Attributes of Embedded Systems

Cuauhtémoc Carbajal

Reference:
Embedded systems are designed to perform a specific task or function.

- The software developed for the platform is specific to the function of the overall device.
- In many cases the compute capability of the device is invisible to the end user; the function provided by the device is all that matters.
- End users of embedded systems do not generally install application software on such devices (although they may have limited ability to upgrade the software).
Embedded systems cover the entire landscape of devices and products.

Embedded systems can range from a simple stand-alone device to a chassis of networked cards to a system composed of many separate networked embedded elements.

They all work collaboratively to achieve the objectives of the overall system.

Such systems operate largely **autonomously** once set up.
In many cases, embedded systems are referred to as cyber-physical systems, where the system exercises significant physical interactions in the real world.

The security features of embedded devices are becoming a critical attribute to comprehend.

Embedded systems often have specific real-time constraints that must be adhered to.
Embedded Platform Characteristics
While 8-bit and 16-bit devices will remain in use for quite some time, it’s clear that there is also a large need for 32-bit devices in the embedded space.

Microcontrollers occupy a large portion of the embedded space. We will not focus on this class of device in this course. Instead, we are going to focus on

- a generally higher performance class of device,
- with sophisticated connectivity and graphics,
- that relies on a RTOS or a full-featured operating system such as an embedded Linux distribution.
As the expectations placed on embedded systems grow exponentially, the ability to ensure that the software scales with these expectations is critical.

In such cases full-featured operating systems such as Linux offer compelling capabilities.

These higher-performance devices are generally known as **embedded microprocessors**, as distinct from microcontrollers, and are dominated by 32-bit CPU architectures at this time.

The clock frequency range of an embedded microprocessor system is from the low hundreds of megahertz (200 MHz) to over 1GHz.

The overall architecture of the processor and the inclusion of caches (and their size) are critical aspects that contribute to overall system performance.
Another important characteristic is the level of parallelism offered by the processor.

- **Instruction-level parallelism**
  - A wide range of processor microarchitecture techniques can improve instruction-level parallelism, such as instruction pipelining, superscalar execution, and out-of-order execution.

- **Data-level parallelism**
  - The operation is applied on several data items instead of one (Single instruction multiple data: SIMD)
  - Useful in multimedia processing

- **Thread-level parallelism**
  - The execution units of a processor are shared between independent threads of the process (or even threads from different processes).

- **Process-level parallelism**
  - Different processes are executed on parallel processors or processor cores.
Many embedded applications require the use of floating-point arithmetic.

If floating-point operations are not an insignificant portion of the application workload, the process should include a hardware-based floating-point unit.

If the floating-point unit is provided as a hardware function, it is ideal if it conforms to the IEEE Standard for Floating-Point Arithmetic (IEEE 754); otherwise, you have to become very familiar with the deviations from such a standard.
The increasing demand for lower-cost, higher-density platforms and smaller form factors has driven the need to increase the level of integration for each of the devices that makes up the embedded platform.

As integration levels increase, more and more logic is added to the processor die, creating families of application-specific service processors.

The term system on chip (SOC) is often used to describe these highly integrated processors. These SOCs include much of the logic and interfaces that are required for a range of specific target applications.
SOCs integrate capabilities to connect the SOC to external memory devices and nonvolatile storage devices using glue-less interfaces.

SOCs provide segment- or application-specific interfaces:
- general purpose input/output pins;
- interfaces such as Ethernet, USB, PCIe, serial ports, I2C;
- expansion parallel buses; and
- integrated display controllers.

As a general rule, these integrated items are predominantly digital logic elements.
The power consumed by the devices is measured in many different ways:

Power consumption of the device
- measured by running an application on the processor that exercises a representative portion of the I/O capabilities.
- In many cases there are actually several different voltage rails in use, so this must be summed across all power supplies.

Total device power figure (TDP)
- TDP is the maximum amount of power the cooling system is required to dissipate.
- This does not mean that the system needs active power cooling, such as a fan. In many embedded cases a heat sink is all that is required.
- The TDP is used as part of an overall thermal design that must ensure that the CPU/SOC does not overheat.
- There can be a considerable delta between the average power dissipation of a system and the figures quoted for TDP.
- TDP is and must be an extremely conservative figure.
Power figures are often highly dependent on the activity level of the system.

A platform power measurement is not just the power of the processor/SOC; other aspects of the system can contribute significantly to the power (rich, colorful displays, for example).

