IEEE Xplore® Digital Library

ISCAS 2020

2020 IEEE International Symposium on Circuits and Systems
Virtual, October 10-21, 2020

Efficient Accelerator for Dilated and Transposed Convolution with Decomposition

Kuo-Wei Chang, Tian-Sheuan Chang

Dept. of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan

andy0703.ee05g@nctu.edu.tw, tschang@mail.nctu.edu.tw

2020 IEEE International Symposium on Circuits and Systems
Virtual, October 10-21, 2020

MOST 科技部 Ministry of Science and Technology

NARLabs 國家實驗研究院 台灣半導體研究中心 Taiwan Semiconductor Research Institute

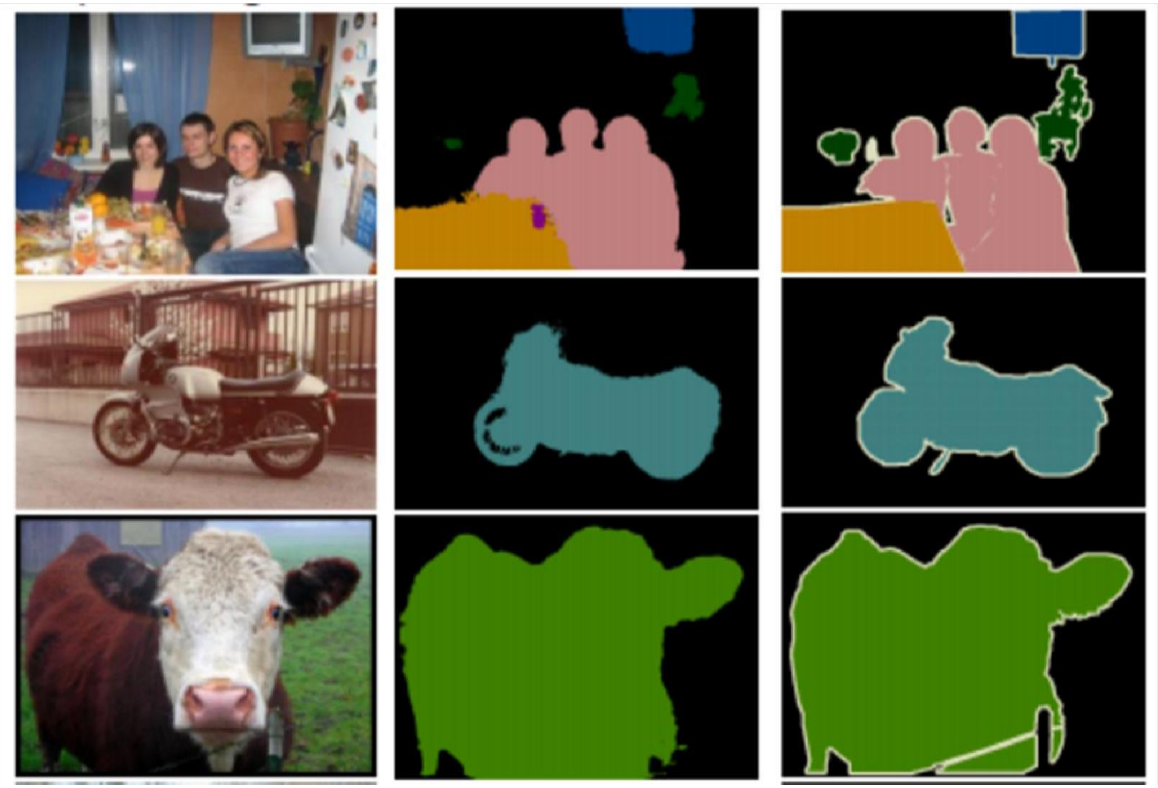
VLSI Signal Processing Lab.

Outline

- Introduction
- Proposed Accelerator
- Proposed Method
- Experiment Results
- Conclusion

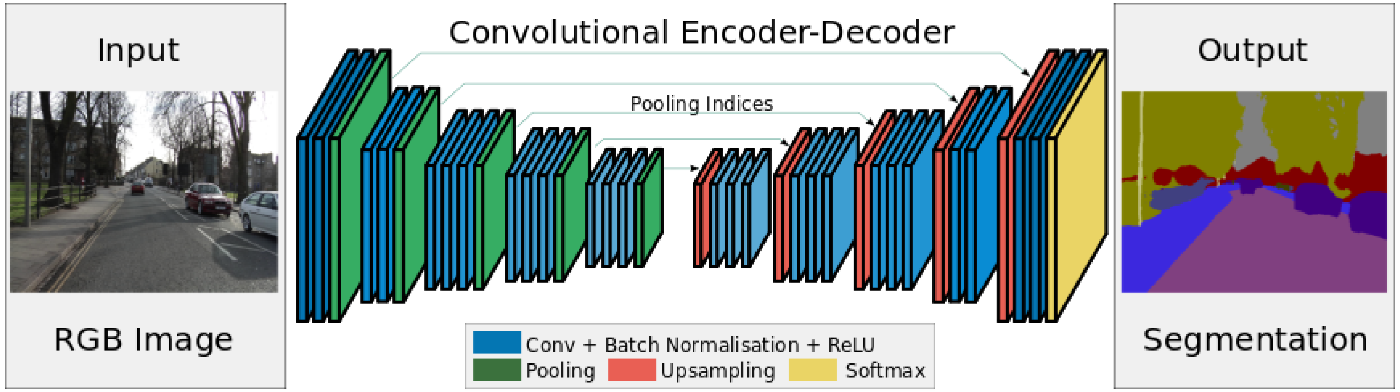
Semantic Segmentation

- Recognize objects and non-objects in a image
  - Label each pixel in the image with a category label
  - Don't differentiate instances, only care about pixels

