Embedded Systems Design: A Unified Hardware/Software Introduction

Introduction

ESD_Cap1 ++

Outline

- Embedded systems overview
  - What are they?
- Design challenge – optimizing design metrics
- Technologies
  - IC technologies
  - Processor technologies
  - Design technologies
- Summary

Embedded Systems Design: A Unified Hardware/Software Introduction

Embedded systems overview

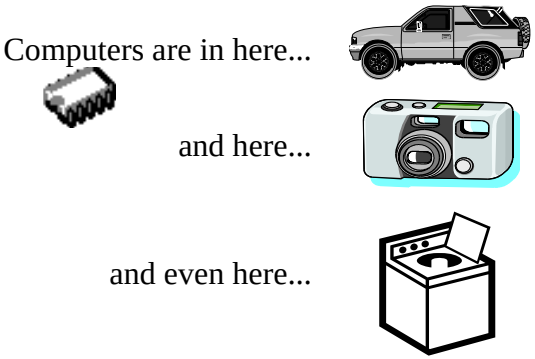
Embedded systems overview

- 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 household and per automobile



A “short list” of embedded systems

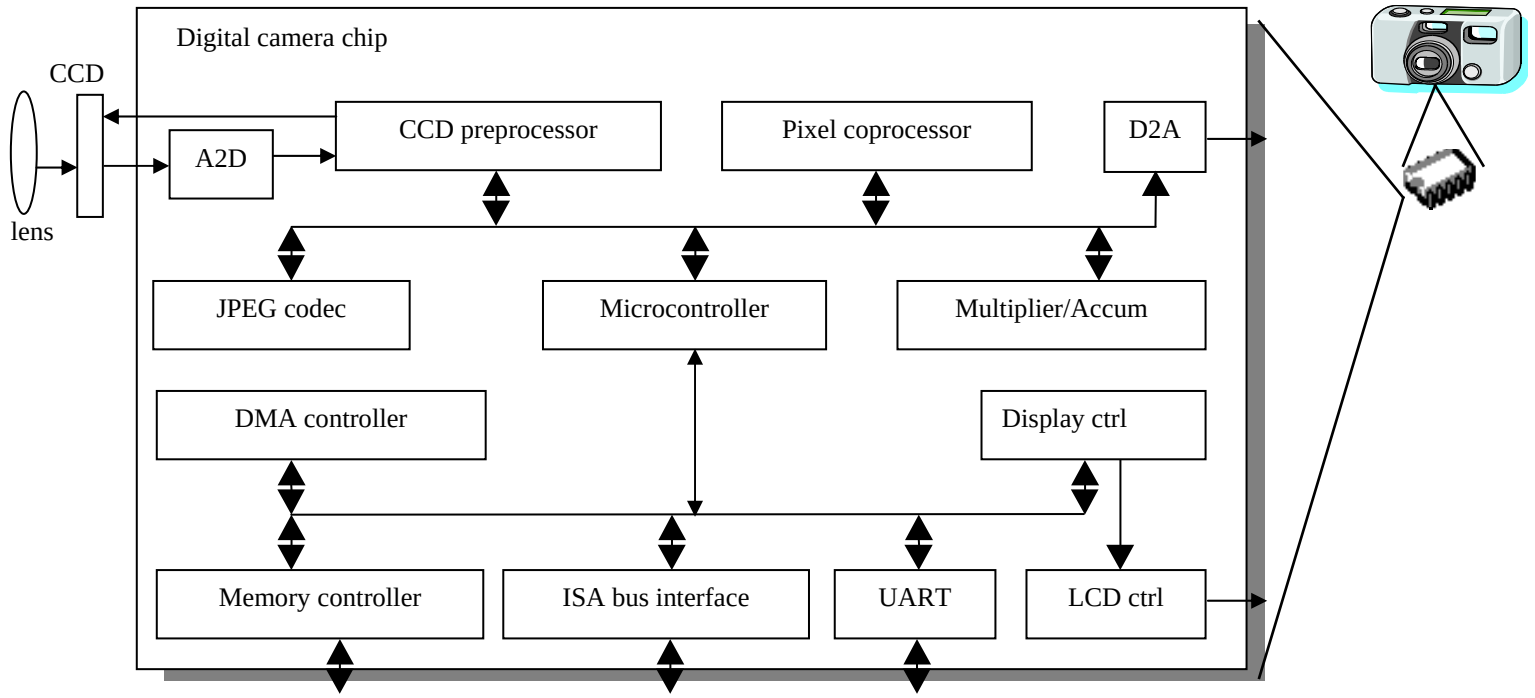
Anti-lock brakes
Auto-focus cameras
Automatic teller machines
Automatic toll systems
Automatic transmission
Avionic systems
Battery chargers
Camcorders
Cell phones
Cell-phone base stations
Cordless phones
Cruise control
Curbside check-in systems
Digital cameras
Disk drives
Electronic card readers
Electronic 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

And the list goes on and on

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 (not always)
  - Continually reacts to changes in the system's environment
  - Must compute certain results in real-time without delay

An embedded system example: a digital camera



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

Considerations

- An important feature of such systems, from a market point of view, is that they are often used to introduce innovation
  - Traditional products with innovative services

Considerations

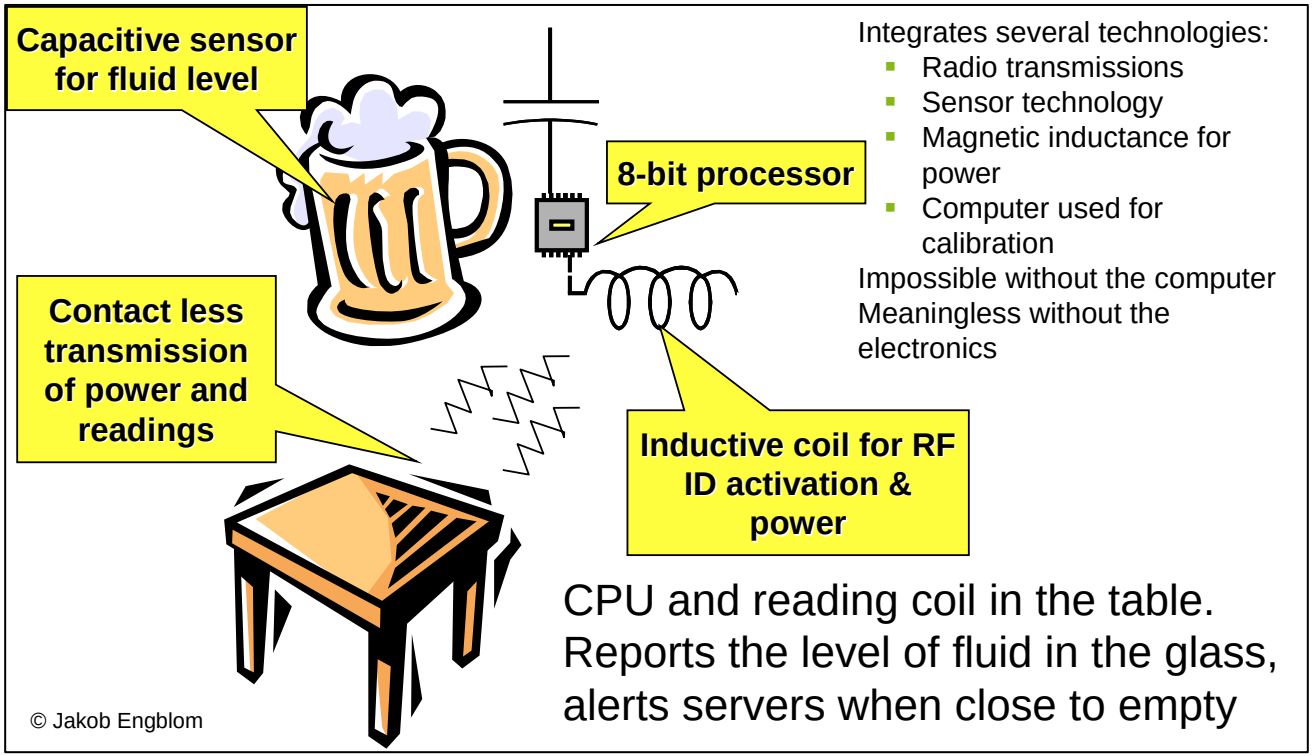
- Traditional products with innovative services
  - Example: *E-Tooth-Brush*

When the tooth brush communicates with the bathroom mirror

- animated cartoon appears on the mirror
- brushing the teeth becomes a computer game, the tooth brush becomes a joy stick
- high-score lists, collect rewarding points,...

