Reading Assignment

Next lecture: 1, 2
Outline

Administrative Issues

Embedded System Design
Instructor

- Professor Jia Wang
- Office: 317 Siegel Hall
- Phone: 312 567-3696
- E-Mail: jwang@ece.iit.edu
  - Please start your email subject line with [ECE587].
- Office hours:
  - Fri. 10:00 AM – 12:00 NOON
  - Or by appointment
Lecture Information

- **Time:** Mon./Wed. 3:15 PM – 4:30 PM
- **Location:** Stuart Building 113
- **Home Page:**
  - We will use Blackboard for grading and online homework submission.
  - Important messages will also be sent to your IIT email.
- **Required Textbook**
Prerequisites

CS 201 and ECE 441

- Object-oriented programming: class, inheritance, polymorphism
  - Please check Object-Oriented Programming Prerequisite on the course website.
- Data structure and algorithm: sorting, vector, linked list
- Computer system: processor/assembly, memory/cache, I/O, interrupts.
- You are recommended to take at least one course among ECE 429, ECE 449, and ECE 485 before taking this course.
Main topic: Embedded System Design

- Hardware/software co-design methodology
- Specification and Modeling
- Synthesis and Optimization
- Verification
Homeworks/Projects

- **3 Homeworks**
  - Submit online in Blackboard.
  - Paper or email submissions are **NOT** recommended.

- **2 Projects**
  - With SystemC
  - Detailed instructions will be posted later.
Projects/homeworks should be done individually.

Discussions on homeworks/projects are encouraged.

All writings and code should be **BY YOURSELF**.

**PLAGIARISM** will call for **DISCIPLINARY ACTION**.

- Check the CODE OF ACADEMIC HONESTY section in the student handbook for possible consequences.

**NEVER** share your writing/code with others.

- It is not possible to determine WHO COPIES FROM WHOM.
- All parties involved will be subject to punishment.
Exams

- Midterm: Wed. Mar 11, 3:15 PM – 4:30 PM
- Final: TBD
- Closed book/notes, cheat sheet allowed
- Check [http://www.iit.edu/registrar/important_dates/final_exam_schedule.shtml](http://www.iit.edu/registrar/important_dates/final_exam_schedule.shtml) for other issues.
- Makeup exams will NOT be given.
Grading

- **Percentage**
  - Homeworks: 10%
  - Projects: 10% + 30% = 40%
  - Midterm Exam: 20%
  - Final Exam: 30%

- **Letter grade**
  - A: 90
  - B: 75
  - C: 60
Administrative Issues

Embedded System Design
What are embedded systems?

- **System**: a computer system whose functionality can be improved and extended with additional hardware and software.
- **Embedded**
  - Various form factors
  - Tailored for certain applications
  - Reactive, respond to external events
What are embedded systems?

- System: a computer system whose functionality can be improved and extended with additional hardware and software.
- Embedded
  - Various form factors
  - Tailored for certain applications
  - Reactive, respond to external events
Embedded System Applications

- Consumer electronics
  - Almost everything
- Others
  - IT infrastructure
  - Industrial and medical equipments
  - Defense systems
- It’s everywhere nowadays.
Embedded System Applications

- Consumer electronics
  - Almost everything
- Others
  - IT infrastructure
  - Industrial and medical equipments
  - Defense systems
- It’s everywhere nowadays.
Embedded System Applications

- Consumer electronics
  - Almost everything
- Others
  - IT infrastructure
  - Industrial and medical equipments
  - Defense systems
- It’s everywhere nowadays.
History

- Enabled by process scaling: Moore’s Law
  - The ability to integrate more transistors on a chip with reduced cost
- ’80s to ’90s: Very-Large-Scale Integration (VLSI)
  - Automatic synthesis from RTL to layout
- ’90s to 2000s: System-on-a-Chip (SoC)
  - The whole system can be integrated into a single chip.
- 2000s to current: Multiprocessor SoC (MPSoC)
  - Power becomes a major limiting factor.
- Future trend: IoTs allow Big Data systems to interact with real physical world.
History

