TOPIC 8 INTERACTING WITH EMBEDDED SYSTEMS

OUTLINE

- What are embedded systems?
- Embedded System Components
  - Hardware/software
- Embedded System applications
- Model, languages and tools
- Hardware/software co-design and synthesis
- Reconfigurable Computing
- Real time Operating systems

EMBEDDED SYSTEM

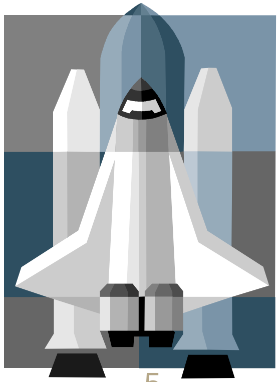
- Is a special purpose system designed to perform a few dedicated functions.
- Small foot prints (in memory)
- Highly optimized code
- Cell phones, mp3 players are examples.
- The components in an mp3 player are highly optimized for storage operations. (For example, no need to have a floating point operation on an mp3 player!)

REAL-TIME SYSTEM CONCEPTS

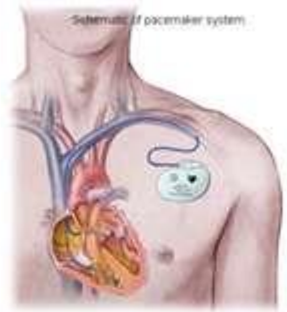
- A system is a mapping of a set of input into a set of outputs.
- A digital camera is an example of a realtime system: set of input including sensors and imaging devices producing control signals and display information.
- Realtime system can be viewed as a sequence of job to be scheduled.
- Time between presentation of a set of inputs to a system and the realization of the required behavior, including availability of all associated outputs, is called the response time of the system.

REAL-TIME SYSTEM CONCEPTS (CONTD.)

- Real-time system is the one in which logical correctness is based on both the correctness of the output as well as their timeliness.
- A soft real-time system is one in which performance is degraded by failure to meet response-time constraints.
- A hard real-time system is one in which failure to meet a single deadline may lead to complete and catastrophic failure.
- More examples:
  - Automatic teller: soft
  - Robot vacuum cleaner: firm
  - Missile delivery system: hard



EMBEDDED SYSTEMS



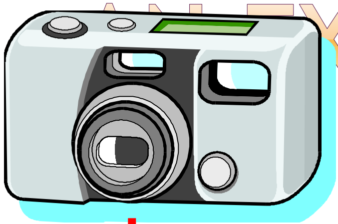
WHAT'S AN EMBEDDED SYSTEM?

- Embedded systems =
  - information processing systems embedded into a larger product
- Two types of computing
  - Desktop – produced millions/year
  - Embedded – billions/year
- Non-Embedded Systems
  - PCs, servers, and notebooks
- The future of computing!
  - Automobiles, entertainment, communication, aviation, handheld devices, military and medical equipments.

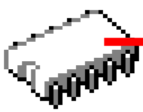
EMBEDDED SYSTEMS

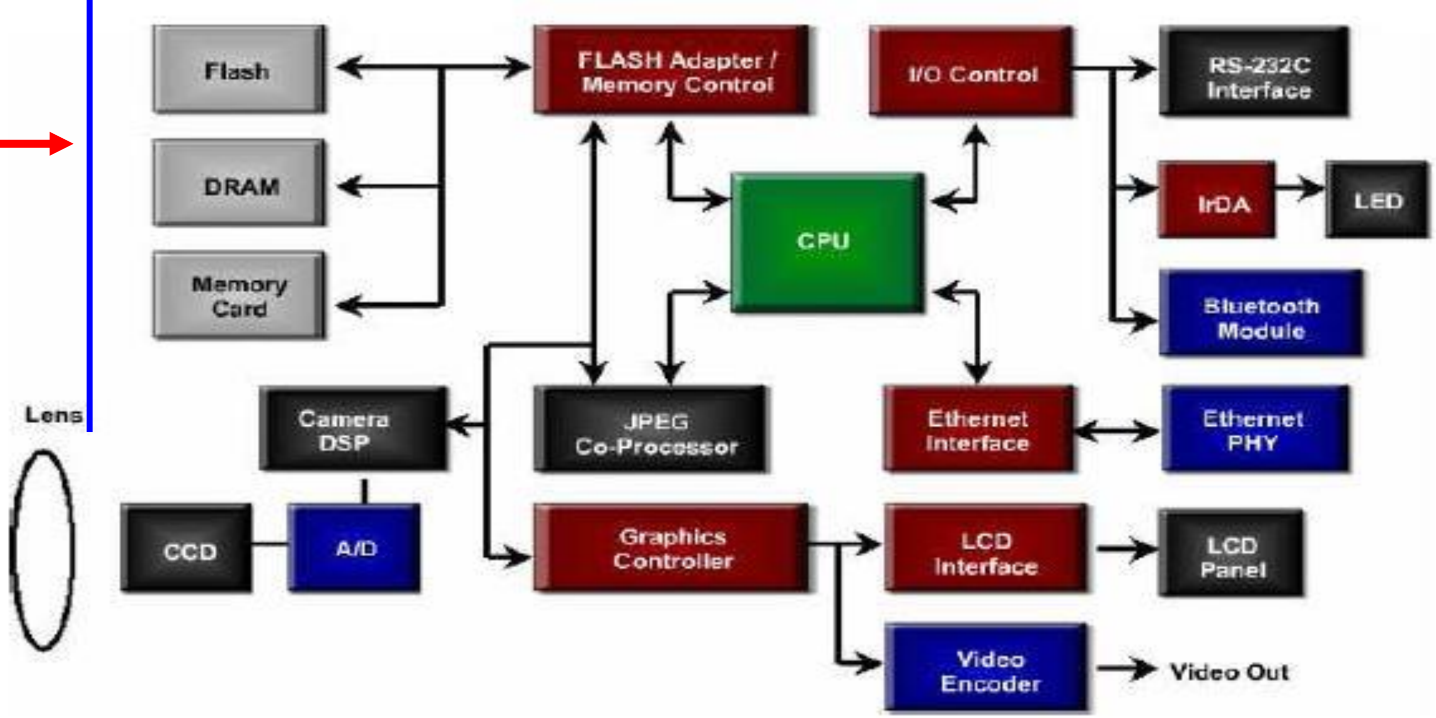
- Devices other than desktop PCs, servers, and notebooks
  - Electricity running through
  - Perform something intelligent
- Hardware/software which form a component of a larger system, but are concealed from user
- Computers camouflaged as non-computers
- The future of computing!

