Embedded Systems

Embedded Systems

- Christian Steger (steger@cs.uni-saarland.de)
- Reinhard Wilhelm (wilhelm@cs.uni-saarland.de), Head of "Compiler Design Lab"
- Sebastian Altmeyer (altmeyer@cs.uni-saarland.de)
- Hans-Jörg Peter (peter@cs.uni-saarland.de)
- Michael Gerke (gerke@cs.uni-saarland.de)

- Lectures:
  - Tuesday 10:15 - 11:45
  - Thursday 14:15 -15:45

Midterm, Thursday December 16, 2010, 16-19
End-of-term, Monday February 14, 2011, 14-17
End-of-semester: tba

Embedded Systems

Registration through **HISPOS** (if HISPOS is not applicable – Non-CS, Erasmus, etc – send email to peter@cs.uni-saarland.de)

- Webpage

http://react.cs.uni-sb.de/teaching/embedded-systems-10-11

The course mailing list is available now. Subscribe to get notifications and the latest information: https://alan.cs.uni-saarland.de/cgi-bin/mailman/listinfo/es

- Tutorials

  Wednesday, 16:00-18:00, SR 107, E 1 3
  Friday, 12:00-14:00, SR 015, E 1 3
  Friday, 14:00-16:00, SR 016, E 1 3

Please indicate tutorial, matr nr, name, e-mail on **homework submissions.**

Embedded Systems

- **Reactive Systems Group at Saarland University**
  (http://react.cs.uni-sb.de/) – Head Bernd Finkbeiner

  The research concerns computer-aided methods for the synthesis and verification of reactive systems.

Current Project

The goal of the AVACS project is to raise the state of the art in automatic verification and analysis techniques from its current level, where it is applicable only to isolated facets (concurrency, time, continuous control, stability, dependability, mobility, data structures, hardware constraints, modularity, levels of refinement), to a level allowing the comprehensive verification of computer systems

Embedded Systems

- **Compiler Design Lab at Saarland University**

  (http://rw4.cs.uni-saarland.de//) – Head Reinhard Wilhelm

  Research at the chair is concerned with the analysis and predictability of real time systems. In this process, various techniques like static analysis and abstract interpretation are applied to improve the safeness of embedded applications. The current research focuses on cache predictability, design for timing predictability, improvements of timing analysis and semi-automatic derivation of abstract processor models.

AbsInt
Angewandte Informatik GmbH

- AbsInt (www.absint.com) provides advanced development tools for embedded systems, and tools for validation, verification and certification of safety-critical software.

Embedded Systems

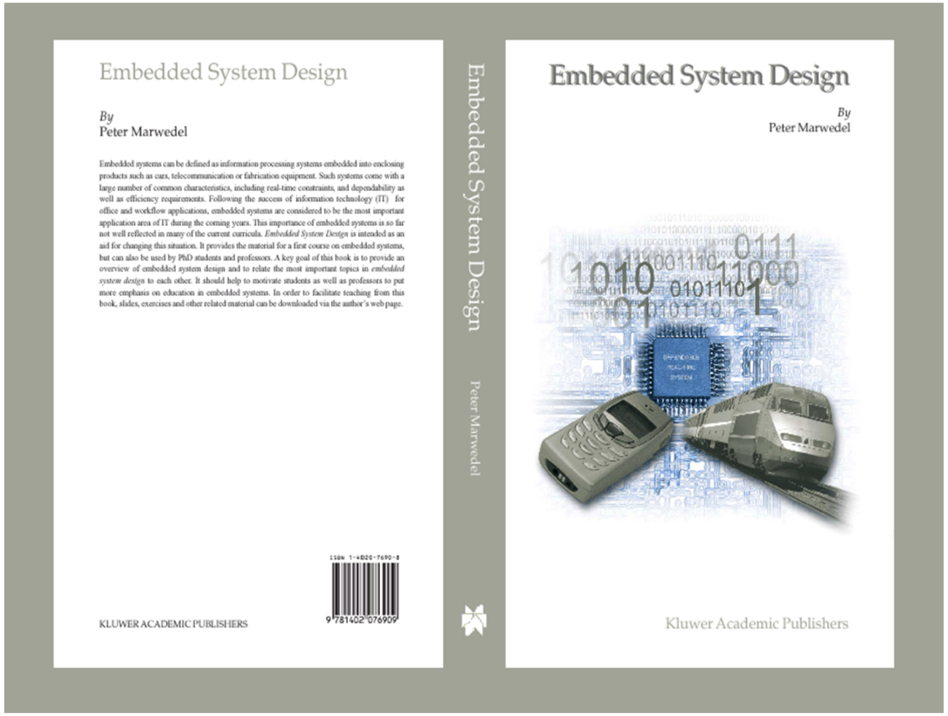
Christian Steger, Head of Working group HW/SW Codesign@ITI, www.iti.tugraz.at/codesign , TU Graz

- Topics:
  - HW/SW codesign (Cosimulation)
  - Model based design
  - Rapid prototyping (HIL/MIL)
  - Power awareness
  - Target application in identifikation (RFID, Smart Cards), automation, automotive und Ad-hoc wireless sensor networks
- Lectures:
  - Hardware Description Languages (VHDL, VHDL-AMS, SystemC)
  - HW/SW Codesign
  - Power Aware Computing

- Industry partners:

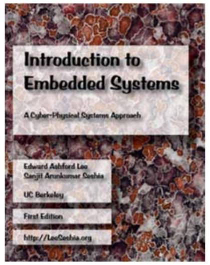
  Austriamicrosystems, AVL, CISC Semiconductor Design + Consulting GmbH, Infineon, Innofreight GmbH, IMEC Eindhoven, IVM, Neosera Ldt. (Dublin), NXP, RF-iT Solutions GmbH, Salomon Automation, Siemens, Comet K2 Competence Center ViF;

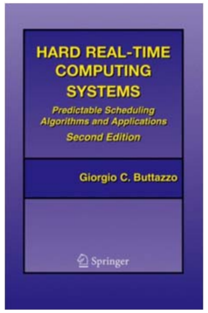
Textbook

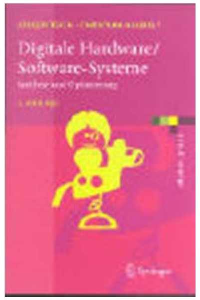


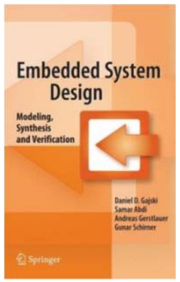
- Peter Marwedel.
  Embedded System Design.
  Springer, Berlin;
  2nd Print
  (1. November 2005)
  ISBN-10: 0387292373

Other Recommended Literature









- Edward Ashford Lee (October 2010)
  Sanjit A. Seshia
  Introduction to Embedded System -
  A Cyber-Physical Systems Approach
  free download: http://leeseshia.org/

- Giorgio C. Buttazzo
  Hard Real-Time Computing
  Systems

- Jürgen Teich,
  Digitale Hardware/Software
  Systeme

- Heinz Wörn,
  Uwe Brinkschulte,
  Echtzeitsysteme

- Gajski, Daniel D. (2010)
  Embedded System Design

Exam Policy

- Midterm/End-of-Term Exam/End-of-Semester Exam

Requirement for admission to end-of-term and end-of-semester exams:

- > 50% of points in problem sets,
- > 50% of points in each project milestone, and
- > 50% of points in midterm exam

- **Final grade:**
- best grade in end-of-term or end-of-semester exam

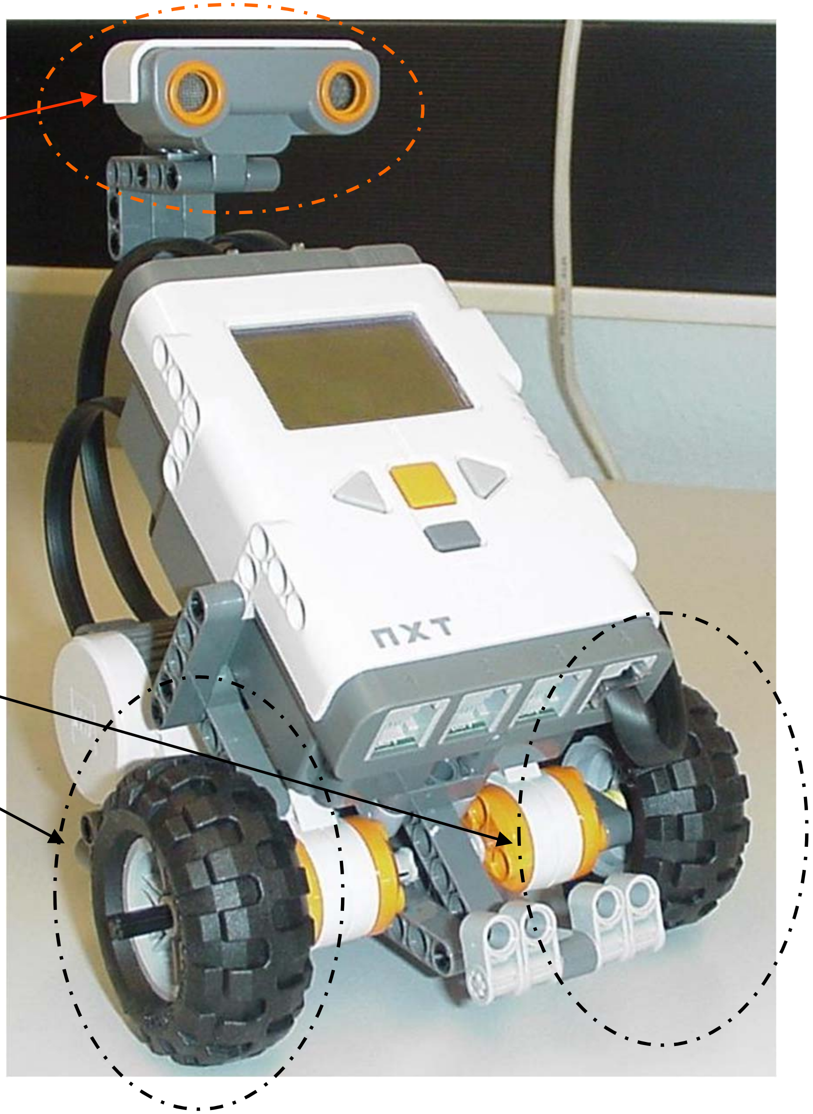
Project Lego Mindstorm Roboter

- Goal
  - Apply state-of-the-art design flow for embedded systems
  - Model-based code generation, task scheduling, worst-case execution time analysis
- Mission
  - Robot navigates completely autonomously
  - Identifies objects via RFID
  - searches for specific objects

- Tools/OS
  - SCADE: CASE tool for safety-critical embedded systems
  - a3 WCET Analyzer: Worst-case execution time analysis
  - leJOS – JAVA for Mindstorm roboter

