THE ESSENTIALS OF

Computer Organization and Architecture

THIRD EDITION

Linda Null
Julia Lobur

Chapter 10

Topics in Embedded Systems

Chapter 10 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

- 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.

As Compared with your Laptop, in which you have a GUI to be used to interact with your system software and Hardware

- 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

- 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. **Importance of H/W and S/W economy**
  - Partitioning of hardware and software is fluid.
  - Embedded systems **programmers** must understand every detail about the hardware.
  - Signal timing and event handling are crucial. **Real time processig and event handling**

10.2 Embedded Hardware Overview

- 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. **How this system is built**
- 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 Embedded Hardware Overview

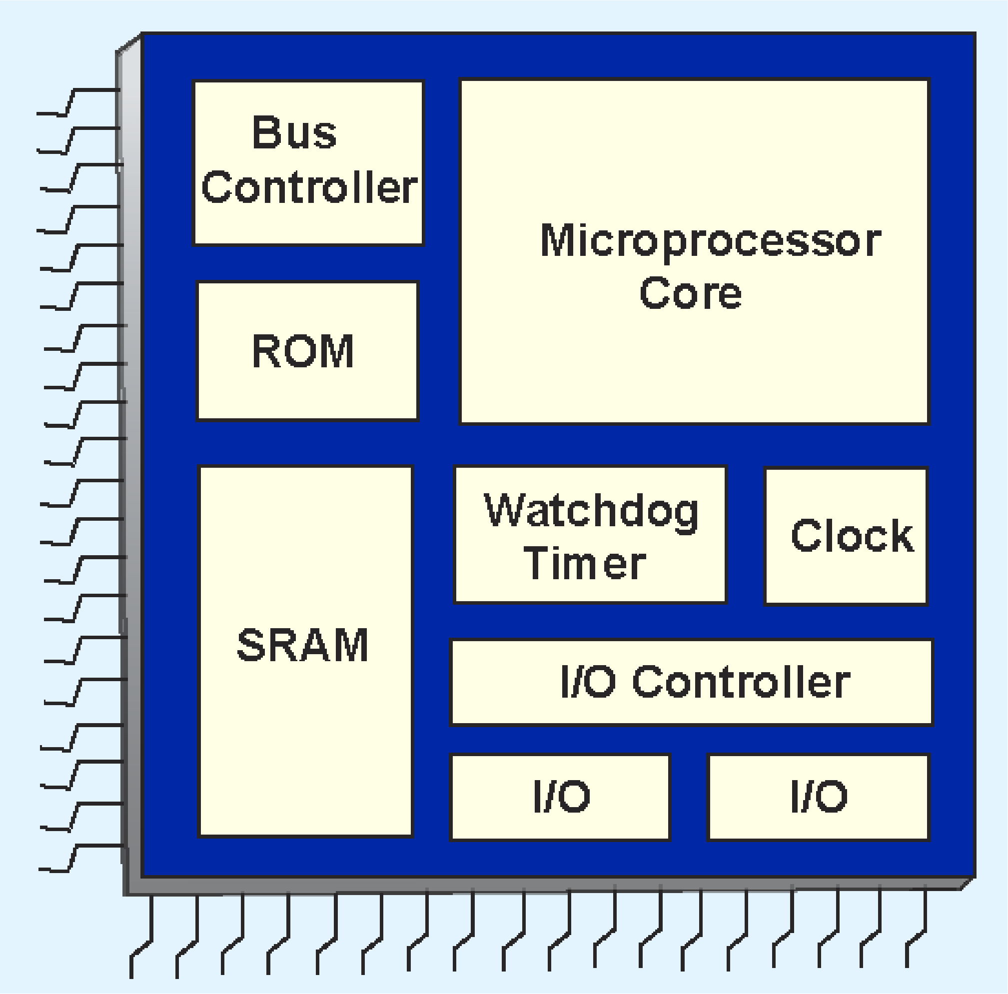
- Using off-the-shelf hardware, minimal hardware customization possible.
  - Perhaps add memory or peripherals. The internal wiring stays the same.
- The most common off-the-shelf hardware is the microcontroller.
  - Microcontrollers are often derivatives of “old” PC technology. They are inexpensive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

Different memory access and different speeds

10.2 Embedded Hardware Overview

– 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.

