BREAKING THE MEMORY WALL

CS433 Fall 2015

Dimitrios Skarlatos

OUTLINE

- Introduction
- Current Trends in Computer Architecture
- 3D Die Stacking
- The memory Wall
- Conclusion

INTRODUCTION

- Ideal Scaling of power with feature size is long gone
- Current feature size 14nm (Skylake), 5nm by 2020
- Power Wall: consume exponentially increasing power with each factorial increase of frequency
- Memory Wall: growing disparity between CPU clock rates and off-chip memory and disk drive I/O rates.

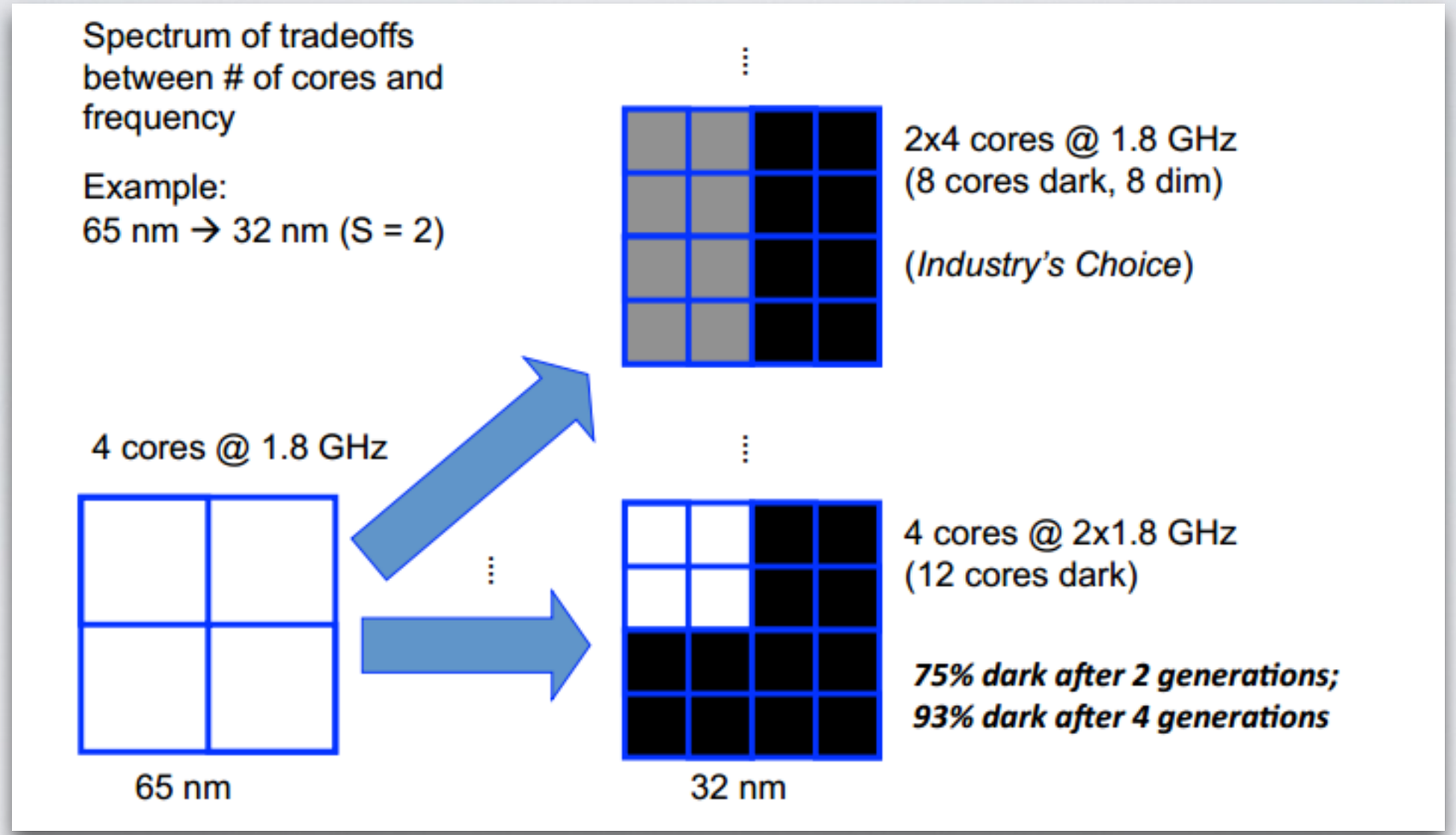
SOLUTIONS

- Dark Silicon
- Accelerators
- NTC
- Go vertical!! 3D die stacking

DARK SILICON

- The amount of "silicon" that can not be powered on at nominal operating voltage for a given thermal design power (TDP) constraint.