Basic robot for lab

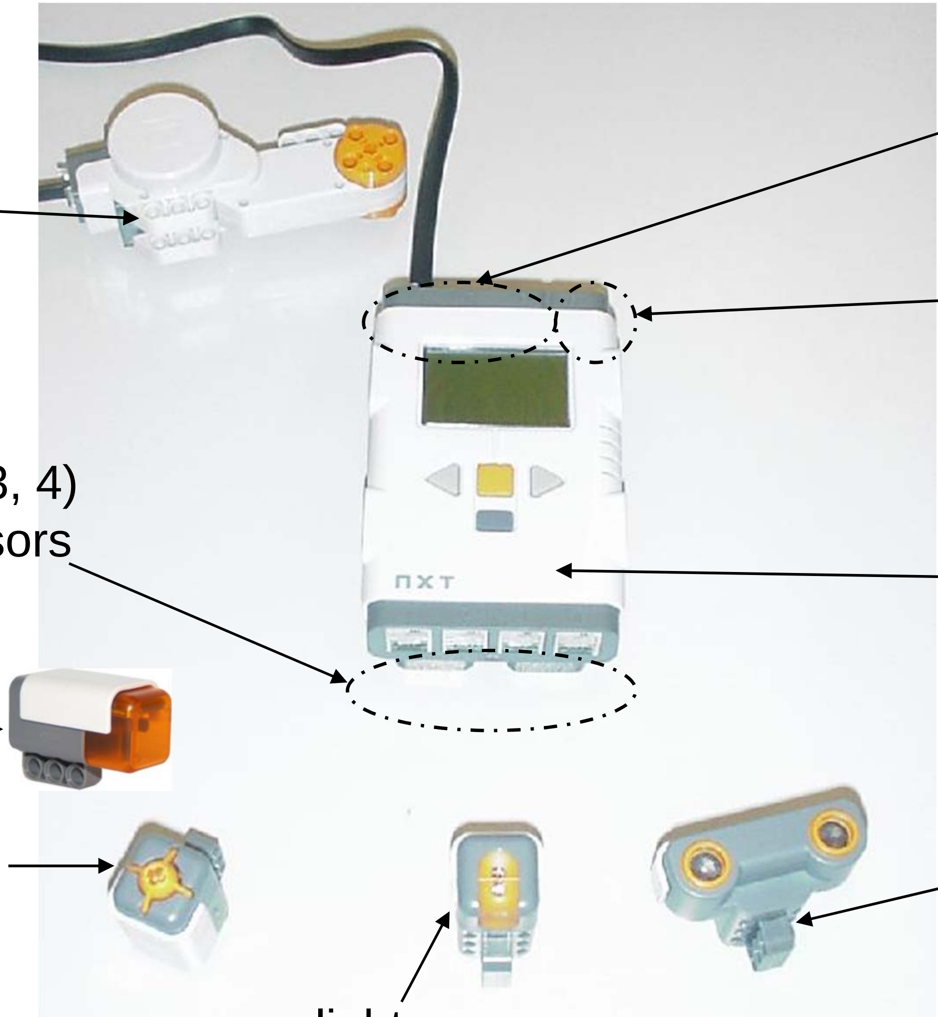
1 ultra sonic sensor



2 independently controlled wheels

The basic robot will be extended by additional sensors and actuators during the labs

Lego Mindstorm® components



motor

3 output ports (A, B, C)

1 USB port for software upload

4 input ports (1, 2, 3, 4) for connecting sensors

Lego NXT control unit

RIFD sensor

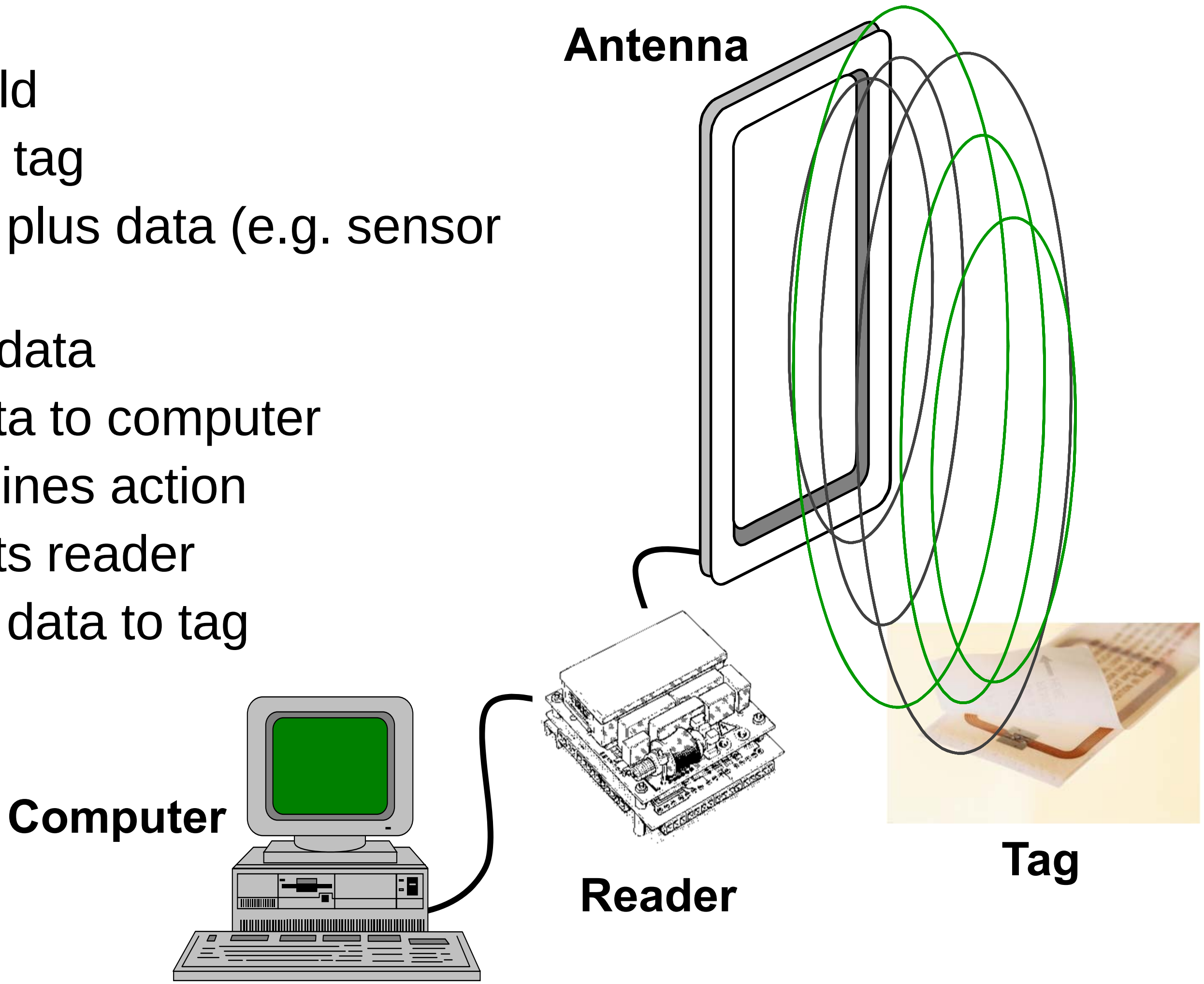
collision sensor

light sensor

Ultra sonic Sensor

What is RFID about?

- Tag enters RF field
- RF signal powers tag
- Tag transmits ID, plus data (e.g. sensor data)
- Reader captures data
- Reader sends data to computer
- Computer determines action
- Computer instructs reader
- Reader transmits data to tag



Lecture Outline

- Introduction into ES
- Specification (MOC)
- Hardware/System description languages
- Embedded operating systems/scheduling /analysis /WCET (Reinhard Wilhelm)
  Termine: 2.11, 4.11, 18.11, 2.12 (SCADE Einführung Daniel Kästner, AbsInt), 18.01, 20.01, 3.02
- Embedded system hardware
- Hardware/software codesign
- Fault tolerant ES, test and formal verification)

Introduction into ES

- **Examples of ES**
- **Complexity and Heterogeneity**
- **Characteristics of Embedded Systems**

Embedded Systems

Embedded system =
system embedded into a large (technical) product which controls the larger system or provides information processing for it.



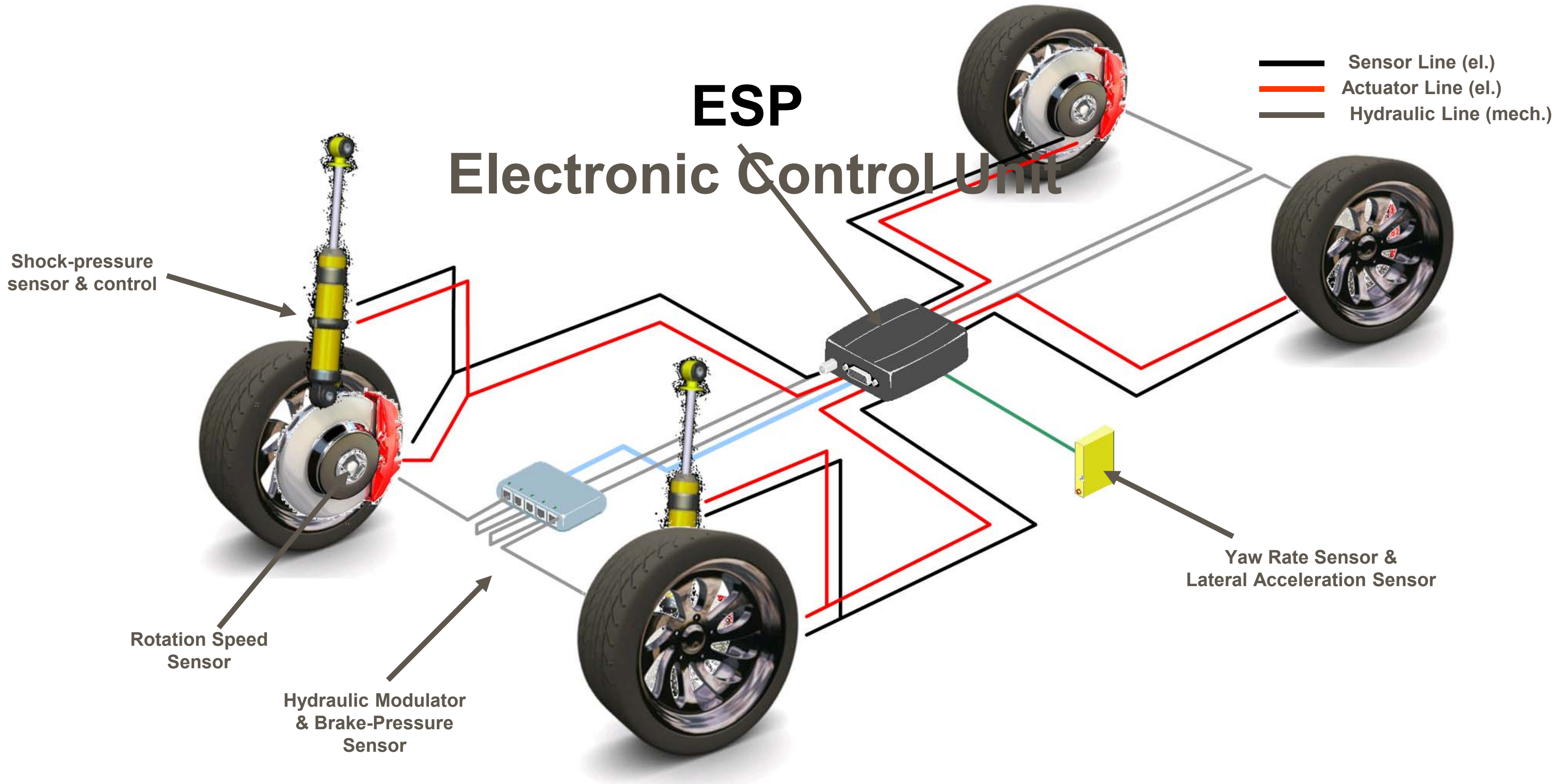





More Examples...

