Get Out of the Valley: Power-Efficient Address Mapping for GPUs

Yuxi Liu, *Xia Zhao, Magnus Jahre, Zhenlin Wang, Xiaolin Wang, Yingwei Luo, Lieven Eeckhout*

Talk Outline

- Background and Motivation
- Window-based Entropy Model
- Analyzing Entropy Distribution for GPU Workloads
- PAE and FAE Address Mapping Schemes
- Evaluation

Shannon Entropy

- It quantifies *the amount of information* contained in a sequence of values.

$$H(p_1, p_2, \ldots, p_v) = -\sum_{i=1}^{v} p_i \log_v p_i.$$

Shannon Entropy

- It quantifies *the amount of information* contained

  in a sequence of values.

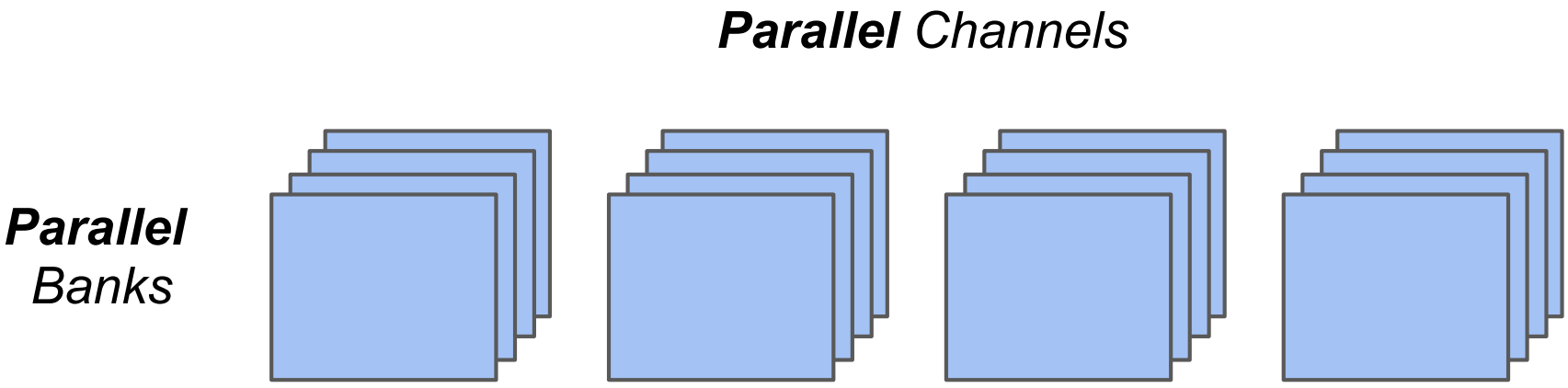
$$H(p_1, p_2, \ldots, p_v) = -\sum_{i=1}^{v} p_i \log_v p_i.$$

Informally, a value that changes

1. **frequently** contains lots of information and has **high** entropy

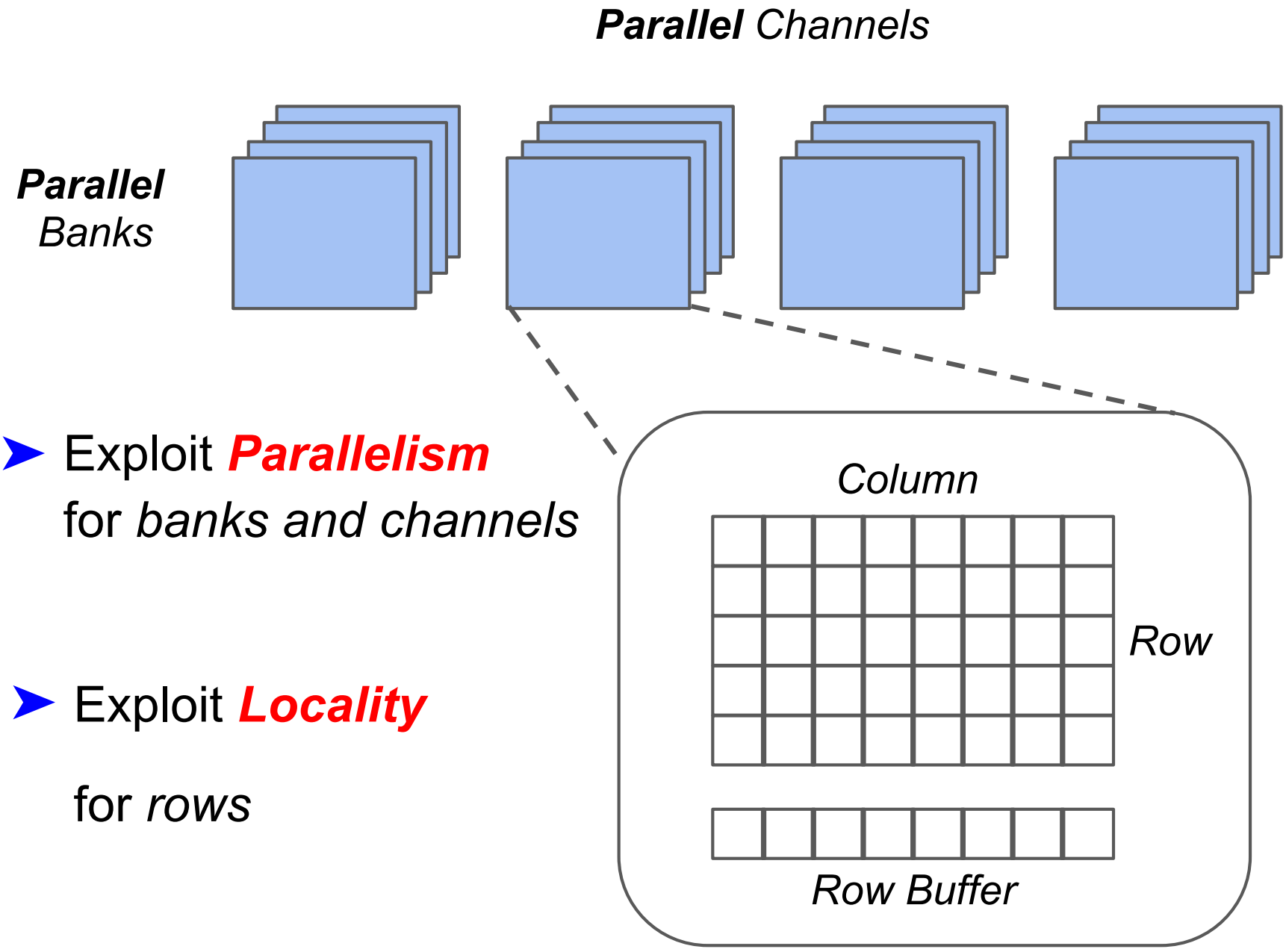
1. **rarely** contains little information and has **low** entropy

DRAM Architecture



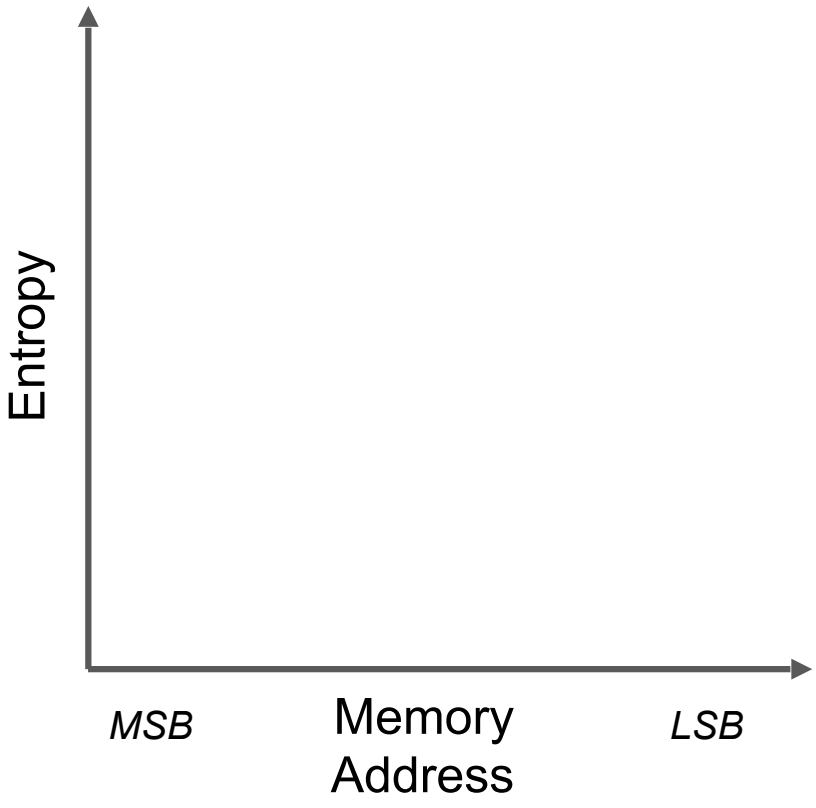
- Exploit ***Parallelism***
  for *banks and channels*

DRAM Architecture

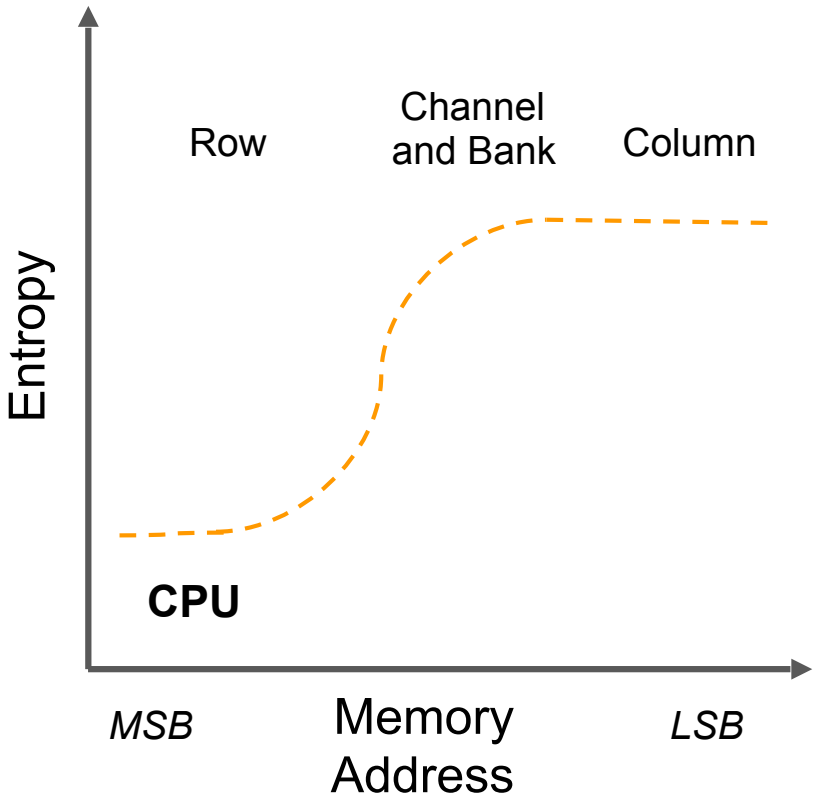


- Exploit ***Parallelism*** for *banks and channels*
- Exploit ***Locality*** for *rows*

Address Bit Entropy Distribution

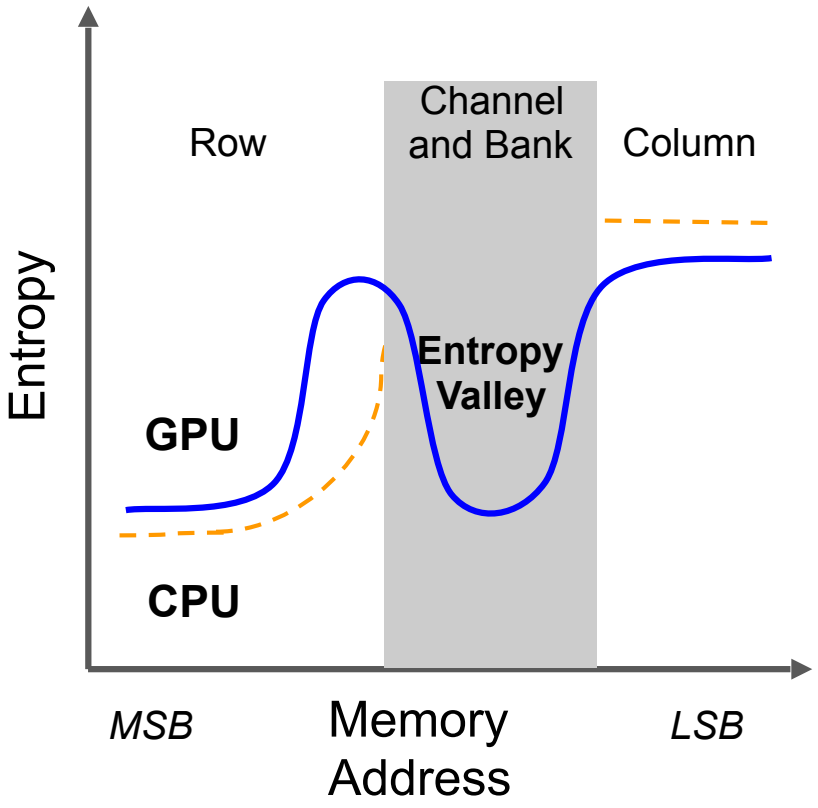


Address Bit Entropy Distribution



- Per-bit entropy for addresses in **CPUs** is monotonic

Address Bit Entropy Distribution



- Per-bit entropy for addresses in **CPUs** is monotonic
- Entropy Valley for addresses in **GPUs**

Why Entropy Valley for GPUs?

