# Embedded System Design

10EC74

Part A / Unit 1

# Introduction to Embedded Systems

Presented by:

**VINAYAK JADHAV**

Assistant Professor, E&C Dept

**AGMRCET, Varur**

# Things to look for

- The definition of an Embedded system.
- The need for embedded applications.
- Some of the vocabulary of the field.
- Philosophy for the development of embedded applications.
- The major Hardware & Software components of an embedded system.
- The design & development process of an embedded system.
- Some of the basic architectures and how they evolved.

# 1.2 Philosophy

- The approach and views on solving engineering problems are different wrt different programmer, particularly as we learn and develop our skills and as the technology changes.
- Not every one feels the same way.
- Every one first drew their interest by looking at the outside view of an object(size, shape ,color and possible uses).
- Later, it was curiosity that drove them to learn more, to understand, to find out what was inside of something or what made it work.
- During early years told that ***necessity was the mother of invention***.
- Unfortunately, they lied. Necessity is not the mother of invention: *Laziness is*.

# 1.2 Philosophy

- With each new design, our first look should be from the outside.
  - What we are designing?
  - How will people use it?
    - What is its behavior?
  - What effect will it have on its operating environments?
    - What are the outputs?
    - What are the inputs?
  - How well do we have to do the job?
    - What are the constraints?
- i.e. first we look at the high-level details and then go on to the lower.
- As the technology advances, we are able to do more and more.
  - Today we can put several powerful computers in the space of a single vacuum tube occupied several years ago.
- Philosophy is fun, but now its time to work.

# 1.3 Embedded Systems

## 1.3.1 What is an embedded systems?

- What is an embedded systems?
  - Embedded systems are a combinations of hardware and software parts, as well as other components that will bring together into products such as cell phone, a music player, a network router or an aircraft guidance system.
  - They are a system within another system as shown in the fig.

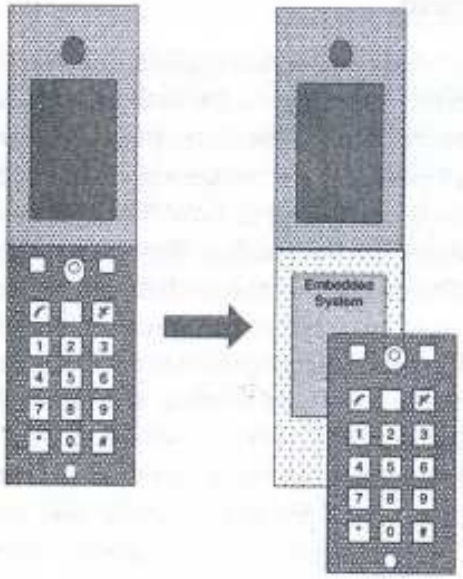



**Figure 0.1** A Simple Embedded System

# 1.3 Embedded Systems

- Embedded systems techniques allow us to make products that are smaller, faster, more reliable and cheaper.
- Allow us to bring features and capabilities to everyday things that could only be dreamed about few years ago.
- VLSI are the key component in enabling all of this to happen.
- With out VLSI embedded systems would not be feasible, and without embedded systems, VLSI would serve little purpose.
- Embedded systems present a variety of challenges as we bring the hardware, the software, and vagaries of the world outside of the microprocessor together.
  - Sending a rover across plains, conducting experiment son Mars, etc.
- Few years ago when microprocessors and PROMs first appeared as new tools, developing applications (firmware) was rather undemanding.
- Armed with a teletype, a simple assembler, and a host minicomputer ready for developing applications.

# 1.3 Embedded Systems

- Applications of several hundred lines of code transformed rather complex, discrete, logic designs into simple.
- Today we are designing embedded applications comprising thousands of lines of codes, multiple microprocessors, VLSI components and array of logics.
- Unlike the desktop PC, an embedded computer must interact with a wide variety of analog and digital devices.
- The skilled embedded developer must know and understand the operation of sensors and transducers, ADC, networks and their protocols and other processors, as well as traditional peripherals.
- As we make our system smaller and smaller, Solving problems arising from the signal coupling, noise, electromagnetic interference, or propagation delay is challenging indeed but necessary.