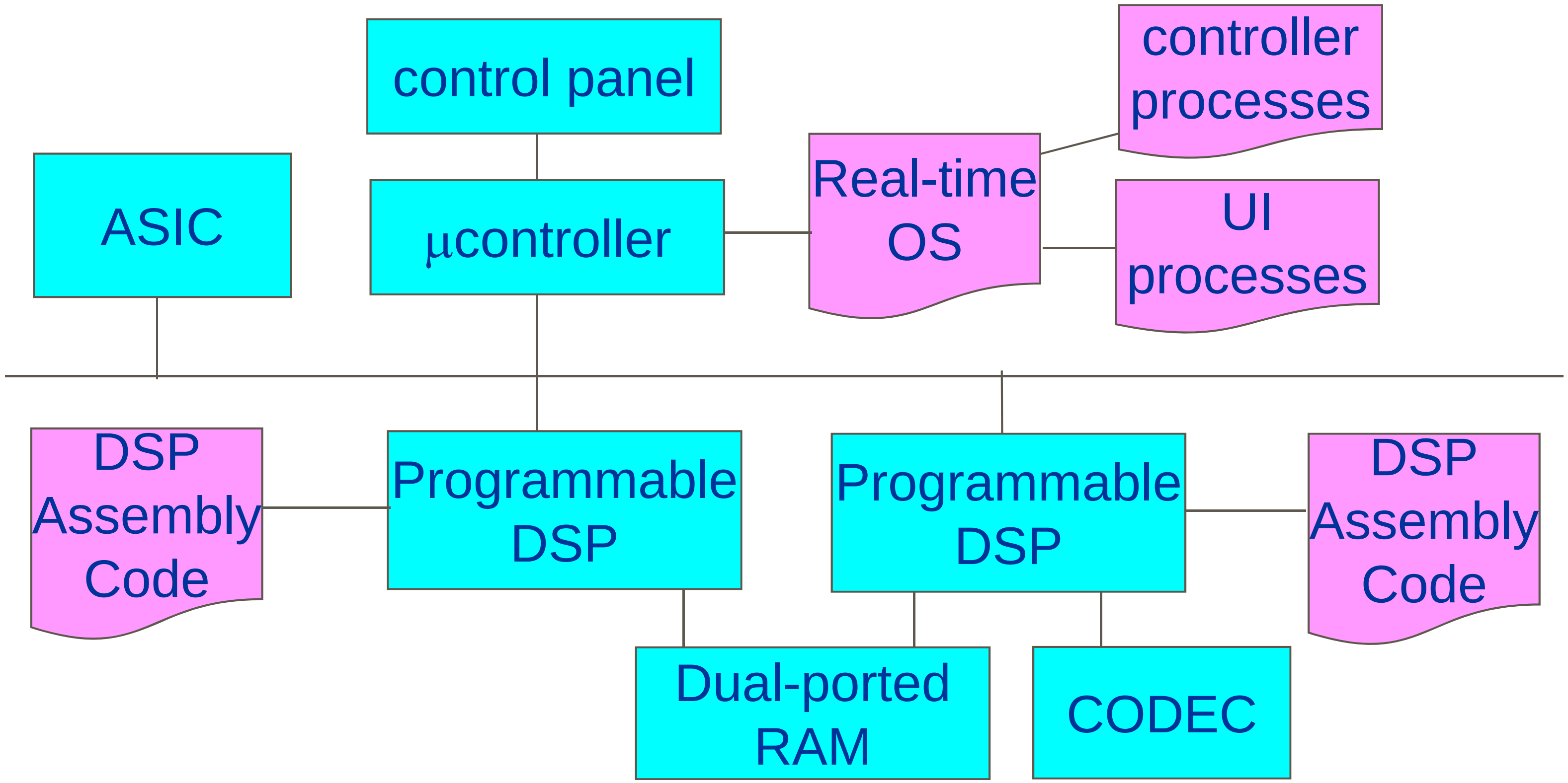
- "Small" systems
  - cellular phones, pagers, home appliances, toys, smart cards, MP3 players, PDAs, digital cameras and camcorders, sensors, smart cards
- Control oriented applications
  - network routers & switches, mass transit systems, elevators in large buildings
- Signal processing systems
  - radar, sonar, video, set-top boxes, DVD players, medical equipment
- Mission critical systems
  - avionics, space-craft control, nuclear plant control

Heterogeneous Embedded Systems



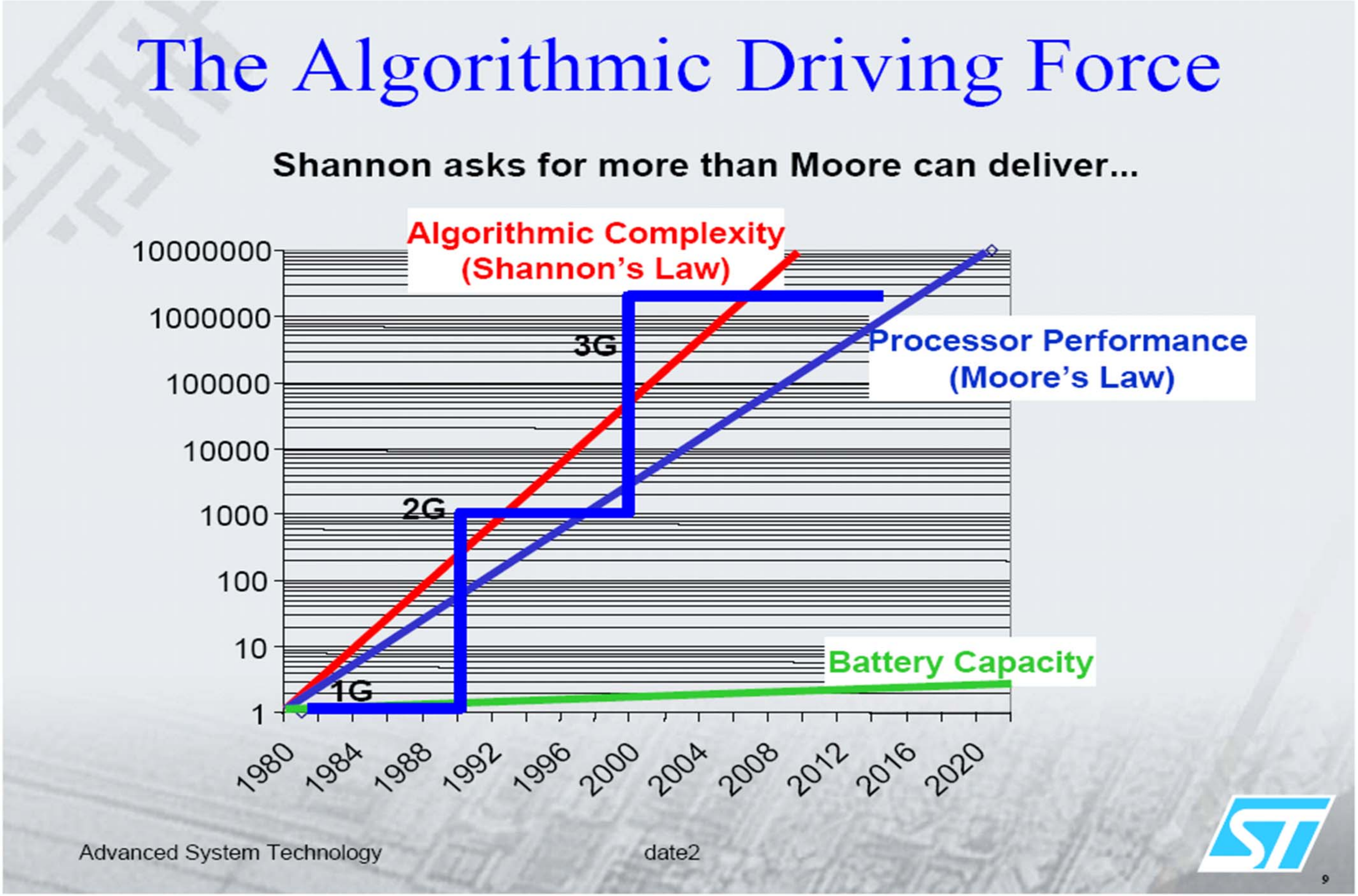
[http://www2.mercedes-benz.co.uk]

Complexity and Heterogeneity



- Heterogeneity within HW & SW parts as well
  - SW: control oriented, DSP oriented
  - HW: FPGAs, ASICs, COTS ICs

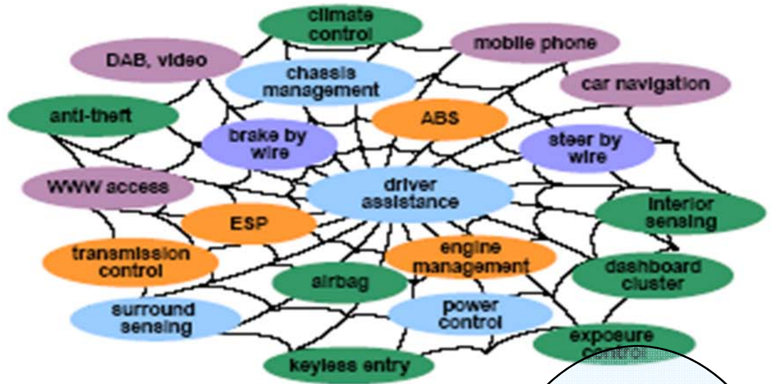
Slide of Mani Srivastava®

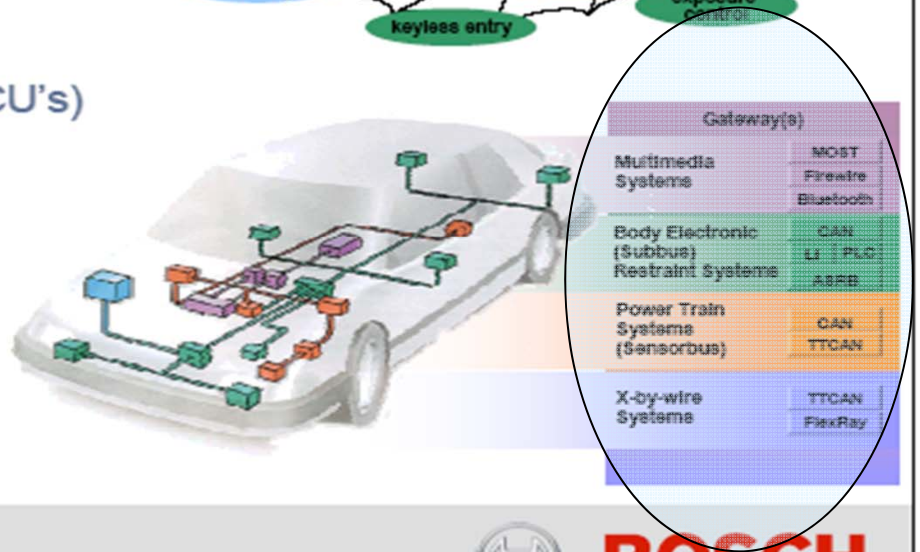

The Algorithmic Driving Force
Shannon asks for more than Moore can deliver...
Algorithmic Complexity
(Shannon's Law)
Processor Performance
(Moore's Law)
Battery Capacity
1G
2G
3G
10000000
1000000
100000
10000
1000
100
10
1
1980
1984
1988
1992
1996
2000
2004
2008
2012
2016
2020
Advanced System Technology
date2
9

Complexity

Automotive Networks

- System functionality is distributed over a network of spatially distributed sensors, actuators and processing units
  - Communicating via busses
  - Sharing a limited number of busses
- High correlation of system functionalities to system components
  - Implementation of new functionality by adding of new system components
- Number of electronic control units (ECU's) is permanently increasing
  - BMW 7 series: up to 80 ECU's
- Integration of system components and side effects on the network are key problems





