Simulation-based Development and Testing Environment for Electric Vehicles

Karsten Krügel, Lars Stockmann, Dominik Holler, Klaus Lamberg

Abstract

Vehicle electrification – where the usual vehicle drives are combined with or replaced by electrical drive concepts – makes particularly tough demands on the development process and development methods compared with the conventional development of vehicles. Because electric and hybrid electric drive concepts involve technologies from many domains and disciplines, there is a growing need to validate design and concept decisions in very early development phases. It is not only a question of validating the individual ECU functions required for electrification, such as energy management. It is more a matter of validating and testing their interaction with classic ECU functions. Thus, early validation requires solutions for simulating all the components involved, from the control functions to the networking of electronic control units (ECUs) to the mechanical and electrical vehicle components.

This article presents a design and test environment for electric vehicles. The environment comprises all the model parts needed for simulating the entire vehicle, such as models of the electric drive, the battery and the auxiliary electric loads. At the heart of the design environment are virtual ECUs, models of the control functions that can be simulated in conjunction with other software parts of an ECU such as the operating system and other basic software. These can be simulated and automatically tested in an overall environment during interaction with conventional model parts such as the vehicle dynamics, maneuvers and road models. The article also shows how models, data sets and tests can be reused in the development process on a hardware-in-the-loop (HIL) simulator.

The available concepts and solution approaches for designing and testing electromobility systems are described with the development of a new brake power recuperation control as an example.

Kurzfassung

1 Introduction

The automotive industry is on the verge of a paradigm shift from combustion engines to electric drives, motivated by factors such as the need to reduce emissions and dwindling fossil fuel resources. Every manufacturer's vehicle range now includes hybrids. The development of purely electric vehicles is still a niche topic, however, especially in Europe. One reason for this is the higher cost of developing E-vehicles by new using technologies, and the investments these involve, compared with the development of conventional drives.

Technologically, electromobility presents a whole range of challenges and opportunities: Optimum control of all consumers and the use of suitable energy storage technologies are important for the vehicle's range, the number of charge cycles, and the charging times and lifetimes of the energy storage media, and therefore decisive for end customers' acceptance of E-vehicles. Connecting to the power grid requires new infrastructure concepts, for example, for charging stations, but also allows vehicle batteries to be used as decentralized storage media. All these aspects involve dependencies and interactions that need to be investigated and tested during E-mobile development. Thus, the overall system must be developed and optimized across multiple domains.

In the conventional vehicle development process, the different control components and ECUs are initially developed largely independently of one another. An ECU is not usually tested as a whole, by means of hardware-in-the-loop (HIL) test benches and test drives, until a relatively late stage. Constructing suitable test benches and vehicle prototypes is also cost- and time-intensive, especially considering the high number of dependencies and interfaces with the environment that are typical with electromobility.

This paper describes a design and simulation environment in which numerous tests can be executed purely virtually with simulation support at an early stage of the development process. The design environment enables control and optimization strategies to be tested for multiple domains and for any kind of vehicle. This helps reduce the cost of test benches and prototypes, avoid design loops, and make the development process considerably more economical.

2 Developing and Testing Automotive ECUs

Over the last 15 years, HIL simulation has proven its value in the field of electronic system quality assurance, both for suppliers and most particularly for car-makers. Instead of being installed and tested in a real vehicle, the ECUs under test are connected to a simulation system that simulates the entire environment, plus any ECUs that are missing.

HIL simulation uses powerful models for the real-time simulation of the controlled system (engine, transmission, vehicle dynamics, etc.), sensors and actuators, and also other ECUs (soft ECUs) with which the test object communicates in the vehicle. Calibration and diagnostic tools provide access to the internal variables and fault memory of the ECU under test. As a rule, other tools are also used to parameterize the real-time models, to manage the parameter sets for the ECUs and the models, and to visualize the behavior of the simulated vehicle by graphical instruments and 3-D animation. Using test automation tools makes it possible to perform comprehensive, systematic and automated tests based on the requirements for specific ECUs or for the overall system.