# 1.3 Embedded Systems

## 1.3.2 Building an Embedded System

- As we begin to study embedded applications, in addition to wide variety of components, we embed three basic kinds of computing engines into our systems: **microprocessors**, **microcomputers**, and **microcontrollers**.
- The microcomputer and other hardware components are connected via the **system bus**, which provides an interconnecting path for electrical signals to flow.
- The system bus is subdivided into three buses: segregated by the information they carry : **Address**, **Data** and **Control**.
- A typical embedded system comprises hardware and software and is designed and optimized to solve a specific problem efficiently.

# 1.3 Embedded Systems

## 1.3.2 Building an Embedded System

- The microprocessor controls the whole system, and ultimately the application, by executing a set of instructions called **firmware** that is stored in ROM in the memory subsystem.
- To run the application, when power is turned on, the microprocessor addresses a predefined location and **fetches**, **decodes**, and **executes** one instruction after another.
- The **instruction cycle** repeats forever, unlike a typical application program which eventually terminates.
- At the top-level, an embedded program is structured as an **infinite loop**, as illustrated in the code fragment below

```
While(1)
{
        Embedded program
}
```

# 1.3 Embedded Systems

## 1.3.2 Building an Embedded System

- The program never executes a return.
- The specific set of instructions that a microprocessors knows how to execute is called its **instruction set**.
  - Includes instructions that bring data in from the outside world, output signals to the external world, and provide a means to exchange data with the memory subsystem.
- The term **embedded system** refers to a system, that is enclosed or **embedded** in a larger system.
- Such system continually interacts with its surrounding environment as it monitors and controls some set of processes.
  - Ex : automobile, contains many microprocessors and microcontrollers for managing engine ignition, antilock braking system, transmission shift, power steering, etc

# 1.3 Embedded Systems

## 1.3.2 Building an Embedded System

- When operating in electrically harsh environment such as engine compartment of an automobile or cockpit of an aircraft, there is a high possibility that electromagnetic noise may cause the system to behave erratically.
- Similar behavior on desktop is easily remedied by resetting the system, but such option is not available in embedded application.
- To address this problem, we have a **watchdog timer**. The microprocessor must reset the timer periodically, if it fails to do so, the timer will reset the microprocessor.
- The signal from the watchdog timer comes in to the processor as a **non-maskable interrupt** (NMI), i.e. an interrupt that cannot be ignored.

# 1.3 Embedded Systems

## 1.3.2 Building an Embedded System

- Timer is often an essential constraint in an embedded application.
- For example stating a system is **real-time** demands that the system must respond to designated external or internal events within a specified time interval.
- Three kinds of real-time systems based on the urgency of meeting the required time constraint.
  - A system considered to be a **soft real-time** if failure to meet the time constraint results only in degraded performance.
  - If a time constraint is not met in **a hard real-time** system, the system is said to have failed. Such a failure may be seen as catastrophic if it can result in considerable risk to people, to the environment, or to the system being monitored and controlled.
  - A **firm real-time** system falls in between with a mix of the two kinds of tasks.

# 1.3 Embedded Systems

## 1.3.2 Building an Embedded System

- When an operating system is used in an embedded microcomputer, typically it is a real-time operating system(RTOS) .
- An RTOS is specifically designed and optimized to predictably handle the strict time constraint associated with events in a real-time context.
- Implementation of a microprocessor based embedded system combines the individual pieces into an integrated whole as shown in figure 0.3, which presents the architecture for a typical embedded system and identifies the minimal set of necessary components.

