UNIT-1

Introduction to Embedded System: Introducing Embedded Systems, Philosophy, Embedded Systems, Embedded Design and Development Process.

INTRODUCING EMBEDDED SYSTEMS

Embedded – fixed, surrounded, integrated and implanted.

Systems- body which has inputs and outputs through inputs it takes receives the data and manipulate, process or transform the data and gives the output or control the environment.

EMBEDDED SYSTEMS

• Computing systems – for a different purpose

• Nearly any computing system other than a desktop computer

APPLICATIONS OF EMBEDDED SYSTEMS

• **Consumer electronics**: Camera, music players, TVs, DVD players, microwave ovens, washing machines, refrigerators and remote control.

• **Household appliances/home security systems**: Air conditioners, intruders, and fire alarm systems.

• **Automobile controls**: Anti-lock braking system, engine and transmission control, door and wiper control.

• **Handheld devices**: Mobile phones, PDA, MP3 players, dig cams.

• **Medical equipments**: Scanners, ECG, and EEG units, testing and monitoring equipments

• **Banking:** ATM, currency counters.

• **Computer peripherals**: Printers, scanners, webcams, etc.

• **Networking:** Routers, switches, hubs.

• **Factories**: control, automation, instrumentation and alarm systems.

• **Aviation**: Airplane controls, guidance and instrumentation systems.

• **Military**: Control and monitoring of military equipments.

• **Robotics**:

• **Toys**

CHARACTERISTICS OF EMBEDDED SYSTEMS

1. Single- functioned

    • Executes a specific program repeatedly (eg:Pager).

2. Tightly constrained

    • Constraints are very tight (cost,size,performance and power).

3. Reactive and real time

    • Must continually react to changes in the system's environment.

    • Must compute certain results in real time without delay.

Design Challenge-Optimizing Design Metrics

As we know the embedded systems are tightly constrained, it is a difficult challenge to construct a system which optimizes numerous design metrics and satisfies the functionality.

Common Design Metrics

Design Metrics is a measure of an implementation's feature and the commonly used design metrics are

1. *NRE cost (Nonrecurring Engineering cost):* it represents the monetary cost that will not be happening again and again mainly the design cost.

2. *Unit cost*: represents the monetary cost spent on manufacturing each copy excluding NRE cost.

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

4. *Performance*: the execution time of the system or throughput of the system.

5. *Power*: the amount of power consumed by the system, which determines the life time of the battery.

6. *Flexibility*: the ability to change the functionality of the system without incurring heavy NRE cost.

7. *tim*e-to-prototype: the time required to build a working model of the system, to refine the system's functionality.

8. *Time-to-Market*: The time required to develop the system to the point that it can be released and sold to customers. The main contributors are design time, manufacturing time and testing time.

9. *Maintainability*: the ability to modify the product design after manufacturing, especially by the designers who did not do it originally.

10. *Correctness*: Testing and verification.

11. *Safety*: The probability that the system will not cause harm.

PHILOSOPHY

• Philosophy - Study of creation of theories about basic things

Once I have taken my students for an excursion and when we arrived to a sea shore, some students started picking stones, and some started picking star fishes, some were collecting shells, and some were collecting sticks.

When I asked them about what they were doing? They started telling different explanation. Those who picked up the stones said, it was in a round shape I just took it to make it to skip over a water for five times, somebody said with the same stone I skipped over a water for ten times, those were picked up the star fishes said, it looks pretty and different than any other fishes, I just want to touch and feel it. Those who were collecting shells said I may use it for decorating house, those who were collecting sticks said I am tired and need a support to walk.

In all these cases

• All of them had seen the objects from outside such as size, shape, colour, and possible uses.

• **The curiosity** (is a desire to know about something) that driven them to explore their uses.

• **Comparison of their work** with others to do more work.

