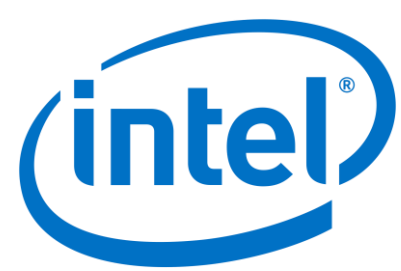
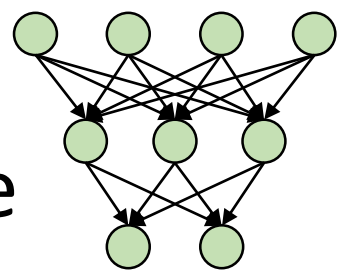
Bit Prudent In-Cache Acceleration of Deep Convolutional Neural Networks

Xiaowei Wang, Jiecao Yu, Charles Augustine,
Ravi Iyer, Reetuparna Das

M-Bits Research Group
UNIVERSITY OF MICHIGAN



Background

Workload	Device	Compute Resource
CNN Inference 	CPU 	Multi-core
	GPU 	CUDA cores
	Neural Cache [ISCA'18] 	Repurposed in-cache ALUs

++ Parallelism

-- Data Movement

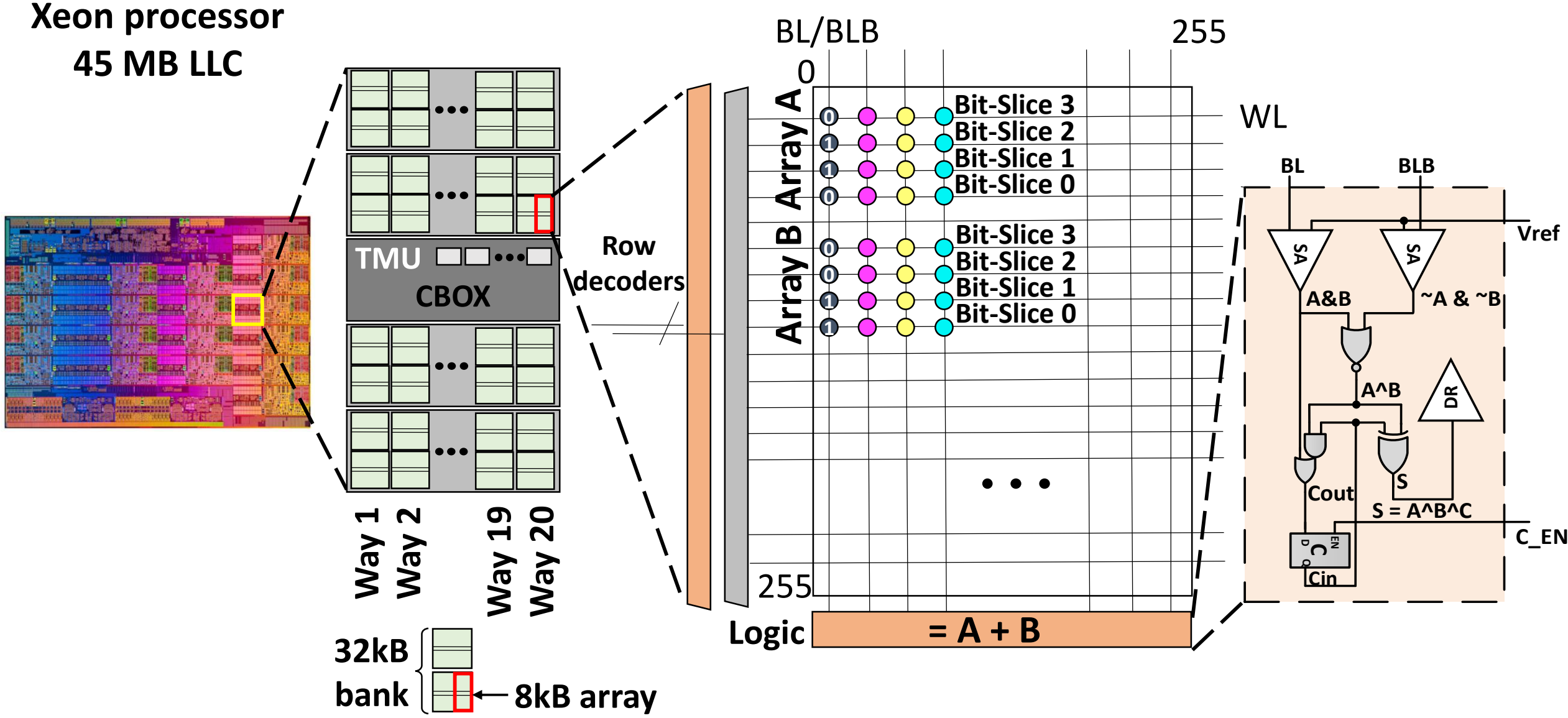
Neural Cache [ISCA'18]

18-core Xeon processor 45 MB LLC

2.5MB LLC slice

8kB SRAM array

Bit-line ALU



18 LLC slices

360 ways

5760 arrays

1,474,560 ALUs

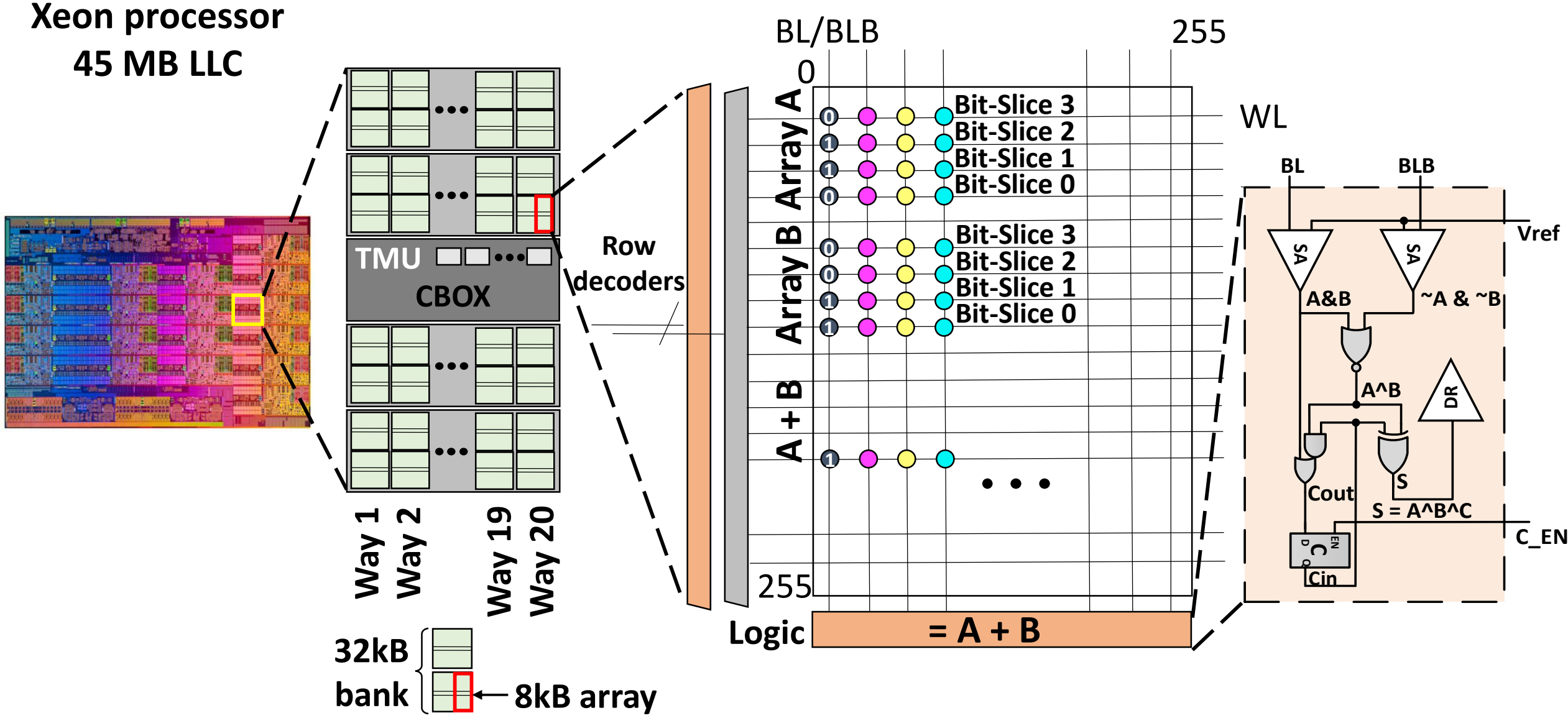
Neural Cache [ISCA'18]

18-core Xeon processor 45 MB LLC

2.5MB LLC slice

8kB SRAM array

Bit-line ALU



18 LLC slices

360 ways

5760 arrays

1,474,560 ALUs

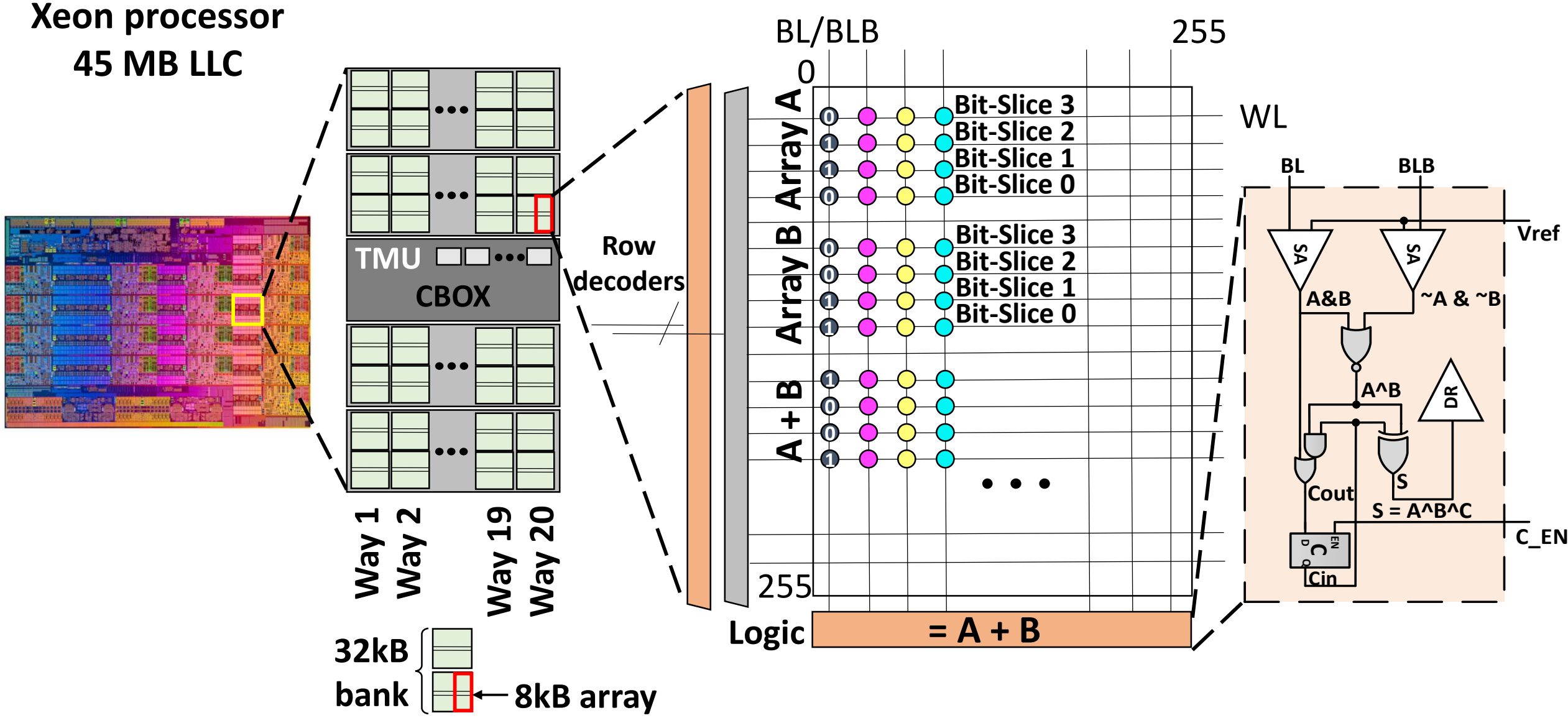
Neural Cache [ISCA'18]

18-core Xeon processor 45 MB LLC

2.5MB LLC slice

8kB SRAM array

Bit-line ALU



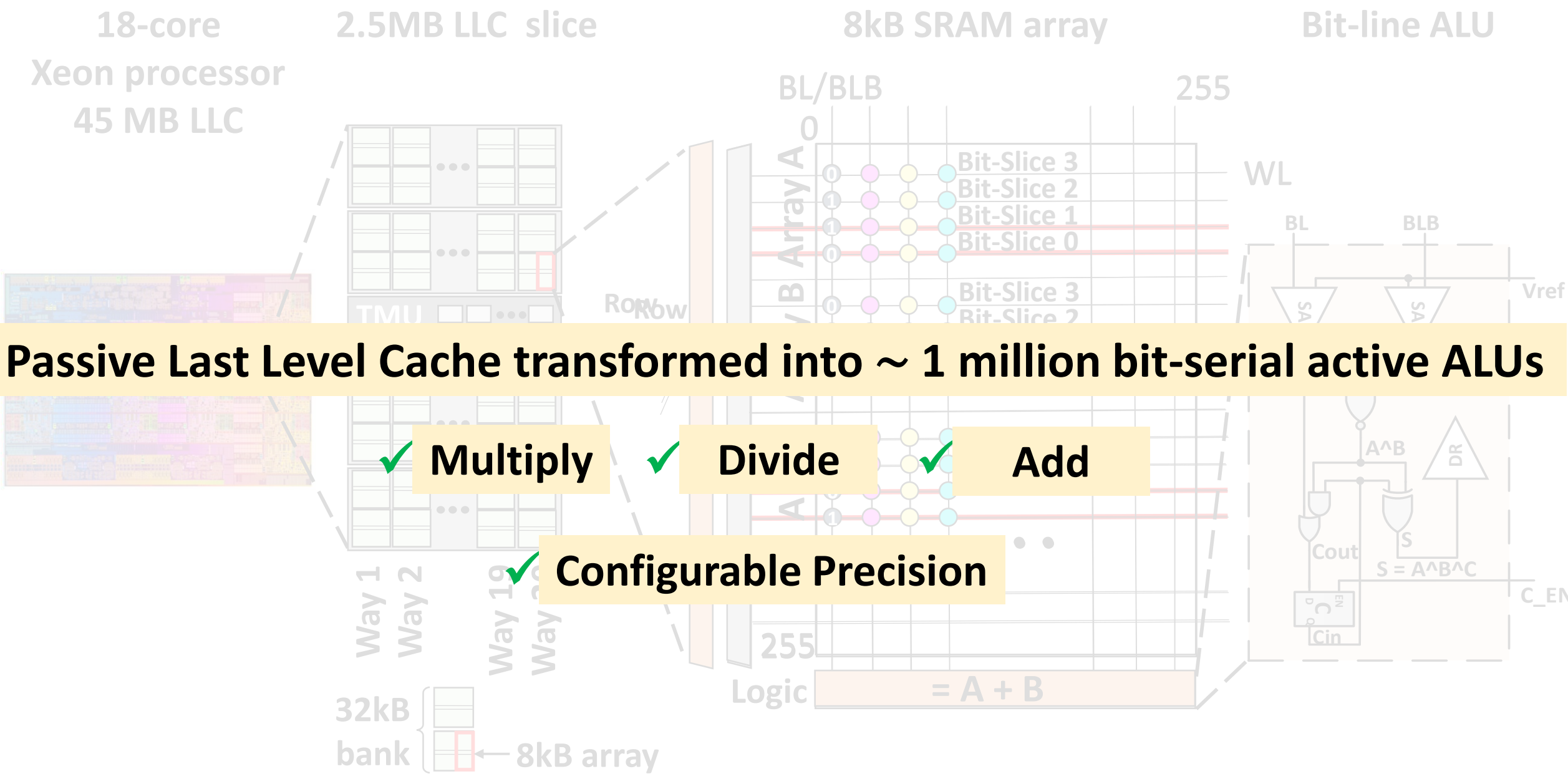
18 LLC slices

360 ways

5760 arrays

1,474,560 ALUs

Neural Cache [ISCA'18]