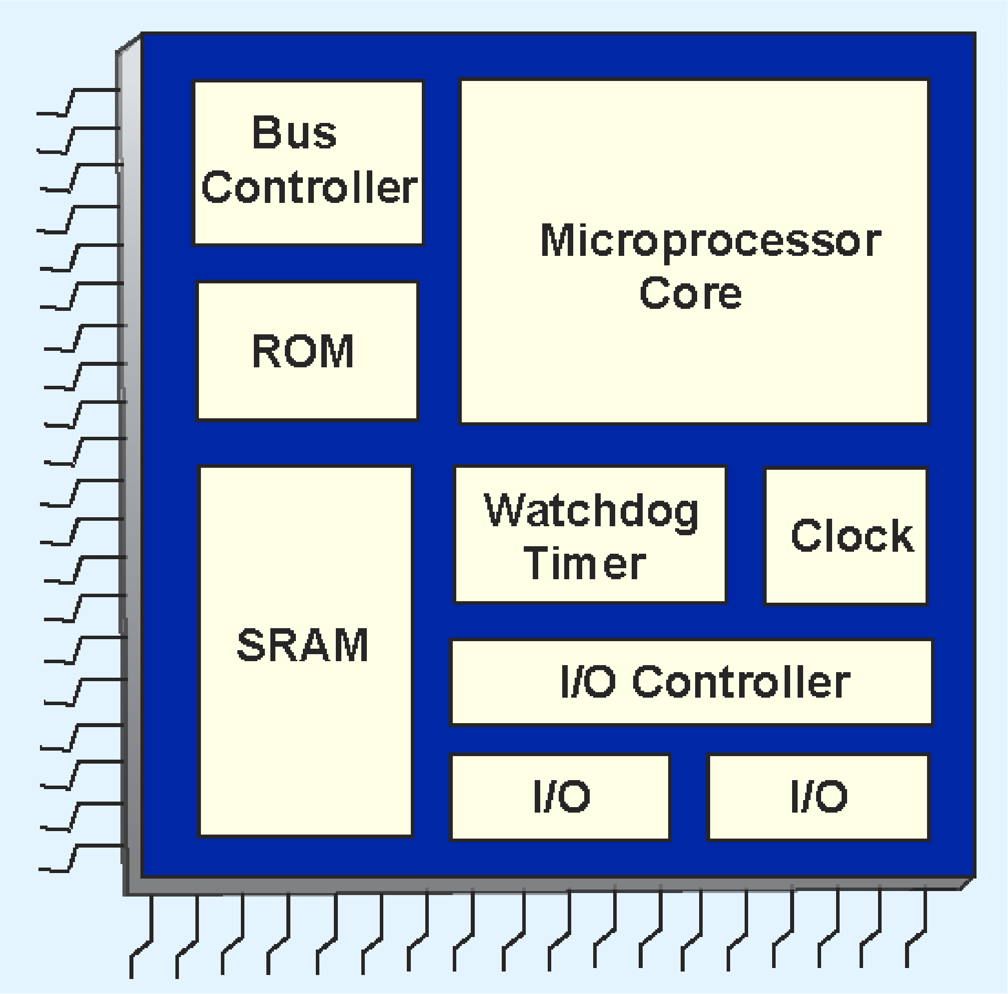


10.2 Embedded Hardware Overview

- 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.

Used for synchronization

- Watchdog timers are not used in all microcontrollers.



10.2 Embedded Hardware Overview

- 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.

  This is adding other modules during the etcheing process on the die

  - **Alternatively**, separate chips are needed to provide the same services.
  - The additional chips are costly and consume power and space.

