Aalto University
School of Electrical
Engineering

ELEC-L352001: Postgraduate Course in Electronic Circuit Design
In-memory computing using charge-based memory elements

Santeri Porrasmaa

Department of Electronics and Nanoengineering
Aalto University, School of Electrical Engineering
santeri.porrasmaa@aalto.fi

20.04.2022

Outline

Background

- Conventional von Neumannn computing architecture based on moving data between Central Processing Unit (CPU) and the memory (e.g Random Access Memory, RAM)
- The interface between the CPU and the memory introduces latency and consumes significant amount of power ("memory wall")
- Especially problematic in computationally intensive tasks, such as Machine Learning (ML) applications
- 1990's introduced near-memory computing, still physical separation between memory and CPU
- In-memory computing: Perform computation inside the physical memory, rather than moving data back and forth between the CPU and memory

Background

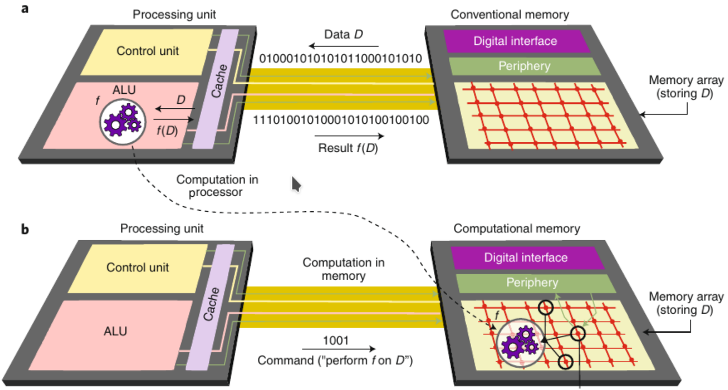


Figure: Illustration of differences between the conventional (a) and in-memory (b) computational architectures. [1]

[1] Sebastian, A. et al. https://doi.org/10.1038/s41565-020-0655-z

Outline

In-memory computing: Introduction

- In-memory computing (IMC) quite literally translates to performing logic operations with the memory elements
- Memory devices used can be roughly grouped to two: **charge-based** and resistance-based devices
- Charge-based memory: state of memory cell is determined by the presence or absence of charge on certain circuit node
- Primitive structures: Dynamic RAM (DRAM), Static RAM (SRAM) or Flash
- Flash-based devices seem to be rare, due to high-voltage requirement for writing, slow access time and reliability issues
- DRAM and SRAM are extensively utilized in IMC as primitives used to realize different logic functions

DRAM IMC macro

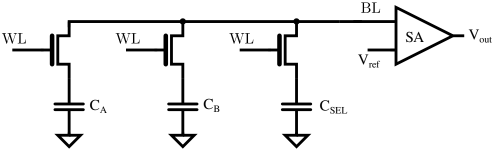


Figure: Example of a DRAM IMC macro

- States stored in capacitances C_A and C_B, C_{SEL} selects between AND and OR
- Connect all capacitors in parallel, bit line voltage approximates average of voltages A, B and SEL
- V_{ref} selected tactically (near V_{DD})
- Using complementary output of SA, we get complete set of Boolean operations!

SRAM IMC macro

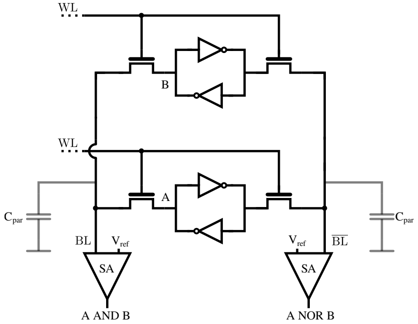


Figure: Example of a SRAM IMC macro

- Bit lines precharged to V_{DD}
- Parasitic capacitance of BL charges/discharges to some intermediate value between $V_{SS} \dots V_{DD}$ depending on inputs
- Once again, complementing outputs of SA yields functionally complete set of Boolean operations

Matrix-vector Multiplication (MVM)

