Introduction to Embedded Systems

Objectives

- Introduction to embedded systems
- Embedded system components
  - Hardware
  - Software
- Embedded system programming

Contents

- Introduction to embedded systems
- Software engineering
- Computer architecture
- Operating systems
- Digital systems
- Programming practice
- Theory for practical works

Wireless Communications

Telematics System for Automobiles

Hand-held GPS Units

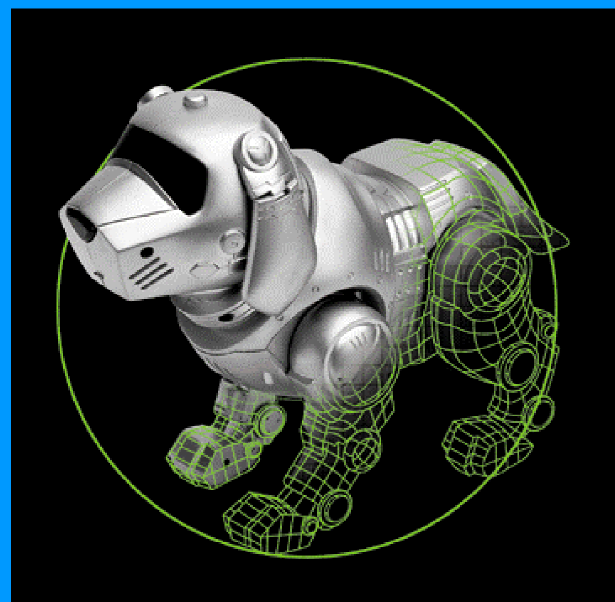
Robotics Control

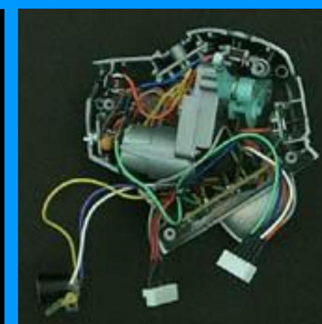
Spider robot – constructed with LEGO Mindstorms Components