Automotive Electronics

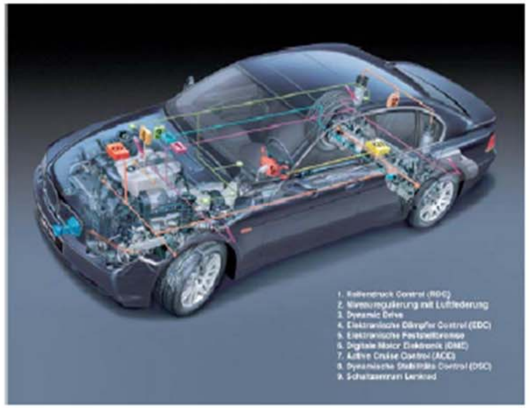
3 AE/EIM3-Gerlach | 05/01/2006 |

BOSCH

21

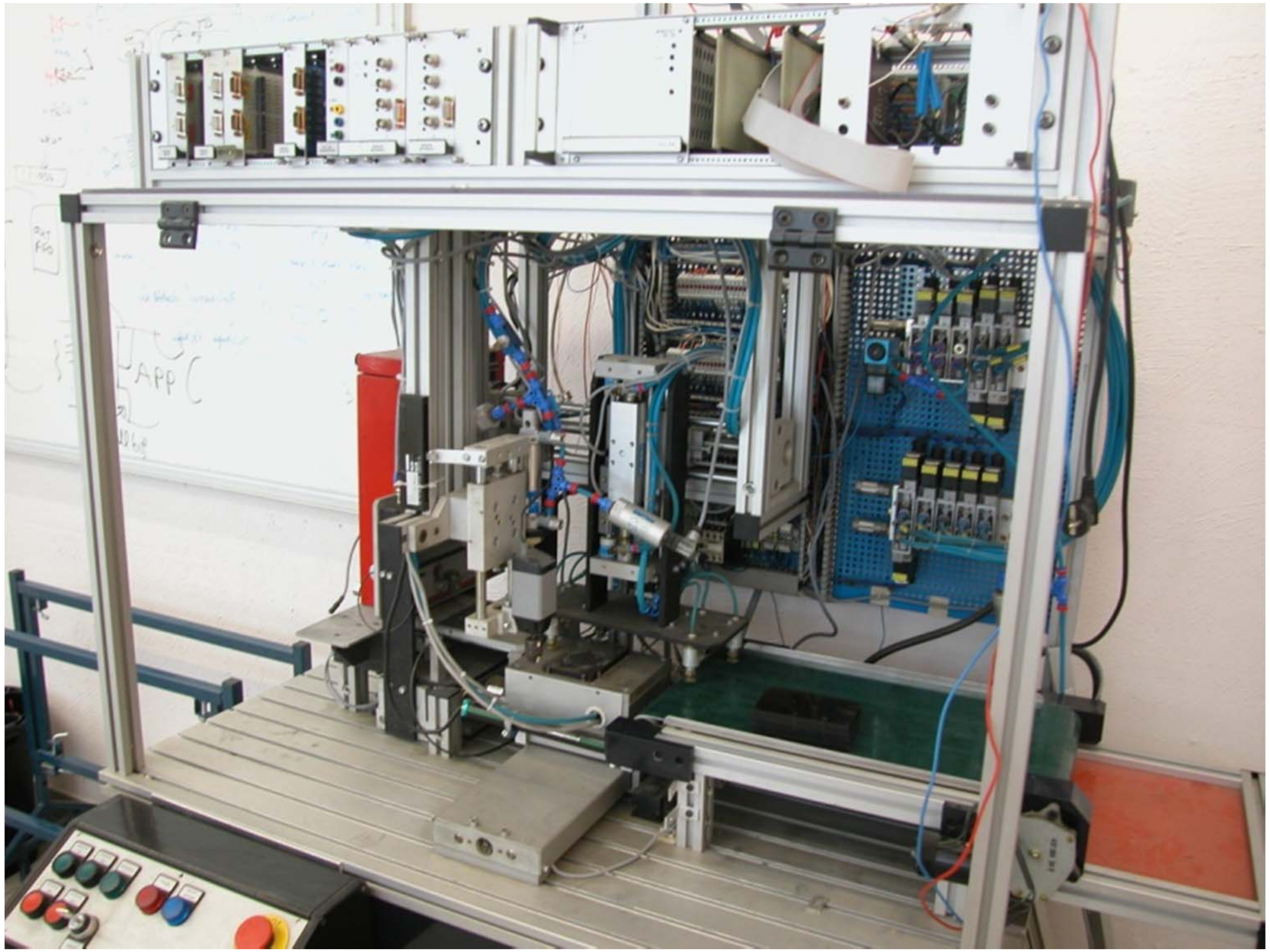
Automotive Software: An Emerging Domain
A Software Perspective

- Up to 40% of the vehicles' costs are determined by electronics and software
- 90% of all innovations are driven by electronics and software
- 50 – 70% of the development costs for an ECU are related to software
- Premium cars have up to 70 ECUs, connected by 5 system busses

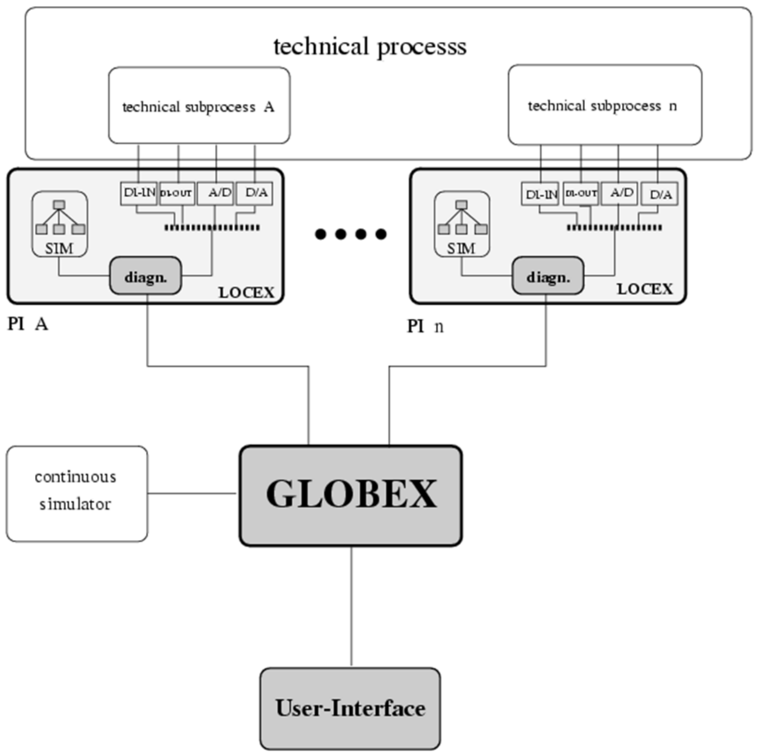


- **Growing system complexity**
- **More dependencies**
- **Costs play significant role**

Distributed Expert Systems @ ITI



Distributed Expert Systems @ ITI

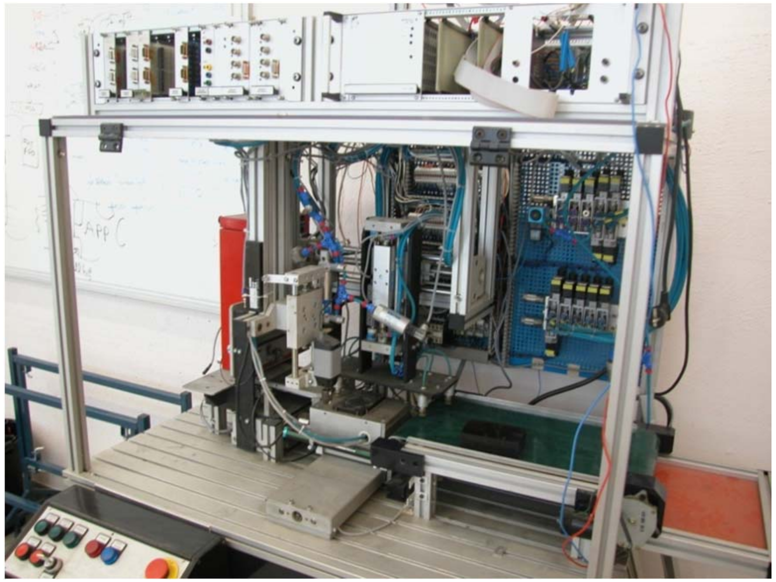


PI ....Process-Interface

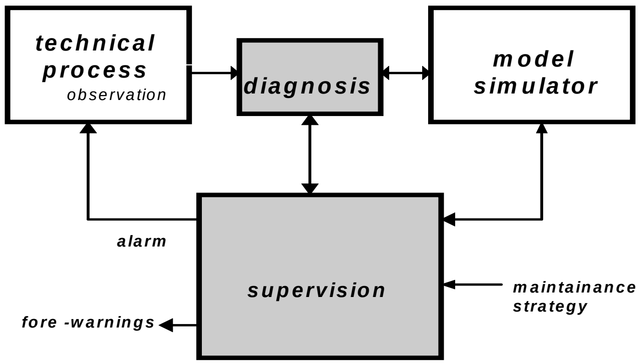
GLOBEX.........Global Expert

LOCEX..........Local Expert

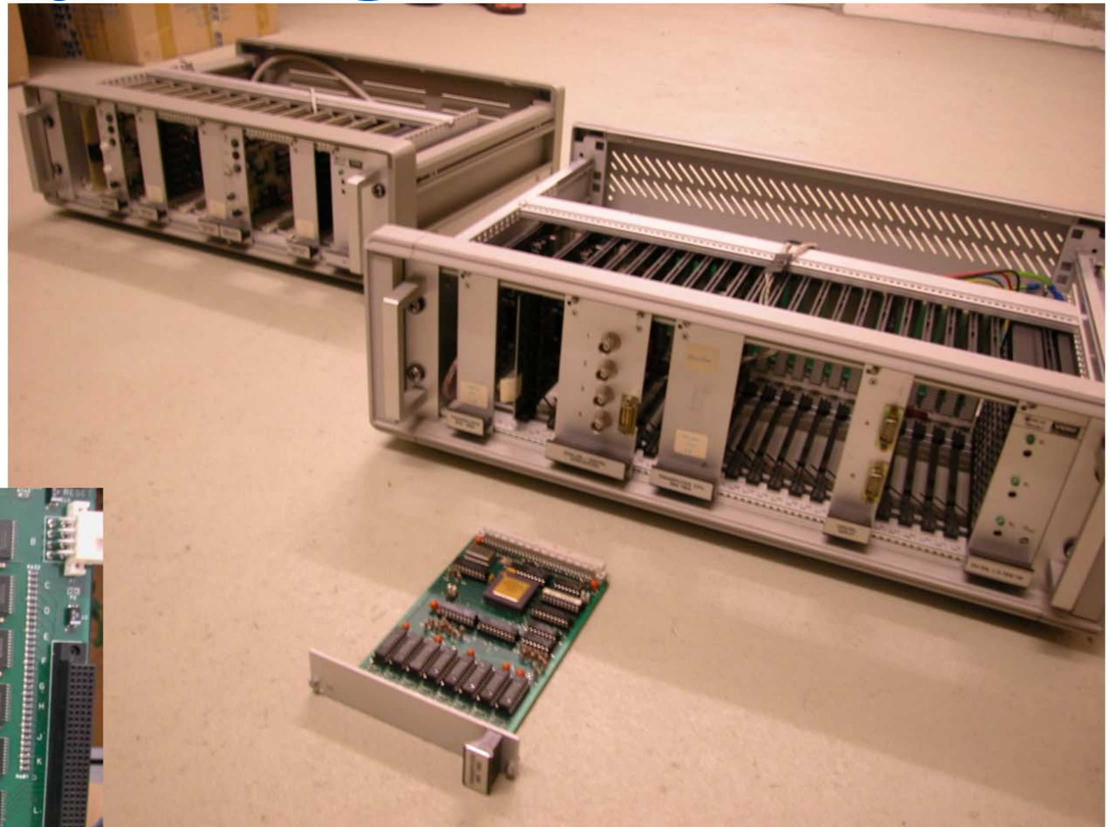
SIM ....Petri-Net Simulator

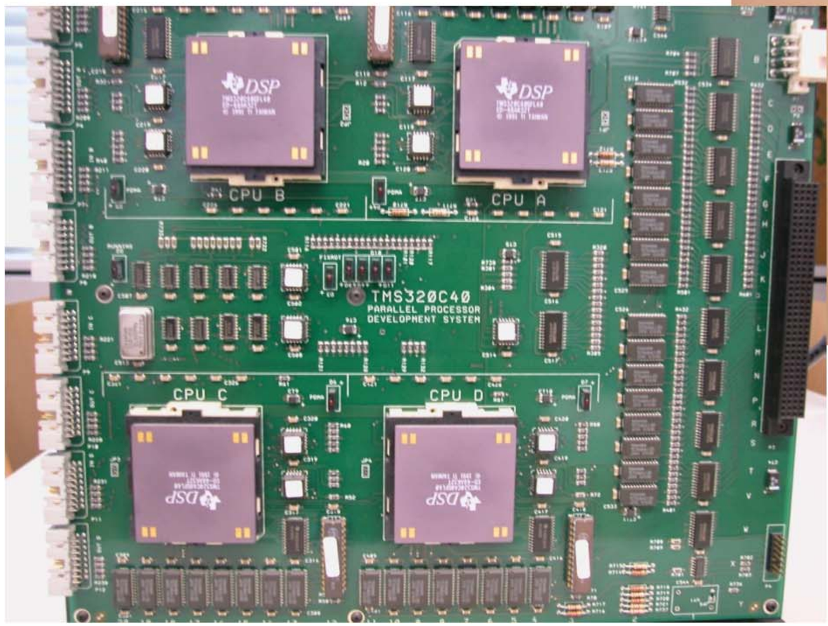


- A Distributed Real-Time Expert System for Model-Based Fault Diagnosis in Modular Production Systems



