



# PROGRAMMERS' INTRODUCTION TO THE INTEL<sup>®</sup> FPGA DEEP LEARNING ACCELERATION SUITE

Class 4

# What We Discussed in the Previous Lesson

Using the Intel® Distribution of Open Visual Inference and Neural Network Optimization (OpenVINO™) toolkit to bring trained models from a DL framework into the Deep Learning Deployment Toolkit and optimize it and then deploy it on a CPU and FPGA.

# Agenda

Intel® FPGA Deep Learning Acceleration Suite

Execution on the FPGA

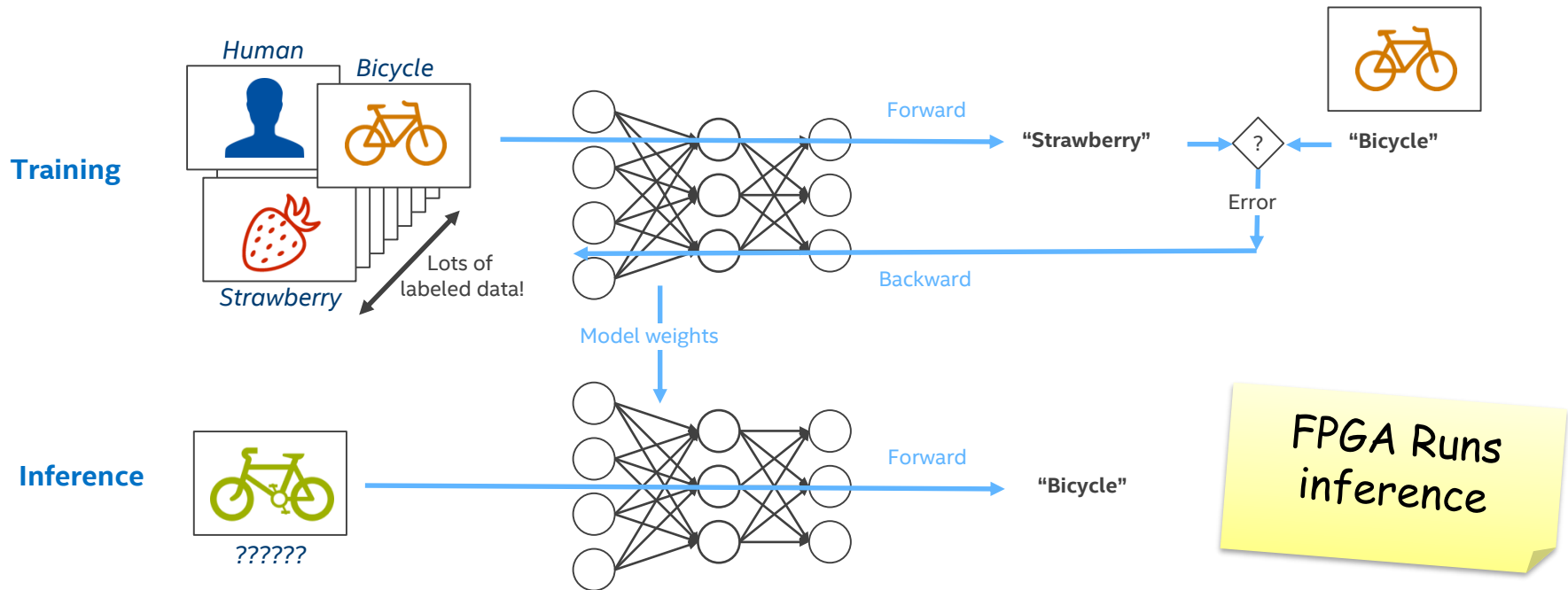
# Objective

Explain the major components of the Deep Learning Acceleration Suite.

Explain how network topologies are mapped onto FPGA architectures using the Intel® Distribution of OpenVINO™ toolkit.

Explain the different architectures available for the FPGA and how lower precision is handled in those architectures.

# Deep Learning: Training vs. Inference



# INTEL® AI PORTFOLIO

## EXPERIENCES



## FRAMEWORKS



## TOOLS



Intel® Deep Learning  
Deployment Toolkit

Intel® Distribution of  
OpenVINO™ toolkit

## LIBRARIES



Intel Python  
Distribution

Intel® DAAL

Intel® FPGA DL  
Acceleration  
Suite

Intel® Nervana™ Graph  
Intel® Math Kernel Library  
(MKL, MKL-DNN)

Associative  
Memory Base

## HARDWARE



Compute



Memory & Storage



Networking



Visual Intelligence

UNLEASH  
FULL  
POTENTIAL

# Solving Machine Learning Challenges with FPGA



## EASE-OF-USE

SOFTWARE ABSTRACTION,  
PLATFORMS & LIBRARIES

*Intel® FPGA solutions enable software-defined programming of customized machine learning accelerator libraries.*



## REAL-TIME

DETERMINISTIC  
LOW LATENCY

*Intel® FPGA hardware implements a deterministic low latency data path unlike any other competing compute device.*



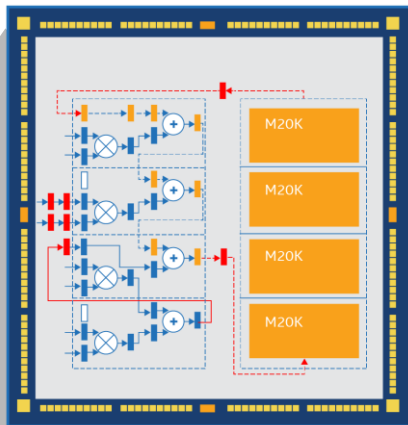
## FLEXIBILITY

CUSTOMIZABLE HARDWARE  
FOR NEXT GEN DNN ARCHITECTURES

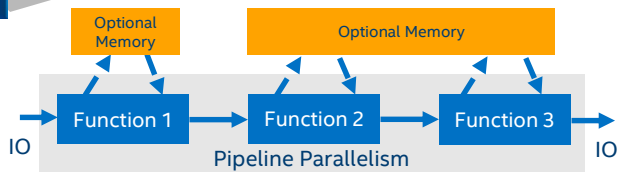
*Intel® FPGAs can be customized to enable advances in machine learning algorithms.*

# Why Intel® FPGAs for Machine Learning?

## Convolutional Neural Networks are Compute Intensive



**Fine-grained & low latency  
between compute and memory**



Feature	Benefit
Highly parallel architecture	Facilitates efficient low-batch video stream processing and reduces latency
Configurable Distributed Floating Point DSP Blocks	FP32 9Tflops, FP16, FP11 Accelerates computation by tuning compute performance
Tightly coupled high-bandwidth memory	>50TB/s on chip SRAM bandwidth, random access, reduces latency, minimizes external memory access
Programmable Data Path	Reduces unnecessary data movement, improving latency and efficiency
Configurability	Support for variable precision (trade-off throughput and accuracy). Future proof designs, and system connectivity

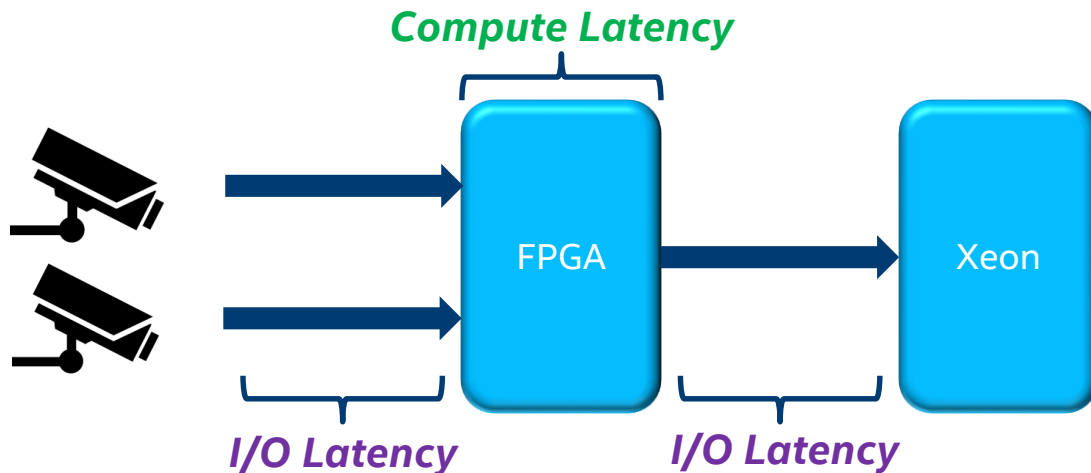


# FPGAs Provide Deterministic System Latency

FPGAs leverage parallelism across the entire chip to reduce compute latency

FPGAs have flexible and customizable I/Os with low & deterministic I/O latency

$$\text{System Latency} = \text{I/O Latency} + \text{Compute Latency}$$

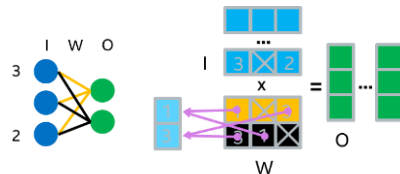
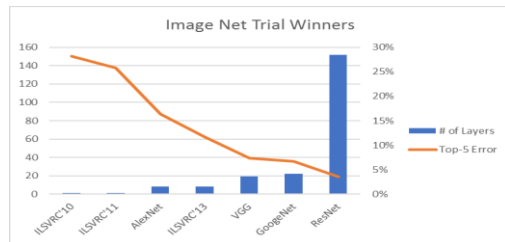


# Intel® FPGAs

## Future Proof: Rapid Innovation of DL Topologies

- Deep Learning is undergoing constant innovation
  - Better Accuracy/Higher Compute Density
- Efforts to improve throughput and efficiency are ongoing
  - Batching, Sparsity, Weight Sharing, Compression, etc.
- This rapid and constant evolution can present a challenge if implemented on a fixed architecture (e.g., a GPU) ...

