Think Different (4): Processing-In-Memory

Hung-Wei Tseng

Limitations of in-storage processing

- Programming
- Processor capabilities
  - Only gives you “moderate” performance gain, not orders-of-magnitude

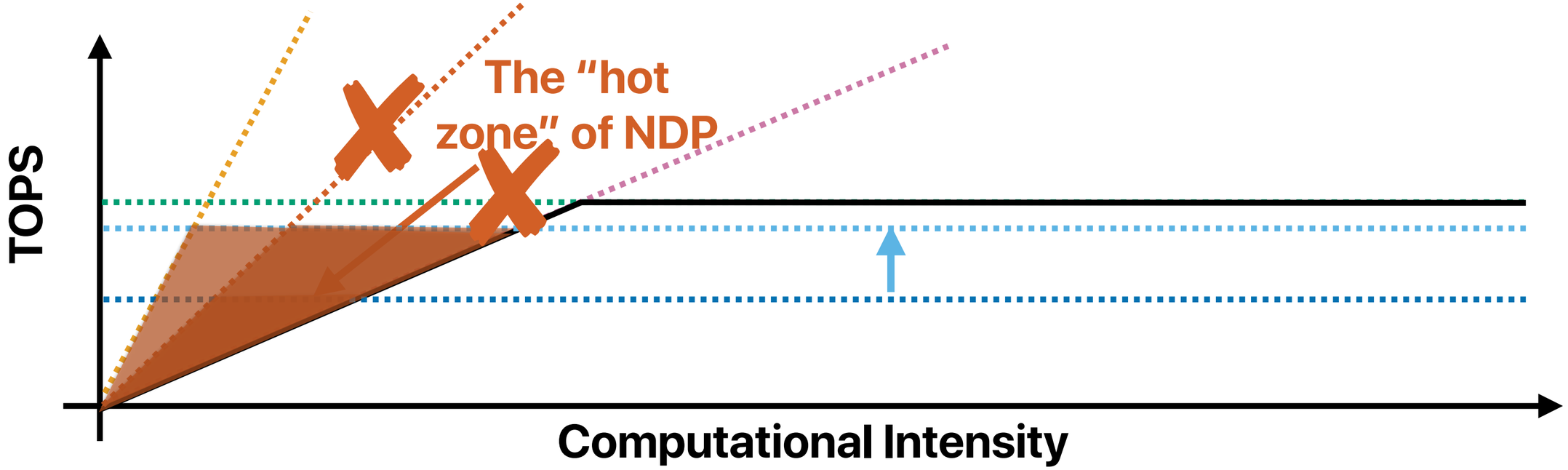
Reviewing the roofline model

~~Peak OPS of target computing resource~~

