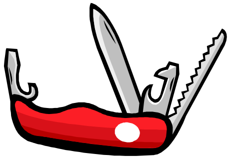
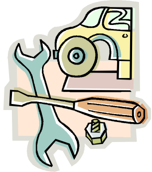
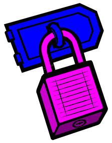
Common characteristics

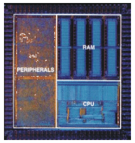
Dependability

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

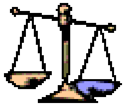
Efficiency

- ES must be **efficient**
  - Code-size efficient
    (especially for systems on a chip)

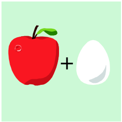


  - Run-time efficient

  - Weight efficient

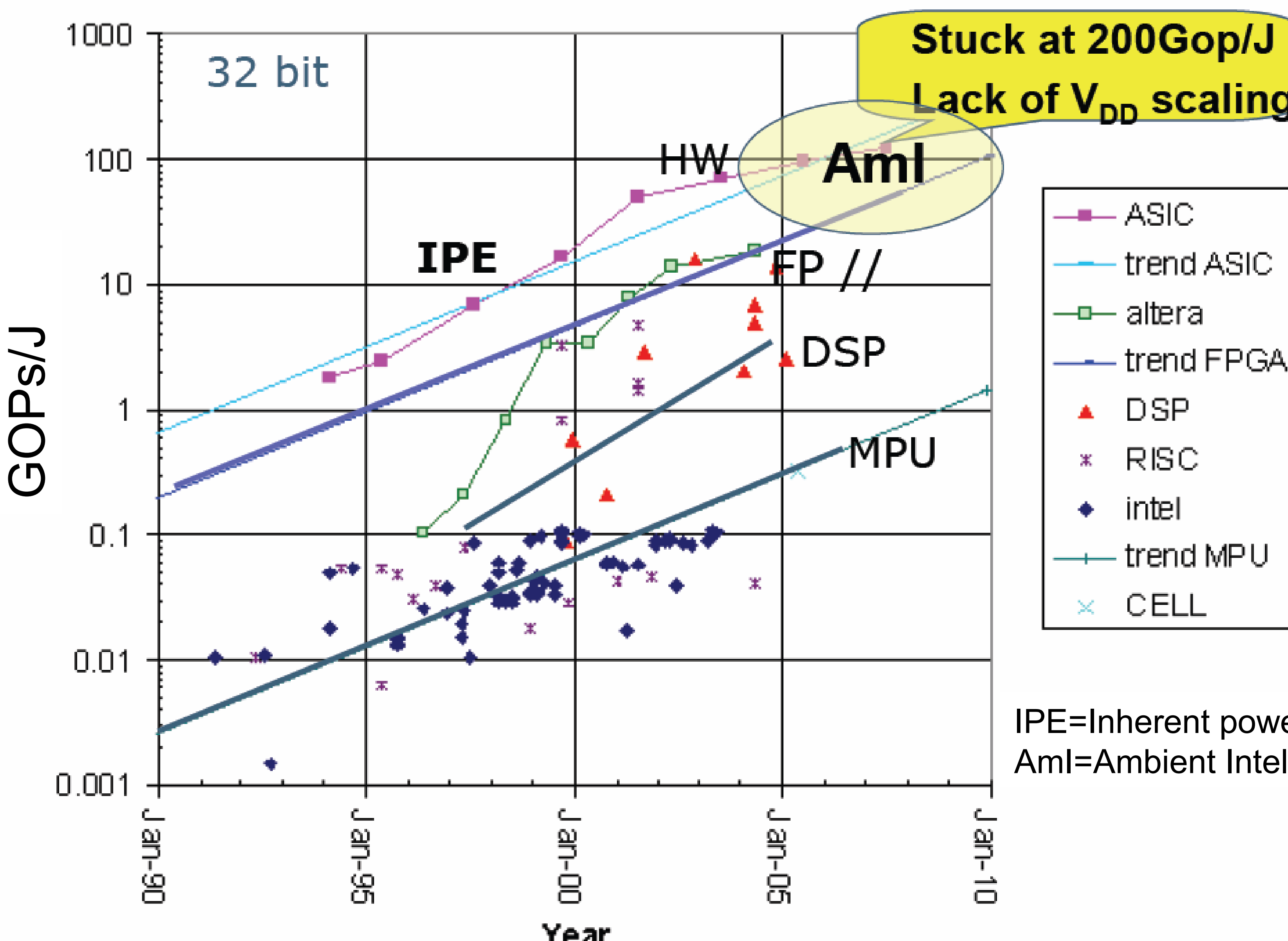


  - Cost efficient



  - Energy efficient

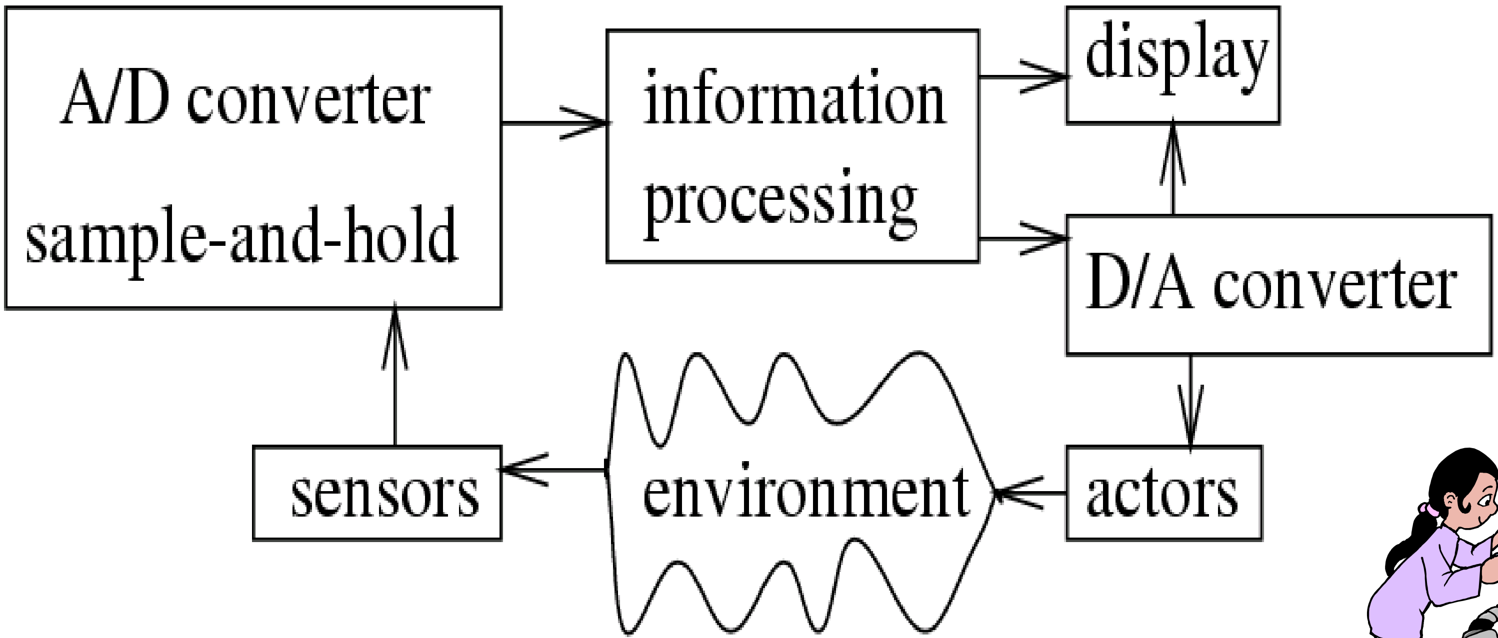
Importance of Energy Efficiency

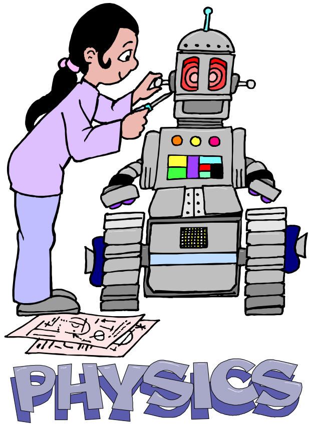


IPE=Inherent power efficiency
AmI=Ambient Intelligence

Courtesy: Philips

