A High-performance CNN Processor Based on FPGA for MobileNets

Di Wu, Yu Zhang, Xijie Jia, Lu Tian, Tianping Li, Lingzhi Sui, Dongliang Xie, Yi Shan
diw@xilinx.com
09/10/2019

XILINX.

Outline

> Introduction

- Depthwise separable convolution
- MobileNetV1 and MobileNetV2

> Our works

- Scalable computation engines
- Pipeline schedule among layers
- Data flow for pipeline
- Channel augmentation

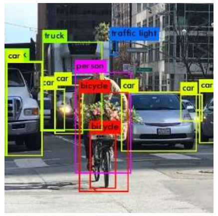
> Experiment results

- Comparison in Classification
- Comparison in Object Detection

XILINX.

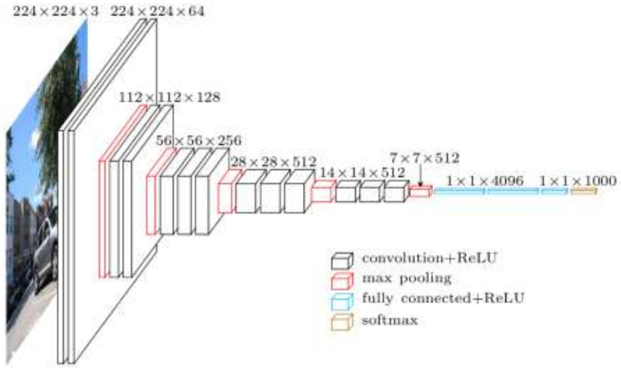
Background

> CNN in computer vision tasks

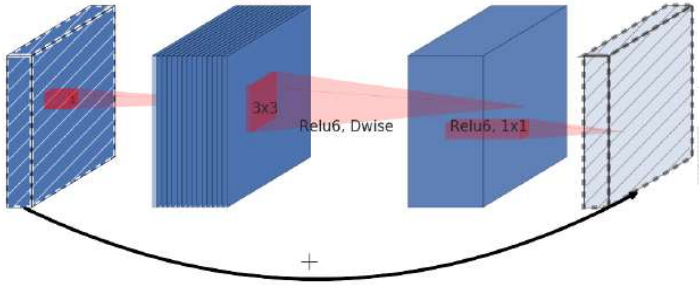


> Traditional CNN
  - >> Huge amounts of operations
  - >> Redundant parameters
  - >> Hard to deploy

> State-of-art CNNs
  - >> Less operations
  - >> Less parameters
  - >> Low bit friendly
  - >> Easy to deploy



Large and deep CNN



State-of-art CNN

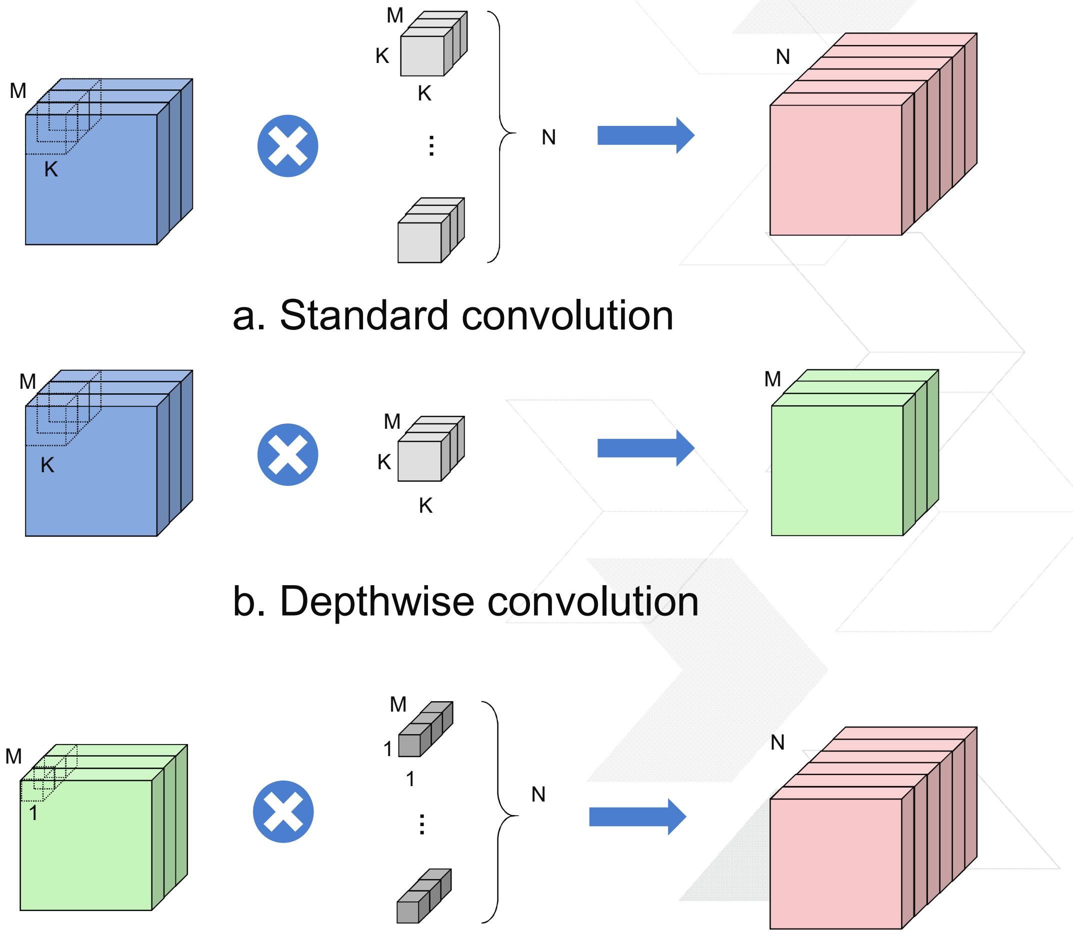
XILINX.

