EE382N (20): Computer Architecture - Parallelism and Locality
Fall 2011

Lecture 23 – Memory Systems

Mattan Erez

The University of Texas at Austin

Outline

- DRAM technology
- DRAM organization and mechanism
- Memory system organization and design option
- New emerging trends

- Most slides courtesy Jung Ho Ahn, SNU

DRAM is Expensive

- $/bit is almost nothing
  - $2 x 10^{-9}$

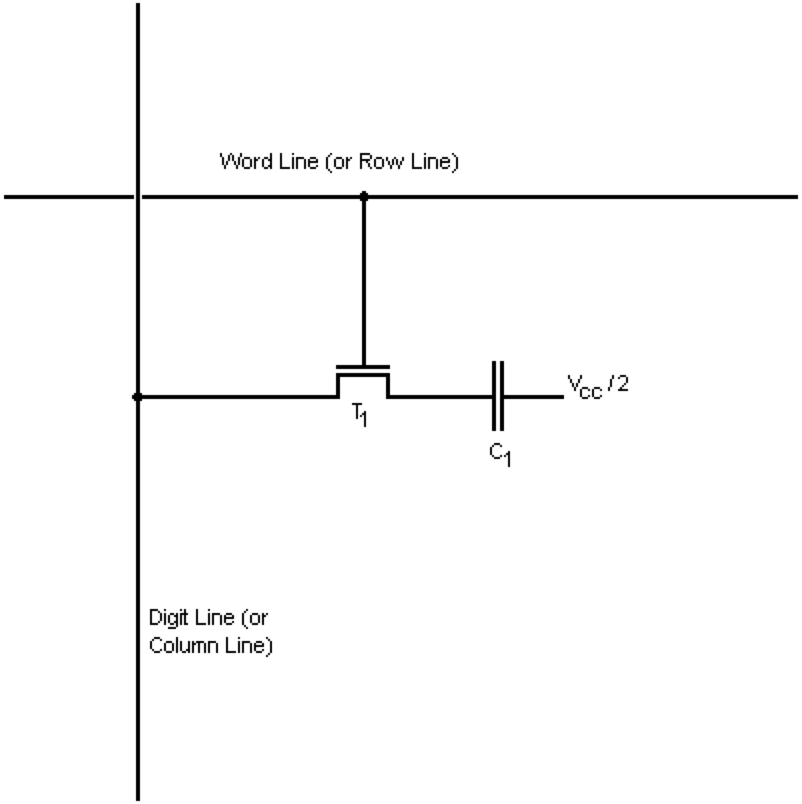
- Memory in system is expensive
  - %10 – 50 of system cost

- Rule of thumb – the more memory the better

Minimize $/bit

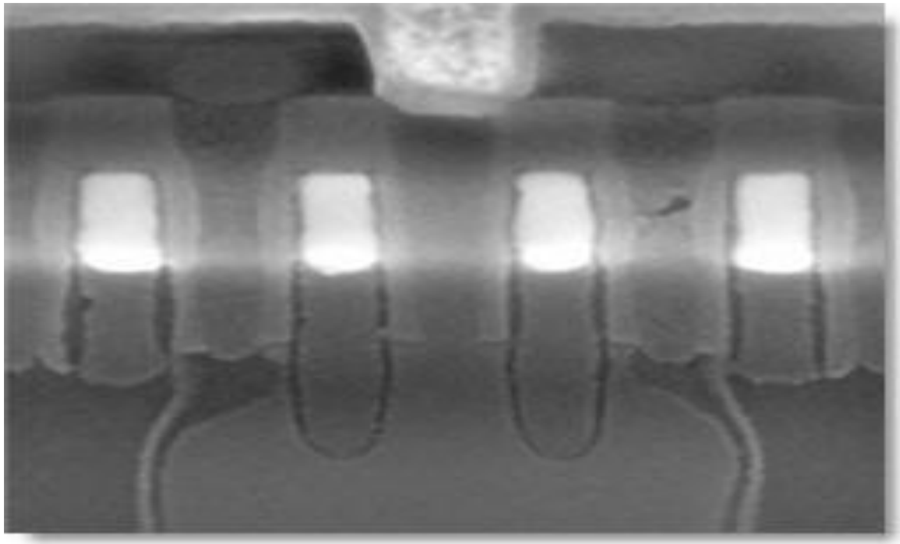
What is DRAM?

DRAM Cell

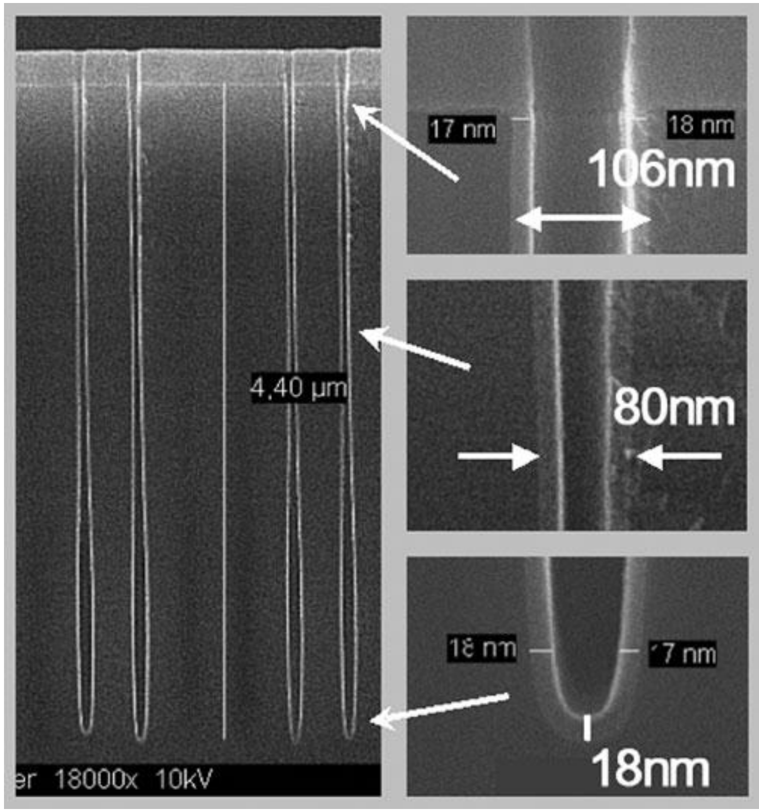


DRAM Cell

- Capacity → density → 3D
  - Recessed Channel Array Transistor (capacitor on top)
    - Samsung, Hynix, Elpida, Micron
  - Trench capacitor
    - Infineon, Nanya, ProMos, Winbond
    - IBM, Toshiba in embedded-DRAM

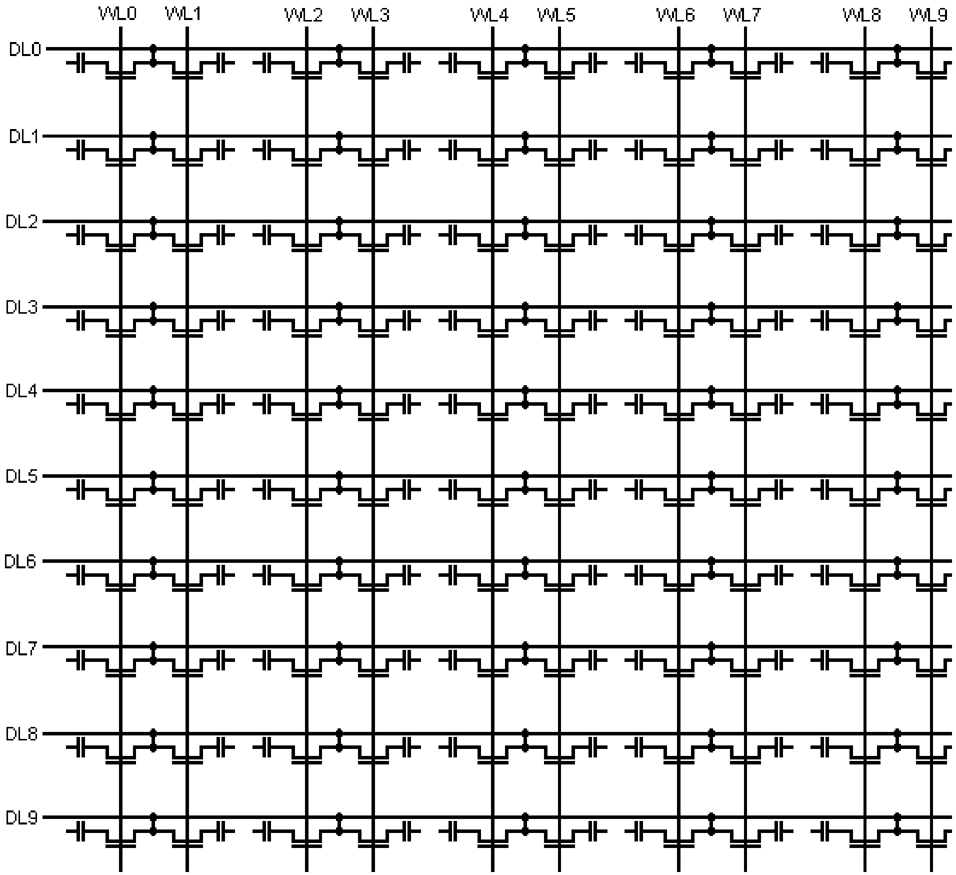


Hynix 80nm

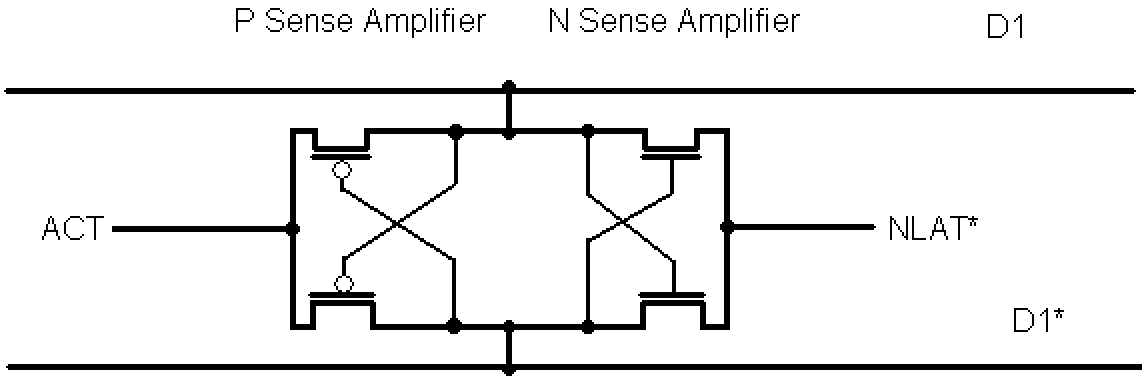


Infineon 80nm

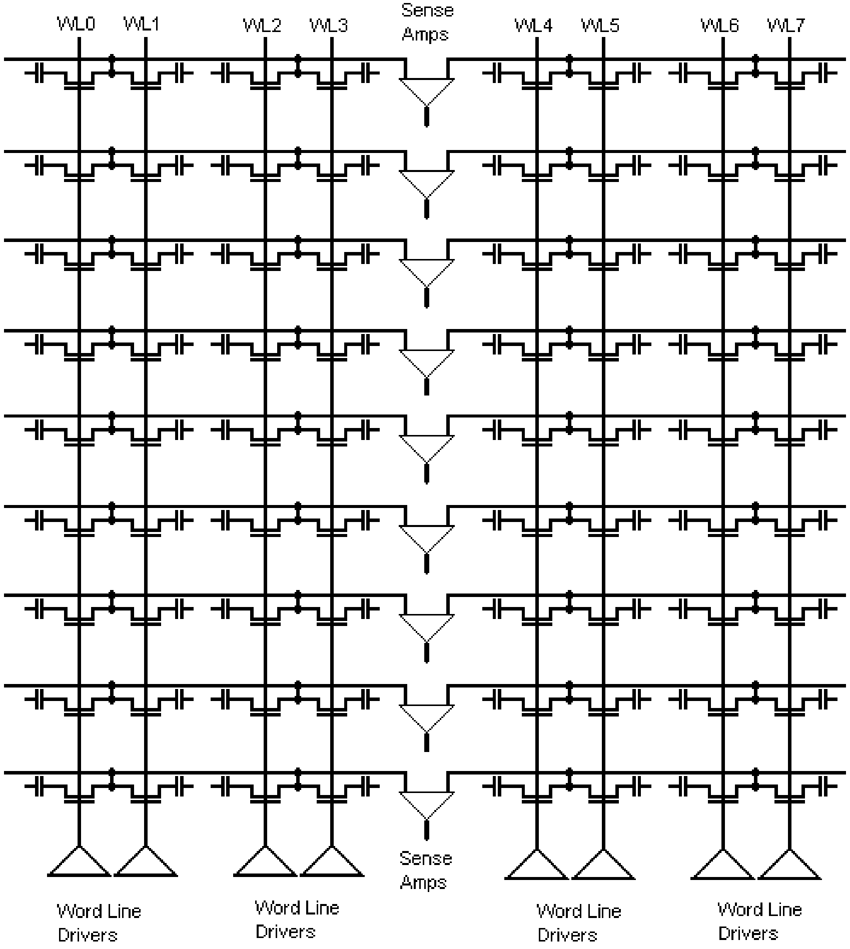
DRAM Array


WL0
WL1
WL2
WL3
WL4
WL5
WL6
WL7
WL8
WL9
DL0
DL1
DL2
DL3
DL4
DL5
DL6
DL7
DL8
DL9

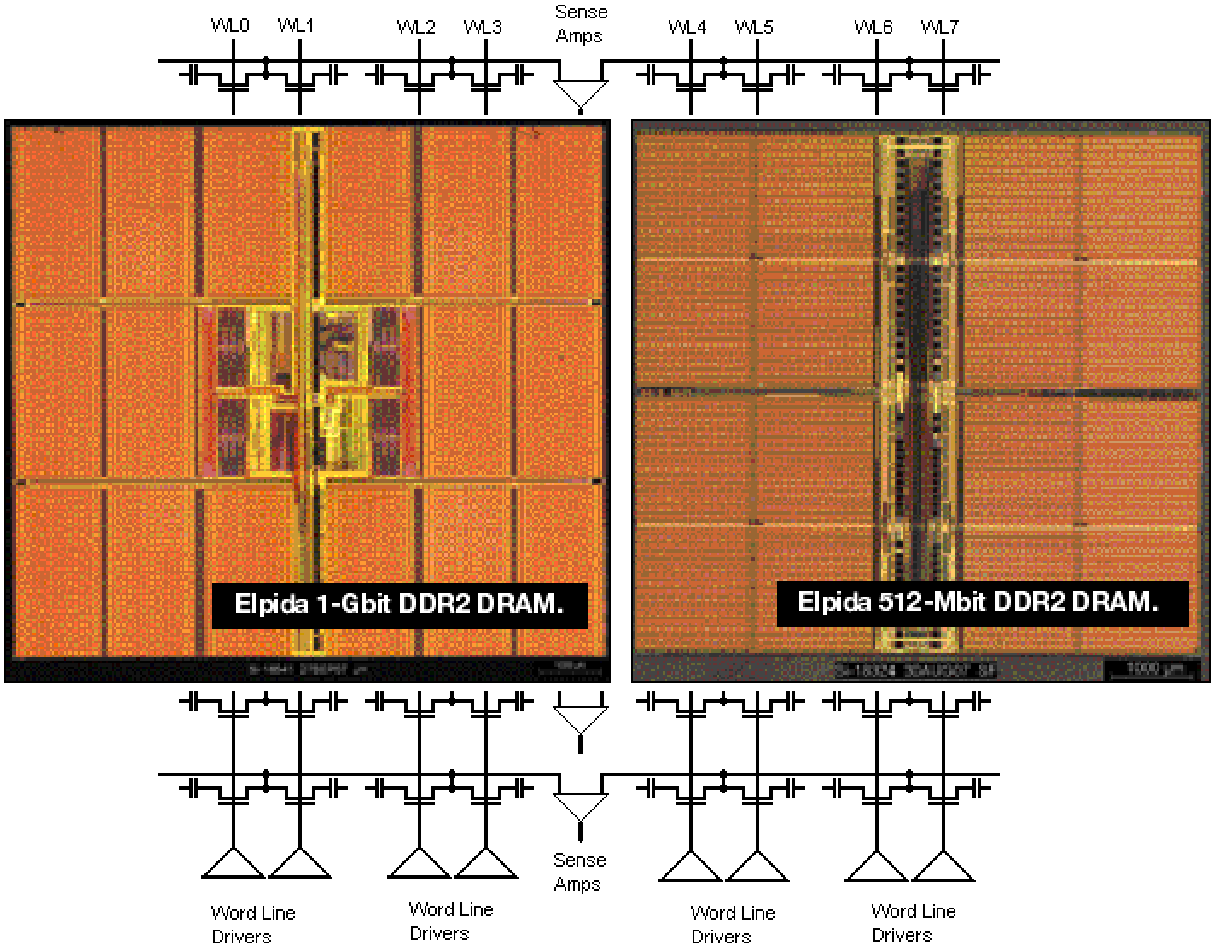
DRAM Sense Amplifier



DRAM Array


WL0
WL1
WL2
WL3
Sense Amps
WL4
WL5
WL6
WL7
Sense Amps
Word Line Drivers
Word Line Drivers
Word Line Drivers
Word Line Drivers

DRAM Array


WL0
WL1
WL2
WL3
Sense Amps
WL4
WL5
WL6
WL7
Elpida 1-Gbit DDR2 DRAM.
Elpida 512-Mbit DDR2 DRAM.
Word Line Drivers
Word Line Drivers
Sense Amps
Word Line Drivers
Word Line Drivers

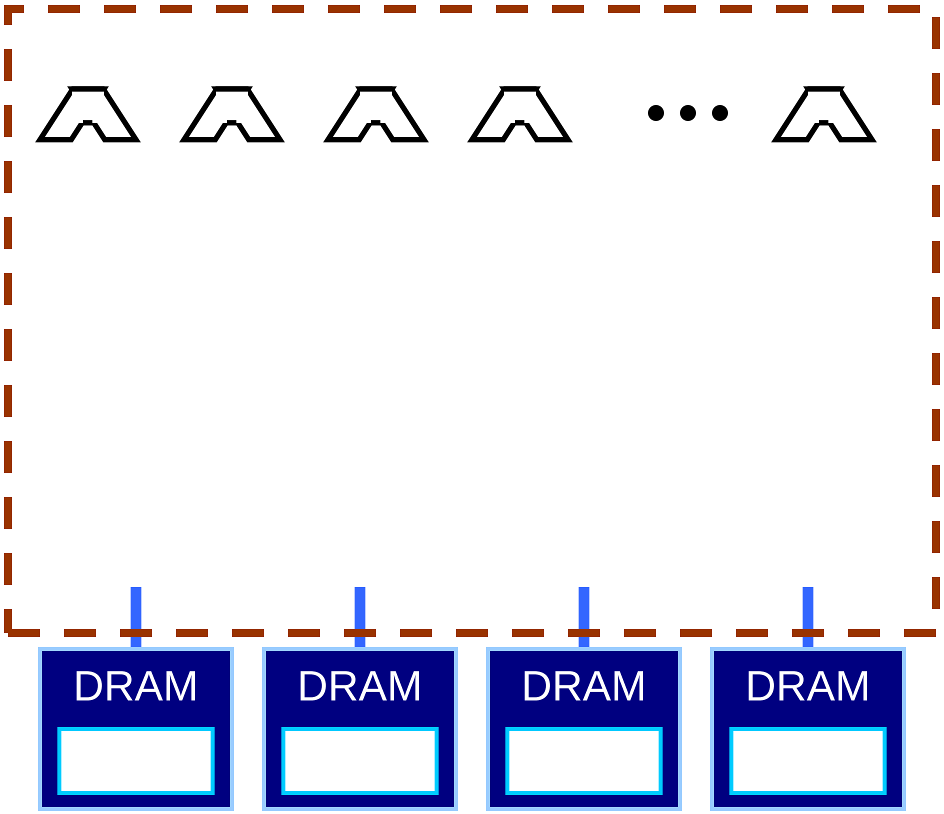
DRAM Optimization

- The more memory the better → **optimize capacity**
- Also need to worry about power and drivers

- Result is compromise in latency
  - Small capacitors and large sub-arrays increase access time

- What about bandwidth?
  - Bandwidth is expensive:
    - $.05 - $.10 per package pin
    - DDR2 requires 80 pins
- Secondary goal is optimizing BW/pin