- *GPU thread organization*: multi-dimensional structures
  - Grid, Block, Warp, Thread
- *DRAM hardware organization*: multi-dimensional structures
  - Channel, Bank, Row, Column

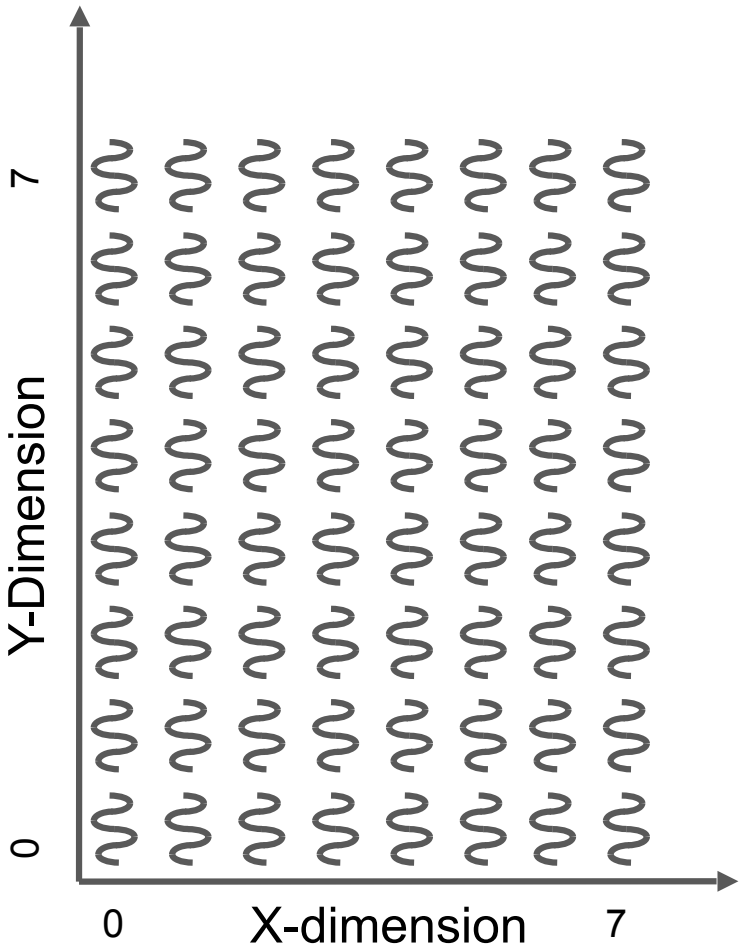
Why Entropy Valley for GPUs?

- *GPU thread organization*: multi-dimensional structures
  - Grid, Block, Warp, Thread
- *DRAM hardware organization*: multi-dimensional structures
  - Channel, Bank, Row, Column

Once these two organizations combine unfavorably,

Entropy valleys will occur

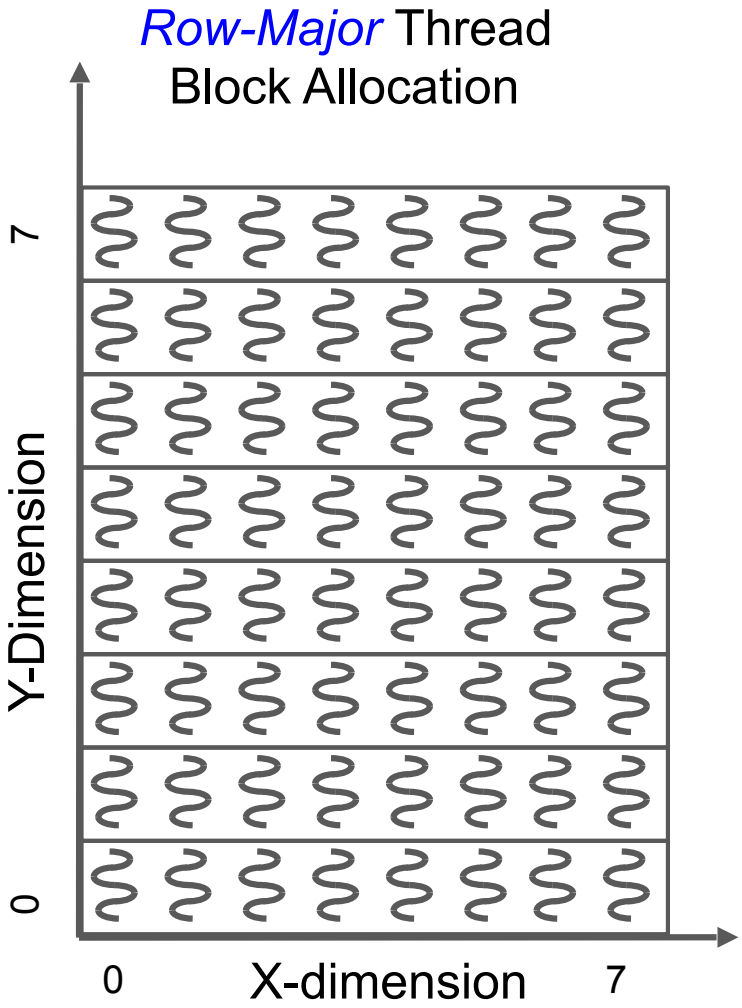
GPU Address Behavior (Case 1)



GPU Address Behavior (Case 1)

Dimension-related index

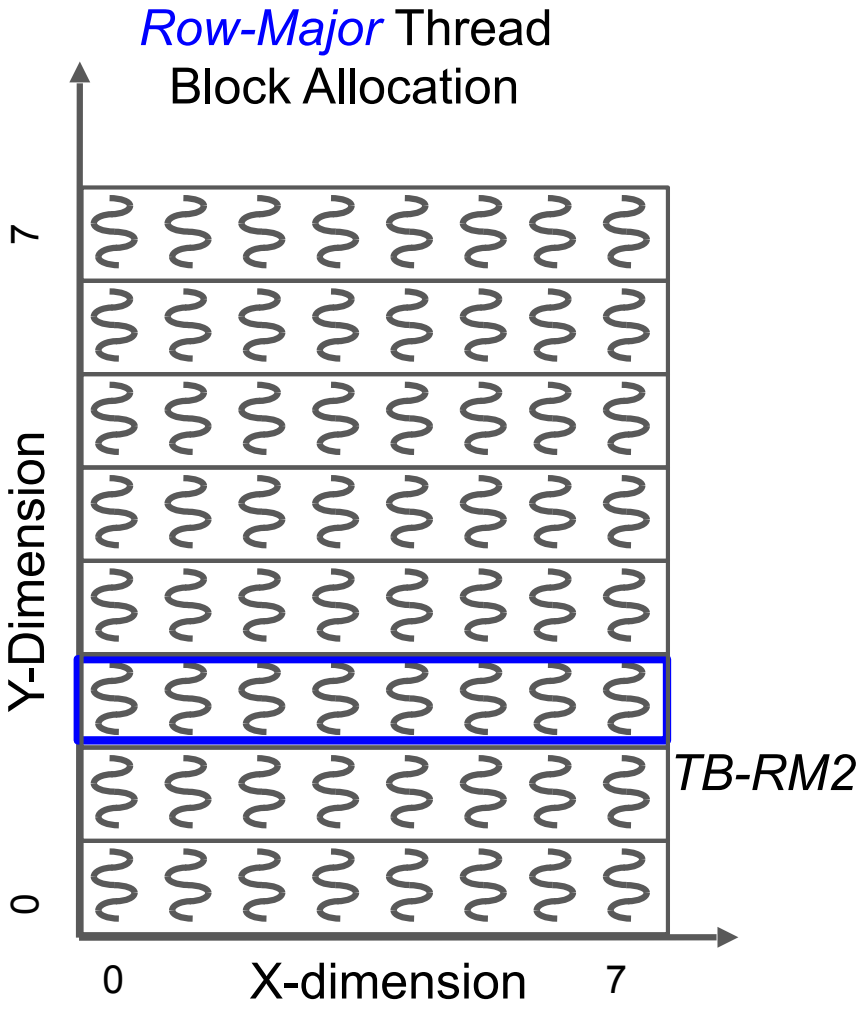
Example: *i = threadIdx.y * blockDim.x + threadIdx.x*



GPU Address Behavior (Case 1)

Dimension-related index

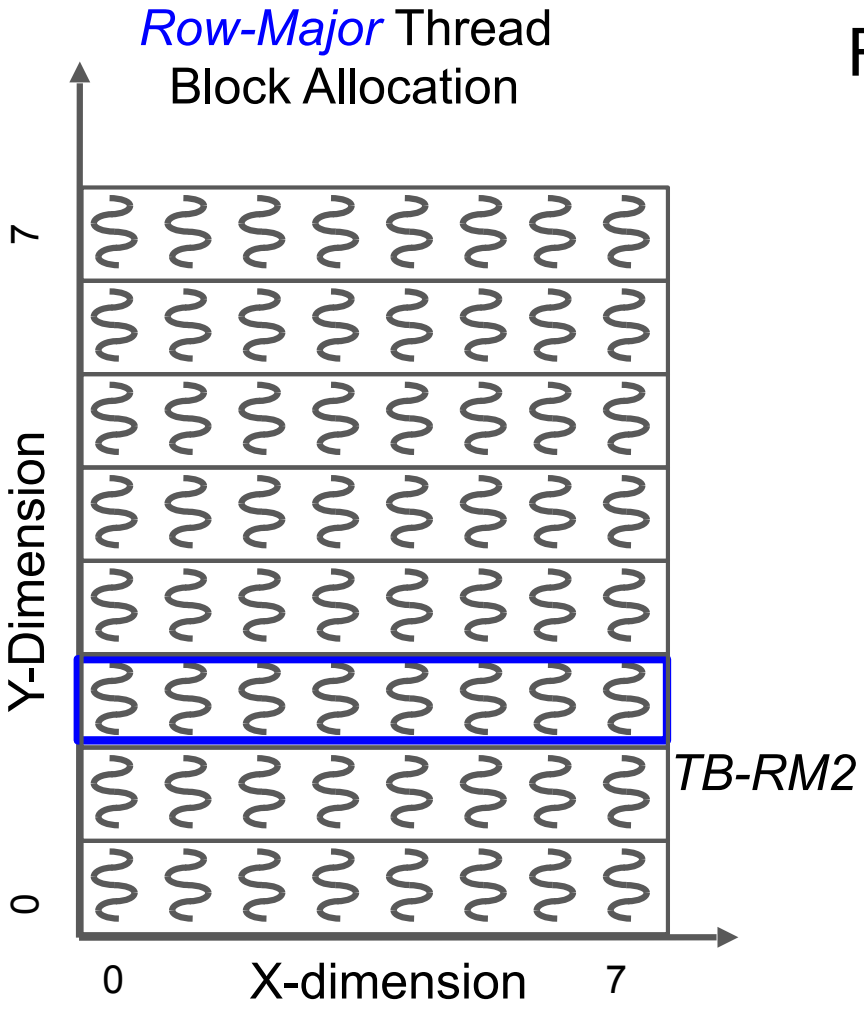
Example: *i = threadIdx.y * blockDim.x + threadIdx.x*



GPU Address Behavior (Case 1)

Dimension-related index

Example: *i = threadIdx.y * blockDim.x + threadIdx.x*

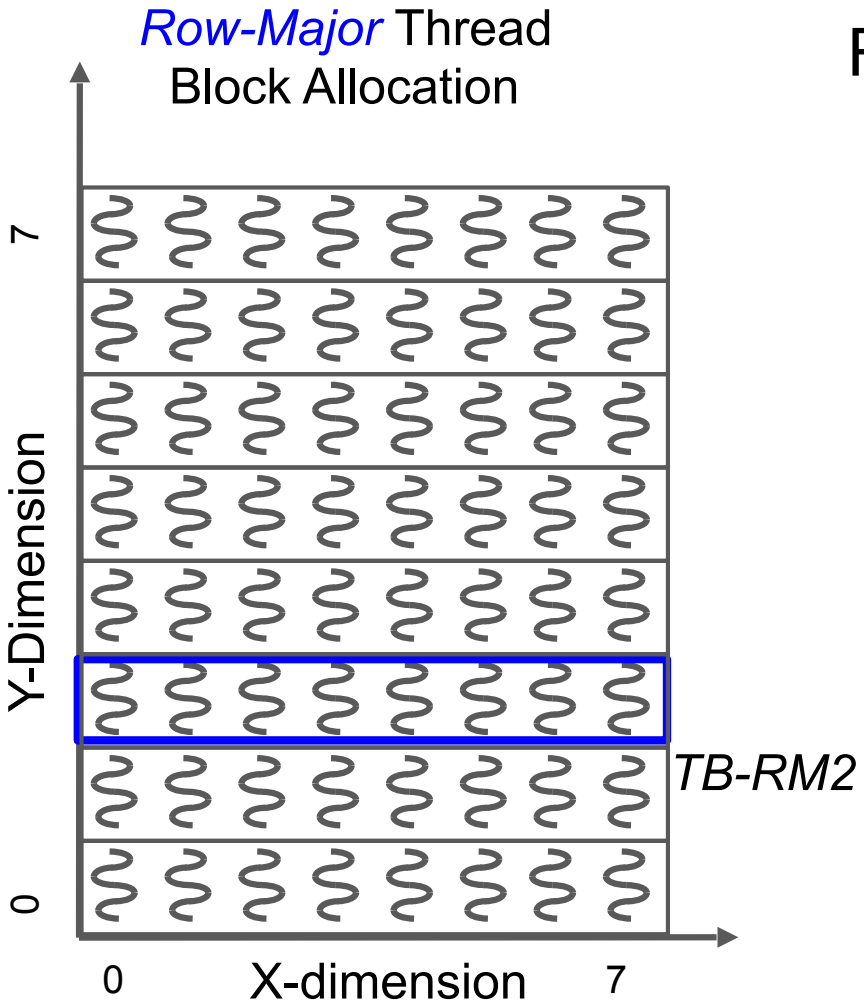


Req. ID	[y, x]	Address
8	[2,7]	…0100 11 ...
7	[2,6]	…0100 10 ...
6	[2,5]	…0100 01 ...
5	[2,4]	…0100 00 ...
4	[2,3]	…0101 11 ...
3	[2,2]	…0101 10 ...
2	[2,1]	…0101 01 ...
1	[2,0]	...0101 00 ...

