list
cea tech

DE LA RECHERCHE À L'INDUSTRIE

Aix*Marseille université
Initiative d'excellence

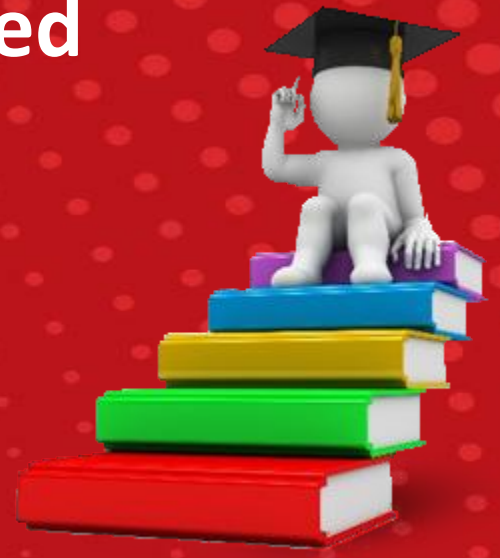
Exploration and conception of computing architecture of type *in-memory computing* based on emerging non volatile memories

23/10/18-22/10/21

PhD Director: Jean-Michel Portal

PhD Supervisor: Mathieu Moreau

CEA Supervisor: Jean-Philippe Noel



Outline

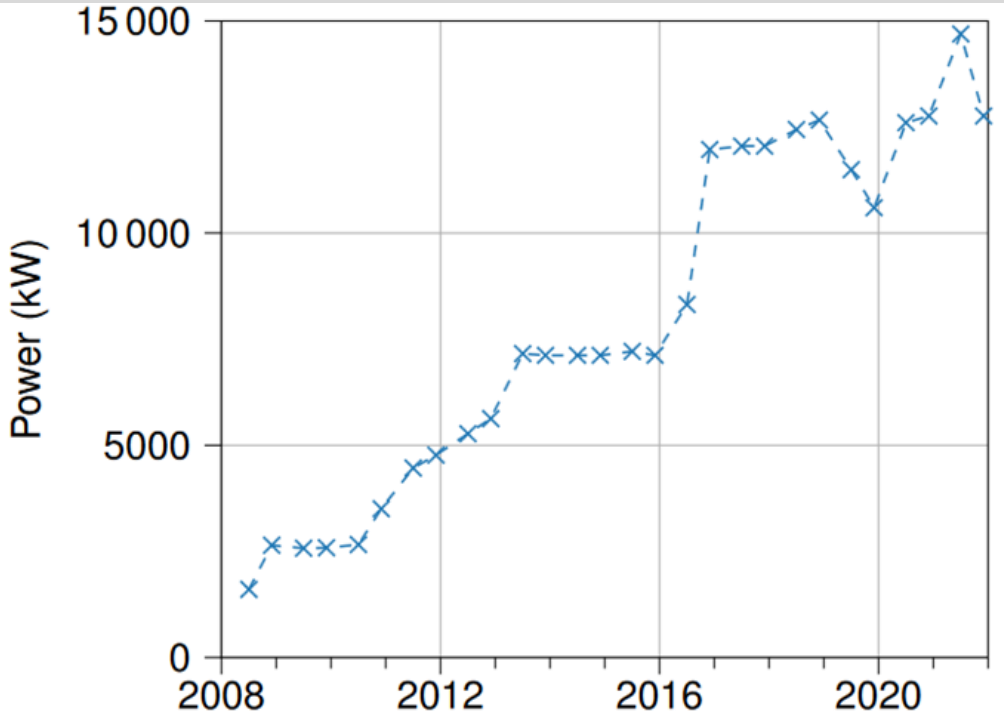
- General context
- State of the art on different technologies and implementations
- CSRAM: our own design approach
- Software platforms and explorations
- Results and analysis on different memory computing architectures
- Conclusion and future works

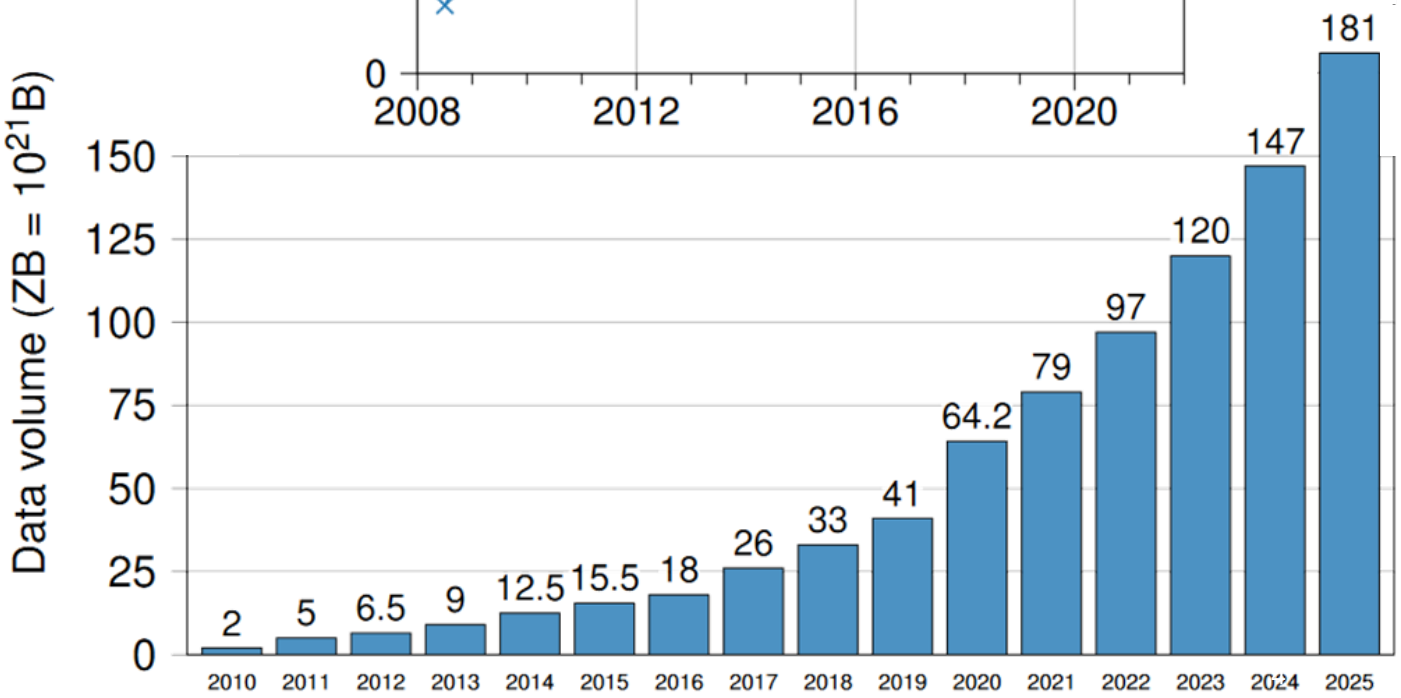
Outline

- **General context**
  - Semiconductor industry and issues on current system architectures
  - Introduction to memory systems and memory hierarchy
  - Main thesis idea → memory computing
- **State of the art on different technologies and implementations**
- **CSRAM: our own design approach**
- **Software platforms and explorations**
- **Results and analysis on different memory computing architectures**
- **Conclusion and future works**

General context : Socioeconomic cost

- **Transition to digital services consumes more and more power**
  - Goes against energy transition
  - Already 3% of world electric energy consumption (data centers + networks)
- **And creates more and more data**
  - Data that needs to be treated and stored
  - 40% growth rate per year

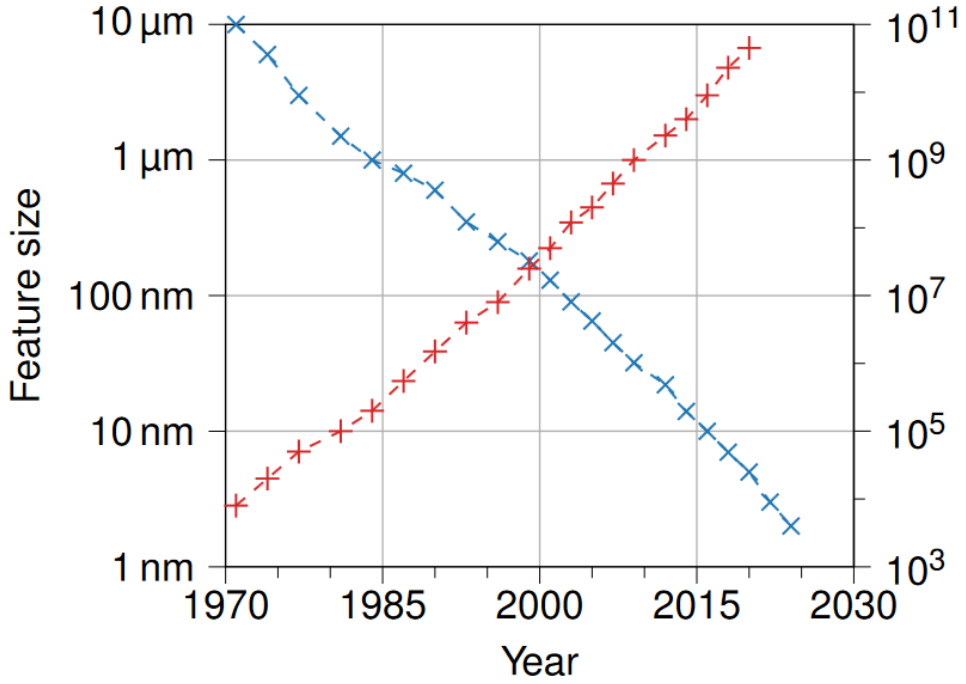




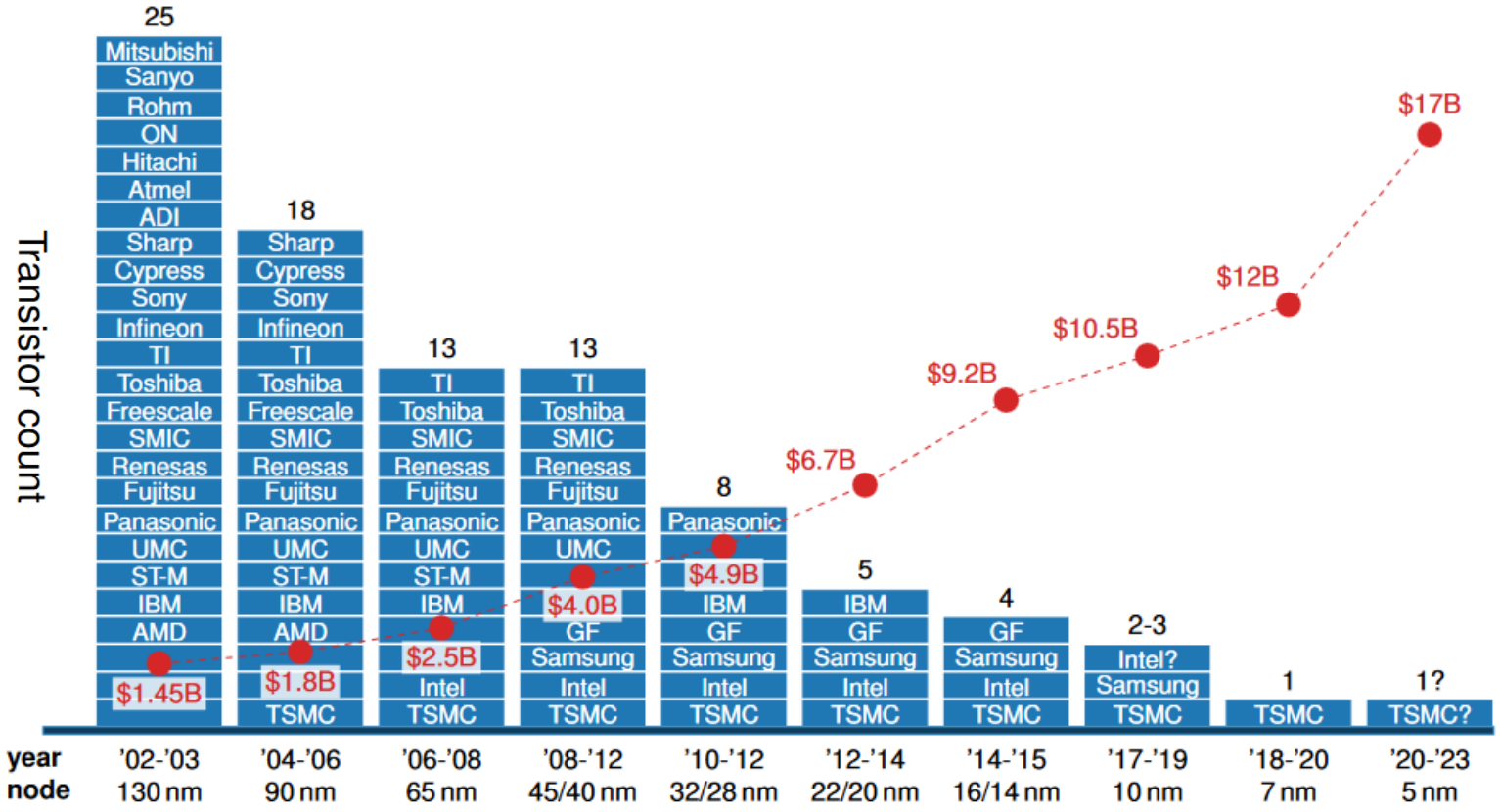
TOP500, 2022, top500.org
S. Brume, "Data Durability, and Back-up at scale: A tale of the Tape" 2022, ibm.com

General context : technology faces a dead end

Feature size reduction reaches an end



Cost of more advanced node is skyrocketing



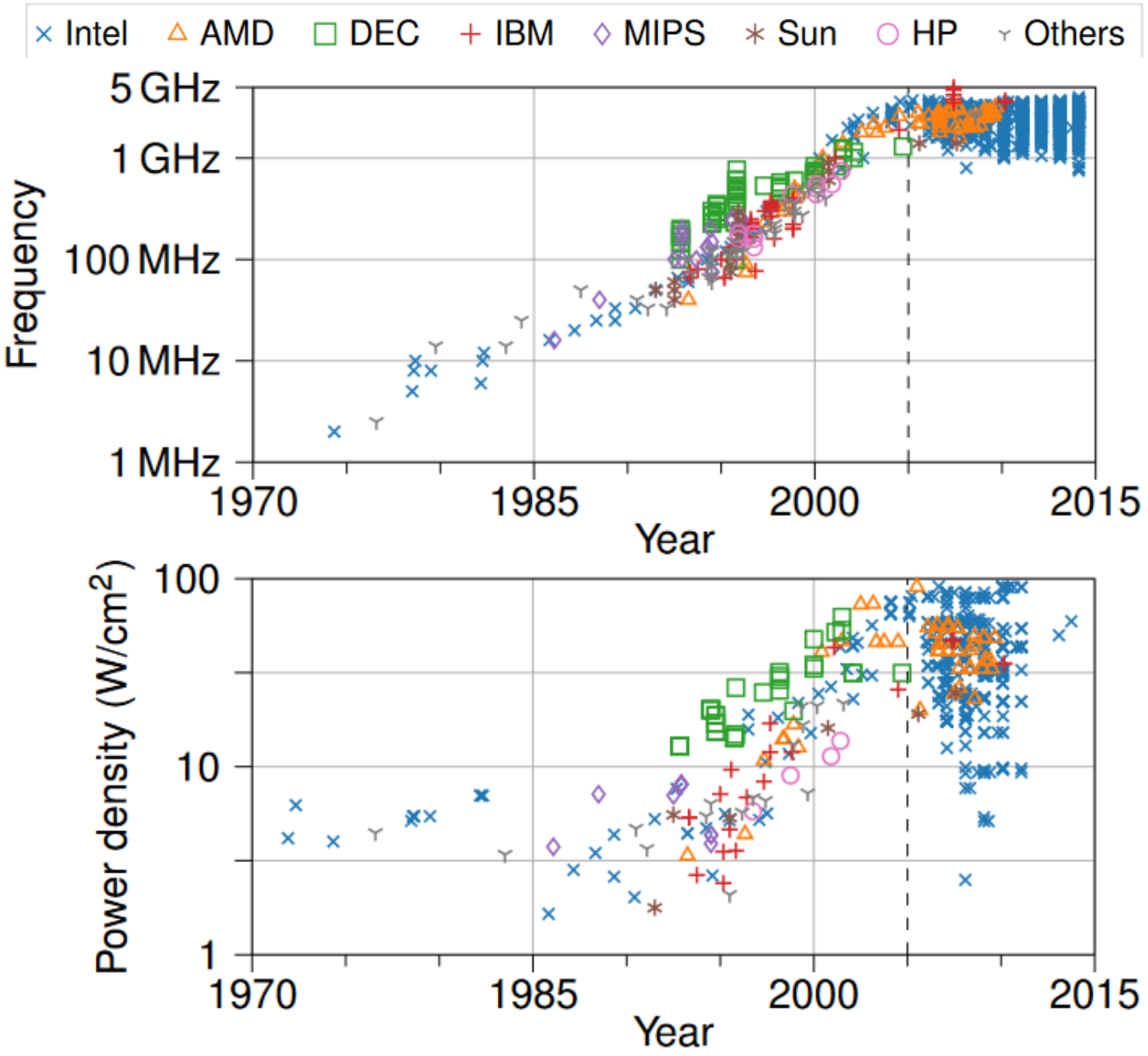
ITRS roadmap, 2022, www.itrs2.net
V. Bertacco. “Re-Imagining Scalable System Design”, 2018, VLSI-SoC

General context : technology faces a dead end

- **Dennard's scaling broke in 2005**
  - Frequency stall in all CPUs around 3GHz

$$P_{dyn} = CV^2 f$$

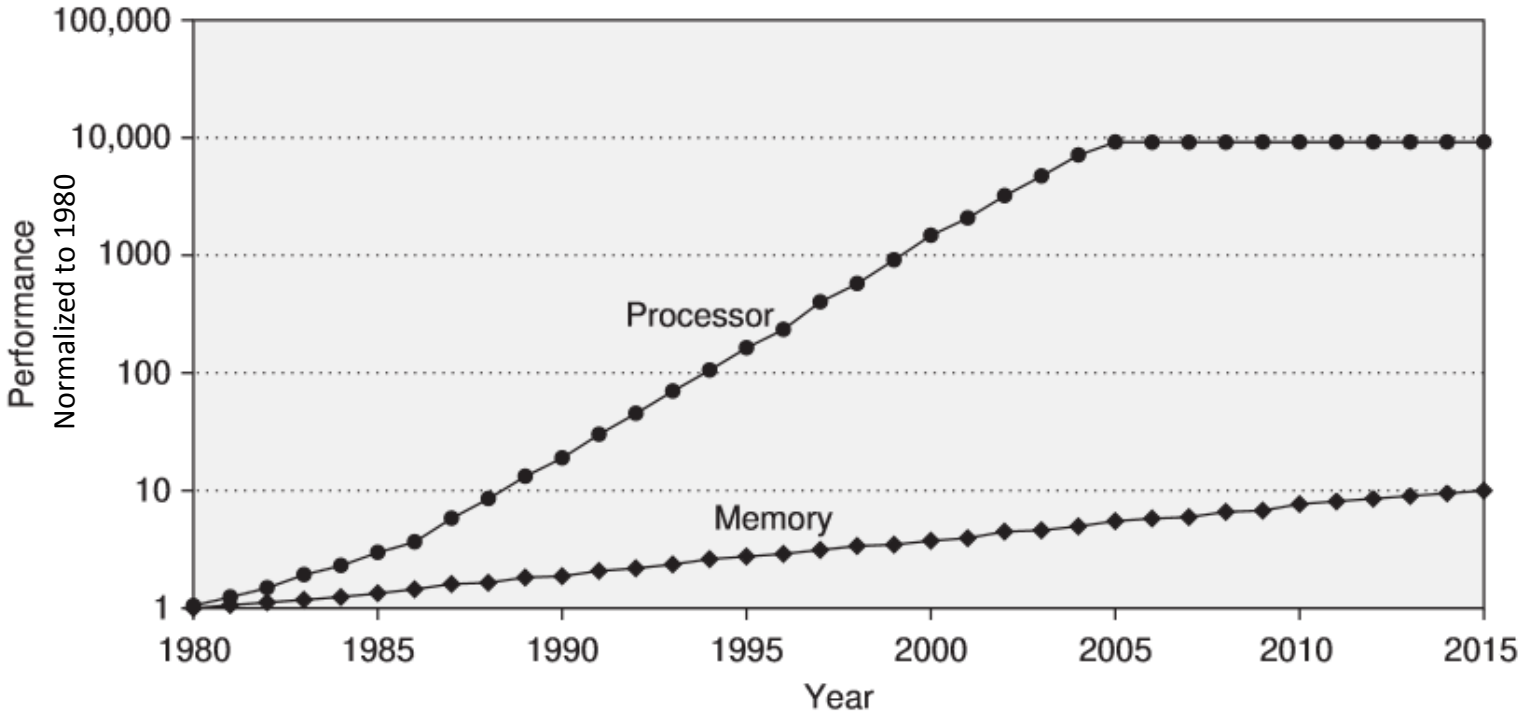
  - Max power density capped at 100W/cm² due to thermal constraint
  - Dark Silicon



A. Danowitz et al., "CPU DB", 2012, cpudb.stanford.edu

General context : memory wall

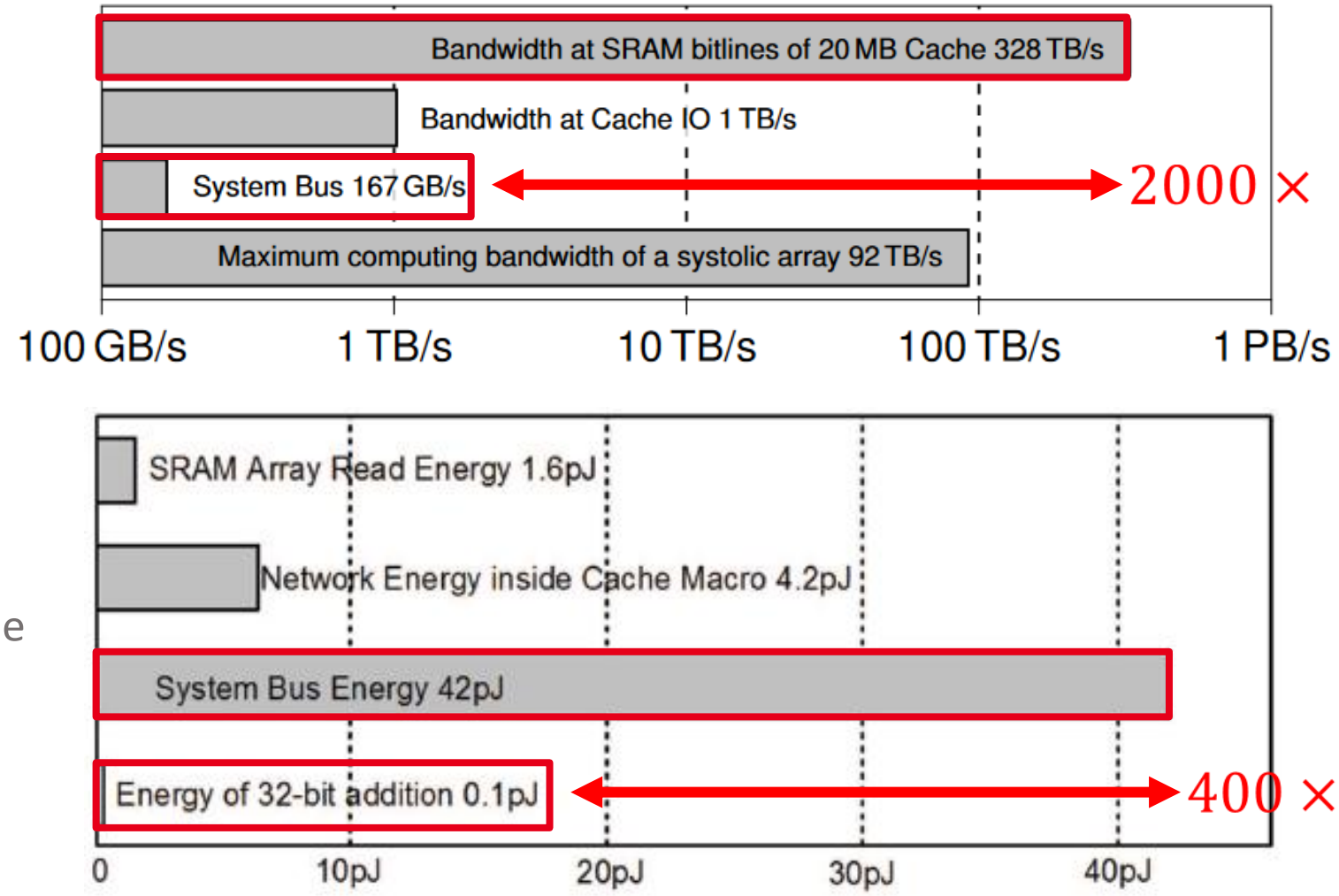
- **Processor performances improvements exceeds memory's**
  - Different foundry processes
  - Limited hardware innovation in memories



John L Hennessy. "Computer Architecture: A Quantitative Approach", 2016

General context : memory wall

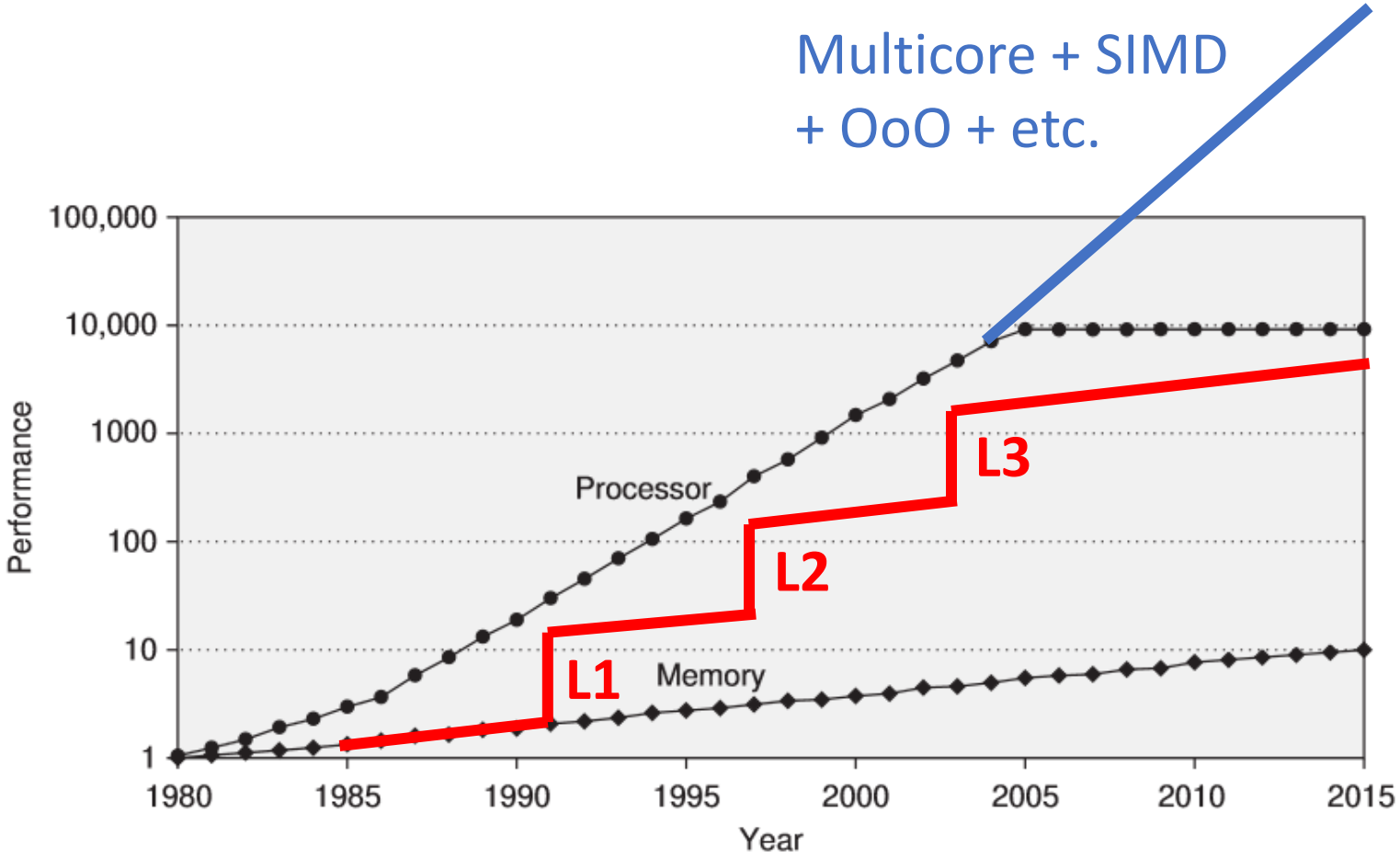
- **Internal bandwidth is decimated by IO interfaces**
  - SA (analog to digital)
  - Bus interfaces
- **Memories are underutilized and waste 50% to 80% of the energy**
  - Full line read (64B) just to access a single byte



J. Wang et al., “A 28-nm Compute SRAM”, 2019, JSSC

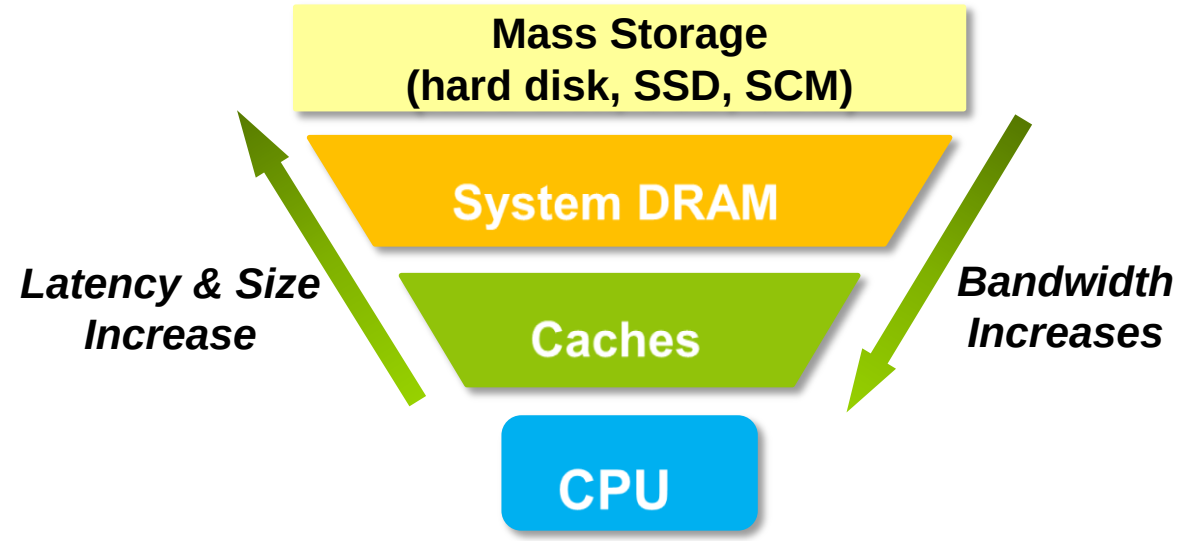
Memory Hierarchy

- **To bring memory to the performance level of processors, a memory hierarchy is instated**
  - Prevents the processor from waiting doing nothing
  - Every time processor performance scales up, a new memory is inserted
- **Instruction centric point of view with lots of hardware innovation on CPU side**
  - Single Instruction Multiple Data (SIMD)
  - Branch prediction
  - Out of Order execution
  - Speculative execution