Massively parallel processor architecture

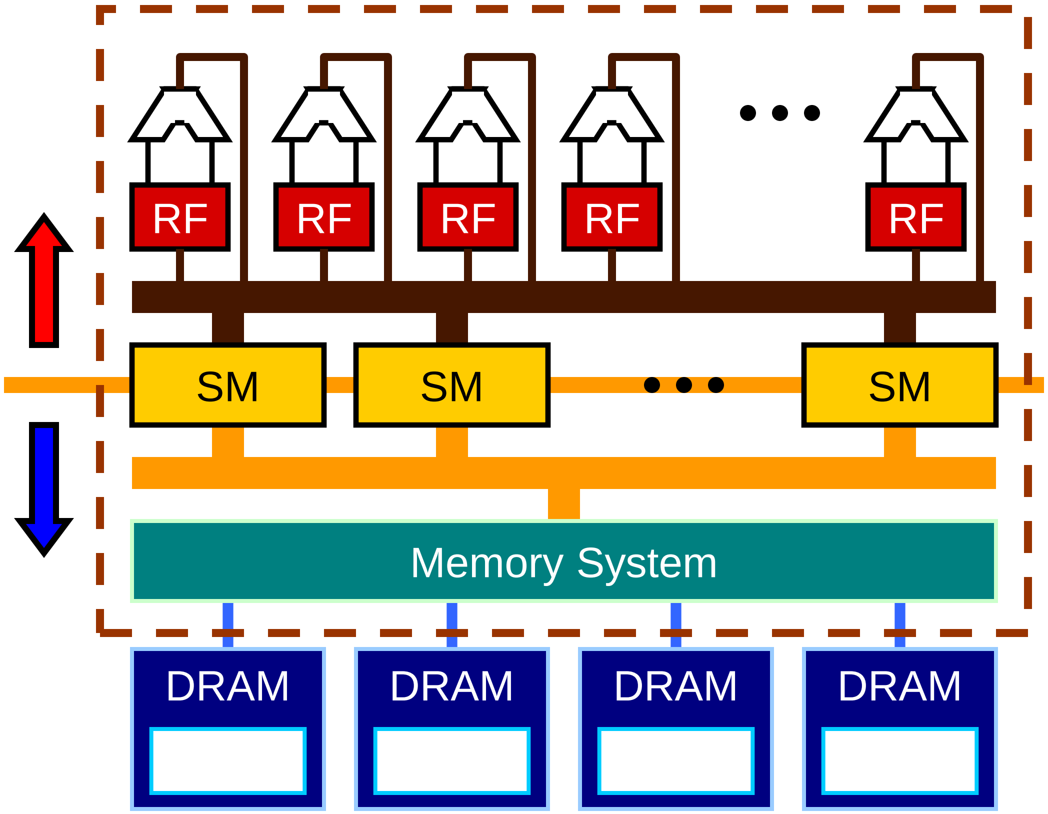


> 4000 GB/s

< 200 GB/s

- BW **demand** of ALUs >> BW **supply** from DRAMs

Stream processor architecture

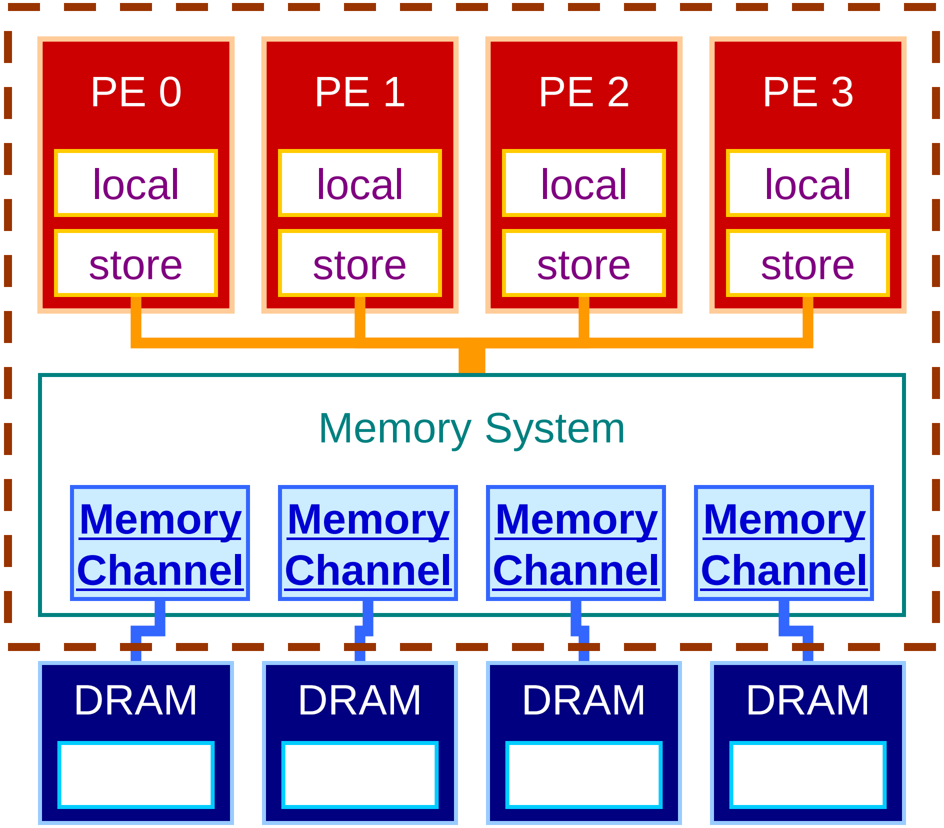


LRF : local register file

SRF : stream register file

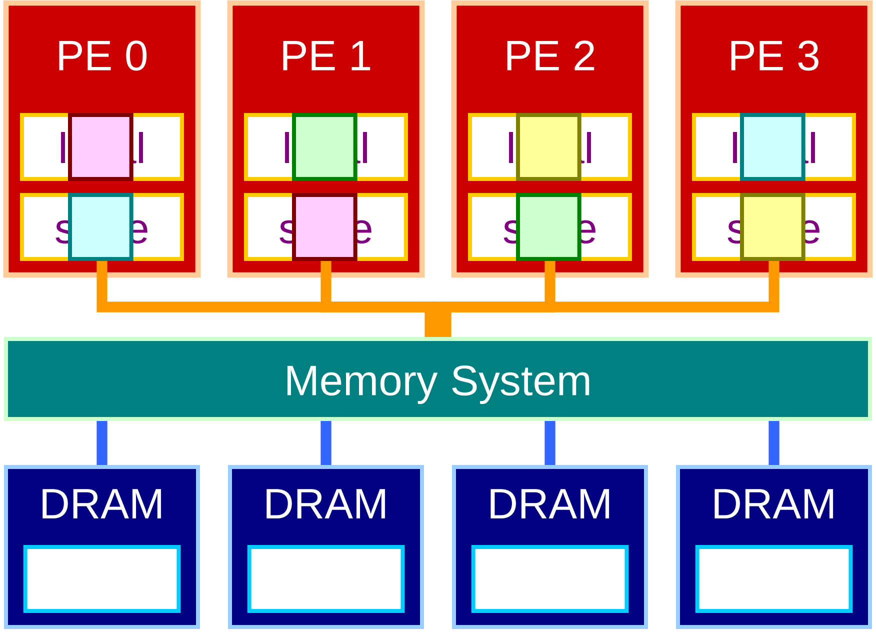
- LRF and SRF provide a hierarchy of bandwidth and locality
- SRF decouples execution from memory

Streaming Memory Systems (SMSs)



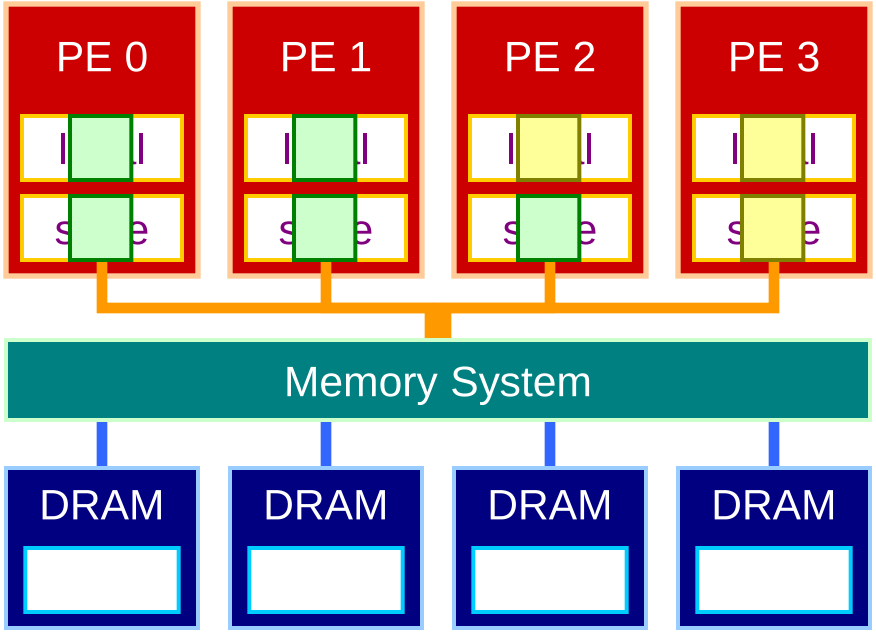
- Off-chip DRAMs need to meet the processor’s bandwidth demands
  - Multiple address-interleaved memory channels
  - High bandwidth DRAM per channel

The performance of a streaming memory system is very sensitive to access patterns



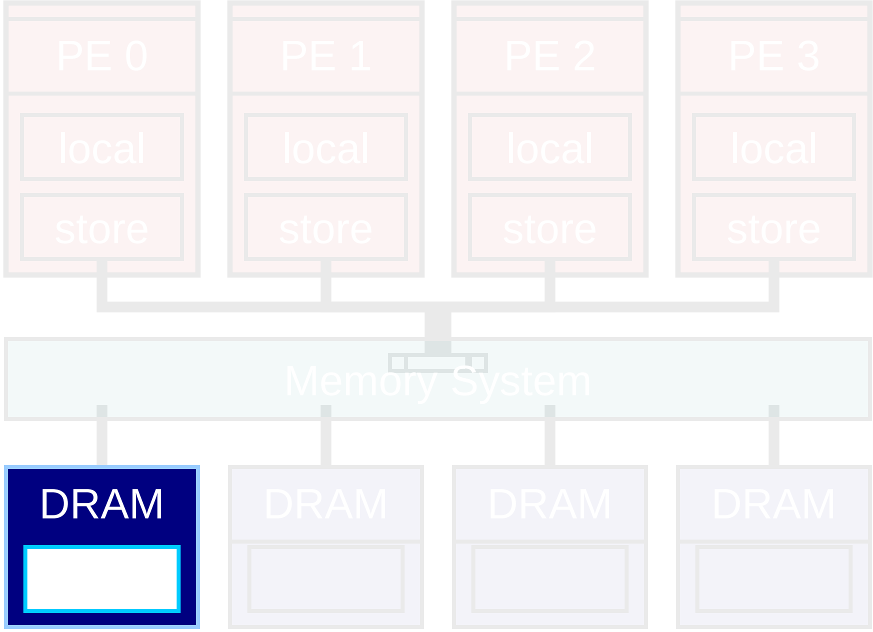
- 1) Because of load imbalance between multiple memory channels

The performance of a streaming memory system is very sensitive to access patterns



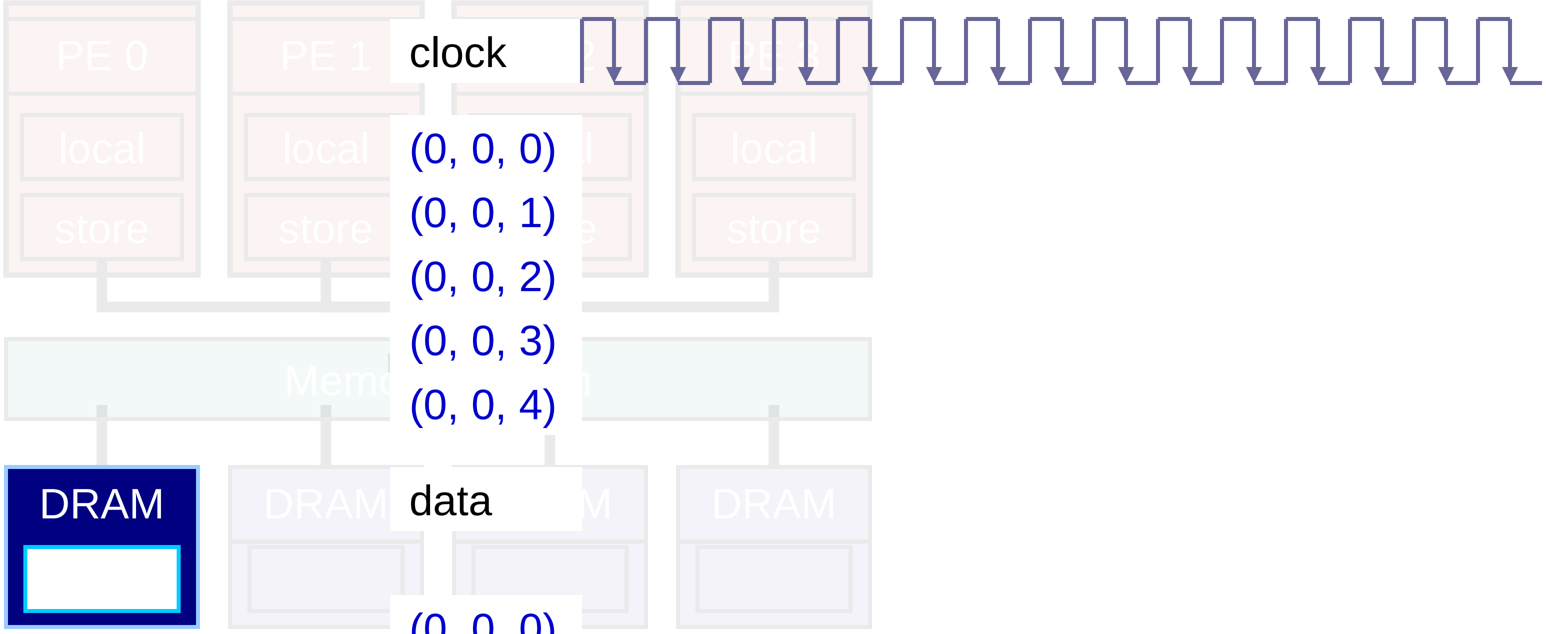
- 1) Because of load imbalance between multiple memory channels

The performance of a streaming memory system is very sensitive to access patterns



- 2) Because the performance of modern DRAMs is very sensitive to access patterns

The performance of a streaming memory system is very sensitive to access patterns



(0, 0, 0)
(0, 0, 1)
(0, 0, 2)
(0, 0, 3)
(0, 0, 4)

(0, 0, 0)
(0, 0, 1)
(0, 0, 2)
(0, 0, 3)
(0, 0, 4)

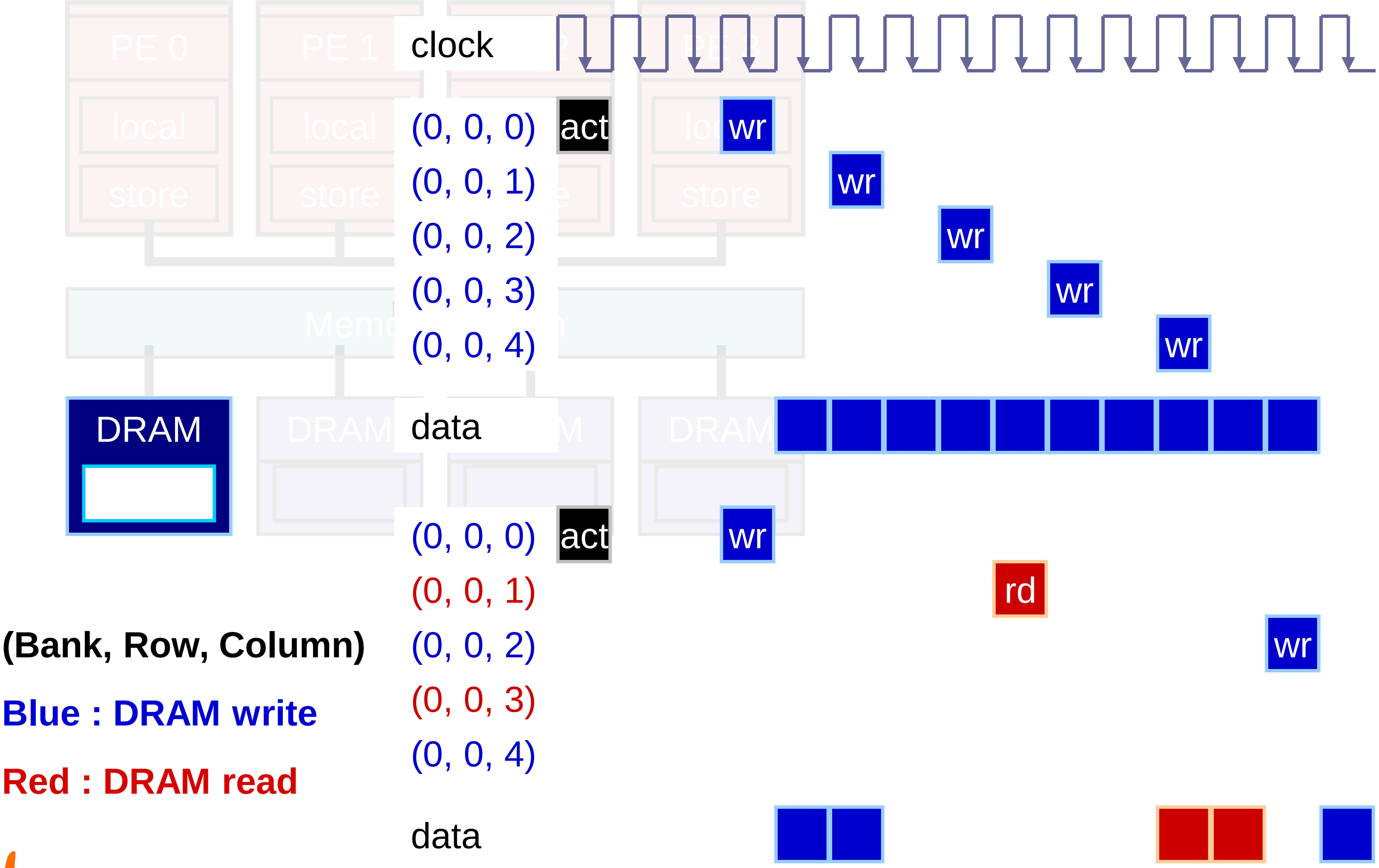
data

(Bank, Row, Column)

Blue : DRAM write

Red : DRAM read

The performance of a streaming memory system is very sensitive to access patterns

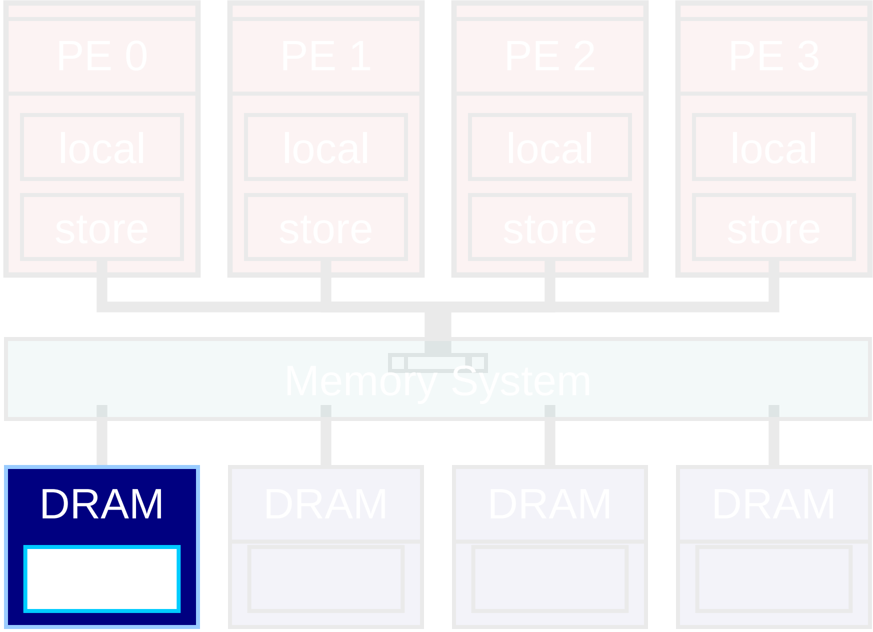


(Bank, Row, Column)

