EC8791 EMBEDDED AND REAL TIME SYSTEMS

UNIT I INTRODUCTION TO EMBEDDEDSYSTEM DESIGN

Complex systems and microprocessors- Embedded system design process -Design example: Model train controller- Design methodologies- Design flows - Requirement Analysis - Specifications-System analysis and architecture design - Quality Assurance techniques - Designing with computing platforms - consumer electronics architecture - platform-level performance analysis.

Unit.I

Embedded and Real time Systems

1. Complex Systems and Microprocessors:

Embedded computer system is defined as any device that includes a programmable computer but is not itself intended to be a general purpose computer. Thus, a PC is not itself an embedded computing system although PCs are often used to build embedded computing systems.

This means that embedded computing system design is a useful skill for many types of product design.

Automobiles, cell phones and even household appliances make extensive use of microprocessors.

Computer engineering like mechanical design or thermodynamics is a fundamental discipline that can be applied in different domains. But embedded computing system design does not stand alone.

The most important and sophisticated use of microprocessors in automobile was to control the engine determining when spark plugs fire, controlling the fuel / ~~air~~ air mixture and so on. There was a trend in general, electronic devices could be used to replace the mechanical distributor.

The combination of low fuel consumption and low emissions is very difficult to achieve; to meet these goals automobile manufacturers turned to sophisticated control algorithms that could be implemented only with microprocessors.

Microprocessors usually classified by their word size.

- An 8-bit microcontroller is designed for low cost applications and includes on board memory and I/O devices.
- A 16-bit microcontroller is often used for more sophisticated applications that may require either longer word length or off chip I/O & memory.
- A 32-bit RISC microprocessor offers very high performance for computation-intensive applications.

BMW 850i brake and stability control system:

The BMW 850i was introduced with a sophisticated system for controlling the wheels of the car. An antilock brake system (ABS) reduces skidding by pumping the brakes.

An automatic stability control (ASC+T) system intervenes with the engine during maneuvering to improve the car's stability. These systems require inputs from and output to automobile.

The ABS system uses sensors on each wheel to measure the speed of the wheel, to prevent the wheels from skidding.

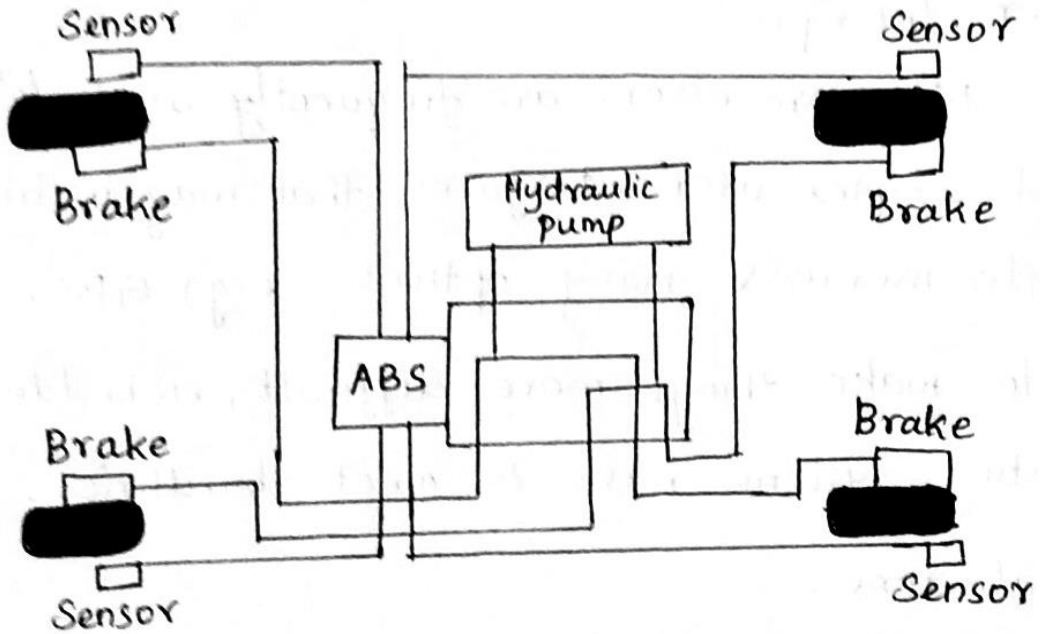


fig: Anti-lock brake system

The engine and control management units include the electronically controlled throttle, digital engine management and electronic transmission control.

i) Functionality is important in both general purpose computing and embedded computing.

* Complex algorithms :

The operations performed by the micro-processor may be very sophisticated. For example, the microprocessor that controls an automobile engine must perform complicated filtering functions such as minimizing pollution & fuel utilization.

* User Interface :

Microprocessors are frequently used to control complex user interfaces that may include multiple menus & many options. e.g) GPS.

ii) To make things more difficult, embedded computing systems have to meet deadlines :

* Real time :

Many embedded computing systems have to perform in real time. In some cases, the

failure in real time is unsafe and can even endanger lives. In other cases, it create unhappy customers e.g) can result in scarambled pages.

* Multirate:

Not only must operations be completed by deadlines but many embedded computing systems have several real time activities going on at same time. They may simultaneously control some operations that run at slow rates and others at high rate. e.g) Multimedia applications

iii) Cost of various sorts are also very important:

* Manufacturing cost:

The total cost of building the system is very important. It is determined by including the type of microprocessor used, amount of memory required and types of I/O devices.

* Power and Energy:

Power consumption directly affects the cost of hardware ie) large power supply needed.

Energy consumption affects battery life which is important in desktop applications.

2. Embedded System Design Process:

First, a design to ensure that we have done everything we need to do such as optimizing performance or performing functional tests.

Second, it allows us to develop computer aided design tools.

Third, a design methodology makes it much easier for members of a design team to communicate.

Design from the top-down concludes with concrete details. The alternative is a bottom-up view in which we start with components to build a system.

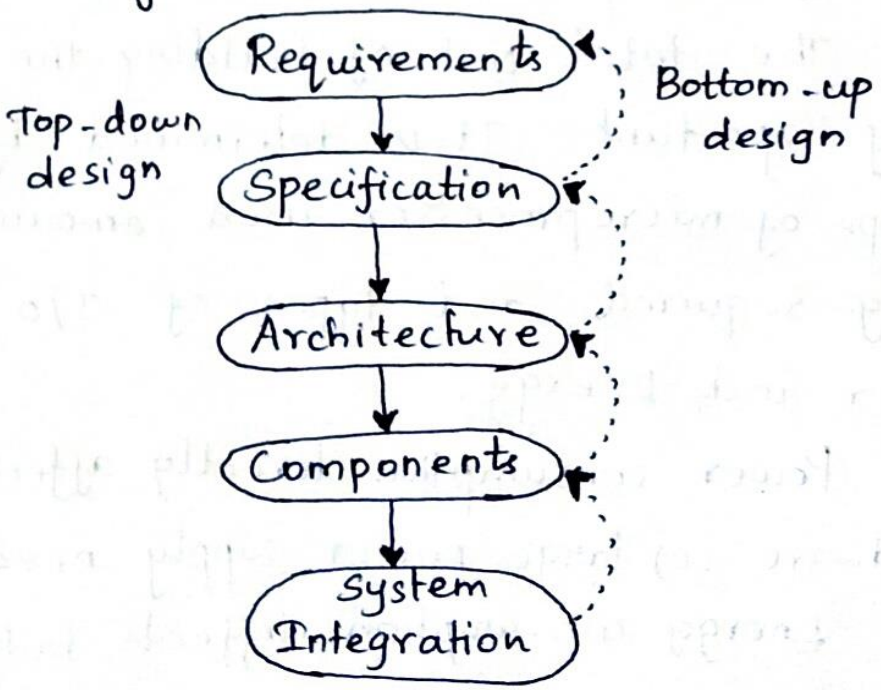


fig: Major levels of abstraction in the design process

To consider the major goals of the design:

- Manufacturing cost
- Performance
- Power consumption

Requirements:

Requirements may be functional or non-functional. Typical nonfunctional requirements include:

- Performance: It may be a combination of soft performance metrics such as approximate time to perform a user level function and hard deadlines by which a particular operation may be completed.
- Cost: Cost typically has two major components :- manufacturing cost and nonrecurring engineering (NRE) cost.
- Physical size and weight: The physical aspects of the final system can vary greatly depending upon the application. A handheld device typically has tight requirements on both size and weight than can ripple through entire design.

