TOSHIBA

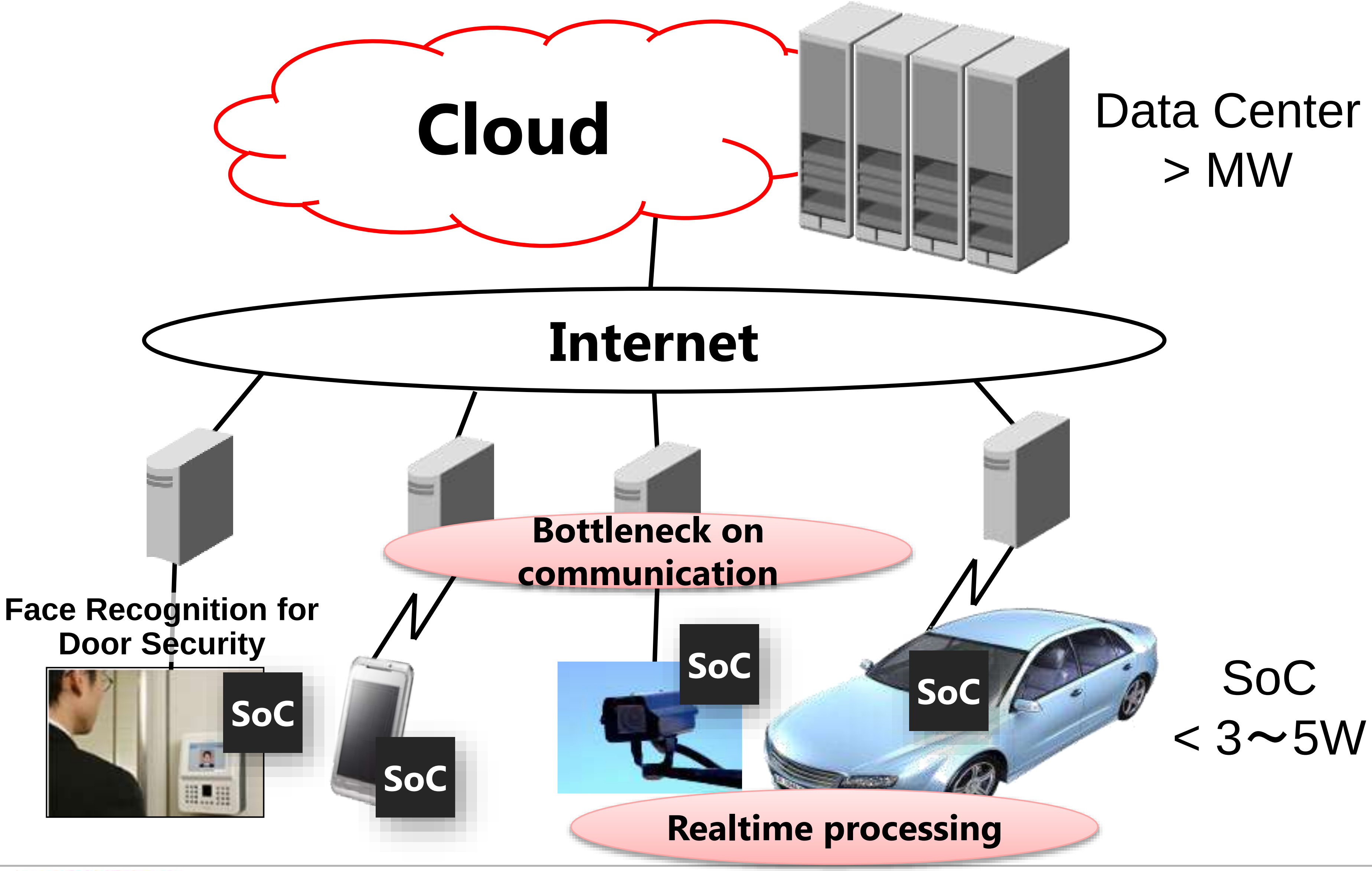
Leading Innovation >>>

Efficient Implementations of Deep Neural Network Hardware

Takashi Miyamori, General Manager
Center for Semiconductor Research & Development

Toshiba Electronic Devices & Storage Corporation

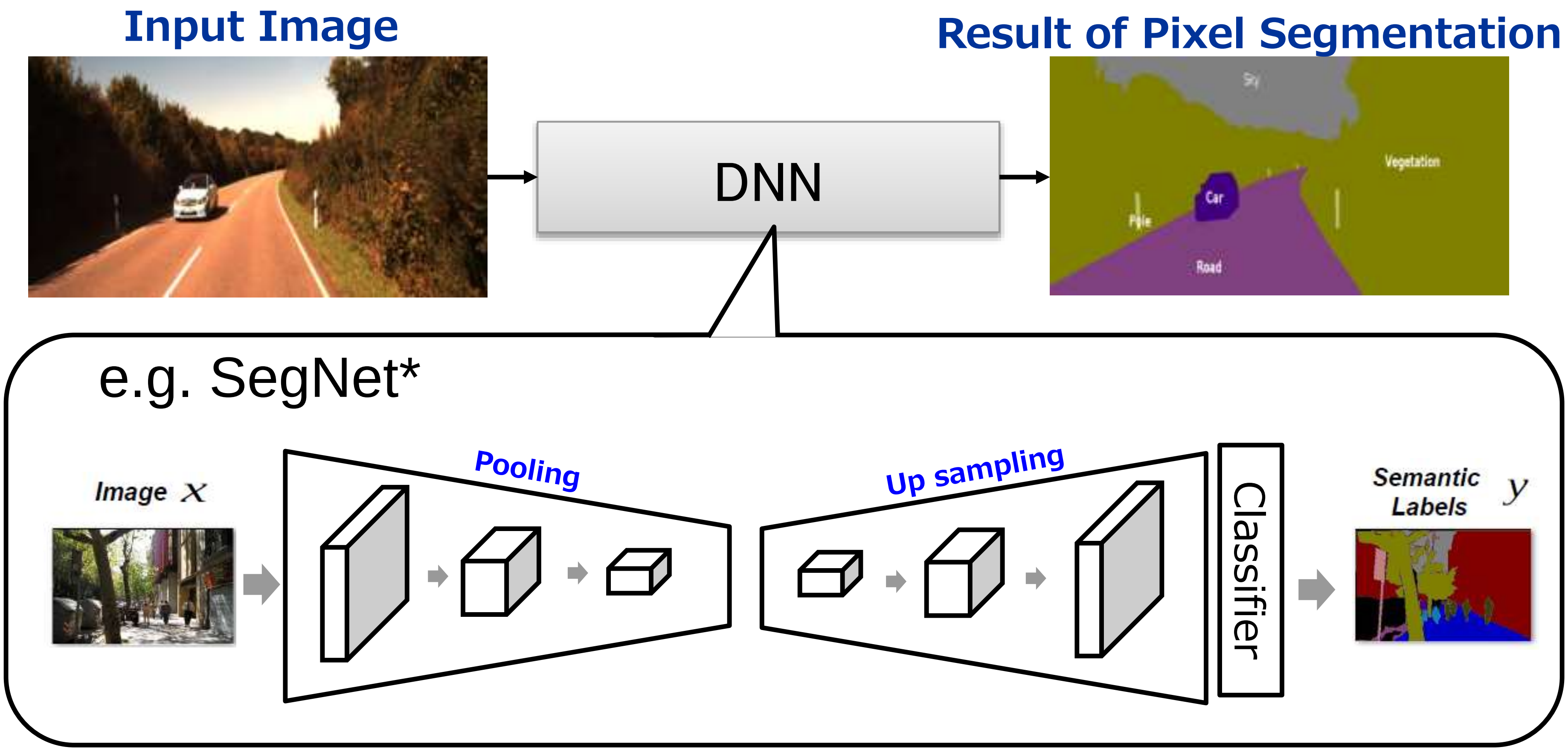
Deep Learning Everywhere



TOSHIBA
Leading Innovation >>>

Semantic Segmentation

- Classify objects in each pixel



Input Image

DNN

Result of Pixel Segmentation

e.g. SegNet*

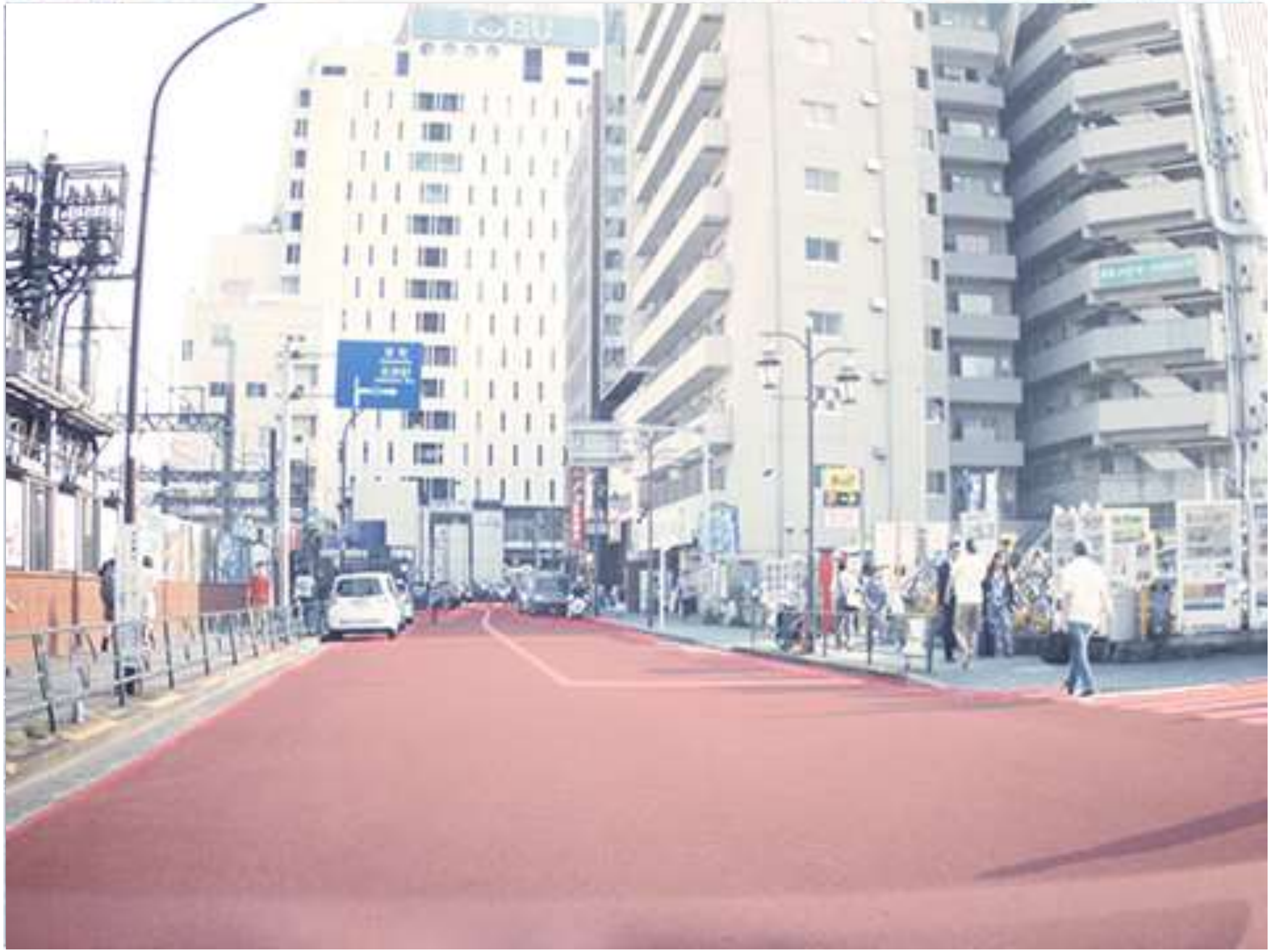
Pooling

Up sampling

Classifier

*) Badrinarayanan, V., Kendall, A., Cipolla, R.: SegNet: A deep convolutional encoder-decoder architecture for image segmentation. arXiv preprint arXiv:1511.00561 (2015)

