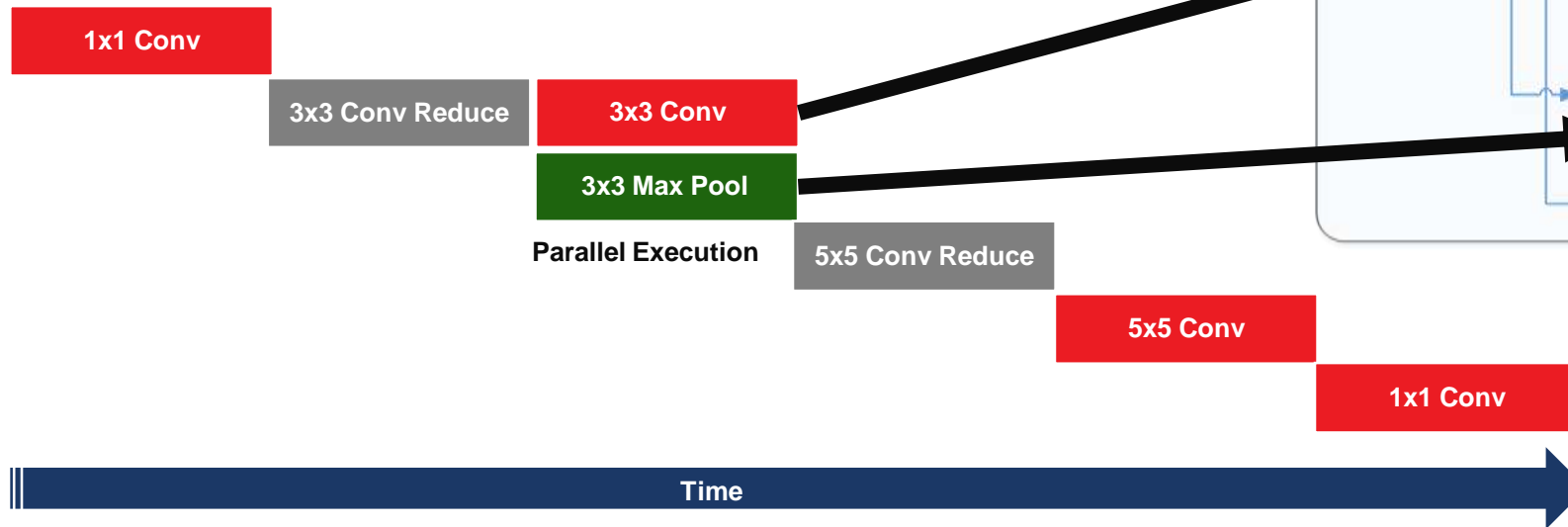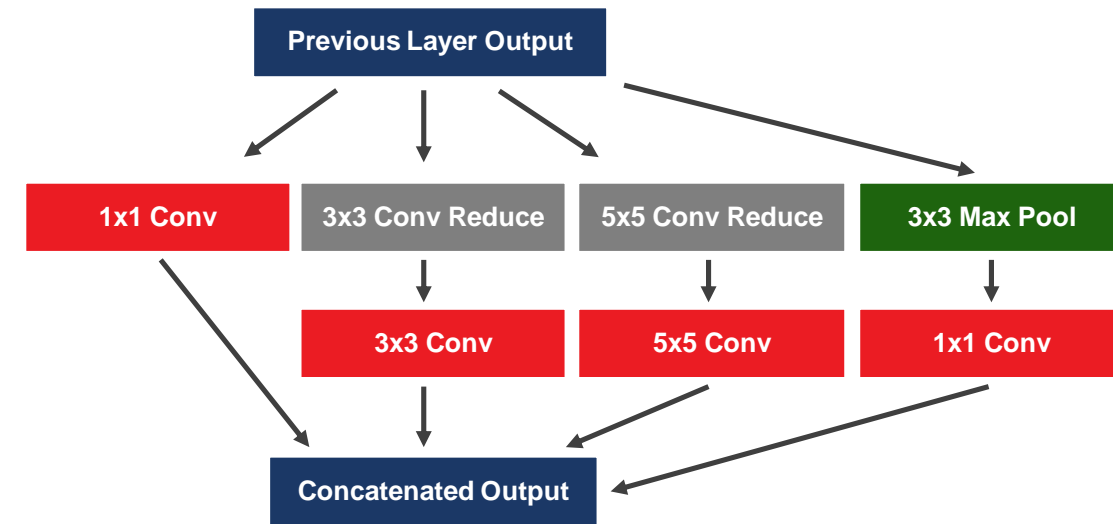
Automatic Intra Layer Tiling

