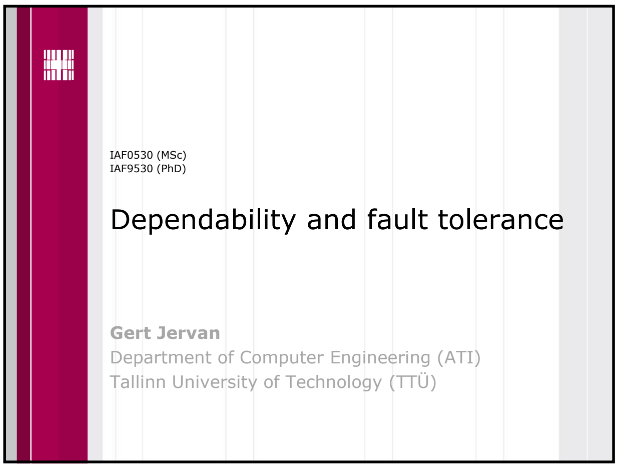
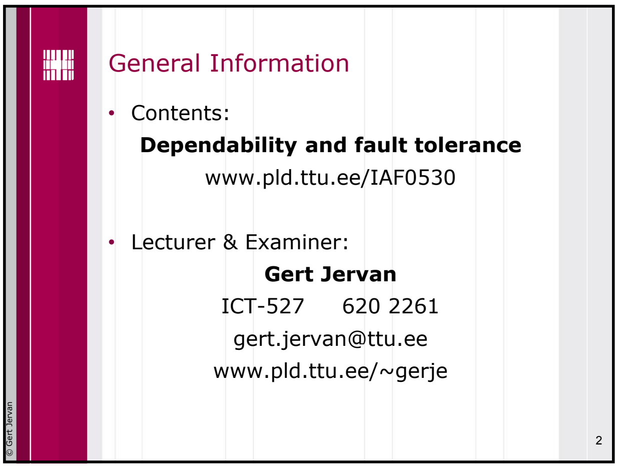


IAF0530 (MSc)
IAF9530 (PhD)

Dependability and fault tolerance

Gert Jervan
Department of Computer Engineering (ATI)
Tallinn University of Technology (TTÜ)



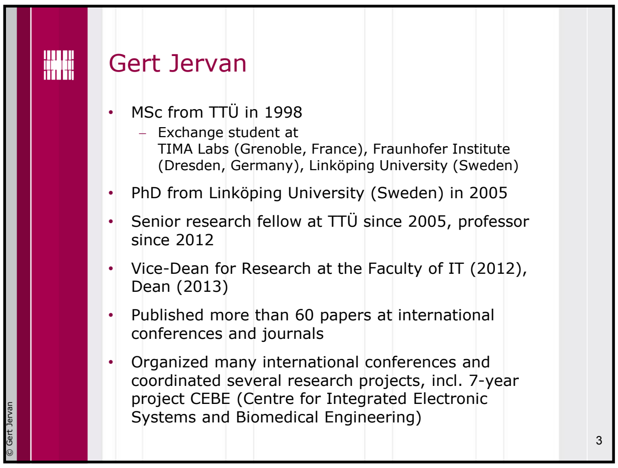
General Information

- Contents:

  Dependability and fault tolerance
  www.pld.ttu.ee/IAF0530

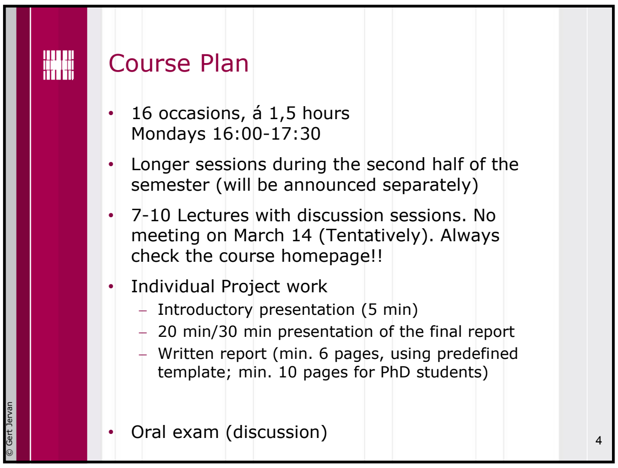
- Lecturer & Examiner:

  Gert Jervan
  ICT-527 620 2261
  gert.jervan@ttu.ee
  www.pld.ttu.ee/~gerje



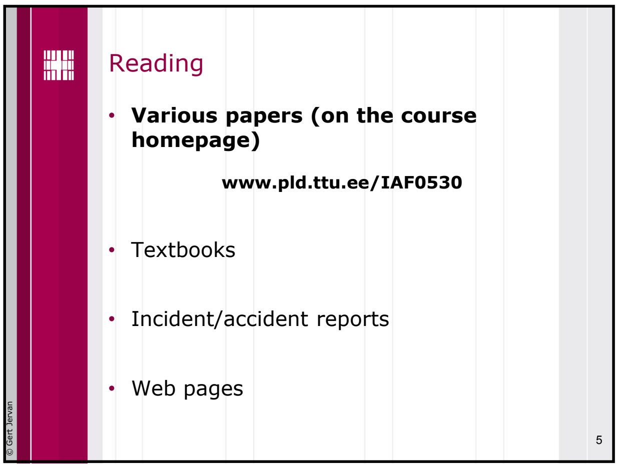
Gert Jervan

- MSc from TTÜ in 1998
  - Exchange student at TIMA Labs (Grenoble, France), Fraunhofer Institute (Dresden, Germany), Linköping University (Sweden)
- PhD from Linköping University (Sweden) in 2005
- Senior research fellow at TTÜ since 2005, professor since 2012
- Vice-Dean for Research at the Faculty of IT (2012), Dean (2013)
- Published more than 60 papers at international conferences and journals
- Organized many international conferences and coordinated several research projects, incl. 7-year project CEBE (Centre for Integrated Electronic Systems and Biomedical Engineering)



Course Plan

- 16 occasions, á 1,5 hours
  Mondays 16:00-17:30
- Longer sessions during the second half of the semester (will be announced separately)
- 7-10 Lectures with discussion sessions. No meeting on March 14 (Tentatively). Always check the course homepage!!
- Individual Project work
  - Introductory presentation (5 min)
  - 20 min/30 min presentation of the final report
  - Written report (min. 6 pages, using predefined template; min. 10 pages for PhD students)
- Oral exam (discussion)

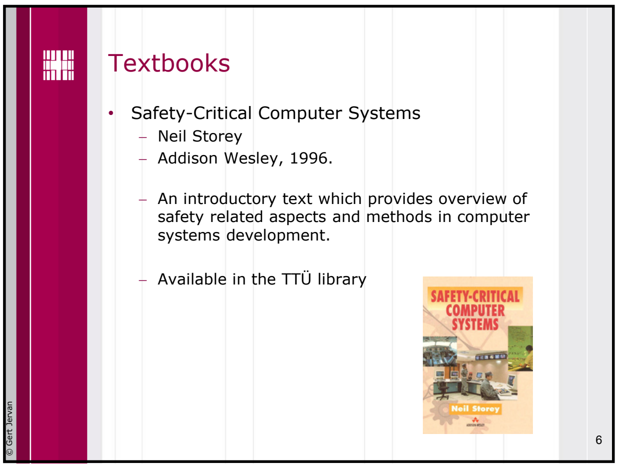


Reading

- **Various papers (on the course homepage)**

  www.pld.ttu.ee/IAF0530

- Textbooks
- Incident/accident reports
- Web pages

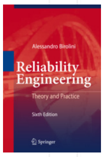


Textbooks

- Safety-Critical Computer Systems
  - Neil Storey
  - Addison Wesley, 1996.
  - An introductory text which provides overview of safety related aspects and methods in computer systems development.
  - Available in the TTÜ library

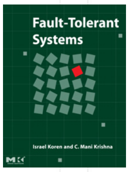
SAFETY-CRITICAL COMPUTER SYSTEMS
Neil Storey

Textbooks



- Reliability Engineering: Theory and Practice.
  - Alessandro Birolini
  - Springer
  - 2014 (7th ed.) 2010 (6th ed.), 2007 (5th ed.)
  - This book shows how to build in, evaluate, and demonstrate reliability & availability of components, equipment, systems. It presents the state-of-the-art of reliability engineering, both in theory and practice
  - TTÜ library has several copies of the latest edition.

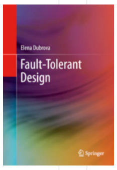
Textbooks



- Fault-Tolerant Systems
  - Israel Koren and C. Mani Krishna
  - Morgan-Kaufman Publishers, 2007

This book covers comprehensively the design of fault-tolerant hardware and software, use of fault-tolerance techniques to improve manufacturing yields and design and analysis of networks. Additionally it includes material on methods to protect against threats to encryption subsystems used for security purposes.

Textbooks



- Fault-Tolerant Design
  - Elena Dubrova
  - Springer, 2013
- This textbook serves as an introduction to fault-tolerance, intended for upper-division undergraduate students, graduate-level students and practicing engineers in need of an overview of the field. Readers will develop skills in modeling and evaluating fault-tolerant architectures in terms of reliability, availability and safety. They will gain a thorough understanding of fault tolerant computers, including both the theory of how to design and evaluate them and the practical knowledge of achieving fault-tolerance in electronic, communication and software systems. Coverage includes fault-tolerance techniques through hardware, software, information and time redundancy. The content is designed to be highly accessible, including numerous examples and exercises. Solutions and powerpoint slides are available for instructors.

Case Studies

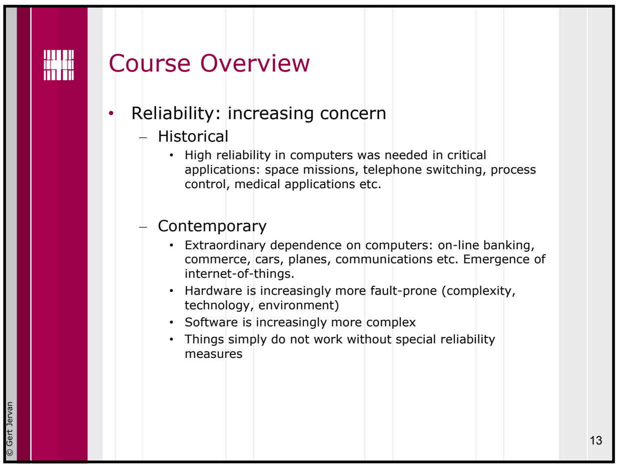
- The exact format will be announced during the second lecture (and it depends of the number of students we will have)
- Topic categories:
  - Accident analysis
  - System safety analysis
  - Literature survey
  - Something else (implementation, tool study, etc.)
  - Requires prior ack.