GPU Address Behavior (Case 1)

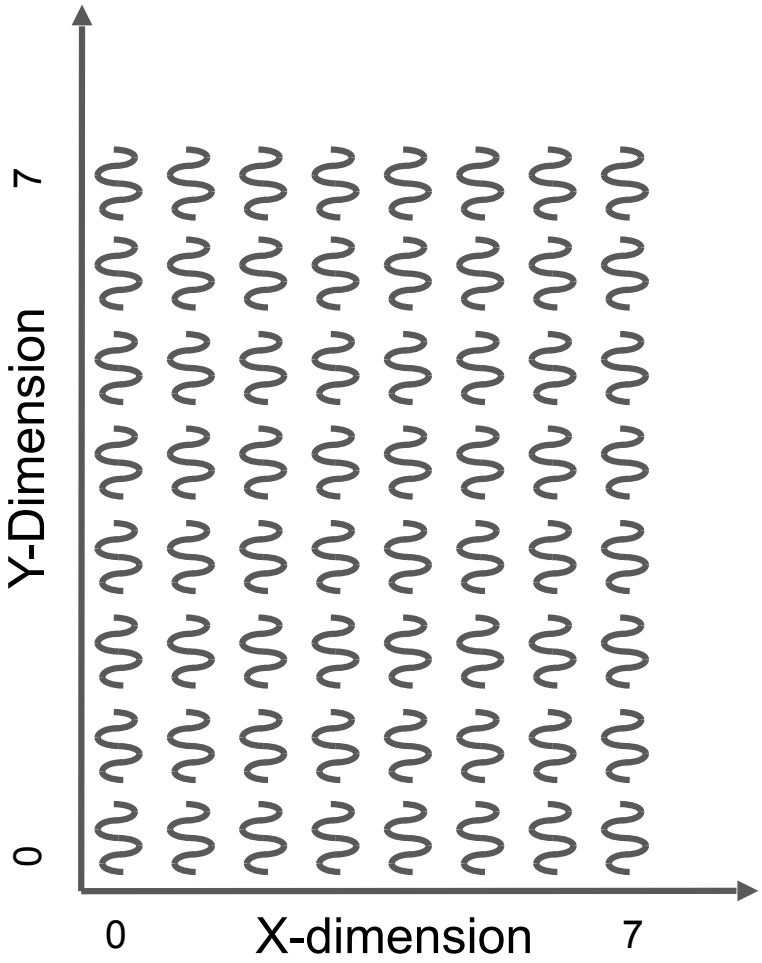
Dimension-related index

Example: *i = threadIdx.y * blockDim.x + threadIdx.x*



Req. ID	[y, x]	Address	Channel Distribution
8	[2,7]	…0100 11 ...	
7	[2,6]	…0100 10 ...	Ch-0: 1, 5
6	[2,5]	…0100 01 ...	Ch-1: 2, 6
5	[2,4]	…0100 00 ...	Ch-2: 3, 7
4	[2,3]	…0101 11 ...	Ch-3: 4, 8
3	[2,2]	…0101 10 ...	
2	[2,1]	…0101 01 ...	
1	[2,0]	...0101 00 ...	

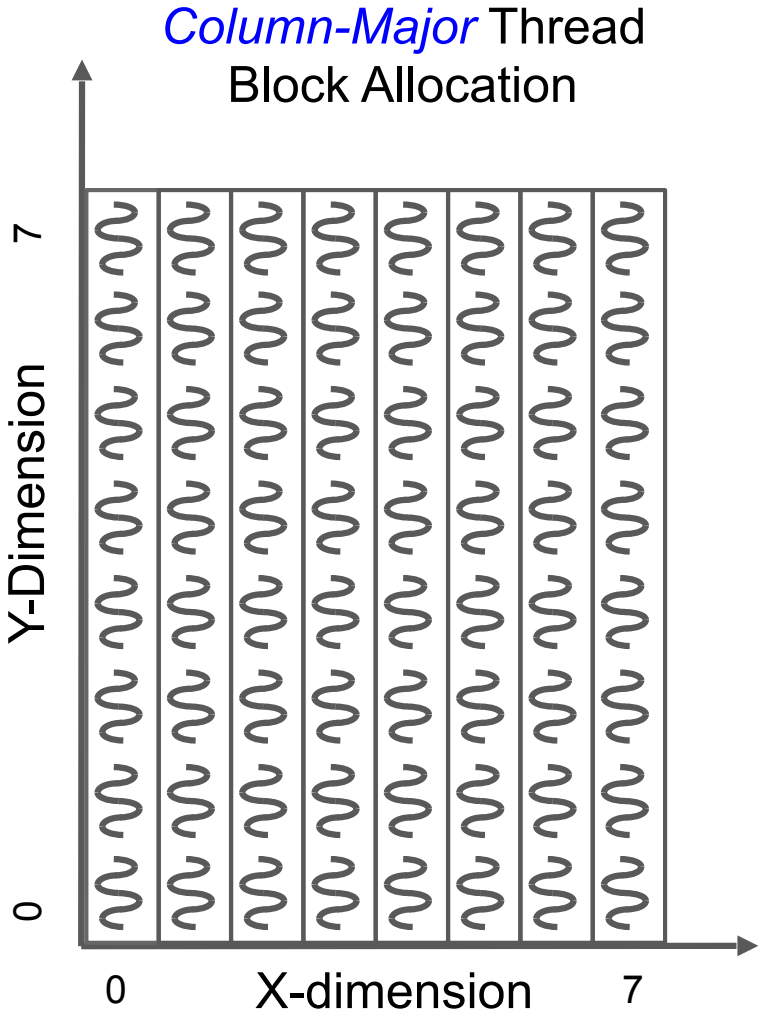
GPU Address Behavior (Case 2)



GPU Address Behavior (Case 2)

Dimension-related index

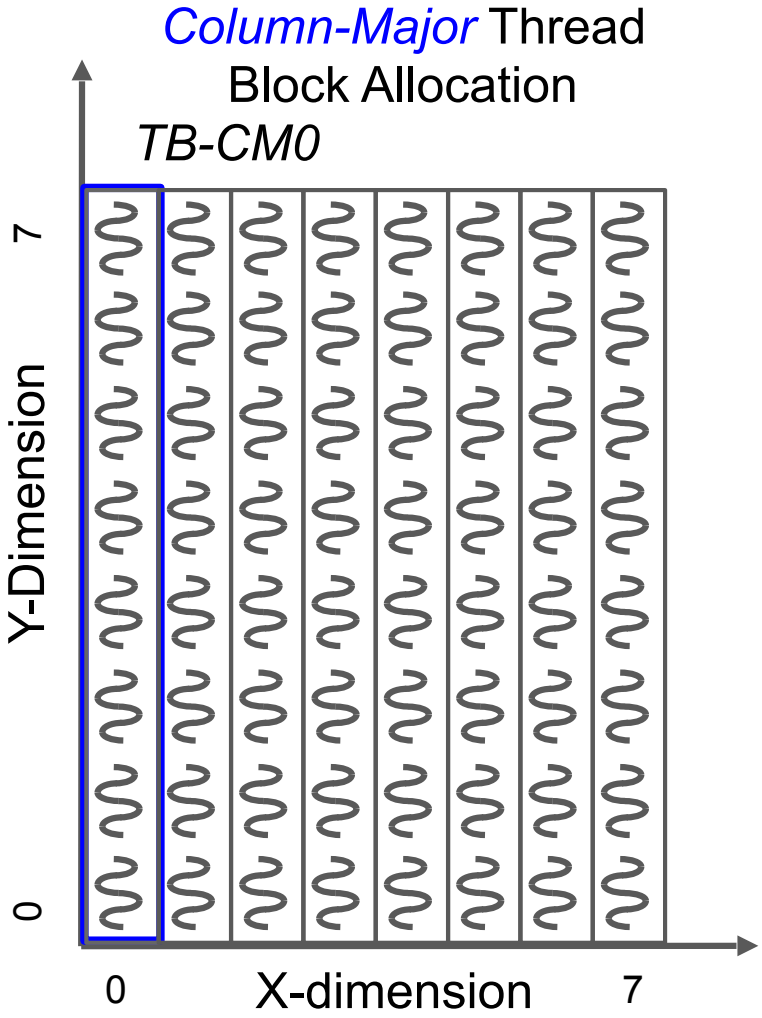
Example: *i = threadIdx.x * blockDim.y + threadIdx.y*



GPU Address Behavior (Case 2)

Dimension-related index

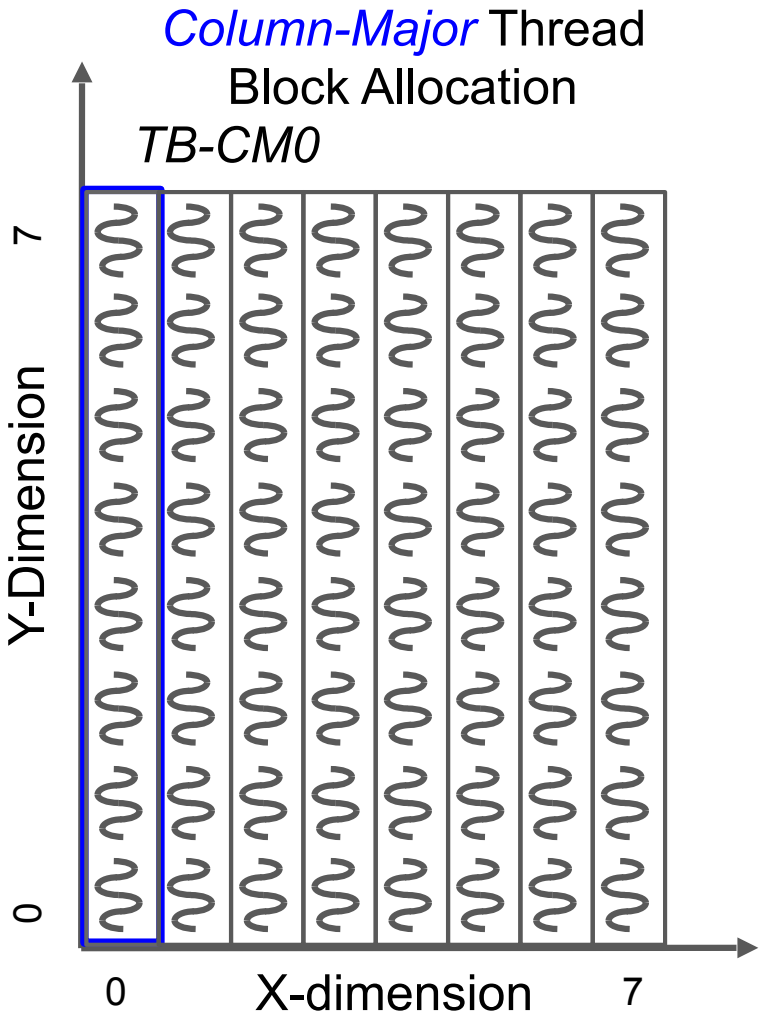
Example: *i = threadIdx.x * blockDim.y + threadIdx.y*



GPU Address Behavior (Case 2)

Dimension-related index

Example: *i = threadIdx.x * blockDim.y + threadIdx.y*

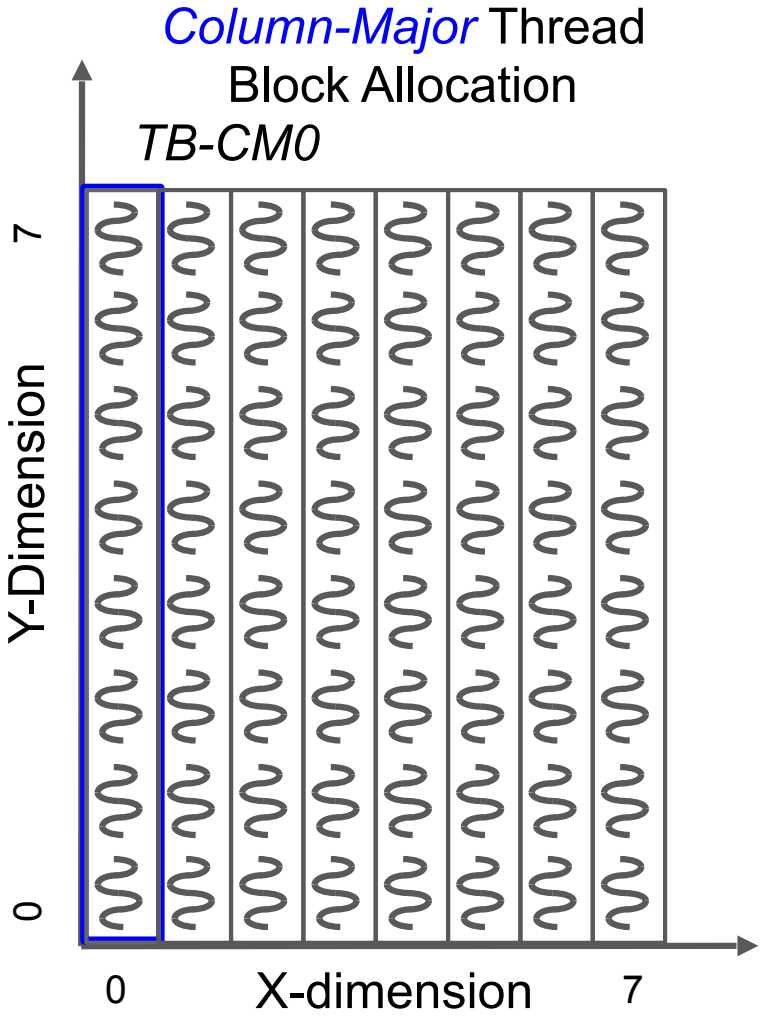


Req. ID	[y, x]	Address
8	[7,0]	…1110 00 ...
7	[6,0]	…1100 00 ...
6	[5,0]	…1010 00 ...
5	[4,0]	…1000 00 ...
4	[3,0]	…0110 00 ...
3	[2,0]	…0100 00 ...
2	[1,0]	…0010 00 ...
1	[0,0]	...0000 00 ...