Depthwise separable convolution

> Standard convolution

> Depthwise separable convolution
>> Depthwise convolution
– output channel equals to Input channel
>> Pointwise convolution
– A standard convolution with 1x1 kernel

> Reduce the operations and parameters
>> $F_O = \frac{O_{DSC}}{O_{SC}} = \frac{1}{N} + \frac{1}{K^2}$ [1]
>> $F_P = \frac{P_{DSC}}{P_{SC}} = \frac{1}{N} + \frac{1}{K^2}$ [1]



a. Standard convolution

b. Depthwise convolution

c. Pointwise convolution

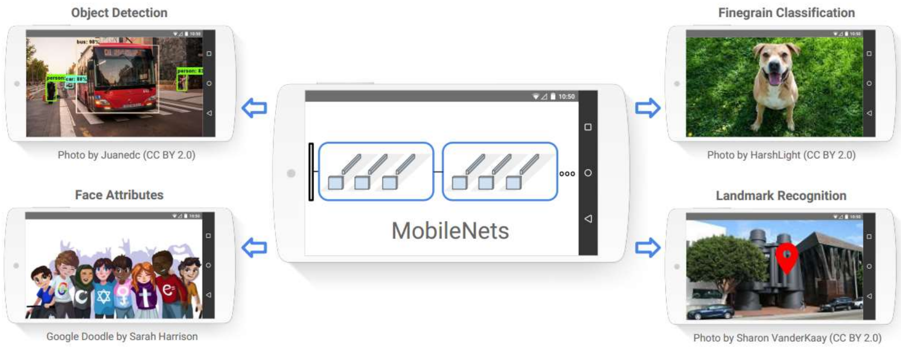
[1] Andrew G. Howard et al. MobileNets: Efficient convolutional neural networks for mobile vision applications, arXiv:1704.04861 [cs], Apr. 2017.

MobileNets

> Less operations and parameters.
> High accuracy
> Efficient model for computer vision tasks on embedded devices

Model	Accuracy	Mult-Adds	Parameters
AlexNet	57.2%	720M	60M
GoogleNet	69.8%	1550M	6.8M
VGG 16	71.5%	15300M	138M
MobileNetV1	70.6%	569M	4.2M
MobileNetV2	72.0%	300M	3.4M

Table1: Classification comparison on ImageNets



MobileNets in mobile phone[1]

Network	mAP	Paras	Mult-Adds
SSD300	23.2	36.1M	35.2B
SSD512	26.8	36.1M	99.5B
YOLOv2	21.6	50.7M	17.5B
MNetV1+SSDLite	22.2	5.1M	1.3B
MNetV2+SSDLite	22.1	4.3M	0.8B

Table2: Object detection comparison on COCO

[1] Andrew G. Howard et al. MobileNets: Efficient convolutional neural networks for mobile vision applications, arXiv:1704.04861 [cs], Apr. 2017.

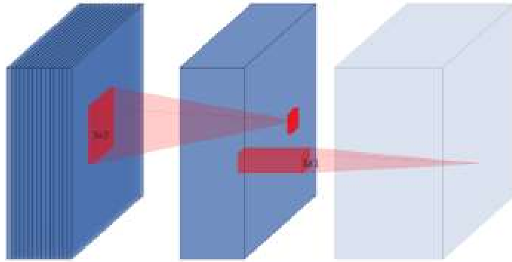
MobileNetV1 vs MobileNetV2

New features in MobileNetV2:

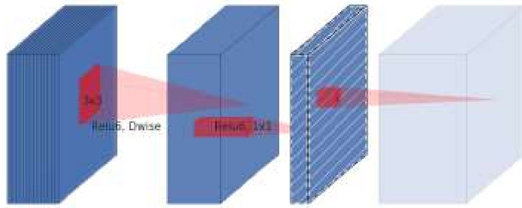
- Linear Bottlenecks (fig. b)
  - Prevent valid information loss
- Inverted Residuals (fig. c)
  - Faster training and better accuracy
- ReLU -> ReLU6

Advantages:

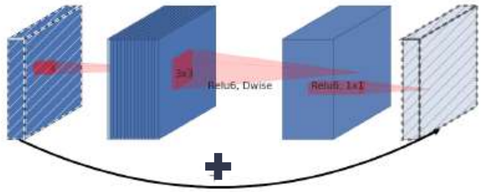
- 2x fewer operations
- 30% fewer parameters
- 30-40% faster
- Reduce the bandwidth requirement



a. MNetV1 : Separable Convolution Block



b. MNetV2: Bottleneck Convolution Block



c. MNetV2 : Inverted residual Block

XILINX.

