



# Reliable Architectures

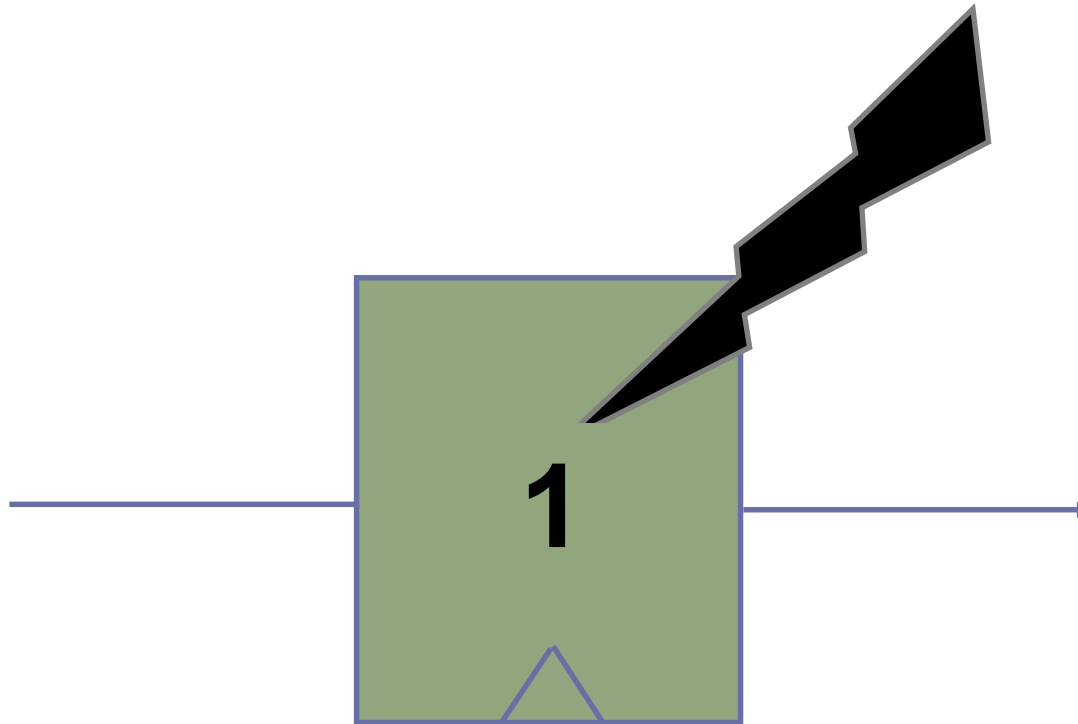
Joel Emer

Computer Science and Artificial Intelligence Laboratory  
Massachusetts Institute of Technology

<http://www.csg.csail.mit.edu/6.823>

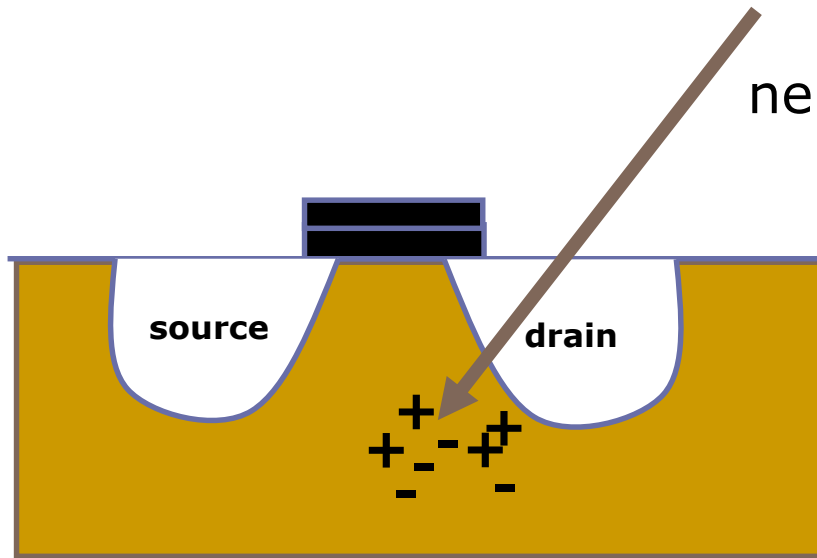
# Strike Changes State of a Single Bit

---



# Impact of Neutron Strike on a Si Device

---



neutron strike

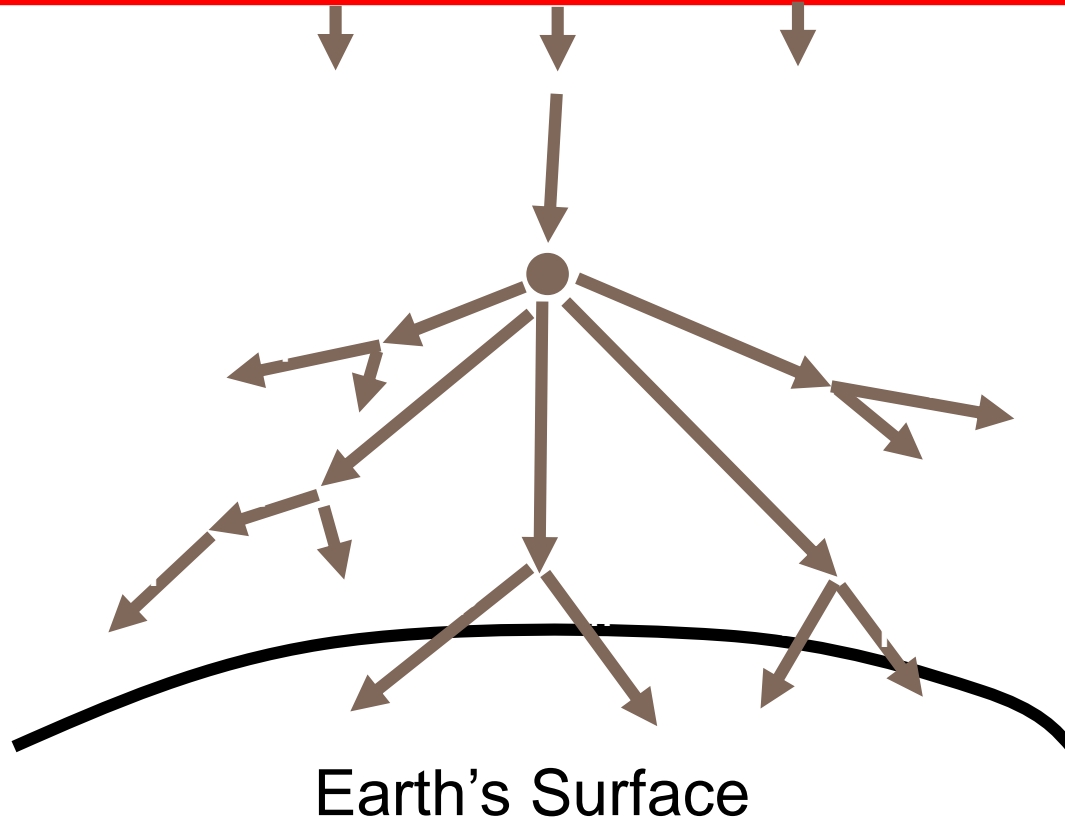
Strikes release electron & hole pairs that can be absorbed by source & drain to alter the state of the device

Transistor Device

- Secondary source of upsets: alpha particles from packaging

# Cosmic Rays Come From Deep Space

---



- Neutron flux is higher at higher altitudes
  - 3x - 5x increase in Denver at 5,000 feet
  - 100x increase in airplanes at 30,000+ feet

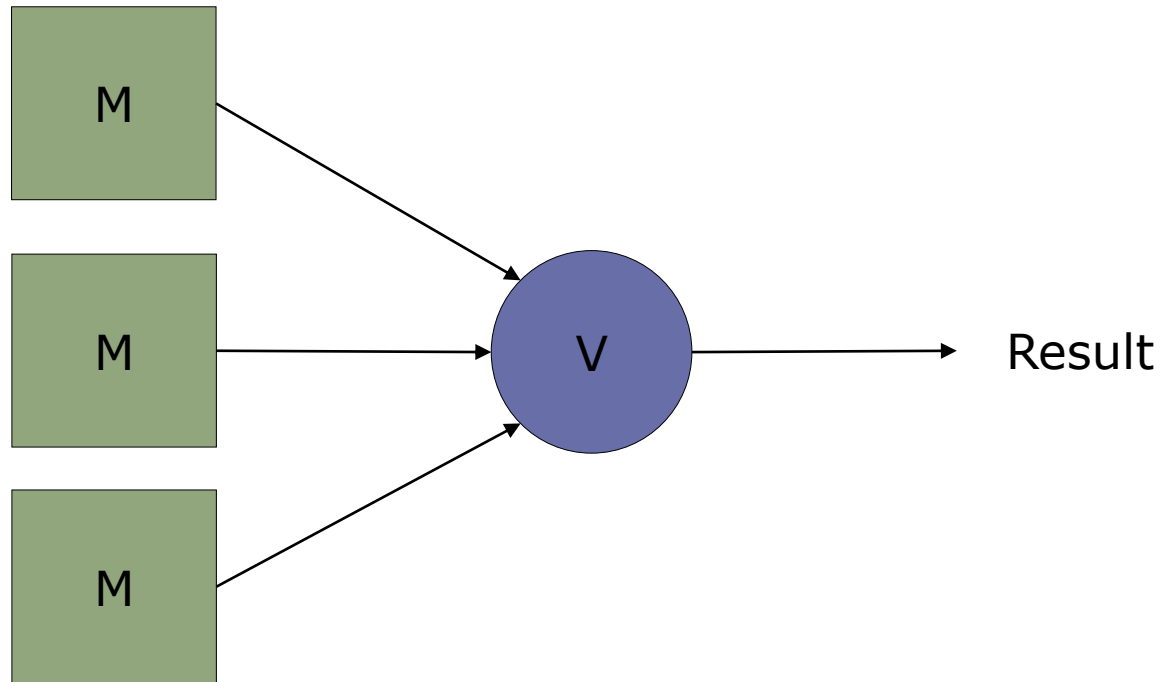
# Physical Solutions are hard

---

- Shielding?
  - No practical absorbent (e.g., approximately  $> 10$  ft of concrete)
  - This is unlike Alpha particles which are easily blocked
- Technology solution: SOI?
  - Partially-depleted SOI of some help, effect on logic unclear
  - Fully-depleted SOI may help, but is challenging to manufacture
- Circuit level solution?
  - Radiation hardened circuits can provide 10x improvement with significant penalty in performance, area, cost
  - 2-4x improvement may be possible with less penalty