~~Peak memory bandwidth × computational intensity ÷ reduction of data volume~~

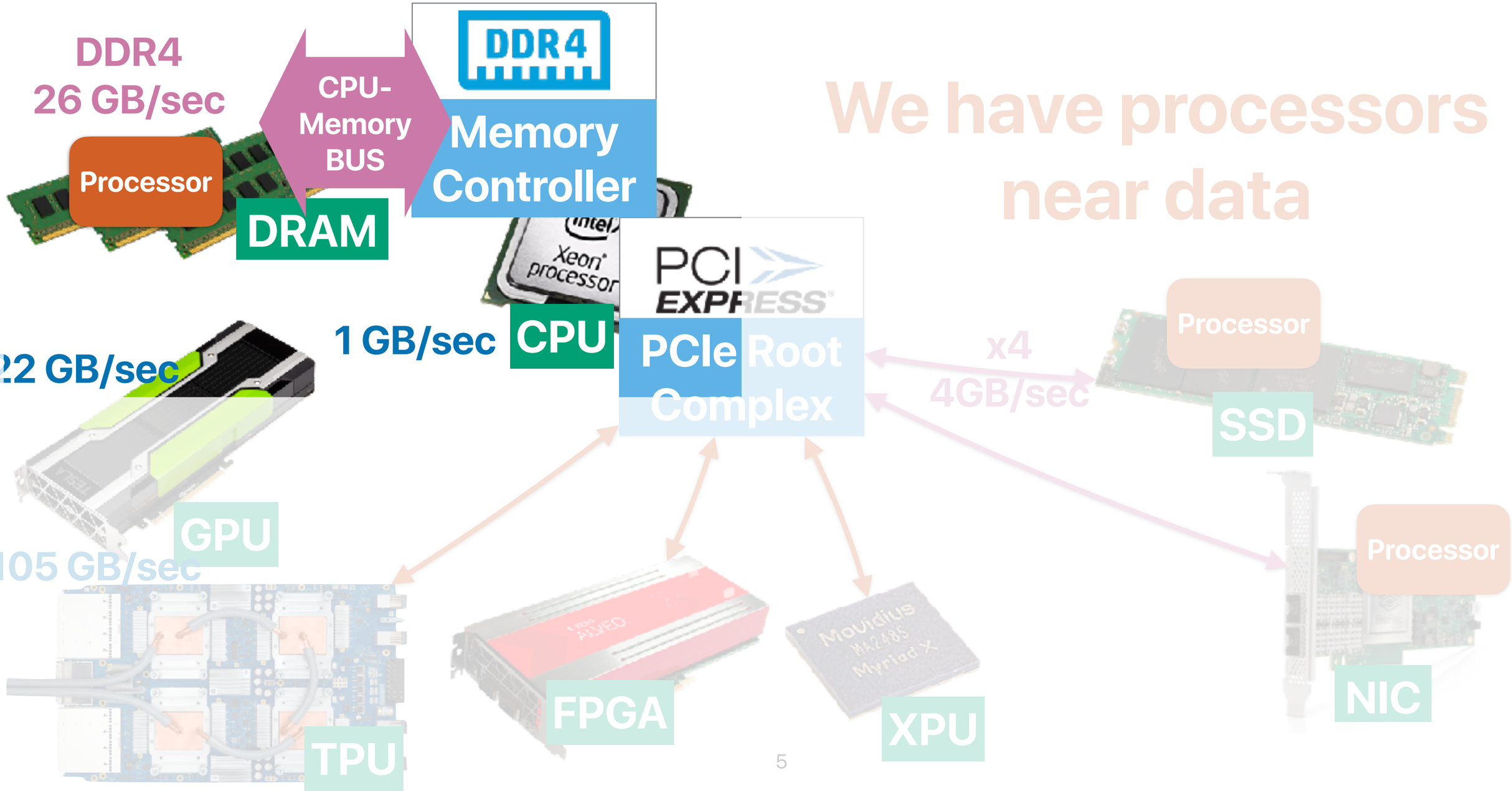
Peak OPS of target NDP device

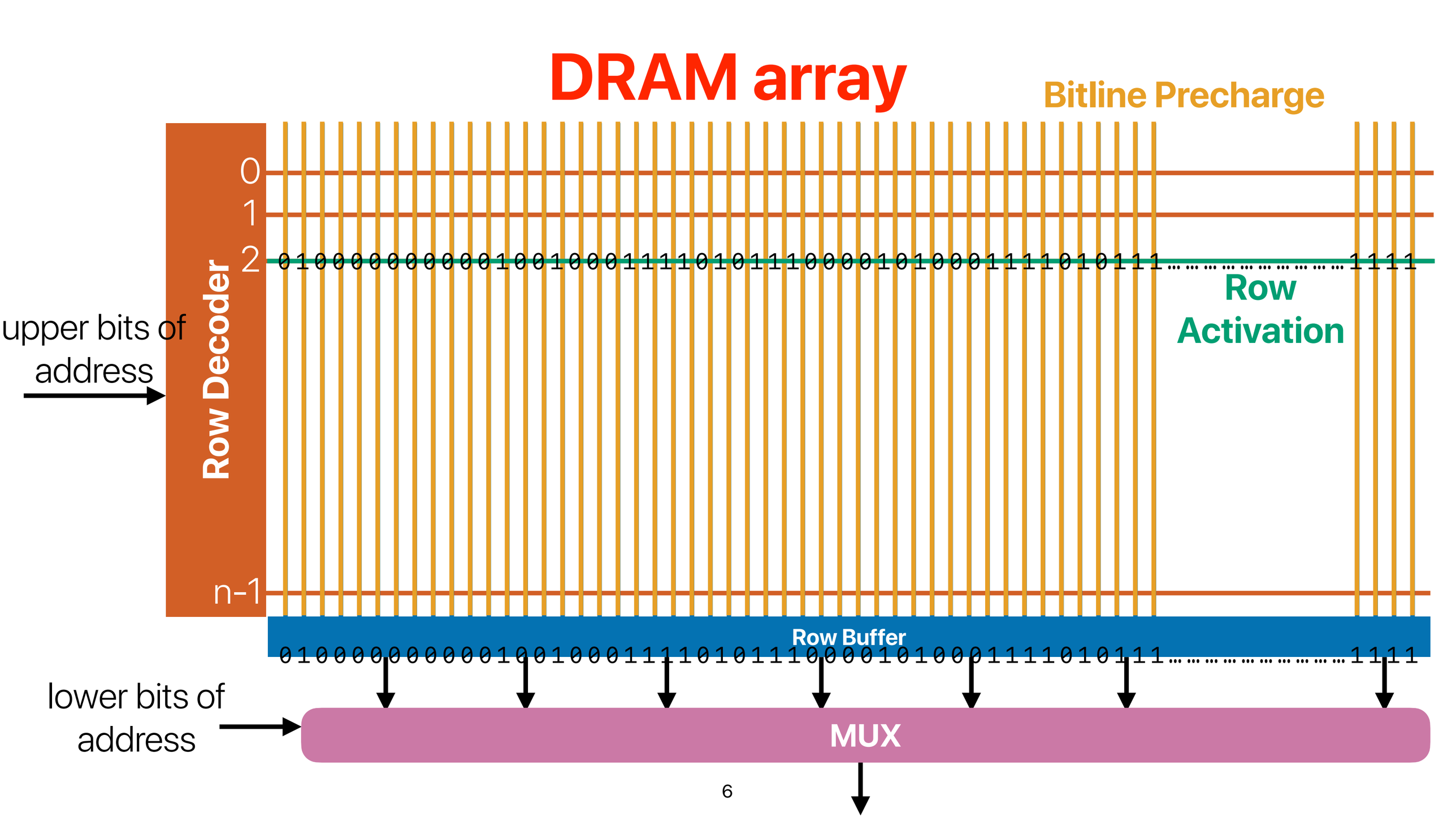
Peak device internal bandwidth × computational intensity of NDP program



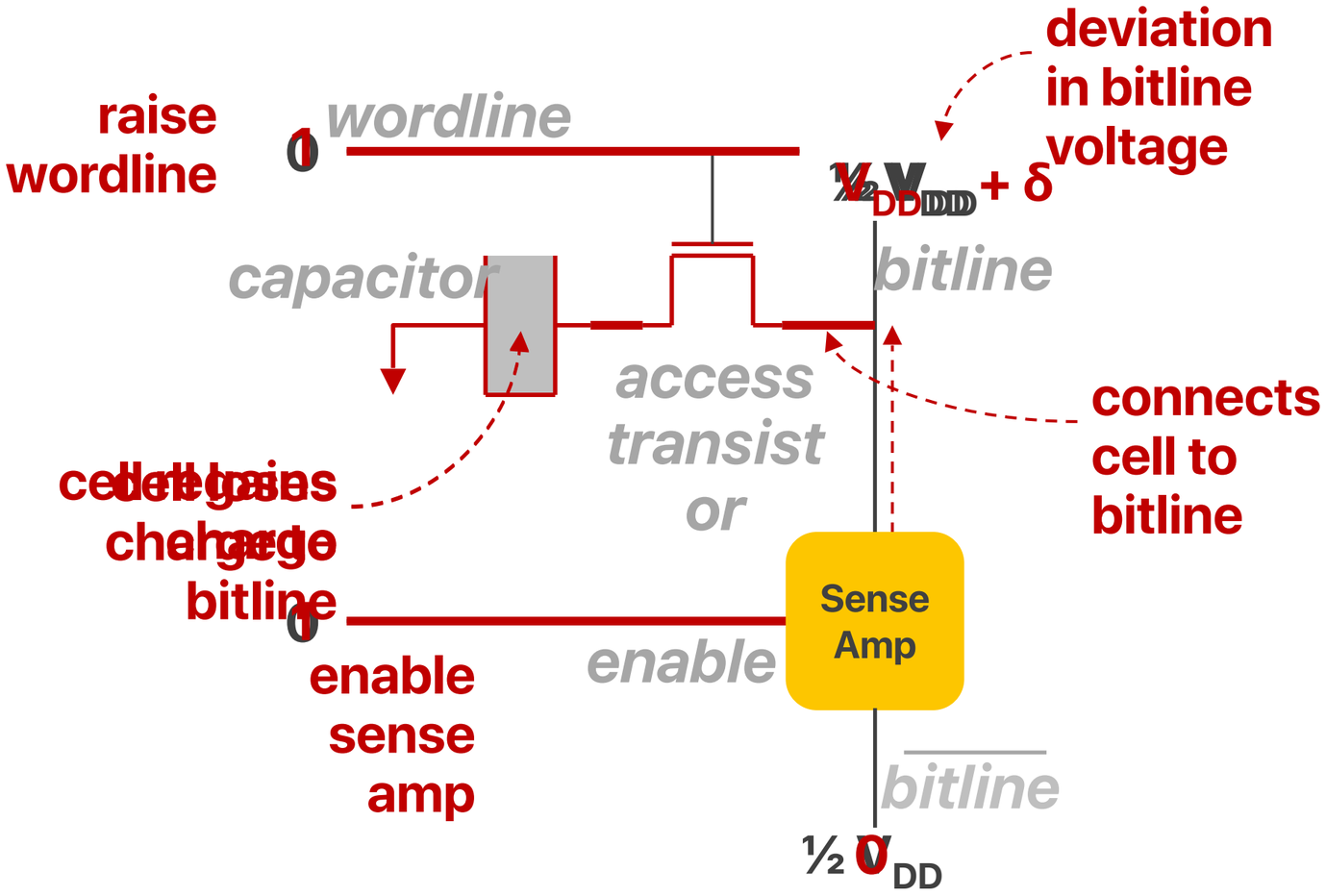
Level up — think different at the main memory level

