

# Co-design of Machine Learning Algorithms for Computer Vision and their Implementation on Reconfigurable Hardware

**Qijing Jenny Huang, Zhen Dong**

\*NVIDIA, UC Berkeley

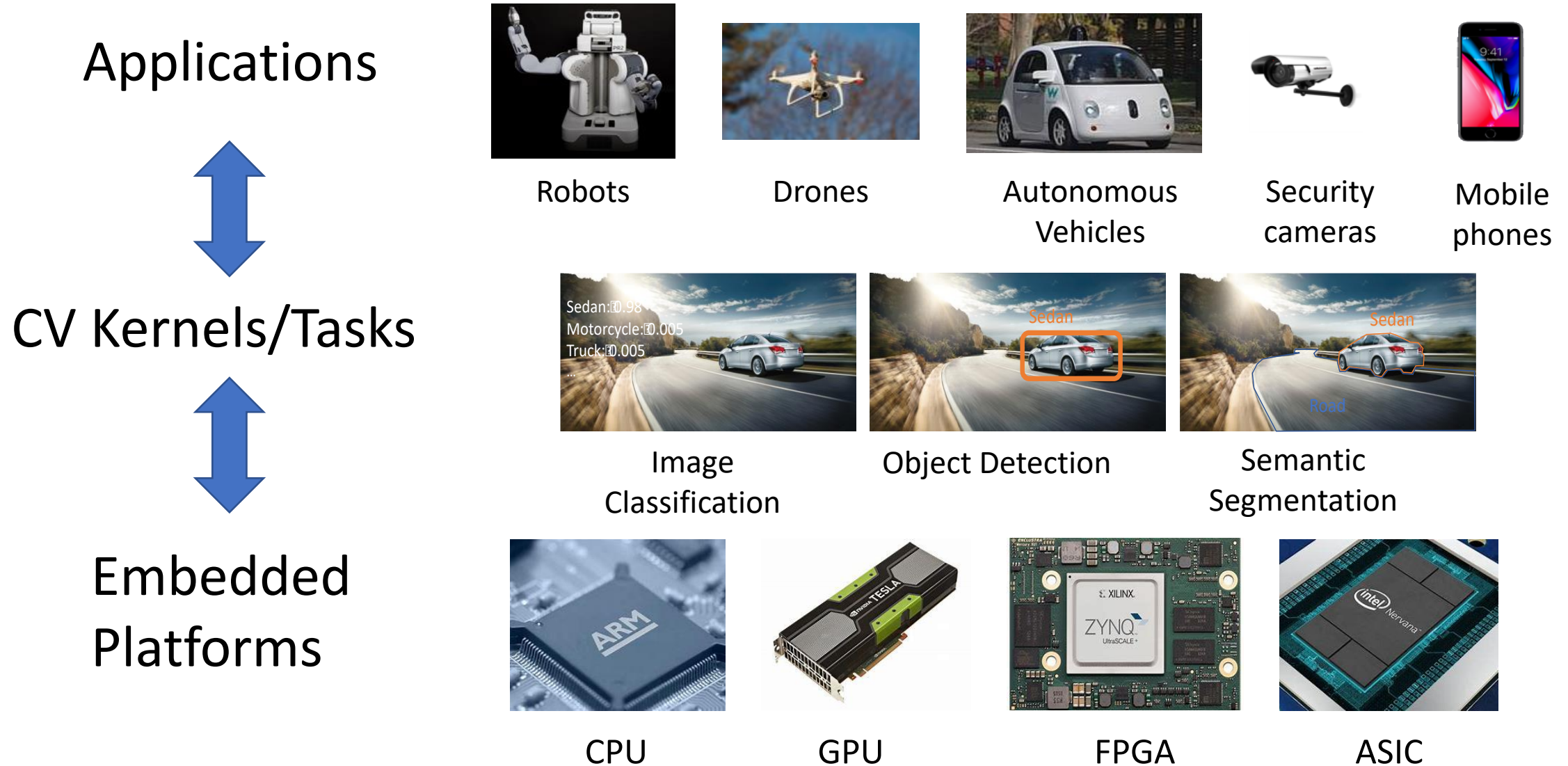
[jennyhuang@nvidia.com](mailto:jennyhuang@nvidia.com)

[zhendong@berkeley.edu](mailto:zhendong@berkeley.edu)

# Outline

- **Motivation**
- Codesign for Image Classification
  - Synergy DiracDeltaNet
- Codesign for Object Detection
  - Deformable Convolution
- Hardware-aware NAS
  - HAO

# Embedded Computer Vision



# Field Programmable Gate Array (FPGA)

- An integrated circuit designed to be reconfigurable after manufacturing



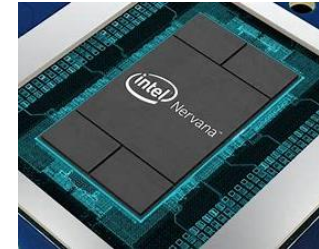
CPU



GPU



FPGA



ASIC

Programmability

Efficiency



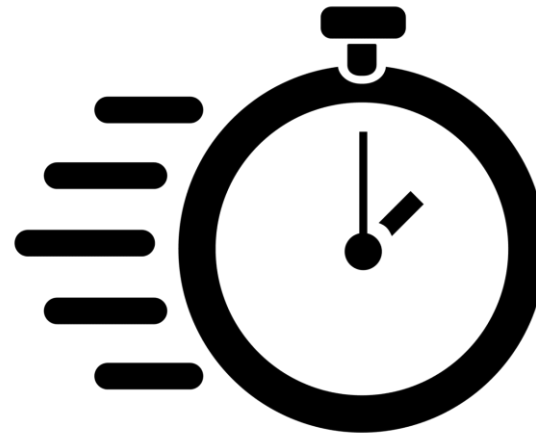
- Low power and energy consumption
- Latency-optimized
  - Good for low batch size inference
- Usually programmed using HW design abstraction
  - Additional timing information is required
  - High level synthesis (HLS) enables SW like abstraction
- More in EECS151

# Goals for Ebedded CV

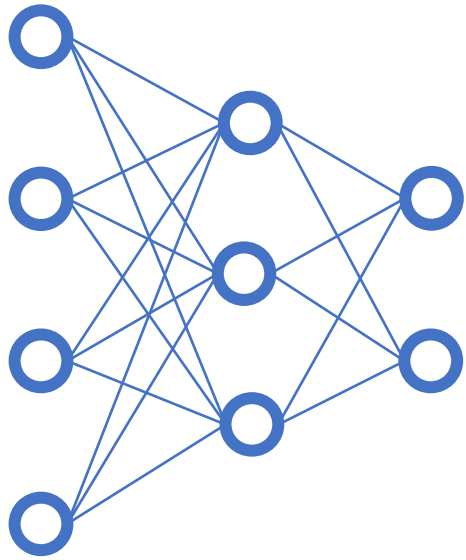
## Accuracy



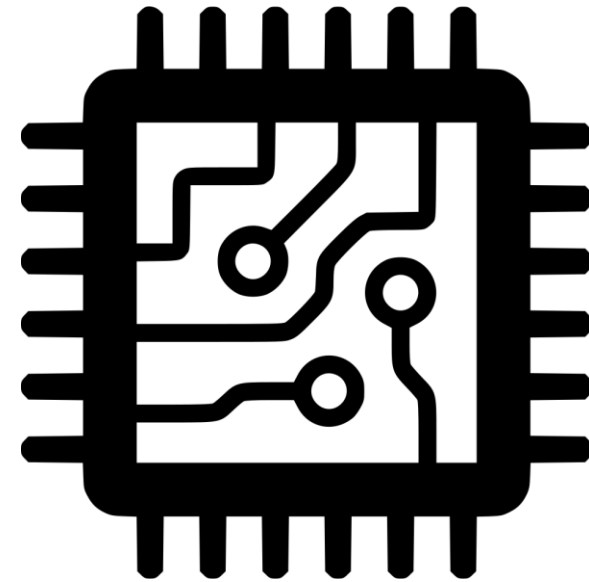
## Efficiency



# How to improve accuracy and efficiency?



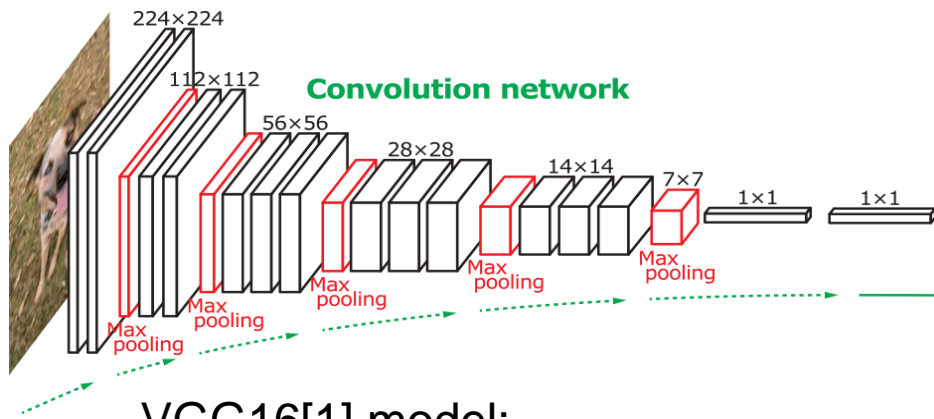
Design better ConvNet



Design better hardware

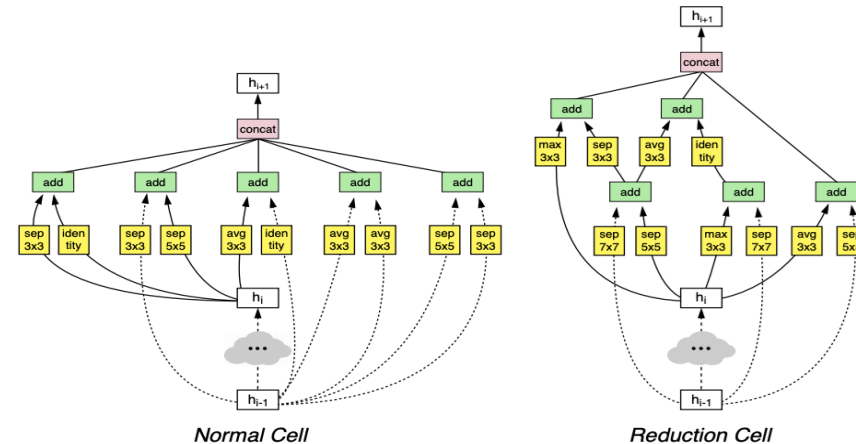
# ConvNet Design

- CV community has evolved ConvNets for good accuracy – efficiency has been less important
- Efficiency proxies have been FLOPs and model size, ignoring hardware friendliness



VGG16[1] model:

- Parameter size: 552 MB
- Computation: 15.8 GOPs/image



NasNet[2] model:

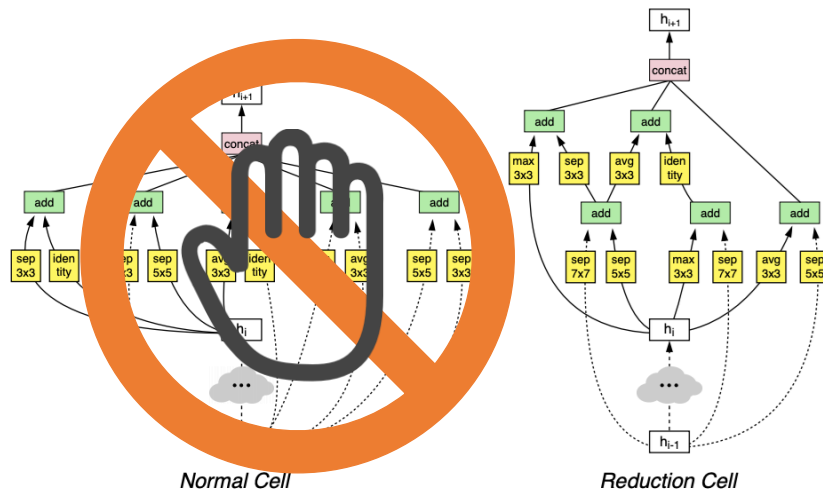
- Parameter size: 5.3 MB
- Computation: 1.28 GOPs/image

[1] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[2] Zoph, B., Vasudevan, V., Shlens, J. and Le, Q.V. Learning Transferable Architectures for Scalable Image Recognition. *arXiv e-prints*. 1707-7012.

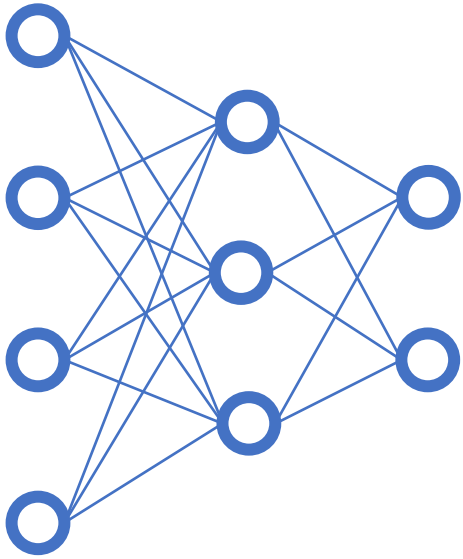
# Hardware Design

- Most work only supports off-the-shelf network designs
- Most effort has focused on reducing precision and on pruning
- Often this throughput improvement comes at the expense of *lower accuracy*

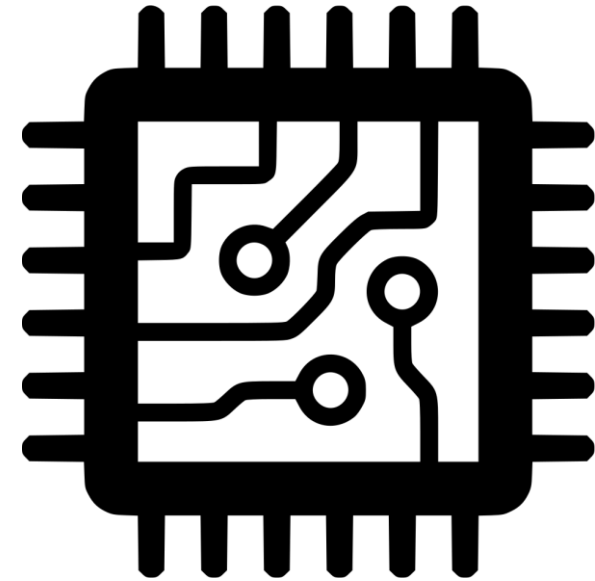




Can we close this gap?