Considerations

- Traditional products with innovative services
  - Example: *Smart Beer Glass*



Considerations

- Traditional products with innovative services
  - Example: *Angel Devil Touch*

Considerations

- Traditional products with innovative services
  - Example: *Remember Ring*

Considerations

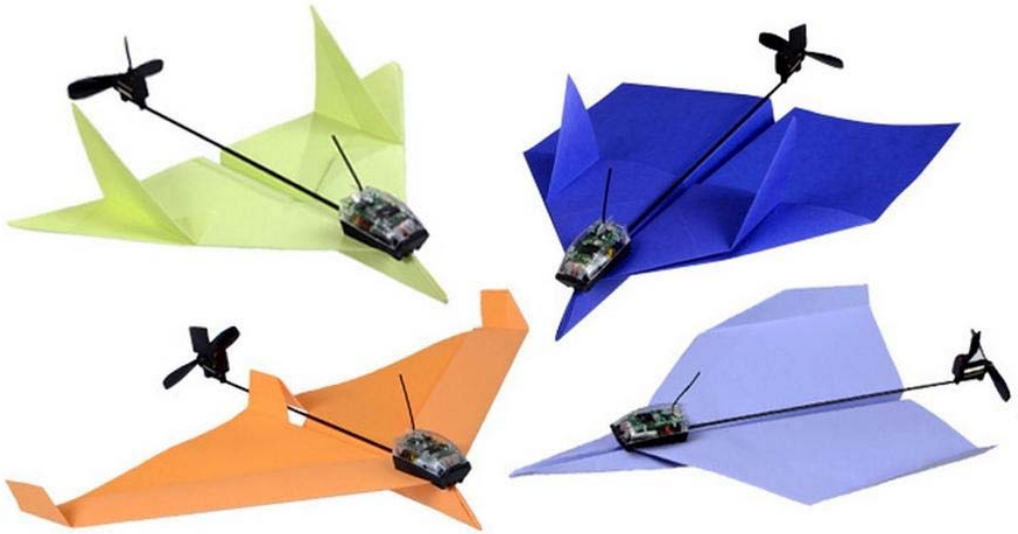
- Traditional products with innovative services
  - Example: *OSRAM DULUX EL SENSOR*

Considerations

- Traditional products with innovative services
  - Example: *PetZilla!*

Considerations

- Traditional products with innovative services
  - Example: *Smartphone Controlled Paper Airplane*

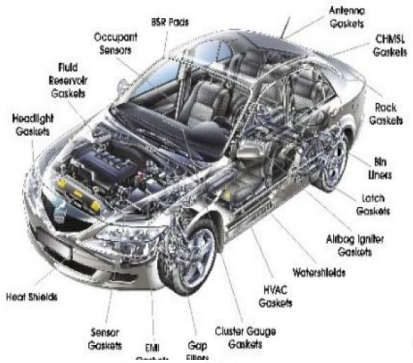


Considerations

- Traditional products with innovative services
  - Example: *e-Sport!*

Considerations

- Traditional products with innovative services
  - And so on…



Embedded Systems Design: A Unified Hardware/Software Introduction

Design challenge – optimizing 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
    - Area (or % of available resources, e.g. memory)
  - Performance: the execution time or throughput of the system
    - Latency, Throughput, *Speedup*, …
  - Power/Energy: the amount of power/energy 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/available to build a working version of the system
  - Time-to-market: the time needed/available 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
  - Robustness, Portability, Wearability, Reusabilty
  - Correctness, Safety, many more
  - …
  - …

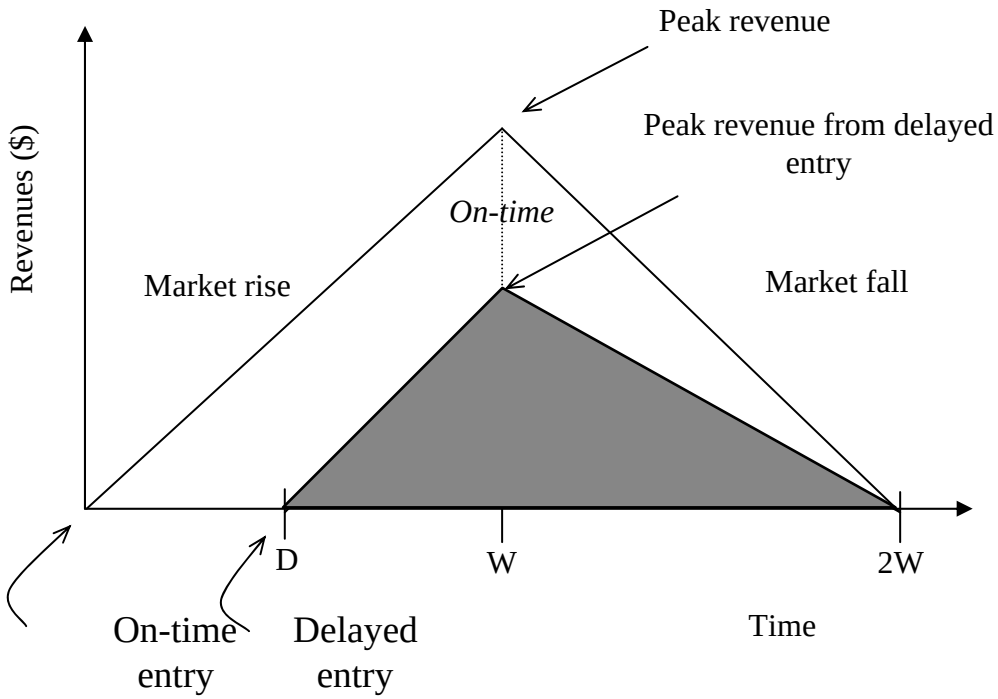
Considerations

- A designer could think that F/NF aspects driving the design choices are only related to technology issues but, in the real industrial life, the ability to exploit the market at the best moment is often the most important one!
  - It is needed to select technologies and design methodologies that lead to the max sales volume and max profit
    - Rarely this means to exploit the most advanced technologies…
    - Not always this means to find the most cheaper design!