Passive Last Level Cache transformed into ~ 1 million bit-serial active ALUs

✓ **Multiply** ✓ **Divide** ✓ **Add**

✓ **Configurable Precision**

18 LLC slices **360 ways** **5760 arrays** **1,474,560 ALUs**

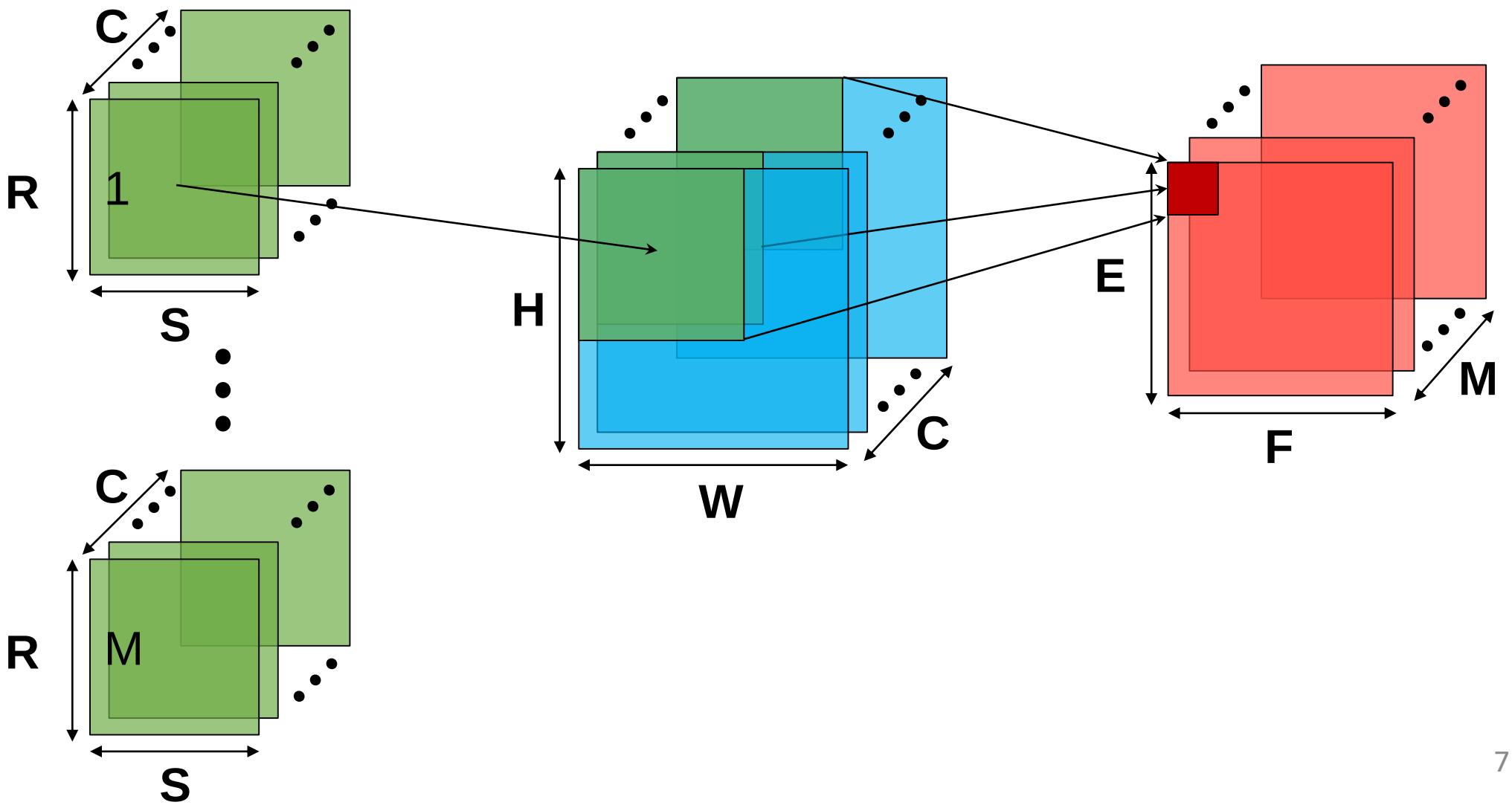
A Convolutional Layer

3D Filters (M)

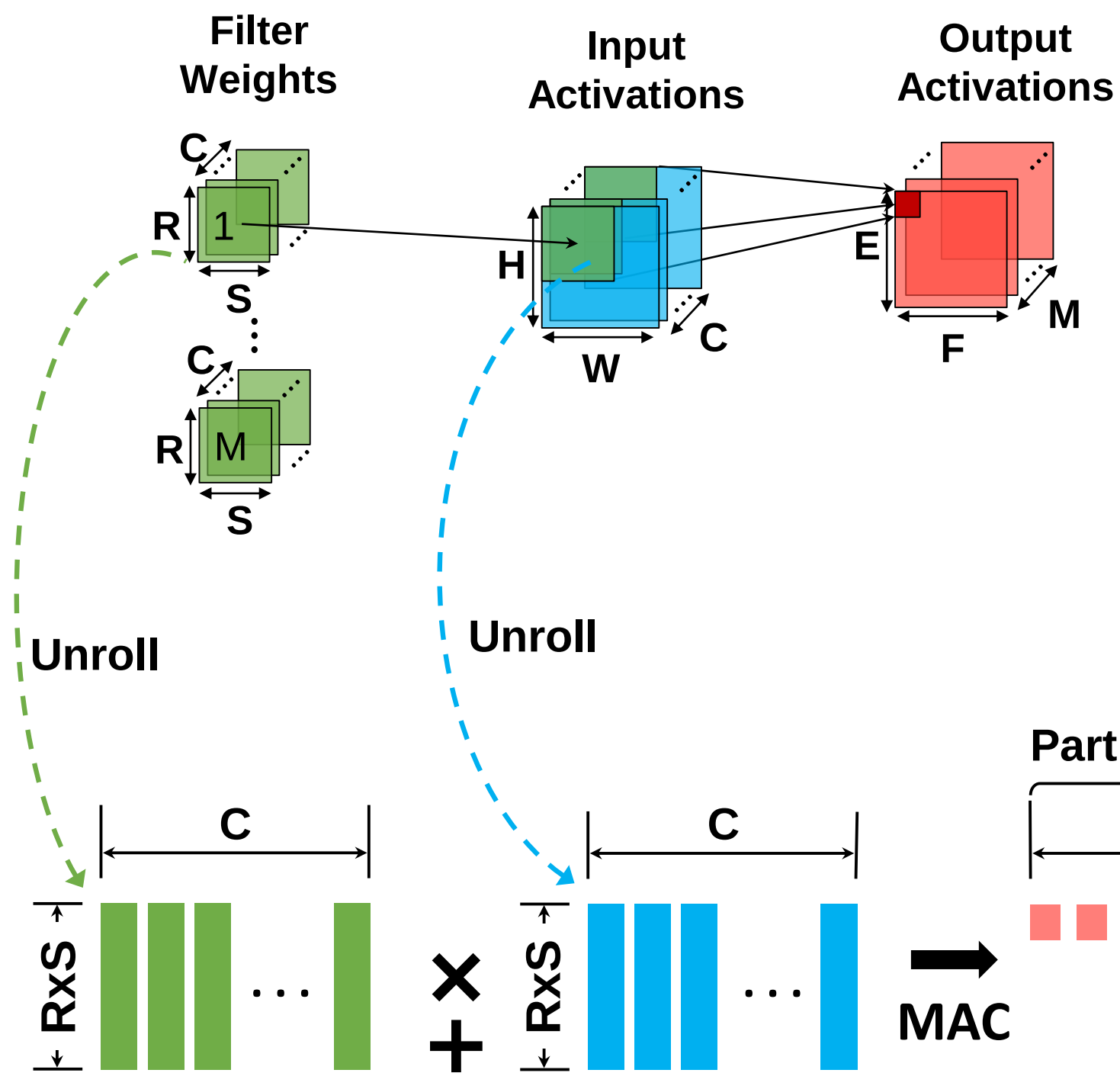
each filter: C channels
each channel: RxS weights

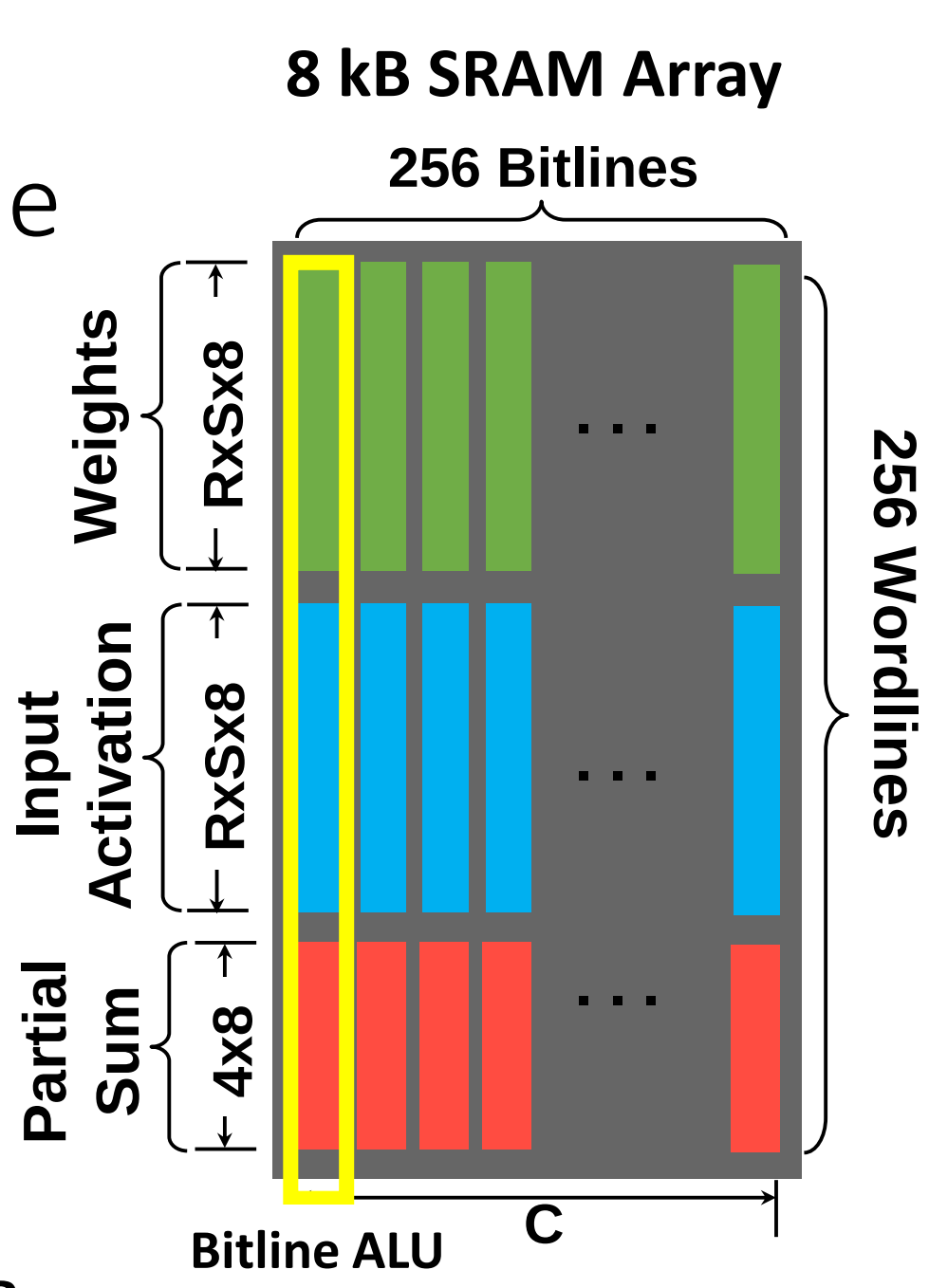
Input Activations (C channels)

Output Activations (M channels)

