Introduction to Embedded System

Universidad de Alcalá

Departamento de Electrónica

Future of IT?

❑According to forecasts characterized by the terms such as

- *Post-PC era*
- *Disappearing computer*
- *Ubiquitous computing*
- *Pervasive computing*
- *Ambient intelligence*
- *Embedded systems*

What is an embedded systems?

iPhone

Laser Keyboard

Nikon D300

Video Watch

GPS

Playstation 3

PC Keyboard

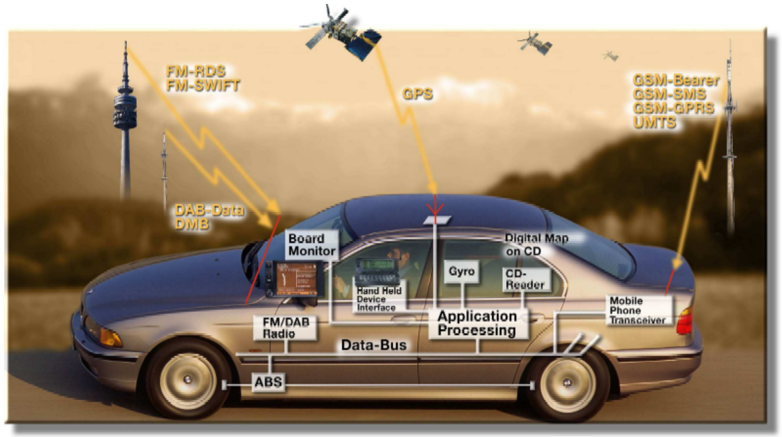
SD Card

Embedded systems (ES) = **information processing systems embedded into a larger product**

What is an embedded system?