22

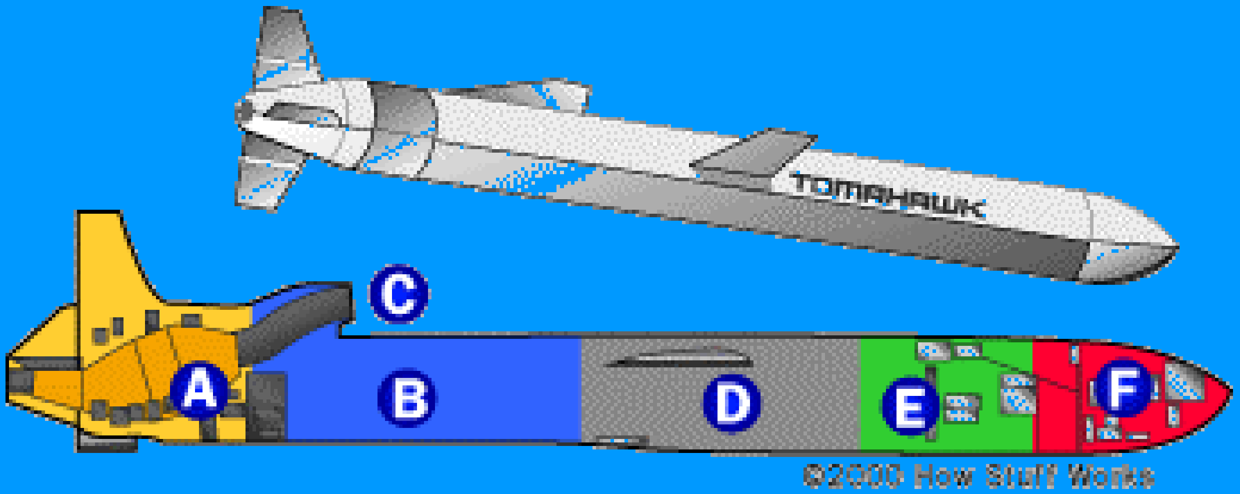
More examples

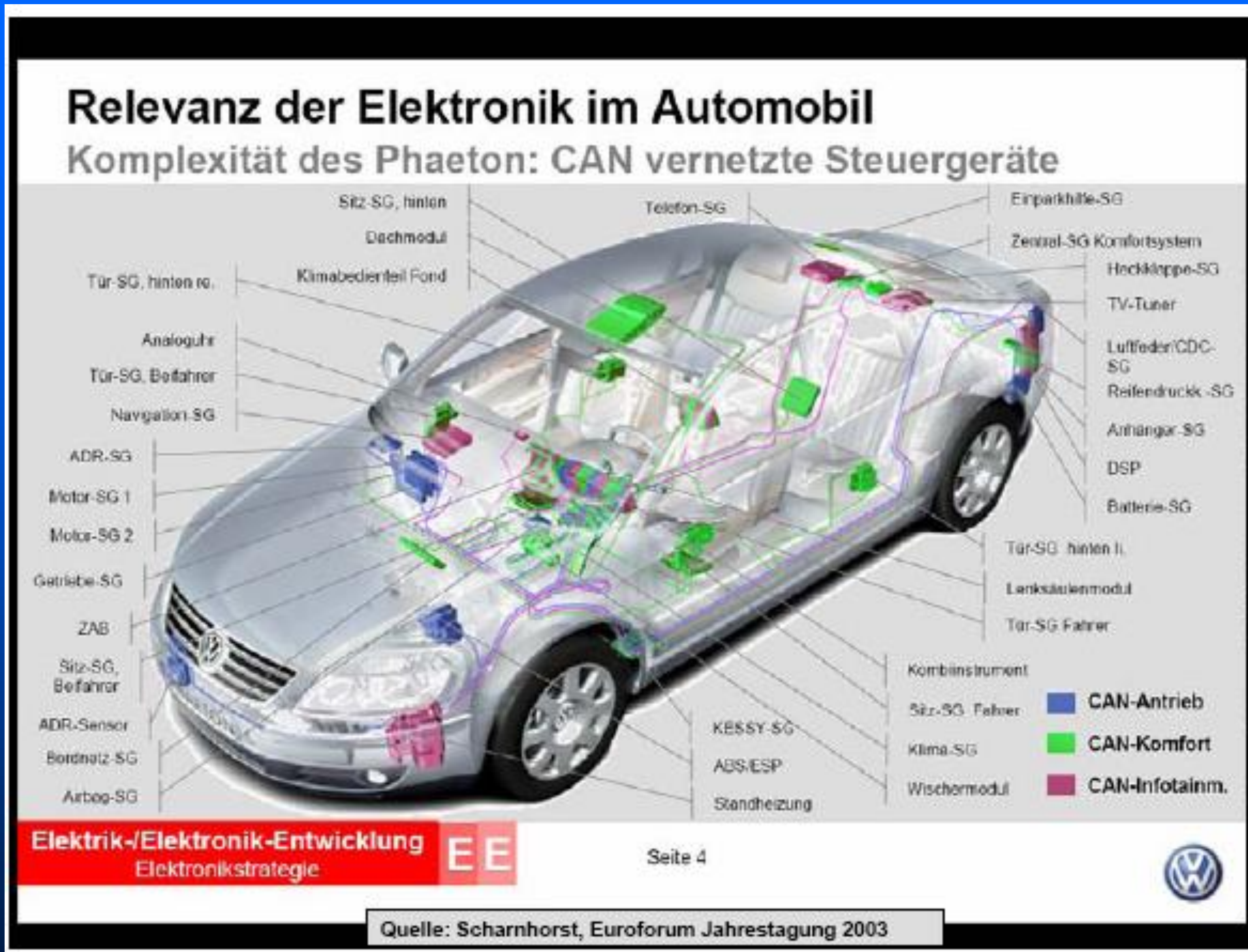
Smart Toys





Cruise Missile Guidance

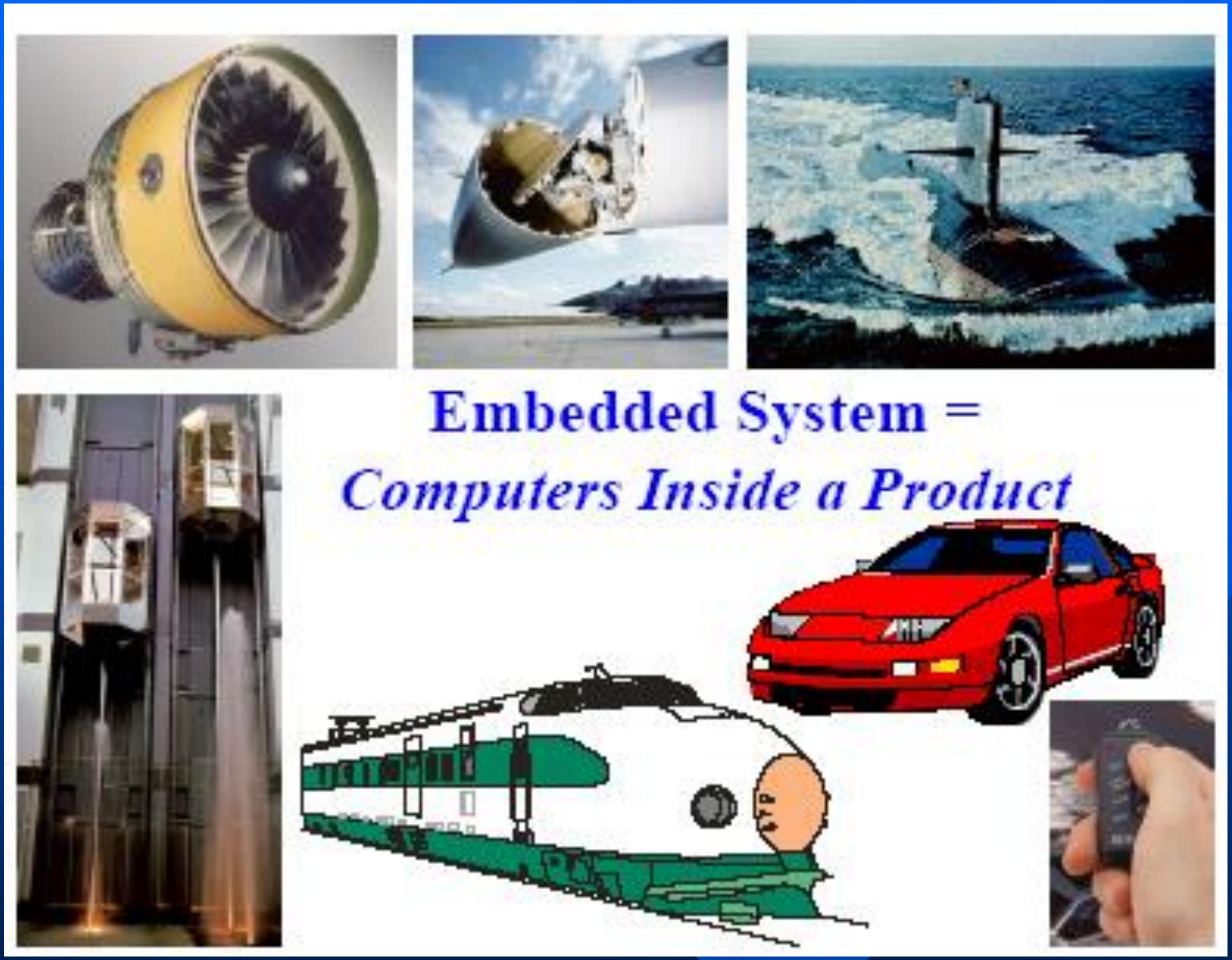



Relevanz der Elektronik im Automobil
Komplexität des Phaeton: CAN vernetzte Steuergeräte
Sitz-SG, hinten
Dachmodul
Klimabedienteil Fond
Tür-SG, hinten re.
Analoguhr
Tür-SG, Beifahrer
Navigation-SG
ADR-SG
Motor-SG 1
Motor-SG 2
Getriebe-SG
ZAB
Sitz-SG, Beifahrer
ADR-Sensor
Bordnetz-SG
Airbag-SG
Telefon-SG
Einparkhilfe-SG
Zentral-SG Komfortsystem
Heckklappe-SG
TV-Tuner
Luftfeder/CDC-SG
Reifendruck -SG
Anhänger-SG
DSP
Batterie-SG
Tür-SG hinten li.
Lenksäulenmodul
Tür-SG Fahrer
Kombiinstrument
Sitz-SG Fahrer
Klima-SG
Wischermodul
KESSY-SG
ABS/ESP
Standheizung
CAN-Antrieb
CAN-Komfort
CAN-Infotainm.
Elektrik-/Elektronik-Entwicklung
Elektronikstrategie
EE
Seite 4
Quelle: Scharnhorst, Euroforum Jahrestagung 2003

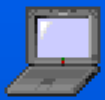
Definition

- "Any sort of device which includes a programmable computer but itself is not intended to be a general-purpose computer"
    - Wayne Wolf

Definition



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

Slide credit Vahid/Givargis, Embedded Systems Design: A Unified Hardware/Software Introduction, 2000

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

How many do we use?

- Average middle-class American home has 40 to 50 embedded processors in it
  - Microwave, washer, dryer, dishwasher, TV, VCR, stereo, hair dryer, coffee maker, remote control, humidifier, heater, toys, etc.
- Luxury cars have over 60 embedded processors
  - Brakes, steering, windows, locks, ignition, dashboard displays, transmission, mirrors, etc.
- Personal computers have over 10 embedded processors
  - Graphics accelerator, mouse, keyboard, hard-drive, CD-ROM, bus interface, network card, etc.

- Mike Schulte

Embedded Systems = ระบบ<u>ฝังตัว</u>

- คิดถึงคอมพิวเตอร์Ťพีซีťเรานึกถึง
  - Processor, Clock speed, ROM/RAM
- พูดถึงเครื่องใช้ไฟฟ้าทั่วไปťเรานึกถึง

  ทีวี- ขนาดจอ, เทคโนโลยีการฉายภาพ

  เครื่องซักผ้า - ขนาดความจุ, ระบบซัก

  แอร์ - BTUťลักษณะการติดตั้ง

  โทรศัพท์มือถือ - จอťกล้อง

  รถยนต์ - ขนาดเครื่องยนต์ťสมรรถนะ

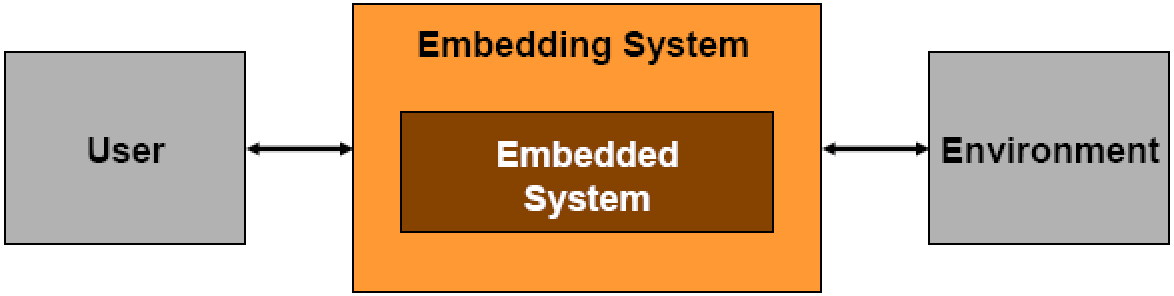
Types of Embedded Systems

Four General Embedded System Types

- **General Computing**
  - Applications similar to desktop computing, but in an embedded package
  - Video games, set-top boxes, wearable computers, automatic tellers
- **Control Systems**
  - Closed-loop feedback control of real-time system
  - Vehicle engines, chemical processes, nuclear power, flight control
- **Signal Processing**
  - Computations involving large data streams
  - Radar, Sonar, video compression
- **Communication & Networking**
  - Switching and information transmission
  - Telephone system, Internet

Types of Embedded Systems

Two different main application areas



Product automation

Embedding system = product

Examples:

- Automotive Electronics
- Avionics
- Health Care Systems

Production automation

Embedding system = production system

Examples:

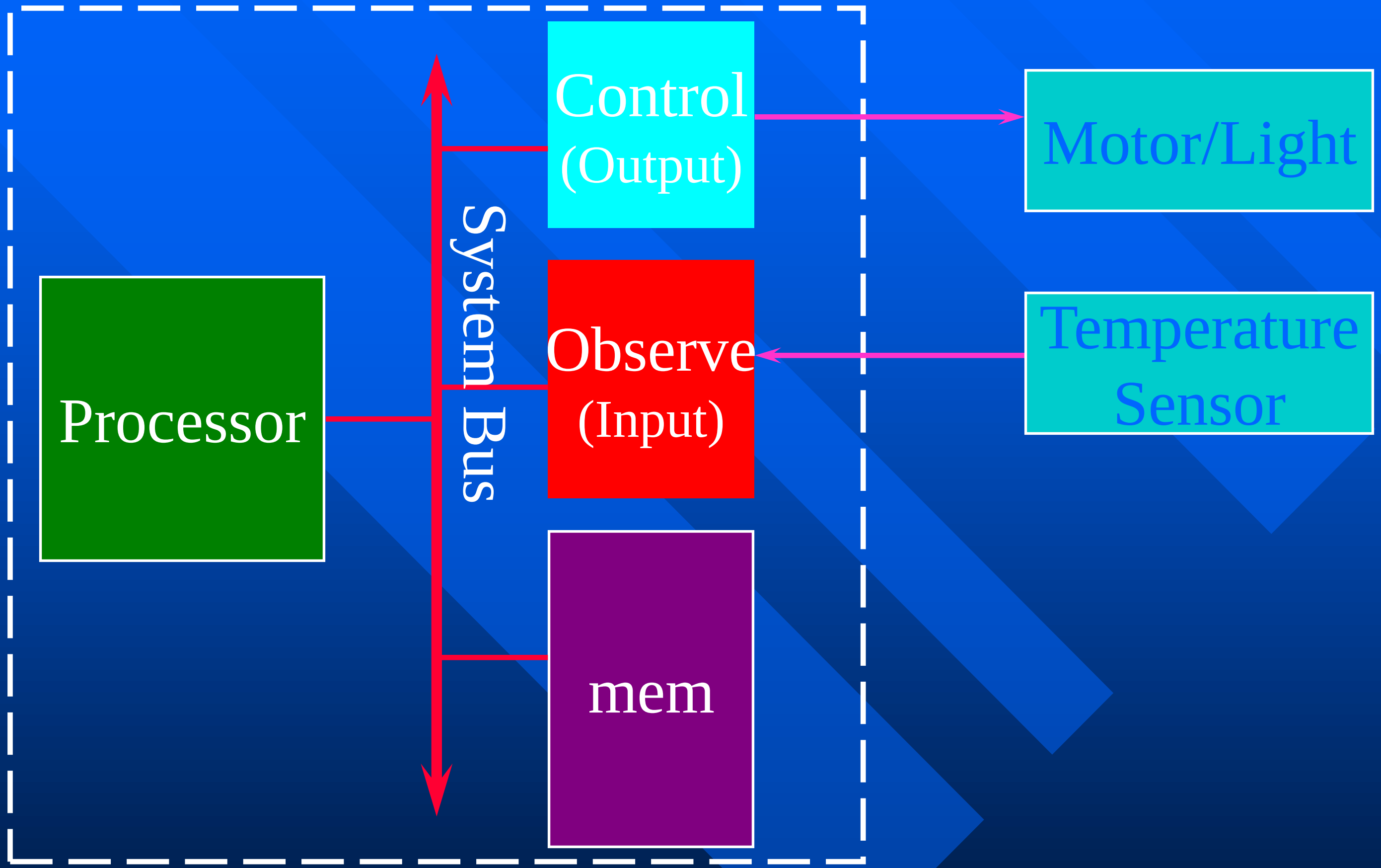
- Manufacturing Control
- Chemical Process Control
- Logistics

Typical Embedded Systems

- Are designed to observed (through sensors) and control something (through actuators)

  E.g. air condition senses room temperature and maintains it at set temperature via thermostat.

