MODULE -6

Networks – Distributed Embedded Architectures, Networks for embedded systems, Network based design, Internet enabled systems. Embedded Product Development Life Cycle – Description – Objectives -Phases – Approaches1. Recent Trends in Embedded Computing.

Distributed Embedded Systems

1.INTRODUCTION

In a distributed embedded system, several processing elements (PEs) (either microprocessors or ASICs) are connected by a network that allows them to communicate. The application is distributed over the PEs, and some of the work is done at each node in the network.

1.1 Reasons to build network-based embedded systems.

1.When the processing tasks are physically distributed, it may be necessary to put some of the computing power near where the events occur.

2. Data reduction is another important reason for distributed processing. It may be possible to perform some initial signal processing on captured data to reduce its volume.

3.Modularity is another motivation for network-based design. For instance, when a large system is assembled out of existing components, those components may use a network port as a clean interface that does not interfere with the internal operation of the component in ways that using the microprocessor bus would.

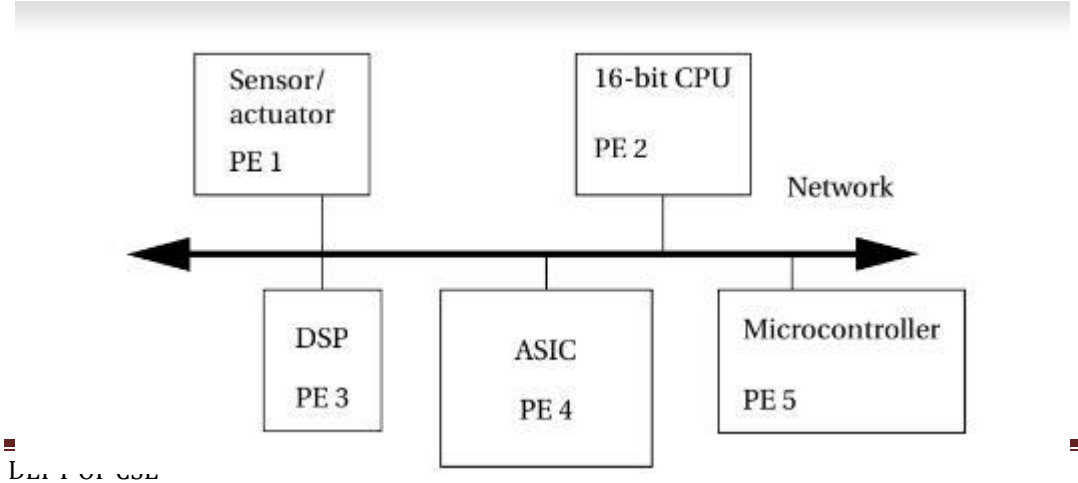
4.A distributed system can also be easier to debug the microprocessors in one part of the network can be used to probe components in another part of the network. Finally, in some cases, networks are used to build fault tolerance into systems.

5.Distributed embedded system design is another example of hardware/software co-design, since we must design the network topology as well as the software running on the network.

2.DISTRIBUTED EMBEDDED ARCHITECTURE

A distributed embedded system can be organized in many different ways, but its basic units are the PE and the network as illustrated in Figure 8.1.

A PE may be an instruction set processor such as a DSP, CPU, or microcontroller, as well as a nonprogrammable unit such as the ASICs used to implement PE 4. An I/O device such as PE 1 (which we call here a sensor or actuator, depending on whether it provides input or output) may also be a PE, so long as it can speak the network protocol to communicate with other PEs. The network in this case is a bus, but other network topologies are also possible. It is also possible that the system can use more than one network, such as when relatively independent functions require relatively little communication among them. We often refer to the connection between PEs provided by the network as a communication link. The system of PEs and networks forms the hardware platform on which the application runs.



.

Fig 8.1 An example of a distributed embedded system.

2.1 Why Distributed?

In some cases,distributed systems are necessary because the devices that the PEs communicate with are physically separated. If the deadlines for processing the data are short, it may be more cost-effective to put the PEs where the data are located rather than build a higher-speed network to carry the data to a distant, fast PE.

An important advantage of a distributed system with several CPUs is that one part of the system can be used to help diagnose problems in another part.Whether you are debugging a prototype or diagnosing a problem in the field, isolating the error to one part of the system can be difficult when everything is done on a single CPU. If you have several CPUs in the system,you can use one to generate inputs for another and to watch its output.

2.2 Network Abstractions

Networks are complex systems. Ideally, they provide high-level services while hiding many of the details of data transmission from the other components in the system. In order to help understand (and design) networks, the International Standards Organization has developed a seven-layer model for networks known as Open Systems Interconnection (OSI) models

The seven layers of the OSI model, shown in Figure 8.2, are intended to cover a broad spectrum of networks and their uses. Some networks may not need the services of one or more layers because the higher layers may be totally missing or an

intermediate layer may not be necessary. However, any data network should fit into the OSI model. The OSI layers from lowest to highest level of abstraction are described below.

■ Physical: The physical layer defines the basic properties of the interface between systems, including the physical connections (plugs and wires), electrical properties, basic functions of the electrical and physical components, and the basic procedures for exchanging bits.

■ Data link: The primary purpose of this layer is error detection and control across a single link. However, if the network requires multiple hops over several data links, the data link layer does not define the mechanism for data integrity between hops, but only within a single hop.

■ Network: This layer defines the basic end-to-end data transmission service.The network layer is particularly important in multihop networks.

■ Transport: The transport layer defines connection-oriented services that ensure that data are delivered in the proper order and without errors across multiple links.This layer may also try to optimize network resource utilization.

■ Session: A session provides mechanisms for controlling the interaction of end-user services across a network, such as data grouping and checkpointing.

■ Presentation: This layer defines data exchange formats and provides transformation utilities to application programs.

■ Application: The application layer provides the application interface between the network and end users.

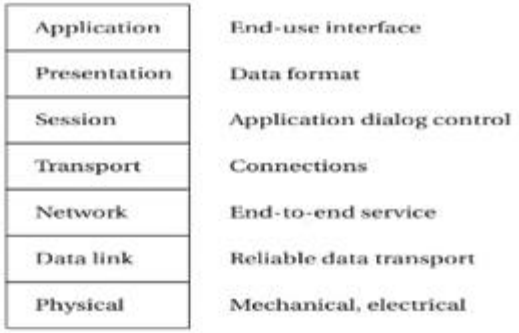


fig 8.2 The OSI Model layers

2.3 Hardware and Software Architectures

Distributed embedded systems can be organized in many different ways depending upon the needs of the application and cost constraints. One good way to understand possible architectures is to consider the different types of interconnection networks that can be used.

A point-to-point link establishes a connection between exactly two PEs. Point-to-point links are simple to design precisely because they deal with only two components.We do not have to worry about other PEs interfering with communication on the link.

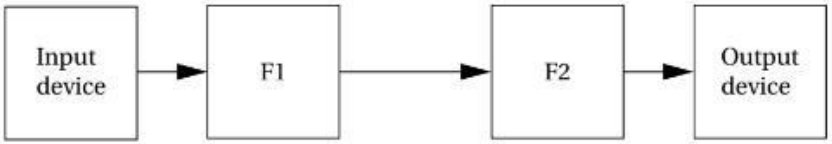


Figure 8.3 shows a simple example of a distributed embedded system built from point-to-point links. The input signal is sampled by the input device and passed to the first digital filter, F1, over a point-to-point link. The results of that filter are sent through a second point-
to-point link to filter F2. The results in turn are sent to the output device over a third point-to-point link. A digital filtering system requires that its outputs arrive at strict intervals, which means that the filters must process their inputs in a timely fashion. Using point-to-point connections allows both F1 and F2 to receive a new sample and send a new output at the same time without worrying about collisions on the communications network.

A bus is a more general form of network since it allows multiple devices to be connected to it. Like a microprocessor bus, PEs connected to the bus have addresses. Communications on the bus generally take the form of packets as illustrated in Figure 8.4. A packet contains an address for the destination and the data to be delivered. It frequently includes error

detection/correction information such as parity. It also may include bits that serve to signal to other PEs that the bus is in use, such as the header shown in the figure. The data to be transmitted from one PE to another may not fit exactly into the size of the data payload on the packet. It is the responsibility of the transmitting PE to divide its data into packets; the receiving PE must of course reassemble the complete data message from the packets.

Distributed system buses must be arbitrated to control simultaneous access,just as with microprocessor buses. Arbitration scheme types are summarized below.

■ Fixed-priority arbitration always gives priority to competing devices in the same way. If a high-priority and a low-priority device both have long data transmissions ready at the same time, it is quite possible that the low-priority device will not be able to transmit anything until the high-priority device has sent all its data packets.

■ Fair arbitration schemes make sure that no device is starved. Round-robin arbitration is the most commonly used of the fair arbitration schemes. The PCI bus requires that the arbitration scheme used on the bus must be fair, although it does not specify a particular arbitration scheme. Most implementations of PCI use round-robin arbitration. A bus has limited available bandwidth. Since all devices connect to the bus,communications can interfere with each other. Other network topologies can be used to reduce communication conflicts.At the opposite end of the generality spectrum from the bus is the crossbar network shown in Figure 8.5.

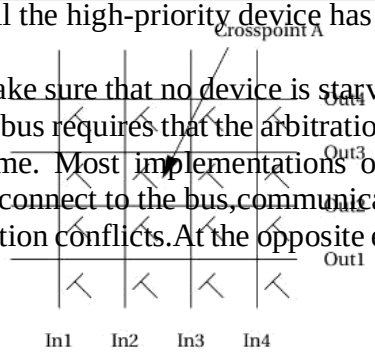


FIGURE 8.5
A crossbar network.

A crossbar not only allows any input to be connected to any output, it also allows all combinations of input/output connections to be made. Thus, for example, we can simultaneouslyconnect in1 to out4, in2 to out3, in3 to out2, and in4 to out1 or any other combinations of inputs. (Multicast connections can also be made from one input to several outputs.) A crosspoint is a switch that connects an input to an output. To connect an input to an output, we activate the crosspoint at the intersection between the corresponding input and output lines in the crossbar. For example, to connect in2 and out3 in the figure, we would activate crossbar A as shown. The major drawback of the crossbar network is expense: The size of the network grows as the square of the number of inputs (assuming the numbers of inputs and outputs are equal).

Many other networks have been designed that provide varying amounts of parallel communication at varying hardware costs. Figure 8.6 shows an example multistage network. The crossbar of Figure 8.5 is a direct network in which messages go from source to destination without going through any memory element. Multistage networks have intermediate routing nodes to guide the data packets.

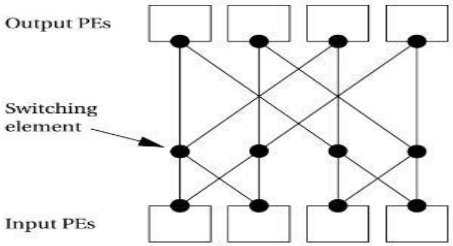


FIGURE 8.6
A multistage network.

2.4 Message Passing Programming

Distributed embedded systems do not have shared memory,so they must communicate by passing messages.We will refer to a message as the natural communication unit of an algorithm; in general, a message must be broken up into packets to be sent on the network. A procedural interface for sending a packet might look like the following:

send_packet(address,data);

The routine should return a value to indicate whether the message was sent successfully if the network includes a handshaking protocol. If the message to be sent is longer than a packet,it must be broken up into packet-size data segments as follows:

for (i = 0; i < message.length;i=i+ PACKET_SIZE)

send_packet(address,&message.data[i]);

The above code uses a loop to break up an arbitrary-length message into packet- size chunks. However, clever system design may be able to recast the message to take advantage of the packet format. For example, clever encoding may reduce the length of the message enough so that it fits into a single packet. On the other hand, if the message is shorter than a packet or not an even multiple of the packet data size, some extra information may be packed into the remaining bits of a packet. Reception of a packet will probably be implemented with interrupts. The simplest procedural interface will simply check to see whether a received message is waiting in a buffer. In a more complex RTOS-based system, reception of a packetmay enable a process for execution.

Network protocols may encourage a **data-push design** style for the system built around the network. In a single-CPU environment,a program typically initiates a read whenever it wants data. In many networked systems, nodes send values out without any request from the intended user of the system. Data-push programming makes sense for periodic data—if the data

will always be used at regular intervals, we can reduce data traffic on the network by automatically sending it when it is needed.

3.NETWORKS FOR EMBEDDED SYSTEMS

Networks for embedded computing span a broad range of requirements; many of those requirements are very different from those for general-purpose networks. Some networks are

used in safety-critical applications, such as automotive control. Some networks, such as those used in consumer electronics systems, must be very inexpensive. Other networks,such as industrial control networks,must be extremely rugged and reliable. IO devices communicate with processor through an IO bus ,which is separate from memory bus that the processor used to communicate with the memory system. Embedded system communicate internally on the same IC or systems with very short and long distances and can be networked by using following bus each functioning according to specific protocols.

