Hardware-Software Codesign
Rapid Prototyping Design Process

**SYSTEM DEF.**

Primarily software

**VIRTUAL PROTOTYPE**

Primarily hardware

**FUNCTION DESIGN**

**HW & SW CODESIGN**

**HW & SW PART.**

**HW DESIGN**

**SW DESIGN**

**HW FAB**

**SW CODE**

**INTEG. & TEST**

REUSE DESIGN LIBRARIES AND DATABASE

**HW & SW Partitioning & Codesign**
Introduction to Embedded Systems and Hardware-Software Codesign

- Introduction
  - Unified HW/SW Representations
  - HW/SW Partitioning Techniques
  - Integrated HW/SW Modeling Methodologies
  - HW and SW Synthesis Methodologies
  - Industry Approaches to HW/SW Codesign
  - Hardware/Software Codesign Research
  - Summary
Codesign Definition and Key Concepts

• Codesign
  – The meeting of system-level objectives by exploiting the trade-offs between hardware and software in a system through their concurrent design

• Key concepts
  – Concurrent: hardware and software developed at the same time on parallel paths
  – Integrated: interaction between hardware and software developments to produce designs that meet performance criteria and functional specifications
Motivations for Codesign

• Factors driving codesign (hardware/software systems):
  – Instruction Set Processors (ISPs) available as cores in many design kits (386s, DSPs, microcontrollers, etc.)
  – Systems on Silicon - many transistors available in typical processes (> 10 million transistors available in IBM ASIC process, etc.)
  – Increasing capacity of field programmable devices - some devices even able to be reprogrammed on-the-fly (FPGAs, CPLDs, etc.)
  – Efficient C compilers for embedded processors
  – Hardware synthesis capabilities
Motivations for Codesign (cont.)

• The importance of codesign in designing hardware/software systems:
  – Improves design quality, design cycle time, and cost
    • Reduces integration and test time
  – Supports growing complexity of embedded systems
  – Takes advantage of advances in tools and technologies
    • Processor cores
    • High-level hardware synthesis capabilities
    • ASIC development
Categorizing Hardware/Software Systems

• Application Domain
  – Embedded systems
    • Manufacturing control
    • Consumer electronics
    • Vehicles
    • Telecommunications
    • Defense Systems
  – Instruction Set Architectures
  – Reconfigurable Systems
• Degree of programmability
  – Access to programming
  – Levels of programming
• Implementation Features
  – Discrete vs. integrated components
  – Fabrication technologies
Categories of Codesign Problems

• Codesign of embedded systems
  – Usually consist of sensors, controller, and actuators
  – Are reactive systems
  – Usually have real-time constraints
  – Usually have dependability constraints

• Codesign of ISAs
  – Application-specific instruction set processors (ASIPs)
  – Compiler and hardware optimization and trade-offs

• Codesign of Reconfigurable Systems
  – Systems that can be personalized after manufacture for a specific application
  – Reconfiguration can be accomplished before execution or concurrent with execution (called evolvable systems)
Components of the Codesign Problem

• Specification of the system
• Hardware/Software Partitioning
  – Architectural assumptions - type of processor, interface style between hardware and software, etc.
  – Partitioning objectives - maximize speedup, latency requirements, minimize size, cost, etc.
  – Partitioning strategies - high level partitioning by hand, automated partitioning using various techniques, etc.
• Scheduling
  – Operation scheduling in hardware
  – Instruction scheduling in compilers
  – Process scheduling in operating systems
• Modeling the hardware/software system during the design process
Embedded Systems

Application-specific systems which contain hardware and software tailored for a particular task and are generally part of a larger system (e.g., industrial controllers)

• Characteristics
  – Are dedicated to a particular application
  – Include processors dedicated to specific functions
  – Represent a subset of reactive (responsive to external inputs) systems
  – Contain real-time constraints
  – Include requirements that span:
    • Performance
    • Reliability
    • Form factor
Embedded Systems: Specific Trends

- Use of microprocessors only one or two generations behind state-of-the-art for desktops
  - E.g. $N/2$ bit width where $N$ is the bit width of current desktop systems
- Contain limited amount of memory
- Must satisfy strict real-time and/or performance constraints
- Must optimize additional design objectives:
  - Cost
  - Reliability
  - Design time
- Increased use of hardware/software codesign principles to meet constraints
Embedded Systems: Examples

- Banking and transaction processing applications
- Automobile engine control units
- Signal processing applications
- Home appliances (microwave ovens)
- Industrial controllers in factories
- Cellular communications
Embedded Systems: Complexity Issues

- Complexity of embedded systems is continually increasing
- Number of states in these systems (especially in the software) is very large
- Description of a system can be complex, making system analysis extremely hard
- Complexity management techniques are necessary to model and analyze these systems
- Systems becoming too complex to achieve accurate “first pass” design using conventional techniques
- New issues rapidly emerging from new implementation technologies
Techniques to Support Complexity Management

