CE 654 – Embedded Systems

Spring 2008

Nikos Bellas

Computer and Communications Engineering Department
University of Thessaly

Administrivia

- Instructor: Νίκος Μπέλλας
- Email: nbellas@inf.uth.gr
- Class web page
  http://support.inf.uth.gr/courses/CE654/index.html
- Office: Ιάσωνος 10, 5ος όροφος –
- Phone #: 4704
- Office Hours: TBA
- NOTE: Please send me your email by the end of the week

Prerequisites for CE654

- CE 232 - “Computer Systems Organizations” or similar
- CE 432 – “Computer Architecture” preferable
- Good programming skills (in C)
- Digital and logic design courses necessary
- All slides, reference papers, data sheets etc. will be in English

- Students are expected to complete a project of significant scope performed over the semester

References

- There is no required textbook for the class
- The lectures are based on a variety of resources such as textbooks, papers, and product data sheets.
- I will be using bits and pieces from the following texts:
  - *"Enbedded Computing", by Josh Fisher, Paolo Faraboschi, Cliff Young, Morgan Kaufmann Publishers, 2005*
  - *"Computer Architecture: A Quantitative Approach", by J. Hennessy, D. Patterson, Morgan Kaufmann Publishers, 3rd or 4th edition*
- Advice: Internet is a vast resource of information on embedded systems. You should use it

Grading (1)

- All the grading will be based on your projects
- The basic premise is to help the instructor prepare a set of lab assignments to be used for the class in subsequent years
- There will be no exams or homeworks

Grading (2)

- **45% : Final demo and presentation of your work (28-30/5)**
  - The entire team must be present for, and participate equally in, the demo and presentation. The outline for the final presentation will be provided. Your team will build up to this presentation through two prior presentations
- **20% : Mid-semester demo and presentation of your work (tentatively, 8/4)**
  - Another presentation will introduce your project to the class (7/3)
- **30% : Online documentation of the project**
- **5% : Class participation**

Curriculum (1)

- Introduction to embedded systems
- Specification and modeling of embedded systems
- Verification and co-simulation of embedded systems - SW/HW partition
- Reduced Instruction Set Computing (RISC) machines – Case study
- Application Specific Instruction-set Processors (ASIP) - Extensible processors (Tensilica Xtensa)
- Very Large Instruction Word (VLIW) processors.
- Reconfigurable systems (FPGAs, structured ASICs)
- Stream-based computing

