

tinyML® Summit

Miniature dreams can come true...

March 28-30, 2022 | San Francisco Bay Area



www.tinyML.org

Programmable In-Memory Computing (IMC) Accelerator with >100 SRAM IMC Macros

Jae-sun Seo

Arizona State University, USA



<https://faculty.engineering.asu.edu/jseo>



**2022 TinyML Summit
March 30, 2022**

Outline

- Background & Motivation
- IMC Macro Design and Challenges of IMC Systems
- Programmable IMC Accelerator (PIMCA)
 - Overall Architecture
 - ISA & Loop Support Control
 - Activation Storage & Access
 - SIMD Design
 - PE Mapping
- Measurement & Comparison
- Summary

Outline

□ **Background & Motivation**

- IMC Macro Design and Challenges of IMC Systems
- Programmable IMC Accelerator (PIMCA)
 - Overall Architecture
 - ISA & Loop Support Control
 - Activation Storage & Access
 - SIMD Design
 - PE Mapping
- Measurement & Comparison
- Summary

Success of AI Algorithms/Applications

Smart Retail



Personalized Healthcare



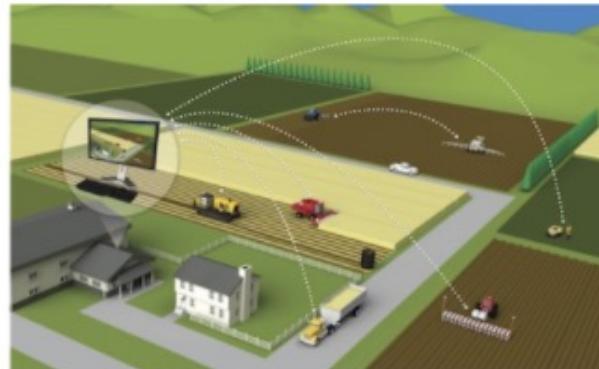
Smart Home



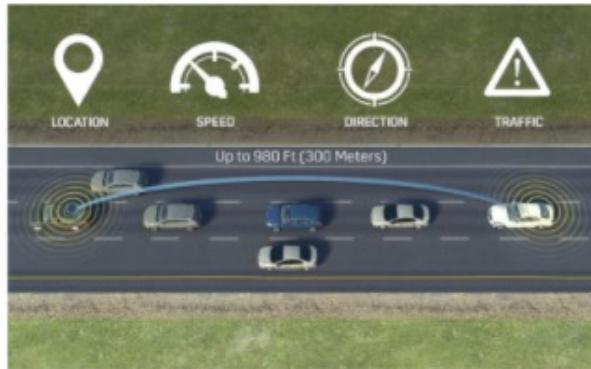
Smart Manufacturing



Precision Agriculture



Autonomous Driving



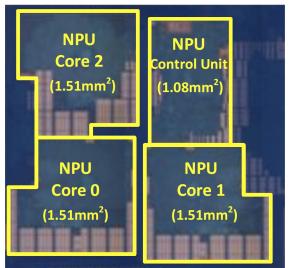
Courtesy: Song Han

Custom Chip Designs for AI

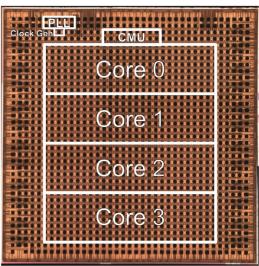
- To efficiently execute deep learning or deep neural network (DNN) workloads, many custom ASIC accelerators have been presented or are being designed.

Industry

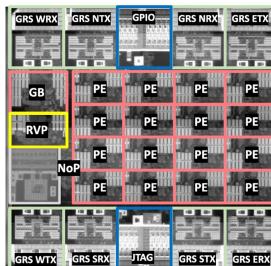
Samsung
ISSCC'21



IBM
ISSCC'21



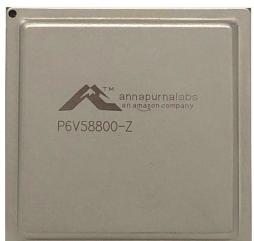
Nvidia
JSSC'20



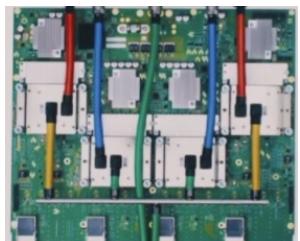
Tesla
FSD Chip



Amazon
Inferentia

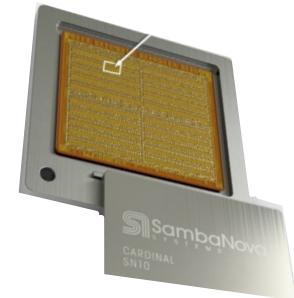


Google
TPUv4



Start-up

Sambanova



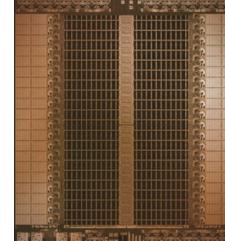
Graphcore
IPU



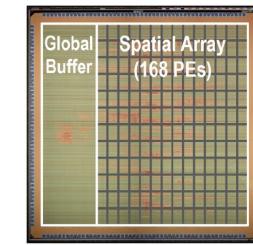
Mythic
M1108



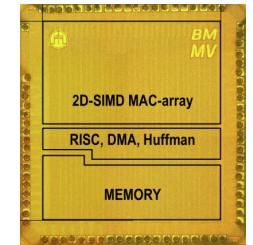
Groq



MIT
Eyeriss



KU Leuven
Envision

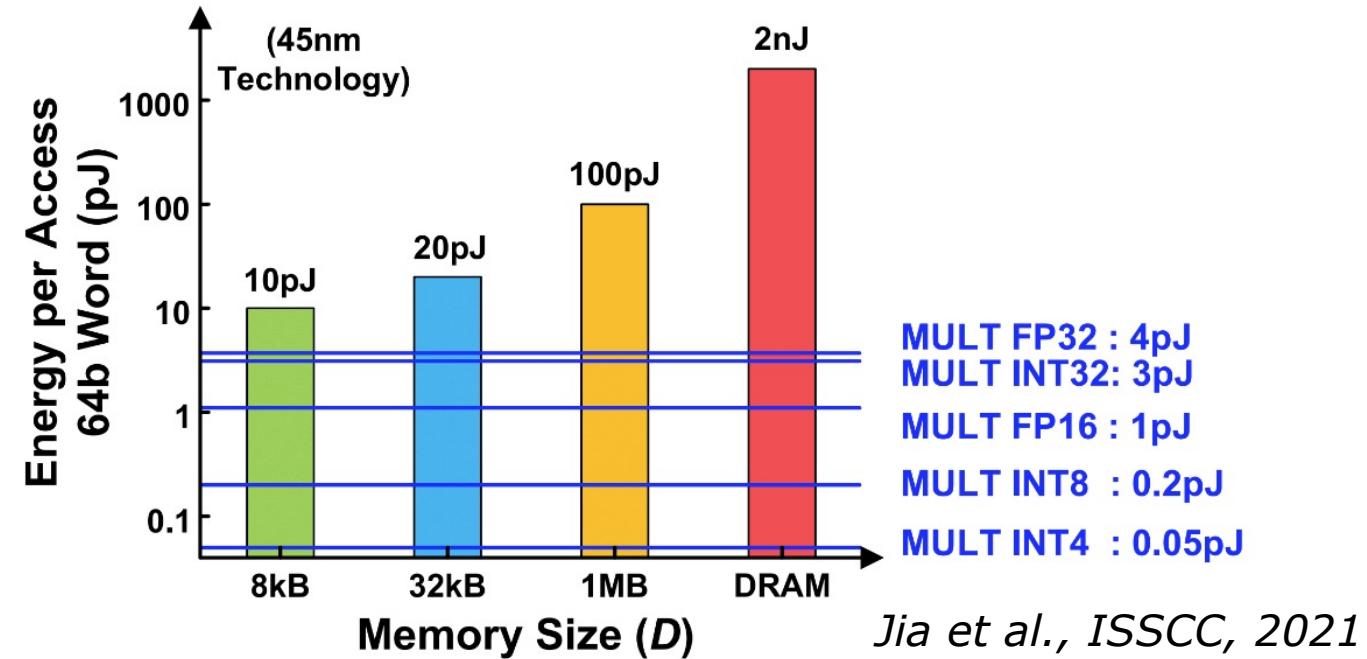
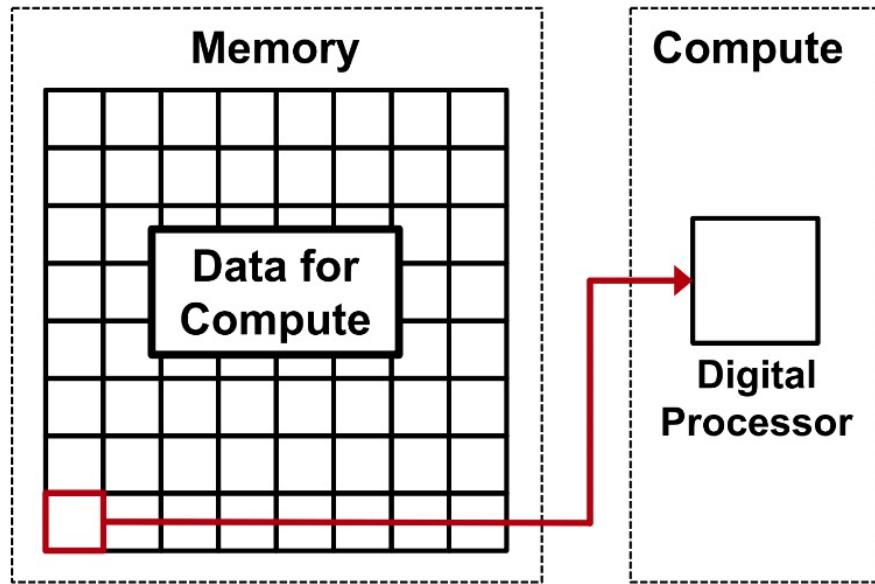


Tsinghua
Sticker-T



Data Movement in AI Inference Accelerators

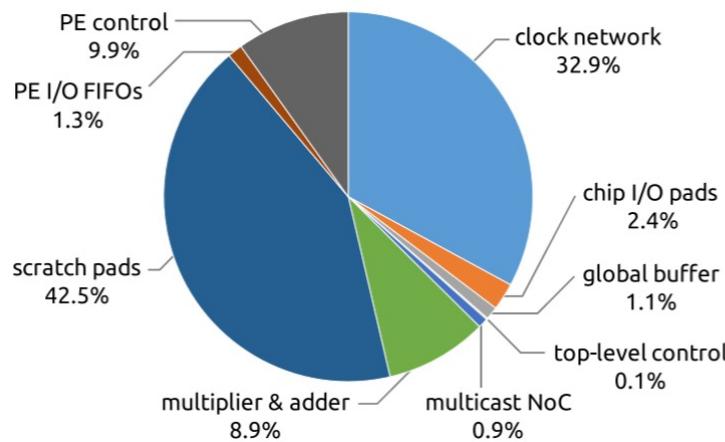
- AI or deep neural network (DNN) inference involves many (>90%) high-dimensional matrix-vector multiplies (MVMs)



Data accessing/movement costs can easily dominate energy/throughput

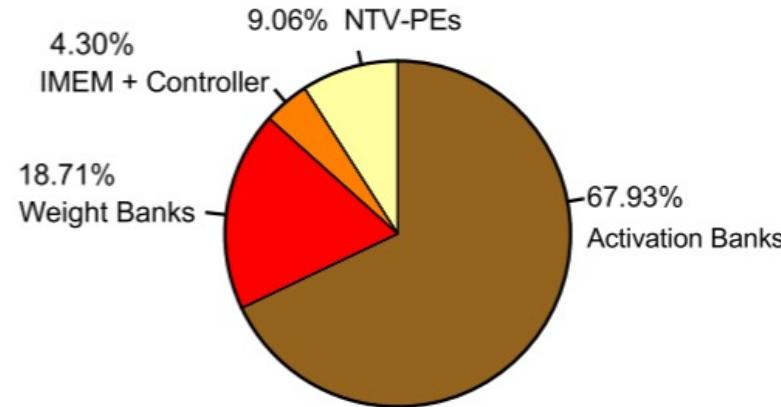
Bottleneck of All-Digital DNN HW Energy/Power

Eyeriss Chip Power for AlexNet



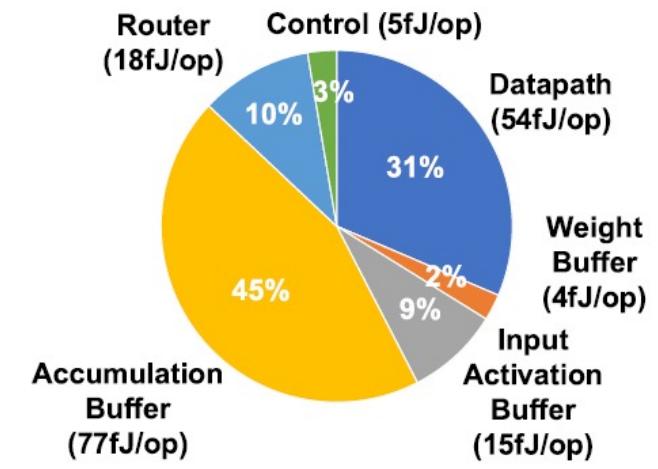
Chen et al., IEEE JSSC, 2017

EOS Chip Power for AlexNet



Sim et al., IEEE TVLSI, 2020

MCM chip PE Energy



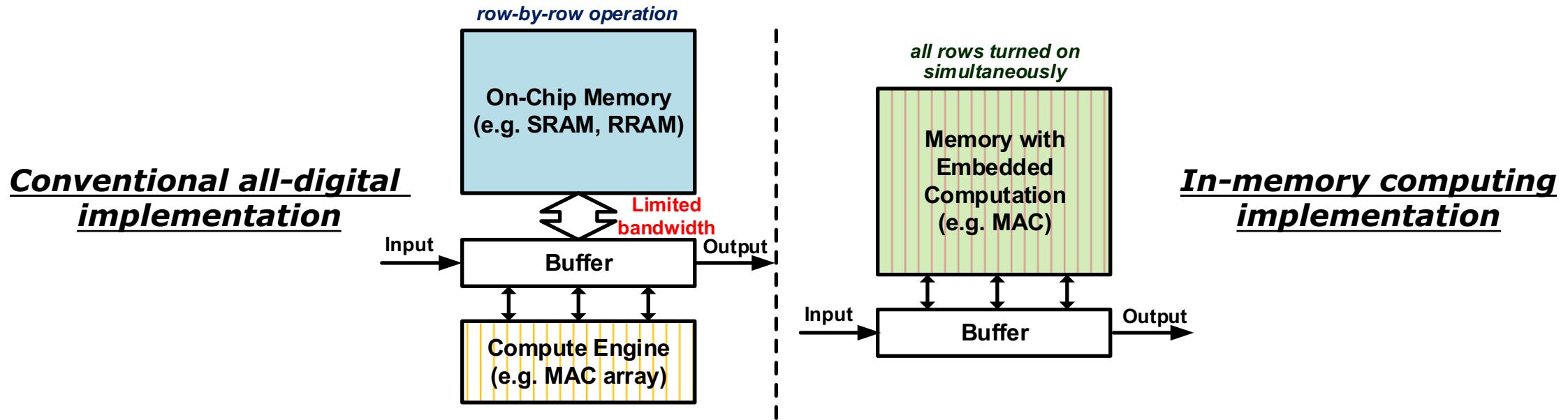
Zimmer et al., IEEE JSSC, 2020

- To perform computation, we need to fetch data from memory
→ High energy consumption
- A large number of memory accesses
→ Dominant portion of total energy consumption

