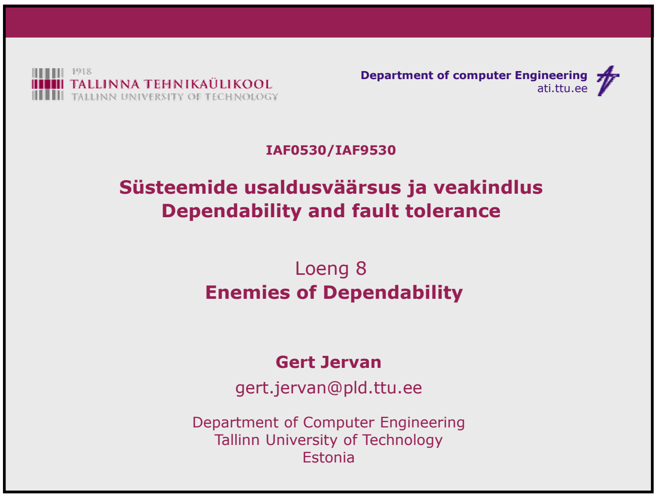


TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Department of computer Engineering
ati.ttu.ee

IAF0530/IAF9530

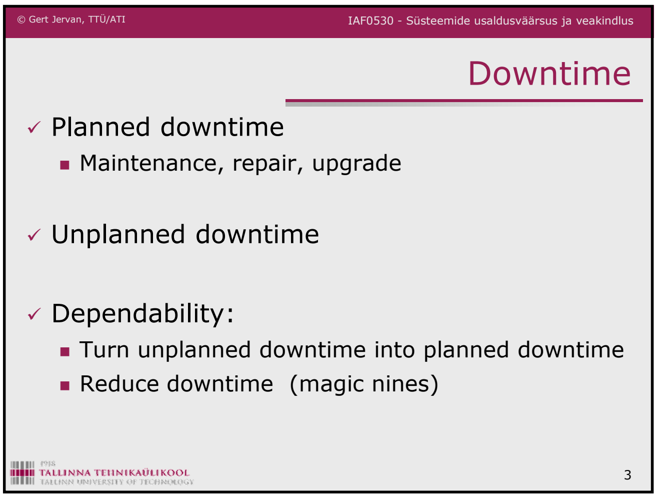
Süsteemide usaldusväärsus ja veakindlus
Dependability and fault tolerance

Loeng 8

Enemies of Dependability

Gert Jervan
gert.jervan@pld.ttu.ee

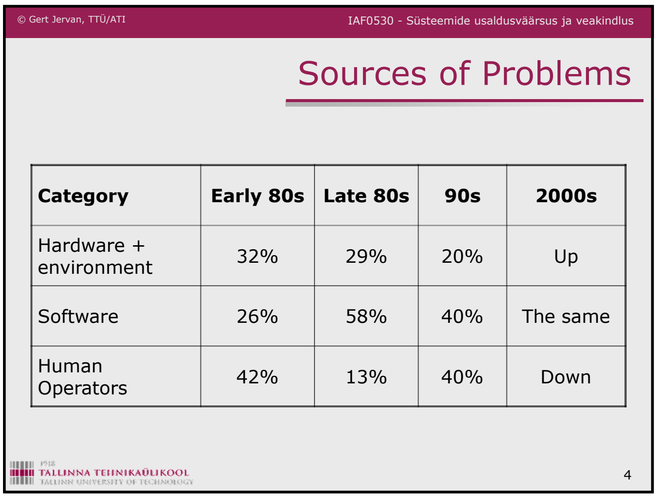
Department of Computer Engineering
Tallinn University of Technology
Estonia



© Gert Jervan, TTÜ/ATI — IAF0530 - Süsteemide usaldusväärsus ja veakindlus

Downtime

- ✓ Planned downtime
  - Maintenance, repair, upgrade
- ✓ Unplanned downtime
- ✓ Dependability:
  - Turn unplanned downtime into planned downtime
  - Reduce downtime (magic nines)



© Gert Jervan, TTÜ/ATI — IAF0530 - Süsteemide usaldusväärsus ja veakindlus

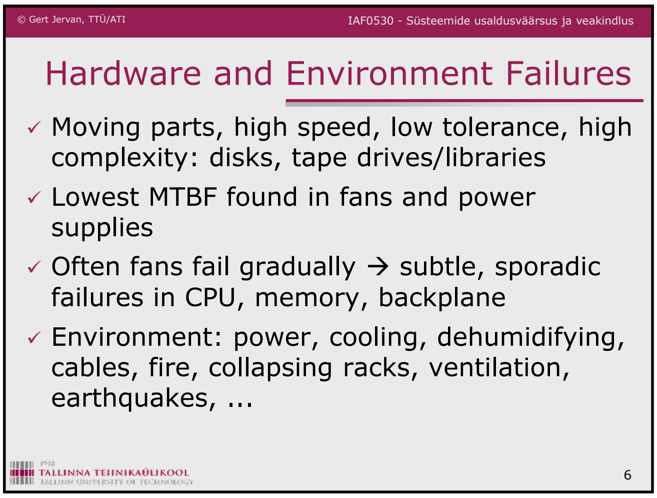
Sources of Problems

Category	Early 80s	Late 80s	90s	2000s
Hardware + environment	32%	29%	20%	Up
Software	26%	58%	40%	The same
Human Operators	42%	13%	40%	Down

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Department of computer Engineering
ati.ttu.ee

Hardware

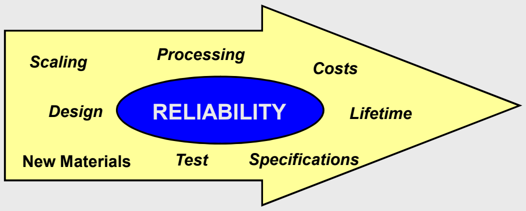


© Gert Jervan, TTÜ/ATI — IAF0530 - Süsteemide usaldusväärsus ja veakindlus

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, …

Hardware Reliability Challenges

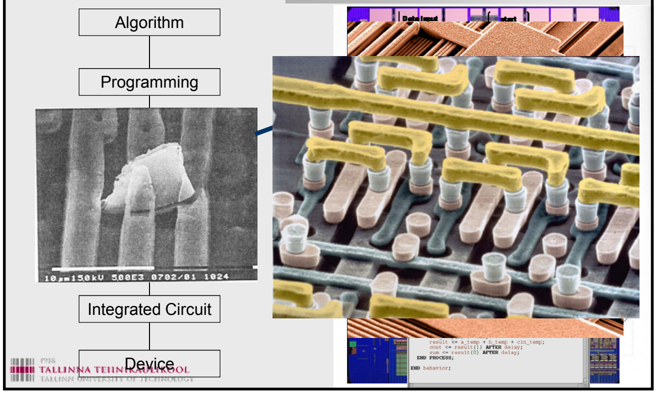

Scaling
Processing
Costs
Design
RELIABILITY
Lifetime
New Materials
Test
Specifications

Reliability Dependencies and Impact to Cost

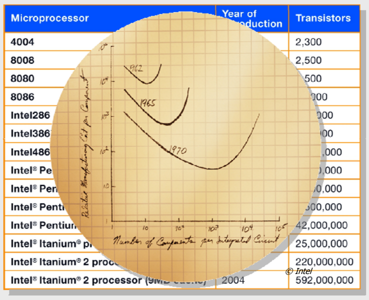
Hardware - Background

- ✓ 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

A Few Layers of Computer Systems


Algorithm
Programming
Integrated Circuit
Device

Moore's Law



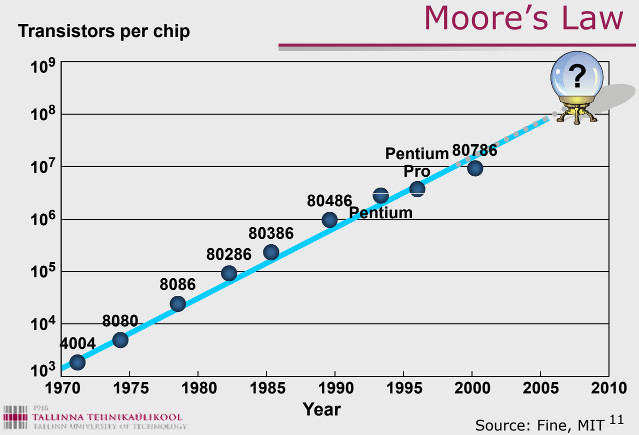
Microprocessor	Year of production	Transistors
4004	[illegible]	2,300
8008	[illegible]	2,500
8080	[illegible]	[illegible]500
8086	[illegible]	[illegible]000
Intel286	[illegible]	[illegible]000
Intel386	[illegible]	[illegible]00
Intel486	[illegible]	[illegible],000
Intel® Pe[illegible]	[illegible]	[illegible],000
Intel® Pen[illegible]	[illegible]	[illegible]0,000
Intel® Penti[illegible]	[illegible]	[illegible]00,000
Intel® Pentiun[illegible]	[illegible]	42,000,000
Intel® Itanium® pr[illegible]	[illegible]	25,000,000
Intel® Itanium® 2 proc[illegible]	[illegible]	220,000,000
Intel® Itanium® 2 processor (9M[illegible])	2004	592,000,000

1962
1965
1970
Relative Manufacturing Cost per Component
Number of Components per Integrated Circuit
1
10
10²
10³
10⁴
10⁵

© Intel

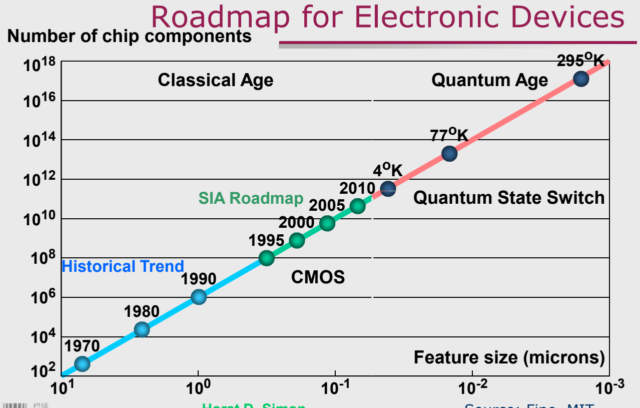
"...(T)he first microprocessor only had 22 hundred transistors. We are looking at something a million times that complex in the next generations—a billion transistors. What that gives us in the way of flexibility to design products is phenomenal."
Gordon E. Moore, April 19, 1965

Moore's Law



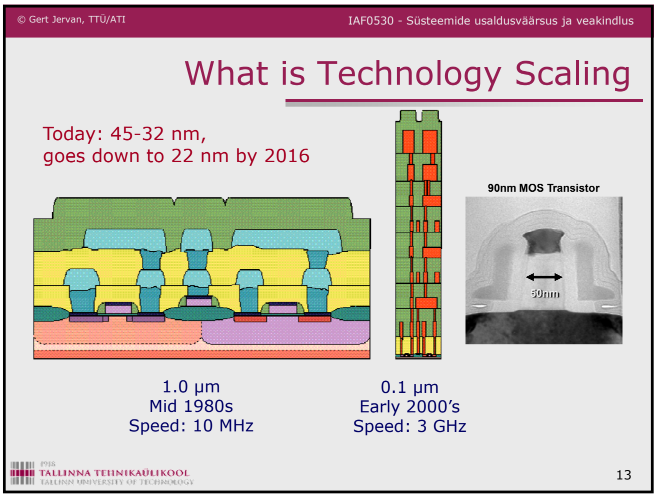
Source: Fine, MIT

Roadmap for Electronic Devices



Horst D. Simon
LAWRENCE BERKELEY NATIONAL LABORATORY

Source: Fine, MIT

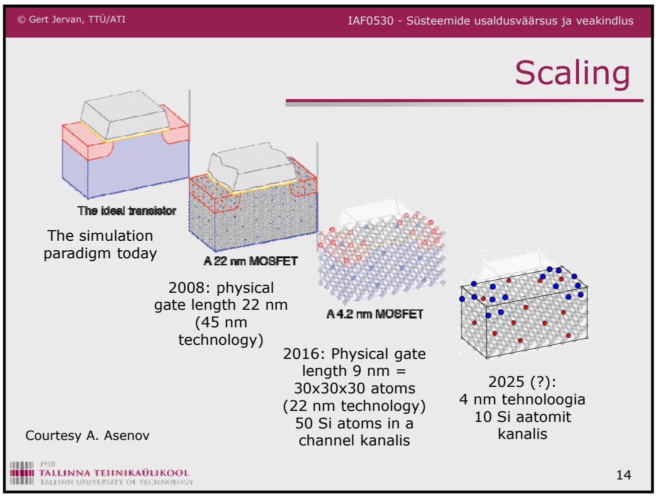


What is Technology Scaling

Today: 45-32 nm,
goes down to 22 nm by 2016

1.0 µm
Mid 1980s
Speed: 10 MHz

0.1 µm
Early 2000's
Speed: 3 GHz



Scaling

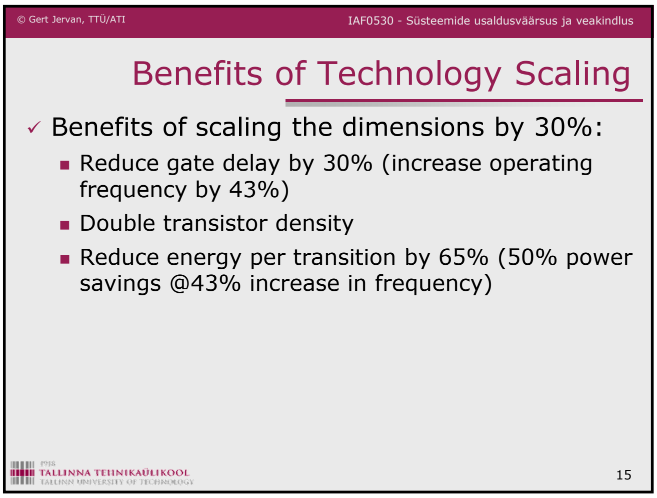
The simulation paradigm today

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 kanalis

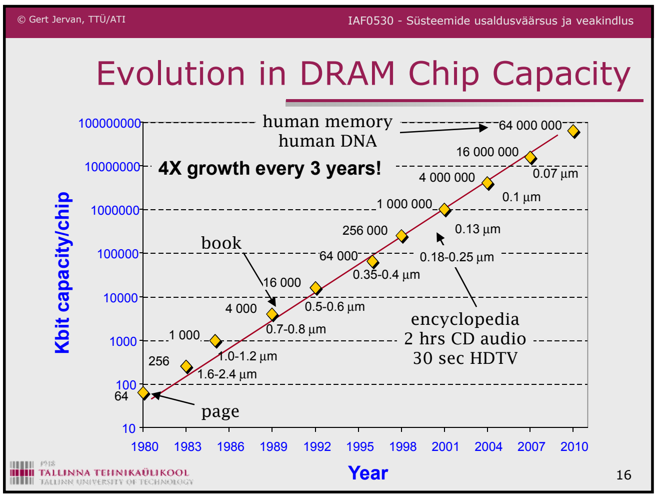
2025 (?): 4 nm tehnoloogia 10 Si aatomit kanalis

Courtesy A. Asenov

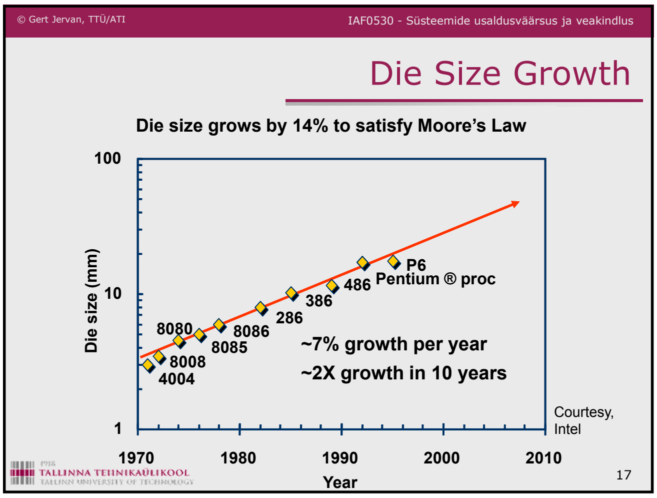


Benefits of Technology Scaling

✓ Benefits of scaling the dimensions by 30%:
  - Reduce gate delay by 30% (increase operating frequency by 43%)
  - Double transistor density
  - Reduce energy per transition by 65% (50% power savings @43% increase in frequency)



