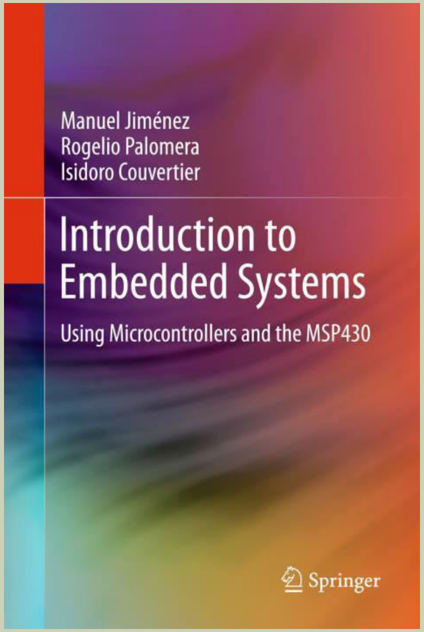
# LECTURE 1: INTRODUCTION

M. Jiménez, R. Palomera, & I. Couvertier

## INTRODUCTION TO EMBEDDED SYSTEMS: Using Microcontrollers and the MSP430






Lecture Slides Series

# OUTLINE

- Embedded Systems: History and Overview
- Structure of an Embedded
- Classification of Embedded Systems
- The Life Cycle of Embedded Designs
- Design Constraints

# 1.1 EMBEDDED SYSTEMS: HISTORY & OVERVIEW

- Early Forms of Embedded Systems
- Birth and Evolution of Modern Embedded Systems
- Contemporary Embedded Systems

# WHAT IS AN EMBEDDED SYSTEM?

- Definition
  - An electronic systems containing tightly coupled hardware and software components
- Characteristics
  - Perform a single function
  - Form part of a larger system
  - Not intended to be independently programmable by the user
  - Are expected to work with minimal or no human interaction
  - Reactive, real-time operation
  - Tightly constrained

# EARLY EMBEDDED SYSTEMS

- Early Computers
  - Similar to an ESYS
    - Single functioned
    - Not user programmable
  - Unlike today's ESYS
    - Large and power thirsty
    - Not integrated

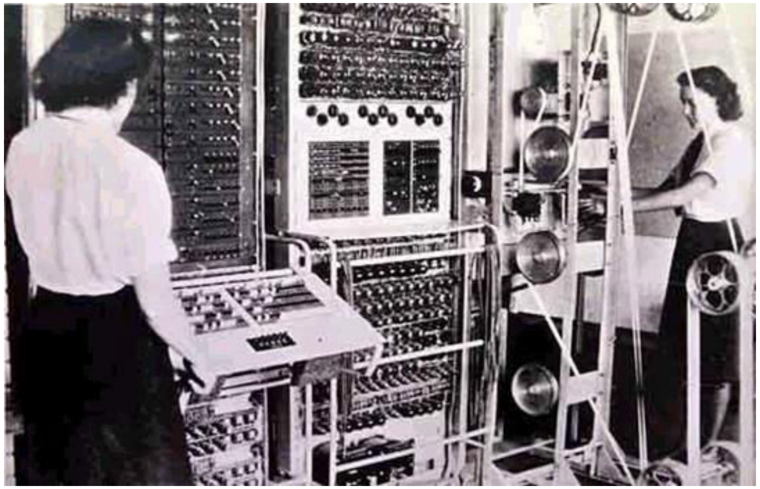


Fig. 1.1: Control panel and paper tape transport view of a Colossus Mark II computer *(Public image by the British Public Record Office, London)*

- First Embedded System
  - The Apollo Guidance Computer (AGC)
    - Guidance & Navigation
    - CPU + MEM + I/O
    - Low-power mode
    - AGC Assembly Programmed

Fig. 1.2: AGC user interface module *(Public photo EC96-43408-1 by NASA)*

# MODERN EMBEDDED SYSTEMS

- **Born with the Microprocessor**
  - TMS1000: The first microcontroller
  - Intel 4004: The first commercial microprocessor
  - US Navy CADC: High-performance embedded system

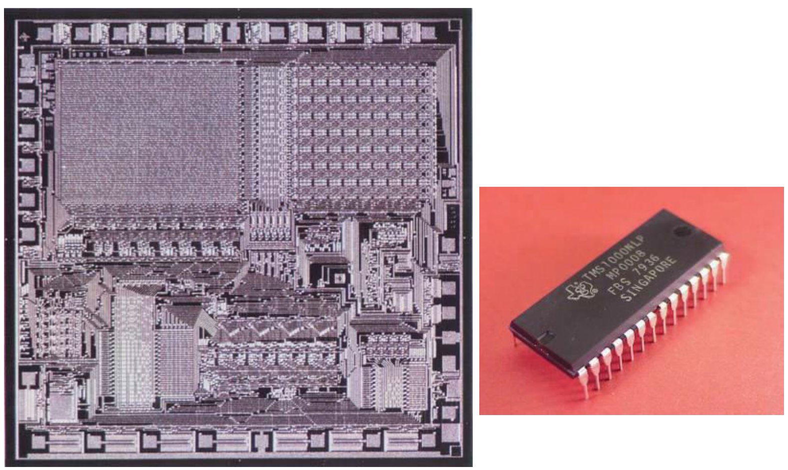


Fig. 1.3: Die microphotograph (left) packaged part for the TMS1000 *(Courtesy of Texas Instruments, Inc.)*