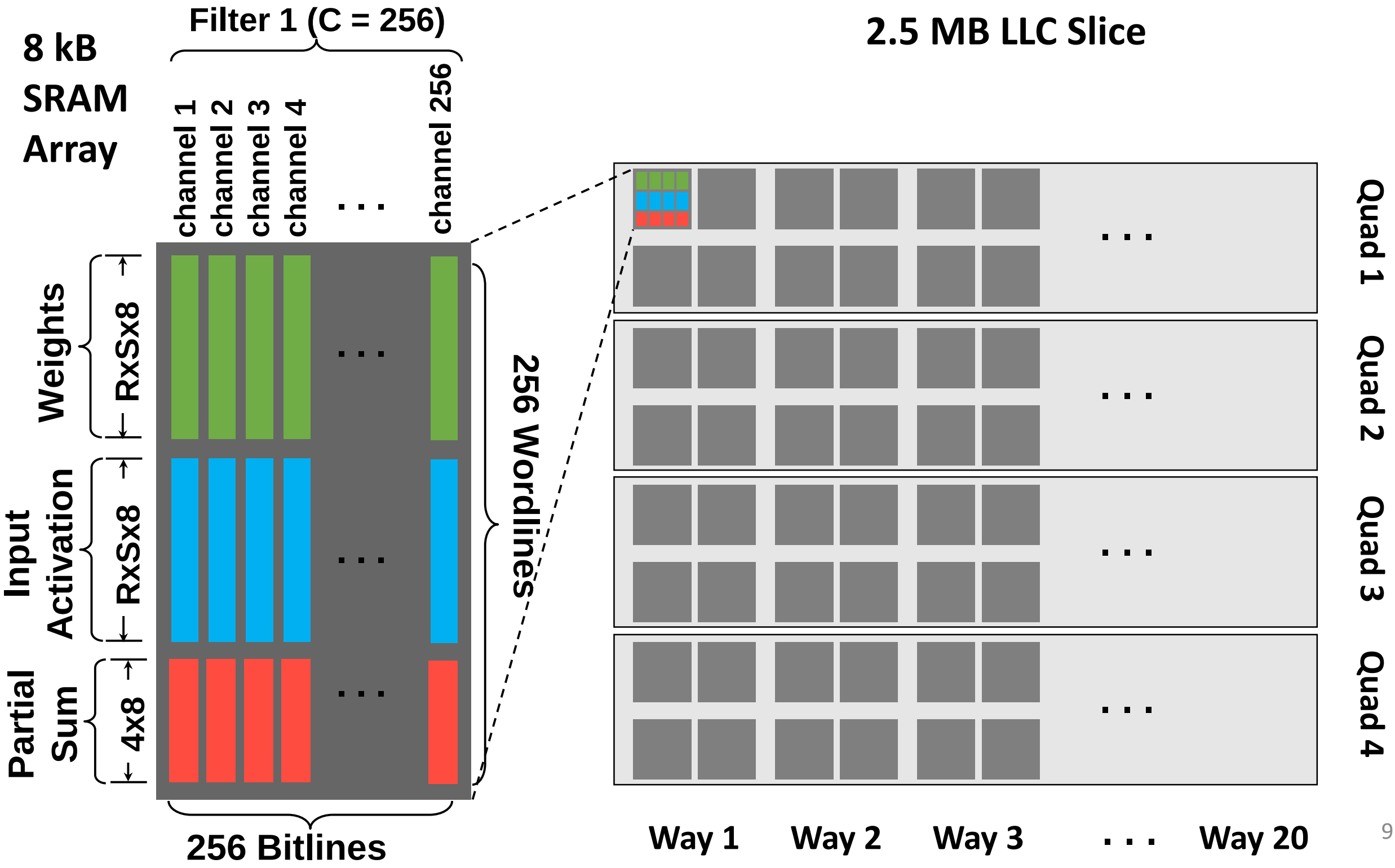


Mapping CNN to Neural Cache

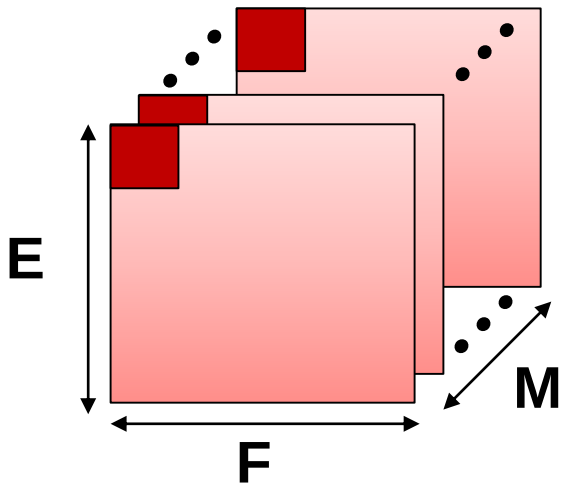


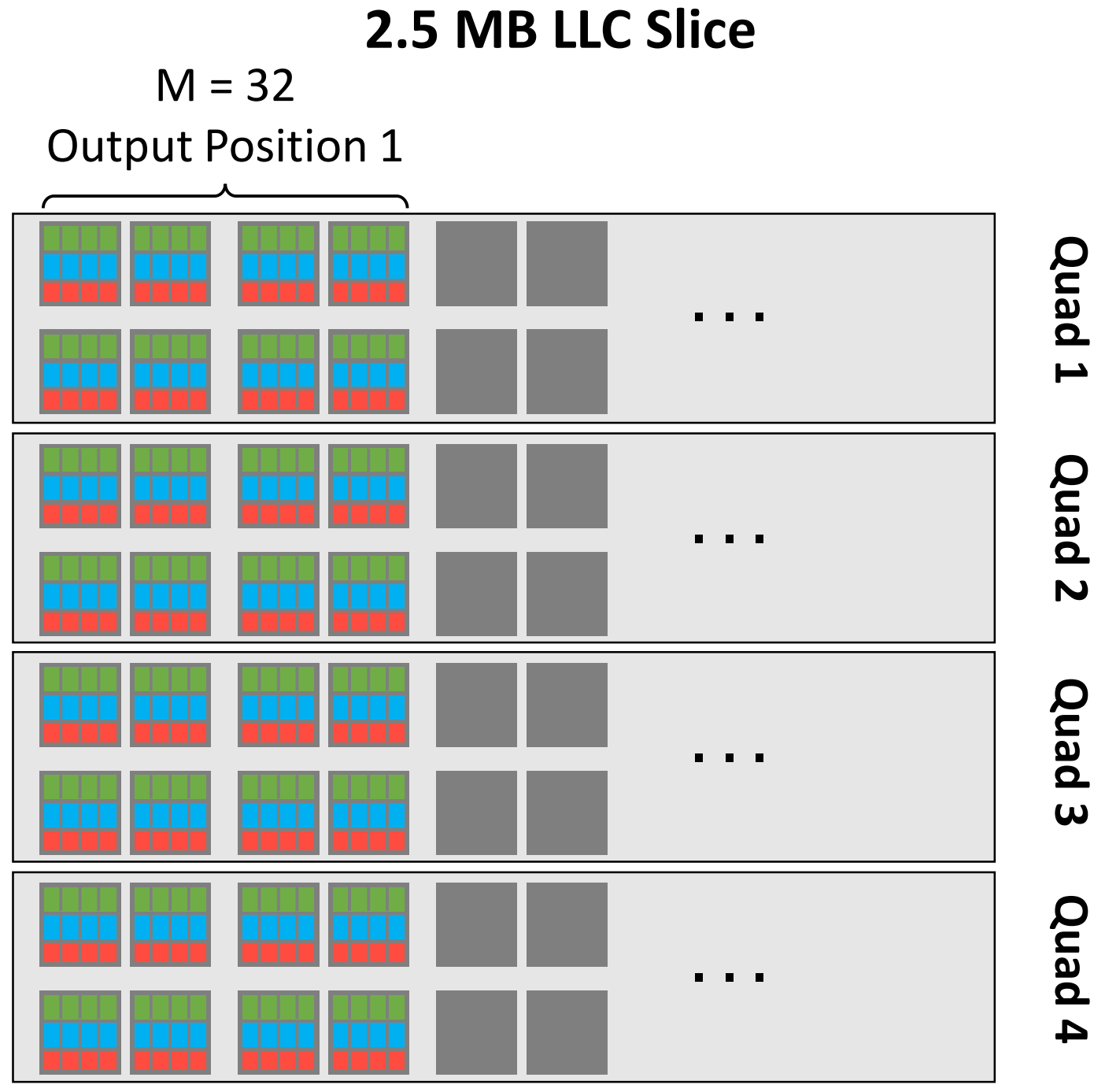


Mapping CNN to Neural Cache

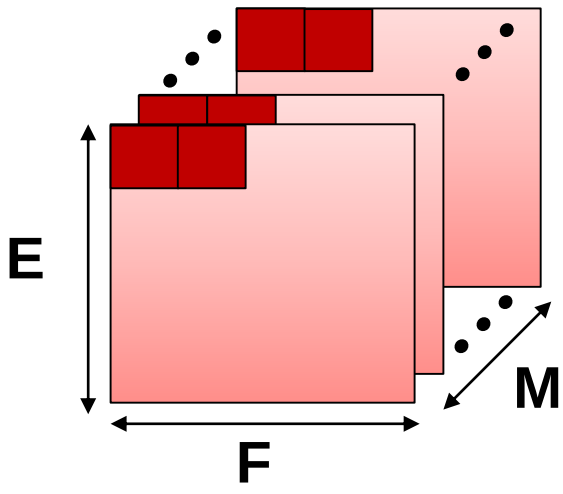


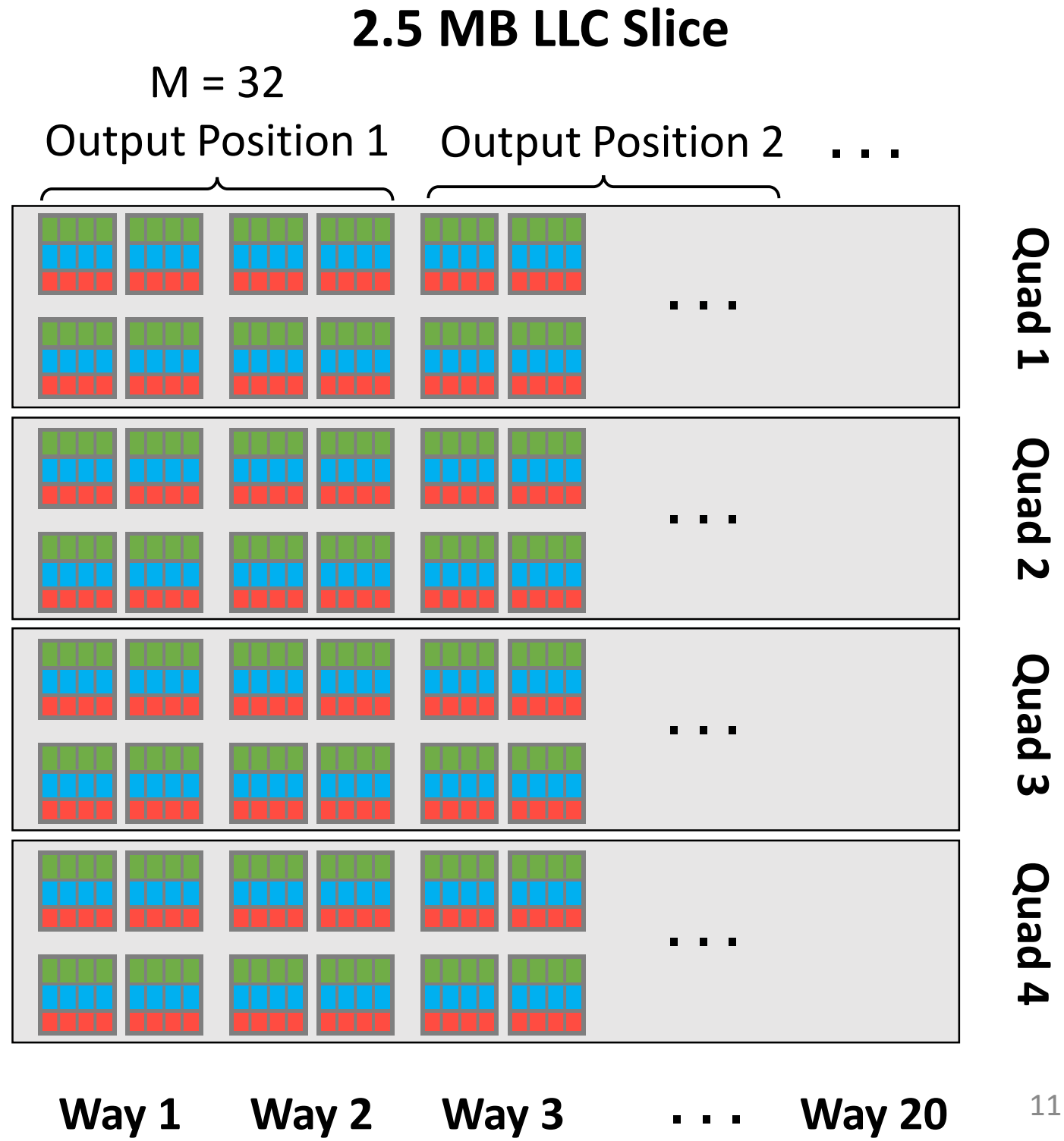
Mapping CNN to Neural Cache



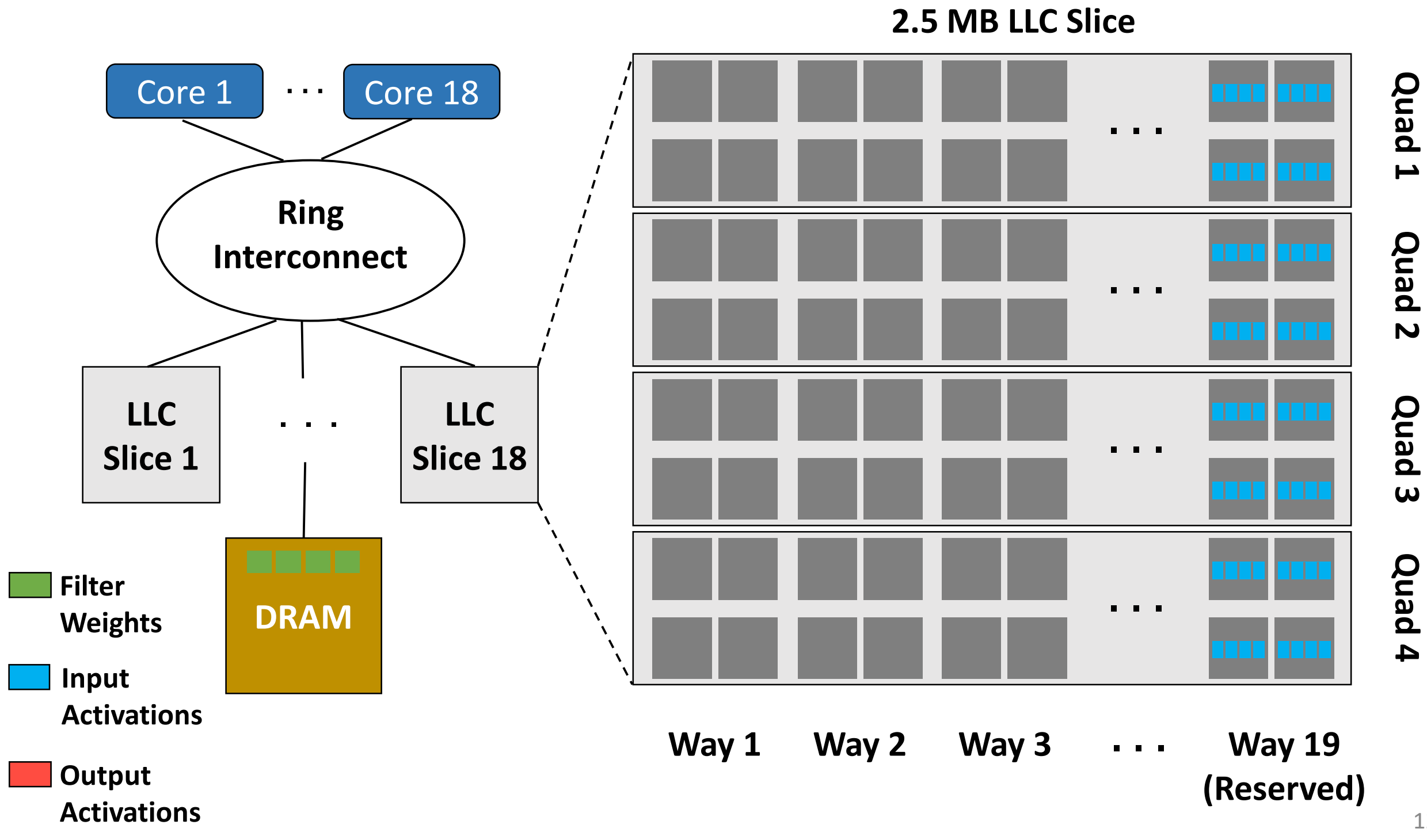


Mapping CNN to Neural Cache

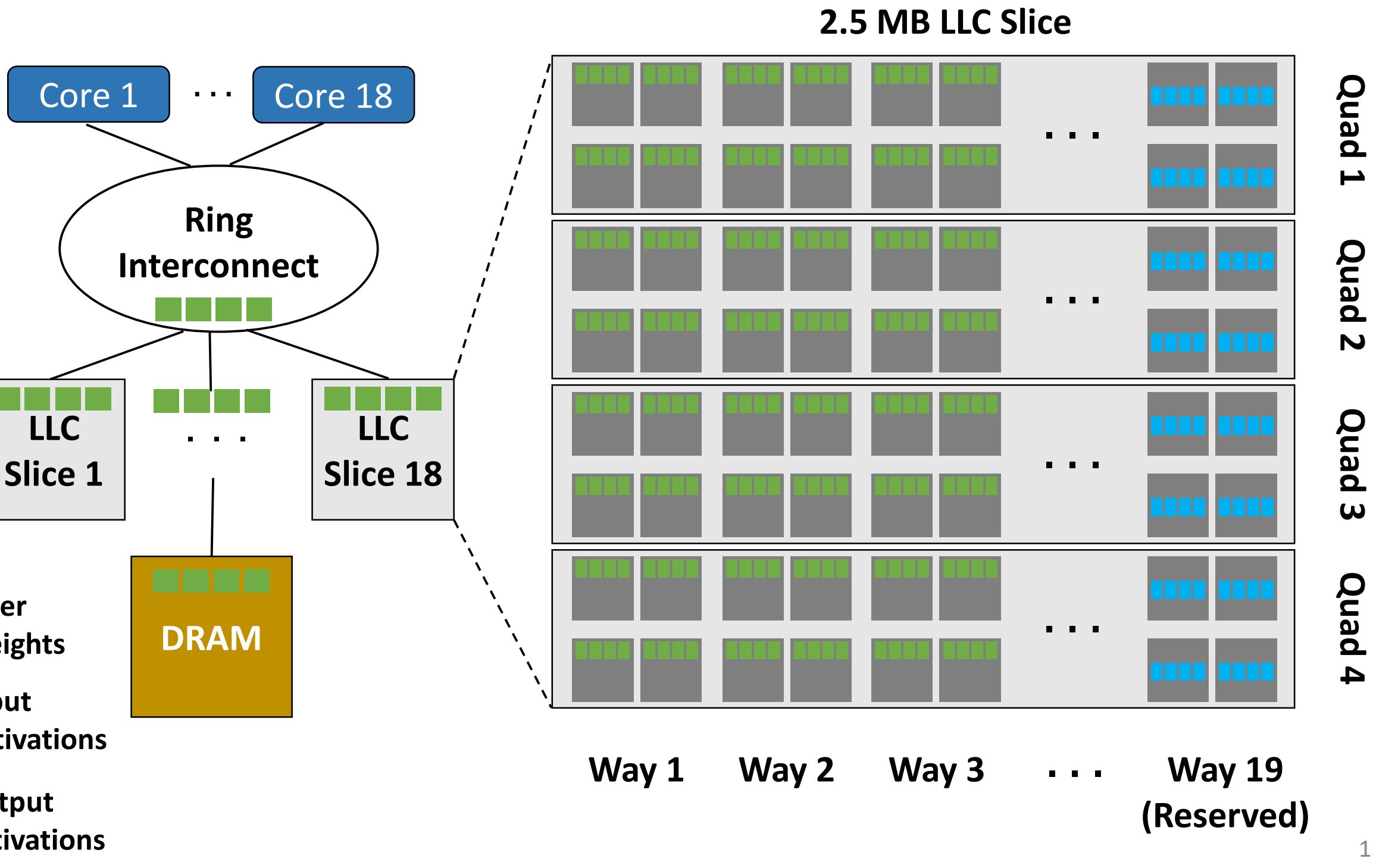




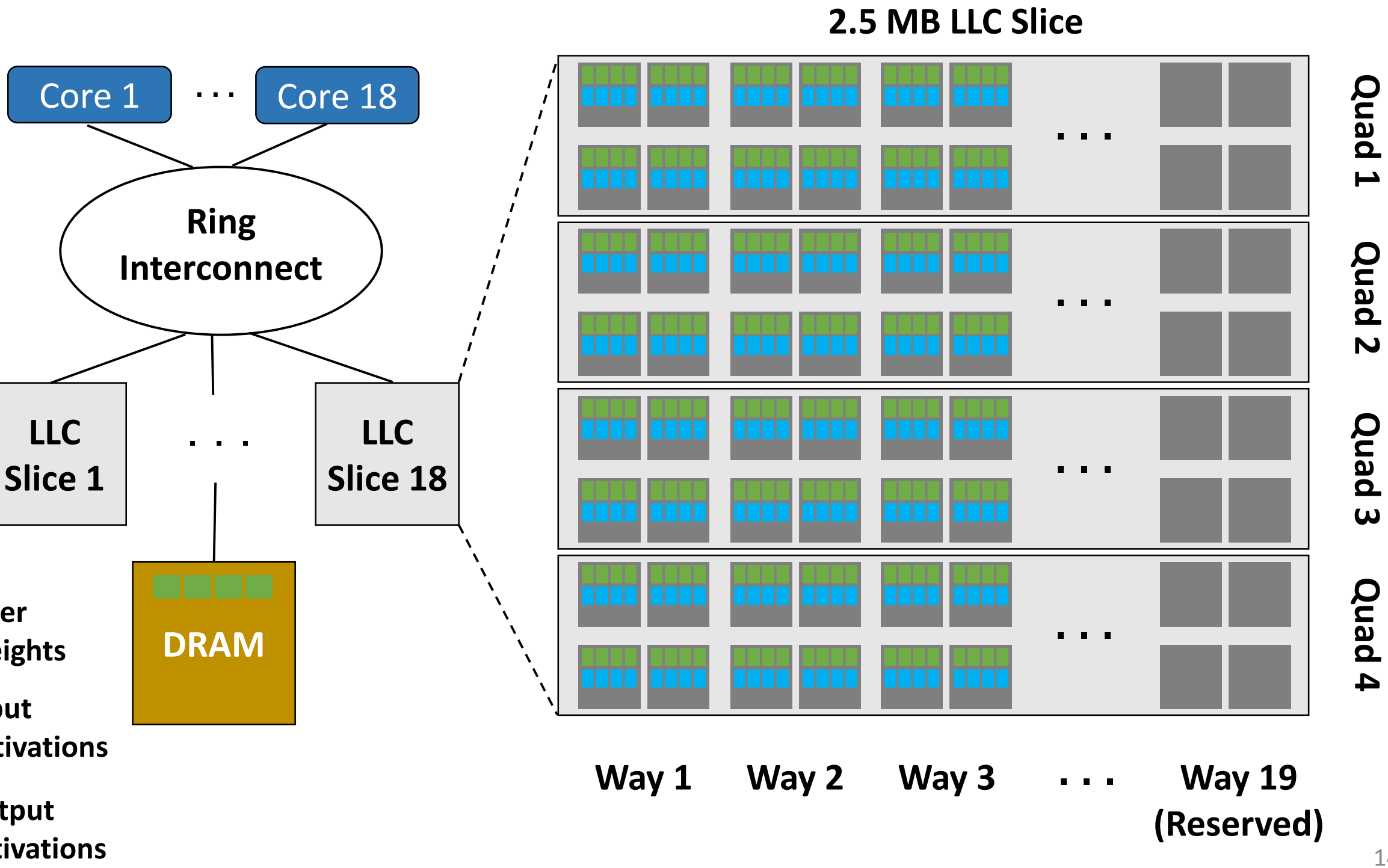
Put it together



