

Chip simulation of automotive ECUs

Ja ob ! auss, ! att"ias Simons

Abstract

Modern ECUs contain ten thousands of engine parameters that need to be tuned. Calibration of all these parameters is time consuming and complex. Simulation on a PC would help to automate and speed up the calibration process, in particular if simulation runs much faster (e.g. 20 times* than real-time). However, engine calibration is typically performed by an OEM, while the ECU code is provided by the supplier of the ECU. Therefore, the OEM is typically unable to set up a ECU simulation based on the original ECU code of the ECU. Instead, to set up a simulation, time consuming and error prone reverse engineering is needed to develop an equivalent model of the ECU function of interest. To improve this situation, we have integrated a chip simulator into the virtual ECU tool Sil-er. This is used to simulate ECU files compiled for various targets directly on the PC. Simulation requires

1. a ECU file that contains program code and parameters of the simulated functions
2. start addresses of the functions to be simulated
3. an ASA (24a2l file that defines the conversion rules for the input, outputs, and characteristics, as well as corresponding addresses

The start addresses of functions can e.g. be extracted from a map file generated together with the ECU file. Sil-er uses the 24a2l file to automatically convert scaled integer values to physical values and vice versa during simulation. A various simulation can also be exported as S5un(tion)me',32 file* for use in MATLAB/Simulink. In a standard PC, the simulation runs about 70 times (S. of only simulating selected functions of an ECU, this is fast enough to run a simulation much faster than real-time. In this paper, we also report how such simulations are used today to support the development of gasoline engines at a smaller.

Introduction: Virtual ECUs in the development process

Simulation has great potential to improve the development process for ECUs. Simulation helps to move development tasks to the PC, thus saving time and resources.

- on a (\$, a &alibration tool li e 0; \$A)#/AS* or \$A ; ape)<e&tor* &an be &onne&ted to a -irtual #\$\$% -ia =\$ (to measure into a running simulation and to tune &"ara&teristi&s online. /"is , ay, many parameters of a #\$\$% &an be tuned using a relati-ely &"eap and "ig"ly a-ailable (\$ platform, , it"out

models can be imported from many simulation tools into Sil-er, including MATLAB/Simulink, dymola, Simulation= and MapleSim, e.g. through the SDF format for model exchange.

However, sometimes code is not available for implementing a virtual test. There are two main sources for such a situation:

- **Proprietary code**: All or major parts of the code have been developed by a supplier and the customer is not interested in building a virtual test. e.g. to support calibration, a task typically performed by an OEM. In such a case there is no access to the code.
- **Commercial code**: The code is available for purchase but the customer is not interested in building a virtual test.

The tasks of categories 1 and 3 typically depend on details of the particular chip, and on the hardware, architecture. In contrast, tasks of category 2 are fairly independent from such hardware-specific details. To simulate the code, it is therefore convenient to run only tasks of category 2. The required inputs for these tasks can either be taken from measurement files (open-loop simulation), or they are computed online by some plant model (closed-loop simulation), bypassing the tasks of category 1. Finally, the outputs of category 2 tasks can be directly compared to measurements (open loop) or fed into the plant model (closed loop), bypassing the category 3 tasks. The signal interface between categories 1-2 and 2-3 is typically well documented and available, e.g. from the MATLAB/Simulink file of the code.

This modelling strategy has a very good cost-benefit ratio. In order to simulate also the tasks of categories 1 and 3, one has to model hundreds or thousands of peripheral and chip-specific registers, and to build state-machine models for logic elements, such as counters, controllers. In principle, this is possible, e.g. with SystemC, but hardly justified by the added value, at least for the applications considered here.

Silver 2.0 uses a specification file similar to the .06 file used to configure the SH to specify, in a file, the tasks of a file to simulate. Silver automatically turns such a specification file into an executable Silver module (dll) or Shared Library. A typical specification file looks as follows:

```
01 # specification of sfunction or Silver module
02 hex_file(m12345.hex,  riore_1.3.1"
03 a2l_file(m12345.a2l"
04 map_file(m12345.map"      # a #S$%' or '( map file
05 frame_file(frame.s"      # assembler code to emulate * +S
0, frame_set(S -._S%/-, 10" # Silver step size in ms
01 frame_set( -2 _S #* , 0xa000000" # location of frame code
03
04 # functions to be simulated, in order of execution
10 tas5_initial(#6!7-_ini"
11 tas5_initial(#6!7-_inis8n"
12 tas5_triggered(#6!7-_s8n, trigger_#6!7-_s8n"
13 tas5_periodic(#6!7-_20ms, 20, 0"
14 tas5_periodic(#6!7-_200ms, 200, 0"
15
1, # interface of the generated sfunction or Silver module
11 a2l_function_inputs(#6!7-"
13 a2l_function_outputs(#6!7-"
14 a2l_function_parameters_defined(#6!7-"
```

