Faster Development of AUTOSAR compliant ECUs through simulation

Andreas Junghanns, Jakob Mauss, Michael Seibt

Faster Development of AUTOSAR compliant ECUs through simulation

Andreas Junghanns, Jakob Mauss
QTronic GmbH
Alt-Moabit 92, D-10559 Berlin

Michael Seibt
Dassault Systèmes Deutschland GmbH
Joseph-Wild-Str.20, D-81829 München

Abstract **-** Virtualization allows the simulation of automotive ECUs on a Windows PC executing in a closed-loop with a vehicle simulation model. This approach enables moving many development tasks from road or test rigs and HiL (Hardware in the loop) to PCs, where they can often be performed faster and cheaper. Technical challenge: How to port ECU tasks and basic software to Windows PC with reasonable effort, so that key development tasks can be performed on a PC, without the need of accessing real hardware such as vehicle prototypes, test rigs or HiL facilities. This paper presents a new solution for the use case of ECUs developed within the emerging AUTOSAR standard: First, the AUTOSAR authoring tool AUTOSAR Builder (Dassault Systèmes) is used to design the application software and system aspects of a single ECU or an distributed embedded system which is then stored as AUTOSAR XML descriptions. The application code can either be developed in the AUTOSAR Builder environment or auto-generated by tools such as Embedded Coder (MathWorks), TargetLink (dSPACE) or Ascet (ETAS). Once tested in AUTOSAR Builder, selected software components or compositions can be exported including an AUTOSAR OS (Operating System) and RTE (Run-Time Environment) as an FMU (Functional Mockup Unit). FMU [4] is a new exchange format for models that has been developed in the EU-funded MODELISAR project (2008 - 2011) and since then gained considerable acceptance across multiple industries and tools. The FMU can then be imported into the virtual ECU tool Silver (QTronic), where it can be co-simulated with vehicle models originating from a wide range of simulation tools, including Dymola, SimulationX, MapleSim and AMESim. Vehicle models are again provided as FMUs, or via proprietary binary export formats, typically Windows DLLs. Tools for measurement and calibration such as CANape (Vector Informatik) or INCA (ETAS) can then be connected to the virtual ECU running on PC, to directly measure or tune its parameters, like an engineer would do in a real car. Virtual ECUs are also used to move testing activities from test rigs and HiLs to Windows PC.

