EMBEDDED SYSTEM

What is an Embedded system?

- An embedded system is one that has computer hardware with software embedded in it as one of its components.

- Embedded system is the system designed for particular or specific task using dedicated controllers.

- We can say that it is **“A combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a dedicated function. In some cases, embedded systems are part of a larger system or product, as is the case of an antilock braking system in a car. ”.**

An embedded system is a special-purpose computer system designed to perform certain dedicated functions. It is usually *embedded* as part of a complete device including hardware and mechanical parts.

An embedded product uses a microprocessor (or microcontroller) to do one task and one task only

- There is only one application software that is typically burned into ROM

A PC, in contrast with the embedded system, can be used for any number of applications

- It has RAM memory and an operating system that loads a variety of applications into RAM and lets the CPU run them
- A PC contains or is connected to various embedded products
- Each one peripheral has a microcontroller inside it that performs only one task

Why not use PCs for all embedded computing?

First, real-time performance requirements often drive us to different architectures. real-time performance is often best achieved by multiprocessors.
Second, low power and low cost also drive us away from PC architectures and toward multiprocessors.

Challenges in Embedded Computing System Design

1. How much hardware do we need?
*(*too little hardware and the system fails to meet its deadlines, too much hardware and it becomes too expensive*)*
2. How do we meet deadlines?
*(*speed up the hardware or increasing the CPU clock rate may not make enough difference to execution time, since the program's speed may be limited by the memory system.*)*
3. How do we minimize power consumption?
*(*In battery-powered applications, power consumption is extremely important, Even in non battery applications, excessive power consumption can increase heat dissipation.*)*
4. How do we design for upgradability?
5. Does it really work?

Significanceof ES

- Due to their compact size, low cost and simple design aspects made embedded systems very popular and encroached into human lives and have become indispensable. They are found everywhere from kitchen ware to space craft. To emphasize this idea here are some illustrations.

Embedded systems everywhere?

Embedded systems span all aspects of modern life and there are many examples of their use.

a) **Biomedical Instrumentation** – ECG Recorder, Blood cell recorder, patient monitor system

b) **Communication systems** – pagers, cellular phones, cable TV terminals, fax and transreceivers, video games and so on.

c) **Peripheral controllers of a computer** – Keyboard controller, DRAM controller, DMA controller, Printer controller, LAN controller, disk drive controller.

d) **Industrial Instrumentation** – Process controller, DC motor controller, robotic systems, CNC machine controller, close loop engine controller, industrial moisture recorder and controller.

e) **Scientific** – digital storage system, CRT display controller, spectrum analyzer.

Were the embedded systems existing earlier ?

- Yes, We have been enjoying the grace of embedded system quite a long time. But they were not so popular because in those days most of the embedded systems were designed around a microprocessor unlike today's systems which were built around a microcontroller.

- As we know a microprocessor by itself do not possess any memory, ports etc. So everything must be connected externally by using peripherals like 8255, 8257, 8259 etc. So the embedded system designed using microprocessor was not only complicated in design but also large in size. At the same time the speed of microprocessor is also a limitation for high end applications.

What is inside an embedded system ?

