



A brief introduction to Memory system proportionality and resilience

Mattan Erez

The University of Texas at Austin



With support from



– NSF Award #0954107



The **constraints**:

- Power/energy
- Time
- Money
- Correctness



Can't work harder –
have to work smarter



Can't work harder –
have to work smarter

Efficient & Proportional



Multiple constraints and needs →

Heterogeneity

asymmetry

specialization



big.LITTLE and per core DvFS

– Static and **dynamic** asymmetry

Accelerators

– **Programmable** and fixed-function



Throughput cores

Latency cores

Specialized cores

→ **proportionality of compute**



Great, we solved the easy part:
proportional compute



Programming is hard
Memory is hard



The memory challenge (outline)

- Why do memory systems look the way they do?
 - Efficiency and reliability
 - Abstractions for architects
- Role of heterogeneity and programs
- Some interesting mechanisms
- Where do we go from here?



Storage **device** options abound

- SRAM
- DRAM
- FLASH
- STT-/M- RAM
- PCM, Memristor, RRAM, ...



Architects should think in **abstractions**
– Research should be general



The **fundamentals**

Cheap

Energy efficient

High capacity

Low latency

High throughput

Reliable



Memory must be **cheap**

- Memory is already too expensive
 - 15 – 50% of system cost (or more)
- But, margins razor thin (commodity)



Memory must be **energy efficient**

- Memory accounts for 15 – 60+% of “compute” energy and getting worse
 - More capacity
 - Proportional compute



Memory must be **high capacity**

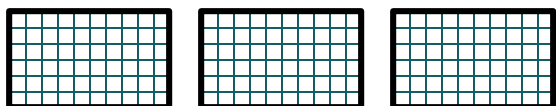
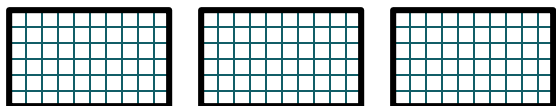
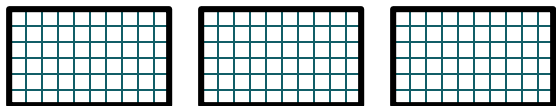
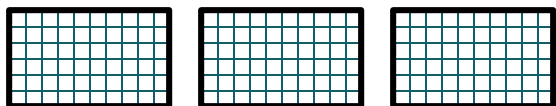
– Memory footprints keep growing

- More interesting data
- More data
- More specialized data structures



Memory must be **high capacity**

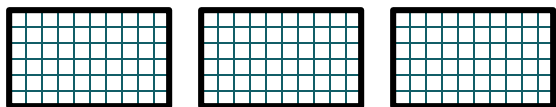
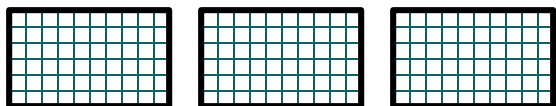
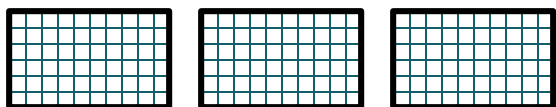
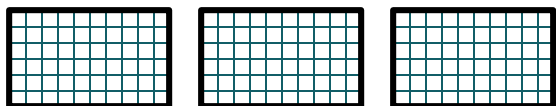
Many chips



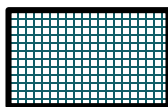
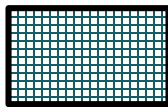
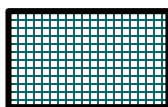
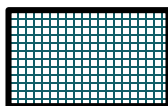


Memory must be **high capacity**

Many chips



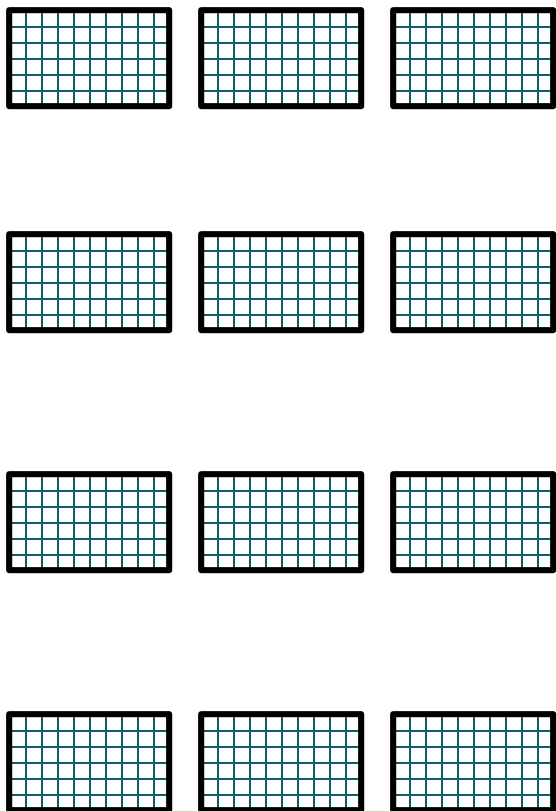
**Denser mem,
Fewer chips**



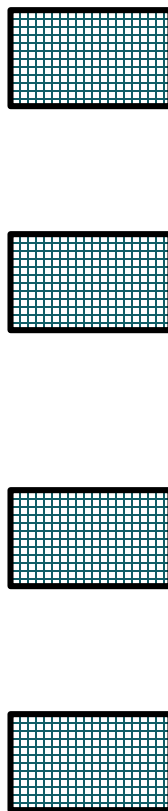


Memory must be **high capacity**

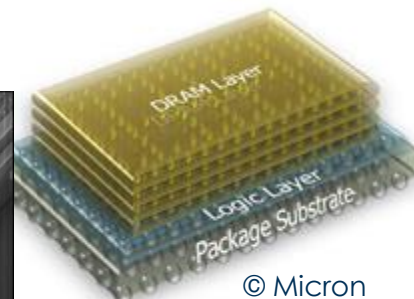
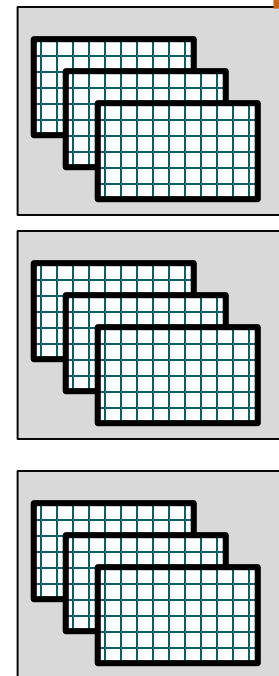
Many chips



**Denser mem,
Fewer chips**



**Many dice,
Fewer chips**





Memory composed of

- Multiple modules
- Each module with $N \geq 1$ components
- Lots of cells per component
- Each cell ≥ 1 bit



Memory should be **low latency**

- Latency == performance
- Except when sufficient concurrency



Low latency **conflicts** with

- Cheap
- Capacity
- Sometimes with
 - Throughput
 - Energy
 - Reliability



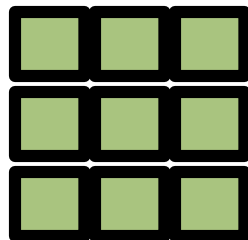
Latency components:

