2018-01-5012 Published 15 Apr 2018

Virtual Functional Testing of a Mechatronic Active Roll Control

Hua Huang Schaeffler Technologies AG & Co. KG

Michael Hartmann QTronic GmbH

Hasan Flaih Awad and Dustin Knetsch Schaeffler Technologies AG & Co. KG

Citation: Huang, H., Hartmann, M., Awad, H.F., and Knetsch, D., “Virtual Functional Testing of a Mechatronic Active Roll Control,” SAE Technical Paper 2018-01-5012, 2018, doi:10.4271/2018-01-5012.

Abstract

Compared with the traditional bench testing approach, the virtual validation on a PC is more time-saving and cost-effective. However, the control software of an electronic control unit is typically developed together with suppliers and original equipment manufacturer, and each owns parts of the source code. So it is impossible to set up a simulation based on the original C code. Alternatively, to set up an equivalent model is usually time-consuming and less effective. In this work, Schaeffler describes how to develop a virtual testing method based on chip simulation for the series produced mechatronic active roll control system.

Keywords

Virtual testing, chip simulation, functional testing, active roll control, automotive

Introduction

Schaeffler’s intelligent active roll control (iARC) is a series produced mechatronic system, shown in Figure 1. The system comprises an external electronic control unit (ECU) and an actuator, which includes a motor/gearbox unit, two torsion bars, and the wire harness. Two iARCs are separately installed on the front and rear axles of a vehicle. The active rotation of the two torsion bars in relation to each other allows the vehicle roll to be reduced during cornering, and also increases the driving comfort over uneven roads [1]. Meanwhile, this active stabilizer also helps to reduce fuel consumption and emissions compared with a hydraulic one.

The iARC system is adapted according to the requirements from the original equipment manufacturer (OEM). For example, for better noise vibration and harshness (NVH) performance or faster dynamic behavior. Before the improved iARC system (both software and hardware) is delivered to the OEM, it undergoes validation on the test bench with various test cases.

One of the main foci of the validation is functional testing, where the iARC is mounted in an environmental chamber and the two torsion bars are separately connected with two hydro-mechanical pulsers via coupling rods.

The control behavior and robustness of the software are validated in such an environment, where the system is tested under the predefined profiles, which includes a target actuator torque with/without external disturbances acting on the torsion bars. The target actuator torque is in the form, such as of a square signal with different amplitudes, of a sinusoidal signal with various frequencies, and of the measured profiles acquired from a driving vehicle.

A basic functional test (with a constant supply voltage in an ambient temperature) lasts about 2 days with non-stop running on the test bench.

FIGURE 1 Schaeffler’s iARC system, integrated in the front axle of an SUV.

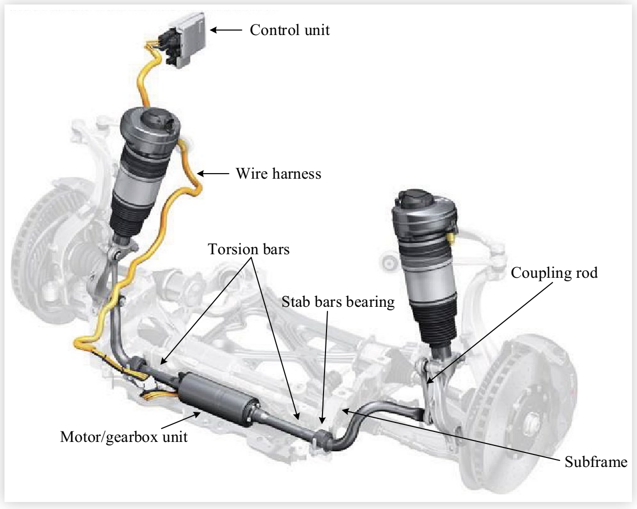


Figure 2 illustrates the costs for this system test, where the preparation includes the assembly, cabling, etc. The evaluation contains the signal post-processing and the measurements analysis. It can be found that the manpower and equipment account for a large portion of costs.

Virtual testing on a PC can offer a more cost-effective validation approach. Different test scenarios can be run in parallel. Meanwhile, a mathematical optimization also needs this kind of simulation, which does not have a real-time constraint.

The challenge of creating such test is in virtualizing the ECU, i.e. to run the control software on a PC by emulating the target processor of the ECU, while the physical model can be developed in a programming environment such as MATLAB®/Simulink®, Dymola®, or SIMPACK®.