Ideal for an Intel® FPGA!



# Intel® FPGA Deep Learning Acceleration Suite

CNN acceleration engine for common topologies executed in a graph loop architecture

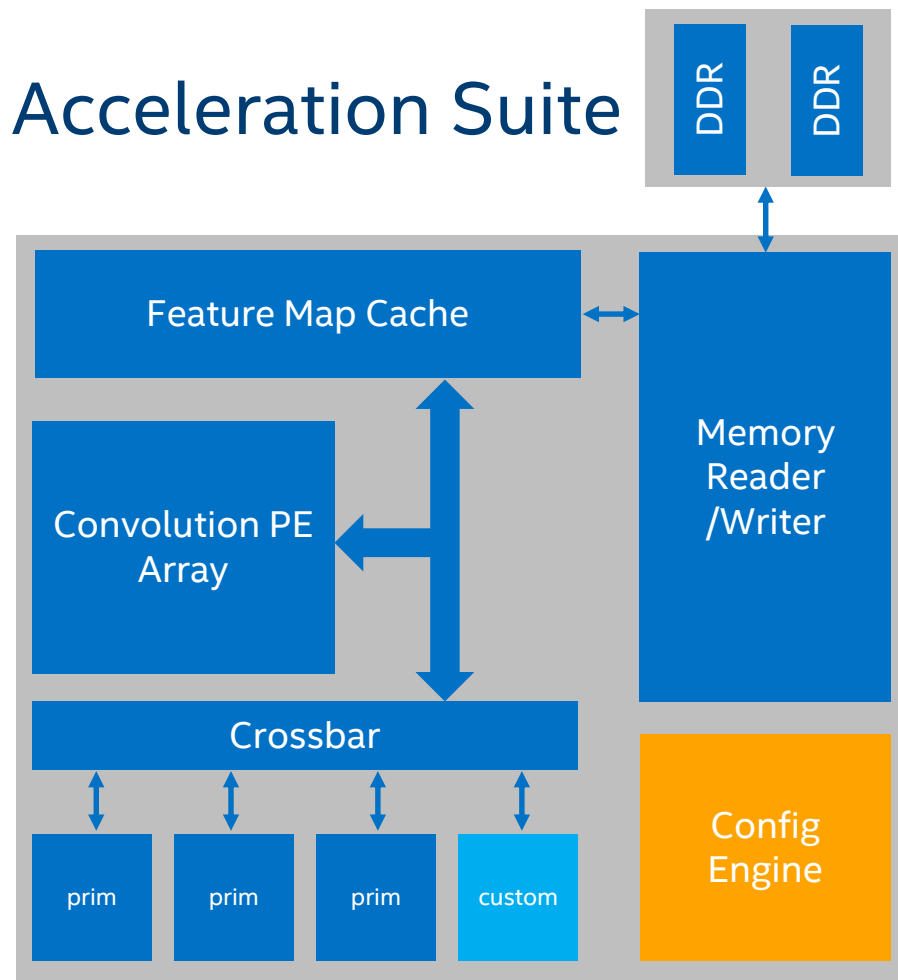
- AlexNet, GoogleNet, LeNet, SqueezeNet, VGG16, ResNet, Yolo, SSD...

## Software Deployment

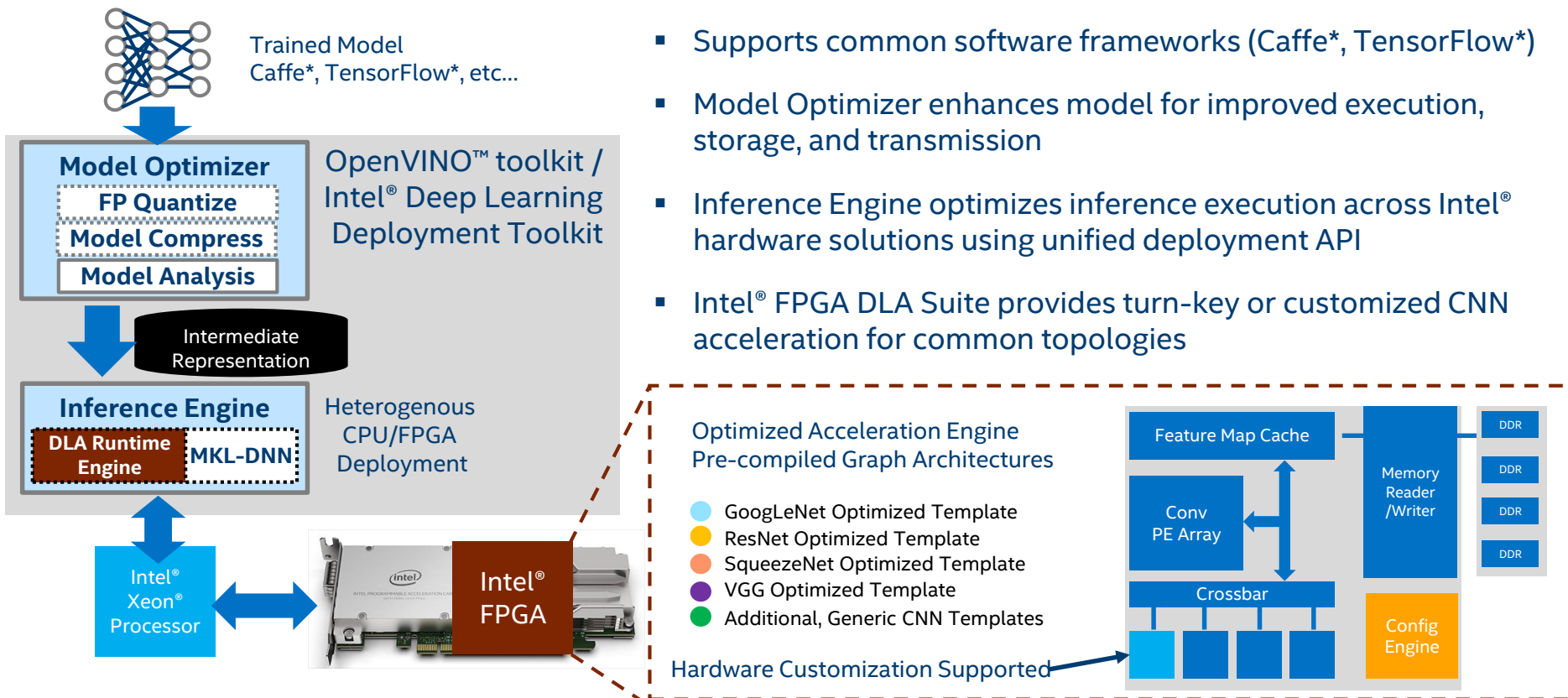
- No FPGA compile required
- Run-time reconfigurable

## Customized Hardware Development

- Custom architecture creation w/ parameters
- Custom primitives using OpenCL™ flow

