Outline

1. Introduction
2. Course Organization
3. General introduction, definition of the field
4. Embedded Systems Examples
5. Embedded Systems Design Methodologies

Examples of Embedded Systems
Examples of Embedded Systems (cont’d)

- Anti-lock brakes
- Auto-focus cameras
- Automatic teller machines
- Automatic toll systems
- Automatic transmission
- Avionic systems
- Battery chargers
- Camcorders
- Cell phones
- Cell-phone base stations
- Cordless phones
- Cruise control
- Curbside check-in systems
- Digital cameras
- Disk drives
- Electronic card readers
- Electronic instruments
- Electronic toys/games
- Factory control
- Fax machines
- Fingerprint identifiers
- Home security systems
- Life-support systems
- Medical testing systems
- Modems
- MPEG decoders
- Network cards
- Network switches/routers
- On-board navigation
- Pagers
- Photocopiers
- Point-of-sale systems
- Portable video games
- Printers
- Satellite phones
- Scanners
- Smart ovens/dishwashers
- Speech recognizers
- Stereo systems
- Teleconferencing systems
- Televisions
- Temperature controllers
- Theft tracking systems
- TV set-top boxes
- VCR’s, DVD players
- Video game consoles
- Video phones
- Washers and dryers

Source: Embedded Systems Design: A Unified Hardware/Software Introduction, (c) 2000 Vahid/Givargis

Course Organization

- Twelve lectures.
- Four lab assignments plus an introductory lab.
  - carried out in groups of two students
  - use special software and hardware (lab E:4115 and E:4118)
  - require preparation
  - deadline for lab reports (last lab assignment) Sunday, May 27, 2018, 23:59.

<table>
<thead>
<tr>
<th>Group</th>
<th>day</th>
<th>time</th>
<th>room</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Wed</td>
<td>10-13</td>
<td>E:4115</td>
</tr>
<tr>
<td>2</td>
<td>Wed</td>
<td>13-15</td>
<td>E:4118</td>
</tr>
</tbody>
</table>

Course Organization (cont’d)

- Seminars
  - 17/05, 8:15 and 10:15 in E:2116
  - 24/05, 8:15 and 10:15 in E:2116

- Obligatory examinations
  - 1 June 2018 at 8-13, Vic:2C, 2D
  - 21 August 2018 at 8-13, MA:8C

- Books:
Lectures
Preliminary Schedule

<table>
<thead>
<tr>
<th>Date</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>18-03-19</td>
<td>Introduction, motivation, etc.</td>
</tr>
<tr>
<td>18-03-23</td>
<td>Design methodology (HW/SW co-design, etc)</td>
</tr>
<tr>
<td>18-03-26</td>
<td>VHDL introduction</td>
</tr>
<tr>
<td>18-03-29</td>
<td>VHDL for synthesis</td>
</tr>
<tr>
<td>18-04-16</td>
<td>Computational models</td>
</tr>
<tr>
<td>18-04-20</td>
<td>Design representations</td>
</tr>
<tr>
<td>18-04-23</td>
<td>System partitioning</td>
</tr>
<tr>
<td>18-04-27</td>
<td>Allocation, assignment</td>
</tr>
<tr>
<td>18-05-04</td>
<td>and scheduling</td>
</tr>
<tr>
<td>18-05-07</td>
<td>Communication synthesis</td>
</tr>
<tr>
<td>18-05-14</td>
<td>Testability</td>
</tr>
<tr>
<td>18-05-17</td>
<td>Low-power design</td>
</tr>
</tbody>
</table>

Embedded Systems

“A device that includes a programmable computer but is not itself a general-purpose computer”

Execution deadlines, Power and energy consumption constraints, :

Embedded Systems (cont’d)

- Computing systems embedded within electronic devices
- Hard to define. Nearly any computing system other than a desktop computer
- Billions of units produced yearly, versus millions of desktop units
- Perhaps 50 per household and per automobile

Source: Embedded Systems Design: A Unified Hardware/Software Introduction, (c) 2000 Vahid/Givargis
Embedded Systems (cont’d)

- Non User-Programmable.
- Based on programmable components (e.g., Micro-controllers, DSP's...) but often contain application specific hardware (IC's, ASIC's).
- Reactive Real-Time Systems:
  - React to external environment,
  - Maintain permanent interaction,
  - Ideally never terminate,
  - Are subject to external timing constraints (real-time).

Characteristics Embedded Systems

- Sophisticated functionality.
- Real-time operation.
- Low manufacturing cost.
- Low power.
- Designed to tight deadlines by small teams.
- "Resource conscious" vs. "Unlimited resources" programming

A Single Processor Embedded System

SoC Embedded System

- Assembly of "prefabricated component" often purchased from external vendors ("IP")
  - "black box" hierarchy
- Design & Verification at the System level
  - rather than the logic level
  - Interface and communication
- Great Importance of Software

A Digital Camera Example

- A/D → CCD preprocessor → Pixel processor → D/A
- JPEG codec → Microcontroller → Multiplier/Accumulator
- DMA controller → Memory controller → ISA bus interface → UART → LCD ctrl

Real-time gas turbine testing system

- MI-2 helicopter engine
- "Minicomputer" 8kB RAM cassette tape
TELEX-I and TELEX-II systems

![TELEX-I and TELEX-II systems](image)

Kris Kuchcinski (LTH) Design of Embedded Systems March 19, 2018 16 / 37

WITAS project

- Autonomous system.
- Real-time system.
- Image processing.
- Mission planning.
- Incorporation of GIS systems.
- Interface with ground operator.
- ...