- Memory cell access
- Data transfer
- Queuing

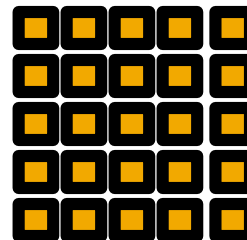


Latency impacted by cell access **denser/efficient → higher latency**

- Small cells → sensitive circuits → slow reads
- Writes even more complicated



- + latency**
- density**
- cost**
- energy**



- latency**
- + density**
- + cost**
- + energy**



Caching to the rescue

- Store some data somewhere fast



Short distances improve latency
– It's all about the wires



Short distances improve latency

– It's all about the wires



+ latency
+ energy

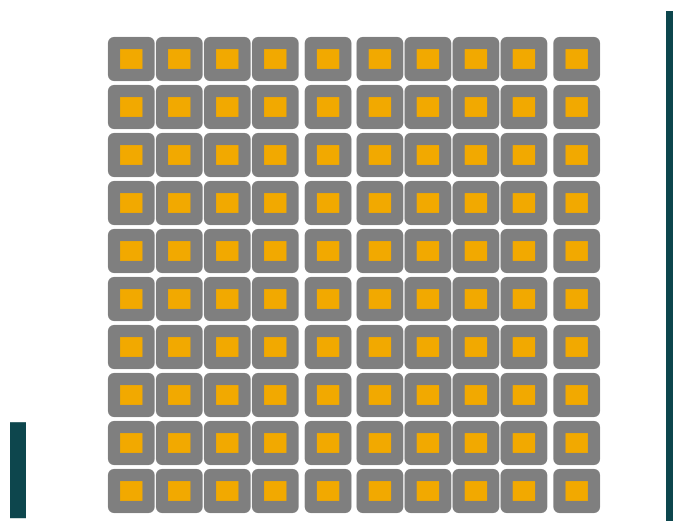


– latency
– energy



Short distances improve latency

– It's all about the wires



+ latency
+ energy
– capacity

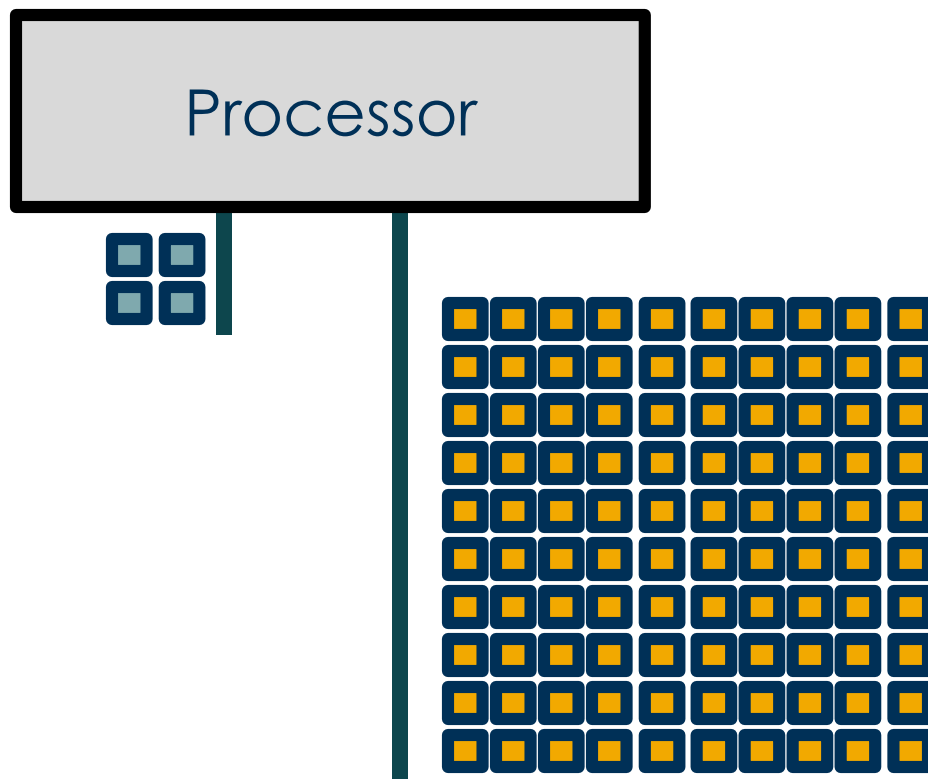
– latency
– energy
+ capacity



Hierarchy to the rescue

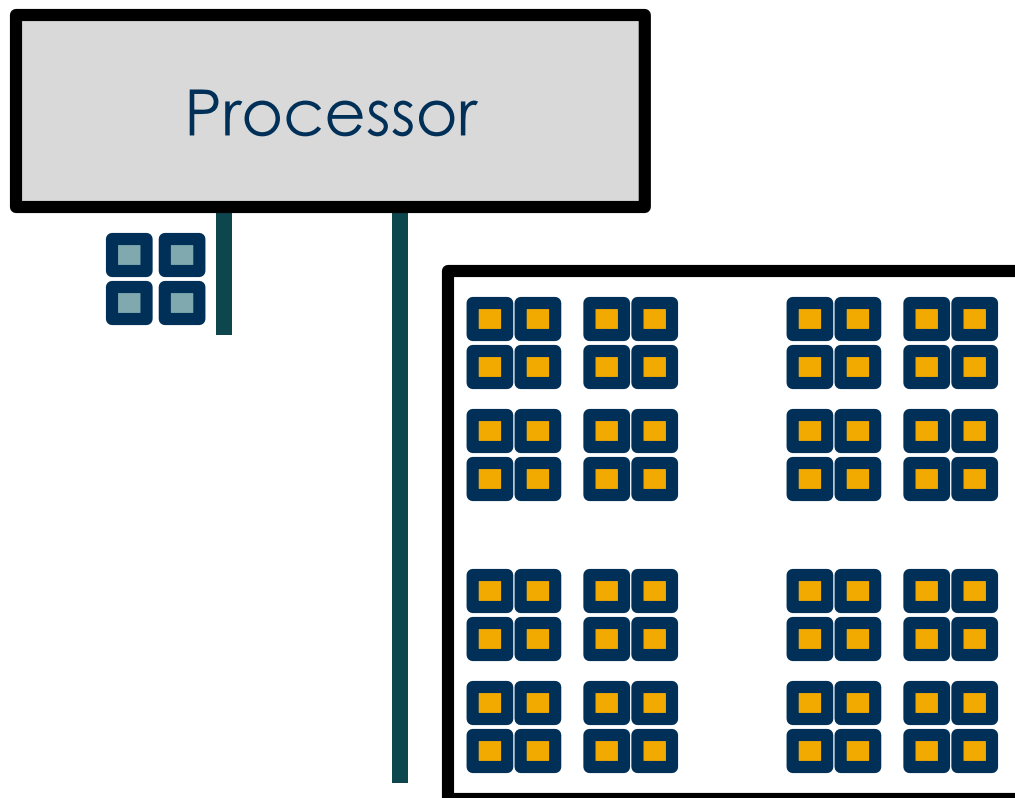


Hierarchy to the rescue





Hierarchy to the rescue



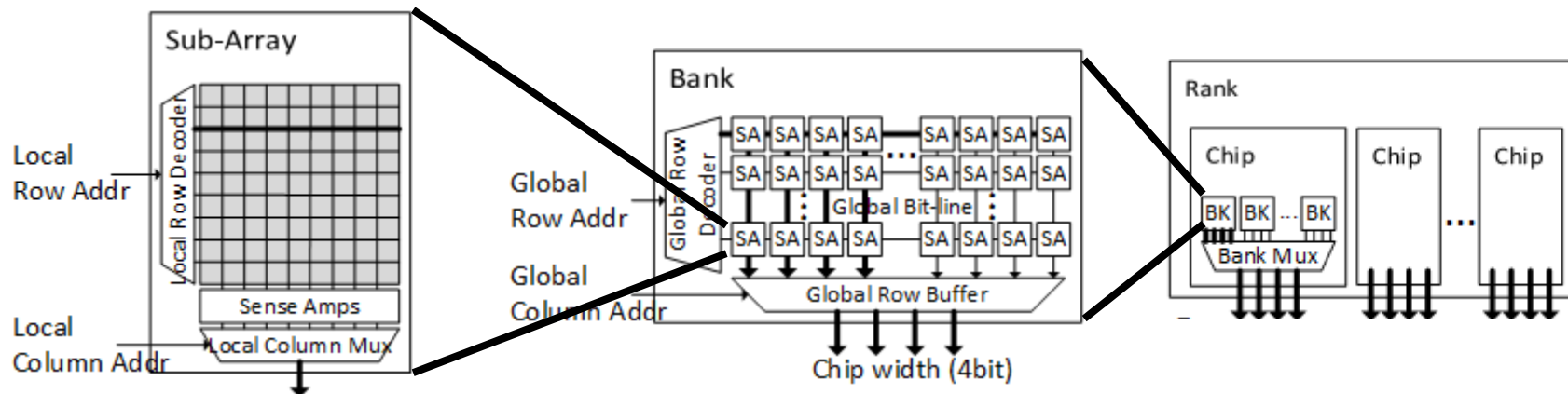


Memory system

- Modules, packages, components
- Arranged in a hierarchy

Each memory component

- Has hierarchy
- Has caching/buffering



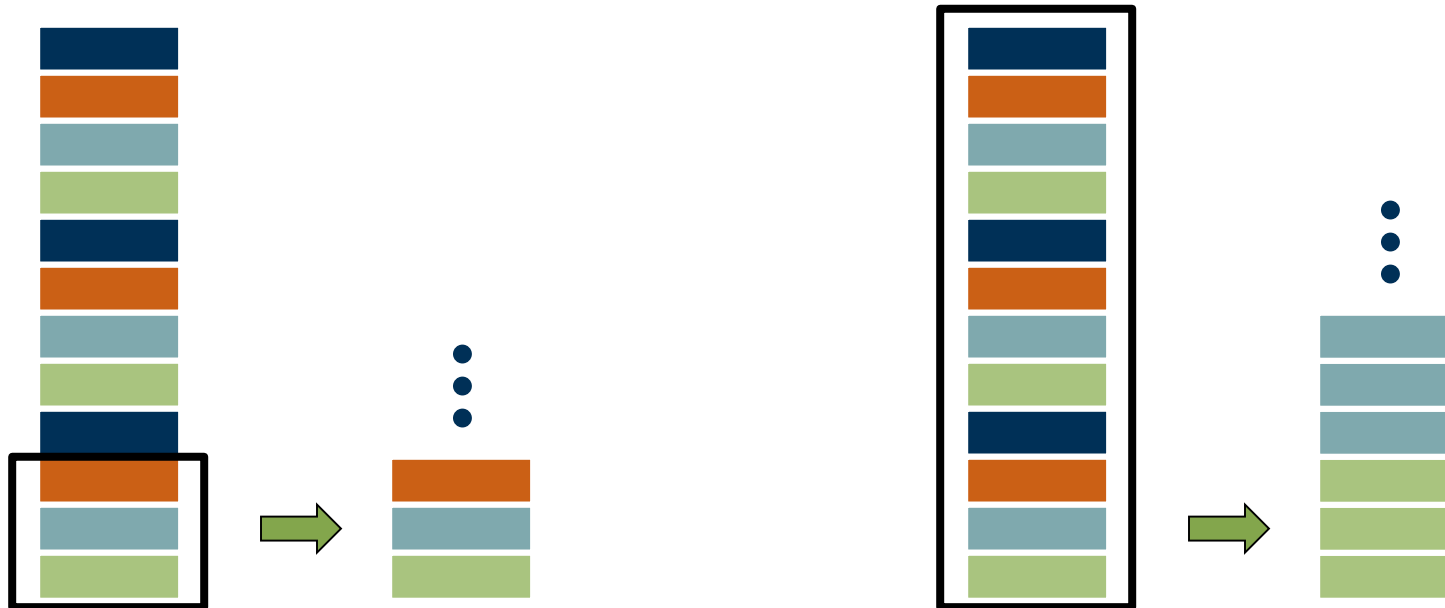


Latency impacted by **queuing**



Deep queues improve **locality**

- Higher throughput
- Higher efficiency
- Often hurts latency



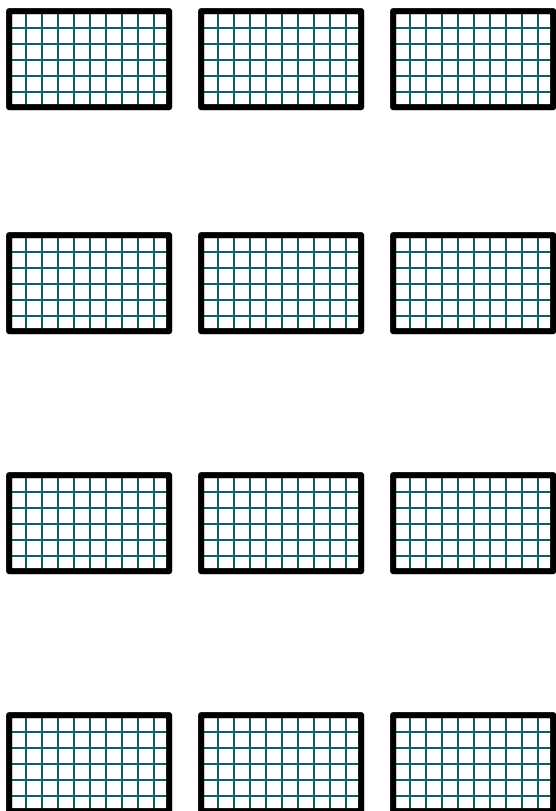


