Lecture Outline

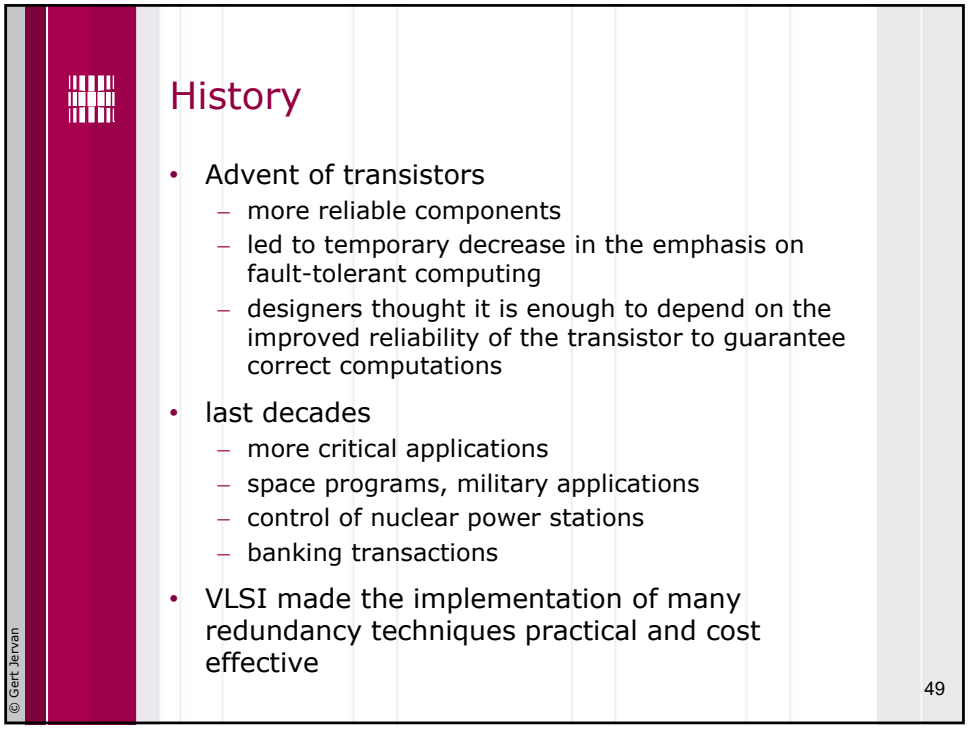
- ✓ **Historical perspective and famous incidents/accidents**
- **Basic terminology**

47

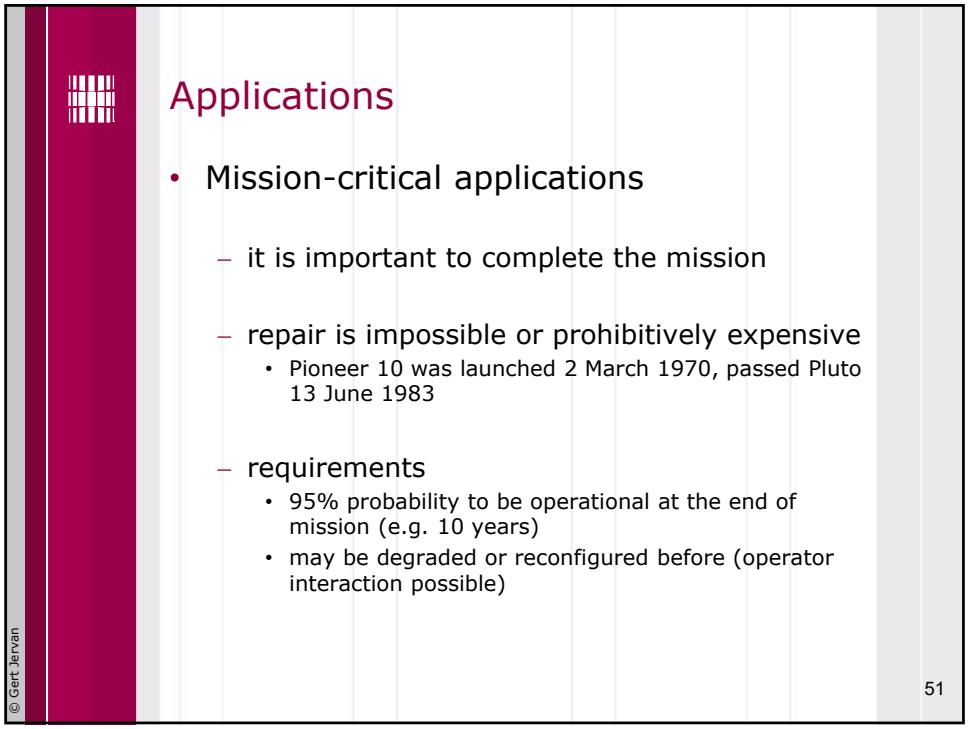
History

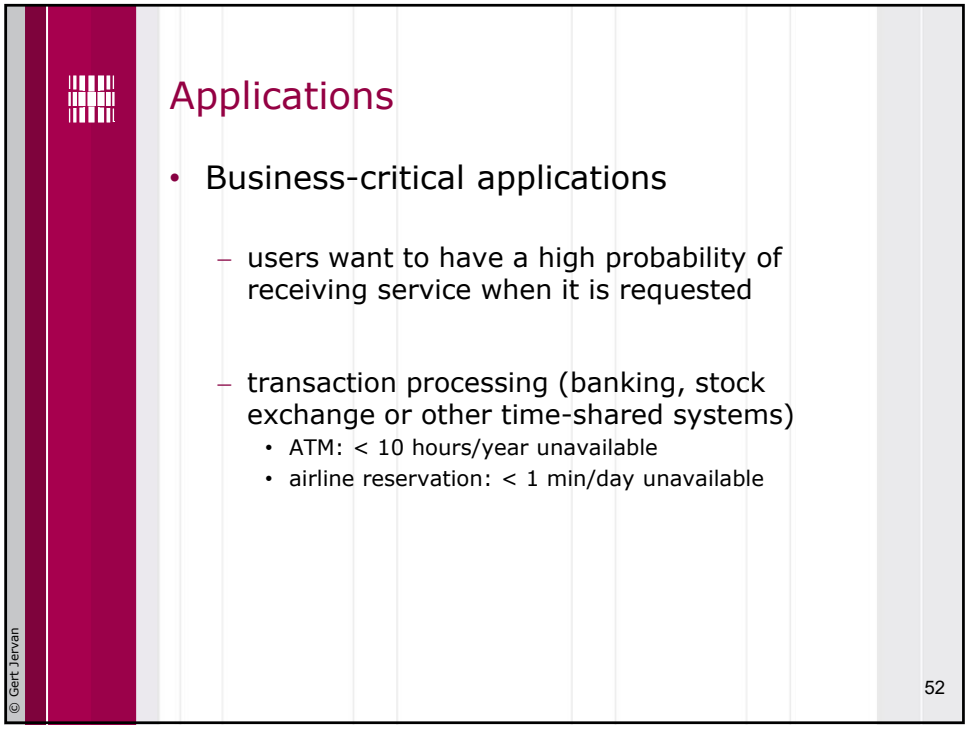
- Early computer systems
  - Basic components had very low reliability (de-bug)
  - Fault tolerant techniques were needed to overcome it by
    - Adding redundant structures with voting
    - Error-detection and error correction Codes
  - EDVAC (1949)
    - Duplicate ALU and compare results of both
    - Continue processing if agreed, else report error
  - Bell Relay Computer (1950)
    - 2 CPUs
    - One unit begin executing the next instruction if the ohter encounts an error
  - IBM 650, UNIVAC (1955)
    - Parity check on data transfers

