

## EDA122/DIT061 Fault-Tolerant Computer Systems

## Welcome to Lecture 14

Error detection techniques

## Outline

- Hardware reliability trends
- Case study: Experimental evaluation of error handling mechanisms in a jet-engine controller
- Layered fault tolerance (from lecture 13)
- Error detection techniques
- Fault detection vs. error detection

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

2

## Transistor reliability trends

Shekhar Borkar, Intel Corp:

"As technology scales, variability in transistor performance will continue to increase, making transistors less and less reliable. ....

Finding solutions to these challenges will require a **concerted effort on the part of all the players in a system design.**"



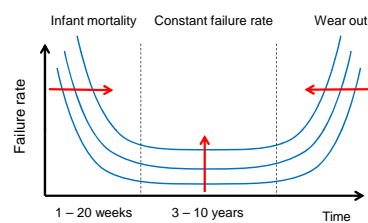
Borkar, S.; "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," IEEE Micro, December 2005.

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

3

## Trends in the bathtub curve



- Infant mortality: **Increasing manufacturing defects**
- Constant failure rate: **Increasing rate of transient, intermittent and permanent faults**
- Wearout: **Acceleration of aging phenomena**

Source: Vikas Chandra, ARM R&D, Dependable Design in Nanoscale CMOS Technologies: Challenges and Solutions  
Keynote address, WDSN, Estoril, Portugal, June 29, 2009

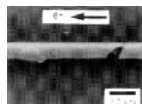
Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

4

## Main sources of transistor faults

- Process variations
  - Random variations related to lithography, etching, dopant count
  - Voltage and temperature variations
- Wear out effects (degradation)
  - NBTI - negative bias temperature instability
  - HCI - hot carrier injection
  - Gate oxide breakdown
  - Electromigration
  - ...
- Soft errors
  - Bit-flips in latches, flip-flops and memory cells
  - Mainly caused by cosmic-induced high energy neutrons (cosmic neutrons)
  - Soft errors  $\Rightarrow$  no permanent damage to hardware

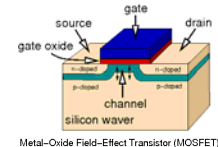
EDA122/DIT061  
Fault-Tolerant  
Computer  
Systems

Lecture 14

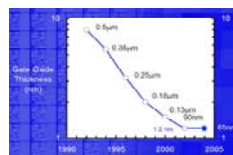
5

## Gate oxide breakdowns

- Gate oxide breakdowns increase leakage currents and change electrical characteristics of transistors



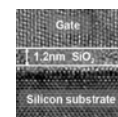
Metal-Oxide Field-Effect Transistor (MOSFET)

Gate oxide scaling  
Source: Intel 2005

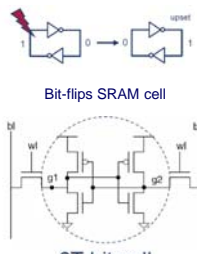
Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

6

Gate oxide in 90 nm technology  
Thickness: 5 atom layers

### Single event upset (SEU)



- SEU:s are particle induced upsets (bit-flips)
- Caused by highly energetic particles such as neutron, protons and muons

Bit-flips SRAM cell

6T bit cell

Particle trajectory

Drain

Poly Si gate

SiO<sub>2</sub> gate

Source

Si substrate

Depletion region

Particle strike in n-channel MOSFET transistor

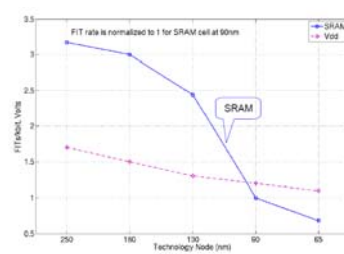
Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

7

### Soft error rate trend for SRAM

(Radiation test data from Sun Microsystems)



1 FIT =  $10^{-9}$  faults per hour

Source: A. Dixit, R. Heald, and A. Wood, "Trends from Ten Years of Soft Error Experimental, SELSE '09, Stanford, CA, USA.