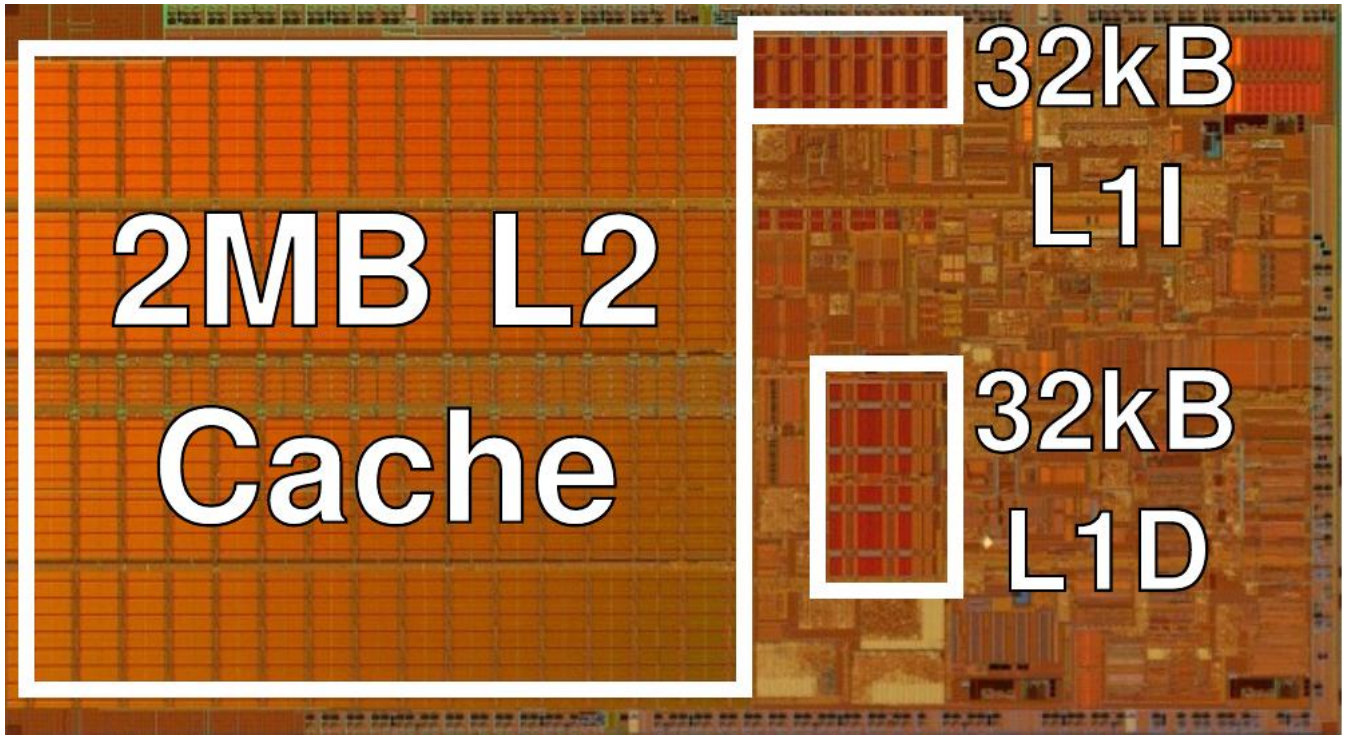
John L Hennessy. “Computer Architecture: A Quantitative Approach”, 2016

Memory hierarchy

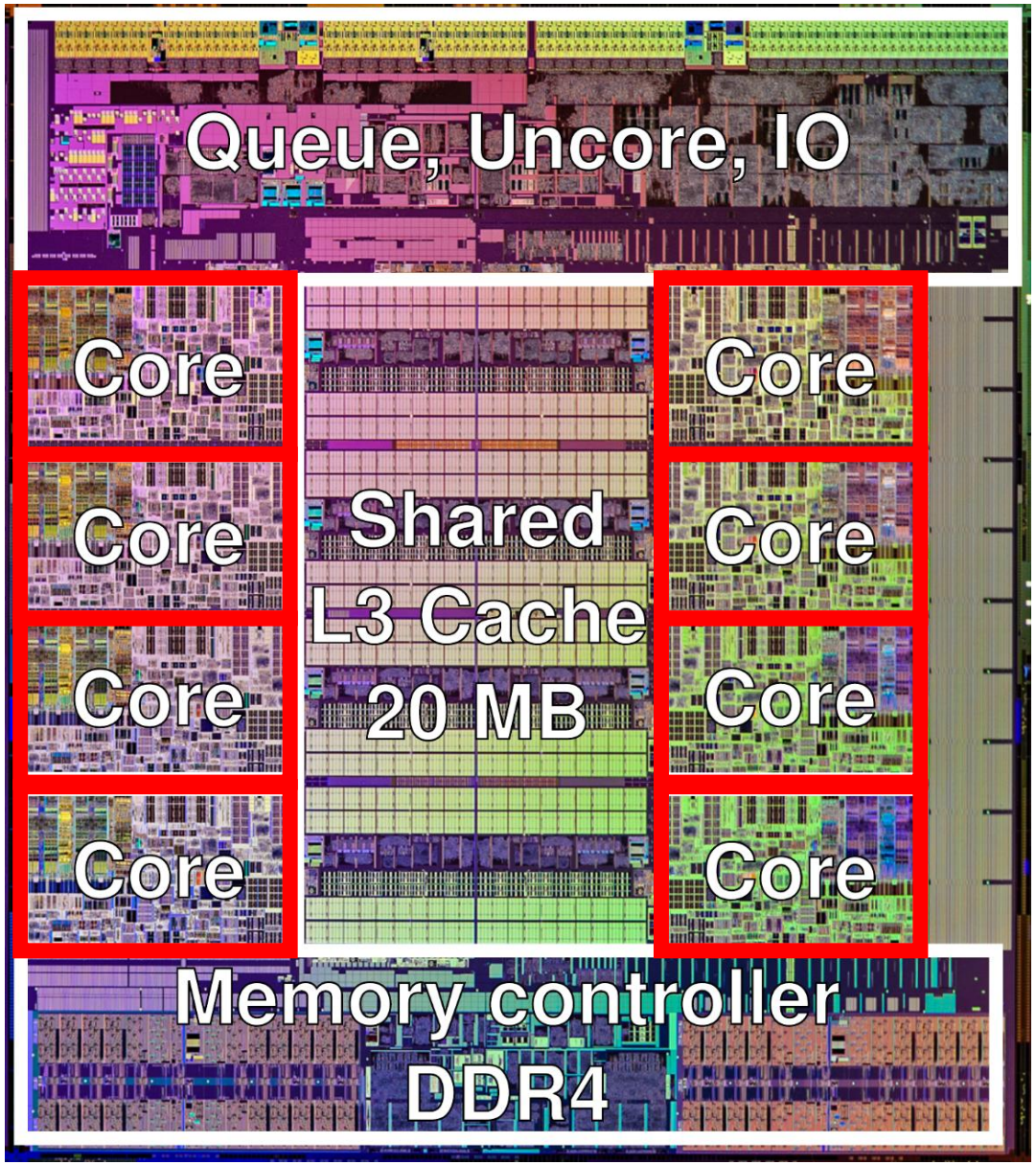


	Price ($/GiB)	Density	Latency	Bandwidth	Persistence	Largest size
SRAM	5000	120 F^2 or 2 Gbit/cm^2	1–50 ns	1 TiB/s	10 µs (Power off)	10–100 MiB
DRAM	20	8 F^2 or 25 Gbit/cm^2	20–400 ns	10–100 GiB/s	64 ms	100–1000 GiB
Flash	4	<1 F^2 or >100 Gbit/cm^2	1–10 µs	1 GiB/s	10–20 yr	10–100 TiB
HDD	0.1	100 Gbit/cm^2	5–20 ms	100 MiB/s	10–100 yr	10–100 TiB

Illustration of low SRAM density



130nm Intel Pentium M (2004)

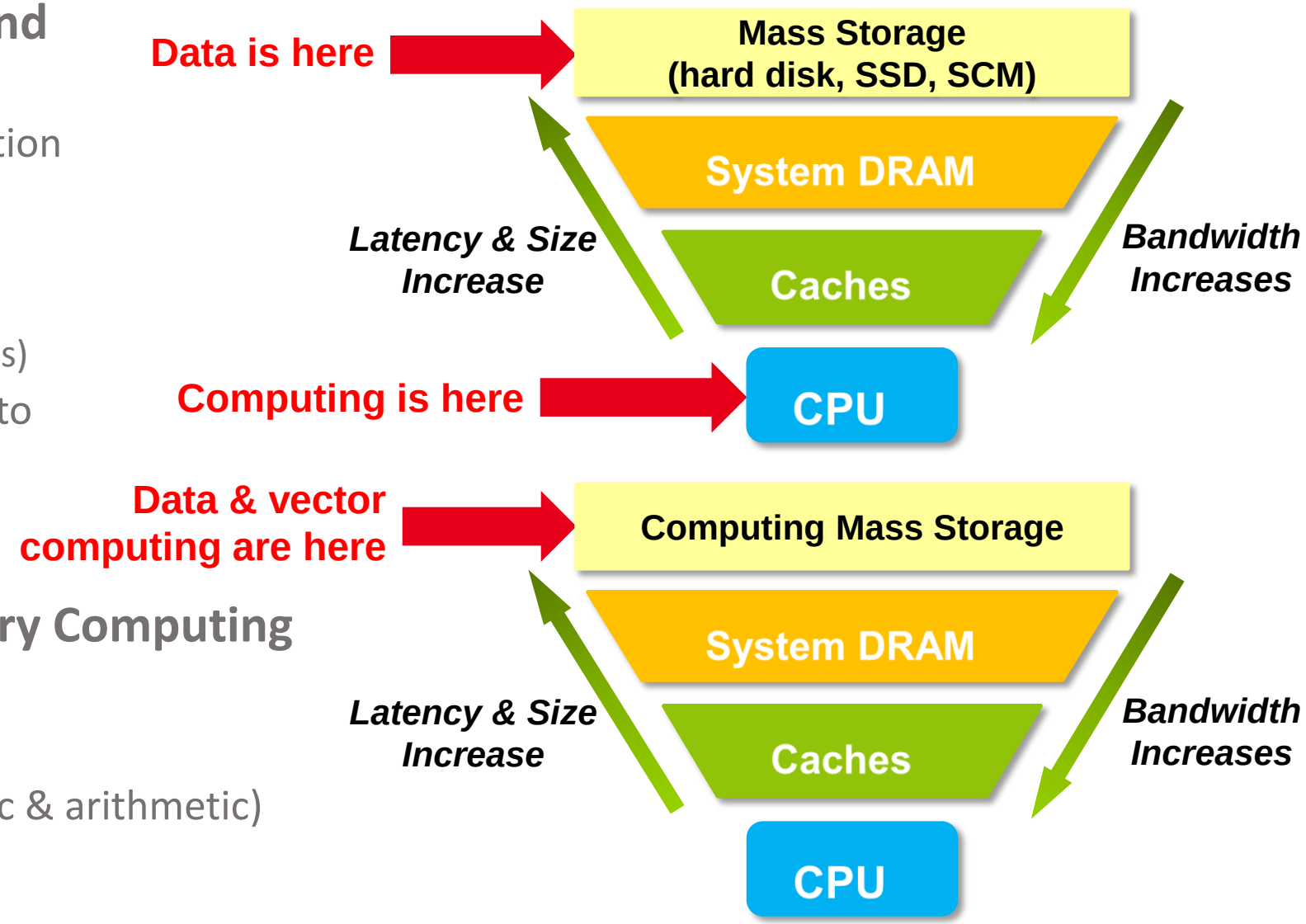


22nm Intel Haswell 8 cores (2014)

wikichip.org

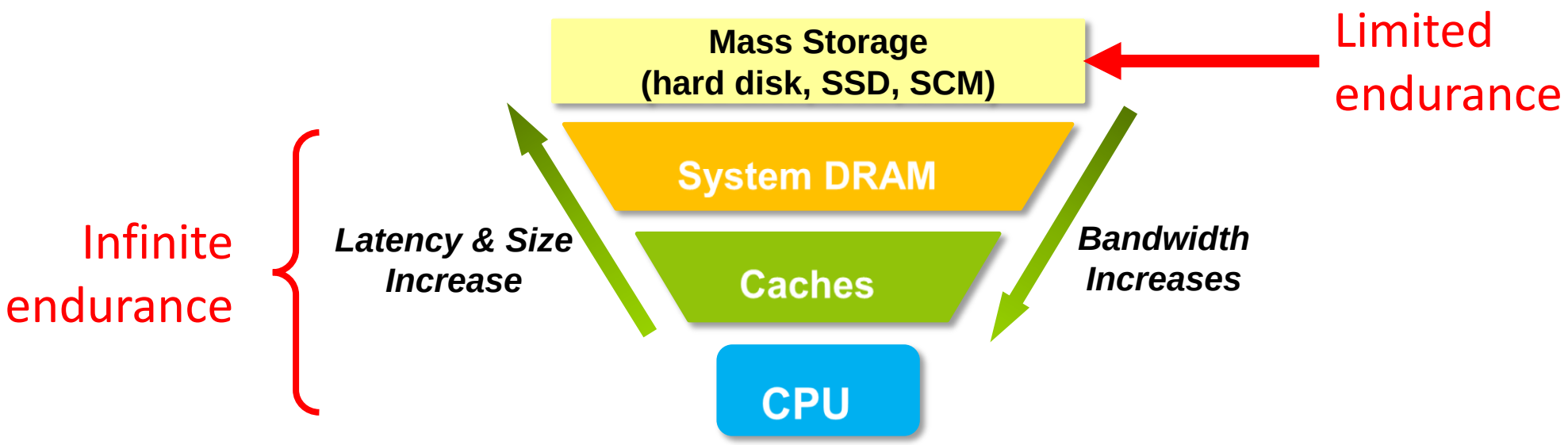
Close computing is far away from computing

- **von-Neumann Architecture and Bottleneck**
  - Linked to data intensive application (memory bound)
    - Big data era
    - Neural networks
    - Irregular access patterns (graphs)
  - Data keeps moving around due to coarse grained blocks



- **Proposed solution: In-Memory Computing**
  - Data is in memory
  - Bigger parallelism
  - But only simple operations (logic & arithmetic)

Memory hierarchy

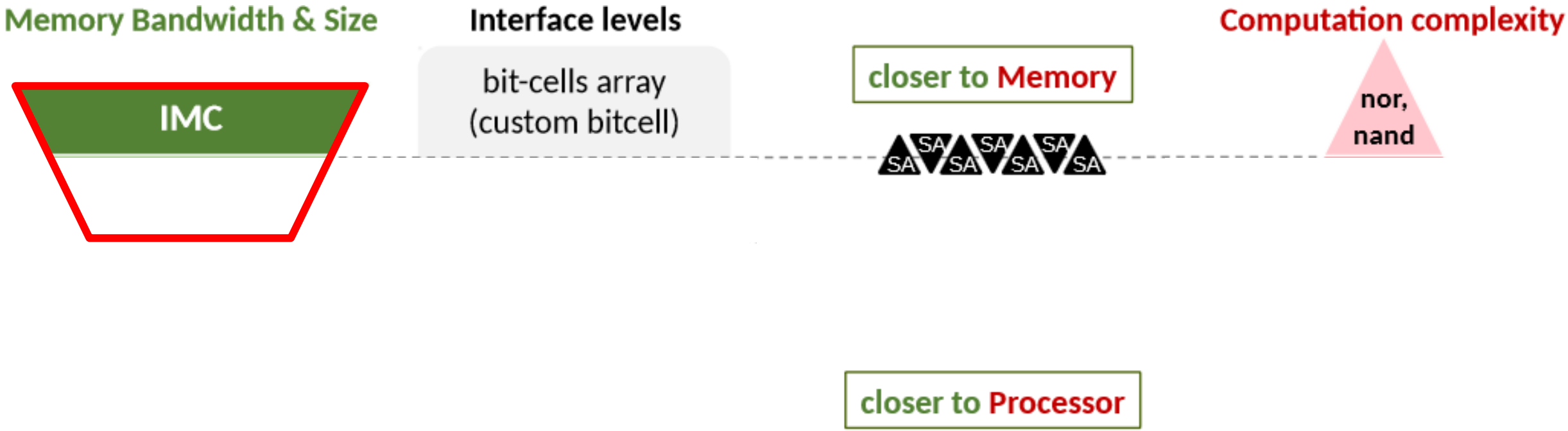


	Price ($/GiB)	Density	Latency	Bandwidth	Persistence	Largest size
SRAM	5000	120 F^2 or 2 Gbit/cm^2	1–50 ns	1 TiB/s	10 µs (Power off)	10–100 MiB
DRAM	20	8 F^2 or 25 Gbit/cm^2	20–400 ns	10–100 GiB/s	64 ms	100–1000 GiB
Flash	4	<1 F^2 or >100 Gbit/cm^2	1–10 µs	1 GiB/s	10–20 yr	10–100 TiB
HDD	0.1	100 Gbit/cm^2	5–20 ms	100 MiB/s	10–100 yr	10–100 TiB

Outline

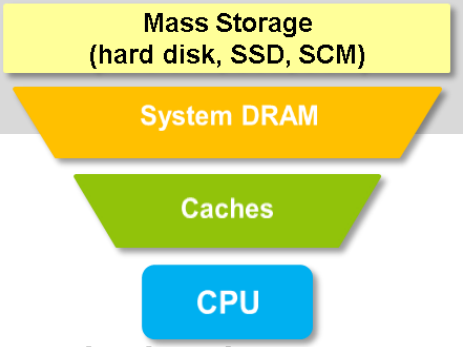
- General context
- **State of the art on different technologies and implementations**
  - Taxonomy
- CSRAM: our own design approach
- Software platforms and explorations
- Results and analysis on different memory computing architectures
- Conclusion and future works

Taxonomy



R. Gauchi, “Exploration of Reconfigurable Tiles of Computing-in-Memory Architecture for Data-intensive Applications”, 2021

State of the art



	Memory Technology	Architecture	Key Results
[1] ReRAM CIM	☑ Circuit is demonstrated physically with ReRAM	☒ Standalone circuit	☒ No mention of endurance ☒ Logic operations only
[2] NAND Flash CIM	☒ Analogic computation (cu		☑ No write to Flash **40 TOPS/W on MAC**
[3] Ambit	☒ DR		Logic operations only **32x speedup** **35x energy saving on database**
[4] UPMEM	☒ Har logi with		No vector processing **20x speedup** **10x better power efficiency**
[5] Compute Caches	☑ SRAM with standard bitcell	☑ HPC Architecture ☒ Stay in cache ☒ Not concerned about DRAM consumption	☒ Logic operations ☑ **1.9x speedup** ☑ **2.4x energy reduction**
[6] Blade	☒ SRAM with non standard organization	☒ MCU Architecture	☑ Logic operation and addition ☑ **6x speedup vs NEON**

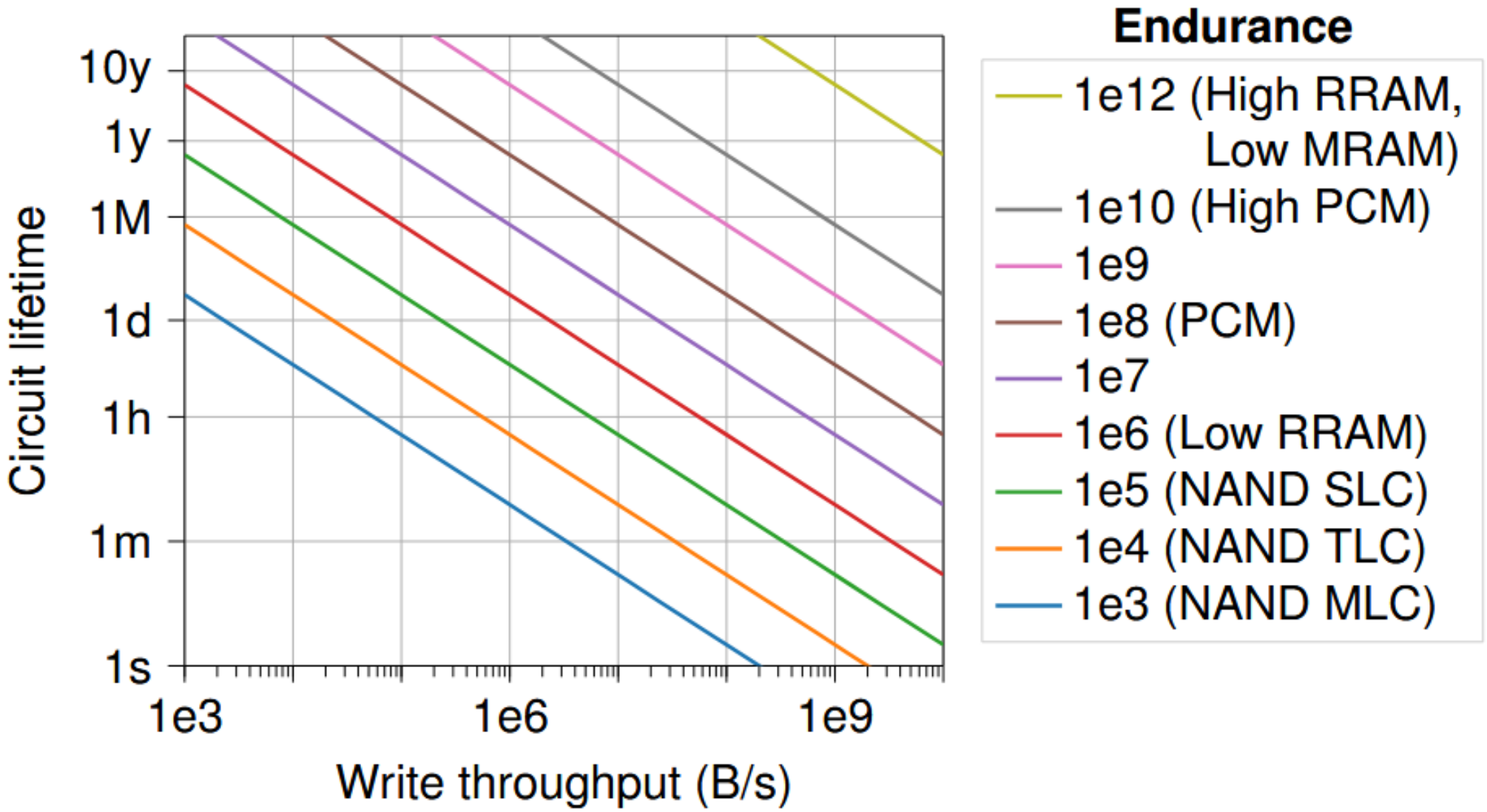
- No paper with full architecture & data coming from SCM
- No paper with a mix of SRAM & SCM to preserve the latter's endurance
- Analog computation in SCM gives very few freedom in operators and has lots of constraints

Limited endurance

High power consumption

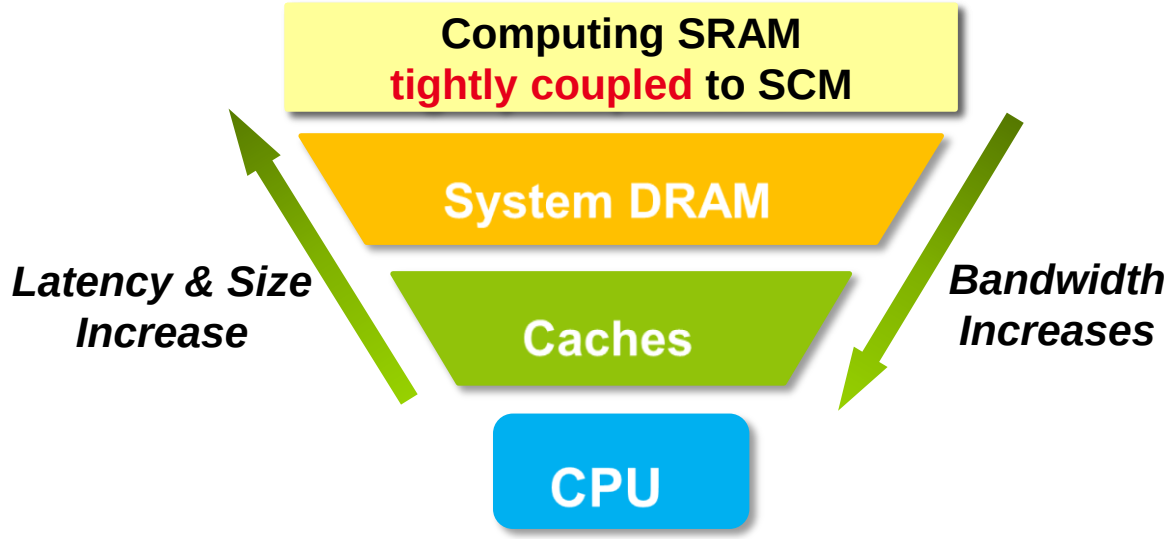
Data is not naturally here

Endurance is the limiting factor in NVMs



Main thesis idea

- **Merge computing and data in the same circuit**
  - Better energy efficiency and low speed
  - Endurance is a problem (10^9)
  - Use existing row buffer and repurpose it for computing



- **DRAM is still used as a write buffer**
- **Caches can be reduced (Instructions & stack)**

Let's move computing to the top with tightly coupled SRAM to get the best of both worlds!

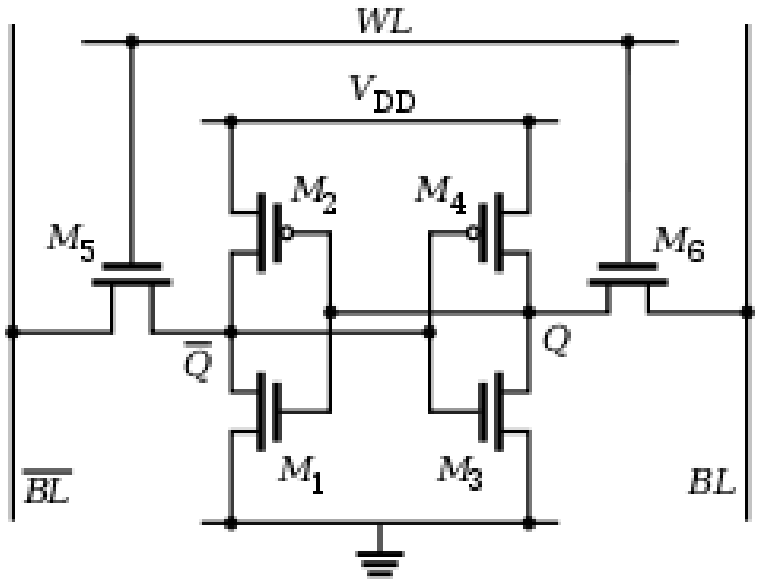
Outline

- General context
- State of the art on different technologies and implementations
- **CSRAM: our own design approach**
  - SRAM based design
  - Automated design approach
  - Comparison with state of the art
- Software platforms and explorations
- Results and analysis on different memory computing architectures
- Conclusion and future works

Different bitcells

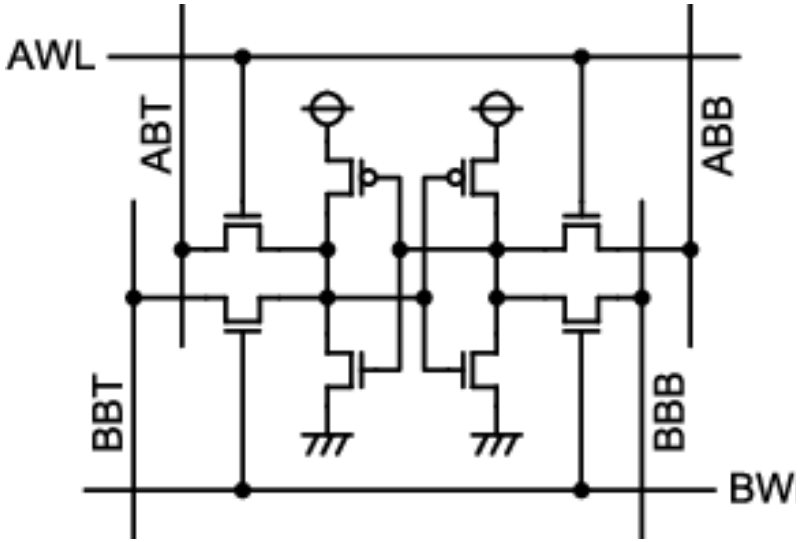
Single Port (1RW)

- 6 transistors



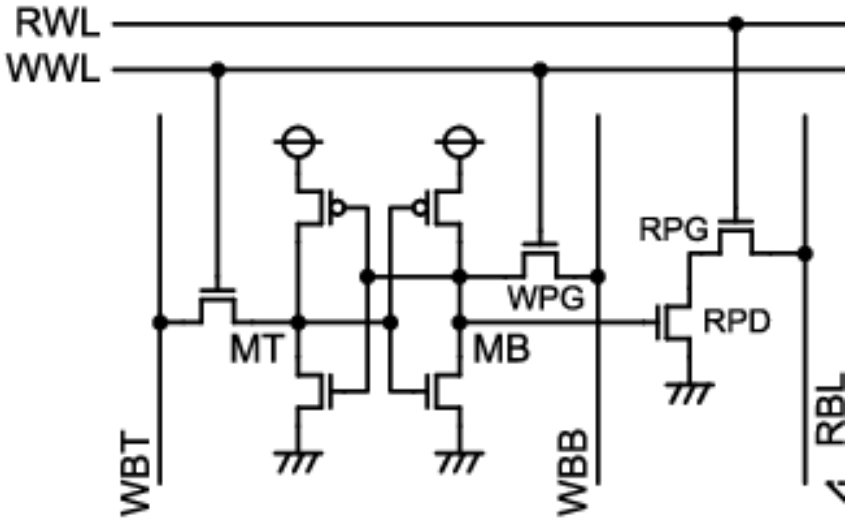
Dual Port (2RW)

- 8 transistors



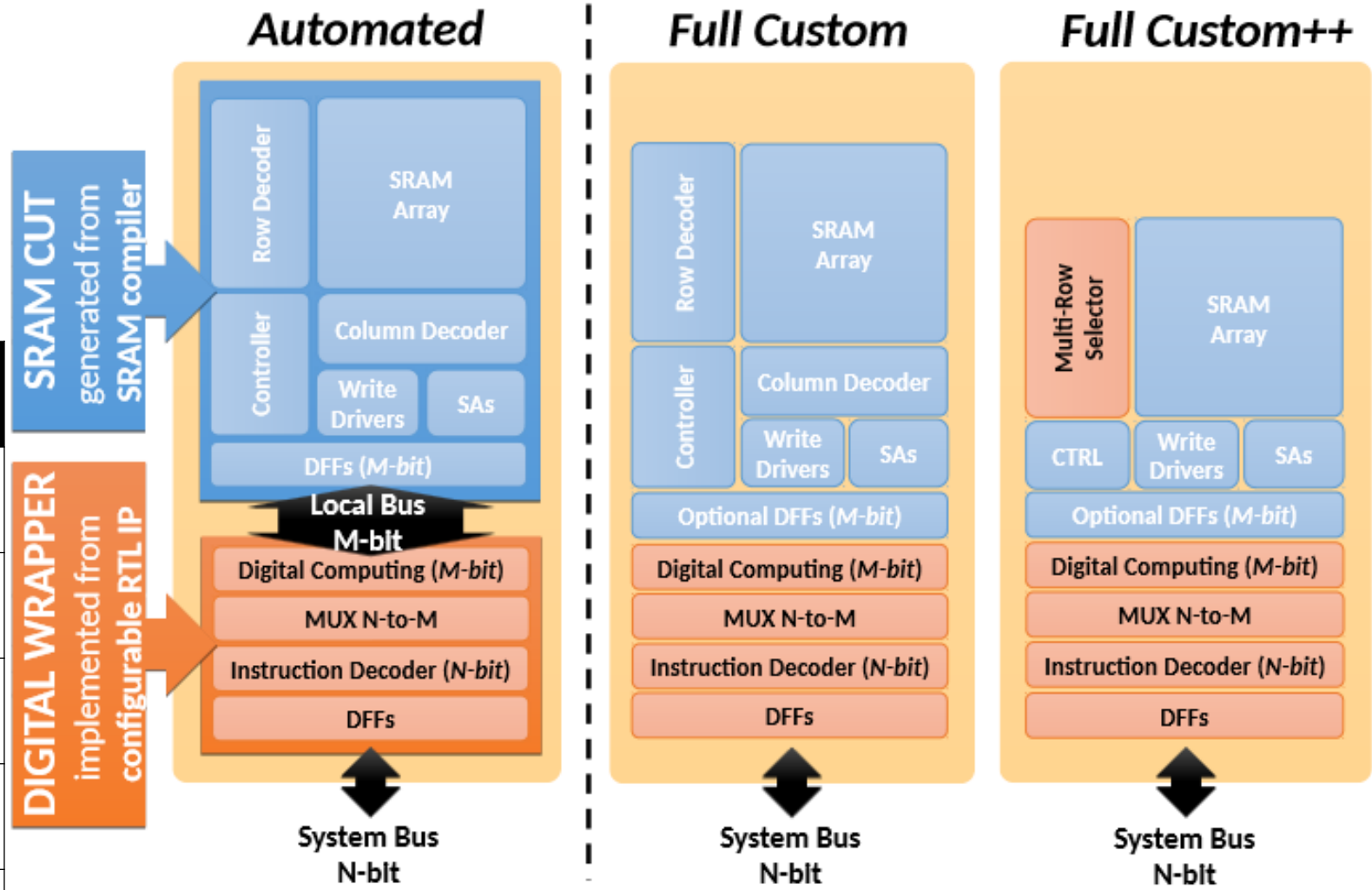
Two Port (1R1RW)

- 8 transistors



Design methodology

Design Methodology	Full Custom ++
Computing classification	*NMC/IMC*
Computing area overhead	*low*
Form factor Flexibility	*low*
Development effort*	*very high*
Silicon quali-fication effort	*very high*

