NAND-Net: Minimizing Computational Complexity of In-Memory Processing for Binary Neural Networks

Hyeonuk Kim, Jaehyeong Sim, Yeongjae Choi, and Lee-Sup Kim

School of Electrical Engineering

KAIST

Multimedia VLSI Laboratory

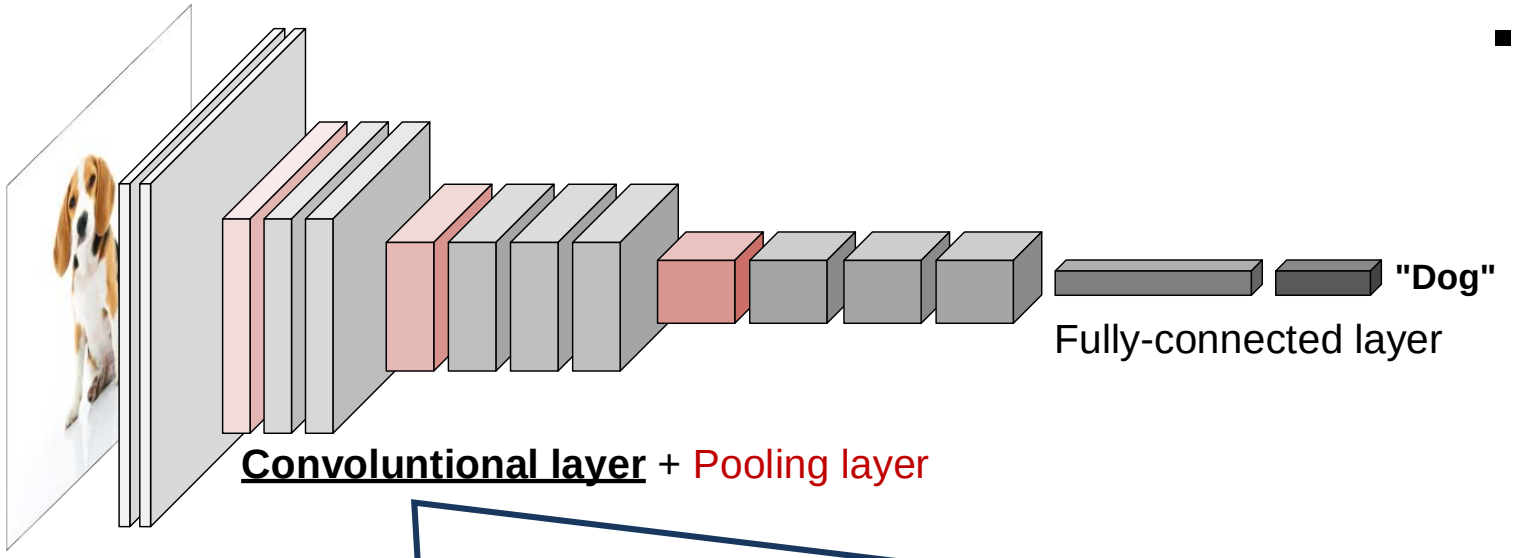
Contents

Contents

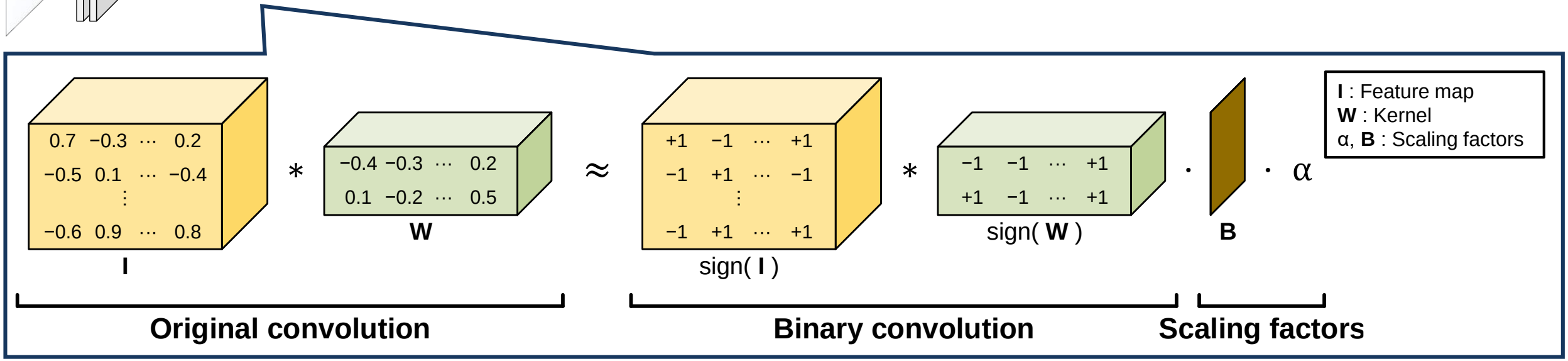
Popular deep learning technologies suffer from memory bottlenecks, which significantly degrade energy-efficiency.

How to solve memory bottlenecks efficiently?

Network Binarization

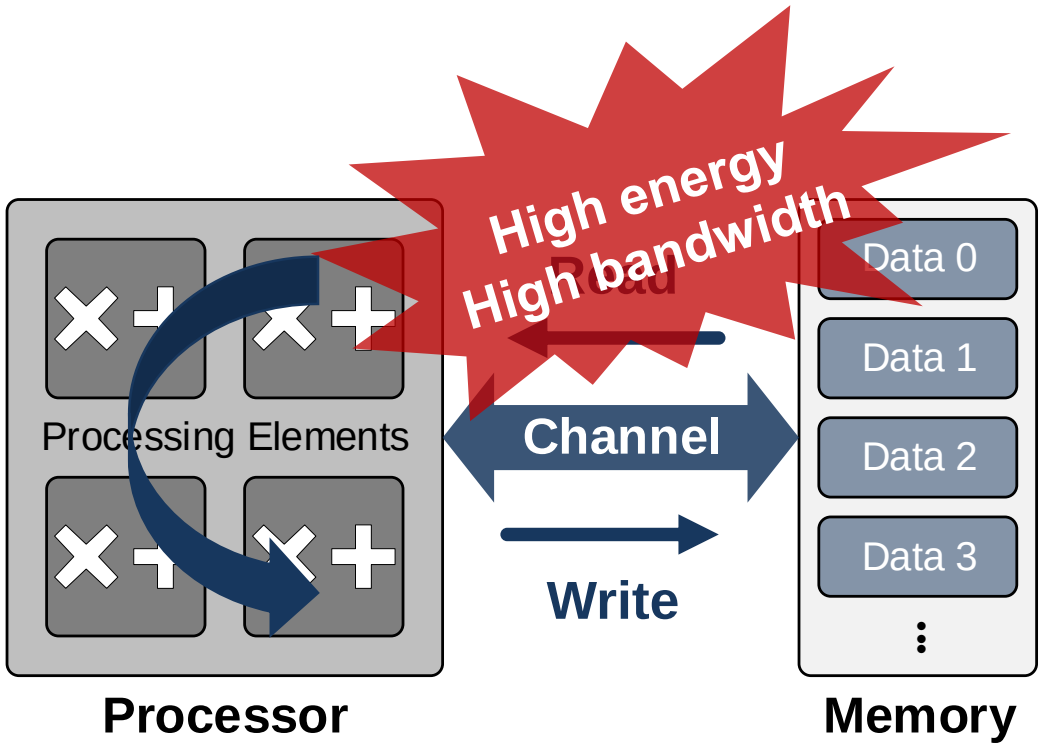


- **Binary Neural Network (XNOR-Net)**
  - ✓ Memory footprint reduction
    - e.g.) 16-bit fixed point → 1-bit binary
  - ✓ Computation simplification
    1) Multiplication → **Bitwise XNOR**
    2) Addition → **Popcount**

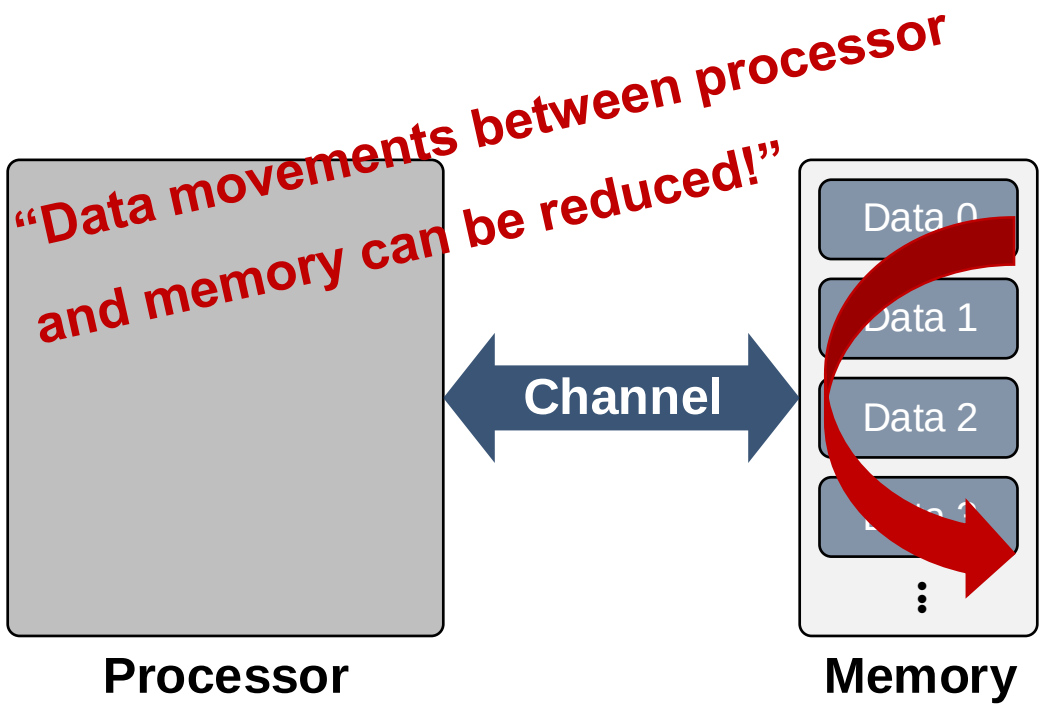


"XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks", ECCV 2016

In-Memory Processing



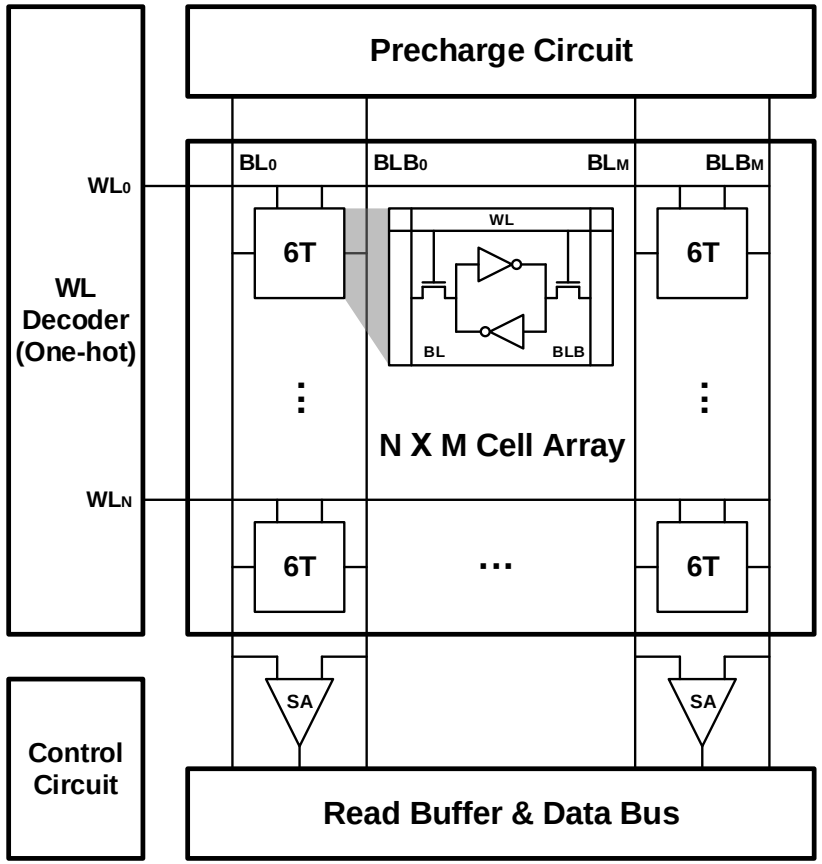
Conventional von Neumann Architecture



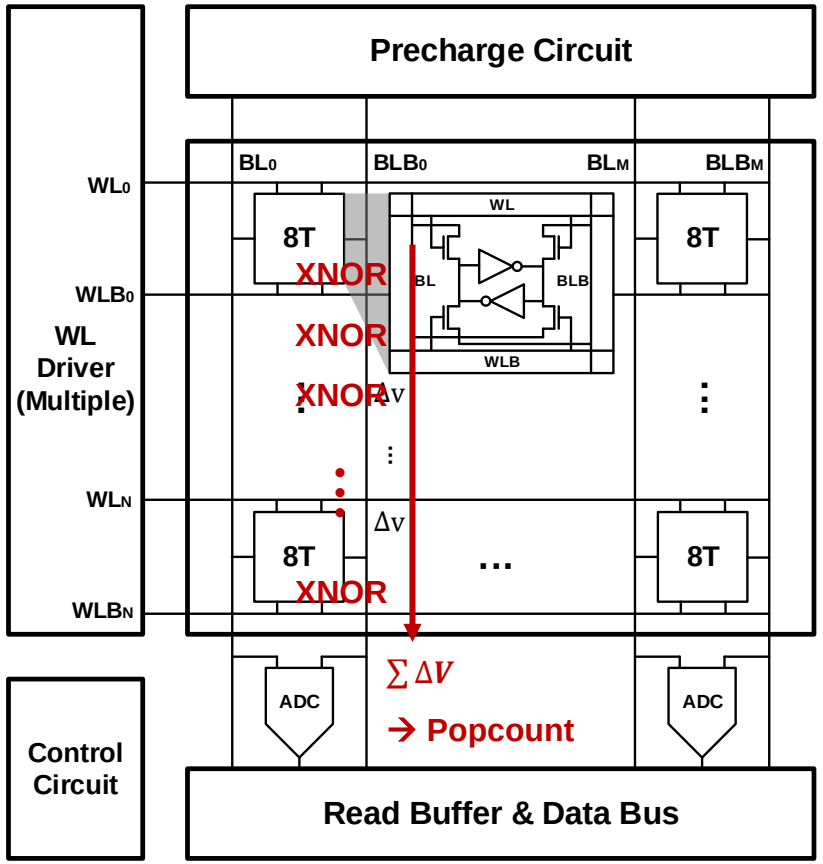
In-Memory Processing Architecture

→ Various in-memory processing designs for XNOR-Net in SRAM & DRAM are proposed.