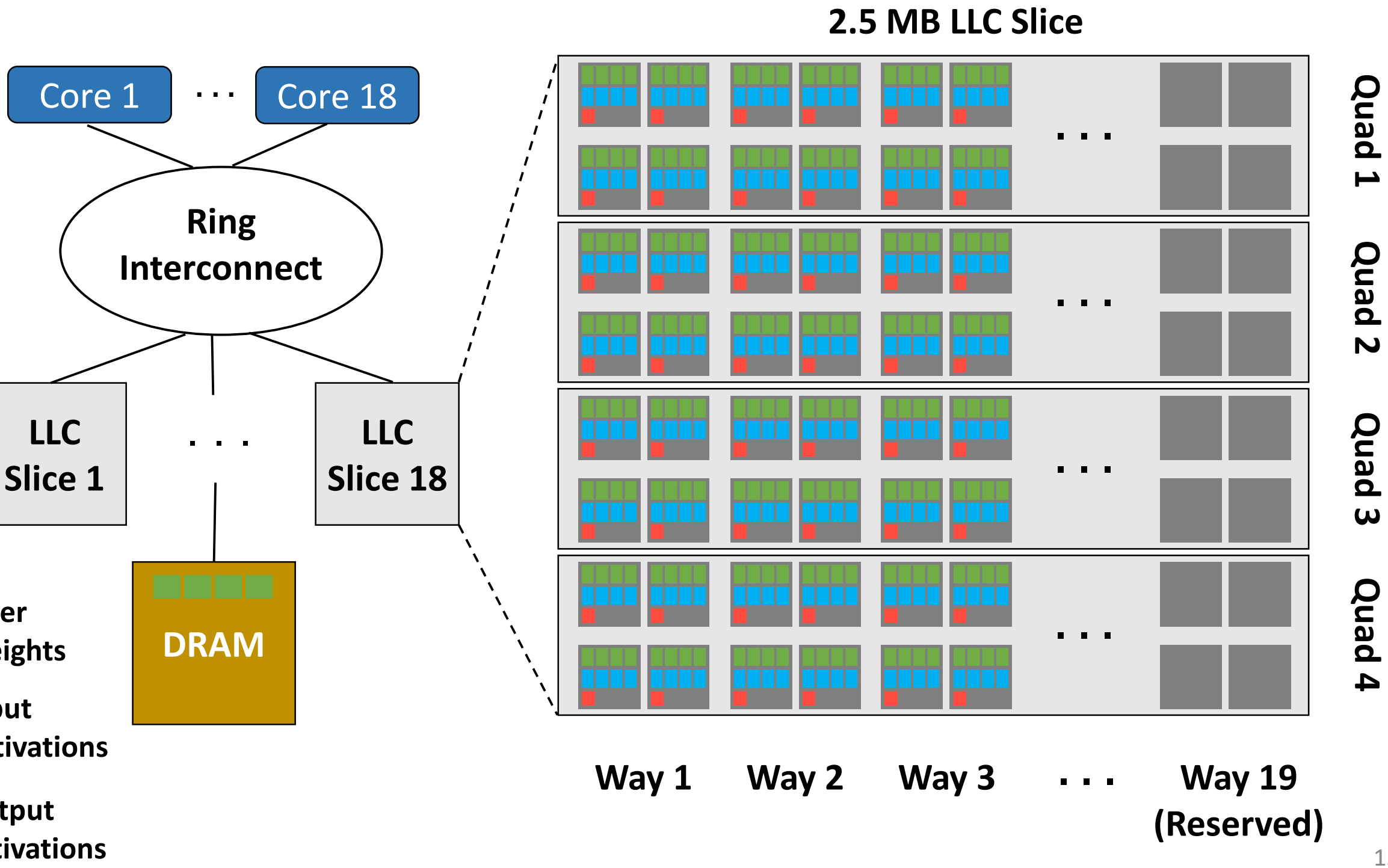
① Filter Loading



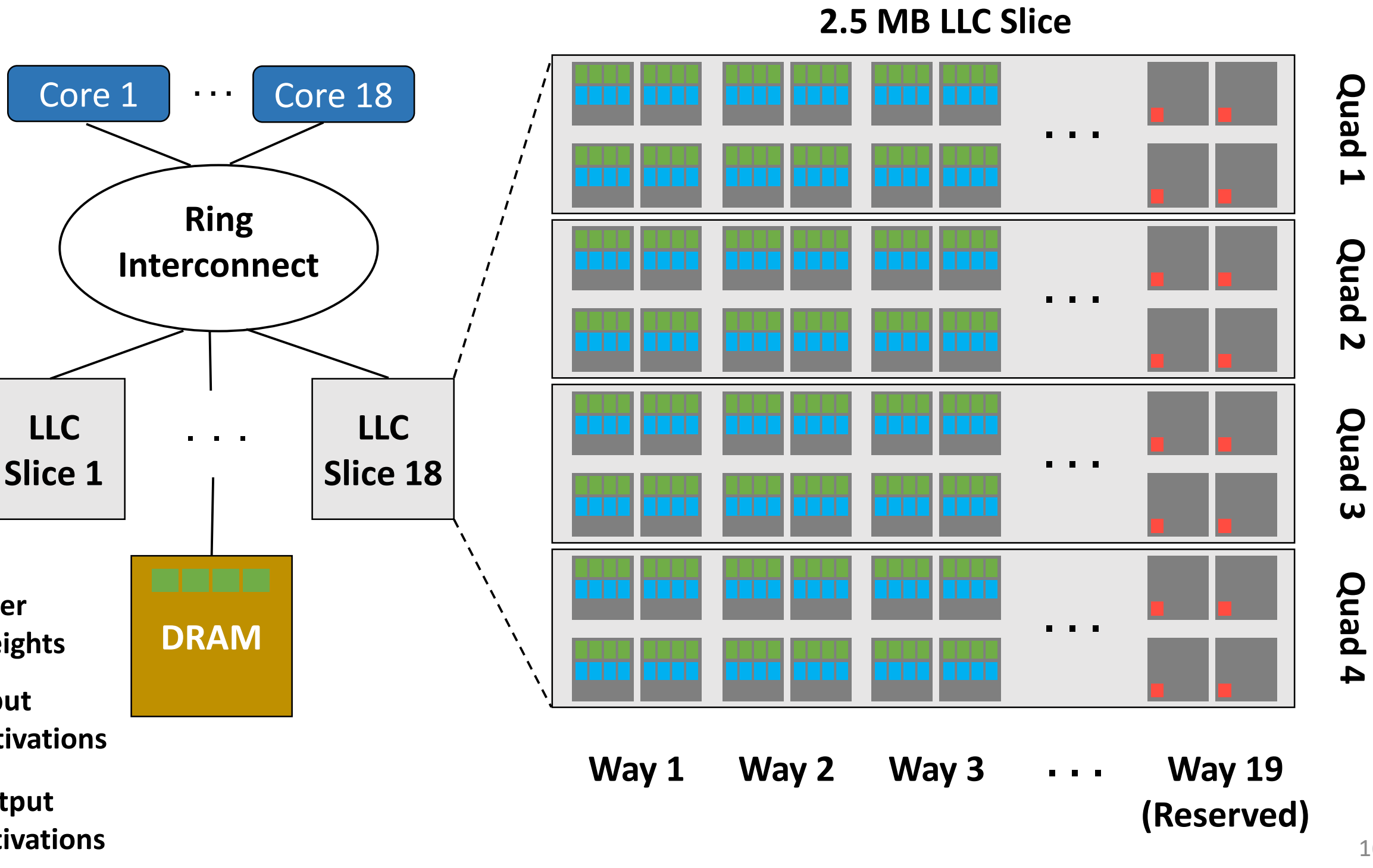
② Input Loading



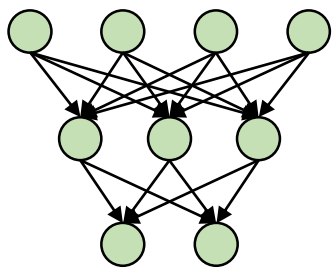
③ MAC + Reduction



④ Output Transfer

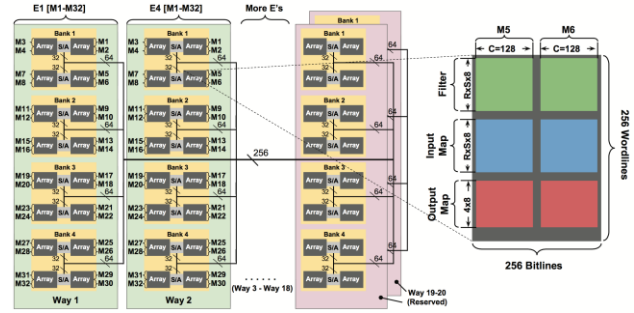


Motivation for This Work



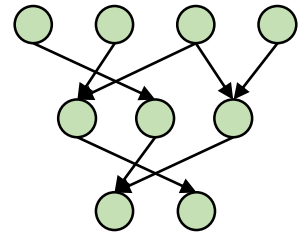
Convolutional Neural Networks

Bit-Serial In-SRAM computing

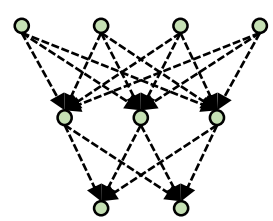


Neural Cache
(Customized Hardware)

Model Compression



Sparse Models
(num of weights ↓)