Avionics

Communication



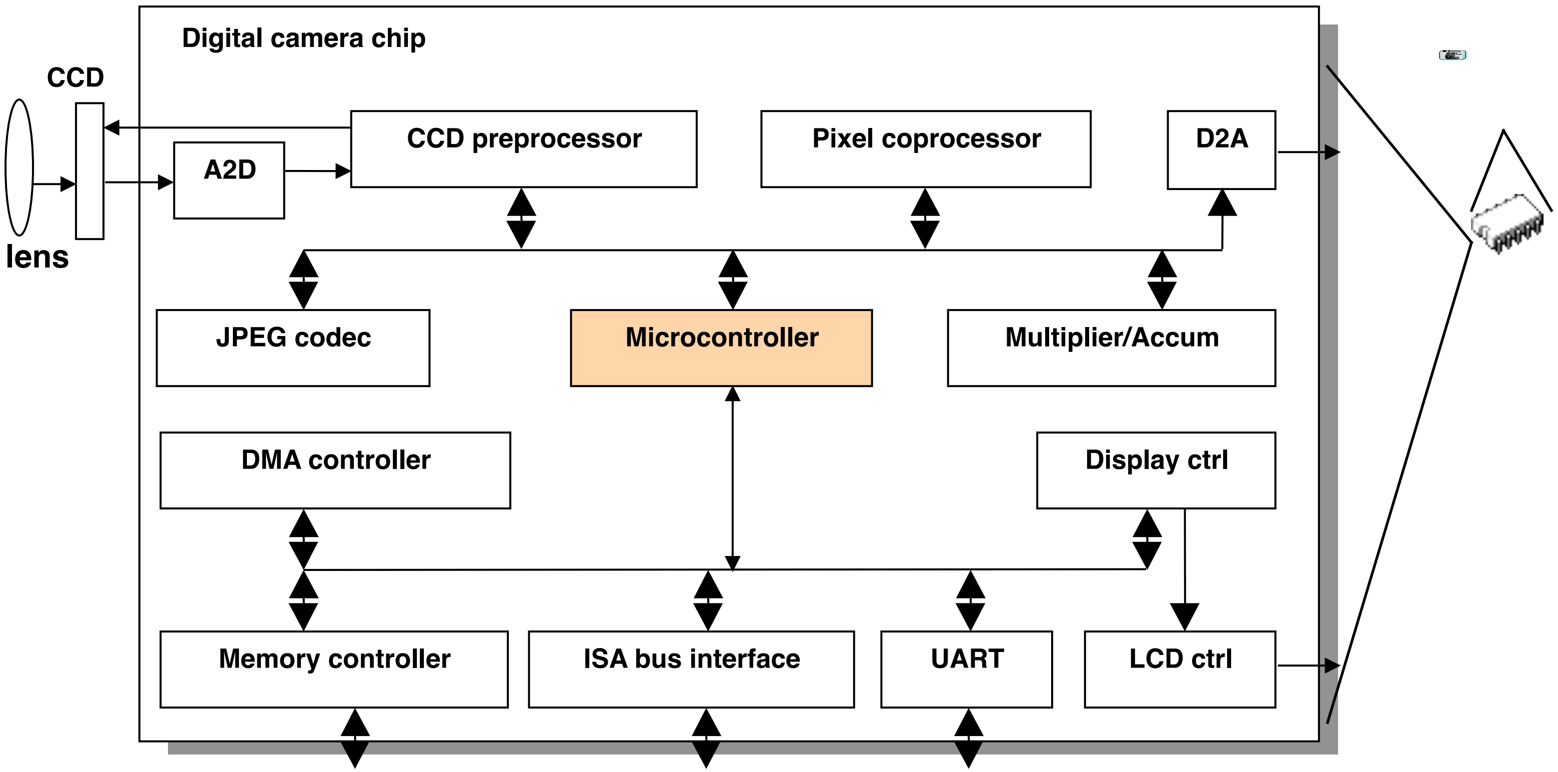
Automobile

Consumer Electronics

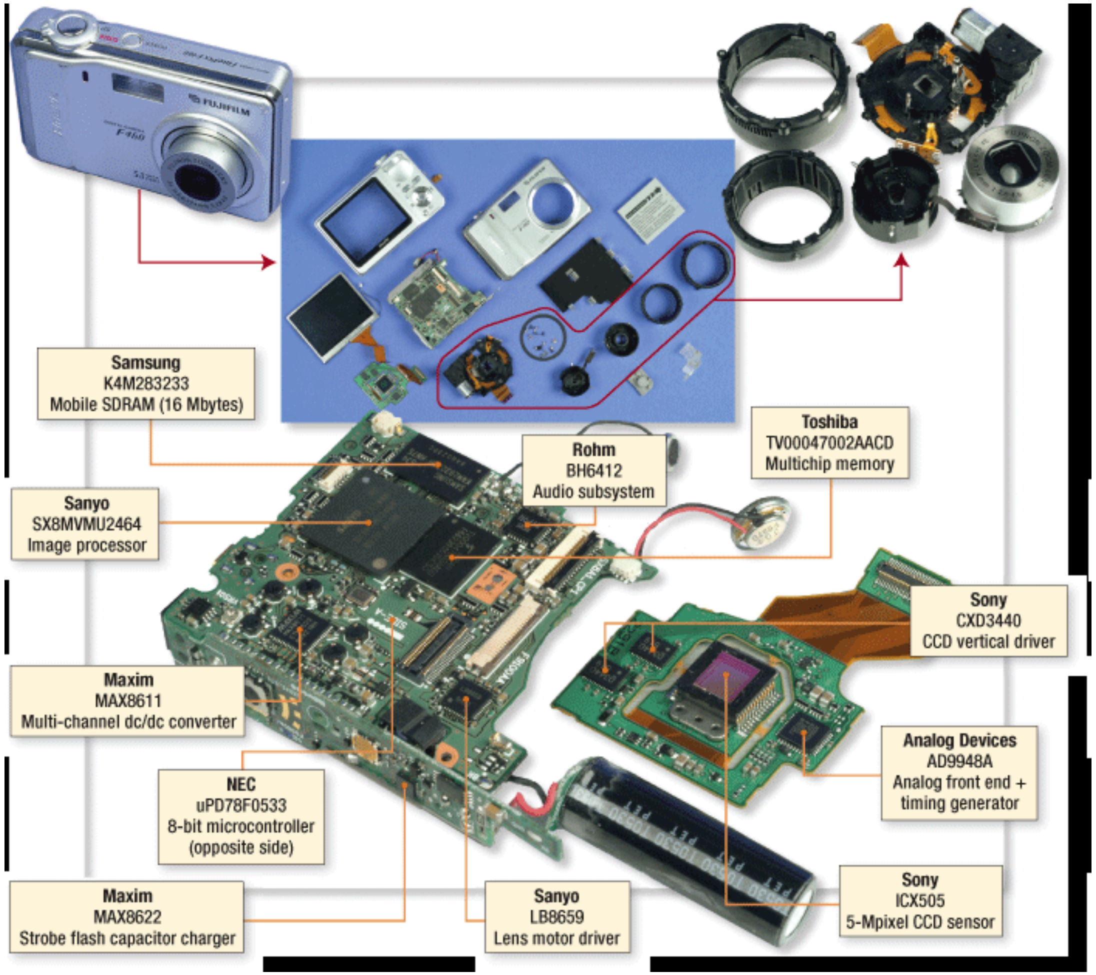
Office Equipments

Household Appliances

An embedded system example -- a digital camera



An embedded system example -- a digital camera

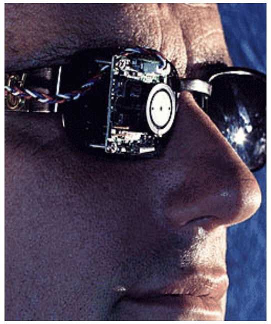


Application areas

1. Automotive electronics
2. Aircraft electronics
3. Trains
4. Telecommunication

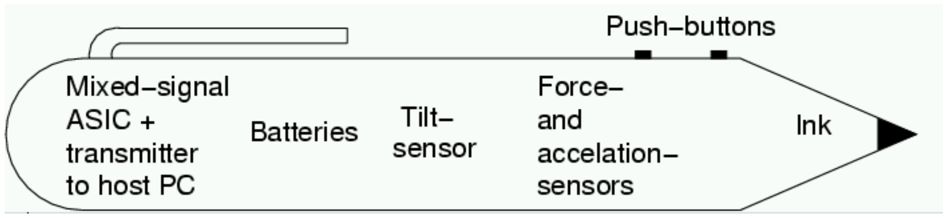
Application areas

5. Medical systems
   e.g. “artificial eye”



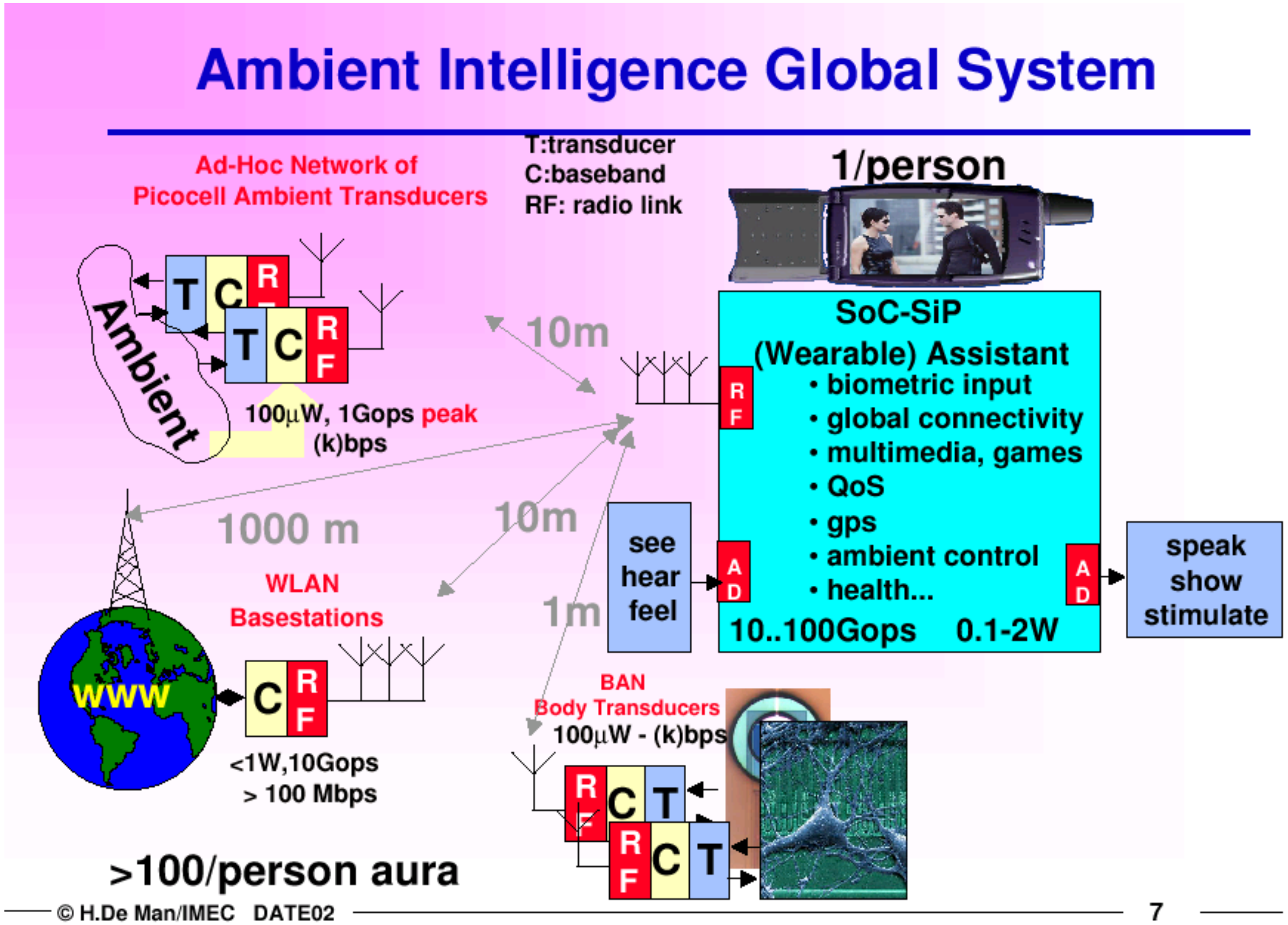
6. Military applications

7. Authentication



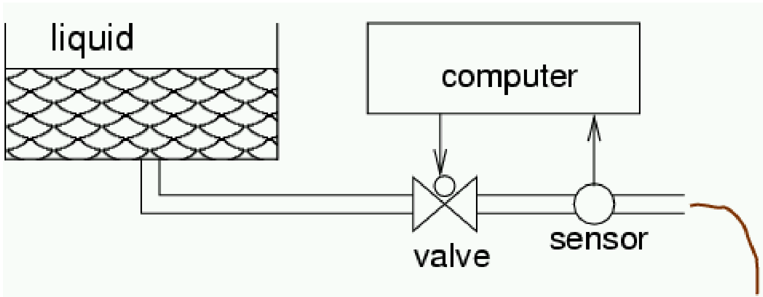
Application areas

8. Consumer electronics

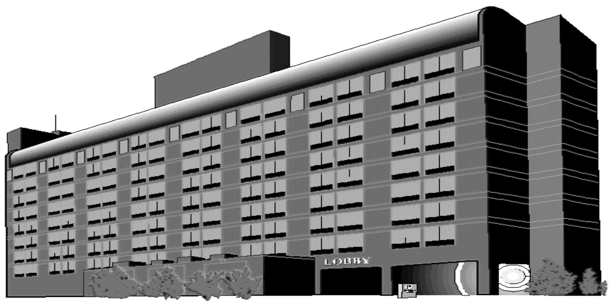


Application areas

9. Fabrication equipment



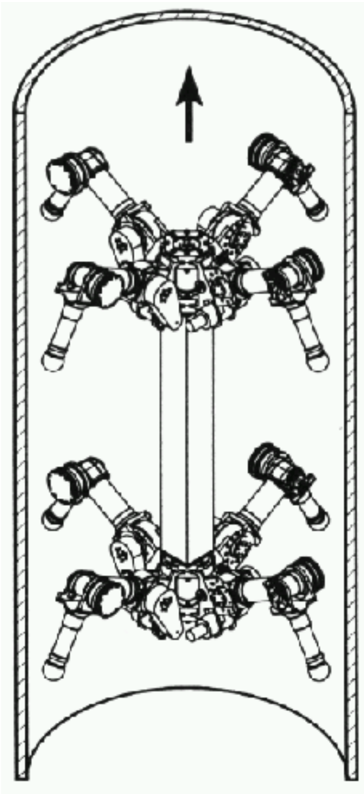
10. Smart buildings



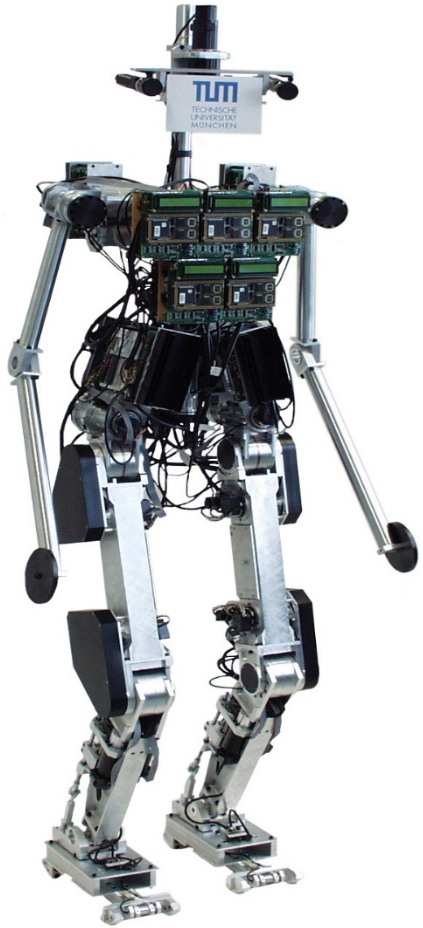
Application areas

11. Robotics

„Pipe-climber“



Robot „Johnnie“ (Courtesy and ©: H.Ulbrich, F. Pfeiffer, TU München)

