INTELLIGENT TEST-CASE GENERATION FOR AUTOMATED VALIDATION OF TCUs.

Lionel Belmon, Technical Director, Global Crown Technology, China
Yijia Xu, Software Engineer, DCT Project engineering dept., SAGW/SAIC group, China

ABSTRACT

SAGW is currently developing a new DCT and is applying an automated test method for validating the transmission control unit. The System Under Test (SUT) is a Simulink model containing the following parts, running in closed loop simulation:

- The TCU application software model
- A plant model of the DCT gearbox and vehicle

This SUT is a relevant object for testing because SAGW uses the application software model to perform code generation. The resulting c-code is the one which is downloaded to the TCU hardware in the car. Moreover, the plant model used is realistic enough to reproduce the main dynamics and behavior of the real vehicle and gearbox.

PRINCIPLES AND IMPLEMENTATION OF TESTWEAVER

The driving concept in TestWeaver is to automate the generation and the analysis of test scenarios. The SUT provided to TestWeaver is a black-box which only exposes pre-defined input/ouputs, so called instruments. The nature of the SUT does not matter and could be a Simulink model[1], a Virtual-ECU/SiL based simulation [2],[3] or a HiL based simulation. Determinism of the SUT is however expected and un-deterministic behavior is checked and reported. TestWeaver will generate and execute scenarios for the SUT and apply complex algorithms in order to maximize the reached states coverage. A schematic is provided in the following figure. Further details on TestWeaver can be found in [4].

For Simulink based SUT, TestWeaver provides a blockset containing the TestWeaver instruments. Instrumenting a Simulink for use with TestWeaver is done by adding and limits/overspeed are specified in the reporting instruments. We illustrate the implementation in the simulink models in the following figures.

Figure 1: Example of implementation of a Chooser for the gear lever control

Figure 2 : Example of implementation of a Simulink subsystem for reporting outputs to TestWeaver

Once instrumented, the SUT is then compiled with Simulink RTW as an executable program that contains the interface functions to TestWeaver. This executable program (.exe) is then called by TestWeaver to perform simulations. This allows high speed simulation and scenario generation, several times faster than real-time.

Since TestWeaver generates 1000's of scenarios, it is not practical to evaluate the scenarii results by hand. TestWeaver provides an automatic reporting/analysis system that needs to be configured according to our project requirements. The reporting system will explore and these variables allows to find issues in the gearshift logic such as a jammed/blocked gearshift due to a TCU bug.

Finally, we also want to assess how well the SUT has been tested. The test coverage is a complex metric that should be evaluated from different angles :

(1) Code coverage : Did we test all functions, branches ? Did we reach all meaningful input/output values ?
(2) System states coverage : Did we reach all possible gearshifts ? Did we consider all possible environment conditions ?

The code coverage and system states coverage are not equivalent: It is possible to reach a high code coverage without reaching a high state coverage for instance. For a TestWeaver/Simulink setup, the code coverage is usually ignored since the final production
c- 1 424.5ETBT1 e1 0fg8(f)-r] TJcoverag are note fen4(no)14(t)13(e)1.1 0 c1agg8(5E)-3(m)-3(e)erpd we o8

We give in the following paragraph an example of how a bug was found by TestWeaver and how it was solved. During

A correction is applied to this transition and we can replay again the scenario to make sure that the problem is fixed. The model is then compiled with RTW and we let TestWeaver generate again scenarios to make sure that this modification does not introduce side effects. A TestWeaver scenario test database can also be used for non-regression testing.

Using TestWeaver/Simulink to debug logical bugs is much easier than debugging on HiL because replaying a scenario with TestWeaver/Simulink will highlight on which line the stateflow jamming occurs. If a first guess is made on where the bug is, then replaying with TestWeaver/Simulink can easily confirm of infirm this first guess, without wasting time on adding flags, compiling and building model to hex, then flashing to TCU.

TESTWEAVER AND HiL SYSTEMS

Hardware

file.