In-SRAM Processing for XNOR-Net



- ✓ 6T → Custom 8T
- ✓ SA → ADC
- ✓ Multiple row act.

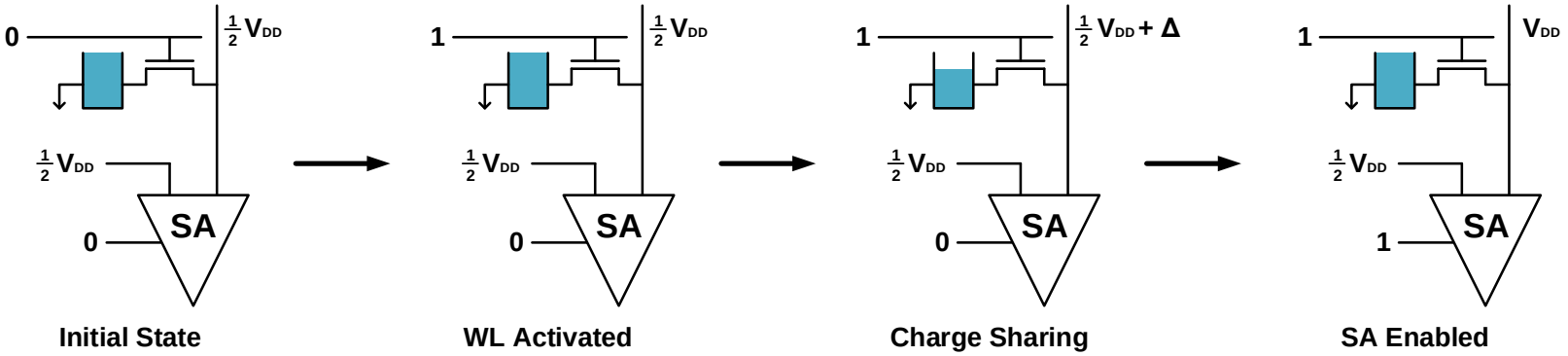


Conventional 6T SRAM

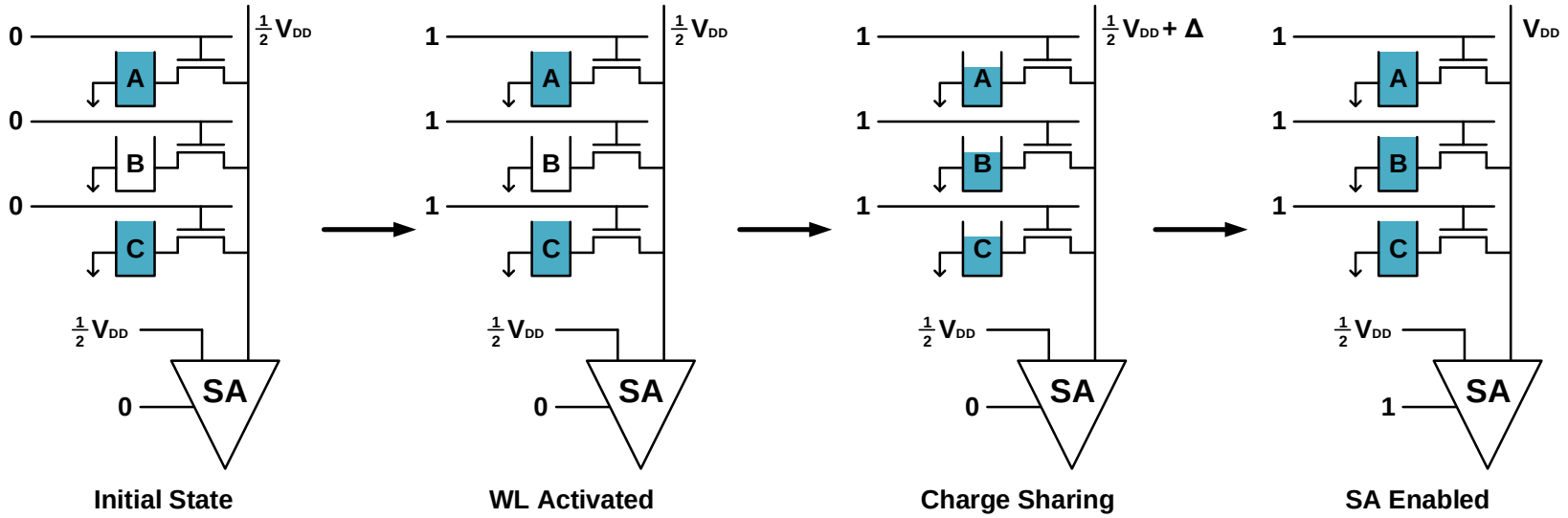
Liu et al. (DAC'18)

→ **Problem**: Lower bit cell density, high energy consumption, …

In-DRAM Processing for XNOR-Net



A read operation (Cell data = 1)



Bitwise OR operation - A OR B (A = 1, B = 0)

- **Conventional 1T1C DRAM**
  - ✓ A read operation occurs to a single target row.
- ***Seshadri et al. (MICRO'17)***
  - ✓ AND, OR: Triple row activation
  - ✓ NOT: Custom cell
  - ✓ XNOR, Popcount: Serially cascading AND, OR, and NOT

→ **Problem: Performance degradation due to serial cascading manner, …**

Final Objective

“ Optimize XNOR-Net to minimize the performance degradation due to in-memory processing.”

Contents

Overview

- Final Objective -
"Optimize XNOR-Net to minimize the performance degradation due to in-memory processing."

Key Feature 1

Conv Decomposition

- No bit cell modifications in SRAM
- No serial cascading manner for XNOR in DRAM

Key Feature 2

Popcount Compression

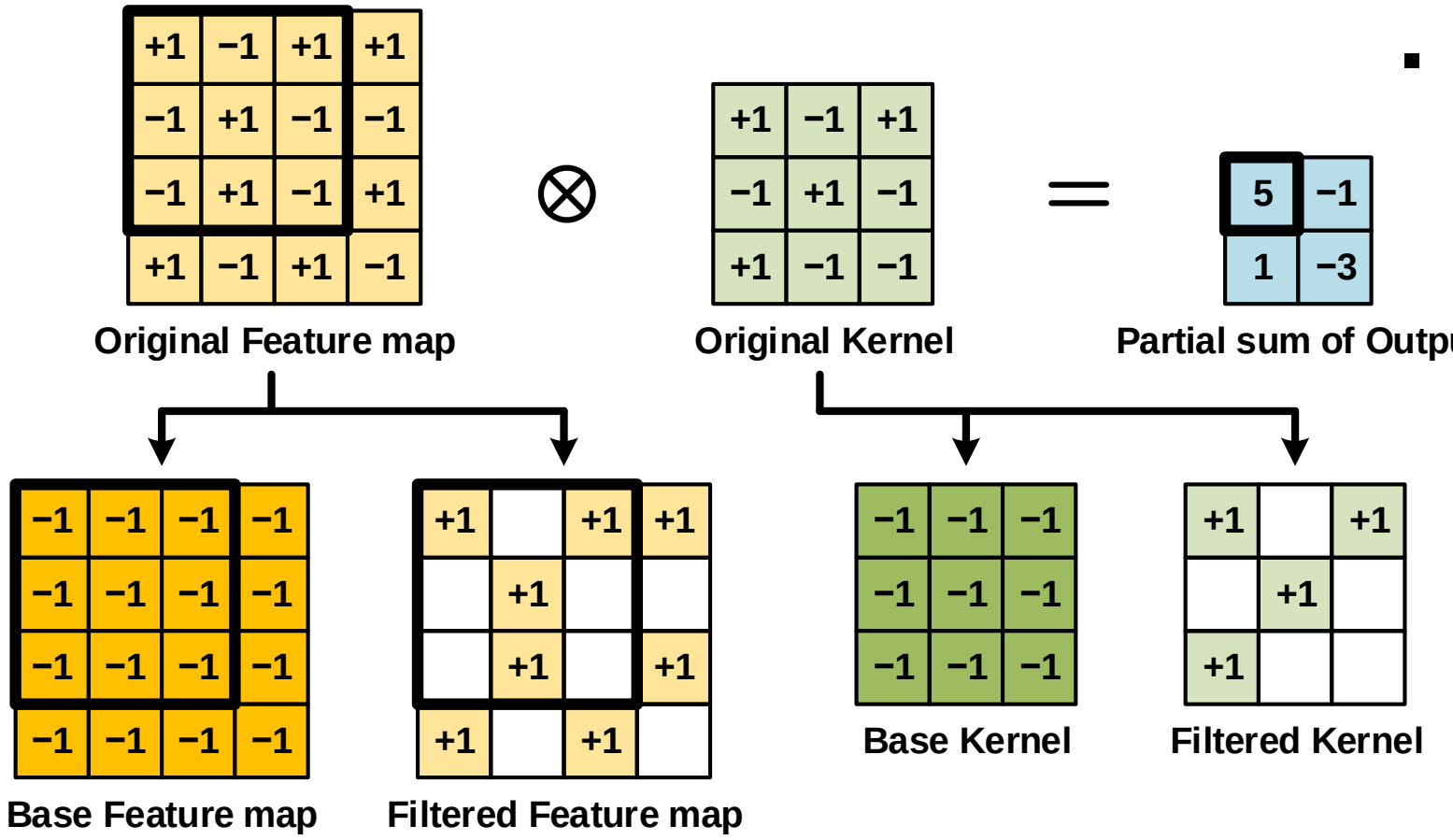
- Reduced size of ADCs in SRAM
- Reduced number of cascading stages in DRAM

"Two accelerators for SRAM and DRAM are implemented and verified by evaluations with the prior state-of-the-arts."