10.2 Embedded Hardware Overview

- **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, pre-tested 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.

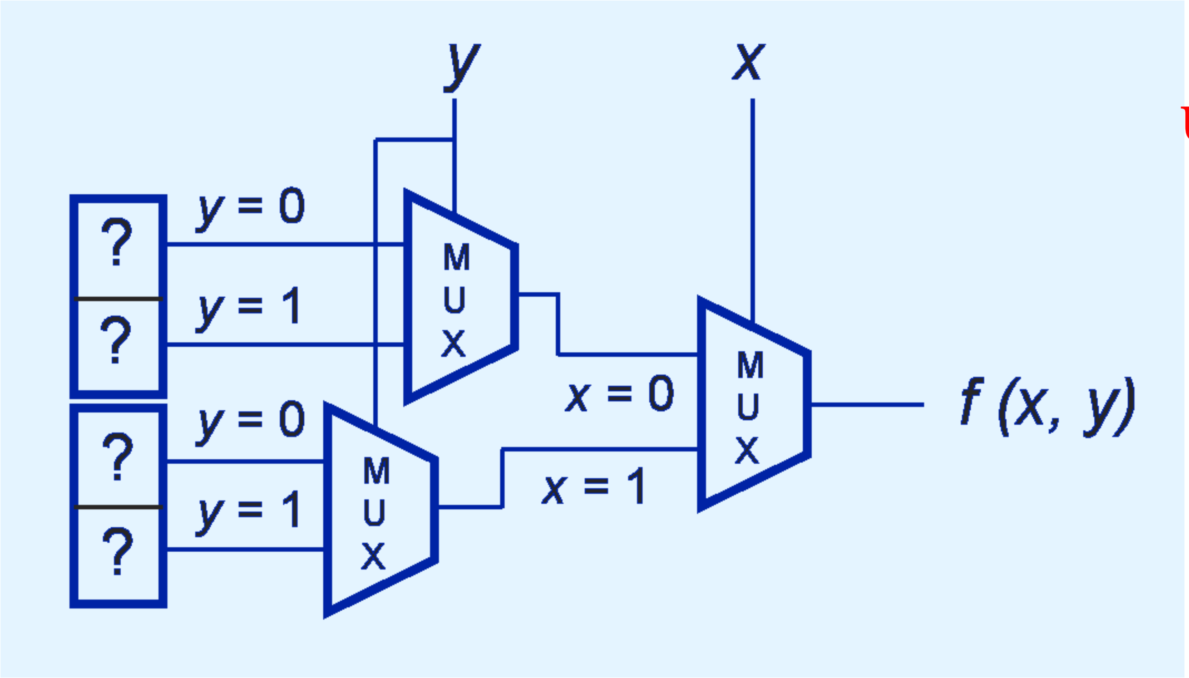
Therefore some of the ready available SOCs, that have extra modules, may be used with those extra modules being used as design required.

10.2 Embedded Hardware Overview

- 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. Contains arrays of programmable AND gates and predefined OR gates
  - Programmable logic array (PLA) chips consist of programmable AND gates connected through programmable OR gates. Used to implement combinational logic circuits

10.2 Embedded Hardware Overview

- 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.