http://www.ida.liu.se/ext/witas

Kris Kuchcinski (LTH) Design of Embedded Systems March 19, 2018 17 / 37

Typical Hardware Components of DSP System

<table>
<thead>
<tr>
<th>Component class</th>
<th>Implements</th>
<th>Compiler</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP processor</td>
<td>Low data-rate DSP</td>
<td>(Retargetable) code generator</td>
<td>Assembly</td>
</tr>
<tr>
<td></td>
<td>Slow control loops</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td>Microcontroller</td>
<td>User interface</td>
<td>C compiler</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>Slow control loops</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware accelerator</td>
<td>High data-rate DSP</td>
<td>High level synth.</td>
<td>C, VHDL</td>
</tr>
<tr>
<td></td>
<td>RT level synth.</td>
<td></td>
<td>Verilog</td>
</tr>
<tr>
<td>Communication blocks and</td>
<td>Internal &amp; external</td>
<td>Memory mgmt.</td>
<td>Data-sheets</td>
</tr>
<tr>
<td>memory</td>
<td>communication</td>
<td>(Asynchronous)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Storage &amp; buffering</td>
<td>interface synth.</td>
<td></td>
</tr>
<tr>
<td>Others</td>
<td>Usually FSMD's</td>
<td>RT level synth.</td>
<td>VHDL</td>
</tr>
<tr>
<td></td>
<td>- clock generators</td>
<td>Asynchronous</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- DMA blocks</td>
<td>synth.</td>
<td></td>
</tr>
</tbody>
</table>

Kris Kuchcinski (LTH) Design of Embedded Systems March 19, 2018 18 / 37
Importance of Embedded System Design Methodologies

- Hardware complexity.
- Heterogeneous systems containing hardware (both digital and analog) and software.
- Heterogeneous components (CPU's, DSP's, ASIC's, buses, point-to-point links, etc.).
- Heterogeneous requirements - performance, cost, power consumption, etc.
- System-on-chip.
- Shorter design cycles required by time-to-market constraints.
- ...

Source: Bryan Preas, Xerox PARC, 35th DAC

Design Complexity and Designer Productivity Gap

![Graph showing design complexity and designer productivity gap.](image)

Source: Bryan Preas, Xerox PARC, 35th DAC

Flexibility and Energy Efficiency

![Graph showing power efficiency (GOPS/Watt).](image)

Source: T.Claasen et al. (ISSCC99)
Flexibility and Energy Efficiency

Design Domains and Abstraction Levels
Design Domains

- behavioral representations - describe only circuit's function, for example

  ```
  if clock=high then counter:= counter+1
  ```

- structural representations - describe the components and their interconnections, for example

  ![NAND gate diagram]

- physical representations - it can describe either a geometrical layout or a topological constraint.

Software vs. Hardware Design

short summary

- Software
  - flexibility,
  - reconfigurability, easy update, etc.,
  - complex functionality,
  - cost,
  - ...

- Hardware
  - speed,
  - power consumption,
  - cost in large volumes,
  - ...

Design of Embedded Systems

- Need to be done using high-level specification, programming and hardware description languages - not assembly languages and gate/transistor level design.

- Requires efficient design space exploration and synthesis/compilation tools.

- Different design requirements has to be taken into account, e.g., cost, performance, testability, quality of service, power consumption.

- Multi-language design framework.
Importance of High-Level Design Methods

System Verification Processing Speeds

<table>
<thead>
<tr>
<th>System implementation</th>
<th>Processing time (s/frame)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behavioral model</td>
<td>1 200 (20 min/frame)</td>
</tr>
<tr>
<td>RTL model</td>
<td>144 000 (1.6 days/frame)</td>
</tr>
<tr>
<td>Gate model</td>
<td>228 000 (2.6 days/frame)</td>
</tr>
<tr>
<td>Gate model on hardware accelerator</td>
<td>1 200</td>
</tr>
<tr>
<td>Rapid prototype</td>
<td>0.5</td>
</tr>
<tr>
<td>Target hardware</td>
<td>0.05</td>
</tr>
</tbody>
</table>


General Design Flow

Specify-Explore-Refine

Specification and Programming

- Specification languages, such as UML, SDL.
- Programming languages, such as C, C++, Java, Esterel, assembly languages.
- Hardware description languages, such as VHDL, Verilog, SystemC.

Example: combining SystemC and C++ gives a unified simulation environment for hardware and software.
**Hardware Description Languages**

- Cover several levels of design abstraction as well as behavioral and structural description domain.
- Contain typical features of programming languages, such as data types and program statements.
- Special features:
  - time concept,
  - structure description,
  - parallelism.
- VHDL (IEEE standard), Verilog, SystemC.

---

**Design Representations (Computational Model)**

- Used to represent/model digital systems under design.
- Generated by a compiler from system specification or coded directly in the model.
- Represent the semantics, structure and timing of the system.
- Usually based on some kind of annotated graph representation.
- Used internally by design automation systems or by the modeler/designer.

---

**Design – Synthesis**

- Software is translated into target code for a processor.
- Real-time operating system might be used.
- Hardware synthesis – translation of a behavioral representation of a design into a structural one.
- Communication synthesis – generates hardware and software which interconnects system components.
Hardware Synthesis Levels

- **System level** – accepts as an input specification in a form of communicating concurrent processes. The synthesis task is to generate the general system structure defined by processors, ASIC’s, buses, etc.
- **High level** – the input specification is given as a behavioral level description of an algorithm describing the functionality of a design and generates implementation at RT-level. Basic synthesis steps are scheduling, allocation and binding.
- **Logic level** – it can be divided into combinational and sequential logic synthesis.
- **Physical design** – it accepts a gate-level netlist and produces final implementation of the design in a given technology.

Hardware/Software Co-design enabling technologies

Discrete Cosine Transform
Partial Design Space
Design Space Exploration

Time-to-market constraint

Summary