TTool/DIPLODOCUS: a UML Model-Driven Environment for Embedded Systems-on-Chip Design

Andrea Enrici
Ludovic Apvrille
Renaud Pacalet

MODELS 2014 Demonstration session
Research context: embedded Systems on Chip (SoCs)

- **Embedded system** = information processing systems embedded into a larger product, normally not directly perceivable by users
Research context: embedded Systems on Chip (SoCs)

- **Embedded system** = information processing systems embedded into a larger product, normally not directly perceivable by users.

- **System on Chip** = software and hardware components working together to perform a predefined set of functions.
Research context: embedded Systems on Chip (SoCs)

- **Embedded system** = information processing systems embedded into a larger product, normally not directly perceivable by users

- **System on Chip** = software and hardware components working together to perform a predefined set of functions

- **We focus on:**
  - data-dominated embedded systems (e.g., signal, video processing)
Our Design Challenge

Hardware/software partitioning: decide if a functionality should be implemented in hardware or in software or both

- Design Space Exploration is the solution!
Design Space Exploration (DSE)

Design Space Exploration = analyzing different implementations, that are functionally equivalent, in order to find an optimal solution according to given performance criteria (e.g., costs, area, power consumption)
Motivation

- Embedded hardware has undergone a tremendous change:
  - from single processor architectures ...
  - ... to on-chip cloud computers with tens or hundreds of cores
Motivation

- Embedded hardware has undergone a tremendous change:
  - from single processor architectures ...
  - ... to on-chip cloud computers with tens or hundreds of cores

- so has done embedded software:
  - from simple routines within control loops ...
  - ... to applications with video conferencing and voice recognition
Motivation

- Embedded hardware has undergone a tremendous change:
  - from single processor architectures ...
  - ... to on-chip cloud computers with tens or hundreds of cores

- so has done embedded software:
  - from simple routines within control loops ...
  - ... to applications with video conferencing and voice recognition

- It is now impossible to design new products from scratch!
A toolkit named TTool (say "tea-tool")
- UML/SysML diagrams to model an embedded system’s functionality, communication services and architecture
TTool/DIPLODOCUS in a nutshell

- UML/SysML diagrams to model an embedded system’s functionality, communication services and architecture

- Simulate and verify formally the system’s models at the push of a button
TTool/DIPLODOCUS in a nutshell

- UML/SysML diagrams to model an embedded system’s functionality, communication services and architecture
- Simulate and verify formally the system’s models at the push of a button
- No need to be an expert in modeling, simulation or formal verification
TTool/DIPLODOCUS in a nutshell

- **UML/SysML diagrams to model an embedded system**’s functionality, communication services and architecture
- **Simulate and verify formally** the system’s models **at the push of a button**
- **No need to be an expert** in modeling, simulation or formal verification
- **Let’s model** a signal processing algorithm running on an embedded System-on-Chip!
High-level models ... what for?

- Increase simulation speed
  - the more models contain details, the longer are the simulation and verification
High-level models ... what for?

- Increase simulation speed
  - the more models contain details, the longer are the simulation and verification

- Reduce design and testing effort
  - the earlier the bugs are found, the less it costs to fix them
High-level models ... what for?

- **Increase simulation speed**
  - the more models contain details, the longer are the simulation and verification

- **Reduce design and testing effort**
  - the earlier the bugs are found, the less it costs to fix them

- **Increase portability**
  - save time and money by re-adapting models for next projects
The $\Psi$-chart design methodology
The system’s functionality is modeled independently of the architecture, it can be simulated and verified before mapping.
The available resources are modeled as a set of generic hw components: buses, memories, CPUs, etc.
The communication services offered by the architecture are modeled and can be simulated before mapping.
The user **selects the architecture resources** that implement the functionalities of the application and communication models.

**The $\Psi$-chart design methodology**

- **Formal Verification**
- **Functional Simulation**

- **Application modeling**
- **Communication modeling**
- **Architecture modeling**

**Design Space Exploration**

---

8/20

Telecom ParisTech/COMELEC

MODELS 2014 Demonstration session
The design space is explored after mapping via functional simulation and formal verification (model-checking).