- Enabled by process scaling: Moore’s Law
  - The ability to integrate more transistors on a chip with reduced cost
- ’80s to ’90s: Very-Large-Scale Integration (VLSI)
  - Automatic synthesis from RTL to layout
- ’90s to 2000s: System-on-a-Chip (SoC)
  - The whole system can be integrated into a single chip.
- 2000s to current: Multiprocessor SoC (MPSoC)
  - Power becomes a major limiting factor.
- Future trend: IoTs allow Big Data systems to interact with real physical world.
History

- Enabled by process scaling: Moore’s Law
  - The ability to integrate more transistors on a chip with reduced cost
- ’80s to ’90s: Very-Large-Scale Integration (VLSI)
  - Automatic synthesis from RTL to layout
- ’90s to 2000s: System-on-a-Chip (SoC)
  - The whole system can be integrated into a single chip.
- 2000s to current: Multiprocessor SoC (MPSoC)
  - Power becomes a major limiting factor.
- Future trend: IoTs allow Big Data systems to interact with real physical world.
History

- Enabled by process scaling: Moore’s Law
  - The ability to integrate more transistors on a chip with reduced cost
- ’80s to ’90s: Very-Large-Scale Integration (VLSI)
  - Automatic synthesis from RTL to layout
- ’90s to 2000s: System-on-a-Chip (SoC)
  - The whole system can be integrated into a single chip.
- 2000s to current: Multiprocessor SoC (MPSoC)
  - Power becomes a major limiting factor.
- Future trend: IoTs allow Big Data systems to interact with real physical world.
History

- Enabled by process scaling: Moore’s Law
  - The ability to integrate more transistors on a chip with reduced cost
- ’80s to ’90s: Very-Large-Scale Integration (VLSI)
  - Automatic synthesis from RTL to layout
- ’90s to 2000s: System-on-a-Chip (SoC)
  - The whole system can be integrated into a single chip.
- 2000s to current: Multiprocessor SoC (MPSoC)
  - Power becomes a major limiting factor.
- Future trend: IoTs allow Big Data systems to interact with real physical world.
Implement a system with desired functionality

Subject to design constraints

Different embedded systems do have quite different design constraints.
Implement a system with desired functionality

Subject to design constraints
  - Different embedded systems do have quite different design constraints.
Design Constraints

- **Time-to-market**
- **Cost**
  - Non-recurring engineering (NRE) cost
  - Production/Unit cost
- **Performance**
  - Speed: latency, throughput
  - Power/energy consumption
- **Robustness and reliability**
- **Others: thermal, form factor, etc.**
Design Constraints

- Time-to-market
- Cost
  - Non-recurring engineering (NRE) cost
  - Production/Unit cost
- Performance
  - Speed: latency, throughput
  - Power/energy consumption
- Robustness and reliability
- Others: thermal, form factor, etc.
Design Constraints

- Time-to-market
- Cost
  - Non-recurring engineering (NRE) cost
  - Production/Unit cost
- Performance
  - Speed: latency, throughput
  - Power/energy consumption
- Robustness and reliability
- Others: thermal, form factor, etc.
Design Constraints

- Time-to-market
- Cost
  - Non-recurring engineering (NRE) cost
  - Production/Unit cost
- Performance
  - Speed: latency, throughput
  - Power/energy consumption
- Robustness and reliability
- Others: thermal, form factor, etc.
Design Constraints

- Time-to-market
- Cost
  - Non-recurring engineering (NRE) cost
  - Production/Unit cost
- Performance
  - Speed: latency, throughput
  - Power/energy consumption
- Robustness and reliability
- Others: thermal, form factor, etc.
Challenges in Modern Embedded System Design

- Rich functionality
  - Short time-to-market
    - New generations of hardware are introduced to market very frequently.
  - Low cost
    - Less NRE cost: less risky for investors
    - Less unit cost: more profitable
- Stringent performance constraints
  - Especially for power consumption due to limitations on battery and heat dissipation.
Challenges in Modern Embedded System Design

- Rich functionality
- Short time-to-market
  - New generations of hardware are introduced to market very frequently.
- Low cost
  - Less NRE cost: less risky for investors
  - Less unit cost: more profitable
- Stringent performance constraints
  - Especially for power consumption due to limitations on battery and heat dissipation.
Challenges in Modern Embedded System Design

- Rich functionality
- Short time-to-market
  - New generations of hardware are introduced to market very frequently.
- Low cost
  - Less NRE cost: less risky for investors
  - Less unit cost: more profitable
- Stringent performance constraints
  - Especially for power consumption due to limitations on battery and heat dissipation.
Challenges in Modern Embedded System Design