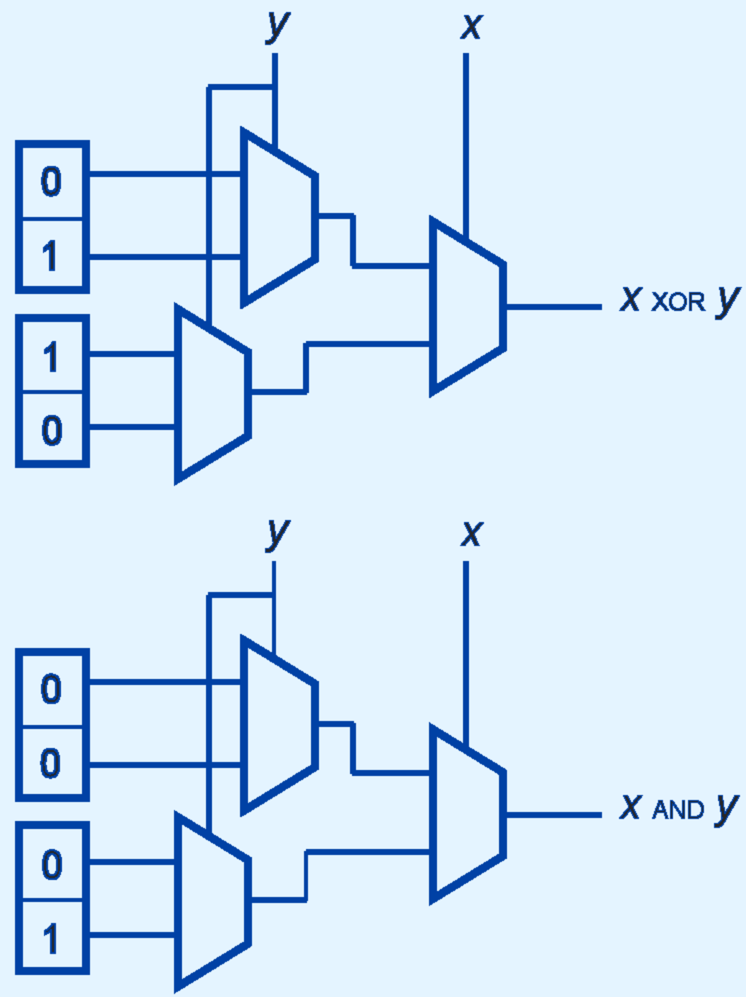
Use of software means to program a hardware chipset

10.2 Embedded Hardware Overview

