# Simulating Microsystems in the Context of an Automotive Drive Application

Georg Pelz*, Christian Decker*, Dieter Metzner*, Lars Voßkämper+, Dirk Dammers+
* Infineon Technologies AG, Automotive Power, P.O.-Box 800 949, D-81609 Munich, Germany
+ Dolphin Integration, Bismarckstr. 142a, D-47057 Duisburg, Germany

## Abstract

The opportunities to integrate more and more functionality into a (micro)system-in-a-package (SiP) heavily call for advanced methodologies in modeling and simulation for complete systems. For a microelectronics company like Infineon Technologies, the top-level simulation of our products, i.e. the above microsystems, is not a nice-to-have feature – it is a must! Iterating on the fabrication runs for design debugging simply is not feasible. Extending these simulations to application level can be accomplished and this paper shows how. An automotive drive application will serve as a demonstrator for that.

## 1 Introduction

Let's first take a look at the microsystem: it may comprise microcontrollers and power electronics. Behavioral modeling provides for sufficient simulation speed, while not sacrificing precision. For all analog and mixed-signal parts of the system, it is carried out in VHDL-AMS. This holds for electronics as well as for thermal processes, which always have to be taken into account when power electronics comes into play. The microcontroller is modeled in SystemC, which provides for an extremely fast, but still cycle-correct treatment. It takes real-life programs on assembly level and executes them as it is simulated.

The application side is the electric motor with gear, load and a hall sensor for position feedback. Again, VHDL-AMS is adopted for behavioral modeling. This allows for the simulation of the complete application, comprising the microsystem as well as the surrounding electric motor and position using a commercially available simulator. This complete system typically simulates at a performance of 20 CPU-minutes per second real-time on a recent Linux workstation, which allows for thorough system verification, software development and even concept assessment (in case the abstract behavioral models can be provided in time, which should be no problem for a design with substantial legacy content).

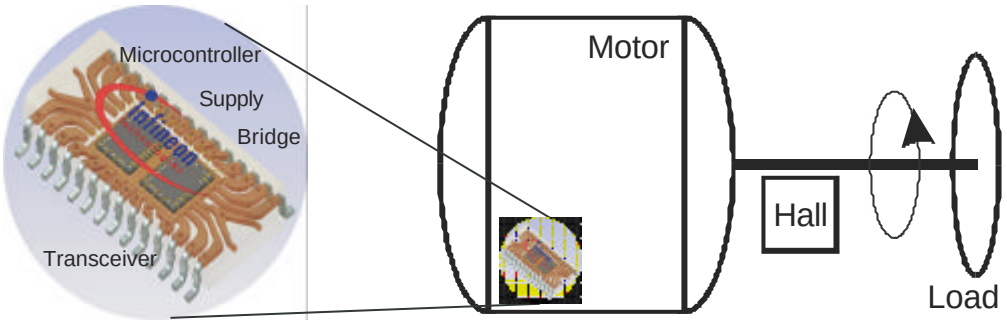


Fig. 1. Example drive system.

# 2 System Modeling

## 2.1 Introduction

When talking about system modeling and simulation, often block-oriented simulators are used like MATLAB/SIMULINK and similar tools. As the focus of a microelectronics company like Infineon is on electronic circuits, this kind of abstraction is too high, as even simple circuitry cannot reasonable be described in this way. Nonetheless, MATLAB/SIMULINK component models may be linked into the kind of simulations, which are proposed in this paper.

In general, this leaves us with two approaches for the modeling of systems spreading over more than one physical domain: simulator coupling and model transformation:

- **Simulator coupling**
  In our case, the simulator coupling approach would have to bind together simulators for analog electronics, digital electronics, software, mechanics, thermal issues and magnetics. Even if it was possible to solve all the interface issues, we would end up with a big hunk of software, which would be extremely difficult to manage and configure. Moreover, the combination of several analog solvers would easily lead to very nasty convergence problems.

- **Model transformation**
  The idea behind model transformation is to provide all models in one common form, so that they are simulatable by one solver.

We chose the second approach in which all domains are covered by one solver. Actually, we have got two solvers – one for analog and one for digital – which is due to the nature of the underlying descriptions. This co-simulation of the analog and digital domain has been a topic in microelectronics for twenty years now. The tools on the market are mature and reliable. The language to formulate the analog models is VHDL-AMS. For the digital part we chose SystemC.

## 2.2 Power Electronics

The typical components of the power electronics are the following:

- Power switches, highside and/or lowside
- Motor bridges and bridge drivers
- Voltage and current regulators
- Transceivers

The basic behavior of these kind of components is determined by the behavior of the respective power MOSFETs and their gate drivers, several protection features (e.g. over-temperature, over-current, etc.), several diagnosis features and the environment, comprising of the loads (e.g. electric motors, bus lines etc.) and the attached microcontroller. With the reduction of the number of chips in an electronic control unit (ECU) and the resulting need for higher integration, many of the above features are integrated into one chip or at least into one package.