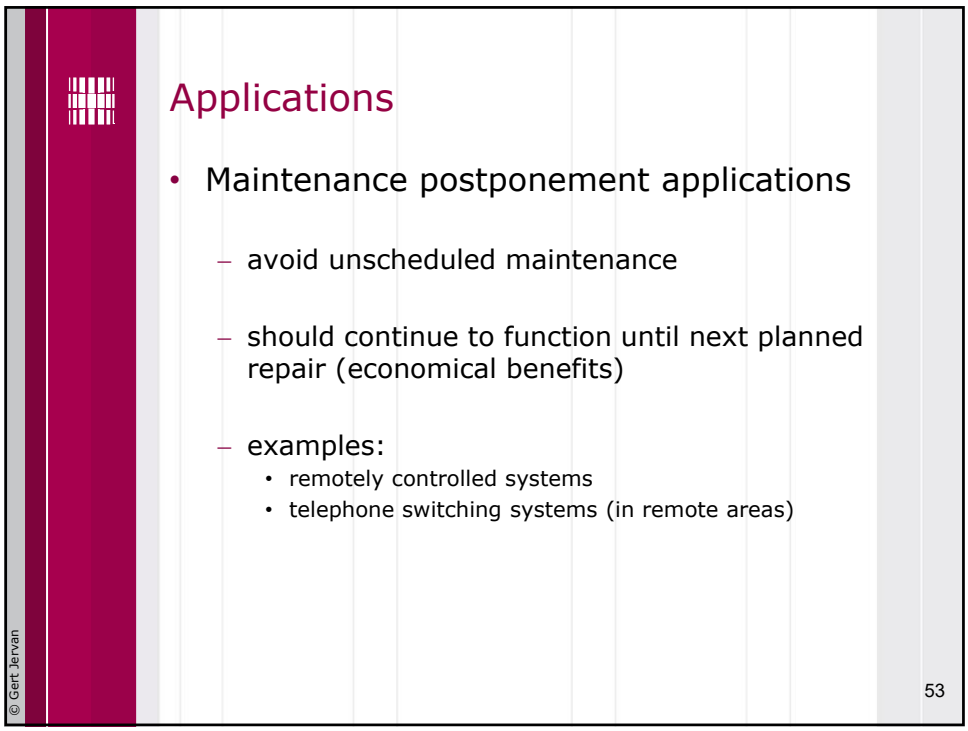
48

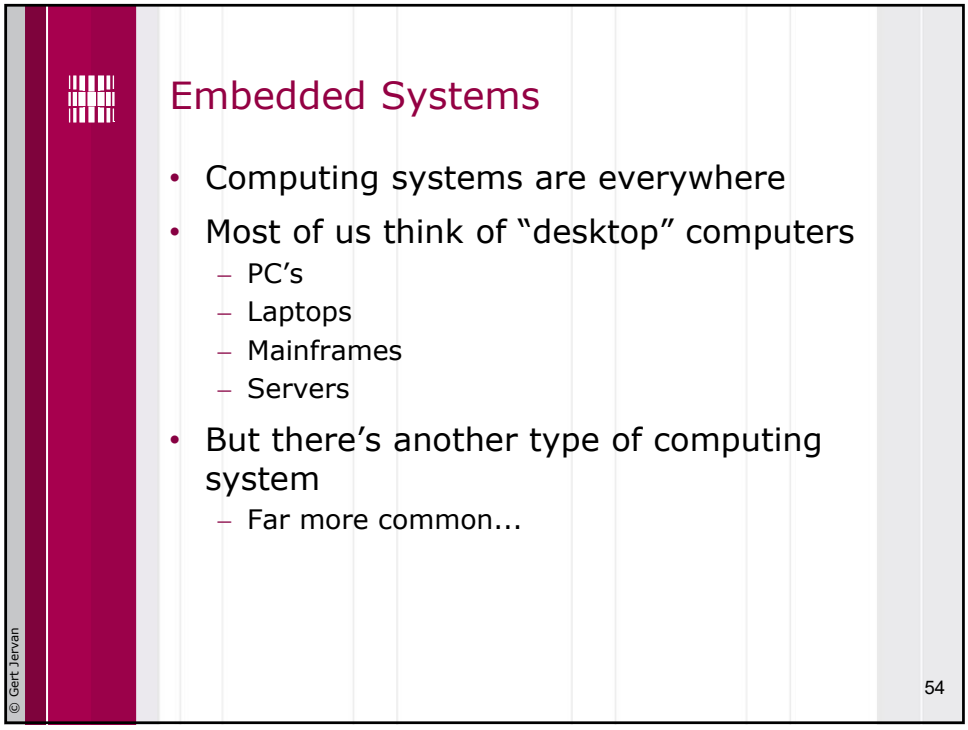

History
• Advent of transistors
– more reliable components
– led to temporary decrease in the emphasis on fault-tolerant computing
– designers thought it is enough to depend on the improved reliability of the transistor to guarantee correct computations
• last decades
– more critical applications
– space programs, military applications
– control of nuclear power stations
– banking transactions
• VLSI made the implementation of many redundancy techniques practical and cost effective
© Gert Jervan
49


Applications
• Safety-critical applications
– critical to human safety
• aircraft flight control
– environmental disaster must be avoided
• chemical plants, nuclear plants
– requirements
• 99.99999% probability to be operational at the end of a 3-hour period
© Gert Jervan
50

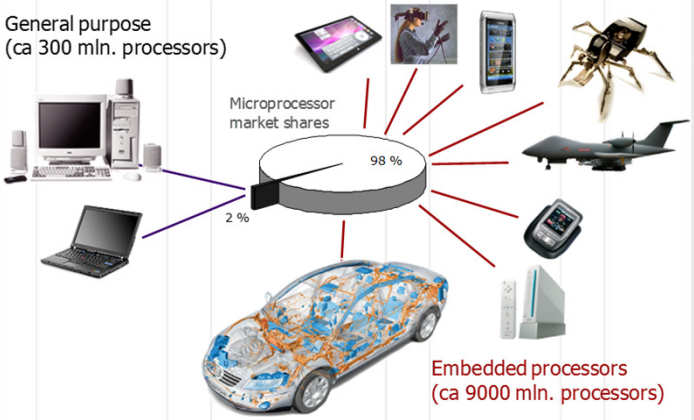

Applications
• Mission-critical applications
– it is important to complete the mission
– repair is impossible or prohibitively expensive
• Pioneer 10 was launched 2 March 1970, passed Pluto 13 June 1983
– requirements
• 95% probability to be operational at the end of mission (e.g. 10 years)
• may be degraded or reconfigured before (operator interaction possible)
© Gert Jervan
51


Applications
• Business-critical applications
– users want to have a high probability of receiving service when it is requested
– transaction processing (banking, stock exchange or other time-shared systems)
• ATM: < 10 hours/year unavailable
• airline reservation: < 1 min/day unavailable
© Gert Jervan
52


Applications
• Maintenance postponement applications
– avoid unscheduled maintenance
– should continue to function until next planned repair (economical benefits)
– examples:
• remotely controlled systems
• telephone switching systems (in remote areas)
© Gert Jervan
53


Embedded Systems
• Computing systems are everywhere
• Most of us think of "desktop" computers
– PC's
– Laptops
– Mainframes
– Servers
• But there's another type of computing system
– Far more common...
© Gert Jervan
54

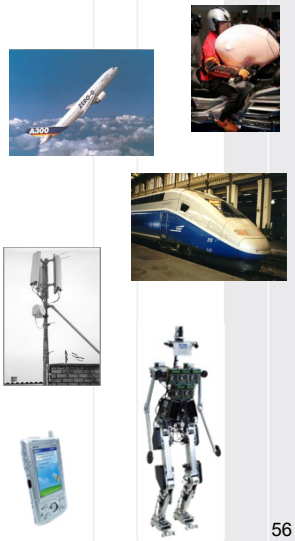
General-Purpose vs. Embedded



55

Embedded Systems, cont.

- Embedded computing systems
  - Computing systems embedded within electronic devices
  - Hard to define. Nearly any computing system other than a desktop computer
  - Billions of units produced yearly, versus millions of desktop units
  - Perhaps 50 per household and per automobile
  - „Internet of things"
  - SmartX (buildings, homes, communities, ...)



56

A "Short List" of Embedded Systems

Anti-lock brakes
Auto-focus cameras
Automatic teller machines
Automobile transmission
Avionic systems
Battery chargers
Cameras
Cell phones
Cell-phone base stations
Cordless phones
Cruise check-in systems
Digital cameras
Disk drives
Electronic card readers
Electronic instruments
Electronic toys/games
Factory controls
Fax machines
Fingerprint identifiers
Home security systems
Life-support systems
Medical testing systems
Modems
MP3 players
Network switches/routers
On-board navigation
Pagers
Photocopiers
Point-of-sale systems
Portable video
Printers
Satellite phones
Scanners
Smart toys
Speech recognizers
Stereo systems
Teleconferencing systems
Televisions
Temperature controllers
Theft tracking systems
TV set-top boxes
VCRs
Video game consoles
Video phones
Washers and dryers

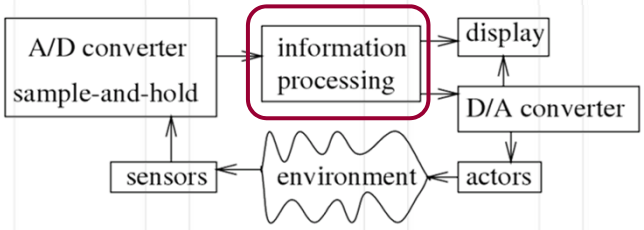
Ambient Intelligence
Medical Systems
Telecommunications
Transport systems
Security & Military
Entertainment