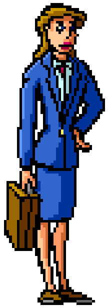
Connected to the physical environment





☞ Cyber-physical systems

Real-time constraints

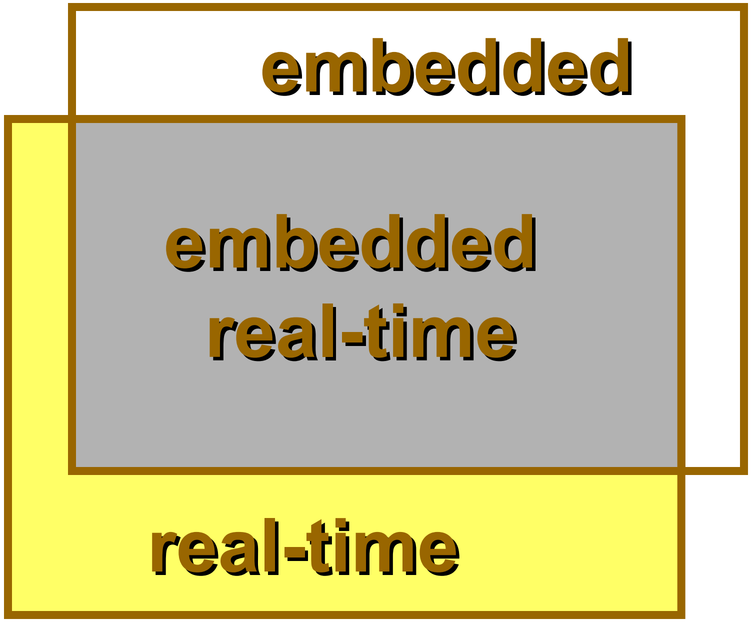


- Many ES must meet **real-time constraints**
  - A real-time system must react to stimuli from the controlled object (or the operator) within the time interval **dictated** by the environment.
  - For real-time systems, right answers arriving too late are wrong.
  - **"A real-time constraint is called hard, if not meeting that constraint could result in a catastrophe"** [Kopetz, 1997].
  - All other time-constraints are called **soft**.
  - A guaranteed system response has to be explained without statistical arguments

Real-Time Systems

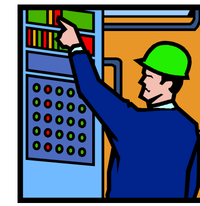
Embedded and Real-Time Synonymous?

