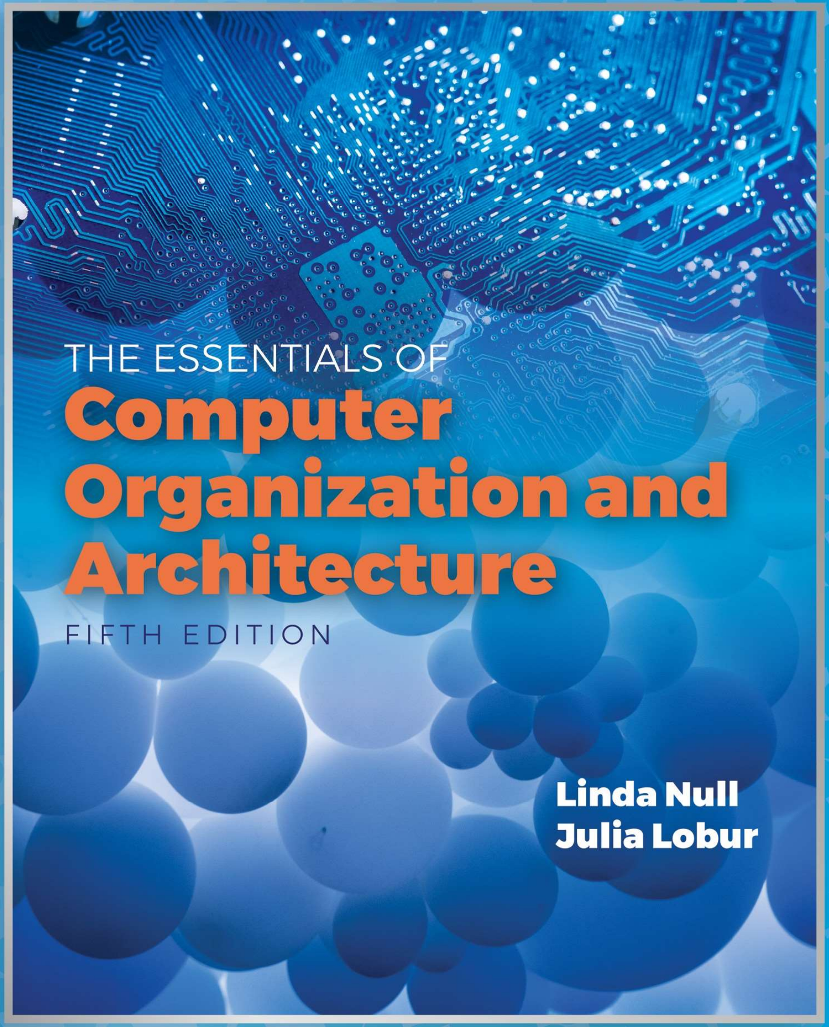
Chapter 10

Topics in Embedded Systems



Objectives

- Understand the ways in which embedded systems differ from general purpose systems.
- Be able to describe the processes and practices of embedded hardware design.
- Understand key concepts and tools for embedded software development.

10.1 Introduction (1 of 2)

- Embedded systems are real computer systems that support the operation of a device (or machine) that usually is not a computer.
- The user of the embedded system is rarely aware of its existence within the device.
- These systems are all around us. They are in watches, automobiles, coffeepots, TVs, telephones, aircraft, and just about any “intelligent” device that reacts to people or its environment.

10.1 Introduction (2 of 2)

- Embedded systems are different from general-purpose systems in several important ways. Some key differences are:
  - Embedded systems are resource constrained. Utilization of memory and power are critical. The economy of hardware and software is often paramount, and can affect design decisions.
  - Partitioning of hardware and software is fluid.
  - Embedded systems programmers must understand every detail about the hardware.
  - Signal timing and event handling are crucial.

10.2 An Overview Embedded Hardware (1 of 22)

- We will classify embedded hardware according to the extent to which it is adapted or adaptable by the people who program and install the system into the device that it supports.
- Accordingly, we say that embedded hardware falls into categories of:
  - Off-the-shelf
  - Configurable
  - Fully-customized