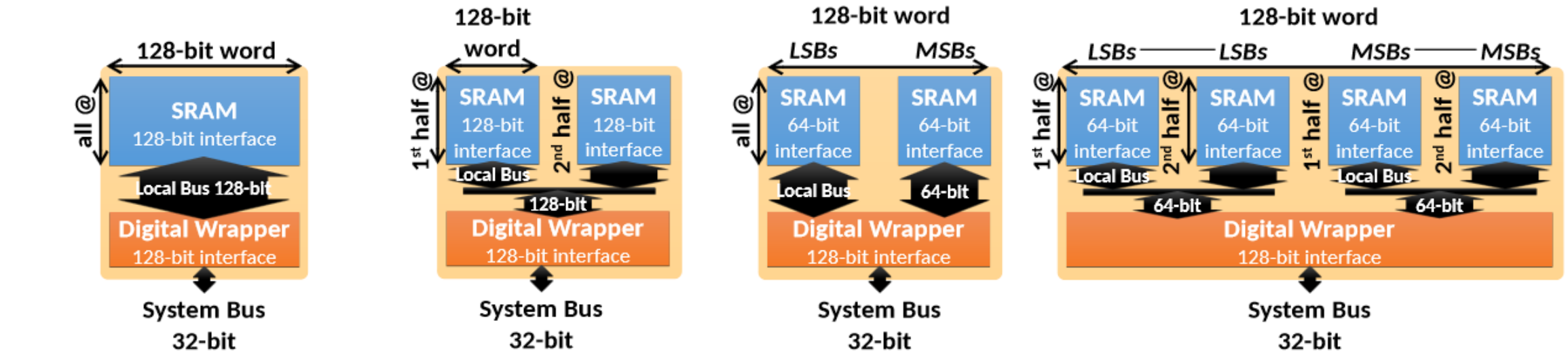


J.P Noel et V. Egloff, “Computational SRAM Design Automation using Pushed-Rule Bitcells for Energy-Efficient Vector Processing”, DATE 2020

Benefits from automated design

Form factor flexibility

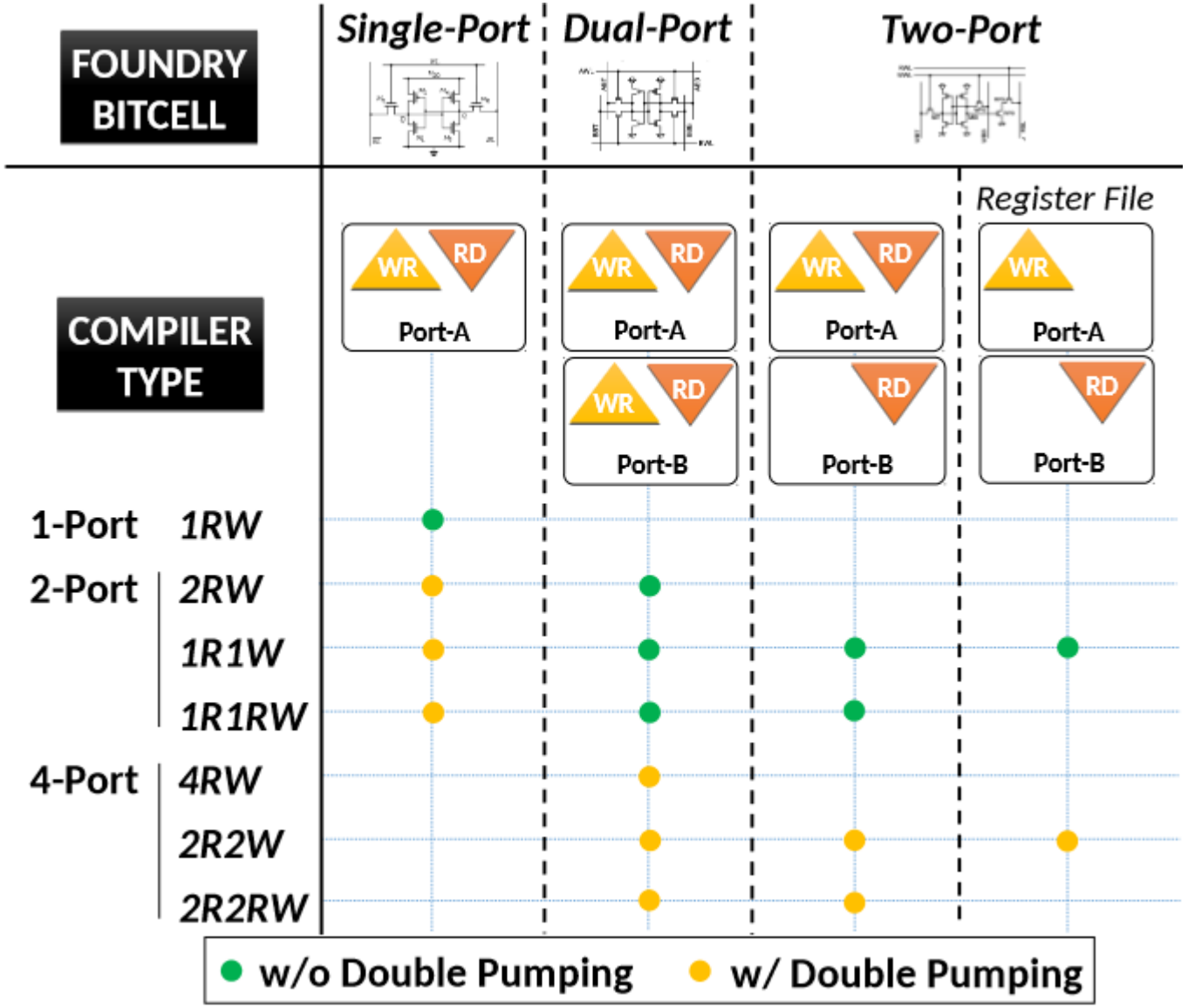
C-SRAM Macro = *Memory Cut Partitioning w/ Digital Wrapper*



J.P Noel et V. Egloff, “Computational SRAM Design Automation using Pushed-Rule Bitcells for Energy-Efficient Vector Processing”, DATE 2020

Benefits from automated design

- **Double pumping**
  - Makes memory active on both rising and falling edges
  - Slightly decreases frequency

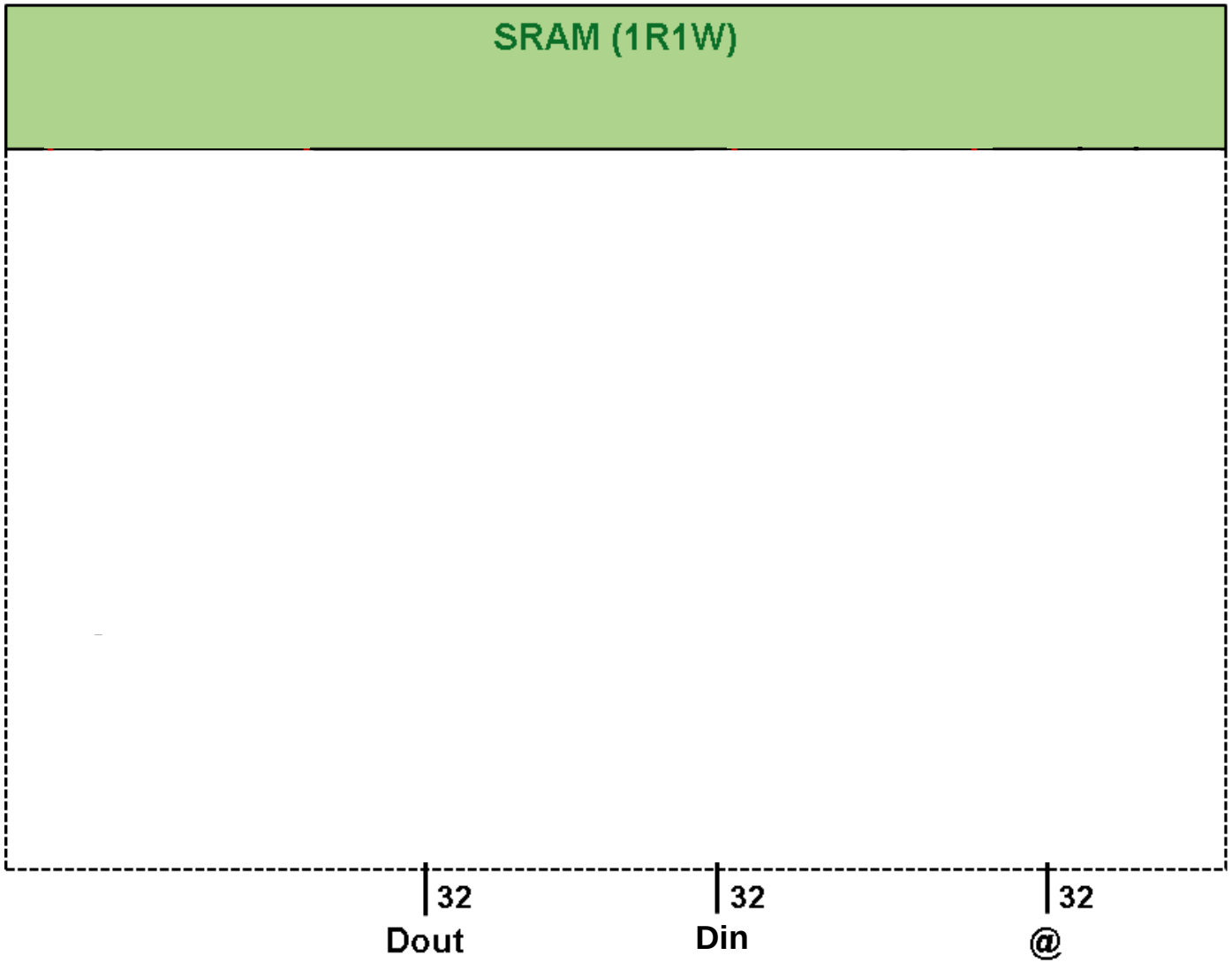


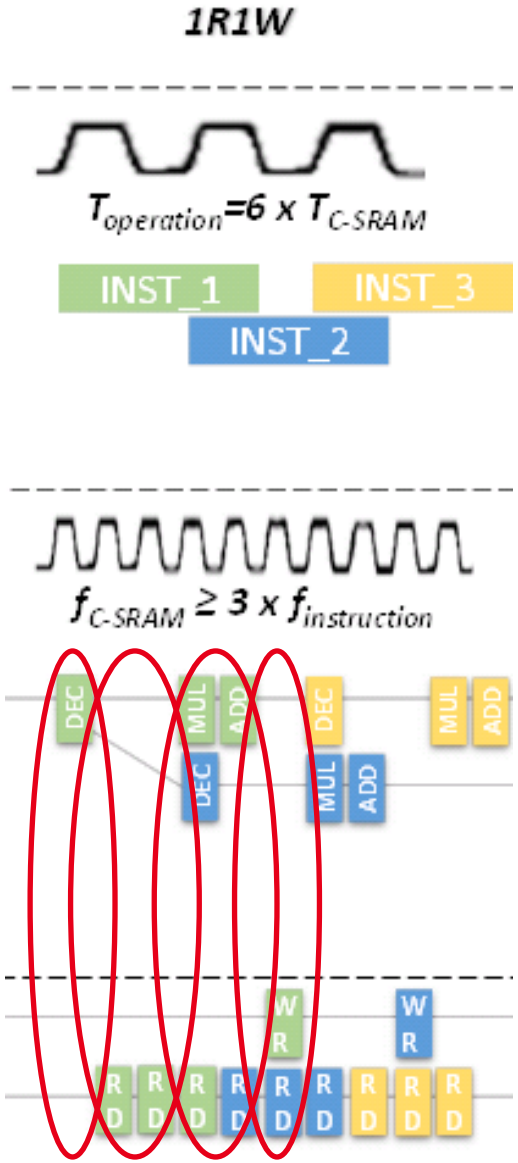
J.P Noel et V. Egloff, “Computational SRAM Design Automation using Pushed-Rule Bitcells for Energy-Efficient Vector Processing”, DATE 2020

Specifications

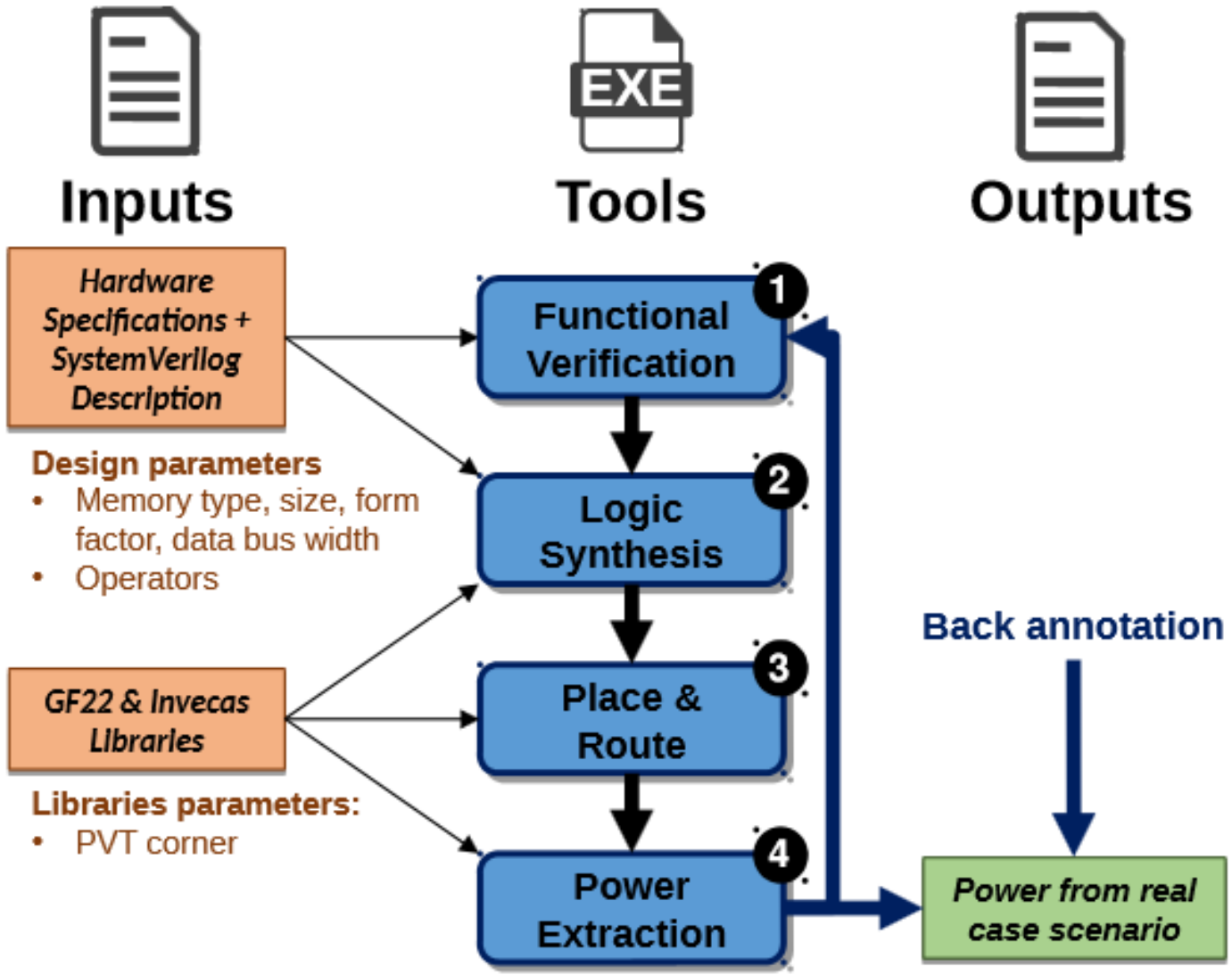
- **To be generic, we need 8, 16 and 32 bits operators**
- **Logical operators**
  - AND, OR, XOR, NOT and their complement → 8 instructions
- **Advanced logical operators**
  - Shift Right, Shift Left and Broadcast → $3 \text{ sizes} \times 3 \text{ operations} = 9 \text{ instructions}$
- **Arithmetic operators**
  - Addition, Subtraction, Comparison and Shift Right Arithmetic
    → $3 \text{ sizes} \times (3 \text{ operations} + 6 \text{ comparisons}) = 27 \text{ instructions}$
- **Advanced arithmetic operators (8 bits only)**
  - Multiplication and Multiply-And-Accumulate (MAC)
    → $3 \text{ multiplications for different signedness and } 1 \text{ MAC} = 4 \text{ instructions}$
- **Other operators**
  - Horizontal swap = 2 instructions

Proposed design





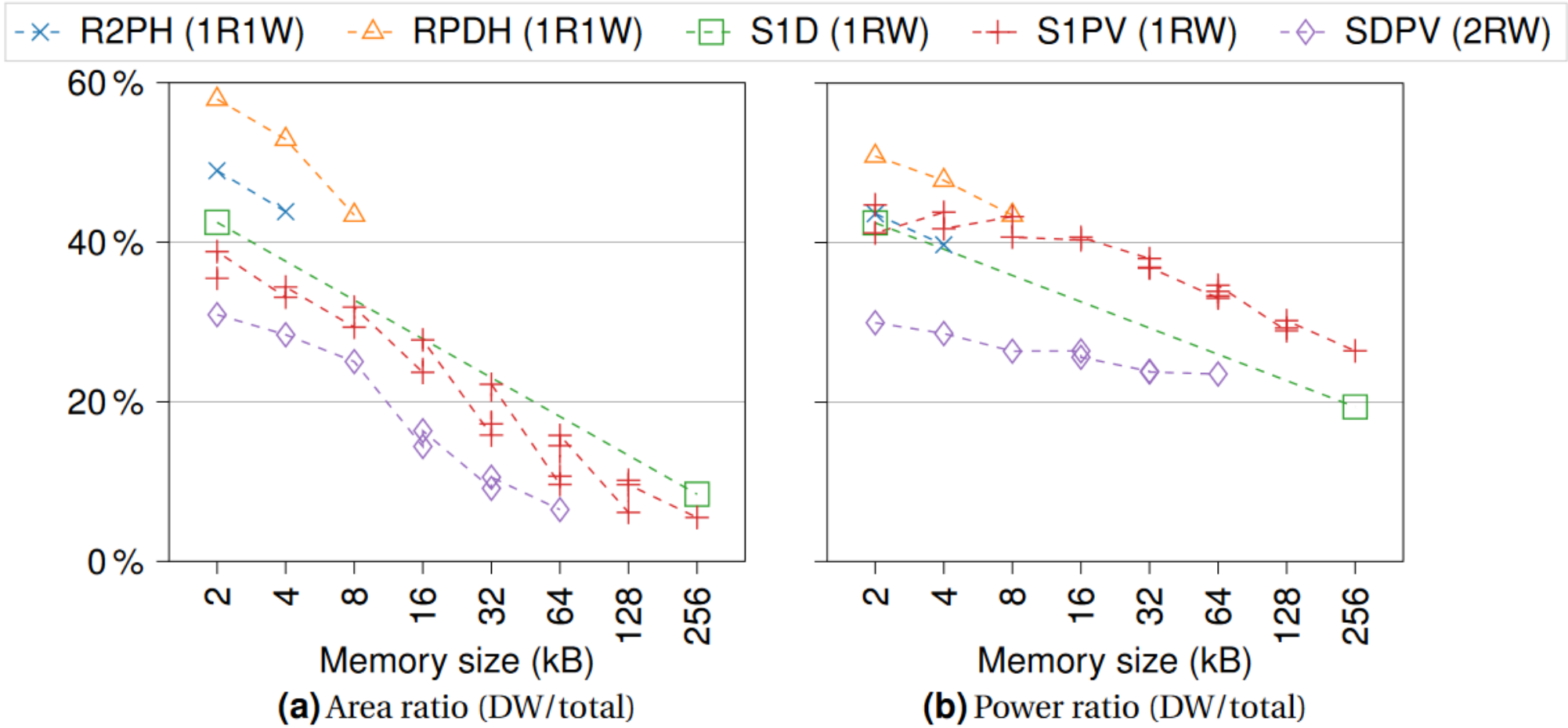
Results extraction: GF22 & Invecas



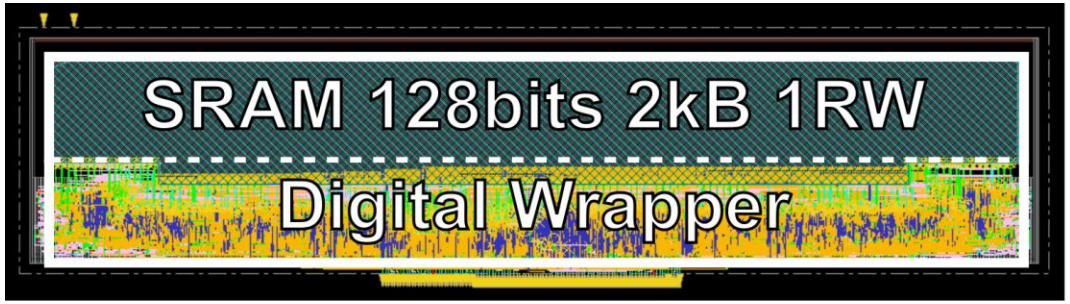
	Tool	Vendor	Version
1	ModelSim	MentorGraphics	10.5c
2	DesignCompiler	Synopsys	n-2017.09-sp5
3	Innovus	Cadence	19.13
4	Star-RC	Synopsys	o-2018.06-sp2
	PrimeTime Power	Synopsys	o-2018.06-sp5

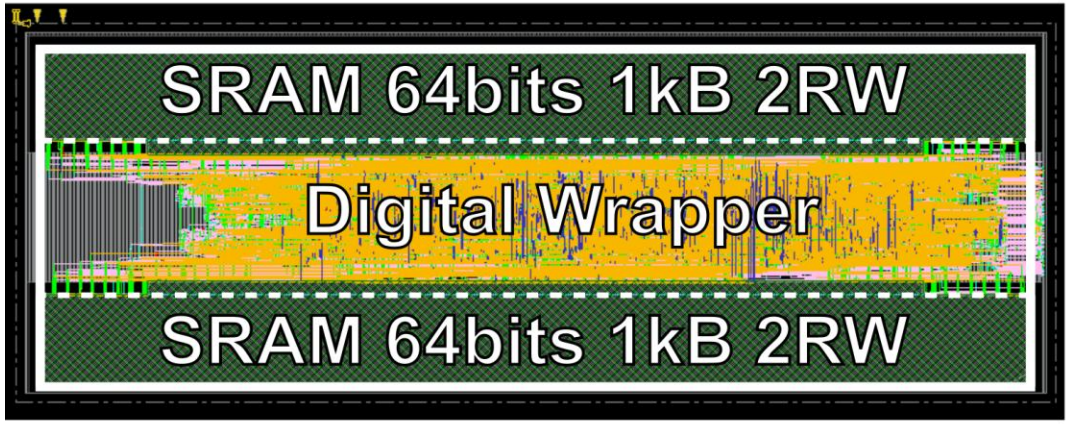
Results: Overhead

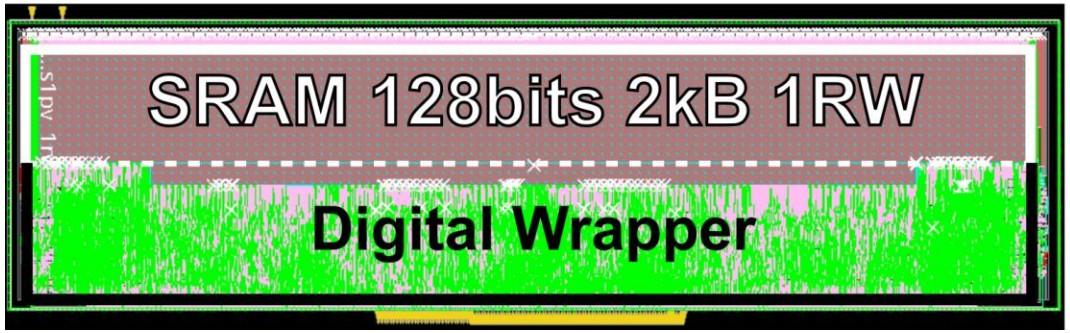
- Area overhead ranging from almost 60% for 2kB memories to 5% for >128kB memories
- Power overhead decreasing more gradually from 50% down to 20%



Results: Floorplan

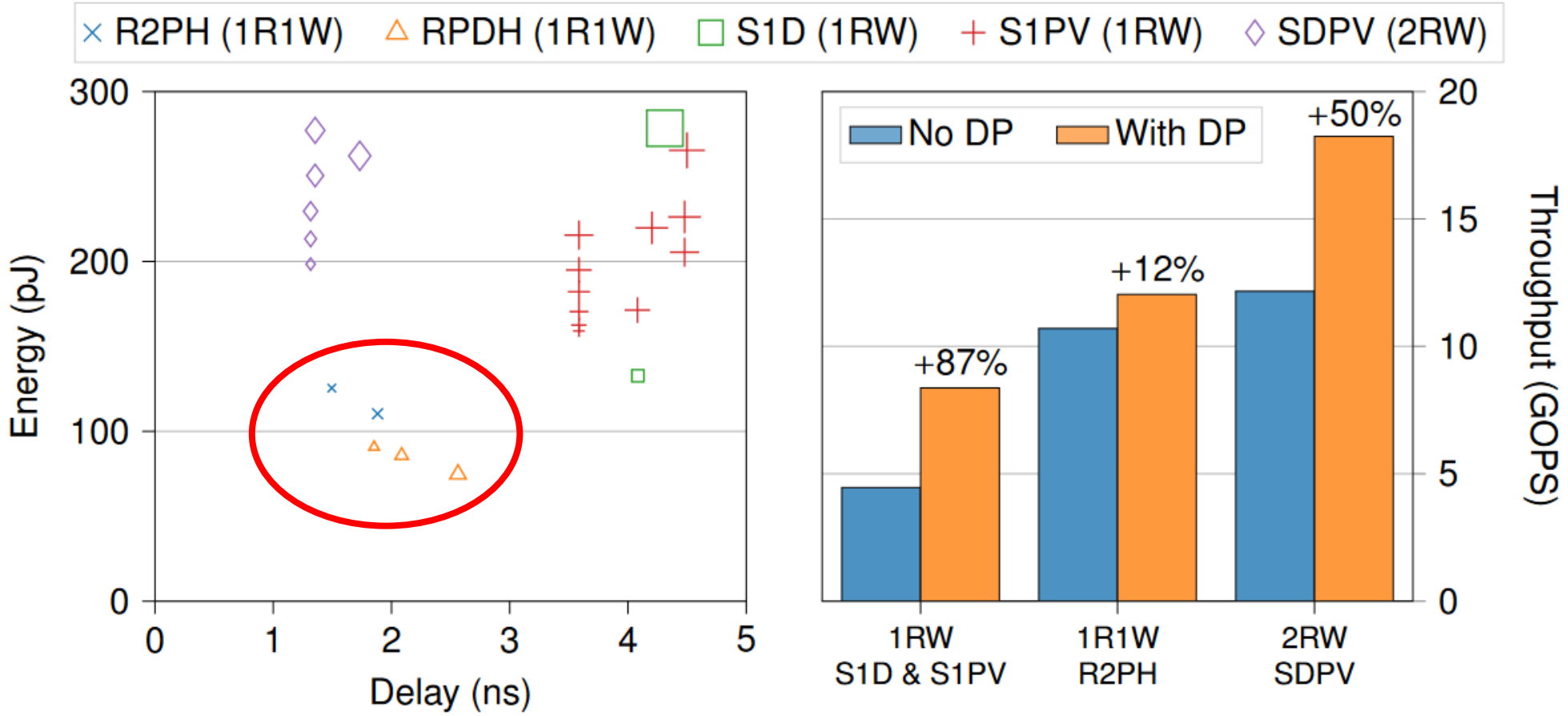


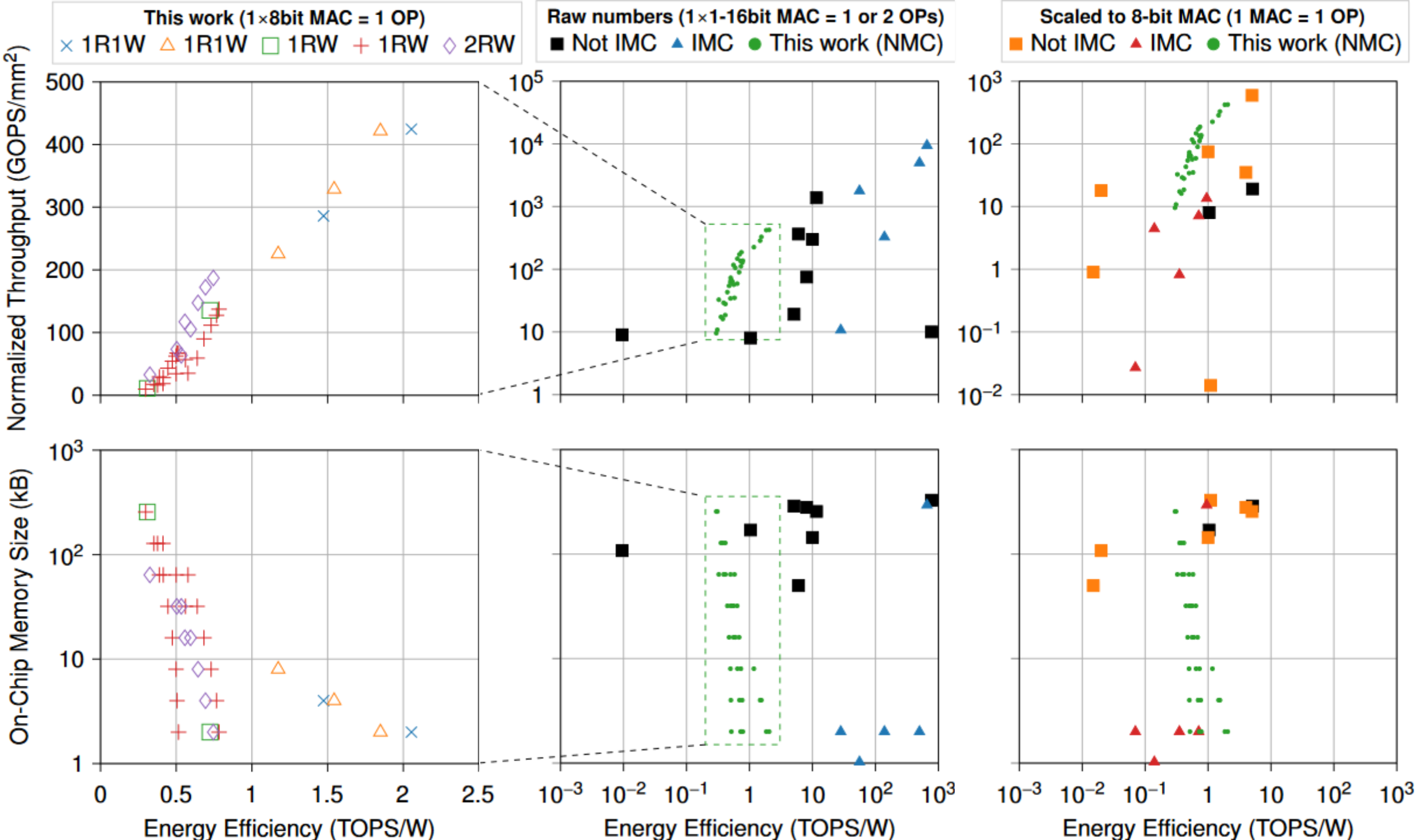




Results: 8-bit MAC instruction

- **Pareto front for optimal in either energy or delay**
  - 1R1W memories are the best choice but limited to small size (8kB max)
- **Doublepump grants +87% throughput for 1RW memories**




This work (1×8bit MAC = 1 OP)
1R1W
1R1W
1RW
1RW
2RW
Raw numbers (1×1-16bit MAC = 1 or 2 OPs)
Not IMC
IMC
This work (NMC)
Scaled to 8-bit MAC (1 MAC = 1 OP)
Not IMC
IMC
This work (NMC)
Normalized Throughput (GOPS/mm²)
On-Chip Memory Size (kB)
Energy Efficiency (TOPS/W)
Energy Efficiency (TOPS/W)
Energy Efficiency (TOPS/W)

Conclusion

- **SRAM based with automated approach**
  - Fast and versatile hardware development
  - Standard pipeline design
  - Acceptable area and power overhead
- **Better than analog IMC implementation**
  - More throughput and efficiency on 8-bit MAC
  - More memory size
- **This work has been published in DATE 2020**

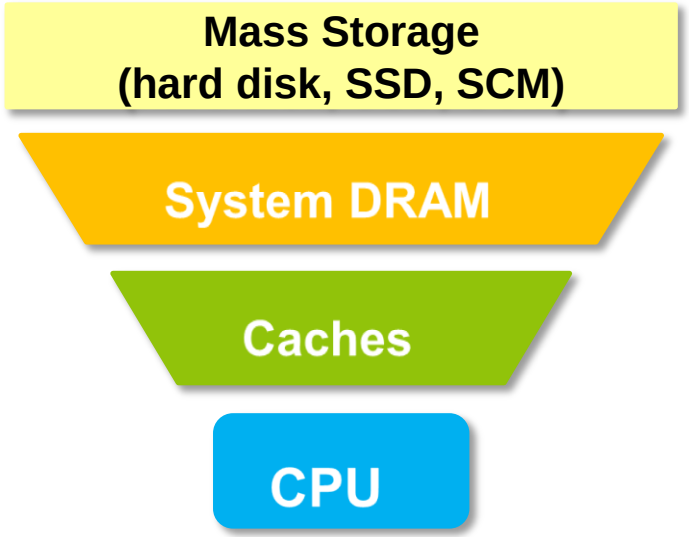
Outline

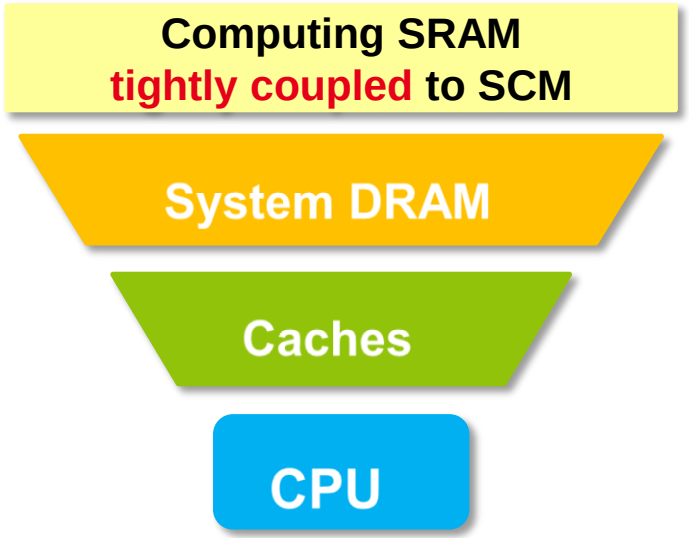
- General context
- State of the art on different technologies and implementations
- CSRAM: our own design approach
- **Software platforms and explorations**
  - Architecture evaluation
  - Memory access cost evaluation
  - Workflow
- Results and analysis on different memory computing architectures
- Conclusion and future works

How to evaluate full architecture?

