

Computer Architecture and Memory System Design for Deep Neural Networks



Author :

Hyun Kim

Affiliation :

Seoul National University of Science and Technology
Electrical and Information Engineering

Position :

Assistant Professor

Contact :

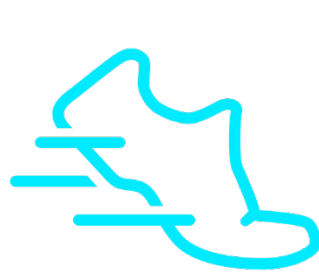
hyunkim@seoultech.ac.kr / 010-9600-5427
idsl.seoultech.ac.kr

Key Issue of Mobile AI Accelerators

- AI accelerators based on digital system design are expected as a solution to these challenges in AI
- Three main goals of AI accelerators: **High accuracy** + **High speed (throughput)** + **Low power**



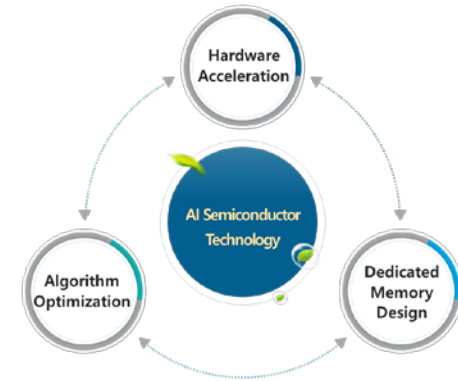
Accuracy ↑



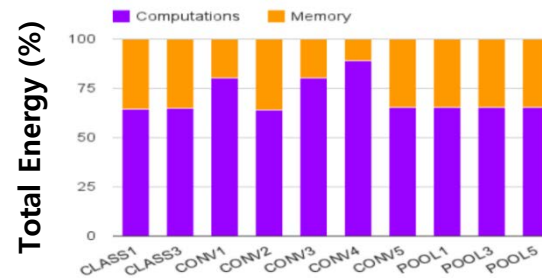
Speed ↑



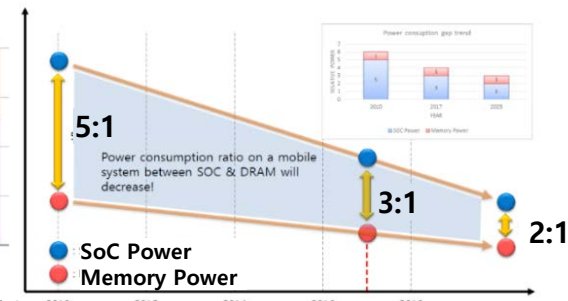
Power ↓



- Necessity of Memory-level approach:** In AI systems, power for memory accesses/data retention accounts for a large portion of total power, and in certain systems, latency for memory accesses may cause a decrease in system speed → Low-power and high-speed memory platform dedicated to DNN leads to **power-saving** and **speed-up** of AI accelerator platforms



[Ref] T. Chen, et al., "DianNao," ASPLOS, 2014.



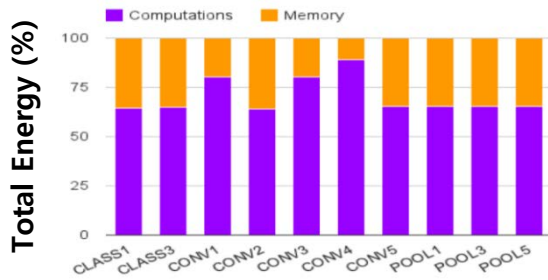
[Ref] SK Hynix

Operation	Energy [pJ]	Relative Cost
32 bit int ADD	0.1	1
32 bit float ADD	0.9	9
32 bit Register File	1	10
32 bit int MULT	3.1	31
32 bit float MULT	3.7	37
32 bit SRAM Cache	5	50
32 bit DRAM Memory	640	6400

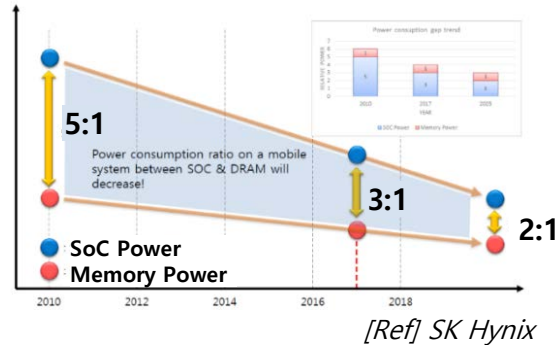
[Ref] Energy table for 45nm CMOS process

Importance of Memory Accesses in DNNs

- Increase of layers in DNN → Increase of Memory BW
- In DNN, **power for memory accesses** accounts for a large portion of total power
- Importance of memory power is also increasing in mobile SoC



[Ref] T. Chen, et al., "DianNao," ASPLOS, 2014.



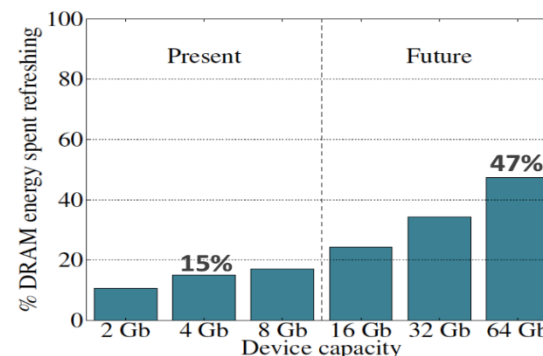
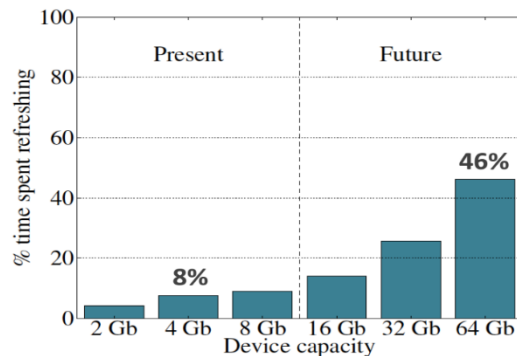
[Ref] SK Hynix

Operation	Energy [pJ]	Relative Cost
32 bit int ADD	0.1	1
32 bit float ADD	0.9	9
32 bit Register File	1	10
32 bit int MULT	3.1	31
32 bit float MULT	3.7	37
32 bit SRAM Cache	5	50
32 bit DRAM Memory	640	6400

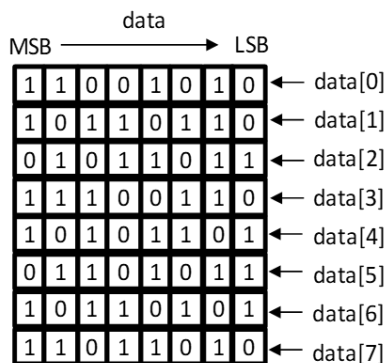
[Ref] Energy table for 45nm CMOS process

Refresh Cost of DRAM

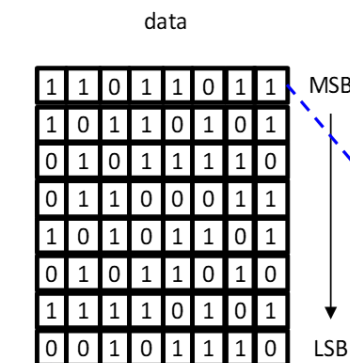
- As device density increases, the proportion of throughput loss and power consumption due to refresh increases → On 64Gb devices, about 50% of throughput loss and power consumption are caused by refresh operations



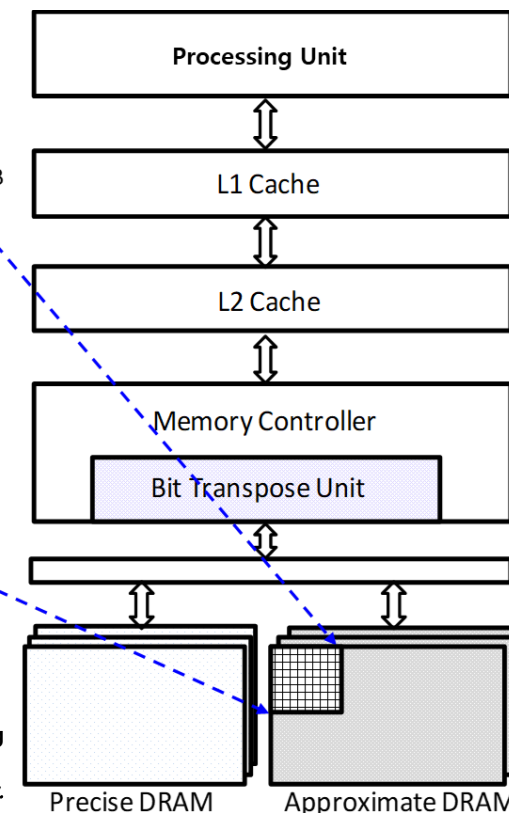
- ◆ **Goal:** Achieve **DRAM power-saving** with **negligible accuracy degradation** in CNN applications
- ◆ **Motivation:** Significant power consumption of **Refresh** operations in DRAM
 - As device density increases, the proportion of throughput loss and power consumption due to refresh increases
 - On 64Gb devices, about 50% of throughput loss and power consumption are caused by refresh operations
- ◆ **Solution/Contribution:** Concept of **approximate DRAM** is to **prevent errors in critical data** and **allow some errors in non-critical data for power-saving**
 - 1) **Bit-transpose module** to reallocate the bits to be stored in DRAM according to bit-significance



(a) Conventional data storage scheme



(b) Transposed data storage scheme



(c) System architecture to support approximate memory

Code for CPU or system → Precise DRAM
 Data for CNN → Approximate DRAM

◆ Related Papers

- "An approximate memory architecture for a reduction of refresh power consumption in deep learning applications," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May. 2018 (Google Scholar Citation: 38)
- "An Approximate Memory Architecture for Energy Saving in Deep Learning Applications," *IEEE Trans. Circuits Syst. I*, vol.67, no.5, pp.1588-1601, May 2020

Approximate Memory for DNNs (2)

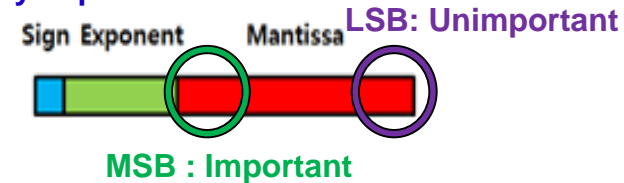
- ◆ **Goal:** Achieve **DRAM power-saving** with **negligible accuracy degradation** in CNN applications
- ◆ **Motivation:** Significant power consumption of **Refresh** operations in DRAM
 - As device density increases, the proportion of throughput loss and power consumption due to refresh increases
 - On 64Gb devices, about 50% of throughput loss and power consumption are caused by refresh operations
- ◆ **Solution/Contribution:** Concept of **approximate DRAM** is to **prevent errors in critical data** and **allow some errors in non-critical data** for power-saving
 - **2) Hard approximation** (Data truncation) to reduce memory access & **Soft approximation** (Refresh control) to reduce the refresh power