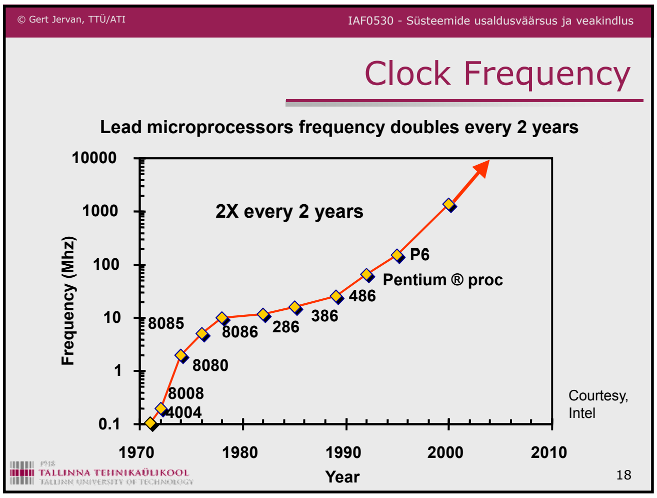
Evolution in DRAM Chip Capacity



Die Size Growth

Die size grows by 14% to satisfy Moore's Law

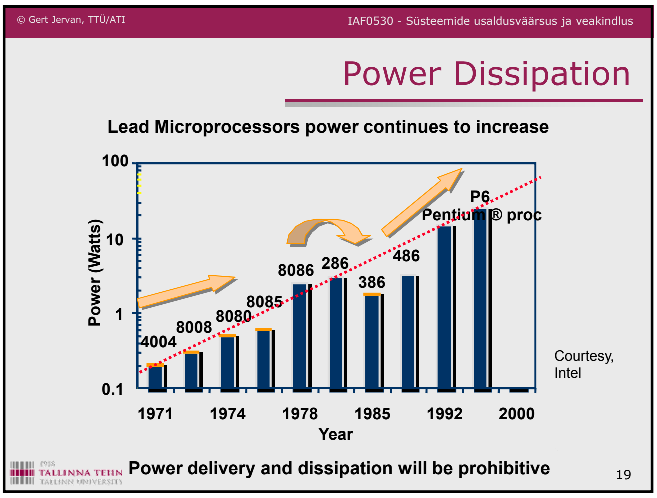
Courtesy, Intel



Clock Frequency

Lead microprocessors frequency doubles every 2 years

Courtesy, Intel

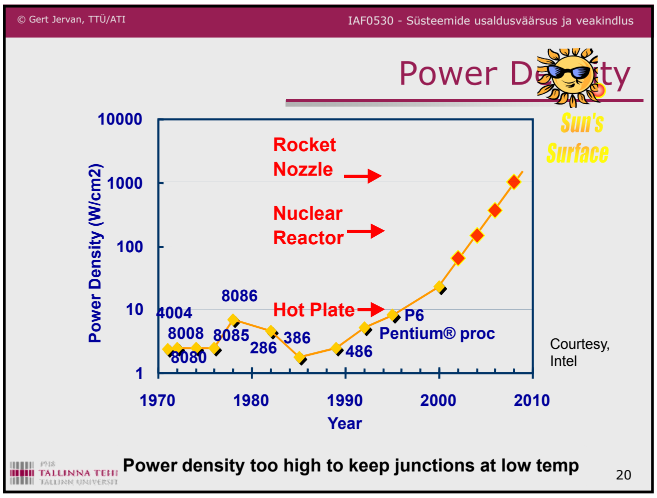


Power Dissipation

Lead Microprocessors power continues to increase

Courtesy, Intel

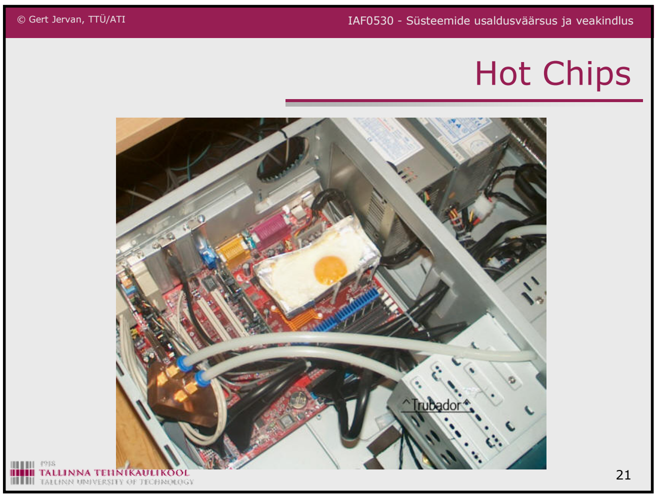
Power delivery and dissipation will be prohibitive



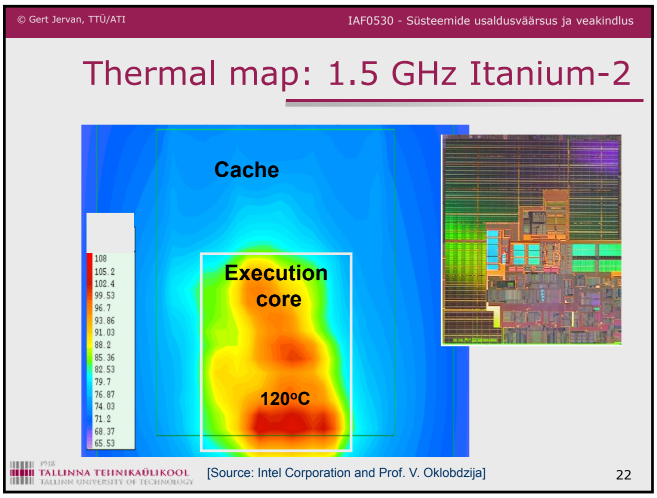
Power Density

Courtesy, Intel

Power density too high to keep junctions at low temp

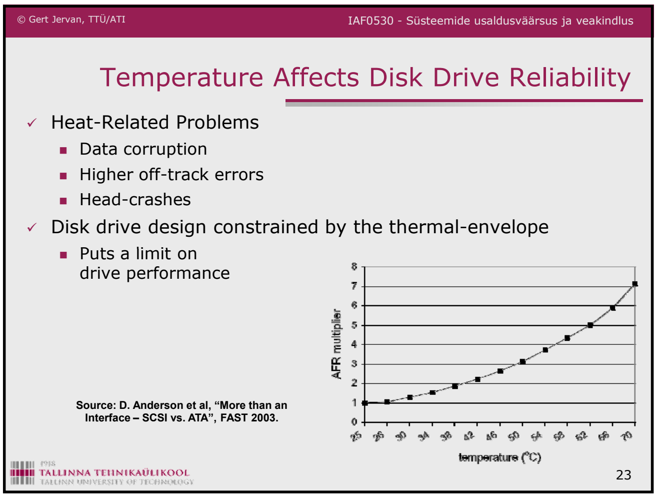


Hot Chips



Thermal map: 1.5 GHz Itanium-2

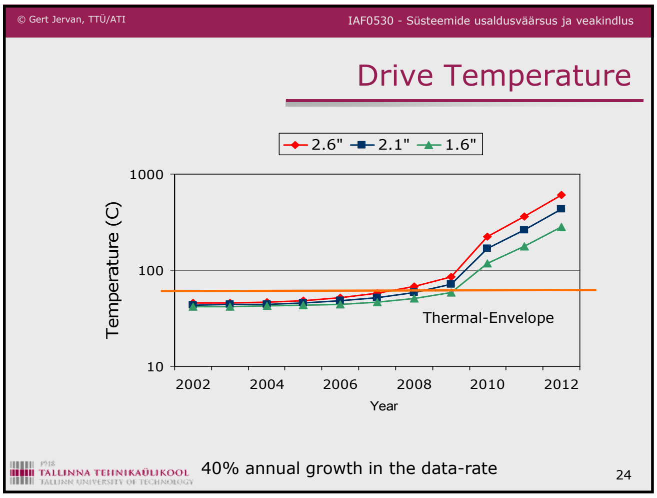
[Source: Intel Corporation and Prof. V. Oklobdzija]



Temperature Affects Disk Drive Reliability

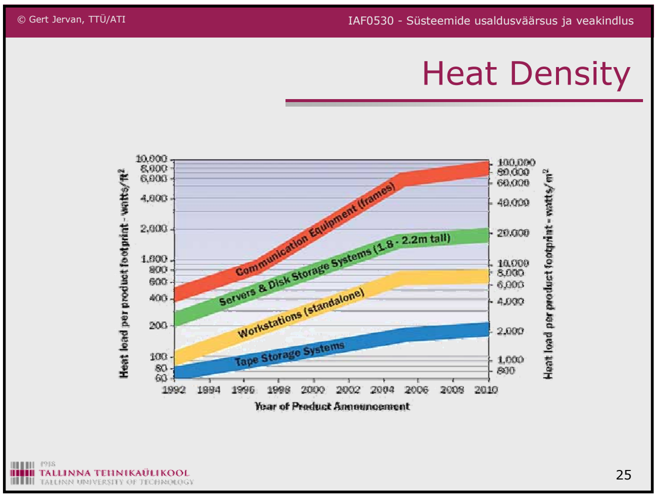
- ✓ Heat-Related Problems
  - Data corruption
  - Higher off-track errors
  - Head-crashes
- ✓ Disk drive design constrained by the thermal-envelope
  - Puts a limit on drive performance

Source: D. Anderson et al, "More than an Interface – SCSI vs. ATA", FAST 2003.

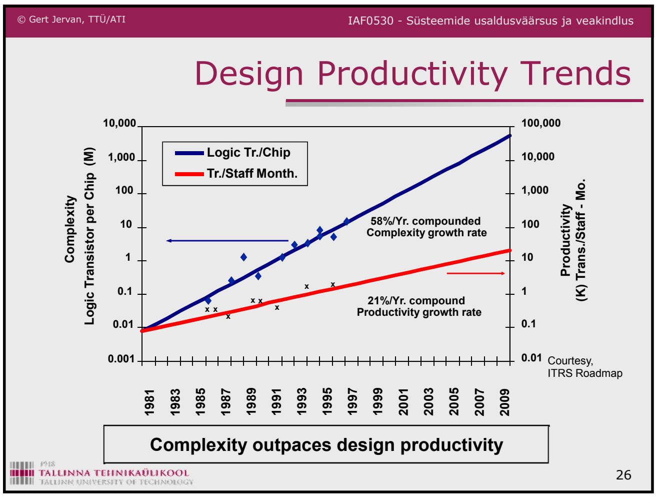


Drive Temperature

40% annual growth in the data-rate



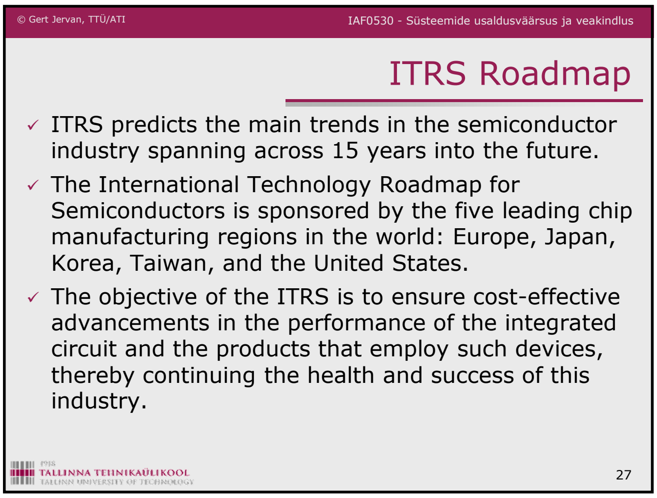
Heat Density



Design Productivity Trends

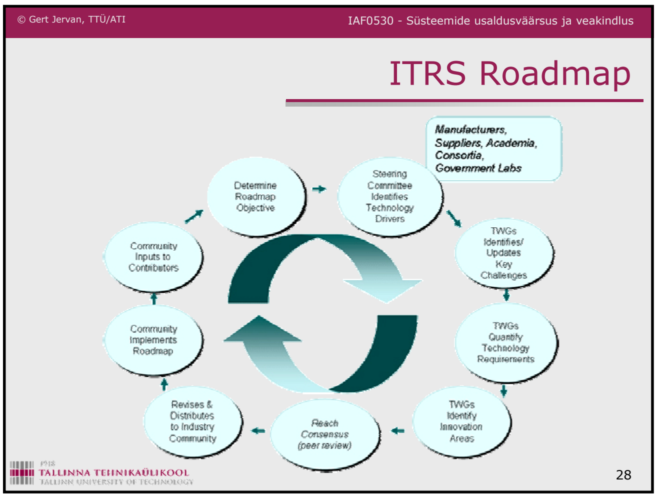
Courtesy, ITRS Roadmap

Complexity outpaces design productivity



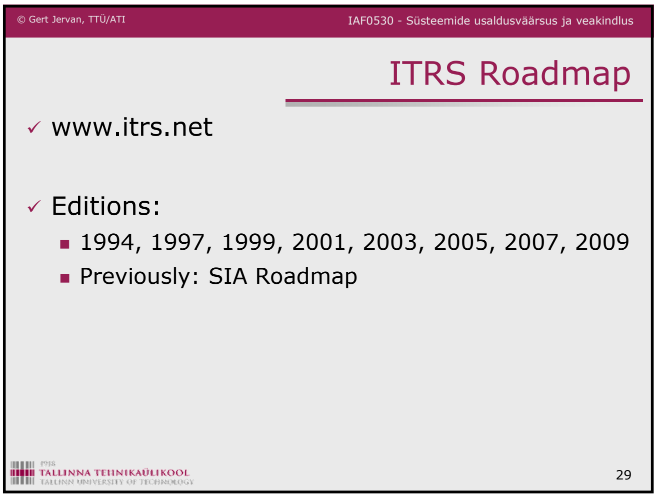
ITRS Roadmap

- ✓ ITRS predicts the main trends in the semiconductor industry spanning across 15 years into the future.
- ✓ The International Technology Roadmap for Semiconductors is sponsored by the five leading chip manufacturing regions in the world: Europe, Japan, Korea, Taiwan, and the United States.
- ✓ The objective of the ITRS is to ensure cost-effective advancements in the performance of the integrated circuit and the products that employ such devices, thereby continuing the health and success of this industry.



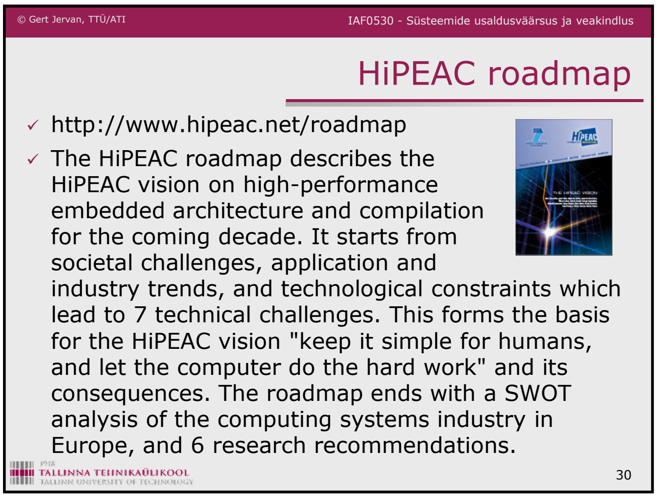
ITRS Roadmap

Manufacturers, Suppliers, Academia, Consortia, Government Labs
Determine Roadmap Objective
Steering Committee Identifies Technology Drivers
TWGs Identifies/ Updates Key Challenges
TWGs Quantify Technology Requirements
TWGs Identify Innovation Areas
Reach Consensus (peer review)
Revises & Distributes to Industry Community
Community Implements Roadmap
Community Inputs to Contributors



ITRS Roadmap

- ✓ www.itrs.net
- ✓ Editions:
  - 1994, 1997, 1999, 2001, 2003, 2005, 2007, 2009
  - Previously: SIA Roadmap



HiPEAC roadmap

- ✓ http://www.hipeac.net/roadmap
- ✓ The HiPEAC roadmap describes the HiPEAC vision on high-performance embedded architecture and compilation for the coming decade. It starts from societal challenges, application and industry trends, and technological constraints which lead to 7 technical challenges. This forms the basis for the HiPEAC vision "keep it simple for humans, and let the computer do the hard work" and its consequences. The roadmap ends with a SWOT analysis of the computing systems industry in Europe, and 6 research recommendations.