The modeling of power electronics – as any other kind of modeling – always requires a reasonable balance between precision and model complexity. Providing for more precision, i.e. adding more related physical effects into for instance the power MOSFET, directly leads to a longer simulation time. The art in it is to stay as precise as necessary and as abstract as possible. In our case, this means that the power MOSFET is to be modeled in higher precision and on an abstraction level, which is comparable with the device models for the circuit simulation. On the other hand, the protection features or the diagnosis features and even the gate drivers are modeled in a relative abstract fashion, as their impact on the outside world is limited or their contribution can be described in an event-oriented or even digital manner.

When talking about power electronics, we also have to talk about self-heating and in general thermal issues. Even more, the circular dependency between the electrical and the thermal behavior of a power transistor has to be taken into account. This is due to the drain current which causes the self-heating and the resulting junction temperature which influences the drain current. Fig. 2 shows a DMOS model with attached network of thermal capacitances and thermal resistances.

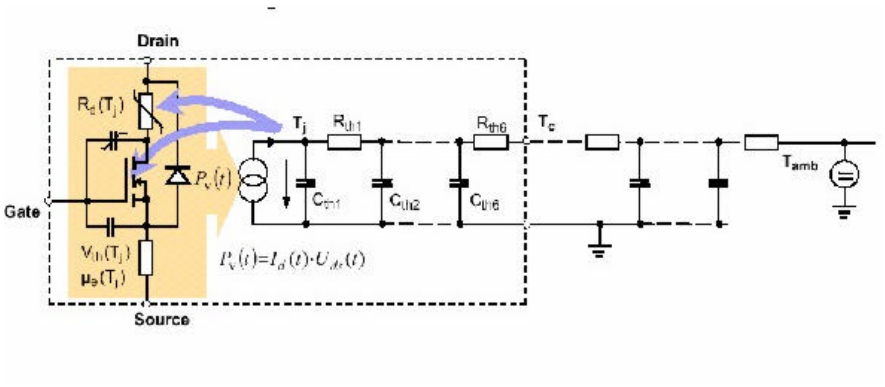


Fig. 2. Electro-thermal DMOS model.

### 2.3 Embedded Microcontroller and Software

The microcontroller's task is to run the embedded software and to provide for the interfacing to the outside world through its peripherals. Again, the issue of precision and abstraction comes up. The first question to answer for a microcontroller model is that of the underlying time base. In all practical cases up to now, it turned out, that microcontroller's clock cycle is precise enough for a reasonable system simulations. On the other hand, it allows for a simulation speed that its one or more orders of magnitude above the gate level simulations, which would form the next level of precision.

For the microcontroller in our case – a state-of-the-art 8-Bit controller – we provided for a SystemC model, which was attached to the VHDL-AMS models for the rest of the system. This was accomplished by using the standard SystemC interface as provided by the vendor of the commercial VHDL-AMS simulator. The modeling strategy again follows the rationale of as precise as necessary and as abstract as possible. As the underlying idea is not to enhance the respective microcontroller, it is sufficient to provide for a model that is only valid at its external interfaces. Basically this model is a command interpreter which reads the current assembler statement and runs the respective actions on its memory map. The models for the peripherals are attached to this command interpreter and bind it to the outer world. The embedded software itself is not modeled. At initialization time, it is loaded into the respective memory model. Then it is executed as the microcontroller model is simulated.

As the SystemC model just provides for digital behavior, an analog shell was attached to it, to accomplish for correct electrical behavior. This turned out to be extremely helpful, as mismatches can be detected easily in this manner.

### 2.4 Mechanics

#### 2.4.1 Introduction

The demonstrator, which is driven by the microsystem, is an electric window winder like it is widely used in car doors, see Figure 2. Such a macro-system consists of different components which comprise different domains, like electrical (DC-motor with Hall-sensor), mechanical (gear, cable, window, guide rails), electromechanical (DC-motor), electromagnetic (DC-motor, Hall sensor). As an example for the basic effect approach we have a closer look at the mechanical effects, which are:

- Inertia: gear wheels (rotational), window (translational)
- Static and dynamic Friction: gear, guide rails
- Transmission (rotation-rotation, rotation-translation) and clearance in a gear
  Note: the cable is implemented as gear with rotational input and translational output. This approach serves to convert the rotation into a translational movement.
- Mechanical stop: upper and lower limit, "children hand"

