

Vehicle System Control Software Validation for the Dual Drive Hybrid Powertrain

Yanan Zhao, Sada Srinivasaiah, Zhang Yan, Yulei Chen, Chat Nguyen and Ray Spiteri

Ford Motor Company

Copyright © 2009 SAE International

ABSTRACT

Through the use of hybrid technology, Ford Motor Company continues to realize enhanced vehicle fuel economy while meeting customer performance and drivability targets. As is characteristic of all Ford Hybrid Electric Vehicles (HEVs), the basis for resolving these competing requirements resides with its Vehicle System Control (VSC) strategy. This strategy implements complex high-level executive controls to coordinate and optimize the desired operational state of the major HEV powertrain subsystems. To ensure that the VSC software meets its intended functionality, a software validation process developed at Research and Advanced Engineering has been integrated as part of the vehicle controls development process.

In this paper, this VSC software validation process implemented for a next generation hybrid powertrain is presented. First, an overview of the hybrid powertrain application and the VSC software architecture is introduced. Then the software validation process is described, which includes unit testing, model-in-the-loop testing, hardware-in-the-loop testing and vehicle testing. After that, issues observed with the current validation process are identified, and an open-loop simulation versus target testing procedure is proposed to enhance the validation process. Finally, conclusions and opportunities for further improvements are summarized.

INTRODUCTION

Ford Motor Company continues to develop Hybrid Electric Vehicles (HEVs) to improve vehicle fuel economy and reduce emissions while still continuing to provide vehicle performance. One of the core technologies that make this all possible for HEVs is the design of the Vehicle System Control (VSC) strategy.

The VSC is a functional supervisory controller that coordinates the operation of the powertrain subsystems. The VSC's main hybrid control functions are to coordinate vehicle and powertrain operating modes, perform energy management to maximize fuel economy, enable regenerative braking and manage battery state of charge operation.

Currently in the automotive industry, with the advent of various alternate fuel vehicles, vehicle control software development faces an increase in complexity of controls design requirements within shrinking design cycle timing. This ever-increasing challenge demands improvements in the software development and validation process. To deliver high quality and timely control software, a model-based software design process using modeling, simulation and code generation tools has begun to be adopted in the automotive industry [1, 2]. With a model-based design process, executable models can be used as the basis for functional analysis prior to code generation. This design process provides engineers with the convenience of developing, simulating and debugging the software using a desktop simulation environment prior to hardware availability. This in turn enhances the model development time and reduces the necessity for hardware thus allowing for potential issues to be detected at earlier design phases. Furthermore, the process allows for the use of automatic code generation tools to implement the production controller software from these models. For model-based design, a software validation process can be defined as part of the control software development process residing in the right-side of the system V [1].

For a new next generation hybrid development vehicle, the VSC was implemented using a new structural design known as Integrated Modeling Environment (IME) [3]. IME uses a model-based design approach to provide the

ability to integrate, simulate, and develop vehicle controls with HEV plant models. With the use of Legacy Code Blocks, developed for Ford by The Mathworks used with Simulink, IME enables controls development in a C/Simulink mixed environment. For VSC development and validation, Ford Research & Advanced Engineering has developed a control system development process which consists of system requirements, controls algorithm design, controls implementation, unit testing, model-in-the-loop (MIL) testing, hardware-in-the-loop (HIL) testing and vehicle testing.

Using the newly developed VSC software validation process, software functions are tested gradually from simulation to real-time hardware implementation. Testing is performed, from unit level testing, to feature level testing and then to vehicle level testing. The unit testing focuses on independent functional tests on the lowest level software functions using an open-loop simulation strategy. MIL testing uses a platform for integration of the vehicle plant model with the developed control application. It allows closed-loop control simulation, so that features and vehicle level control functions can be tested before hardware is available. However, tests are not real-time, and the ECU hardware control module interface and communication are not represented in the MIL [4]. HIL testing in contrast tests the generated C code with ECU hardware in real-time and also provides the capability to test the I/O interface. Because the generated code used in the HIL and the vehicle are the same, HIL testing can significantly reduce the cost and usage of the vehicle while detecting potential issues at an early stage [5]. Vehicle testing is used in the last stage to validate interactions between networked ECUs, or to validate the system level function of an individual ECU considering the modeling limitation of the HIL vehicle plant model. During VSC development, it was observed that the generated code used in the HIL and vehicle did not behave exactly the same as the model used in the MIL simulation. To capture the difference, a Simulation versus Target (SVT) test was designed. This open-loop test successfully identified several implementation issues, and has been proven to be an effective enhancement as part of the software validation process.