Our daily lives depend on embedded systems

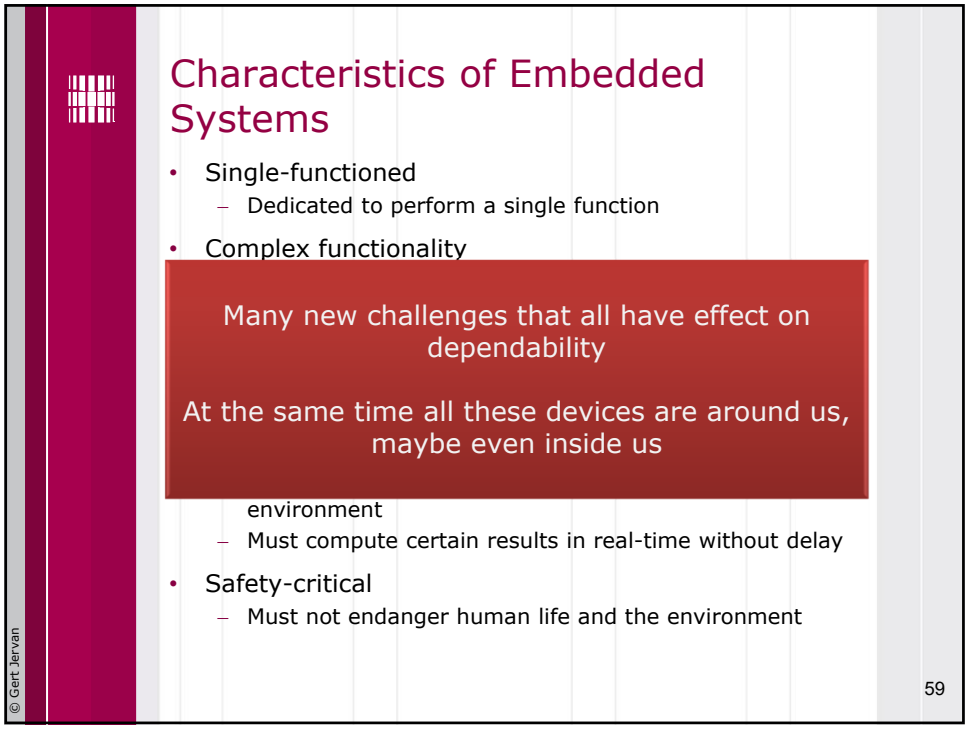
57

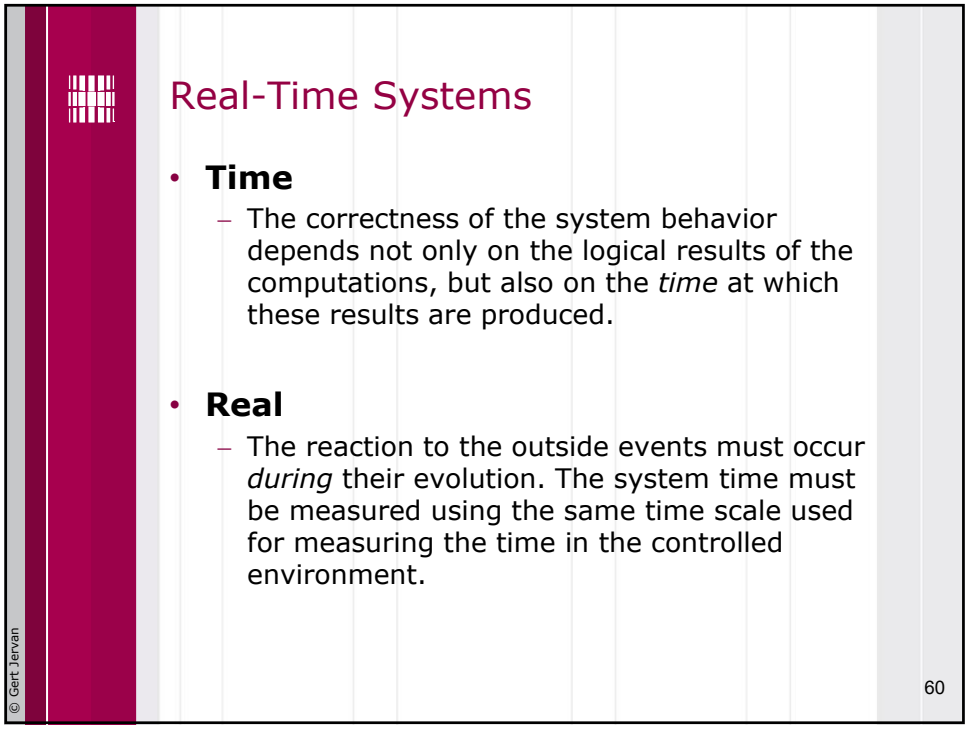
What is an Embedded System?

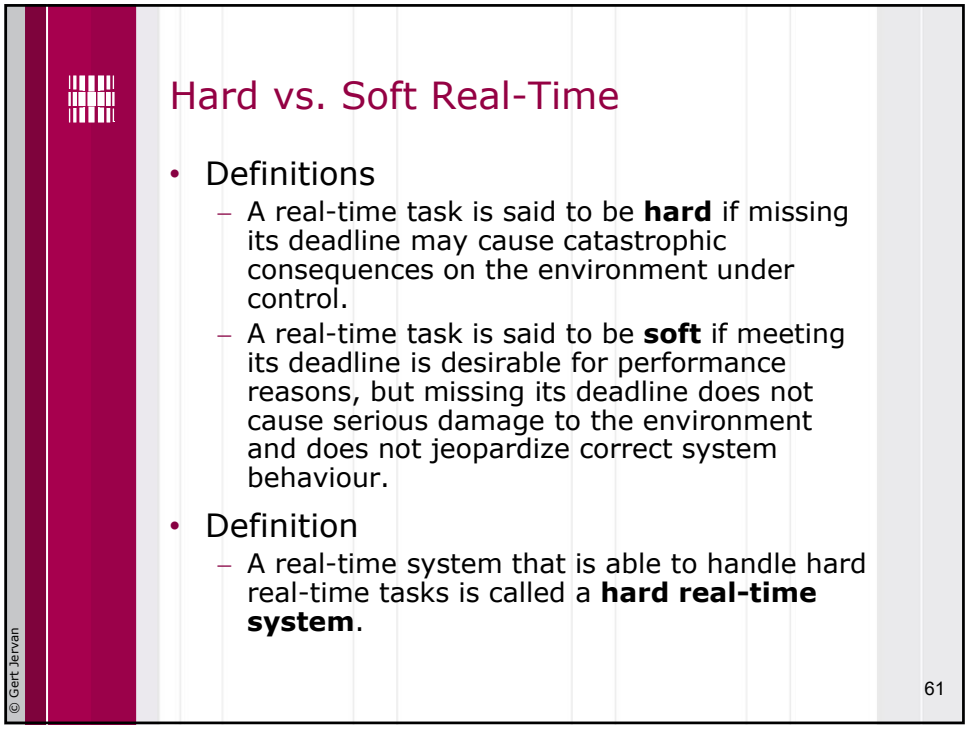
- Definition
  - an **embedded system** special-purpose computer system, part of a larger system which it controls.
- Notes
  - A computer is used in such devices primarily as a means to simplify the system design and to provide flexibility.
  - Often the user of the device is not even aware that a computer is present.