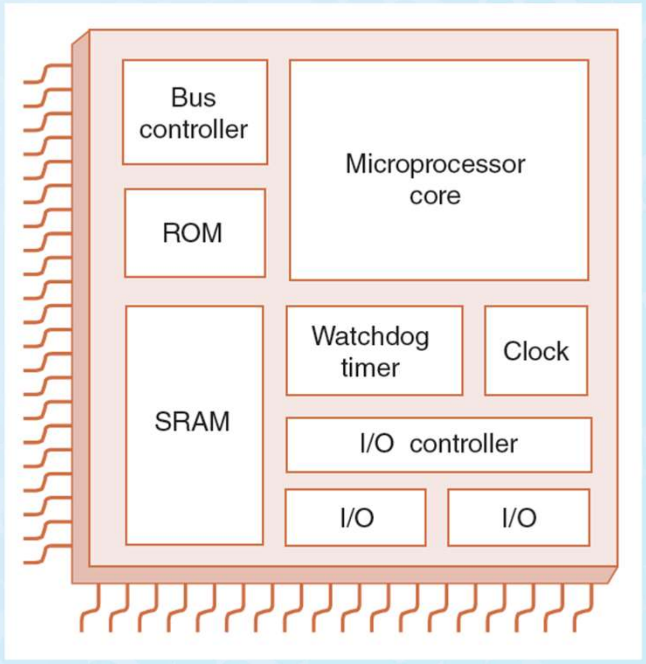
Note: There are many other taxonomies. This one is convenient for our purposes.

10.2 An Overview Embedded Hardware (2 of 22)

- Using off-the-shelf hardware, minimal hardware customization possible.
  - Perhaps add memory or peripherals. The internal wiring stays the same.
- The most common off-the-shelf hardware is the microcontroller.
  - Microcontrollers are often derivatives of "old" PC technology. They are inexpensive because development costs were recouped long ago.
  - There are thousands of different microcontrollers.

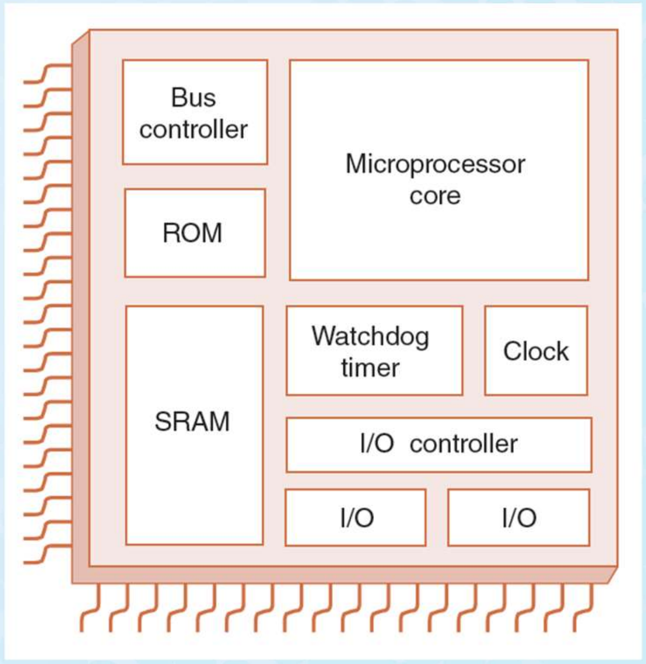
10.2 An Overview Embedded Hardware (3 of 22)

- Example: Microcontrollers are Motorola's 68HC12, Intel’s 8051, Microchip's 16F84A, and the PIC family.
- A simplified block diagram of a microcontroller is shown at the right.



10.2 An Overview Embedded Hardware (4 of 22)

- We have seen all of these components before except for the watchdog timer.
- A watchdog timer helps guard against system hangs by continually checking for liveness.
- Watchdog timers are not used in all microcontrollers.



10.2 An Overview Embedded Hardware (5 of 22)

- For some applications, microcontrollers are too limited in their functionality.
- Systems-on-a-chip (SOCs) are full blown computer systems—including all supporting circuits—that are etched on a single die.
  - Alternatively, separate chips are needed to provide the same services.
  - The additional chips are costly and consume power and space.