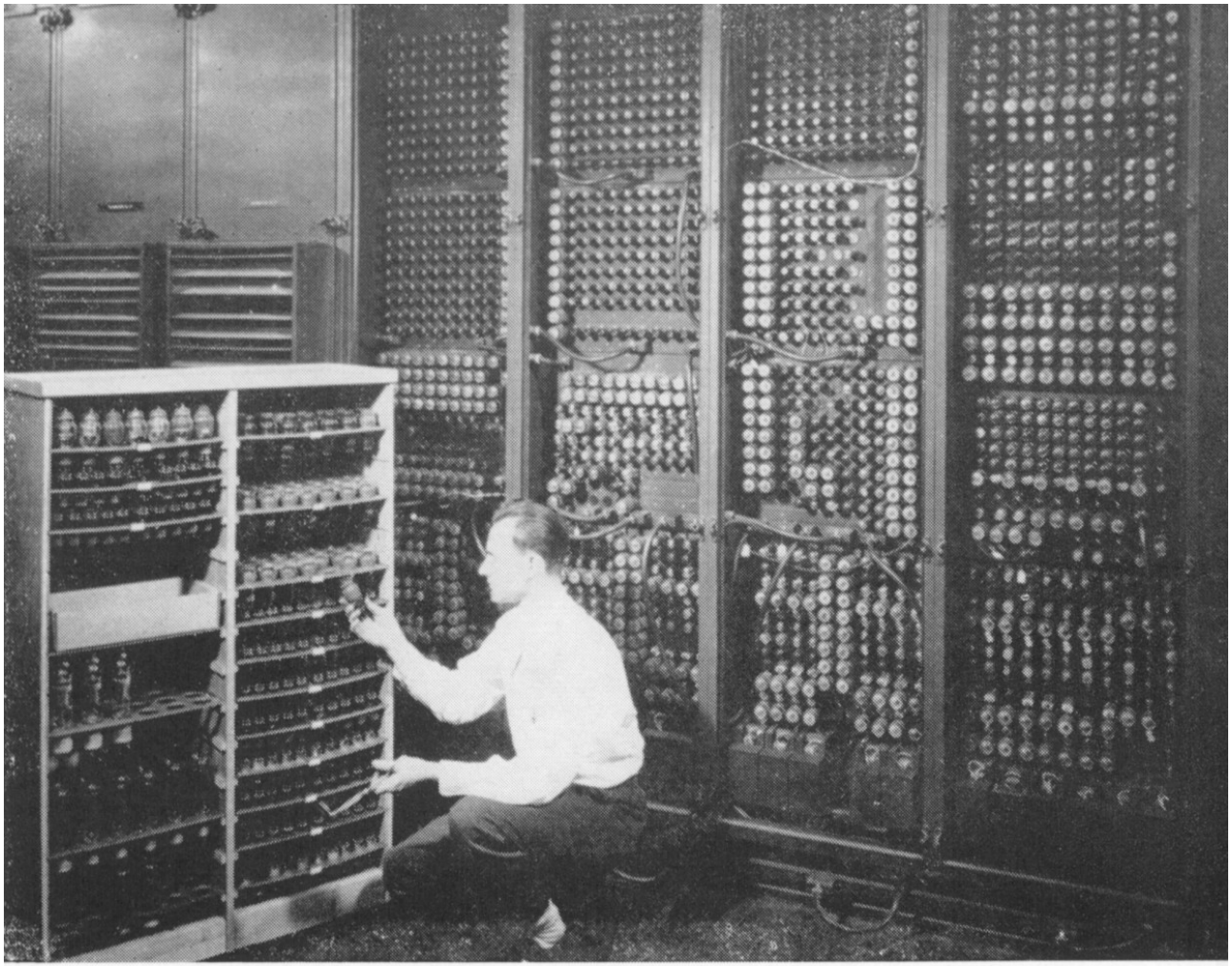
Curriculum (2)

- Communication in embedded systems - buses -switches – Network On Chips
- Putting it all together: System-on-chip design and prototyping platforms - Heterogeneous multiprocessors – Case studies
- Real time systems (Real time Scheduling, Interrupts, RTOS)
- Case studies of SoCs

Lecture 1

Introduction to Embedded Systems

The Past…

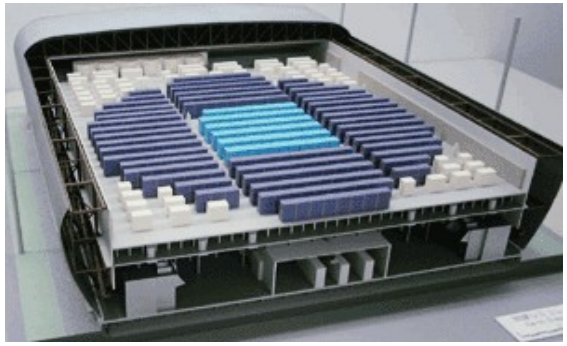


Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.