Literature and sample (!) topics on the webpage

Case Studies

- Some examples:
  - Clock synchronization
  - Atomic and reliable broadcast
  - Algorithmic based fault-tolerance
  - System level diagnosis - distributed algorithms
  - Fault-tolerant transaction processing systems
  - Measures of software reliability
  - SaaS reliability
  - Validation and verification techniques
  - CAN (Controller Area Network) protocol
  - Fault-Tolerance in Web Servers
  - Fault tolerance in wired and wireless systems
  - Nano tubes
  - Cy-Phy systems dependability
  - …

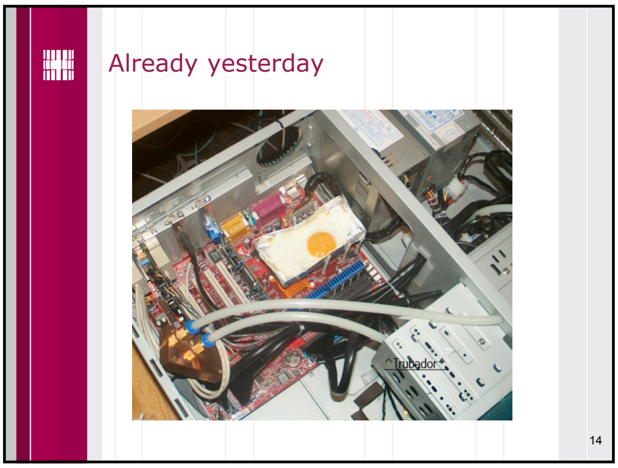
Course overview



Course Overview

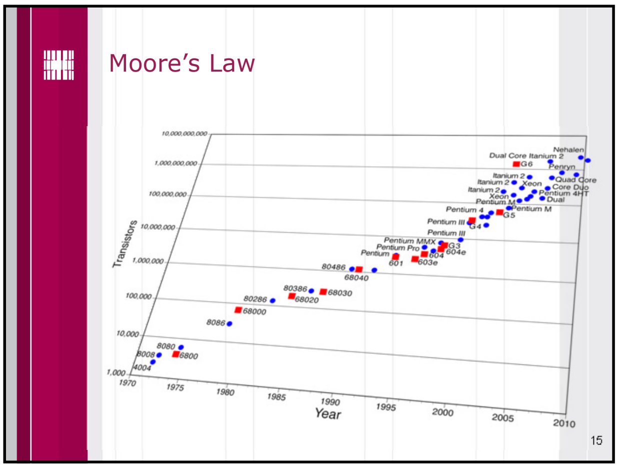
- Reliability: increasing concern
  - Historical
    - High reliability in computers was needed in critical applications: space missions, telephone switching, process control, medical applications etc.
  - Contemporary
    - Extraordinary dependence on computers: on-line banking, commerce, cars, planes, communications etc. Emergence of internet-of-things.
    - Hardware is increasingly more fault-prone (complexity, technology, environment)
    - Software is increasingly more complex
    - Things simply do not work without special reliability measures

13



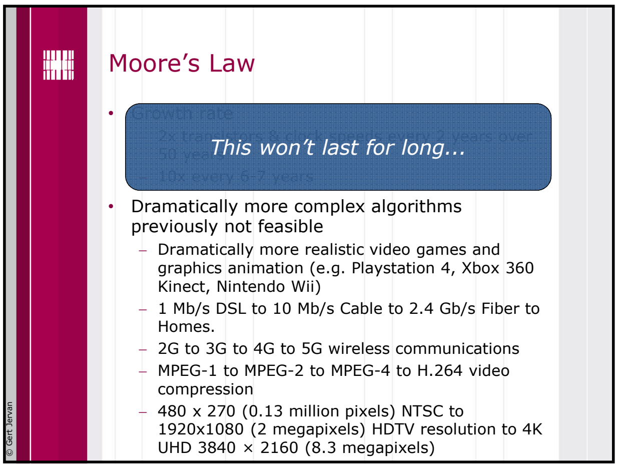
Already yesterday

14



Moore’s Law

15

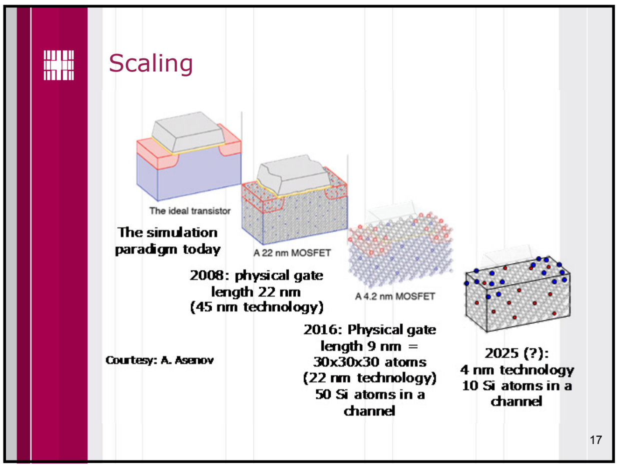


Moore’s Law

- Growth rate
  - 2x transistors & clock speeds every 2 years over 50 years
  - 10x every 6-7 years

This won’t last for long…

- Dramatically more complex algorithms previously not feasible
  - Dramatically more realistic video games and graphics animation (e.g. Playstation 4, Xbox 360 Kinect, Nintendo Wii)
  - 1 Mb/s DSL to 10 Mb/s Cable to 2.4 Gb/s Fiber to Homes.
  - 2G to 3G to 4G to 5G wireless communications
  - MPEG-1 to MPEG-2 to MPEG-4 to H.264 video compression
  - 480 x 270 (0.13 million pixels) NTSC to 1920x1080 (2 megapixels) HDTV resolution to 4K UHD 3840 × 2160 (8.3 megapixels)



Scaling

The simulation paradigm today

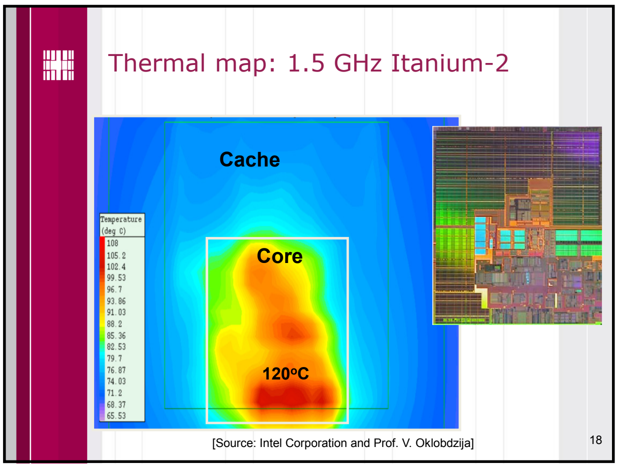
2008: physical gate length 22 nm (45 nm technology)

2016: Physical gate length 9 nm = 30x30x30 atoms (22 nm technology) 50 Si atoms in a channel

2025 (?): 4 nm technology 10 Si atoms in a channel

Courtesy: A. Asenov

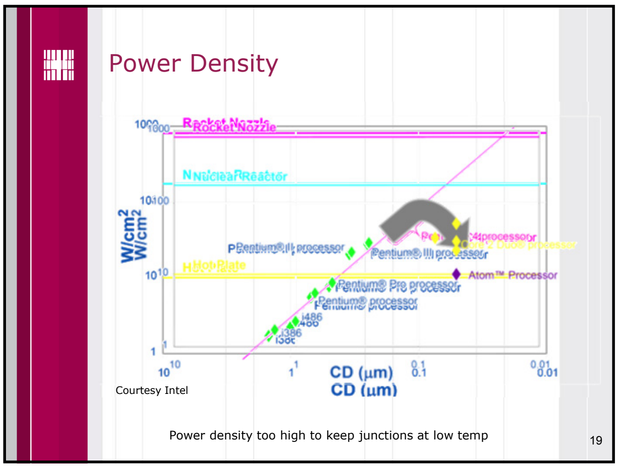
17



Thermal map: 1.5 GHz Itanium-2

[Source: Intel Corporation and Prof. V. Oklobdzija]

18

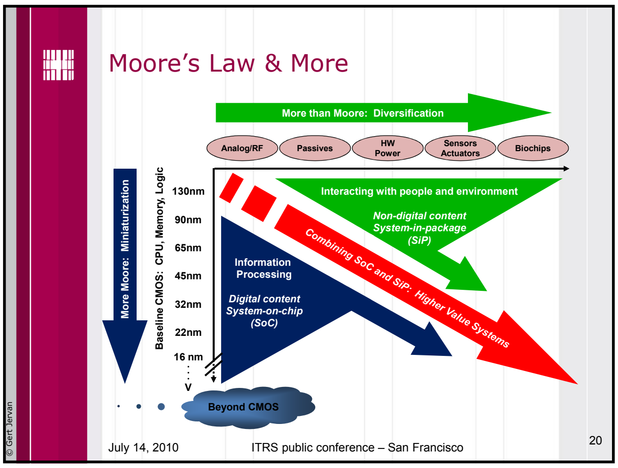


Power Density

Courtesy Intel

Power density too high to keep junctions at low temp

19



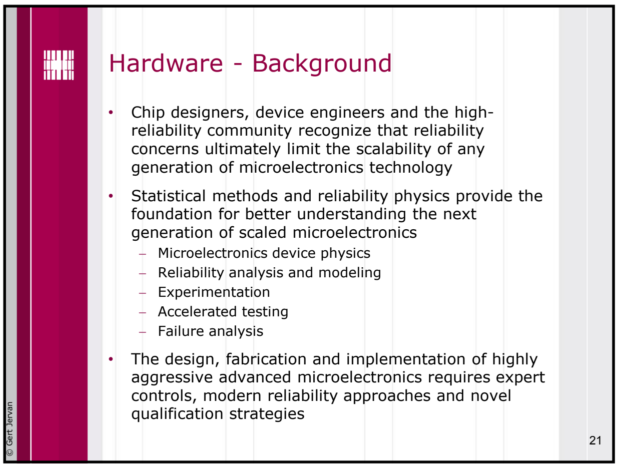
Moore’s Law & More

More than Moore: Diversification
Analog/RF
Passives
HW Power
Sensors Actuators
Biochips
More Moore: Miniaturization
Baseline CMOS: CPU, Memory, Logic
130nm
90nm
65nm
45nm
32nm
22nm
16 nm
V
Interacting with people and environment
Non-digital content System-in-package (SiP)
Information Processing
Digital content System-on-chip (SoC)
Combining SoC and SiP: Higher Value Systems
Beyond CMOS

© Gert Jervan

July 14, 2010 ITRS public conference – San Francisco

20

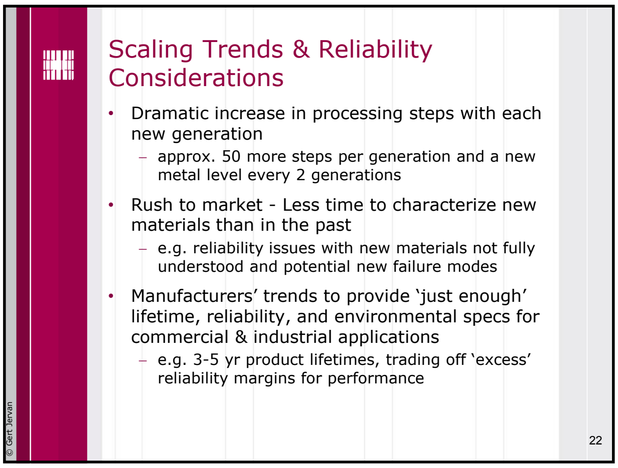


Hardware - Background

- Chip designers, device engineers and the high-reliability community recognize that reliability concerns ultimately limit the scalability of any generation of microelectronics technology
- Statistical methods and reliability physics provide the foundation for better understanding the next generation of scaled microelectronics
  - Microelectronics device physics
  - Reliability analysis and modeling
  - Experimentation
  - Accelerated testing
  - Failure analysis
- The design, fabrication and implementation of highly aggressive advanced microelectronics requires expert controls, modern reliability approaches and novel qualification strategies

© Gert Jervan

21

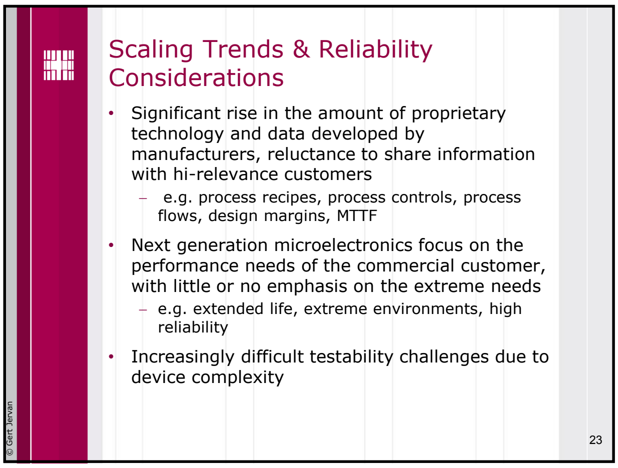


Scaling Trends & Reliability Considerations

- Dramatic increase in processing steps with each new generation
  - approx. 50 more steps per generation and a new metal level every 2 generations
- Rush to market - Less time to characterize new materials than in the past
  - e.g. reliability issues with new materials not fully understood and potential new failure modes
- Manufacturers’ trends to provide ‘just enough’ lifetime, reliability, and environmental specs for commercial & industrial applications
  - e.g. 3-5 yr product lifetimes, trading off ‘excess’ reliability margins for performance

