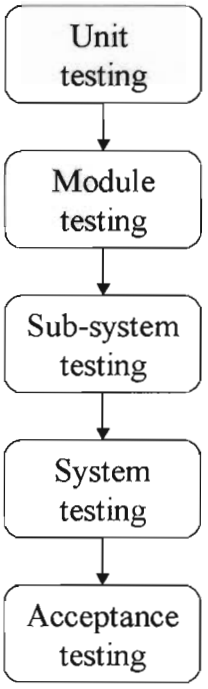
Answers EEE115 Spring 2005

1.

a. To ensure that every aspect of the design has been tested in some way the V.V.T. phase must be related to the structure/hierarchy of the implementation phase. Systems are designed from the top-down (from a concept to many components – see partitioning and decomposition) but they are built/implemented from the bottom-up (modules built using components, sub-systems from modules, whole system from the sub-systems). Thus, applying V.V.T. inline with implementation allows errors or omissions to be more easily identified and the subsequent corrections to be incorporated within the design. The V.V.T. process, therefore, proceeds in stages where testing is carried out incrementally in conjunction with system implementation to test every element of the design at successive stages of assembly. While in any particular case the hierarchy has its own form, it will trace the following general pattern in which the testing process consists of five stages.



1. Unit testing - Individual components are tested to ensure that they operate correctly. Unit testing treats each component as a stand-alone entity that does not need other components during the testing process.
2. Module testing - A module is a collection of dependent components. A module encapsulates related components so can be tested without other system modules.
3. Sub-system testing - This phase involves testing collections of modules which have been integrated into sub-systems. Sub-systems may be independently designed and implemented, and the most common problems which arise in large systems are sub-system interface mismatches. The sub-system test process should concentrate on the detection of interface errors by rigorously exercising these interfaces.

4. System testing - The sub-systems are integrated to make up the entire system. The testing process is concerned with finding errors which normally result from unanticipated interactions between sub-systems and components. It is also concerned with validating that the system meets its functional and non-functional requirements.
5. Acceptance testing - This is the final stage in the testing process before the system is accepted for operational use. It involves testing the system with data supplied by the customer (or his agent such as the marketing department) rather than simulated data developed as part of the testing process. Acceptance testing often reveals errors and omissions in the system requirements definition. The requirements may not reflect the actual facilities and performance required by the user and testing may demonstrate that the system does not exhibit the anticipated performance and functionality.

b. System testing is expensive. For some large systems, such as real-time systems with complex timing constraints, testing may consume about half the overall development costs. So, careful planning is necessary to get the most out of testing and to control testing costs.

Test planning is concerned with setting out standards for the testing process rather than describing product tests. The major components of a test plan are:

1. The testing process - a description of the major phases of the testing process, as described above.
2. Requirements traceability - users are most interested in the system meeting its requirements and testing should be planned so that all requirements are individually tested.
3. Testing schedule - an overall testing schedule and resource allocation for this schedule. This, obviously, is linked to the more general project development schedule.
4. Test recording procedures - it is not enough simply to run tests. The results of the tests must be systematically recorded. It must be possible to audit the testing process to check that it has been carried out correctly.
5. Hardware and software requirements - this section should set out software tools required and estimated hardware utilisation.
6. Constraints - constraints affecting the testing process (such as staff shortages) should be anticipated in this section.

Test plans are not just management documents. They are also intended to be used by engineers and technical staff involved in designing and conducting the system tests. In addition, it must be recognised that the test plan is not a static document and should revised as the system evolves during the design implementation and testing phases.

c. To test a low-pass filter one must evaluate the frequency response. This is done by sweeping the input frequency whilst keeping the magnitude of the input voltage constant and measuring the output voltage. The results of this test would then be compared to a mathematical model to ensure a match has been obtained. In addition to this, one would also have to test for the effects of component tolerances.

d. The stuck-at fault model assumes that any failure in a unit can be modelled by saying the output of the unit is "stuck-at" a particular logic value (0 or 1). In the SAF model we assume that the effect of the physical fault (whatever it may be) is to create only two kinds of logical fault: a stuck-at-1 fault (abbreviated to SA1 or s@1) and a stuck-at-0 fault (SAO or s@0). We place faults on a node, or on an input of a circuit, or on an output of a circuit. Fault simulation is then used to see what happens in a design when we deliberately introduce faults. In a production test there is only limited access to the circuit. To test a circuit we must devise a series of sets of input patterns that will detect any faults that there are in the circuit. A stimulus is the application of one such set of inputs (a test vector) to the inputs of the circuit.