- **Need to monitor all memories accesses**
  - Analytic models cannot describe exchange CPU ↔ CSRAM
  - Hardware counter not reliable for L3, DRAM and above
  - Use existing simulators such as gem5 or QEMU

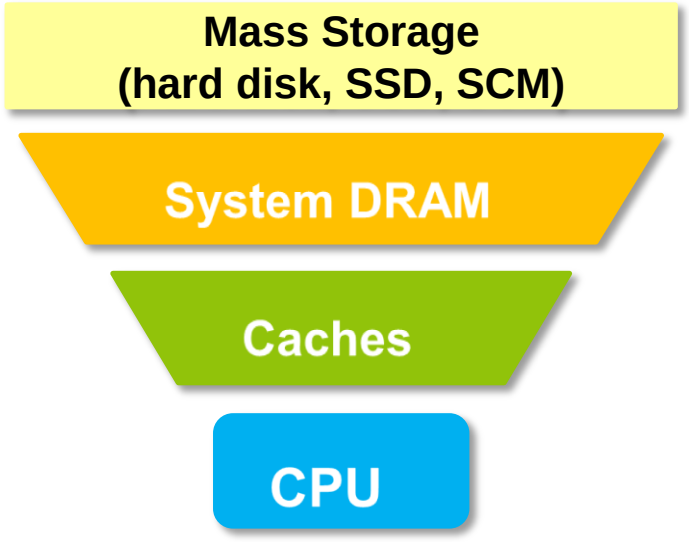
Platform	Memory interfaces	Accuracy	Speed	Development effort
QEMU	Callback functions	Instruction	High	High
gem5	Ruby	Cycle	Slow	High
ArchSim	TLM sockets	Transaction	Medium	Moderate
ZSim	None	Instruction	Very high	Moderate
Sniper	N/A	Cycle	Slow	Moderate
LLVM	None	Instruction	Medium	Low
Pin	Callback functions	Instruction	High	Low

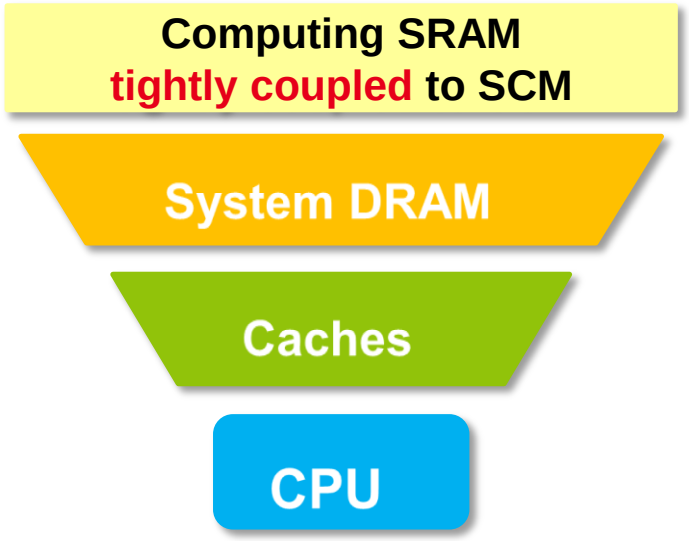




How to evaluate each memory access cost?

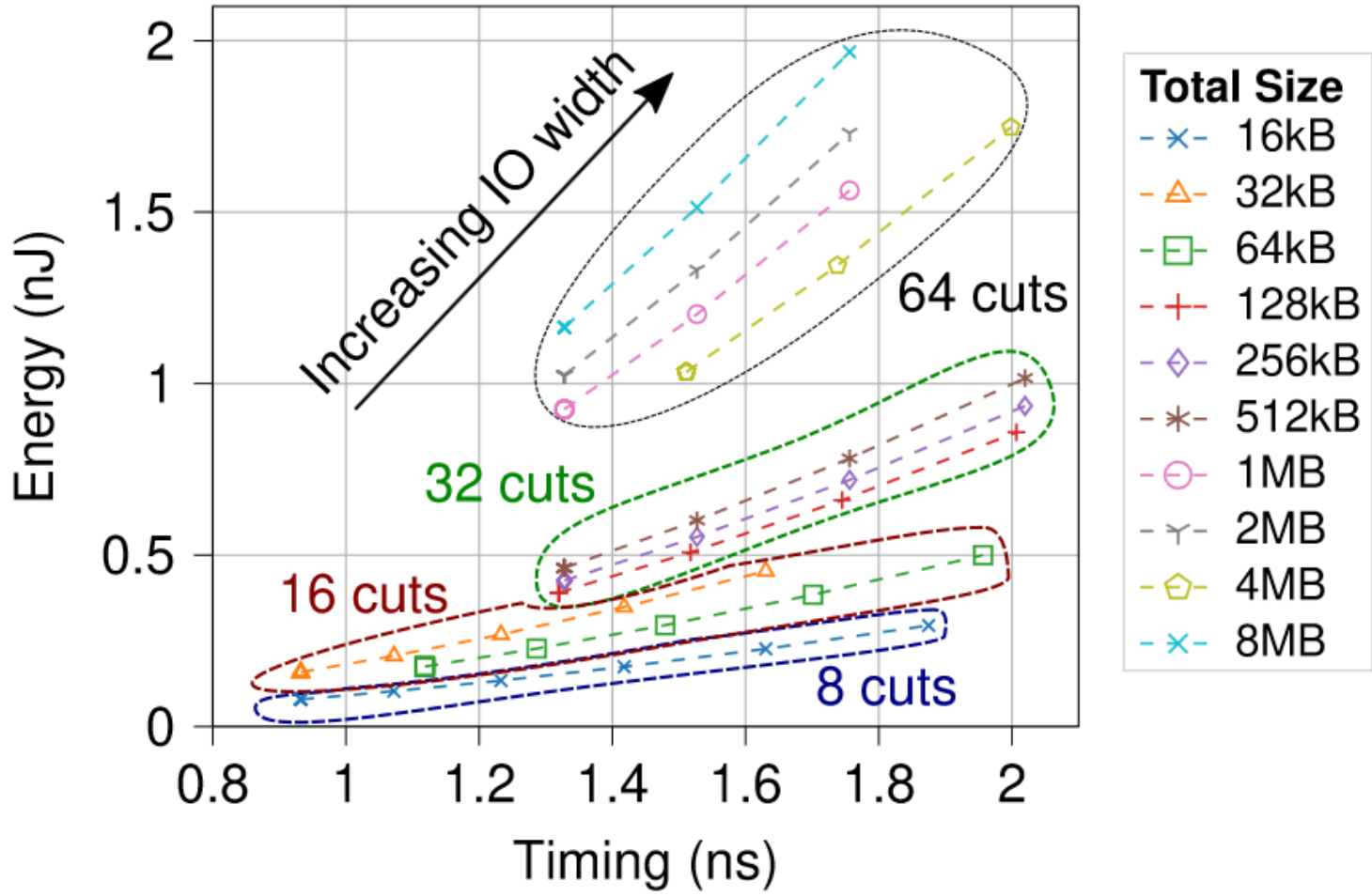
- **Use specialized hardware model tools**
  - NVSim (based on cacti) for caches and NVM
    - PCM based SCM with technology parameters from state of the art
  - DRAM evaluation is tricky
    - Calibration done with VAMPIRE
  - CSRAM evaluation made in-house





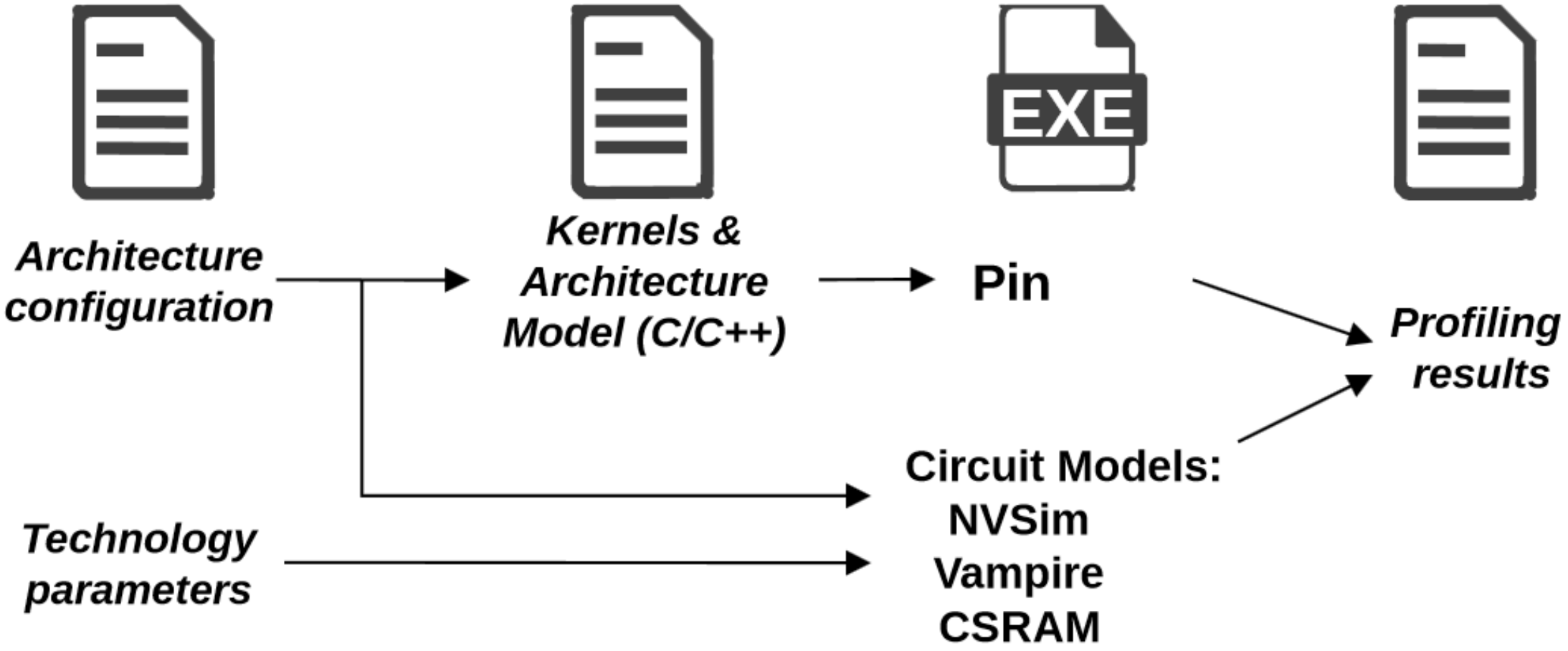
CSRAM tiling

- L3 cache like size to contain working dataset
- Tiling and widening 1RW memories
  - From 16B wide IO up to 4kB
  - From 2kB to 256kB total size up to 8MB



R. Gauchi, “Exploration of Reconfigurable Tiles of Computing-in-Memory Architecture for Data-intensive Applications”, 2021

Final workflow

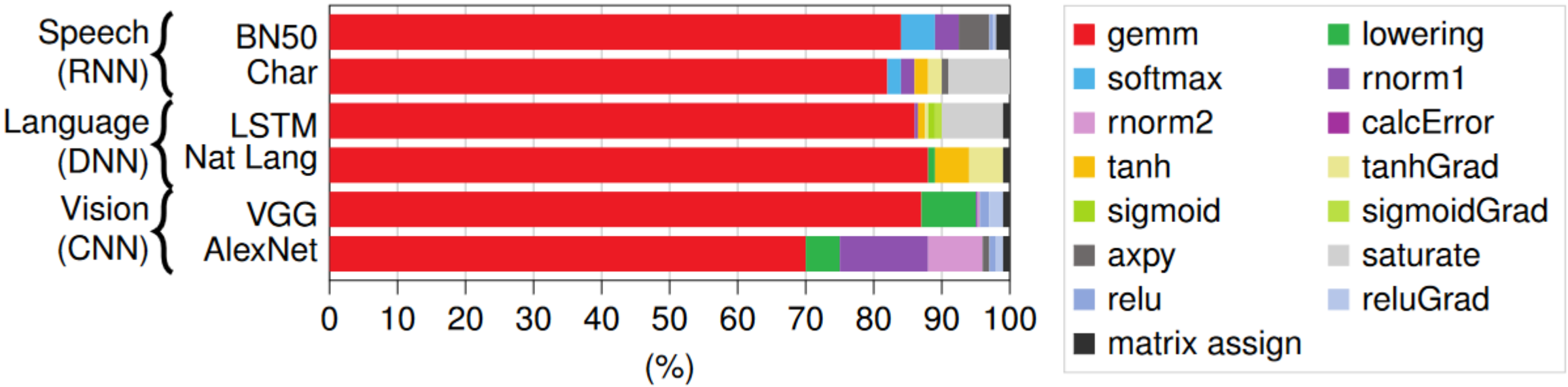


Outline

- General context
- State of the art on different technologies and implementations
- CSRAM: our own design approach
- Software platforms and explorations
- **Results and analysis on different memory computing architectures**
  - Benchmarks used
  - Computing at the top
- Conclusion and future works

Used benchmarks

- **Big data era with neural networks and sensors everywhere**
  - Neural networks mostly use matrix multiplication
  - Convolution can be turned into matrix multiplication through *matrix assign* operation



B. Fleischer and S. Shukla, “Approximate Computing for On-Chip AI Acceleration”, 2018

Used benchmarks

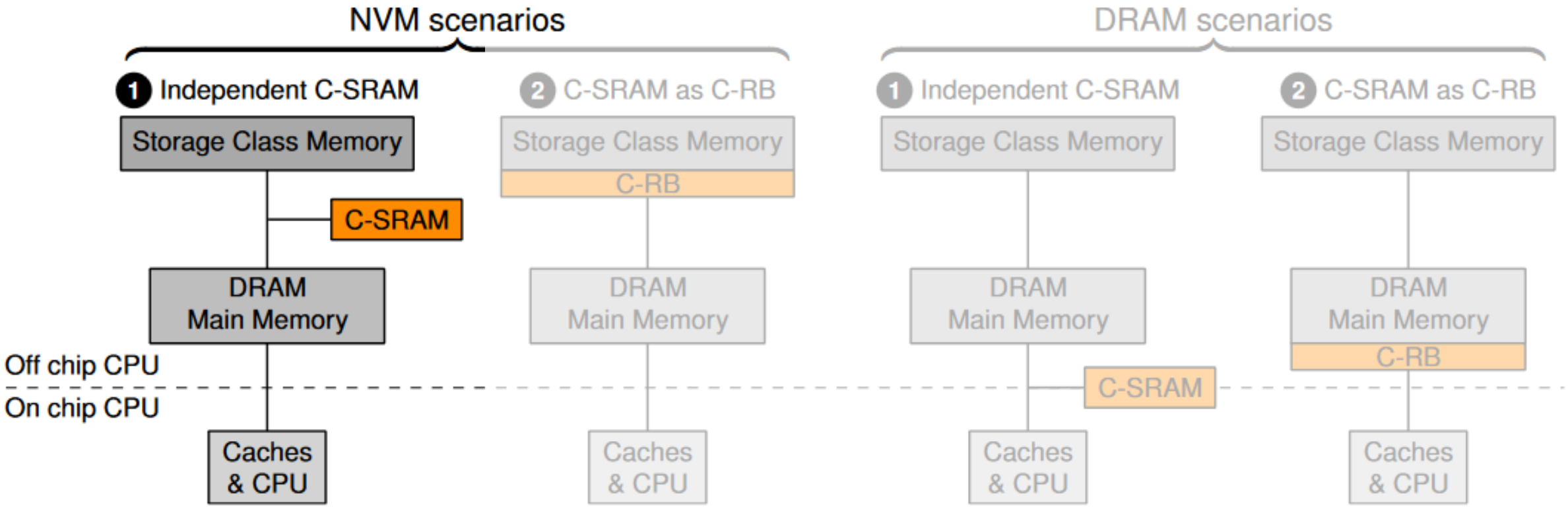
- **Linear kernels**
  - Hamming weights → information theory and also BNN with popcnt operation
  - shift-or → pattern matching in DNA sequence
  - AXPY ($y = \alpha x + y$) → Basic linear algebra, very high disk access
- **Quadratic kernels**
  - atax ($y = A^T(Ax)$) → linear solver and signal processing
  - gesummv ($y = \alpha Ax + \beta Bx$) → signal processing
- **Cubic kernels**
  - gemm ($C = \alpha AB + \beta C$) → matrix multiplication, image processing, neural networks
- **Real case application**
  - darknet → implementation of CNN for image classification

Compute bound

Memory bound

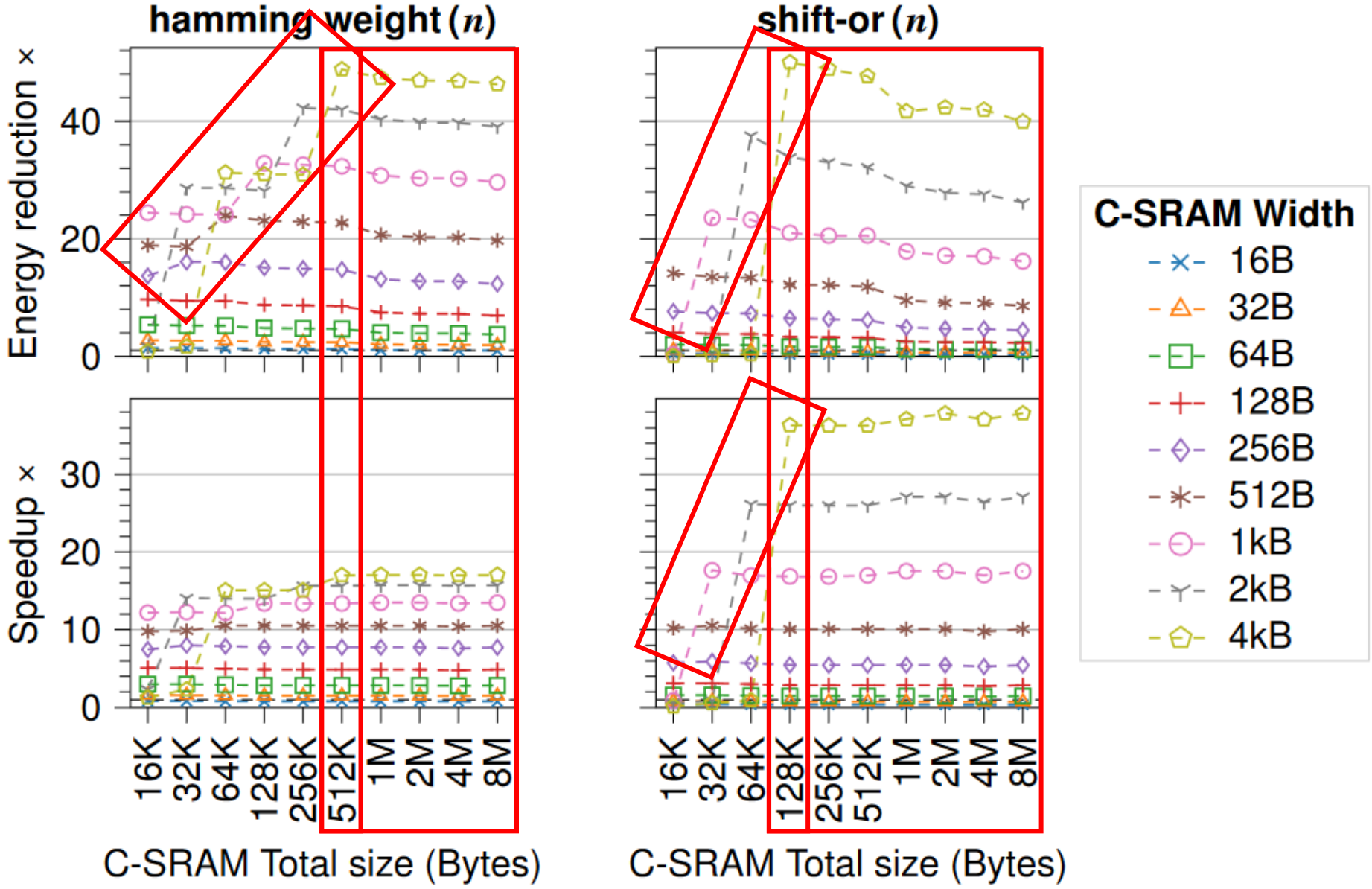
L-N. Pouchet and T. Yuki, "PolyBench/C – The polyhedral benchmark suite", 2015
J. Redmon, "Darknet: Open Source Neural Networks in C", 2013

Different integration possibility



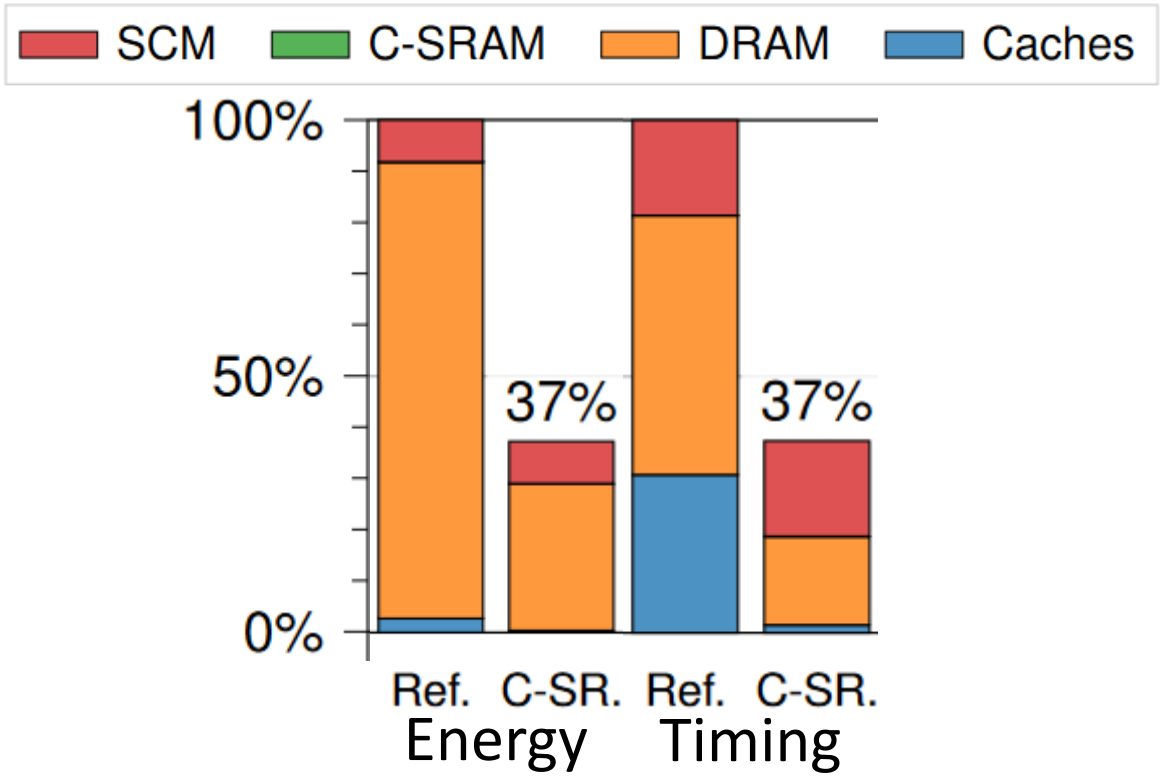
Linear kernels: hamming weight & shift-or

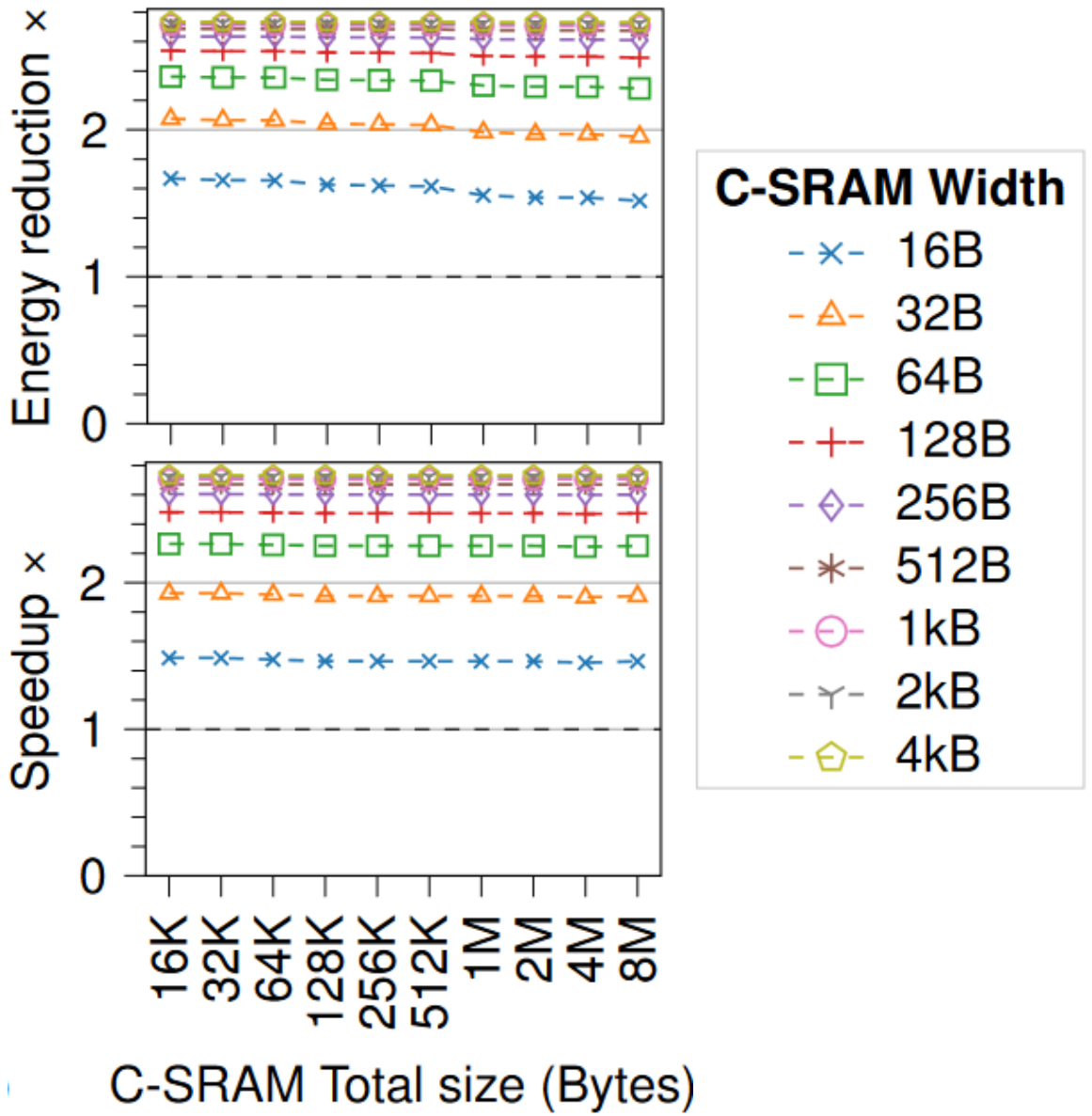
- Optimal size between 128kB and 512kB
- Larger vector width leads to more improvements
- Larger than optimal total size decreases energy reduction due to more costly accesses
- Stair like shape due to minimal working dataset size



Linear kernels: axpy

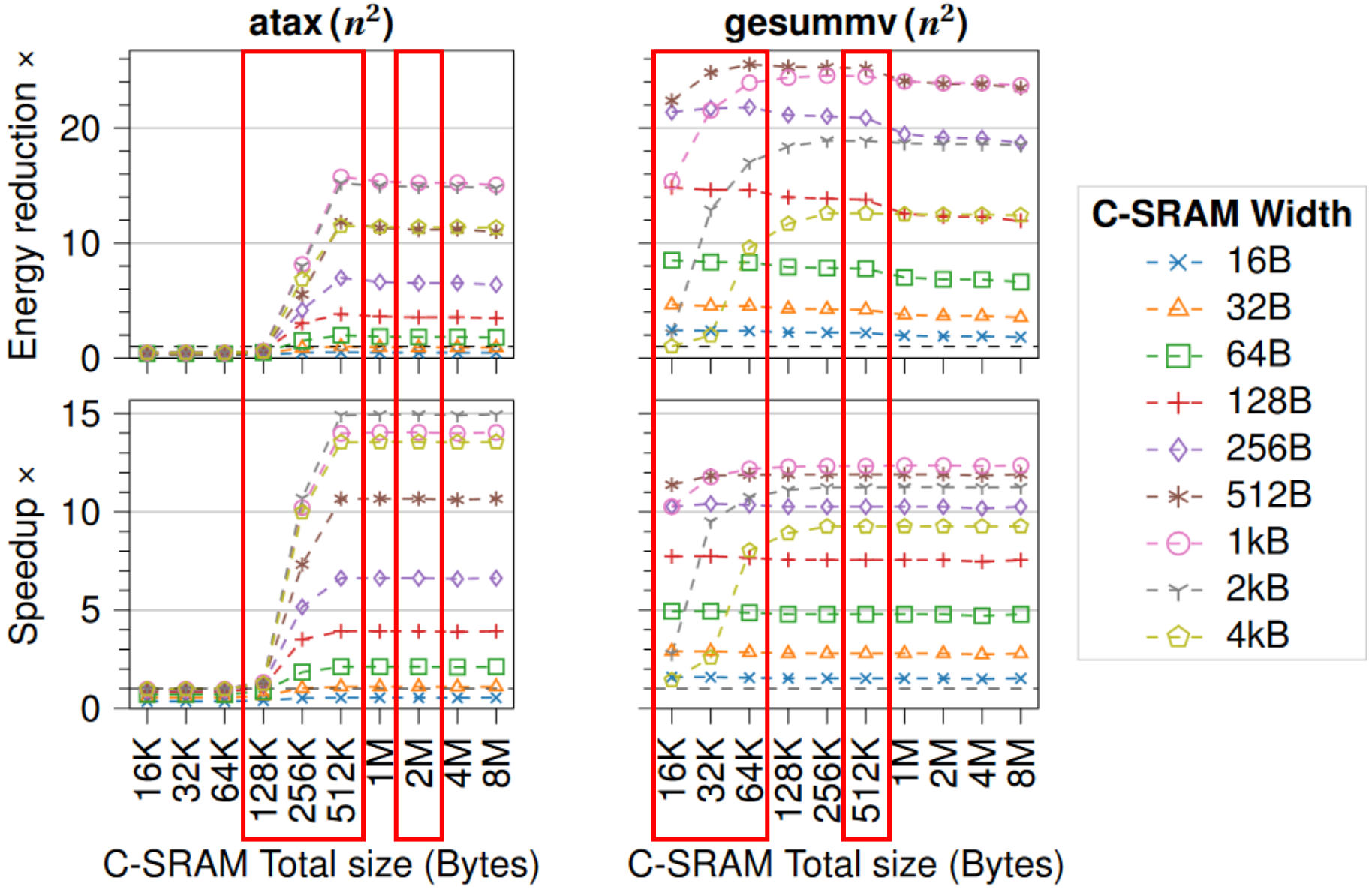
- **Total size has almost no impact**
  - Working dataset is just 4 data
- **Vector width shows very fast saturation**