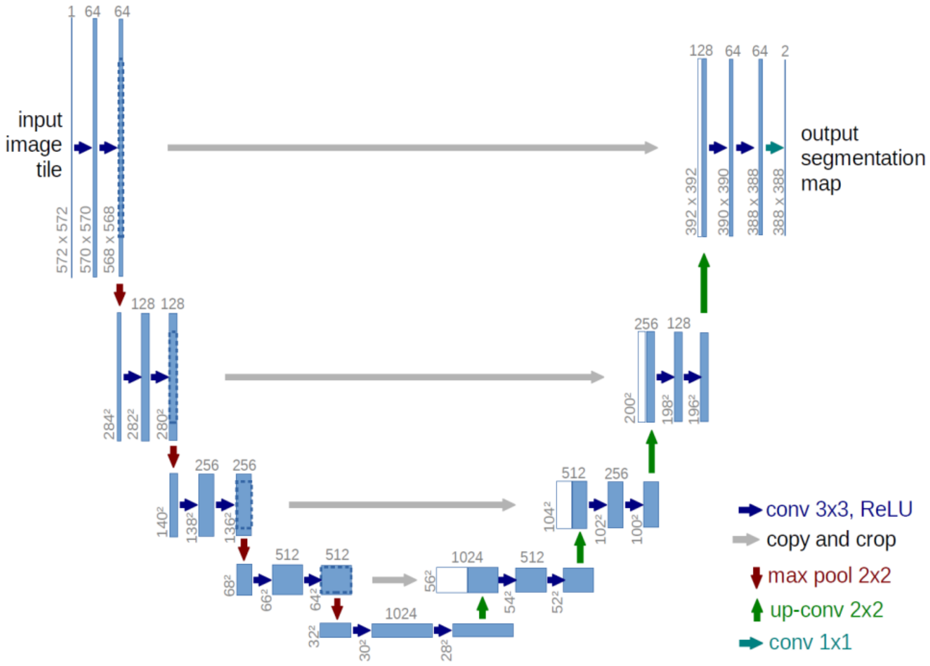


Semantic segmentation State of the art Methods

- SegNet



- U-Net

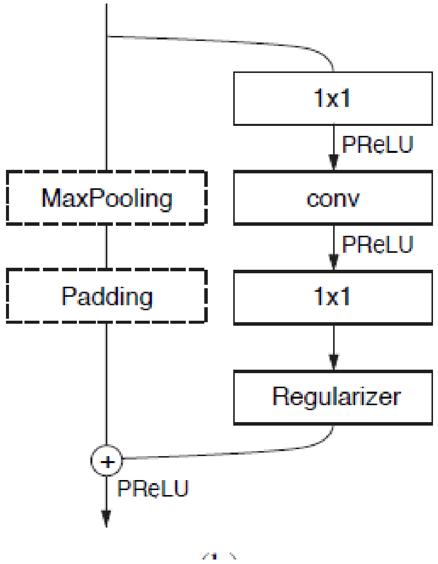


Source: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, 2015

Ronneberger et al, “U-Net: Convolutional Networks for Biomedical Image Segmentation”, arXiv 2015

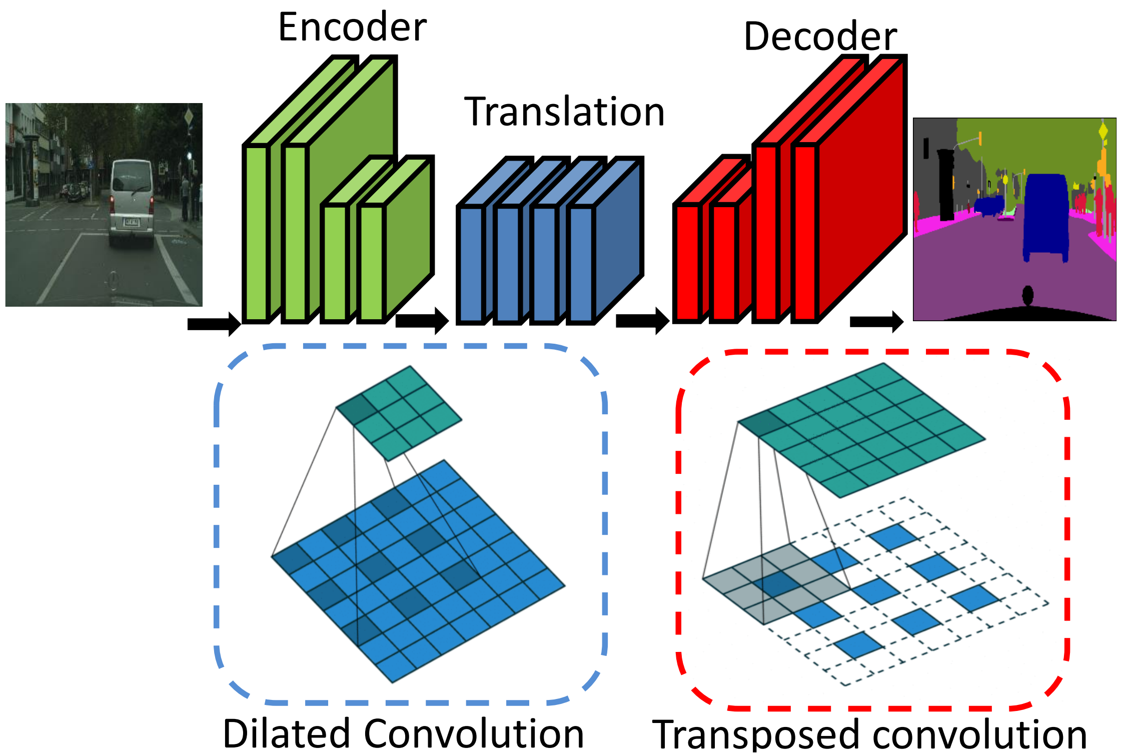
Semantic segmentation State of the art Methods

