모델 기반 설계를 활용한 Legacy C,C++ 코드의 통합과 검증

김학범 차장, 매스웍스코리아
Model-Based Design

Development Process

- Executable models
  - Unambiguous
  - Only “one truth”

- Executable Specifications

- Design with Simulation

- Simulation
  - Reduces physical prototypes
  - Systematic “what-if” analysis

- Continuous Test and Verification

- Analyze and Test the Design
  - Create reusable tests
  - Detects errors earlier
  - Formal methods

- Automatic Code Generation
  - Minimizes coding errors

- Automatic code generation

- Model-Based Design
How to get started MBD with Legacy Code?
Agenda

▪ How to get started MBD with Legacy Code?

▪ Legacy Code Integration using Simulink

▪ Legacy Code Verification

▪ Key Takeaways
Agenda

▪ How to get started MBD with Legacy Code?
  ▪ Legacy Code Integration using Simulink
  ▪ Legacy Code Verification
  ▪ Key Takeaways
Verify Legacy Code using Simulink

Legacy code

System Design

MBD

SIL Coverage Report by Model
Top Model: crs_controller_harness_SIL

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Decision</th>
<th>Condition</th>
<th>Statement</th>
<th>Function</th>
<th>Function call</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL COVERAGE</td>
<td>81%</td>
<td>88%</td>
<td>86%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

1. . . Model(s) 2 2 -- 100% 100% 100% 100% 100%
2. . . Cmnn Code File(s) 50 81% 86% 63% 100% 100% --
3. . . Test Case 1 -- -- 100% 100% 100% --

S-Function

Hand Coding

C file

project

Test Case
Experiment with a Small Piece of the Project

Legacy code

Hand Coding

System Design

project

doc

MBD

project

Modeling

C file

slx
Adopt Full MBD to Project
Model-Based Design With Legacy C/C++ Code?

- Legacy Code
- MBD with Legacy Code
- Full MBD

Legacy Code Verification using Simulink

MBD with C/C++ code
Agenda

▪ How to get started MBD with Legacy Code?

▪ Legacy Code Integration using Simulink

▪ Legacy Code Verification

▪ Key Takeaways
Model-Based Design with Legacy Code

Example: EV Vehicle Simulation for VCU

Simulink block calling Legacy codes
Legacy Code Integration Methods

- C code integration
  - C MEX S-function
  - `coder.ceval` in MATLAB Function Block
  - Stateflow Custom Code interface

- Legacy Code Tool
- C Caller Block
- C Function Block
- Simulink Code Importer
- S-Function Builder
- Manual C code wrapping
## Legacy Code Integration Methods

<table>
<thead>
<tr>
<th>Type</th>
<th>Features</th>
<th>Manual Building Process</th>
<th>Block / UI Icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Legacy Code Tool</td>
<td>▪ Full flexibility, Generate and build S-function with easy-to-use MATLAB API</td>
<td>Need to build when changing codes</td>
<td>![S-Function Icon]</td>
</tr>
<tr>
<td></td>
<td>▪ Creation of device driver blocks for HW input and output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C Caller Block</td>
<td>▪ Easy to call a function in legacy code</td>
<td></td>
<td>![C Caller Icon]</td>
</tr>
<tr>
<td></td>
<td>▪ Calls a single function in one block</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>▪ Good for Unit test of C code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C Function Block</td>
<td>▪ Advantage of C Caller Block + Easy to add C code in a Simulink block</td>
<td>No manual build process</td>
<td>![C Function Icon]</td>
</tr>
<tr>
<td></td>
<td>▪ Call multiple functions in one block</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>▪ Unit and integration test of C code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simulink Code Importer</td>
<td>▪ Easy to access (UI in Toolstrip)</td>
<td></td>
<td>![Code Importer Icon]</td>
</tr>
<tr>
<td></td>
<td>▪ Create a block library for C function</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>▪ Unit and integration test of C code with Simulink Test</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Legacy Code Tool

- Integrate existing C/C++ functions, such as device drivers, lookup tables, and general functions and interfaces, into Simulink.
C Caller Block

- Key feature
  - Automate the process
  - Synchronize with custom code changes with C Caller

- Tedious
- Error prone
- Hard to maintain

Define Block Interface → Build Simulation MEX → Write Codegen TLC

Legacy Code Tool → Must build after modifying codes

C Caller: Automatically synchronized

```c
legacy_code('compile', def);
```
C Caller Block
Specify Custom Code in the Configuration Parameters

- Custom code is specified on the Configuration Parameters
  - **Header file section:** Any code that needs to be inserted into the header file
  - **Source files section:** List of source files that needs to be compiled
C Caller Block

Select the function that you want to call

<FunctionName>
C Caller Block
Customize the function that you want to call

- Mapping inputs, outputs or parameters to C Caller Block

1) Change argument scope to “Output”
2) (Optional) Override with a better port name
3) Complete the test model with connecting signal ports
C Function Block

- **Motivation**
  - Make it really easy to add custom code to Simulink
  - Make the simpler uses of S Function Builder easy
  - Replace Legacy Code Block