In the following, an overview of the Dual Drive hybrid powertrain and the VSC software architecture are given first. Then the model-based VSC validation process is introduced. After that, the unit testing, MIL testing, HIL testing and vehicle testing are described respectively with regard to its test environment, test plan, test set up and test cases. Later, issues observed with the current validation process are identified, and the Simulation versus Target test is presented. Finally, conclusions are given with opportunities for further improvements.

OVERVIEW OF HYBRID POWERTRAIN AND VSC SOFTWARE ARCHITECTURE

Development of the software validation process was based on a new next generation HEV project developed

at Ford Research and Advanced Engineering. The following is a brief overview of this new HEV powertrain and its associated VSC software architecture.

DUAL DRIVE HYBRID POWERTRAIN – The Dual Drive hybrid powertrain architecture consists of a Crank Integrated Starter Generator (CISG) and an Electric Rear Axle Drive (ERAD). The major components of this system are shown in Figure 1. This hybrid powertrain has two propulsion torque paths. The first torque path is provided by the engine and the CISG electric motor delivering propulsion torque to the front axle through a transmission. For the second torque path, propulsion torque is delivered from the ERAD motor to the rear axle. Both electric machines are powered by a high voltage battery through an inverter. The Dual Drive HEV hybrid provides the capability for the vehicle to operate in several different modes of operation. The three main modes of operation of the hybrid powertrain are Electric Drive, Series Drive and Hybrid Drive. In Electric Drive, the vehicle is propelled electrically by the ERAD motor powered by the battery. The vehicle operates in this mode during low driver power demand situations when the battery is sufficiently charged. In Series Drive, the engine does not drive the wheels directly, but is used to provide power to the battery through the CISG. The battery then provides power to drive the ERAD motor for propelling the vehicle. In Hybrid Drive, the engine is the primary driving source while the ERAD motor can provide additional electric boost torque if necessary. Other modes exist, but they are non-driving modes and are not discussed here.

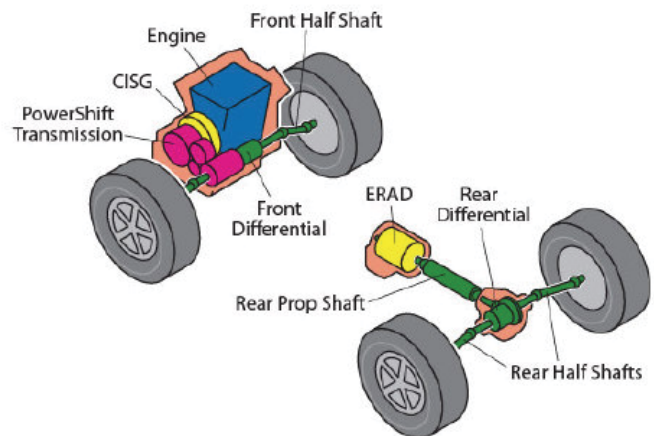


Figure 1 Dual Drive HEV System

Coordination and control for the hybrid powertrain is provided by the VSC. The VSC is a high level executive process which controls and determines the desired operational state of major vehicle powertrain control subsystems such as the engine, battery, motor, generator and brake systems. For the Dual Drive HEV project, a MicroAutoBox (MABX) from dSPACE was chosen as the rapid prototyping control module for implementation of the VSC.

VSC SOFTWARE ARCHITECTURE – The Integrated Modeling Environment (IME) is used for the VSC control software development [3]. This approach provides the flexibility to develop the VSC in a combined legacy C code and Simulink model-based environment. Using this approach development time is reduced and the system design can be proven out prior to development of a prototype hardware system. In addition, this environment can be further developed to generate production level code.

The top level of the VSC model has Input Organizer, Task Scheduler, Torque Manager (TM) core algorithm, and Output Organizer. The Input Organizer reads in TM required input signals and regroups the signals based on TM features. The Output Organizer regroups TM output signals and feeds the signals back to the TM or sends them outside the TM. The Task Scheduler is used to generate desired task rate triggers for the VSC features. The function-call generator blocks of Simulink generate triggers at 8ms, 16ms, 32ms, 48ms and 96ms task rate. The triggers are then grouped by features and sent to a signal bus from which each feature can obtain its own triggers at the feature implementation level as described below. The Task Scheduler controls the execution order of the VSC software.

The TM core algorithm is designed to meet VSC related High Level Functions (HLFs). A HLF is a collection of atomic functions that contains algorithms to achieve a task at the vehicle level. The TM consists of subsystems representing software features. A feature is a functionally cohesive unit which implements design requirements from one or more VSC high level functions. The features are organized based on the vehicle functionalities.

