Hardware Software Codesign of Embedded System

CPSC689-602

Rabi Mahapatra

Today's topics

- Course Organization

- Introduction to HS-CODES

- Codesign Motivation

- Some Issues on Codesign of Embedded System

Course Organization

Lectures: HRBB 104, TR 3:55 - 5:10 PM

Laboratory: HRBB 218, TBD

Instructor's office Hours: By appointment

Contact: rabi@cs.tamu.edu, 5-5787

Course organization: Gradings

Projects: 70%

Seminar/Term paper/Assignment: 30%

Labs: Team work

Seminar/Term papers/Project: Individual or Team of two students

Course policies

- Required to access the course web page for relevant info during the semester

- Class attendance is required

- use emails for effective communication with Instructor

- all assigned papers are required materials

- follow lab and univ rules

Topics to be covered
(order and details may change)

1. Codesign overview
2. Models and methodologies of system design
3. Hardware software partitioning and scheduling
4. Cosimulation, synthesis and verifications
5. Architecture mapping, HW-SW Interfaces and Reconfigurable computing
6. System on Chip (SoC) and IP cores
7. Low-Power RT Embedded Systems
8. On-chip Networking
9. Software for Embedded Systems
10. Sensor Networks

Laboratory Activites

- Synopsys's Design Environment
  - Cocentric Tools for the class – use of SystemC
- FPGA based design – use of Xilinx's design environment and devices (VHDL / Verilog)
- Can use Lab for project work
- Embedded Systems: PowerPC, StrongARM and Motorola's ColdFire
- Low-Power measurement setup using LabView tools
- Use of tools and language based on choice of projects

Texts & References

- G Micheli, R Ernst and W.Wolf, editors, "Readings in Hardware/Software Co-Design", Morgan Kaufman Publisher, 2002

- Lecture notes: faculty.cs.tamu.edu/rabi/cpsc689/

- Staunstrup and Wolf Ed. "Hardware Software codesign: principles and practice", Kluwer Publication, 1997 (Reference)

Reading assignment

- Giovanni De Micheli and Rajesh Gupta, "Hardware/Software co-design", IEEE Proceedings, vol. 85, no.3, March 1997, pp. 349-365.

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

Microelectronics trends

- 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

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
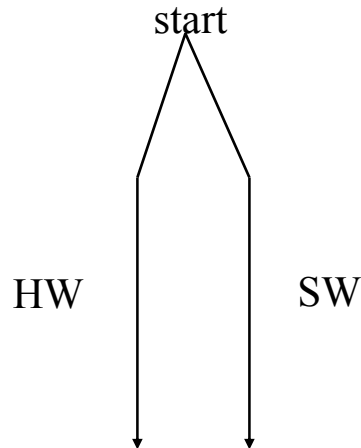
Hardware/Software Codesign

A definition:

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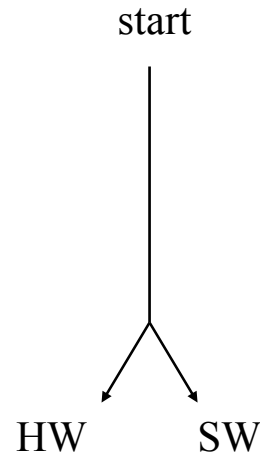
Concurrent design

Traditional design flow

Concurrent (codesign) flow

start

HW   SW

Designed by independent groups of experts

start

HW   SW

Designed by Same group of experts with cooperation

Codesign motivation

Trend toward smaller mask-level geometries leads to:

- Higher integration and cost of fabrication.

- <u>Amortize hardware design over large volume productions</u>

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

Story of IP cores

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

- Hardware (core)

- Software (micro kernels)

Are viewed as ***intellectual 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.

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.

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

Why codesign?

- Reduce time to market

- Achieve better design
    - Explore alternative designs
   - Good design can be found by balancing the HW/SW

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
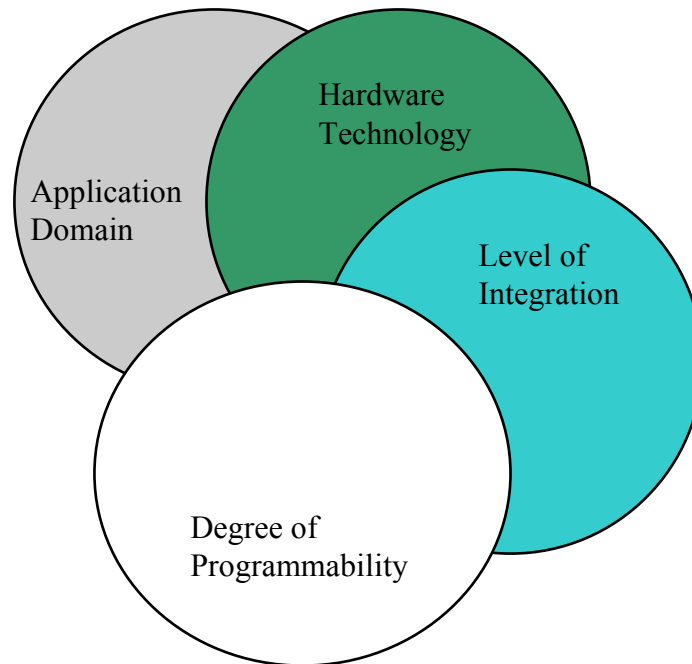
Application Domains

- General purpose computing system
  - usually self contained and with peripherals
  - Information processing systems

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 System

- Uses a computer to perform certain functions

- Conceived with specific application in mind
  - examples: dash controller in automobiles, remote controller for robots, answering machines, etc.