© Gert Jervan

22

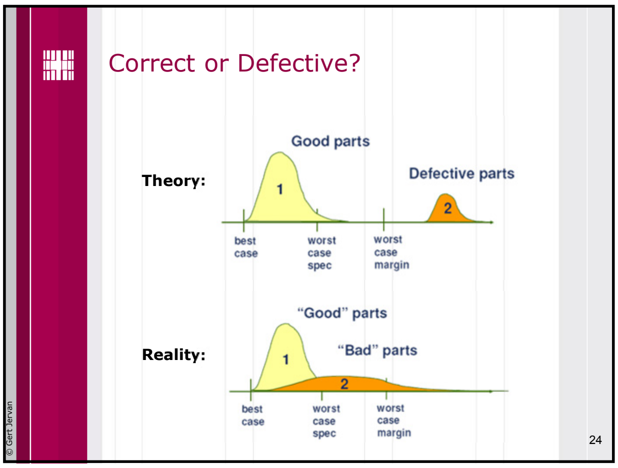


Scaling Trends & Reliability Considerations

- Significant rise in the amount of proprietary technology and data developed by manufacturers, reluctance to share information with hi-relevance customers
  - e.g. process recipes, process controls, process flows, design margins, MTTF
- Next generation microelectronics focus on the performance needs of the commercial customer, with little or no emphasis on the extreme needs
  - e.g. extended life, extreme environments, high reliability
- Increasingly difficult testability challenges due to device complexity

© Gert Jervan

23



Correct or Defective?

© Gert Jervan

24

Product Technical Trends

	1990	2000	2010
Operating temperature, °C	-55 to 125	-40 to +85	0 to 70
Supply voltage	5v	1.5v	0.6v
Max. power (high perf.)	5	100	170
No. of package types	<10	<60	??
Design support life	>10 yrs.	1-5 yrs.	<1yr.
Production life	>10 yrs.	3-5 yrs.	<3yrs.
Service life	*>20 yrs.*	*5-10 yrs.*	*<5yrs.*

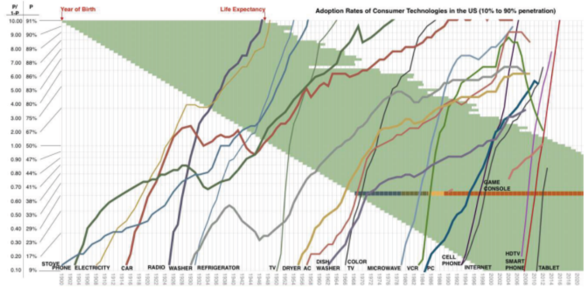
*MRQW-2002, Bernstein

Growing Internet Traffic

Year	Global Internet Traffic
1992	100 GB/Day
1997	100 GB/Hour
2002	100 GB/Sec
2007	2 000 GB/Sec
2012	12 000 GB/Sec
2017	35 000 GB/Sec

Cisco VNI, 2013

Ubiquitous Computing



School of Computer and Communication Sciences | End of the World | 9

James Larus, EPFL

Software complexity is a challenge

Aviation:

- Boeing 747 → 0.4 M LOC
- Boeing 777 → 4 M LOC
- Technology Review 2002

Software:

- Exponential increase in software complexity
- In some areas code size is doubling every 9 months [ST Microelectronics, Medea Workshop, Fall 2003]
- ... > 70% of the development cost for complex systems such as automotive electronics and communication systems are due to software development [A. Sangiovanni-Vincentelli, 1999]

Automotive:

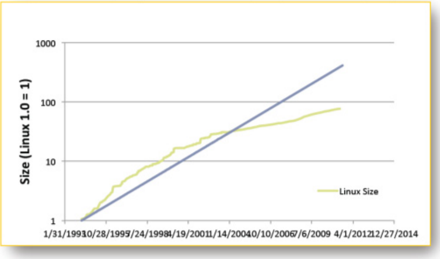
- ✓ 2010 Premium → 100 M LOC
- ✓ 1995 – 2000 → 52%/Year
- ✓ 2001 – 2010 → 35%/Year

Tony Scott, GM CIO

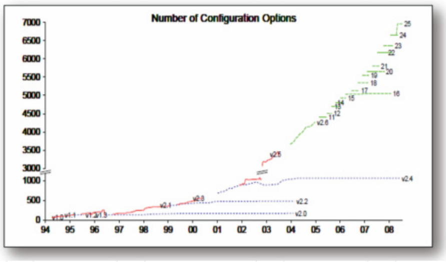
- ✓ 2011 – BMW is the first manufacturer to break the 1Gb barrier

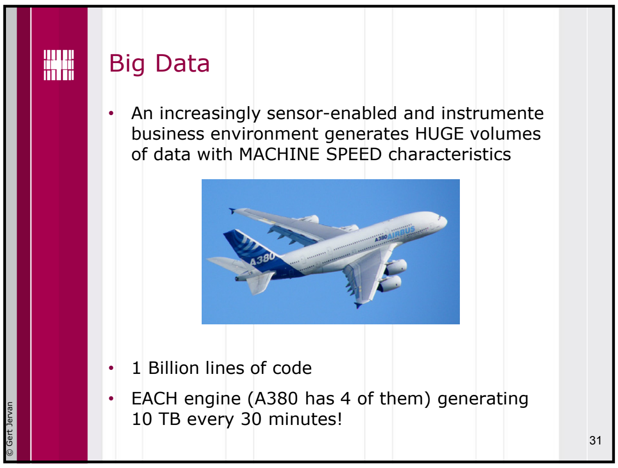
Rob van Ommering, COPA Tutorial, as cited by: Gerrit Müller: Opportunities and challenges in embedded systems, Eindhoven Embedded Systems Institute, 2004

Linux Growth



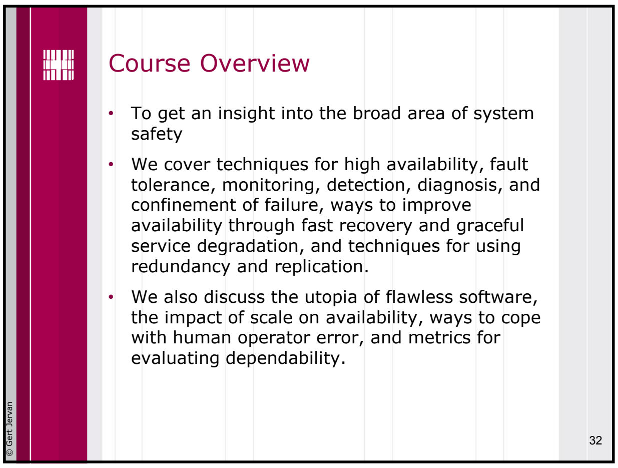
Linux Complexity





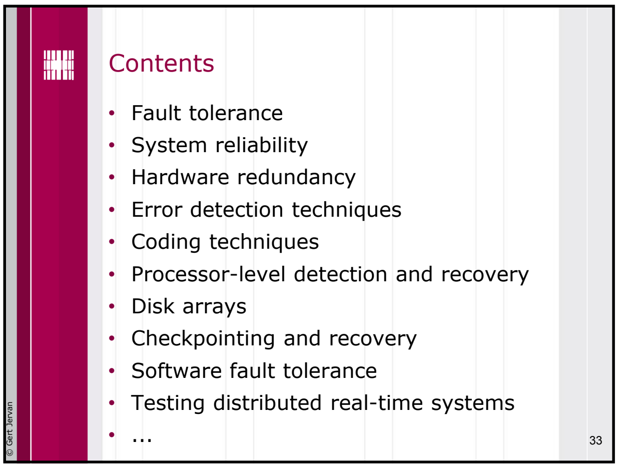
Big Data

- An increasingly sensor-enabled and instrumente business environment generates HUGE volumes of data with MACHINE SPEED characteristics
- 1 Billion lines of code
- EACH engine (A380 has 4 of them) generating 10 TB every 30 minutes!



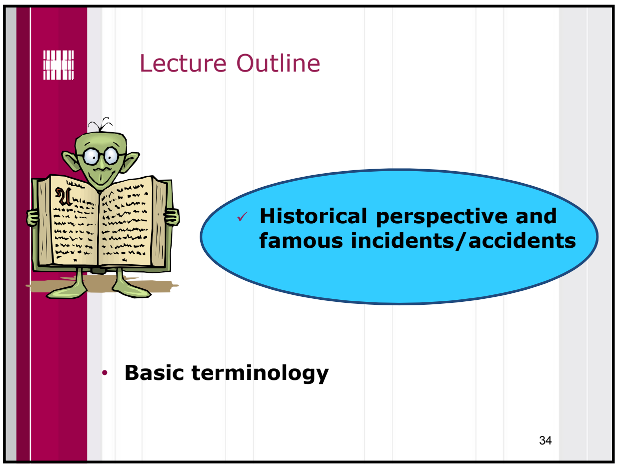
Course Overview

- To get an insight into the broad area of system safety
- We cover techniques for high availability, fault tolerance, monitoring, detection, diagnosis, and confinement of failure, ways to improve availability through fast recovery and graceful service degradation, and techniques for using redundancy and replication.
- We also discuss the utopia of flawless software, the impact of scale on availability, ways to cope with human operator error, and metrics for evaluating dependability.



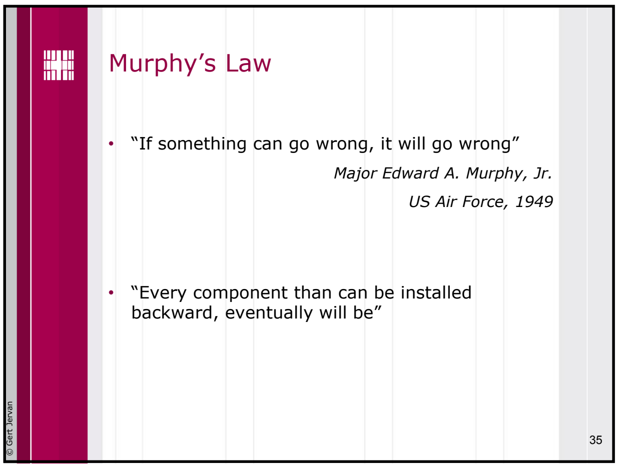
Contents

- Fault tolerance
- System reliability
- Hardware redundancy
- Error detection techniques
- Coding techniques
- Processor-level detection and recovery
- Disk arrays
- Checkpointing and recovery
- Software fault tolerance
- Testing distributed real-time systems
- …



Lecture Outline

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



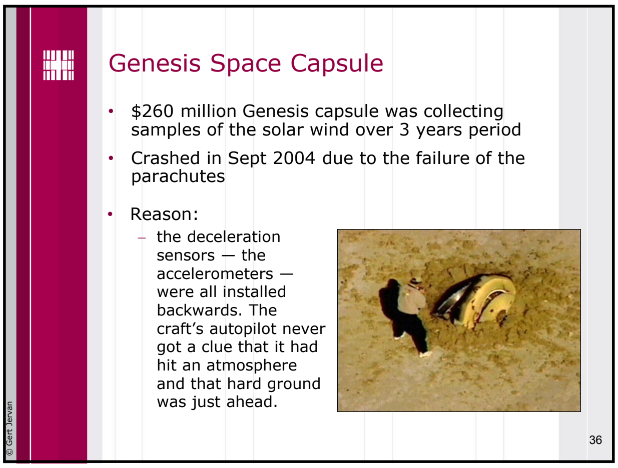
Murphy's Law

- "If something can go wrong, it will go wrong"

  Major Edward A. Murphy, Jr.