   A fault simulation mimics the behaviour of the production test. The fault simulator deliberately introduces all possible faults into our circuit, one at a time, to see if the test program will find them. As each fault is inserted, the fault simulator runs our test program. If the fault simulation shows that the outputs of the faulty circuit are different from the outputs of a good circuit at any time, then we have a detected fault; otherwise we have an undetected fault. This allows us to generate a set of test vectors that can be used to test a circuit during production.

e. As more functionality is integrated onto digital IC's testing them becomes more complex due to the limited access to the logic cells used to make such devices and, therefore, testing such devices is difficult. Previously, scan-paths have been proposed for testing logic devices but today's current state-of-the-art devices are far too complex to use these techniques. JTAG (Joint Test Action Group) therefore proposed a boundary-scan test (BST) interface to simplify testing.

   BST is an industry standard testing interface developed for testing digital IC's, associated circuitry and the PCB based on boundary scan test techniques. Using a 4-wire interface test data is loaded onto the IC under test. A predefined serial bit-pattern is then applied to each input pin of the IC and corresponding output pins are then clocked out to test machine or PC using a shift register. The interface also provides facility for cascading a number of IC with JTAG interfaces together so that several chips can be tested using just one interface thereby simplifying the testing of more complex systems.

f. Verification and validation are 2 distinct processes that are used to confirm that the system conforms to the Requirements Specification and the System Specification.

   Validation ensures the final system meets the Requirements Specification (i.e. the user requirements).

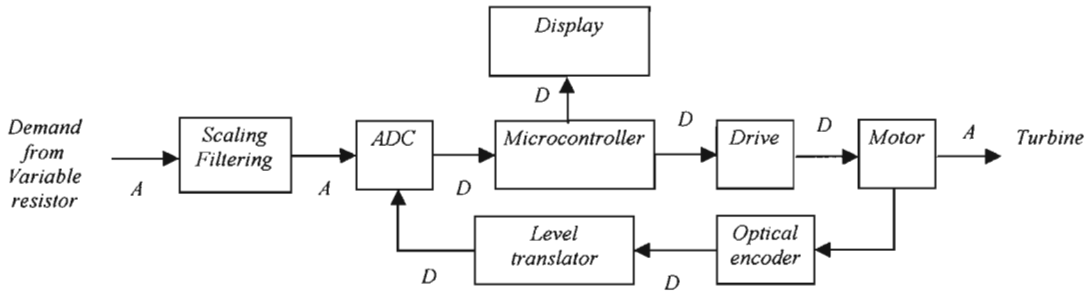
   Verification is the process of ensuring the system is being built according to the System Specification (or Design Specification).

2.

a. Inputs and outputs of the system

Input/Output	Interface requirements	Signal Type
Demand variable resistor (input)	Voltage therefore may require scaling and filtering. To interface with micro-controller would require converting into digital via ADC.	Analogue
Speed (input)	Optical based therefore provides pulses. Could be interfaced to micro by converting frequency to voltage i.e. Phase Lock Loop or by counting pulses with micro. To interface use buffer/level shifting circuit.	Digital
Display (output)	Digital in nature so could simply access it as a memory device or via serial comms. Alternatively generate analogue voltage and use a panel meter type instrument.	Either

b.