Workload analysis

Table3: MobileNetV2 Model Architecture

Input	Operator	*t*	*c*	*n*	*s*
$224^2 \times 3$	Conv2D	-	32	1	2
$112^2 \times 32$	Bottleneck	1	16	1	1
$112^2 \times 16$	Bottleneck	6	24	2	2
$56^2 \times 24$	Bottleneck	6	32	3	2
$28^2 \times 32$	Bottleneck	6	64	4	2
$14^2 \times 64$	Bottleneck	6	96	3	1
$14^2 \times 96$	Bottleneck	6	160	3	2
$7^2 \times 160$	Bottleneck	6	320	1	1
$7^2 \times 320$	Conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	Avgpool 7x7	-	-	1	-
$1^2 \times 1280$	Conv2d 1x1	-	k	-	

Each line describes a sequence of 1 or more identical layers, repeated ***n*** times. All layers in the same sequence have the same number ***c*** of output channels. The ***s*** is for stride and ***t*** is the expansion factor.

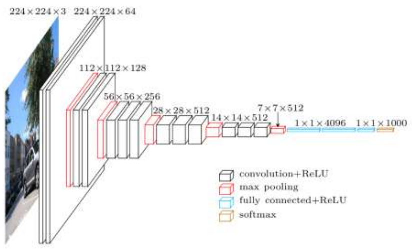
Convolution Type	MAdds Operations	Parameters
Pointwise Conv 1x1	94.86%	74.59%
Depthwise Conv 3x3	3.06%	1.06%
Standard Conv 3x3	1.19%	0.02%
Fully Connected (1x1)	0.18%	24.33%

Table4: Proportion of Per Convolution Type

- The ratio of operation between standard convolution and depthwise convolution is nearly 32.
- This ratio instructs us to design the computation parallelism of different convolution engines.

CNN Accelerators

- Large and deep CNN:
  - Low efficiency
  - A lot of accelerators
- State-of-art CNN:
  - High efficiency
  - Less accelerators
- Motivation:
  - Deploy the state-of-art CNN MobileNets into a high-performance platform