Technology Directions: ITRS Roadmap

Year	1999	2002	2005	2008	2011	2014
Feature size (nm)	180	130	100	70	50	35
Mtrans/cm^2	7	14-26	47	115	284	701
Chip size (mm^2)	170	170-214	235	269	308	354
Signal pins/chip	768	1024	1024	1280	1408	1472
Clock rate (MHz)	600	800	1100	1400	1800	2200
Wiring levels	6-7	7-8	8-9	9	9-10	10
Power supply (V)	1.8	1.5	1.2	0.9	0.6	0.6
High-perf power (W)	90	130	160	170	174	183
Battery power (W)	1.4	2.0	2.4	2.0	2.2	2.4

For Cost-Performance MPU
(L1 on-chip SRAM cache; 32KB/1999 doubling every two years)

http://www.itrs.net/

Industry Scaling Trends & Reliability Considerations

- ✓ Reduced gate oxide thicknesses
- ✓ Increased thermal/power densities
- ✓ Reduced interconnect dimensions
- ✓ Higher device operating temperatures
- ✓ Increased sensitivity to defects and statistical process variations
- ✓ Introduction of new materials with each new generation, replacing proven materials
  - e.g. Cu and low K inter-level dielectrics for Al and SiO2

Industry 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

Industry Scaling Trends & Reliability Considerations

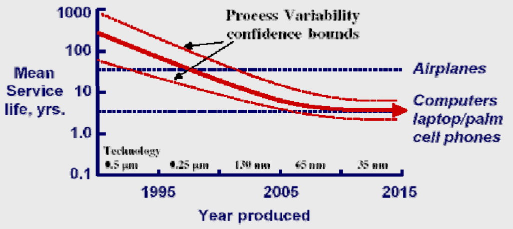
- ✓ Significant rise in the amount of proprietary technology and data developed by manufacturers, reluctance to share information with hi-rel customers
  - e.g. process recipes, process controls, process flows, design margins, MTTF
- ✓ Next generation microelectronics focus on the performance needs of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

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.***

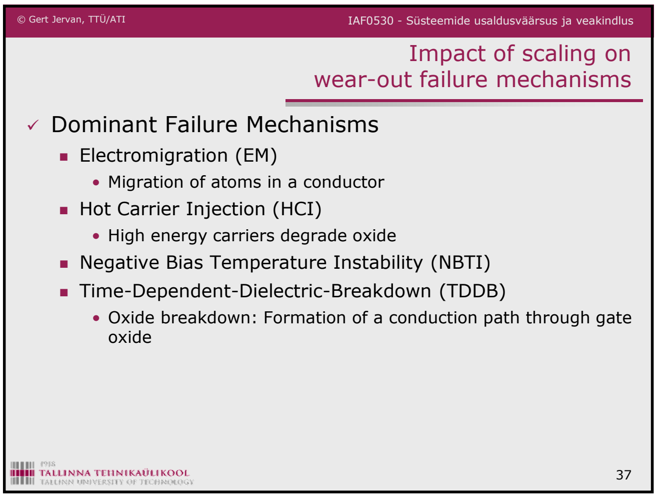
*MRQW-2002, Bernstein

Commercial Chip Reliability Estimation



Known trends for TDDB, EM and HCI degradation

*Extrapolated from ITRS roadmap, MRQW-2002, Bernstein

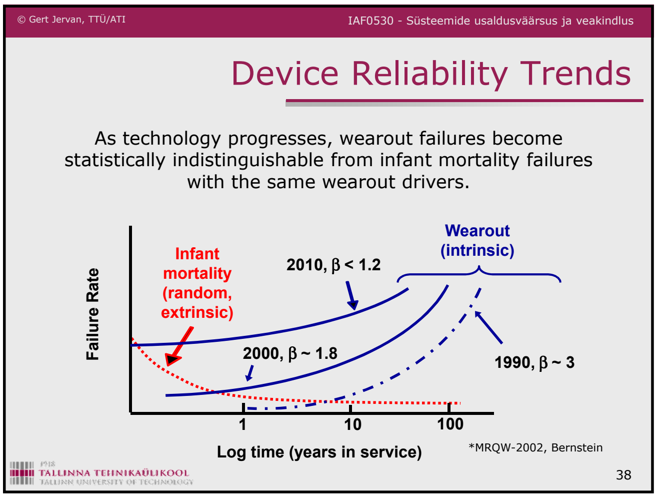


© Gert Jervan, TTÜ/ATI IAF0530 - Süsteemide usaldusväärsus ja veakindlus

Impact of scaling on wear-out failure mechanisms

- ✓ Dominant Failure Mechanisms
  - Electromigration (EM)
    - Migration of atoms in a conductor
  - Hot Carrier Injection (HCI)
    - High energy carriers degrade oxide
  - Negative Bias Temperature Instability (NBTI)
  - Time-Dependent-Dielectric-Breakdown (TDDB)
    - Oxide breakdown: Formation of a conduction path through gate oxide

TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY 37



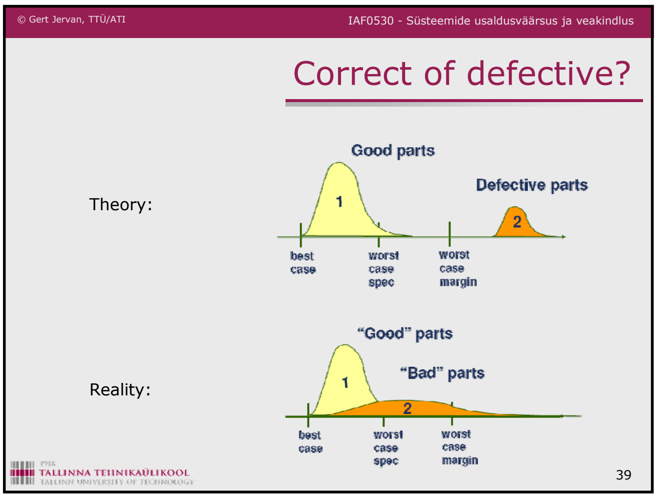
© Gert Jervan, TTÜ/ATI IAF0530 - Süsteemide usaldusväärsus ja veakindlus

Device Reliability Trends

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

*MRQW-2002, Bernstein

TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY 38



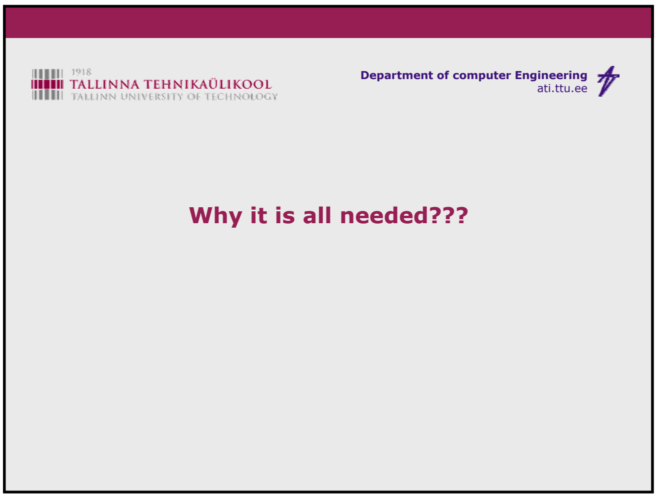
© Gert Jervan, TTÜ/ATI IAF0530 - Süsteemide usaldusväärsus ja veakindlus

Correct of defective?

Theory:

Reality:

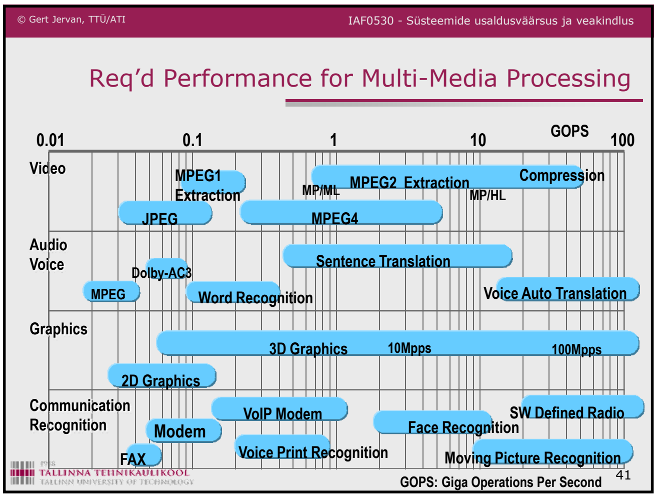
TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY 39



TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

Department of computer Engineering
ati.ttu.ee

Why it is all needed???

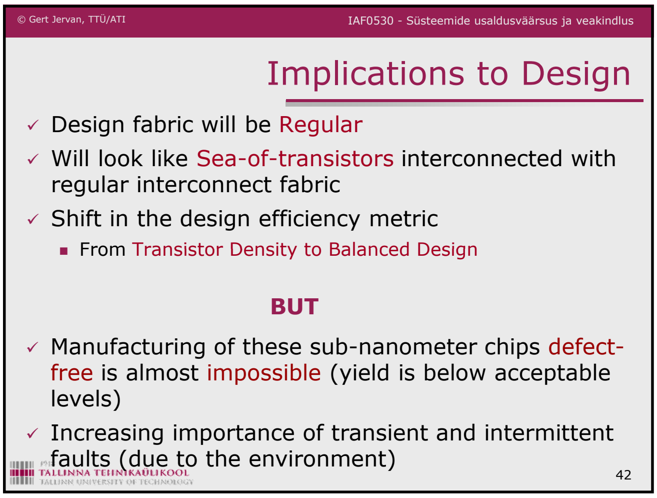


© Gert Jervan, TTÜ/ATI IAF0530 - Süsteemide usaldusväärsus ja veakindlus

Req’d Performance for Multi-Media Processing

GOPS: Giga Operations Per Second

TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY 41



© Gert Jervan, TTÜ/ATI IAF0530 - Süsteemide usaldusväärsus ja veakindlus

Implications to Design

- ✓ Design fabric will be Regular
- ✓ Will look like Sea-of-transistors interconnected with regular interconnect fabric
- ✓ Shift in the design efficiency metric
  - From Transistor Density to Balanced Design

BUT

- ✓ Manufacturing of these sub-nanometer chips defect-free is almost impossible (yield is below acceptable levels)
- ✓ Increasing importance of transient and intermittent faults (due to the environment)

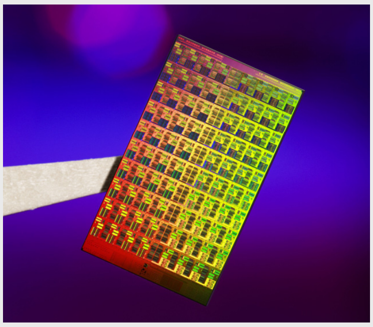
TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY 42

New Architectures

- ✓ Massively parallel architectures (Von Neuman is dead...) based on hundreds (millions) of (non-) reliable components
  - Multiple Input stream, Multiple Data stream machines
  - Wide use of network infrastructures (Networks-on-Chip)
  - Built-In Self-Repair will become a widespread technology
    - Dynamic reconfiguration

Intel Polaris 8x10 Network on Chip

- ✓ 8x10 processors on one chip, 65 nm
- ✓ Teraflops performance under 100 W
- ✓ Peak performance up to 2 Tflops
- ✓ Each processor:
  - 5 GHz
  - 20 Gflops
  - @1.2V



1918 TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

Department of computer Engineering
ati.ttu.ee

The problem to be solved:

How to design reliable system out of non-reliable hardware?

1918 TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

Department of computer Engineering
ati.ttu.ee

Software Failures

Some information on the following slides: © George Candea

Software

- ✓ Is software getting worse?
  - Tandem OS (1985): 4 MLOC
  - Linux (2001): 30 MLOC (kernel 2.6.29: 11 MLOC)
  - Windows XP (2001): 35 MLOC
  - MS Vista (2006): 50 MLOC
  - Jim Gray’s estimate: 1 bug/KLOC
  - Reducing bugs/KLOC vs. increasing KLOCs/product

Failures

- ✓ Hard to pinpoint a single root cause:
  - Coca-cola → disk crash → database failure
- ✓ Software bugs are faults!

Types of Bugs

- ✓ **Heisenbug**: disappears (or manifests differently) when you try to research it
  - Named after "Heisenberg uncertainty principle"
  - Debug mode versus release mode
    - Uninitialized variables
    - Fandango on core
  - Race conditions

Types of Bugs

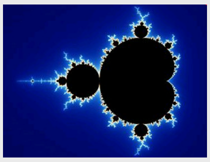
- ✓ **Bohrbug**: constant, reproducible, easy to deal with
  - Named after the Bohr atom model
  - Bohrbug does not disappear or alter its characteristics when it is researched
  - Ghost in the code
    - Overflow

Types of Bugs

- ✓ **Schrödingbug**: only starts manifesting when
  - is used in an unusual way
  - someone realizes it should be there
  - Named after Schrödinger's cat thought experiment
  - Determinism!
  - It is important to repair, not to determine the cause
  - For example: DB system works with small amount of data but not with many records

Types of Bugs

- ✓ **Mandelbug**: underlying cause is so complex and obscure, it makes the bug seems nondeterministic
  - Named after fractal innovator Benoît Mandelbrot
  - A bug whose behavior does not appear chaotic, but whose causes are so complex that there is no practical solution.
  - For example: a flaw in the fundamental design of the entire system.



Duration of Failures

- ✓ Permanent failure: once it manifests, won't go away unless you repair the system
  E.g., cut a network cable
- ✓ Intermittent failure: only occurs on occasion, for unknown reasons (until debugged… often workload)
  E.g., Patriot missile defense
- ✓ Transient failure: if you wait or retry, goes away
  E.g., various media corruption

Software Failures

- ✓ crash
- ✓ hang
- ✓ respond correctly but too late
- ✓ provide wrong data

- ✓ how to classify ? (fail-stop, fail-fast, Byzantine)
- ✓ how does recovery affect classification ?

Bug Triggers

- ✓ Timing
  - interleaving of events → many execution traces
  - hard to test all
- ✓ Recovery code
  - deals with exceptions → hard to simulate prior to shipping (ex. check NULL on return from malloc())
  - fault injection often used
- ✓ Third-party code
  - customer software, drivers, extensions, library users
  - Microsoft's "driver certification" → a way to combat this
- ✓ Boundary conditions
  - simple ones found through static analysis, complex ones are hard
- ✓ Bug-fix patches
  - customer system diverges over time
  - OS patches particularly evil

1918 TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

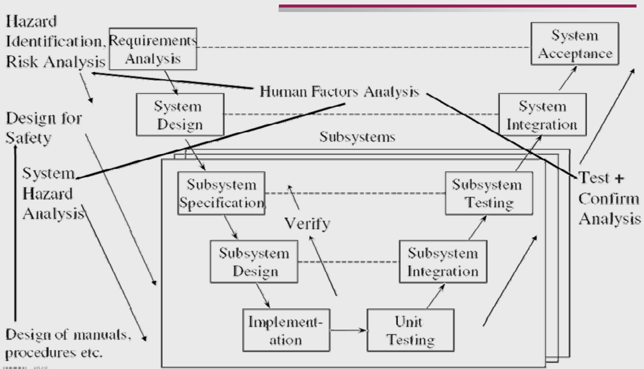
Department of computer Engineering
ati.ttu.ee

Human Factors

Human Factors

- ✓ The role of humans in safety-critical systems
- ✓ Human Reliability Analysis
  - task analysis
  - human error identification
    - human error model: Reason
  - human reliability quantification
  - mitigating human error
- ✓ Safe user interface design

Human Factors



Have we learnt since Therac-25

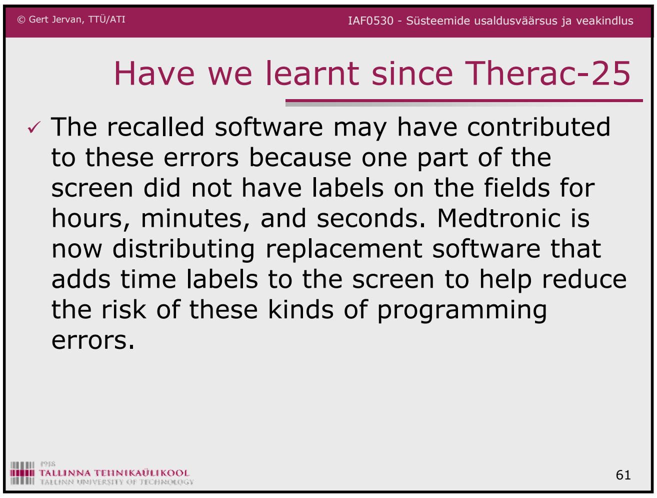
Software for Certain Medtronic Implanted Infusion Pumps Recalled

FDA Patient Safety News: Show #32, October 2004

- ✓ Medtronic is recalling certain software application cards.They're used in the company's Model 8840 N'Vision Clinician Programmers. These hand-held devices are used to program a number of implantable devices, including the SynchroMed and SychroMed EL implantable infusion pumps.

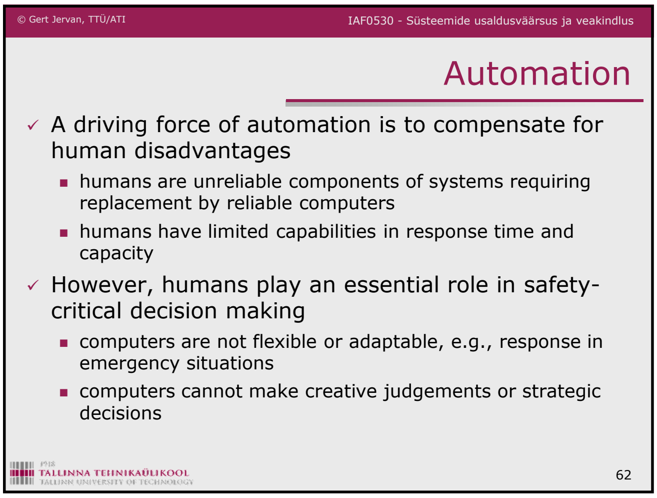
Have we learnt since Therac-25

- ✓ The recall is prompted by reports of data entry errors that have led to serious drug overdoses, including two patient deaths. The overdoses occurred when clinicians who were programming the pump entered the wrong time duration or the wrong interval --- for example, mistakenly putting the time interval between periodic drug boluses in the "minutes‟ field, instead of the "hours" field.



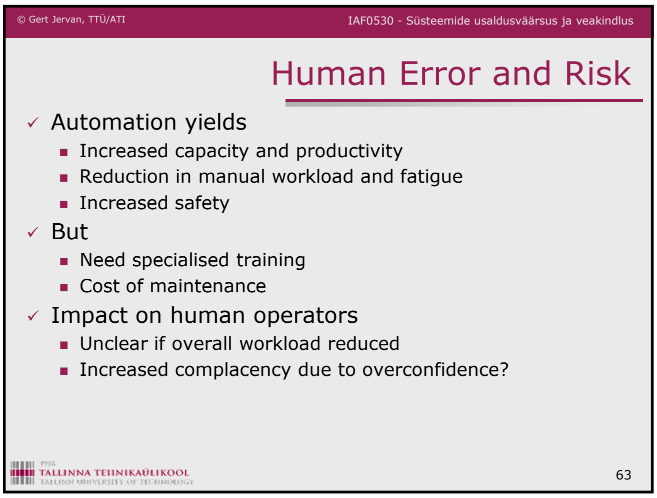
Have we learnt since Therac-25

- ✓ The recalled software may have contributed to these errors because one part of the screen did not have labels on the fields for hours, minutes, and seconds. Medtronic is now distributing replacement software that adds time labels to the screen to help reduce the risk of these kinds of programming errors.