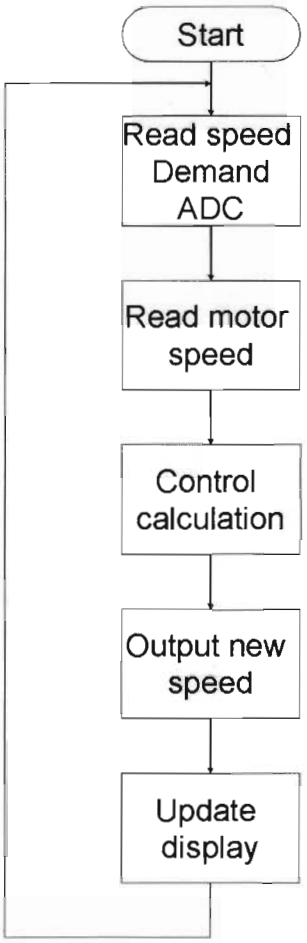


A - Analogue
D – Digital

Note ADC may be incorporated in microcontroller.
Also Display may be analogue but this would require two DACs or something similar.

c. Functions microcontroller needs to perform

1. Read ADC
2. Read speed
3. Perform control calculation
4. Send motor drive signal
5. Update display



d. Minimum ADC resolution

2^8=256 1/256=0.0039
2^9=512 1/512=0.002

9-bits are required to achieve minimum resolution.

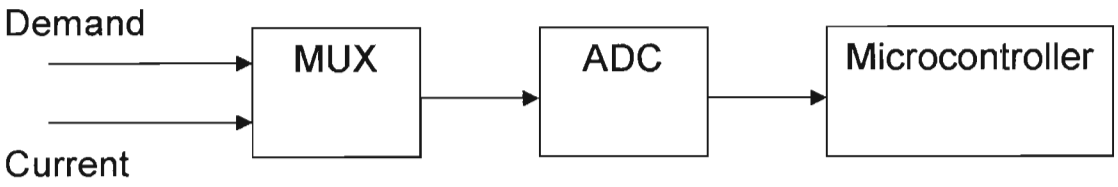
Assuming linear scaling,
$2.2/5 \times 512 = E1_{hex}$

e.
Would choose Class-D drive system to achieve high-efficiency. Therefore the most obvious output would be to employ the PWM output. The DAC output would require comparing with ramp to achieve PWM and hence would be more complicated.

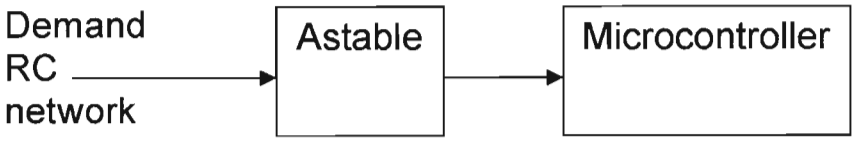
Using a linear amplifier would dissipate too much power and, hence, waste energy.

e.
Two methods could be used:

1. Multiplexer
2. RC astable


Demand
Current
MUX
ADC
Microcontroller

OR

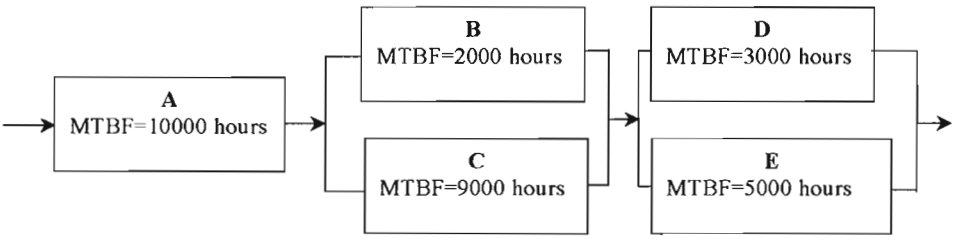

Demand
RC
network
Astable
Microcontroller

3.

a. Mean-time-between-failure is the average time it takes a system to fail. It is defined mathematically as:

$$MTBF = \frac{\text{No. of operating hours}}{\text{No. of failures}}$$

b. Reliability for 1500 hours of operation.



$$R(t) = e^{-\frac{t}{M}} = e^{-\lambda t}$$

R_A=0.8607, R_B=0.4724, R_C=0.8465, R_D=0.6065, R_E=0.7408

R_{p1} is the parallel branch B and C, R_{p2} is E and E

$R_{p1}=R_B+R_C- R_B\times R_C=0.4724+0.8465-0.4724\times0.8465=0.9190$
$R_{p2}=R_D+R_E- R_D\times R_E=0.6065+0.7408-0.6065\times0.7408=0.8980$

$R_{tot}=R_A\times R_{p1}\times R_{p2}=0.8607\times0.9190\times0.8980=0.7103$

c. Now MTBF of A is reduce to 8000 hours and an additional system, called F, is placed in parallel with system E with a MTBF of 4000 hours.