- E-Net
- Parameter reduction
  - ResNet-like
  - Use dilated convolution
  - Use asymmetric convolution
  - No bias terms



Name	Type	Output size
initial		$16 \times 256 \times 256$
bottleneck1.0	downsampling	$64 \times 128 \times 128$
$4\times$ bottleneck1.x		$64 \times 128 \times 128$
bottleneck2.0	downsampling	$128 \times 64 \times 64$
bottleneck2.1		$128 \times 64 \times 64$
bottleneck2.2	dilated 2	$128 \times 64 \times 64$
bottleneck2.3	asymmetric 5	$128 \times 64 \times 64$
bottleneck2.4	dilated 4	$128 \times 64 \times 64$
bottleneck2.5		$128 \times 64 \times 64$
bottleneck2.6	dilated 8	$128 \times 64 \times 64$
bottleneck2.7	asymmetric 5	$128 \times 64 \times 64$
bottleneck2.8	dilated 16	$128 \times 64 \times 64$
Repeat section 2, without bottleneck2.0		
bottleneck4.0	upsampling	$64 \times 128 \times 128$
bottleneck4.1		$64 \times 128 \times 128$
bottleneck4.2		$64 \times 128 \times 128$
bottleneck5.0	upsampling	$16 \times 256 \times 256$
bottleneck5.1		$16 \times 256 \times 256$
fullconv		$C \times 512 \times 512$

Summary of Image Segmentation

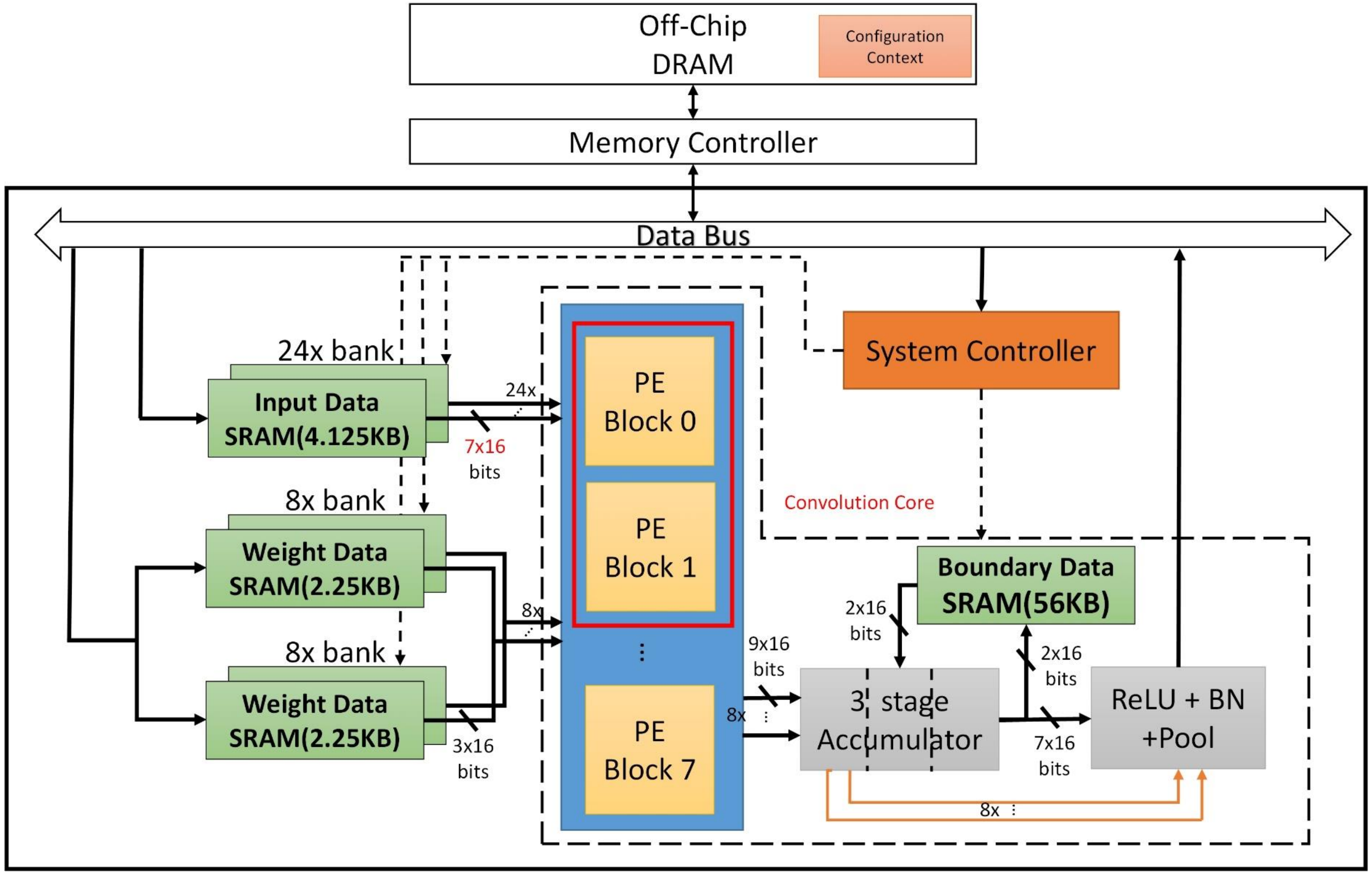


Proposed Architecture

- Base hardware
  - Vector based array
- Reconfiguration
  - Various convolution types
    - 3x3, 4x4, 5x5, 7x7, 11x11 with different strides
    - 1x1
    - Depth wise
    - Dilated convolution
    - Deconvolution
    - Vector level sparsity
  - Through reconfigurable input and output selections