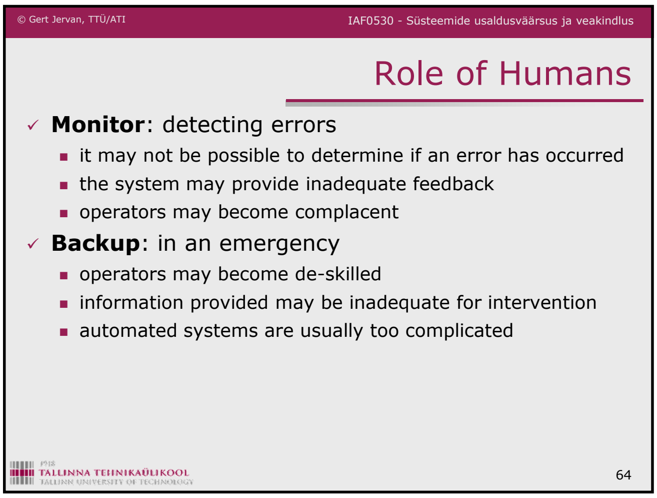
Automation

- ✓ A driving force of automation is to compensate for human disadvantages
  - humans are unreliable components of systems requiring replacement by reliable computers
  - humans have limited capabilities in response time and capacity
- ✓ However, humans play an essential role in safety-critical decision making
  - computers are not flexible or adaptable, e.g., response in emergency situations
  - computers cannot make creative judgements or strategic decisions



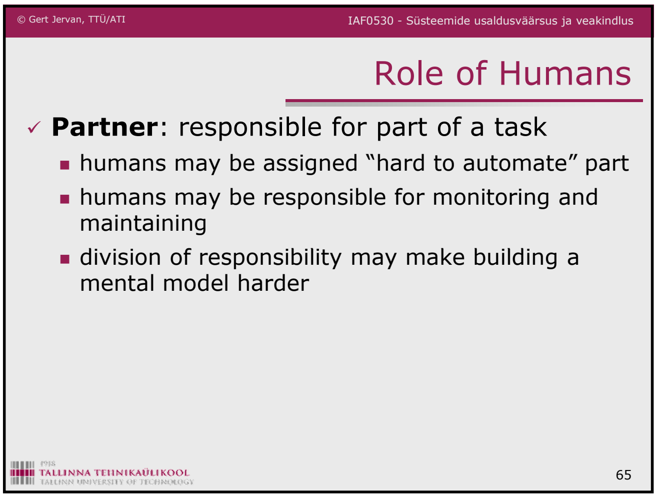
Human Error and Risk

- ✓ Automation yields
  - Increased capacity and productivity
  - Reduction in manual workload and fatigue
  - Increased safety
- ✓ But
  - Need specialised training
  - Cost of maintenance
- ✓ Impact on human operators
  - Unclear if overall workload reduced
  - Increased complacency due to overconfidence?



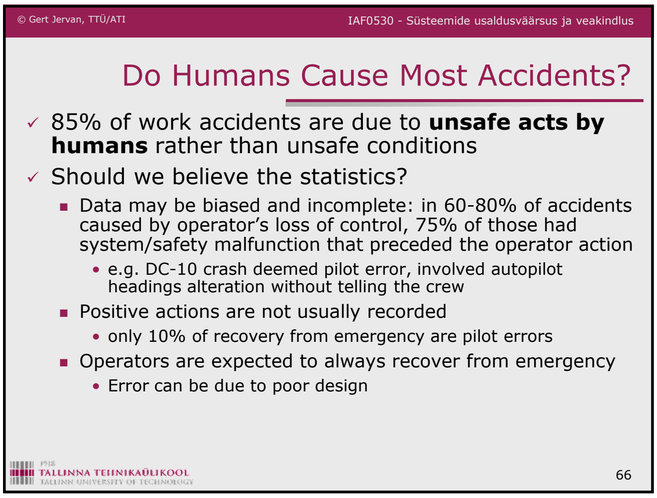
Role of Humans

- ✓ **Monitor**: detecting errors
  - it may not be possible to determine if an error has occurred
  - the system may provide inadequate feedback
  - operators may become complacent
- ✓ **Backup**: in an emergency
  - operators may become de-skilled
  - information provided may be inadequate for intervention
  - automated systems are usually too complicated



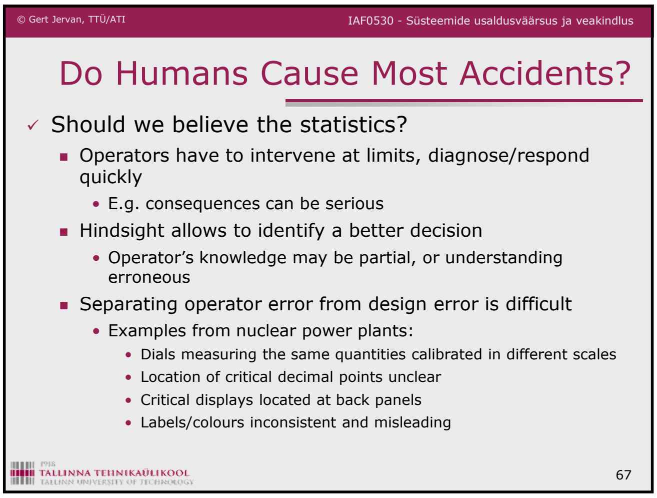
Role of Humans

- ✓ **Partner**: responsible for part of a task
  - humans may be assigned “hard to automate” part
  - humans may be responsible for monitoring and maintaining
  - division of responsibility may make building a mental model harder



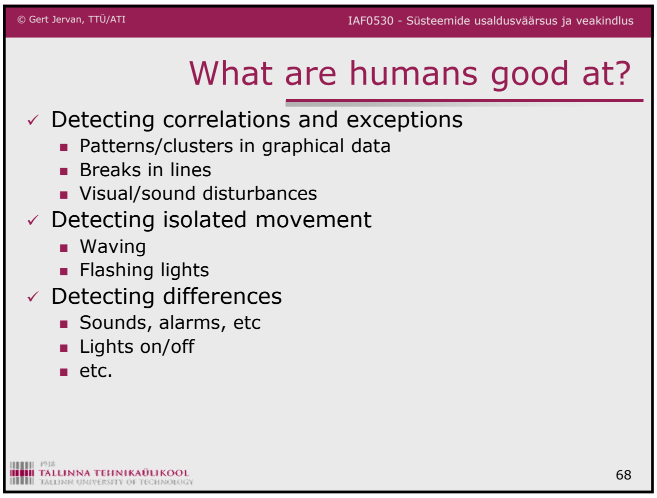
Do Humans Cause Most Accidents?

- ✓ 85% of work accidents are due to **unsafe acts by humans** rather than unsafe conditions
- ✓ Should we believe the statistics?
  - Data may be biased and incomplete: in 60-80% of accidents caused by operator’s loss of control, 75% of those had system/safety malfunction that preceded the operator action
    - e.g. DC-10 crash deemed pilot error, involved autopilot headings alteration without telling the crew
  - Positive actions are not usually recorded
    - only 10% of recovery from emergency are pilot errors
  - Operators are expected to always recover from emergency
    - Error can be due to poor design



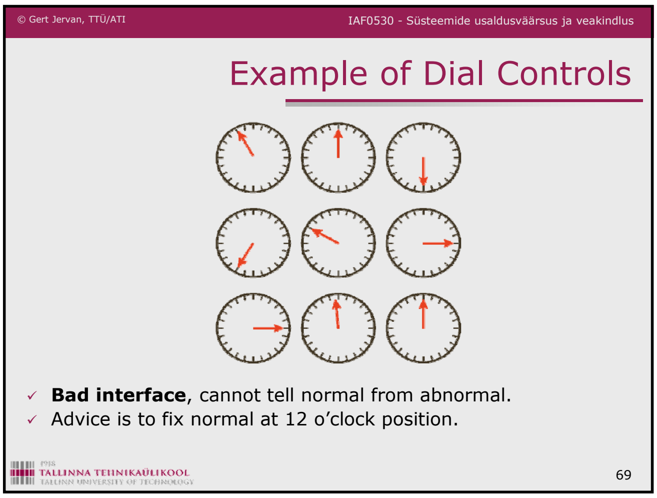
Do Humans Cause Most Accidents?

- ✓ Should we believe the statistics?
  - ▪ Operators have to intervene at limits, diagnose/respond quickly
    - • E.g. consequences can be serious
  - ▪ Hindsight allows to identify a better decision
    - • Operator’s knowledge may be partial, or understanding erroneous
  - ▪ Separating operator error from design error is difficult
    - • Examples from nuclear power plants:
      - • Dials measuring the same quantities calibrated in different scales
      - • Location of critical decimal points unclear
      - • Critical displays located at back panels
      - • Labels/colours inconsistent and misleading



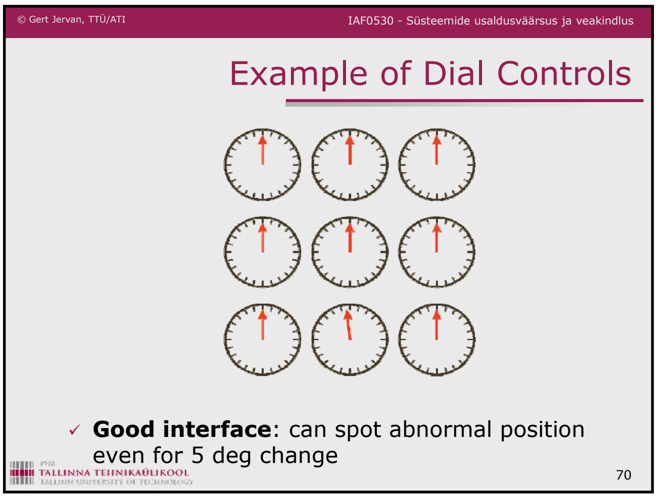
What are humans good at?

- ✓ Detecting correlations and exceptions
  - ▪ Patterns/clusters in graphical data
  - ▪ Breaks in lines
  - ▪ Visual/sound disturbances
- ✓ Detecting isolated movement
  - ▪ Waving
  - ▪ Flashing lights
- ✓ Detecting differences
  - ▪ Sounds, alarms, etc
  - ▪ Lights on/off
  - ▪ etc.



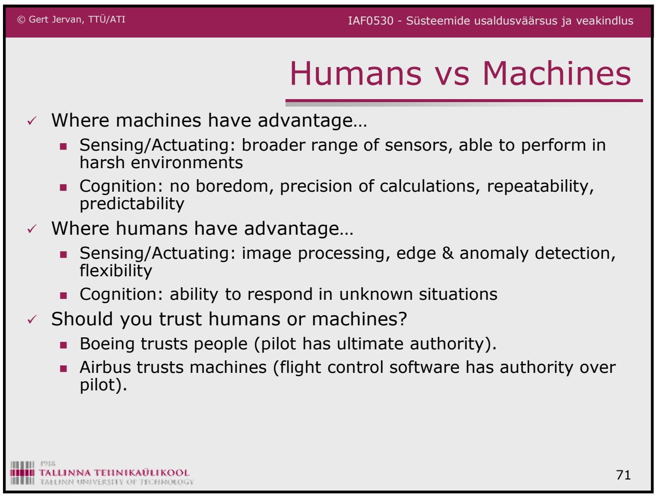
Example of Dial Controls

- ✓ **Bad interface**, cannot tell normal from abnormal.
- ✓ Advice is to fix normal at 12 o’clock position.



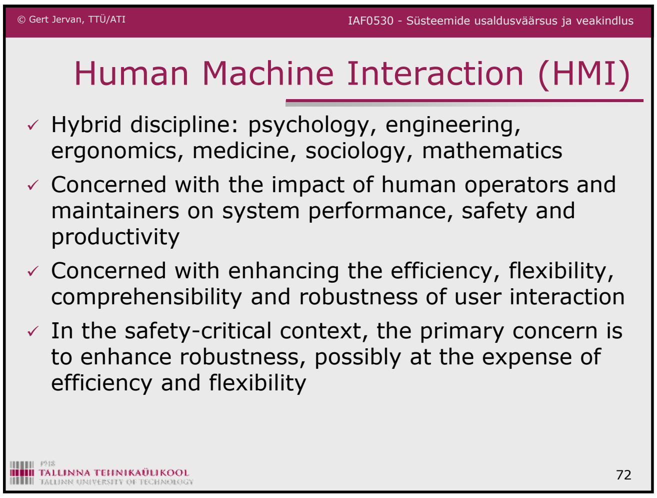
Example of Dial Controls

- ✓ **Good interface**: can spot abnormal position even for 5 deg change



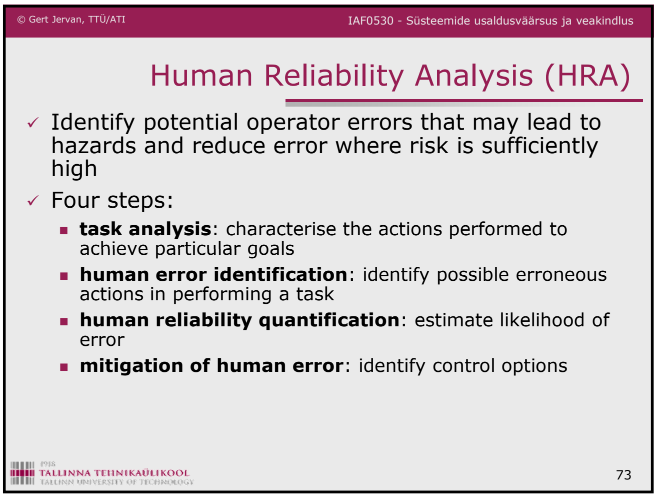
Humans vs Machines

- ✓ Where machines have advantage…
  - ▪ Sensing/Actuating: broader range of sensors, able to perform in harsh environments
  - ▪ Cognition: no boredom, precision of calculations, repeatability, predictability
- ✓ Where humans have advantage…
  - ▪ Sensing/Actuating: image processing, edge & anomaly detection, flexibility
  - ▪ Cognition: ability to respond in unknown situations
- ✓ Should you trust humans or machines?
  - ▪ Boeing trusts people (pilot has ultimate authority).
  - ▪ Airbus trusts machines (flight control software has authority over pilot).



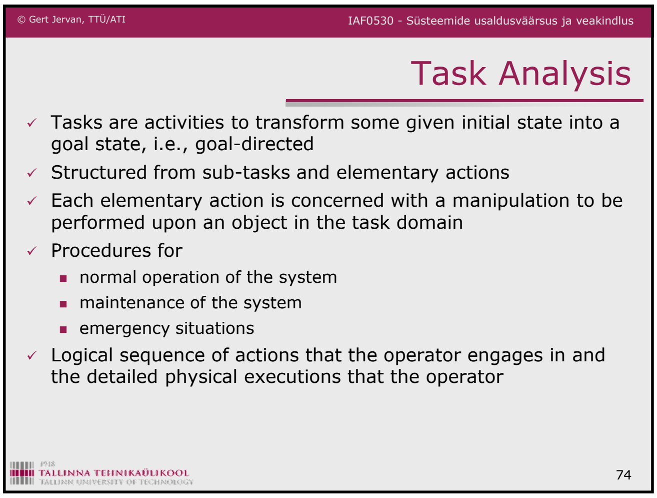
Human Machine Interaction (HMI)

- ✓ Hybrid discipline: psychology, engineering, ergonomics, medicine, sociology, mathematics
- ✓ Concerned with the impact of human operators and maintainers on system performance, safety and productivity
- ✓ Concerned with enhancing the efficiency, flexibility, comprehensibility and robustness of user interaction
- ✓ In the safety-critical context, the primary concern is to enhance robustness, possibly at the expense of efficiency and flexibility



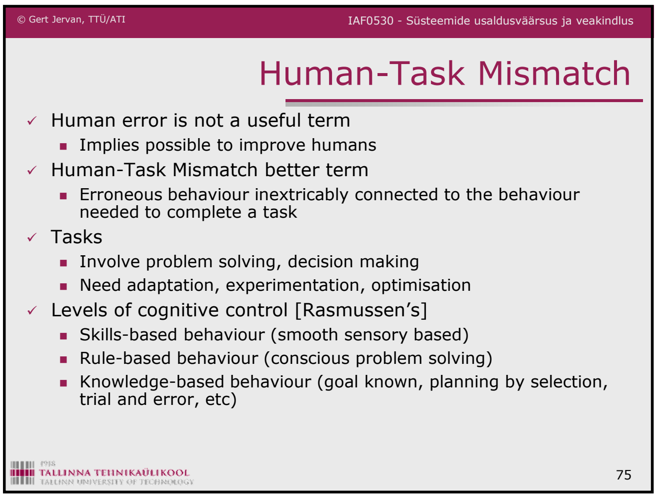
Human Reliability Analysis (HRA)

- ✓ Identify potential operator errors that may lead to hazards and reduce error where risk is sufficiently high
- ✓ Four steps:
  - **task analysis**: characterise the actions performed to achieve particular goals
  - **human error identification**: identify possible erroneous actions in performing a task
  - **human reliability quantification**: estimate likelihood of error
  - **mitigation of human error**: identify control options



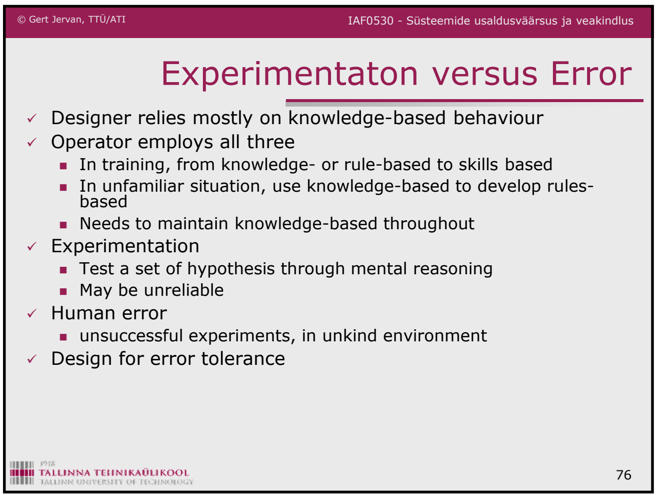
Task Analysis

- ✓ Tasks are activities to transform some given initial state into a goal state, i.e., goal-directed
- ✓ Structured from sub-tasks and elementary actions
- ✓ Each elementary action is concerned with a manipulation to be performed upon an object in the task domain
- ✓ Procedures for
  - normal operation of the system
  - maintenance of the system
  - emergency situations
- ✓ Logical sequence of actions that the operator engages in and the detailed physical executions that the operator



