Embedded & Real Time Operating Systems

Unit-I Notes

Table of Contents

Unit 1

Introduction to Embedded Systems

An embedded system is a combination of hardware and software with some attached peripherals to perform a specific task or a narrow range of tasks with restricted resources. It is an electronic system that is not directly programmed by the user, unlike a personal computer. An embedded system is a device that incorporates a computer within its implementation, primarily as a means to simplify the system design, and to provide flexibility; and the user of the device is not even aware that a computer is present. It is a microcontroller-based, software-driven, reliable, real time control system; Autonomous or human or network interactive, operating on diverse physical variables and in diverse environments, and sold in a competitive and cost conscious market. Generally, an embedded system is a subsystem in larger system and it is application specific. The generic block diagram of an embedded system is shown in Figure 1. Every embedded system consists of certain input devices such as: key boards, switches, sensors, actuators; output devices such as: displays, buzzers, sensors; processor along with a control program embedded in the off-chip or on-chip memory, and a real time operating system (RTOS).

Definition: An Electronic/Electro-mechanical system which is designed to perform a specific function and is a combination of both hardware and firmware (Software). E.g. Electronic Toys, Mobile Handsets, Washing Machines, Air Conditioners, Automotive Control Units, Set Top Box, DVD Player etc.

Every embedded system is unique, and the hardware as well as the firmware is highly specialised to the application domain. Embedded systems are becoming an inevitable part of any product or equipment in all fields including household appliances, telecommunications, medical equipment, industrial control, consumer products, etc.

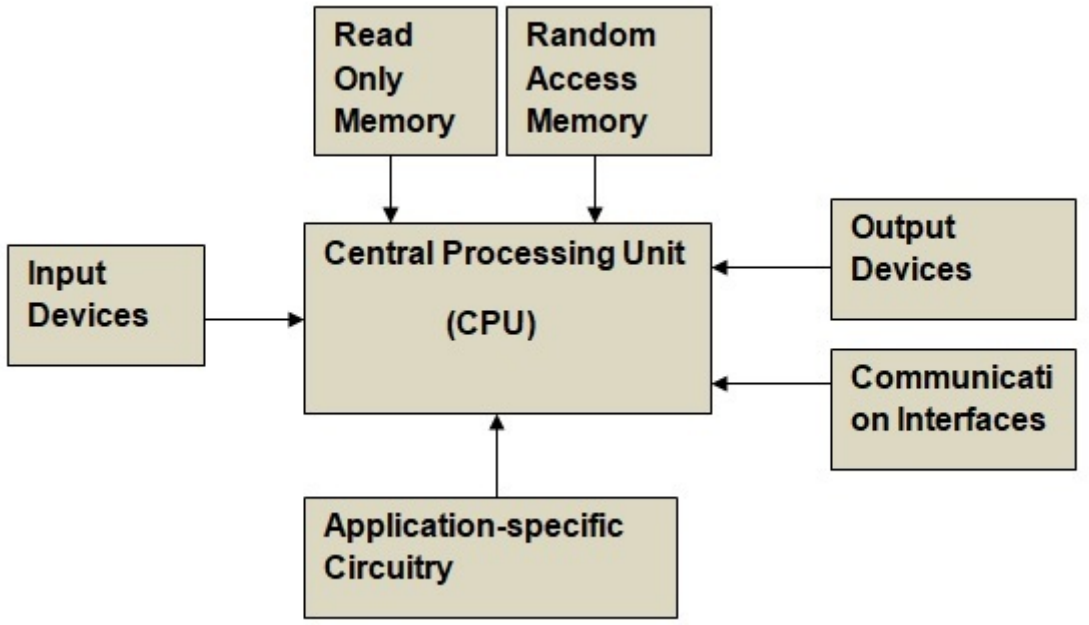

Figure 1: Basic blocks in an Embedded system

1.1 Embedded Systems vs. General Computing Systems

The embedded computing requirements demand 'something special' in terms of response to stimuli, meeting the computational deadlines, power efficiency, limited memory availability, etc. Let's take the case of your personal computer, which may be either a desktop PC or a laptop PC or a tablet PC. It is built around a general purpose processor like an Intel or an AMD. You can write or purchase a multitude of applications for your PC and can use your PC for running a large number of applications. Now, let us think about the DVD player you use for playing DVD movies. Is it possible for you to change the operating system of your DVD? Is it possible for you to write an application and download it to your DVD player for executing? Is it possible for you to add a printer software to your DVD player and connect a printer to your DVD player to take a printout? Is it possible for you to change the functioning of your DVD player to a television by changing the embedded software? The answers to all these questions are 'NO'. The only interface you can find out on the DVD player is the interface for connecting the DVD player with the display screen and one for controlling the DVD player through a remote. Indeed your DVD player is an embedded system. We will see comparison of embedded system and general purpose computing system with the help of the table below.

Table 1.1: Embedded Systems vs. General Computing Systems

Embedded Systems	General Computing Systems
A system which is a combination of special purpose hardware and embedded OS for executing a specific set of applications	A system which is a combination of generic hardware and General Purpose Operating System for executing a variety of applications
May or may not contain an operating system for functioning	Contain a General Purpose Operating System (GPOS)
The firmware of the embedded system is pre-programmed and it is non-alterable by end-user	Applications are alterable (programmable) by user (It is possible for the end user to re-install the Operating System, and add or remove user applications)
Application specific requirements (like performance, power requirements, memory usage etc) are the key deciding factors	Performance is the key deciding factor on the selection of the system. Always "Faster is Better"
Highly tailored to take advantage of the power saving modes supported by hardware and Operating System	Less/not at all tailored towards reduced operating power requirements, options for different levels of power management
For certain category of embedded systems like mission critical systems, the response time requirement is highly critical	Response requirements are not time critical
Execution behavior is deterministic for certain type of embedded systems like "Hard Real Time" systems	Need not be deterministic in execution behavior

1.2 Classification of Embedded Systems

It is possible to have a multitude of classifications for embedded systems, based on different criteria. Some of the criteria used in the classification of embedded systems are as follows:

1. Based on generation
2. Based on complexity and performance requirements
3. Based on deterministic behaviour
4. Based on triggering

1.2.1 Classification Based on Generation

First Generation The early embedded systems built around 8-bit microprocessors like 8085 and Z80 and 4-bit microcontrollers Ex. Stepper motor control units, Digital Telephone Keypads etc.

Second Generation Embedded Systems built around 16-bit microprocessors and 8 or 16-bit microcontrollers, following the first generation embedded systems Ex. SCADA (Supervisory control and data acquisition), Data Acquisition Systems etc.

Third Generation Embedded Systems built around high performance 16/32 bit Microprocessors/controllers, Application Specific Instruction set processors like Digital Signal Processors (DSPs), and Application Specific Integrated Circuits (ASICs).The instruction set is complex and powerful. Ex. Robotics, Industrial process control, networking etc.