DARK SILICON IN THE MULTICORE ERA

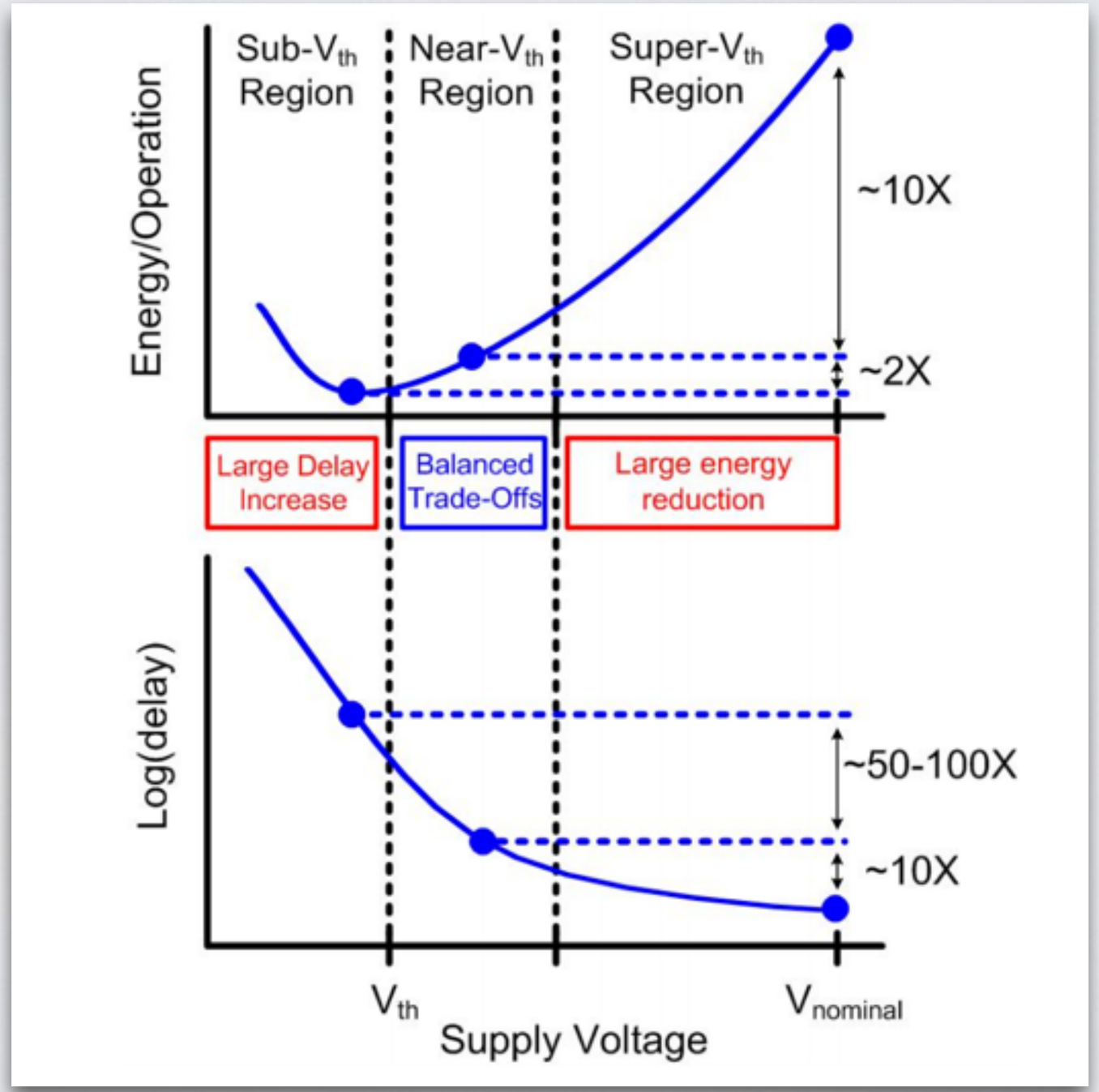


M.B. Taylor : Harnessing the Four Horsemen of the Coming Dark Silicon Apocalypse

ACCELERATORS

- Specialized hardware -> High performance @ Low Power
- FPU (?)
- Video | Audio (H.264)
- GPUs - FPGAS

NEAR THRESHOLD COMPUTING

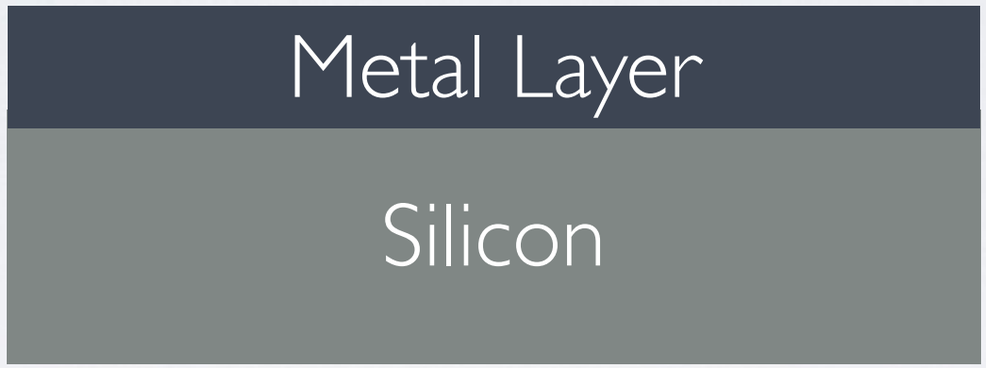


SCALCORE

- ScalCore: Designing a Core for Voltage Scalability
- How to design a core to efficiently scale from Near threshold to High Performance Mode