NAND-Net: Optimized binary neural network architecture for in-memory processing in various types of memory

Key Feature 1
Conv Decomposition

Decomposition Method



- **Conv Decomposition**
  - ✓ **Base**: Replace +1 elements with -1s.
  - ✓ **Filtered**: Eliminate all -1 elements.

$$\mathbf{I}_o = \mathbf{I}_b + 2 \cdot \mathbf{I}_f$$

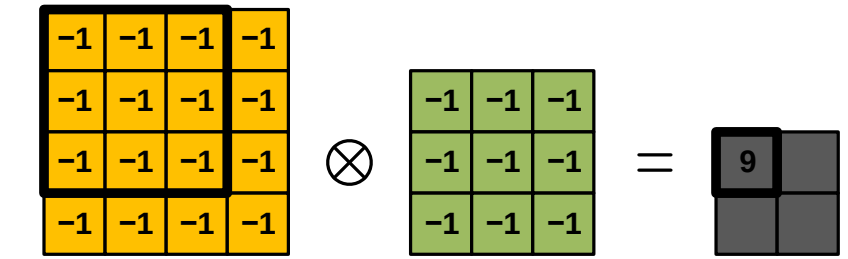
$$\mathbf{W}_o = \mathbf{W}_b + 2 \cdot \mathbf{W}_f$$

$\mathbf{I}$: Feature map data
$\mathbf{W}$: Kernel data
o : 'original'
b : 'base'
f : 'filtered'

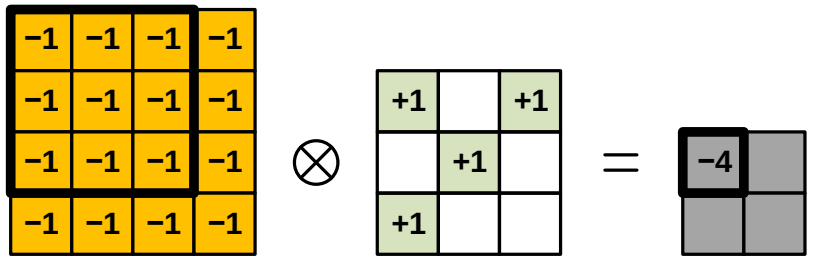
$$\mathbf{I}_o * \mathbf{W}_o = (\mathbf{I}_b + 2 \cdot \mathbf{I}_f) * (\mathbf{W}_b + 2 \cdot \mathbf{W}_f)$$
$$= \mathbf{I}_b * \mathbf{W}_b + 2 \cdot (\mathbf{I}_b * \mathbf{W}_f) + 2 \cdot (\mathbf{I}_f * \mathbf{W}_b) + 4 \cdot (\mathbf{I}_f * \mathbf{W}_f)$$

Decomposed Sub-Convolutions

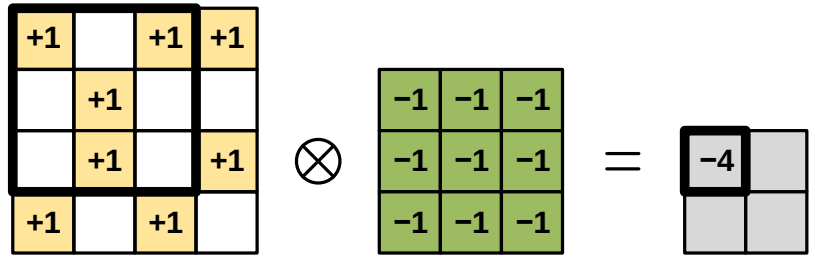
$$\mathbf{I}_o * \mathbf{W}_o = (\mathbf{I}_b + 2 \cdot \mathbf{I}_f) * (\mathbf{W}_b + 2 \cdot \mathbf{W}_f)$$
$$= \mathbf{I}_b * \mathbf{W}_b + 2 \cdot (\mathbf{I}_b * \mathbf{W}_f) + 2 \cdot (\mathbf{I}_f * \mathbf{W}_b) + 4 \cdot (\mathbf{I}_f * \mathbf{W}_f)$$



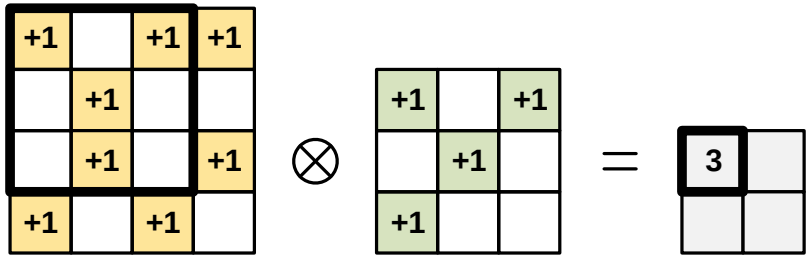
Base Feature map ⊗ Base Kernel = BF−BK



Base Feature map ⊗ Filtered Kernel = BF−FK

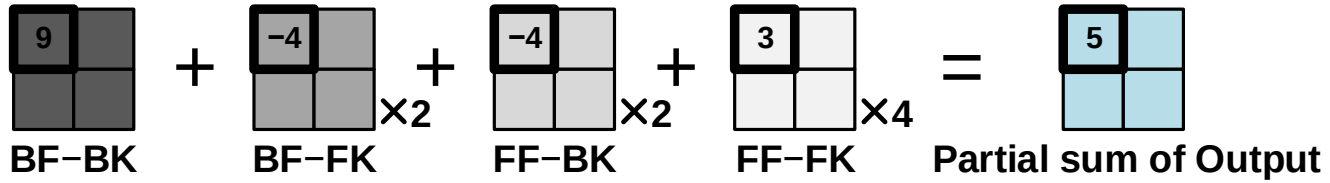


Filtered Feature map ⊗ Base Kernel = FF−BK



Filtered Feature map ⊗ Filtered Kernel = FF−FK

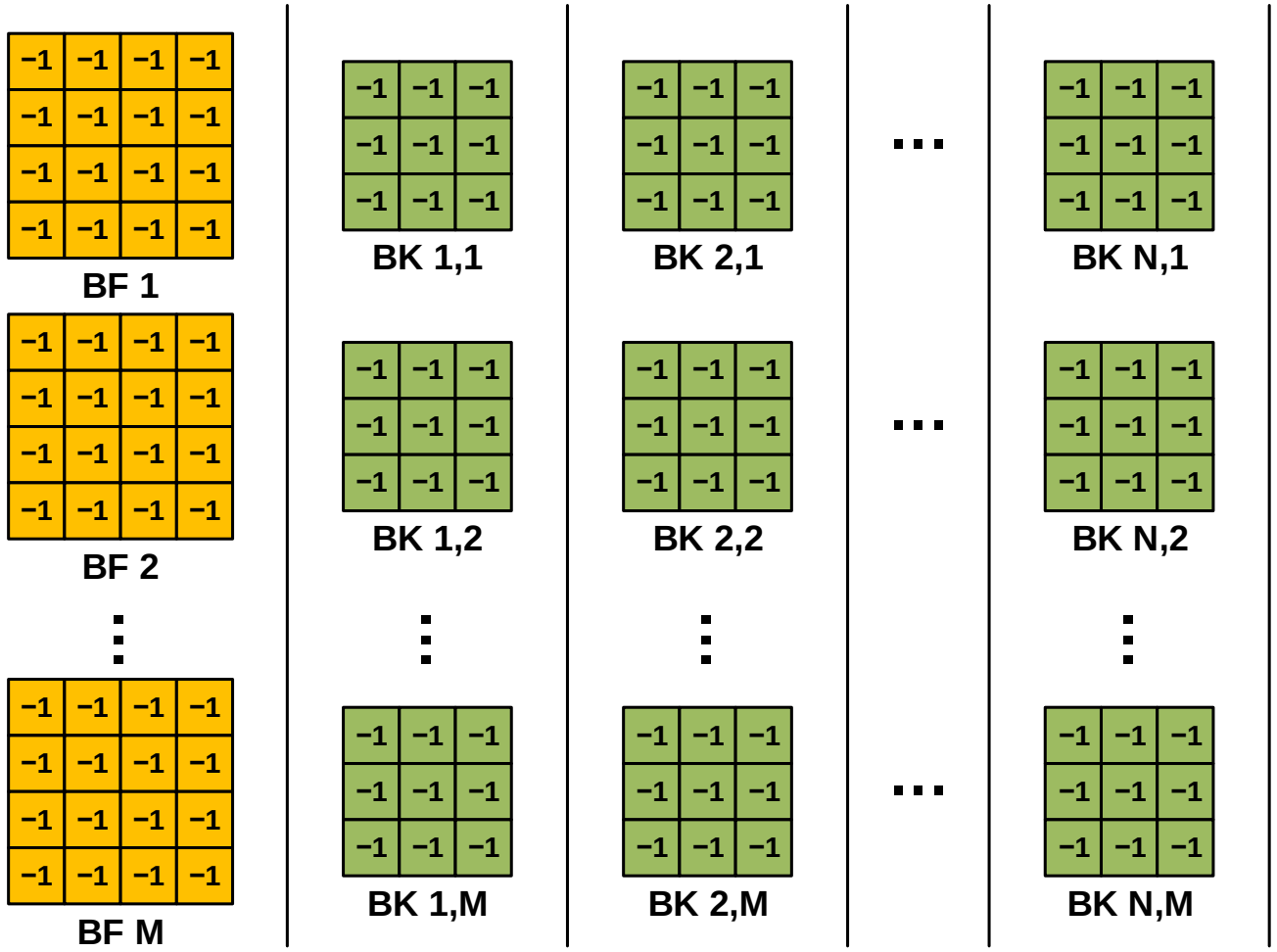
Reconstruction of the original output →



BF−BK + BF−FK ×2 + FF−BK ×2 + FF−FK ×4 = Partial sum of Output

×2 : Left-shift by 1-bit
×4 : Left-shift by 2-bit

Base Feature map - Base Kernel (BF-BK)

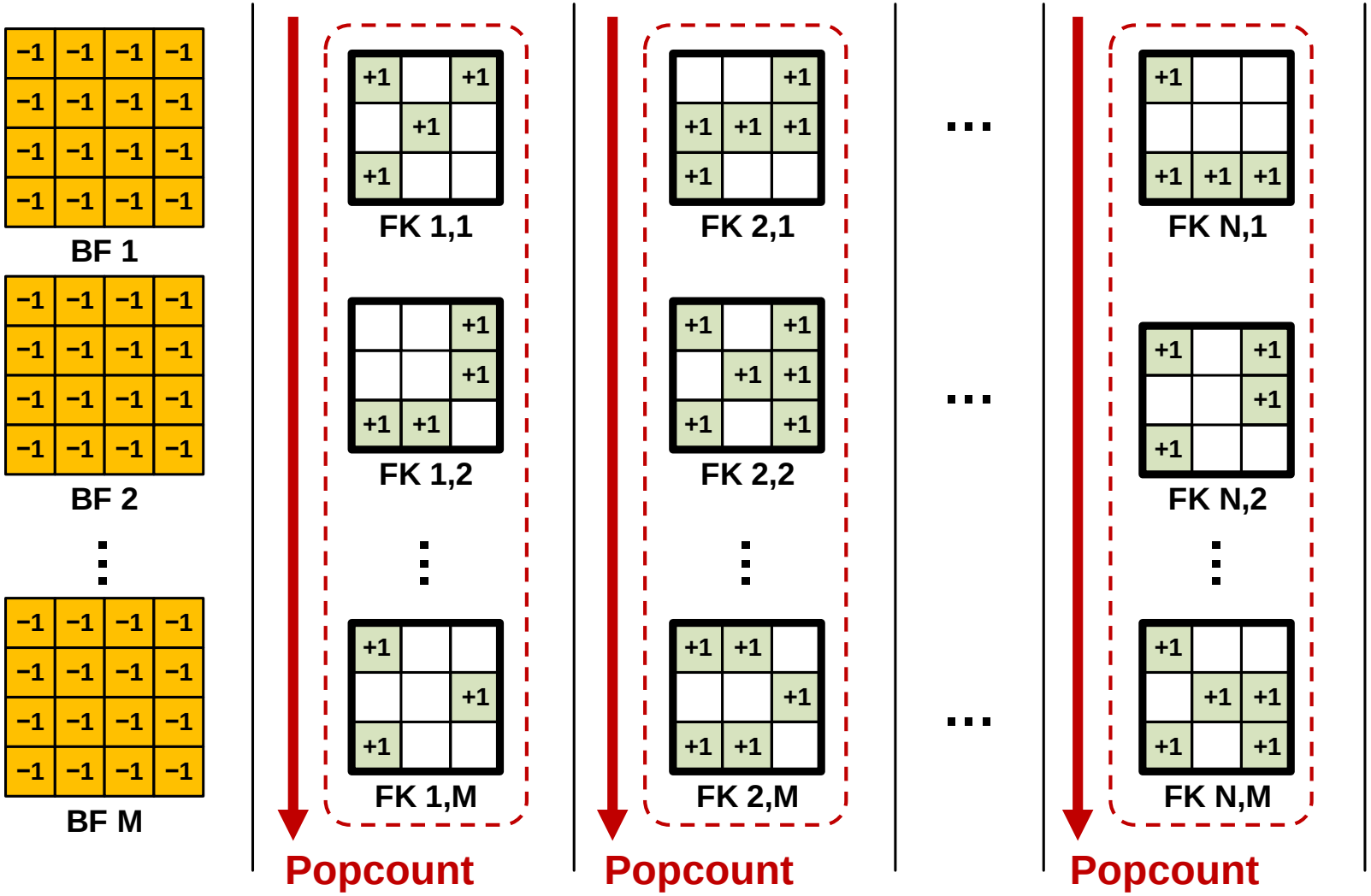


- **BF-BK Convolution**
  - ✓ The computation is deterministic.
  - ✓ All the output values are identical to $\boldsymbol{k} \times \boldsymbol{k} \times \boldsymbol{M}$
  - ✓ Can be precomputed offline and merged to bias elements.