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

8

### Raw soft error rate trend for microprocessors

(Data from Sun Microsystems)

Technology node (nm)	Year introduced	Relative SEU rate in FITs/kbit	Mbits/processor	Relative uncorrected SEU rate / microprocessor
250	1998	3.2	1.52	5.0
180	1999	3.0 ↓	1.52	4.3 ↓
130	2000	2.4 ↓	3.28	7.9 ↑
90	2002	1.0 ↓	33.6	33.6 ↑
65	2006	0.7 ↓	44.3	30.5 ↑
40	2008	0.94 ↑	71	67 ↑

1 FIT =  $10^{-9}$  faults per hour

Source: A. Dixit, R. Heald, and A. Wood, "The Impact of New Technology on Soft Error Rates, SELSE-6, Stanford, CA, USA, 2010

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

9

### Outline

- Hardware reliability trends
- Case study: Experimental evaluation of error handling mechanisms in a jet-engine controller (see separate presentation)
- Layered fault tolerance (from lecture 13)
- Fault detection vs. error detection
- Error detection techniques

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

10

### Outline

- Hardware reliability trends
- Case study: Experimental evaluation of error handling mechanisms in a jet-engine controller
- Layered fault tolerance (from lecture 13)
- Error detection techniques
- Fault detection vs. error detection

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

11

### Layered fault tolerance

Fault tolerance can be implemented at three abstraction layers:

- Hardware layer** – mechanisms implemented in hardware either *within one integrated circuit*, or by *replication of integrated circuits* within a node.
- Software layer** – mechanisms implemented in software dealing with errors occurring within a node
- System layer** – mechanisms implemented in software dealing with errors occurring in other nodes in the system, or in the system's communication network

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

12

## Purpose of different layers

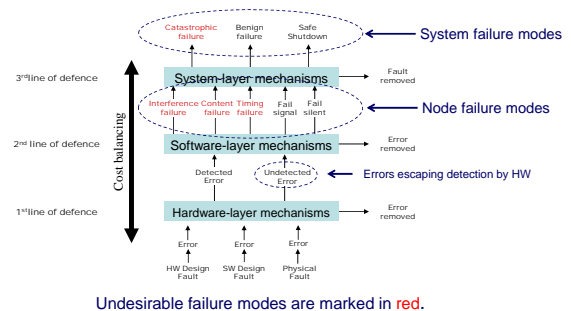
- **Hardware layer** – serves as a *first line of defense* that should
  - Correct as many errors as is economically feasible
  - Detect errors that cannot be corrected
- **Software layer** – serves as a *second line of defense* that should
  - Correct errors detected, but **not corrected by the hardware layer**.
  - Detect errors that are **not detected or corrected by the hardware layer**
  - Ensure **appropriate failure semantics** for the node for any error that cannot be corrected.
- **System layer** – serves as a *third line of defense* that should
  - Detect and correct any errors that are **not detected or corrected by the software and hardware layers**

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

13

## Layered fault tolerance



Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

14

## Outline

- Hardware reliability trends
- Case study: Experimental evaluation of error handling mechanisms in a jet-engine controller
- Layered fault tolerance (from lecture 13)
- **Error detection techniques**
- Fault detection vs. error detection

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

15

## Error detection techniques (slide from Lecture 2)

Two examples:

- **Duplication and comparison**
  - Two modules produce replicated results
  - Errors are detected by comparing the results
  - Ensures fail-silence
- **End-to-end checksums**
  - The service provider adds a checksum to its outputs
  - Checksums are checked by the service user
  - Provides detectability of value failures
  - Protects the content of a service while it is being transferred from the service provider (the producer) to the service users (the consumers).