Think different



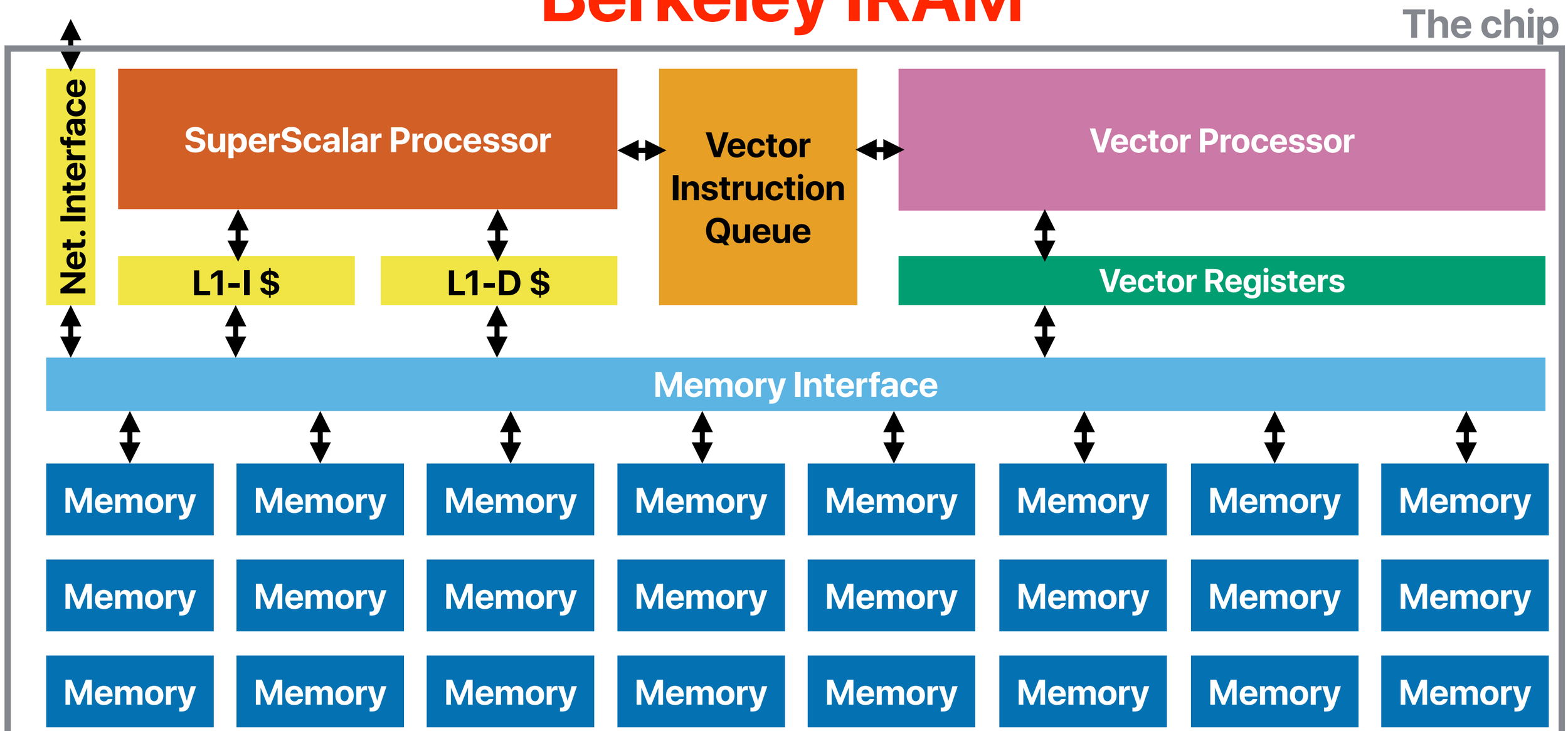

DRAM array
Bitline Precharge
Row Decoder
0
1
2
n-1
Row
Activation
upper bits of
address
Row Buffer
lower bits of
address
MUX

Basic DRAM Cell Operation



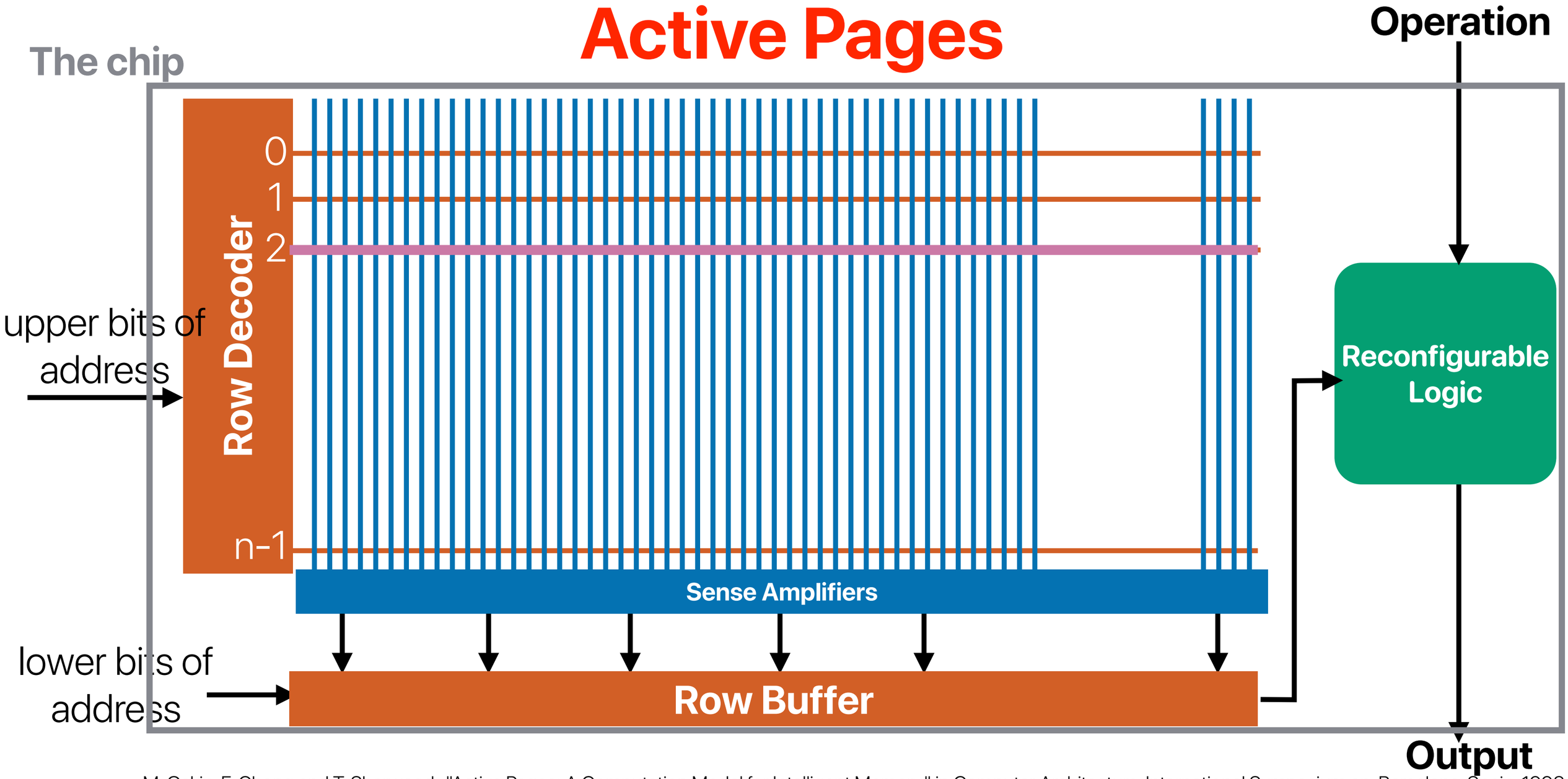
Near-memory processing/ Processor-In-Memory (PIM)

Berkeley IRAM



David Patterson, Thomas Anderson, Neal Cardwell, Richard Fromm, Kimberly Keeton, Christoforos Kozyrakis, Randi Thomas, and Katherine Yelick. 1997. A Case for Intelligent RAM. IEEE Micro 17, 2 (March 1997), 34–44.

Active Pages



M. Oskin, F. Chong and T. Sherwood, "Active Pages: A Computation Model for Intelligent Memory," in Computer Architecture, International Symposium on, Barcelona, Spain, 1998

3:00

What kinds of and what applications would fit well in IRAM or Active Pages?

