Sistemi Embedded
Introduzione

Riferimenti bibliografici

"Embedded System Design: A Unified Hardware/Software Introduction", Frank Vahid, Tony Givargis, John Wiley & Sons Inc., ISBN:0-471-38678-2, 2002.

"Computers as Components: Principles of Embedded Computer Systems Design", Wayne Wolf, Morgan Kaufmann Publishers, ISBN: 1-55860-541-X, 2001

Embedded systems overview

- Computing systems are everywhere
- Most of us think of "desktop" computers
  - PC's
  - Laptops
  - Mainframes
  - Servers
- But there's another type of computing system
  - Far more common...

Embedded systems overview

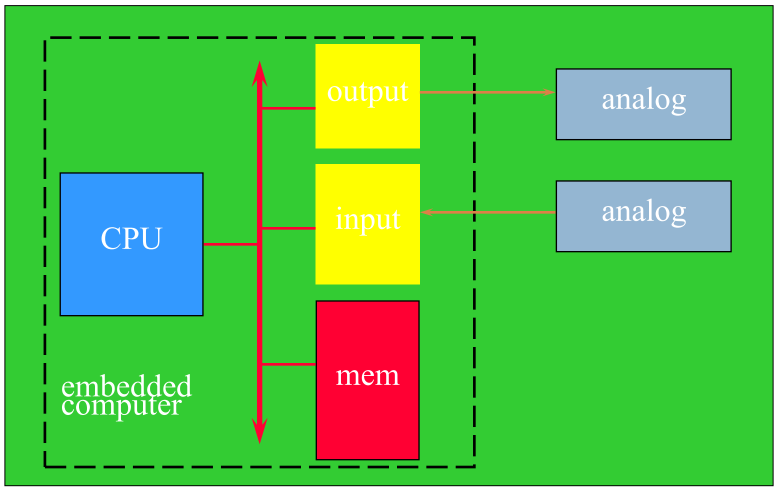
- Embedded computing systems
  - Computing systems embedded within electronic devices
  - Hard to define. Nearly any computing system other than a desktop computer
  - Billions of units produced yearly, versus millions of desktop units
  - Perhaps 50 per automobile

Embedded Systems

Embedded system: any device that includes a programmable computer but is not itself a general-purpose computer.

Take advantage of application characteristics to optimize the design

Embedding a computer



Application areas (1)

- Automotive electronics

- Aircraft electronics

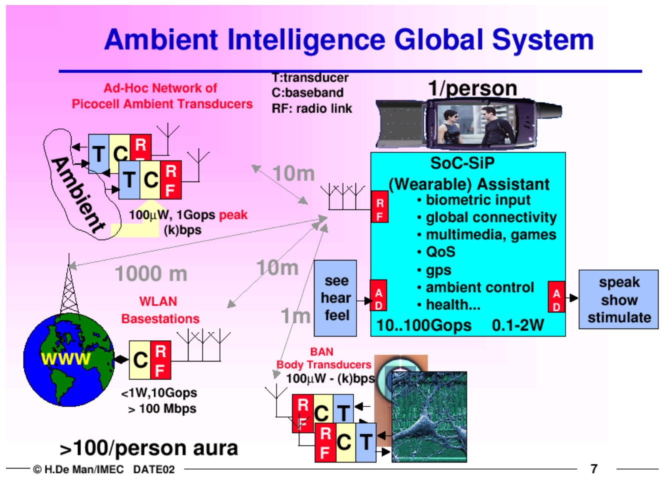
- Trains

- Telecommunication

- Military applications

Application areas (2)