R_A= 0.8290, R_F= 0.6873

Now $R_{p2}=1-(1-R_D)(1-R_E)(1- R_F)=1-(1-0.6065)(1-0.7408)(1-0.6873)=0.9681$

$R_{tot}=R_A\times R_{p1}\times R_{p2}=0.8290\times0.9190\times0.9681=0.7375$

d. Over T_h hours F_h failures occur with a downtime of each failure of D_h hours.

Therefore total failure time is $FT = F_h D_h$

Total time system is operating is $T = T_h - F_h D_h$

Failure rate is $\lambda = \frac{F_h}{T} = \frac{F_h}{T_h - F_h D_h}$

For the example T_h=250h, F_h=5 and D_h=2h

$$\lambda = \frac{5}{250-5\times2} = \frac{5}{240} = 0.0208 \text{ failures per hour}$$

$$MTBF = \frac{1}{\lambda} = \frac{1}{0.0208} = 48 \text{ hours}$$

e.

Redundancy

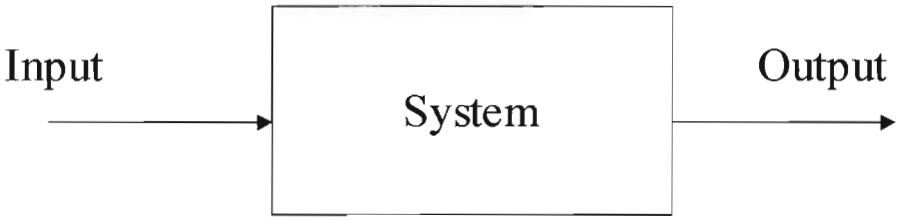
Redundancy is a design practice that allows a system to maintain working although parts of the system may have failed. Redundancy is important when designing critical systems, such as aircraft, where failure can result in the loss of life or many millions of pounds. Therefore, if the project requires high-reliability it is essential that certain key sub-systems be identified during the partitioning phase so that appropriate design measures can be taken. Redundancy is achieved by paralleling critical systems.

Component de-rating

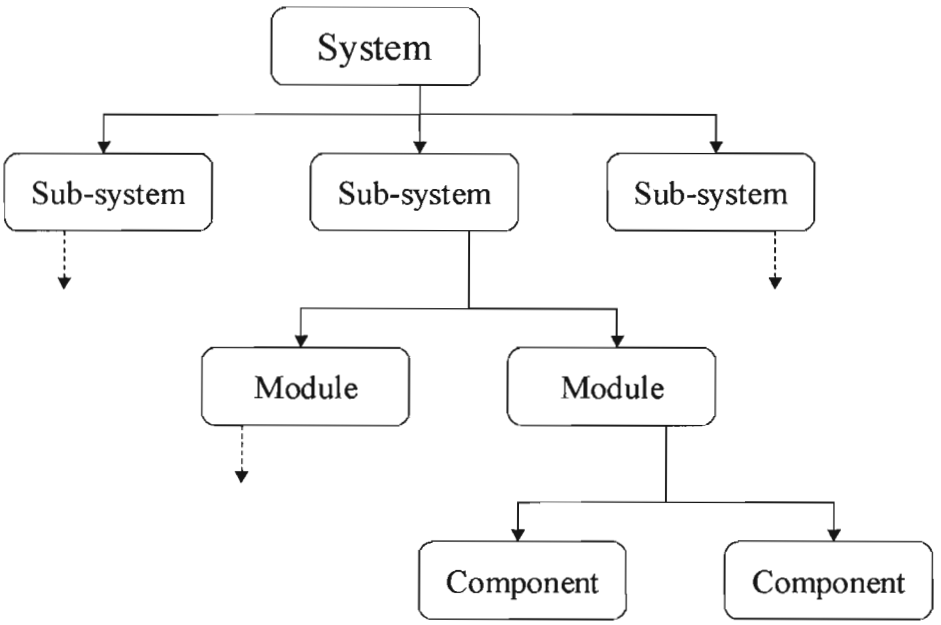
Component manufacturers provide the designer with all manner of design data relating to their components. For example, a transistor datasheet provides information above maximum current, voltage and power dissipation rating. If these ratings are exceeded during operation then it is more likely that the component will fail. It has been shown experimentally that a component's lifetime is reduced the closer it operates to its maximum ratings. Designers can take advantage of this phenomenon and de-rate their components to achieve greater reliability. For example, imagine you are developing a circuit that requires a 100μF capacitor at a maximum voltage of 100V then the reliability can be increased by choosing a 200V working capacitor.