Outline

- Background & Motivation
- **IMC Macro Design and Challenges of IMC Systems**
- Programmable IMC Accelerator (PIMCA)
 - Instruction Set Architecture
 - Activation Storage & Access
 - SIMD Design
 - PE Mapping
- Measurement & Comparison
- Summary

In-Memory Computing for DNNs



- Embedding computation inside memory (e.g. bitline)
- Large reduction in data transfer
- Assert multiple/all rows together → increased parallelism
- ✓ Large potential for energy-efficiency
- ✗ Key challenge: noise/variability, DNN accuracy loss (?), programmability

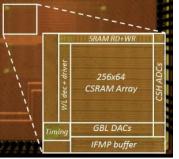
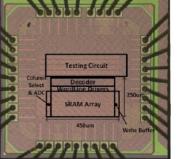
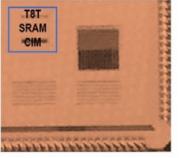
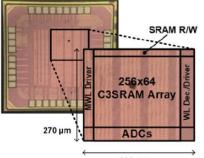
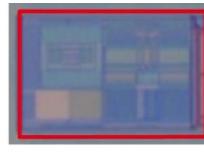
Memory Technologies for IMC

Memory device	Advantages	Challenges
SRAM	<ul style="list-style-type: none">Tapeout-able in latest CMOS technologyHigh on/off ratioReliable, large-scale integration	<ul style="list-style-type: none">Push-rule bitcell not openly availableRelatively large area
DRAM	<ul style="list-style-type: none">Can significantly reduce DRAM communication	<ul style="list-style-type: none">Read is destructiveModification of bitcell & array is very difficult
eNVM (e.g. RRAM, PCM)	<ul style="list-style-type: none">Non-volatileHigh density	<ul style="list-style-type: none">Low on/off ratioLimited CMOS nodes, large-scale integrationLarge peripheral circuits

- Here we focus on SRAM-based IMC designs, since it is most robust and viable for large-scale integration in any CMOS node

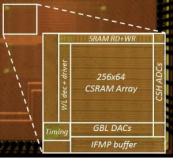
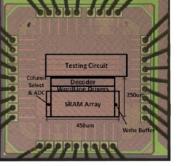
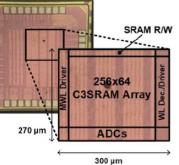
SRAM IMC Macro Designs

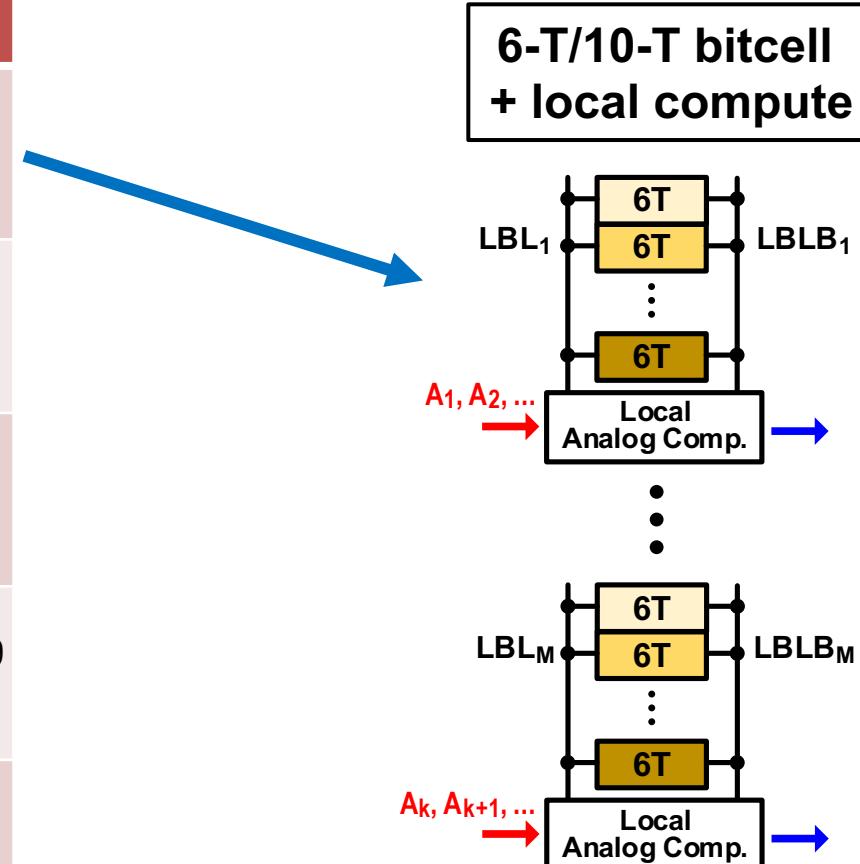
Single IMC Macro Designs

IMC Macro	Size	Publication
 MIT “CONV-SRAM”	256x64	ISSCC, 2018 JSSC, 2019
 ASU / Columbia “XNOR-SRAM”	256x64	S. VLSI, 2018 JSSC, 2020
 NTHU “Twin-8T SRAM”	64x60	ISSCC, 2019 JSSC, 2020
 ASU / Columbia “C3SRAM”	256x64	ESSCIRC, 2019 JSSC, 2020
 TSMC • 7nm IMC	64x64	ISSCC, 2020 JSSC, 2021

SRAM IMC Macro Designs

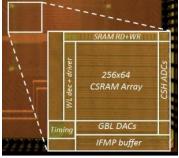
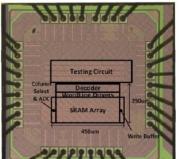
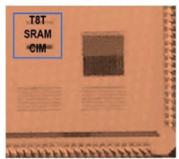
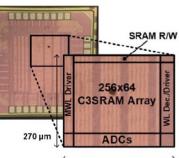
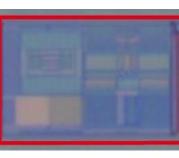
Single IMC Macro Designs

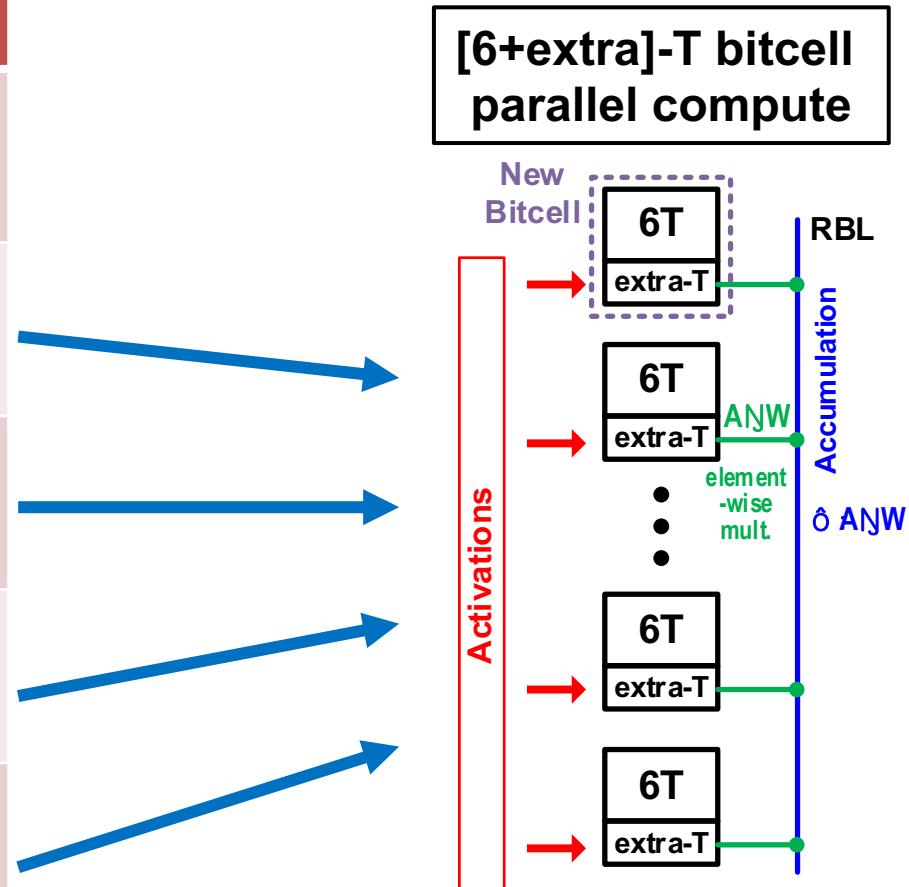
IMC Macro	Size	Publication
	256x64	ISSCC, 2018 JSSC, 2019
	256x64	S. VLSI, 2018 JSSC, 2020
	64x60	ISSCC, 2019 JSSC, 2020
	256x64	ESSCIRC, 2019 JSSC, 2020
	64x64	ISSCC, 2020 JSSC, 2021



SRAM IMC Macro Designs

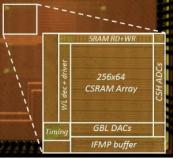
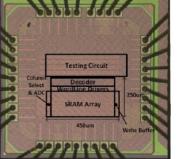
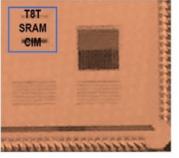
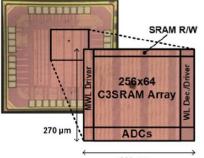
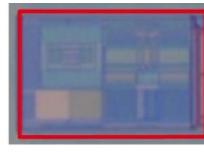
Single IMC Macro Designs

IMC Macro	Size	Publication
 MIT "CONV-SRAM"	256x64	ISSCC, 2018 JSSC, 2019
 ASU / Columbia "XNOR-SRAM"	256x64	S. VLSI, 2018 JSSC, 2020
 NTHU "Twin-8T SRAM"	64x60	ISSCC, 2019 JSSC, 2020
 ASU / Columbia "C3SRAM"	256x64	ESSCIRC, 2019 JSSC, 2020
 TSMC • 7nm IMC	64x64	ISSCC, 2020 JSSC, 2021

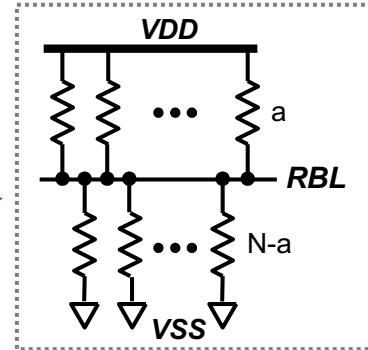


SRAM IMC Macro Designs

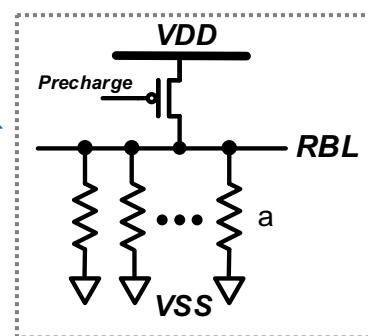
Single IMC Macro Designs

IMC Macro	Size	Publication
	256x64	ISSCC, 2018 JSSC, 2019
	256x64	S. VLSI, 2018 JSSC, 2020
	64x60	ISSCC, 2019 JSSC, 2020
	256x64	ESSCIRC, 2019 JSSC, 2020
	64x64	ISSCC, 2020 JSSC, 2021

Resistive IMC



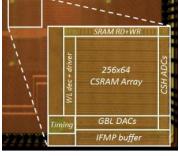
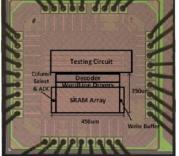
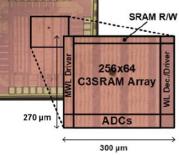
resistive divider



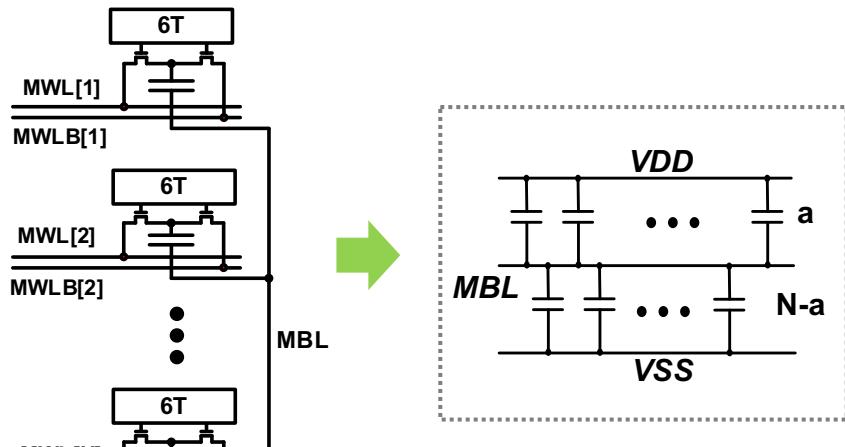
resistive pull-down
time-dependent

SRAM IMC Macro Designs

Single IMC Macro Designs

IMC Macro	Size	Publication
 MIT "CONV-SRAM" 256x64 ISSCC, 2018 JSSC, 2019	256x64	ISSCC, 2018 JSSC, 2019
 ASU / Columbia "XNOR-SRAM" 256x64 S. VLSI, 2018 JSSC, 2020	256x64	S. VLSI, 2018 JSSC, 2020
 NTHU "Twin-8T SRAM" 64x60 ISSCC, 2019 JSSC, 2020	64x60	ISSCC, 2019 JSSC, 2020
 ASU / Columbia "C3SRAM" 256x64 ESSCIRC, 2019 JSSC, 2020	256x64	ESSCIRC, 2019 JSSC, 2020
 TSMC • 7nm IMC 64x64 ISSCC, 2020 JSSC, 2021	64x64	ISSCC, 2020 JSSC, 2021