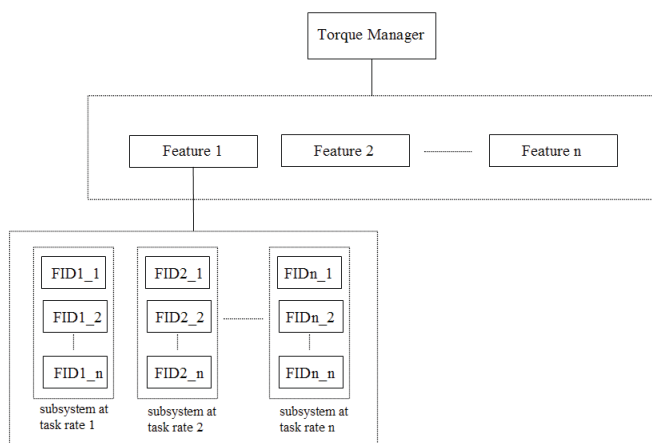


Figure 2 Torque Manager Model Structure

As illustrated in Figure 2, the Torque Manager can be decomposed into features, and each feature contains subsystems running at different task rates. At each task rate the subsystem is composed of Function Identifications (FIDs), and a FID represents the lowest level functional unit with a unique ID which implements specific function requirements of a feature. For the VSC,

feature design requirements are cascaded from HLFs and documented in a Feature Requirement Specification (FRS). Figure 2 uses Feature 1 as an example, and the same structure applies to the other features as well.

Figure 3 illustrates the feature implementation architecture and shows the task trigger flow. First, the triggers for each feature at different task rates are obtained from the TM Task Scheduler. Then in a subsystem at the specific task rate of a feature, the execution order of the FID is determined by FID triggers generated by the function-call subsystem. As shown in the Figure, feature1_trigger is obtained from signal Feature_Trigger which is the output of TM Task Scheduler. Then it is demuxed into feature1_trigger_rate1, feature1_trigger_rate2, and feature1_trigger_rate3. In "subsystem of task rate 1" feature1_trigger_rate1 is used to generate FIDtrig_1, FIDtrig_2, and FIDtrig_3 which controls the execution order of FID1_1, FID1_2 and FID1_3.

In the VSC software, a FID is implemented by legacy C code or a Simulink based model. For legacy C code, Legacy Code Block is used to integrate the legacy C code into the Simulink environment. Legacy Code Block, a Simulink block provided by Mathworks/Ford Pilot Project, allows developers to connect the input and output signals of the legacy C code with the other Simulink models. It also allows developers to specify the C code function to be used for simulation and code generation, the header and source files for the C code, and the data type for input and output signals. For newly created VSC functions, native Simulink blocks are used directly for model-based control development.

To integrate the TM into the MABX, the low level drivers of the VSC hardwired and CAN I/O signals, and high level drivers of the hardwired I/O signals are required. In addition, the integration requires the modeling of input/output signal processing blocks which include some system specific modeling, and the unit conversion, signal routing and testing harness for the input/output signals.

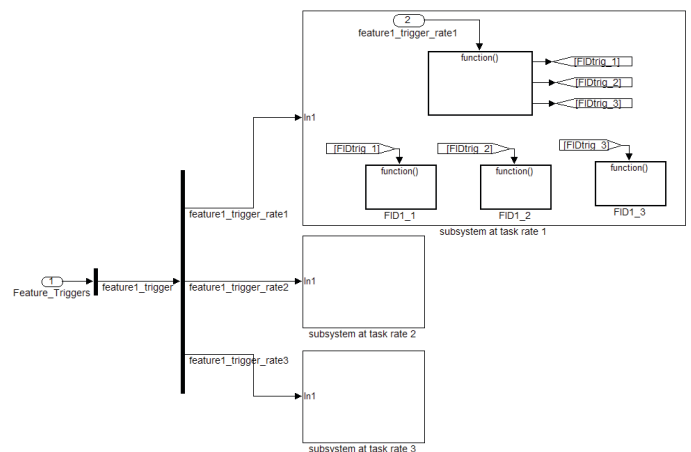


Figure 3 Feature Implementation Architecture

VSC SOFTWARE VALIDATION PROCESS

For this project a model-based control development process was followed as shown in Figure 4. As can be seen multiple validation steps are included for software validation including unit testing, MIL testing, HIL testing and vehicle testing as shown on the right hand side of the System V. This part of the model-based design is known as the software validation process. For this project the validation process was developed to ensure that a variety of issues can be captured during the development process to provide a quality software release. These issues can range from implementation issues, to system and/or component interaction issues, to missed or inadequate requirements, to production hardware usage issues.