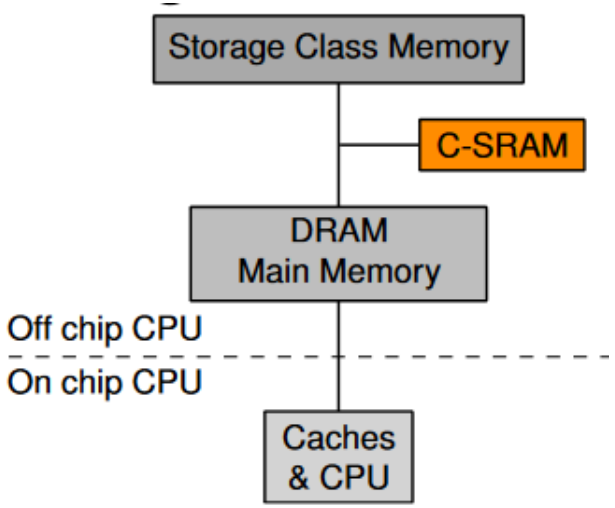
Quadratic kernels: atax & gesummv

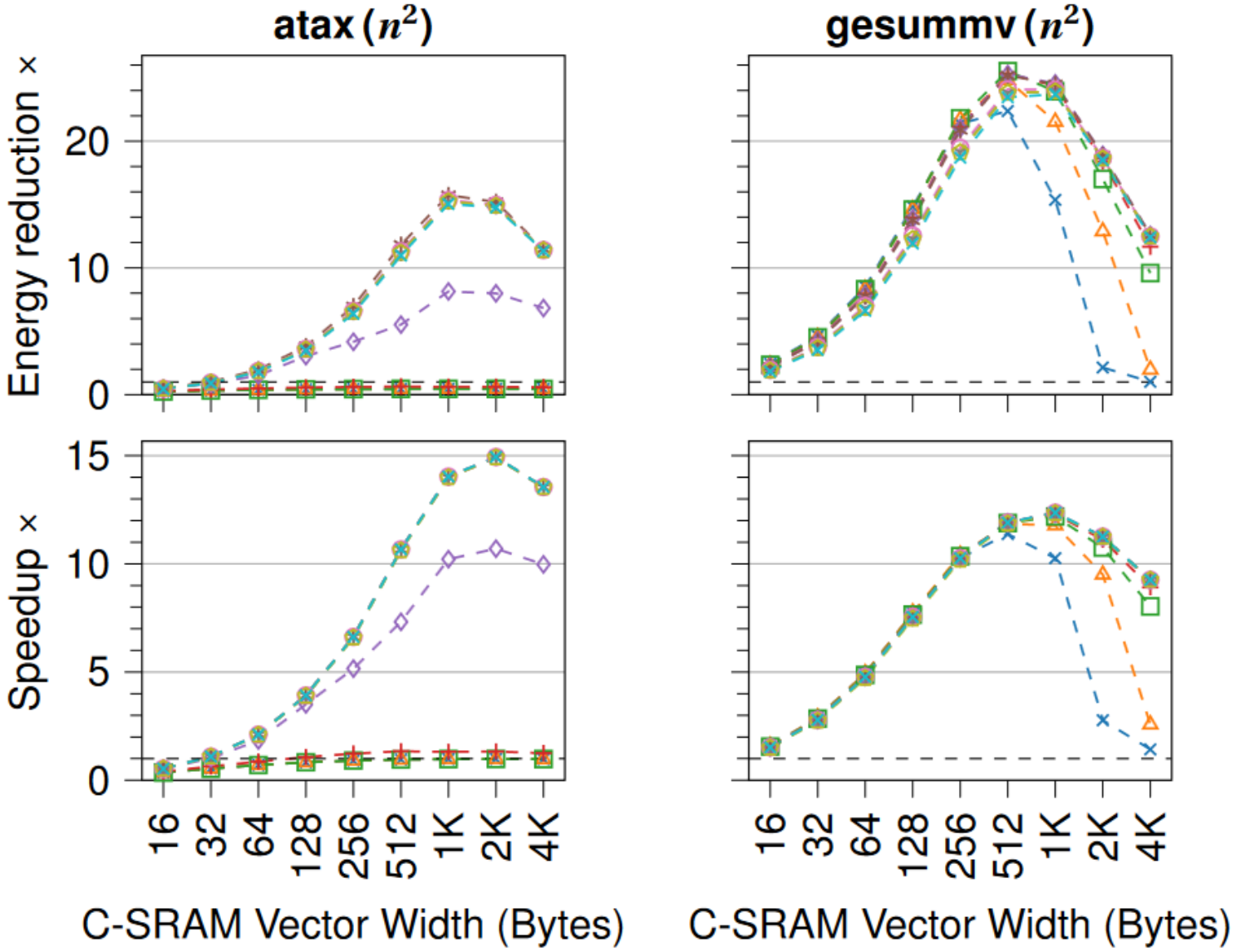
- Atax needs minimal total size of 512kB
- Gesummv benefits from small memories
- Both kernels, larger vector width has negative impact
  - Reduction loop performed by the CPU



Quadratic kernels: atax & gesummv

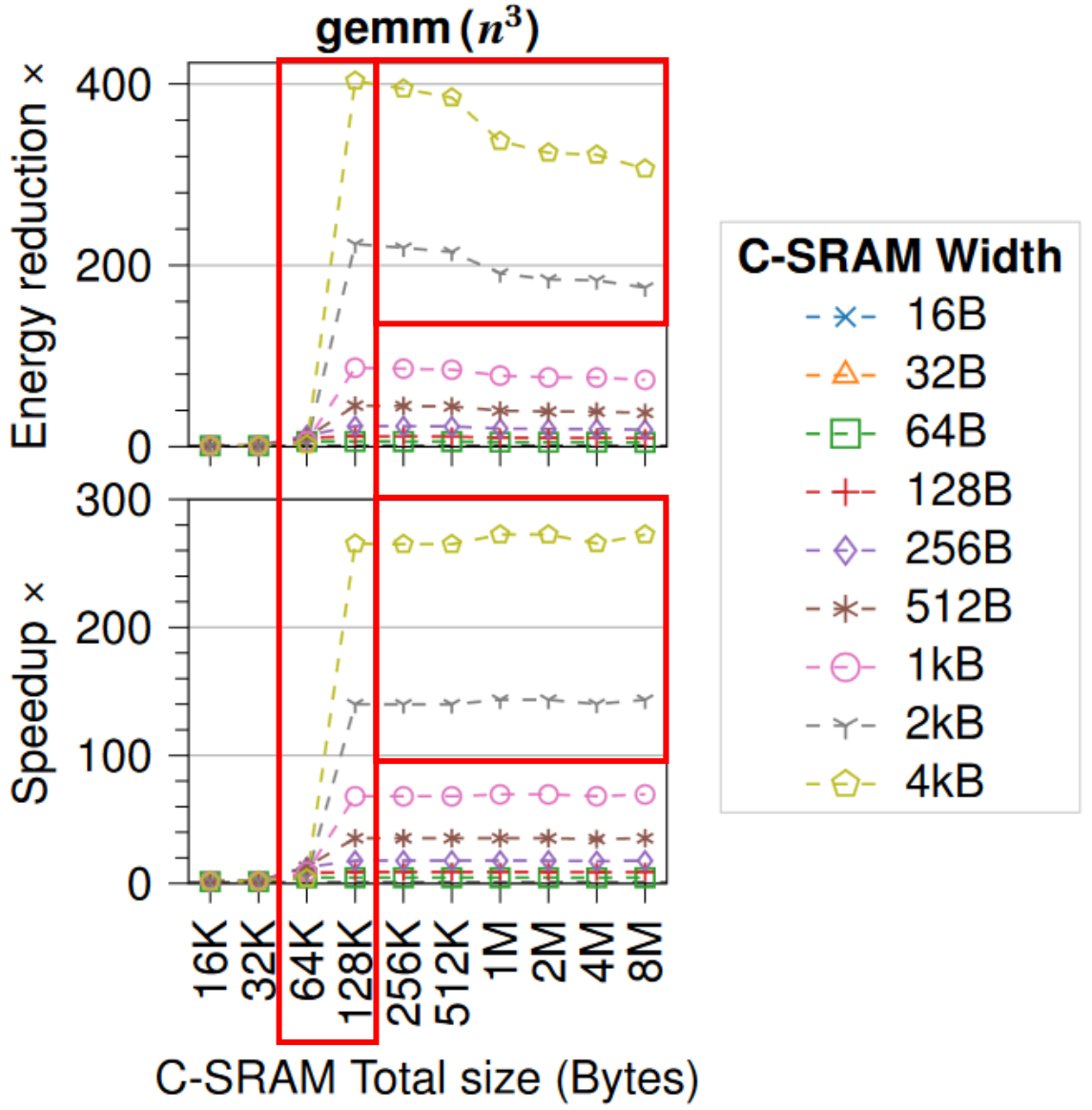
- Atax needs minimal total size of 512kB
- Gesummv benefits from small memories
- Both kernels, larger vector width has negative impact
  - Reduction loop performed by the CPU



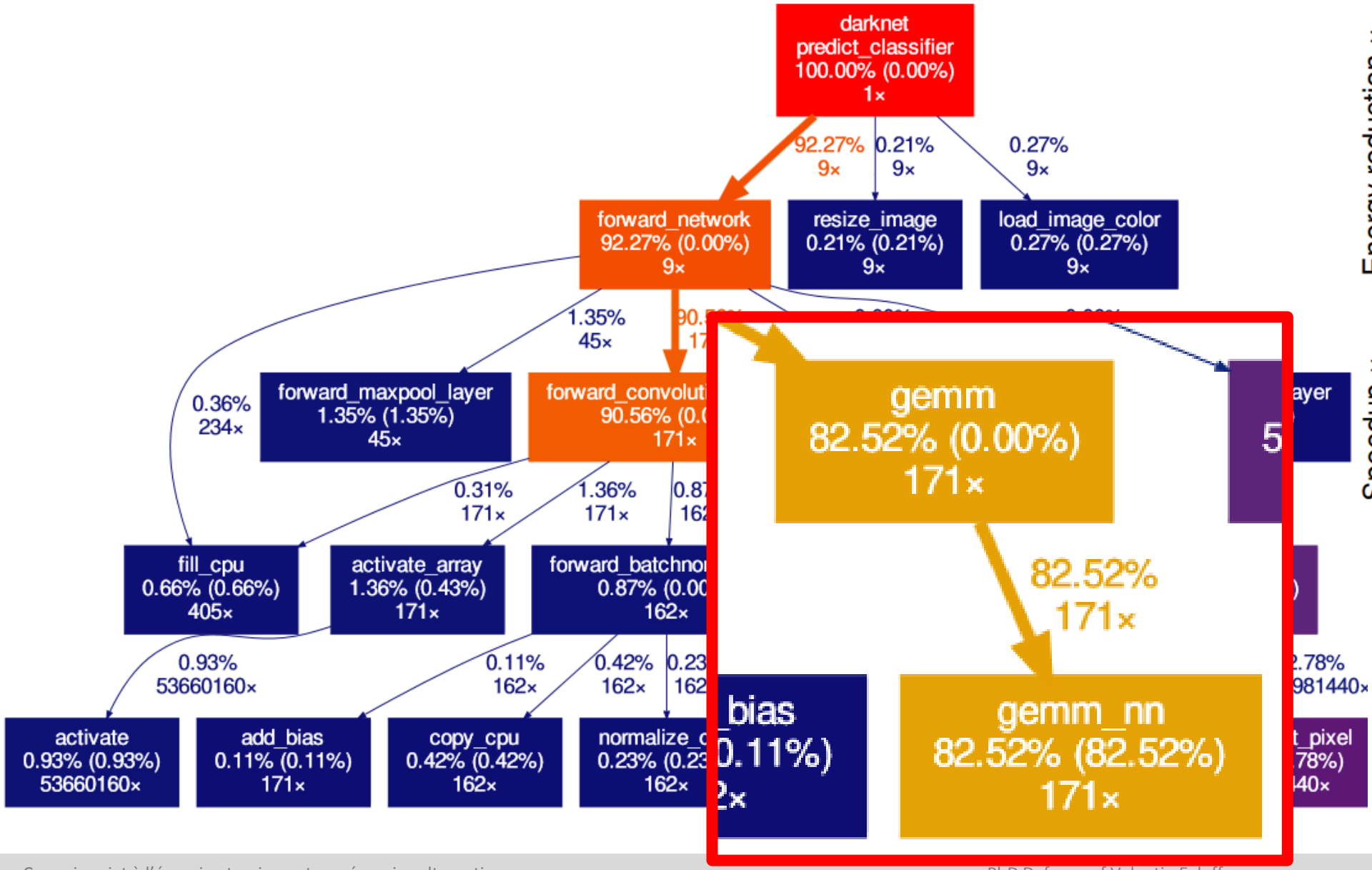


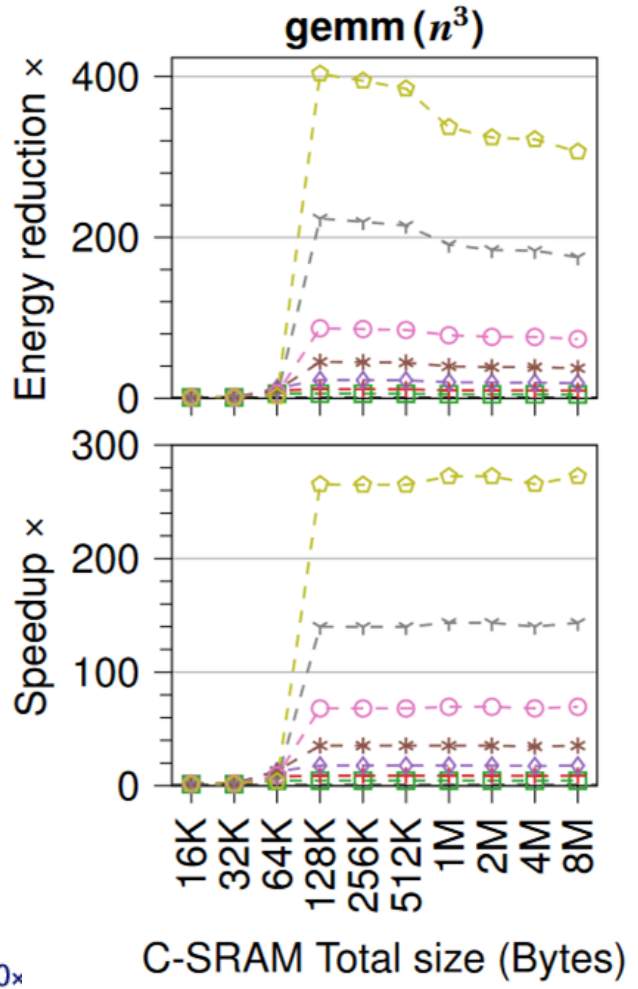
Cubic kernel: gemm

- Similar to atax, gemm requires minimal total size of 128kB
- More than optimal total size costs more similar as linear kernels
  - Timing unaffected due to less eviction
- Best case of 403x energy reduction and 271x speedup

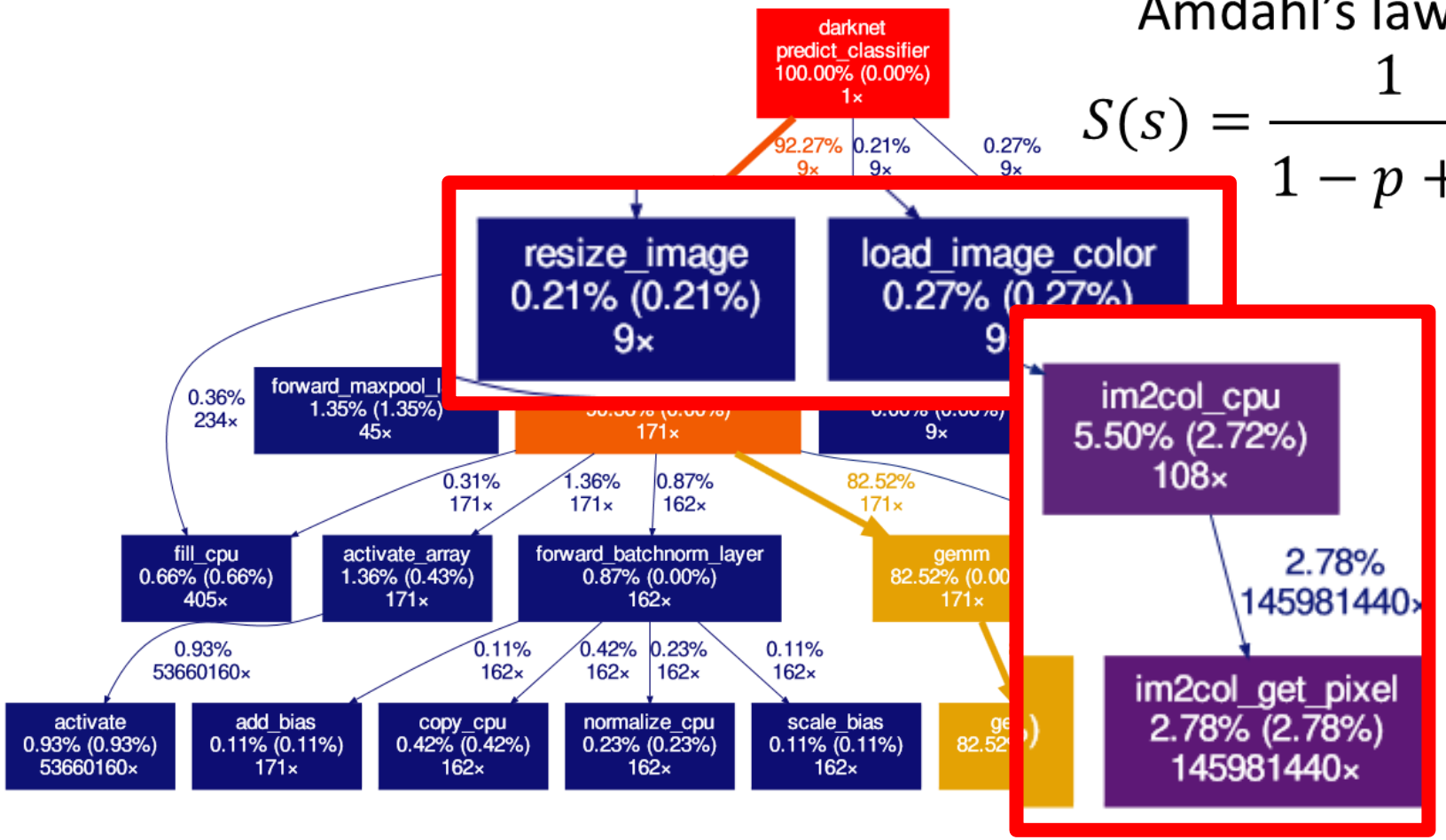


Real application case: Darknet



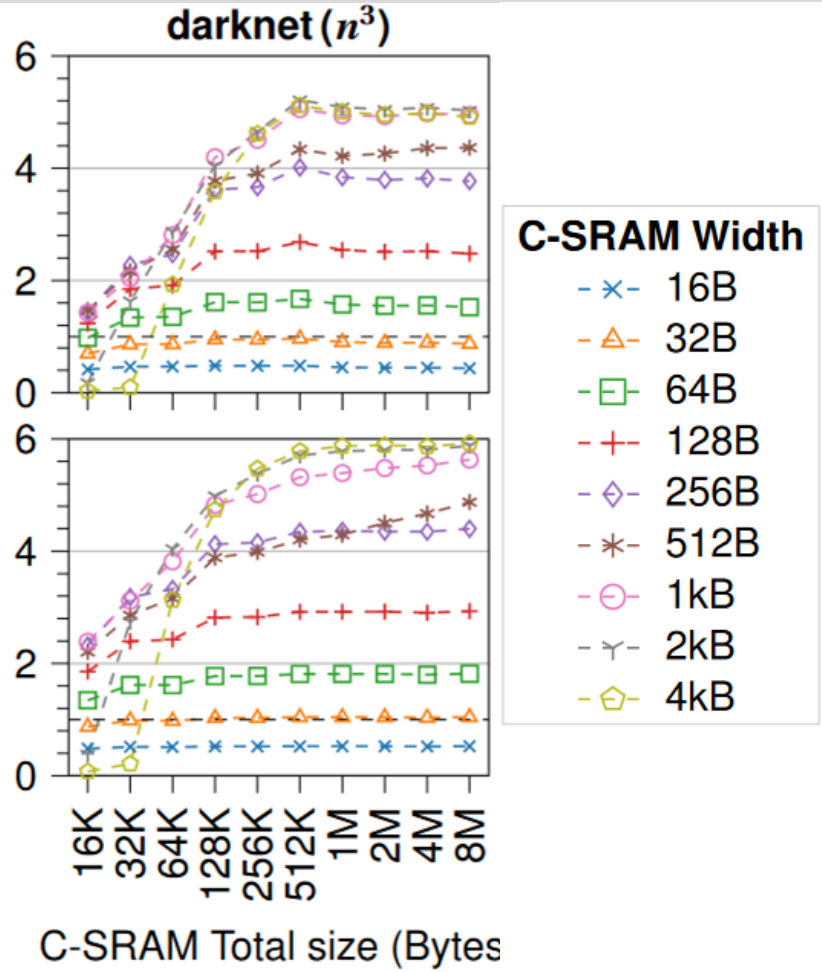


Real application case: Darknet



Amdahl's law

$$S(s) = \frac{1}{1 - p + \frac{p}{s}}$$



- Data are stored on disks but must be decoded (jpg, png, mp4, etc.) before applying algorithms
- Need to layout data for neural network to turn convolution into gemm

Conclusion

- **No more room for performance improvements**
  - **Technology** reaches an **end**
    - Energy wall & dark silicon
  - CPU **architecture** has been **pushed** to its **maximum**
    - Memory wall & von Neumann bottleneck
  - **Energy efficiency** of current solutions is quite **low**
- **Need for a more energy efficient and data oriented architecture**
  - Big data era : neural networks and social networks
- **→ In-Memory computing is promising solution**
  - **Tackle von Neumann bottleneck** at its source
  - **Computes** where data is and severely **reduces data movement**
- **SOTA is limited**
  - Analog computing
  - NVM endurance is often not discussed

Conclusion

- **CSRAM design**
  - **SRAM based** and **automated** design approach
  - **Competes** with state-of-the-art solutions
- **Results**
  - Moving **computing closer** to where **data** live seems to be the **best option**
    - **Computing** in the **SCM** will **kill** it in less than a **day**
    - **Data** are stored on **disks** but must be **decoded** (jpg, png, mp4, etc.) **before** applying **algorithms**
    - Need to **layout data** for **neural network** to turn **convolution into gemm**
  - **DRAM** is not yet evinced as it serves as a **write buffer**
  - **Cache** hierarchy can be reduced but is still **used** for **instructions, stack**
- **Average 17.4x energy reduction and 12.9x speedup**
  - **Optimal** memory size around **512kB**

Perspectives and future works

- **Future works**
  - **CPU** should be used for **parallel computing**
  - **Compare** against other **architectures** such as **GPUs**
  - **Silicon implementation**
- **Perspectives**
  - **Take** all **application** into **account**, **not** just **kernel**
  - **Watch out** for **rebound** effect

Publications & Patents

Publications

- “Shuffle operator for matrix multiplication in in-memory computing architecture”, **Author**, Poster in COMPAS, 06/19, and also in ACACES HiPEAC, 07/19
- “Computational SRAM Design Automation using Pushed-Rule Bitcells for Energy-Efficient Vector Processing”, **2nd author**, DATE 2020 (special session)
- “Reconfigurable Tiles of Computing-In-Memory SRAM Architecture for Scalable Vectorization”, **2nd author**, ISLPED 2020
- “Storage Class Memory with Computing Row Buffer: A Design Space Exploration”, **author**, DATE 2021

Patents

- “Outil de selection automatique de mémoires pour circuits de type *in-* ou *near-memory computing*”, brevet **co-inventeur**, 2020
- “Tampon de lignes mémoire volatile calculant pour mémoires non volatiles ”, **inventeur**, 2021

Thank you for your attention

Backup

Dark Silicon illustration: DVFS adaptation

- Frequency scaling of an Intel Xeon Silver 4116, a 12 cores chip, function of active cores and active SIMD extension.

Mode	Base	Turbo Frequency/Active Cores											
		1	2	3	4	5	6	7	8	9	10	11	12
Normal	2.1 GHz	3.0 GHz	3.0 GHz	2.8 GHz	2.8 GHz	2.7 GHz	2.7 GHz	2.7 GHz	2.7 GHz	2.4 GHz	2.4 GHz	2.4 GHz	2.4 GHz
AVX2	1.7 GHz	2.9 GHz	2.9 GHz	2.7 GHz	2.7 GHz	2.4 GHz	2.4 GHz	2.4 GHz	2.4 GHz	2.1 GHz	2.1 GHz	2.1 GHz	2.1 GHz
AVX512	1.1 GHz	1.8 GHz	1.8 GHz	1.6 GHz	1.6 GHz	1.5 GHz	1.5 GHz	1.5 GHz	1.5 GHz	1.4 GHz	1.4 GHz	1.4 GHz	1.4 GHz

Xeon Silver 4116 - Intel - WikiChip. URL: https://en.wikichip.org/wiki/intel/xeon_silver/4116

Instruction energy breakdown

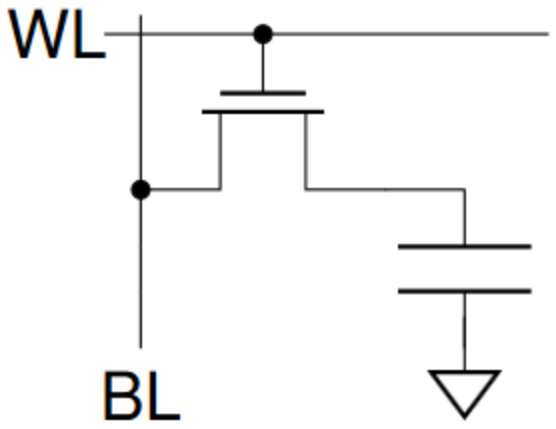
Integer		Floating Point		Memory	
Addition				Cache	64 bit access
8 bit	0.03pJ	16 bit	0.4pJ	8kB	10pJ
32 bit	0.1pJ	32 bit	0.9pJ	32kB	20pJ
Multiplication				1MB	100pJ
8 bit	0.2pJ	16 bit	1.1pJ	DRAM	1.3-2.6nJ
32 bit	3.1pJ	32 bit	3.7pJ		

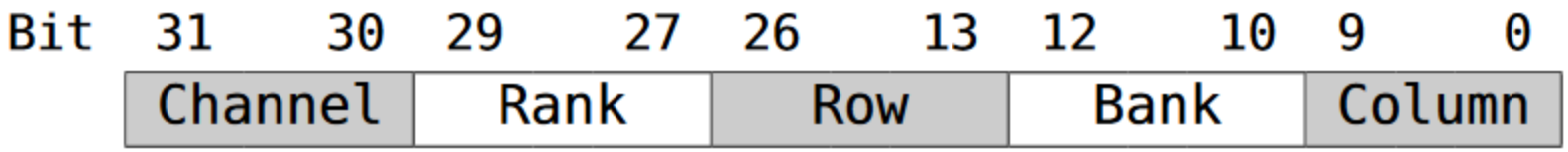
M. Horowitz. “1.1 Computing’s energy problem (and what we can do about it)” In: 2014 ISSCC. DOI: 10.1109/ISSCC.2014.6757323

Emerging non volatile memories and storage class memories

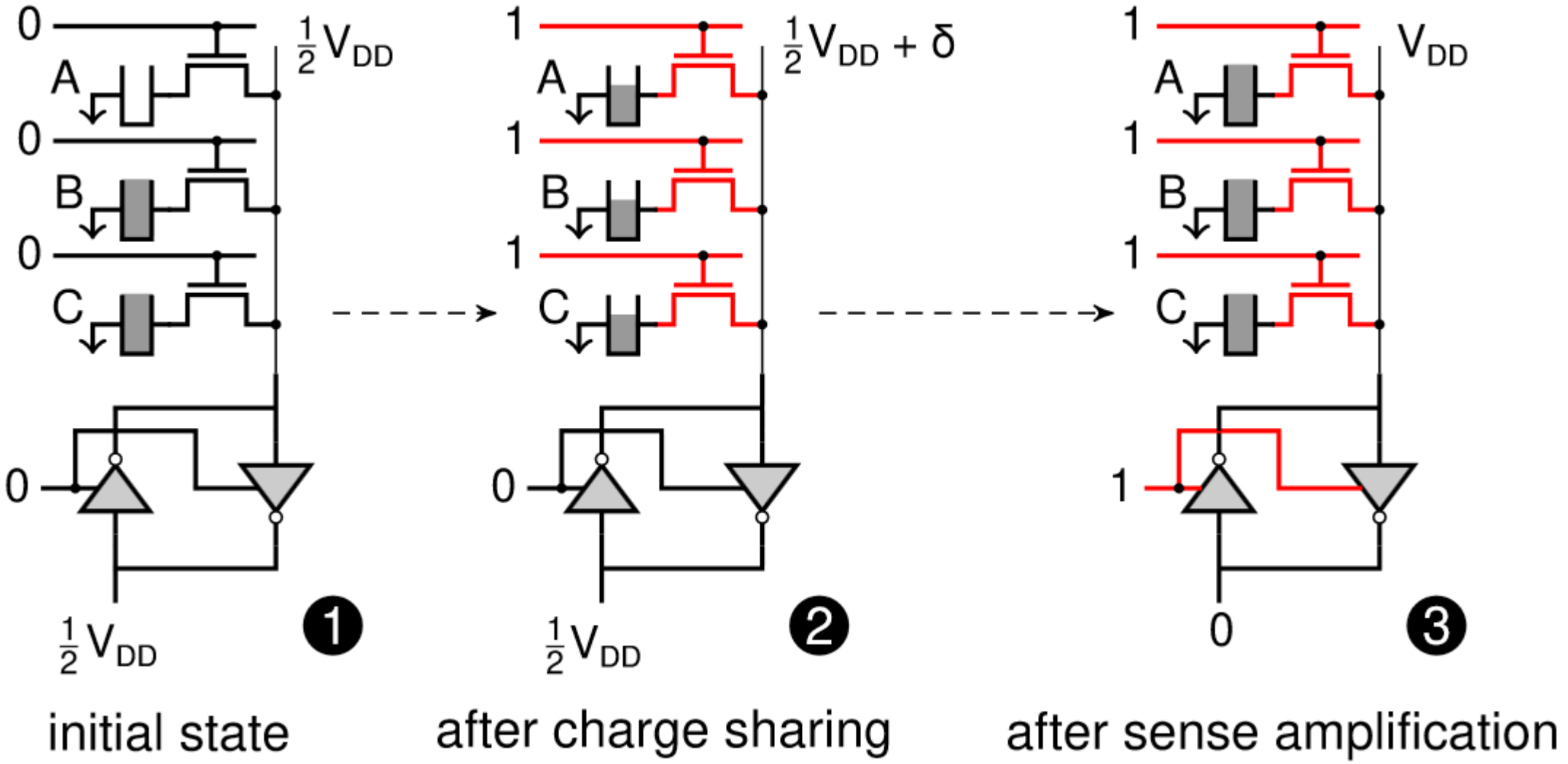
	DRAM	SRAM	NAND Flash	STT-RAM	PCM	RRAM
Cell Size	6 F^2	120–200 F^2	<4 F^2 (3D)	6–50 F^2	4–30 F^2	4–12 F^2
Multibit	1	1	3	2	4	2
Read latency	10–50 ns	0.2–2 ns	15–35 µs	2–35 ns	10–60 ns	5–20 ns
Write latency	10–50 ns	0.2–2 ns	100–1000 µs	3–50 ns	20–150 ns	10–50 ns
Write energy (/bit)	10 fJ	1 fJ	10 nJ	10 fJ–1 pJ	10–500 pJ	0.1–10 pJ
Endurance	∞	∞	1e4–1e5	1e12–1e15	1e7–1e10	1e6–1e12
Cons	Read destructive + Refresh	High leakage	Erase block (4kB) before write	Process incompati-bility	Heat transfer + high current	Sneak current + IR drop
Maturity	Mature	Mature	Mature	Testchips	Com.	Testchips