Embedded System Block Diagram

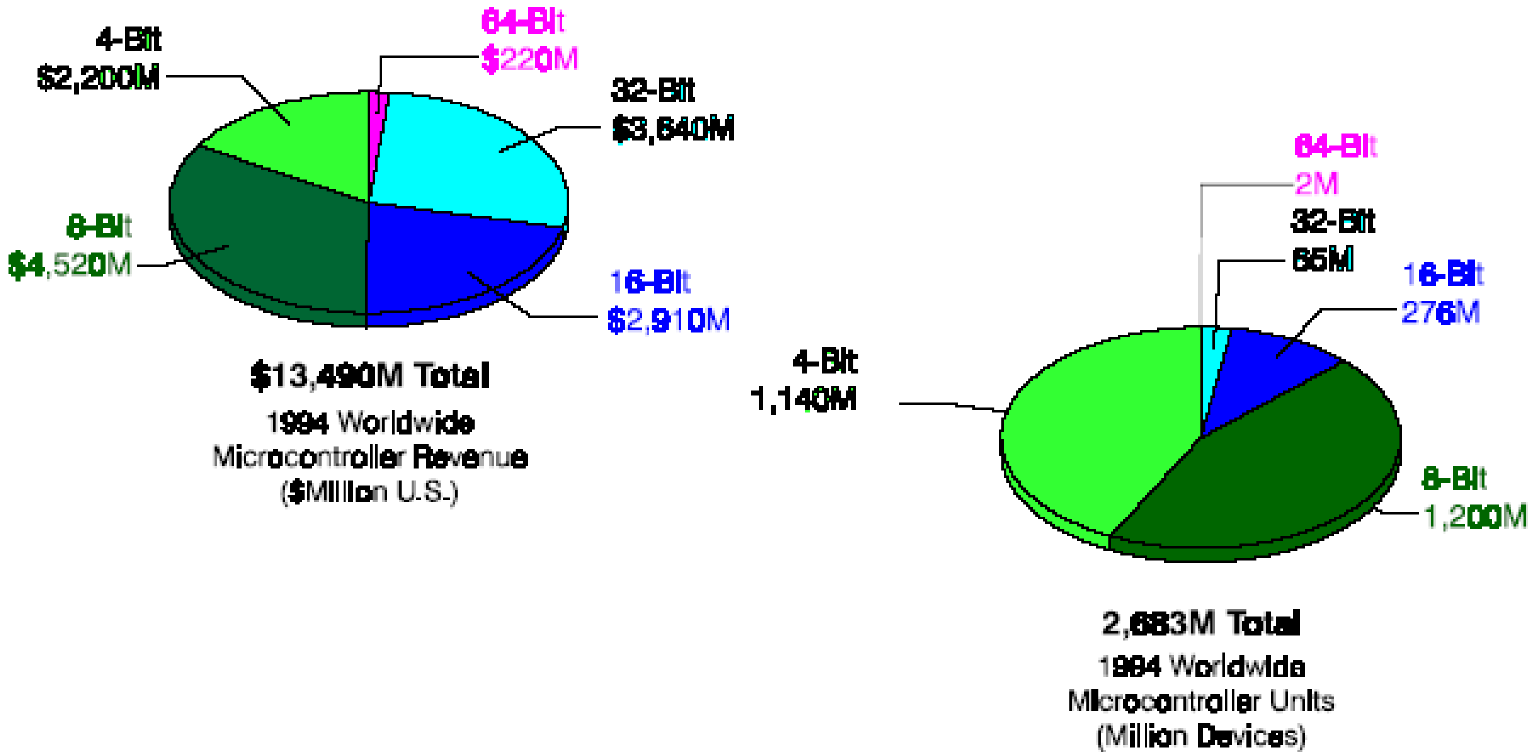


Processors

- Microprocessors for PCs
- Embedded processors or Microcontrollers for embedded systems
  - Often with lower clock speeds
  - Integrated with memory and
  - I/O devices e.g. A/D D/A PWM CAN
  - Higher environmental specs

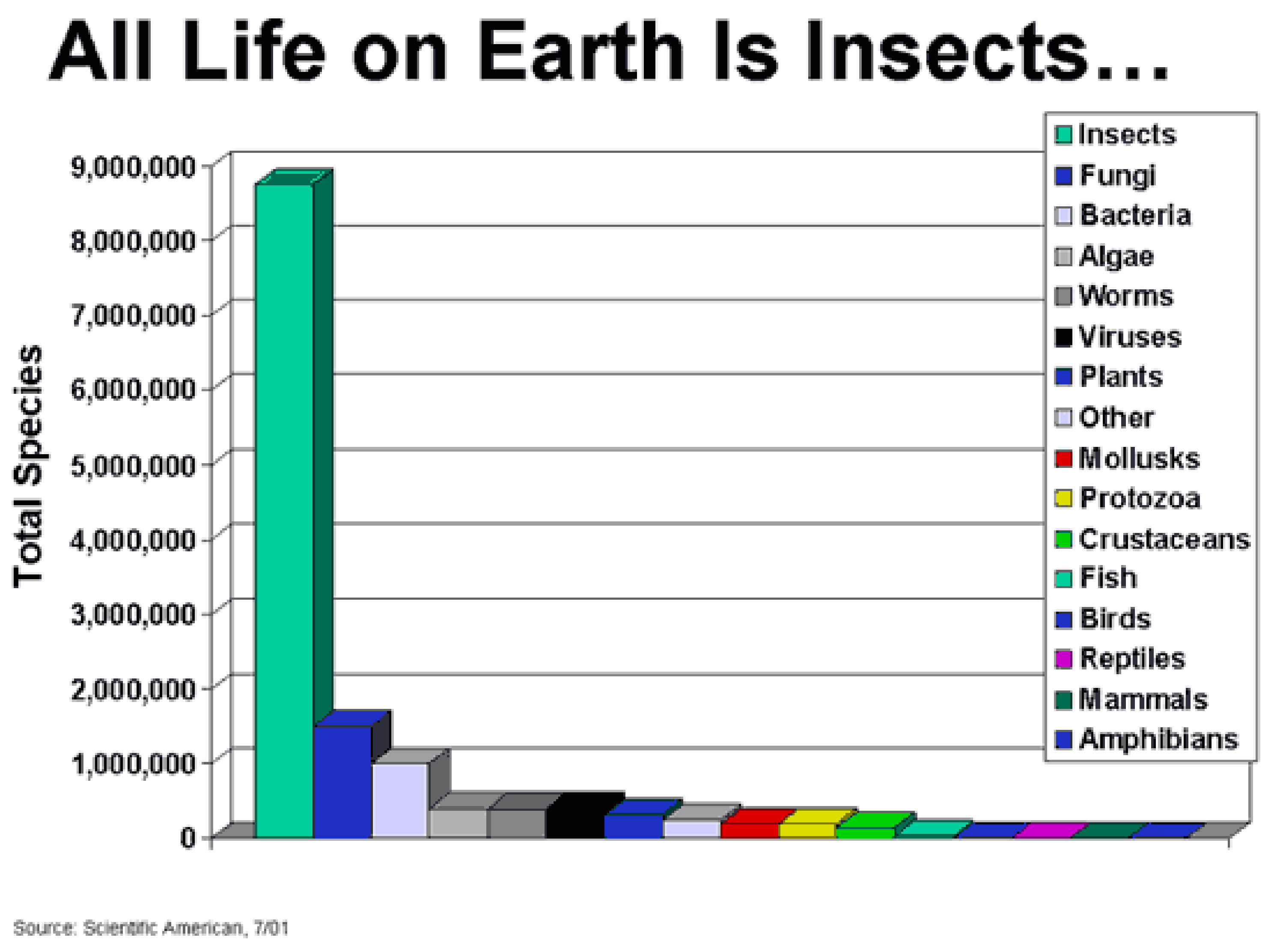
Microcontrollers dominates processor market

- ~80 Million PCs vs. ~3 Billion Embedded CPUs Annually
  - Embedded market growing; PC market mostly saturated



Approximated from EE Times, March 20, 1995
Source: The Information Architects

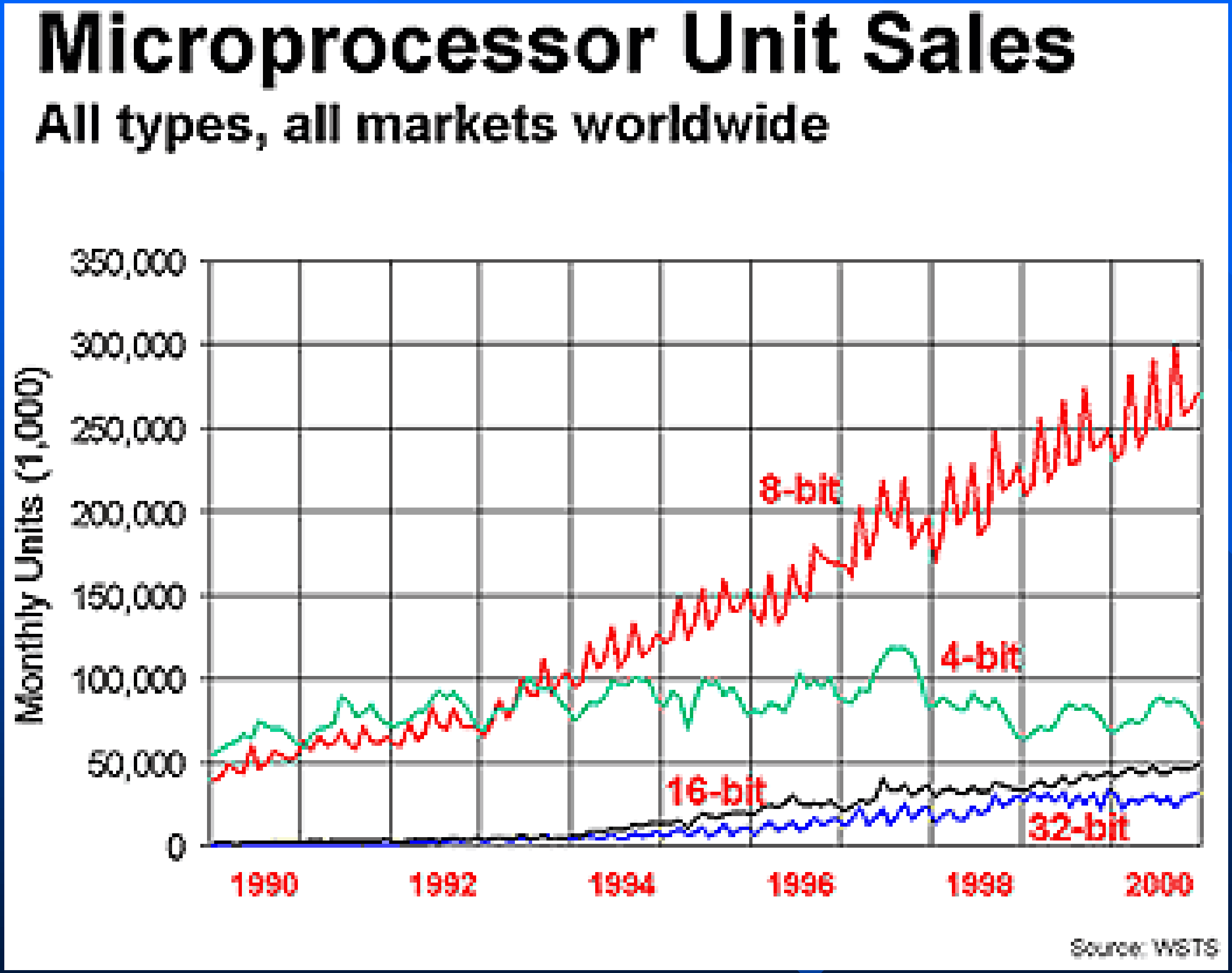
There are so many microcontrollers in the world



Types of Embedded Processors

- Computational micros (32- or 64-bit datapaths)
  - CPU of workstations, PCs, or high-end portable devices (PDAs)
  - x86, PA-RISC, PowerPC, SPARC, etc.
- Embedded general purpose micros (32-bit datapaths)
  - Designed for a wide range of embedded applications
  - Often scaled-down version of computational micros
  - ARM, PowerPC, MIPS, x86, 68K, etc.
- Microcontrollers (4-, 8-, or 16-bit datapaths)
  - Integrate processing unit, memory, I/O buses, and peripherals
  - Often low-cost, high-volume devices
- Domain-specific processors (datapath size varies greatly)
  - Designed for a particular application domain
  - Digital signal processors, multimedia processors, graphics processors, network processors, security processors, etc.