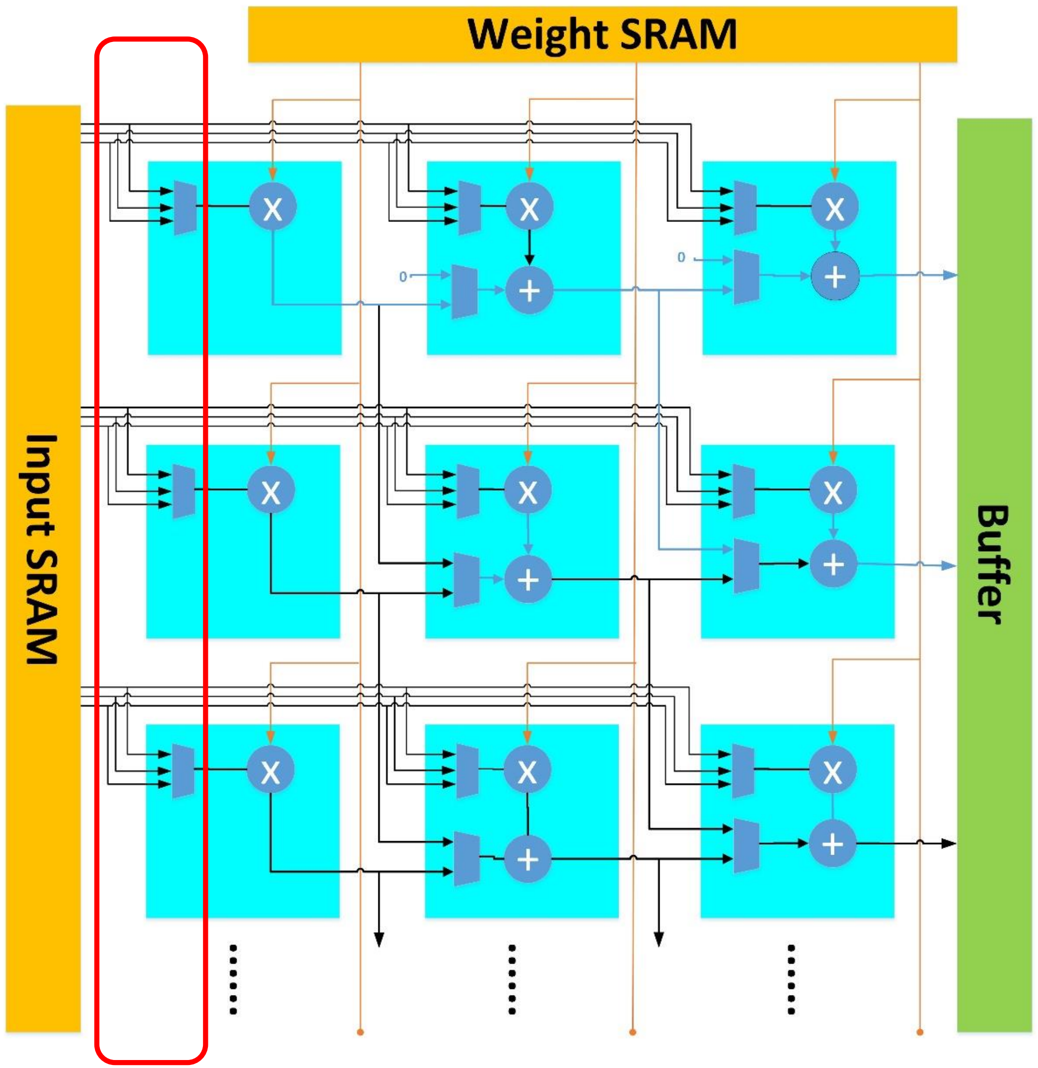
K. Chang and T. Chang, "VWA: Hardware Efficient Vectorwise Accelerator for Convolutional Neural Network," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 1, pp. 145-154, Jan. 2020, doi: 10.1109/TCSI.2019.2942529.

Architecture – Top



Architecture – PE block

- General convolution
  - vectorwise input
- Non-unit stride
  - Interleaved input
- 1x1 convolution
  - Elementwise input



General Convolution

- Illustration of computation and order
  - A to F and WA to WE represent input and weight row vector. Number 1 to 12 represent cycle.