Any Feature Map Size

> Tile when Feature Map size exceeds on-chip memory

> Work on full feature map depth

XILINX
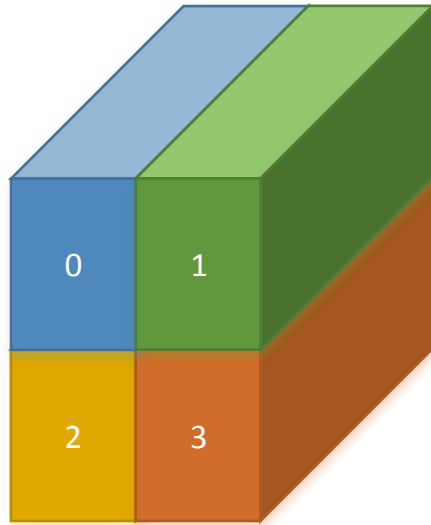
xfDNN Runtime Engine

> ML specific engine built on top of SDx runtime

> Lightweight and portable with no dependency on ML frameworks

> Extensive C++/Python API with simplified use model

> Asynchronous XDNN execution

> Streaming/pipeline/Dataflow support for large imageset

> Multiple CNN models running on single FPGA

> Multiple FPGA support

ΣXILINX

Simple Usage

```python
import xdnn, xdnn_io

args = xdnn_io.processCommandLine()
xdnn.createHandle(args['xclbin'], "kernelSxdnn_0", args['xlnxlib'])

(weightsBlob, fcWeight, fcBias) = xdnn_io.loadWeights(args)

(fpgaInputs, batch_sz) = xdnn_io.prepareInput(args)
fpgaOutput = xdnn_io.prepareOutput(args['fpgaoutsz'], batch_sz)

xdnn.execute(args['netcfg'],
  weightsBlob, fpgaInputs, fpgaOutput,
  batch_sz, args['quantizecfg'])

fcOut = xdnn.computeFC(fcWeight, fcBias, fpgaOutput)

softmaxOut = xdnn.computeSoftmax(fcOut)

xdnn_io.printClassification(softmaxOut, args);

xdnn.closeHandle()
```

Blocking

```python
# exec_async() enqueues FPGA task and returns immediately
xdnn.exec_async(args['netcfg'],
  weightsBlob, fpgaInputs, fpgaOutput,
  batch_sz, args['quantizecfg'], args['PE'])

# get_result() blocks for FPGA result
xdnn.get_result(args['PE'])
```

```python
numFPGAs = 8
args = xdnn_io.processCommandLine()
xdnn.createHandle(args['xclbin'], "kernelSxdnn_0", args['xlnxlib'], numFPGAs)

(weightsBlob, fcWeight, fcBias) = xdnn_io.loadWeights(args)
```

8-FPGA

Non-Blocking

XILINX

https://github.com/Xilinx/ml-suite

Xilinx ML Suite

| Tag: 1.1 ▾ | New pull request | | Find file | Clone or download ▾ |

kamranjk Update README.md		Latest commit 233b9ad 27 days ago
📁 apps	Update README.md	a month ago
📁 docs	Update README.md	27 days ago
📁 examples	Update README.md	28 days ago
📁 ext	Release 1.1 to master	a month ago
📁 gemx	Doc updates, Notebook bug fix, Binary upgrade	a month ago
📁 models	Delete yolo-xdnn-tend-20180206-bnremove.weights	28 days ago
📁 notebooks	Doc updates, Notebook bug fix, Binary upgrade	a month ago
📁 overlaybins	Add support for perpetual demo	a month ago
📁 tests/gemx	Release 1.1 to master	a month ago
📁 xfdnn	Add support for perpetual demo	a month ago

