Introduction to Embedded Systems

CSE0420 – Embedded Systems

By Z. Cihan TAYŞİ

Outline

- Definition
- Why embedded systems are different ?
- Embedded design life cycle

Definition

- An embedded system is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints.

- They are everywhere!!!
  - House appliances
  - Aircrafts
  - Vehicles
  - so on…

Why embedded systems are different ?!

- **Dedicated to specific tasks,**
  - It is programmed to perform only one, or perhaps, a few, specific tasks.
  - Changing the task is usually associated with obsolescing the entire system and redesigning it.
  - whereas PCs are generic computing platforms.
- **Supported by a wide array of processors and processor architectures.**
  - In 1998, there were 140 different microprocessors available from more than 40 semiconductor vendors.
  - Now it is impossible to count :)
- **Usually cost sensitive.**
  - It is "**usually**" because the cost of the embedded processor in the Mars Rover was probably not on the design team's top 10 list of constraints.
  - However, if you save 10 cents on the cost of the Engine Management Computer System, you'll be a hero at most automobile companies.
  - Cost does matter in most embedded applications.

Why embedded systems are different ?!

- **Usually have real-time constraints**
  - Let's think about our computers!
  - Sometimes they might ignore us !
    - not responding to our actions such as shortcut keys or mouse clicks.
    - Probably they are busy with some other tasks.
  - Suppose that your computer was connected to a radar antenna in the nose of a commercial jetliner!!!
  - If the computer's main function in life is to provide a collision alert warning, then suspending that task could be disastrous.
- Real-time constraints generally are grouped into two categories: ***time-sensitive constraints*** and ***time-critical constraints***.
- If a task is time critical, it must take place within a set window of time, or the function controlled by that task fails.
  - Controlling the flight-worthiness of an aircraft is a good example of this. If the feedback loop isn't fast enough, the control algorithm becomes unstable, and the aircraft won't stay in the air.
- A time-sensitive task can die gracefully. If the task should take, for example, 4.5ms but takes, on average, 6.3ms.
  - perhaps the inkjet printer will print two pages per minute instead of the design goal of three pages per minute

Why embedded systems are different ?!

- **If an embedded system is using an operating system at all, it is most likely using an RTOS**
  - Like embedded processors, embedded operating systems also come in a wide variety of flavors and colors.
  - RTOSs are not democratic. They need not give every task that is ready to execute the time it needs.
  - RTOSs give the highest priority task that needs to run all the time it needs. If other tasks fail to get sufficient CPU time, it's the programmer's problem.
  - You won't get the dreaded **Blue Screen of Death** that many Windows users see on a
    regular basis

Why embedded systems are different ?!

- **The implications of software failure are much more severe in embedded systems than in desktop systems**
  - Remember the Y2K hysteria !?
  - We all know of the tragic consequences of a medical radiation machine that miscalculates a dosage.
  - Software failure is far less tolerable in an embedded system than in your average desktop PC.
    - That is not to imply that software never fails in an embedded system,
    - just that most embedded systems typically contain some mechanism, such as a ***watchdog timer***, to bring it back to life if the software loses control.

Why embedded systems are different ?!

- **Embedded systems have power constraints**
  - A desktop PC needs a massive heat sink and fan assembly to keep the processor from baking itself to death.
    - This is not a particularly serious constraint for a desktop system
  - However, consider an embedded system operating on battery.
  - Power constraints impact every aspect of the system design decisions.
  - Power constraints affect the processor choice, its speed, and its memory architecture.

Why embedded systems are different ?!

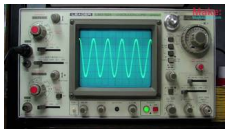
- **Embedded systems must operate under extreme environmental conditions**
  - Embedded systems are everywhere. Everywhere means everywhere.
    - Embedded systems must run in aircraft, in the polar ice, in outer space,
  - Harsh environments usually mean more than temperature and humidity. Devices that are qualified for military use must meet a long list of environmental requirements and have the documentation to prove it.
    - If you've wondered why a simple processor, such as the 8086 from Intel, should cost several thousands of dollars in a missile, think paperwork and environment.
  - The environmental concerns often overlap other concerns, such as power requirements.
    - Sealing a processor under a silicone rubber conformal coating because it must be environmentally sealed also means that the capability to dissipate heat is severely reduced, so processor type and speed is also a factor .

Why embedded systems are different ?!