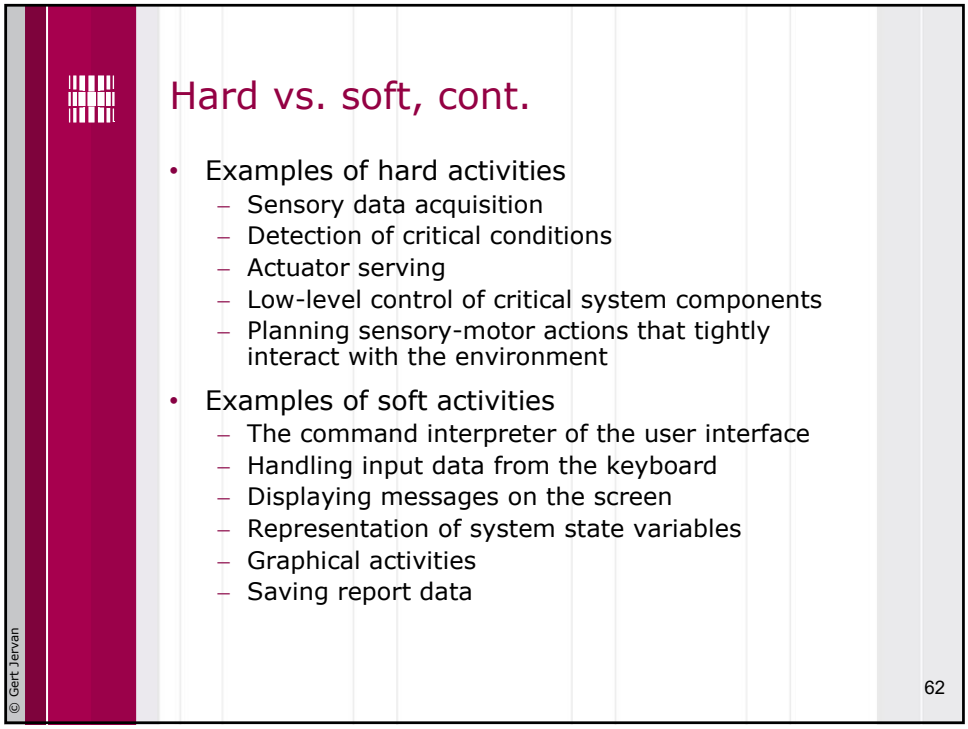


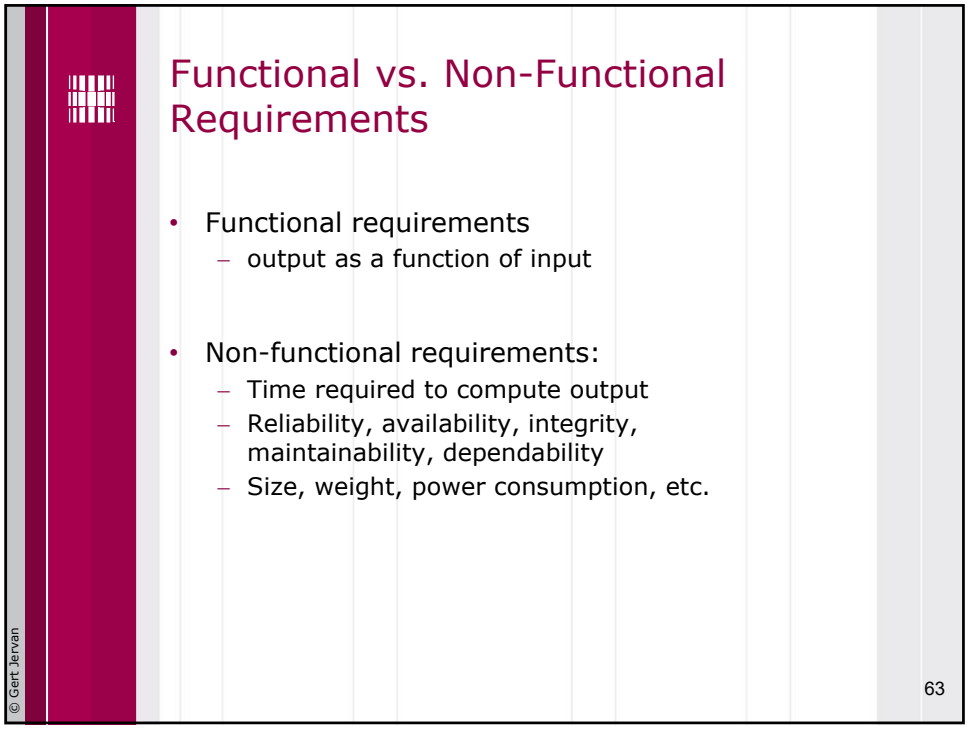
58

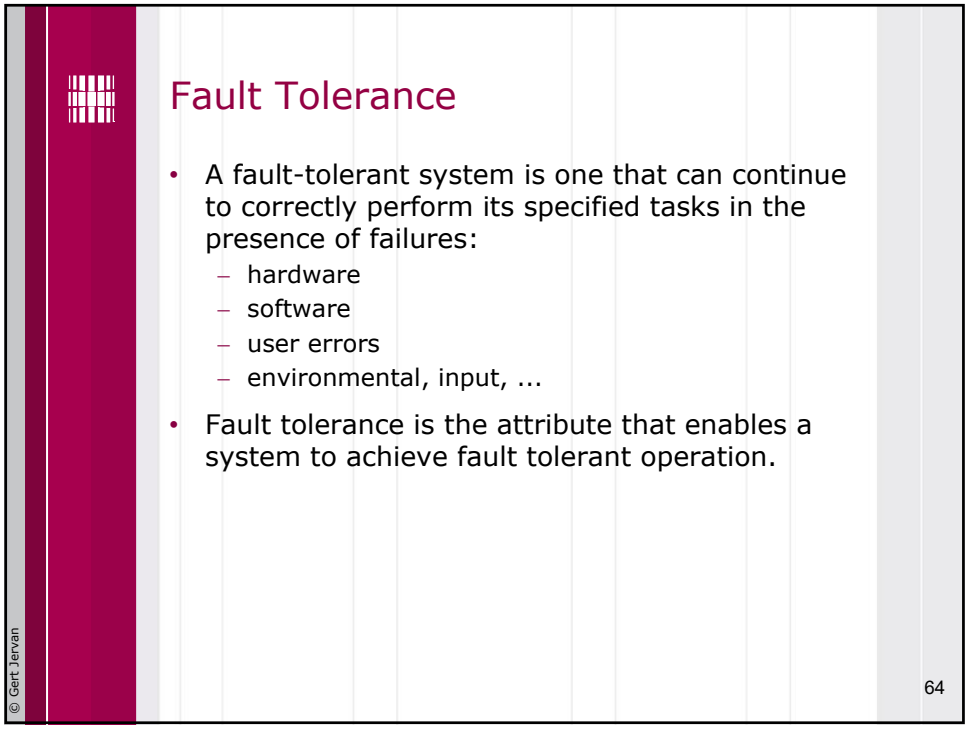

Characteristics of Embedded Systems
• Single-functioned
– Dedicated to perform a single function
• Complex functionality
Many new challenges that all have effect on dependability
At the same time all these devices are around us, maybe even inside us
environment
– Must compute certain results in real-time without delay
• Safety-critical
– Must not endanger human life and the environment
© Gert Jervan
59

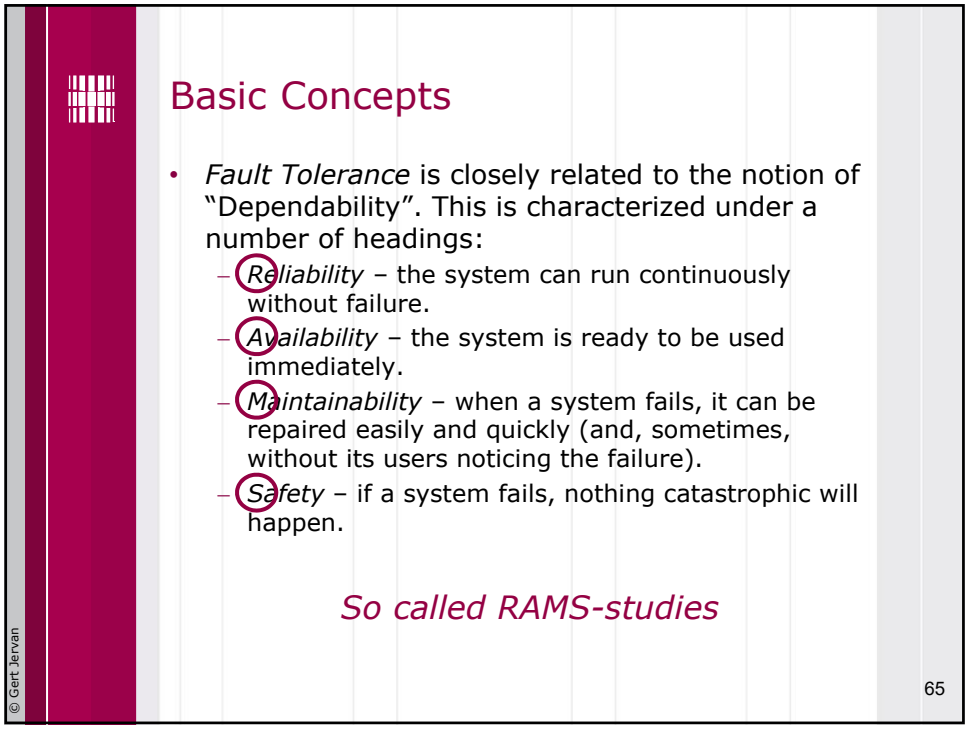

Real-Time Systems
• Time
– The correctness of the system behavior depends not only on the logical results of the computations, but also on the *time* at which these results are produced.
• Real
– The reaction to the outside events must occur *during* their evolution. The system time must be measured using the same time scale used for measuring the time in the controlled environment.
© Gert Jervan
60

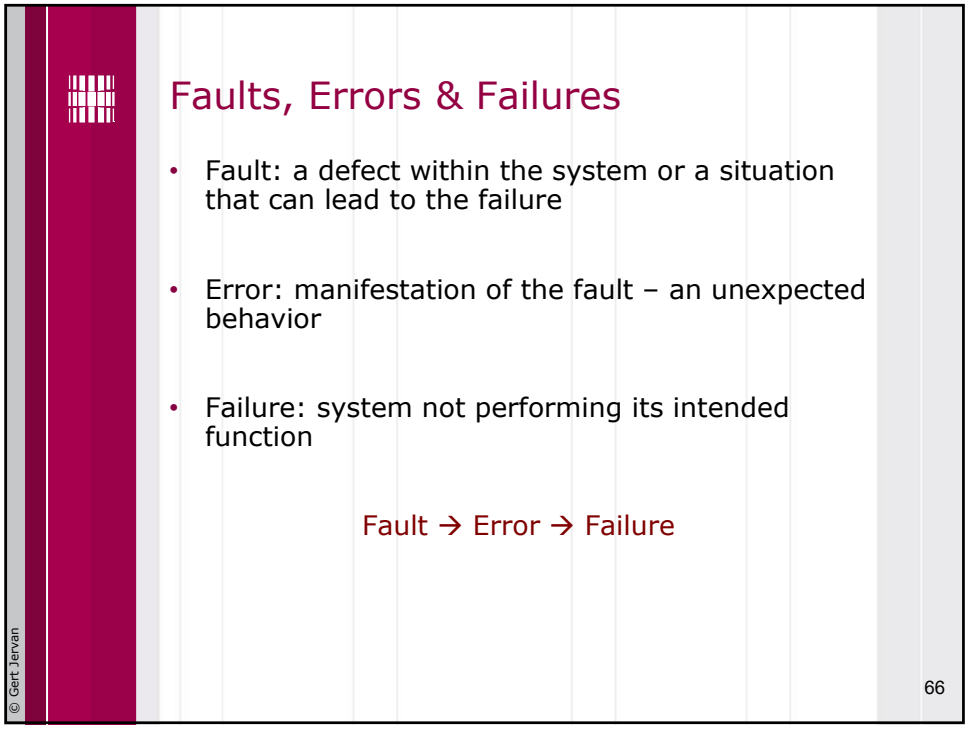

Hard vs. Soft Real-Time
• Definitions
– A real-time task is said to be **hard** if missing its deadline may cause catastrophic consequences on the environment under control.
– A real-time task is said to be **soft** if meeting its deadline is desirable for performance reasons, but missing its deadline does not cause serious damage to the environment and does not jeopardize correct system behaviour.
• Definition
– A real-time system that is able to handle hard real-time tasks is called a **hard real-time system**.
© Gert Jervan
61