Server Platforms
Intel x86, AMD Epyc, Power9, ARM

FaaS
aws marketplace   NIMBIX   Alibaba Cloud aliyun.com   HUAWEI *
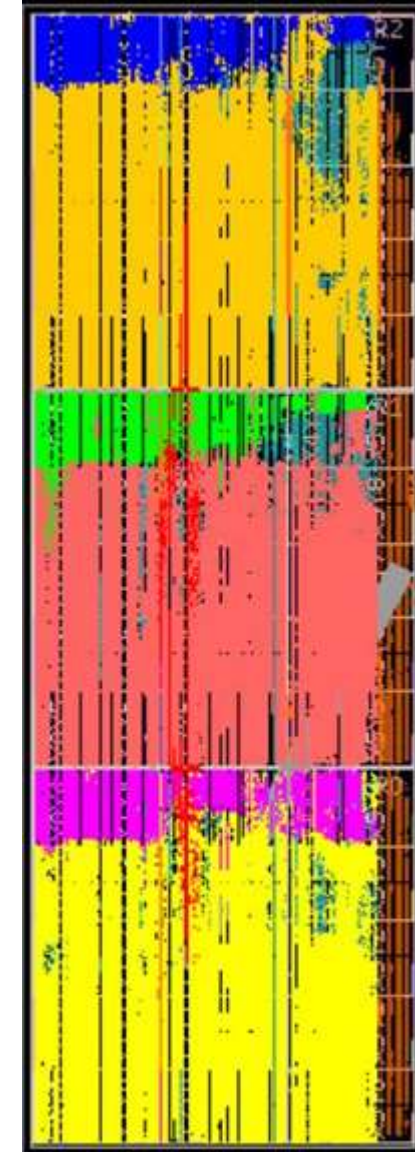
Xilinx Boards

*Under Development

xDNN v3 Implementation on Alveo U200

> 3 Large 96x16 PEs– 1 in each SLR – 5.2 ML Shell

> Kernels @ 720 MHz/360MHz

Resource	Count	Utilization
LUTs	658k	52%
DSPs	5661	80%
BRAM	1258	58%
URAM	864	92%

XILINX

xDNN v3 Implementation on Alveo U250

> 4 Large 96x16 PEs– 1 in each SLR – standard 5.2 Shell
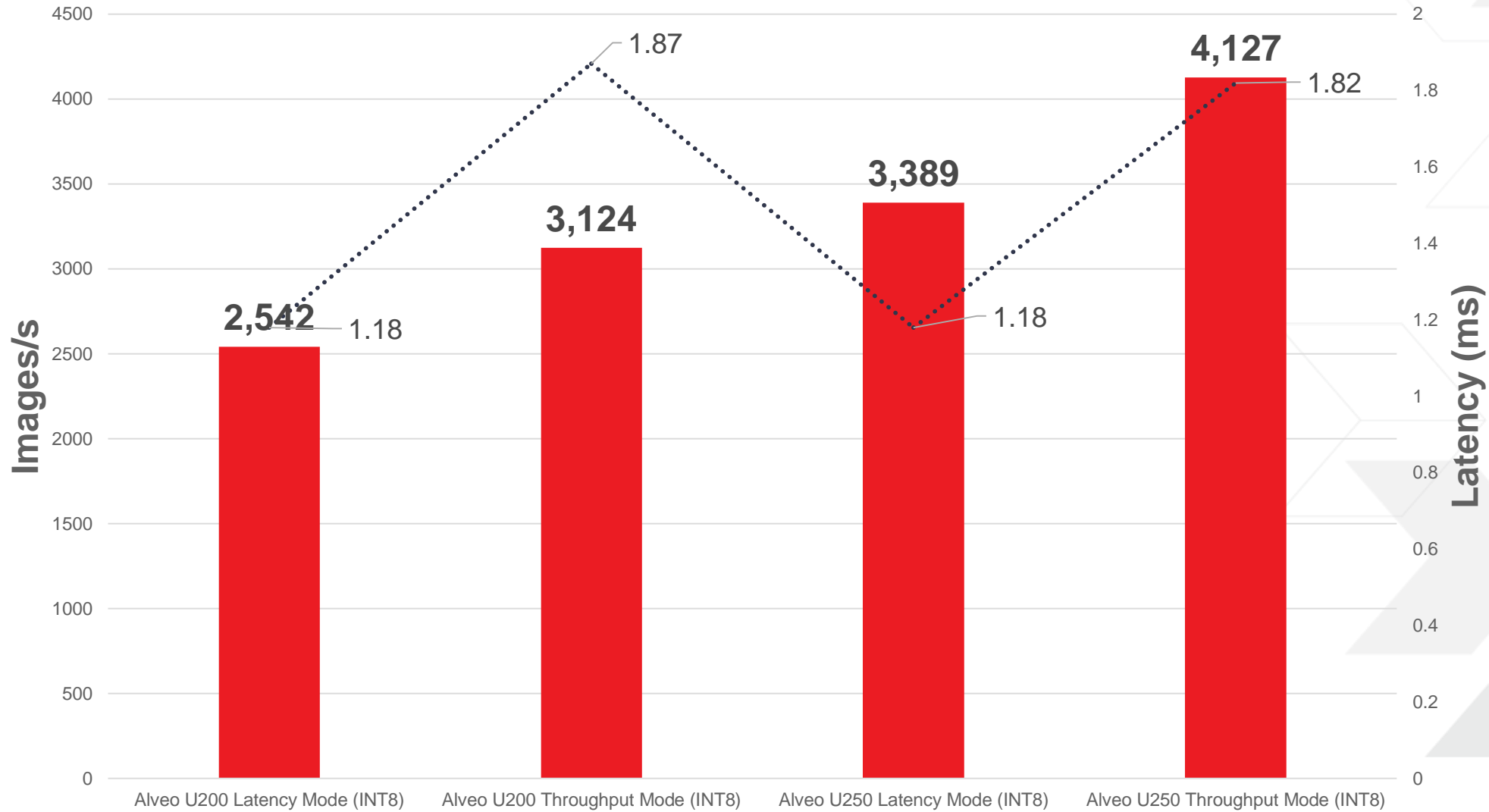
> Kernels at 700 MHz/350 MHz