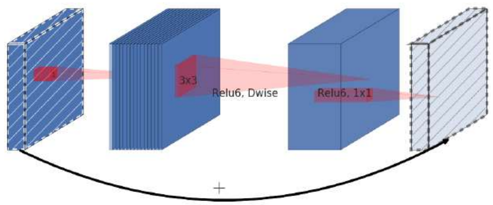


Large and deep CNN

GPU: High performance, Huge power consumption

ASIC: High performance, Low power consumption, Low flexibility

FPGA: High performance, Efficient power consumption, High flexibility



State-of-art CNN → FPGA

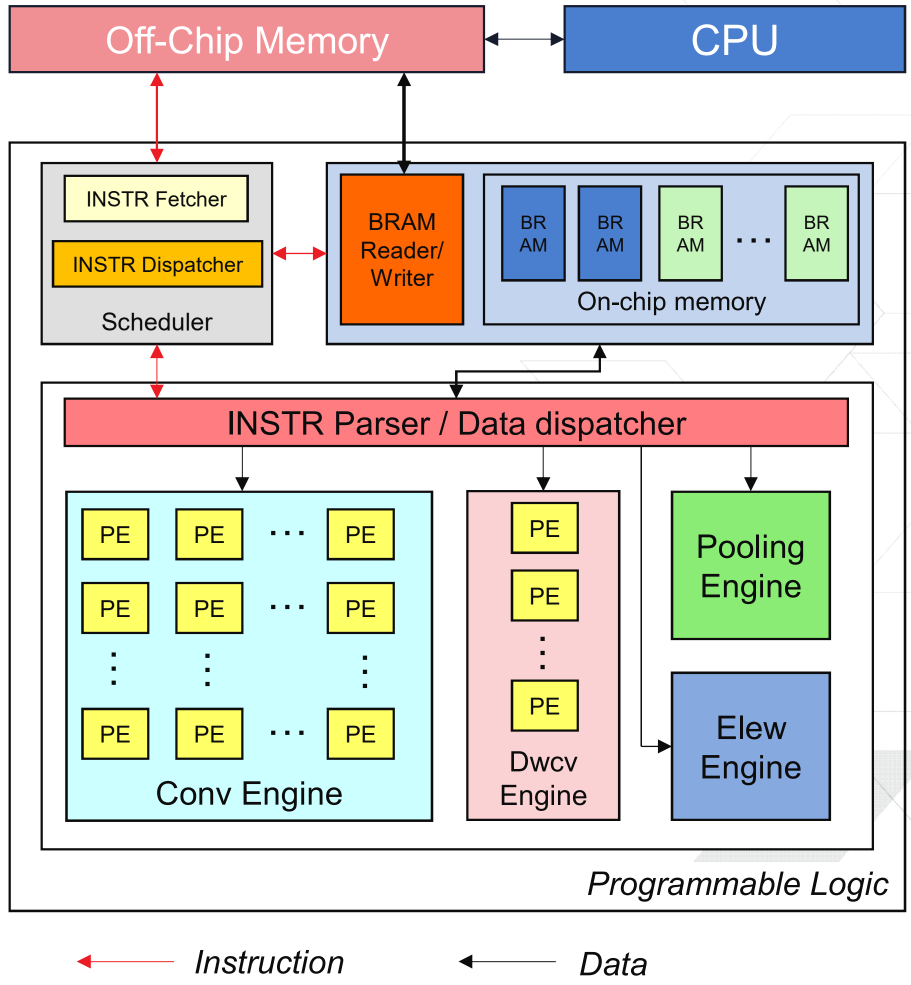
XILINX

Our works

XILINX

Hardware architecture

- Deep learning Processing Unit (DPU)
  - Instruction-driven
  - On-chip memory bank
  - Convolution Engine (Conv Engine)
  - Depthwise Conv Engine (Dwcv Engine)
  - Pooling Engine
  - Elementwise Engine
- Scalable number of on-chip bank
- Scalable size of Conv Engine and Dwcv Engine

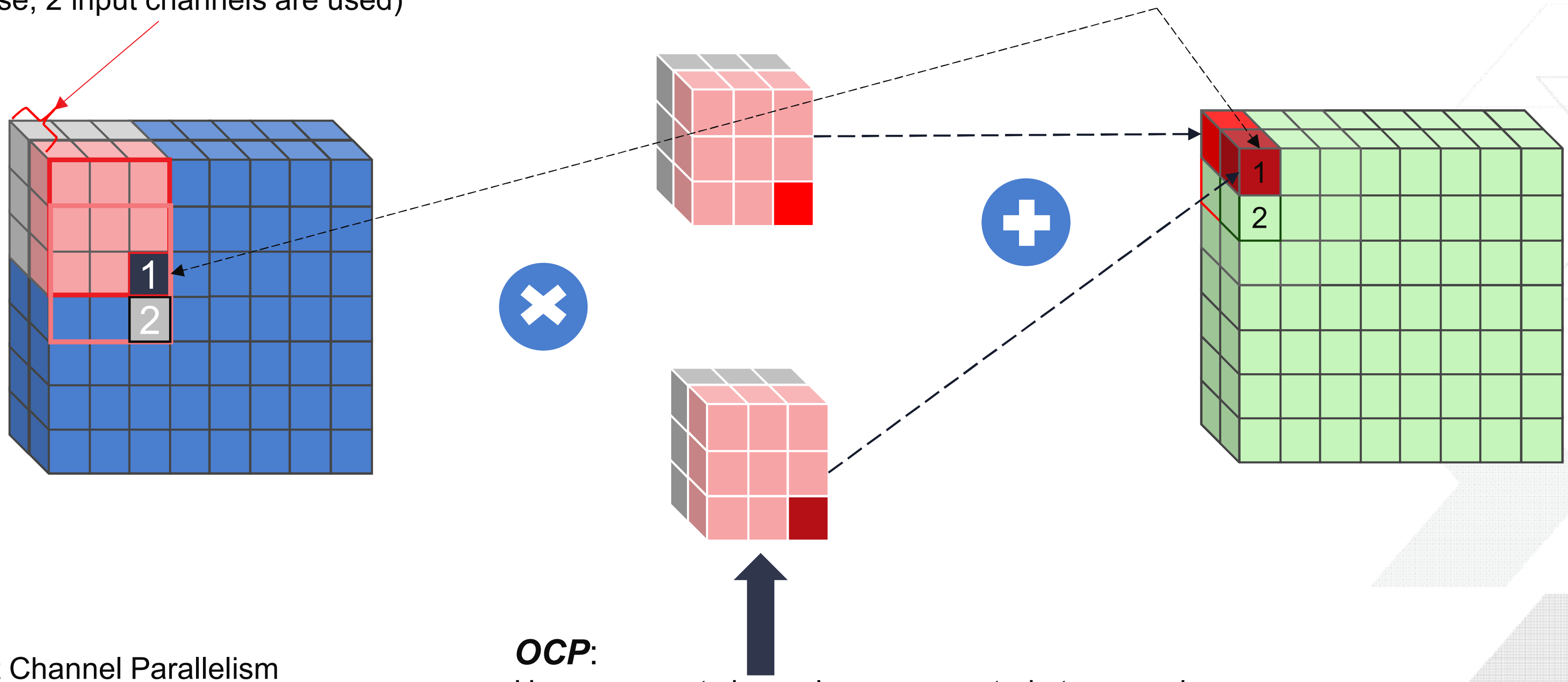


Three Parallelisms (example of 2*2*2)

PP ICP OCP

ICP:
How many input channels are computed in one cycle
(in this case, 2 input channels are used)