# 1.3 Embedded Systems

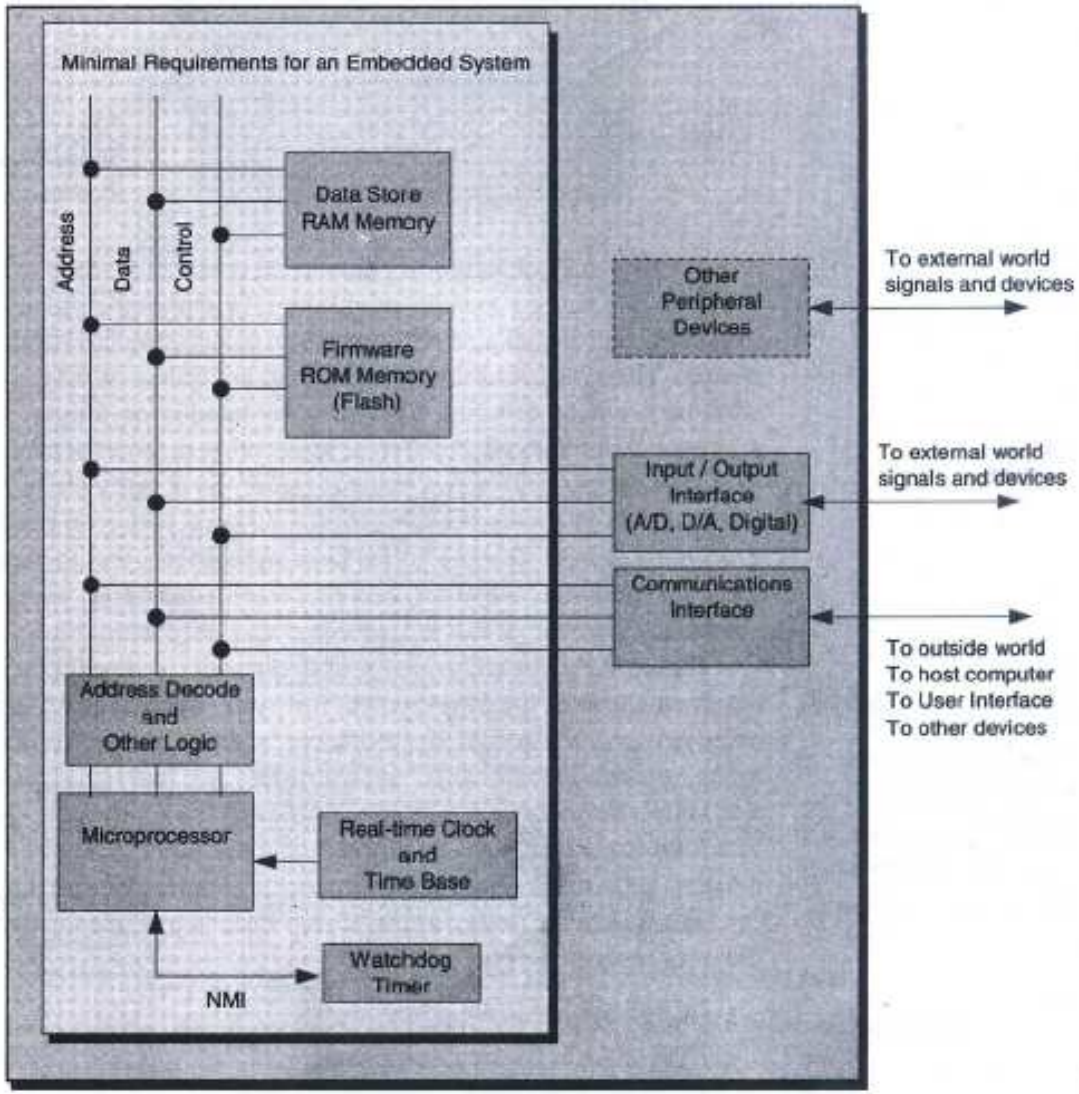



**Figure 0.3** A Microprocessor-Based Embedded System

# 1.4 The Embedded Design and Development Process



- Not too many years ago, we began the design of a new embedded application with some thought about the problem, wrapped some registers, logic and busses around the microprocessor, wrote a few lines of assembly language code, downloaded the assembled object file to the development environment, debugged it, and shipped it.

# 1.4 The Embedded Design and Development Process



- Such an approach worked great when all we had to be concerned about was the microprocessor, a handful of inputs and outputs, a few small-scale integrated (SSI) or medium-scale integrated (MSI) gate packs, and firmware that fit into a couple of PROMs as we see in the simple diagram in figure 0.4