Blue : DRAM write

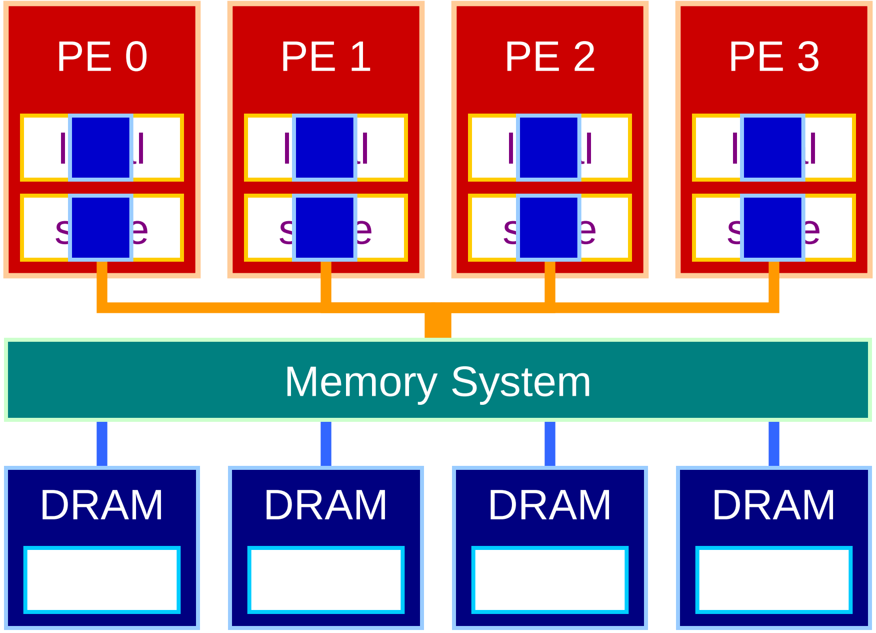
Red : DRAM read

The performance of a streaming memory system is very sensitive to access patterns



- **Parallelism** and **locality** are necessary for efficient DRAM usage

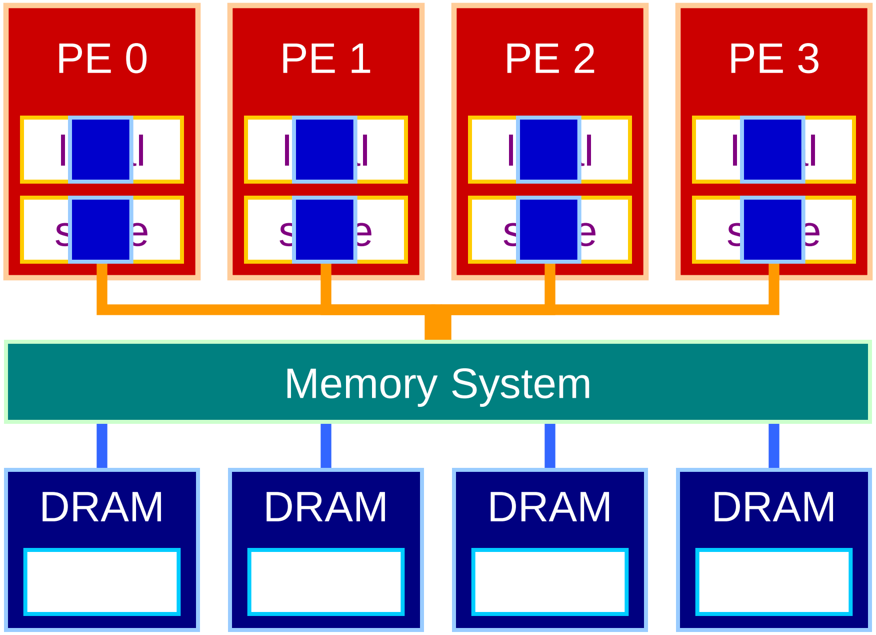
Memory system designs rely on the inherent parallelism/locality of memory accesses



- A stream load or store operation yields a large number of related memory accesses,

Stream store **A**

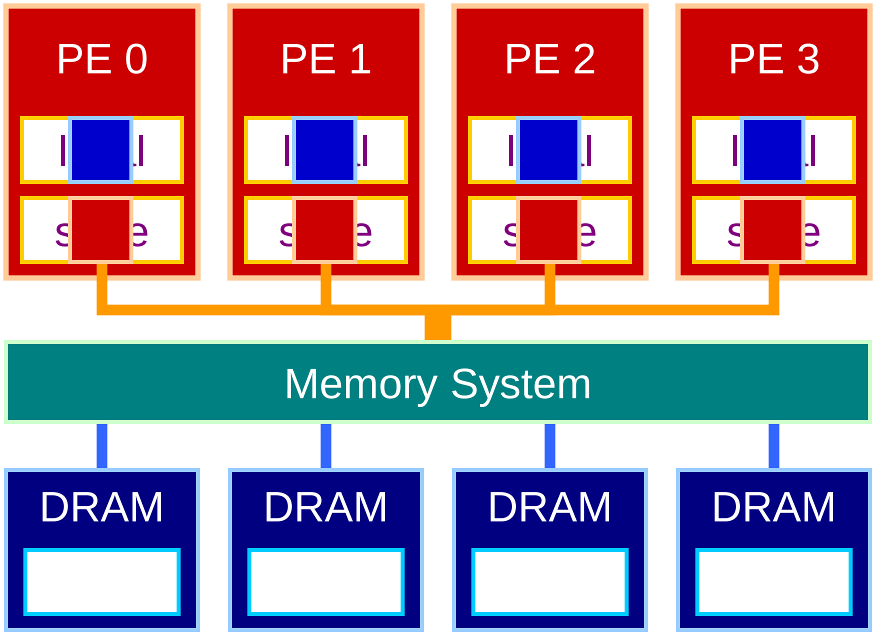
Memory system designs rely on inherent parallelism/locality of memory accesses



- Due to the blocked feature of accesses, a SMS can exploit
  - Parallelism by generating multiple references per cycle per thread

Stream store A

Memory system designs rely on inherent parallelism/locality of memory accesses

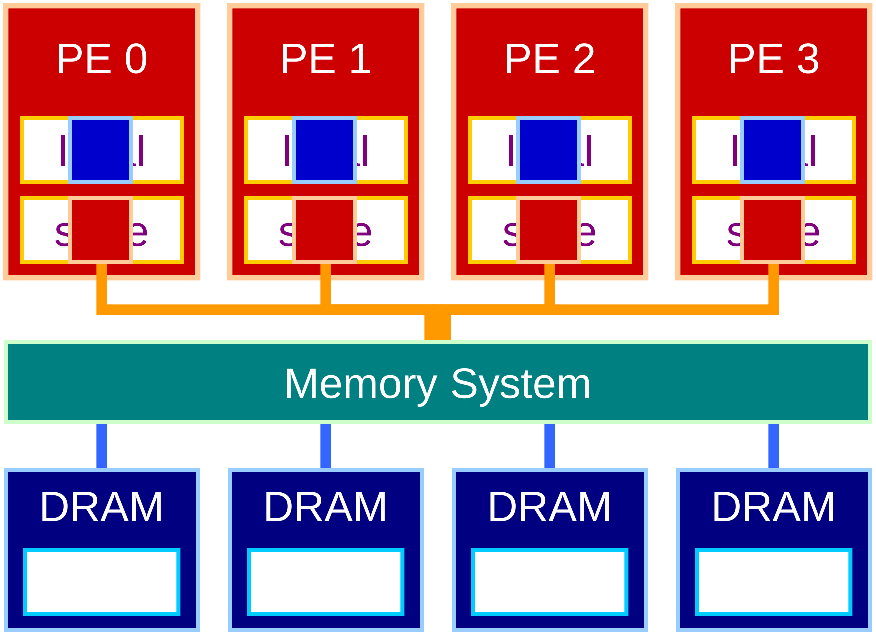


- Due to the blocked feature of accesses, a SMS can exploit
  - Parallelism by generating multiple references per cycle per thread
  - Parallelism by generating references from multiple threads

Stream store **A**

Stream load **B**

Memory system designs rely on inherent parallelism/locality of memory accesses



- Due to the blocked feature of accesses, a SMS can exploit
  - Parallelism by generating multiple references per cycle per thread
  - Parallelism by generating references from multiple threads
  - Parallelism by both of above
  - Locality by generating entire references of a thread

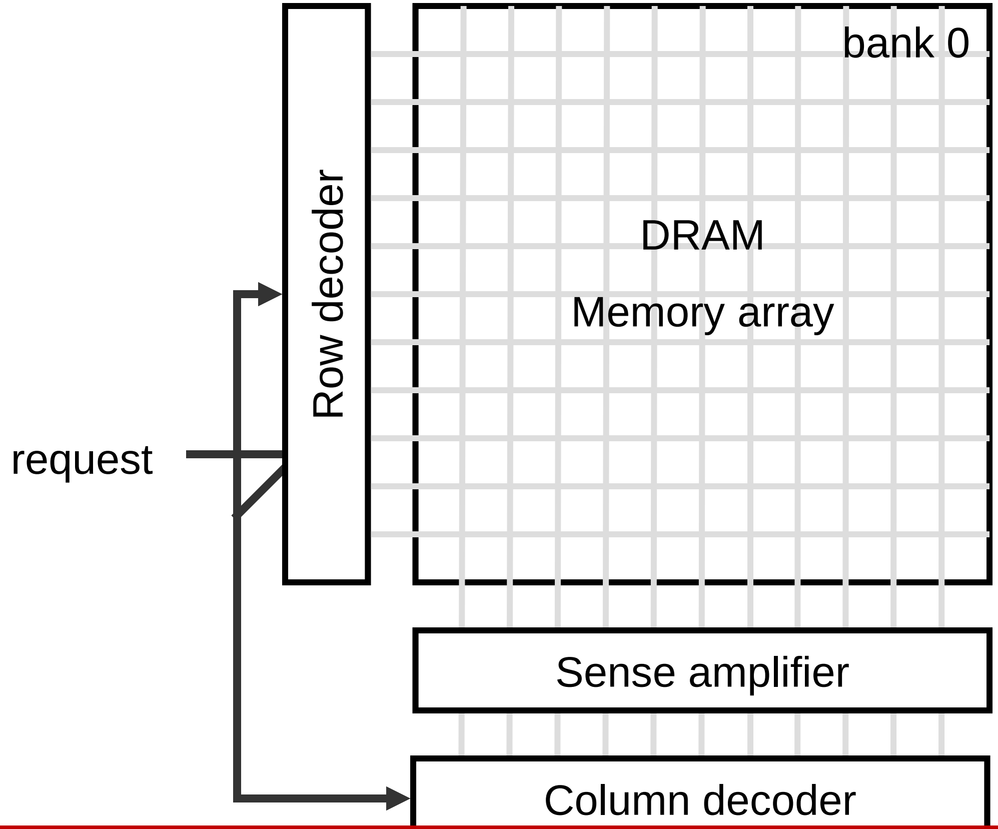
It is important to understand how the interactions between these different factors affect performance

Outline

- DRAM technology
- **DRAM organization, mechanism, and trends**
- Memory system organization and design option
- A few results

- Most slides courtesy Jung Ho Ahn, HP Labs

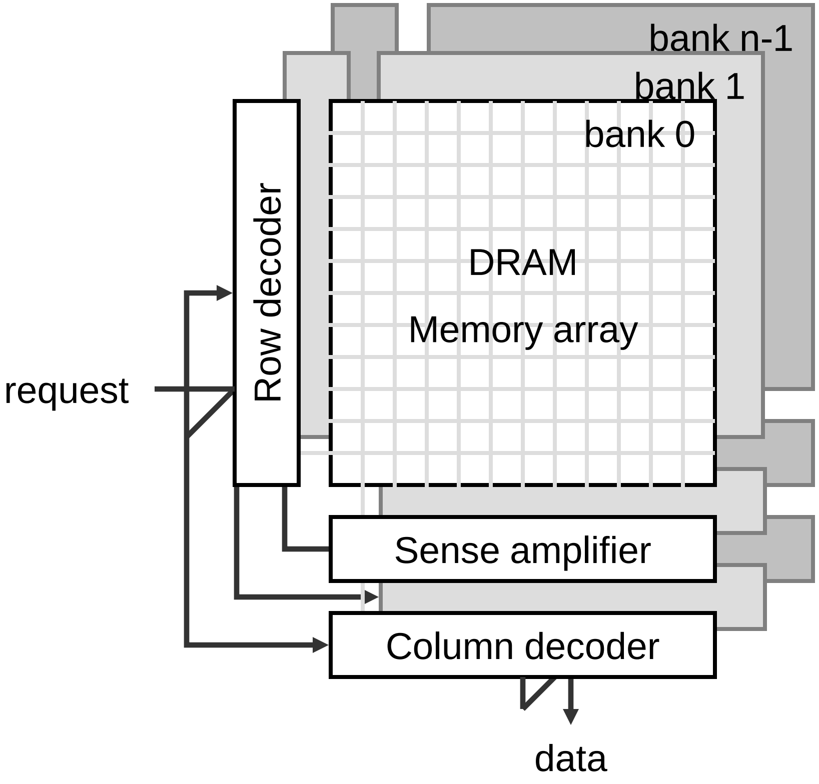
A DRAM chip is containsmultiple memory banks where each bank is a 2-D array



- (bank, row, column)
- Many shared resources on a DRAM chip
  - Row & column accesses shared by request path.
  - Data read & written through shared data path.
  - All banks share request and data path.

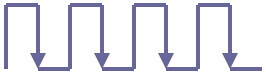
This sharing and the dynamic nature of DRAM result in strict access rules and timing constraints

A DRAM follows rules and occupies resources to access a location



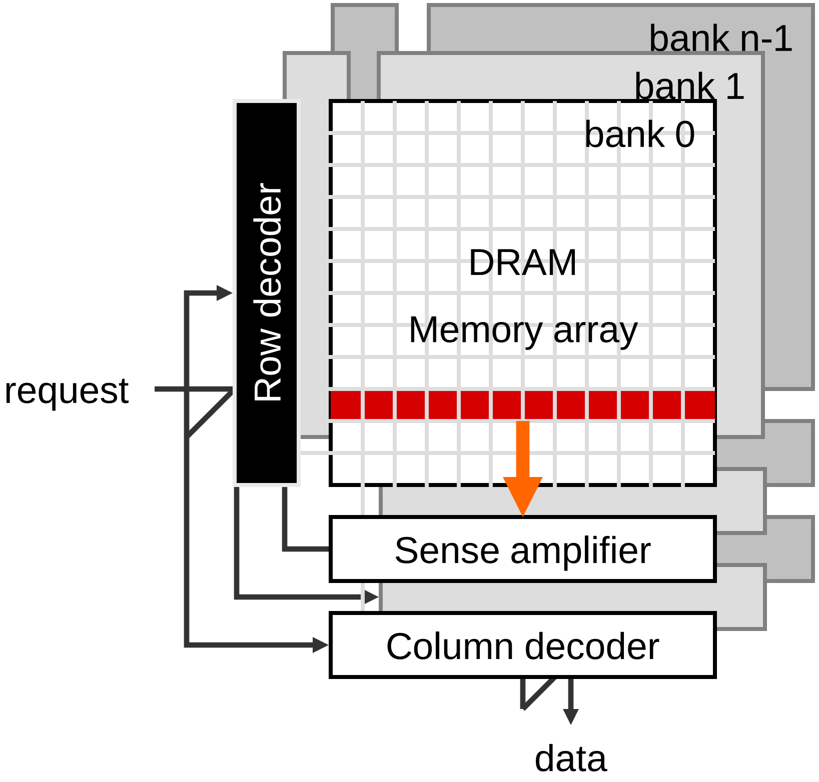
Operation Resource Utilization

Cycle

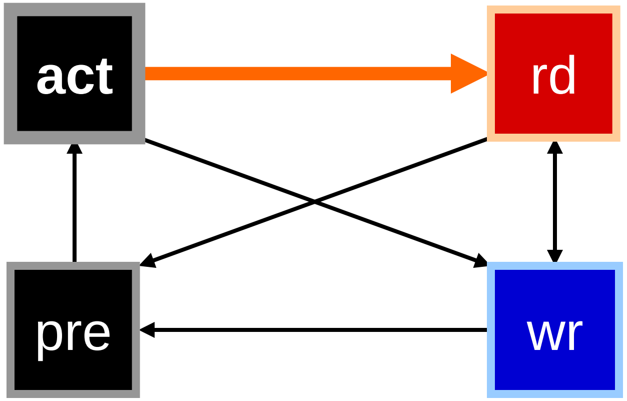


Precharge	Bank	■	■	■	
	Request	■			
	Data				
Activate	Bank	■	■	■	
	Request	■			
	Data				
Read	Bank				
	Request	🟥			
	Data				🟥
Write	Bank				
	Request	🟦			
	Data		🟦		