10.2 An Overview Embedded Hardware (6 of 22)

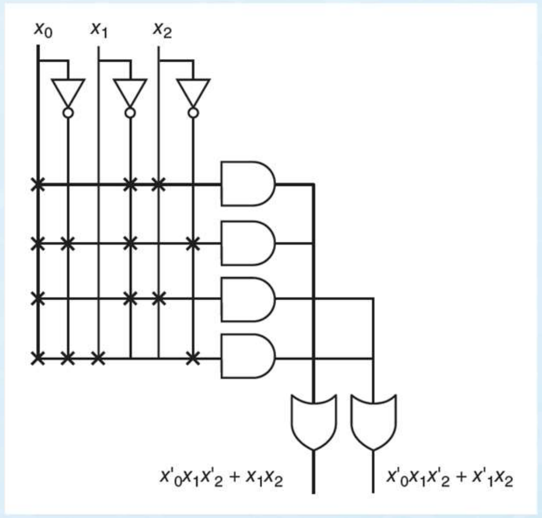
- Semi-custom systems-on-a-chip can be fabricated whenever a suitable off-the-shelf SOC is unavailable.
- The chip mask is created using blocks of pre-designed, pretested intellectual property (IP) circuits.
- The semi-custom approach is costly. To save money, off-the-shelf SOCs are preferred, even when their functionality is not an exact fit for the application.

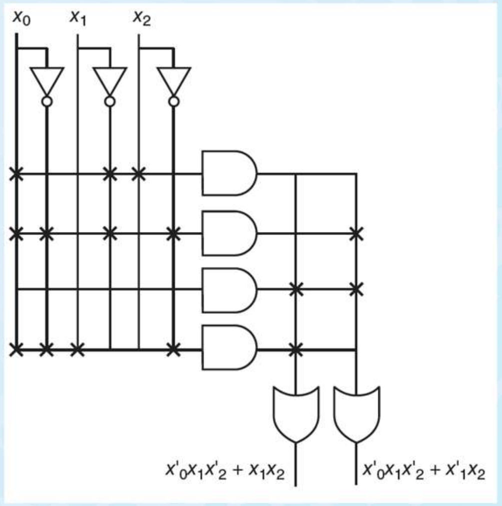
10.2 An Overview Embedded Hardware (7 of 22)

- Programmable logic devices (PLDs) are configurable devices in which the behavior of the circuits can be changed to suit the needs of an application.
  - Programmable array logic (PAL) chips consist of programmable AND gates connected to a set of fixed OR gates.
  - Programmable logic array (PLA) chips consist of programmable AND gates connected through programmable OR gates.

10.2 An Overview Embedded Hardware (8 of 22)

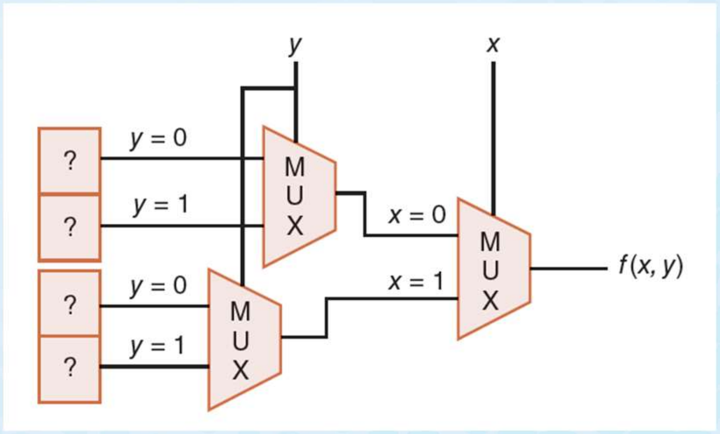
- A programmed PAL and a programmed PLA:





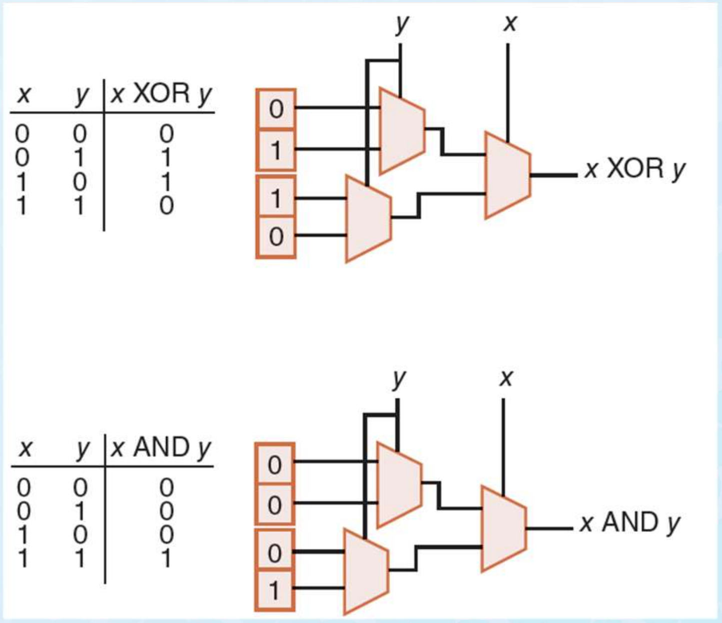
10.2 An Overview Embedded Hardware (9 of 22)

- The behavior of field programmable gate arrays (FPGAs) is controlled through values stored in memory lookup tables rather than by changing connections between logic elements.