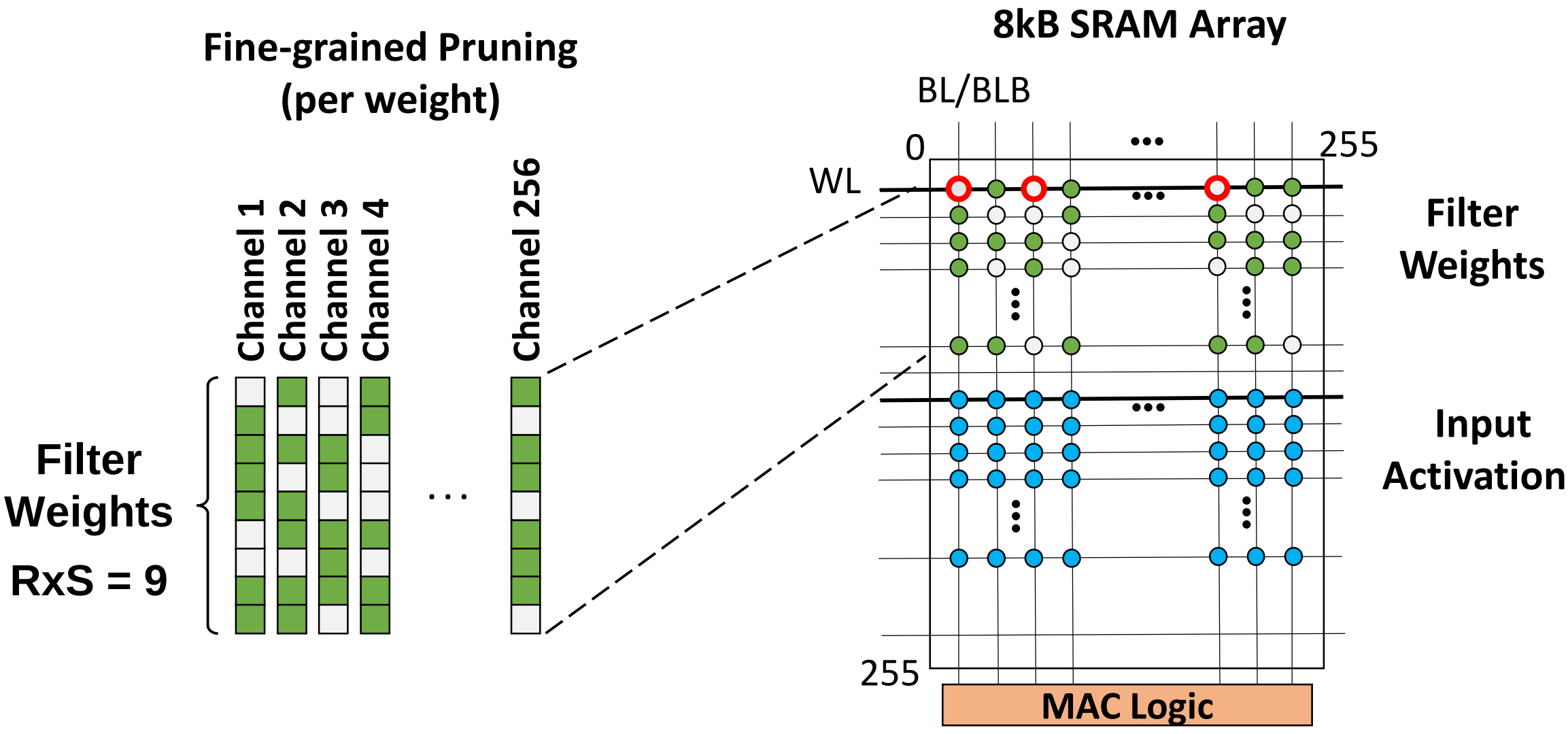
Low-Precision Models
(weight precision ↓)

+

This work:

Bit-Prudent In-Cache Acceleration of Deep CNNs

Sparse Models – Hard to gain speedup

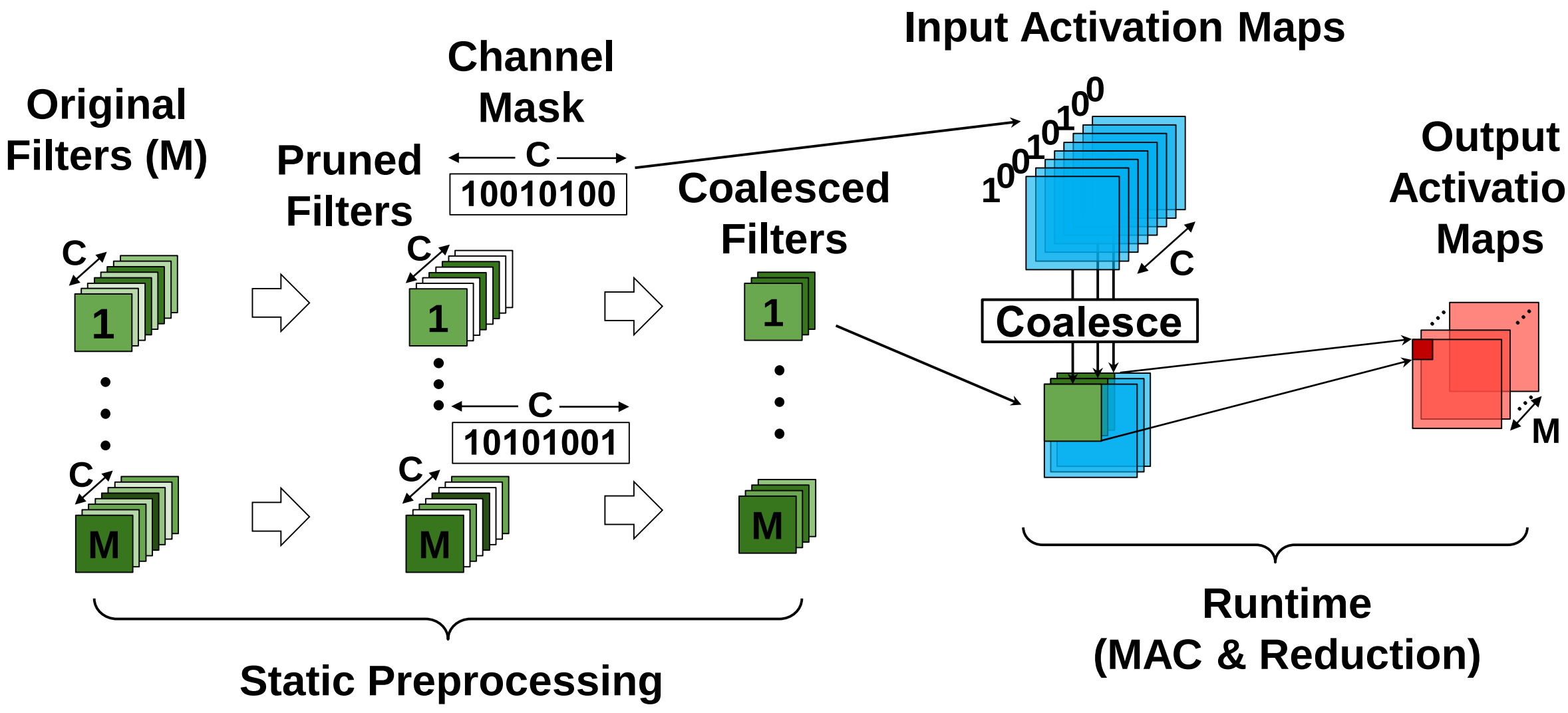


Unable to skip computation in SIMD model

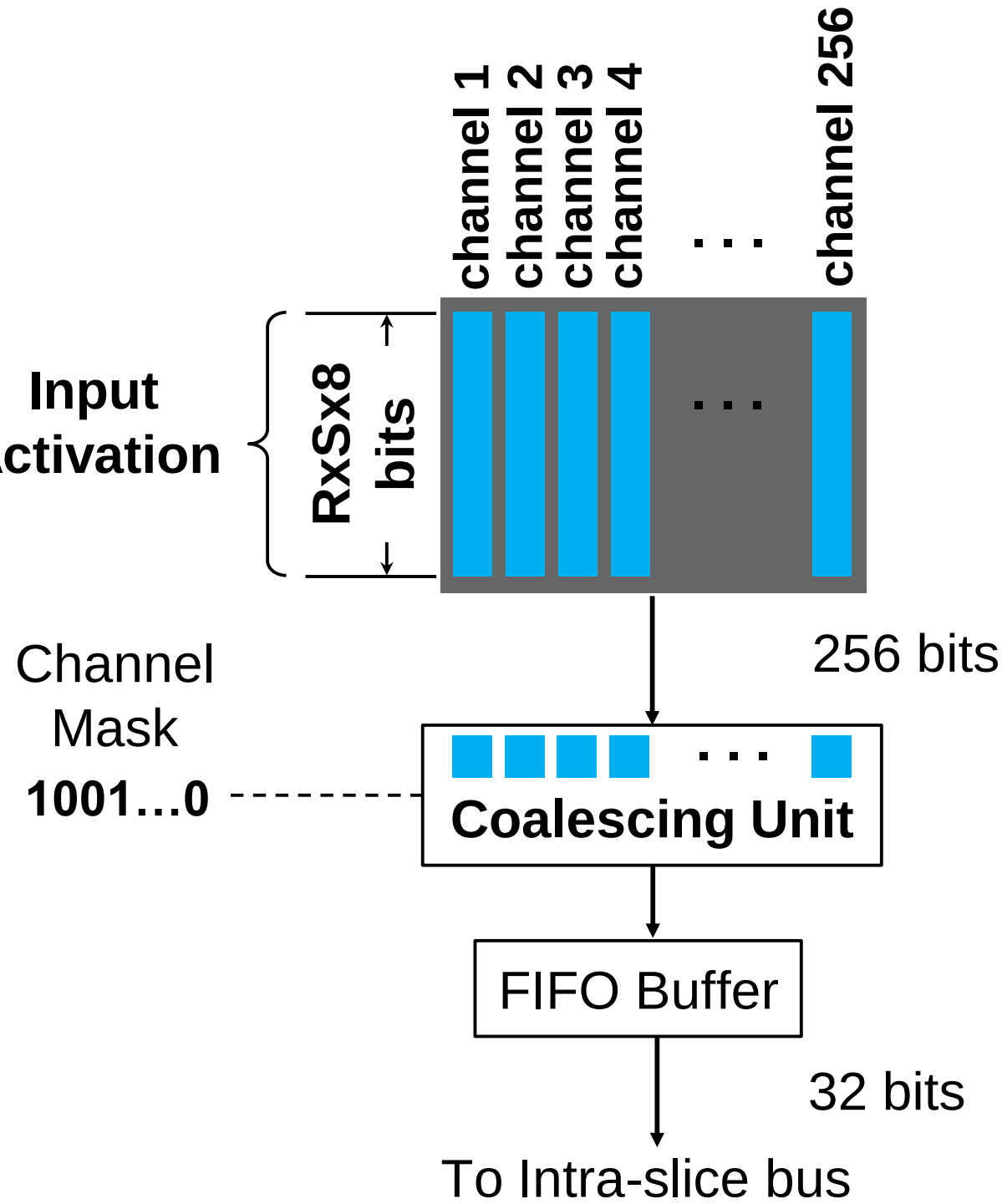
Outline

- Background
- Sparsity-Aware Architecture
  - **Coalescing channels**
  - Overlapping filters
- Low-Precision Architecture
- Evaluation Methodology
- Results

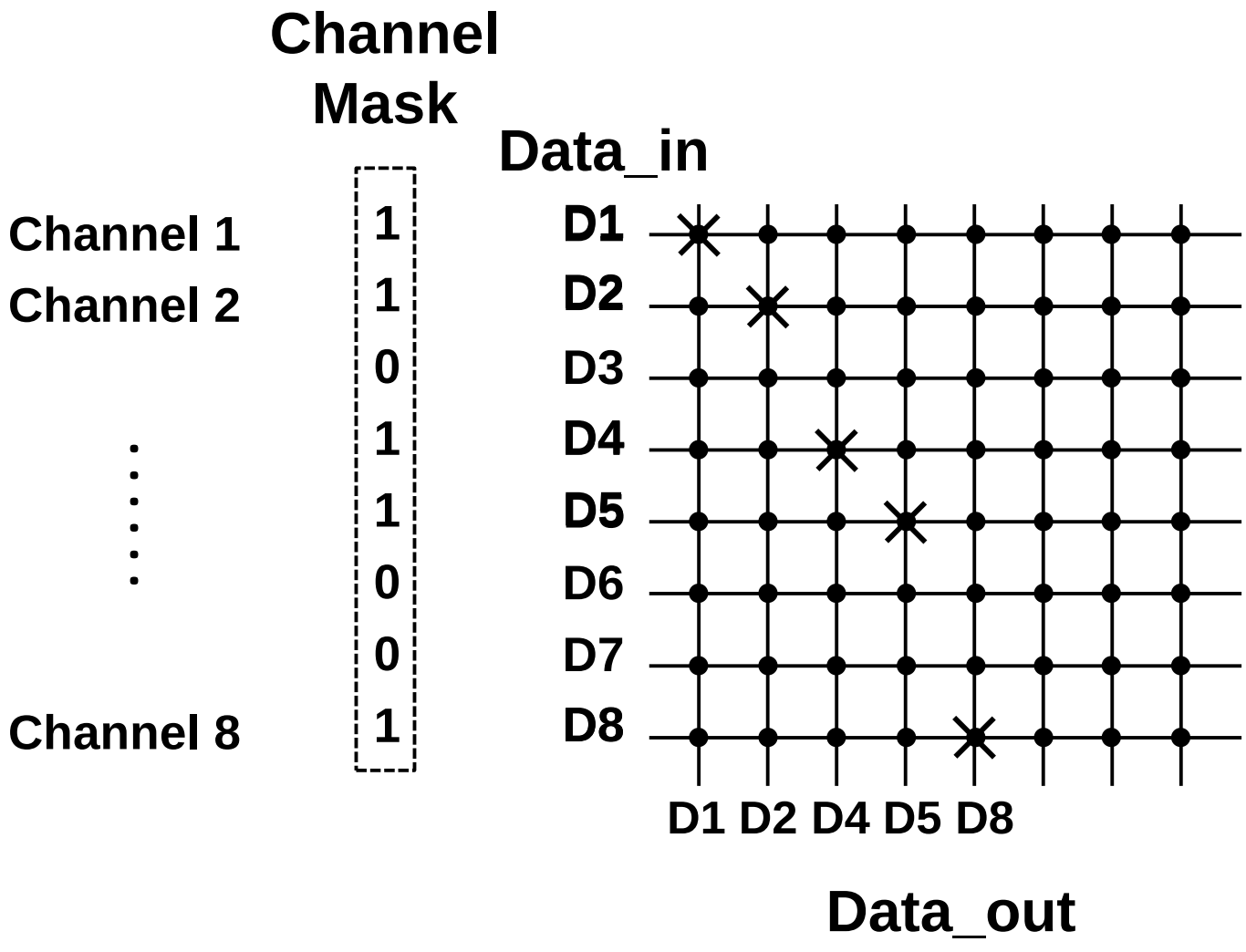
Architecture for Sparsity: **Coalescing**



