
INTEGRATED TECHNICAL EDUCATION CLUSTER
AT ALAMEERIA



SPRING 2015

# E-626-A
# Real-Time Embedded Systems (RTES)

# Lecture #9
# Hardware Software Codesign

**Instructor:**
**Dr. Ahmad El-Banna**

# Agenda

- Introduction to HS-CODES
- Codesign Motivation
- Some Issues on Codesign of Embedded System

# Introduction

Digital systems designs consists of hardware components and software programs that execute on the hardware platforms

Hardware-Software Codesign ?

# Microelectronics trends

- Better device technology
  - reduced in device sizes
  - more on chip devices > higher density
  - higher performances

- Higher degree of integration
  - increased device reliability
  - inclusion of complex designs

# Digital Systems

Judged by its objectives in application domain

- Performance
- Design and Manufacturing cost
- Ease of Programmability

***It depends on both the hardware and software components***

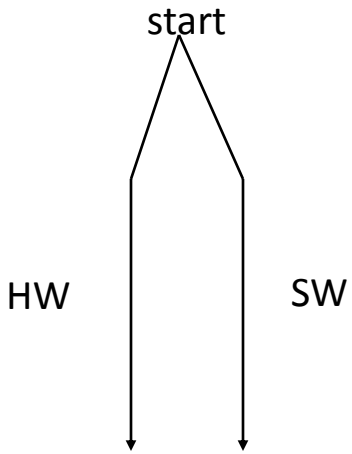
# Hardware/Software Codesign

A definition:

Meeting System level objectives by exploiting the synergism of hardware and software through their concurrent design

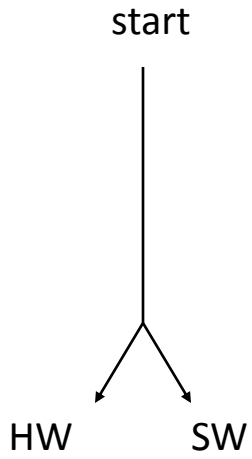
# Concurrent design

## Traditional design flow




Designed by independent groups of experts

## Concurrent (codesign) flow




Designed by Same group of experts with cooperation

# Codesign motivation

Trend toward smaller mask-level geometries leads to:

- Higher integration and cost of fabrication.
- Amortize hardware design over large volume productions

Suggestion:

***Use software as a means of differentiating products based on the same hardware platform***.

# Story of IP cores

What are these IP Cores?

*Predesigned, preverified silcon circuit block, usually containing 5000 gates, that can be used in building larger application on a semiconductor chip.*

Complex macrocells implementing instruction set processors (ISP) are available as cores

- Hardware (core)
- Software (microkernels)

Are viewed as ***intelectual property***

# IP core reuse

- Cores are standardized for reuse as system building blocks

  Rationale: leveraging the existing software layers including OS and applications in ES

Results:

1. **Customized VLSI chip with better area/ performance/ power trade-offs**
2. **Systems on Silicon**

# Hardware Programmability

Traditionally

- Hardware used to be configured at the time of manufacturing
- Software is variant at run time

The Field Programmable Gate Arrays (FPGA) has blurred this distinction.

# FPGAs

- FPGA circuits can be configured on-the-fly to implement a specific software function with better performance than on microprocessor.
- FPGA can be reprogrammed to perform another specific function without changing the underlying hardware.

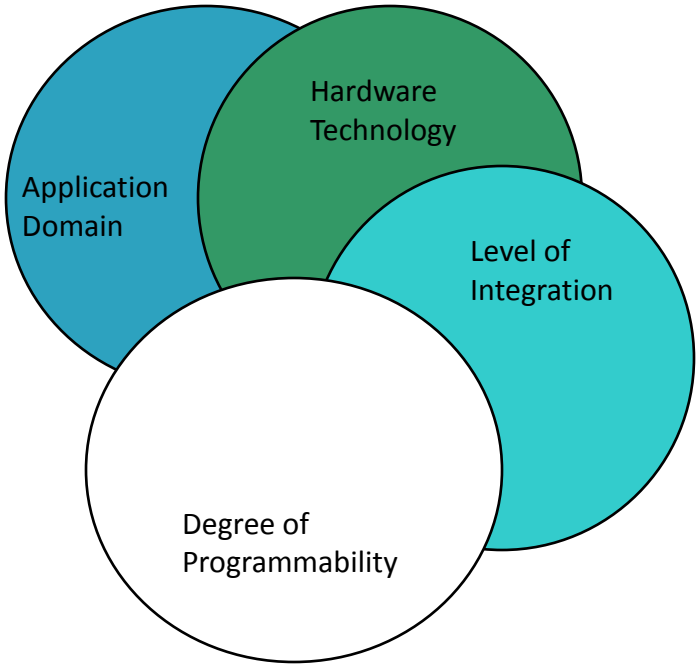
*This flexibility opens new applications of digital circuits.*