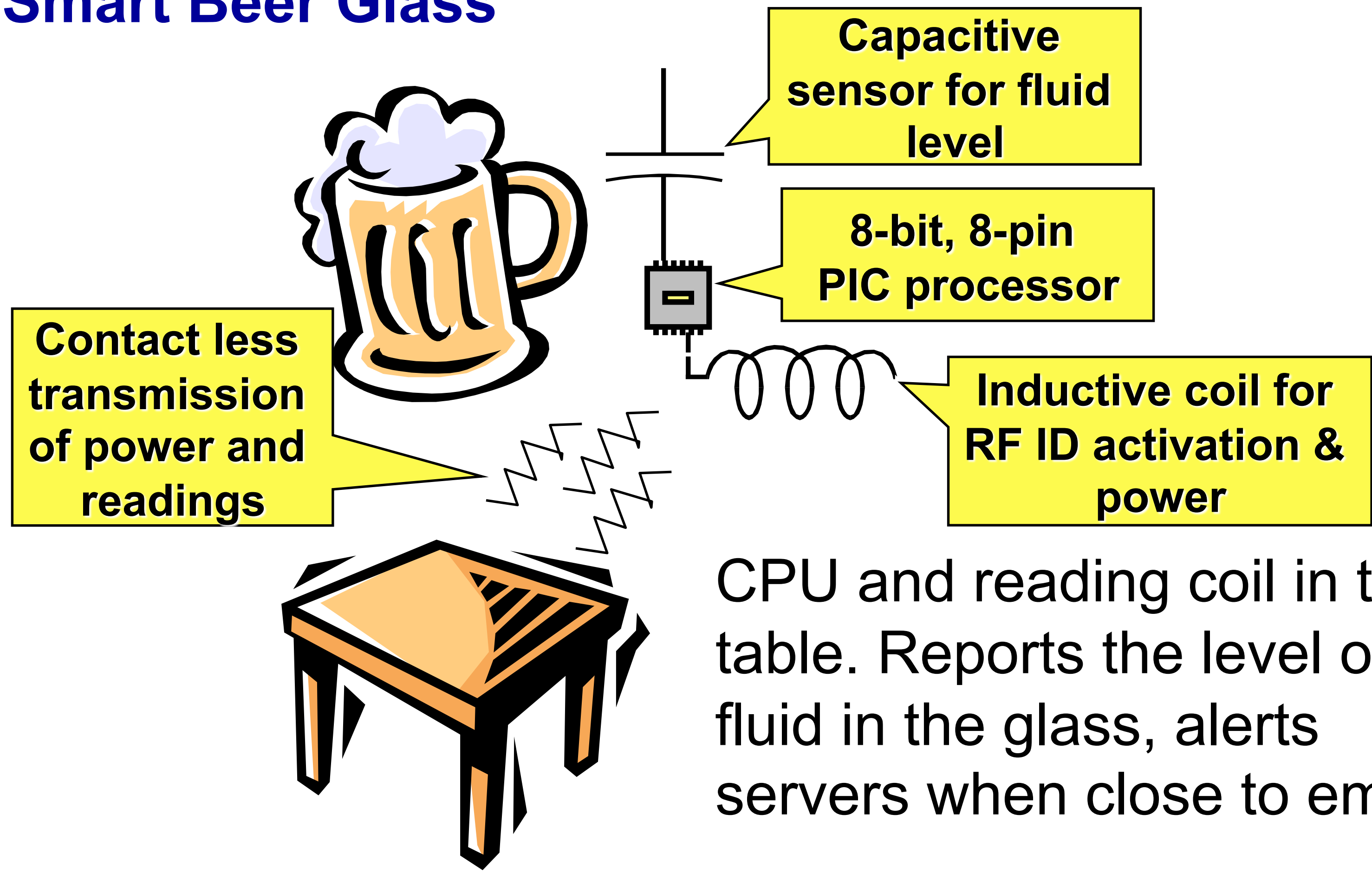


Embedded systems from real life

- ❑ Typical embedded solution
- ❑ Integrates several technologies:
  - Radio transmissions
  - Sensor technology
  - Magnetic inductance for power
  - Computer used for calibration
- ❑ Impossible without the computer
- ❑ Meaningless without the electronics

Embedded systems from real life

Smart Beer Glass

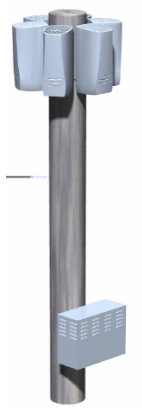


CPU and reading coil in the table. Reports the level of fluid in the glass, alerts servers when close to empty

Embedded systems from real life

Mobile Phones and Base Stations

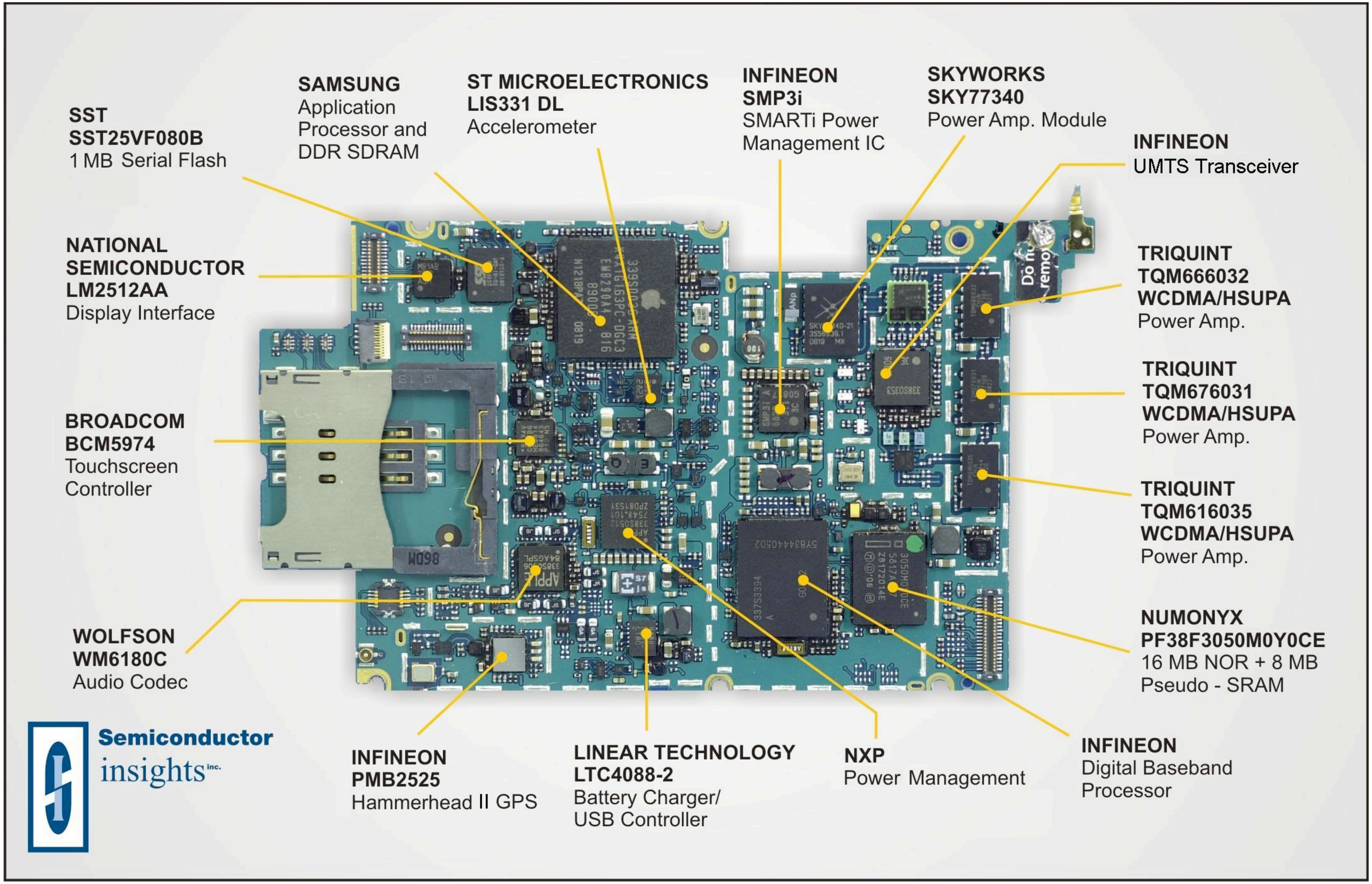
- Multiprocessor
  - 8-bit/32-bit for UI; DSP for signals
  - 32-bit in IR port; 32-bit in Bluetooth
- 8-100 MB of memory
- All custom chips



- Massive signal processing
  - Several processing tasks per connected call
- Based on DSPs
  - Standard or custom
  - 100s of processors

Embedded systems from real life

Embedded systems from real life



Embedded systems from real life

Cars

Multiple processors
- Up to 100
- Networked together

Multiple networks
- Body, engine, telematics, media, safety

Large diversity in processor types:
- 8-bit – door locks, lights, etc.
- 16-bit – most functions
- 32-bit – engine control, airbags

Functions by embedded processing:
- ABS: Anti-lock braking systems
- ESP: Electronic stability control
- Airbags
- Efficient automatic gearboxes
- Theft prevention with smart keys
- Blind-angle alert systems
- ... etc ...

Embedded systems from real life

Extremely Large

Functions requiring computers:
- Radar
- Weapons
- Damage control
- Navigation
- basically everything

