Harnessing 100 Billion Unruly Transistors

Babak Falsafi

Presenting work of many

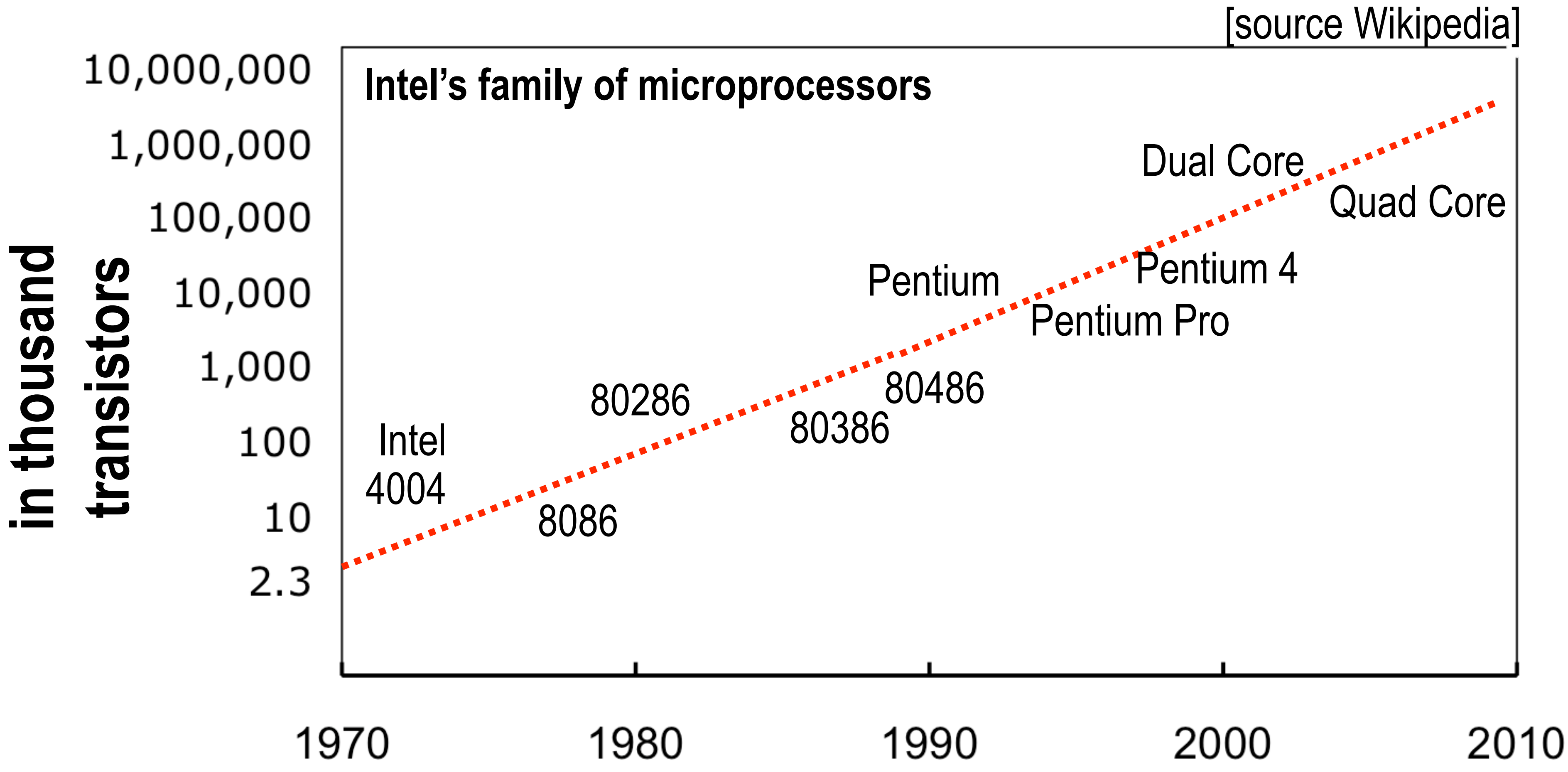
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

PARSA
Parallel Systems Architecture Lab
EPFL
people.epfl.ch/babak.falsafi
www.c2s2.org

Computers: A Fabric of Our Society

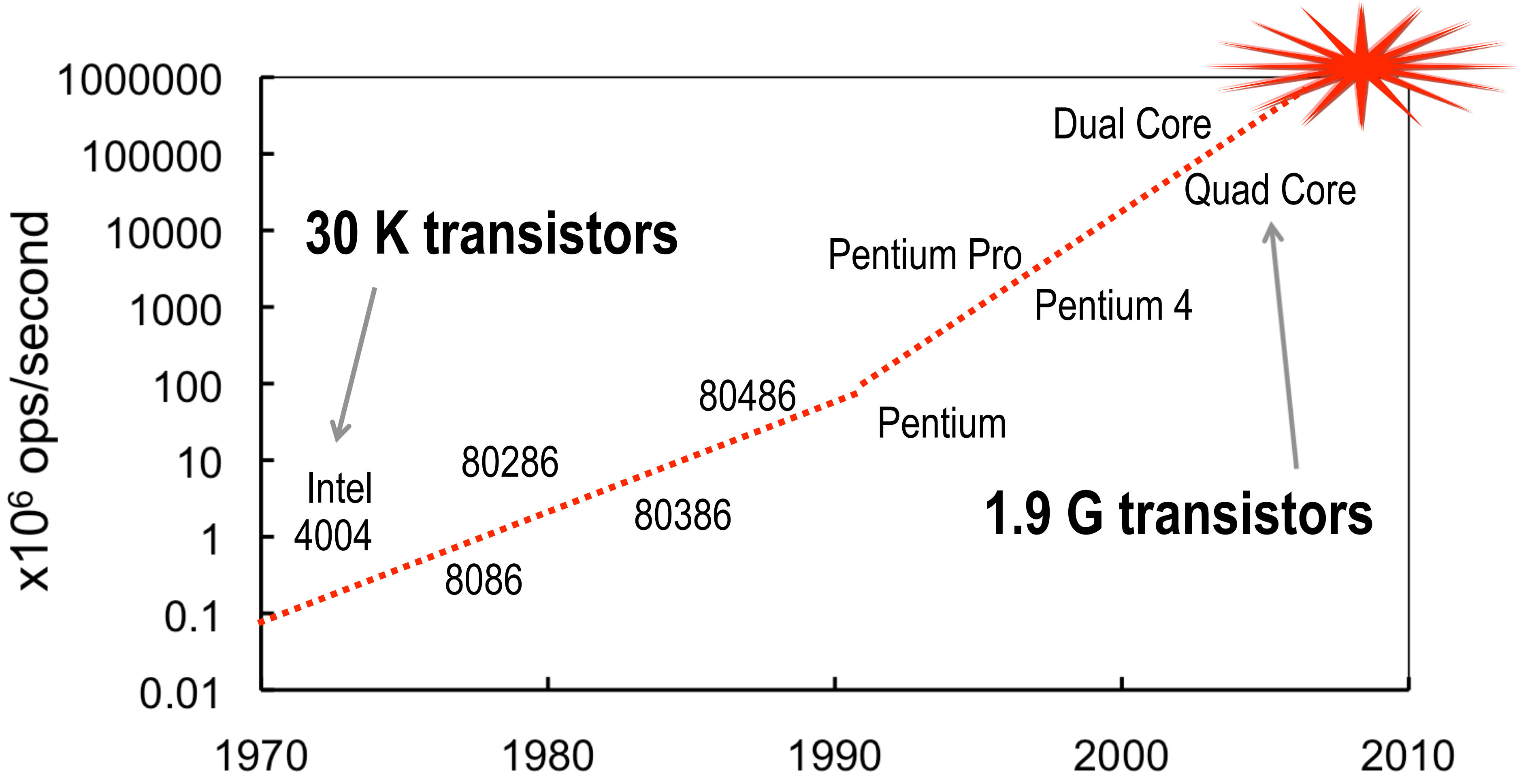
Communication, commerce, entertainment, health services, transportation, government, …

How did we get here? Moore’s Law



Technology forecast in 1965 by Gordon Moore
= 2x transistors every 24 months

Perceived Moore's Law: Performance



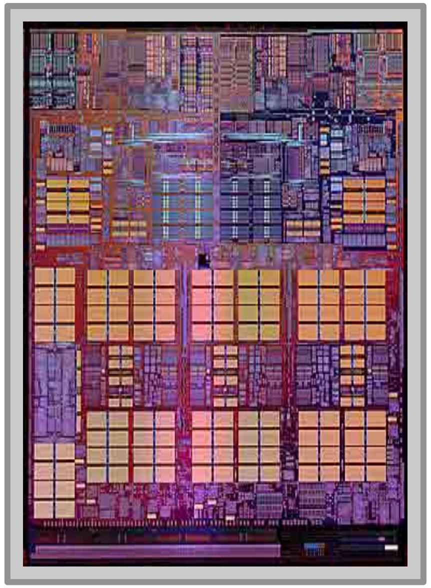
Computer architecture + circuits

➔ Performance doubles every 18 months!

Our ideal 100-billion trans. chip

We have so far succeeded in riding the Moore's Law because microprocessors

1. Ran legacy SW (serial)
2. Scaled in performance
3. Maintained power envelope
4. Did not fail (were robust)



Expectations are high

➔ can we continue delivering?

Our likely 100-billion trans. chip

Several key challenges, or "walls", facing computer system designers

Hardware may fail (this talk)

➔ in-the-field solutions

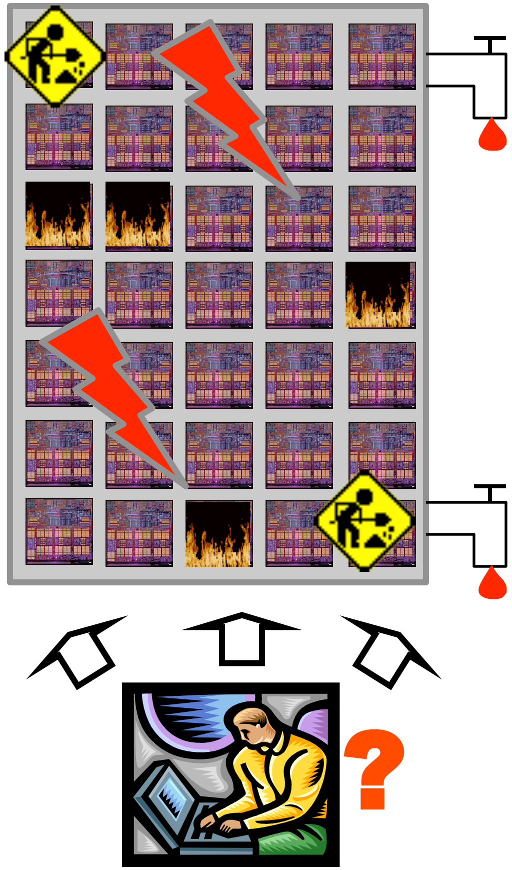
Power does not scale

➔ customize

Multicore chips

➔ need parallel SW

Memory…..