Hard vs. soft, cont.
• Examples of hard activities
– Sensory data acquisition
– Detection of critical conditions
– Actuator serving
– Low-level control of critical system components
– Planning sensory-motor actions that tightly interact with the environment
• Examples of soft activities
– The command interpreter of the user interface
– Handling input data from the keyboard
– Displaying messages on the screen
– Representation of system state variables
– Graphical activities
– Saving report data
© Gert Jervan
62


Functional vs. Non-Functional Requirements
• Functional requirements
– output as a function of input
• Non-functional requirements:
– Time required to compute output
– Reliability, availability, integrity, maintainability, dependability
– Size, weight, power consumption, etc.
© Gert Jervan
63


Fault Tolerance
• A fault-tolerant system is one that can continue to correctly perform its specified tasks in the presence of failures:
– hardware
– software
– user errors
– environmental, input, ...
• Fault tolerance is the attribute that enables a system to achieve fault tolerant operation.
© Gert Jervan
64


Basic Concepts
• Fault Tolerance is closely related to the notion of "Dependability". This is characterized under a number of headings:
– Reliability – the system can run continuously without failure.
– Availability – the system is ready to be used immediately.
– Maintainability – when a system fails, it can be repaired easily and quickly (and, sometimes, without its users noticing the failure).
– Safety – if a system fails, nothing catastrophic will happen.
So called RAMS-studies
© Gert Jervan
65


Faults, Errors & Failures
• Fault: a defect within the system or a situation that can lead to the failure
• Error: manifestation of the fault – an unexpected behavior
• Failure: system not performing its intended function
Fault → Error → Failure
© Gert Jervan
66

Measuring

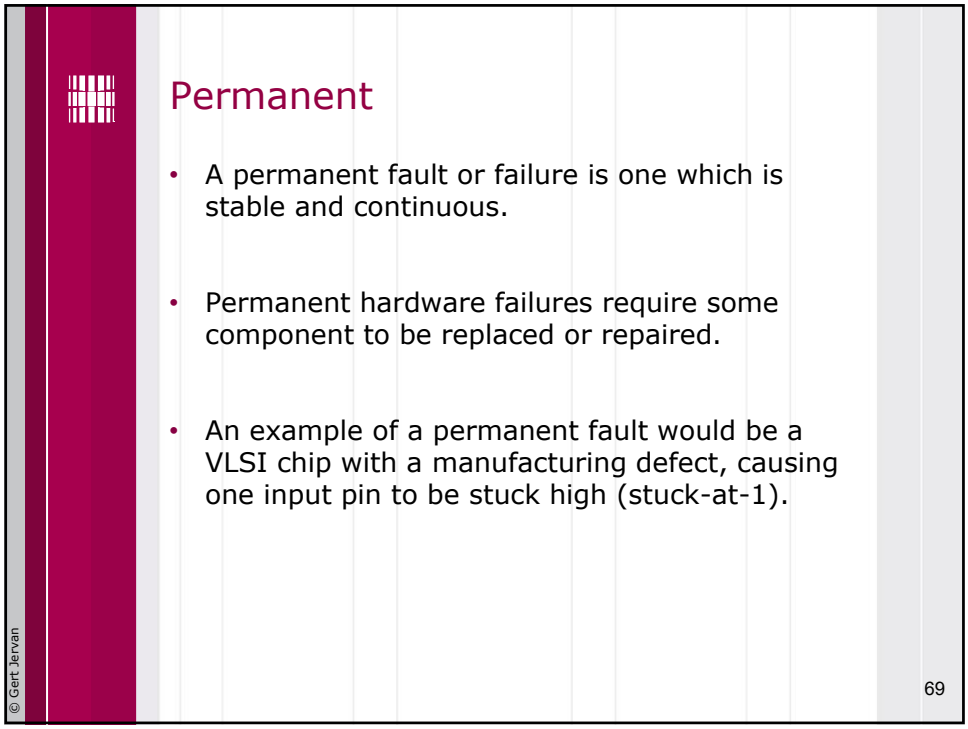
- Failures are measured in FITs
  - 1 FIT (failures in time), is the number of failures in 1 billion device-operation hours. A measurement of 1000 FITs corresponds to a MTTF (mean time to failure) of approximately 114 years.
- Example: Bit flips in hardware due to cosmic radiation
  - A person on an airplane over the Atlantic at 35,000 ft working on a laptop with 256 Mbytes (2 Gbits) of memory. At this altitude, the soft error rate (SER) of 600 FITs per megabit becomes 100,000 FITs per megabit, resulting in a potential error every five hours.

67

Fault Examples

- Year 2000 bug
- Loose wire
- Aircraft retracting its landing gear (while on ground)
- Effects in time:
  - Permanent
  - Transient
  - Intermittent

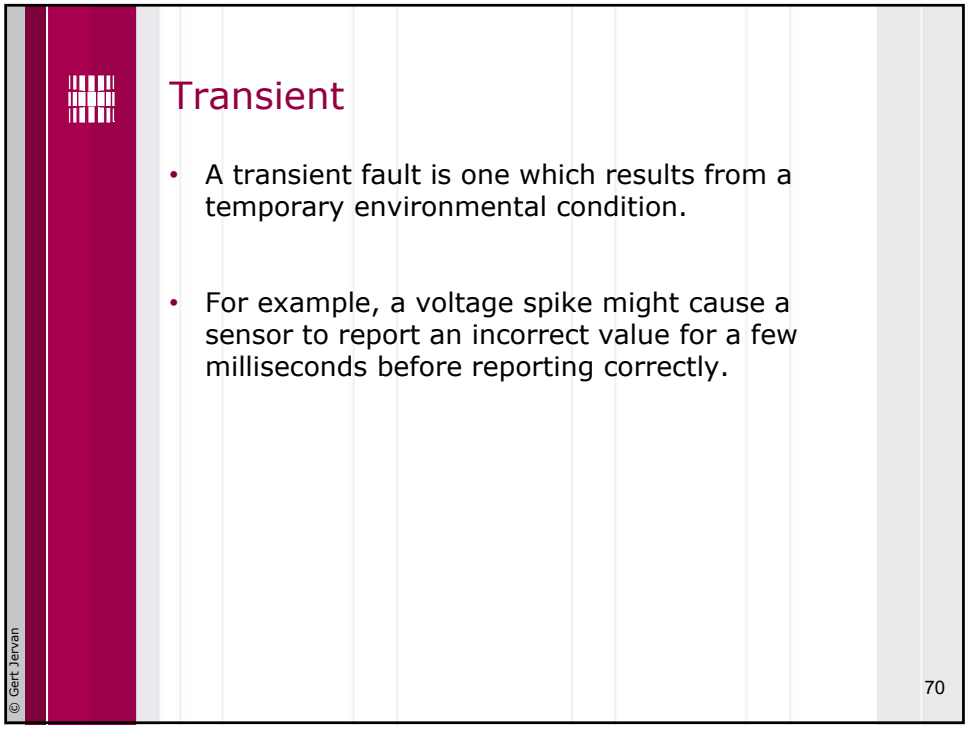
68



Permanent

- A permanent fault or failure is one which is stable and continuous.
- Permanent hardware failures require some component to be replaced or repaired.
- An example of a permanent fault would be a VLSI chip with a manufacturing defect, causing one input pin to be stuck high (stuck-at-1).

69



Transient

- A transient fault is one which results from a temporary environmental condition.
- For example, a voltage spike might cause a sensor to report an incorrect value for a few milliseconds before reporting correctly.

70

Transient faults

- Happen for a short time
- **Corruptions of data, miscalculation in logic**
- Do not cause a permanent damage of circuits
- Causes are outside system boundaries

Electromagnetic interference (EMI)

Radiation

Lightning storms

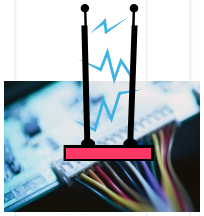
71

Intermittent

- An intermittent fault is one which only manifests occasionally, due to unstable hardware or certain system states.
- A loose contact on a connector will often cause an intermittent fault.
- Intermittent electrical faults, as a rule, are notoriously difficult to detect. Typically, whenever the fault doctor shows up, the system works fine.

