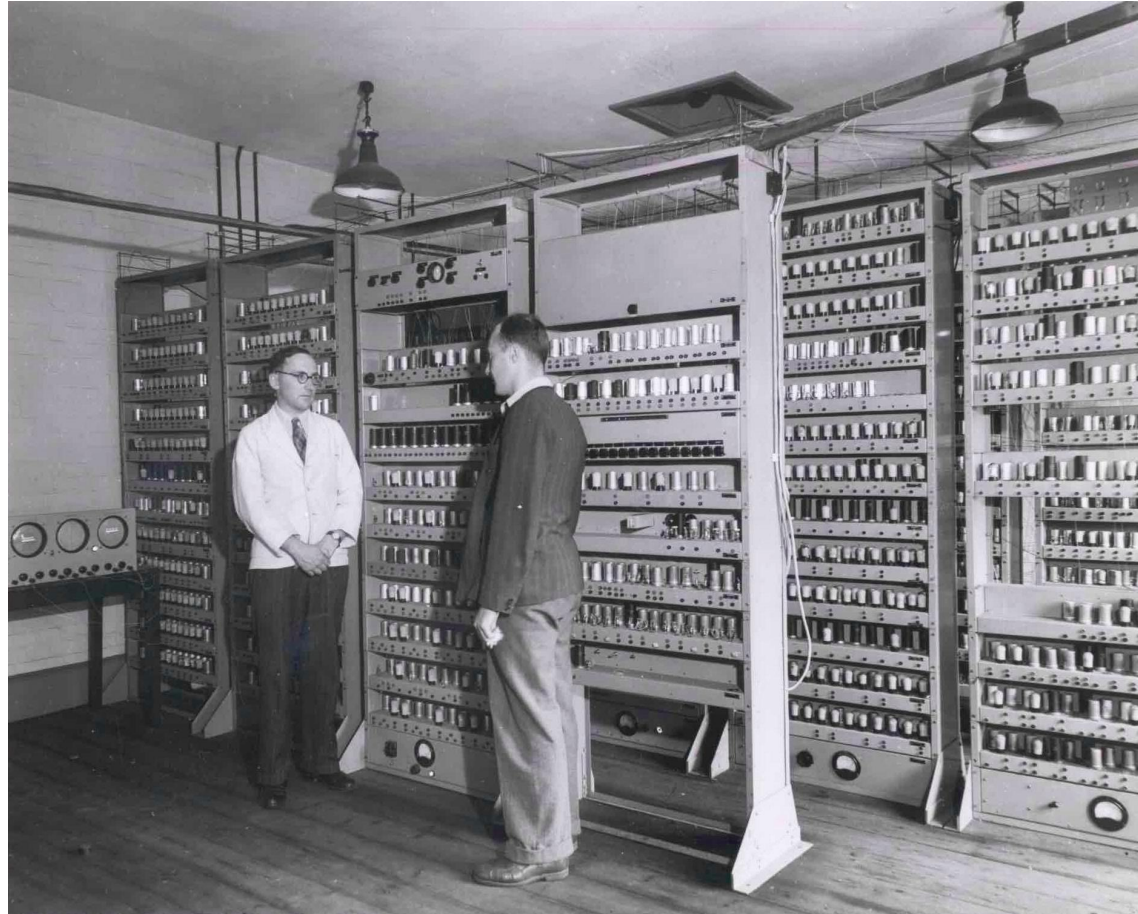


Reliability

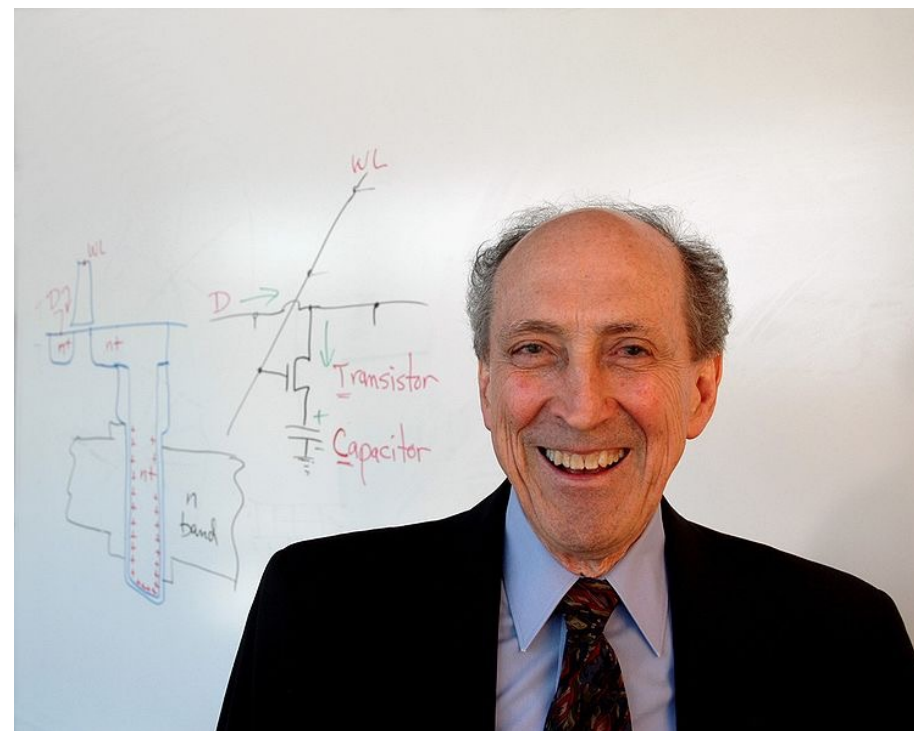
Advanced Topics in Computer Architecture

Timothy Jones

Historic reliability



Silicon trends



Why we care now

- Microprocessors are increasingly used in situations where we want to be sure of their correctness
 - Self-driving cars, nuclear power stations, medical devices, etc
- Many industrial sectors mandate the use of error-detection strategies
 - For example, ASIL standards in automotive
- With increased susceptibility to faults, even non-safety-critical computing starts to require fault tolerance