- Most embedded systems are real-time
- Most real-time systems are embedded



Reactive & hybrid systems

- Typically, ES are **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.**
  ☞ automata model appropriate,
  model of computable functions inappropriate.



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

Dedicated systems

- **Dedicated** towards a certain **application**
  Knowledge about behavior at design time can be used to minimize resources and to maximize robustness
- **Dedicated user interface**
  (no mouse, keyboard and screen)

Underrepresented in teaching

- ES are **underrepresented in teaching** and public discussions:
  "*Embedded chips aren't hyped in TV and magazine ads ...*" [Mary Ryan, EEDesign, 1995]

Not every ES has all of the above characteristics.

Def.: Information processing systems having most of the above characteristics are called embedded systems.

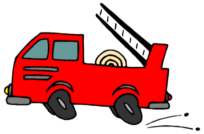
Course on embedded systems makes sense because of the number of common characteristics.

Challenges in ES Design

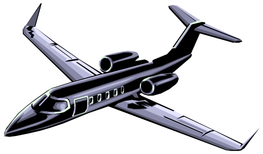
Quite a number of challenges, e.g. dependability

Dependability? 

- Non-real time protocols used for real-time applications (e.g. Berlin fire department)

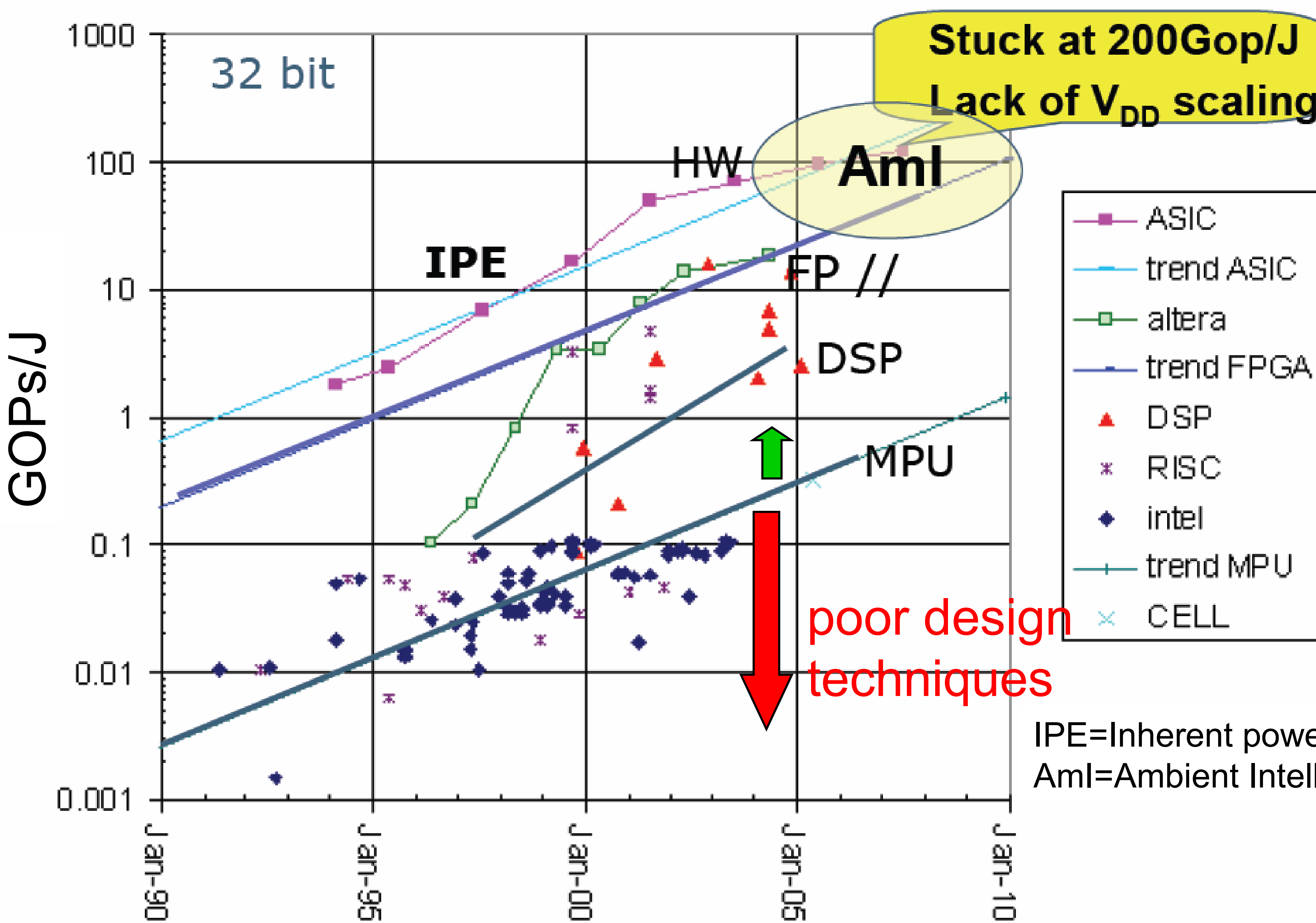


- Over-simplification of models (e.g. aircraft anti-collision system)