Road Detection by DNN



Outline

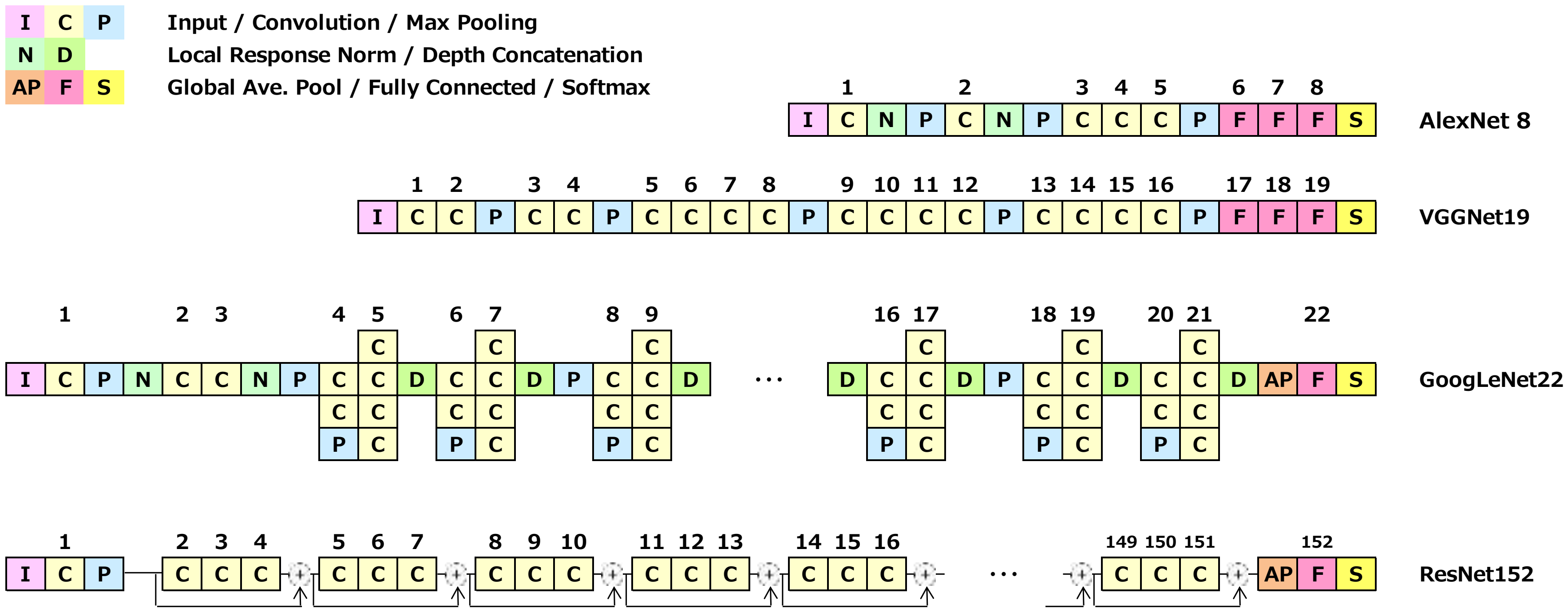
- Technology Trends
- LOGNET: energy-efficient neural networks using logarithmic computation (Stanford Univ. & Toshiba) [ICASSP 2017]
- TDNN: Time-Domain Neural Network (Toshiba) [A-SSCC 2016]

Efficient DNN Implementations

- **Improvement of Network Models**
  - GoogLeNet, ResNet

- **Reduction of Parameters（# of data, bit width） and Compression**
  - Deep Compression (Stanford): Pruning, Quantization, Huffman Coding
  - Binarized Neural Networks (Univ. of Montreal)

Improvement of Network Models


I C P
N D
AP F S
Input / Convolution / Max Pooling
Local Response Norm / Depth Concatenation
Global Ave. Pool / Fully Connected / Softmax
1 2 3 4 5 6 7 8
I C N P C N P C C C P F F F S
AlexNet 8
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
I C C P C C P C C C C P C C C C P C C C C P F F F S
VGGNet19
1 2 3 4 5 6 7 8 9 … 16 17 18 19 20 21 22
I C P N C C N P C C D C C D P C C D … D C C D P C C D C C D AP F S
C C C C P C
GoogLeNet22
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 … 149 150 151 152
I C P C C C + C C C + C C C + C C C + C C C + … + C C C + AP F S
ResNet152

Model	AlexNet	VGG	GoogLeNet	ResNet
Organization	Univ. of Tronto	Oxford Univ.	Google	Microsoft Research Asia
Year	2012	2014	2014	2015
ILSVRC* Error Rate	15.30%	7.33%	6.66%	3.57%
# of Layers	8	19	22	152
# of Parameters[M]	62.4	144	7.0	56.0
# of Operations[B]	1.14	19.6	1.5	11.3

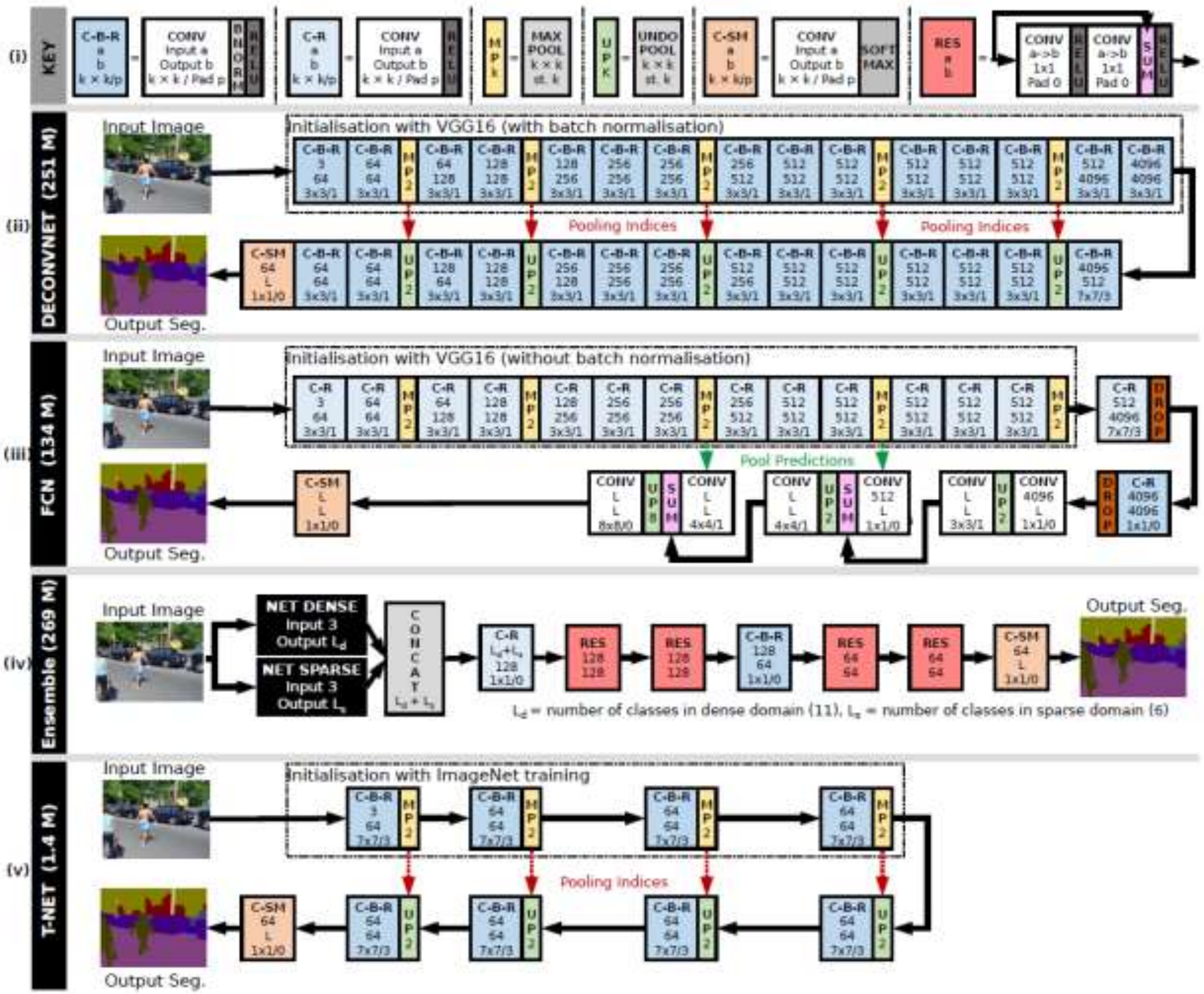
*) ILSVRC: ImageNet Large Scale Visual Recognition Challenge

Based on slides of Dr. Momose, Hokkaido Univ.
https://www.semiconportal.com/archive/contribution/applications/160804-neurochip2-2.html

Road Scene Semantic Segmentation

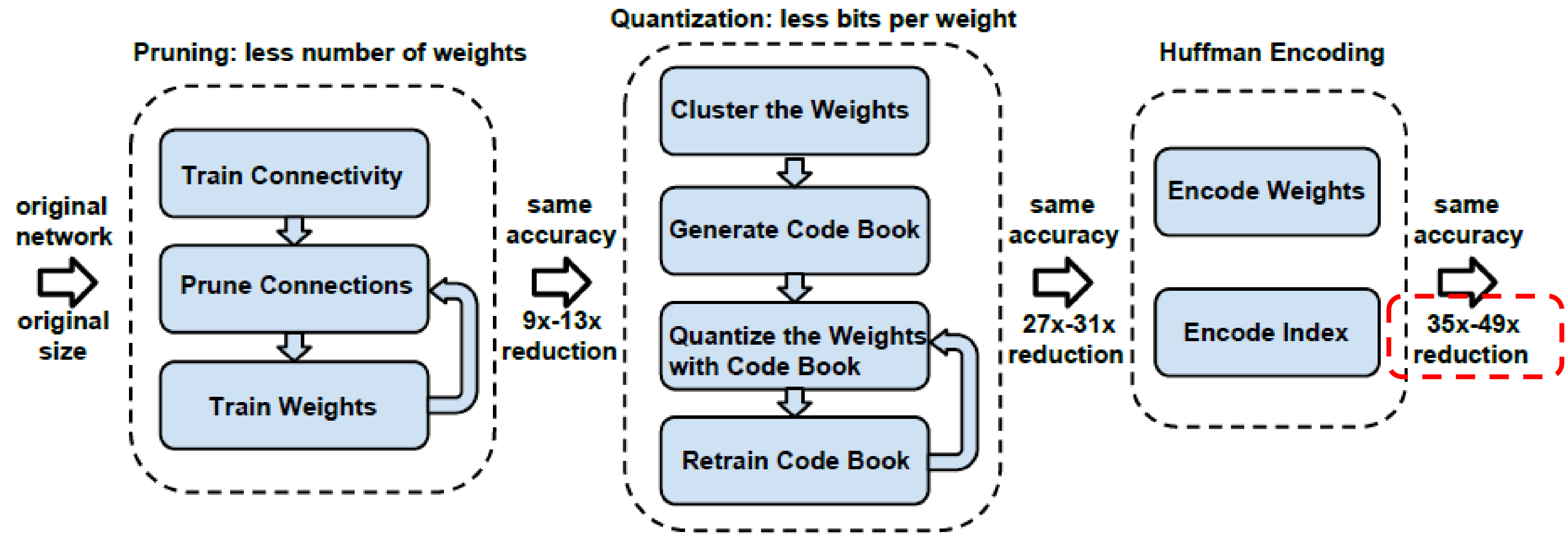
	# of Parameters
DECONVNET	251M
FCN	134M
Ensemble	269M
T-Net*	1.4M