Outline

- ~~Overview~~
- Computers with unruly transistors
- Detecting/correcting error in logic
- Detecting/correcting error in memory

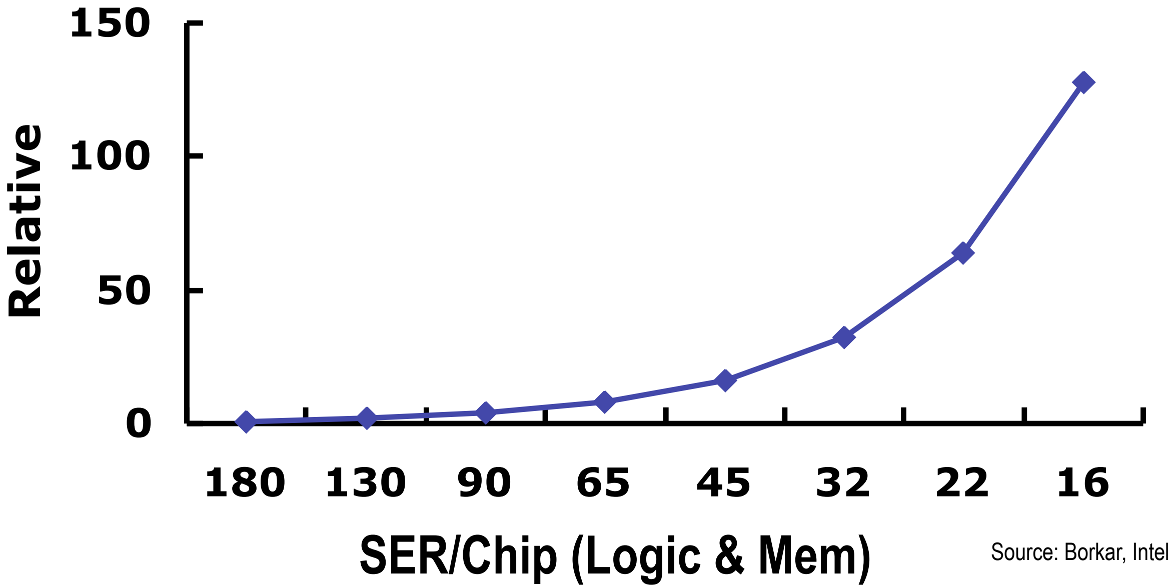
Why would hardware fail?

As devices scale, there are three emerging sources of error that manifest in circuits:

1. Transient (soft error)
   - ❑ Upsets in latches & SRAM
2. Gradual (variability)
   - ❑ Sensitivity in device performance
3. Time-dependent (degradation)
   - ❑ Small devices age faster

Sources of Error: Transient

- Scaling — increasing density, decreasing charge
- In pipeline latches and memory
  - ❑ Complex, large-scale ➔ coding techniques don’t apply



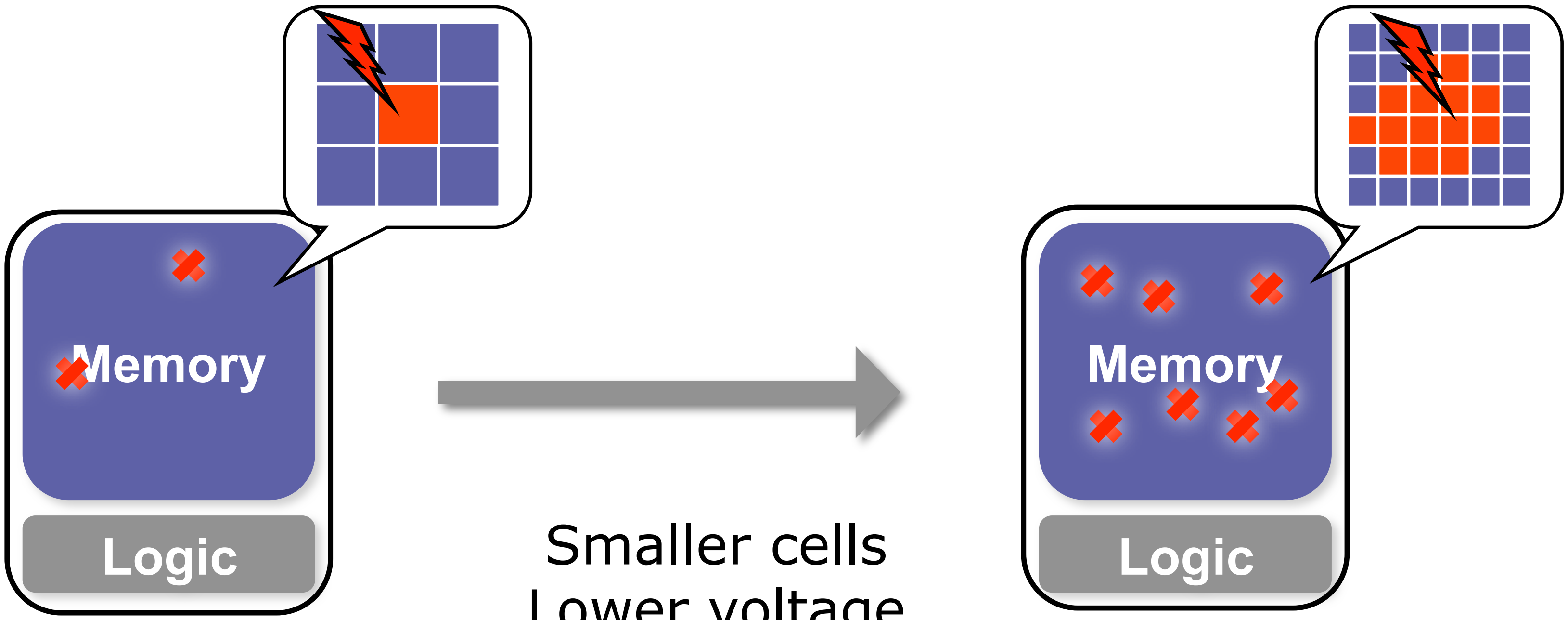
Source: Borkar, Intel

Exponential increase in bitflips!

Source of Error: Transient

Naturally occurring cosmic rays upset charges in latches & memory cells:

- ❑ Future chips: single strike → multiple upsets



Smaller cells
Lower voltage
Higher variability

Sources of Error: Manufacturing

Manufacturing uses lithography to fabricate

- Increasingly difficult to produce transistor of certain size when below wavelength
- Two identically designed transistors on chip each will have different speeds

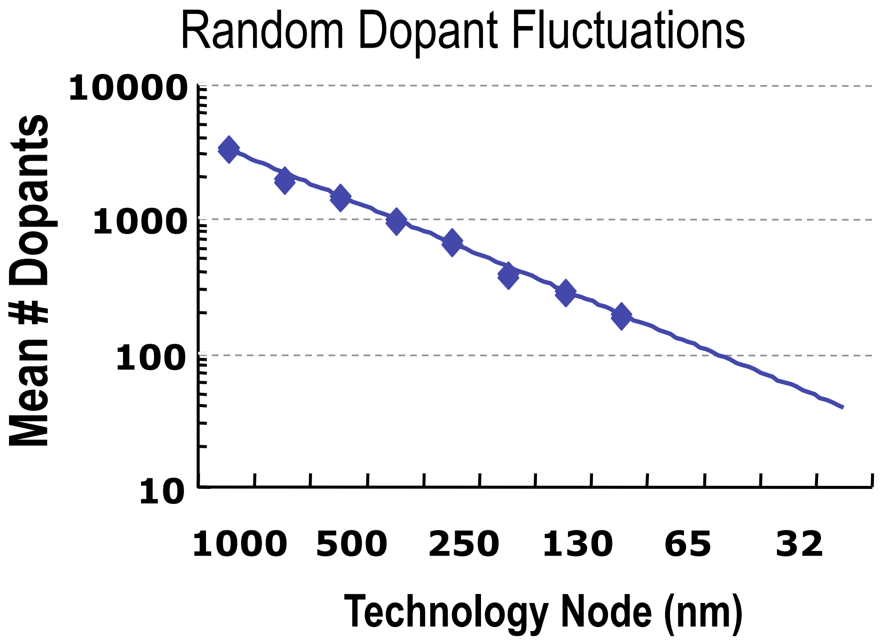
Small fluctuation affects transistor speed

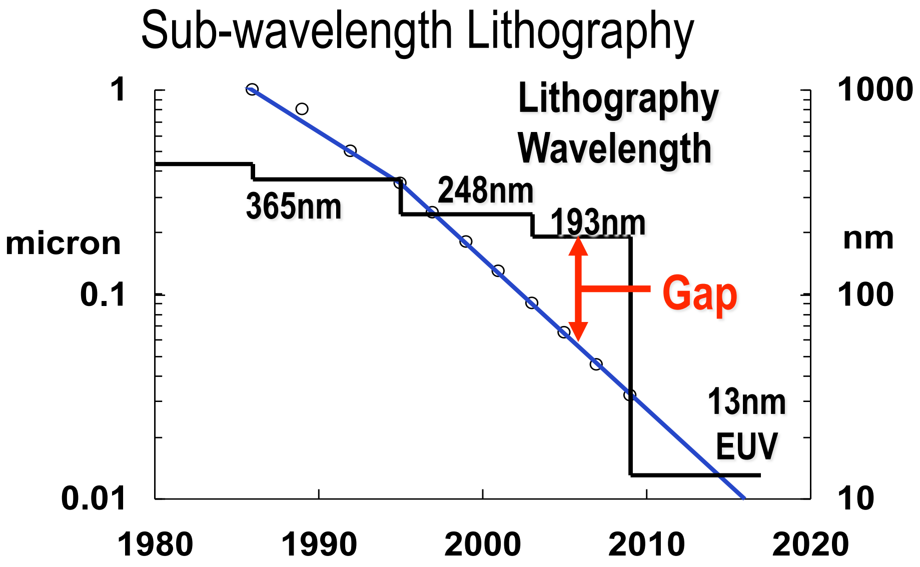
- in material density across chip
- in size across chip

Dramatic increase in defect density!

Sources of Error: Manufacturing

- Increasing variability at manufacture



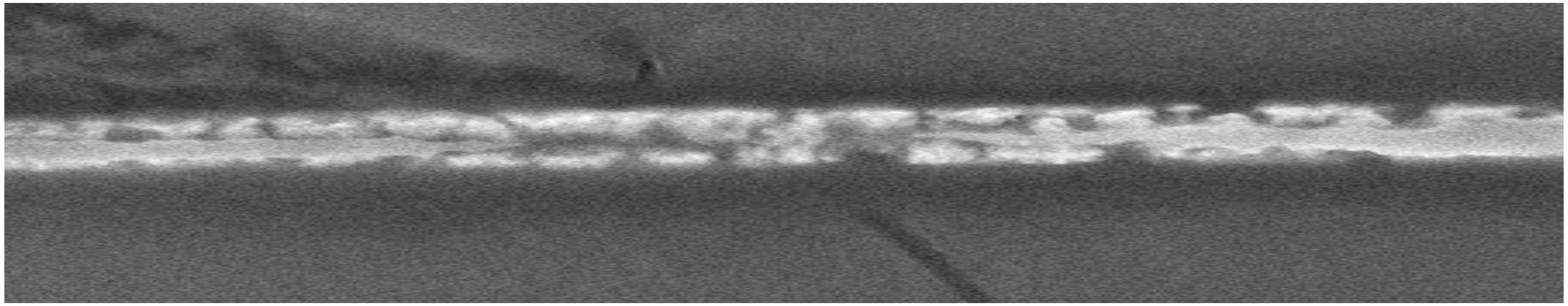


Sources: Borkar/Bohr, Intel

Need to deal with manufacturing variability & defects!

Sources of Error: Lifetime

- Transistors/wires degrade through time
  - ❑ Electromigration, oxide breakdown,…
  - ❑ As we scale, transistors/wires age faster



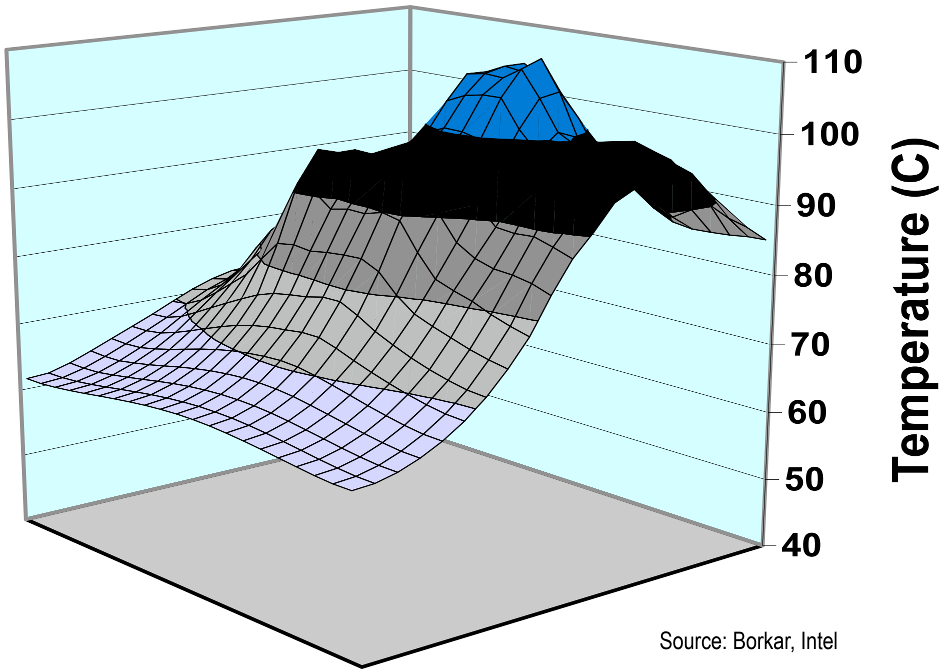
Electromigration

Source:Zörner

Accelerated chip failure!