Memory must be **high throughput**
– More and more latency hiding

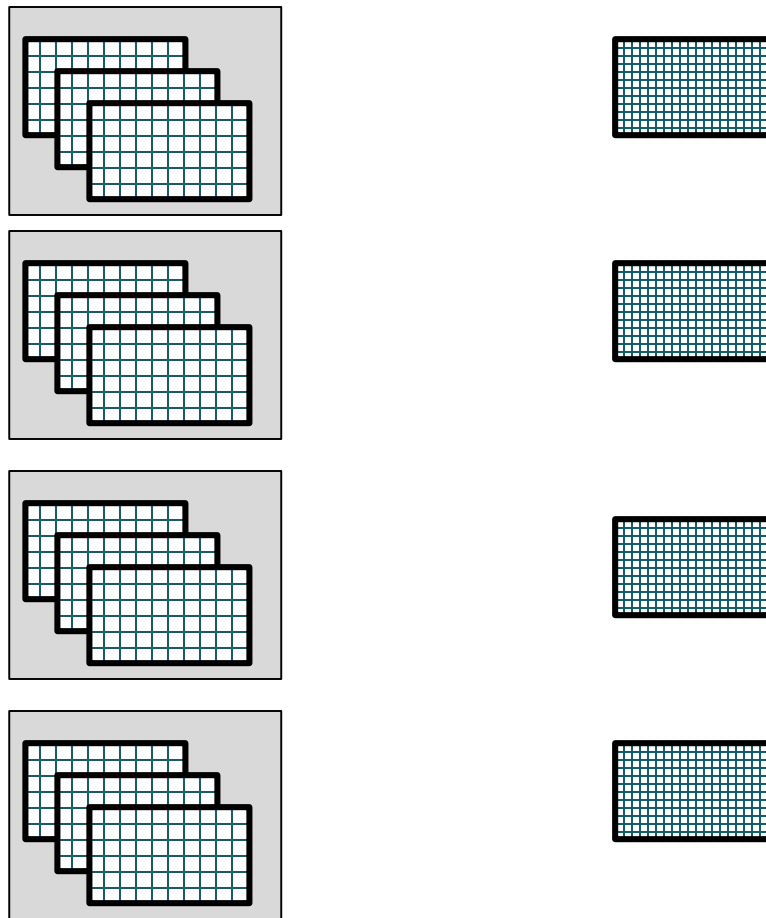


Memory must be **high throughput**

Many chips



Fewer chips

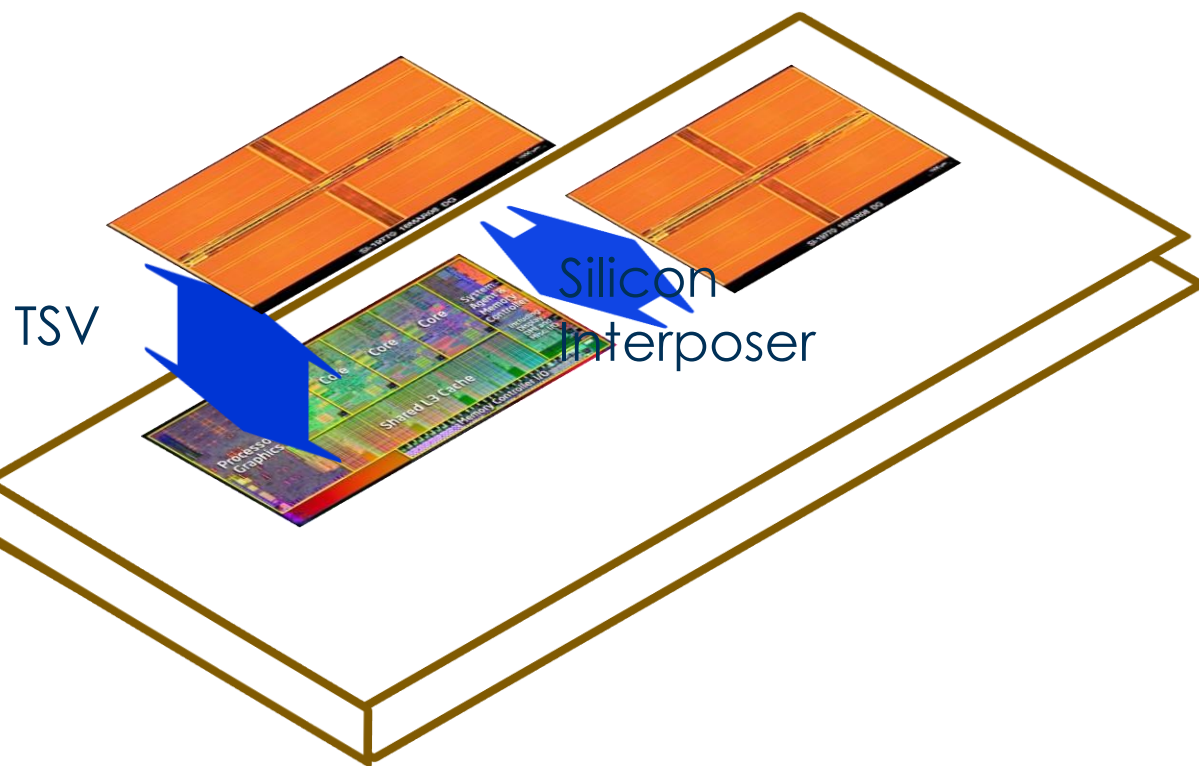


High capacity per pin requires efficient access



Packaging/interfaces improve throughput

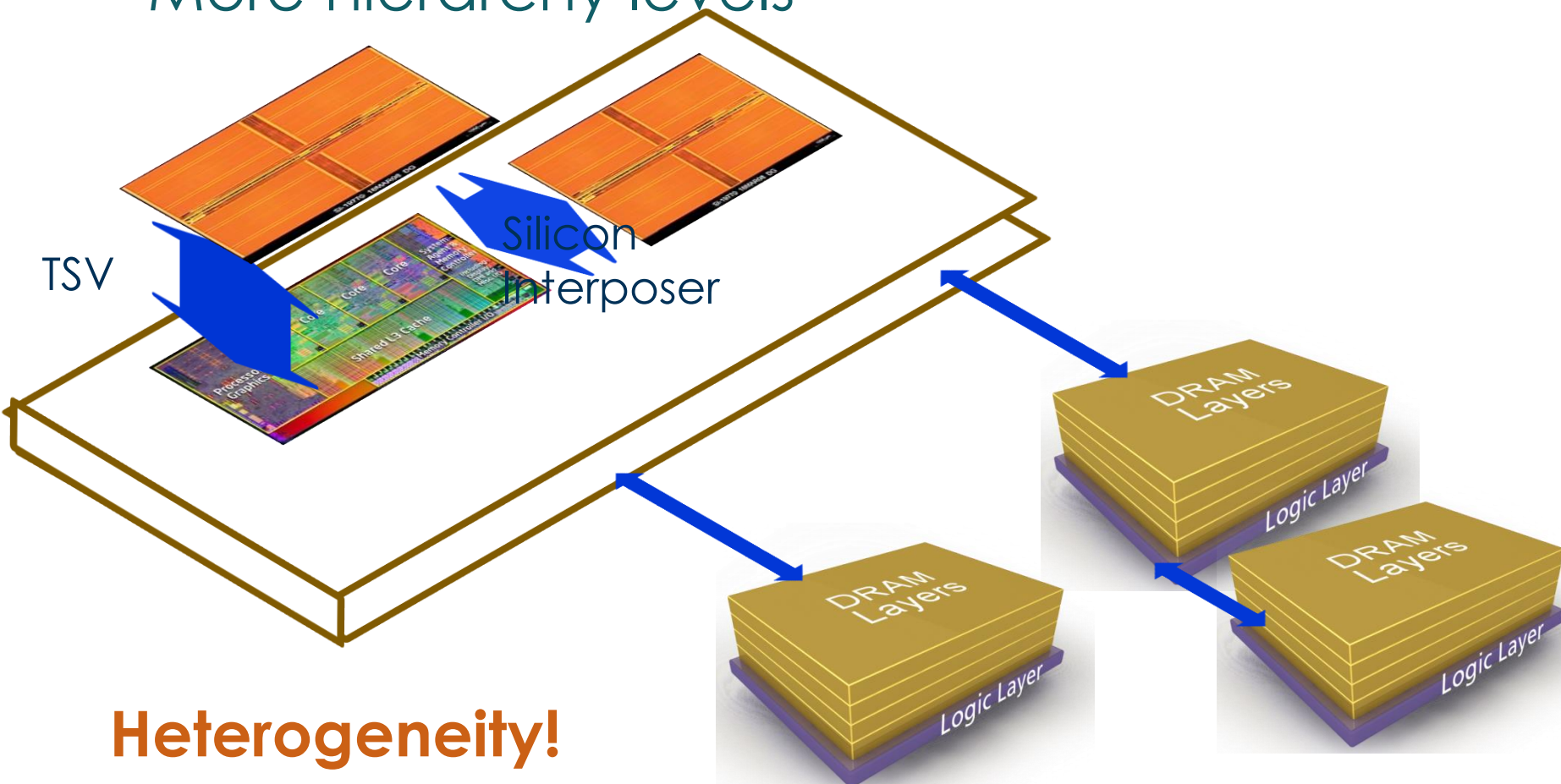
- 3D integration
- Limited capacity





Packaging/interfaces improve throughput

- 3D integration
- Limited capacity
- More hierarchy levels





Latency + throughput + efficiency
→ **parallelism**



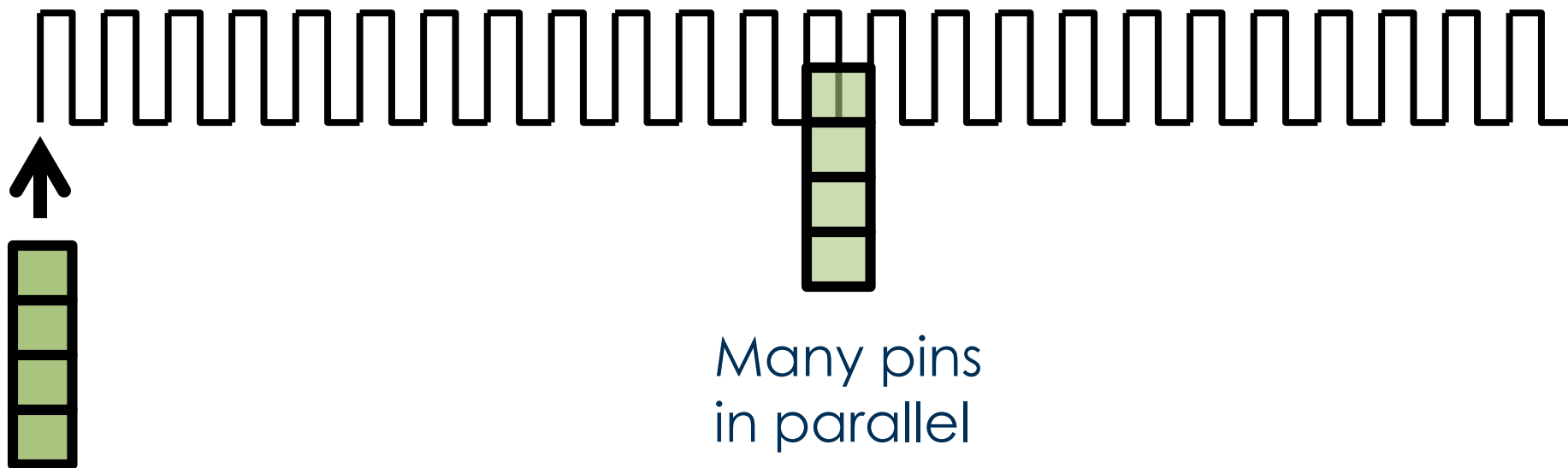
Latency + throughput + efficiency
→ **parallelism**



Access cell



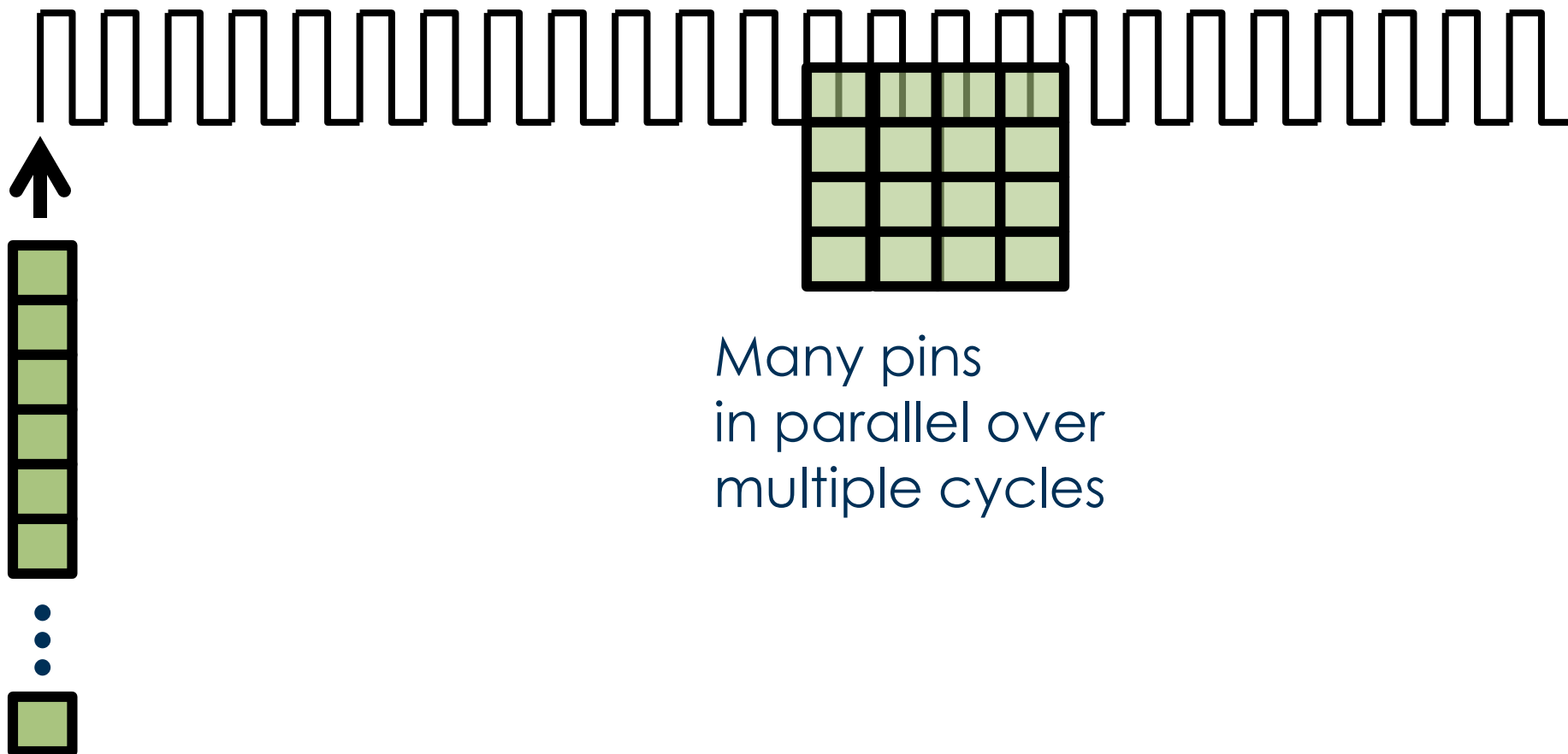
Latency + throughput + efficiency
→ **parallelism**



Access many
cells in parallel



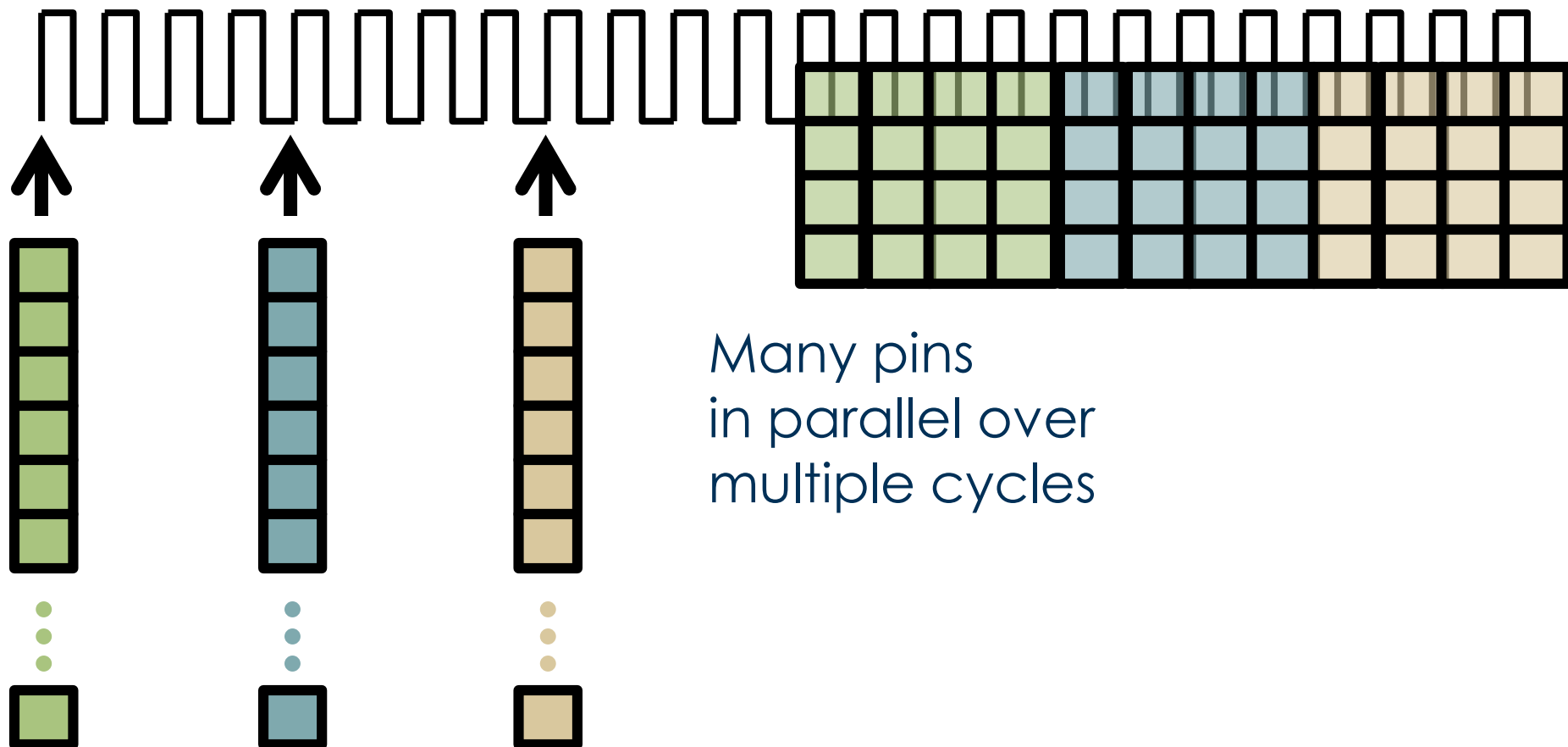
Latency + throughput + efficiency
→ **parallelism**



Access even more
cells in parallel



Latency + throughput + efficiency → **parallelism**



Access even more
cells in parallel



To recap

- Locality
- Hierarchy
- Parallelism



Memory must be **reliable**

- Written data must be recalled “correctly”

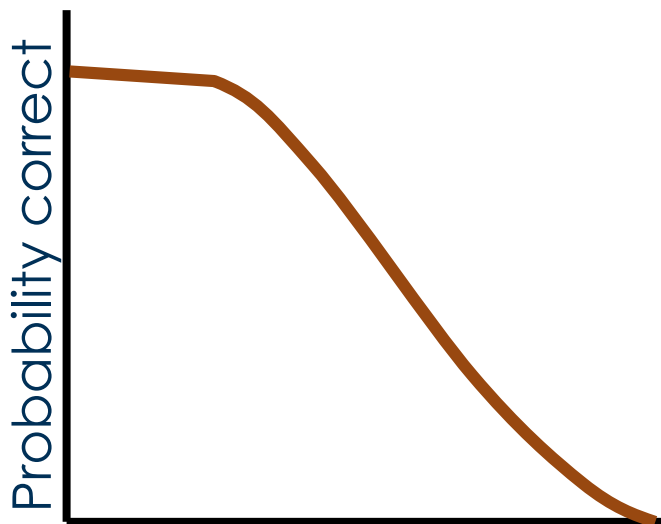


Reliability **challenges**

- “Natural” change in cell value
- Induced change in cell value
- Read errors
- Write errors
- Wearout and defects