Dynamic Input Coalescing

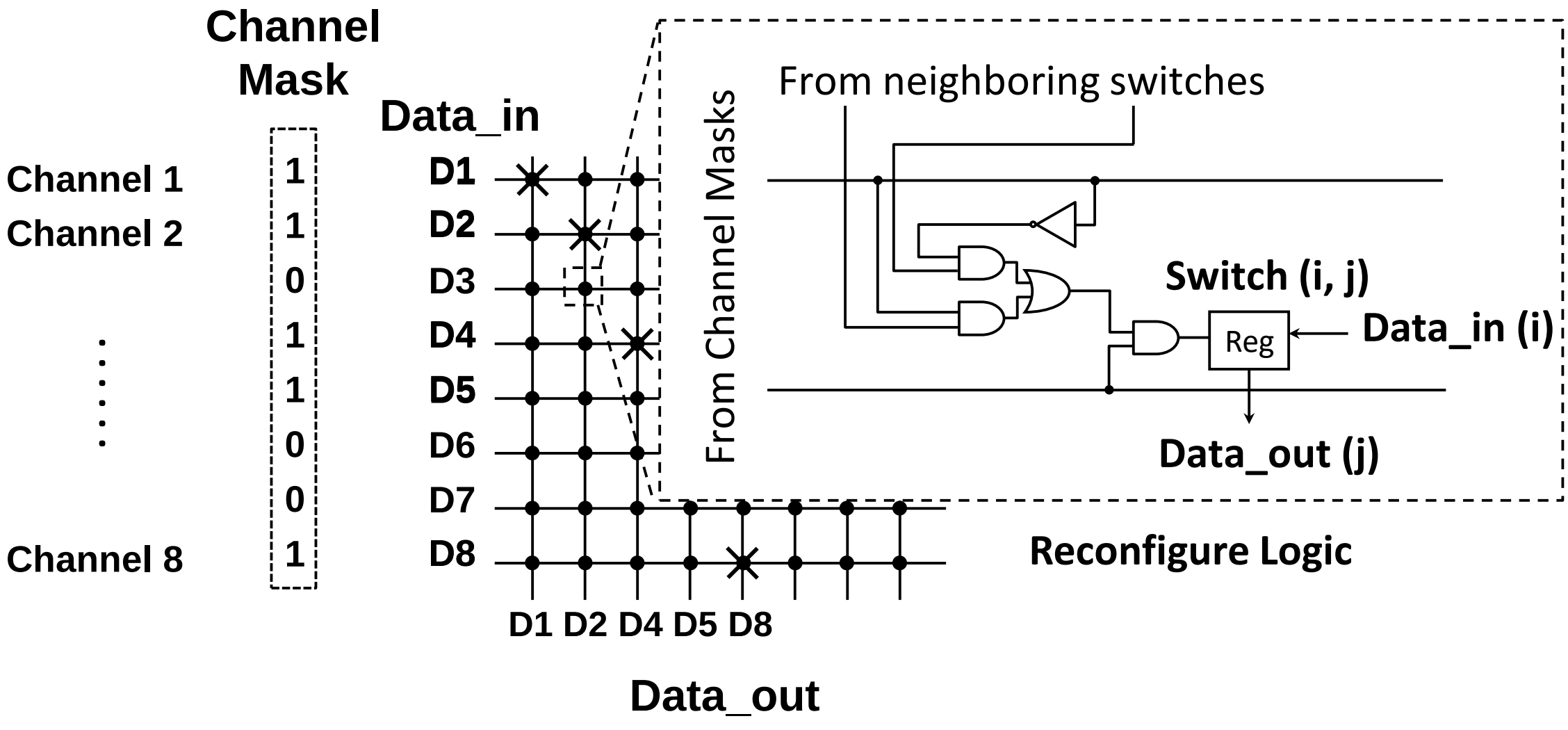


Coalescing Unit Architecture – 8 channel example



Coalescing Unit Architecture – 8 channel example

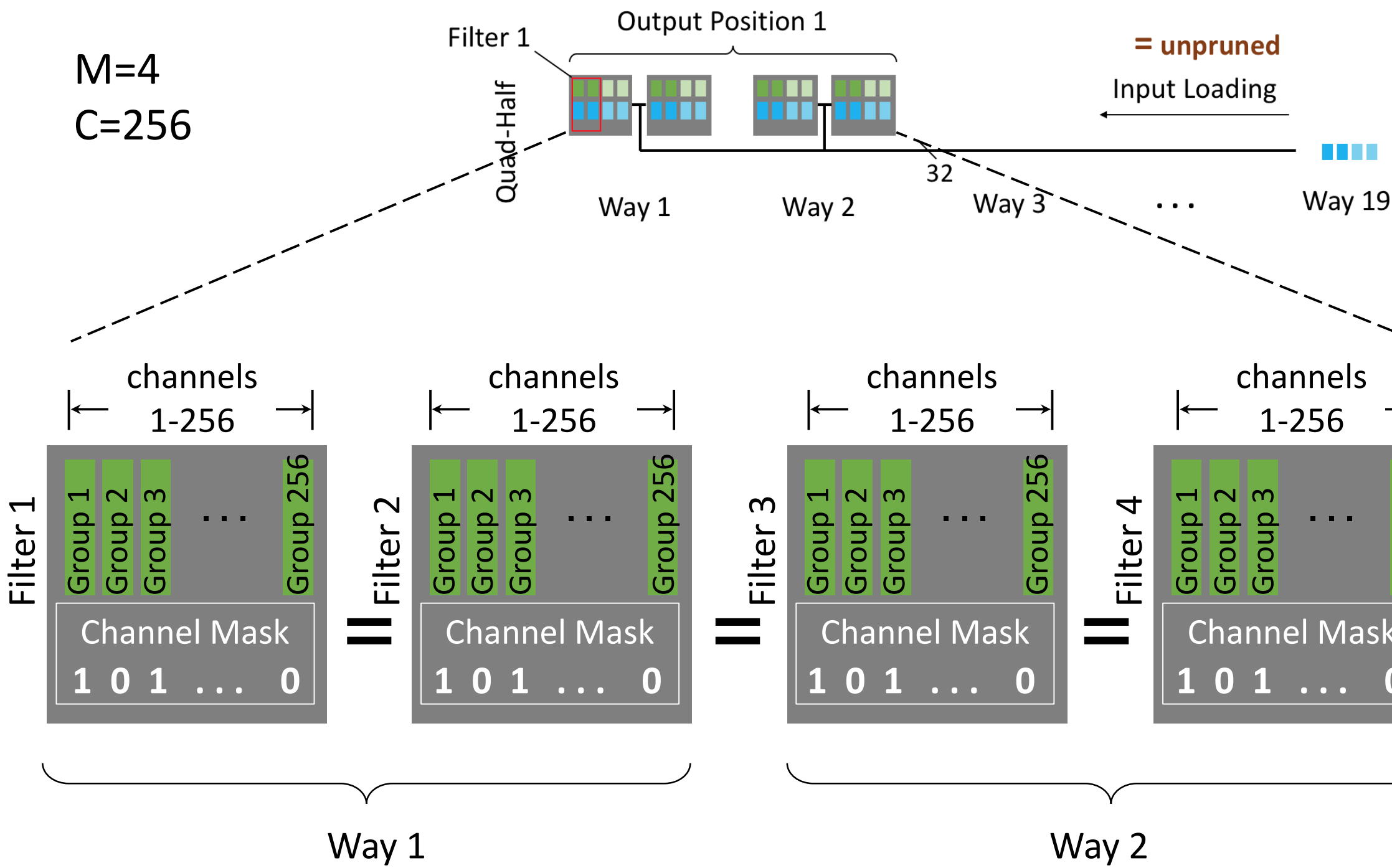
Dependency in **Combinational Logic**



Input Loading for Coalescing

Input-loading-aware Structured Pruning

M=4
C=256

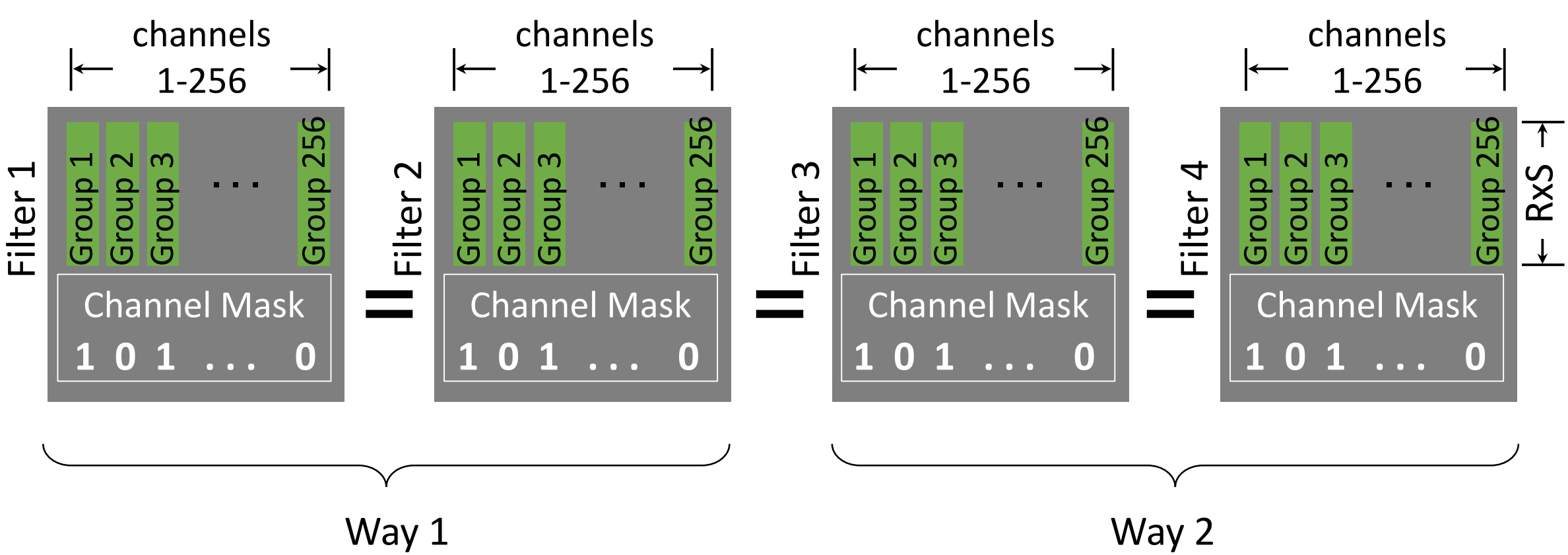


Input-loading-aware Structured Pruning

M=4
C=256

$$\sum_{r=1}^{R}\sum_{s=1}^{S}\sum_{m=1}^{M}(w_{r,s,m,c=1})^2$$

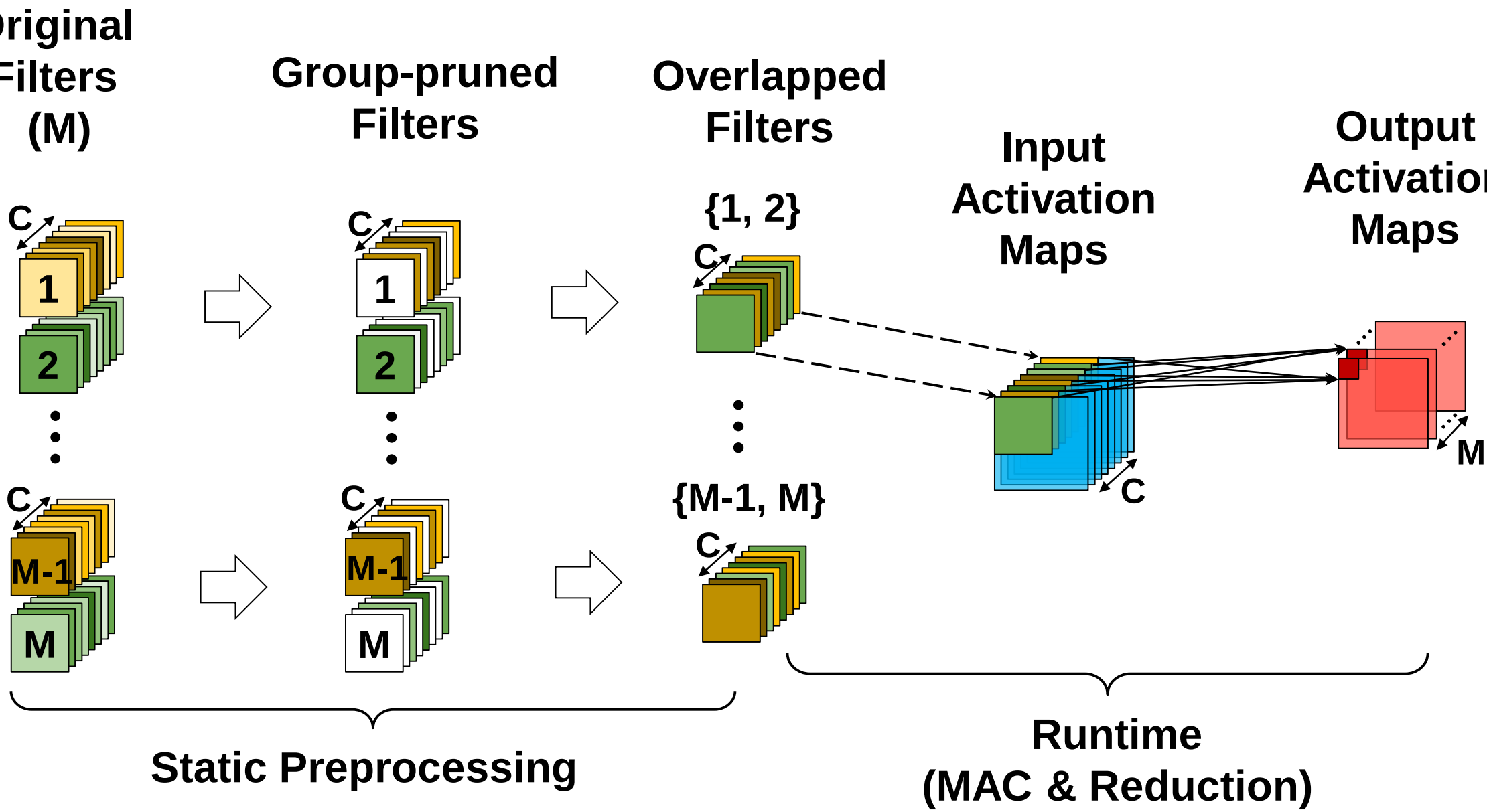
Channel masks are the same across all arrays



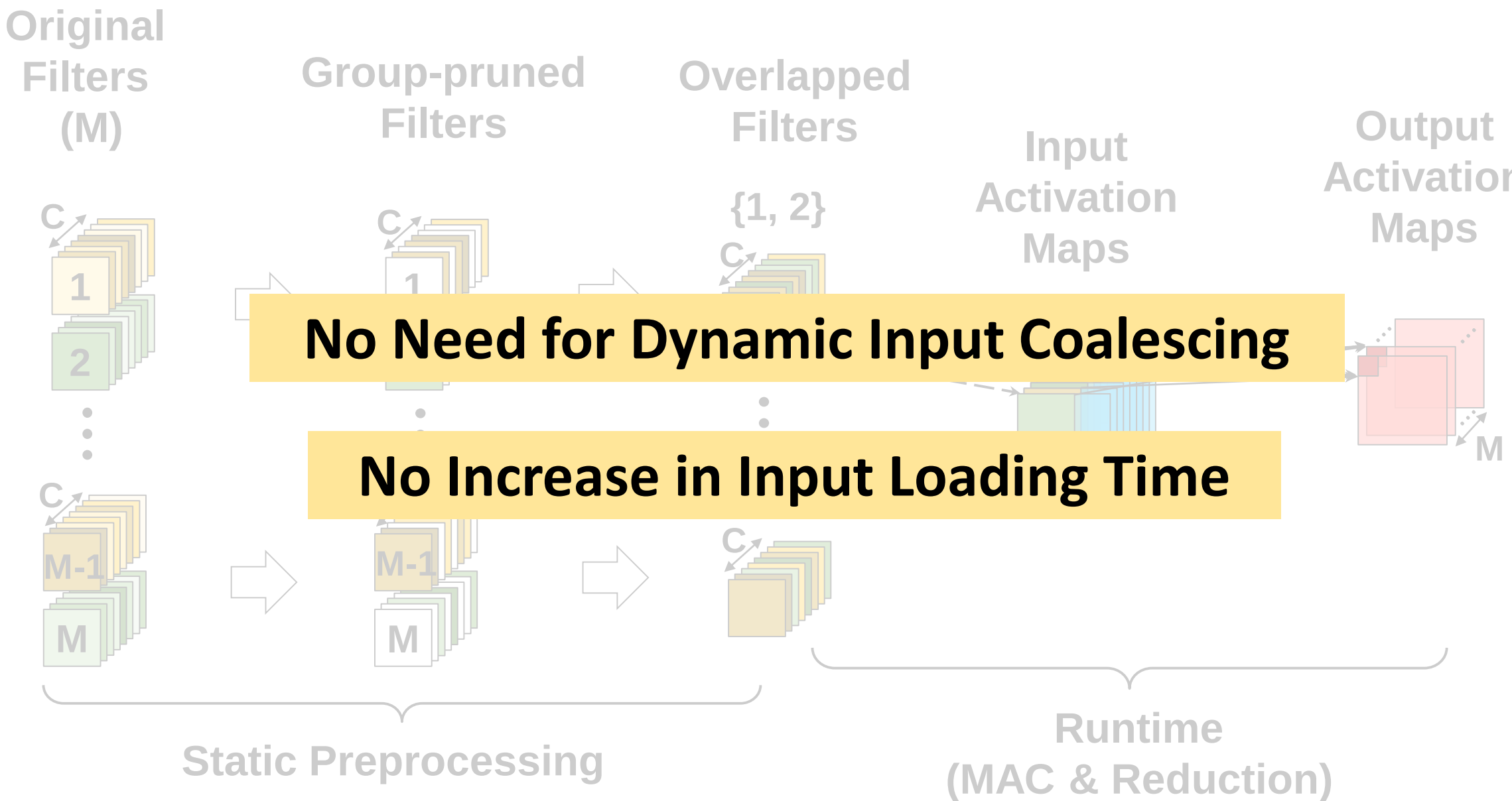
Outline

- Background
- Sparsity-Aware Architecture
  - Coalescing channels
  - **Overlapping filters**
- Low-Precision Architecture
- Evaluation Methodology
- Results

