An end-to-end approach to Design and Verify BMS: from Requirements to Virtual Field Testing

Conrado Ramirez
MathWorks

Irina Costachescu
NXP

Marius Andrei
NXP

Carlos Villegas
Speedgoat
Agenda

• System-level Modeling and Verification

• From Model to Embedded Hardware: SiL and PiL Verification

• Real-time verification with HiL Testing
System-level Modeling and Verification
Motivation

Collaboration Gap

Long Iteration Cycles

Safety Critical System

Multi-Domain System Modeling

Simulations & Code Generation

Model V&V and HIL Testing
Evaluate Battery Management System Behavior

• Simulate interaction between software modules

• Design & test algorithms for different operating conditions

• Calibrate software before putting into battery pack or vehicle
Battery Management System Architecture

Battery Pack

Cell Monitoring

Software

Supervisory tasks
SOC estimation
Contactor management
Isolation monitoring
Fault detection and recovery
Thermal management
Current & power limits

Measurement
Cell Diagnostic, Cell Balancing
A BMS for a battery pack is typically composed of:

1) Battery Management Unit (BMU)
   Centralized control of battery pack. Includes state estimation (SoC, SoH, SoX). Typically uses CAN as well as proprietary protocols to interface to CMU.

2) Cell Management Units (CMU)
   Takes care of cell balancing (active or passive) and measurement of individual cell voltages (1s) and temperatures. Typically interfaced using proprietary communication protocols.

3) Battery Junction Box (BJB)
   Switching unit connecting to the load that typically includes current sensors. May interface via CAN bus.
Model-Based Design Verification Workflow

**Model Verification**
*Discover design errors at design time*

**Code Verification**
*Gain confidence in the generated code*

- **Modeling**
- **Textual requirements**
- **Executable specification**
- **Generated code**
- **Object code**
- **Model used for production code generation**
- **Back-to-Back Testing**
- **Prevention of unintended functionality**
- **Reviews and analysis at model level**
Requirements Import and Export

- **Import** from:
  - Word / Excel
  - IBM® Rational® DOORS®
  - ReqIF™ standard

- **Update** synchronizes changes from source

- **Export** ReqIF
  - Enables roundtrip with external tools

**Simulink Requirements**

- **Import**
  - References to crs_req.docx
    - 1 Overview
    - 2 System overview
      - 2.1 System inputs
      - 2.2 Cruise control mode indicator
      - 2.3 Cruise control modes

- **Export**
  - Driver Switch Request Handling
    - 1.1 Switch precedence
    - 1.2 Avoid repeating commands
    - 1.3 Long Switch recognition
    - 1.4 Cancel Switch Detection
Manage Requirement Traceability

<table>
<thead>
<tr>
<th>Implementation Status</th>
<th>Verification Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implemented</td>
<td>Passed</td>
</tr>
<tr>
<td>Justified</td>
<td>Failed</td>
</tr>
<tr>
<td>Missing</td>
<td>Unexecuted</td>
</tr>
</tbody>
</table>

- **Implementation Status**: Implemented, Justified, Missing
- **Verification Status**: Passed, Failed, Unexecuted, Missing
Systematic Functional Testing

Test Inputs  Component Under Test  Test Results
Create Test Harnesses

Test Cases

Inputs
- MAT / Excel file (input)
- Signal Editor
- Test Sequence

Assessments
- MAT / Excel File (baseline)
- MATLAB Unit Test

Test Harness
Test Management

- **Test Manager**
  - Author, manage, organize tests
  - Execute simulation, equivalence and baseline tests
  - Review, export, report

- **Test Harnesses**
  - Isolate Component Under Test
  - Synchronized, simulation test environment

- **Test Authoring**
  - Specify test inputs, expected outputs, and tolerances
  - Construct complex test sequences and assessments

---

**Test Results**

**Test Harness**

**Temporal Assessments**

**Test Sequence**

**Reports**
From Model to Embedded Hardware: SiL and PiL Verification
Hardware Aware Models
Embedded Applications Development Environment

**Drivers, Middleware, Libraries**
- Simplify hardware access by using hardware optimized software

**Configuration, Build, Debug Tools**
- Application development inside an IDE
- Build Tools, Debug Tools and Configuration Tools integrated within the IDE
- Drivers, Middleware and Libraries configuration and initialization in a graphical environment