DRAM bitcell and addressing scheme



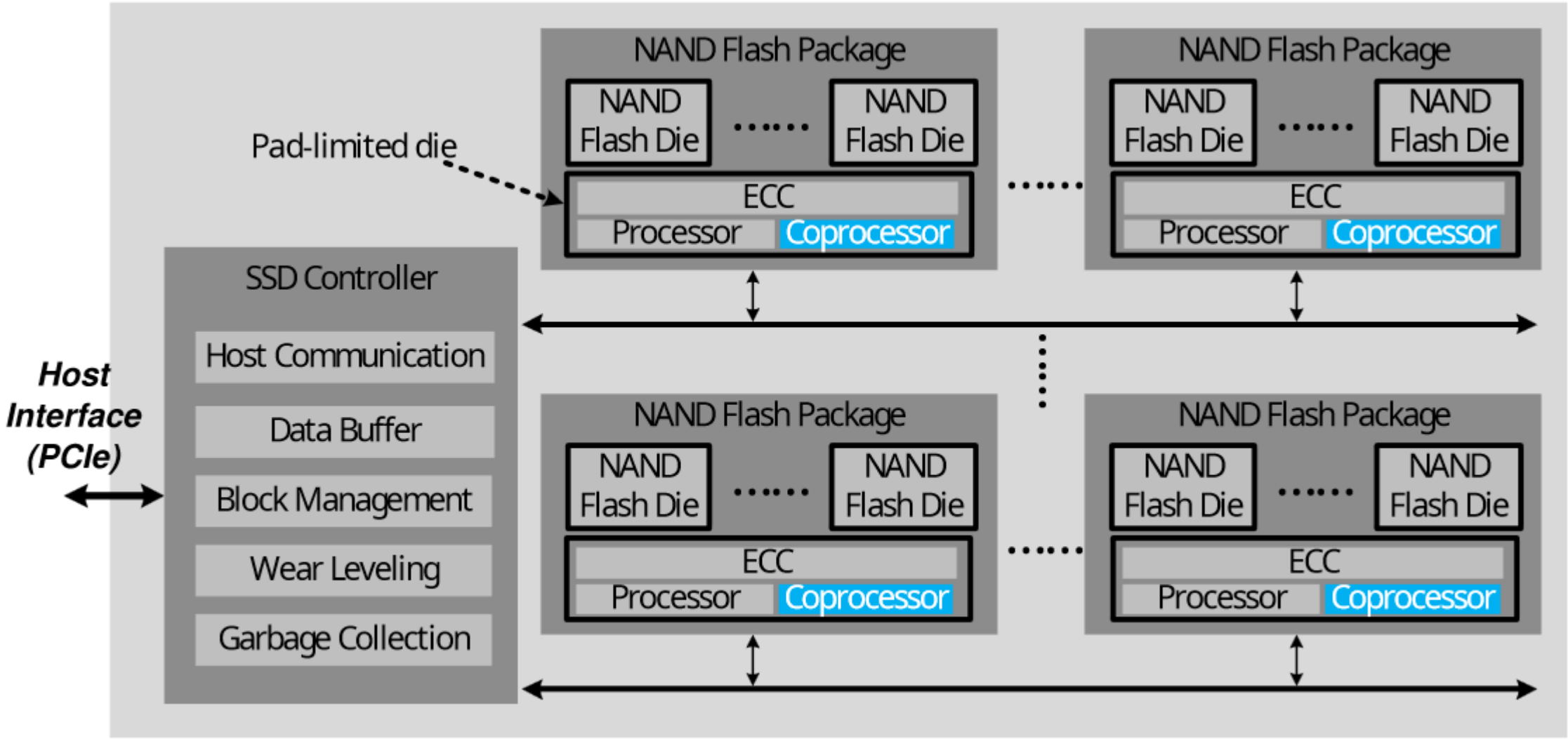


DRAM IMC with charge sharing



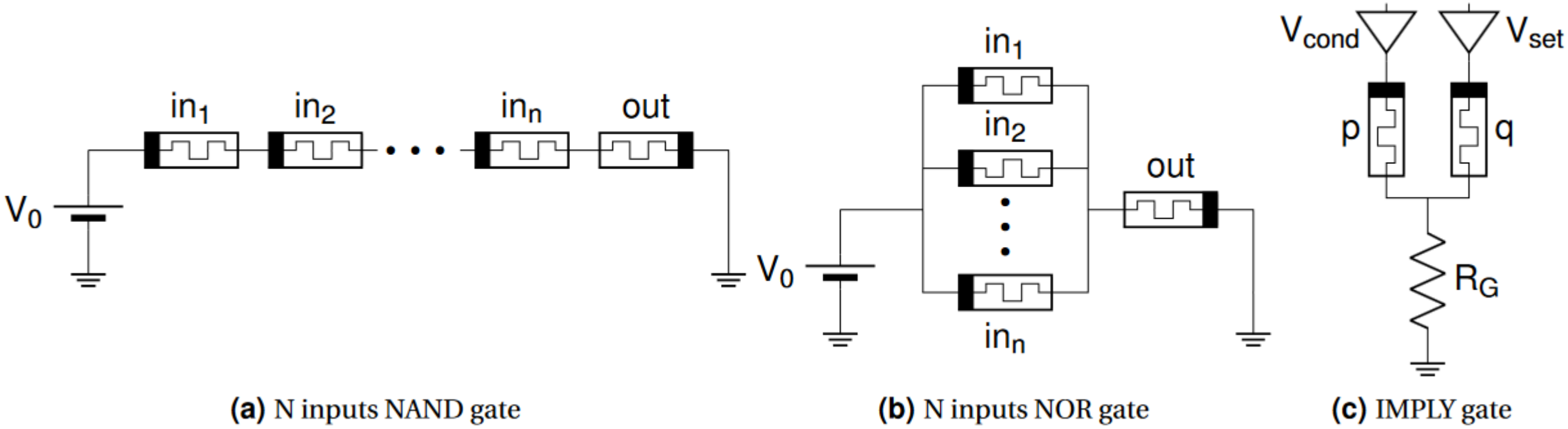
Vivek Seshadri et al. “Ambit” MICRO-50, 2017, DOI: 10.1145/3123939.3124544

NAND Flash computing commercial proposal



Peng Li et al. In: 2013 HPEC, DOI: 10.1109/HPEC.2013.6670317

RRAM IMC



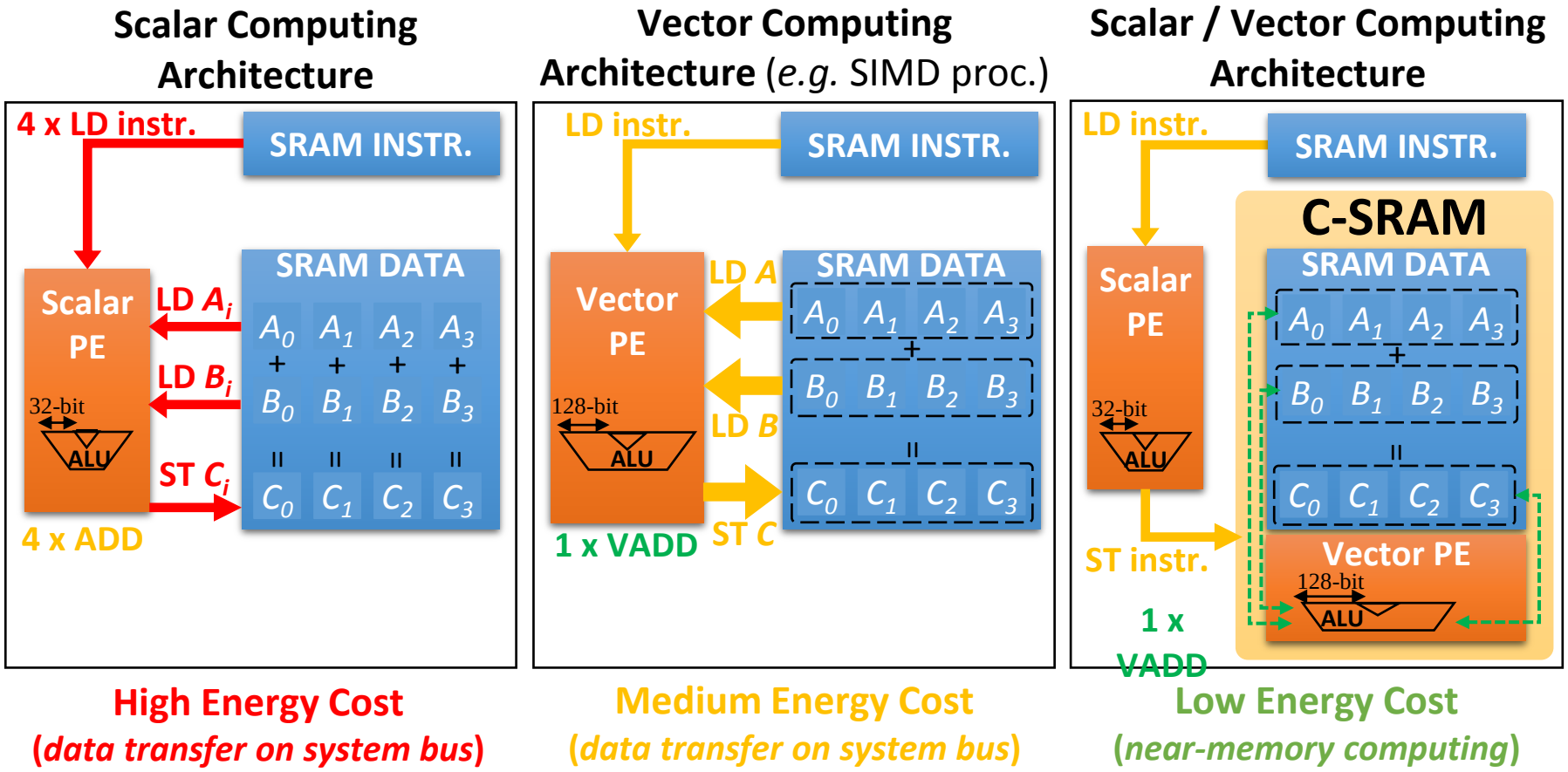
(a) N inputs NAND gate

(b) N inputs NOR gate

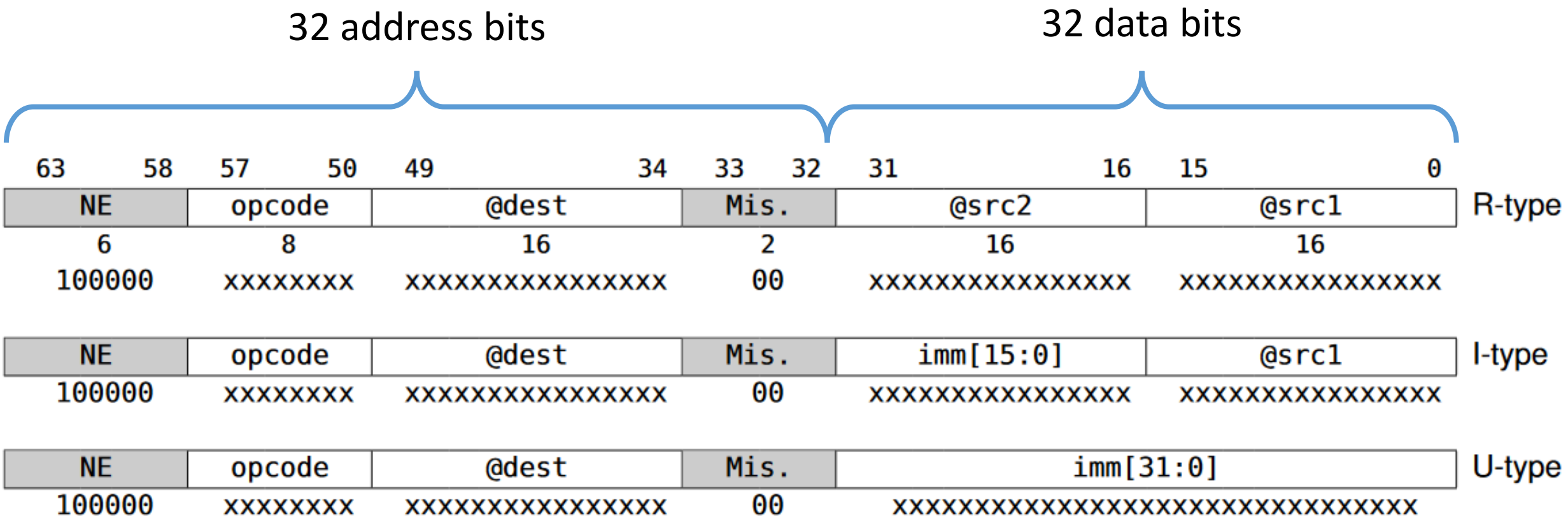
(c) IMPLY gate

S. Kvatinsky et al. “MAGIC”. In: IEEE Transactions on Circuits and Systems II (2014-11), DOI: 10.1109/TCSII.2014.2357292

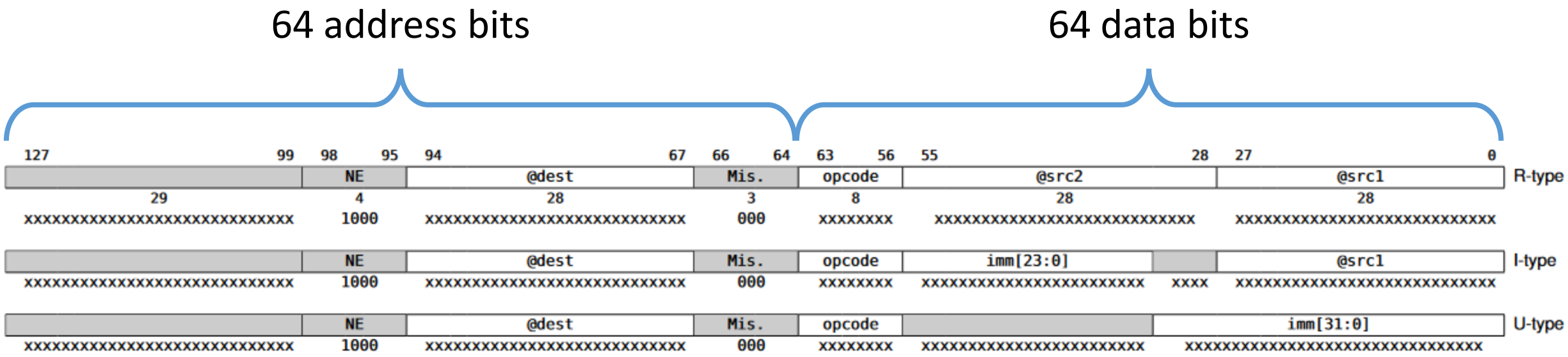
From scalar CPU to NMC vector architecture



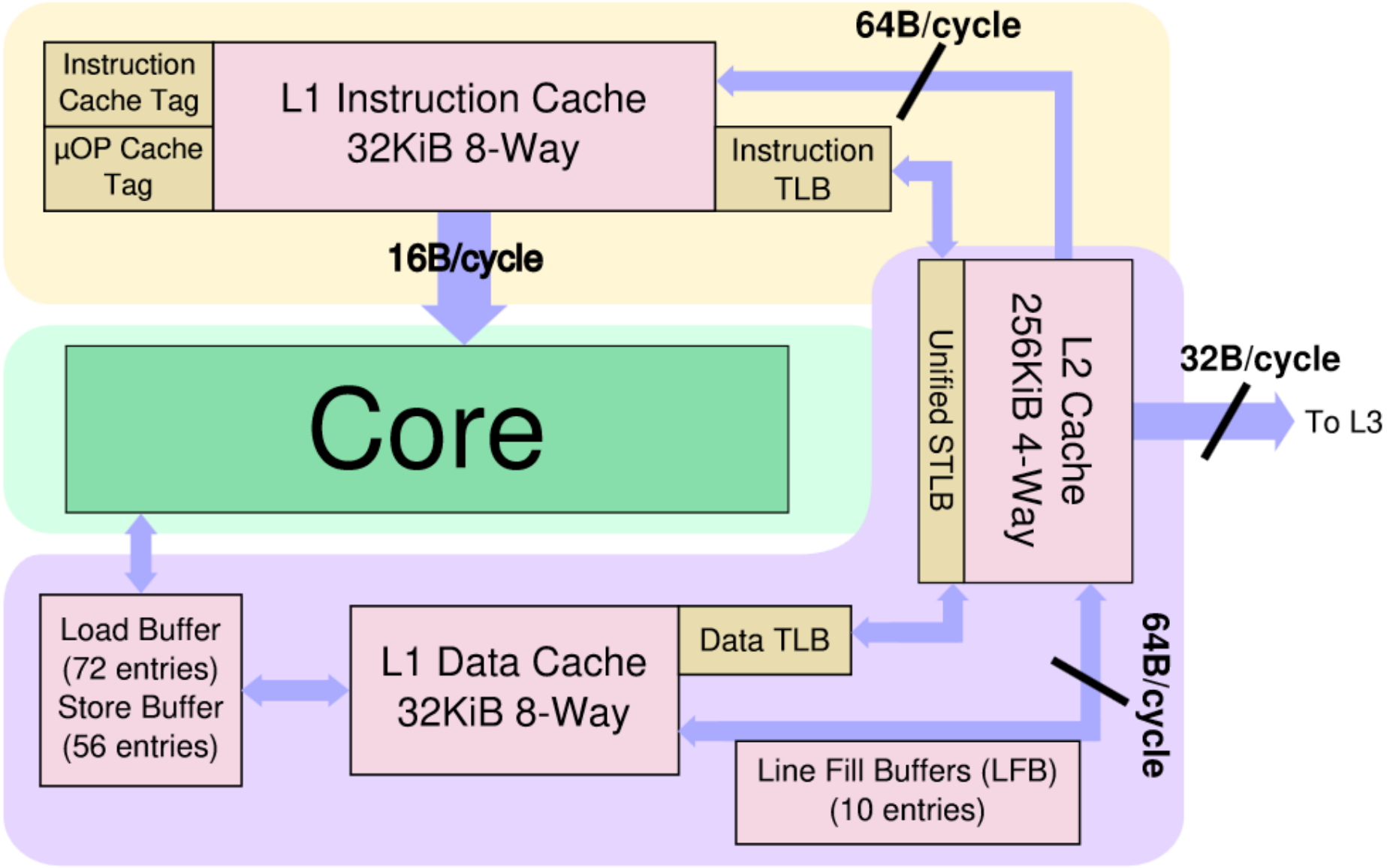
Defined 32 bits ISA (32 address bits + 32 data bits)



Extended 64 bits ISA



Simplified Intel Skylake memory system



PCM scaling

- Scaling only applies to routing

(a) Original 128 bits IO 128 MB PCRAM

	Energy	Latency
Read	0.875 nJ	142 ns
Write	4.88 nJ	512 ns

(b) Scaled 4 GB PCRAM

IO Width	Read		Write		Widen Factor
	Energy	Latency	Energy	Latency	
512 B	38.5 nJ	263 ns	215 nJ	949 ns	0
1 kB	77.0 nJ	447 ns	429 nJ	1.61 µs	1
2 kB	154 nJ	759 ns	858 nJ	2.74 µs	2
4 kB	308 nJ	1.29 µs	1.72 µJ	4.66 µs	3

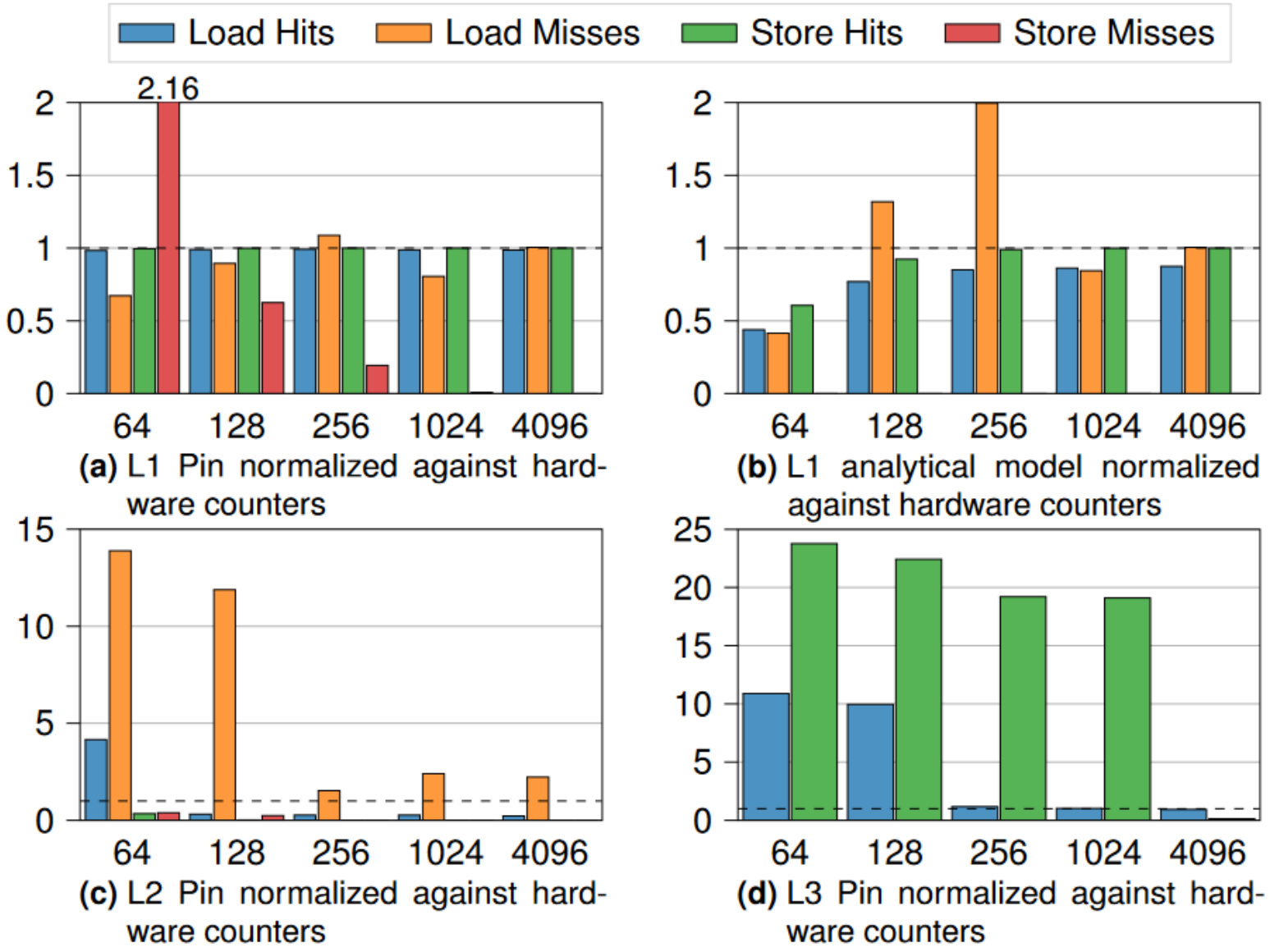
SRAM tiling factors

Size	16 kB	32 kB	64 kB	128 kB	256 kB	512 kB	1–8 MB
N_{cuts}	8	16	16	32	32	32	64
E_{tf}	10 %	10 %	19 %	19 %	37 %	37 %	37 %
T_{tf}	30 %	30 %	56 %	56 %	85 %	85 %	85 %

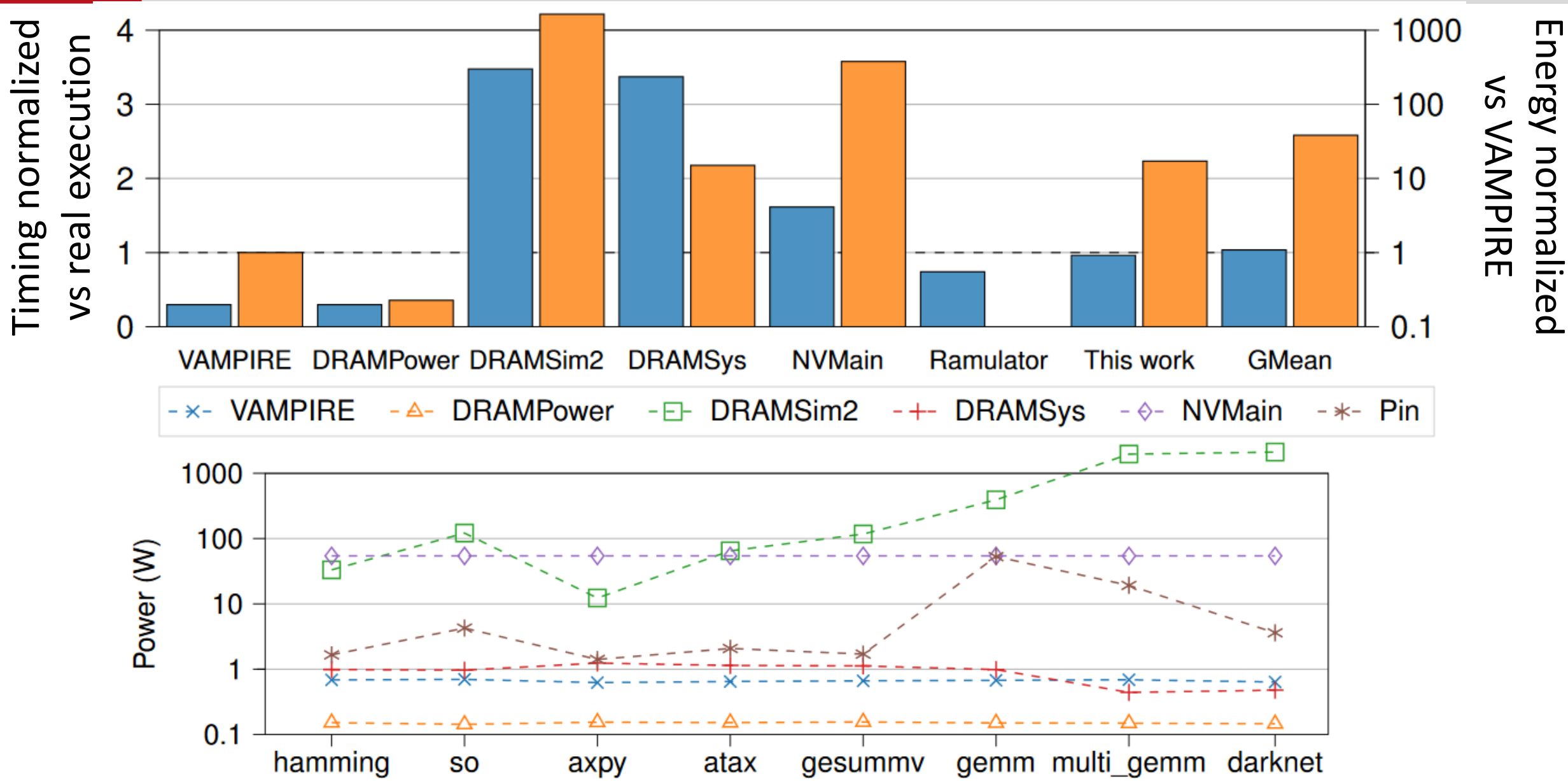
$$E = (E_{base} \times E_{Wf}^{W_f}) \times E_{tf} \times N_{cuts}$$

$$T = (T_{base} \times T_{Wf}^{W_f}) \times T_{tf}$$

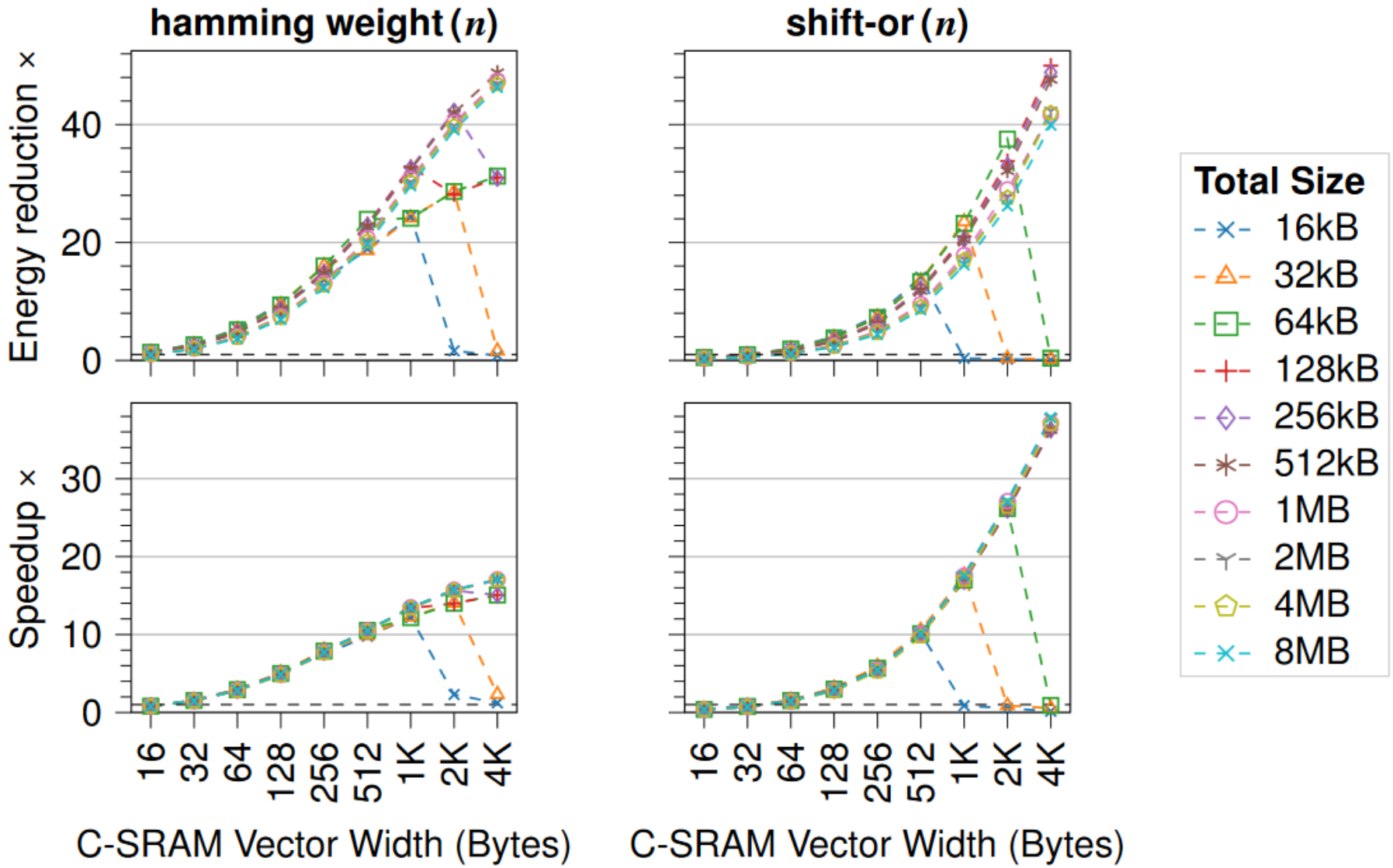
Platform validation: caches accesses on gemm