# Triple Modular Redundancy (Von Neumann, 1956)

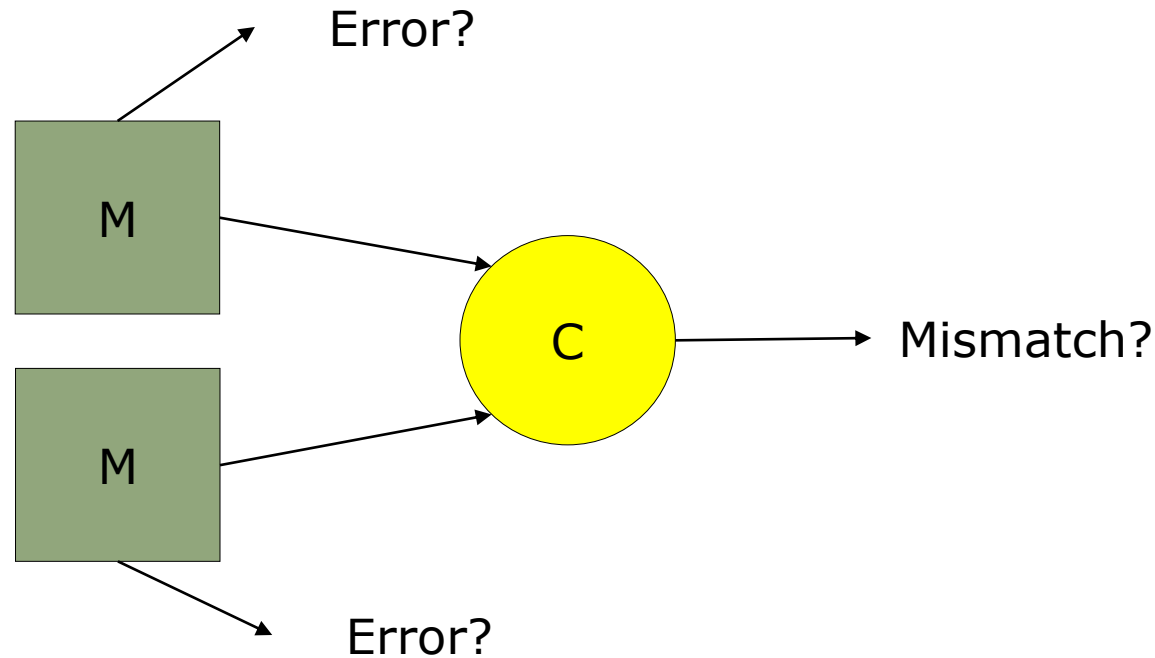
---



V does a majority vote on the results

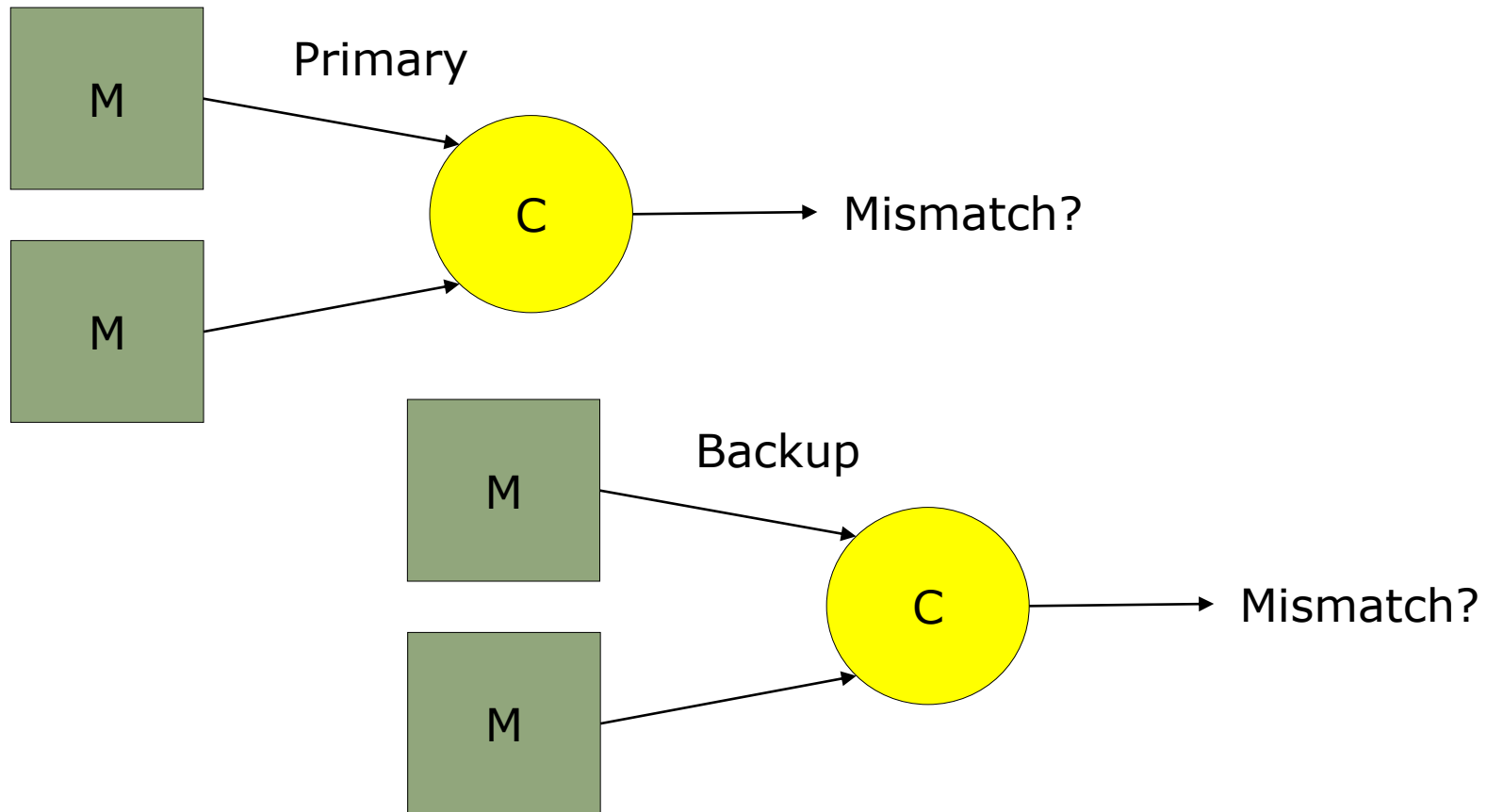
# Dual Modular Redundancy (eg., Binac, Stratus)

---



- Processing stops on mismatch
- Error signal used to decide which processor be used to restore state to other

# Pair and Spare Lockstep (e.g., Tandem, 1975)

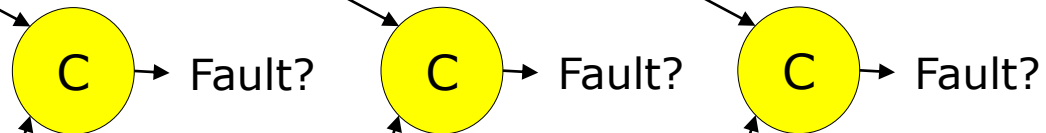


- Primary creates periodic checkpoints
- Backup restarts from checkpoint on mismatch