Fourth Generation Embedded Systems built around System on Chips (SoC's), Re-configurable processors and multicore processors. It brings high performance, tight integration and miniaturization into the embedded device market E. Smart phone devices, MIDs etc.

1.2.2 Classification based on Complexity & Performance

Small Scale The embedded systems built around low performance and low cost 8 or 16 bit microprocessors/ microcontrollers. It is suitable for simple applications and where performance is not time critical. It may or may not contain OS.

Medium Scale Embedded Systems built around medium performance, low cost 16 or 32 bit microprocessors/microcontrollers or DSPs. These are slightly complex in hardware and firmware. It may contain GPOS/RTOS.

Large Scale Embedded Systems built around high performance 32 or 64 bit RISC processors/controllers, RSoC or multi-core processors and PLD. It requires complex hardware and software. These system may contain multiple processors/controllers and co-units/hardware accelerators for offloading the processing requirements from the main processor. It contains RTOS for scheduling, prioritization and management.

1.2.3 Classification Based on deterministic behavior

It is applicable for Real Time systems. The application/task execution behavior for an embedded system can be either deterministic or non-deterministic. These are classified in to two types:

1. Soft Real time Systems: Missing a deadline may not be critical and can be tolerated to a certain degree

2. Hard Real time systems: Missing a program/task execution time deadline can have catastrophic consequences (financial, human loss of life, etc.)

1.2.4 Classification Based on Triggering

These are classified into two types-

1. **Event Triggered:** Activities within the system (e.g., task run-times) are dynamic and depend upon occurrence of different events.

2. **Time triggered:** Activities within the system follow a statically computed schedule (i.e., they are allocated time slots during which they can take place) and thus by nature are predictable.

1.3 Major Application Areas of Embedded Systems

The application areas and the products in the embedded domain are countless. A few of the important domains and products are listed below:

1. **Consumer electronics:** Camcorders, cameras, etc.

2. **Household appliances:** Television, DVD players, washing machine, fridge, microwave oven, etc.

3. **Home automation and security systems:** Air conditioners, sprinklers, intruder detection alarms, closed circuit television cameras, fire alarms, etc.

4. **Automotive industry:** Anti-lock breaking systems (ABS), engine control, ignition systems, automatic navigation systems, etc.

5. **Healthcare:** Different kinds of scanners, EEG, ECG machines etc.

6. **Banking & Retail:** Automatic teller machines (ATM) and currency counters, point of sales (POS)

1.4 Purpose of Embedded Systems

As mentioned in the previous section, embedded systems are used in various domains like consumer electronics, home automation, telecommunications, automotive industry, healthcare, control & instrumentation, retail and banking applications, etc. Within the domain itself, according to the application usage context,

they may have different functionalities. Each embedded system is designed to serve the purpose of any one or a combination of the following tasks:

1. Data Collection/Storage/Representation

   - Performs acquisition of data from the external world. The collected data can be either analog or digital.

Figure 2: A Digital camera

   - Data collection is usually done for storage, analysis, manipulation and transmission.
   - The collected data may be stored directly in the system or may be transmitted to some other systems or it may be processed by the system or it may be deleted instantly after giving a meaningful representation.

2. Data Communication

   - Embedded Data communication systems are deployed in applications ranging from complex satellite communication systems to simple home networking systems.
   - Embedded Data communication systems are dedicated for data communication. The data communication can happen through a wired interface

Figure 3: A router device

(like Ethernet, RS-232C/USB/IEEE1394 etc) or wireless interface (like Wi-Fi, GSM,/GPRS, Bluetooth, ZigBee etc)

- Network hubs, Routers, switches, Modems etc are typical examples for dedicated data transmission embedded systems

3. Data (Signal) Processing

- Embedded systems with Signal processing functionalities are employed in applications demanding signal processing like Speech coding, synthesis, audio video codec, transmission applications etc.

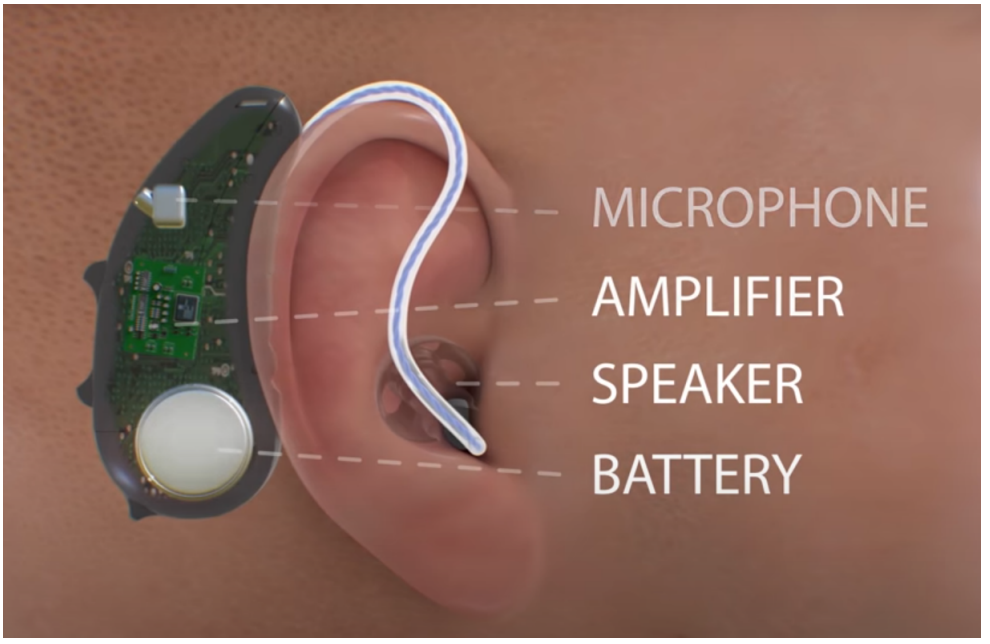

Figure 4: Earpiece is an example of signal processing embedded system

- These are computational intensive systems and typically employ Digital Signal Processors (DSPs)

4. Monitoring

- Embedded systems coming under this category are specifically designed for monitoring purpose.
- They are used for determining the state of some variables using input sensors.
- Electro Cardiogram (ECG) machine for monitoring the heart beat of a patient is a typical example for this. The sensors used in ECG are the different Electrodes connected to the patient's body. Measuring instruments like Digital CRO, Digital Multi meter, Logic Analyzer etc used in Control & Instrumentation applications are also examples of embedded systems for monitoring purpose.