72

Intermittent faults



Crosstalk

Internal EMI

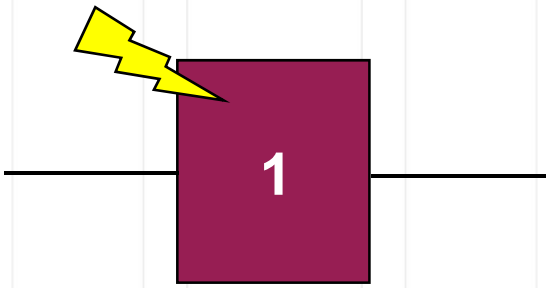
- Manifest similar as transient faults
- Happen repeatedly
- Causes are inside system boundaries

Software errors (Heisenbugs)

Power supply fluctuations

73

Soft Errors



- Transient bit-flip (soft memory error)
  - Random event
  - Corrupts the value but not the cell
  - Can be corrected (in contrast to hard errors caused by faults in the hardware itself)
  - Happen continuously during system lifetime (i.e., can not be screened by burn-in tests)

74

Sources

- First traced to alpha particlce emissions from chip packaging materials
  - Most sources removed (pure materials, different designs, shielding)
- Today's main problem: cosmic radiation
  - Cosmic particles from deep space (actually 5th- or 6th-hand collision particles)
    - At ground level ca 95% neutrons, 5% protons
  - Radioactive material in manufacturing process

75

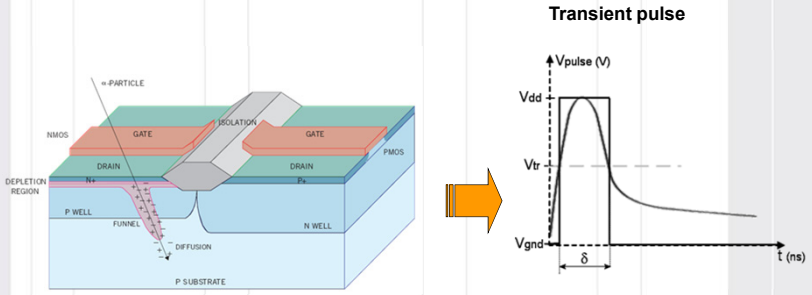
Sources (cont.)

- Four main sources:
  - Low-energy alpha particles
  - High-energy cosmic particles
  - Thermal neutrons
  - Poor system design

SER type	Source	Mechanism	Trend
Alpha	Thorium and uranium contamination in-mold compound, silicon, or lead bumps	2- to 9-MeV alpha particle creating electron-hole funnel traveling 25 microns in silicon	Exponential increase with scaling
Cosmic	Intergalactic sources modulated by solar flares	High-energy neutrons/protons (10 MeV to 1 GeV) colliding with silicon nuclei	Decrease in failures in time per megabit
Thermal neutron	Boron present in BPSG25-meV neutrons	Collision with B10 in BPSG	Highest, always dominates if present

76

Soft Errors



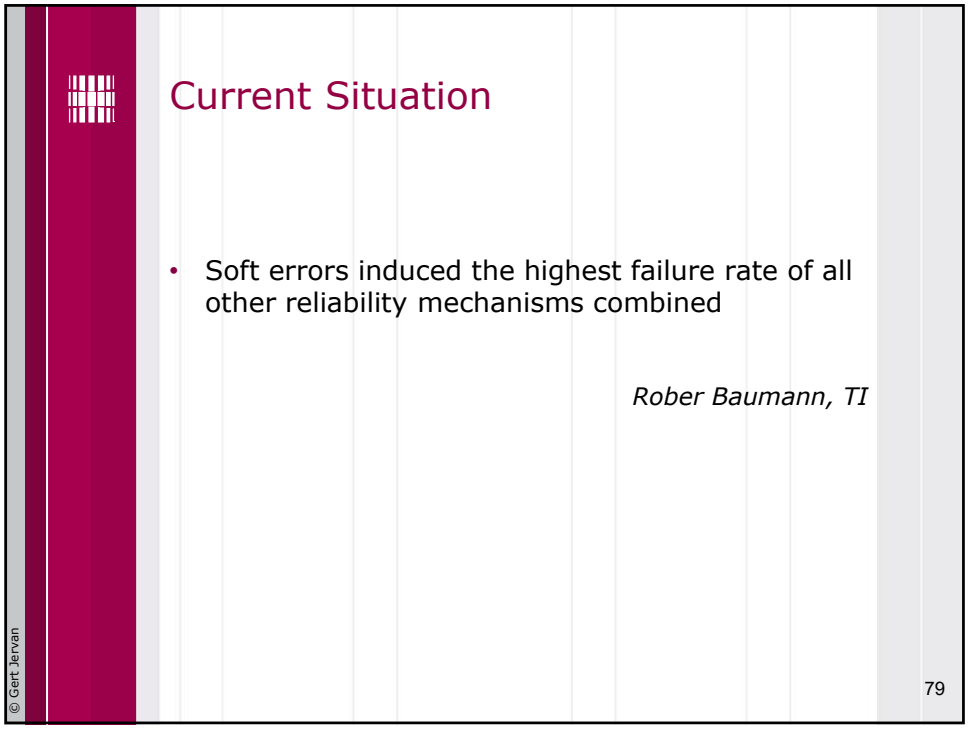
The electric field in the depletion region directly generates electron-hole pairs in its wake, causing the charges to drift so that the transistor sees a current disturbance

