Embedded systems are everywhere... ... and control vital functions in our daily life!

Designers have large responsibility!

Between 1985-87 several deaths and serious injuries of cancer patients were due to overdoses of radiation resulting from a race condition between concurrent tasks in the Therac-25 software (1985-87).
Characteristics of Embedded Systems

An embedded system
- is usually designed for one single task. Its functionality will never change.
- is often a mass product. Design cost is critical.
- is often safety-critical systems and has to fulfill hard real-time requirements.
- is often a hand-held device. Power-efficiency is critical.
- is often a consumer products. Time-to-market is critical.

To be fast is not enough!
The system has to react to a heterogeneous environment at the right time instance, otherwise there can be fatal consequences.

Design process for embedded systems is very different from general purpose programming!
- Embedded systems can be highly optimized
- All unneeded features are a disadvantage (cost, power)
- Design process must
  - be cost-efficient
  - ensure the correct functionality and timing of the implementation
  - be fast to ensure a short time-to-market
- Embedded systems have a complex architecture consisting of many heterogeneous components
  - Analog and digital hardware, general-purpose and specialized processors, real-time operating system and other software components

A disciplined design methodology is needed to design future embedded systems!
Full-custom design methodology

- A top-down full custom design methodology will use all levels of abstraction during the design process.
- Maximal flexibility, all details can be fine-tuned.
- Maximal performance can theoretically be achieved.

Design-time and thus time-to-market can be very large!

Platform-Based Design

Trade-Off: Design Space vs. Time-to-Market

- A platform limits the design choices.
- The designer needs only to analyze the alternatives that are implemented by the platform.
- The platform can itself be configured to a certain degree.
An industrial platform: OMAP

- The Open Multimedia Application Platform (OMAP) is developed by Texas Instruments. Many mobile phones are using this platform.

- The OMAP 4 platform has been designed to drive smart phones and mobile internet devices (MIDs).

OMAP Hardware Platform Instance

OMAP is not only a hardware platform, but provides several layers of software, which together comprise the OMAP software platform.

Designer can work at a high-level of abstraction!
Platform Benefits

In addition to a shorter time-to-market the platform concept gives more benefits:

- The platform can be highly optimized since the development costs are shared by several designs.
  - Library of software or IP-blocks
  - Tool support in form of compilers, verification tools, simulators
  - Full custom design of critical platform components
- Other platforms can be developed on top of a platform.
  - Design entry can be moved to higher levels of abstraction
  - Development of synthesis tools to automatically refine a design from an abstract level to a more detailed level

Mapping of Function to Platform

In order to map a function onto a platform

- functional requirements needs to be implemented
- non-functional requirements need to be fulfilled (cost, power, timing, . . .)

Accurate estimates must be provided by the platform!
Current Industrial Design Practice

- It is very difficult to accurately estimate the performance of an embedded system.
- There is a huge difference between average and worst case execution time

As a consequence industry bases new designs rather on old experiences than on performance analysis and introduces sufficient safety margins in form of more powerful components and extra communication bandwidth

Surely, there must be a better way to design systems...

A Dream: Correct-by-Construction Refinement

- New platforms can be based on existing platforms
- Platform must export accurate performance figures
- Mapping from platform to platform must either be
  - based on previously verified transformations
  - accompanied with a verification method that can prove correctness
Towards a predictable platform

If the base platform lacks predictability, it is very difficult to build a predictable platform on top of it

- **Digital hardware** is very predictable due to a simple mechanism: synchronization by a clock signal
- **Advanced processors** are difficult to predict due to mechanisms, that are aimed for improvement of average performance.
  - Cache memory: The access time difference between a cache hit and miss can be easily a factor of 10.
- **Multicore architectures** share main memory and communication channel
  - Execution time of a program running on one processor depends on the other processors, even if there is no communication in between them.

Focus on predictable architectures is needed!

At present the focus is on average performance. But for embedded systems predictable is often much more important. Some suggestions:

- Use tightly-coupled on-chip memories instead of caches.
  - Possible to predict at design time, if data is on-chip or off-chip
- Give guaranteed access to buses or communication links (round-robin)
  - Possible to predict, when processor can access the bus
- Use languages that are based on a simpler communication scheme for concurrent programs
  - Synchronous languages (Esterel, Lustre, Signal, ...) use an implicit synchronous clock and have been shown very successful for safety-critical applications

Formal Methods

Formal methods can help a lot, but require a simple and formal model of the underlying platform. Then they can be employed at all levels of abstraction.
The SYSMODEL project

- The Artemis project 'SYSMODEL' (System Level Modeling Environment for SMEs) addresses the design process of heterogeneous embedded systems.
- Industrial Partners: Technoconsult, SIB Development (Denmark), Sting Networks, Catena, Solidux (Sweden), DA-Design, Finnelpro (Finland), Novelda (Norway)
- Academic Partners: Technical University of Denmark, Royal Institute of Technology (Sweden), Tampere University of Technology (Finland)
- The project has started in January 2009 and has a duration of three years.