**Real Time Monitor, Demo Tools**
- Check the status of the running on target application in real time
- Write and read variables, registers, memory locations
- Monitor signals on the embedded target
- Fast demo design
From Model to Embedded Hardware

MODEL-BASED DESIGN TOOLBOX

- Collection of Drivers, Libraries and Tools
- Embedded systems design and deployment on NXP MCUs directly from Simulink

MATHWORKS ECOSYSTEM MATLAB/SIMULINK

- Model-Based Design
- Simulation
- Automatic Code Generation
- Verification and Validation
From Idea to Application

MODEL-BASED DESIGN TOOLBOX
- Collection of Drivers, Libraries and Tools
- Embedded systems design and deployment on NXP MCUs directly from Simulink

MATHWORKS ECOSYSTEM MATLAB/SIMULINK
- Model-Based Design
- Simulation
- Automatic Code Generation
- Verification and Validation
Model-Based Design Toolbox
Embedded Applications Development Flow
Embedded Applications Development Flow

- APPLICATION
- LIBRARIES
- MIDDLEWARE
- DRIVERS
- HARDWARE
Virtual Field Testing

Model-in-the-Loop
- Plant model with test signals
- Controller Model

Software-in-the-Loop
- C code with test signals
- C Code on host PC

Processor-in-the-Loop
- Embedded code with test signals
- C Code on target MCU
Virtual Field Testing

Model-in-the-Loop
- Plant model with test signals
- Controller Model

Software-in-the-Loop
- C code with test signals
- C Code on host PC

Processor-in-the-Loop
- Embedded code with test signals
- C Code on target MCU

External Mode
- Device monitoring
- C code on target MCU
Virtual Field Testing

Model-in-the-Loop
- Plant model with test signals
- Controller Model
- C Code on host PC

Software-in-the-Loop
- C code with test signals
- C Code on host PC
- Embedded code with test signals

Processor-in-the-Loop
- Embedded code with test signals
- C Code on target MCU
- Device under test

Hardware-in-the-Loop
- Device under test
- C code on target MCU
- Plant model emulated on Speedgoat
MiL - SiL - PiL - HiL
MiL - SiL - PiL - HiL

Model-in-the-Loop
- Plant model with test signals
  - Desktop Simulation (on PC)

Software-in-the-Loop
- C code with test signals
  - Embedded Code
  - Generated Code
  - PC Compiler

Processor-in-the-Loop
- Embedded code with test signals
  - Object File
  - Object Code Execution (on PC)

Hardware-in-the-Loop
- Device under test
  - C code on target MCU
  - Plant model emulated on Speedgoat

Test Vectors

Results

Compare

== ?

Results
Object Code Execution (on target MCU)

Model-in-the-Loop
Plant model with test signals

Desktop Simulation (on PC)

Software-in-the-Loop
C code with test signals

C code on host PC

Embedded Code on Target MCU

Processor-in-the-Loop

Hardware-in-the-Loop

C code on target MCU

Plant model emulated on Speedgoat

Device under test

Model
Embedded Coder

Generated Code

Test Vectors

Cross Compiler

Object File

Object Code Execution (on target MCU)

Compare

Results

MiL - SiL - PiL - HiL
MiL and PiL Equivalence Test

Model
Desktop Simulation (on PC)

Embedded Coder
Generated Code

Cross Compiler
Object File
Object Code Execution (on target MCU)

Compare
== ?

Results

Results
StateRequest

BMS_Software

BMS_Algorithms

BMS_Info

BMS_Cmd

Sensors

"Copyright 2018 - 2022 The MathWorks, Inc."
Real-Time Verification with HiL Testing
Real-Time Systems for BMS Testing

**Requirements**

**Desktop Simulation**
- **System Model**
  - **Plant**
    - Batteries, loads, cell balancing electronics, batteries, passive circuit components
  - **Controller**
    - Algorithms for energy and thermal management

**Real-Time Simulation Platform**
- **Hardware Prototype**
  - Battery packs, power electronics, electrical loads, fast charging
- **Rapid Control Prototyping**
  - Control algorithms running on a real-time computer, microcontroller, or Simulink-programmable FPGA