Processor Sales Data

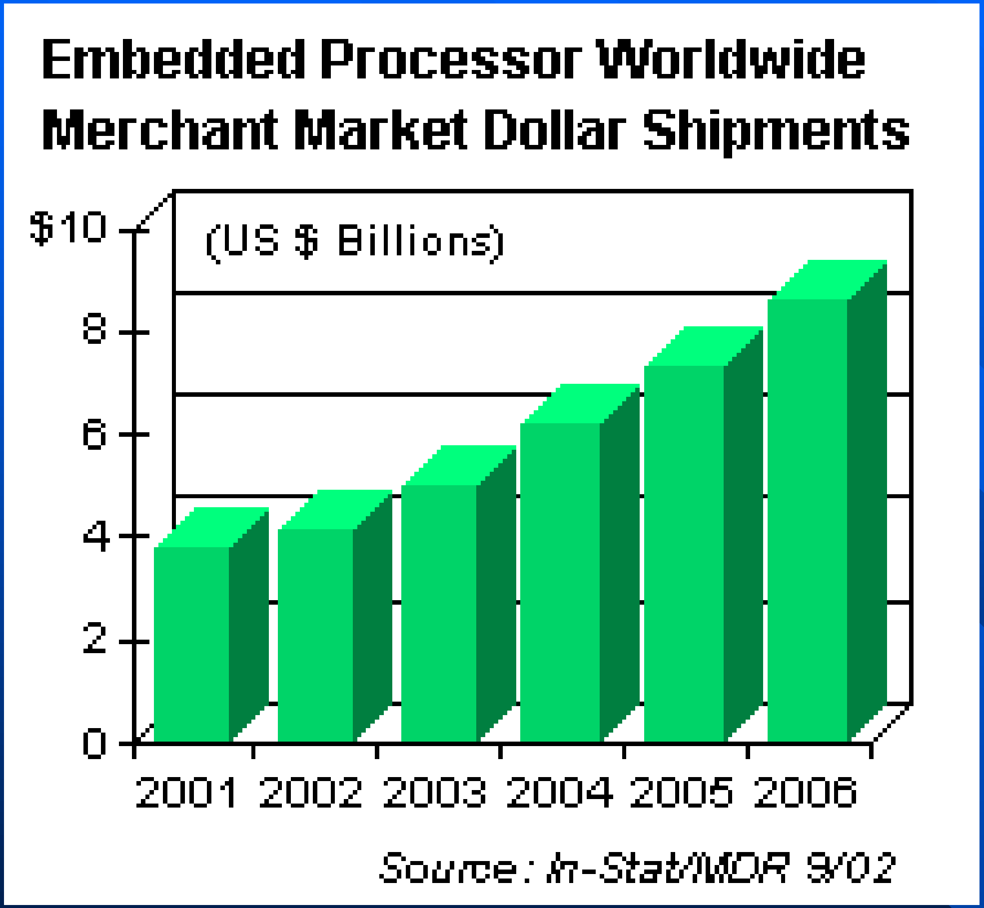


Processor Market

- 2001 processor market by volume:
  - Computational micros: 2%
  - Embedded general-purpose micros: 11%
  - DSPs: 10%
  - Microcontrollers: 80%
- 2001 processor market by revenue:
  - Computational micros: 51%
  - Embedded general-purpose micros: 8%
  - DSPs: 13%
  - Microcontrollers: 28%
- Higher growth expected for embedded micros, DSPs, and microcontrollers

Slide credit - Mike Schulte

Growing Demand



- Embedded processors account for
  - Over 97% of total processors sold
  - Over 60% of total sales from processors
- Sales expected to increase by roughly 15% each year

Slide credit - Mike Schulte

Moore's Law

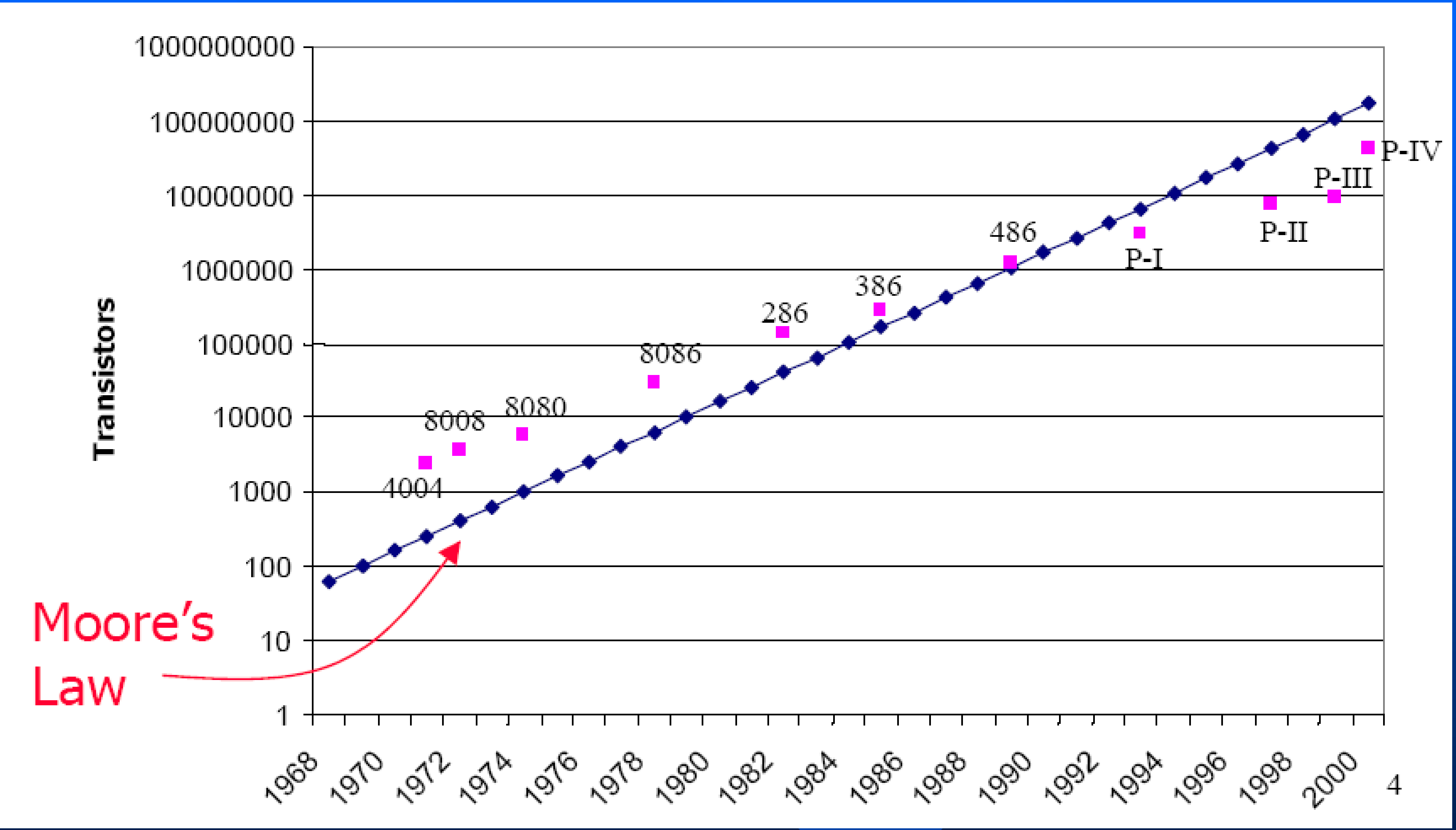
"Transistor capacity doubles every 18 months."

Where has Moore's law brought us?

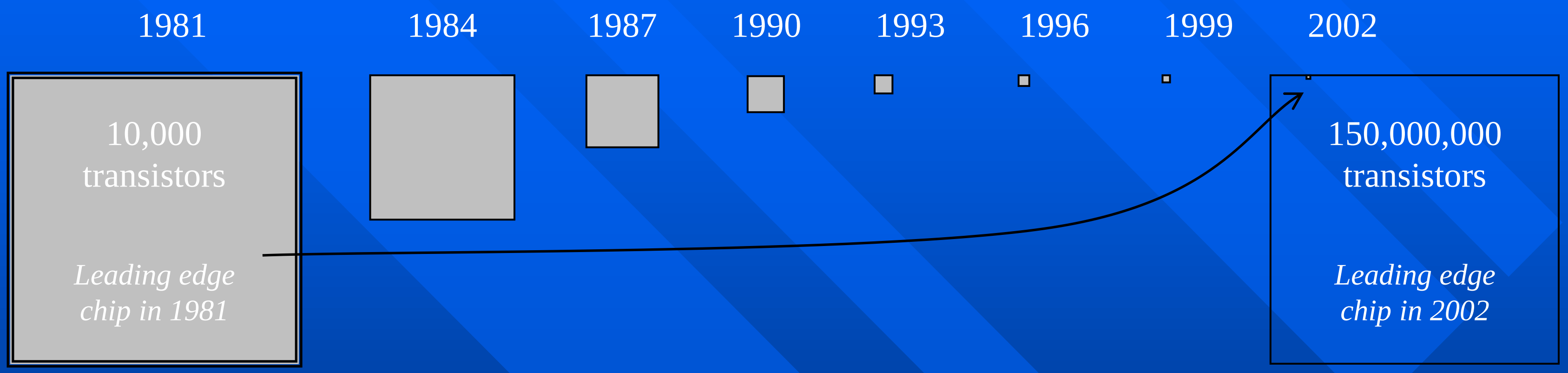
- Moore's law has had almost 40 years of validity
- The Semiconductors Industry Association (SIA) roadmap for process technology predicts the progress to go on for at least the next 10 - 15 years
- We are presently
  - Developing products in 90 nm
  - 65 nm is in development
  - 45 nm in research

- Physical limits (like e.g. the wavelength of light) have been predicted to stall the progress.......but over time all issues were resolved
- So we may be able to realize 10 nm and smaller geometries, which would produce another factor of 81 more transistors on the same area

Number of Transistors on Chips



Graphical illustration of Moore's law



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

Characteristics of Embedded Systems

- Application-specific functionality – specialized for one or one class of applications
- Deadline constrained operation – system may have to perform its function(s) within specific time periods to achieve successful results
- Resource challenged – systems typically are configured with a modest set of resources to meet the performance objectives
- Power efficient – many systems are battery-powered and must conserve power to maximize the usable life of the system.
- Form factor – many systems are light weight and low volume to be used as components in host systems
- Manufacturable – usually small and inexpensive to manufacture based on the size and low complexity of the hardware.

Design Constraints

- **Small Size, Low Weight**
  - Hand-held electronics
  - Transportation applications -- weight costs money
- **Low Power**
  - Battery power for 8+ hours (laptops often last only 2 hours)
  - Limited cooling may limit power even if AC power available
- **Harsh environment**
  - Heat, vibration, shock
  - Power fluctuations, RF interference, lightning
  - Water, corrosion, physical abuse
- **Safety-critical operation**
  - Must function correctly
  - Must *not* function *in*correctly
- **Extreme cost sensitivity**
  - $.05 adds up over 1,000,000 units

Design Challenges

- 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?