Architecture for Sparsity: **Overlapping**



Architecture for Sparsity: **Overlapping**



Outline

- Background
- Sparsity-Aware Architecture
- **Low-Precision Architecture**
- Evaluation Methodology
- Results

Architecture for MAC in Low-Precision Models

	c1	c2	c3	c4	
2-bit Weight	W[1]	**0**	**1**	**0**	**0** - Sign Bit
	W[0]	**1**	**1**	**1**	**0** - Magnitude Bit
4-bit Activation	A[3]	**0**	**1**	**1**	**1**
	A[2]	**1**	**0**	**1**	**0**
	A[1]	**0**	**0**	**0**	**1**
	A[0]	**1**	**1**	**0**	**0**
Product	P[3]	0	0	0	0
	P[2]	0	0	0	0
	P[1]	0	0	0	0
	P[0]	0	0	0	0
Accumulated Partial Sum	S[7]	1	0	0	1
	…	…	…	…	…
	S[3]	0	0	1	1
	S[2]	0	1	0	1
	S[1]	1	0	1	0
	S[0]	1	1	1	0

W[1:0]	00	01	11
Actual Value	0	1	-1

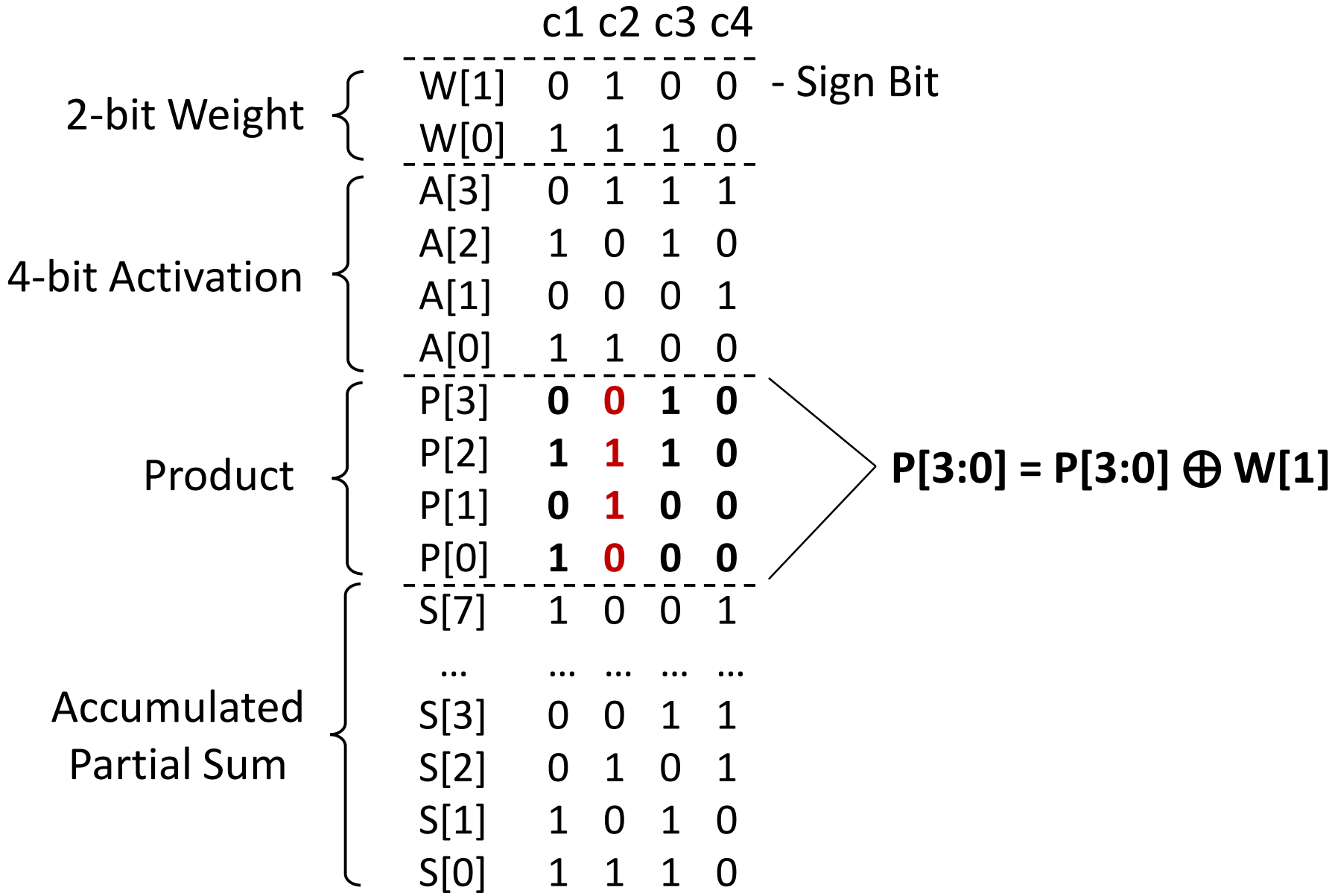
Architecture for MAC in Low-Precision Models

Step 1: Multiply Zero Weights

		c1	c2	c3	c4	
2-bit Weight	W[1]	0	1	0	0	
	W[0]	1	1	1	0	- Magnitude Bit
4-bit Activation	A[3]	0	1	1	1	
	A[2]	1	0	1	0	
	A[1]	0	0	0	1	
	A[0]	1	1	0	0	
Product	P[3]	**0**	**1**	**1**	**0**	**P[3:0] = A[3:0] & W[0]**
	P[2]	**1**	**0**	**1**	**0**	
	P[1]	**0**	**0**	**0**	**0**	
	P[0]	**1**	**1**	**0**	**0**	
Accumulated Partial Sum	S[7]	1	0	0	1	
	…	…	…	…	…	
	S[3]	0	0	1	1	
	S[2]	0	1	0	1	
	S[1]	1	0	1	0	
	S[0]	1	1	1	0	

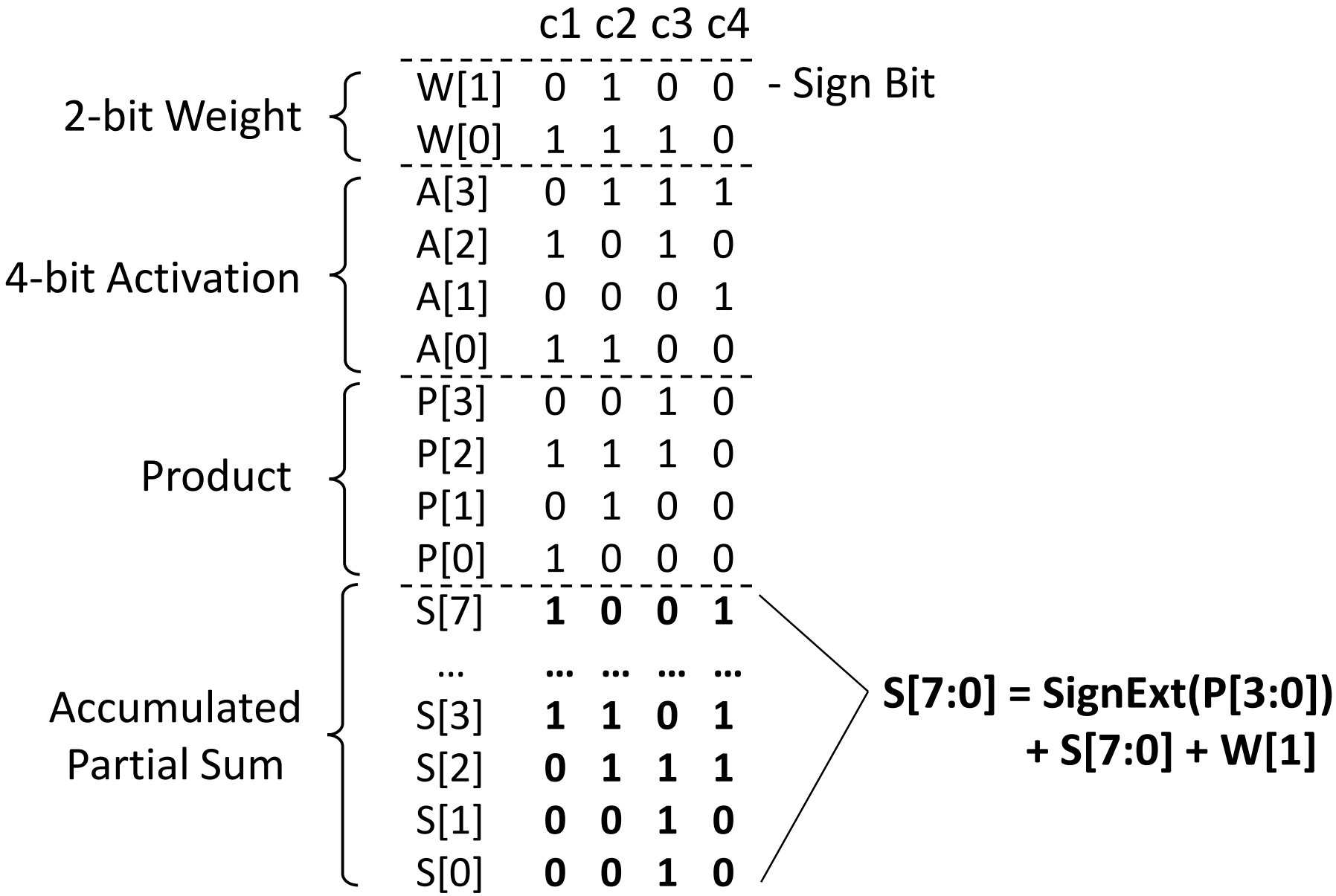
Architecture for MAC in Low-Precision Models

Step 2: Calculate 1's complement



Architecture for MAC in Low-Precision Models

Step 3: Partial sum accumulation (2's complement conversion)



MAC cycle goes down with Low-Precision Arch.

Bit-width Configuration	MAC cycles
8-bit weight/ 8-bit activation	94
Ternary weight/ 4-bit activation	12
Binary weight/ 4-bit activation	8

Evaluation Methodology

CNN Models

- AlexNet and Inception v3 convolutional layers
- Performance-aware pruning
- 8-bit quantization for weights and inputs

