Developing Battery Management System using Simulink

Prashant Hegde
MathWorks

Durvesh Kulkarni
MathWorks

Batteries everywhere !


Aerospace and Defense


Automotive


Electronics


Medical Devices


Handheld Devices

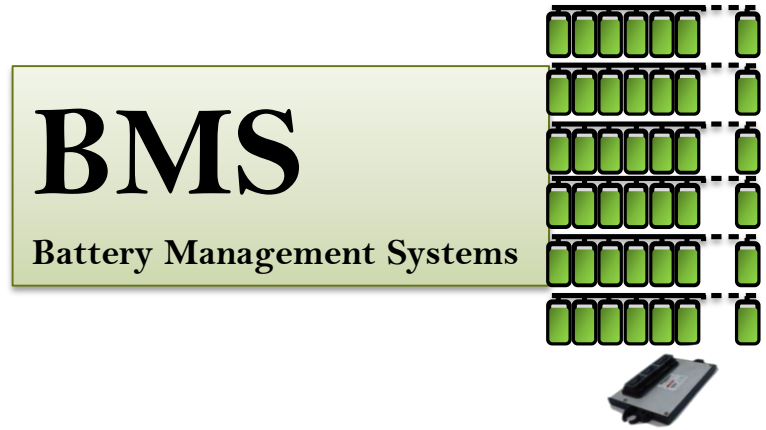
Batteries for Traction
Batteries for Aviation / Aerospace
Batteries for Consumer Electronics
Stationary Batteries
Energy Storage Systems

Battery Operated Systems

- Driving Range : 450 Kms in case of vehicle
- Talking Duration : 14 hrs. in case mobile
- Back-Up time : 6 hrs. in case of UPS / Storage

By 2030, ~ 30% of all cars are expected to be electric, according to the International Energy Agency