Normal DRAM refresh rate to prevent errors

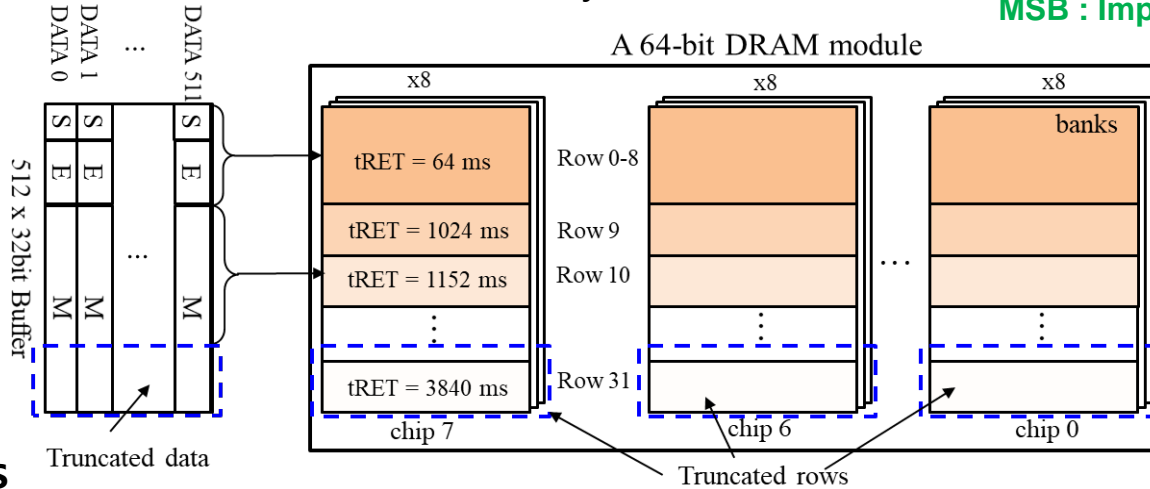
$$RP(n) = \begin{cases} 64 & \text{for } 0 \leq n \leq 8 \quad \text{Sign, Exponent} \\ (n-9) * incr + offset & \text{for } 9 \leq n \leq 31 \quad \text{Mantissa} \end{cases}$$

● Adaptive refresh rate for FP32

Very important



Slowed down DRAM refresh rate for power-saving where *incr* and *offset* are adjustable



No change in existing DRAM devices

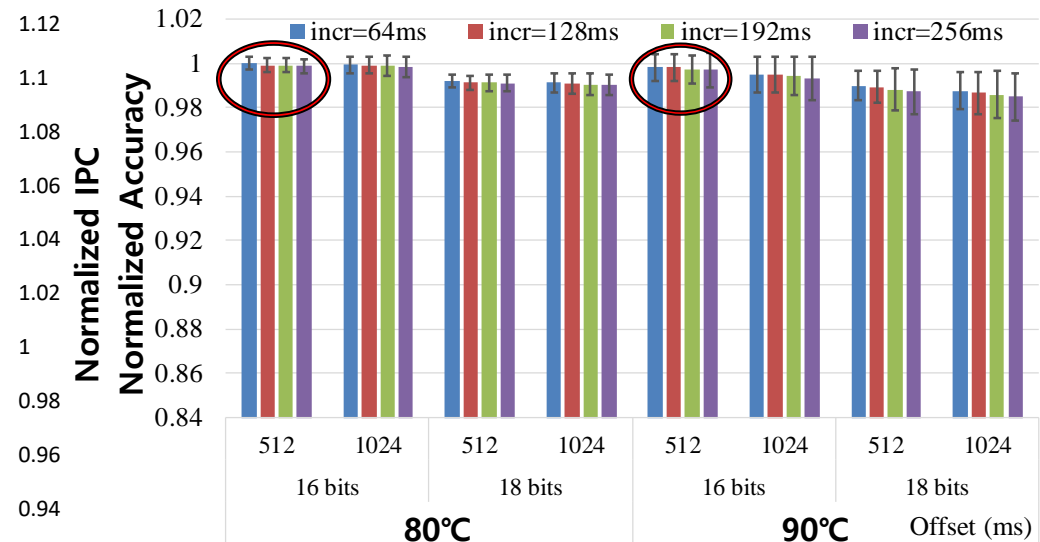
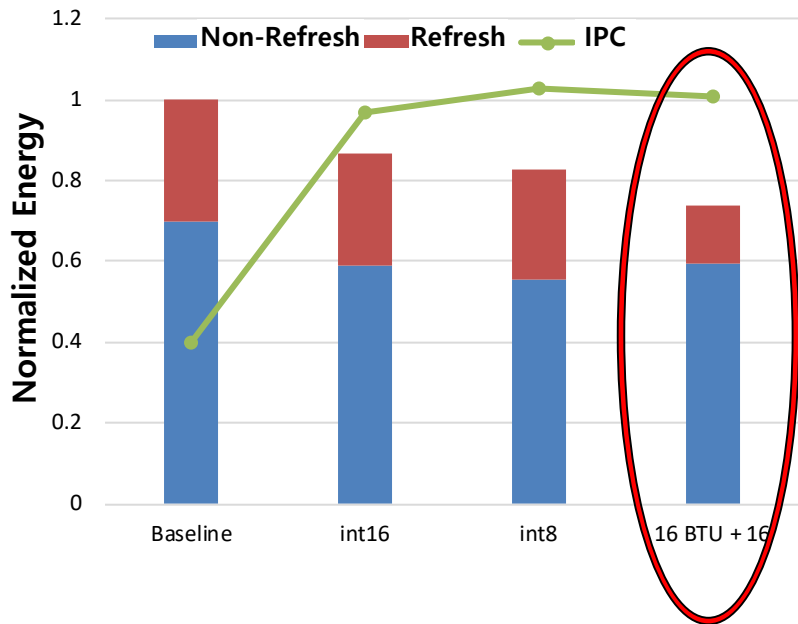
incr = 128 & *offset* = 1024

Related Papers

- "An approximate memory architecture for a reduction of refresh power consumption in deep learning applications," in *Proc. IEEE ISCAS*, May, 2018 (Google Scholar Citation: 38)
- "An Approximate Memory Architecture for Energy Saving in Deep Learning Applications," *IEEE Trans. Circuits Syst. I*, vol.67, no.5, pp.1588-1601, May 2020

◆ Experimental Results

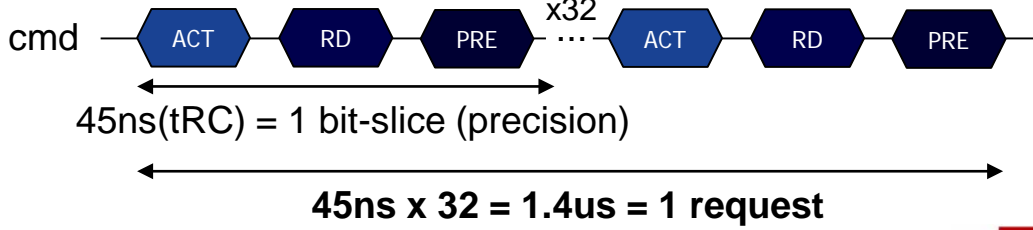
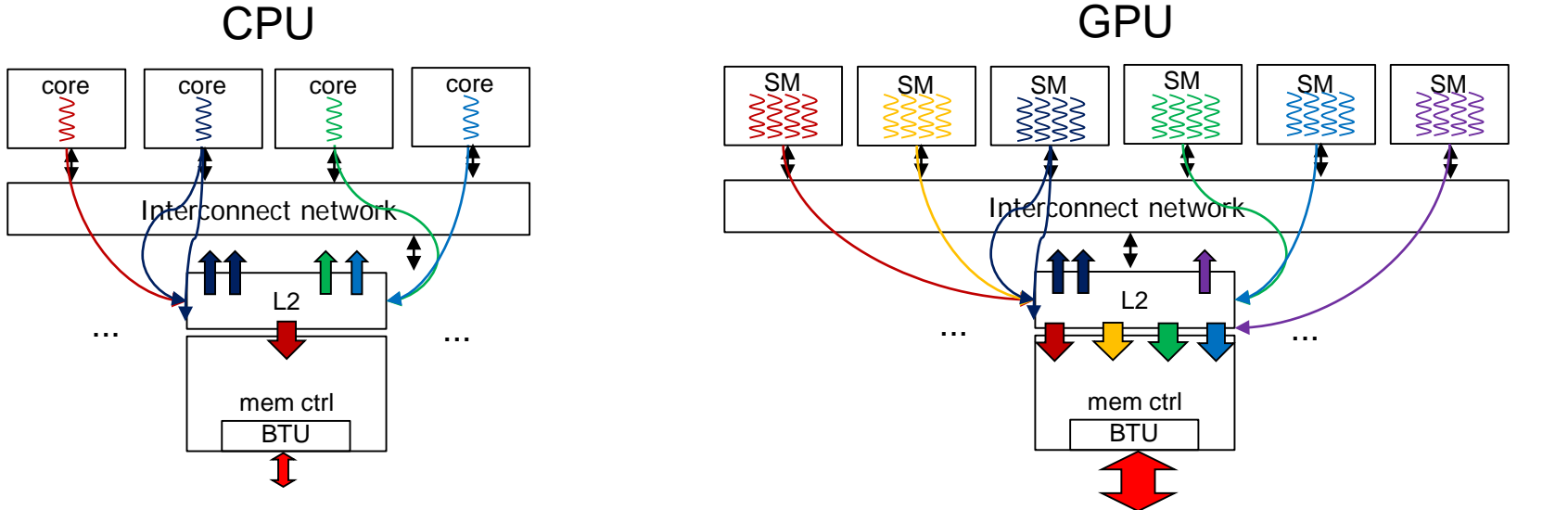
- 16bit-truncation + soft approximation with (offset, incr)=(512, 256) performs the best in terms of both power saving and accuracy
- **69.71% of DRAM refresh energy saving** and **26% of total DRAM energy saving** compared to Baseline (GoogLeNet)
- **<1% loss** even at the high (80°C & 90°C) temperature (GoogLeNet & AlexNet)
- In DNN applications, low BTU miss rate is possible owing to the high locality of memory access → **Minimize the performance (IPC) degradation** caused by BTU



← Verification on the Xilinx Virtex FPGA!

