Virtualizing the TCU of BMW's 8 speed transmission

Rui Gaspar, Benno Wiesner, Gunther Bauer

Abstract

Virtualization allows the simulation of automotive ECUs on Windows PC in closed-loop with a vehicle simulation model. This has a great potential to improve the development process for ECUs, because it allows to move several development tasks from road, test rigs and HiL (Hardware in the loop) to PCs, where they can be performed faster and cheaper. In this paper, we report the implementation of this idea for the case of BMW's 8 speed transmission ZF 8HP. The technical challenge is: 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. Since different parties (OEM and suppliers) jointly developed the TCU, the protection of intellectual property (models and source code) required special attention as well.

1. Introduction: Simulation used to speed-up development

Automotive control software is of often jointly developed by an OEM and 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

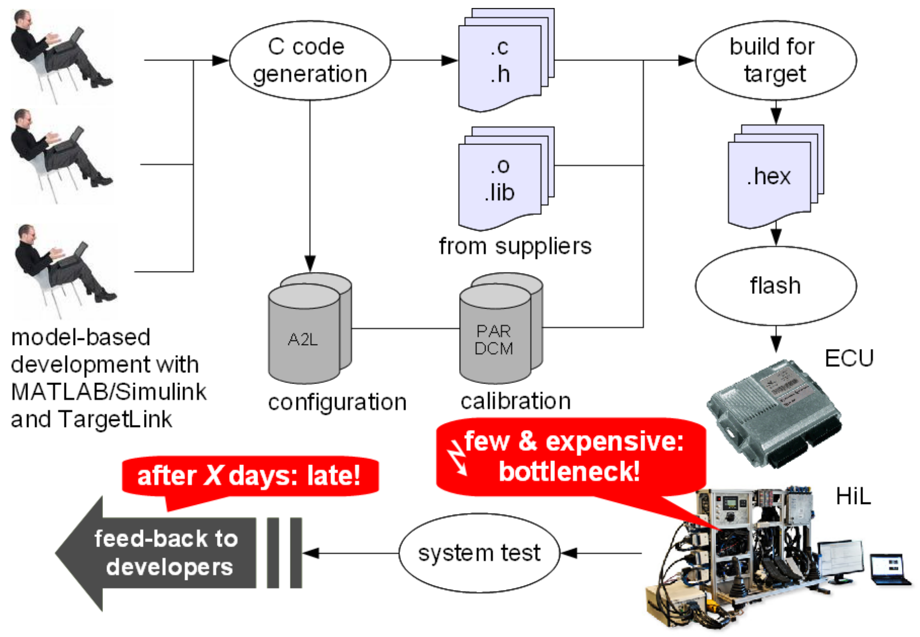


Fig. 1: Development process for automotive software

and tested using test rigs, HiL systems, and road tests. The test and validation results are fed back to the developers, which closes the development cycle. This process, although standard in the automotive industry today, has two major drawbacks:

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

This paper demonstrates how to improve the process. The key idea is to provide each development engineer with a *virtual ECU.* This can be simulated, calibrated and measured on the developer's laptop - either in closed-loop with a vehicle simulation model, or in real-time for rapid control prototyping. This way, more development tasks can be performed faster and cheaper on the developer's laptop. As experience shows, this helps to shorten development cycles and to reduce the critical dependency on scarce resources and real hardware.

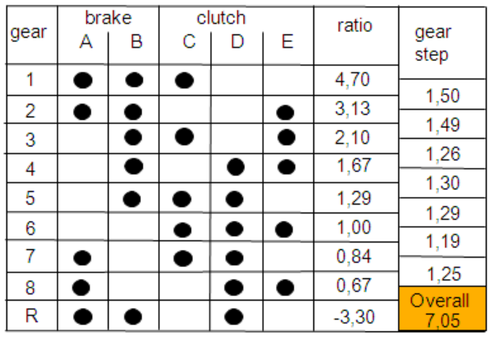
There are two main options to set up a virtual ECU on PC:

- *Re-host the native binary code using chip simulation.* The native ECU code (binary) is executed on PC by emulating the instruction set of the ECU processor [3]. This requires no access to the C code.
- *Re-target the C code*. Compile the C code of the ECU for execution on Windows PC. Obviously, this requires access to the C code to build a Windows executable or DLL.