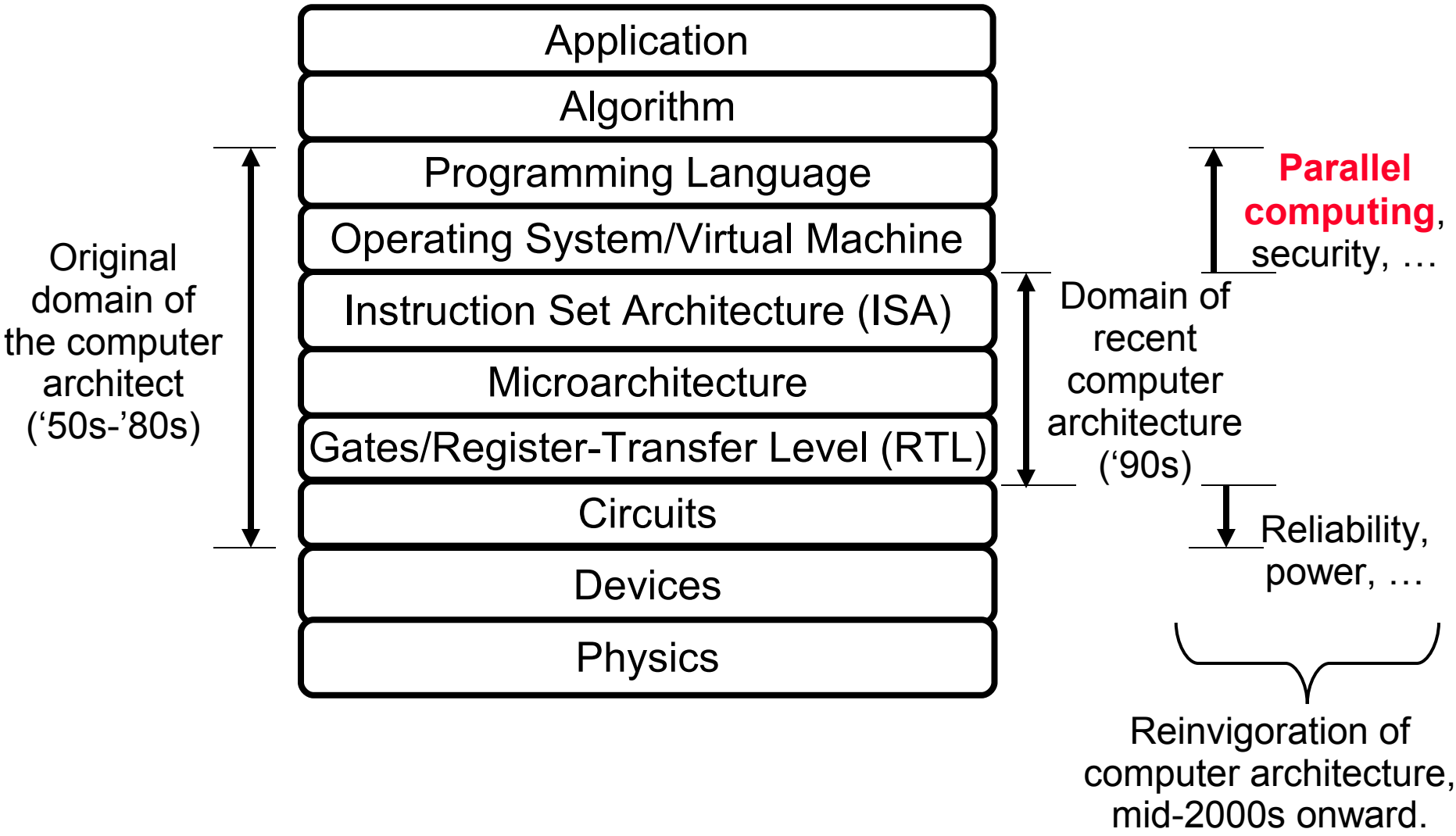
ENIAC, "US Army photo, around 1946"

10

The present…



Abstraction Layers in Modern Systems



What is Computer Architecture

- Computer architecture is a description of the structure and the functionality of a computer system.

- Computer architecture comprises at least three main subcategories:

- Instruction set architecture (ISA), also known as assembly language is the lowest point of control of the programmer on the processor.
- Microarchitecture or Computer Organization is at a lower level description of the system. What are the modules of the system, how they interconnect and how they interact. Microarchitecture is beyond the control of the programmer. For example, the number of functional units in a CPU is a microarchitectural detail.
- System Design which includes all of the other hardware components within a computing system such as system interconnects, memory hierarchies, peripherals, etc. System design is sometimes visible to the programmer.

ISA vs. Computer Architecture

- Old definition of computer architecture = instruction set design
  - Other aspects of computer design called implementation
  - Insinuates implementation is uninteresting or less challenging
- Today computer architects do much more. Technical hurdles *more* challenging than in the earlier days
- Two very important trends:
  - Implementation of microarchitecture has become critical, and more so as technology scales down
  - What really matters now is the whole system, NOT only the CPU (end-to-end system design). Computer architecture is an *integrated approach*
- All these are in the plate of the computer architect.

What is an Embedded System?