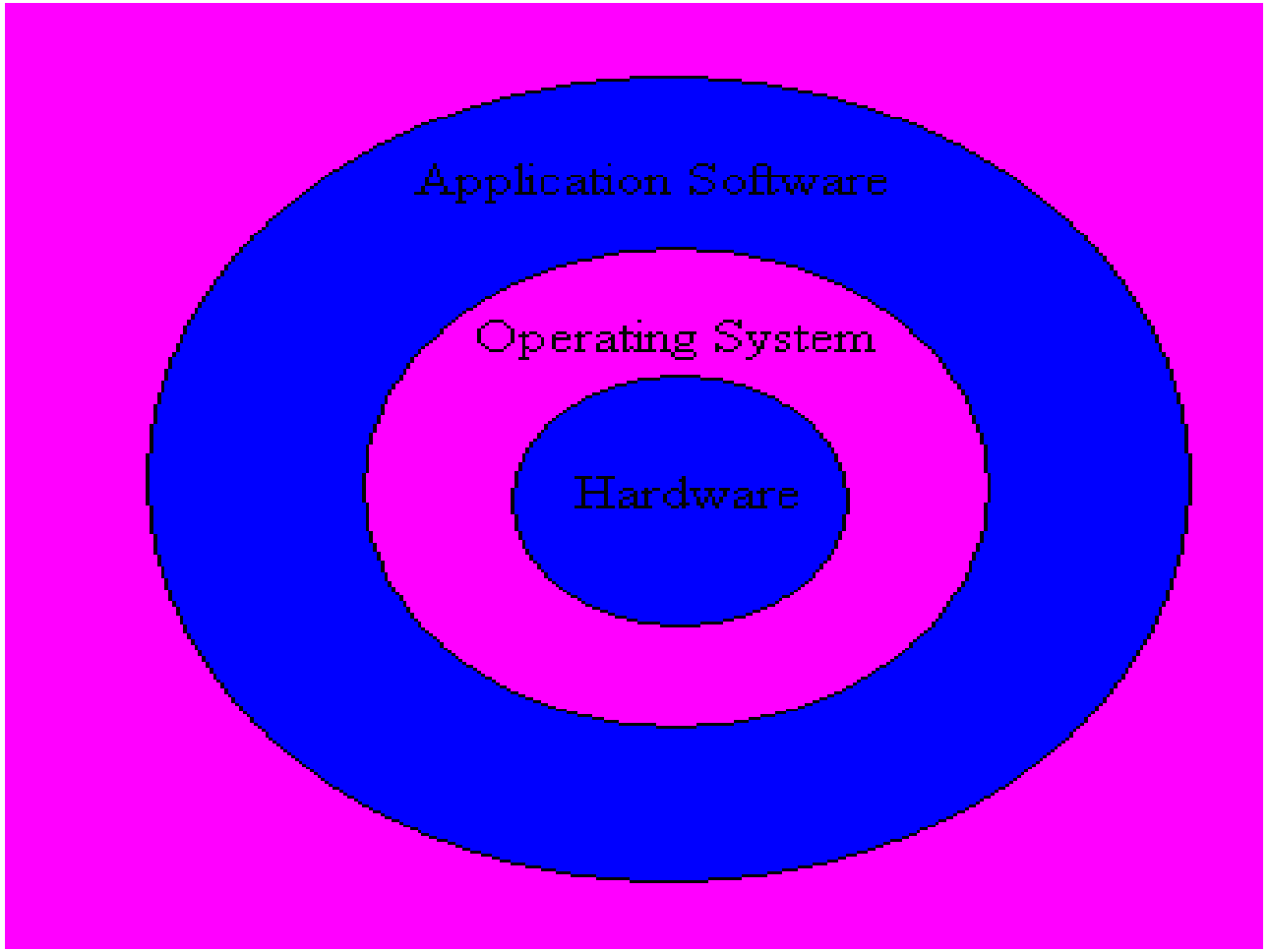
- Every embedded system consists of custom-built hardware built around a Central Processing Unit (CPU). This hardware also contains memory chips onto which the software is loaded. The software residing on the memory chip is also called the ‘firmware’.

- The operating system runs above the hardware, and the application software runs above the operating system. The same architecture is applicable to any computer including a desktop computer. However, there are significant differences. It is not compulsory to have an operating system in every embedded system.

- For small appliances such as remote control units, air- conditioners, toys etc., there is no need for an operating system and we can write only the software specific to that application. For applications involving complex processing, it is advisable to have an operating system.

- In such a case, you need to integrate the application software with the operating system and then transfer the entire software on to the memory chip. Once the software is transferred to the memory chip, the software will continue to run for a long time and you don't need to reload new software

.