- Power Consumption: Power is important in battery powered systems and other applications as well.

Name
Purpose
Inputs
Outputs
Functions
Performance
Manufacturing cost
Power
Physical size & weight

Sample requirements form

Name:

This is simple but helpful. Giving a name to the project not only simplifies talking about it to other people but can crystallize purpose of machine.

Purpose:

This should be a brief one or two line description of what the system is supported to do.

Inputs and Outputs:

It describes the buttons, analog / digital converters, video displays.

Performance:

The computations must be performed within a certain time frame. It is essential that the performance requirements be identified early since they must be carefully measured.

Manufacturing cost:

This includes primarily the cost of the hardware components. Cost has a substantial influence on architecture ie) A machine meant to sell at $10 most likely has a very different internal structure than a $100 system.

Power:

The most important decision is whether the machine will be battery powered or plugged into the wall.

Physical size & weight:

Give some indication of the physical size of the system to help guide certain architectural decisions. A desktop machine has much more flexibility in the components used than, for example, a lapel mounted voice recorder.

Specifications:

The specification should be understandable enough so that someone can verify that it meets system requirements and overall expectations of the customer. A specification of the GPS system would include several components:

→ Data received from the GPS satellite constellation.

→ Map data

→ User Interface

→ Operations that must be performed to satisfy customer requests.

→ Background actions required to keep the system running, such as operating the GPS receiver.

Architecture design:

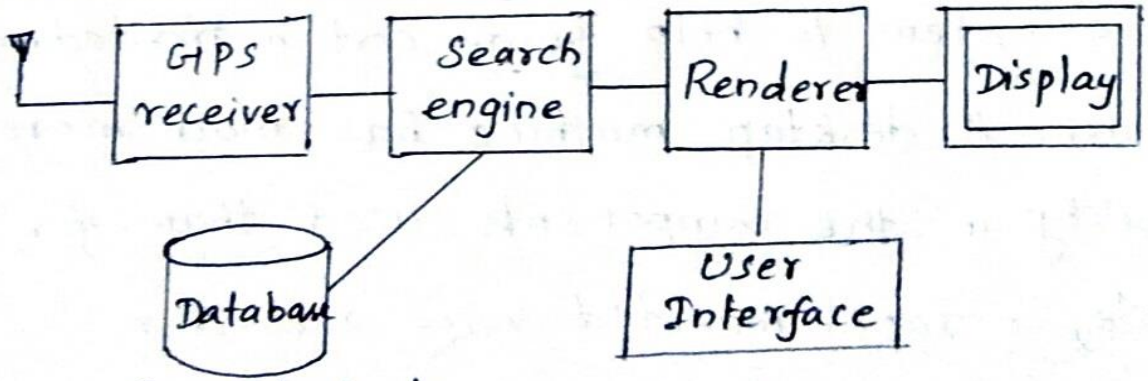


fig: Block diagram for the moving map

Designing hardware & software Components:

To design some components ourself. When creating embedded software modules, make use of your expertise to ensure that the system runs properly in real time and does not take up more memory space than is allowed.

System Integration:

System Integration is difficult because it usually uncovers problems. It is often hard to observe the system in sufficient detail to determine exactly what is wrong with debugging facilities on desktop systems.

3. Design Example : Model Train Controller

The user sends messages to the train with a control box attached to the tracks.

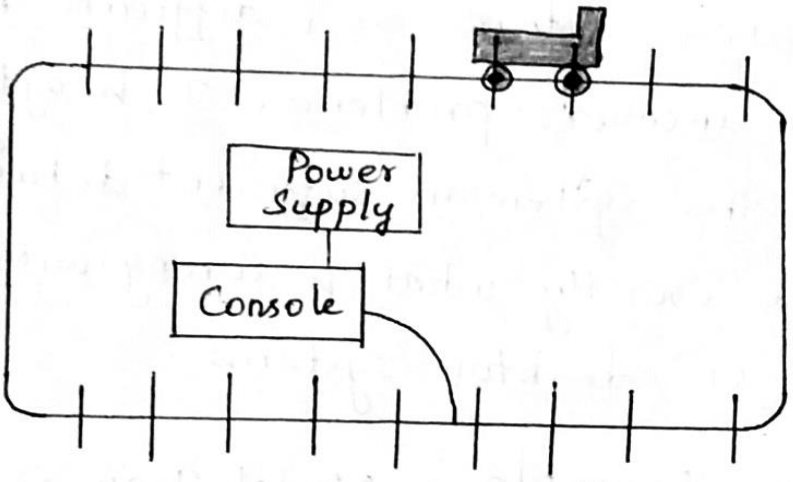
The control box may have controls such as throttle, emergency stop button and so on.

Since the train receives its electrical power from the two rails of the track, the control box can send signals to the train.

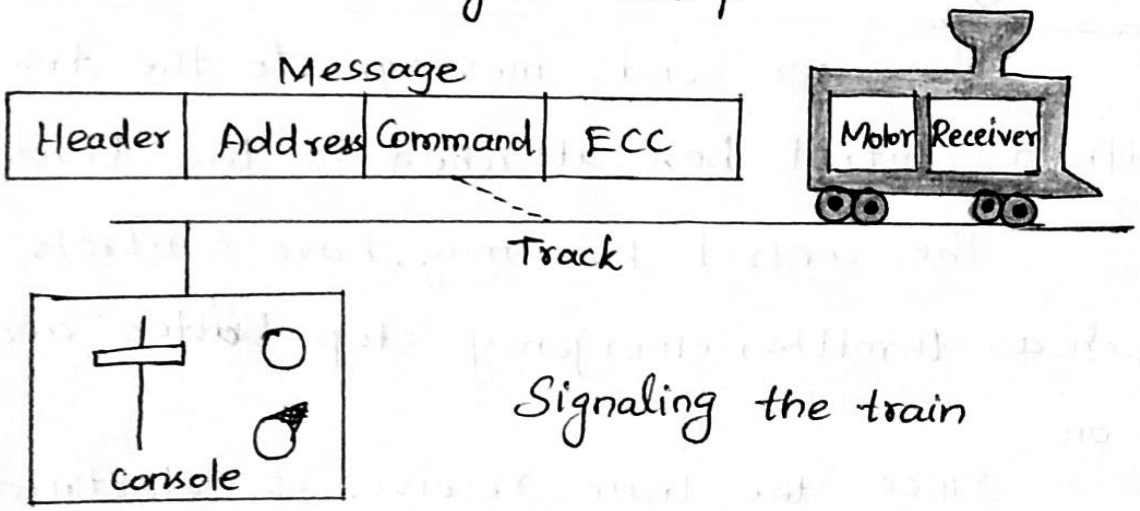
The control panel sends packets over the tracks to the receiver on the train.

Each packet includes an address so that the console can control speed of several trains on same track; the packet also includes an Error Correction Code (ECC) to guard against transmission errors.

This is a one way communication.



System setup



Signaling the train

fig: A model train control system

Requirements:

The console shall be able to control up to eight trains on a single track.

The speed of each train shall be controllable by a throttle to atleast 63 different levels in each direction.

There shall be an inertia control that shall allow the user to adjust the responsiveness of train.

The inertia control will provide at least eight different levels.

There shall be an emergency stop button.

An error detection scheme will be used to transmit messages.