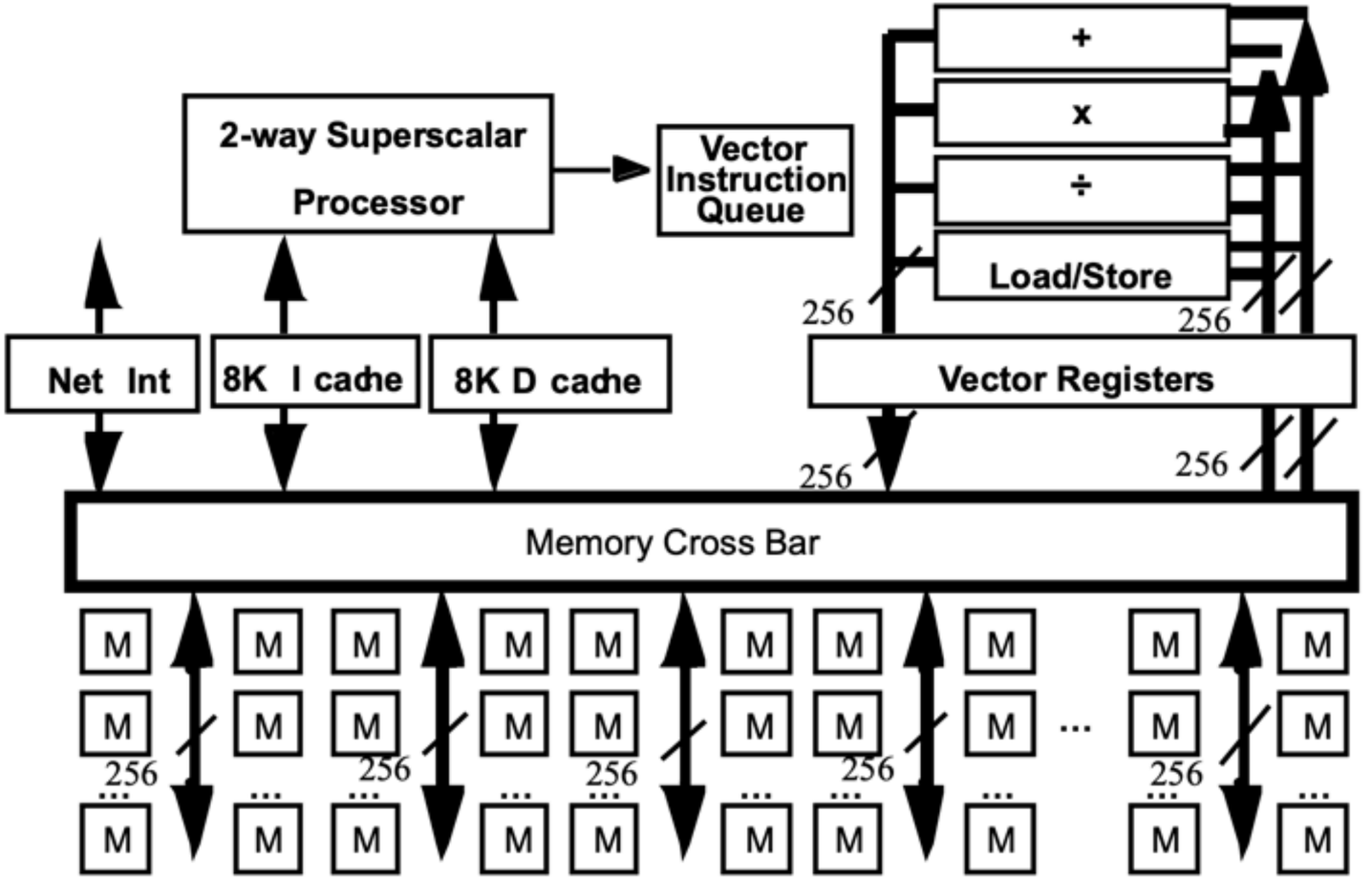
What kind of applications?

- Both Active Pages and IRAM operates on a "page" or "pages" of data simultaneously
- Throughput oriented rather than latency sensitive
- Cannot be cached well
  - DRAM-intensive
  - Low locality (hard-to-predict patterns)
  - Large data footprint

Table 2: CPI, cache misses, and time spent in Alpha 21164 for four programs

Category	SPECint92	SPECfp92	Database	Sparse
Clocks Per Instruction (CPI)	1.2	1.2	3.6	3.0
I cache misses per 1000 instructions	7	2	97	0
D cache misses per 1000 instructions	25	47	82	38
L2 cache misses per 1000 instructions	11	12	119	36
L3 cache misses per 1000 instructions	0	0	13	23
Fraction of time in processor	0.78	0.68	0.23	0.27
Fraction of time in I cache misses	0.03	0.01	0.16	0.00
Fraction of time in D cache misses	0.13	0.23	0.14	0.08
Fraction of time in L2 cache misses	0.05	0.06	0.20	0.07
Fraction of time in L3 cache misses	0.00	0.02	0.27	0.58

0.25 µm, Fast Logic IRAM: ≈500MHz
5 GFLOPS(64b)/40 GOPS(8b)/24MB



19

benchmark	Small			Large		
	Conven-tional	IRAM		Conven-tional	IRAM	
		(.75 **X**)	(1.0 **X**)		(.75 **X**)	(1.0 **X**)
hsfsys	138	112 *(0.81)*	150 *(1.08)*	149	114 *(0.77)*	152 *(1.02)*
noway	111	99 *(0.89)*	132 *(1.19)*	127	104 *(0.82)*	139 *(1.09)*
nowsort	109	104 *(0.95)*	138 *(1.27)*	136	110 *(0.81)*	147 *(1.08)*
gs	119	107 *(0.90)*	142 *(1.20)*	141	109 *(0.78)*	146 *(1.04)*
ispell	145	113 *(0.78)*	151 *(1.04)*	149	115 *(0.77)*	153 *(1.03)*
compress	91	102 *(1.13)*	137 *(1.50)*	127	104 *(0.82)*	139 *(1.09)*
go	97	96 *(0.99)*	128 *(1.31)*	128	98 *(0.76)*	130 *(1.02)*
perl	136	106 *(0.78)*	141 *(1.04)*	140	107 *(0.76)*	142 *(1.01)*

Table 6: **Performance (in MIPS) of IRAM versus conventional processors, as a function of processor slowdown in a DRAM process.** Only the models with the 32:1 DRAM to SRAM-cache area density ratio are shown. The values in parentheses are the ratios of performances of the IRAM models compared to the CONVENTIONAL implementations. Ratios greater than 1.0 indicate that IRAM has higher performance.

Active Pages: Applications

Memory-Centric Applications			
Name	Application	Processor Computation	Active Page Computation
Array	C++ standard template library array class	C++ code using array class Cross-page moves	Array insert, delete, and find
Database	Address Database	Initiates queries Summarizes results	Searches unindexed data
Median	Median filter for images	Image I/O	Median of neighboring pixels
Dynamic Prog	Protein sequence matching	Backtracking	Compute MINs and fills table
Processor-Centric Applications			
Name	Application	Processor Computation	Active Page Computation
Matrix	Matrix multiply for Simplex and finite element	Floating point multiplies	Index comparison and gather/scatter of data
MPEG-MMX	MPEG decoder using MMX instructions	MMX dispatch Discrete cosine transform	MMX instructions

Table 2: Summary of partitioning of applications between processor and active pages