10.2 An Overview Embedded Hardware (10 of 22)

- Truth tables are entered directly into FPGA memory.

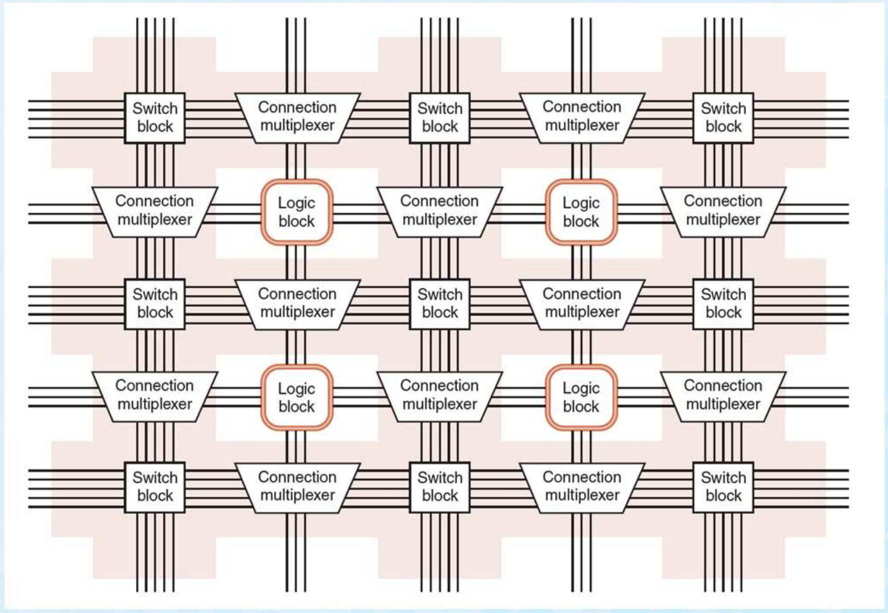


x	*y*	*x* XOR *y*
0	0	0
0	1	1
1	0	1
1	1	0

x	*y*	*x* AND *y*
0	0	0
0	1	0
1	0	0
1	1	1

10.2 An Overview Embedded Hardware (11 of 22)

- FPGAs typically consist of blocks of logic elements interconnected by switches and multiplexers in an “island” configuration.



10.2 An Overview Embedded Hardware (12 of 22)

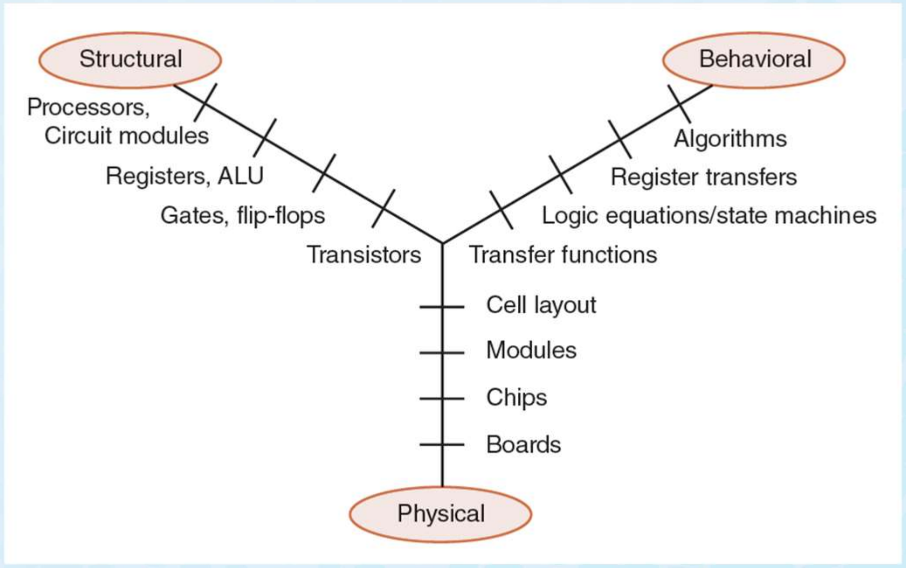
- When:
  - Off-the-shelf microcontrollers and SOCs do not have sufficient functionality for the task at hand...
  - Or off-the-shelf microcontrollers and SOCs have too much functionality, with the excess consuming resources needlessly…
  - And a semi-custom chip cannot be economically fabricated from commercially available IP designs...
  - And PLDs are too expensive or too slow…
- The only option left is to design an application-specific integrated circuit (ASIC) from scratch.