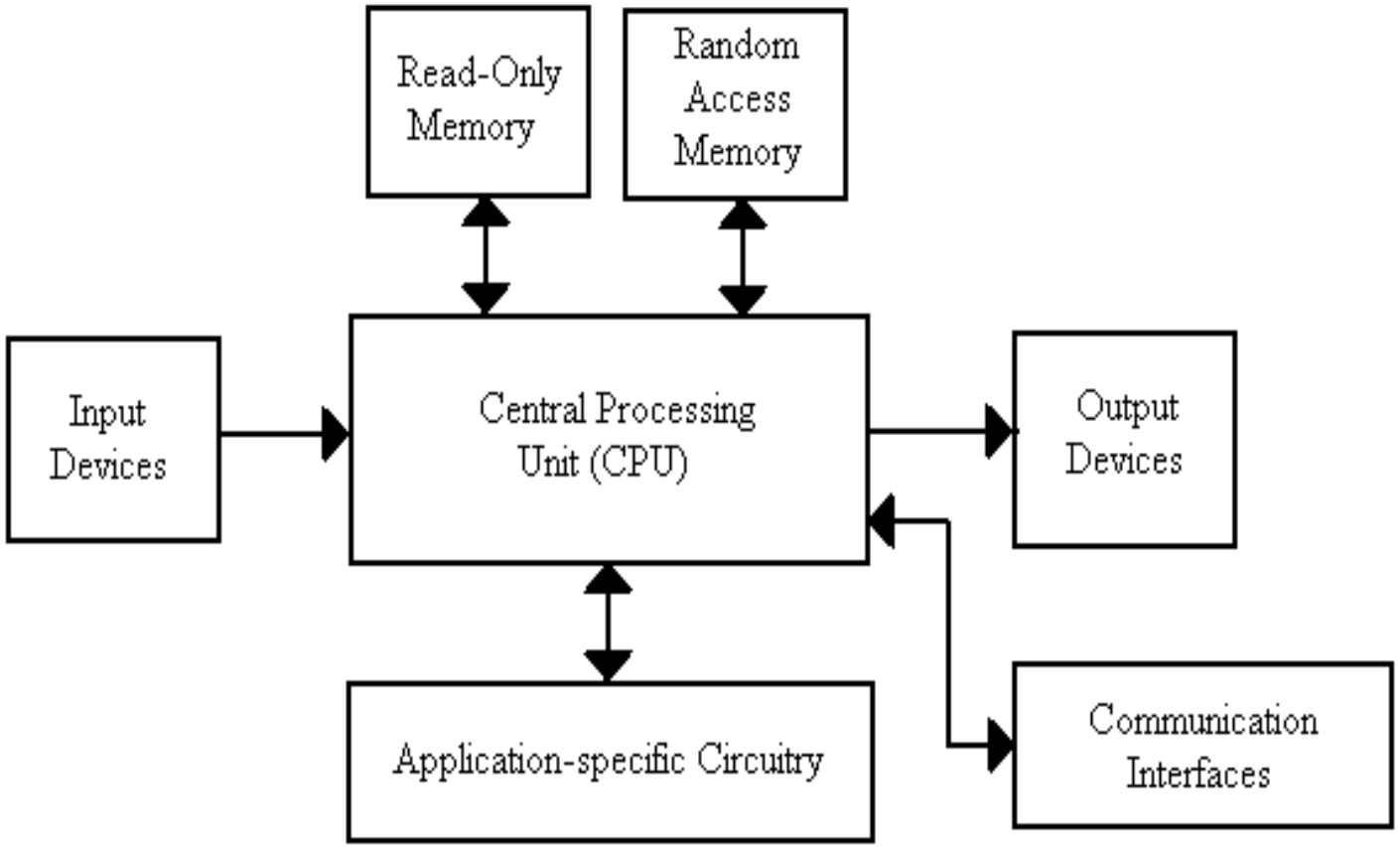


Layered architecture of an Embedded System

Now let us see the details of the various building blocks of the hardware of an embedded system.

- Central Processing Unit (CPU)
- Memory (Read only memory and Random access memory)
- Input Devices
- Output Devices
- Communication interfaces
- Application specific circuitry

Hardware architecture of an embedded system



Features of an embedded system

Embedded systems do a very specific task, they cannot be programmed to do different things.

- Embedded systems have very limited resources, particularly the memory. Generally, they do not have secondary storage devices such as the CDROM or the floppy disk.

- Embedded systems have to work against some deadlines. A specific job has to be completed within a specific time. In some embedded systems, called real-time systems, the deadlines are stringent. Missing a dead line may cause a catastrophe – loss of life or damage to property.

- Embedded systems are constrained for power, As many embedded systems operate through a battery, the power consumption has to be very low.

- Embedded systems need to be highly reliable. Once in a while, pressing ALT-CTRL-DEL is OK on your desktop, but you cannot afford to reset your embedded system.

- Some embedded systems have to operate in extreme environmental conditions such as very high temperatures and humidity.

- Embedded systems that address the consumer market (for example electronic toys) are very cost-effective. Even a reduction of Rs.10 is lot of cost saving, because thousands or millions systems may be sold.

- Unlike desktop computers in which the hardware platform is dominated by Intel and the operating system is dominated by Microsoft, there is a wide variety of processors and operating systems for the embedded systems. So, choosing the right platform is the most complex task .

Classification of Embedded Systems

Based on functionality and performance requirements,

embedded systems are classified as :

- Stand-alone Embedded Systems
- Real-time Embedded Systems
- Networked Information Appliances
- Mobile Devices