1. Using serial IO bus allows a computer or controller or embedded system to interface network with a wide range of IO devices without having to implement a specific interface for each IO device.when the IO devices in the distributed embedded system are networked at long distances of 25cm and bove all can communicate through a common serial bus.A serial bus has very few lines
2. Using parellel IO bus allows a computer or controller or embedded system to interface with a number of internal systems at very short distances without having to implement a specific interface for each IO device
3. Using the internet or intranet a computer, controller or embedded system IO device can interface globally and can network with othersystems or computers and a wide range of devices in the distributed system.
4. Using wireless protocols allows a handheld computer ,controller or embedded system IO device to interface and network with a number of handheld system IO devices at short devices upto 100 m using a wireless personel area(WPAN) protocol without having to implement a specific wireless interface for each IO device

Embedded systems are distributed and networked using a serial or parallel bus or wireless protocol software and appropriate hardware.Several interconnect networks have been developed especially for distributed embedded computing:

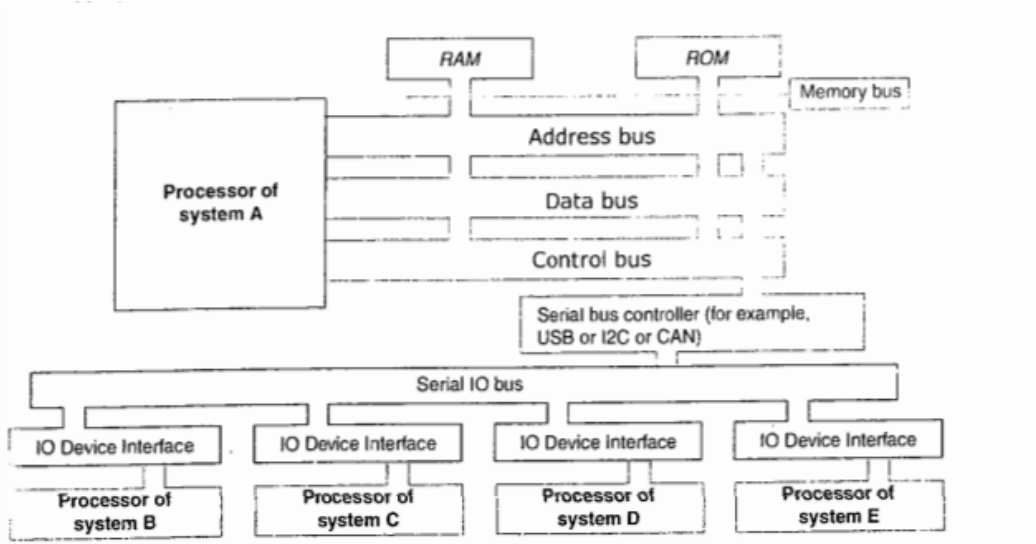


Fig. 3.9 A processor of embedded system connected to system memory bus and networked to other systems through a serial bus

3.1 Serial Bus Communication Protocols

The following describes most popular serial buses

1. The I2 C bus is used in microcontroller-based systems.
2. The Controller Area Network (CAN) bus was developed for automotive electronics. It provides megabit rates and can handle large numbers of devices.

3. USB Bus

4. Firewire-IEEE 1394 bus standard
5. Advanced high speed buses
6. Ethernet and variations of standard Ethernet are used for a variety of control applications. In addition, many networks designed for general-purpose computing have been put to use in embedded applications as well.

3.1 .1 The I2C Bus

The ***I 2C bus*** is a well-known bus commonly used to link microcontrollers into systems. I 2C is designed to be low cost,easy to implement, and of moderate speed (up to 100 KB/s for the standard bus and up to 400 KB/s for the extended bus). As a result, it uses only two lines: the serial data line (SDL) for data and the serial clock line (SCL), which indicates when valid data are on the data line. Figure 8.7 shows the structure of a typical I2C bus system. Every node in the network is connected to both SCL and SDL. Some nodes may be able to act as bus masters and the bus may have more than one master. Other nodes may act as slaves that only respond to requests from masters.

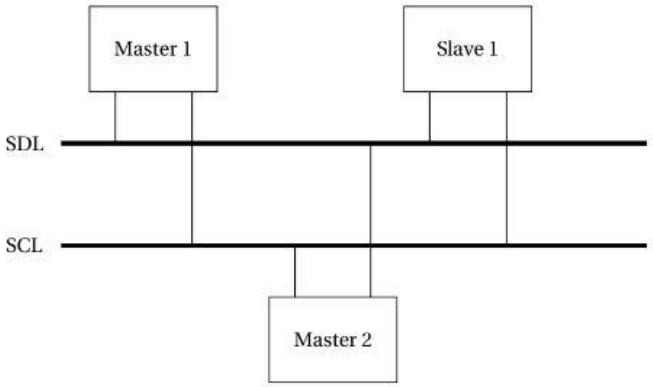


FIGURE 8.7
Structure of an I^2C bus system.

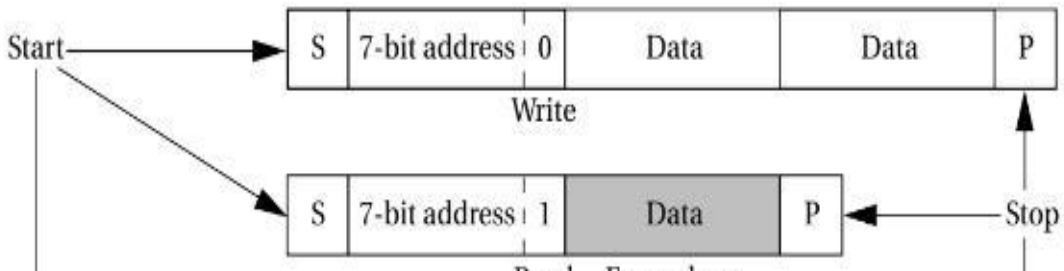
The I2C bus is designed as a multimaster bus—any one of several different devices may act as the master at various times. As a result, there is no global master to generate the clock signal on SCL. Instead, a master drives both SCL and SDL .when it is sending data.When the bus is idle, both SCL and SDL remain high.When two devices try to drive either SCL or SDL to different values, the open collector open drain circuitry prevents errors, but each master device must listen to the bus while transmitting to be sure that it is not interfering with another message—if the device receives a different value than it is trying to transmit, then it knows that it interfering with another message.

Every I2 C device has an address. The addresses of the devices are determined by the system designer, usually as part of the program for the I2 C driver. The addresses must of course be chosen so that no two devices in the system have the same address. A device address is 7 bits in the standard I2 C definition (the extended I2 C allows 10-bit addresses).

A bus transaction is initiated by a start signal and completed with an end signal as follows:

■ A start is signaled by leaving the SCL high and sending a 1 to 0 transition on SDL.

■ A stop is signaled by setting the SCL high and sending a 0 to 1 transition on SDL.

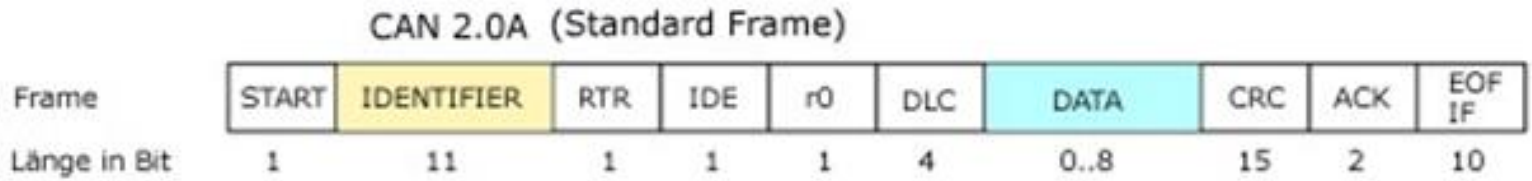


3.1.2 CAN BUS

- CAN or Controller Area Network or CAN-bus is an ISO standard computer network protocol and bus standard, designed for microcontrollers and devices to communicate with each other without a host computer.
- Designed earlier for industrial networking but recently more adopted to automotive applications, CAN have gained widespread popularity for embedded control in the areas like industrial automation, automotives, mobile machines, medical, military and other harsh environment network applications.
- Development of the CAN-bus started originally in 1983 at Robert Bosch GmbH. The protocol was officially released in 1986. and the first CAN controller chips, produced by Intel and Philips, introduced in the market in the year of 1987.
- The CAN is a "broadcast" type of bus. That means there is no explicit address in the messages. All the nodes in the network are able to pick-up or receive all transmissions. There is no way to send a message to just a specific node. To be more specific, the messages transmitted from any node on a CAN bus does not contain addresses of either the transmitting node, or of any intended receiving node. Instead, an identifier that is unique throughout the network is used to label the content of the message. Each message carries a numeric value, which controls its priority on the bus, and may also serve as an identification of the contents of the message. And each of the receiving nodes performs an

acceptance test or provides local filtering on the identifier to determine whether the message, and thus its content, is relevant to that particular node or not, so that each node may react only on the intended messages. If the message is relevant, it will be processed; otherwise it is ignored.

- How do they communicate?
  If the bus is free, any node may begin to transmit. But what will happen in situations where two or more nodes attempt to transmit message (to the CAN bus) at the same time. The identifier field, which is unique throughout the network helps to determine the priority of the message. A "non-destructive arbitration technique" is used to accomplish this, to ensure that the messages are sent in order of priority and that no messages are lost. The lower the numerical value of the identifier, the higher the priority. That means the message with identifier having more dominant bits (i.e. bit 0) will overwrite other nodes' less dominant identifier so that eventually (after the arbitration on the ID) only the dominant message remains and is received by all nodes.



- **SOF- Start of Fr**ame. The message starts from this point.
- **Identifier**: It decides the priority of the message. Lower the binary value, higher is the priority. It is 11 bit.
- **RTR**– Remote Transmission Request. It is dominant when information is required from another node. Each node receives the request, but only that node whose identifier matches that of the message is the required node. Each node receives the response as well.z
- **IDE**– Single Identification Extension. If it is dominant, it means a standard CAN identifier with no extension is being transmitted.
- **R0**– reserved bit.
- **DLC**– Data Length Code. It defines the length of the data being sent. It is 4 bit
- **Data**– Up to 64 bit of data can be transmitted.
- **CRC**– Cyclic Redundancy Check. It contains the checksum (number of bits transmitted) of the preceding application data for error detection.
- **ACK**– Acknowledge. It is 2 bit. It is dominant if an accurate message is received.
- **EOF**– end of frame. It marks end of can frame and disables bit stuffing.
- **IFS**– Inter Frame Space. It contains the time required by the controller to move a correctly received frame to its proper position.

Different message types are:

1. **Data Frame**: It consists of arbitrary field, data field, crc field and the acknowledge fields.
2. **Remote Frame**: It requests for transmission of data from another node. Here the RTR bit is recessive.
3. **Error Frame**: It is transmitted when a error is detected.
4. **Overload Frame**: It is used to provide delay between messages. It is transmitted when the nodes become too busy.
5. **Valid Frame**: A message is valid if the EOF field is recessive. Else the message is transmitted again.

3.1.3 USB BUS

It connets flash memory cards, pen-like memory devices, digital camera, printer, mouse-device, PocketPC, video games, Scanner etc

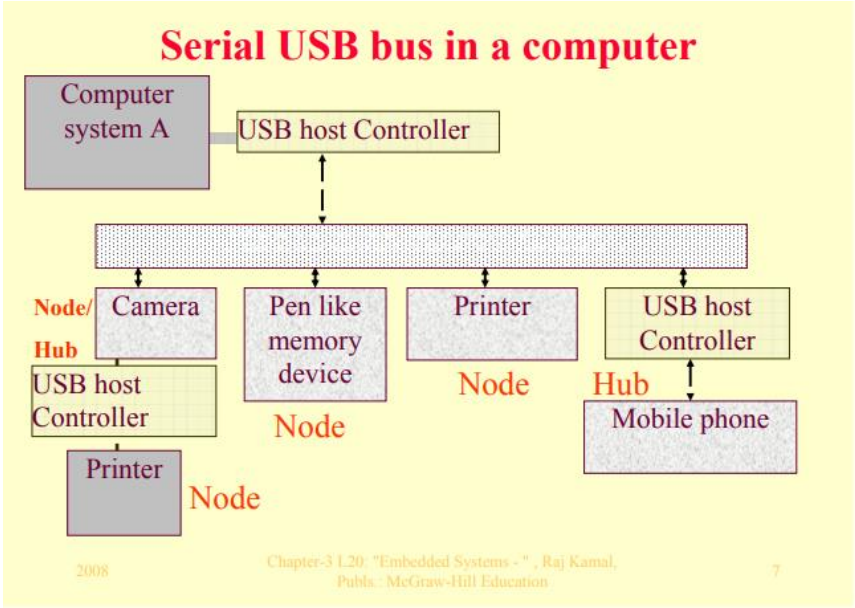
USB allows Serial transmission and reception between host and serial devices . The data transfer is of four types:

(a) Controlled data transfer, (b) Bulk data transfer, (c) Interrupt driven data transfer, (d) Iso-synchronous transfer

A bus between the host system and interconnected number of peripheral devices .Maximum 127 devices can connect a host. Three standards:

- USB 1.1 (a low speed 1.5 Mbps 3 meter channel along with a high speed 12 Mbps 25 meter channel)
- USB 2.0 (high speed 480 Mbps 25 meter channel)
- wireless USB (high speed 480 Mbps 3 m)

Host connection to the devices or nodes is using USB port driving software and host controller, Host computer or system has a hostcontroller, which connects to a root hub. A hub is one that connects to other nodes or hubs. A tree- like topology is used



The root hub connects to the hub (s) and node (s) at level 1. A hub at level 1 connects to the hub (s) and node (s) at level 2 and so on. Root hub and each hub at a level have a star topology with the next level. Only the nodes are present at the last level.