Natural causes of value change

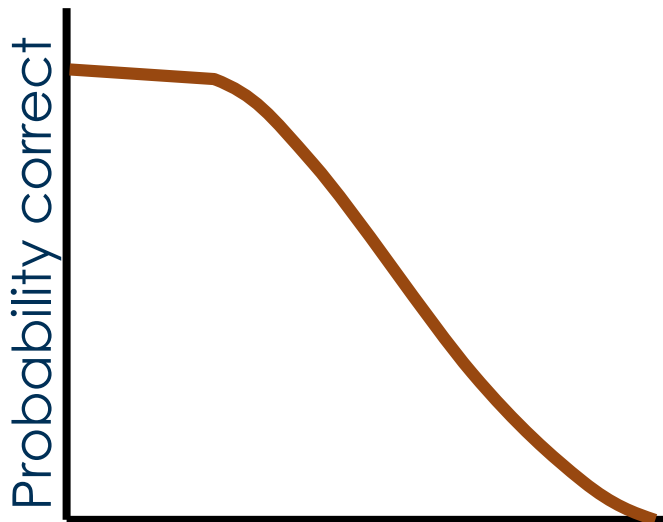


Decay Time

– Refresh

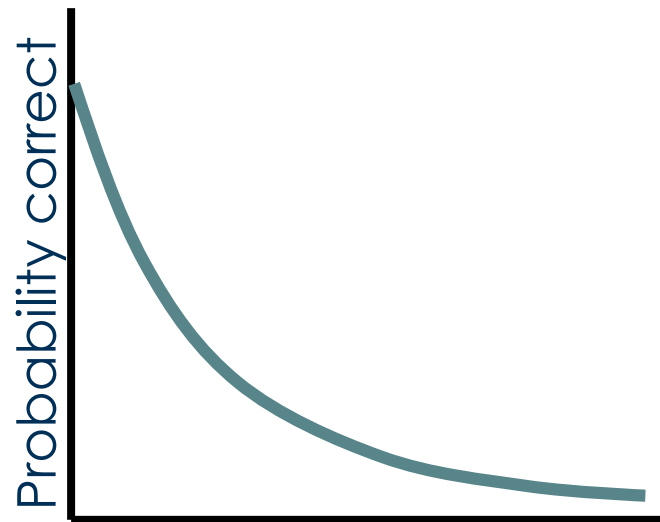


Natural causes of value change



Decay Time

– Refresh

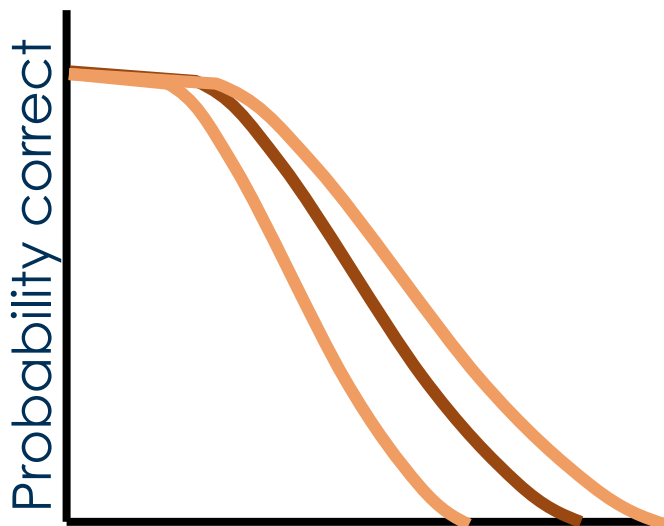


Flip Time

– ECC + scrub

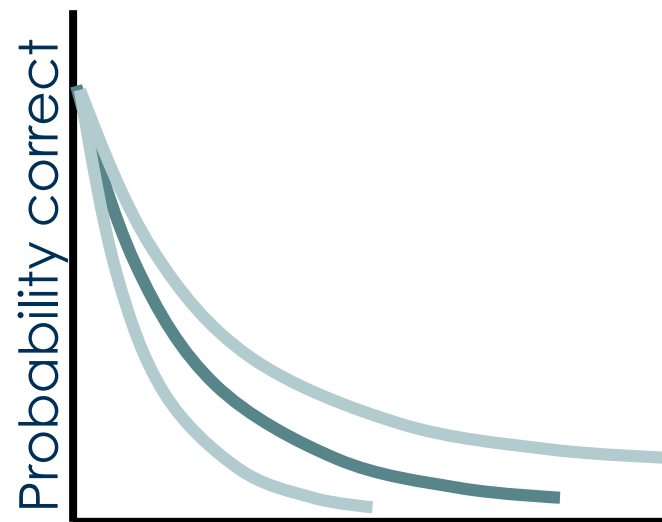


Natural causes of value change



Decay Time

– Refresh / scrub

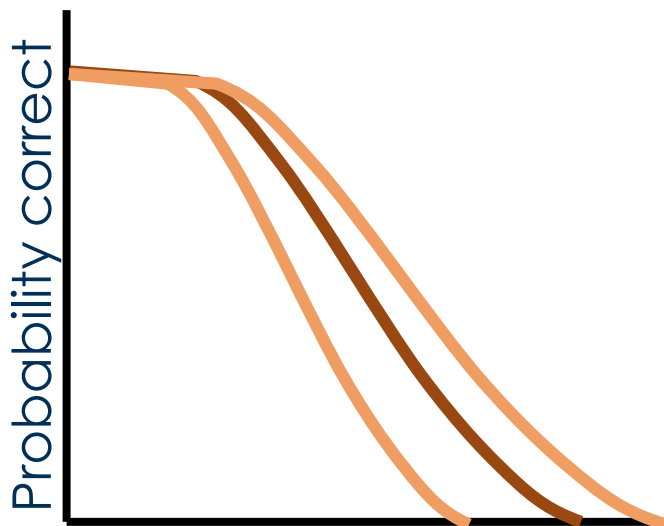


Flip Time

– ECC + scrub

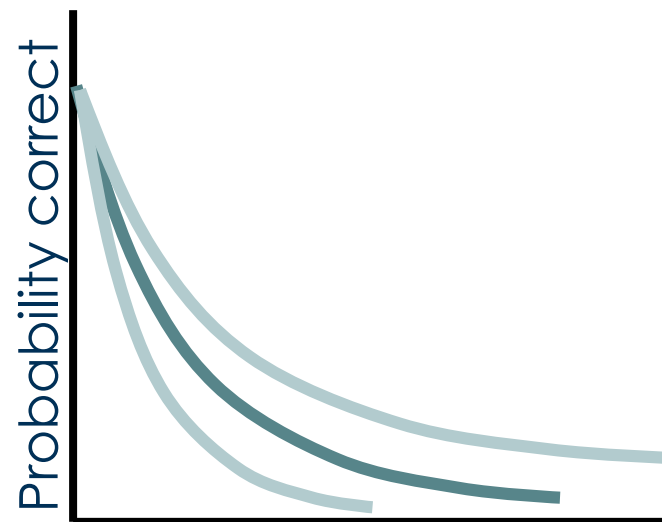


Natural causes of value change



Decay

– Refresh / scrub



Flip

– ECC + scrub

– Retention control

- Less dense
- Writes slower/higher-energy
- Can use different devices



Induced change in value

- Writing or reading disturbs nearby cells
- Worse for writes
- Technology and design dependent



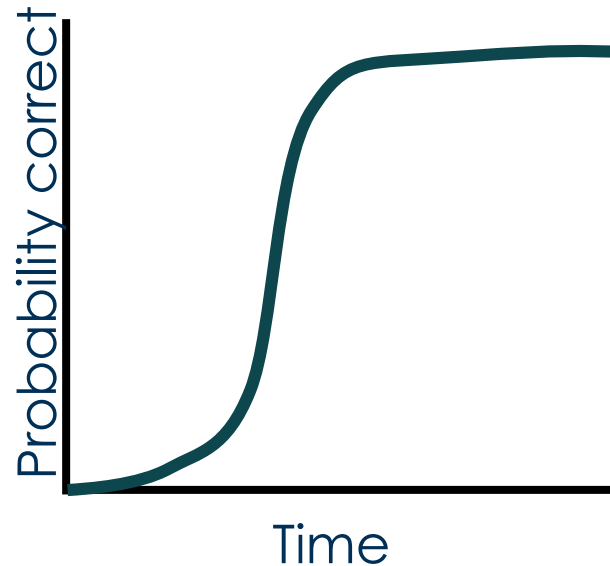
Read errors

- Because of small margins for efficiency



Write errors

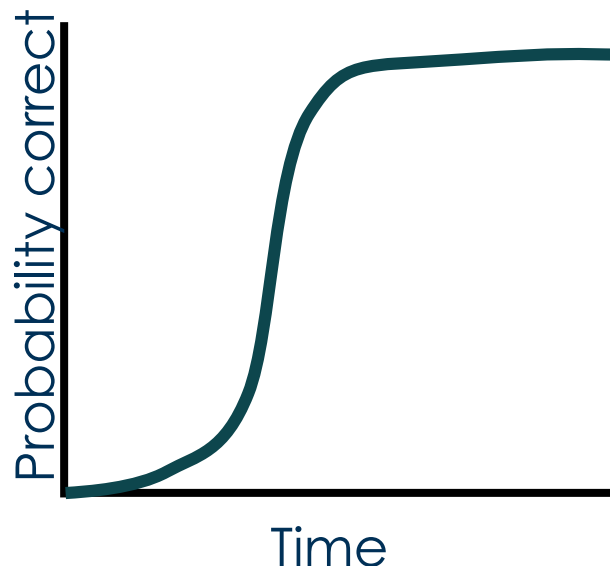
- Writing implies changing a state or value
- Stochastic





Write errors

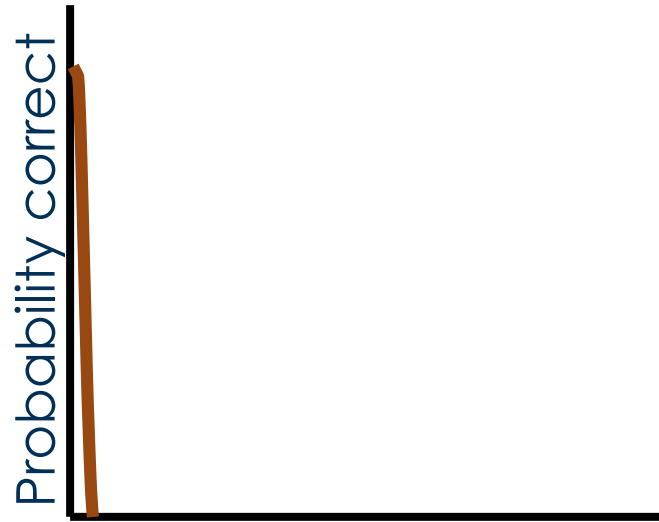
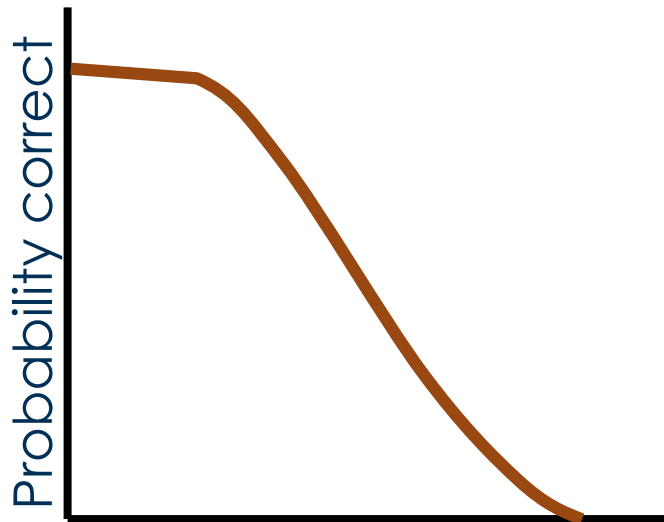
- Writing implies changing a state or value
- Stochastic



- Waiting inefficient and slow
- Write error control conflicts w/ other errors

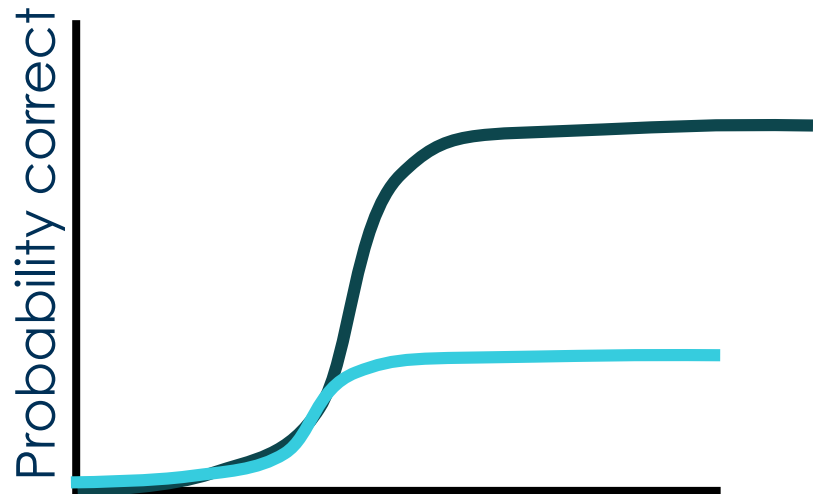
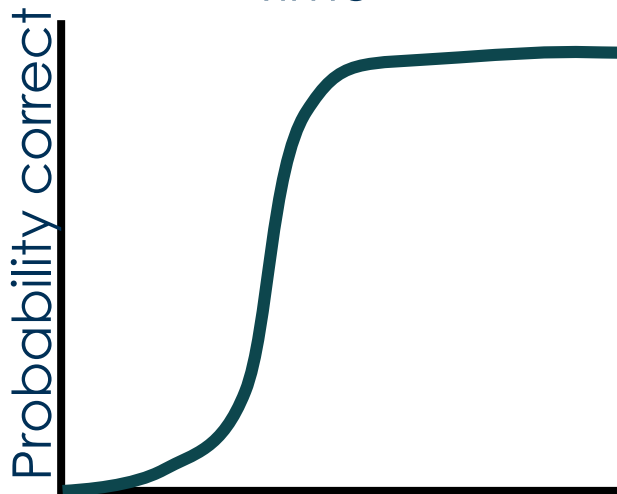


Wearout and defects



Time

Time



Time

Time



Wearout and defects

- Sparing
- Extra margins
- Retirement
- Compensation (ECC)



Summary: no “good” memory
– It’s all about tradeoffs



Example: **DRAM**

- Efficient and reliable reads and writes
- Leaky
- Density problematic
- Fairly fast reads and writes
- Parallelism determined by cost
- Fast decay (short retention)
 - High variability
- Rare, but important flips
- Very rare disturbs
- Slow wearout



Example: **FLASH**

- High-energy writes
- Very dense
- Slow reads
- Very slow writes
- Parallelism determined by technology
- Very slow decay (good retention)
- Flips only on periphery
- Write disturbs problematic
- Fast wearout



Example: **PCM**

- High-energy writes
- Dense
- Fast reads
- Fast-ish writes
- Parallelism determined by cost
- Slow decay (OK retention)
- Flips only on periphery
- Write disturbs problematic
- Fast wearout



Summary of heterogeneity:

interrelated tradeoffs

- Efficiency
- Capacity
- Latency
- Bandwidth
- Parallelism (granularity)
- Reliability



The role of
heterogeneous **processors**
heterogeneous **applications**



Proportional **compute**

- Latency oriented
- Throughput oriented
- Specialized



Application characteristics

- Latency sensitive
- Bandwidth sensitive
- Few tasks or many tasks



Application compute **styles**

- Batch
- Realtime
- Response time



Application **data** usage

- Capacity
- Locality
- Precision
- Persistence



Memory *must be heterogeneous*
but **illusion of homogeneity** is nice



The solution:
Adaptive proportional memory systems



The solution:
Adaptive proportional memory systems



Adaptive **reliability**



Adaptive reliability

- Adapt the **mechanism**
- Adapt the **error rate**
 - precision (stochastic precision)