AN EXAMPLE EMBEDDED SYSTEM

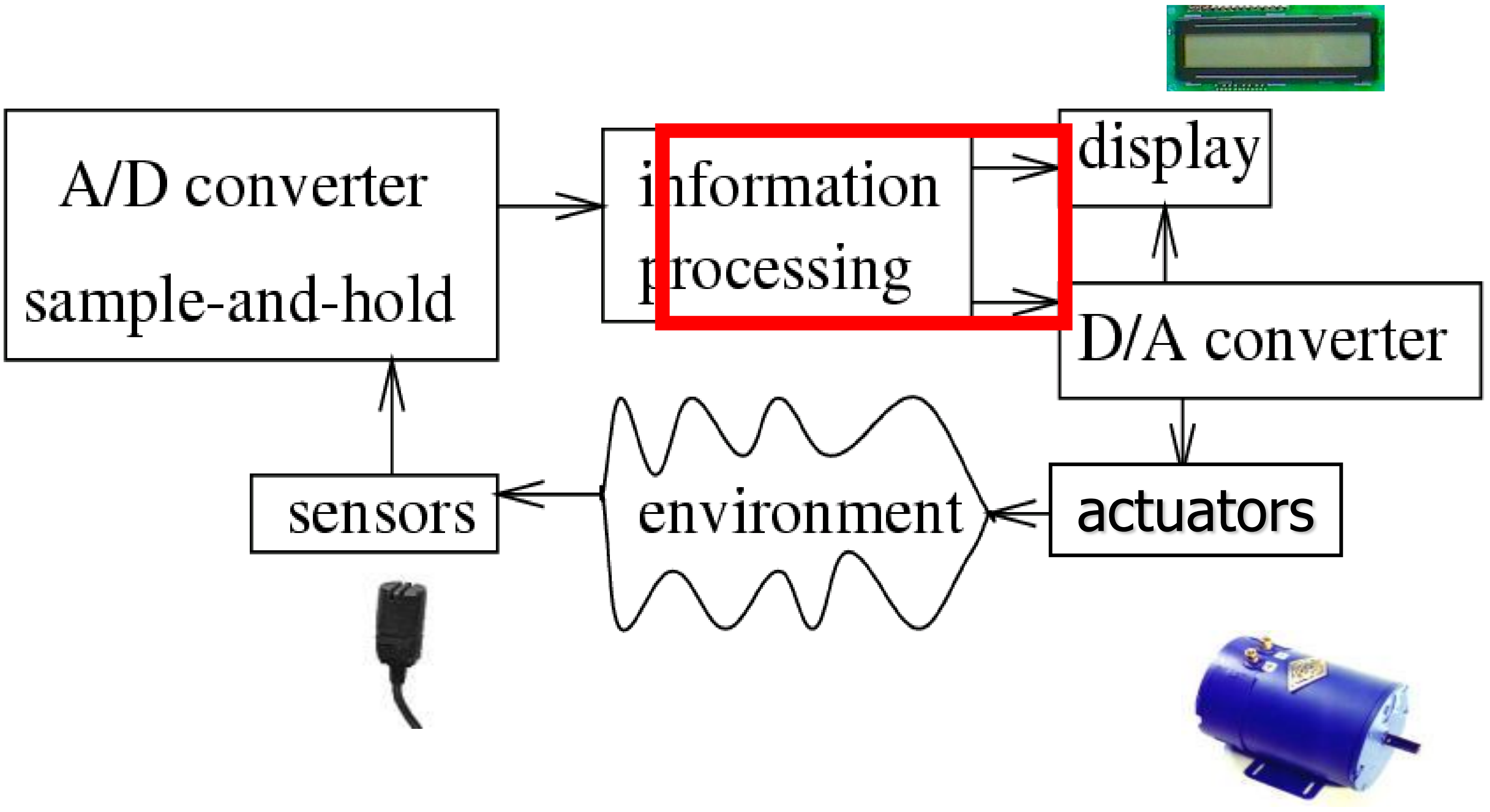


Digital Camera Block Diagram

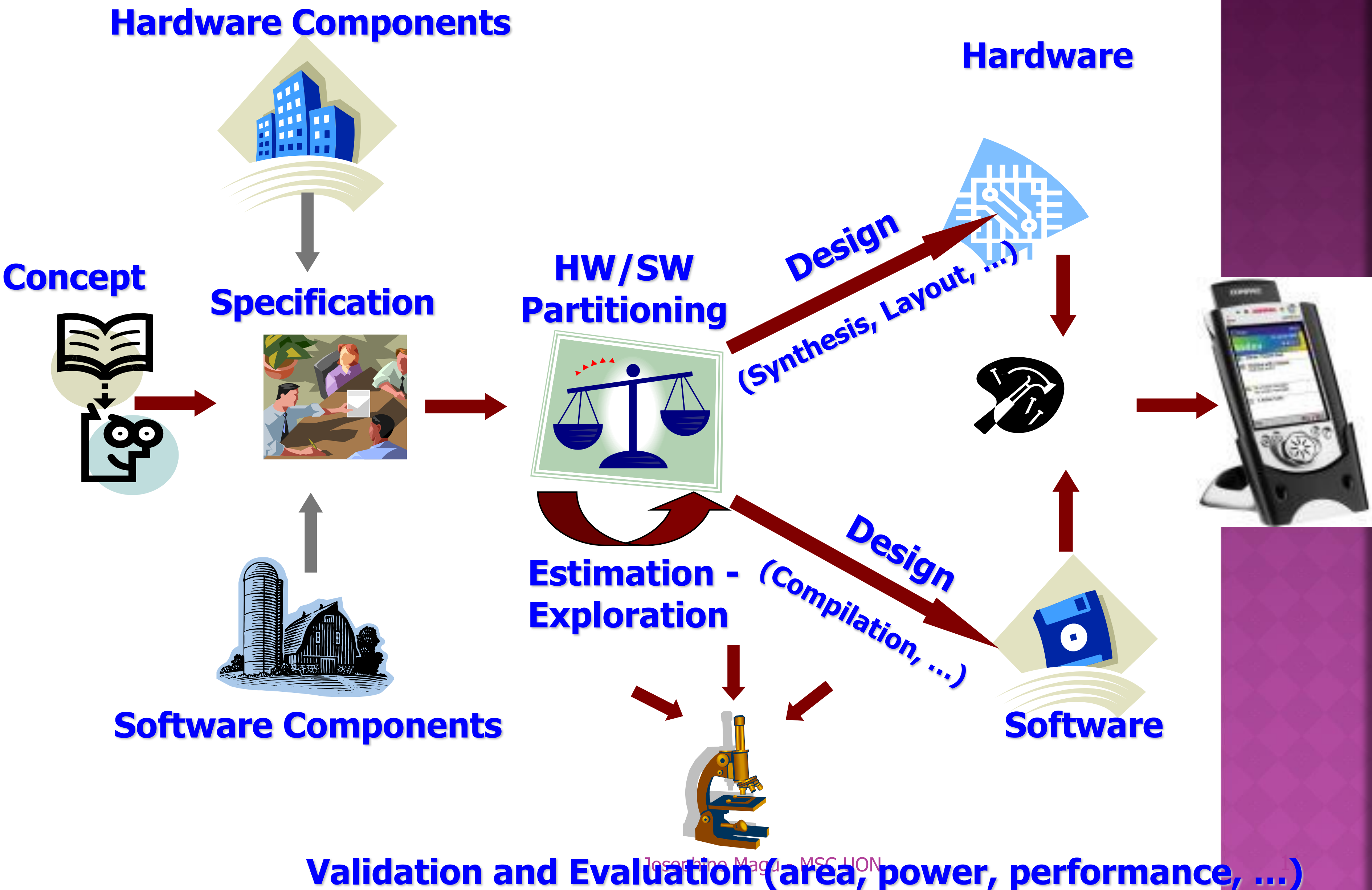




ES: Simplified Block Diagram



COURSE OUTLINE



COMPONENTS OF EMBEDDED SYSTEMS

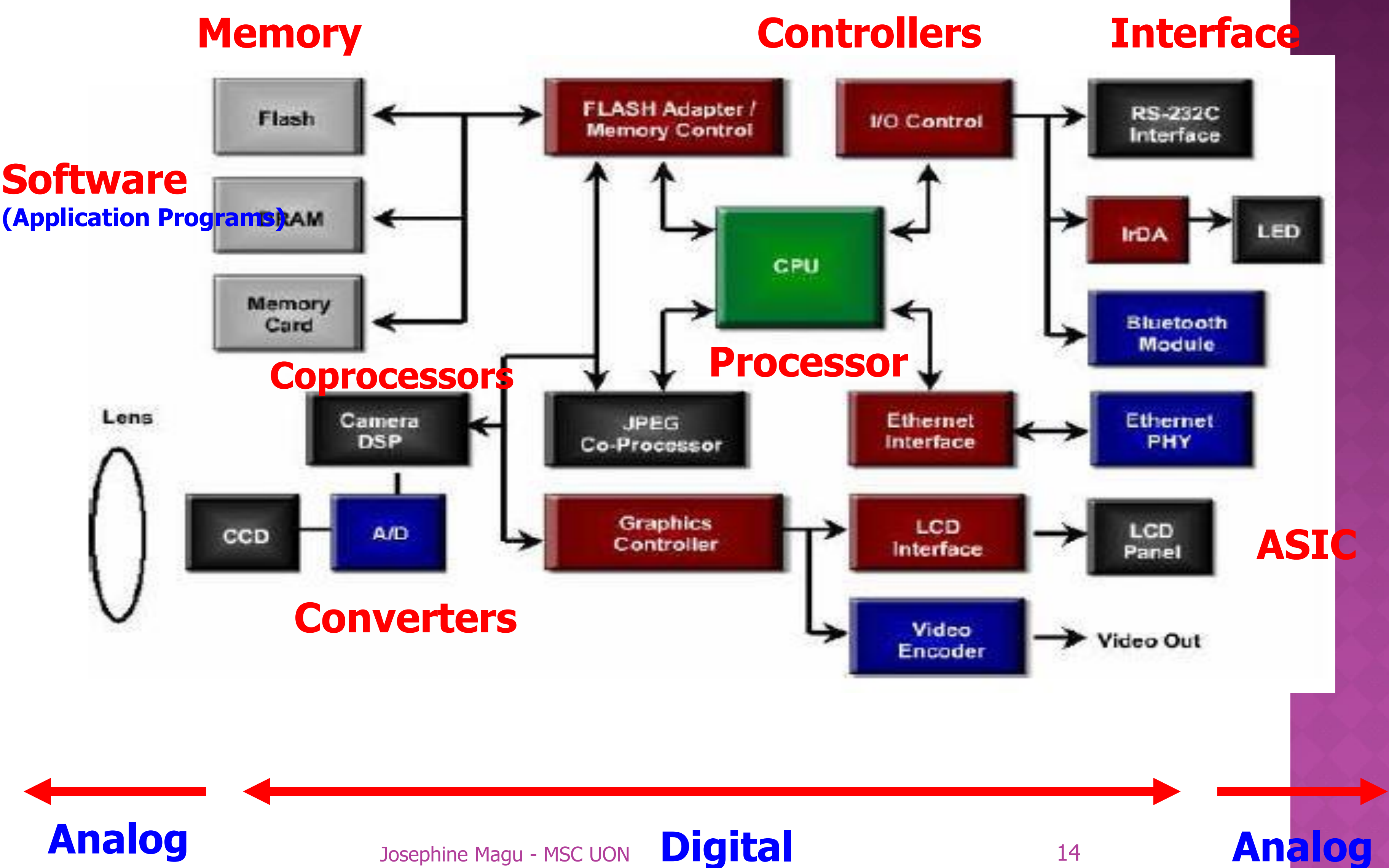
- Analog Components
  - Sensors, Actuators, Controllers, …
- Digital Components
  - Processor, Coprocessors
  - Memories
  - Controllers, Buses
  - Application Specific Integrated Circuits (ASIC)
- Converters – A2D, D2A, …
- Software
  - Application Programs
  - Exception Handlers