There are three principal options to set up an ECU simulation:

- Re-host the C code
- Reverse engineering
- Chip simulation

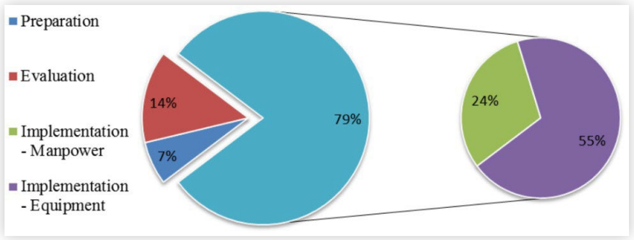
Since the control software of iARC is developed together with another supplier and OEM, who own parts of the original C code, it is not possible to set up an ECU simulation through re-hosting. Even when the C code is available, it still has its own specific constructs, which prevents compilation for other targets, such as Windows® x86 platforms. Reverse engineering is also impractical because setting up an equivalent software is time-consuming, and such model cannot represent the real ECU realistically. Chip simulation is a suitable solution for the ECU virtualization, which requires neither C code nor reverse engineering. However, usually the chip simulators lack required interfaces to automotive standards and tools used for closed-loop simulation, measurement, and calibration [2]. This limits the application of such a method.

In cooperation with QTronic GmbH and its virtual ECU tool Silver [3, 4, 5, 6], it is now possible to effectively use chip simulation for the virtual testing. The Silver's chip simulator uses a HEX file complied for the target processor of the ECU, and an ASAP2/A2L file is taken to define the interfaces involved in the simulation.

In contrast to other related work, such as the virtual system prototypes (VSP) [7], Silver does not virtualize the entire system on a chip but the specific tasks selected by the user (detailed implementation is described in the section "Development of the Virtual ECU"). This significantly simplifies the setup of such simulations and accelerates the simulation speed because only a fraction of the code is run. Another technical alternative is to connect a PC to the real ECU through JTAG debug interface [8], but the drawbacks are that a real ECU is needed and the simulation must run in real-time. dSPACE's products, such as SystemDesk® and VEOS, can also supply a PC-based simulation platform to validate the software of ECUs [9]. But they do not provide support to directly simulate based on HEX files, this restricts the application on chip simulation.

The remainder of this article is structured as follows. Next section describes the virtual testing platform. The approach of ECU modeling and simulation is presented in detailed in section "Development of the Virtual ECU". Afterwards, the application of virtual system functional tests for the iARC is shown in section "Application". Finally, a summary is provided, and an overview of future work is also introduced.

FIGURE 2 Costs of the functional test on the bench.



Virtual Testing Platform

Figure 3 shows the virtual testing platform used by Schaeffler. It mainly consists of following parts: A virtual controller obtained through chip simulation, a detailed dynamic model library for different validation requirements, and a restbus simulation to emulate parts of in-vehicle buses, such as the controller area network (CAN). Additionally, the library of test profiles are defined to simulate testing scenarios. When the test profiles include external disturbances, two hydro-mechanical models of the pulser system are also needed. The pulser system supplies the external displacements respectively on the left and the right sides of iARC, through the coupling rods. Moreover, in order to realize the automation, a Python® based script is developed. The post-processing is also implemented for data analysis, together with the saved measurements. In the end, a testing report is automatically generated.

Modeling

A simplified physical model of iARC can be described in equation 1, where J_{mot}, M_{mot}, φ_{mot} are respectively the moment of inertia, torsion torque and angle of motor. M_{fri} is the friction torque. M_{act} is the torsion torque at the end of the torsion bars. i is the transmission ratio of the gear box. φ_{offset} and φ_{ext} are the offset of the torsion angle in the initial phase and the external torsion angle effected on the torsion bars (through the level arms). f is the stiffness of torsion bars.

$$\begin{aligned} J_{mot}\ddot{\varphi}_{mot} &= M_{mot} - M_{fri} - \frac{M_{act}}{i} \\ M_{act} &= f\left(\varphi_{act}\right) \\ \varphi_{act} &= \frac{\varphi_{mot}}{i} - \varphi_{ext} - \varphi_{offset} \end{aligned} \tag{1}$$

FIGURE 3 Virtual testing platform.