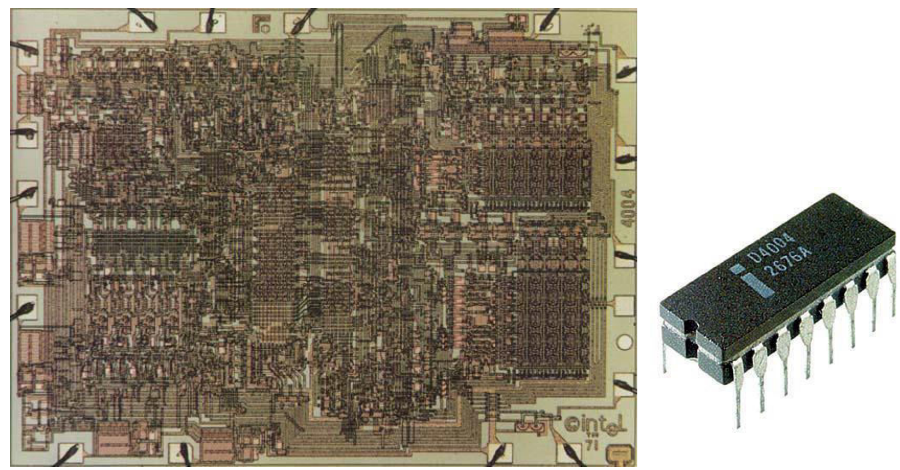


Fig. 1.4: Die microphotograph (left) and packaged part for the Intel 4004 *(Courtesy of Intel Corporation)*

# TODAY'S MCU MARKET

- Plenty of Vendors
  - TI (MSP430)
  - Microchip (PIC)
  - Intel (8051, 80x86)
  - Freescale (HC11, HC08)
  - ARM Limited (ARM7)
  - Atmel (ATmega)
- Plenty of Sizes
  - 4-, 8-, 16-, 32-, and 64-bit
  - CISC Vs. RISC
  - Harvard Vs. vonNeumann
- Wide Market
  - Over 6 Billion chips per year
  - Over $50 billion sales

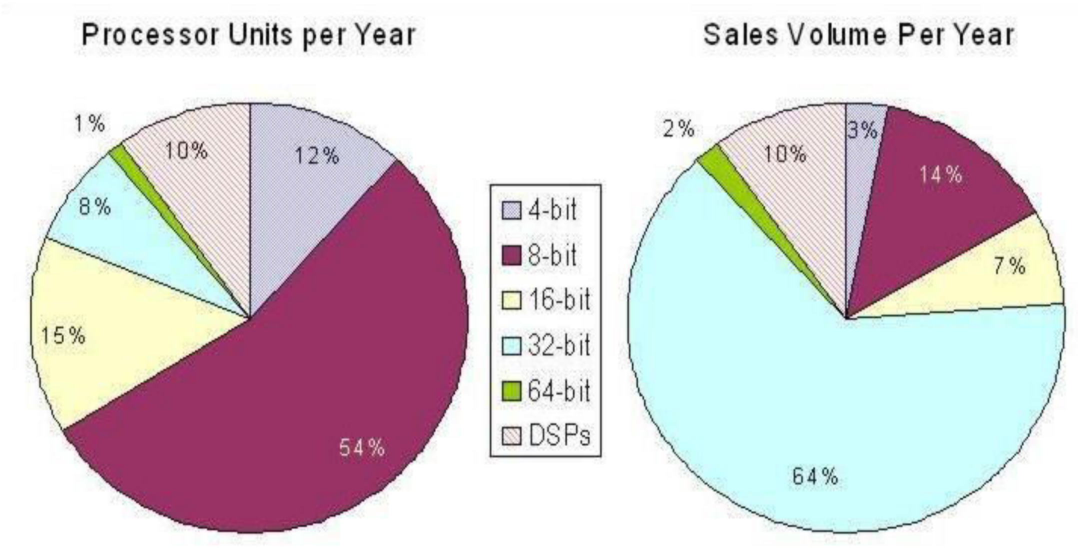



Fig. 1.5: Estimates of processor market distribution *(Source: Embedded Systems Design — [www.embedded.com](www.embedded.com))*