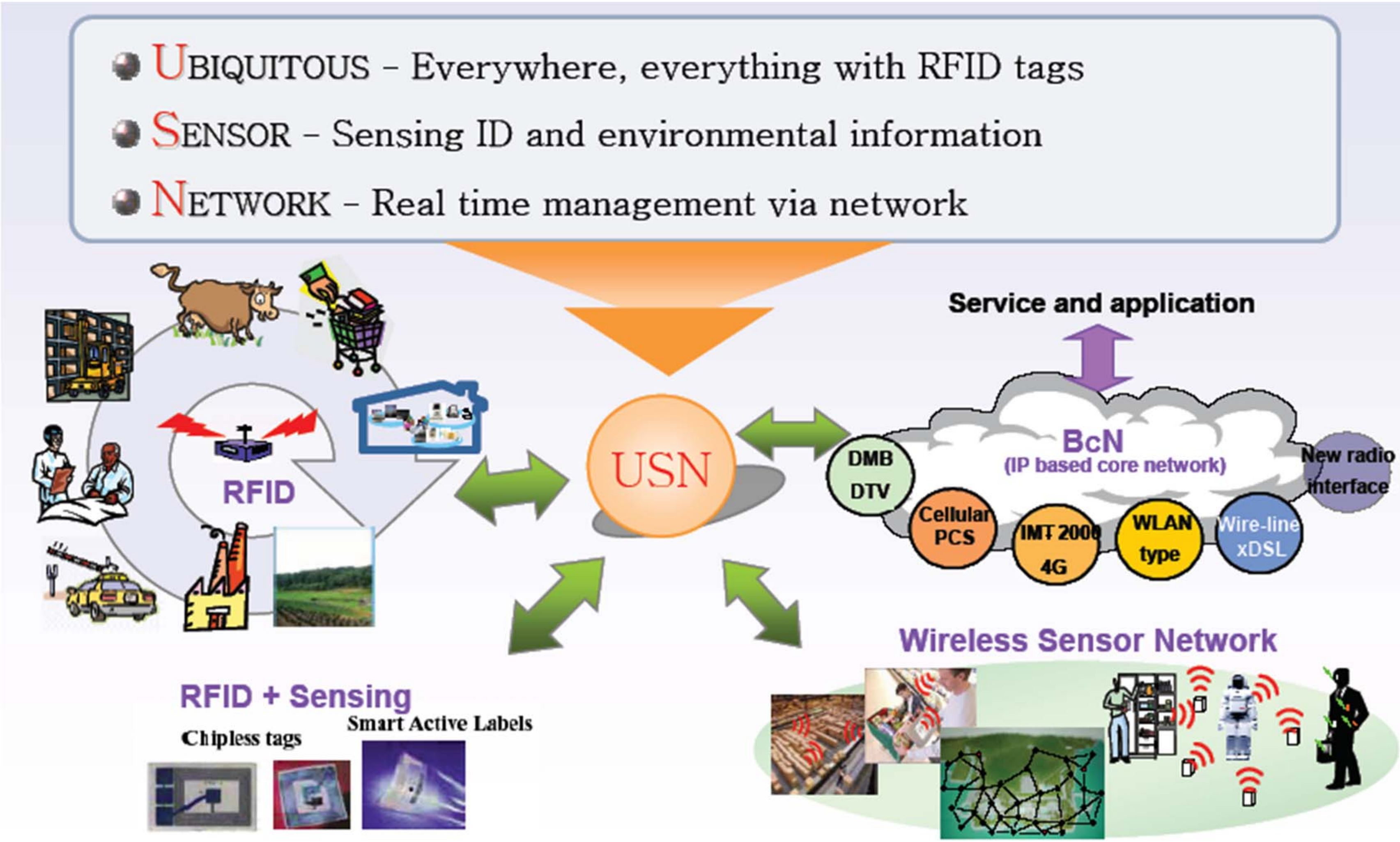
Distributed Expert Systems @ ITI





Distributed Systems

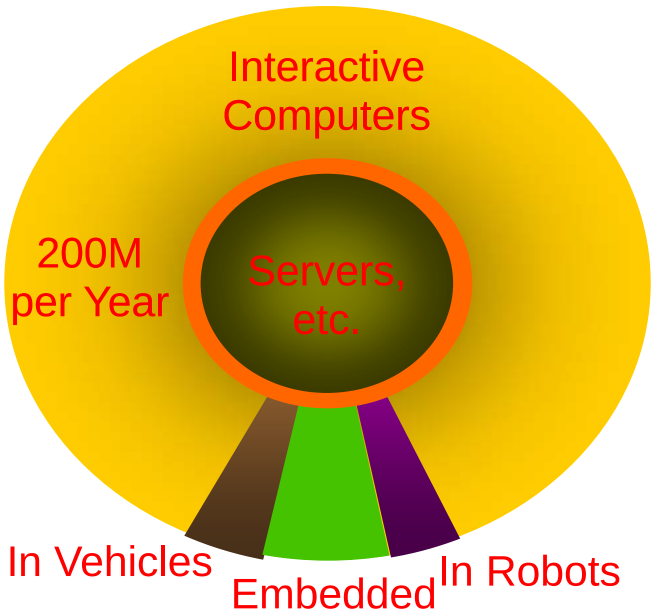
[ETRI, RFID/USN Researc
Group, Korea]



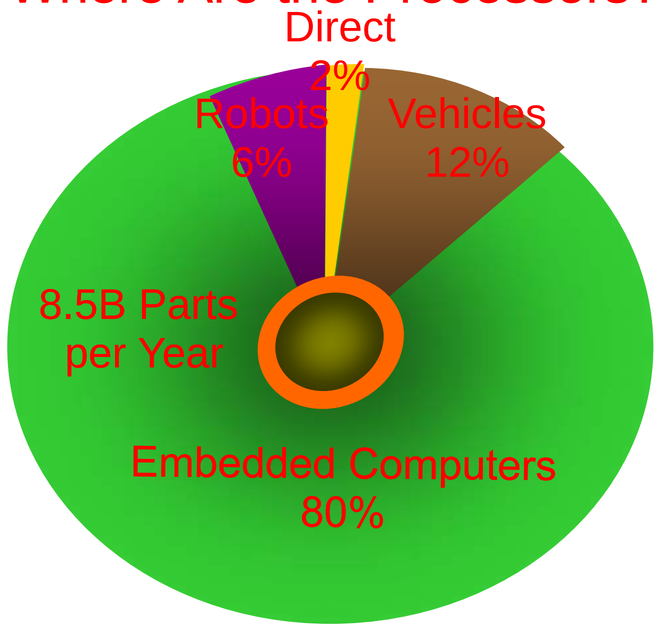
Where are the CPUs?

Estimated 98% of 8 Billion CPUs produced in 2000 used for embedded apps

Where Has CS Focused?



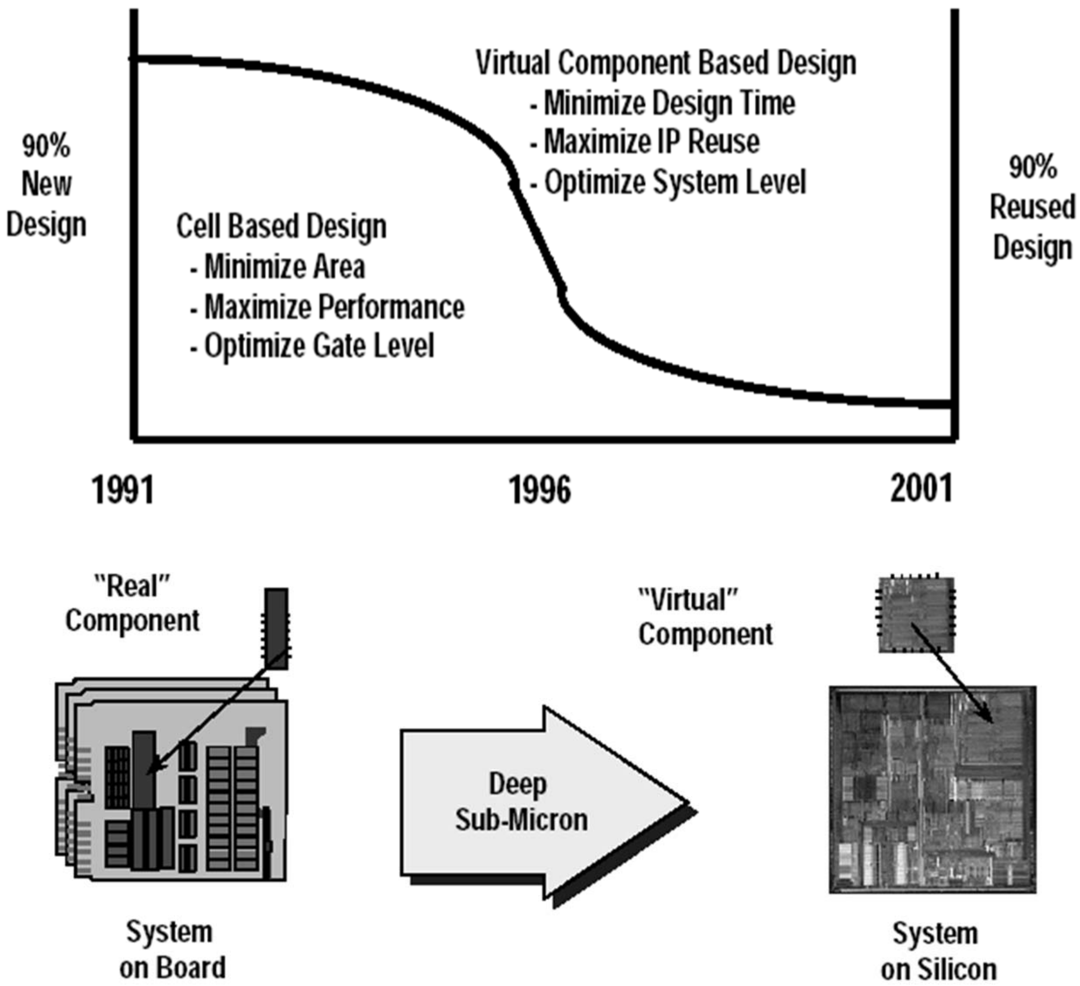
Where Are the Processors?



Look for the CPUs…the Opportunities Will Follow!

Source: DARPA/Intel (Tennenhouse)

On-going Paradigm Shift in Embedded System Design



- Change in business model due to SoCs
  - Currently many IC companies have a chance to sell devices for a single board
  - In future, a single vendor will create a System-on-Chip
  - But, how will it have knowledge of all the domains?
- Component-based design
  - Components encapsulate the intellectual property
- Platforms
  - Integrated HW/SW/IP
  - Application focus
  - Rapid low-cost customization

Embedded Systems

Embedded system = **engineering artifact involving computation that is subject to physical constraints**

Constraint #1: Reaction to the physical environment

Reaction constraints: deadlines, throughput, jitter

Constraint #2: Execution on a physical platform

Execution constraints: Bounds on available processor speeds, power, hardware failure rates

Challenge: Gain control over the interplay of computation with reaction and execution constraints, so as to meet given requirements.

Characteristics of Embedded Systems

Must be **dependable**:

Reliability *R(t)* = probability of system working correctly provided that is was working at t=0

Maintainability *M(d)* = probability of system working correctly d time units after error occurred.

Availability *A*(*t*): probability of system working at time t

Safety: no harm to be caused

Security: confidential and authentic communication

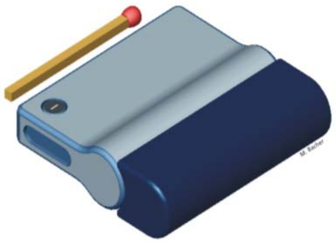
Even perfectly designed systems can fail if the assumptions about the workload and possible errors turn out to be wrong → e.g., Mars parthfinder ("priority inversion"), Ariane 5 rocket (overflow)

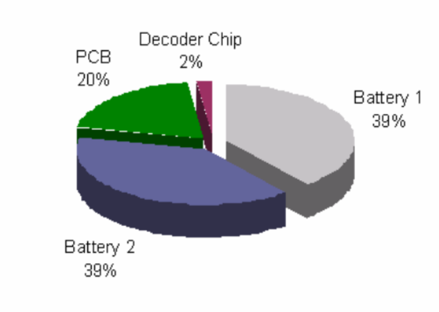
Making the system dependable must not be an after-thought, it must be considered from the very beginning.

