Effective Verification Strategies for Distributed Body Control Applications based on Plant Modeling and Test Case Reuse

Jinming Yang, Jason Bauman
Lear Corporation

April 27, 2010
Introduction

- Challenges in developing body-control ECUs
- Run simulations with one or more plant models in-the-loop
- Create fault conditions with the use of plant models
- Reuse test cases by collecting test data from simulation
- Hardware-in-the-loop testing using plant models imported to HIL tester
- Resolve test sequence and dependency issues by using Stateflow model to do both test vector generation and results checking
- Summary of multi-phased systems verification and validation
- Conclusion and discussion

Verification and validation are the key to high quality products
Challenges in developing body-control ECUs

The challenges in Verification & Validation

- The V&V pose a big concern to both OEMs and suppliers when a user function is implemented across multiple ECUs distributed on vehicle communication networks.
- Many user functions usually cannot be fully tested and verified when the other ECUs and vehicle network(s) are not all available.
- The capability of handling fault conditions can hardly be testable even if other ECUs are available.
- System-level design always needs to be verified in the early stage of product development.
- Product quality and timely delivery cannot be ensured if the design issues are not found in vehicle level integration testing.

Design verification in early development stage is the key to product quality and delivery timing.
Challenges in developing body-control ECUs

Why user functions are distributed?

**Technical reasons:**
- Modular design philosophy preferred by OEMs
- Required security between ECUs (AES, SHARK,…)
- More signals on the vehicle communication bus means better visibility to systems engineers
- More powerful micros lead to less concern for time latency when implementing body-control functions across vehicle bus

**Business reasons:**
- Promote the reuse of ECUs that leads to cost-saving
- One of the effective means for OEMs to protect their IPs
- OEMs leverage supplier’s expertise/experience and suppliers gain more exclusive capability by focusing on particular products
- Design flexibility for OEMs – system design can be easily scaled up/down

There exist valid reasons why user functions are implemented across multiple ECUs.
A typical vehicle body-control architecture
Many user functions are implemented across multiple ECUs.
Many user functions are implemented across multiple ECUs.

User function A

User function B

Vehicle Network A

Gateway ECU

Vehicle Network B
Many user functions are implemented across multiple ECUs.
Challenges in developing body-control ECUs

Then if we were to develop the ECU #A-1, we would have the following ECUs that interact with the one to be developed:

- ECU #A-2
- ECU #A-3
- ECU #A-(N-1)
- Gateway ECU
- ECU #B-(N-1)

User function C involves the following ECUs:

- ECU #A-2
- ECU #A-3
- ECU #A-(N-1)
- Gateway ECU
- ECU #B-(N-1)

An ECU can not be properly designed and fully validated without the knowledge of other ECUs on the vehicle bus.
The interaction among ECUs is often closed-loop so they cannot be simulated with a simple CAN tool or equivalent.

Challenges in developing body-control ECUs

Parallel Programming Issues
- Race conditions
- Synchronization
- Deadlocks
- Timeout

Vehicle Network Issues
- Time latency
- Missing messages
- Handshakes between ECUs

The closed-loop interactions between ECUs add a lot of complications to implementation and testing.
Run simulations with “in-the-loop” plant models

A plant model is necessary in design verification

Signal Builder
Unit-level Test Scenarios for Model

Scope Output Capture

Plant Model

Feature Model
A plant model is placed “in-the-loop”

The closed-loop will look like...

Signal Builder
Unit-level Test Scenarios for Model

Feature Model

Plant Model
If using the same example, we will have 5 plant models. Then the test harness model will look like:
It is both necessary and beneficial to create fault conditions using plant models

- Even if other ECUs are available, fault conditions from other ECUs cannot be easily created, so the capability of handling fault conditions cannot be validated without simulation.
- A fault insertion mechanism is needed, which can be realized using plant models.
- Besides fault conditions, race conditions from other ECUs are not easily created.
- It is often challenging to duplicate issues found in pre-production – manipulating plant models makes it possible/easier to duplicate particular issues.

It’s always crucial to fully validate the exception handling capability, which is the key to vehicle safety…
Adding a fault control to a plant model

A plant model fault control is necessary ...

Signal Builder
Unit-level Test Scenarios for Model

Scope Output Capture

Feature Model

Plant Model
Adding fault conditions to multiple plant models

If using the previous example, fault control signals can be introduced in Signal Builder that create fault conditions for plant models.
The Signal Builder provides the benefits of creating intricate fault cases. In the charts below, one can easily create cases where fault happens during switch input, fault goes away during switch being pressed and fault happens after the switch...