Active Pages

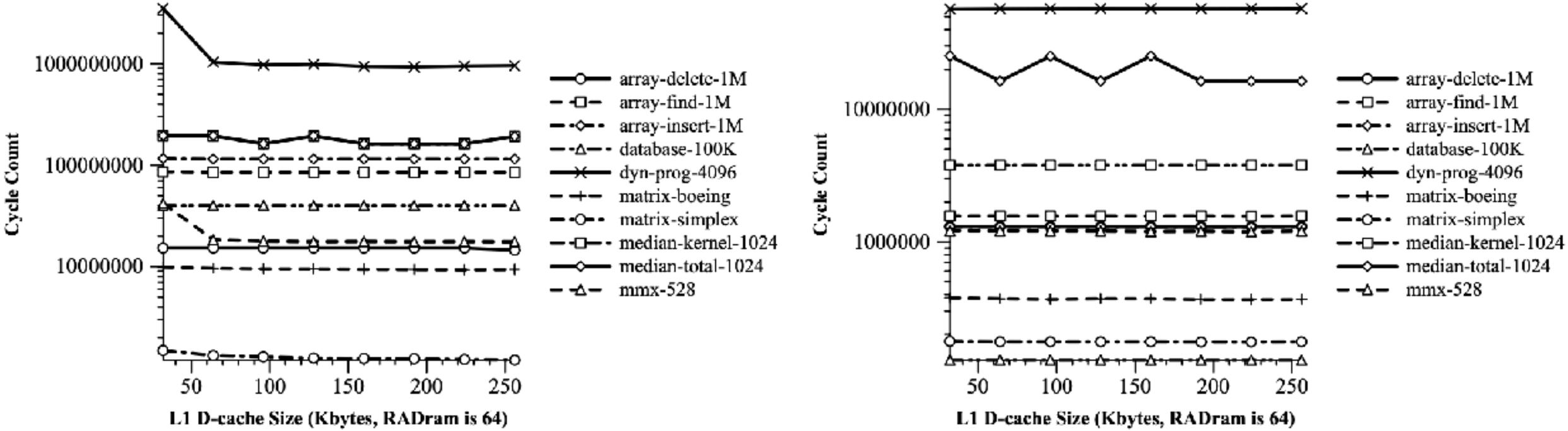


Figure 5: Conventional **(left)** and RADram **(right)** Execution Time vs. L1 Data Cache Size

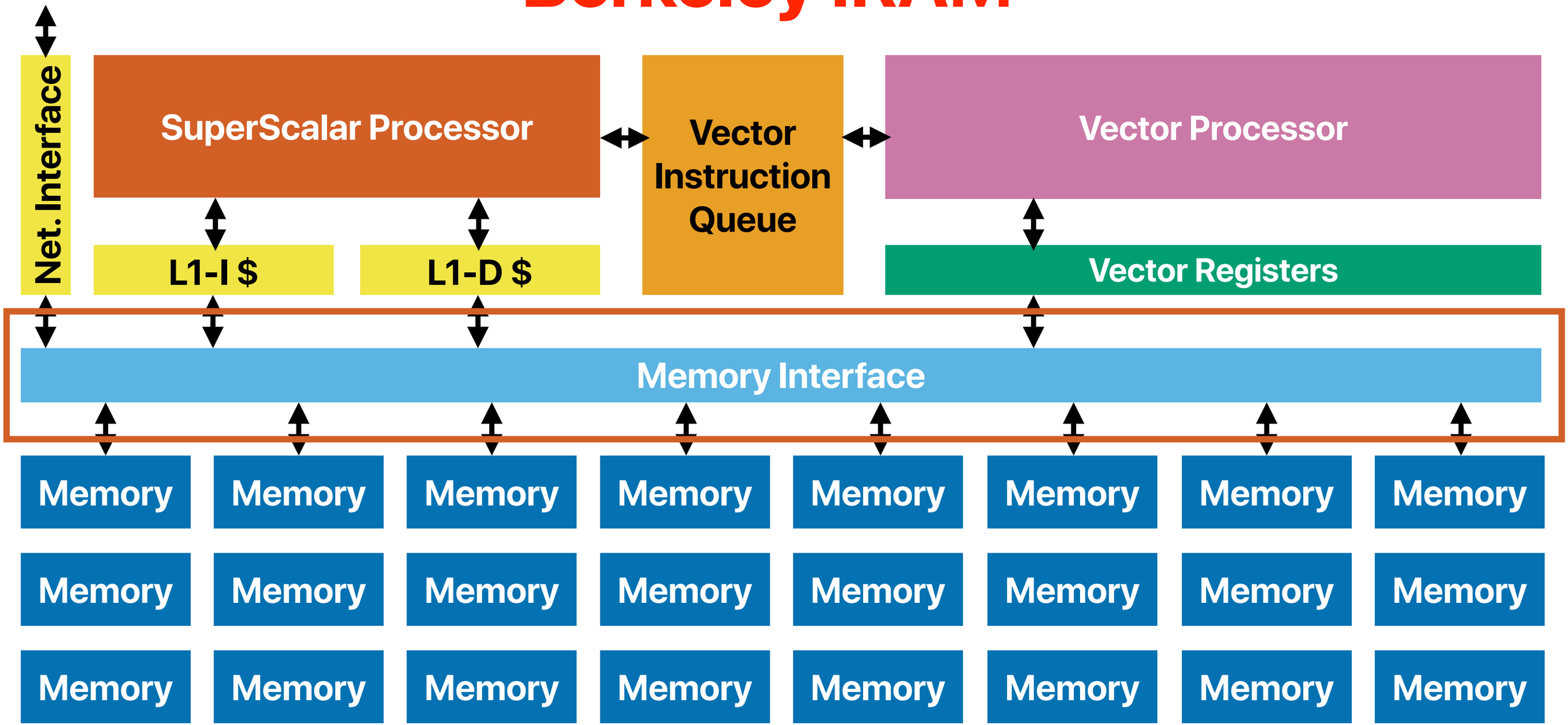
3:00

Why “active pages” and “iRAM” do not work out during that era?

Why IRAM or Active Pages does not work out?

- Programming
  - Why GPU succeed later?
- Applications
- Lack of industrial implementation, support
- Hardware design — Processor v.s. DRAM process technologies
- Cost
  - For Active Pages — small processor each module
  - For IRAM
    - High bandwidth on-chip interconnect was expensive
    - Other hardware design issues
      - Significant differences in clock rates among units

Berkeley IRAM



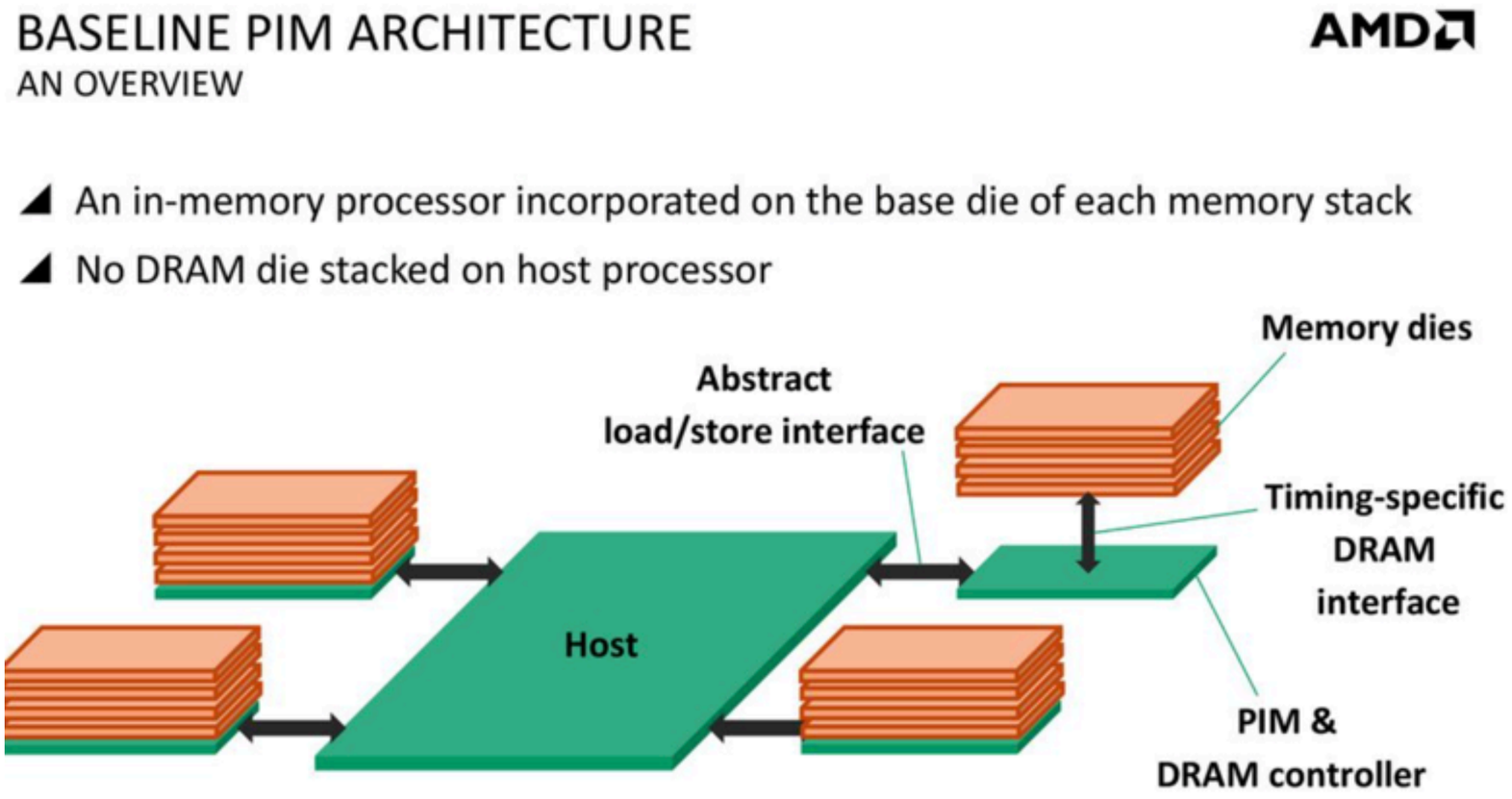
David Patterson, Thomas Anderson, Neal Cardwell, Richard Fromm, Kimberly Keeton, Christoforos Kozyrakis, Randi Thomas, and Katherine Yelick. 1997. A Case for Intelligent RAM. IEEE Micro 17, 2 (March 1997), 34–44.