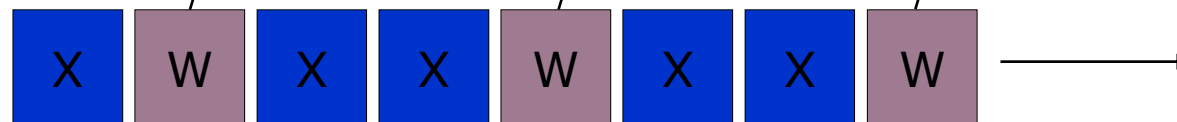


# Redundant Multithreading (e.g., Reinhardt, Mukherjee, 2000)

Leading Thread

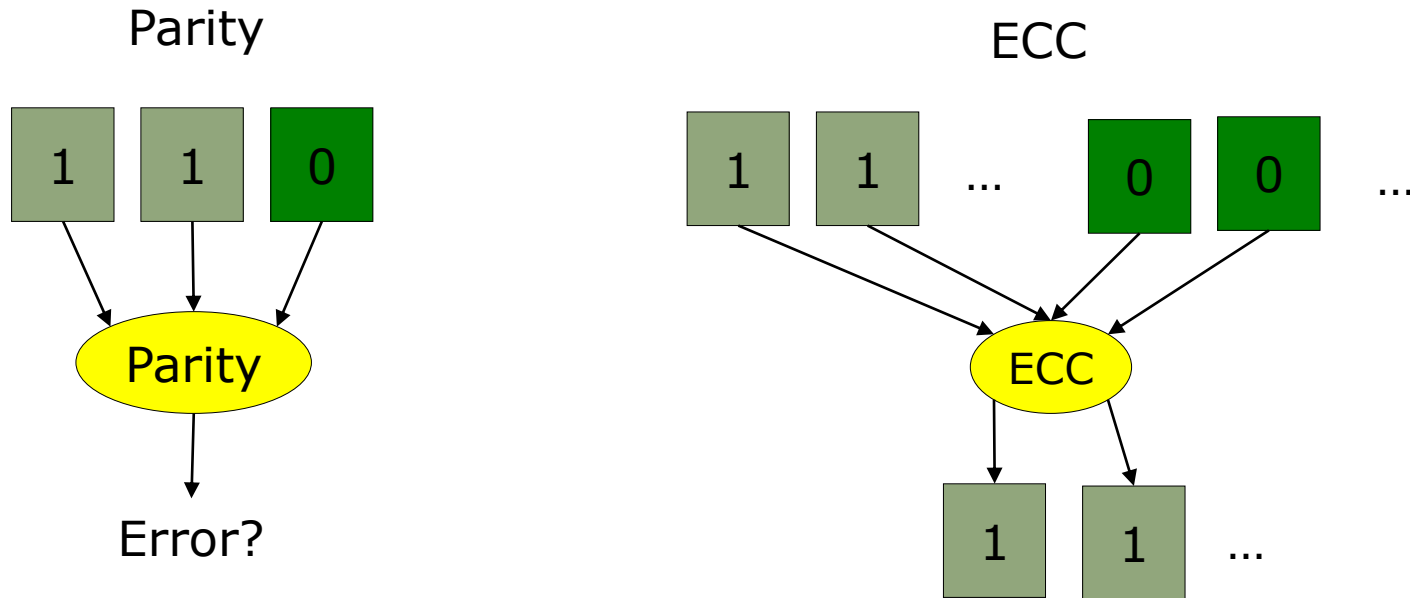


Trailing Thread



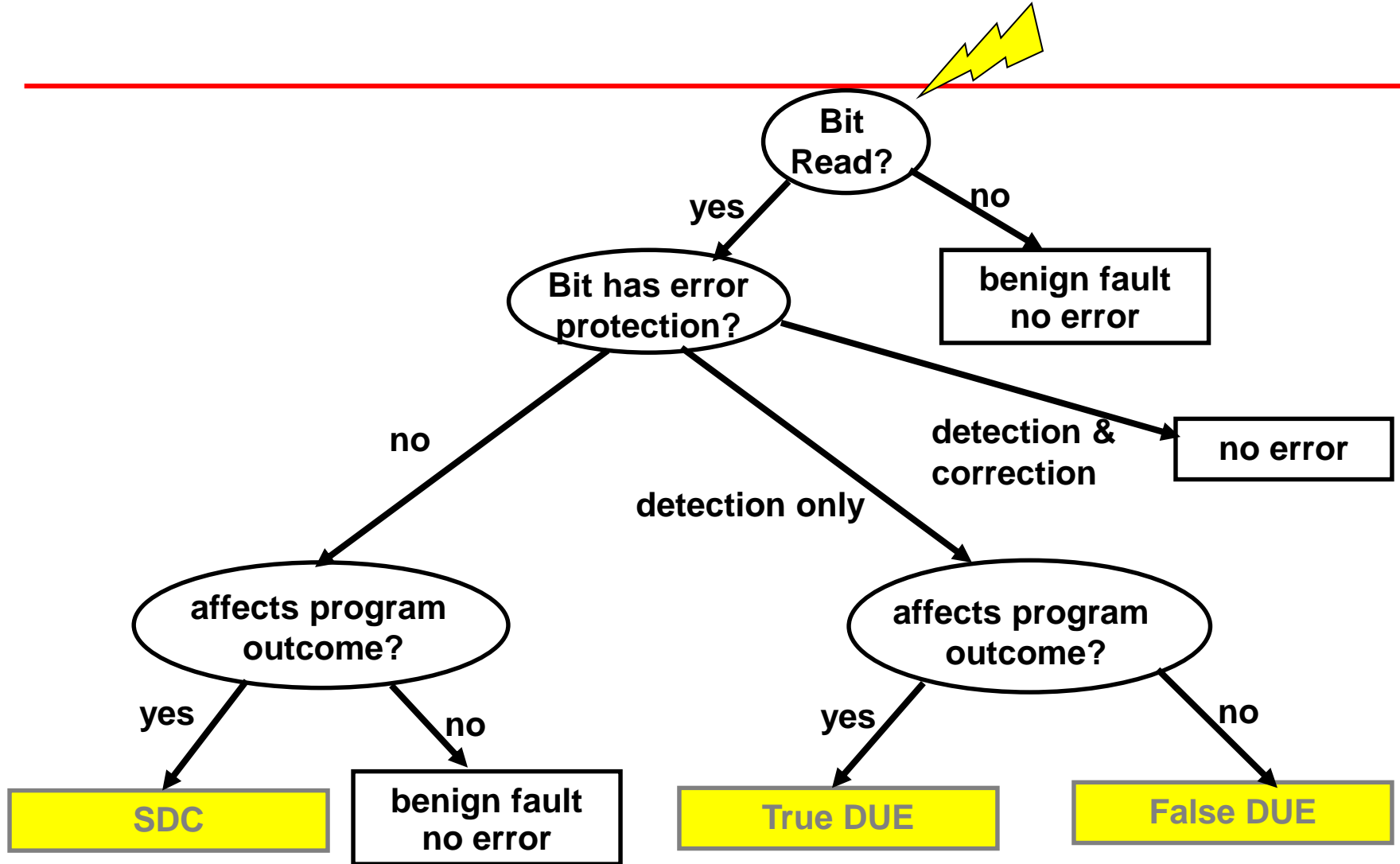
- Writes are checked

# Component Protection



- Fujitsu SPARC in 130 nm technology (ISSCC 2003)
  - 80% of 200k latches protected with parity
  - versus very few latches protected in commodity microprocessors

# Strike on a bit (e.g., in register file)

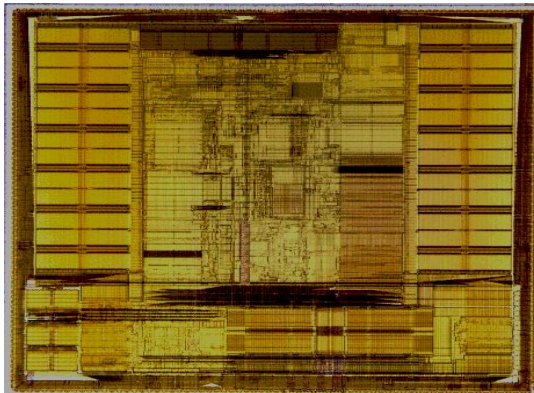


**SDC = Silent Data Corruption, DUE = Detected Unrecoverable Error**

# Metrics

---

- Interval-based
  - MTTF = Mean Time to Failure
  - MTTR = Mean Time to Repair
  - MTBF = Mean Time Between Failures = MTTF + MTTR
  - Availability = MTTF / MTBF
- Rate-based
  - FIT = Failure in Time = 1 failure in a billion hours
  - 1 year MTTF =  $10^9 / (24 * 365)$  FIT = 114,155 FIT
  - SER FIT = SDC FIT + DUE FIT