  US Air Force, 1949

- "Every component than can be installed backward, eventually will be"



Genesis Space Capsule

- $260 million Genesis capsule was collecting samples of the solar wind over 3 years period
- Crashed in Sept 2004 due to the failure of the parachutes
- Reason:
  - the deceleration sensors — the accelerometers — were all installed backwards. The craft's autopilot never got a clue that it had hit an atmosphere and that hard ground was just ahead.

Mars Orbiter

- One of the Mars Orbiter probes crashed into the planet in 1999.
- It did turn out that engineers who built the Mars Climate Orbiter had provided a data table in "pound-force" rather than newtons, the metric measure of force.
- NASA flight controllers at the Jet Propulsion Laboratory in Pasadena, Calif., had used the faulty table for their navigation calculations during the long coast from Earth to Mars.

Lockheed Martin Titan 4

- In 1998, a LockMart Titan 4 booster carrying a $1 billion LockMart Vortex-class spy satellite pitched sideways and exploded 40 seconds after liftoff from Cape Canaveral, Fla.
- Reason: frayed wiring that apparently had not been inspected. The guidance systems were without power for a fraction of a second.

A Titan 4 rocket explodes shortly after takeoff in August 1998.

Therac-25

- Therac-25:
  - the most serious computer-related accidents to date (at least nonmilitary and admitted)
  - machine for radiation therapy (treating cancer)
  - between June 1985 and January 1987 (at least) six patients received severe overdoses (two died shortly afterward, two might have died but died because of cancer, the other two had permanent disabilities)
  - scanning magnets are used to spread the beam and vary the beam energy
  - dual-mode: electron beams for surface tumors, X-ray for deep tumors

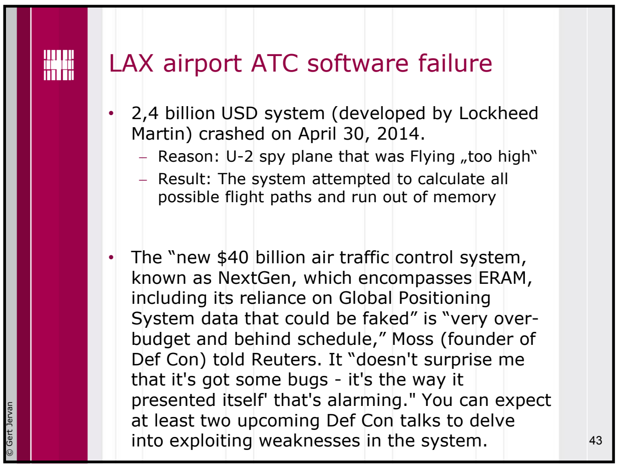
Denver Airport

- Denver International Airport, Colorado: intelligent luggage transportation system with 4000 “Telecars”, 35km rails, controlled by a network of 100 computers with 5000 sensors, 400 radio antennas, and 56 barcode readers. Price: $186 million (BAE Automated Systems).
- Due to SW problems about one year delay which costs $1.1 million per day (1993).
- Abondoned in 2005 to save $1 million per month on maintenance
- Today we have the on-going story with the new Berlin Brandenburg Airport
  - Scheduled to open in 2011, the new estimate is 2014

Boeing 787 Dreamliner

- Program launched in 2003, roll-out in 2007, first delivery in 2011. 114 delivered so far.
- Grounded on January 16, 2013 due to the problems with electrical circuitry
  - Leading to thermal runaway of Li-ion batteries and causing several fires in the battery compartment (several emergency landings, one aircraft (ET) was heavily damaged on ground)
  - Comprehensive review of the 787's critical systems, including the design, manufacture and assembly.
  - Japanese ANA alone lost 1.1 M USD per day (17 aircrafts)

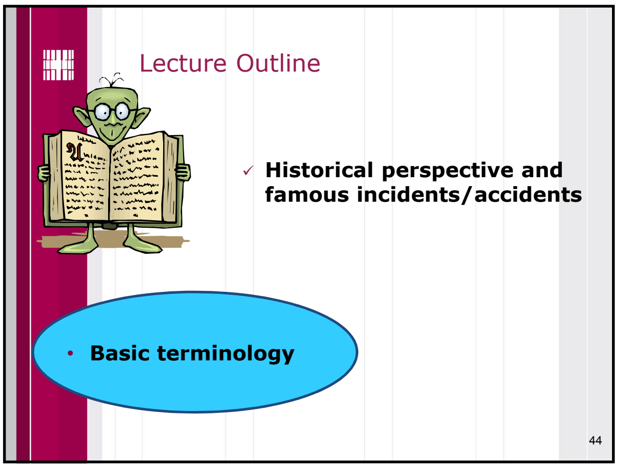
- Grounding lifted on April 26, 2013



LAX airport ATC software failure

- 2,4 billion USD system (developed by Lockheed Martin) crashed on April 30, 2014.
  - Reason: U-2 spy plane that was Flying „too high"
  - Result: The system attempted to calculate all possible flight paths and run out of memory
- The "new $40 billion air traffic control system, known as NextGen, which encompasses ERAM, including its reliance on Global Positioning System data that could be faked" is "very over-budget and behind schedule," Moss (founder of Def Con) told Reuters. It "doesn't surprise me that it's got some bugs - it's the way it presented itself' that's alarming." You can expect at least two upcoming Def Con talks to delve into exploiting weaknesses in the system.