Design better ConvNet



Design better hardware

# Motivation

- Why codesign algorithm and hardware?
  - **Inefficient Model Designs** – many CV tasks use large inefficient models and operations solely optimized for accuracy
  - **Limited Hardware Resources** – embedded devices have limited compute resources and strict energy and power budgets
  - **Real-time Requirements** – accelerators must guarantee response within certain time constraints
- Goal: codesign **algorithms** and **accelerators** that *satisfy embedded system constraints and fall on the pareto curve of the accuracy-latency tradeoff.*

# Outline

- Motivation
- Codesign for Image Classification
  - Syntegy DiracDeltaNet
- Codesign for Object Detection
  - Deformable Convolution
- Hardware-aware NAS
  - HAO



# Synetgy: Algorithm-hardware Co-design for ConvNet Accelerators on Embedded FPGAs

Yifan Yang<sup>1,2</sup>, **Qijing Huang**<sup>1</sup>, Bichen Wu<sup>1</sup>, Tianjun Zhang<sup>1</sup>, Liang Ma<sup>3</sup>, Giulio Gambardella<sup>4</sup>, Michaela Blott<sup>4</sup>, Luciano Lavagno<sup>3</sup>, Kees Vissers<sup>4</sup>, John Wawrzynek<sup>1</sup>, and Kurt Keutzer<sup>1</sup>

<sup>1</sup>University of California, Berkeley, <sup>2</sup>Tsinghua University,  
<sup>3</sup>Politecnico di Torino, and <sup>4</sup>Xilinx Research Labs



POLITECNICO  
DI TORINO



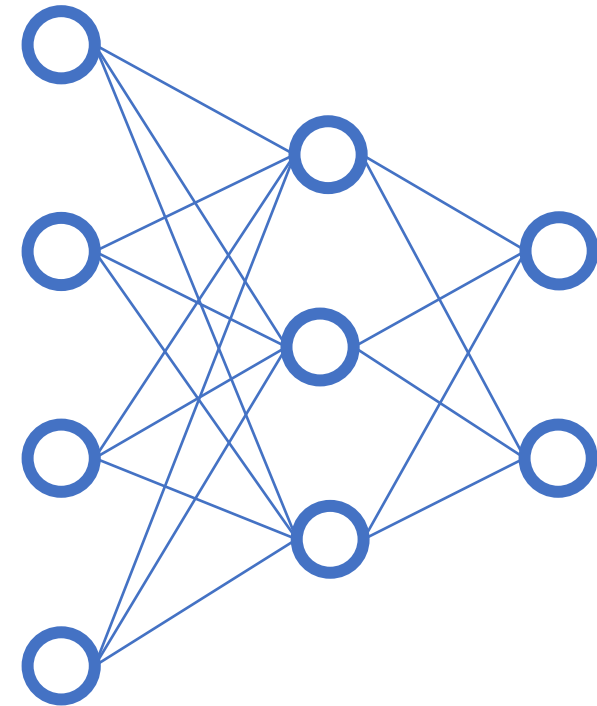
XILINX®

# Codesign for Image Classification

- ConvNet Design
- Hardware Design
- Results

# ConvNet Design Strategies

- Strategy 1: Use efficient models
- Strategy 2: Simplify operators
- Strategy 3: Quantize

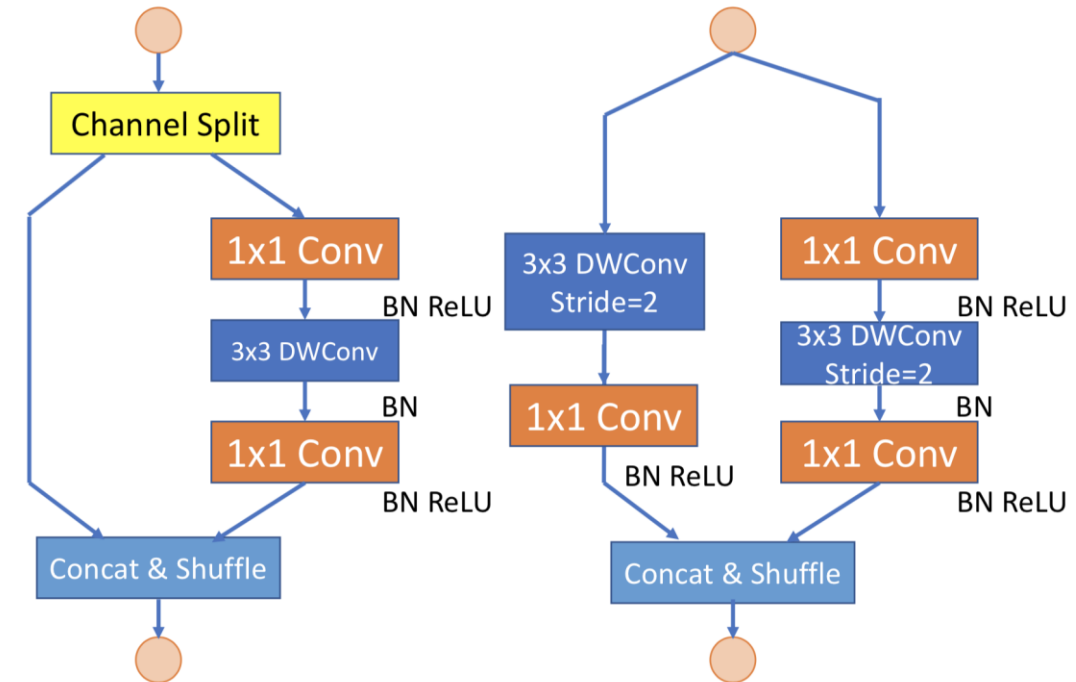


Design better ConvNet

# CNN Strategy 1: Use efficient models

- **ShuffleNetV2-1.0x** [1] as our starting point
- Compared to VGG16:
  - 65x fewer OPs
  - 48x fewer parameters
  - Near equal accuracy on ImageNet

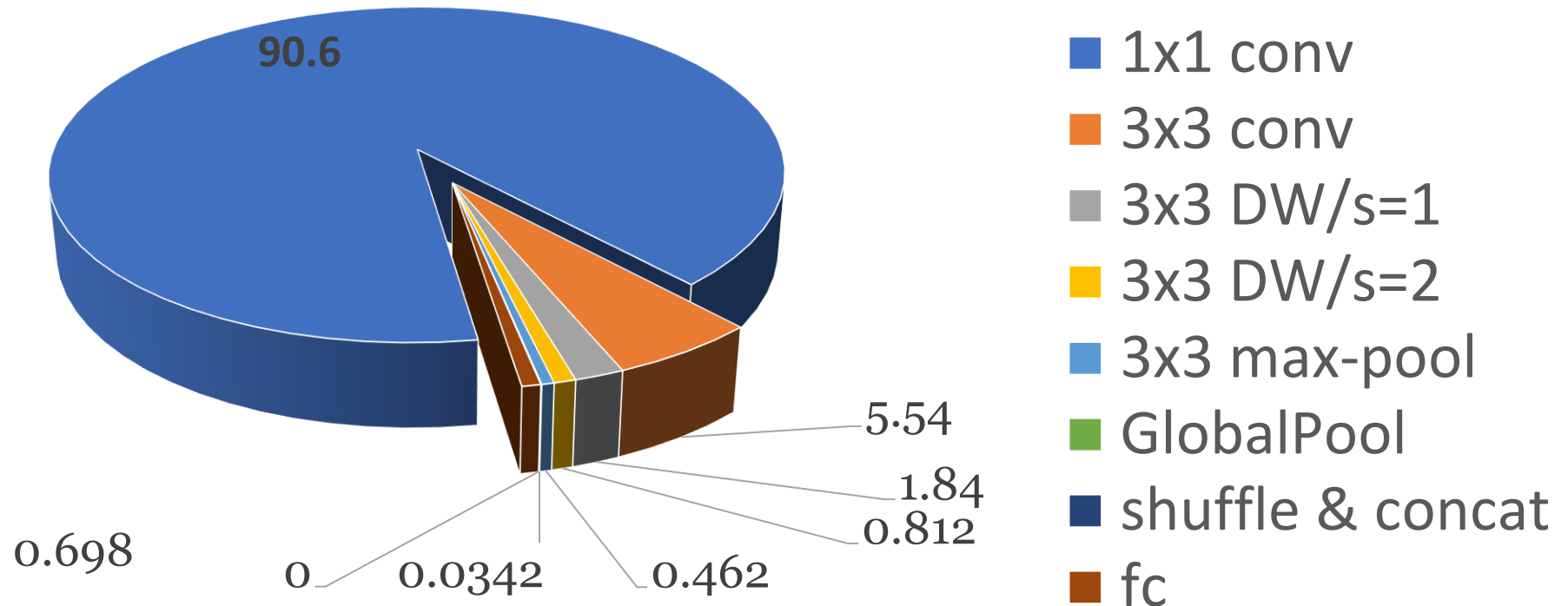
	MACs	#Params	Top-1 Acc
ShuffleNetV2-1.0x	146M	2.3M	69.4%
VGG16	15.3G	138M	71.5%



ShuffleNetV2 building blocks

## CNN Strategy 2: Simplify operators

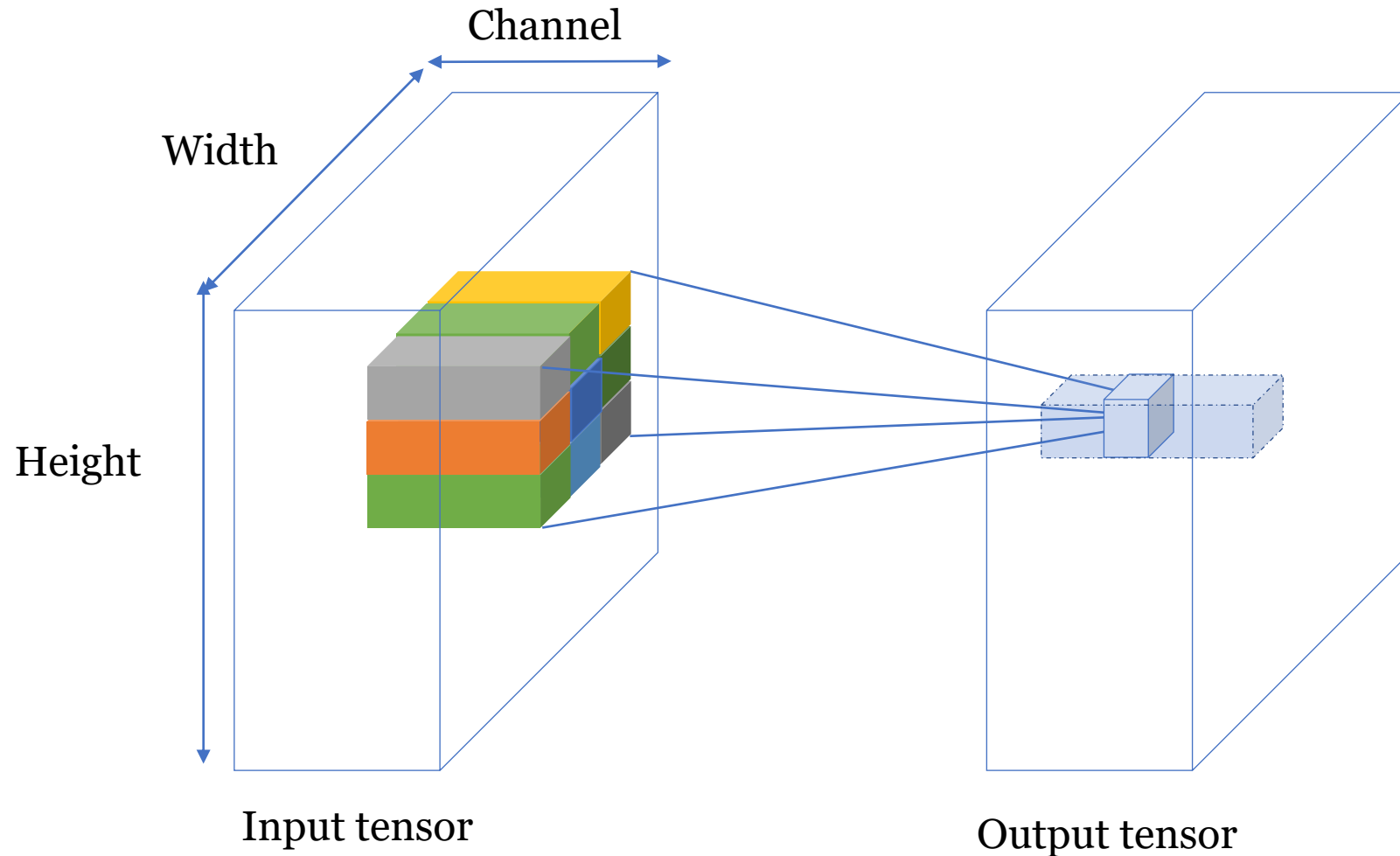
- Can we reduce the number of operator types?
- Can we make the operation more hw-friendly?





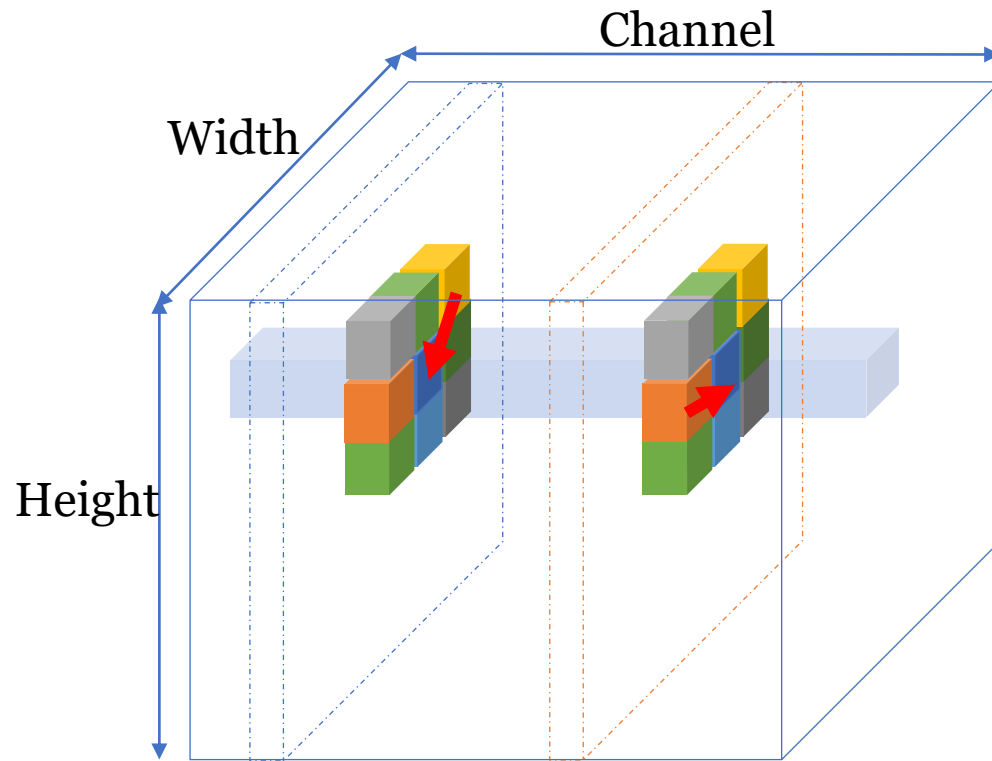
# CNN Strategy 2: Simplify operators

- Can we replace 3x3 convolutions?