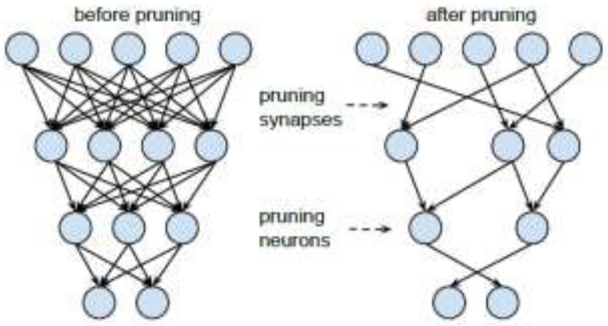
SegNet 30M

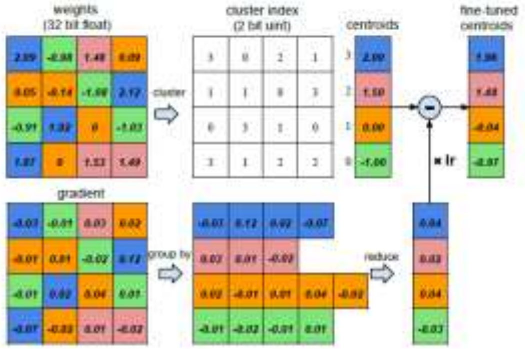


*) Training constrained deconvolutional networks for road scene semantic segmentation
G Ros, S Stent, PF Alcantarilla, T Watanabe, arXiv preprint, arXiv:1604.01545 (2016)

Deep Compression (Stanford)





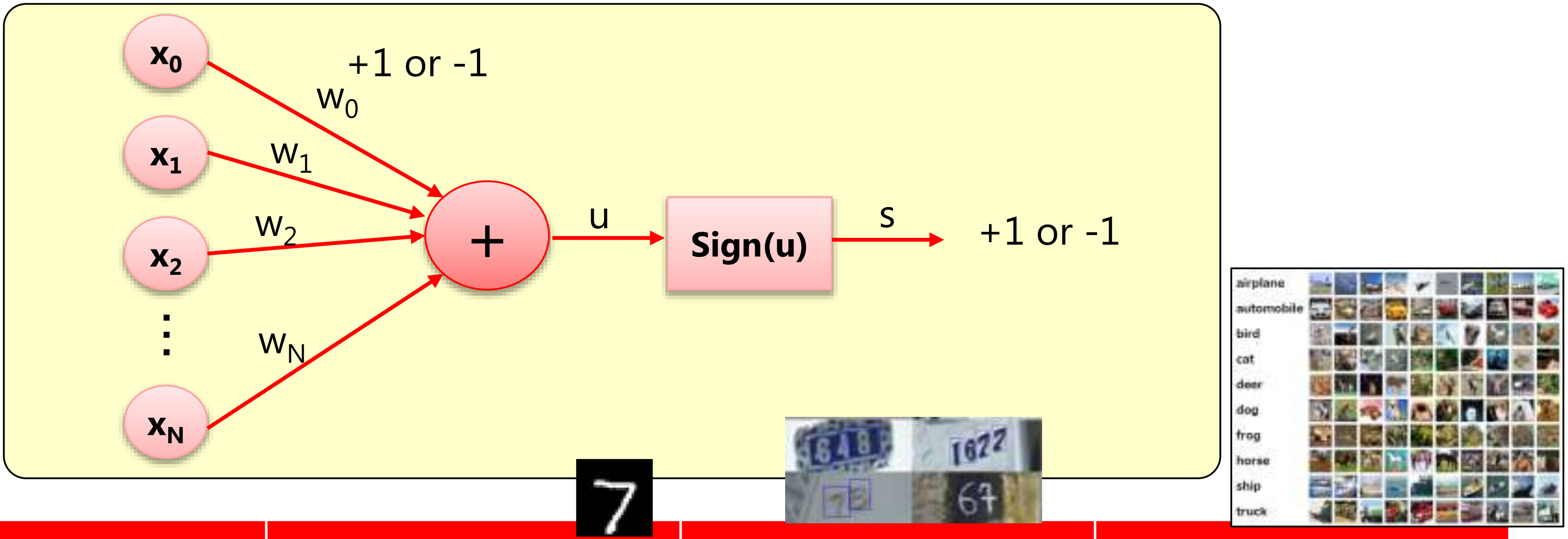


Song Han, Huizi Mao, and William J. Dally, “Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding,” arXiv preprint arXiv:1510.00149, 2015.

Song Han, Jeff Pool, John Tran, and William Dally, “Learning both weights and connections for efficient neural network,” in Proceedings of Advances in Neural Information Processing Systems 28 (NIPS2015), 2015, pp. 1135-1143.

Binarized Neural Networks (Univ. of Montreal)

- **Neural networks with binary weights and activations (+1/-1) except for the first and the last layers**



Error Rate	MNIST (Handwritten digits)	SVHN (Street View House Numbers)	CIFAR-10
BNN (Theano)	0.96%	2.80%	11.40%
Maxout	0.94%	2.47%	11.68%
Gated pooling	-	1.69%	7.62%

I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio,
“Binarized neural networks,” Advances in Neural Information Processing Systems 29, 2016, pp. 4107-4115..

Outline

- Technology Trends

- **LOGNET: energy-efficient neural networks using logarithmic computation (Stanford Univ. & Toshiba) [ICASSP 2017] [arXiv:1603.01025]**

- TDNN: Time-Domain Neural Network (Toshiba) [A-SSCC 2016]

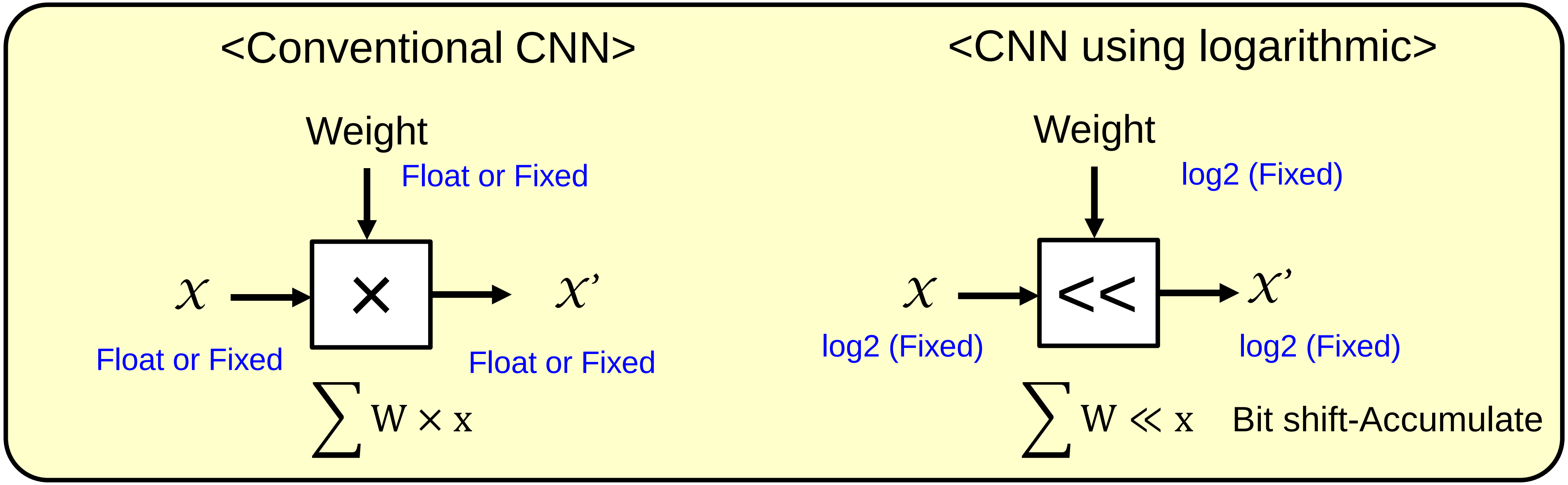
Motivation of LOGNET

- **To realize energy-efficient neural networks**
  - Data representation with fewer bits
  - Eliminate multiplications

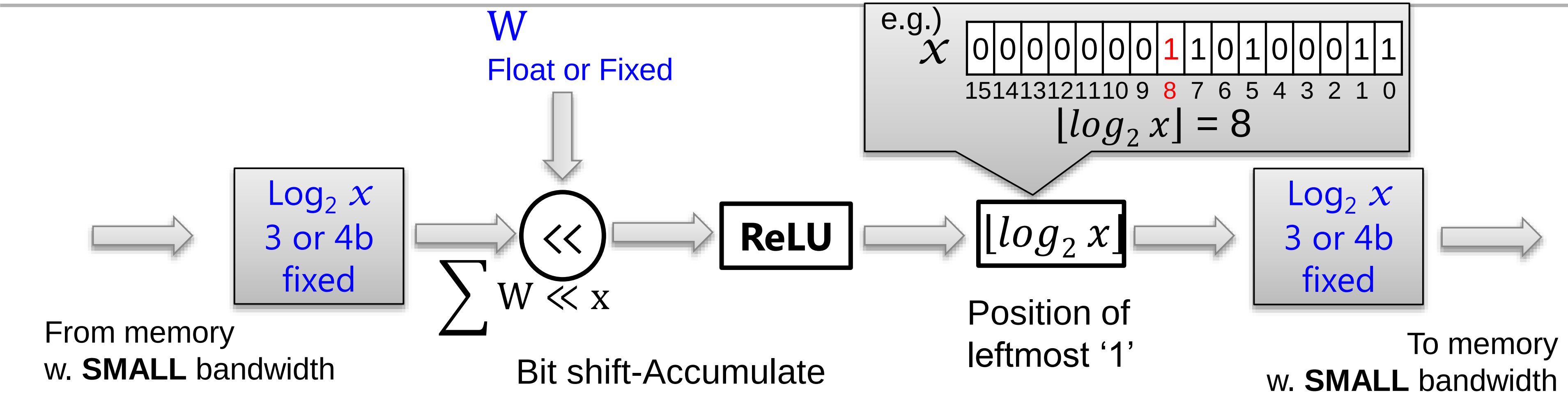
Proposal

- Use logarithmic encoding

Efficient implementation for dedicated HW in SoC rather than SW on GPU



Evaluation of Proposed 1



W

Float or Fixed

e.g.) x

0	0	0	0	0	0	0	1	1	0	1	0	0	0	1	1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

