

**ECE 254 / CPS 225**  
**Fault Tolerant and Testable Computing Systems**

**Faults and Their Causes**

Copyright 2008 Daniel J. Sorin  
Duke University

**Outline**

- **Intro and Terminology**
- **Causes of Faults**
- **Fault Models**
- **Research Papers**

**Why We Study Faults**

- **Know thy enemy!**
- **If we don't understand faults, it is much more difficult to design systems that can tolerate them**
  - We at least have to know how to model them

**Faults**

- **Fault: incorrect state of hardware or software resulting from physical defect, design flaw, or operator error**
- **Faults introduced during system design**
  - Pentium's incorrect floating point division design
  - Bug in software could cause infinite loop
- **Faults introduced during manufacturing**
  - Bad solder connection between chip pin and motherboard
  - Broken wire within chip
- **Faults that occur during operation**
  - Cosmic ray knocks charge off DRAM cell
  - System administrator incorrectly installs new software

## Errors

- **Error: manifestation of a fault**
  - Bit in main memory is a 0 instead of a 1 (due to cosmic ray)
  - Software pointer that mistakenly points to NULL (due to bug)
- **But not all faults lead to errors!**
  - Trees falling in empty forests don't make sounds
- **Examples of masked faults**
  - Cosmic ray knocks charge off logic signal, but after it had been correctly latched in and saved
  - Buggy software that isn't reached

## Failures

- **Failure: system level effect of an error (user-visible)**
  - System produces incorrect result of computation (e.g.,  $2+2=5$ )
  - System "hangs" (e.g., Blue Screen of Death)
- **Not all errors lead to failures!**
- **Examples of masked errors**
  - Bit flip in memory location that's not accessed again
  - NULL pointer that's not referenced again

## Fault → Error → Failure Examples

- Cosmic ray knocks charge off of DRAM cell
  - Error: bit flip in memory
  - Failure: computation produces incorrect result
- Software bug could allow for NULL pointer
  - Bug gets exercised and we get NULL pointer
  - Program seg faults when it tries to access pointer

## Duration of Faults/Errors

- **Transient (soft): occurs once and disappears**
  - E.g., Cosmic ray knocks charge off transistor → bit flip
  - Tend to be due to transient physical phenomena
  - Also known as **Single Event Upset (SEU)**
- **Intermittent: occurs occasionally**
  - E.g., Loose connection → occasionally open circuit
  - E.g., Bug in little-used software for rounding → incorrect data
- **Permanent (hard): occurs and doesn't go away**
  - E.g., Broken connection → always open circuit

## Masking

- **Logical**
  - E.g., if a fault flips a bit from 0 to 1 and it is then ANDed with a bit that is 0, then the fault cannot manifest itself as an error
- **Functional**
  - E.g., incorrect data is produced by an instruction that gets squashed due to a branch misprediction
  - E.g., the destination register of a NOP is corrupted by a fault

## Outline

- **Intro and Terminology**
- **Causes of Faults and Trends**
- **Fault Models**
- **Research Papers**

## Physical Defects: Transient Phenomena

- **Cosmic radiation (refer to Ziegler's paper)**
  - High energy particles that constantly bombard Earth
  - May have enough energy to disrupt charge on transistor ( $Q_{crit}$ )
  - Used to be only a problem for DRAM, but becoming a problem for SRAM and even for logic (as  $Q_{crit}$  decreases)
  - Trends:
    - »  $Q_{crit}$  decreasing
    - » Probability increasing that a cosmic ray that hits a transistor will disrupt its charge
    - » Transistor size decreasing → smaller probability that a cosmic ray will hit a particular transistor
    - » More transistors per system → greater probability of fault

## Physical Defects: Transient Phenomena

- **Alpha particle radiation**
  - Similar to cosmic rays, but radiation comes from metal decay
  - Often, the metal housing of the computer is the source
  - Lead solder joints also a problem → want to use "old lead"
  - Trends (same as for cosmic radiation):
    - »  $Q_{crit}$  decreasing
    - » Probability increasing that an alpha particle that hits a transistor will disrupt its charge
    - » Transistor size decreasing → smaller probability that an alpha particle will hit a particular transistor
    - » More transistors per system → greater probability of fault

## Physical Defects: Transient Phenomena

- **Electromagnetic Interference (EMI)**
  - Electromagnetic waves from other sources (e.g., microwave oven, power lines, etc.) can cause transient disruptions
  - EMI can be created by the circuit itself! Called “crosstalk”
  - EMI can induce electrical current on wires and thus change the signals on wires
- **There are other sources of transient faults, but they tend to be less significant**