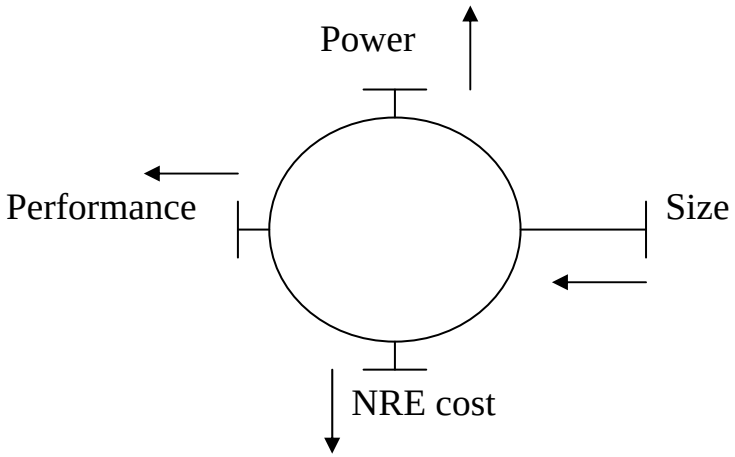
Considerations

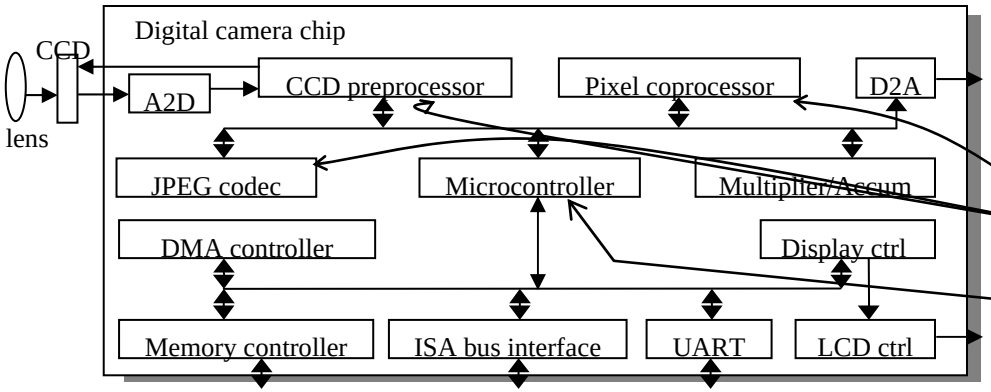
- Time-to-market
  - Sales volume loss

$$R_L = R_0 \frac{D(3W - D)}{2W^2}$$



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

Hardware

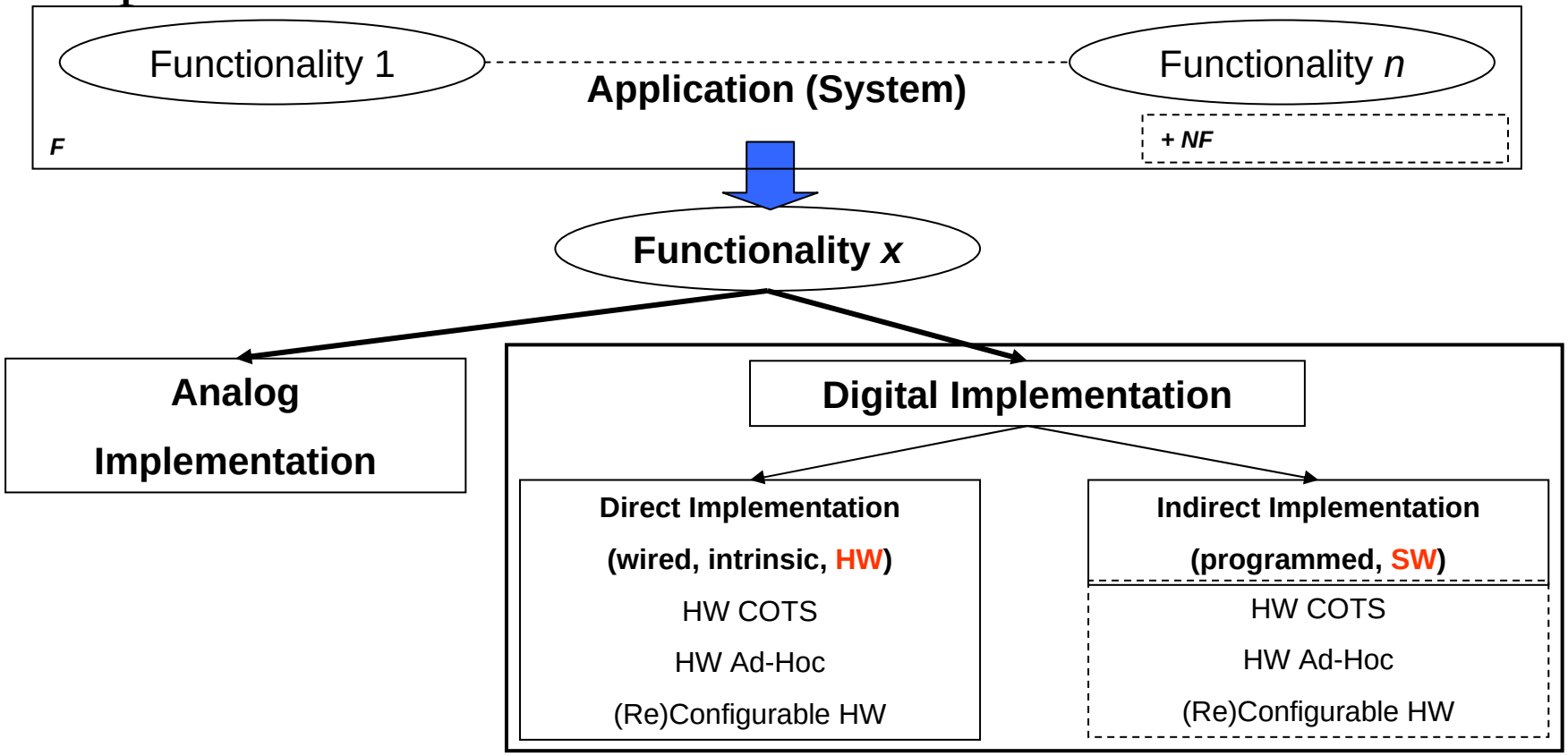
Software

Considerations

- Given the same F requirements, the best technologies could be very different depending on NF ones
  - The whole HW/SW architecture could be very different too!
- For this, it is quite difficult to standardize embedded system architectures (as normally done for general-purpose systems)
  - Some exceptions could be found in specific domains
    - e.g. the AUTOSAR reference architecture in the automotive domain
- This exacerbate the need for cross-competence designers!