Next thing that help us to understand the philosophy is the well known saying "Necessity was the mother of invention" probably as you read you may think that is correct but actually **laziness is**. Because when some one invented bicycle they might have been lazy about pedaling the cycle. In the early days semaphores were used for communication in this process two tanks filled with water were used at sender and receiver places with the float in it. And in the different regions of the float some information of were written on both the floats. When sender wants to send a message he will use the torch light to indicate he is going to send message first he will open the tap and as the water is drained and receiver also will open the tap. Once the float reaches the particular message sender will bring the torchlight down that indicates that both have to close the tap and receiver will read the information. What a complicated process! Somebody might have felt this was a complicated process that led them to invent telephony.

Whenever we want to design a system to gather the required information first **we should look from the outside** to find out

• What is its behaviour? – What are we designing? How people will use it?

• What are the outputs? – What effect will it have on its operating environment?

• What are the inputs? – What will be the effect of its operating environment?

• What are the constraints? – How well do we have to do the job?

So when you have a problem, we have many **text books and reference materials** available in the library and in the internet, by looking into these we can find out different solutions, and also we need to understand the fact the text book solutions may be different from your solutions so use your knowledge and think beyond a boundaries to do the work quickly and efficiently.

Also understand the **current trends in the design field**, keeping track of the behaviour of a vacuum tube offers minimum challenge, but orchestration of the information flow and managing the computation schedules offer more challenges.

To address these complexities **we need tools** to attack complexities and tools to do more job quickly and efficiently.

EMBEDDED SYSTEMS

• Embedded system is not a stand alone field because we need tools, techniques and knowledge from other fields.

• Embedded system is a combination of hardware and software parts

• Embedded system is a system within another system see figure 1.1

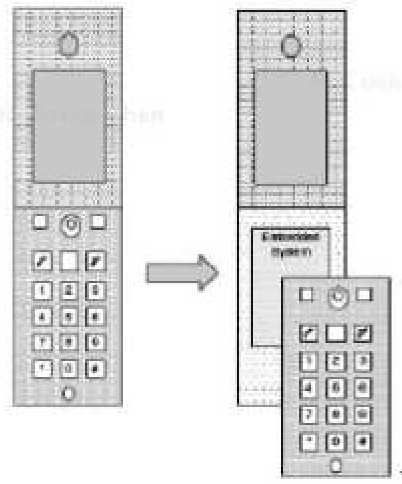

Figure 1.1: A simple embedded system

When we design application program people should be able to see it, like it, and to buy it, if this fails entire work goes in vain. Similarly when we design an embedded system it should be reliable, quiet and efficient.

An embedded system design is basically **unified design that means it is a combination of software and hardware components**. In the **earlier days** for the embedded system design only microprocessor and PROM was used, and for the operation firmware (program stored on a chip), simple assembler and host microcomputer was used. In **today's embedded system** design more than thousands lines of codes are needed, and also it uses multiple microprocessors, VLSI components and array logics, so the pressing need is to develop and learn new tools Then to do the design process **skilled embedded developer** is needed with the following skills.

• Operation of sensor

• Transducers

• Operation of A2D and D2A

• Protocols and motors

• Other processors

• Traditional peripherals

• Importance of the laws such as Faraday’s, Maxwell …..

BUILDING AN EMBEDDED SYSTEM

To build an embedded system three basic kinds of computing engines may be used

1. Microprocessor – takes the input and process the data

2. Microcomputer- A **microcomputer** is a small, relatively inexpensive computer with a microprocessor as its central processing unit (CPU). It includes a microprocessor, memory, and input/output (I/O) facilities.

3. Microcontroller – takes the input, process the data and control the

Environments that is connected to

To connect the computing engines to the circuits internally and externally **three types of buses** are used namely Address, Control and Data.

For the operation of the computing engine it uses the **firmware**, which is basically a set of instructions and using the sequence fetch, decode and execute. The main interesting thing about the program execution for the embedded system project will be in an infinite loop such as follows.

```
While (1)
{
…….
……
}
```

The important features of the computation engine are **watchdog timer and interrupts**, a watchdog timer will generate a signal for the computation engine after some specified time interval. And the interrupt is mainly used to attract the computation engine attention when it busy in executing some other program.