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

16

## On-line error detection techniques mentioned in the course book (1) (Hardware layer techniques)

- Watchdog timers (p. 130 in course book)
  - Hardware layer technique supported by software
- Bus monitoring (p. 130 in course book)
  - Checking the range of addresses generated by a CPU
  - Examples
    - Checking that the CPU use an even address when reading a 32 or 64-bit word.
    - Checking CPU (or program) memory accesses using a memory management unit (MMU).
- Power supply monitoring (pp. 130-131 in the course book)

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

17

## On-line error detection techniques mentioned in the course book (2) (Generic principles)

- Duplication and comparison (pp. 128 and 138-141 in the course book)
  - Comparison of redundant signals
  - Self-checking pair
- Consistency checking
  - Uses a priori knowledge about information or system behaviour.
  - Examples:
    - Hardware exceptions in CPUs checking for illegal operations, e.g., division by zero.
    - Range checking in software of constrained program variables.
- Information redundancy
  - Use of error detecting and error correcting codes

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

18

## Generic principles for error detection

- *Duplication and comparison, consistency checking and information redundancy* are examples of **generic principles** for error detection<sup>1</sup>.
- These principles can be used at all abstraction layers, i.e., the **hardware, software and system layers**.

<sup>1</sup>Note: Information redundancy can also be used to *correct* errors.

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

19

## Combination of error detection techniques

- Error detection techniques at different abstraction layers are often combined to achieve **high error detection coverage**.

Example:

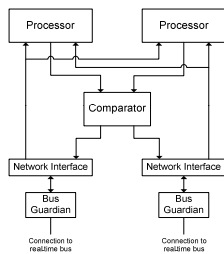
- The next slide (from lecture 2) describes how **duplication and comparison in hardware** and **end-to-end checksums** are combined to ensure fail silence for a node in a distributed system.
- End-to-end checksums is an example of a system layer technique that uses **information redundancy** (a checksum).

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

20

## HW architecture for a fail-silent node in a distributed system (slide from Lecture 2)



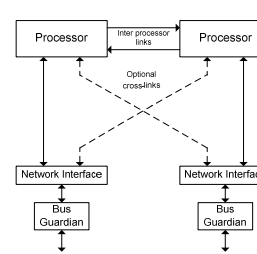
- Processor failures are detected by **duplication and comparison**
- The processors produce replicated messages that are compared by the comparator.
- The network interfaces receive messages from the comparator and send them to other nodes via two redundant real-time busses.
- The message content (payload) is protected by **end-to-end checksums** calculated by the content producer.
- The **end-to-end checksums** ensures that faults in the comparator and network interfaces are detectable by the service users (other nodes).

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

21

## Self-checking node supporting software implemented message comparison



- The processors execute the same programs and exchange copies of outgoing messages via the inter-processor links
- They compare the message copies and stop execution if the copies do not match.
- An error counter stores the number of mismatches that has occurred.
- The node is restarted after a mismatch only if the value of the error counter is below a predefined threshold
- The bus guardian protects the bus from erratic behavior (e.g., babbling idiot) of the network interfaces

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

22

## Watchdog timers

- Watchdog timers are used to detect slow programs and programs that hang in infinite loops
- The principle is simple:
  - When a program starts to execute, either the program itself or the operating system starts a hardware timer.
  - The timer must be reset by the program within a given deadline, otherwise the timer will send an interrupt signal to the CPU.
  - The interrupt signal causes the CPU to take appropriate recovery actions, such as restarting the program or rebooting the node.
- Watchdog timers are common in embedded real-time systems.
- They are used to "transform" *timing failures* into *signalled failures* or *silent failures*.

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

23

## Initiating recovery via a watchdog timer

- Watchdog timers are sometimes used in conjunction with other error detection mechanisms to simplify the implementation of recovery.
- This works as follows:
  - When an error is detected, the error detection mechanism stores an error flag in a designated memory area (preferably a "crash-proof" memory)
  - The error detection mechanism then forces a program hang, which subsequently causes the watchdog timer to raise an interrupt.
  - The interrupt invokes a recovery routine which reads the error flag and then initiates the appropriate recovery actions.
  - Restart and recovery could be done for an entire node, a single program, or a group of programs.

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

24

## Restarting a node in a distributed system

- Restarting a node in a distributed system involves an elaborate set of actions, including
  - recovering the node's view of the system state
  - reintegrating the node into the set of operational nodes
- These actions are handled by a system-layer mechanism called a **node membership service**.