■ Problems due to Naïve Transposed Memory on GPU

- GPU consists of a lot of Streaming Multiprocessor (SM)
 - SIMT (Single Instruction Multiple Thread) architecture : cache hit rate degradation → DRAM access increase
- Compared to CPU-based systems, DRAM bandwidth is a performance bottleneck



"PCM: Precision-Controlled Memory System for Energy Efficient Deep Neural Network Training," 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 2020, pp. 1199-1204

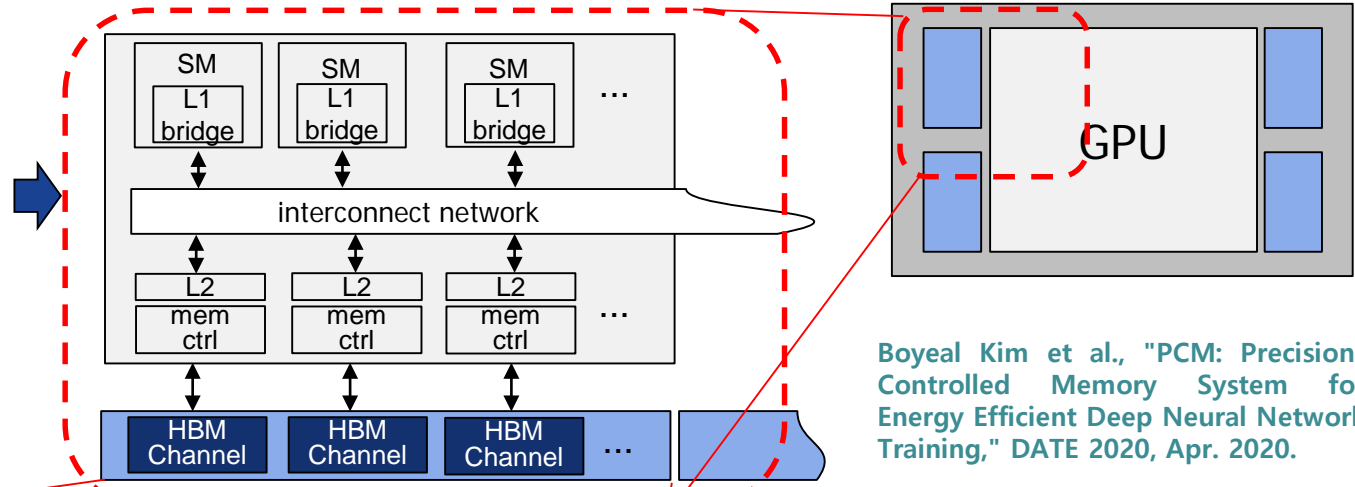
- Memory system design to reduce **both dynamic energy and refresh energy**
- Targeting **DNN training on GPU**
- By efficiently supporting two approximation scheme in arbitrary bit-width (e.g., 9bits)
- With transposed data mapping on DRAM and redesigning L1 cache

```

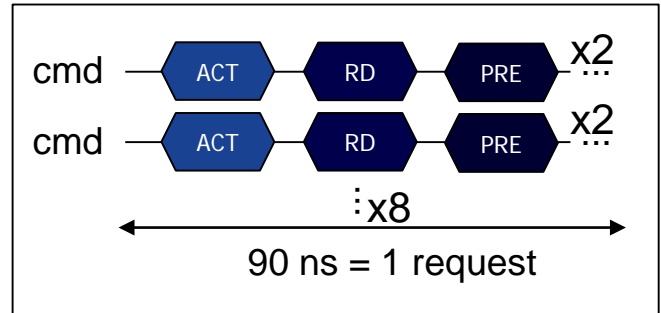
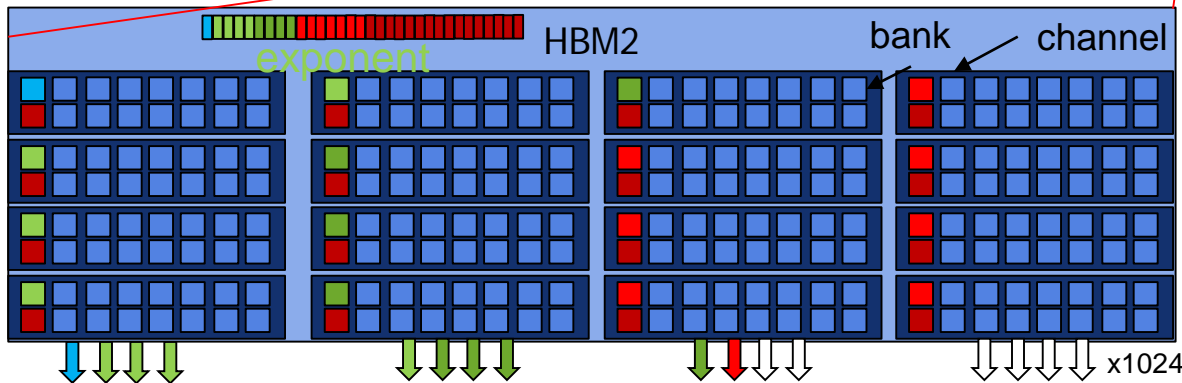
struct FP10_t{
  m_fetch_mask = 0xFFC00000
  m_fill_mask = 0x00200000
  m_soft_error = 1e-7
}
approx_training(){
  for(iter in epoch){
    setPCMFormat(FP10_t)
    run_epoch()
  }
}
    
```


ex) float(32 bit) mapping / 10 bit fetching

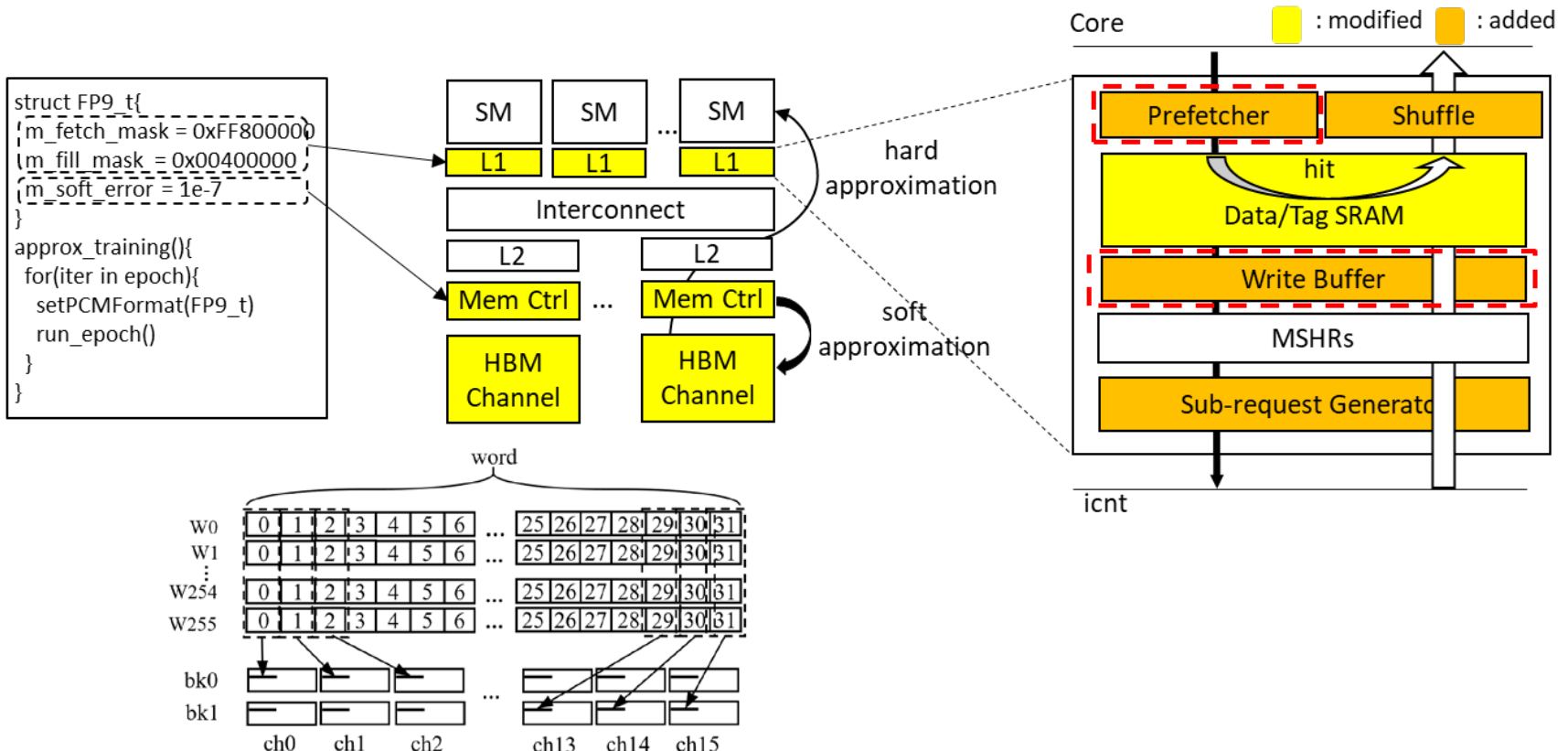
sign mantissa



Boyeal Kim et al., "PCM: Precision-Controlled Memory System for Energy Efficient Deep Neural Network Training," DATE 2020, Apr. 2020.

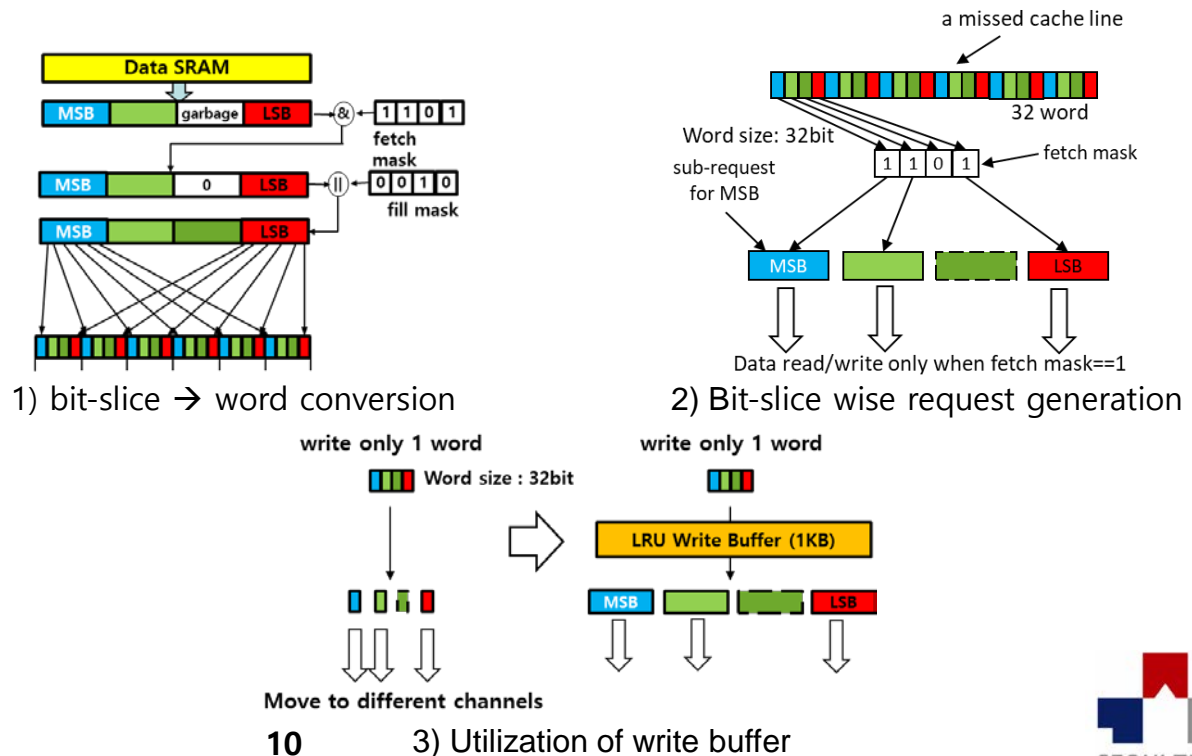
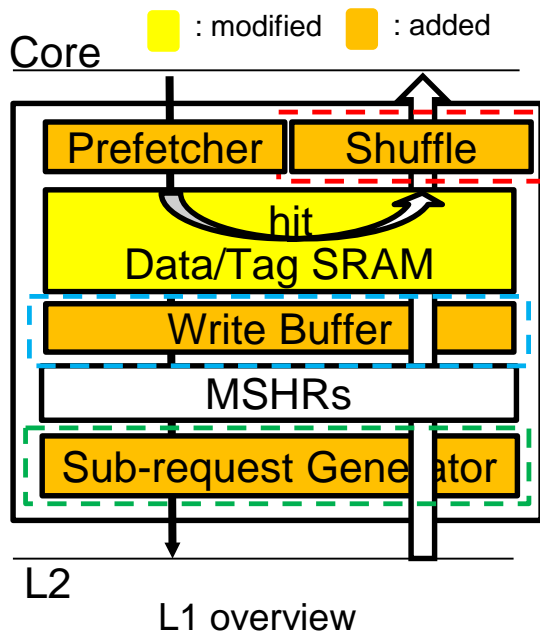