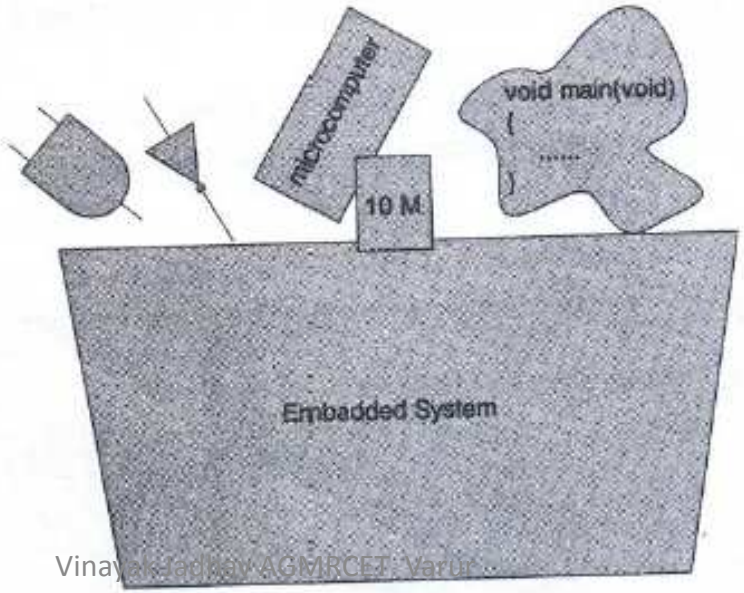



**Figure 0.4** Building an Embedded System

# 1.4 The Embedded Design and Development Process



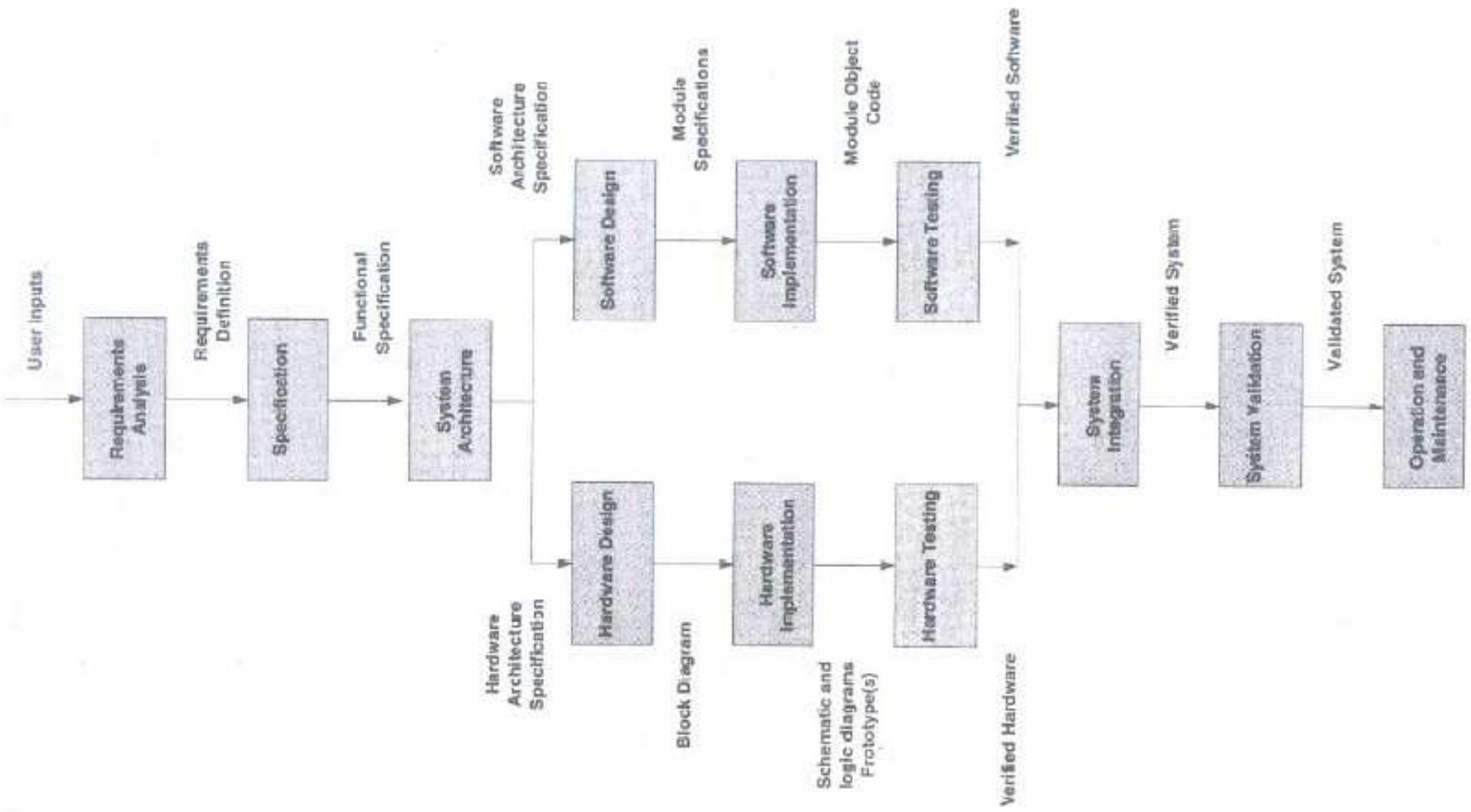
- Contemporary embedded applications tend to fall into two groups:
  - **Simple** (toaster, microwave, video game)
  - **Sophisticated** ( control a jet aircraft, control of a nuclear reactor)
- The systems in the second group are orders of magnitude more complex than any of those we use to build.
- Designing by hand is not feasible.
- Utilizing an Ad-hoc approach tailored to each different application is too expensive and too fraught with error.
- We can't simply wire few parts together, hack out a bit of software, and spend days or weeks trying to debug the collection.
- We need tools, we need formal methods.