Adaptive reliability

– By **type**

- Application guided

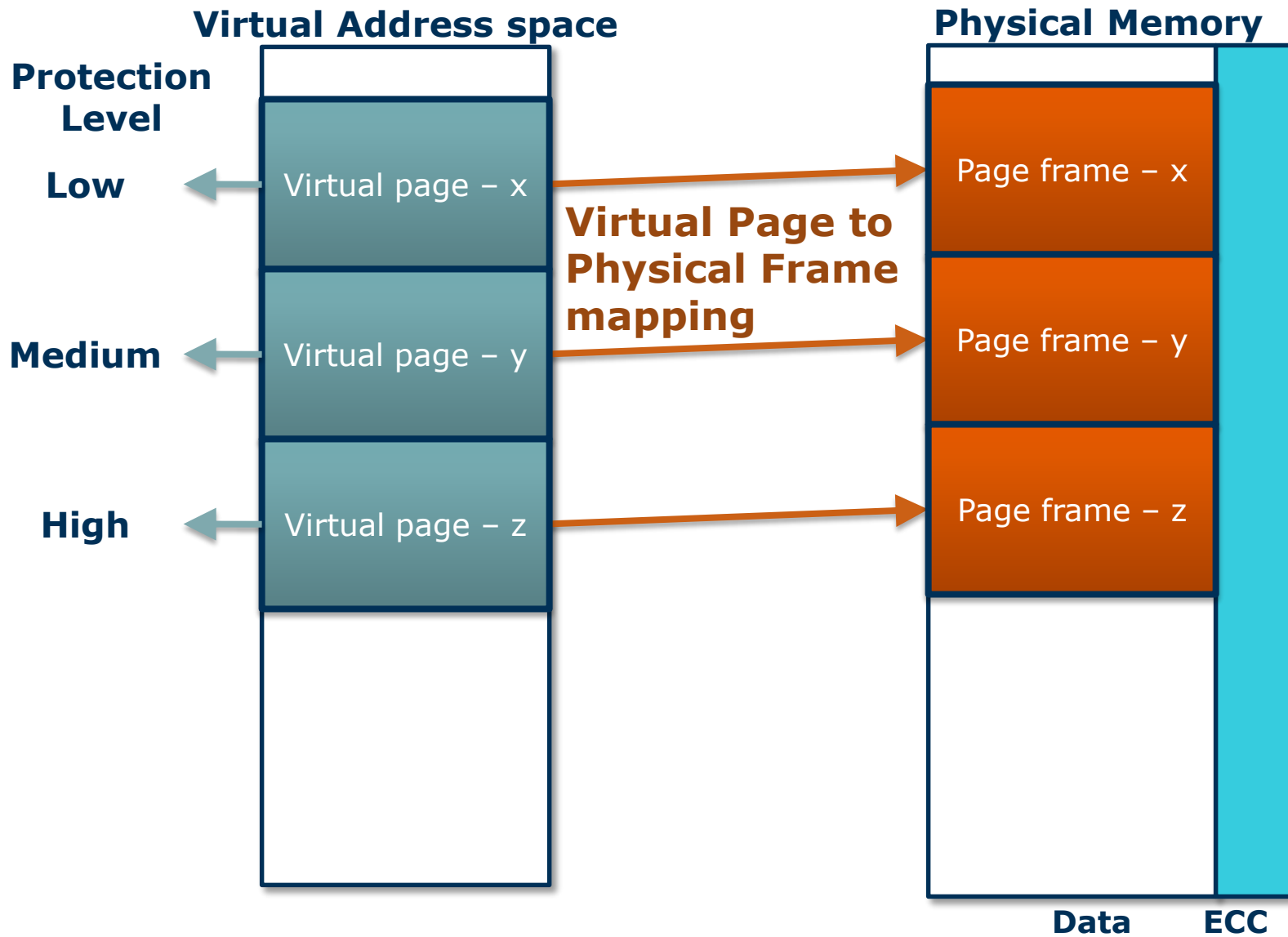
– By **location** (83)