In addition to HIL simulation tests, another now established procedure is to run PC-based tests of single functions. These single functions are developed model-based, i.e., with tools such as MATLAB/Simulink for modeling and simulation, and
TargetLink for automatic production code generation. With the aid of model-in-the-loop (MIL), software-in-the-loop (SIL) and processor-in-the-loop (PIL) simulation, single functions can be verified on a developer PC at a very early time, even before they are implemented in the ECU [1]. However, errors are caused not only by single functions, but also increasingly by the interaction between functions and between ECUs. Over the past few years, a growing number of projects have aimed to develop the entire ECU software on a model basis. These system and software architecture models involve modeling ECUs and entire electric/electronic systems at a higher level than single functions and controllers. The best-known example of such models, AUTOSAR [2], standardizes the description of (distributed) applications and networks, interfaces, and the configuration of basic software modules. System models allow systematic, top-down system design with the architecture of the ECU software as the starting point. The next step is to deploy the software components on the ECUs that are used in the vehicle and interconnected via bus systems such as CAN, LIN or FlexRay.

![Figure 1: Mapping of Software Components on ECUs](image)

The software architectures and systems that are modeled in this way are closely connected to actual function development with tools like Simulink or TargetLink. The behavior of single functions and software components is developed with Simulink or TargetLink, and can be simulated and tested directly in this environment. Because they are embedded in the system context in the architecture model, the functional connections between software components, and the connection between the software components and lower-level software such as diagnostics, are also described.

If the resulting, software-integrated function code is first executed on a PC, large parts of the overall system behavior can be verified even before code implementation on the real ECU. This requires simulation of the basic software such as the operating system or of fault memory management. ECU simulation like this is called a virtual ECU (V-ECU).
The virtual ECUs created this way enable the frontloading of HIL tests. This requires a PC-based simulation platform on which virtual ECUs can be simulated in conjunction with powerful real-time models to which tools used in HIL simulation can be connected for access to controlled system models and ECUs, and with which automated testing can be performed. Such a simulation platform is available: the dSPACE Offline Simulator [3]. The Offline Simulator is able to simulate models from multiple domains, e.g., virtual ECUs together with the models of the controlled system, sensors and actuators used in HIL applications. dSPACE Offline Simulator is the foundation for the design and test environment for electric vehicles that is described in the next section.

3 Simulation-Based Development and Early Testing

A simulation-based development and testing environment was designed with the offline simulation techniques described in the previous chapter. It allows ECU functionality to be tested on a much more integrated level compared to conventional unit or module tests in a common MIL or SIL testing environment. One of the biggest strengths of this new environment is that it is based on the same tools, protocols and standards that are normally used in a classical HIL testing environment. Thus, it is possible to reuse models, data, parameters and tests throughout the ECU development process.

A part of this process is shown in Figure 2 and is divided into three different levels. First, there is the design phase, where the new controller functionality is designed in MATLAB/Simulink by one (or more) function developers. The functionality of the resulting models is roughly validated by unit or module tests, in which the “open ends” of the function model are stimulated by signal generators to determine whether the function behaves “reasonably”. Later more complex test scenarios are performed with other functions or realistic plant models. In this early phase, function developers prefer ready-made Simulink models. The dSPACE Automotive Simulation Models (ASMs), a model library commonly used in HIL scenarios, can be used as a basis for this early function verification. They include a multi-body system of a car, customizable maneuvers, roads and ready-to-use models of electric components. They can also be parameterized with the graphical parameterization environment ModelDesk and extended with users’ own models.

After the functionality of the new controller is validated, the functional behavior model is converted into production code. This can be done automatically by using dSPACE TargetLink. With this code, SIL and PIL simulation can be performed for the function model with the same environment models as were used previously. The environment model itself is not affected by code generation. The behavior of the production code can therefore be compared with the results from the earlier MIL simulations.

After confirming the correct functional behavior of the single controller models, it is important to ensure that the application software of an entire ECU or even an ECU network behaves correctly. This is done by the system architect.
Thus, alongside the functional models, a system architecture is designed or adapted to facilitate the integration of the application code. This is depicted in the middle level of Figure 2. There is no chronological order implied here, e.g., the function developer and the system architect can both start at the same time. This satisfies two typical use cases: the creation of a system architecture from scratch as a starting point for simulating functional code together with nonfunctional code, and the integration of new or modified function code into an existing ECU topology. Both use cases are common for the early development stages of electric vehicles and both are complex tasks. In the development and testing environment described in this paper, dSPACE SystemDesk is used to assist the system architect.

With SystemDesk, the system architect generates V-ECU executables, which can be simulated on a standard PC using the dSPACE Offline Simulator, shown between the system architect level and the tester level in Figure 2. In addition to the application software of an ECU, a virtual ECU includes an ECU operating system as well as some hardware-independent basic software. If information about the communication behavior of the ECU network is included in SystemDesk, the system architect can also use a virtual bus simulation.