- Matrix-vector Multiplication is a predominant kernel used in ML applications (e.g. image processing) [2]
- Multiplication requires cascaded logic operations using IMC macros, what if we want to do better?
- Idea is to multiply vector b with matrix A to get output c

$$A \cdot b = c \tag{1}$$

- A in above equation represents the weights (multipliers) for the input data in vector b
- Multiply and accumulate (MAC) between 2-D memory array (weights) and input data

[2] N. Verma et al., doi: 10.1109/MSSC.2019.2922889

MVM kernel for MAC operations

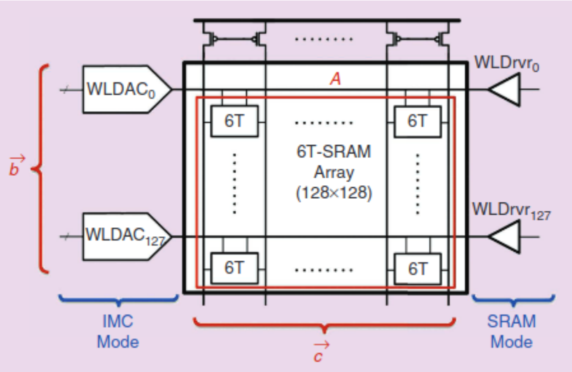
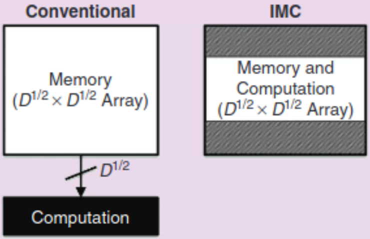


Figure: Conceptual illustration of a MVM kernel performing MAC operations on input data *b*. [2]

[2] N. Verma et al., doi: 10.1109/MSSC.2019.2922889

Comparison of conventional and IMC architectures



Metric	Traditional	In-Memory
Bandwidth	$1/D^{1/2}$	1
Latency	D	1
Energy	$D^{3/2}$	$\sim D$
SNR	1	$\sim 1/D^{1/2}$

Figure: Comparison of conventional and IMC architectures in terms of basic performance metrics. [2]

[2] N. Verma et al., doi: 10.1109/MSSC.2019.2922889

Outline

Background and motivation

Charge-based in-memory computing

State-of-the-Art

Conclusion

Homework assignment

In-memory classifier using 6T SRAM [3]

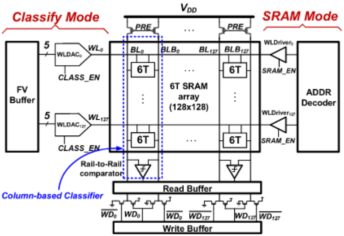


Figure: Top-level architecture.

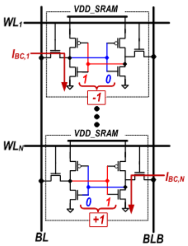


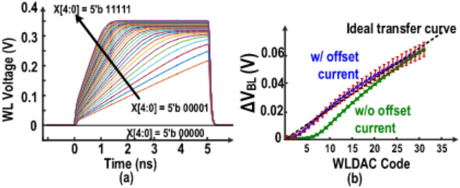
Figure: Column-based weak classifier.

- Application: low-power continuous coarse detection for a fully functional ML core
- 1-bit weights for each input

---

[3] J. Zhang et al., doi: 10.1109/JSSC.2016.2642198.

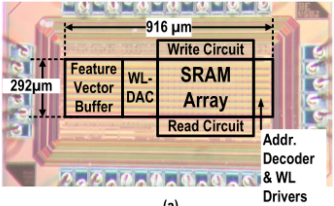
In-memory classifier using 6T SRAM: The DAC [3]



- Problem: pre-charged BL/$\overline{BL}$ voltages may pull the internal nodes high or low, flipping the state of the SRAM latch!
- The work solves the problem to driving the WL voltage only to 0.4V (1/3 of supply)
- Offset current source in parallel with IDAC to reduce non-linearity of lower range of input codes

[3] J. Zhang et al., doi: 10.1109/JSSC.2016.2642198.