- Using unsafe systems for safety-critical missions (e.g. voice control system in Los Angeles; ~ 800 planes without voice connection to tower for > 3 hrs

Efficiency



IPE=Inherent power efficiency
AmI=Ambient Intelligence

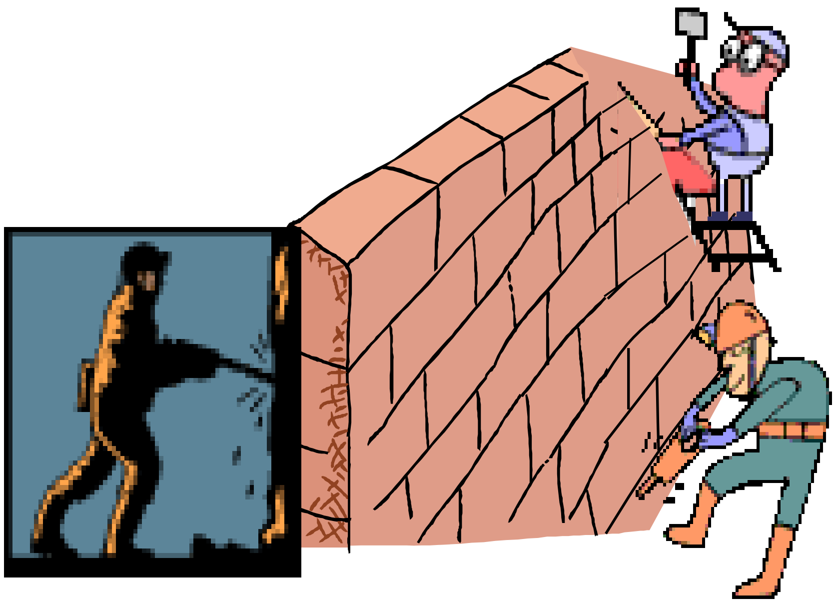
Efficient software design needed, otherwise, the price for software flexibility cannot be paid.

It is not sufficient to consider ES just as a special case of software engineering

EE knowledge must be available,
Walls between EE and CS must be torn down

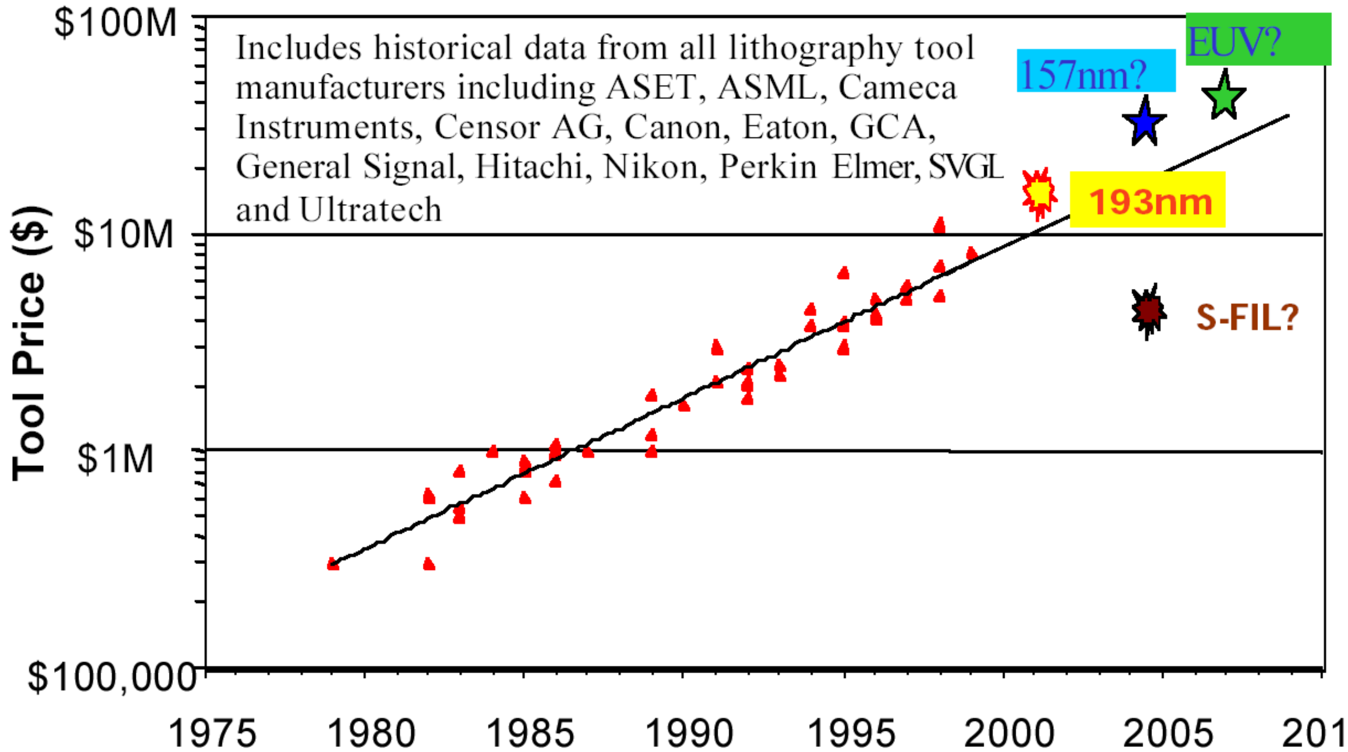
CS

EE



Challenges for implementation in hardware

- Lack of flexibility (changing standards).
- Mask cost for specialized HW becomes very expensive