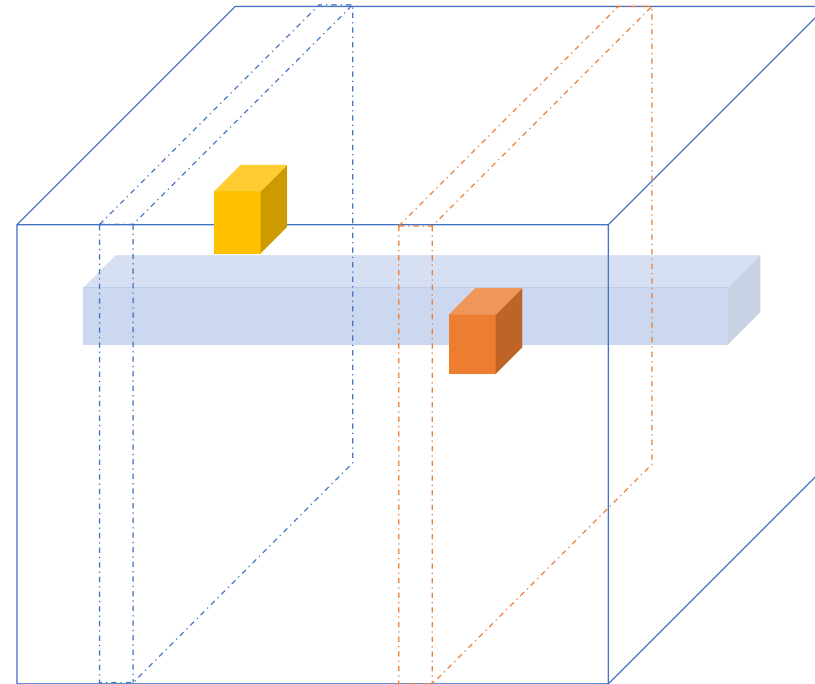


# The Shift Operation

- The shift operation moves a neighboring pixel to the center position



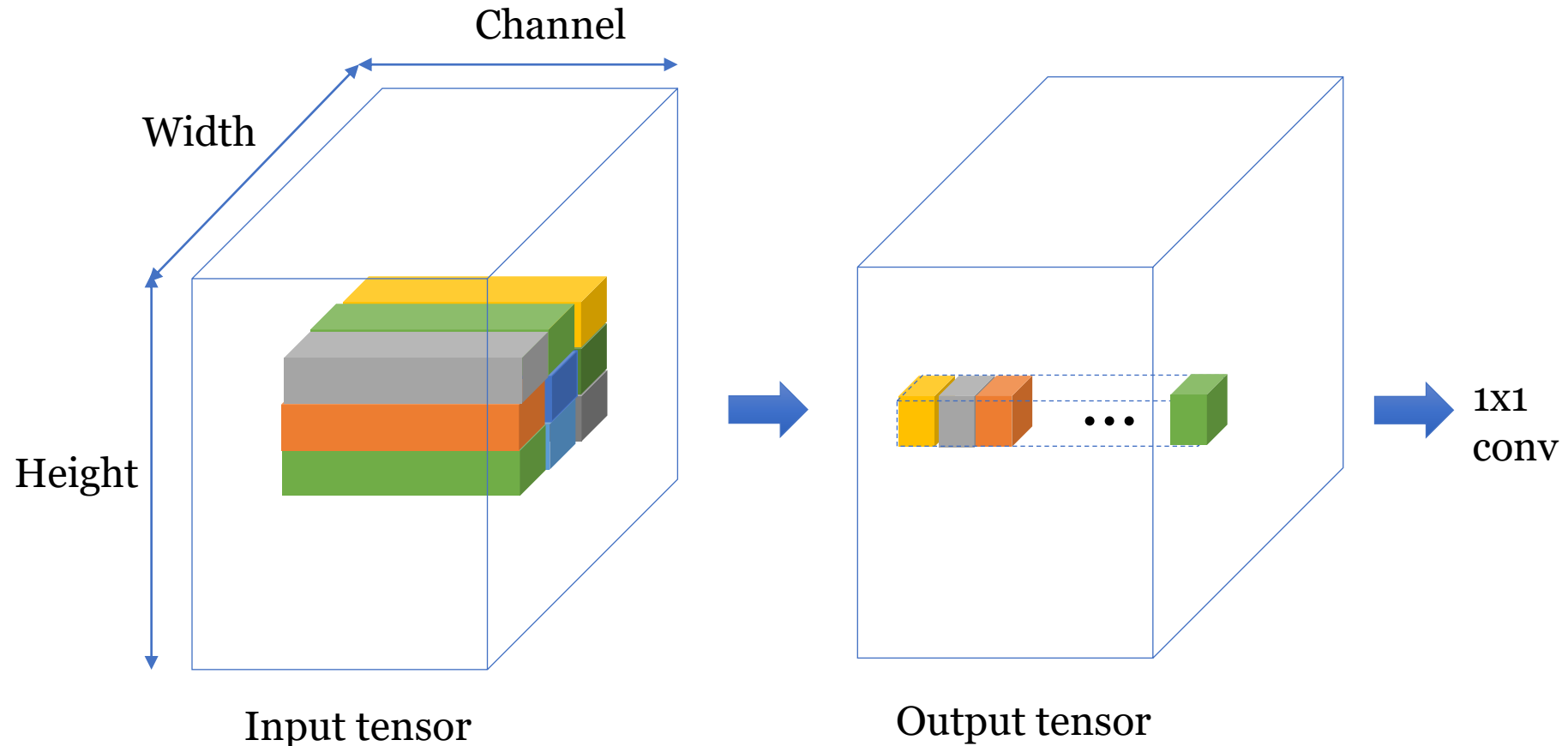
Input tensor



Output tensor

# The Shift Operation

- 1x1 conv aggregates spatial information along the channel dimension

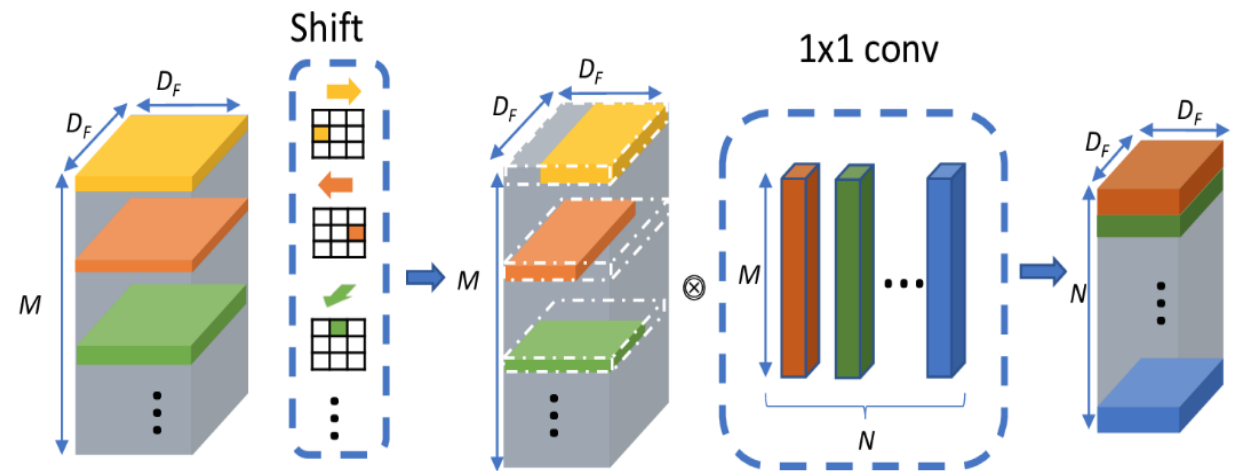
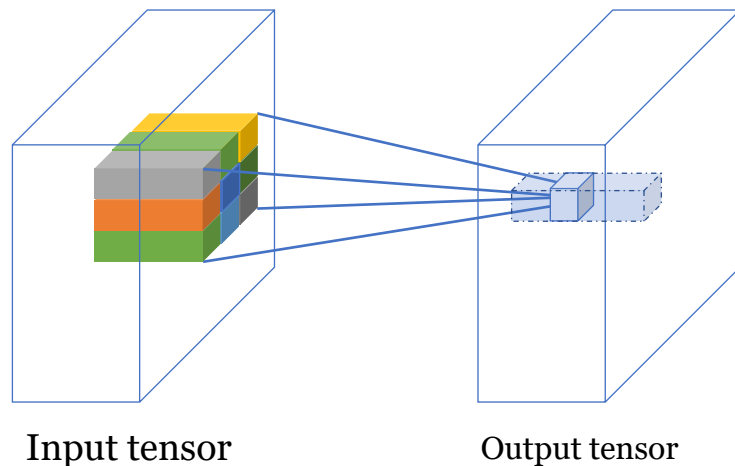


# Replace 3 x 3 Conv

- **3x3 conv:**
  - Aggregates neighboring pixels
  - Mixes channel info

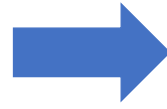


- **shift:** Re-aligns pixels
- **1x1 conv:** Mixes channel info

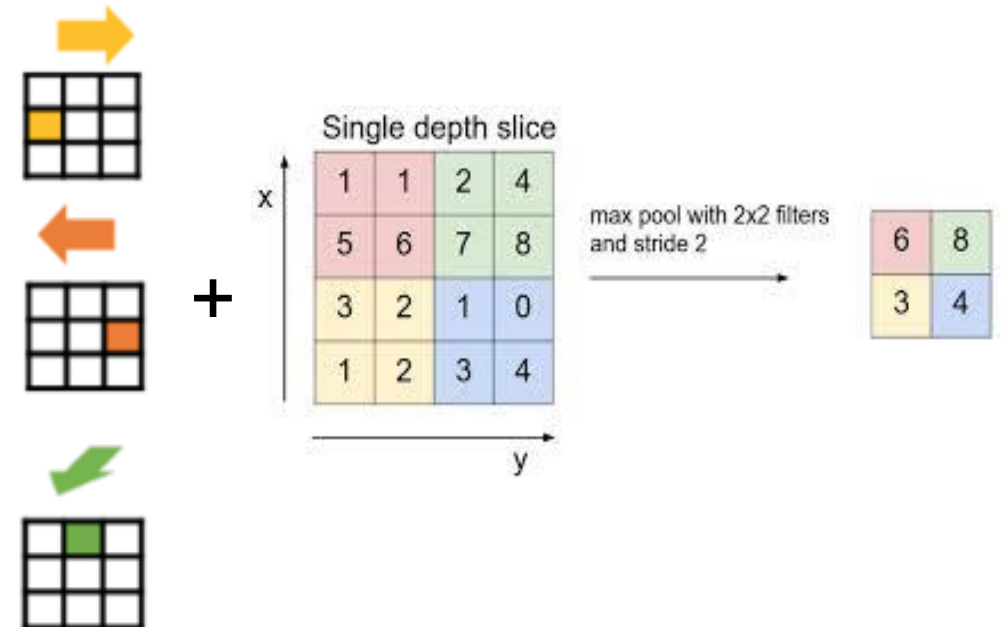
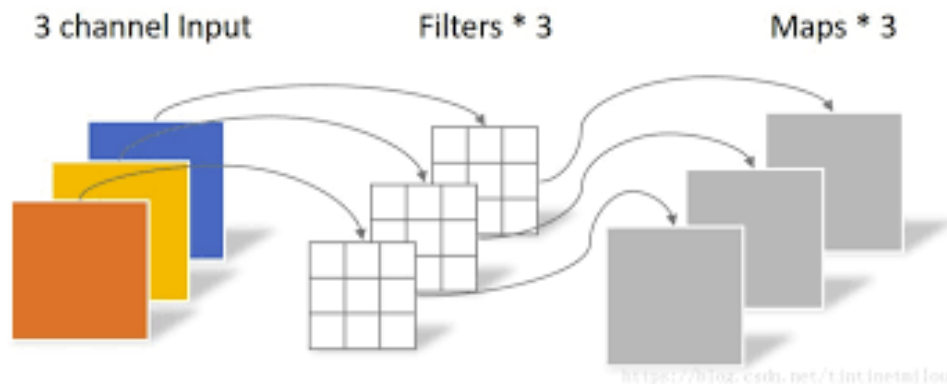


# Replace 3 x 3 DW Conv

- **3x3 DW conv w/ stride 2:**
  - Aggregates neighboring pixels
  - Downsamples

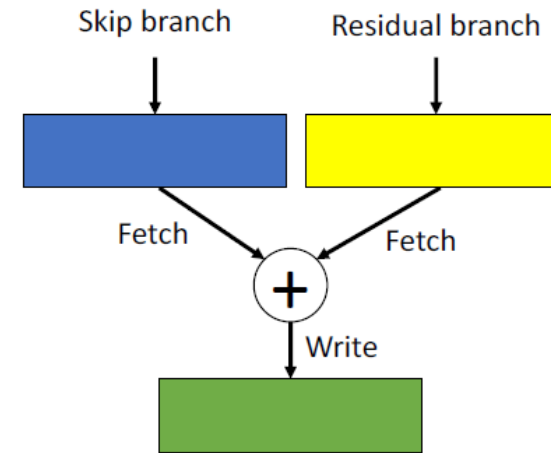


- **shift:** Re-aligns pixels
- **2x2 pooling w/ stride 2:** Downsamples

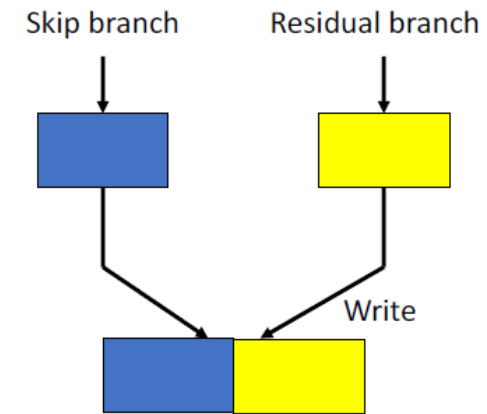


# Strategy 2: Concatenative Connection

- Concatenative skip connection
  - Achieve similar accuracy
  - Less CPU-FPGA data movement
  - Less on-chip synchronization and buffer
  - Quantization friendly