The "as" # character starts a comment, and is ignored by Silver. The specification file first lists the required files (line 2-4). The map file is optional. If a map file is given, the specification file may use symbolic names for files.

emulation. For event triggered tasks, Sil-er offers two alternative event models. Line 12 shows a function that is executed times at each Sil-er step, where is the value of the input variable trigger_#6!7-_s8n at the beginning of the step. Typically, is 0 or 1 during simulation. Trigger values occur only, when more than one trigger event occurs during one step. Sil-er also offers a more accurate event model, that allows execution of an event triggered task at each event time, not just at the beginning of a step.

Finally, lines 16-19 define the inputs, outputs and parameters of the generated module or S5unit. On this case, we must reuse the interface of a S5U ; element of the a2l file, for a function called AB\$8#. It is also possible, to list individual variables here by name, as long as their properties (such as address, conversion rule, data type* are described in the a2l file.

In addition, the spe file offers means to specify

- properties of the =\$(emulation, if any, to support online calibration and measurement using tools such as 0 ; \$A and \$A ; ape
- data sections to be included into the generated Sil-er module or S5unit. /"is , ay, initial loading of the "e' file into simulated memory can be avoided, to speed up simulation.
- memory areas to be copied to other (faster* memory by the start-up code
- functions to be replaced by other functions. /"is , ay, a function called by a task of category 1 or 3 to access sensors or actuators can be replaced by a function that directly accesses a plant model or measured values instead.
- logging options, e.g. to track memory access during simulation

The Sil-er module or S5unit generated this , ay performs exactly the same computations on (\$, as on the real target, since the effect of every machine instruction on memory and flip registers is exactly simulated on (\$. +o , e-er:

- simulation is just instruction accurate, not byte accurate. /"is means, the simulation on (\$ cannot be used to exactly predict execution time on the real target. For example, pipeline effects of different access times to memory (e.g. fast on-flip CA! -s. external CA! * are not modelled.
- conceptually, simulated tasks execute infinitely fast. /"is means that the emulated C/ . S ne-er interrupts a task . /"e corresponding effects cannot be analysed using the generated model.
- Sili&on bugs are not simulated. If a compiler for the real target does not , or

instruction. It is also possible to set code and data break points, for example to pause a simulation, whenever a certain variable is accessed.

In order to measure the execution speed of this simulation, we have ported a complete #C% function implemented by different C functions that run initially, every 10 and 200 ms, and synchronous to the random shaft. The speed file is very similar to

! "#\$ %

Sil-er &an also turn a spe& file as des&ribed in se&tion 2.1 into a S5un&tion, i.e. a me', 32 file t"at runs in Simulin . /"is is parti&ularly interesting , "en using &"ip simulation to support automated optimi9ation of parameters, be&ause many optimi9ation tools are implemented on top of !A/6AB4Simulin . /"e generated S5un&tion a&&epts all &"ara&teristi&s listed in t"e spe& file as S5un&tion parameters. /"is ma es it easy to &onne&t t"e generated S5un&tion ,it" an optimi9ation pro&edure. 5or e'ample, t"e S5un&tion &an be &alled ,it" , or spa&e -ariables t"at are t"en automati&ally -aried by t"e optimi9ation pro&edure bet ,een S5un&tion &alls. /"e performan&e of a generated S5un&tion is again about 70 !0(S.

Applications of chip simulation

0n t"is se&tion, ,e s"ortly s et&" &urrent appli&ations of t"e presented approa&" at 8aimler.

"& % ' () *

8uring de-elopment of an engine &ontroller, a de-eloper mig"t ,ant to repla&e a &ertain fun&tion of t"e #\$\$% by its o ,n -ersion of t"at fun&tion, bypassing t"e original fun&tion. 5or real #\$\$%, t"is &an be done ,it" tools su&" as #+ . . HS)#/AS* or ;o-+oo s)A/0*. /"ese tools manipulate t"e original "e' file, su&" t"at t"e bypassed fun&tion is not e'e&uted any more, but [ust &alls t"e ne , fun&tion instead. /"e ne , fun&tion is e. g. de-eloped ,it" !A/6AB4Simulin in &onfun&tion ,it" a &ode generator and a &ompiler for t"e target pro&essor. /"is met"odology still re2uires a&&ess to t"e real #\$\$%: t"e manipulated "e' file needs to be flas"ed into t"e #\$\$%, and t"e #\$\$% needs to run t"e ne , fun&tion, su&" t"at its be"a-iour &an be assessed. 0n order to furt"er simplify t"e assessment of t"e ne , fun&tion, ,e e'e&ute t"e manipulated "e' file in Sil-er on (\$ using &"ip simulation as des&ribed abo-e. Su&" simulations are typi&ally dri-en open loop by measurement files) ! 85*.