Computers:
- Large servers
- 1000s of processors

Embedded systems from real life

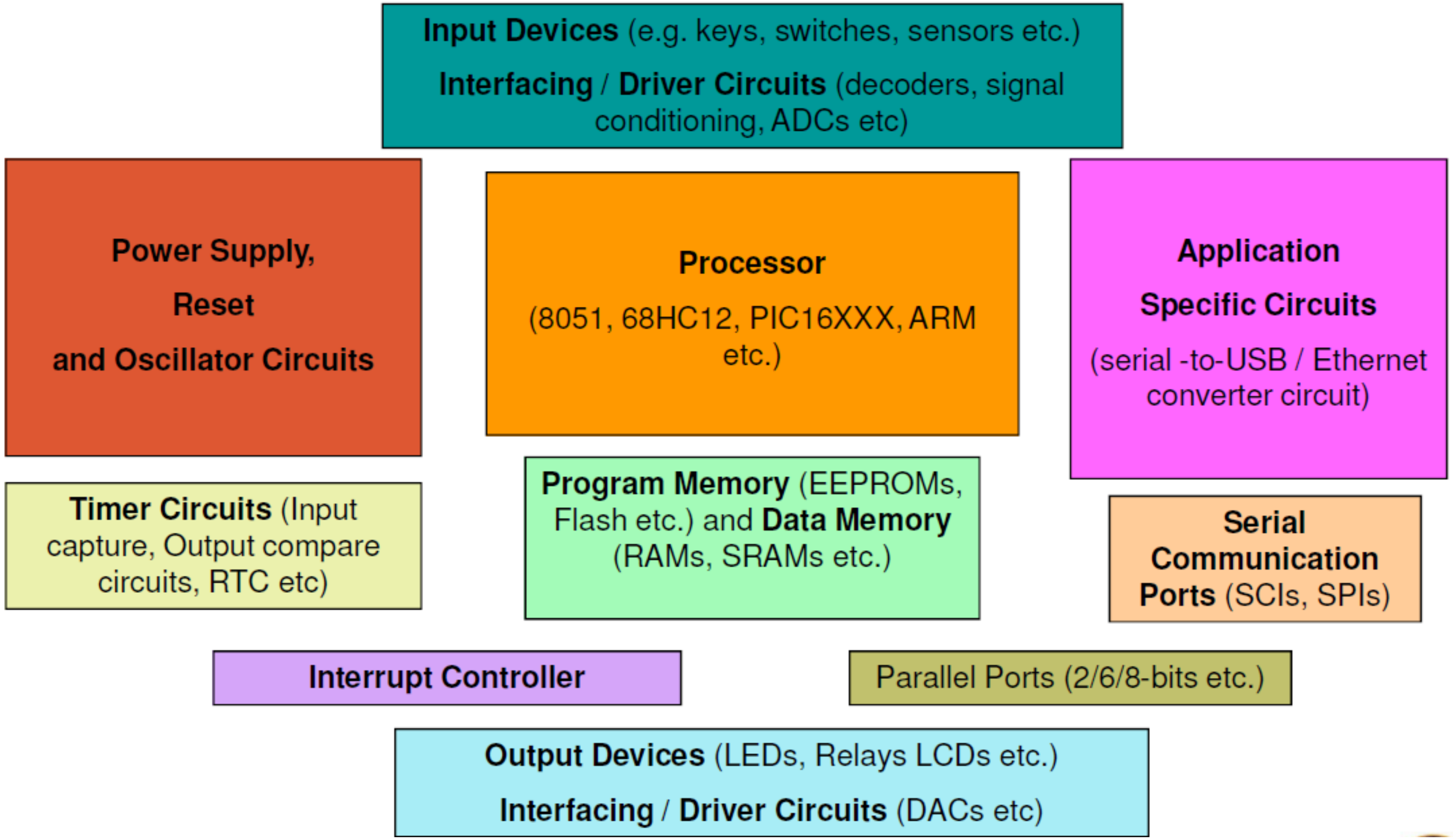
Inside Your PC

- Custom processors
  - Graphics, sound
- 32-bit processors
  - IR, Bluetooth
  - Network, WLAN
  - Harddisk
  - RAID controllers
- 8-bit processors
  - USB
  - Keyboard, mouse

Components of Embedded Systems

- ❑ Analog Components
  - ✓ Sensors, Actuators, Controllers, …
- ❑ Digital Components
  - ✓ Processor, Coprocessors
  - ✓ Memories
  - ✓ Controllers, Buses
  - ✓ Application Specific Integrated Circuits (ASIC)
- ❑ Converters – A2D, D2A, …
- ❑ Software
  - ✓ Application Programs
  - ✓ Exception Handlers

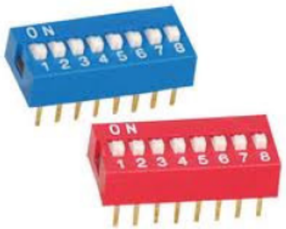
Components of Embedded Systems



Components of Embedded Systems

Input Devices

Switches

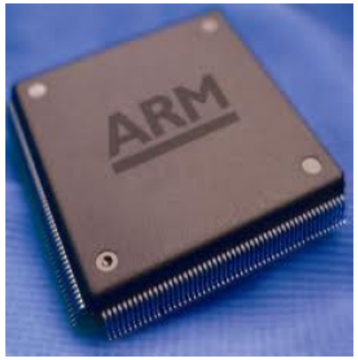


Keypads

Sensors

Components of Embedded Systems

Processors



Components of Embedded Systems

- **1) Main application software:**
  - ✓ performs series of tasks or multiple tasks concurrently.
  - ✓ constrained due to low memory, low processing power requirement etc.
- **2. Real Time Operating System (RTOS):**
  - ✓ supervises the application software.
  - ✓ provides a mechanism to let the processor run a process as per scheduling.
  - ✓ performs context-switching between various processes (tasks).
  - ✓ organizes access to a resource in sequence of the series of tasks of the system.
  - ✓ schedules their working and execution by following a plan to control the latencies and to meet the deadlines.
  - ✓ sets the rules during the execution of the application software.

Growing importance of embedded systems

- Growing economical importance of embedded systems: [www.itfacts.biz]
  - *Worldwide mobile phone sales surpassed 156.4 mln units in Q2 2004, a 35% increase from Q2 2003.*
  - *The worldwide portable flash player market exploded in 2003 and is expected to grow from 12.5 mln units in 2003 to over 50 mln units in 2008.*
  - *Global 3G subscribers will grow from an estimated 45 mln at the end of 2004 to 85 mln in 2005.*
  - *The number of broadband lines worldwide increased by almost 55% to over 123 mln in the 12 months to the end of June 2004.*
  - *Today's DVR (digital video recorders) users - 5% of households - will grow to 41% within five years.*
  - 79% of all high-end processors are used in embedded systems
  - ➢ The future is embedded, Embedded is the future!

Characteristics of Embedded Systems


Price


Functionality


Performance


Size

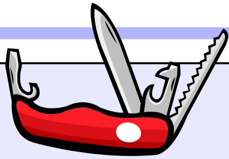

Power


Time-to-market

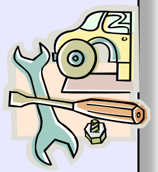

Maintainability


Safety

Characteristics of Embedded Systems (1)

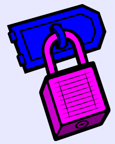


- Must be **dependable**,
  - **Reliability *R(t)* =** probability of system working correctly provided that is was working at t=0



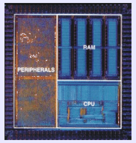
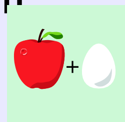
  - **Maintainability *M(d)* =** probability of system working correctly d time units after error occurred.
  - **Availability *A*(*t*)**: probability of system working at time t
  - **Safety**: no harm to be caused

  - **Security**: confidential and authentic communication



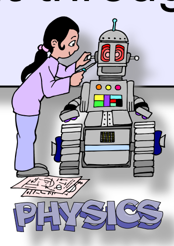
Even perfectly designed systems can fail if the assumptions about the workload and possible errors turn out to be wrong.
Making the system dependable must not be an after-thought, it must be considered from the very beginning

Characteristics of Embedded Systems (2)

- 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
- **Dedicated user interface**
  (no mouse, keyboard and screen) 
- **Hybrid systems** (analog + digital parts).  

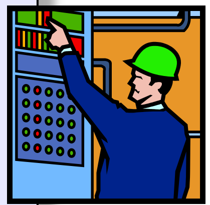
Characteristics of Embedded Systems (3)

- 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
- Frequently **connected to physical environment** through sensors and actuators,



Characteristics of Embedded Systems (4)

- 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.

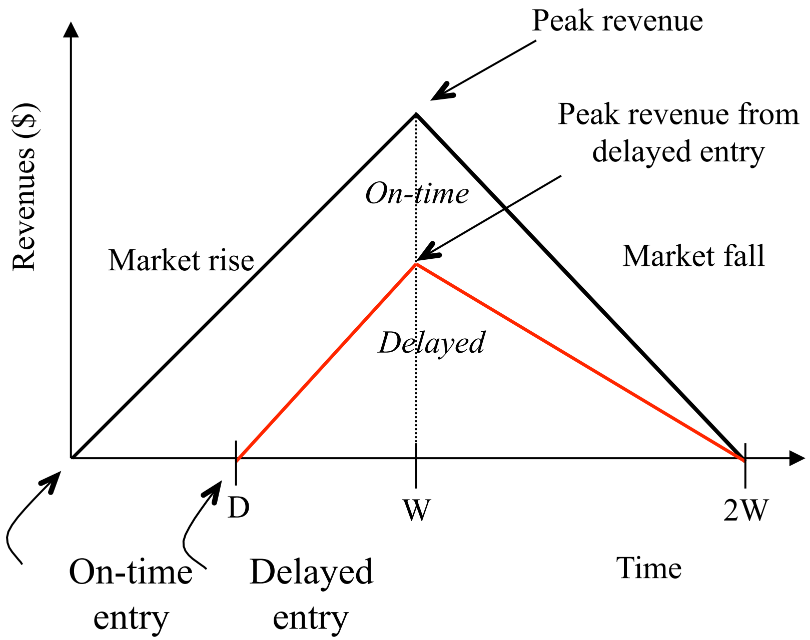


Not every ES has all of the above characteristics.