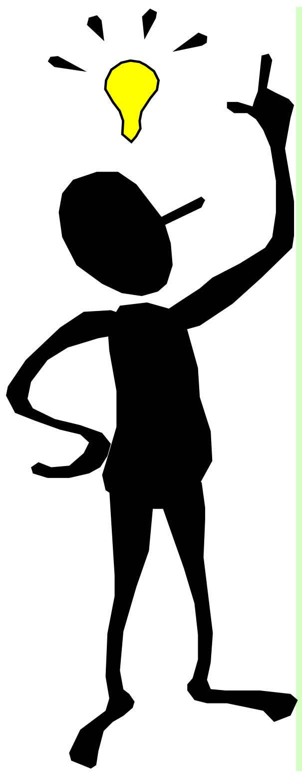
☞Trend towards implementation in Software

[http://www.molecularimprints.com/Technology/tech_articles/MII_COO_NIST_2001.PDF9]

Importance of Embedded Software and Embedded Processors

"... the New York Times has estimated that the average American comes into contact with about 60 micro-processors every day...." [Camposano, 1996]

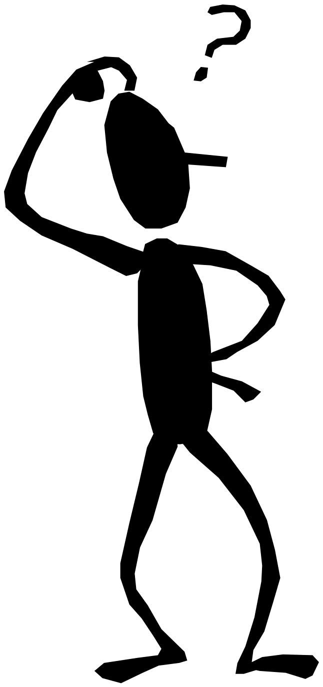
Latest top-level BMWs contain over 100 micro-processors [Personal communication]



Most of the functionality will be implemented in software

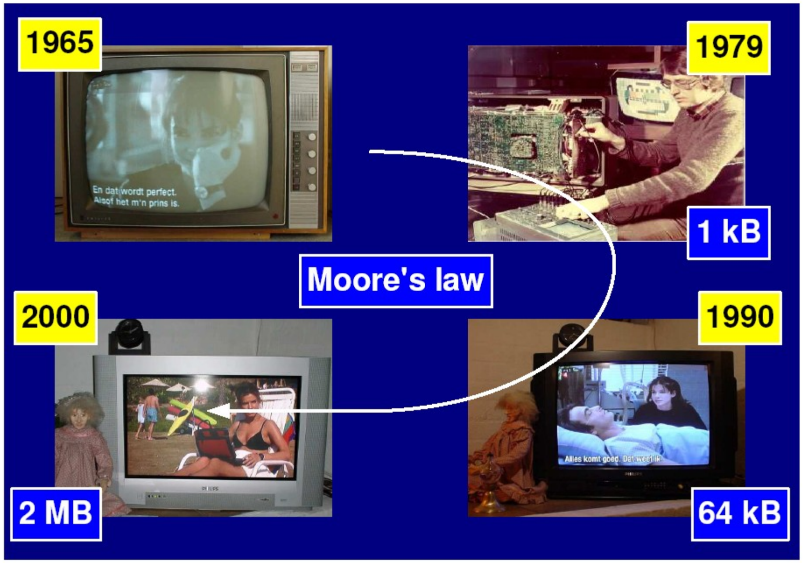
Challenges for implementation in software

If embedded systems will be implemented mostly in software, then why don‘t we just use what software engineers have come up with?



Software complexity is a challenge

- 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]

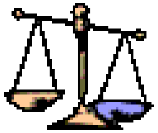
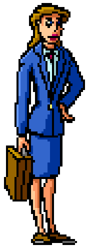


Rob van Ommering, COPA Tutorial, as cited by: Gerrit Müller: Opportunities and challenges in embedded systems, Eindhoven Embedded Systems Institute, 2004