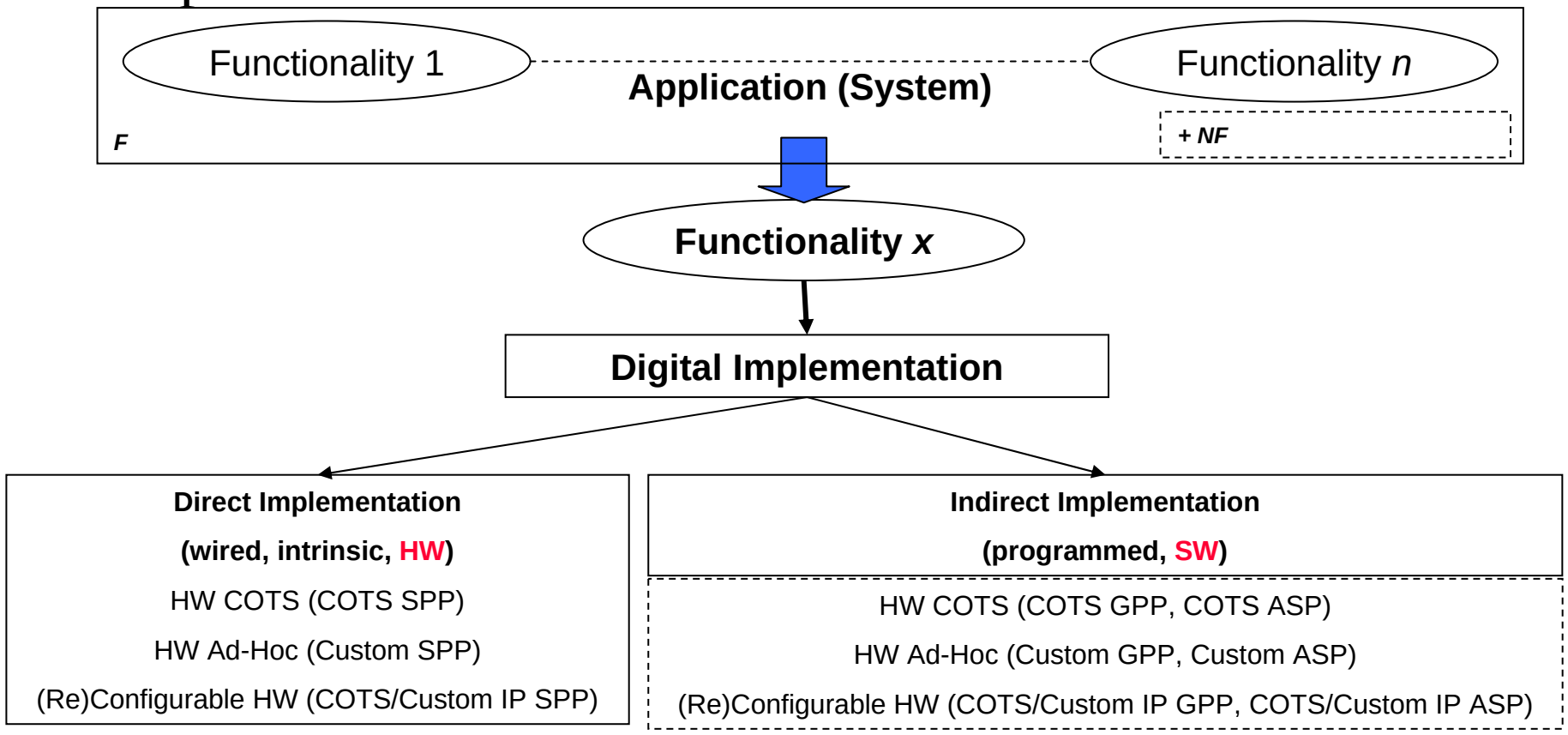
Considerations

- Implementation alternatives



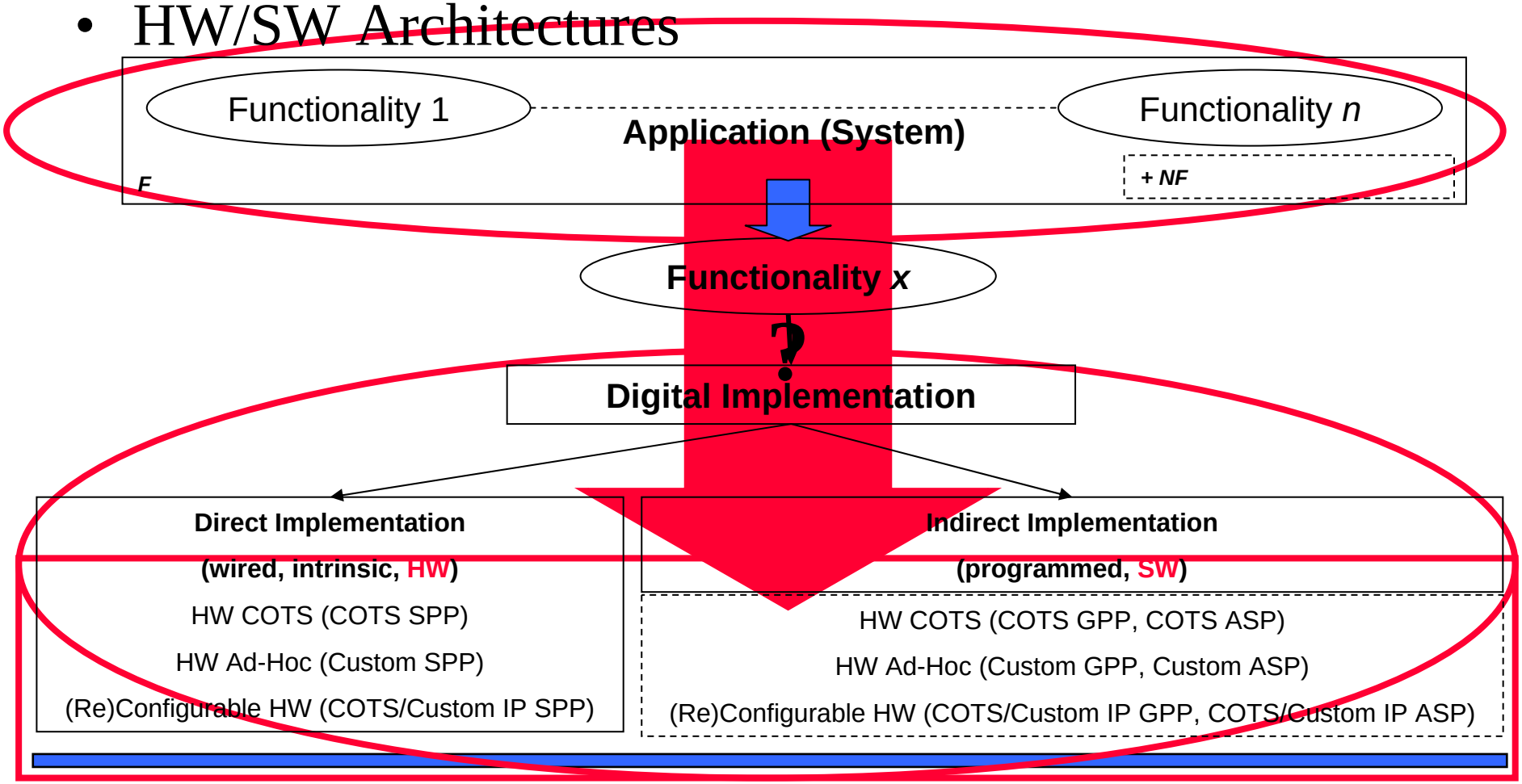
Considerations

- Implementation alternatives



Considerations

- HW/SW Architectures



Embedded Systems Design: A Unified Hardware/Software Introduction

Technologies

Three key embedded system technologies

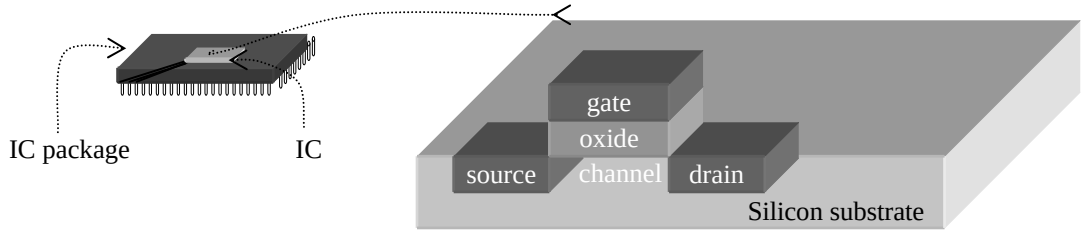
- Technology
  - A manner of accomplishing a task, especially using technical processes, methods, or knowledge
- Three key technologies for embedded systems
  - IC technology
  - Processor technology
  - Design technology

Three key embedded system technologies

- Technology
  - A manner of accomplishing a task, especially using technical processes, methods, or knowledge
- Three key technologies for embedded systems
  - **IC technology**
    - In which way can we realize processors?
  - Processor technology
  - Design technology

IC technology

- The manner in which a digital (gate-level) implementation is mapped onto an IC
  - IC: Integrated circuit, or “chip”
  - IC technologies differ in their customization to a design
  - IC’s consist of numerous layers (perhaps 10 or more)
    - IC technologies differ with respect to who builds each layer and when



IC technology

- Three types of IC technologies
  - Full-custom/VLSI
  - Semi-custom ASIC (gate array and standard cell)
  - PLD (Programmable Logic Device)

Full-custom/VLSI

- All layers are optimized for an embedded system's particular digital implementation
  - Placing transistors
  - Sizing transistors
  - Routing wires
- Benefits
  - Excellent performance, small size, low power
- Drawbacks
  - High NRE cost, long time-to-market

Semi-custom

- Lower layers are fully or partially built
  - Designers are left with routing of wires and maybe placing some blocks
- Benefits
  - Good performance, good size, less NRE cost than a full-custom implementation
- Drawbacks
  - Still require weeks to months to develop

PLD (Programmable Logic Device)

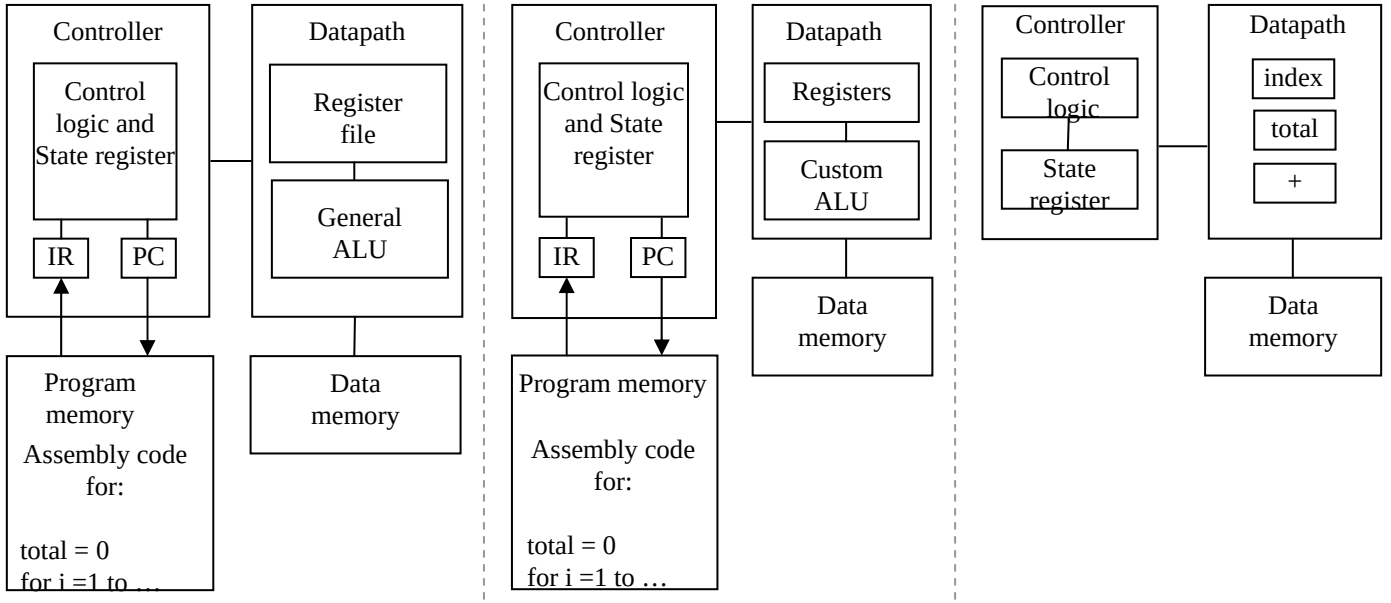
- All layers already exist
  - Designers can purchase an IC
  - Connections on the IC are either created or destroyed to implement desired functionality
  - Field-Programmable Gate Array (FPGA) very popular
- Benefits
  - Low NRE costs, almost instant IC availability
- Drawbacks
  - Bigger, expensive (average of 15 € per unit for a mid-range FPGA), power hungry, slower

Three key embedded system technologies

- Technology
  - A manner of accomplishing a task, especially using technical processes, methods, or knowledge
- Three key technologies for embedded systems
  - IC technology
    - In which way can we realize processors?
  - **Processor technology**
    - In which way can we perform computations?
  - Design technology

Processor technology

- The architecture of the computation engine used to implement a system's desired functionality
- Processor does not have to be programmable
  - "Processor" *not* equal to general-purpose processor



Processor technology

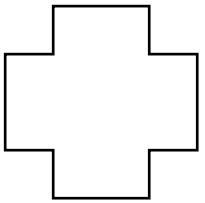
- Processors vary in their customization for the problem at hand

Desired functionality

```
total = 0
for i = 1 to N  loop
   total += M[i]
end loop
```