In-memory classifier using 6T SRAM [3]



(a)

CHIP SUMMARY	
Technology	*130 nm*
SRAM Size	*128 × 128 bits*
Bit cell Size	*1.26 µm × 3.44 µm*
Energy / 10-way Class.	*633.4 pJ*
Speed (Classify)	*50 MHz*
Speed (SRAM)	*300 MHz*
Energy/cycle (Classify)	*46.6 pJ*
Energy/cycle (SRAM)	*14.7 pJ*

(b)

[3] J. Zhang et al., doi: 10.1109/JSSC.2016.2642198.

Robust In-Memory Machine Learning Classifier with On-Chip Training [4]

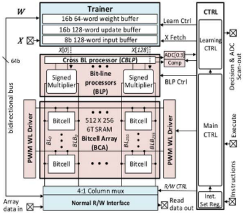
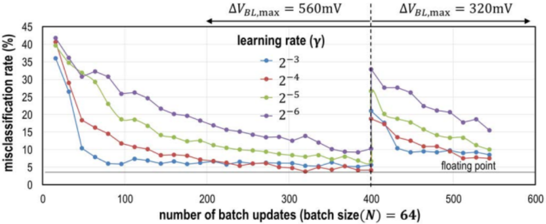


Figure: Top-level architecture.

- Instead of voltage-based DAC, generate PWM signal (pulse width proportional to input)
- PVT compensation by on-chip training: reduction in misclassification rate from 18 % down to 8% with training

[4] S. K. Gonugondla, et al., doi: 10.1109/ISSCC.2018.8310398.

Robust In-Memory Machine Learning Classifier with On-Chip Training [4]



misclassification rate (%)
($\Delta V_{BL,\max} = 320\text{mV}$)
tested on

trained on	Chip1	Chip2	Chip3	Chip4	Chip5
Chip1	8.25	38.3	48.3	51.5	48.8
Chip2	45.8	9	48	49.8	34.5
Chip3	47	51.3	8.5	29.8	49.3
Chip4	51.5	51	17.5	8.25	51.3
Chip5	38.3	18	48.5	48.5	8

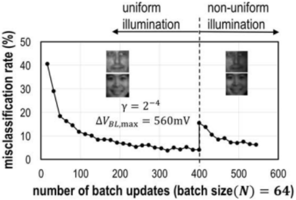


Figure: Effect of PVT compensation by on chip-training.

[4] S. K. Gonugondla, et al., doi: 10.1109/ISSCC.2018.8310398.

Robust In-Memory Machine Learning Classifier with On-Chip Training [4]

	[1]	[2]	[5]	[6]	[3]	[4]	this work
Technology	65nm	28nm HPC	40nm	65nm	65nm	180nm	65nm
Algorithm	CNN	FC-DNN	matrix mult.	filtering	SVM	AdaBoost	SVM
Data set	ImageNet	MNIST			MIT-CBCL	MNIST	MIT-CBCL
Architecture	digital	digital	analog	analog	in-memory	in-memory	in-memory
On-chip learning	No	No	No	No	No	No	Yes
Total SRAM size (kb)	1449.2	9248	–	–	128	103.6	128
Energy/Decision	7.94mJ[d]	0.56uJ	–	–	0.4nJ	0.6nJ	0.042nJ
Decisions/s	35	28.8k[d]	–	–	9.2M	7.9M	32M
# of MACs/Decision	2663M	334k	–	–	512	–	128
Max. accuracy (%)	–	98	–	–	96	91	96
MAC level metrics							
MAC precision[a] ($B_x \times B_w$)	$16^s \times 16^s$	$8^s \times 8^s$	$3^s \times 6^s$	8×14^s	8×8	5×1	8×8^s
Efficiency (TOPS/W)	0.336[d]	0.56[d]	3.84[b]	0.5[b]	1.25	–	**3.125**
MAC energy (E_{MAC}) (pJ)	2.98[d]	1.79[d]	0.26[b]	2[b]	0.8	–	**0.32**
precision-scaled MAC energy[c] (fJ)	11.6	28	14.4[b]	17.857[b]	12.5	–	**4.9**
Estimated performance of prior art to realize SVM algorithm with vector dimension of 128							
Energy/Decision (nJ)	0.381	0.229	0.033[b]	0.256[b]	0.102	–	**0.042**
Decisions/s	250M	75M	19.5M	350k	36.8M	–	32M
# MACs per cycle	168	8	1	64	256	10,368	128