USB Device features : Device Can be hot plugged (attached), configured and used, reset, reconfigured and used Bandwidth sharing with other devices: Host schedules the sharing of bandwidth among the attached devices at an instance. Can be detached (while others are in operation) and reattached. Attaching and detaching USB device or host without rebooting

USB device descriptor: Has data structure hierarchy as follows: It has device descriptor at the root, which has number of configuration descriptors, which has number of interface descriptor and which has number of end point descriptor.

Powering USB device : A device can be either bus-powered or self- powered. In addition, there is a power management by software at the host for USB ports

USB protocol :

- USB bus cable has four wires, one for +5V, two for twisted pairs and one for ground.Termination impedances at each end as per the device-speed.
- Electromagnetic Interference (EMI)- shielded cable for the 15 Mbps USB devices.
- Serial signals NRZI (Non Return to Zero (NRZI) The synchronization clock encoded by inserting synchronous code (SYNC) field before each USB packet Receiver synchronizes its bits recovery clock continuously
- A polled bus
- Host controller regularly polls the presence of a device as scheduled by the software. It sends a token packet. The token consists of fields for type, direction, USB device address and device end-point number. • The device does the handshaking through a handshake packet, indicating successful or unsuccessful transmission. A CRC field in a data packet permits error detection

USB supported three types of pipes

1. 'Stream' with no USB- defined protocol. It is used when the connection is already established and the data flow starts
2. 'Default Control' for providing access. •
3. 'Message' for the control functions for of the device.

Host configures each pipe with the data bandwidth to be used, transfer service type and buffer sizes.

Wireless USB:

Wireless extension of USB 2.0 and it operates at UWB (ultra wide band) 3.1 GHZ to 10.6 GHz frequencies. For short-range personal area network (high speed 480 Mbps 3 meter or 110 Mbps 10 meter channel)

3.1.4 FireWire-IEEE 1395 Bus Standard

Firewire connecting

• FireWire IEEE 1394a port up to 400 Mbps

• 1394b up to 800 Mbps

• Serial isosynchronous data transfer

• Transfers data at a guaranteed rate

• Also used in real time devices, such as video device data transfers

Applications

Multimedia streaming devices

• digital video cameras,

• digital camcorders,

• digital video disk (DVD),
• set-top boxes,
• music systems multimedia peripherals,
• latest hard disk drives,
• latest high speed printers

FireWire IEEE 1394 Protocol Features

- A single 1394 port can interface up to 63 external FireWire devices.
- Supports both plug and play and hot plugging.
- Provides self-powered and buspowered support on the bus

3.1.5 Advanced Serial High Speed Bus 3

Are for handheld devices.The following protocols were used

1. IEE 802.3 -2000[Gbps bandwidth Gigabit Ethernet MAC for 125Mhz performance]
2. IEEP802.3oe draft 4.1[10mbps Ethernet performance]
3. IEEP802.3oe draft 4.1[12.5mbps Ethernet performance]
4. XAUI(10 Gigabit attachment unit)
5. XSBI(10 Gigabit Serial Bus Interchange)
6. SONET OC-48
7. SONET OC-192
8. SONET OC-768
9. ATM OC-12/46/192F

3.2 Parallel bus device protocols

Parallel bus interconnects IO devices and peripherals over very short distances and at high speed. ISA, PCI and ARM buses are the examples of parallel buses. A parallel bus interfaces the system memory bus through a bridge or switching circuit.

3.2.1. ISA BUS

An Industry Standard Architecture bus (ISA bus) is a computer bus that allows additional expansion cards to be connected to a computer's motherboard. It is a standard bus architecture for IBM compatibles. Introduced in 1981, the ISA bus was designed to support the Intel 8088 microprocessor for IBM's first-generation PC. The ISA bus has evolved from its original 8-bit standard to 16-bit standard available in most PCs today. It operates at 8.33MHz. Its data transfer data rate is 8MB/s. It has 24 address lines and 16 data lines. It is used to connect printer, scanner, modem, sound card, CD-ROM etc. In the late 1990s the faster peripheral component interconnect (PCI). Soon afterwards, use of the ISA bus began to diminish, and most IBM motherboards were designed with PCI slots.The ISA bus provides direct memory access using multiple expansion cards on a memory channel allowing separate interrupt request transactions for each card. Depending on the version, the ISA bus can support a network card, additional serial ports, a video card and other processors and architectures, including:

- IBM PC with Intel 8088 microprocessor
- IBM AT with Intel 80286 processor (1984)
- Extended Industry Standard Architecture (1988)

EISA bus is the 32 bit data and address line version of ISA and devices also are supported

3.2.2 PCI and PCI/X Buses

Parallel bus enables a host computer or system to communicate simultaneously 32-bit or 64-bit with other devices or systems, for example, to a network interface card (NIC) or graphic card.
When the I/O devices in the distributed embedded subsystems are networked all can communicate through a common parallel bus. • PCI connects at high speed to other subsystems having a range of I/O devices at very short distances (<25cm) using a parallel bus without having to implement a specific interface for each I/O device.
PCI bus Applications: PCI bus connects display monitor, printer, character devices, network subsystems, video card, modem card, hard disk controller, thin client, digital video capture card, streaming displays, 10/100 Base T card, Card with 16 MB Flash ROM with a router gateway for a LAN and Card using DEC 21040 PCI Ethernet LAN controller.

PCI Bus Features

- 32- bit data bus extendible to 64 bits.

- PCI protocol specifies the ways of interaction between the different components of a computer.
- A specification version 2.1 ─ synchronous/asynchronous throughput is up to 132/ 528 MB/s [33M × 4/ 66M × 8 Byte/s], operates on 3.3V to 5V signals.
- PCI driver can access the hardware automatically as well as by the programmer assigned addresses. Automatically detects the interfacing systems and assigns new addresses Thus, simplified addition and deletion (attachment and detachment) of the system peripherals.
- Each device may use a FIFO controller with a FIFO buffer for maximum throughput.

Identification Numbers : A device identifies its address space by three identification numbers, (i) I/O port (ii) Memory locations and (iii) Configuration registers of total 256B with a four 4-byte unique ID. Each PCI device has address space allocation of 256 bytes to access it by the host computer

PCI device identification A sixteen16-bit register in a PCI device identifies this number to let that device auto- detect it. Another sixteen16-bit register identifies a device ID number. These two numbers let allow the device to carry out its auto-detection by its host computer.

PCI Standards

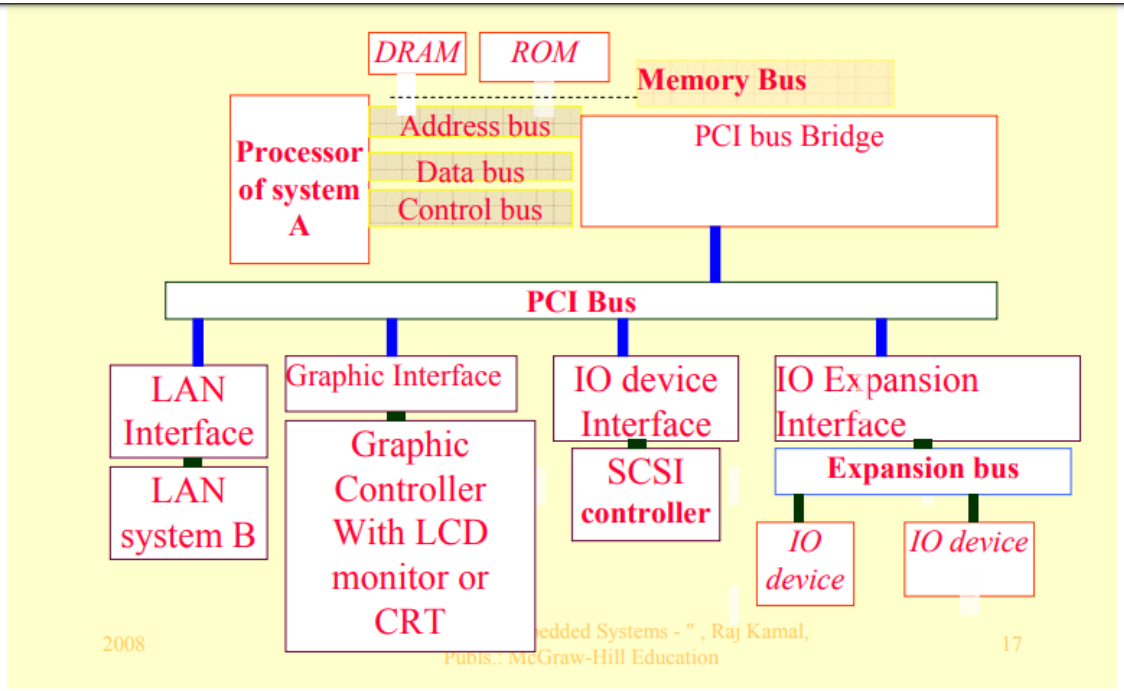
- PCI 32bit/33 MHz, and 64bit/66 MHz
- PCI Extended (PCI/X) 64 bit/100 MHz
- Compact PCI (cPCI) Bus

Two super speed versions

- PCI Super V2.3 264/528 MBps 3.3V (on 64- bit bus), and 132/264 (on 32-bit bus)
- PCI-X Super V1.01a for 800MBps 64- bit bus 3.3Volt.

PCI bridge : PCI bus interface switches a processor communication with the memory bus to PCI bus. In most systems, the processor has a single data bus that connects to a switch module PCI bridge Some processors integrate the switch module onto the same integrated circuit as the processor to reduce the number of chips required to build a system and thus the system cost.

PCI bridge/switch Communicates with the memory through a memory bus (a set of address, control and data buses), a dedicated set of wires that transfer data between these two systems. A separate I/O bus connects the PCI switch to the I/O devices.



PCI/X(PCI Extended)

- 33 MBps to as much as 1 GBps
- Backward compatible with existing PCI cards
- Used in high bandwidth devices (Fiber Channel, and processors that are part of a cluster and Gigabit Ethernet)
- Maximum 264 MBps throughput, uses 8, 16, 32, or 64 bit transfers
- 6U cards contain additional pins for user defined I/Os
- Live insertion support (Hot-Swap),
- Supports two independent buses on the back plane (on different connectors)
- Supports Ethernet, Infiniband, and Star Fabric support (Switched fabric based systems) Compact PCI (cPCI)

Each PCI device on Bus Each PCI device on Bus Perform a specific function, May contain a processor and software to perform a specific function. Each device has the specific memory address-range, specific interrupt-vectors (pre-assigned or auto configured) and the device I/O port addresses. A bus of appropriate specifications and protocol interfaces these to the host computer system or compute

Unique feature of PCI bus unique feature is its configuration address space.

PCI Controller features

• Accesses one device at a time

• All the devices within host device or system can share the I/O port and memory addresses, but cannot share the configuration registers

• Device cannot modify other configuration registers but can access other device resources or share the work or assist the other device

PCI device initilization

• A device can initialize at booting time

• Avoids any address collision

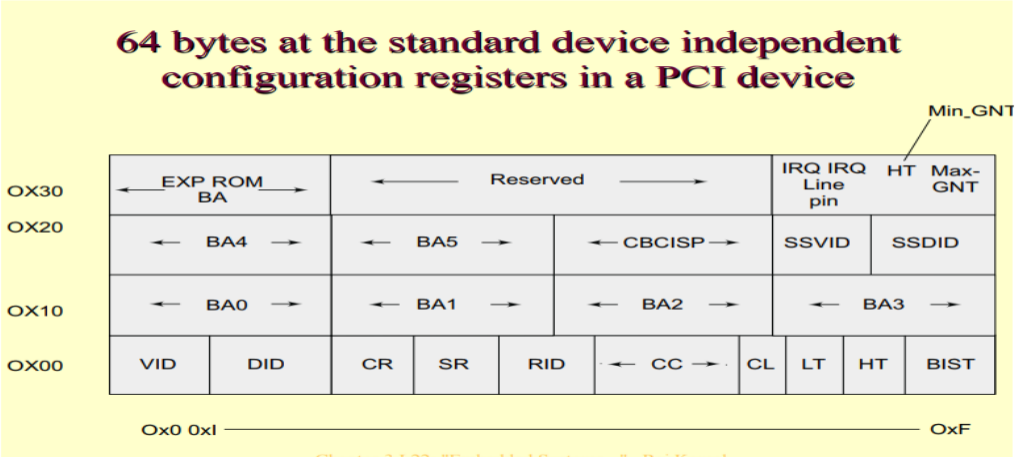
• Device on boot up disables its interrupt and closes its door to its address space except to the configuration registers space

PCI BIOS

Performs the configuration transactions and then, memory and address spaces automatically map to the address space in the device hosting system

PCI device Interrupt Handling : A uniquely assigned interrupt type (a number) handles an interrupt. For example, interrupt type 3 has the interrupt vector address 0x0000C and four bytes at the address specify the interrupt service routine address. Interrupt type can be a number between 0x00 and 0xFF.

Configuration register number 60 :Stores the one byte for the interrupt type n (pci) The PCI device when interrupted handles the interrupt of type n(pci)



Meaning of Terms in Figure
VID: Vendor ID.
DID: Device ID.
RID: Revision ID.
CR: Common Register.
CC: Class Code.
SR: Status Register.
CL: Cache Line.
LT: Latency Timer.
BIST: Base Input Tick.
HT: Header Type.
BA: Base Address. C
BCISB: Card Base CIS Pointer.
SS: Sub System.
ExpROM: Expansion ROM.
MIN_GNT: Minimum Guaranteed time
MAX_GNT: Maximum Guaranteed Time.