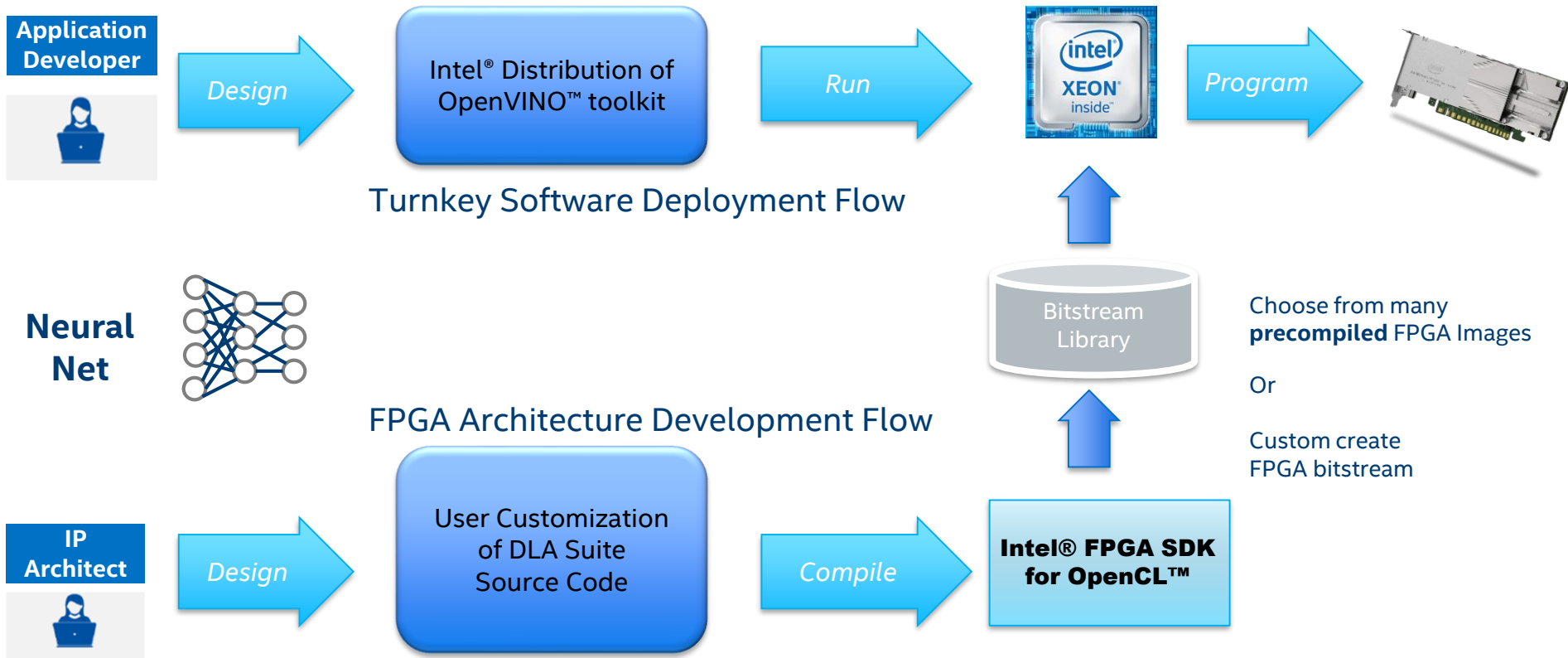


# FPGA Usage with Intel Distribution of OpenVINO™ toolkit



\*Other names and brands may be claimed as the property of others

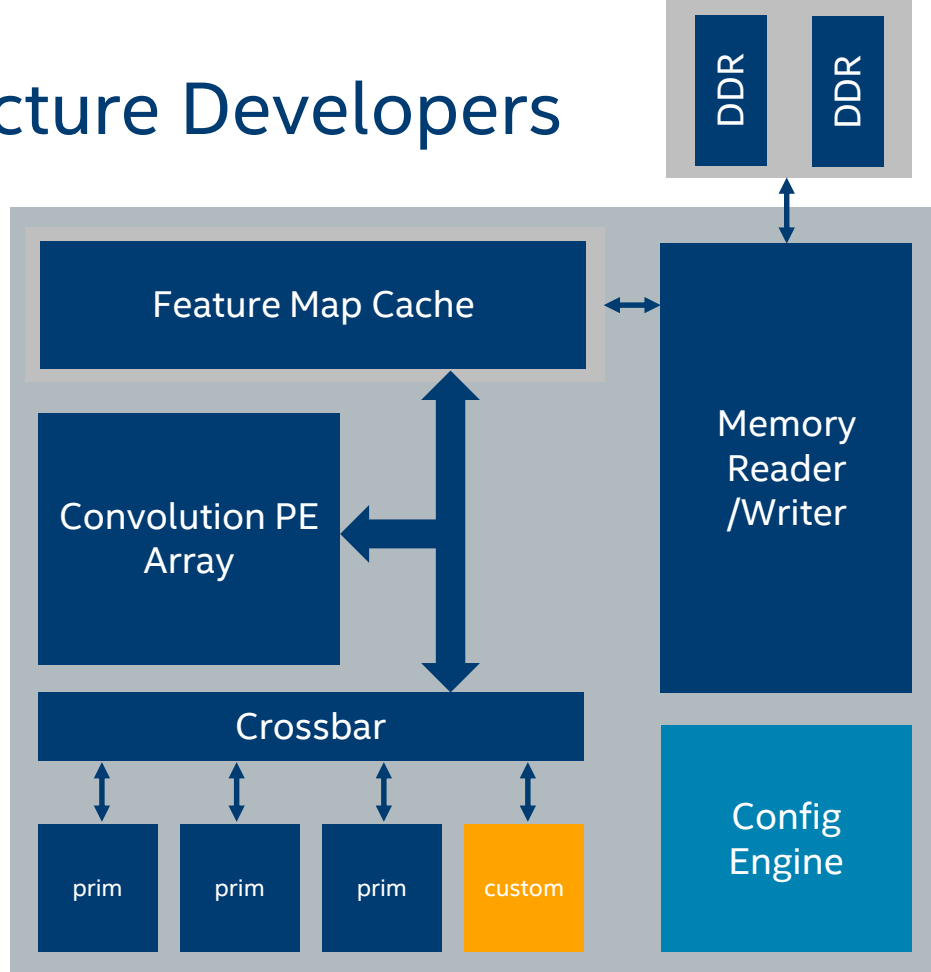
# DLA User Flows



# Customization for Architecture Developers

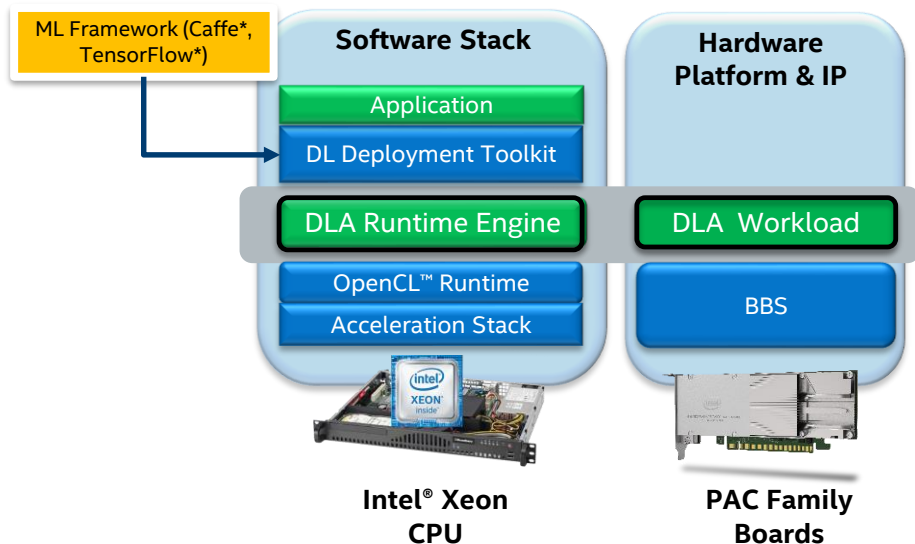
## Add a custom primitive into crossbar

- Primitive types supported:
  - Unary (ReLU, Tanh, Abs, Power, ...)
  - Binary (Eltwise Add, Mult, ...)
- Parameterizable from Deep Learning Framework



# Machine Learning on Intel® FPGA Platform

## Acceleration Stack Platform Solution



Also will be available on other platforms in the near future

For more information on the Acceleration Stack for Intel® Xeon® CPU with FPGAs on the Intel® Programmable Acceleration Card, visit the [Intel® FPGA Acceleration Hub](#)

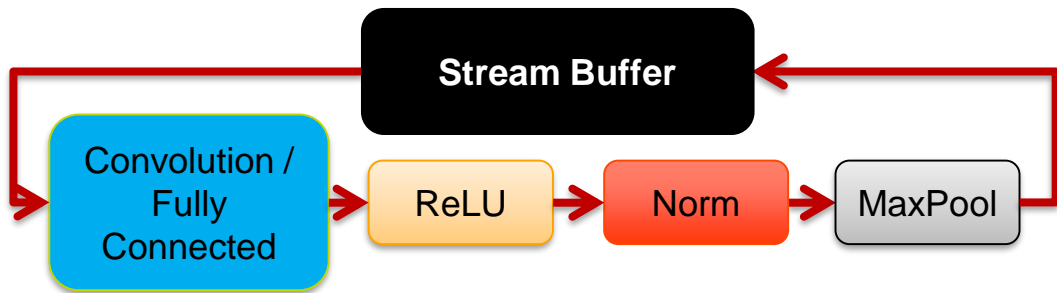
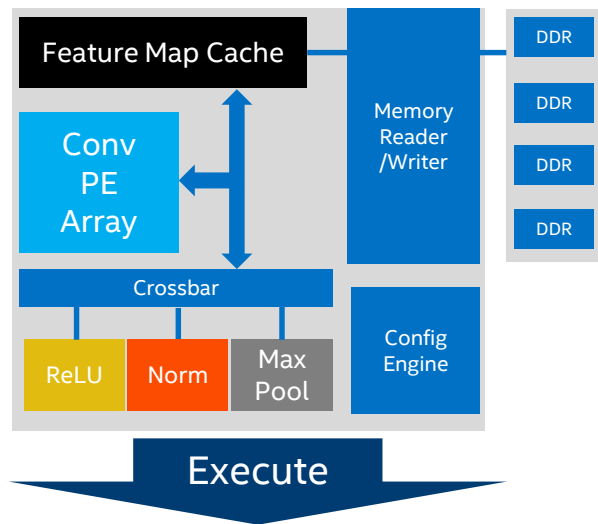
# DLA Architecture: Built for Performance

## Maximize Parallelism on the FPGA

- Filter Parallelism (Processing Elements (PE))
- Input-Depth Parallelism
- Winograd Transformation
- Batching
- Feature Stream Buffer
- Filter Cache

