ECE 587 – Hardware/Software Co-Design
Lecture 01 Introduction

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

January 12, 2026

Reading Assignment

- Next lecture: 1, 2

Outline

Administrative Issues

Computing System Design

Instructor

- Professor Jia Wang
- E-Mail: jwang34@illinoistech.edu
- Office hours: TBD

Lectures

- ▶ Mon./Wed. 11:25 AM – 12:40 PM, Perlstein Hall 131
- ▶ Course website: https://wngjia.github.io/ece587-web/
- ▶ Office hour: Mon. 1:00 PM – 2:00 PM, Siegel Hall 317

Textbooks

- ▶ Recommended Textbook
  - ▶ "Embedded System Design: Modeling, Synthesis and Verification" D. D. Gajski, S. Abdi, A. Gerstlauer, G. Schirner, Springer, 2009. ISBN-13: 978-1-4419-0503-1
- ▶ Plus additional online material and research papers

Prerequisites

CS 201 and ECE 441

- ▶ Object-oriented programming: class, inheritance, polymorphism
- ▶ 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 in VLSI design, software development, and computer architecture concurrently or before taking this course.

Course Outline

Main topic: Hardware/Software Co-Design

- ▶ Models of computation and system modeling
- ▶ Multicore and CUDA programming
- ▶ High-level synthesis
- ▶ Hardware acceleration and interconnection networks.

Homeworks/Projects

- 3 Homeworks
    - 5 points each for a total of 15 points.
- 4 GEMM projects
    - 15 points each for a total of 60 points.
- Literature survey project
    - 30 points.
- Submit online in Canvas only.
- Late homeworks and projects will NOT be graded, unless
    - A request to extend the deadline is received by email 48 hours BEFORE the deadline.
    - With 48 hours of the deadline or after, the request should be accompanied by extraordinary reasons with documented proof like docter's notes, or it will be rejected.

Grading

- 100+5(bonus) points total for Homeworks and Projects
- A: 90
- B: 80
- C: 60

GEMM Projects Setup

- ▶ To explore different software and hardware platforms,
    - ▶ Multicore programming with OpenMP.
    - ▶ CUDA programming.
    - ▶ Hardware accelerator simulation with Chipyard.
- ▶ A recent Windows computer with at least 4 CPU cores, 16GB memory, and 512GB SSD.
    - ▶ Or a x64 Ubuntu server with 4 CPU cores, 8GB memory, and 100GB storage.
    - ▶ We are not able to support ARM-based computers like Apple MacBooks and Raspberry Pi's, as well as computers that are more than 5 years old.
- ▶ Access to Google Colab for CUDA programming.
- ▶ Internet access to common services and repositories like GitHub.

How to ~~survive~~ succeed in this course?

- Read: all instructions are in written.
  - Tutorials, source code, documents, and don't overlook command outputs.
- Communicate: we are very happy to solve any issue you may meet but you need to let us know what's wrong.
  - `https://stackoverflow.com/help/how-to-ask`
- Learn to use AI assistants.
  - `https://www.deeplearning.ai/short-courses/chatgpt-prompt-engineering-for-developers/`
- Feel free to explore new computer hardware and software but make sure they do not interfere with your schedule to meet deadlines.

Ethics (Very Seriously)

- ▶ Read "IIT Code of Academic Honesty" and "IEEE Code of Conduct" (posted on the course website).
  - ▶ Projects/homeworks should be done individually.
  - ▶ Discussions on homeworks/projects are encouraged.
  - ▶ Interactions with AI assistants (prompts and answers) should not be shared since they are considered as your own work.
  - ▶ Source code from the lectures and instructions in this course can be used directly.
  - ▶ Source code from other online sources not directly related to this course may be used with proper references.
- ▶ All other writings and code should be BY YOURSELF.
  - ▶ NEVER SHARE YOUR WRITINGS/CODE WITH OTHERS!
  - ▶ NEVER USE WRITINGS/CODE FROM OTHERS!
  - ▶ NEVER POST YOUR PROJECT CODE OR ASK FOR HELP DIRECTLY ONLINE!
- ▶ Please review our Academic Honesty Guidelines.

  `https://www.iit.edu/academic-affairs/academic-honesty-guidelines`

Outline

Computing Systems

- ▶ Any system that may compute.
    - ▶ Whose functionality can be improved and extended with additional hardware and software.
- ▶ Embedded systems, IoTs, Cloud, etc.
    - ▶ Everywhere.
- ▶ Very different characteristics
    - ▶ Sizes
    - ▶ Applications
    - ▶ Speed

History of Computing Hardware

- ▶ 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.
- ▶ Since 2000s: Multiprocessor SoC (MPSoC)
  - ▶ Power becomes a major limiting factor.
  - ▶ End of Moore's Law.
- ▶ New hardware technology?
  - ▶ Have we exploited all potential of existing device technology?
  - ▶ Will new device technology demand completely different design methodology?

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

- ▶ Rich/complex functionality within short time-to-market.
- ▶ 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.

Computing System as
Integration of Hardware and Software

- ▶ Hardware
  - ▶ Processors, memories, and standard interfaces
  - ▶ Programmable and reconfigurable hardware, e.g. FPGA
  - ▶ Application specific integrated circuits (ASICs)
- ▶ Operating System
  - ▶ Provide abstractions of hardware
  - ▶ Common OS supports, e.g. file system and multitasking
  - ▶ Other functionality, e.g. real-time guarantee
- ▶ Software
  - ▶ Run on processors
  - ▶ Reusable, could be ported from existing systems

Economics of Computing 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
- ▶ Implement as ASICs
    - ▶ High speed, low power
    - ▶ Low unit cost if produced in large quantities
    - ▶ Long time-to-market and high NRE cost
- ▶ Implement as something in the middle, e.g. GPU and FPGA
    - ▶ Achieve trade-offs between low latency, high throughput, and/or low power with short time-to-market and low NRE cost.

Ad-Hoc Computing System Design

- ▶ Choose a set of hardware from the market
  - ▶ Reduce NRE cost by NOT designing your own
- ▶ Come up with a HW/SW partitioning of functionality, implement it, and evaluate performances and various characteristics
  - ▶ Use CAD/EDA tools whenever possible to reduce design time and thus time-to-market
- ▶ Repeat the above step until all design constraints are met
- ▶ Concerns
  - ▶ Need both software and hardware developers
  - ▶ Very few design alternatives can be implemented and evaluated within short time-to-market
  - ▶ Delay between when the whole system becomes available and when parts become 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
  - ▶ Unified HW/SW design enables designers to explore more HW/SW partitionings with short time-to-market to meet stringent power, performance, and cost constraints.
- ▶ How to unify HW/SW designs?
  - ▶ What is the difference between software design and hardware design?

Summary

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