# CONTEMPORARY EMBEDDED SYSTEMS

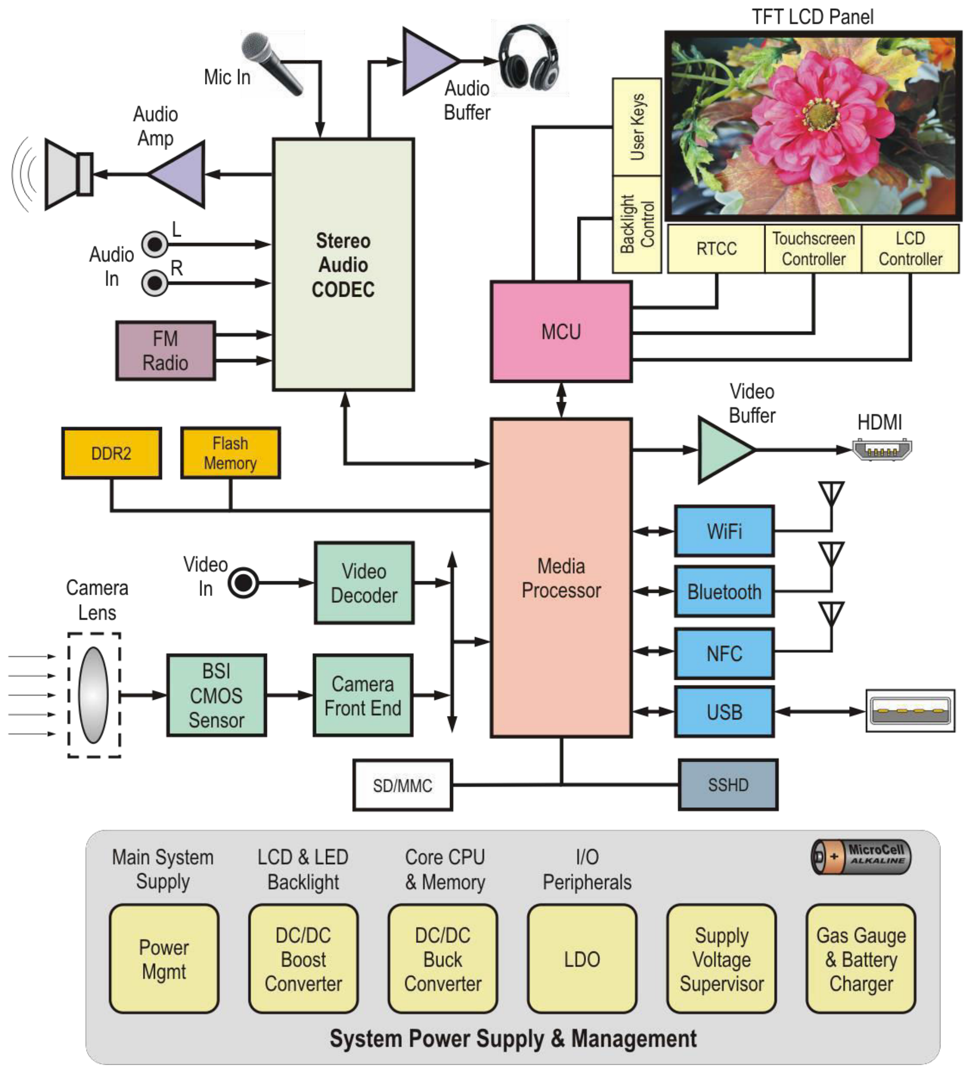



Fig. 1.6 Generic multi-function media player.

- System Components
  - Power management
  - Video processing
  - Audio processing
  - Communications
  - User interfaces
  - Dedicated ASICs
  - Memory management
  - Storage
- Multi-embedding
  - Most components are embedded system by themselves
  - System integration

# 1.2 STRUCTURE OF AN EMBEDDED SYSTEM

- Hardware Components
- Software Components

# EMBEDDED SYSTEM STRUCTURE

- Hardware Components: Electronics Infrastructure
  - CPU
  - Memory
  - I/O Subsystem
- Software Components: System Functionality
  - Firmware
  - Operating System
  - Application Programs

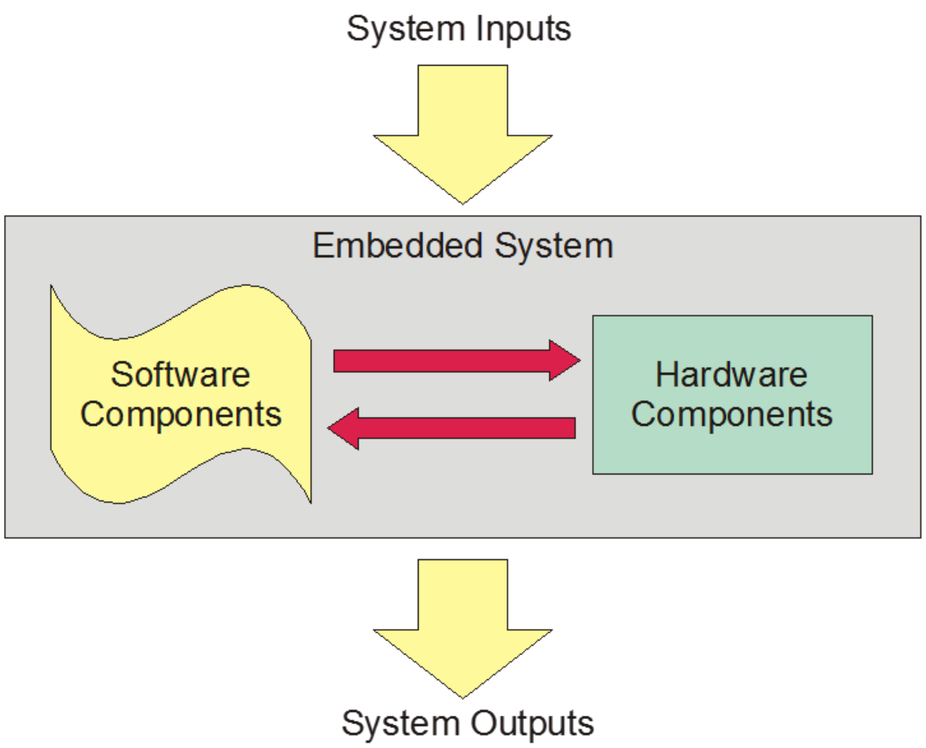



Fig. 1.7: General view of an embedded system

# HARDWARE COMPONENTS

- Central Processing Unit
  - Registers, ALU, CU
- Memory
  - Program Memory
  - Data Memory
- I/O Devices
  - Communication ports
  - User Interfaces
  - Sensors & actuators
  - Diagnostics support
  - System controllers
  - Power management
  - Specialized ASICs