Sources of Error: Heat & Voltage

- Time-dependent variability
  - ❑ Switch slower in hot spots or change in V
  - ❑ Smaller devices, more sensitive to fluctuation



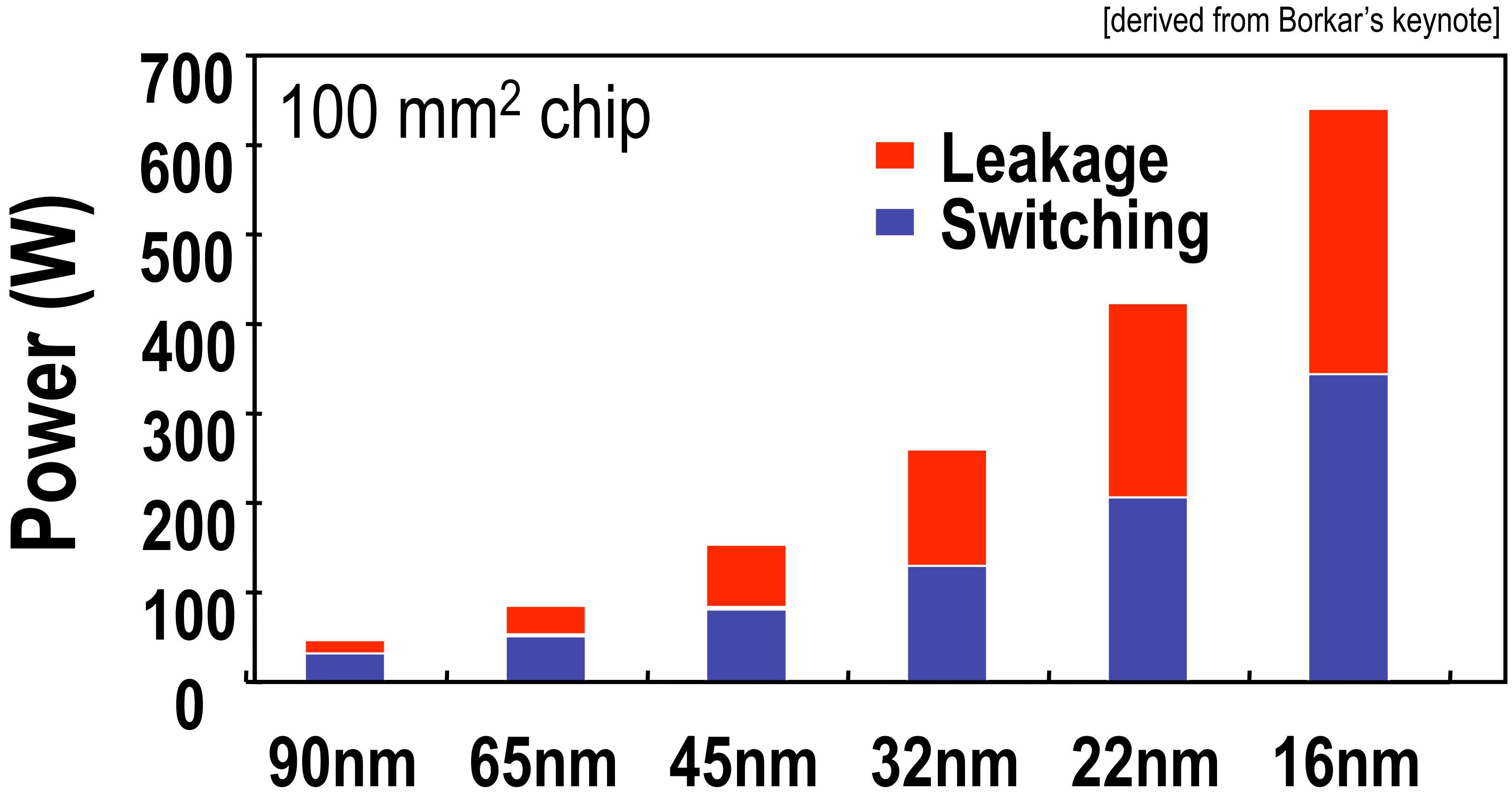
Source: Borkar, Intel

Temperature hot spots

Need to deal with gradual error!

Increase in Leakage Power

[derived from Borkar's keynote]

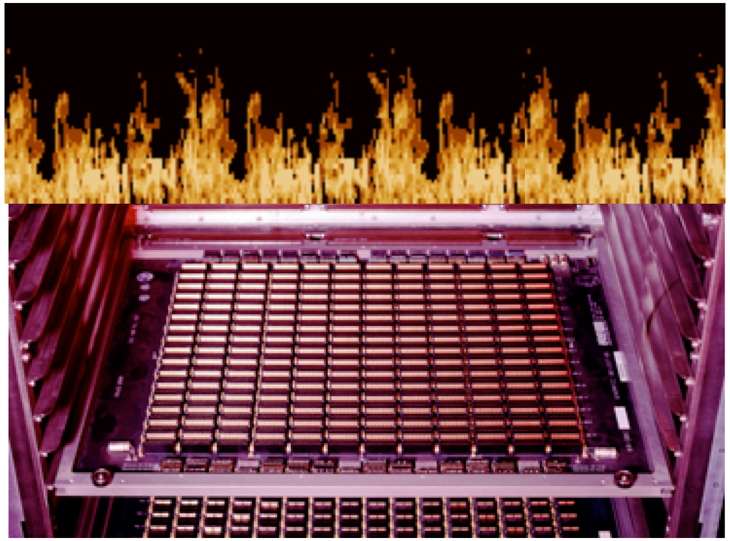


Leakage is exponentially dependent on temperature ➔ exacerbates heat swing

Burn-in may phase out?

Chips are stress-tested in “burn-in” ovens

- At high temperatures, device failure accelerated
- Historically, reliable way to catch chips that die early



With rising leakage power,

burn-in may phase out:

- all chips will burn!

Need to deal with high chip infant mortality in the field!

Why does it matter? [S. Mitra]

Today:

- 20,000-processor datacenter
- One "major" error every 20 days

Undetected errors can be unwavering:

- ❑ Which way did the bit flip?
- ❑ Bank account deposit of 20K CHF could be either 3.6K CHF or 53K CHF

May need fast repair:

- ❑ downtime cost 100K-10M CHF/hour

Conventional Approaches are too Expensive!

Building all circuits redundantly can only be for a small market segment (e.g., IBM z990)

Not affordable for all!

Need "cheap" techniques

- Little hardware & fast
- Current codes too complex
- Software (e.g., Google) too slow

Need fast detectors if always engaged

- Correctors only when error occurs

What should we do?

Must design reliable systems with unreliable components

- Can’t even count on circuits

Need cost-effective solutions to reliability at all computing stack layers:

- Algorithmic
- Programming model
- System software
- Architecture
- Circuit

Outline

- ~~Overview~~
- Computers with unruly transistors
- Detecting/correcting error in logic
- Detecting/correcting error in memory