- Simple answer:
  - Any system that is not desktop is traditionally called embedded
- More complex answer:
  - A special purpose computing system built to perform one or more set of dedicated functions, sometimes with real-time constraints.
  - It is usually embedded within a larger system

Embedded System examples

Mobile telephony:

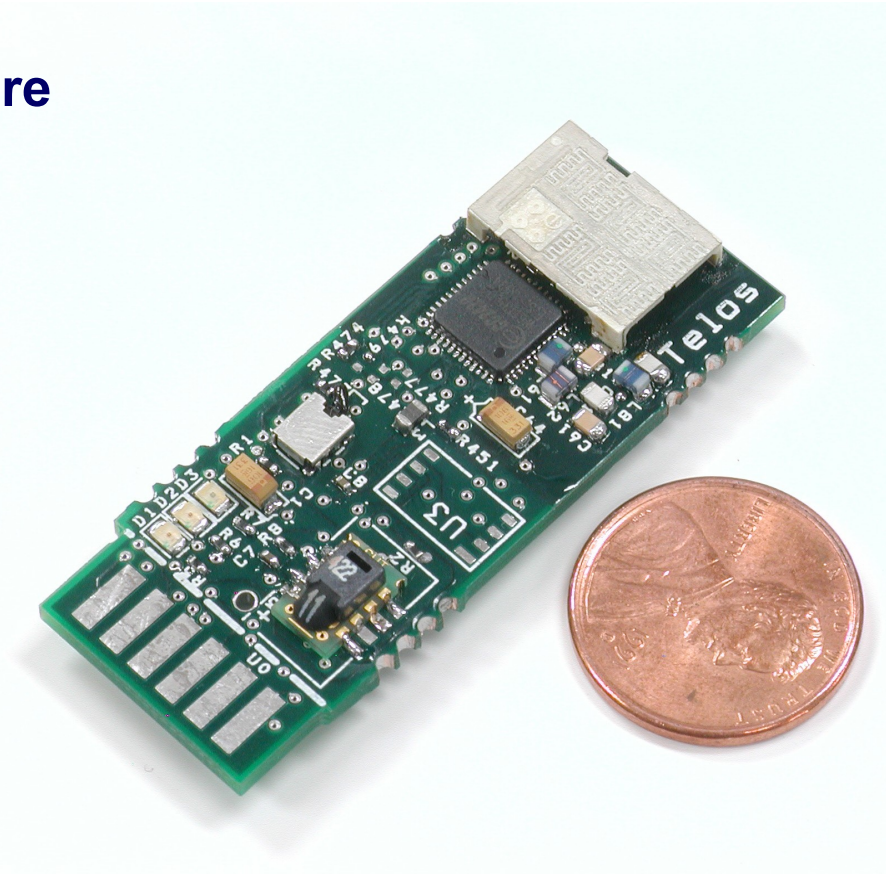
- Analog part for radio
  - Antenna, RF amplifier, mixer, modulator/demodulator, filter
- Digital part for DSP and applications
  - Heterogeneous multiprocessor system includes a DSP processorsignal processing and a microprocessor for applications.
  - Sometimes, a graphics processors for high-end gaming
  - Embedded camera, bluetooth, mic, speaker, etc.

Playstation3 video console

- 3.2 GHz Cell processor
  - Consists of 8 SIMD processors and 1 PowerPC
  - Used for graphics, multimedia processing and OS
- 550 Mhz Nvidia “Reality synthesizer”
  - Used for graphics rendering
- Blue-ray DVD, Super Audio CD, Hard drive, etc.
- In the borders between embedded and general purpose system

Sensor networks

- Single board philosophy
  - » Robustness, Ease of use, **Lower Cost**
  - » Integrated **Humidity & Temperature** sensor
- First platform to use **802.15.4**
  - » CC2420 radio, 2.4 GHz, **250 kbps** (12x mica2)
  - » 3x RX power consumption of CC1000, 1/3 turn on time
  - » Same TX power as CC1000
- Motorola HCS08 processor
  - » **Lower power consumption, 1.8V operation**, faster wakeup time
  - » **40 MHz** CPU clock, 4K RAM
- Support in upcoming TinyOS 1.1.3 Release
- Codesigned by UC Berkeley and Intel Research