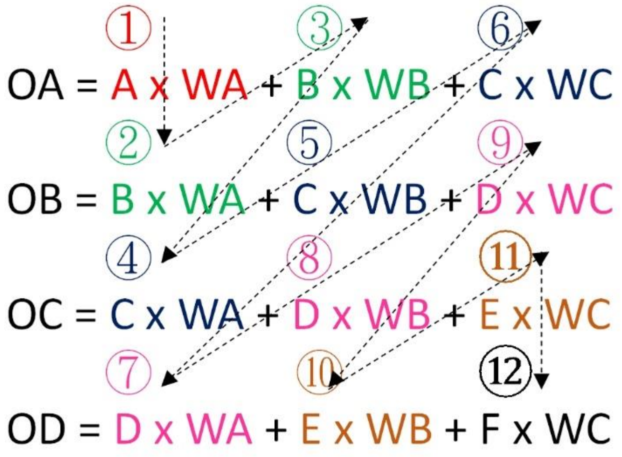
A0	B0	C0	D0	E0	F0
A1	B1	C1	D1	E1	F1
A2	B2	C2	D2	E2	F2
A3	B3	C3	D3	E3	F3
A4	B4	C4	D4	E4	F4

×

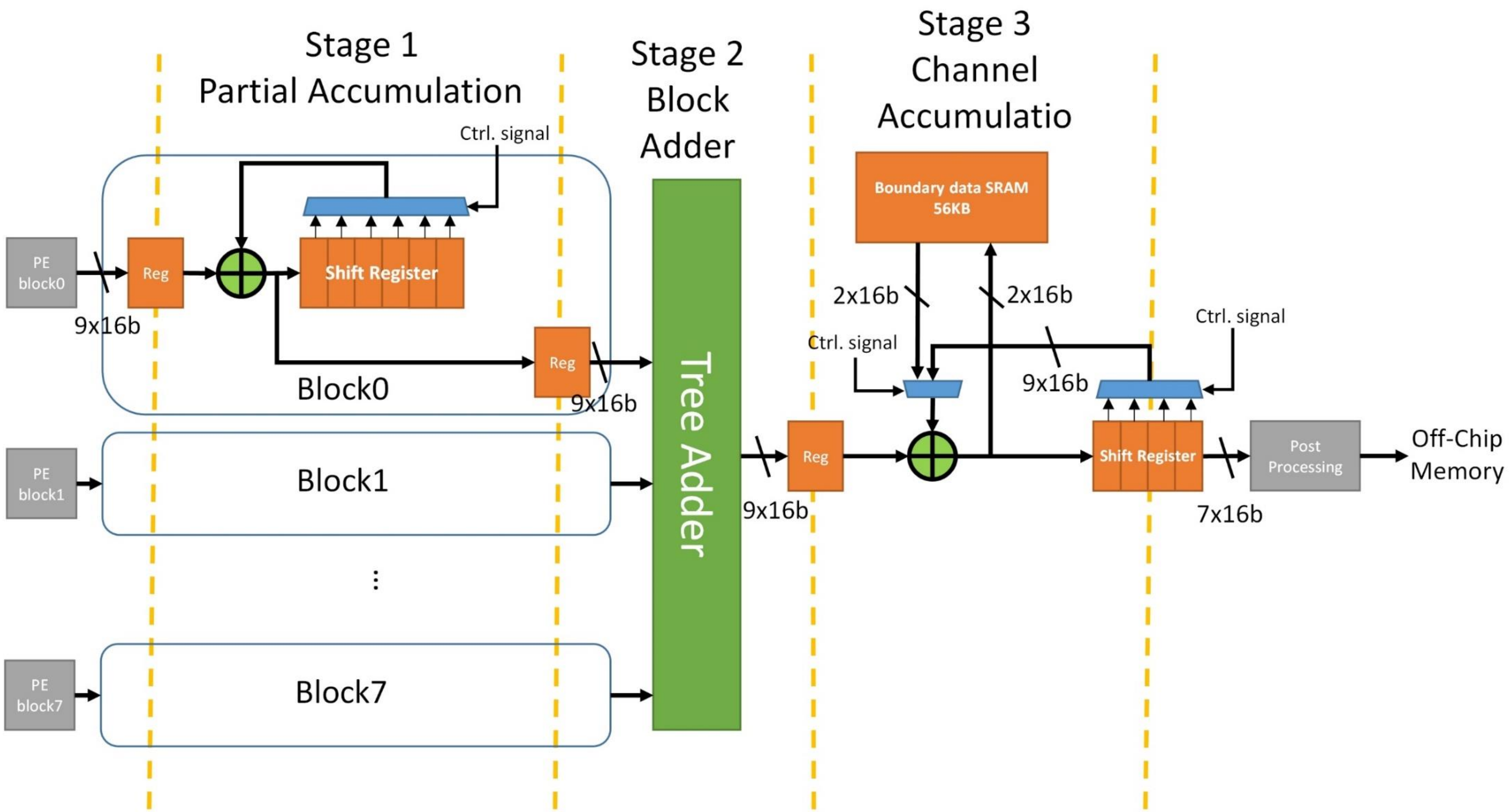
WA0	WB0	WC0
WA1	WB1	WC1
WA2	WB2	WC2

=

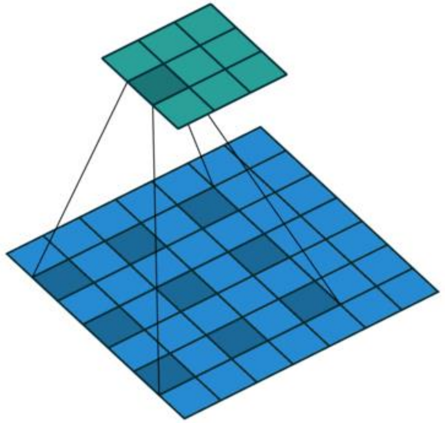
OA0	OB0	OC0	OD0
OA1	OB1	OC1	OD1
OA2	OB2	OC2	OD2
OA3	OB3	OC3	OD3
OA4	OB4	OC4	OD4
OA5	OB5	OC5	OD5
OA6	OB6	OC6	OD6