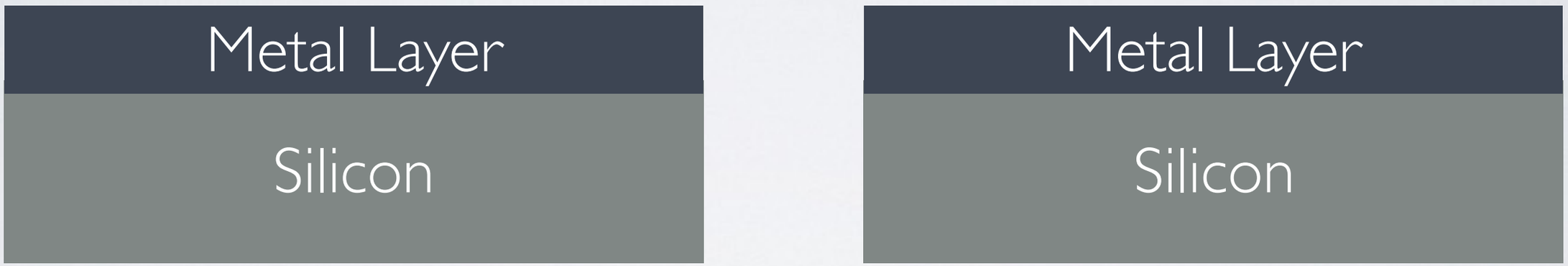
B. Gopireddy et al. HPCA 2016

3D DIE STACKING



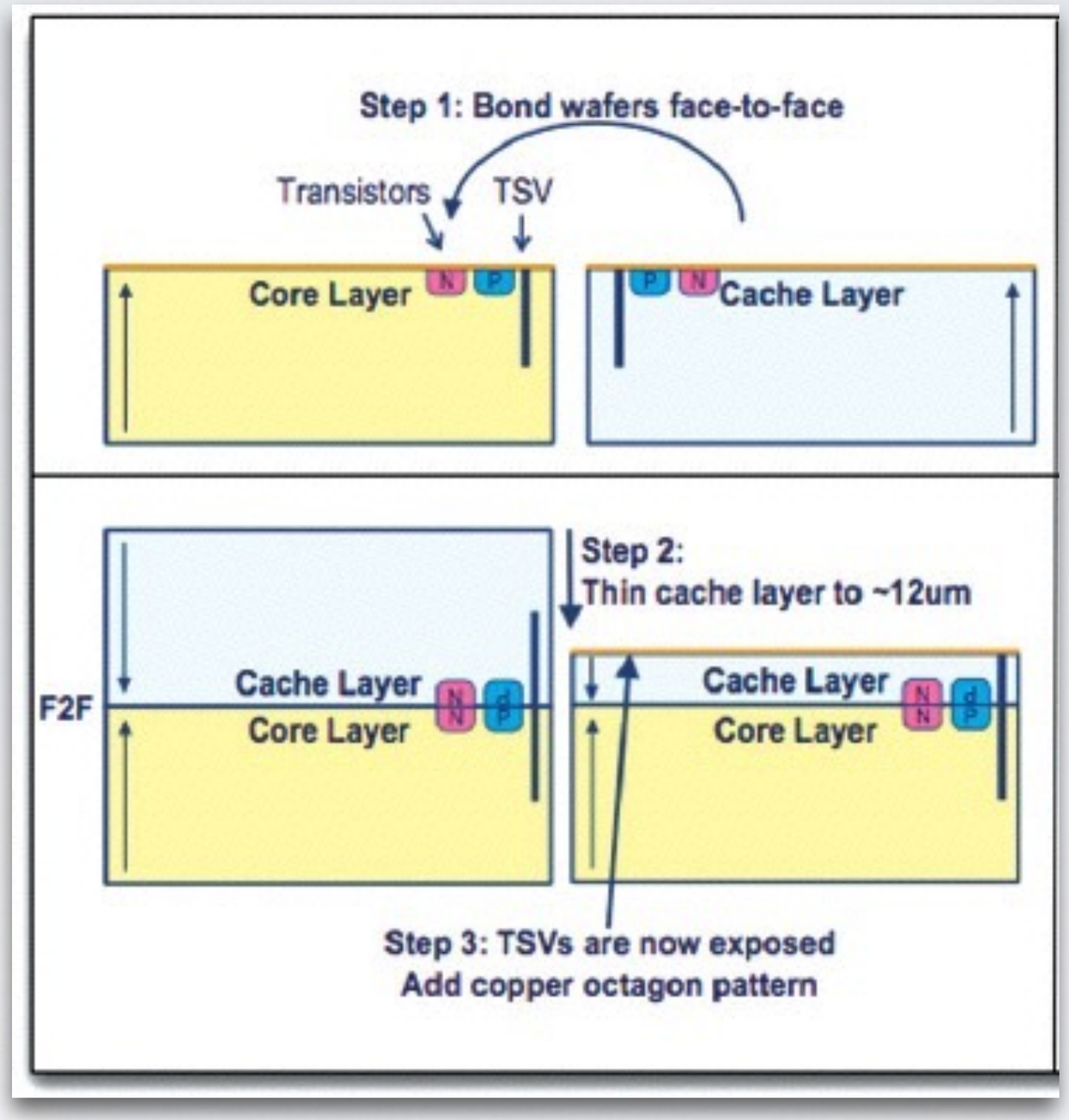
One Die

3D DIE STACKING



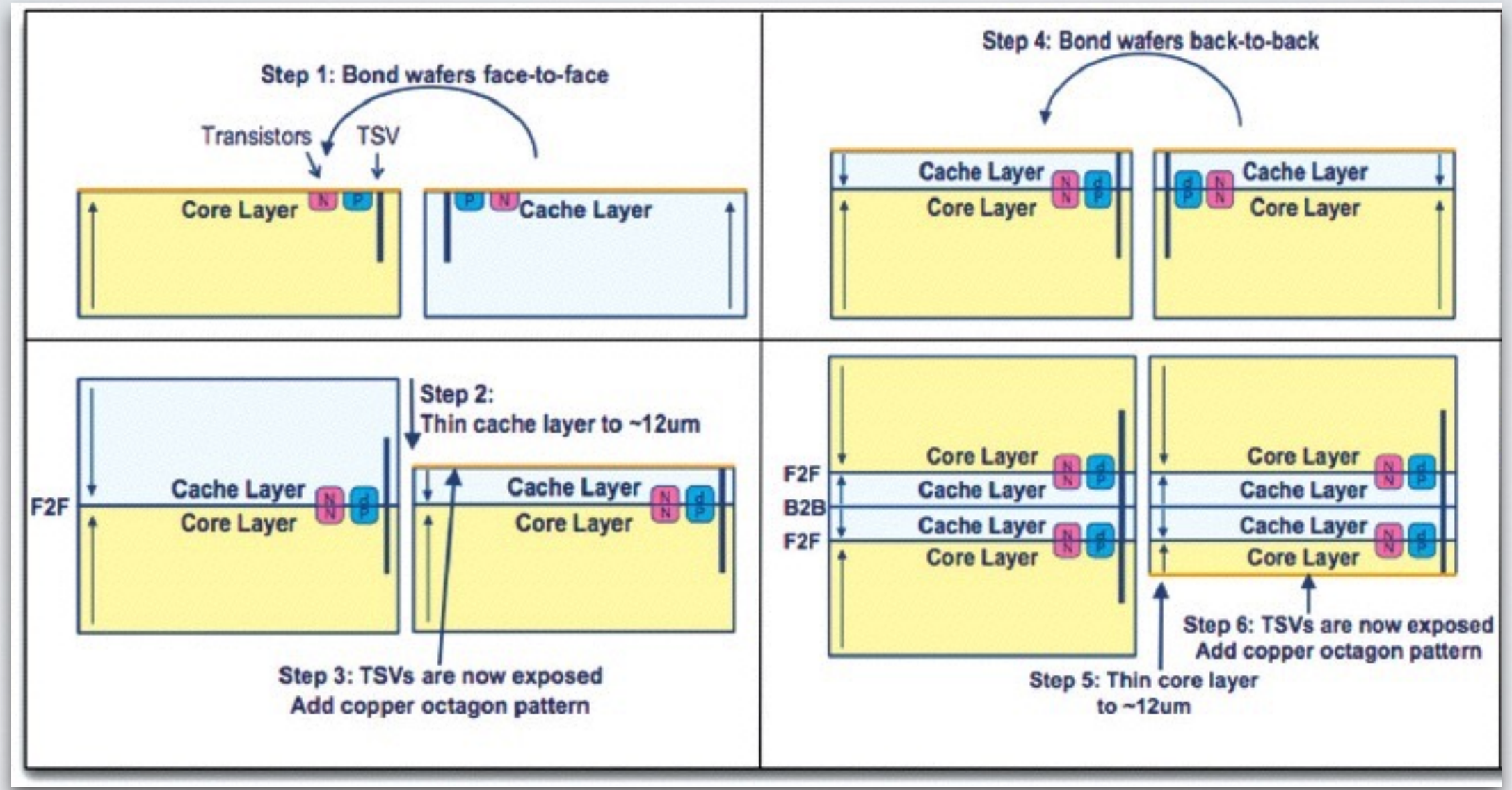
Two Dies

CENTIP3DE



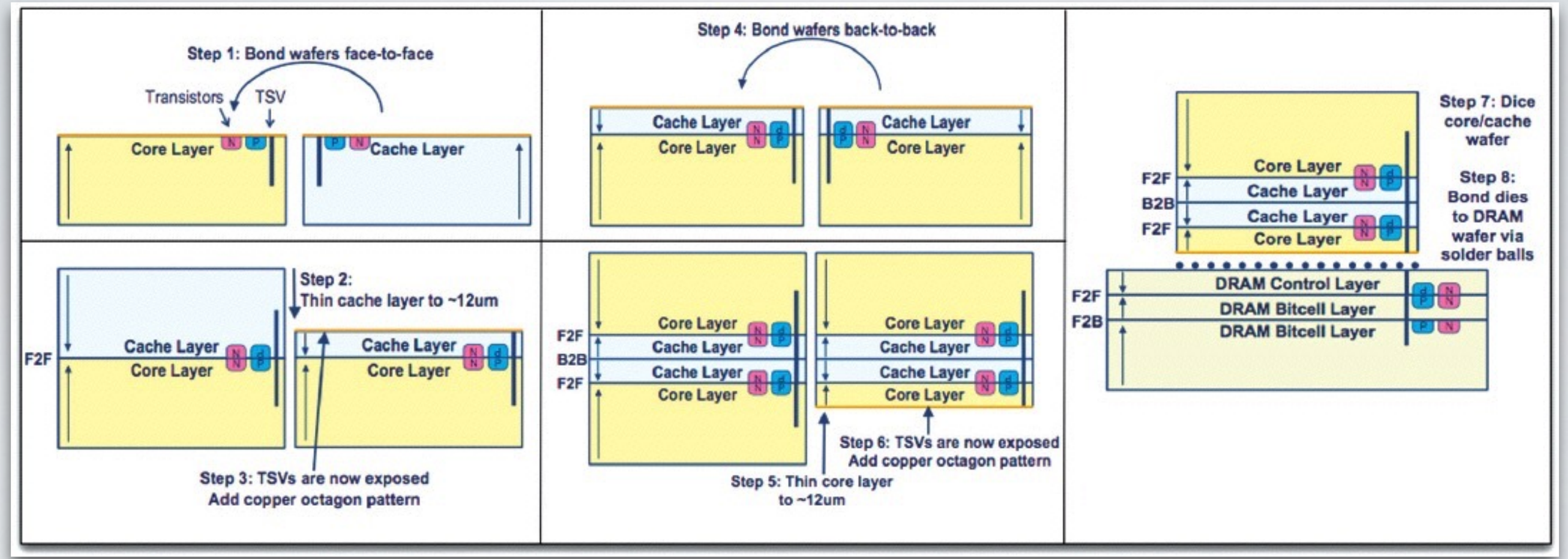
Dreslinski, R.G et al: Centip3De: A 64-Core, 3D Stacked Near-Threshold System