Name	Model train controller
Purpose	Control speed of upto 8 model trains
Inputs	Throttle, inertia setting, stop button, train no
Outputs	Train control signals
Functions	Set engine speed based upon inertia settings, respond to emergency stop
Performance	Can update train speed 10 times / second
Manufacturing cost	$50
Power	10 W
Physical size & weight	~~Contr~~ Console should be comfortable for two hands, approximate size of std. keyboard; weight <2 pounds

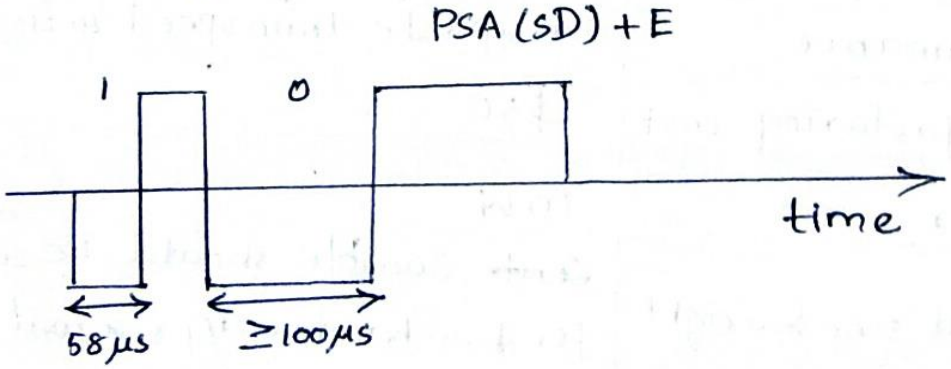
DCC:

The Digital Command Control (DCC) standard (http://www.nmra.org/standards/DCC/standards_rps/DCCStds.html) was created by the National Model Railroad Association to support digitally controlled model trains.

The DCC standard is given in two documents:

- Standard S-9.1, the DCC Electrical standard, describes how bits are encoded on the rails for transmission.
- Standard S-9.2, the DCC Communication Standard, describes the packets that carry information.

The data signal swings between two voltages around the power supply voltage. Bits are encoded in the time between transitions, not by voltage levels.



Bit encoding in DCC

Regular expression:

P is the preamble which is a sequence of atleast 10 1bits.

S is the packet start bit; it is a 0bit.

A is an address which is eight bits long. The addresses 00000000, 11111110 & 11111111 are reserved.

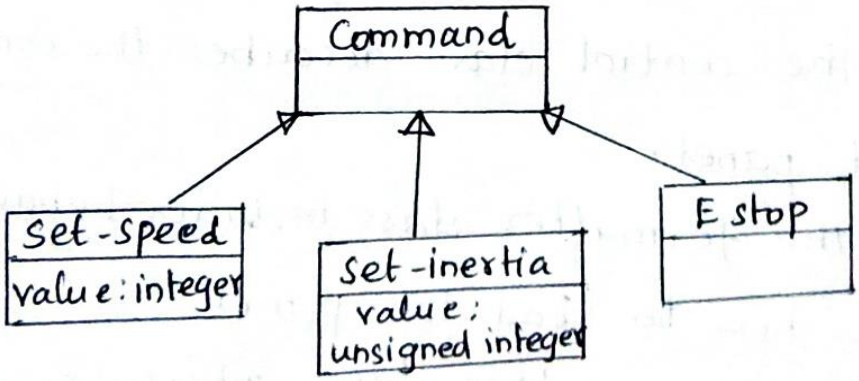
S is the data byte start bit like the packet start bit is a 0.

D is the data byte which includes 8 bits.

E is a packet end bit which is a 1 bit.

Conceptual specification:

A conceptual specification allows us to understand the system a little better. A train control system turns commands into packets. Commands and packets may not be generated in a 1 to 1 ratio.



Class diagram for the train controller messages

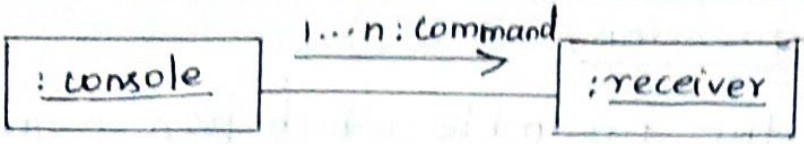


fig: UML collaboration diagram for major subsystems of the train controller system.

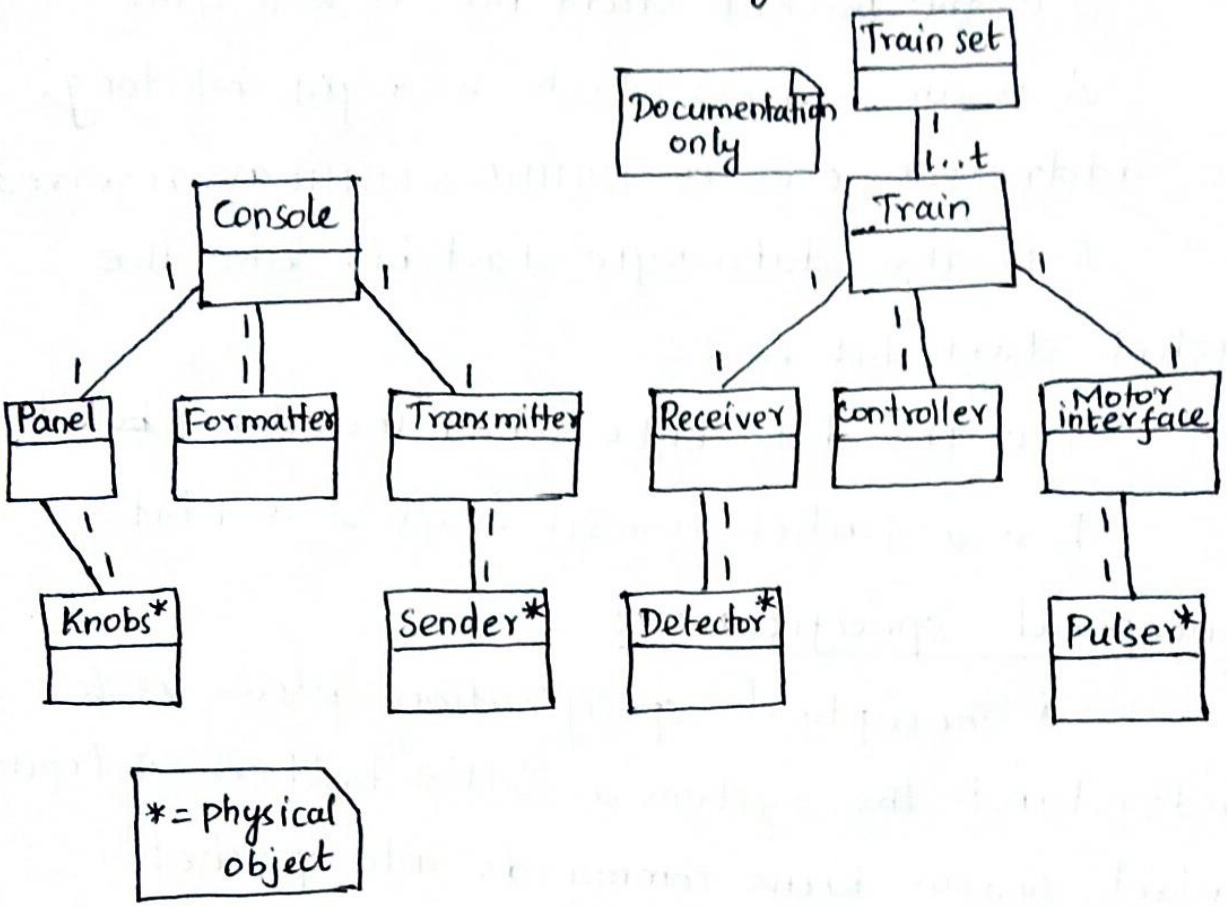


fig: A UML class diagram for the train controller showing the composition of the subsystem.

- The control class describes the command's unit front panel.
- The formatter class includes behaviors that know how to read the panel.
- The transmitter class interfaces to analog electronics to send the message along the track.

• Knobs* describes the actual analog knobs, buttons and levers on the control panel.

• Sender* describes the analog electronics that send bits along the track.

• The Receiver class knows how to turn the analog signals on the track into digital form.

• The motor interface class defines how to generate the analog signals required to control the motor.