- Range
- Charging time
- Battery life



Motivation

Collaboration Gap

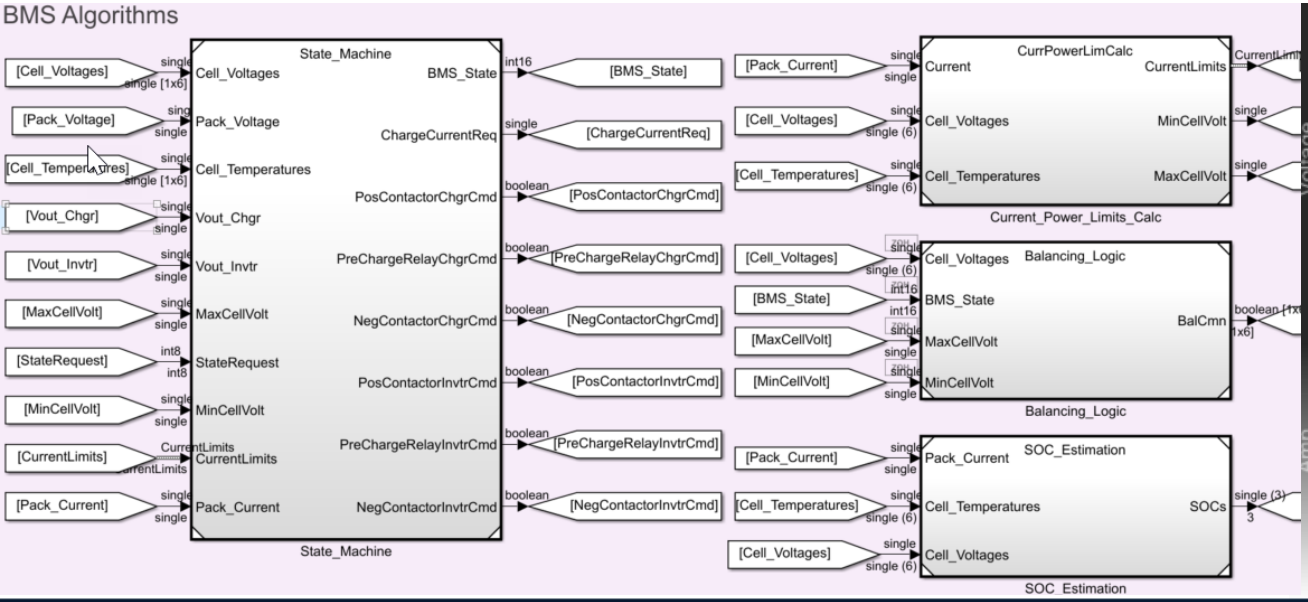
Long Iteration Cycles

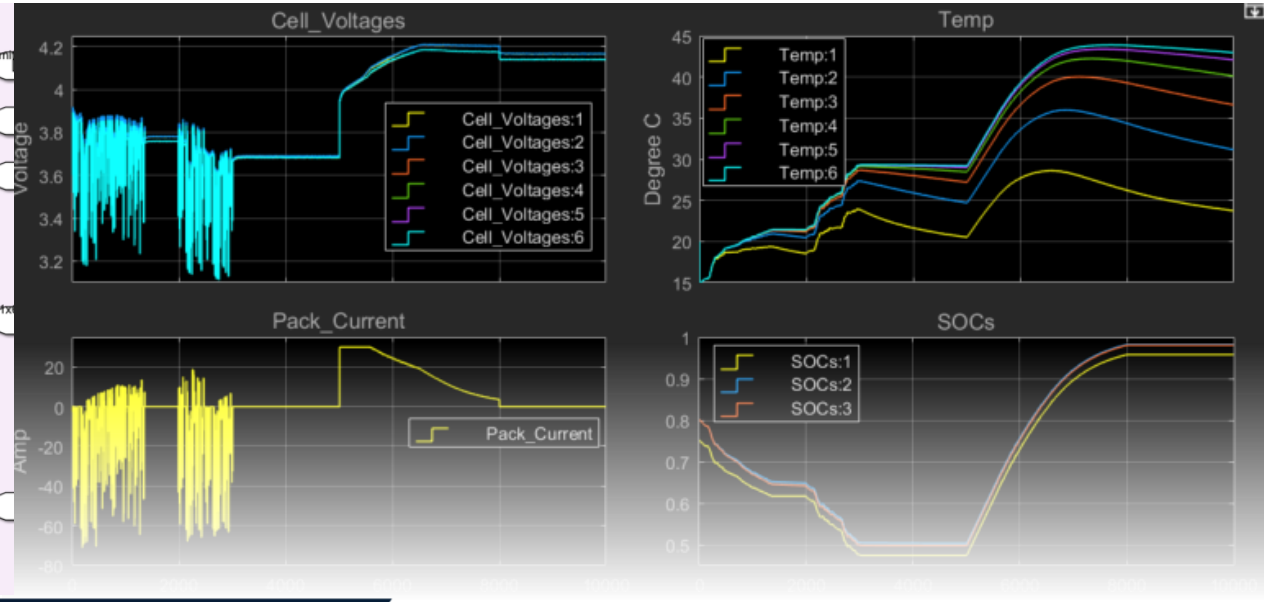
Safety Critical System

Multi-Domain Modeling Environment

Simulations & Auto Code Generation

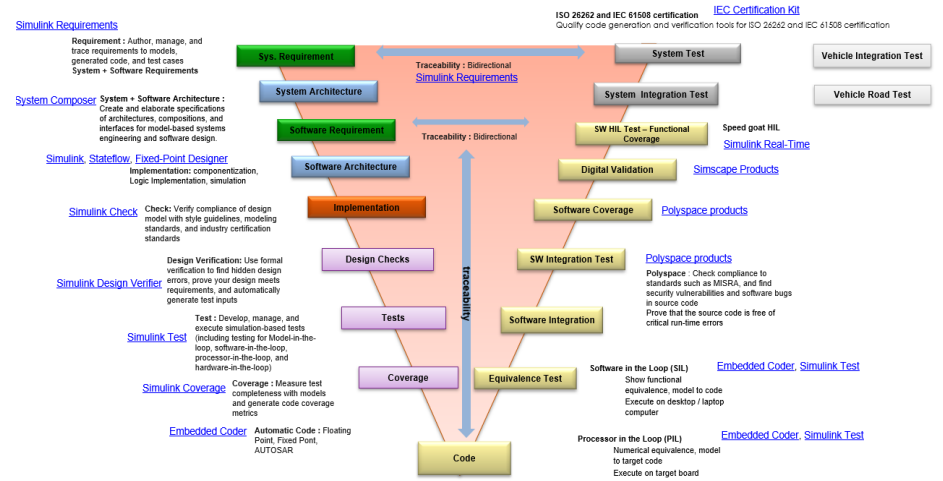
V&V and Hardware-In-Loop Testing

