EE382V: Embedded System Design and Modeling

Lecture 1 – Introduction

Andreas Gerstlauer
Electrical and Computer Engineering
University of Texas at Austin
gerstl@ece.utexas.edu

Lecture 1: Outline

• Introduction
  • Embedded systems
  • System-level design

• Course information
  • Topics
  • Logistics
  • Projects

• Design methodology
  • System-level design flow
  • Models and methodologies
Embedded Systems

- **System-in-a-system**
  - Application-specific
  - Tightly constrained

- **Ubiquitous**
  - Far bigger market than general-purpose computing (PCs, servers)
    - 98% of all processors sold
      [Turley02, embedded.com]

- **Exponentially growing complexities**
  - Application demands & technological advances
    - Increasingly networked and programmable

---

Cyber-Physical Systems (CPS)

- **Not transformative**
  - Output = F(Input)
    - Procedural/batch processing

- **But reactive**
  - Continuous interaction with environment
    - Sense and act on the physical world

  - **Concurrency and time (order)**
General-Purpose Computing

- We are reaching physical limits of technology scaling
  - Dark silicon: power/utilization/… walls
  - Mobile devices and no leap in battery technology
    - Efficiency is the new performance

- Opportunity and need for specialization
  - Heterogeneous multi-core / Asynchronous CMP
    - Big/little architectures
    - GP-GPUs
  - Flexibility vs. specialization
    - Right mix of cores?
    - How to “program”?

Implementation Options

### Multi-Processor System-on-Chip (MPSoC)

- **System Memory**
- **Memory Controller**
- **CPU**
- **GPU**
- **Local RAM**

#### Frontside Bus
- **Bridge**
- **DSP**
- **DSP RAM**
- **Hardware Accelerator**

#### DSP Bus
- **Shared RAM**
- **Hardware Accelerator**
- **Video Front End**

#### Local Bus

---

### Design Challenges

- **Reactivity**
  - Concurrency & time
  - Safety, reliability, robustness

- **Heterogeneity**
  - Of components
    - Processors, memories, busses
  - Of design tasks
    - Architecture, mapping, scheduling

- **Complexity**
  - High degree of parallelism
  - High degree of design freedom
  - Multiple optimization objectives & design constraints
### Complexity Trends

- **LoC SW/Chip**: Gates/Chip, Gates/Day, LoC/Day
- **Moore's Law**: 2x/18 Months
- **System Design Gap**: Hardware Design Productivity 1.6x/18 Months
- **Software Productivity**: 2x/5 years
- **Additional SW required for HW**: 2x all 10 months

<table>
<thead>
<tr>
<th>Year</th>
<th>LoC/Chip (Gates/Chip)</th>
<th>Gates/Day</th>
<th>LoC/Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>1981</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1985</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1993</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1997</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


### Abstraction Levels

- **Move to higher levels of abstraction [ITRS07, itrs.net]**
  - Electronic system-level (ESL) design

*Source: R. Doemer, UC Irvine*
System-Level Design

- From specification
  - Functionality, behavior
    - Concurrency, order
    - Constraints

- To implementation
  - Architecture
    - Spatial and temporal order
    - Components and connectivity

➤ Design automation
  - Modeling
  - Exploration
  - Synthesis

———

System-On-Chip Environment (SCE)

- Specification
- System Design (Specify Explore Refine)
- System models
- Hardware Synthesis
- Software Synthesis
- Implementation Model

➤ Commercial derivative for Japanese Aerospace Exploration Agency

———

(c) 2014 A. Gerstlauer
Lecture 1: Outline

- **Introduction**
  - Embedded systems
  - System-level design

- **Course information**
  - Topics
  - Logistics
  - Projects

- **Design methodology**
  - System-level design flow
  - Models and methodologies
Course Topics

• **System-level design**
  • Methodologies and languages: SpecC, SystemC

• **Specification modeling**
  • Formal Models of Computation (MoC)
    – Parallel programming models, threads, dataflow, process networks
    – Hierarchical and concurrent finite state machine (FSM) models

• **System synthesis**
  • Design space exploration and optimization
    – Mapping and scheduling algorithms, exploration heuristics
    – System-level design tools: SCE

• **Architecture modeling**
  • Implementation and simulation (virtual prototyping) models
    – Host-compiled OS and processor models for computation
    – Transaction-level modeling of communication

➢ **Prerequisites**
  ➢ Software: C/C++ (algorithms and data structures)
  ➢ Hardware: VHDL/Verilog (digital design)
  ➢ Embedded systems and embedded software

Class Administration

• **Schedule**
  • Lectures: TTh 11:00am-12:30pm, WEL 3.402

• **Instructor**
  • Prof. Andreas Gerstlauer <gerstl@ece.utexas.edu>
    – Office hours: ACE 6.118, TW 2-3pm, or after class/by appt.