– Truth tables are entered directly into FPGA memory.

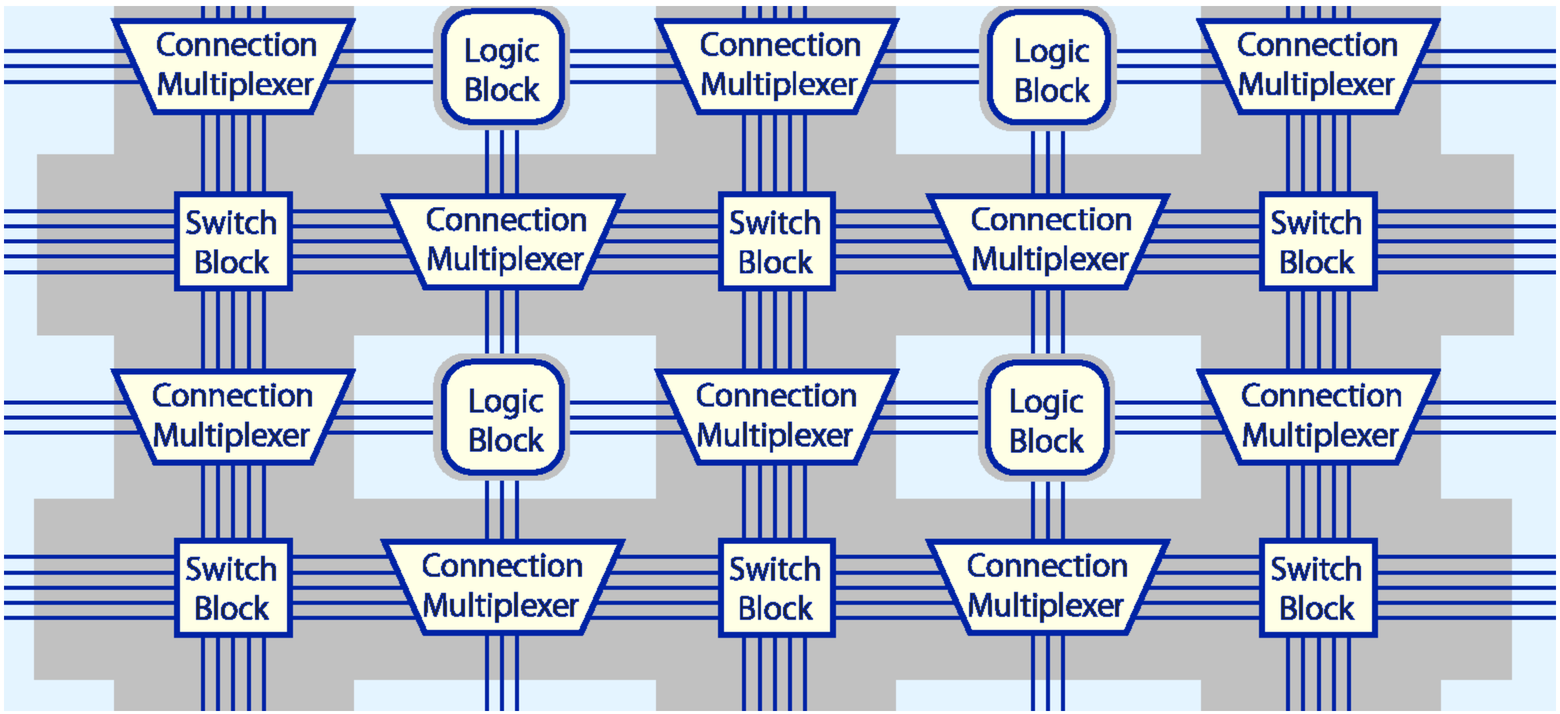
x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