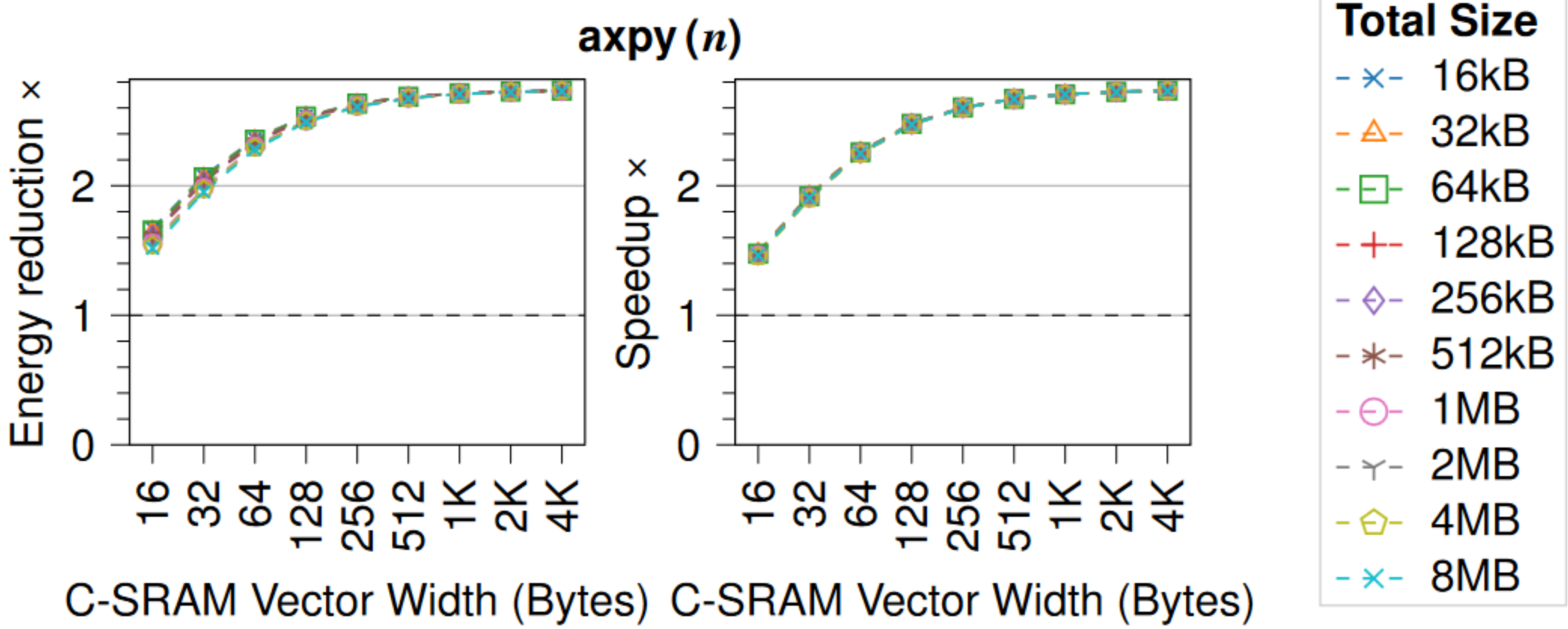
Platform validation: DRAM accesses cost



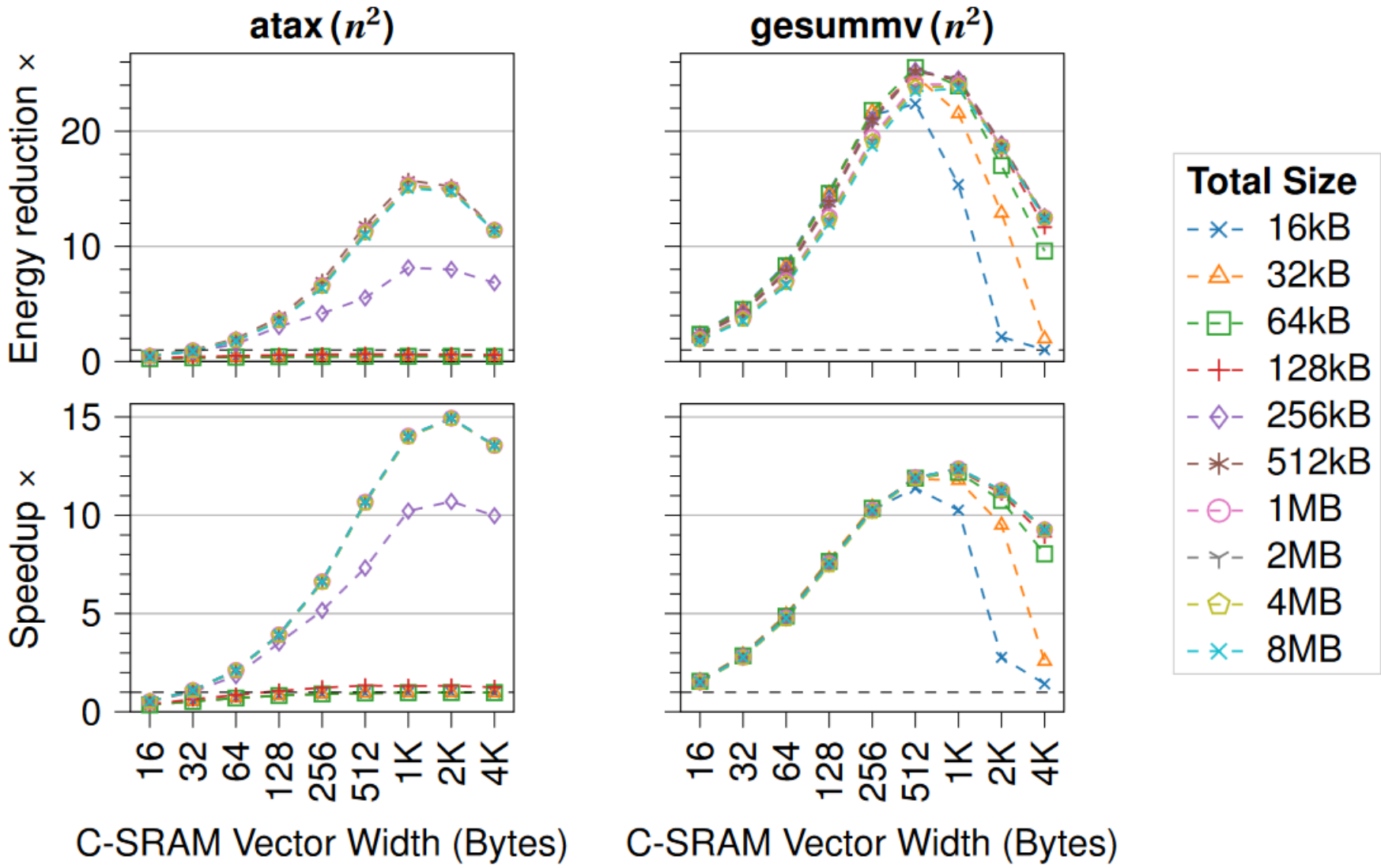
Linear kernels: hamming weight & shift-or



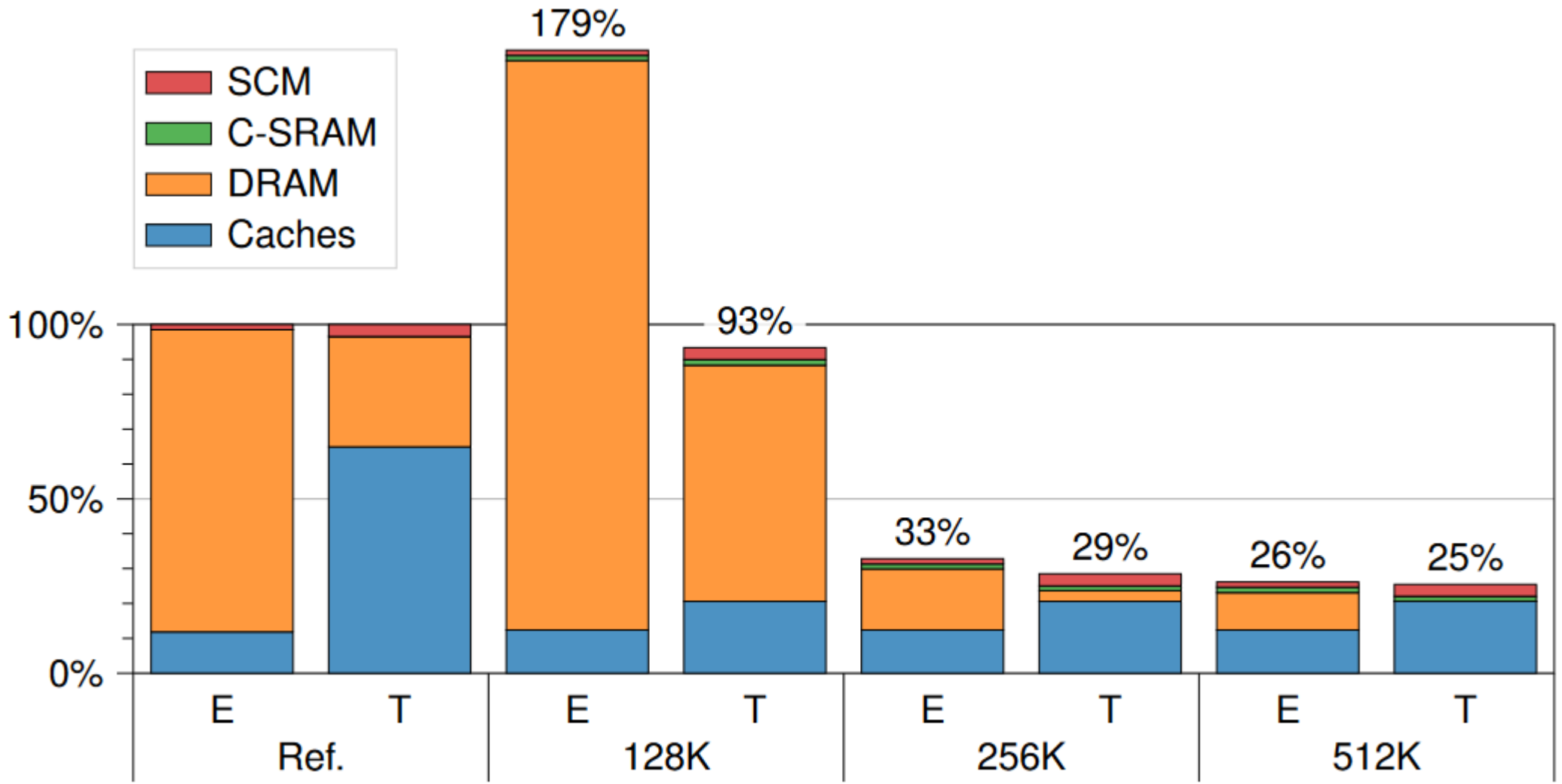
Linear kernels: axpy



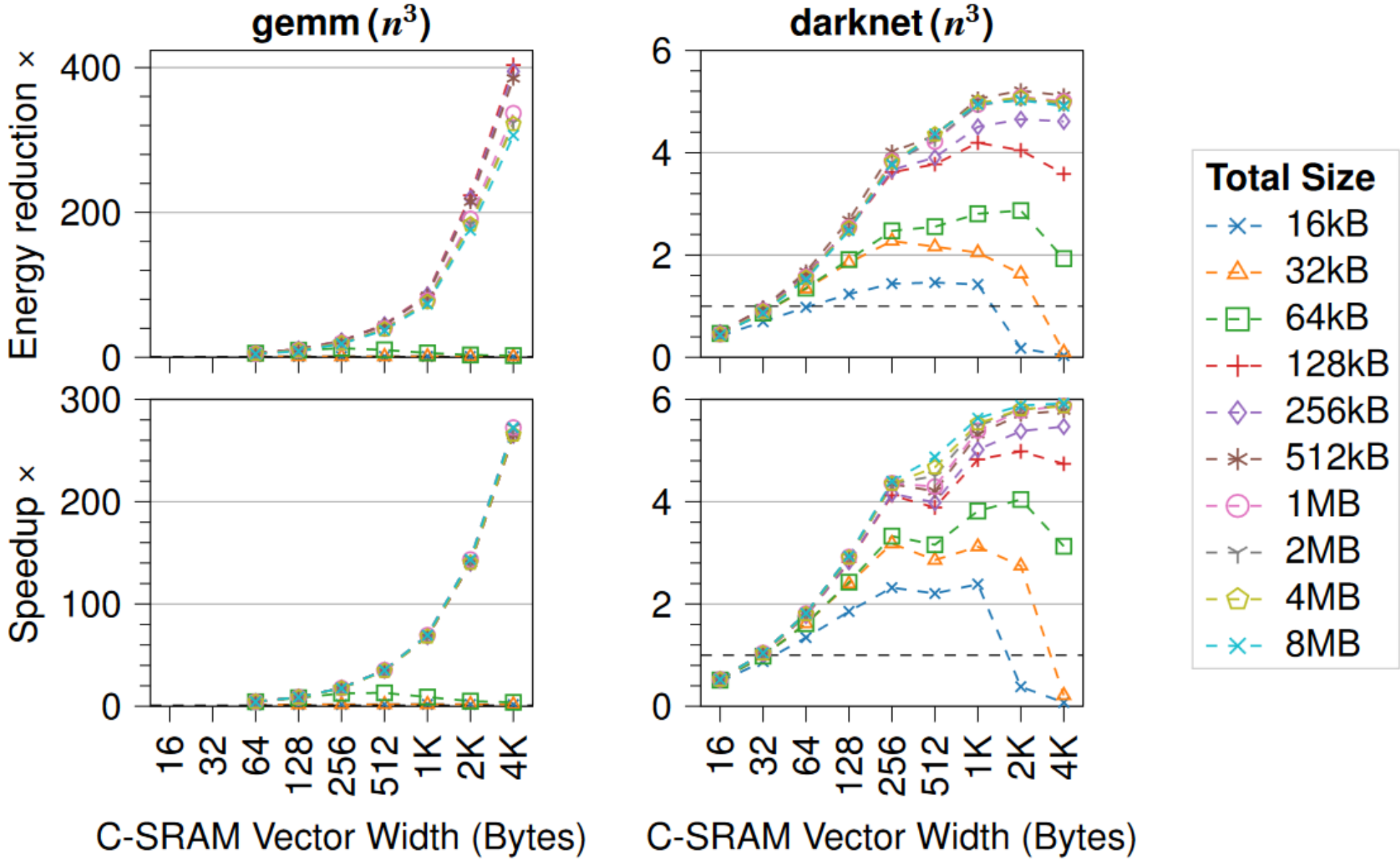
Quadratic kernels: atax & gesummv



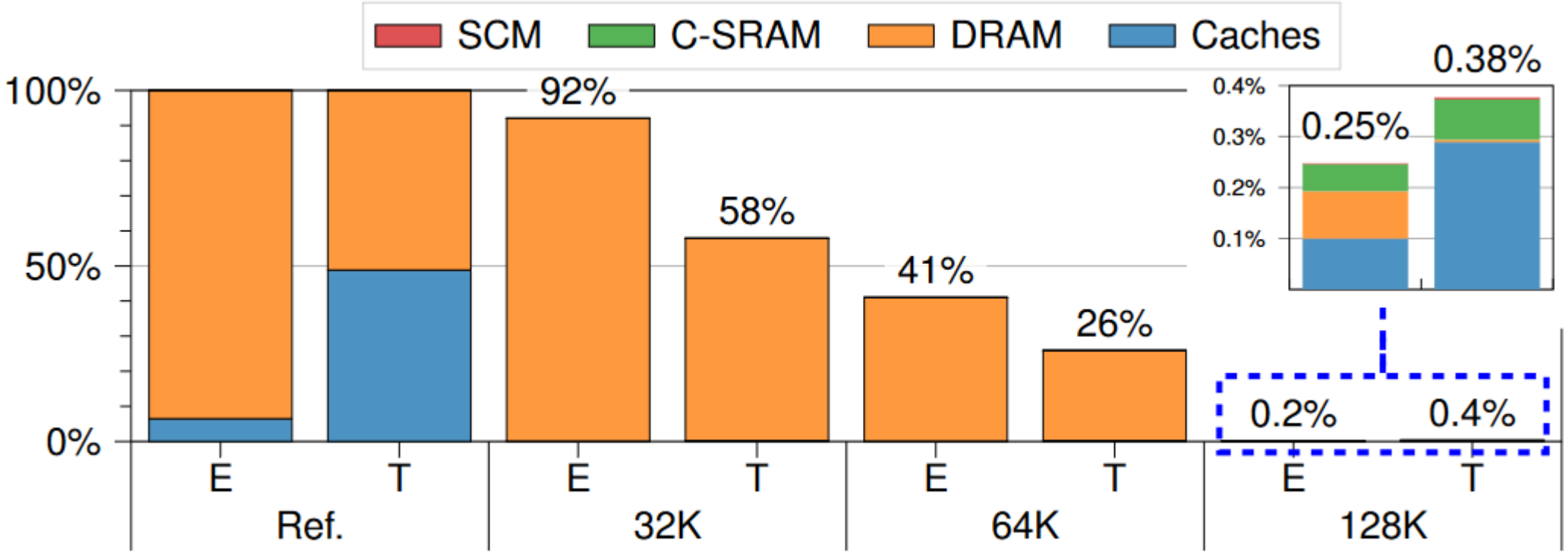
Quadratic kernels distribution: atax sudden jump