The $\Psi$-chart design methodology

- Formal Verification
- Functional Simulation

- Application modeling
- Communication modeling
- Architecture modeling

- Mapping
- Exploration

8/20 Telecom ParisTech/COMELEC MODELS 2014 Demonstration session
Let’s model with TTool/DIPLODOCUS!

Let’s model a signal processing algorithm that runs on an embedded System-on-Chip!
Let’s model with TTool/DIPLODOCUS!

Diagrams are drawn by dragging and dropping the modeling assets from the toolbars.
Let's model first the processing operations interconnected by data dependencies (blue ports)
Then let's **model a control operation** and the **control dependencies** among the other operations (violet ports).
Applications are modeled as communicating tasks which are then mapped on hardware components.
events are used to synchronize tasks' execution
requests serve as means to spawn other tasks
Application modeling

**channels** model data exchanges between tasks
The **behavior of each task** is described by a **state machine**.

![State Machine Diagram](image)

*an event is sent*
The behavior of each task is described by a state machine.
The behavior of each task is described by a state machine.

Write a sample to a channel.
The behavior of each task is described by a state machine.
The behavior of each task is described by a state machine. Select between multiple execution paths.
The behavior of each task is described by a state machine.
The **behavior of each task** is described by a **state machine**.

**wait for an event**
The behavior of each task is described by a state machine.

Retrieve a sample from a channel.
Application modeling

The TTool/DIPLODOCUS abstraction principles:

1) only the amount of data exchanged between tasks is modeled. Data-dependent decisions are expressed by non-deterministic and static operators.
2) algorithms are described using abstract cost operators. The complexity of computations is taken into account without having to actually execute them.
An **architecture** model is a set of **generic components**. A CPU executes control and data processing tasks.
An **architecture** model is a set of **generic components**

A **memory** stores data and control information.
An architecture model is a set of generic components. A DMA transfers data and control information.
An **architecture** model is a set of **generic components**. A **bus** enables other hw components to communicate.
An **architecture** model is a set of **generic components**. A **bridge** enables two buses to communicate.
Each component is characterized by a set of performance parameters.
**Communication modeling**

**Sequence Diagrams** model **message exchanges** among **generic resources** that will be mapped onto the architecture.

**Dedicated models (Communication Patterns)** capture the **communication services** offered by embedded systems' interconnects.

![Sequence Diagram](image_url)
Activity Diagrams serve to compose Sequence Diagrams with control flows into a higher-level description.
Tasks are mapped on CPUs and hardware accelerators.

Channels are mapped onto buses, bridges and memories.
Communication Patterns are mapped via dedicated artifacts.
Simulation is started at the push of a button.
The simulator GUI allows to easily explore and debug a design while animating the diagrams.
The **formal static analysis** allows to prove certain properties. For example, check the reachability and liveness of a given operator.
Formal Verification

Model checking properties are verified with UPPAAL and the formal semantics of LOTOS, at the push of a button.
Debugging facilities

- Debugging and design exploration facilities are available from simulation and verification:
  - Gantt charts
  - Execution waveforms
  - Reachability graphs
An environment for the design of Systems-on-Chip:

- easy and rapid to deploy, open source
Summary

An environment for the design of Systems-on-Chip:

▶ easy and rapid to deploy, open source

▶ used by industrial and academic partners ...
  ▶ Texas Instruments
  ▶ Freescale
  ▶ ISAE (Toulouse, France)
Summary

An environment for the design of Systems-on-Chip:

- easy and rapid to deploy, open source
- used by industrial and academic partners ... 
  - Texas Instruments
  - Freescale
  - ISAE (Toulouse, France)
- ... and projects
  - EVITA (automotive embedded systems)
  - Embb (Software-Defined Radio)
Future works and research directions

Integrate *model transformations*:

- **into a code generation environment** to produce the application control code
Integrate **model transformations:**

- **into a code generation environment** to produce the application control code

- **into a run-time environment** for application scheduling and memory management
Do you want to know more?

Visit TTool’s website:

- http://ttool.telecom-paristech.fr/

Contact us:

- andrea.enrici@telecom-paristech.fr
- ludovic.apvrille@telecom-paristech.fr
- renaud.pacalet@telecom-paristech.fr