Embedded systems are called as **real time systems**; this system will give a response within a specified time interval. There are three types of real time systems

1. Hard real time system - always it will meet the deadlines.

2. Soft real time system – it takes best effort to meet the deadlines.

3. Firm real time system – it is the mix of both.

Since the embedded system is used in a real time environment the operating system that is used here is called **real time operating system RTOS**.

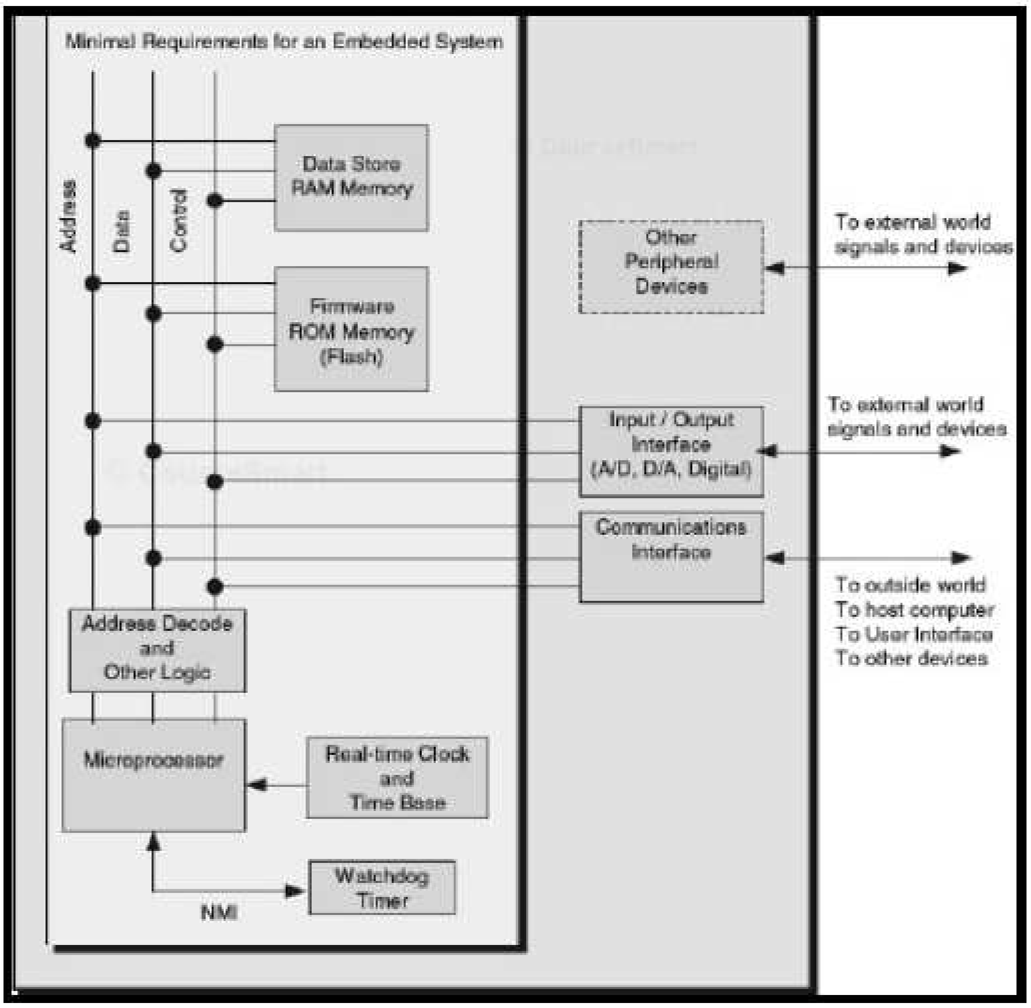
Minimal Requirements for an Embedded System
Address
Data
Control
Data Store RAM Memory
Firmware ROM Memory (Flash)
Other Peripheral Devices
To external world signals and devices
Input / Output Interface (A/D, D/A, Digital)
To external world signals and devices
Communications Interface
To outside world
To host computer
To User Interface
To other devices
Address Decode and Other Logic
Microprocessor
Real-time Clock and Time Base
Watchdog Timer
NMI