# 1.4 The Embedded Design and Development Process



- We need tools to model our design, to perform simulations, and to simplify and interactively optimize the hardware, software and firmware.
- We need tools we can use to synthesize portions of that design either as a programmable logic device or a VLSI circuit.
- Figure 1.4 gives a high-level flow through the development process and identifies the major elements of the **development life cycle**.
- The hardware portion of the life cycle involves the design, development, and test of physical system architecture, packaging, PCBs, and ultimately the individual components.
- The software portion entails the tasks or algorithmic portion of the application. Such software may be written in a high-level language, assembler, or a mixture of the two.

# 1.4 The Embedded Design and Development Process



# 1.4 The Embedded Design and Development Process



- Work in assembly requires detailed knowledge of the microprocessor architecture and its register structure.
- The traditional design approach has been to traverse the two sides of the accompanying diagram separately. i.e.
  - Design the hardware components.
  - Design the software components.
  - Bring the two together.
  - Spend time testing and debugging the system.
- Contemporary methodologies favor the combined and simultaneous design of both the hardware and the software components, with the objective of meeting the system-level requirements through trade-offs between these two.
- The key point in such an approach are to specify and (iteratively) design and develop both aspects of the system concurrently with the goals of **increased productivity – reduced design cycle time** – and **improved product quality**.

# 1.4 The Embedded Design and Development Process



- Such an approach focuses on the major areas of the design process:
  - Ensuring a sound hardware and software specification and input to the process.
  - Formulating the architecture for the system to be designed.
  - Partitioning the hardware and software.
  - Providing an iterative approach to the design of hardware and software.
- **Major aspects in the development of the Embedded Applications**
  - Digital hardware and software architecture.
  - Formula design, development, and optimization process
  - Safety and reliability
  - Digital hardware and software / firmware design.
  - The interface to the physical world analog and digital signals
  - Debug, troubleshooting, and test of our design.

# 1.4 The Embedded Design and Development Process



- The contemporary creative design and development process begins with an abstracted notion of the system to be built and moves through an iterative series of transformations to the final product.
- Computer based tools are essential to that process.
- To be able to use such tools an engineer must understand how to develop accurate hardware and software models of the systems we intend to design and built.
- A comprehensive study of the design process and all related modeling tools and methodologies could easily take up several volumes – and does.
- Good system designers and designs proceed using minimum of five steps.

# 1.4 The Embedded Design and Development Process



- The five steps are:
  - Requirements definition
  - System specification
  - Functional design
  - Architectural design
  - Prototyping
- In today's world, contemporary design process must also take into consideration design reuse and intellectual property at every design stage.
- Through hierarchical modeling and functional decomposition, we became familiar and work with formal design methodologies.
- Verilog HDL – as hardware modeling and simulation tool.
- For software lifecycle, the Unified Modeling Language(UML) and structured methods are software modeling tools.

# 1.4 The Embedded Design and Development Process



- A good, solid and reliable design always begin with a firm foundation (figure 0.8).
- Without that, every thing we add later is fragile.