CENTIP3DE

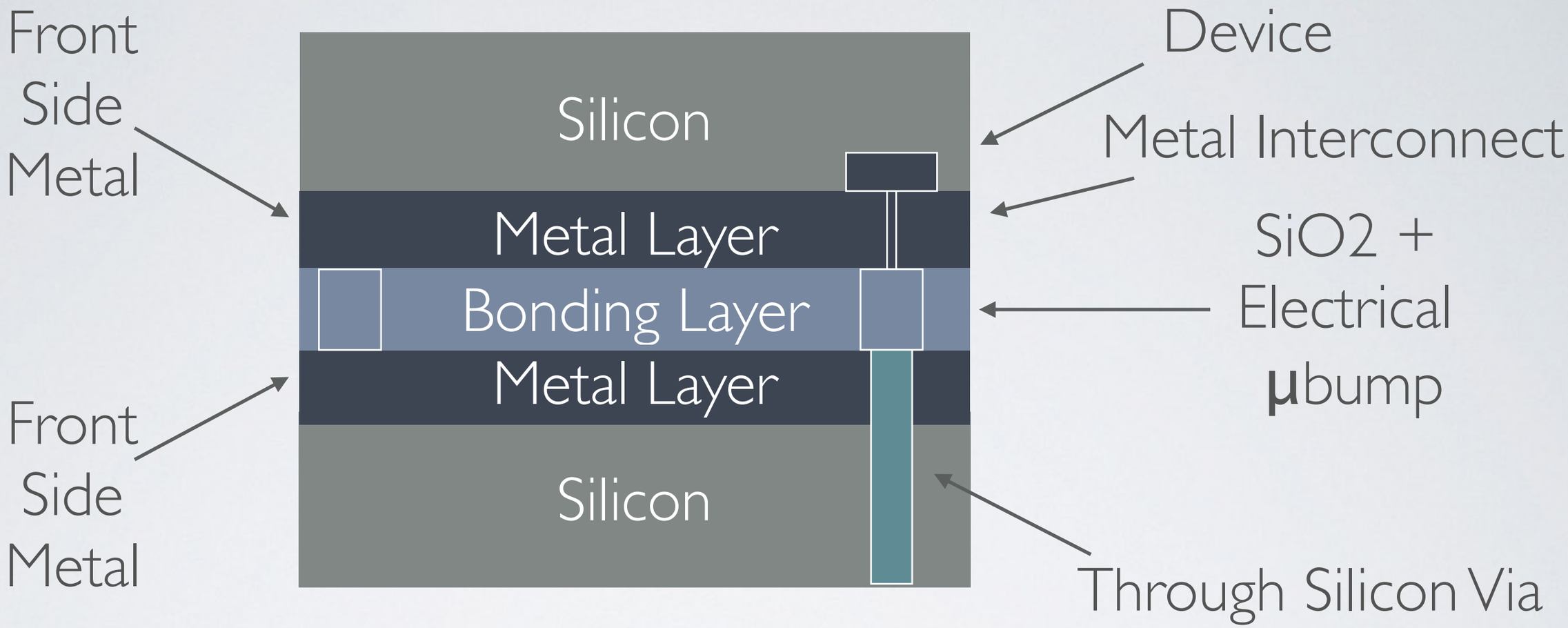


Dreslinski, R.G et al: Centip3De: A 64-Core, 3D Stacked Near-Threshold System

CENTIP3DE



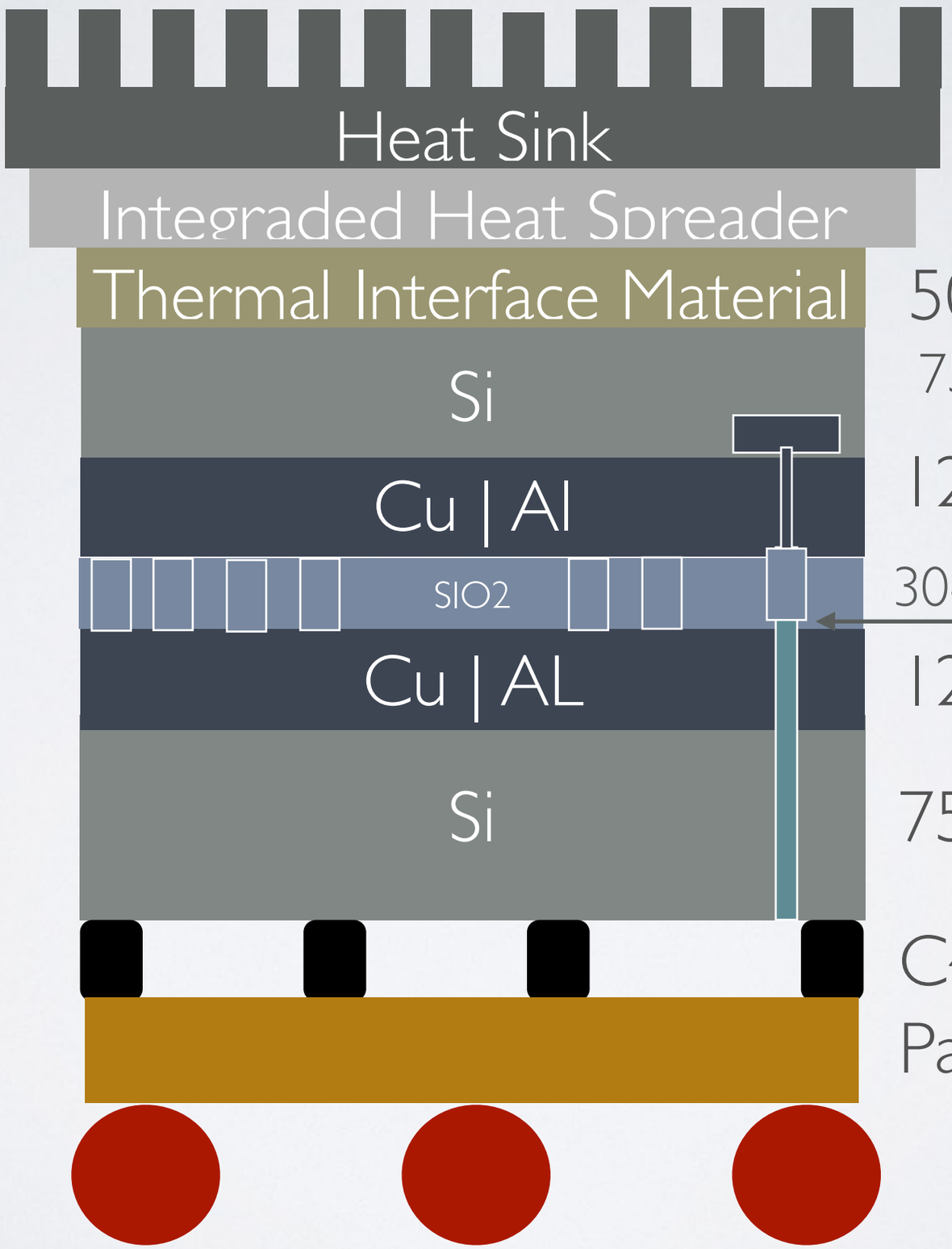
Dreslinski, R.G et al: Centip3De: A 64-Core, 3D Stacked Near-Threshold System



TSV-Based 3D Die-Stacking Face-to-Face

MODELING

TSV 3D Die-Stacking
Face-to-Face



6x6x0.7cm^3
3x3x0.1cm^3
50 µm
75µm
12µm
30-1µm nm TSV
12µm
75µm
C4 pads
Package Substrate
BGA

3D BENEFITS

- Reduced interconnect length and power
- Smaller form factor
- Heterogeneity
- New micro-architectural possibilities

PARALLEL INTEGRATION

- Fabricate each die separately
- Use traditional fabrication process
- Plus an extra thinning process
- Connect the dies

PARALLEL 3D

- Die-to-die stacking
- Face-to-face: active layers facing each other
- Back-to-back: bulk layers facing each other
- Face-to-back: active layer of one facing the bulk of the other