GPU Address Behavior (Case 2)

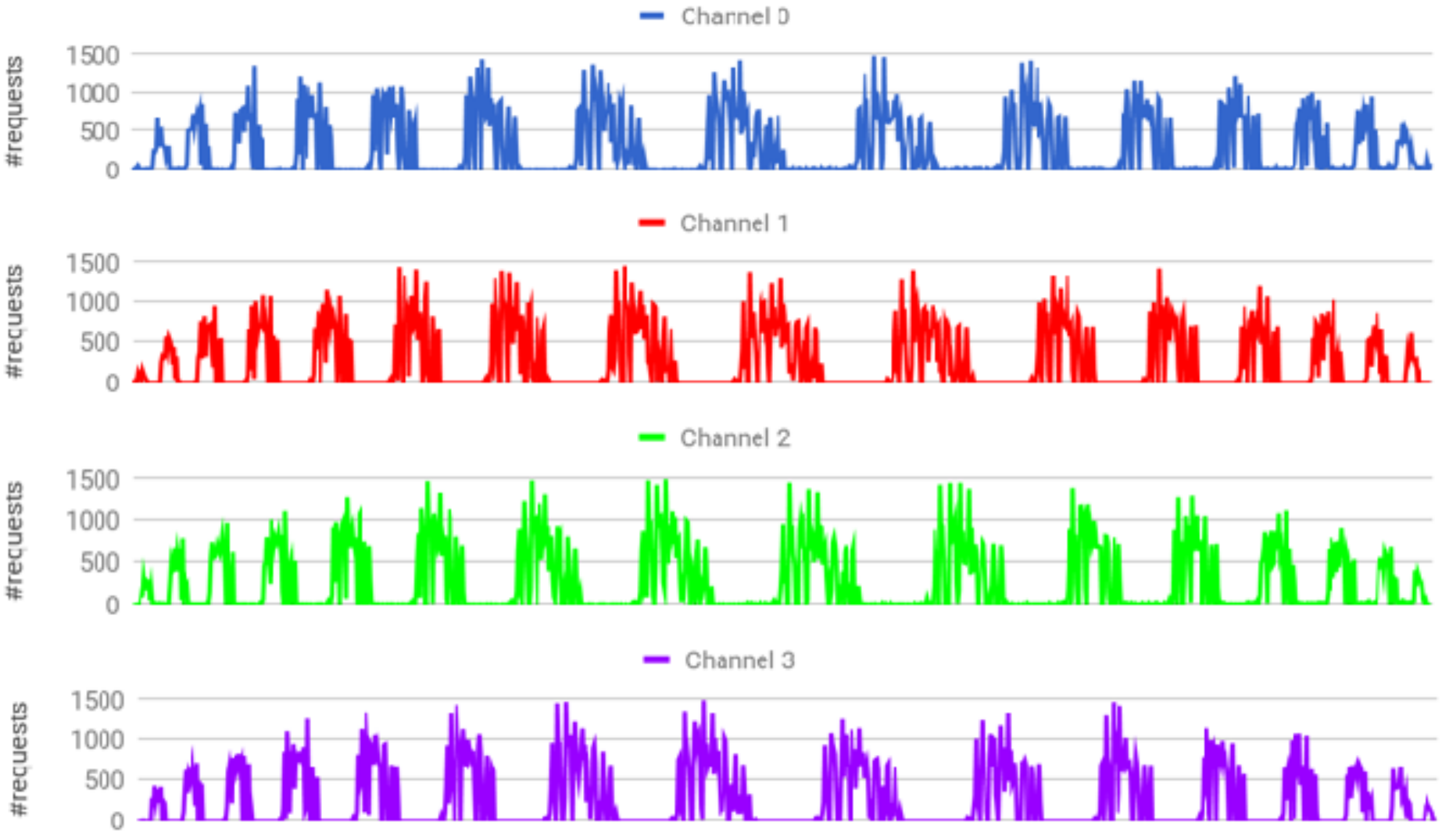
Dimension-related index

Example: *i = threadIdx.x * blockDim.y + threadIdx.y*



Req. ID	[y, x]	Address	Channel Distribution
8	[7,0]	…1110 00 ...	
7	[6,0]	…1100 00 ...	Ch-0: 1 to 8
6	[5,0]	…1010 00 ...	
5	[4,0]	…1000 00 ...	Ch-1: None
4	[3,0]	…0110 00 ...	Ch-2: None
3	[2,0]	…0100 00 ...	
2	[1,0]	…0010 00 ...	Ch-3: None
1	[0,0]	...0000 00 ...	

Channel Imbalance of NW benchmark



GPU Workloads' Entropy

How to quantify the address entropy for GPU workloads?

- CPU's Entropy
  - Memory request ordering
  - Bit Flip Rate

- GPU's Entropy
  - Memory requests are highly *interleaved*
  - Requests from single TB co-exist : *Intra-TB Entropy*
  - Requests from concurrent TBs co-exist : *Inter-TB Entropy*

GPU Workloads' Entropy

- *Window-based Entropy*
  - Motivation: capture coexisting memory requests
  - Accounts for both *Intra- and Inter-TB entropy*

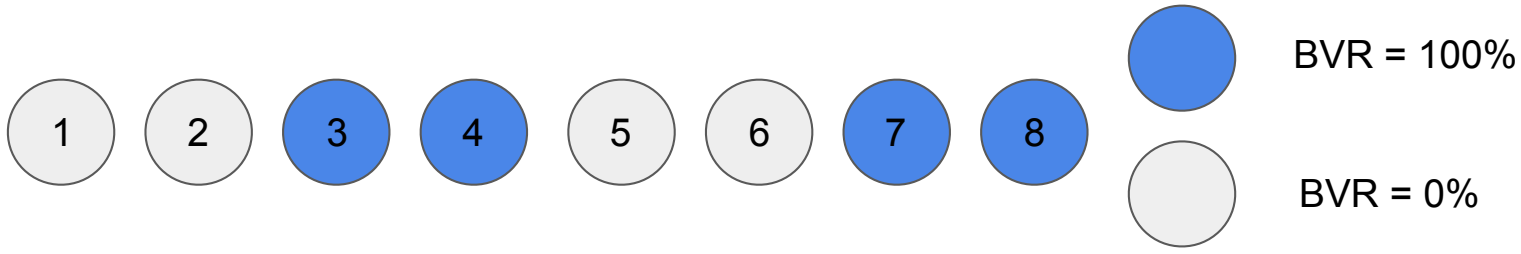
GPU Workloads' Entropy

- *Window-based Entropy*
  - Motivation: capture coexisting memory requests
  - Accounts for both *Intra- and Inter-TB entropy*

$$H^* = \frac{\sum_{i=1}^{n-w+1} H_i^W(p_1^{\mathrm{BVR}}, p_2^{\mathrm{BVR}}, \ldots, p_v^{\mathrm{BVR}})}{n-w+1}.$$

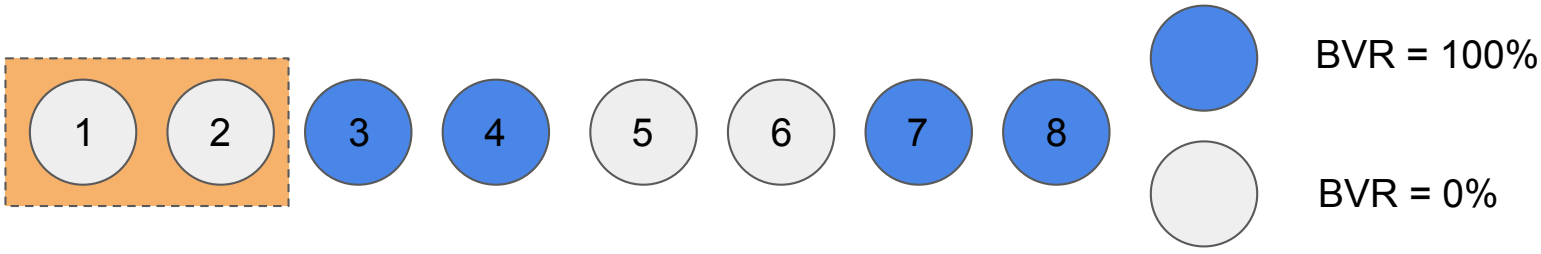
- BVR: Bit Value Rate
- p_*^{BVR} : probability of each BVR value
- W : window size
- H_i^W : entropy value of Window-i
- H^* : the average entropy of all windows

Window-based Entropy



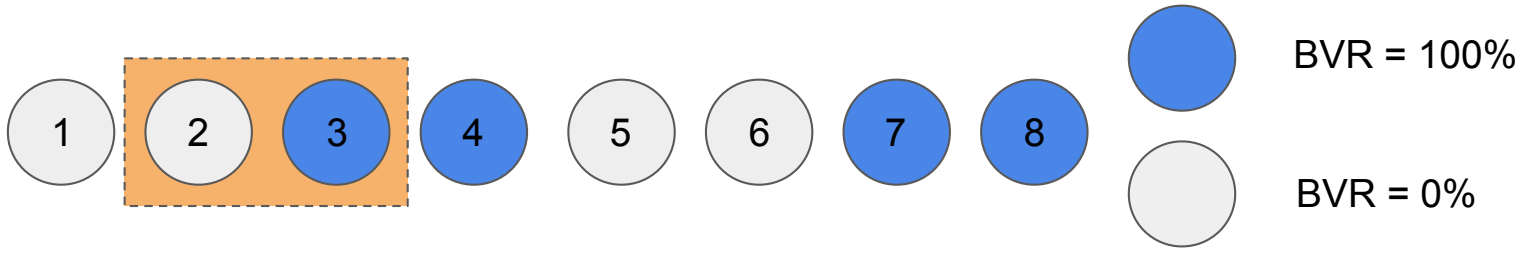
Window #							
#BVR 100% TBs							
#BVR 0% TBs							
Window Entropy							

Window-based Entropy #1



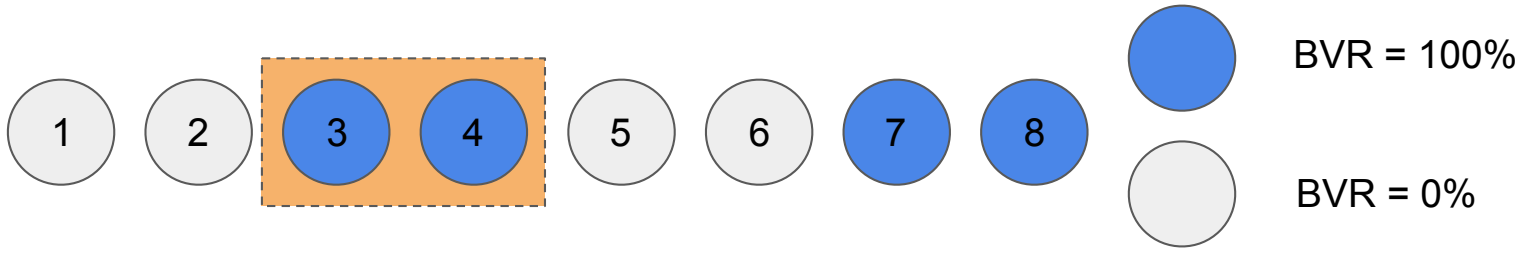
Window #	1						
#BVR 100% TBs	0						
#BVR 0% TBs	2						
Window Entropy	0						

Window-based Entropy #2



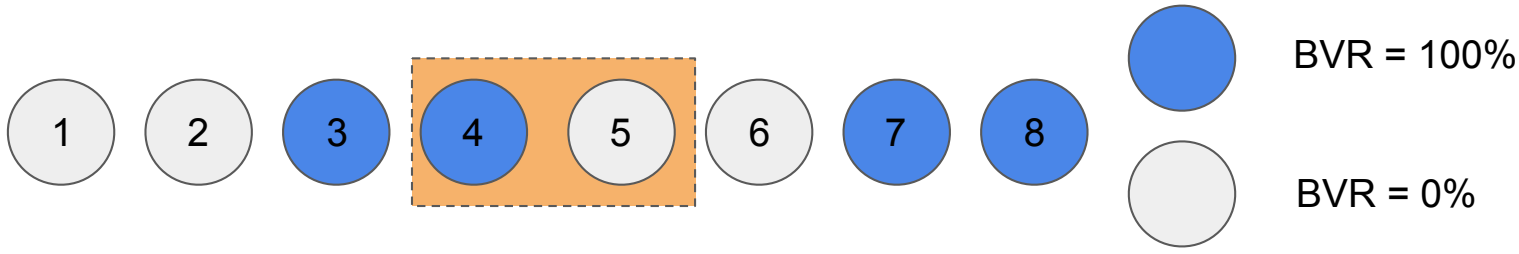
Window #	1	2					
#BVR 100% TBs	0	1					
#BVR 0% TBs	2	1					
Window Entropy	0	1					

Window-based Entropy #3



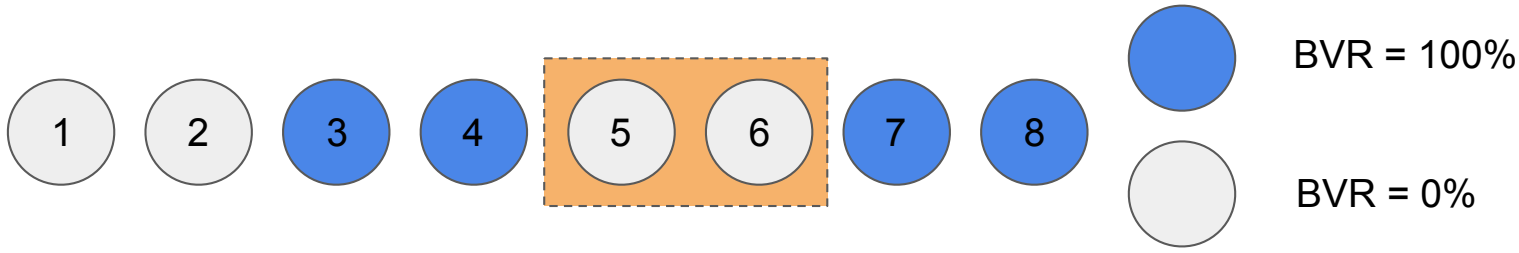
Window #	1	2	3				
#BVR 100% TBs	0	1	2				
#BVR 0% TBs	2	1	0				
Window Entropy	0	1	0				