- Transposed memory based GPU architecture : Bit distribution according to bit-significance for **each channel**
- Transposed mapping supporting high bandwidth for GPU → One bit-slice per channel/bank
 - Ex) In the existing structure, 16 commands are needed to get FP16, but the proposed structure is possible with 1 command
- Bit-slice at different channel and bank → Preserve row locality
- Data transpose using L1 cache as a bridge → L1 structure modification (No need for BTU buffer)
- HW structure that enables hard approximation on SW → Compatibility with the algorithm 

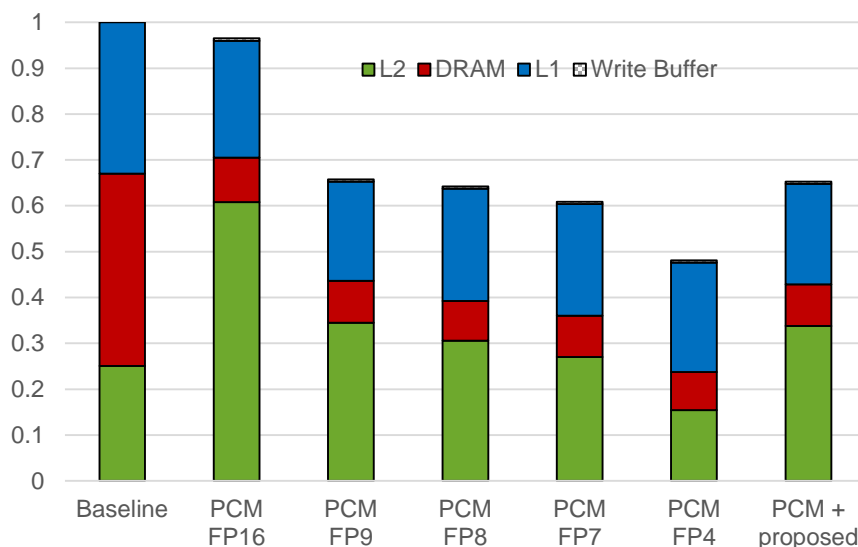


Transposed Mapping on HBM

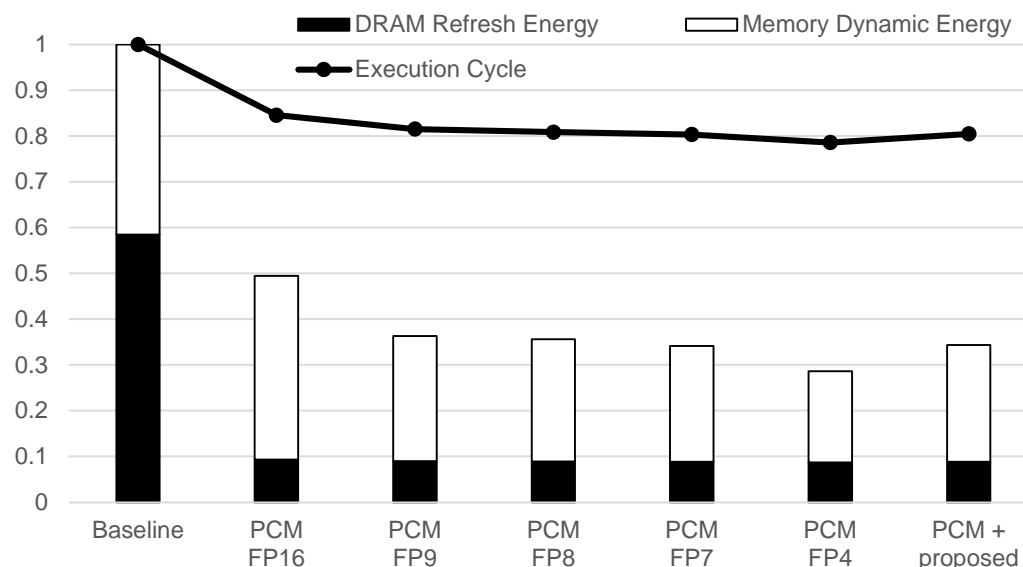
- **Modify** existing structures : Increase the cache line size, 128B \rightarrow 1KB (32bit precision x 32B)
- **Add** new structures
 - 1) **Shuffle Module** : For Read operation, restore transposed data by word unit (Gather bit-slice in a cache line & Re-arrange and fill holes before load) \rightarrow low overhead word conversion
 - 2) **Sub-request Gen.** : Generate memory request by bit-slice if cache miss (Using fetch mask)
 - 3) **Write buffer** : Correct cache miss data and send it to DRAM to increase DRAM BW utilization
- Resolve the problem of conflict increase due to the decreased number of cache lines by the increase of cache line size through **SW optimization** such as thread scheduling, thread number limit, thread flattening, prefetching, etc



- Memory dynamic energy is reduced according to the reduced number of bits**
 - 16 bit → 9 bit : 35% reduction with fixed training & 38% reduction with adaptive training (9bit→7bit)
 - L1 : Power reduction effect compared to baseline thanks to SW optimization
 - L2 : Increased power compared to baseline by increased L2 utilization due to channel distribution
 - DRAM : Significantly reduced power due to utilization of low precision and increased L2 hit ratio
- It effectively reduces the refresh energy (85%)**
- As a result, the total energy is effectively reduced (66%) → Low-power memory design!**
- Speed-up : Execution cycles are also reduced by 20%**



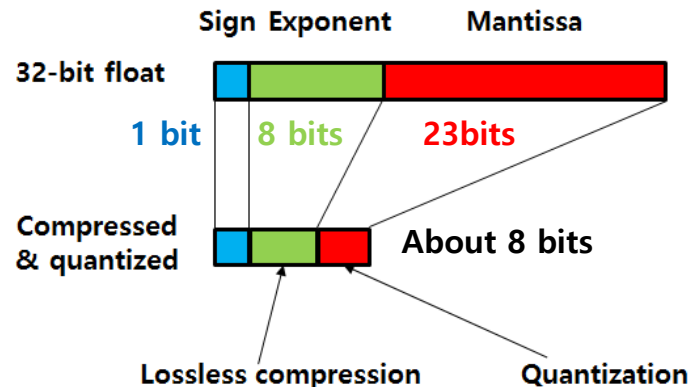
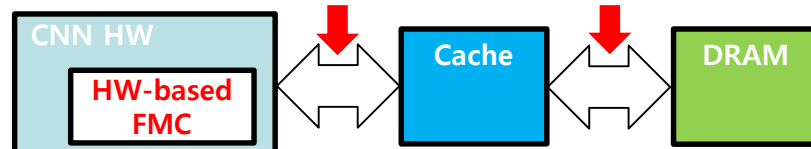
Dynamic Energy Breakdown



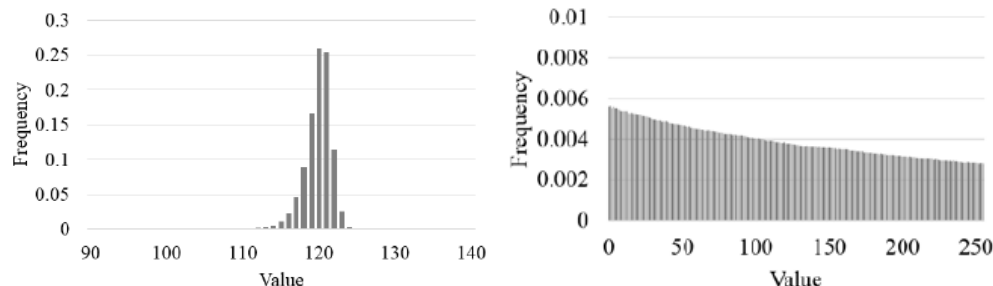
Energy and Cycle Reduction of PCM

◆ Goal : Implement **HW-based** FMC for DNN to enhance cache hit rate and to reduce mem. BW

● Concept of the proposed scheme



▷ Distribution of convolution weights for exponent & mantissa



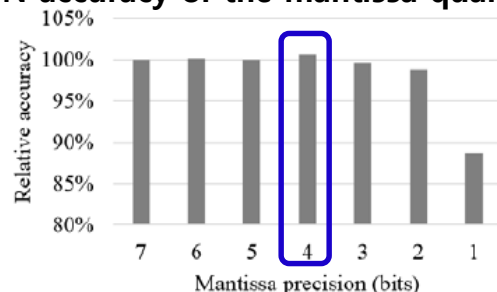
● Experimental results

Entropy & average compressed length of exponent

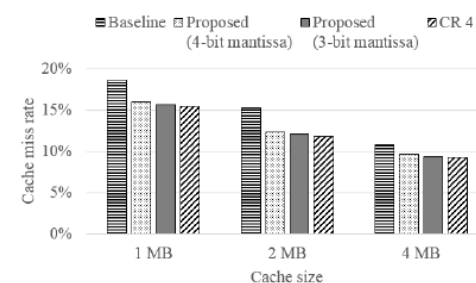
Data type	Entropy	Golomb-Rice (except 0)	w/ element-wise zero bitmap	w/ block bitmap flag
Convolution weights	2.77 bits	2.88 bits	3.88 bits	2.95 bits
Feature maps	2.50 bits	3.07 bits	2.55 bits	2.61 bits

▪ $1(\text{sign}) + 3(\text{Exponent}) + 4(\text{Mantissa}) = 8\text{bits}$

CNN accuracy of the mantissa quant.