Resource	Count	Utilization
LUTs	876k	51%
DSPs	7548	62%
BRAM	1632	61%
URAM	1152	90%

© Copyright 2018 Xilinx

XILINX
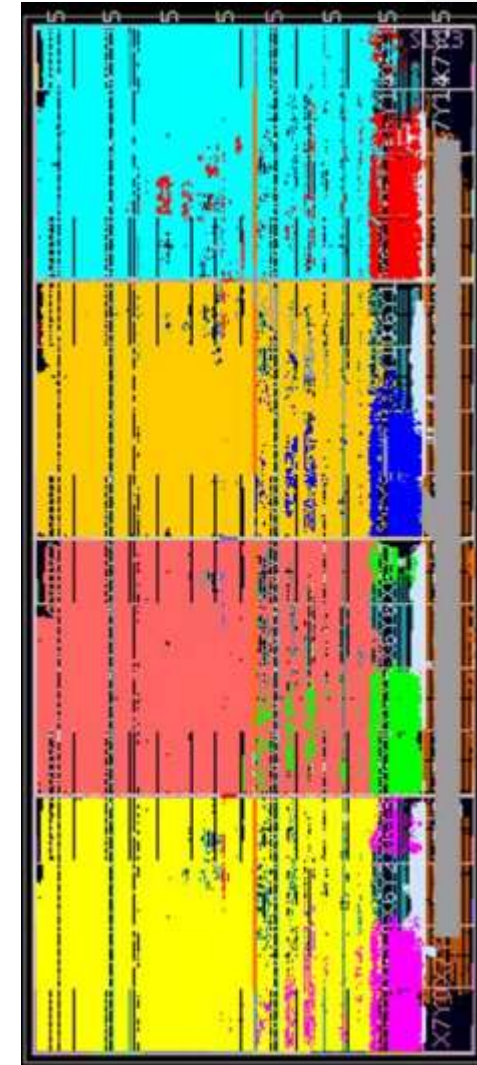
xDNN GoogLeNet v1 Performance – Image Size 224x224

© Copyright 2018 Xilinx

XILINX

xDNN GoogLeNet v1 Energy Efficiency

Bar chart showing Images/s/w energy efficiency comparison:
- Alveo U200 Throughput Mode (INT8): 35.5
- Alveo U250 Throughput Mode (INT8): 37.5