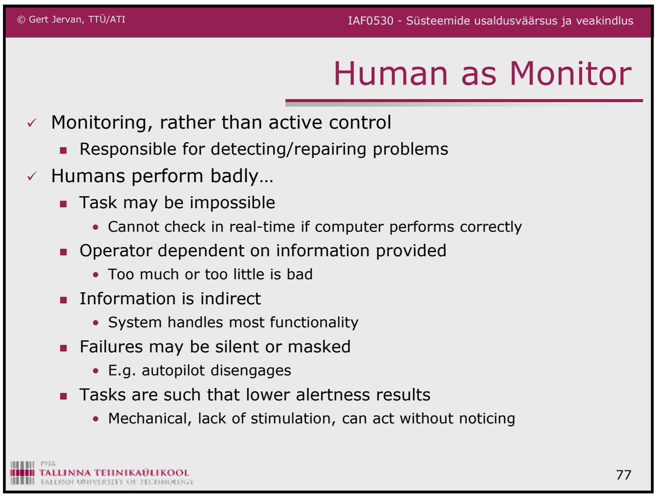
Human-Task Mismatch

- ✓ Human error is not a useful term
  - Implies possible to improve humans
- ✓ Human-Task Mismatch better term
  - Erroneous behaviour inextricably connected to the behaviour needed to complete a task
- ✓ Tasks
  - Involve problem solving, decision making
  - Need adaptation, experimentation, optimisation
- ✓ Levels of cognitive control [Rasmussen's]
  - Skills-based behaviour (smooth sensory based)
  - Rule-based behaviour (conscious problem solving)
  - Knowledge-based behaviour (goal known, planning by selection, trial and error, etc)



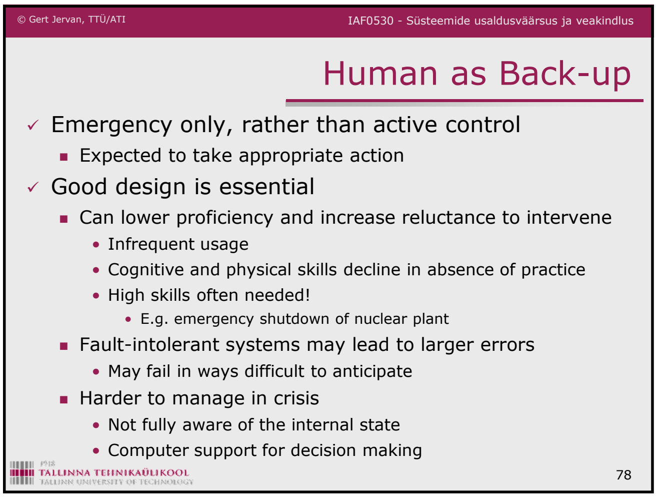
Experimentaton versus Error

- ✓ Designer relies mostly on knowledge-based behaviour
- ✓ Operator employs all three
  - In training, from knowledge- or rule-based to skills based
  - In unfamiliar situation, use knowledge-based to develop rules-based
  - Needs to maintain knowledge-based throughout
- ✓ Experimentation
  - Test a set of hypothesis through mental reasoning
  - May be unreliable
- ✓ Human error
  - unsuccessful experiments, in unkind environment
- ✓ Design for error tolerance



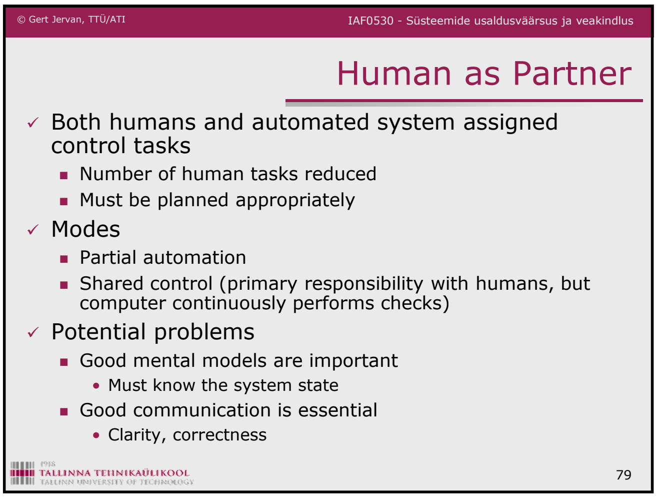
Human as Monitor

- ✓ Monitoring, rather than active control
  - Responsible for detecting/repairing problems
- ✓ Humans perform badly…
  - Task may be impossible
    - Cannot check in real-time if computer performs correctly
  - Operator dependent on information provided
    - Too much or too little is bad
  - Information is indirect
    - System handles most functionality
  - Failures may be silent or masked
    - E.g. autopilot disengages
  - Tasks are such that lower alertness results
    - Mechanical, lack of stimulation, can act without noticing



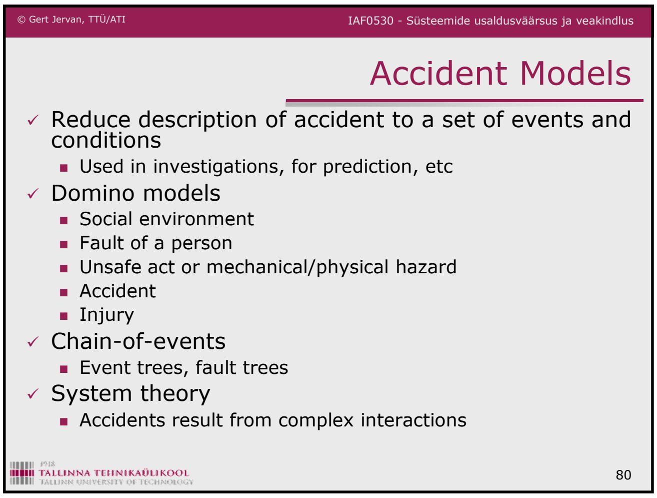
Human as Back-up

- ✓ Emergency only, rather than active control
  - Expected to take appropriate action
- ✓ Good design is essential
  - Can lower proficiency and increase reluctance to intervene
    - Infrequent usage
    - Cognitive and physical skills decline in absence of practice
    - High skills often needed!
      - E.g. emergency shutdown of nuclear plant
  - Fault-intolerant systems may lead to larger errors
    - May fail in ways difficult to anticipate
  - Harder to manage in crisis
    - Not fully aware of the internal state
    - Computer support for decision making



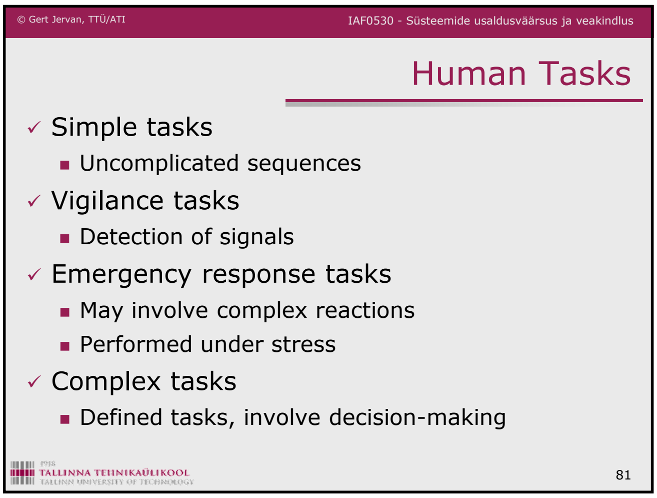
Human as Partner

- ✓ Both humans and automated system assigned control tasks
  - Number of human tasks reduced
  - Must be planned appropriately
- ✓ Modes
  - Partial automation
  - Shared control (primary responsibility with humans, but computer continuously performs checks)
- ✓ Potential problems
  - Good mental models are important
    - Must know the system state
  - Good communication is essential
    - Clarity, correctness



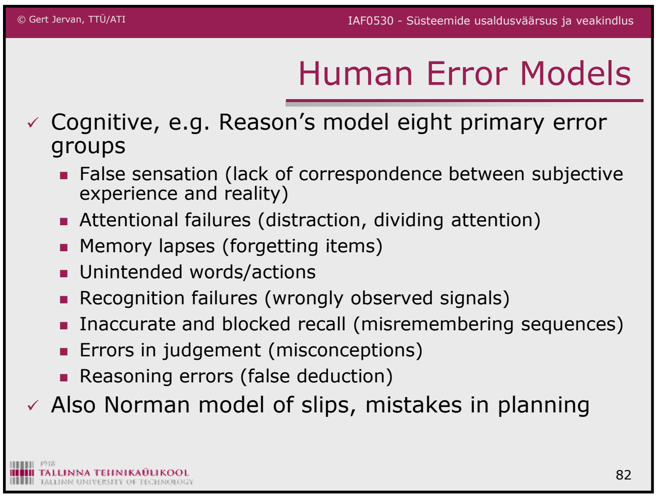
Accident Models

- ✓ Reduce description of accident to a set of events and conditions
  - Used in investigations, for prediction, etc
- ✓ Domino models
  - Social environment
  - Fault of a person
  - Unsafe act or mechanical/physical hazard
  - Accident
  - Injury
- ✓ Chain-of-events
  - Event trees, fault trees
- ✓ System theory
  - Accidents result from complex interactions



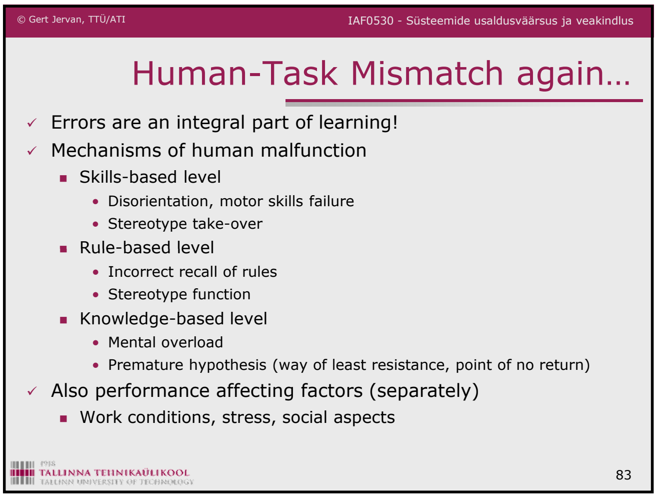
Human Tasks

- ✓ Simple tasks
  - Uncomplicated sequences
- ✓ Vigilance tasks
  - Detection of signals
- ✓ Emergency response tasks
  - May involve complex reactions
  - Performed under stress
- ✓ Complex tasks
  - Defined tasks, involve decision-making



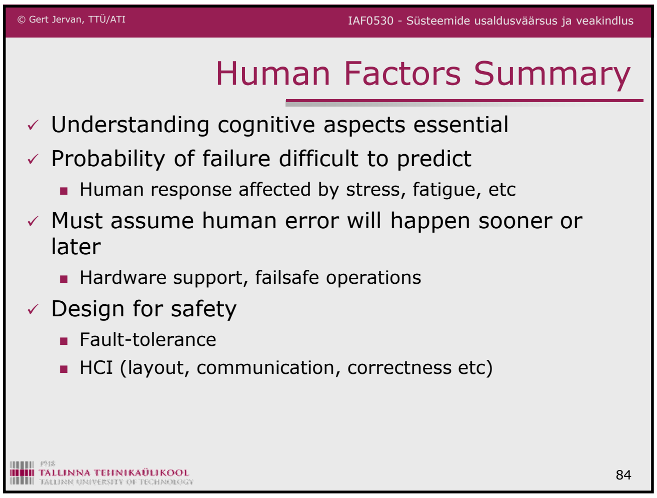
Human Error Models

- ✓ Cognitive, e.g. Reason's model eight primary error groups
  - False sensation (lack of correspondence between subjective experience and reality)
  - Attentional failures (distraction, dividing attention)
  - Memory lapses (forgetting items)
  - Unintended words/actions
  - Recognition failures (wrongly observed signals)
  - Inaccurate and blocked recall (misremembering sequences)
  - Errors in judgement (misconceptions)
  - Reasoning errors (false deduction)
- ✓ Also Norman model of slips, mistakes in planning



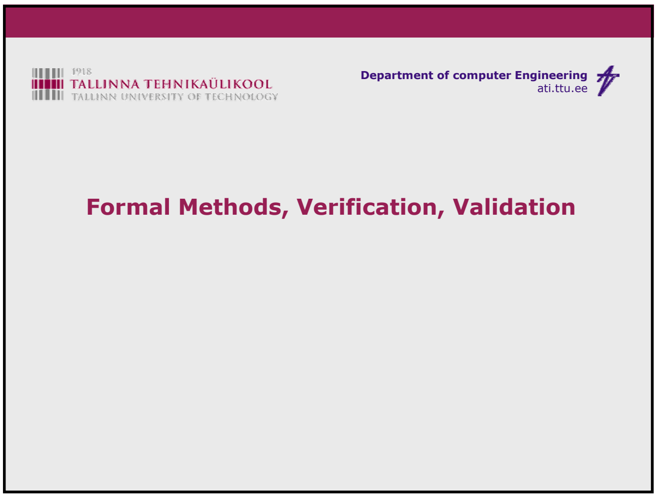
Human-Task Mismatch again…

- ✓ Errors are an integral part of learning!
- ✓ Mechanisms of human malfunction
  - Skills-based level
    - Disorientation, motor skills failure
    - Stereotype take-over
  - Rule-based level
    - Incorrect recall of rules
    - Stereotype function
  - Knowledge-based level
    - Mental overload
    - Premature hypothesis (way of least resistance, point of no return)
- ✓ Also performance affecting factors (separately)
  - Work conditions, stress, social aspects



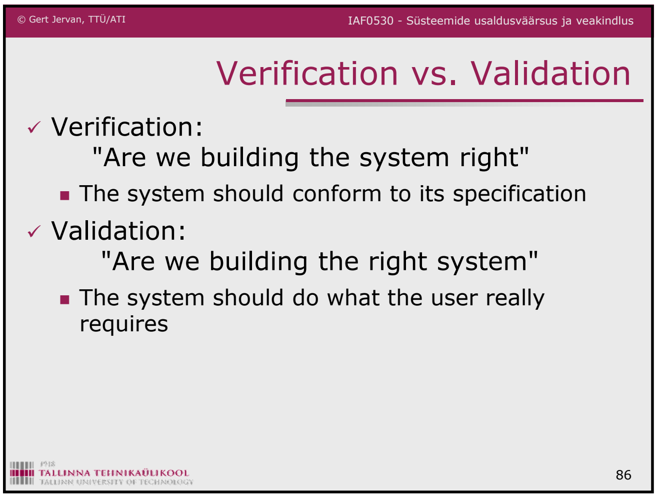
Human Factors Summary

- ✓ Understanding cognitive aspects essential
- ✓ Probability of failure difficult to predict
  - Human response affected by stress, fatigue, etc
- ✓ Must assume human error will happen sooner or later
  - Hardware support, failsafe operations
- ✓ Design for safety
  - Fault-tolerance
  - HCI (layout, communication, correctness etc)



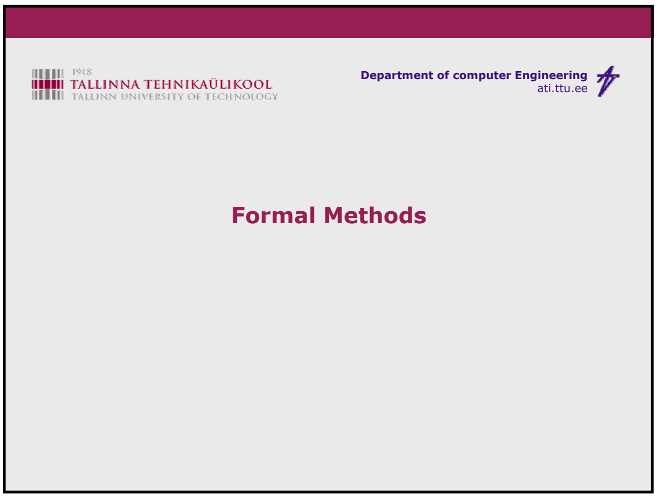
Department of computer Engineering
ati.ttu.ee

Formal Methods, Verification, Validation



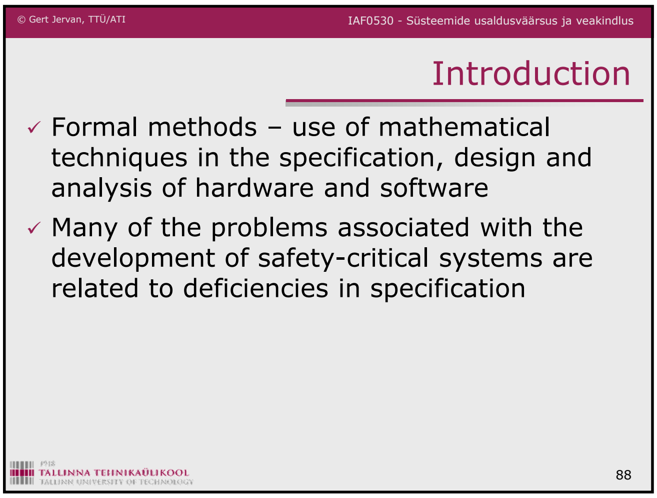
Verification vs. Validation

- ✓ Verification:
  "Are we building the system right"
  - The system should conform to its specification
- ✓ Validation:
  "Are we building the right system"
  - The system should do what the user really requires