Cache miss rate

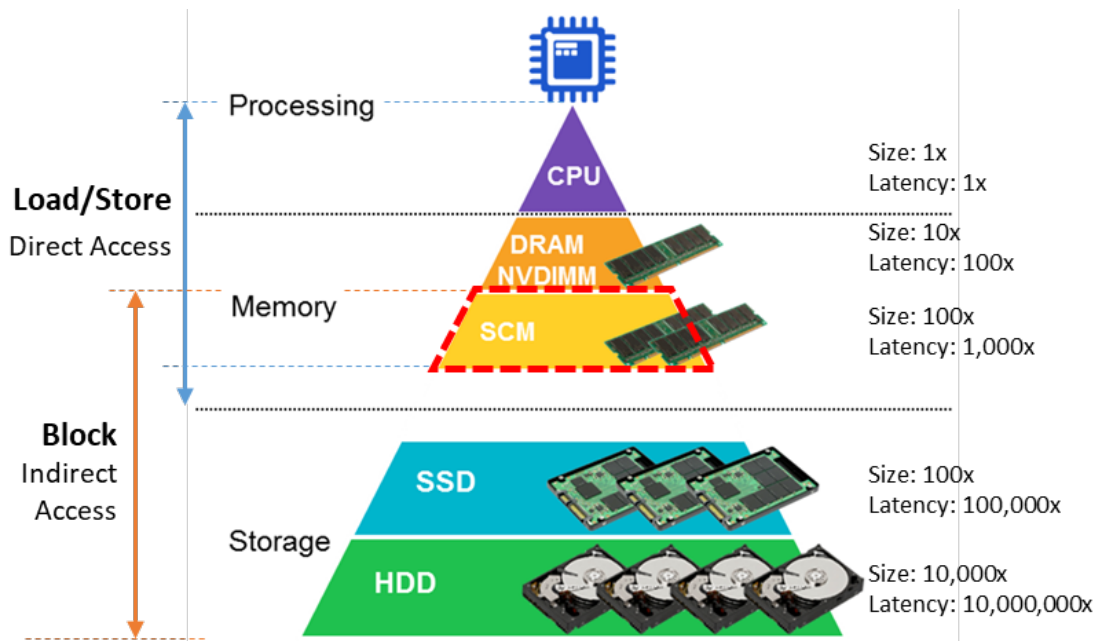


◆ Contribution

- 1) Reduce 32bits data → 8 bits data (**75% reduction**) without any accuracy drop and additional latency in HW design
- 2) Cache hit rate enhancement & External power consumption reduction w/o accuracy drop

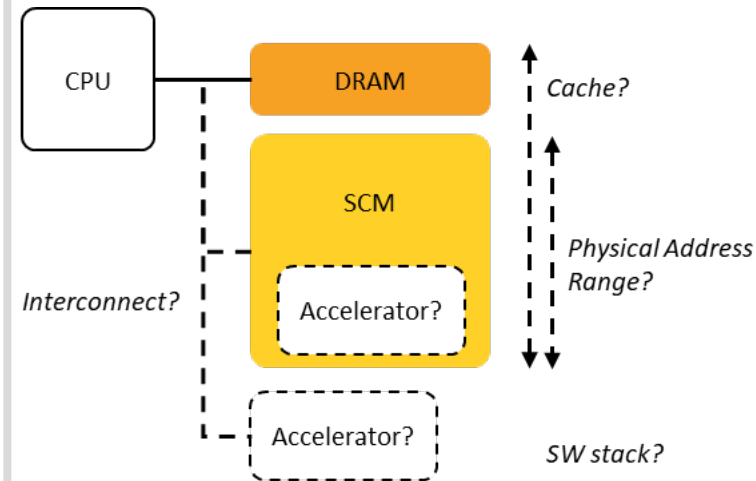
- **SCM (Storage Class Memory) could be useful in increasing the memory capacity: slower but larger, cost efficient compared to DRAM, and persistent (NVM)**
- **SCM utilization models are actively being researched including the application in the Near Data Processing structure**

Memory Hierarchy



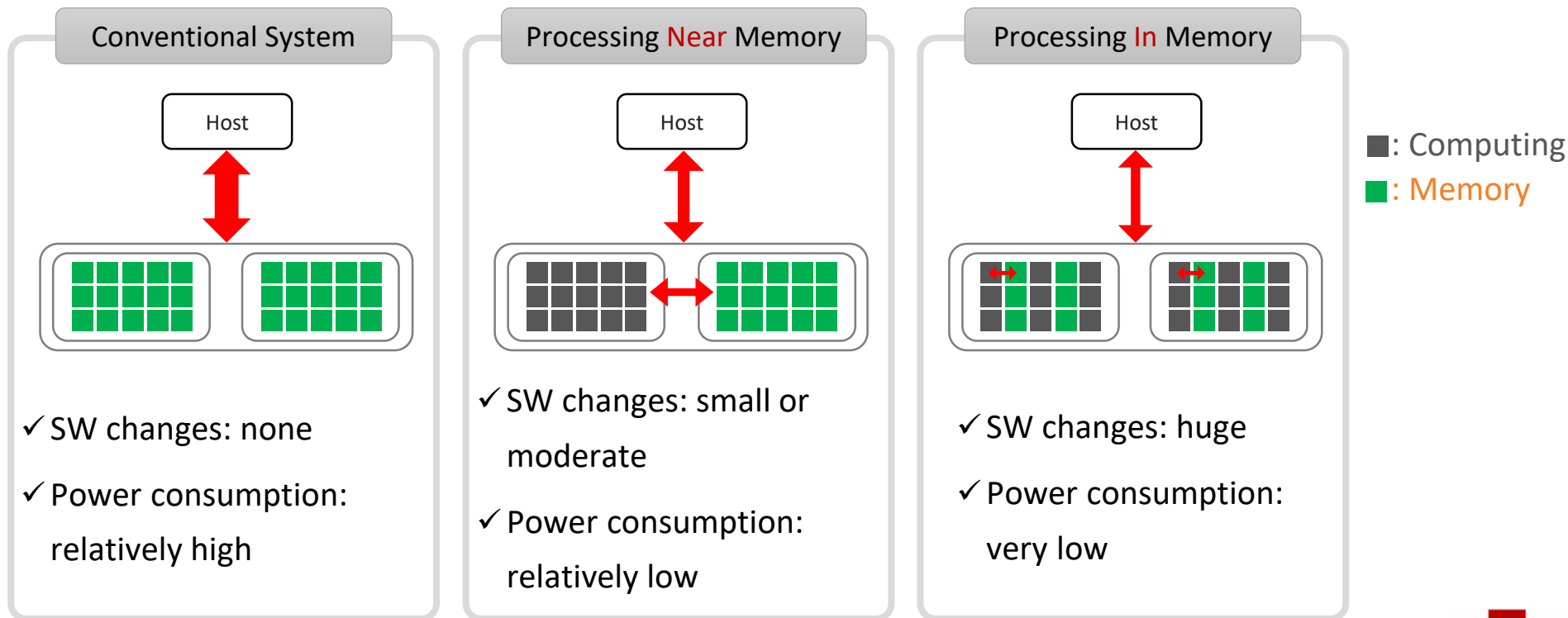
Source: Rambus

SCM Utilization Model



Near Data Processing at Memory Level

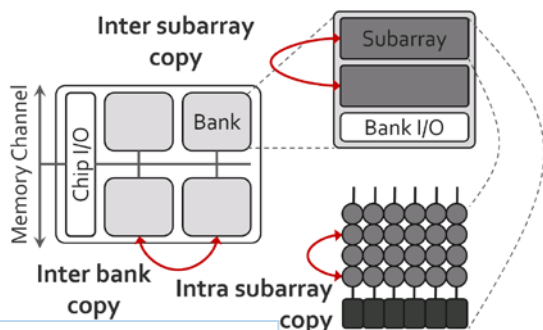
- **PIM (Processing In Memory), PNM (Processing Near memory)**
 - **Pros: Overcome bandwidth limitation between logic and memory devices with lower power consumption**
 - **Cons: Not backward compatible with legacy software stack, requiring some changes in system software**



- PIM (Processing In Memory), PNM (Processing Near memory)**
 - Pros: Overcome bandwidth limitation between logic and memory devices with lower power consumption
 - Cons: Not backward compatible with legacy software stack, requiring some changes in system software

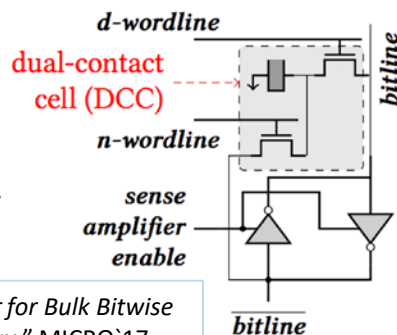
PIM

- Row data copy
e.g.) Bulk copy



Seshadri et al., "RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization," MICRO`13

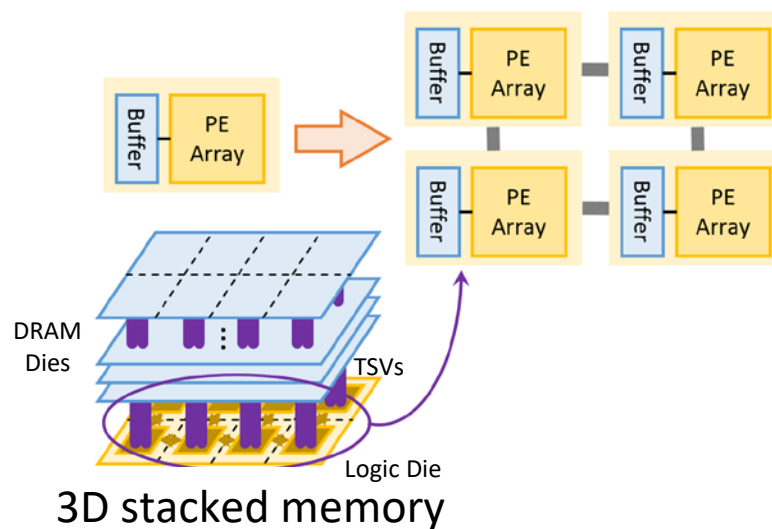
- Bulk bitwise operations
e.g.) In-DRAM bit-wise NOT



Seshadri et al., "Ambit: In-Memory Accelerator for Bulk Bitwise Operations using Commodity DRAM Technology," MICRO`17

PNM

- 3D stacked Memory
e.g.) NN accelerator



3D stacked memory

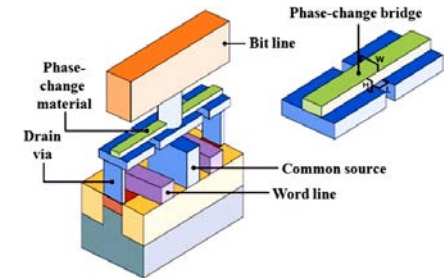
M. Gao et al., "TETRIS: Scalable and Efficient Neural Network Acceleration with 3D Memory," ASPLOS`17

	Emerging Memory			Established Memory	
	STTMRAM	PCMS "3D Xpoint"	RRAM	DRAM	Flash NAND
Non-Volatile	YES	YES	YES	NO	YES
Endurance (Nb cycles)	High (10^{12})	Medium (10^8)	Low (10^6)	High (10^{15})	Low (10^5)
2016 latest technological node produced (nm)	40 nm	20 nm	130 nm	1X nm	15 nm
Cell size (cell size in F ²)	Medium (6-12)	?	Medium (6-12)	Small (6-10)	Very small (4)
Read latency (ns)	Fast (10-20 ns)	Fast (50-100 ns)	Medium (250 ns)	Very fast (few ns)	Slow (100,000 ns)
Power consumption	Medium (50 pJ/bit)	Medium	Medium (6nJ/bit)	Low	Very high
2016 price (\$/Gb)	High (\$3000-\$200/Gb)	Low (\$ < 0.5/Gb)	High (\$100/Gb)	Low (< \$1/Gb)	Very low (\$ < 0.05/Gb)
Suppliers	Everspin	Micron/Intel	Adesto	Samsung, Micron, SK Hynix	Samsung, Micron, Toshiba, SK Hynix, Intel

New Memory for CNN : PCRAM

◆ Phase-Change Memory : Next-generation non-volatile random-access memory

- Performance gap between DRAM & NAND
 - It happens due to different design requirements
 - DRAM for main memory : low latency & high throughput
 - NAND Flash for storage : low cost/GB & high density & non-volatile