Training Methods

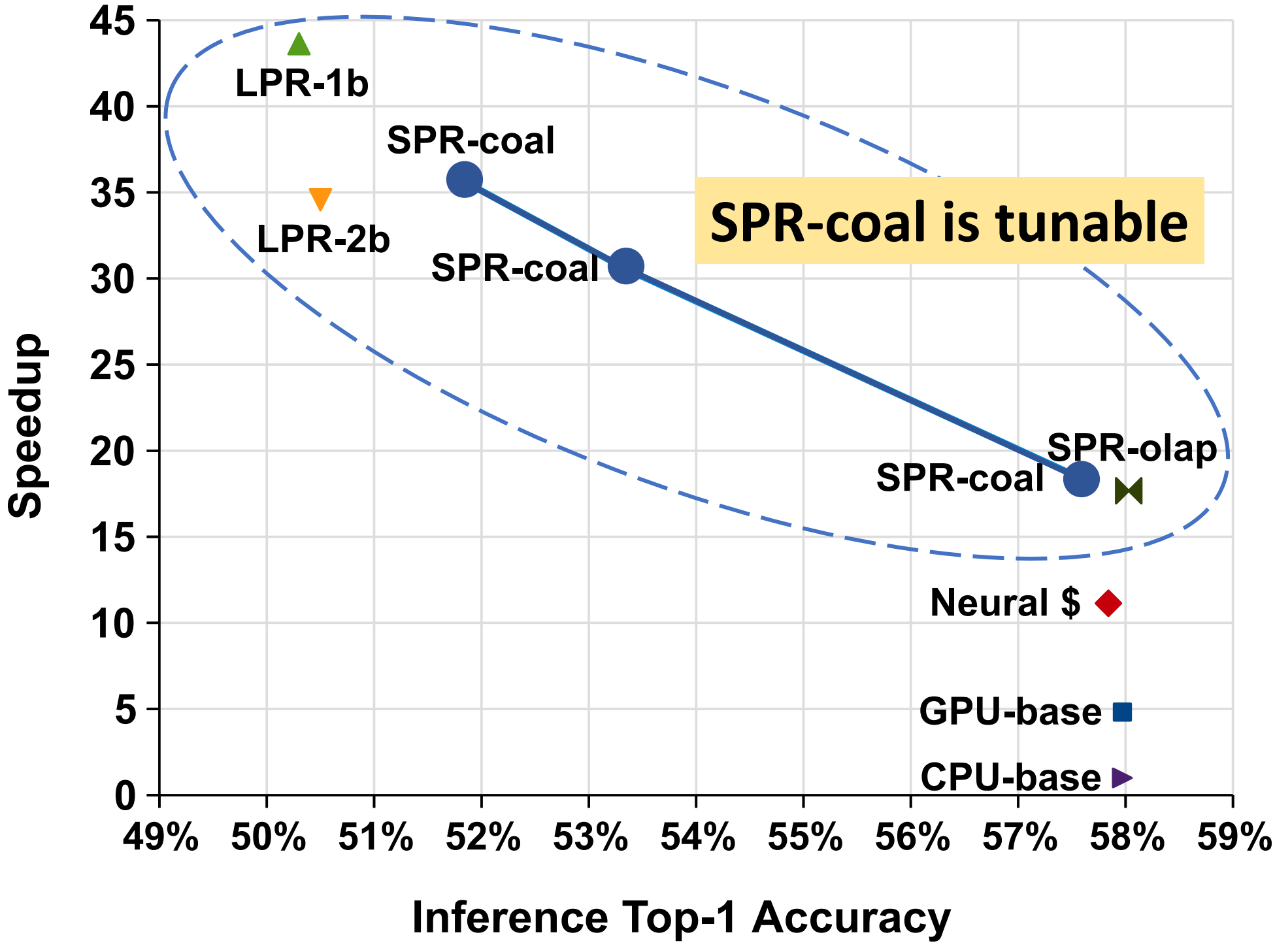
- modified TensorFlow tool-chain

Configurations

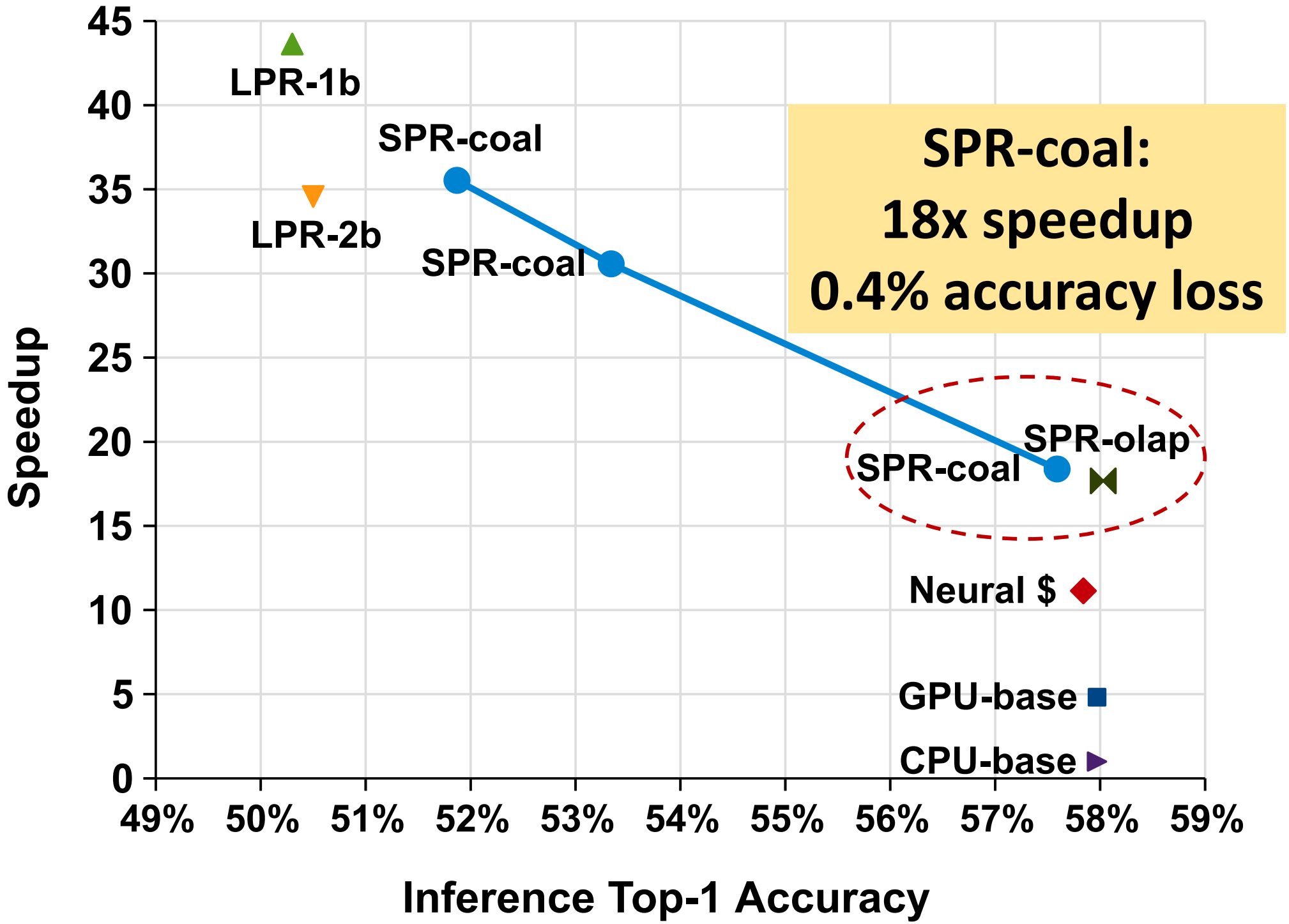
- CPU-base
- GPU-base
- Neural $
- SPR-coal/SPR-olap
- LPR-2b/LPR-1b

	CPU (2 sockets)	GPU (1 card)	Neural Cache
Processor	Intel Xeon E5-2597 v3, 2.6GHz, 28 cores, 56 threads	Nvidia Titan Xp, 1.6GHz, 3840 CUDA cores	2.5GHz Compute SRAM, 1032192 Bit-serial ALUs
On-chip memory	78.96 MB	9.14 MB	35 MB
Off-chip memory	64 GB DRAM	12 GB DRAM	64 GB DRAM
Profiler / Simulator (Performance)	TensorFlow tfprof	TensorFlow tfprof	Cycle accurate simulator + C Microbench
Profiler / Simulator (Energy)	Intel RAPL Interface	NVIDIA System Management Interface	SPICE simulation + Intel RAPL Interface

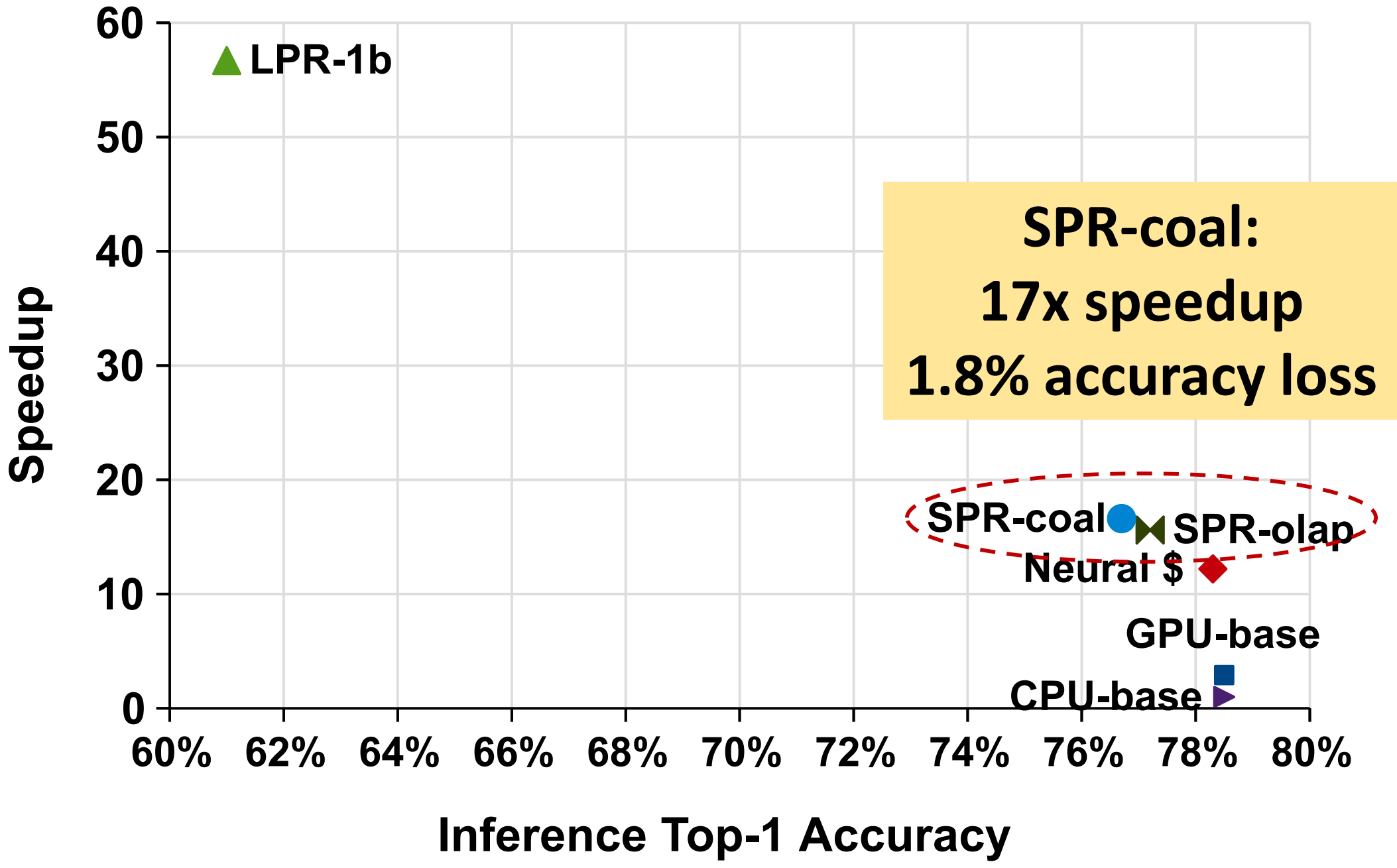
Performance/Accuracy Trade-off (AlexNet)



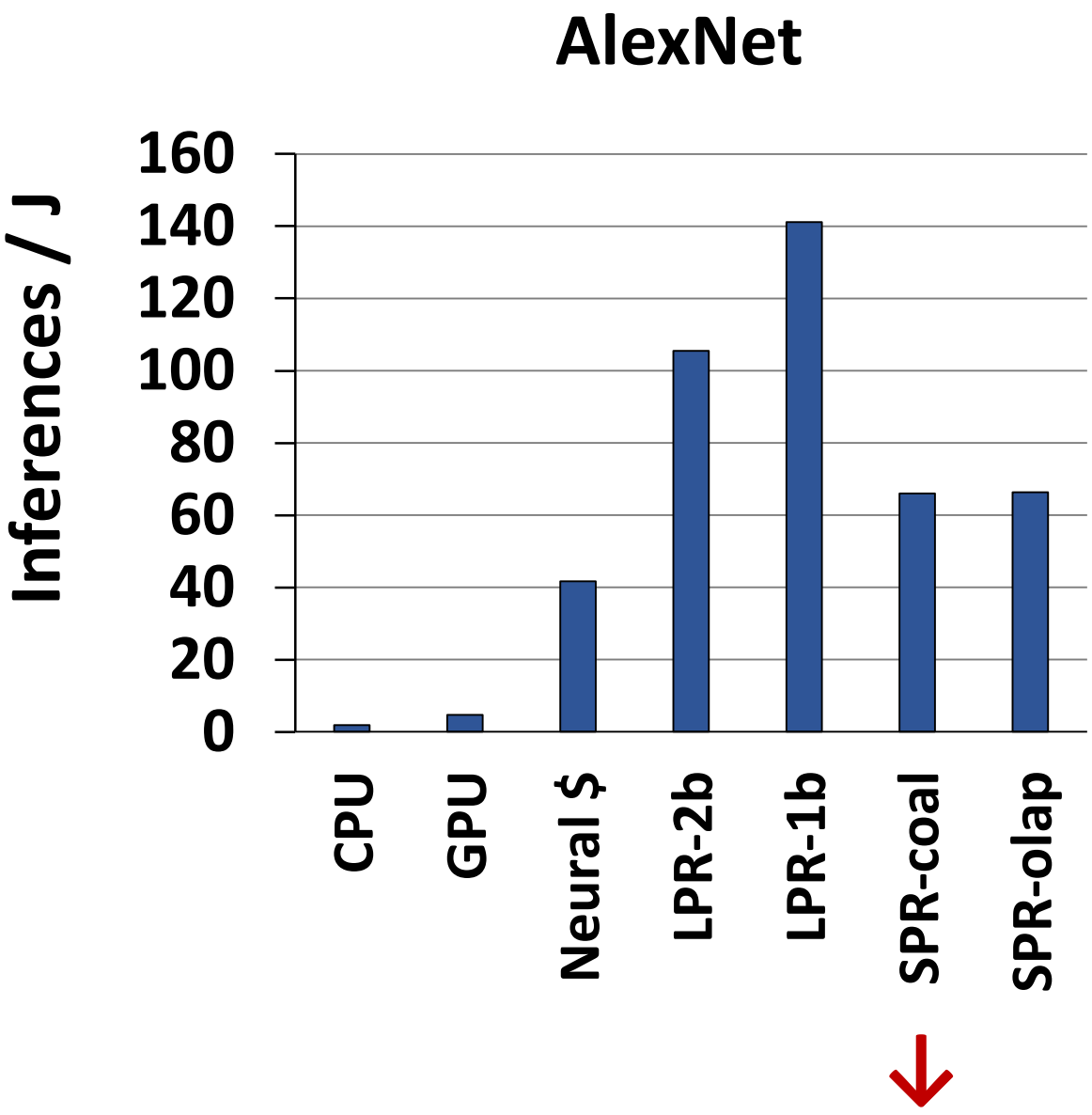
Performance/Accuracy Trade-off (AlexNet)



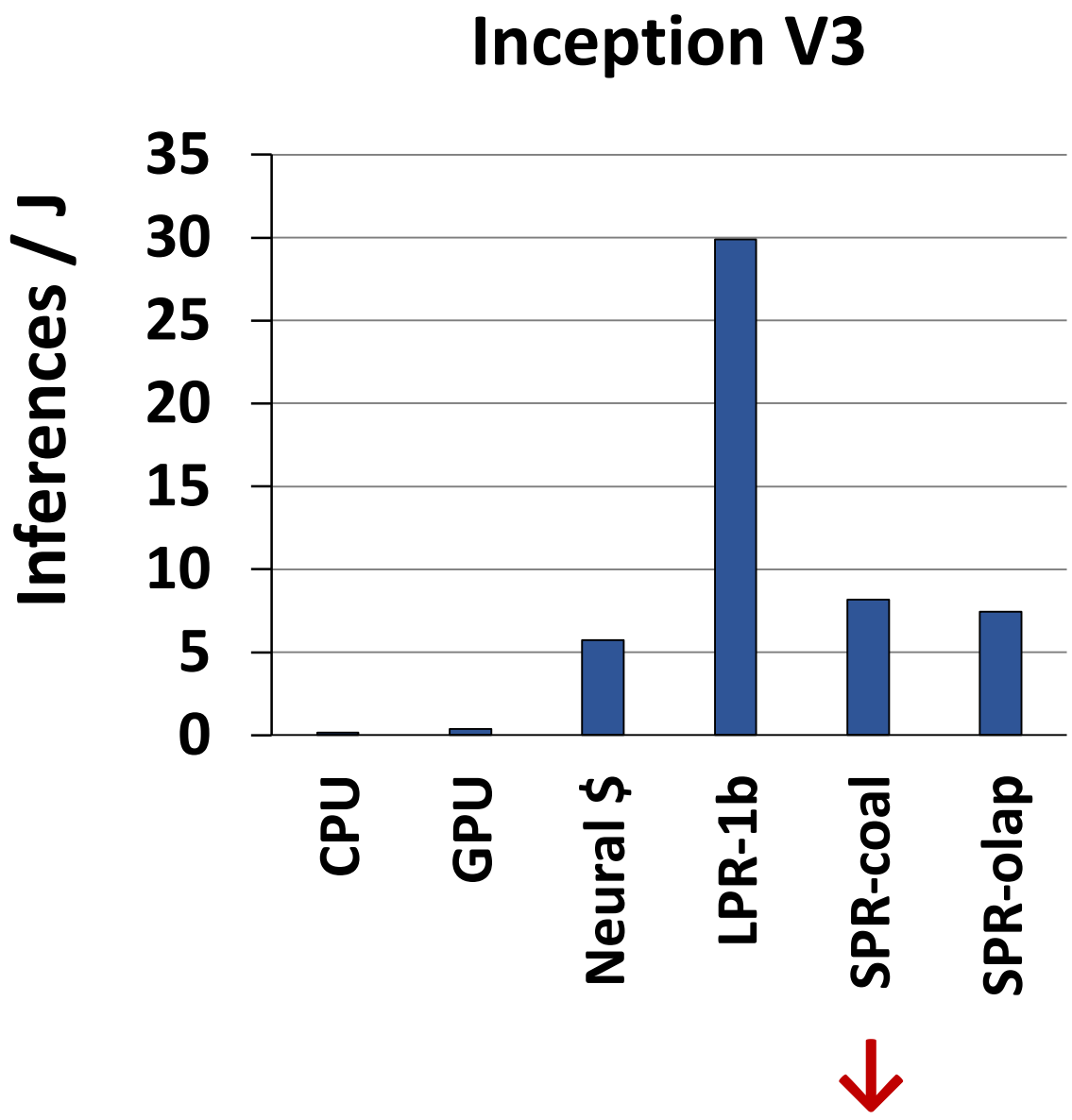
Performance/Accuracy Trade-off (Inception V3)



Energy Efficiency for All Conv. Layers



34x over CPU

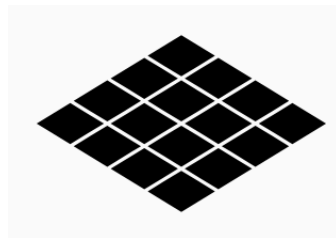


50x over CPU

Summary

Improving on a highly parallel in-cache accelerator by exploiting …

1) **sparsity** in filter weights:
    a) offline **coalescing** of weights + online **coalescing** of inputs
    b) **overlapping** filters
2) **low-precision** weights:
    efficient MAC algorithm

Overall Efficiency	Accuracy	Area Overhead
		
622x over CPU 52x over GPU 2.6x over Neural Cache	0.4% accuracy loss	2% of CPU chip

Bit Prudent In-Cache Acceleration of Deep Convolutional Neural Networks

Xiaowei Wang, Jiecao Yu, Charles Augustine,

Ravi Iyer, Reetuparna Das

Thank you!

Questions & Answers

M-Bits Research Group

UNIVERSITY OF MICHIGAN