$\lfloor log_2\, x \rfloor = 8$

Log$_2$ x 3 or 4b fixed

$\ll$

$\sum W \ll x$

ReLU

$\lfloor log_2\, x \rfloor$

Log$_2$ x 3 or 4b fixed

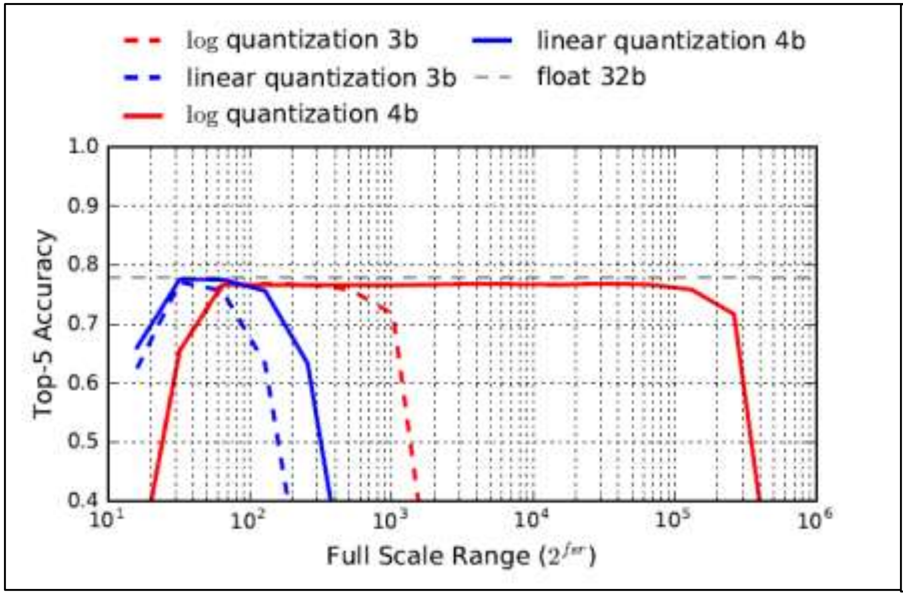
From memory w. **SMALL** bandwidth

Bit shift-Accumulate

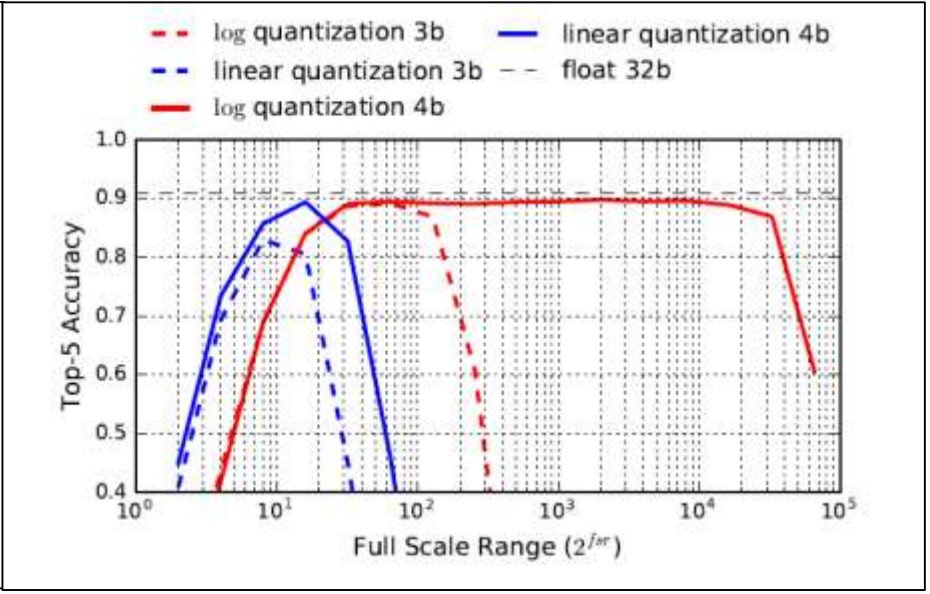
Position of leftmost ‘1’

To memory w. **SMALL** bandwidth

- Evaluation results: ILSVRC-2012 using Chainer

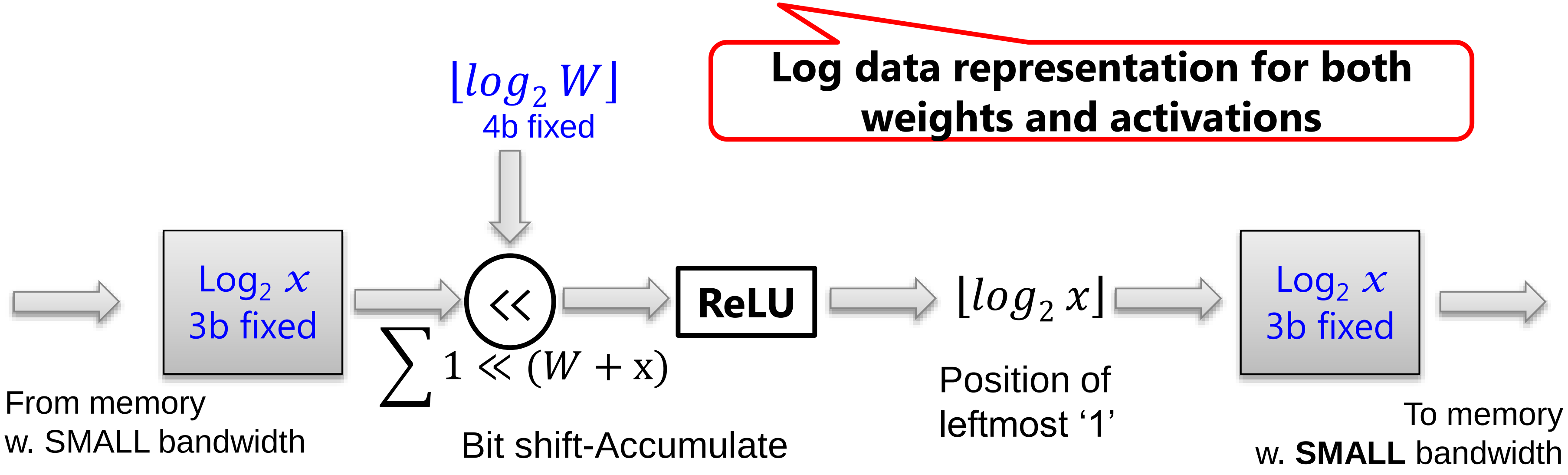


Top5 accuracy vs Full scale range: AlexNet



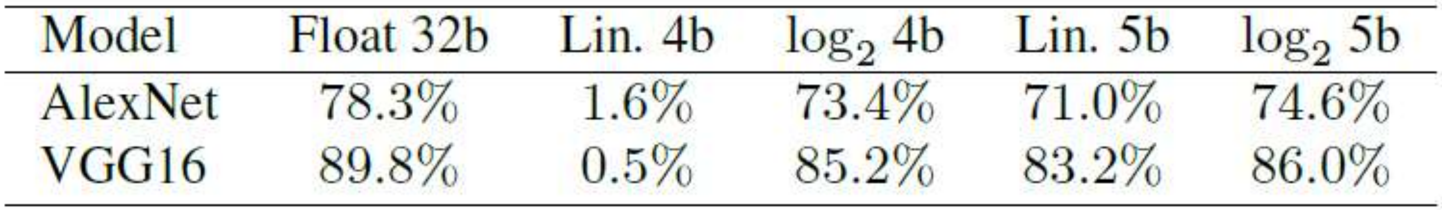
Top5 accuracy vs Full scale range: VGG16

Evaluation of Proposed 2



Log data representation for both weights and activations

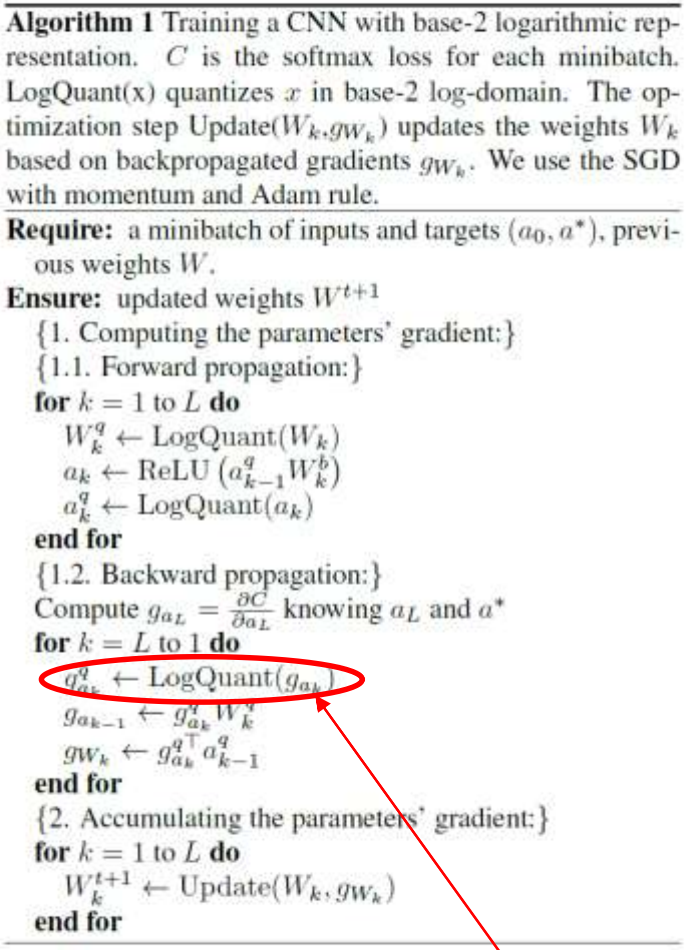
- **Top-5 accuracies after linear and log2 encoding on all layers' weight without retraining**



Model	Float 32b	Lin. 4b	$\log_2$ 4b	Lin. 5b	$\log_2$ 5b
AlexNet	78.3%	1.6%	73.4%	71.0%	74.6%
VGG16	89.8%	0.5%	85.2%	83.2%	86.0%