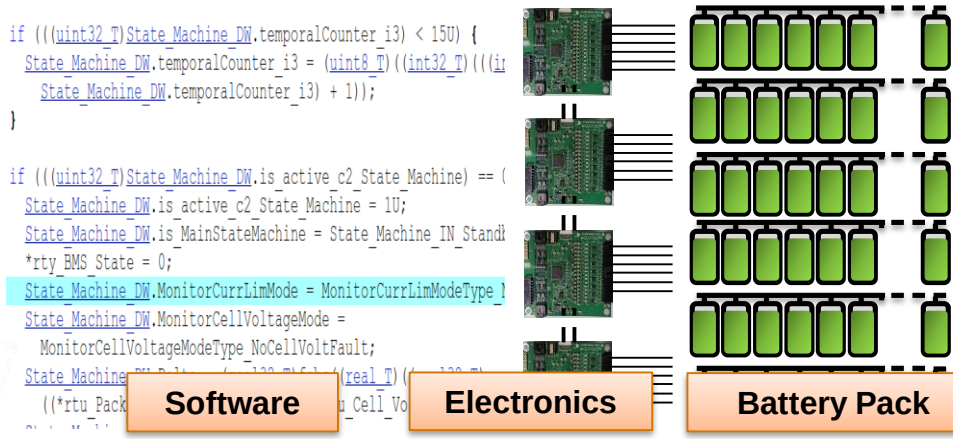




Agenda

- What is BMS and what engineers worry about?
- Developing the architecture
- Developing battery models
- Design, Verify and Deploy BMS algorithms
- Hardware-in-Loop testing
- Summary - Q&A





Software | Electronics | Battery Pack

What is Battery Management System?