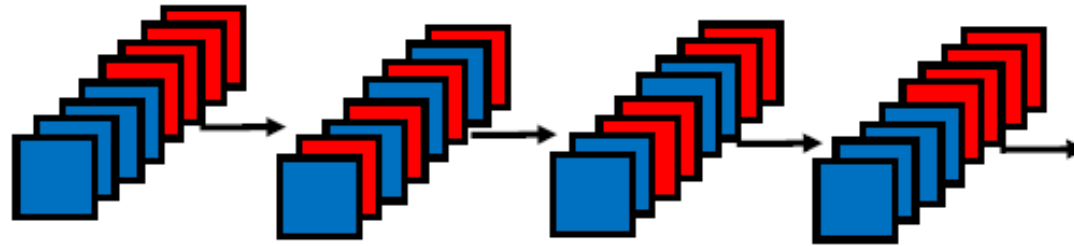
(a) Additive skip connection



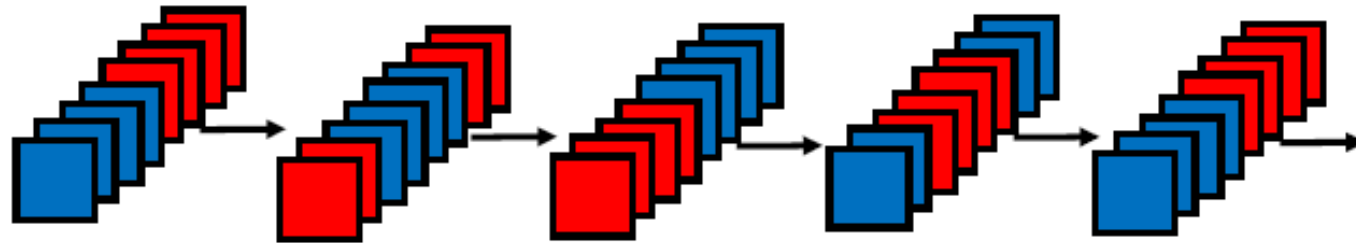
(b) Concatenative skip connection

## Strategy 2: Use hw-friendly operators

- 3x3 max-pooling -> 2x2 max-pooling
- Hw-friendly channel shuffle

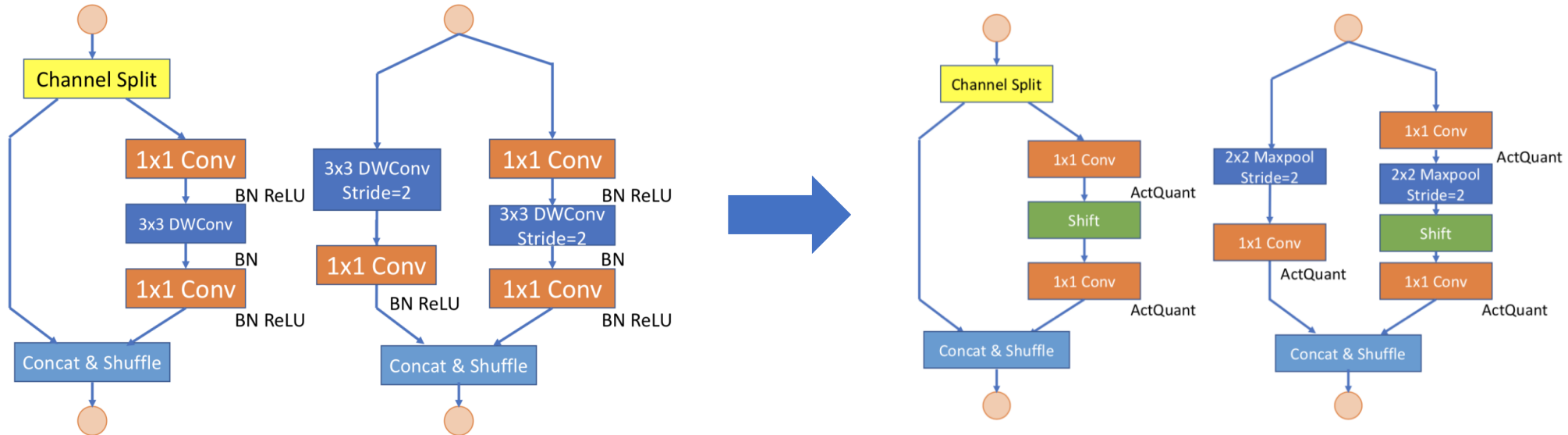


(a) Transpose based channel shuffle



(b) Our channel shuffle

# ShuffleNetV2 -> DiracDeltaNet



- Accuracy (full precision): **69.4%**
- Operators involved:
  - 1x1 convolution
  - 3x3 convolution
  - 3x3 DW convolution
  - 3x3 max pooling
  - Channel split/shuffle/concat

- Accuracy (full precision): **69.7%**
- Operators involved:
  - 1x1 convolution
  - 2x2 max pooling
  - Channel split/shuffle/shift/concat



# CNN Strategy 3: Quantize

- Quantization has been mostly demonstrated on large networks. Is it effective on the small ones like DiracDeltaNet?
- We used existing quantization methods:
  - DoReFaNet [1] method for weights
  - Modified PACT [2] method for activations
- We achieved 4-bit weight and 4-bit activation precision with competitive accuracy

	Network	Pruning	Precision	Top-1 Acc
[3]	VGG16	Yes	8-8b	67.72%
Ours	DiracDeltaNet	No	4-4b	67.52%

[1] Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H. and Zou, Y. DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients.

[2] Choi, J., Wang, Z., Venkataramani, S., I-Jen Chuang, P., Srinivasan, V. and Gopalakrishnan, K. PACT: Parameterized Clipping Activation for Quantized Neural Networks.

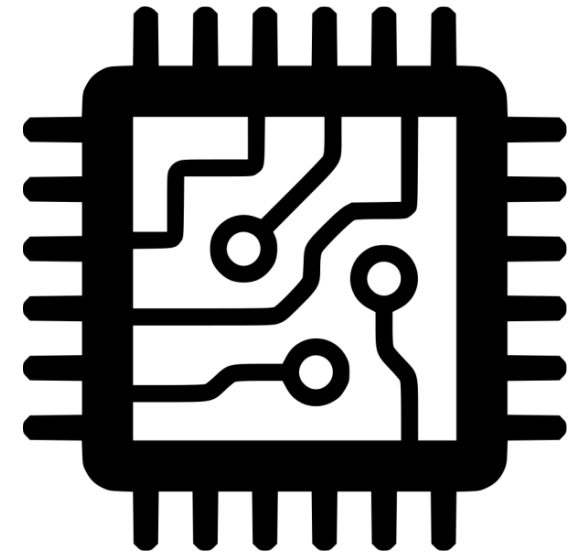
[3] Guo, K., Han, S., Yao, S., Wang, Y., Xie, Y. and Yang, H. Software-Hardware Codesign for Efficient Neural Network Acceleration. IEEE Micro

# Codesign for Image Classification

- ConvNet Design
- Hardware Design
- Results

# Hardware Design Strategies

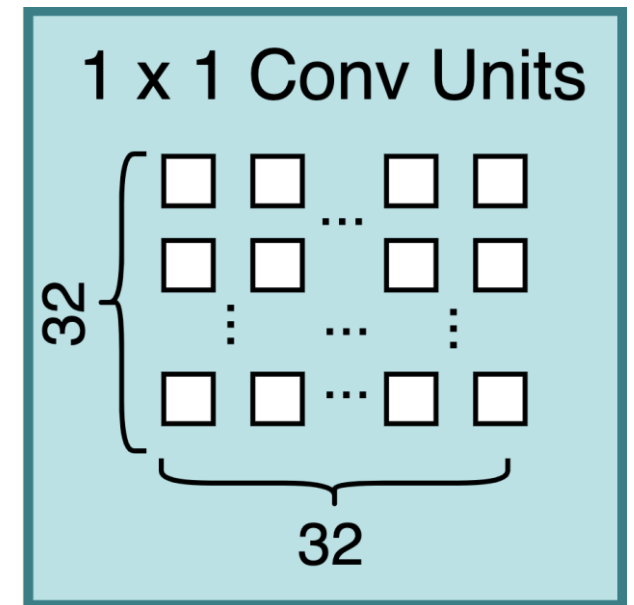
- Strategy 1: Specialize conv engine
- Strategy 2: Use dataflow architecture
- Strategy 3: Merge layers



Design better hardware

# HW Strategy 1: Specialize conv engine

- 1x1 Conv Unit:
  - Supports matrix-vector multiplication
    - 4-bit inputs
    - 4-bit weights
    - 17-bit partial sums
  - Buffers weights and partial sums on-chip
  - Performs 32 x 32 MACs per iteration
  - Each input gets reused *output channel size* times



# Strategy 2: Use dataflow architecture

- 1x1 conv
  - No line-buffer
- shift
  - 3x3 sliding window,  $ll=1$
- 2x2 max-pooling
  - 2x2 sliding window,  $ll=2$

**2x2 max-pooling  
example:**

4	2	5	6	9
1	3	8	7	3
6	4	2	8	1

## Strategy 2: Use dataflow architecture

- 1x1 conv
  - No line-buffer
- shift
  - 3x3 sliding window,  $ll=1$
- 2x2 max-pooling
  - 2x2 sliding window,  $ll=2$

**2x2 max-pooling  
example:**

4	2	5	6	9
1	3	8	7	3
6	4	2	8	1
4				

## Strategy 2: Use dataflow architecture

- 1x1 conv
  - No line-buffer
- shift
  - 3x3 sliding window,  $ll=1$
- 2x2 max-pooling
  - 2x2 sliding window,  $ll=2$

**2x2 max-pooling  
example:**

4	2	5	6	9
1	3	8	7	3
6	4	2	8	1

4			
---	--	--	--

## Strategy 2: Use dataflow architecture

- 1x1 conv
  - No line-buffer
- shift
  - 3x3 sliding window,  $ll=1$
- 2x2 max-pooling
  - 2x2 sliding window,  $ll=2$

**2x2 max-pooling  
example:**

4	2	5	6	9
1	3	8	7	3
6	4	2	8	1

4			
---	--	--	--



## Strategy 2: Use dataflow architecture

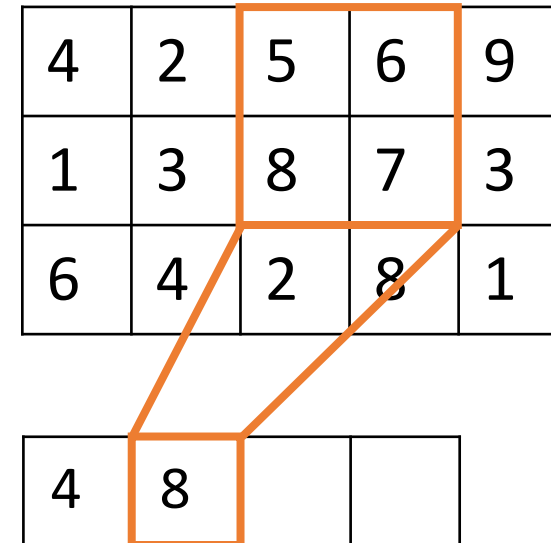
- 1x1 conv
  - No line-buffer
- shift
  - 3x3 sliding window,  $ll=1$
- 2x2 max-pooling
  - 2x2 sliding window,  $ll=2$