## Hypothetical Example

Cache: 0 FIT  
+ IQ: 100K FIT  
+ FU: 58K FIT

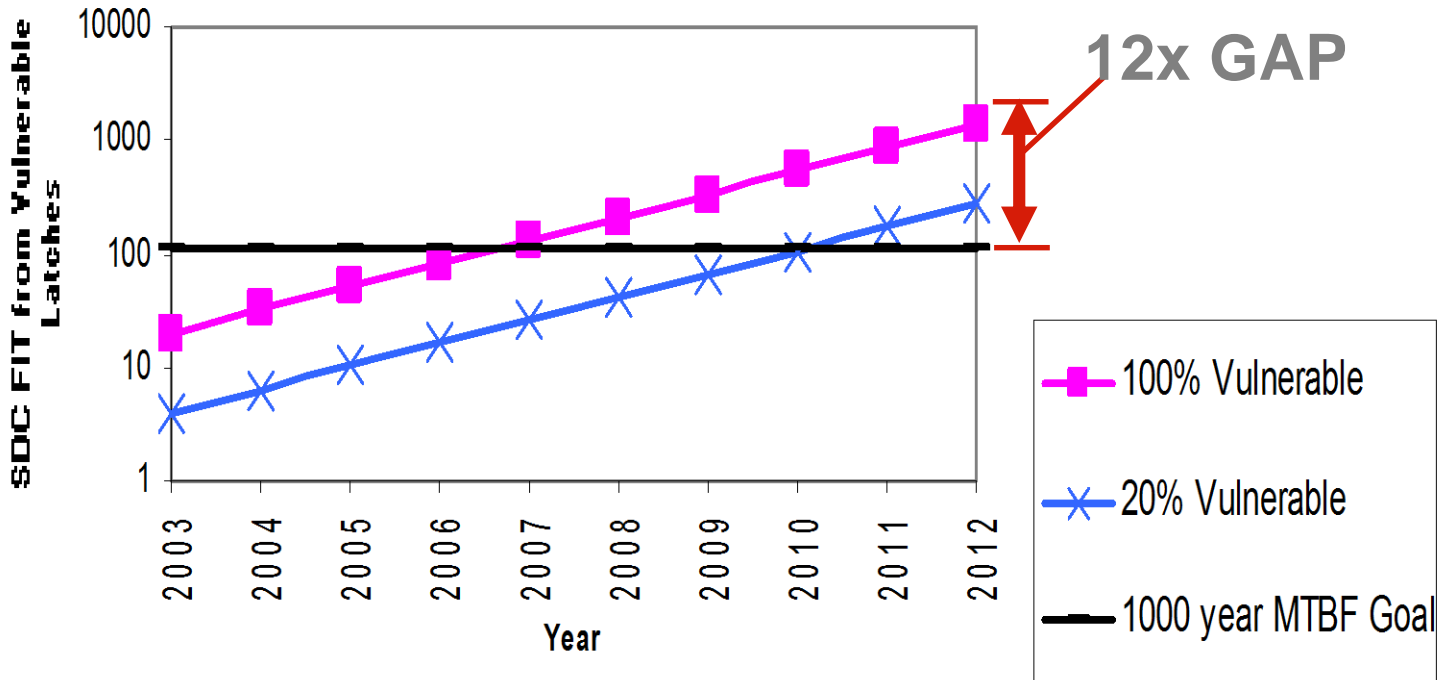
---

**Total of 158K FIT**

# Cosmic Ray Strikes: Evidence & Reaction

- Publicly disclosed incidence
  - Error logs in large servers, E. Normand, “Single Event Upset at Ground Level,” IEEE Trans. on Nucl Sci, Vol. 43, No. 6, Dec 1996.
  - Sun Microsystems found cosmic ray strikes on L2 cache with defective error protection caused Sun’s flagship servers to crash, R. Baumann, IRPS Tutorial on SER, 2000.
  - Cypress Semiconductor reported in 2004 a single soft error brought a billion-dollar automotive factory to a halt once a month, Zielger & Puchner, “SER – History, Trends, and Challenges,” Cypress, 2004.

# # Vulnerable Bits Growing with Moore's Law



Typical SDC goal: 1000 year MTBF  
 Typical DUE goal: 10-25 year MTBF

# Architectural Vulnerability Factor (AVF)

$AVF_{bit}$  = Probability Bit Matters

$$AVF_{bit} = \frac{\text{\# of Visible Errors}}{\text{\# of Bit Flips from Particle Strikes}}$$

$$FIT_{bit} = \text{intrinsic } FIT_{bit} * AVF_{bit}$$

# Architectural Vulnerability Factor

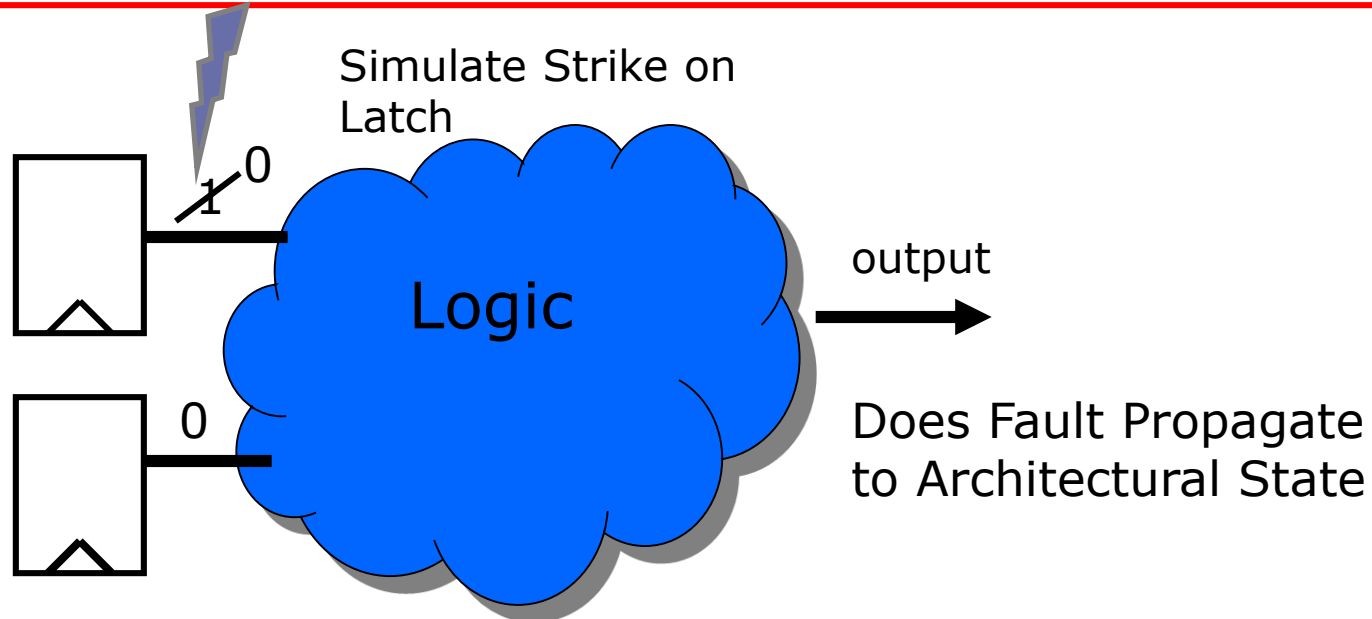
## Does a bit matter?

---

- Branch Predictor
  - Doesn't matter at all (AVF = 0%)
- Program Counter
  - Almost always matters (AVF  $\sim$  100%)

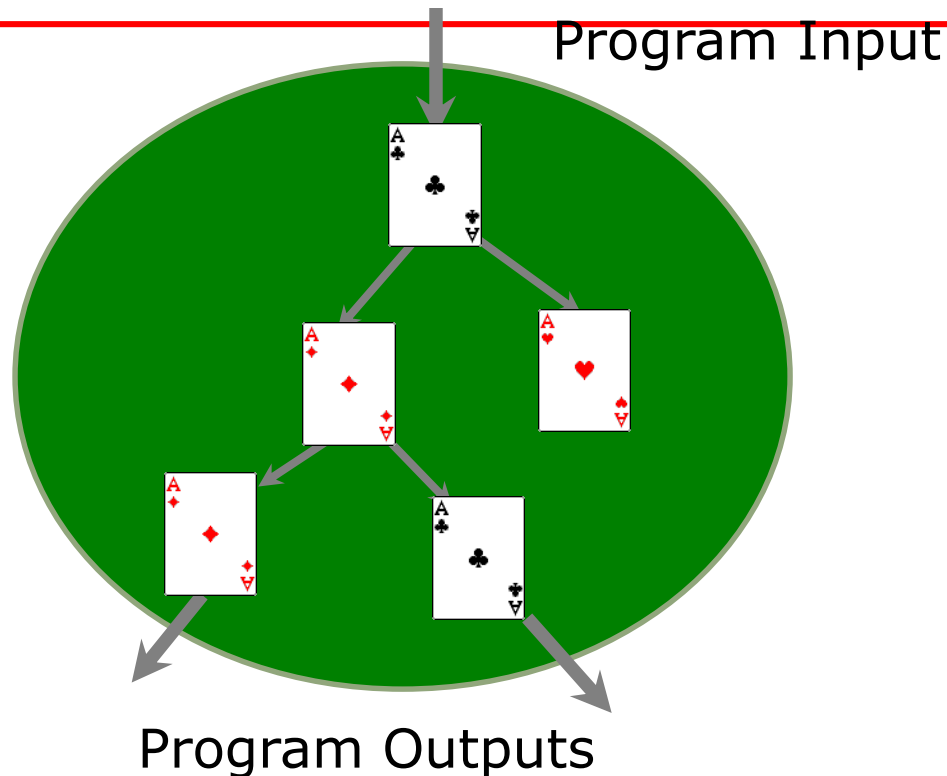


# Statistical Fault Injection (SFI) with RTL



- + Naturally characterizes all logical structures
- RTL not available until late in the design cycle
- Numerous experiments to flip all bits
- Generally done at the chip level
  - Limited structural insight