Capacitive IMC



capacitive divider

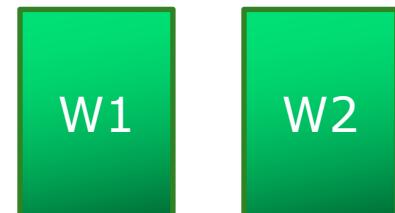
Challenges of SRAM IMC Accelerators

- Limited IMC capacity
 - **Lower throughput/parallelism**
 - Require frequent weight update
 - Limited data reuse opportunity

Smaller Capacity

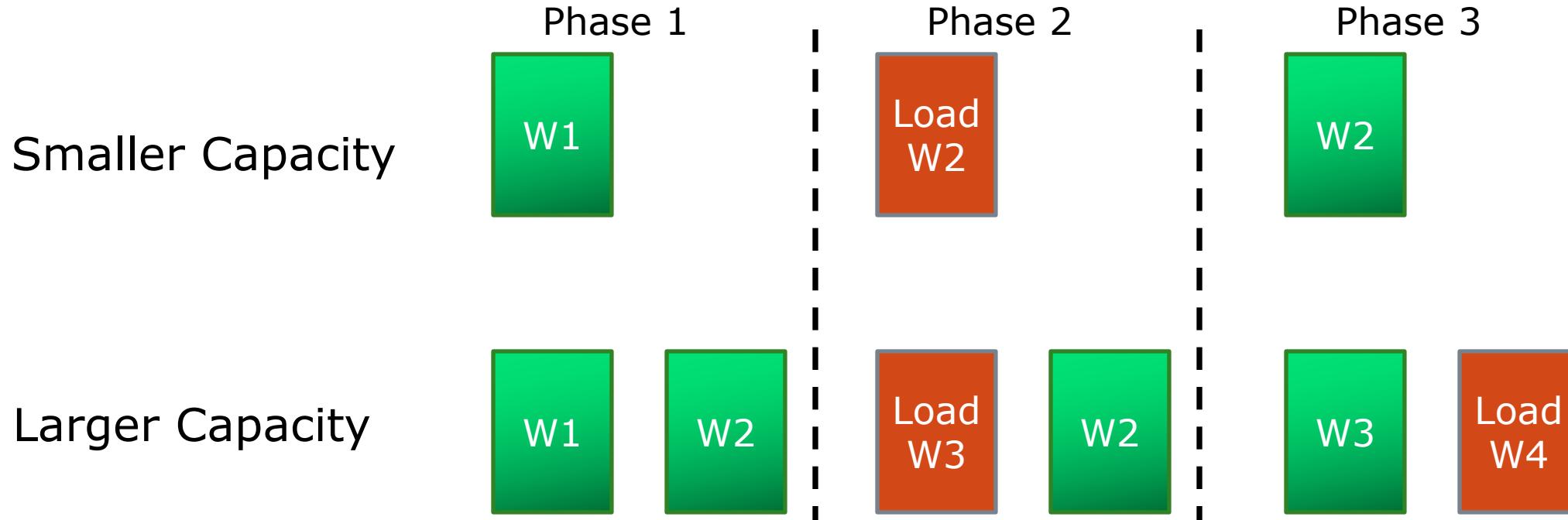


Larger Capacity



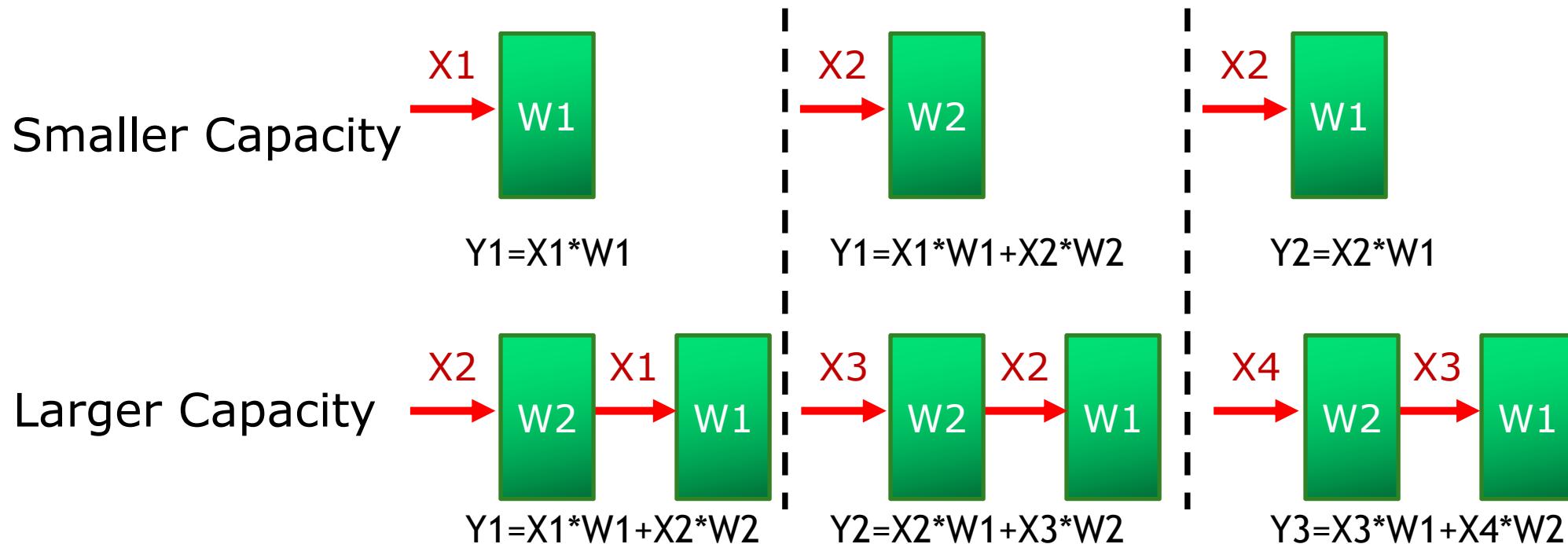
Challenges of SRAM IMC Accelerators

- Limited IMC capacity
 - Lower throughput/parallelism
 - **Require frequent weight update**
 - Limited data reuse opportunity



Challenges of SRAM IMC Accelerators

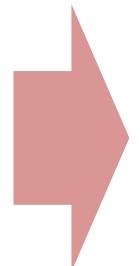
- Limited IMC capacity
 - Lower throughput/parallelism
 - Require frequent weight update
 - **Limited data reuse opportunity**



IMC Macro → IMC System Designs

Single IMC Macro Designs

IMC Macro	Size	Publication
MIT “CONV-SRAM”	256x64	ISSCC, 2018 JSSC, 2019
ASU / Columbia “XNOR-SRAM”	256x64	S. VLSI, 2018 JSSC, 2020
NTHU “Twin-8T SRAM”	64x60	ISSCC, 2019 JSSC, 2020
ASU / Columbia “C3SRAM”	256x64	ESSCIRC, 2019 JSSC, 2020
TSMC • 7nm IMC	64x64	ISSCC, 2020 JSSC, 2021

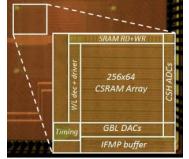
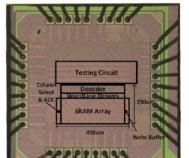
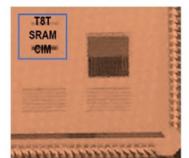
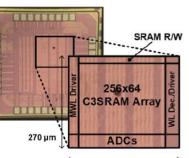


IMC System Designs

IMC System	# of macros (macro size)	Total IMC SRAM size	Publication
Tsinghua / NTHU “Thinker-IM”	16 macros (64x64)	64 kb	S. VLSI, 2019
Tsinghua / NTHU • sparsity-aware IMC	4 macros (64x64)	16 kb	ISSCC, 2020
Princeton • heterogeneous processor	16 macros (576x64)	576 kb	Hotchips, 2020 JSSC, 2020
Princeton • programmable, scalable IMC NN acc.	16 macros (1152x256)	4.5 Mb	ISSCC, 2021
ASU / Columbia “PIMCA”	108 macros (256x128)	3.4 Mb	S. VLSI, 2021

IMC Macro → IMC System Designs

Single IMC Macro Designs

IMC Macro	Size	Publication
	256x64	ISSCC, 2018 JSSC, 2019
	256x64	S. VLSI, 2018 JSSC, 2020
	64x60	ISSCC, 2019 JSSC, 2020
	256x64	ESSCIRC, 2019 JSSC, 2020
	64x64	ISSCC, 2020 JSSC, 2021

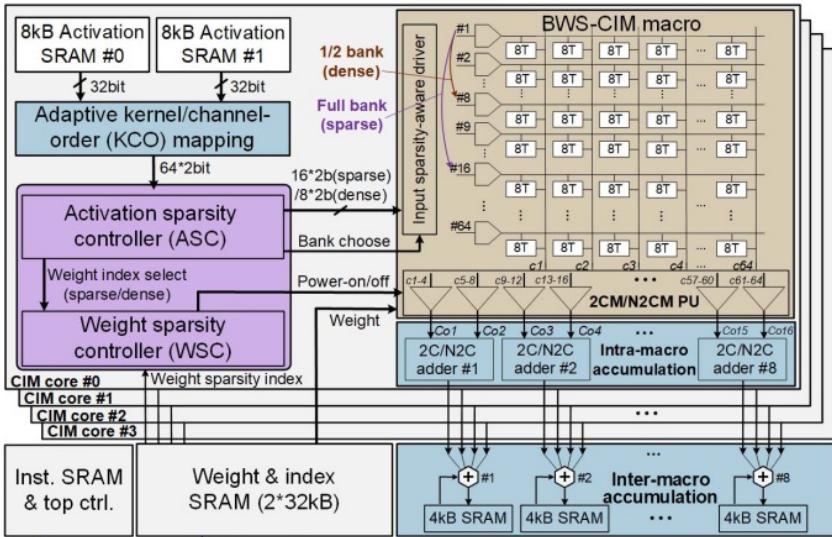


IMC System Designs

IMC System	# of macros (macro size)	Total IMC SRAM size	Publication
Tsinghua / NTHU "Thinker-IM"	16 macros (64x64)	64 kb	S. VLSI, 2019
Tsinghua / NTHU • sparsity-aware IMC	4 macros (64x64)	16 kb	ISSCC, 2020
Princeton • heterogeneous processor	16 macros (576x64)	576 kb	Hotchips, 2020 JSSC, 2020
Princeton • programmable, scalable IMC NN acc.	16 macros (1152x256)	4.5 Mb	ISSCC, 2021
ASU / Columbia "PIMCA"	108 macros (256x128)	3.4 Mb	S. VLSI, 2021

Limitations of Prior IMC Systems

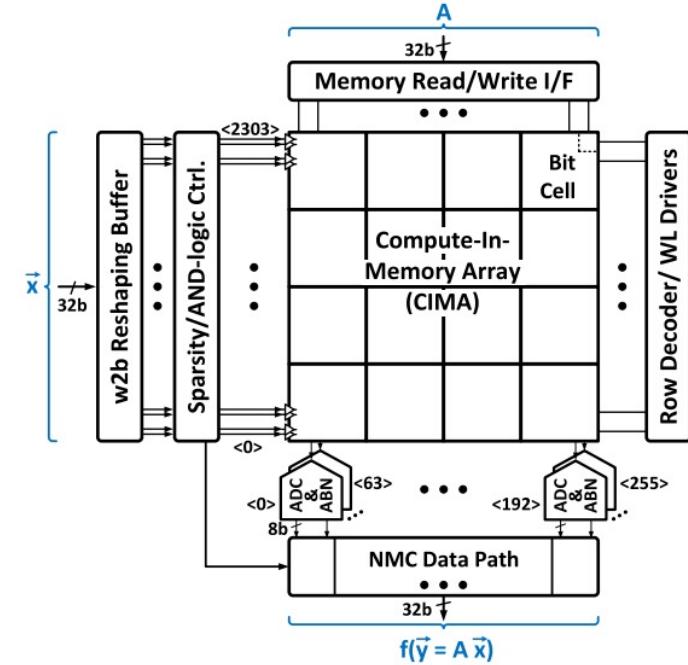
- Limited non-MAC operation support
 - Limited DNN layer types supported



Yue et al.,
ISSCC, 2020

Sparsity-aware, IMC SRAM capacity: 16 kb

Non-MAC operations not supported



Jia et al.,
JSSC, 2020

Programmable w/ RISC-V interface
IMC SRAM capacity: 576 kb

Other scalar operations such as pooling,
residual addition not supported

PIMCA: A 3.4-Mb Programmable In-Memory Computing Accelerator in 28nm for On-Chip DNN Inference

Shihui Yin¹, Bo Zhang², Minkyu Kim¹, Jyotishman Saikia¹,
Soonwan Kwon³, Sungmeen Myung³, Hyunsoo Kim³, Sang Joon Kim³,
Mingoo Seok², and Jae-sun Seo¹