Window-based Entropy #4



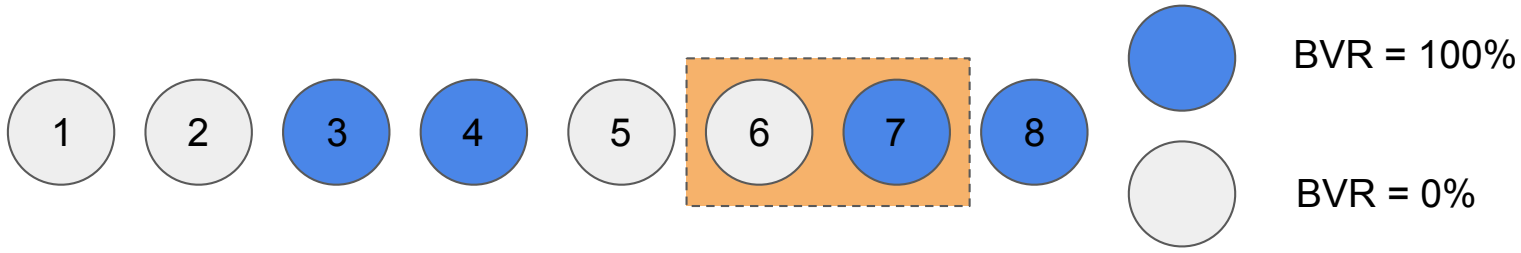
Window #	1	2	3	4			
#BVR 100% TBs	0	1	2	1			
#BVR 0% TBs	2	1	0	1			
Window Entropy	0	1	0	1			

Window-based Entropy #5



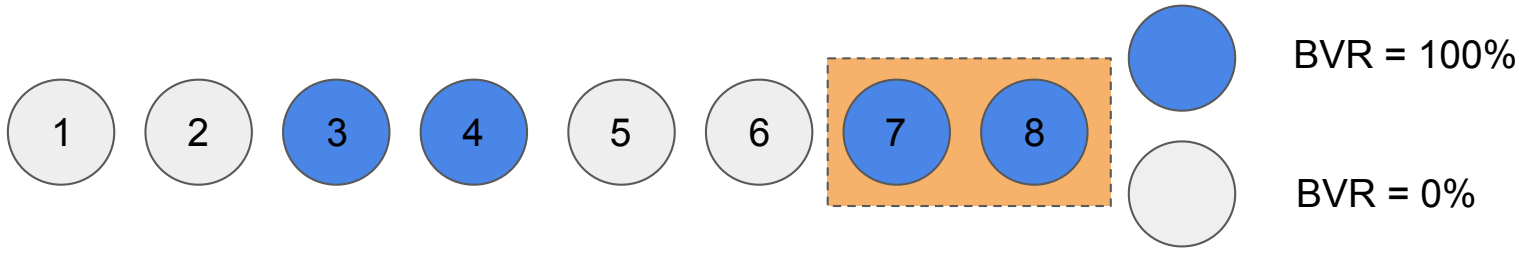
Window #	1	2	3	4	5		
#BVR 100% TBs	0	1	2	1	0		
#BVR 0% TBs	2	1	0	1	2		
Window Entropy	0	1	0	1	0		

Window-based Entropy #6



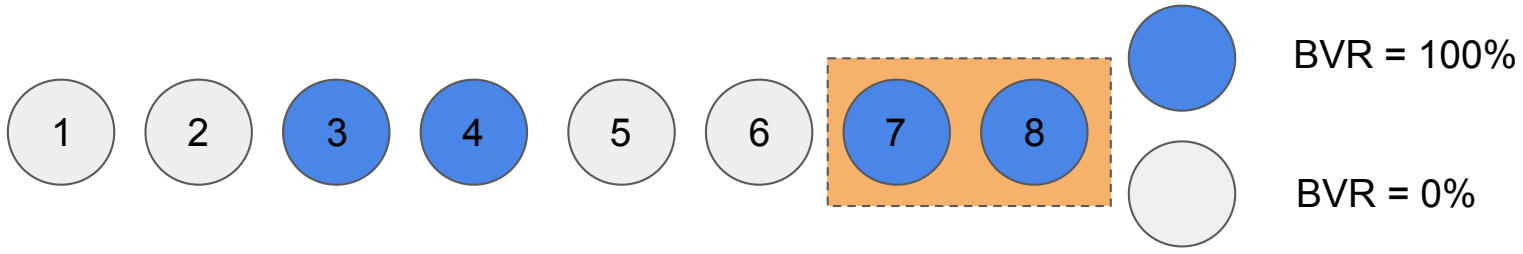
Window #	1	2	3	4	5	6	
#BVR 100% TBs	0	1	2	1	0	1	
#BVR 0% TBs	2	1	0	1	2	1	
Window Entropy	0	1	0	1	0	1	

Window-based Entropy #7



Window #	1	2	3	4	5	6	7
#BVR 100% TBs	0	1	2	1	0	1	2
#BVR 0% TBs	2	1	0	1	2	1	0
Window Entropy	0	1	0	1	0	1	0

Window-based Entropy #7

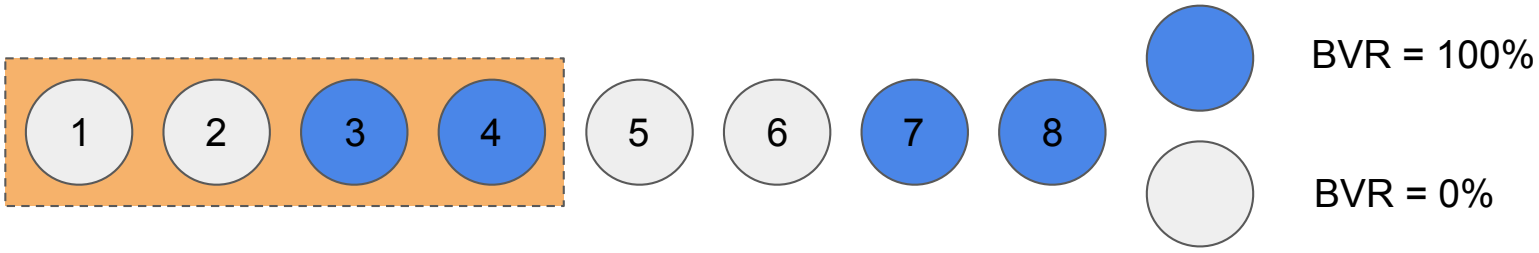


Window #	1	2	3	4	5	6	7
#BVR 100% TBs	0	1	2	1	0	1	2
#BVR 0% TBs	2	1	0	1	2	1	0
Window Entropy	0	1	0	1	0	1	0

Window Size = 2

H* = 3/7 = 0.43

Window-based Entropy (Size = 4)



Window #	1	2	3	4	5
#BVR 100% TBs	2	2	2	2	2
#BVR 0% TBs	2	2	2	2	2
Window Entropy	1	1	1	1	1

Window Size = 4

H* = 5/5 = 1

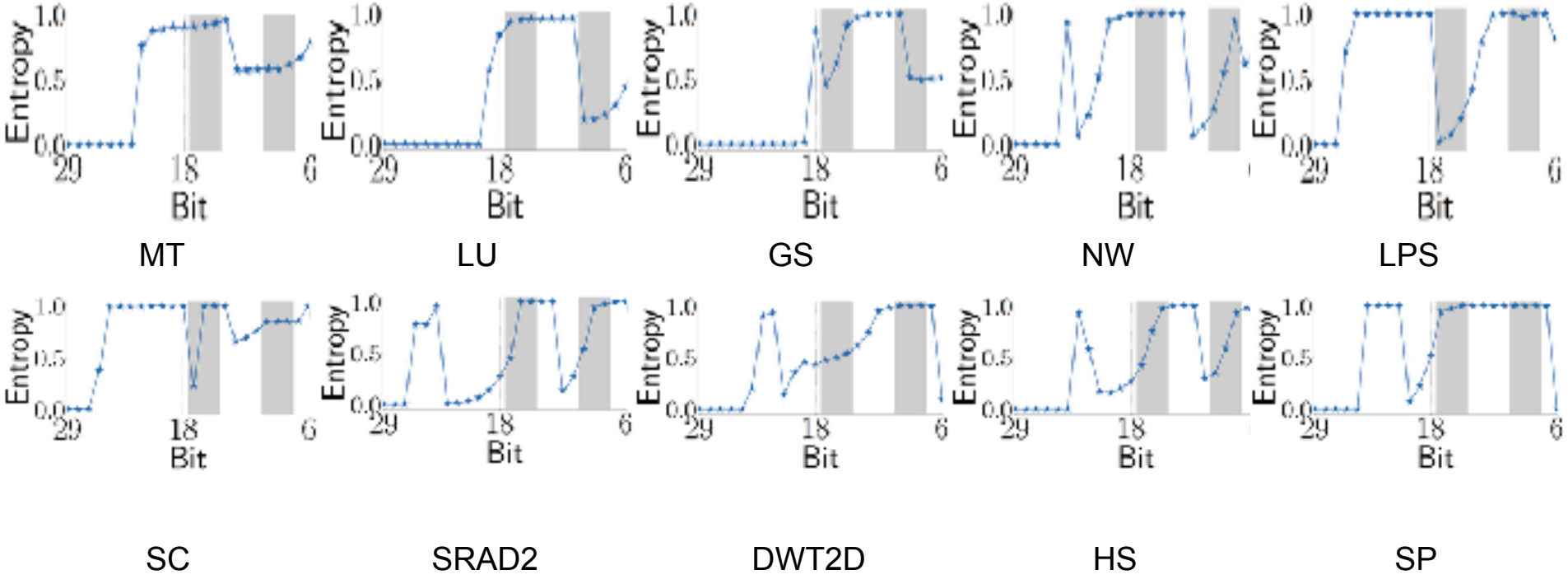
How to Choose Window Size?

- Window size is affected by:
  - Application, i.e, each TB's hardware requirement
  - GPU architecture, i.e, #SMs, warp scheduling policy

- *Maximum* window size is GPU hardware capacity
  - #SMs
  - Hardware resources / SM

- *Warp scheduling policy affects concurrent running TBs*
  - GTO policy: #SMs * 1
  - LRR policy: #SMs * (TBs/SM)