• Detector* detects analog signals on the track and converts them into digital form.

• Pulser* turns digital commands into the analog signals required to control the motor speed.

4. Design Methodologies:

Design methodologies is important because without it, we can't reliably deliver the products want to create.

i) Time-to-market:

Customers always want new features. The product that comes out first can win the market. The profitable market life for some

products is 3 to 6 months. After that you will never make money. For example, calculators are sold just before school starts. After that you have to wait for another sales season.

ii) Design Cost:

Many customer products are very cost sensitive. Industrial buyers are also increasingly concerned about cost. The costs are distinct from manufacturing cost, engineer's salaries, computers used in design and so on. Design costs can also be important for high volume consumer devices when time to market pressures cause teams to swell in size.

iii) Quality:

Customers not only want their products fast and cheap, they also want them to be right. A design methodology that cranks out shoddy products will soon be forced out of the marketplace. Correctness, reliability and usability must be explicitly addressed from the beginning of the design job to obtain a high quality product at the end.

5. Design Flows :

A design flow is a sequence of steps to be followed during a design. Some of the steps can be performed by tools such as compilers or CAD systems, other steps can be performed by hand.

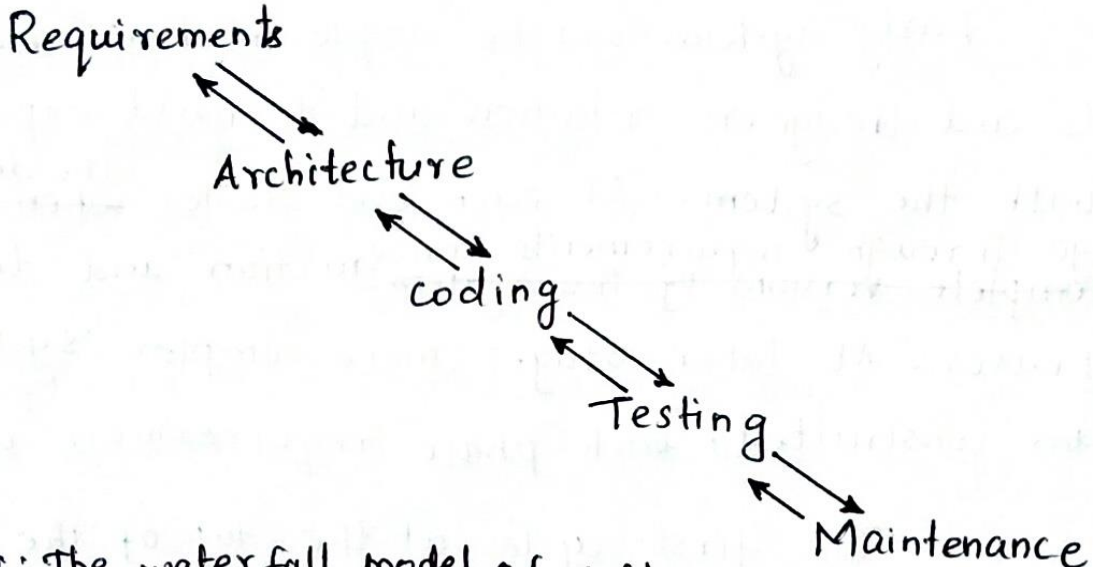


fig: The waterfall model of software development

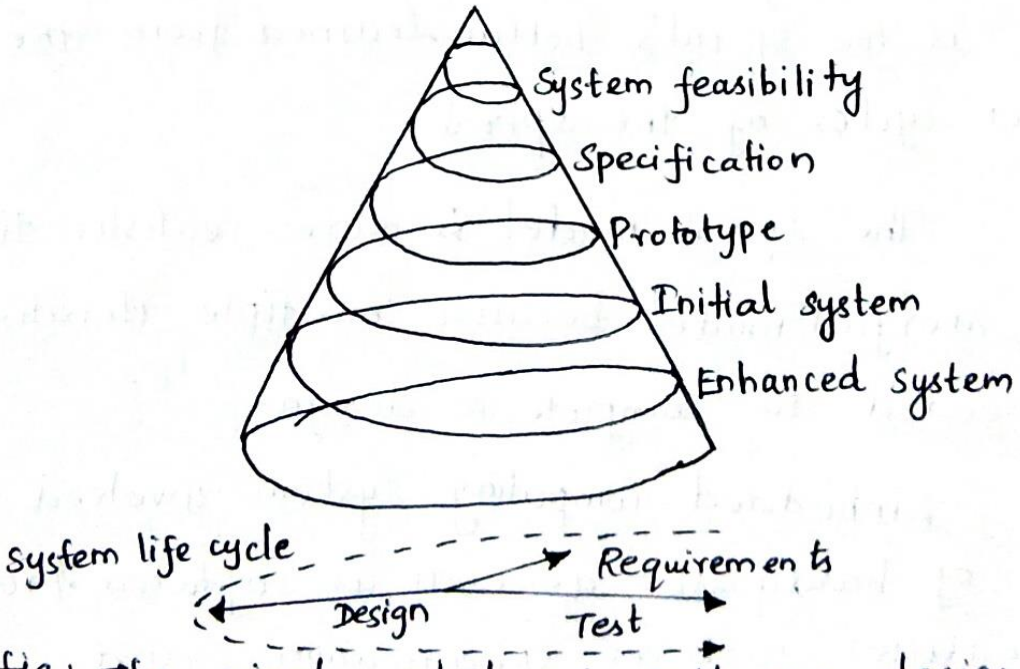


fig: The spiral model of software design

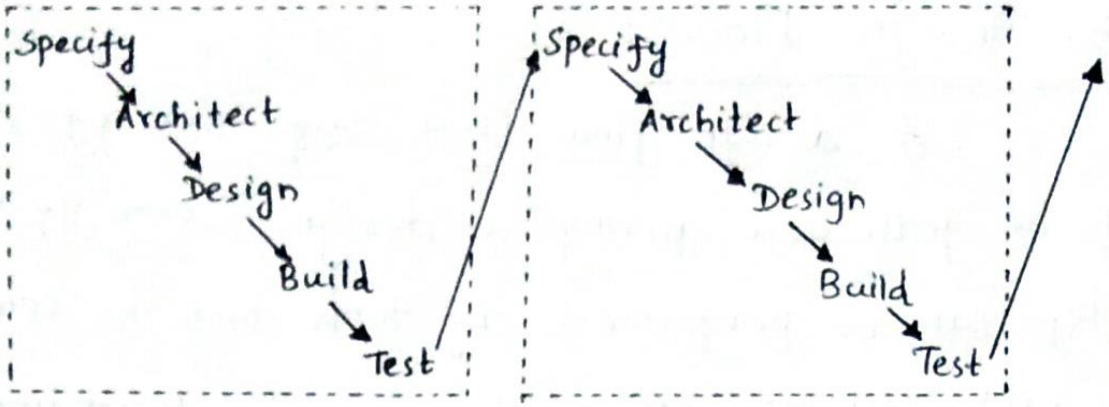


fig: A successive refinement development model

Early systems will be simple mock ups constructed to aid designers intuition and to build experience with the system. At each level stages ~~when more complete versions of the system~~ the designers go through requirements, construction and testing phases. At later stages, more complex systems are constructed, each phase requires more work.

The first cycles at the top of the spiral are very small and short while the final cycles at the spiral's bottom learned from the earlier cycles of the spiral.

The spiral model is more realistic than the waterfall model because multiple iterations are needed to complete a design.

Embedded computing system involved the design of hardware as well as software. Front end activities such as specification and

architecture simultaneously consider hardware and software aspects. Lastly, back end integration and testing consider the entire system.