10.2 An Overview Embedded Hardware (13 of 22)

- To design a chip from scratch we need to think about it from three points of view:
- What do we need the chip to do?
- Which logic components can provide the behavior we need?
- What is the best way to position the components on the silicon die in order to reduce cost and provide the best performance?

10.2 An Overview Embedded Hardware (14 of 22)

- Gajski's Logic Synthesis Y-Chart depicts the relationship of these three dimensions of circuit design.



10.2 An Overview Embedded Hardware (15 of 22)

- Creating circuit designs along all three dimensions is an enormously complex task that is nearly impossible to do—with any amount of accuracy or effectiveness—without a good toolset.
- Hardware definition languages (HDLs) were invented in the latter part of the twentieth century. HDLs help designers manage circuit complexity by expressing circuit logic in algorithmic terms.

10.2 An Overview Embedded Hardware (16 of 22)

- Two of the most popular HDLs are Verilog and VHDL.
- Verilog is a C-like language invented in 1983. It is now IEEE 1364-2001.
- VHDL is an ADA-like HDL released in 1985. It is now IEEE 1097-2002.
- The output from the compilation of both of these languages is a netlist, which is suitable for use as input to electronic design automation machines that produce integrated circuit masks.

10.2 An Overview Embedded Hardware (17 of 22)

- Traditional HDLs manipulate circuit definitions in terms of RTL and discrete signal patterns.
- Using these languages, engineers are strained to keep up with the complexity of today's SOCs.
- To make design activities more accurate and cost efficient, the level of abstraction must be raised above the RTL level.
- SystemC and SpecC are two recent HDLs that were invented to help solve this problem.

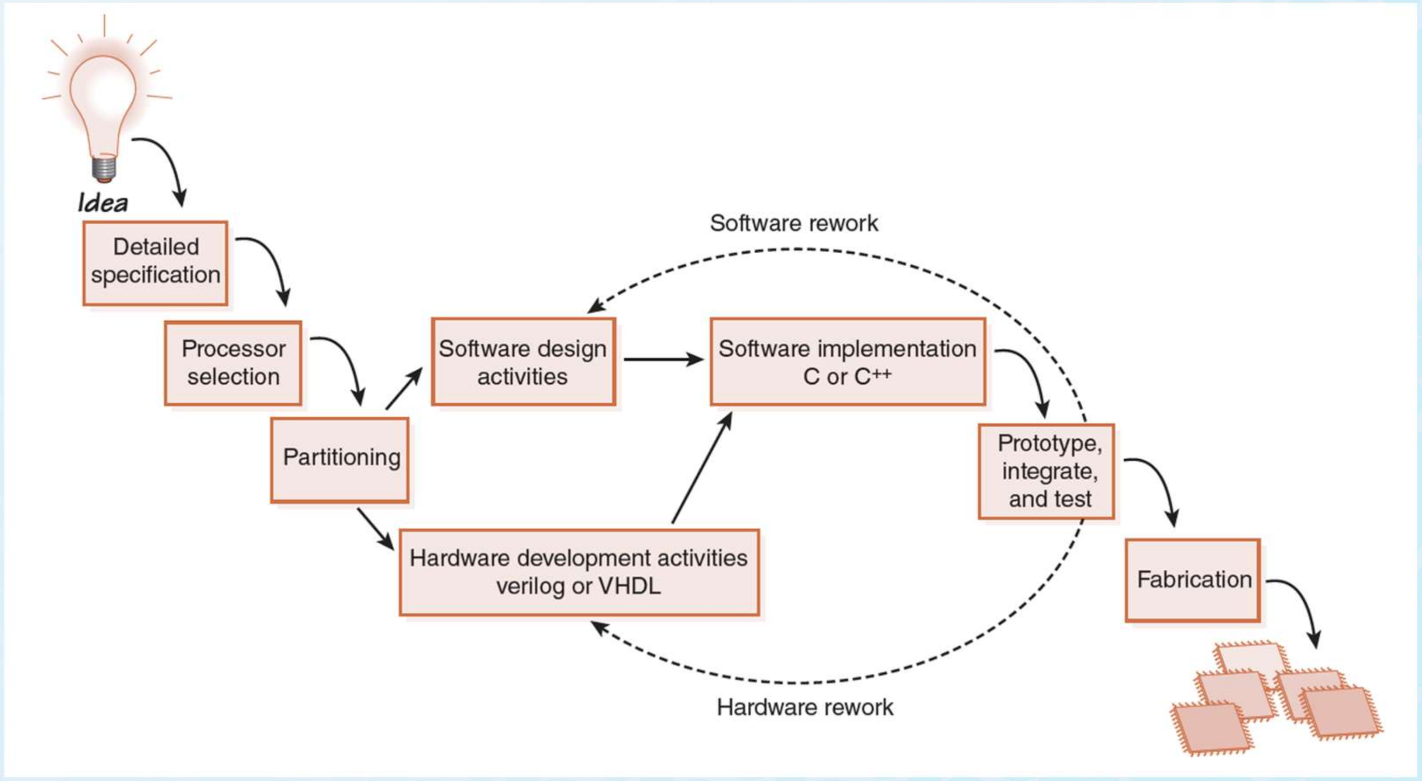
10.2 An Overview Embedded Hardware (18 of 22)