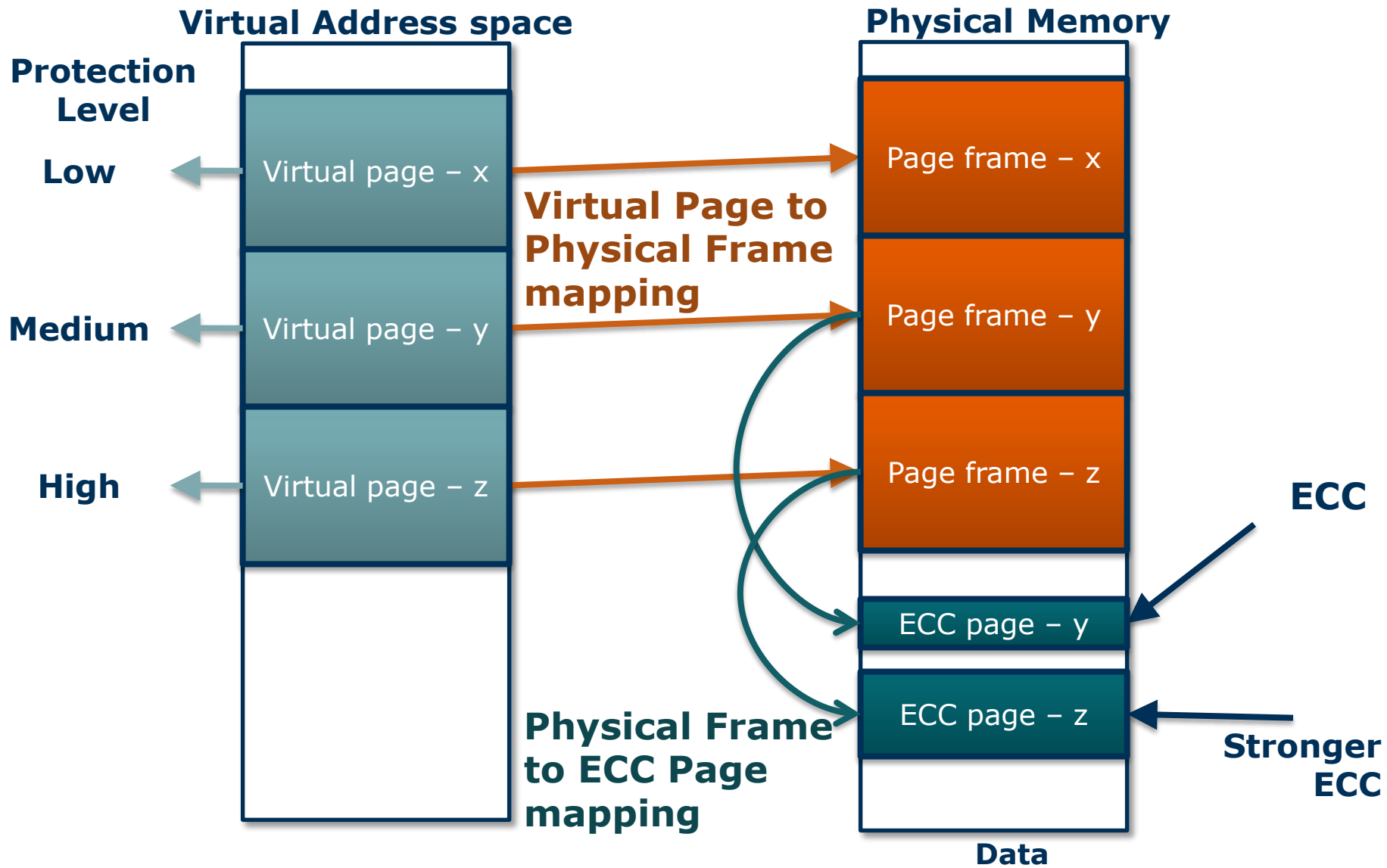
- Machine-state guided

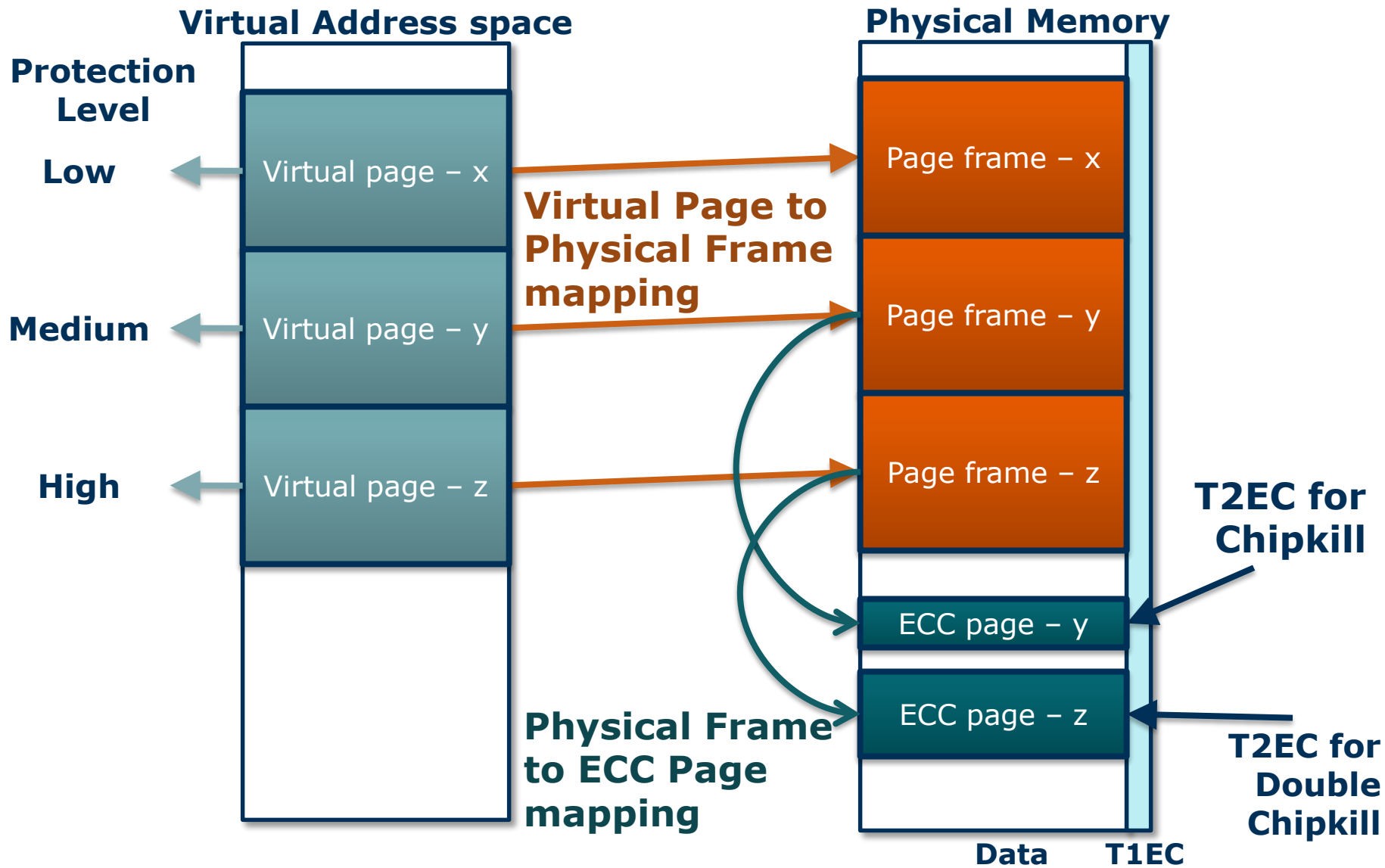


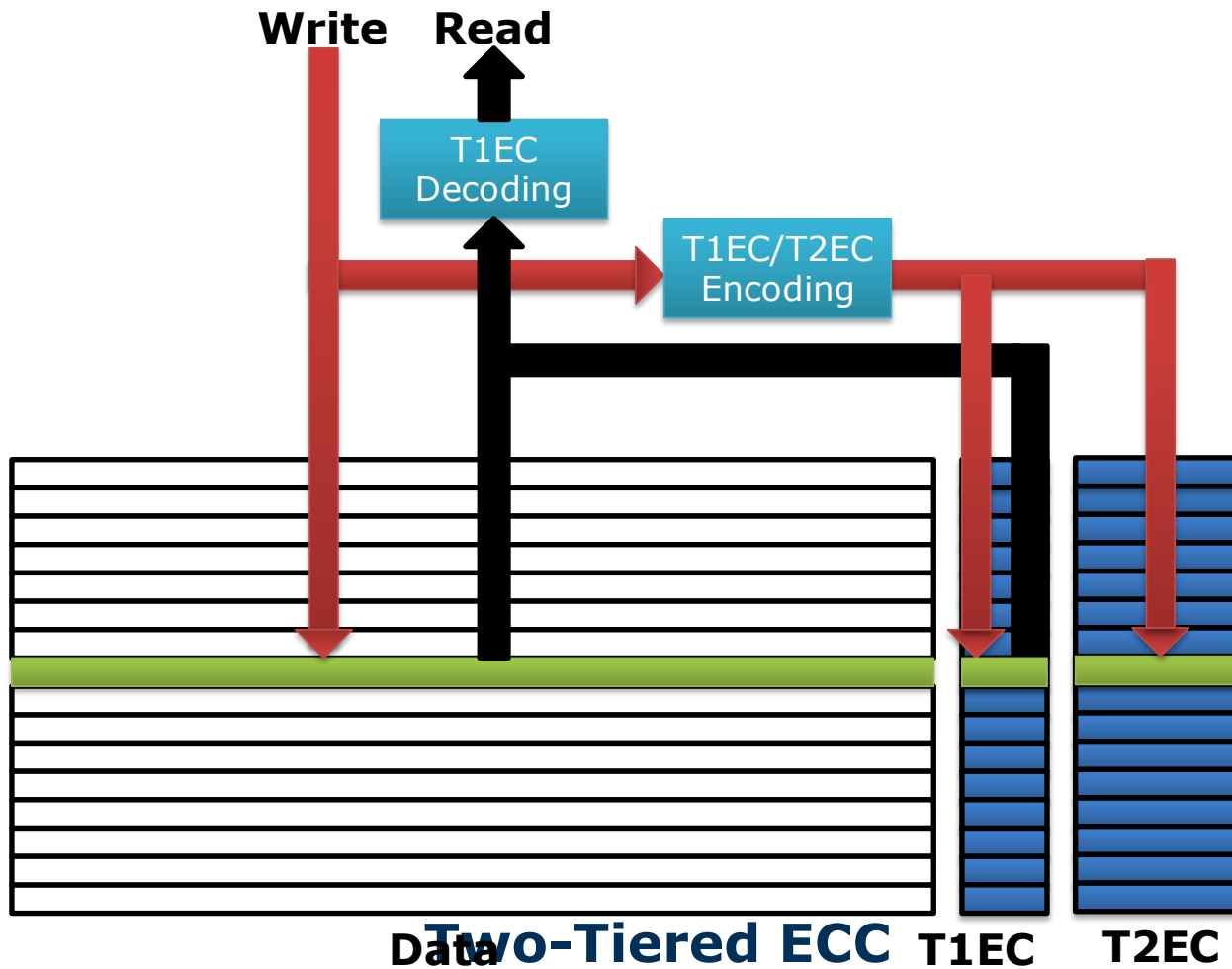
Example: **Virtualized ECC**

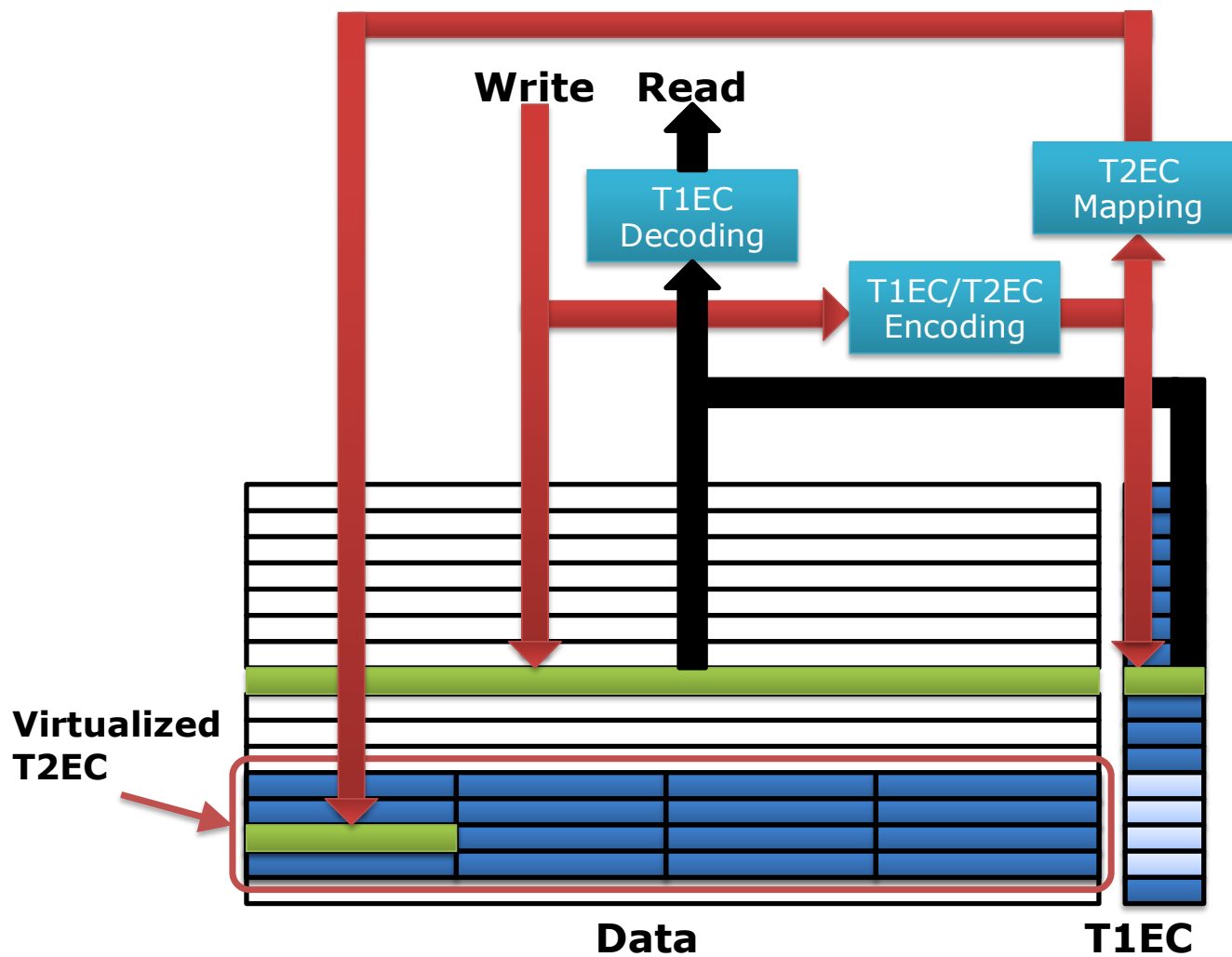
– Adapt the mechanism





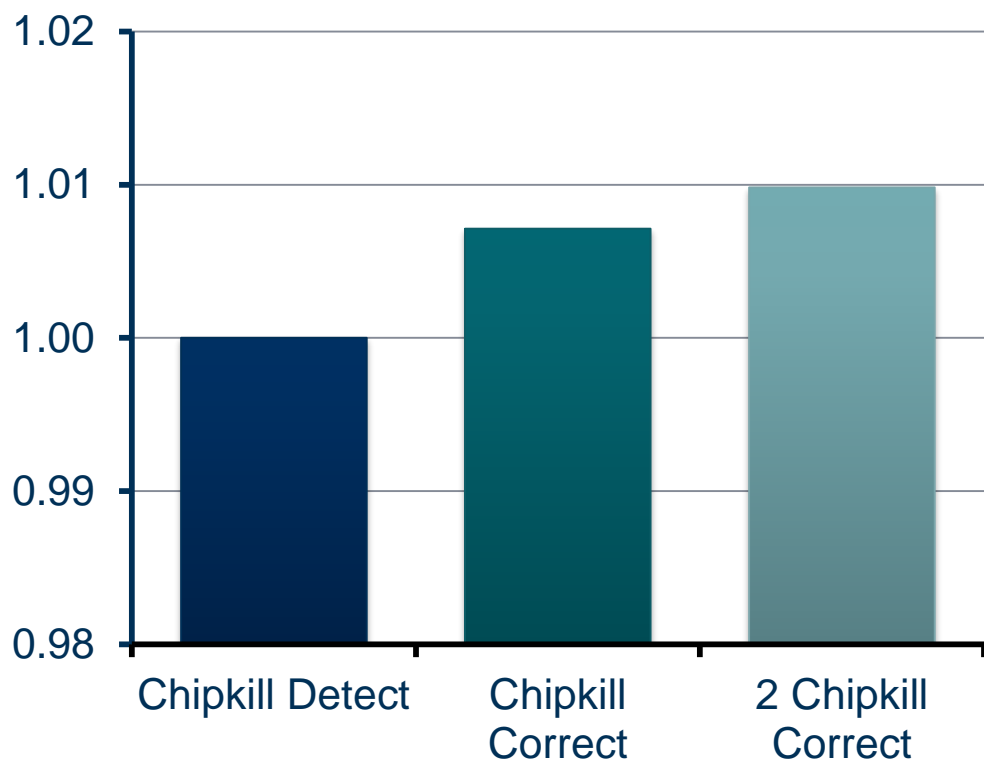








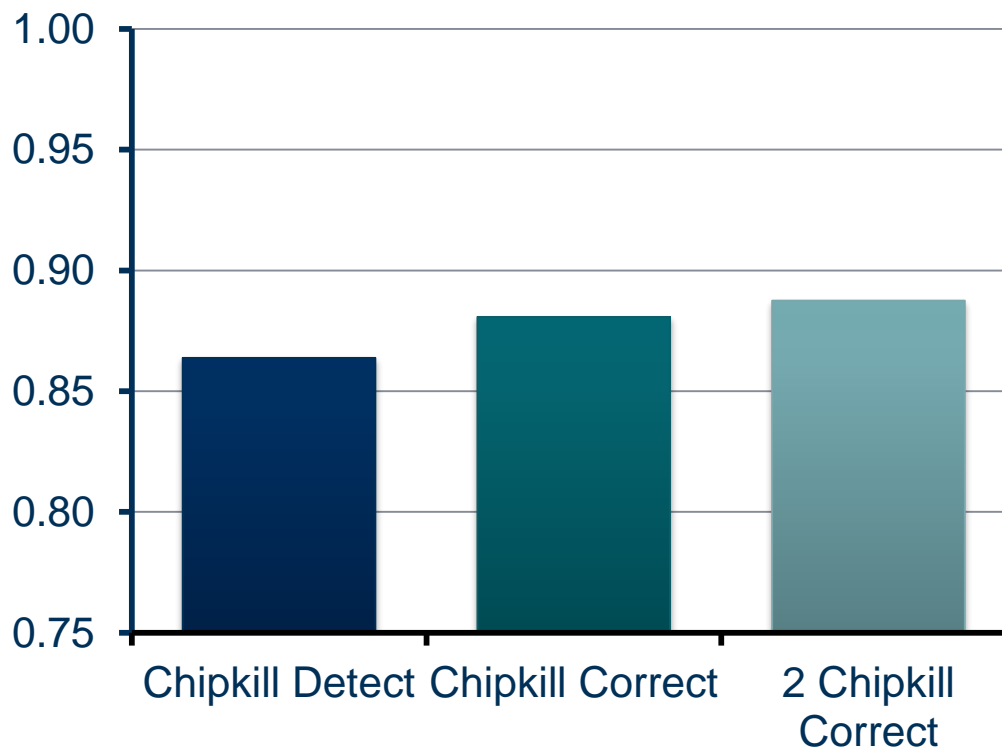
Caching work great



Normalized Execution Time



Bonus: **flexibility** of ECC



Normalized Energy-Delay Product



Example: **FREE-p**

– Adapt to wearout state



FREE-p insights:

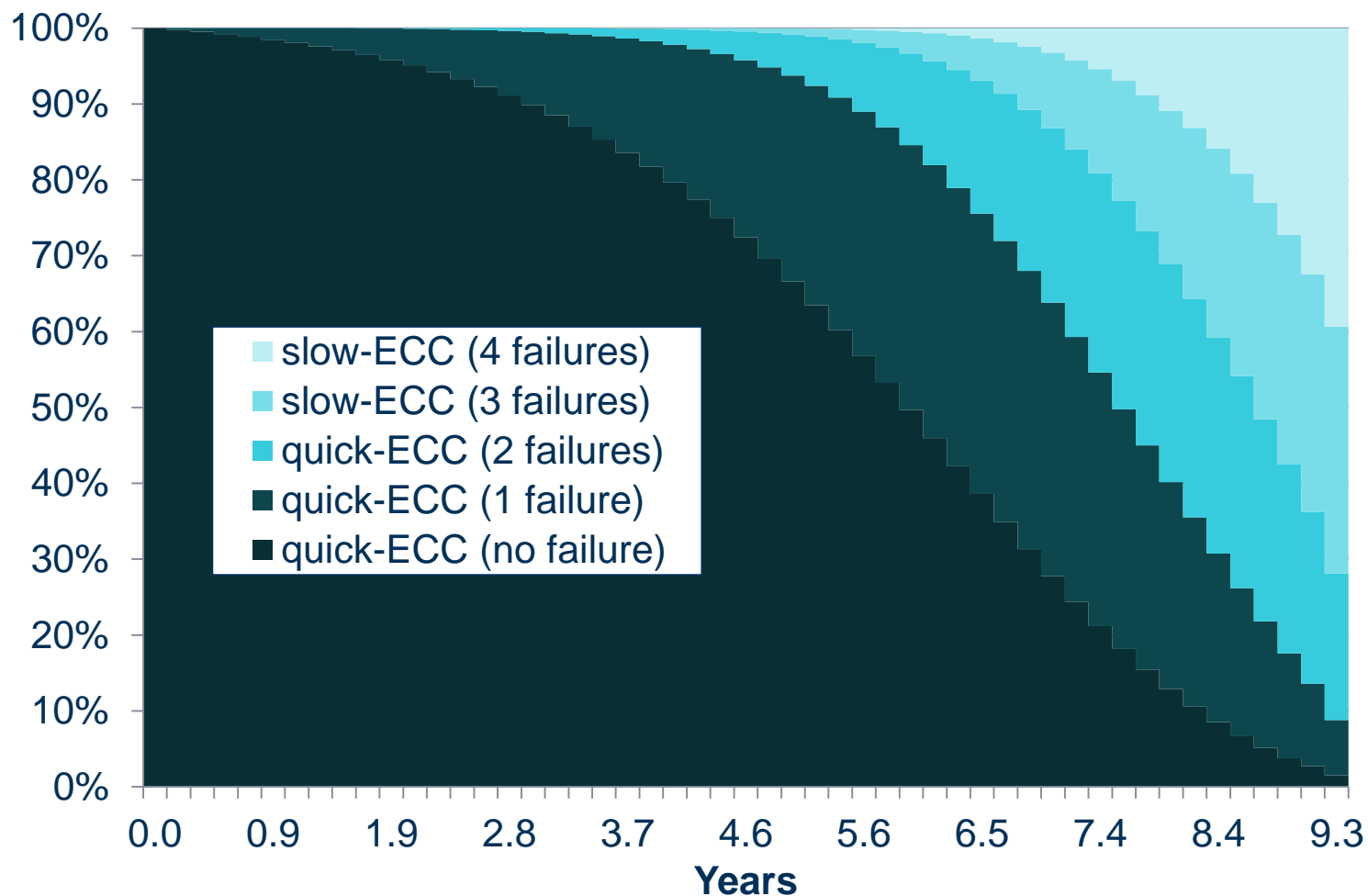
- Wearout is **gradual**
- Adapt ECC strength to wearout degree
- Limit ECC need with adaptive repair
 - Retire and remap