3.2.3 ARM Bus

ARM processor interfaces the memory ,external DRAM using AMBA .

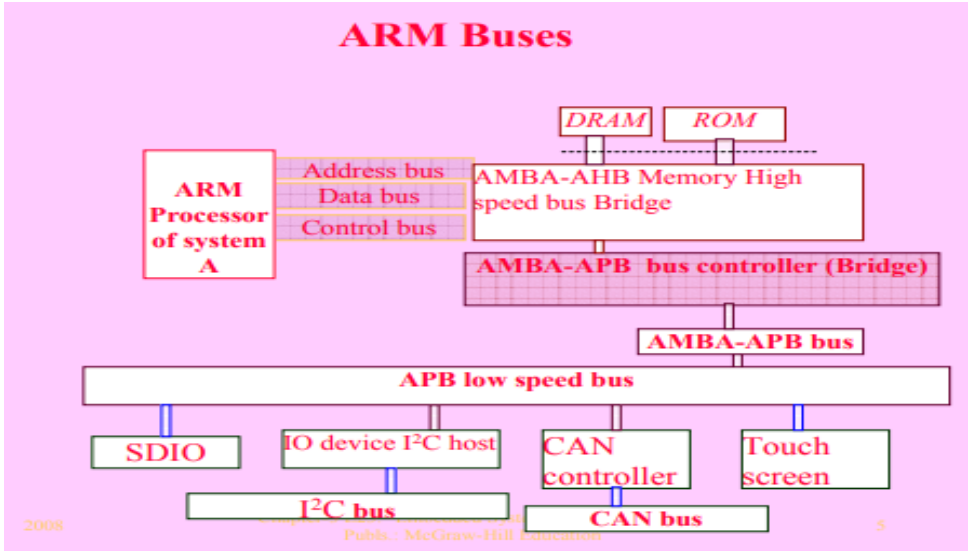
AMBA (ARM Main Memory Bus Architecture) AMBA (ARM Main Memory Bus Architecture) AHB (ARM High Performance Bus)

- AMBA-AHB interfaces the memory, external DRAM (dynamic RAM controller and on-chip I/O devices
- AMBA-AHB connects to 32-bit data and 32-bit address lines at high speed
- AHB maximum bps bandwidth─ sixteen times ARM processor clock

3.2.4 AMBA (ARM Main Memory Bus Architecture) AMBA (ARM Main Memory Bus Architecture) APB (ARM Peripheral Bus)

AMBA -APB interfaces ARM processor with the memory AMBAAHB and external -chip I/O devices, which operate at low speed using a bridge (AMBA-APB bridge)

AMBA -APB bridge :Switches ARM CPU communication with the AMBA bus to APB bus. ARM processor based microcontroller has a single data bus in AMBA-AHB that connects to the bridge, which integrate the bridge onto the same integrated circuit as the processor to reduce the number of chips required to build a system and thus the system cost. The bridge communicates with the memory through a AMBA-AHB, a dedicated set of wires that transfer data between these two systems. A separate APB I/O bus connects the bridge to the I/O devices.



APB bus

connects I ^{2} C , touch screen, SDIO ,MMC (multimedia card) , USB , CAN bus and other required interfaces to an ARM microcontroller

3.2.5 Advanced parallel high speed buses

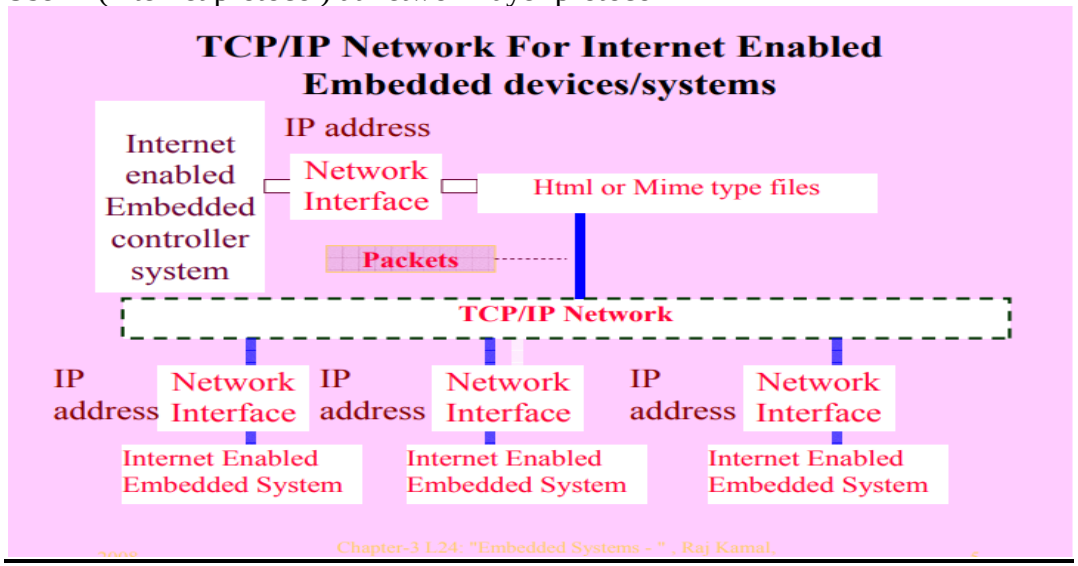
An embedded system may need to connect IO system using gigabit parallel synchronous interfaces. The following are advanced bus standard and proprietary protocols developed recently

1. GMII-Gigabit Ethernet MAC internet Interface
2. XGMI-10 Gigabit Ethernet MAC internet Interface
3. CSIX-1.6.6 Gbps 32bit HSTL with 200Mhz performance
4. Rapidio ^{TM} Interconnect specification v1.1 at 8Gbps with 500Mbps performance

4. Internet enabled systems-Network protocols

- Internet enabled system have Communication to other on the Internet.
- Use html (hyper text markup language) or MIME (Multipurpose Internet Mail Extension) type files

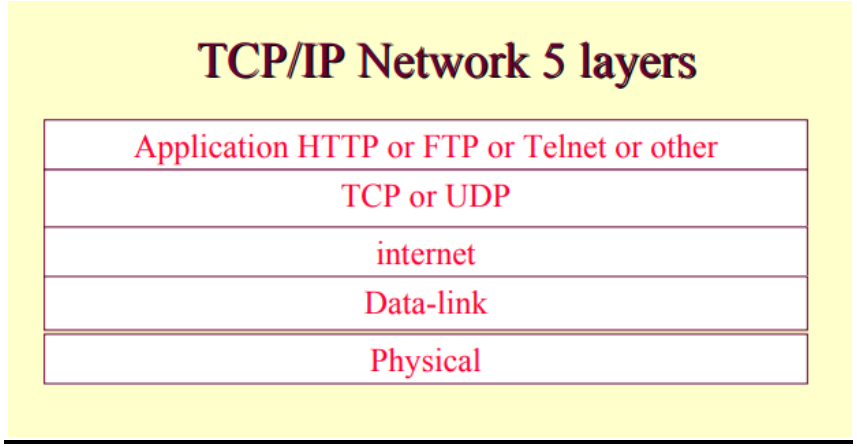
- Use TCP (transport control protocol) or UDP (user datagram protocol) as transport layer protocol addressed by an IP address
- Use IP (internet protocol) at network layer protocol



MIME Format to enable attachment of multiple Format to types of files txt (text file) , doc (MSOFFICE Word document file) , gif (graphic image format file) jpg (jpg format image file) , wav format voice or music file

A system at one IP address Communication with other system at another IP address using the physical connections on the Internet and routers . Since Internet is global network, the system connects to remotely as well as short range located system.

There are five layers in a TCP/IP network ─ Application, transport, network, data-link and physical Application layer protocol also specifies presentation ways. Transport layer protocol also specifies provide session establishment and termination ways. Each layer has a protocol, which specifies the way in which the data or message from previous layer transfer to next layer



4.1 Hyper-Text Transfer protocol(HTTP)

This Layer accepts the data, for example, in HTML or text format and puts the header words as per the protocol and sends application header and data to transport layer. A port number specifies the application in the header. A port assigned number supports multiple logical connections using a socket, and a socket has an IP address and port number. A registered port number is between 0 and 1023. Registration is done by IANA (Internet Assigned Number Authority. Port number 0 means host itself.

Following are the important application layer protocols that support TCP/IP networking

1. HTTP (Port 80) enables Internet connectivity by Hyper-Text Transfer Protocol (HTTP).
2. FTP (Port 21 for control, 20 for data) enables file transfer connectivity by File Transfer Protocol.
3. TFTP (Port 69) for Trivial FTP.
4. NFS (Network File System) is used for sharing files on a network.
5. TELNET (Port 23) enables remote login to remote terminals by Terminal Access Protocol.
6. SMTP (Port 25) enables e-mail transfer, store and forward by Simple Mail Transfer Protocol.

7. PoP3 (Port 110) enables e-mail retrieval.
8. NNTP (Port 119) by (Network News Transfer Protocol).
9. DNS (Port 53) for Domain Name Service.
10. SNMP (Port 161) is Simple Network Management Protocol.
11. Bootps and Bootpc (Ports 67 and 68) for Bootstrap Protocol Server and Client, respectively.
12. DHCP (Dynamic Host Configuration Protocol) is used for remote booting as well as for configuring a system.

HTTP (Port 80) features HTTP (Port 80) features

- standard protocol for requesting for a URL (universal resource locator, for example, http://www.mcgraw-hill.com)

- stateless protocol. For HTTP request, the protocol assumes a fresh request. It means there is no session or sequence number field or no field that is retained in the next exchange. This makes a current exchange by an HTTP request independent of the previous exchanges

.

- A file–transfer like protocol for HTML files. This makes it easy to explore a web site URL. A request (from a client) is sent and reply (response from a server) is received.

- The HTTP protocol is very light (a small format) and thus speedy as compared to other existing protocols. HTTP is able to transfer any type of data to a browser (a client) provided it is capable of handling that data.

- HTTP is flexible

- HTTP protocol is based on Object Oriented Programming System. Methods are applied to objects identified by URL
  HTTP specific methods

1	**GET** The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.
2	**HEAD** Same as GET, but transfers the status line and header section only.
3	**POST** A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.
4	**PUT** Replaces all current representations of the target resource with the uploaded content.
5	**DELETE** Removes all current representations of the target resource given by a URI.

6	**CONNECT** Establishes a tunnel to the server identified by a given URI.
7	**OPTIONS** Describes the communication options for the target resource.
8	**TRACE** Performs a message loop-back test along the path to the target resource.

HTTP interaction scheme A client requests a server directly or through proxy- or gateway An HTTP message therefore either a request or a response

4.2 Transport control protocol(TCP)

TCP protocol use in transport layer Accepts the message from the upper layer (application layer) on a transmission by an application or session layer. Accepts a data stream from the network layer at the receiving end. Before communicating a message to the next network layer, it may add a header.

TCP message : The message may communicate in parts or segments or fragments. The header generally has the additional bits for the source and destination addresses. Also there are bits in it for the sequence and the acknowledge management, flow and error controls, etc.

TCP feature : Specifies a format of byte streams at the transport layer of the TCP/IP suite. TCP is used for a full duplex acknowledged flow. Its format has a TCP header of five plus (n −5) words for options and padding and data of maximum l words.

4.3 User datagram protocol(UDP)

When message is connection less and stateless, then transport layer protocol in TCP/IP suite specifies a protocol called UDP. UDP supports the broadcast networking mode. Example ─ application for communicating a header before a data stream. UDP header specifies the bits for the source and destination ports, the total length of message including header and check sum (optional). During reception, UDP message to the upper layer flows after deleting the header bits from the received transport layer header. Header bits added at the transmitting time from the application or session layers are thus strip from the message

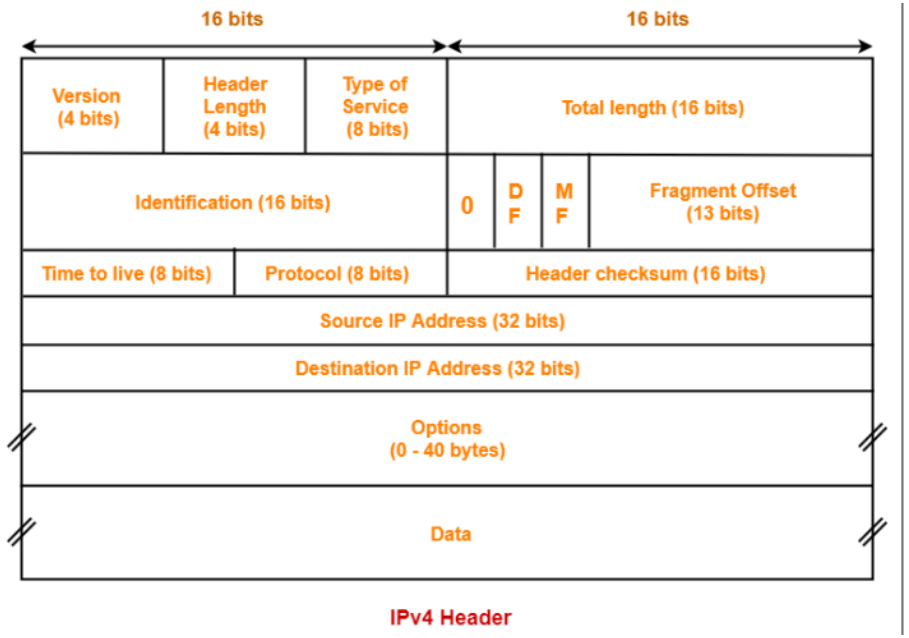
4.4 Internet protocol

All internet enabled devices communicate using internet protocol.Transport layer data in TCP or UDP message format transmits on the network after first division into the packets at the network layer, called internet layer.