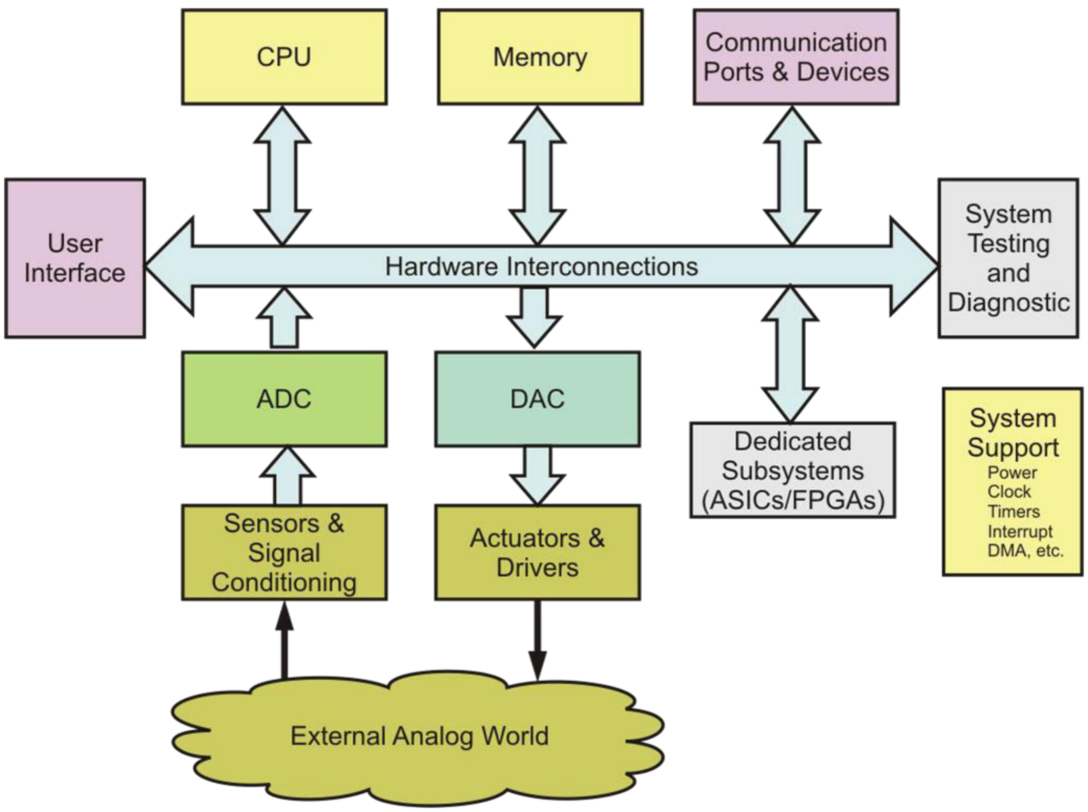



Fig. 1.8: Hardware elements in an embedded system

# SOFTWARE COMPONENTS

- System Tasks
  - Actions making use of system resources
- System Kernel
  - Manages system resources
  - Coordinates task services
- Services
  - Routines performing specific tasks

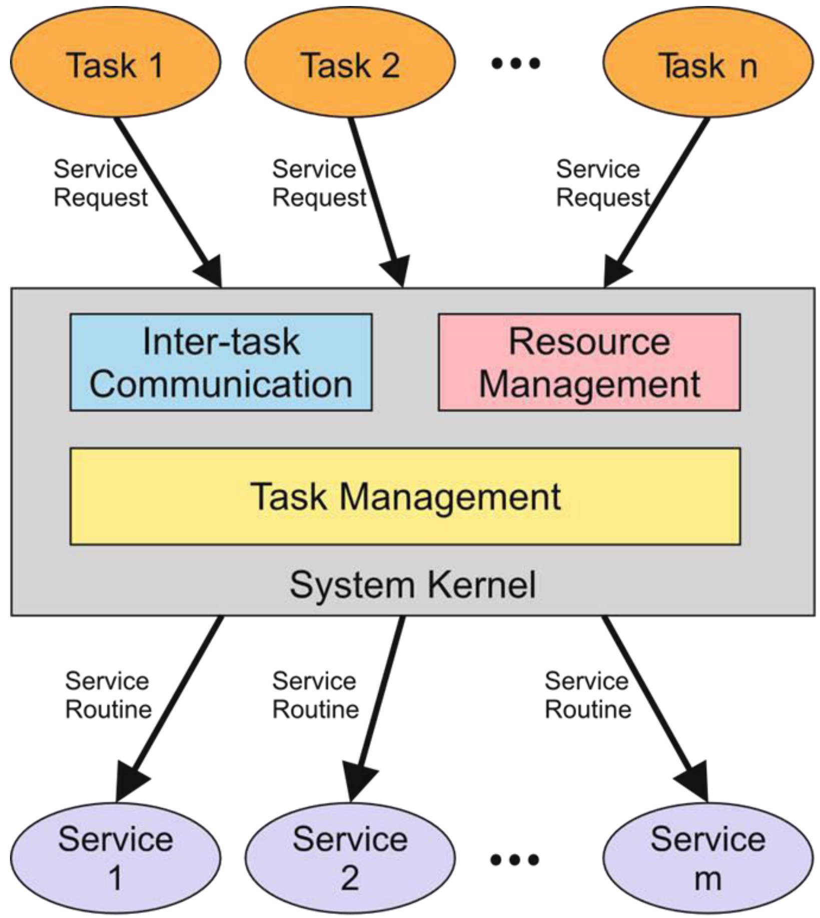



Fig. 1.9: Software structure in an embedded system

# 1.3 A CLASSIFICATION OF EMBEDDED SYSTEMS

- Small
- Distributed
- High-performance

# A CLASSIFICATION OF EMBEDDED

- Small
  - MCU-based, low component count
  - Large volume
  - Single tasked
  - Low-cost, maintenance free
- Distributed
  - Multi-chip, board-level
  - Multi-tasked
  - Medium volume & cost
  - Maintainable, upgradeable
- High-performance
  - Dedicated board-level hardware
  - Task intensive, RTOS-based
  - Low-volume, high cost
  - High maintenance

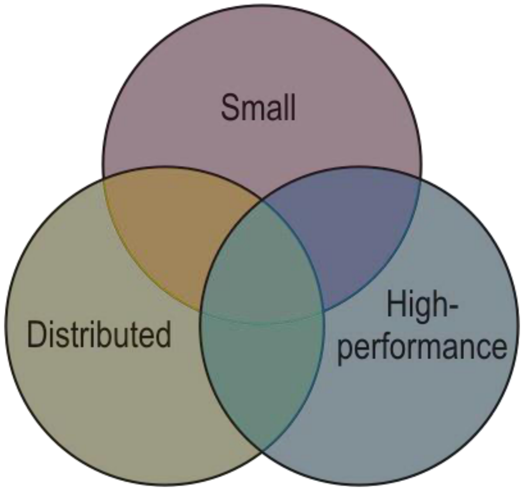



Fig. 1.10 A classification of embedded systems

# 1.4 THE LIFE CYCLE OF EMBEDDED DESIGNS

- Birth
- Design
- Growth
- Maturity
- Decline

# EMBEDDED DESIGN LIFE CYCLE

- Birth
  - Need & opportunity
  - Specifications
- Design
  - Proof-of-concept
  - Manufacturing design
- Growth
  - Production & deployment
- Maturity
  - Maintenance & upgrade
- Decline
  - System disposal