General-purpose processor

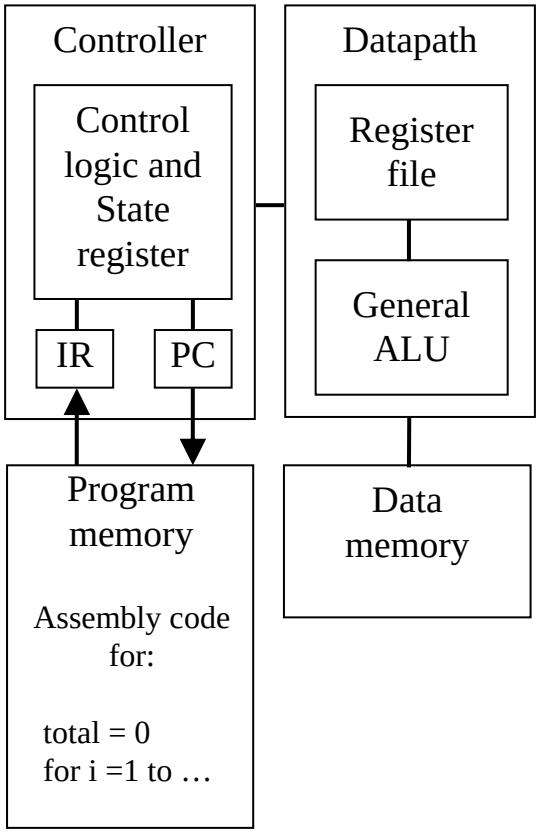
Application-specific processor



Single-purpose processor

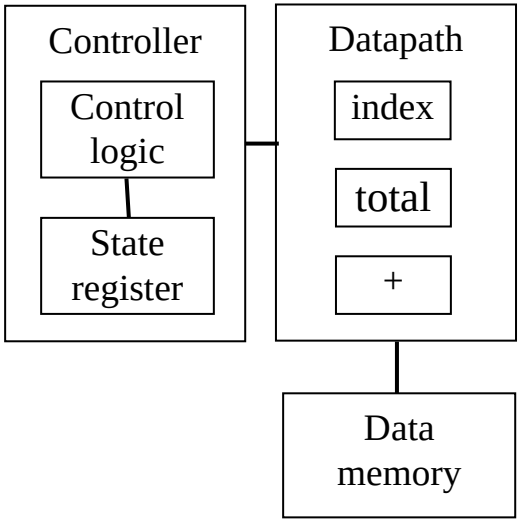
General-purpose processors

- Programmable device used in a variety of applications
  - Also known as “microprocessor”
- Features
  - Program memory
  - General datapath with large register file and general ALU
- User benefits
  - Low time-to-market and NRE costs
  - High flexibility
- “Pentium” the most well-known, but there are hundreds of others



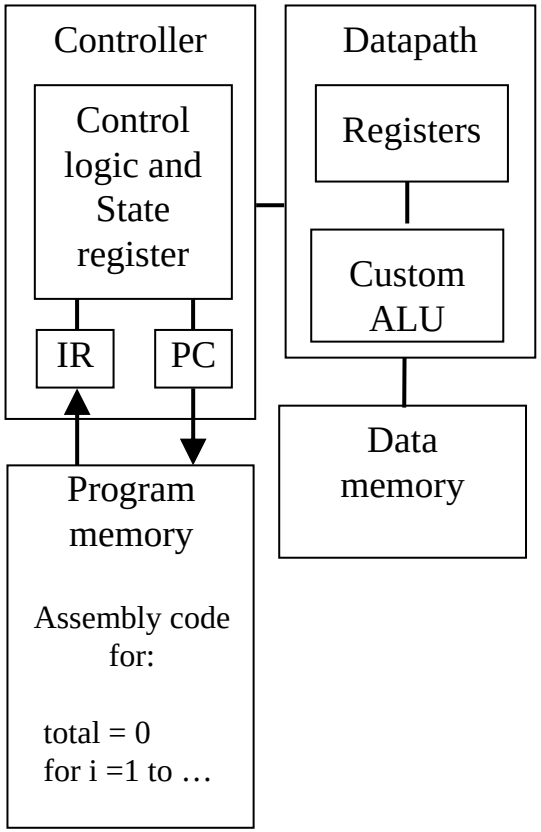
Single-purpose processors

- Digital circuit designed to execute exactly one "program"
  - a.k.a. coprocessor, accelerator or peripheral
- Features
  - Contains only the components needed to execute a single program
  - No program memory
- Benefits
  - Fast
  - Low power
  - Small size



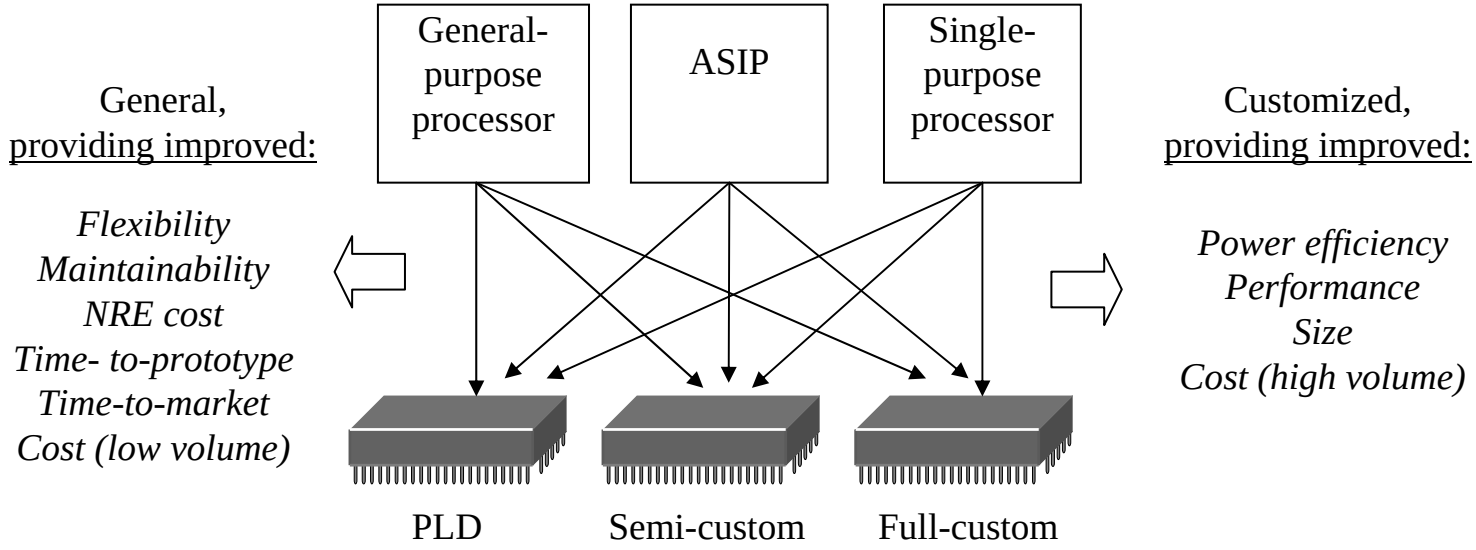
Application-specific processors

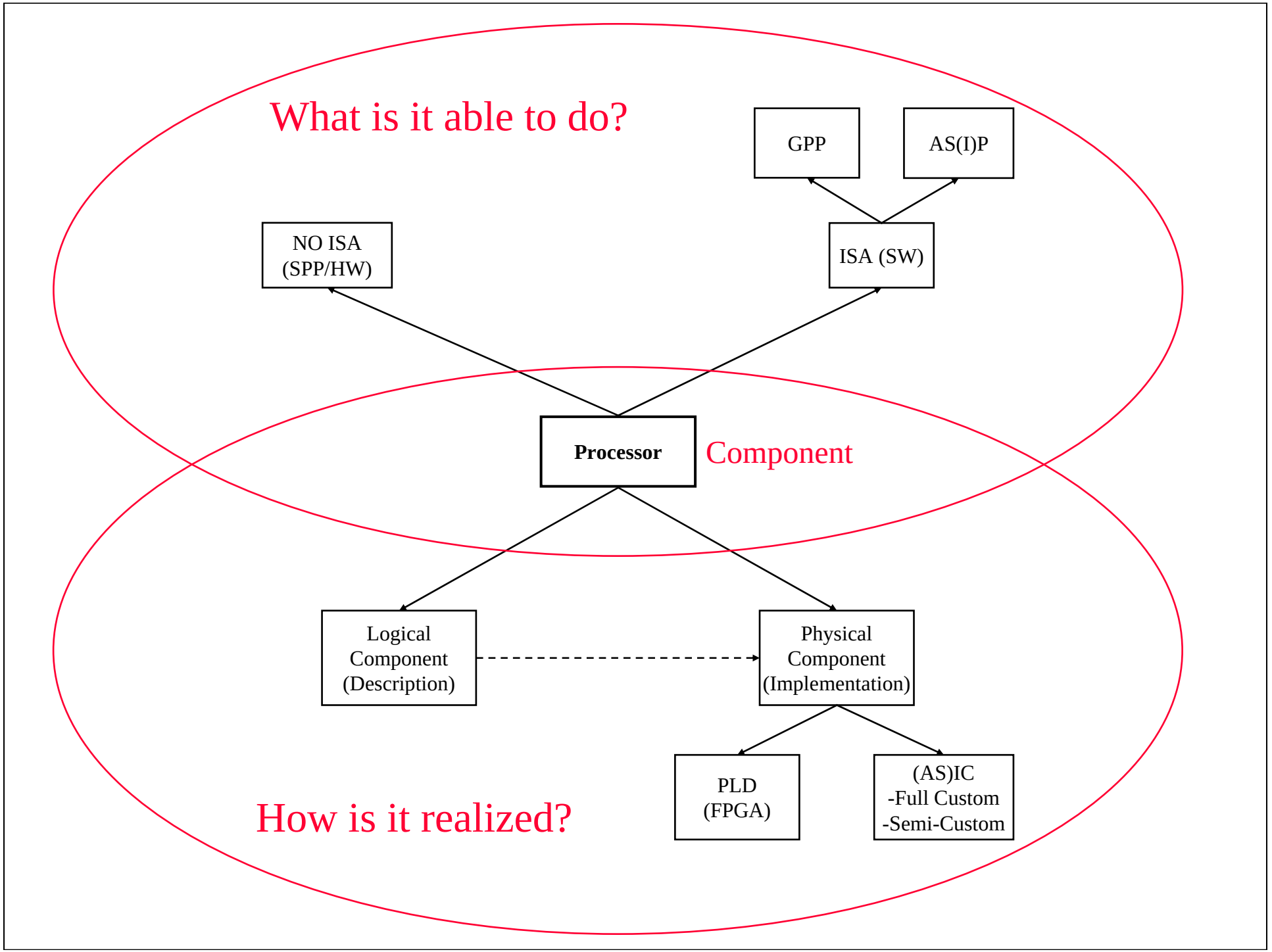
- Programmable processor optimized for a particular class of applications having common characteristics
  - Compromise between general-purpose and single-purpose processors
- Features
  - Program memory
  - Optimized datapath
  - Special functional units
- Benefits
  - Some flexibility, good performance, size and power