<https://perspectives.mvdirona.com/2009/10/you-really-do-need-ecc-memory/>

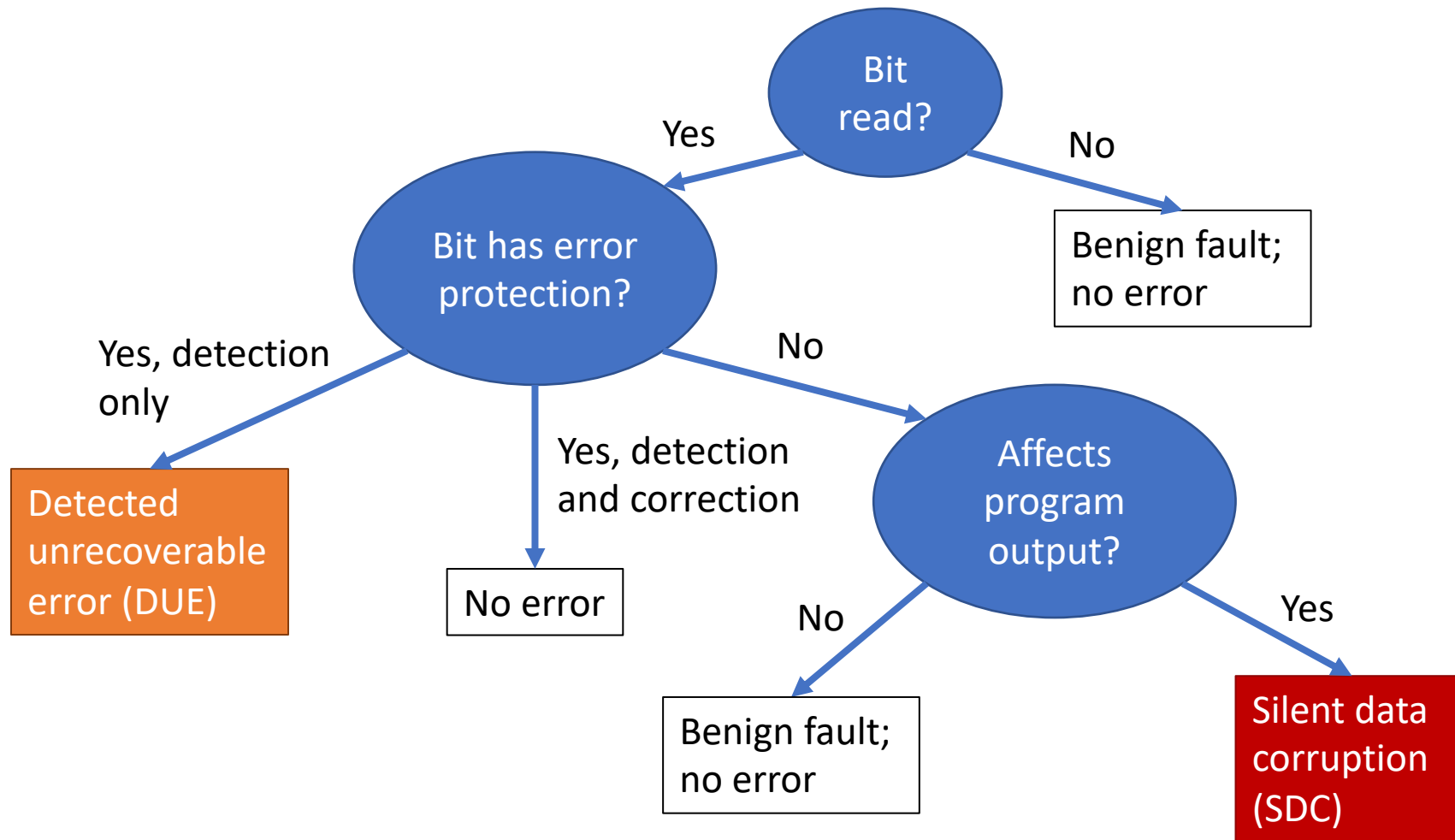
Hard errors

- Permanent errors that affect operation
- Caused by device wearout in-the-field
- Also can occur from manufacturing variabilities

Soft errors

- Transient errors that can affect operation
 - They are transient because their effects don't last
 - They are not repeatable
- Caused by
 - Alpha particle strikes
 - Cosmic rays!