XILINX
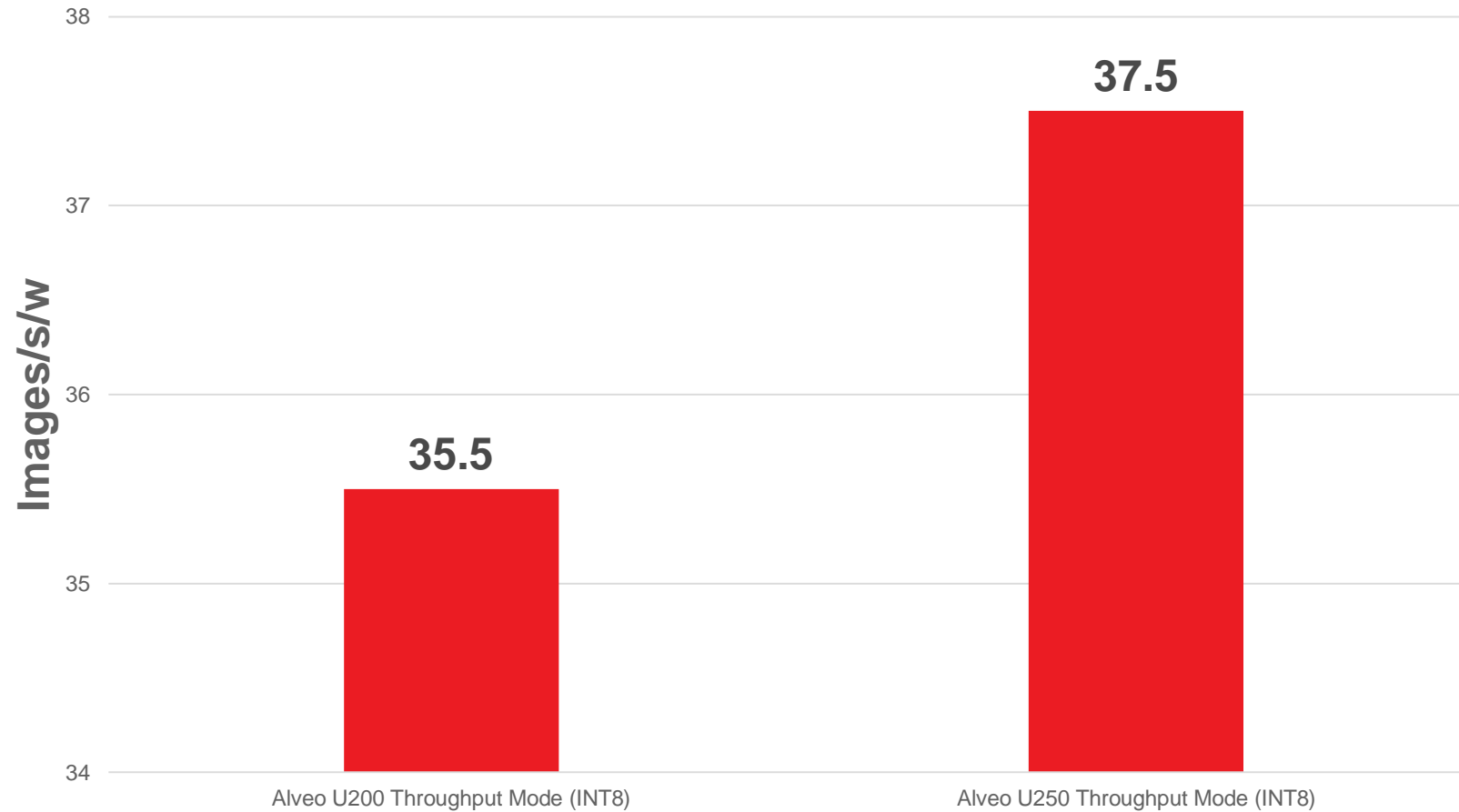
xDNN YOLO v2 Performance – Image Size 608x608