Hardware

Software

HARDWARE COMPONENTS

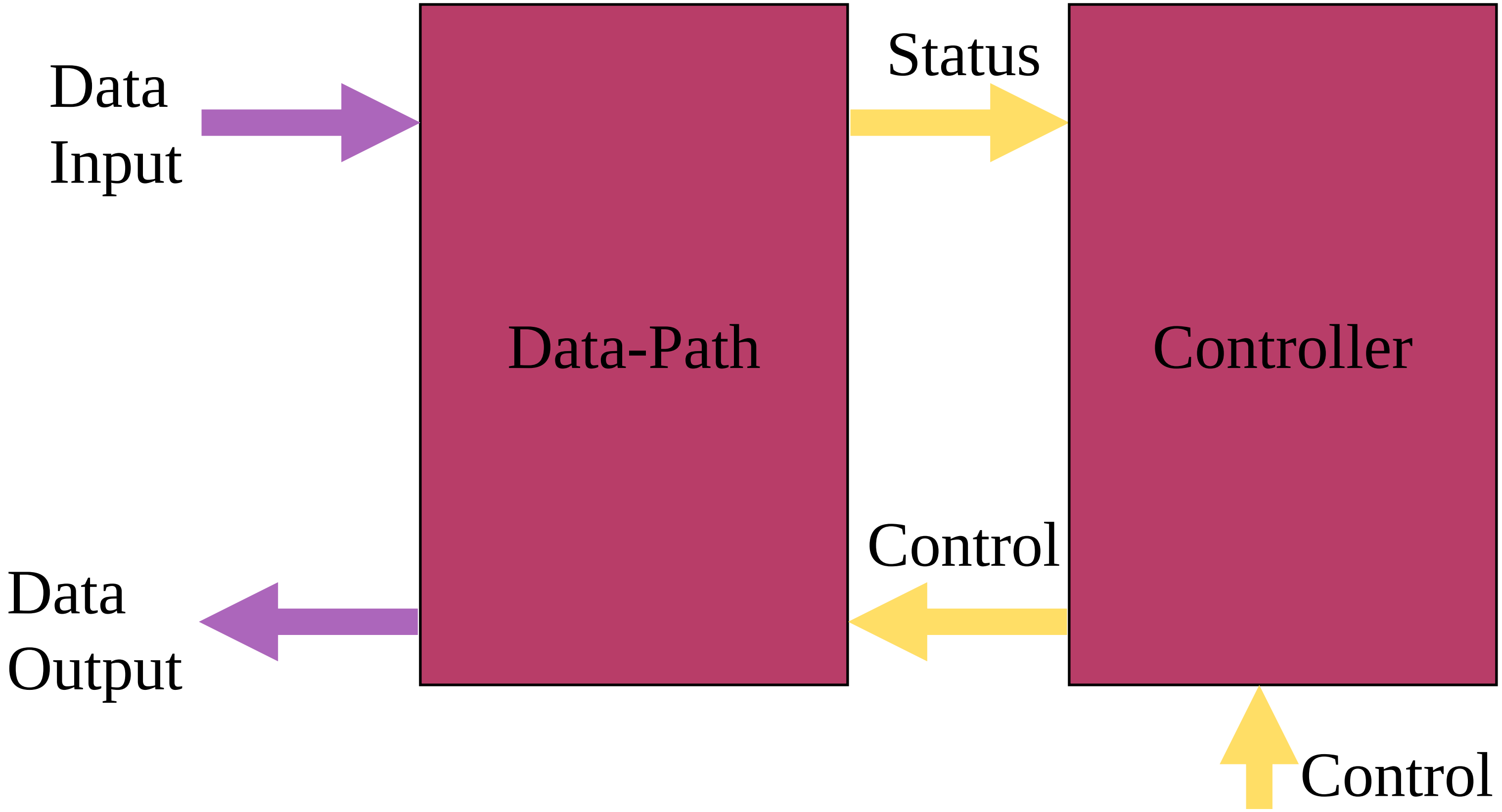
HARDWARE COMPONENTS OF EMBEDDED SYSTEMS- AN EXAMPLE



PROCESSORS

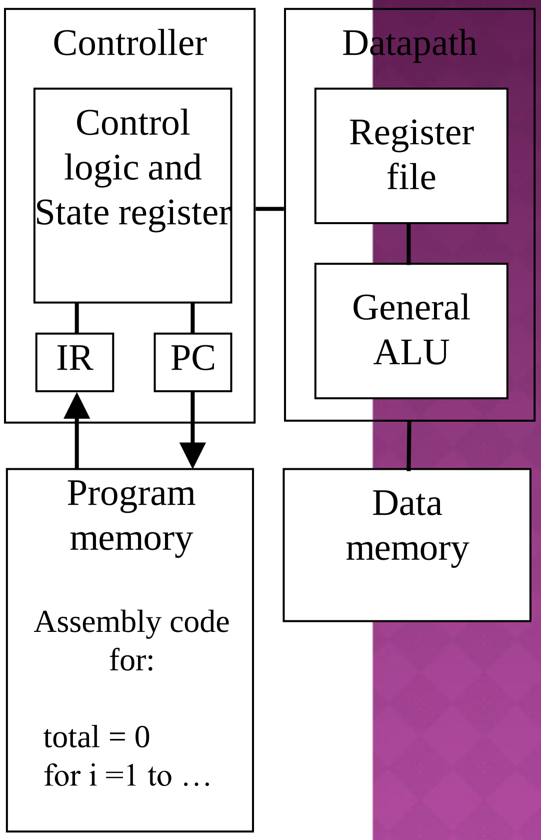
- What is a processor?
  - Artifact that computes (runs algorithms)
  - Controller and data-path
- General-purpose (GP) processors:
  - Variety of computation tasks
  - Functional flexibility and low cost at high volumes (maybe)
  - Slow and power hungry
- Single-purpose (SP) processors (or ASIC)
  - One particular computation task
  - Fast and power efficient
  - Functional inflexibility and high cost at low volumes (maybe)

GP/SP PROCESSOR ARCHITECTURE



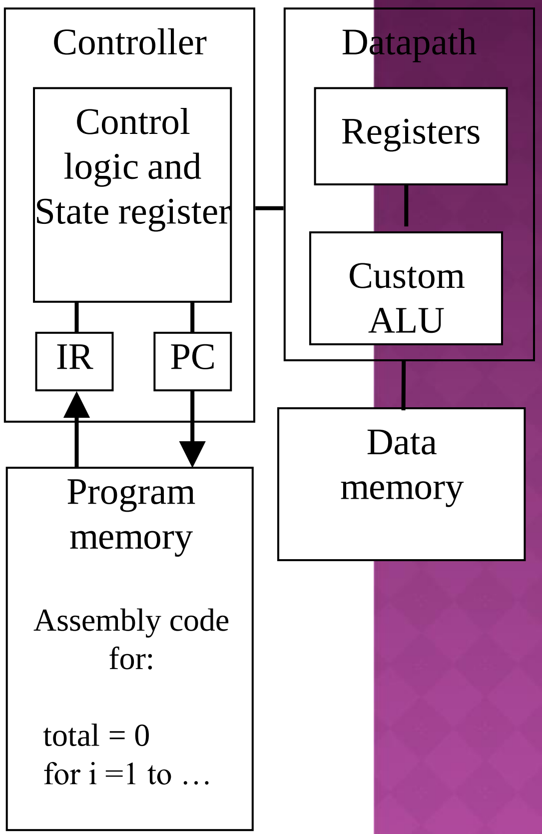
GENERAL-PURPOSE PROCESSORS

- Programmable device used in a variety of applications
  - Also known as “microprocessor”
- Features
  - Program memory
  - General datapath with large register file and general ALU
- User benefits
  - Low time-to-market and NRE costs
  - High flexibility
- Examples
  - Pentium, Athlon, PowerPC