- Consumer electronics



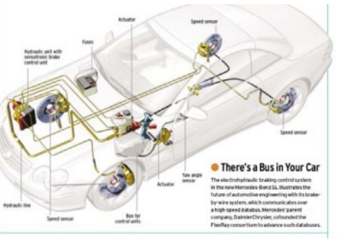
A “short list” of embedded systems

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

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

Cars

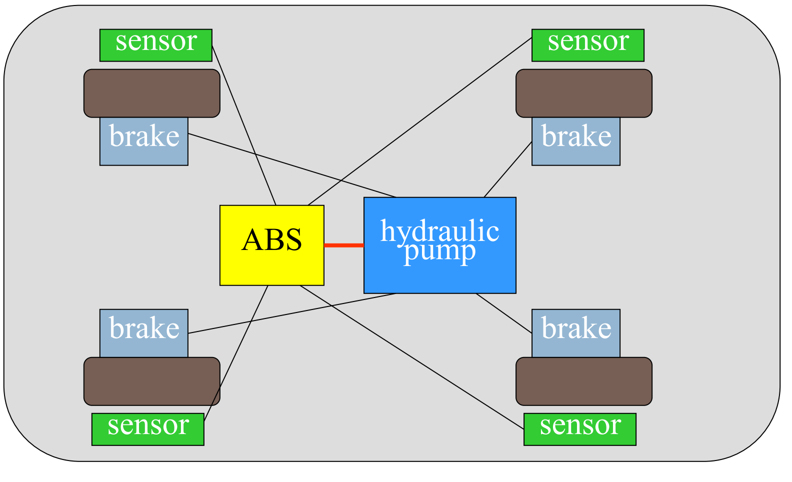
- Multiple processors
  - ✓ Up to 100
  - ✓ Networked together
- Multiple networks



–Functions:

- ABS: Anti-lock braking systems
- ESP: Electronic stability control
- Airbags
- Theft prevention with smart keys
- Blind-angle alert systems
- ... etc ...

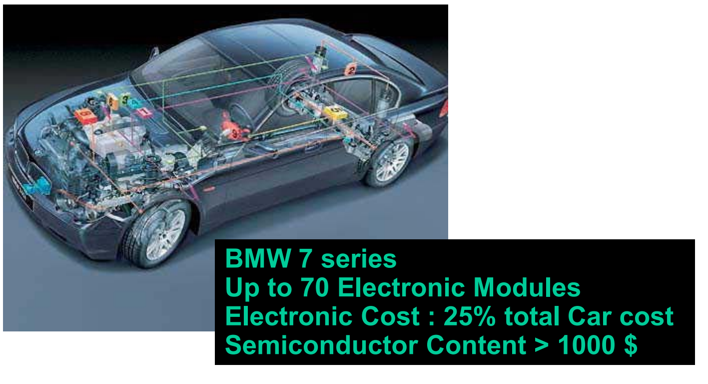
BMW 850i



Cars

- Large diversity in processor types:
  - 8-bit – door locks, lights, etc.
  - 16-bit – most functions
  - 32-bit – engine control, airbags
- Form follows function
  - Processing where the action is
  - Sensors and actuators distributed all over the vehicle

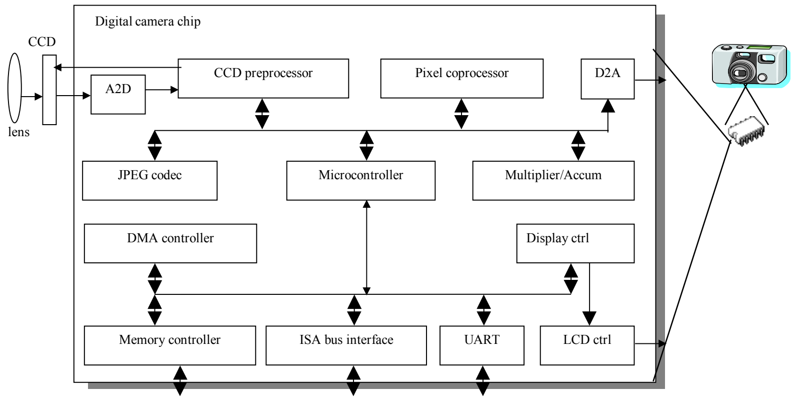
Cars



Some common characteristics of embedded systems

- Single-functioned
  - Executes a single program, repeatedly
- Tightly-constrained
  - Low cost, low power, small, fast, etc.
- Reactive and real-time
  - Continually reacts to changes in the system’s environment
  - Must compute certain results in real-time without delay