Error manifestation



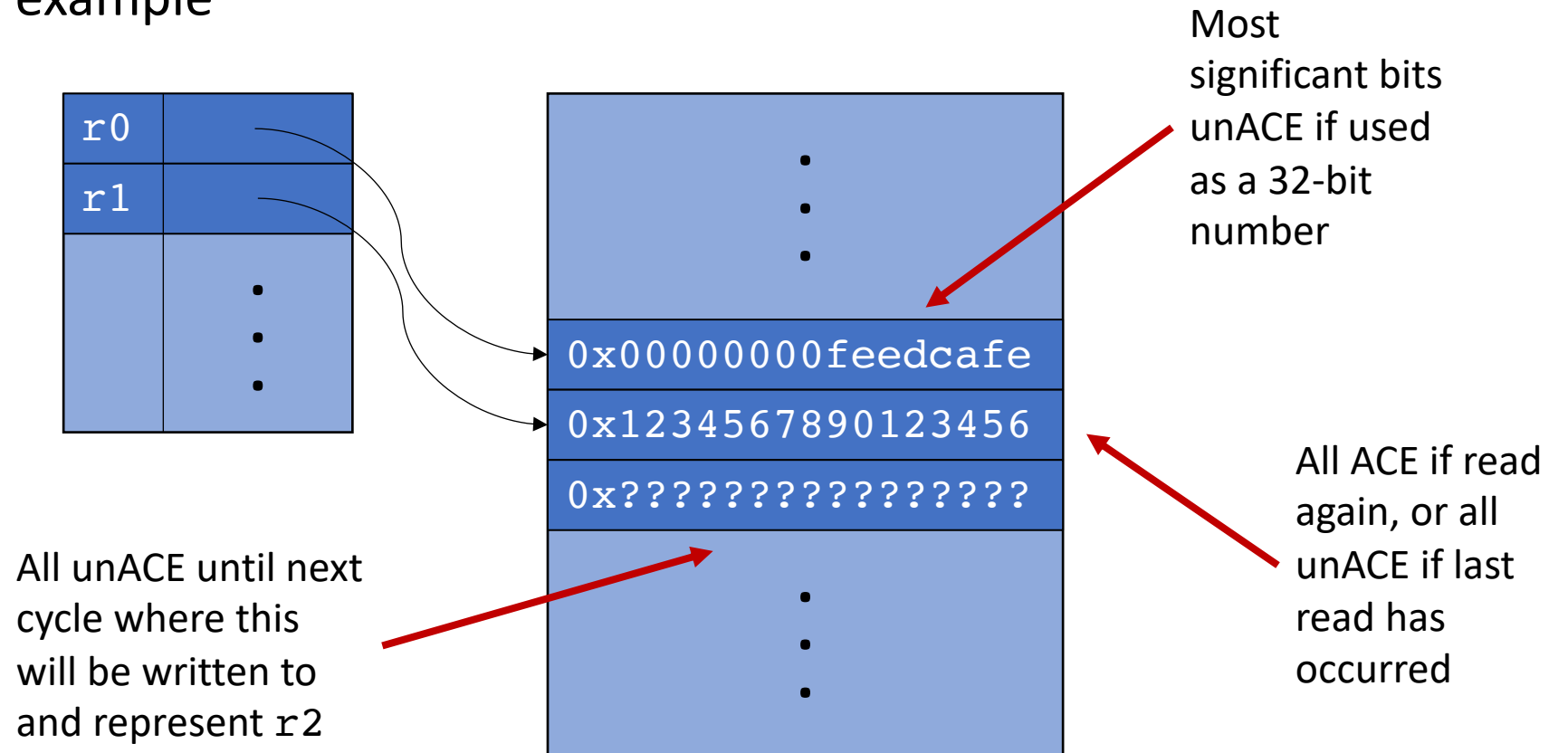
Identifying vulnerabilities

- We can perform an analysis of processor structures to identify vulnerable state
- We identify the bits that are required for architecturally correct execution (ACE)
- These bits *could* result in incorrect output if they were flipped
- The architectural vulnerability factor (AVF) is a useful metric