Architectural Techniques to Protect Computation

Checker processor

- DIVA, SHREC, …
- High coverage, but dedicated HW

Symptom-based techniques

- Cheap, but low coverage

Signature-based techniques

- Distributed checkers in HW/SW

Redundant multithreading

- AR-SMT, RMT, Reunion, etc…
- Pay overhead when needed

Redundant Multithreading

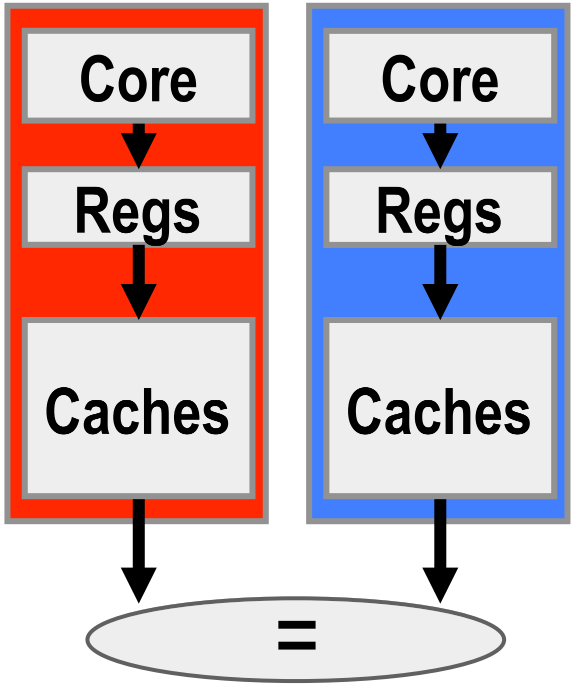
Redundant execution

- ❑ Single pipe or across cores
- ❑ Detect soft error
- ❑ Within core hard error

Across chips

- ❑ Tolerate chip failure

DMR across cores

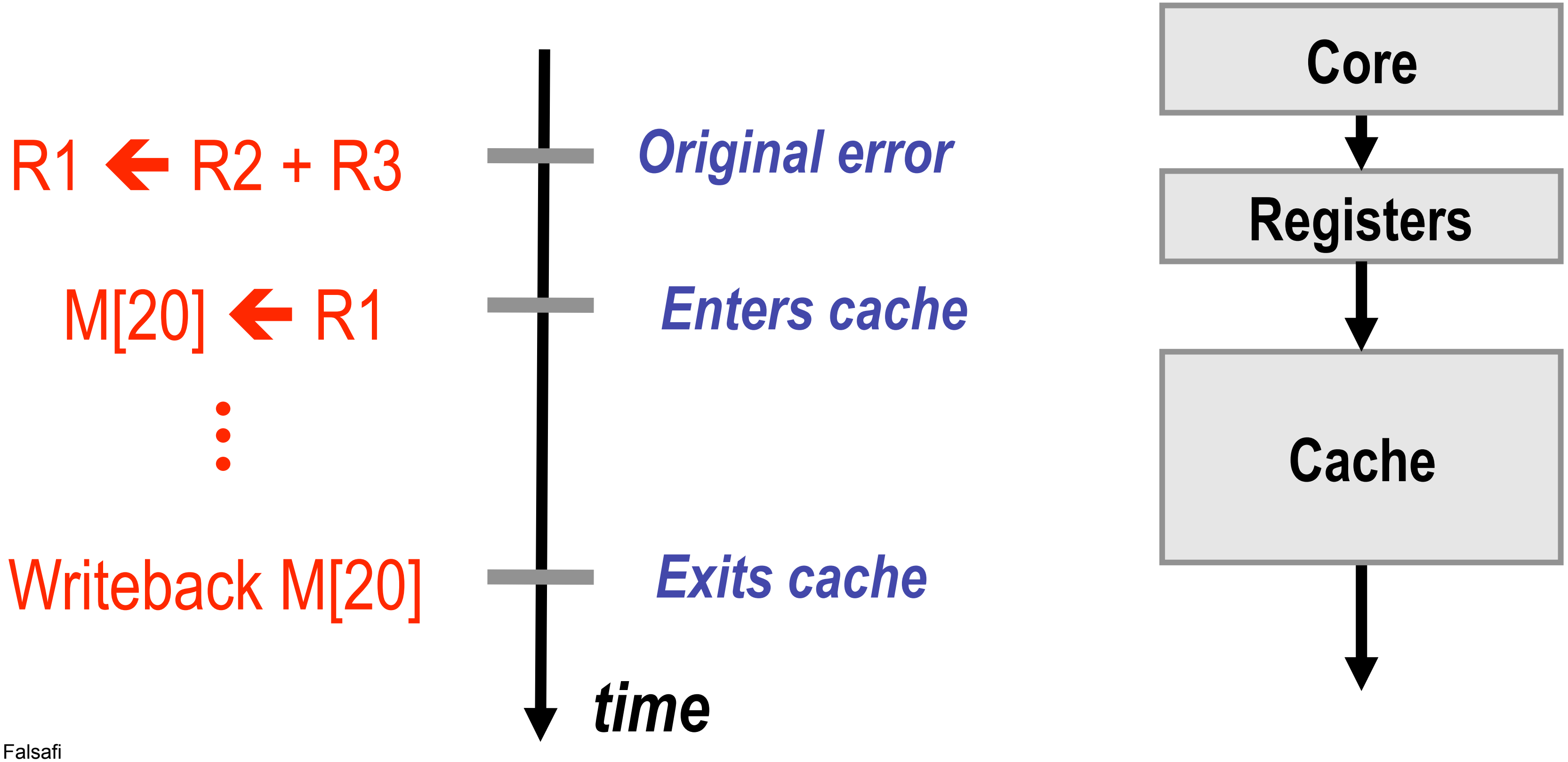


Key challenges

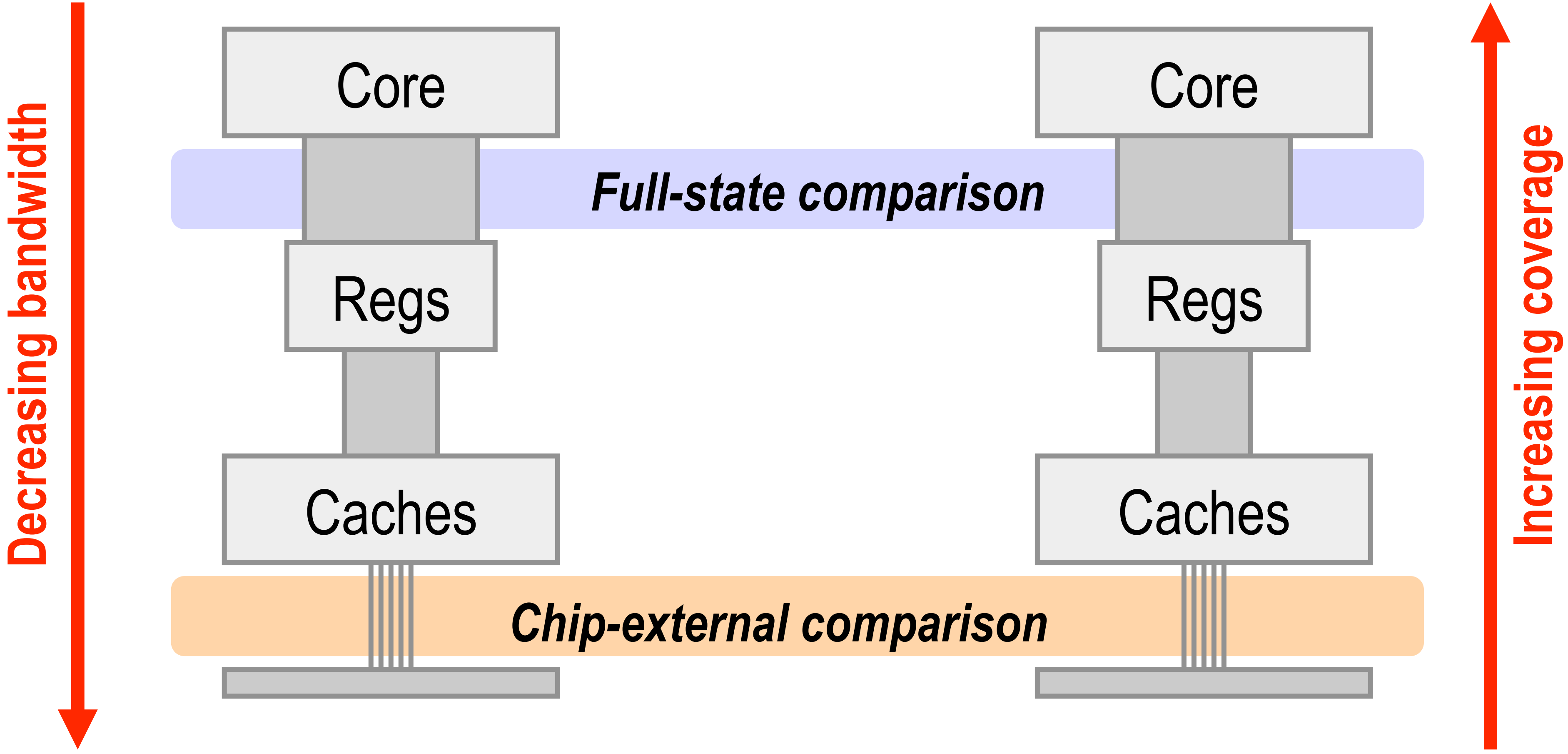
- ❑ How to detect errors?
  - ‣ Need low latency, low bandwidth
- ❑ How to replicate input

Error Detection: Latency

- Existing solution: compare chip-external traffic
  - ❑ Errors can hide in cache for millions of instructions
  - ❑ Recovery harder with longer detection latencies



Error Detection: Tradeoffs



Want high coverage with low bandwidth

Fingerprinting: Low-Overhead Error Detection

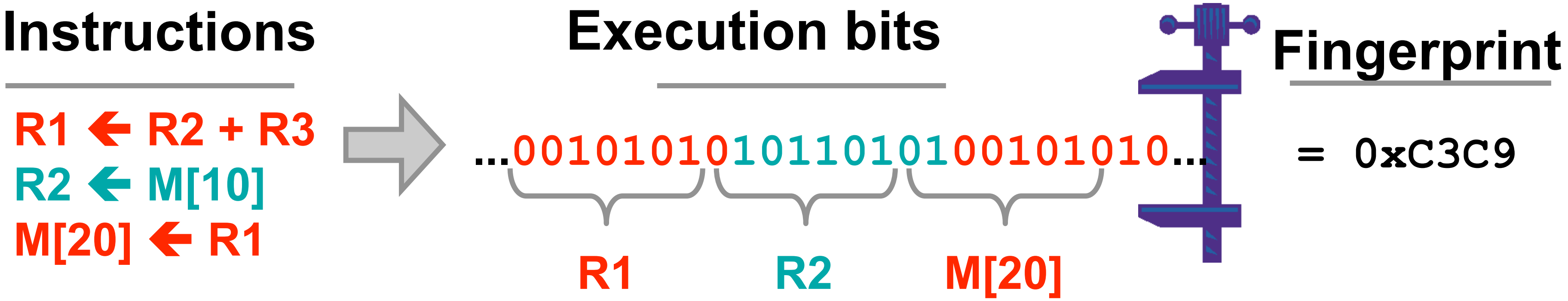
[IEEE MICRO top pick'04]

- Hash updates to execution state
- Compare across redundant threads (or against pre-computed values)

✓ Bounded error detection latency

✓ Reduced comparison bandwidth

✓ Little hardware overhead



Instructions

R1 ← R2 + R3

R2 ← M[10]

M[20] ← R1

Execution bits

...001010101011010100101010...

R1

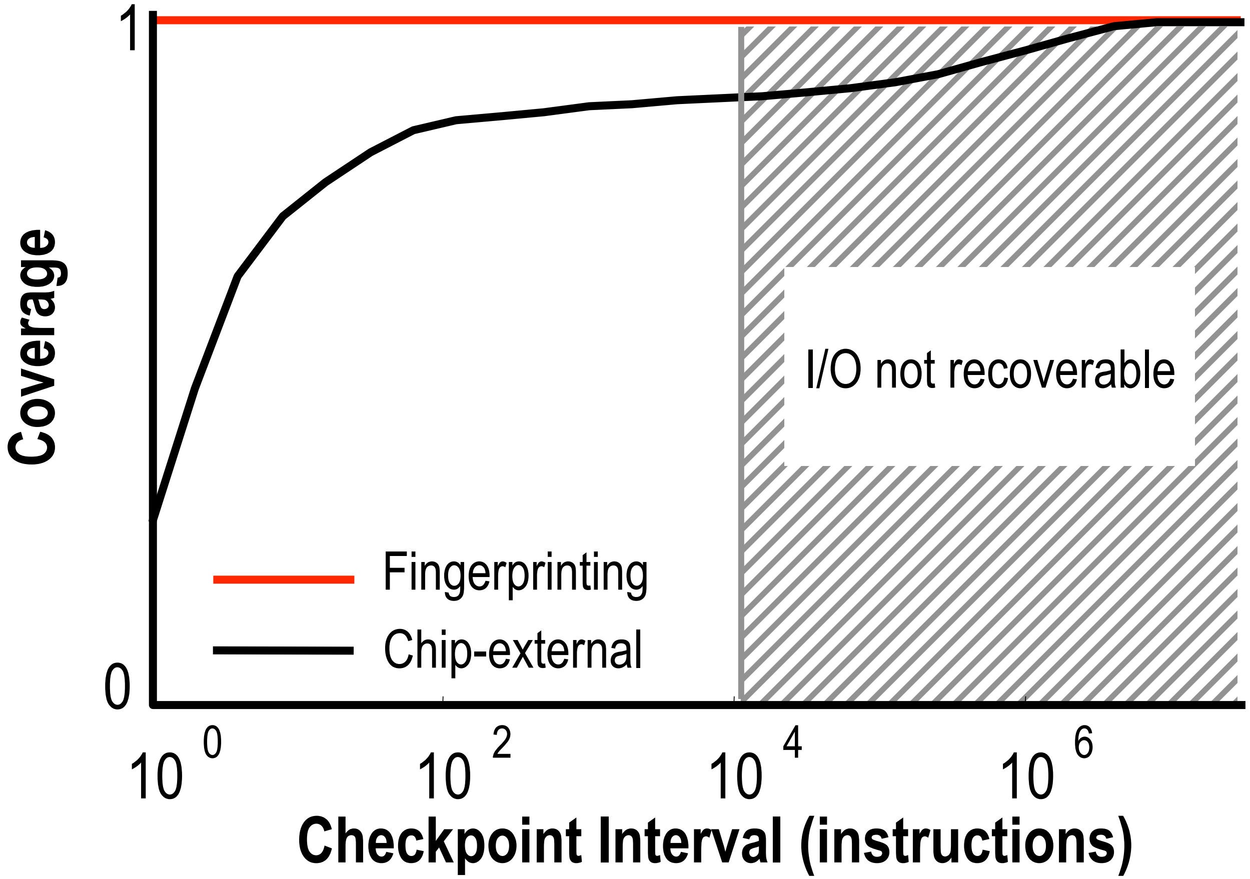
R2

M[20]