With these V-ECUs, more integrated and thus more realistic closed-loop test scenarios can be executed with the dSPACE Offline Simulator on a standard PC compared to a common MIL/SIL simulation environment. Moreover, the environment models used here were also used on the function developer level and can also be used later in a HIL scenario.
The last part of the process is the tool chain for experimentation, visualization, and testing, which can be seen in the lower third of Figure 2. One of the main goals of this simulation and testing environment for electric vehicle ECUs is to provide a way to use the same tools, models and tests on a standard PC as are normally used in a HIL testing environment. To achieve this, a V-ECU, like a real ECU, supports measurement and calibration according to the ASAM MCD-1 MC (XCP) standard. This means that ASAP2 files (according to the ASAM MCD-2 MC standard) are used to describe the mapping of the variables. This is important, as common HIL test and experimentation tools like ControlDesk Next Generation need these variable descriptions. The offline simulation environment supports established interfaces like the HIL API for the same reason. Classical automated test tools like dSPACE AutomationDesk make use of this standardized interface. Thus, the huge amount of tests that are normally used for testing real ECUs on a HIL simulator can be reused for testing V-ECUs in an offline simulation on a standard PC.

In addition, MotionDesk is used for plausibility tests, as it renders the simulated ASM model as a 3-D animation in vehicle dynamics scenarios.

To evaluate this tool chain, a concrete application example is needed. The “E-Mobil” project provides such an example. Both the project and the application example are presented in the next section.

4 The “E-Mobil” Project

The last section presented a development environment for the early development of software for electric vehicles. In order to evaluate and further enhance this environment in the context of electromobility, the “Simulation-Based Development for Electric Vehicles” (in German: “Simulationsgestützter Entwurf für Elektrofahrzeuge”, or “E-Mobil” for short) project was set up. The project supplies the necessary application example. The aims of the project are to create new controller and environment models from the context of electric vehicles, and to evaluate the tool chain presented in section 3 by going through a concrete development process. The project partners are DMecS GmbH & Co. KG, the LEA institute, the C-LAB (both University of Paderborn) and dSPACE GmbH. Every partner has its designated role in the development process of the application example. LEA creates the controller specifications and environment models relating to power electronics and electric drives. They mainly work on the left side of the function developer level depicted in Figure 2. As an expert on integrated mechatronic systems, DMecS creates controllers and environment models relating to vehicle dynamics, and also sets up the overall model. They are mainly positioned on the function developer and the tester level. The C-LAB devises methodologies to transfer and design the controller with regard to AUTOSAR. This applies primarily to the right part of the function developer level and the system architect level. Finally, dSPACE serves as the supplier of tools and know-how for developing and testing automotive software on all levels. The project partners chose a recuperation scenario for the example application. In the following this scenario is described and how it is implemented.

While fuel has an energy density of about 1200 Wh/kg, modern electric energy storage systems only supply an energy density of about 140 to 180 Wh/kg. Hence, efficient energy use is essential. This requires the OEMs to integrate additional optimized ECUs that implement algorithms to efficiently actuate the electric components, especially the controller for the electric machine. This controller greatly influences the performance of the whole drivetrain. Aside from that, energy recuperation plays an important role in every electric vehicle. It is based on the fact that electric machines can natively operate in both directions, i.e., as motors and as generators. When an electric machine operates as a generator, the kinetic energy of
the car is transformed back into electrical energy that can be used to recharge the electrical storage system. This effectively results in a negative torque that decelerates the vehicle. However, the torque that the motor can deliver is limited by the maximum allowed current. Thus, an additional electrohydraulic brake system is necessary for full braking. A control mechanism is needed that appropriately distributes the brake request onto the electrical and the hydraulic part of the brake. In addition to maximizing recuperation, ‘appropriately’ here means satisfying safety requirements and also providing a suitable brake feeling and comfort for the driver. Another field is energy management, which monitors and controls energy distribution among the electric loads. For example, in a situation where the driver requests maximum acceleration, like in overtaking maneuvers, energy management might reroute the power of the air condition system to the electric machine.

In the E-Mobil project, the three controllers – for the electric machine, for the brake system and for energy management – are designed, implemented and integrated into an AUTOSAR software architecture using the tool chain presented in section 3. They are tested with closed-loop simulation against an environment model that realizes a recuperation scenario. As already mentioned in section 3, the environment model is based on the dSPACE ASM library, which provides models for the mechanical parts of the vehicle, including a multi-body system. The electric components, the driver and the customizable maneuver models are also used. Additionally, the project partners create their own models for power electronics and electrical drives (LEA), powertrain, brake system and steering (DMecS) and further models of electric loads (dSPACE). Together with the models originating from the ASM library these models are integrated into an overall model. The structure of the overall model, which is depicted in Figure 3, has been designed with electric vehicles in mind, so the original ASM blocks must be adapted to fit into this new structure.