Slide credit – P Koopman, CMU

- **More importantly – optimising design metrics!!**

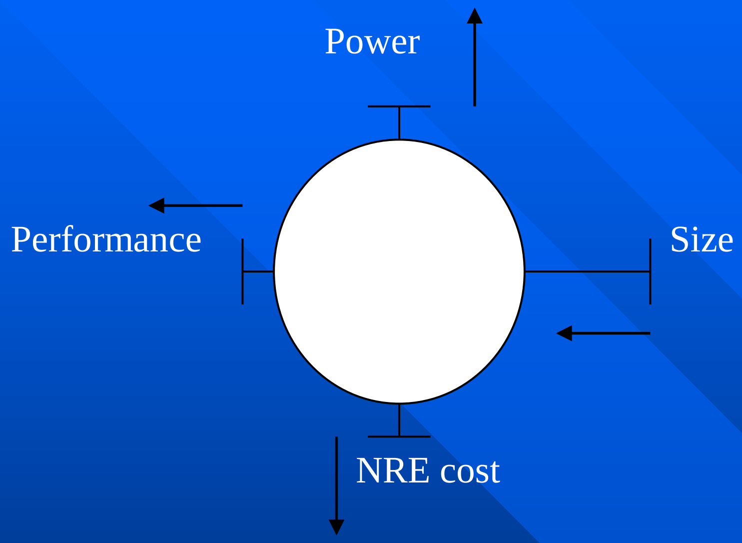
Design Metrics

- 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 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

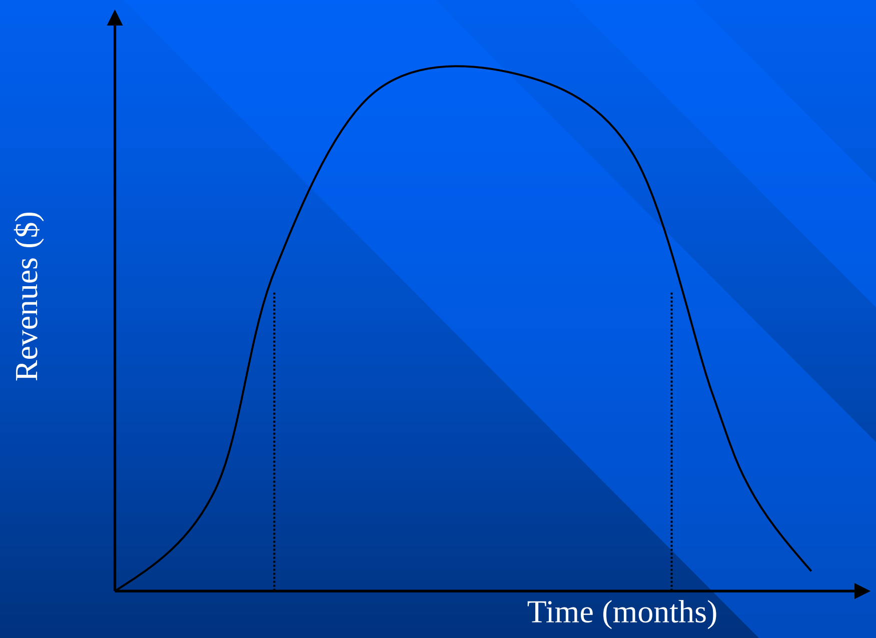
Trade-off in Design Metrics



- 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

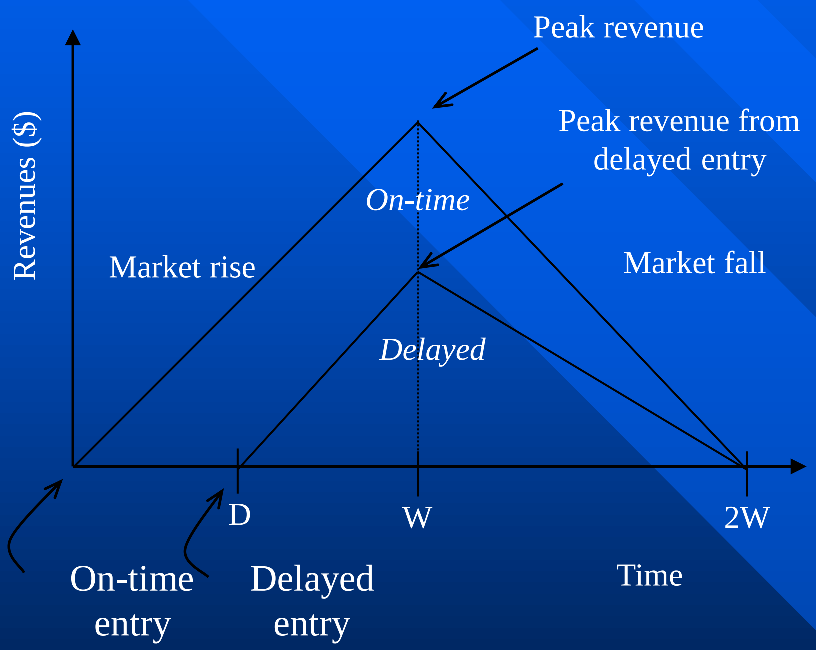
Slide credit Vahid/Givargis, Embedded Systems Design: A Unified Hardware/Software Introduction

Time-to-market: a demanding design metric



- 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

Other Design Considerations

- Dependability
  - Reliability: probability of system working correctly provided that it worked at time t=0
  - Maintainability: probability of system working correctly d time units after error occurred. [Some systems require no maintenance throughout their operating lives (e.g. electric kettles, computer keyboards), while some may need it such as mobile phones and airplane flight control (software upgrade)]

Other Design Considerations

- Dependability
  - Availability: probability of system working at time t
  - Safety
  - Security: in communication

Basically, critical applications have to operate correctly at all time e.g. airplane flight control computer. This includes both **<u>hardware and software</u>** aspects.

Example of System Fault

Finite Precision Can Lead to Disaster

Example: Failure of Patriot Missile (1991 Feb. 25)

Source http://www.math.psu.edu/dna/455.f96/disasters.html

American Patriot Missile battery in Dharan, Saudi Arabia,
failed to intercept incoming Iraqi Scud missile
The Scud struck an American Army barracks, killing 28

Cause, per GAO/IMTEC-92-26 report: "software problem"
(inaccurate calculation of the time since boot)

Specifics of the problem: time in tenths of second
as measured by the system's internal clock
was multiplied by 1/10 to get the time in seconds
Internal registers were 24 bits wide
1/10 = 0.0001 1001 1001 1001 1001 100 (chopped to 24 b)
Error $\cong 0.1100\ 1100 \times 2^{-23} \cong 9.5 \times 10^{-8}$
Error in 100-hr operation period
$\cong 9.5 \times 10^{-8} \times 100 \times 60 \times 60 \times 10 = 0.34$ s
Distance traveled by Scud = (0.34 s) $\times$ (1676 m/s) $\cong$ 570 m
This put the Scud outside the Patriot's "range gate"

Ironically, the fact that the bad time calculation
had been improved in some (but not all) code parts
contributed to the problem,
since it meant that inaccuracies did not cancel out

Other Design Considerations

- Operating environment

  Some engine Electronic Control Units (ECUs) in cars are located under the bonnets. So they have to work at high temperature, as well as dusty and wet environment.

- EMI (Electromagnetic Interference)

Real-Time Consideration

- Correct operation of real-time systems means:
  - Working correctly (functionally correct)
  - Producing outputs **<u>in time</u>!**
- i.e. correct result at the right time

Hard Real-time

- System designed to meet all deadlines
- A missed deadline is a design flaw
- For examples: ABS brake, nuclear reactor monitoring system
- System hardware (over) designed for worst-case performance
- System software rigorously tested
- Formal proofs used to guarantee timing correctness

Firm Real-time

- System designed to meet all deadlines, but occasional missed deadline is allowed
  - Sometimes statistically quantified (e.g. 5% misses)
- For examples: multimedia systems
- System hardware designed for average case performance
- System software tested under average (ideal) conditions

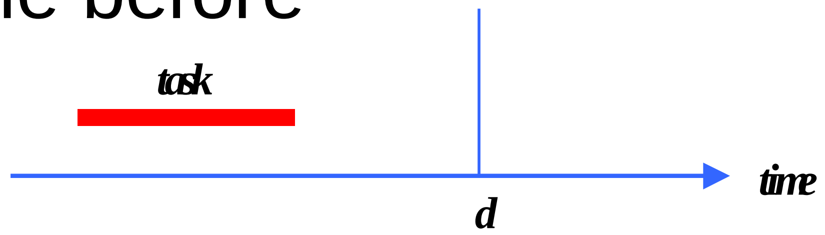
Soft Real-time

- System designed to meet as many deadlines as possible
  - Best effort to complete within specified time, but may be late
- For examples: network switch or router
- System hardware designed for average case performance
- System software tested under averaged (ideal) conditions

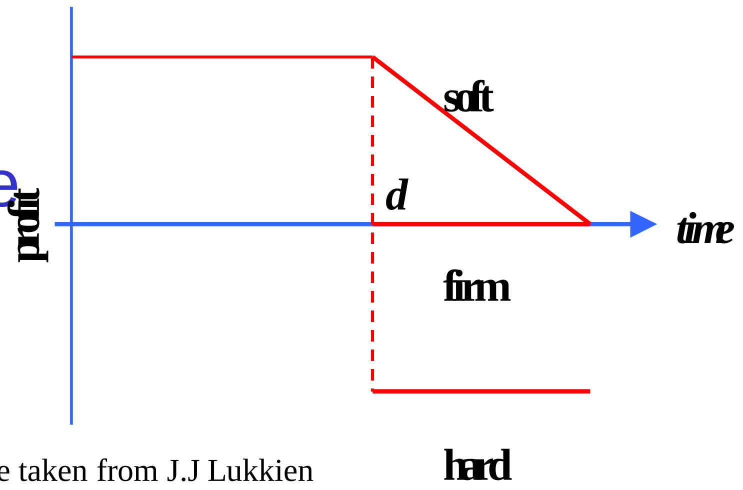
Real-time Systems Deadlines

Deadlines

- **Deadline**: maximum time before a task must complete



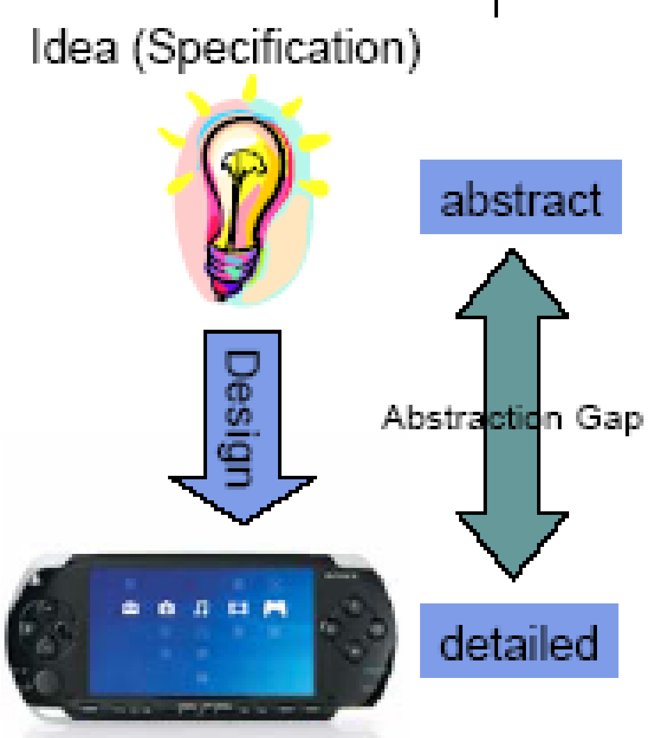
- The *profit* associated with execution of a task is after the deadline:
  - **Hard** deadline: negative
  - **Firm** deadline: 0 (either make it or just don't do it)
  - **Soft** deadline: decreasing with time