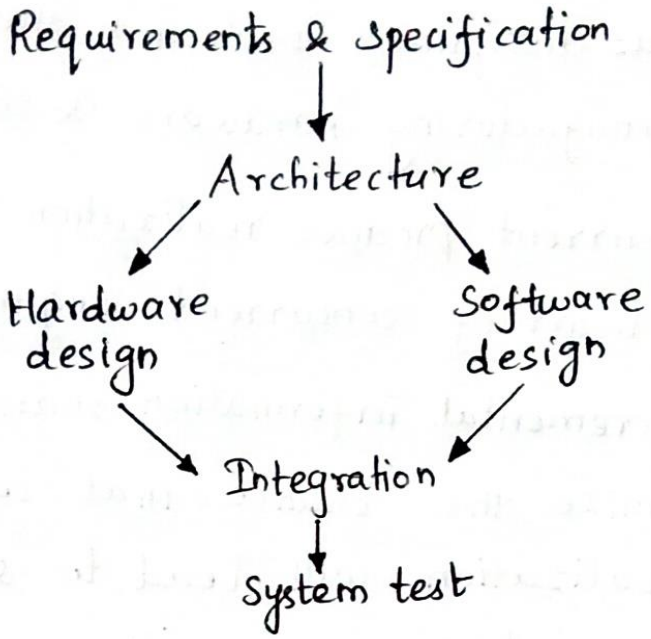


fig: A simple hardware / software design methodology

The design flow follows the levels of abstraction in the system from complete system design flows at the most abstract to design flows for individual components.

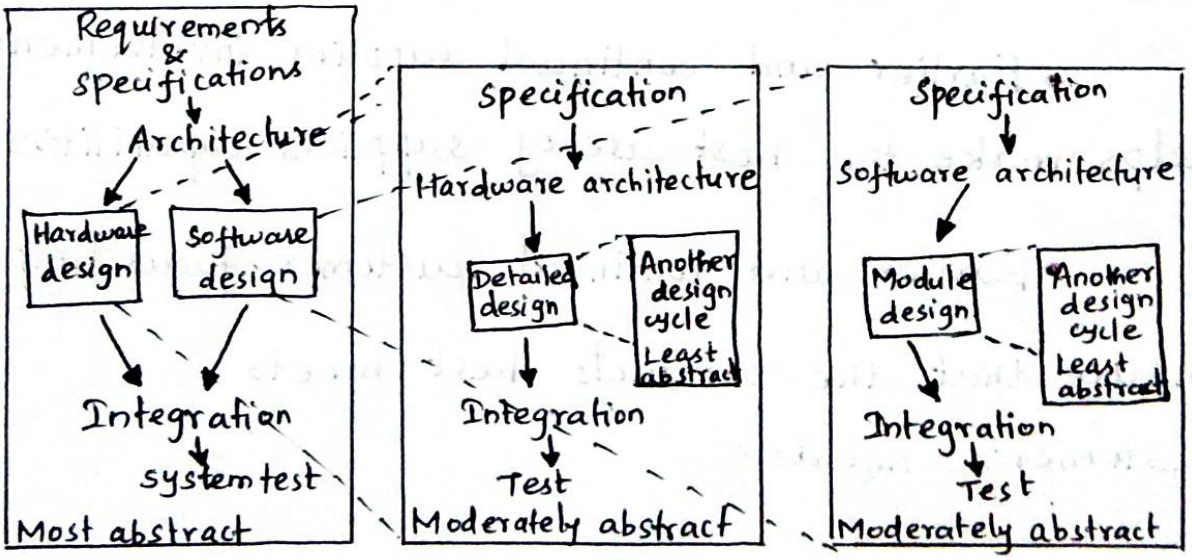


fig: A hierarchical design flow for an embedded system

Concurrent engineering efforts are comprised of several elements.

- Cross functional teams include members from various disciplines involved in the process including manufacturing hardware & software design.
- Concurrent product realization process activities are at the heart of concurrent engineering.
- Incremental information sharing and use helps minimize the chance that concurrent product realization will lead to surprise. Cross functional teams are important to the effective sharing of information in a timely fashion.
- Integrated project management ensure that someone is responsible for the entire project.
- Earlier and continual supplier involvement helps make the best use of suppliers capabilities.
- Earlier and continual customer focus helps ensure that the products best meets customer's needs.

6. Requirements Analysis:

The overall goal of creating a requirements document is effective communication between the customers and the designers. A good set of requirements should meet several tests:

i) Correctness:

The requirements should not mistakenly describe what the customer wants. Part of correctness is avoiding over requiring.

ii) Unambiguousness:

The requirements document should be clear and have only one plain language interpretation.

iii) Completeness:

All requirements should be included.

iv) Verifiability:

There should be a cost effective way to ensure that each requirement is satisfied in the final product.

v) Consistenty:

One requirement should not contradict another requirement.

vi) Modifiability:

The requirements document should be structured so that it can be modified to meet changing requirements without losing consistency verifiability and so on.

vii) Traceability:

Each requirement should be traceable

- backward from the requirements to know why each requirement exists.
- forward from documents created before the requirements.
- forward to understand how each requirement is satisfied in the implementation.
- backward from the implementation to know which requirements they were intended to satisfy.

If the product is a continuation of series, then many of the requirements are well understood.

7. Specifications:

State based specification that introduced some important concepts.

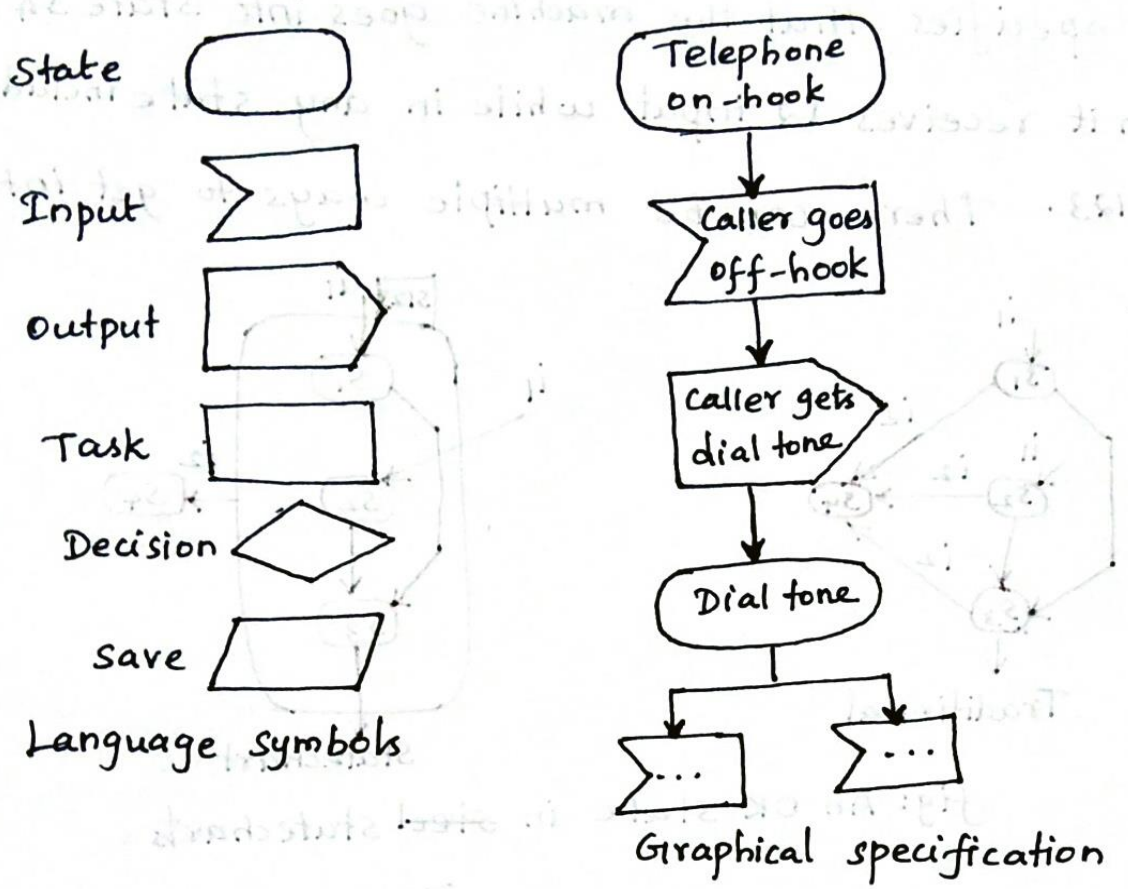


fig: The SDL Specification language

The statechart notation uses an event-driven model. Statecharts allows states to be grouped together to show common functionality. There are two basic groupings OR and AND.