Digital Camera



- Single-functioned -- always a digital camera
- Tightly-constrained -- Low cost, low power, small, fast
- Reactive and real-time

Some common characteristics of embedded systems

An embedded system is designed to perform one or a few specific applications:

- The applications to be executed are known at design time
- It is often desirable flexibility of the system for future updates or for re-use of the component. Normally this goal is obtained by making the system reprogrammable
- Often have to run sophisticated algorithms or multiple algorithms.
  - Cell phone, laser printer.

Characteristics of embedded systems

- Embedded Systems interact with the physical environment. They include devices such as sensors and actuators
  - Transducers, sensors and actuators are essential enabling technologies for embedded systems
    - MEMS (microelectromechanical sensors) accelerometers, gyroscopes, inertial modules, pressure sensors
- Embedded Systems are Hybrid Dystems (digital + analogic)
  – A/D and D/A are included
- Dedicated user interface:
  - no mouse, keyboard and screen
  - display with reduced size
  – reduced number on I/O devices

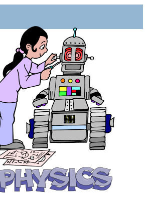
Characteristics of embedded systems

- Some functions are more efficiently executed using dedicated hardware devices such as DSP, IP cell, etc.
- Typical DSP applications:
  – generic signals : filtering, DFT, FFT, etc.
  – voice: encoding, decoding, equalization, etc.
  – modem: modulation, demodulation

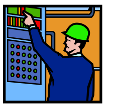
Characteristics of embedded systems

- Many ES must meet **real-time constraints**
- 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.
- Must finish operations by deadlines.
  - Hard real time: missing deadline causes failure.
  - Soft real time: missing deadline results in degraded performance.
- Many systems are multi-rate: must handle operations at widely varying rates.

Characteristics of embedded systems



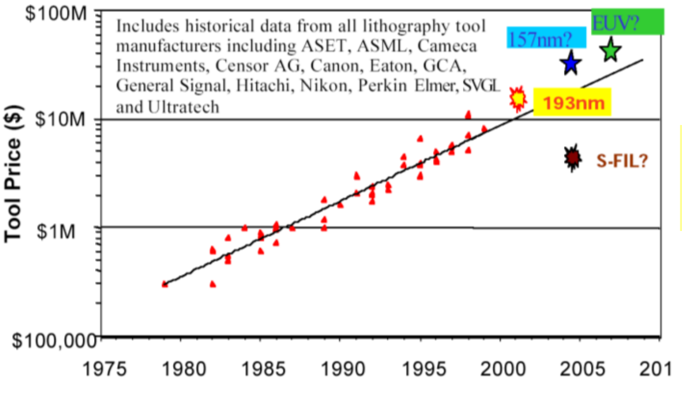
- Typically, ES are **reactive systems**:
  "**A reactive system is one which is in continual interaction with is environment and executes at a pace determined by that environment"** [Bergé, 1995]



- Behavior depends on input **and current state.**
  - ✓ automata model appropriate,

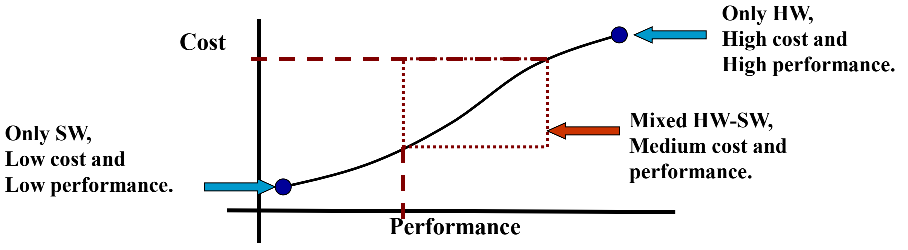
Challenges for implementation in hardware

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



☞Trend towards implementation in Software

Heterogeneous HW/SW Implementations of Embedded Systems



Additionally, flexibility and tight time to market requirements favour SW implementations.