Fingerprint

= 0xC3C9

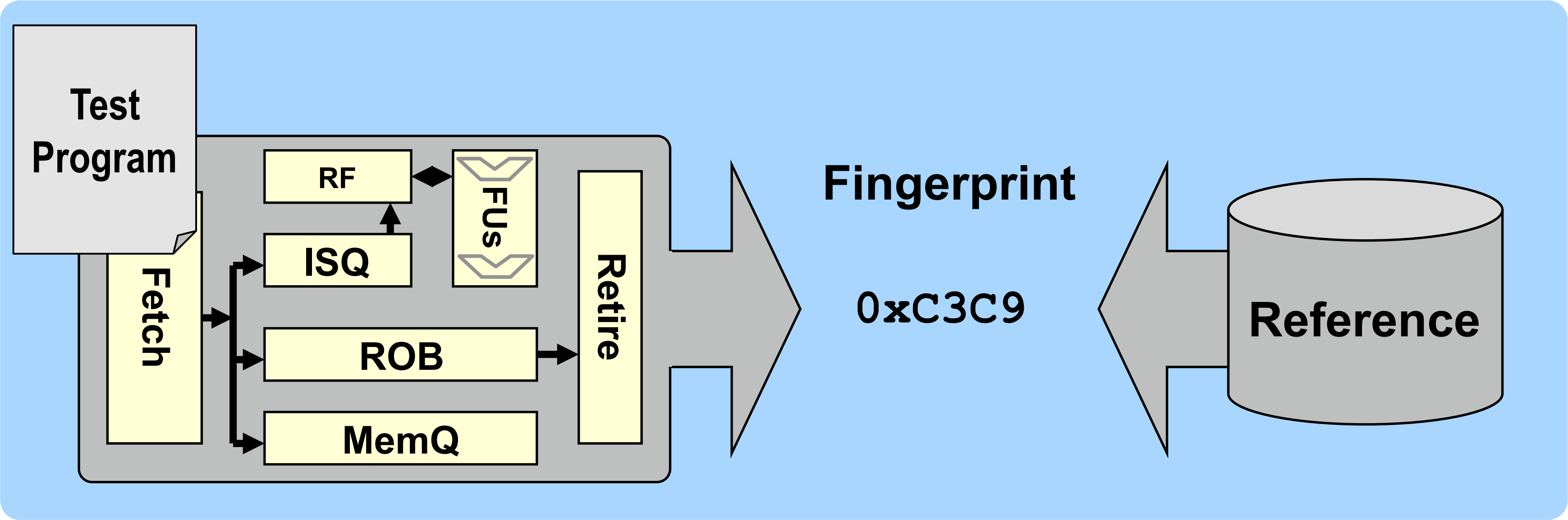
Error Detection: Coverage



- 16-bit (CRC) fingerprint → near perfect coverage
- Chip-external → acceptable coverage for >1M

FIRST: Fingerprinting in Reliability & Self Test [SELSE'07]

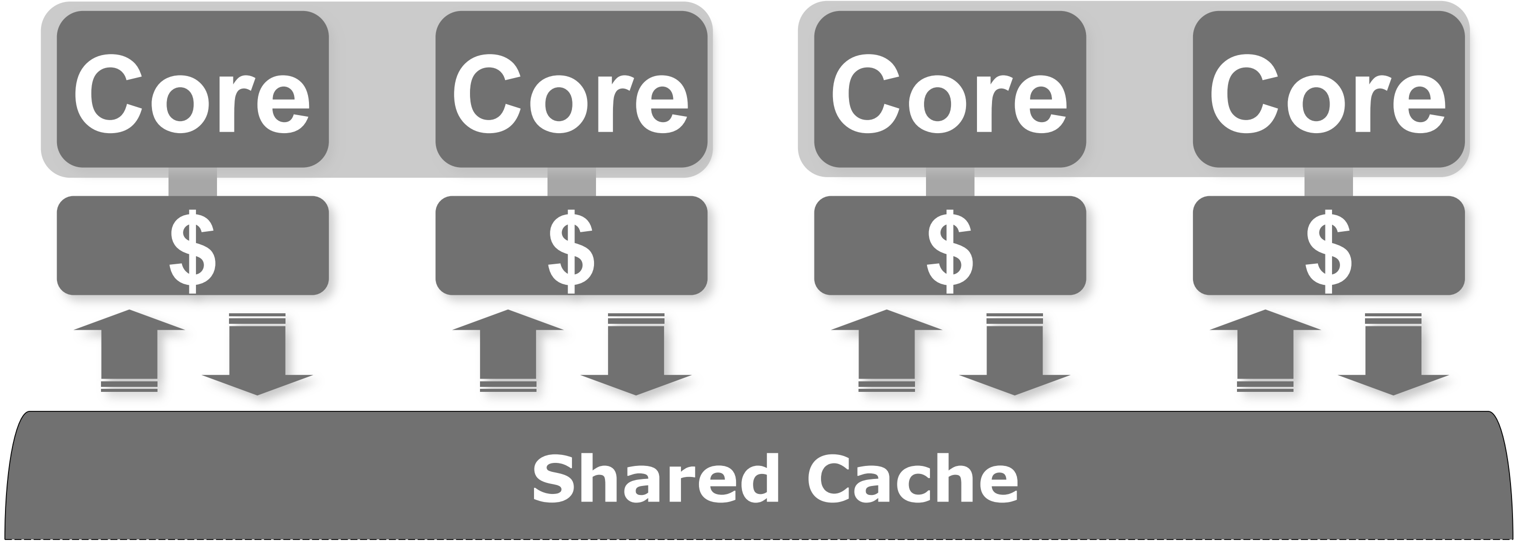
- Periodically stress test system
  - ❑ initialize processor and load fault tests
  - ❑ Lower voltage, increase frequency
  - ❑ continuously monitor and summarize internal state
  - ❑ compare w/reference (e.g., RTL or unstressed core)



Signature comparison exposes faults

Reunion: Fingerprinting DMR

[Micro'06]

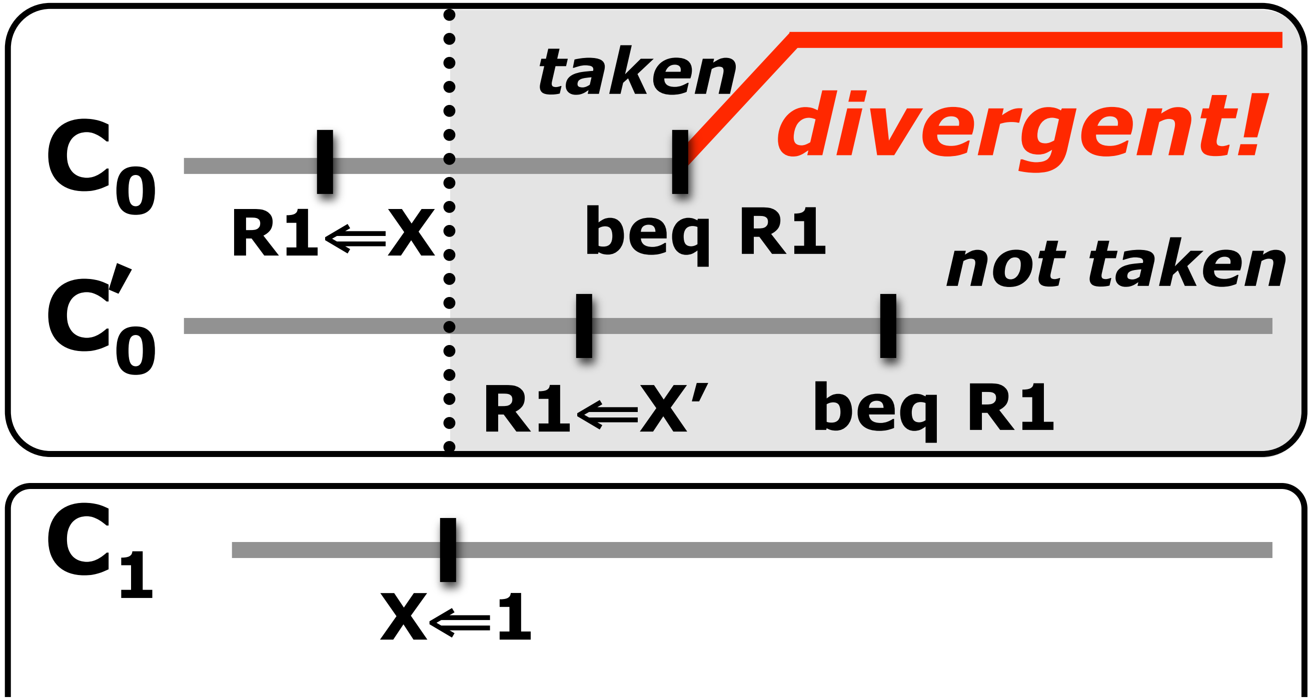


N-way CMP → N/2-way reliable CMP

Use on-chip cache hierarchy to supply memory

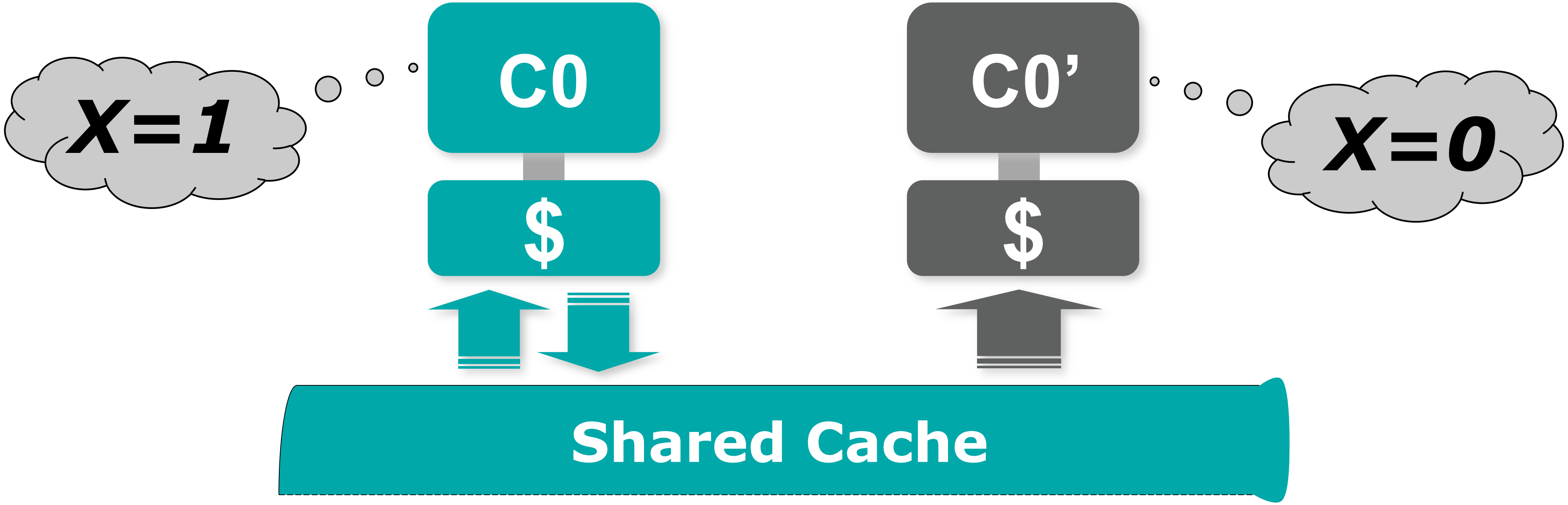
- ❑ minimizes complexity (no need for custom queues)
- ❑ but, we need same input at the same time