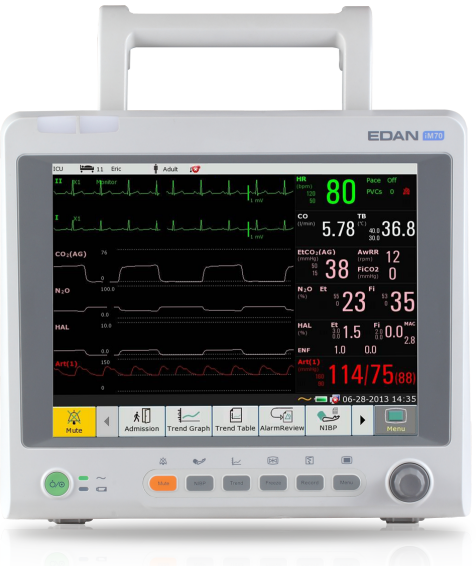

Figure 5: ECG Machine

5. Control

- Embedded systems with control functionalities are used for imposing control over some variables according to the changes in input variables.
- Embedded system with control functionality contains both sensors and actuators. Sensors are connected to the input port for capturing the changes in environmental variable or measuring variable.

Figure 6: Example of appliances that contain both sensors and actuators

- The actuators connected to the output port are controlled according to the changes in input variable to put an impact on the controlling variable to bring the controlled variable to the specified range.
- Air conditioner for controlling room temperature is a typical example for embedded system with 'Control' functionality.

6. Application Specific User Interface

   - Embedded systems which are designed for a specific application
   - Contains Application Specific User interface (rather than general standard UI) like key board, Display units etc.

Figure 7: Elevator key board with limited interface

1.5 Characteristics of Embedded Systems

Embedded systems possess certain specific characteristics and these are unique to each Embedded system.

1. Application and Domain Specific:

   - Each E.S has certain functions to perform and they are developed in such a manner to do the intended functions only.
   - They cannot be used for any other purpose.
   - Ex. The embedded control units of the microwave oven cannot be replaced with AC'S embedded control unit because the embedded control units of microwave oven and AC are specifically designed to perform certain specific tasks.

2. Reactive and Real Time:

   - E.S are in constant interaction with the real world through sensors and user-defined input devices which are connected to the input port of the system.

- Any changes in the real world are captured by the sensors or input devices in real time and the control algorithm running inside the unit reacts in a designed manner to bring the controlled output variables to the desired level.
- E.S produce changes in output in response to the changes in the input, so they are referred as reactive systems.
- Real Time system operation means the timing behavior of the system should be deterministic i.e. the system should respond to requests in a known amount of time.
- Ex. E.S which are mission critical like flight control systems, Antilock Brake Systems (ABS) etc are Real Time systems.

3. Operates in Harsh Environment :

   - The design of E.S should take care of the operating conditions of the area where the system is going to implement.
   - Ex. If the system needs to be deployed in a high temperature zone, then all the components used in the system should be of high temperature grade.
   - Also proper shock absorption techniques should be provided to systems which are going to be commissioned in places subject to high shock.

4. Distributed:

   - It means that embedded systems may be a part of a larger system.
   - Many numbers of such distributed embedded systems form a single large embedded control unit.
   - Ex. Automatic vending machine. It contains a card reader, a vending unit etc. Each of them are independent embedded units but they work together to perform the overall vending function.

5. Small Size and Weight:

   - Product aesthetics (size, weight, shape, style, etc) is an important factor in choosing a product.
   - It is convenient to handle a compact device than a bulky product.

6. Power Concerns:

   - Power management is another important factor that needs to be considered in designing embedded systems.
   - E.S should be designed in such a way as to minimize the heat dissipation by the system.

7. Single-functioned:

   Dedicated to perform a single function.

8. Tightly-constrained:

   Low cost, low power, small, fast, etc.

9. Safety-critical:

   Must not endanger human life and the environment.

1.6 Quality Attributes of Embedded Systems

Quality attributes are the non-functional requirements that need to be documented properly in any system design. Quality attributes can be classified as-

1.6.1 Operational Quality Attributes

The operational quality attributes represent the relevant quality attributes related to the embedded system when it is in the operational mode or online mode. For example:

1. Response : It is the measure of quickness of the system. It tells how fast the system is tracking the changes in input variables. Most of the E.S demands fast response which should be almost real time. Ex. Flight control application.

2. Throughput : It deals with the efficiency of a system. It can be defined as the rate of production or operation of a defined process over a stated period of time. The rates can be expressed in terms of products, batches produced or any other meaningful measurements.

   e.g. In case of card reader throughput means how many transactions the reader can perform in a minute or in an hour or in a day.

   Throughput is generally measured in terms of "Benchmark". A Benchmark is a reference point by which something can be measured.

3. Reliability : It is a measure of how much we can rely upon the proper functioning of the system.

   Mean Time Between Failure (MTBF) and Mean Time To Repair (MTTR) are the terms used in determining system reliability.

   MTBF gives the frequency of failures in hours/weeks/months.

   MTTR specifies how long the system is allowed to be out of order following a failure. For embedded system with critical application need, it should be of the order of minutes.

4. Maintainability: It deals with support and maintenance to the end user or client in case of technical issues and product failure or on the basis of a routine system checkup.

   Reliability and maintainability are complementary to each other.

   A more reliable system means a system with less corrective maintainability requirements and vice versa.

   Maintainability can be broadly classified into two categories-

   (a) Scheduled or Periodic maintenance (Preventive maintenance)

   (b) Corrective maintenance to unexpected failures

5. Security: Confidentiality, Integrity and Availability are the three major measures of information security.

   Confidentiality deals with protection of data and application from unauthorized disclosure.

   Integrity deals with the protection of data and application from unauthorized modification.

   Availability deals with protection of data and application from unauthorized users.

6. Safety: Safety deals with the possible damages that can happen to the operator, public and the environment due to the breakdown of an Embedded System.

   The breakdown of an embedded system may occur due to a hardware failure or a firmware failure.

   Safety analysis is a must in product engineering to evaluate the anticipated damages and determine the best course of action to bring down the consequences of damage to an acceptable level.

1.6.2 Non-Operational Quality Attributes

The quality attributes that needs to be addressed for the product not on the basis of operational aspects are grouped under this category.

1. Testability and Debug-ability:

   - Testability deals with how easily one can test the design, application and by which means it can be done.
   - For an E.S testability is applicable to both the embedded hardware and firmware.

- Embedded hardware testing ensures that the peripherals and total hardware functions in the desired manner, whereas firmware testing ensures that the firmware is functioning in the expected way.
- Debug-ability is a means of debugging the product from unexpected behavior in the system
- Debug-ability is two level process-
  (a) Hardware level: It is used for finding the issues created by hardware problems.
  (b) Software level: It is employed for finding the errors created by the flaws in the software.

2. Evolvability (The capacity of the system for adaptive evolution):

   - It is a term which is closely related to Biology. It is referred as the non-heritable variation.
   - For an embedded system evolvability refers to the ease with which the embedded product can be modified to take advantage of new firmware or hardware technologies.