$\boldsymbol{M}$: Channel of input feature map
$\boldsymbol{N}$: Channel of output feature map
$\boldsymbol{k}$: Size of kernel

→ **No runtime overhead.**

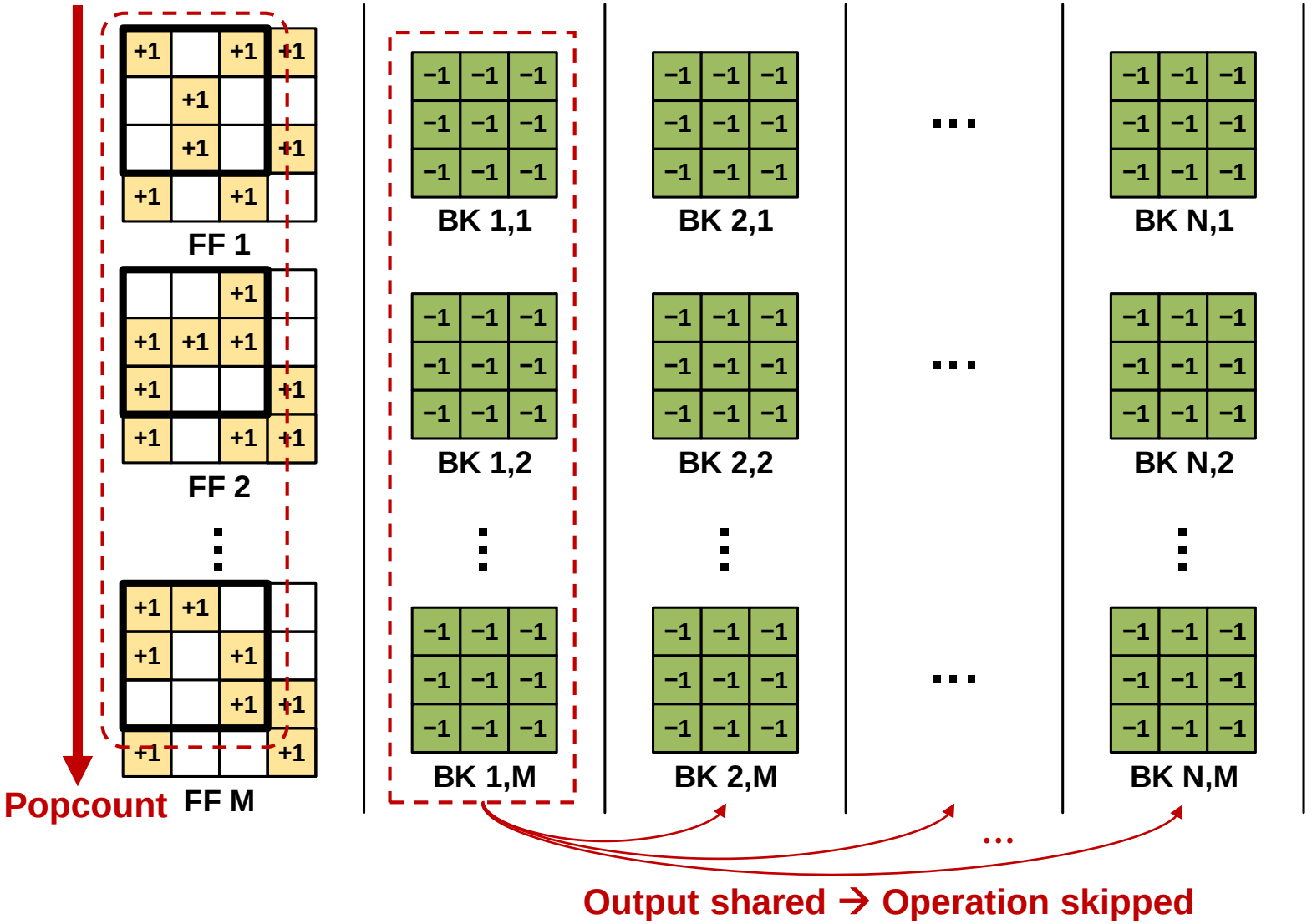
Base Feature map - Filtered Kernel (BF-FK)



- **BF-FK Convolution**
  - ✓ Popcount over all kernels in the same output channel.
  - ✓ Can be precomputed offline and merged to bias elements.

→ **No runtime overhead.**

Filtered Feature map - Base Kernel (FF-BK)

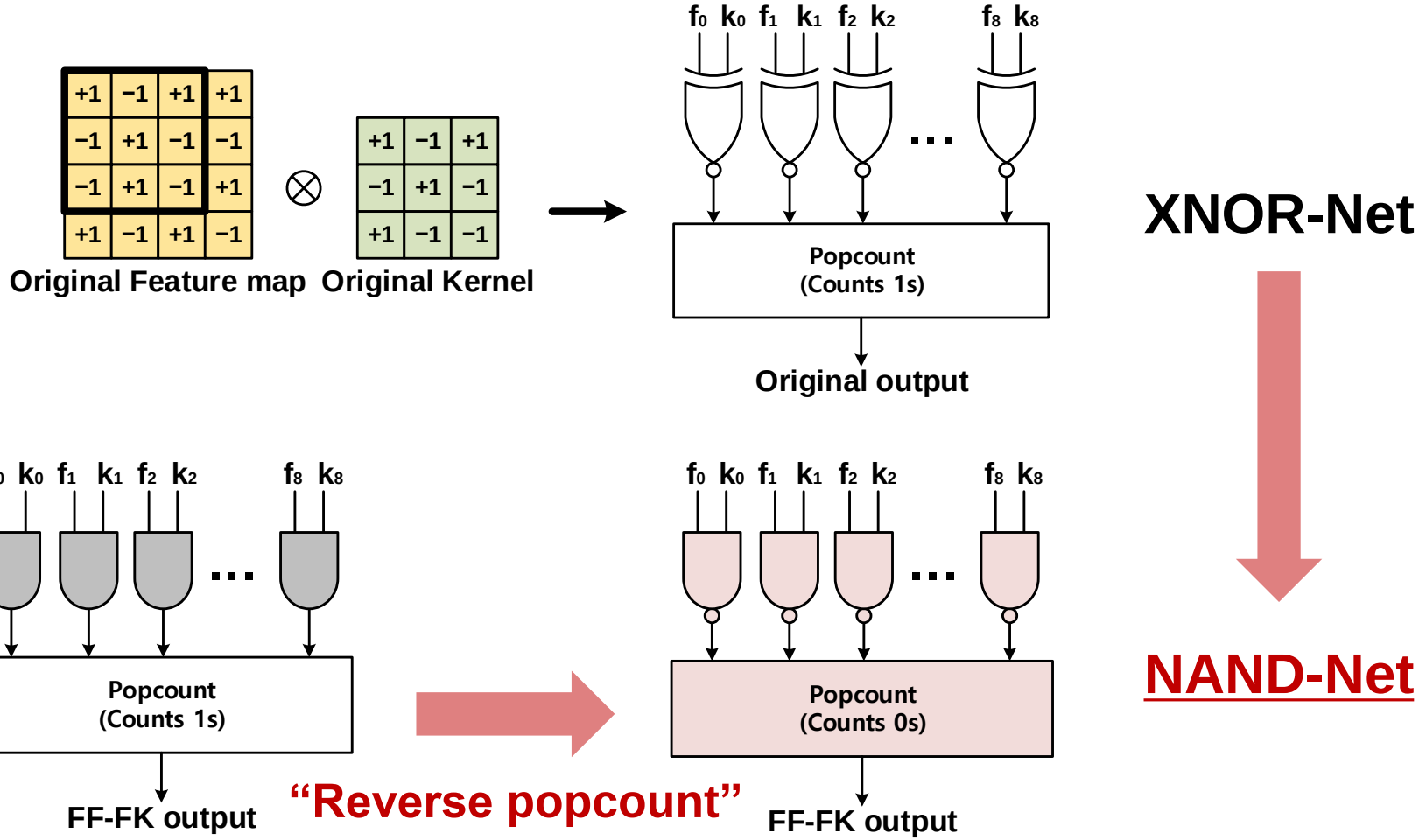


- **FF-BK Convolution**
  - ✓ Popcount over all kernel windows in the same position of the input feature maps.
  - ✓ Except the first output channel, the remaining channels can be skipped.

→ Causes runtime overhead.

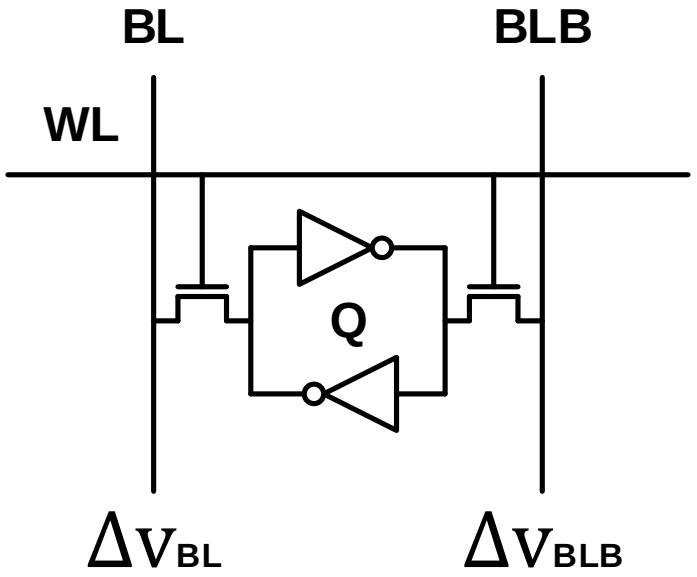
Filtered Feature map - Filtered Kernel (FF-FK)

- **FF-FK Convolution**
  - ✓ The main part among the four sub-convolutions
  - ✓ No pre-computing
  - ✓ No operation skipping

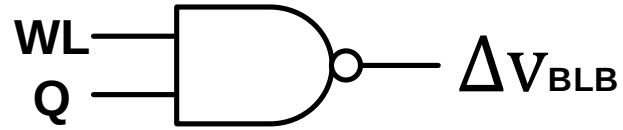


· f_i : feature map element
· k_i : kernel element

Bitwise NAND using Conventional 6T Cell in SRAM



WL	Q	ΔV_{BL}	ΔV_{BLB}
0	0	0	0
0	1	0	0
1	0	Δ	0
1	1	0	Δ

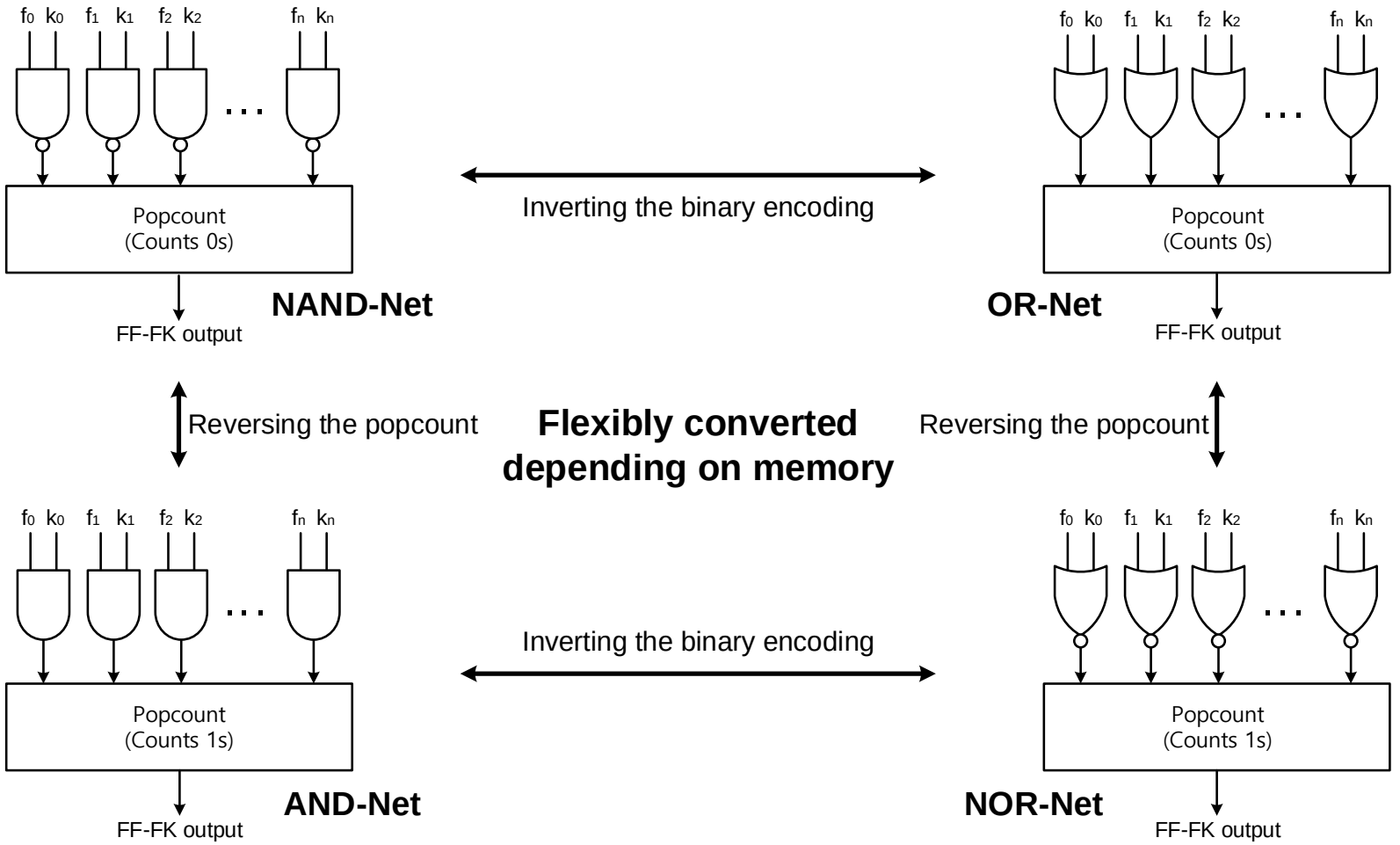


0: No voltage discharge (High)
Δ: Voltage discharge (Low)

- ✓ Bitwise XNOR requires bit cell modifications. (e.g., custom 8T cell in *Liu et al. (DAC'18)*)
- ✓ Bitwise NAND can be performed by the 6T cell without any modifications.