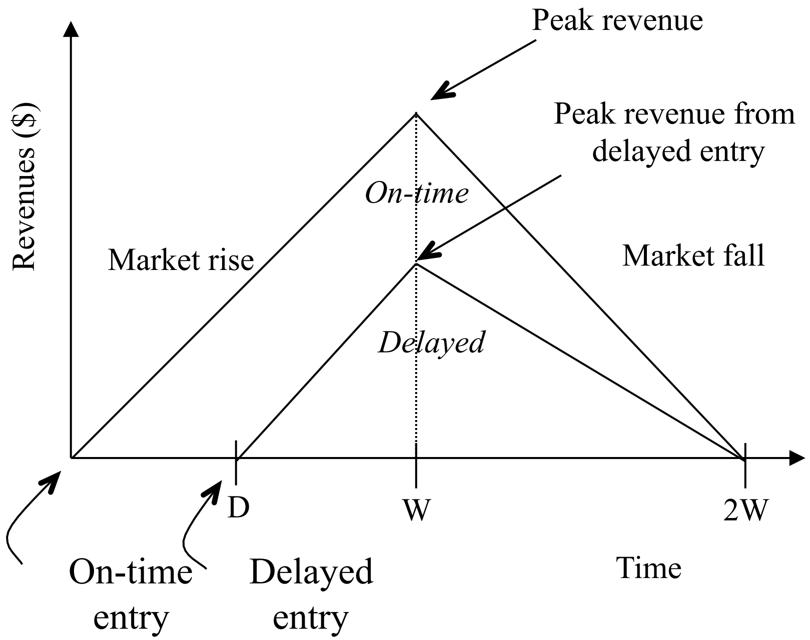
Def.: **Information processing systems having most of the above characteristics are called embedded systems.**

Time-to-market

- ❑ Often must meet tight deadlines.
  - ✓ 6 month market window is common.
  - ✓ Can't miss back-to-school window for calculator.



Losses Due to Delayed Market Entry



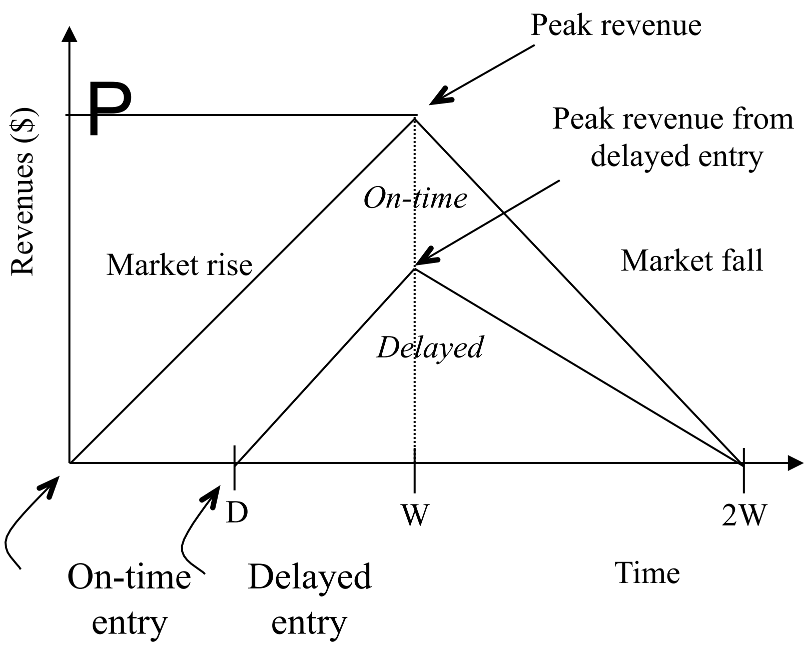
Simplified revenue model

- Product life = 2W, peak at W
- Time of market entry defines a triangle, representing market penetration
- Triangle area equals revenue

Loss

- The difference between the on-time and delayed triangle areas

Losses Due to Delayed Market Entry, cont'd.



Area = 1/2 * base * height

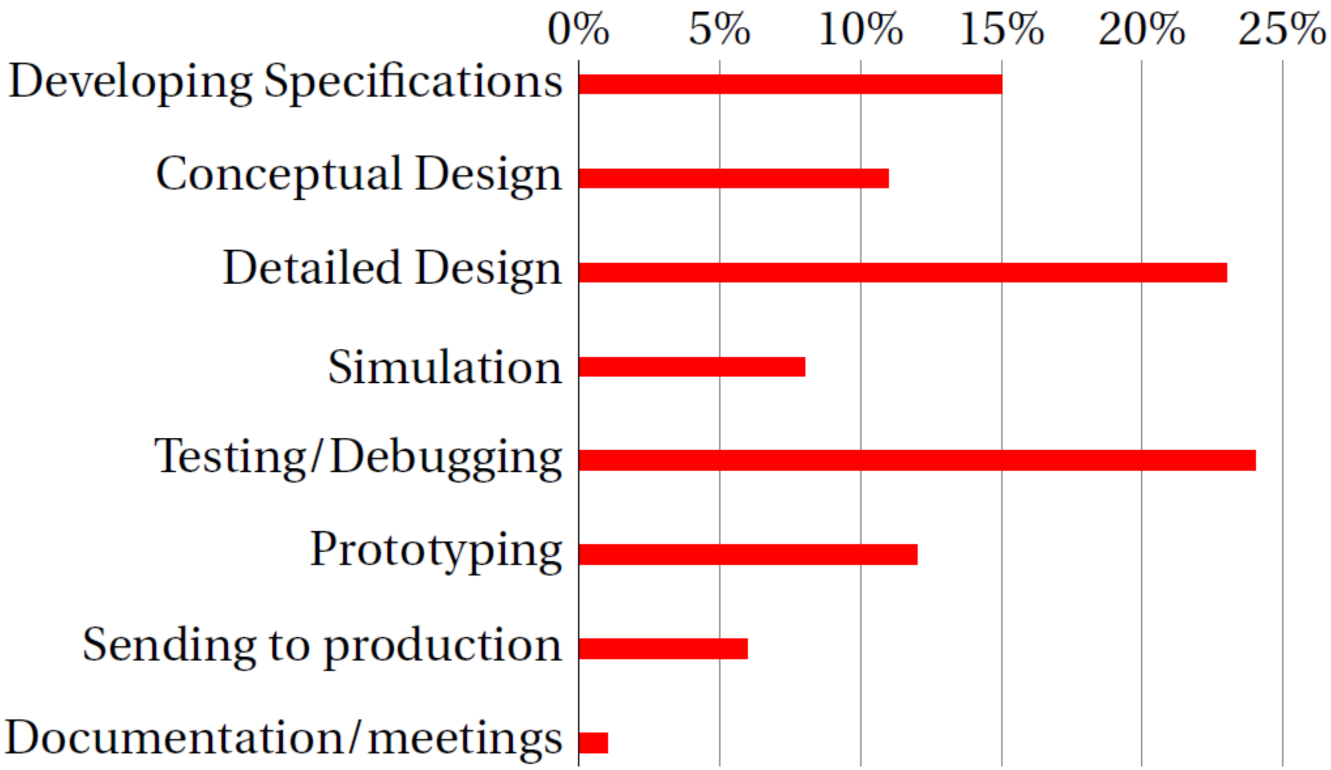
- On-time = 1/2 * 2W * P
- Delayed = 1/2 * (W-D+W)*(W-D)*P/W

Percentage revenue loss = (D(3W-D)/ $2W^2$)*100%

Try some examples

- Lifetime 2W=52 wks, delay D=4 wks
- (4*(3*26 –4)/2*26^2) = 22%
- Lifetime 2W=52 wks, delay D=10 wks
- (10*(3*26 –10)/2*26^2) = 50%
- Delays are costly!