IP Packet transmission : Each packet transmits through a chain of routers on the Internet. Packet a minimum unit of data, which transmit on Internet through the routers. Several packets form a source can reach a destination using different routes and can have different delays.

IP packet and routing of packets: The packet consists of an IP header plus data or an IP header plus a routing protocol with the routing messages. The packet has a maximum of 216 bytes (214 words, 1 word = 32 bits = 4 bytes).

The network routing is as per the standard IPv4 (Internet protocol version 4) or IPv6 (Internet protocol version 6). IPv6 broadband protocol.



IPv4 Header

1. **Version-** is a 4 bit field that indicates the IP version used.The most popularly used IP versions are version-4 (IPv4) and version-6 (IPv6).

2. **Header length** -is a 4 bit field that contains the length of the IP header.It helps in knowing from where the actual data begins.

3. **Type of service**- is a 8 bit field that is used for Quality of Service (QoS).The datagram is marked for giving a certain treatment using this field.
4. **Total length**- is a 16 bit field that contains the total length of the datagram (in bytes).
5. **Identification**- is a 16 bit field.It is used for the identification of the fragments of an original IP datagram.
6. **DF bit-** stands for Do Not Fragment bit.Its value may be 0 or 1. When DF bit is set to 0,It grants the permission to the intermediate devices to fragment the datagram if required.When DF bit is set to 1,It indicates the intermediate devices not to fragment the IP datagram at any cost.
7. **Fragment Offset** is a 13 bit field.It indicates the position of a fragmented datagram in the original unfragmented IP datagram.
8. **Time to live** (TTL) is a 8 bit field.-It indicates the maximum number of hops a datagram can take to reach the destination.
9. **Protocol** is a 8 bit field.It tells the network layer at the destination host to which protocol the IP datagram belongs to.
10. **Header checksum** is a 16 bit field.It contains the checksum value of the entire header.
11. **Source IP Address** is a 32 bit field.it contains the logical address of the sender of the datagram.
12. **Destination IP Address** is a 32 bit field it contains the logical address of the receiver of the datagram.
13. Options is a field whose size vary from 0 bytes to 40 bytes.This field is used for several purposes such as Record route,Source routing &Padding
14. **Padding**-Addition of dummy data to fill up unused space in the transmission unit and make it conform to the standard size is called as padding.

4.5 Ethernet

Ethernet is very widely used as a local area network for general-purpose computing. The physical organization of an Ethernet is very simple, as shown in Figure 8.14. The network is a bus with a single signal path; the Ethernet standard allows for several different implementations such as twisted pair and coaxial cable. Unlike the I2 C bus, nodes on the Ethernet are not synchronized—they can send their bits at any time. I2 C relies on the fact that a collision can be detected and quashed within a single bit time thanks to synchronization. But since Ethernet nodes are not synchronized, if two nodes decide to transmit at the same time, the message will be ruined.

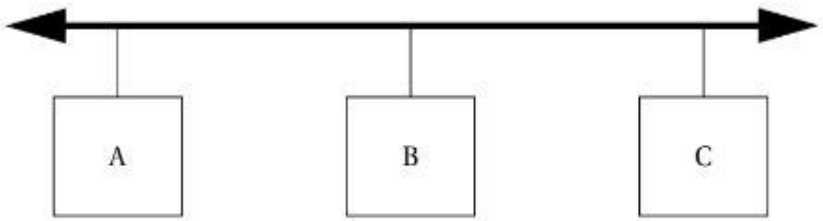


FIGURE 8.14

Ethernet organization.

The Ethernet arbitration scheme is known as ***Carrier Sense Multiple Access*** with ***Collision Detection (CSMA/CD)***. The algorithm is outlined in Figure 8.15. A node that has a message waits for the bus to become silent and then starts transmitting. It simultaneously listens, and if it hears another transmission that interferes with its transmission, it stops transmitting and waits to retransmit. The waiting time is random, but weighted by an exponential function of the number of times the message has been aborted. Figure 8.16 shows the expo- nential backoff function both before and after it is modulated by the random wait time. Since a message may be interfered with several times before it is successfully transmitted, the ***exponential backoff*** technique helps to ensure that the network does not become overloaded at high demand factors. The random factor in the wait time minimizes the chance that two messages will repeatedly interfere with each other. The maximum length of an Ethernet is determined by the nodes' ability to detect collisions.

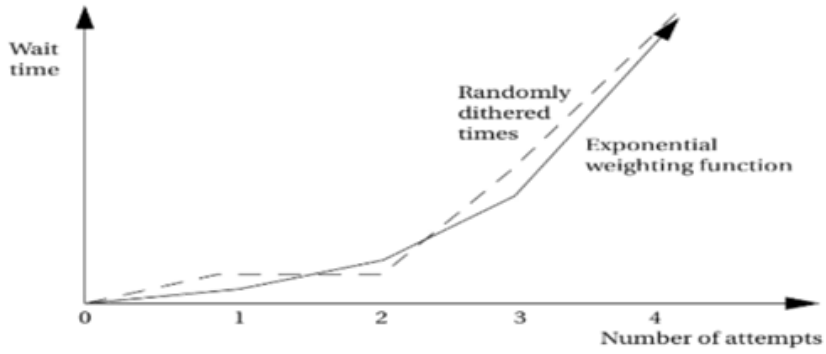
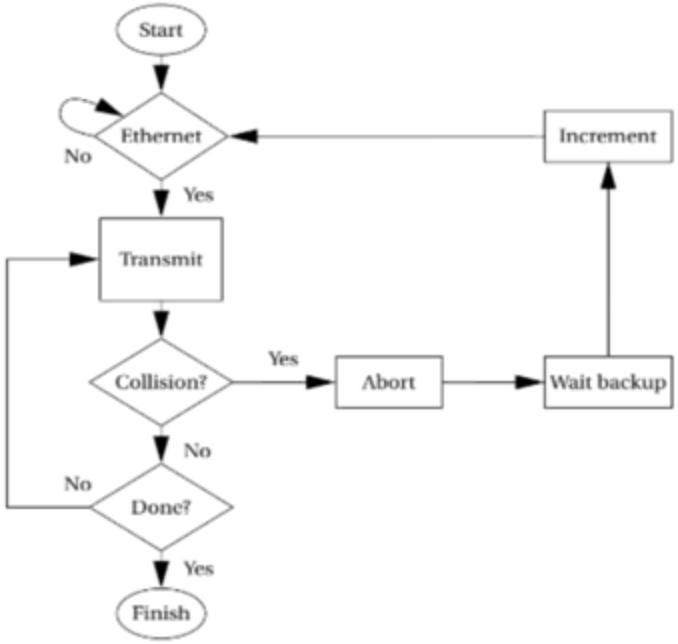


FIGURE 8.16
Exponential backoff times.

FIGURE 8.15

The Ethernet CSMA/CD algorithm.



Preamble	Start frame	Destination address	Source address	Length	Data	Padding	CRC

FIGURE 8.17

Ethernet packet format.

5 . Wireless and Mobile network protocols

5.1 IrDA (Infrared Data Association)

Used in mobile phones, digital cameras, keyboard, mouse, printers to communicate to laptop computer and for data and pictures download and synchronization. Used for control TV, air-conditioning, LCD projector, VCD devices from a distance.Use infrared (IR) after suitable modulation of the data bits. Communicates over a line of sight Phototransistor receiver for infrared rays

5.2 Bluetooth 2.4 GHz

Bluetooth is a wireless technology standard for exchanging data between fixed and mobile devices over short distances using short-wavelength UHF radio waves in the industrial, scientific and medical radio bands, from 2.400 to 2.485 GHz, and building personal area networks (PANs). It was originally conceived as a wireless alternative to RS-232 data cables.

5.3 802.11

In IEEE 802.11 wireless local area networking standards (including Wi-Fi), a **service set** is a group of wireless network devices that are operating with the same networking parameters.

Service sets are arranged hierarchically,: **basic service sets** (**BSS**) are units of devices operating with the same medium access characteristics (i.e. radio frequency, modulation scheme etc.), while **extended service sets** (**ESS**) are logical units of one or more basic service sets on the same logical network segment (i.e. IP subnet, VLAN etc.). There are two classes of basic service sets: those that are formed by infrastructure mode redistribution points (access points or mesh nodes), and those that are formed by independent stations in a peer-to-peer ad hoc topology. Basic service sets are identified by **BSSIDs** (**basic service set identifiers**), which are 48-bit labels that conform to MAC-48 conventions. Logical networks (including extended

service sets) are identified by **SSIDs** (**service set identifiers**), which serve as "network names" and are typically natural language labels.

5.4 ZigBee

Zigbee is an IEEE 802.15.4-based specification for a suite of high-level communication protocols used to create personal area networks with small, low-power digital radios, such as for home automation, medical device data collection, and other low-power low-bandwidth needs, designed for small scale projects which need wireless connection. Hence, Zigbee is a low-power, low data rate, and close proximity (i.e., personal area) wireless ad hoc network.

The technology defined by the Zigbee specification is intended to be simpler and less expensive than other wireless personal area networks (WPANs), such as Bluetooth or more general wireless networking such as Wi-Fi. Applications include wireless light switches, home energy monitors, traffic management systems, and other consumer and industrial equipment that requires short-range low-rate wireless data transfer.

6. EMBEDDED PRODUCT DEVELOPMENT LIFE CYCLE (EDLC)

Just like the SDLC used in Software Development, there is EDLC used in Embedded product development. This chapter explains what is the EDLC, its objectives, the phases that are involved in the EDLC.

EDLC is Embedded Product Development Life Cycle

It is an Analysis – Design – Implementation based problem solving approach for embedded systems development.

Analysis involves understanding what product needs to be developed

Design involves what approach to be used to build the product

Implementation is developing the product by realizing the design.

Need for EDLC

- EDLC is essential for understanding the scope and complexity of the work involved in embedded systems development

- It can be used in any developing any embedded product
- EDLC defines the interaction and activities among various groups of a product development phase including project management, system design ,System testing, Release management and quality assurance
  ,

Objectives of EDLC

- The ultimate aim of any embedded product in a commercial production setup is to produce Marginal benefit
- Marginal is usually expressed in terms of Return On Investment
- The investment for product development includes initial investment, manpower, infrastructure investment etc.
- EDLC has three primary objectives are:

1. Ensuring that high quality products are delivered to user
2. Risk minimization and defect prevention in product development through project management
3. Maximize productivity

1. Ensuring that high quality products are delivered to user

- The primary definition of quality in any embedded product development is return on investment achieved by the product. The expenses incurred for developing the product the product are:-

  Initial investment

  Developer recruiting

  Training

  Infrastructure requirement related

- In order to survive in market, quality is very important factor to be taken care of while developing the product.
- Qualitative attributes depends on the budget of the product so budget allocation is very important.
- Budget allocation might have done after studying the market, trends & requirements of product, competition .etc.
- ELDC must ensure that the development of the product has taken account of all the quality attributes of the embedded system

2. Risk minimization and defect prevention in product development through project management

- There are projects in embedded product development which requires loose or tight management. If the development is a simple one a senior developer itself can take the charge of a management activity and there is no need for a skilled project manager to look after this with dedicated effort. But there will be an overall supervision form the skilled project management team for ensuring that the development process is going in the right direction. Project which are complex and requires timeliness should have a dedicated and a skilled project management part and hence they are said to be tightly bounded to project management.
- Project management -

  -Adds an extra cost on budget

  -But essential for ensuring the development process is going in right direction

- Projects in EDLC requires Loose project management or tight project management.
- PM is required for

Predictability

□□□□Analyze the time to finish the product (PDS = no of person days).this is estimated on the basis of past experience

Co-ordination

Resources (developers) needed to do the job

Risk management

- Backup of resources to overcome critical situation
- Ensuring defective product is not developed

- project management is essential for ' predictability co-ordination and risk minimization
- Resource allocation is critical and it is having a direct impact on investment
- Microsoft @ Project Tool is a typical example for CASE tools for project management

3. Increased Productivity

- Productivity is a Measure of efficiency as well as ROI This productivity measurement is based on total manpower efficiency
- Different ways to improve the productivity are

- Saving the manpower
  - X members – X period
  - X/2 members – X period(the productivity is doubled)
- Use of automated tools where ever is required
- Re-usable effort – work which has been done for the previous product can be used if similarities present b/w previous and present product.
- Use of resources with specific set of skills which exactly matches the requirements of the product, which reduces the time in training the resource. Recruiting people with desired skill set for current project is another option ,this is worth only if you expect to have more work on the near future on the same technology or skill sets.