→ **NAND-Net effectively solves the performance degradation issue of SRAM.**

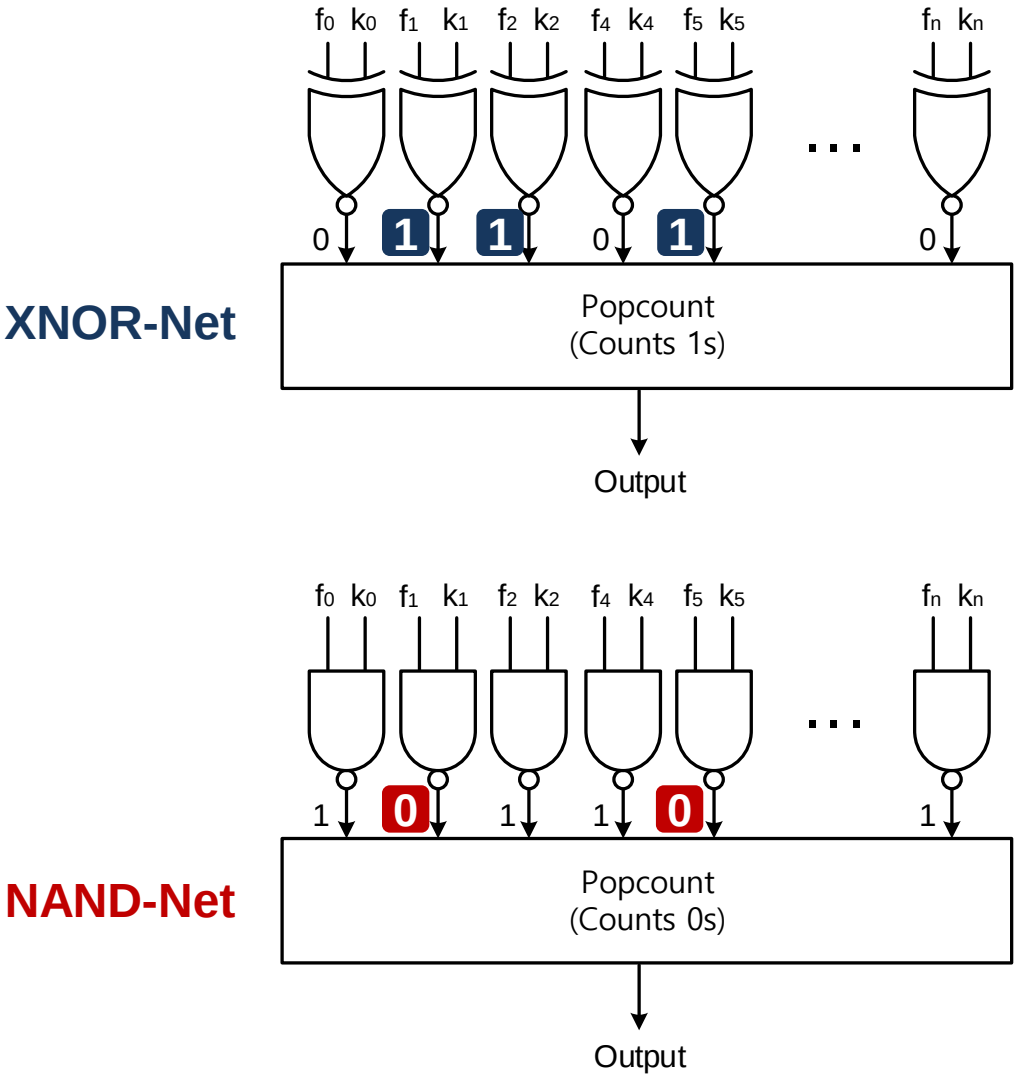
Flexibility of NAND-Net



→ NAND-Net can be flexibly switched to the other forms depending on memory characteristics.

Key Feature 2
Popcount Compression

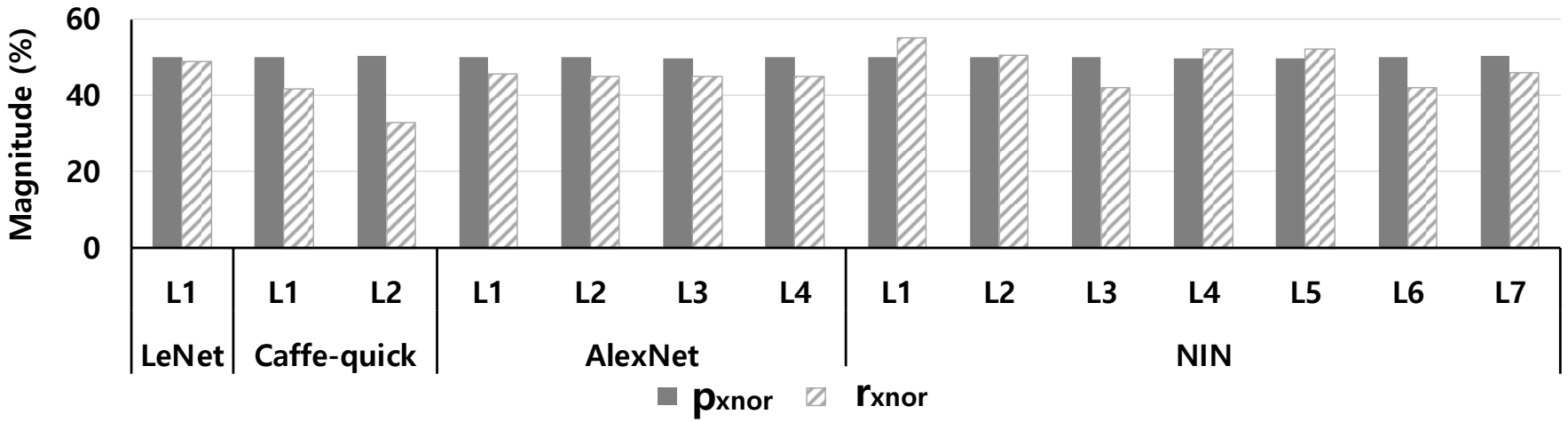
Target Bit of Popcount

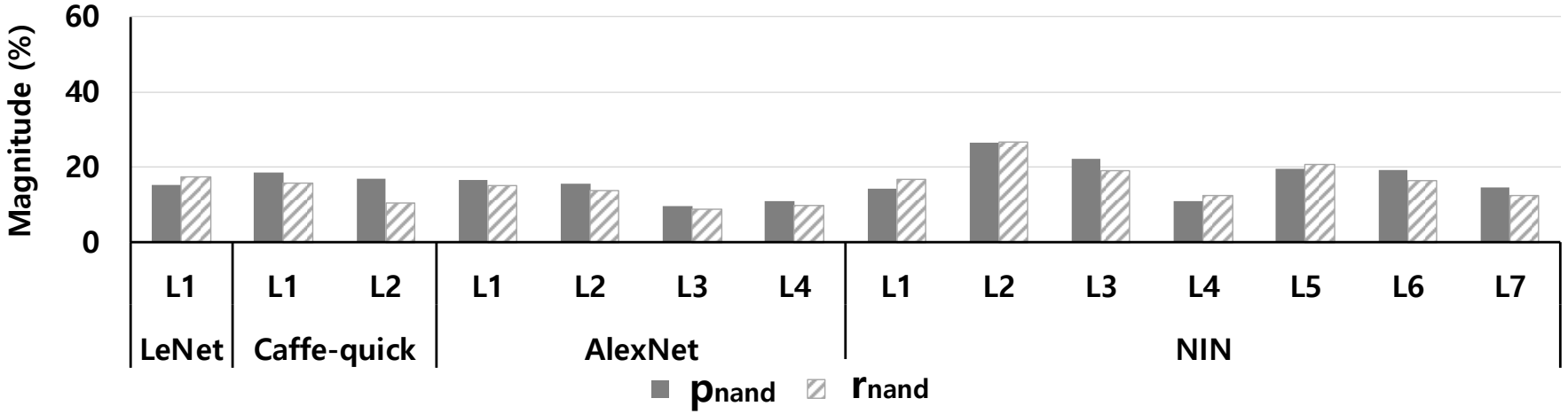


- **Target Bit of Popcount Operations**
  - ✓ **Target bit**: The symbol to be counted by the popcount. (e.g., '1' or '0')
  - ✓ Probabilistically, NAND-Net produces less target bits than XNOR-Net.

f	k	XNOR	NAND
0	0	1	1
0	1	0	1
1	0	0	1
1	1	1	0

Target Bit Statistics





p_{XNOR}: Target bit probability of XNOR-Net

r_{XNOR}: Measured proportion of XNOR-Net

p_{NAND}: Target bit probability of NAND-Net

r_{NAND}: Measured proportion of NAND-Net

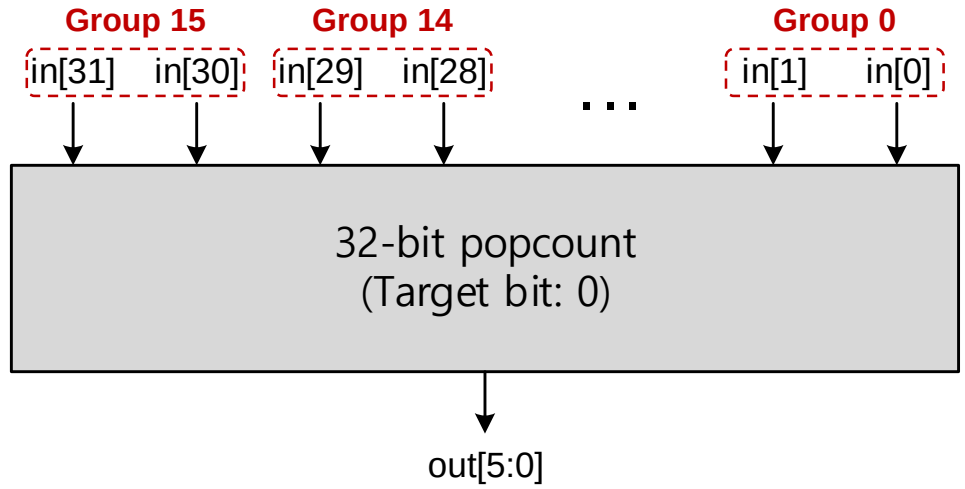
XNOR-Net → 42% ~ 55.2%

66.9% reduction on average!

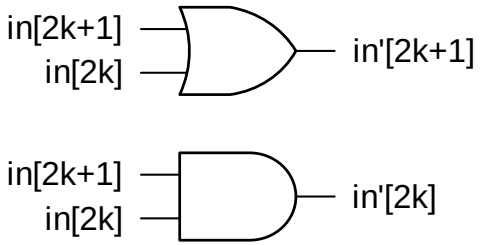
NAND-Net → 8.69% ~ 26.6%

Popcount Compression in DRAM

- **Applying popcount compression in DRAM**
  - ✓ **Baseline**: m-bit popcount (e.g., 32-bit popcount)
  - ① Divide the m bit input sequence into groups of n bits ($m > n$) (e.g., groups of 2 bits)
  - ② Add sorting logic which gathers every target bit in each group to the right side of that group.
  - ③ Discard the leftmost bits from all the groups.



in[2k+1]	in[2k]	in'[2k+1]	in'[2k]
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

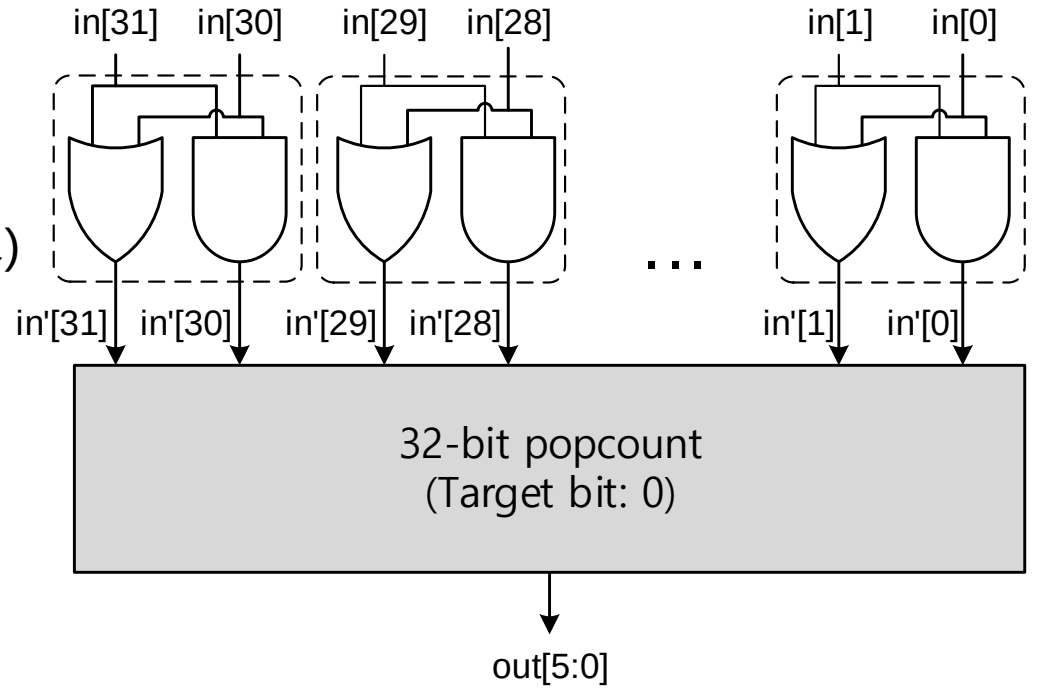


Sorting logic for each group

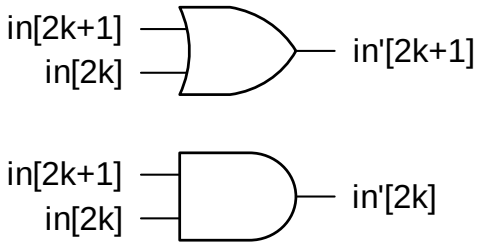
→ The popcount size is reduced by half at the cost of 16 AND operations!