Independence of processor and IC technologies

- Basic tradeoff
  - General vs. custom
  - With respect to processor technology or IC technology
  - The two technologies are independent




What is it able to do?
GPP
AS(I)P
NO ISA
(SPP/HW)
ISA (SW)
Processor
Component
Logical
Component
(Description)
Physical
Component
(Implementation)
PLD
(FPGA)
(AS)IC
-Full Custom
-Semi-Custom
How is it realized?

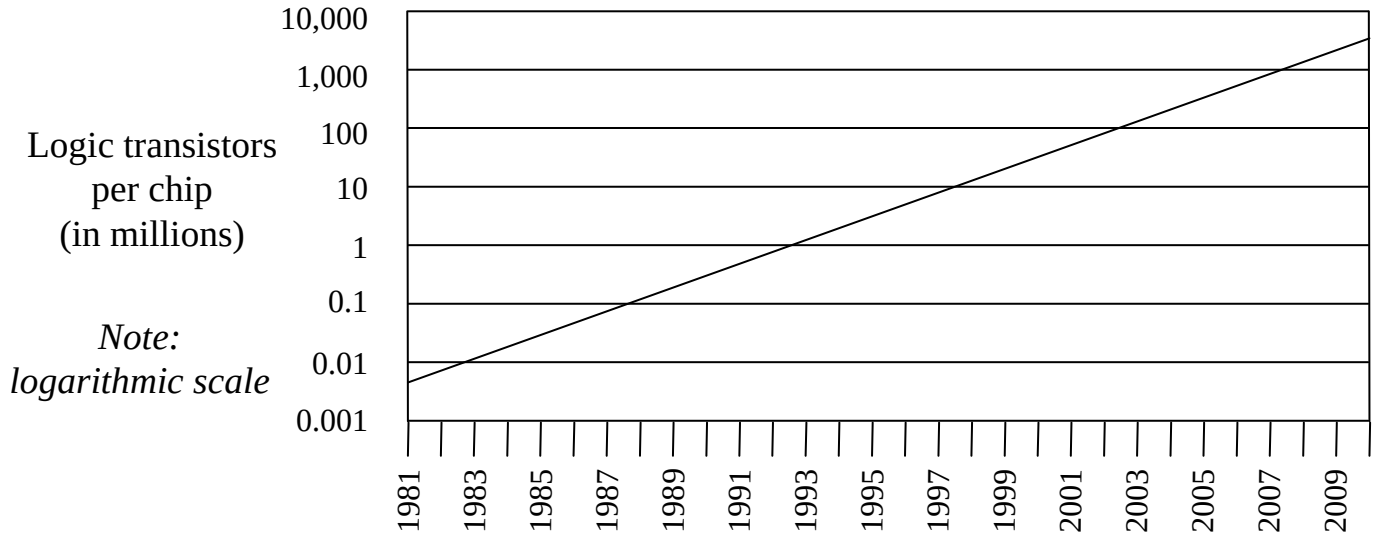
Three key embedded system technologies

- Technology
  - A manner of accomplishing a task, especially using technical processes, methods, or knowledge
- Three key technologies for embedded systems
  - IC technology
    - In which way can we realize processors?
  - Processor technology
    - In which way can we perform computations?
  - **Design technology**
    - In which way can we design and exploit processors and their combinations?

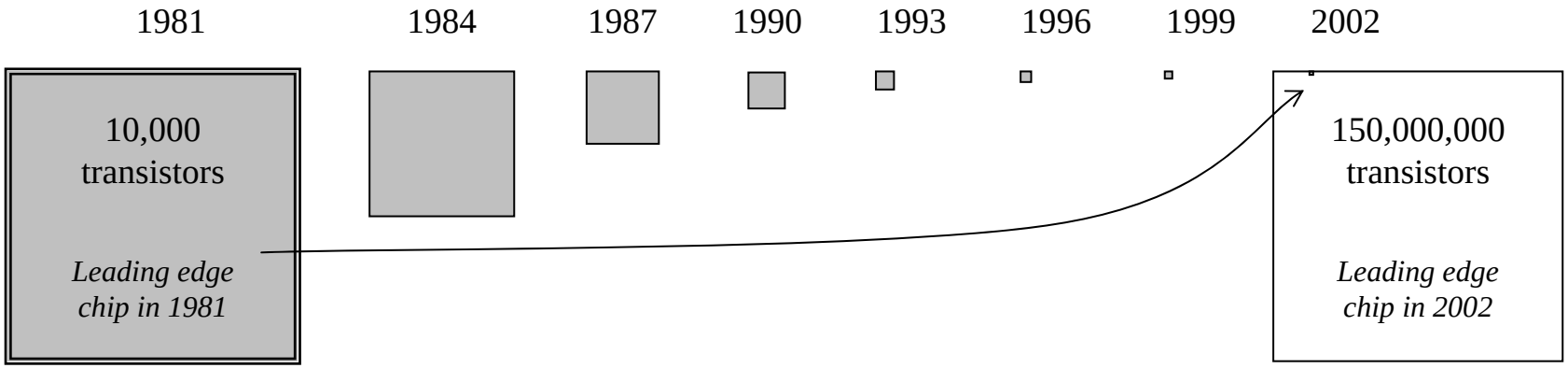
Moore's law

- The most important trend in embedded systems
  - Predicted in 1965 by Intel co-founder Gordon Moore

IC transistor capacity has doubled roughly every 18 months for the past several decades