Challenges for Embedded Software

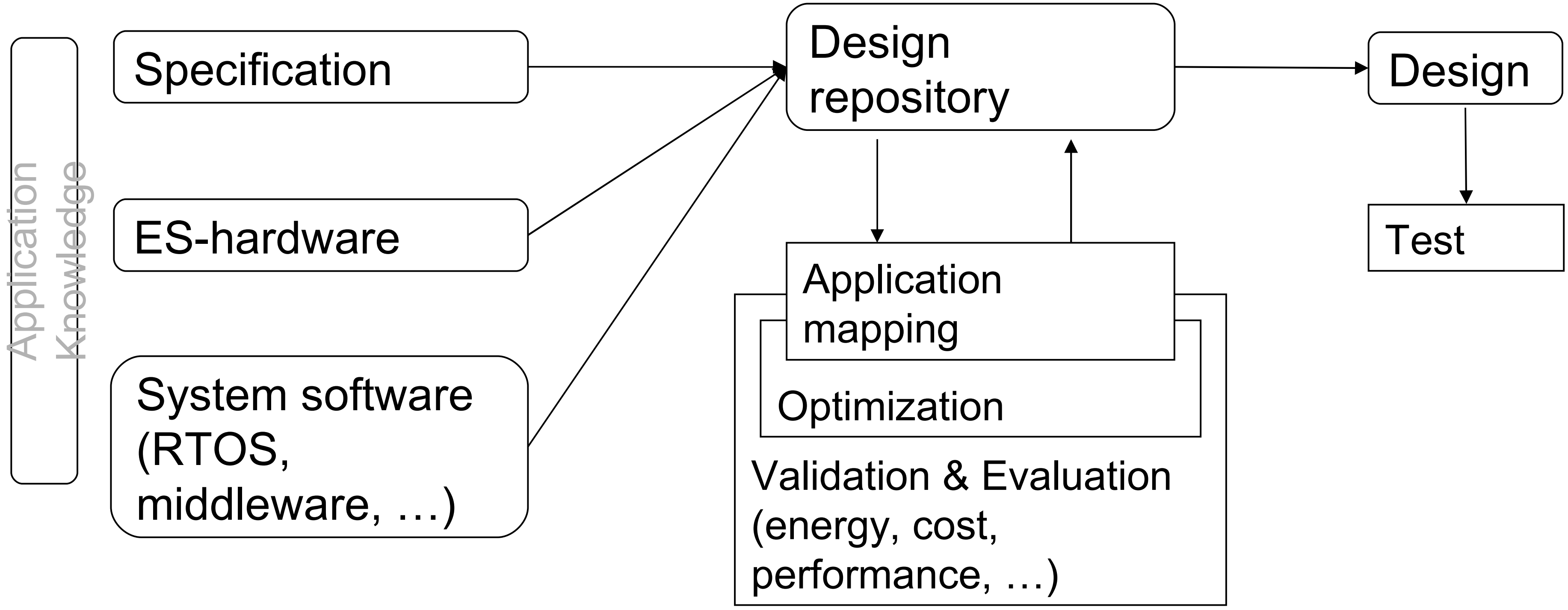
- Dynamic environments
- Capture the required behaviour!
- Validate specifications
- Efficient translation of specifications into implementations!
- How can we check that we meet real-time constraints?
- How do we validate embedded real-time software? (large volumes of data, testing may be safety-critical)