Load Value Incoherence



Challenge: making redundant cores agree on inputs

Detecting Load Value Incoherence

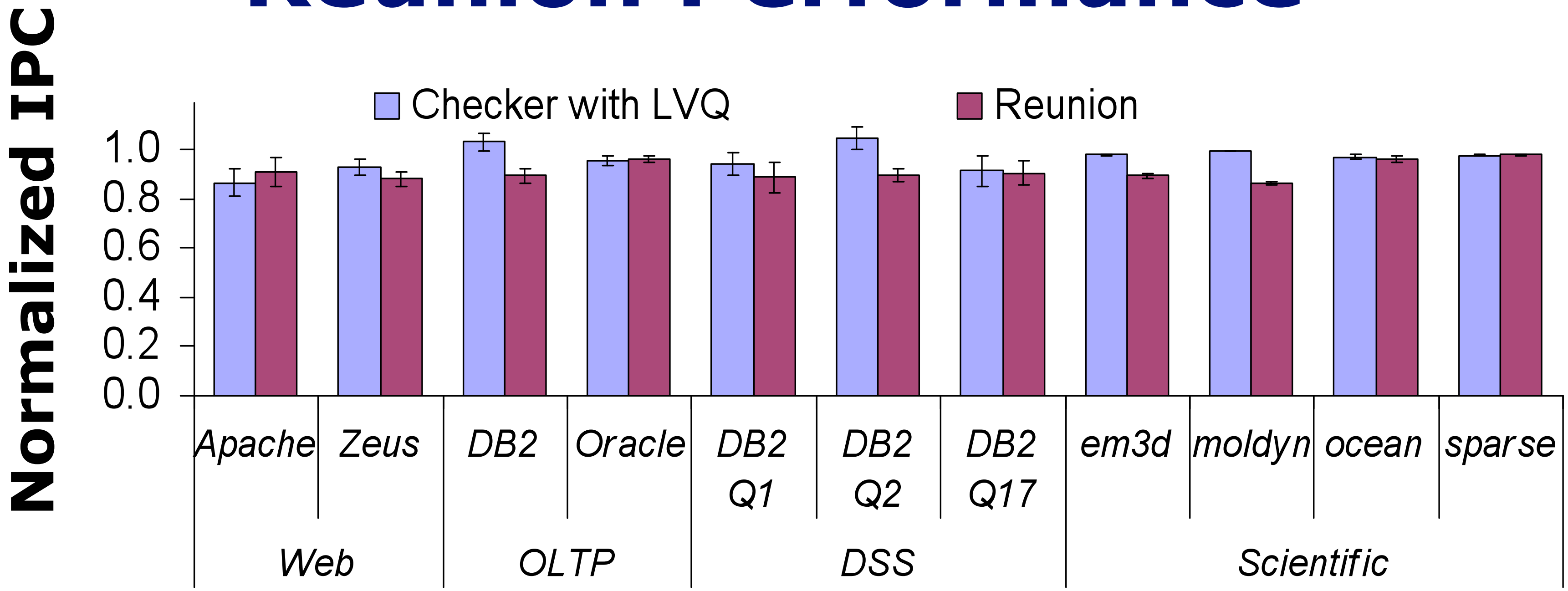


Cores disagree on a load value

- ❑ Appears as difference in retiring register values
- → Fingerprint mismatch (as in soft error)!

One mechanism detects both soft errors and load value incoherence

Reunion Performance



Reunion incurs a small performance overhead

- Slip between cores exposed at serializing events
- More requests at shared cache

Incremental performance cost for a design without strict input replication hardware

DMR across chips

Fingerprinting has minimal overhead

Can run Reunion across chips or in a distributed system

- As long as two threads do not synch often, can have threads far apart
- Machine isolation is key in many reliability applications

Have working design for a multi-chip system

Other examples of signature-based techniques

Argus [Sorin, et al., Top picks 07]

Use distributed checker logic

- ❑ Check control-flow & data-flow using signatures
- ❑ Compute correctly (adds, multiplies, etc.)
- ❑ Interact correctly with memory (loads, stores)

Enables comprehensive error detection in a single core!

Architectural Support for Monitoring in Software

Blackboxes record crashing of planes

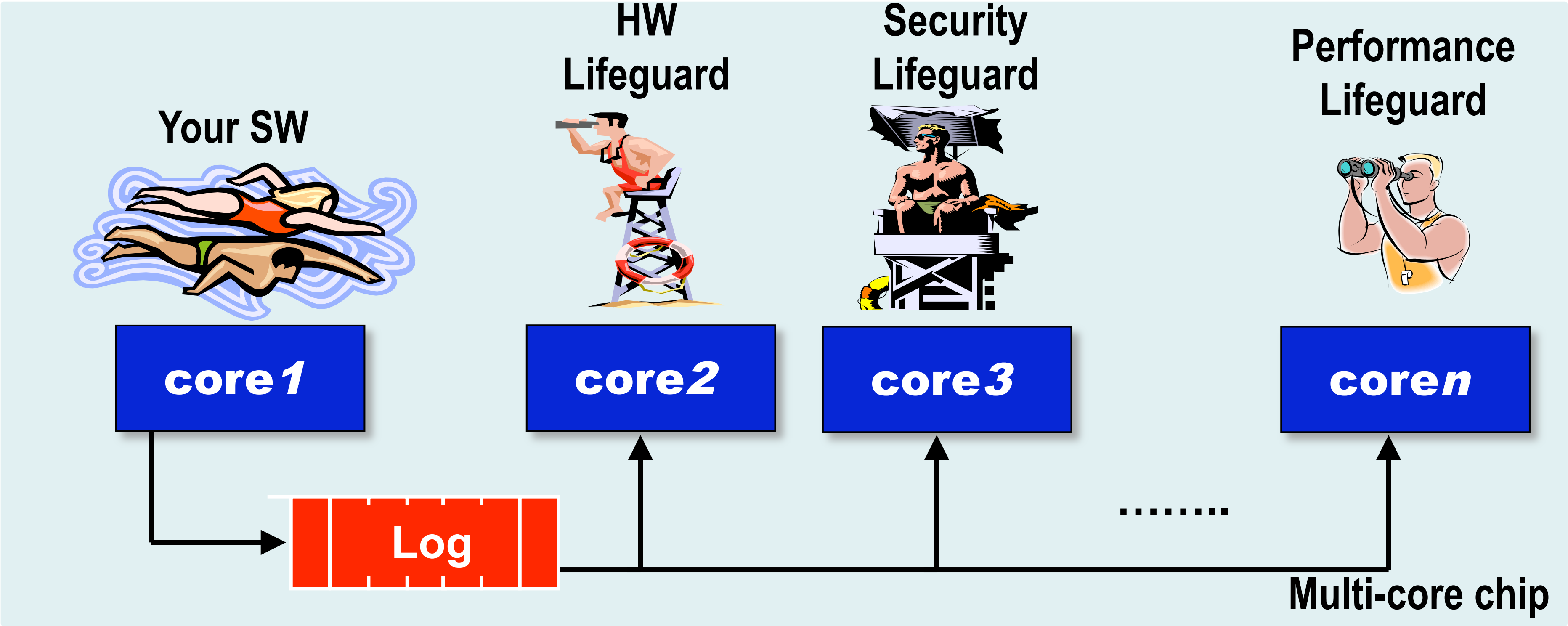
- Why can’t machines provide “execution” recorder?
- Wouldn’t it be nice for machines to allow replay?

Systems may crash because of SW or HW bugs or security attacks

- Monitoring may detect (and correct) bugs

Example: Logs & Lifeguards

[IEEE Top Pick 08]



Your SW

HW Lifeguard

Security Lifeguard

Performance Lifeguard

core*1*

core*2*

core*3*

core*n*

Log

……..

Multi-core chip

Store/examine “log” of execution

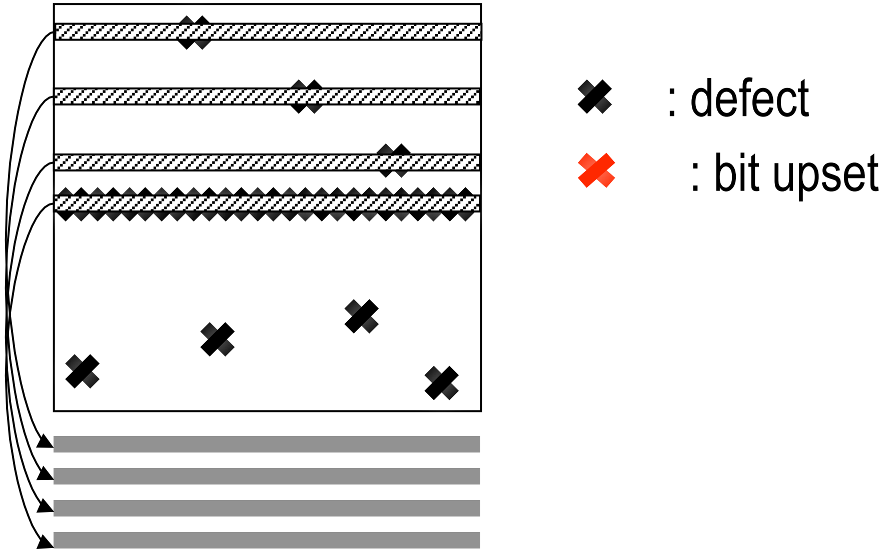
- Support a broad range of monitors (“lifeguards”)
  - Can monitor functionality (HW & SW) and performance
    - Unify HW & SW debugging
  - Great use of lots of cores on chip

Outline

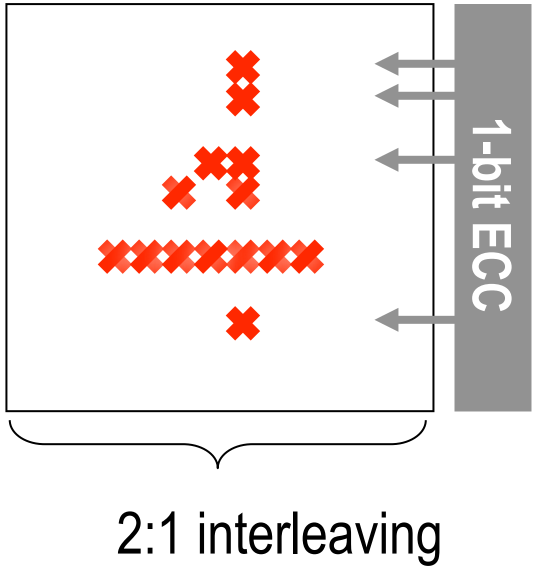
- ~~Overview~~
- Computers with unruly transistors
- Detecting/correcting error in logic
- Detecting/correcting error in memory

Conventional memory protection

Small amount of redundancy



Small-scale error correction



Can’t detect large-scale defects

Can’t repair large-scale error

Significant overhead for high coverage

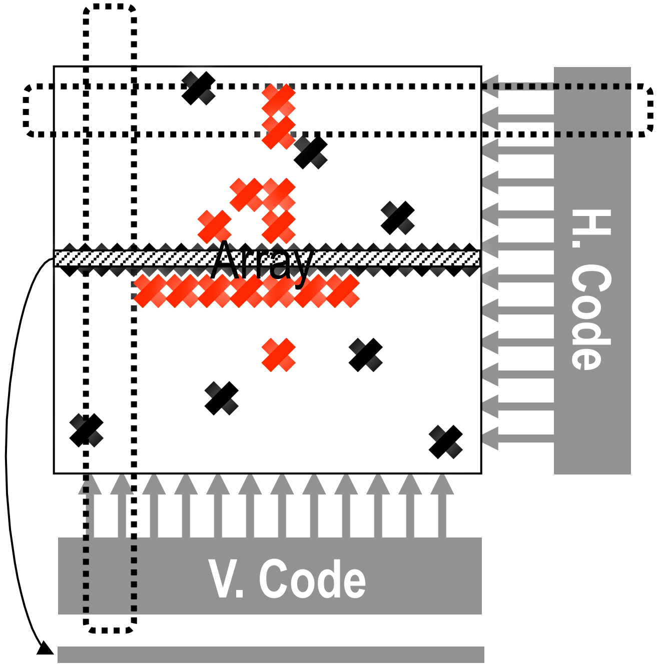
- Multi-bit ECC
  - Large area overhead
  - High power overhead
  - Long latency

- High degrees of bit interleaving
  - Only clustered error coverage
  - High power overhead

- Larger amount of hardware redundancy
  - Large area overhead for high defect coverage

No low-overhead solution for high-density defects and large-scale multi-bit error coverage