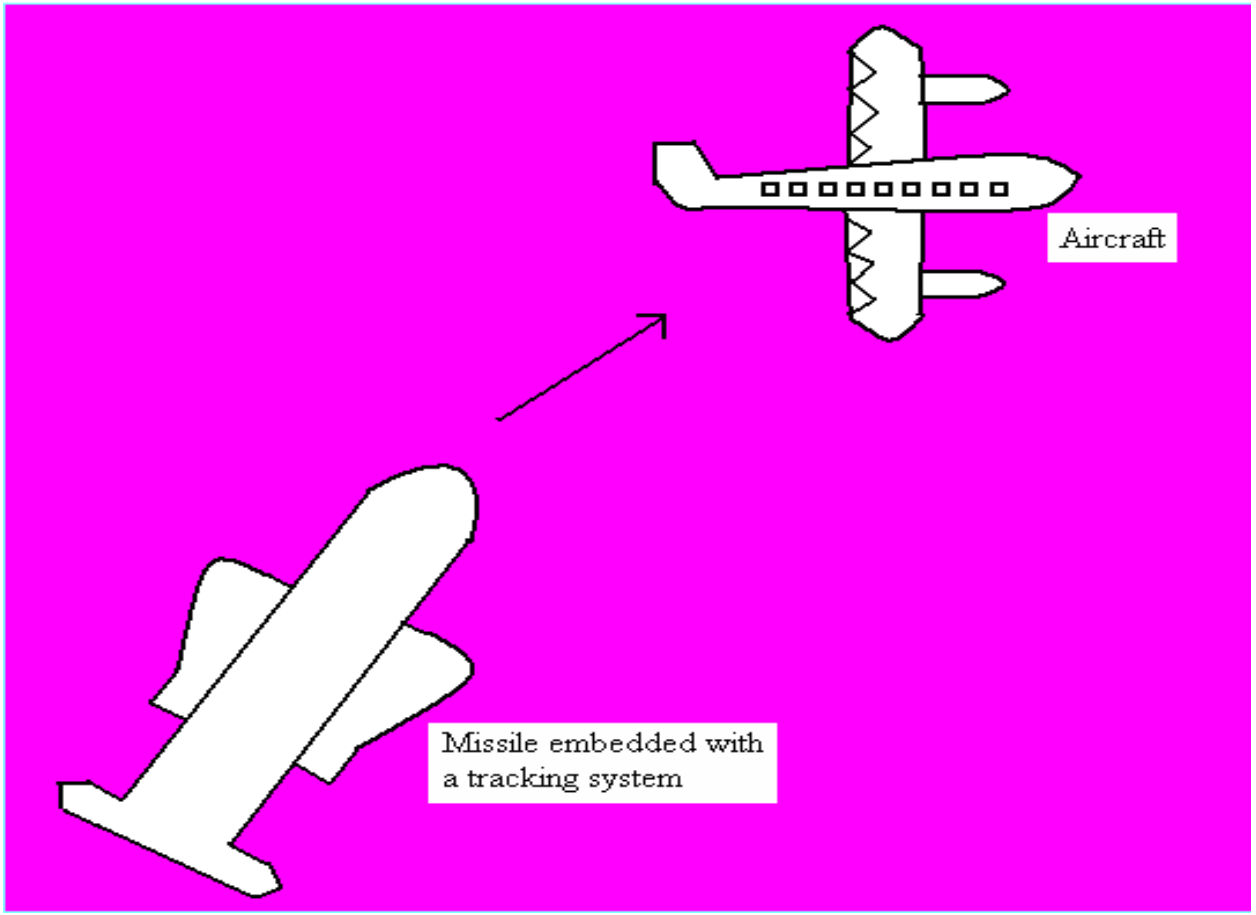
Stand-alone Embedded Systems

As the name implies, stand-alone systems work in stand-alone mode. They take inputs, process them and produce the desired output. The input can be electrical signals from transducers or commands from a human being such as the pressing of a button. The output can be electrical signals to drive another system, an LED display or LCD display for displaying of information to the users. Embedded systems used in process control, automobiles, consumer electronic items etc. fall into this category.

Real-time Systems

- Embedded systems in which some specific work has to be done in a specific time period are called real-time systems. For example, consider a system that has to open a valve within 30 milliseconds when the humidity crosses a particular threshold. If the valve is not opened within 30 milliseconds, a catastrophe may occur. Such systems with strict deadlines are called ***hard real-time*** systems.

- In some embedded systems, deadlines are imposed, but not adhering to them once in a while may not lead to a catastrophe. For example, consider a DVD player. Suppose, you give a command to the DVD player from a remote control, and there is a delay of a few milliseconds in executing that command. But, this delay won't lead to a serious implication. Such systems are called ***soft real-time*** systems .



Hard Real-Time Embedded System

Networked Information Appliances

- Embedded systems that are provided with network interfaces and accessed by networks such as Local Area Network or the Internet are called networked information appliances. Such embedded systems are connected to a network, typically a network running TCP/IP (Transmission Control Protocol/Internet Protocol) protocol suite, such as the Internet or a company's Intranet.