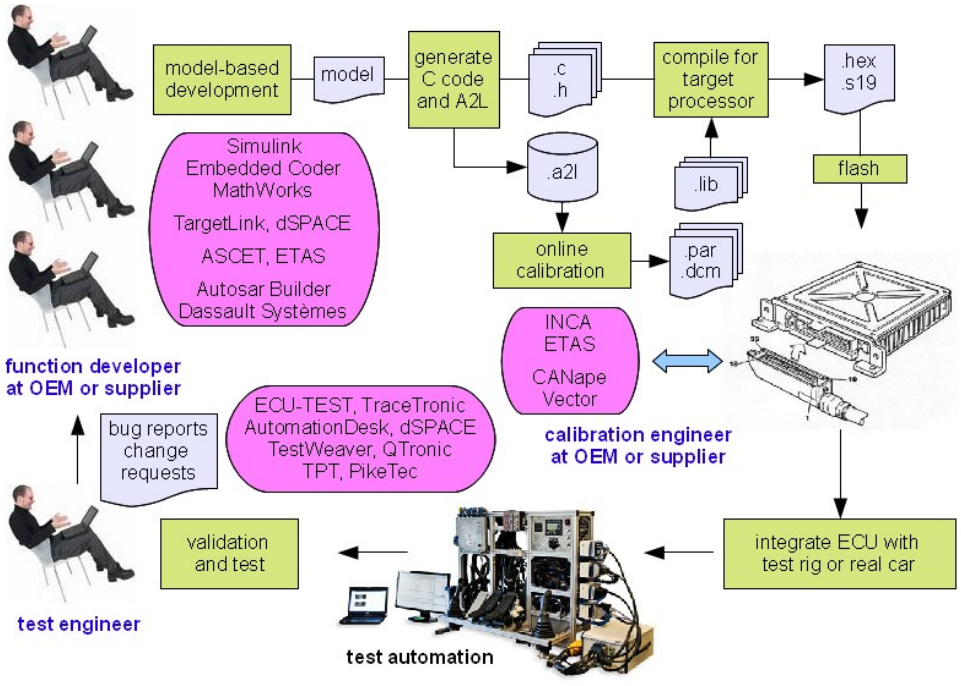

Fig 1: Development of automotive ECUs

1. Introduction

Software for automotive ECUs is jointly developed by OEMs and their suppliers as shown in Fig. 1. Typically, a team of function developers uses a model-based tool chain to develop a model of the ECU and to generate C code from that. The resulting C code is then compiled for the target processor, and the resulting ECU is validated and tested using test rigs, HiL systems, and road tests. Test and validation results are fed back to the developers, which closes the development cycle. See [10] for details and typical numbers.

This process, although standard in the automotive industry today, has two major drawbacks:

- a single iteration takes days or weeks, that is, feedback reaches developers late
- the process depends on prototype vehicles and test rigs. These are typically rare resources during development. Their limited availability causes additional delays during development.

This paper presents a tool chain that supports an improved development process. The key idea is to provide development engineers with virtual ECUs, which enables them to move many development tasks from test rigs and HiLs to Windows PC. As experience shows,this helps to shorten development cycles (down to 5 minutes for an incremental build) and to reduce the critical dependency to real hardware. We focus here on AUTOSAR projects, although the method does not depend on the framework (hand coded, model-based, AUTOSAR) used to develop the ECU.

The remaining paper is structured as follows: In the next section, we summarize various benefits of a virtual ECU for automotive development, with references to real applications. Section 3 explains how to build a virtual ECU for development projects that use the emerging AUTOSAR framework and section 4 demonstrates how to use the resulting virtual ECU on Windows PC to speed up development.

2. Virtual ECUs in the development process

Simulation has great potential to improve the development process for ECUs. It helps to move development tasks to PC, where they can often be performed faster, cheaper or better in many respects. In general, the speed-up generated with virtual ECUs is the product of three independent factors

- **Parallelization**: Virtual ECUs help to split development into parallel threads. With virtual ECUs, many development tasks can be moved from shared resources (vehicles, test rigs, HiL) to PC, where engineers can work in parallel, without blocking each other.
- **Faster feedback on system level**: On a PC, a development engineer can rebuild the entire ECU within 5 minutes after modification of a software module, thanks to incremental build, and test drive the result in a simulated environment. This helps detecting problems early on the developer's PC, and decreases the number of problems that show up late, when integrating all modules. As experience shows, such early checks speed up development.
- **Independence of real time simulation**: A virtual ECU might run on PC many times faster than in real time. When used in combination with test automation, a simple PC gives a higher test throughput than a considerable more expensive HiL test system. S. Gloss et. al. [1] reports how this effect is exploited to test 200 variants of a transmission controller.

Other points that illustrate the benefit of virtual ECUs:

- On a PC, an engineer can easily "freeze time", i. e. stop simulation and inspect the call stack and all variables of a virtual (i. e. simulated) ECU without bandwidth limitation and repeat a simulation deterministically as often as needed. In contrast, a real ECU being used in a HiL environment or test rig must run in real time. Stopping and stepping is impossible then, or requires considerable extra effort, e. g. based on the JTAG debug interface. Exact reproduction of experiments is difficult or impossible on a HiL or test rig, due to non-deterministic effects.
- Having a PC-based solution, a calibration tool like INCA (ETAS) or CANape (Vector) can be connected to a virtual ECU via XCP to measure into a running simulation and to tune characteristics online. In this way, many parameters of an ECU can be tuned using a relatively cheap and highly available PC platform, without blocking rare and more expensive resources like physical prototypes and test rigs.
- A virtual ECU runs independent from real time and can hence also easily be coupled with very time consuming simulations, such as combustion simulation. This is impossible for test rigs or HiL systems, where all parts of the simulation must run in real-time.

To exploit these and other benefits when developing an ECU, the ECU must first be ported to a PC. This is typically done based on the C code of the ECU, which is either hand-coded, or generated by tools such as Ascet (ETAS), TargetLink (dSPACE) or Embedded Coder (MathWorks). For example, QTronic's virtual ECU tool Silver [2] provides a framework to

- Compile given ECU tasks for Windows PC,
- Emulate the underlying RTOS and other services (CAN, XCP),
- Execute the resulting virtual ECU in a closed-loop with a simulated vehicle.