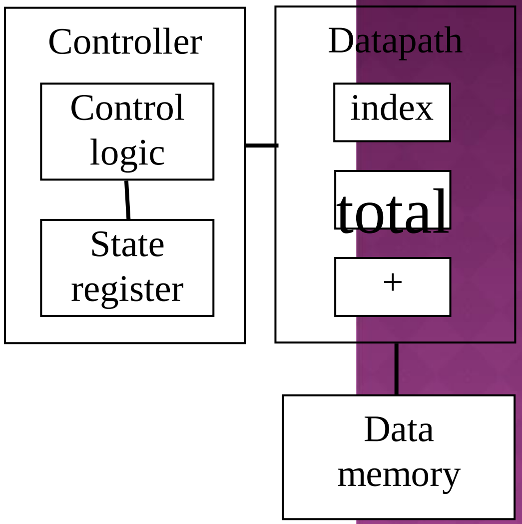
APPLICATION-SPECIFIC IS PROCESSORS (ASIPS)

- Programmable processor optimized for a particular class of applications having common characteristics
  - Compromise between general-purpose and ASIC (custom hardware)
- Features
  - Program memory
  - Optimized datapath
  - Special functional units
- Benefits
  - Some flexibility, good performance, size and power
- Examples
  - DSPs, Video Signal Processors, Network Processors,..



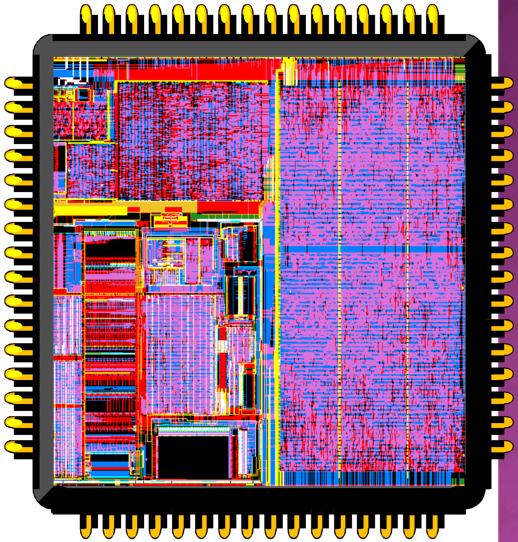
APPLICATION-SPECIFIC ICS (ASICS)

- Digital circuit designed to execute exactly one program
  - coprocessor, hardware accelerator
- Features
  - Contains only the components needed to execute a single program
  - No program memory
- Benefits
  - Fast
  - Low power
  - Small size



APPLICATION SPECIFIC CIRCUITS (ASIC)

- Custom-designed circuits necessary if ultimate speed or energy efficiency is the goal and large numbers can be sold.
- Approach suffers from long design times and high costs.



GP VS. SP PROCESSORS

GP:

- Programmable controller
  - Control logic is stored in memory
  - Fetch/decode overhead
- Highly general data-path
  - Typical bit-width (8, 16, 32, 64)
  - Complete set of arithmetic/logic units
  - Large set of registers
- High NRE/sale-volume

ASIC:

- Hardwired controller
  - No need for program memory and cache
  - No fetch/decode overhead
- Highly tuned data-path
  - Custom bit-width
  - Custom arithmetic/logic units
  - Custom set of registers
- Low NRE/sale-volume

STORAGE

- What is a memory?
  - Artifact that stores bits
  - Storage fabric and access logic
- Write-ability
  - Manner and speed a memory can be written
- Storage-permanence
  - ability of memory to hold stored bits after they are written
- Many different types of memories
  - Flash, SRAM, DRAM, etc.
- Common to compose memories