- **Behavior**
  - Code is parsed and managed in Simulink
  - Better customizing and diagnostics
  - Supports Start/Terminate
C Function Block
Customize the code and variables that you want to call

- Customize code in Output Code, Start Code, Terminate Code
- Mapping inputs, outputs or parameters to C Function Block
C Function Block
Support Persistent Scope and specify different code

- Write C code directly in your models
  - Call multiple external functions
- Support Persistent scope
- Specify different code for code generation and simulation using the flag `MATLAB_MEX_FILE`
- Interface directly with C++ classes in R2022a
Simulink Code Importer
Import C code as reusable Simulink libraries

- Import C Code as reusable Simulink libraries
- Block representation of C Code algorithms using C caller
- Wizard UI provides a step-by-step guidance
- Intuitive setup MATLAB APIs also available
- Integrated with Simulink Test for V&V workflow
DEMO: Simulink Code Importer
Integrating Legacy Code using Simulink Code Importer
Code Generation in Integrated Model

Integrating Legacy Code in Simulink and Generating Code

Integrate Legacy code in Simulink block

Call main function of Legacy code
Agenda

- How to get started MBD with Legacy Code?
- Legacy Code Integration using Simulink
- Legacy Code Verification
- Key Takeaways
Why Using Simulink for Legacy Code Testing?

Test harness model

S-Function or C Caller

Test case generation

Code coverage analysis

Test Cases

SIL Coverage Report by Model

Top Model: crs_controller_harness_SIL

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Decision</th>
<th>Condition</th>
<th>Statement</th>
<th>Function</th>
<th>Function call</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL</td>
<td>81%</td>
<td>86%</td>
<td>96%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>
void CruiseControl_MdlAdv_ReqLink_step(boolean_T arg_CruiseOnOff, boolean_T arg_Brake, uint8_T arg_Speed, boolean_T arg_CoastSetSw, boolean_T arg_AccelResSw, boolean_T *arg_engaged, uint8_T *arg_tspeed)
{
    boolean_T AccelResSw_prev;
    boolean_T CoastSetSw_prev;

    /* Chart: '<Root>/Compute target speed' incorporates:
     * Inport: '<Root>/AccelResSw'
     * Inport: '<Root>/Brake'
     * Inport: '<Root>/CoastSetSw'
     * Inport: '<Root>/CruiseOnOff'
     * Inport: '<Root>/Speed'
     */
    /* Gateway: Compute target speed */
    if (((uint32_T)DW.temporalCounter_i1 < 3U) {
        DW.temporalCounter_i1 = (uint8_T)(uint32_T)((uint32_T)DW.temporalCounter_i1 + 1U);
    }
Legacy Code Verification Workflow

```c
void CruiseControl_RJ1AL_Hreqr_step(boolean_T arg_cruiseOnOff, boolean_T
arg_brake, uint8_T arg_speed, boolean_T arg_CostSetSw, boolean_T
arg_AccelResSw, boolean_T *arg_engaged, uint8_T *arg_tspeed)
{
    boolean_T AccelResSw_prev;
    boolean_T CoastSetSw_prev;

    /* Chart: 'Root/Compute target speed' incorporates:
    * Import: 'Root/AccelResSw'
    * Import: 'Root/brake'
    * Import: 'Root/CostSetSw'
    * Import: 'Root/CruiseOnOff'
    * Import: 'Root/Speed'
    */
    /* Gateway: Compute target speed */
    if ((uint32_T)CruiseOnOffCounter_i1 < 3u) {
        CruiseOnOffCounter_i1 = (uint8_T)(uint32_T)(CruiseOnOffCounter_i1
        + 1u);
    }
}
```
Legacy Code Verification Workflow
Legacy Code Verification Workflow

<table>
<thead>
<tr>
<th>ANALYZED MODEL</th>
<th>REPORT</th>
<th>COMPLEXITY</th>
<th>DECISION</th>
<th>CONDITION</th>
<th>MCDC</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CruiseControl_MdlAdv_ReqLink</code></td>
<td>🚀</td>
<td>31</td>
<td>47%</td>
<td>31%</td>
<td>6%</td>
</tr>
</tbody>
</table>

Partial Coverage

Test Cases

Test Generator

Simulink Design Verifier
Legacy Code Verification Workflow

<table>
<thead>
<tr>
<th>ANALYZED MODEL</th>
<th>REPORT</th>
<th>COMPLEXITY</th>
<th>DECISION</th>
<th>CONDITION</th>
<th>MCDG</th>
</tr>
</thead>
<tbody>
<tr>
<td>CruiseControl_MdlAdv_ReqLink</td>
<td>🔄</td>
<td>31</td>
<td>47%</td>
<td>31%</td>
<td>6%</td>
</tr>
</tbody>
</table>

New Test Cases

Test Cases

Test Generator

Simulink Design Verifier

Partial Coverage

```
if (Speed < 100) && (inStopPhase == true && (inStopPhase == true))
    % Test Cases
    % Code Generation
else
    % Test Cases
    % Code Generation
end
```
Legacy Code Verification Workflow