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?

- 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*

Challenges for Embedded Software

- How can we capture the required behavior of complex systems ?
- How do we validate specifications?
- How do we translate specifications efficiently into implementation?
- Do software engineers ever consider power dissipation?
- How can we check that we meet real-time constraints?
- Which programming language provides real-time features?
- How do we validate embedded real-time software? (large volumes of data)

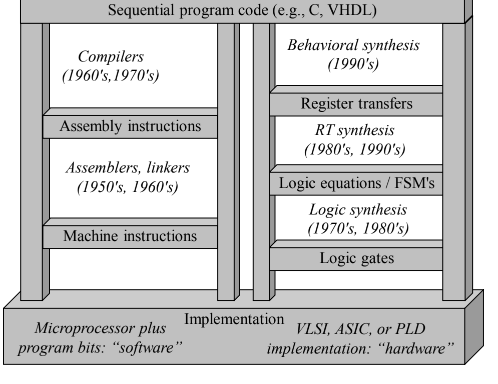
Challenges for Embedded Software

- 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

Co-design

In the past:
- – Hardware and software design technologies were very different
- – Recent maturation of synthesis enables a unified view of hardware and software

• Now:
- – Hardware/software "codesign"



The choice of hardware versus software for a particular function is simply a tradeoff among various design metrics. there is no fundamental difference between what hardware or software can implement

Design metrics

Design challenge – optimizing design metrics

- Obvious design goal:
  - Construct an implementation with desired functionality
- Key design challenge:
  - Simultaneously optimize numerous design metrics
- Design metric
  - A measurable feature of a system’s implementation
  - Optimizing design metrics is a key challenge

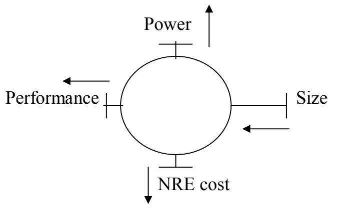
Design challenge – optimizing design metrics

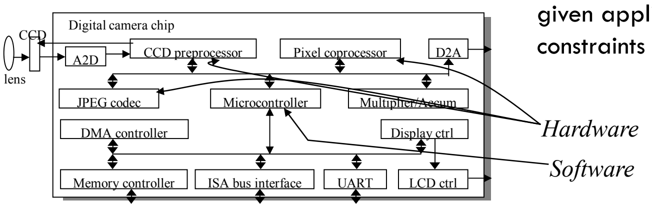
- Common metrics
  - **Unit cost:** the monetary cost of manufacturing each copy of the system, excluding NRE cost
  - **NRE cost (Non-Recurring Engineering cost):** The one-time monetary cost of designing the system
  - **Size:** the physical space required by the system
  - **Performance:** the execution time or throughput of the system
  - **Power:** the amount of power consumed by the system
  - **Flexibility:** the ability to change the functionality of the system without incurring heavy NRE cost

Design challenge – optimizing design metrics

- Common metrics (continued)
  - **Time-to-prototype:** the time needed to build a working version of the system
  - **Time-to-market:** the time required to develop a system to the point that it can be released and sold to customers
  - **Maintainability:** the ability to modify the system after its initial release
  - Correctness, safety, many more

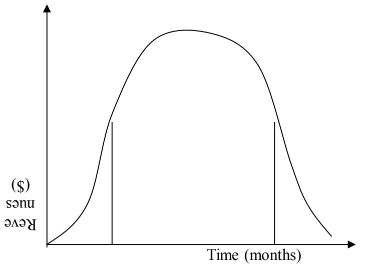
Design metric competition -- improving one may worsen others