A DRAM follows rules and occupies resources to access a location

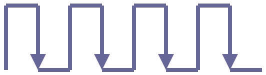


Simplified Bank State Diagram



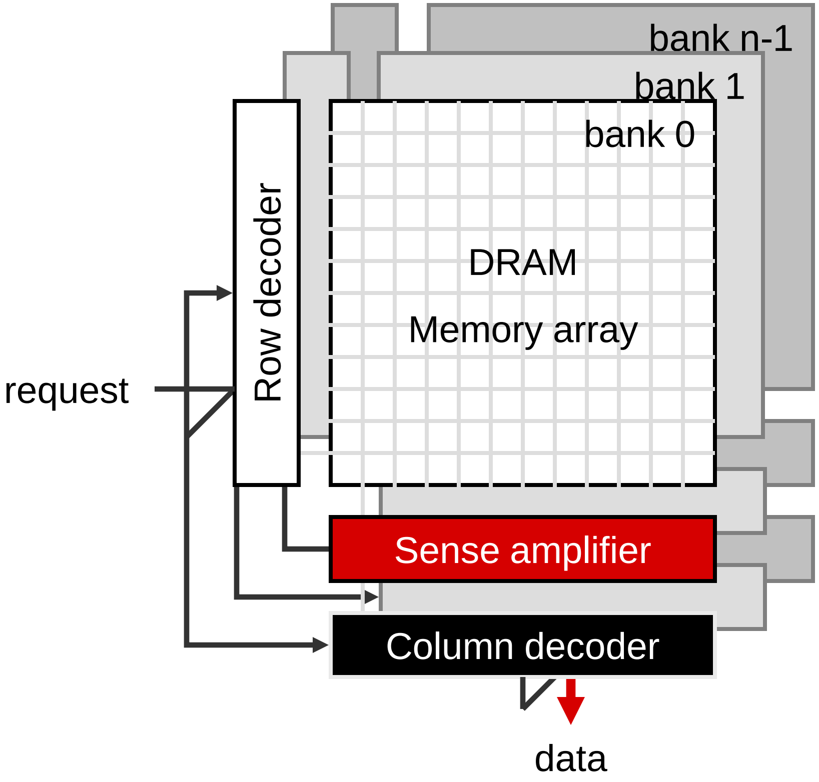
Operation Resource Utilization

Cycle

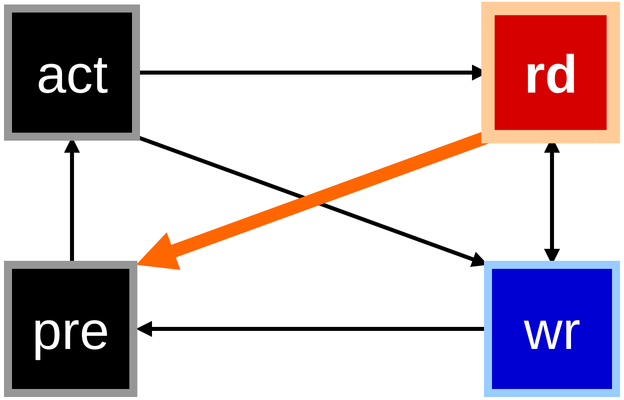


Activate	Bank	■	■	■	
	Request	■			
	Data				

A DRAM follows rules and occupies resources to access a location



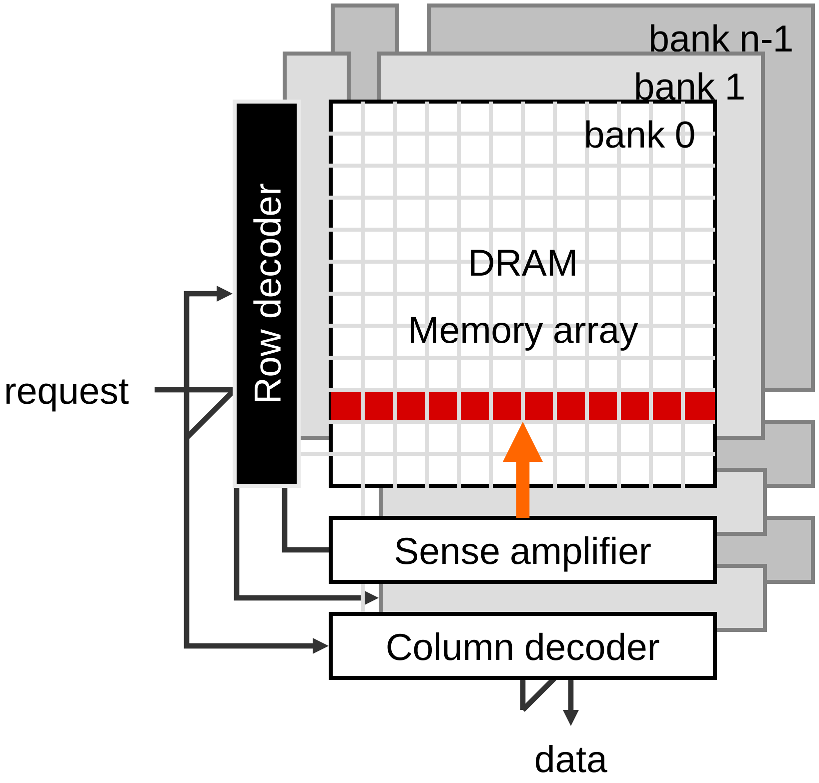
Simplified Bank State Diagram



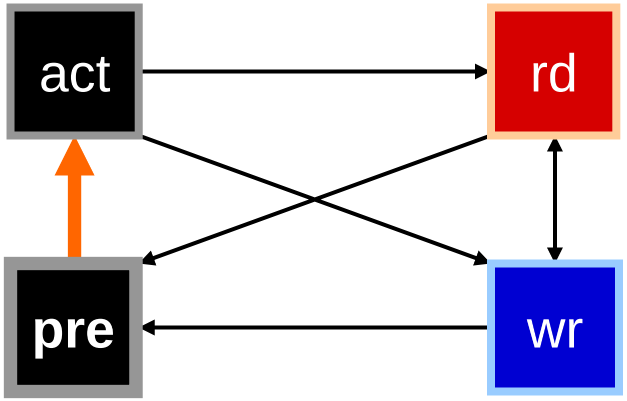
Operation Resource Utilization

	Cycle				
Read	Bank				
	Request	🟥			
	Data				🟥

A DRAM follows rules and occupies resources to access a location



Simplified Bank State Diagram



Operation Resource Utilization

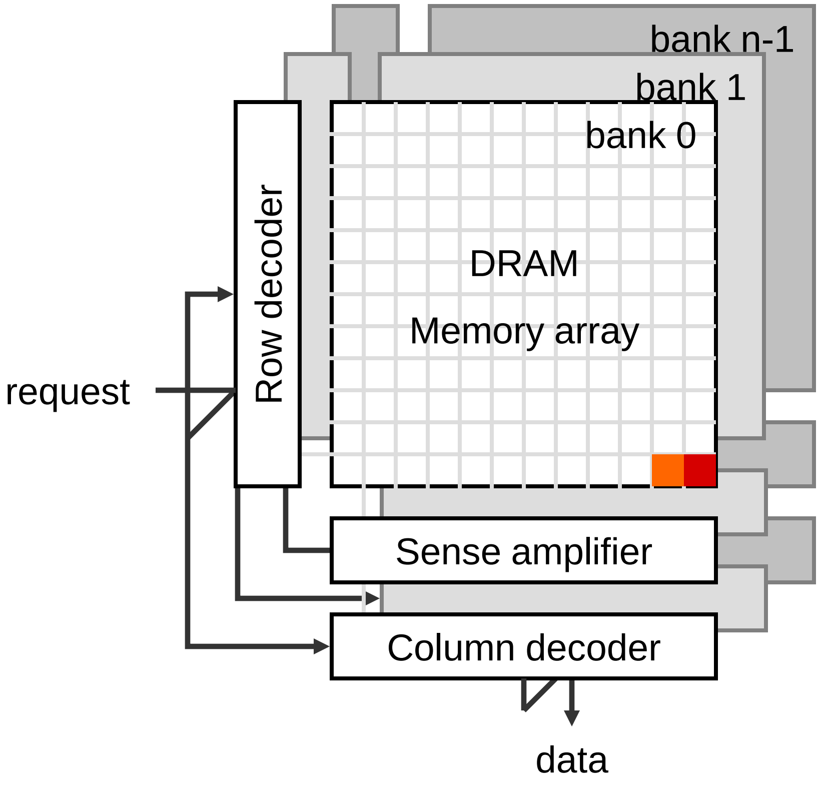
	Cycle				
Precharge	Bank	■	■	■	
	Request	■			
	Data				

DRAM operation sequences for two memory read requests

Read (Bank, Row, Column)

1. (0, 0, 0) → (0, 0, 1) : different column

act (0, 0, *) → rd (0, *, 0) → rd (0, *, 1)



DRAM operation sequences for two memory read requests

Read (Bank, Row, Column)

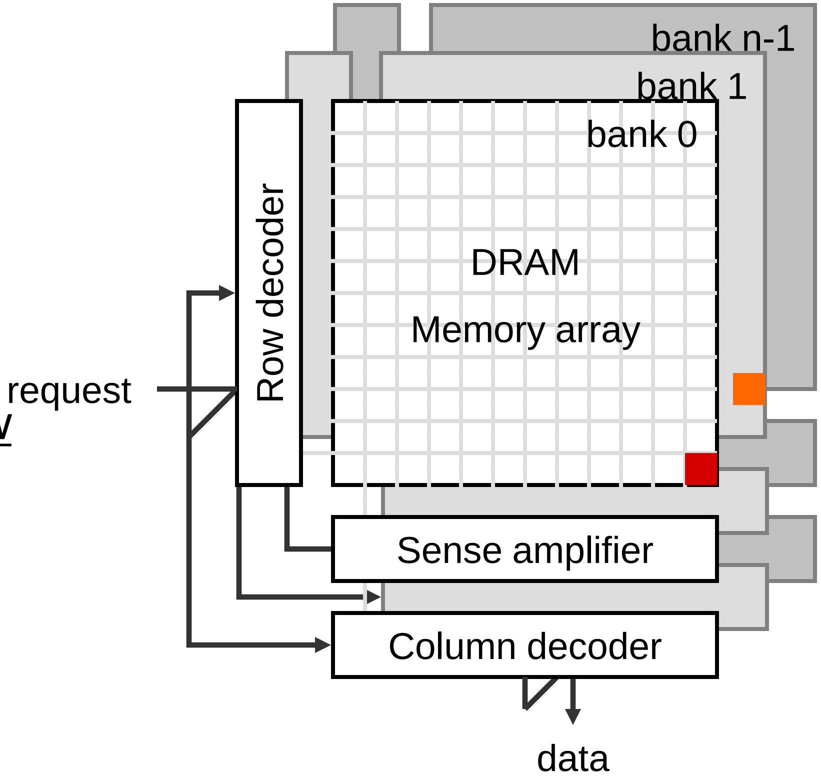
1. (0, 0, 0) → (0, 0, 1) : different column

act (0, 0, *) → rd (0, *, 0) → rd (0, *, 1)

2. (0, 0, 0) → (1, 1, 0) : different bank & row

act (0, 0, *) → rd (0, *, 0)

act (1, 1, *) → rd (1, *, 0)

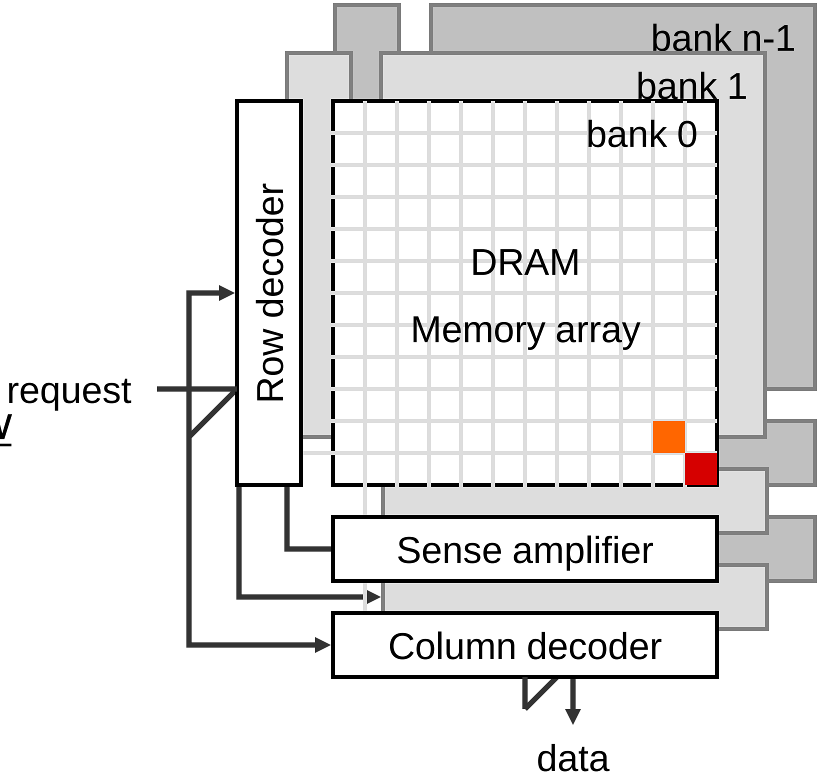


DRAM operation sequences for two memory read requests

Read (Bank, Row, Column)

1. (0, 0, 0) → (0, 0, 1) : different column

act (0, 0, *) → rd (0, *, 0) → rd (0, *, 1)



2. (0, 0, 0) → (1, 1, 0) : different bank & row

act (0, 0, *) → rd (0, *, 0)

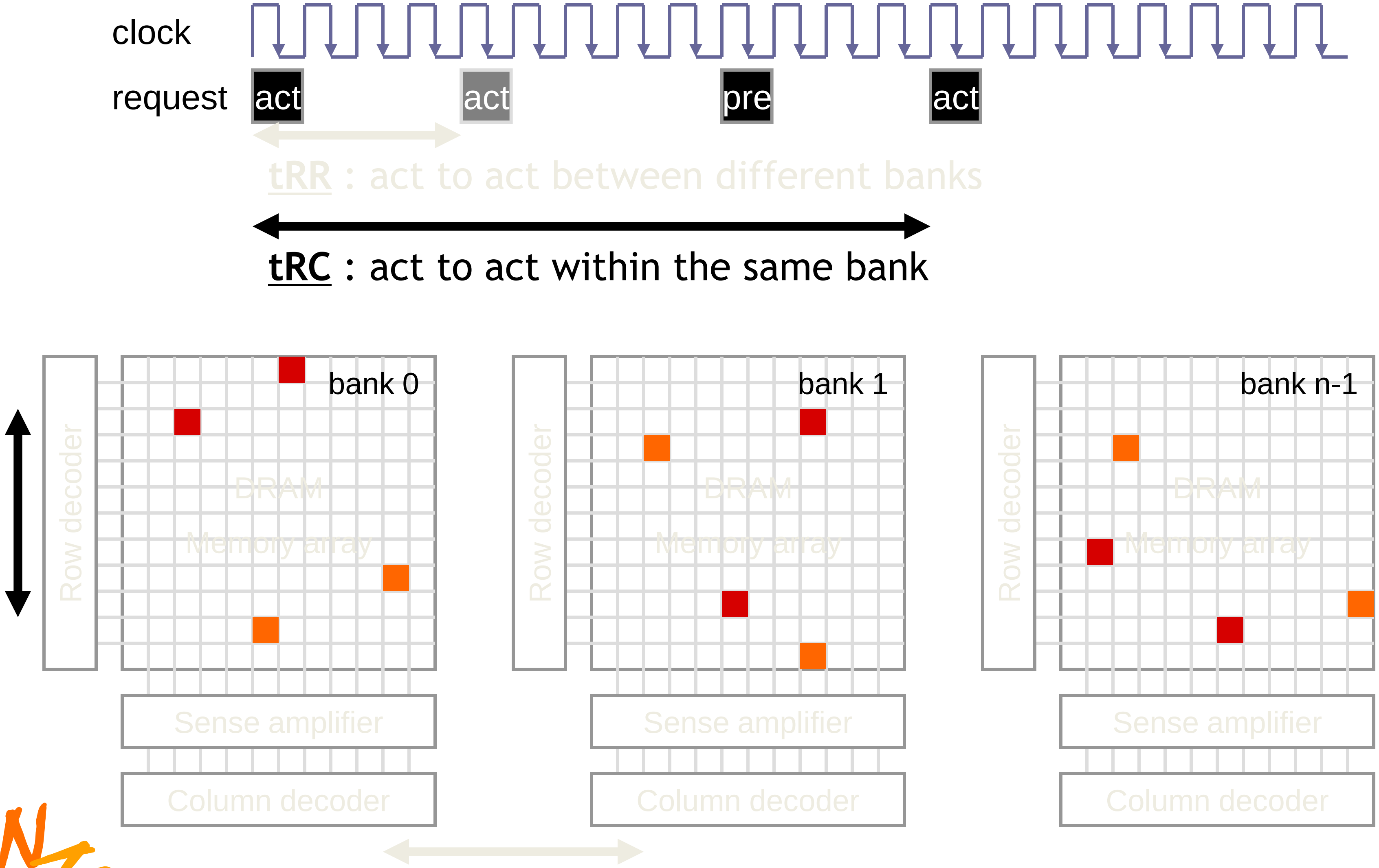
act (1, 1, *) → rd (1, *, 0)

3. (0, 0, 0) → (0, 1, 1) : different row & column

act (0, 0, *) → rd (0, *, 0) → pre (0, 0, *) → act (0, 1, *) → rd (0, *, 1)

Internal bank conflict : requiring more commands and cycles

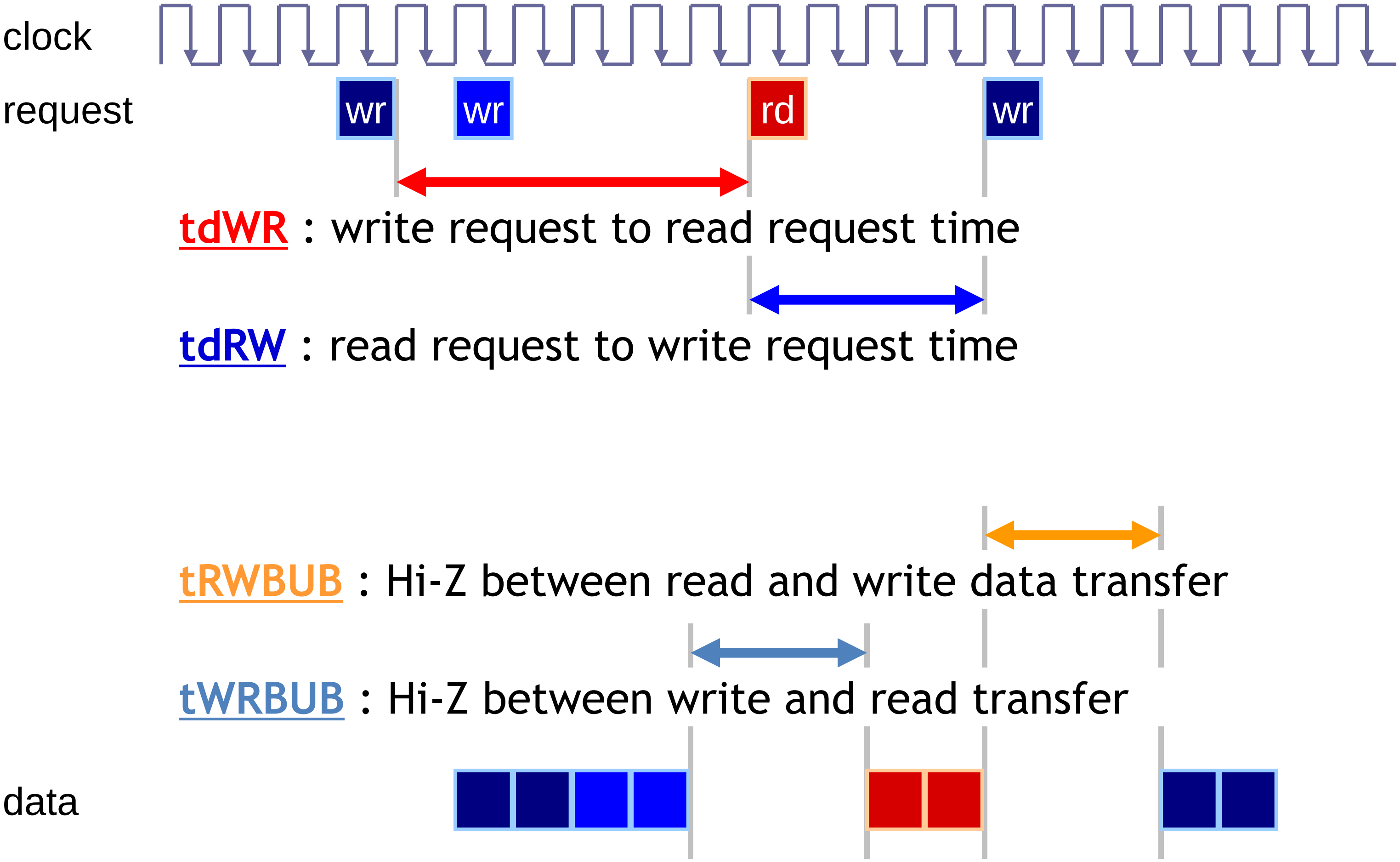
Activate to active time determines random access performance



tRR : act to act between different banks

tRC : act to act within the same bank

Switches between read/write commands and data transfer require timing delay



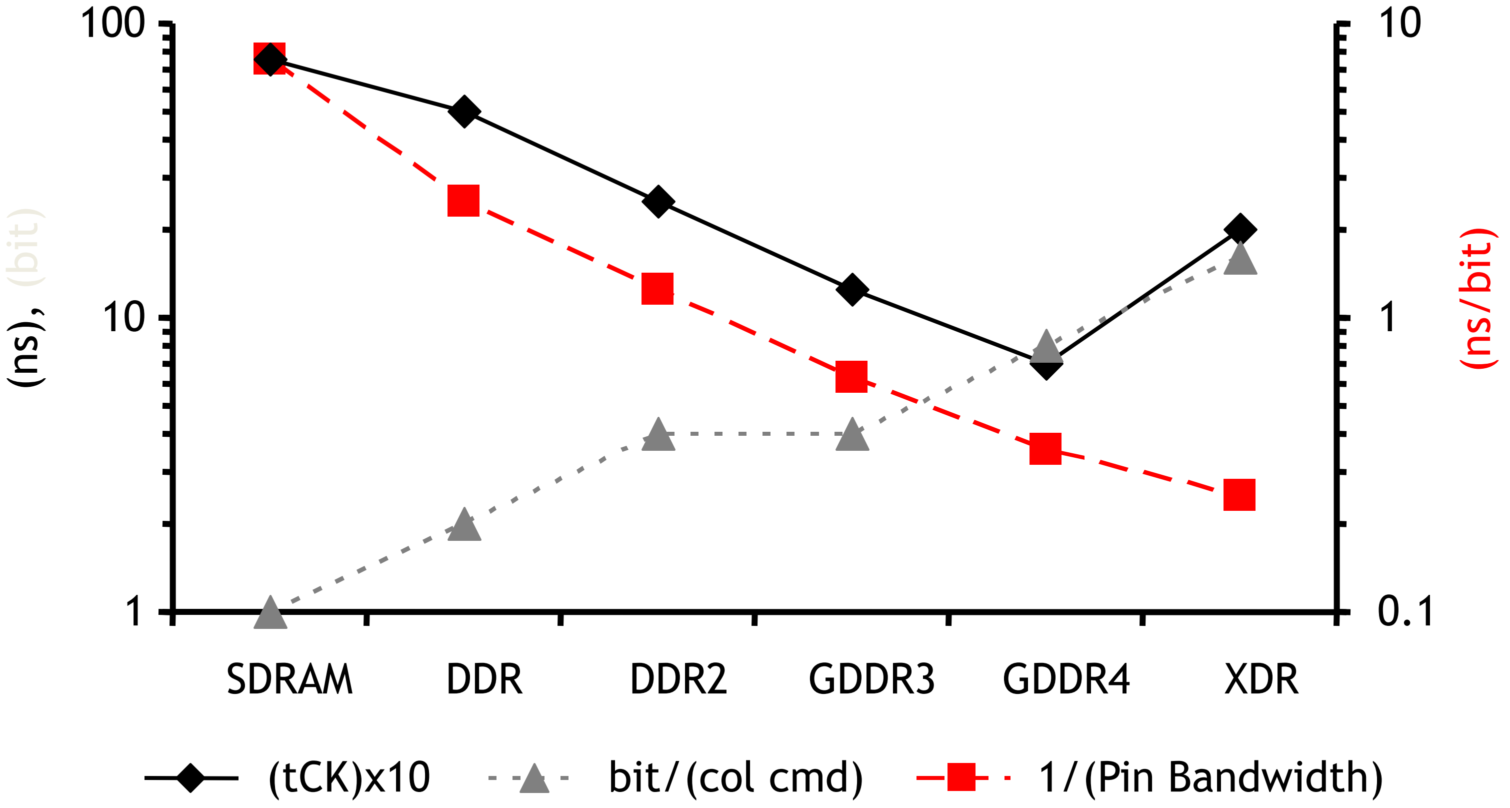
tdWR : write request to read request time