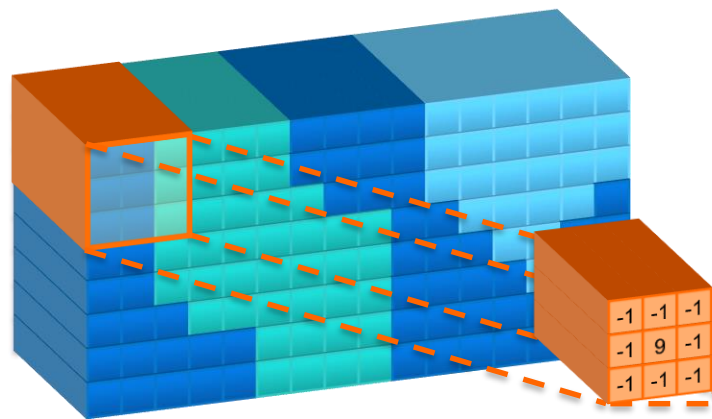
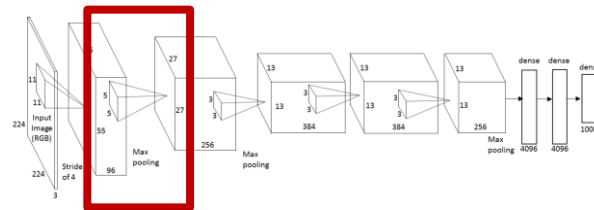
## Choosing FPGA Bitstream

- Data Type / Design Exploration
- Primitive Support





# CNN Computation in One Slide

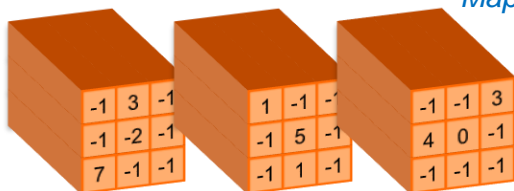


Input Feature Map  
(Set of 2D Images)

Filter  
(3D Space)

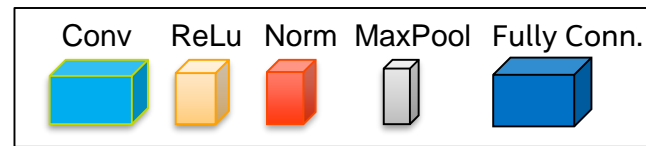
$$I_{\text{new}}[x][y] = \sum_{x'=-1}^1 \sum_{y'=-1}^1 I_{\text{old}}[x+x'][y+y'] \times F[x'][y']$$

Output Feature Map

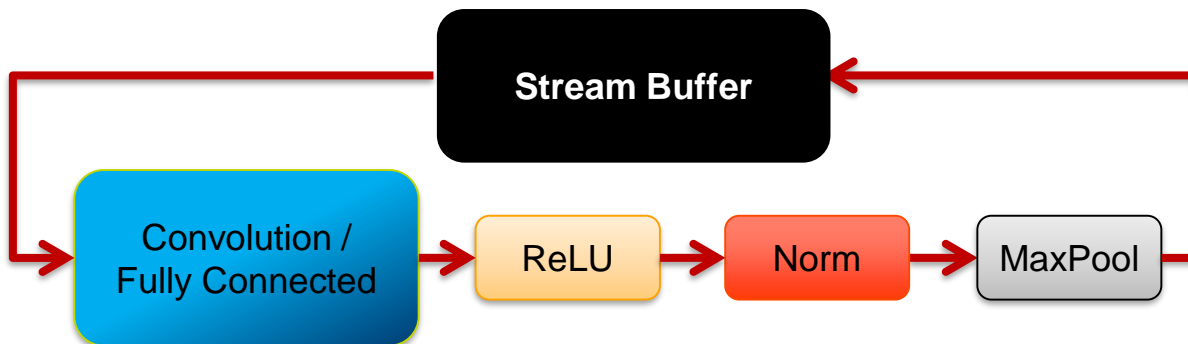
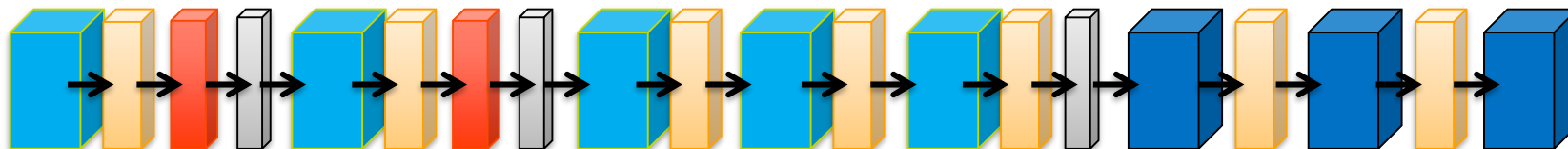


**Repeat for Multiple Filters  
to Create Multiple “Layers”  
of Output Feature Map**

# Mapping Graphs in DLA

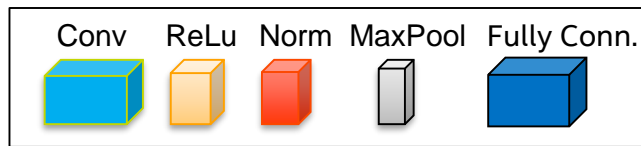


AlexNet Graph

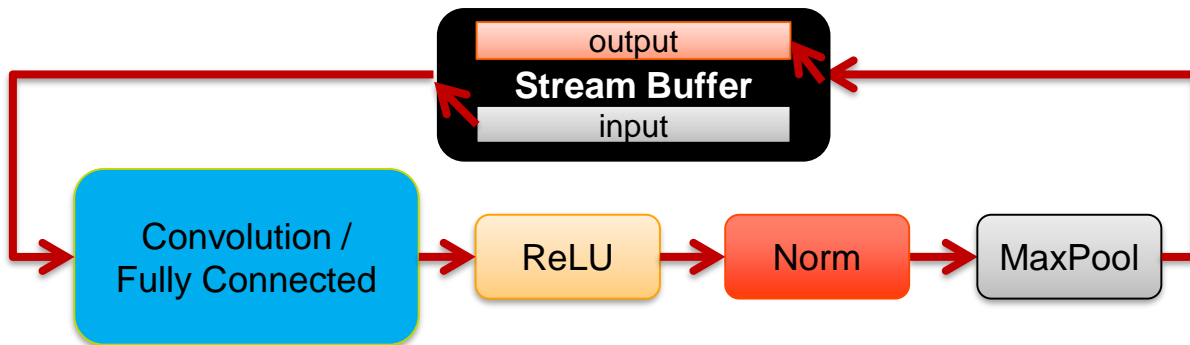
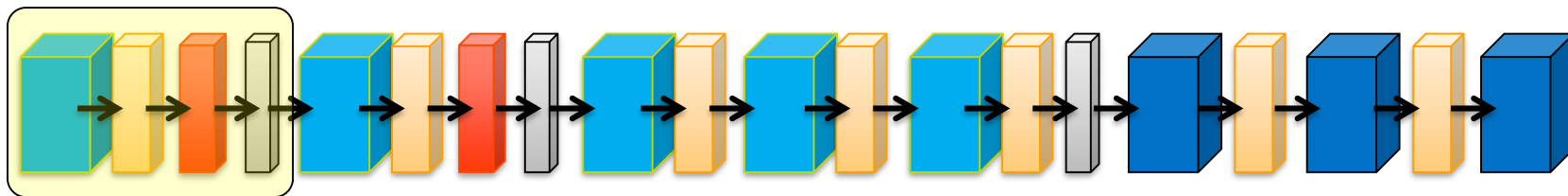


Blocks are run-time reconfigurable and bypassable

# Mapping Graphs in DLA

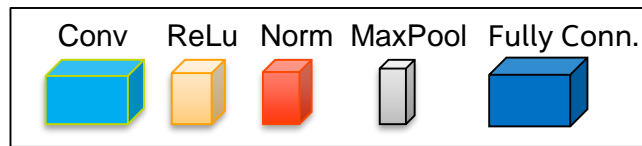


AlexNet Graph

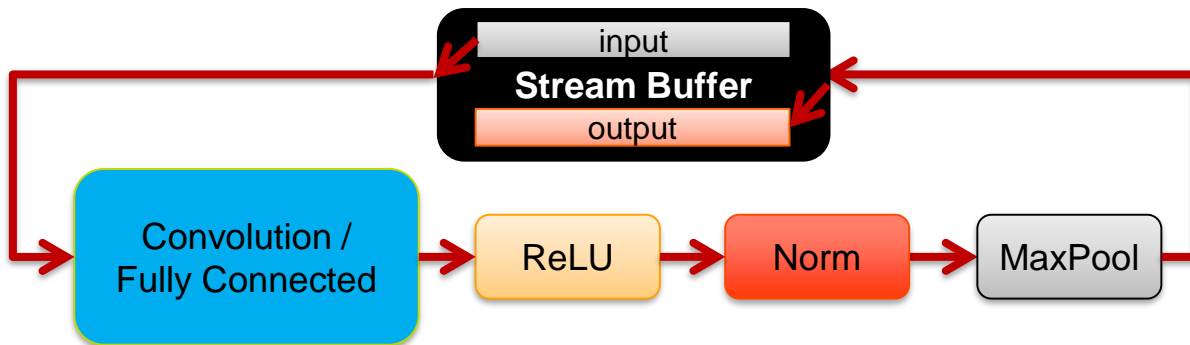
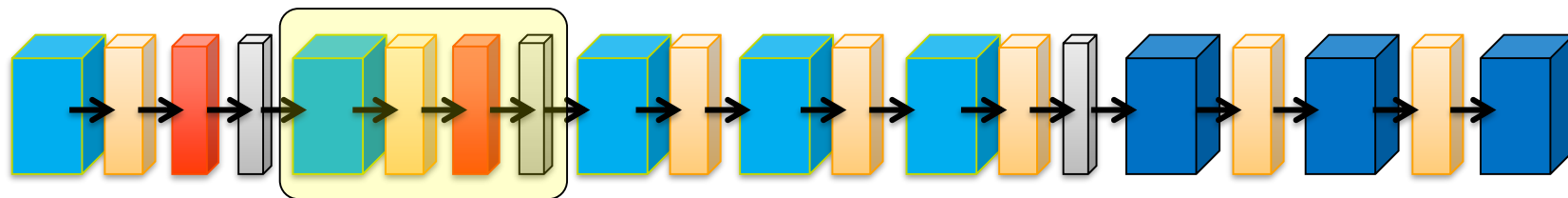