3D-die stacking

- Providing huge bandwidth between memory chips and processors (e.g., HBM)
- The renaissance of near-memory processing!
  - Zhu, Qiuling, Berkin Akin, H. Ekin Sumbul, Fazle Sadi, James C. Hoe, Larry Pileggi, and Franz Franchetti. A 3D-stacked logic-in-memory accelerator for application-specific data intensive computing.In 3DIC. 2013.
  - Dongping Zhang, Nuwan Jayasena, Alexander Lyashevsky, Joseph L. Greathouse, Lifan Xu, and Michael Ignatowski. TOP-PIM: throughput-oriented programmable processing in memory. HPDC '14. 2014
  - Farmahini-Farahani, Amin, Jung Ho Ahn, Katherine Morrow, and Nam Sung Kim. NDA: Near-DRAM acceleration architecture leveraging commodity DRAM devices and standard memory modules. In HPCA. 2015.
  - Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi. A scalable processing-in-memory accelerator for parallel graph processing. ISCA '15. 2015.
  - Youngeun Kwon, Yunjae Lee, and Minsoo Rhu. TensorDIMM: A Practical Near-Memory Processing Architecture for Embeddings and Tensor Operations in Deep Learning. In MICRO '52. 2019

25 years later...

PIM in the HBM era



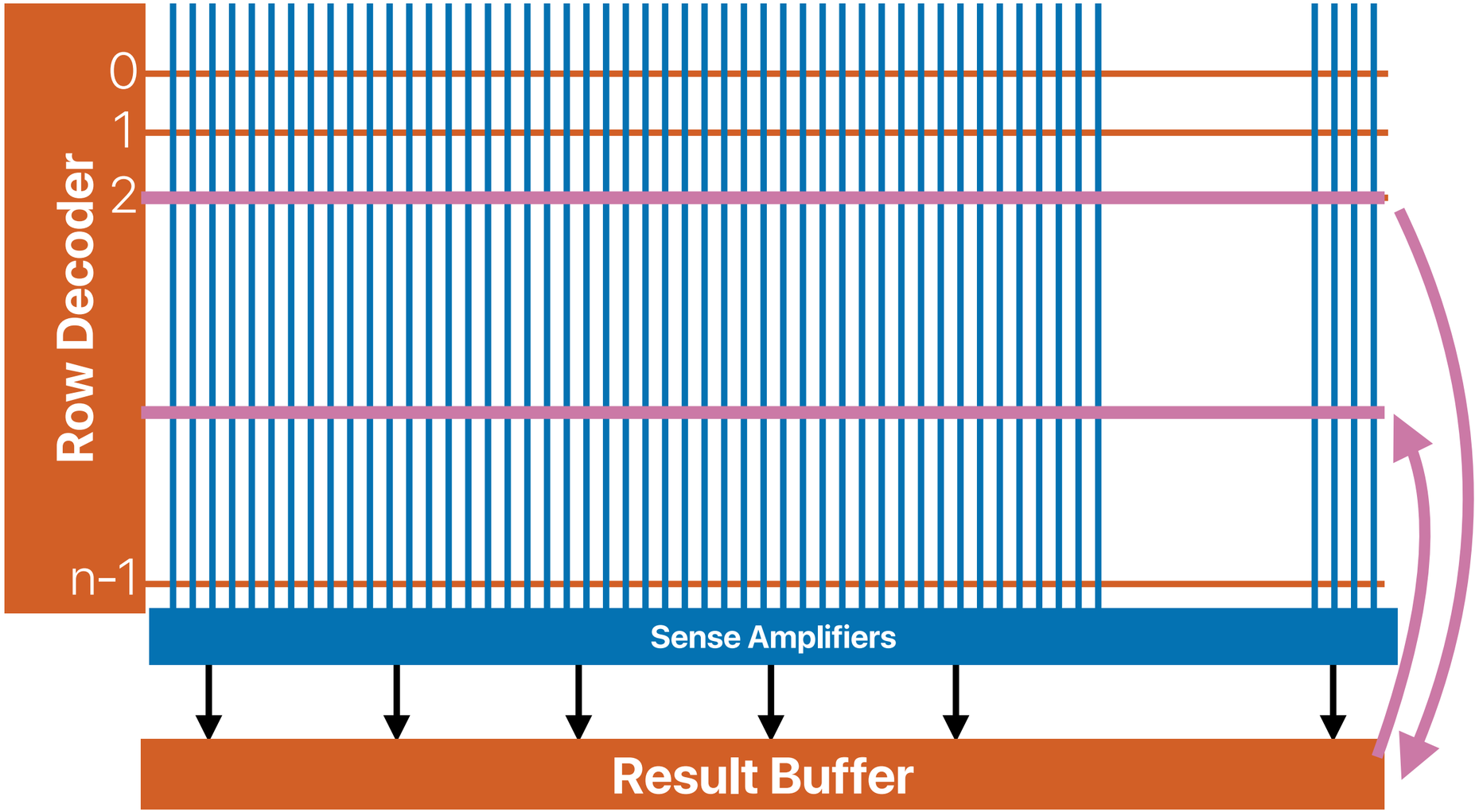
Dongping Zhang, Nuwan Jayasena, Alexander Lyashevsky, Joseph L. Greathouse, Lifan Xu, and Michael Ignatowski. TOP-PIM: throughput-oriented programmable processing in memory. HPDC '14. 2014

Don’t forget the limitations of 3D-die stacking

- Total power consumption of the chip
- If the temperature goes crazy, memory cannot function correctly either

Can we not moving? — “In”-DRAM processing

Recap: Or we just clone/move?