Slide taken from J.J Lukkien

Levels of Embedded System Design

- Specification
  - Design productivity increases with the level of abstraction
  - The task of functional verification is very difficult at low abstraction levels
- Implementation
  - Efficient implementations require to exploit the low-level features of the target architecture



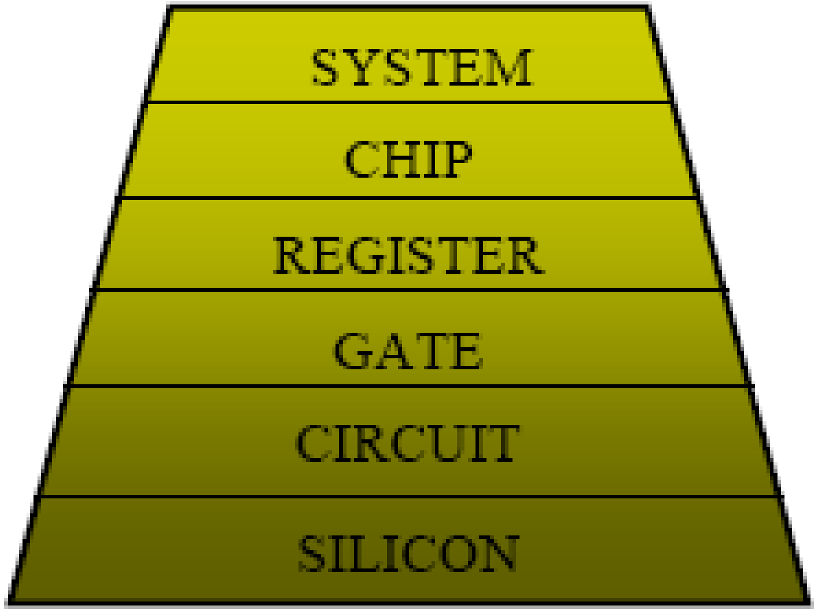
Design Abstraction

- Start of design process
  - overall functionality has to be understood and captured
  - system components have to be identified
  - details are not important yet
- ➢ system shall be modeled at a high abstraction level

- Implementation Phase
  - Implementation details are important to fine-tune the design
- ➢ Low abstraction level is needed

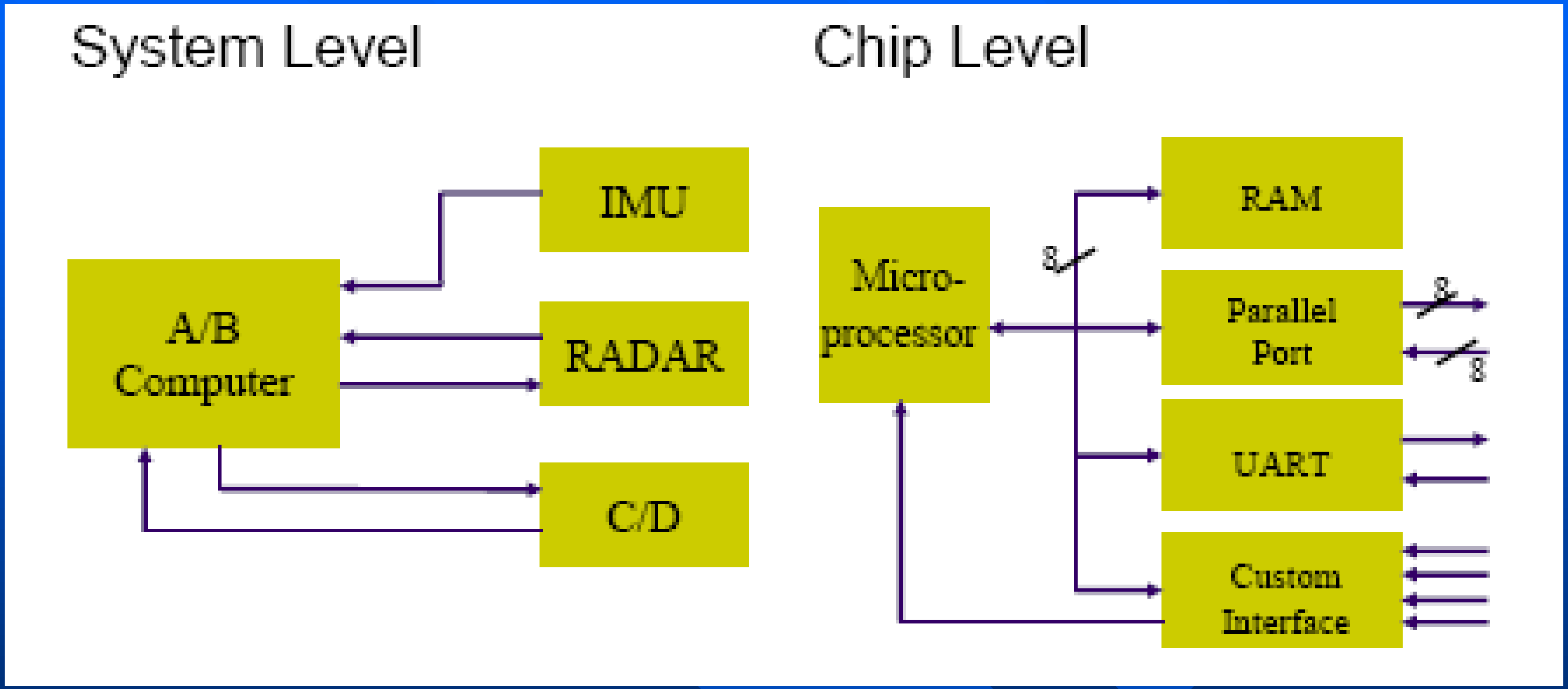
Abstraction Levels

- It is important to work on the right level of abstraction
- The higher the level of abstraction, the shorter the design time
- The lower the level of abstraction, the more details can be fine-tuned