Adapt ECC strength – **Multi-tiered** ECC

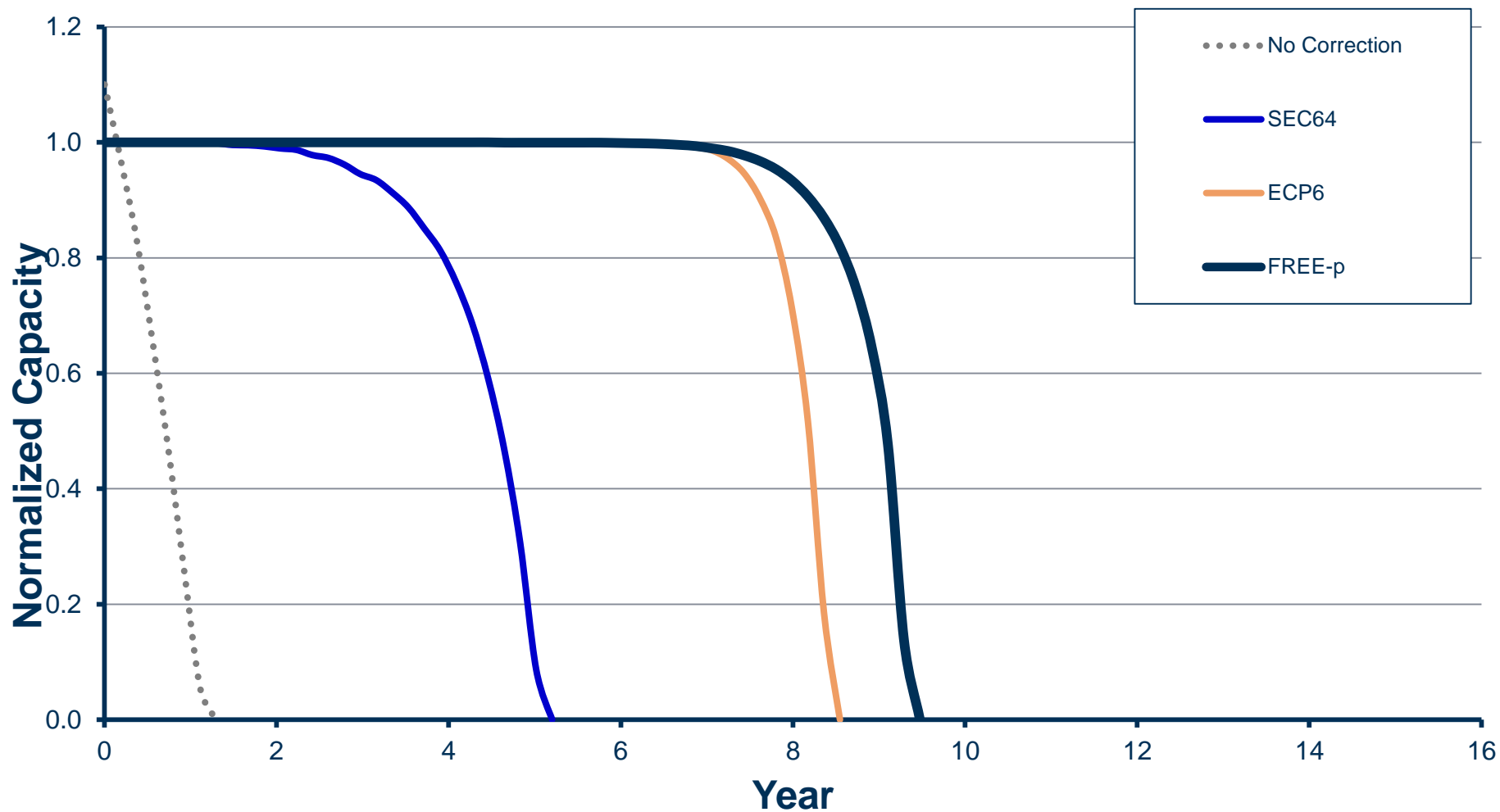


Different cells need **different protection**



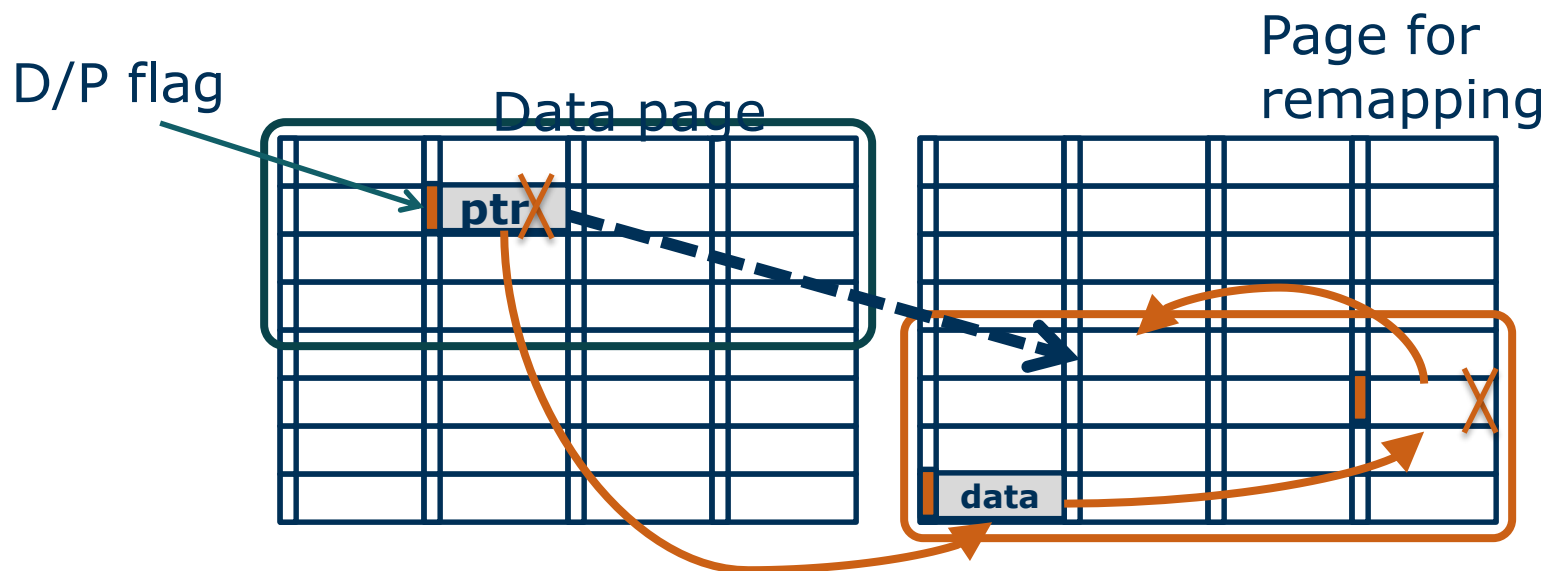


Fine-grain retirement is key



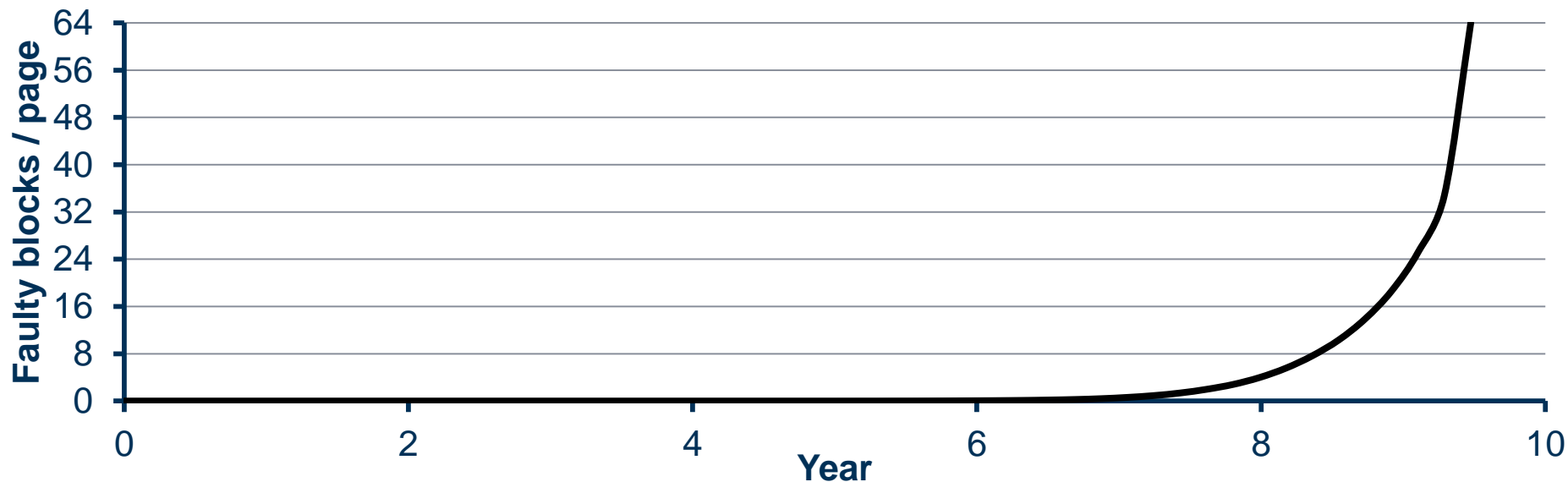


Adaptive FG repair





Impact of repair and remap is **gradual**

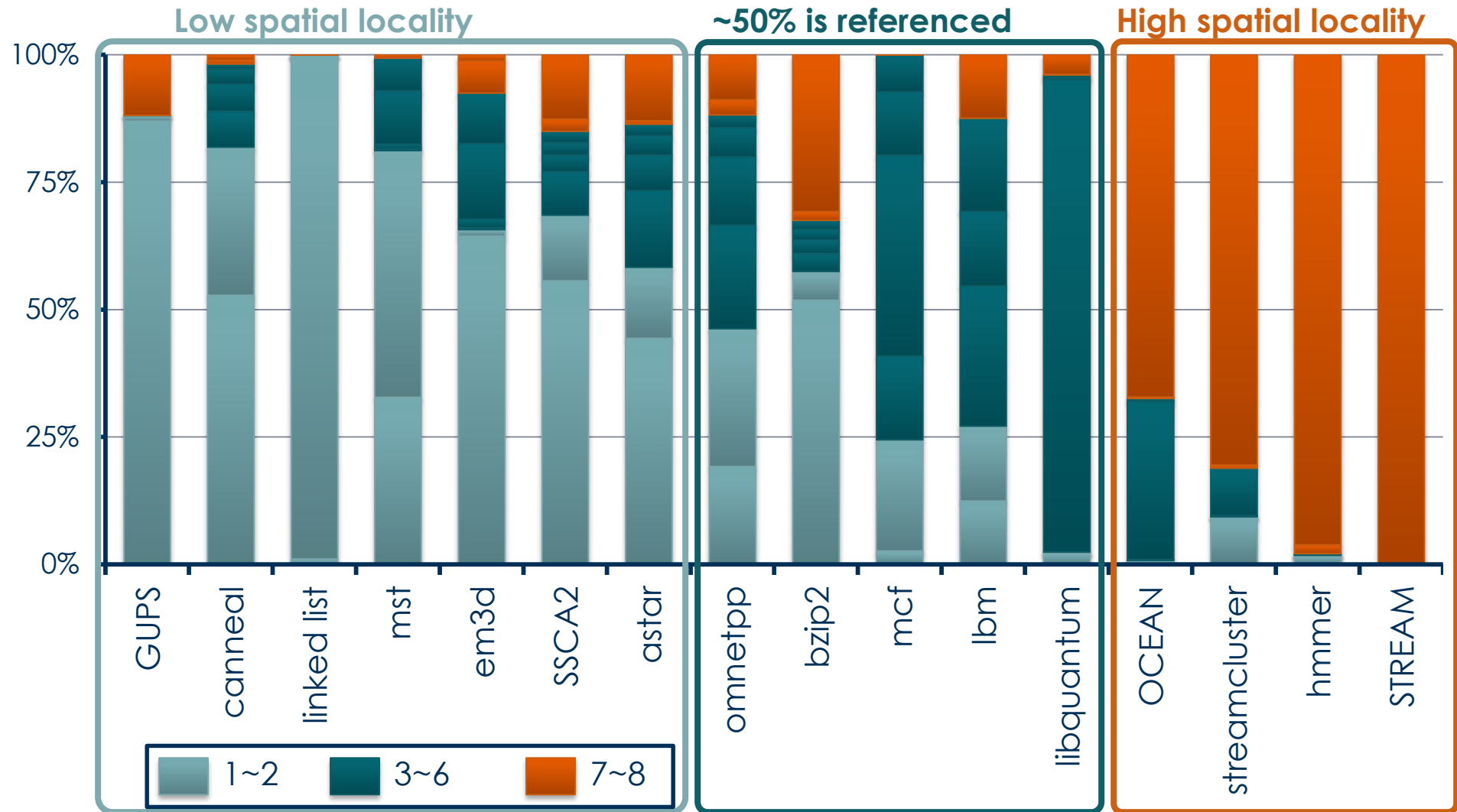




Adaptive **granularity** (parallelism)



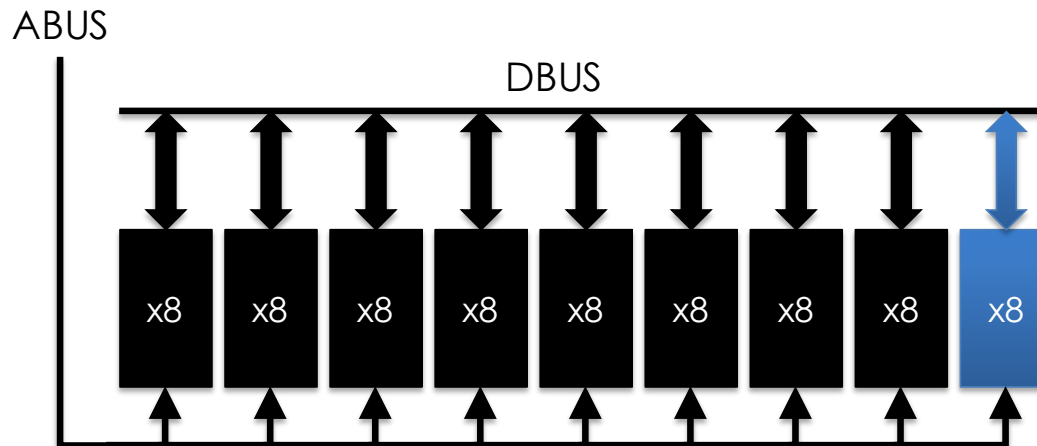
Applications have **diverse locality**



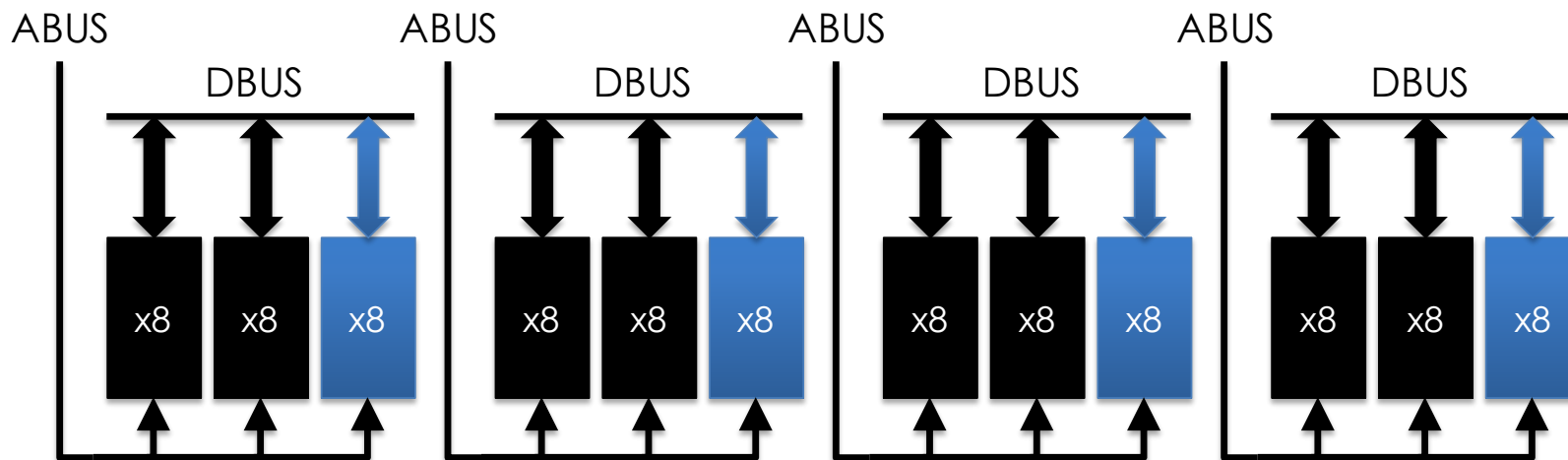


Coarse- or fine-grained memory access?