- SystemC is an extension of C++ that includes classes and libraries specifically created for embedded systems design, to include modeling events, timing specifications, and concurrency.
- SpecC is a C-like language, created from the outset as a system design language.
- A SpecC development package includes a methodology that guides engineers through four phases of system development:
  - Specification, architecture, communication channels, and implementation.

10.2 An Overview Embedded Hardware (19 of 22)

- Embedded systems have been traditionally developed by specialized teams that collaboratively:
  - Produce a detailed specification derived from a functional description.
  - Select a suitable processor or decide to build one.
  - Determine the hardware-software partition.
  - Design the circuit and write the program(s) that will run on the system.
  - Prototype and test the system.
- This system design cycle is shown on the next slide.

10.2 An Overview Embedded Hardware (20 of 22)

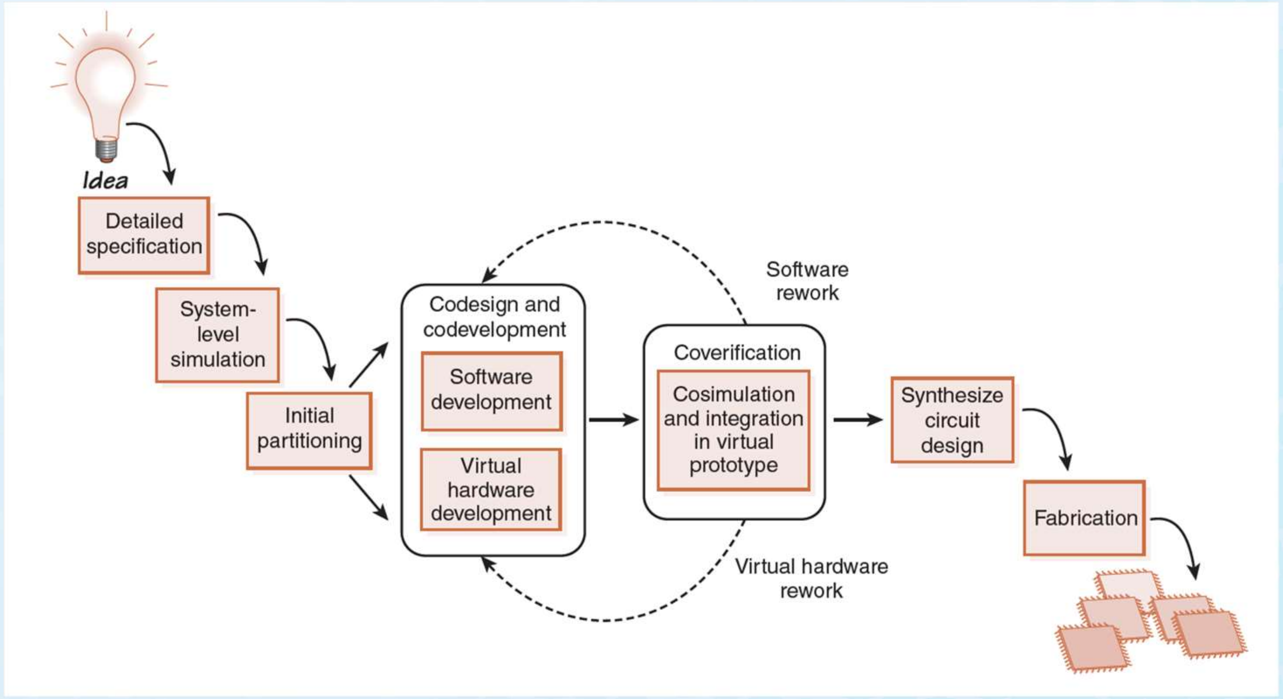

Idea
Detailed specification
Processor selection
Partitioning
Software design activities
Software implementation C or C++
Hardware development activities verilog or VHDL
Prototype, integrate, and test
Fabrication
Software rework
Hardware rework

Notice the back arrows. These steps are costly.