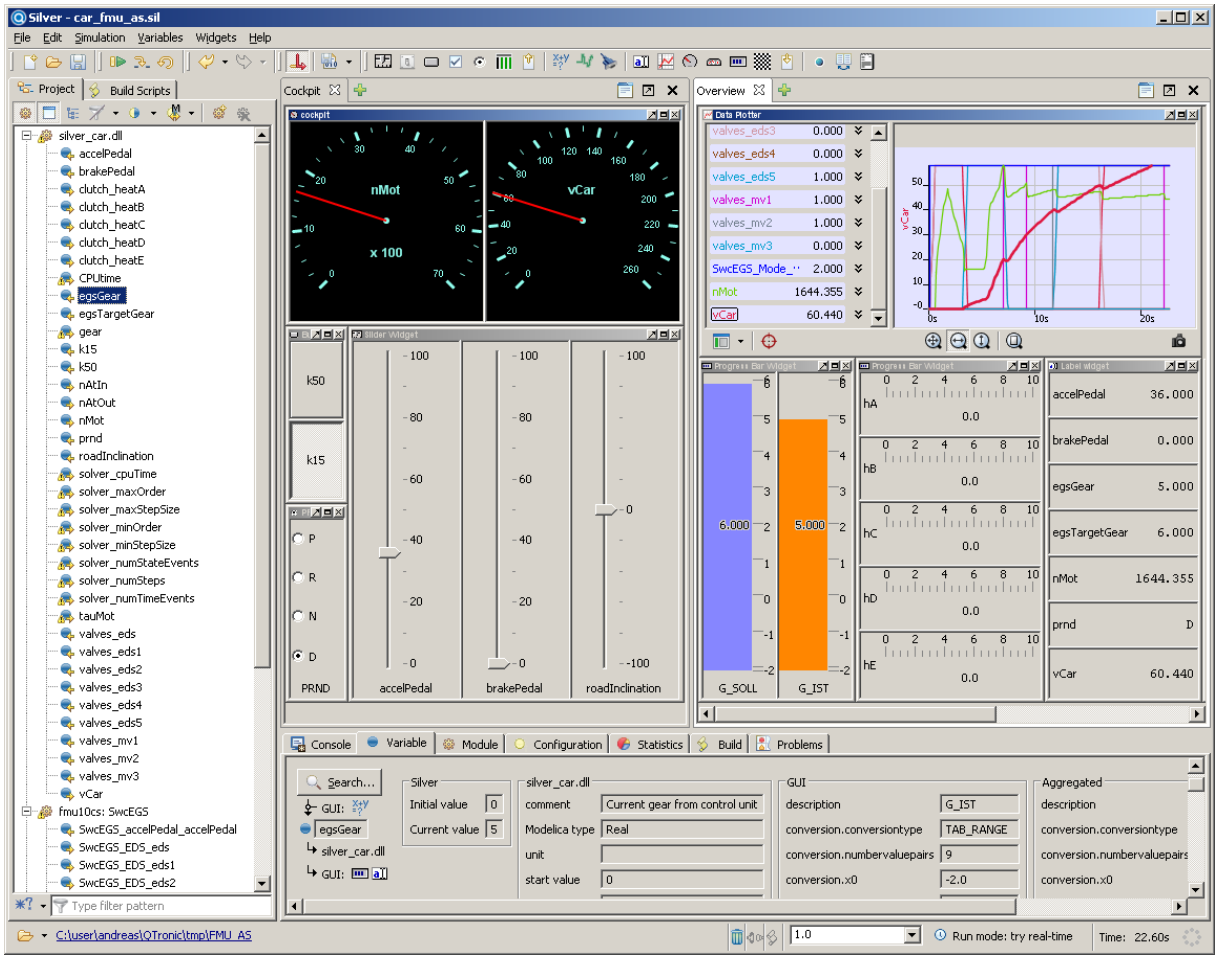

Fig 2: Simulation of an AUTOSAR Software Component in Silver

Typical applications are [1,2,3,4], where a virtual ECU is used to develop the controller for an automatic transmission. For closed-loop simulation, vehicle models can be imported from many simulation tools into Silver, including MATLAB/Simulink, Dymola, SimulationX and MapleSim, e.g. through the FMI (Functional Mockup Interface) format for model exchange [5].