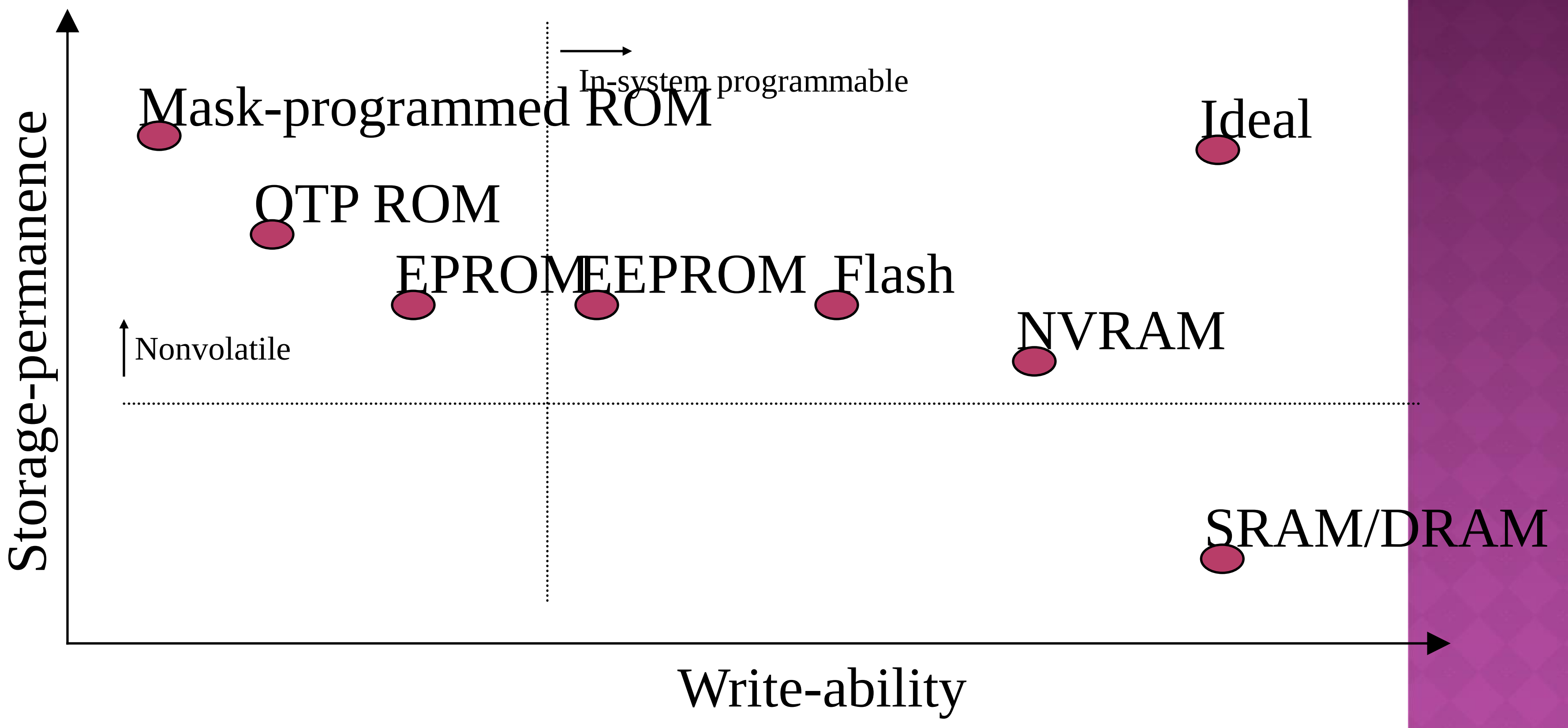
WRITE-ABILITY

- Ranges of write ability
  - High end
    - Processor writes to memory simply and quickly
    - E.g., RAM
  - Middle range
    - Processor writes to memory, but slower
    - E.g., FLASH, EEPROM
  - Lower range
    - Special equipment, "programmer", must be used to write to memory
    - E.g., EPROM, OTP ROM
  - Low end
    - Bits stored only during fabrication
    - E.g., Mask-programmed ROM

STORAGE-PERMANENCE

- Range of storage permanence
  - High end
    - Essentially never loses bits
    - E.g., mask-programmed ROM
  - Middle range
    - Holds bits days/months/years after memory's power source turned off
    - E.g., NVRAM
  - Lower range
    - Holds bits as long as power supplied to memory
    - E.g., SRAM
  - Low end
    - Begins to lose bits almost immediately after written
    - E.g., DRAM

MEMORY TYPES



COMMUNICATION

- What is a bus?
  - An artifact that transfers bits
  - Wires, air, or fiber and interface logic
- Associated with a bus, we have:
  - Connectivity scheme
    - Serial Communication
    - Parallel Communication
    - Wireless Communication
  - Protocol
    - Ports
    - Timing Diagrams
    - Read and write cycles
  - Arbitration scheme, error detection/correction, DMA, etc.

SERIAL COMMUNICATION

- A single wire used for data transfer
- One or more additional wires used for control (but, some protocols may not use additional control wires)
- Higher throughput for long distance communication
  - Often across processing node
- Lower cost in terms of wires (cable)
- E.g., USB, Ethernet, RS232, I^2C, etc.

PARALLEL COMMUNICATION

- Multiple buses used for data transfer
- One or more additional wires used for control
- Higher throughput for short distance communication
  - Data misalignment problem
  - Often used within a processing node
- Higher cost in terms of wires (cable)
- E.g., ISA, AMBA, PCI, etc.

WIRELESS COMMUNICATION

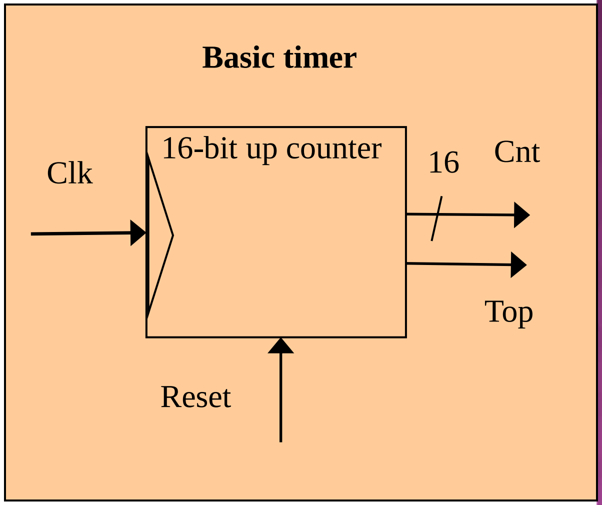
- Infrared (IR)
  - Electronic wave frequencies just below visible light spectrum
  - Diode emits infrared light to generate signal
  - Infrared transistor detects signal, conducts when exposed to infrared light
  - Cheap to build
  - Need line of sight, limited range
- Radio frequency (RF)
  - Electromagnetic wave frequencies in radio spectrum
  - Analog circuitry and antenna needed on both sides of transmission
  - Line of sight not needed, transmitter power determines range

PERIPHERALS

- Perform specific computation task
- *Custom* single-purpose processors
  - Designed by us for a unique task
- *Standard* single-purpose processors
  - “Off-the-shelf”
  - pre-designed for a common task

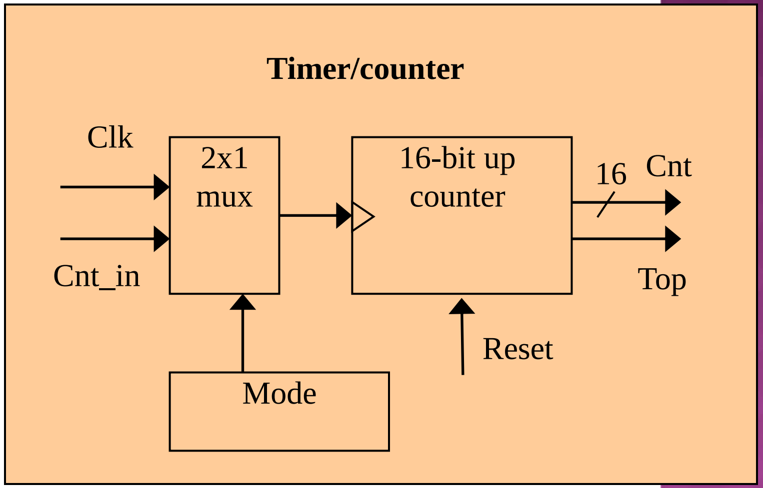
TIMERS

- Timers: measure time intervals
  - To generate timed output events
  - To measure input events
  - Top: max count reached
- Range and resolution



COUNTERS

- Counter: like a timer, but counts pulses on a general input signal rather than clock
  - e.g., count cars passing over a sensor
  - Can often configure device as either a timer or counter



WATCHDOG TIMER

- Must reset timer every X time unit, else timer generates a signal
- Common use: detect failure, self-reset

UART

- UART: Universal Asynchronous Receiver Transmitter
  - Takes parallel data and transmits serially
  - Receives serial data and converts to parallel
- Parity: extra bit for simple error checking
- Start bit, stop bit
- Baud rate
  - Signal changes per second
  - Bit rate, sometimes different