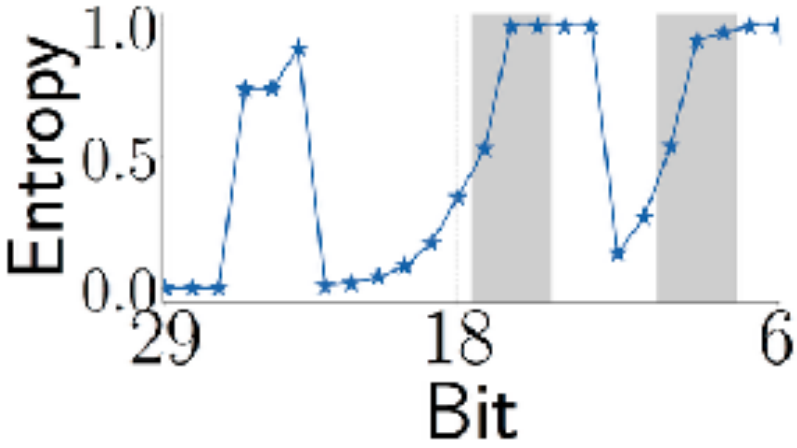
Entropy Valley Workloads



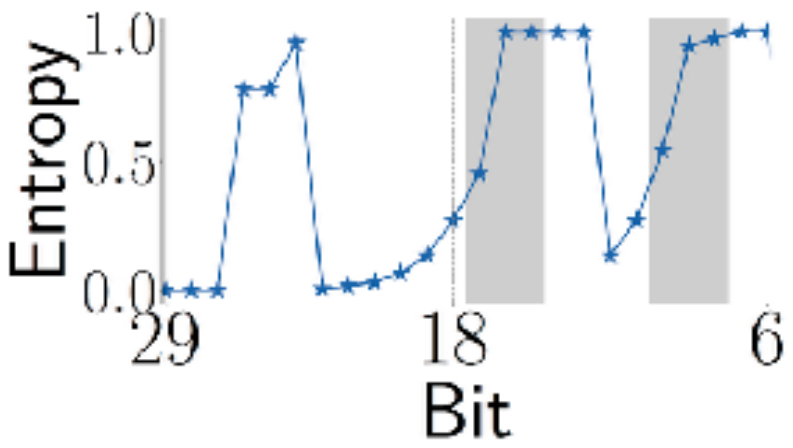
- Channel & Bank selection

Entropy Valleys of Applications and Kernels

- **Similar** behavior

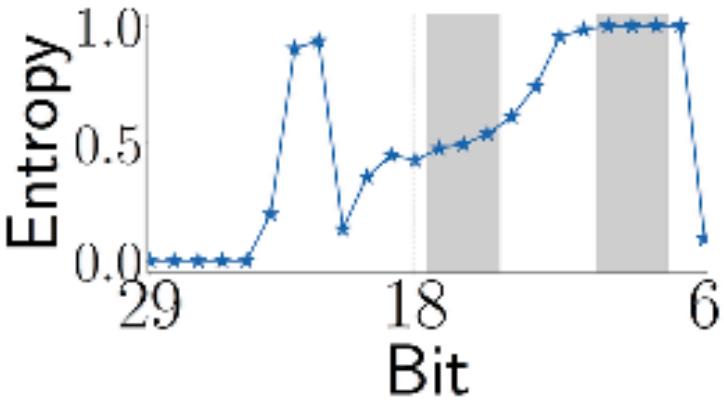


SRAD2

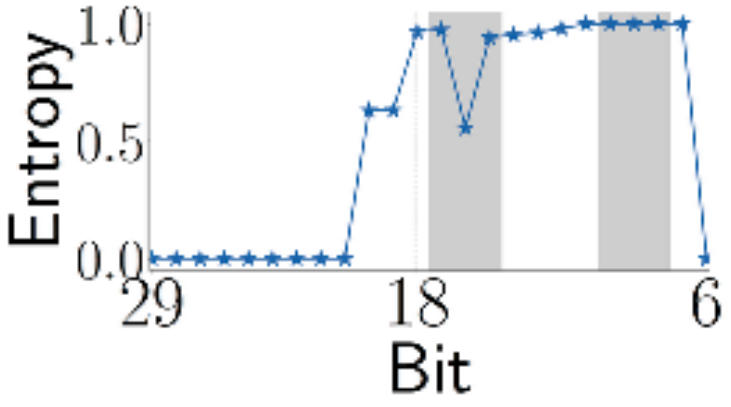


Kernel-1 of SRAD2

- **Different** behavior



DWT2D



Kernel-1 of DWT2D

Address Mapping

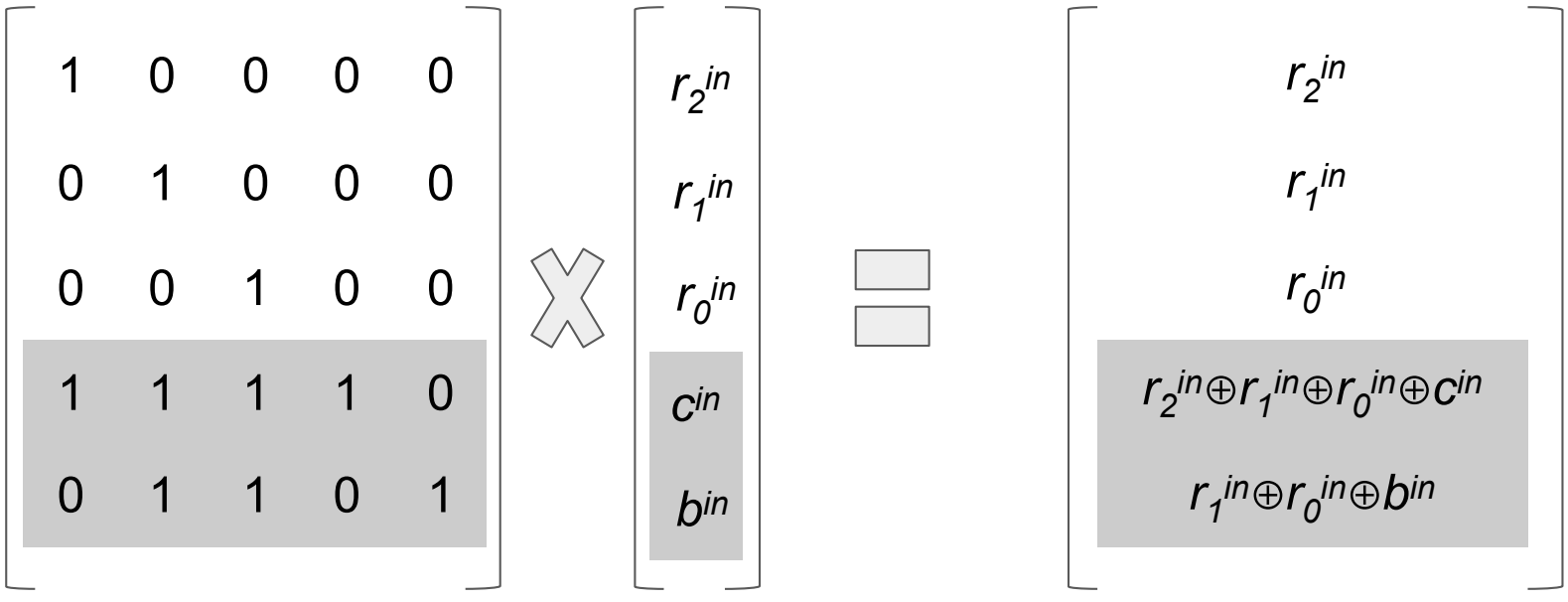
- *Entropy valley* in the channel or bank bits limit parallelism

- High GPU's memory bandwidth depends on *Memory-level parallelism (MLP)*
  - Channel-level Parallelism (CLP)
  - Bank-level Parallelism (BLP)

Get Out of the Valley:
Power-Efficient *Address Mapping* for GPUs

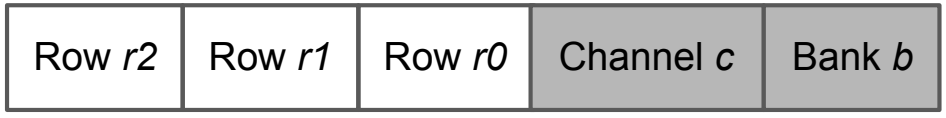
BIM Abstraction

- Binary Invertible Matrix (BIM) is a generic abstraction for representing all address mapping schemes
  - Matrix-vector product
  - Invertible criterion: 1-to-1 mapping
  - Multiplication maps to bit-wise AND-operation
  - Addition maps to XOR-operation



Remap Address Mapping (RMP)

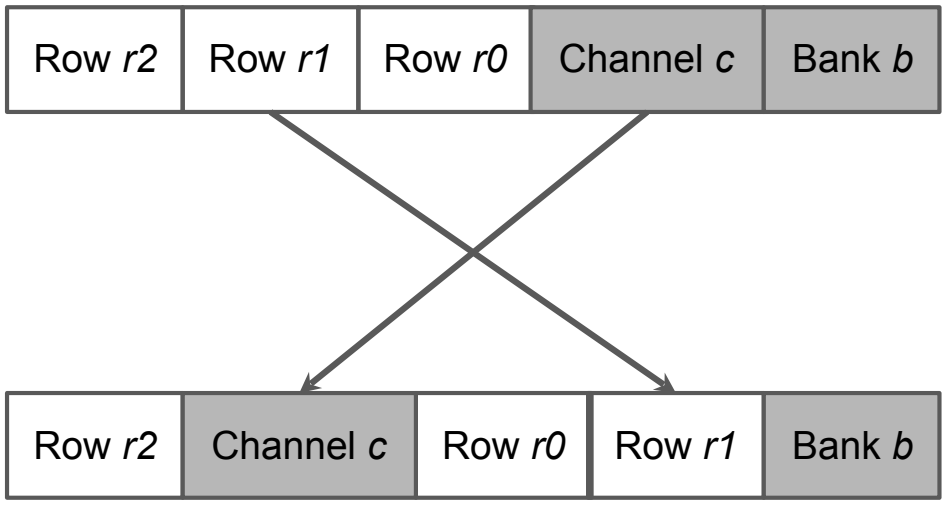
- RMP scheme: remap high and low bits

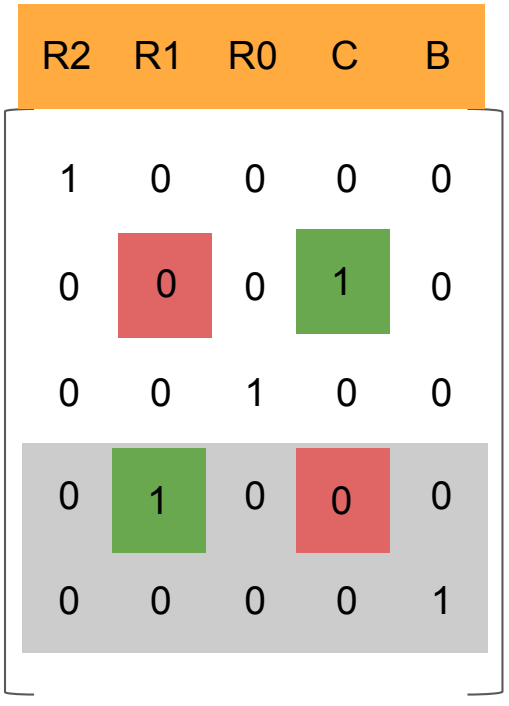


R2	R1	R0	C	B
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Remap Address Mapping (RMP)

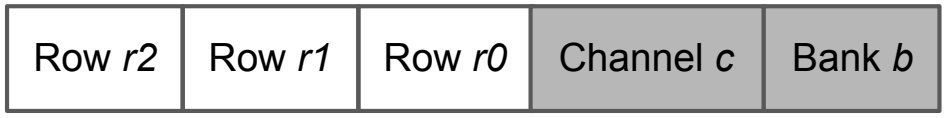
- RMP scheme: remap high and low bits





Permutation-Based Address Mapping (PM)

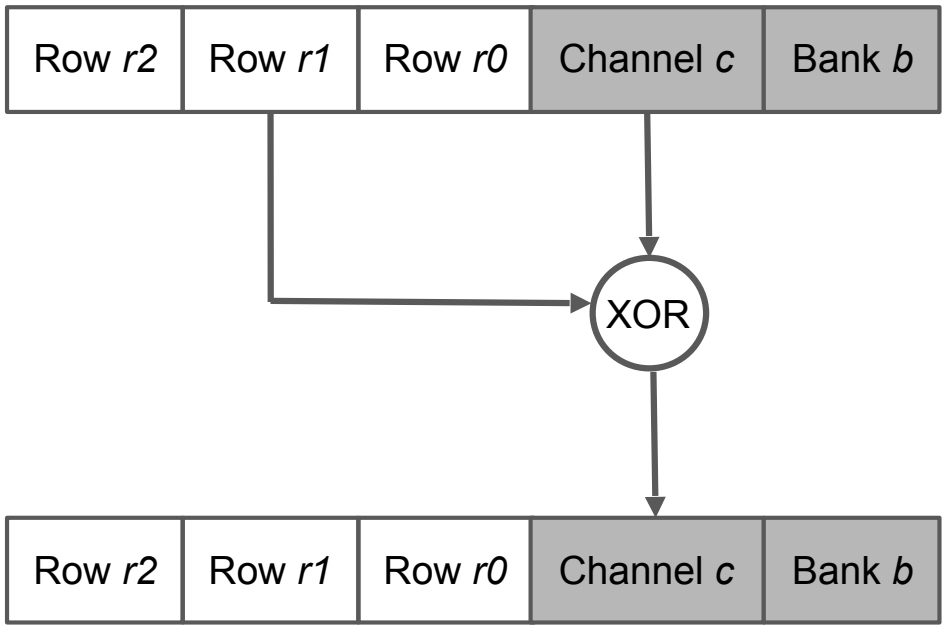
- PM scheme: XOR channel or bank bits with a row bits



R2	R1	R0	C	B
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Permutation-Based Address Mapping (PM)

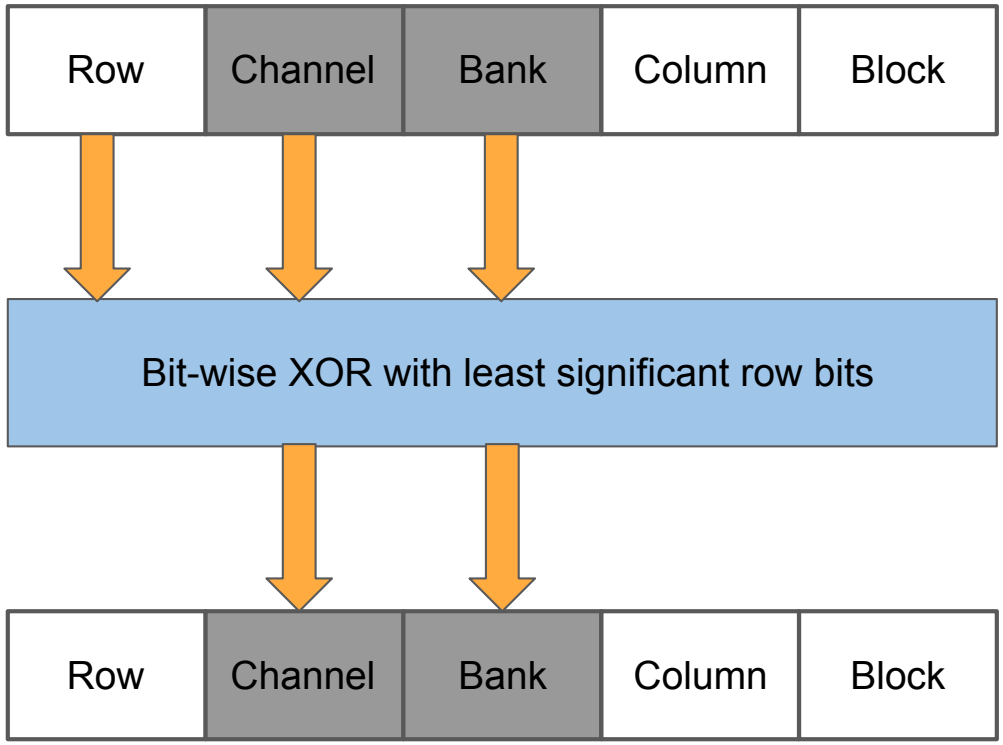
- PM scheme: XOR channel or bank bits with a row bits



R2	R1	R0	C	B
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	1	0	1	0
0	0	0	0	1

PM Address mapping

- Concentrate *row* bits' entropy into channel and bank bits



Drawback:

1. *Low entropy of row bits*
2. *Application-dependent*

Broad Address Mapping

- Harvest entropy across a broad selection of address bits

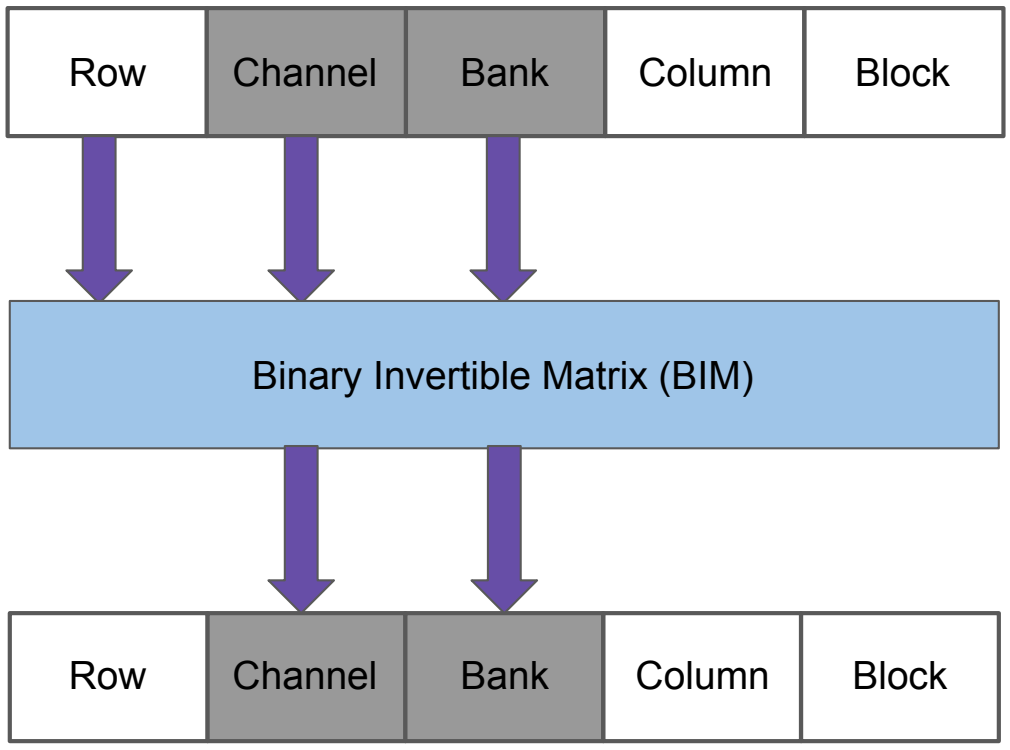
1. Page Address Entropy (**PAE**): gathers entropy from *page address bits* to generate new bank and channel bits

1. Full Address Entropy (**FAE**): gathers entropy from full address bits to generate new bank and channel bits

1. All Address Mapping (**ALL**): gathers entropy from full address bits to generate full bits

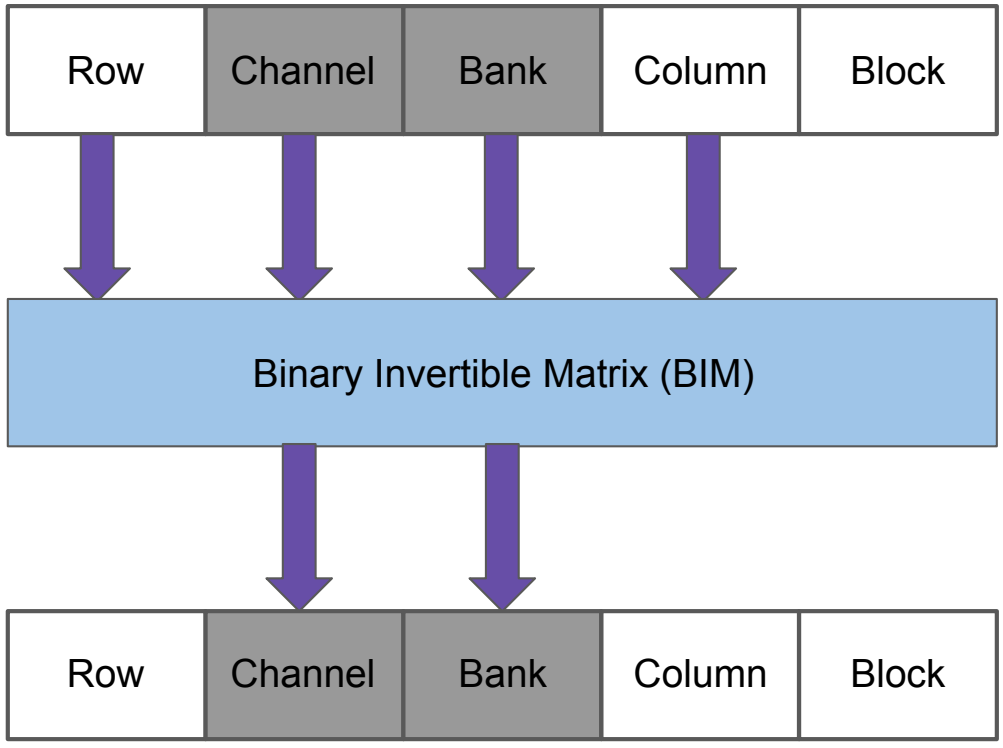
PAE Address Mapping

- Input: *page* address bits, i.e., channel, bank, row
- Output: channel and bank bits



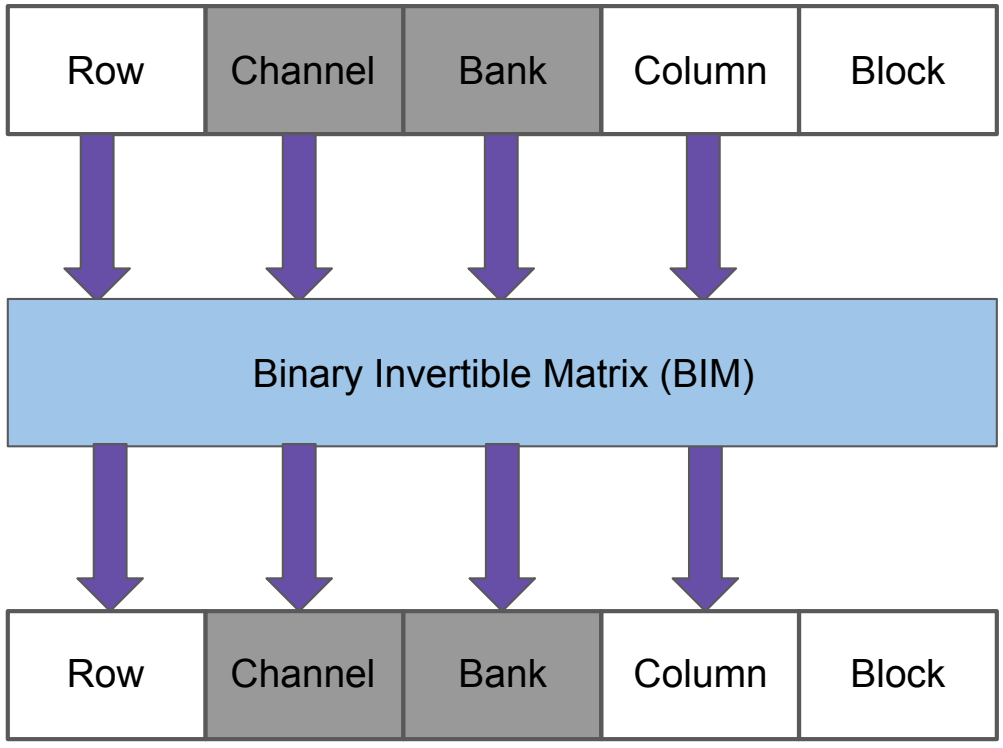
FAE Address Mapping

- Input: *full* address bits
- Output: channel and bank bits



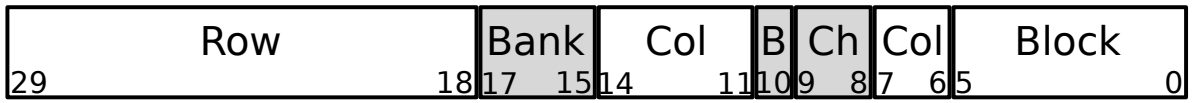
ALL Address Mapping

- Input: *full* address bits
- Output: *full* address bits



Experimental Methodology

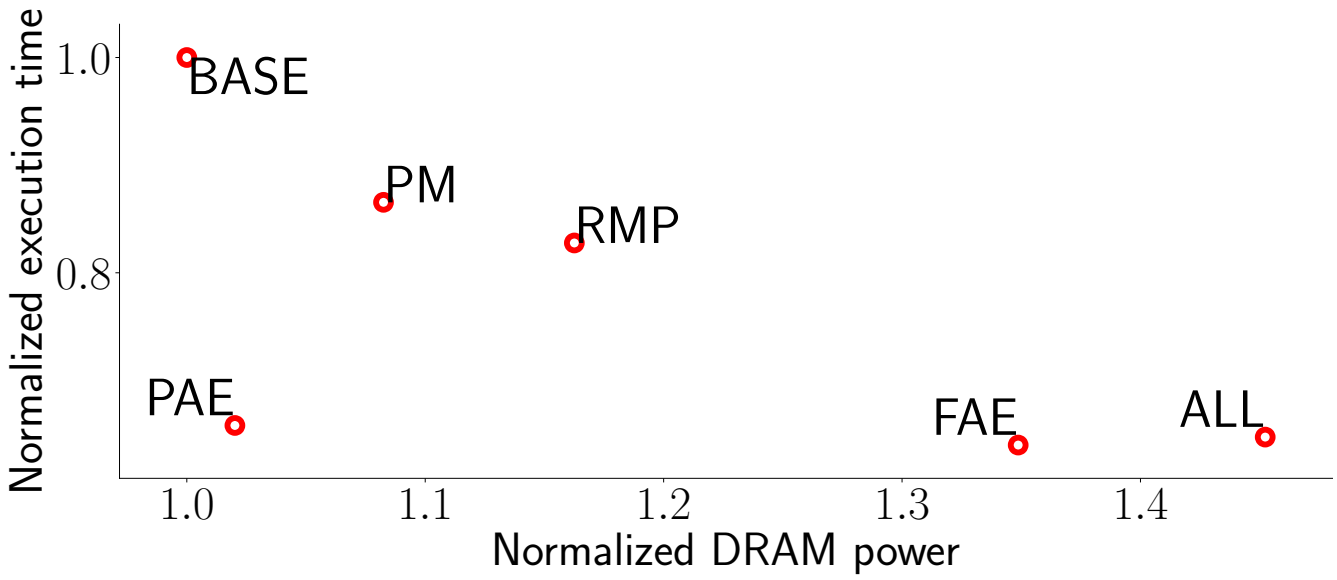
- GPU Architecture:
  - 12 SMs running at 1.4Ghz
  - Max TBs per SM is 8, Max warps per SM is 48
- 1GB Hynix' GDDR5, similar with MICRO11[1]



- Address mapping schemes:
  - BASE scheme
  - RMP scheme
  - PM scheme
  - PAE, FAE, ALL schemes

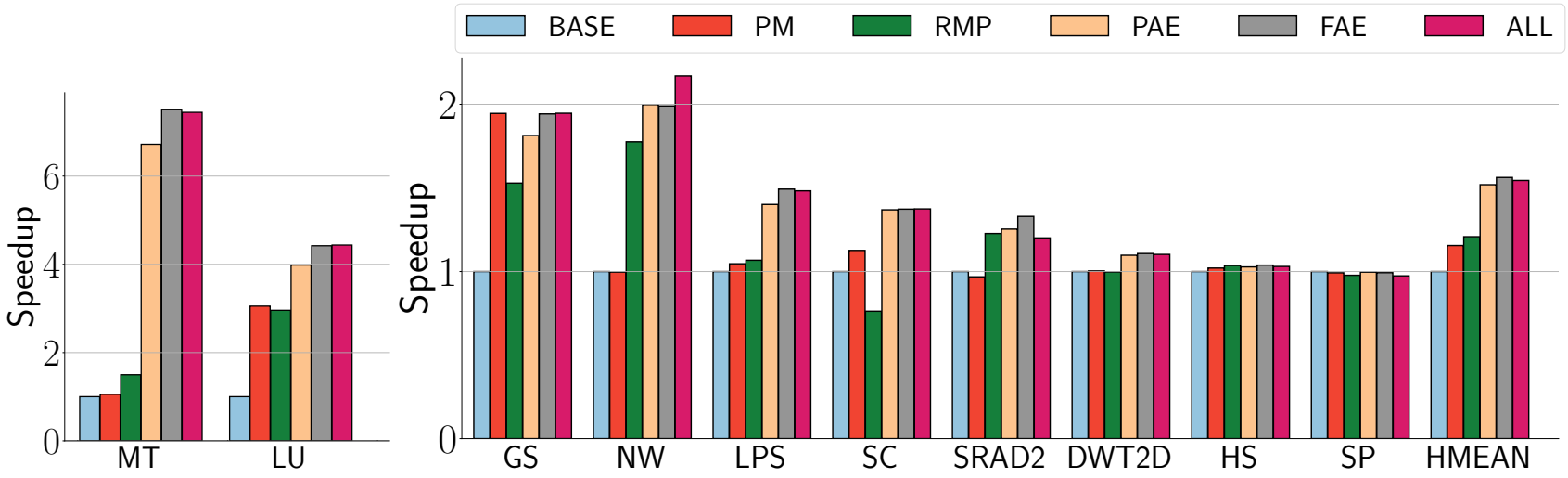
[1] Minimalist Open-page: A DRAM Page-mode Scheduling Policy for the Many-core Era