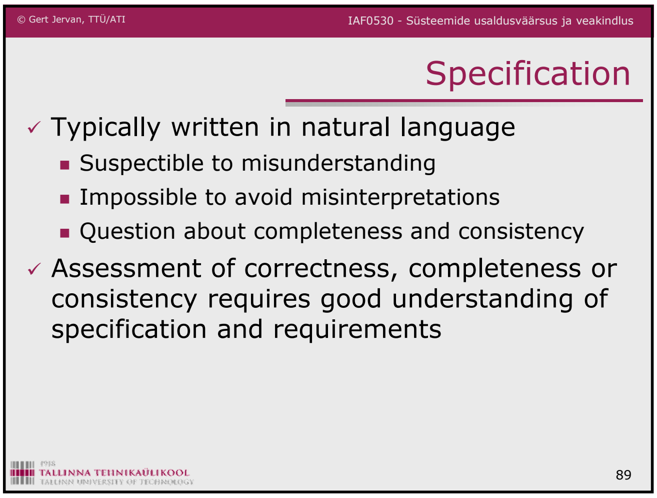
Department of computer Engineering
ati.ttu.ee

Formal Methods



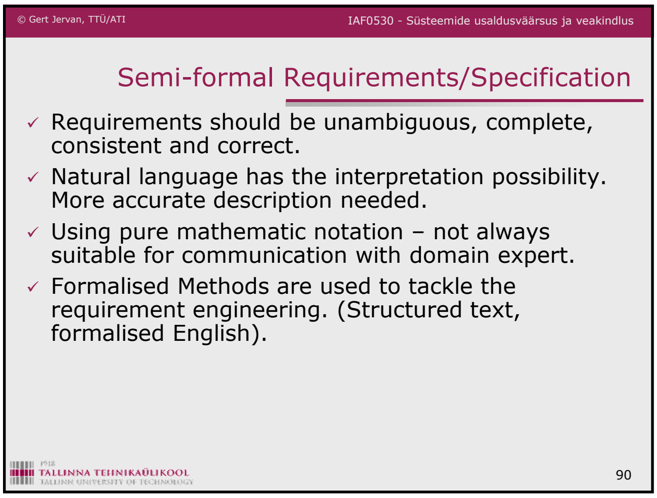
Introduction

- ✓ Formal methods – use of mathematical techniques in the specification, design and analysis of hardware and software
- ✓ Many of the problems associated with the development of safety-critical systems are related to deficiencies in specification



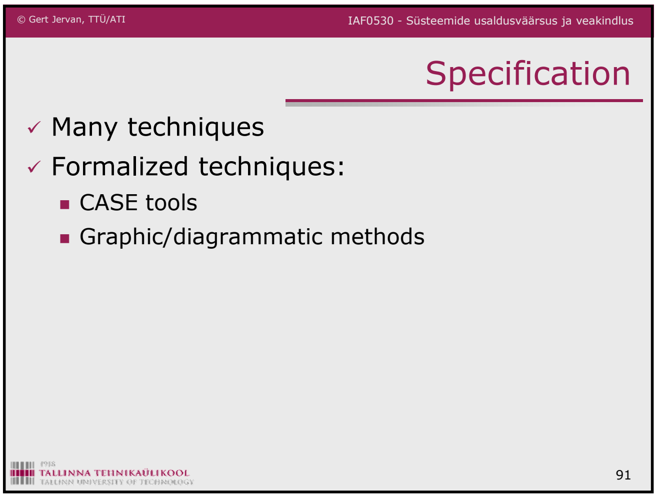
Specification

- ✓ Typically written in natural language
  - Suspectible to misunderstanding
  - Impossible to avoid misinterpretations
  - Question about completeness and consistency
- ✓ Assessment of correctness, completeness or consistency requires good understanding of specification and requirements



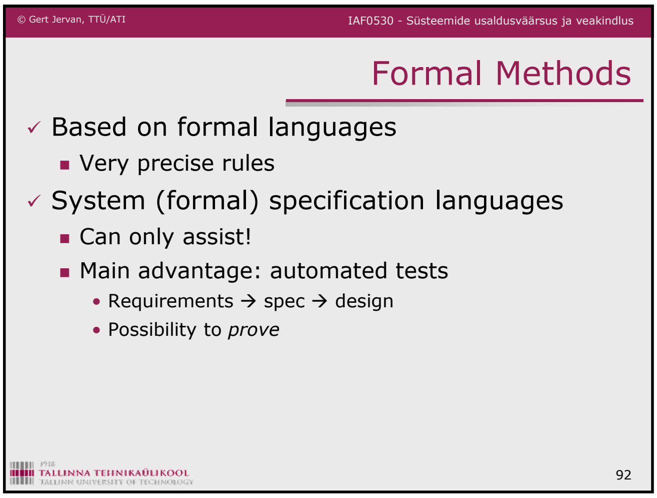
Semi-formal Requirements/Specification

- ✓ Requirements should be unambiguous, complete, consistent and correct.
- ✓ Natural language has the interpretation possibility. More accurate description needed.
- ✓ Using pure mathematic notation – not always suitable for communication with domain expert.
- ✓ Formalised Methods are used to tackle the requirement engineering. (Structured text, formalised English).



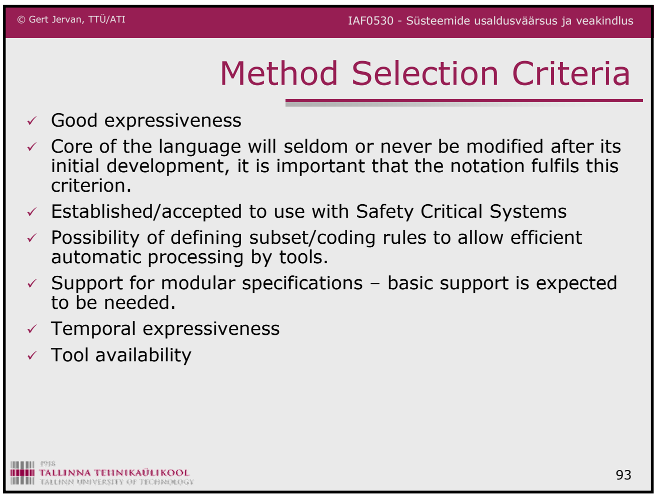
Specification

- ✓ Many techniques
- ✓ Formalized techniques:
  - CASE tools
  - Graphic/diagrammatic methods



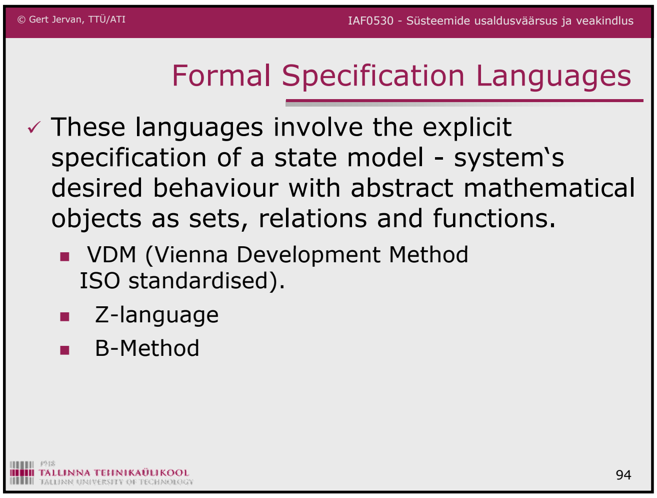
Formal Methods

- ✓ Based on formal languages
  - Very precise rules
- ✓ System (formal) specification languages
  - Can only assist!
  - Main advantage: automated tests
    - Requirements → spec → design
    - Possibility to *prove*



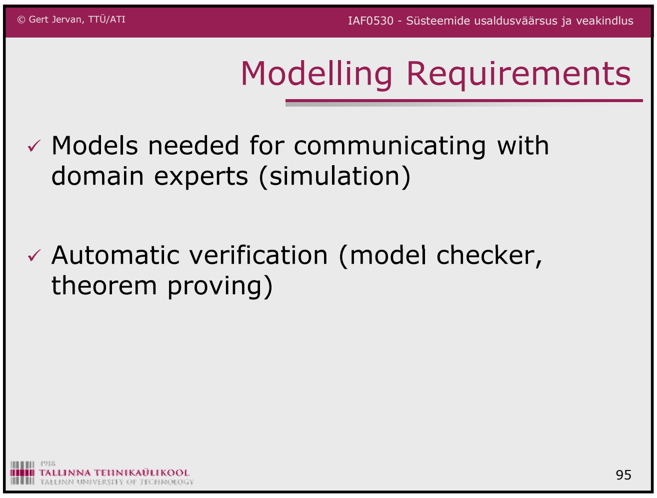
Method Selection Criteria

- ✓ Good expressiveness
- ✓ Core of the language will seldom or never be modified after its initial development, it is important that the notation fulfils this criterion.
- ✓ Established/accepted to use with Safety Critical Systems
- ✓ Possibility of defining subset/coding rules to allow efficient automatic processing by tools.
- ✓ Support for modular specifications – basic support is expected to be needed.
- ✓ Temporal expressiveness
- ✓ Tool availability



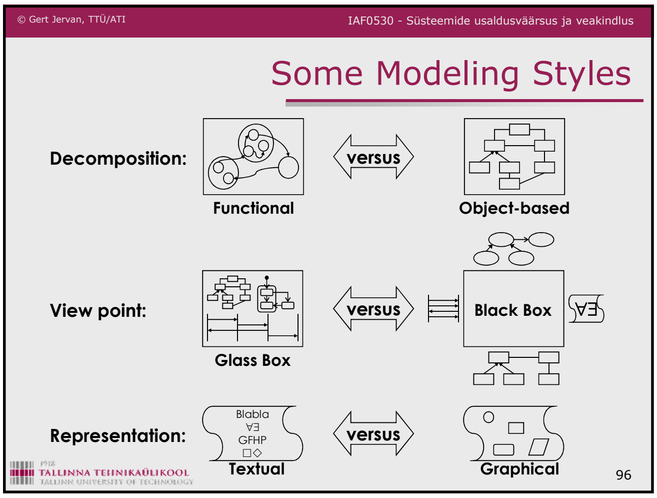
Formal Specification Languages

- ✓ These languages involve the explicit specification of a state model - system�based desired behaviour with abstract mathematical objects as sets, relations and functions.
  - VDM (Vienna Development Method ISO standardised).
  - Z-language
  - B-Method

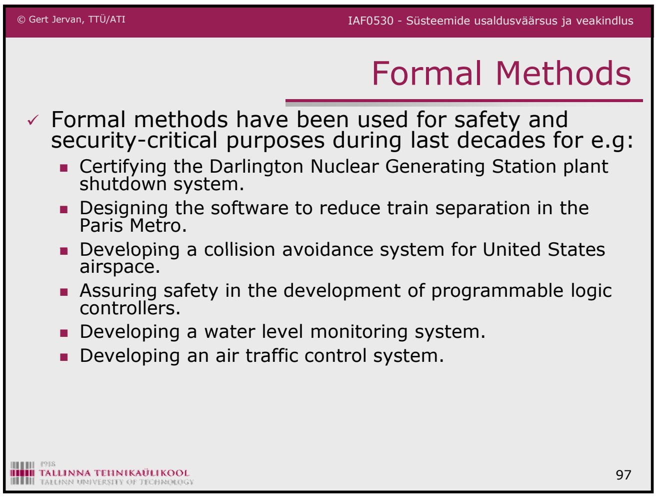


Modelling Requirements

- ✓ Models needed for communicating with domain experts (simulation)
- ✓ Automatic verification (model checker, theorem proving)

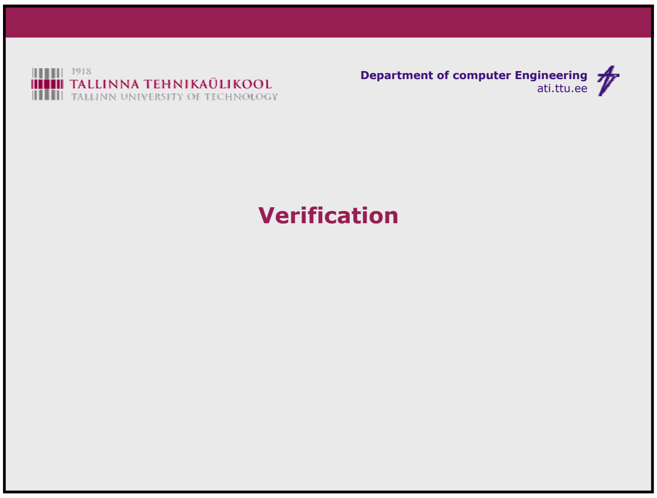


Some Modeling Styles

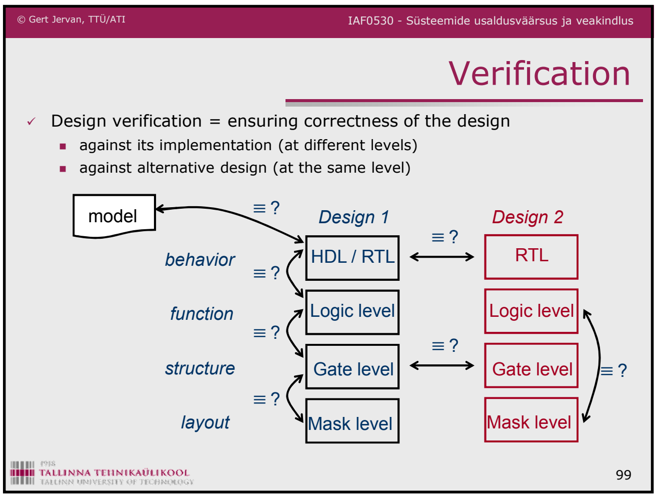


Formal Methods

- ✓ Formal methods have been used for safety and security-critical purposes during last decades for e.g:
  - Certifying the Darlington Nuclear Generating Station plant shutdown system.
  - Designing the software to reduce train separation in the Paris Metro.
  - Developing a collision avoidance system for United States airspace.
  - Assuring safety in the development of programmable logic controllers.
  - Developing a water level monitoring system.
  - Developing an air traffic control system.

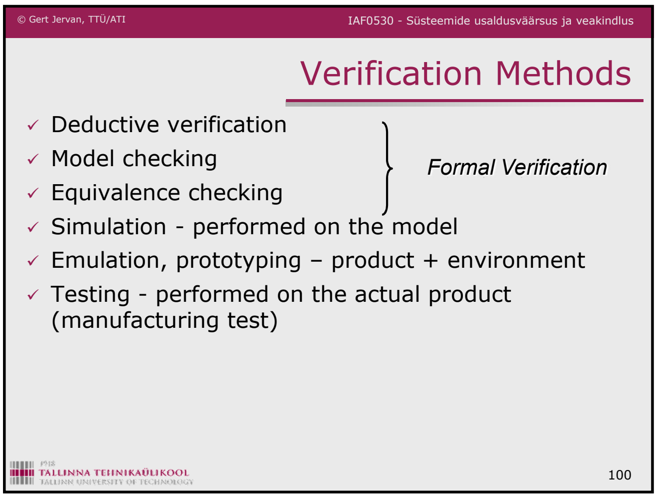


Verification



Verification

- ✓ Design verification = ensuring correctness of the design
  - against its implementation (at different levels)
  - against alternative design (at the same level)

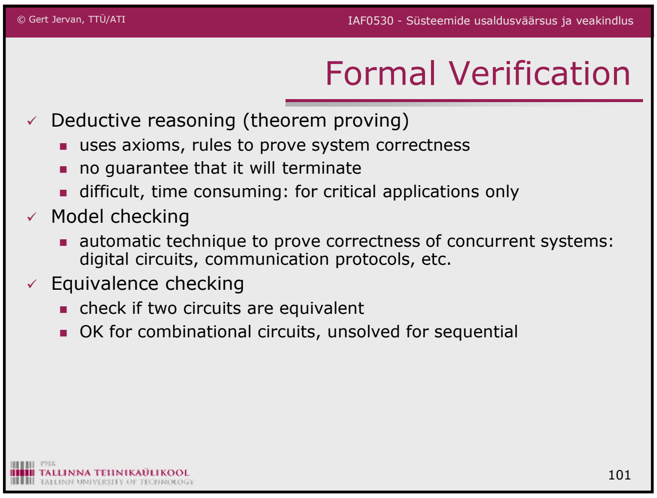


Verification Methods

- ✓ Deductive verification
- ✓ Model checking
- ✓ Equivalence checking

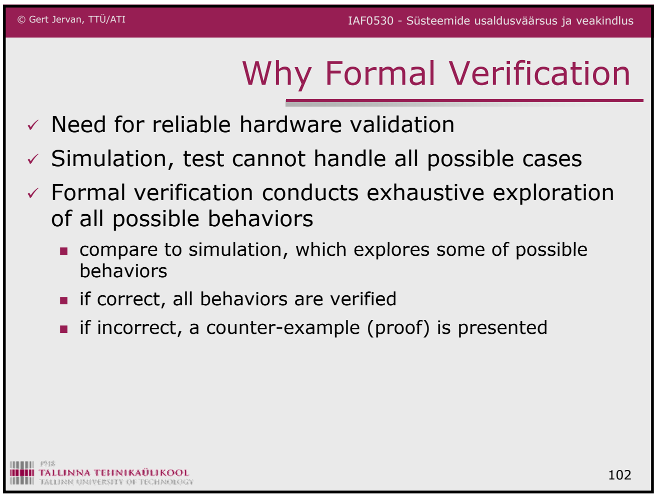
} *Formal Verification*

- ✓ Simulation - performed on the model
- ✓ Emulation, prototyping – product + environment
- ✓ Testing - performed on the actual product (manufacturing test)