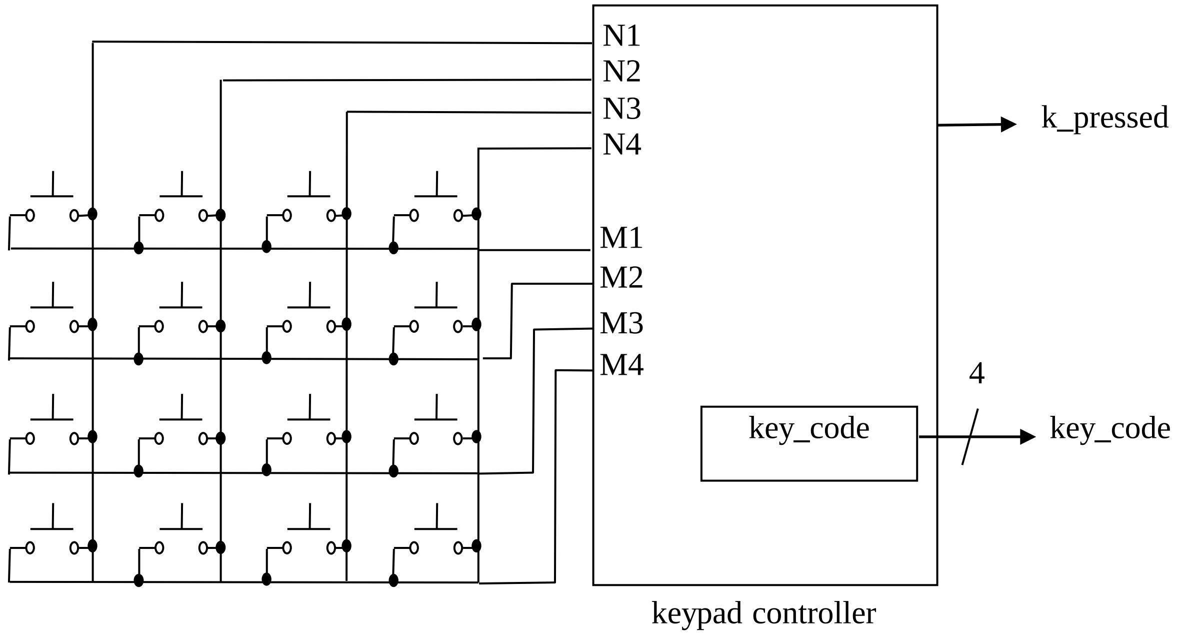
PULSE WIDTH MODULATOR (PWM)

- Generates pulses with specific high/low times
- Duty cycle: % time high
  - Square wave: 50% duty cycle
- Common use: control average voltage to electric device
  - Simpler than DC-DC converter or digital-analog converter
  - DC motor speed, dimmer lights

LCD

- Liquid Crystal Display
- N rows by M columns
- Controller build into the LCD module
- Simple microprocessor interface using ports
- Software controlled

KEYPAD

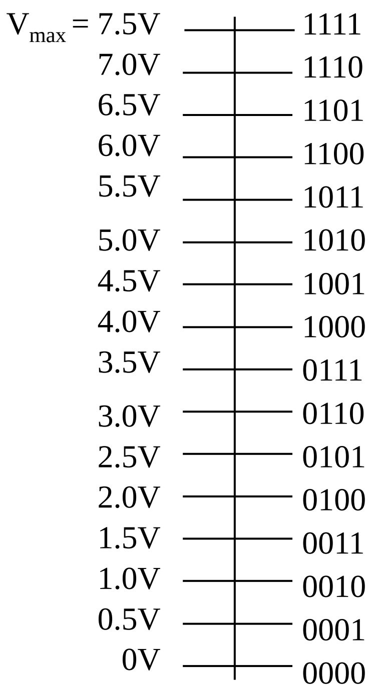

N1
N2
N3
N4
M1
M2
M3
M4
k_pressed
4
key_code
key_code
keypad controller

N=4, M=4

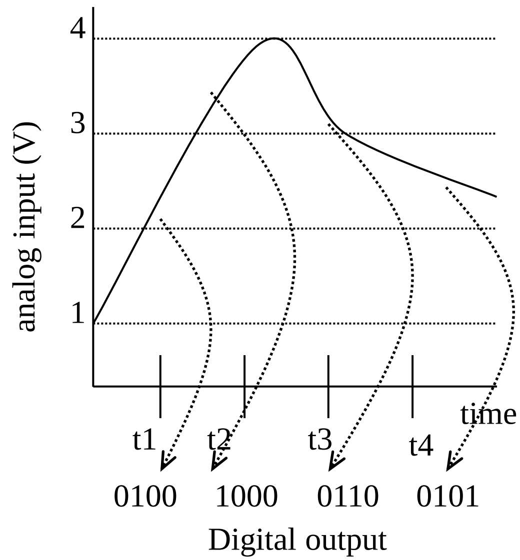
STEPPER MOTOR CONTROLLER

- Stepper motor: rotates fixed number of degrees when given a “step” signal
  - In contrast, DC motor just rotates when power applied, coasts to stop
- Rotation achieved by applying specific voltage sequence to coils
- Controller greatly simplifies this

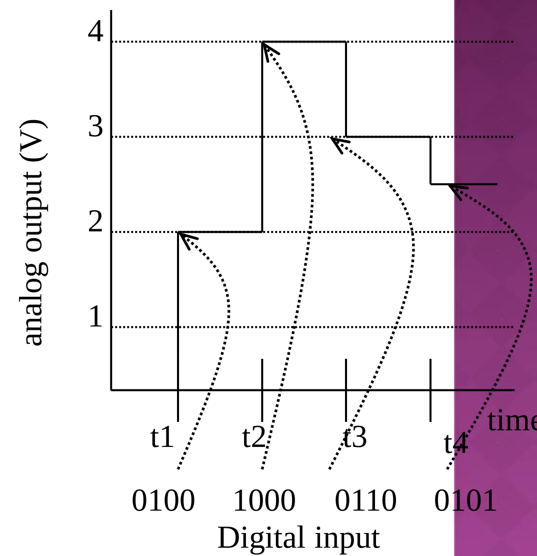
ANALOG-TO-DIGITAL CONVERTER



proportionality



Analog to Digital (A/D)



Digital to Analog (D/A)

THE NUMBER GAME (1)

1	3	5	7
9	11	13	15
17	19	21	23
25	27	29	31

THE NUMBER GAME (2)

2	3	6	7
10	11	14	15
18	19	22	23
26	27	30	31

THE NUMBER GAME (4)

4	5	6	7
12	13	14	15
20	21	22	23
28	29	30	31

THE NUMBER GAME (8)

8	9	10	11
12	13	14	15
24	25	26	27
28	29	30	31

THE NUMBER GAME (16)

16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31

ANALYSIS

- What is theory /concept behind this game?
- How did I arrive at the number you guessed?
- How can I automate this process?
- What is the data and what is the algorithm?
- How can we convey these to a computing machine?
- While a computer talks binary, we humans write programs in languages such as Java, C#, C++, Basic etc.