$$AVF = \frac{\sum_{i=0}^{ncycles} ACEbits}{ncycles * nbits}$$

Identifying vulnerabilities

- Bits can be ACE in some cycles, not ACE in others
 - Registers, for example



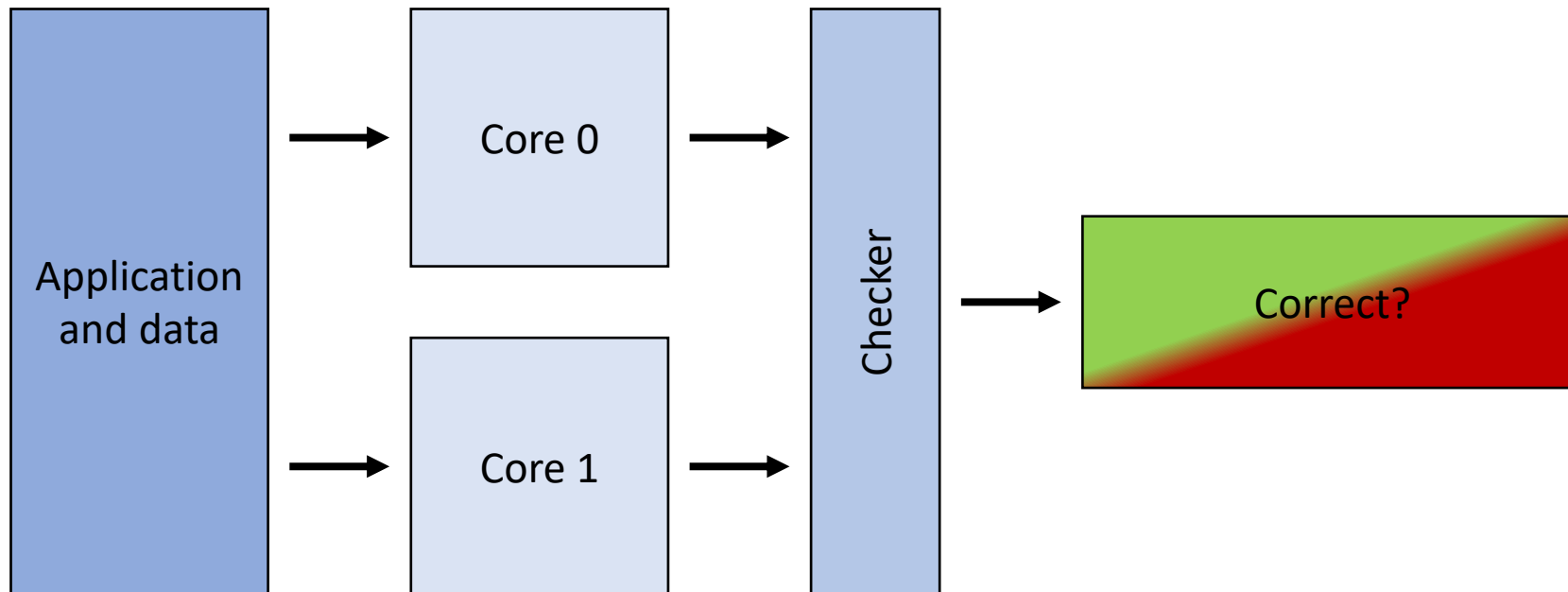
Metrics

- Two related metrics are often used to define reliability
- The FIT rate (failures in time)
 - Defined as the total number of errors per billion device hours
- MTTF (mean time to failure)
 - Represents the time between two errors