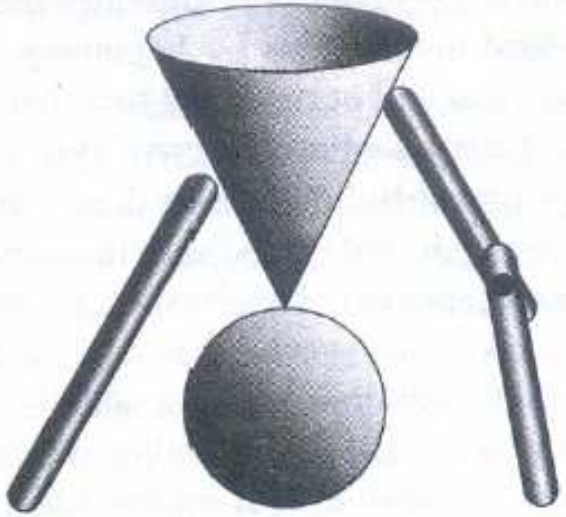


**Figure 0.8** The Essential Foundation

- Today, engineering student who want to work in embedded field must have a sound understanding of the basics of digital hardware and software architecture as well as familiar with complex systems.

# 1.4 The Embedded Design and Development Process



- We will open with the high-level structure and components coupled with an introduction to the von Neumann and Harvard architectures that comprises the hardware and computing core of an embedded application.
- The core is usually manifest as a microprocessor, microcomputer or microcontroller.
- At the opposite end of the system hierarchy, we will look at the bits, bytes, and volts as we study how the various and essential kinds of information are represented inside of a digital system.
- Next, we examine control and flow in such machines and discuss how each is manifest as a microprocessor, microcomputer or microcontroller.
- Analog and digital peripheral devices are incorporated to extend the basic architecture into the world of embedded application.

# 1.4 The Embedded Design and Development Process



- The programmer's view of the machine is introduced through a discussion of register transfer level(RTL) and instruction set architecture (ISA) models.
- The system designer's view brings the hardware and software together.
- With the high-level view in place, we proceed with a solid review of logic fundamentals and C language basics essential to developing robust embedded firmware.
- Today's embedded systems are used in many applications that can affect people's lives, result in significant environmental impact, or cost millions of dollars in the event of failure.
- One of our goals in the design of embedded applications is to provide the highest performance at the lowest cost while still delivering a safe and reliable system.

# 1.4 The Embedded Design and Development Process



- The design process does not stop with the first cut at a design; performance evaluation, like testing, must occur at every stage in the development.
- As a lead-in to the design cycle, we examine many of the considerations necessary for the execution of a safe, robust, and reliable design.
- Design is a process of translating a customer's requirements into a working system.
- Working from the specification, we partition the system into its major functions and then map those functions onto architectural structure.
- The applications itself is generally manifest in software and sits on top of the hardware infrastructure.

# 1.4 The Embedded Design and Development Process



- An essential component of any embedded system is the software/firmware.
- Early systems utilized an infinite loop to continuously cycle through the various jobs in the application. Such approach is still effective in a number of today's simpler designs.
- For more complex designs, more powerful methods are needed; such methods organize the required jobs into formal tasks or processes and carefully schedule when each is executed.
- The schedule becomes more complex as various constraints are added.
- Tasks will often need to share information: to cooperate and to communicate with each other with the world outside of the processor in well-controlled ways.

# 1.4 The Embedded Design and Development Process



- In the chapters ahead, we will introduce the fundamentals of the control and management of embedded applications by beginning with the simple polling and event-driven schemes.
- These ideas are first extended to non-preemptive and then to preemptive task based systems.
- Time based and reactive systems will lead to the development of simple scheduling algorithms. Building from these concepts, the real-time kernel will be introduced, and extended to include multitasking and multithreaded control.
- A comprehensive presentation of intertask communication methods and problems will lead to the need to coordinate and synchronize data exchange, the concept of critical sections, and the semaphore and monitor as tools for addressing such needs.

# 1.4 The Embedded Design and Development Process



- The concept of real-time operating system as an extension to the basic kernel concepts will provide the next-level of sophisticated and power.
- The notations of task priorities and scheduling criteria will lead to a formal discussion of several fundamental scheduling algorithms, which include first-come first-served, round robin, and rate monotonic.
- Once again, addressing the needs for safe and reliable systems, priority inversion, deadlocks, and starvation will present as potential problems in multitasking systems.
- Approaches for identifying and resolving such difficulties will be studied in depth.
- Embedded applications are intended to work with the physical world, sensing various analog or digital signals while controlling, manipulating, or responding to others.