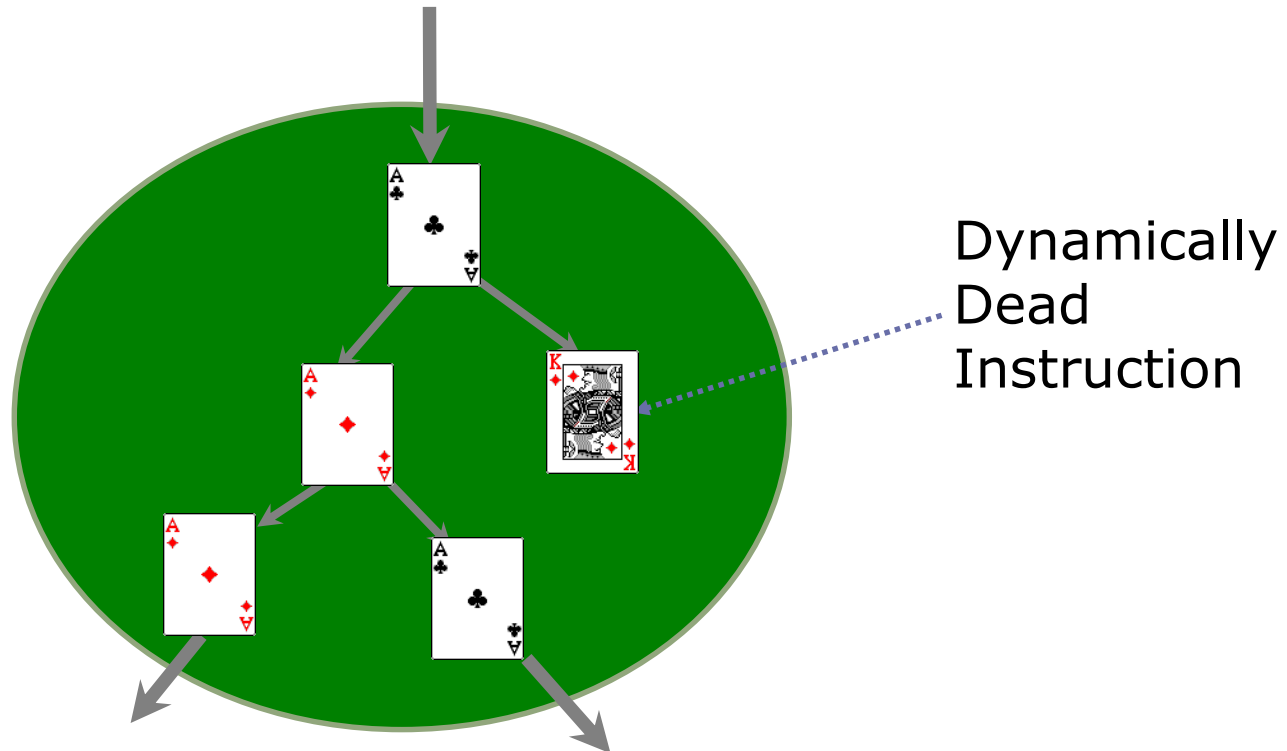
# Architecturally Correct Execution (ACE)



- ACE path requires only a subset of values to flow correctly through the program's data flow graph (and the machine)
- Anything else (un-ACE path) can be derated away

# Example of un-ACE instruction: Dynamically Dead Instruction

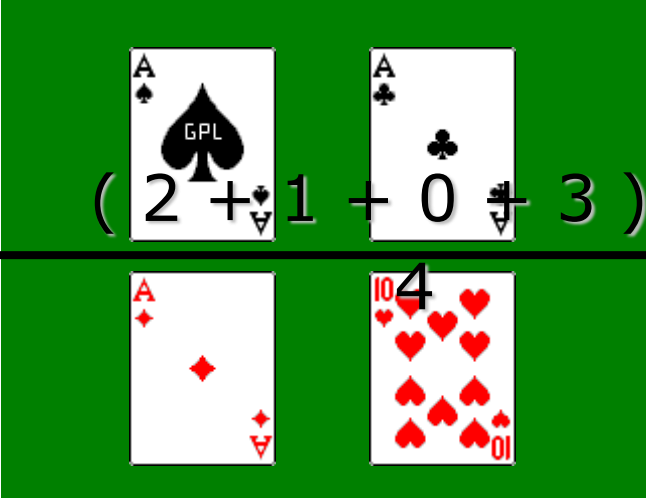
---



Most bits of an un-ACE instruction do not affect program output

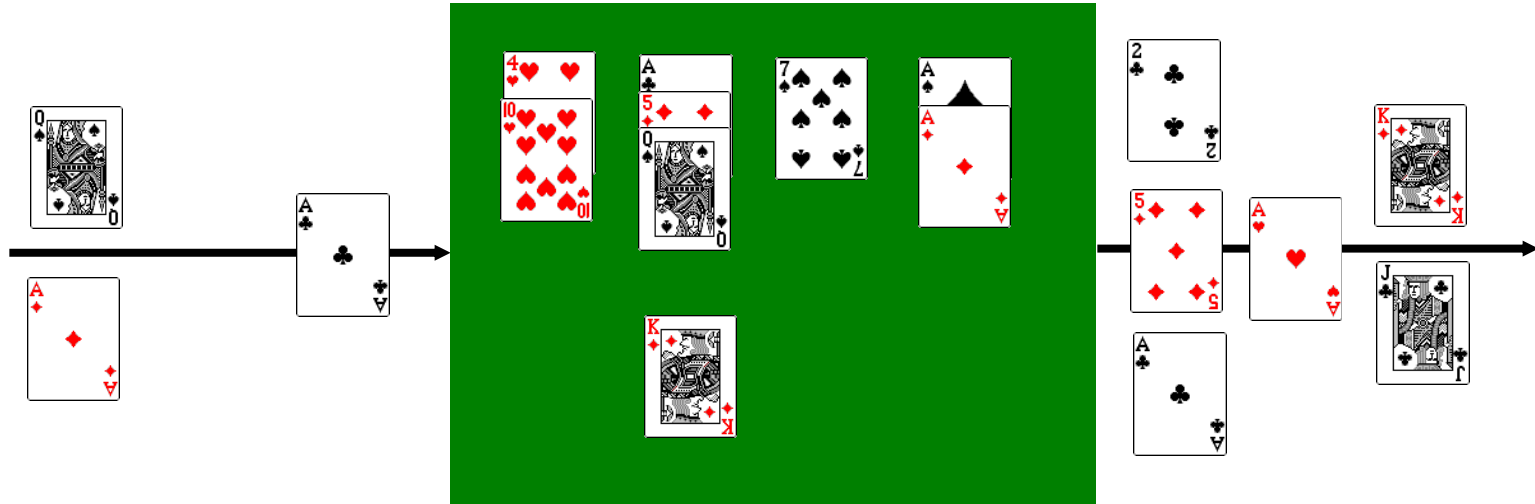
# Vulnerability of a structure

AVF = fraction of cycles a bit contains ACE state

$$\bar{x} = \frac{(2 + 1 + 0 + 3)}{4} = \text{ACE\%} = 0.75$$


$$= \frac{\text{Average number of ACE bits in a cycle}}{\text{Total number of bits in the structure}}$$

# Little's Law for ACEs



$$\overline{N}_{ace} = \overline{T}_{ace} \times \overline{L}_{ace}$$

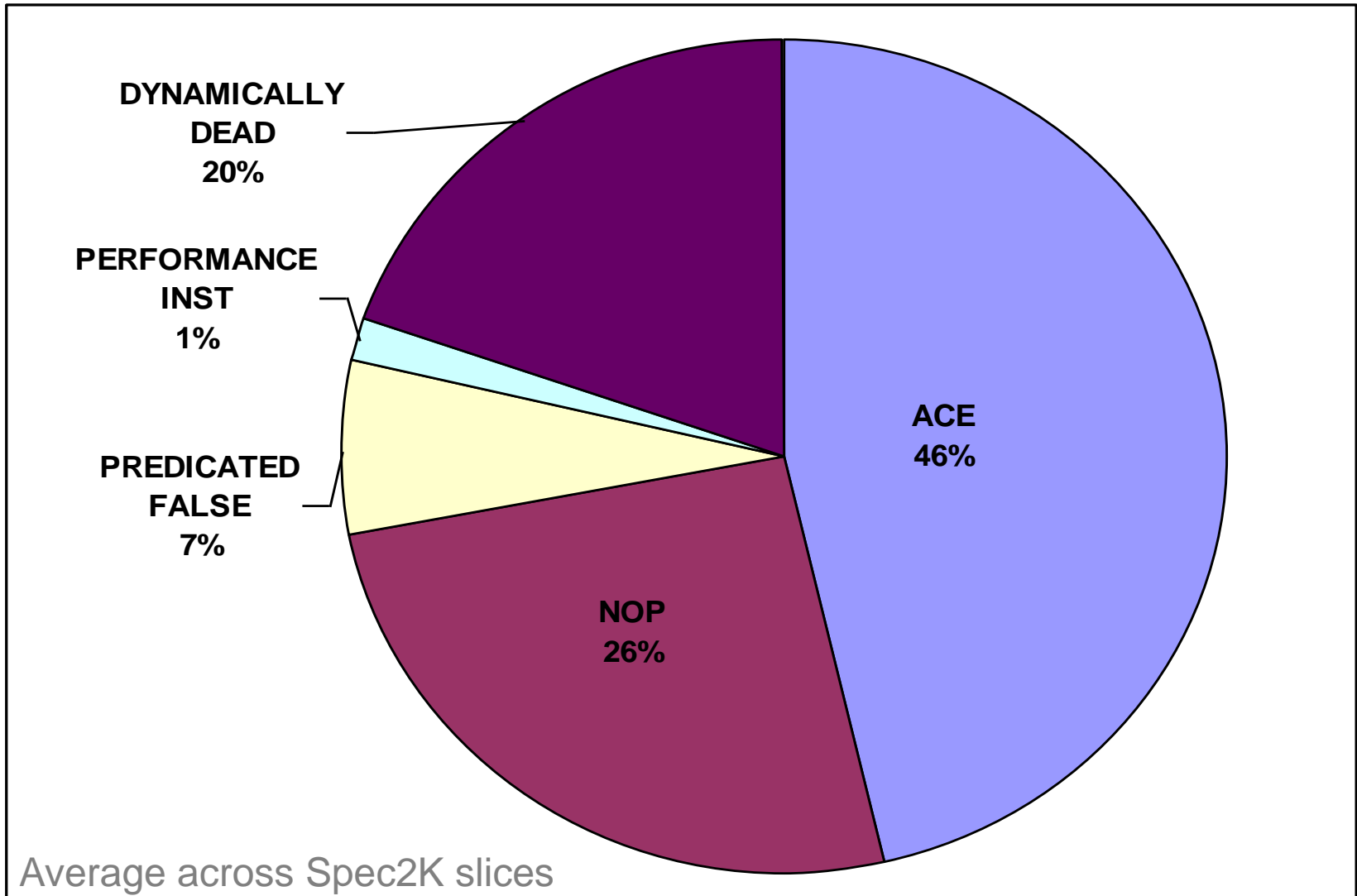
$$AVF = \frac{\overline{N}_{ace}}{N_{total}}$$