PP:
How many pixels are generated in parallel
(in this case 2 pixels are generated, pixel 1&2)



Note:
ICP: Input Channel Parallelism
OCP: Output Channel Parallelism
PP: Pixel Parallelism

OCP:
How many out channels are generated at one cycle,
i.e, how many kernels are used at one cycle
(in this case, 2 kernels used & 2 output channels generated)

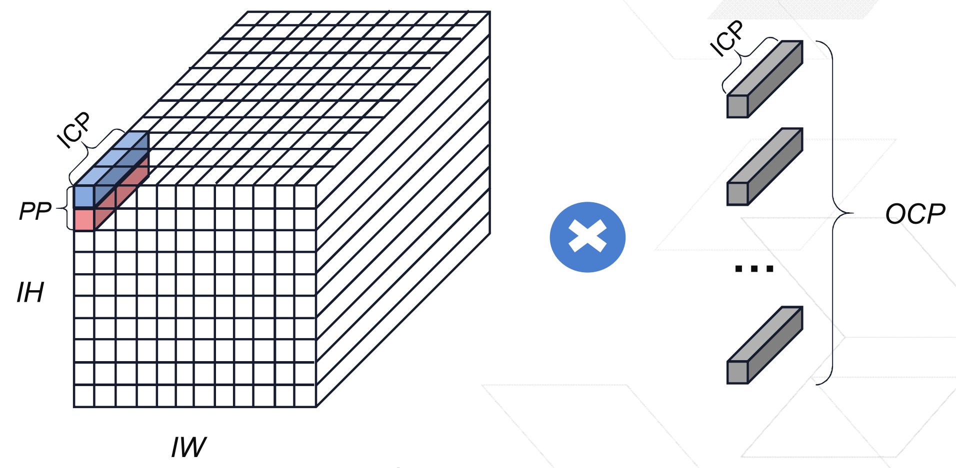
XILINX

Mechanism of Conv Engine & Dwcv Engine

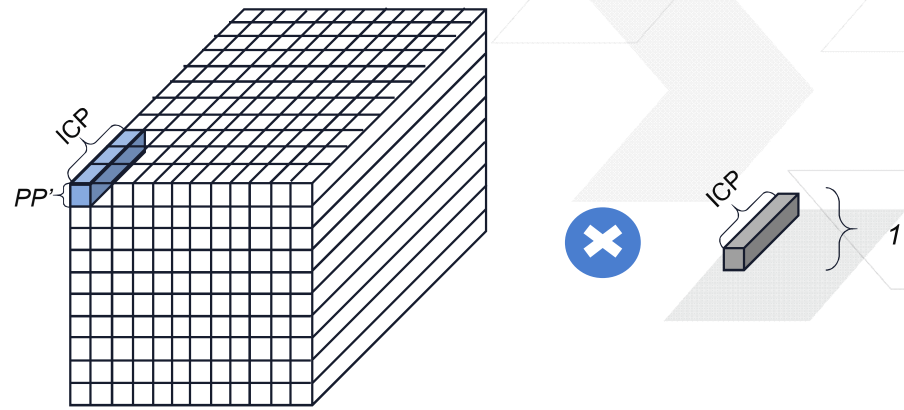
> Conv Engine
  >> Channel direction
  >> Image width direction
  >> Image height direction

> Dwcv Engine
  >> Same orientation of Conv Engine
  >> Output channel parallelism fixed to be 1

> The parallelisms of Conv Engine and Dwcv Engine are adjustable.