OR state by comparing a traditional state transition diagram with a statechart described via an OR state.

The state machine specifies the machine goes to state S_4 from any of S_1, S_2 or S_3 when input i_2. A single transition out of the OR states $S123$ specifies that the machine goes into state S_4 when it receives i_2 input while in any state included in $S123$. There can be multiple ways to get into $S123$.

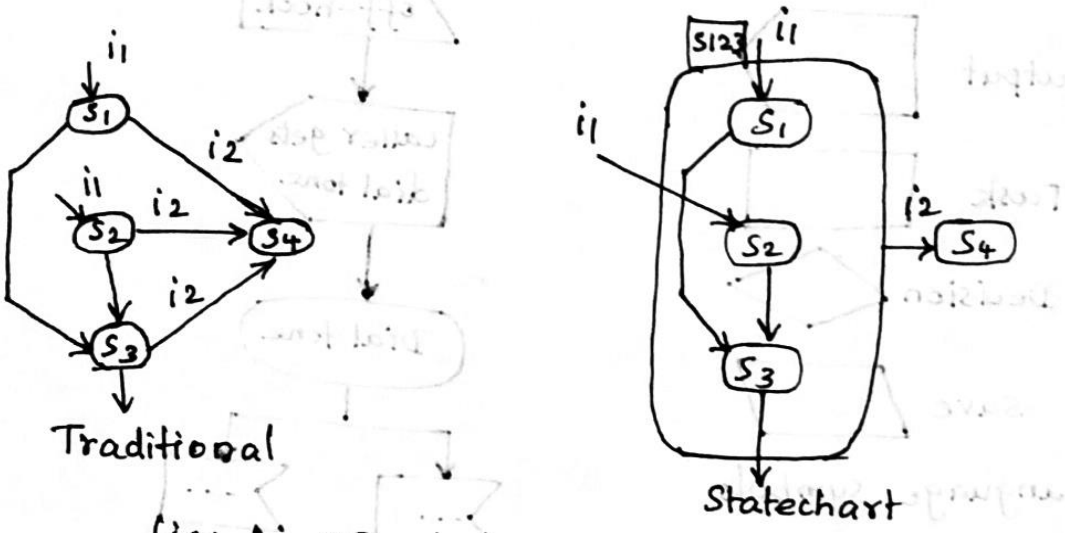


fig: An OR state in ~~steel~~ statecharts.

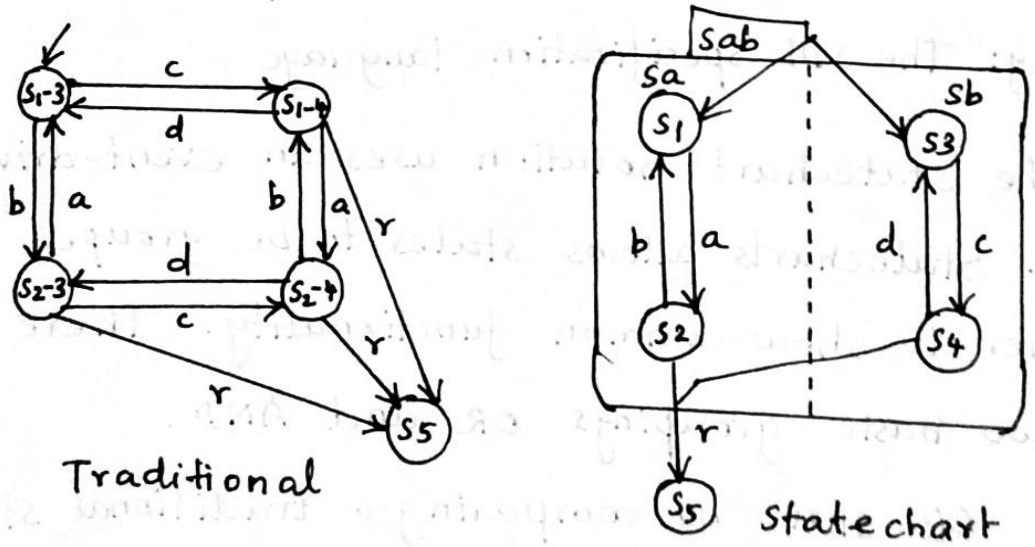


fig: An AND state in statecharts

An AND state specified in statechart notation as compared to the equivalent in the traditional state machine model. In the traditional model, there are numerous transitions between the states and one entry point.

The most important difference between the notation and statechart is that don't cares is represented in the table.

cond1 or (cond2 and !cond3)

Expression

	OR			
AND	cond 1		T	–
	cond 2		–	T
	cond 3		–	F

AND /OR table.

8. System Analysis and Architecture Design:

The CRC card methodology is a well known and useful way to help analyze system's structure. It is particularly well suited to object-oriented design since it encourages the encapsulation of data and functions.

The acronym CRC stands for:

- Classes define the logical groupings of data and functionality.
- Responsibilities describe what the classes do.
- Collaborators are the other classes with which a given class works.

The name CRC comes from the fact that the methodology is practiced by having people on write on index cards.

It has space to write down the class name, its responsibilities and collaborators & other information.

The essence of the CRC card methodology is to have people write on these cards, talk about them and update the cards until they satisfied with the results.

The following steps are used to analyze a system:

- Develop an initial list of classes.
- Write an initial list of responsibilities & collaborators.

- Create some usage scenarios.
- Walk through the scenarios.
- Refine the classes, responsibilities and collaborators.
- Add class relationships

CRC card Analysis:

A CRC card analysis of an elevator system

Real-world classes: elevator car, passenger, floor control, car control & car sensor

Architectural classes: car state, floor control reader, car control reader, car control sender and scheduler.

The basic operation of the elevator system as well as some unusual scenarios:

1. One passenger requests a car on a floor, gets in the car when it arrives, requests another floor, and gets out when the car reaches that floor.

2. One passenger requests a car on a floor gets in the car when it arrives and requests the floor that the car is currently on.

3. A second passenger requests a car

while another passenger is riding in the elevator.

4. Two people push floor buttons on different floors at the same time.

5. Two people push car control buttons in different cars at the same time.

Class	Responsibilities	Collaborators
Elevator car*	Moves up & down	Car control, car sensor, car control sender
Passenger*	Pushes floor control & car control buttons	Floor control, car control
Floor control*	Transmits floor requests	Passenger, floor control reader
Car control*	Transmits car requests	Passenger, car control reader.
car sensor*	Senses car position	scheduler
car state	Records current position of car.	Scheduler, car sensor
Floor control reader	Interface b/w floor control & rest of system	Floor control, scheduler
Car control reader	Interface b/w car control & rest of system	Car control, scheduler.
Car control sender	Interface b/w scheduler and car	Scheduler, elevator car
Scheduler	Sends commands to cars based upon requests	Floor control reader, car control reader, car control sender, car state

9. Quality Assurance:

The quality of a product or service can be judged by how well it satisfies its intended function.

The Quality Assurance (QA) process is vital for the delivery of a satisfactory system.

Quality Assurance Techniques:

The International Standards Organization (ISO) has created a set of quality standards known as ISO 9000. ISO 9000 was created to apply to a broad range of industries, including but not limited to embedded hardware & software.

The following observations about quality management based on ISO 9000:

i) Process is crucial:

Haphazard development leads to haphazard products and low quality. Knowing what steps are to be followed to create a high quality product is essential to ensuring that all the necessary steps are in fact followed.

ii) Documentation is important:

Documentation has several roles. The creation of the documents describing processes helps those involved understand the process; understand the process and how they are being implemented.

iii) Communication is important:

Quality ultimately relies on people. Good documentation is an aid for helping people understand the total quality process.

The following five levels of maturity:

1. Initial:

A poorly organized process with very few well defined processes. Success of a project depends on the efforts of individuals.