Embedded System Constraints

Design optimal device that meets constraints on


Price


Functionality


Performance


Size


Power

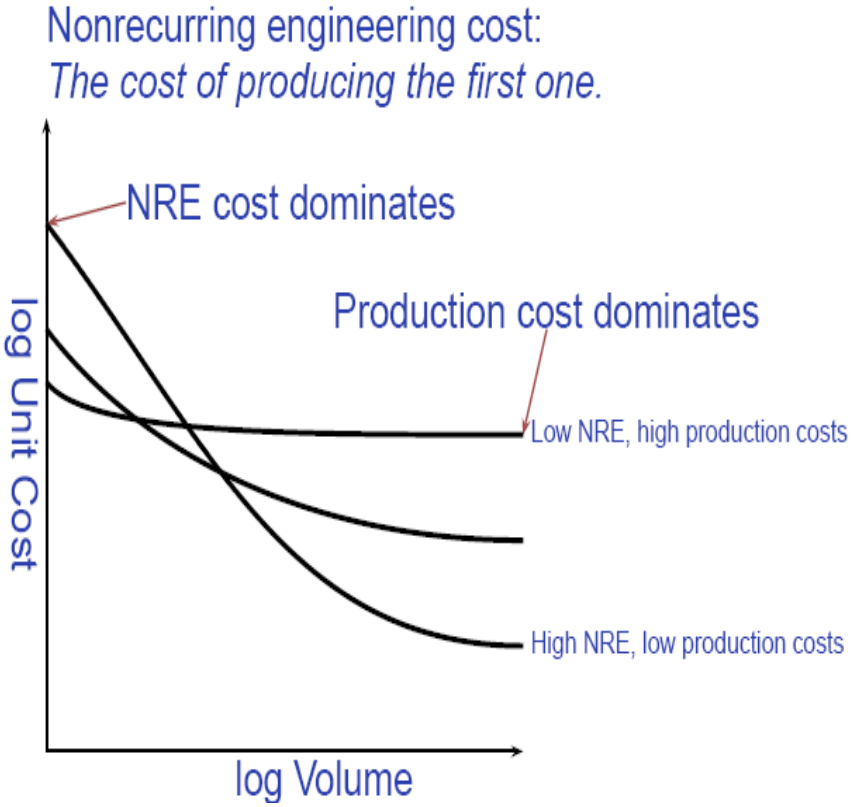

Time-to-market


Maintainability


Safety

Price constraints

- Embedded systems are often mass produced in highly competitive markets where low cost is the main constraint (e.g. cell phones)
- Two sources of cost:
- Production cost includes the Bill Of Materials (BOM), manufacturing, marketing, shipping, etc.
- Non recurring costs (NRE) includes engineering costs (salaries, training, etc.) as well as prototype development, mask set costs for SoC design, capital expenditure for the design, etc



Performance constraints

- Most of the embedded systems have to perform under real-time constraints which means that if data are not ready by some specific deadline, the system fails
- A real-time constraint is called hard if a failed deadline results into system failure.
  - For example, a system that controls the deployment of airbags in a car accident
- A real-time constraint is called soft if a failed deadline results into operational degradation, but not failure.
- Most embedded systems have real time requirements, and all real-time systems are embedded.

Power dissipation constraints

- High power consumption leads to lower battery life and higher system costs
  - Critical in systems such sensor networks, mobile communications, etc.
  - For such systems, power efficiency is equally, if not more, important than performance
- High power dissipation needs stronger power supply and expensive colling system
- High power dissipation affects system reliability and reduce mean time to failure (MTTF)

Maintanability

- A characteristic of design expressed as the probability that an item will be retained in or restored to a specified condition within a given period of time, when the maintenance is performed in accordance with prescribed procedures and resources.
- The ease with which maintenance of a functional unit can be performed in accordance with prescribed requirements.

- Some systems are “deploy and forget”, i.e. their maintainance is impossible (e.g. some cases of sensors) or it is very expensive and/or time consuming (e.g the Mars Pathfinder system).

Functionality

- Most embedded systems are optimized for one or a small set of functionalities.
- Good designs do few things well
  - Functionality overloading causes cost overruns, schedules to slip, or even unsatisfied customers
- High-end embedded systems , on the other hand, trade off programmability and performance.
  - Video consoles offer general purpose APIs and are used as general purpose multimedia hubs.
  - Identify the applications that can benefit from the supercomputing capabilities of video consoles without extra cost increases.

Size and Weight

- Size and weight is critical for mobile embedded systems, such as cell phones, cameras, etc.
- Very critical for some medical applications, such as pills with embedded camera and data acquisition system.

Time to market

- In highly competitive markets, it is critical to catch the market window. A short delay or a premature deployment may have catastrophic consequences, even if the product is of high quality.