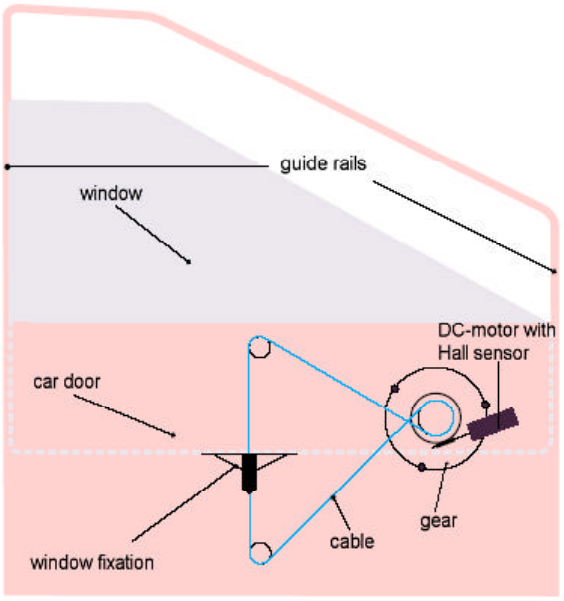


Fig. 3. Automotive window winder system

### 2.4.2 Modeling and Implementation of the Mechanical Parts

The electro-mechanical and electro-magnetic parts of the envisaged demonstrator is based on the modeling of basic physical effects which occur in or between single components of the mechatronic sub-system. The advantage of this kind of modeling is the high reusability of already modeled and validated effects. This approach benefits from the option that the designer is able to decide whether an effect will be considered or not and so influences the simulation precision and speed. Another advantage is that no knowledge of the model equation of the complete system is necessary - it is automatically build inside the simulator by combining the needed effect models.

In VHDL-AMS we are not bound to given variables. This is an advantage, since we can implement variable types of non electric domains, which we want to simulate. VHDL-AMS offers outstanding possibilities for the definition of user-defined variable types. Table 1 shows the definitions of the flow and potential variables like they are recommended by the IEEE and for this system are excellent suitable.

| *Definition of flow and potential variables* | | | |
|---|---|---|---|
| ***Variable type*** | Electrical network | Rotation | Translation |
| ***Through flow-variable*** | Current i | Torque T | Force F |
| ***Across potential-variable*** | Voltage v | Angle φ | Displacement x |

Tab. 1. IEEE recommended definition of flow and potential variables

The restrictive port definition guarantees both the compatibility of the effect models among themselves, if one wants to interconnect them, and a simple expandability of the model library.

A simple example of physical effect implementation (the model of the gear transmission rotation/rotation without static friction or clearance) in VHDL-AMS is shown in Listing 1. The model contains two formulas:

- relation between the output torque and the input torque: $\boldsymbol{T}_{out} = -\boldsymbol{T}_{in} * \boldsymbol{g}_r * h$
- relation between the angle of the input gear wheel and the output gear wheel: $q_{out} = - q_{in} / \boldsymbol{g}_r$

With $\boldsymbol{T}$ the torque, $\boldsymbol{g}$ the gear ratio, $h$ the efficiency factor and $q$ the rotation angle.

```
01. LIBRARY IEEE;
02. USE IEEE.MATH_REAL.ALL;
03. USE IEEE.MECHANICAL_SYSTEMS.ALL;
04. ENTITY gear_rot_rot IS
05. GENERIC (g_r : REAL := 1.0;  -- gear ratio
06.          eta : REAL := 1.0); -- degree of effectiveness (0 <= eta <= 1)
07.
08. PORT (TERMINAL rot_p_in, rot_n_in,
09.                rot_p_out, rot_n_out: ROTATIONAL);
10. END ENTITY gear_rot_rot;
11.
12. ARCHITECTURE simple OF gear_rot_rot IS
13. QUANTITY theta_in  ACROSS T_in  THROUGH rot_p_in  TO rot_n_in;
14. QUANTITY theta_out ACROSS T_out THROUGH rot_p_out TO rot_n_out;
15.
16. BEGIN
17. T_out     == - T_in * g_r * eta;
18. theta_out == - theta_in / g_r;
19. END ARCHITECTURE simple;
```

Listing 1. VHDL-AMS implementation of the rotational/rotational gear transmission

### 2.4.3 Modeling and Simulation of the Mechanical Sub-system

For modeling the mechanical sub-system, the available models have to be interconnected. Due to the defined flow and potential variables the related models are port compatible. A symbol was created to each VHDL-AMS model, which can be interconnected then with another symbol in the schematic editor. The creation of the net list of the system can be done automatically with the net list generator from the schematic editor. For exemplification the schematic of the sub-system is shown in Figure 4.