2. Repeatable:

This level provides basic tracking mechanisms that allow management to understand cost, scheduling and goal achievement.

3. Defined:

The management and engineering processes are documented and standardized.

4. Managed:

This phase makes detailed measurements of the development process and product quality.

5. Optimizing:

At the highest level, feedback from detailed measurements is used to continually improve the organization's processes.

Verifying the specification:

The requirements and specification are generated very early in the design process.

Verifying the requirements & specification is very important for the simple reason that bugs in the requirement or specification can be extremely expensive to fix later on.

A coding bug, if not found until after system deployment will cost money to recall and reprogram existing systems among other things.

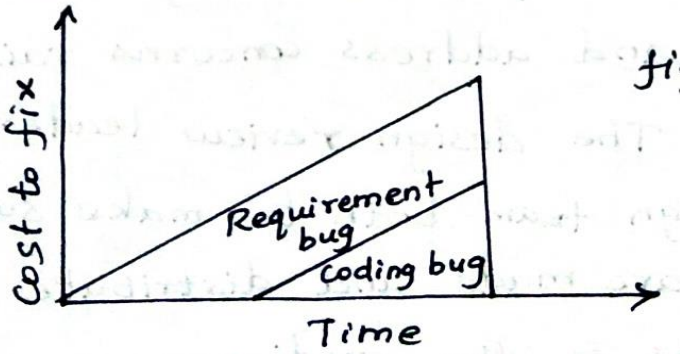


fig: Long lived bugs are more expensive to fix

Design reviews:

The design review is a critical component of any QA process. The design review is a simple, low cost way to catch bugs early in design process.

A design review is simply a meeting in which team members discuss a design, reviewing how a component of the system works.

A design review team has the following members:

- Designers of the component
- Review leader
- Review scribe
- Review audience

The design team prepares a set of documents that will used to describe the component. These documents are given to other members in advance of the meeting.

The notes taken by the scribe are used in meeting follow up. The design team should correct bugs and address concerns raised at the meeting. The design review leader coordinates with the design team both to make sure that the changes are made and distribute the change results to the audience.

10. Designing with computing platforms:

The design complexity of the hardware platform tends to a highly customized design. A platform may consists of anywhere from one to dozen of chips.

The BeagleBoard is the result of an open source project to develop a low cost platform.

The processor is an ARM cortex™_A8. The board itself includes many connectors and support for a variety of I/O.

Chip vendors often provide their own evaluation boards or evaluation modules for their chips. If the evaluation board does not meet your needs, you can modify the design using the netlist and board layout without starting from scratch.

Choosing a platform:

The hardware architecture of the platform is the more obvious manifestation of the architecture because you can touch it and feel it. The various components may all play a factor in the suitability of the platform.

i) Bus:

The choice of a bus is closely tied to

that of a CPU, because bus is an integral part of the microprocessor. But in applications, bus may be more of a limiting factor than the CPU.

ii) Memory:

The ratio of ROM to RAM and selection of DRAM versus SRAM can have a significant influence on the cost of the system. The speed of the memory will play a large part in determining system performance.

iii) Input and Output devices:

Platforms based on highly integrated chips only come with certain combination of I/O devices. The combination of I/O devices available may be a prime factor in platform selection.

There are two components available in the software platform – run time components and support components.

i) Run time components:

These are the critical part of the platform. An operating system is used to control the CPU and its multiple processes. A file system is used in embedded system to organize data.

ii) Support components:

These are critical to making use of complex hardware platforms. Without proper code and development of operating systems, the hardware itself is useless.

Intellectual property:

Intellectual Property (IP) is that we can own but not touch e.g) software netlist.

Examples of wide range of IP that we are using in embedded system design:

- run time software libraries
- software development environments
- schematics, netlists and other hardware design information.

Development environments:

Instead of using evaluation board for software development, it is done on a PC or workstation known as host. The hardware on which the code will finally run is known as the target. The host and target are finally connected by a USB link; Ethernet can also be used in higher speed link.

The target must include a small amount of software to talk to the host system. The host should be able to do the following:

- load programs into the target
- start and stop program execution on the target and
- examine memory & CPU registers

A cross compiler is a compiler that runs on one type of machine but generates code for another. After compilation, the executable code is typically downloaded to the embedded system by USB.

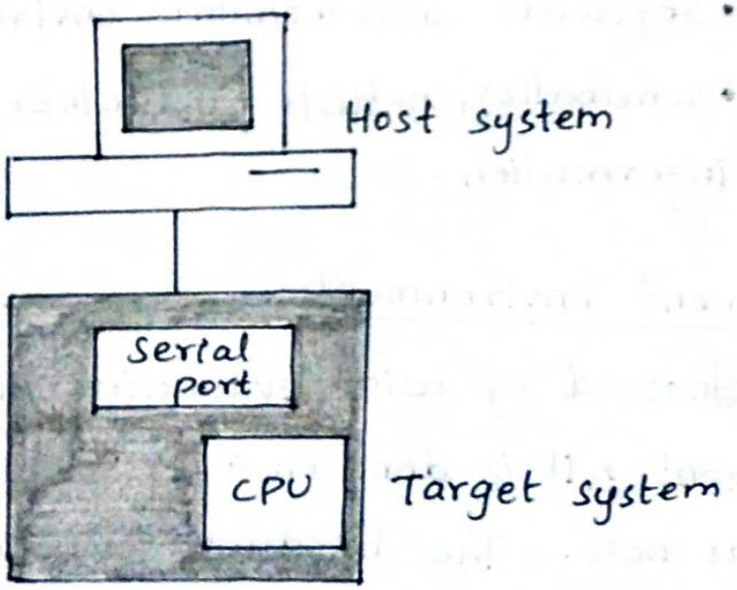


fig: Connecting a host and target system

Debugging techniques:

A good deal of software debugging can be done by compiling and executing the code on a PC or workstation. Embedded systems are usually less friendly programming environments than PC.

The USB port found on evaluation board is one of the most important debugging tools. USB can also used for diagnosing problems in the field or field upgrades of software.

Another very important debugging tool is the breakpoint. The simplest form of a breakpoint is for the user to specify an address at which the program's execution is to break.

Debugging challenges:

Logical errors in software can be hard to track down, but errors in real time code can create problems that are even harder to diagnose. Real-time programs are required to finish their work within a certain amount of time; they can create very unexpected behavior.

11. Consumer electronics architecture:

Consumer electronics devices has converged over the past decade around a set of common features that are supported by common architectural features.

There is no single platform for consumer electronics devices but the architecture in use are organized around some common themes.

This convergence is possible because these devices implement a few basic types of functions in various combinations: multimedia and communications. These requires basic architectural templates.

- Multimedia:

The media may be audio, image or video. These are generally stored ~~and~~ in compressed form and must be uncompressed to be played. The standards for audio is MP3, Dolby Digital™, for images is JPEG1 and for video is MPEG1-2, MPEG4, H.264 ; ~~for video is~~

- Data storage and management:

Because people want to select what multimedia objects they save or play, data storage goes hand-in-hand with multimedia capture and display. Many devices provide PC compatible file systems so that data can be shared easily.

H. ~~Consumer electronics architecture:~~

~~Consumer~~

- Communication:

Communications may be relatively simple such as a USB interface to a host computer.

Consumer electronic devices must meet several types of strict non functional requirements as well.

Many devices are battery operated ; they must operate under strict energy budgets.

Consumer electronics are very inexpensive. A typical primary processing chip must sell for $10. These devices must also provide high performance.