**SYSMODEL: Objectives**

The SYSMODEL project follows the platform-based design approach. There are the following main objectives:

- Development of modeling and simulation framework for analysis and design of embedded systems and systems-on-chip.
  - System Functionality Framework
  - Platform Architecture Framework
  - Design Space Exploration: Analysis, Mapping and Testability
  - Verification of models
- The results of the project shall increase productivity in industry.
Models and Languages

- To provide a base for formal methods, the system functionality framework is based on the **ForSyDe** modeling framework.
- To allow for industrial exploitation **SystemC** is used as the main modeling language both for the system functionality framework and the platform architecture framework.
  - Already today several dialects for different models of computation exist. However, they often lack a formal semantics.
  - Other languages, such as VHDL for hardware models or C/C++ for algorithms shall be integrated in the framework.
- An industrial ‘refinement-by-replacement’ approach shall be supported. Parts of the system functionality models shall be replaced by executable platform models and then co-simulated with the system functionality models.

**ForSyDe**

ForSyDe (Formal System Design) is a design methodology for systems-on-chip, which allows to model heterogeneous systems.

- ForSyDe is implemented as domain specific language in Haskell.
- Several libraries for different models of computation exist and can be simulated as integrated model.
- ForSyDe processes are formally defined.
- ForSyDe supports modeling at different levels of abstraction.
- There exists a back-end for hardware design and synthesis (VHDL).
- High-level and synthesizable models can be co-simulated giving access to powerful test benches.
ForSyDe System Model

- A system is modeled as hierarchical concurrent process model
- Processes of different models of computation communicate via domain interfaces
  - Supported MoCs: Synchronous, Untimed (Synchronous Data Flow), Continuous Time

Systems containing analog, digital and software parts can be modeled at different levels of abstraction!

ForSyDe Process

A process takes $m$ input signals as argument and produces $n$ output signals. ForSyDe processes are deterministic.

- A process is always designed by means of a **process constructor**
- The process constructor defines the communication interface of the process
- The process constructor takes side-effect-free functions and variables as arguments and returns a process

**Process Constructor** + **Functions** + **Variables** = **Process**

```
mooreSY

f

+ g

v

= mooreSY

f

+ g

v
```
Process Constructor - Benefits

The concept of process constructor

- separates communication from computation
- process constructor: communication and interface
- function: computation

- forces the designer to develop a structured formal model that allows for formal analysis
  - transformational refinement
  - implementation mapping
  - formal verification

ForSyDe: Heterogeneous System Model

The example has been developed as part of the European FP6 project ANDRES.
ForSyDe: Simulating a Heterogeneous System

1. Synchronous Input Signal
   \( \text{in} = \{0, 1, 2, 3, 4, 5\} \)

4. Gaussian Noise

7. Synchronous Output Signal
   \( \text{out} = \{0, 1, 10, 3, 4, 5\} \)

3. Transceiver Output

5. Noisy Transceiver Input

SystemC

- SystemC is a class library built on top of the C++ language.
- SystemC adds concurrency and notion of time to C++ to allow to model systems.
- The semantics of SystemC is very different from C++.
- Systems in SystemC are modeled network of communicating processes. Computational processes are encapsulated in modules, while the communication among them is performed through channels.

![SystemC Diagram](image)
SYSMODEL: ForSyDe is implemented in SystemC

SYSMODEL: Modeling rules will be developed in order to write SystemC models to comply to the ForSyDe semantics!

The System Functionality Model is modeled in SystemC. Other languages can be imported by means of SystemC-wrappers. The whole model can be co-simulated.
A SystemC block has been refined and replaced by C-code that runs on an instruction set simulator, which belongs to the platform architecture framework. The whole model can be co-simulated.

### The ANDRES project

ForSyDe has been used as formal framework in the FP6 ANDRES project: (Analysis and Design of run-time Reconfigurable, heterogeneous Systems). The ANDRES project used SystemC as design language. As part of the project

- a specification methodology
- a performance analysis method

for reconfigurable systems has been developed.

#### Facts on ANDRES

- **Industrial Partners:** Thales Communications (France), DS2 (Spain)
- **Academic Partners:** OFFIS (Germany), Royal Institute of Technology (Sweden), Technical University Vienna (Austria), University of Cantabria (Spain)
- The project started in June 2006 and will finish in September 2009.
Conclusion

- Platform-based design is a promising approach to tackle the increasing complexity of embedded system design.
- The design of a platform is a huge investment and does only pay off, if the platform is reused.
- Accurate performance characteristics must be available to allow a mapping of the functionality onto the platform.
- In order to move the design process to a higher level of abstraction, well defined models and interfaces between different abstraction layers, each constituting a platform, are needed.
- Projects like SYSMODEL are required to move system design to a higher level of abstraction.

Further Information

- More information on ForSyDe: http://www.ict.kth.se/forsyde/
- More information on the SYSMODEL project: http://www.sysmodel.eu/
- More information on the ANDRES project: http://andres.offis.de/