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



10.2 Embedded Hardware Overview

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



10.2 Embedded Hardware Overview

- **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 **a**pplication-**s**pecific **i**ntegrated **c**ircuit (**ASIC**) from scratch.

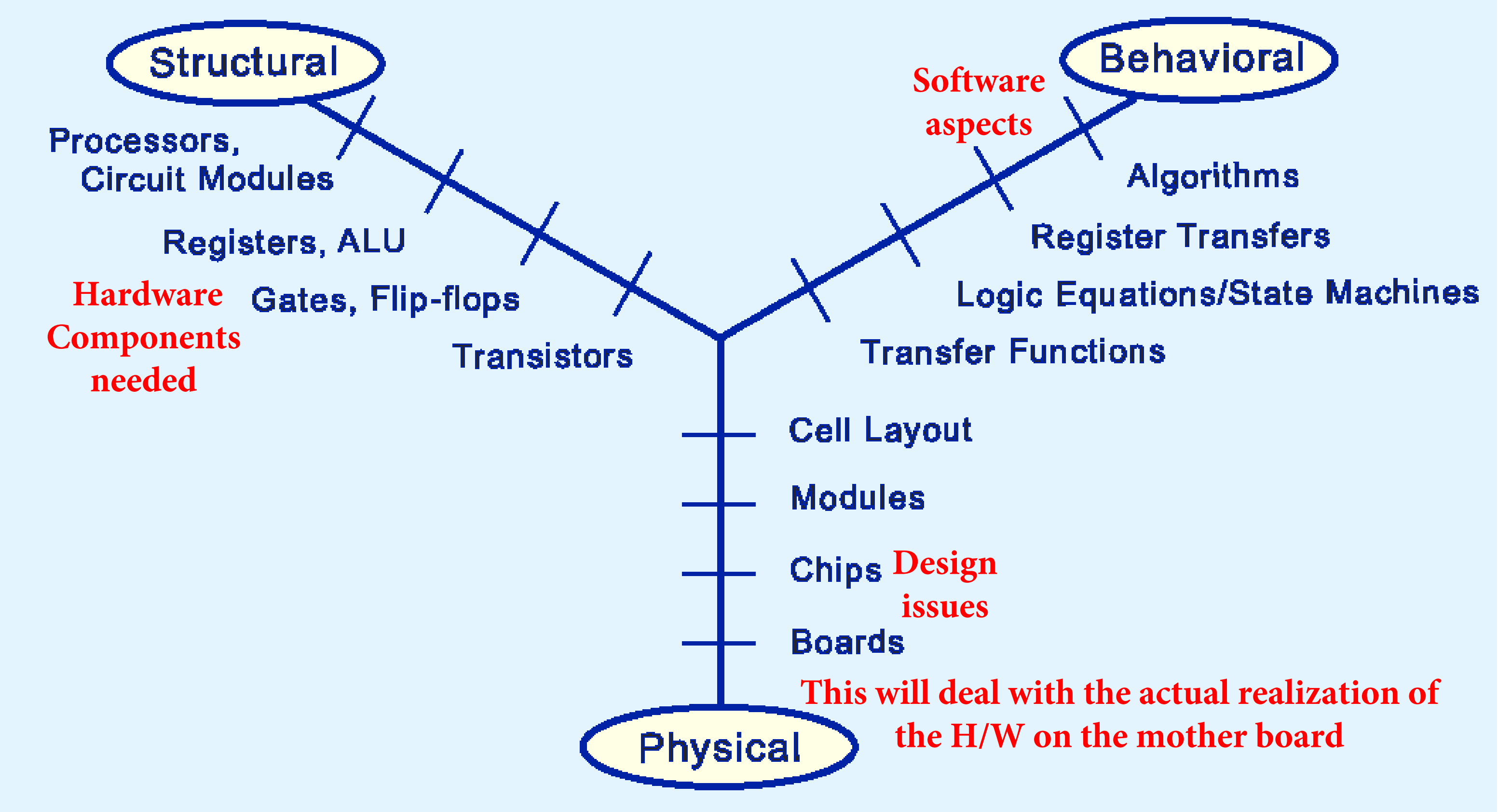
SOCs are System On Chips

10.2 Embedded Hardware Overview

- 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 Embedded Hardware Overview

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



10.2 Embedded Hardware Overview

- 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.

This is the use of S/W implementation to program a H/W to realize a particular task

10.2 Embedded Hardware Overview

- 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. **Compiler Based**
- VHDL is an ADA-like HDL released in 1985. It is now IEEE 1097-2002. **Interpreter Based**
- 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 Embedded Hardware Overview

- Traditional HDLs manipulate circuit definitions in terms of **RTL** and **discrete signal patterns**.

RTL " Rgister Transfer Level"

- Using these languages, engineers are strained to keep up with the complexity of today's SOCs.

Systems On Chips

- To make design activities more accurate and cost efficient, the level of abstraction must be raised above the RTL level.

This will mean that other components needed will be made general rather than specific

- SystemC and SpecC are two recent HDLs that were invented to help solve this problem.