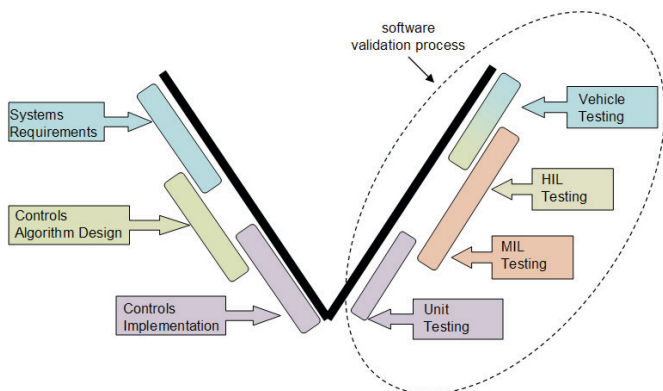


Figure 4 Model-based Control Development Process

Figure 5 illustrates the evolution of the controls software validation strategy with different test platforms. To begin the validation process, the first step is to perform unit testing which tests each implementation of the algorithms at the lowest level of functionality. This lowest level of functionality is known as an atomic unit defined as a group of code/model with a common basic functionality. Unit testing incorporates developing an input test harness and testing the algorithms and outputs based on various conditions within the Simulink environment.

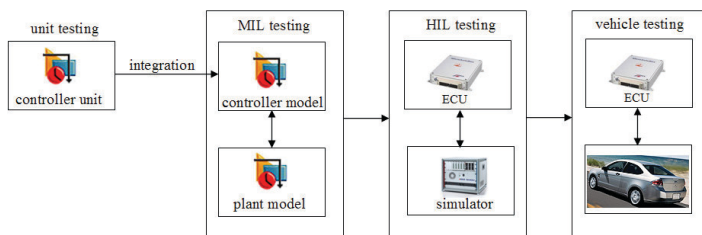


Figure 5 Different Test Platforms

Once all the controls algorithms have been developed to meet the requirements as atomic functional units, they need to be integrated together to see how they function as a system. For this phase of the controls development

process, a MIL environment was identified for the validation process as shown in Figure 5. MIL is a simulation environment which incorporates a vehicle plant model and the implemented controller model. The vehicle plant model includes the modeling of vehicle subsystems and controllers not under test. It is used to test interactions between functional units, features or system level interactions between the plant and controller. Its main function is to test implemented algorithms without worrying about the specifics of hardware platforms and without generating code.

The MIL environment provides the convenience for controls engineers to simulate control algorithms, implementation engineers to verify control implementation and test engineers to test control functionality on a PC. Also because control system functions can be simulated and tested prior to hardware availability, the use of MIL not only minimizes the use of rapid prototyping and vehicle testing but also helps to detect potential issues earlier in the controls development process. However, the limitations of the MIL environment are two-fold. On the one side, the simulation model of the vehicle plant and the interfaces between different subsystems can only represent the actual hardware with assumptions and simplifications, and secondly, the controls algorithms are still in a model form.

After the algorithms have been integrated and tested using MIL testing, the next step is to test in a HIL environment as shown in Figure 5. Usually in HIL testing, the controller to be tested exists in the form of a real hardware module or some rapid prototyping module. The rest of the vehicle system including the plant and other controllers are running in some real-time operating machine such as a simulator. In this test setup there are physical hardwired connections and/or data bus communication between the controller and the simulator. The benefit of this environment is the incorporation of the algorithms together with low and high level drivers. This ensures the correct mapping of signals and allows for the verification of CAN and hardwired signal I/O communications to the module prior to vehicle integration. Meanwhile HIL testing allows for the testing of timing critical signals which is not possible in the MIL environment. As identified above, the same limitation exists for the modeling of vehicle plant where the simulation model for the vehicle plant can only represent the actual hardware with assumptions and simplifications.

Once hardware is available and a vehicle is built, engineers can test their algorithms on the vehicle as illustrated in Figure 5. The benefits of testing in this platform are the availability of all electronic controllers and their interactions with each other for vehicle controls development. This includes the complete interface for testing the algorithms and strategies. In addition, the actual hardware components are present, which can provide the nonlinear inputs to the system and present dynamic interactions in many degrees of freedom which is not possible in the MIL/HIL environment. The