3. Portability:

   - It is the measure of system independence.
   - An embedded product is said to be portable if the product is capable of functioning in various environments, target processors and embedded operating systems.
   - "Porting" represents the migration of embedded firmware written for one target processor to a different target processor.

4. Time-to-Prototype and Market:

   - It is the time elapsed between the conceptualization of a product and the time at which the product is ready for selling.
   - The commercial embedded product market is highly competitive and time to market the product is critical factor in the success of commercial embedded product.
   - There may be multiple players in embedded industry who develop products of the same category (like mobile phone).

5. Per Unit Cost and Revenue:

   - Cost is a factor which is closely monitored by both end user and product manufacturer.
   - Cost is highly sensitive factor for commercial products
   - Proper market study and cost benefit analysis should be carried out before taking a decision on the per-unit cost of the embedded product.

1.7 Elements of an Embedded System

An embedded system is a combination of 3 things- Hardware, Software and Mechanical Components and it is supposed to do one specific task only. A typical embedded system contains a single chip controller which acts as the master brain of the system. Diagrammatically an embedded system can be represented as shown in figure.

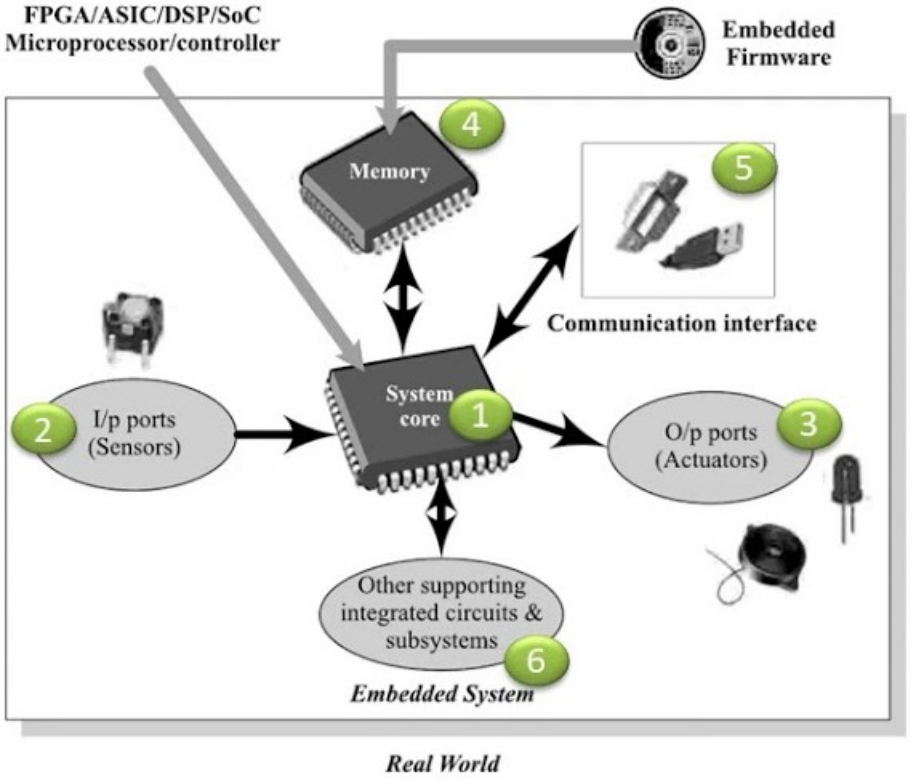

Figure 8: Elements of an Embedded System

Embedded systems are basically designed to regulate a physical variable (such as Microwave Oven) or to manipulate the state of some devices by sending some signals to the actuators or devices connected to the output port system (such as temperature in Air Conditioner), in response to the input signal provided by the end users or sensors which are connected to the input ports. Hence the embedded systems can be viewed as a reactive system.

The core of the system performs some predefined operations on input data with the help of embedded firmware in the system and sends some actuating signals to the actuator connect connected to the output port of the system. The memory of the system is responsible for holding the code (control algorithm and other important configuration details). There are two types of memories are used in any embedded system. Fixed memory (ROM) is used for storing code or program. The user cannot change the firmware in this type of memory. The most common types of memories used in embedded systems for control algorithm

storage are OTP, PROM, UVEPROM, EEPROM and FLASH.

Sometimes the system requires temporary memory for performing arithmetic operations or control algorithm execution and this type of memory is known as "working memory". Random Access Memory (RAM) is used in most of the systems as the working memory. Various types of RAM like SRAM, DRAM, and NVRAM are used for this purpose. The size of the RAM also varies from a few bytes to kilobytes or megabytes depending on the application.

1.7.1 Core of the Embedded Systems

Embedded systems are domain and application specifi c and are built around a central core. The core of the embedded system falls into any one of the following categories:

1. General Purpose and Domain Specific Processors

   Almost 80% of the embedded systems are processor/controller based. The processor may be a microprocessor or a microcontroller or a digital signal processor, depending on the domain and application.

   (a) Microprocessors

   It is a silicon chip representing a Central Processing Unit (CPU), which is capable of performing arithmetic as well as logical operations according to a pre-defined set of Instructions, which is specific to the manufacturer In general the CPU contains the Arithmetic and Logic Unit (ALU), Control Unit and Working registers Microprocessor is a dependant unit and it requires the combination of other hardware like Memory, Timer Unit, and Interrupt Controller etc. for proper functioning.

   (b) Microcontroller

   A highly integrated silicon chip containing a CPU, scratch pad RAM, Special and General purpose Register Arrays, On Chip ROM/FLASH memory for program storage, Timer and Interrupt control units and dedicated I/O ports.

   Microcontrollers can be considered as a super set of Microprocessors.

   Microcontroller can be general purpose (like Intel 8051, designed for generic applications and domains) or application specific (Like Automotive AVR from Atmel Corporation. Designed specifically for automotive applications) Since a microcontroller contains all the necessary functional blocks for independent working, they found greater place in the embedded domain in place of microprocessors

   Microcontrollers are cheap, cost effective and are readily available in the market. Texas Instruments TMS 1000 is considered as the world's first

microcontroller.

(c) Digital Signal Processors (DSPs)

Powerful special purpose 8/16/32 bit microprocessors designed specifically to meet the computational demands and power constraints of today's embedded audio, video, and communications applications. Digital Signal Processors are 2 to 3 times faster than the general purpose microprocessors in signal processing applications. DSPs implement algorithms in hardware which speeds up the execution whereas general purpose processors implement the algorithm in firmware and the speed of execution depends primarily on the clock for the processors. DSP can be viewed as a microchip designed for performing high speed computational operations for addition, subtraction, multiplication and division. A typical Digital Signal Processor incorporates the following key units-

- Program Memory
- Data Memory
- Computational Engine
- I/O Unit

Audio video signal processing, telecommunication and multimedia applications are typical examples where DSP is employed.