tdRW : read request to write request time

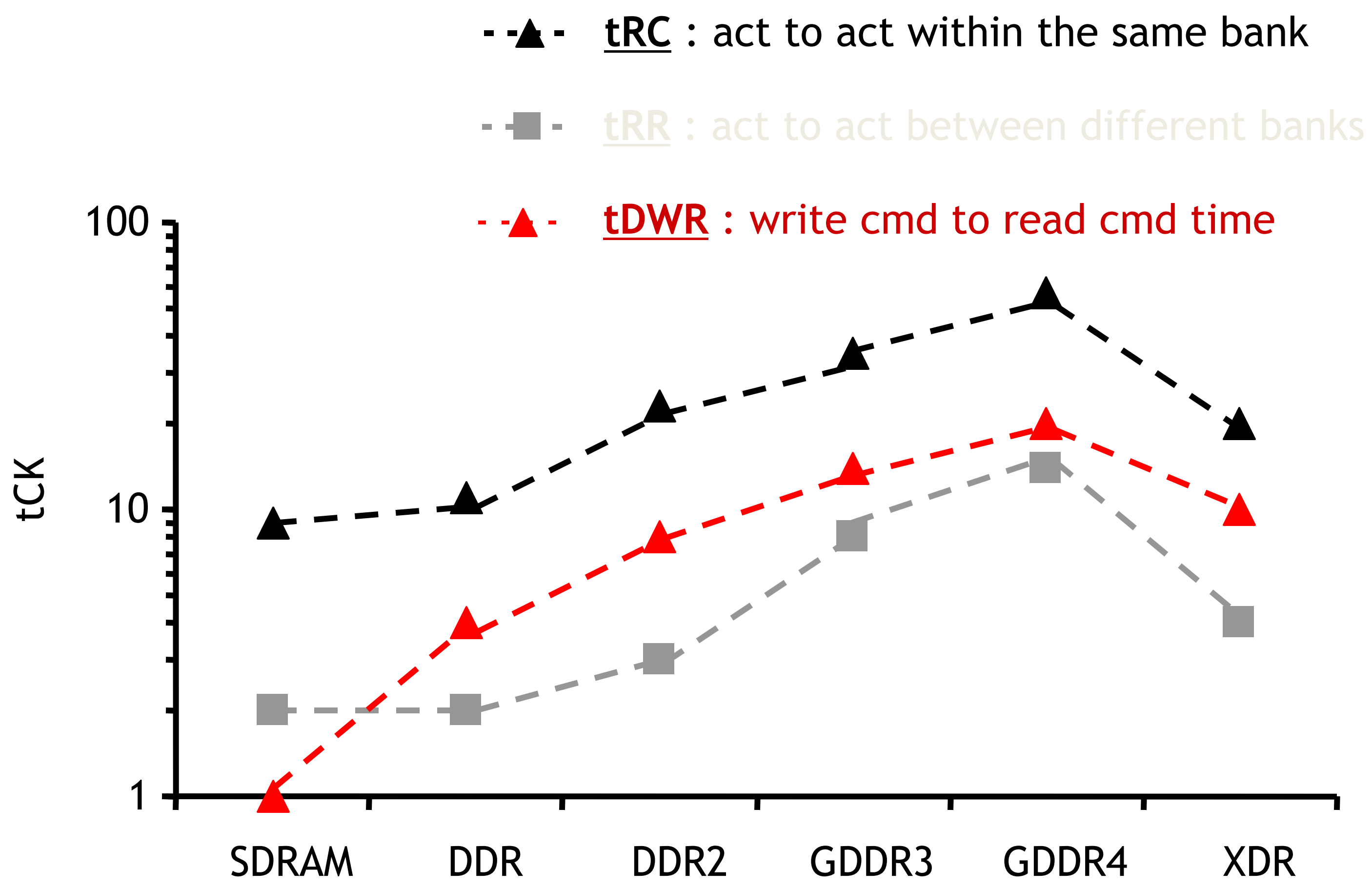
tRWBUB : Hi-Z between read and write data transfer

tWRBUB : Hi-Z between write and read transfer

DRAM parameter trends over various DRAM generations



DRAM parameter trends over various DRAM generations



Timing trends show that DRAM performance is very sensitive to the presented access patterns

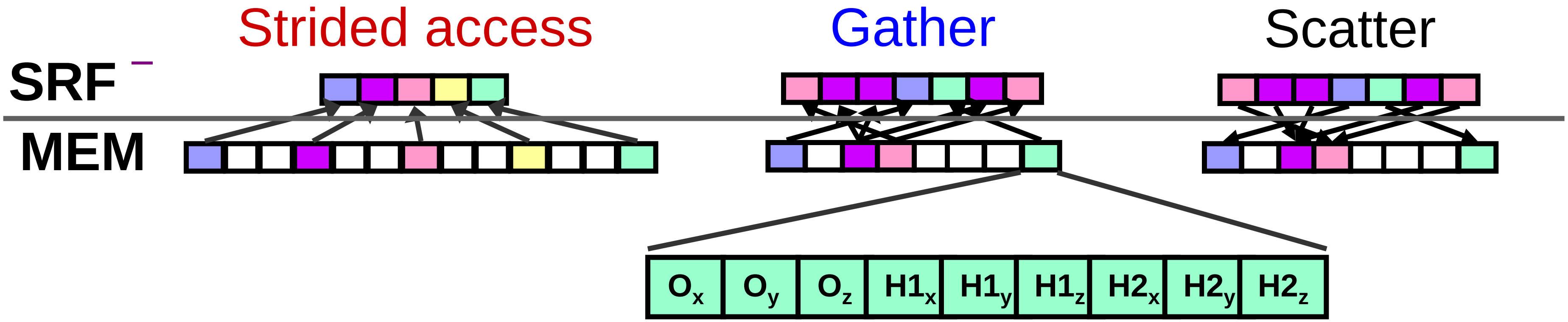
Outline

- DRAM technology
- DRAM organization, mechanism, and trends
- **Memory system organization and design option**
- A few results

- Most slides courtesy Jung Ho Ahn, SNU

Streaming Memory Systems

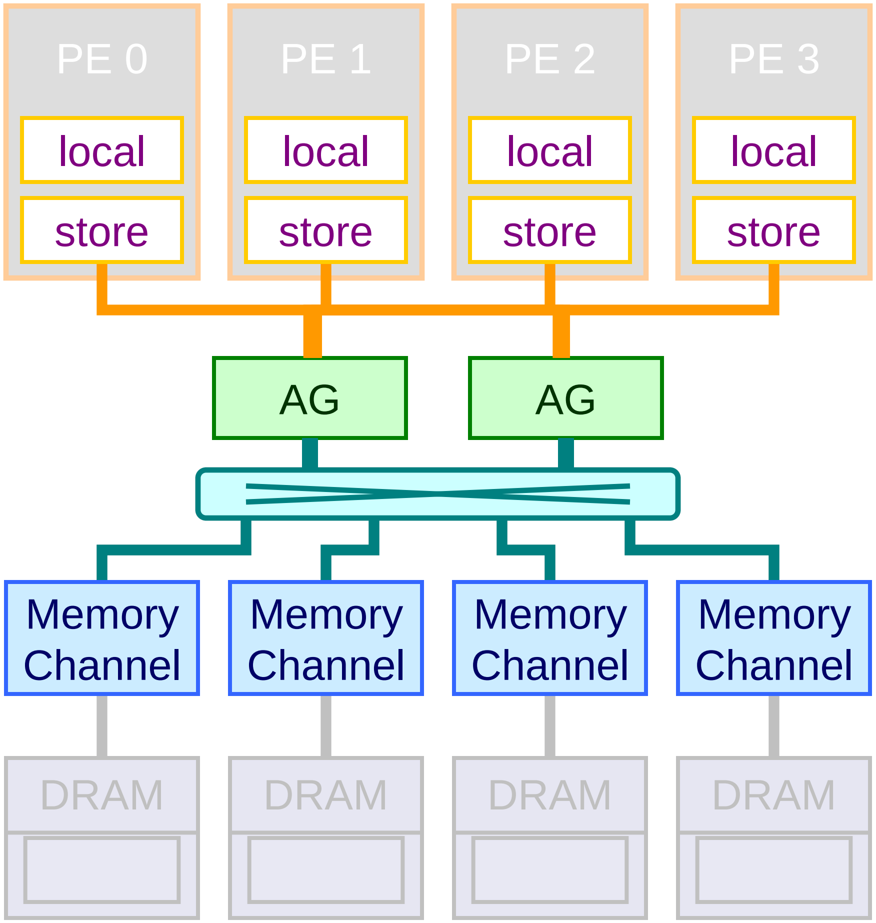
- Bulk stream loads and stores
  - Hierarchical control
- Expressive and effective addressing modes
  - Can't afford to waste memory bandwidth
  - Use hardware when performance is non-deterministic



- Automatic SIMD alignment
  - Makes SIMD trivial (SIMD ≠ short-vector)

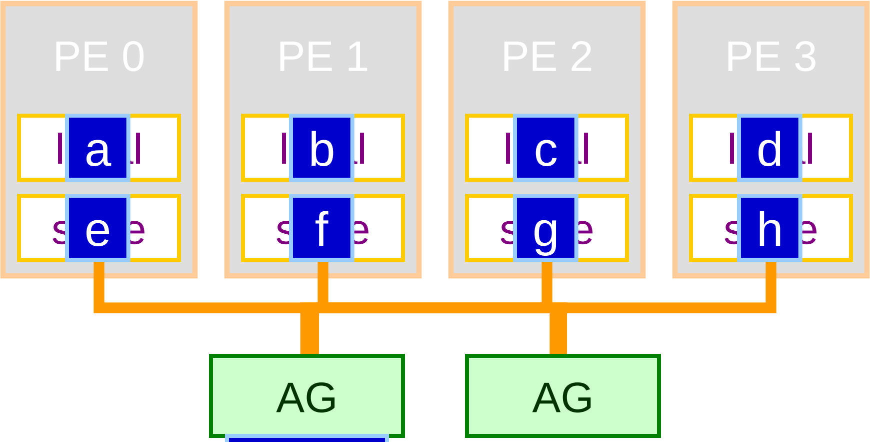
Stream memory system helps the programmer and maximizes I/O throughput

A streaming memory system consists of AGs, cross-point switch and MCs



- AG : address generator
- MC : memory channel

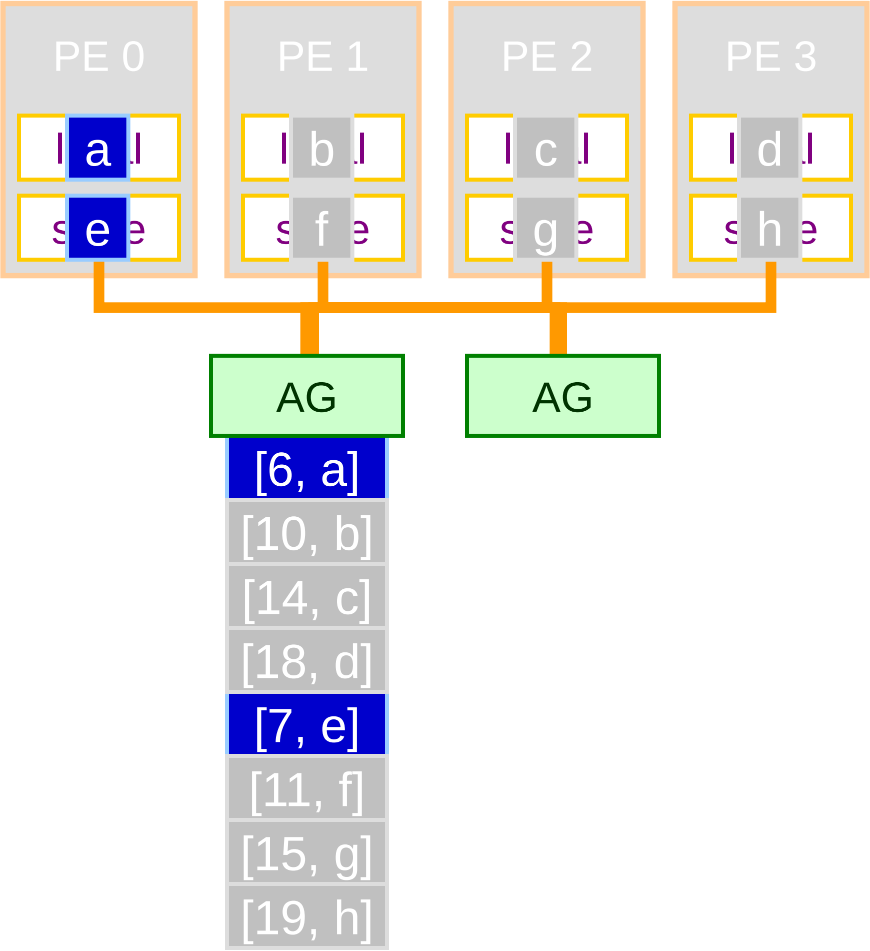
An AG translates memory access thread into a sequence of individual memory requests



[6, a]
[10, b]
[14, c]
[18, d]
[7, e]
[11, f]
[15, g]
[19, h]

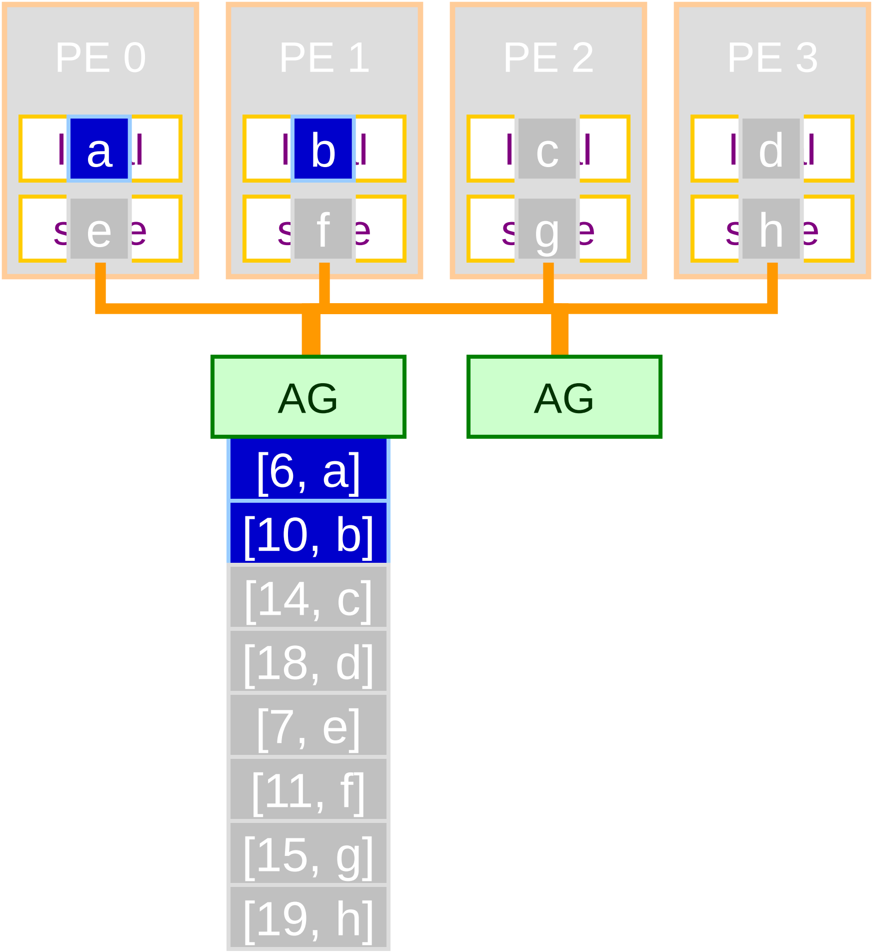
- AG : address generator
- [address, data]

An AG translates memory access thread into a sequence of individual memory requests



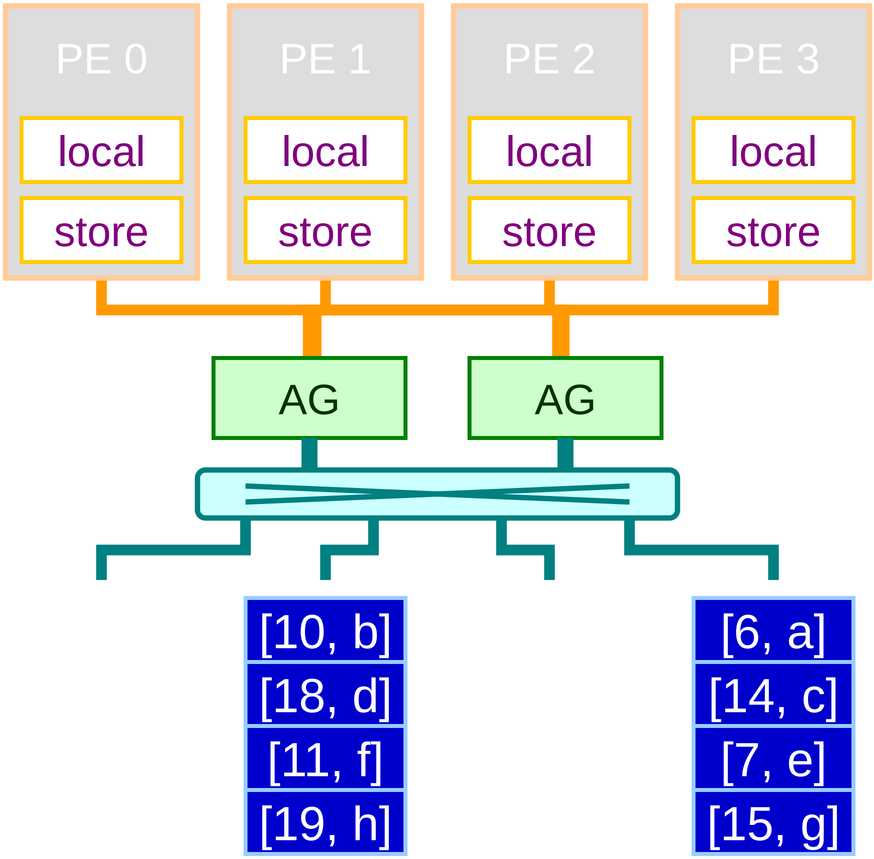
- AG : address generator
- [address, data]
- **Record size : # of consecutive words per data record mapped to a PE**

An AG translates memory access thread into a sequence of individual memory requests