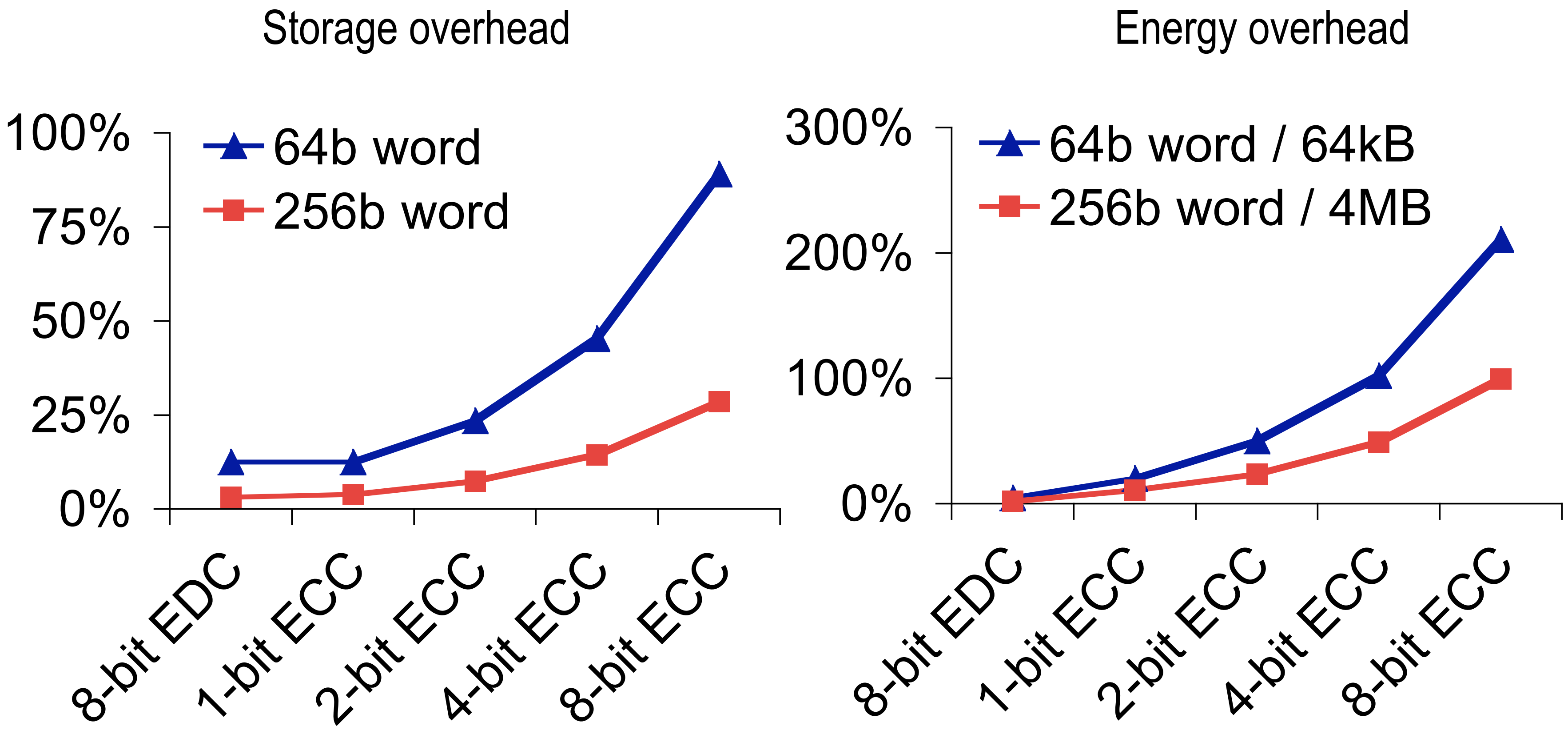
2D error coding [Micro 07]



- ❑ Fast horizontal coding
  - Multi-bit error detection
  - Optional small-scale correction
- ❑ Vertical coding in background
  - Also low-overhead code
  - Large-scale correction (with H. code)
- ❑ Less hardware redundancy
  - Repair only large-scale defects

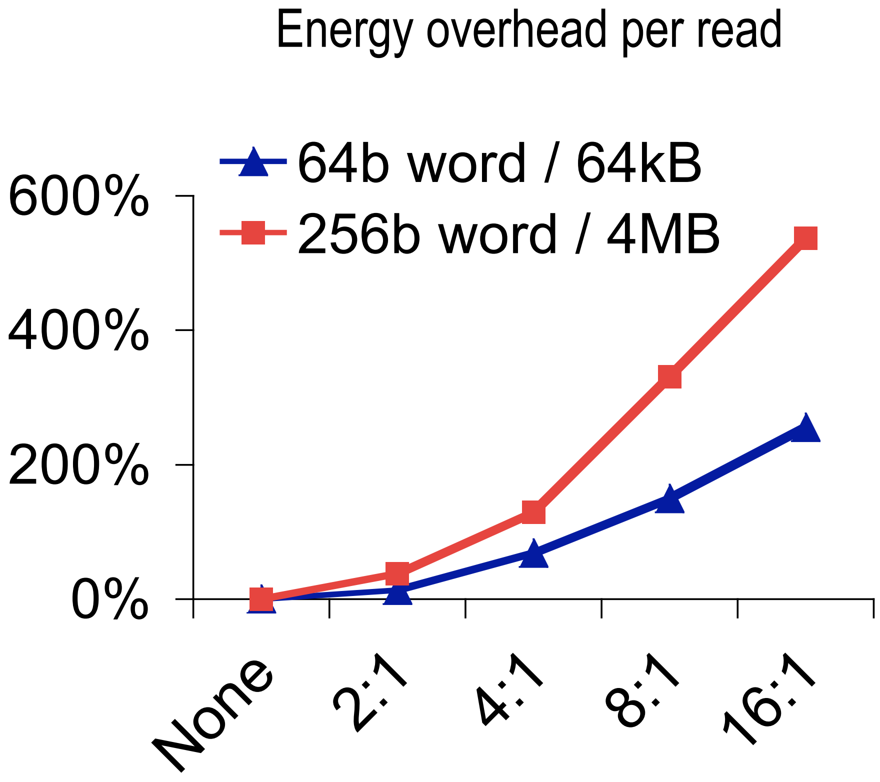
Higher multi-bit error coverage
Higher defect coverage
Lower VLSI overhead

Multi-bit ECC does not scale



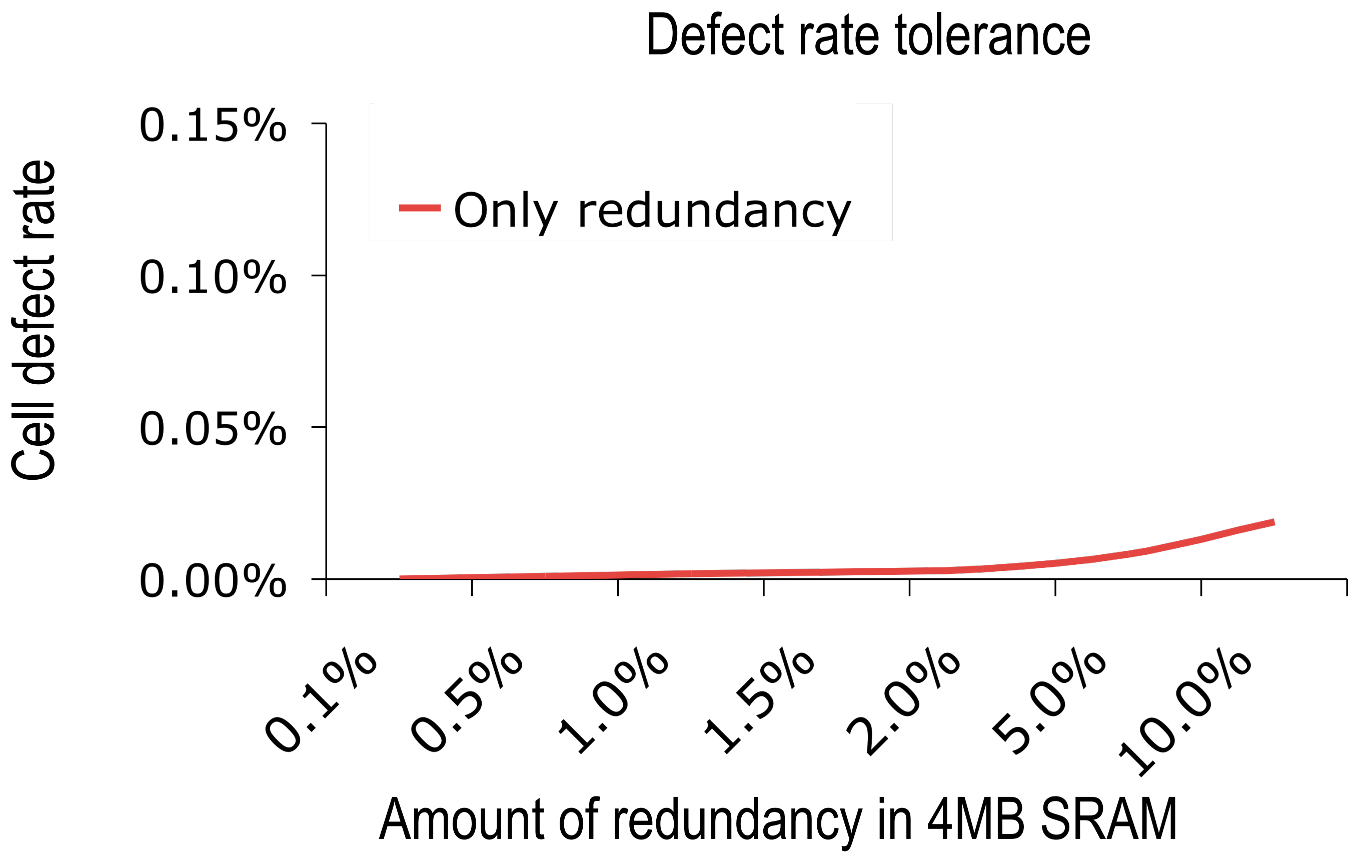
Significant increase in area and energy

Bit interleaving does not scale



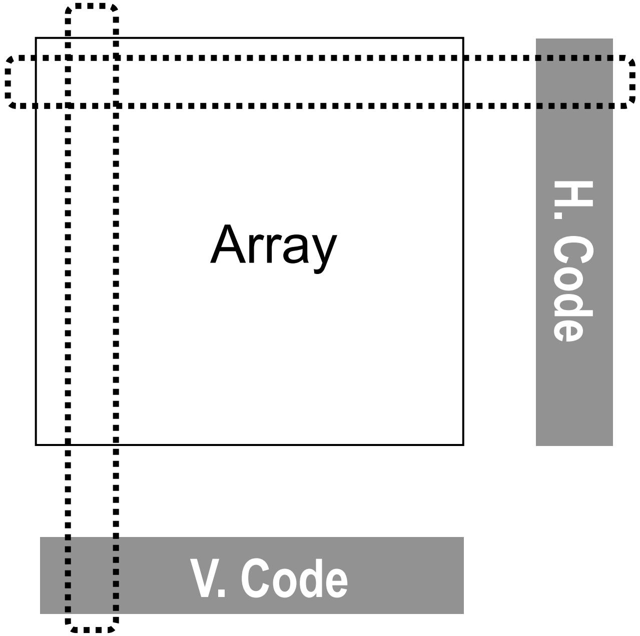
Significant increase in energy

Hardware redundancy does not scale



Low defect tolerance even with large redundancy

2D coding: concept



❑ Horizontal code

- Multi-bit error detection (e.g., logically interleaved parity)
- Optional small-scale correction
- Fast common-case operation