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

25

## CPU Exceptions (Hardware layer technique)

- Modern central processing units (CPUs) are equipped with hardware implemented error detection mechanisms called *hardware exceptions*
- The number and type of hardware exceptions varies depending on the CPU design
- When a hardware exception is raised, the CPU stops the program execution and jumps to an exception routine
- The handling of exceptions is very similar to how a CPU responds to interrupt signals
- Some examples of common hardware exceptions is given in the next two slides

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

26

## Examples of CPU exceptions (1)

**Bus error:** detects errors during read and write accesses to the main memory. This exception is raised (triggered) when the CPU attempts to access an address to which no memory or any I/O device is connected.

**Address error:** detects when the CPU makes an attempt to access memory using an odd numbered address; only even numbered addresses are allowed in many CPUs.

**Illegal opcode:** detects if the CPU during an instruction fetch reads a value from memory (or the instruction cache) that doesn't correspond to a valid instruction. This error can occur if the program counter is erroneously loaded with an address pointing to a data area rather than a program code area.

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

27

## Examples of CPU exceptions (2)

**Privilege violation:** detects if a user program attempts to execute an instruction which is allowed only for programs that execute in the superuser mode (privileged mode), such as the operating system or device drivers. User programs normally execute in user mode (normal mode).

**Division by zero:** detects if a program tries to divide a number with zero.

**Spurious interrupt:** detects if an interrupt is signalled but no interrupt vector is provided by the interrupting device. (The interrupt vector tells the CPU which device it was that raised the interrupt signal and thereby indicates which interrupt service routine that the CPU shall execute.)

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

28

## Operating System and Compiler Generated Software Assertions (Software layer techniques)

- Operating system assertions:
  - Examples:
    - Integrity checks of data structures used by the operating system
    - Execution time monitoring of application and system processes
- Compiler generated run-time assertions:
  - Examples:
    - Value range overflow checking
    - Loop iteration bound overflow checking
    - Type checking of constrained variables
- When an error is detected by any of these mechanisms, typically a "trap" or software interrupt instruction is executed, which initiates appropriate recovery actions.

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

29

## Overview of error detection mechanisms at different layers



### Hardware layer

- CPU hardware exceptions:** Bus error, Address error, Illegal opcode, Privilege violation, Division by zero, Spurious interrupt, etc.
- Error detecting and correcting codes** in main memory, caches, internal buffers
- Special hardware circuits** (often connected to the non-maskable interrupt signal of a CPU): Power supply monitor, Network (bus) guardian.
- Watchdog timer** (usually implemented as combination of HW and SW)



### Software layer

- Compiler:** Type checking of constrained variables, Value range overflow, Loop iteration bound overflow
- OS:** Processing time overflow, consistency checks on OS data,
- Application:** time-redundant execution of tasks, application specific consistency checks



### System layer

- End-to-end checksums**
- Comparison of results produced by two nodes**
- Voting on results produced by three or more nodes**

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

30

## Outline

- Hardware reliability trends
- Case study: Experimental evaluation of error handling mechanisms in a jet-engine controller
- Layered fault tolerance (from lecture 13)
- Error detection techniques
- **Fault detection vs. error detection**

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

21

## Fault Detection vs. Error Detection

- Terminology
  - Both the terms *fault detection* and *error detection* are used in the literature, see discussion in the beginning of Chapter 6.4 in the course book
- We distinguish between
  - Concurrent (on-line) error detection
    - Detection of errors during operation
    - Purpose is to mask or minimize adverse effects of errors
  - Non-concurrent (off-line) fault detection
    - Testing to find physical hardware faults while the system is off-line
    - Purpose is to identify faulty hardware units

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

22

## Off-line fault detection techniques

- Functionality checking (p. 127 in the course book)
  - Examples:
    - Test of random access memory (RAM) by writing and reading back test patterns to all memory words
    - Test of CPU by running special test programs
- Loop back testing (p. 129 in the course book)
  - Example:
    - "echo" testing of communication paths

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

23

## Overview of Lecture 15

- Time-Triggered Systems
- Read *before the lecture*:
  - Lecture slides
  - The Time-Triggered Architecture (see reading instructions)

Lecture 14

EDA122/DIT061 Fault-Tolerant Computer Systems

24