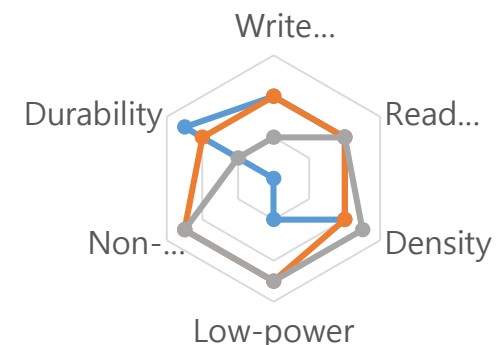


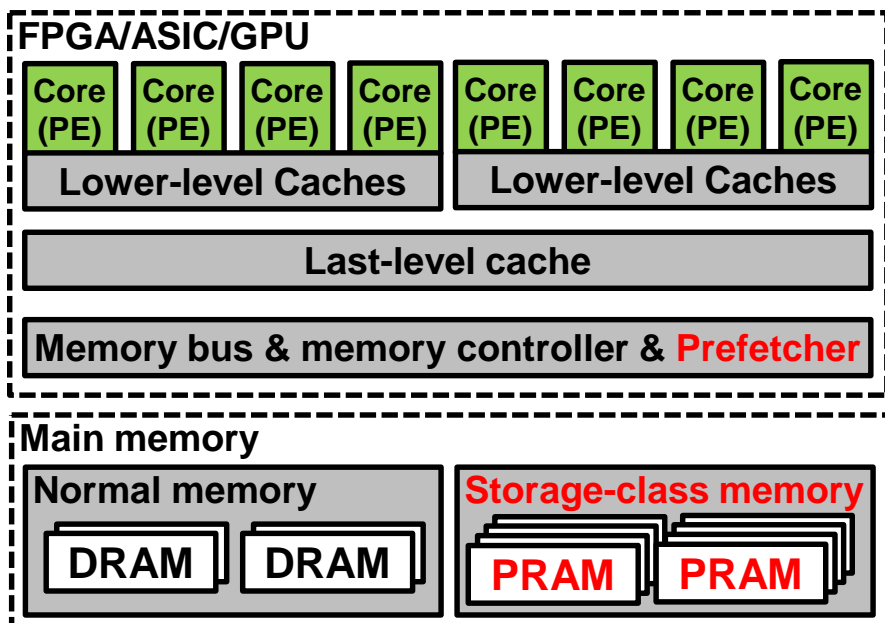
→ PRAM is performance **gap filler between DRAM & NAND**

- Advantages:** Faster than NAND & higher density than DRAM & non-volatile & low P_{IDLE}
- Because it has high density with low cost (5x times cheaper than DRAM), it is often used in data centers that require a main memory with large capacity
- Disadvantages:** **Endurance** + **Reliability** problems & low bandwidth & Large write energy
- It is expected to replace DRAM, but there is a problem of durability limitation
 - Various lifetime enhancement schemes for NAND Flash can be applied
 - Durability problems in PCRAM can be solved by using with DRAM which has almost infinite durability

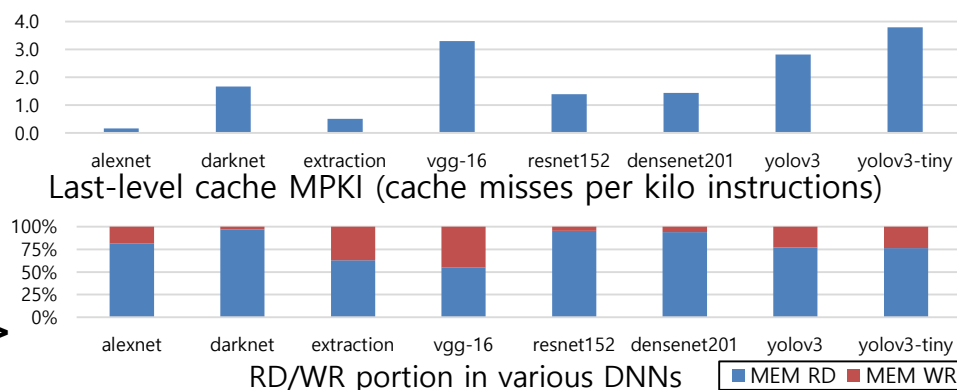
Items	DRAM	PRAM	NAND Flash
Read/Write Access Time	10/10 ns	20/100 ns	> 25/300 us
Cell Size	6-10 F ²	6-12 F ²	5 F ²
Cycling	10 ¹⁶	10 ¹⁰	10 ⁵
Retention	64 ms	10 year	1 year
Random Access	○	○	X
Non-volatile	X	○	○

—●— DRAM —●— PRAM —●— NAND Flash





Items	DRAM	PRAM	NAND Flash
RD/WR Time	10/10 ns	20/100 ns	> 25/300 us
Idle Power	100mW/GB	1mW/GB	1mW/GB
Cycling	10^{16}	10^{10}	10^5
Retention	64 ms	10 year	1 year
Random Access	O	O	X
Non-volatile	X	O	O



<PRAM-based memory platform for mobile inference>

◆ Motivation

- **PRAM** (Phase change memory) has good features to construct a **low power architecture** for CNN inference because P_{idle} of PRAM is 100x lower than DRAM, and PRAM has a **higher density** than DRAM and is **non-volatile**
- WR of PRAM is 10x slower than that of DRAM, but RD of PRAM is only 2x slower than that of DRAM
- **CNN inference** is **computation-intensive**, has **high spatial locality**, and its commands consist mostly of **RDs**

◆ **Challenging point:** Slowdown of CNN inference + PRAM reliability problems (Disturbance errors and endurance)

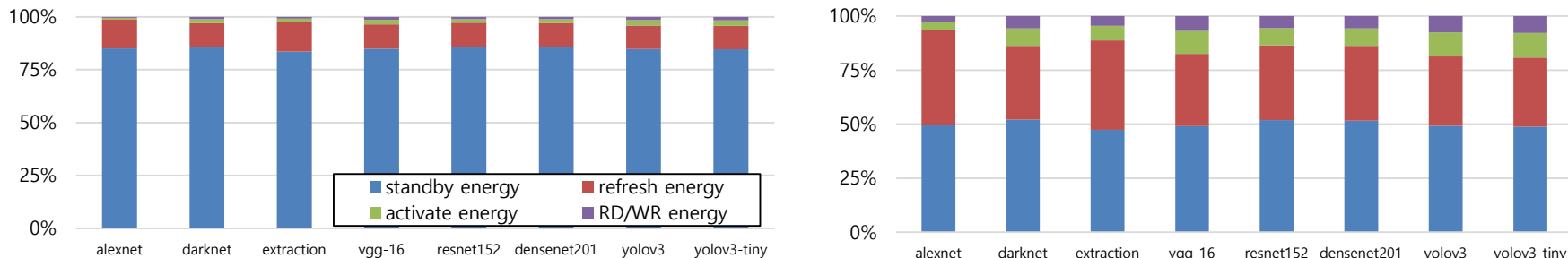
◆ **Solution:** **PCM-based low-power memory dedicated to CNN inference** by **prefetcher** and **reliability schemes**

- Development of HW **prefetcher** considering CNN memory access → Prevent slowdown by hiding memory latency
- Development of **PRAM reliability solution** optimized for CNN memory access patterns

◆ **Contribution:** **Up to 50% of memory power reduction** is expected without slowing down in CNN applications

Energy breakdown of DRAM/LPDDR in CNN applications

- DRAM: Since LLC MPKI is quite low, standby power occupies about 80%
- LP: RD/WR proportion slightly increases, but standby/refresh proportion still stand out
- These results mean that it is efficient to apply PCM to CNN applications

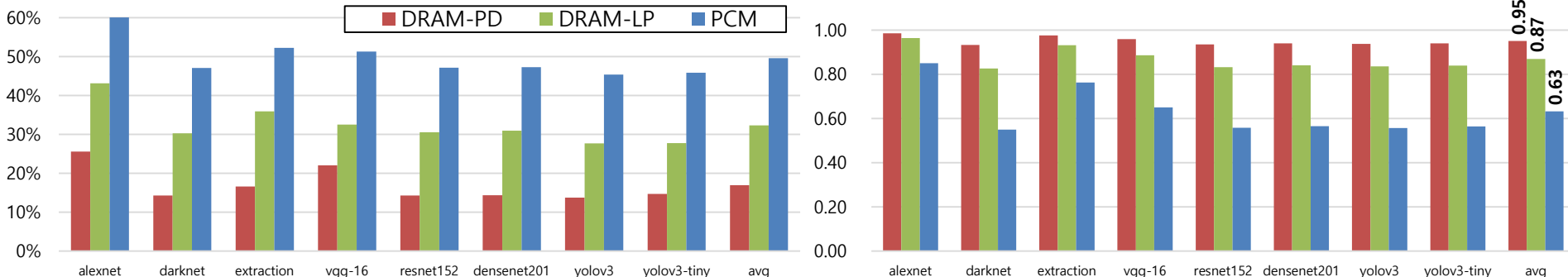


Energy breakdown in CNN applications: DRAM (Left), LPDDR (Right)

Energy & IPC evaluation of each memory device

- Energy: On average, **49.6% and 25.4% reduction** compared to DRAM and LPDDR
- Normalized IPC: On average, **36% and 27.3% increase** compared to DRAM and LPDDR