C code is not always available for implementing a virtual ECU, for example when all or major parts of the ECU have been developed by a supplier and the OEM that is interested in building a virtual ECU has therefore no access to the C source code. Silver supports such cases with a chip simulator [6, 10]. The chip simulator takes the binary code for the target processor (currently TriCore or PowerPC) and simulates the execution of the instructions by the target processor on a Windows PC.

3. Designing and testing a Virtual ECU with AUTOSAR Builder

An AUTOSAR authoring tool such as AUTOSAR Builder allows the modeling of a single ECU or a distributed embedded system based on templates defined by the AUTOSAR standard. To ensure the validity and completeness of an AUTOSAR description, consistency or design rule checks have to be applied iteratively. Once this step has been passed, unit-testing can be performed by simulating single Software Components (SWCs) or even full compositions (CSWCs) that also take a real AUTOSAR OS and RTE into account. The application code (which can be manually coded or auto-generated) needs to be provided as source or compiled. In addition the test vectors need to be defined through a dedicated environment or can be imported. Everything else, including the OS and the RTE configuration and the Testbench is generated automatically. Once the SWCs or CSWCs have passed the tests in simulation, they can be exported as FMUs including the AUTOSAR OS and RTE.

Fig 3 shows a screen shot of AUTOSAR Builder, with an AUTOSAR software component EGS1 (a simple controller for a 6-speed automatic transmission) loaded. AUTOSAR Builder provides means to export this controller as an FMU for Co-Simulation or Model Exchange [5].