**2x2 max-pooling  
example:**

4	2	5	6	9
1	3	8	7	3
6	4	2	8	1

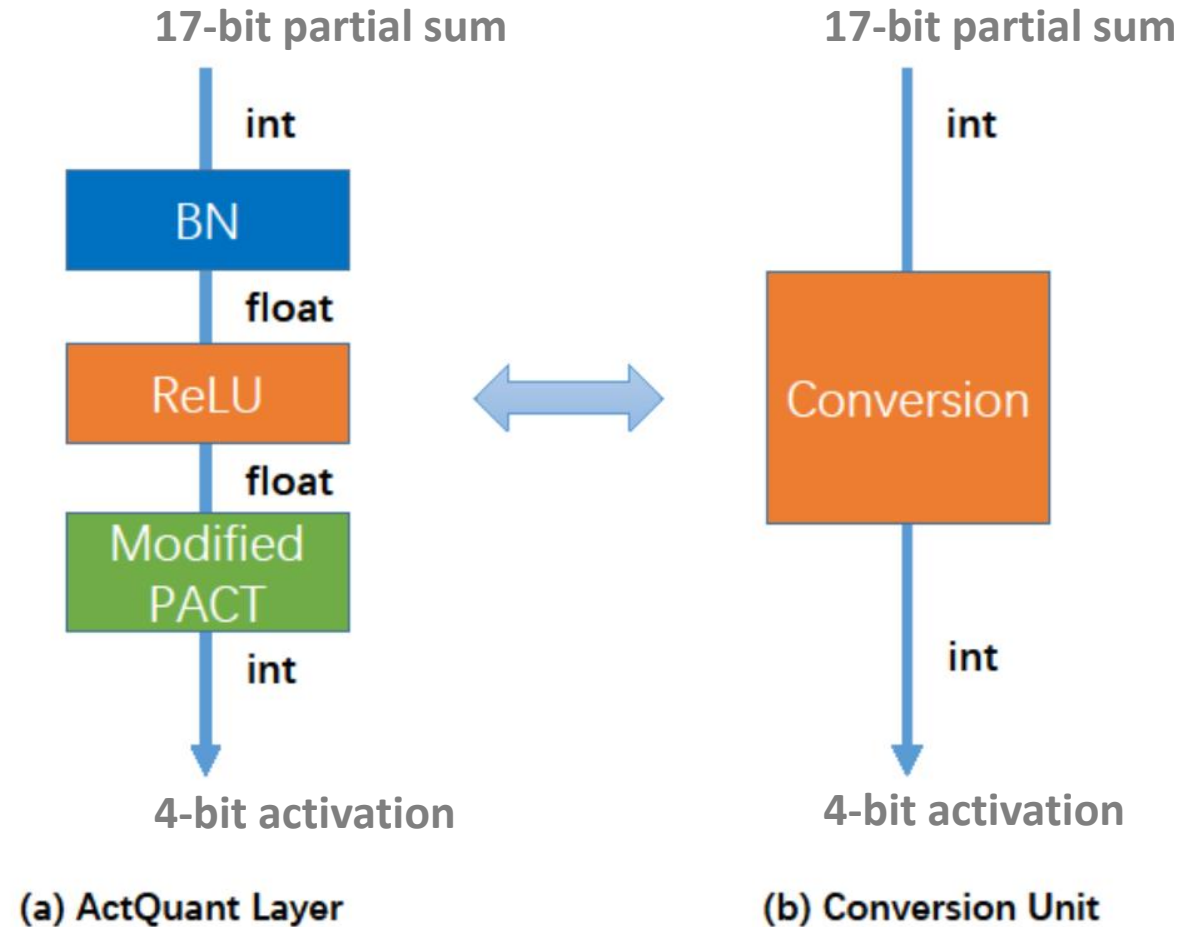
  

4	8		
---	---	--	--



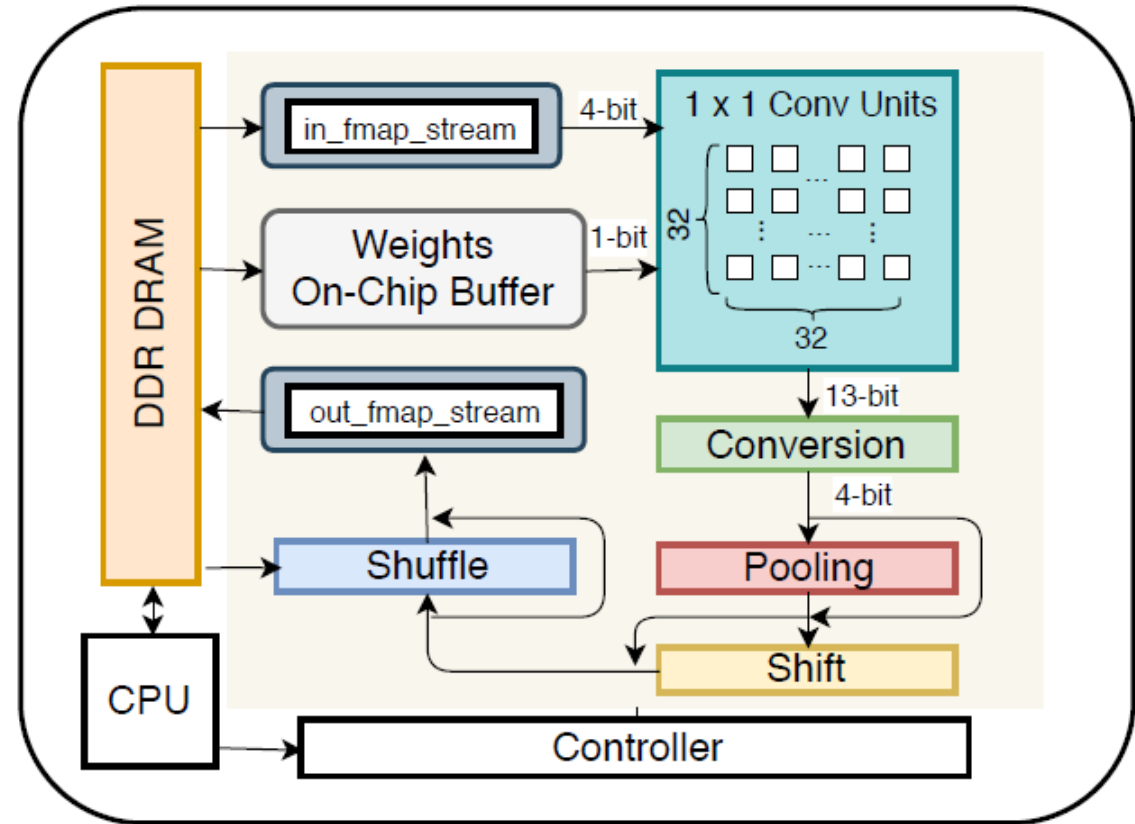
# Strategy 3: Merge layers

- Conversion unit includes:
  - Batch Norm
  - ReLU
  - Modified PACT
- It performs 17-bit to 4-bit conversion
- It is implemented with comparators

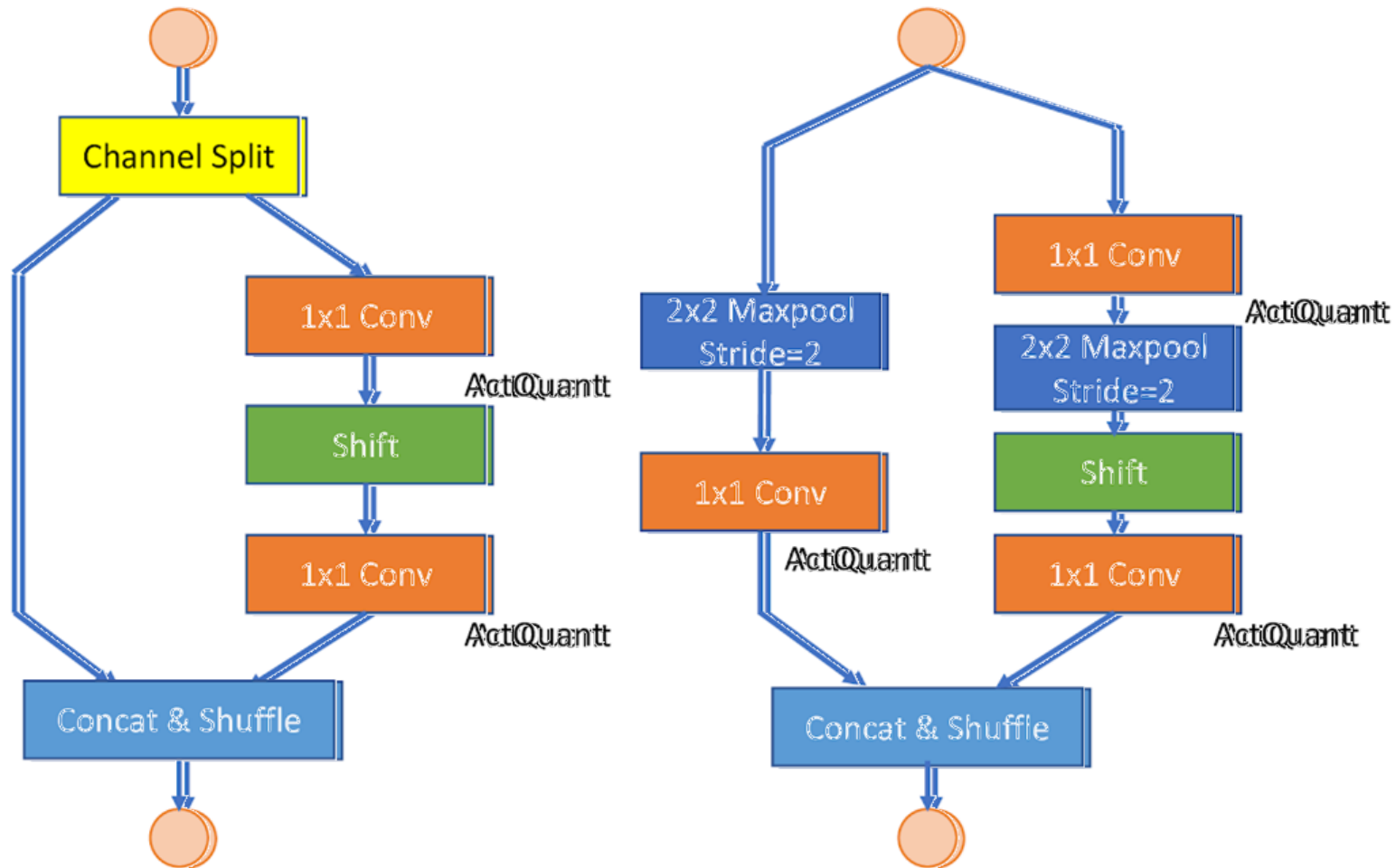


# Synetgy Architecture Overview

- HW engine supports:
  - 1x1 conv
  - 2x2 max-pooling
  - shift
  - shuffle
- Layer-based design
- Implemented with Vivado HLS and PYNQ