• **Teaching Assistant**
  • Parisa Razaghi <parisa.r@utexas.edu>
    – Office hours: MF 3-4pm, ENS 110

• **Information**
  • Web page: http://www.ece.utexas.edu/~gerstl/ee382v_s14
  • Announcements, assignments, grades: Blackboard
  • Questions, discussions: Blackboard
Textbooks (1)

- **Recommended**

- **Optional**
    - Models of computation (MoCs)
    - [http://leeseshia.org](http://leeseshia.org)

Textbooks (2)

- **Background material**
    - Practical, example-driven introduction using SpecC
    - Electronic copy of selected chapters on Blackboard

    - Reference for SystemC language and methodology
Policies

- **Grading**
  - Homeworks: 20%
  - Labs: 20%
  - Midterm: 25%
  - Project: 35%
  - No late submissions!

- **Academic dishonesty**
  - Homeworks are independent
    - Discuss questions and problems with others
    - Turn in own, independently developed solution
  - Labs and project are teamwork
    - Teams of up to 3 students
    - One report and presentation

Homeworks and Labs

- **Two to three homeworks and one exam**
  - Cover theoretical aspects of system design
    - Languages
    - Models
    - Exploration and optimization
  - Some practical implementation
    - Exposure to general language and modeling concepts

- **Two to three labs**
  - Real-world system design
    - Design example using SpecC and System-on-Chip Environment (SCE)
  - From specification to implementation
    - Modeling
    - Design space exploration
    - Hardware/software synthesis
Project

- **Two options**
  - Research project
    - System design research problem
    - Literature survey on system design research area
  - Implementation project
    - Non-trivial system design example/case study
    - Specification, exploration, implementation

- **Project timeline (tentative)**
  - Abstract: March 4 (email)
  - Proposal, literature survey: March 18
  - Presentations: last week of classes (Apr. 29 & May 1)
  - Report: finals week (May 8)
  - Final report and presentation in publishable quality

Some Possible Projects

- **Design projects**
  - (Embedded) system design example
    - Specify, model, simulate, explore, synthesize using SCE
      - Existing examples: MP3 Decoder, AC3 Decoder, Jpeg Encoder, GSM Vocoder
      - Backend synthesis down to ARM+FPGA prototyping board

- **Research projects**
  - Modeling
    - Specification modeling
      - Develop/modify a language or MoC: data parallel extensions of SpecC/SystemC
      - Translation between MoCs & languages: from Matlab/SDF/… to SpecC/SystemC
    - Architecture modeling
      - Component modeling: QEMU-SpecC/SystemC integration, bus modeling
      - Automatic model generation: generate bus TLMs from abstract protocol descriptions
      - OS modeling: OS-internal timing estimation and back-annotation
      - Performance estimation and modeling (timing, power, reliability, …): statistical simulation, parallel or hardware/software co-simulation of functional & performance models
      - Assertion-based verification in a TLM environment

- **Synthesis**
  - Pick an implementation problem and solve it
    - Decision making: machine learning for optimization (allocation, partitioning, scheduling), design space exploration for dataflow models/signal processing systems
    - OS scheduling for power, performance, reliability
    - Hardware or software synthesis for new OS/processors: targeting Linux in SCE
Successful Past Projects

- **Modeling**

- **Exploration and synthesis**

Lecture 1: Outline

- **Introduction**
  - Embedded systems
  - System-level design

- **Course information**
  - Topics
  - Logistics
  - Projects

- **Design methodology**
  - System-level design flow
  - Models and methodologies
Evolution of Design Flows

- **Capture & Simulate**
  - Specs
  - Algorithms
  - SW?
  - Design
  - Logic
  - Physical
  - Manufacturing
  - 1960's