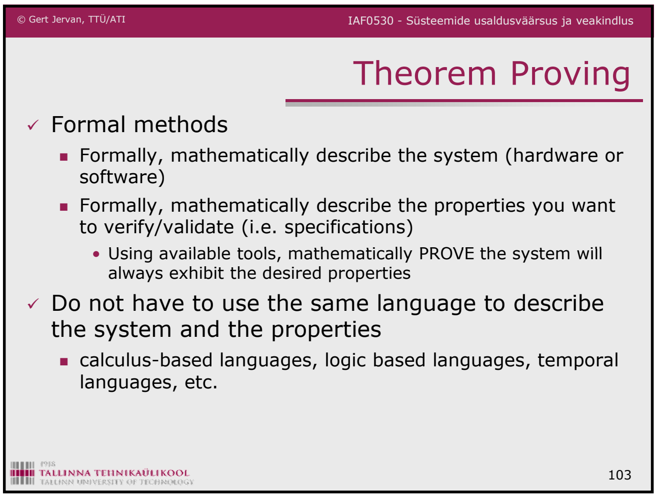
Formal Verification

- ✓ Deductive reasoning (theorem proving)
  - uses axioms, rules to prove system correctness
  - no guarantee that it will terminate
  - difficult, time consuming: for critical applications only
- ✓ Model checking
  - automatic technique to prove correctness of concurrent systems: digital circuits, communication protocols, etc.
- ✓ Equivalence checking
  - check if two circuits are equivalent
  - OK for combinational circuits, unsolved for sequential



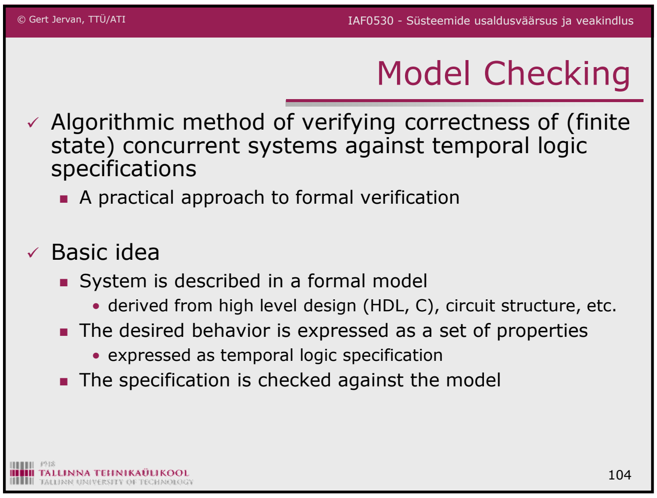
Why Formal Verification

- ✓ Need for reliable hardware validation
- ✓ Simulation, test cannot handle all possible cases
- ✓ Formal verification conducts exhaustive exploration of all possible behaviors
  - compare to simulation, which explores some of possible behaviors
  - if correct, all behaviors are verified
  - if incorrect, a counter-example (proof) is presented



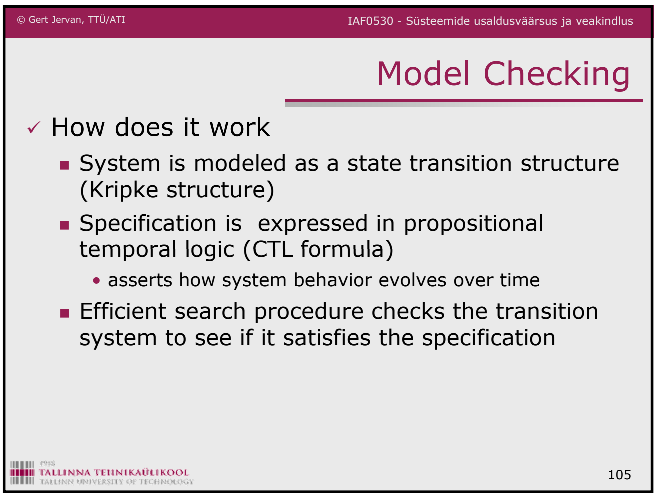
Theorem Proving

- Formal methods
  - Formally, mathematically describe the system (hardware or software)
  - Formally, mathematically describe the properties you want to verify/validate (i.e. specifications)
    - Using available tools, mathematically PROVE the system will always exhibit the desired properties
- Do not have to use the same language to describe the system and the properties
  - calculus-based languages, logic based languages, temporal languages, etc.



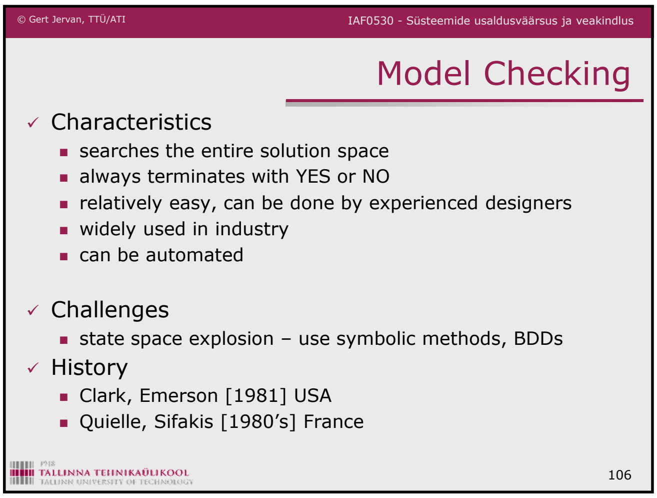
Model Checking

- Algorithmic method of verifying correctness of (finite state) concurrent systems against temporal logic specifications
  - A practical approach to formal verification
- Basic idea
  - System is described in a formal model
    - derived from high level design (HDL, C), circuit structure, etc.
  - The desired behavior is expressed as a set of properties
    - expressed as temporal logic specification
  - The specification is checked against the model



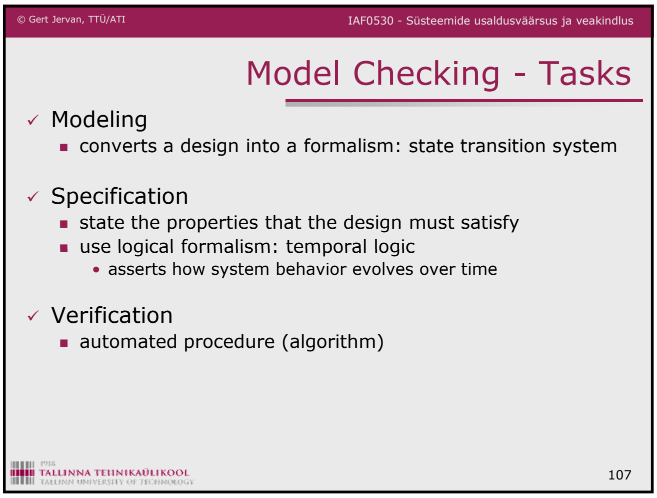
Model Checking

- How does it work
  - System is modeled as a state transition structure (Kripke structure)
  - Specification is expressed in propositional temporal logic (CTL formula)
    - asserts how system behavior evolves over time
  - Efficient search procedure checks the transition system to see if it satisfies the specification



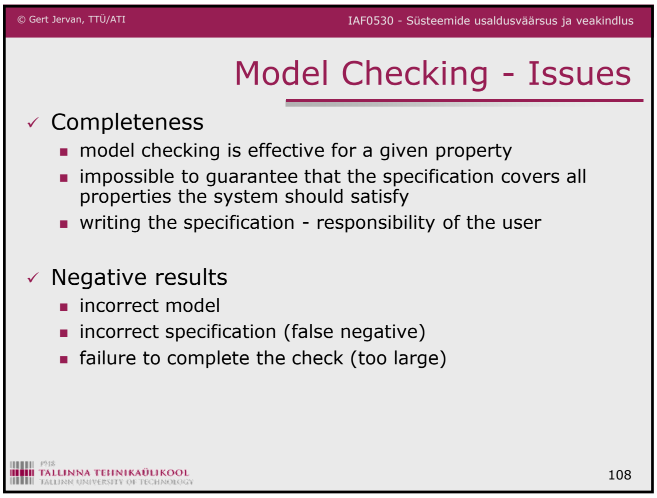
Model Checking

- Characteristics
  - searches the entire solution space
  - always terminates with YES or NO
  - relatively easy, can be done by experienced designers
  - widely used in industry
  - can be automated
- Challenges
  - state space explosion – use symbolic methods, BDDs
- History
  - Clark, Emerson [1981] USA
  - Quielle, Sifakis [1980's] France



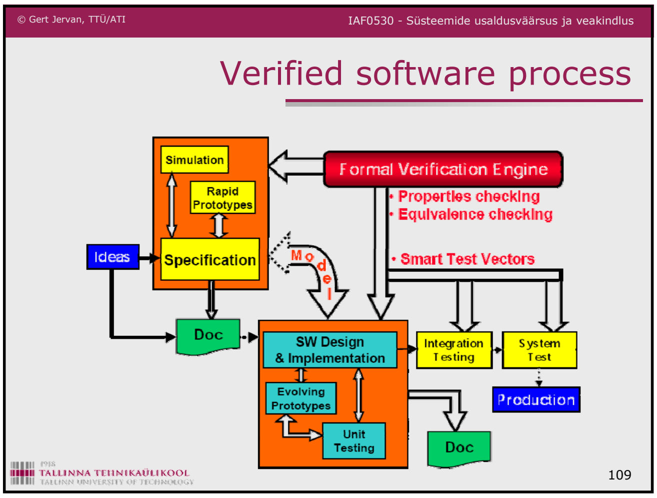
Model Checking - Tasks

- Modeling
  - converts a design into a formalism: state transition system
- Specification
  - state the properties that the design must satisfy
  - use logical formalism: temporal logic
    - asserts how system behavior evolves over time
- Verification
  - automated procedure (algorithm)



Model Checking - Issues

- Completeness
  - model checking is effective for a given property
  - impossible to guarantee that the specification covers all properties the system should satisfy
  - writing the specification - responsibility of the user
- Negative results
  - incorrect model
  - incorrect specification (false negative)
  - failure to complete the check (too large)



© Gert Jervan, TTÜ/ATI IAF0530 - Süsteemide usaldusväärsus ja veakindlus

Verified software process

Simulation

Rapid Prototypes

Formal Verification Engine

- Properties checking
- Equivalence checking

Ideas

Specification

Model

- Smart Test Vectors

Doc

SW Design & Implementation

Integration Testing

System Test

Evolving Prototypes

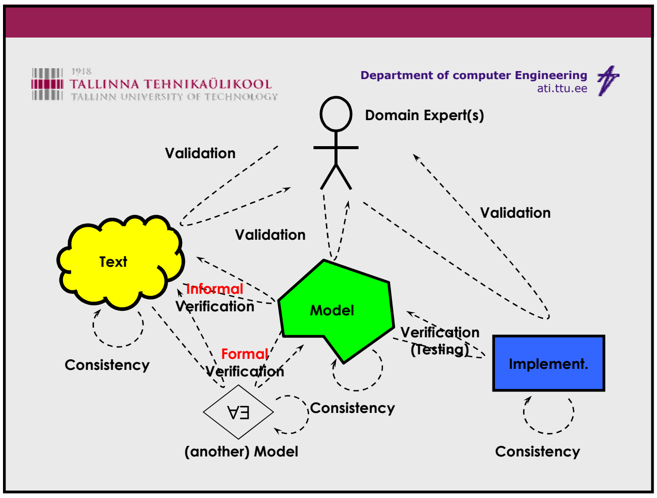
Production

Unit Testing

Doc

TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

109



1918 TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

Department of computer Engineering ati.ttu.ee

Domain Expert(s)

Validation

Validation

Validation

Text

Informal Verification

Model

Verification (Testing)

Consistency

Formal Verification

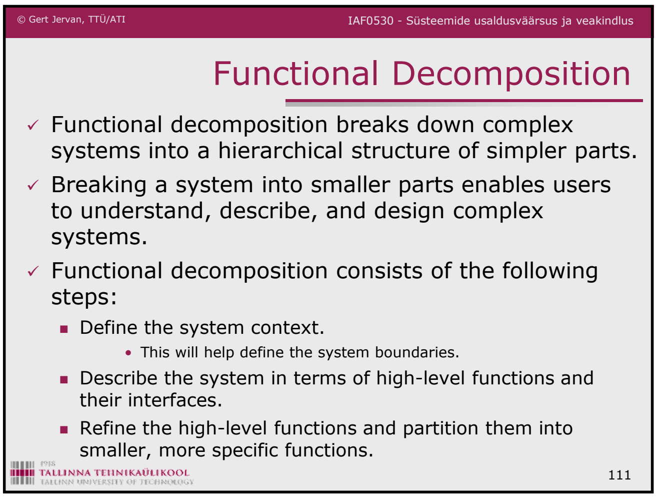
Implement.

∀∃

Consistency

(another) Model

Consistency



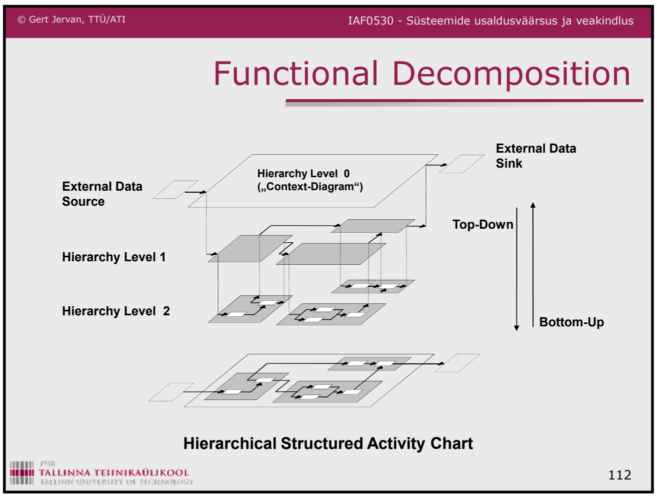
© Gert Jervan, TTÜ/ATI IAF0530 - Süsteemide usaldusväärsus ja veakindlus

Functional Decomposition

- ✓ Functional decomposition breaks down complex systems into a hierarchical structure of simpler parts.
- ✓ Breaking a system into smaller parts enables users to understand, describe, and design complex systems.
- ✓ Functional decomposition consists of the following steps:
  - Define the system context.
    - This will help define the system boundaries.
  - Describe the system in terms of high-level functions and their interfaces.
  - Refine the high-level functions and partition them into smaller, more specific functions.

TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

111



© Gert Jervan, TTÜ/ATI IAF0530 - Süsteemide usaldusväärsus ja veakindlus

Functional Decomposition

External Data Sink

External Data Source

Hierarchy Level 0 („Context-Diagram“)

Top-Down

Hierarchy Level 1

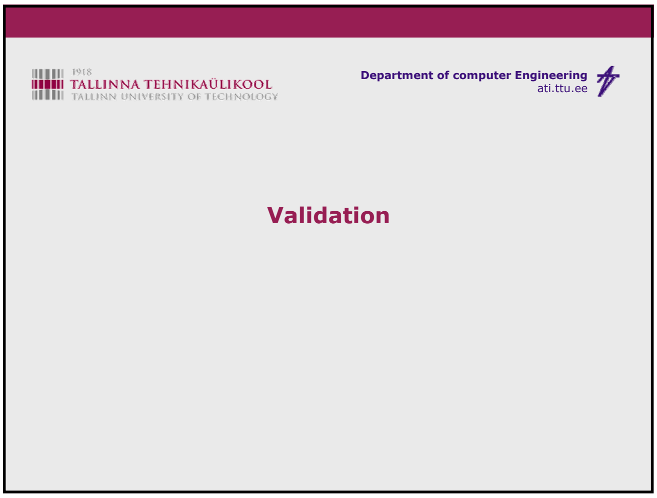
Hierarchy Level 2

Bottom-Up

Hierarchical Structured Activity Chart

TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

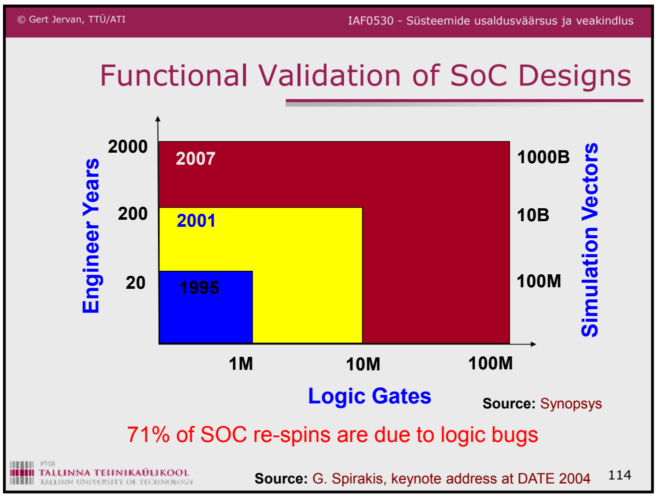
112



1918 TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

Department of computer Engineering ati.ttu.ee

Validation



© Gert Jervan, TTÜ/ATI IAF0530 - Süsteemide usaldusväärsus ja veakindlus

Functional Validation of SoC Designs

Engineer Years

2000

200

20

2007

2001

1995

1000B

10B

100M

Simulation Vectors

1M

10M

100M

Logic Gates

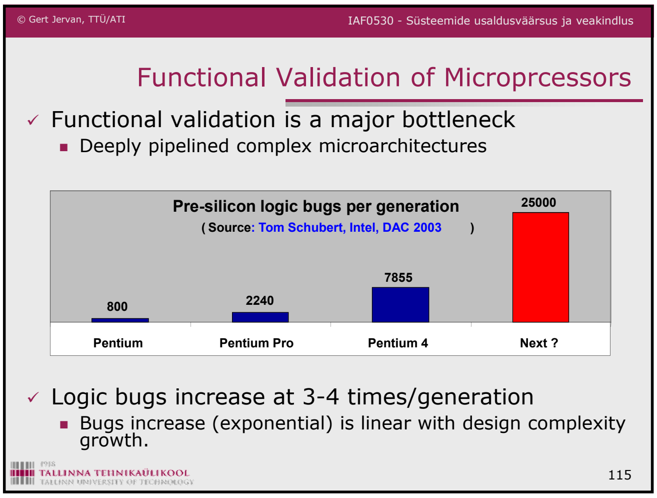
Source: Synopsys

71% of SOC re-spins are due to logic bugs

TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

Source: G. Spirakis, keynote address at DATE 2004

114



© Gert Jervan, TTÜ/ATI IAF0530 - Süsteemide usaldusväärsus ja veakindlus

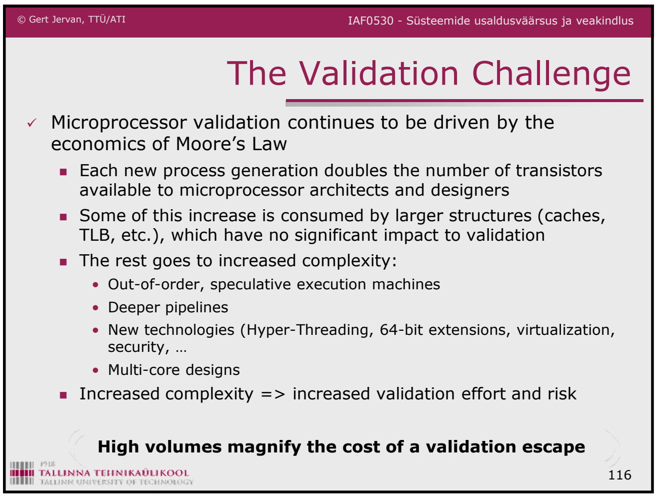
Functional Validation of Microprcessors

- ✓ Functional validation is a major bottleneck
  - Deeply pipelined complex microarchitectures