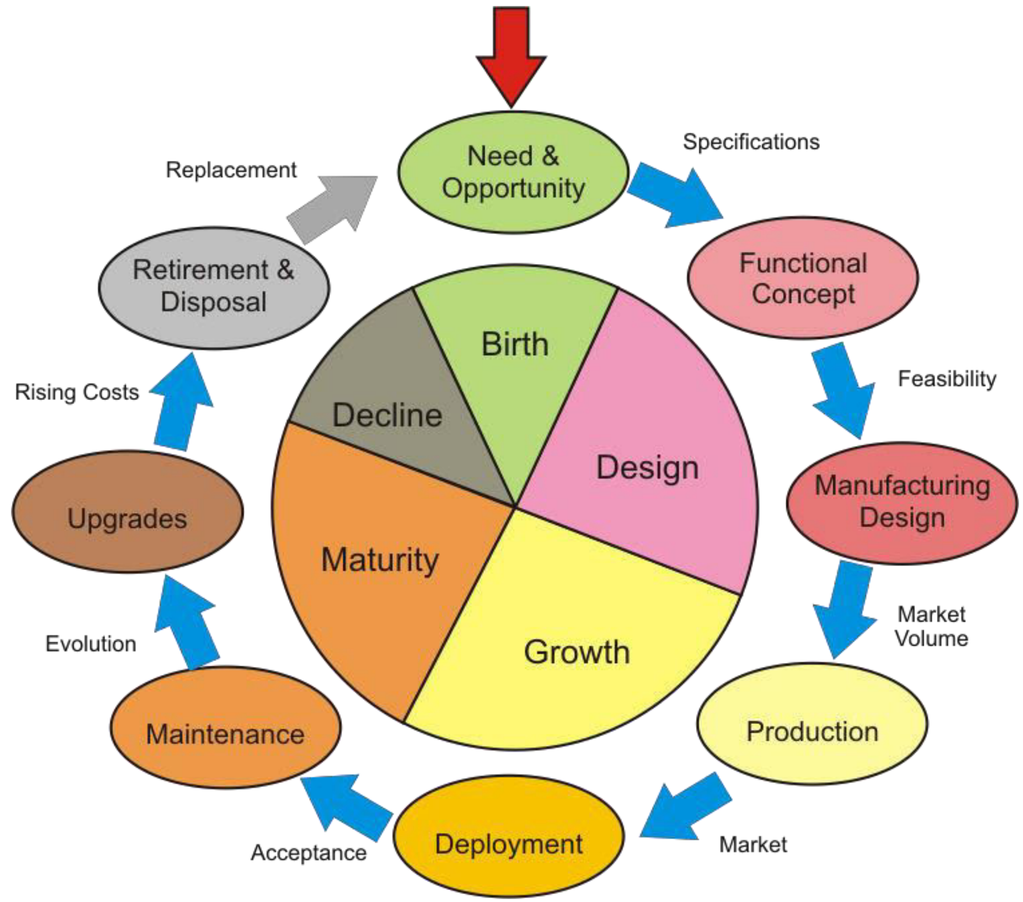



Fig. 1.11: Life cycle of an embedded design

# DESIGN ENDURANCE

- Embedded Design Goal
  - Design must successfully complete all pertinent stages
  - Not all designs go through all stages

- Plan for Each Stage
  - Designer's vision and planning needed for success
  - Good designs do not happen by chance

# 1.5 DESIGN CONSTRAINTS

- Functionality
- Cost
- Performance
- Power and Energy
- Time-to-Market
- Reliability and Maintainability

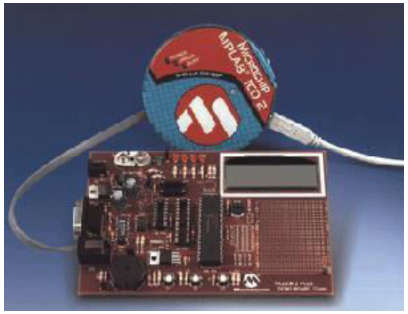
# DESIGN CONSTRAINTS

- Functionality
  - System ability to perform the function it was designed for (REQ)
- Cost
  - Amount of resources needed to conceive, design, and produce an embedded system
- Performance
  - System ability to perform its function in time.
  - Affected by both HW & SW factors
- Size
  - Physical space taken by a system solution.
- Power and Energy
  - Energy required by a system to perform its function.
- Time to Market
  - The time it takes from system conception to deployment.
- Maintainability
  - System ability to be kept functional during its mature life.

# FUNCTIONALITY (1/2)

- Functional verification is a difficult task
  - Can consume up to 70% o development time
- Verification Methods
  - Simulation Techniques
    - Behavioral (HDL-based)
    - Logic (Circuit Modeling)
    - Processor (Software)
  - JTAG Debugger
    - Hardware supported through dedicated ports
    - Used also for testing (boundary scan test)
    - Cost effective

MSP430 FET Tool
Courtesy of Texas Instruments Inc.