$$MTTF \sim \frac{1}{FIT}$$

Dual-core lockstep

- In a system with dual-core lockstep, a program is run twice on different cores
 - Results compared at each cycle
 - Introduces temporal and spatial redundancy into the system

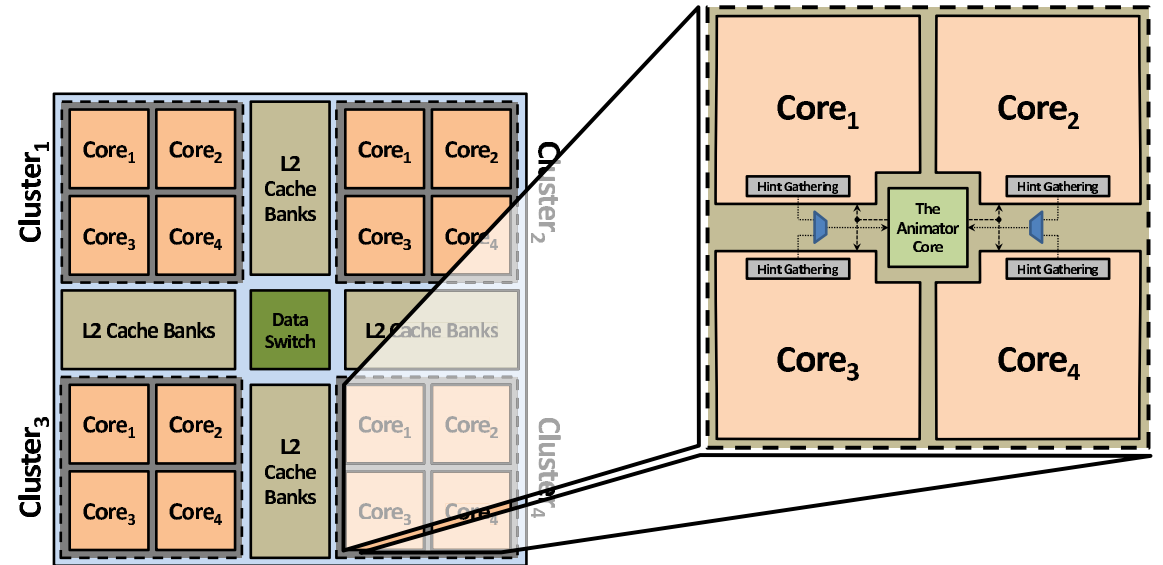


Redundant multithreading

- Run two versions of code and compare results
 - Can be a software scheme, perhaps with some hardware support
 - Or a purely hardware approach
- Can run on different cores with one passing the other data
- Or the same core, within a different SMT context

Taking advantage of faulty hardware

- Some systems use the faulty core to provide hints to others
- For example, *Necromancer: Enhancing System Throughput by Animating Dead Cores*
Ansari, Feng, Gupta and Mahlke
ISCA 2010



Approximate computing

- In certain situations we can embrace errors



Summary

- Reliability is a problem that has come back to haunt us
- Required for safety-critical systems
 - Increasing needed / desired in others too
- A variety of techniques developed to
 - Identify which parts of the core are vulnerable
 - Reduce vulnerability to errors by re-executing parts of the code
 - Embrace the unreliability for performance

This week's papers

- BlackJack: Hard Error Detection with Redundant Threads on SMT
 - Schuchman and Vijaykumar, DSN 2007
- Utilizing Dynamically Coupled Cores to Form a Resilient Chip Multiprocessor
 - LaFrieda, İpek, Martínez and Manohar, DSN 2007
- StageWeb: Interweaving Pipeline Stages into a Wearout and Variation Tolerant CMP Fabric
 - Gupta, Ansari, Feng and Mahlke, DSN 2010