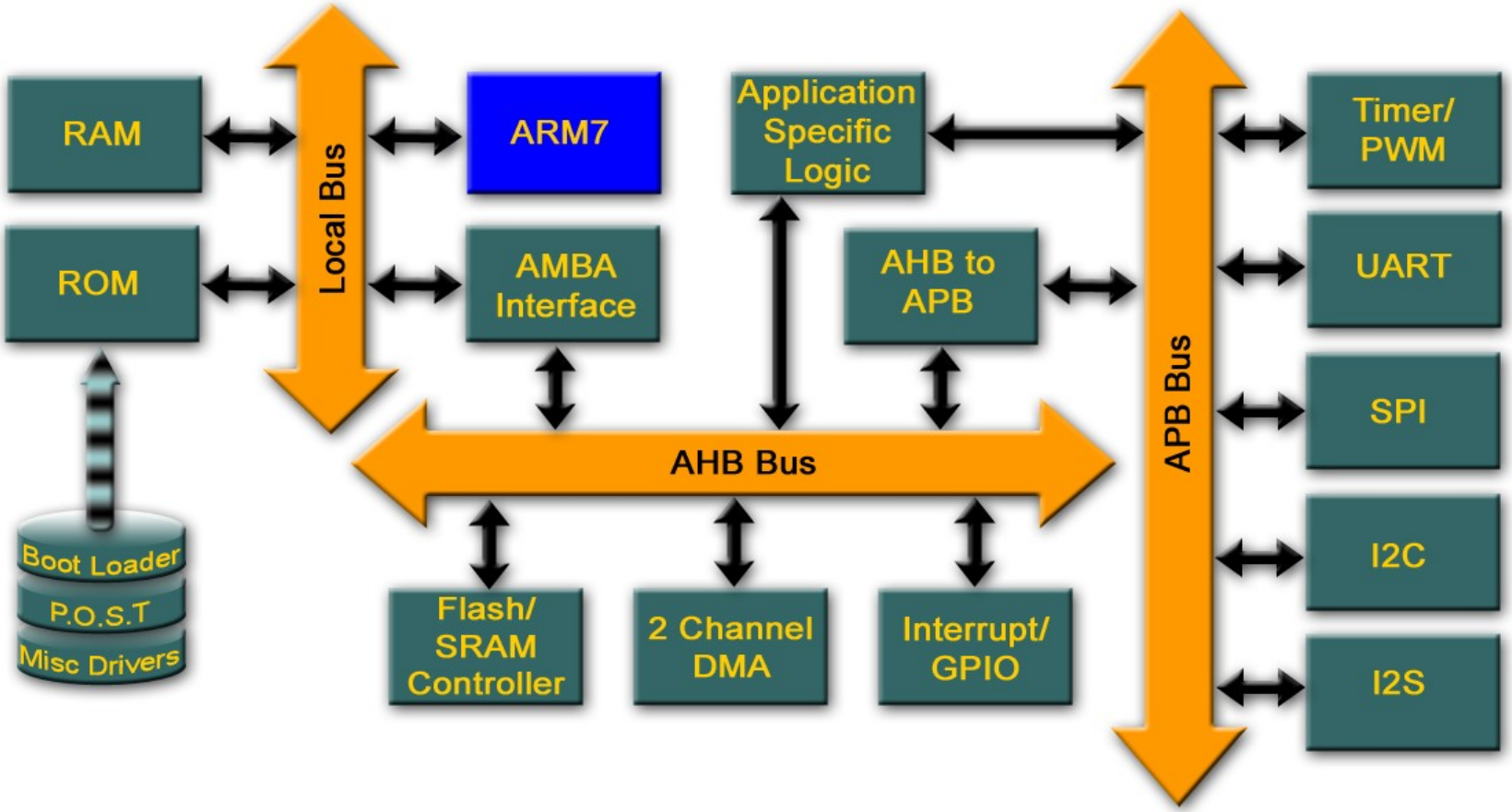
- Development time can be reduced by:
  - Efficient and robust design methodologies and tools
  - Reuse of previously verified hardware and software components
  - Use of early prototyping platforms
  - Design team understanding the system requirements EARLY

Safety and Security

- Embedded systems are often used in life critical situations (automotive or aviation electronics, nuclear plants, medical applications).
  - Verification that the system is working even under abnormal input stimuli is required in most such cases.