¹ Arizona State University, USA

² Columbia University, USA

³ Samsung Advanced Institute of Technology, Korea

2021 Symposium on VLSI Circuits



Motivation of Proposed PIMCA

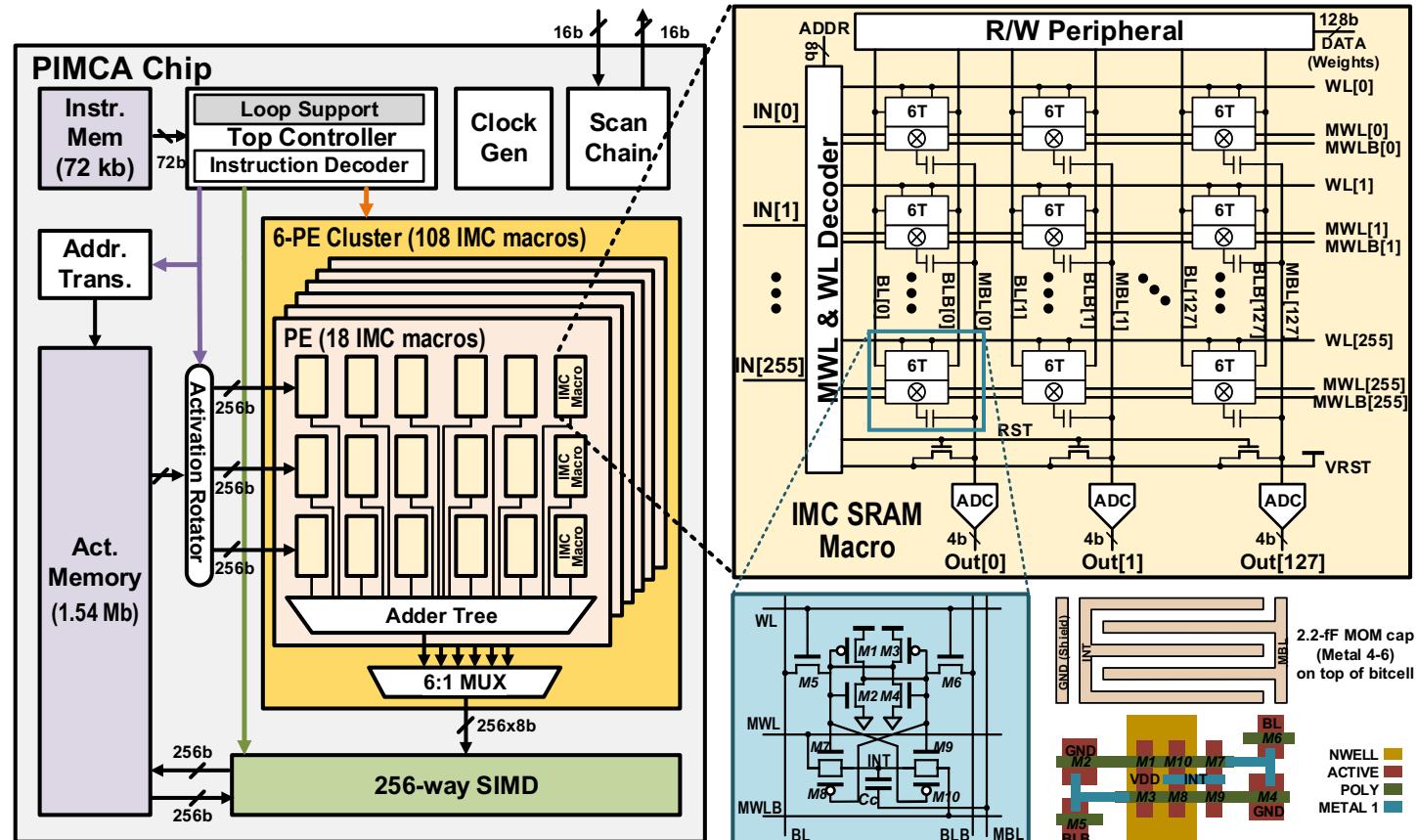
Programmable In-Memory Computing Accelerator

- Propose a programmable IMC architecture
- Custom ISA with hardware loop support control
 - Reduce the # of instructions for repetitive DNN workloads
- Programmable for different networks (VGG, ResNet, etc.), activation/weight precision (1b/2b) exploiting recent ML algorithms
- Integrate 100+ IMC SRAM macros, demonstrating one of the largest integrations for IMC systems
 - DNNs for edge devices can fully fit on-chip

Outline

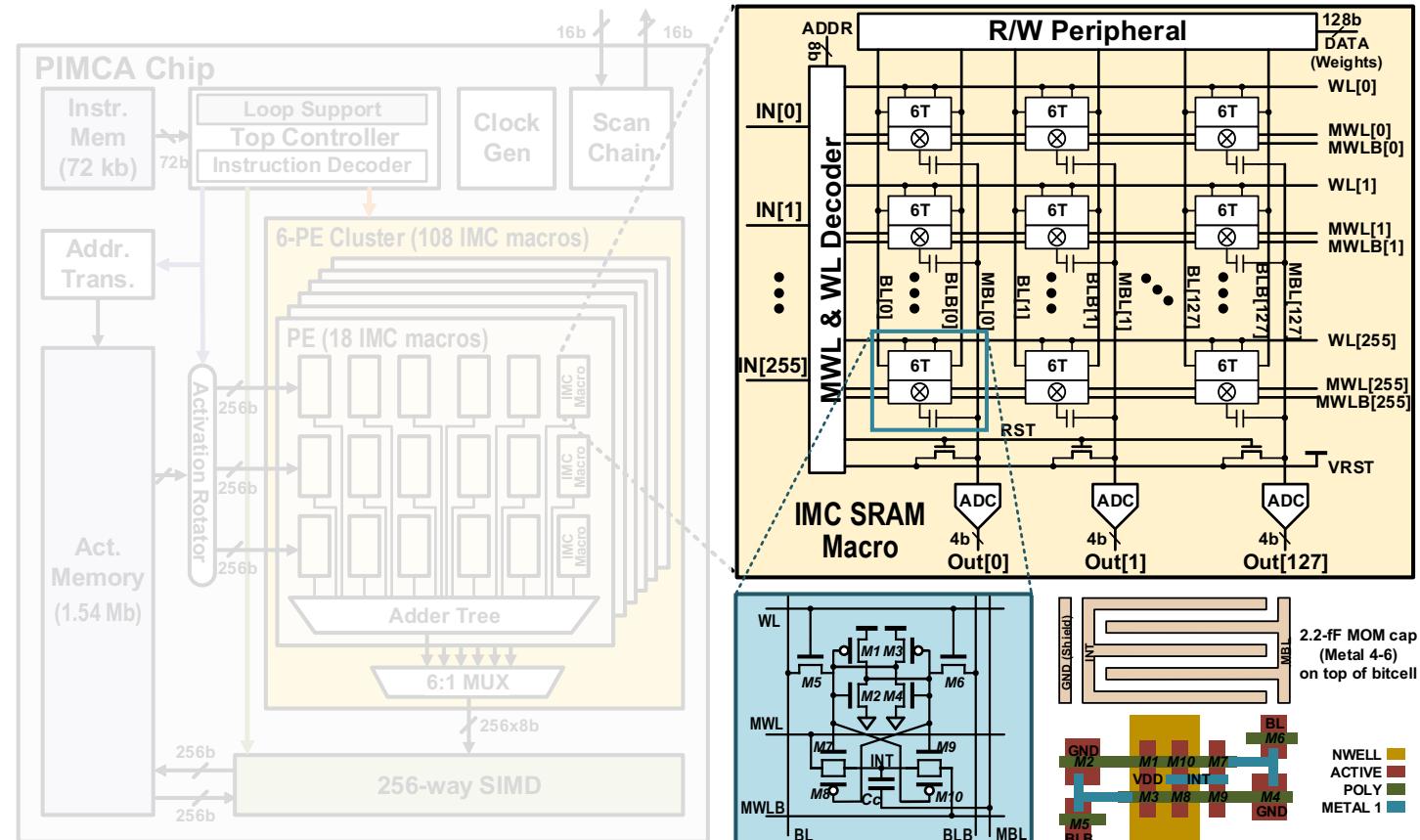
- Background & Motivation
- IMC Macro Design and Challenges of IMC Systems
- **Programmable IMC Accelerator (PIMCA)**
 - **Overall Architecture**
 - ISA & Loop Support Control
 - Activation Storage & Access
 - SIMD Design
 - PE Mapping
- Measurement & Comparison
- Summary

Overall PIMCA Architecture (1/3)



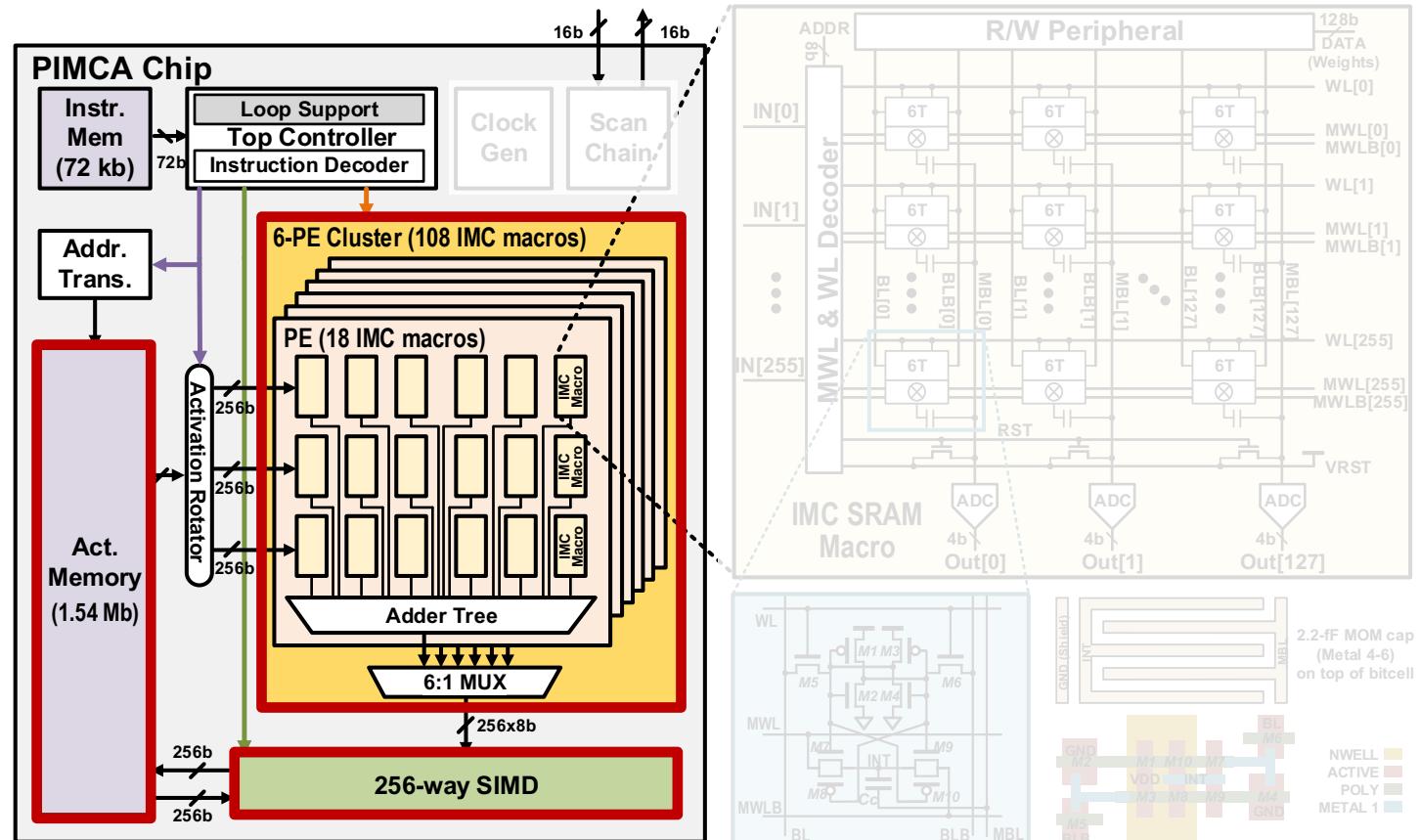
- 108 IMC 256-by-128 SRAM macros (3.4Mb) organized in 6 PEs
- 18 macros in a PE, whose outputs are summed by an adder tree

Overall PIMCA Architecture (2/3)



- 10T1C SRAM cell (2.2 fF MOM cap)
- 256x128 IMC SRAM array, 4-bit flash ADC for output

Overall PIMCA Architecture (3/3)

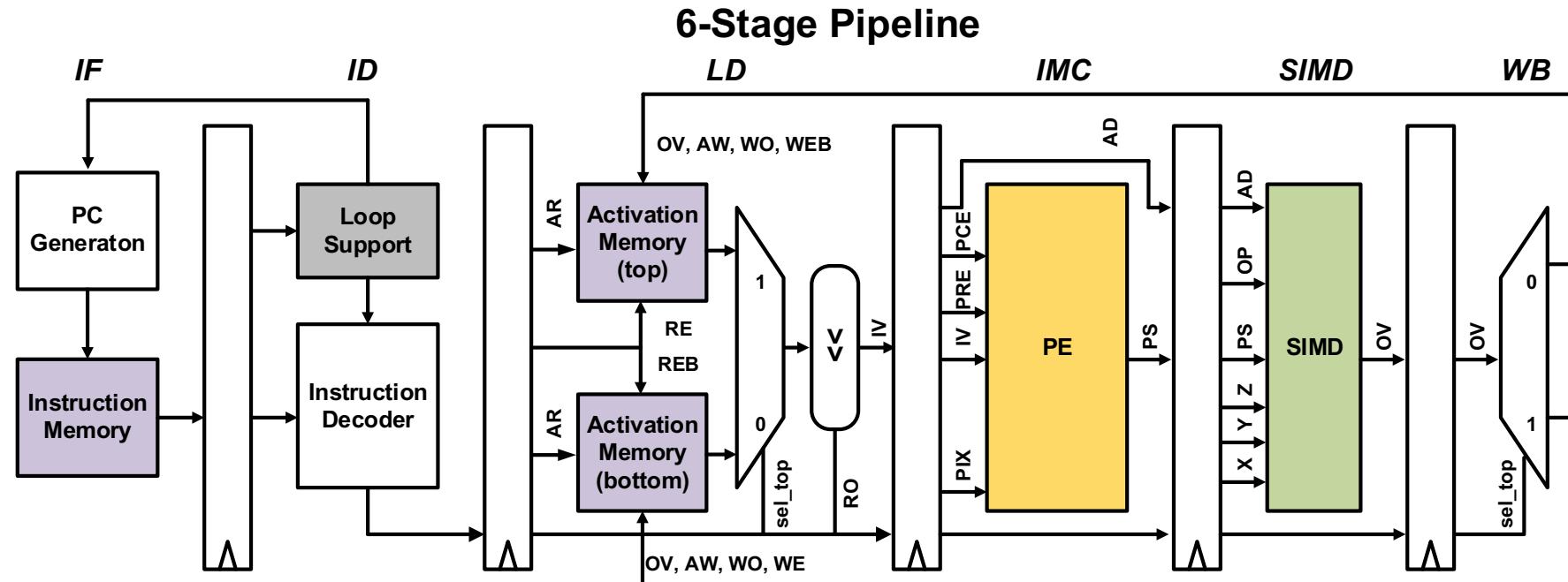


- Activation memory (1.54Mb), PE and SIMD are pipelined
- Top controller fetches/decodes instructions from instruction memory every cycle