[a] s indicates signed; B_x: input precision; B_w: weight precision
[b] does not include SRAM memory access
[c] normalized to account for operand precision ($E_{MAC}/(B_x \times B_w)$)
[d] estimated from reported data

Figure: Comparison to prior work.

[4] S. K. Gonugondla, et al., doi: 10.1109/ISSCC.2018.8310398.

Compute-in-Memory SRAM Macro with Multi-bit Input, Weight and Output [5]

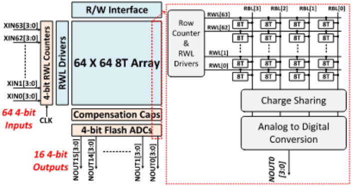


Figure: Top-level architecture.

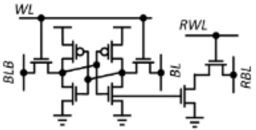


Figure: 8T SRAM cell.

- Replace 6T SRAM with 8T for better resilience to internal SRAM errors
- 4-bit input realized with 4 bit unit pulses on RWL line, more linear than PWM
- Weights realized by sampling RBL on binary weighted capacitors
- 4-b flash ADC for digital output

[5] M. E. Sinangil et al., doi: 10.1109/JSSC.2020.3031290.

Compute-in-Memory SRAM Macro with Multi-bit Input, Weight and Output [5]

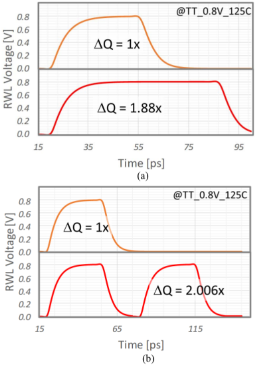


Fig. 6. Comparison of (a) pulsewidth modulation and (b) multiple unit pulses with respect to linearity of multi-bit representation. Multiple unit pulses provide better linearity when comparing the total charge removed for 4'd1 and 4'd2 input activations.

[5] M. E. Sinangil et al., doi: 10.1109/JSSC.2020.3031290.

Compute-in-Memory SRAM Macro with Multi-bit Input, Weight and Output [5]