© Gert Jervan

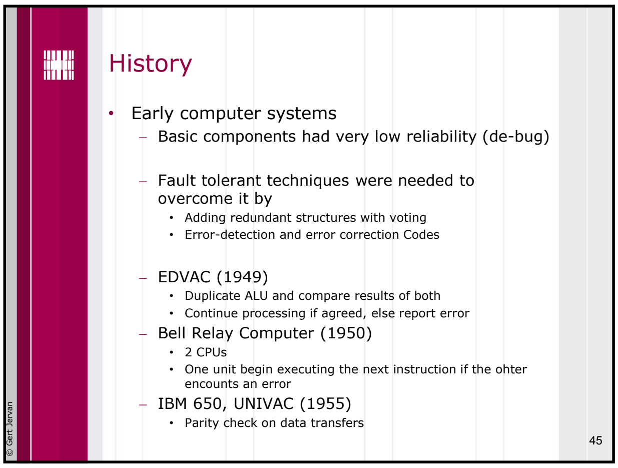


Lecture Outline

✓ **Historical perspective and famous incidents/accidents**

- **Basic terminology**

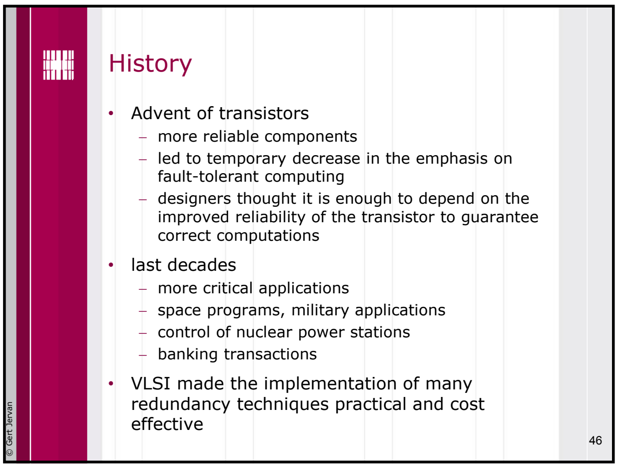
© Gert Jervan



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

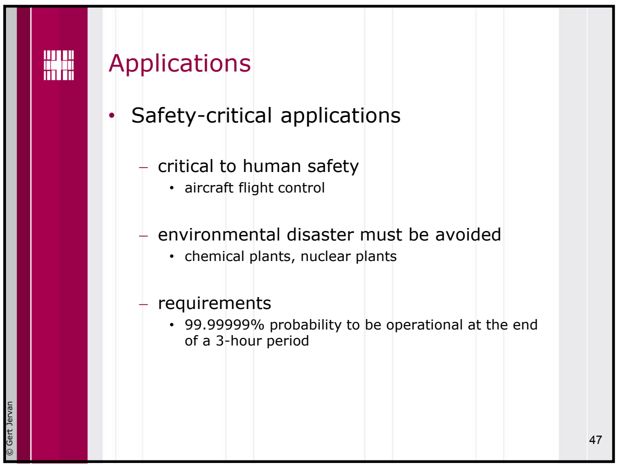
© Gert Jervan



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

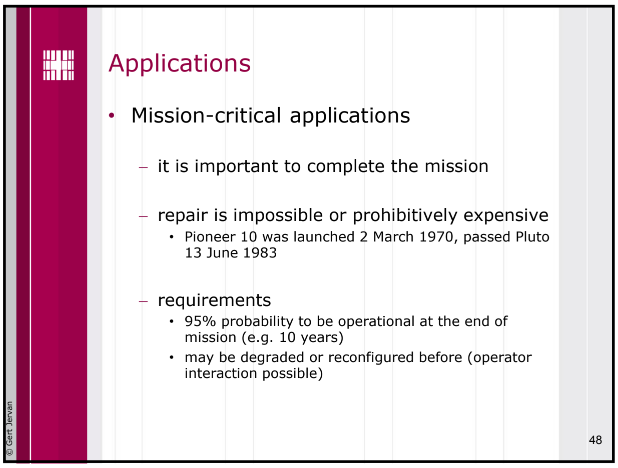
© Gert Jervan



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

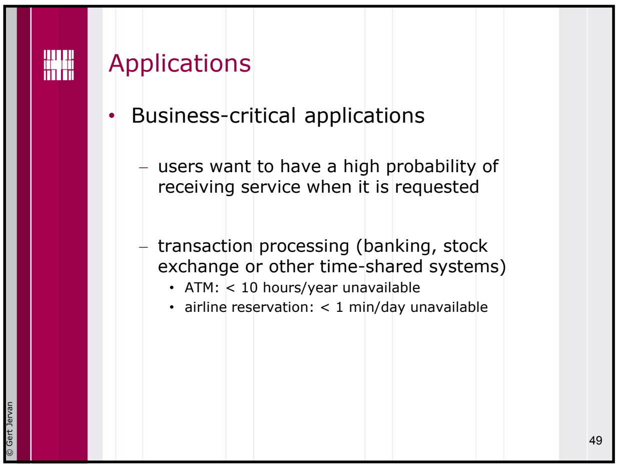
© Gert Jervan



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

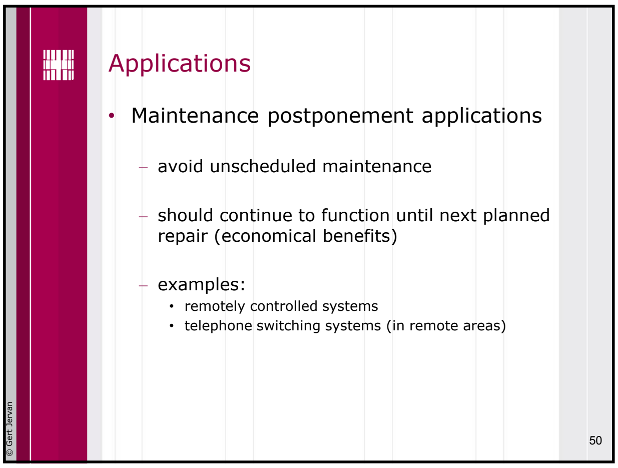


Applications

- 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

49

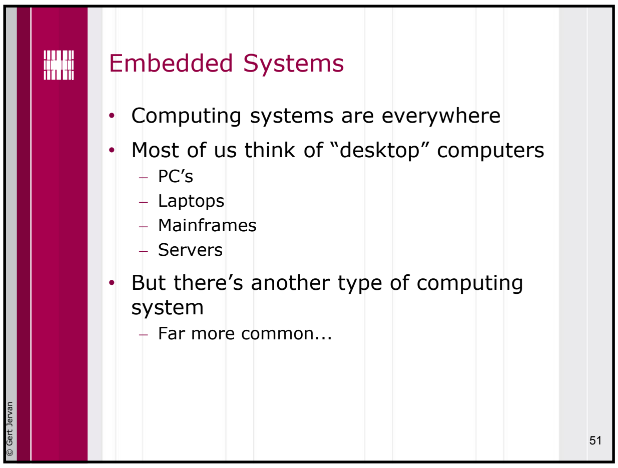


Applications

- 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

50

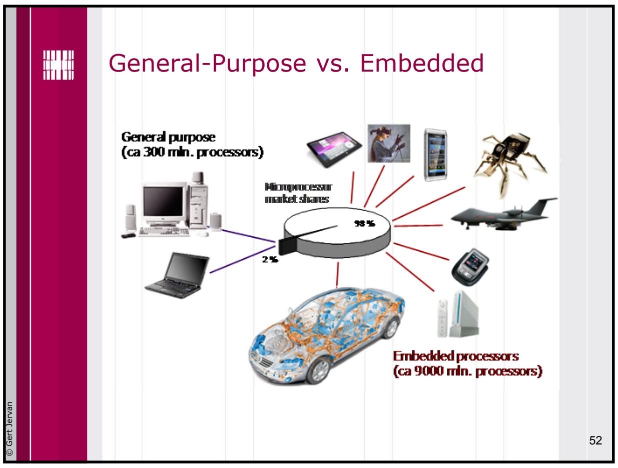


Embedded Systems

- 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

51



General-Purpose vs. Embedded

General purpose (ca 300 mln. processors)

Microprocessor market shares

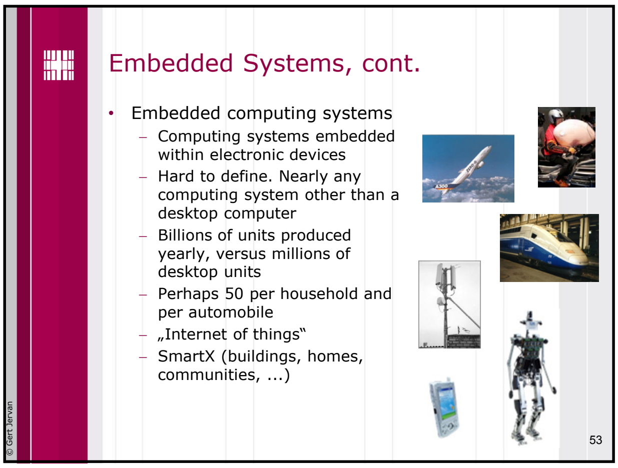
98 %

2 %

Embedded processors (ca 9000 mln. processors)

© Gert Jervan

52

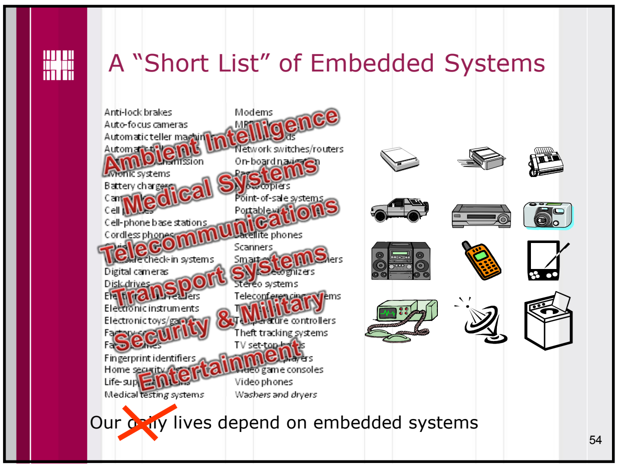


Embedded Systems, cont.

- 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, ...)

© Gert Jervan

53



A "Short List" of Embedded Systems

Anti-lock brakes
Auto-focus cameras
Automatic teller machines
Automatic toll systems
Automatic transmission
Avionic systems
Battery chargers
Camcorders
Cell phones
Cell-phone base stations
Cordless phones
Cruise control
Curbside check-in systems
Digital cameras
Disk drives
Electronic card readers
Electronic instruments
Electronic toys/games
Factory control
Fax machines
Fingerprint identifiers
Home security systems
Life-support systems
Medical testing systems

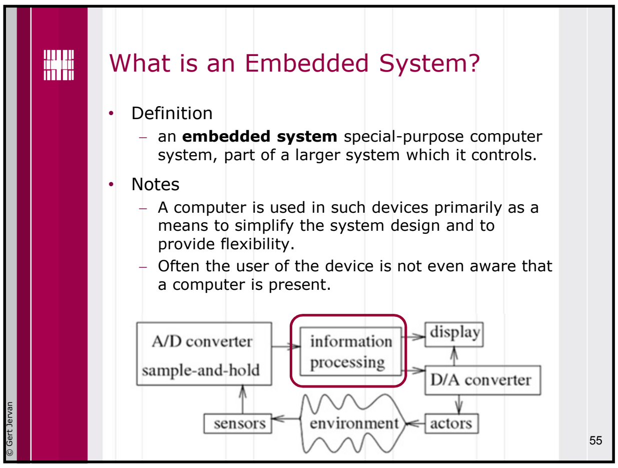
Modems
MPEG decoders
Network cards
Network switches/routers
On-board navigation
Pagers
Photocopiers
Point-of-sale systems
Portable video games
Printers
Satellite phones
Scanners
Smart ovens/dishwashers
Speech recognizers
Stereo systems
Teleconferencing systems
Televisions
Temperature controllers
Theft tracking systems
TV set-top boxes
VCR's, DVD players
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

© Gert Jervan

54

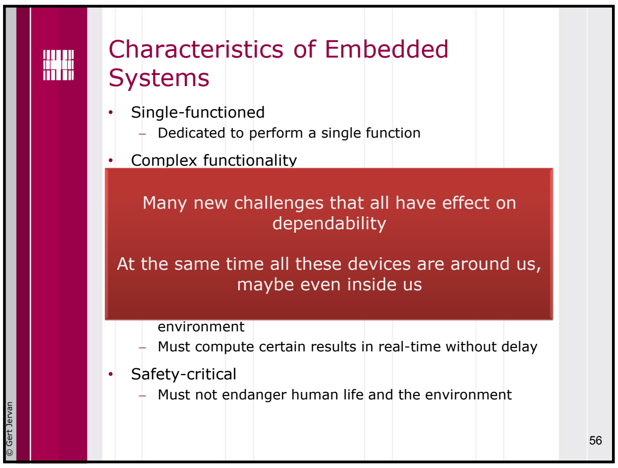


What is an Embedded System?

- 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.

A/D converter
sample-and-hold
information processing
display
D/A converter
sensors
environment
actors

55



Characteristics of Embedded Systems

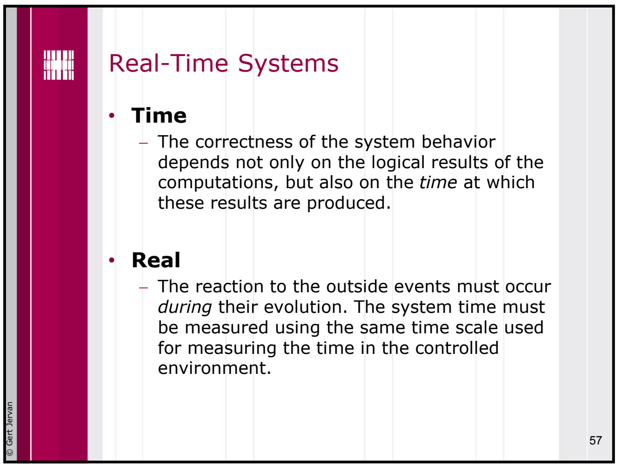
- Single-functioned
  - Dedicated to perform a single function
- Complex functionality

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

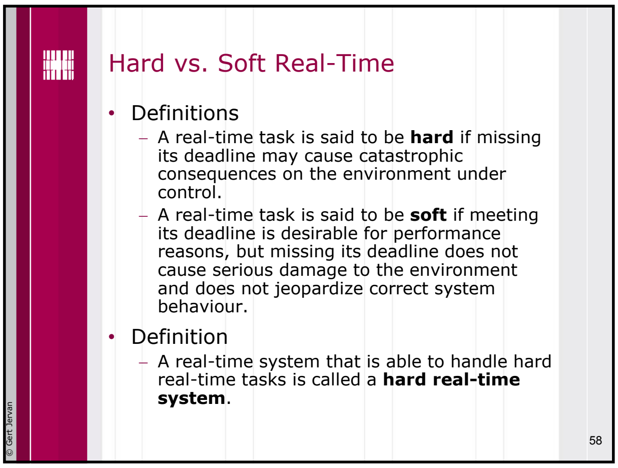
56



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.

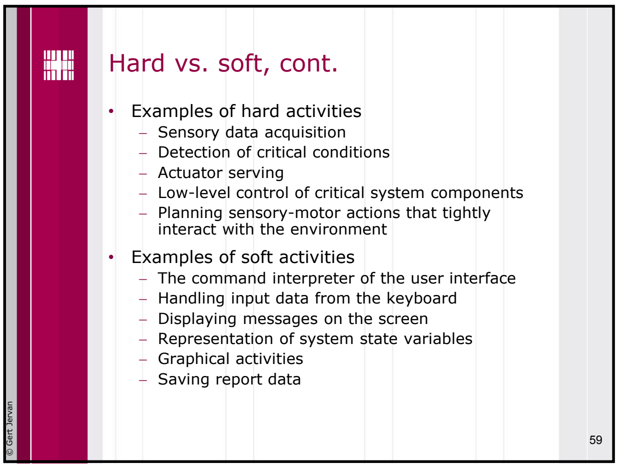
57



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**.

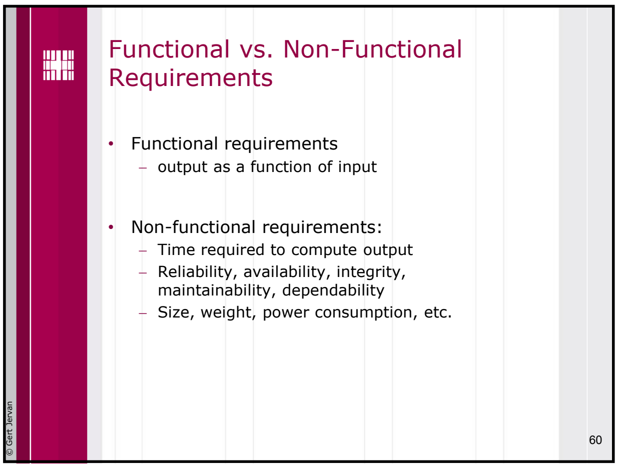
58



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

59



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.

60

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.

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

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

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.