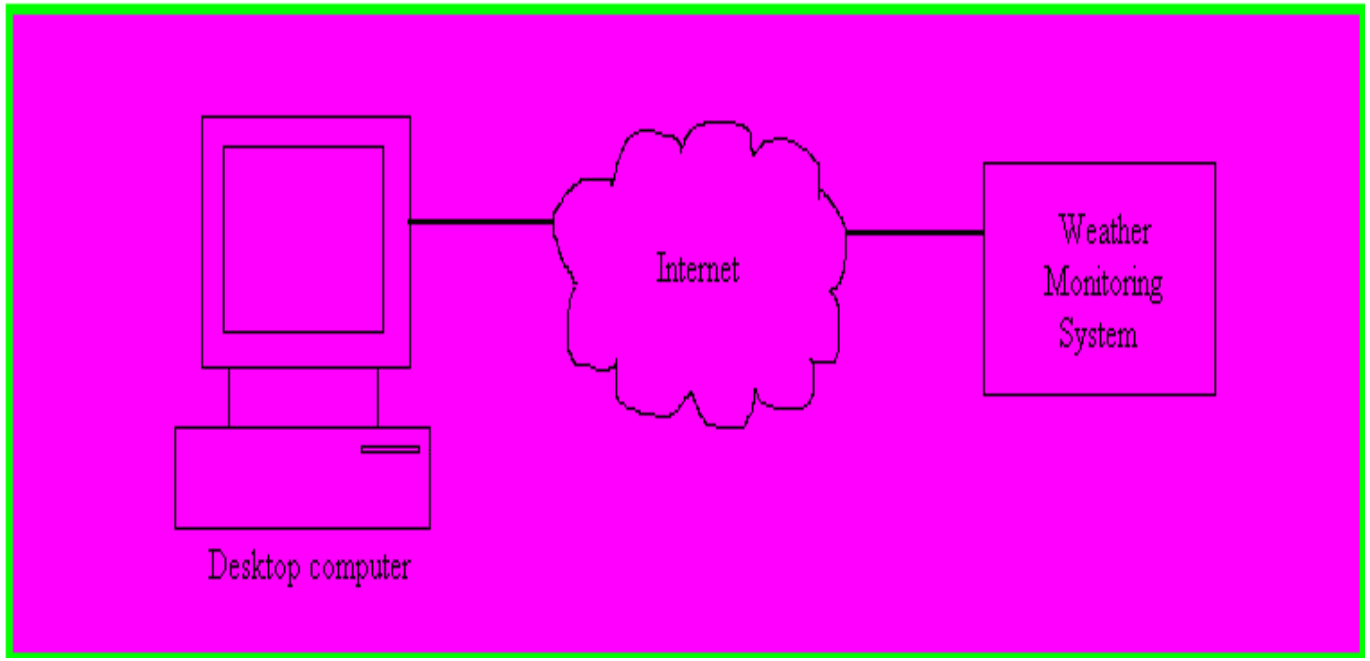
- These systems have emerged in recent years. These systems run the protocol TCP/IP stack and get connected through PPP or Ethernet to a network and communicate with other nodes in the network.

Here are some examples of such systems

- A networked process control system consists of a number of embedded systems connected as a local area network. Each embedded system can send real-time data to a central location from where the entire process control system can be monitored. The monitoring can be done using a web browser such as the Internet Explorer.

- A web camera can be connected to the Internet. The web camera can send pictures in real-time to any computer connected to the Internet. In such a case, the web camera has to run the HTTP server software in addition to the TCP/IP protocol stack.

- The door lock of your home can be a small embedded system with TCP/IP and HTTP server software running on it. When your children stand in front of the door lock after they return from school, the web camera in the door-lock will send an alert to your desktop over the Internet and then you can open the door-lock through a click of the mouse.



This image shows a weather monitoring system connected to the Internet. TCP/IP protocol suite and HTTP web server software will be running on this system. Any computer connected to the Internet can access this system to obtain real-time weather information.

The networked information appliances need to run the complete TCP/IP protocol stack including the application layer protocols. If the appliance has to provide information over the Internet, HTTP web server software also needs to run on the system.

Mobile Devices

Mobile devices such as mobile phones, Personal Digital Assistants (PDAs), smart phones etc. are a special category of embedded systems. Though the PDAs do many general purpose tasks, they need to be designed just like the 'conventional' embedded systems.

The limitations of the mobile devices – memory constraints, small size, lack of good user interfaces such as full fledged keyboard and display etc. are same as those found in the embedded systems discussed above. Hence, mobile devices are considered as embedded systems.

However, the PDAs are now capable of supporting general purpose application software such as word processors, games, etc.

Communication Interfaces

For embedded systems to interact with the external world, a number of communication interfaces are available. They are

- Serial Communication Interfaces (SCI): RS-232, RS-422, RS-485 etc
- Synchronous Serial Communication Interface: I2C, JTAG, SPI, SSC and ESSI
- Universal Serial Bus (USB)

- Networks:
  Ethernet, Controller Area Network, LonWorks, etc

- Timers:
  PLL(s), Capture/Compare and Time Processing Units

- Discrete IO:
  General Purpose Input/Output (GPIO)

- Analog to Digital/Digital to Analog (ADC/DAC)

DESIGN METRICS OF EMBEDDED SYSTEMS

- A Design Metric is a measurable feature of the system's performance, cost, time for implementation and safety etc.
- Most of these are conflicting requirements i.e. optimizing one shall not optimize the other.
- A cheaper processor may have a lousy performance as far as speed and throughput is concerned.

1. NRE cost (nonrecurring engineering cost)
It is one-time cost of designing the system. Once the system is designed, any number of units can be manufactured without incurring any additional design cost; hence the term nonrecurring.