TABLE I

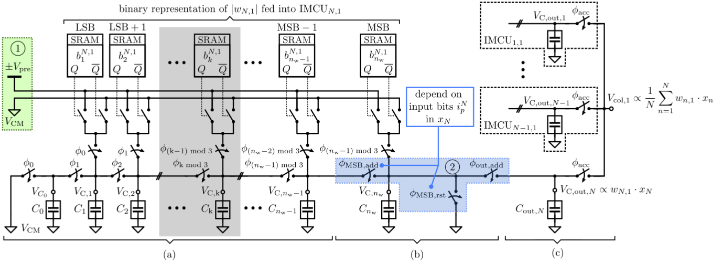
COMPARISON OF THIS WORK TO PREVIOUSLY PUBLISHED WORK

	JSSC'17 [15]	ISSCC'18 [16]	ISSCC'18 [17]	ISSCC'18 [18]	ISSCC'19 [19]	This Work	
Technology	130nm	65nm	65nm	65nm	55nm	7nm	
Array Size	16kb	128kb	4kb	16kb	3.8k	4kb	
Cell Type	6T	6T	S6T	10T	T8T	8T	
Push Rule	No	No	Yes	No	Yes	Yes	
Bitcell Area (μm²)	4.334	NA	0.525	NA	0.865	0.053	
Input Bits	4	1	1	7	4	4	
Weight Bits	1	8	1	1	5	4	
Output Bits	1	4	1	7	7	4	
Power Supply (V)	1.2 & 0.4	1.0	1 & 0.8	1.2 & 0.9	1.0	0.8	1.0
Cycle Time (ns)	20	NA	2.3	150	10.2	5.5	4.5
Throughput (GOPS)	NA	4	1780 1)	10.67	17.6	372.4 2)	455.1
Energy Efficiency (TOPS/W)	NA	3.125	55.8	28.1	18.4	262.3 ~ 610.5 351 in average	189.3 ~ 435.5 321 in average

1) Each operation is only 1b X 1b
2) Each 4b X 4b is considered as 2 operations

[5] M. E. Sinangil et al., doi: 10.1109/JSSC.2020.3031290.

An SRAM-Based Multibit In-Memory MVM With a Precision That Scales Linearly in Area, Time, and Power [6]



- Pipelined MAC operation enables scaling of area, time and power linearly with resolution

[6] R. Khaddam-Aljameh, et al., doi: 10.1109/TVLSI.2020.3037871.

An SRAM-Based Multibit In-Memory MVM With a Precision That Scales Linearly in Area, Time, and Power [6]

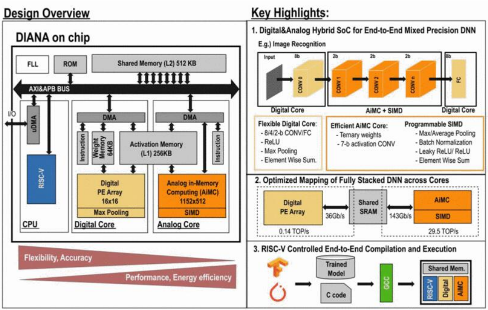
Metric	This work	ISSCC'20 [30]	JSSC'19 [18]	JSSC'19 [17]	JSSC'20 [31]
Technology	14 nm	7 nm	65 nm	65 nm	65 nm
Operating Voltage in V	0.8	1.0	1.0	0.94, 0.68, 1.2	1
Numbers obtained from:	Simulation	Measurement	Measurement	Measurement	Measurement
Input D/A conversion	Fully Digital FSM	PWM-based	Partially PWM-based	Not required (binary only)	Not required (binary only)
Weight D/A conversion	Fully Digital FSM	Charge-sharing-based	Not required (binary only)	Not required (binary only)	Not required (binary only)
Output A/D conversion	8bit SAR ADC	4bit Flash-ADC	7bit charge-sharing ADC	Batch-Norm. using 6bit DAC	5bit Flash-ADC
SRAM size	256 KB	4 KB	2 KB	295 KB	16 KB
SRAM bitcell	8T	8T	10T	10T1C	8T1C
SRAM words per IMCU (n_{shared})	32	1	16	1	1
Number of weight-bits (n_{w} + sign)	6	4	1	1	1
Number of input-bits (n_{x} + sign)	6	4	6	1	1
Number of output-bits	8	4	7	1	5
Relation between number of weight-bits n_{w}					
... and (cell) area	linear	exponential[1]	binary only	binary only	binary only
... and latency	linear	constant	binary only	binary only	binary only
... and power	linear	linear	binary only	binary only	binary only
Relation between number of input-bits n_{x}					
... and latency	linear	exponential[1]	exponential[1]	binary only	binary only
... and power	constant	constant	linear	binary only	binary only
Peak Throughput (TOP/s)	2.43 | 0.52*	0.372 | 0.04*	0.008	18.79	1.638
... with precision scaling	87.38 | 18.82*	5.958 | 0.64*	0.048	18.79	1.638
Energy Efficiency (TOP/s/W)	16.94 | 3.65*	351 | 37.8*	40.3	866	671.5
... with precision scaling	609.7 | 131.3*	5616 | 604.8*	241.8	866	671.5
Area Efficiency (TOP/s/mm^2)	3.98 | 0.86*	116.4 | 12.53*	0.092	1.5	20.22
... with precision scaling	143.2 | 30.84*	1862 | 200.5*	0.553	1.5	20.22

[1] scales with power of 2, * Normalized to 65 nm

- Precision scaling indicates efficiency while taking into account the accuracy

[6] R. Khaddam-Aljameh, et al., doi: 10.1109/TVLSI.2020.3037871.