Blocks are run-time reconfigurable and bypassable

# Mapping Graphs in DLA

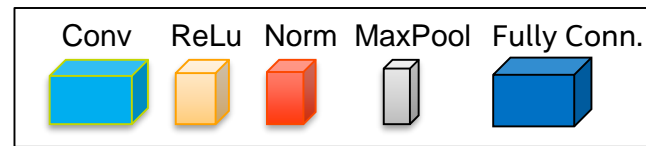


AlexNet Graph

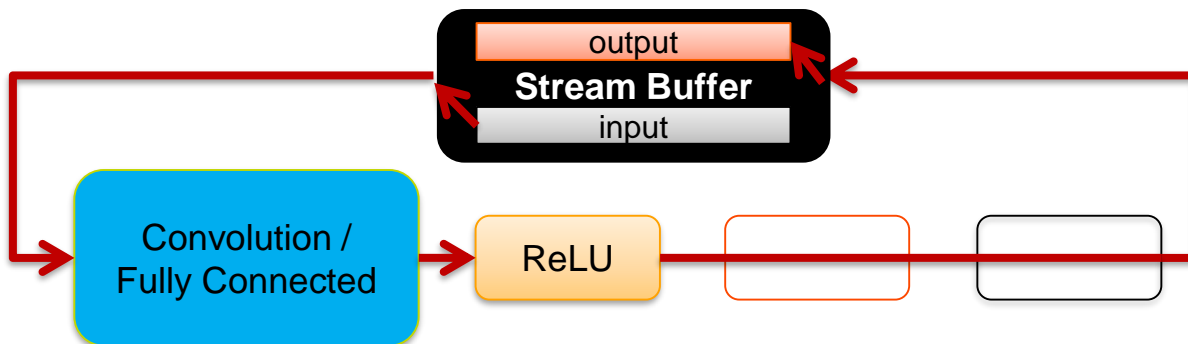
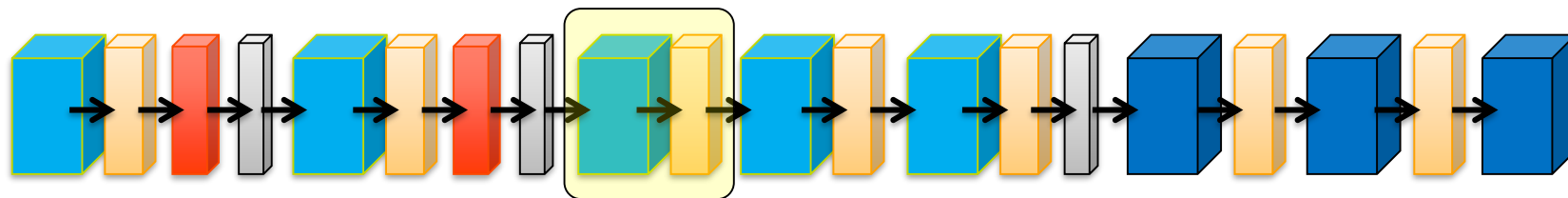


Blocks are run-time reconfigurable and bypassable

# Mapping Graphs in DLA

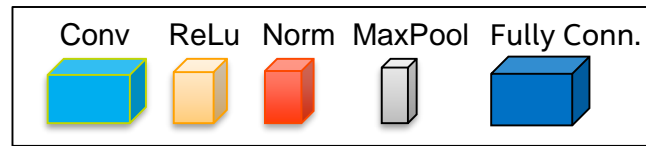


AlexNet Graph

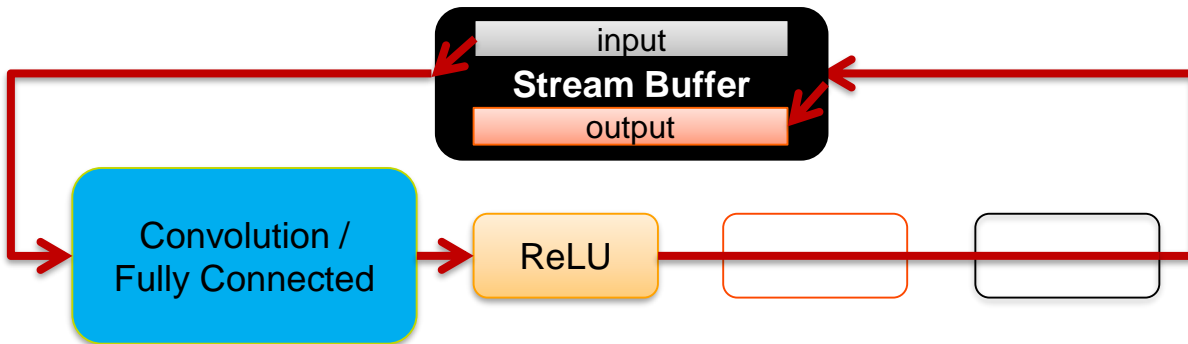
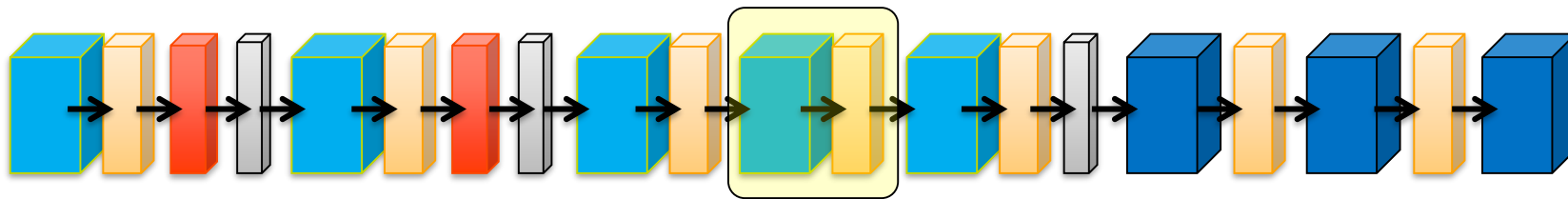


Blocks are run-time reconfigurable and bypassable

# Mapping Graphs in DLA

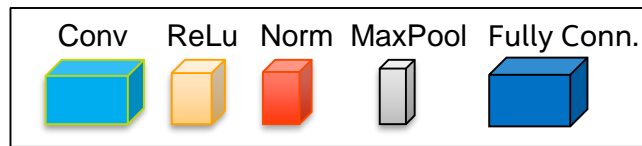


# AlexNet Graph

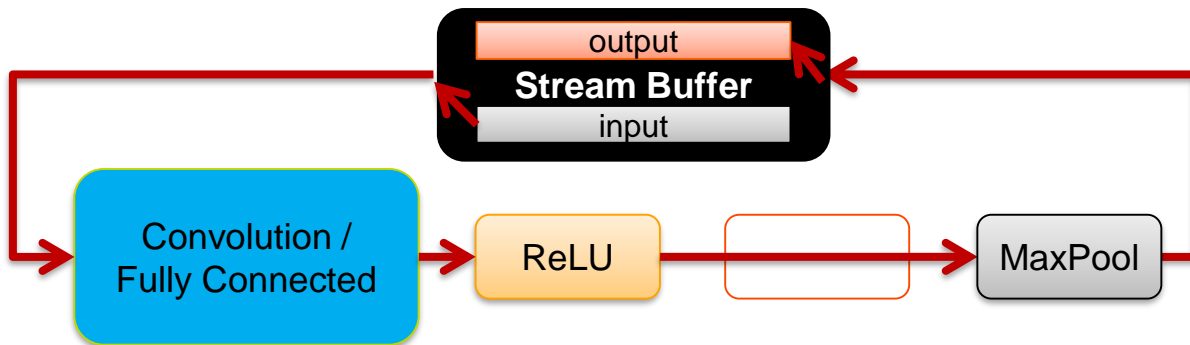
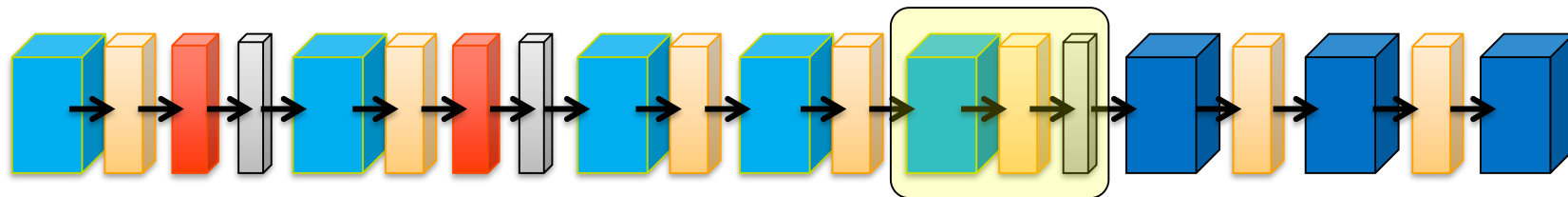