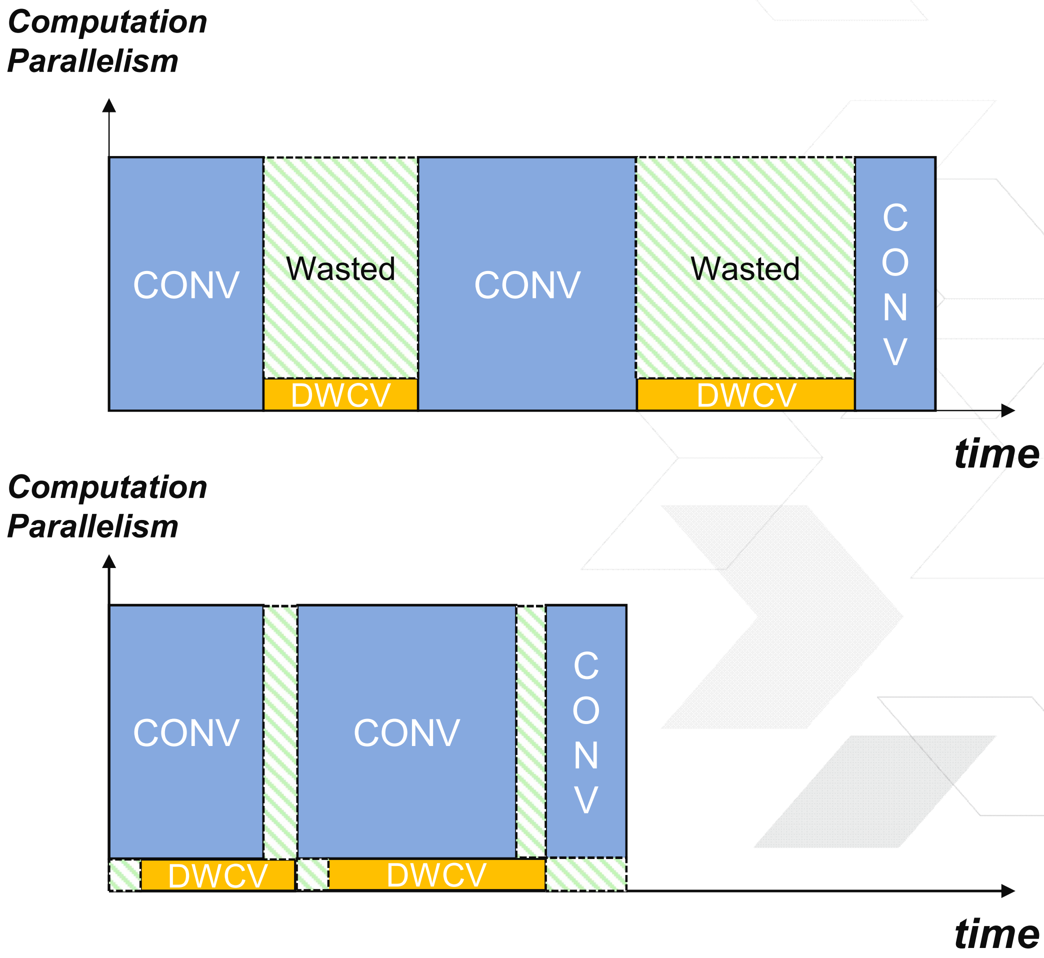
a. Conv Engine



b. Dwcv Engine

Pipeline schedule among layers

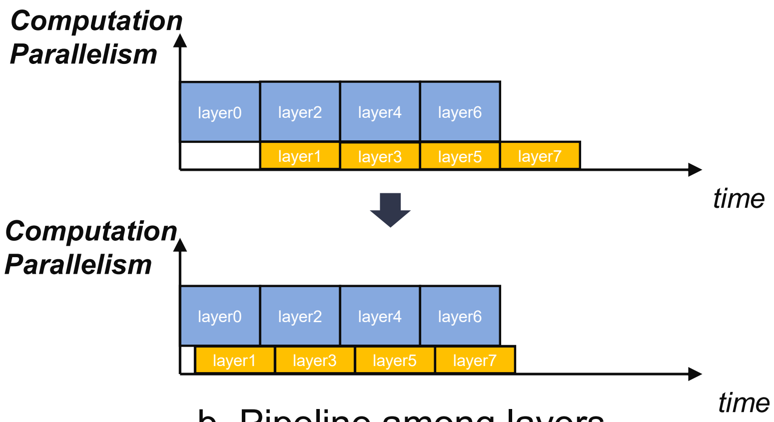
- Traditional CNN accelerator
  - One compute engine for standard convolution and depthwise convolution
  - A lot of waste
  - Low efficiency
- Our accelerator
  - Dedicated Dwcv engine for depthwise convolution
  - Less waste
  - Less runtime and higher efficiency
- Adjust the computation parallelism of Conv Engine and Dwcv Engine.



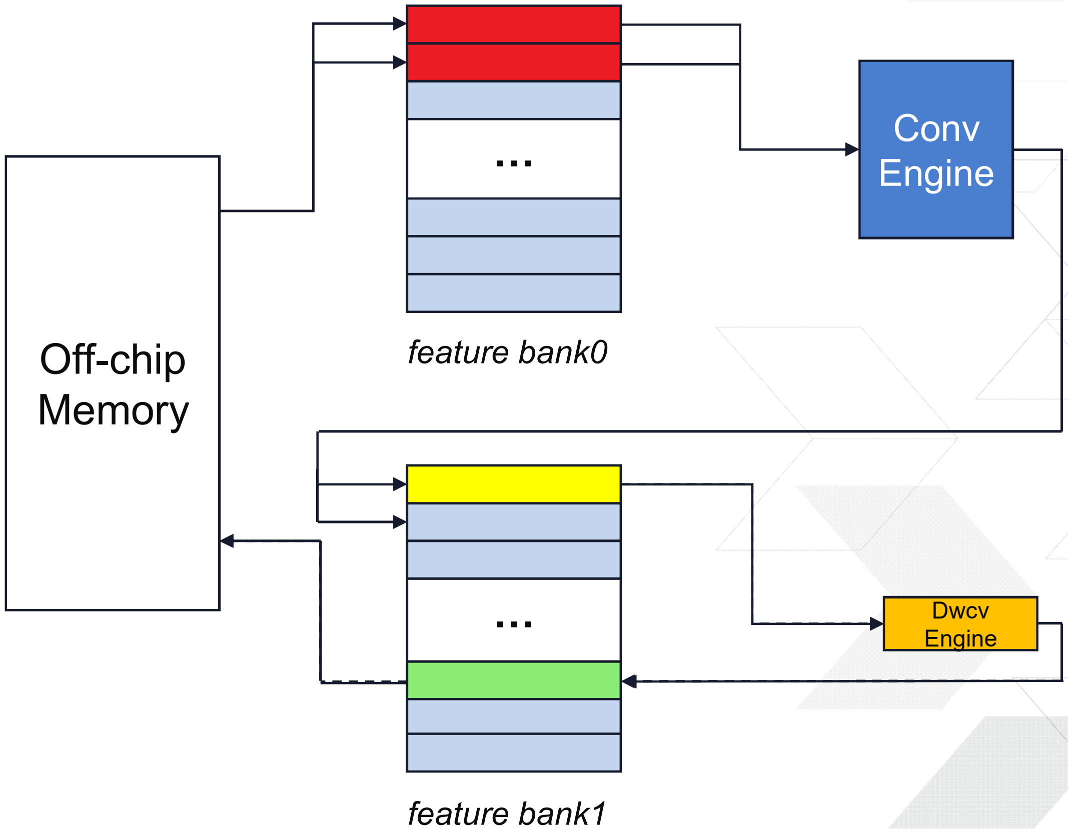
XILINX.