© Copyright 2018 Xilinx
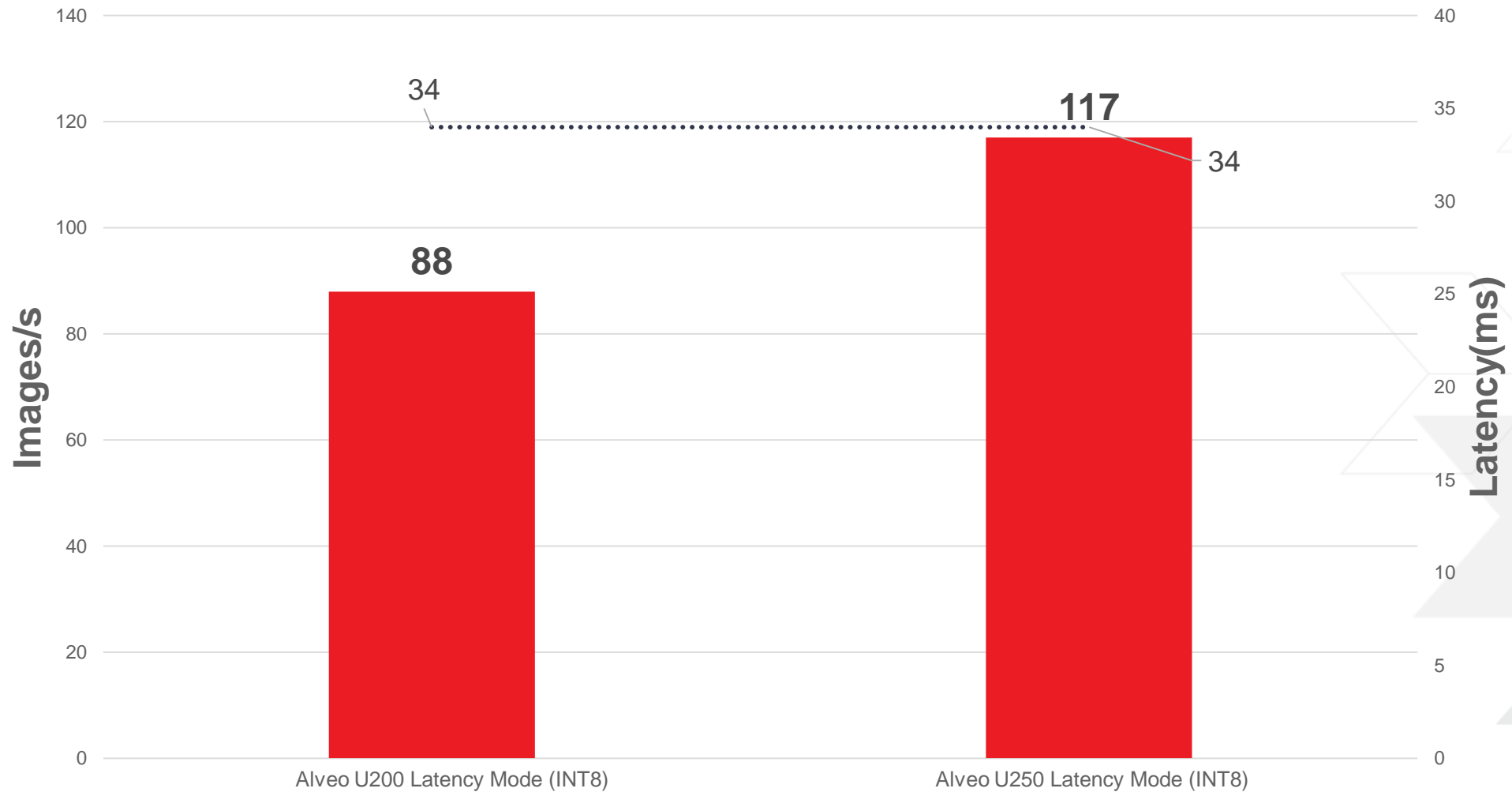
XILINX

Custom Deep Learning Pipeline
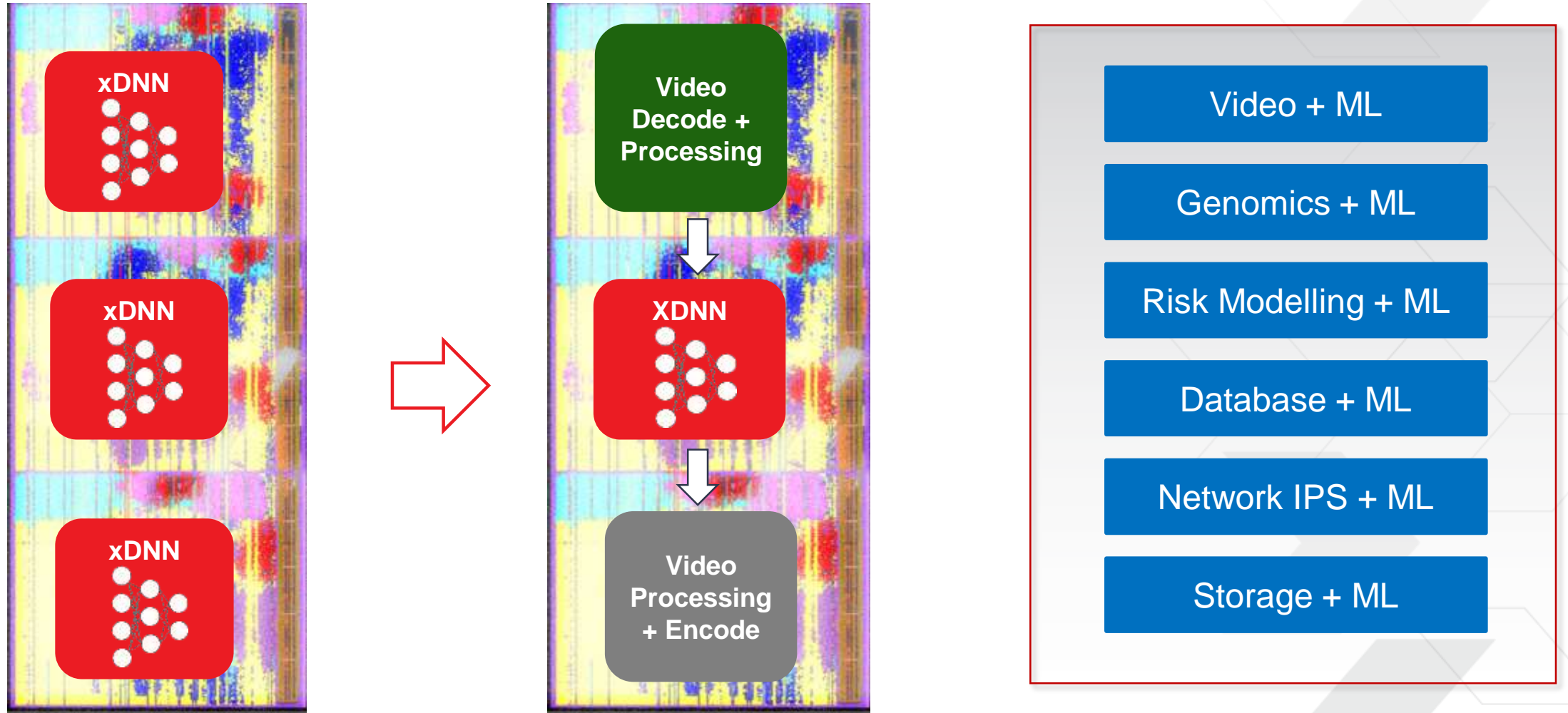
Integrate Custom Applications with xDNN. Lower end-to-end latency

© Copyright 2018 Xilinx

XILINX

Additional Information

> Visit Alveo Demo Room

> Refer to White Paper WP504
>> "Accelerating DNNs with Xilinx Alveo Accelerator Cards"

> **https://www.xilinx.com/applications/megatrends/machine-learning.html**

White Paper: Alveo Data Center Accelerator Cards

XILINX®

WP504 (v1.0) October 2, 2018

Accelerating DNNs with Xilinx Alveo Accelerator Cards

The Xilinx xDNN processing engine, using Xilinx Alveo Data Center accelerator cards, is a high-performance energy-efficient DNN accelerator and outperforms many common CPU and GPU platforms today in raw performance and power efficiency for real-time inference workloads. The xDNN processing engine is delivered through the ML Suite available on many cloud environments, such as AWS EC2 or Nimbix NX5.

ABSTRACT

The Xilinx® Deep Neural Network (xDNN) engine provides high-performance, low-latency, energy-efficient DNN acceleration using Xilinx® Alveo™ Data Center accelerator cards. By keeping energy costs low and minimizing the number of specific accelerators needed in the implementation, total cost of ownership (TCO) can be significantly reduced.

Xilinx Alveo accelerator cards excel at high-performance, energy-efficient, flexible Machine Learning (ML) inference. The xDNN processing engine has been developed to generically execute Convolutional Neural Networks (CNNs) like ResNet50, GoogLeNet v1, Inception v4—even CNNs containing custom layers.

This white paper presents an overview of the xDNN hardware architecture and software stack, as well as benchmarking data that supports the claim of "Best in Class" energy-efficient inference.

Guidance to the reader is provided to enable re-creation of the results on the

© Copyright 2018 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

WP504 (v1.0) October 2, 2018   www.xilinx.com   1

© Copyright 2018 Xilinx

Adaptable.
Intelligent.

XILINX.