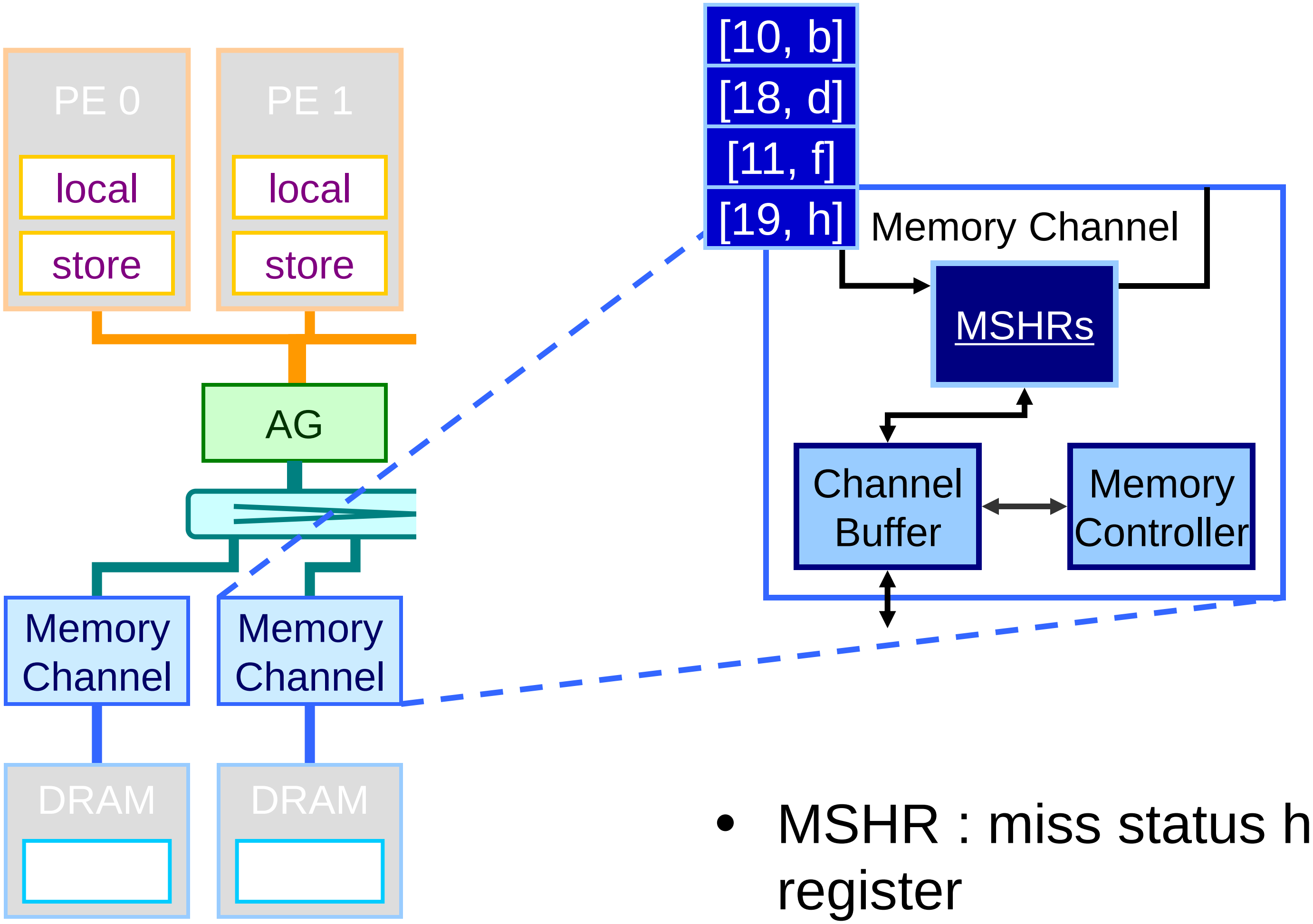
- AG : address generator
- [address, data]
- **Stride : address gap between consecutive records**

Cross-point Switch



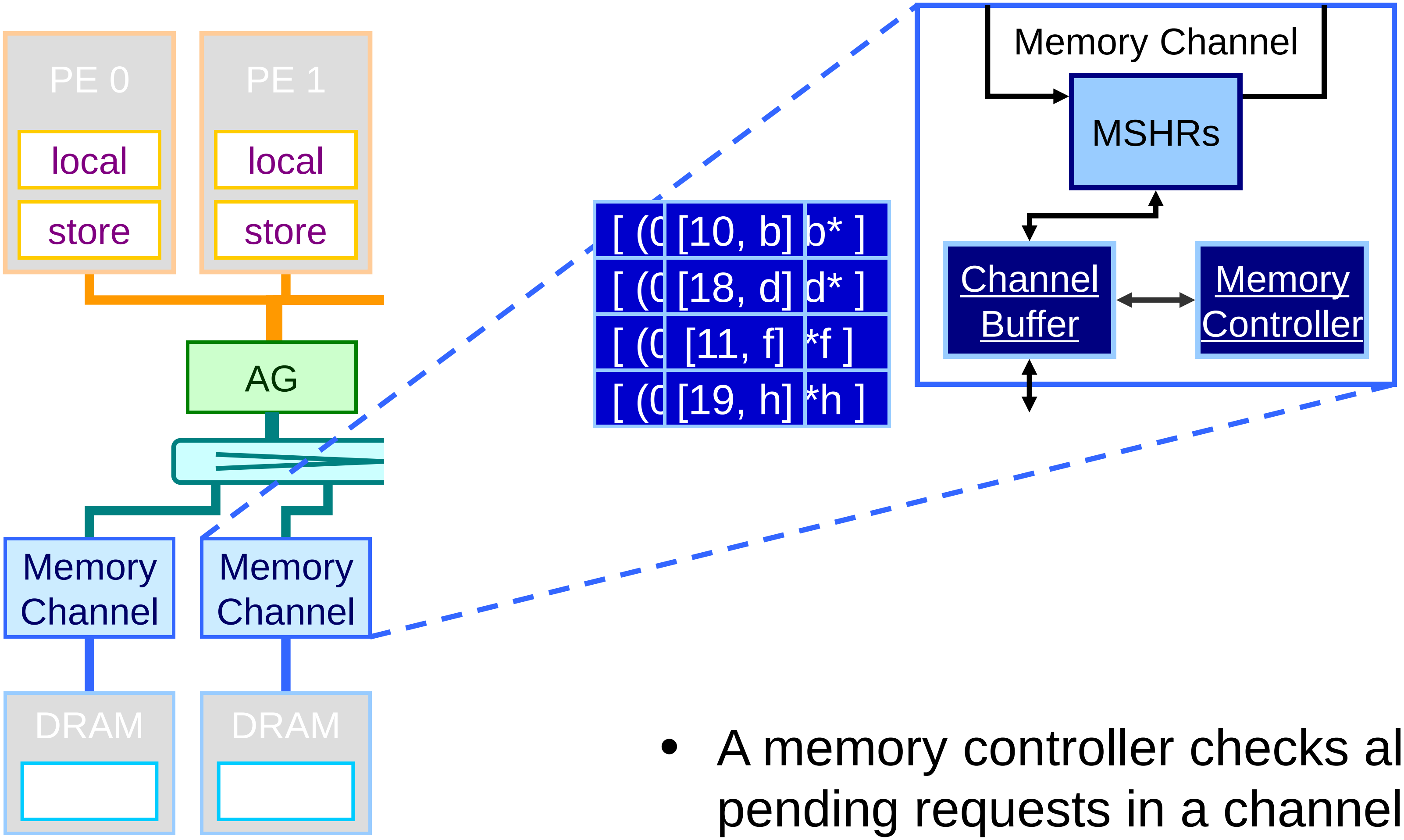
- On-chip and off-chip have **different address spaces**

A memory channel contains MSHRs, channel buffer entries, and a memory controller



- MSHR : miss status handling register

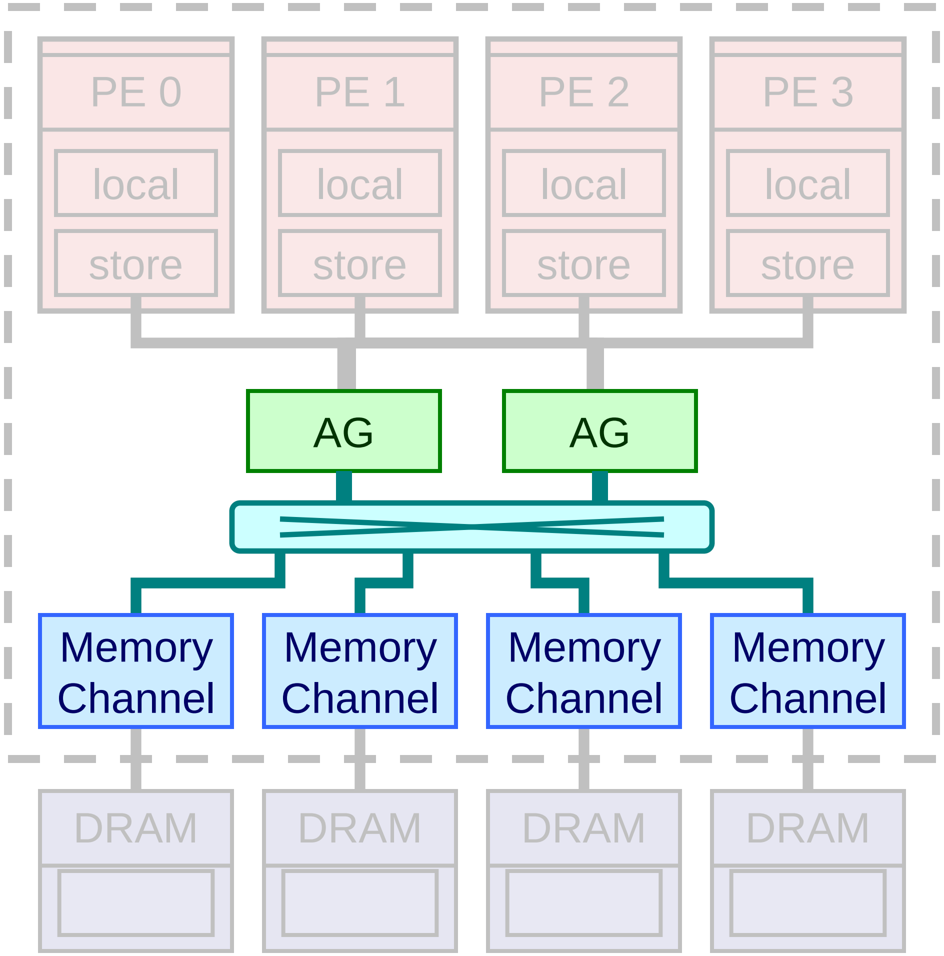
A memory channel contains MSHRs, channel buffer entries, and a memory controller



[((	[10, b]	b*]
[((	[18, d]	d*]
[((	[11, f]	*f]
[((	[19, h]	*h]

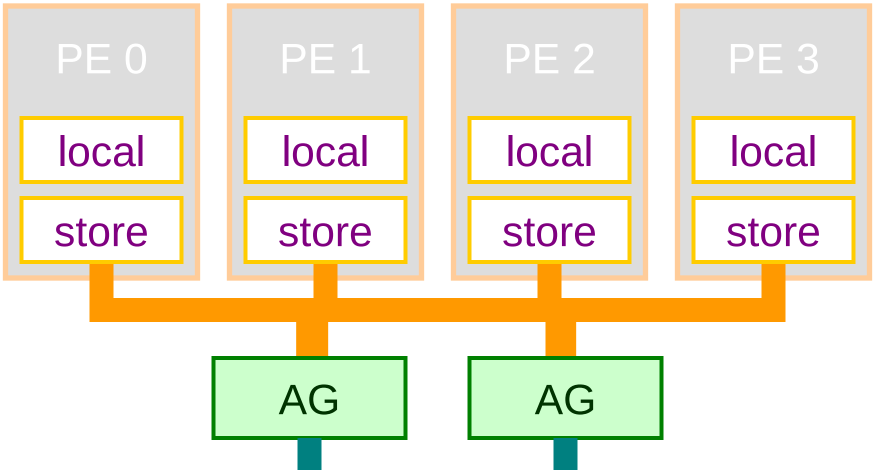
- A memory controller checks all the pending requests in a channel buffer and generates proper DRAM commands

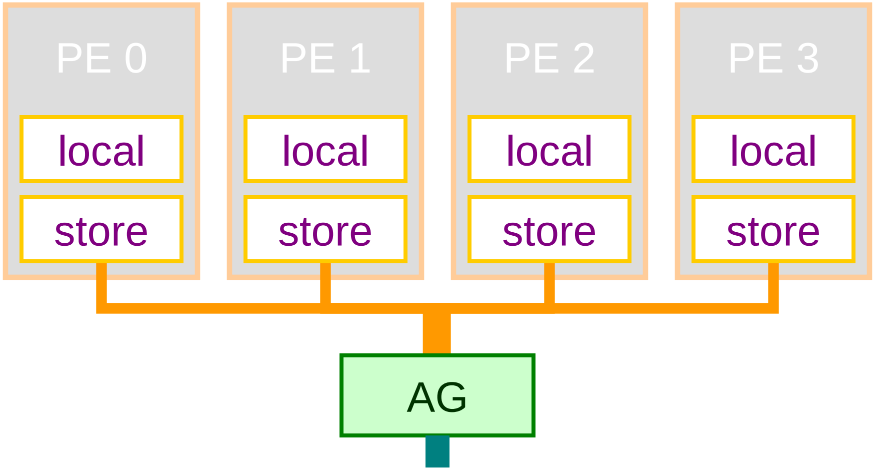
The design space of a Streaming Memory System



- Address generator
  - # of AG
  - AG width
- Memory channel
  - # of channel buffer entries
  - MAS policy
  - Channel-split configuration

AG design space





- A single wide AG vs. multiple narrow AGs
  - Inter-thread vs. intra-thread parallelism
  - **Load balancing** across MC
- The AG width
  - # of accesses each AG can generate per cycle

Memory controller in a MC determines DRAM command per cycle based on the MAS policy

- Per DRAM command cycle, the memory controller
  - Looks at the status of every DRAM bank

DRAM bank status

bank 0 : row 2 active

bank 1 : row 0 active

bank 2 : row 3 precharging

bank 3 : idle

Memory controller in a MC determines DRAM command per cycle based on the MAS policy

- Per DRAM command cycle, the memory controller
  - Looks at the status of every DRAM bank
  - Finds an available command per pending access without violating timing and resource constraints

DRAM bank status

bank 0 : row 2 active

bank 1 : row 0 active

bank 2 : row 3 precharging

bank 3 : idle

Pending requests in channel buffer

(0, 0, 0) write - precharge

(1, 0, 0) read - read

(0, 2, 1) write

(1, 0, 1) read - read

Read occurred in the previous cycle

Memory controller in a MC determines DRAM command per cycle based on the MAS policy

- Per DRAM command cycle, the memory controller
  - Looks at the status of every DRAM bank
  - Finds an available command per pending access without violating timing and resource constraints
  - Selects the command to issue among all available commands based on the priority of the chosen policy

DRAM bank status	Pending requests in channel buffer
bank 0 : row 2 active	(0, 0, 0) write - precharge
bank 1 : row 0 active	**(1, 0, 0) read - read**
bank 2 : row 3 precharging	(0, 2, 1) write
bank 3 : idle	(1, 0, 1) read - read
	Read occurred in the previous cycle

Memory controller in a MC determines DRAM command per cycle based on the MAS policy

- Scheduling policies
  - **Open** : a row is precharged when there are **no pending** accesses **to the row** and there is at least **one pending** access **to a different row** in the same bank

Pending requests in channel buffer

Case 1	(1, 0, 0) write	Case 2	(1, 0, 0) write
	(1, 0, 0) read		(1, 3, 0) read
	(2, 0, 1) write		(2, 5, 1) write
	(2, 0, 1) read		(0, 1, 1) read

bank 0 : row 0 is active

Memory controller in a MC determines DRAM command per cycle based on the MAS policy

- Scheduling policies
  - **Open** : a row is precharged when there are **no pending** accesses **to the row** and there is at least **one pending** access **to a different row** in the same bank
  - **Closed** : a row is precharged **as soon as the last** available reference to that row is performed

Pending requests in channel buffer

Case 1	Case 2
(1, 0, 0) write	(1, 0, 0) write
(1, 0, 0) read	(1, 3, 0) read
(2, 0, 1) write	(2, 5, 1) write
(2, 0, 1) read	(0, 1, 1) read

bank 0 : row 0 is active

MAS determines DRAM command order to serve pending memory accesses

Algorithm	Window size	Reorder row commands	Reorder column commands	Precharging	Access selection
inorder	**1**	N/A	N/A	N/A	N/A

Inorder policy processes pending requests **one by one**, effectively having window size of 1

MAS determines DRAM command order to serve pending memory accesses

Algorithm	Window size	Reorder row commands	Reorder column commands	Precharging	Access selection
inorder	1	N/A	N/A	N/A	N/A
inorderla	**nCB**	**Yes**	**No**	Open	Column First

Inorderla **looks ahead** of other pending requests and generates row commands, **not** column commands

MAS determines DRAM command order to serve pending memory accesses

Algorithm	Window size	Reorder row commands	Reorder column commands	Precharging	Access selection
inorder	1	N/A	N/A	N/A	N/A
inorderla	nCB	Yes	No	Open	Column First
firstready	nCB	Yes	**Yes**	Open	N/A

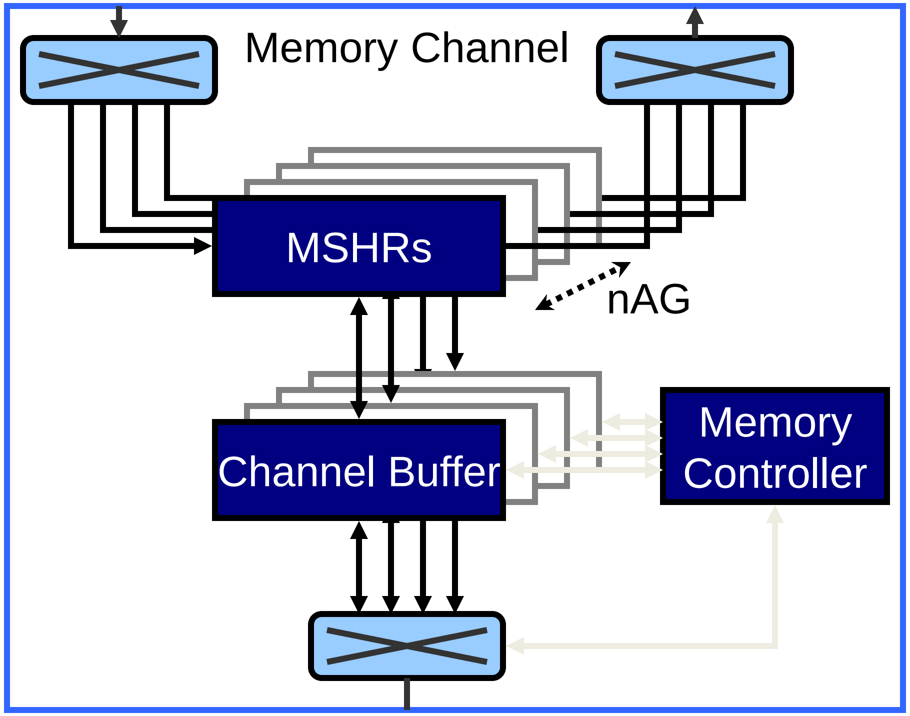
Firstready policy checks and processes pending requests one by one from the oldest until it finishes looking at all the CBEs

MAS determines DRAM command order to serve pending memory accesses

Algorithm	Window size	Reorder row commands	Reorder column commands	Precharging	Access selection
inorder	1	N/A	N/A	N/A	N/A
inorderla	nCB	Yes	No	Open	Column First
firstready	nCB	Yes	Yes	Open	N/A
opcol	nCB	Yes	Yes	Open	Column First
oprow	nCB	Yes	Yes	Open	Row First
clcol	nCB	Yes	Yes	Closed	Column First
clrow	nCB	Yes	Yes	Closed	Row First

Remaining four policies reorder both row and column commands using open/closed or column first/row first

A new micro-architecture design for a memory channel – a channel-split configuration



- MSHR and CBE **per AG**
- **Switch** thread when the requests are **completely drained**
- Avoid
  - **Resource monopolization**
  - **Internal bank conflicts**
  - **Read/write turnaround penalty**

Outline

- DRAM technology
- Impact on memory system
  - Stream architecture review
- DRAM organization, mechanism, and trends
- Memory system organization and design option
- **A few results**

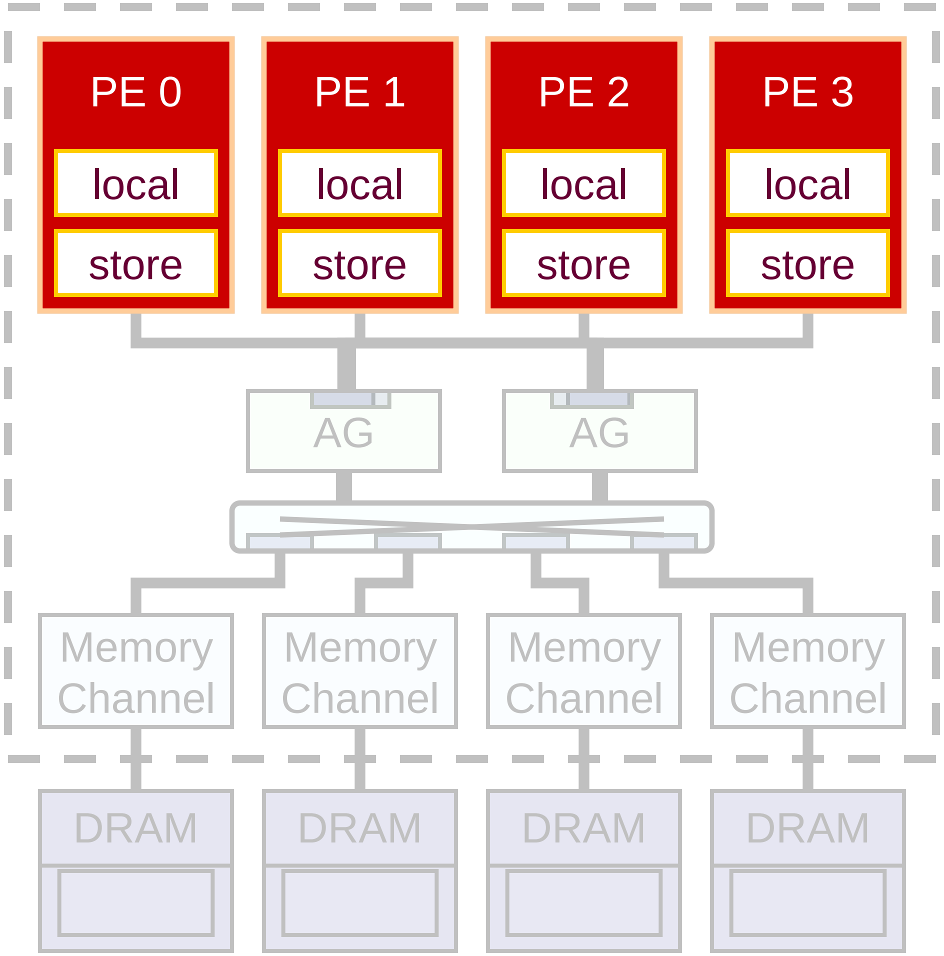
- Most slides courtesy Jung Ho Ahn, HP Labs