DIANA: An End-to-End Energy Efficient Digital and Analog Hybrid Neural Network SoC [7]



- SRAM-based analog core used for low-accuracy, but computationally intensive tasks (convolutional problems, etc)
- Digital core used for simpler, medium-accuracy, tasks

[7] K. Ueyoshi et al., doi: 10.1109/ISSCC42614.2022.9731716

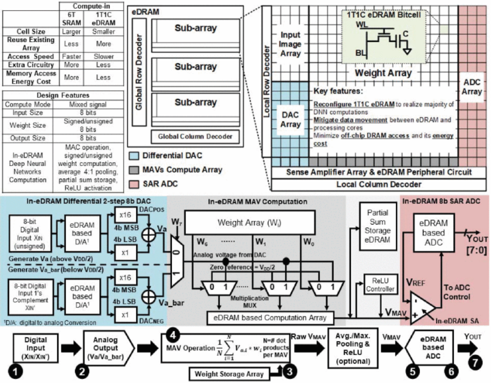
DIANA: An End-to-End Energy Efficient Digital and Analog Hybrid Neural Network SoC [7]

	ISSCC21 Digital [2]	ISSCC21 In-memory [3]	ISSCC21 In-memory [4]	DIANA Our work
Technology	28nm	65nm	16nm	22nm
Architecture	Digital	AiMC	AiMC	CRISC-V+digital+AiMC
Precision(A)	8	2/4/6/8	1-8	7(analog), 2/4/8 (digital)
Precision(W)	8	1-8	1-8	ternary (analog) 2/4/8 (digital)
Clock freq. (MHz)	100- 470	25-100	200	50-320
Peak performance (TOPs)	0.14 @ 470MHz	3.16 @100MHz	11.8 @ 200MHz	29.5 (analog) @250MHz 0.14 (digital) @250MHz
Area efficiency (Tops/mm2)	0.745	0.380	2.67	12.88 (AiMC macro) 3.33 (analog + digital)
Peak efficiency (TOPs/W)	12.1 @0.9V, 470MHz	370 (macro) 75.9 (system*)	121 (macro)	**Digital Core:** 4.1 (0.55V, 50MHz) (measured) **Analog Core:** 600 (Logic&mem:0.55V, 70MHz AiMC: 0.55V) (measured)
CIFAR10 TOPs/W	-	24.14(macro) 9.01(system*) (4b-4b)	78.3 (macro)	14.4 (Logic&mem.: 0.55V, 75MHz, AiMC: 0.55V) (system, end-to-end all digital and analog) (measured)
CIFAR10 Latency	-	7.95 ms	0.13ms	1.24 ms (same condition as above row)
ImgNet TOPs/W	12.62 **	7.32 (macro*) 2.75 (system*)	11.67 (macro)	67.8 (system on analog-executed layers) 19.0 (system, end-to-end all digital and analog) (Logic&mem: 0.8V, 250MHz, AiMC: 0.8V, simulated)
ImgNet Latency	24.8 ms (end-to-end)	112 ms (estimated from typical layer result)	1.72 ms (excluding CONV1, FC)*	6.15ms (end-to-end, simulated)
End-to-End	Yes	No	No	**Yes**

* Not including subsampling layers (batch normalization, pooling).
** Skips computation by predicting zero output

[7] K. Ueyoshi et al., doi: 10.1109/ISSCC42614.2022.9731716

DRAM-based approach [8]



- Memory, DAC and ADC all based on DRAM cells

[8] S. Xie, et al. doi: 10.1109/ISSCC42613.2021.9365932.