Conv Engine and Dwcv Engine work in pipeline

- Transform the granularity of data dependence from whole output feature map to a row of feature map
  - Tighten the pipeline between Conv Engine and Dwcv Engine
  - Improve the efficiency of whole network



b. Pipeline among layers

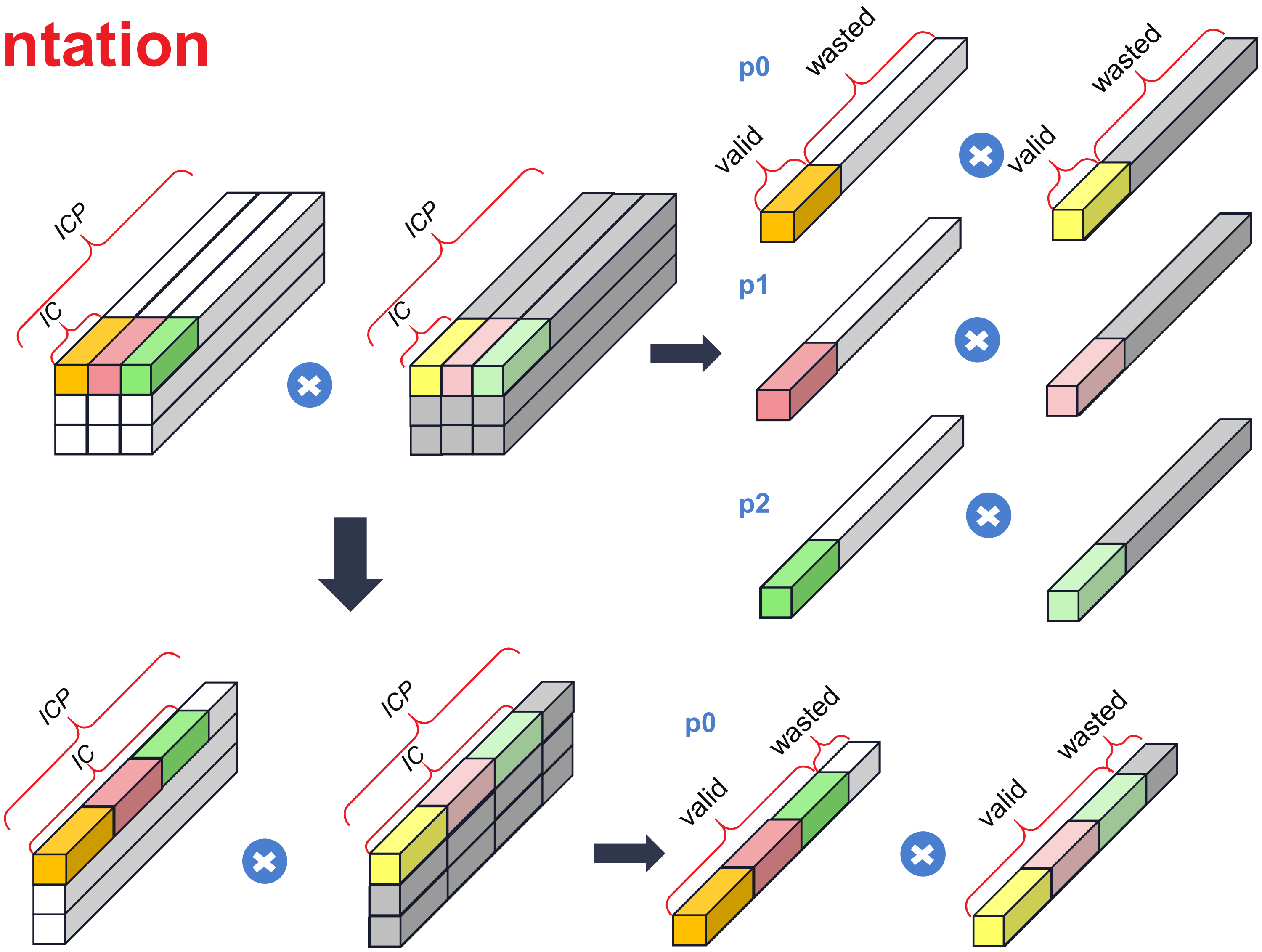


a. Data flow of Conv Engine and Dwcv Engine

XILINX.