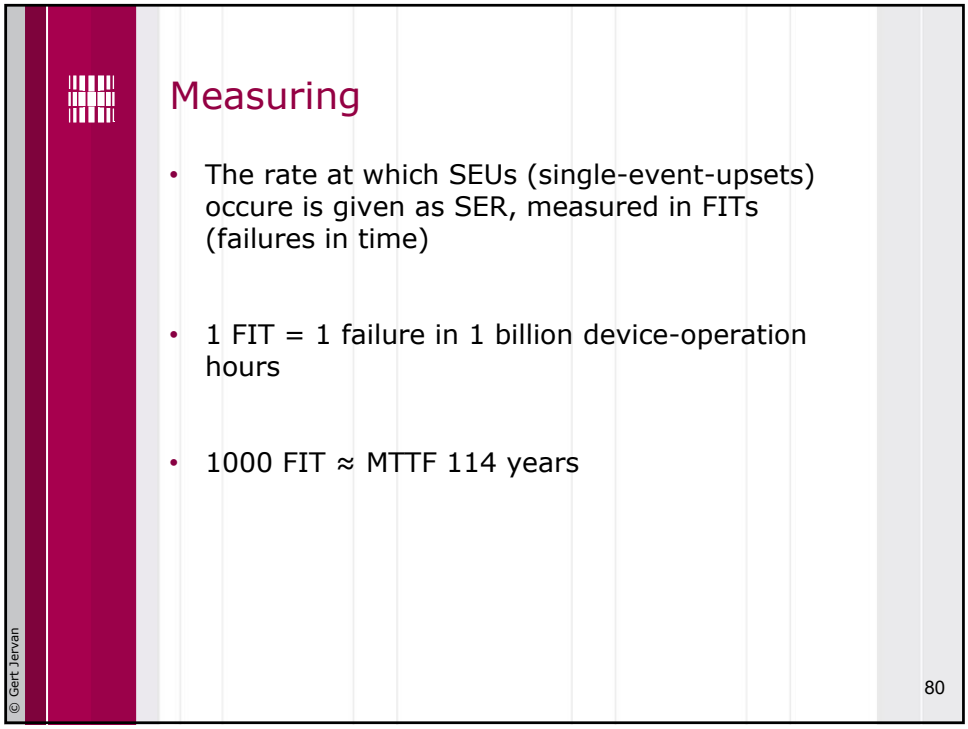
77

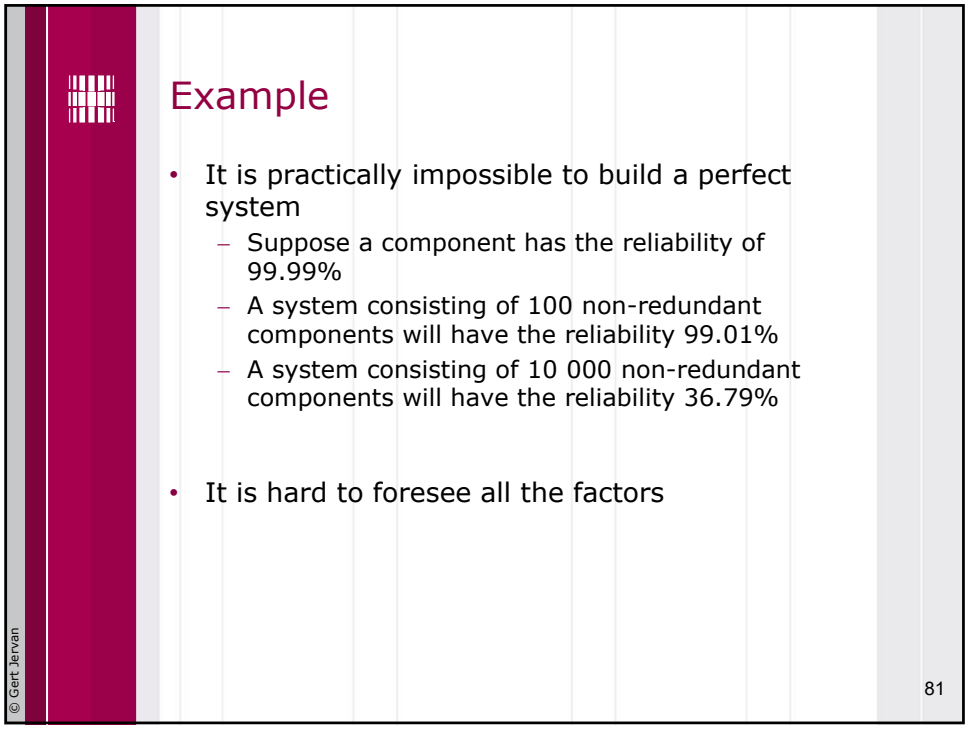
Evidence of Cosmic Ray Strikes

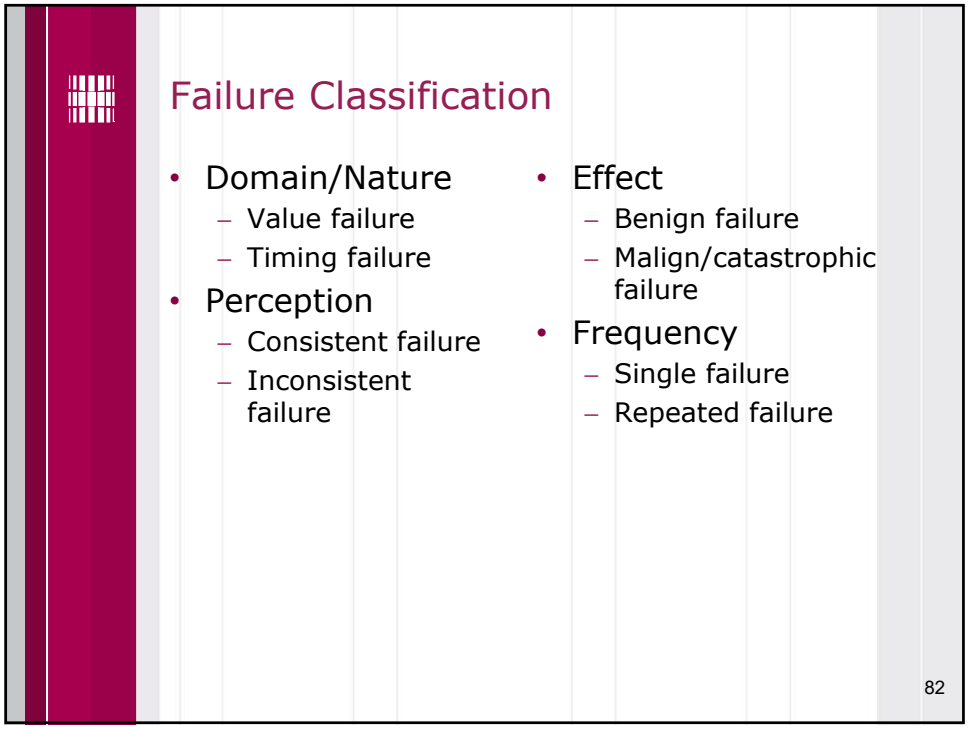
- Documented strikes in large servers found in error logs
  - Normand, "Single Event Upset at Ground Level," IEEE Transactions on Nuclear Science, Vol. 43, No. 6, December 1996.
- Sun Microsystems, 2000 (R. Baumann, Workshop talk)
  - Cosmic ray strikes on L2 cache with defective error protection
    - caused Sun's flagship servers to suddenly and mysteriously crash!
  - Companies affected
    - Baby Bell (Atlanta), America Online, Ebay, & dozens of other corporations
    - Verisign moved to IBM Unix servers (for the most part)
- 2005 – Los Alamos 2048-CPU HP server system crashed frequently due to defective cache
- 2010 Toyota brake problem (still no agreement)
- More recently: problems with GPGPU based HPC

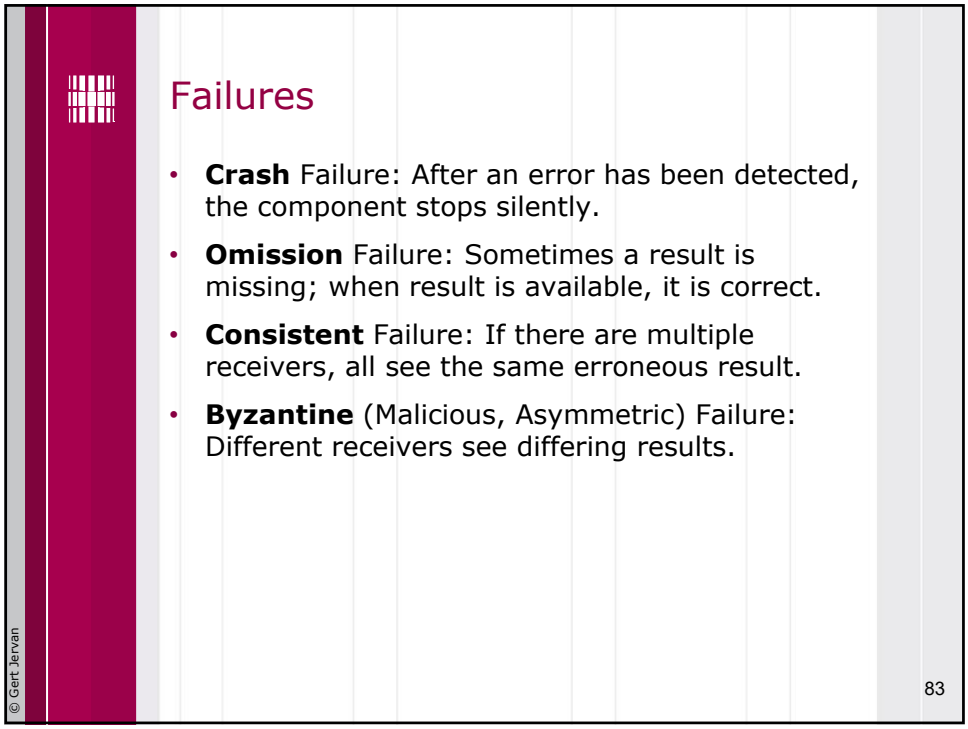
78

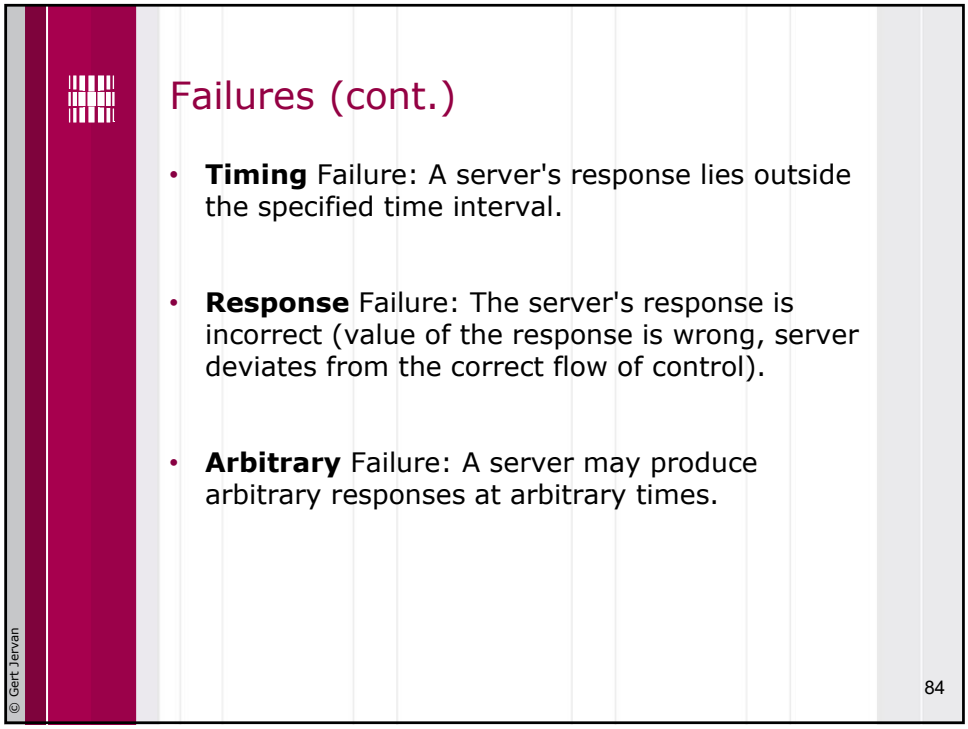

Current Situation
• Soft errors induced the highest failure rate of all other reliability mechanisms combined
Rober Baumann, TI
© Gert Jervan
79