2. Unit cost
The monetary cost of manufacturing each copy of the system, excluding NRE cost.

3. Size
The physical space required by the system, often measured in bytes for software, and gates or transistors for hardware.

CONTD..

4. Performance
The execution time of the system

5. Power Consumption
It is the amount of power consumed by the system, which may determine the lifetime of a battery, or the cooling requirements of the IC, since more power means more heat.

6. Flexibility
The ability to change the functionality of the system without incurring heavy NRE cost. Software is typically considered very flexible.

7. Time-to-prototype
The time needed to build a working version of the system, which may be bigger or more expensive than the final system implementation, but it can be used to verify the system's usefulness and correctness and to refine the system's functionality.

CONTD..

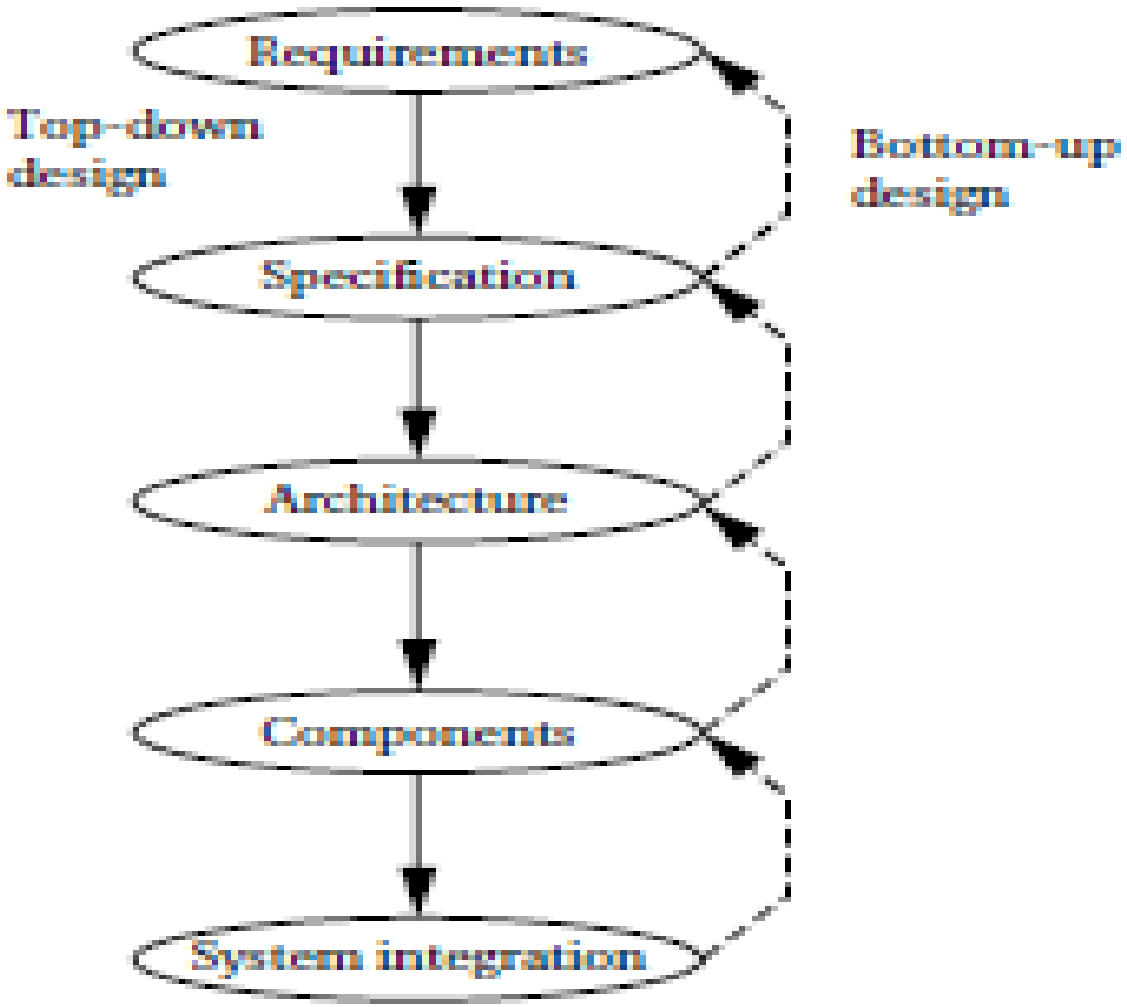
- **8. Time-to-market**
  The time required to develop a system to the point that it can be released and sold to customers. The main contributors are design time, manufacturing time, and testing time. This metric has become especially demanding in recent years. Introducing an embedded system to the marketplace early can make a big difference in the system's profitability.

  9. Maintainability
  It is the ability to modify the system after its initial release, especially by designers who did not originally design the system.