# Computing AVF

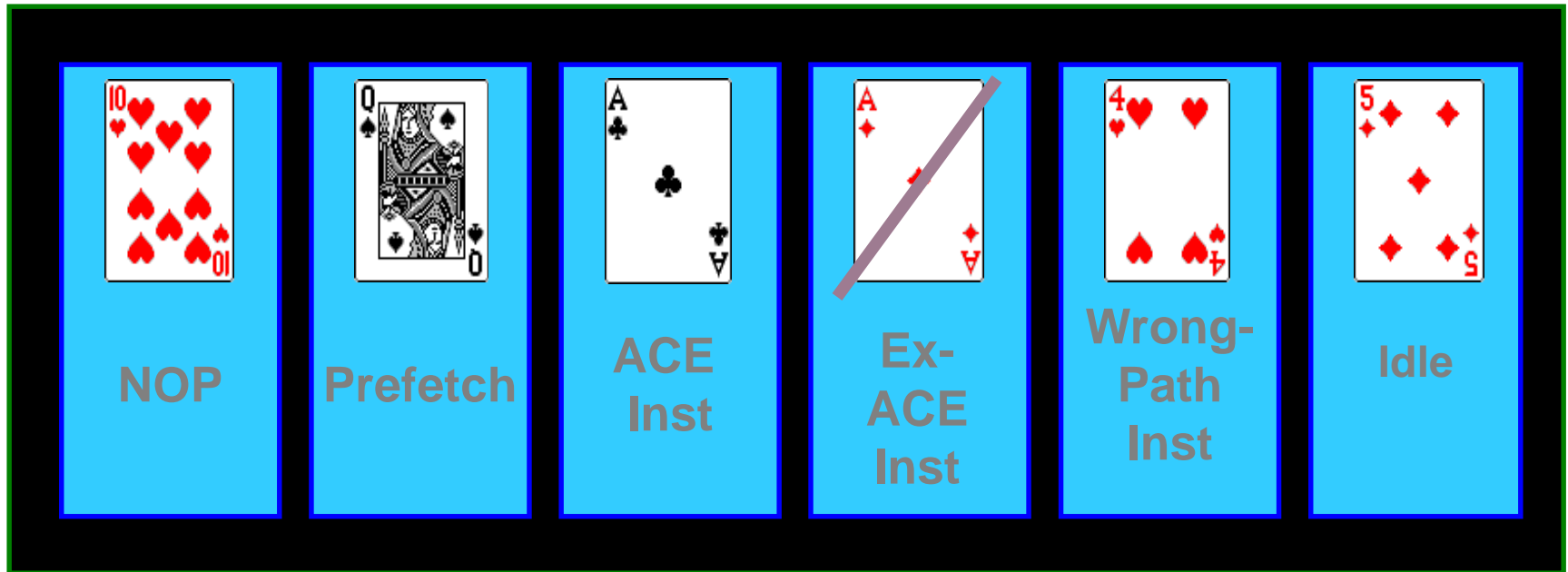
---

- Approach is conservative
  - Assume every bit is ACE unless proven otherwise
- Data Analysis using a Performance Model
  - Prove that data held in a structure is un-ACE
- Timing Analysis using a Performance Model
  - Tracks the time this data spent in the structure

# Dynamic Instruction Breakdown



# Mapping ACE & un-ACE Instructions to the Instruction Queue



**Architectural un-ACE**



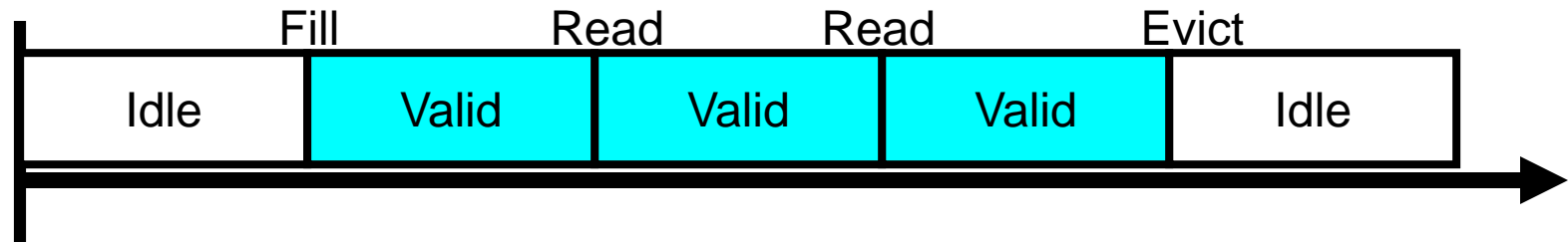
**Micro-architectural un-ACE**



# ACE Lifetime Analysis (1)

(e.g., write-through data cache)

- Idle is unACE

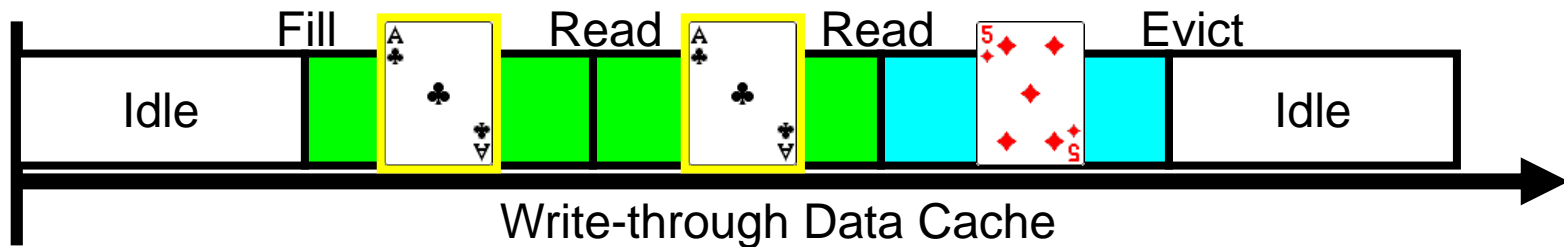


- Assuming all time intervals are equal
- For 3/5 of the lifetime the bit is valid
- Gives a measure of the structure's utilization
  - Number of useful bits
  - Amount of time useful bits are resident in structure
  - Valid for a particular trace

# ACE Lifetime Analysis (2)

(e.g., write-through data cache)

- Valid is not necessarily ACE

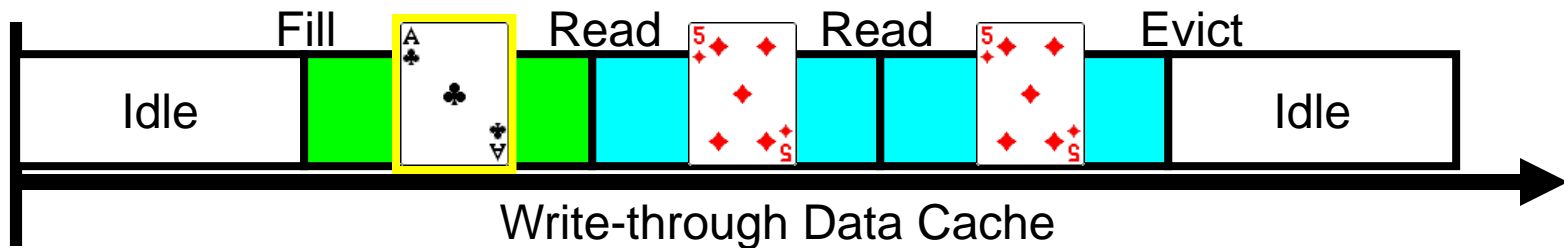


- ACE % = AVF =  $2/5 = 40\%$
- Example Lifetime Components
  - ACE: fill-to-read, read-to-read
  - unACE: idle, read-to-evict, write-to-evict