# Why codesign?

- Reduce time to market
- Achieve better design
  - Explore alternative designs
  - Good design can be found by balancing the HW/SW
- To meet strict design constraint
  - power, size, timing, and performance trade-offs
  - safety and reliability
  - system on chip

# Distinguishing features of digital system

- Interrelated criteria for a system design



# Application Domains

- General purpose computing system
  - usually self contained and with peripherals
  - Information processing systems
- Dedicated control system
  - part of the whole system, Ex: digital controller in a manufacturing plant
  - also, known as embedded systems

# Embedded Systems

- Uses a computer to perform certain functions
- Conceived with specific application in mind
  - examples: controllers in autombiles, remote controller for robots, answering machines, etc.
- User has limited access to system programming
  - system is provided with system software during manufacturing
  - not  used as a computer

# Degree of Programmability

Most digital systems are programmed by some software programs for functionality.

Two important issues related to programming:

- who has the access to programming?
- Level at which programming is performed.

# Degree of Programmability: Accessibility

Understand the role of:

*End users, application developers, system integrator and component manufacturers.*

Application Developer: System to be retargetable.

System Integrator: Ensure compatibility of system components

Component Manufactures: Concerned with maximizing product reuse

# Degree of Programmability

**Example 1**: Personal computer

End User: Limited to application level

Application Dev.: Language tools, Operating System, high-level programming environment (off the self components)

Component Manf.: Drive by bus standards, protocols etc.

# Example 2.

Embedded Systems

- End user: Limited access to programming
- Most software is already provided by system integrator who could be application developer too!

# Level of Programmability

- Systems can be programmed at *application, instruction and hardware* levels
- Application Level: Allows users to specify “option of functionality” using special language.
- Example: Programming VCR or automated steering control of a ship

# Level of Programmability

- Instruction-level programmability
  - Most common ways with ISA processors or DSP
    - compilers are used in case of computers
    - In case of embedded systems, ISA is NOT visible

# Level of programmability

- Hardware level programmability

⇓

configuring the hardware (after manufacturing) in the desired way.

Example: Microprogramming (determine the behavior of control unit by microprogram)

- Emulating another architecture by alternation of μp
- Some DSP implementations too
- Never in RISC or ISA processors

# Programmability

Microprogramming Vrs. Reconfigurability

Microprogram allows reconfigure the control unit versus Reconfigurable system can modify both datapath and controller.

Reconfigurability increases usability but not the performance of a system.

# Performance and Programmability

- **General computing applications**: use of superscalar RISC architecture to improve the performance (instruction level programming)
- **Dedicated Applications**: Use of application specific designs (ASICs) for power and performance
  - Neither reusable nor cheap!

What if ASICs with embedded cores?

# Performance and Programmability

- Any other solutions?

  **How about replacing the standard processors by application specific processors that can be programmed at instruction level (ASIPs).**

  - Better power-performance than standard processor ?
  - Worse than ASICs

# Programmability and Cost: ASIPs

- Cost can typically be reduced over larger volume than on ASICs (with multiple applications using ASIPs).
- Ease to update the products and engineering changes through programming the HW,
- However, includes compiler as additional cost

# Hardware Technology

- Choice of hardware to implement the design affects the performance and cost
- VLSI technology (CMOS or bipolar, scale of integration and feature size etc.) can affect the performance and cost.

# Hardware Technology: FPGAs

- Performance is an order of magnitude less than corresponding non-programmable technology with comparable mask size
- For high volume production, these are more expensive than ASICs

# Level of Integration

- Integration leads to reducing number of parts, which means, increased reliability, reduced power and higher performances
- But it increases the chip size (cost) and makes debugging more challenging.
- Standard components for SoC are cores, memory, sensors and actuators.