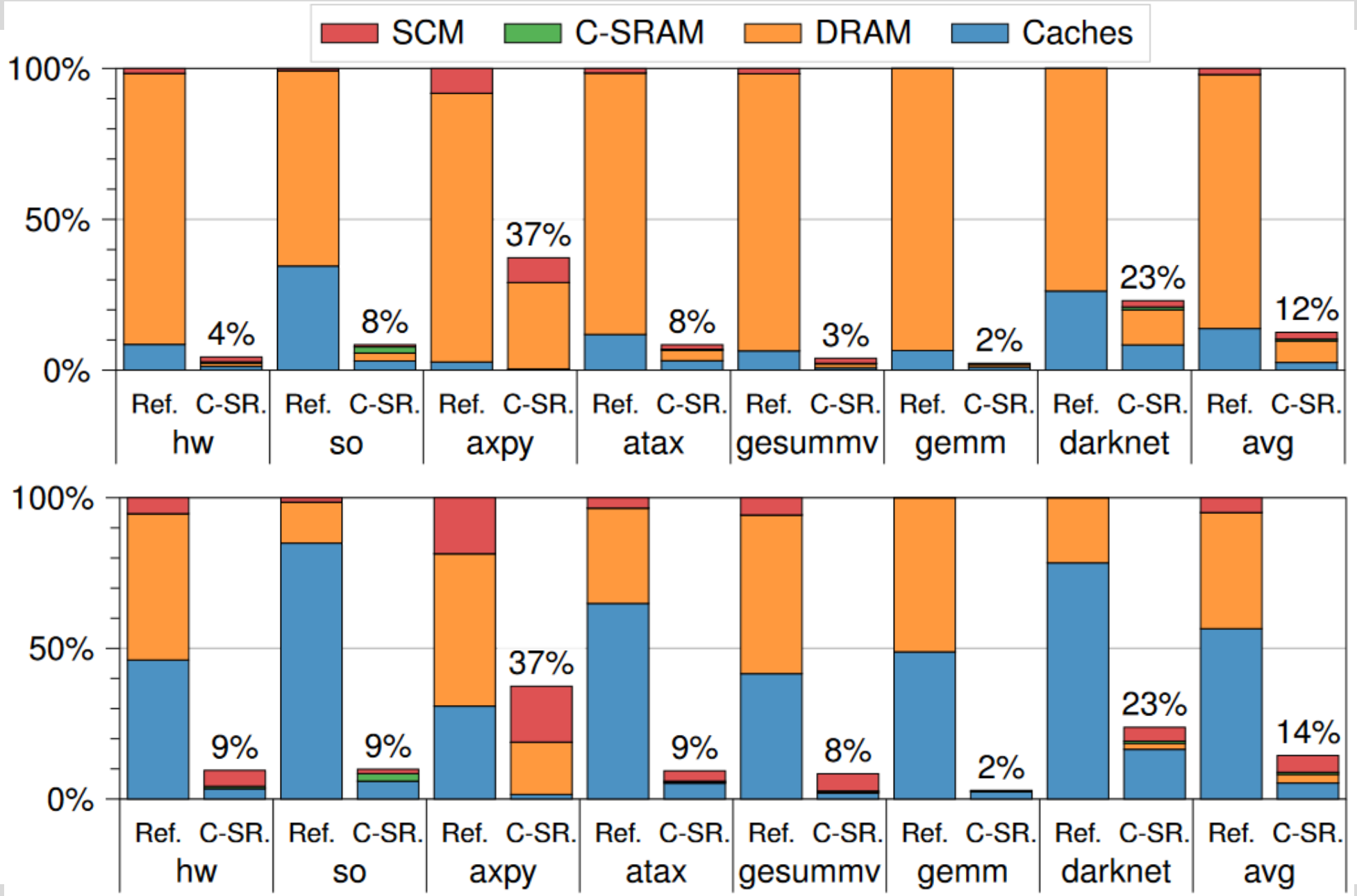
Cubic kernels: gemm & darknet



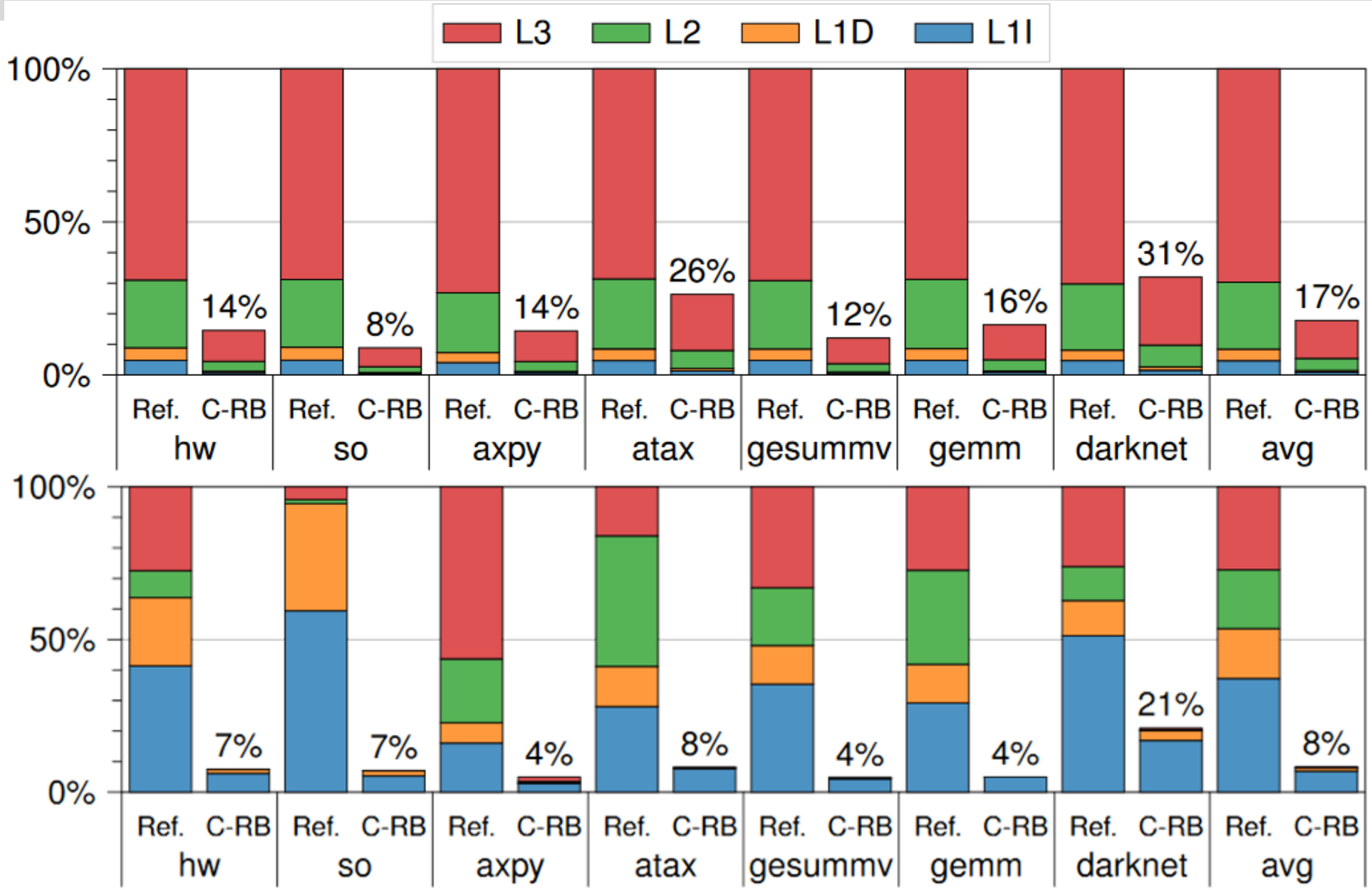
Cubic kernels: gemm sudden jump



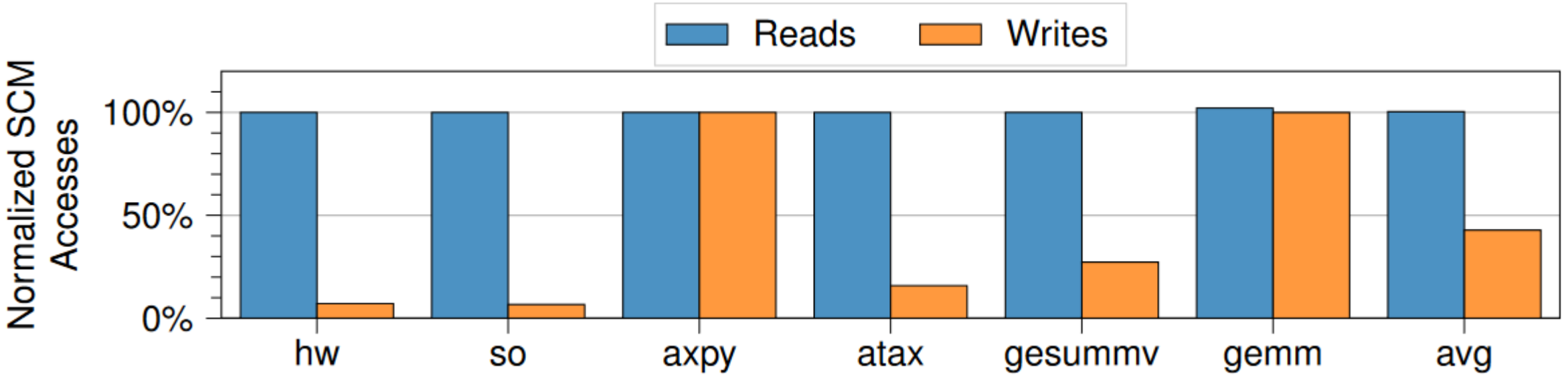
All benchmarks distribution



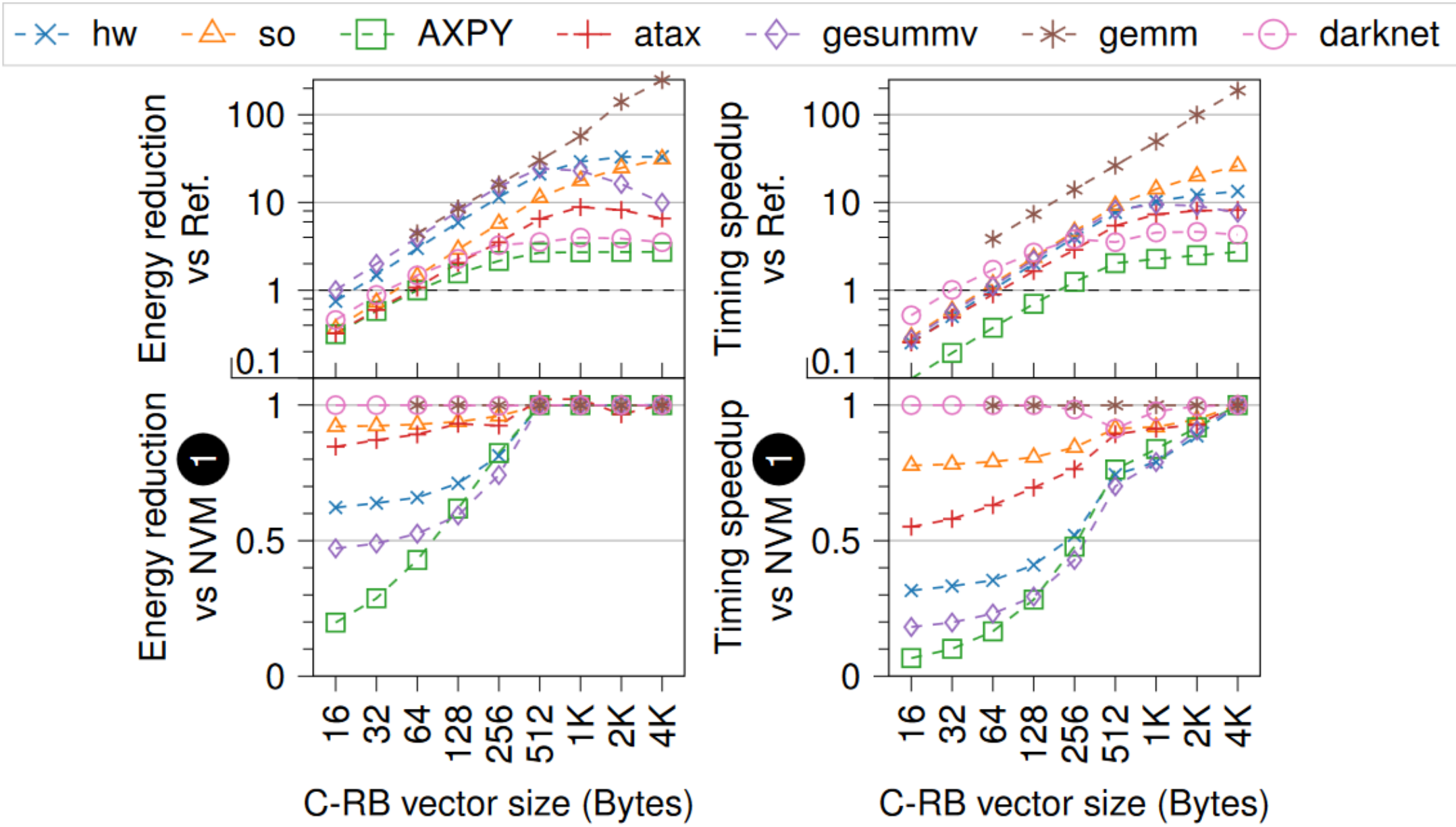
All benchmarks caches distribution



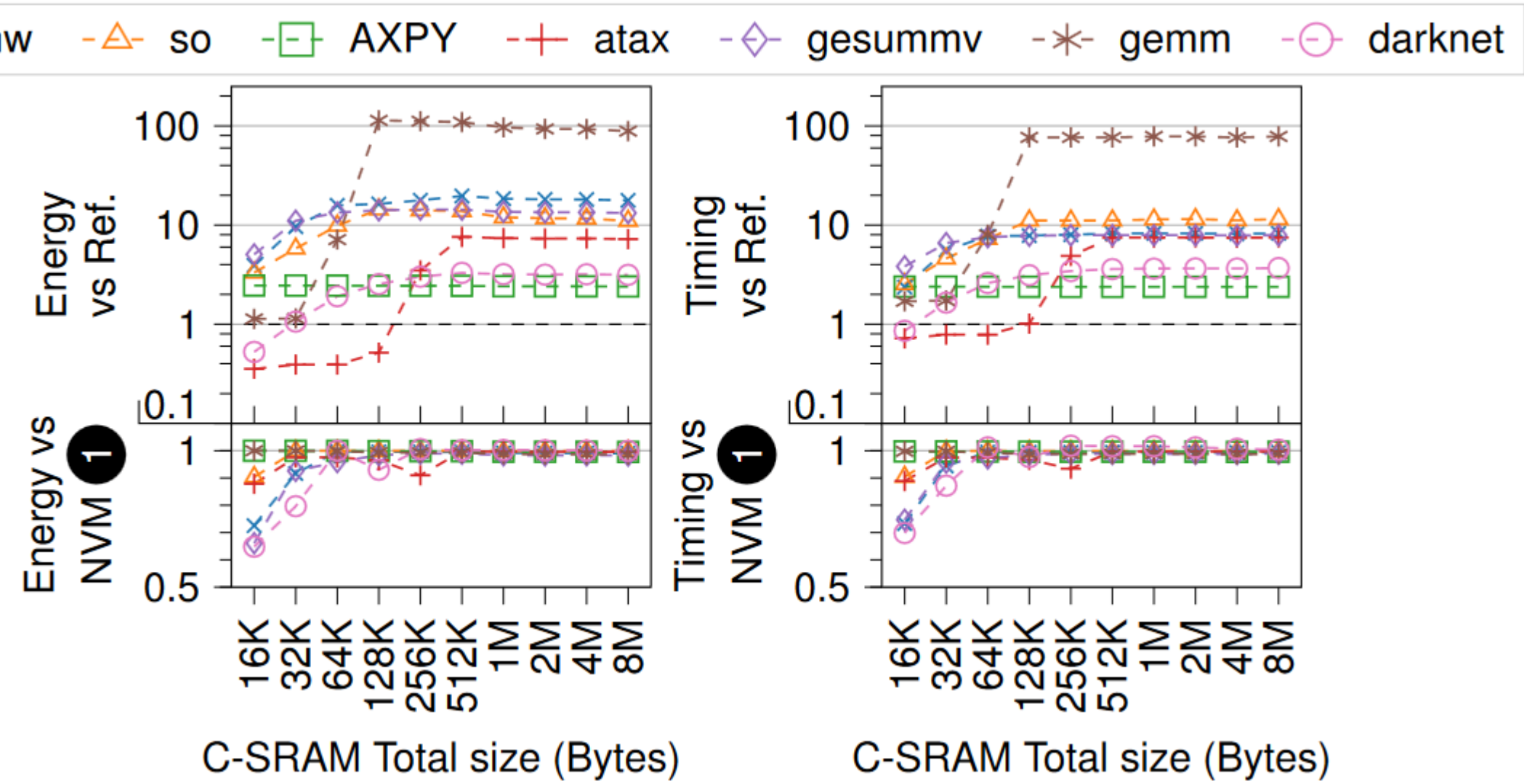
Endurance is saved



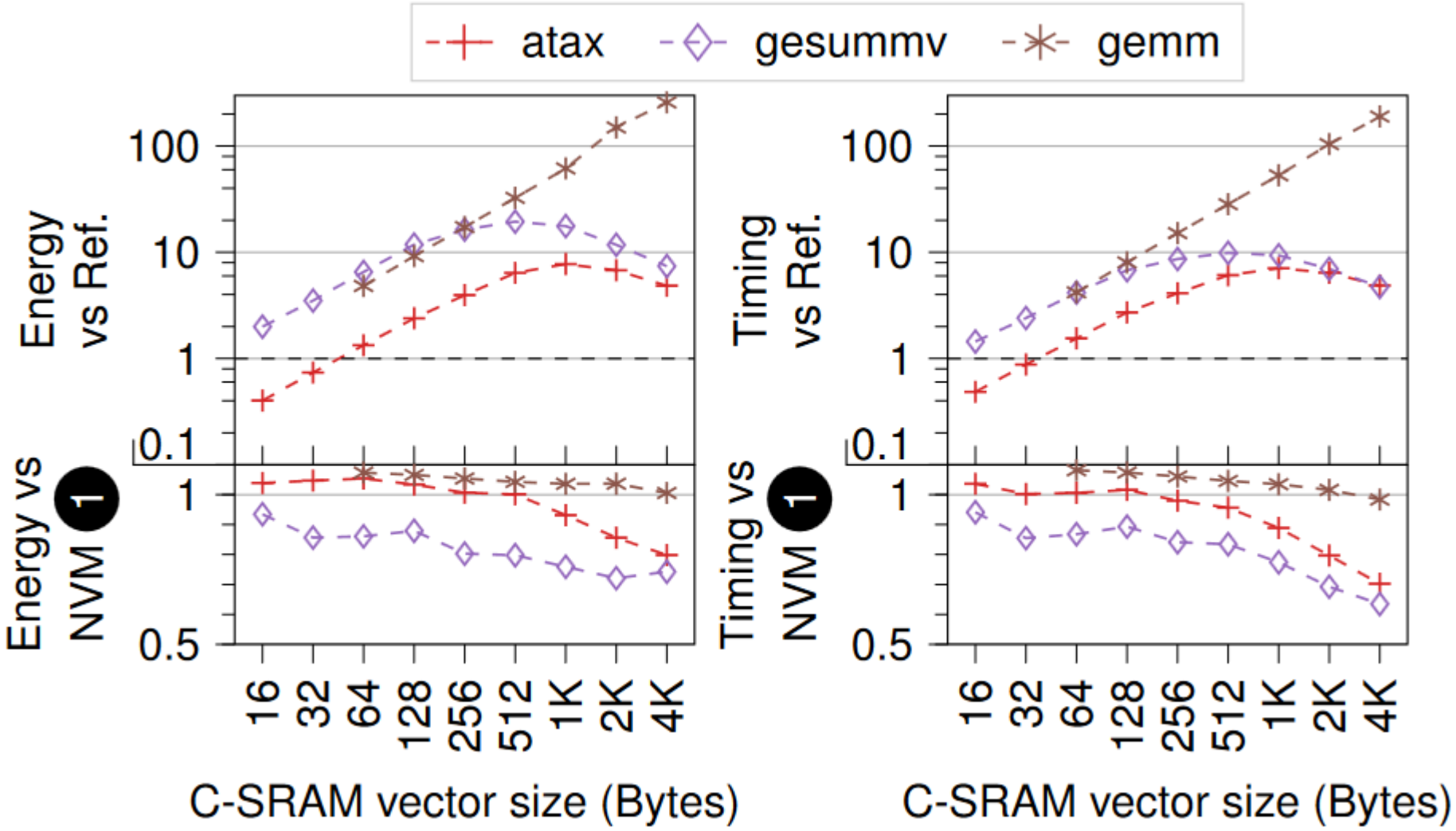
Row buffer scenario



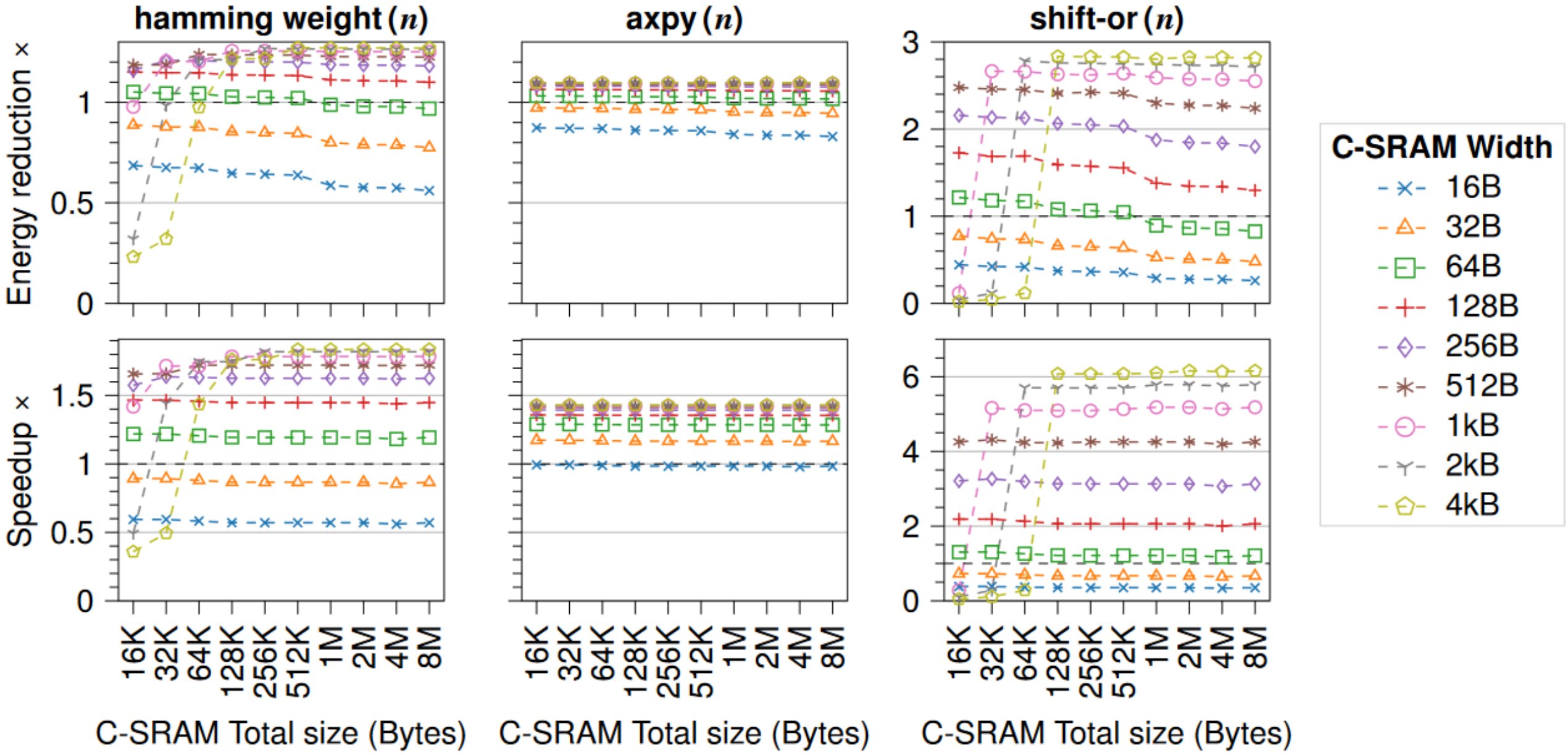
Base scenario but with paged size transfer



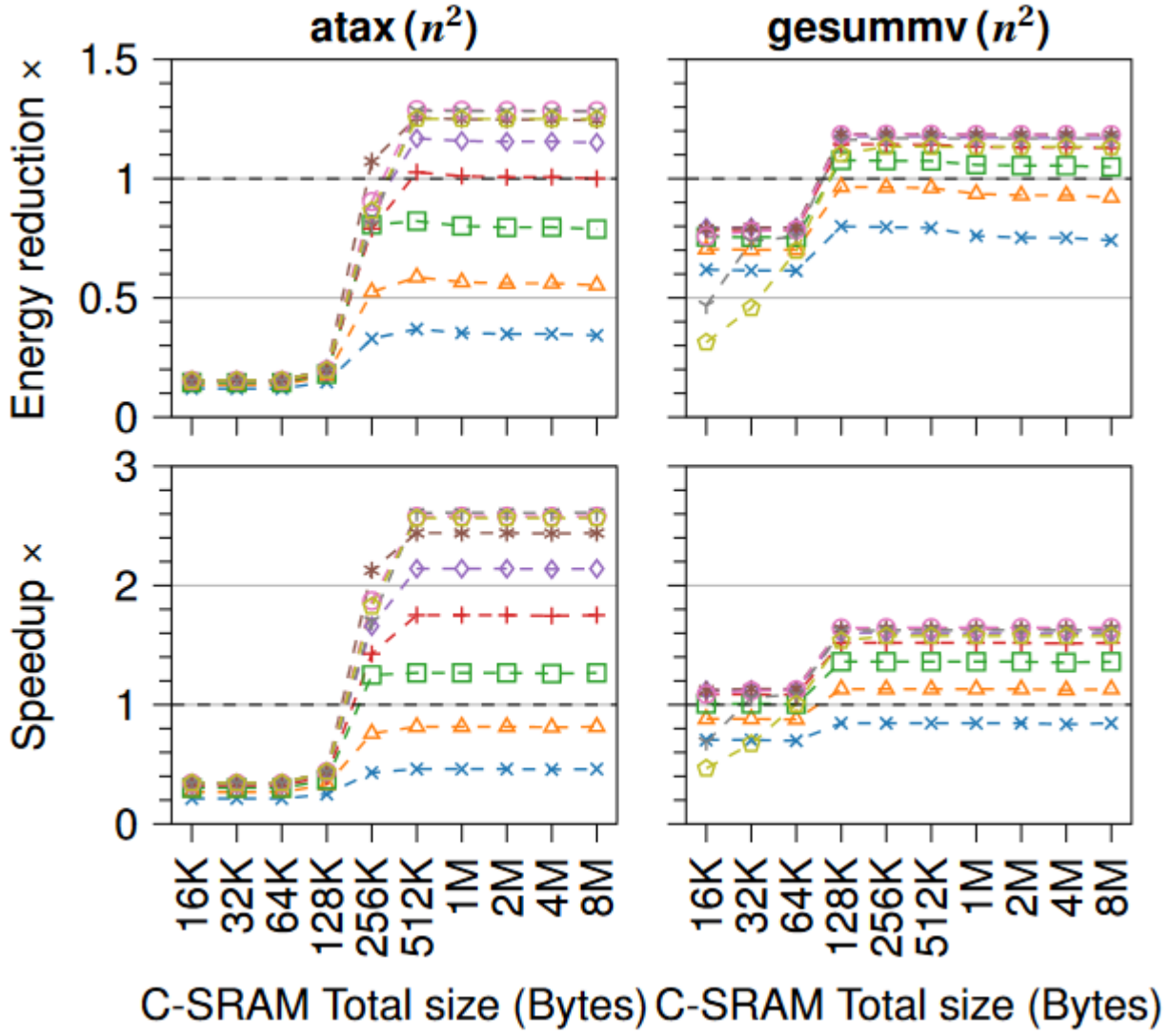
Base scenario with reduction loop performed in CSRAM



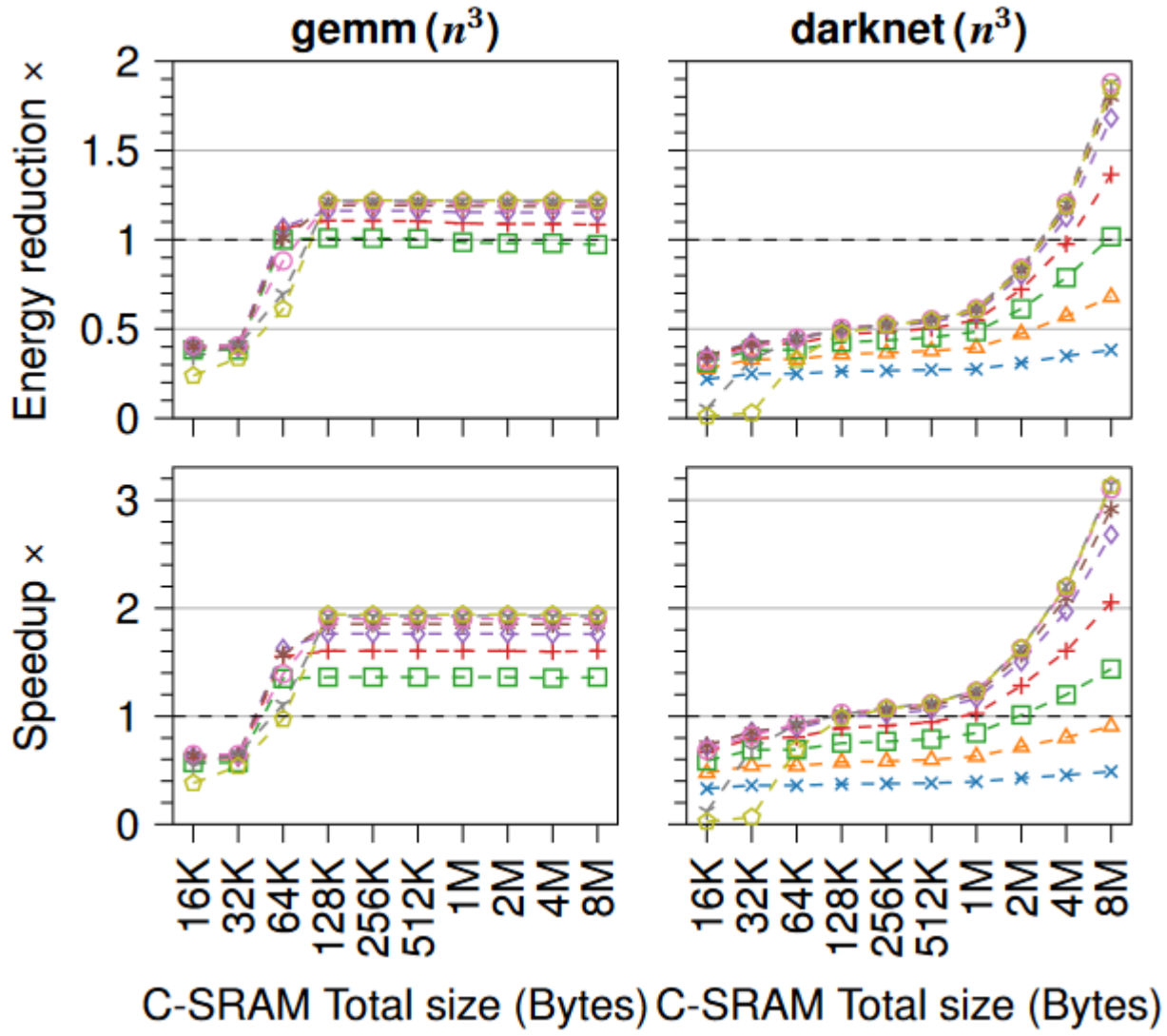
DRAM scenario: linear kernels



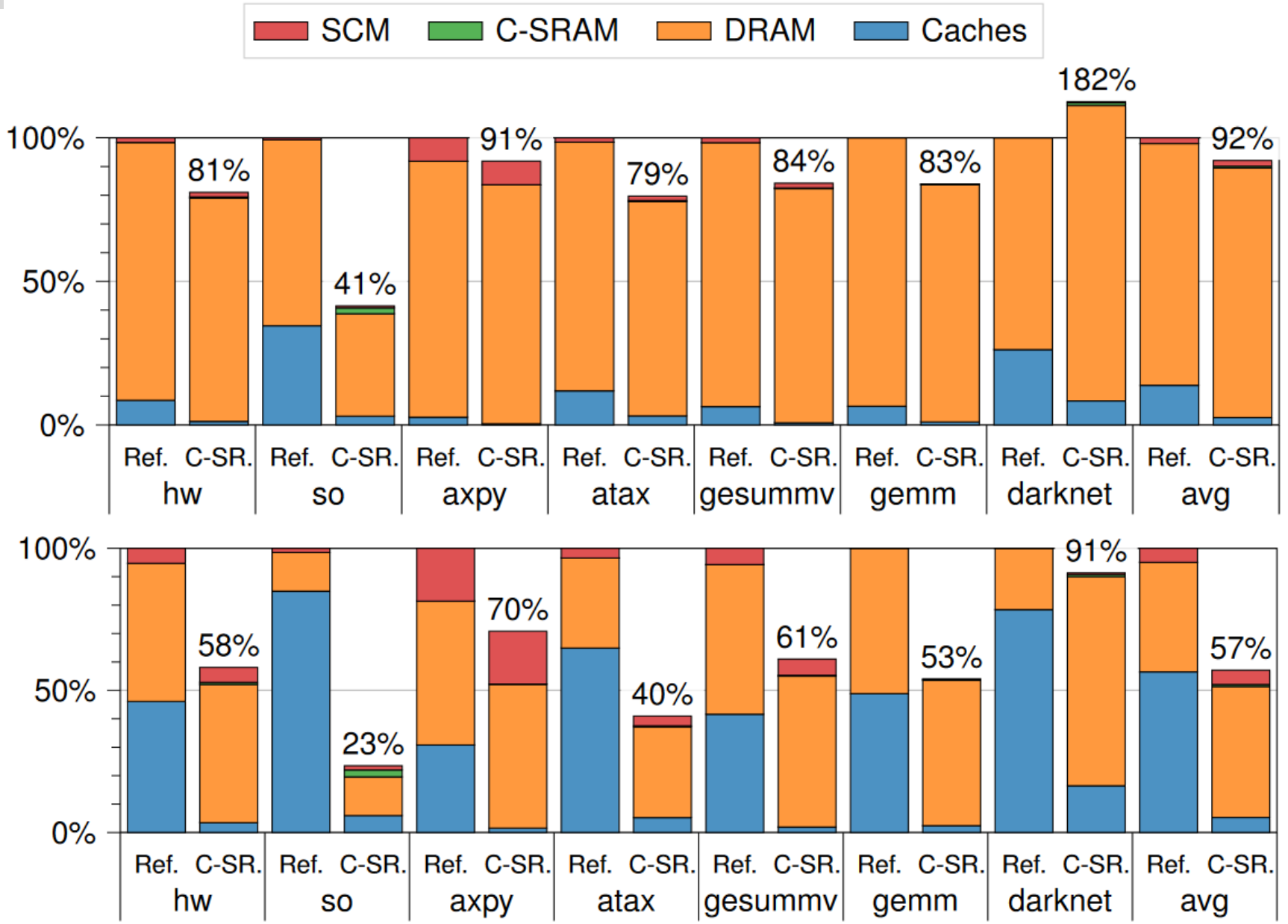
DRAM Scenario: quadratic kernels

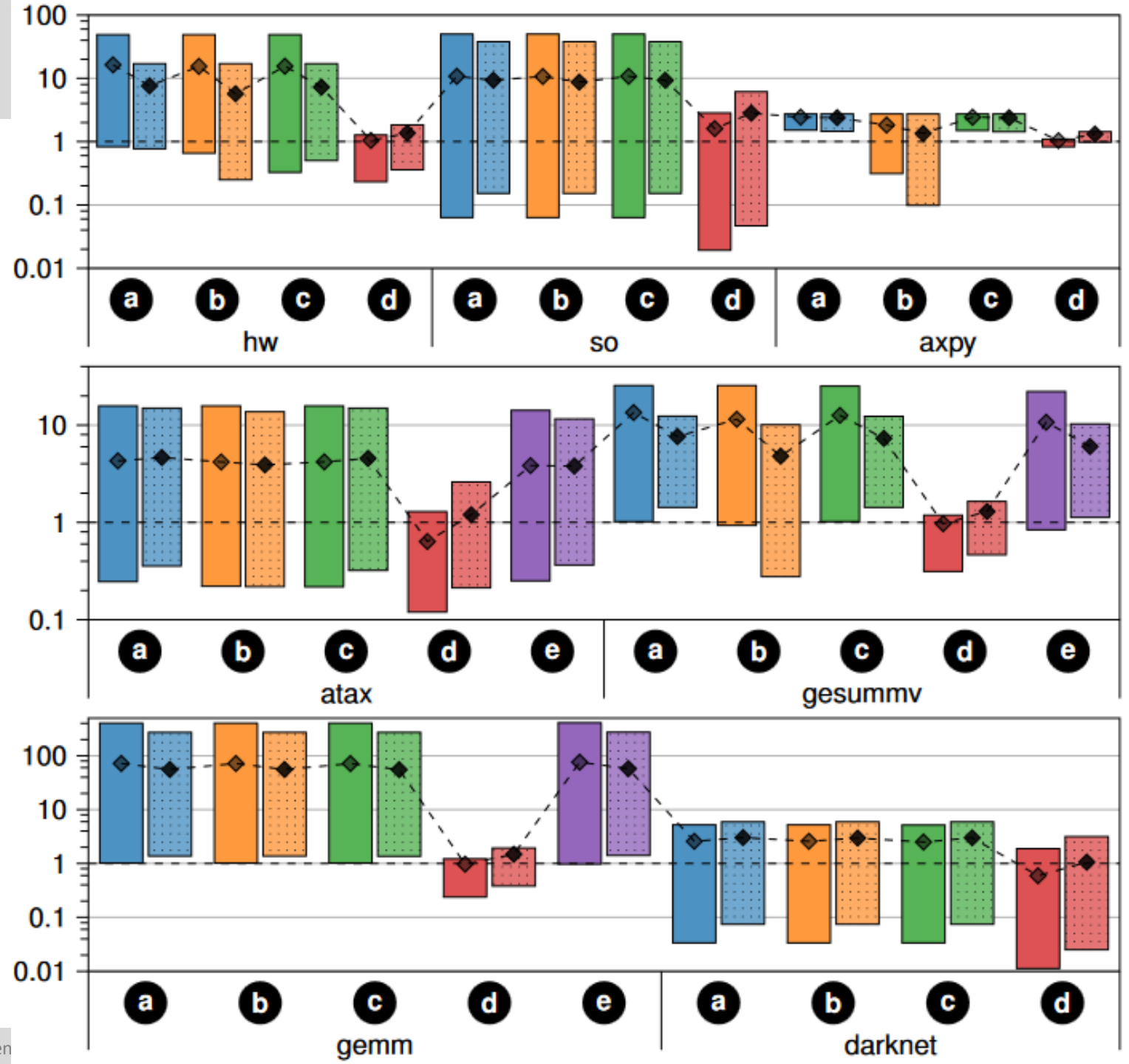


DRAM Scenario: cubic benchmarks

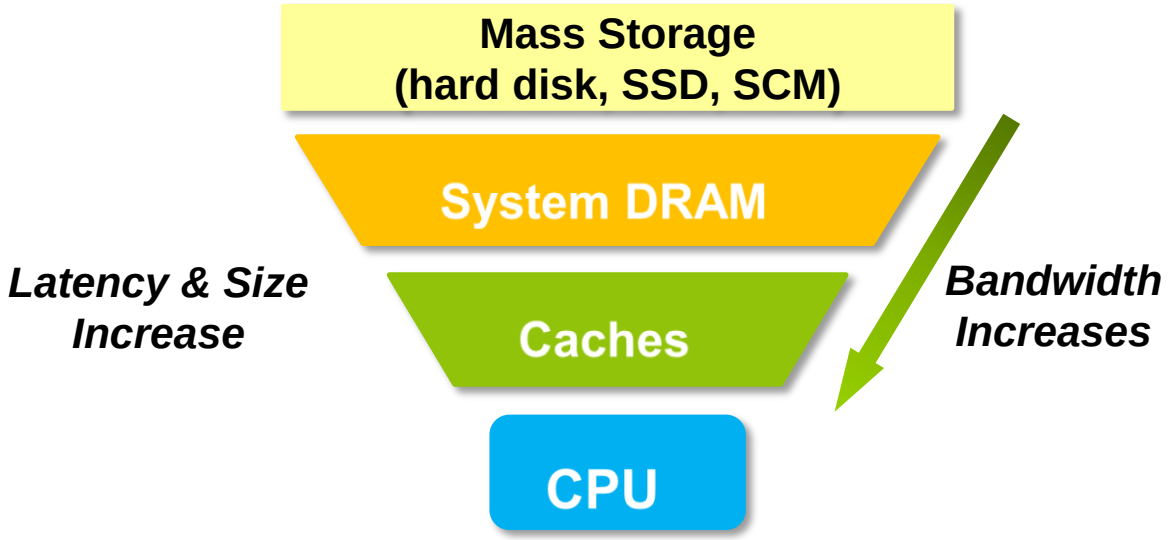


DRAM scenario distribution

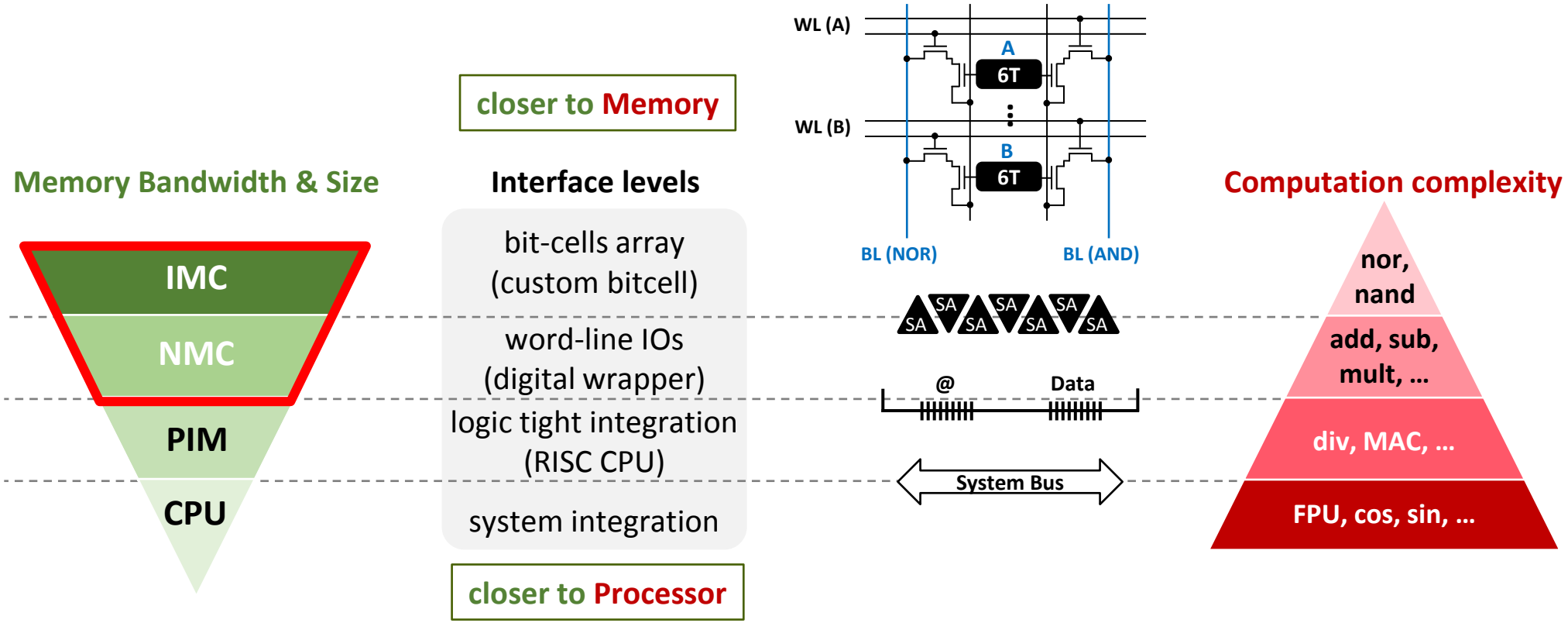



100
10
1
0.1
0.01
a
b
c
d
hw
so
axpy
e
atax
gesummv
gemm
darknet

Copy paste

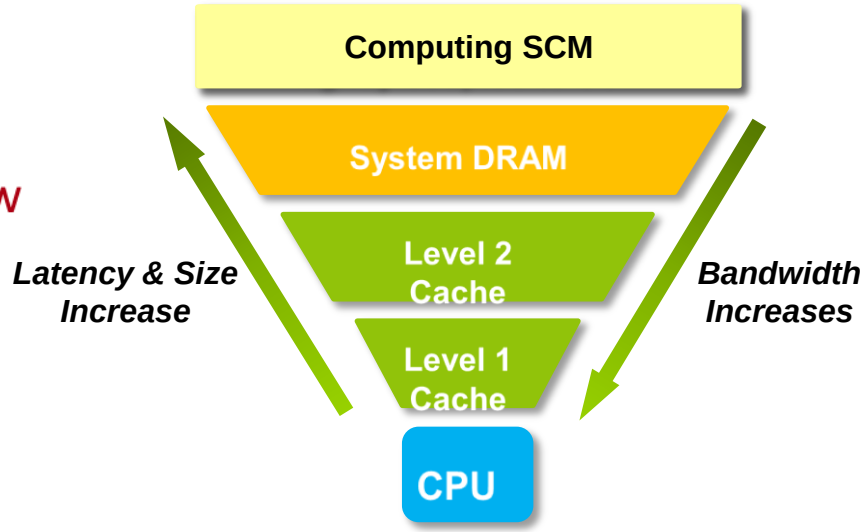

Mass Storage
(hard disk, SSD, SCM)
System DRAM
Caches
CPU
Latency & Size
Increase
Bandwidth
Increases

Taxonomy


Memory Bandwidth & Size
IMC
NMC
PIM
CPU
closer to Memory
Interface levels
bit-cells array
(custom bitcell)
word-line IOs
(digital wrapper)
logic tight integration
(RISC CPU)
system integration
closer to Processor
WL (A)
WL (B)
A
6T
B
6T
BL (NOR)
BL (AND)
SA
@
Data
System Bus
Computation complexity
nor,
nand
add, sub,
mult, …
div, MAC, …
FPU, cos, sin, …

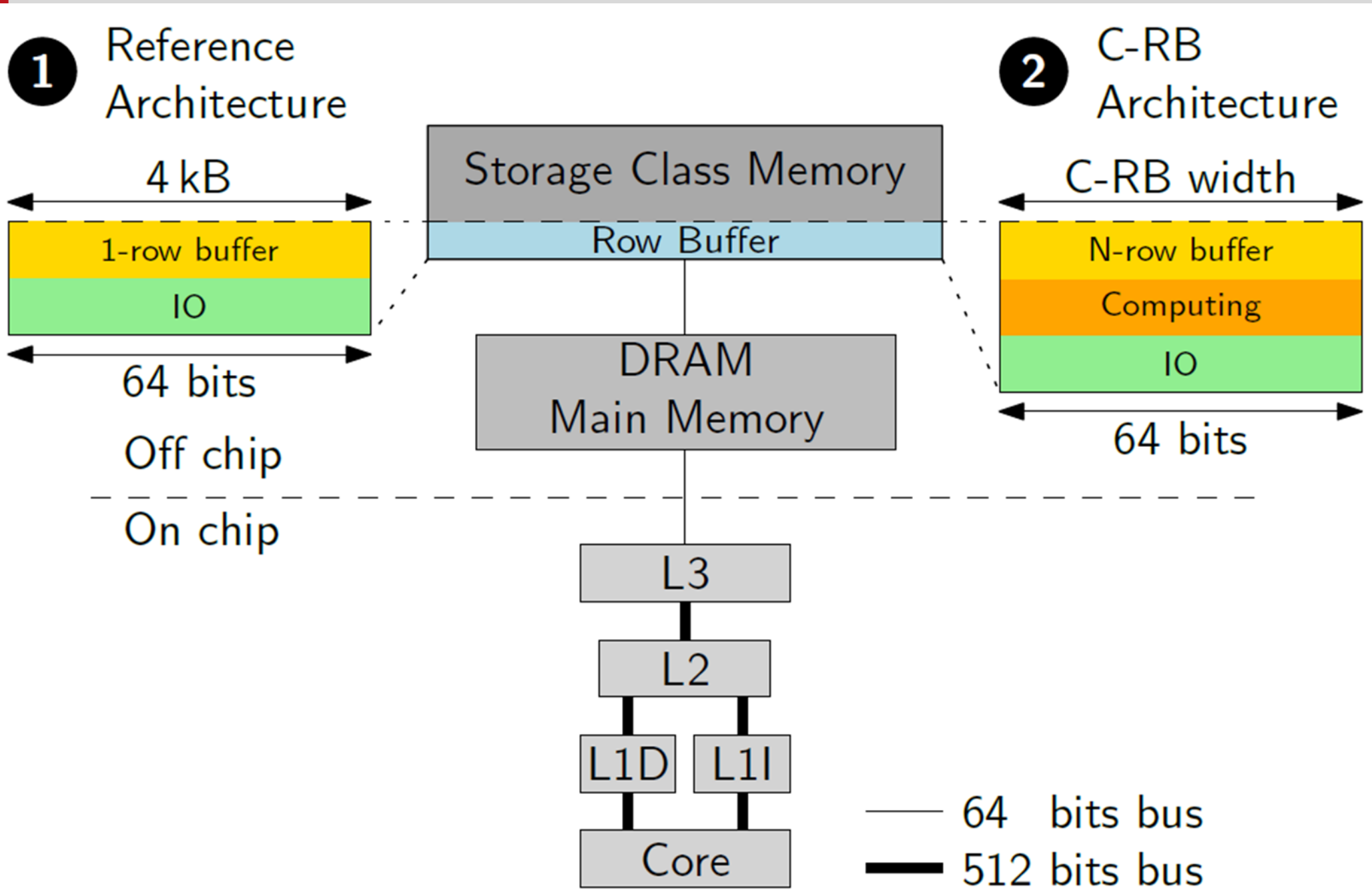
Main thesis idea

- Merge computing and data in the same circuit
  - Between energy efficiency and slow
  - High speed is a problem (10^9)
  - Endurance is unlimited
  - Use existing row buffer and repurpose it for computing



- DRAM is still used as a write buffer
- Caches can be reduced (Instructions & stack)

Let's move computing at the top with tightly coupled SRAM to get the best of both worlds!


1
Reference Architecture
2
C-RB Architecture
4 kB
Storage Class Memory
C-RB width
1-row buffer
Row Buffer
N-row buffer
IO
Computing
IO
64 bits
DRAM Main Memory
64 bits
Off chip
On chip
L3
L2
L1D
L1I
Core
64 bits bus
512 bits bus