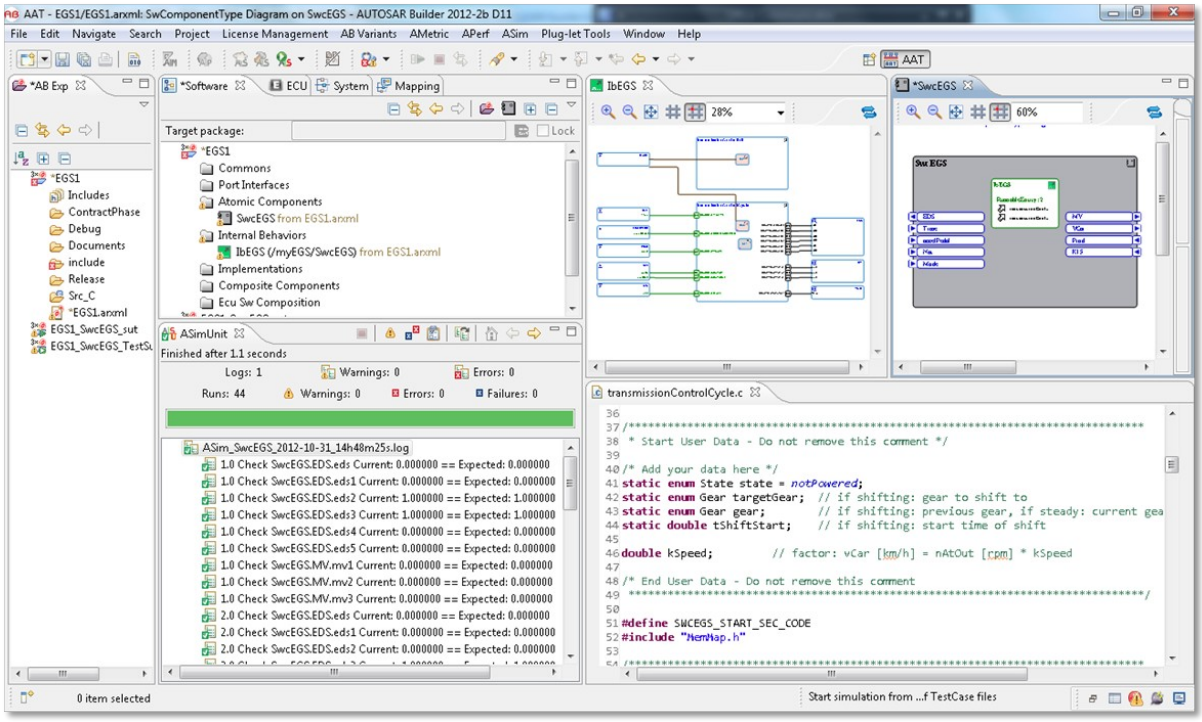

Fig 3: AUTOSAR Builder in simulation mode

4. Running a simulation in Silver

A virtual ECU exported from AUTOSAR Builder as FMU can be loaded into the Silver runtime environment, which runs on Windows. Fig. 2 shows a screen shot of Silver with the transmission controller EGS exported with AUTOSAR Builder loaded as virtual ECU. Besides the virtual ECU, also a vehicle model developed in Dymola / Modelica has been loaded into Silver, which enables closed-loop execution of the transmission controller. The tree shown on the left of the Silver window shows all loaded modules, and their input and output variables. In real projects, this tree might contain hundred thousands of scalar variables. Silver has been optimized to efficiently deal with such large models. The top-right part of the Silver window contains the plotters, gauges, sliders, and push buttons configured by the development engineer to interact with the vehicle model. In the case shown here, the engineer can start the engine, switch the PRND lever to any position, accelerate and brake using sliders, and watch the temperatures of the transmission brakes and clutches, and other key variables of the vehicle.

Once loaded into Silver, the services shown in Fig. 4 are available to implement various engineering tasks. Example uses cases are:

- Connect INCA (ETAS) or CANape (Vector) to the virtual ECU via the extended CAN calibration protocol (XCP, an ASAM standard) to diagnose a running simulation and to tune software parameters during simulation. This approach allows the expertise of calibration engineers with INCA or CANape to be also available in the PC environment.
- Import a vehicle model from simulation tools such as Dymola, SimulationX, MapleSim, AMESim, Simulink, GT-Power, or axisuite.
- Connect a test automation solution to Silver, to test the ECU on a PC as shown in [1] and [11].
- Connect Visual Studio Debugger to the virtual ECU, define code or data breakpoint and debug the virtual ECU at the code level, e.g. to fix problems found by test automation. Virtual ECUs running in a fully simulated environment on PC are particularly attractive here, because the corresponding step-debug functionality is much harder or impossible to achieve in a real-time environment.
- Drive the virtual vehicle interactively on PC to explore behaviour of the virtual ECU, without limitations of how much or what can be measured during simulation.
- Drive the simulation using measured data, available as MDF, DAT or CSV file. This can be used to implement open-loop tests of the virtual ECU. In simple cases, this does not even require a vehicle model: the virtual ECU is then directly driven by measurements.
- Drive the simulation using a Python script. Python scripting can be used to implement reactive tests (including e. g. 'do until' test steps) or an optimization loop that automatically calibrates ECU software parameters using mathematical optimization.