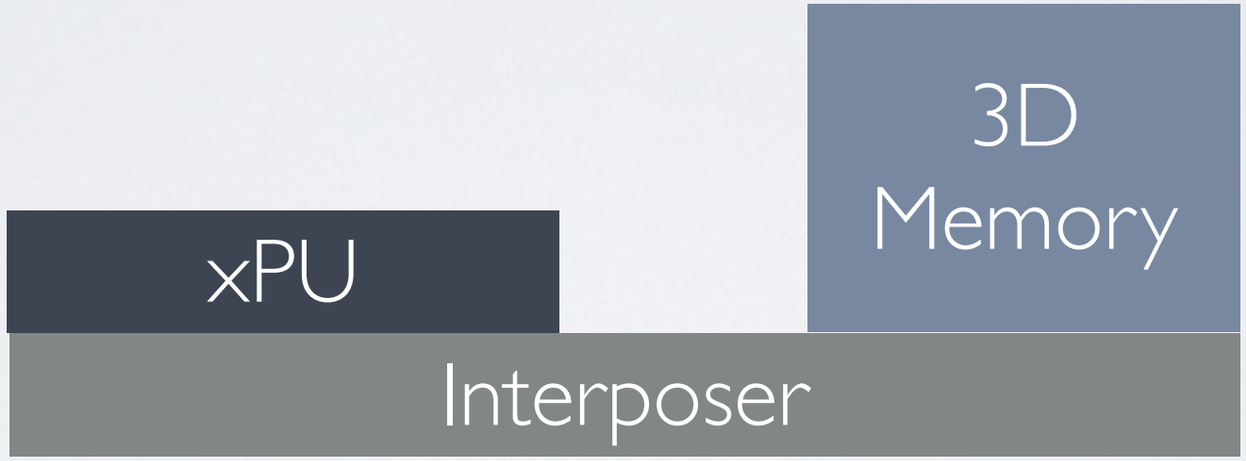
THERMAL ISSUES

- Bonding layer required for stress related issues
- Bonding Layer (underfil) = 3μm
- Impedes heat flow from layer 0 to layer1
- Thermal Conductivity BCB = 0.29 W/m-K
- E.g air = 0.03W/m-K silicon 140 W/m-K

TSV ISSUES

- Through-Silicon Via (TSV) = 30-1μm
- Copper(Cu) or Tungsten (W)
- Used to connect the layers
- We want high density of TSVs (more connections)
- Technology Constrained (KOZ + Aspect Ratio)

WHAT DO WE HAVE NOW?