Outline

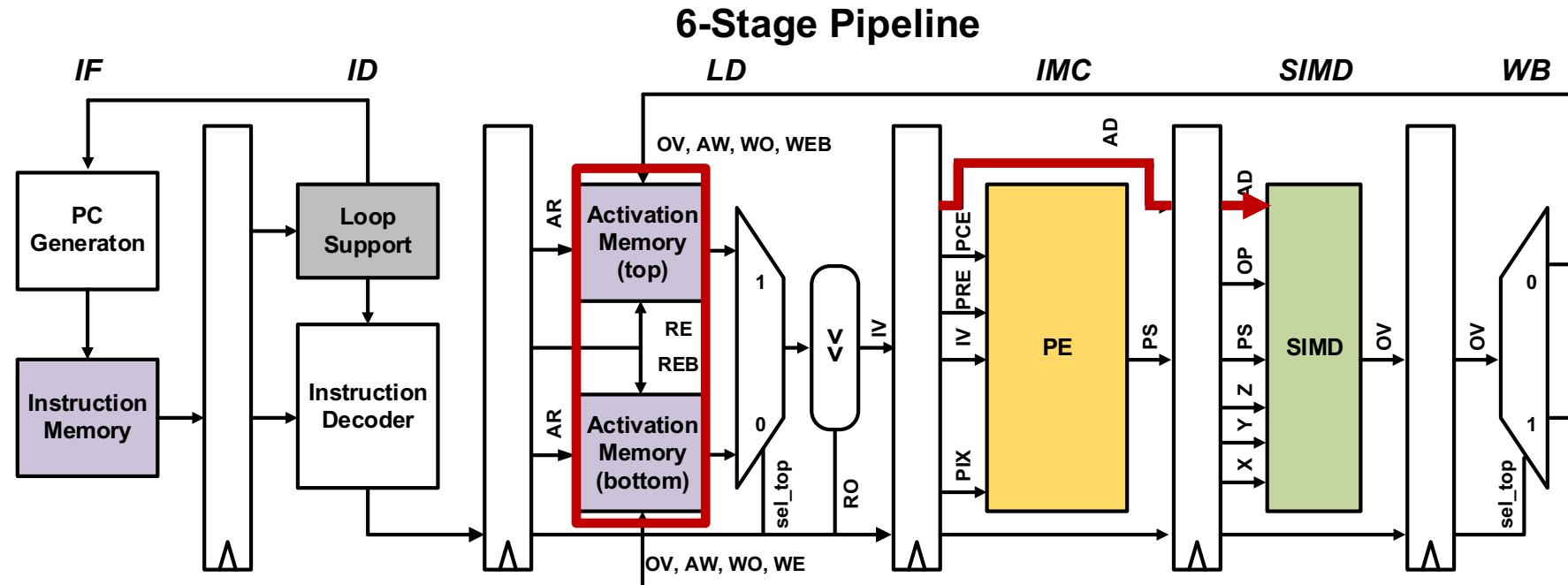
- Background & Motivation
- IMC Macro Design and Challenges of IMC Systems
- **Programmable IMC Accelerator (PIMCA)**
 - Overall Architecture
 - **ISA & Loop Support Control**
 - Activation Storage & Access
 - SIMD Design
 - PE Mapping
- Measurement & Comparison
- Summary

PIMCA IMC+SIMD Pipeline (1/2)



- IF:** instruction fetch
- ID:** instruction decode
- LD:** load input activation
- IMC:** in-memory computing & partial sum accumulation
- SIMD:** 256-way SIMD operation
- WB:** write back

PIMCA IMC+SIMD Pipeline (2/2)



IV: input vector, AD: data directly from activation memory, OV: output vector from SIMD, PS: partial sums from PE

- Activation memory organized in two groups of single-port SRAMs (top/bottom); While one group is used for reading, the other group is used for writing.
- Note PE is bypass-able and SIMD can process data from activation memory directly

Instruction Fields

PE+SIMD Instruction Fields

	Field Name	Width	Explanation
Loop	RP	6b	Repetition times of current instr.
Act. Mem. Access	AR/AW	10/10b	Read/write address
	RE/WE	6/1b	Read/write enable
	Sel_top	1b	Read from top group if true
	RO/WO	3b/2b	Read/write bank mapping order
PE	PIX	3b	PE select index
	PRE/PCE	3/6b	Macro array row/column enable
	PSK	1b	Skip PE operation if 1
	C5x5	1b	5x5 conv. mode if 1
	MSB	1b	Feeding MSB/LSB of act. if 1/0
SIMD	OP	3b	Operation code
	X/Y	3/3b	Operand X and Y
	Z	3b	Destination Z of result

- Instruction width = 72-bit
- A regular PE+SIMD instruction configures the data location for R/W, IMC PE operation mode, SIMD instruction.

Loops in DNNs

- Loops exist in DNN inference computation, which could be exploited to reduce the size of instructions needed.

E.g., in a 2D conv. layer

$$I(N_I, X_I, Y_I) \times W(N_O, N_I, X_w, Y_w) = O(N_O, X_O, Y_O)$$

for n_O in range(N_O): Loop-6

 for x_O in range(X_O): Loop-5

 for y_O in range(Y_O): Loop-4

 for n_I in range(N_I): Loop-3

 for x_w in range(X_w): Loop-2

 for y_w in range(Y_w): Loop-1

$$O(n_O, x_O, y_O) += I(n_I, x_O+x_w, y_O+y_w) * W(n_O, n_I, x_w, y_w)$$

ISA w/ Loop Support Control (1/3)

- Two types of loop support in PIMCA top controller
 - A) Repeat a single instruction for 'RP' times (addresses automatically increase in each repetition)

PE+SIMD Instruction Fields

	Field Name	Width	Explanation
Loop	RP	6b	Repetition times of current instr.
Act. Mem. Access	AR/AW	10/10b	Read/write address
	RE/WE	6/1b	Read/write enable
	Sel_top	1b	Read from top group if true
	RO/WO	3b/2b	Read/write bank mapping order
PE	PIX	3b	PE select index
	PRE/PCE	3/6b	Macro array row/column enable
	PSK	1b	Skip PE operation if 1
	C5x5	1b	5x5 conv. mode if 1
	MSB	1b	Feeding MSB/LSB of act. if 1/0
SIMD	OP	3b	Operation code
	X/Y	3/3b	Operand X and Y
	Z	3b	Destination Z of result

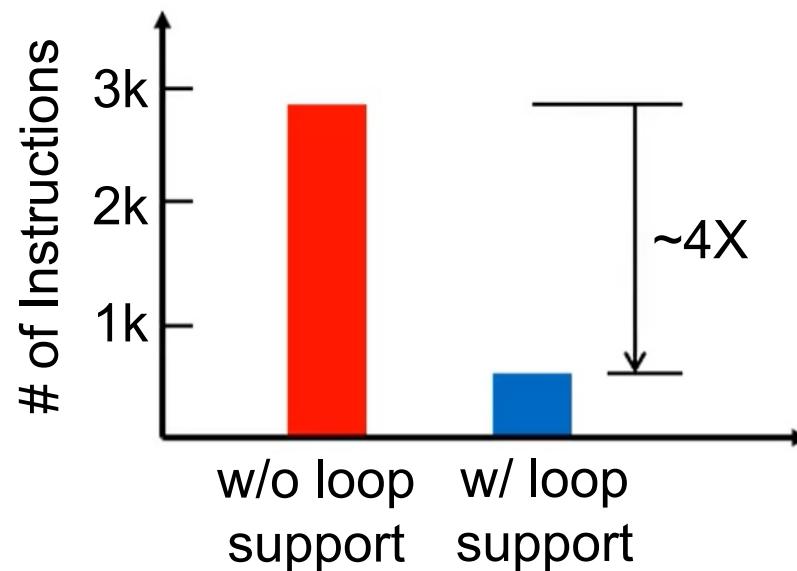
ISA w/ Loop Support Control (2/3)

- Two types of loop support in PIMCA top controller
 - A) Repeat a single instruction for 'RP' times (addresses automatically increase in each repetition)
 - B) Generic nested loop definitions by special loop instructions w/ dedicated counters and registers

Register File (16x10b)		Loop Support Registers		Loop Instructions	
LR[0:7]	w/ HW loop support	STR[0:7]	Loop step sizes (9b)	LS: LIX(3b), LIN(10b), LST(9b), LRP(6b)	Loop setup for LIX: LR[LIX] \leftarrow LIN; STR[LIX] \leftarrow LST; RPR[LIX] \leftarrow LRP; CTR[LIX] \leftarrow 0
LR[8:15]	w/o HW loop support	CTR[0:7]	Loop counters (6b)	LE: LIX(3b), LET(10b)	Loop end check for LIX: LR[LIX] += STR[LIX]; CTR[LIX] += 1; PC \leftarrow PC + 1 if CTR[LIX] == RPR[LIX] else PC \leftarrow loop entry address (LET)
		RPR[0:7]	Loop iterations (6b)	LA: LIXD(4b), LIXS(4b), LB(10b), LC (10b)	$LR[LIXD] \leftarrow LR[LIXS] \times LB + LC$

ISA w/ Loop Support Control (3/3)

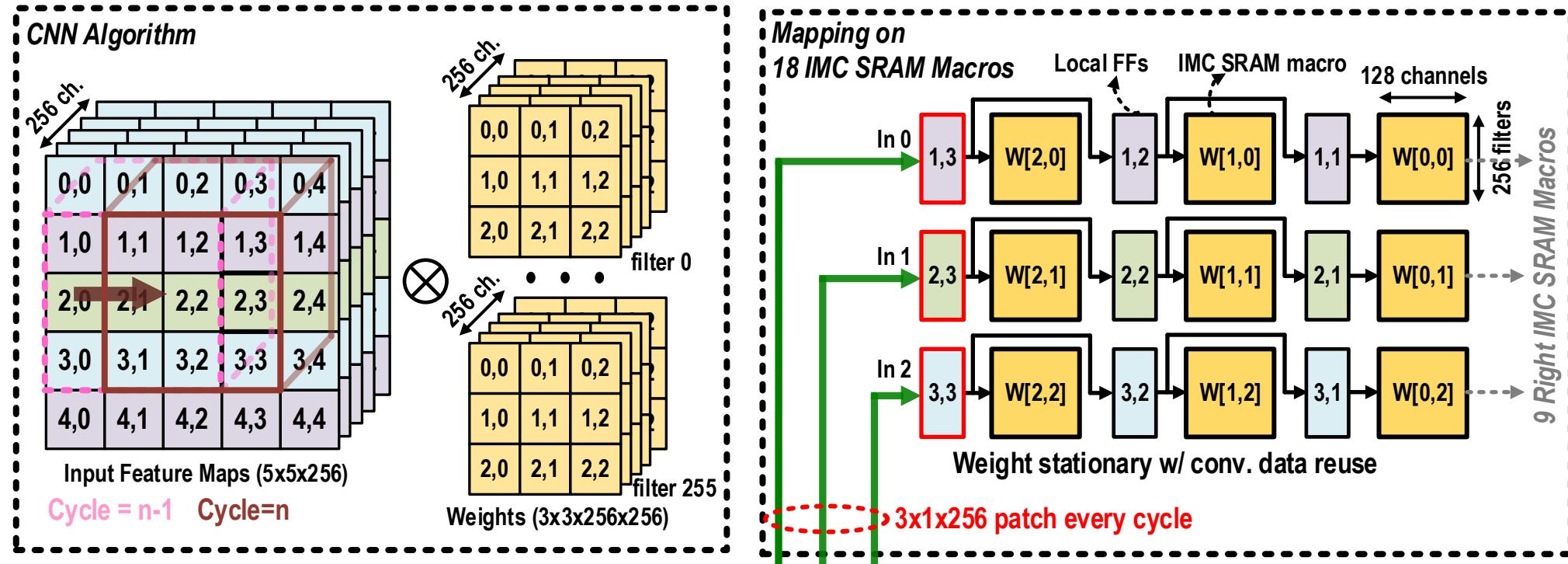
- Two types of loop support in PIMCA top controller
 - A) Repeat a single instruction for 'RP' times (addresses automatically increase in each repetition)
 - B) Generic nested loop definitions by special loop instructions w/ dedicated counters and registers
 - Instruction compression ratio: ~4X for VGG-9



Outline

- Background & Motivation
- IMC Macro Design and Challenges of IMC Systems
- **Programmable IMC Accelerator (PIMCA)**
 - Overall Architecture
 - ISA & Loop Support Control
 - **Activation Storage & Access**
 - **SIMD Design**
 - PE Mapping
- Measurement & Comparison
- Summary