# 1.4 The Embedded Design and Development Process



- The study of the interface to the external world extends the i/o portion of the von Neumann machine (figure 0.9) with a detailed study of busses, their constituents, and their timing considerations.

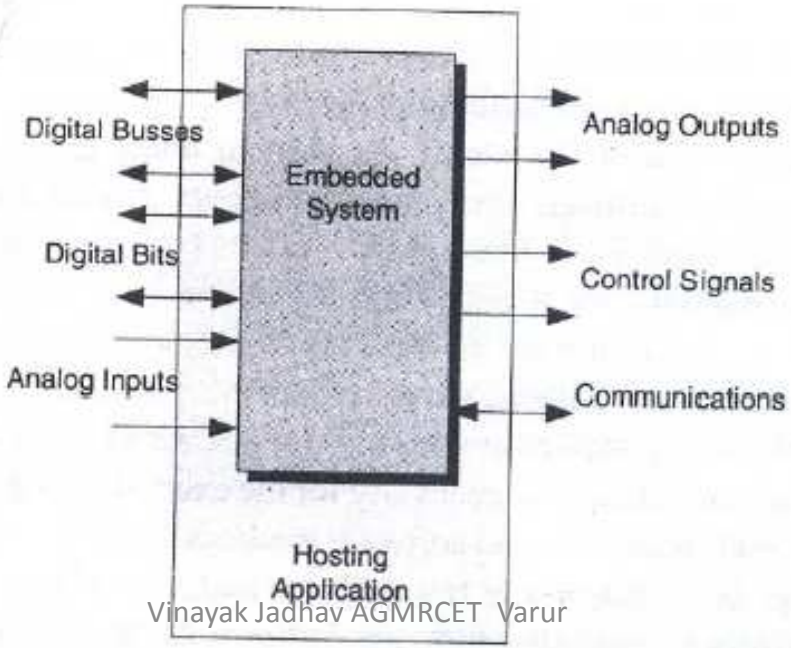



**Figure 0.9** Interfacing to the Outside World

# 1.4 The Embedded Design and Development Process



- Local exchange is extended to include distributed and remote systems.
- The study of basic transaction management, consistency models, idempotent systems, and error management continues the thread of designing safe and reliable systems.
- A substantial presentation of the development of interfaces for a variety of the more familiar serial and parallel hardware devices provides a solid set of tools on which to build.
- A design is not complete when the first prototype is delivered; the system must also be tested once the design is released to manufacturing.
- We test at different times for different reasons; testing and debugging, like modeling, must be interwoven with the development process.

# 1.4 The Embedded Design and Development Process



- Let us look back to the good old days , once we built the hardware and firmware for our system, confirming that it worked was generally a straightforward task.
- A handful of switches, hardwired signals, a good emulator, an oscilloscope, and a logic analyzer, peppered with a little ingenuity, would generally suffice as a comprehensive test system.
- Today that is not even close. We are now working with dozens of simultaneously changing high speed signals, on dozens of components that may exhibit failures based on infrequently occurring patterns in the data.
- Further complicating the problem is the simple fact that we have little visibility into the processors, VLSI circuits, or array logics with which we're working.

# 1.4 The Embedded Design and Development Process



- Complicating the task even further, we must now deal with state-of the-art operating systems, timing constraints measured in nano or pico seconds, multiple processors, and systems scattered all over the world.
- As with modeling, the literature, tools, and techniques in the field of test are extensive.
- Nonetheless, having a solid grounding in test and basic test techniques is essential, yet this knowledge is missing from the toolkit of most contemporary undergraduates and many working engineers.
- Tool chest is an understanding of basic debugging techniques, in particular, when confronted with the complexity of today's systems.

# 1.4 The Embedded Design and Development Process



- In today's world, we continuously strive to make our designs faster, smaller, and cheaper, and to consume less power.
- As engineers, we always believe that we can make our design just a little better.

Text & Reference Books

**Embedded Systems** – A contemporary Design Tool,
James K. Peckol, John Weily India Pvt. Ltd, 2008.

1. **Embedded Systems: Architecture and Programming,** Raj Kamal, TMH. 2008.
2. **Embedded Systems Architecture** – A Comprehensive Guide for Engineers and Programmers, Tammy Noergaard, Elsevier Publication, 2005.
3. **Programming for Embedded Systems**, Dreamtech Software Team, John Wiley India Pvt. Ltd, 2008.