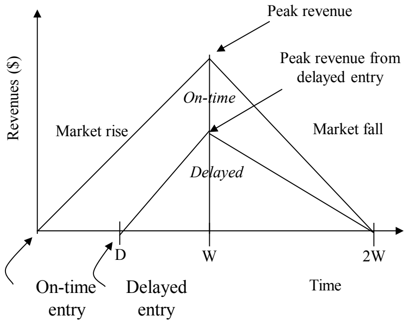
- Expertise with both **software and hardware** is needed to optimize design metrics
  - Not just a hardware or software expert, as is common
  - A designer must be comfortable with various technologies in order to choose the best for a given application and constraints

Time-to-market



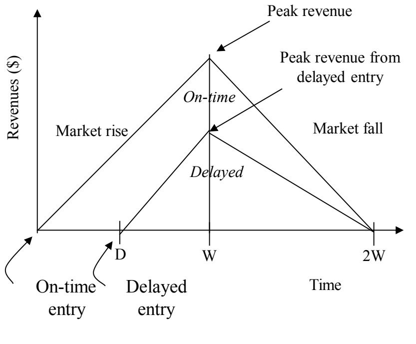
- Time required to develop a product to the point it can be sold to customers
- Market window
  - Period during which the product would have highest sales
- Average time-to-market constraint is about 8 months
- Delays can be costly

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



- Area = 1/2 * base * height
  - On-time = 1/2 * 2W * W
  - Delayed = 1/2 * (W-D+W)*(W-D)
- 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!

NRE and unit cost

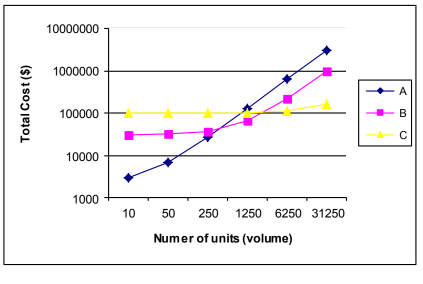
- Costs:
  - Unit cost: the monetary cost of manufacturing each copy of the system, excluding NRE cost
  - NRE cost (Non-Recurring Engineering cost): The one-time monetary cost of designing the system
  - *total cost = NRE cost + unit cost * # of units*
  - *per-product cost = total cost / # of units*
    = (NRE cost / # of units) + unit cost

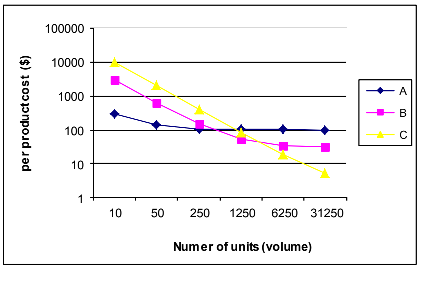
- Example
  - NRE=$2000, unit=$100
  - For 10 units
    - total cost = $2000 + 10*$100 = $3000
    - per-product cost = $2000/10 + $100 = $300

Amortizing NRE cost over the units results in an additional $200 per unit

NRE and unit cost

- Compare technologies by costs -- best depends on quantity
  - Technology A: NRE=$2,000, unit=$100
  - Technology B: NRE=$30,000, unit=$30
  - Technology C: NRE=$100,000, unit=$2





- But, must also consider time-to-market

The performance design metric

- Widely-used measure of system, widely-abused
  - Clock frequency, instructions per second – not good measures
  - Digital camera example – a user cares about how fast it processes images, not clock speed or instructions per second
- Latency (response time)
  - Time between task start and end
  - e.g., Camera's A and B process images in 0.25 seconds
- Throughput
  - Tasks per second, e.g. Camera A processes 4 images per second
  - Throughput can be more than latency seems to imply due to concurrency, e.g. Camera B may process 8 images per second (by capturing a new image while previous image is being stored).
- *Speedup* of B over S = B's performance / A's performance
  - Throughput speedup = 8/4 = 2