Six applications from multimedia and scientific domains are used for study

- **DEPTH** : stereo depth encoder
- **MPEG** : MPEG-2 video encoder
- **RTSL** : graphics rendering pipeline

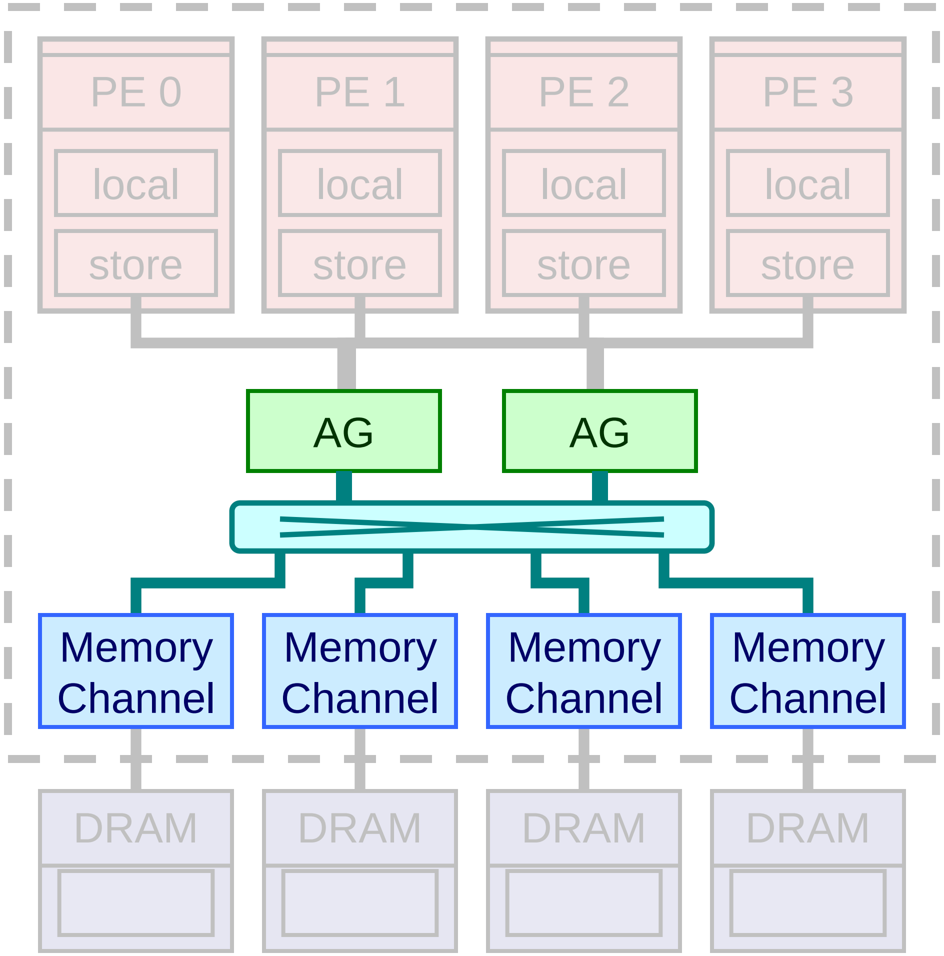
- **QRD** : complex matrix→upper triangular&othorgonal
- **FEM** : finite element method
- **MOLE** : n-body molecular dynamics

A cycle-accurate Imagine simulator is used for performance evaluation



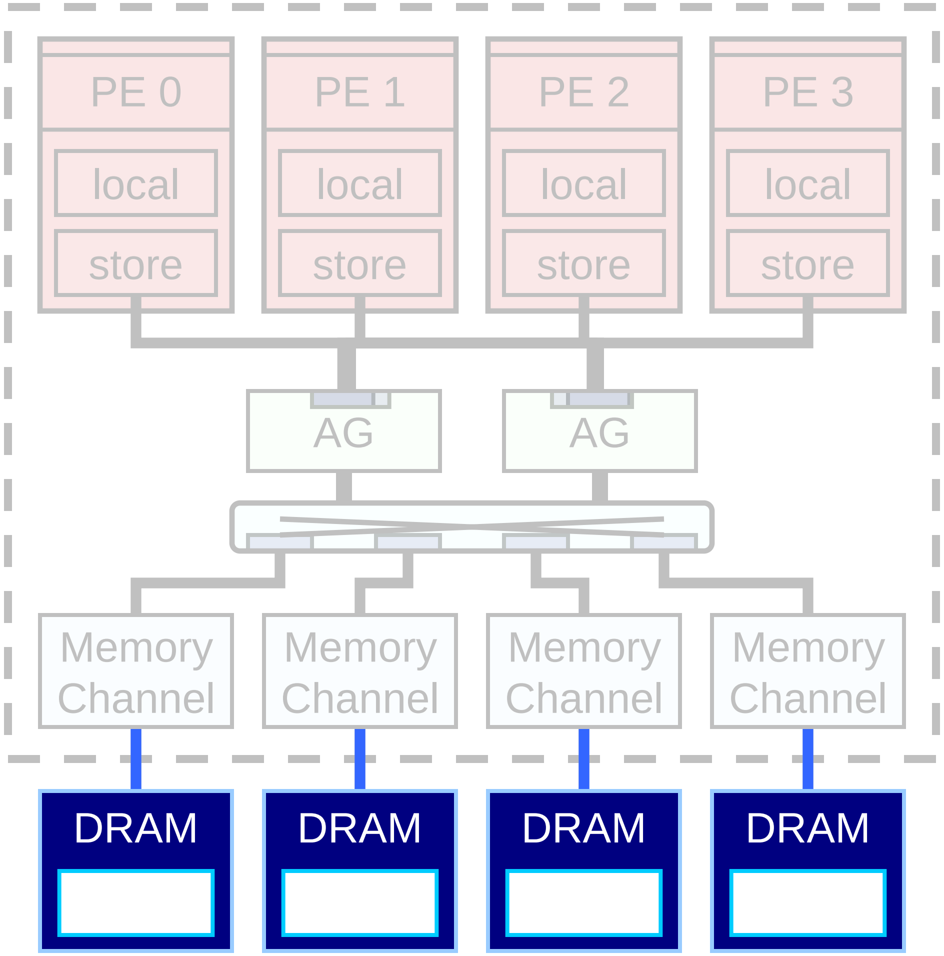
- 1 GHz
- 8 processing elements

A cycle-accurate Imagine simulator is used for performance evaluation



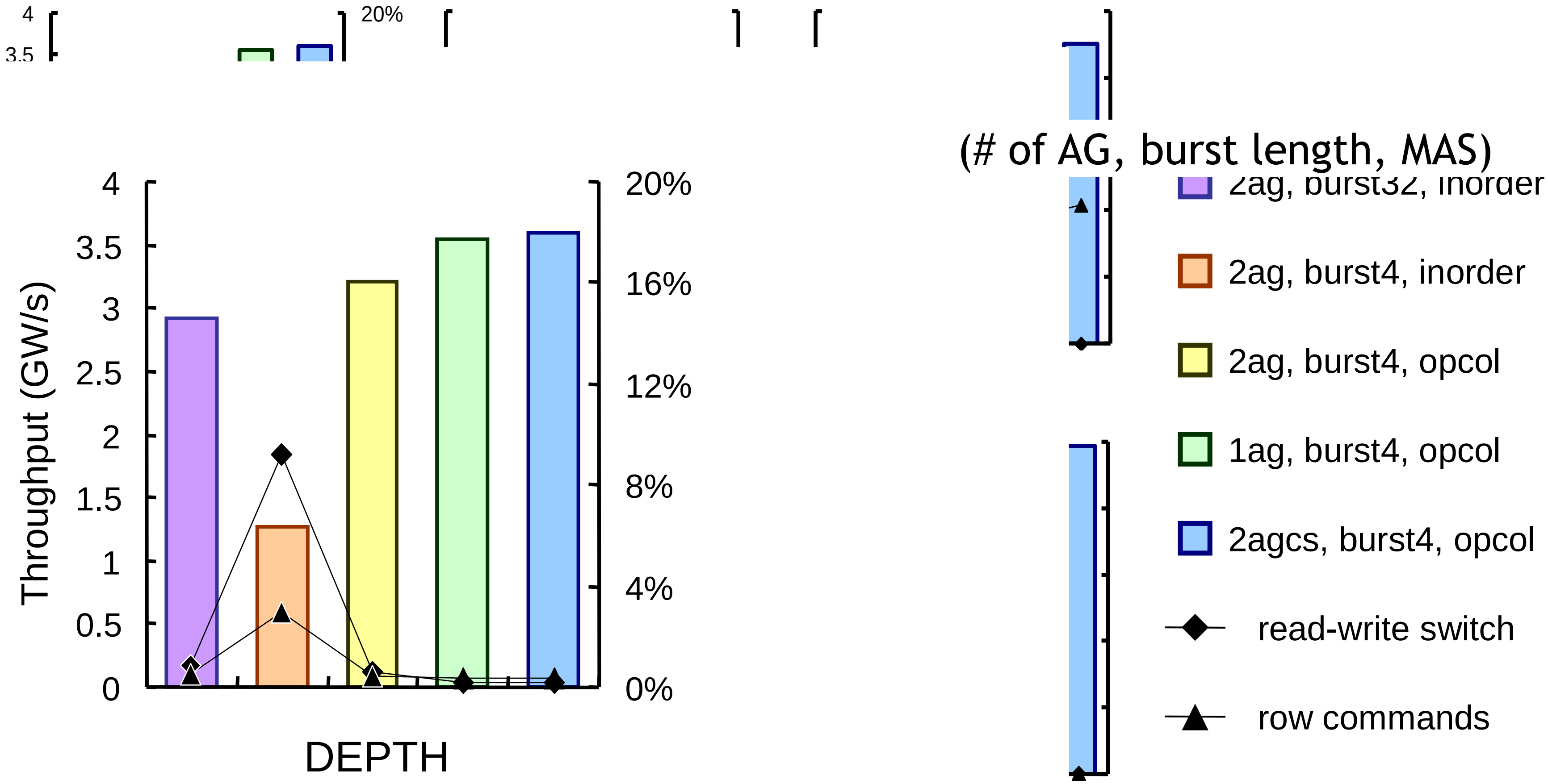
- AG width : 4
- # of MCs : 4
- # of CBEs : 16
- CBEs for channel-split config : 16 per AG
- MAS policy : opcol

A cycle-accurate Imagine simulator is used for performance evaluation



- DRAM burst length : 4 words
- Peak DRAM BW : 4 GW/s
- # of internal DRAM banks : 8
- DRAM typing params : XDR
- Peak DRAM BW : 2
- # of DRAM commands for accessing an inactive row 3

Memory system performance for representative configurations on six apps

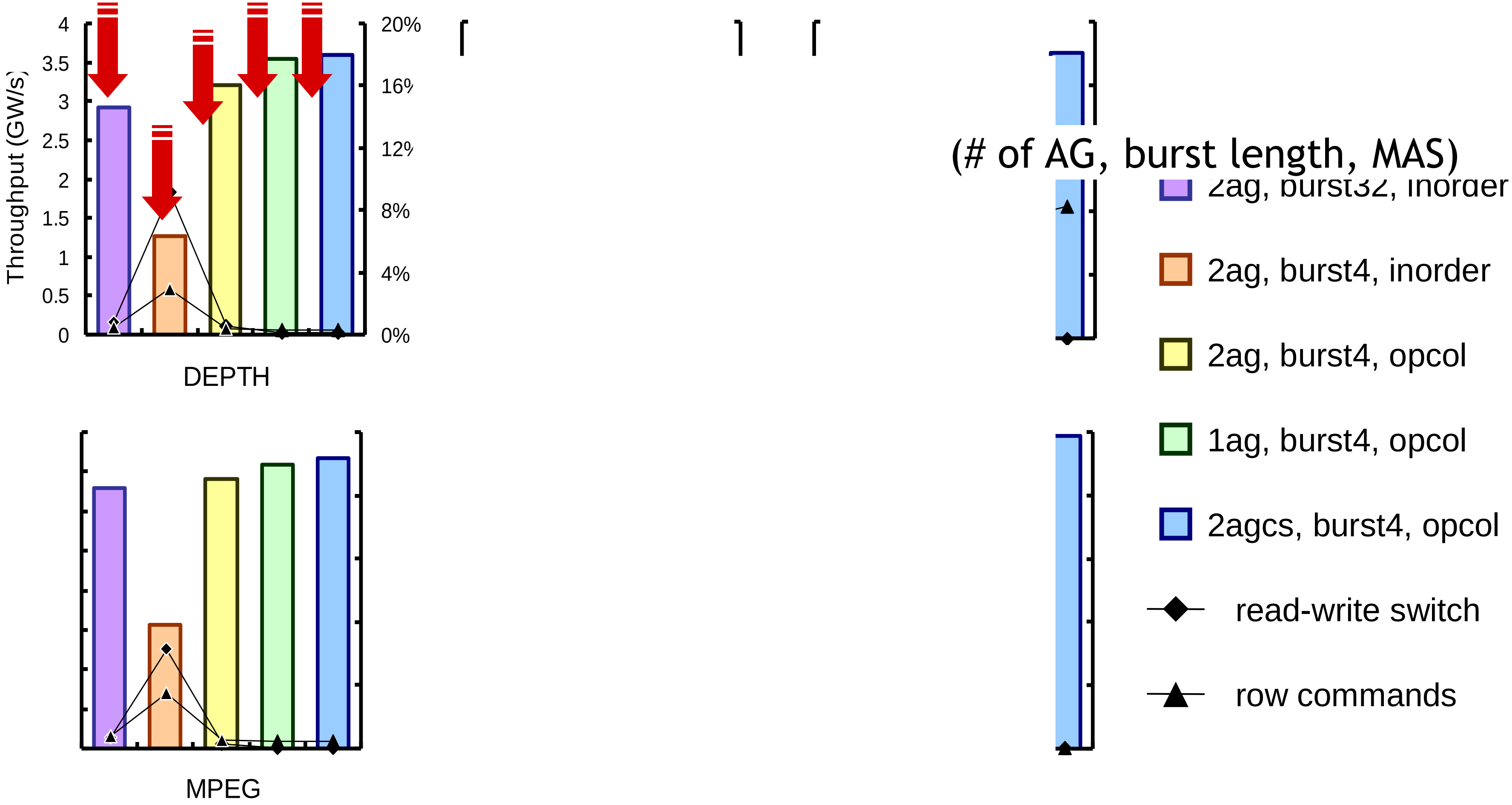


The key memory system related characteristics of six applications

	Average strided			Average indexed				
Application	record size (W)	stream length (W)	stride/ record	record size (W)	stream length (W)	index range	strided access	read access
DEPTH	**1.96**	1802	**1.95**	**1**	1170	**1180**	46.6%	**63.0%**
MPEG	**1**	1515	**1**	**1**	1280	**2309**	90.1%	**70.2%**

DEPTH & MPEG has small record, stride size, and index range

Memory system performance for representative configurations on six apps



(# of AG, burst length, MAS)

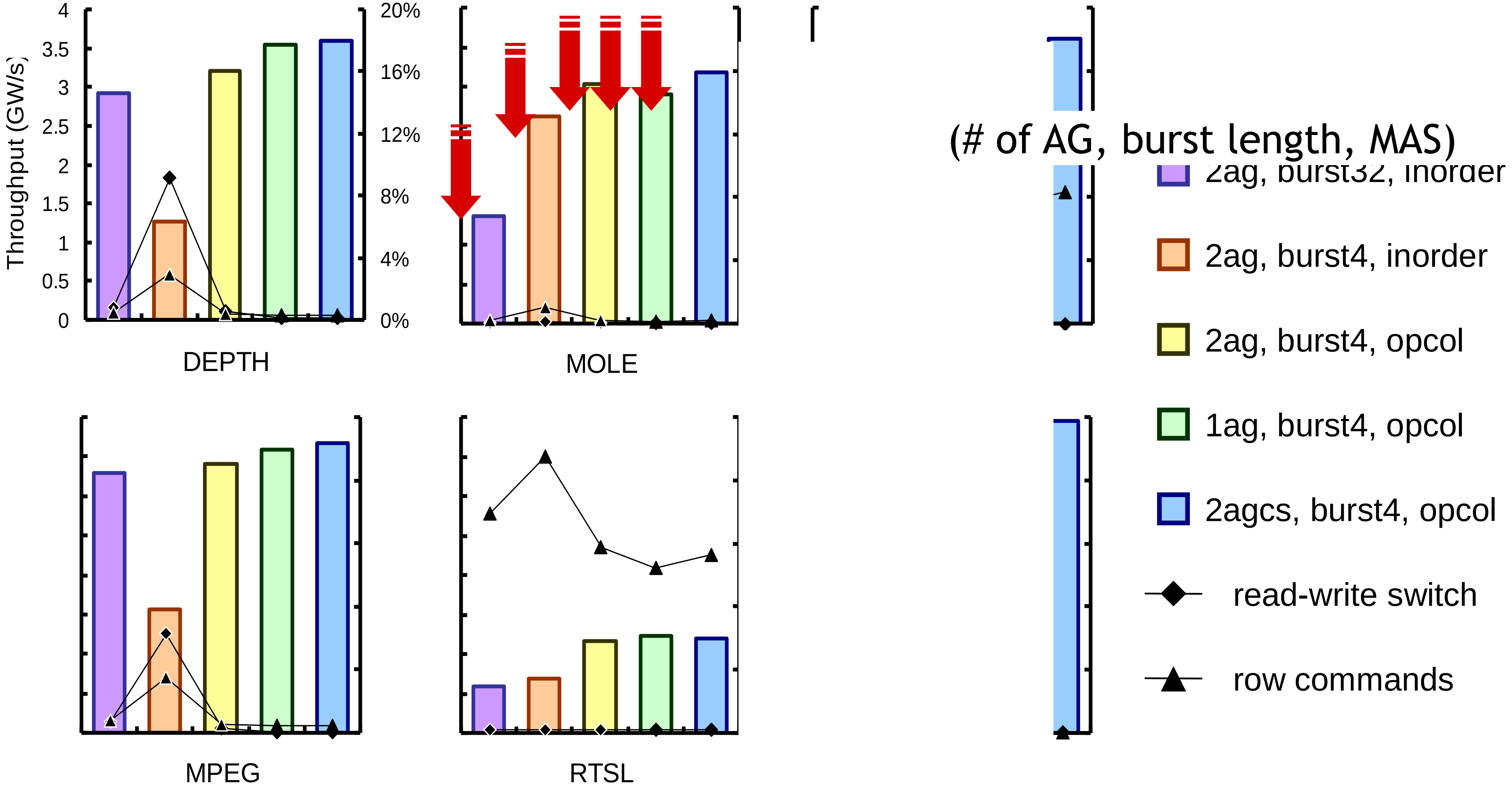
Small record, stride size and index range means high spatial locality between generated requests from an access thread

The key memory system related characteristics of six applications

	Average strided			Average indexed				
Application	record size (W)	stream length (W)	stride/ record	record size (W)	stream length (W)	index range	strided access	read access
DEPTH	1.96	1802	1.95	1	1170	1180	46.6%	63.0%
MPEG	1	1515	1	1	1280	2309	90.1%	70.2%
RTSL	**4**	1170	**4**	1	264	**216494**	**65.1%**	**83.5%**
MOLE	**1**	480	**1**	9	3252	**7190**	**9.9%**	**99.5%**