/ "e pla&ing of bypass "oo s by dire&t manipulation of t"e "e' file is a mig"ty but error-prone tool. Sometimes a "oo ed fun&tion is not &alled at all or only some -ariables are o-er , ritten and some not. ;ormally, su&" errors are only dete&t&ed after

, it" ! A/6AB4Simulin . / "is "as been time & consuming and error prone. > e "a-e no, partially repla&ed t"ese "and-&oded models ,it" S5un&tions generated automati&ally by Sil-er from a gi-en "e' file. / "e generated S5un&tions proofed to run as fast as t"eir "and &oded & counterparts. / "e repla&ement of "and-&oded floating-point models by generated fi'-point S5un&tions raises t"e follo , ing problem: Some optimi9ation pro&edures re2uire gradient information to guide t"e sear&" for optimal parameter -alues. 5or e' ample, , "en sear&"ing for an t"at minimi9es) , t"e deri-ati-e is to be &omputed during optimi9ation for different -alues of . 5inite differen&es are often used "ere, i.e. is &omputed as)) | * -) ** 4 for small , say J 10^{-D}. Of is &omputed using &"ip simulation, and are often bot" mapped to t"e same integer, resulting in a 9ero gradient. As a &onse2uen&, t"e optimi9ation pro&edure is la& ing guidan&, and mig"t return a suboptimal solution. / "ere are also so-&alled deri-ati-e-free pro&edures for optimi9ation. . b-iously, t"ese are not affe&tet by t"e abo-e problem.

Conclusions

As demonstrated abo-e, an # \$ % " e' file &ompile for some target pro&essor &an be e'e&uted by t"e -irtual # \$ % tool Sil-er on > indo , s (\$, eit"er open-loop dri-en by measurements or in &losed-loop , it" a -e"i&le model. 8epending on t"e appli&ation, sele&tet # \$ % fun&tions are simulated, or nearly t"e entire # \$ %.

/ "is ind of simulation opens ne , possibilities to mo-e de-elopment tas s from road, test rig or +i6 to (\$s , "ere t"ey &an be pro&essed faster, &"eaper or better in some respe&t, , it"out re2uireing a&&ess to t"e underlying \$ &ode. 8aimler &urrently uses t"is inno-ati-e simulation approa&" to support &ontrols de-elopment for gasoline engines. . t"er appli&ations, su&" as online &alibration on (\$ -ia = \$ (seem to be doable as , ell.

! eferences

- A1B A. Jung"anns, C. Ser , ay, /. 6iebe9eit, ! . Bonin: Building <irtual # \$ %s @ui& ly and #&onomi&ally, A/K ele troni 0342012, Juni 2012. See , , , .A/Konline.de or "tp:42troni&.de4do&4A/KeL2012Len.pdf
- A2B +. Brü& mann, J. Stren ert, %. Heller, B. > iesner, A. Jung"anns: ! odel-based 8e-elopment of a 8ual-\$lut&" /ransmission using Capid (rototyping and Si6. 0nternational <80 \$ongress /ransmissions in <e"i&les 2009, 5riedri&"s"afen, : ermany, 30.0D.-01-0G.2009. "tp:42troni&.de4do&48 \$ /L2009.pdf
- A3B H. Cmp e)ed.*: 8esign of #'periments)8o#* in #ngine 8e-elopment - 0nno-ati-e 8e-elopment ! et"ods for <e"i&le #ngines. #'pert <erlag, 2011.
- A7B /. Blo&" , it9, ! . . tter et. al.: 5un&tional ! o& up 0nterfa&e 2.0: / "e Standard for /ool independent #'&"ange of Simulation ! odels. 9t" 0nternational ! odeli&a \$onferen&, ! uni&" , 2012.
- A?B System\$, 6anguage for System-6e-el ! odeling, 8esign and <erifi&ation, see , , , .system&.org
- ADB ! . /atar, C. S&"ai&" , /. Breitinge: Automated test of t"e A! : Speeds"ift 8 \$ / &ontrol soft , are. 9t" 0nternational \$ /0 Symposium 0nno-ati-e Automoti-e /ransmissions, Berlin, 2010. "tp:42troni&.de4do&4/est>ea-erL \$ /0L2010Lpaper.pdf

•

G0?7D Stuttgart