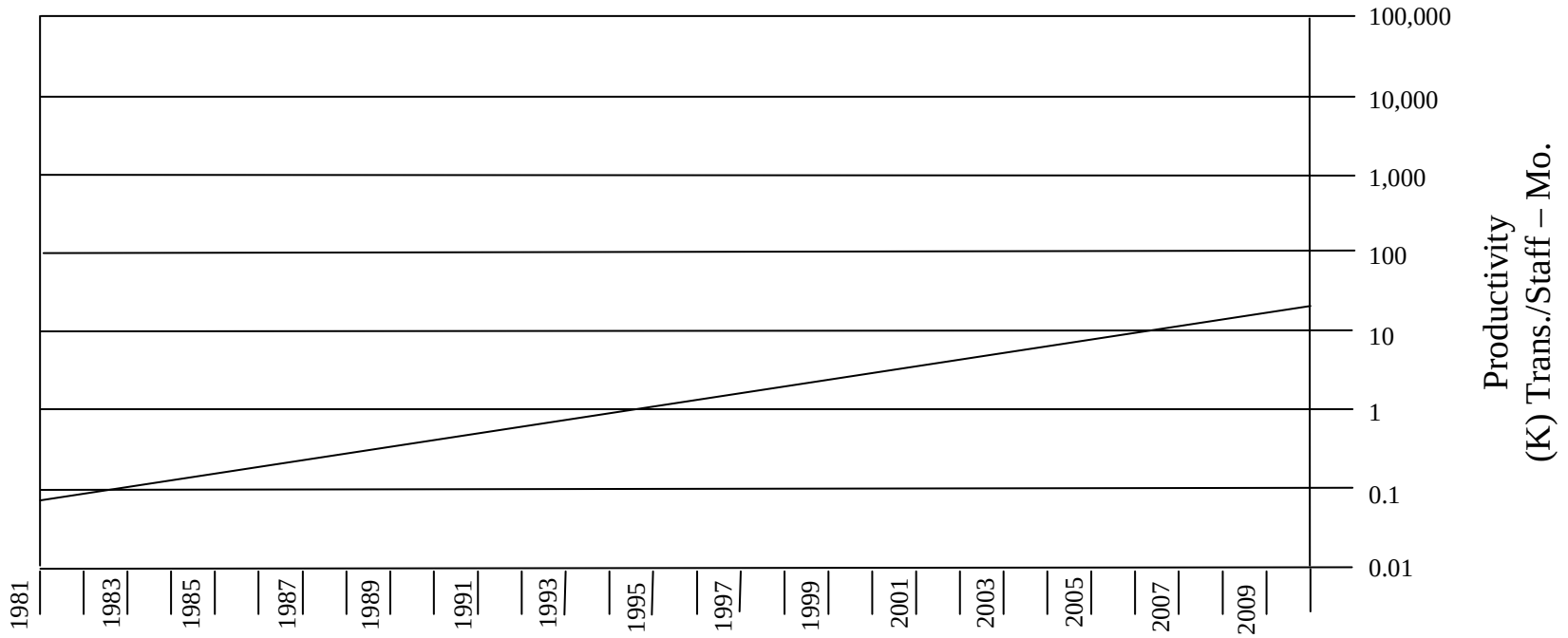


Graphical illustration of Moore's law



- Something that doubles frequently grows more quickly than most people realize!
  - A 2002 chip can hold about 15,000 1981 chips inside itself

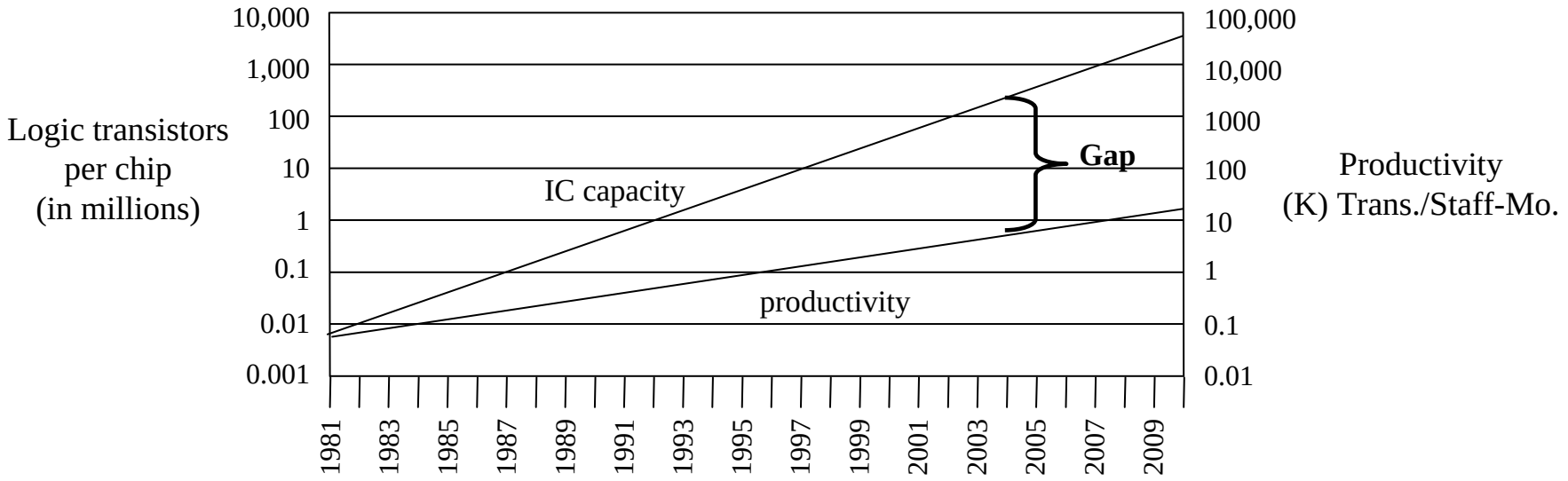
Design productivity exponential increase



- Exponential increase over the past few decades

Design productivity gap

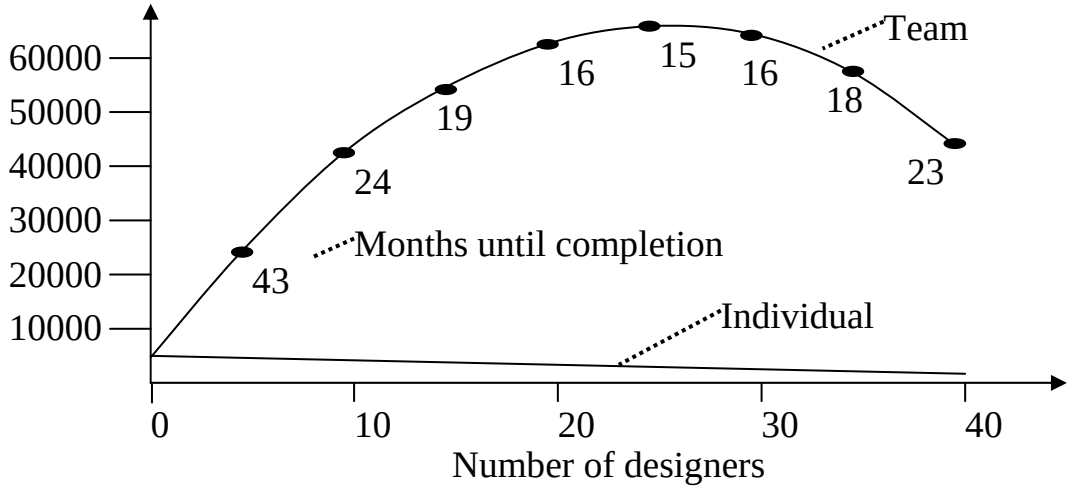
- While designer productivity has grown at an impressive rate over the past decades, the rate of improvement has not kept pace with chip capacity



The mythical man-month

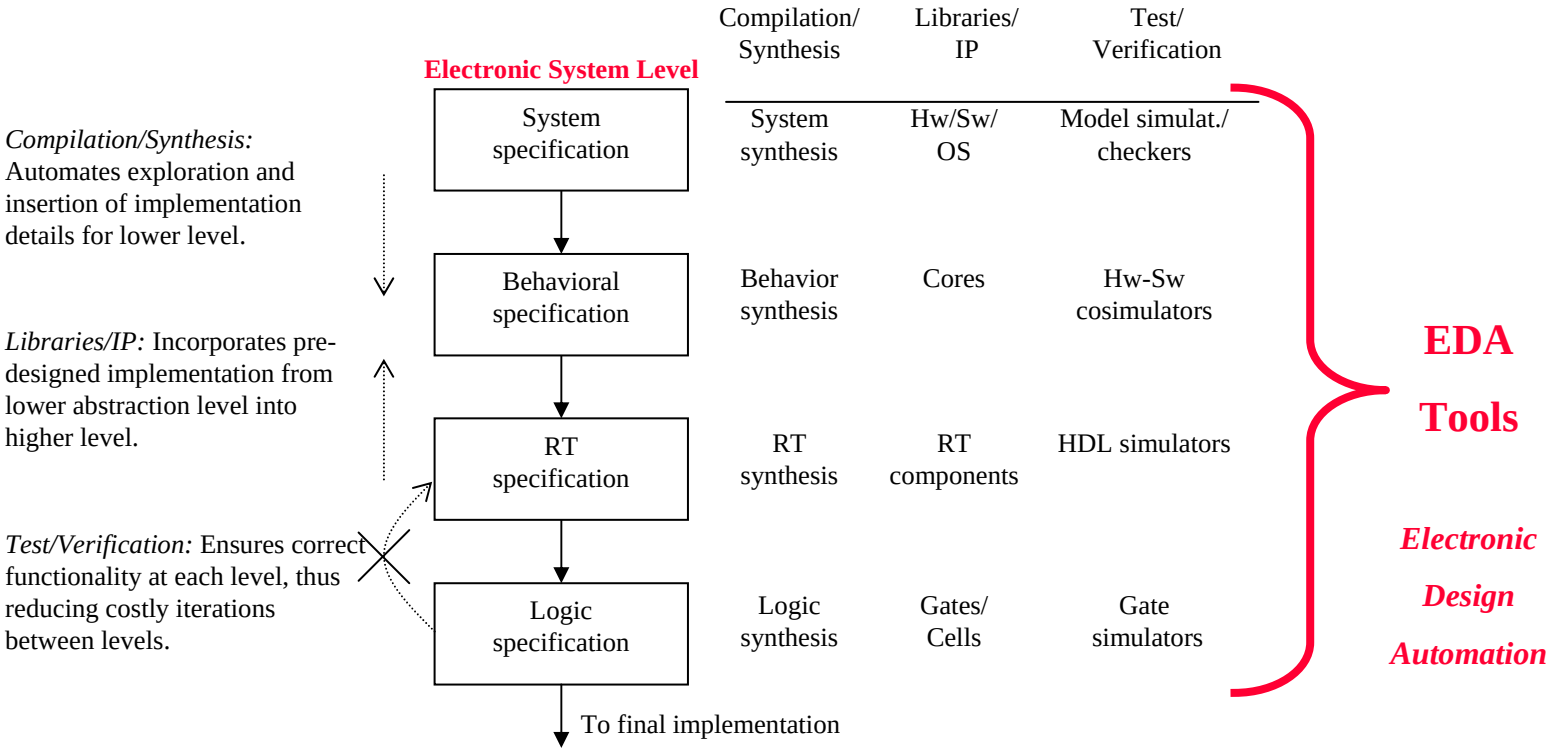
- The situation is even worse than the productivity gap indicates
- In theory, adding designers to team reduces project completion time
- In reality, productivity per designer decreases due to complexities of team management and communication
- In the software community, known as “the mythical man-month” (Brooks 1975)
- At some point, can actually lengthen project completion time! (“Too many cooks”)