Characteristics of Embedded Systems

Must be **efficient:**

- **Energy** efficient
- **Code-size** efficient (especially for systems on a chip)
- **Run-time** efficient
- **Weight** efficient
- **Cost** efficient





Dedicated towards a certain application

Knowledge about behavior at design time can be used to minimize resources and to maximize robustness

Some degree of re-programmability is essential

flexibility in upgrading, bug fixing, product differentiation, product customization

Dedicated user interface

(no mouse, keyboard and screen)

Characteristics of Embedded Systems

Many ES must meet real-time constraints

A real-time system must react to stimuli from the controlled object (or the operator) within the time interval dictated by the environment.

For real-time systems, right answers arriving too late are wrong.

„A real-time constraint is called **hard**, if not meeting that constraint could result in a catastrophe“ [Kopetz, 1997].

All other time-constraints are called **soft.**

Characteristics of Embedded Systems

Frequently connected to physical environment through sensors and actuators.

Typically Embedded Systems are

- **Hybrid systems** (analog + digital parts)
- **Reactive systems**

„A reactive system is one which is in continual interaction with is environment and executes at a pace determined by that environment“ [Bergé, 1995]

Behavior depends on input and current state.

Low-power requirements

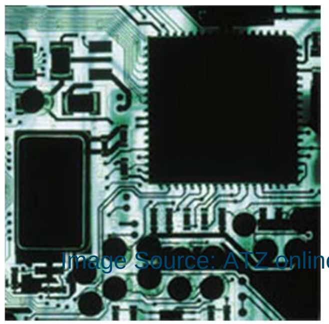
- Increasing emphasis on low-power
  - Longer battery life
  - Cheaper packaging
  - Smaller system size
  - Higher reliability

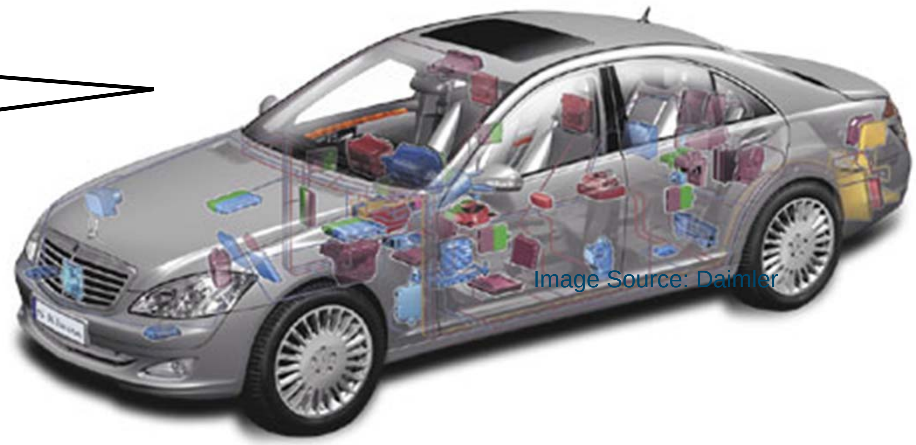
- Increasing computation demand from applications
  - More features/services in mobile devices
  - Mobile Multimedia → 0.3 GOPS … 1 GOPS
  - Home-Dialog/Robotics → ~ 1 GOPS
  - Automotive Vision → ~ 5 GOPS
  - 3D-Video → ~ 20 GOPS
  - High-Performance Video Restoration → ~ 1 TOPS

Simulation of Embedded Systems

System level simulation for embedded systems:

- Strongly required to validate internal architecture as well as interfaces between modules
- Abstracted models of the components:
  - Early available
  - enabling fast simulation





Important challenge:

- Integration of selected detailed component models within the simulation
  → early and accurate architecture validation

Introduction: Innovation Potential in Cars

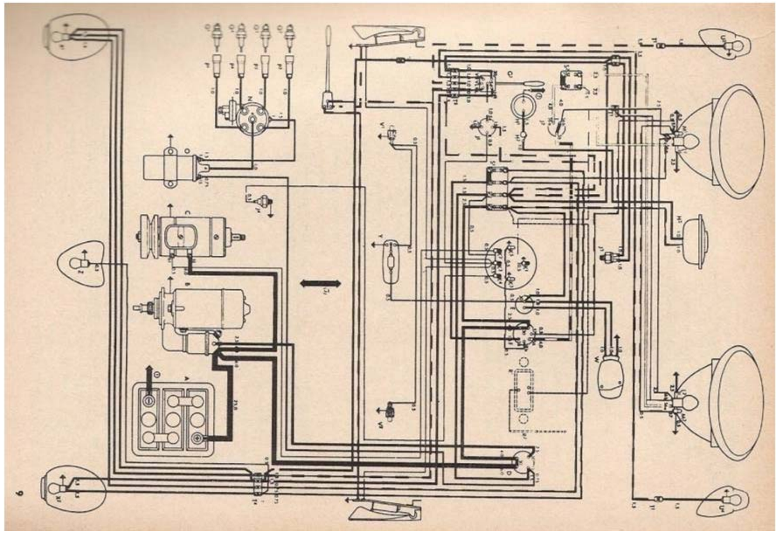
≥ 80% of innovation potential in cars comes from electronics

- Introduction of new functionalities (e.g. distance sensors)
- Enhancement of mechanic solutions by electronic components (e.g. engine management, car dynamics - ESP)
- Replacement of mechanic components by electronic counterparts (e.g. drive-by-wire)

Introduction: Networks in cars

Snapshot 2004: the VW Phaeton

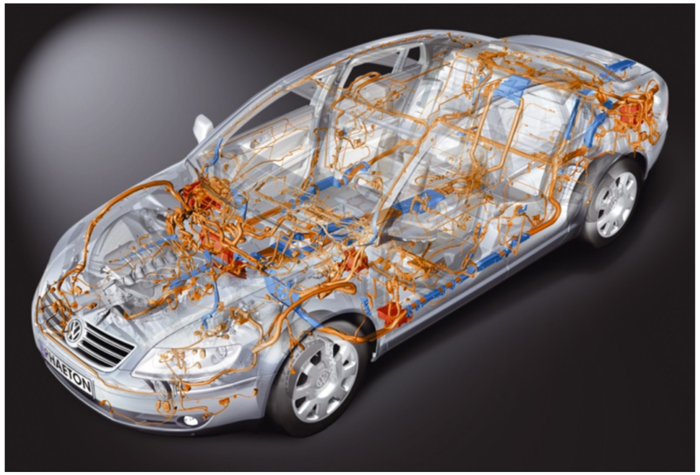
- 2110 cables
- 3860 meters cable
- Weight: 64kg
- 70 ECUs



Source: Volkswagen; Beetle 1960

Wide requirements

- Low cost for non safety-critical systems (e.g. LIN, CAN)
- High bandwidth for infotainment (e.g. MOST)
- Dependability for safety-critical applications (e.g. FlexRay)

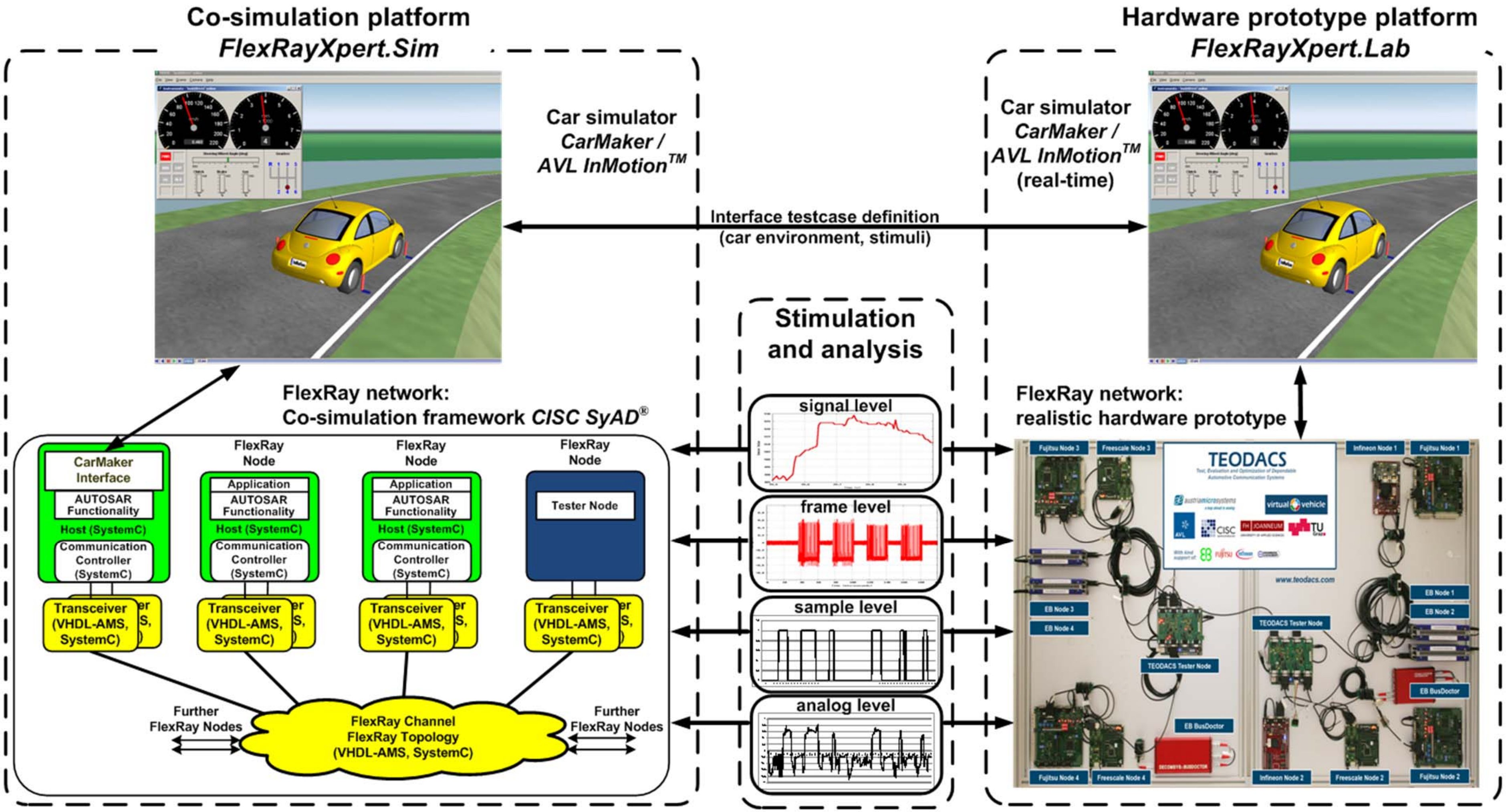


Source: Volkswagen; Phaeton 2004

➔ **System complexity difficult to manage**

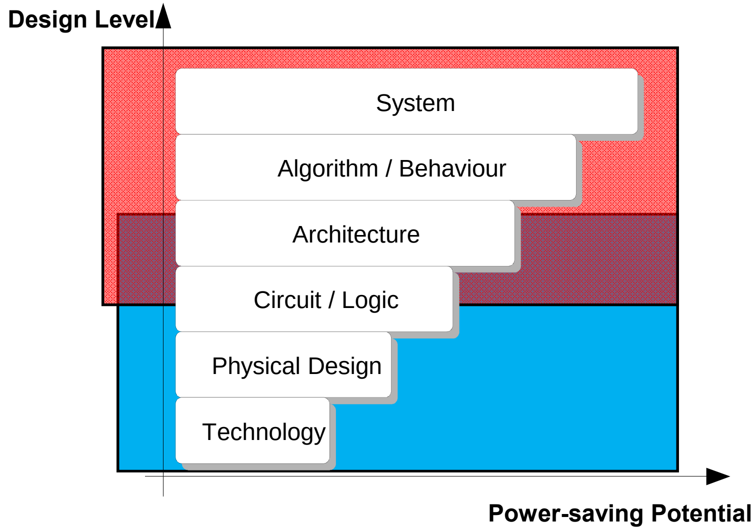
TEODACS: Overview

TEODACS: <u>**T**</u>est, <u>**E**</u>valuation and <u>**O**</u>ptimization of <u>**D**</u>ependable <u>**A**</u>utomotive <u>**C**</u>ommunication <u>**S**</u>ystems