Training with Logarithmic Representation

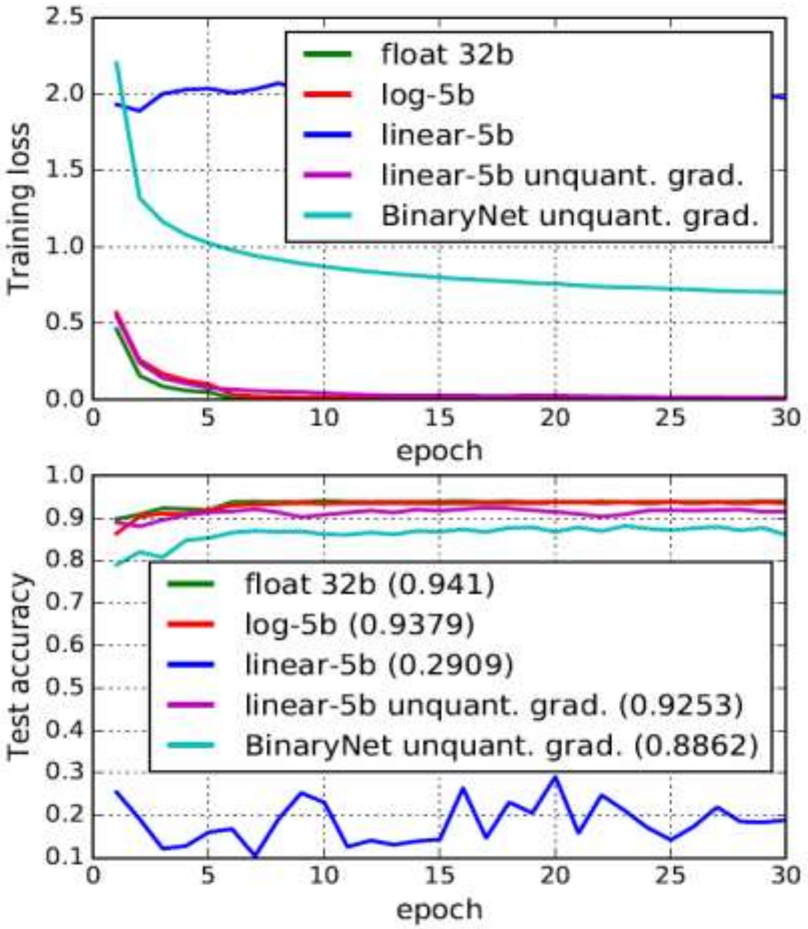
- Training Algorithm



Algorithm 1 Training a CNN with base-2 logarithmic representation. C is the softmax loss for each minibatch. LogQuant(x) quantizes x in base-2 log-domain. The optimization step Update(W_k,g_{W_k}) updates the weights W_k based on backpropagated gradients g_{W_k}. We use the SGD with momentum and Adam rule.

```
Require: a minibatch of inputs and targets $(a_0, a^*)$, previous weights $W$.
Ensure: updated weights $W^{t+1}$
  {1. Computing the parameters' gradient:}
  {1.1. Forward propagation:}
  for $k = 1$ to $L$ do
    $W_k^q \leftarrow \mathrm{LogQuant}(W_k)$
    $a_k \leftarrow \mathrm{ReLU}\,(a_{k-1}^q W_k^b)$
    $a_k^q \leftarrow \mathrm{LogQuant}(a_k)$
  end for
  {1.2. Backward propagation:}
  Compute $g_{a_L} = \frac{\partial C}{\partial a_L}$ knowing $a_L$ and $a^*$
  for $k = L$ to 1 do
    $g_{a_k}^q \leftarrow \mathrm{LogQuant}(g_{a_k})$
    $g_{a_{k-1}} \leftarrow g_{a_k}^q W_k^q$
    $g_{W_k} \leftarrow g_{a_k}^{q\top} a_{k-1}^q$
  end for
  {2. Accumulating the parameters' gradient:}
  for $k = 1$ to $L$ do
    $W_k^{t+1} \leftarrow \mathrm{Update}(W_k, g_{W_k})$
  end for
```

- **CIFAR10 database**
- **VGG-like network**



Quantize gradients

Enables end-to-end training using logarithmic representation at 5b level

Outline

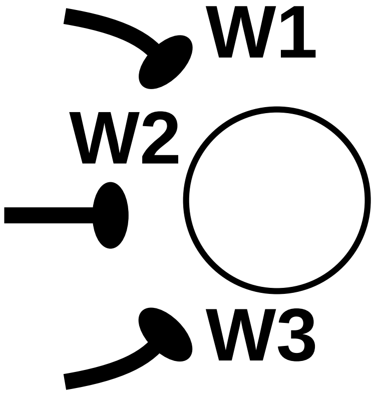
- Technology Trends

- LOGNET: energy-efficient neural networks using logarithmic computation (Stanford Univ. & Toshiba) [ICASSP 2017]

- **TDNN: Time-Domain Neural Network (Toshiba) [A-SSCC 2016]**

Why the brain is so energy efficient?

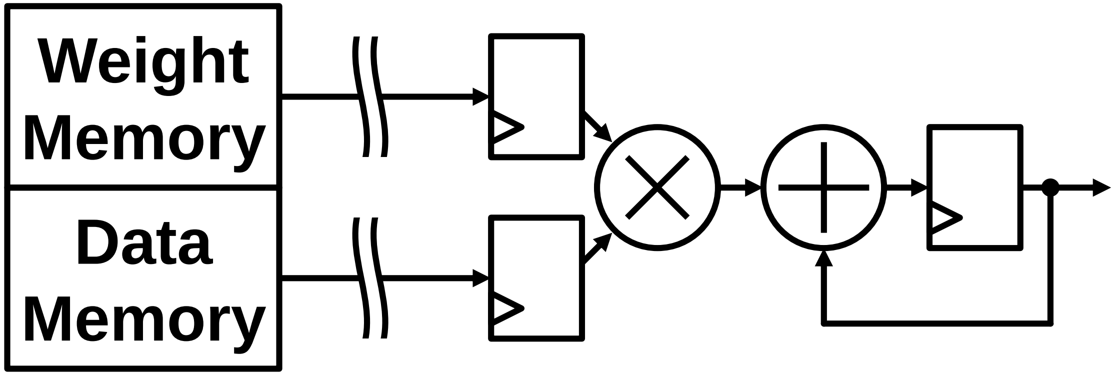
Brain



Weight is built into each synapse. Don’t need to move weight at all.

Power efficient!!

Conventional computer



Load weight from memory for EVERY calculation.

Power hungry!!

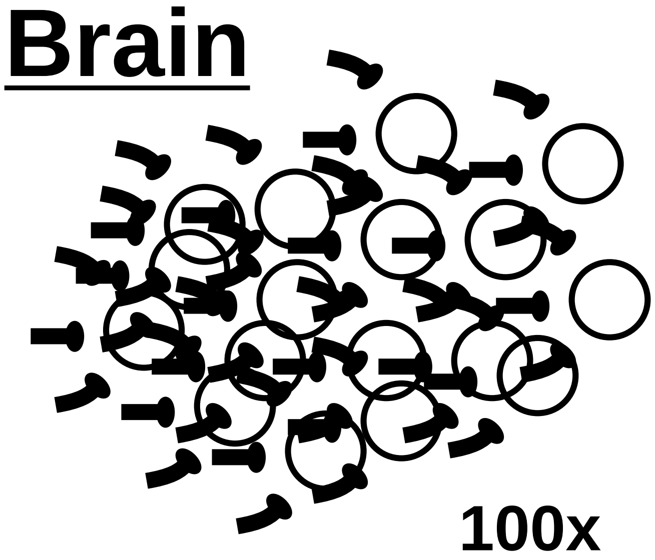
Energy consumption

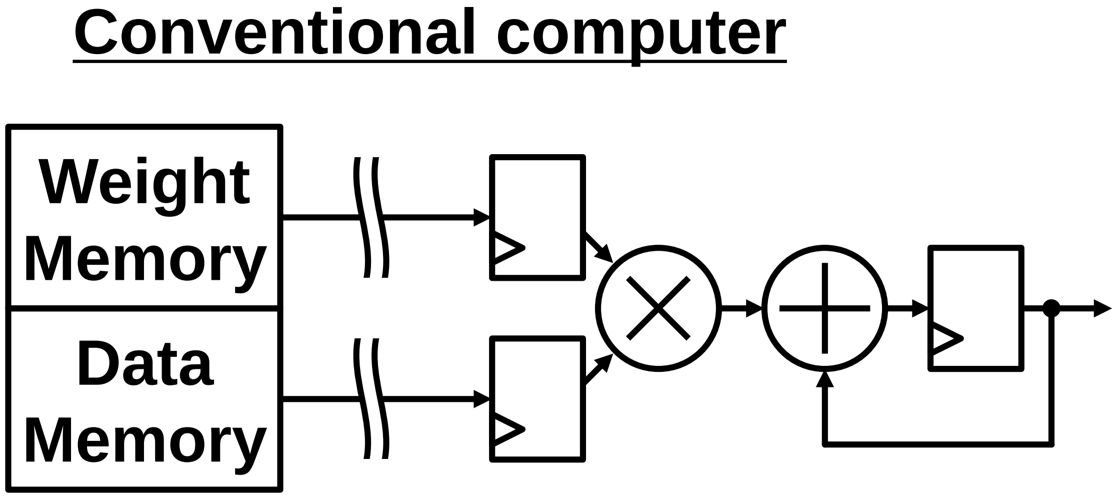
Operation	Relative Cost (Energy)
32 bit int ADD	1
32 bit int MULT	31
32 Register File	10
32 bit SRAM	50
32 bit DRAM Memory	6400

Cf. Mark Horowitz. Energy table for 45nm process, Stanford VLSI wiki.

How about hardware efficiency?

e.g. The number of weights → 100x





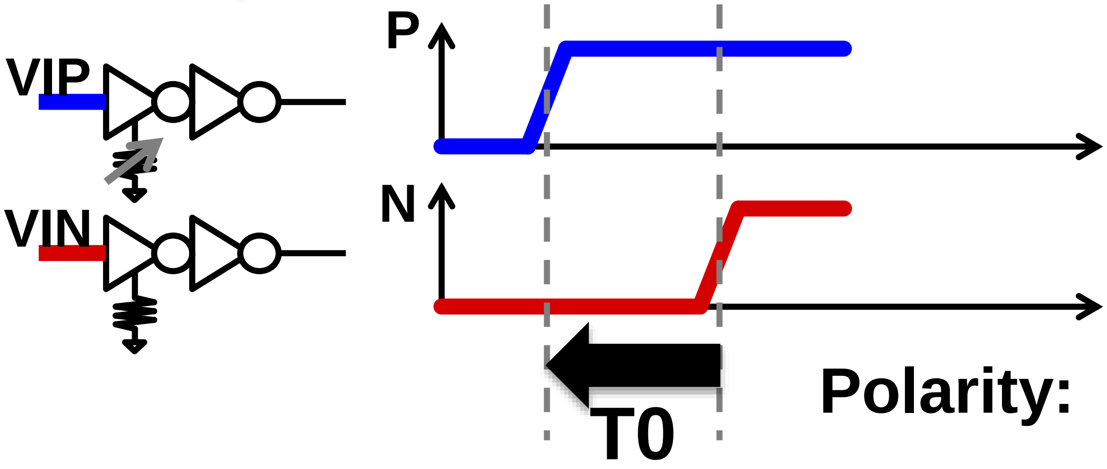
Need to have 100x processing elements. Because each processing element (PE) is dedicated to each weight.

Need to minimize each PE!!

Our strategy

- In order to maximize the energy efficiency, we propose to employ fully spatially unrolled architecture (like the brain).
- In order to minimize the hardware size, we propose to employ Time Domain Analog and digital Mixed Signal processing (TDAMS) [11].

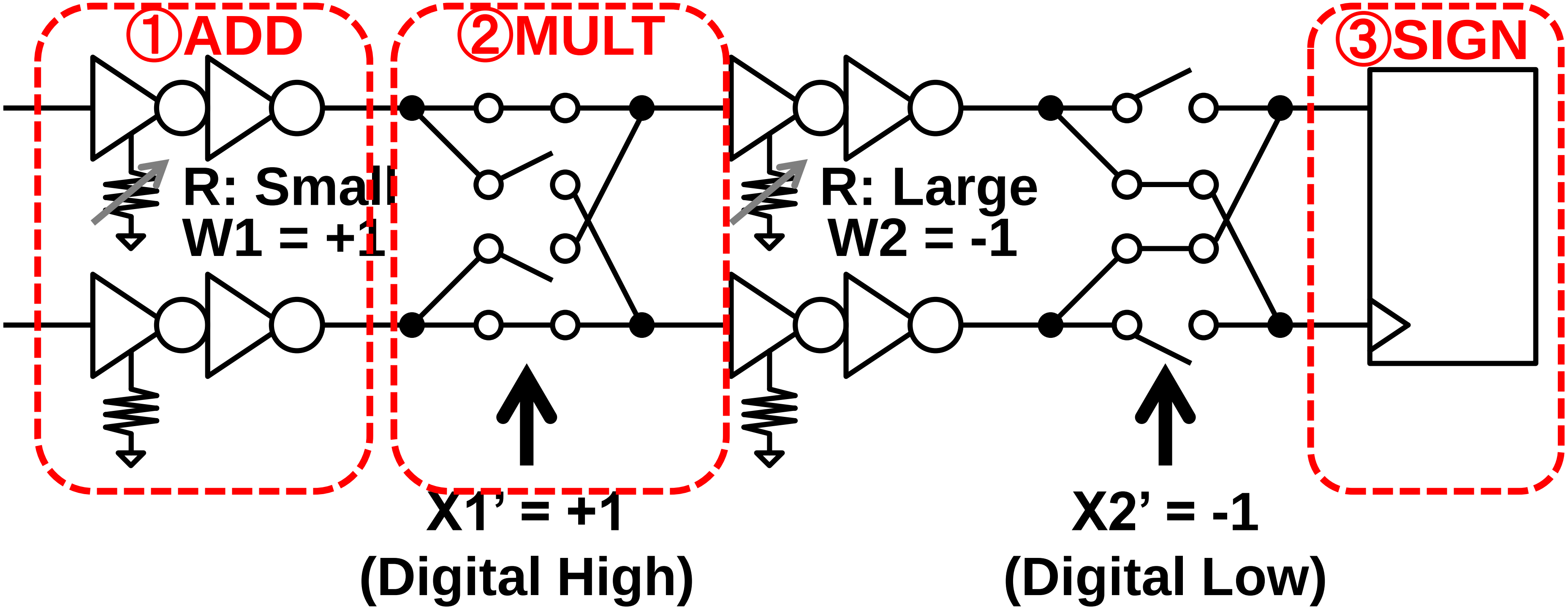
→ **TDNN** (Time Domain Neural Network)



Value is represented by the time difference between edges.