# ACE Lifetime Analysis (3)

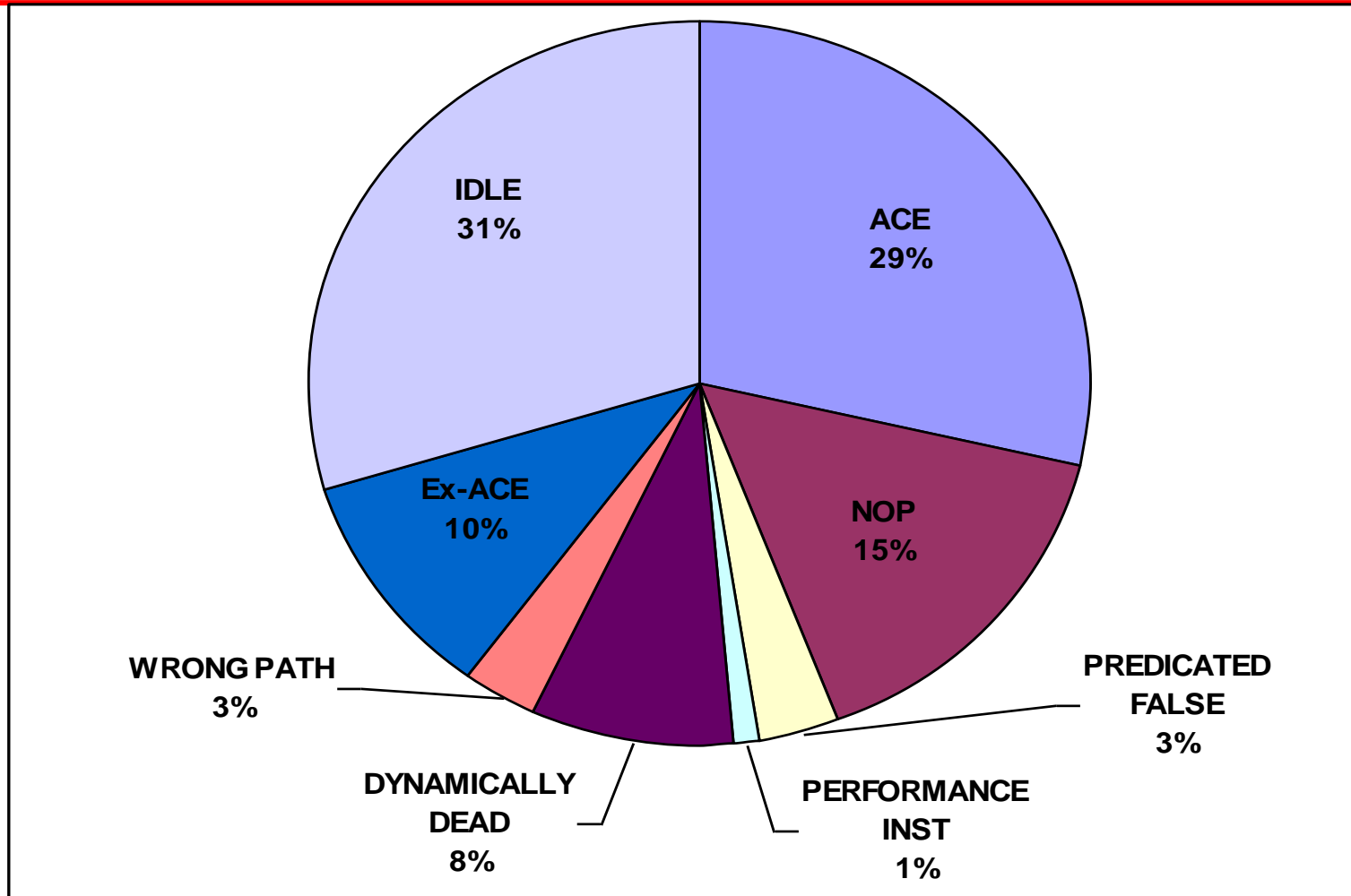
(e.g., write-through data cache)

- Data ACEness is a function of instruction ACEness



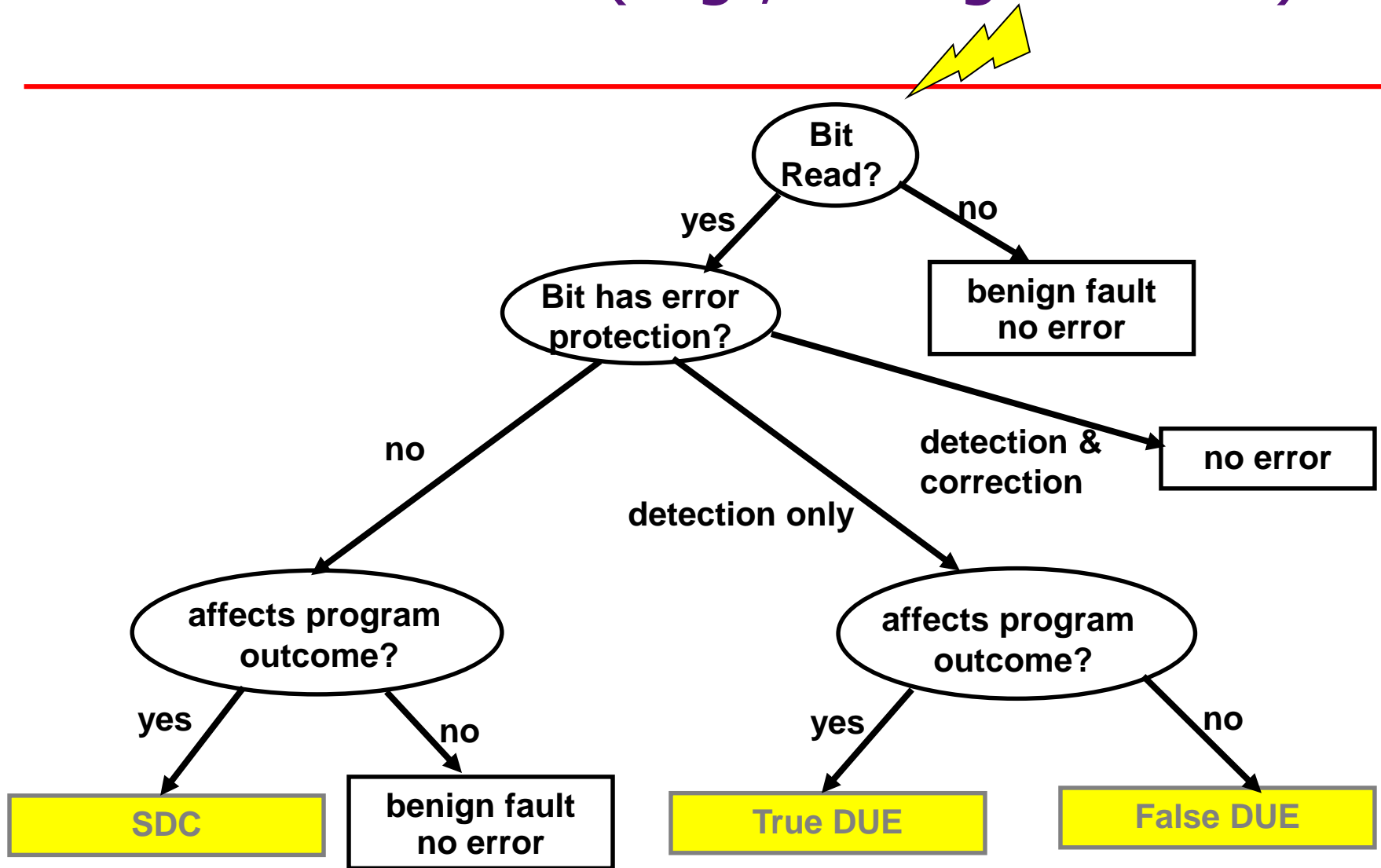
- Second Read is by an unACE instruction
- $AVF = 1/5 = 20\%$

