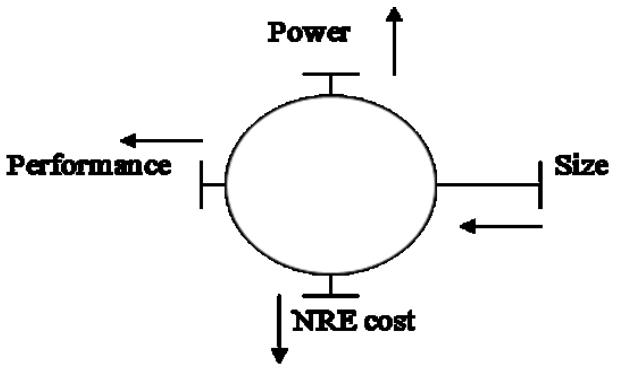
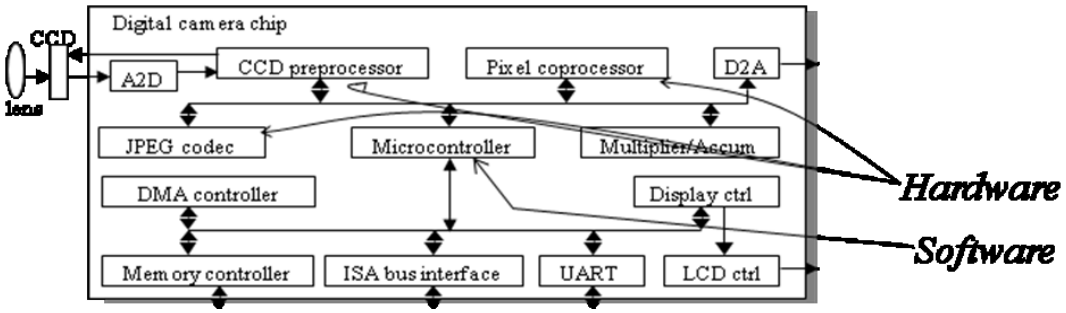
Q.2 a. What is a design metric? List pair of design metrics that may compete with one another, providing an intuitive explanation of the reason behind the competition. (8)

Answer: A design metric is a measurable feature of system's implementation. Commonly used metrics use various parameters of an embedded system are as follow. (***1 Mark)***

1. NRE Cost (Nonrecurring Engineering Cost):
2. Time to market
3. Unit Cost
4. Size
5. Performance
6. Power
7. Safety and reliability
8. Flexibility
9. Maintainability

Response Time (***2 Marks)***





(***3 Marks)***

- Expertise with both **software and hardware** is needed to optimize design metrics
  - Not just a hardware or software expert, as is common
  - A designer must be comfortable with various technologies in order to choose the best for a given application and constraints

(2 Marks)

b. List and define the three main processor technologies. What are the benefits of using each of the three different processor technologies? (8)

Answer: Processor technology relates to the architecture of the computation engine used to implement a systems desired functionality. Each such processor differs in its specialization towards a particular function, thus manifesting design metrics different than other processors. Several types of processors can implement this functionality, they are

1. General – purpose processors (Software)
2. Single – purpose processors (Hardware)
3. Application – specific processors

Description of each: (Refer Text1: page no: 9 -12)

Q.3 a. Explain the programmer and Operating System considerations in ESD. (8)

Answer:

Programmer Considerations: ***(4 Marks)***

- Program and data memory space
  - Embedded processors often very limited
    - e.g., 64 Kbytes program, 256 bytes of RAM (expandable)
- Registers: How many are there?
  - Only a direct concern for assembly-level programmers
- I/O
  - How communicate with external signals?
- Interrupts

Operating System: ***(4 Marks)***

- Optional software layer providing low-level services to a program (application).
  - File management, disk access
  - Keyboard/display interfacing
  - Scheduling multiple programs for execution
    - Or even just multiple threads from one program
  - Program makes system calls to the OS

b. What are ASIP's? Explain popular ASIP's used in ESD. (8)

Answer:

Application-Specific Instruction-Set Processors (ASIPs): ***(2 Marks)***

- General-purpose processors
  - Sometimes too general to be effective in demanding application
    - e.g., video processing – requires huge video buffers and operations on large arrays of data, inefficient on a GPP
  - But single-purpose processor has high NRE, not programmable
- ASIPs – targeted to a particular domain