(C) 2008 Daniel J. Sorin

ECE 254 / CPS 225

13

## Physical Defects: Manufacturing Defects

- **Manufacturing is not a perfect process, especially for microprocessors**
  - It’s not easy to manufacture something with dimensions on the order of 45nm
  - Many stages of chip processing which have to be done perfectly and avoid contamination
- **And testing doesn’t filter out all defective systems**
  - Often impossible to test for every possible defect in a reasonable amount of time
  - Also, testing won’t detect defects that don’t manifest immediately
- **Nanotechnology makes this problem even worse**

(C) 2008 Daniel J. Sorin

ECE 254 / CPS 225

14

## Physical Defects: Manufacturing Defects

- **Manufacturing flaws**
  - Bad solder connection between chip and board
  - VLSI defects (e.g., broken wire, bad via, etc. – see ECE 261)
  - Trends:
    - » Flaws may decrease as manufacturing process matures
    - » But flaws increase at start of each new process
    - » Tougher to avoid VLSI defects as dimensions shrink

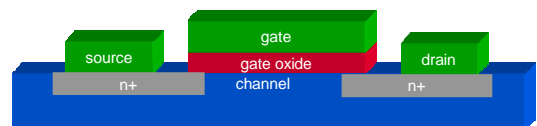
(C) 2008 Daniel J. Sorin

ECE 254 / CPS 225

15

## Physical Defects: Manufacturing Defects

- **VLSI fabrication process variability**
  - During fab, there’s some amount of variability in dimensions
    - » Thickness of gate oxide dielectric
    - » Length of channel
    - » Area of via
    - » Etc.



(C) 2008 Daniel J. Sorin

ECE 254 / CPS 225

16

### Physical Defects: Manufacturing Defects

- **Variability can lead to undesirable behavior**
  - Gate thickness falls below usable threshold → extra leakage current
  - Wire resistance is too high → signal too slow for clock
- **Trend: variability rising as VLSI dimensions shrink**
  - When dimensions are on the order of a handful of atoms, it doesn't take much variability to cause significant problems

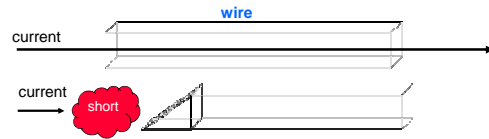
(C) 2008 Daniel J. Sorin

ECE 254 / CPS 225

17

### Physical Defects: Operational Defects

- **Permanent (hard) defects can occur during operation**
- **Electromigration**
  - If current density is too large for wire, wire metal will "migrate" away and potentially lead to a broken link
  - Exacerbated by thermal cycling due to hotter chips
  - Trend: getting worse as wires become smaller and chips become hotter



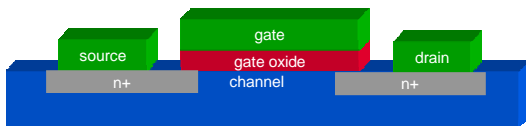
(C) 2008 Daniel J. Sorin

ECE 254 / CPS 225

18

### Physical Defects: Operational Defects

- **Gate oxide breakdown**
  - MOSFET transistor has a gate oxide that insulates the gate from the channel
  - If this oxide breaks down, will get a short between gate and channel
  - Trend: getting worse as gate oxides become thinner (only a handful of atoms thick!)



(C) 2008 Daniel J. Sorin

ECE 254 / CPS 225

19

### Hardware Design Flaws: Logical Bugs

- **Famous example: Intel Pentium floating point divide didn't work in every single case due to bug in design → very costly recall**
- **Sun UltraSPARC III had design flaw in a special cache that meant that it couldn't be used → loss in performance**
- **AMD's quad-core Barcelona chip had design bug in TLB hardware → long, expensive delay in shipping chips**

(C) 2008 Daniel J. Sorin

ECE 254 / CPS 225

20

## Hardware Design Flaws: Timing Bugs

- Logic is fine, but the timing analysis is flawed
- Example: clocking a processor at 4 GHz when there's a slow path in the pipeline that can only run at 3.8 GHz
- Timing analysis must consider critical path delay and environmental effects (operating temperature, EMI, cross-talk, etc.) to determine the maximum operating speed
- This problem is exacerbated by process variability

## Design Flaws: Software

- We all know that software has bugs
- Types of bugs
  - Incorrect algorithm
  - Memory leak (C, C++, but not Java)
    - » Allocating memory, but not deallocating it
  - Reference to NULL pointer (C, C++, but not Java)
    - » This usually leads to a seg fault and core dump
  - Incorrect synchronization in multithreaded code
    - » Allowing more than 1 thread in critical section at a time
    - » If you've taken CPS 110 (or any OS class), you've seen this!

## Operator Error

- It has been argued that operator error is the leading cause of computer system failures
  - We'll read a paper that discusses this
- Examples
  - `rm -R *` (in the wrong directory)
  - Incorrect installation of software
  - Frying a board when installing new memory chips
  - Dropping the laptop (and/or kicking it)
  - Etc.