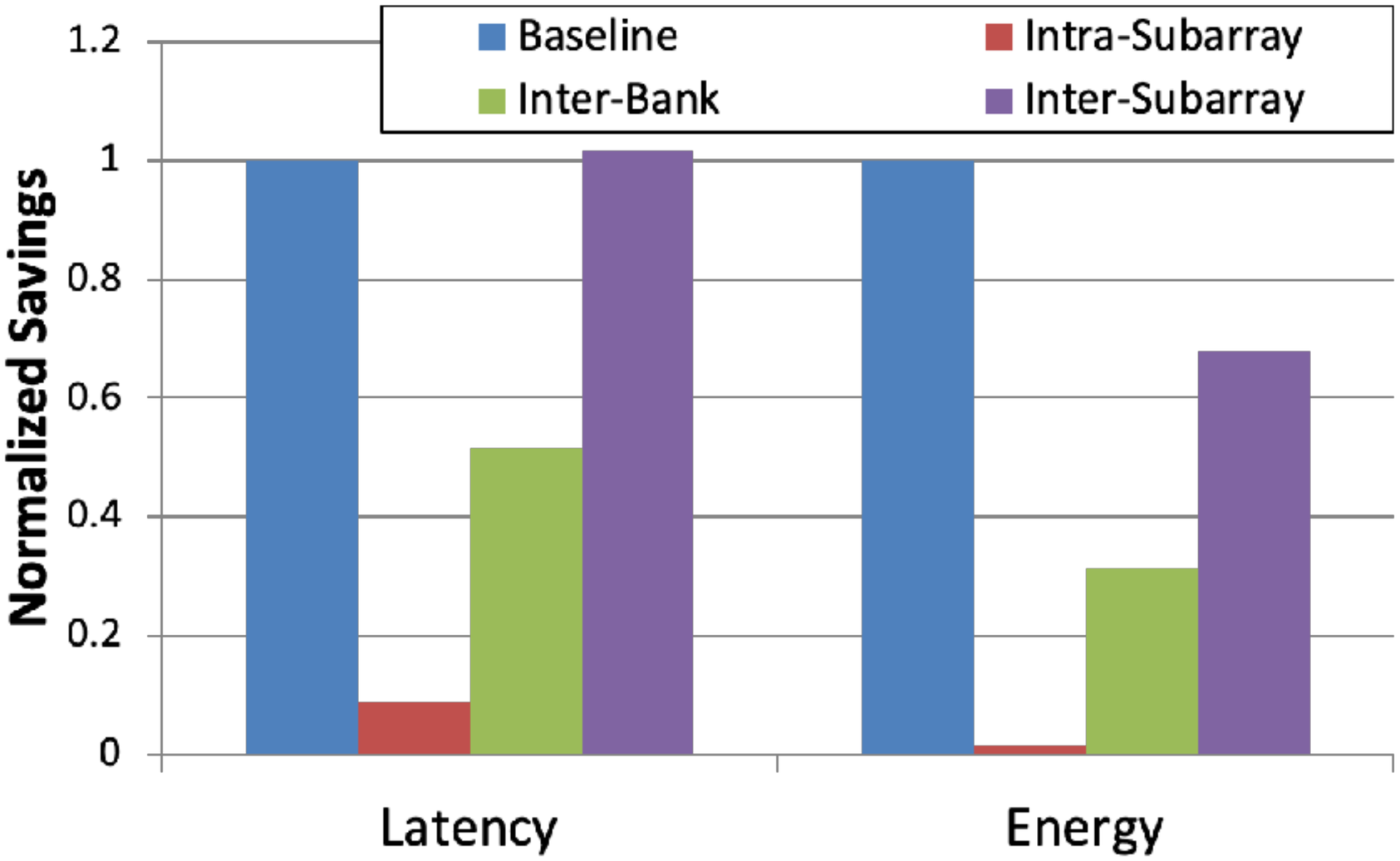


memmove & memcpy: 5% cycles in Google's datacenter

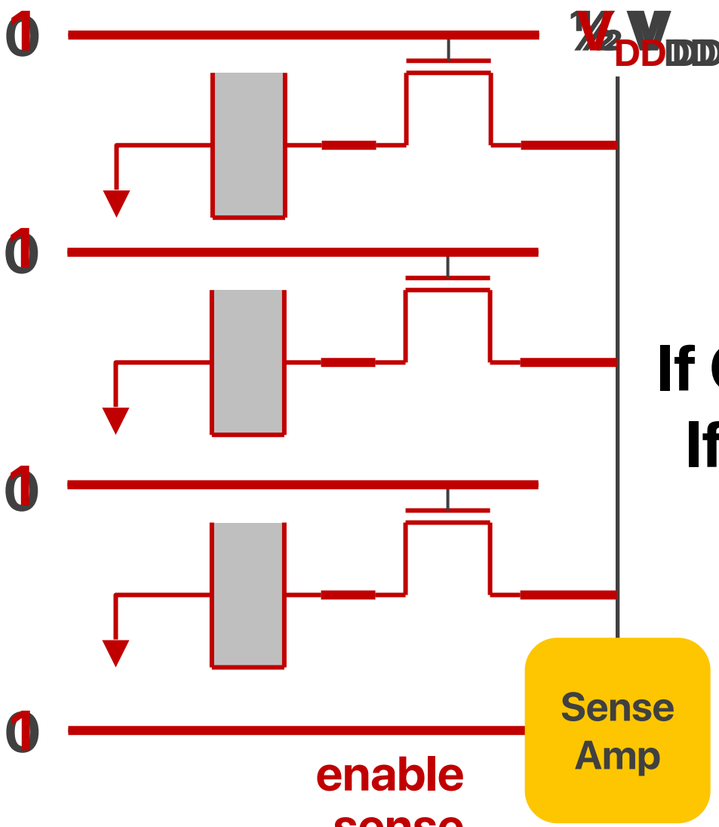
Svilen Kanev, Juan Pablo Darago, Kim Hazelwood, Parthasarathy Ranganathan, Tipp Moseley, Gu-Yeon Wei, and David Brooks. ISCA '15

Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun, Gennady Pekhimenko, Yixin Luo, Onur Mutlu, Phillip B. Gibbons, Michael A. Kozuch, and Todd C. Mowry. RowClone: fast and energy-efficient in-DRAM bulk data copy and initialization. In MICRO-46.

The effect of RowClone



Activate Three Rows



AB + BC + AC =
C(A+B) + C′AB

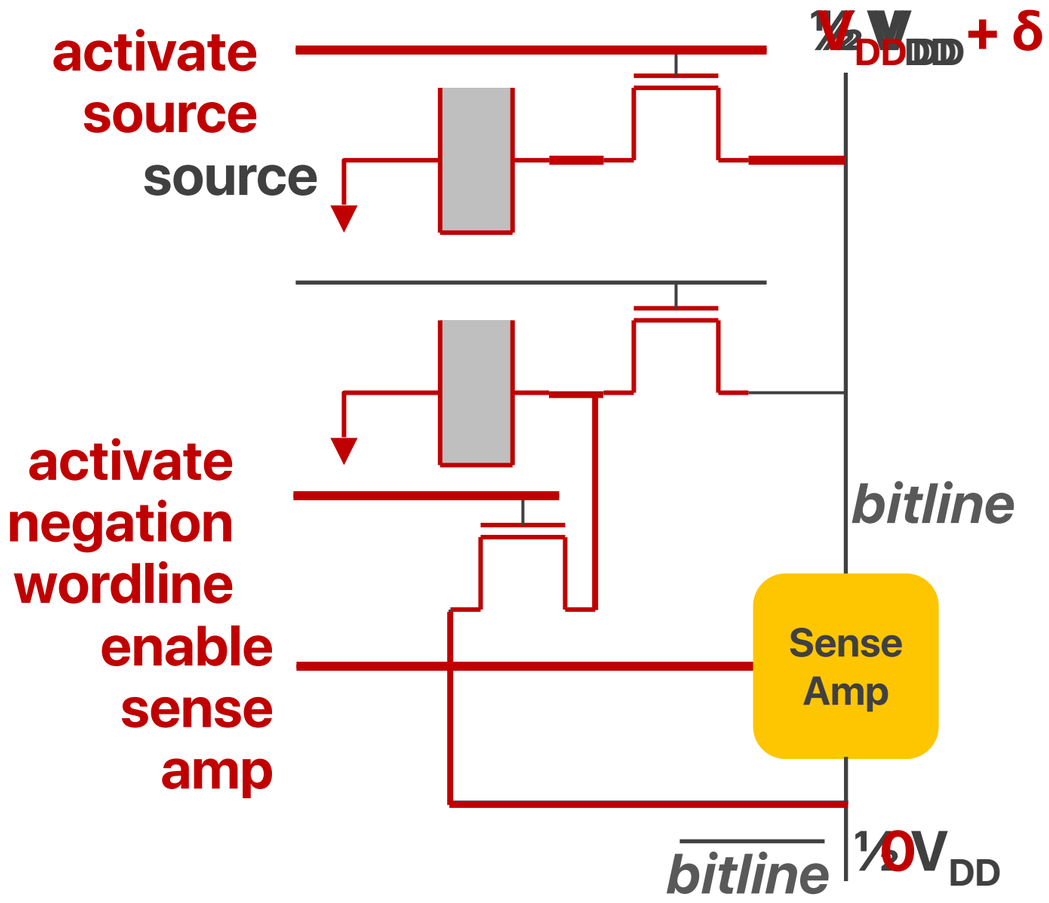
If C is 0, we can compute AND
If C is 1, we can compute OR