Binary numbers (1's and 0's) is the number system used by the computer systems.

We humans use decimal number system that has 10 distinct symbols (0,1,2,3,4,5,6,7,8,9)

Your task: Write a C program to computerize this

1	3	5	7
9	11	13	15
17	19	21	23
25	27	29	31

2	3	6	7
10	11	14	15
18	19	22	23
26	27	30	31

4	5	6	7
12	13	14	15
20	21	22	23
28	29	30	31

8	9	10	11
12	13	14	15
24	25	26	27
28	29	30	31

16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31

REQUIREMENTS-ENGINEERING PROCESS

- Deals with determining the goals, functions, and constraints of systems, and with representation of these aspects in forms amenable to modeling and analysis.

TYPES OF REQUIREMENTS

- Standard scheme for realtime systems is defined by IEEE standard IEEE830.
- It defines the following kind of requirements:

    I. Functional

    II. Non-functional

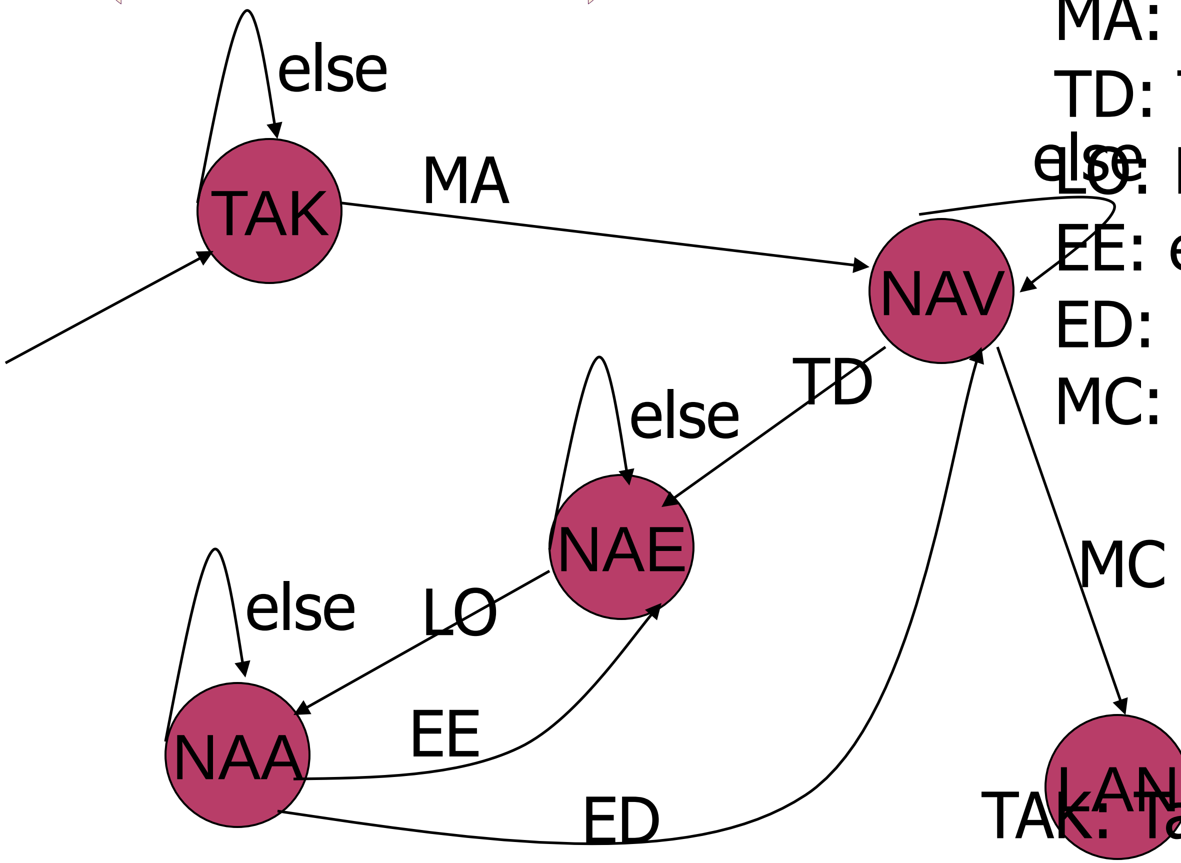
        1. External interfaces
        2. Performance
        3. Logical database
        4. Design constraints (ex: standards compliance)
        5. Software system attributes

        Reliability, availability, security, maintainability, portability

DESIGN METHODS: FINITE STATE MACHINES

- Finite state automaton (FSA), finite state machine (FSM) or state transition diagram (STD) is a formal method used in the specification and design of wide range of embedded and realtime systems.
- The system in this case would be represented by a finite number of states.
- Lets design the avionics for a drone aircraft.

DRONE AIRCRAFT AVIONICS (SIMPLIFIED)



MA: Mission Assigned
TD: Target Detected
LO: Locked On
EE: enemy Evaded
ED: Enemy Destroyed
MC: Mission Complete

TAK: Take off
NAV: Navigate
NAE: Navigate & Evade
NAA: Navigate & Attack
LAN: Land

FINITE STATE MACHINE (FSM)

- M = five tuple ➔ { S, i, T, Σ, δ }
- S = set of states
- i = initial state
- T = terminal state (s)
- Σ = events that bring about transitions
- δ = transitions
- Lets do this exercise for the avionics for fighter aircraft

STATE TRANSITION TABLE

	MA	LO	TD	MC	EE	ED
TAK	NAV					
NAV			NAE	LAN		
NAE		NAA				
NAA					NAE	NAV
LAN						

LETS DESIGN A SIMPLE EMBEDDED/ REALTIME SYSTEM

- Use the table to code a function with case statement
- Or write a table-driven code
- Which is better and why?
- Try your binary game using FSM method

SUMMARY

- We examined the course objectives for embedded and realtime systems
- We looked at sample systems
- Homework#1:
  1. Write a program that automates the number game
  2. Write a program that simulates the drone
  3. Write C program, compile, debug, test and submit the programs online.
  4. submit-cse321 yourInitialsHwk1_1.c yourInitialsHwk1_2.c

SUMMARY

- Hardware: Information Processing
  - Processors
  - Memories
  - Communication
  - Peripherals
  - Design methods:FSM
  - Analysis
  - Required engineering-
  - State transition table