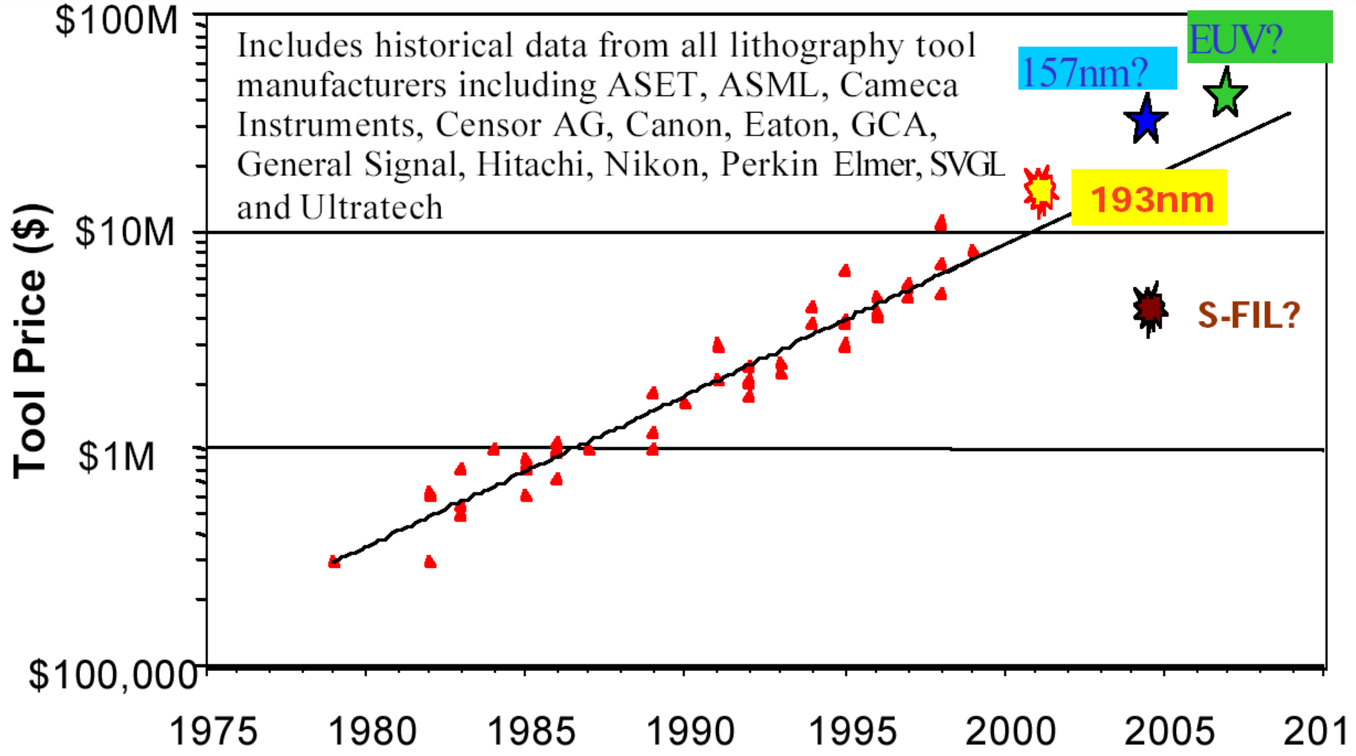
Development costs



Source: 2009 Embedded Market Study

Challenges for implementation in hardware

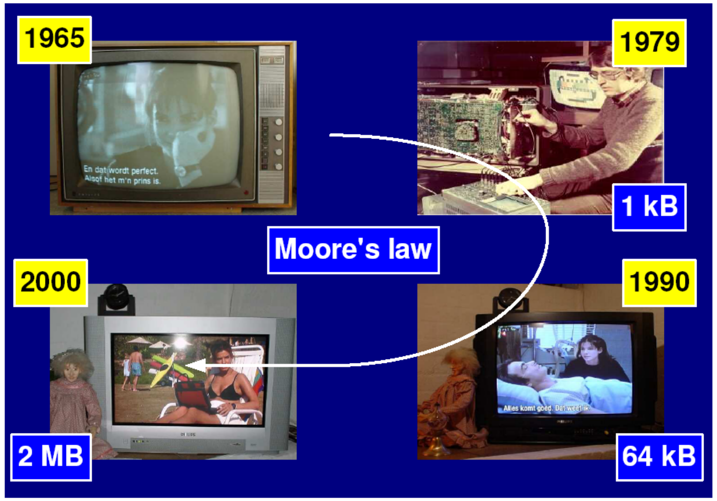
- Lack of flexibility (changing standards).
- Mask cost for specialized HW becomes very expensive



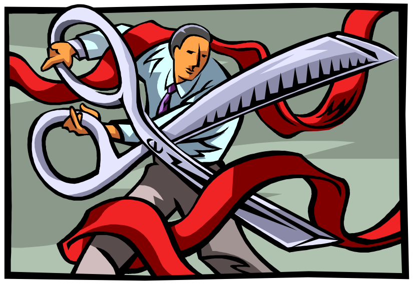
☞Trend towards implementation in Software

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 in embedded system design

- ❑ How much hardware do we need?
  - ✓ How many processors? How big are they? How much memory?
- ❑ How do we meet performance requirements?
  - ✓ What's in hardware? What's in software?
  - ✓ Faster hardware or cleverer software?
- ❑ How do we minimize power?
  - ✓ Turn off unnecessary logic? Reduce memory accesses?
- ❑ How do we ship in time?
  - ✓ Off-the-shelf chips? IP-reuse?

More 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)

Challenges, etc.

- ❑ Does it really work?
  - ✓ Is the specification correct?
  - ✓ Does the implementation meet the spec?
  - ✓ How do we test for real-time characteristics?
  - ✓ How do we test on real data?
- ❑ How do we work on the system?
  - ✓ Observability, controllability?
  - ✓ What is our development platform?
- ❑ How do we make our ends meet?
- ❑ How do we reduce size/weight?

Departamento de Electrónica

Universidad de Alcalá

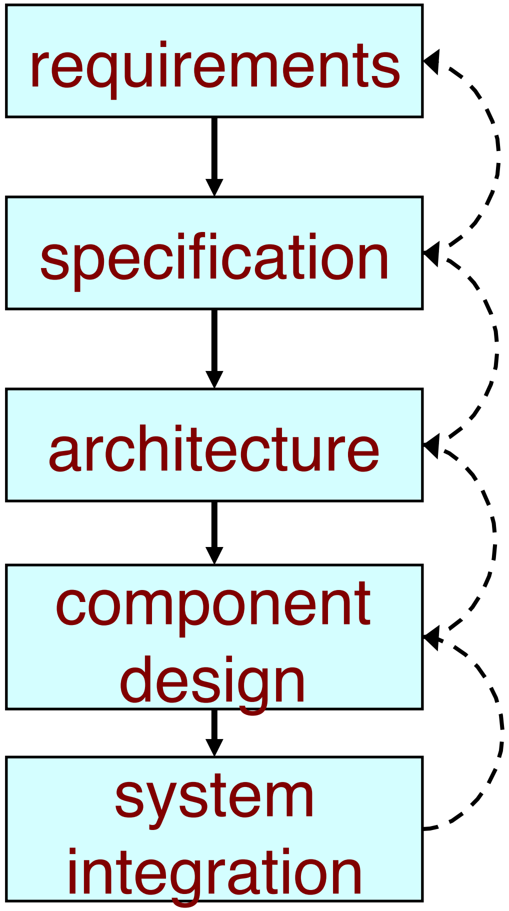
Required Designers

- Expertise with both **software and hardware** is needed to optimize design metrics
  - ✓ Not just a hardware or software expert
  - ✓ A designer must be comfortable with various technologies in order to choose the best for a given application and constraints
  - ✓ A designer must be able to communicate with teammates of various background

Design methodologies

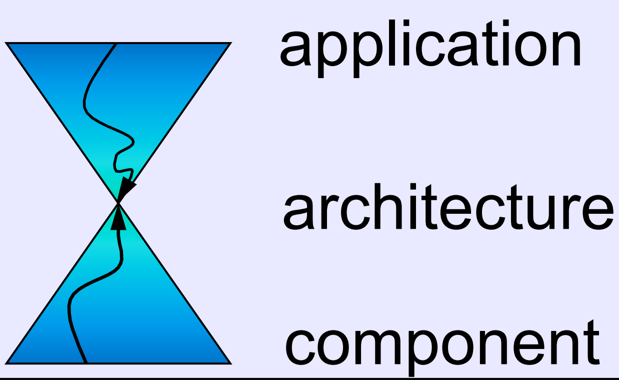
- A procedure for designing a system.
- Understanding your methodology helps you ensure you didn't skip anything.
- Compilers, software engineering tools, computer-aided design (CAD) tools, etc., can be used to:
  - ✓ help automate methodology steps;
  - ✓ keep track of the methodology itself.

Levels of abstraction



Top-down vs. bottom-up

- Top-down design:
  - ✓ start from most abstract description;
  - ✓ work to most detailed.
- Bottom-up design:
  - ✓ work from small components to big system.
- Real design uses both techniques.



Requirements

- Plain language description of what the user wants and expects to get.
- May be developed in several ways:
  - ✓ talking directly to customers;
  - ✓ talking to marketing representatives;
  - ✓ providing prototypes to users for comment.

Functional vs. non-functional requirements

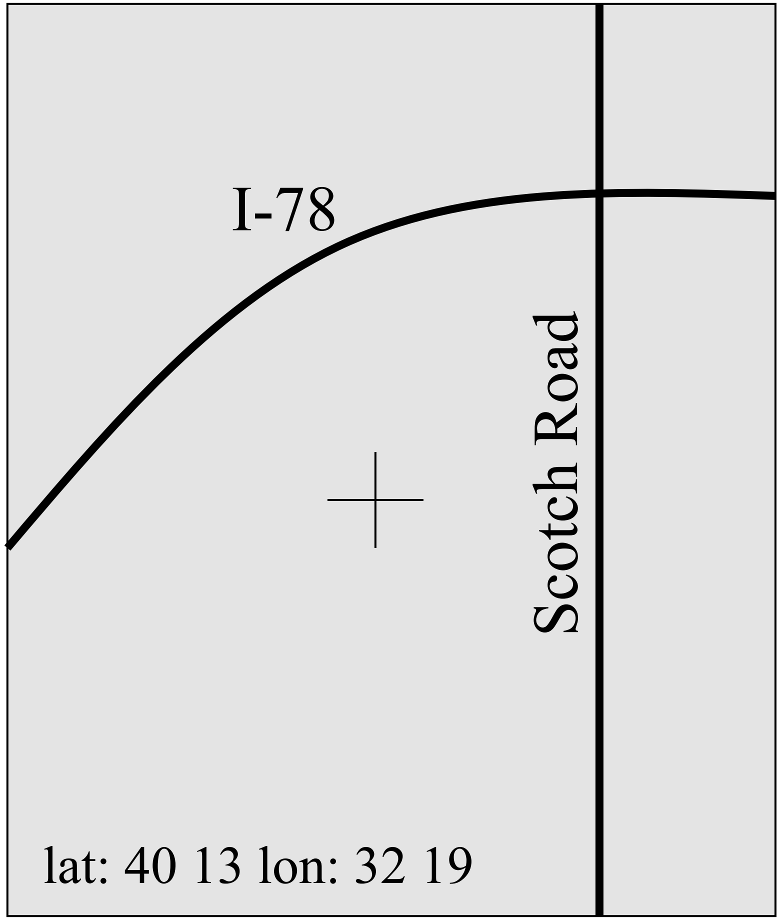
- Functional requirements:
  - ✓ output as a function of input.
- Non-functional requirements:
  - ✓ time required to compute output;
  - ✓ size, weight, etc.;
  - ✓ power consumption;
  - ✓ reliability;
  - ✓ etc.

One requirements form

name
purpose
inputs
outputs
functions
performance
manufacturing cost
power
physical size/weight

Example: GPS moving map requirements

- ❑ Moving map obtains position from GPS, paints map from local database.