Figure1. 2: Microprocessor based embedded system

The embedded system design and development process

Design – process of translating customer's requirements into a working system.

In this section we discuss design and development in the following aspects.

1. Requirements of an embedded system design

2. The designer

3. Tools

4. Testing and debugging

We need to understand that **embedded systems are used in different environments**, near the engine, inside the air conditioner, at high altitude, below sea level by sensing analog and digital signal while processing or monitoring or controlling others. See figure 1.3.

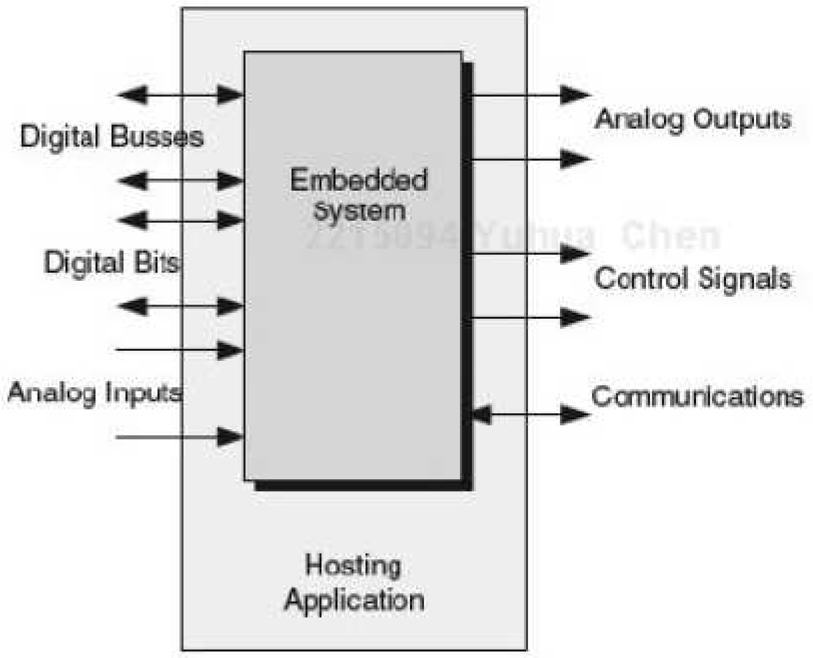

Figure1. 3: interfacing to the outside world

Now days embedded applications has an **impact** in people lives therefore if something goes wrong with the design, it result in significant environmental impact, loss of cost of millions of dollars.

To **understand trends in the design**, in earlier days design a small firmware stored in PROM, and microprocessor with small assemblers are used as shown in figure 1.4. But the current trend is confronting the designers with the need for robust, reliable and well designed systems Figure 1.4:

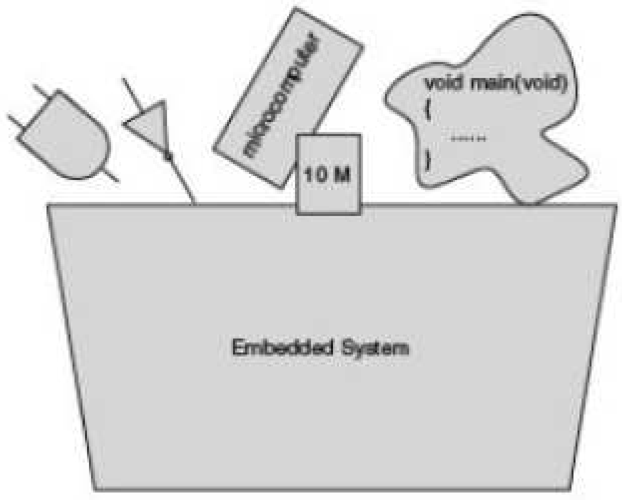

Figure 1.4:Building an embedded system

As we start to design the embedded system then comes the **two types of design methodologies traditional and contemporary** (things are modern and relate to the present time) design. The simple design can be done easily by writing a firmware and executing whereas the contemporary design increases the complexity, requires more cost, more prone to error and time consuming debugging process.

The design flow that is used in traditional design is given in the figure 1.5 is the embedded system life cycle. In this **hardware portion** deals with the design, development, test of physical system area, packaging and PCBs. and the **software portion** deals with algorithms, high level language and assembler.