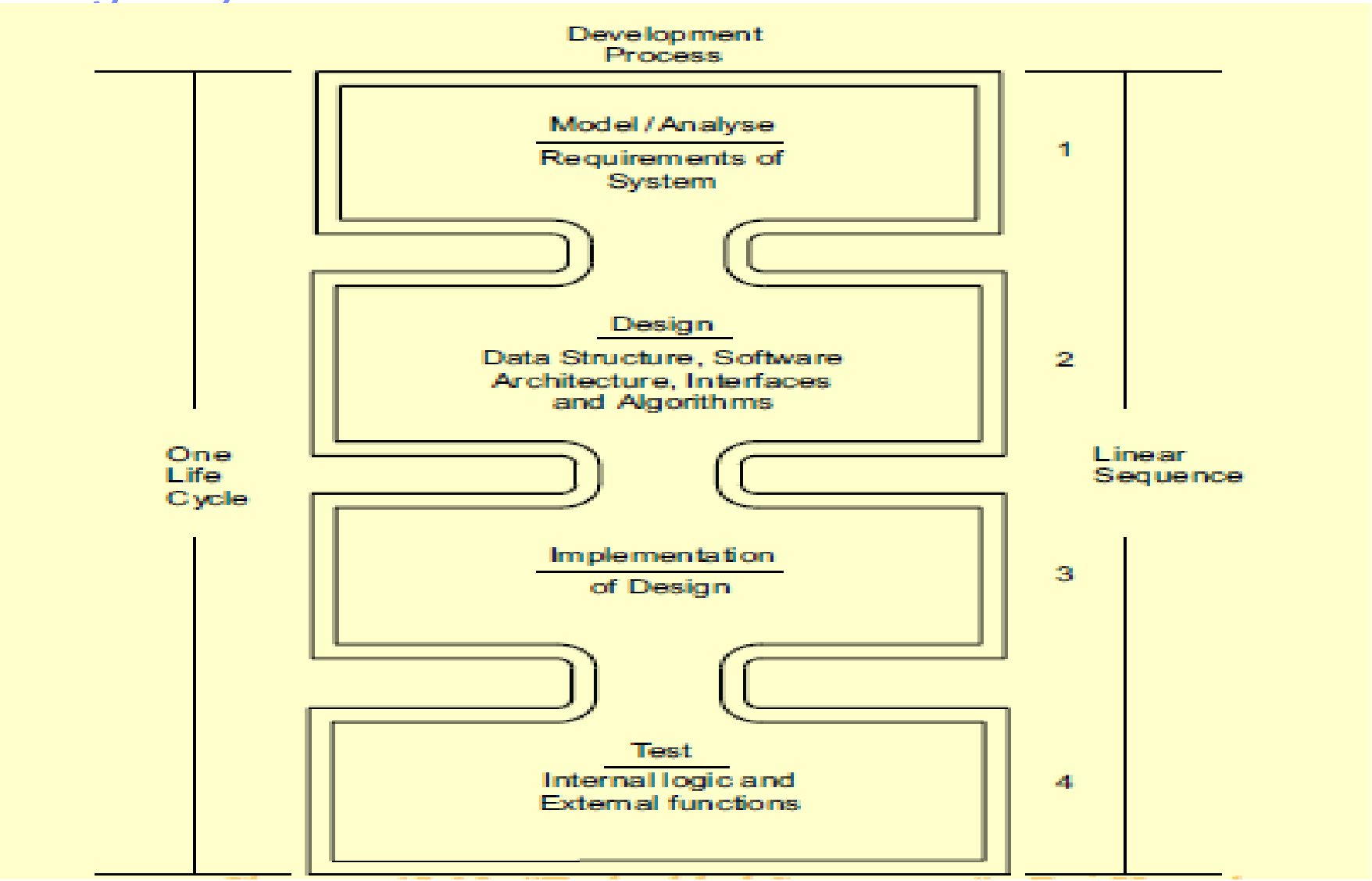
  10. Correctness
  This is the measure of the confidence that we have implemented the system's functionality correctly. We can check the functionality throughout the process of designing the system, and we can insert test circuitry to check that manufacturing was correct.

ES LIFE CYCLE:

1. Need/opportunity
2. Concept development
3. Manufacturing process design
4. Production
5. Deployment
6. Support/maintenance
7. Upgrades
8. Retirement/Disposals


Requirements
Top-down design
Bottom-up design
Specification
Architecture
Components
System integration

Software Design Cycle



Hardware/Software Codesign

A definition:

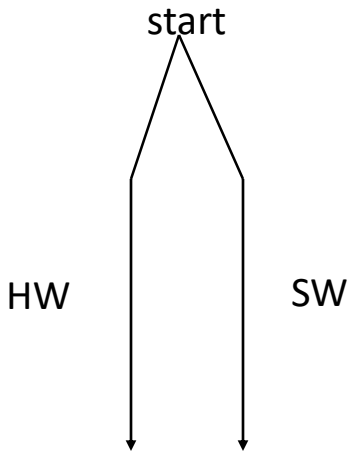
Meeting System level objectives by exploiting the synergism of hardware and software through their concurrent design

Why codesign?