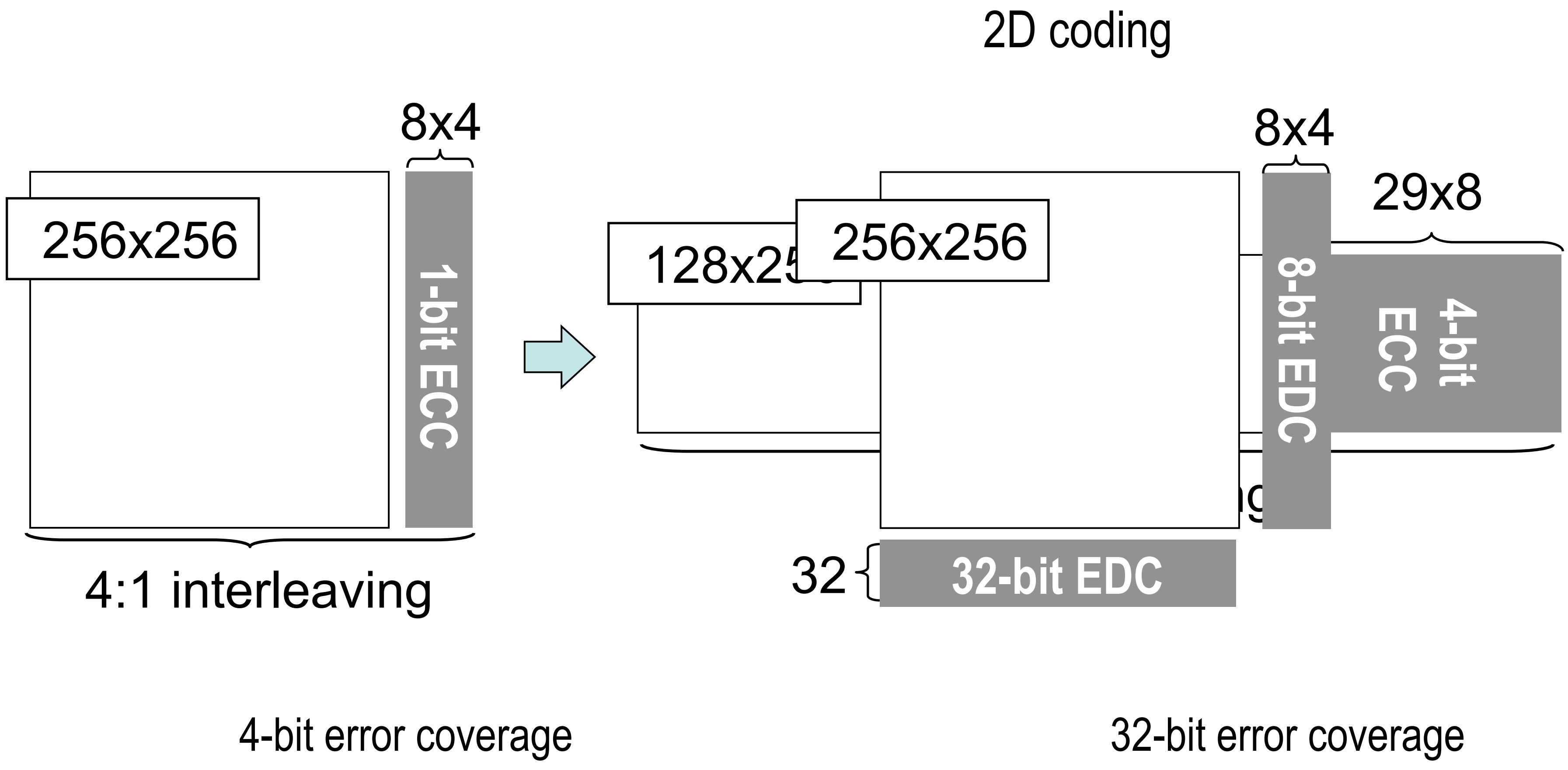
❑ Vertical code

- Multi-bit error detection (e.g., logically interleaved parity)
- Updated in background

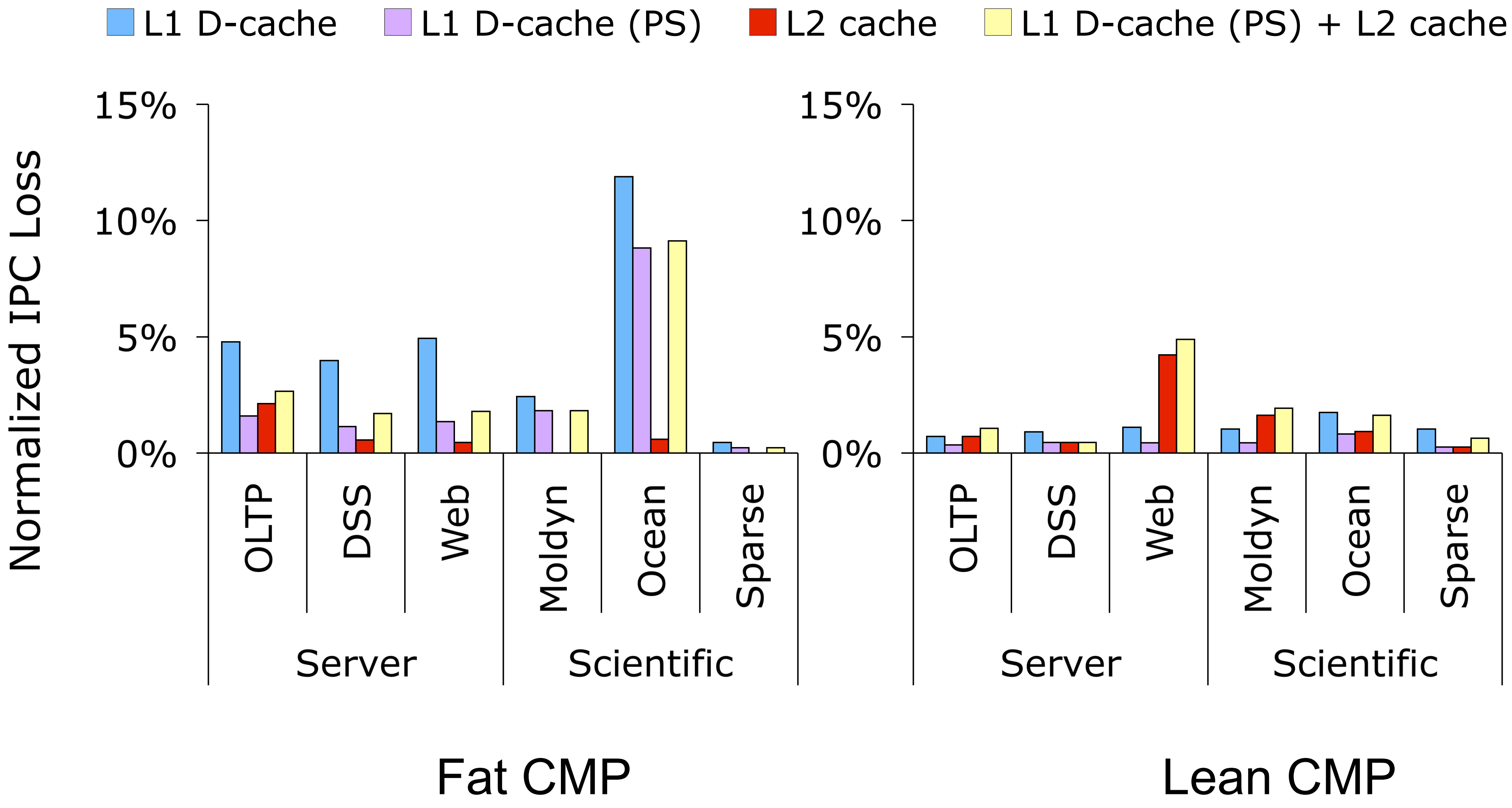
Combining two low-overhead coding

➔ Effective multi-bit error correction

2D coding: scalable protection

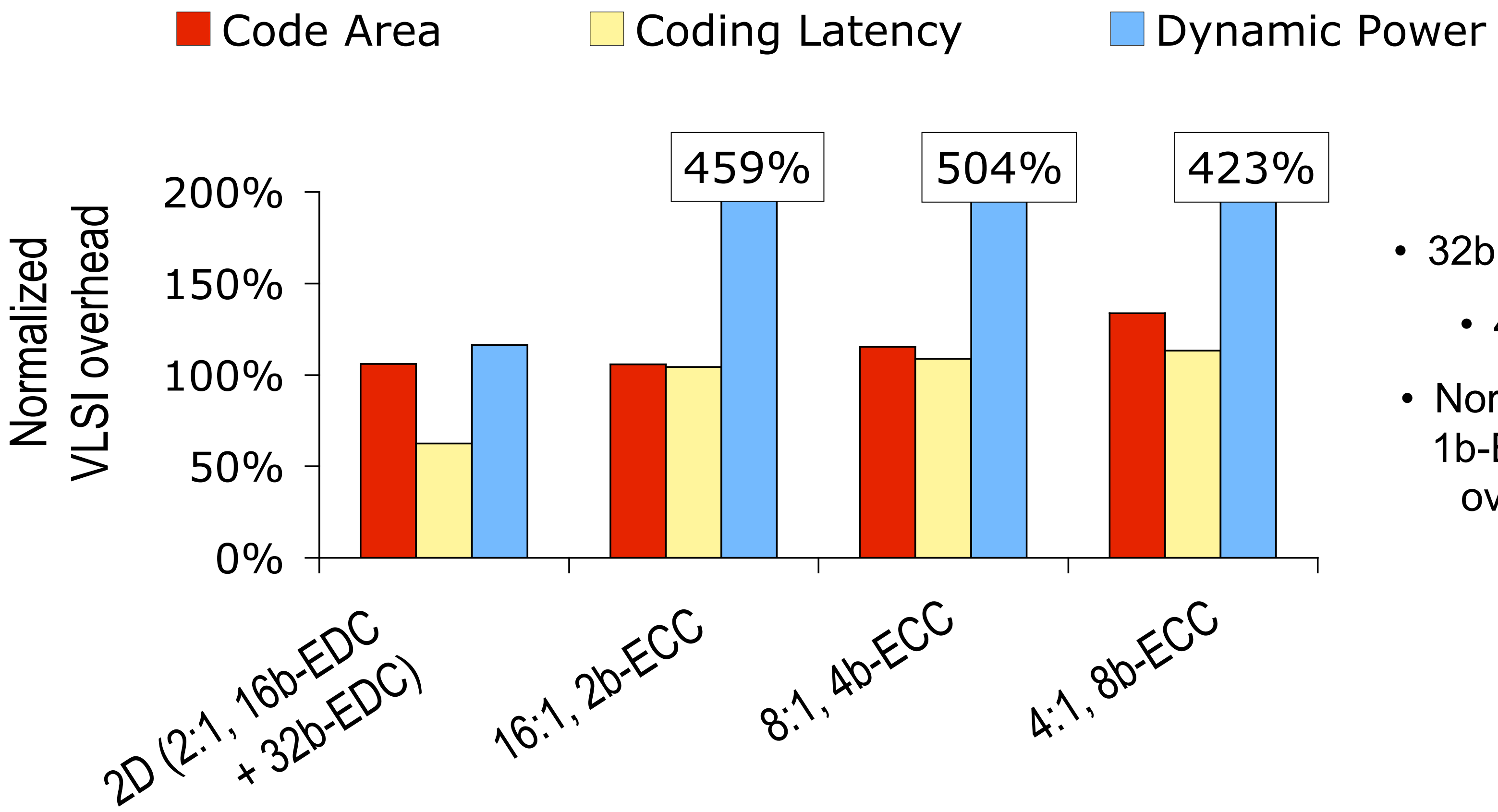


Architectural performance overhead



Overall average performance loss < 3%

VLSI overhead



- 32b error coverage
- 4MB SRAM
- Normalized to 2:1, 1b-ECC protection overhead costs

2D coding incurs much less VLSI overheads

Other techniques

Remapping of cells:

- Under aggressive voltage scaling: Wilkerson et al., Top Picks ’08
- And/or when high defect rates with erasure codes

DRAM memory

- Chipkill, distributed parity, ….

Summary

These are best of times I can imagine for computer system designers & architects

- Must build reliable systems from unreliable components
- Need cheap mechanisms, configured only when needed
- There are no silver bullets → these are great times for academia to lead and have impact

Thank you!

Visit our website:

http://parsa.epfl.ch/babak.falsafi

EPFL

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

PARSA

Parallel Systems Architecture Lab

EPFL

www.c2s2.org