GPS moving map needs

- Functionality: For automotive use. Show major roads and landmarks.
- User interface: At least 400 x 600 pixel screen. Three buttons max. Pop-up menu.
- Performance: Map should scroll smoothly. No more than 1 sec power-up. Lock onto GPS within 15 seconds.
- Cost: $500 street price = approx. $100 cost of goods sold.
- Physical size/weight: Should fit in dashboard.
- Power consumption: 8 hours on 4 AAs

GPS moving map requirements form

name	GPS moving map
purpose	consumer-grade moving map for driving
inputs	power button, two control buttons
outputs	back-lit LCD 400 X 600
functions	5-receiver GPS; three resolutions; displays current lat/lon
performance	updates screen within 0.25 sec of movement
manufacturing cost	$100 cost-of-goods-sold
power	100 mW
physical size/weight	no more than 2: X 6:, 12 oz.

Specification

- A more precise description of the system:
  - ✓ should not imply a particular architecture;
  - ✓ provides input to the architecture design process.
- May include functional and non-functional elements.
- May be executable or may be in mathematical form for proofs.

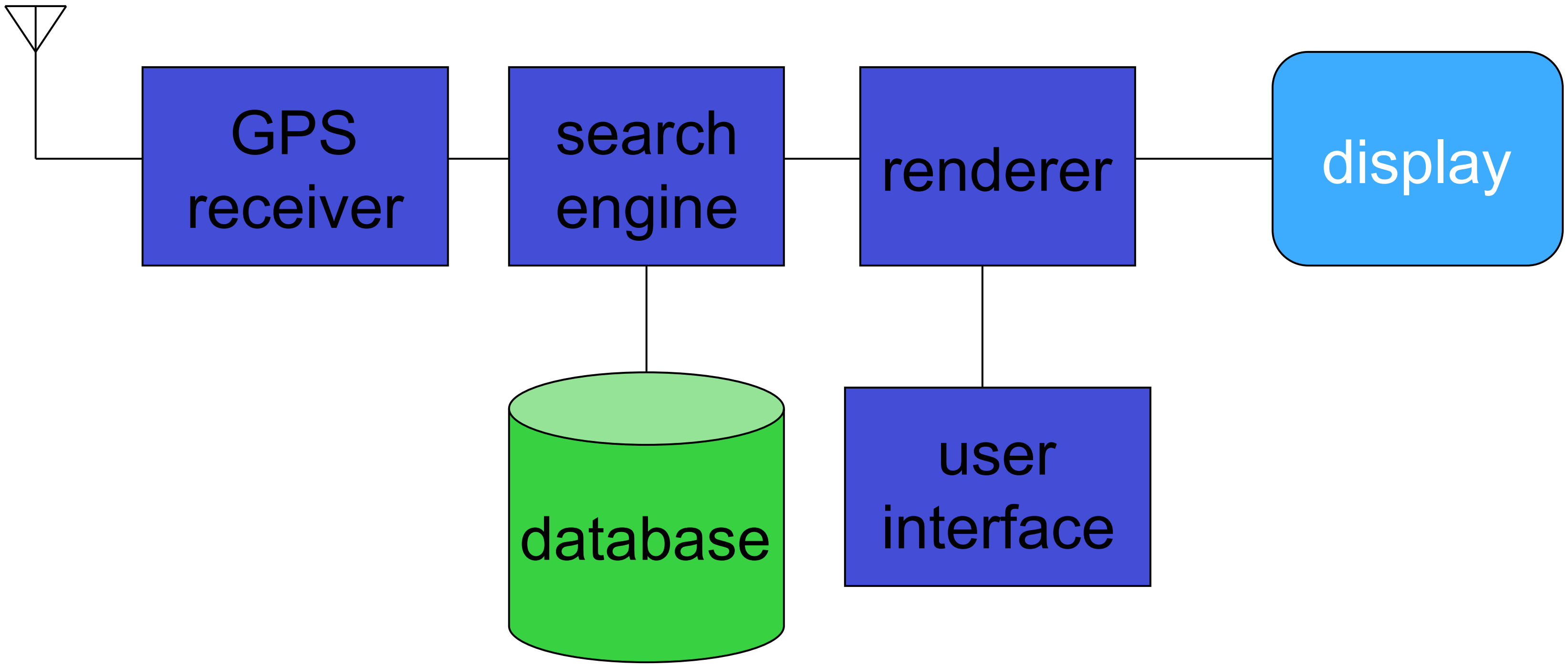
GPS specification

- Should include:
  - ✓ What is received from GPS;
  - ✓ map data;
  - ✓ user interface;
  - ✓ operations required to satisfy user requests;
  - ✓ background operations needed to keep the system running.

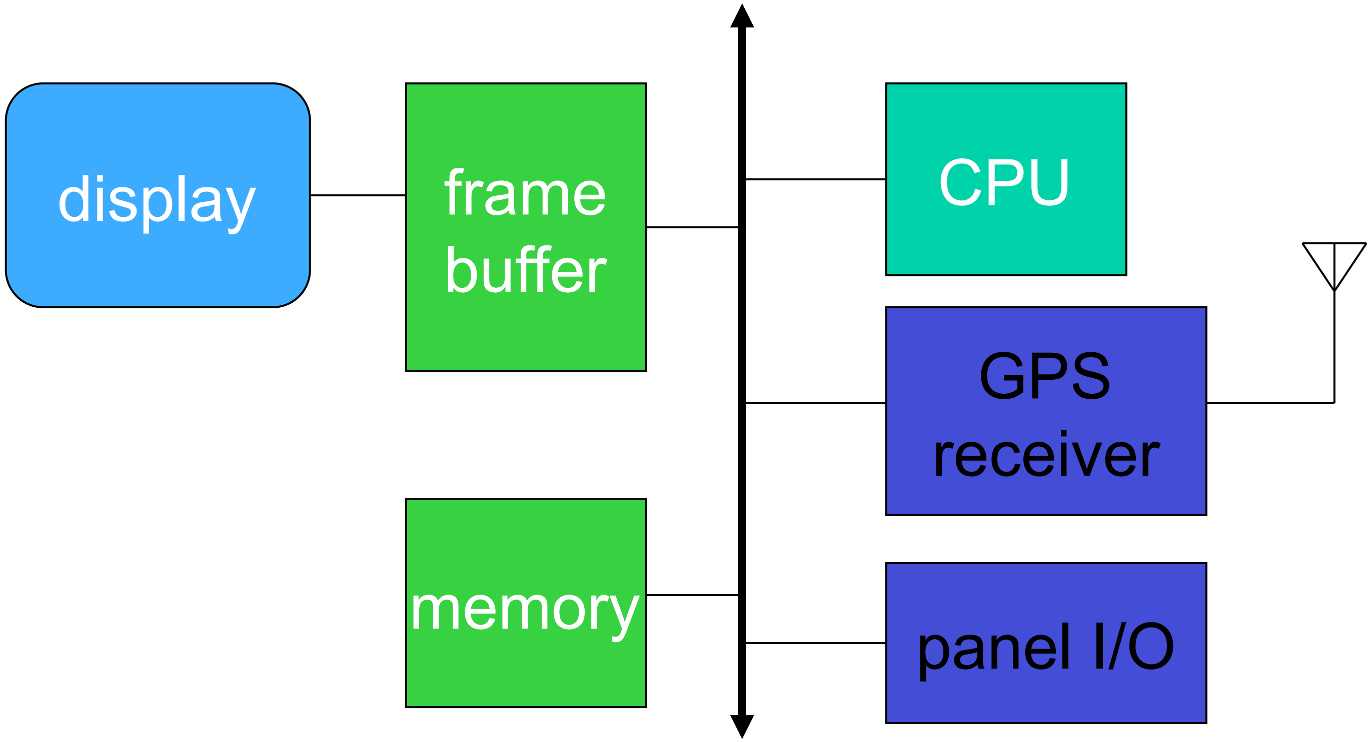
Architecture design

- ❑ What major components go satisfying the specification?
- ❑ Hardware components:
  - ✓ CPUs, peripherals, etc.
- ❑ Software components:
  - ✓ major programs and their operations.
- ❑ Must take into account functional and non-functional specifications.