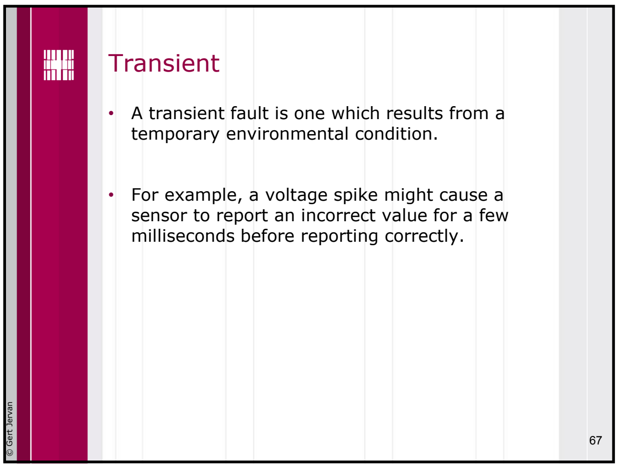
Fault Examples

- Year 2000 bug
- Loose wire
- Aircraft retracting its landing gear (while on ground)

- Effects in time:
  - Permanent
  - Transient
  - Intermittent

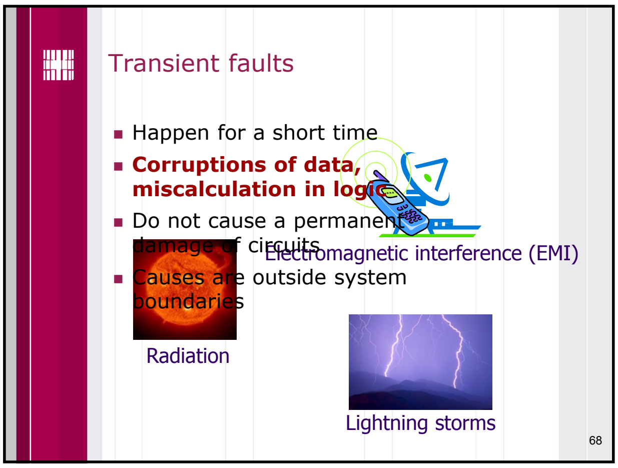
Permanent

- 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).



Transient

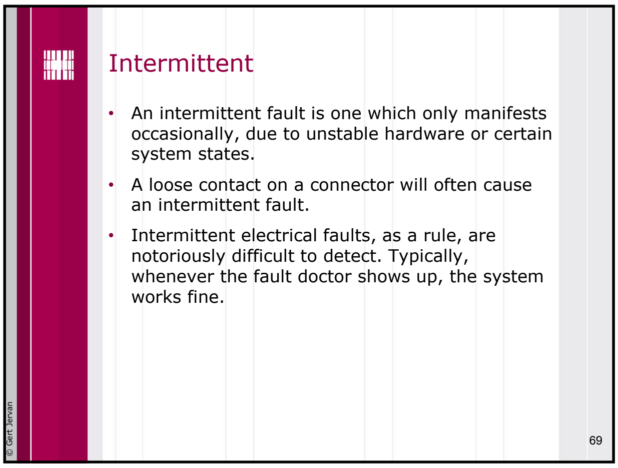
- A transient fault is one which results from a temporary environmental condition.
- For example, a voltage spike might cause a sensor to report an incorrect value for a few milliseconds before reporting correctly.



Transient faults

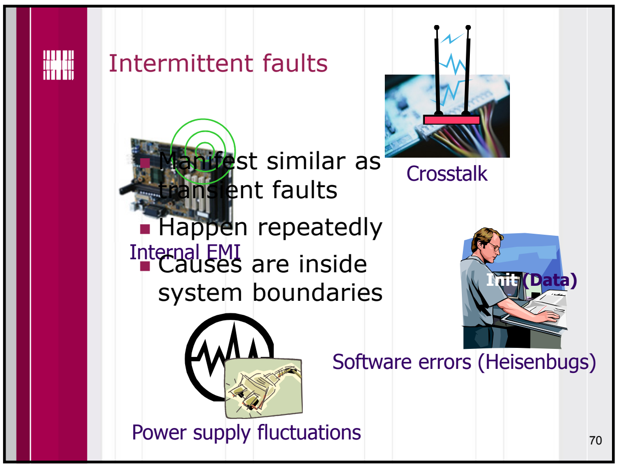
- Happen for a short time
- **Corruptions of data, miscalculation in logic**
- Do not cause a permanent damage of circuits
- Causes are outside system boundaries

Electromagnetic interference (EMI)
Radiation
Lightning storms



Intermittent

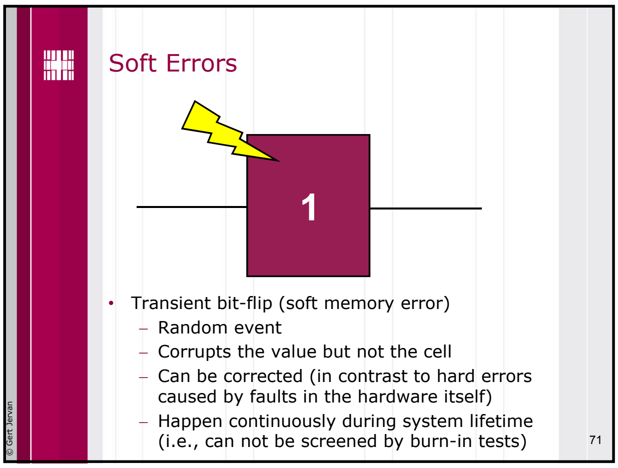
- An intermittent fault is one which only manifests occasionally, due to unstable hardware or certain system states.
- A loose contact on a connector will often cause an intermittent fault.
- Intermittent electrical faults, as a rule, are notoriously difficult to detect. Typically, whenever the fault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



Intermittent faults

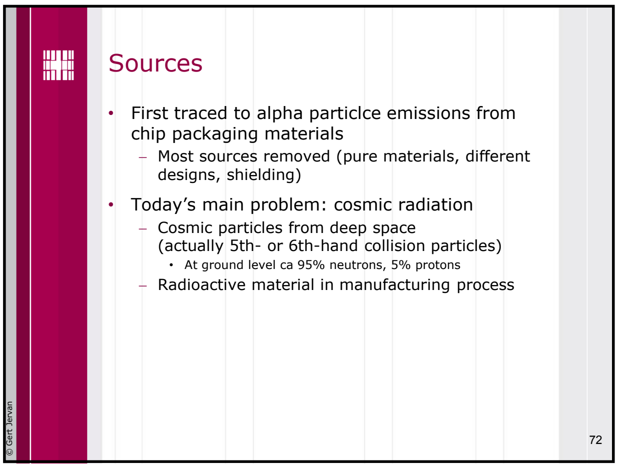
- Manifest similar as transient faults
- Happen repeatedly
- Causes are inside system boundaries

Crosstalk
Internal EMI
Init (Data)
Software errors (Heisenbugs)
Power supply fluctuations



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)



Sources

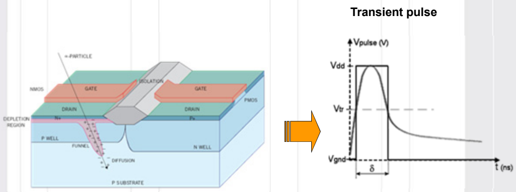
- First traced to alpha particlce emissions from chip packaging materials
  - Most sources removed (pure materials, different designs, shielding)
- Today's main problem: cosmic radiation
  - Cosmic particles from deep space (actually 5th- or 6th-hand collision particles)
    - At ground level ca 95%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

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

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

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

Current Situation

- Soft errors induced the highest failure rate of all other reliability mechanisms combined

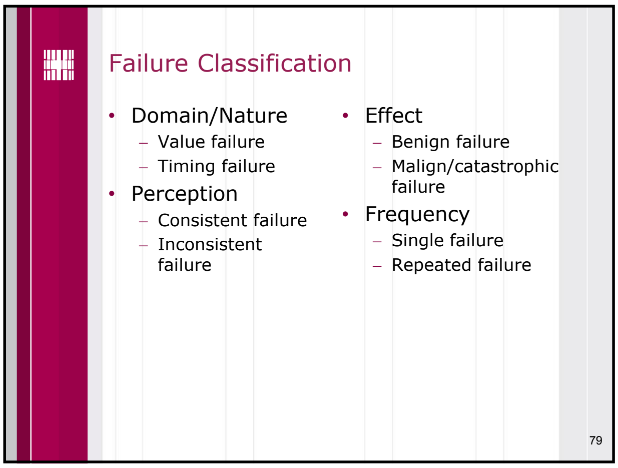
Rober Baumann, TI

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

Example

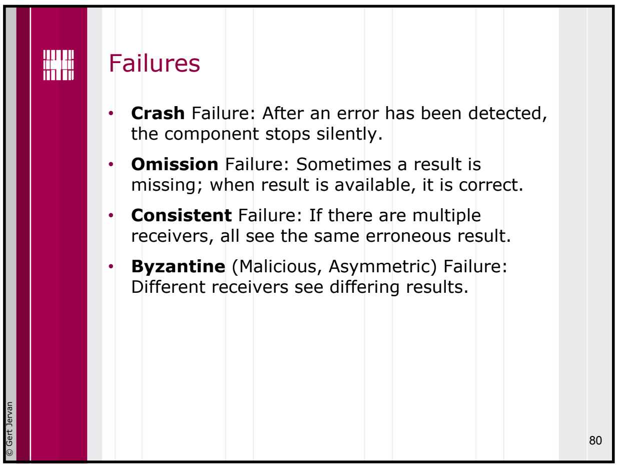
- It is practically impossible to build a perfect system
  - Suppose a component has the reliability of 99.99%
  - A system consisting of 100 non-redundant components will have the reliability 99.01%
  - A system consisting of 10 000 non-redundant components will have the reliability 36.79%
- It is hard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



Failure Classification

- Domain/Nature
  - Value failure
  - Timing failure
- Perception
  - Consistent failure
  - Inconsistent failure
- Effect
  - Benign failure
  - Malign/catastrophic failure
- Frequency
  - Single failure
  - Repeated failure

79

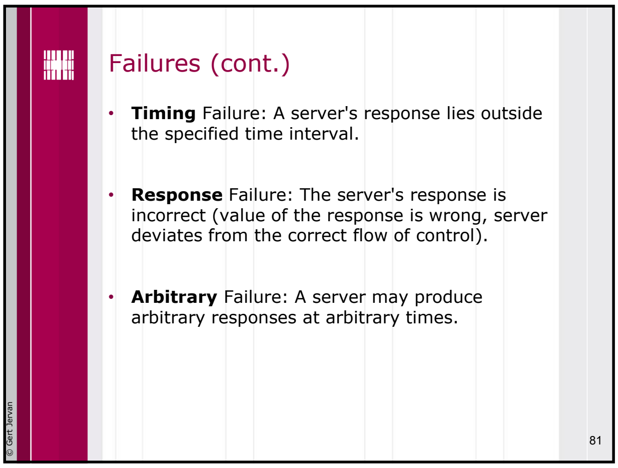


Failures

- **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

80

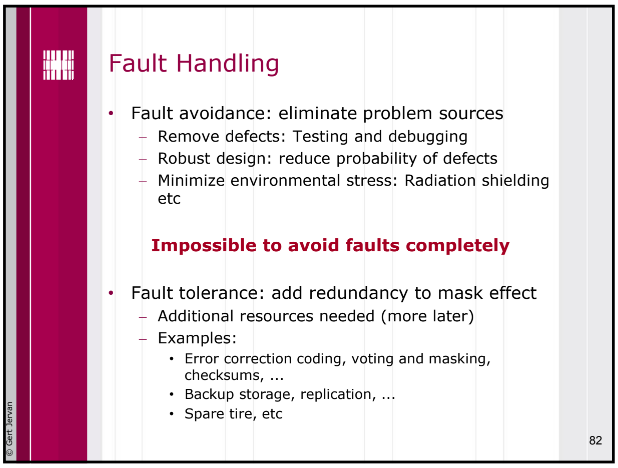


Failures (cont.)