**Hardware-in-the-Loop Simulation**
- Behavioural model running on a real-time computer with virtual battery cells and packs

**Real-Time Communication**
- C or HDL Code generated from plant model
- C or HDL Code generated from controller model

**System Model**
- **Plant**
  - Batteries, loads, cell balancing electronics, batteries, passive circuit components
- **Controller**
  - Algorithms for energy and thermal management
Typical Battery Management System Architecture

A BMS for a battery pack is typically composed of:

1) Battery Management Unit (BMU)
   Centralized control of battery pack. Includes state estimation (SoC, SoH, SoX). Typically uses CAN as well as proprietary protocols to interface to CMU.

2) Cell Management Units (CMU)
   Takes care of cell balancing (active or passive) and measurement of individual cell voltages (1s) and temperatures. Typically interfaced using proprietary communication protocols.

3) Battery Junction Box (BJB)
   Switching unit connecting to the load that typically includes current sensors. May interface via CAN bus.
BMS Currents

• **Full load currents**: linked to load or charger. Typically in the range of hundreds of Amperes for EVs.

• **Balancing currents**: Currents flowing through cell balancing circuit. Such circuit could be passive (discharge cells) or active (redistribute charge). Typically between 100mA and 5A.
HIL Testing of BMU

- **Device under Test (DUT):** technique to test embedded control systems where part of the plant is simulated.
HIL Testing of BMU

- **Hardware-in-the-Loop Testing**: technique to test embedded control systems where part of the plant is simulated.
HIL Testing of BMU and CMU

- **Cell emulation**: mimic the battery cell electrical power (voltage and current) using a real-time simulation (HIL simulator and power amplification). It is a type of Power HIL.
HIL Testing for BMU and CMU

- Test control and electrical interfaces
- Each cell emulated: up to 6V and 5A
- Emulate the electrical behaviour of battery cells
- Stack up to 312 of virtual battery cells (1600 V)
- Include communication interfaces like isolated SPI or CAN
HIL Testing of full BMS and load

Device under Test can also include the real load (e.g. powertrain or EV charger). The interfaces include electrical power.
HIL Testing of BMS

BJB

Power Amplifier

Hardware-in-the-Loop Simulator
Real-Time Simulator

Built for High-Performance
• Multicore CPU
• Simulink-programmable FPGAs

Battery Emulation I/O
• Serial and bus communication (SPI, CAN, UART)
• Cell temperature emulation
• Shunt emulation
• Cell controller emulation
• Fault insertion
Battery Cell Emulator

Cell-level emulation and measurement
• Simulate up to 8V
• Sink and source current up to 5A

Series and Parallel connections
For optimized capacity and voltage ratings

Small to Large BMS test systems
• 12 cell per unit
• Stack up to 312 cells
• Voltage isolation of 1.6 kV
Interface Panel & BMS Controller

- Battery Management Unit
- Cell Measurement Unit
- Interface Cable
HIL Testing for BMU and CMU

- Test control and electrical interfaces
- Each cell emulated: up to 6V and 5A
- Emulate the electrical behaviour of battery cells
- Stack up to 312 of virtual battery cells (1600 V)
- Include communication interfaces like isolated SPI or CAN
Device under Test can also include the real load (e.g. powertrain or EV charger). The interfaces include electrical power.
Continuous Integration with Hardware

**Workflow**

Controller developer

HL system engineer

Controller tester

Source code repository

Build and test results

Continuous Integration Server

**Benefits**

- **Repeatability** – repeatable, automated process
- **Quality** – always test against latest changes
- **Speed** – test early, test often
- **Collaboration** – identify integration issues quickly
- **Audit-ready** – trace issues to the source
Continuous Integration with Hardware

Controller developer

HIL system engineer

Controller tester

Build and test results

Source code repository

Continuous Integration Server

Source control in projects

Support for main CI platforms

Compare changes

Check model quality and standard compliance

Quickly visualize test reports and results

Jenkins  GitLab  GitHub

HIL test server 1

HIL test server 2

accellera SYSTEMS INITIATIVE
Questions