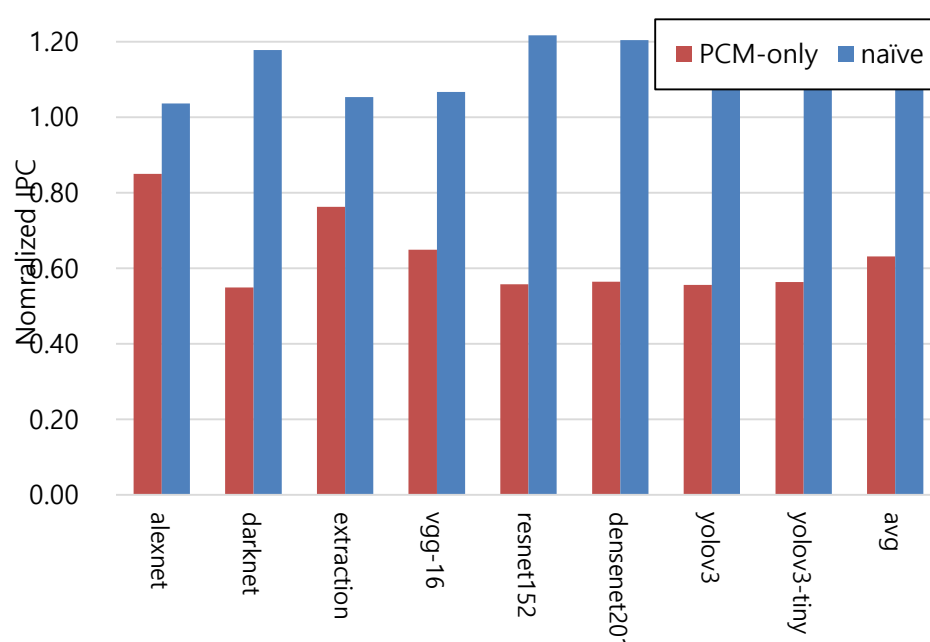
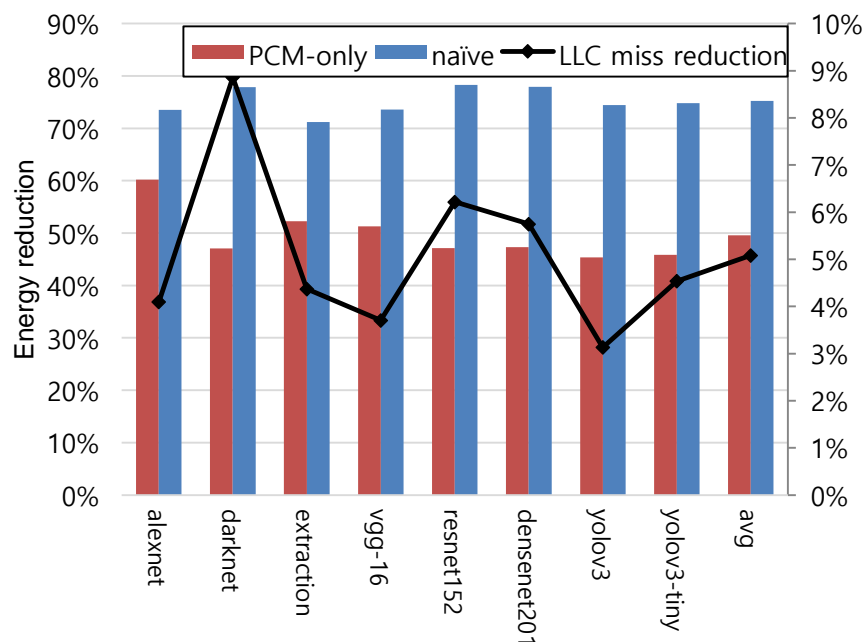
→ Necessity of **prefetch algorithm** for speed-up!



Performance evaluation of each memory: Energy reduction (Left) Normalized IPC (Right)

◆ Possibility verification through Naïve prefetcher

- A method of reading next four data of the input request address together
- Energy consumption is reduced by 25% and 75% compared to PCM-only and baseline: Prefetcher reduces additional LLC miss by 5%, reducing more energy
- IPC is improved by 50% and 12% compared to PCM-only and baseline
- Remarkable IPC improvement can be expected when prefetcher is applied, and more performance enhancement can be expected if the processor provides the filter size and feature map size of CNN as prior knowledge



Performance evaluation by naïve prefetcher: Energy reduction (Left) Normalized IPC (Right)

◆ Reliability

- Bit-cell value changes due to various reasons

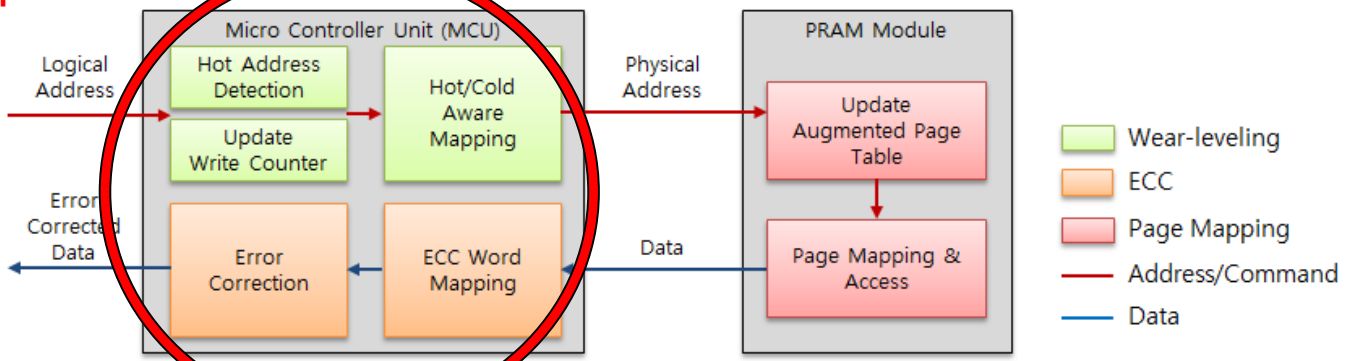
	Condition	Position	Bit change	Value	Recovery
Write Disturbance	Write	Adjacent cells	0→1	10k-100k	Rewrite
Read Disturbance	Read	Accessed cell	1→0	100k-	Rewrite
R-Drift	Time lapse	All cells	1→0	1000s-2000s	Recovery
Retention	High temperature /Time lapse	All cells	0→1	40°C/5 years 85°C/2 weeks	Rewrite

◆ Endurance

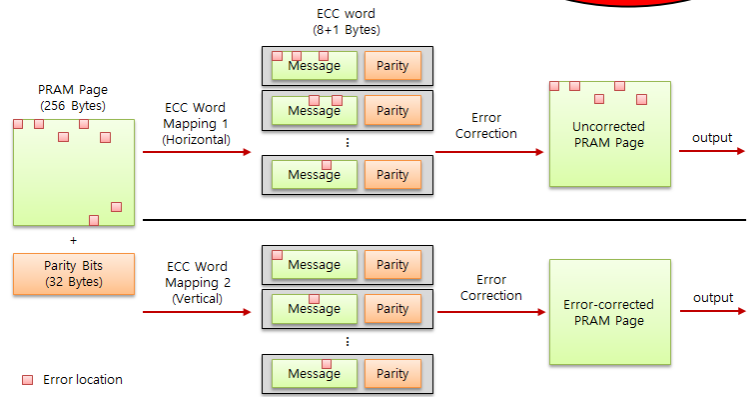
- Wear-out : Cell destruction due to too many accesses

- ◆ Goal : Improving PRAM reliability & durability
 - PCRAM : Faster than NAND flash & higher density than DRAM & non-volatile
 - Implement Wear-leveling, ECC, Page mapping schemes on **digital circuit design** of mem. controller

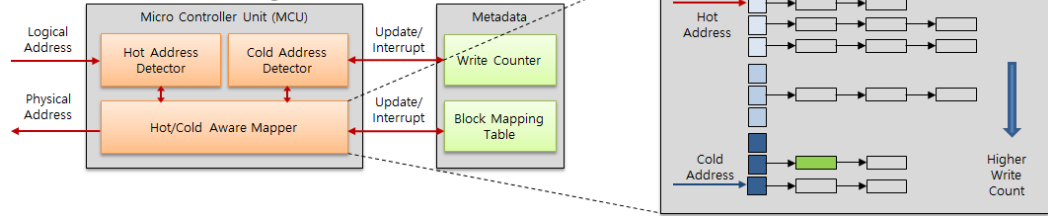
HW implementation



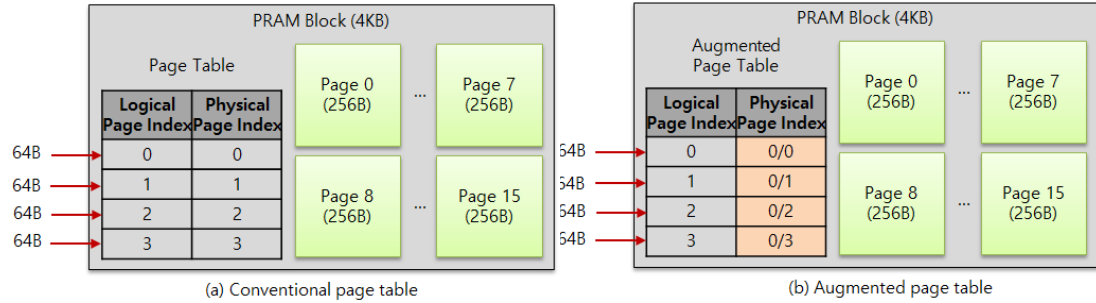
● **ECC**



● **Wear-leveling**



● **Page mapping : Read-Modify-Write**



H. Lee, M. Kim, H. Kim, H. Kim and H. Lee, "Integration and Boost of a Read-Modify-Write Module in Phase Change Memory System," in IEEE Transactions on Computers, vol. 68, no. 12, pp. 1772-1784, Dec. 2019. link: <https://ieeexplore.ieee.org/document/8792091>

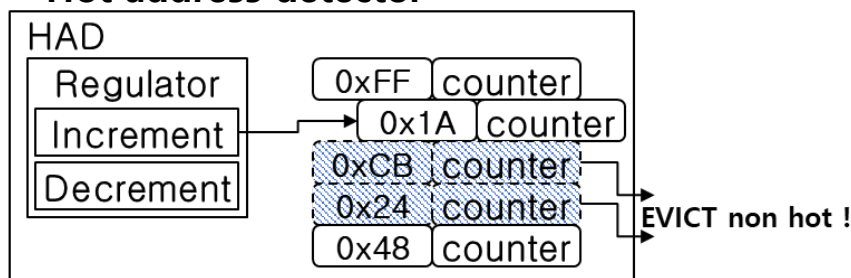
◆ Background

- Periodic operation of static wear leveling (*e.g.*, start gap) → Swapping overhead
- Remapping logically hot address to physically cold address is IDEAL wear leveling
 - Requires write counter for each address → Large area overhead

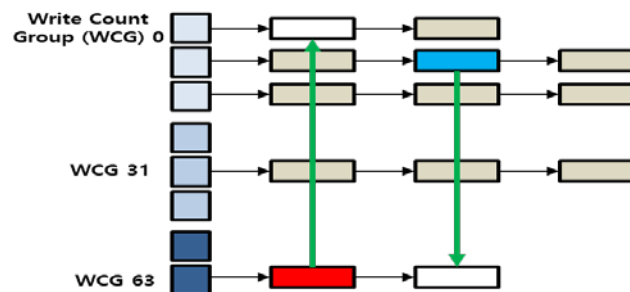
◆ Hot/Cold-aware wear leveling

- Detects hot data and cold block address with **hot address detector (HAD)**
- Write counts of block addresses are managed as a moving window (**On-demand method** like in Rejuvenator, Block-level WL) & Start gap is used for page-level WL

● Hot address detector



● Rejuvenator



◆ Simulation results: Swap overhead <0.1%, normalized lifetime >90%

- HAD+WL provides higher performance than the previous SOTA, start-gap (97% NL & 30% Swap overhead)
- Periodic WL (*i.e.*, WL for every 1024 writes) yields **negligible swap overhead** and **high lifetime** performance

	HAD + WL	Periodic WL (HAD+WL)
Normalized lifetime (%)	98.5	92.8
Swap overhead (%)	1.085	0.0098

◆ Background

- Large page size is required (>128B) in PRAM system for enhancing throughput
- Prior works in RMW assume: cache line size in the host = PRAM page size
- General processors have 64B cache line → Read-Modify-Write (RMW) is required between PRAM sub-system & host system

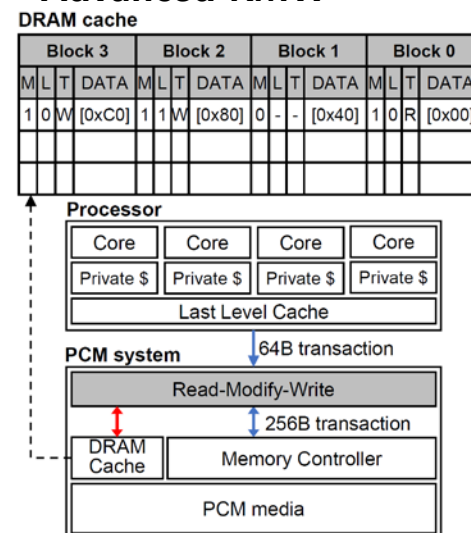
◆ Basic RMW

- Read page data for every write operations
- Partially update page data and write back