The power of the system must also include loss in conversion of power from the primary single supply into the power rails needed by all components in the system.
The form factors for embedded systems are as diverse as the embedded use cases themselves.

A large number of embedded systems are composed of a single PCB and are often called single-board computers (SBCs).

provide a power connect for a single input voltage and connectors for devices such as mass storage SATA/SDIO, USB, and displays.

most cost-effective way to produce an embedded platform design for a specific target use case, but the platform is not readily upgradeable over time.

Given that many people and companies have a need for a generic compute module, there are a number of standard form factors and connector formats for such modules (PC/104)
PC/104

- Compact, 3.6-3.8 inch (90-96 mm) module size
- Self-stacking: expands without backplanes or card cages
- Rugged, reliable connectors: reliable in harsh environments
- Four-corner mounting holes: resistance to shock and vibration
- Fully PC-compatible: reduced development costs and time to market
Beagleboard

- Low-power open-source hardware single-board computer produced by Texas Instruments in association with Digi-Key.
- Designed with open source software development in mind, and as a way of demonstrating the Texas Instrument's OMAP3530 system-on-a-chip.
Pandaboard

- Low-power, low-cost single-board computer development platform based on the Texas Instruments OMAP4430 system on a chip (SoC).
Credit-card-sized single-board computer developed in the UK by the Raspberry Pi Foundation

- It has a Broadcom BCM2835 system on a chip (SoC), which includes an ARM1176JZF-S 700 MHz processor (the user can attempt overclocking, up-to 1 GHz), VideoCore IV GPU, shipped with 512 megabytes of RAM, priced at US$ 35.
- It uses an SD card for booting and long-term storage.
- The Foundation provides:
  - Debian and Arch Linux ARM distributions,
  - tools for supporting Python as the main programming language, and
  - BBC BASIC, via the RISC OS image or the "Brandy Basic" clone for Linux, C, and Perl.
Raspberry Pi (2)

http://www.youtube.com/watch?v=zd3hn9q1vhw
An attribute often sacrificed in designing embedded systems is the ability to expand hardware capabilities over time.

Given that the platform has a specific purpose, the designer has the ability to dimension the platform for the specific usage.

The DRAM and nonvolatile memory are usually soldered down on the platform. There are generally no expansion slots to add additional hardware.

Software capabilities are, however, often added over time.

As a general guideline, use no more than 70% of the installed DRAM/flash at the time the product is released if you plan on adding software features over time.
Many embedded systems must remain running for significant amounts of time without any intervention (often years).

It is important to validate and test your system by running your system for several days and reviewing such resources.

It’s not a good idea to assume your system will be restarted regularly.

In many cases you must provide a handler to catch fatal errors (such as no more memory) and trigger a system restart; this is a failsafe option but does result in a loss of system availability during restart.

Additionally, it’s good to partition the system so that subsystems can be restarted without having to restart the entire system.

There are also features such as error-correcting-code (ECC) memory, which automatically corrects single bit errors and detects multiple bit errors in memory.
There is tremendous variability in the user interface requirements of embedded systems.

There are two general classes of embedded devices:

- headless (those without a display)
  - the system must still be managed or controlled by a user, and the device usually provides a command console or simple web interface to the device.
- headed (those providing a display)
  - When users directly interact with the display-based embedded device there are increasingly higher expectations placed on the user experience, in term of ease of use, high quality graphics, and touch screen controls.
A dramatic increase in the level of connectivity for all embedded devices is expected. From a physical connectivity view, many wired and wireless interfaces are used.

- **Ethernet** is the ubiquitous wired interface available on many platforms. For wireless interfaces, 802.11 and Wi-Fi are the most prevalent.
- Other wireless technologies such as **Bluetooth** and those based on IEEE 802.15.4 such as **Zigbee** are provided, depending on the application.
- Wide area wireless technologies such as those based on **3G/4G** cellular technologies are also an important growing area of connectivity for remote managing of devices for which mobility is important.
Security

- The security of embedded systems is becoming an increasingly critical aspect.
- In many embedded cases today, there is no attention paid to security aspects, which leaves systems vulnerable to compromise and attack.
- No matter how secure you believe your platform is, you should assume that you have released a platform with vulnerabilities.
Summary

- We have provided insight into the overall landscape and characteristics of embedded systems.
- We believe that, overall, embedded devices are becoming more sophisticated, and connectivity (local and Internet) is an increasingly important attribute of the system.
- Where specific examples are needed we will use the Raspberry Pi platform to illustrate our point.
- However, in most cases the points being made are applicable to a more general embedded context.