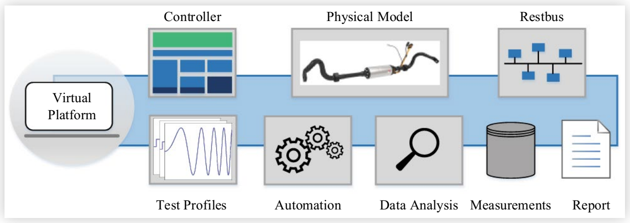


FIGURE 4 Schematic diagram of the pulser.

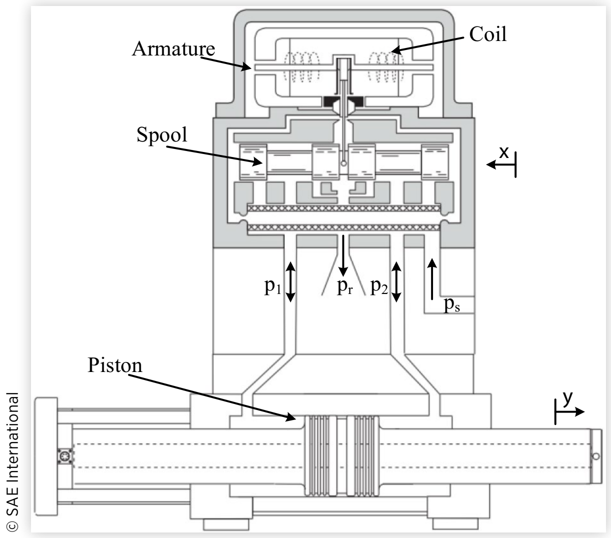
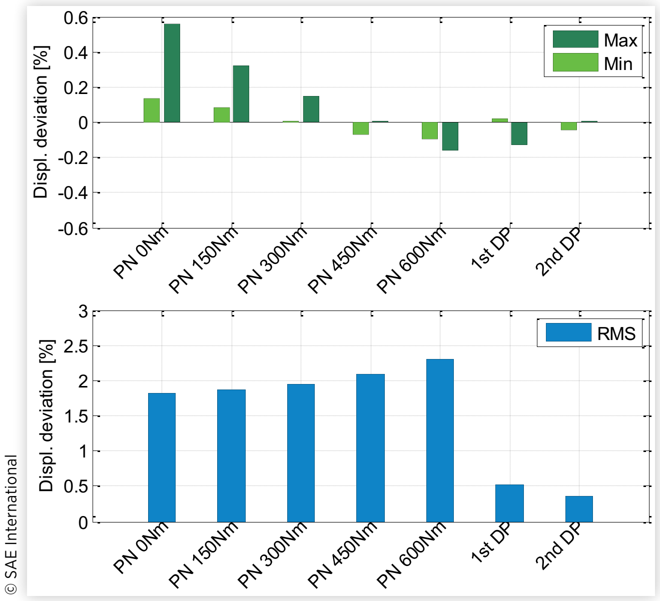


FIGURE 5 Validation results of hydro-mechanical pulser model.



In order to meet different test requirements, a library with the physical models from 1D to 3D are developed. For example, a Simulink® based model with a lookup-table of the stiffness of torsion bars is developed based on equation 1. While for the NVH analysis or the consideration of the influence from the elastic deformation of the torsion bars, a more detailed physical model built from SIMPACK® or Dymola® is necessary.

Besides the mechatronic model of iARC, two hydro-mechanical models of the pulser system [10], in Figure 4, are also needed. The hydro-mechanical models are used to respectively reproduce the external disturbance at the left and the right side of the actuator. The external disturbance is generated based on pre-defined testing profiles.

The transient response of the spool position (x) and the displacement of piston (y) of the pulser is expressed as follows,

$$\begin{aligned}
&m_p\ddot{y} = p_1A_1 - p_2A_2 - F_c\\
&p_1 = \int\left((Q_1 - Q_L) - A_1\dot{y}\right)\frac{E}{V_{01} + A_1\gamma}dt + p_{01}\\
&p_2 = \int\left(A_2\dot{y} - (Q_2 - Q_L)\right)\frac{E}{V_{02} - A_2\gamma}dt + p_{02}\\
&Q_L = (p_1 - p_2)K_L\\
&Q_1 = K_V x\sqrt{\left|p_s stp(x) - p_r stp(-x) - sign(x)p_1\right|}sign(p_s stp(x)\\
&\qquad - p_r stp(-x) - sign(x)p_1)\\
&Q_2 = K_V x\sqrt{\left|sign(x)p_2 + \left(p_s stp(-x) - p_r stp(x)\right)\right|}sign\left(sign(x)p_2\right.\\
&\qquad \left.+\left(p_s stp(-x) - p_r stp(x)\right)\right)
\end{aligned} \tag{2}$$