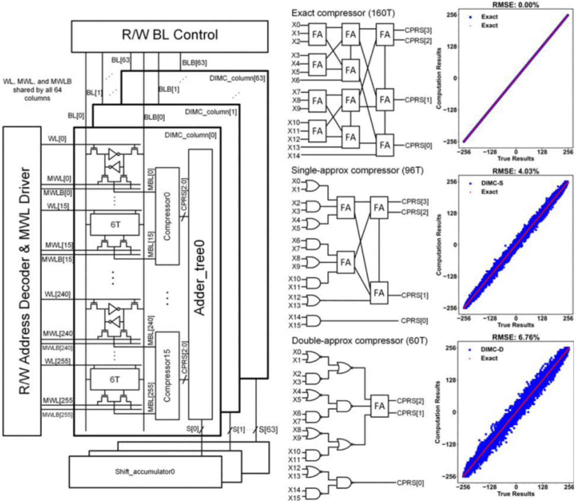
DRAM-based approach [8]

	This work	ISSCC'20 [3]	ISSCC'20 [4]	ISSCC'20 [5]	ISSCC'18 [6]
Technology	65nm	28nm	28nm	22nm	65nm
Memory Cell Structure	1T1C eDRAM	6T SRAM	6T + Local Computing SRAM	1T1R SLC ReRAM	6T SRAM
Array Size	16Kb	64Kb	64Kb	2Mb	128Kb
Input Precision (bit)	8	8	8	4	8
Weight Precision (bit)	8	8	8	4	8
Supply Voltage (V)	1~1.2	0.85~1.0	0.7~0.9	0.8	1
Dataset	CIFAR-10	CIFAR-10			
Model	CNN: 4 CONV + 2 Pooling + 2 FC	CNN: ResNet-20	CNN: ResNet-20	N/A	SVM
Measured Accuracy	80.1% (Top-1), 98.1 % (Top-5)	[5]91.91%	[5]92.02%	N/A	[5]83.27%
Throughput (GOPS)	[1,3]4.71	N/A	N/A	N/A	4
Average Energy Efficiency (TOPS/W)	[1]4.76	7.3 [2](1.35)	14.08 [2](2.61)	28.93 [2](3.31)	3.125
$GOPS/mm^2$	8.26	N/A	N/A	N/A	2.78
[4]FoM	304.6	86.4	167	53	201.6

[1]measured at 1.1V
[2]Scaled to 65nm, assume energy ∝ $(Tech.)^2$ [8]
[3] Limited by clocking infrastructure, chip size, technology and bit cell area
[4]FoM = input precision x weight precision x energy efficiency (scaled to 65nm)
[5]Top-1 or Top-5 is not mentioned

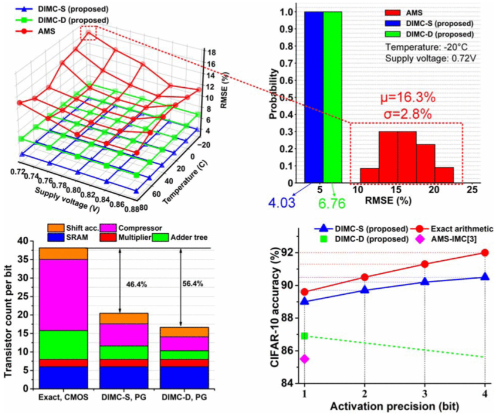
[8]S. Xie, et al. doi: 10.1109/ISSCC42613.2021.9365932.

DIMC: Digital In-Memory Computing Macro Based on Approximate Arithmetic Hardware [9]



[9] D. Wang, et al. doi: 10.1109/ISSCC42614.2022.9731659.

DIMC: Digital In-Memory Computing Macro Based on Approximate Arithmetic Hardware [9]



[9] D. Wang, et al. doi: 10.1109/ISSCC42614.2022.9731659.