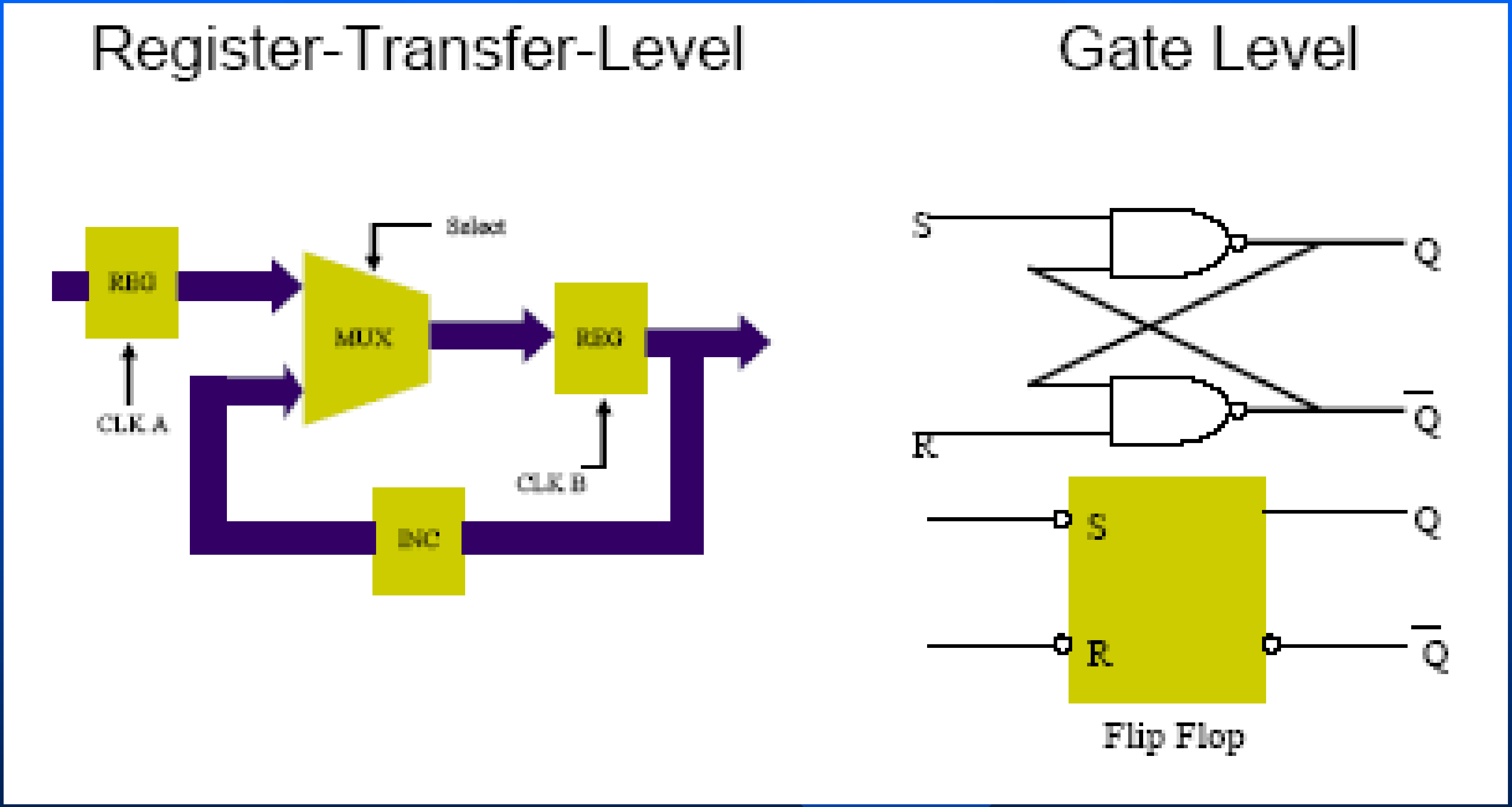
Slide credit – Ingo Sander

Abstraction Levels



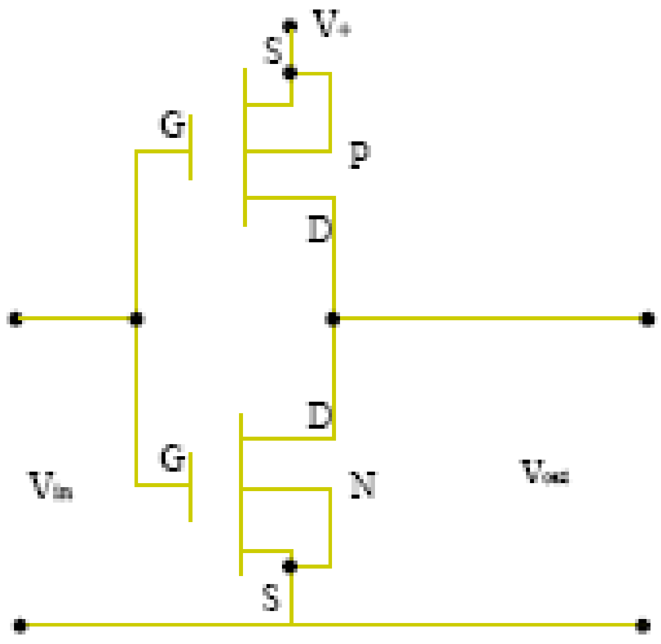
Slide credit – Ingo Sander

Abstraction Levels

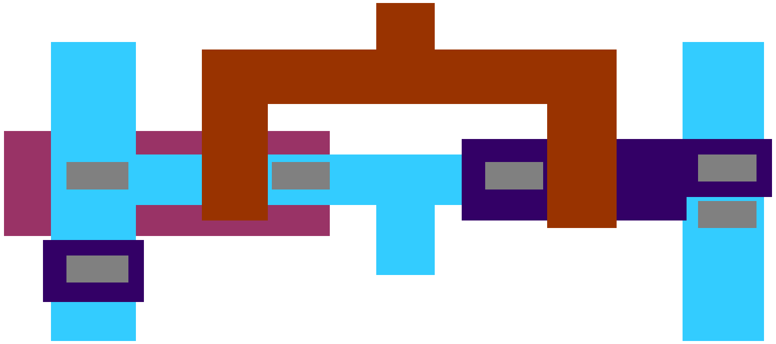


Abstraction Level

Transistor Level



Silicon Level



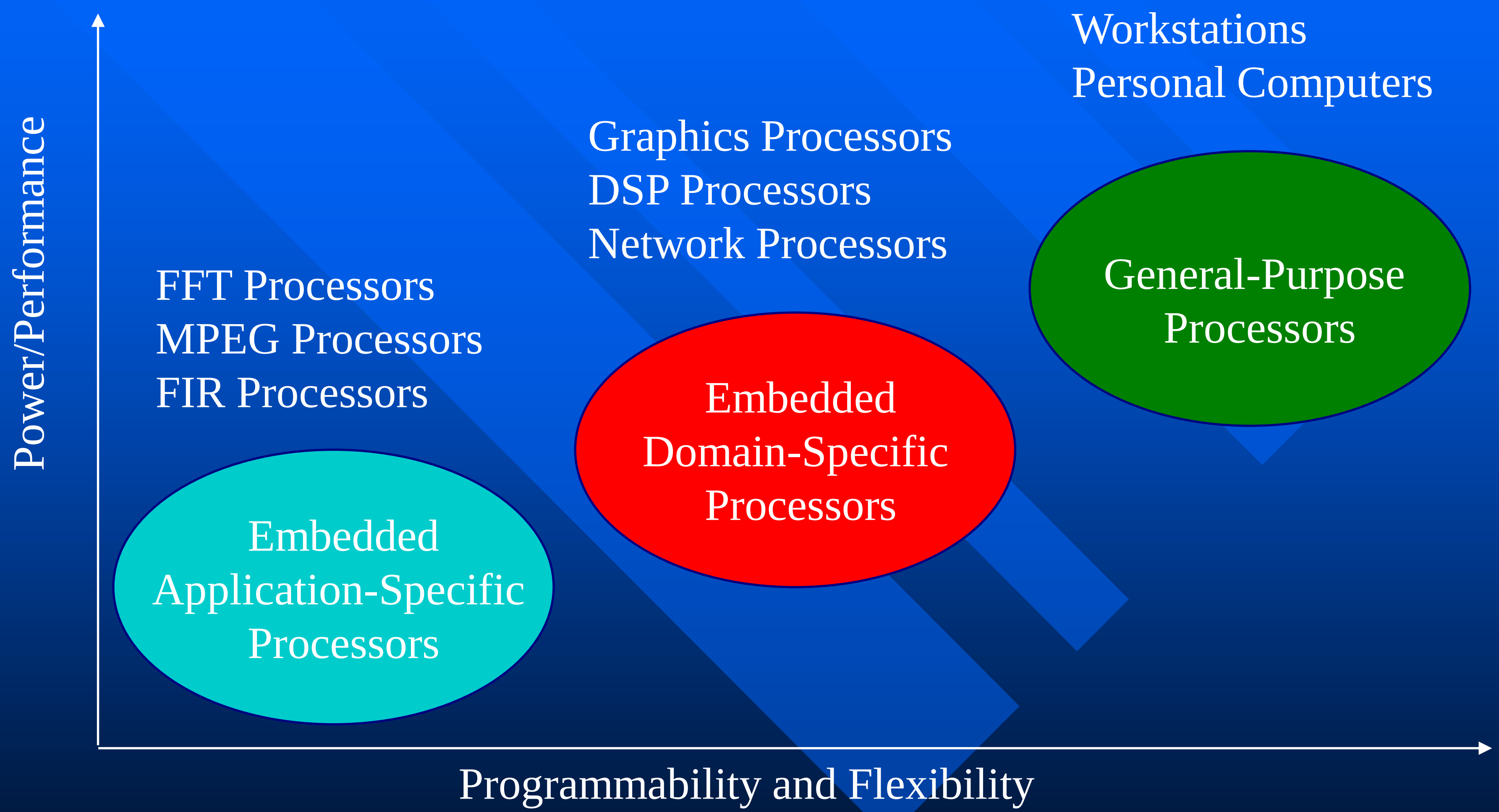
Hardware vs Software

- Many functions can be done by **software** on a general purpose microprocessor OR by **hardware** on an application specific ICs (ASICs)
- For examples: game console graphic, PWM, PID control
- Leads to Hardware/Software Co-design concept

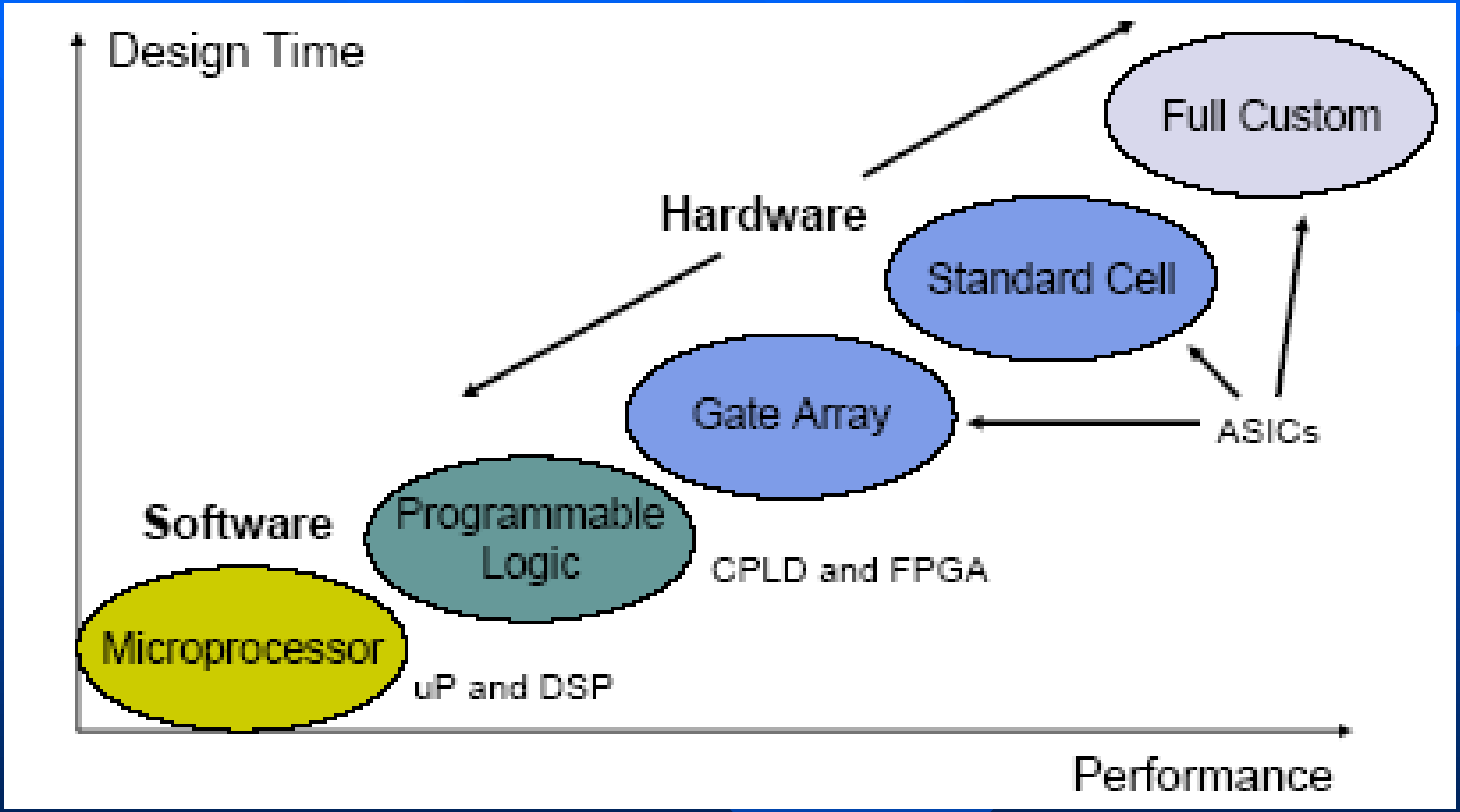
Hardware or Software?

- Where to place functionality?
  - ex: A Sort algorithm
    - » Faster in hardware, but more expensive.
    - » More flexible in software but slower.
    - » Other examples?
- Must be able to explore these various trade-offs:
  - Cost.
  - Speed.
  - Reliability.
  - Form (size, weight, and power constraints.)