Performance vs. DRAM power consumption



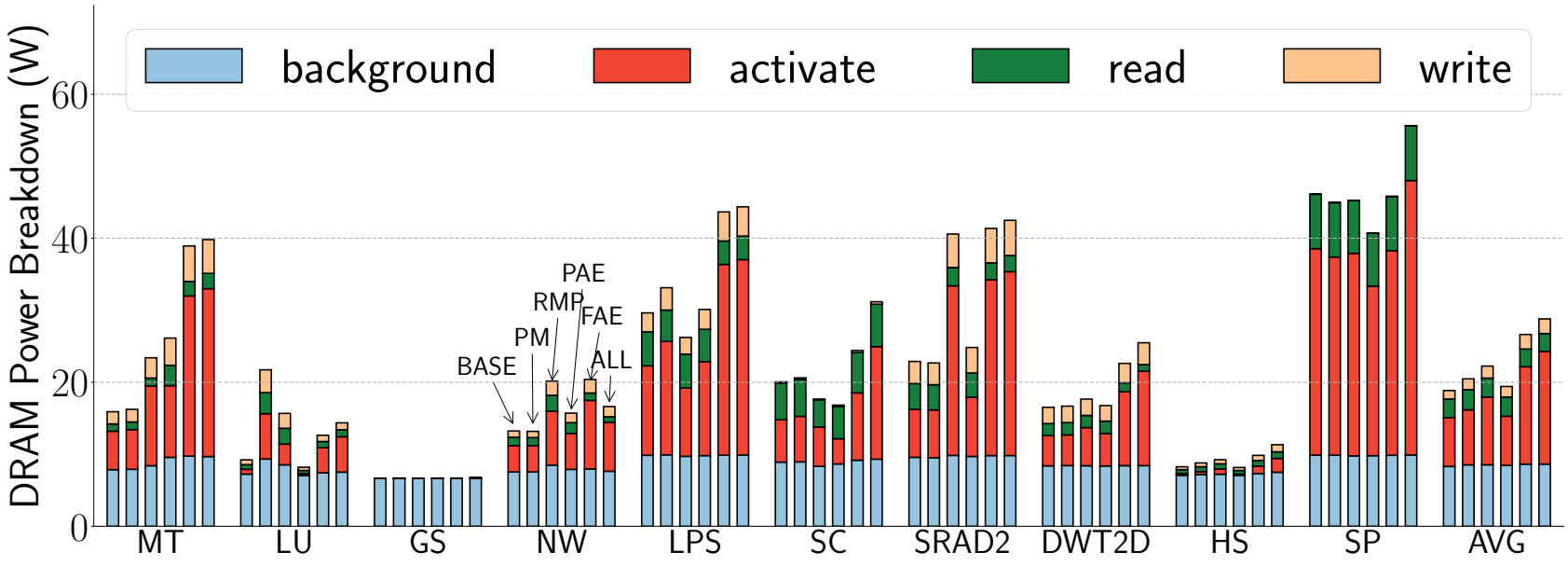
- PAE is *most power-efficient*, achieving an average *1.52x* speedup while consuming *3%* more DRAM power
- FAE and ALL are *slightly performance-wise* (*1.56x* and *1.54x* speedup) but consume *35%* and *45%* more DRAM power, respectively

Performance Improvement



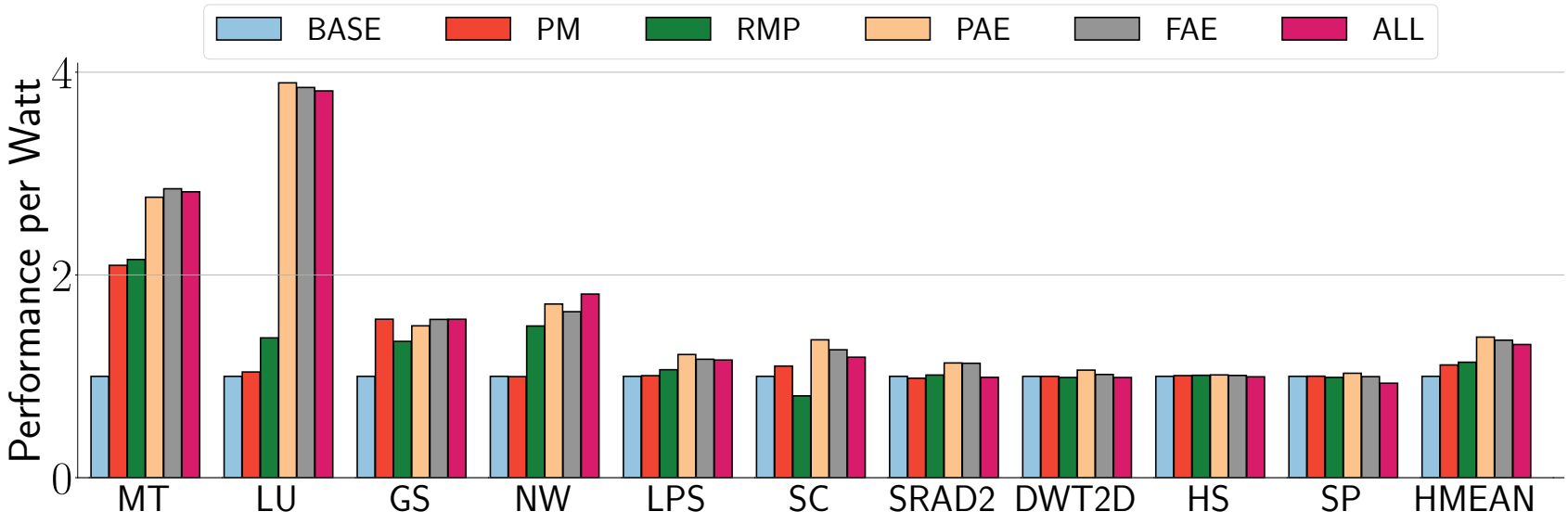
- PAE, FAE and ALL lead to dramatic speedups averaging to *1.52x*, *1.56x* and *1.54x*, respectively

DRAM Power Consumption



- Address mapping primarily affects the *activate power*
- PAE has a small increase in DRAM power consumption by *3%* on average
- FAE and ALL lead to a substantial increase in DRAM power consumption, by *35%* and *45%* on average

Total System (GPU+DRAM) Power Consumption

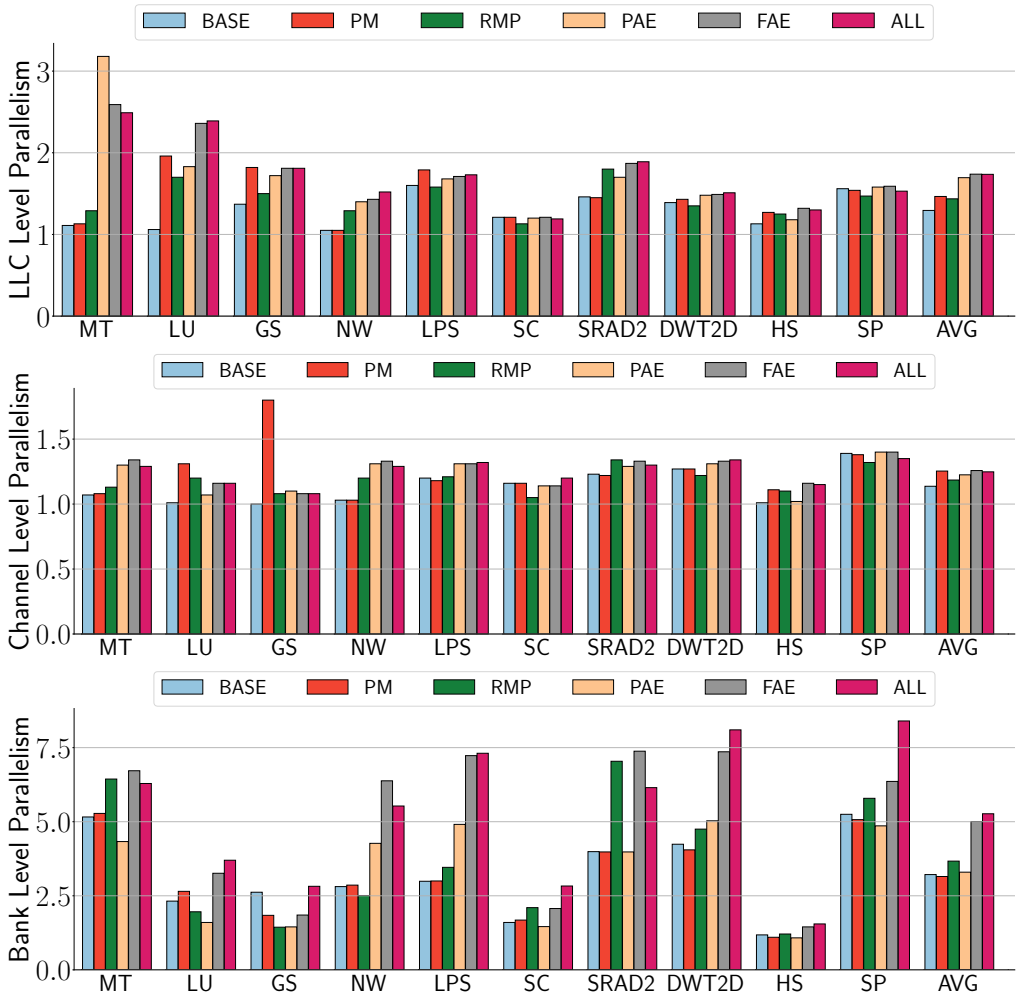


- PAE, FAE and ALL increase system power consumption increases by *9%*, *15%* and *18%* on average
- PAE, FAE and ALL improve *performance per Watt* by *1.39x*, *1.36x* and *1.31x* on average respectively

NoC Packet Latency and LLC Miss Rate

- *Serialized memory access stream* leads to a dramatically high NoC packet latency and LLC miss rate
- PAE, FAE and ALL *distribute the accesses uniformly* and ultimately lead to a dramatic reduction

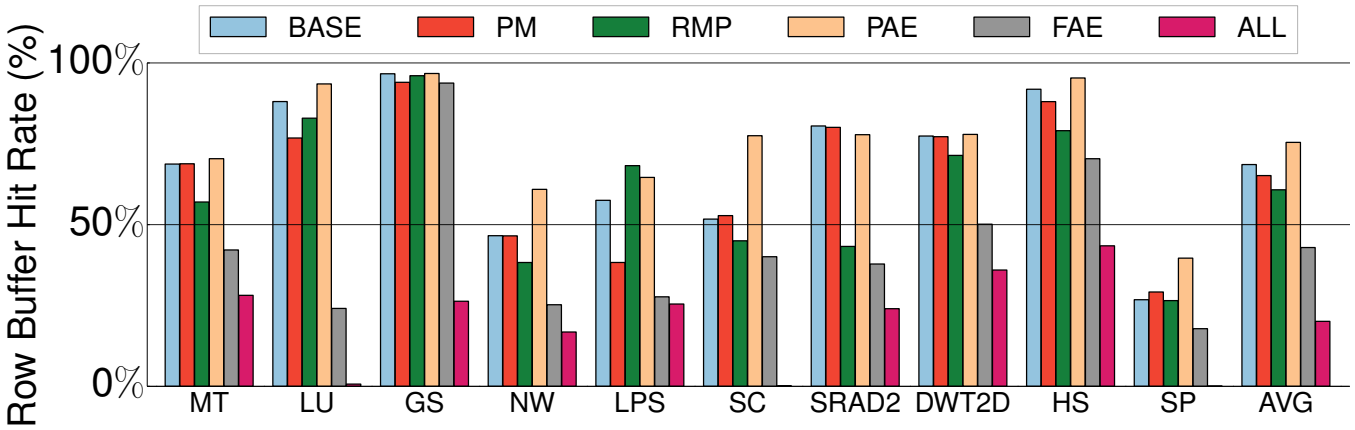
Memory Level Parallelism



- PAE, FAE and ALL improve *memory level parallelism* including LLC level, channel level and bank level parallelism

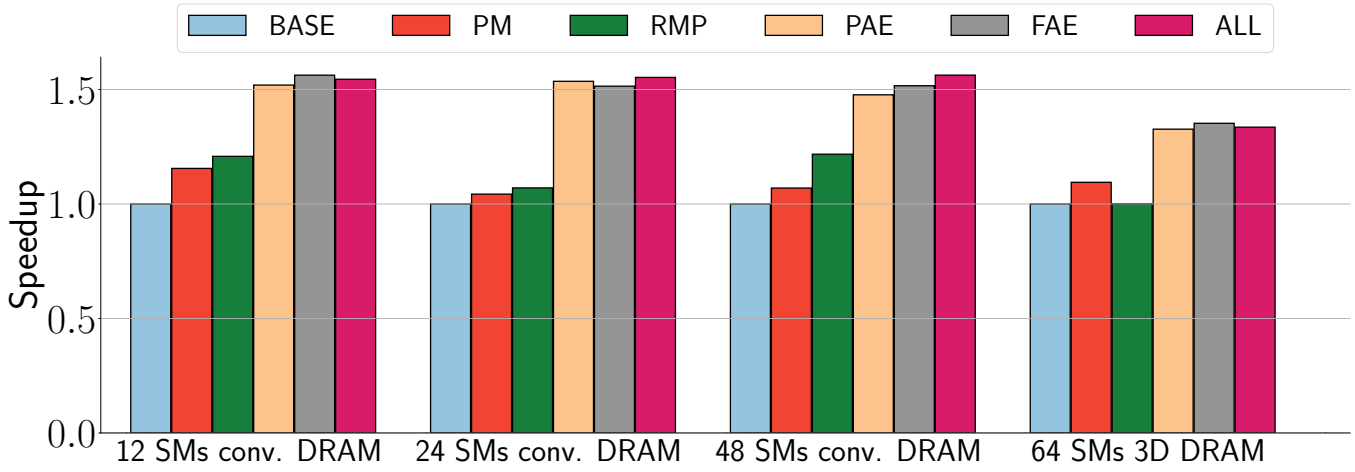
Row Buffer Hit Rates

- *Tradeoff* between bank level parallelism and row buffer locality



- PAE creates *sufficiently good load balancing* while keeping good-locality requests within the same bank
- FAE sometimes *reduces row buffer hit rates* as it distributes good-locality requests to different banks

Sensitivity



- PAE, FAE and ALL *consistently improve performance* across SM counts (from 12 to 48)
- 3D stacked memory system with 64 SMs:
  - PAE, FAE and ALL still achieve high performance
  - RMP *performs similarly to BASE* since it cannot have enough high entropy bits to achieve good load balancing

Conclusions

- Provided a *window-based entropy analysis* tailored for the highly concurrent memory request behavior in GPU-compute workloads
- Observed that GPU-compute workloads exhibit *entropy valleys* distributed throughout the lower order address bits
- Developed *Page Address Entropy* (*PAE*) mapping scheme which provide significantly higher performance and power-efficiency than previously proposed address mapping schemes

Thank you!

Questions?