The paper is structured as follows: In the next section, we describe how we have virtualized the transmission controller of the ZF HP8 automatic transmission, and list key differences between the real and the virtual TCU. Section 3 presents some applications of this virtualization. Section 4 concludes with a brief outlook on future work.

2. Virtualization of the ZF 8HP automatic transmission

The ZF 8HP is an eight-speed automatic transmission. The eight gears are implemented using four planetary gear-sets controlled by five hydraulically operated brake and clutch elements as shown in Fig. 2.



gear	brake		clutch			ratio	gear step
	A	B	C	D	E		
1	●	●	●			4,70	
2	●	●			●	3,13	1,50
3		●	●		●	2,10	1,49
4		●		●	●	1,67	1,26
5		●	●	●		1,29	1,30
6			●	●	●	1,00	1,29
7	●		●	●		0,84	1,19
8	●			●	●	0,67	1,25
R	●	●		●		-3,30	Overall 7,05

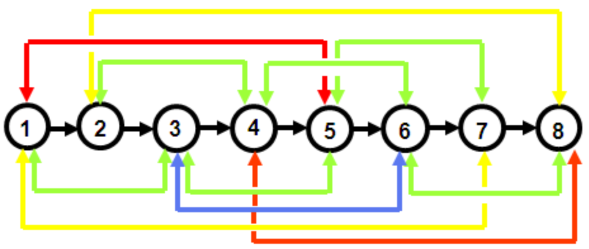


Fig. 2: Gears and possible gear shifts of the ZF 8HP 8-speed transmission

2.1 Virtualizing the transmission control unit

The shift strategy of a transmissions is often developed by an OEM, while the remaining control software comes from a supplier. For the ZF 8HP transmission considered here, about 30% of the shift strategy module is generated with TargetLink from a model developed with MATLAB/Simulink. The remaining part of that module is hand-coded using C.

We used the virtual ECU tool Silver [2] to re-target the entire control code as a Windows dynamic link library (dll). We could have used Silver's chip simulator [3] as well, but decided to use here the variant based on re-targeting for two reasons:

- Re-targeted code runs about 10 times faster on a PC than a corresponding chip simulation. Therefore, chip simulation is most useful in cases where re-targeting the C code is not possible.
- The control software for the 8HP transmission was partly developed using the virtual ECU tool SoftCar [1]. For that reason, a re-targeted version of the control software was already available as Windows libraries (.lib files created with MS Visual Studio).

Silver provides a framework for re-targeting control code to Windows. This framework, called Silver Basis Software (SBS), uses the same source files that are available during the normal ECU development, but *bypasses* the original basic software of the ECU with services supplied by Silver:

- The RTOS is replaced by a simple execution loop in Silver. This runs tasks either initially, periodically with defined offsets, or at certain events (interrupts).
- The DBC files describing the CAN communication of the ECU are used to emulate CAN processing.
- The ASAP2/A2L file describing all tunable and all measurable variables of the ECU is used to bypass analogue input and output processing of the ECU. For example, instead of simulating the low-level processing of a certain pulse-width modulated (PWM) signal that encodes the target current for a magnetic valve, we directly use the corresponding high-level current variable. This might be a 16-bit unsigned integer, which - after application of an associated scaling rule - represents the target current in Ampere. The scaling rule is part of the A2L description of the integer variable. Silver knows how to apply directly and how to invert the rule, and uses this to automatically convert the raw integer values to physically meaningful values during simulation (and vice-versa). This way, low-level processing of the basic software can be easily bypassed. In our example, the target current in Ampere is directly fed into the simulation model of the magnetic valve, which is part of the vehicle model described in section 2.3. A similar mechanism is used to bypass the low-level stages (AD conversion, signal filters) used for sensor value acquisition.

We used the above framework to re-target the AGS module of the TCU, based on the C code, both hand-coded and generated. This took us about three person days. To validate the result in Silver, the re-targeted AGS module was then simulated using measured inputs. The PC simulation commanded exactly the same gear shifts as those measured in the real vehicle. Encouraged by this quick success, we re-targeted then the entire TCU, based on the Windows library received from ZF.

2.2 Key differences between real and virtual ECUs

Fig. 3 a shows the execution of a 10 ms task on a real ECU. Every 10 ms, the task is started by the RTOS, and terminates in time, before the next cycle. At t = 23 ms, the task is interrupted by an event-triggered task. At that time, the context of the interrupted task (current register content or the CPU) is saved, the interrupt handler runs, and after that, the context is restored and the interrupted task is continued.