➢ ELDC should take all this aspects into consideration to provide maximum productivity

7.. DIFFERENT PHASES OF EDLC

- A life cycle of product development is commonly referred as the "model"
- A simple model contains five phases
  - Requirement analysis
  - Design
  - Development and test

- Deployment and maintenance

- In real product development the phases given in this model may vary and can have sub models or model contain only certain important phases of classic model
- The no of phases involved in EDLC model depends on the complexity of the product.
- The following section describes each phase of classic ELDC model in detail

The following figure depicts the different phases in EDLC:

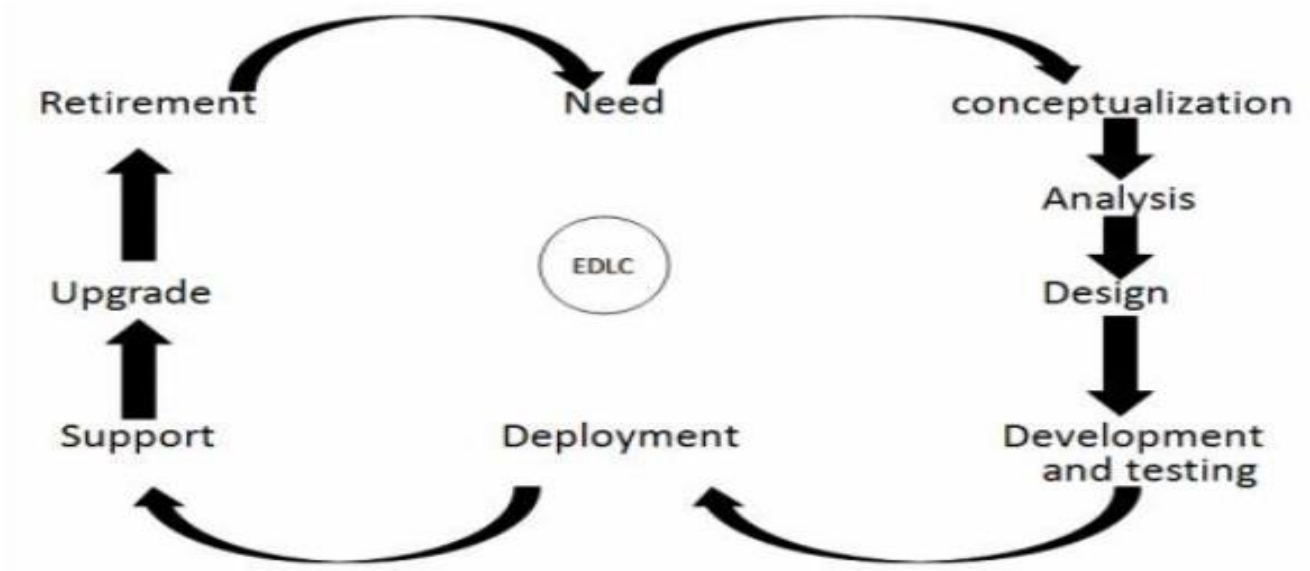


Figure : Phases of EDLC

7.1 Need

- The embedded product involves the output of a need

- The need may come from an individual or from the public or from a company.
- 'Need' should be articulated to initiate the Development Life Cycle; a 'Concept Proposal' is prepared which is reviewed by the senior management for approval.

Product development Need can be visualized in any one of the following three needs:

a. **New or Custom Product Development**.-The need for a product that doesn't exist in the market or a product which act as a competitor to an existing product in the current market will need to the development of a completely new product. Example of an embedded product which act as a competitor in market is Mobile handset

b. **Product Re-engineering.-**The embedded product market is competitive and dynamic. Re-engineering an existing product comes as a result of following needs.
   I. Change in Business requirement
   II. User interface Enhancements
   III. Technology Upgrades

c. **Product Maintenance.-**This need deals with providing technical support to the end user for an existing product in the market. The maintenance request may comes as the result of product nonfunctioning or failure

7.2 Conceptualisation

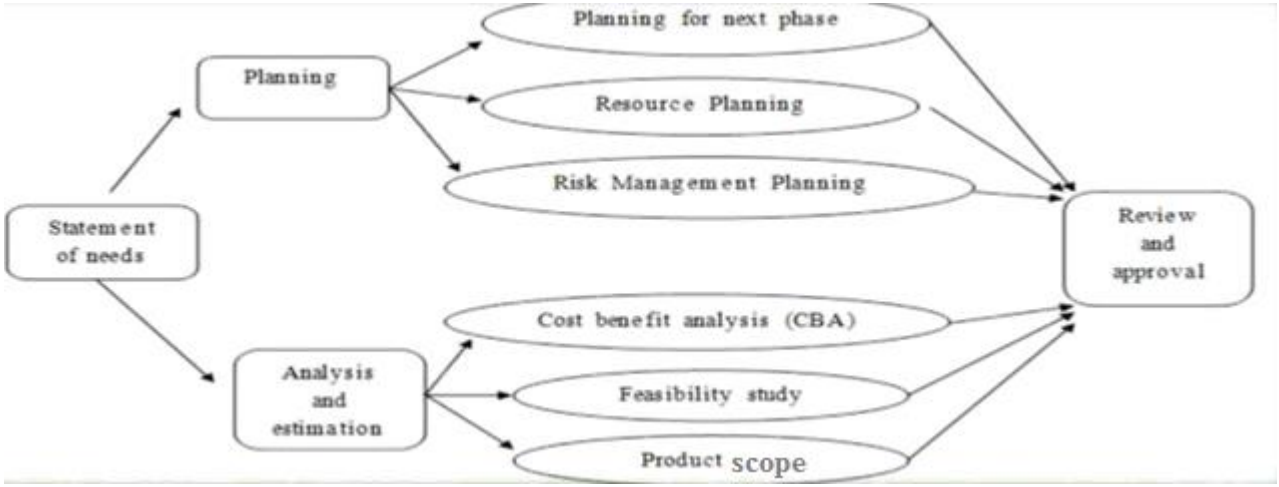
- Is the ***product concept development phase*** and it begins immediately after the concept proposal is formally approved.
- These phases define the scope of concept, performs cost benefit analysis and feasibility study and prepare project management and risk management plans.
- The conceptualization involves two types of activities, *planning activity and analysis and study activity*. This are performed to understand the opportunity of the product in the market.

- **Planning activity** Requires various plans to be developed first before development like
  - Planning for next phase
  - Resource planning
  - risk management planning
- **Analysis and Study activities i**nvolves

Feasibility Study : Examine the need and suggest possible solutions.

Cost Benefit Analysis (CBA): Revealing and assessing the total development cost and profit expected from the product. It is somewhat related to loss gain analysis in business. Some Common underlying principles are given below like Common Unit of measurement, market choice based benefit measurement, target end users etc..

Product Scope: Deals with what is scope(functionalities need to be considered) and what is not in scope.Should be properly documented for future reference



7.3. Analysis

- Requirement analysis phase start immediately after the documents submitted during the conceptualization phase is approved by the client / sponser of the project
- This phase is performed to develop a detailed functional model of the product under consideration
- The product is defined in detail with respect to the inputs, processes, outputs, and interfaces at a functional level.
- Emphasize on what functions must be performed by the product rather than how to perform those functions

The various activities performed during this phase..

I. **Analysis and Documentations:** This activity consolidates the business needs of the product under development.
Requirements that need to be addressed during this phase are

1. Functional Capabilities like performance
2. Operational and non-operational quality attribute
3. Product external interface requirements
4. User manuals θ Data requirements
5. Operational requirements
6. Maintenance requirements
7. General assumptions

II. Interface definition and documentation
This activity should clearly analyse on the physical interfaces as well as data exchange through this interfaces and should document it.

III. **Defining Test Plan and Procedures**: The various type of testing performed in a product development are:

1. Unit testing – Testing Individual modules

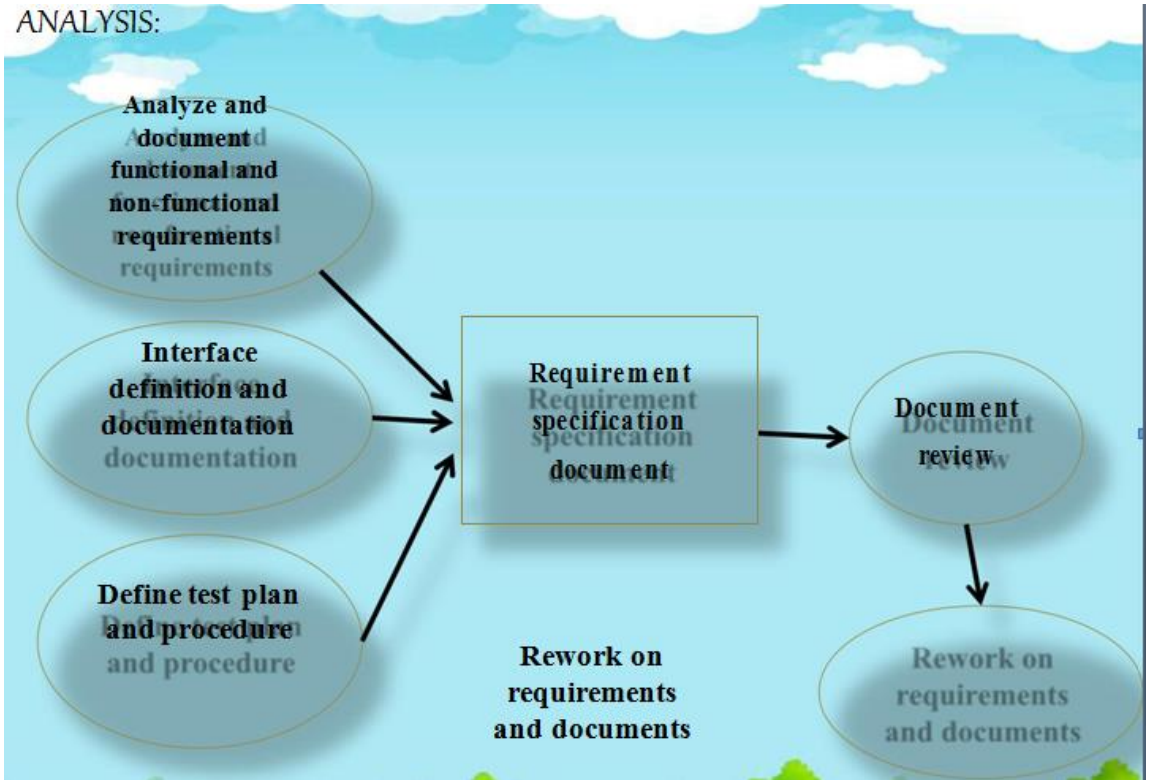
2. Integration testing – Testing a group of modules for required functionality

3. **System testing-** Testing functional aspects or functional requirements of the product after integration. Different tests are

- **Usability testing**: test the usability of the product
- **Load testing** :test the behavior of the product under different loading conditions
- **Security testing**: test the security aspects of the product
- **Scalability** testing: test the scalability aspects of the product
- **Sanity testing**: Superficial testing is performed to ensure that the product is functioning properly
- **Smoke testing**: ensures the crucial requirement of the product are functioning properly
- **Performance testing**: test the performance aspects of the product after integration
- **Endurance testing**:durability test of the product

4. User acceptance testing- Testing the product to meet the end user requirements.

At the end all requirements are put in a template suggested by a standard process model



7.4 Design

- The design phase identifies application environment and creates an overall architecture for the product.
- It starts with the Preliminary Design. It establishes the top level architecture for the product. On completion it resembles a 'black box' that defines only the inputs and outputs. The final product is called Preliminary Design Document (PDD).
- Once the PDD is accepted by the End User the next task is to create the 'Detailed Design'.

- It encompasses the Operations manual design, Maintenance Manual Design and Product Training material Design and is together called the 'Detailed Design Document'.

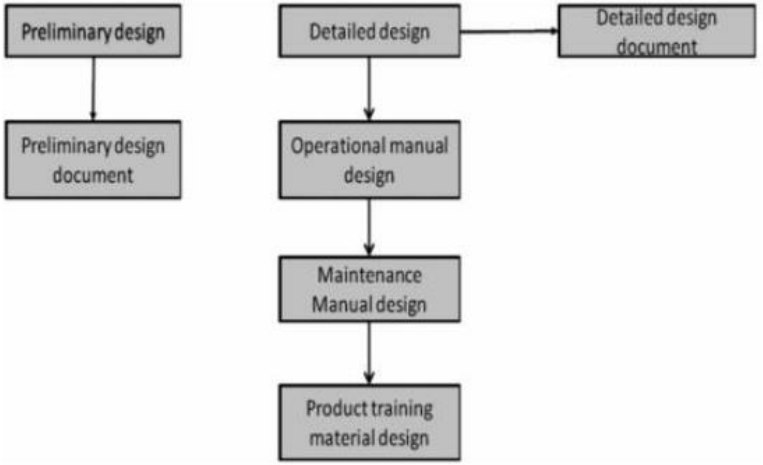
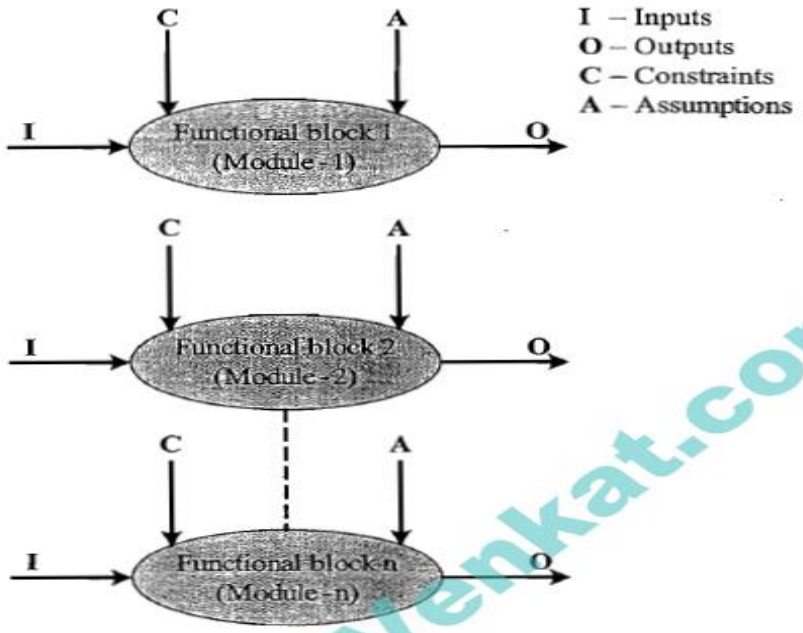