PIC In-circuit Debugger
Courtesy of Microchip Corporation

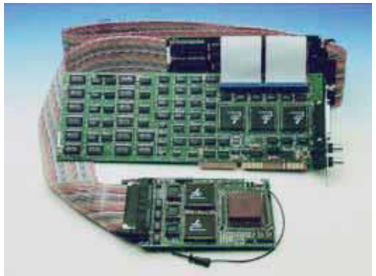
# FUNCTIONALITY (2/2)

- In-Circuit Emulators
  - Replace MCU in target system
  - A powerful debugger
  - Expensive
- ROM Monitors
  - Monitor functions in ROM
  - Status sent via serial port


68HC11 In-circuit Emulator
Courtesy of Nohau Systems


8051 In-circuit Emulator
Courtesy of Signum Systems

ICE Test Pod
Courtesy of Signum Systems



8086 In-circuit Emulator
Courtesy of Nohau Systems

# SYSTEM COST

- The cost of a given Volume (V) of units:

$$C_T = NRE + (RP \cdot V) \therefore$$
$$U_C = \frac{C_T}{V} = \frac{NRE}{V} + RP$$

- NRE = Non-Recurrent Engineering costs (Fixed)
  - Investment to complete all design aspects
  - Very large and independent of volume in CT
  - Include man-hours, infrastructure, and R&D
- RP = Recurrent Production costs (Variable)
  - Expenses in producing each unit of a given volume
  - Small but affected by V in CT
  - Include components, boards, packages, and testing

# COTS-BASED NRE COSTS

- Commercial off-the-shelf parts-based design
  - Traditional methodology for Embedded Systems
  - Minimizes Hardware costs
  - Increases design & verification costs
- NREs in UC are diluted by a large production volume
- Balance between technology choice and production volume

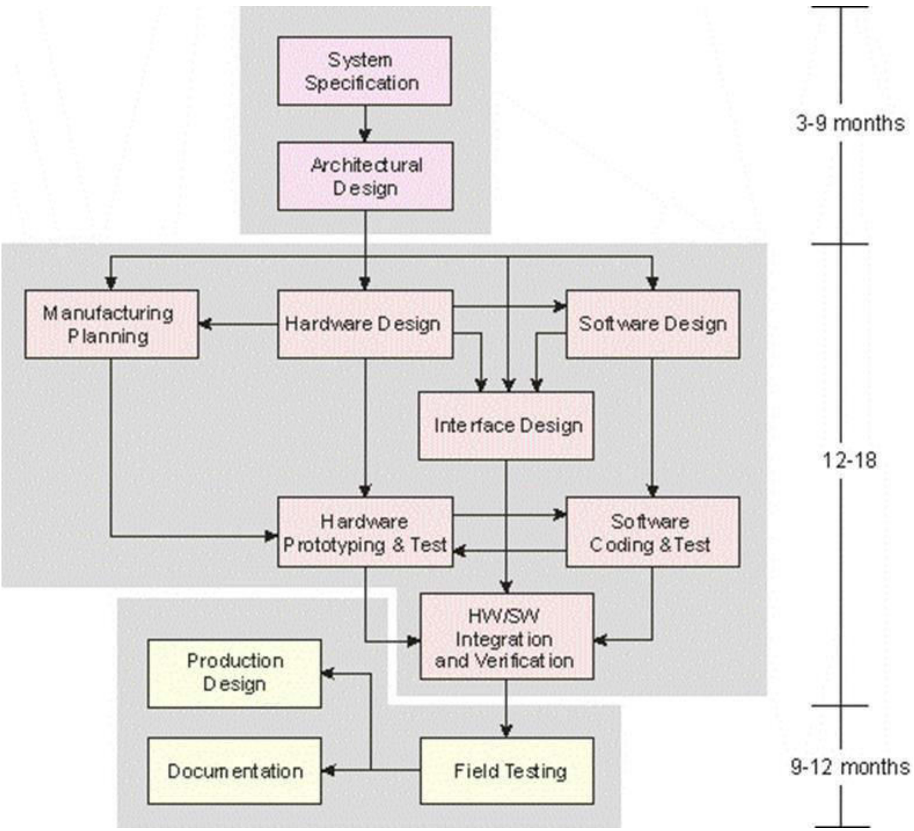



Fig. 1.12 Embedded systems design flow model based on COTS parts

# PERFORMANCE: HW FACTORS

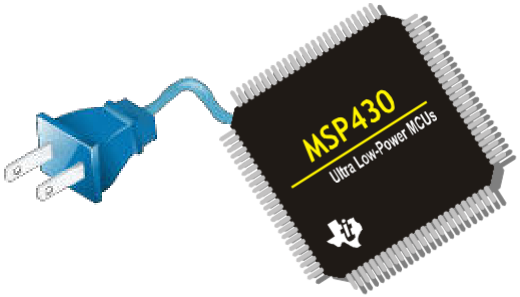
- Clock Frequency
  - System clock speed: not an absolute performance metric
- Architecture
  - Determines how clock cycles are used
- Component Speed
  - Response time and access time
- Handshaking
  - Signalization required to complete a transaction
- Low-power Modes
  - Wake-up times might affect application speed
- High speed is expensive!!!
  - Use it wisely

# PERFORMANCE: SW FACTORS

- Algorithm Complexity
  - Steps and resources needed to complete a task
- Task Scheduling
  - Affects waiting time in multitasking system
- Inter-task Communication
  - Time taken by tasks to exchange information
- Level of Parallelism
  - Software usage of system hardware resources

# POWER & ENERGY (1/2)

- Critical Parameter
  - A long chain of design events depend on it
- System reliability
  - Stress, noise, and heat
- Cooling Costs
  - High power = lot of heat to remove
- Power Supply Requirements
  - Larger batteries of power supply
- Size, Weight, and Form
  - Mechanical system parameters affected by heat density



# POWER & ENERGY (2/2)

- Environmental Impact of Embedded Systems
  - Average individual uses 60 microprocessors per day
  - Household electronics accounts for 11% of all energy consumed in the USA
    - 147,000,000,000 KWh (147TWh) per year
  - Excludes digital TVs and large appliances
  - Excludes industry, schools, hospitals, etc
  - Trend continues to grow… Is there a limit?