Fig. 3 b shows the execution of the same task on a virtual ECU as generated with Silver. The simulated time is a special variable of the simulation environment and it is not (necessarily) related to the real clock time. Execution of a task is assumed to take zero simulation time here: Simulation does not try to predict how long the execution of the tasks will take on the real ECU, and assumes infinitely fast execution instead. As a consequence, a running task cannot be interrupted on a virtual ECU, and all tasks run exactly as scheduled.

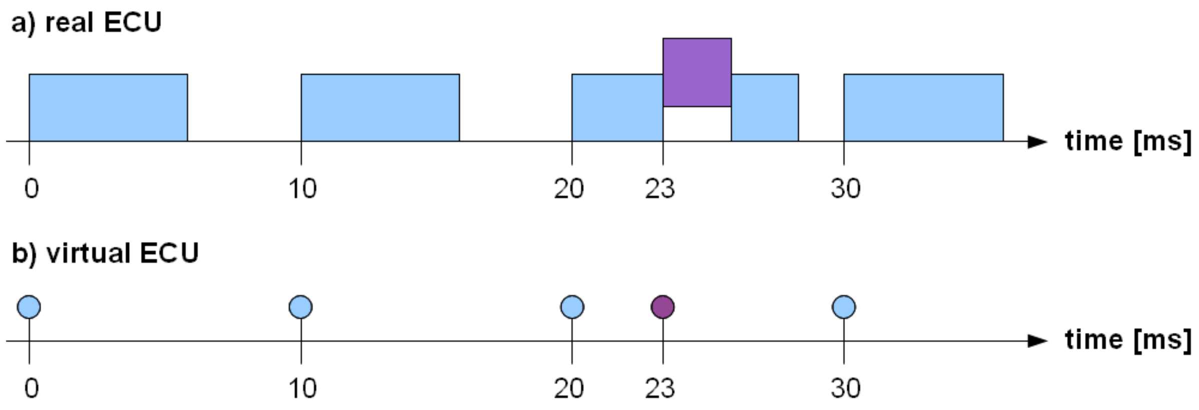


Fig. 3: Task execution on a real and on a virtual ECU

The above discussion makes clear that detailed timing and processor/bus load analyses cannot be performed with our virtual ECU environment. Such analyses must be performed with real ECUs and real bus-loads.

On the other side, certain analysis tasks cannot be easily performed using real ECUs - due to real-time limitations. Functional analysis is much easier performed in simulation:

- in contrast to a real ECU, a virtual ECU does not limit the number of variables that can be measured simultaneously. This eases detailed investigation of ECU behaviour. In Silver, all measurable variables listed in the A2L file are measured simultaneously during simulation.
- a real ECU limits the memory available for program code and data (typical size: 4 MB), while a virtual ECU might use the entire address space (typical size: 4 GB). With a virtual ECU, new functions can be virtually explored without considering memory constraints.
- a real ECU *must* run in real-time, while a virtual ECU is free to run *faster* or *slower* than real time. Execution might even be halted to step, debug and insert faults. This independence of real time enables coupling of virtual ECUs with very fast (much faster than real time) or very detailed (not real-time cap-