Fig.various activities involved in design phase



ig. 15.5 **Preliminary Design Illustration**

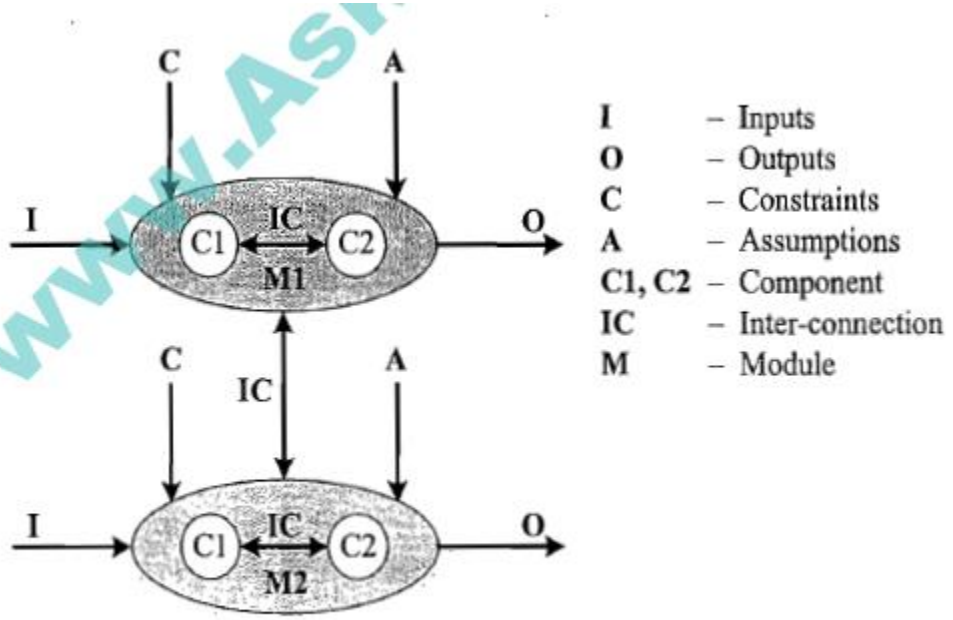


Fig. 15.6 **Detailed Design Illustration**

7.5 **Development and Testing**

- Development phase transforms the design into a realizable product.
- The detailed specification generated during the design phase is translated into hardware and firmware.
- The Testing phase can be divided into independent testing of firmware and hardware that is:

Unit testing

Integration testing

System testing

User acceptance testing

7.6 **Deployment**

- Deployment is the process of launching the first fully functional model of the product in the market.It is also known as First Customer Shipping (FCS).

Tasks performed during this phase are:

I. **Notification of Product Deployment:** Tasks performed here include:

Deployment schedule

Brief description about the product

Targeted end user

Extra features supported

Product support information

II. **Execution of training plan**

Proper training should be given to the end user top get them acquainted with the new product.

III. **Product installation**

Install the product as per the installation document to ensure that it is fully functional.

IV. **Product post Implementation Review**

After the product launch, a post implementation review is done to test the success of the product

7.7 Support

- The support phase deals with the operational and maintenance of the product in the production environment
- Bugs in the product may be observed and reported.
- The support phase ensures that the product meets the user needs and it continues functioning in the production environment.

Activities involved under support are

Setting up of a dedicated support wing: Involves providing 24 x 7 supports for the product after it is launched.

Identify Bugs and Areas of Improvement: Identify bugs and take measures to eliminate them.

7.8 Upgrades

- Deals with the development of upgrades (new versions) for the product which is already present in the market.
- Product upgrade results as an output of major bug fixes.
- During the upgrade phase the system is subject to design modification to fix the major bugs reported.

7.9 Retirement/Disposal

- The retirement/disposal of the product is a gradual process.
- This phase is the final phase in a product development life cycle where the product is declared as discontinued from the market.
- The disposal of a product is essential due to the following reasons
  - Rapid technology advancement
  - Increased user needs

8. ELDC APPROACHES(modeling the eldc)

Following are some of the different types of approaches that can be used to model embedded products.

1. Waterfall or Linear Model
2. Iterative/ Incremental or Fountain Model
3. Prototyping Model/Evolutionary Model
4. Spiral Model

Waterfall or Linear Model

- Linear or waterfall model is the one adopted in most of the olden systems.
- In this approach each phase of EDLC (Embedded Development Product Lifecycle) is executed in sequence.

- It establishes analysis and design with highly structured development phases.
- The execution flow is unidirectional.
- The output of one phase serves as the input of the next phase
- All activities involved in each phase are well planned so that what should be done in the next phase and how it can be done.
- The feedback of each phase is available only after they are executed.
- It implements extensive review systems To ensure the process flow is going in the right direction.
- One significant feature of this model is that even if you identify bugs in the current design the development process proceeds with the design.
- The fixes for the bug are postponed till the support phase.

Advantages

Product development is rich in terms of:

Documentation

Easy project management

Good control over cost & Schedule

Drawbacks

It assumes all the analysis can be done without doing any design or implementation

The risk analysis is performed only once.

The working product is available only at the end of the development phase

Bug fixes and correction are performed only at the maintenance/support phase of the life cycle.

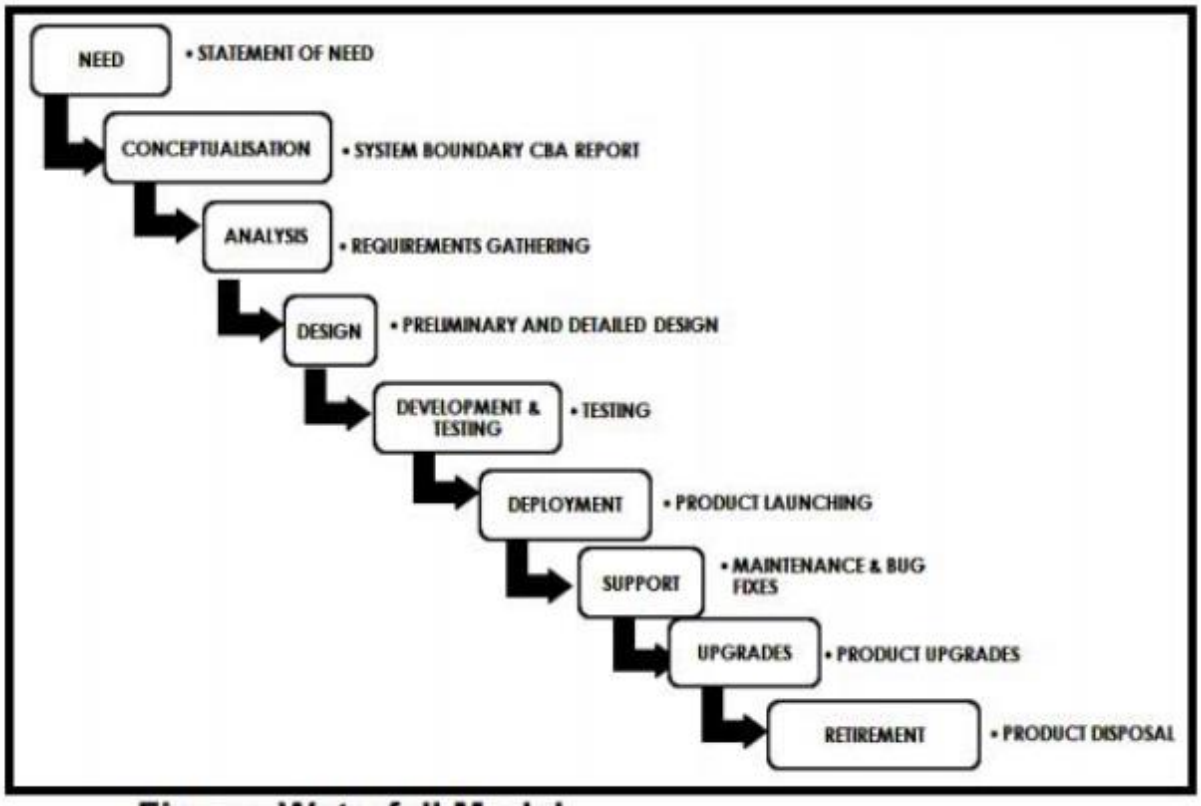
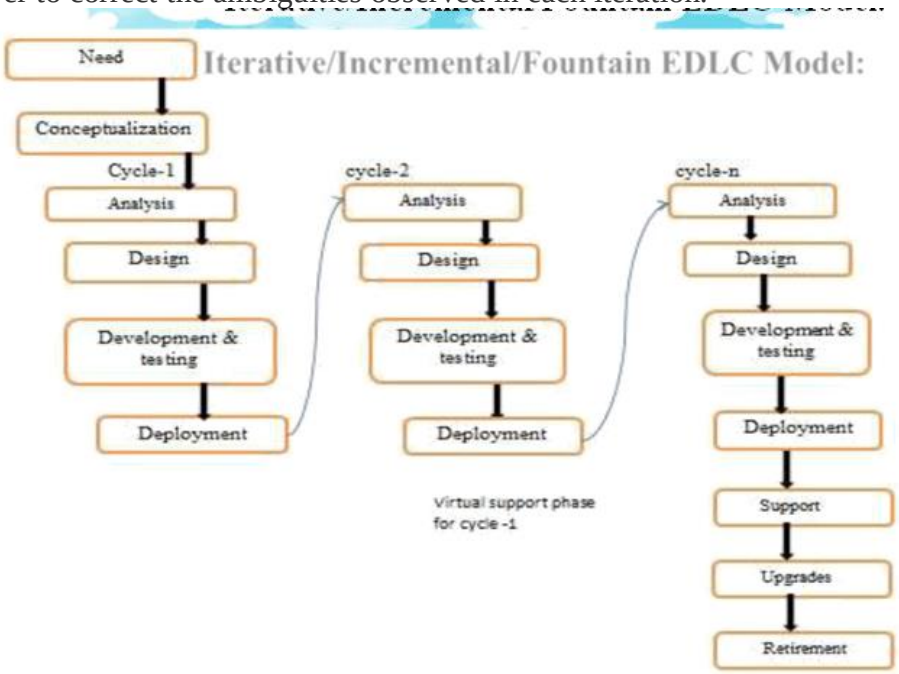


Figure: Waterfall Model

2. ITERATIVE/ INCREMENTAL OR FOUNTAIN MODEL

- Iterative and Incremental development is at the heart of a cyclic software development process developed in response to the weaknesses of the waterfall model.
- The iterative model is the repetitive process in which the Waterfall model is repeated over and over to correct the ambiguities observed in each iteration.



- The above figure illustrates the repetitive nature of the Iterative model.
- The core set of functions for each group is identified in the first cycle, it is then built, deployed and release. This release is called as the first release.
- Bug fixes and modification for first cycle carried out in second cycle.
- Process is repeated until all functionalities are implemented meeting the requirements.

Advantages

- Good development cycle feedback at each function/feature implementation
- Data can be used as reference for similar product development in future.
- More responsive to changing user needs.
- Provides working product model with at least minimum features at the first cycle.
- Minimized Risk
- Project management and testing is much simpler compared to linear model.
- Product development can be stopped at any stage with a bare minimum working product.

Disadvantages

- Extensive review requirement each cycle.
- Impact on operations due to new releases.
- Training requirement for each new deployment at the end of each development cycle.
- Structured and well documented interface definition across modules to accommodate changes

3. PROTOTYPING MODEL

- It is similar to iterative model and the product is developed in multiple cycles.
- The only difference is that, Prototyping model produces a refined prototype of the product at the end of each cycle instead of functionality/feature addition in each cycle as performed by the iterative model.
- There won't be any commercial deployment of the prototype of the product at each cycle's end.
- The shortcomings of the proto-model after each cycle are evaluated and it is fixed in the next cycle.
- After the initial requirement analysis, the design for the first prototype is made, the development process is started.
- On finishing the prototype, it is sent to the customer for evaluation.
- The customer evaluates the product for the set of requirements and gives his/her feedback to the developer in terms of shortcomings and improvements needed.
- The developer refines the product according to the customer's exact expectation and repeats the proto development process.
- After a finite number of iterations, the final product is delivered to the customer and launches in the market/operational environment
- In this approach the product undergoes significant evolution as a result of periodic shuttling of product information between the customer and developer

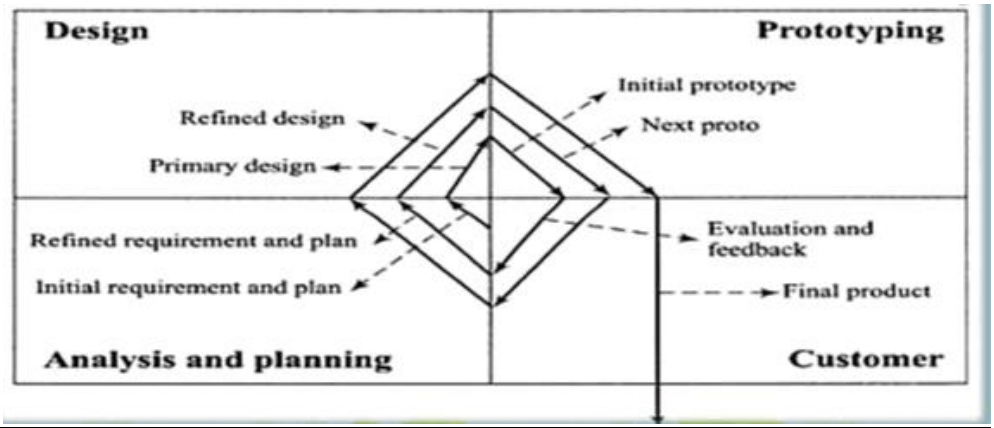
The prototyping model follows the approach-

Requirement definition

Proto-type development

Proto-type evaluation

Requirements refining



4. SPIRAL MODEL

- **Spiral model is developed by Barry Boehm in 1988.**
- The Product development starts with project definition and traverse through all phases of EDLC(Embedded Product Development Life Cycle).