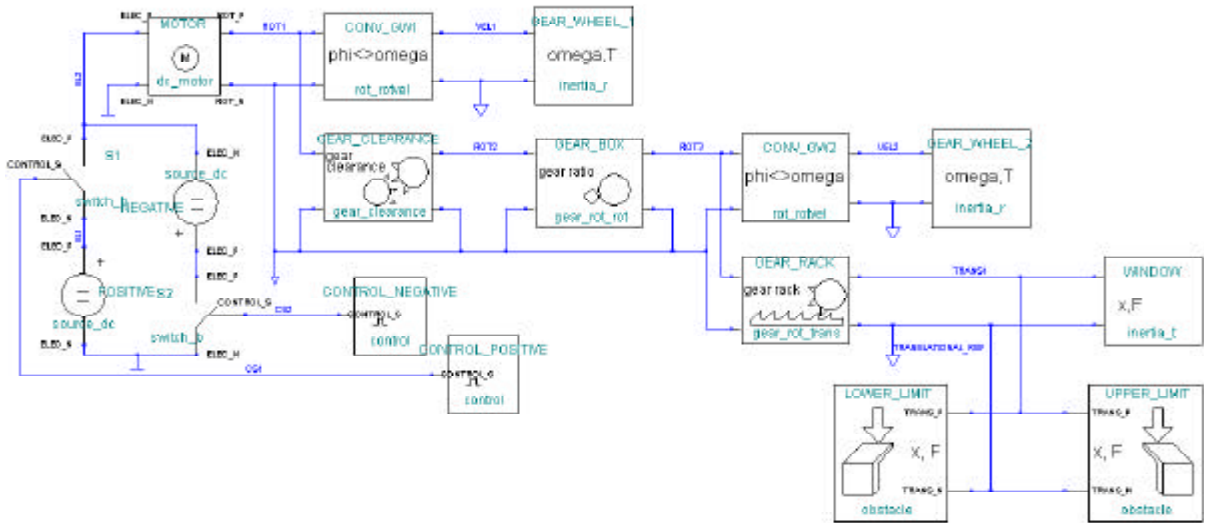


Fig. 4. Schematic of the automotive window winder sub-system with testbench

After the fundamental physical effects were identified, implemented in VHDL-AMS and combined to the electro-magneto-mechanical macro system, the model of the mechatronic sub-system is ready to simulate.

## 3 Simulation Results and Conclusion

The model setup as described above has been simulated using 12 tests and related 12 embedded test programs. These were developed to verify the design of the (micro)system-in-a-package as described above. Virtually the complete functionality of the product can be assessed in its environment. These simulations turned out to be a cornerstone in the SiP-verification, as they helped to avoid a redesign. The simulation speed is in the range of 20 CPU-minutes per second real-time, which allows to accomplish extensive verification runs.

The limitation of this kind of simulations lies in the fact, that only those effects are taken into account, which have been introduced into the models. So if we know, that self-heating is an issue, we can take care of that. But if we do not take into account some certain effect, the system may later fail because of that, though the simulations were successful. In other words, this means that we get some information on whether the system can work under certain conditions rather than whether it will work under all conditions.

The shown drive system served as a demonstrator to assess the opportunities of complete system simulations. With that, the neuralgic scenarios can be assessed before tapeout of the chips. Even more, the complete controllability and observability of a simulation helps a lot in quickly tracking down the problems. All in all, the proposed approach will help to jointly get a much better understanding on the behavior of an even complex SiP in its environment.

## 4 References


[1] G. Pelz, "Mechatronic Systems: Modelling and Simulation with HDLs", John Wiley & Sons, 2003.

[2] L.M. Voßkämper, R. Schmid and G. Pelz, "Combining Models of Physical Effects for Describing Complex Electromechanical Devices", IEEE/VIUF Workshop on Behavioral Modeling and Simulation (BMAS), Orlando, Florida, 2000.

[3] D. Dammers, P. Binet, L.M. Voßkämper and G. Pelz, "Motor Modeling Based on Physical Effect Models", IEEE International Workshop on Behavioral Modeling and Simulation (BMAS), Santa Rosa, California, USA 2001

[4] D. Maurer and L.M. Voßkämper, "From Chip to System Design using a Co-verification environment", ECE Magazine September 2005, Page 34-36

[5] G. Pelz, "The Virtual Disk Drive - Mixed-domain support for disk electronics over the complete life-cycle", IEEE International Workshop on Behavioral Modeling and Simulation (BMAS), Santa Rosa, California, USA 2001

[6] M. Otter, "Objektorientierte Modellierung Physikalischer Systeme, Teil 1", at - Automatisierungstechnik 47 (1999) 1, R. Oldenbourg Verlag


Authors


Georg Pelz, Christian Decker, Dieter Metzner

Infineon Technologies AG
Automotive Power
Am Campeon 1-12, 85579 Neubiberg
E-mail: Georg.Pelz@infineon.com

Dirk Dammers, Lars M. Voßkämper

Dolphin Integration GmbH
Bismarckstr. 142a, D – 47057 Duisburg
E-mail: mems@dolphin-integration.com



**Keywords: microsystem, macrosystem, mechatronic, modeling, electro-magneto-mechanical effects, VHDL-AMS, multi domain, mixed-signal**