4.

a. The main elements of are input, output and a process. All systems process some input to produce an output.



b.



System

System or system specification is the top-level of our hierarchy. At this level the system is described only in a terms of its external behaviour. Details about the internal behaviour of the system are left to the subsequent levels.

Sub-system

Sub-systems represent divisions in the functionality/requirements of the overall system. Often, sub-system definitions directly relate to the system specification statements.

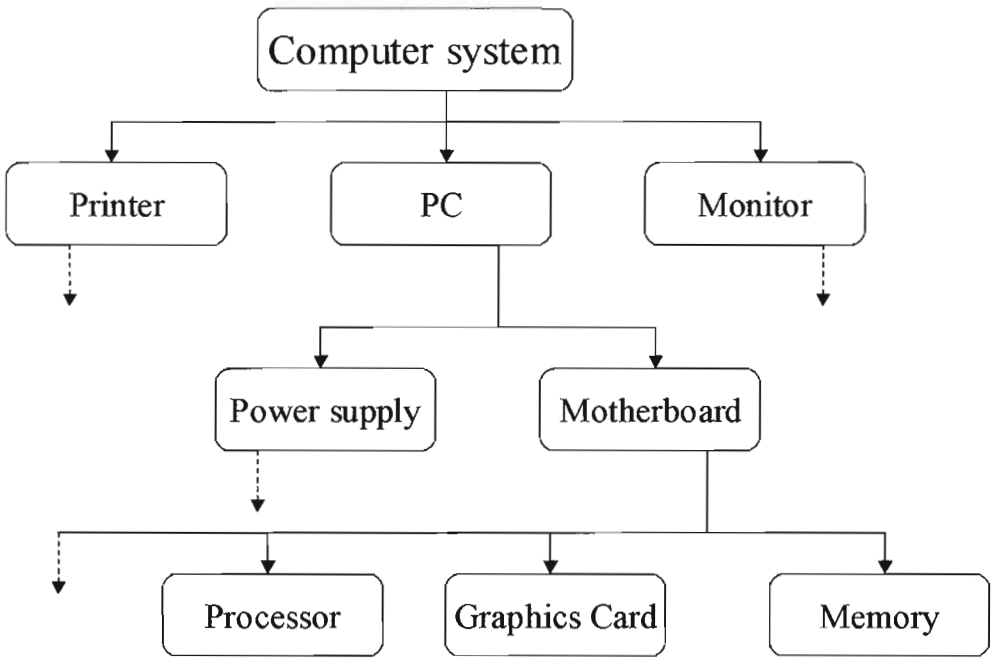
Modules

Modules are high-level functional blocks required to construct a sub-system.

Component

Components are the lowest level building blocks for the system. Electronic components and programming language functions fall into this category.