Polarity: +, if P is earlier than N
-, if P is later than N

TDAMS ー Convolution



$$(W1 \times X1' + W2) \times X2' = W1X1'X2' + W2X2'$$

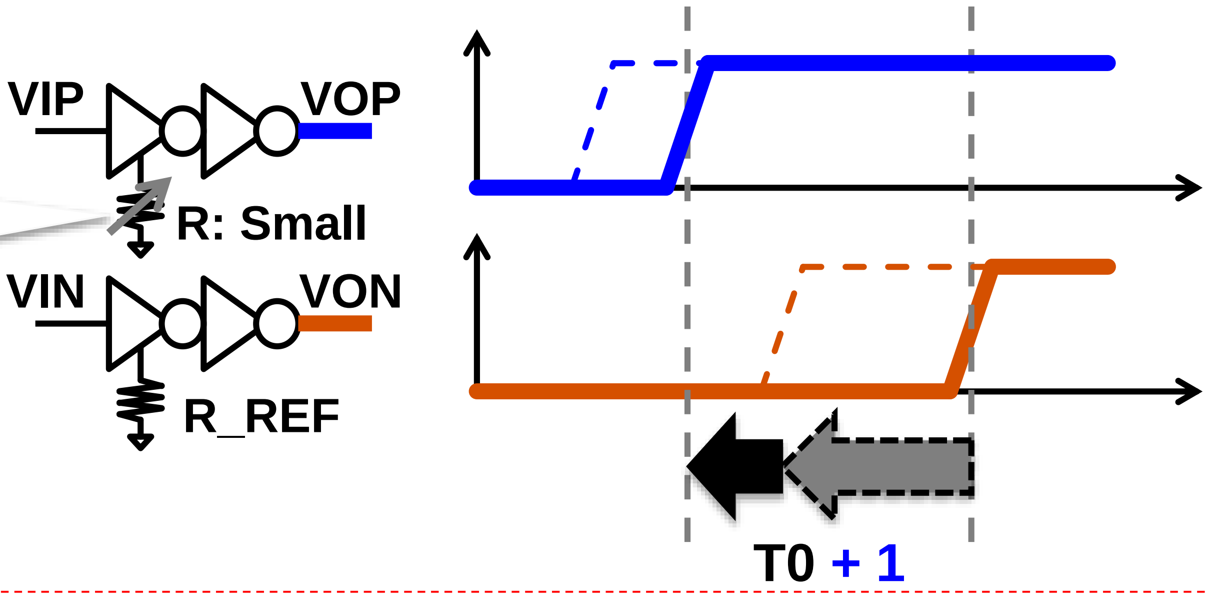
↓ $X\_i' = XOR(X\_i, X\_i+1)$

SIGN(W1X1 + W2X2)