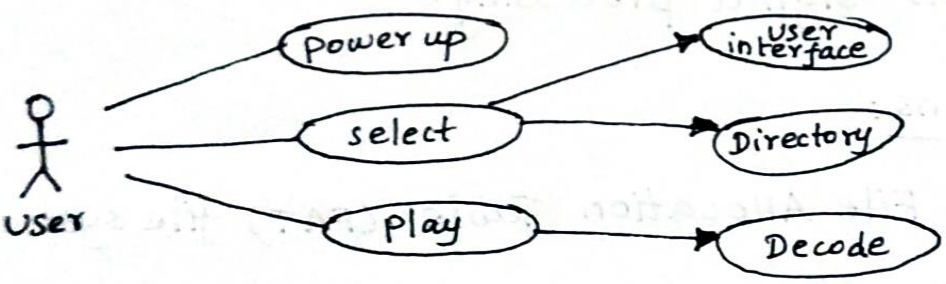


fig: Use case for playing multimedia

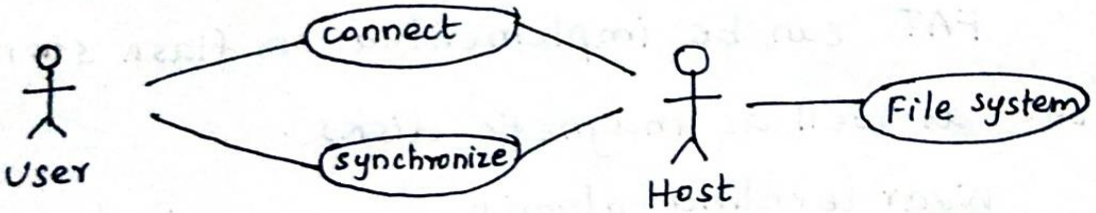


fig: Use case of synchronizing with a host system

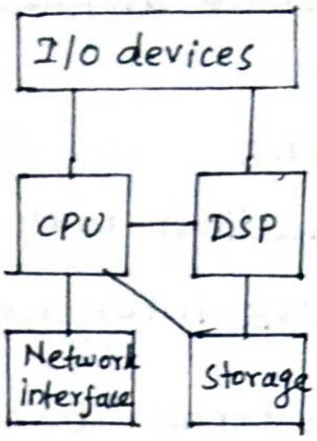


fig: Hardware architecture of generic consumer electronics device.

The figure shows two multiprocessor architecture. If more computation is required, more DSPs and CPUs may be added.

The RISC CPU runs the operating system, runs the User Interface, maintains the file system; The DSP performs signal processing.

File Systems:

DOS File Allocation Table (FAT) file systems refer to the file system developed by Microsoft for early versions of the DOS Operating System (MicOO).

FAT can be implemented on flash storage devices as well as magnetic disks.

Wear leveling algorithm can be implemented without disturbing the file system.

FAT can be implemented in small amount of code.

Many consumer electronic devices use flash memory for mass storage. Flash memory is a type of semiconductor memory provides permanent storage. Values are stored in flash memory cell. This cell does not require an external power supply.

The limitation of flash memory is: writing a flash memory cell causes mechanical stress that eventually wears out the cell.

A wear leveling flash file system (Ban95) manages the use of flash memory locations to equalize wear while maintaining compatibility with existing file systems.

12. Platform level performance analysis:

Bus based systems add another layer of complication to performance analysis. Platform level performance involves much more than the CPU. Most precisely, CPU provides an upper bound on performance but any other part of the system can slow down the CPU. Merely counting instruction execution time is not enough.

To move data from memory to the CPU, we must:

- read from the memory
- transfer over the bus to the cache
- transfer from the cache to the CPU.

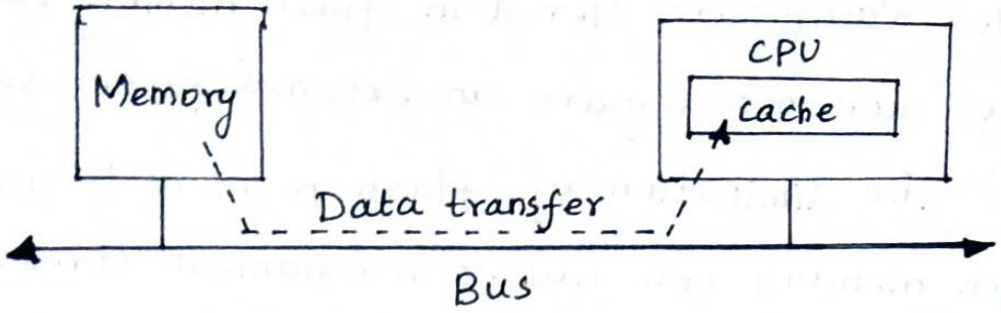


fig: Platform - level data flows & performance

The time required to transfer from the cache to the CPU is included in the instruction execution time.

The most basic measure of performance is bandwidth (the rate at which the data can be moved). The simplest way to measure performance is in units of clock cycles.

In units of bus cycles T, convert bus cycle counts to real tim t using bus clock period P.

$$t = TP \quad \text{——} \quad ①$$

A basic bus transfer transfers a W-wide set of bytes. The data transfer itself takes D clock cycles.

Addresses, handshaking and other activities constitute overhead that may occur before O_1 or

after O_2. For simplicity, $O = O_1 + O_2$.

This gives a total transfer time in clock cycles of:

$$T_{basic}(N) = (D+O)\frac{N}{W} \quad \text{——— (2)}$$

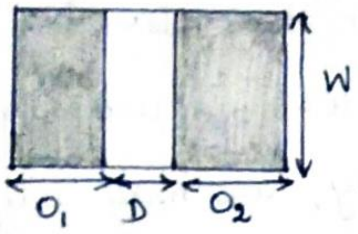


fig: Times & data volumes in basic bus transfer

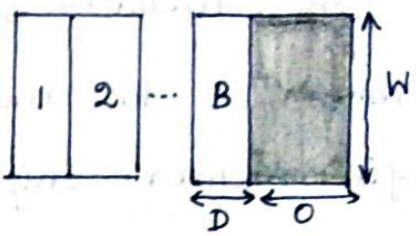


fig: Times & data volumes in a burst bus transfer

A burst transaction performs B transfers of W bytes each, which requires D clock cycles. The bus also introduces O clock cycles of overhead per burst. This gives,

$$T_{burst}(N) = (BD+O)\frac{N}{BW} \quad \text{——— (3)}$$

The width of a memory determines the number of bits read from the memory in 1 cycle.

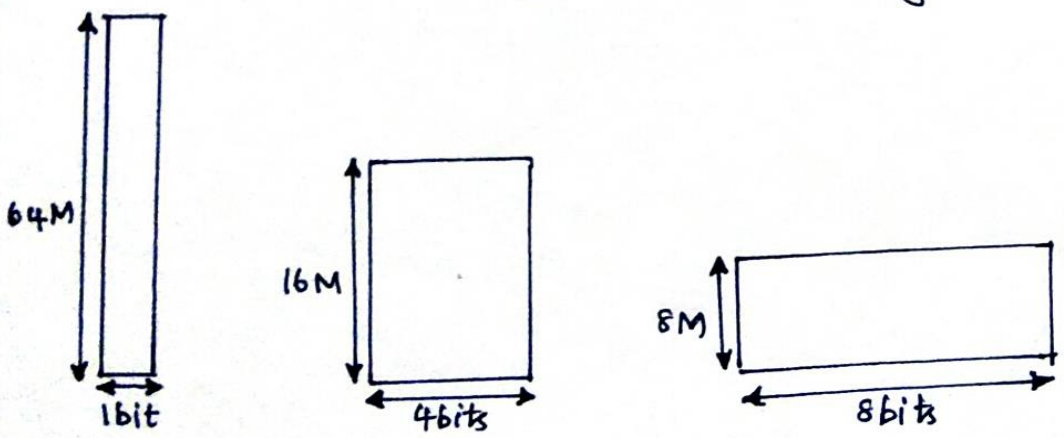


fig: Memory aspect ratios

The figure shows memories of same size can have different aspect ratios. For e.g) a 64 Mbit memory have 64 million addresses, the same size memory in a 4 bit wide format have 16 million addresses and a 8 bit wide memory have 8 million addresses.

The memory system width may also be determined by the memory modules we use. Instead of buying memory chips individually, we may buy memory as SIMMs or DIMMs.

The situation is slightly more complex, if the data types don't fit naturally into the width of the memory. The total number of accesses required to read E data elements of w bits each out of a memory of width W is:

$$A = \left[\left(\frac{E}{w}\right) \bmod W\right] + 1$$