- 1M transistors, 1 designer=5000 trans/month
- Each additional designer reduces for 100 trans/month
- So 2 designers produce 4900 trans/month each



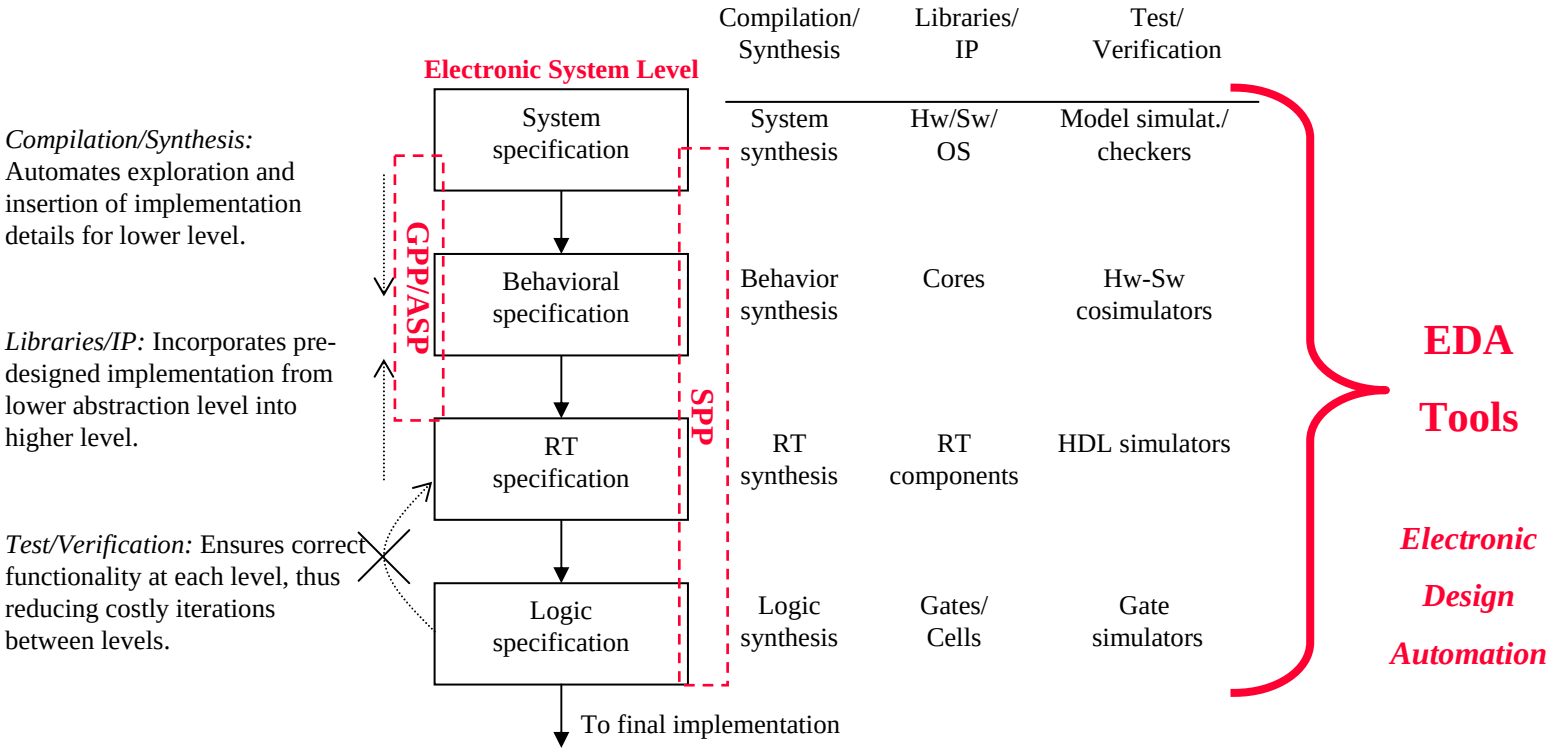
Design Technology

- The manner in which we convert our concept of desired system functionality into an implementation



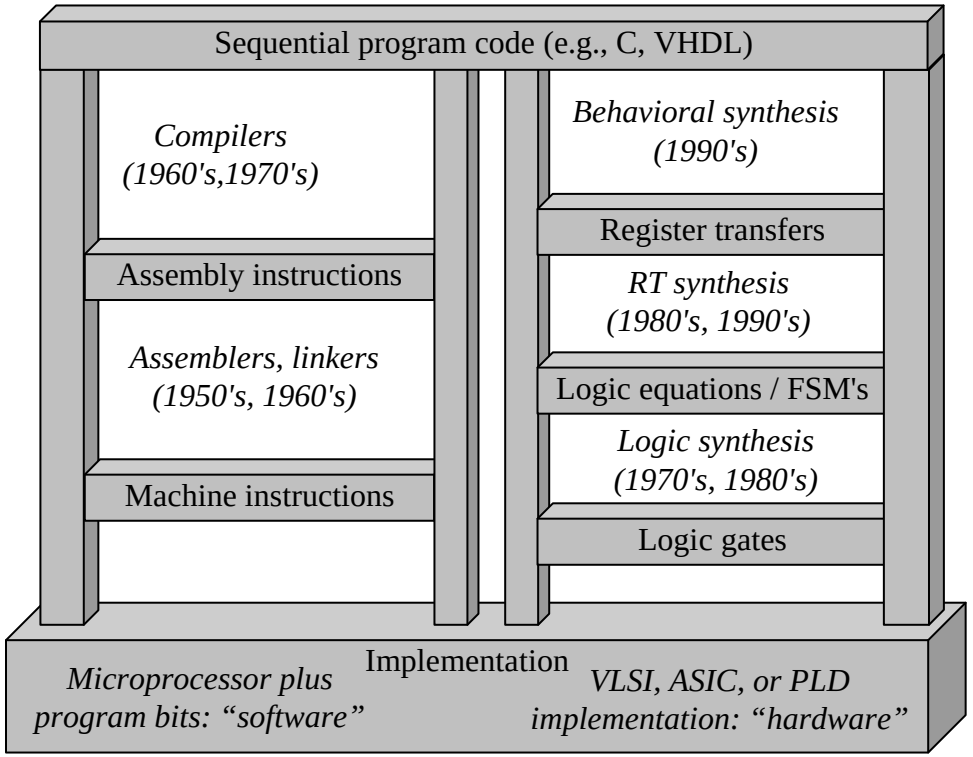
Design Technology

- The manner in which we convert our concept of desired system functionality into an implementation



The co-design ladder

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



The choice of hardware versus software for a particular function is simply a tradeoff among various design metrics, like performance, power, size, NRE cost, and especially flexibility; there is no fundamental difference between what hardware or software can implement.

Considerations

- Embedded systems development is still far from being a discipline with mature and general design methodologies
  - SW Engineering vs ES Engineering

- It requires cross-competences and good knowledge of the applicative domain

Considerations

- An helpful improvement is currently provided by the so called *Platform Based Design* approach
  - The goal is to define and/or exploit domain-oriented HW/SW architectures (i.e. the platforms) that could be easily re-used for all the applications of a given domain
    - e.g. Automotive: *AUTOSAR*, *Aeronautics: ARINC*,…
      - A platform can be
        - Virtual: it is a set of models of the different parts of the system (may be executable, analyzable, synthesizable,…)
        - Real: it is a set of well known existing hw/sw components
    - Benefits: re-use of basic blocks already validated and commercially available to move the design effort at higher levels of abstraction
    - Drawbacks: less customizable/optimizable

Summary

- Embedded systems are everywhere
- Key challenge: optimization of design metrics
  - Design metrics compete with one another
- A unified view of hardware and software is necessary to improve productivity
- Three key technologies
  - IC: Full-custom, semi-custom, PLD
  - Processor: general-purpose, application-specific, single-purpose
  - Design: Compilation/synthesis, libraries/IP, test/verification