## Outline

- Intro and Terminology
- Causes of Faults and Trends
- **Fault Models**
- Research Papers

## Purpose of Fault Modeling

- **Model = abstraction of physical phenomenon**
- **Simple, tractable way to analyze effects of faults**
- **Example: “fail-stop” network switch → if a fault occurs, the switch will just stop doing anything**
  - Model reflects the behavior of many potential underlying faults
  - E.g., switch has short from power to ground, switch is on fire, etc.
  - Easier to work with this model than to consider all possible faults

## Limitations of Fault Modeling

- **Garbage In → Garbage Out (GIGO)**
- **If model doesn't reflect behavior of a type of fault, an analysis based on that model won't handle that type of fault**
- **“Deflated car tire” fault model doesn't handle case where brakes stop working**
  - Having a spare tire to handle this fault isn't helpful
- **More computer-y example: Fail-stop fault model for network switch doesn't handle case where switch starts routing packets incorrectly**
  - And this fault model represents several realistic underlying faults

## Using Fault Models

- **Testing**
  - Instead of testing for every possible fault, test for faults that are in the assumed fault model
  - We'll cover testing in more depth towards end of semester
- **System design**
  - Instead of designing system to tolerate every possible fault, design system to tolerate faults that are in the assumed fault model

## Low-Level Hardware Fault Modeling

- **Stuck-at-x fault model ( $x=0$  or  $x=1$ )**
  - Useful for modeling faults in state or on wires
  - Many real faults can be modeled this way (but not all!)
  - Most prevalent model, particularly for testing purposes
  - Easy to use and analyze
- **Stuck-open and/or stuck-closed**
  - Models faults in switches (i.e., transistors)

## Low-Level Hardware Fault Modeling

- **Coupling (bridging) faults**
  - Signal on one wire is equal/inverse (coupled) to signal on other wire
  - Models effects of short circuits or cross-talk
- **Fail-stop faults**
  - Assumes that faulty component stops doing anything
  - Often used for modeling entire processor or network switch

(C) 2008 Daniel J. Sorin

ECE 254 / CPS 225

29

## Transition and Delay Faults

- **Most faults consider the signal value on a wire or transistor (e.g., stuck-at-1)**
- **Transition faults** model the situations in which the correct value is produced on a given wire or transistor ... but not at the right time
- **Delay faults** model the situations in which paths (not just given nodes) exhibit incorrect timing behavior
- **These fault models model many known physical phenomena, but their effects are generally much less tractable to analyze**

(C) 2008 Daniel J. Sorin

ECE 254 / CPS 225

30

## Other Fault Models

- **Can we develop fault models that aren't so low-level?**
  - Do we need to consider each transistor?
  - How about logic gate level?
  - Or even coarser granularity (e.g., ALU, cache line, processor core)
- **What about non-hardware fault models?**
  - What is an appropriate fault model for software bugs?
  - How about for security breaches?

(C) 2008 Daniel J. Sorin

ECE 254 / CPS 225

31

## How Many Faults at Once?

- **Many fault models include the assumption that only one fault can occur at a given instance**
  - Helps to make analysis more tractable
  - E.g., "single stuck-at fault" model
- **Reasonable assumption if:**
  - Faults are rare
  - System doesn't require extreme reliability
  - **Faults are detected and, if necessary, removed quickly**
- **The problem with latent faults**
  - Fault occurs, but isn't detected
  - Later, a "single" fault occurs, but this is now a double fault scenario
  - If you only plan for single faults, then this situation is a problem

(C) 2008 Daniel J. Sorin

ECE 254 / CPS 225

32

## The Art of Choosing a Fault Model

- **What is an appropriate hardware fault model for:**
  - Mainframe system
  - Desktop processor
  - iPhone
  - Embedded controller for anti-lock brakes
  - Embedded controller for toaster oven
- **Remember: only include those faults that matter for your situation**
  - Do you care about transient errors in an iPhone?
  - Do you care about transistor wearout in a laptop?

## Outline

- **Intro and Terminology**
- **Causes of Faults and Trends**
- **Fault Models**
- **Research Papers**

## Paper #1

- **“IBM Experiments in Soft Fails in Computer Electronics” (Ziegler)**
- **Famous paper on how IBM has studied soft errors**

## Paper #2

- **“Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation” (Borkar)**
- **Highly-cited 2005 paper from Intel that discusses the challenges of designing reliable processors despite the high probabilities of imperfect fabrication and device wearout**

### Paper #3

- “A Large-Scale Study of Failures in High-Performance Computing Systems” (Schroeder and Gibson)
- Recent paper that characterizes failures in large computing clusters at Los Alamos

### Paper #4

- “Why Do Internet Services Fail, and What Can be Done About It?” (Oppenheimer et al.)
- Paper from David Patterson’s Recovery Oriented Computing (ROC) group at UC-Berkeley

### Outline

- Intro and Terminology
- Causes of Faults
- Fault Models
- Research Papers