Architecture – Accumulator



Dilated Convolution



- Problems with pooling for downsampling
  - Loss position information
  - Keep position information while expand receptive filed exponentially => dilated convolution

wa_1	wb_1	wc_1
wa_2	wb_2	wc_2
wa_3	wb_3	wc_3

D = 0

wa_1		wb_1		wc_1
wa_2		wb_2		wc_2
wa_3		wb_3		wc_3

D = 1

wa_1			wb_1			wc_1
wa_2			wb_2			wc_2
wa_3			wb_3			wc_3

D = 2

Input decomposition for Dilated Convolution (D = 1)

- For D= 1 dilated convolution
- Like stride 2 convolution
- Input is decomposed to four block by stride 2.

wa_1	wb_1	wc_1
wa_2	wb_2	wc_2
wa_3	wb_3	wc_3

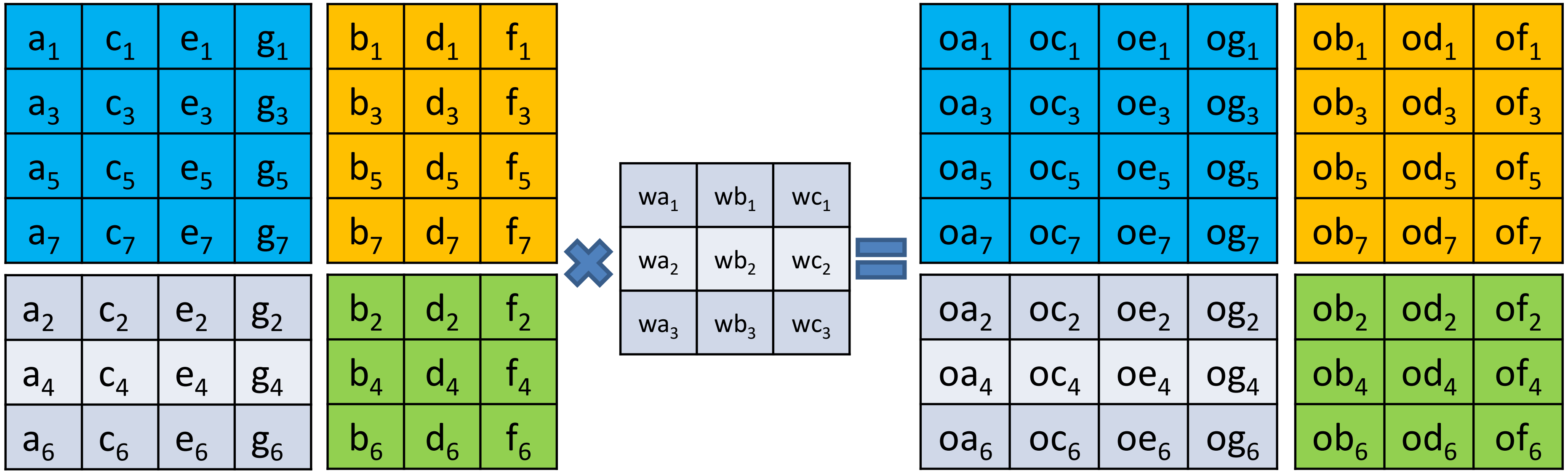
wa_1		wb_1		wc_1
wa_2		wb_2		wc_2
wa_3		wb_3		wc_3

Illustration of input deompostion

a_1	b_1	c_1	d_1	e_1	f_1	g_1
a_2	b_2	c_2	D_2	e_2	f_2	g_2
a_3	b_3	c_3	d_3	e_3	f_3	g_3
a_4	b_4	c_4	d_4	e_4	f_4	g_4
a_5	b_5	c_5	d_5	e_5	f_5	g_5
a_6	b_6	c_6	d_6	e_6	f_6	g_6
a_7	b_7	c_7	d_7	e_7	f_7	g_7

Input decomposition for Dilated Convolution (D = 1)

- General convolution after input decomposition
- Skip all zero computation in sparse dilated weight.



Input decomposition for Dilated Convolution (D = 2)

D=2 Weight

wa_1			wb_1			wc_1
wa_2			wb_2			wc_2
wa_3			wb_3			wc_3

Input decomposed

a_1	b_1	c_1	d_1	e_1	f_1	g_1
a_2	b_2	c_2	d_2	e_2	f_2	g_2
a_3	b_3	c_3	d_3	e_3	f_3	g_3
a_4	b_4	c_4	d_4	e_4	f_4	g_4
a_5	b_5	c_5	d_5	e_5	f_5	g_5
a_6	b_6	c_6	d_6	e_6	f_6	g_6
a_7	b_7	c_7	d_7	e_7	f_7	g_7

a_1	d_1	g_1
a_4	d_4	g_4
a_7	d_7	g_7

b_1	e_1
b_4	e_4
b_7	e_7

c_1	f_1
c_4	f_4
c_7	f_7

a_2	d_2	g_2
a_5	d_5	g_5

b_2	e_2
b_5	e_5

c_2	f_2
c_5	f_5

a_3	d_3	g_3
a_6	d_6	g_6

b_3	e_3
b_6	e_6

c_3	f_3
c_6	f_6

×

wa_1	wb_1	wc_1
wa_2	wb_2	wc_2
wa_3	wb_3	wc_3

=

oa_1	od_1	og_1
oa_4	od_4	og_4
oa_7	od_7	og_7

ob_1	oe_1
ob_4	oe_4
ob_7	oe_7

oc_1	of_1
oc_4	of_4
oc_7	of_7

oa_2	od_2	og_2
oa_5	od_5	og_5

ob_2	oe_2
ob_5	oe_5

oc_2	of_2
oc_5	of_5

oa_3	od_3	og_3
oa_6	od_6	og_6

ob_3	oe_3
ob_6	oe_6

oc_3	of_3
oc_6	of_6

Transposed convolution (Deconvolution)