- Contain architectural features specific to that domain
  - e.g., embedded control, digital signal processing, video processing, network processing, telecommunications, etc.
- Still programmable

A Common ASIP: Microcontroller (***3 Marks)***

• For embedded control applications
- Reading sensors, setting actuators
- Mostly dealing with events (bits): data is present, but not in huge amounts
- e.g., VCR, disk drive, digital camera (assuming SPP for image compression), washing machine, microwave oven

• Microcontroller features
- On-chip peripherals
  - Timers, analog-digital converters, serial communication, etc.
  - Tightly integrated for programmer, typically part of register space
- On-chip program and data memory
- Direct programmer access to many of the chip's pins
- Specialized instructions for bit-manipulation and other low-level operations

Another Common ASIP: Digital Signal Processors (DSP): (***3 Marks)***

• For signal processing applications
- Large amounts of digitized data, often streaming
- Data transformations must be applied fast
- e.g., cell-phone voice filter, digital TV, music synthesizer

• DSP features
- Several instruction execution units
- Multiple-accumulate single-cycle instruction, other instructions.
- Efficient vector operations – e.g., add two arrays
  - Vector ALUs, loop buffers, etc.

Q.4 a. Draw the 4×4 RAM's internal structure and explain its each individual blocks. **(8)**

Answer:

Explanation (Refer Text1: page no: 118 – 119, Figure 5.6) (***4+4 Marks)***

b. Draw and explain the profile of cache performance trade-offs. (8)

Answer:

Explanation (Refer Text1: page no: 128 – 129, Figure 5.13) (***5+3 Marks)***

Q.5 a. Explain how ATM timeout can be implemented using a Watchdog Timer. (8)

Answer:

Explanation (Refer Text1: page no: 89 – 90)

b. Draw the hardware and explain controlling of a stepper motor using a driver. **(8)**

Answer: Explanation (Refer Text1: page no: 98 – 99, Figure 4.9) (***4+4 Marks)***

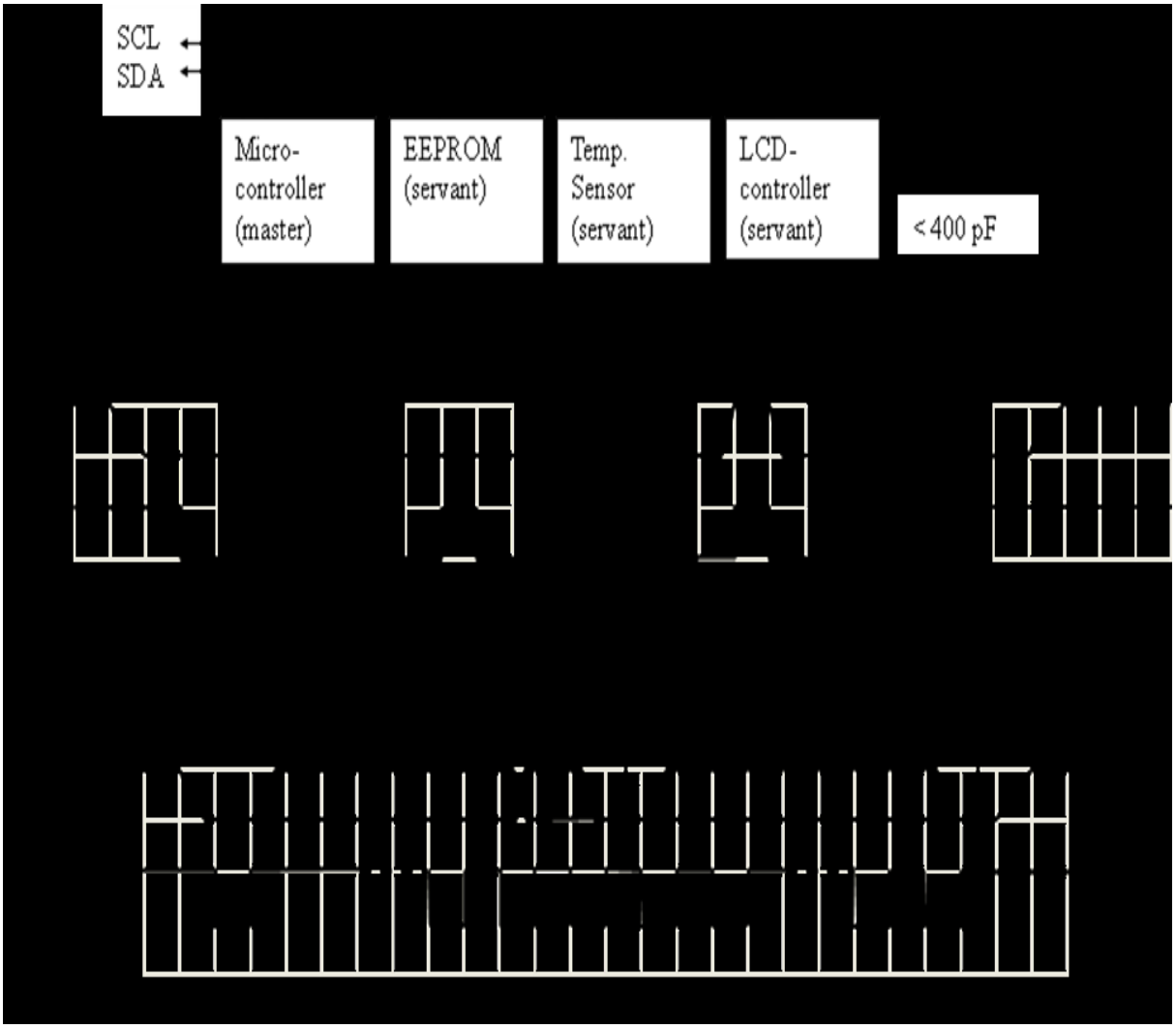
Q.6 a. Define arbitration method and explain any one. **(8)**