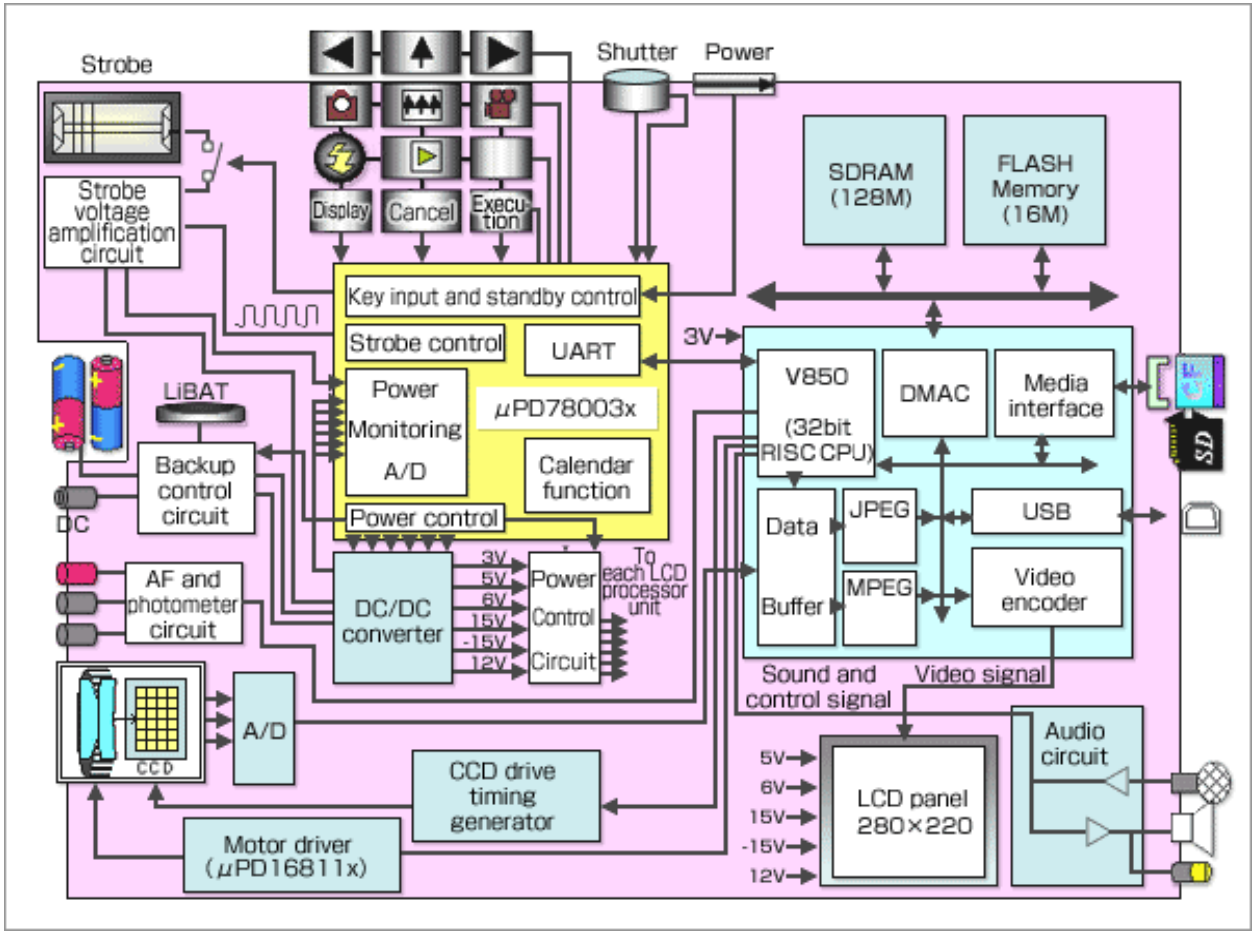
- Embedded systems for communication may have to support confidentiality and authentication.

Embedded System block diagram



The focus is on the system, not the processor

Digital camera SoC



- Completely distributed system

Conclusions

- Embedded systems are dedicated and application-specific, require continuous interaction with the environment at real-time, and have to meet multiple constraints
- Such constraints include cost, performance, power, maintanability, functionality trade offs, size/weight, time to market and safety.
- Designers are asked to balance these often conflicting requirements
- Much design effort is often required to handle the complexity of today's embedded systems and applications.