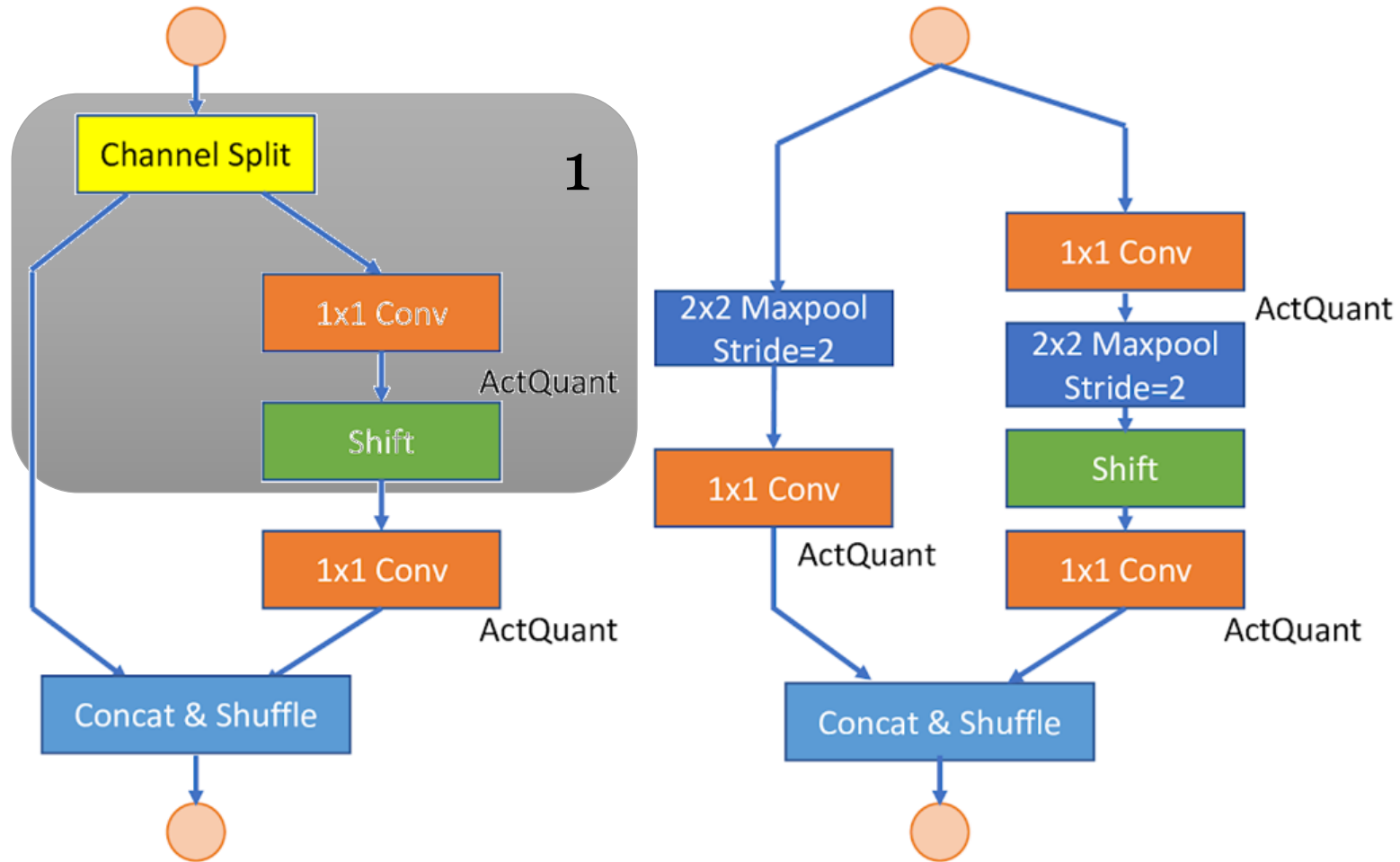


# Execution Model



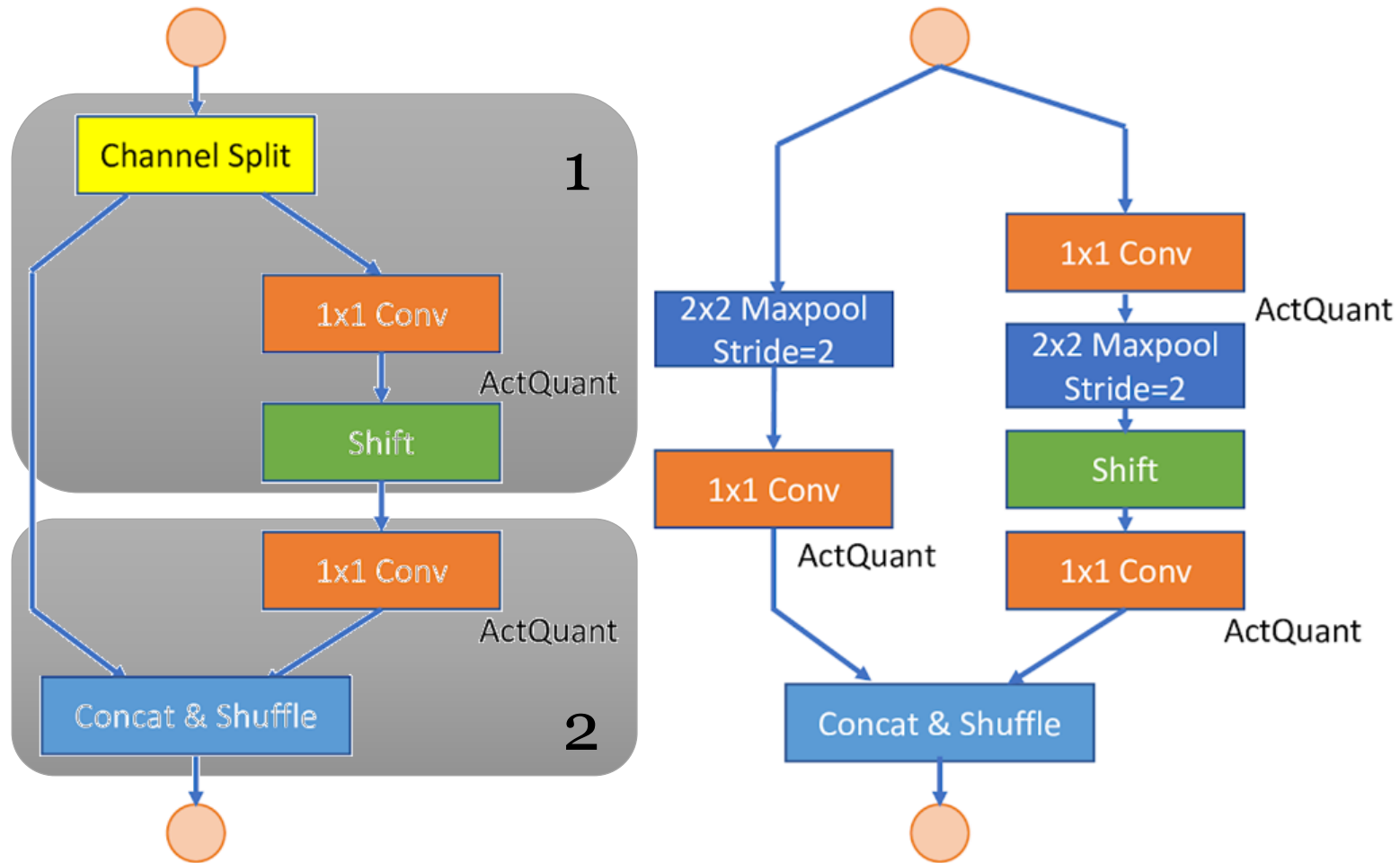
DiracDeltaNet Block

# Execution Model



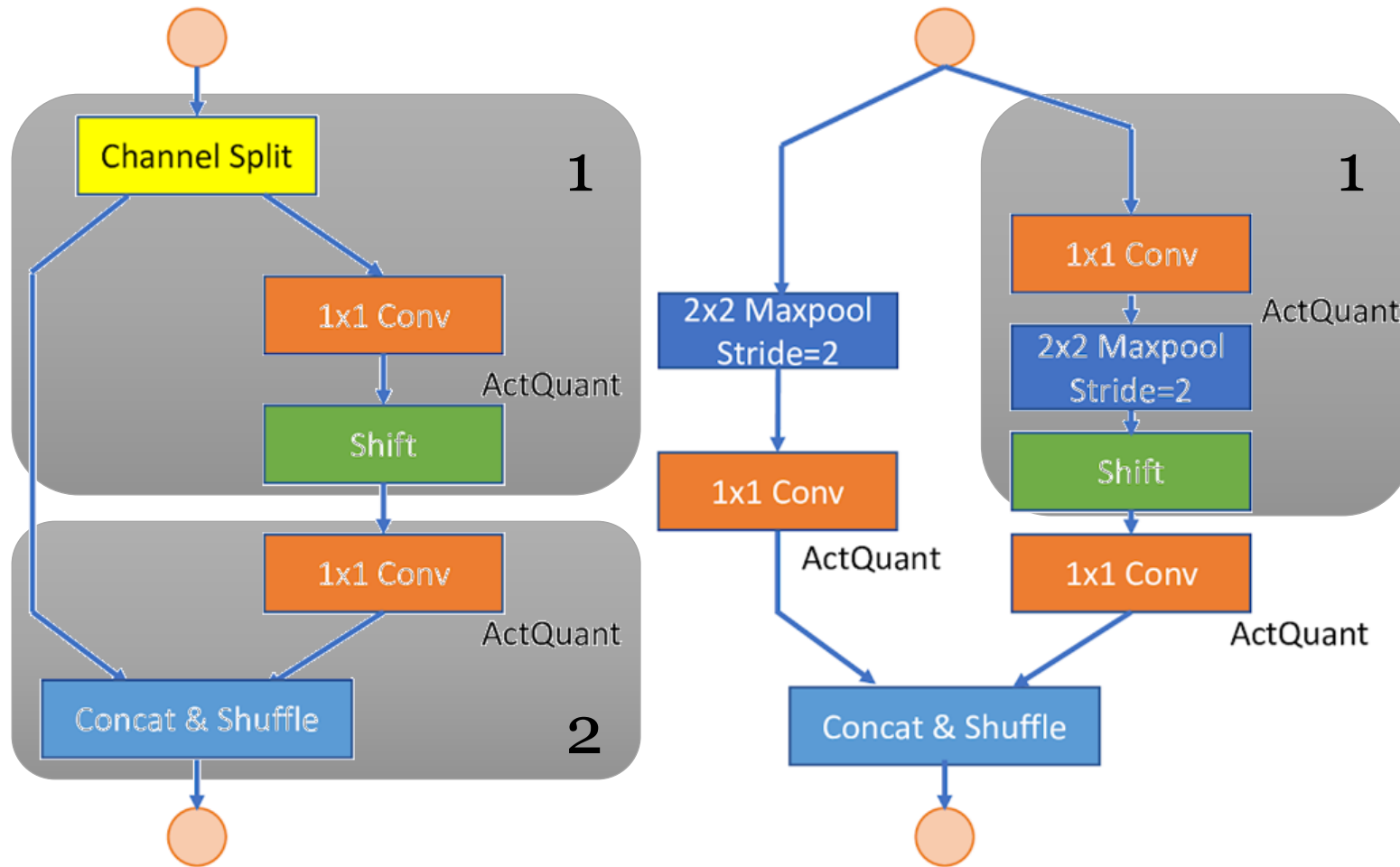
DiracDeltaNet Block

# Execution Model



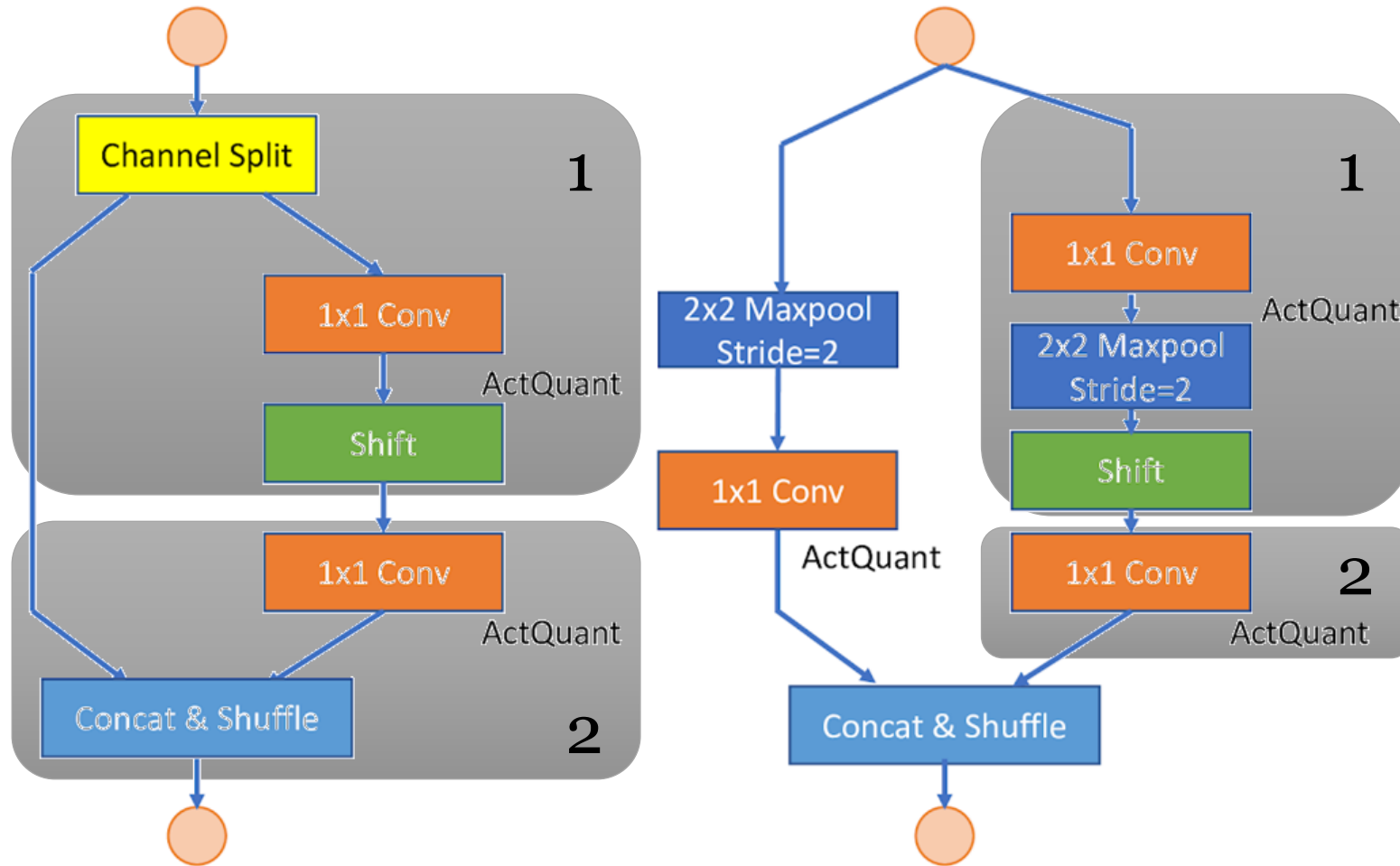
DiracDeltaNet Block

# Execution Model



DiracDeltaNet Block

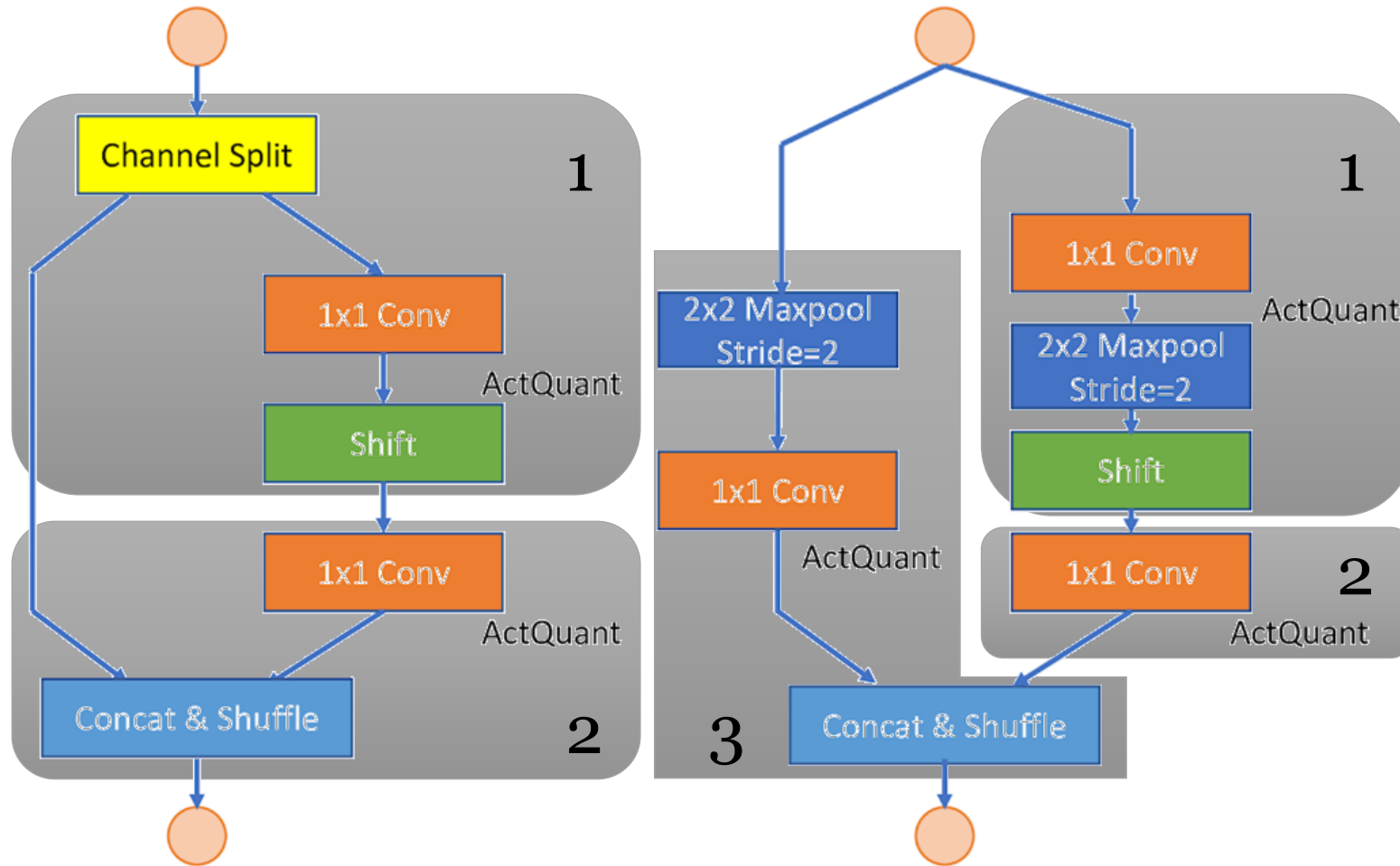
# Execution Model



DiracDeltaNet Block



# Execution Model



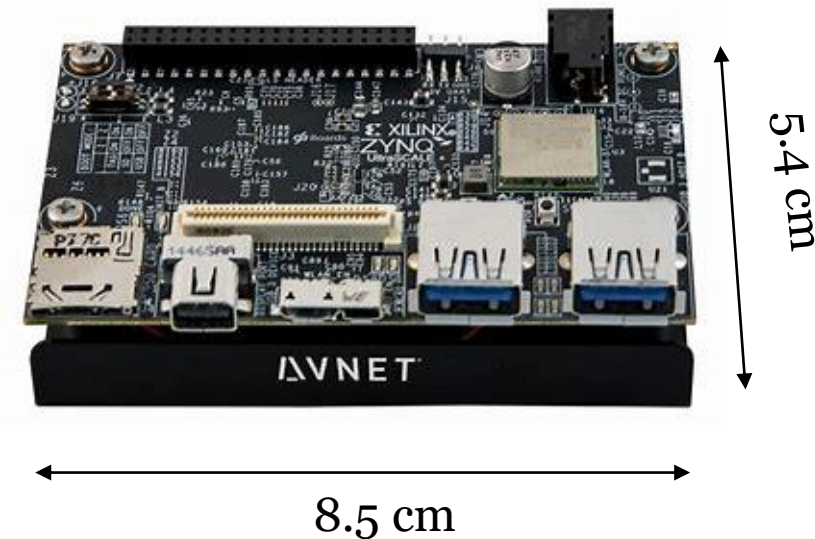
DiracDeltaNet Block

# Codesign for Image Classification

- ConvNet Design
- Hardware Design
- Results

# Experimental Setup

- Avnet Ultra96 Board
- With Xilinx ZU3EG FPGA – the second smallest device in the Ultrascale+ family



# Comparison with Previous Work

					Energy/
[	Algorithm-hardware co-design can achieve both high accuracy (67.5% top-1) and good efficiency (66 FPS) for embedded CV applications				
[					
[					
Ours	Zynq ZU3EG	66.3	67.52%	4-4b	0.083

- Equal top-1 accuracy
- 11.6x higher framerate
- 6.3x more power efficient

[1] Guo, K., Han, S., Yao, S., Wang, Y., Xie, Y. and Yang, H. Software-Hardware Codesign for Efficient Neural Network Acceleration. IEEE Micro, 37 (2). 18-25.

[2] Qiu, J., Wang, J., Yao, S., Guo, K., Li, B., Zhou, E., Yu, J., Tang, T., Xu, N., Song, S., Wang, Y. and Yang, H. Going Deeper with Embedded FPGA Platform for Convolutional Neural Network, 2016, 26-35.

[3] Suda, N., Chandra, V., Dasika, G., Mohanty, A., Ma, Y., Vrudhula, S.B.K., Seo, J.S. and Cao, Y. Throughput-Optimized OpenCL-based FPGA Accelerator for Large-Scale Convolutional Neural Networks, 2016, 16-25.