10.2 Embedded Hardware Overview

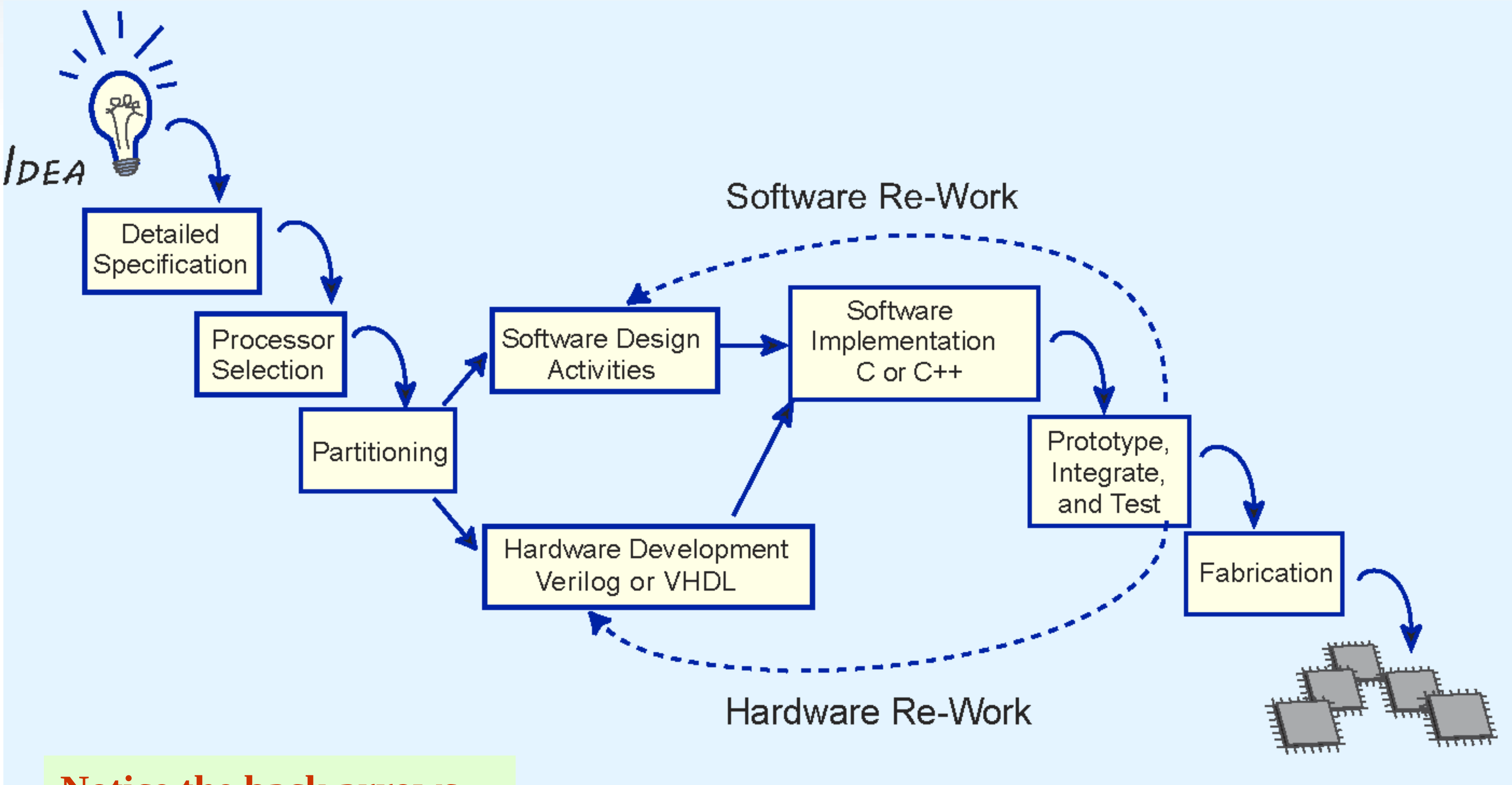
- SystemC is an extension of C++ that includes classes and libraries specifically created for embedded systems design, to include modeling events, timing specifications, and concurrency.
- SpecC is a C-like language, created from the outset as a system design language. This was started from scratch
-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

10.2 Embedded Hardware Overview

- 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 Embedded Hardware Overview