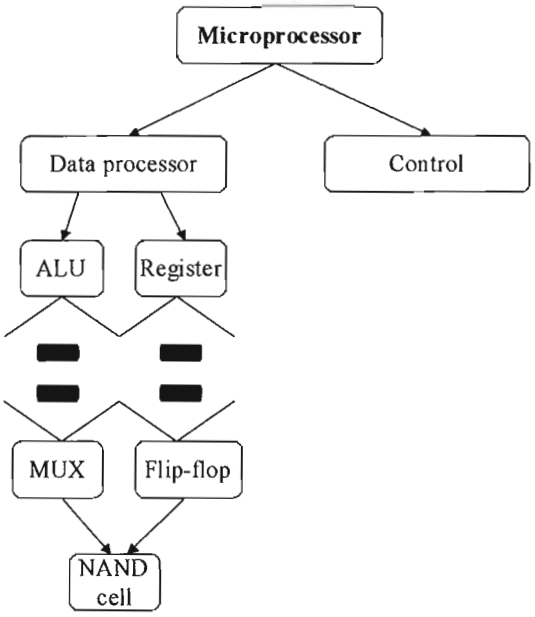
c.



There may be variations on this diagram but the important parts are Computer system be broken down into printer, PC, Monitor, Keyboard and Mouse (not shown here). The PC sub-system should then be expanded to include the common elements of a PC such as those shown in the figure and other components such as hard disk, CD-ROM drive, etc.

d. A reconvergent type hierarchy where, initially, the systems structure is tree-like branching out at each stage of the partitioning process. Ultimately, however, the structure begins to converge together where each building-block or low-level element is essentially built out of similar components.

A microprocessor is a typical example, where the structure initially branches out in subsequent stages but ultimately the processor is constructed from simple logic gates.



e. The waterfall model represents the discrete stages of the design process as a sequence of activities following one another other in a defined hierarchy. Each stage of the design process is defined as:

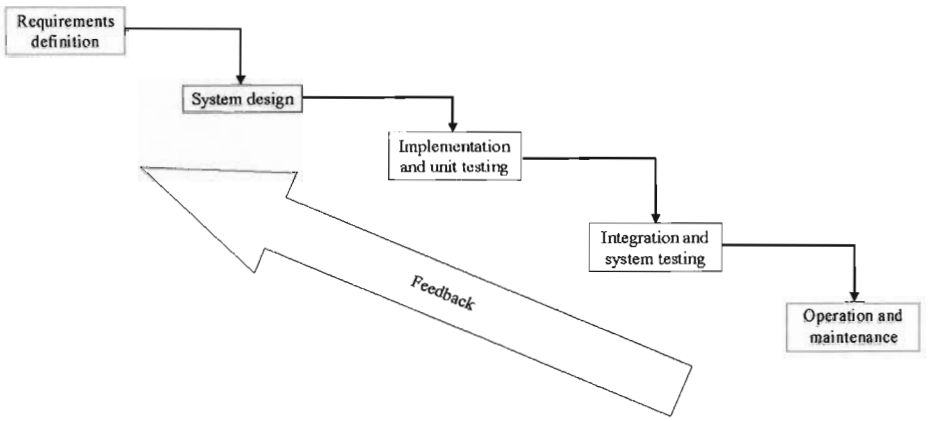
1.Requirements capture
Systems users requirements are defined here in detail.
System specification is developed.

2.System design
The actual system is designed here. Usually involves partitioning the whole system into smaller, more tangible elements/modules suitable for implementation.

3.Implementation and unit testing
Individual modules (units) from the design stage are constructed and tested in isolation to ensure each unit meet the required specification.

4.Integration and system testing
Here the individual system modules are integrated to form the whole system. The whole system is tested to ensure the system meets the specification.

5.Operation and maintenance
The system is installed/commissioned and put into practical use. Normally, throughout the life of a system, the system is maintained to remove errors not initially detected during the testing phases or to correct for errors or omissions from the specification or design stages. In addition, the system may be continually updated, as the end-user requires extra functionality.



f. Abstraction is the process of filtering out certain details about a system (sub-system or element) to obtain a more generic description of the system's behaviour. Abstraction is a powerful concept that, by removing a system's specifics, allows the engineer to classify the system (or part of the system) according to its behaviour without the restrictions imposed by its' implementation or by its' purpose (end usage). Abstraction concepts are particularly useful in the analysis phase of the design cycle for modelling and analysing systems that have similar behaviours. This feature means engineer can benefit by transferring knowledge gained in analysing one system to other similar systems even though their final usage may be completely different.