- **Embedded systems have far fewer system resources than desktop systems**
  - Just make a list of resources that your laptop has!
  - Now consider the resources taht Satellite receiver has!!!
  - Obviously, it has far fewer resources that it must manage than the laptop.
    - Of course, this is because it is **dedicated to a few well-defined tasks** and nothing else
- **Embedded systems store all their object code in ROM**
  - Even your PC has to store some of its code in ROM.
  - ROM is needed in almost all systems to provide enough code for the system to initialize itself (boot-up code).
  - However, most embedded systems must have all their code in ROM.
    - This means severe limitations might be imposed on the size of the code image that will fit in the ROM space

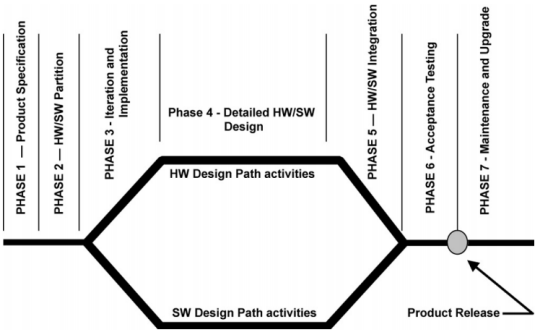
Why embedded systems are different ?!

- **Embedded systems require specialized tools and methods to be efficiently designed**
  - You will also need special tools for debugging in your projects
  - Multimeter, oscilloscope, logic analyzer, etc.



- **Embedded microprocessors often have dedicated debugging circuitry**
  - Mandatory inclusion of dedicated debugging circuitry in silicon on the chip.
  - This is almost counter-intuitive to all of the previous discussion on the cost sensitivity of embedded systems.
  - It seems almost foolish to think that every microprocessor in production contains circuitry that is only necessary for debugging a product under development

Embedded design life cycle

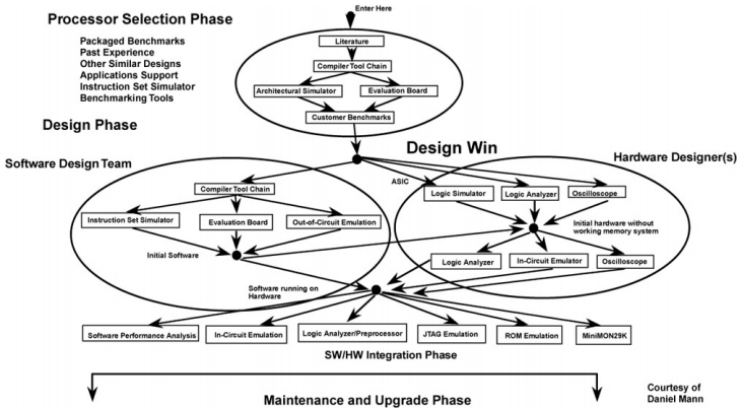


- **Time flows from the left and proceeds through seven phases:**
  - Product specification
  - Partitioning of the design into its software and hardware components
  - Iteration and refinement of the partitioning
  - Independent hardware and software design tasks
  - Integration of the hardware and software components
  - Product testing and release
  - On-going maintenance and upgrading

Embedded design life cycle

- The embedded design process is not as simple as previous figure depicts.
- A considerable amount of iteration and optimization occurs within phases and between phases.
- Defects found in later stages often cause you to "go back to square 1."
  - For example, when product testing reveals performance deficiencies that render the design non-competitive,
    - you might have to rewrite algorithms,
    - redesign custom hardware, such as Application-Specific Integrated Circuits (ASICs),
    - speed up the processor, choose a new processor, and so on.

Embedded design life cycle



- Dr. Daniel Mann, Advanced Micro Devices (AMD), Inc., has developed a tool-based view of the development cycle.
- In Mann's model, processor selection is one of the first tasks.
- it can be argued that including the choice of the microprocessor and some of the other key elements of a design in the specification phase is the correct approach.
  - What if your existing code base is written for the 80X86 processor family,
  - What if your design team is highly experienced using the Green Hills© compiler,

Product Specification

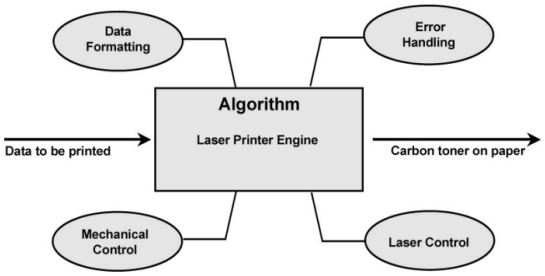
- **The ideal research team is three or four people, usually a marketing or sales engineer and two or three R&D types.**
  - What did each member hear?
  - What was explicitly stated? What was implicit?
  - Did they like what we had or were they being polite?
  - Was someone really turned on by it?
  - Did we need to refine our presentation or the form of the questionnaire?
  - Were we talking to the right people?
- **Customer wants everything yesterday and is unwilling to pay for any of it.**
  - If you ask a customer whether he wants a feature, he'll say yes every time.
- **So, how do you avoid building an aircraft carrier when the customer really needs a fishing boat?**
  - First of all, don't ask the customer whether the product should have a flight deck.
  - Focus your efforts on understanding what the customer wants to accomplish and then extend his requirements to your product.