Popcount Compression in DRAM

- **Applying popcount compression in DRAM**
  - ✓ **Baseline**: m-bit popcount (e.g., 32-bit popcount)
  - ① Divide the m bit input sequence into groups of n bits ($m > n$) (e.g., groups of 2 bits)
  - ② Add sorting logic which gathers every target bit in each group to the right side of that group.
  - ③ Discard the leftmost bits from all the groups.



in[2k+1]	in[2k]	in'[2k+1]	in'[2k]
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

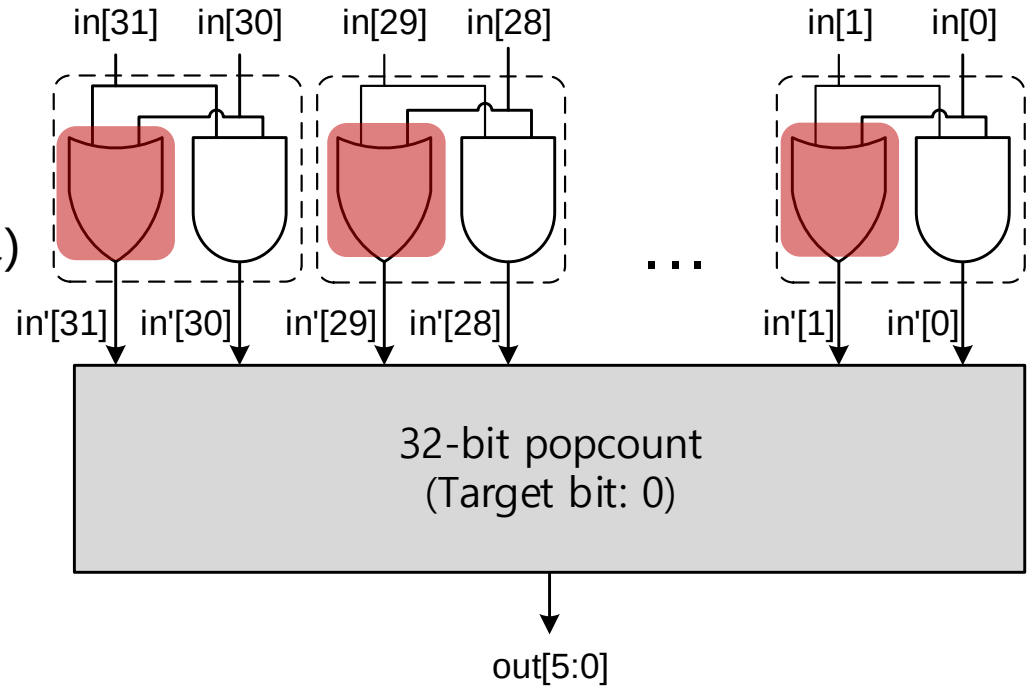


Sorting logic for each group

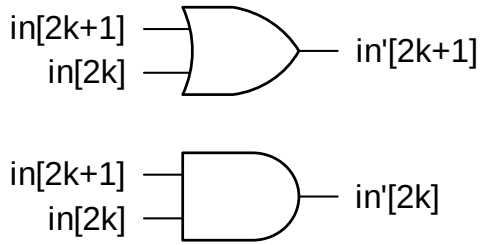
→ The popcount size is reduced by half at the cost of 16 AND operations!

Popcount Compression in DRAM

- **Applying popcount compression in DRAM**
  - ✓ **Baseline**: m-bit popcount (e.g., 32-bit popcount)
  - ① Divide the m bit input sequence into groups of n bits ($m > n$) (e.g., groups of 2 bits)
  - ② Add sorting logic which gathers every target bit in each group to the right side of that group.
  - ③ Discard the leftmost bits from all the groups.



in[2k+1]	in[2k]	in'[2k+1]	in'[2k]
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

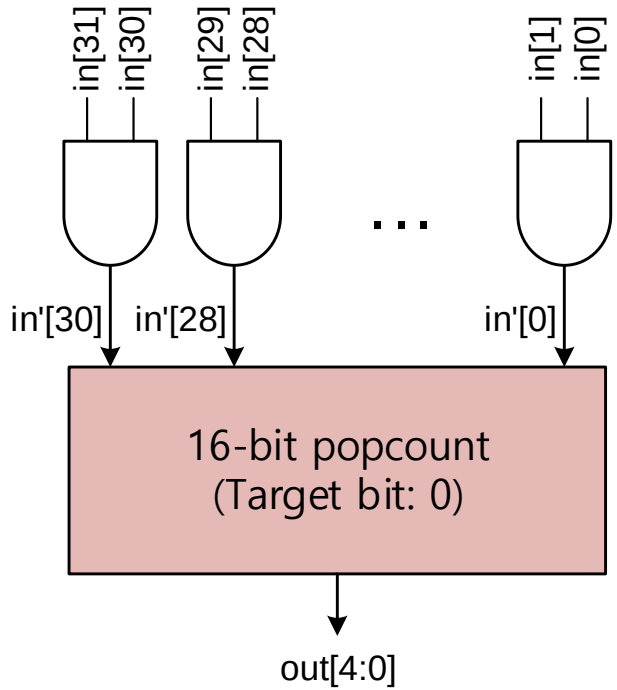


Sorting logic for each group

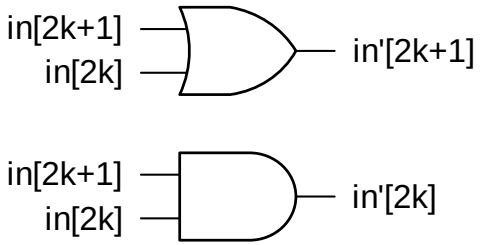
→ The popcount size is reduced by half at the cost of 16 AND operations!

Popcount Compression in DRAM

- **Applying popcount compression in DRAM**
  - ✓ **Baseline**: m-bit popcount (e.g., 32-bit popcount)
  - ① Divide the m bit input sequence into groups of n bits ($m > n$) (e.g., groups of 2 bits)
  - ② Add sorting logic which gathers every target bit in each group to the right side of that group.
  - ③ Discard the leftmost bits from all the groups.



in[2k+1]	in[2k]	in'[2k+1]	in'[2k]
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1



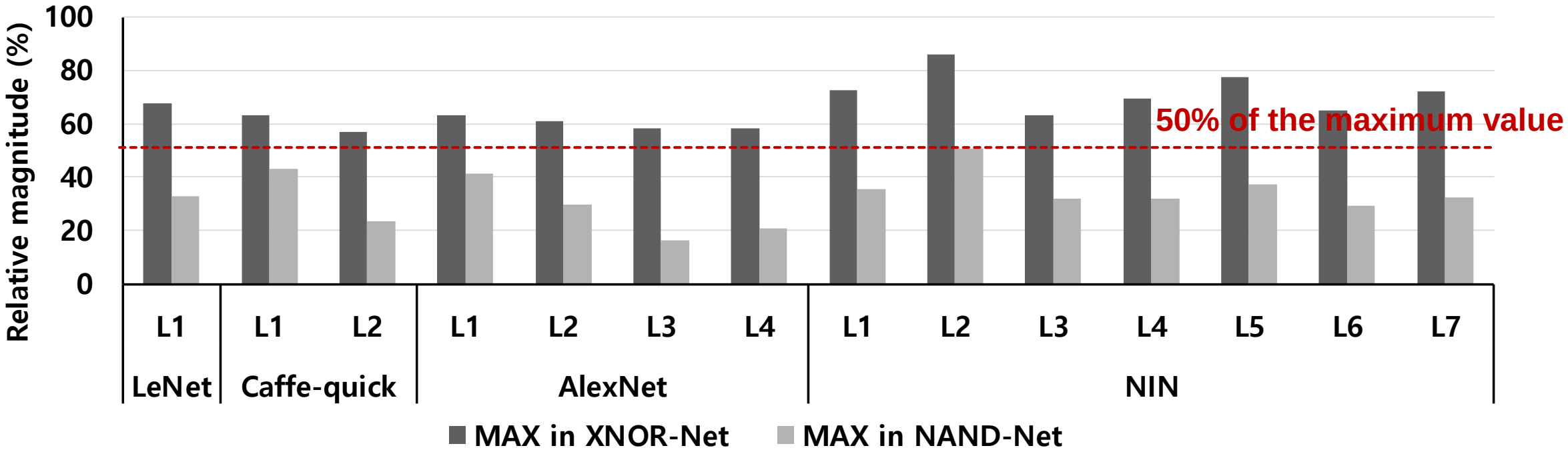
Sorting logic for each group

→ **The popcount size is reduced by half at the cost of 16 AND operations!**

Popcount Compression in SRAM

- **Applying popcount compression in SRAM**
  - ✓ The method used in DRAM cannot be applied to SRAM.
  - ✓ Instead, the resolution of the ADCs can be reduced by 1-bit based on the following observation.

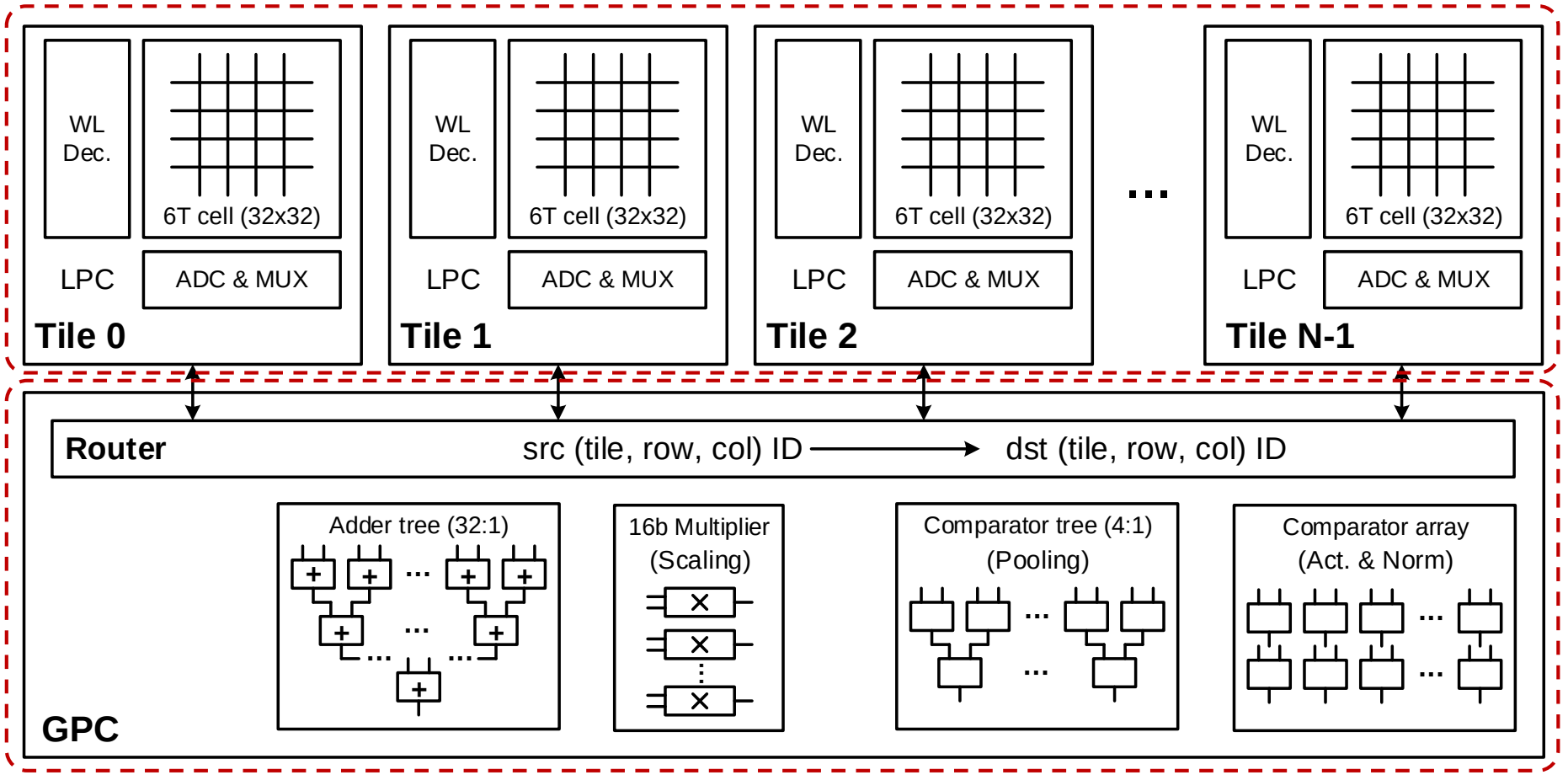
→ **NAND-Net does not utilize the upper half voltage range of the ADCs!**



The relative magnitude of the maximum popcount results

Hardware Design
NAND-Net Accelerators

NAND-Net Accelerator Architecture



- **Tiles (For conv. layer)**
  - → Multiplication (NAND)
  - → Accumulation (Popcount)
- **GPC (For minor op.)**
  - → Scaling
  - → Pooling
  - → Act. & Norm.