◆ Advanced RMW

- Leveraging **existing data cache** in a PRAM module to respond data that are prefetched with page-wise operation
- **Merging** consecutive commands **as one command** by leveraging data cache to reduce redundant RD operations

● Advanced RMW



◆ 256B is selected as appropriate page size considering both read disturbance, speedup, and energy consumption

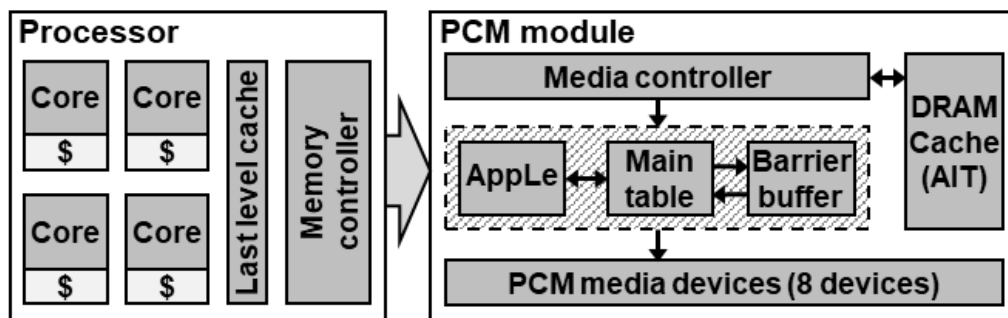
- Baseline: 64B PRAM system without RMW
- 2.88x speedup (normalized cycles) → **65.3% redundant RD operations are reduced**
- **54% reduction of read disturbance** thanks to reduction of redundant RD operations
- Negligible energy overhead (4%) within total computer system when considering speedup and read disturbance reduction

◆ Motivation

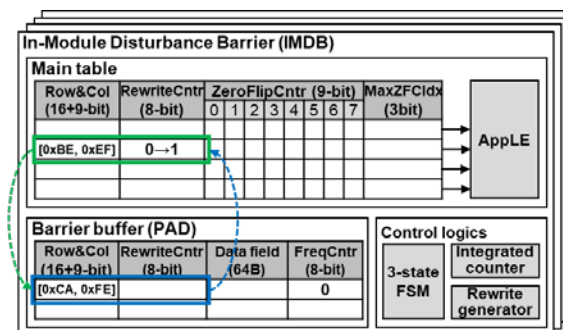
- Verify-and-Correction (VnC)-based method that accompanies at least four RD operations cannot completely eliminate verification RD, resulting in performance degradation
- Previously, there was no scheme to reduce WDE by **utilizing the WD limitation number**

◆ In-module disturbance barrier (IMDB)

- Add IMDB module within PCM system to manage WD-vulnerable write patterns (1-to-0 flip) → Record the number of 1-to-0 flips in the data word
- IMDB is located between the media controller and PCM media devices
- Record only addresses in the main table** → Reduce the burden on supercapacitor constraints
 - If the number of 1-to-0 flips exceeds the threshold set based on the WD limitation number, it is determined that WDE will occur soon in the surrounding address and therefore, rewrite (=restore) operation is performed → **Reduction of verification read** is possible
 - Restored addresses are stored with data in the **barrier buffer** to prevent additional WDEs
 - Approximate lowest number estimator(AppLE)**: Sampling-based method that defines multiple entries as a group → Reduce Area & Energy overhead



Structure of IMDB



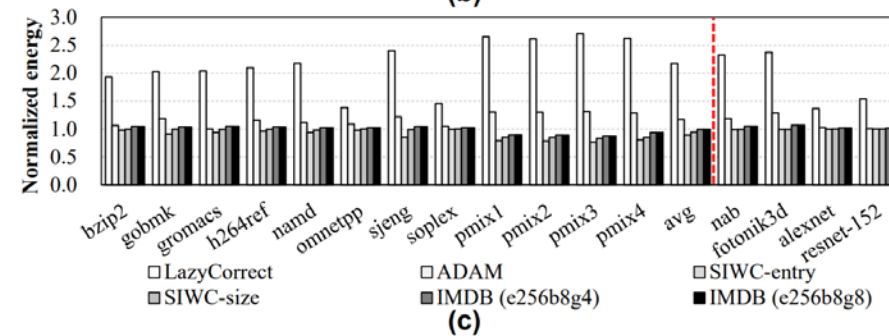
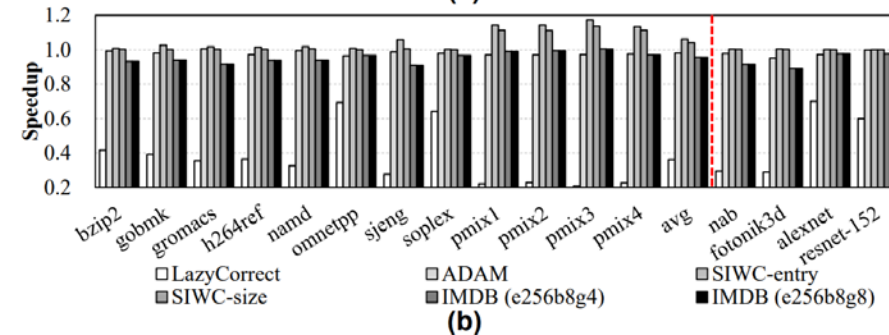
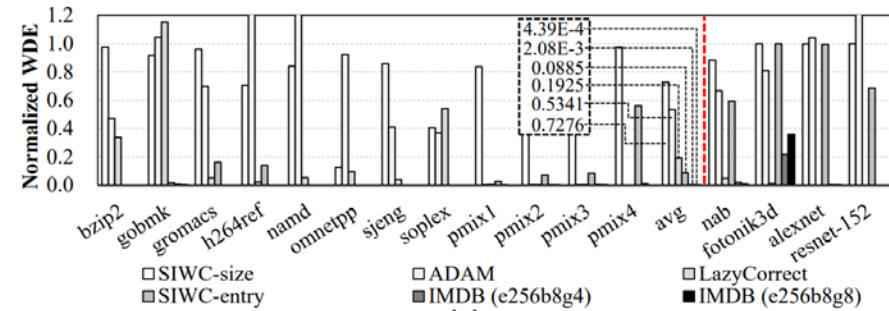
◆ Write disturbance error is reduced by up to 1218x more compared to the previous studies

- In terms of absolute WDE numbers, the average number of WDEs in the previous study (SIWC-entry) was 297.9, so it is not overkill
- Expect to reduce WDEs even more when used with ADAM with on-the-fly compression: **High compatibility**

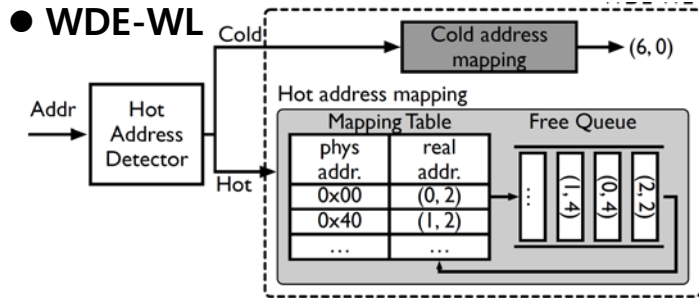
◆ In case of speedup, there is some performance degradation (~4%)

- It is not a big disadvantage as it is similar to the baseline (no scheme applied) while greatly reducing WDE
- The proposed method shows much better performance than LazyCorrect

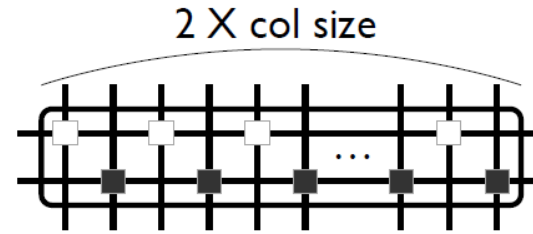
◆ In case of energy, energy overhead occupied by SRAM is negligible: 0.007% of total energy



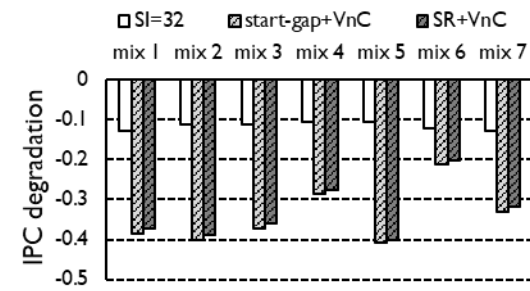
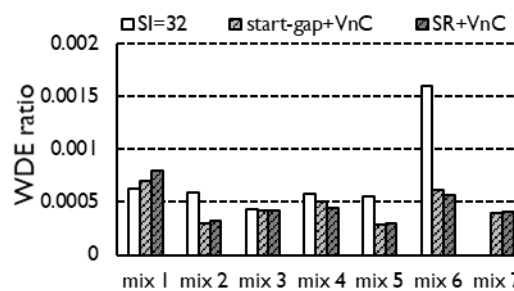
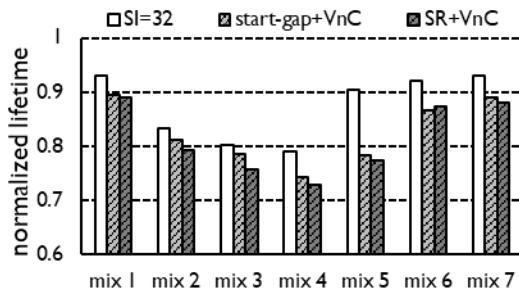
- ◆ Background: Wear leveling (WL) can distribute write operation for maximizing cell endurance and lowering write disturbance errors (WDE)
- ◆ Wear leveling considering both WDE and lifetime is required!
- ◆ WDE-WL: Hot addresses are managed in **hot region**
 - Hot addresses are filtered by HAD and **dynamic mapping** is applied to hot region using mapping table ↔ Cold addresses are managed by simple linear mapping for minimizing HW resources
 - Hot region stores cells in **“checker board”** manner to prevent WDEs



● **Bit mapping in hot region**



- ◆ WDE is reduced to less than 0.2% (start gap: 40%, security refresh: 30%)
- ◆ Up to 8% of normalized lifetime improvement & Up to 27% of IPC degradation reduction compared to the combination of existing WDE and WL solutions



- ◆ Background: Long simulation time is required on software-based PRAM sub-system simulators (*e.g.*, NVMain)
- ◆ FPGA with ARM sub-system can be implemented as cycle-exact PRAM emulation environment
 - ARM sub-system acts as block-level wear leveling module
 - DRAM with latency tuner is wrapped as a virtual PRAM
- ◆ Commands extracted from gem5 are transferred through USB interface
- ◆ 11.9x faster than cycle-accurate NVMain with same configuration
- ◆ 2.54x faster than in-house simulator without cycle-accurate mechanism

● PRAM emulator on FPGA