10.2 An Overview Embedded Hardware (21 of 22)

- SystemC and SpecC facilitate changes to the traditional design lifecycle.
  - Hardware developers and software developers can speak the same language.
  - Codevelopment teams work side-by-side simultaneously creating hardware designs and writing programs.
  - Codevelopment shortens the development lifecycle and improves product quality.
- The embedded system codesign lifecycle is shown on the next slide.

10.2 An Overview Embedded Hardware (22 of 22)



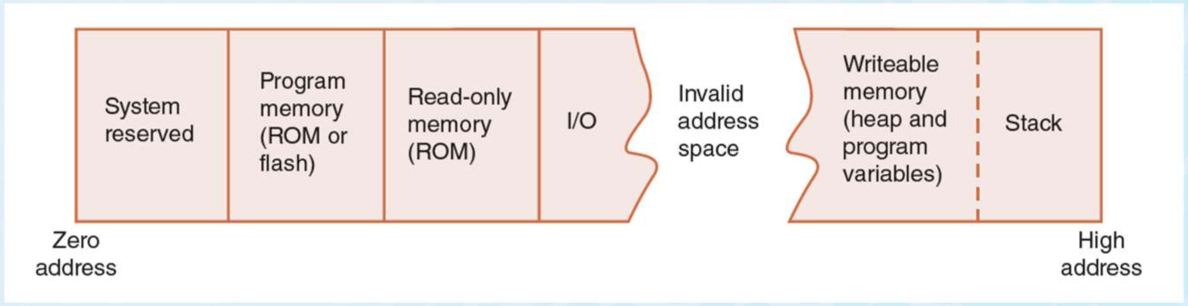
Rework takes place on a virtual system.

10.3 An Overview Embedded Software (1 of 11)

- Software development for embedded systems presents a distinct set of challenges.
- Some of these challenges are related to the uniqueness of the hardware, such as its particular memory organization.
  - Memory limitations are almost always a software development constraint.
  - Virtual memory is not suitable for most embedded applications.

10.3 An Overview Embedded Software (2 of 11)

- Embedded system memory can consist of several different kinds, including RAM, ROM, and flash, all sharing the same address space.



Memory leaks in embedded systems are especially problematic.

10.3 An Overview Embedded Software (3 of 11)

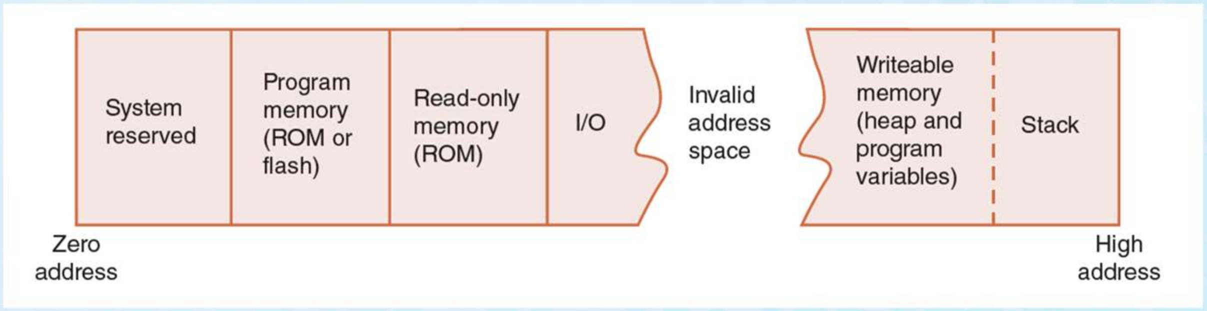
- Embedded operating systems differ from general-purpose operating systems in a number of ways.
  - Responsiveness is one of the major distinguishing features.
- Not all embedded operating systems are real-time operating systems.
  - Timing requirements may differ little from a desktop computer.
  - Hard real-time systems have strict timing constraints.
  - In soft real-time systems, timing is important but not critical.