Hardware / Software Partitioning

- **Embedded design will involve both hardware and software components,**
  - someone must decide which portion of the problem will be solved in hardware and which in software.
  - This choice is called the "partitioning decision."
- **For example, in the early days of the PC the 8086, 80286, and 80386 CPUs didn't have an on-chip floating-point processing unit. (FPU)**
  - These processors required companion devices, the 8087, 80287, and 80387 (FPUs),
  - If the PC did not have an FPU, the application code had to trap the floating-point instructions and execute an exception or trap routine to emulate the behavior of the hardware FPU in software.

Hardware / Software Partitioning

- **Laser printer design**
  - The processor places the incoming data stream via the parallel port, RS-232C serial port, USB port, or Ethernet port into a memory buffer.
- **Concurrently, the processor services**
  - the data port and converts the incoming data stream into a stream of modulation and control signals to a laser tube, rotating mirror, rotating drum, and assorted paper-management "stuff."



Hardware / Software Partitioning

- **This would bog down most modern microprocessors and limit the performance of the system**
  - You could try to improve performance by adding more processors,
  - When you analyze the algorithm, you see that certain tasks critical to the performance of the system are also bounded and well-defined.
    - These tasks can be easily represented by design methods that can be translated to a hardware-based solution; **an ASIC**
  - Fine-tune the software so that the hardware-assisted circuit devices are not necessary

Iteration and Implementation

- **Before Hardware and Software Teams Stop Communicating**
- **The design is still very fluid in this phase.**
  - major blocks might be partitioned between the hardware components and the software components,
  - plenty of leeway remains to move these boundaries
- **The hardware designers**
  - Simulation tools to model the performance of the processor and memory systems.
- **The software designers**
  - running code benchmarks on self-contained, single-board computers **(evaluation boards)**

Detailed Hardware and Software Design

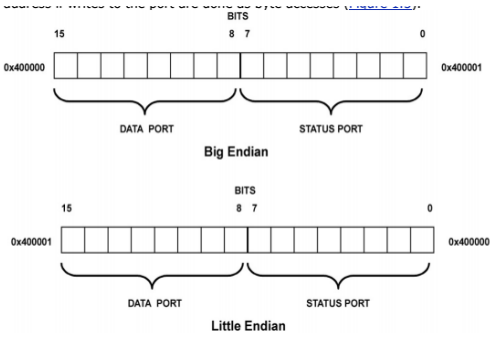
- **Software development in embedded systems requires specialized programming techniques.**
  - Bitwise operations
  - Storage class modifier volatile
  - Intterrupts and interrupt service routines (ISR)
  - Control loop
- This course is **NOT** intended to teach you how to design hardware!

Hardware/Software Integration

- The hardware/software integration phase of the development cycle must have special tools and methods to manage the complexity.
- The process of integrating embedded software and hardware is an exercise in debugging and discovery.
  - Discovery is an especially apt term because the software team now finds out whether it really understood the hardware specification document provided by the hardware team.

Hardware/Software Integration

- **Suppose, for example that a serial port is designed for an ASIC with a 16-bit I/O bus.**
  - The port is memory mapped at address 0x400000.
  - Eight bits of the word are the **data portion** of the port,
  - and the other eight bits are the **status portion** of the port
- **If byte addressing is used and the big endian model is assumed,**
  - then the algorithm should check the status at address 0x400001.
  - Data should be read from and written to address 0x400000.
- If the **little endian** memory model is assumed, then the reverse is true.



Product Testing and Release

- Testing and reliability requirements for an embedded system are much more stringent than the vast majority of desktop applications.
  - Consider the embedded systems currently supporting your desktop PC: IDE disk drive, CD-ROM, scanner, printer, and other devices are all embedded systems in their own right.
  - How many times have they failed to function so that you had to cycle power to them?
- Many desktop applications have small memory leaks. Presumably, if the application ran long enough, the PC would run out of heap space, and the computer would crash

Maintaniance and Upgrading

- The majority of embedded system designers maintain and upgrade existing products, rather than design new products.
  - Most of these engineers were not members of the original design team for a particular product,
  - They must rely on only their experience, their skills, the existing documentation, and the old product to understand the original design well enough to maintain and improve it
- This phase of a product's life cycle requires tools that are especially tailored to reverse engineering and rapidly facilitating "what if …" scenarios.
  - For example, it's tempting to try a quick fix by speeding up the processor clock by 25 percent;
  - however, this could cause a major ripple effect through the entire design, from memory chip access time margins to increased RF emissions