- ✓ In-SRAM accelerator consists of tiles and global peripheral circuit (GPC).
- ✓ In-DRAM accelerator is similar to this architecture except for the width size of the memory.

Contents

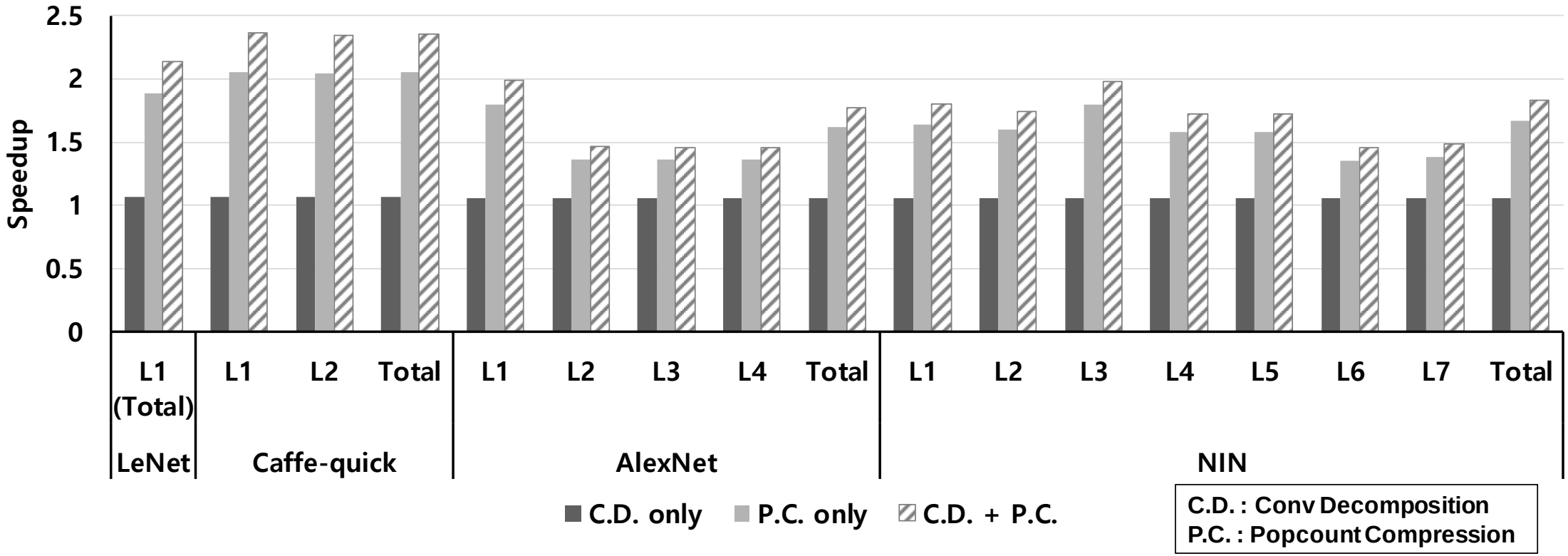
Evaluation Methodology

- **Simulator**
  - ✓ DRAM: In-house simulator based on CACTI
  - ✓ SRAM: NeuroSim (Cell area), Cadence tools (ADC modeling), Synopsys DC (Peripheral area)
- **Benchmarks**
  - ✓ LeNet, Caffe-quick, AlexNet, NIN
- **Comparison Baseline**
  - ✓ DRAM: Ambit (*Seshadri et al. (MICRO'17)*)
  - ✓ SRAM: 8T-based architecture (*Liu et al. (DAC'18)*)
- **Memory Specifications**

DRAM	
Model	4GB DDR4-2400 (JEDEC)
Config. Info.	16 banks, 2KB row buffer,
Timing Param.	tRAS: 32ns, tRP: 14.16ns
Width Info.	Word-length: 64B

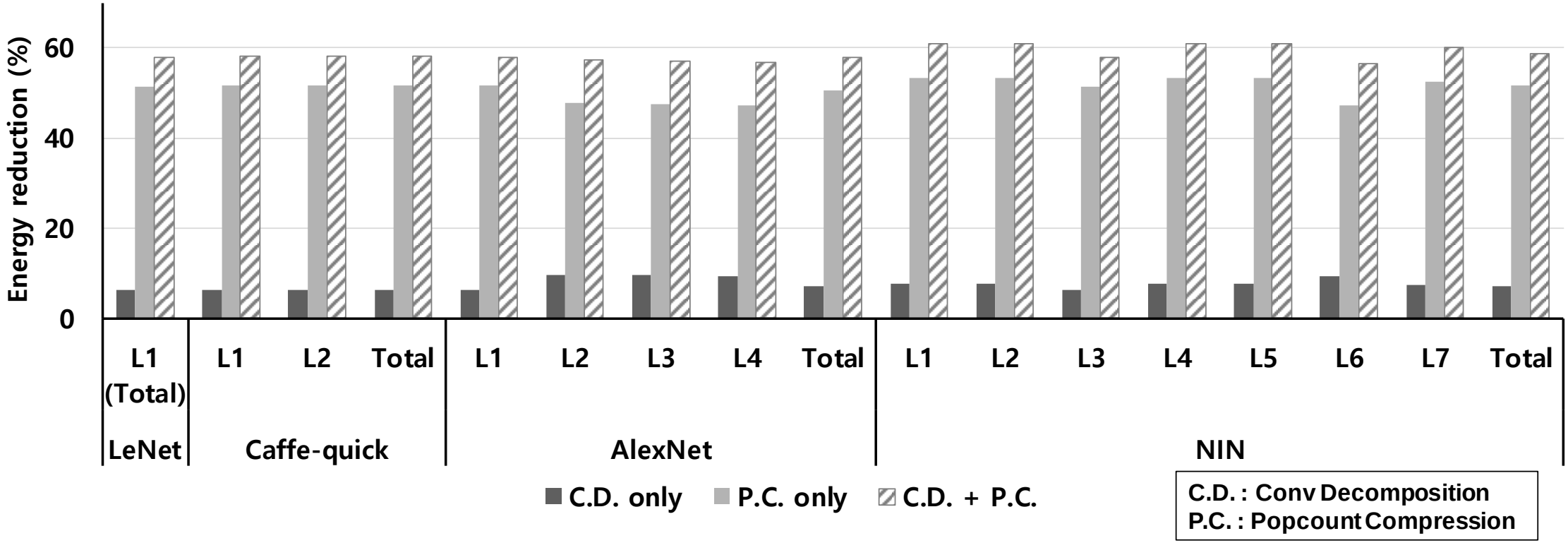
SRAM	
Model	2KB Standard 6T (ITRS) -65nm CMOS
Config. Info.	16 tiles (128B per each), Word: 16B
Cell Sizing (F^2)	Area: 146, AR = 1.46
ADC	4-bit Multi-level-SA

Speedup of NAND-Net in DRAM



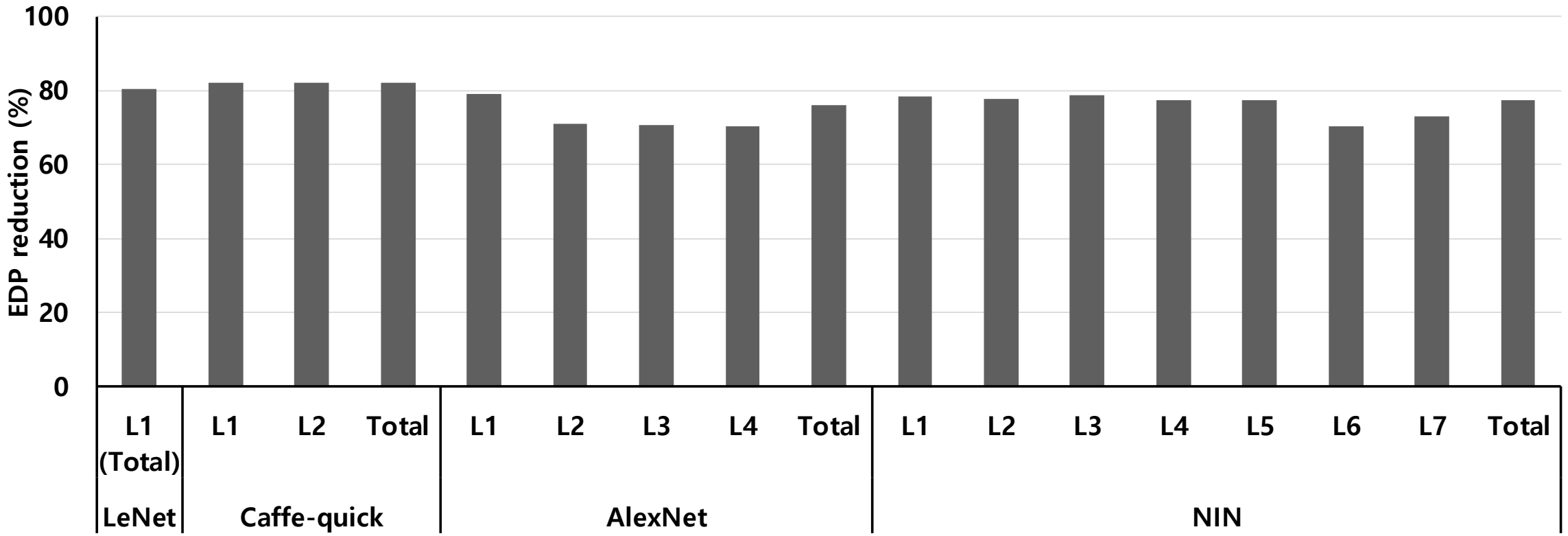
- ✓ The latency bottleneck of in-memory popcounts due to the large number of operation cascading is effectively alleviated by more than a factor of 1.6x.
- ✓ The average speedup is **2.03x** for the target benchmarks

Energy Reduction of NAND-Net in DRAM



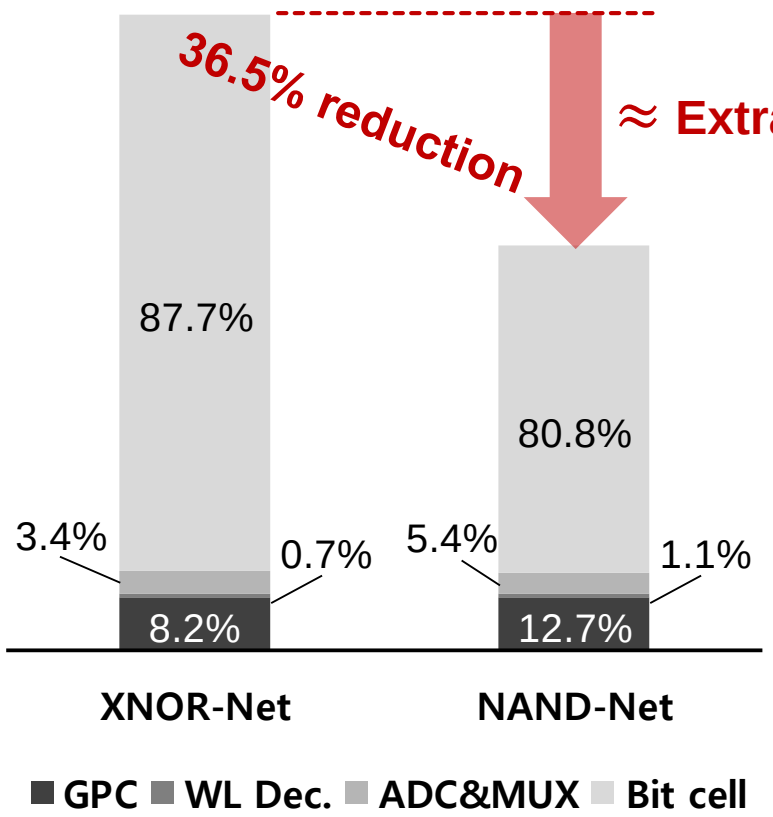
- ✓ The average energy reduction is around **58.15%** for all the target benchmarks.
- ✓ The effect of C.D. is around 5~6%, and P.C. contributes to the remaining portion of the total value.