where p_1, p_2, A_1, A_2, Q_1, and Q_2 are respectively the hydraulic pressures, areas and flow rates in the first and second cylinder chambers. m_p is the mass of piston. F_c is the count-force on the cylinder (from the coupling rod). p_s and p_r mean the supply pressure from hydraulic accumulator and the return pressure. Q_L is the leakage rate. K_L is the flow coefficient of leakage. E is the compression module. p_{01}, V_{01}, p_{02}, and V_{02} are separately the pressures and volumes of the first and second cylinder chambers at the middle piston position. K_V is the flow coefficient. The symbol "stp()" denotes the step function, "sign()" signum function.

The effectiveness of the virtual testing is also related to the accuracy of the physical models, such as iARC and hydro-mechanical pulser. Therefore, the validation of the simulation models are also carried out.

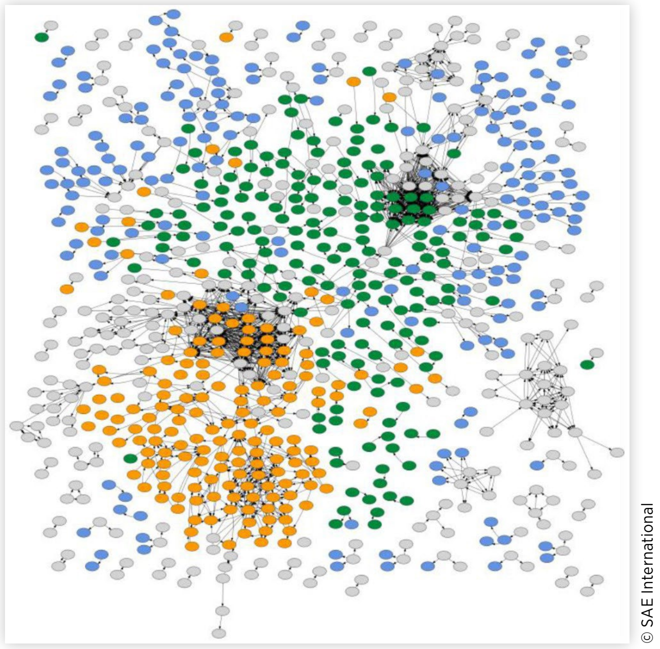
Figure 5 shows the comparison results of the pulser model with the measurements from test bench, where the pulser generates the corresponding displacements according to the pre-defined signals, such as pink noise (PN) or the acquired signals from the driving vehicles (DP). Meanwhile, the actuator operates at a required torque value, e.g. 150 Nm. The validation results shows that the physical model has a high fidelity to be used.

Controller

The main functions of iARC are anti-roll support during lateral acceleration and increasing comfort when driving on poor road surfaces. The controller is developed together with suppliers and OEM. The structure of the control software is depicted in Figure 6, where each node represents the function defined in the code. Functions colored green are developed in-house by Schaeffler. The orange nodes represent the functions delivered by another supplier. The functions from OEM are marked with blue. The gray nodes denote the basic software.

It is not easy to set up a full controller simulation since the lack of partial source codes. Through co-simulation with a simplified MATLAB®/Simulink® based function module from cooperator is possible. However, this kind of simulation cannot fully represent the functions of controller, and it also brings tedious work such as name matching of parameters

FIGURE 6 Structure of the functions in the ECU.



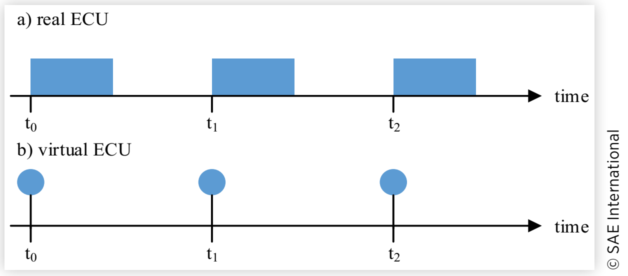
during calibration. The difference, such as the code efficiency, between the Simulink® based function modules and the final execution code also influences the control results. Therefore, a chip simulation based on the final delivered binary file is a suitable solution.