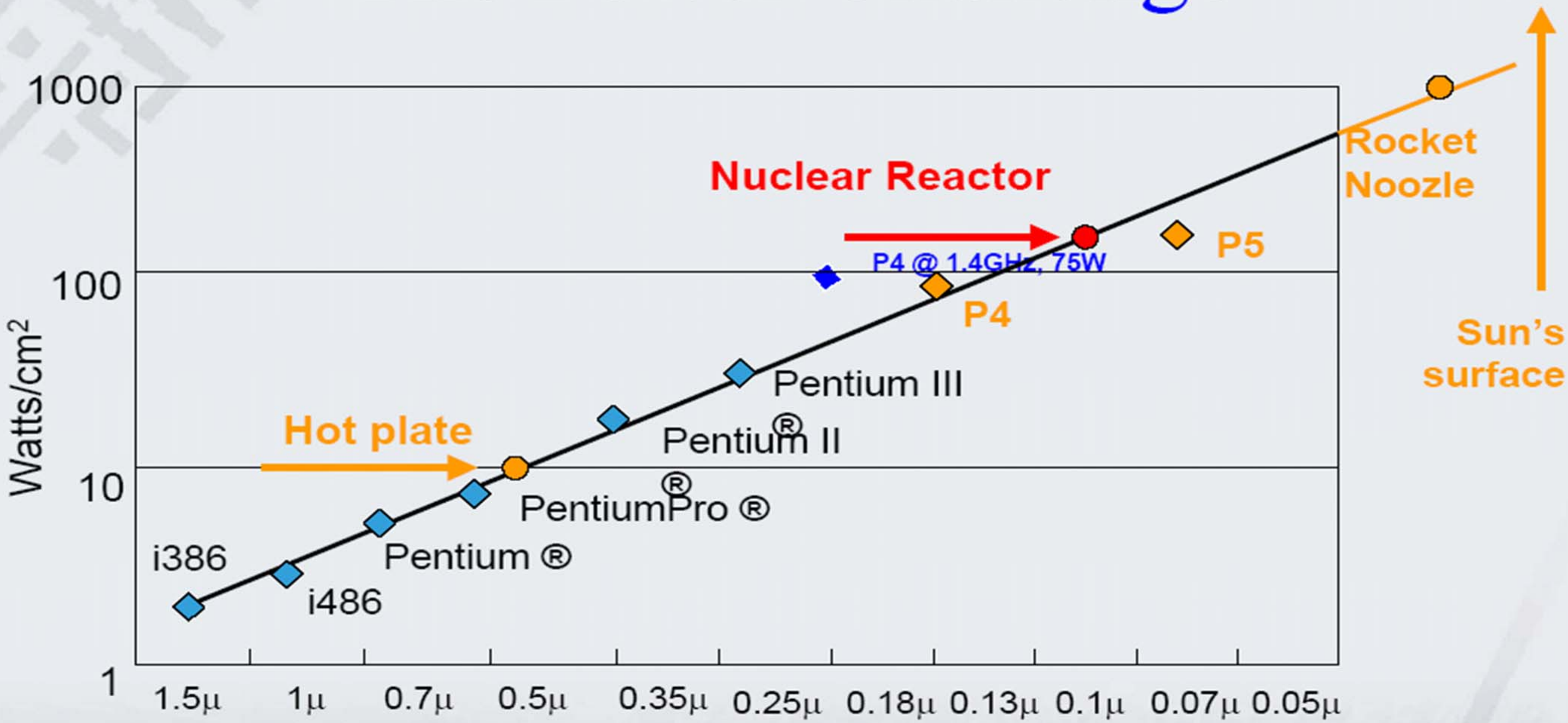
Power Aware Computing

- Power aware computing
- Low power design



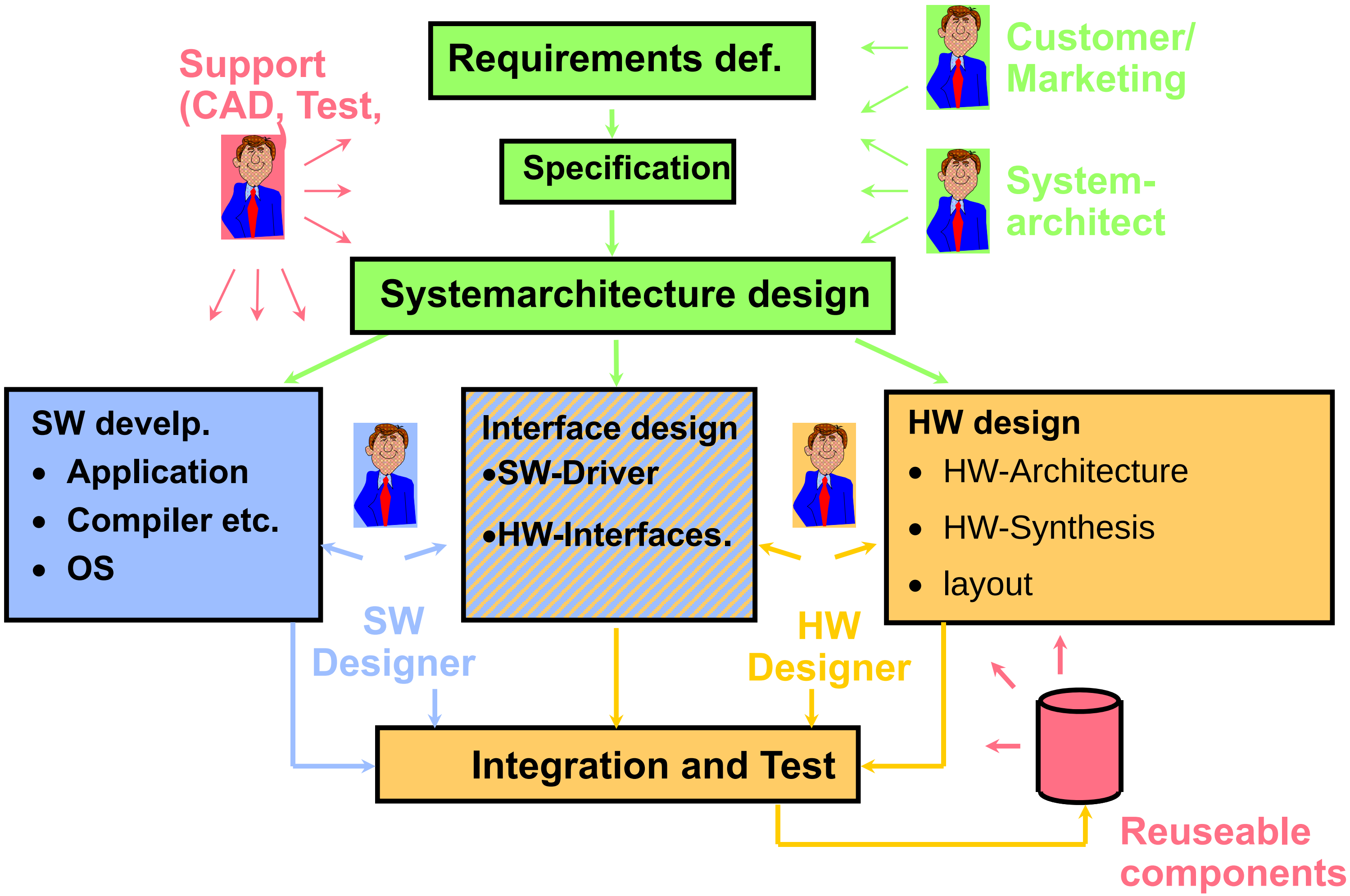
- Mobile devices
  - high performance
  - small
  - less power consumption
- Temperature Aware Computing (in general purpose processors)

The Power Challenge



Courtesy of Fred Pollack, Intel
Keynote speech, MICRO-32

Design flow

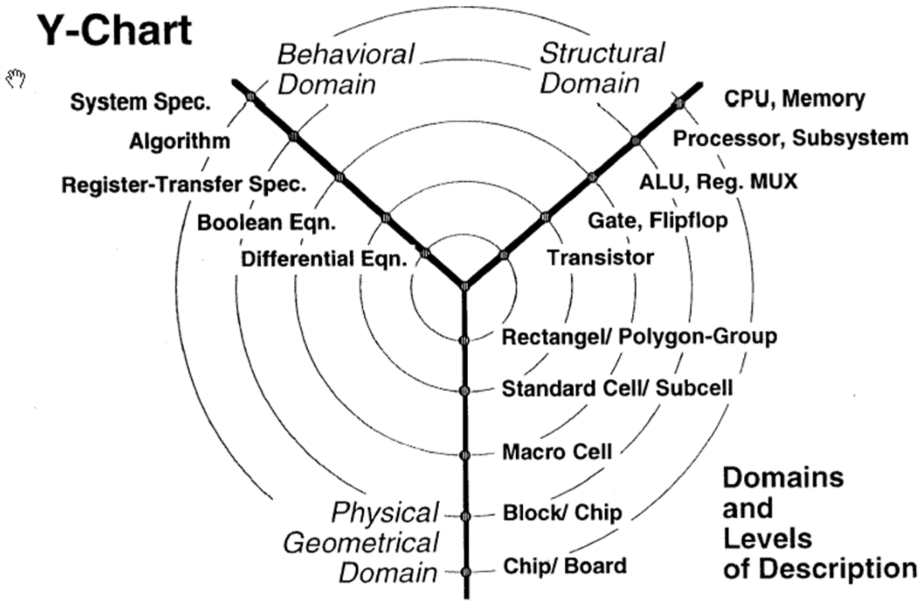


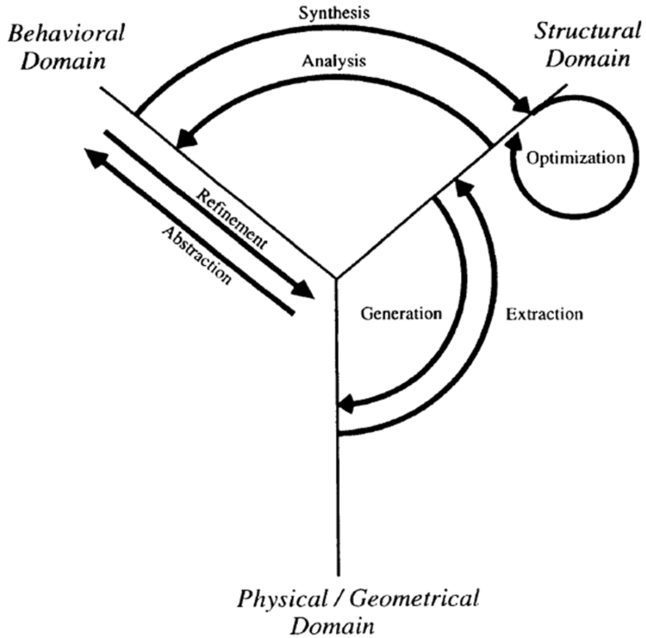
Design Process for Embedded Systems (Prof. Ernst, TU Braunschweig)

Y-Chart

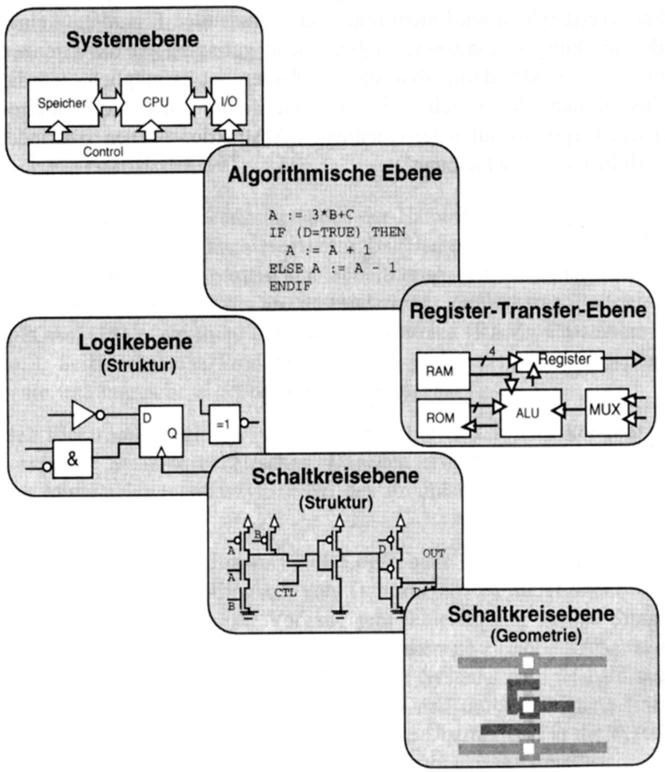
- 3 design views
  - Behavior (functionality)
  - Structure (netlist)
  - Physical (layout)