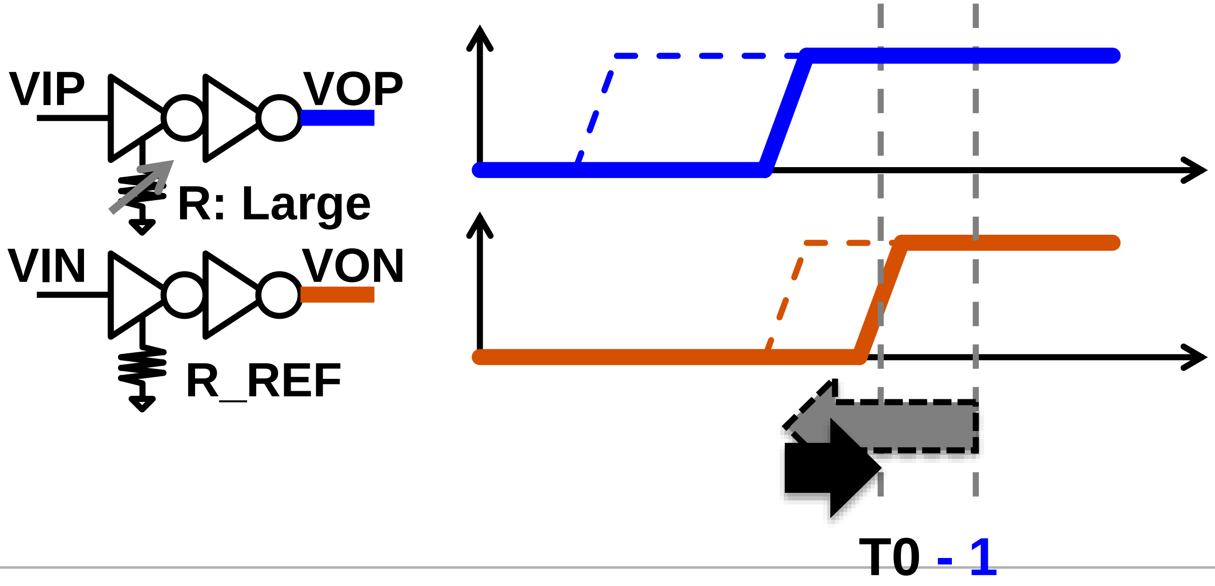
TDAMS - ADD

+ 1

Delay time is proportional to value of resistance

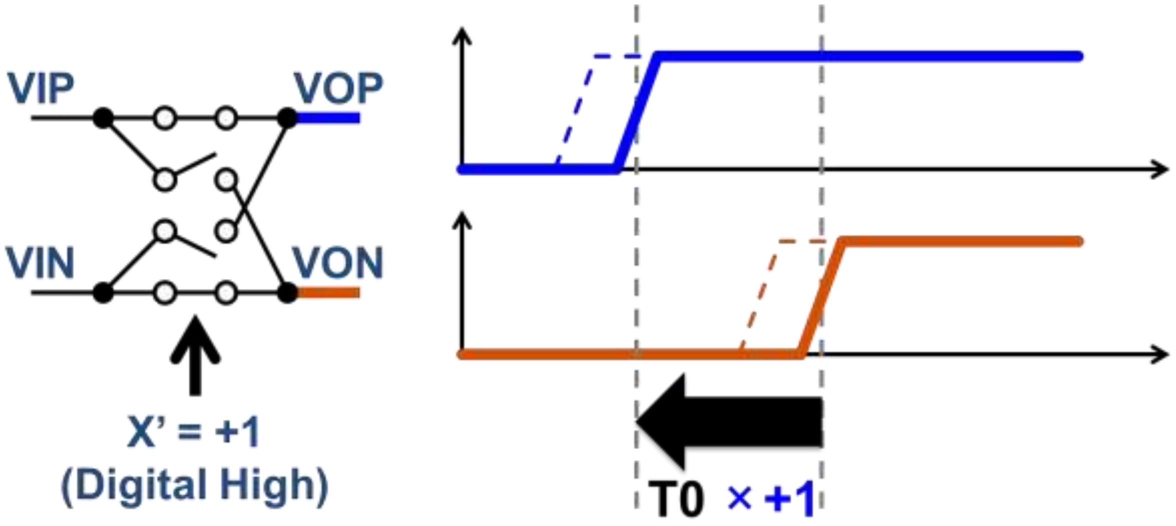


- 1

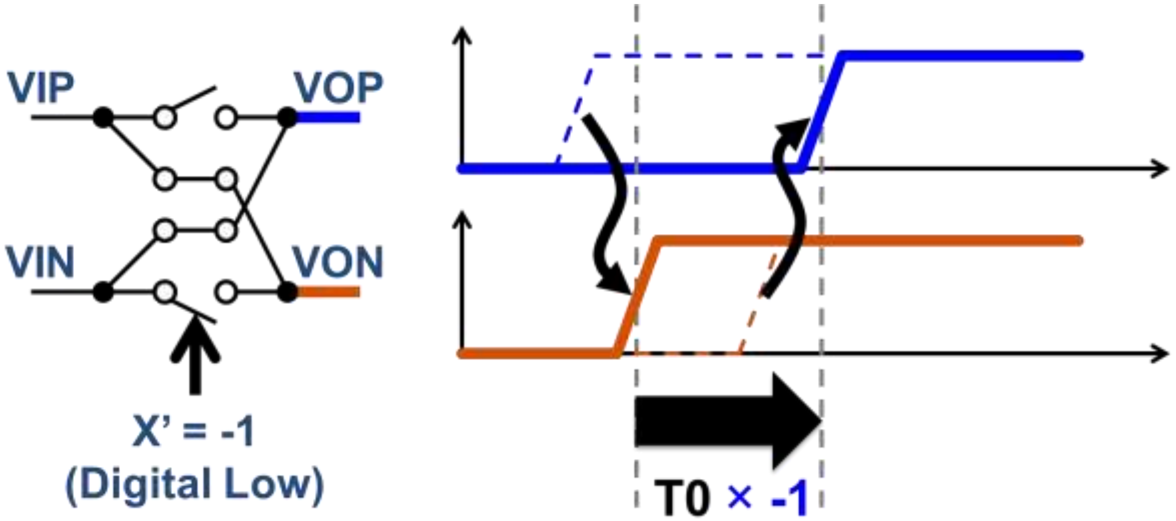


TDAMS - Multiplication

x =+1

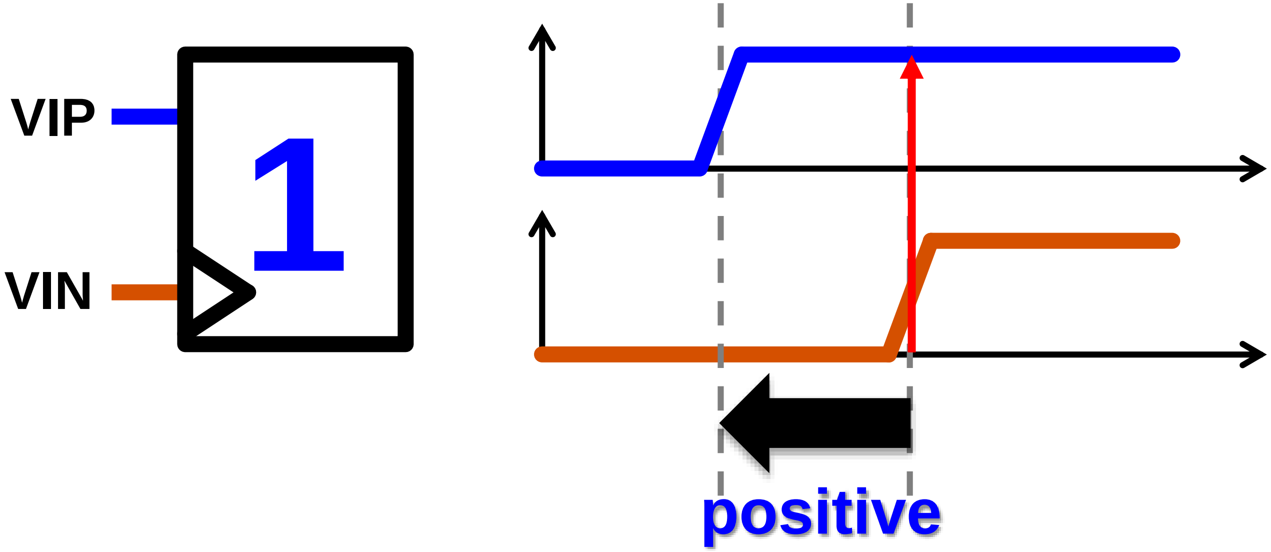


x =-1

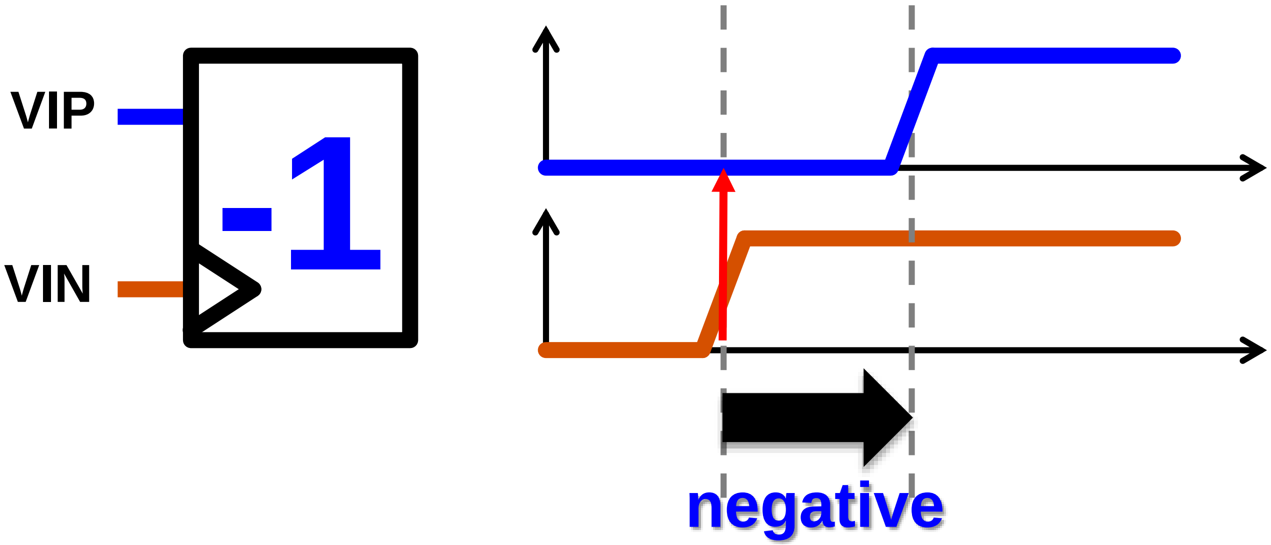


TDAMS ~ Activation (SIGN)

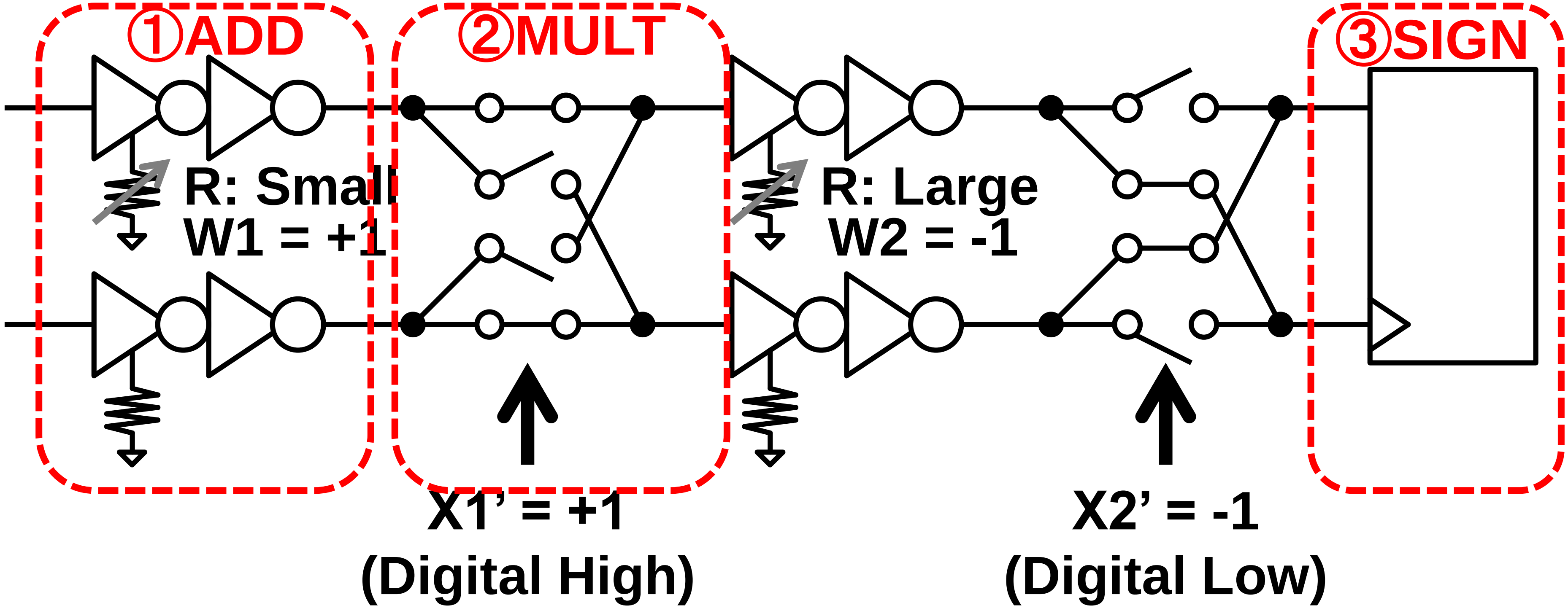
Positive
(+1)



Negative
(-1)



TDAMS ー Convolution

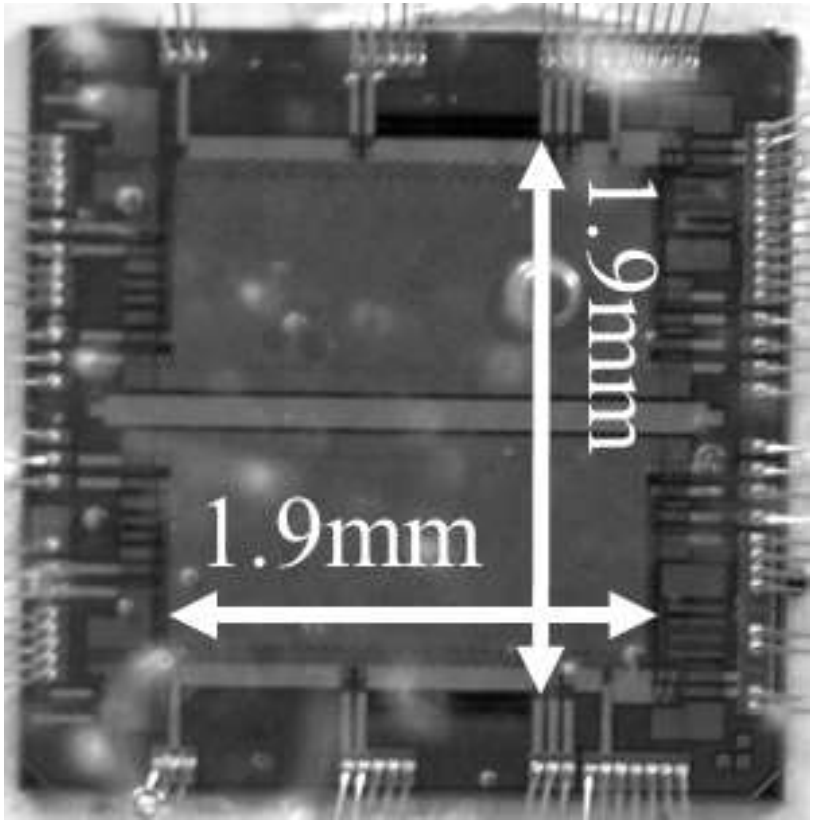


$$(W1 \times X1' + W2) \times X2' = W1X1'X2' + W2X2'$$

X_i’ = XOR(X_i, X_i+1)

SIGN(W1X1 + W2X2)

Chip photograph



65nm CMOS technology

# of processing elements: 32768

Experimental results

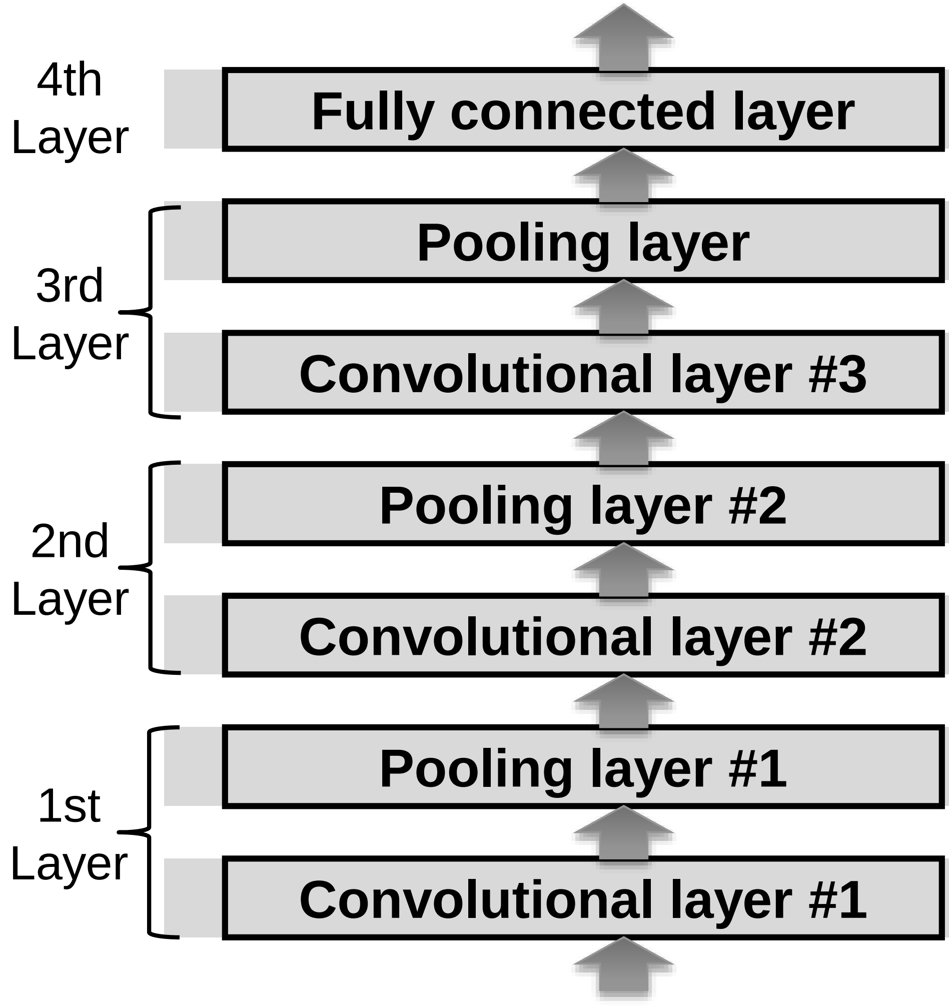
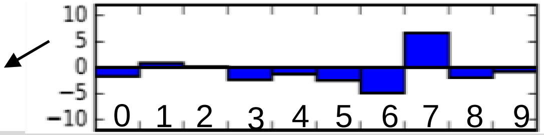