## Blocks are run-time reconfigurable and bypassable

# Mapping Graphs in DLA

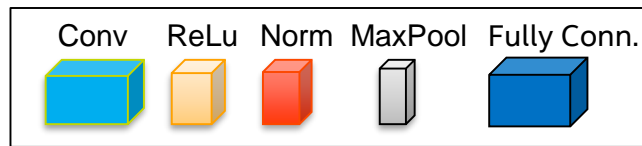


AlexNet Graph

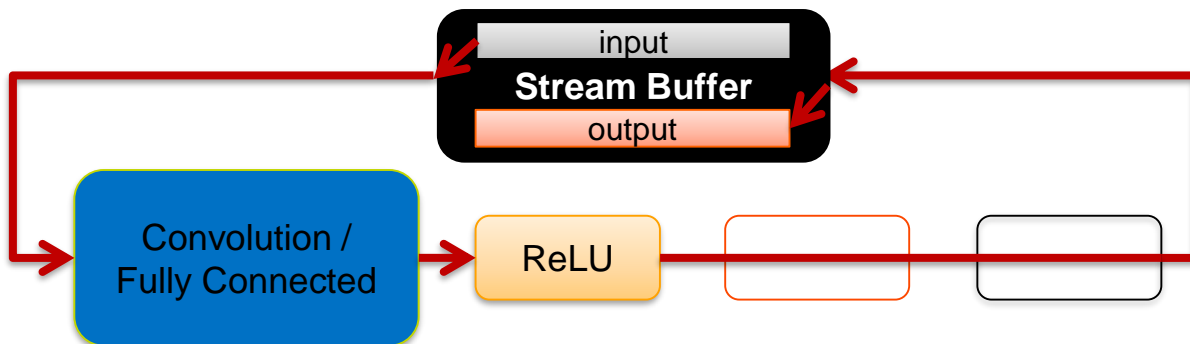
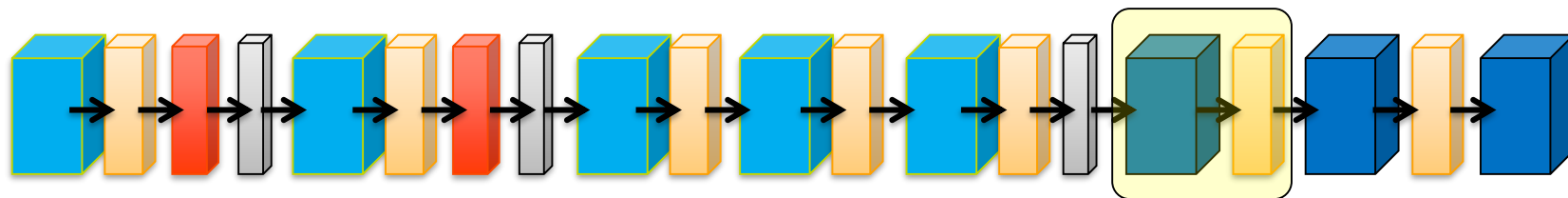


Blocks are run-time reconfigurable and bypassable

# Mapping Graphs in DLA



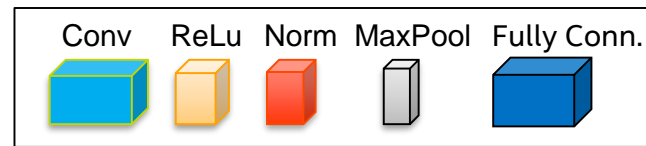
AlexNet Graph



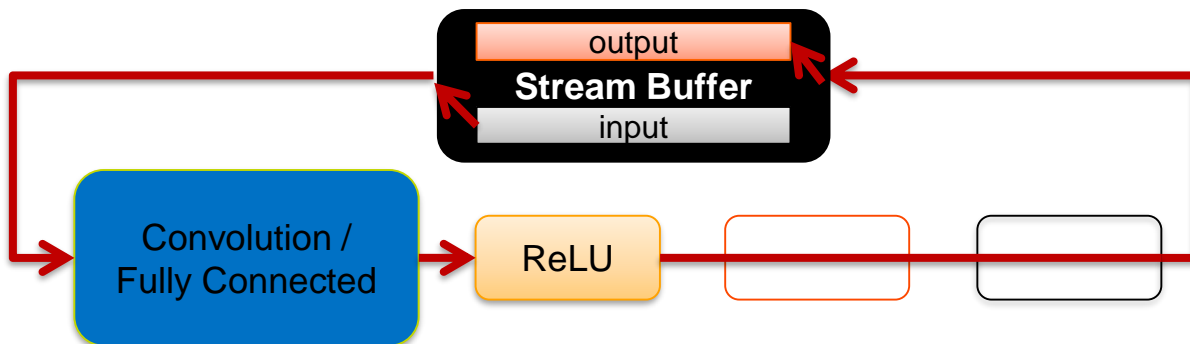
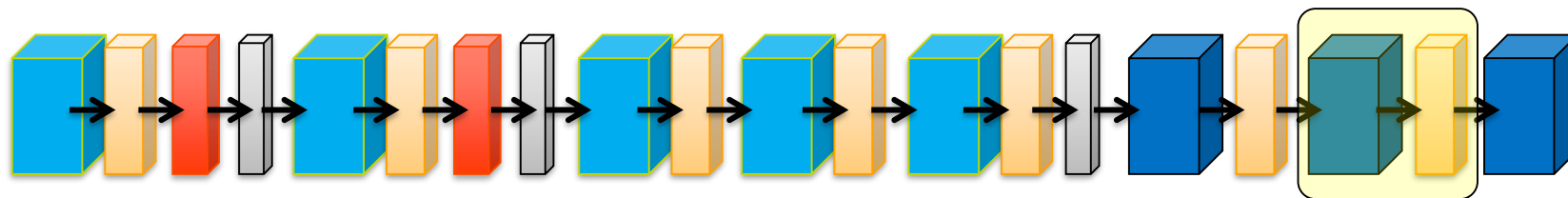
Blocks are run-time reconfigurable and bypassable



# Mapping Graphs in DLA

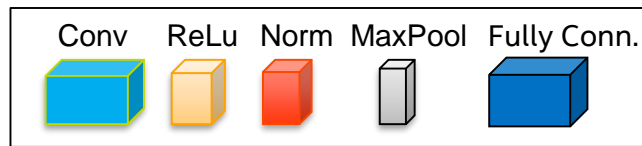


AlexNet Graph

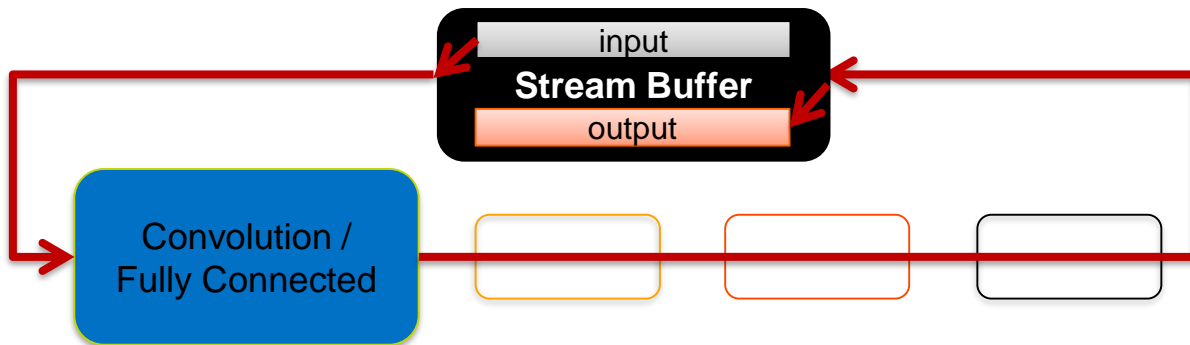
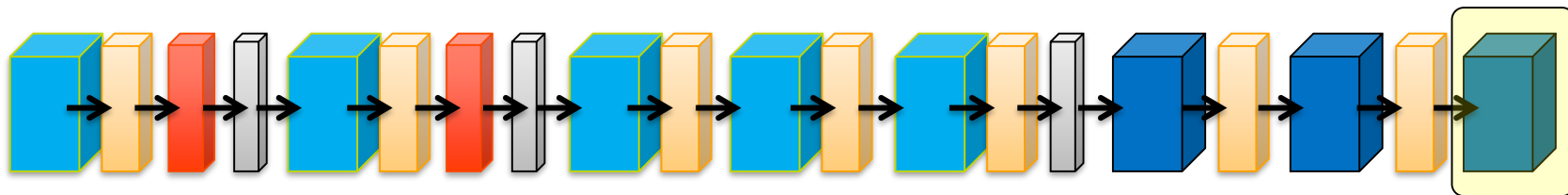