Development of the Virtual ECU

The objective of virtualization presented here is to run the functions at the application layer on a PC, such that the system performance can be verified via simulation in a closed-loop. The low-level performance, e.g. energy consumption of the CPU and bus load, is of no interest. The advantage is that the simulation can be set up quickly and requires little information about the chip and ECU, and the simulation can also run very fast because only a portion of the code is executed.

The processor of the ECU used for iARC is STMicroelectronics PowerPC, which is supported by Silver (another being Infineon TriCore). The general principle of chip simulation is: The chip simulator uses binary translation to map the instruction set of the target processor to the instruction set of the host processor on Windows® PC (instead of directly compiling C code for the Windows® x86 platform). Therefore, the chip simulator applied here is an instruction accurate one instead of a cycle accurate one in the real ECU. As described in Figure 7, in contrast to try to predict how long the execution of the tasks will take on the real ECU, the execution of the tasks in virtual ECU is assumed to take zero simulation time. The reason for this is that fast execution speed is considered a high priority during simulation, especially in the case of mathematical optimization.

FIGURE 7 Function execution on a real ECU and a virtual ECU.



Implementation

The following files are needed for preparing a chip simulation:

- Program code which contains the functions to run. It can be Intel® HEX file (.hex), or Motorola® SRecord file (.s19).
- An ASAP2/A2L file (.a2l) which defines the inputs, outputs, and parameters. The A2L file describes the data types of the variables, memory addresses, and the conversion rules used to map physical values (e.g. torque) to raw integer values used by the program code.
- An optional MAP file, which is used to specify a function directly with its name rather than memory address.
- An optional DCM file with calibration parameters to be flashed.

The chip simulation is set up as follows: The user has to write a specification file to list the interface (inputs, outputs, and parameters) of the chip simulation, as well as which functions of the ECU to run, and how exactly these function are scheduled, i.e. initially, periodically, or interrupt triggered.

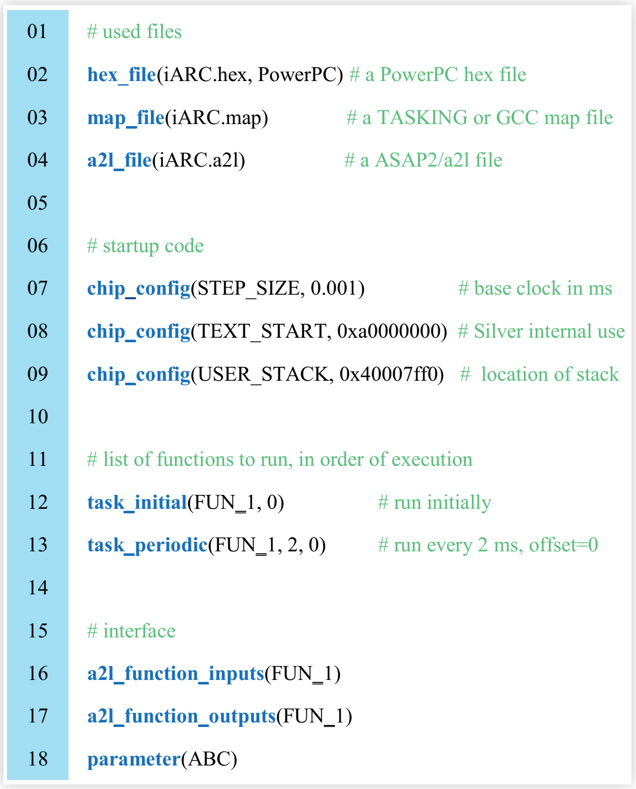
A typical specification file (written in a text file with extension .scs) is similar to that shown in Figure 8. The symbol # starts a comment, which is ignored by Silver.

The required files (HEX file, A2L file, and MAP file) are firstly defined in lines 2-4. The MAP file is optional. If it is given, the specification file can directly use the symbolic names for the functions, e.g. FUN_1. Otherwise, the addresses of the functions, e.g. 0x000667e8, must be used.

Lines 7-9 configure the startup code used by Silver to initialize the chip model. The functions which need to be run are listed in lines 12-13 using the names defined in the MAP file. The start time and running sequence of the functions are also defined here. Silver uses this information to emulate the RTOS. It is worth noting that the execution times of all initial and periodic functions listed afterwards must be a multiple of STEP_SIZE.