EDP Reduction of NAND-Net in DRAM

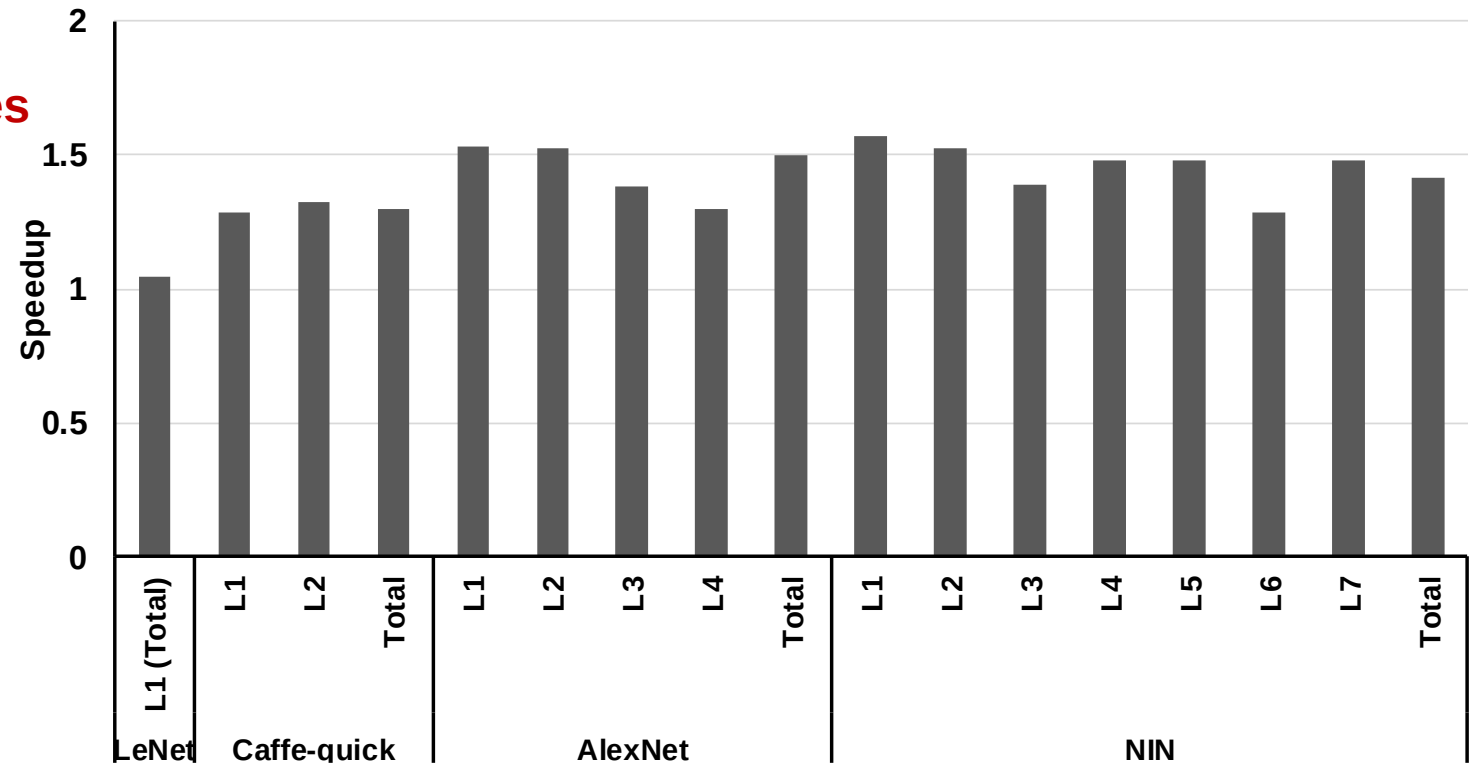


- ✓ Since NAND-Net considerably reduces both the latency and the energy consumption regardless of the benchmark, EDP is also significantly improved.
- ✓ The average EDP reduction is **79.1%** for the target benchmarks.

Speedup of NAND-Net in SRAM



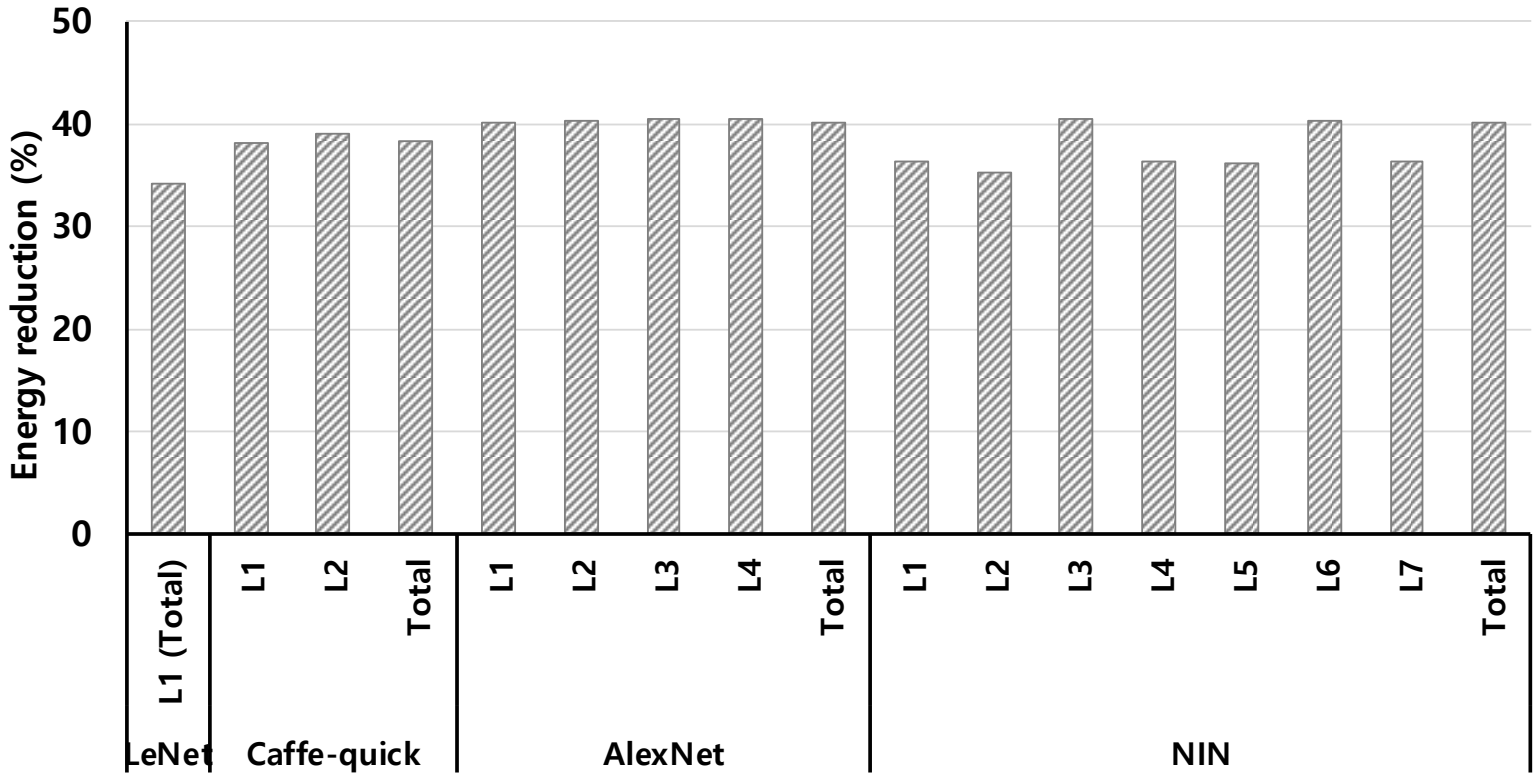
< Area Breakdown>



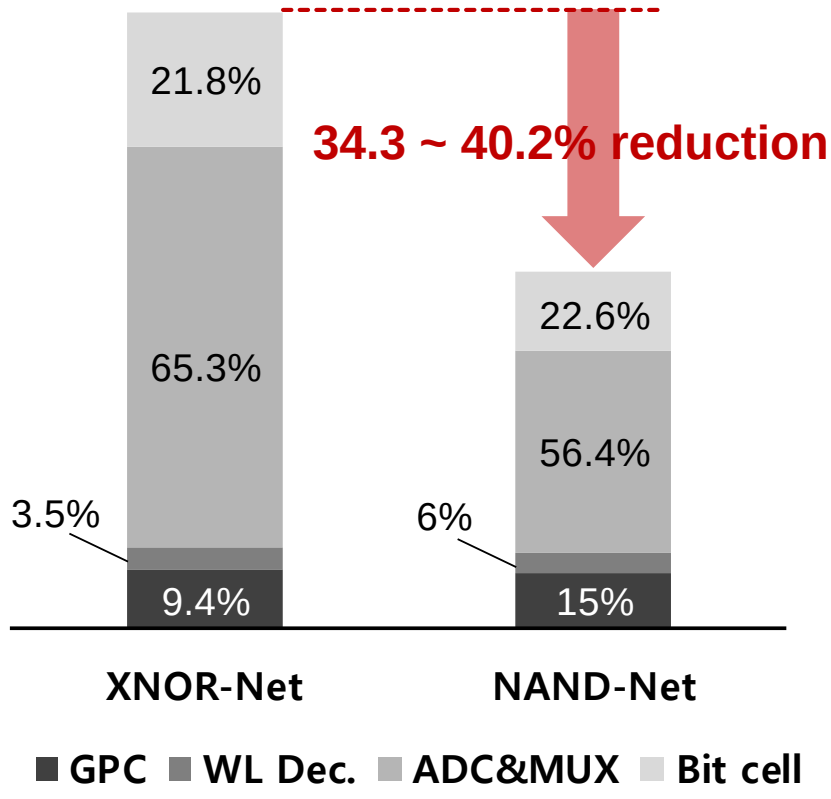
< Speedup with Additional Tiles>

- ✓ 6T cells consume 36.5% less area than 8T cells, and the area reduction can be exploited for speedup.
- ✓ The average speedup is **1.31x** for the target models.

Energy Reduction of NAND-Net in SRAM



< Energy Reduction>



< Energy Breakdown>

- ✓ The energy consumption of bit cell arrays and ADCs are significantly reduced.
- ✓ The average energy reduction is **38.3%** for the target models.

Overhead Analysis

Overhead by FF-BK Convolution

Model	LeNet	Caffe-quick	AlexNet	NIN
Memory footprint (%)	2	2.35	0.33	0.61
Latency (%)	0.67	0.13	0.08	0.22

The Effect of Popcount Compression on The Accuracy

Model	LeNet	Caffe-quick	AlexNet	NIN
Original accuracy (XNOR-Net)	99.23%	73.92%	44.87%	86.28%
Retrained accuracy (NAND-Net)	99.30%	73.50%	43.77%	86.02%
Accuracy drop	-0.07%	0.42%	1.10%	0.26%

→ Given values are negligible with respect to the overall improvements.

Contents

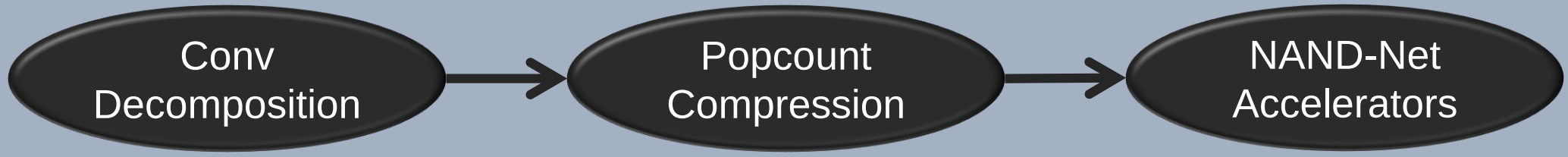
Conclusion

Problem

"In-memory processing for XNOR-Net is emerging as a powerful solution to memory bottleneck issues of deep learning. **However, it causes significant degradation on the memory performance in terms of speed, energy, and area.**"

Solution: Optimize XNOR-Net architecture to adapt to in-memory processing

NAND-Net: Optimized binary neural network architecture for in-memory processing



- Bitwise XNOR → Bitwise NAND
- No bit cell modification in SRAM
- No serial cascading of XNOR in DRAM

- Reduced size of popcount operation
- Reduced resolution of ADC in SRAM
- Reduced number of serial cascading in DRAM

- In-SRAM accelerator for NAND-Net
- In-DRAM accelerator for NAND-Net

➢ NAND-Net achieves 1.04 ~ 2.36x speedup, 34 ~ 59% energy saving, and 41.5% area reduction.

“Thank You For Listening”

NAND-Net: Minimizing Computational Complexity of In-Memory Processing for Binary Neural Networks

Hyeonuk Kim, Jaehyeong Sim, Yeongjae Choi, and Lee-Sup Kim

School of Electrical Engineering

KAIST