- **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

81



Fault Handling

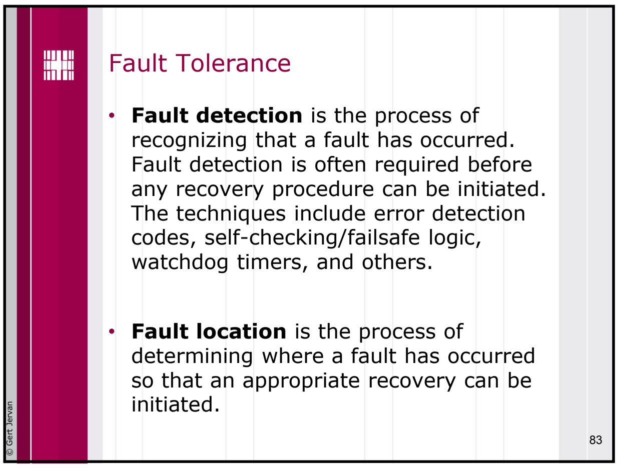
- Fault avoidance: eliminate problem sources
  - Remove defects: Testing and debugging
  - Robust design: reduce probability of defects
  - Minimize environmental stress: Radiation shielding etc

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

© Gert Jervan

82

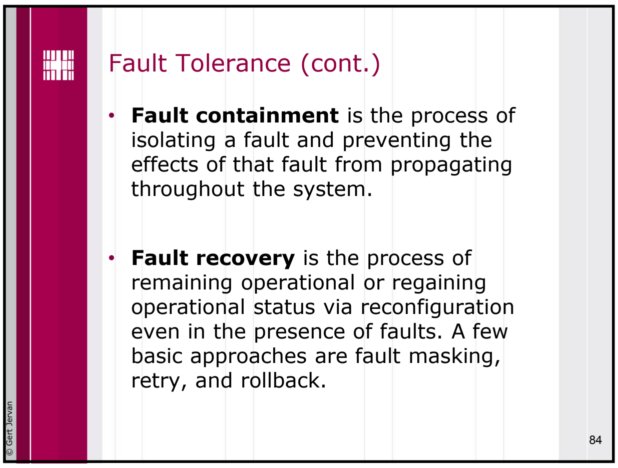


Fault Tolerance

- **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.

© Gert Jervan

83

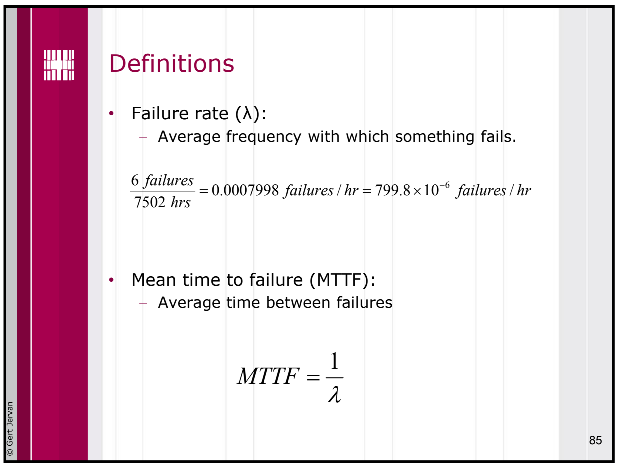


Fault Tolerance (cont.)

- **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.

© Gert Jervan

84



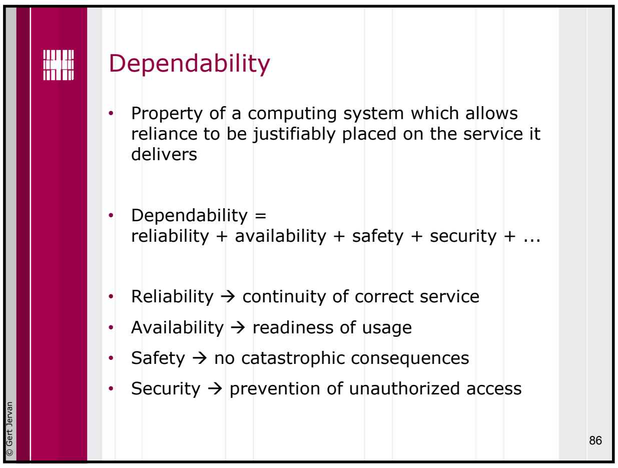
Definitions

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

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

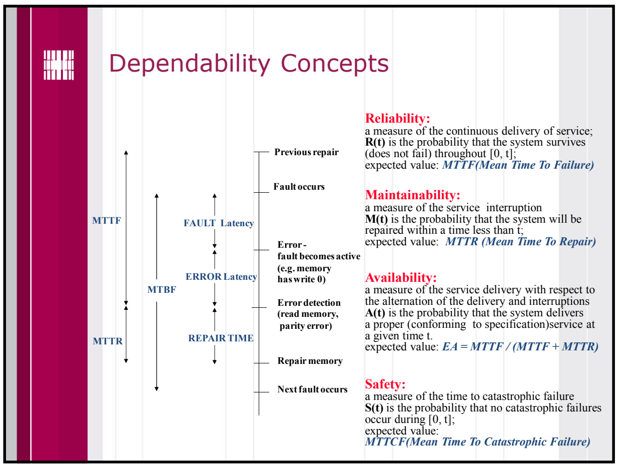
- Mean time to failure (MTTF):
  - Average time between failures

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



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



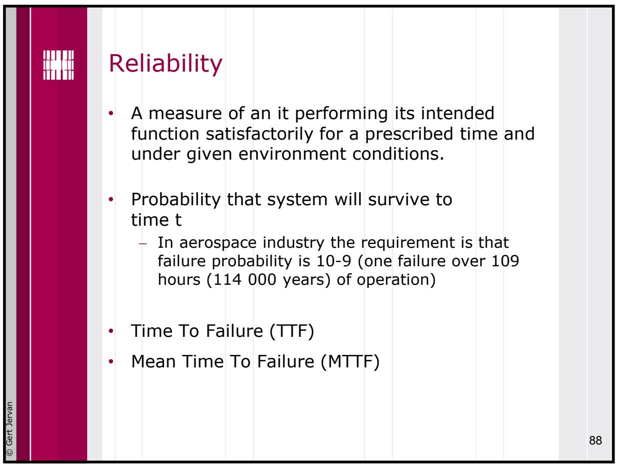
Dependability Concepts

Reliability:
a measure of the continuous delivery of service;
R(t) is the probability that the system survives (does not fail) throughout [0, t];
expected value: *MTTF(Mean Time To Failure)*

Maintainability:
a measure of the service interruption
M(t) is the probability that the system will be repaired within a time less than t;
expected value: *MTTR (Mean Time To Repair)*

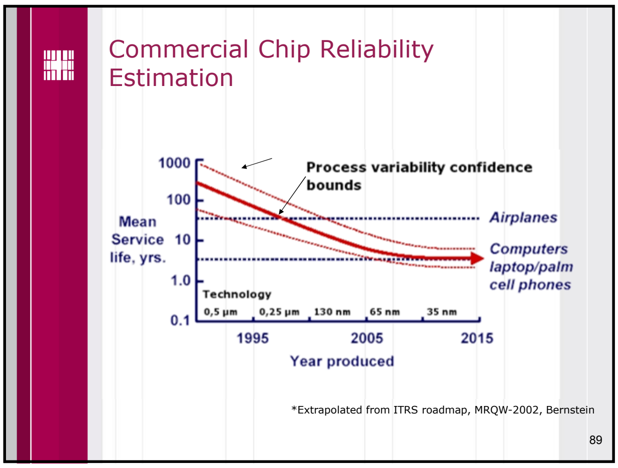
Availability:
a measure of the service delivery with respect to the alternation of the delivery and interruptions
A(t) is the probability that the system delivers a proper (conforming to specification)service at a given time t.
expected value: *EA = MTTF / (MTTF + MTTR)*

Safety:
a measure of the time to catastrophic failure
S(t) is the probability that no catastrophic failures occur during [0, t];
expected value:
MTTCF(Mean Time To Catastrophic Failure)



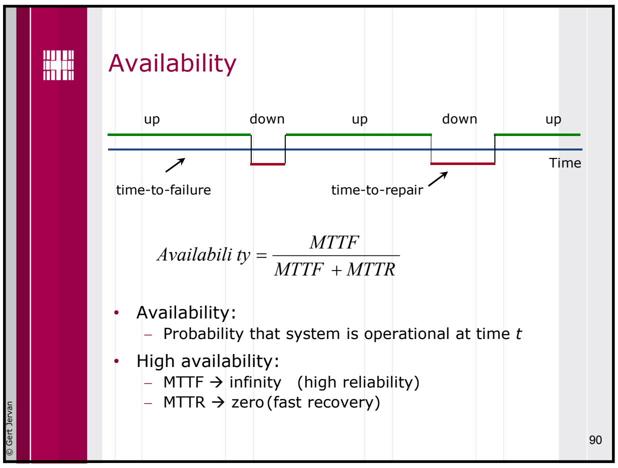
Reliability

- A measure of an it performing its intended function satisfactorily for a prescribed time and under given environment conditions.
- Probability that system will survive to time t
  - In aerospace industry the requirement is that failure probability is 10-9 (one failure over 109 hours (114 000 years) of operation)
- Time To Failure (TTF)
- Mean Time To Failure (MTTF)



Commercial Chip Reliability Estimation

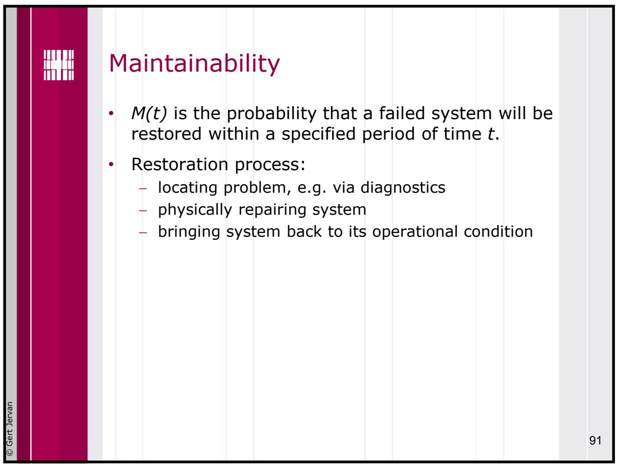
*Extrapolated from ITRS roadmap, MRQW-2002, Bernstein



Availability

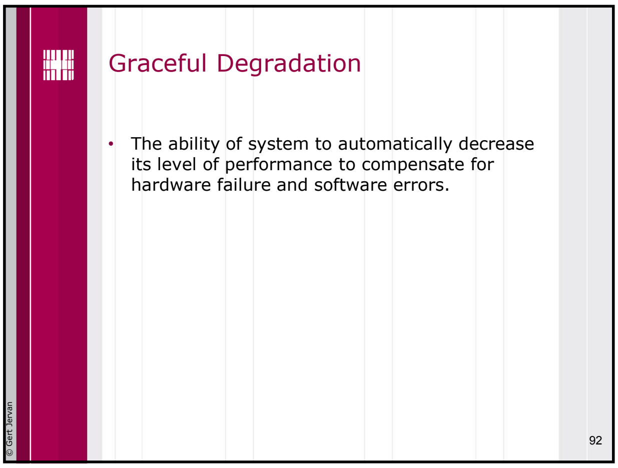
$$Availability = \frac{MTTF}{MTTF + MTTR}$$

- Availability:
  - Probability that system is operational at time *t*
- High availability:
  - MTTF → infinity (high reliability)
  - MTTR → zero (fast recovery)