# Instruction Queue



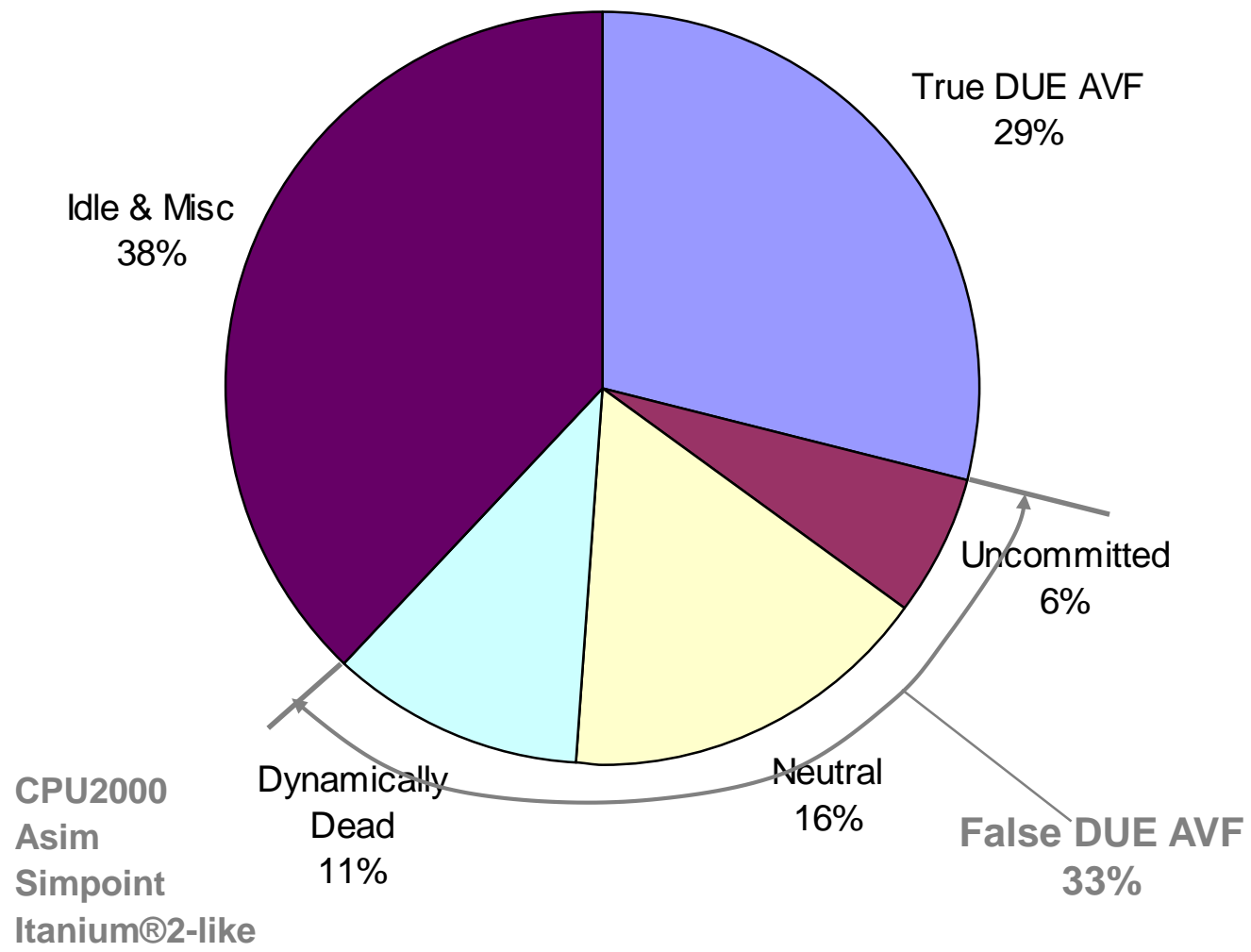
**ACE percentage = AVF = 29%**

# Strike on a bit (e.g., in register file)



**SDC = Silent Data Corruption, DUE = Detected Unrecoverable Error**

# DUE AVF of Instruction Queue with Parity

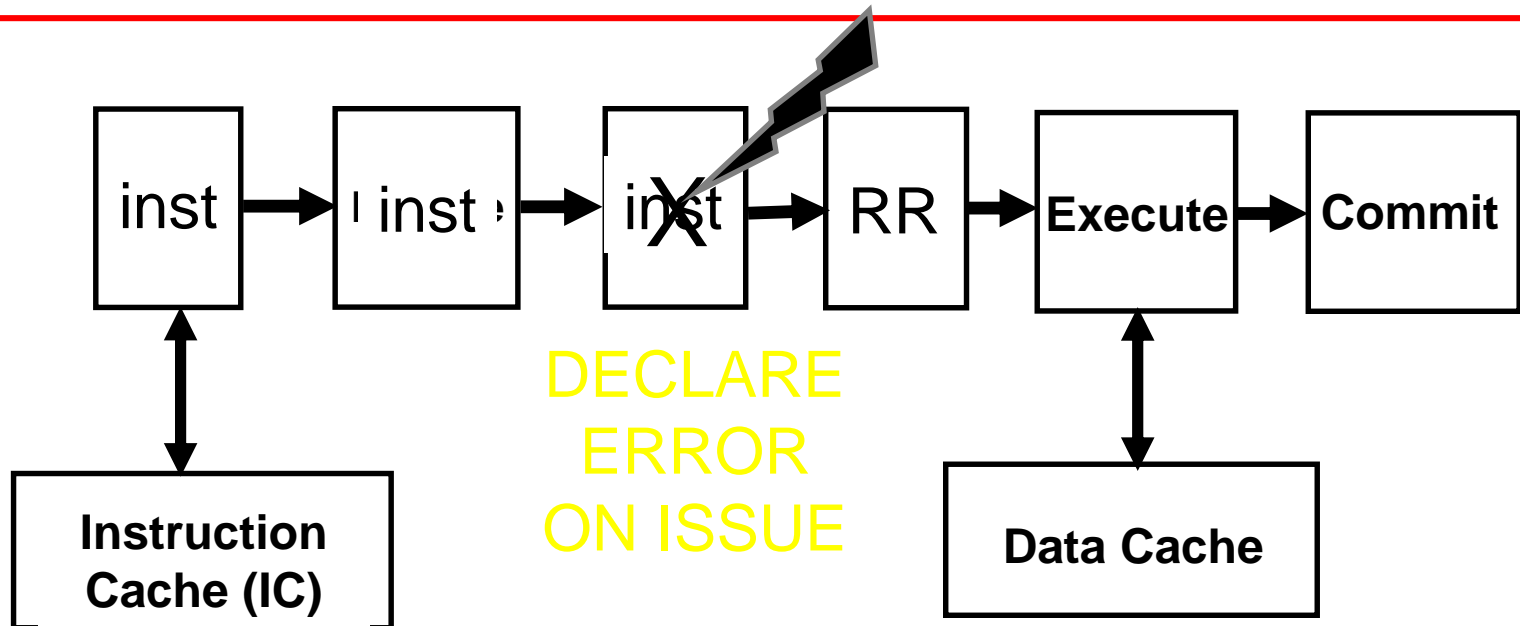


# Sources of False DUE in an Instruction Queue

---

- Instructions with uncommitted results
  - e.g., wrong-path, predicated-false
  - solution:  $\pi$  (possibly incorrect) bit till commit
- Instruction types neutral to errors
  - e.g., no-ops, prefetches, branch predict hints
  - solution: anti-  $\pi$  bit
- Dynamically dead instructions
  - instructions whose results will not be used in future
  - solution:  $\pi$  bit beyond commit

# Coping with Wrong-Path Instructions (assume parity-protected instruction queue)

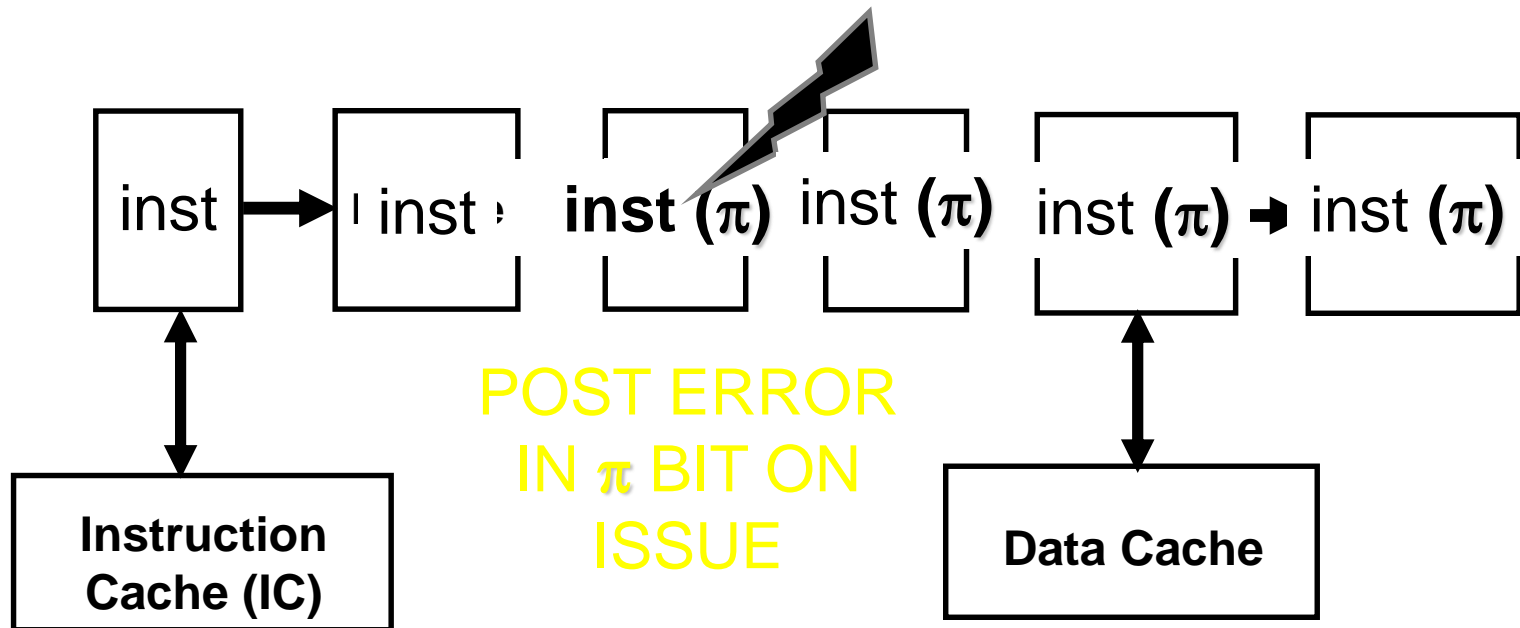


- Problem: not enough information at issue



# The $\pi$ (Possibly Incorrect) Bit

(assume parity-protected instruction queue)



At commit point, declare error only if not wrong-path instruction and  $\pi$  bit is set