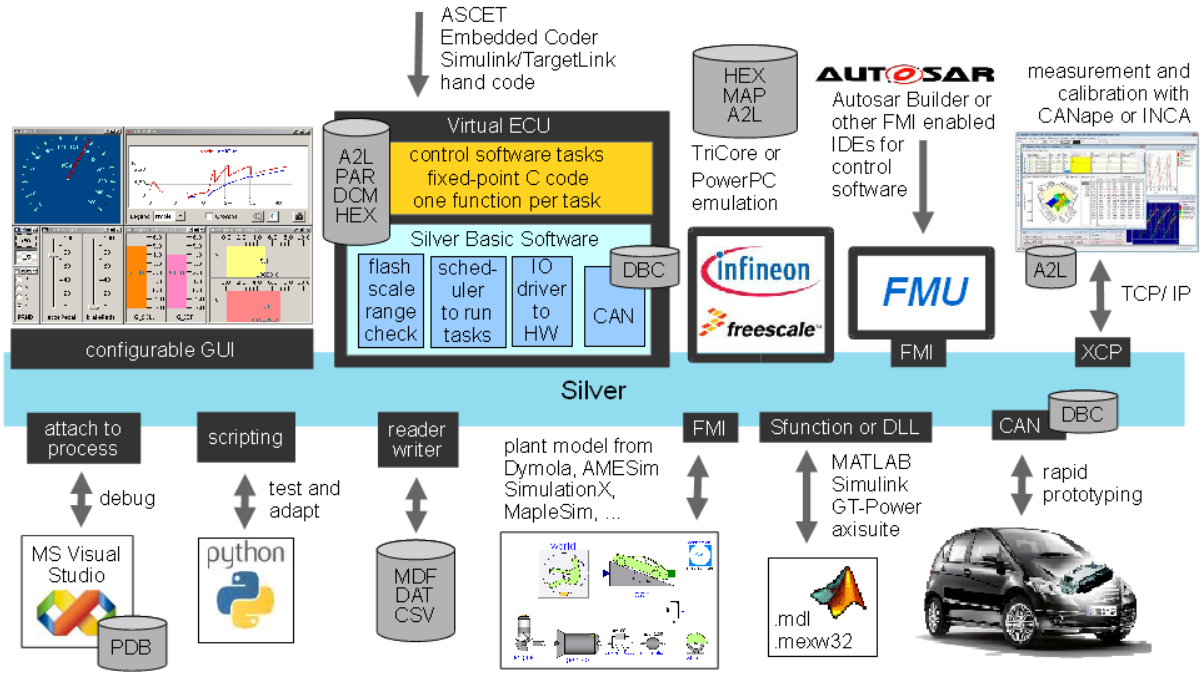

Fig 4: The Silver runtime environment for virtual ECUs

5. Conclusion

As demonstrated above, virtual ECUs help to move development tasks to a PC, which speeds up automotive embedded systems development tremendously. We have demonstrated a technical realization of virtual ECUs for the special case of AUTOSAR. Silver supports the same idea also for ECUs that run software generated with non-AUTOSAR tool chains. Use of virtual ECUs is increasing but not yet wide-spread in the automotive industry today. We expect to see significant growth of automotive applications in the coming years, mainly driven by the speed up of development that can be achieved as outlined in this paper.

References

[1] S. Gloss, M. Slezák, A. Patzer: Systematic validation of over 200 transmission variants. ATZ elektronik 4/2013. August 2013. http://www.qtronic.de/doc/ATZe_04_2013_en.pdf

[2] A. Junghanns, R. Serway, T. Liebezeit, M. Bonin: Building Virtual ECUs Quickly and Economically, ATZ elektronik 03/2012, June 2012. http://www.qtronic.de/doc/ATZe_2012_en.pdf

[3] H. Brückmann, J. Strenkert, U. Keller, B. Wiesner, A. Junghanns: Model-based Development of a Dual-Clutch Transmission using Rapid Prototyping and SiL. International VDI Congress Transmissions in Vehicles 2009, Friedrichshafen, Germany, 30.06.-01-07.2009. http://qtronic.de/doc/DCT_2009.pdf

[4] M. Tatar, R. Schaich, T. Breitinger: Automated test of the AMG Speedshift DCT control software. 9th International CTI Symposium Innovative Automotive Transmissions, Berlin, 2010. http://qtronic.de/doc/TestWeaver_CTI_2010_paper.pdf

[5] T. Blochwitz, M. Otter et. al.: Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. 9th International Modelica Conference, Munich, 2012. http://www.ep.liu.se/ecp/076/017/ecp12076017.pdf

[6] M. Simons, M. Feier, J. Mauss: Using Chip Simulation to optimize Engine Control. 7th Conference on Design of Experiments (DoE) in Engine Development, Berlin, 2013. http://qtronic.de/doc/doe2013_chip_simulation.pdf

[7] http://www.autosar.org

[8] M. Seibt, A. Gauthier, Denis Laroudie: Start "virtual" to make it "real". Hanser Automotive, 02/2013

[9] http://www.3ds.com/products/catia/portfolio/geensoft/autosar-builder/xtmc=autosar_builder&xtcr=3

[10] J. Mauss: Chip simulation used to run automotive software on PC. ERTS-2014, Toulouse, 05 - 07.02.2014.

[11] M. Tatar, J. Mauss: Systematic Test and Validation of Complex Embedded Systems, ERTS-2014, Toulouse, 05 - 07.02.2014.