As of June/July 2015 Radeon R9 Fury : Fiji Pro

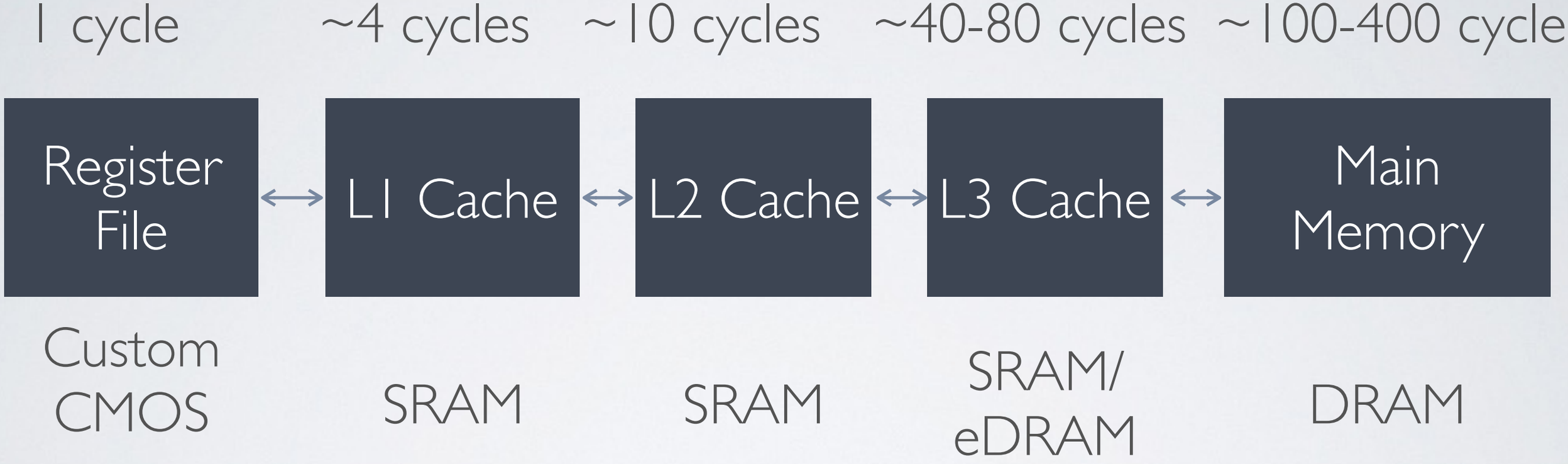
2.5D is the flavor of the month

Breaking The Memory Wall

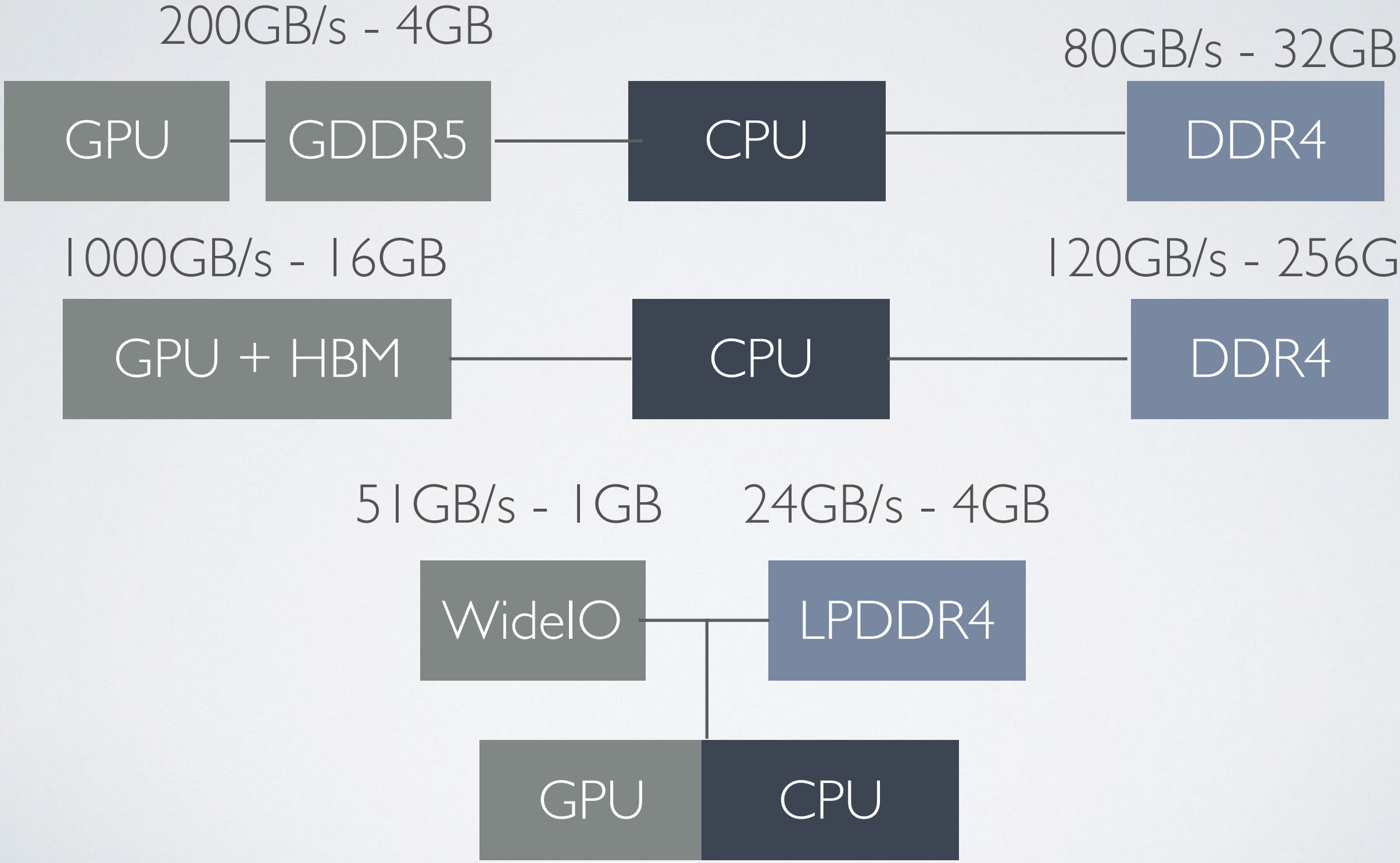
CHALLENGES OF MEMORIES

- Satisfy Bandwidth Requirements
- Reduce Power Consumption
- Low Cost

LATENCY


1 cycle
~4 cycles
~10 cycles
~40-80 cycles
~100-400 cycles
Register File
L1 Cache
L2 Cache
L3 Cache
Main Memory
Custom CMOS
SRAM
SRAM
SRAM/ eDRAM
DRAM

RANDOM ACCESS MEMORIES



WHAT DO WE DO WITH SO MUCH MEMORY?

- Use it as a huge cache

- Use it as part of memory

ARCHITECTING DRAM CACHES

- Tag Storage
- Hit Latency
- Handle misses efficiently

3D DRAM AS CACHE

- Low lookup latency
- High hit rate
- Efficient off-chip BW use
- Data-granularity: page (4KB) vs block (64B)

BLOCK BASED - ALLOY CACHE

- 64B block
- Low off-chip BW utilization
- Low locality of data
- Store tags in the DRAM,
- Tag management becomes a problem

Moinuddin K. Qureshi et al. Fundamental Latency Trade-offs in Architecting DRAM Caches

BLOCK BASED - ALLOY CACHE

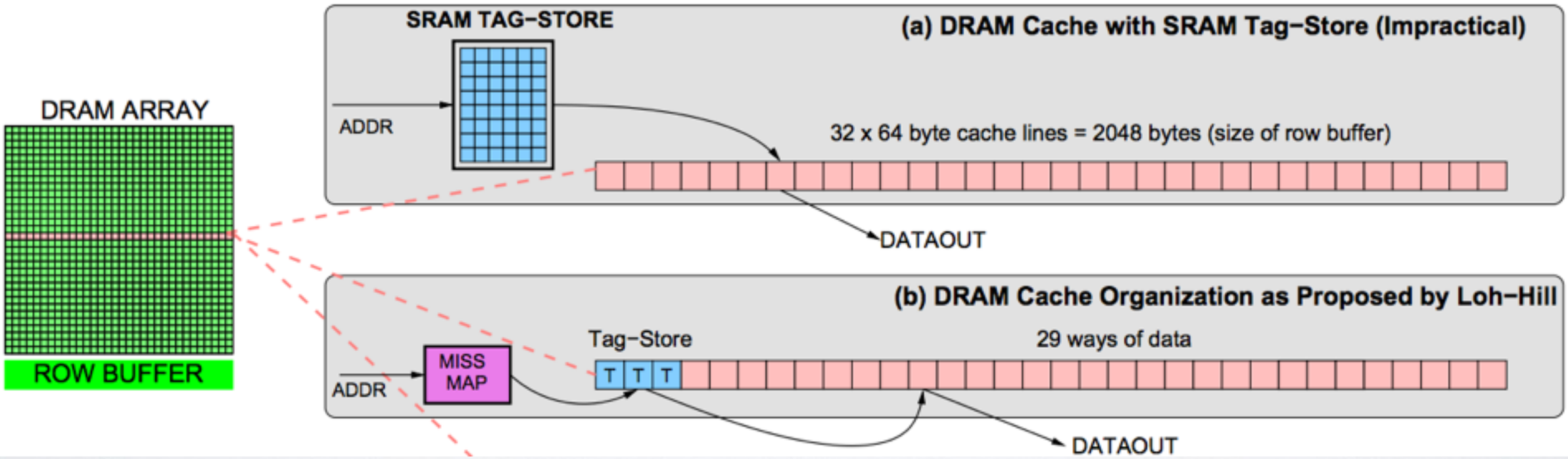
- Store tags in SRAM is prohibitive (24MB for 256MB DRAM cache)

- Store tags in DRAM -> 2x the access time, 1 for the tag 1 for the data (Tag Serialization Latency)

- Solution: Store the tags with the data in the same Row

Moinuddin K. Qureshi et al. Fundamental Latency Trade-offs in Architecting DRAM Caches

BLOCK BASED - ALLOY CACHE



Moinuddin K. Qureshi et al. Fundamental Latency Trade-offs in Architecting DRAM Caches

BLOCK BASED - ALLOY CACHE

- MissMap keeps track of lines in the DRAM
- On miss go to off-chip without tag access
- Several MBs -> Place it in L3
- Access MissMap on every L3 miss
- Predictor Serialization Latency (PSL)

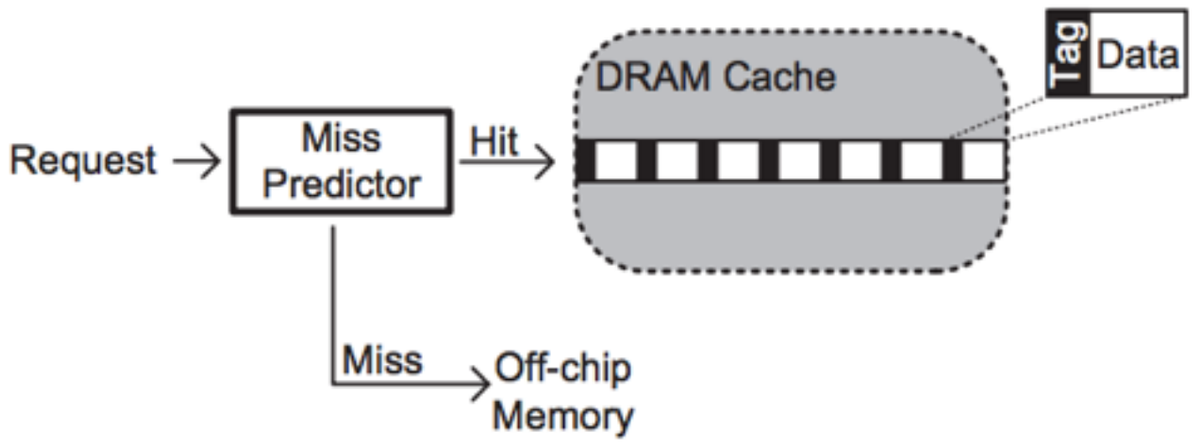
Moinuddin K. Qureshi et al. Fundamental Latency Trade-offs in Architecting DRAM Caches

BLOCK BASED - ALLOY CACHE

- More Acronyms
- Alloy Cache tightly alloys tag and data into a single entity called TAD (Tag and Data).
- Access MissMap and DRAM in parallel

Moinuddin K. Qureshi et al. Fundamental Latency Trade-offs in Architecting DRAM Caches