CG-Only: Wide DRAM channel

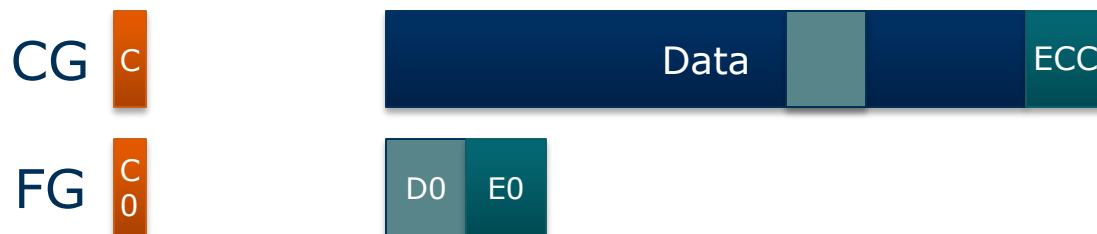
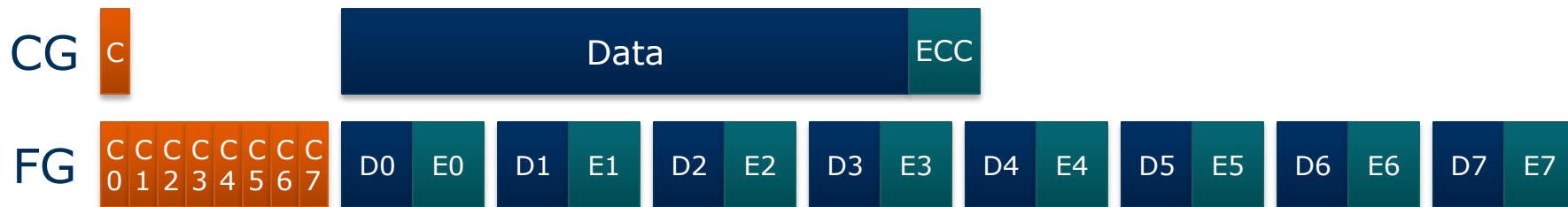


FG-Only: Many narrow channels



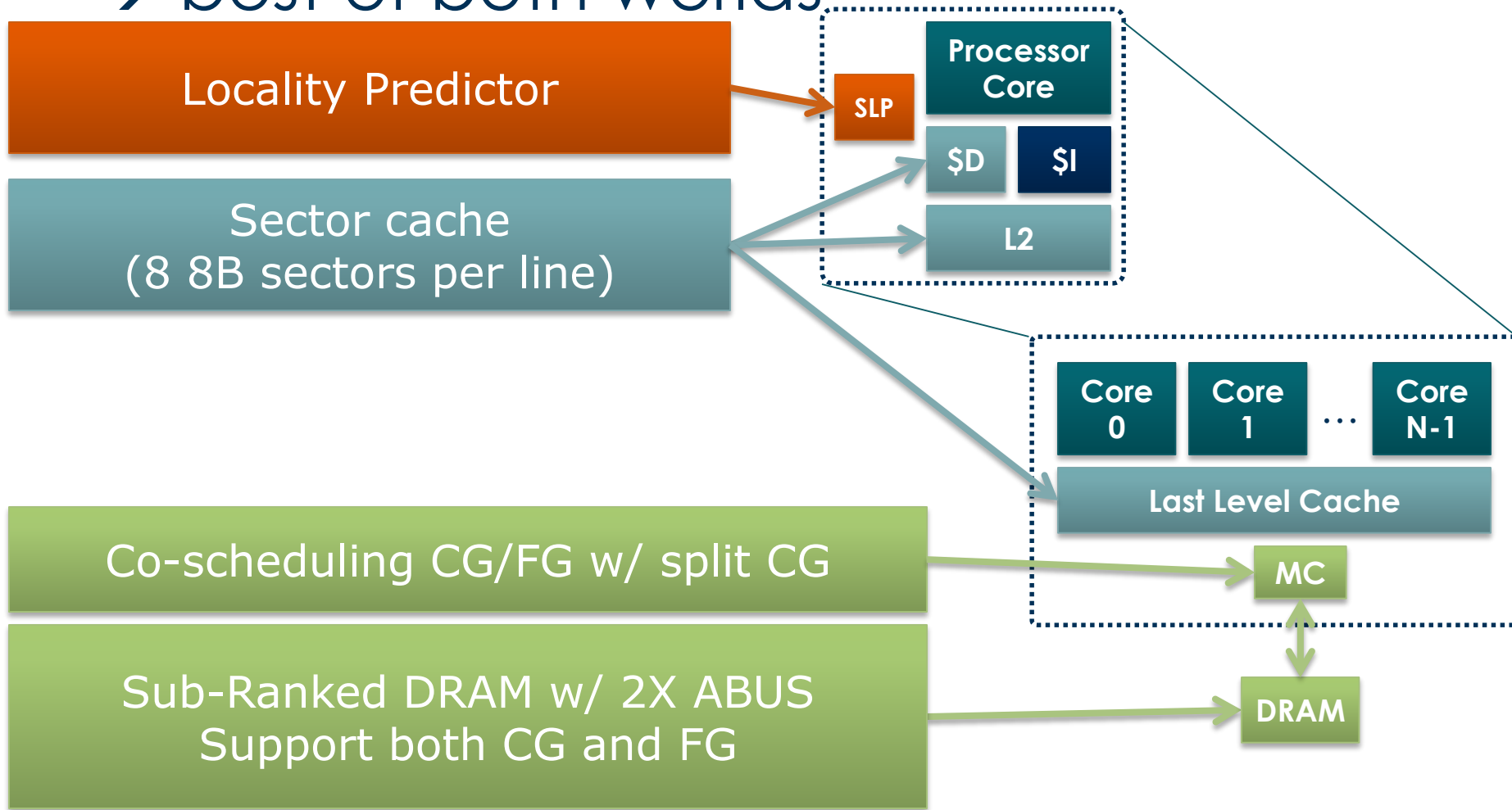


Coarse- or fine-grained memory access?





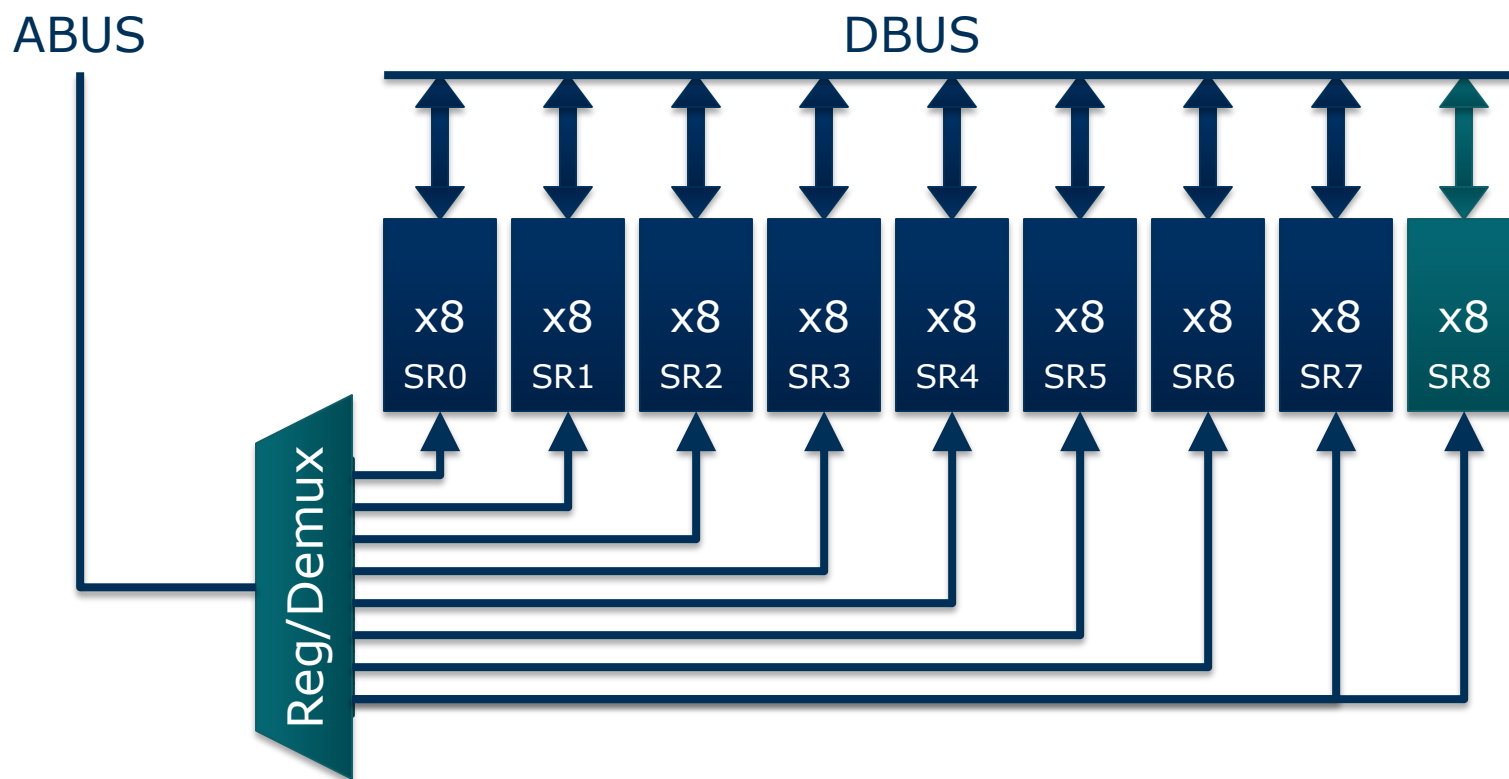
Dynamically **adapt** granularity → best of both worlds





Subranked memory

– Adapt parallelism





Sectored caches

– Track granularity

Long cache line



Short cache line



Sector cache



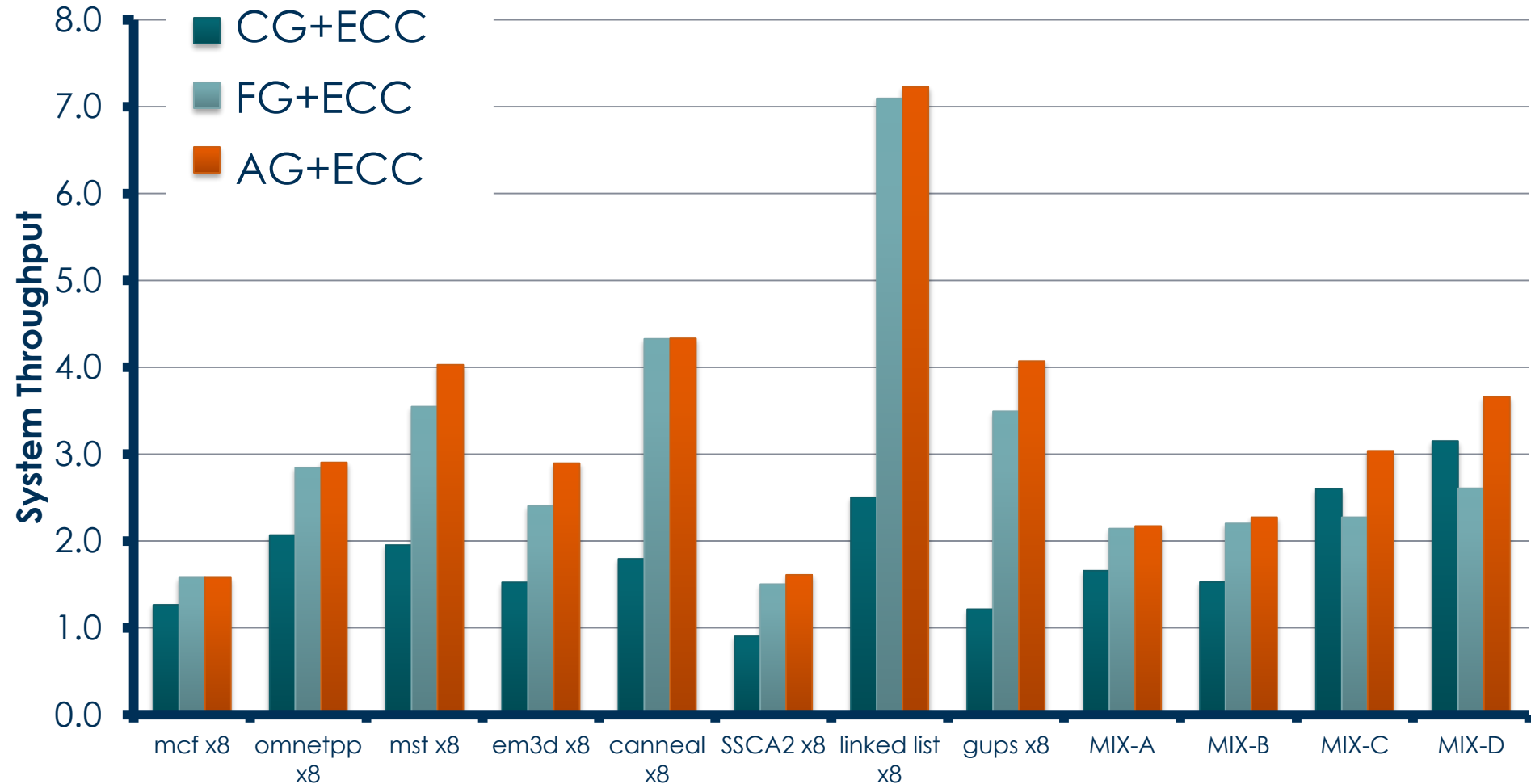


Granularity **information?**

- Application provided (when allocated)
- System learned (when accessed)



Dynamically **adapt** granularity



PROPORTIONALITY



Memory *is* heterogeneous
Applications *are* heterogeneous
this is a good thing



Adaptivity is key



Sometimes need application help

– **Programming models and analyses crucial**



Think **abstractions** rather than examples



Memory is heterogeneous

Applications are heterogeneous

this is a good thing

Adaptivity is key

Sometimes need application help

– **Programming models and analyses crucial**

Think abstractions rather than examples