▶ Rich functionality
▶ Short time-to-market
  ▶ New generations of hardware are introduced to market very frequently.
▶ Low cost
  ▶ Less NRE cost: less risky for investors
  ▶ Less unit cost: more profitable
▶ Stringent performance constraints
  ▶ Especially for power consumption due to limitations on battery and heat dissipation.
Embedded System as Integration of Hardware and Software

- **Embedded Hardware**
  - Processors, memories, and standard interfaces
  - Programmable and reconfigurable hardware, e.g. FPGA
  - Application specific integrated circuits (ASICs)
  - Sensors and actuators

- **Embedded Operating System**
  - Provide abstractions of hardware
  - Common OS supports, e.g. file system and multitasking
  - May provide real-time guarantees

- **Embedded Software**
  - Run on processors, could be ported from existing systems
  - Utilize certain embedded hardware to interact with external environment
  - Could be updated
Embedded System as Integration of Hardware and Software

- **Embedded Hardware**
  - Processors, memories, and standard interfaces
  - Programmable and reconfigurable hardware, e.g. FPGA
  - Application specific integrated circuits (ASICs)
  - Sensors and actuators

- **Embedded Operating System**
  - Provide abstractions of hardware
  - Common OS supports, e.g. file system and multitasking
  - May provide real-time guarantees

- **Embedded Software**
  - Run on processors, could be ported from existing systems
  - Utilize certain embedded hardware to interact with external environment
  - Could be updated
Embedded System as
Integration of Hardware and Software

- **Embedded Hardware**
  - Processors, memories, and standard interfaces
  - Programmable and reconfigurable hardware, e.g. FPGA
  - Application specific integrated circuits (ASICs)
  - Sensors and actuators

- **Embedded Operating System**
  - Provide abstractions of hardware
  - Common OS supports, e.g. file system and multitasking
  - May provide real-time guarantees

- **Embedded Software**
  - Run on processors, could be ported from existing systems
  - Utilize certain embedded hardware to interact with external environment
  - Could be updated
There are multiple choices to implement certain functionality

- Implement as software
  - Short time-to-market and low NRE cost
  - Performance is a concern, especially with low budget on unit cost
- Implement as hardware, i.e. ASICs
  - High speed, low power, low unit cost
  - Long time-to-market and high NRE cost
- Implement as something in the middle, e.g. FPGA
- Partitioning functionalities among hardware/software (HW/SW partitioning) is critical for any successful embedded system design.
There are multiple choices to implement certain functionality:

- **Implement as software**
  - Short time-to-market and low NRE cost
  - Performance is a concern, especially with low budget on unit cost

- **Implement as hardware, i.e. ASICs**
  - High speed, low power, low unit cost
  - Long time-to-market and high NRE cost

- **Implement as something in the middle, e.g. FPGA**

- Partitioning functionalities among hardware/software (HW/SW partitioning) is critical for any successful embedded system design.
There are multiple choices to implement certain functionality

- **Implement as software**
  - Short time-to-market and low NRE cost
  - Performance is a concern, especially with low budget on unit cost

- **Implement as hardware, i.e. ASICs**
  - High speed, low power, low unit cost
  - Long time-to-market and high NRE cost

- **Implement as something in the middle, e.g. FPGA**

- Partitioning functionalities among hardware/software (HW/SW partitioning) is critical for any successful embedded system design.
There are multiple choices to implement certain functionality

Implement as software
  - Short time-to-market and low NRE cost
  - Performance is a concern, especially with low budget on unit cost

Implement as hardware, i.e. ASICs
  - High speed, low power, low unit cost
  - Long time-to-market and high NRE cost

Implement as something in the middle, e.g. FPGA
  - Partitioning functionalities among hardware/software (HW/SW partitioning) is critical for any successful embedded system design.
Economics of Embedded System Design

- There are multiple choices to implement certain functionality
  - Implement as software
    - Short time-to-market and low NRE cost
    - Performance is a concern, especially with low budget on unit cost
  - Implement as hardware, i.e. ASICs
    - High speed, low power, low unit cost
    - Long time-to-market and high NRE cost
  - Implement as something in the middle, e.g. FPGA
  - Partitioning functionalities among hardware/software (HW/SW partitioning) is critical for any successful embedded system design.
Ad-Hoc Embedded System Design Methodology

