Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems

Jian-Jia Chen
(slides are based on Peter Marwedel)
TU Dortmund,
Informatik 12

2019年 10 月 09 日
Common characteristics
Dependability

- CPS/ES must be **dependable**, 
  - **Reliability** $R(t) = \text{probability of system working correctly provided that it was working at } t=0$
  - **Maintainability** $M(d) = \text{probability of system working correctly } d \text{ time units after error occurred.}$
  - **Availability** $A(t): \text{probability of system working at time } t$
  - **Safety**: no harm to be caused
  - **Security**: confidential and authentic communication

Even perfectly designed systems can fail if the assumptions about the workload and possible errors turn out to be wrong.

Making the system dependable must not be an after-thought, it must be considered from the very beginning.
Efficiency

- CPS & ES must be **efficient**
  - Code-size efficient (especially for systems on a chip)
  - Run-time efficient
  - Weight efficient
  - Cost efficient
  - Energy efficient
Importance of Energy Efficiency

Efficient software design needed, otherwise, the price for software flexibility cannot be paid.
CPS & ES Hardware

CPS & ES hardware is frequently used in a loop ("hardware in a loop"): 

A/D converter → information processing → display 

sample-and-hold → D/A converter → actuators 

sensors → (physical) environment → Cyber-physical systems (!)
Real-time constraints

- Embedded systems must meet **real-time constraints**
  - A guaranteed system response has to be explained without statistical arguments [Kopetz, 1997].
  - A real-time system must react to stimuli from the controlled object (or the operator) within the time interval **dictated** by the environment.
    - “A real-time constraint is called hard, if not meeting that constraint could result in a catastrophe“ [Kopetz, 1997].
    - All other time-constraints can be roughly called **soft**.
Typical Misconceptions

“Real time” is performance engineering/tuning.
  - **Timeliness** is more important in real-time systems.

Real-time computing is equivalent to fast computing.
  - Real-time computing means predictable and reliable computing.

Advances in supercomputing hardware will take care of real-time requirements.
  - Buying a ”faster” processor may result in timeliness violation.
CPS, ES and Real-Time Systems synonymous?

- For some embedded systems, real-time behavior is less important (Telecommunication)
- For CPS, real-time behavior is essential, hence $\text{RTS} \cong \text{CPS}$
- CPS models also include a model of the physical system
Typically, ES/CPS are reactive systems: “A reactive system is (one which is) in continual interaction with its environment and executes at a pace determined by that environment“ [Bergé, 1995]

Behavior depends on input **and current state**.

*automata model appropriate, model of computable functions inappropriate.*

**Hybrid systems** (analog + digital parts).
Dynamics

Frequent change of environment
Characteristics lead to corresponding challenges

- Dependability
- Efficiency
  - In particular: Energy efficiency
- Hardware properties, physical environment
- Meeting real time requirements
  - ....
Space of Design

Requirements
(Performance, scale, etc.)

Interactive Video
High-quality Audio
Network File Access
Remote Login

Interesting Region for Real-Time Systems

Sufficient but scarce resources

Insufficient resources

Hardware resources in year X

1980 1990 2000

Abundant Resources
Challenges for implementation in hardware

- Early embedded systems frequently implemented in hardware (boards)

- Mask cost for specialized application specific integrated circuits (ASICs) becomes very expensive (M$ range, technology-dependent)

- Lack of flexibility (changing standards).

- Trend towards implementation in software (or possibly FPGAs, see chapter 3)
Challenges for implementation in software

If CPS/ES will be implemented mostly in software, then why don’t we just use what software engineers have come up with?
It is not sufficient to consider CPS/ES as a special case of SW engineering

Knowledge from many areas must be available,
Walls between disciplines must be torn down

medicine, statistics,
ME, biology
Challenges for CPS/ES Software

- Dynamic environments
- Capture the required behaviour!
- Validate specifications
- Efficient translation of specifications into implementations!
- How can we check that we meet real-time constraints?
- How do we validate embedded real-time software? (large volumes of data, testing may be safety-critical)
Software complexity is a challenge

Software in a TV set

- Source 1*: Exponential increase in software complexity

<table>
<thead>
<tr>
<th>Year</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1965</td>
<td>0</td>
</tr>
<tr>
<td>1979</td>
<td>1 kB</td>
</tr>
<tr>
<td>1990</td>
<td>64 kB</td>
</tr>
<tr>
<td>2000</td>
<td>2 MB</td>
</tr>
</tbody>
</table>

- Source 2°: 10x per 6-7 years

<table>
<thead>
<tr>
<th>Year</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1986</td>
<td>10 KB</td>
</tr>
<tr>
<td>1992</td>
<td>100 kB</td>
</tr>
<tr>
<td>1998</td>
<td>1 MB</td>
</tr>
<tr>
<td>2008</td>
<td>15 MB</td>
</tr>
</tbody>
</table>


... > 70% of the development cost for complex systems such as automotive electronics and communication systems are due to software development [A. Sangiovanni-Vincentelli, 1999]
Hypothetical design flow

Generic loop: tool chains differ in the number and type of iterations

- Specification
- ES-hardware
- System software (RTOS, middleware, …)
- Design repository
- Application mapping
- Optimization
- Evaluation & Validation (energy, cost, performance, …)
- Design
- Test *

* Could be integrated into loop
Summary

- Common characteristics
- Challenges (resulting from common characteristics)
- Design Flows