DIMC: Digital In-Memory Computing Macro Based on Approximate Arithmetic Hardware [9]

	This work		ISSCC21[2]	JSSC20[3]	ISSCC21[5]	ESSCIRC19[7]
	DIMC-D	DIMC-S				
Technology[nm]	28	28	16	65	22	65
MAC operation	Digital	Digital	AMS	AMS	Digital	Digital
Array size	16Kb	16Kb	4.5Mb	16Kb	64Kb	16Kb
Macro size [mm^2]	0.033	0.049	11	0.081	0.202	0.227
Area efficiency [F^2/b]	2,569	3,814	9,179	1,170	6,368	3,279
Supply voltage [V]	0.45-1.10	0.45-1.10	0.8	0.8	0.72	0.6-0.8
Activation precision [bit]	1	1-4	1-8	1	1-8	1-16
Weight precision [bit]	1	1	1-8	1	4/8/12/16	4/8/12/16
Operating frequency [MHz]	280	250	20[1]	50	500	138
Input toggle rate	25%	25%	NA	NA	18%	NA
Energy efficiency [TOPS/W]	1,108 @ 0.9V	154 @ 0.9V (4b1b)	121 @ 0.8V (4b4b)	671 @ 0.8V	89 @ 0.72V (4b4b)	117 @ 0.6V (1b1b)
	2,219 @ 0.5V	248 @ 0.5V (4b1b)				
Throughput [GOPS][2]	9,175 @ 0.9V	2,035 @ 0.9V (4b1b)	41 @ 0.8V (4b4b)	1,638 @ 0.8V	825 @ 0.72 (4b4b)	567 @ 0.8V (1b1b)
	20,032 @ 1.1V	4,804 @ 1.1V (4b1b)				
CIFAR-10 accuracy	86.96%	90.41%	91.51%	85.50%	NA	NA

[1] Computed from throughput and array size; [2] Normalized array size to 16kb.

[9] D. Wang, et al. doi: 10.1109/ISSCC42614.2022.9731659.

Outline

Conclusion

- Traditional von-Neumann architectures suffer from CPU-RAM bottleneck ("memory wall")
  - Throughput (speed)
  - Energy penalty
- Bottleneck becomes significant especially computationally intensive applications, such as ML
- In-memory computing (IMC) is one possible approach of alleviating the "memory wall" problem
- IMC utilizes memory macros
  - SRAM (predominant)
  - DRAM
  - Flash

Conclusion

- Possiblity of using AMS, fully digital approach (or combination of both)
  - AMS-based approaches
    - Massive parallelism $\Rightarrow$ improved bandwidth.
    - Loss of generality, flexibility
    - Suffer from PVT variations and reduction in SNR
    - Suitable for low-accuracy applications
  - Digital approaches
    - Parallelism requires huge area footprint, limited bandwidth.
    - Robust with respect to PVT and noise
    - More flexible than AMS approaches
- Surveyed state-of-the-art approaches almost exclusively use AMS-based approaches and compensate for PVT

Outline

Homework assignment

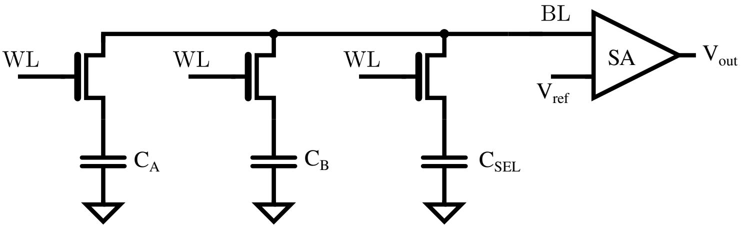


Figure: Example of a DRAM macro

Assumptions: $C_A = C_B = C_{SEL} = C$, $C_{par,BL}$ is negligible, SA input is high-impedance, $V_{ref} = \frac{V_{DD}}{2}$

1. Derive output voltages for each input combination assuming $V_{SEL} = 0$
2. Derive output voltages for each input combination assuming $V_{SEL} = V_{\mathrm{DD}}$
3. Which Boolean operation (AND, OR) is realized in case 1? What about case 2?