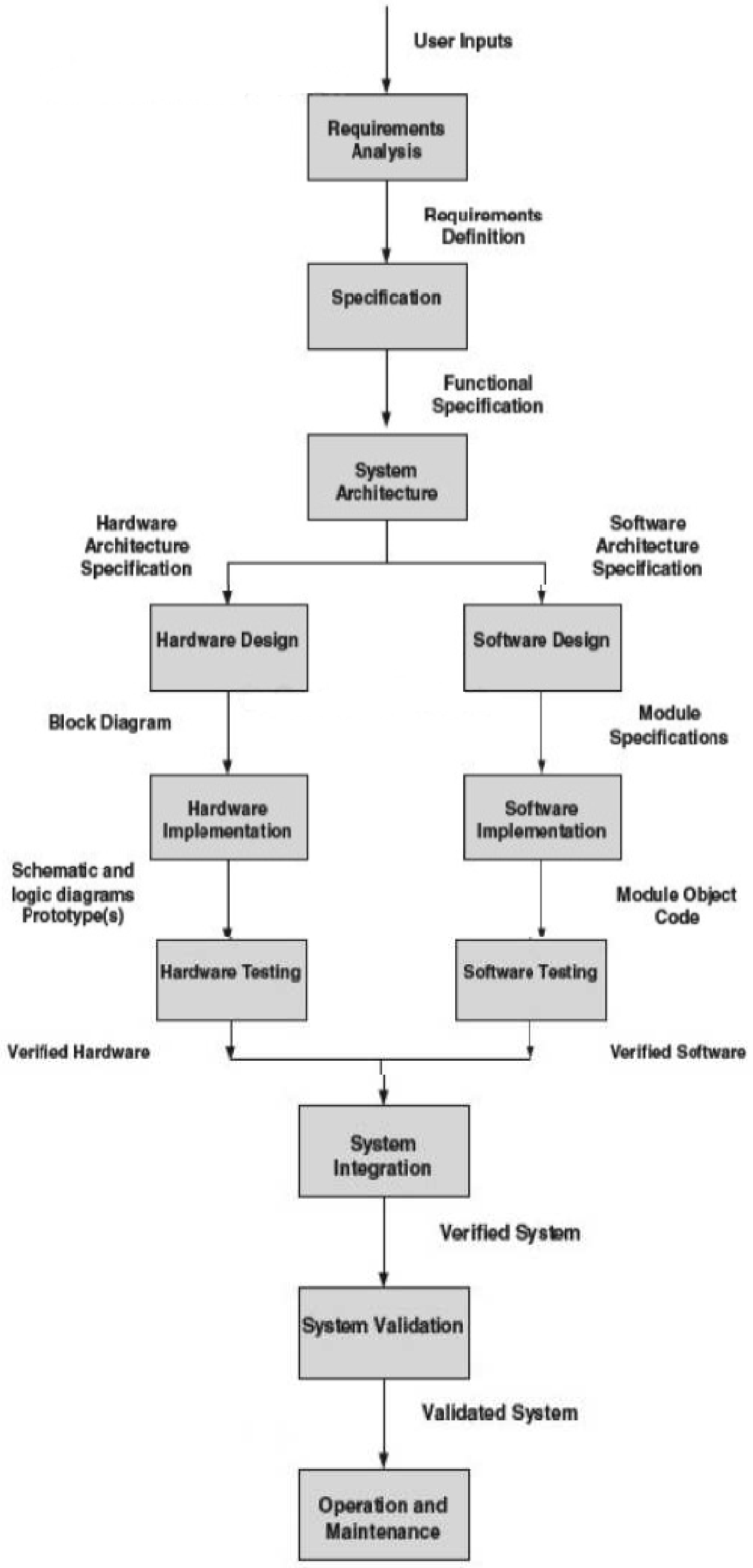

Figure 1.5: The embedded system life cycle

The traditional design approach involves the following steps

• Design of hardware components

• Design of software components

• Bring the two together

• Spend time on testing and debugging the system

In the contemporary method the design and **hardware and software will be combined or simultaneous** done with some trade-offs (loosing one quantity to gain a quality of other) to increase productivity, reduced design cycle time, and improved product quality.

This 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

The contemporary design process must consider the design reuse and intellectual property at every phase of the design. The Verilog HDL can be used as hardware modeling and simulation tool. The Unified Modeling Language (UML) and structured methods can be used as software modeling tools.

The designer plays a vital role in the entire process; he or she must have a firm foundation on the following things as shown in figure 1.6.

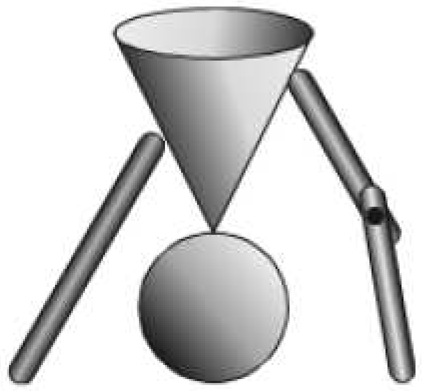

Figure 1.6: the essential foundation

• Understanding of the basics of digital hardware and software architecture

• Understanding of Von Neumann and Harvard architecture

• Examining control and data flow

• Operation of analog and digital peripherals

• Programmer’s view such as RTL (Register Transfer Level) and ISA (Instruction Set Architecture).

• Systems designer’s view about bringing the software and hardware together logic.

• C Language basics

Now we will introduce **the fundamentals of the control and management** of embedded applications, the following things are mainly deals with the control and management of embedded application.

• Event driven

• Polling

• Real Time Kernel – RTK

• Inter task communication methods

• Critical sections

• Semaphores

Among these RTOS mainly deals with the following things

• Task priorities

• Scheduling fundamentals

- FCFS
- Round – Robin
- Rate monotonic
- Priority inversion
- Dead locks
- Starvation

We learn that good system designers and designs proceed using a minimum of five steps. These steps are listed below.

1. Requirements definition

2. System specification

3. Functional design

4. Architectural design

5. Prototyping

A design is not complete when the first prototype is delivered, the system must also be tested. However the need for testing does not start with the release to manufacturing we test at different times for different reasons. Testing and debugging must be interwoven with development process.

In the earlier days once the hardware was built the hardware and firmware for our system, confirming that it worked was generally a straight forward task. A handful of switched, hardwired signals, a good emulator, an oscilloscope and a logic analyzer would generally enough as a comprehensive test system. Today that is not even close, we are now working with dozens of simultaneously changing high speed signals, or 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 are working. Complicating the tasks 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 we conclude this topic the pressing demand in the society is skilful testing and debug engineers not a magic answers such as place like this, only in the afternoon, near the window, there are lot of resources available over internet and library, equip yourself, the door of the embedded world is wide open to you despite of technological changes are you ready to do design faster, smaller, cheaper and low power device.