10.3 An Overview Embedded Software (4 of 11)

- *Interrupt latency* is the elapsed time between the occurrence of an interrupt and the execution of the first instruction of the interrupt service routine (ISR).
  - Interrupt latency is indirectly related to system responsiveness. The smaller the latency, the faster the response.
- Interrupts can happen at any time and in any order.
- The ISR for one interrupt possibly may not be completed before another interrupt occurs.
  - High-quality systems support such *interrupt nesting*.

10.3 An Overview Embedded Software (5 of 11)

- Memory organization in resource-constrained systems differs from traditional systems.
- The entire address space might not be used.
- The stack and the heap typically start at different ends of the address space.



10.3 An Overview Embedded Software (6 of 11)

- Memory footprint is a critical concern with embedded operating systems.
  - If an operating system takes up too much memory, additional memory may be required.
  - Memory consumes power.
  - Thus, the smaller the operating system, the better.
- Most embedded operating systems are modular, allowing only the most necessary features to be installed.

10.3 An Overview Embedded Software (7 of 11)

- IEEE 1003.1-2001, POSIX, is the specification for standardized Unix, to which Embedded Linux adheres.
- Other popular embedded operating systems include Windows 10 IoT, QNX, and MS-DOS.
  - Windows has several versions, each intended for a particular application area.
- There are hundreds of others, each having its distinctive behavior and target hardware.
  - Licensing costs for the operating system are as great a concern as hardware costs.

10.3 An Overview Embedded Software (8 of 11)

- General-purpose software development is usually iterative and incremental.
  - Code a little, test a little.
- Embedded systems development requires a much more rigorous and linear path.
- Functional requirements must be clear, complete, and accurate when work begins.
- Formal languages, such as Z, are helpful in providing accuracy and correctness.

10.3 An Overview Embedded Software (9 of 11)

- Large software projects are usually partitioned into chunks so that the chunks can be assigned to different teams.
- Embedded software doesn’t partition so easily, making team assignments difficult.
- To improve performance, some embedded programmers advocate the use of global variables and unstructured code.
- Others rail against this idea, saying that it is not good engineering practice regardless of the platform for which the software is written.

10.3 An Overview Embedded Software (10 of 11)

- Event handling is a major challenge to the embedded programmer.
  - It lies at the heart of embedded systems functionality.
- Events can happen asynchronously and in any order.
- It is virtually impossible to test all possible sequences of events.
- Testing must be rigorous and thorough.

10.3 An Overview Embedded Software (11 of 11)

- Embedded programming is essentially a matter of raising and responding to signals.
- Hardware support may be designed into a chip to facilitate the tracing and debugging of signal patterns.
  - Examples are ICE, Motorola's BDM, IEEE 1149.1 JTAG, and IEEE 5001 Nexus.
- Some platforms offer no tool support in the way of debuggers or even compilers.
  - Writing software for these systems is called *bare metal programming*.

Conclusion (1 of 5)

- Embedded systems differ from general-purpose systems because:
  - They are resource constrained.
  - Programming requires deep awareness of the underlying hardware.
  - Signal timing and event handling are critical.
  - The hardware-software partition is moveable.
- Embedded hardware can be off-the-shelf, semi-customized, fully-customized, or configurable.

Conclusion (2 of 5)

- Programmable logic devices include:
  - PALs: Programmable AND gates connected to a set of fixed OR gates.
  - PLA: Programmable AND gates connected through programmable OR gates.
  - FPGA: Logic functions provided through lookup tables.
- PLDs tend to be slow and expensive as compared to off-the-shelf ICs.

Conclusion (3 of 5)

- Hardware definition languages Verilog, VHDL specify the functions and layout of full-custom chips.
- SpecC and SystemC raise the level of abstraction in chip design.
- Hardware-software codesign and cosimulation reduces errors and brings products to market faster.

Conclusion (4 of 5)

- Embedded operating systems differ from general purpose operating systems in their timing and memory footprint requirements.
- IEEE 1003.1-2001, POSIX, is the specification for standardized Unix, to which Embedded Linux adheres.
- Other popular embedded operating systems include Windows 10 IoT, QNX, and MS-DOS.

Conclusion (5 of 5)

- Embedded software requires accurate specifications and rigorous development practices.
  - Formal languages help.
- Event processing requires careful specification and testing.
- Embedded system debugging can be supported by hardware interfaces to include ICE, BDM, JTAG, and Nexus.