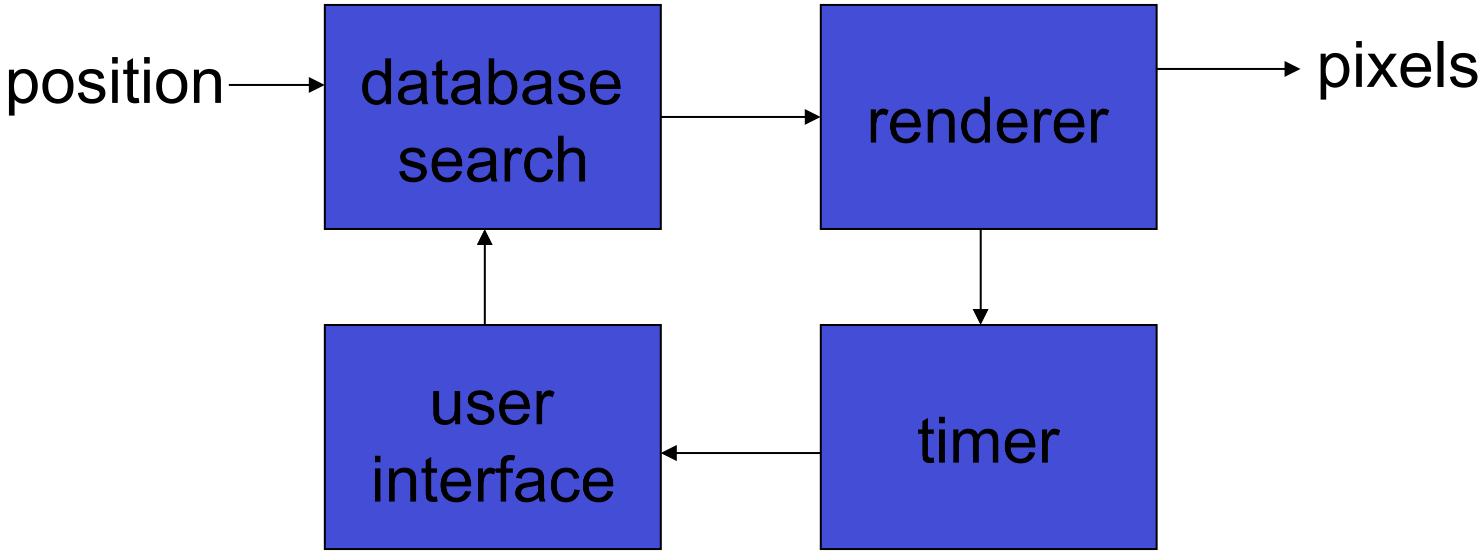
GPS moving map block diagram



GPS moving map hardware architecture



GPS moving map software architecture



Designing hardware and software components

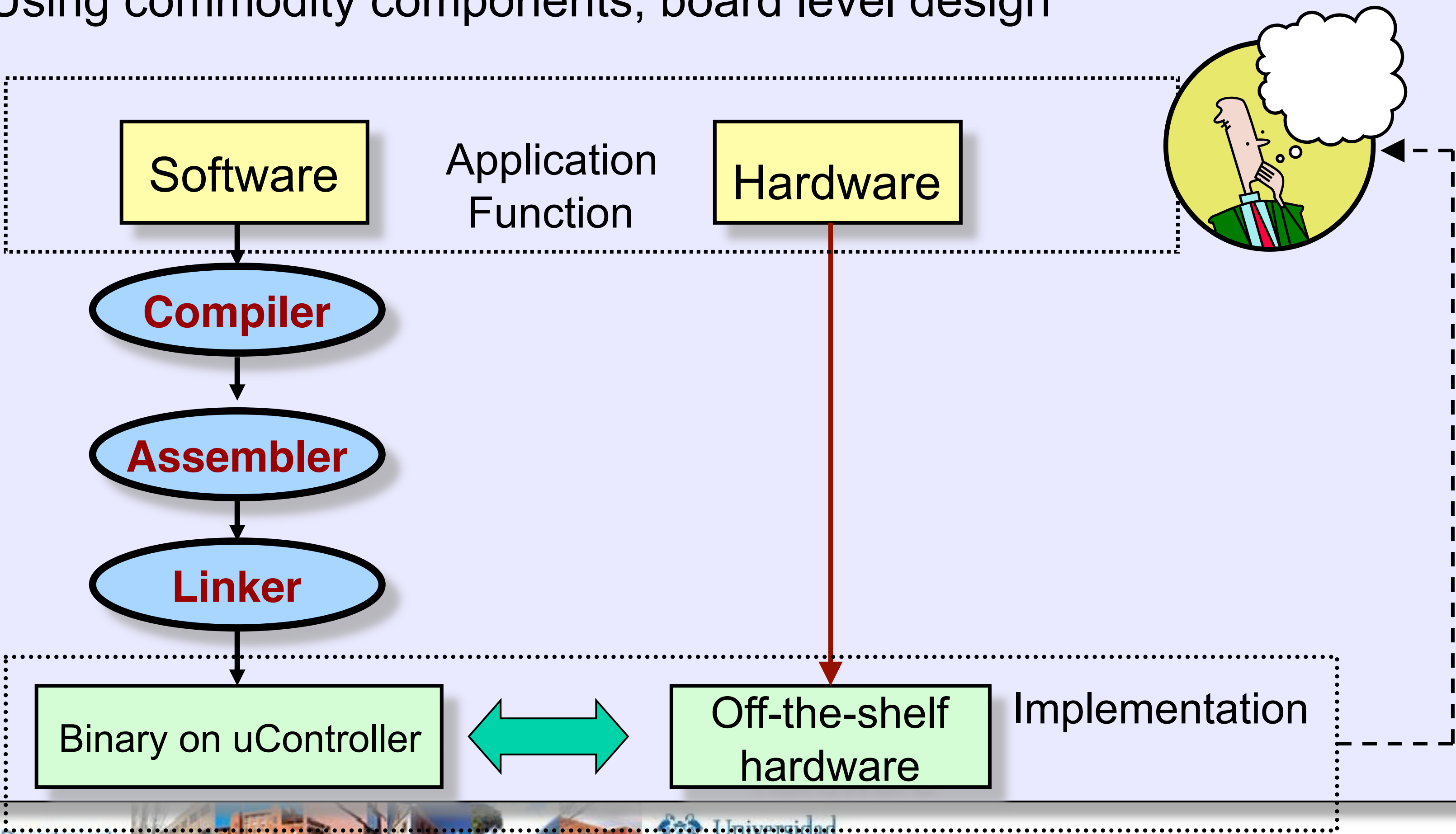
- Must spend time architecting the system before you start coding.
- Some components are ready-made, some can be modified from existing designs, others must be designed from scratch.

System integration

- ❑ Put together the components.
  - ✓ Many bugs appear only at this stage.
- ❑ Have a plan for integrating components to uncover bugs quickly, test as much functionality as early as possible.

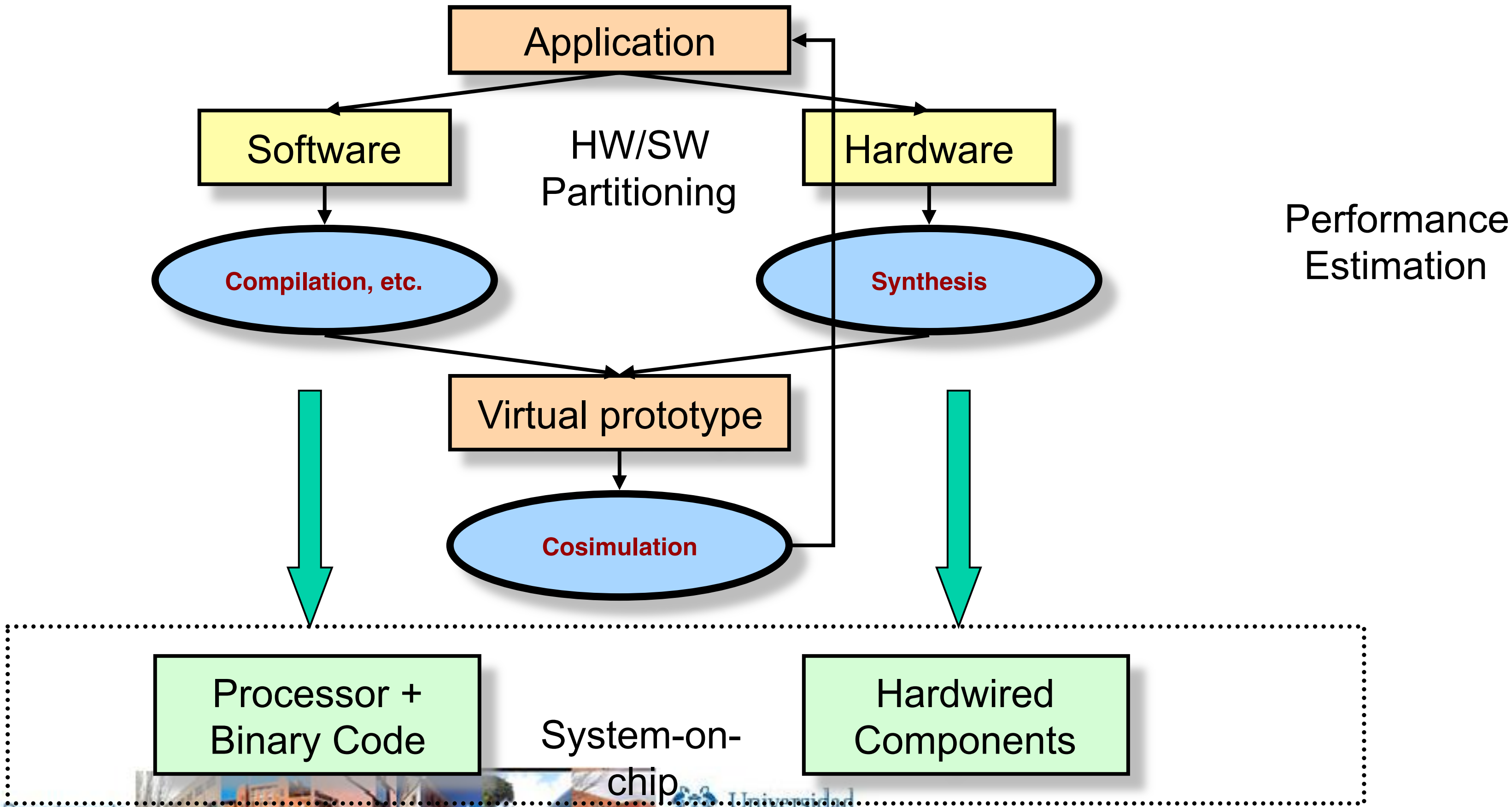
Traditional Embedded System Design

- SW and HW partitioning is decided at an early stage
- Using commodity components, board level design



HW/SW Codesign Style

- An integrated design flow of hardware and software
- Often targeting SoCs, not constrained by commodity components



Summary

- ❑ Embedded computers are all around us.
  - ✓ Many systems have complex embedded hardware and software.
- ❑ Embedded systems pose many design challenges: design time, deadlines, power, etc.
- ❑ Design methodologies help us manage the design process.