Channel Augmentation

- Input Channel far less than the Input Channel Parallelism:
  - Consume more cycle to finish operation
  - Low efficiency
- Channel Augmentation: Rearrange the data format of input feature and weights
  - Consume less cycle
  - Higher efficiency



XILINX

Experiment results

Implementation Notes

- Xilinx ZU2EG and ZU9EG
- 8bit Activation and Weights quantization
- Without fine tune
- Clock frequency:
  - ZU2 - 430MHz
  - ZU9 – 330MHz

DPU	LUT	FF	BRAM(36k)	DSP48
DPU_ZU2	31198	46809	145	212
DPU_ZU9	161944	301416	771	2070

Table5: Resources of different DPUs

Acceleration based on Channel Augmentation

- For the first layer:
  - Input channel : 3 (general)
  - Kernel size :3x3
  - The ICP is lager than the IC
  - Theoretical efficiency:
    - 3/12 -> 9/12
    - 3/16 -> 9/16
  - The operation of first layer accounts for a large proportion : 6%
  - The Channel Augmentation has a significant acceleration for the first layer.
    - Speedup on ZU2 : 1.9x
    - Speedup on ZU9: 2.1x

DPU	ICP	OCP	PP
DPU_ZU2	12	12	4
DPU_ZU9	16	16	8

Table3: Parallelisms of two DPUs

Platform	Configuration	Runtime(ms)	Speedup
ZU2	Without Augm	0.42	1.9x
	With Augm	0.22	
ZU9	Without Augm	0.21	2.1x
	With Augm	0.10	

Table4: Acceleration based on Channel Augmentation for MobileNetV2 layer1

XILINX.

Classification Results

Design	Network	Platform	Speed (fps)	Top1 Accuracy	Prec. (W/a)
[1]	MobileNetV2	CPU	13.3	72.0%	32b
[2]	RR-MobileNet	ZU9EG	127.4	64.6%	8-4b
[3]	MobileNetV1	Stratix V	231.7	-	-
[4]	DiracDeltaNet	ZU3EG	96.5	68.47%	1-4b
[5]	MobileNetV2	Arria10	266.2	-	16b
This work	MobileNetV2	ZU2EG	205.3	68.1%	8b
This work	MobileNetV2	ZU9EG	809.8	68.1%	8b

Table6: Comparison of classification on ImageNet

[1] Sandler, et al. MobileNetV2:Inverted Residuals and Linear Bottlenecks. CVPR, 2018.
[2] J. Su et al., "Redundancy-reduced mobilenet acceleration on reconfigurable logic for ImageNet classification", Proc. Appl. Reconfig. Comput. Archit. Tools Appl., pp. 16-28, 2018.
[3] R. Zhao, X. Niu, W. Luk, "Automatic optimising CNN with depthwise separable convolution on FPGA: (Abstract only)", FPGA, 2018.
[4] Yifan Yang, et al. Synetgy: Algorithm-hardware Codesign for ConvNet Accelerators on Embedded FPGAs. (FPGA'19).
[5] Bai L , Zhao Y , Huang X . A CNN Accelerator on FPGA Using Depthwise Separable Convolution[J]. IEEE Transactions on Circuits & Systems II Express Briefs, 2018.

Detection Results

Framework	Platform	Speed (fps)	Prec. (W/a)	mAP
MobileNetV1 + SSD	ZU2EG (this work)	31.0	8b	22.1^{α}
MobileNetV1 + SSD	ZU9EG (this work)	124.3	8b	16.2^{α}
MobileNetV2 + SSD	ZU9EG (this work)	138.0	8b	29.4^{β}

Table7: Performance of object detection

Note: $^{\alpha}$The detection results are tested on the COCO dataset. Input size 320x320
$^{\beta}$The detection results are tested on the Bdd100k dataset, Input size : 480x360

XILINX.

Demo - Classification

Demo - Detection

XILINX

Conclusion

- Deploy the MobileNets into FPGAs
  - Efficient model
  - Low bandwidth requirement
- Improve the implementation efficiency of MobileNets
  - Design a dedicated Dwcv Engine for depthwise convolution
  - Adjust the computation parallelism of Conv Engine and Dwcv Engine for MobileNets architecture
  - Pipeline the standard convolution and depthwise convolution in MobileNets
  - Use Channel Augmentation for the layer which input channel is far less than the input channel parallelism
- Implementation results
  - Image classification on ImageNet: ZU2EG: 205.3fps ZU9EG: 809.8fps
  - Object detection : MobileNetV1 + SSD on ZU2 : 31fps
    MobileNetV1 + SSD on ZU9 : 124.3fps
    MobileNetV2 + SSD on ZU9 : 138.0fps

Adaptable.
Intelligent.