2. Application Specific Integrated Circuit (ASIC)

Application Specific Integrated Circuit (ASIC) is a microchip designed to perform a specific or unique application. It is used as replacement to conventional general purpose logic chips. It integrates several functions into a single chip and there by reduces the system development cost.

Most of the ASICs are proprietary products. As a single chip, ASIC consumes very small area in the total system and thereby helps in the design of smaller systems with high capabilities/functionalities.

ASICs can be pre-fabricated for a special application or it can be custom fabricated by using the components from a re-usable *building block* library of components for a particular customer application.

Fabrication of ASICs requires a non-refundable initial investment (Non Recurring Engineering (NRE) charges) for the process technology and configuration expenses If the Non-Recurring Engineering Charges (NRE) is born by a third party and the Application Specific Integrated Circuit (ASIC) is made openly available in the market, the ASIC is referred as **Application Specific Standard Product (ASSP)**.

3. Programmable Logic Devices (PLDs)

Logic devices provide specific functions, including device-to-device interfacing, data communication, signal processing, data display, timing and control operations, and almost every other function a system must perform.

Logic devices can be classified into two broad categories - Fixed and Programmable. The circuits in a fixed logic device are permanent, they perform one function or set of functions - once manufactured, they cannot be changed.

Programmable logic devices (PLDs) offer customers a wide range of logic capacity, features, speed, and voltage characteristics - and these devices can be re-configured to perform any number of functions at any time.

Designers can use inexpensive software tools to quickly develop, simulate, and test their logic designs in PLD based design. The design can be quickly programmed into a device, and immediately tested in a live circuit.

PLDs are based on re-writable memory technology and the device is reprogrammed to change the design.

3.1 CPLDs and FPGAs

The two major types of programmable logic devices are Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs). Of the two, FPGAs offer the highest amount of logic density, the most features, and the highest performance. The largest FPGA now shipping, part of the Xilinx Virtex line of devices, provides eight million “system gates” (the relative density of logic).

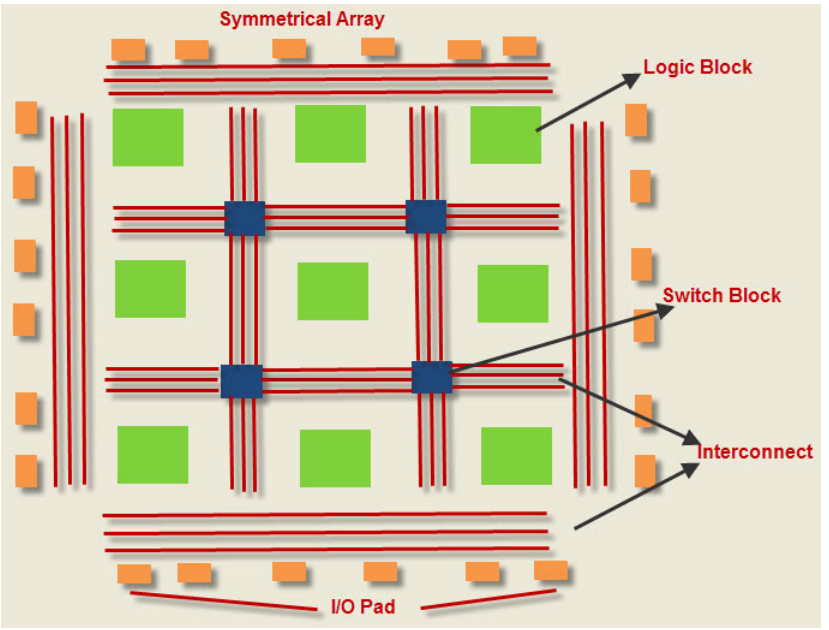

Figure 9: Architectre of an FPGA

CPLDs, by contrast, offer much smaller amounts of logic–up to about 10,000

gates. But CPLDs offer very predictable timing characteristics and are therefore ideal for critical control applications. CPLDs such as the Xilinx CoolRunner™ series also require extremely low amounts of power and are very inexpensive, making them ideal for cost-sensitive, battery-operated, portable applications such as mobile phones and digital handheld assistants.

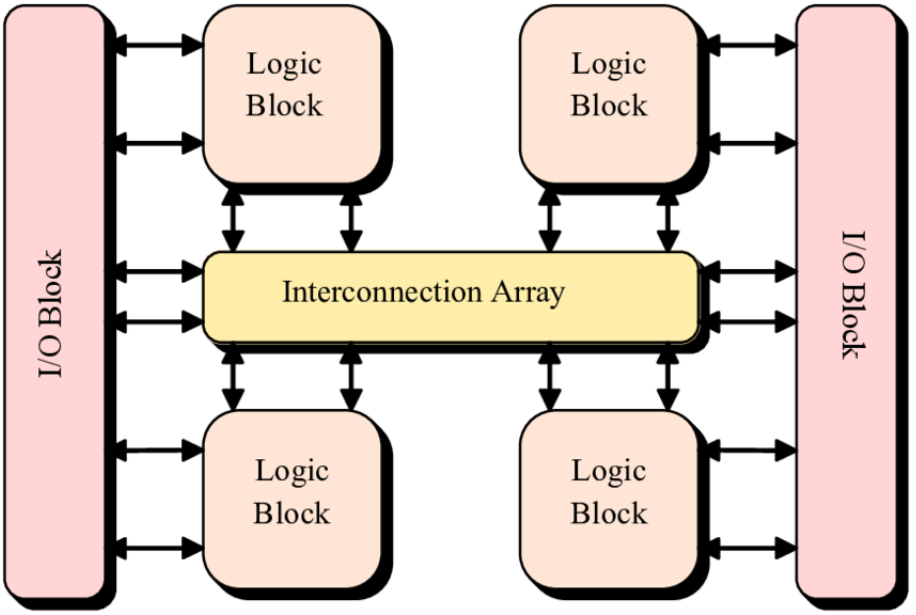

Figure 10: Architectre of a CPLD

Advantages of PLD:

(a) PLDs offer customers much more flexibility during the design cycle because design iterations are simply a matter of changing the programming file, and the results of design changes can be seen immediately in working parts.

(b) PLDs do not require long lead times for prototypes or production parts–the PLDs are already on a distributor's shelf and ready for shipment.

(c) PLDs do not require customers to pay for large NRE costs and purchase expensive mask sets–PLD suppliers incur those costs when they design their programmable devices and are able to amortize those costs over the multi-year lifespan of a given line of PLDs.

(d) PLDs allow customers to order just the number of parts they need, when they need them, allowing them to control inventory. Customers who use fixed logic devices often end up with excess inventory which must be scrapped, or if demand for their product surges, they may be caught short of parts and face production delays.