PAGE BASED - FOOTPRINT CACHE



(a) Block-based cache collocating tags with data blocks

LH Cache

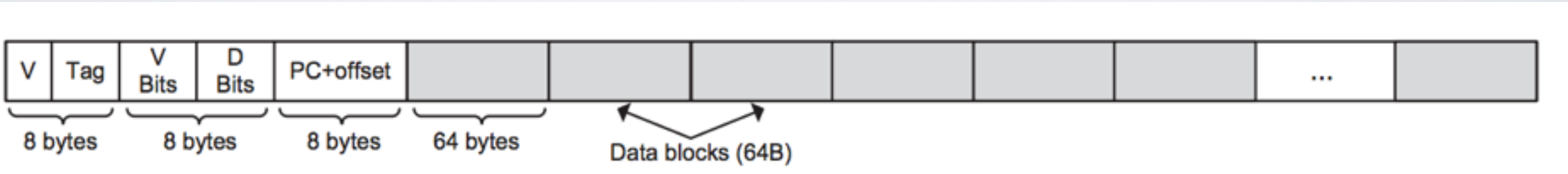
D. Jevdjic et al. Die-Stacked DRAM Caches for Servers
Hit Ratio, Latency, or Bandwidth? Have It All with Footprint Cache

PAGE BASED - FOOTPRINT CACHE

- Page granularity 4KB
- Fetch only the blocks that are likely to be touched in a page
- Page Allocation & Block Fetching
- Spatial Correlation Predictor (trigger prefetching and store the metadata(PC+Offset) for later)

PAGE BASED - UNISON CACHE

- Merge Alloy cache ideas with Footprint cache



D. Jevdjic et al. Unison Cache: A Scalable and Effective Die-Stacked DRAM Cache

OVERVIEW

	Alloy Cache	Footprint Cache	Unison Cache
Cache Miss Rate	Medium-High	**Low**	**Low**
Hit Latency	**Predictor + DRAM TAD Read**	SRAM Tag + DRAM Data Read	**Overlapped DRAM Tag + Data Reads**
Miss Latency	**Predictor Lookup**	SRAM Tag Lookup	DRAM Tag Lookup
Associativity	Direct-mapped	**32-way**	4-way (two pages)
64B Blocks per 8KB Row	112	**128**	120-124
SRAM Tag Array @ 8GB	—	~48MB	—
In-DRAM Tag Size @ 8GB	1GB (12.5% of DRAM)	—	256-512MB (3.1-6.2% of DRAM)
Miss-Predictor Size	96B per core, 1.5KB total	—	—
Way Predictor	—	—	1-16KB
Footprint History Table	—	144KB	144KB
Singleton Table	—	3KB	3KB

D. Jevdjic et al. Unison Cache: A Scalable and Effective Die-Stacked DRAM Cache

PART OF MEMORY (POM)

- Use the stacked DRAM as part of memory
- Fast memory (3D) - Slow memory (Off-chip)
- OS usage monitoring and managing pages
- Proposal: Hardware managed pages

J. Sim et al. Transparent Hardware Management of Stacked DRAM as Part of Memory

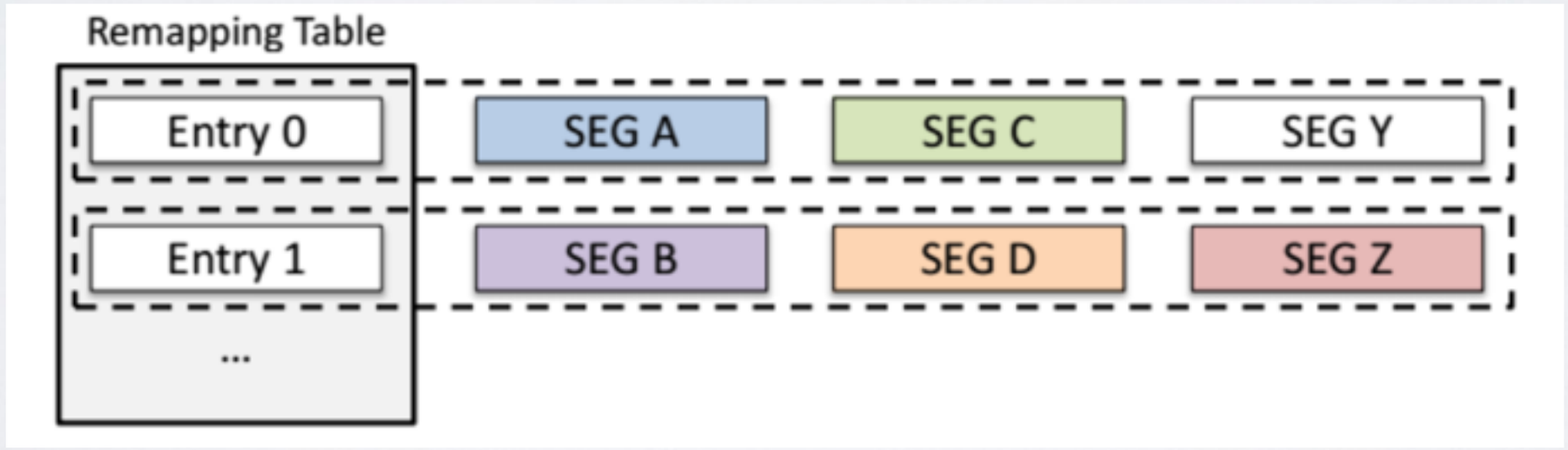
PART OF MEMORY (POM)

- Single address space
- Two-level indirection with remapping cache
- On request check segment remapping cache (SRC)
- On miss fetch from segment remapping table (SRT)
- On hit fetch the data from its location and update SRC

J. Sim et al. Transparent Hardware Management of Stacked DRAM as Part of Memory

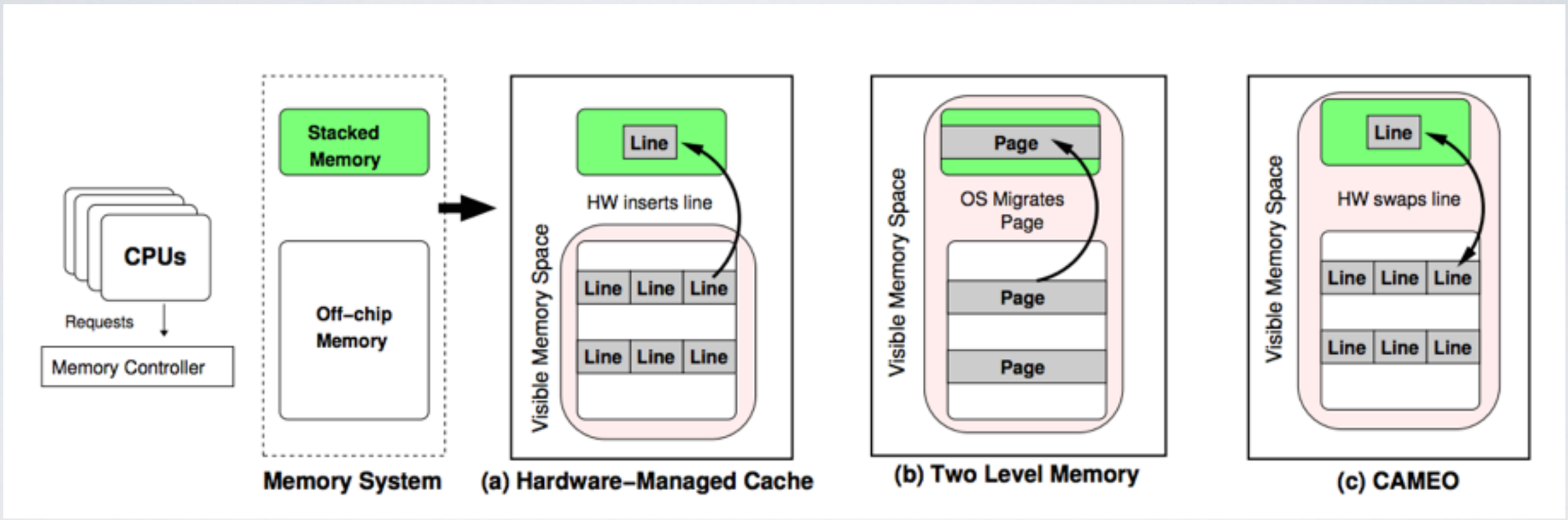
PART OF MEMORY (POM)