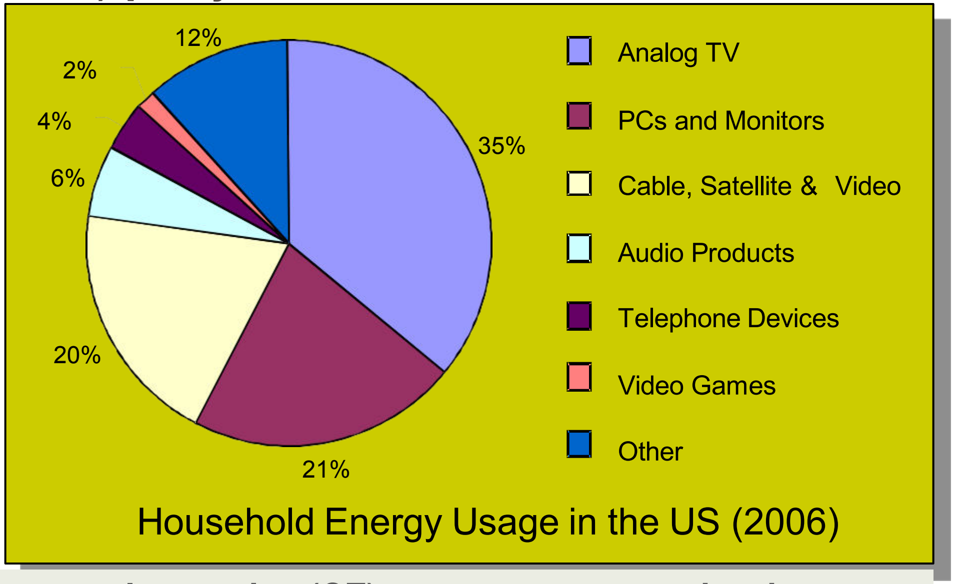



Fig. 1.13 Distribution of U.S. residential consumer electronics (CE) energy consumption in one year *(Source Consumer Electronics Association)*

# TIPS FOR LOW-POWER DESIGNS

- Use low-power MCUs and Peripherals
  - Activate CPU standby and sleep modes
  - Let peripherals do the work while the CPU is off
- Stop the Energy Waste
  - Turn off unused peripherals
- Write power efficient code
  - Every wasted CPU cycle is energy that will never come back
- Use power management techniques
  - Power and clock gating plus efficient coding techniques

# TIME-TO-MARKET (1/2)

- **Critical Constraint for Applications with a Narrow Market Window (W)**

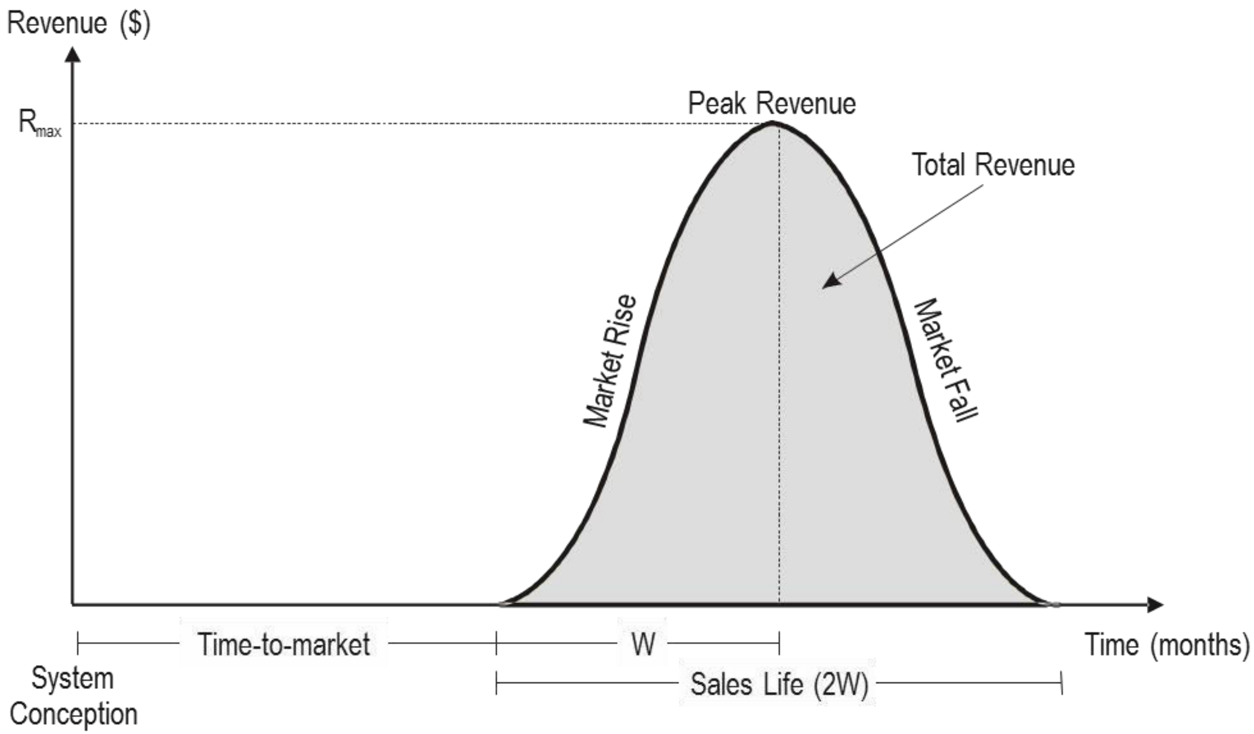
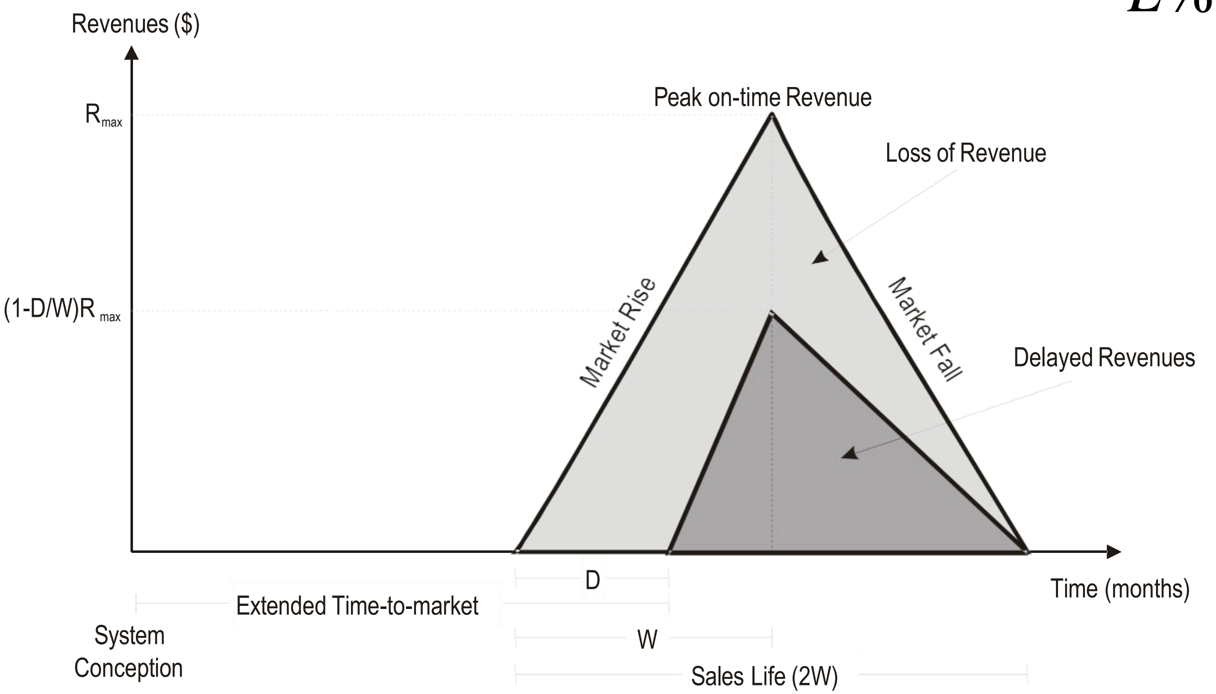