Design flows

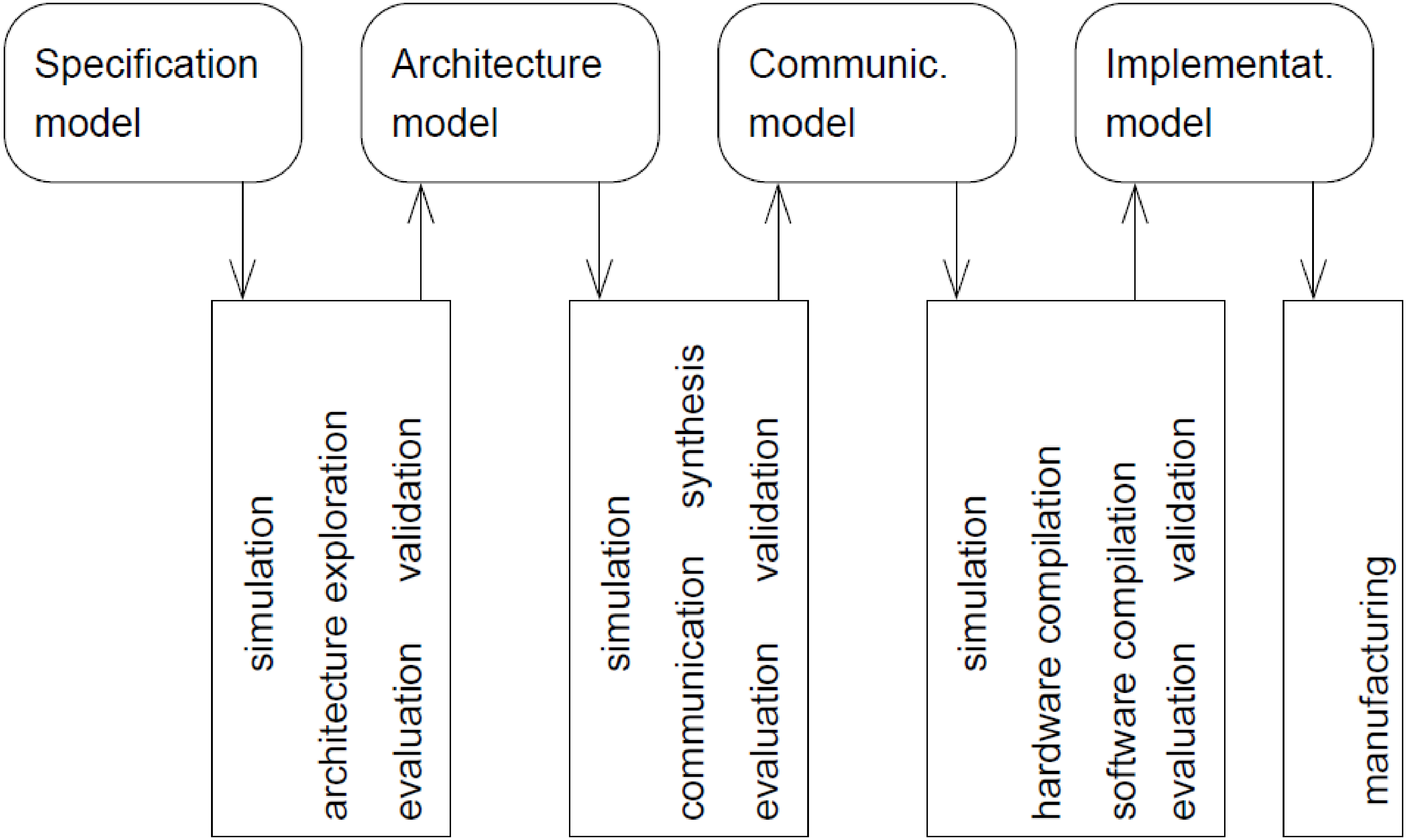
Hypothetical design flow



Generic loop: tool chains differ in the number and type of iterations

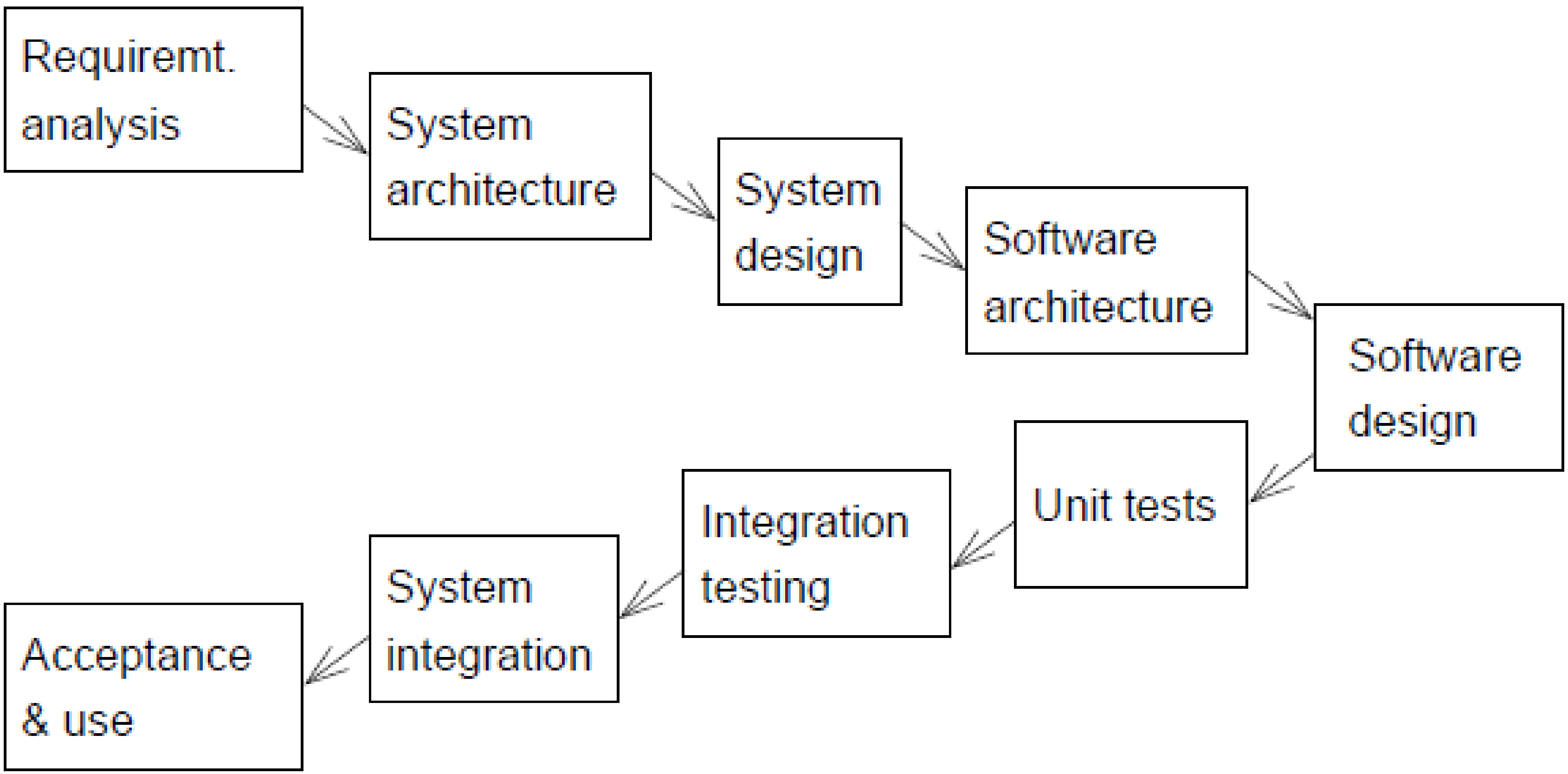
Iterative design (1)
- After unrolling loop -