- Delayed HW/SW partitioning
  - Postpone as many decisions as possible that place constraints on the design
- Abstractions and decomposition techniques
- Incremental development
  - “Growing” software
  - Requiring top-down design
- Description languages
- Simulation
- Standards
- Design methodology management framework
A Model of the Current Hardware/Software Design Process

DOD-STD-2167A

System Concepts
- HW Development
  - Sys/HW Require. Analysis
  - Hardware Design
- SW Development
  - Sys/SW Require. Analysis
  - Software Require. Analysis
  - Prelim. Design
  - Detailed Design

SW Development
- Fabric.
- HWCI Testing
- System Integ. and test
- CSCI Testing
  - Coding, Unit test., Integ. test

© IEEE 1991

[Franke91]
Current Hardware/Software Design Process

• Basic features of current process:
  – System immediately partitioned into hardware and software components
  – Hardware and software developed separately
  – “Hardware first” approach often adopted

• Implications of these features:
  – HW/SW trade-offs restricted
    • Impact of HW and SW on each other cannot be assessed easily
  – Late system integration

• Consequences these features:
  – Poor quality designs
  – Costly modifications
  – Schedule slippages
Incorrect Assumptions in Current Hardware/Software Design Process

• Hardware and software can be acquired separately and independently, with successful and easy integration of the two later

• Hardware problems can be fixed with simple software modifications

• Once operational, software rarely needs modification or maintenance

• Valid and complete software requirements are easy to state and implement in code
Directions of the HW/SW Design Process

Integrated Modeling Substrate

HW Development
- Sys/HW Require. Analysis
- Hardware Design
- System Integ. and test
- HWCI Testing
- SW Development
- System Concepts
- Prelim. Design
- Detailed Design
- Coding, Unit test., Integ. test
- CSCI Testing
- Operation. Testing and Evaluation

Sys/SW Require. Analysis
- Hardware Require. Analysis
- Software Require. Analysis

SW Development
- SW Development

© IEEE 1991

[Franke91]
Requirements for the Ideal Codesign Environment

• Unified, unbiased hardware/software representation
  – Supports uniform design and analysis techniques for hardware and software
  – Permits system evaluation in an integrated design environment
  – Allows easy migration of system tasks to either hardware or software

• Iterative partitioning techniques
  – Allow several different designs (HW/SW partitions) to be evaluated
  – Aid in determining best implementation for a system
  – Partitioning applied to modules to best meet design criteria (functionality and performance goals)
Requirements for the Ideal Codesign Environment (cont.)

• Integrated modeling substrate
  – Supports evaluation at several stages of the design process
  – Supports step-wise development and integration of hardware and software

• Validation Methodology
  – Insures that system implemented meets initial system requirements
Cross-fertilization Between Hardware and Software Design

• Fast growth in both VLSI design and software engineering has raised awareness of similarities between the two
  – Hardware synthesis
  – Programmable logic
  – Description languages

• Explicit attempts have been made to “transfer technology” between the domains
Cross-fertilization Between Hardware and Software Design (cont.)

- EDA tool technology has been transferred to SW CAD systems
  - Designer support (not automation)
  - Graphics-driven design
  - Central database for design information
  - Tools to check design behavior early in process
Cross-fertilization Between Hardware and Software Design (cont.)

• Software technology has been transferred to EDA tools
  – Single-language design
    • Use of 1 common language for architecture spec. and implementation of a chip
  – Compiler-like transformations and techniques
    • Dead code elimination
    • Loop unrolling
  – Design change management
    • Information hiding
    • Design families
Typical Codesign Process

- System Integration
- Instruction set level
- HW/SW evaluation
- Unified representation (Data/control flow)
- Concurrent processes Programming languages
- FSM-directed graphs
- HW/SW Partitioning
- Software Synthesis
- Interface Synthesis
- Hardware Synthesis
- System Integration
- Instruction set level HW/SW evaluation
- Another HW/SW partition
Conventional Codesign Methodology

Analysis of Constraints and Requirements

System Specs..

HW/SW Partitioning

Hardware Descript.

Software Descript.

HW Synth. and Configuration

Interface Synthesis

Software Gen. & Parameterization

Configuration Modules

Hardware Components

HW/SW Interfaces

Software Modules

HW/SW Integration and Cosimulation

Integrated System

System Evaluation

Design Verification

© IEEE 1994
[Rozenblit94]
Codesign Features

Basic features of a codesign process

• Enables mutual influence of both HW and SW early in the design cycle
  – Provides continual verification throughout the design cycle
  – Separate HW/SW development paths can lead to costly modifications and schedule slippages

• Enables evaluation of larger design space through tool interoperability and automation of codesign at abstract design levels

• Advances in key enabling technologies (e.g., logic synthesis and formal methods) make it easier to explore design tradeoffs