Blocks are run-time reconfigurable and bypassable

# Mapping Graphs in DLA

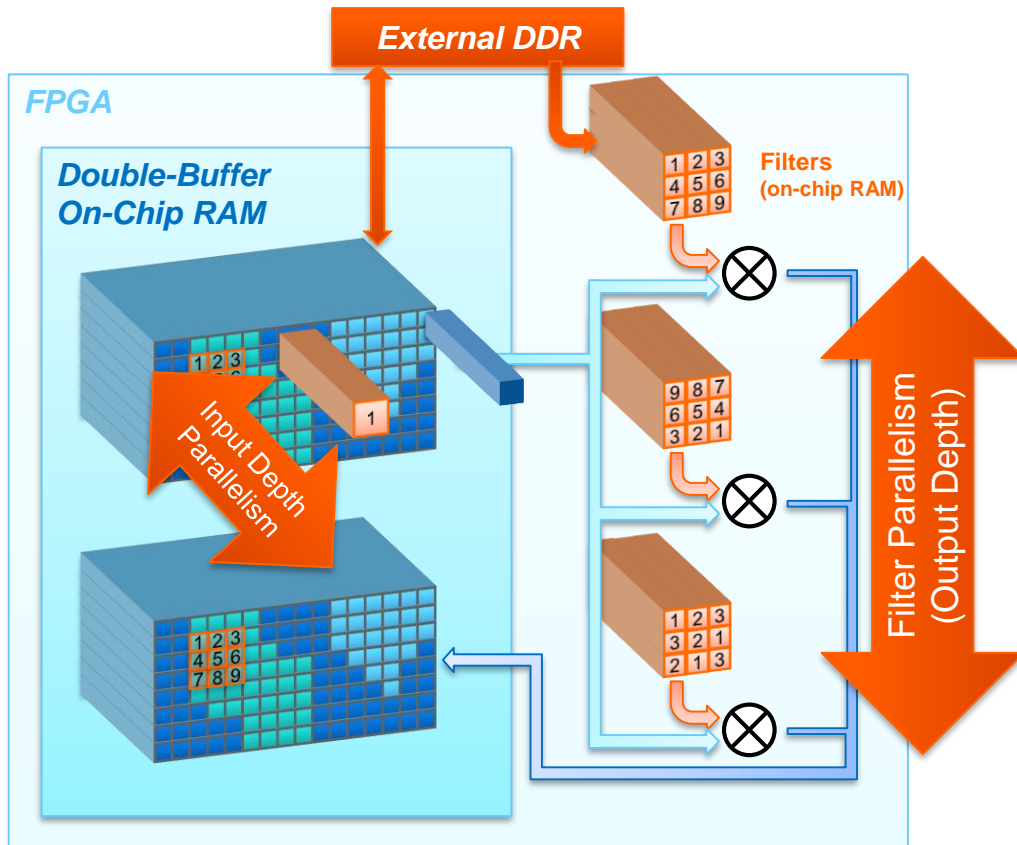


AlexNet Graph



Blocks are run-time reconfigurable and bypassable

# Efficient Parallel Execution of Convolutions



## Parallel Convolutions

- Different filters of the same convolution layer processed in parallel in different processing elements (PEs)

## Vectorized Operations

- Across the depth of feature map

PE Array geometry can be customized to hyperparameters of given topology

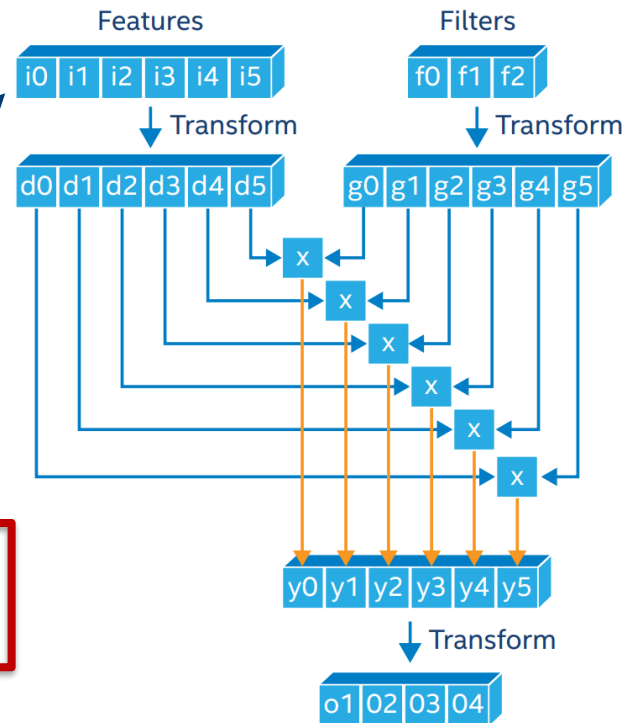
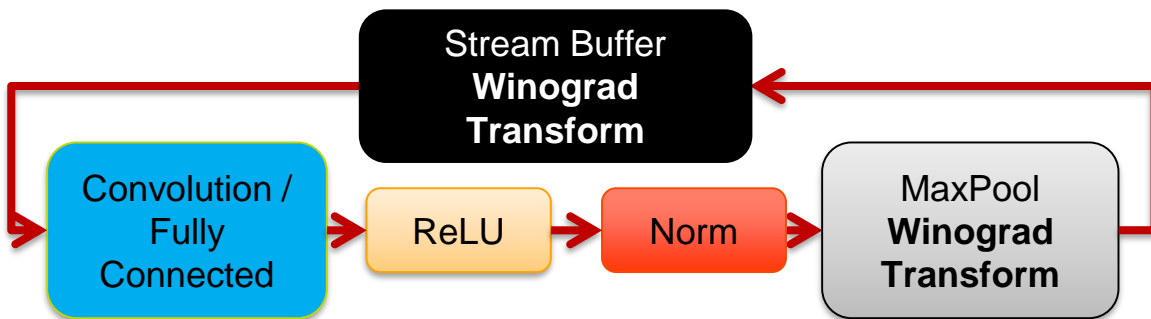
# Winograd Transformation

Perform convolutions with fewer multiplication

- Allows more convolutions to be done on FPGA

Take 6 input features elements and 3 filter elements

- Standard convolution requires 12 multiplies
- Transformed convolution requires just 6 multiplies



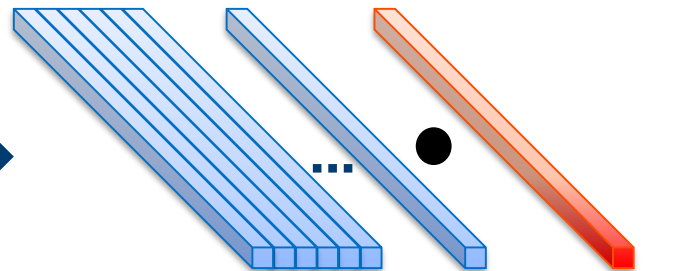
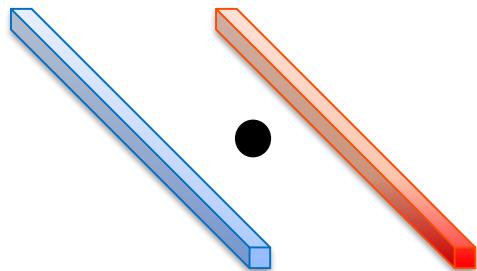
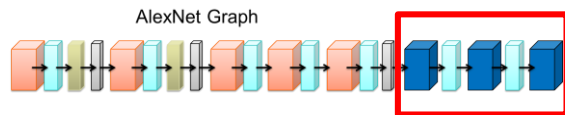
# Fully Connected Computation and Batching