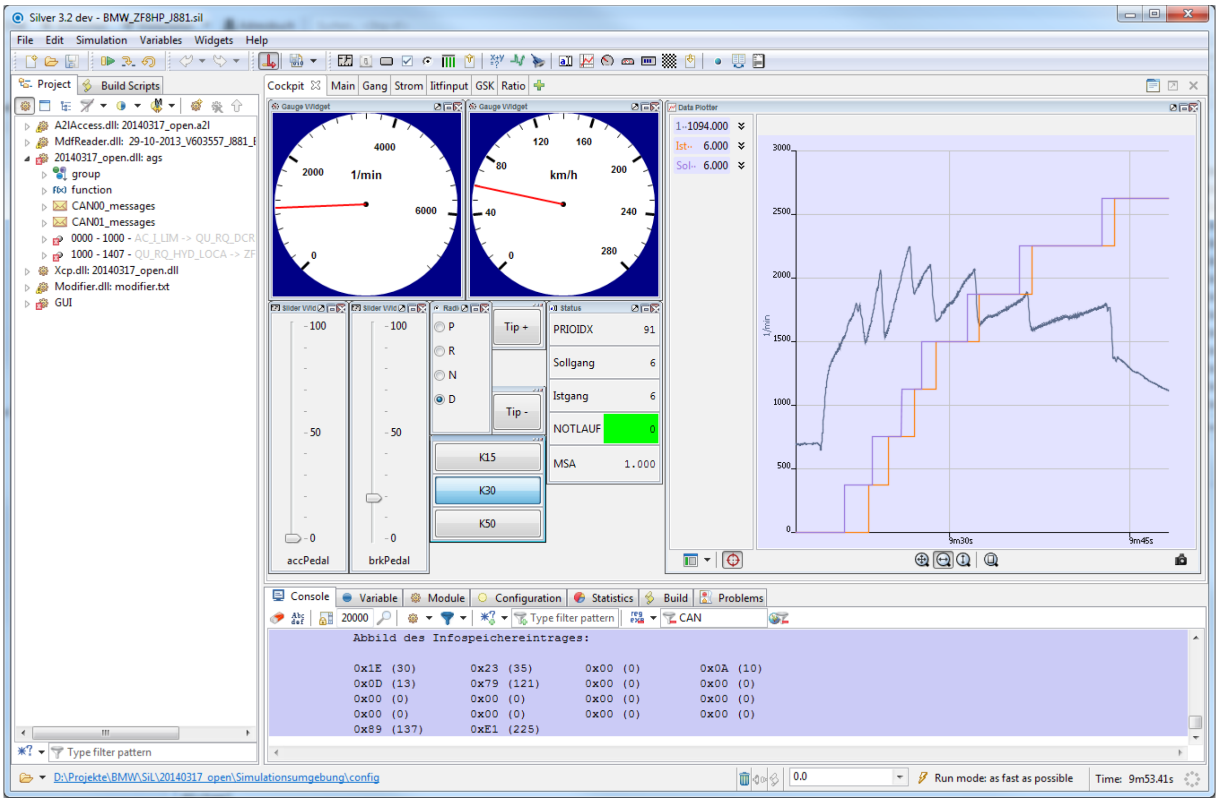
able) plant models, or coupling with a source-level debugger (breakpoints, stepper).

To summarize: for some development tasks (e.g. measuring execution times), it is better to use a real ECU; for other tasks (such as debugging or running functional tests or optimization procedures), it is more convenient to use a virtual ECU.

2.3 Closing the loop: The vehicle and the transmission model

After having virtualized the entire TCU code, we coupled the virtual ECU with two existing plant models provided by ZF: a vehicle model that includes a detailed model of the 8HP transmission, developed with Modelica and exported as *FMU for Co-Simulation 1.0* [4], and a model of the transmission hydraulics developed independently with MATLAB/Simulink and exported as mew32 file. Both models run in Silver with communication period of 1 ms.

The added vehicle model enables closed-loop simulation of the TCU on PC, including the validation of calibration data: Silver provides functions to 'flash' calibration files (ETAS DCM, Vector PAR, Intel HEX) into the virtual ECU.



Silver 3.2 dev - BMW_ZF8HP_J881.sil
File Edit Simulation Variables Widgets Help
Project
Build Scripts
A2lAccess.dll: 20140317_open.a2l
MdfReader.dll: 29-10-2013_V603557_J881_I
20140317_open.dll: ags
group
function
CAN00_messages
CAN01_messages
0000 - 1000 - AC_I_LIM -> QU_RQ_DCR
1000 - 1407 - QU_RQ_HYD_LOCA -> ZF
Xcp.dll: 20140317_open.dll
Modifier.dll: modifier.txt
GUI
Cockpit
Main
Gang
Strom
Istfinput
GSK
Ratio
Gauge Widget
1/min
km/h
Data Plotter
accPedal
brkPedal
P
R
N
D
Tip +
Tip -
K15
K30
K50
Status
1/min
9m30s
9m45s
Console
Variable
Module
Configuration
Statistics
Build
Problems
Type filter pattern
CAN
D:\Projekte\BMW\SiL\20140317_open\Simulationsumgebung\config
Run mode: as fast as possible
Time: 9m53.41s

PRIOIDX	91
Sollgang	6
Istgang	6
NOTLAUF	0
MSA	1.000

```
Abbild des Infospeichereintrages:

0x1E (30)      0x23 (35)      0x00 (0)       0x0A (10)
0x0D (13)      0x79 (121)     0x00 (0)       0x00 (0)
0x00 (0)       0x00 (0)       0x00 (0)       0x00 (0)
0x00 (0)       0x00 (0)       0x00 (0)       0x00 (0)
0x89 (137)     0xE1 (225)
```

Fig. 4: Open-loop simulation of the ZF 8HP transmission in Silver

3. Return on invest: Applications of the virtualized transmission

Virtual ECUs do not come for free: As shown above, setting up an ECU simulation requires certain investments and efforts, which must be justified by later applications that return these investments.