INPUT #1 to ECU A (e.g., Ignition Status)

INPUT #2 to ECU A (e.g., A Switch Status)

Plant Model Fault Control
Test data can be collected from multiple sources.

The MATLAB tools offer various means of recording the test data and converting them to required formats for reusing test cases.
Reuse test cases for hardware-in-the-loop testing

The test cases can be reused by importing plant models to a hardware-in-the-loop tester
Many test cases have to be in sequence and have dependency on the preceding ones

- Many tests are required to be conducted in a pre-determined sequence
- If one test case fails, it would make no sense to run the rest of test cases
- Certain test results need to be checked prior to kicking off the following test cases
- Results-checking and dynamically deployed test cases (vs. static test cases) are useful for testing diagnostics functions
- A Stateflow model can be used to dynamically produce test vector as well as perform results checking
- Building a subset of all the test cases into a Stateflow model makes it possible to validate an ECU product in a “fast run” manner (i.e., only run tests for basic functions)
This will be a dynamic test case generator and useful to testing diagnostics features and commands…

Generate test cases and dynamically deploy tests

A Stateflow Model

Test stage #1

Test stage #2

Test stage #3

Test stage #n

Feature Model (ECU #A-1)

ECU #A-2

ECU #A-3

ECU #A-(N-1)

Gateway ECU

ECU #B-(N-1)

Generate test report

Plant models
## In-The-Loop Test/Verification

<table>
<thead>
<tr>
<th></th>
<th>Unit Model Verification</th>
<th>Verification with Plant Models</th>
<th>ECU Test with Plant Models</th>
<th>Vehicle Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purposes</strong></td>
<td>Verify if the unit model functions match requirement specs.</td>
<td>Find out if the feature model functions properly with its plant models running in-the-loop.</td>
<td>Find out if the ECU works properly with its interactive/surrounding plant models.</td>
<td>Find out if the ECU works on vehicle with the presence of other ECUs.</td>
</tr>
<tr>
<td><strong>Test Scope</strong></td>
<td>Unit-level model behaviors and implementation models</td>
<td>Model behaviors that interact with their surrounding behavioral plant models</td>
<td>ECU (application SW and its low-level drivers) that interacts with other ECUs on vehicle bus</td>
<td>ECU functions and performance on actual vehicle</td>
</tr>
<tr>
<td><strong>Issues (that could be found)</strong></td>
<td>Implementation - functionality may not match the requirement spec.</td>
<td>Design/implementation issues - interactive functions with other ECUs and vehicle bus.</td>
<td>Interface and time latency issues - existent between SW and HW, and with other ECUs.</td>
<td>Connector and/or wiring issues and vehicle data bus issues.</td>
</tr>
<tr>
<td><strong>Test Method Limitations</strong></td>
<td>Unable to identify systems level issues</td>
<td>Unable to identify interactive behaviors with other functional modules and lower level implementation issues</td>
<td>Unable to unveil actual interactive functions among ECUs</td>
<td>Uneasy to identify issues in handling fault conditions</td>
</tr>
<tr>
<td><strong>Tools</strong></td>
<td>MATLAB and SimuLink/Stateflow</td>
<td>MATLAB and SimuLink/Stateflow</td>
<td>HIL software and hardware</td>
<td>CANoe, VehicleSpy, etc.</td>
</tr>
</tbody>
</table>
More test cases are created in different phases while reusing test cases is always a goal.

- **Unit Model Verification**: 80% of test cases can be reused.
  - New test cases are created from requirements.
- **Verification with Plant Models**: 100% of test cases can be reused.
  - More test cases are added to reused test cases.
- **ECU Test with Plant Models**: 60% of test cases can be reused.
  - More test cases are added to reused test cases.
- **Vehicle Testing**: More vehicle level test cases are added to reused test cases.
Conclusion and discussion

The Lear product development approach focuses on verification and validation:

- Leverage executable specifications
- Create test cases by extensively running model-in-the-loop simulations
- Emphasize the use of plant models
- Reuse test cases through the entire development process
- Reuse plant models in both virtual test harness models and ECU in-the-loop testing
- Use simplified version of plant models (no full version of requirements specification available, not realistic to create full models)
- The methodology is also very useful for a large scale of project when multiple large models need to go into one micro
Thank you!