- Reduce time to market
- Achieve better design
    - Explore alternative designs
    - Good design can be found by balancing the HW/SW
- To meet strict design constraint
    - power, size, timing, and performance trade-offs
    - safety and reliability
    - system on chip

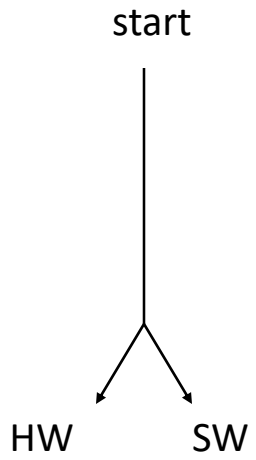
Concurrent design

Traditional design flow



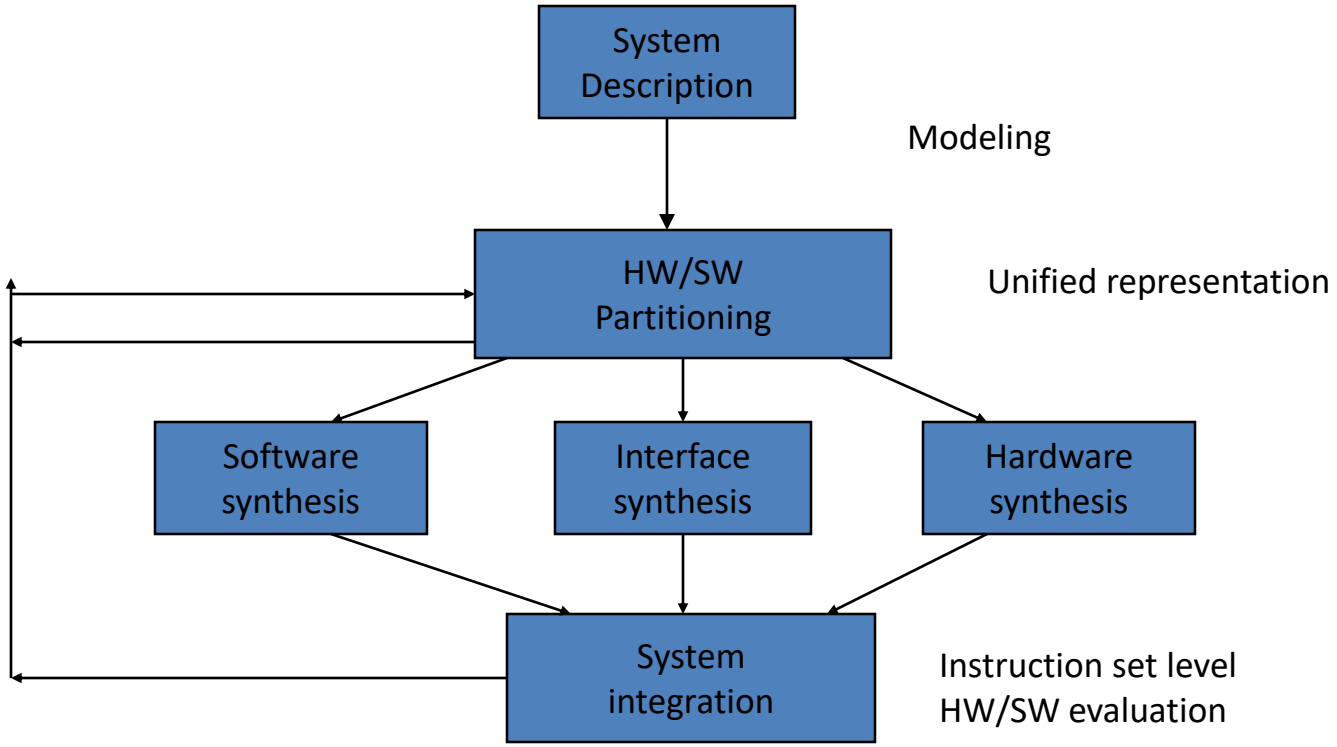
Designed by independent groups of experts

Concurrent (codesign) flow



Designed by Same group of experts with cooperation

Typical codesign process



HW/SW Co-design: Main Advantages

- To explore different design alternatives.
- To evaluate cost-performance trade-offs
- To reduce system design time

⇒ Reduction of product time-to-market and cost

- To improve product quality through design process optimization
- To support system-level specifications

⇒ To facilitate the reuse of hardware and software parts.

- To provide an integrated framework for the synthesis and validation of hardware and software components.