Example:
SpecC
tools



Iterative design (2)
- After unrolling loop -

Example: V-model



Design approaches

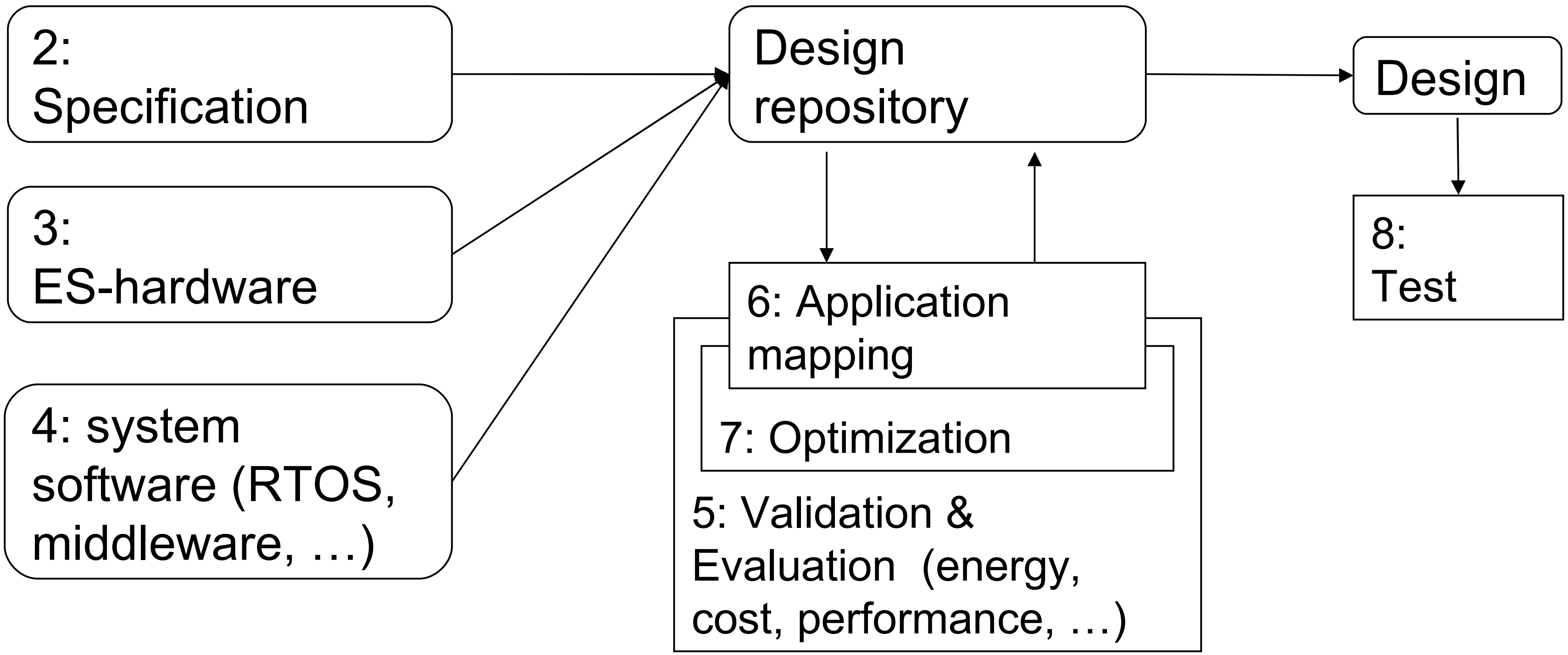
Definition: **Synthesis** is the process of generating the description of a system in terms of related lower-level components from some high-level description of the expected behavior.

☞"describe-and-synthesize" paradigm by Gajski, 1994

In contrast to the traditional "specify-explore-refine" approach, also known as "design-and-simulate" approach.

Manual design steps are more error-prone than automatic synthesis and, therefore, simulation is more important.

Structure of the course



Numbers indicate sequence of chapters

Other sources of information

- Micro-controllers, including their memory, I/O and interrupt structure
  - [Ball, 1996], [Ball, 1998], [Ganssle, 2000], [Ganssle, 2008], [Ganssle et al., 2008], [Heath, 2000], [Barr, 1999], [Barrett and Pack, 2005], [Labrosse, 2000].
- Specification languages: [Young, 1982], [Burns and Wellings, 1990], [Bergé et al., 1995], Micheli [de Micheli et al., 2002].
- Information on new languages: SystemC [Müller et al., 2003], SpecC [Gajski et al., 2000], Java etc.
- Approaches for designing RTOSes: [Kopetz, 1997].
- Real-time scheduling: [Buttazzo, 2002], [Krishna and Shin, 1997].
- Laplante [Laplante, 1997], Vahid [Vahid, 2002],
- ARTIST road map [Bouyssounouse and Sifakis, 2005],
- Embedded Systems Handbook [Zurawski, 2006].
- Robotics
- Slides of Rajiv Gupta, UCSD
- D. Gajski et al.: Embedded System Design, Springer, 2009

Summary

- Characteristics
  - Reliability, efficiency, …
- Challenges in embedded system design
  - Reliability, efficiency, …
- Design flows
- Structure of this course