The PC simulation of the 8HP transmission system can be used as follows:

Open-loop analysis of measurements on PC: Use measurements (e.g. a MDF or DAT file) taken on the road or on test rig to drive the virtual ECU on PC. This way it becomes possible to look at all TCU variables in detail (lets say, 100.000). This gives a fairly complete picture of the ECU behaviour. This usage of a virtual ECU is the easiest to implement because it does not require any vehicle model.

Debugging on C source level: Attach the MS Visual Studio debugger to the virtual ECU running in Silver to debug problems on C code level. On a real ECU, a runtime exception like an integer *division by zero* or a *memory access violation* will typically trigger an ECU reset. These kinds of problems are difficult to catch and analyse in real-time environments, e.g. on a HiL system or on the road. With the virtual ECU, there is no real-time constraint: Simulation can be stopped to inspect variables, call stack, to modify variable settings and to resume simulation. This works equally well with open-loop simulation (as above) and with closed-loop simulation.

Pre-calibration: Closed-loop simulation of TCU and vehicle model on PC allows us to start calibration much earlier during development, before transmission or vehicle prototypes or real ECU hardware are available. This way, better initial settings for tunable parameters of the TCU can be provided, which speeds up later calibration stages using real vehicles. Silver virtual ECUs support coupling with calibration tools such as INCA (ETAS) and CANape (Vector) via XCP on TCP/IP. This means that calibration engineers can use the same tooling for pre-calibration on PC and later for calibration on the road.

Virtual HiL on a laptop: System test with virtual ECUs [5]. The idea is to move certain tests from the expensive HiL environment to cheap and highly available PCs, using closed-loop simulation with a virtual ECU. Based on the quality of our current virtualization, we estimate that this can be done for most of our test cases. The tests using the virtual platform has additional advantages that are not achievable using real ECUs and HiLs:

- All internal measurable ECU and plant model signals can be continuously measured without band-width limitations.
- The test results are 100% reproducible. Even software failures such as division-by-zero, access violations, that reset the real ECU, can be easily reproduced and analysed.
- The simulation can run faster than real-time. Several simulations can be easily conducted in parallel on several PCs or on the same multi-core PC.
- The simulation can use, when necessary, detailed plant models that are not able to run in real time on a HiL – all physical effects that are relevant for the system can be simulated for a complete functional validation.
- All development, test and calibration engineers involved in a project at the OEM or at the supplier site can load and run a virtual system simulation on their laptop. Integration and system tests are faster and more comprehensive due to this parallelization of work.

4. Conclusion

We reported how we virtualized the controller for the ZF 8HP transmission for closed-loop simulation on Windows PC, using the virtual ECU tool Silver. We intend to improve and to extend this set-up in order to perform more and more development tasks faster and cheaper on standard Windows PCs.

References

[1] Bieber, E.; Gillich, U.; Neumann, M.; Paulus, C.; Welt, C.: Systematische Absicherung von Steuerungssoftware für Hybridsysteme bei ZF. In: Simulation und Test für die Automobilelektronik III, pp. 333, expert Verlag, 2010.

[2] Junghanns, A.; Mauss, J.: Faster Development of AUTOSAR-compliant ECUs Through Simulation. Embedded Real Time Software and Systems. ERTS-2014, Toulouse, 05 - 07.02.2014. https://qtronic.de/doc/ERTS_2014_autosar_sil.pdf

[3] Mauss, J.: Chip Simulation used to Run Automotive Software on PC. Embedded Real Time Software and Systems. ERTS-2014, Toulouse, 05 - 07.02.2014. https://qtronic.de/doc/ERTS_2014_chip_simulation.pdf

[4] FMI for Co-Simulation 1.0, released 12.10.2010, see https://fmi-standard.org

[5] Tatar, M.; Mauss, J.: Systematic Test and Validation of Complex Embedded Systems. Embedded Real Time Software and Systems - ERTS-2014, Toulouse, 05 – 07.02.2014. https://qtronic.de/doc/ERTS_2014_TestWeaver_Paper.pdf