Finally, lines 16-17 are the inputs and outputs of the simulation obtained through the A2L file, and line 18 contains the parameter names in the DCM file. Alternatively, the interfaces of the simulation can also be listed separately as desired by referring to the MEASUREMENT or CHARACTERISTIC elements of the A2L file, e.g. input (VoltageKl40).

FIGURE 8 Specification file for a chip simulation in Silver.



```
# used files
hex_file(iARC.hex, PowerPC) # a PowerPC hex file
map_file(iARC.map)          # a TASKING or GCC map file
a2l_file(iARC.a2l)          # a ASAP2/a2l file

# startup code
chip_config(STEP_SIZE, 0.001)          # base clock in ms
chip_config(TEXT_START, 0xa0000000)  # Silver internal use
chip_config(USER_STACK, 0x40007ff0)  #  location of stack

# list of functions to run, in order of execution
task_initial(FUN_1, 0)          # run initially
task_periodic(FUN_1, 2, 0)      # run every 2 ms, offset=0

# interface
a2l_function_inputs(FUN_1)
a2l_function_outputs(FUN_1)
parameter(ABC)
```

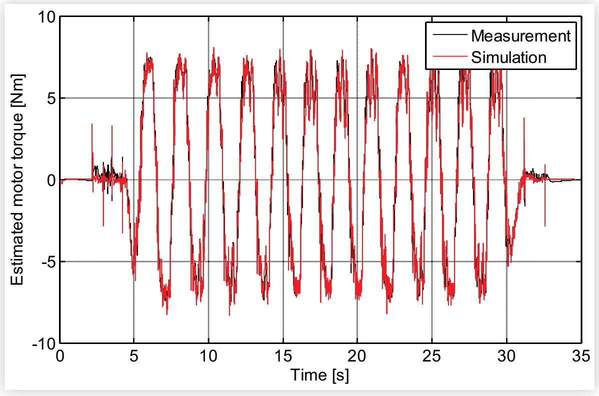
The peripherals of the ECU, such as AD/DA converters and amplifiers are not modeled by Silver. Functions accessing them are replaced by stubs. Usually the physical values measured by sensors, as well as physical target values for actuators are exchanged with the simulation platform. Functions that need to be stubbed are identified by off-line analysis of the provided binary, or catching access to peripheral registers in trial runs.

Validation

The virtualization of the ECU is typically validated against the behavior measured from the real object through an open-loop, where the inputs of the virtual ECU are driven by the inputs from measurement, and the outputs and intermediate variables of the virtual ECU are compared with the ones in the measurement. Figure 9 shows such an open-loop comparison of the motor controller. It can be seen that the simulation represents almost the same results as the real target. The small deviation between measured and simulated behavior is tiny and acceptable.

Once a chip simulation works as expected in the Silver ppcdebug/tcdebug module (based on the specification file with the name extension of .scs), the virtual ECU can be compiled to an executable Silver module (a Windows® DLL file), a MATLAB® S- function, or a FMU (Functional Mock-up Unit) [11]. This greatly speeds up simulation, typically by a factor of 100 [12]. The variety of the wrapping files also enables the application of the virtual ECU to be closed-loop simulated with physical models in other environments, e.g. MATLAB®/Simulink®, which is typically used for optimization.

FIGURE 9 Comparison of intermediate variable between virtual ECU and measurement.



Application

As mentioned previously, with the help of the virtual ECU it is possible to emulate the functional testing in a virtual environment. All the functions at the application layer of the ECU (in Figure 6) are virtualized according to the implementation method described in the section "Development of the Virtual ECU", i.e. extracting from the HEX file, A2L file, and a MAP file. A DCM file is additionally needed if the parameters in the function software need to be calibrated. All of these files are obtained easily because they are also needed for the real test on the bench or HiL (Hardware-in-the-Loop). Additionally, the testing environment such as restbus simulation and signal post-processing can also be got from the ready-made ones on test bench or HiL. This totally reduces the development effort of the virtual testing.

The following plots show some simulation results of the virtual testing. Figure 10 is the system's dynamic response to a step torque, where both torsion bars of iARC are fixed and only the electric motor rotates. From the comparison with the test bench measurements, it can be seen that the real behavior of system can be represented accurately through simulation.

The virtual test for a driving scenario, i.e. 18-Meter-Slalom, is also carried out. In this testing event, both torsion bars of iARC are separately effected by an external disturbance from the pulser system. These external disturbances are used to emulate the coupling rod displacements, which are caused by the roll forces during steering. Meanwhile, the electric motor of iARC is activated to track a desired torque trajectory, which is used to reduce the roll angle. As shown in Figure 11, it can be found that the virtual test has very similar results as the one from measurement on the test bench.