New Test Cases

Test Cases

Higher Coverage

Coverage Report for CruiseControl_MdlAdv_ReqLink

Table of Contents
1. Analysis information
2. Associated Tests
3. Summary
4. Details

Analysis Information

Model Information
Model version: 1.491
Author: The MathWorks Inc.
Last saved: Wed Jul 06 15:02:21 2023

Harness Information
Harness model(s): CruiseControl_MdlAdv_ReqLink, Harness3
Harness model owner: CruiseControl_MdlAdv_ReqLink

Generate Report
Simulink Code Importer in Simulink Test

- Supports unit test and integration test
- Test setup automation
  - Importing c code to Simulink C Caller block, creating harness, creating test file and running the test file

※ For unit tests, additionally creates a sandbox to isolate the imported functions.
Simulink Code Importer in Simulink Test

- Import legacy code using Simulink Code Importer in Simulink Test

Test Manager

Simulink Code Importer Wizard
Simulink Code Importer in Simulink Test

- Build library model and test harness
- Automatically configured test setting in Test Manager

Test harness calling Legacy codes

Auto configuration for legacy code verification
Simulink Code Importer in Simulink Test

- Simulink Code Importer calls code using C Caller Block
  - Using C Caller block's features after building library
  - Edit port specification in Block Parameters
  - Configure custom code settings

Custom code settings
Edit port specification
Simulink Code Importer in Simulink Test

- Support two options for C code test
  - **Unit Test** for a subset of custom code
  - **Integration Test** for entire custom code

```c
void foo(void) {
}

void bar(void) {
}
```
Simulink Code Importer in Simulink Test

- Easy to support Integration Test using Simulink Code Importer
  - Import integrated custom code in Simulink
  - Support function, function call coverage analysis

```c
void foo(void) {
}

void bar(void) {
}

void main(void) {
    foo();
    ...
    bar();
}
```
Simulink Code Importer in Simulink Test

- How to verify unit functions which include interface functions?
  Ex) Application SW uses interface functions which is provided from Basic SW

```
void foo(void)
{
  Vehicle_Speed = Get_Speed();
  Engine_Torque = Get_Torque();
  .
  }
```

Ex) Application SW uses middleware APIs for data interface such as AUTOSAR

```
In16t Get_Speed(void)
{
  Return Speed;
}
```

```
In16t Get_Torque(void)
{
  Return Torque;
}
```
Simulink Code Importer in Simulink Test

- **Create Sandbox for C code unit testing**
  - **Auto-stub files**: Contains the `auto_stub.h` and `auto_stub.c` files, which are generated only if the imported code has undefined symbols
  
  ```
  void BasicRoll()
  { 
    float32 rb 우리가; 
    boolean rb_Not-engaged; 
    // Logic: 'CSIV/Not-Engaged' incorporates:
    // Import: 'CPS/IMAP' 
    rb_Not-engaged = Rte_IRead_rtwdemo_roll_test_step_AP_Eng_AP_Eng();
    // Discrete-Integrator: 'CSIV/Integrator' */
    if (rb_Not-engaged || (rtdm_integrator_PrevAssetState != 0)) { 
      rtdm_integrator_SSTATE = 0.0; 
    }
  }
  ```

  - **Manual stub files**: Contains the `man_stub.h` and `man_stub.c` files, which you can use to manually stub symbols
  
  - **Aggregated header**: Contains all definitions of functions, interfaces which are related unit function

  ```
  Boolean SLStubOut_Rte_IRead_rtwdemo_roll_test_rtwdemo_roll_test_step_AP_Eng_AP_Eng;
  Boolean Rte_IRead_rtwdemo_roll_test_rtwdemo_roll_test_step_AP_Eng_AP_Eng(void) {
    return SLStubOut_Rte_IRead_rtwdemo_roll_test_rtwdemo_roll_test_step_AP_Eng_AP_Eng;
  }
  ```
DEMO: Simulink Code Importer in Simulink Test

Import C code and Create Sandbox for unit testing
DEMO: Simulink Code Importer in Simulink Test
Test case generation and unit test in Test Manager
Static Code Analysis with Polyspace

- Code metrics and standards
  - Comment density, cyclomatic complexity,…
  - MISRA and Cybersecurity standards
  - Support for DO-178, ISO 26262, ….

- Bug finding and code proving
  - Check data and control flow of software
  - Detect bugs and security vulnerabilities
  - Prove absence of runtime errors

Results from Polyspace Code Prover
Static Code Analysis with Polyspace and Simulink

- Run time error / MISRA rule check
- Polyspace report from Simulink
- Reducing Polyspace set-up efforts
Agenda

- How to get started MBD with Legacy Code?
- Legacy Code Integration using Simulink
- Legacy Code Verification

- Key Takeaways
Key Takeaways

- **How to get started MBD?**
  - Verify Legacy Code using Simulink
  - Experiment with a Small Piece of the Project
  - Adopt Full MBD to Project

- **Legacy Code Integration**
  - Legacy code tool, C Caller Block, C Function Block, Simulink Code Importer

- **Legacy Code Verification in Simulink**
  - High flexibility for test input
  - Automation of verification workflow
  - Easy test case management and nice visualization
Thank you