(e) PLDs can be reprogrammed even after a piece of equipment is shipped to a customer. In fact, thanks to programmable logic devices, a number of equipment manufacturers now tout the ability to add new features or

upgrade products that already are in the field. To do this, they simply upload a new programming file to the PLD, via the Internet, creating new hardware logic in the system.

4. Commercial off the Shelf Component (COTS)

   A Commercial off-the-shelf (COTS) product is one which is used *as-is*. COTS products are designed in such a way to provide easy integration and interoperability with existing system components. Typical examples for the COTS hardware unit are Remote Controlled Toy Car control unit including the RF Circuitry part, high performance, high frequency microwave electronics (2 to 200 GHz), high bandwidth analog-to-digital converters, devices and components for operation at very high temperatures, Electro-optic IR imaging arrays, UV/IR Detectors etc.

   A COTS component in turn contains a General Purpose Processor (GPP) or Application Specific Instruction Set Processor (ASIP) or Application Specific Integrated Chip (ASIC)/Application Specific Standard Product (ASSP) or Programmable Logic Device (PLD).

   The major advantage of using COTS is that they are readily available in the market, cheap and a developer can cut down his/her development time to a great extend. There is no need to design the module yourself and write the firmware . Everything will be readily supplied by the COTs manufacturer. The major drawback of using COTs component in embedded design is that the manufacturer may withdraw the product or discontinue the production of the COTs at any time if rapid change in technology.

Figure 11: Example of COTS- A Serial Ethernet Converter module

1.8 ARM Design Philosophy

1.8.1 The RISC CISC Design Philosophy

The ARM core uses a RISC architecture. RISC is a design philosophy aimed at delivering simple but powerful instructions that execute within a single cycle at a high clock speed.

The RISC philosophy concentrates on reducing the complexity of instructions performed by the hardware. Provide greater flexibility and intelligence in software rather than hardware.

A RISC design places greater demands on the compiler. In contrast, the traditional complex instruction set computer (CISC) relies more on the hardware for instruction functionality, and consequently the CISC instructions are more complicated.

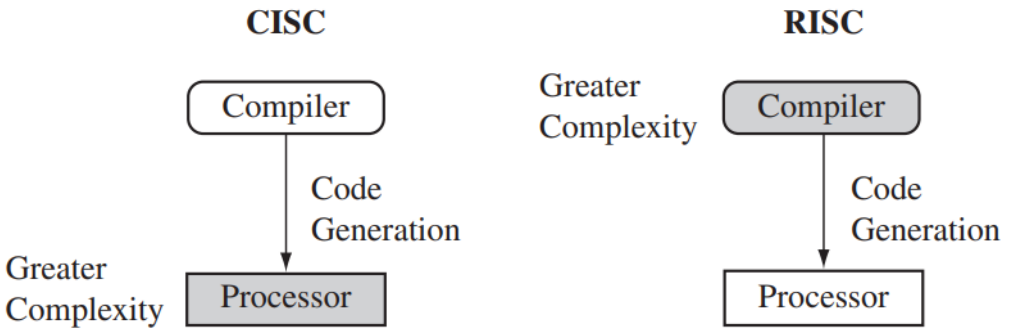

Figure 12: CISC emphasizes hardware complexity, RISC emphasized compiler complexity

1.8.2 The RISC Philosophy

The RISC philosophy is implemented with four major design rules:

1. Instructions

   RISC processors have a reduced number of instruction classes. These classes provide simple operations that can each execute in a single cycle. The compiler or programmer synthesizes complicated operations (for example, a divide operation) by combining several simple instructions. Each instruction is a fixed length to allow the pipeline to fetch future instructions before decoding the current instruction. In contrast, in CISC processors the instructions are often of variable size and take many cycles to execute.

2. Pipelining

   The processing of instructions is broken down into smaller units that can be executed in parallel by pipelines. Ideally the pipeline advances by one step

Table 1.2: RISC vs CISC

RISC	CISC
Lesser number of instructions	Greater number of Instructions
Instruction pipelining and increased execution speed	Generally no instruction pipelining feature
Orthogonal instruction set (Allows each instruction to operate on any register and use any addressing mode)	Non-orthogonal instruction set (All instructions are not allowed to operate on any register and use any addressing mode. It is instruction-specific)
Operations are performed on registers only, the only memory operations are load and store	Operations are performed on registers or memory depending on the instruction
A large number of registers are available	Limited number of general purpose registers
Single, fixed length instructions	Variable length instructions

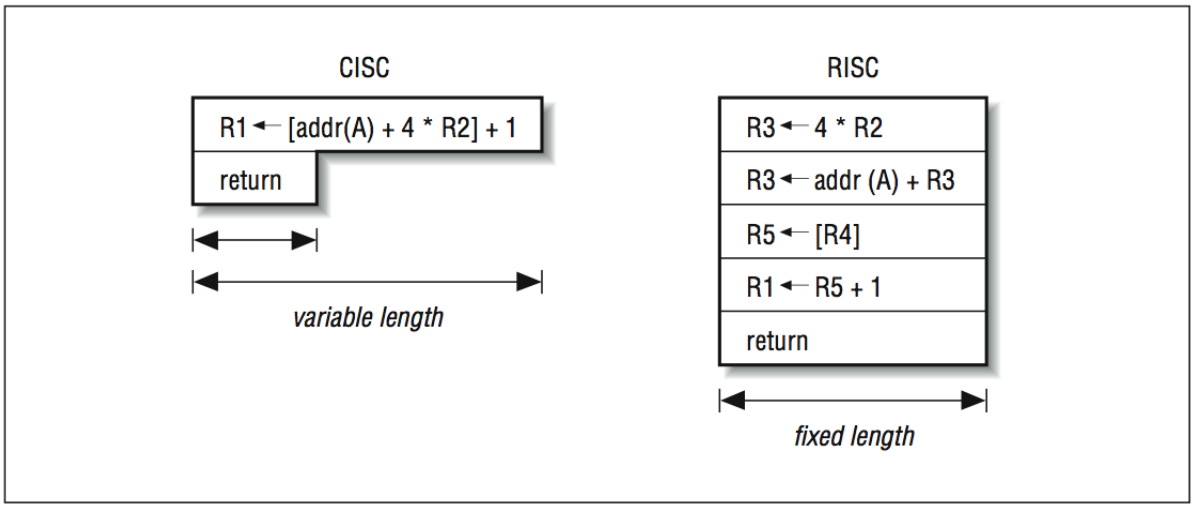

Figure 13: Instruction length in RISC and CISC

on each cycle for maximum throughput. Instructions can be decoded in one pipeline stage. Using a pipeline speeds up execution by fetching the next instruction while other instructions are being decoded and executed. One way to view the pipeline is to think of it as an automobile assembly line, with each stage carrying out a particular task to manufacture the vehicle.