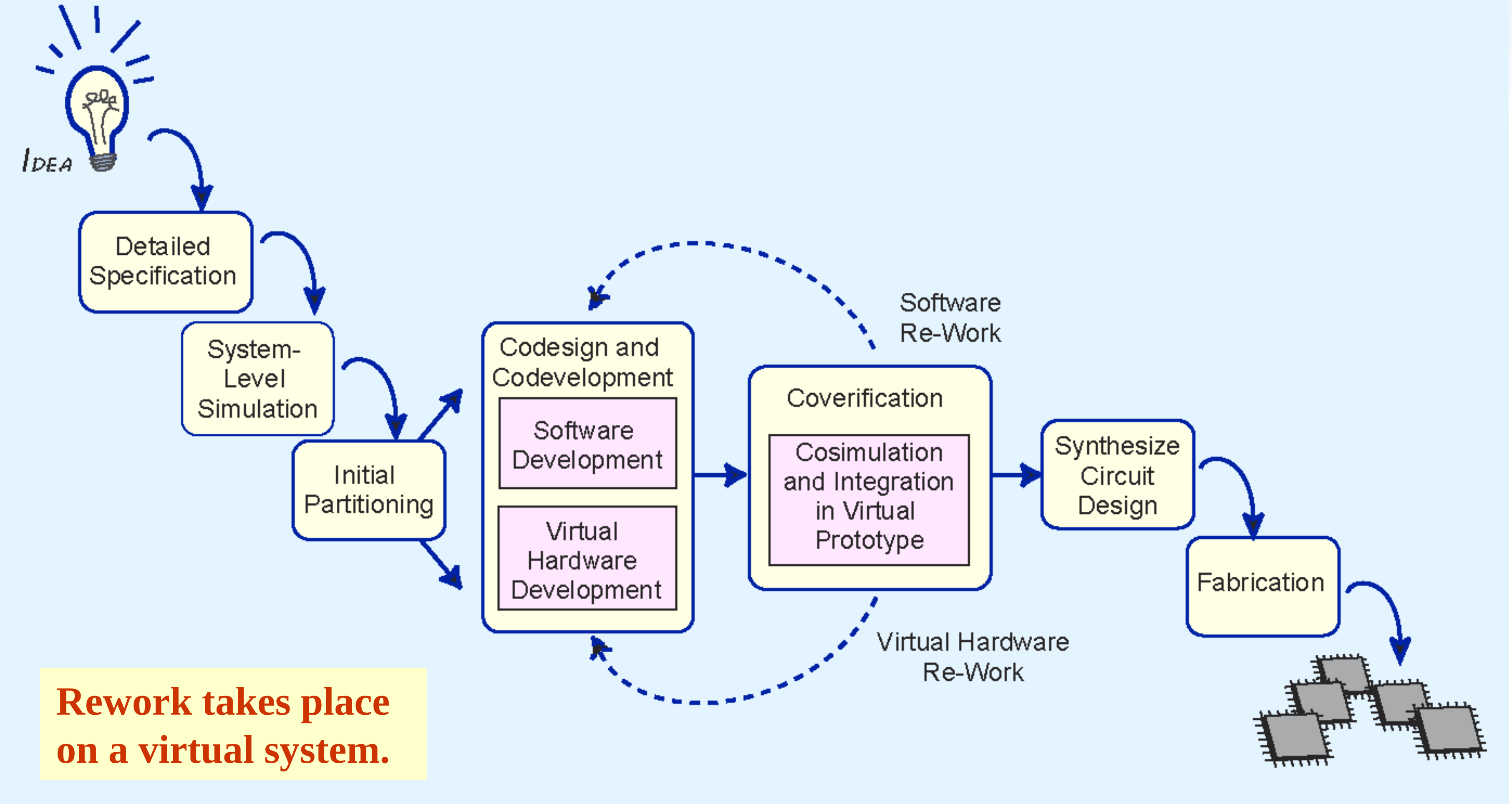
Notice the back arrows. These steps are costly.

10.2 Embedded Hardware Overview

- 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 Embedded Hardware Overview



Rework takes place on a virtual system.

10.3 Embedded Software Overview

- **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. Lack of memory space for memory mapping
  - **Virtual memory** is not suitable for most embedded applications. Lack of external memory devices

10.3 Embedded Software Overview

- 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. i.e. deadline misses are tolerable, but degrades the system's quality of service**

10.3 Embedded Software Overview

- *Interrupt latency* is the elapsed time between the occurrence of an interrupt and the execution of the first instruction of the **i**nterrupt **s**ervice **r**outine (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 Embedded Software Overview

- Memory footprint is another critical concern with embedded 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.