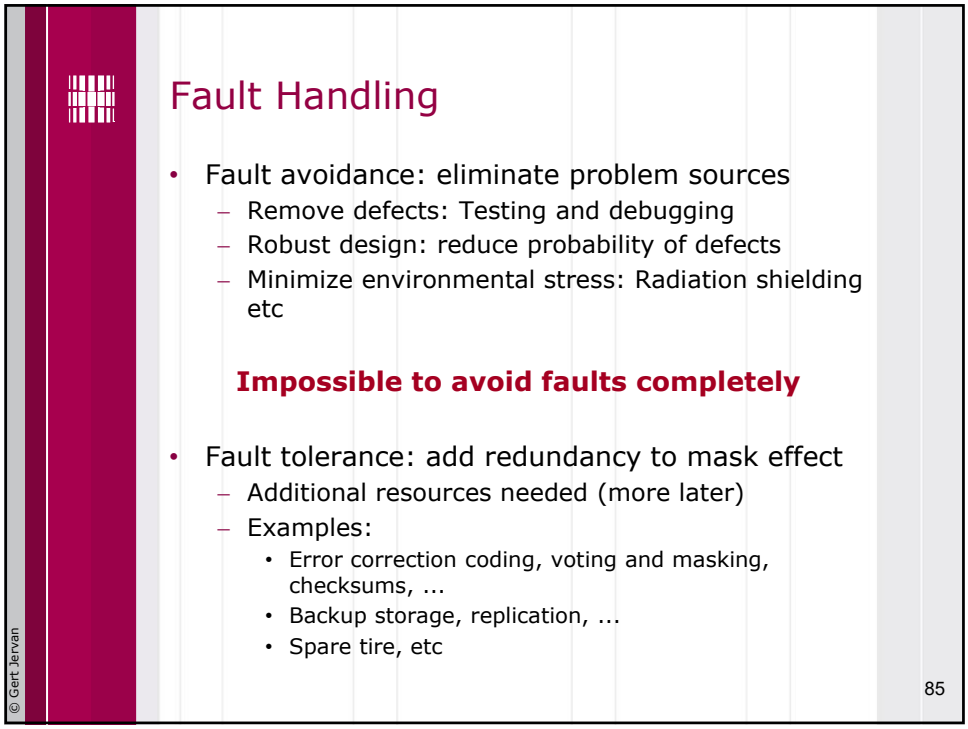

Measuring
• The rate at which SEUs (single-event-upsets) occure is given as SER, measured in FITs (failures in time)
• 1 FIT = 1 failure in 1 billion device-operation hours
• 1000 FIT ≈ MTTF 114 years
© Gert Jervan
80


Example
• It is practically impossible to build a perfect system
– Suppose a component has the reliability of 99.99%
– A system consisting of 100 non-redundant components will have the reliability 99.01%
– A system consisting of 10 000 non-redundant components will have the reliability 36.79%
• It is hard to foresee all the factors
© Gert Jervan
81


Failure Classification
• Domain/Nature
– Value failure
– Timing failure
• Perception
– Consistent failure
– Inconsistent failure
• Effect
– Benign failure
– Malign/catastrophic failure
• Frequency
– Single failure
– Repeated failure
82


Failures
• Crash Failure: After an error has been detected, the component stops silently.
• Omission Failure: Sometimes a result is missing; when result is available, it is correct.
• Consistent Failure: If there are multiple receivers, all see the same erroneous result.
• Byzantine (Malicious, Asymmetric) Failure: Different receivers see differing results.
© Gert Jervan
83


Failures (cont.)
• Timing Failure: A server's response lies outside the specified time interval.
• Response Failure: The server's response is incorrect (value of the response is wrong, server deviates from the correct flow of control).
• Arbitrary Failure: A server may produce arbitrary responses at arbitrary times.
© Gert Jervan
84



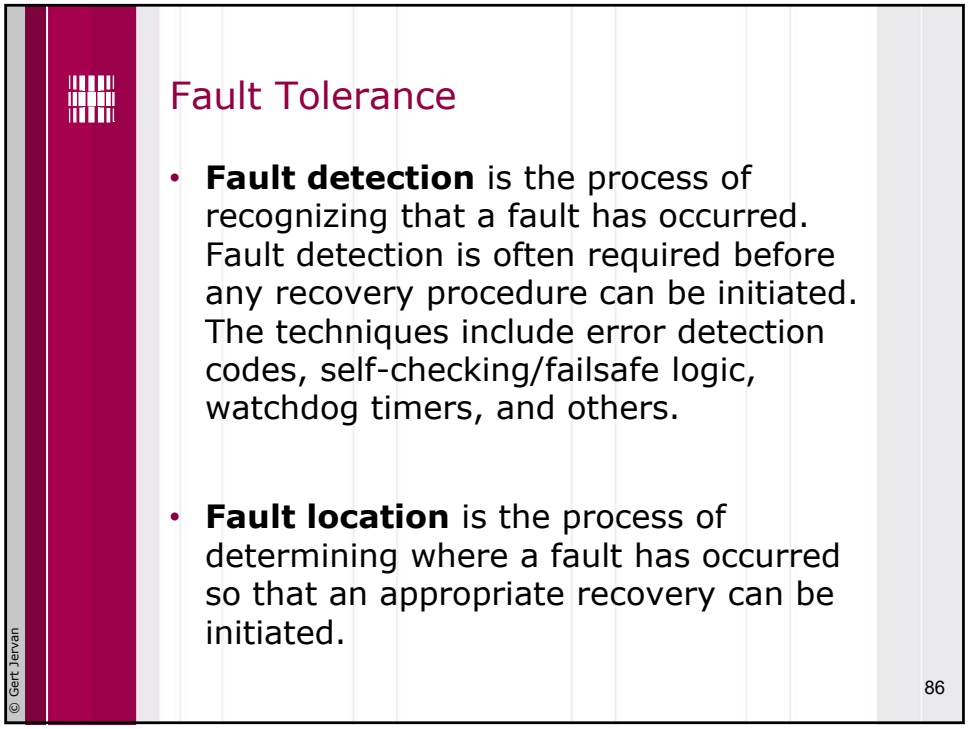
Fault Handling

- Fault avoidance: eliminate problem sources
  - Remove defects: Testing and debugging
  - Robust design: reduce probability of defects
  - Minimize environmental stress: Radiation shielding etc

Impossible to avoid faults completely

- Fault tolerance: add redundancy to mask effect
  - Additional resources needed (more later)
  - Examples:
    - Error correction coding, voting and masking, checksums, ...
    - Backup storage, replication, ...
    - Spare tire, etc

85



Fault Tolerance

- **Fault detection** is the process of recognizing that a fault has occurred. Fault detection is often required before any recovery procedure can be initiated. The techniques include error detection codes, self-checking/failsafe logic, watchdog timers, and others.
- **Fault location** is the process of determining where a fault has occurred so that an appropriate recovery can be initiated.

86

Fault Tolerance (cont.)

- **Fault containment** is the process of isolating a fault and preventing the effects of that fault from propagating throughout the system.
- **Fault recovery** is the process of remaining operational or regaining operational status via reconfiguration even in the presence of faults. A few basic approaches are fault masking, retry, and rollback.

87

Definitions

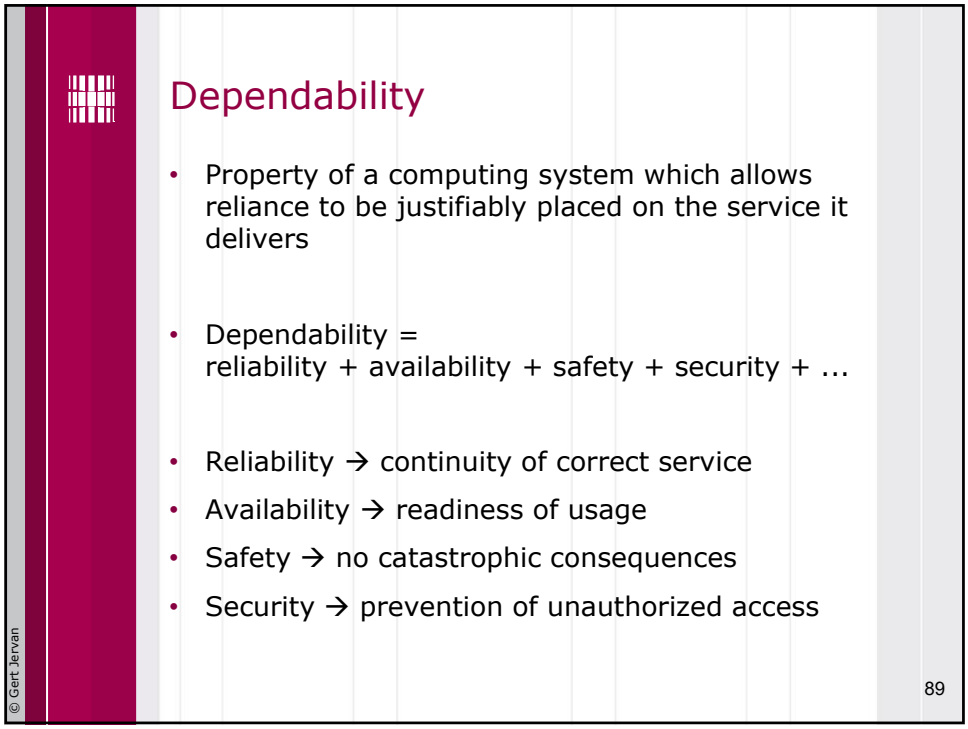
- Failure rate (λ):
  - Average frequency with which something fails.

$$\frac{6\ failures}{7502\ hrs} = 0.0007998\ failures/hr = 799.8 \times 10^{-6}\ failures/hr$$

- Mean time to failure (MTTF):
  - Average time between failures

$$MTTF = \frac{1}{\lambda}$$

88


Dependability
• Property of a computing system which allows reliance to be justifiably placed on the service it delivers
• Dependability = reliability + availability + safety + security + ...
• Reliability → continuity of correct service
• Availability → readiness of usage
• Safety → no catastrophic consequences
• Security → prevention of unauthorized access
© Gert Jervan
89