- On miss: access SRC, access SRT, search SRT
- Segment-restricted remapping (page table physical address based) similar to Direct-Mapped Cache



J. Sim et al. Transparent Hardware Management of Stacked DRAM as Part of Memory

CAMEO

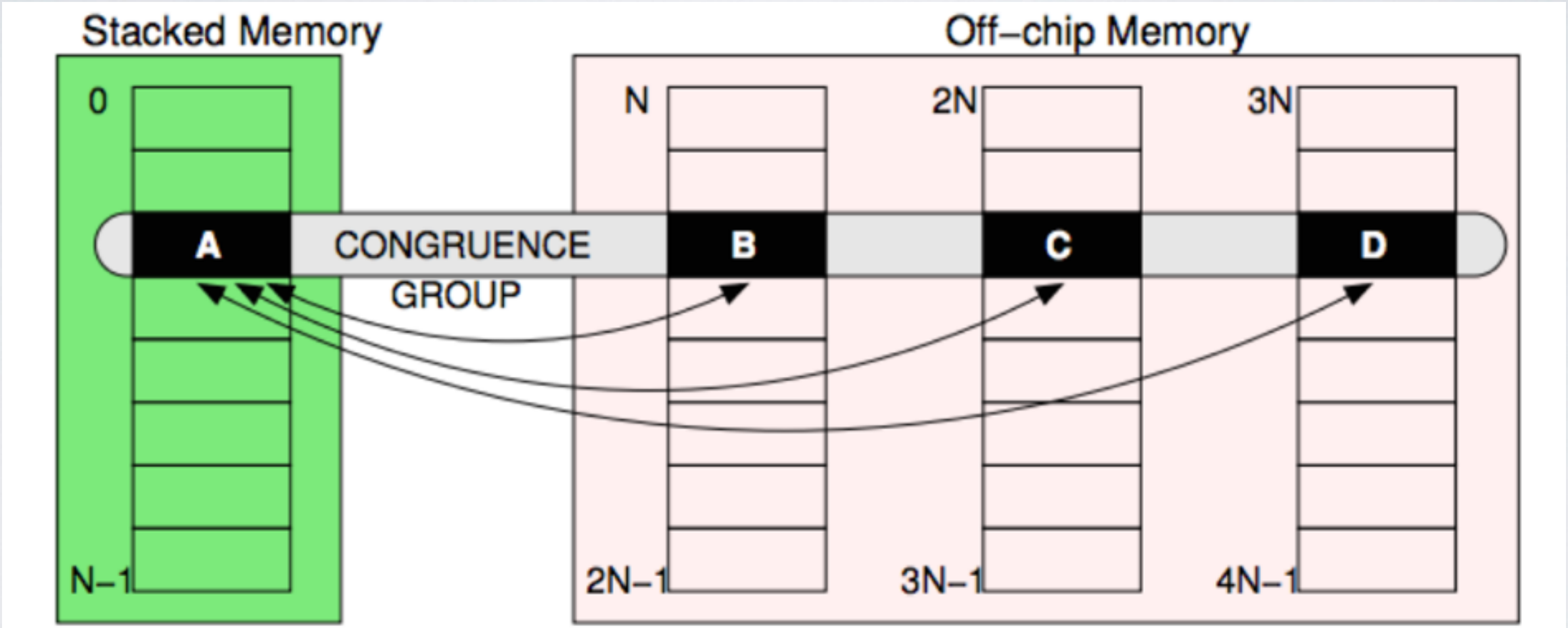


Line Location Table (LLT) tracks the physical location of memory lines

Line Location Predictor (LLP) predicts the physical address of the cache line

C Chou et al. CAMEO: A Two-Level Memory Organization with Capacity of Main Memory and Flexibility of Hardware-Managed Cache

CAMEO

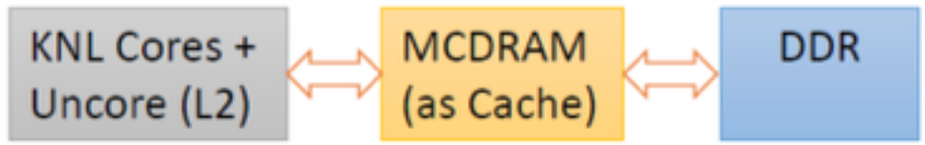


C Chou et al. CAMEO: A Two-Level Memory Organization with Capacity of Main Memory and Flexibility of Hardware-Managed Cache

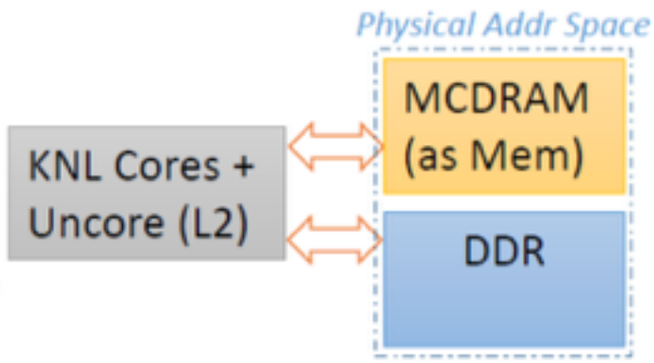
WHAT DO WE HAVE NOW?

HBM Modes

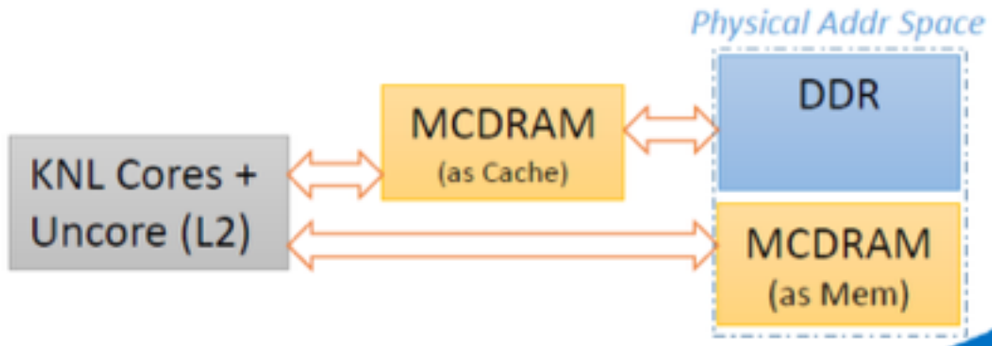
- Cache mode
  - No source changes needed to use
  - Misses are expensive (higher latency)
    - Needs HBM access + DDR access



- Flat mode
  - MCDRAM mapped to physical address space
  - Exposed as a NUMA node
    - Use numactl --hardware, lscpu to display configuration
  - Accessed through memkind library or numactl



- Hybrid
  - Combination of the above two
    - E.g., 8 GB in cache + 8 GB in Flat Mode



SUMMARY

- 3D Die stacking is happening (Intel, AMD, nVIDIA)
- How do we use all this memory ***efficiently*** is still an open question!!
- New architecture and microarchitecture paradigms