3. Registers

   RISC machines have a large general-purpose register set. Registers act as the fast local memory store for all data processing operations.

4. Load Store Architecture

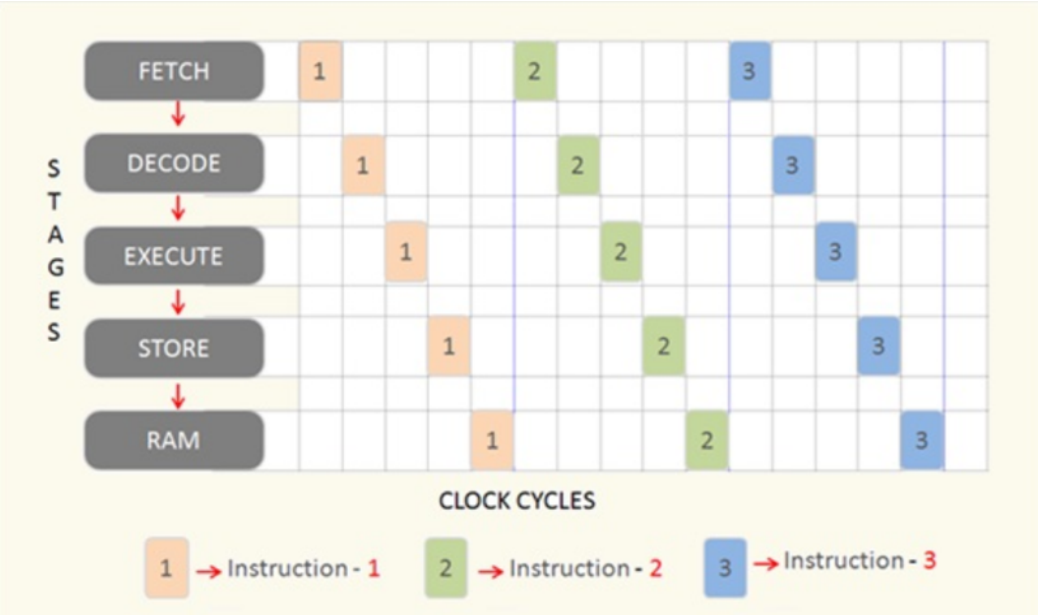

Figure 14: Execution without pipelining

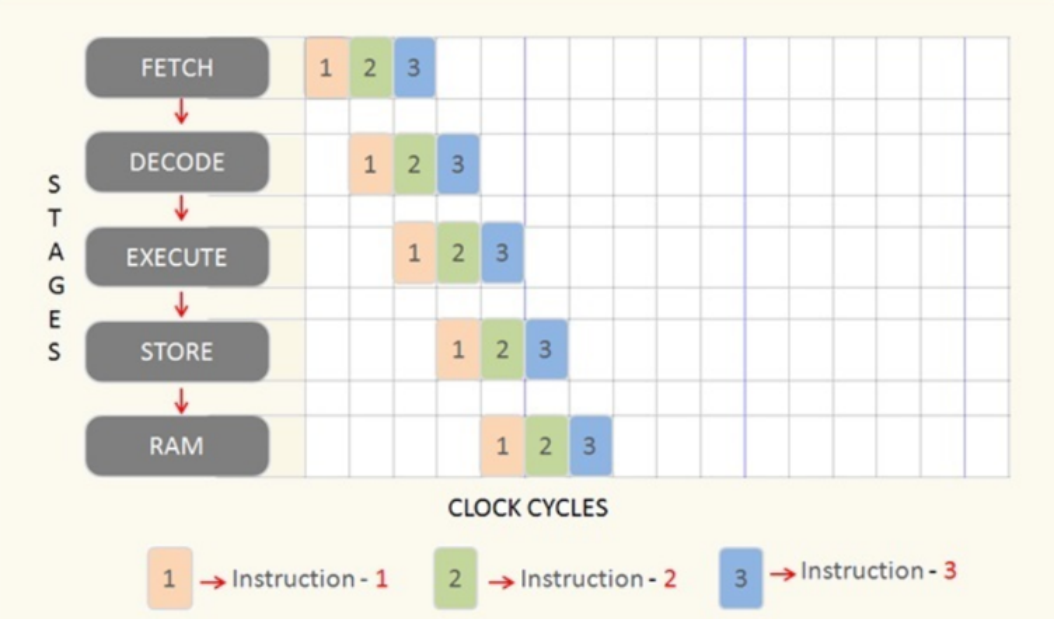

Figure 15: Execution with pipelining

The processor operates on data held in registers. Separate load and store instructions transfer data between the register bank and external memory. Memory accesses are costly, so separating memory accesses from data processing provides an advantage because you can use data items held in the register bank multiple times without needing multiple memory accesses. In contrast, with a CISC design the data processing operations can act on memory directly.

These design rules allow a RISC processor to be simpler. and thus the core can operate at higher clock frequencies In contrast, the traditional CISC processors

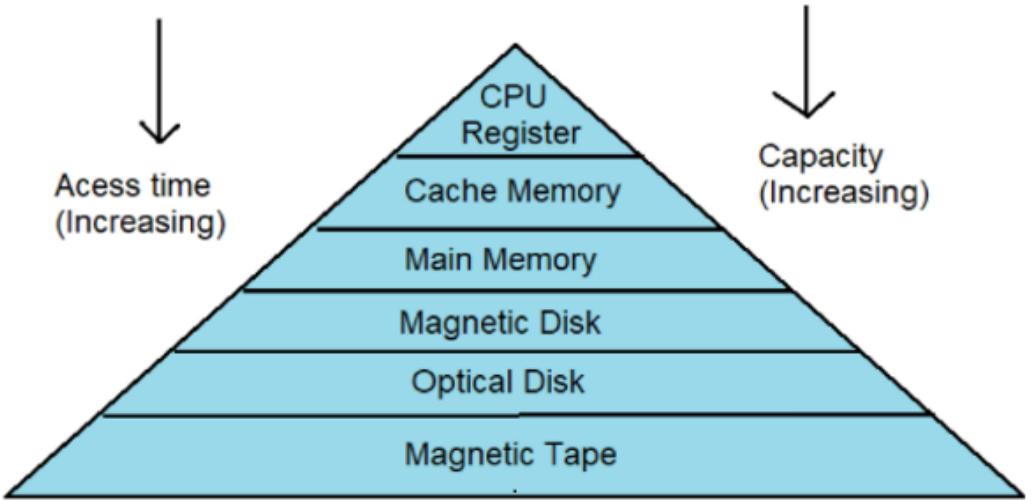

Figure 16: Memory Hierarchy

are more complex and operate at lower clock frequencies.

1.8.3 ARM Design

There are a number of physical features that have driven the ARM processor design.