# Outline

- Motivation
- Codesign for Image Classification
  - Synergy DiracDeltaNet
- Codesign for Object Detection
  - Deformable Convolution
- Hardware-aware NAS
  - HAO



Berkeley DeepDrive



# CoDeNet: Efficient Deployment of Input-Adaptive Object Detection on Embedded FPGAs

**Qijing Huang\***, **Zhen Dong\***, Dequan Wang\*, Yizhao Gao †, Yaohui Cai ‡, Tian Li ‡,  
Bichen Wu, Kurt Keutzer, John Wawrzynek

University of California, Berkeley,

†University of Chinese Academy of Science

‡Peking University



Berkeley  
UNIVERSITY OF CALIFORNIA

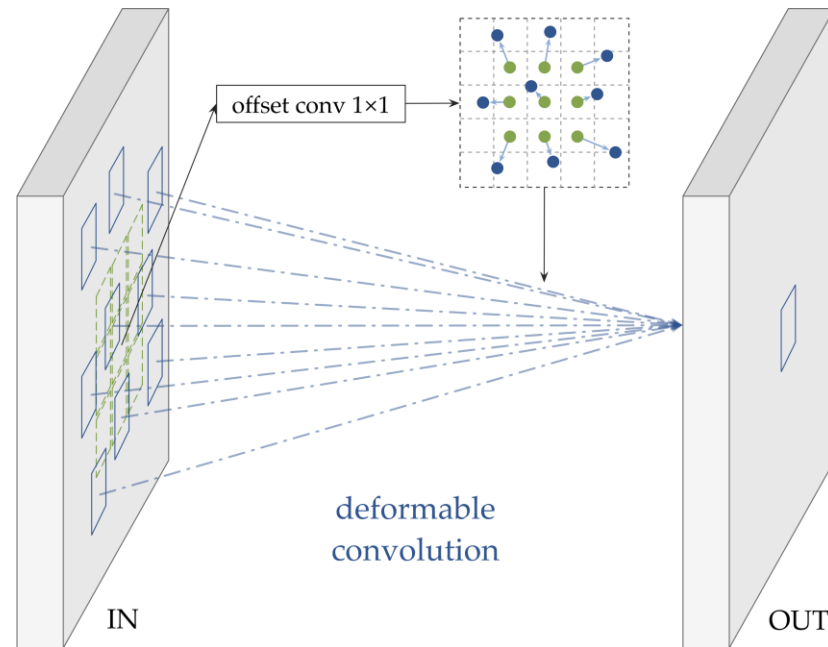


# Codesign for Objection Detection

- **Deformable Convolution**
- Operation Codesign
- Detection System Codesign
- Results

# Deformable Convolution

- **Deformable Convolution** is an input-adaptive dynamic operation that samples inputs from variable spatial locations



1. Generate offsets
2. Sample from input feature map



# Deformable Convolution

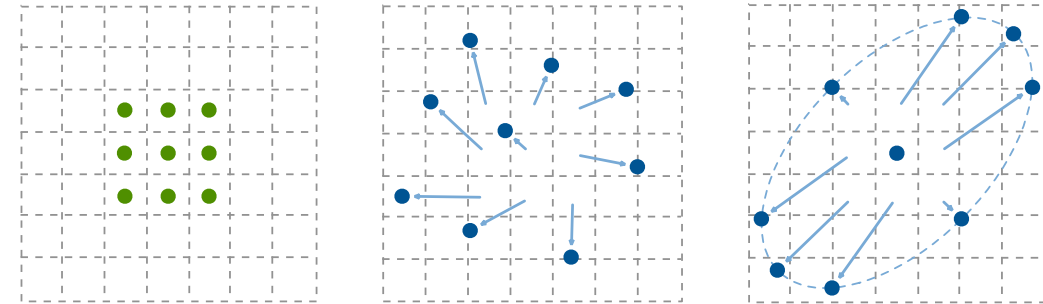
- **Deformable Convolution** is an input-adaptive dynamic operation that samples inputs from variable spatial locations
- Its sampling locations vary with:
  - Different input images
  - Different output pixel locations



Sampling Locations (in red) for Different Output Pixels (in green)

# Deformable Convolution

- **Deformable Convolution** is an input-adaptive dynamic operation that samples inputs from variable spatial locations
- Its sampling locations vary with:
  - Different input images
  - Different output pixel locations
- It captures the spatial variance of objects with different:
  - Scales
  - Aspect Ratios
  - Rotation Angles
- Challenges:
  - Increased compute and memory requirements
  - Irregular input-dependent memory access patterns
    - Not friendly for dataflows that leverage the spatial reuse



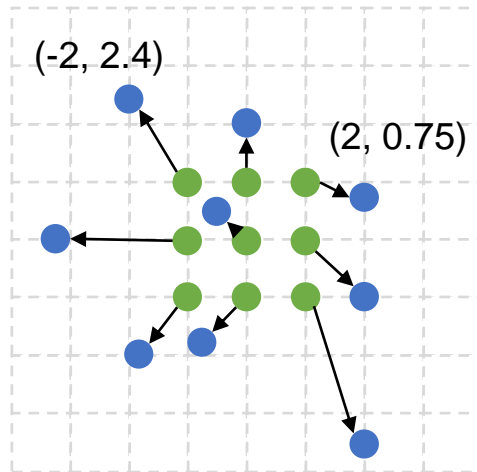
Variable Receptive Fields

# Codesign for Objection Detection

- Deformable Convolution
- Operation Codesign
- Detection System Codesign
- Results

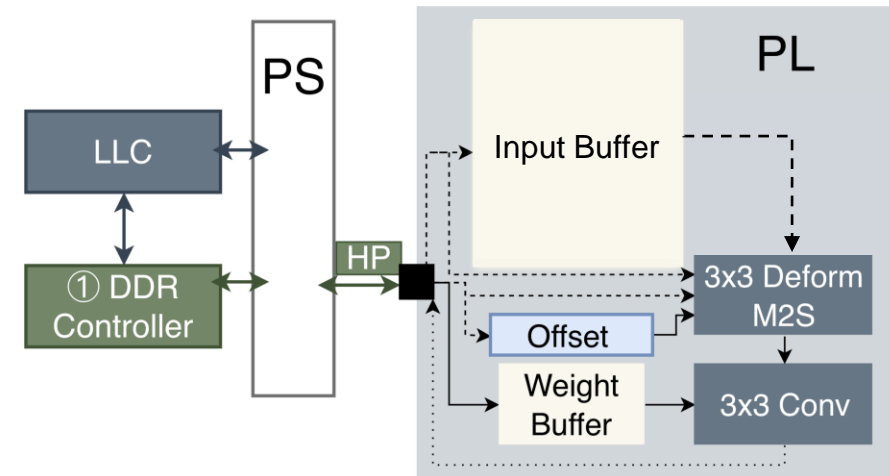
# Operation Codesign

## Algorithm Modification:



0. Original Deformable

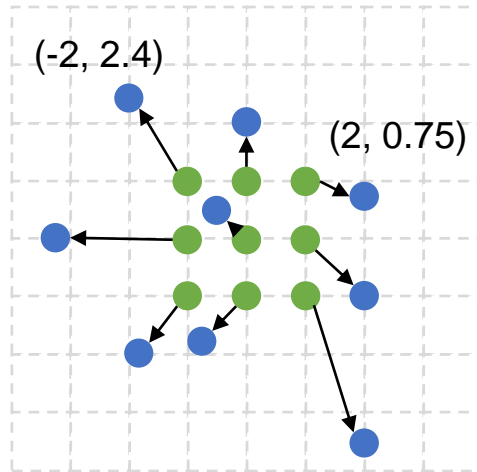
## Hardware Optimization:



- **Preloads weights to on-chip buffer**
- **Loads input and offsets directly from DRAM**

# Operation Codesign

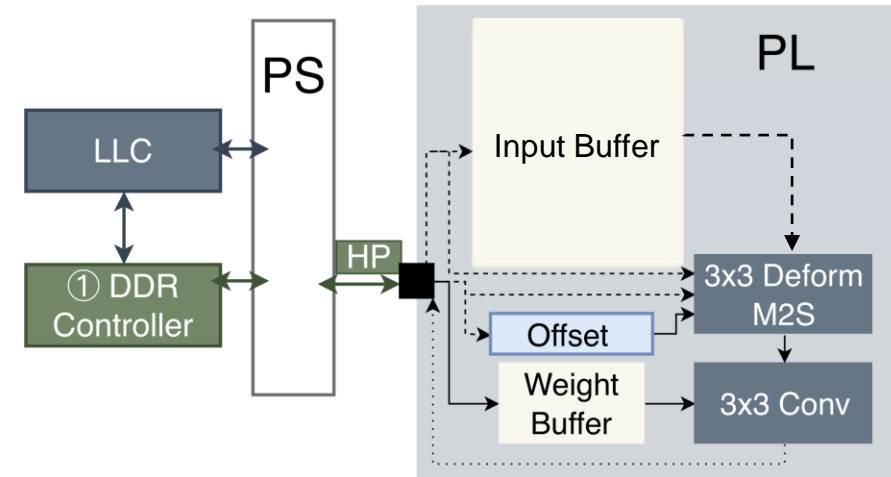
## Algorithm Modification:



### 1. Depthwise Deformable

Accuracy<sup>1</sup>(AP): 42.9

## Hardware Optimization:

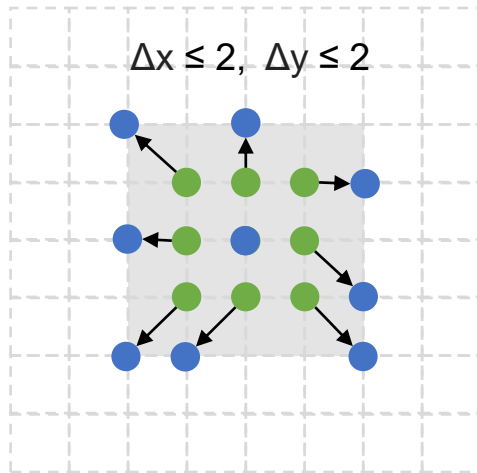


- Reduce the total MACs

<sup>1</sup> Accuracy for Object Detection on Pascal VOC

# Operation Codesign

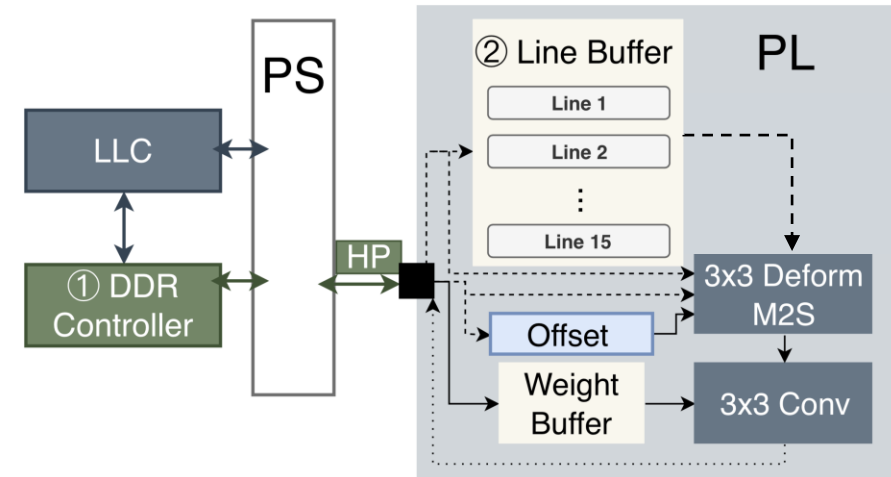
Algorithm Modification:



**2. Bounded Range**

Accuracy<sup>1</sup>(AP): **41.0** ↓ **1.9**

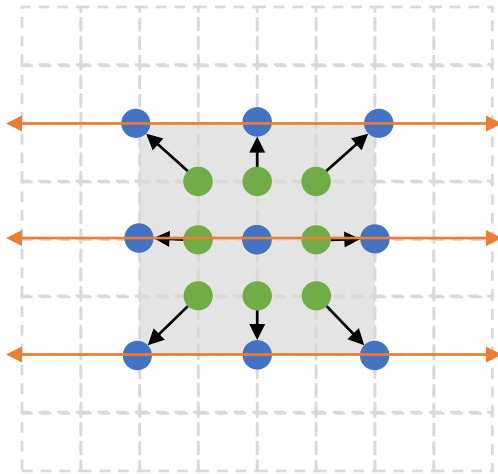
Hardware Optimization:



- **Buffers inputs in the on-chip line buffer to allow spatial reuse**

# Operation Codesign

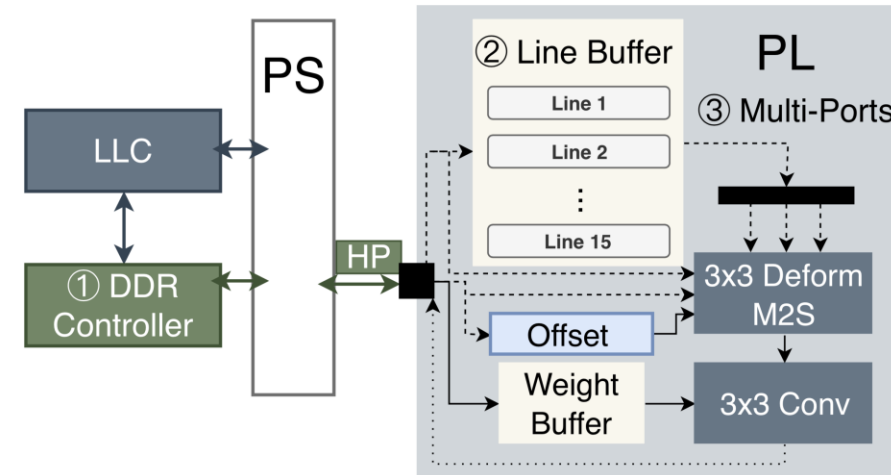
Algorithm Modification:



**3. Rectangular Shape**

Accuracy<sup>1</sup>(AP): 41.1 ↑ 0.1

Hardware Optimization:



- Improves on-chip memory bandwidth

# Operation Accuracy

## Object Detection Accuracies

Operation	Depthwise	Bound	Square	VOC			COCO					
				AP	AP50	AP75	AP	AP50	AP75	APs	APm	API
$3 \times 3$				39.2	60.8	41.2	21.4	36.5	21.5	7.3	24.1	33.0
$3 \times 3$	✓			39.1	60.9	40.9	19.8	34.3	19.7	6.3	22.6	31.5
$5 \times 5$	✓			40.6	62.4	42.6	21.3	36.4	21.3	6.7	23.7	34.2
$7 \times 7$	✓			41.9	63.8	43.8	21.7	37.2	21.5	6.9	24.0	35.2
$9 \times 9$	✓			42.3	64.8	44.3						
deform	✓			42.9	64.4	45.7	23.0	38.4	23.3	6.9	24.4	37.8
deform	✓	✓		41.0	63.0	42.9	21.3	36.4	21.1	7.2	23.6	34.4
deform	✓	✓	✓	41.1	63.1	43.7	21.5	36.8	21.5	6.5	23.7	34.8

5x less  
compute

< 2 AP  
change

- 3x3 deformable conv is more efficient than large convolution kernels
- The codesigned deformable conv still achieves good accuracy
  - 1.8 AP difference on Pascal VoC and 1.5 AP difference on COCO