Streams with small record size and large index ranges lacks spatial locality between generated requests

Memory system performance for representative configurations on six apps



(# of AG, burst length, MAS)

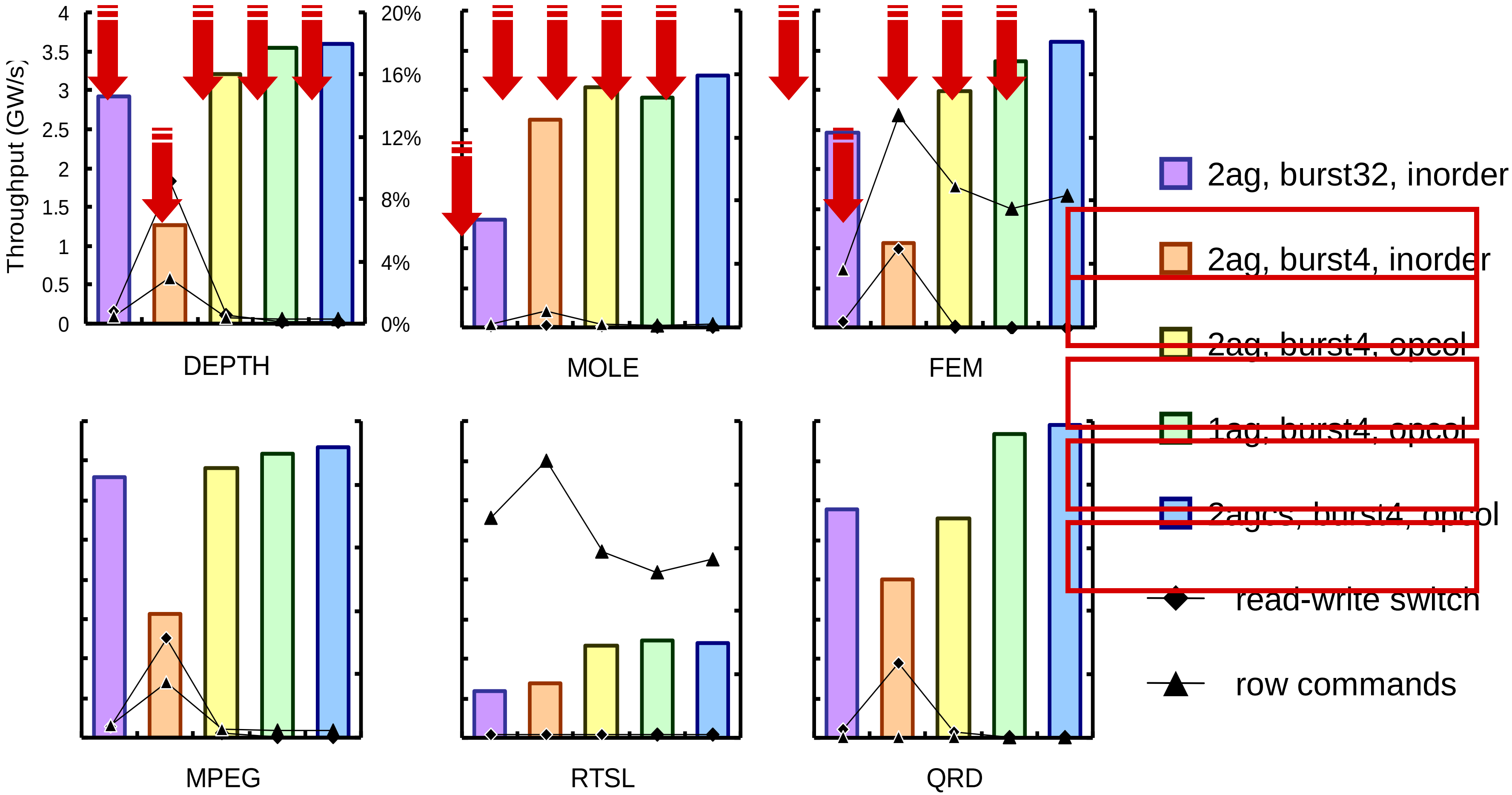
Long bursts hurt memory system performance

The key memory system related characteristics of six applications

Application	Average strided			Average indexed			strided access	read access
	record size (W)	stream length (W)	stride/record	record size (W)	stream length (W)	index range		
DEPTH	1.96	1802	1.95	1	1170	1180	46.6%	63.0%
MPEG	1	1515	1	1	1280	2309	90.1%	70.2%
RTSL	4	1170	4	1	264	216494	65.1%	83.5%
MOLE	1	480	1	9	3252	7190	9.9%	99.5%
QRD	**115**	1053	**350**	N/A	N/A	N/A	100%	**69.0%**

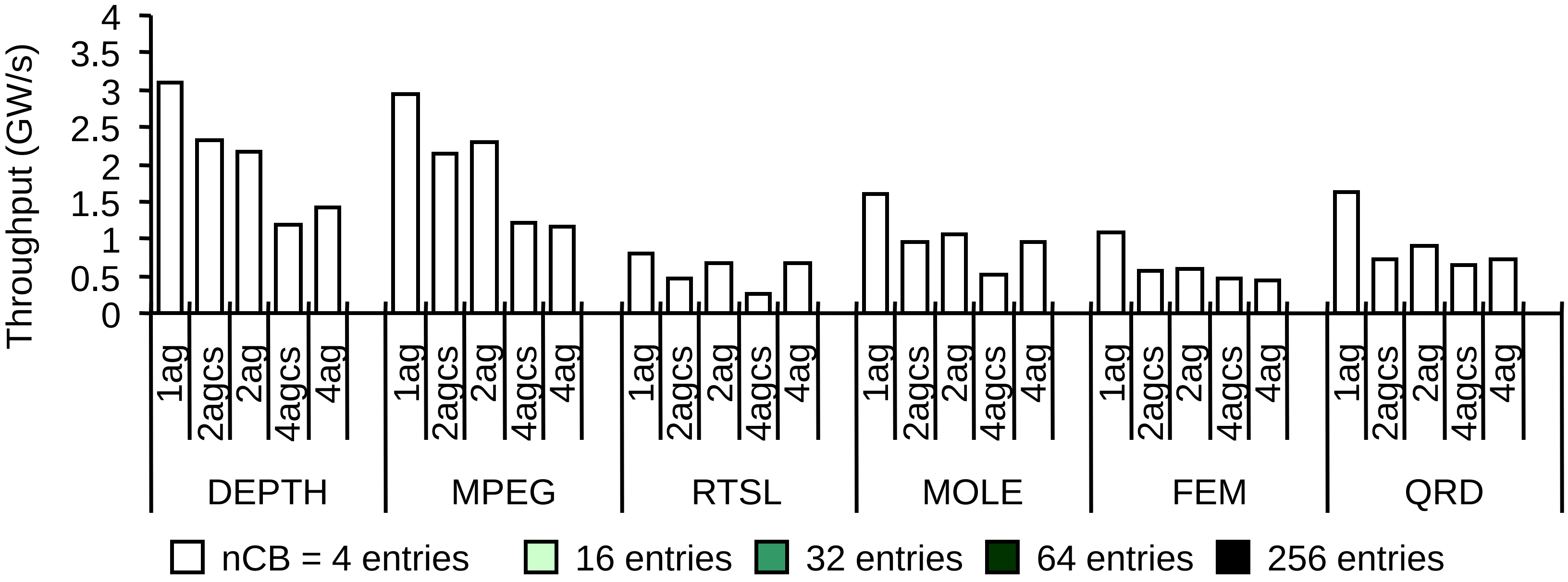
Large record size means high spatial locality in generated requests

Memory system performance for representative configurations on six apps

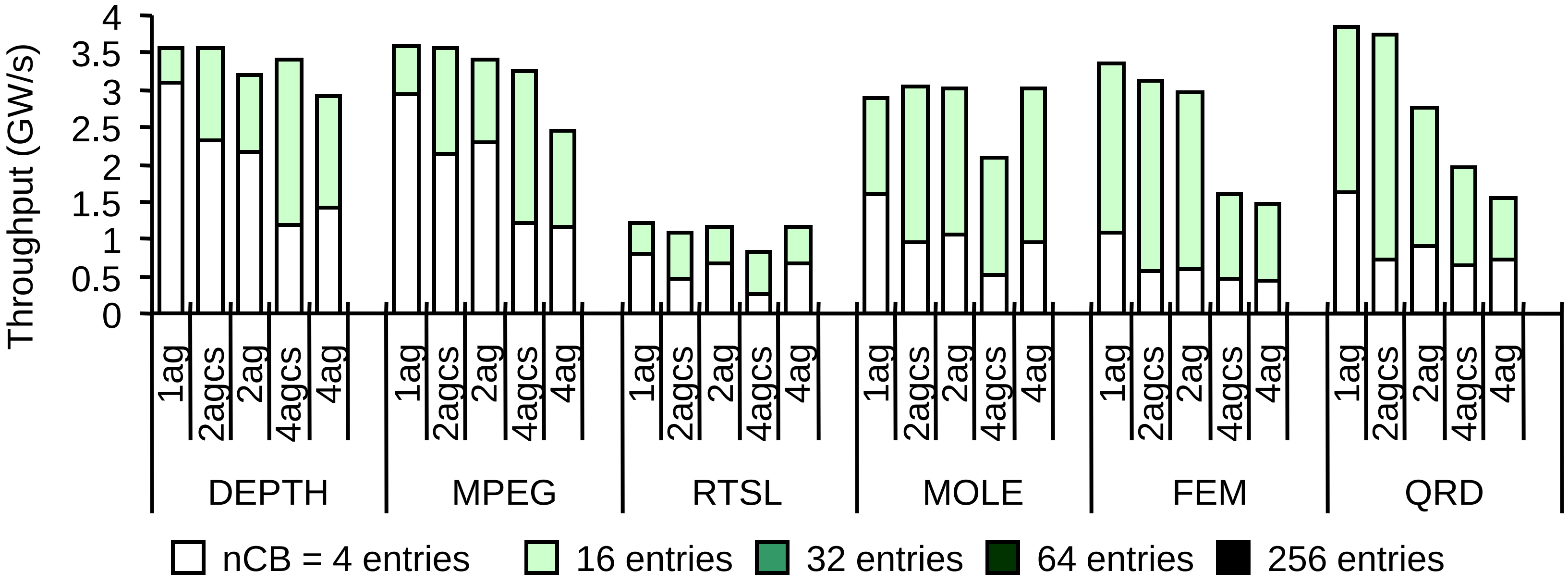


QRD & FEM perform similar to DEPTH & MPEG

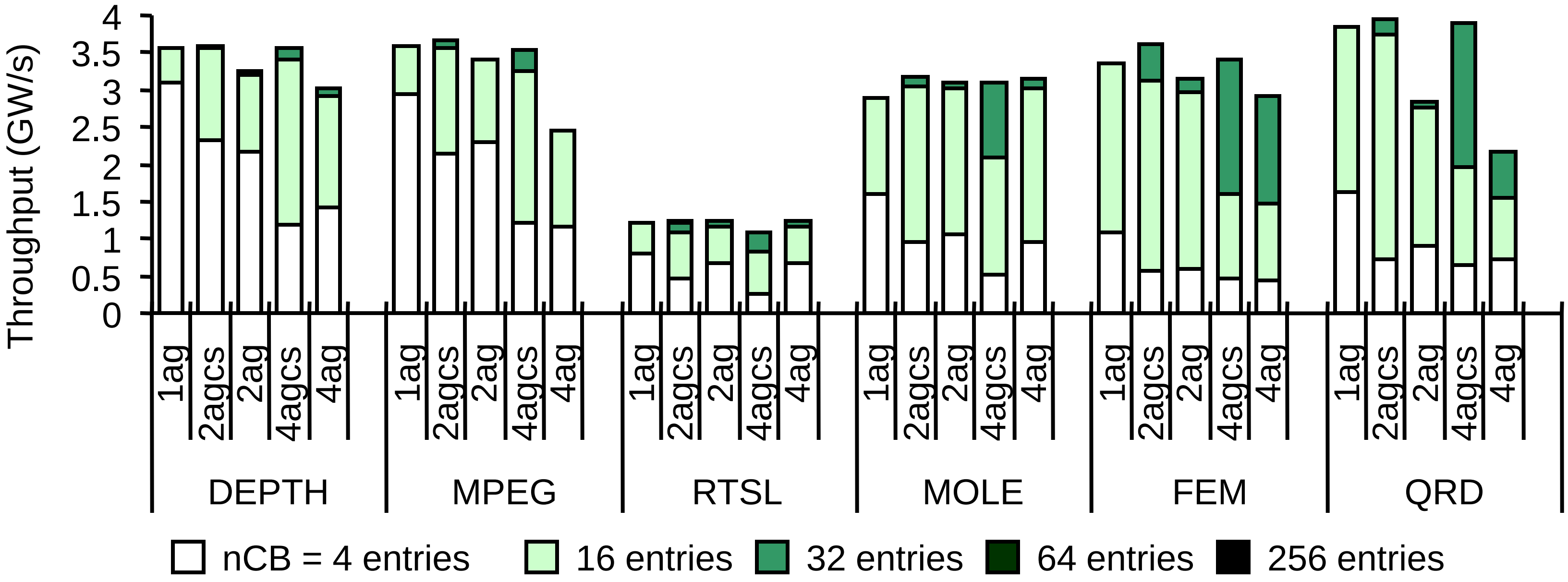
Performance sensitivity to the size of MC buffers



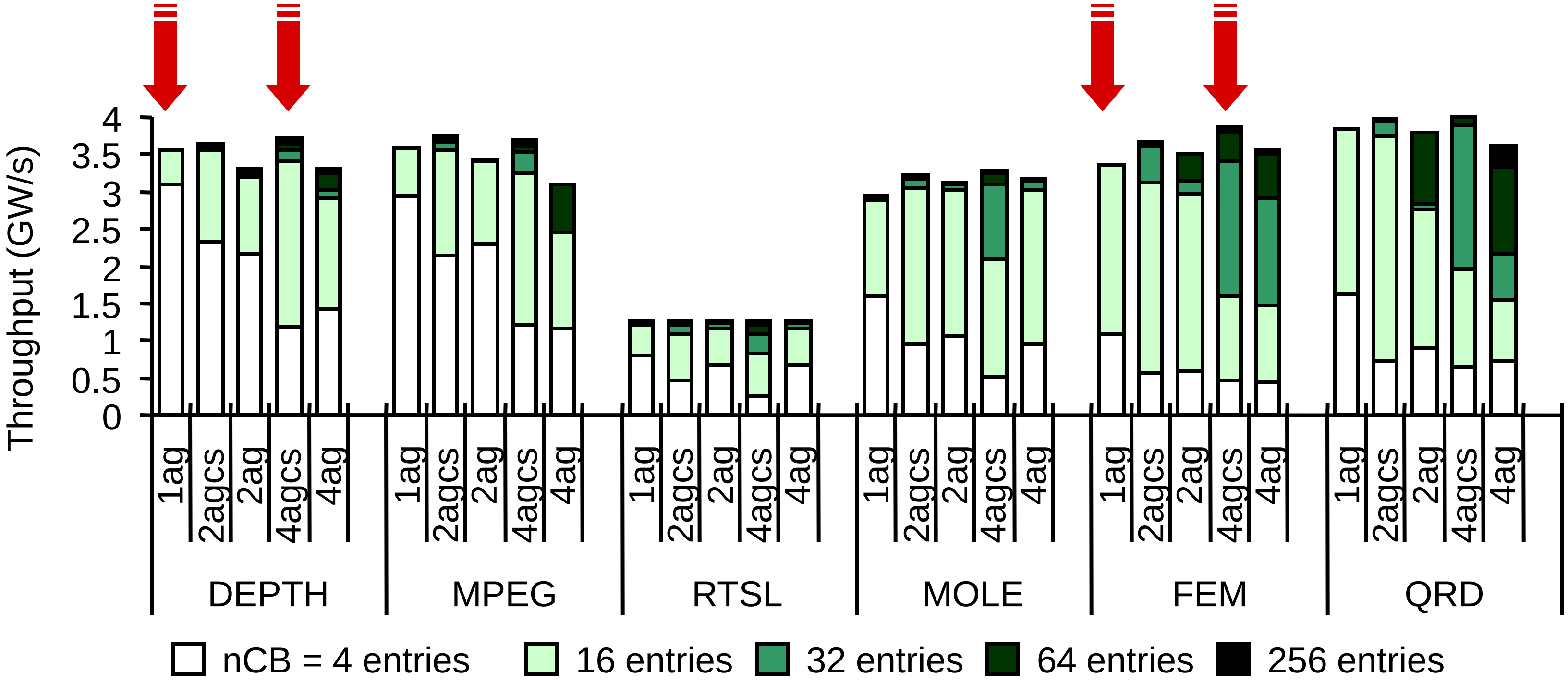
Performance sensitivity to the size of MC buffers



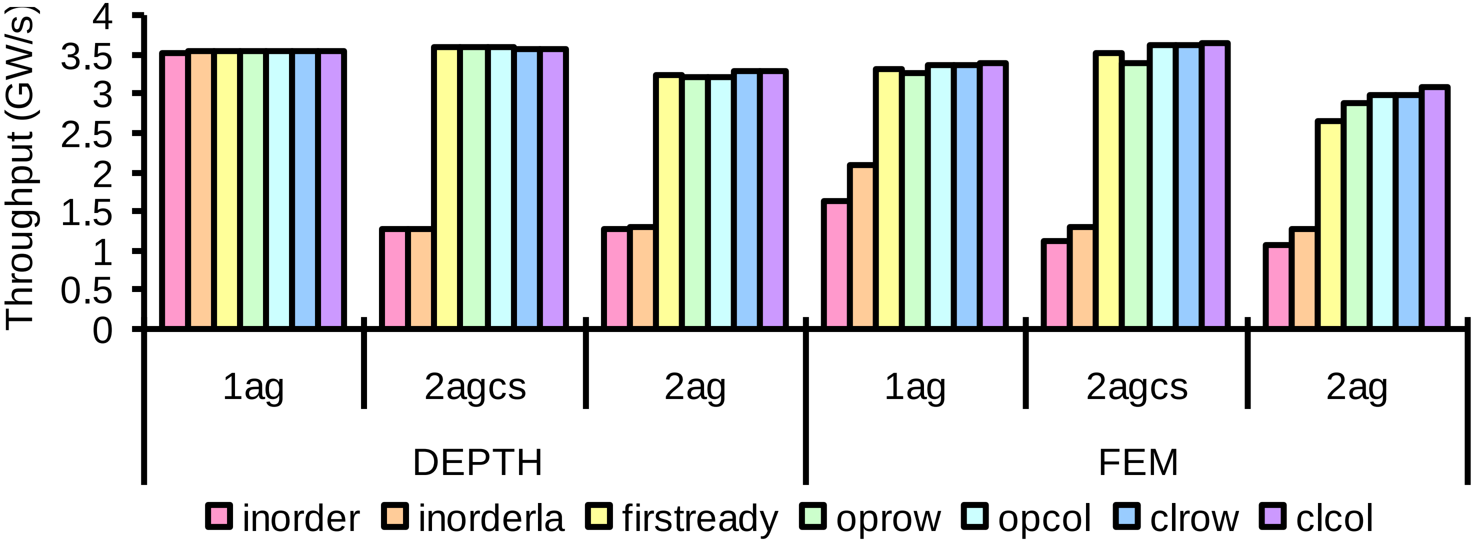
Performance sensitivity to the size of MC buffers



Performance sensitivity to the size of MC buffers rises as the number of AGs is increased



Reordering row & column commands are important, but specific MAS policies are not



Emerging Memory Technology

- New non-volatile storage devices
  - Fine-grained access like DRAM, non-volatile like FLASH
    - No refresh (or almost no refresh)
  - Density equal to or better than DRAM
    - Potentially more scalable than DRAM
  - Higher latencies, especially for writes
  - Higher write energy
  - Possible endurance issues
  - Very similar array structure and interface design to DRAM
- New interface options
  - 3D integration
    - Memory on top of a processor
    - Memory cubes
  - Optical interconnect

Conclusion

- DRAM trends
  - Data BW increases rapidly while latency and cmd BW improve slowly
    - DRAM access granularity grows
    - Throughput is very sensitive to access patterns
  - Locality must be exploited
    - To minimize internal bank conflicts and read-write turnaround penalties
- Memory system design space
  - Number of AGs – inter-thread vs. intra-thread parallelism
  - Load balance across MCs – channel interleaving, multiple threads, and AG width
  - The amount of MC buffering determines the window size of MAS
- Design suggestions
  - A single wide AG exploits DRAM locality well
  - Channel-split mechanism exploits locality and balances loads across multiple channels simultaneously at the cost of additional hardware