0	0	0	0	0	0	0
0	**a_1**	0	**b_1**	0	**c_1**	0
0	0	0	0	0	0	0
0	**a_2**	0	**b_2**	0	**c_2**	0
0	0	0	0	0	0	0
0	**a_3**	0	**b_3**	0	**c_3**	0
0	0	0	0	0	0	0

Inserting zeros
between the adjacent input

✖

wa_1	wb_1	wc_1
wa_2	wb_2	wc_2
wa_3	wb_3	wc_3

Weight

=

oa_1	ob_1	oc_1	od_1	oe_1
oa_2	ob_2	oc_2	od_2	oe_2
oa_3	ob_3	oc_3	od_3	oe_3
oa_4	ob_4	oc_4	od_4	oe_4
oa_5	ob_5	oc_5	od_5	oe_5

Output

Weight decomposed

1.

wa_1	wc_1
wa_3	wc_3

2.

wa_2	wc_2

3.

wb_1
wb_3

4.

wb_2

Input row vector multiply weight row vector, so I. and II. accumulate partial sum in 2 cycle to get output III. and IV. Can get output in 1 cycle.

Data flow chart of Transposed Convolution

Cycle 1	wa_1	wa_3	wa_2
a_1	ob_2		ob_1
a_2	ob_4	ob_2	ob_3
a_3		ob_4	ob_5

	wc_1	wc_3	wc_2
b_1	ob_2		ob_1
b_2	ob_4	ob_2	ob_3
b_3		ob_4	ob_5

	wb_1	wb_3	wb_2
a_1	oa_2		oa_1
a_2	oa_4	oa_2	oa_3
a_3		oa_4	oa_5

Cycle 2	wa_1	wa_3	wa_2
b_1	od_2		od_1
b_2	od_4	od_2	od_3
b_3		od_4	od_5

	wc_1	wc_3	wc_2
c_1	od_2		od_1
c_2	od_4	od_2	od_3
c_3		od_4	od_5

	wb_1	wb_3	wb_2
b_1	oc_2		oc_1
b_2	oc_4	oc_2	oc_3
b_3		oc_4	oc_5

Cycle 3

Idle

Idle

	wb_1	wb_3	wb_2
c_1	oe_2		oe_1
c_2	oe_4	oe_2	oe_3
c_3		oe_4	oe_5

	a_1		b_1		c_1	
	a_2		b_2		c_2	
	a_3		b_3		c_3	

Input

wa_1	wb_1	wc_1
wa_2	wb_2	wc_2
wa_3	wb_3	wc_3

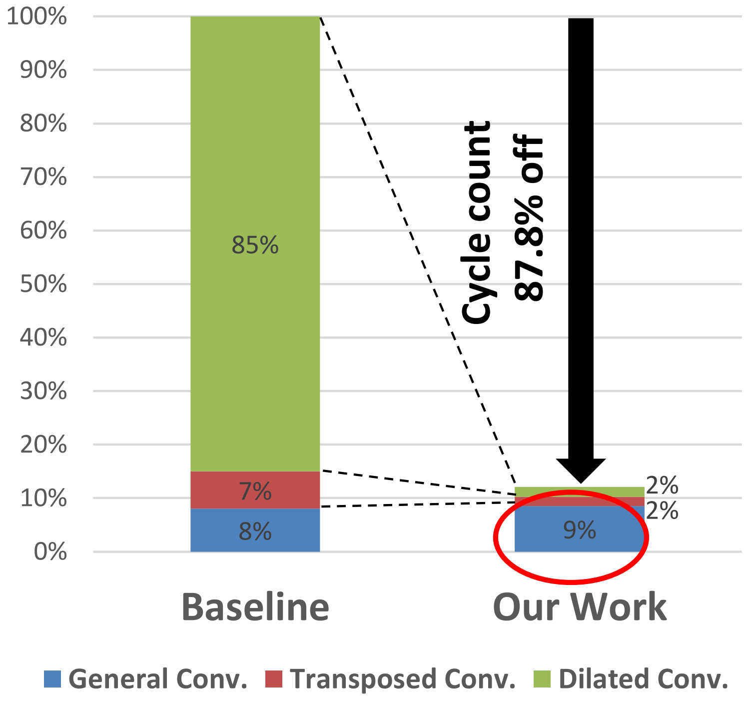
Weight

oa_1	ob_1	oc_1	od_1	oe_1
oa_2	ob_2	oc_2	od_2	oe_2
oa_3	ob_3	oc_3	od_3	oe_3
oa_4	ob_4	oc_4	od_4	oe_4
oa_5	ob_5	oc_5	od_5	oe_5