The activities involved are:

I. Determine objectives, alternatives, constraints
II. Evaluate alternatives, identify and resolve risks
III. Develop and test
IV. Plan

It is a combines the concept of Linear Model and iterative nature of Prototyping Model.

Prototyping Model

- In prototyping after the requirement analysis the design for the prototype is made and development process is started.
- On finishing the prototype it is send to the customer for evaluation ie. Judgment.
- After customer evaluation for the product the feedback is taken from the customer in term of what improvement is needed.
- Then developer refines the product according to the customer expectation.

Linear Model

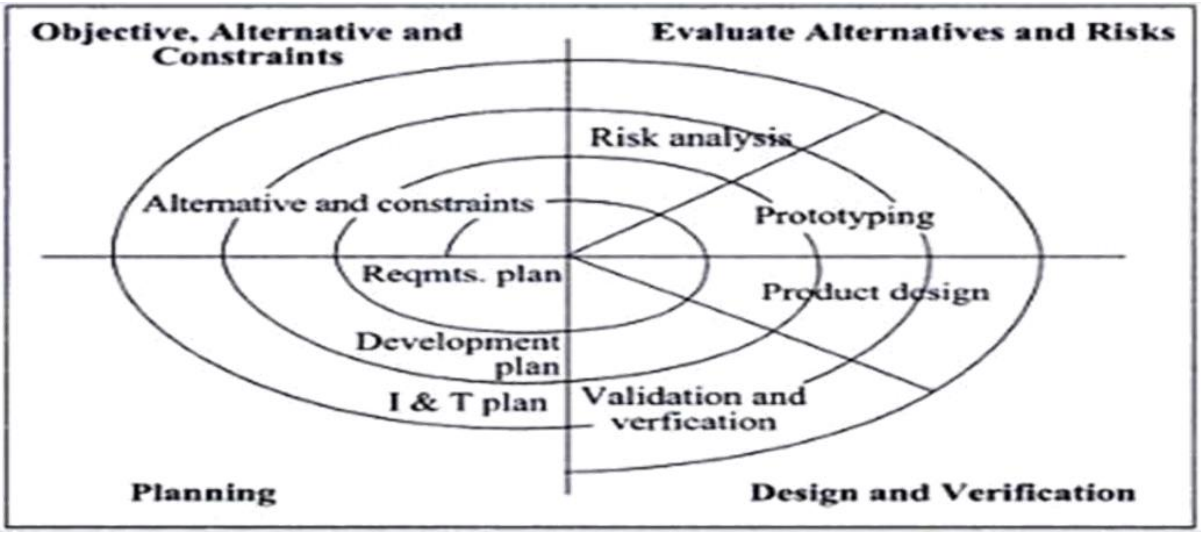
- Spiral Model contains the concept of linear model, having following type.

  Requirement

  Analysis

  Design

  Implementation



Requirement:

- This process is focused specifically on embedded software, to understand the nature of the software to be build and what are the requirement for the software.
- And the requirement for both the system & the software is documented & viewed to customer.

Analysis:

- Analysis is performed to develop a detailed functional module under consideration.
- The product is defined in detailed with respect to the input, processing & output.
- This phase emphasis on determining 'what function must be performed by the product' & how to perform those function.

Design:

- Product design deals with the entire design of the product taking the requirement into consideration.
- The design phase translates requirement into representation**.**

Implementation:

- In this process the launching of first fully functional model of the product in the market is done or handing over the model to an end user/client

- In this product modifications are implemented & product is made operational in production environment.

9. TRENDS IN EMBEDDED SYSTEMS

1. Processor Trends
2. Operating System Trends
3. Development Language Trends
4. Open Standards, Frameworks and alliances
5. Bottlenecks faced by Embedded Industry

9.1. PROCESSOR TRENDS

There have been tremendous advancements in the area of processor design.

Following are some of the points of difference between the first generation of processor/controller and today's processor/ controller.

Number of ICs per chip: Early processors had a few number of IC/gates per chip. Today's processors with Very Large Scale Integration (VLSI) technology can pack together ten of thousands of IC/gates per processor.

Need for individual components: Early processors need different components like brown out circuit, timers, DAC/ADC separately interfaced if required to be used in the circuit. Today's processors have all these components on the same chip as the processor.

Speed of Execution: Early processors were slow in terms of number of instructions executed per second. Today's processor with advanced architecture support features like instruction pipeline improving the execution speed.

Clock frequency: Early processors could execute at a frequency of a few MHz only. Today's processors are capable of achieving execution frequency in rage of GHz.

Application specific processor: Early systems were designed using the processors available at that time. Today it is possible to custom create a processor according to a product requirement.

Following are the major trends in processor architecture in embedded development.

9.1.1 System on Chip (SoC)

This concept makes it possible to integrate almost all functional systems required to build an embedded product into a single chip.

SoC are now available for a wide variety of diverse applications like Set Top boxes, Media Players, PDA, etc.

SoC integrate multiple functional components on the same chip thereby saving board space which helps to miniaturize the overall design.

9.1.2 Multicore Processors/ Chiplevel Multi Processor

This concept employs multiple cores on the same processor chip operating at the same clock frequency and battery.

Based on the number of cores, these processors are known as:

- Dual Core – 2 cores
- Tri Core – 3 cores
- Quad Core – 4 cores

These processors implement multiprocessing concept where each core implements pipelining and multithreading.

9.1.3 Reconfigurable Processors

It is a processor with reconfigurable hardware features.

Depending on the requirement, reconfigurable processors can change their functionality to adapt to the new requirement. Example: A reconfigurable processor chip can be configured as the heart of a camera or that of media player.

These processors contain an Array of Programming Elements (PE) along with a microprocessor. The PE can be used as a computational engine like ALU or a memory element.

9.2 EMBEDDED OPERATING SYSTEM TRENDS

The advancements in processor technology have caused a major change in the Embedded Operating System Industry.

There are lots of options for embedded operating system to select from which can be both commercial and proprietary or Open Source.

Virtualization concept is brought in picture in the embedded OS industry which replaces the monolithic architecture with the microkernel architecture.

This enables only essential services to be contained in the kernel and the rest are installed as services in the user space as is done in Mobile phones.

Off the shelf OS customized for specific device requirements are now becoming a major trend.

9.3 DEVELOPMENT LANGUAGE TRENDS

There are two aspects to Development Languages with respect to Embedded Systems Development

Embedded Firmware

It is the application that is responsible for execution of embedded system.

It is the software that performs low level hardware interaction, memory management etc on the embedded system.

Embedded Software

It is the software that runs on the host computer and is responsible for interfacing with the embedded system.

It is the user application that executes on top of the embedded system on a host computer.

Early languages available for embedded systems development were limited to C & C++ only. Now languages like Microsoft C$, ASP.NET, VB, Java, etc are available.

9.3.1 Java for embedded development

Java is not a popular language for embedded systems development due to its nature of execution.

Java programs are compiled by a compiler into bytecode. This bytecode is then converted by the JVM into processor specific object code.

During runtime, this interpretation of the bytecode by the JVM makes java applications slower that other cross compiled applications.

This disadvantage is overcome by providing in built hardware support for java bytecode execution.

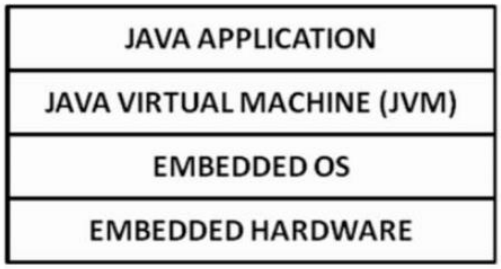


Figure: Java based Embedded Application Development

Another technique used to speed up execution of java bytecode is using Just In Time (JIT) compiler. It speeds up the program execution by caching all previously executed instruction.

Following are some of the disadvantage of Java in Embedded Systems development:

- For real time applications java is slow
- Garbage collector of Java is non-deterministic in behavior which makes it not suitable for hard real time systems.
- Processors need to have a built in version of JVM
- Those processors that don't have JVM require it to be ported for the specific processor architecture.
- Java is limited in terms of low level hardware handling compared to C and C++
- Runtime memory requirement of JAVA is high which is not affordable by embedded systems.

9.3.2 .NET CF for embedded development

It stands for .NET Compact Framework.

.NET CF is a replacement of the original .NET framework to be used on embedded systems.

The CF version is customized to contain all the necessary components for application development.

The Original version of .NET Framework is very large and hence not a good choice for embedded development.

The .NET Framework is a collection of precompiled libraries.

Common Language Runtime (CLR) is the runtime environment of .NET. It provides functions like memory management, exception handling, etc.

Applications written in .NET are compiled to a platform neutral language called Common Intermediate Language (CIL).

For execution, the CIL is converted to target specific machine instructions by CLR.

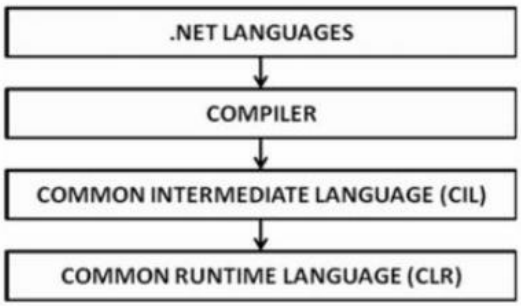


Figure: .NET based Embedded Application Development

9.4.OPEN STANDARDS, FRAMEWORKS AND ALLIANCES

Standards are necessary for ensuring interoperability. With diverse market it is essential to have formal specifications to ensure interoperability.

Following are some of the popular strategic alliances, open source standards and frameworks specific to the mobile handset industry.

9.4.1 Open Mobile Alliance (OMA)

- It is a standard body for creating open standards for mobile industry.
- OMA is the Leading Industry Forum for Developing Market Driven – Interoperable Mobile Service Enablers
- OMA was formed in June 2002 by the world's leading mobile operators, device and network suppliers, information technology companies and content and service providers.
- OMA delivers open specifications for creating interoperable services that work across all geographical boundaries, on any bearer network. OMA's specifications support the billions of new and existing fixed and mobile terminals across a variety of mobile

networks, including traditional cellular operator networks and emerging networks supporting machine-to-machine device communication.

- OMA is the focal point for the development of mobile service enabler specifications, which support the creation of interoperable end-to-end mobile services.

Goals of OMA

- Deliver high quality, open technical specifications based upon market requirements that drive modularity, extensibility, and consistency amongst enablers to reduce industry implementation efforts.
- Ensure OMA service enabler specifications provide interoperability across different devices, geographies, service providers, operators, and networks; facilitate interoperability of the resulting product implementations.
- Be the catalyst for the consolidation of standards activity within the mobile data service industry; working in conjunction with other existing standards organizations and industry fora to improve interoperability and decrease operational costs for all involved.

- Provide value and benefits to members in OMA from all parts of the value chain including content and service providers, information technology providers, mobile operators and wireless vendors such that they elect to actively participate in the organization.

9.4.2 Open Handset Alliance (OHA)

- The Open Handset Alliance is a group of 84 technology and mobile companies who have come together to accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience. Together they have developed Android™, the first complete, open, and free mobile platform and are committed to commercially deploy handsets and services using the Android Platform.

- Members of OHA include mobile operators, handset manufacturers, semiconductor companies, software companies, and commercialization companies.

9.4.3 Android

- Android is an operating system based on the Linux kernel, and designed primarily for touchscreen mobile devices such as smartphones and tablet computers.

- Initially developed by Android, Inc., which Google supported financially and later bought in 2005, Android was unveiled in 2007 along with the founding of the Open Handset Alliance: a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices.

- The first publicly-available Smartphone to run Android, the HTC Dream, was released on October 18, 2008

9.4.4Openmoko

Openmoko is a project to create a family of open source mobile phones, including the hardware specification and the operating system.

The first sub-project is Openmoko Linux, a Linux-based operating system designed for mobile phones, built using free software.

The second sub-project is developing hardware devices on which Openmoko Linux runs.

10 . Bottlenecks faced by Embedded Industry

Following are some of the problems faced by the embedded devices industry:

Memory Performance

The rate at which processors can process may have increased considerably but rate at which memory speed is increasing is slower.

Lack of Standards/ Conformance to standards

Standards in the embedded industry are followed only in certain handful areas like Mobile handsets.

There is growing trend of proprietary architecture and design in other areas.

Lack of Skilled Resource

Most important aspect in the development of embedded system is availability of skilled labor. There may be thousands of developers who know how to code in C, C++, Java or .NET but very few in embedded software.