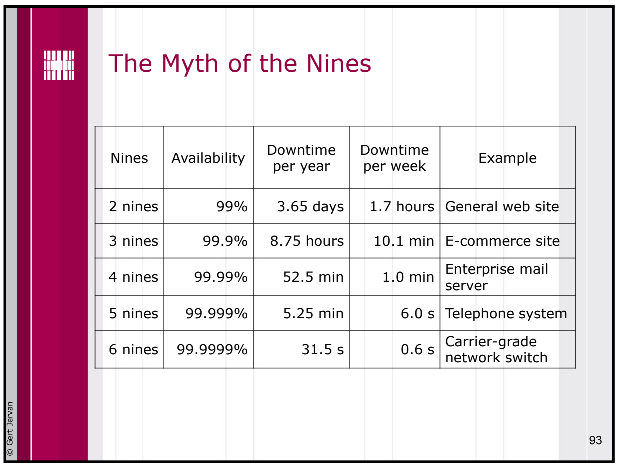
Maintainability

- *M(t)* is the probability that a failed system will be restored within a specified period of time *t*.
- Restoration process:
  - locating problem, e.g. via diagnostics
  - physically repairing system
  - bringing system back to its operational condition



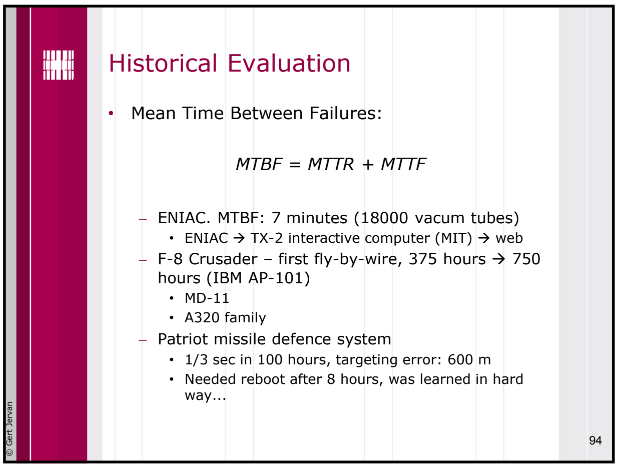
Graceful Degradation

- The ability of system to automatically decrease its level of performance to compensate for hardware failure and software errors.



The Myth of the Nines

Nines	Availability	Downtime per year	Downtime per week	Example
2 nines	99%	3.65 days	1.7 hours	General web site
3 nines	99.9%	8.75 hours	10.1 min	E-commerce site
4 nines	99.99%	52.5 min	1.0 min	Enterprise mail server
5 nines	99.999%	5.25 min	6.0 s	Telephone system
6 nines	99.9999%	31.5 s	0.6 s	Carrier-grade network switch

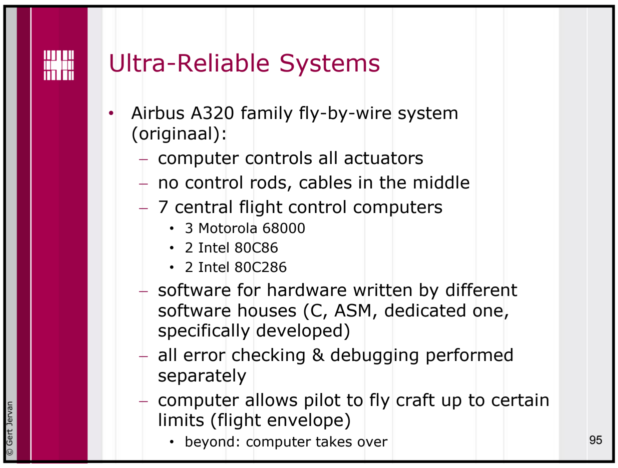


Historical Evaluation

- Mean Time Between Failures:

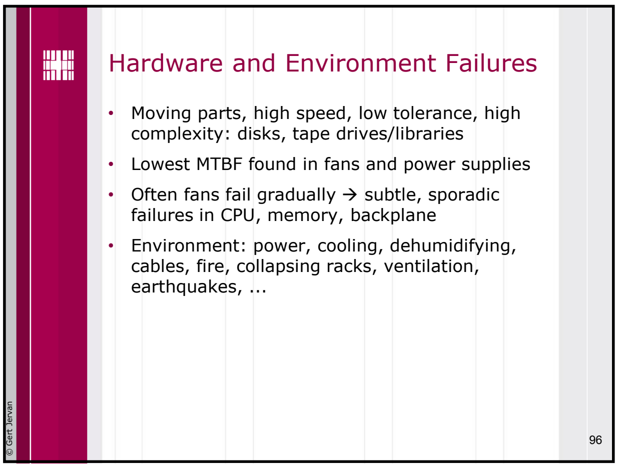
$$MTBF = MTTR + MTTF$$

  - ENIAC. MTBF: 7 minutes (18000 vacum tubes)
    - ENIAC → TX-2 interactive computer (MIT) → web
  - F-8 Crusader – first fly-by-wire, 375 hours → 750 hours (IBM AP-101)
    - MD-11
    - A320 family
  - Patriot missile defence system
    - 1/3 sec in 100 hours, targeting error: 600 m
    - Needed reboot after 8 hours, was learned in hard way…



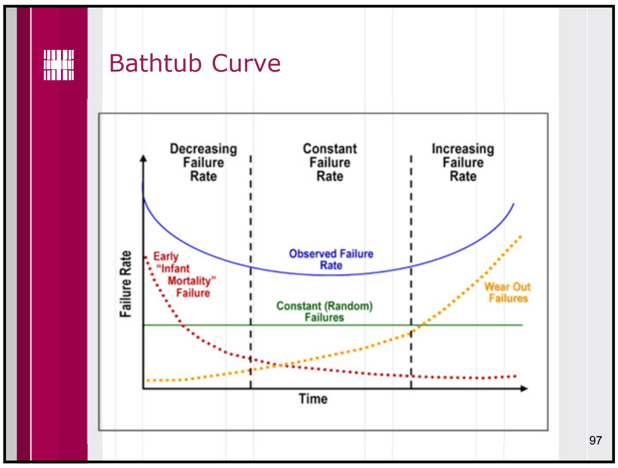
Ultra-Reliable Systems

- Airbus A320 family fly-by-wire system (originaal):
  - computer controls all actuators
  - no control rods, cables in the middle
  - 7 central flight control computers
    - 3 Motorola 68000
    - 2 Intel 80C86
    - 2 Intel 80C286
  - software for hardware written by different software houses (C, ASM, dedicated one, specifically developed)
  - all error checking & debugging performed separately
  - computer allows pilot to fly craft up to certain limits (flight envelope)
    - beyond: computer takes over



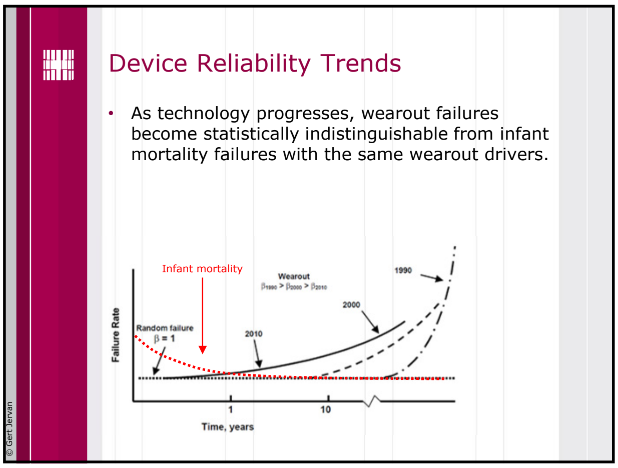
Hardware and Environment Failures

- Moving parts, high speed, low tolerance, high complexity: disks, tape drives/libraries
- Lowest MTBF found in fans and power supplies
- Often fans fail gradually → subtle, sporadic failures in CPU, memory, backplane
- Environment: power, cooling, dehumidifying, cables, fire, collapsing racks, ventilation, earthquakes, …



Bathtub Curve

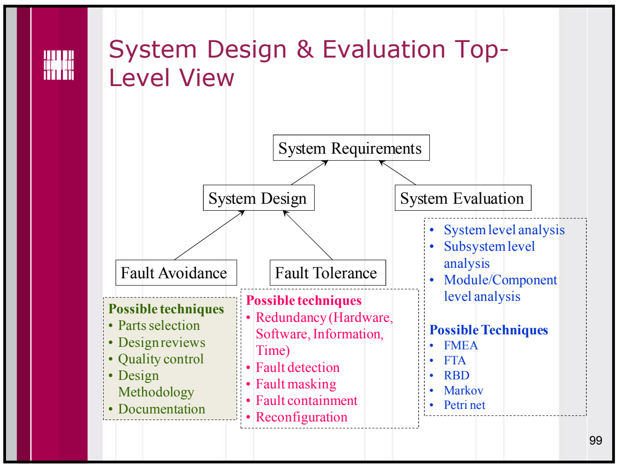
97



Device Reliability Trends

- As technology progresses, wearout failures become statistically indistinguishable from infant mortality failures with the same wearout drivers.

© Gert Jervan



System Design & Evaluation Top-Level View

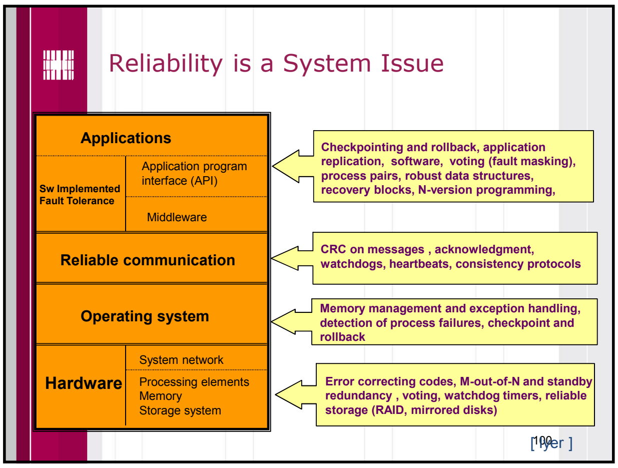
Possible techniques
- Parts selection
- Design reviews
- Quality control
- Design Methodology
- Documentation

Possible techniques
- Redundancy (Hardware, Software, Information, Time)
- Fault detection
- Fault masking
- Fault containment
- Reconfiguration

- System level analysis
- Subsystem level analysis
- Module/Component level analysis

Possible Techniques
- FMEA
- FTA
- RBD
- Markov
- Petri net

99



Reliability is a System Issue

Checkpointing and rollback, application replication, software, voting (fault masking), process pairs, robust data structures, recovery blocks, N-version programming,

CRC on messages , acknowledgment, watchdogs, heartbeats, consistency protocols

Memory management and exception handling, detection of process failures, checkpoint and rollback

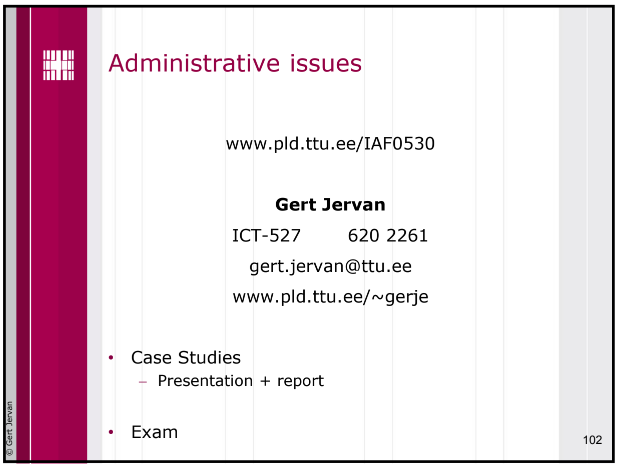
Error correcting codes, M-out-of-N and standby redundancy , voting, watchdog timers, reliable storage (RAID, mirrored disks)

[Iyer]

100

Questions?

Gert Jervan



Administrative issues

www.pld.ttu.ee/IAF0530

Gert Jervan
ICT-527 620 2261
gert.jervan@ttu.ee
www.pld.ttu.ee/~gerje

- Case Studies
  - Presentation + report
- Exam

© Gert Jervan

102