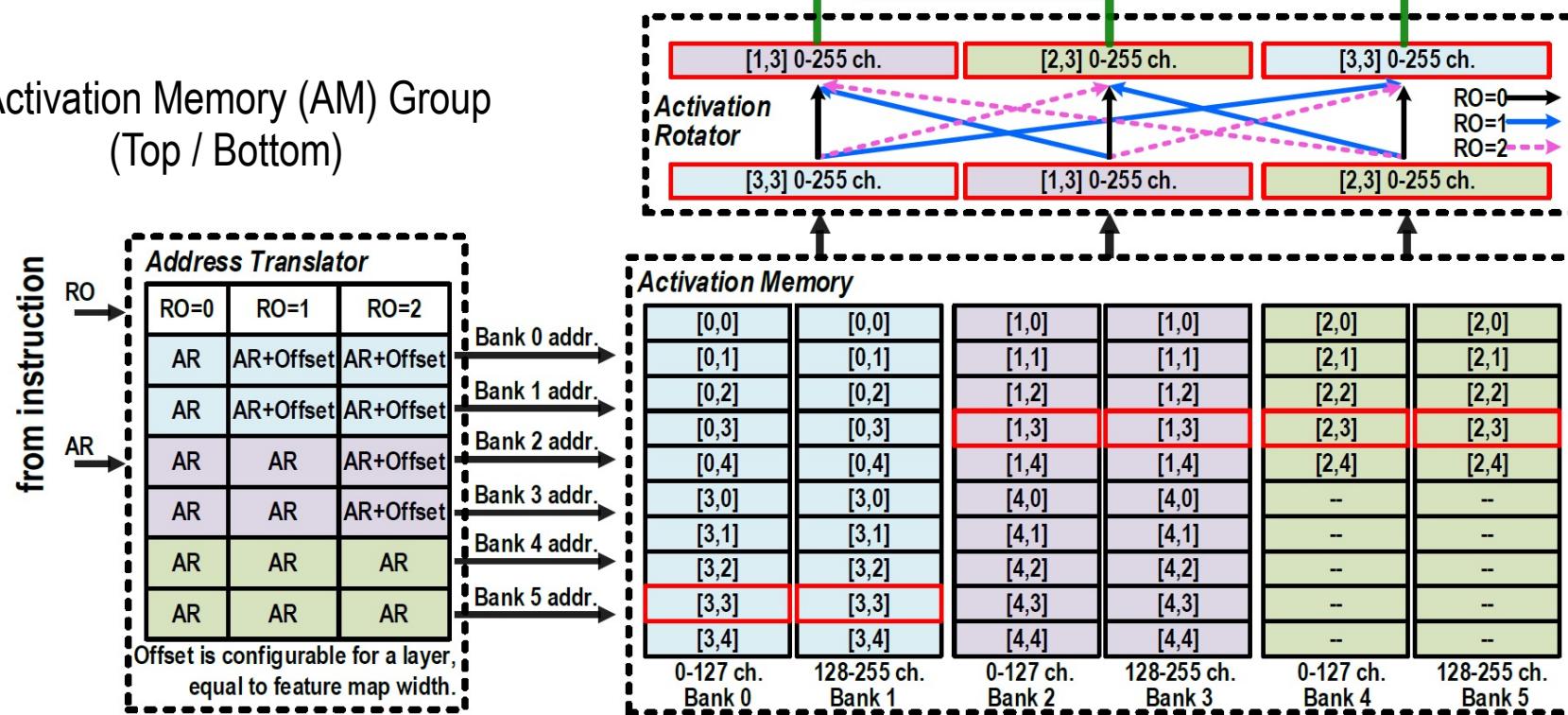
Activation Storage & Access (1/2)



- 3x3x256x256 conv. kernels are mapped onto 18 IMC SRAM macros based on kernel_x and kernel_y indices
- Shift registers are added for conv. data reuse
- 3x1x256 patch of input features are required to be fetched every cycle

Activation Storage & Access (2/2)

Each Activation Memory (AM) Group
(Top / Bottom)



- ❑ Each AM group has 6 banks to support flexible/parallel AM access
- ❑ PE can access any $3 \times 1 \times 256$ input patch from the 6 banks in one cycle
 - With proper address generation and activation rotation
 - Eliminates the need for extra buffering between AM and PE

SIMD Operations

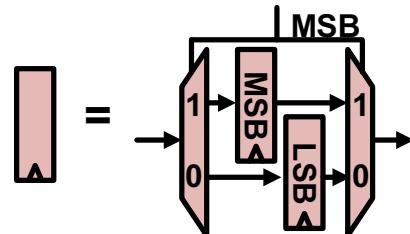
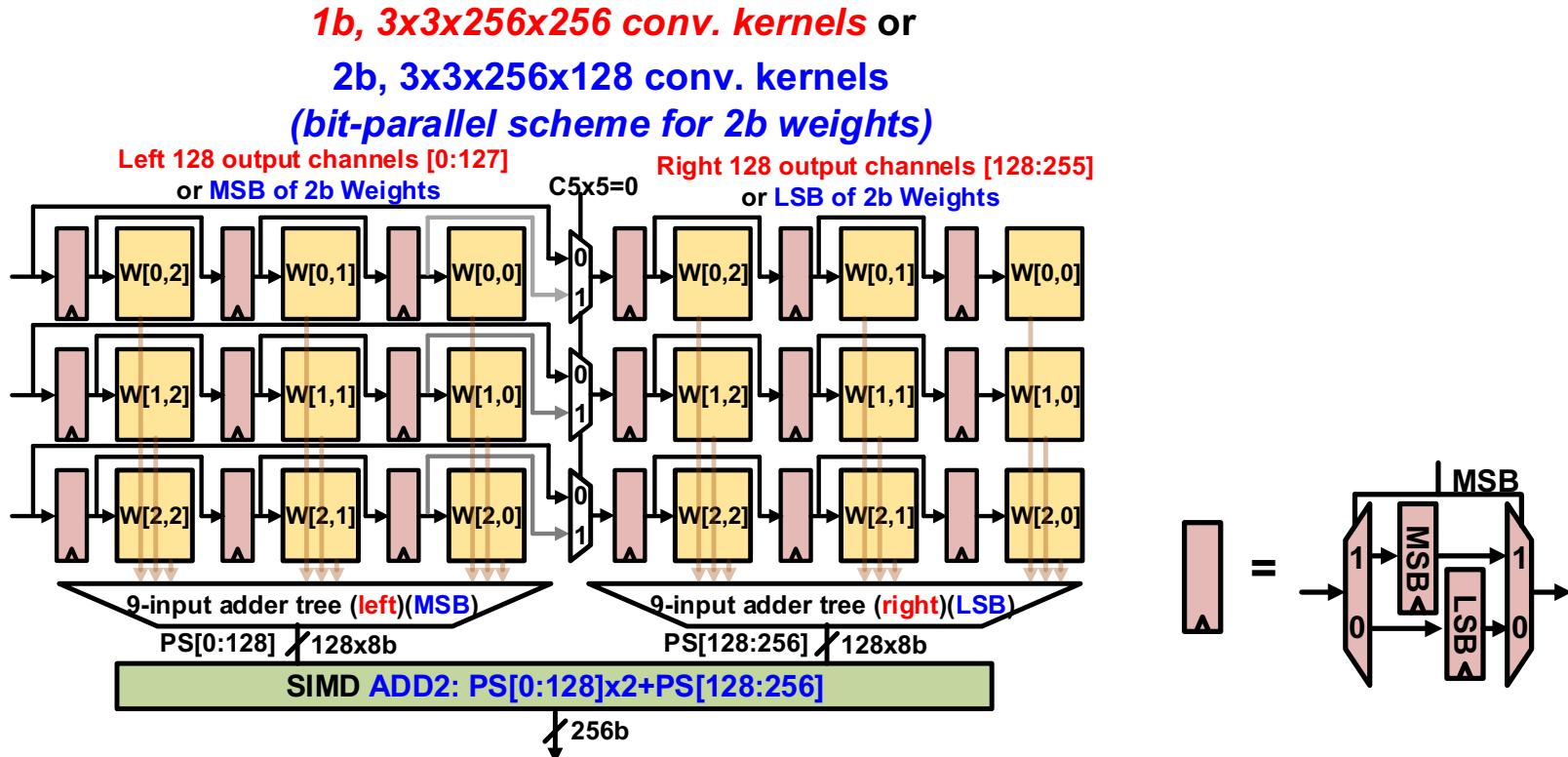
- 256-way SIMD
- 8 supported operations:
 - ADD: $X+Y \rightarrow Z$
 - ADD2: $X*2+Y \rightarrow Z$
 - COMP: $(X>Z) \rightarrow Z$ (for 1-bit activation)
 - CMP2: $(X>R0)+(X>R1)+(X>R2) \rightarrow Z$ (for 2-bit activation)
 - MAX: $\max(X, Y) \rightarrow Z$ (for max pooling)
 - RSHIFT: $(X >> Y) \rightarrow Z$
 - LSHIFT: $(X << 1) + 1\text{-b data_from_memory} \rightarrow X$
- ADD2 can support both bit-serial scheme for activation (X, Y from the same way) and bit-parallel scheme for weights (X, Y from left and right 128 ways)

Outline

- Background & Motivation
- IMC Macro Design and Challenges of IMC Systems
- **Programmable IMC Accelerator (PIMCA)**
 - Overall Architecture
 - ISA & Loop Support Control
 - Activation Storage & Access
 - SIMD Design
 - **PE Mapping**
- Measurement & Comparison
- Summary

PE Mapping Example (1/4)

- The 18 IMC SRAM macros are organized in left and right 3x3 arrays
- Left/right array assigned for 1b weights or MSB/LSB weights of 2b weights
- Shift-add happens in SIMD via ADD2 operations

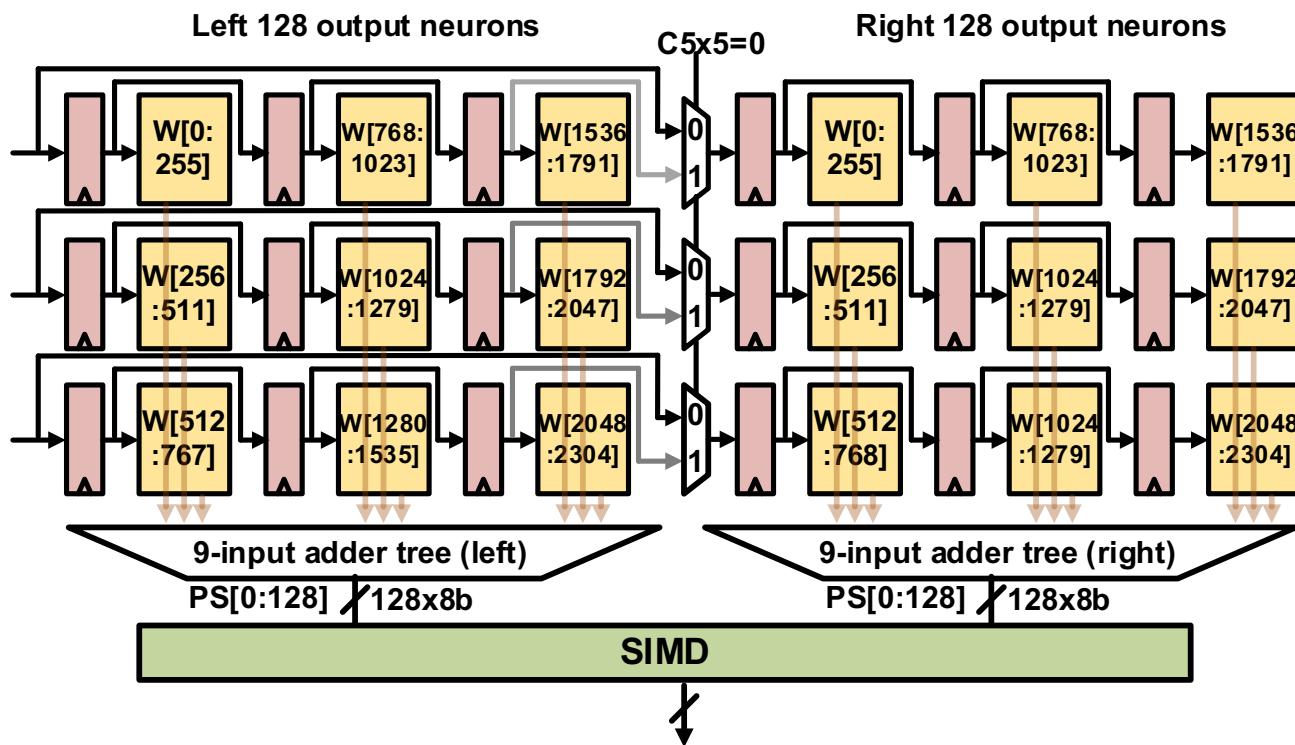


Separate shift registers for
MSB and LSB of 2b input
(bit-serial scheme)

PE Mapping Example (2/4)

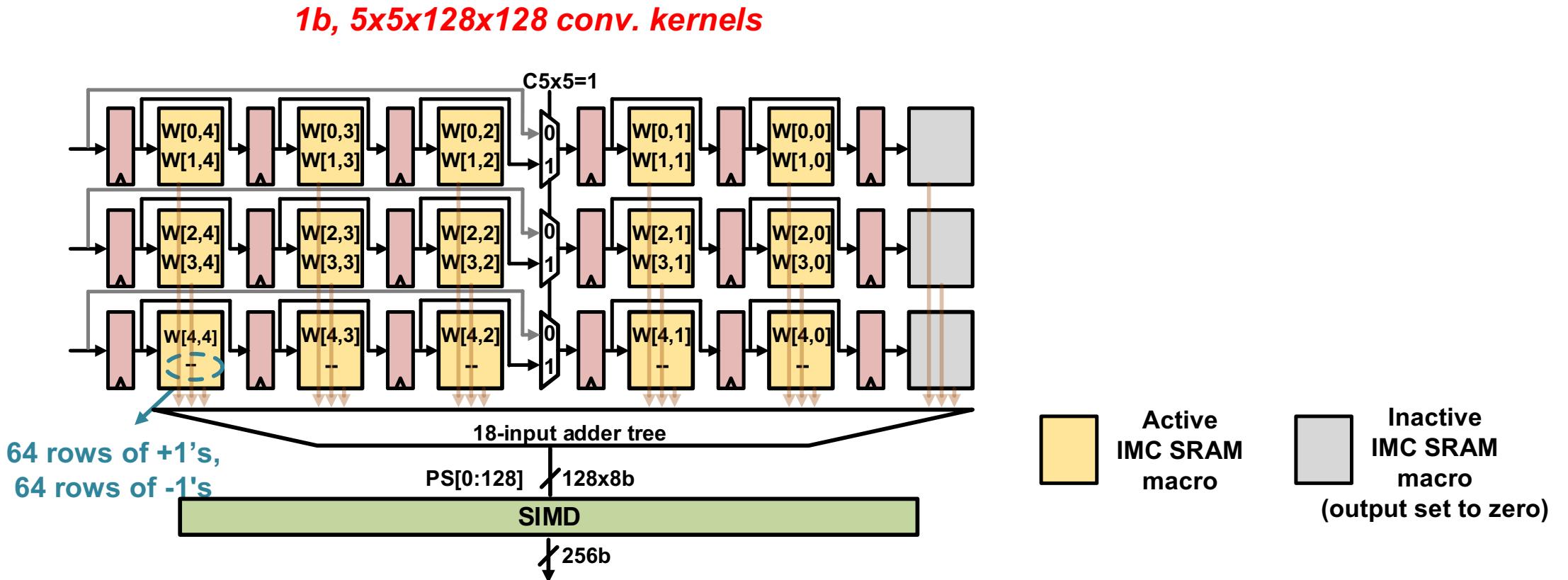
- A PE can support up to 2304x256 fully-connected matrix-vector multiplication