1. Portable embedded systems require some form of battery power. The ARM processor has been specifically designed to be **small to reduce power consumption** and extend battery operation which is essential for applications such as mobile phones and personal digital assistants (PDAs).

2. **High code density** is another major requirement since embedded systems have limited memory due to cost and/or physical size restrictions. e.g. for applications with limited on-board memory like mobile phones and mass storage devices.

3. Embedded systems are price sensitive and use slow and low-cost memory devices. For high-volume applications like digital cameras, every cent has to be accounted for in the design. The ability to use **low-cost memory devices** produces substantial savings.

4. Another important requirement is to reduce the area of the die taken up by the embedded processor. For a **single-chip solution**, the smaller the area used by the embedded processor, the more available space for specialized peripherals. This in turn reduces the cost of the design and manufacturing since fewer discrete chips are required for the end product.

5. ARM has incorporated **hardware debug** technology within the processor so that software engineers can view what is happening while the processor is

executing code. With greater visibility, software engineers can resolve issues faster, which has a direct effect on the time to market and reduces overall development costs.

6. The ARM core is **not a pure RISC architecture** because of the constraints of its primary application—the embedded system. In some sense, the strength of the ARM core is that it does not take the RISC concept too far. In today's systems, the key is total effective system performance and power consumption rather than only processor speed.

1.8.4 Instruction Set for Embedded Systems

The ARM instruction set differs from the pure RISC definition in several ways that make the ARM instruction set suitable for embedded applications:

- Variable cycle execution for certain instructions:

  Not every ARM instruction executes in a single cycle. For example, load-store-multiple instructions vary in the number of execution cycles depending upon the number of registers being transferred. The transfer can occur on sequential memory addresses, which increases performance since sequential memory accesses are often faster than random accesses. Code density is also improved since multiple register transfers are common operations at the start and end of functions.

- Inline barrel shifter leading to more complex instructions:

  The inline barrel shifter is a hardware component that preprocesses one of the input registers before it is used by an instruction. The ARM arithmetic logic unit has a 32-bit barrel shifter that is capable of shift and rotate operations. The second operand to many ARM and Thumb data-processing and single register data-transfer instructions can be shifted, before the data-processing or data-transfer is executed, as part of the instruction. This expands the capability of many instructions to improve core performance and code density.

- Thumb 16-bit instruction set:

  ARM enhanced the processor core by adding a second 16-bit instruction set called Thumb that permits the ARM core to execute either 16- or 32-bit instructions. The 16-bit instructions improve code density by about 30% over 32-bit fixed-length instructions.

- Conditional execution:

  An instruction is only executed when a specific condition has been satisfied. This feature improves performance and code density by reducing branch instructions.

■ Enhanced instructions:

The enhanced digital signal processor (DSP) instructions were added to the standard ARM instruction set to support fast 16×16-bit multiplier operations and saturation. These instructions allow a faster-performing ARM processor in some cases to replace the traditional combinations of a processor plus a DSP. These additional features have made the ARM processor one of the most commonly used 32-bit embedded processor cores. Many of the top semiconductor companies around the world produce products based around the ARM processor.

1.9 Embedded System models and Development Cycle

The Embedded System's Model represents the layers in which all components existing within an embedded system design can reside. The hardware, which contains all the physical components located on an embedded systems board. System software, which is the device's application-independent software.

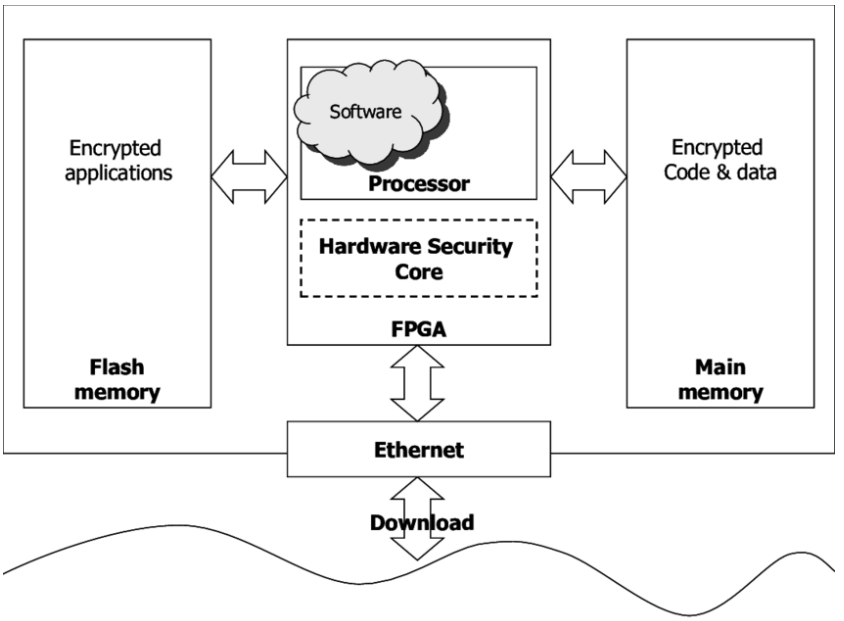

Figure 17: High level model of a typical Embedded system

1.9.1 Stages of embedded system development process

1. Ideation/purpose of the product:

   This is the stage where an idea is discussed with all the stakeholders and brain-stormed to conclude if the idea is worth taking to the development stage.

2. Technical specification:

   A detailed technical design requirement specification document should be created. Product specifications should cover the main things like the purpose of the product, block diagram, main features, environmental conditions, manufacturing requirements, etc.

3. Architecting the Solution (detailing):

   On a high level, how the design will look like, what are strategies you will use to build a solution, which type of power supply, which type of wired or wireless connectivity you will use, how the firmware design will look like, Mobile App/PC software/cloud-based application design needs to be decided. Selection of appropriate component for each block or part of the circuit based on various critical parameters like quality, size, features, cost, lead time, EOL, etc.

4. Component selection & design finalization:

   Power supply design, Hardware component selection.

5. Test Plan:

   Hardware & Software Testing is very important to check the reliability of the product. Companies spend a huge amount of time testing their products to check it's reliability. It includes hardware design validation, software design validation, production level product testing.

6. Design Implementation:

   This is the core engineering stage of the implementation. Here the architecture is converted into the design. For hardware: schematic capture, PCB layout will be done. MCU firmware & software will be developed.

7. PoC / Prototype Development:

   Building the prototype and test it again for the functionality & the environmental conditions.

8. Field Trials:

   Sometimes the product works well in their lab but has issues in the field.

9. Final Product Improvements:

   Improvisation of the test procedures you follow during the design validation/ production testing stage to ensure a flawless product comes out of production.

10. Product Release:

    Various product certifications based on features it has and the countries it will be sold in. Documentation: Product page on the website, Quick introduction video, Datasheet, User guide, Setting up the Support channel, etc.