- **Describe & Synthesize**
  - Specs
  - Algorithms
  - SW?
  - Design
  - Logic
  - Physical
  - Manufacturing
  - 1980's

- **Specify, Explore & Refine**
  - Executable Spec
  - Algorithms
  - Architecture
  - Network
  - SW/HW
  - Logic
  - Physical
  - Manufacturing
  - 2000's

- **System Gap**
- **Functionality**
- **Connectivity**
- **Protocols**
- **Performance**
- **Timing**

Source: D. Gajski, UC Irvine

Classical System Design Flow

1. **System requirement specification**
2. **System architecture design**
3. **Modeling**
4. **Hardware design**
5. **Software development**
6. **Integration & Verification**
7. **System**

- **Manual**
- **(semi)automatic**
Classical Design Cycle

- Specification
- Specification Fixes
- HW design
- HW design Fixes
- HW verification
- SW design
- SW design Fixes
- SW verification
- Integration & verification

Electronic System-Level (ESL) Design Flow

- System requirement specification
- High-level model
- System-level design
- Architecture model
- Hardware design
- Software development
- Integration & Verification
- System implementation

The design flow is illustrated with semi-automatic and manual techniques.
New ESL Design Cycle

<table>
<thead>
<tr>
<th>Task</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification (high-level &amp; arch. models)</td>
<td></td>
</tr>
<tr>
<td>Fixes in specification</td>
<td></td>
</tr>
<tr>
<td>HW design</td>
<td></td>
</tr>
<tr>
<td>Fixes in hardware</td>
<td></td>
</tr>
<tr>
<td>HW verification</td>
<td></td>
</tr>
<tr>
<td>SW design</td>
<td></td>
</tr>
<tr>
<td>Fixes in software</td>
<td></td>
</tr>
<tr>
<td>SW verification</td>
<td></td>
</tr>
<tr>
<td>Integration &amp; verification</td>
<td></td>
</tr>
</tbody>
</table>

Design Methodology

- **Formalization of a design flow**
  - Break into well-defined, repeatable steps
    - Set of models and transformations between them
    - Manual or automated

- **Models**
  - Design representations
    - Specification and documentation at interface between steps

- **Transformations**
  - Design decisions
    - Refine input model into an output model reflecting decisions
**Y-Chart**

- **Behavior (Function)**
  - Algorithm
- **Structure (Netlist)**
  - Logic
- **Physical (Layout)**
  - Circuit

**Models of Computation (MoCs)**
- Specification
- Algorithm
- Boolean logic ($a \lor b$)
- Transfer

**Models of Structure (MoSs)**
- Processor
- Network
- Processor

Source: D. Gajski, UC Irvine

---

**Processor Synthesis**

- **Software processor**
  - Compilation and linking
- **Hardware processor**
  - High-level synthesis

Source: D. Gajski, UC Irvine
System Synthesis

- **Structure**
  - Partitioning, mapping

- **Timing**
  - Scheduling

Bottom-Up Methodology

- **Each level generates library for the next higher level**
  - Circuit: Standard cells for logic level
  - Logic: RTL components for processor level
  - Processor: Processing and communication components for system level
  - System: System platforms for different applications
  - **Floorplanning and layout on each level**

Source: D. Gajski, UC Irvine
Top-down Methodology

- Functional description is converted into component netlist on each level
- Each component function is decomposed further on the next abstraction level
- Layout is given only for transistor components

Meet-in-the-Middle Methodology

- Gate netlist is hand-off
- Three levels of synthesis
  - System is synthesized with processor components
  - Processor components are synthesized with RTL library
  - RTL components are synthesized with standard cells
- Two levels of layout
  - System layout is performed with standard cells
  - Standard cells layout with transistors

Source: D. Gajski, UC Irvine
Platform-Based Design

- Meet-in-the-middle at the system level
  - System platform with standard components
  - System design reduced to mapping of specification onto pre-defined platform

---

Lecture 1: Summary

**✓ Introduction**
- Embedded systems
- System-level design

**✓ Course information**
- Topics
- Logistics
- Projects

**✓ Design methodology**
- System-level design flow
- Models and methodologies

Source: D. Gajski, UC Irvine