Hardware vs Software

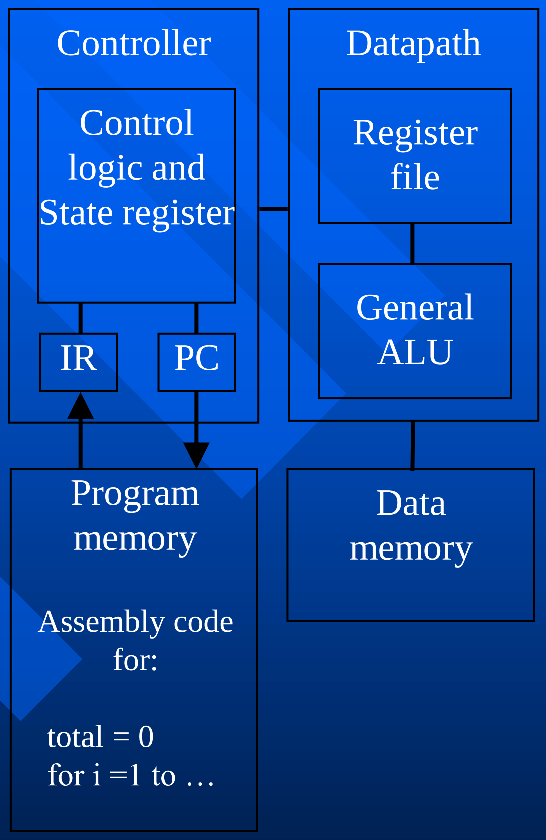


Hardware vs Software



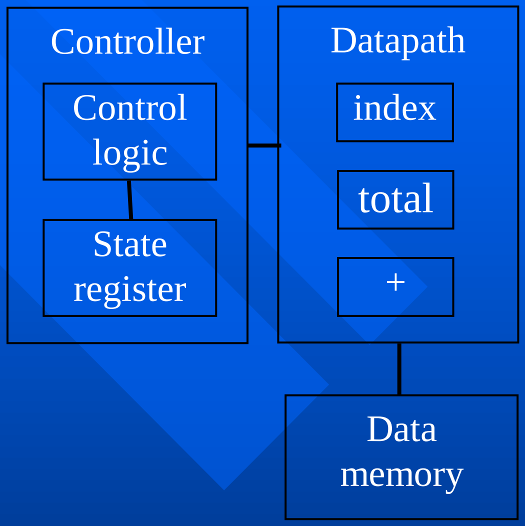
General-purpose processors

- 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



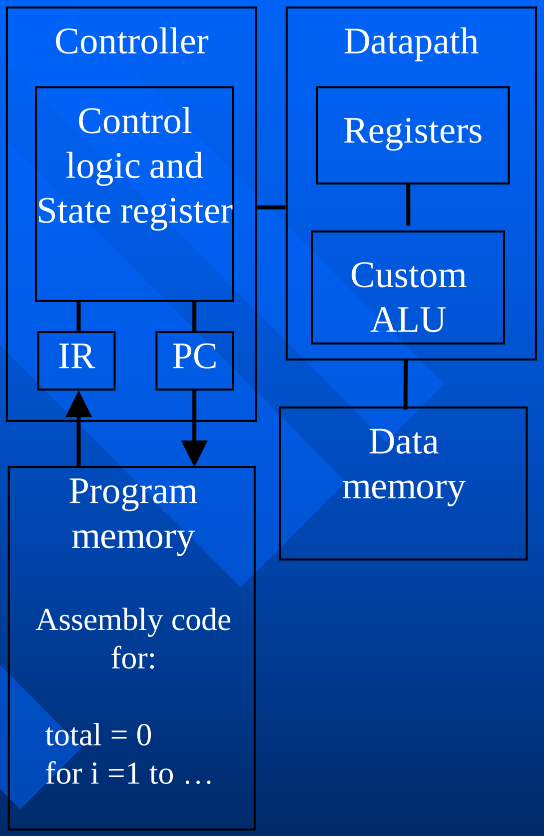
Single-purpose processors

- 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

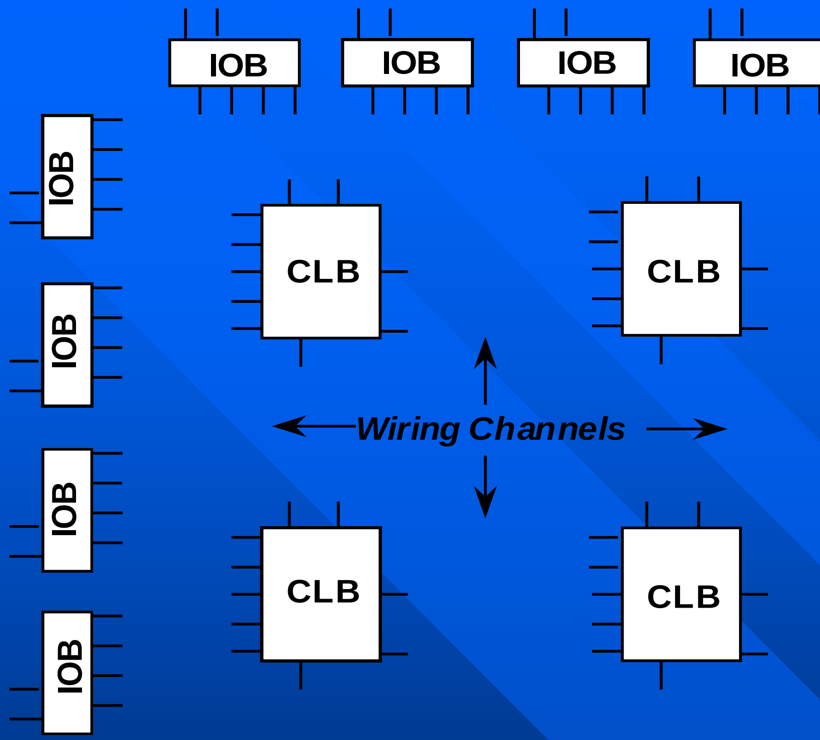


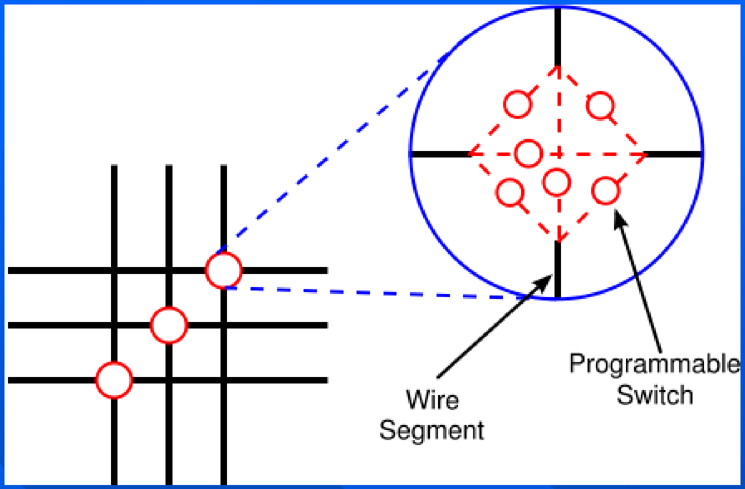
Application-specific processors

- 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
- DSPŤจัดอยู่ในประเภทนี้ด้วย



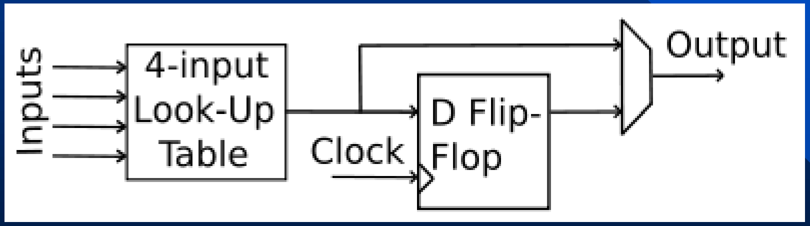
FPGA Architecture





Programmable switch at wiring intersection
(credit: www.wikipedia.com)

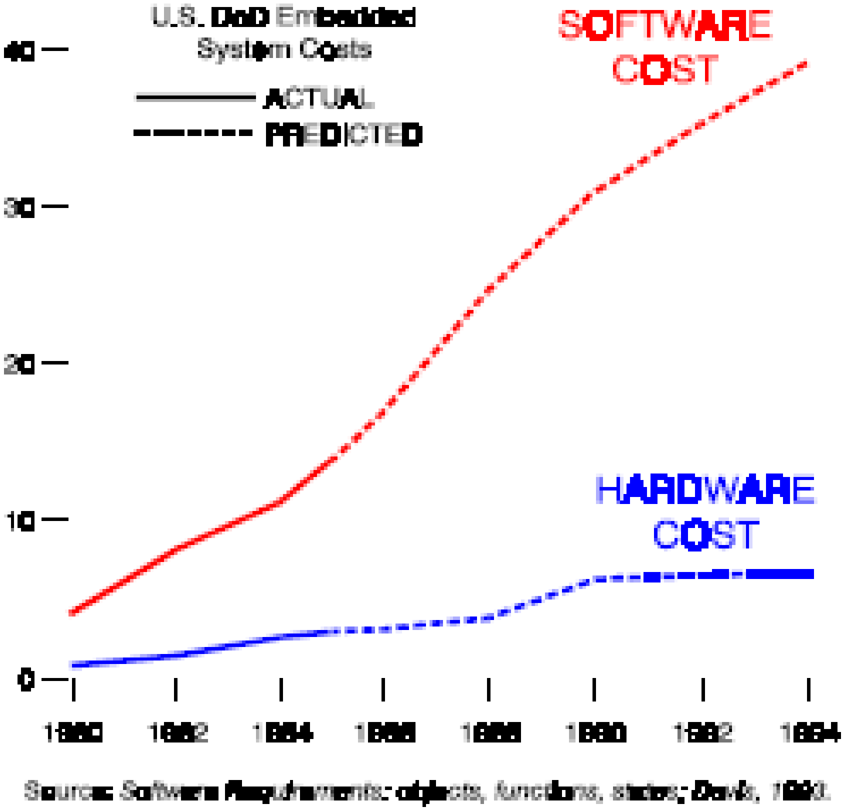
FPGA layout with Configurable Logic Blocks (CLB) and I/O Blocks (IOB) *(credit: Katz's Contemporary Logic Design)*



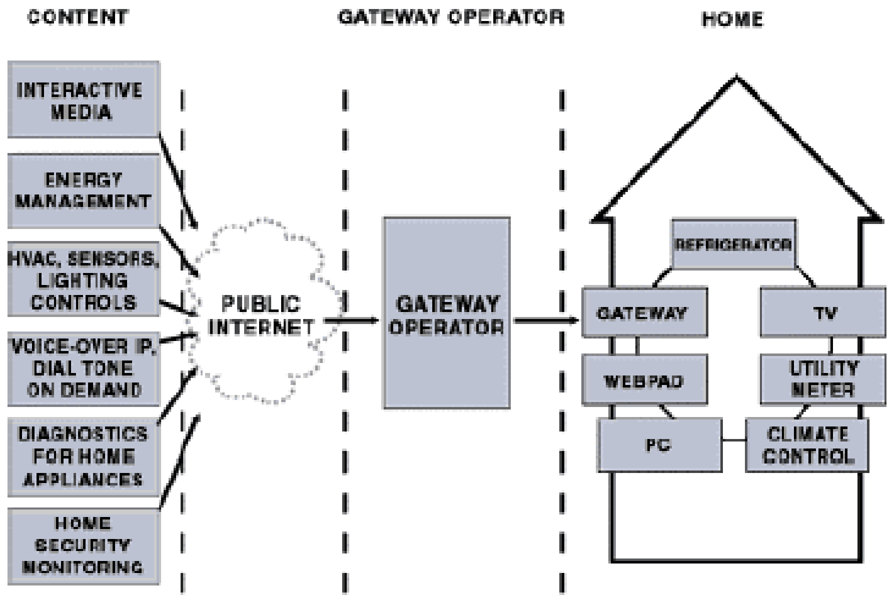
- Highly constrained products tend to use application specific processors
  - Many mobile phones (power&size constrained) contain ARM chips
  - Hi-Fi (high performance&time constrained) contain DSP chips

Software Costs

- **Hardware is mostly a recurring cost**
  - Cost proportional to number of units manufactured
- **Software is a "one-time" non-recurring engineering design cost (NRE)**
  - Paid for "only once"
    - But bug fixes may be expensive, or impossible
  - Cost is related to complexity & number of functions
  - Market pressures lead to feature creep
  - <u>*SOFTWARE Is Not FREE!!!!!*</u>



Future Embedded Systems



HOME GATEWAY USAGE SCENARIO

- **Will people adopt this other than as a toy?**
  - Will the same people who can't set time on a VCR be able to debug their house?

- **If we can make the system readily accessible, reliable, affordable, …the possibilities are almost endless**