From the above comparisons between the virtual test and the real one, it can be seen that with chip simulation it is possible to take the system functional test from the test bench to the PC. The main purpose of the system functional test is to validate the actuator behavior with specified requirements. For example, the step response time to an applied actuator torque, the achievable relative dynamic stiffness of the actuator, and so on. Since chip simulation can represent accurate behavior as compared to the real one, it enables the

FIGURE 10 Comparison results of the testing under an actuator torque jump.

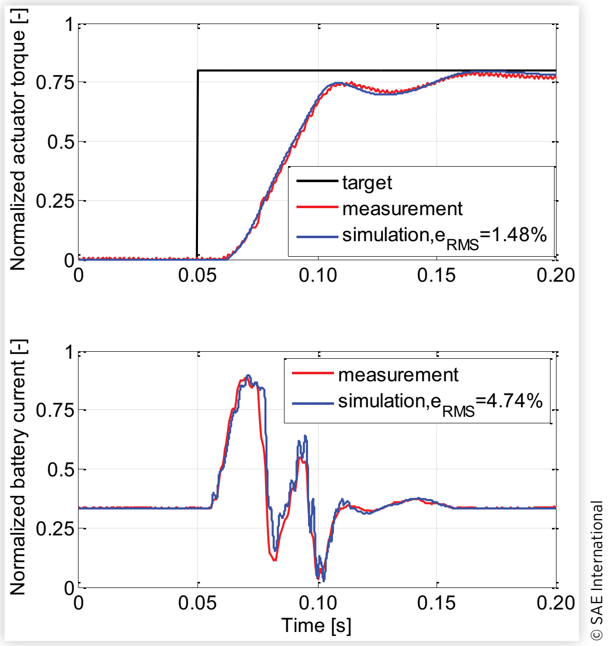
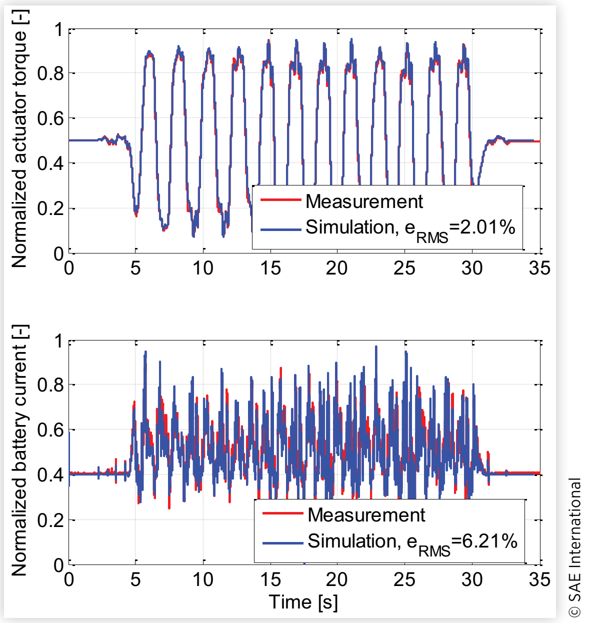


FIGURE 11 Comparison results of the testing with a driving scenario.



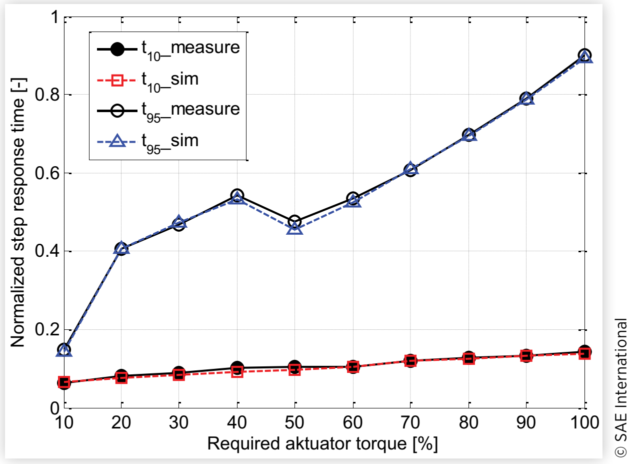
criteria evaluation based on simulation. For example, Figure 12 shows the step response time to actuator torques, where t_{10} and t_{95} denote, respectively, the response time when the actual actuator torque reaches 10% and 95% of the required one.