**Fig. 1.14 Typical revenue-time curve for embedded products, denoting the time-to-market and market window**

# TIME-TO-MARKET (2/2)

- A Moderate Market Entry Delay Could Cause a Large Loss of Revenue

$$L\% = \frac{D(3W - D)}{2W^2} \times 100\%$$




Assume:

- 2W = 24 months
- D = 4 months

Estimated Revenue Loss:

- L% = 50%

Fig. 1.15 Linear revenue model with a delayed system deployment

# MAINTAINABILITY

- Maintenance enables reliable system operation throughout entire useful life
- Relevance of maintenance depends on application
  - Expected lifespan
  - Application criticality
- Maintainability is a design requirement
  - Must be included among system specifications
- Must consider both aspects:
  - Hardware Maintenance
  - Software Maintenance

- Four maintenance dimensions
  - Corrective: Fixes faults
  - Adaptive: Copes with a changing environment
  - Perfective: Adds enhancements
  - Preventive: Anticipates events

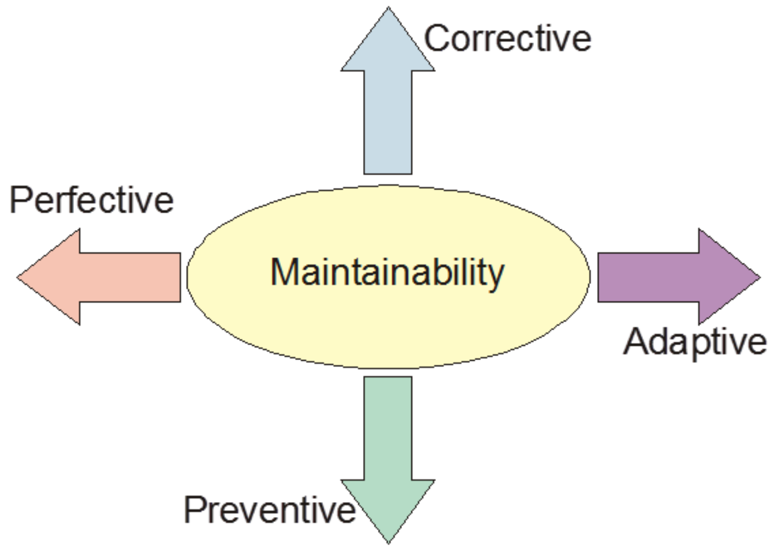



Fig. 1.16 The four actions supporting system maintainability

# HARDWARE MAINTENANCE ISSUES

- Increased NREs
  - Design overhead to support HW maintenance
- Time-to-market Impact
  - Additional development time
- Increases Recurrent Cost
  - More components in system
- Component Obsolescence
  - Limit system useful life span

# SOFTWARE MAINTENANCE ISSUES

- Hardware Constraints
  - Stringent HW constraints leave little room for support functions
- Cost of Verification
  - Undiscovered software bugs become maintenance headaches
- Inadequate Code Documentation
  - Meaningful and up-to-date
- Technology Changes
  - Compatibility with tool newer versions
- Ripple Effect of Changes
  - Identifying effect down the code
- Qualified Personnel
  - Everybody wants to design

# END OF LECTURE 1 SLIDES