Output

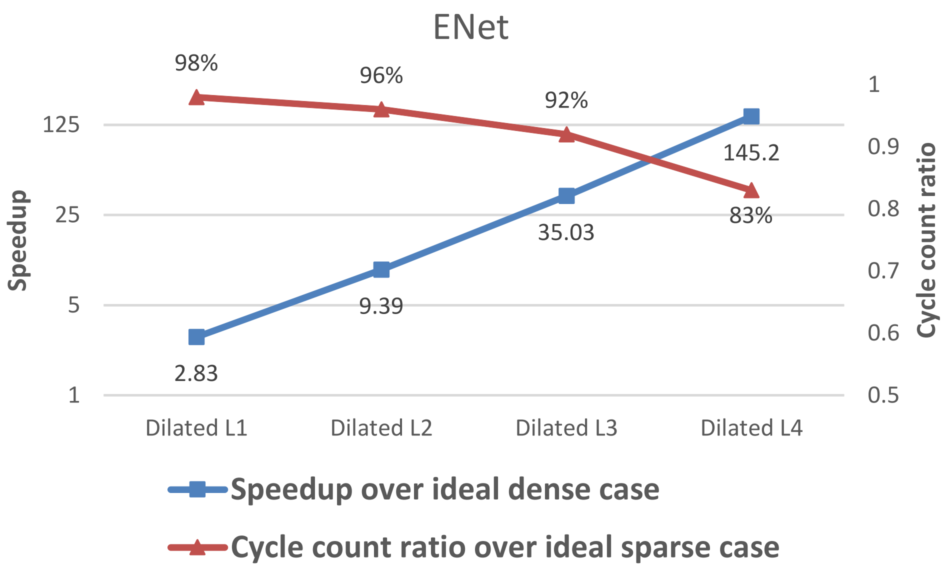
Experimental Result

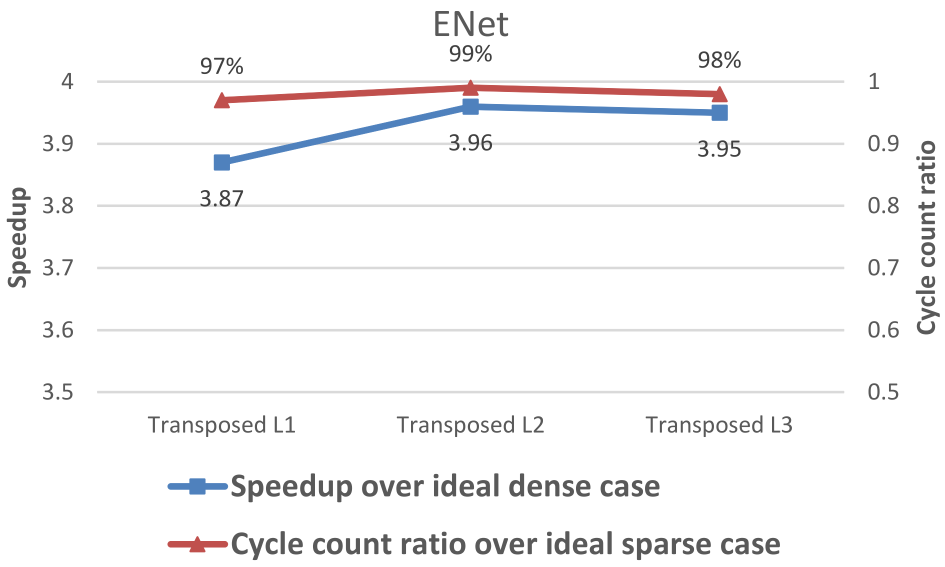
- The performance enhancement for our work on ENet.
  - The baseline is cycle counts on the ideal dense case.
  - The number of MACs are the same in our work and the ideal dense case.



Experimental Result

- The performance of dilated and transposed convolutional layers on ENet.
  - Ideal dense case = Computation including zeros.
  - Ideal sparse case = Computation without all zeros.





Design Comparison

TABLE I
IMPLEMENTATION RESULT AND COMPARISONS WITH OTHER DESIGNS

	Our work	[1] DT-CNN [12]	[2] USCA [13]
Technology	40nm	65nm	28nm
Measurements	Post-layout	Post-layout	Synthesis
Precision	16 fixed	8	-
On-chip SRAM (KB)	191	220.5	114.7
Frequency (MHz)	500	200	1449
Throughput (GOPS)^a	168^d/ 1377^e 168^{bd}/ 1377^{be}	96^d / 639^e 156^{bd}/ 1039^{be}	374 261^{bd}
Supply Voltage (V)	0.99	1.2	-
Core Area (mm^2)	1.5625	6.8	-
Core Power (mW)	155	196	201.1
Area efficiency(GOPS/mm^2)	107^d / 881^e **107**^{bd} / **881**^{be}	14^d / 94^e 23^{bd} / 152^{be}	- -
Power efficiency (TOPS/W)	1.08^d / 8.88^e 1.08^{cd} / **8.88**^{ce}	0.49^d / 3.26^e **1.16**^{cd}/ 7.79^{ce}	1.86^d -

^{a}1 GMACS= 2 GOPS

^{b}Technology scaling $(\frac{process}{40nm})$

^{c}Normalized power efficiency $=$ power efficiency$\times(\frac{process}{40nm}) \times (\frac{Voltage}{0.99V})^2$.

^{d}The peak throughput for computing all the operations including zeros.

^{e}The logical throughput with zero skipping on ENet [8] [12].

[1] D. Im et al., “DT-CNN: dilated and transposed convolution neural network accelerator for real-time image segmentation on mobile devices,” in IEEE ISCAS, May 2019 pp. 1-5.

[2] W. Liu, J. Lin and Z. Wang, ”USCA: A unified systolic convolution array architecture for accelerating sparse neural network,” in IEEE ISCAS, May 2019 pp. 1-5.

Conclusion

- Proposed high performance hardware to support dilated and transposed convolutions.
- Decompose input and weight matrices to convert sparse computations to dense computations.
- Performance
  - Cut down 87.8% of the cycle count and 8.2X speedup over ideal dense CNN.
  - The area efficiency is up to 5.79X higher and the power efficiency is up to 4.77X than other designs for segmentation.

Thanks for your attention!