Embedded Systems

- Uses a computer to perform certain functions

- Conceived with specific application in mind
  - examples: dash controller in automobiles, 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.

Degree of Programmability: Accessibility

Understand the role of:

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

Application Developer: System to be retargetable.

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

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.

Observe that: coalescing the system components due to higher chip density
Result: Few but more versatile system hardware components

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 Vs.. Reconfigurability

> Microprogram allows to reconfigure the control unit whereas Reconfigurable system can modify both datapath and controller.

Programmability

Microprogramming Vs.. 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)

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!

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 amortized 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

- Power Issues?

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

Codesign of ISA

- ISA is fundamental to digital system design

- An instruction set permits concurrent design of HW and compiler developments

- Good ISA design is critical in achieving system usability across applications

- Goal of codesign in ISP development is to optimize HW utilization by application & OS

Codesign of ISA

- For high performance in ES, selection of instruction set that matches the application is very important

    - replace the standard core by ASIP

- ASIPs are more flexible than ASICs but less than ISP

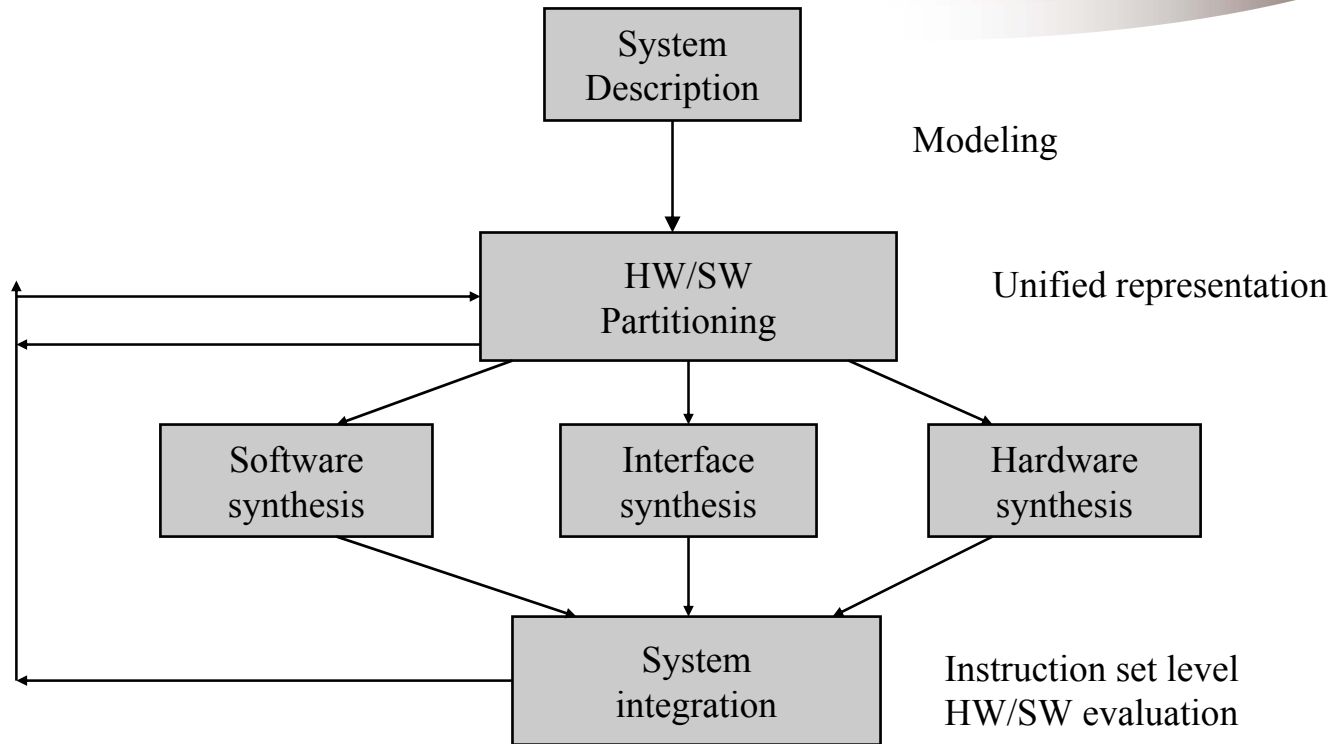- ASIP performs better than ISP

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

System
Description

Modeling

HW/SW
Partitioning

Unified representation

Software
synthesis

Interface
synthesis

Hardware
synthesis

System
integration

Instruction set level
HW/SW evaluation

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 without 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.

Summary:
Research areas in codesign

- Languages
- Architectural exploration tools
- Algorithms for partitioning
- Scheduling
- SW, HW and interface Synthesis
- Verification and Testing