Pre-silicon logic bugs per generation
(Source: Tom Schubert, Intel, DAC 2003)

- ✓ Logic bugs increase at 3-4 times/generation
  - Bugs increase (exponential) is linear with design complexity growth.

TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY 115



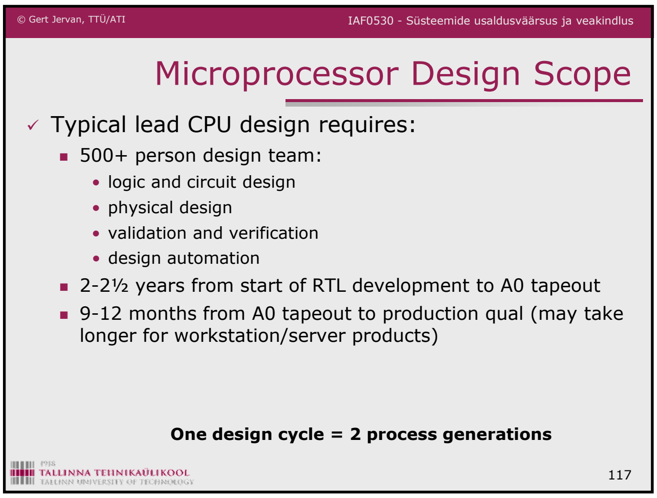
© Gert Jervan, TTÜ/ATI IAF0530 - Süsteemide usaldusväärsus ja veakindlus

The Validation Challenge

- ✓ Microprocessor validation continues to be driven by the economics of Moore's Law
  - Each new process generation doubles the number of transistors available to microprocessor architects and designers
  - Some of this increase is consumed by larger structures (caches, TLB, etc.), which have no significant impact to validation
  - The rest goes to increased complexity:
    - Out-of-order, speculative execution machines
    - Deeper pipelines
    - New technologies (Hyper-Threading, 64-bit extensions, virtualization, security, …
    - Multi-core designs
  - Increased complexity => increased validation effort and risk

High volumes magnify the cost of a validation escape

TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY 116



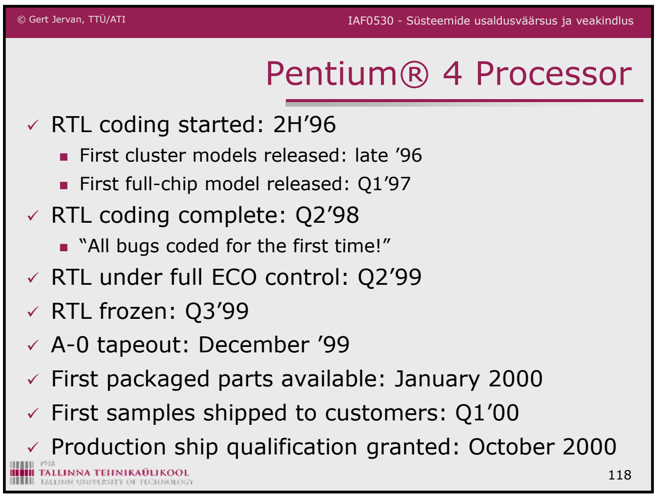
© Gert Jervan, TTÜ/ATI IAF0530 - Süsteemide usaldusväärsus ja veakindlus

Microprocessor Design Scope

- ✓ Typical lead CPU design requires:
  - 500+ person design team:
    - logic and circuit design
    - physical design
    - validation and verification
    - design automation
  - 2-2½ years from start of RTL development to A0 tapeout
  - 9-12 months from A0 tapeout to production qual (may take longer for workstation/server products)

One design cycle = 2 process generations

TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY 117

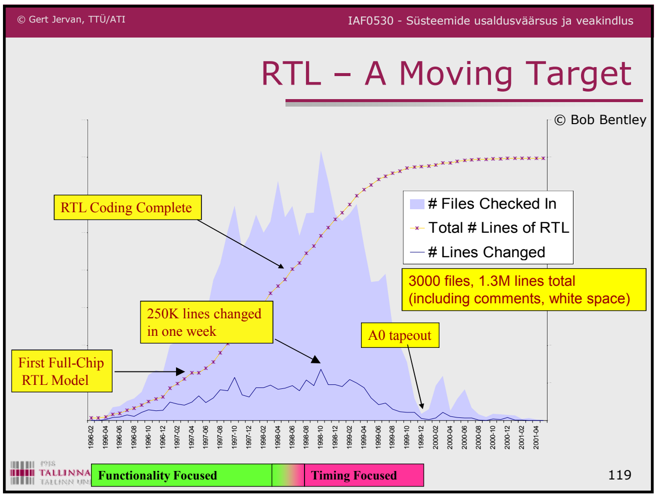


© Gert Jervan, TTÜ/ATI IAF0530 - Süsteemide usaldusväärsus ja veakindlus

Pentium® 4 Processor

- ✓ RTL coding started: 2H'96
  - First cluster models released: late '96
  - First full-chip model released: Q1'97
- ✓ RTL coding complete: Q2'98
  - "All bugs coded for the first time!"
- ✓ RTL under full ECO control: Q2'99
- ✓ RTL frozen: Q3'99
- ✓ A-0 tapeout: December '99
- ✓ First packaged parts available: January 2000
- ✓ First samples shipped to customers: Q1'00
- ✓ Production ship qualification granted: October 2000

TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY 118

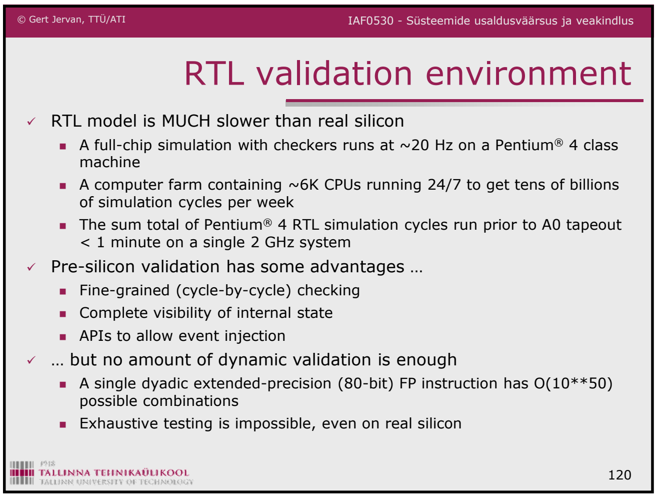


© Gert Jervan, TTÜ/ATI IAF0530 - Süsteemide usaldusväärsus ja veakindlus

RTL – A Moving Target

© Bob Bentley

TALLINNA TEHNIKAÜLIKOOL 119



© Gert Jervan, TTÜ/ATI IAF0530 - Süsteemide usaldusväärsus ja veakindlus

RTL validation environment

- ✓ RTL model is MUCH slower than real silicon
  - A full-chip simulation with checkers runs at ~20 Hz on a Pentium® 4 class machine
  - A computer farm containing ~6K CPUs running 24/7 to get tens of billions of simulation cycles per week
  - The sum total of Pentium® 4 RTL simulation cycles run prior to A0 tapeout < 1 minute on a single 2 GHz system
- ✓ Pre-silicon validation has some advantages …
  - Fine-grained (cycle-by-cycle) checking
  - Complete visibility of internal state
  - APIs to allow event injection
- ✓ … but no amount of dynamic validation is enough
  - A single dyadic extended-precision (80-bit) FP instruction has O(10**50) possible combinations
  - Exhaustive testing is impossible, even on real silicon

TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY 120

How do you verify a design with...

- ✓ 42 million transistors
- ✓ 1 million lines of RTL code
- ✓ 600 – 1000 people working on it
- ✓ A 3-year design time
- ✓ Daily design changes

How do you verify a design which has bugs like this??

- ✓ The FMUL instruction, when the rounding mode is set to “round up”, incorrectly sets the sticky bit when the source operands are:

  src1[67:0] = X*2i+15 + 1*2i

  src2[67:0] = Y*2j+15 + 1*2j

  where i+j = 54 and {X,Y} are integers

And the answer is...

- ✓ Hire 70+ validation engineers
- ✓ Buy several thousand compute servers
- ✓ Write 12,000 validation tests
- ✓ Run up to 1 billion simulation cycles per day for 200 days
- ✓ Check 2,750,000 manually-defined properties
- ✓ Find, diagnose, track, and resolve 7,855 bugs
- ✓ Apply formal verification with 10,000 proofs to the instruction decoder and FP units
  - ▪ This found that obscure FMUL bug!

Pentium 4 Validation - Staffing

- ✓ 10 people in initial “nucleus” from previous project
- ✓ 40 new hires in 1997
- ✓ 20 new hires in 1998

P4 Validation Environment

- ✓ Hardware
  - ▪ IBM RS/6000 workstations (0.5-0.6Hz full processor model)
  - ▪ Pentium III Linux systems (3-5Hz full processor model)
  - ▪ Computing pool of “several thousand” systems
- ✓ Simulation statistics
  - ▪ About 1 million lines of code in SRTL model
  - ▪ 5-6 billion clock cycles simulated / week
  - ▪ 200 billion total clock cycles simulated overall

About 2 minutes of execution with a 1GHz clock!

Cluster-Level Testing

- ✓ Divide overall design into 6 “clusters” + microcode
  - ▪ Develop “cluster testing environments” (CTEs) to validate each cluster separately (e.g. floating point, memory)
  - ▪ Then validate using full processor model
- ✓ Advantages of the approach
  - ▪ Controllability - control behavior at microarchitecture level
  - ▪ Early validation possible for each cluster
  - ▪ Decoupled validation possible for each cluster

Other Validation Features

- ✓ Extensive validation of power-reduction logic
- ✓ Code coverage and code inspections a major part of methodology
- ✓ Formal verification used for Floating Point & Instruction Decode Logic

Power Reduction Validation

- ✓ Power consumption was a big concern for Pentium 4
  - Need to stay within the cost-effective thermal envelope for desktop systems at 1.5+ GHz
- ✓ Extensive clock gating in every part of the design
- ✓ Mounted a focused effort to validate that:
  - Committed features were implemented as per plan
  - Functional correctness was maintained in the face of clock gating
  - Changes to the design did not impact power savings
- ✓ ~12 person years of effort, 5 heads at peak
- ✓ Fully functional on A-step silicon, measured savings of ~20W achieved for typical workloads

Formal Verification in P4 Validation

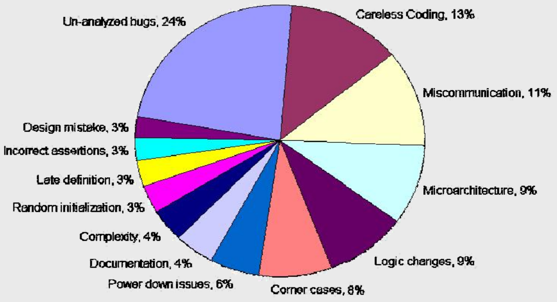
- ✓ Based on model checking
  - Given a finite-state concurrent system
  - Express specifications as temporal logic formulas
  - Use symbolic algorithms to check whether model holds
- ✓ Constructed database 10,000 "proofs"
- ✓ Over 100 bugs found
- ✓ 20 were "high quality" bugs not likely to be found by simulation
- ✓ Example errors: FADD, FMUL

Validation Results

- ✓ 5809 bugs identified by simulation
  - 3411 bugs found by cluster-level testing
  - 2398 found using full-chip model
- ✓ 1554 bugs found by code inspection
- ✓ 492 bugs found by formal verification
- ✓ Largest sources of bugs: memory cluster (25%)

Pentium® 4 Bugs Breakdown

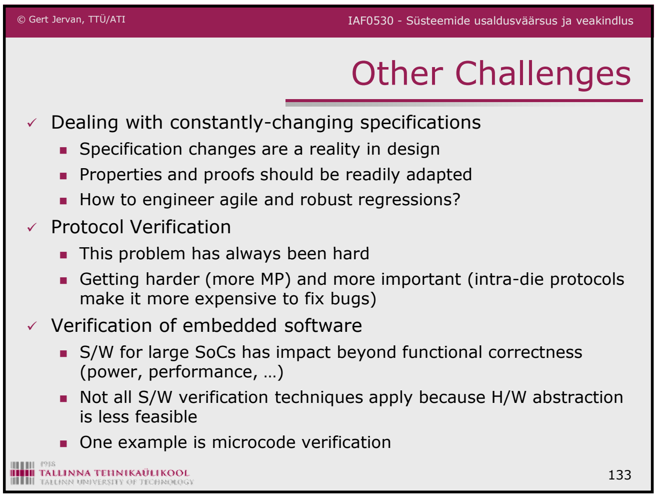
Source: Bob Bentley, HLDVT 2002



Micro-architectural complexity is a major contributor

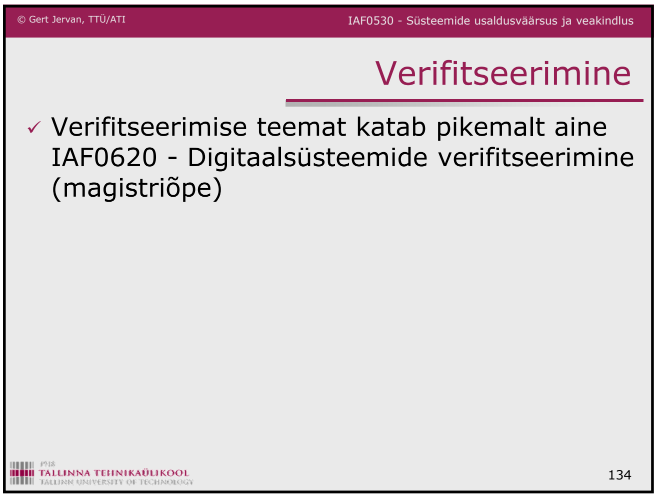
Methodology drivers

- ✓ Regression
  - RTL is "live", and changes frequently until the very last stages of the project
  - Model checking automation at lower levels allows regression to be automated and provides robustness in the face of ECOs
- ✓ Debugging
  - Need to be able to demonstrate FV counter-examples to designers and architects
  - Designers want a dynamic test that they can simulate
  - Waveform viewers, schematic browsers, etc. can help to bridge the gap
- ✓ Verification in the large
  - Proof design: how do we approach the problem in a systematic fashion?
  - Proof engineering: how do we write maintainable and modifiable proofs?



Other Challenges

- ✓ Dealing with constantly-changing specifications
  - Specification changes are a reality in design
  - Properties and proofs should be readily adapted
  - How to engineer agile and robust regressions?
- ✓ Protocol Verification
  - This problem has always been hard
  - Getting harder (more MP) and more important (intra-die protocols make it more expensive to fix bugs)
- ✓ Verification of embedded software
  - S/W for large SoCs has impact beyond functional correctness (power, performance, …)
  - Not all S/W verification techniques apply because H/W abstraction is less feasible
  - One example is microcode verification



Verifitseerimine

- ✓ Verifitseerimise teemat katab pikemalt aine IAF0620 - Digitaalsüsteemide verifitseerimine (magistriõpe)

1918 TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

Department of computer Engineering
ati.ttu.ee

Questions?

Gert Jervan

Tallinna Tehnikaülikool
Arvutitehnika instituut