# Operation Performance

## Hardware Performance

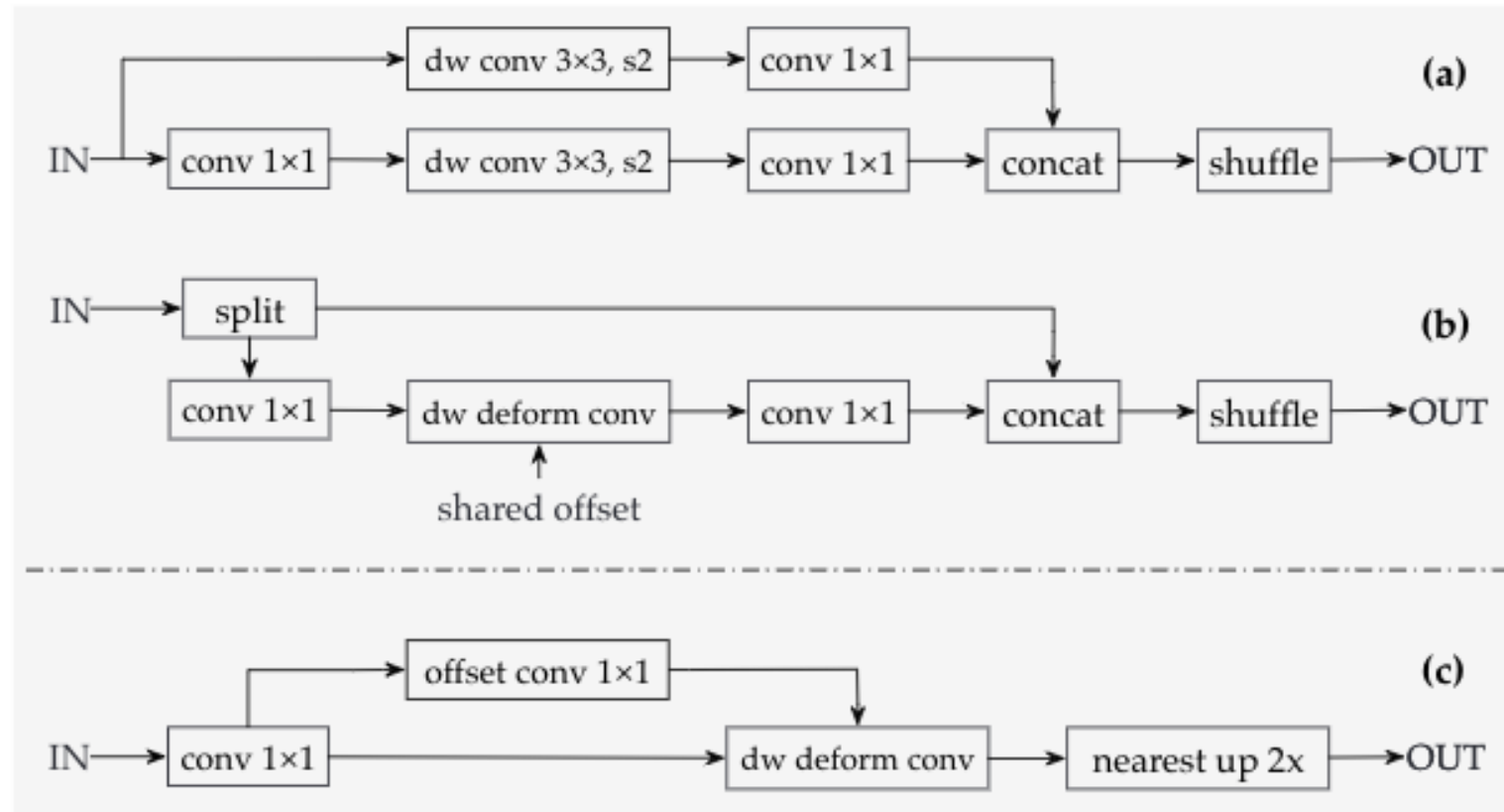
Operation	Original	Deformable	Bound (buffered)	Square (multi-ported)	Without LLC		With LLC	
					Latency (ms)	GOPs	Latency (ms)	GOPs
Full 3×3 Conv	✓	✓	✓	1.36x	43.1	112.0	41.6	116.2
					59.0	81.8	42.7	113.1
					43.4	111.5	41.8	115.5
					43.4	111.5	41.8	115.6
Depthwise 3×3 Conv	✓	✓	✓	9.76x	1.9	9.7	2.0	9.6
					20.5	0.9	17.8	1.1
					3.0	6.2	3.4	5.5
					2.1	9.2	2.3	8.2

**1.36×** and **9.76×** speedup for full and depthwise deformable conv

# Codesign for Objection Detection

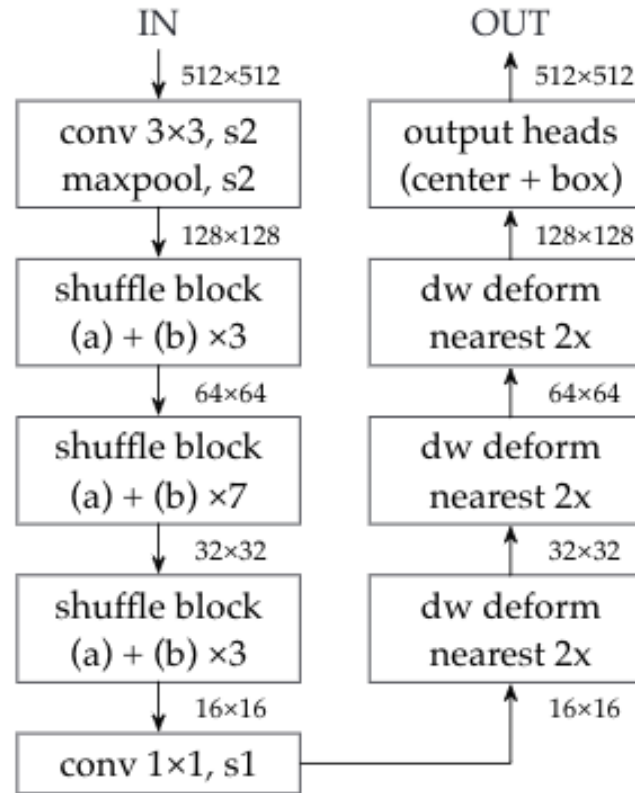
- Deformable Convolution
- Operation Codesign
- Detection System Codesign
- Results

# Building Blocks



- Simple building blocks to reduce the hardware complexity

# ShuffleNetV2 + CenterNet

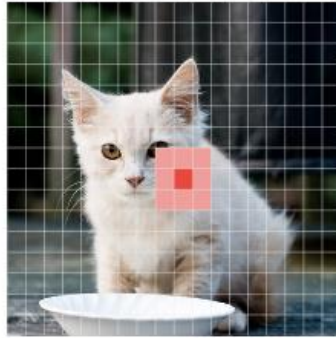


- Anchor-free detection system to reduce the postprocessing overhead for Non Maximum Suppression (NMS)

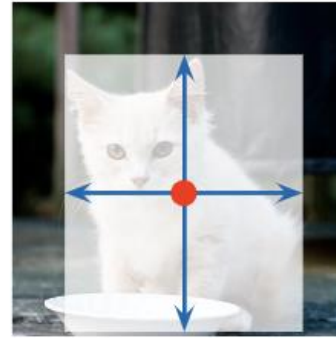
# Detection Heads



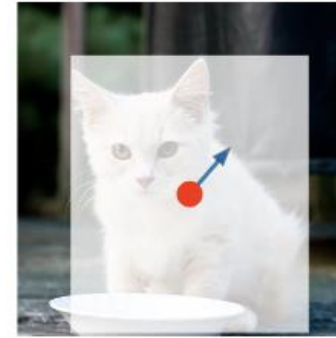
(a) image



(b) center heatmap



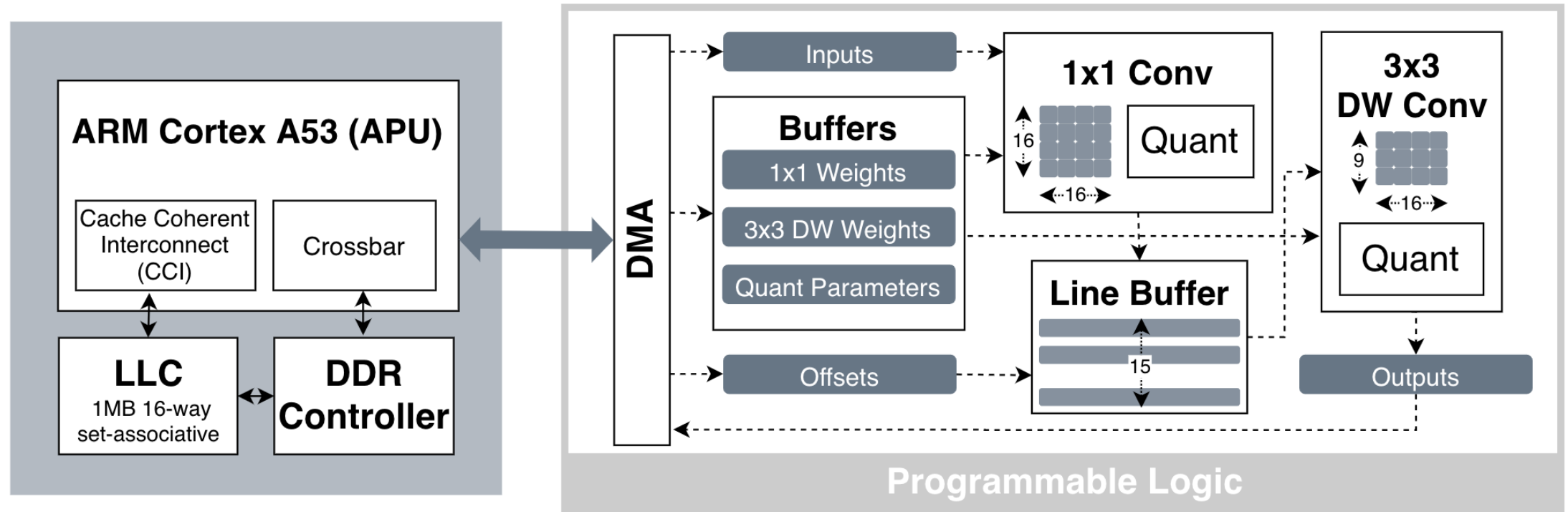
(c) width & height



(d) local shift

1. The center heatmap
2. The object size
3. The local offset

# Overall Accelerator Architecture



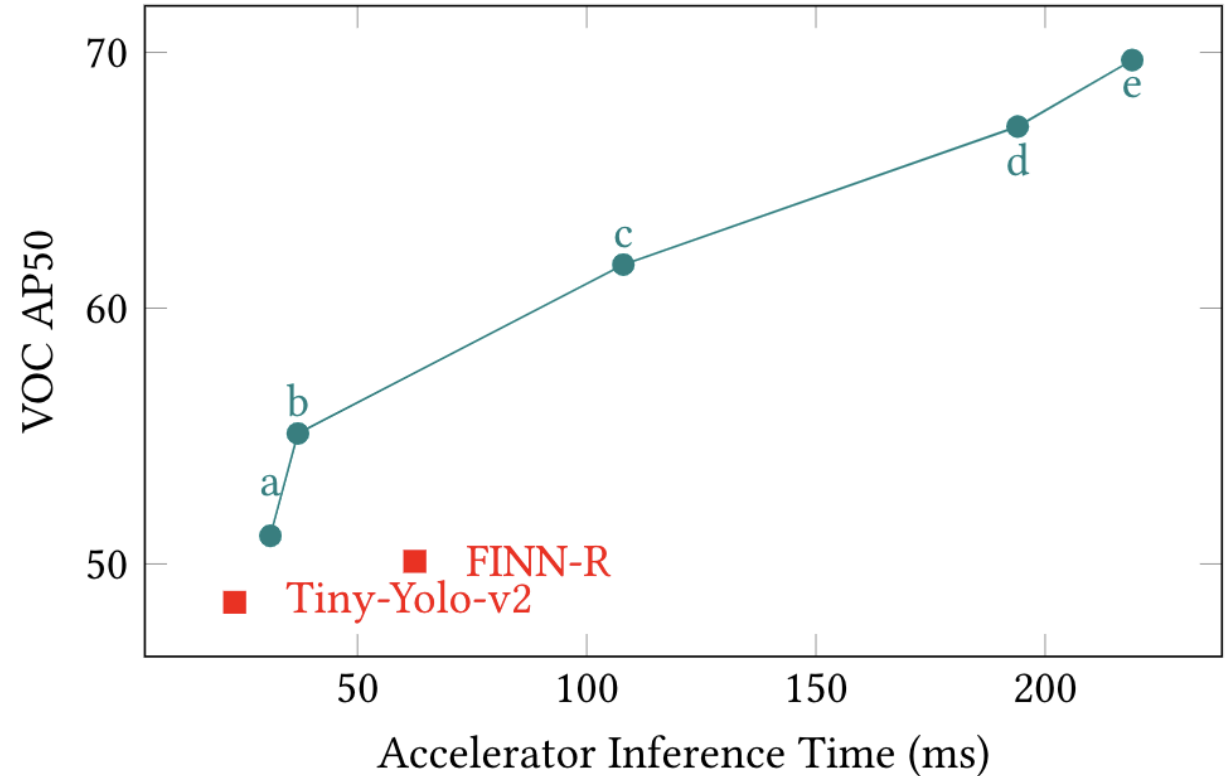
- **Dataflow** architecture for executing a subgraph of *1x1 conv* and *3x3 dw deformable conv*

# Codesign for Objection Detection

- Deformable Convolution
- Operation Codesign
- Detection System Codesign
- Results

# Accuracy-Latency Tradeoff

- 4-bit weights, 8-bit activations
- Batch size 1
- On VOC dataset



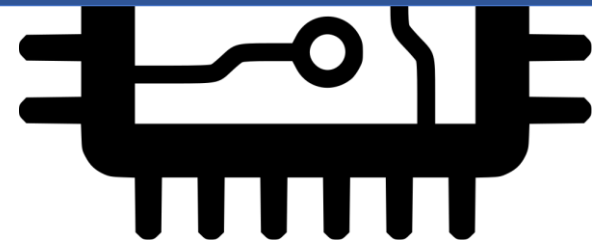


# Can we perform co-design to close this gap?

Algorithm-hardware co-design can achieve both high accuracy and good efficiency for embedded CV applications



Design better ConvNet



Design better hardware

# Can we replace the Conv operation?

- **Binary operation**
  - [“XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks”](#) M Rastegari, et al. 2016 ECCV.
- **Look up tables**
  - [“Circuit-Based Intrinsic Methods to Detect Overfitting.”](#) Chatterjee S, Mishchenko A. 2020 ICML.
  - [“LUTNet: Rethinking Inference in FPGA Soft Logic”](#) Wang E, et al. 2019 FCCM.
- **Log number systems (LNS)**
  - [“Low-Precision Training in Logarithmic Number System using Multiplicative Weight Update”](#) J Zhao, et al. Arxiv.

# Outline

- Motivation
- Codesign for Image Classification
  - Synergy DiracDeltaNet
- Codesign for Object Detection
  - Deformable Convolution
- Hardware-aware NAS
  - HAO

Thanks!