1b, 2304x256 fully connected layer



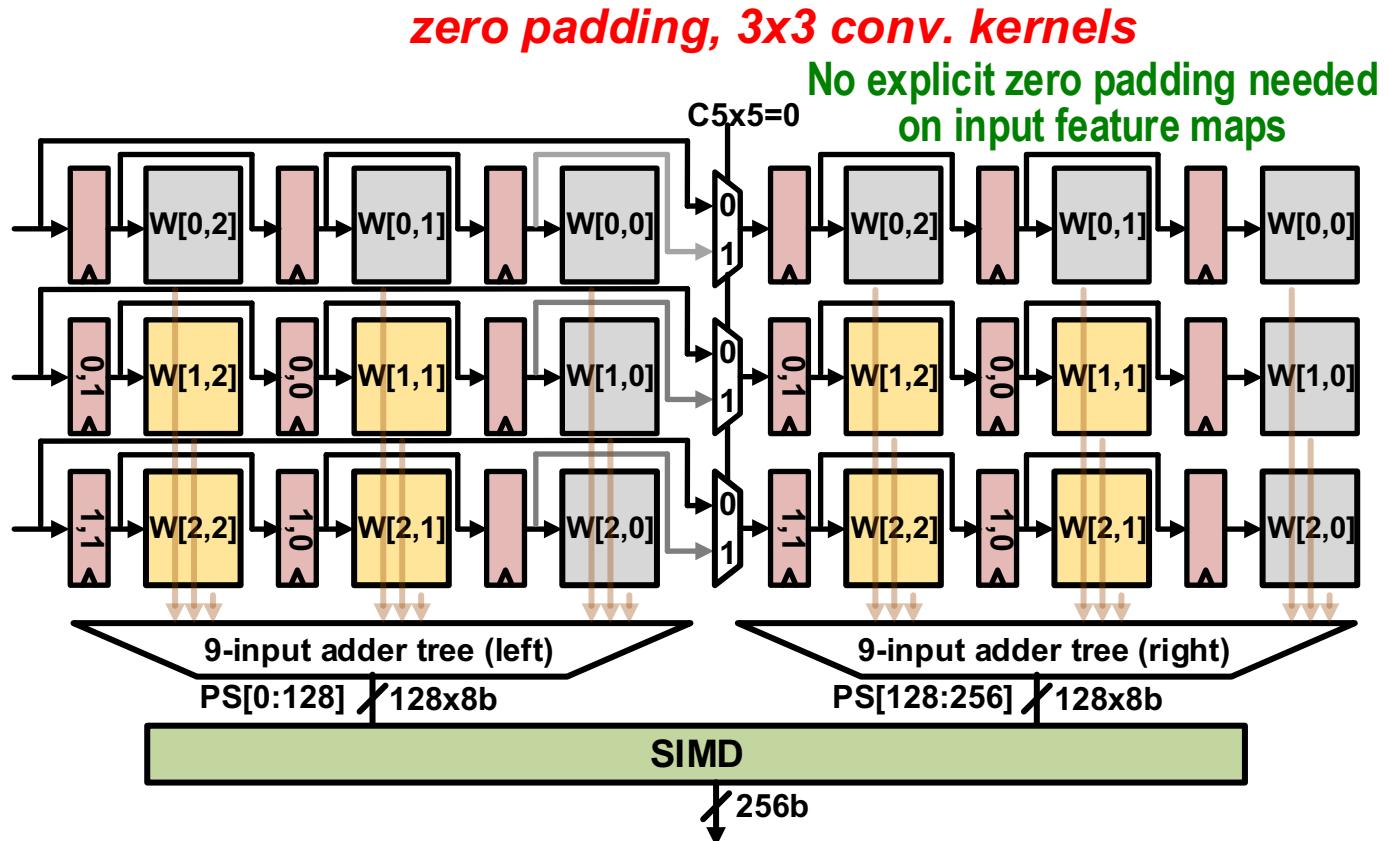
PE Mapping Example (3/4)

- 5x5 convolution can be supported by enabling the left 5 columns of IMC SRAM macros and turning adder tree to 18-input mode.

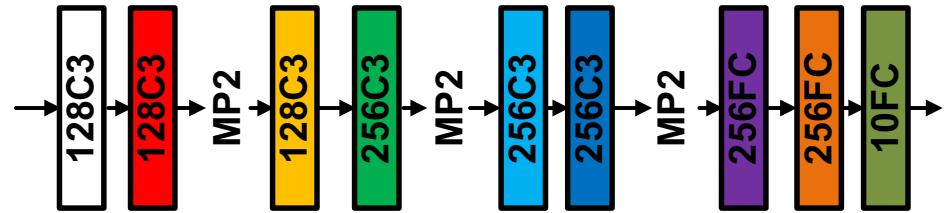


PE Mapping Example (4/4)

- IMC macros disabled when input vector is all-zero
- No explicit zero padding needed on input feature maps



PIMCA Chip Mapping for VGG-9



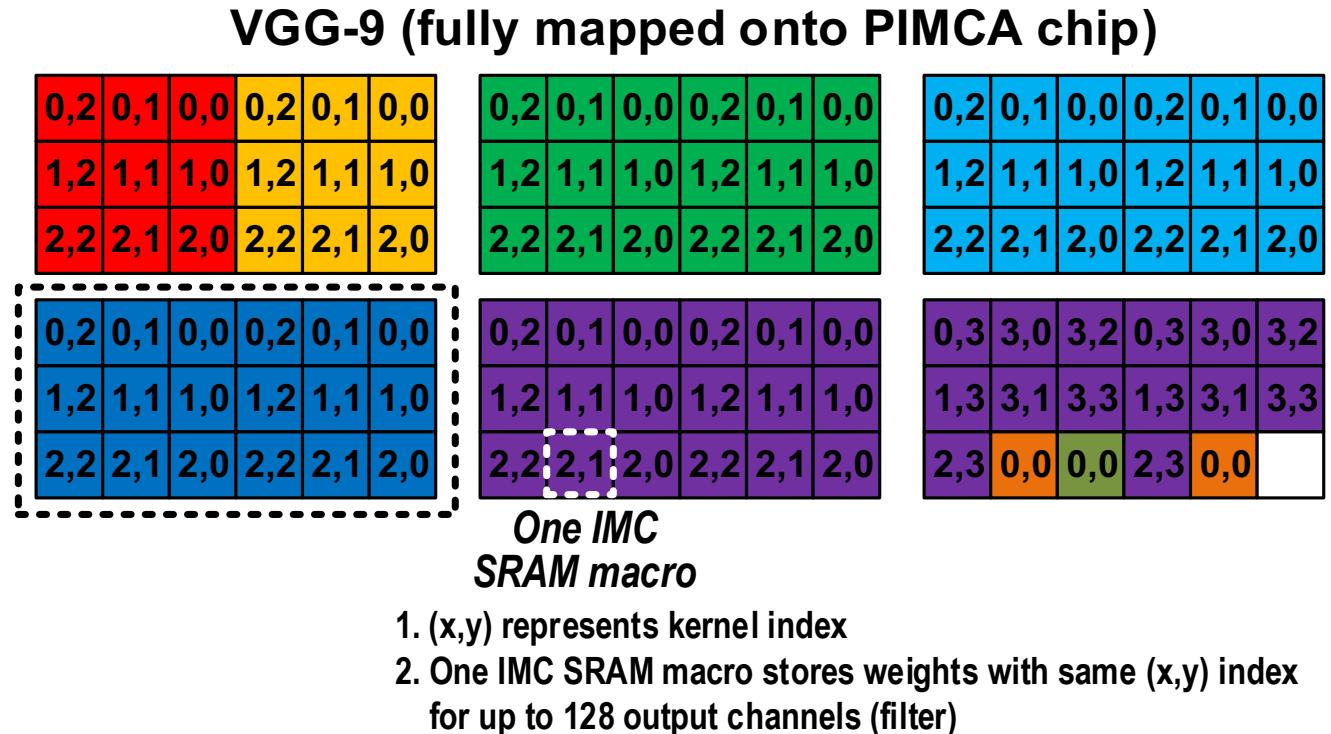
nC3: n 3x3 convolution filters (stride = 1)

nC3S2: n 3x3 convolution filters (stride = 2)

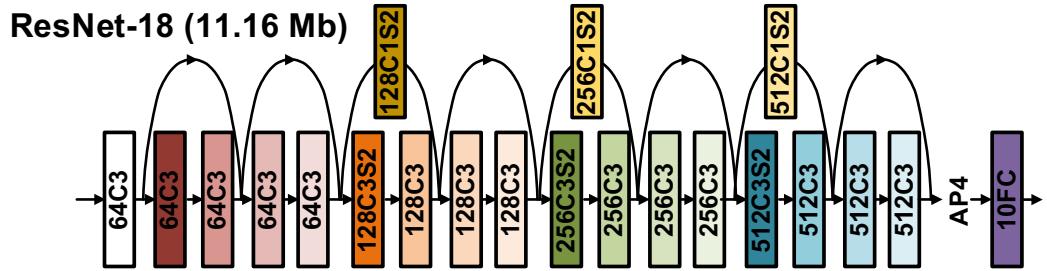
nC1S2: n 1x1 convolution (stride = 2)

MP2: 2x2 max pooling

AP4: 4x4 average pooling



PIMCA Chip Mapping for ResNet-18



ResNet-18 (11.16 Mb)

4X time-multiplexing w/ PIMCA
chip required

0,2	0,1	0,0	0,2	0,1	0,0
1,2	1,1	1,0	1,2	1,1	1,0
2,2	2,1	2,0	2,2	2,1	2,0

0,2	0,1	0,0	0,2	0,1	0,0
1,2	1,1	1,0	1,2	1,1	1,0
2,2	2,1	2,0	2,2	2,1	2,0

0,2	0,1	0,0	0,2	0,1	0,0
1,2	1,1	1,0	1,2	1,1	1,0
2,2	2,1	2,0	2,2	2,1	2,0

ResNet-18 (1/4)

0,2	0,1	0,0	0,2	0,1	0,0
1,2	1,1	1,0	1,2	1,1	1,0
2,2	2,1	2,0	2,2	2,1	2,0

0,2	0,1	0,0	0,2	0,1	0,0
1,2	1,1	1,0	1,2	1,1	1,0
2,2	2,1	2,0	2,2	2,1	2,0

0,2	0,1	0,0	0,2	0,1	0,0
1,2	1,1	1,0	1,2	1,1	1,0
2,2	2,1	2,0	2,2	2,1	2,0

ResNet-18 (3/4)

0,2	0,1	0,0	0,2	0,1	0,0
1,2	1,1	1,0	1,2	1,1	1,0
2,2	2,1	2,0	2,2	2,1	2,0

0,2	0,1	0,0	0,2	0,1	0,0
1,2	1,1	1,0	1,2	1,1	1,0
2,2	2,1	2,0	2,2	2,1	2,0

0,2	0,1	0,0	0,2	0,1	0,0
1,2	1,1	1,0	1,2	1,1	1,0
2,2	2,1	2,0	2,2	2,1	2,0

ResNet-18 (2/4)

0,2	0,1	0,0	0,2	0,1	0,0
1,2	1,1	1,0	1,2	1,1	1,0
2,2	2,1	2,0	2,2	2,1	2,0

0,2	0,1	0,0	0,2	0,1	0,0
1,2	1,1	1,0	1,2	1,1	1,0
2,2	2,1	2,0	2,2	2,1	2,0

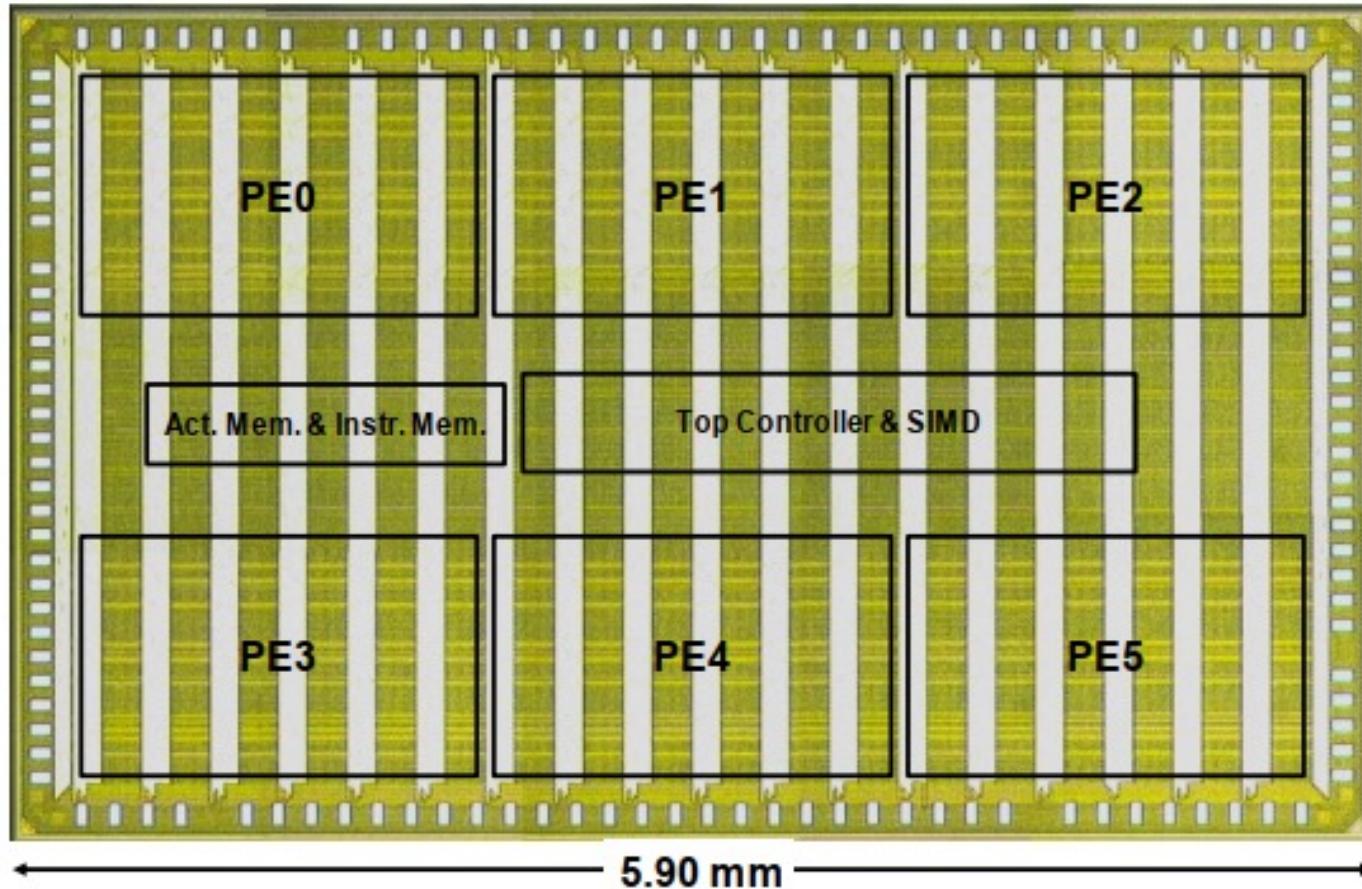
0,2	0,1	0,0	0,2	0,1	0,0
1,2	1,1	1,0	1,2	1,1	1,0
2,2	2,1	2,0	2,2	2,1	2,0