# Embedded System Design Objective

- Embedded systems:

  control systems: reactive, real-time

  ♦ function & size: micro controller to high throughput data-processor

    - requires leveraging the components and cores of microprocessors

- reliability, availability and safety are vital
    - use of formal verification to check the correctness
    - may use redundancy

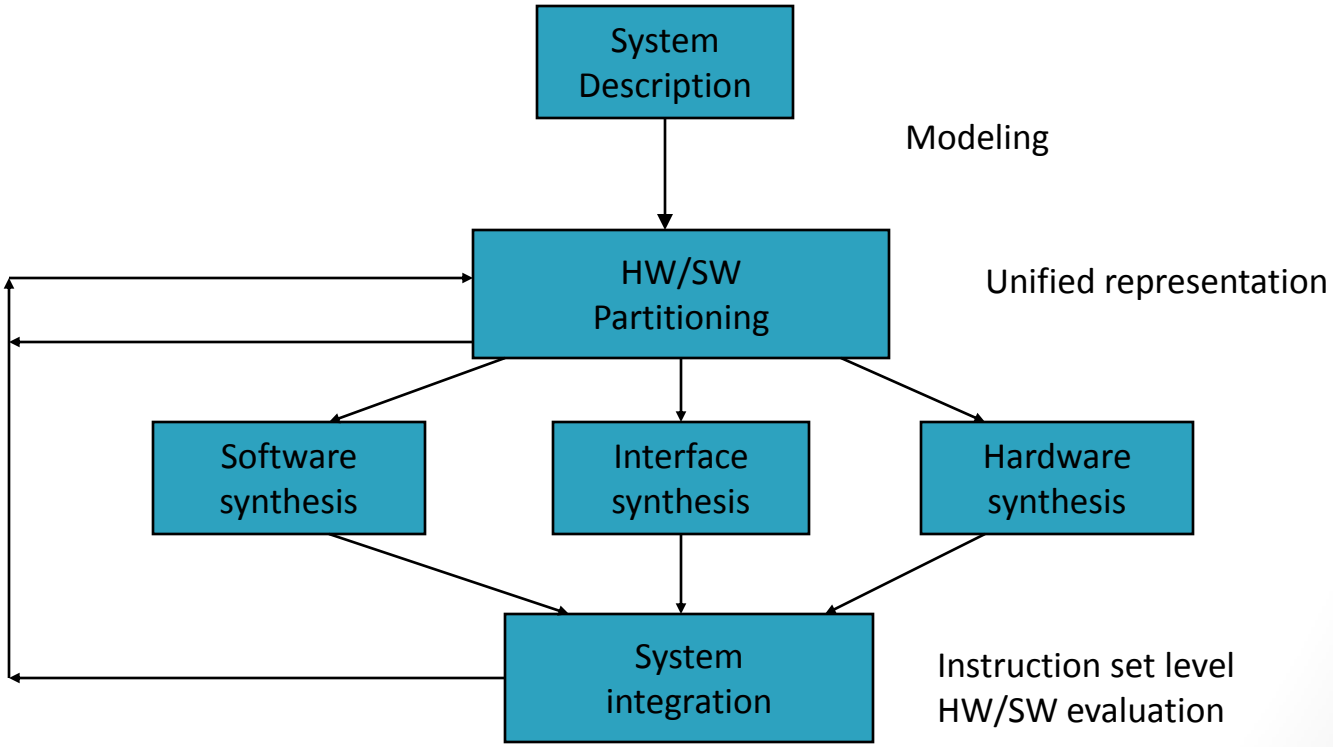
# Challenges with ASIP

- Compatibility requirement is less important
- Goal: support specific instruction mixes

> Price of the flexibility in choosing mixed instruction set is to develop the application specific compiler.

- CAD of compiler is partly solved problem

# Typical codesign process



# Steps in Codesign

HW-SW system involves

- **specification**
- **modeling**
- design space exploration and partitioning
- synthesis and optimization
- **validation**
- **implementation**

# Steps in codesign

Specification

- List the functions of a system that describe the behavior of an abstraction clearly with out ambiguity.

Modeling:

- Process of conceptualizing and refining the specifications, and producing a hardware and software model.

# Modeling style

- Homogeneous: a modeling language or a graphical formalism for presentation
  - partitioning problem used by the designer
- Heterogeneous: multiple presentations
  - partitioning is specified by the models

# Steps in codesign

Validation:

Process of achieving a reasonable level of confidence that the system will work as designed.

- Takes different flavors per application domain: cosimulation for performance and correctness

# Steps in codesign

Implementation:

Physical realization of the hardware (through synthesis) and of executable software (through compilation).

# Partitioning and Scheduling (where and when)

- A hardware/software *partitioning* represents a physical partition of system functionality into application-specific hardware and software.
- *Scheduling* is to assign an execution start time to each task in a set, where tasks are linked by some relations.

- For more details, refer to:
  - **Hardware Software Codesign of Embedded System**, CPSC489-501 by Rabi Mahapatra.
  - Staunstrup and Wolf Ed. "**Hardware Software codesign: principles and practice**", Kluwer Publication, 1997

- The lecture is available online at:
  - http://bu.edu.eg/staff/ahmad.elbanna-courses/12134
- For inquires, send to:
  - ahmad.elbanna@feng.bu.edu.eg