Answer:

Explanation (Refer Text1: page no: 159 - 160)

b. Explain the four popular serial bus protocols. **(8)**

Answer: The four popular serial protocols, namely the I2C protocol, the CAN protocol, the FireWire protocol, and the USB protocol.

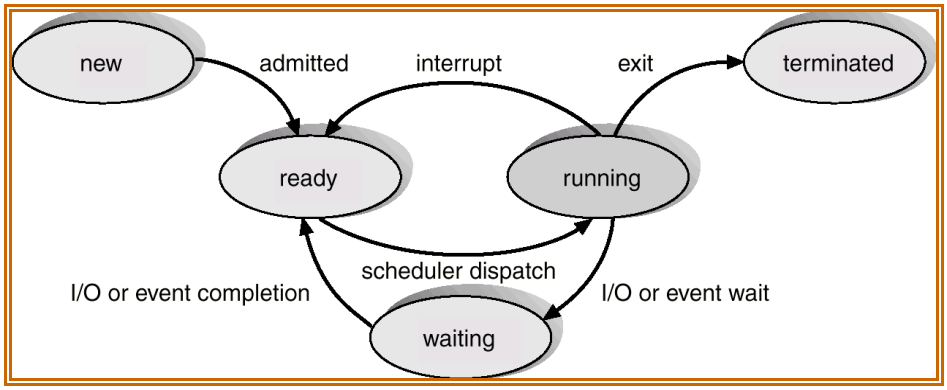


(2+2+2+Exaplanation-2 Marks)

Q.7 a. Explain TASKS & TASKS STATES in RTOS. (8)

Answer:

- **Bottom line**: keep track of which process/task runs (has the CPU)
- As a process executes, it changes *state (in some states, process/task does need CPU)*
  - **new**: The process is being created
  - **running**: Instructions are being executed
  - **waiting**: The process is waiting for some event to occur
  - **ready**: The process is waiting to be assigned to a process
  - **terminated**: The process has finished execution (***4 Marks)***



(***4 Marks)***

b. Explain reentrancy & the rules to decide if a function is reentrant. (2+6)

Answer: Page No 168 and 169 Text 2

Q.8 a. Explain the RTOS memory management subsystem. (8)

b. In RTOS environments, what are the rules Interrupt routines must follow that do not apply to task code? (8)

Answer: Refer: Text 2- pg-no: 199

Q.9 a. Explain some few techniques to save memory space. (8)

b. Discuss how to encapsulate Semaphores and Queues. (8)

TEXT BOOKS

1. **Embedded System Design, A Unified Hardware/Software Introduction, Frank Vahid / Tony Givargis, 2006 reprint, John Wiley Student Edition**

2. **An Embedded Software Primer, David .E. Simon, Fourth Impression 2007, Pearson Education**