		Image (Data)	Kernel (Weight)
		10	
4th Layer	Fully connected layer		128
		128	
3rd Layer	Pooling layer		3×3
		128×3×3	
	Convolutional layer #3		64×3×3
		64×5×5	
2nd Layer	Pooling layer #2		3×3
		64×11×11	
	Convolutional layer #2		16×3×3
		16×13×13	
1st Layer	Pooling layer #1		3×3
		16×30×30	
	Convolutional layer #1		1×3×3
		1×32×32	



Experimental results

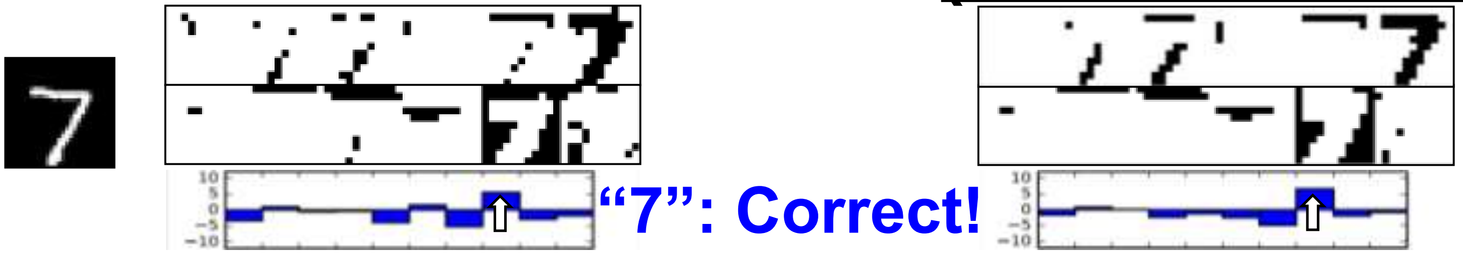
	Image (Data)	Kernel (Weight)
	10	
Fully connected layer		128
	128	
Pooling layer		
	128	
Convolutional la		×3×3
	3	
		3×3
	64×11×11	
Convolutional layer #2		16×3×3
	16×13×13	
Pooling layer #1		3×3
	16×30×30	
Convolutional layer #1		1×3×3
	1×32×32	

Executed on TDNN

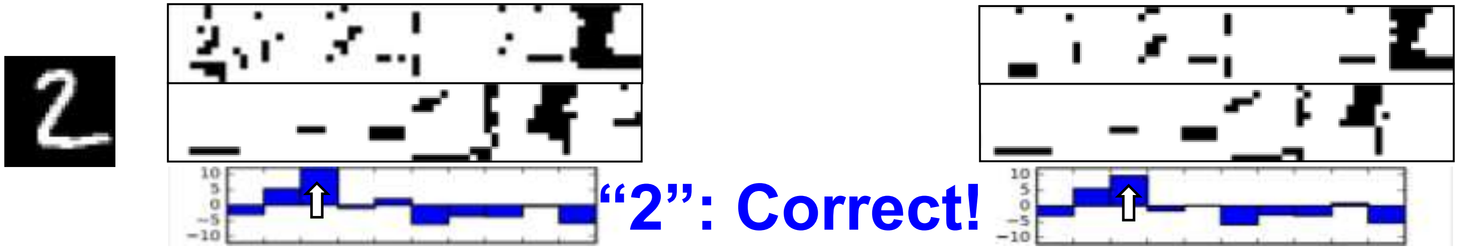
Experimental results

Measured

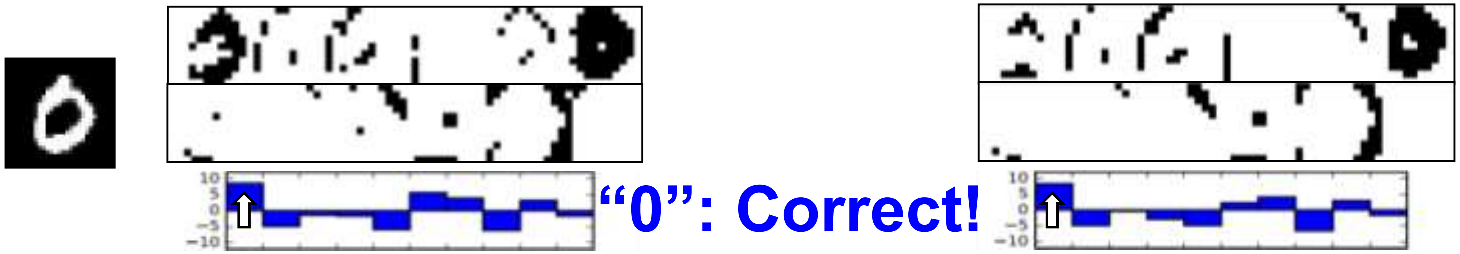
Expected (Simulated on PC)



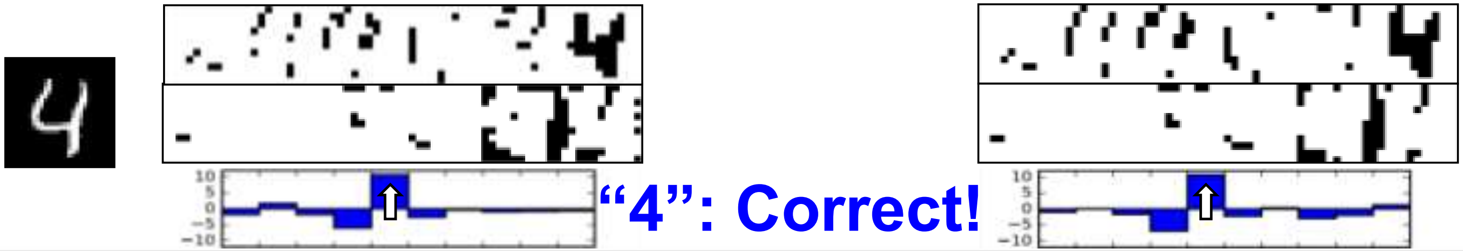
“7”: Correct!



“2”: Correct!



“0”: Correct!



“4”: Correct!

Tested 1000 images. Same accuracy as for simulation.

Performance comparison

Energy efficiency is 10x better than ISSCC 2016[3].

	Blueprint w/ ReRAM	Test chip w/ SRAM	GLS-VLSI 2015[1]	Science [2]	ISSCC2016 [3]
Tech.[nm]	65	65	65	28	40
Chip area [mm^2]	-	3.61[a]	1.31[a]	430	0.012
Energy efficiency [TSOp/s/W]	**48.2**[b]	**48.2**[b]	0.402	0.039[6] 0.4[7]	**3.86**[c]
Hardware efficiency[d] [GE/PE]	**3**	76.5	4641[a]	6.5	288

[a] core area including SRAM, [b] excludes external I/O, [c] excludes CML

[d] 1GE:1.44um^2 (65nm), 0.65 um^2 (40nm), 0.49 um^2 (28nm)

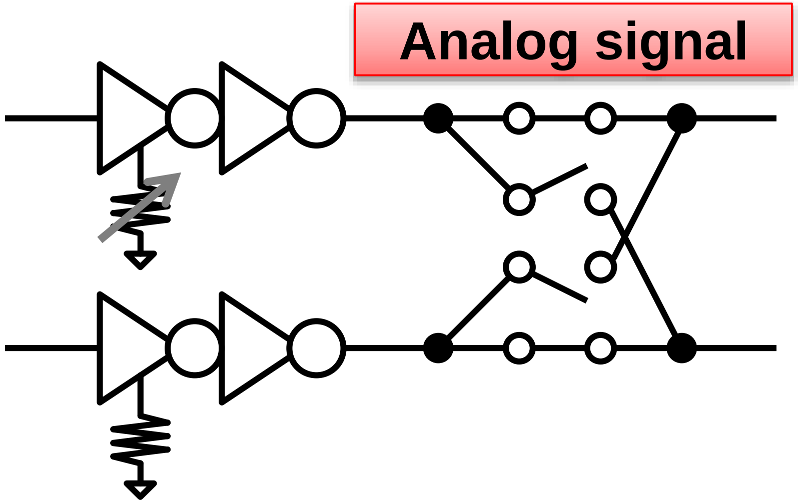
[1] L. Cavigelli and L. Benini, “Origami: A 803 gop/s/w convolutional network accelerator,” arXiv preprint arXiv: 1512.04295, 2015

[2] P. A. Merolla, et al., ”A million spiking-neuron integrated circuit with a scalable communication network and interface,” Science, vol. 345, no.6197, pp. 668-673, 2014.

[3] E. H. Lee and S. S. Wong, “A 2.5ghz 7.7tops/w switched-capacitor matrix multiplier with co-designed local memory in 40nm,” in ISSCC Dig. Tech. Papers, pp. 418-419, 2016.

Blue print with ReRAM

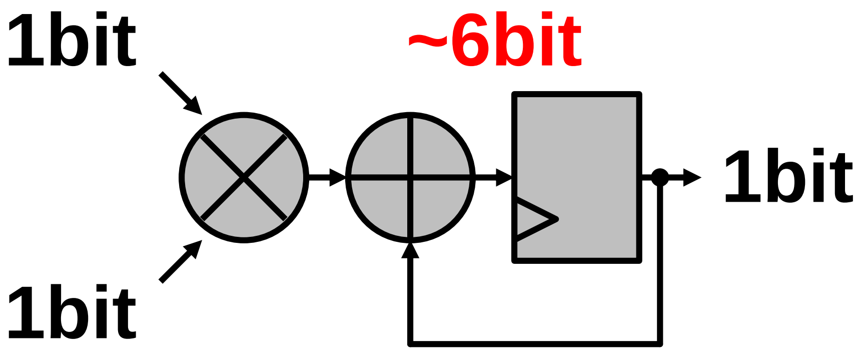
TDNN



= 3 2-input NAND

+ memory cell (e.g. ReRAM)

Conventional computer



>> 50 2-input NAND

1.5 μm^2 @28nm = 230M PEs / 4 cm^2

cf. *ResNet : 230M parameters**

*) K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *arXiv preprint* arXiv: 1512.03385, 2015.

Summary

- **Efficient Implementations of DNN are required for embedded systems and edge devices**
- **Efficient Implementations**
  - Improvement of Network Models.
    - Simple Network Models (e.g. GoogLeNet, ResNet)
  - Reduction of Parameters（# of data, bit width）and Compression
    - Deep Compression (Stanford): Pruning, Quantization, Huffman Coding
    - Binarized Neural Networks
- **Efficient Hardware Implementations**
  - LOGNET: energy-efficient neural networks using logarithmic computation
  - TDNN(Time Domain Neural Network)
    - Fully spatially unrolled architecture (like the brain).
    - Time Domain Analog and digital Mixed Signal processing (TDAMS)

TOSHIBA

Leading Innovation >>>