With this virtual simulation on a PC, the testing duration is reduced to 20% of the standard testing time on test bench. The cost for the implementation is significantly reduced. About 60% of the functional testing can be transferred to the virtual platform. The testing, such as that of the derating function, is currently planned to be virtually tested as well. Therefore a thermal model of the actuator is also needed.

Virtual testing enables the system function test to be realized in a fast and economical way, and the test results are 100% reproducible. All the internal signals from ECU and physical model can be continuously measured without any band-width limitation. Such a simulation can even be halted and stepped, which makes the debugging more convenient. Calibration tasks can also be moved from test bench to a readily available and economical PC platform.

The virtual testing with chip-simulation helps to easily get information of the system behavior, where the common use-cases and limit scenarios are traditionally validated in physical testing. A good correlation between simulation and testing helps to solve the trade-off in automotive projects to be fast and cost-effective without losing quality by not testing completely. Since the chip simulation is instruction accurate, it cannot be used to exactly predict execution time on the real target. Low-level processes, such as ECU communication, error storage and interrupt handling, are also not modeled in detail in such simulation environment, and are therefore beyond the scope of chip simulation based testing.

FIGURE 12 Step response to required actuator torques.



Summary and Outlook

In this work Schaeffler describes how to develop a cost-effective virtual testing approach for its system functional test. The virtual ECU can be set up through a HEX file, an ASAP2/A2L file, and a MAP file. The validation results show that this virtualization method allows the test to be taken from the real environment to a virtual one, which reduces the test process in a convenient way.

With this virtual testing method, some further work will be carried out. For example, temperature influence will be considered in the physical models to increase the test scenarios. A multi-objective optimization algorithm will be applied to determine the trade-off control parameters between dynamic behavior and NVH performance of iARC.

References

1. Schaeffler, *Product Information: Roll Stabilizer* (Germany: Schaeffler Technologies GmbH & Co. KG, 2014).
2. Mauss, J., “Chip Simulation Used to Run Automotive Software on PC,” in: *Embedded Real Time Software and Systems*, Toulouse, France, February 05-07, 2014.
3. Junghanns, A., Mauss, J., and Seibt, M., “Faster Development of AUTOSAR Compliant ECUs through Simulation,” in: *Embedded Real Time Software and Systems*, Toulouse, France, February 05-07, 2014.
4. Junghanns, A., Serway, R., Liebezeit, T. et al., “Building Virtual ECUs Quickly and Economically,” *ATZ Elektronik* 7:48-51, 2012.
5. Linssen, R., Uphaus, F., and Mauss, J., “Simulation of Networked ECUs for Drivability Calibration,” *ATZ Elektronik Worldwide* 11(4):16-21, 2016.
6. Gaspar, R., Wiesner, B., Bauer, G., “Virtualizing the TCU of BMW's 8 Speed Transmission,” in: *10th Symposium on Automotive Powertrain Control Systems*, Berlin, Germany, September 11-12, 2014.
7. Holmes, T., Passerelli A., and Connor, J., "SoC Development and Prototype with VDK," in: *16th International Workshop on Microprocessor and SOC Test and Verification (MTV)*, Austin, TX, 10-14, 2015.
8. Liu, Y., Wu, W.H., Zhou, X.F., and Zhou, D., “A Novel On-chip Debug System with Quick All-registers Scan Chain Based on JTAG,” in: *Eighth International Conference on Solid-State and Integrated Circuit Technology Proceedings*, Shanghai, 1941-1943, 2006.
9. dSPACE, *Virtual Validation with dSPACE* (Germany: dSPACE GmbH, 2017).
10. Landersheim, V., Fischer, F., Möller, R., Huang, H., et al. “Digitaler Zwilling eines servohydraulischen Prüfstands zur Durchführung virtueller Prüfungen an mechatronischen Wankstabilisatoren,” in: *DVM-Tagung effiziente Auslegung und Absicherung in der Betriebsfestigkeit*, Germany, September 26-27, 2018.
11. Blochwitz, T., Otter, M., Arnold, M. et al. “The Functional Mockup Interface for Tool independent Exchange of Simulation Models,” in: *Proceedings of the 8th International Modelica Conference*. Dresden, 105-114, March 2011.
12. Qtronic GmbH, “Silver (2016), version 3.3,” Product Help, 2016.

ISSN 0148-7191