- Choose a set of embedded hardware from the market
  - Reduce NRE cost by NOT designing your own
- Come up with a HW/SW partitioning, implement it, and evaluate its performance and various characteristics
  - Use CAD/EDA tools whenever possible to reduce design time and thus time-to-market
- Repeat the above step until all requirements are met
- Essentially hardware and software are designed in sequel.
  - However, as new hardware comes out very quickly, the overall system design will become obsolete very soon.
  - Time-to-market should be further reduced.
  - Can we make the whole system available as soon as the hardware becomes available?
Ad-Hoc Embedded System Design Methodology

- Choose a set of embedded hardware from the market
  - Reduce NRE cost by NOT designing your own
- Come up with a HW/SW partitioning, implement it, and evaluate its performance and various characteristics
  - Use CAD/EDA tools whenever possible to reduce design time and thus time-to-market
- Repeat the above step until all requirements are met
- Essentially hardware and software are designed in sequel.
  - However, as new hardware comes out very quickly, the overall system design will become obsolete very soon.
  - Time-to-market should be further reduced.
  - Can we make the whole system available as soon as the hardware becomes available?
Ad-Hoc Embedded System Design Methodology

- Choose a set of embedded hardware from the market
  - Reduce NRE cost by NOT designing your own
- Come up with a HW/SW partitioning, implement it, and evaluate its performance and various characteristics
  - Use CAD/EDA tools whenever possible to reduce design time and thus time-to-market
- Repeat the above step until all requirements are met
  - Essentially hardware and software are designed in sequel.
    - However, as new hardware comes out very quickly, the overall system design will become obsolete very soon.
    - Time-to-market should be further reduced.
    - Can we make the whole system available as soon as the hardware becomes available?
Ad-Hoc Embedded System Design Methodology

- Choose a set of embedded hardware from the market
  - Reduce NRE cost by NOT designing your own
- Come up with a HW/SW partitioning, implement it, and evaluate its performance and various characteristics
  - Use CAD/EDA tools whenever possible to reduce design time and thus time-to-market
- Repeat the above step until all requirements are met
- Essentially hardware and software are designed in sequel.
  - However, as new hardware comes out very quickly, the overall system design will become obsolete very soon.
  - Time-to-market should be further reduced.
  - Can we make the whole system available as soon as the hardware becomes available?
Hardware/Software Co-Design

- A design methodology that enables designers to design hardware and software together.
  - Concurrent design
    - HW/SW are designed at the same time on parallel paths to reduce the time-to-market.
  - Integrated design
    - Interaction between HW/SW designs enable designers to explore more flexible HW/SW partitionings in order to meet stringent design constraints.
Hardware/Software Co-Design

- A design methodology that enables designers to design hardware and software together.
- Concurrent design
  - HW/SW are designed at the same time on parallel paths to reduce the time-to-market.
- Integrated design
  - Interaction between HW/SW designs enable designers to explore more flexible HW/SW partitionings in order to meet stringent design constraints.
Hardware/Software Co-Design

- A design methodology that enables designers to design hardware and software together.
- Concurrent design
  -HW/SW are designed at the same time on parallel paths to reduce the time-to-market.
- Integrated design
  -Interaction between HW/SW designs enable designers to explore more flexible HW/SW partitionings in order to meet stringent design constraints.
Challenges in Hardware/Software Co-Design

▶ How can one design HW/SW at the same time?
  ▶ The whole system is highly complicated and the overall design should be validated and evaluated.

▶ Is there any tool to help designers exploring various HW/SW partitionings?
  ▶ A more sensible question is that: what methodology should designers follow such that CAD/EDA people could create tools for such purpose?
Challenges in Hardware/Software Co-Design

- How can one design HW/SW at the same time?
  - The whole system is highly complicated and the overall design should be validated and evaluated.
- Is there any tool to help designers exploring various HW/SW partitionings?
  - A more sensible question is that: what methodology should designers follow such that CAD/EDA people could create tools for such purpose?
Summary

- Embedded system design is challenging.
- But we can follow proper design methodology to make it possible.
Summary

- Embedded system design is challenging.
- But we can follow proper design methodology to make it possible.