ResNet-18 (4/4)

Outline

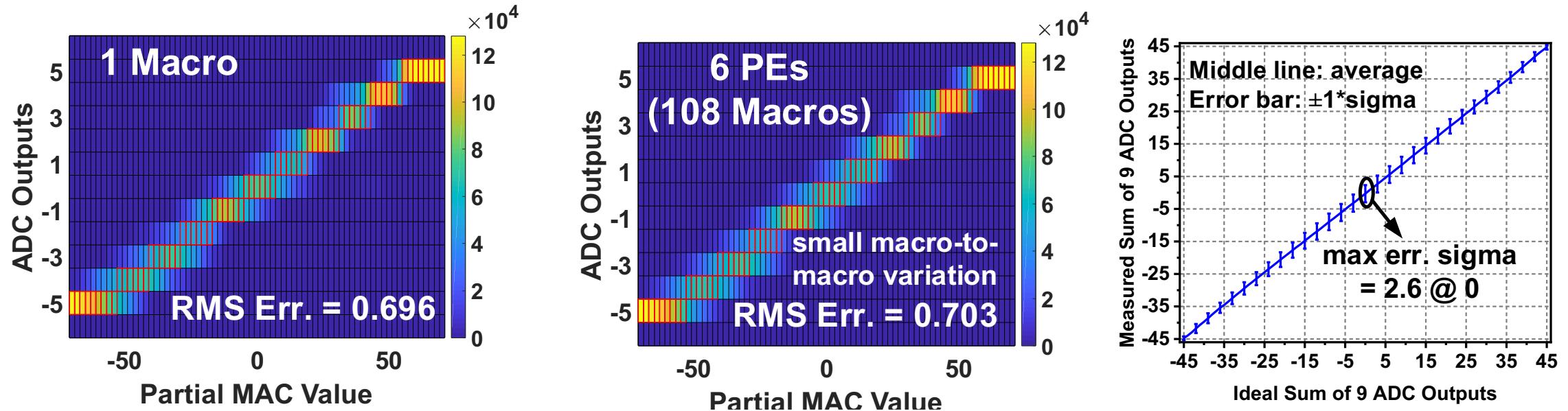
- Background & Motivation
- IMC Macro Design and Challenges of IMC Systems
- Programmable IMC Accelerator (PIMCA)
 - Overall Architecture
 - ISA & Loop Support Control
 - Activation Storage & Access
 - SIMD Design
 - PE Mapping
- **Measurement & Comparison**
- Summary

Chip Micrograph & Performance Summary



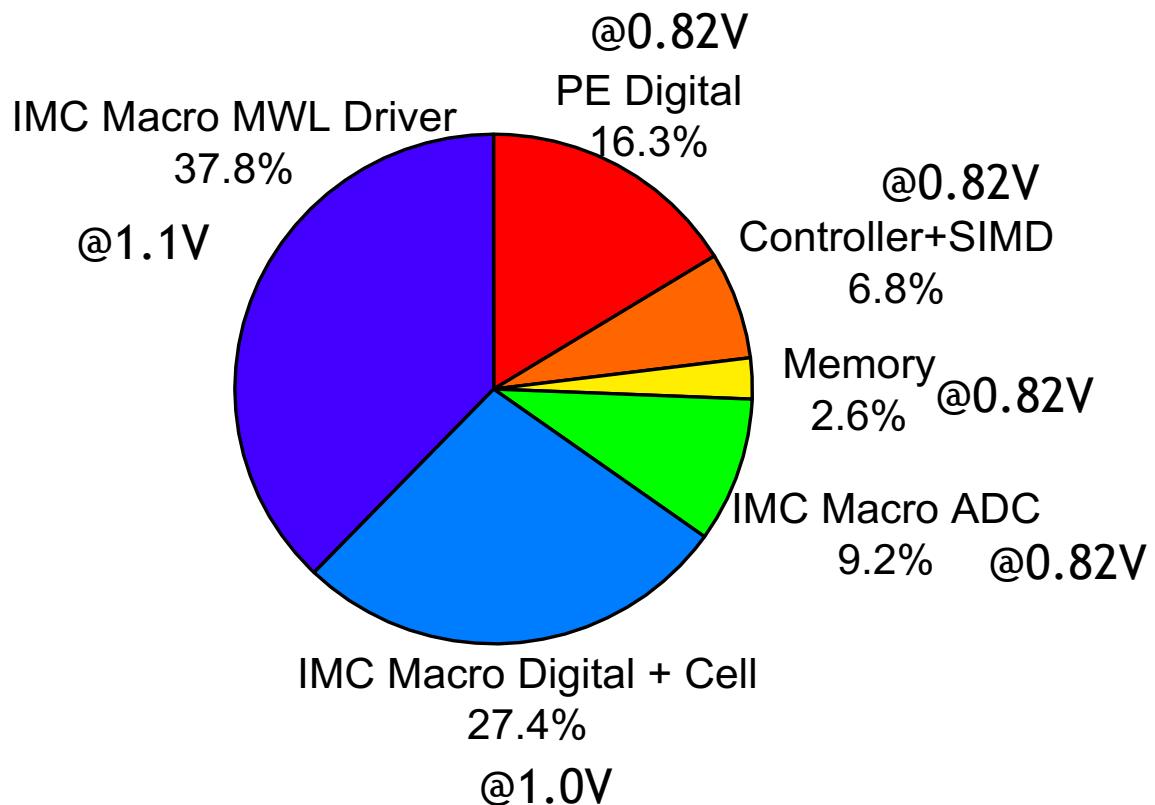
Technology	28nm
Chip Area	20.9 mm ²
Digital SRAM	201 kB
IMC SRAM	432 kB
Supply Voltage	0.82/1 V
Frequency	40 MHz
Activation Precision	1/2 bit
Weight Precision	1/2 bit
Peak Performance	4.9 TOPS (1b)
Power	108 mW (1b)
Peak/Avg. Energy Efficiency	437/289 TOPS/W (1b)

IMC Measurements and Variability



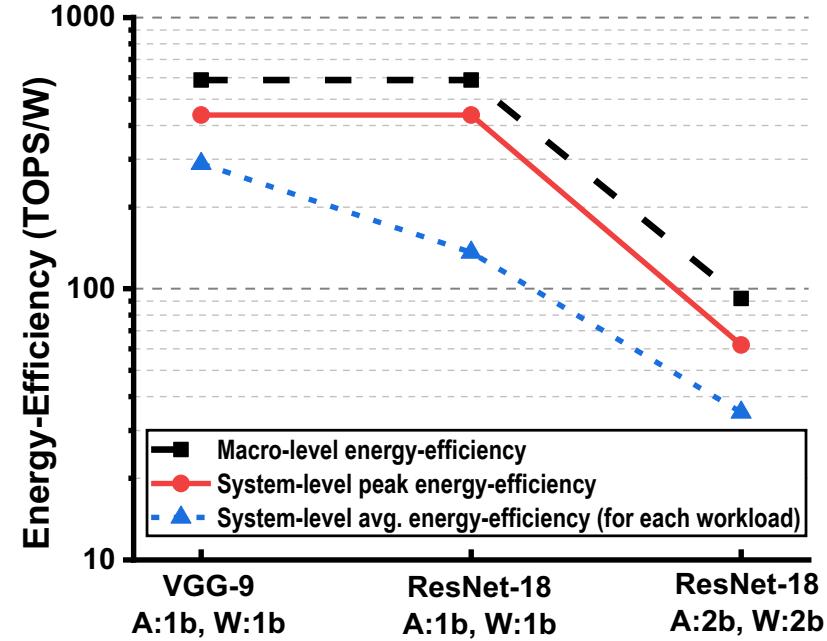
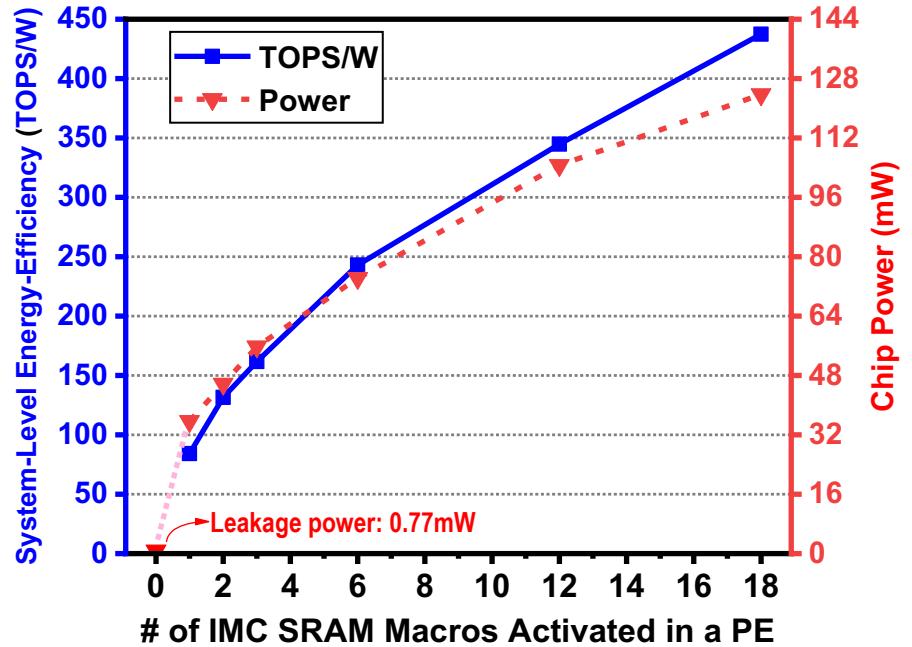
- ❑ Measurements for 2-D histograms / ADC plot: 1,000 random input vectors x 128 columns per macro (128k samples) generated for each partial MAC value, red boxes highlight ideal ADC outputs.
- ❑ Note that small macro-to-macro variation is observed.

Chip Power Breakdown



- IMC SRAM based PEs dominate power consumption
 - IMC SRAM macros' power dominate PE power consumption

Energy Efficiency Results



- PIMCA demonstrates high “system-level” energy-efficiency
- VGG-9 can be fit entirely on chip (no off-chip DRAM weight access needed); energy efficiency evaluated on the full network
- ResNet-18 requires time-multiplexing use of IMC macros; energy efficiency evaluated on the 6th basic block; off-chip DRAM power/latency not included in reported energy

Comparison with Prior Works

	Guo et al., VLSI'19	Yue et al., ISSCC'20	Valavi et al., JSSC'19	Jia et al., JSSC'20	Jia et al., ISSCC'21	This Work
Technology (nm)	65	65	65	65	16	28
DNN Model	RNN	CNN/FC	CNN	CNN/FC	CNN/RNN/FC	CNN/FC
Supported CNN Kernel	N/A	3x3	3x3	3x3	3x3, 1x1	3x3, 5x5, 1x1
Area (mm ²)	9.6	5.66	12.6	13.5	25	20.9
Digital SRAM (kb)	80	1,312	64	256		1,608
IMC SRAM (kb)	64	4	2,304	576	4,500	3,456
Bit Precision	1b	2/4/6/8b (Act.) 4/8b (Weight)	1b	1-8b	1-8b	1-2b
Performance (TOPS)	0.61	0.17-2.0	18.9	2.2 (1b)	11.8 (4b)	4.9 (1b)
IMC-macro-level Peak Energy Efficiency (TOPS/W)	51.6	158.7	886	400 (1b)	121 (4b)	588 (1b)
System-level Peak Energy Efficiency (TOPS/W)	11.7	35.8	N/A	N/A	N/A	437 (1b)
Energy per Inference for VGG-9 (μ J)	N/A	N/A	3.55 (1b)	5.31 (1b)	19.4 (4b)	2.36 (1b)

Outline

- Background & Motivation
- IMC Macro Design and Challenges of IMC Systems
- Programmable IMC Accelerator (PIMCA)
 - Overall Architecture
 - ISA & Loop Support Control
 - Activation Storage & Access
 - SIMD Design
 - PE Mapping
- Measurement & Comparison
- **Summary**

Summary

- Many SRAM IMC macros were presented (resistive / capacitive), nowadays demonstrating >1,000 of TOPS/W
- Going beyond IMC macro design
 - IMC accelerators that integrate many IMC macros
- PIMCA: programmable IMC accelerator with >100 IMC macros
 - 28nm prototype chip with 3.4Mb IMC SRAM & 1.5Mb activation SRAM
 - Programmability with custom ISA, HW loop support
 - Demonstrated high “system-level” energy-efficiency of 437 TOPS/W

Acknowledgments

- Faculty: Mingoo Seok (Columbia Univ.)
- Students: Shihui Yin, Minkyu Kim, Sai Kiran Cherupally, Jyotishman Saikia, Jian Meng (ASU), Zhewei Jiang, Bo Zhang (Columbia Univ.)
- Samsung: Soonwan Kwon, Sungmeen Myung, Hyunsoo Kim, Sang Joon Kim

- Sponsors:



This work was supported in part by C-BRIC, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.





tinyML Summit 2022 Sponsors



emza
visual sense



Sony
Semiconductor
Solutions
Corporation



SA STREAM ANALYZE





Copyright Notice

This presentation in this publication was presented as a tinyML® Summit 2022. The content reflects the opinion of the author(s) and their respective companies. The inclusion of presentations in this publication does not constitute an endorsement by tinyML Foundation or the sponsors.

There is no copyright protection claimed by this publication. However, each presentation is the work of the authors and their respective companies and may contain copyrighted material. As such, it is strongly encouraged that any use reflect proper acknowledgement to the appropriate source. Any questions regarding the use of any materials presented should be directed to the author(s) or their companies.

tinyML is a registered trademark of the tinyML Foundation.

www.tinyml.org