Fully Connected Layer computation does not allow for data reuse of weights

- Different from convolutions
- Very memory bandwidth intensive

Solution: Batch up images

- Weights reused across multiple images



$$O = I_{vec} * W_{vec}$$

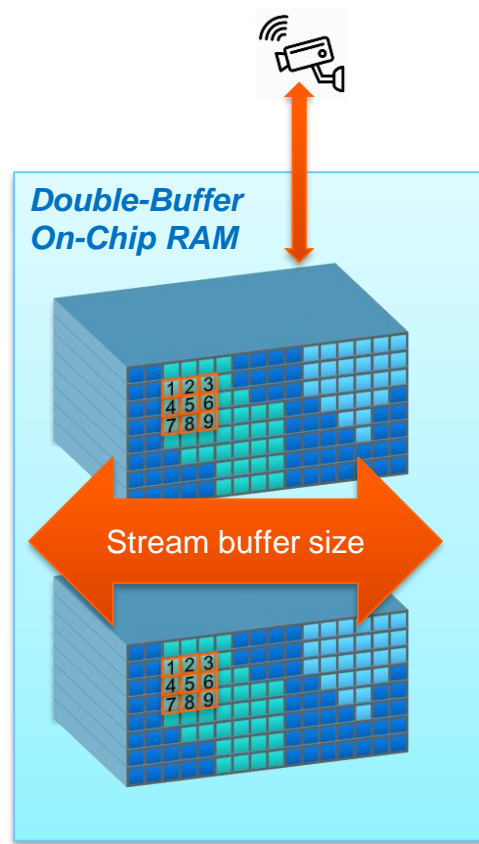
$$O_{vec} = I_{mat} * W_{vec}$$

# Feature Data Cached On-Chip

Streamed to a daisy chain of parallel processing elements

Double buffered

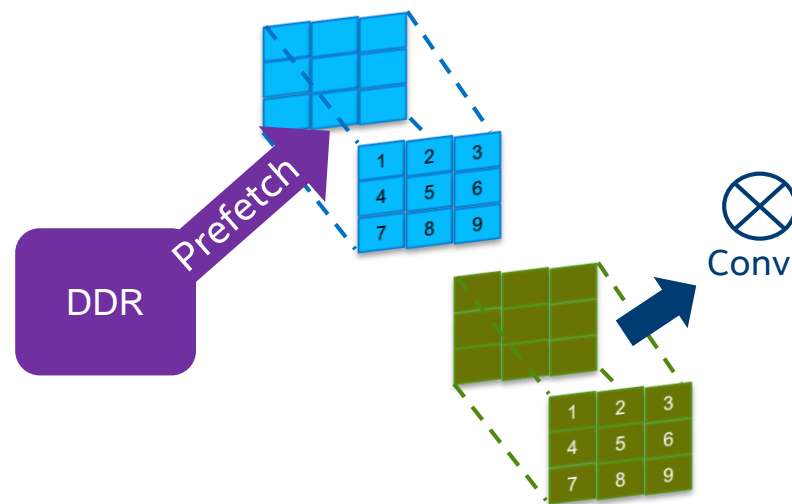
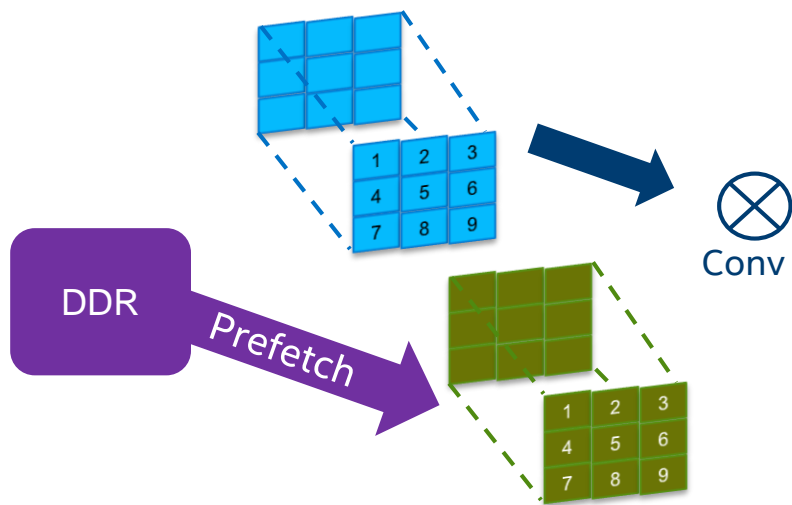
- Overlap convolution with cache updates
- Output of one subgraph becomes input of another
- Eliminates unnecessary external memory accesses



# Filter Weights Cached in Each PE

Double buffered in order to support prefetching

- While one set is used to calculate output feature maps, another set is prefetched



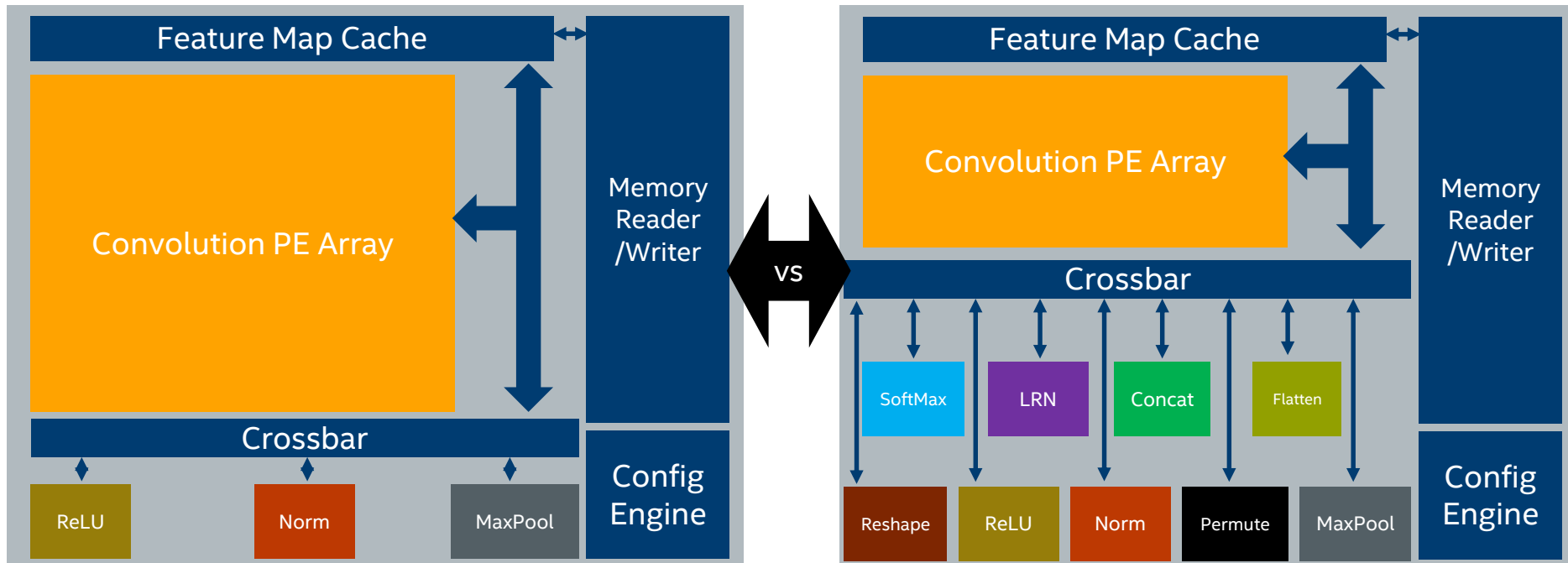
# DLA Architecture Selection

- DLDT ships with many FPGA images for various boards/data types/topologies
  - `<version>_<board>_<data type>_<Topologies/Feature>.aocx`
    - `4-0_PL1_FP11_GoogleNet.aocx`
    - `4-0_RC_FP16_MobileNet_ResNet_VGG_Clamp.aocx`
- Find ideal FPGA image that meets your needs
- Check documentation for list of FPGA images and supported topologies
  - <https://software.intel.com/en-us/articles/OpenVINO-InferEngine#fpga-plugin>
- Create custom FPGA image based on need
- Example: ResNet focused image does not have Norm (better performance)



# Support for Different Topologies

Tradeoff between features and performance



# Supported Primitives and Topologies

## Primitives

- |                   |                     |
|-------------------|---------------------|
| ✓ Conv            | ✓ ReLu, Leaky ReLU  |
| ✓ Concat          | ✓ Eltwise           |
| ✓ Pooling         | ✓ Power             |
| ✓ ScaleShift      | ✓ Batch Norm        |
| ✓ Fully Connected | ✓ LRM Normalization |
| ✓ Custom          |                     |

## Topologies

- |                        |       |
|------------------------|-------|
| ✓ AlexNet              |       |
| ✓ GoogLeNet            | ✓ SSD |
| ✓ ResNet-18/50/101/152 | ✓ SSD |
| ✓ SqueezeNet           | ✓ SSD |
| ✓ VGG-16/19            | ✓ SSD |
| ✓ Tiny Yolo            |       |
| ✓ LeNet                |       |
| ✓ MobileNet v1/v2      |       |

# Design Exploration with Reduced Precision

## Tradeoff between performance and accuracy

- Reduced precision allows more processing to be done in parallel
- Using smaller Floating Point format does not require retraining of network
- FP11 benefit over using INT8/9
  - No need to retrain, better performance, less accuracy loss

FP16 

Sign, 5-bit exponent, 10-bit mantissa

FP11 

Sign, 5-bit exponent, 5-bit mantissa

FP10 

Sign, 5-bit exponent, 4-bit mantissa

FP9 

Sign, 5-bit exponent, 3-bit mantissa

# Summary

Deep Learning Accelerator Suite contains a library of architectures that support a variety of different primitive and topologies.

Intel® Distribution of OpenVINO™ toolkit can be used to map a trained network onto the FPGA architecture without needing an FPGA recompile or reload.

FPGAs can implement lower precision without the need to retrain.

The DLA architectures can be customized to add new primitives.

# Legal Disclaimers/Acknowledgements

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at [www.intel.com](http://www.intel.com).

Intel, the Intel logo, Intel Inside, the Intel Inside logo, MAX, Stratix, Cyclone, Arria, Quartus, HyperFlex, Intel Atom, Intel Xeon and Enpirion are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

OpenCL is the trademark of Apple Inc. used by permission by Khronos

\*Other names and brands may be claimed as the property of others

© Intel Corporation