![Figure 3: Structure of the overall model](image-url)
In the “E-mobil” project special care is taken to examine and model the different interfaces between the components. An additional Sensors & Actuators layer, which is not yet available in the standard ASMs, is also modeled before the actual ECU functionality.

For the environment part, it is planned to model and use driving cycles that are commonly used to assess the performance of vehicles in terms of fuel consumption and emission, like the New European Driving Cycle (NEDC) and the American Federal Test Procedure (FTP-75) as well as custom driving cycles that best fit the recuperation scenario.

The left part of the structure is related to the control functionality. Thus, this is the place where the brake controller, the controller for the electric machine and the energy management is specified and part of which is transformed into actual production-level software and finally integrated onto a V-ECU as described in section 3. The next section describes this process in one example case, specifically a recuperation-ready brake controller.

5 Simulation-Based Control Software Development for a Recuperation-Ready Brake Controller

This section presents a model of an ASM-based electric vehicle and its environment for the recuperation scenario introduced in the previous section. Note, however, that in this example only native ASM blocks are used as proposed in section 3. This means that some components may not be specifically designed for electric vehicles (like a brake system with a vacuum servo or a conventional transmission) but in spite of that are still quite usable for early closed-loop simulations. Moreover it is possible to adapt the standard ASMs to behave like an electric vehicle by using appropriate parameterization. The core functionality of the modeled controller is to split the requested braking torque between the recuperative braking of the electric machine and the braking system that is provided by the ASMs. Three development steps are presented: the modeling of the behavior, the modeling of the software architecture, and the modeling of the software implementation.

Figure 4 shows the top-level view of the behavior model functionality of recuperative braking in MATLAB/Simulink. The Vehicle Simulation Model in red on the right side provides a model of the environment to analyze the recuperative braking behavior.
This environment is composed of a battery model, a model of the electric drive and its related power electronics, models of the drive train, maneuver, road and driver. The recuperative braking functionality is located in the blue EnergyManagement subsystem on the left side. It is modeled in a simple manner to focus on the concept of controller software development. For example, the controller assumes that the car always moves forward. Aspects of electronic stability control functions like anti-lock braking, anti-slip regulation, etc., are also ignored for the same reason. The controller here uses only the driver’s request through pedal positions and the maximum torque the electrical machine is able to recuperate. A real controller would take more information into account, e.g., the maximum electrical energy in the electrical storage system, which may consist of capacities, the battery and its management system. The controller used here provides set points for the electric machine and the mechanic brake.

![Figure 5: Simulink model of the brake controller](image)

As Figure 5 shows, the controller is split into five components: Three components preprocess the input values, one implements the core functionality and one postprocesses the output. The components Deduce_Brake_Req and Deduce_Acc_Req deduce the driver’s torque request from the positions of the brake and acceleration pedals. Calc_Max_Brake_Trq_Machine provides the actual maximum torque the motor is able to convert, which depends on its characteristics, including effects like field weakening mode. Because it is not possible to set a concrete braking torque in the ASM brake, the internal value Trq_Brake is mapped by the component Calc_p_Brake to a desired pressure in the braking system.

![Figure 6: Split_Brake_Trq subsystem, the actual recuperation control functionality](image)

The core functionality is provided by the component Split_Brake_Trq. The modeled behavior is quite simple, as shown in Figure 6. If the driver requests greater deceleration than the electric machine can provide, the mechanical brake system must make up the difference. In this behavior modeling step, it is possible to execute the model of the controller together with its environment in MATLAB/Simulink. This is the MIL simulation explained in section 3.
The next step is to model the software architecture. In dSPACE SystemDesk, the components are modeled as AUTOSAR software components, including ports, interfaces, data elements and scalings. Figure 7 shows the composition of the software components. The description of the software components is transported to dSPACE TargetLink through standardized AUTOSAR files. The implementation is generated in TargetLink. First, TargetLink creates a frame which reflects the information about the software components and represents every function (AUTOSAR runnables). TargetLink also creates a subsystem with ports according to the AUTOSAR description. Then, the behavior modeled in MATLAB/Simulink is inserted in the frame and enhanced with implementation-specific information, like the scaling of internal fixed-point values. The scalings of the outer ports’ data elements are specified by the AUTOSAR description. TargetLink uses this implementation model to generate source code in the C programming language. A SIL simulation based on this source code and the same environment model in MATLAB/Simulink is used to compare this implementation with the MIL results.