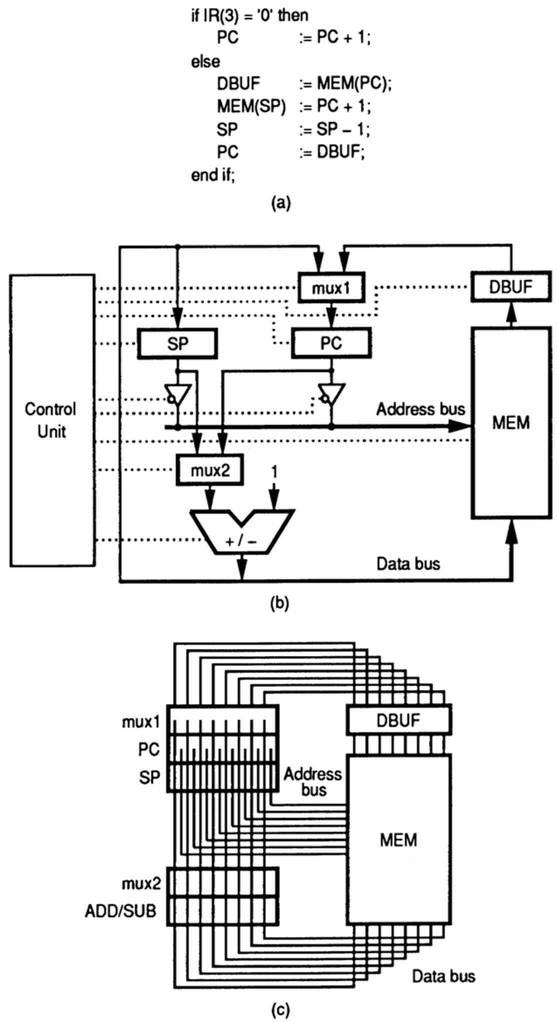
- 5 abstraction levels





Layers in the design process





Behavior

Structure

Physical

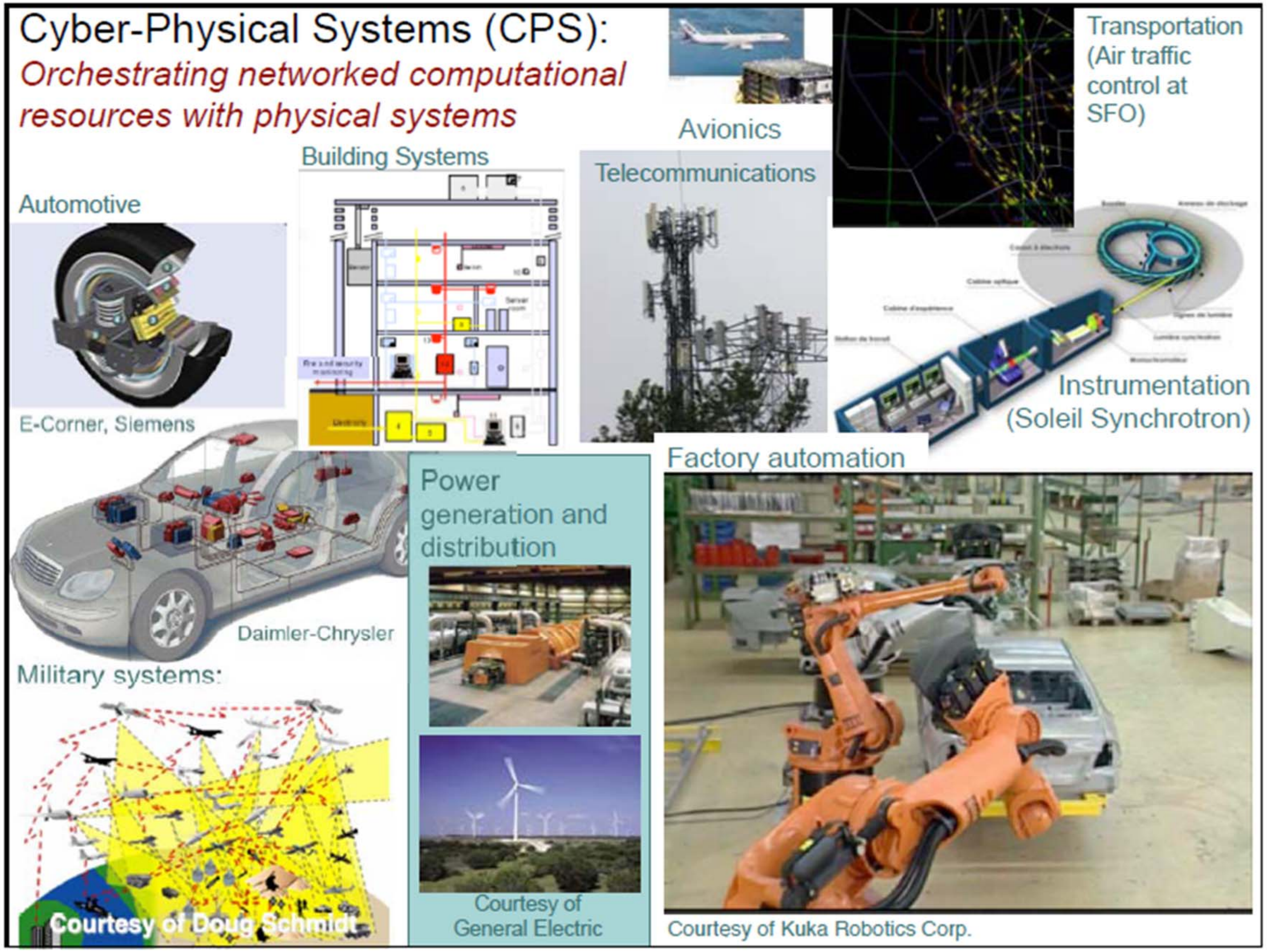
Comparison

Embedded Systems

- Few applications that are known at design-time.
- Not programmable by end
- Fixed run-time requirements (additional computing power not useful).
- Criteria:
  - cost
  - power consumption
  - predictability
  - …
- Development environment is not the runtime environment

General Purpose Computing

- Broad class of applications
- Programmable by end user
- Faster is better.
- Criteria:
  - cost
  - average speed
  - …
- Development environment is the runtime environment


Cyber-Physical Systems (CPS):
Orchestrating networked computational resources with physical systems
Avionics
Transportation (Air traffic control at SFO)
Building Systems
Telecommunications
Automotive
E-Corner, Siemens
Instrumentation (Soleil Synchrotron)
Factory automation
Power generation and distribution
Daimler-Chrysler
Military systems:
Courtesy of Doug Schmidt
Courtesy of General Electric
Courtesy of Kuka Robotics Corp.

Prof. E. Lee

Where CPS Differs from

- The traditional embedded systems problem:

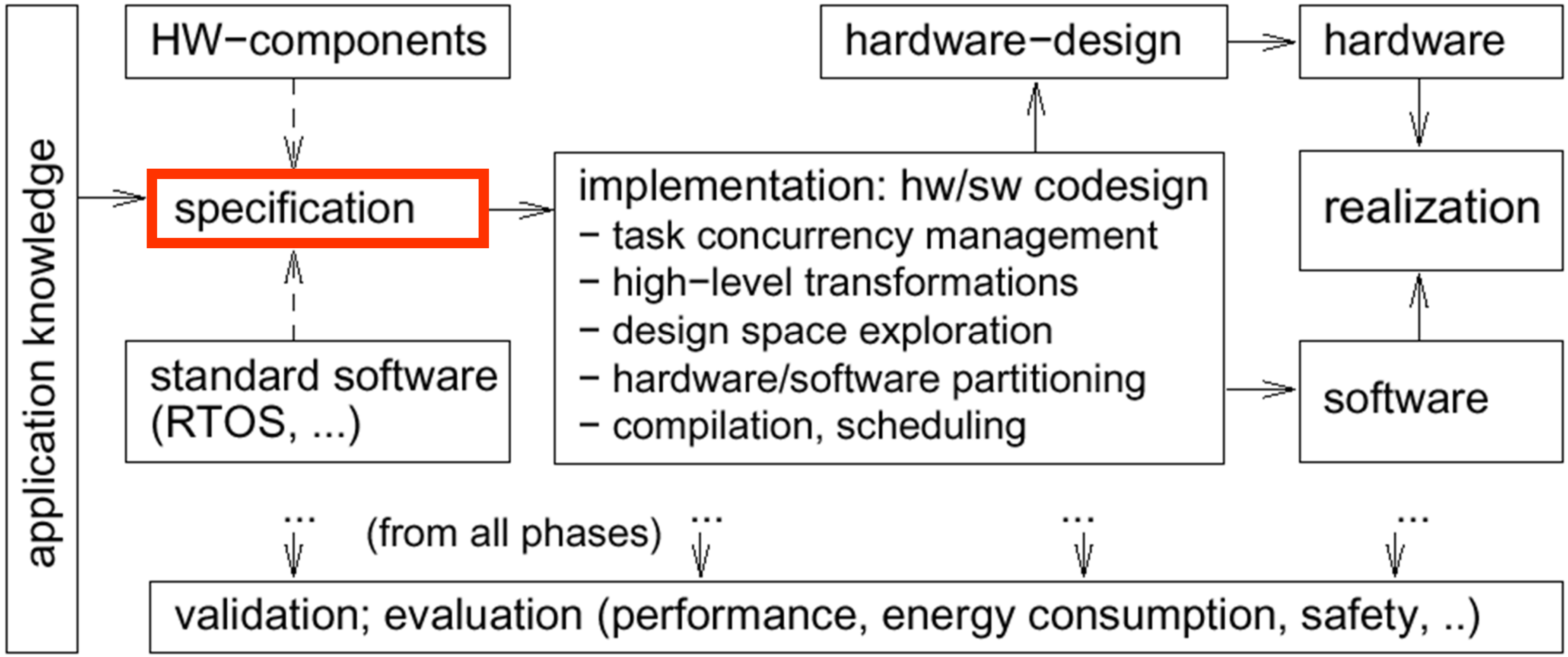
Embedded software is software on small computers. The technical problem is one of optimization (coping with limited resources and extracting performance).

- *The CPS problem:*

Computation and networking integrated with physical processes. The technical problem is managing dynamics, time, and concurrency in networked computational + physical systems.

Specification – Models of Computation (MOC)

Specifications



Specification of embedded systems: Requirements for specification techniques (1)

- **Hierarchy**
  Humans not capable to understand systems containing more than a few objects.
  Most actual systems require far more objects.
  two kinds of hierarchy are used:
  - Behavioral hierarchy
    Examples: states, processes, procedures.
  - Structural hierarchy
    Examples: multipliers, FPUs, processors, printed circuit boards
- **Timing behavior**
- **State-oriented behavior**
  suitable for reactive systems

Requirements for specification techniques (2)

- **Event-handling** (external or internal events)
- **No obstacles for efficient implementation**
- **Support for the design of dependable systems**
  Unambiguous semantics, ...
- **Exception-oriented behavior**
  Not acceptable to describe exceptions for every state.

Requirements for specification techniques (3)

- **Concurrency**
  Real-life systems are concurrent
- **Synchronization and communication**
  Components have to communicate!
- **Presence of programming elements**
  For example, arithmetic operations, loops, and function calls should be available
- **Executability**
- **Support for the design of large systems**
- **Domain-specific support**

Requirements for specification techniques (4)

- **Readability**
- **Portability and flexibility**
- **Non-functional properties**
  fault-tolerance, availability, EMC-properties, weight, size, user friendliness, extendibility, expected life time, power consumption...
- **Adequate model of computation**