Input			Output
A	B	C	
0	0	0	**0**
0	1	0	**0**
1	0	0	**0**
1	1	0	**1**
0	0	1	**0**
0	1	1	**1**
1	0	1	**1**
1	1	1	**1**

		A′B′	A′B	AB	AB′
	Out(A, B)	0,0	0,1	1,1	1,0
C′	0	0	0	1	0
C	1	0	1	1	1

BC AB AC

We can also do NOT



What's the meaning of the ability to perform NAND and NOR?

NAND and NOR are "universal gates" that you can achieve all logical functions with them!

Ambit

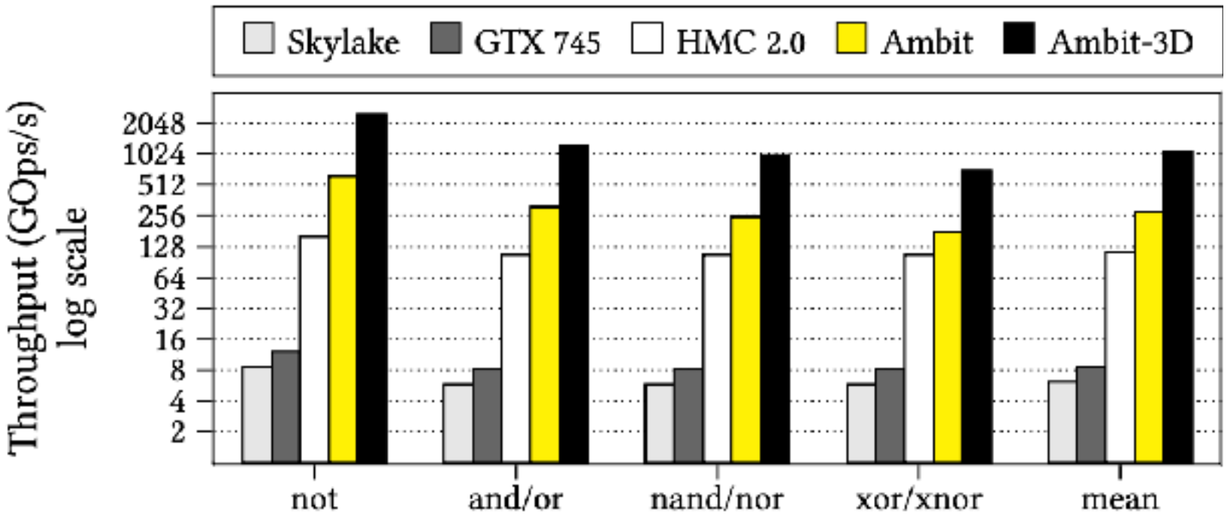


Figure 9: Throughput of bulk bitwise operations.

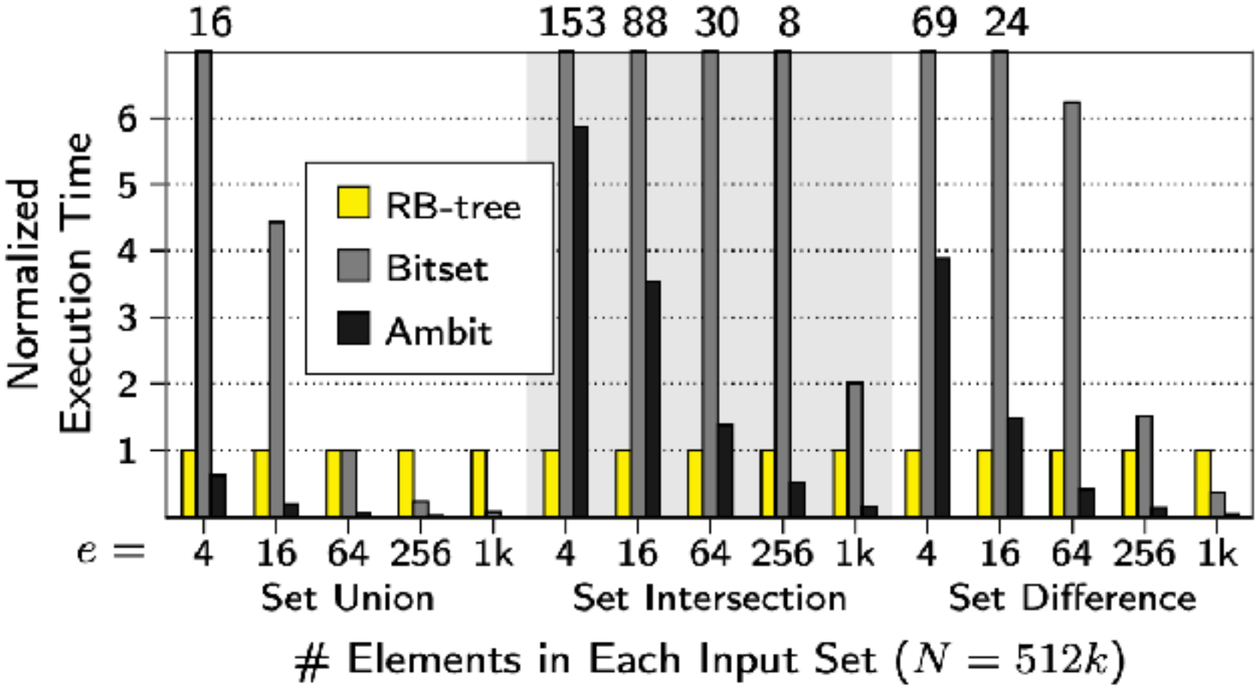


Figure 12: Performance of set operations

Vivek Seshadri, Donghyuk Lee, Thomas Mullins, Hasan Hassan, Amirali Boroumand, Jeremie Kim, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry. Ambit: in-memory accelerator for bulk bitwise operations using commodity DRAM technology. MICRO-50.

3:00

What are the limitations of In-DRAM processing?

Limitations of in-DRAM processing

- Programming
- Hardware design
  - 3 rows of inputs, 3 addresses
  - Not that many operations are available
- How fast is DRAM?
  - Let’s say 100 ns latency — it takes 300 ns to fill three rows and 100 ns to compute
  - Goodput — 4KB every 400 ns each module — 10^10 B — 10GB/sec (but remember, it’s “per-AND/OR/NOT”)

Announcement

- No lecture next Tuesday
- Your presentations starts 5/17 — check website for detailed schedule
  — 18-minute talk (7 minutes for Q/A)
  - Why — the motivation (7 minutes)
    - What's the main problem you're addressing?
    - Why should we care about this?
    - Why is it the right time to do this?
  - What — the idea (6 minutes)
    - What's the proposed idea?
    - What's new about this idea?
  - How — the method/approach (5 minutes)
    - How can we implement the idea?
    - How does the idea perform?

Electrical
Computer Science
Engineering

277

つづく