Figure 7: Software architecture of the brake controller in dSPACE SystemDesk

Figure 8: MIL/SIL simulation (left) and SIL simulation in ControlDesk using the dSPACE Offline Simulator with A2L measurement support (right)
On the left side in Figure 8, the result of the MIL simulation (smooth green line) differs from the SIL simulation using the fixed-point implementation (red line) because of quantization effects.

The implementation source code files are transmitted to SystemDesk using AUTOSAR mechanisms. SystemDesk is able to combine these files to make one or more V-ECU(s). It uses information on the software architecture for this, e.g., the distribution of the software components across different ECUs and the configuration of the basic software including the operating system and communication stack. To finally simulate the V-ECU on the dSPACE Offline Simulator both the V-ECU and the model of the environment (the red box in Figure 4) are converted into an Offline Simulator executable. In this way the controller software is prepared for experimentation, visualization, and testing as described at the end of section 3. The left side of Figure 8 shows the comparison between MIL and SIL simulation in the TargetLink environment. The right side presents the simulation results of the virtual ECU in dSPACE ControlDesk.

Finally, to evaluate the controllers, the overall environment model presented in section 4 is used. A maneuver that consists of two segments was devised. In the first segment the vehicle is accelerated to a speed of 50 km/h (13.9 m/s). Then we feed a ramp-shaped stimulus to the brake pedal until the vehicle stops.

Figure 9: Behavior of the simplified recuperation controller

Figure 10: Behavior of the early version of the new recuperation controller together with the new electrohydraulic brake system
It was expected to see the controller distributing as much of the requested brake torque onto the electric machine as possible. Then, at the point where the request can no longer be fulfilled by recuperation, the mechanical brake must start to operate. This behavior can be seen in Figure 9. In this simulation the vehicle drove a total of 142 m, and about 69% of the energy that was used for acceleration was recuperated.

Our project partners provided an early version of their controller specification and a model of an electrohydraulic brake system. The latter is used to replace the brake system of the ASM that models an vacuum servo. The behavior of this system is shown in Figure 10.

It can be seen that the results are comparable though, the behavior of the controller is different with regard to the calculation of the requested brake torque from the brake pedal position.

6 Conclusion and Future Work

The growing number of ECU functionalities, and the resulting increase in complexity, necessitates new, efficient ways to develop and test ECU software systems, especially for electric vehicles. A cross domain approach that allows for validation and verification by PC-based simulation helps to speed up the process in early development phases. Seamless interplay between the participating tools is essential for this. From function design to system architecture design and simulation to experimentation and creation of automated tests, tools must allow reusability. This means that environment models, tests and protocols used in early phases are utilizable for simulation in every development phase including later HIL setups.

This paper presented a dedicated simulation-based development and test environment for electric vehicles. This environment was evaluated in a concrete example application from the field of electromobility. It was shown how an example ECU functionality for a recuperation-ready brake system is developed and tested in this environment. This demonstrated that a control functionality can reach the quality level of a virtual ECU within a few days of development. Also, an environment model made solely from ASM blocks is suitable for first experiments and tests in a complex closed-loop scenario.

In the course of the “E-Mobil” project, two more ECU functionalities will be realized this way. At the end of 2013 these functionalities will be integrated on two V-ECUs and simulated against an ASM-based environment model of an electric vehicle. This model will be enhanced with new sophisticated models from the context of electromobility contributed by the project partners. At the same time, the tools will be advanced and extended according to the needs of the project.

In the coming years, a trend towards “virtual development” and “virtual testing” is expected. The overall goal here is to increase the maturity of the software and systems through simulation-based validation and verification using virtual ECUs and detailed environment models in early phases of development. In the future, the models and simulation platforms will therefore have to be scalable in terms of the available CPU power. Moreover, the huge number of models, whose parameters, tests and test results will be reused throughout the whole process require new ways to handle and manage all the data.
Acknowledgements

The project “Simulationsgestützter Entwurf für Elektrofahrzeuge” is funded by the European Union and the state of Northrhine-Westphalia (grant No. 64.65.69-EM-1008A).

References


Autoren

Dr. Karsten Krügel, dSPACE GmbH, Paderborn
Lars Stockmann, dSPACE GmbH, Paderborn
Dominik Holler, dSPACE GmbH, Paderborn
Dr. Klaus Lamberg, dSPACE GmbH, Paderborn