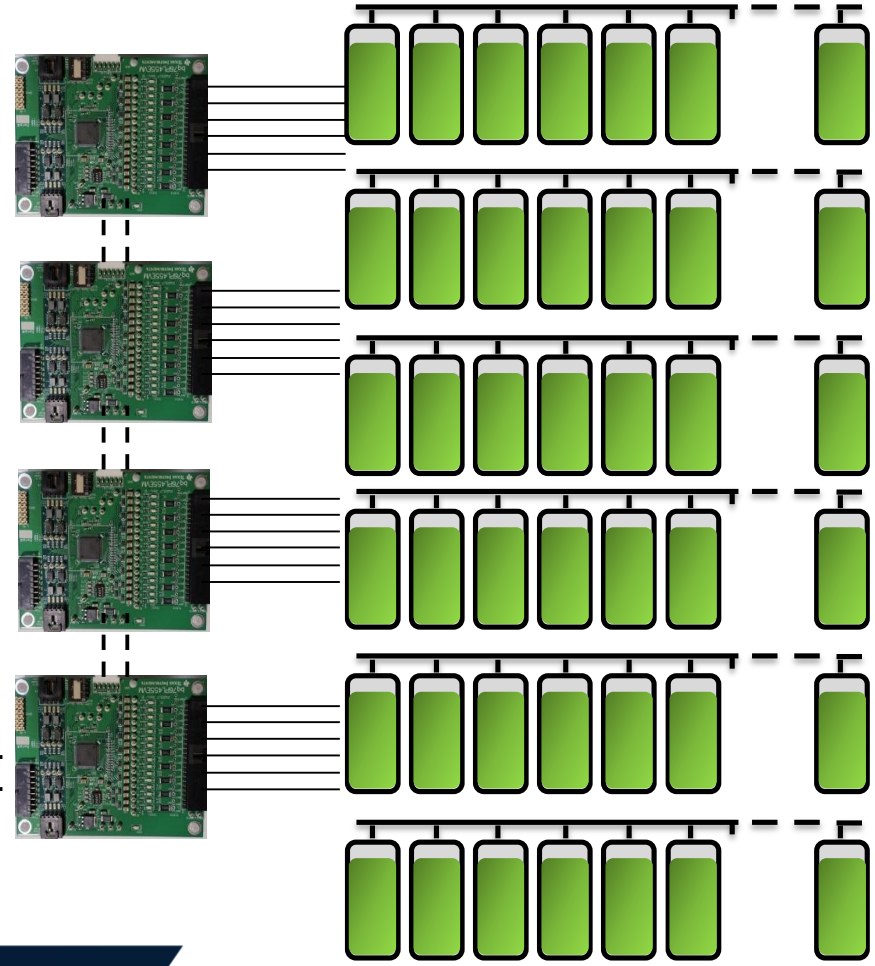
Software | **Electronics** | **Battery Pack**

```
if (((uint32_T)State_Machine_DW.temporalCounter_i3) < 15U) {
  State_Machine_DW.temporalCounter_i3 = (uint8_T)((int32_T)(((ir
    State_Machine_DW.temporalCounter_i3) + 1));
}

if (((uint32_T)State_Machine_DW.is_active_c2_State_Machine) == (
  State_Machine_DW.is_active_c2_State_Machine = 1U;
  State_Machine_DW.is_MainStateMachine = State_Machine_IN_Standb
  *rty_BMS_State = 0;
  State_Machine_DW.MonitorCurrLimMode = MonitorCurrLimModeType_N
  State_Machine_DW.MonitorCellVoltageMode =
    MonitorCellVoltageModeType_NoCellVoltFault;
  State_Machine_DW.Delta = (real32_T)fabs((real_T)((real32_T)
    ((*rtu_Pack_Voltage) - sum_gyOCKAG3(rtu_Cell_Voltages))));
```

Supervisory tasks
SOC estimation
Contactor management
Isolation monitoring
Fault detection and recovery
Thermal Management
Current & Power Limits

Measurement
Cell Diagnostic,
Cell Balancing

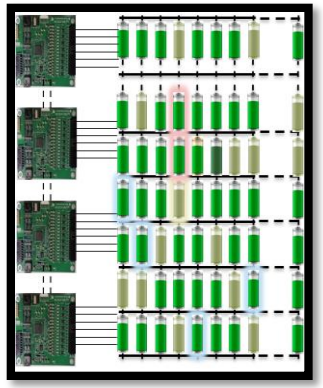


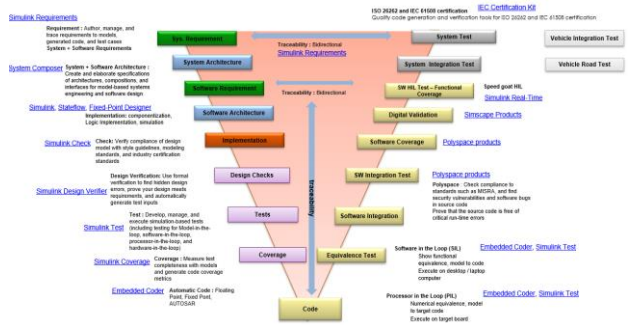
Where do we start ? What to be considered ?

BMS Architecture

Battery Models

Safety and Process





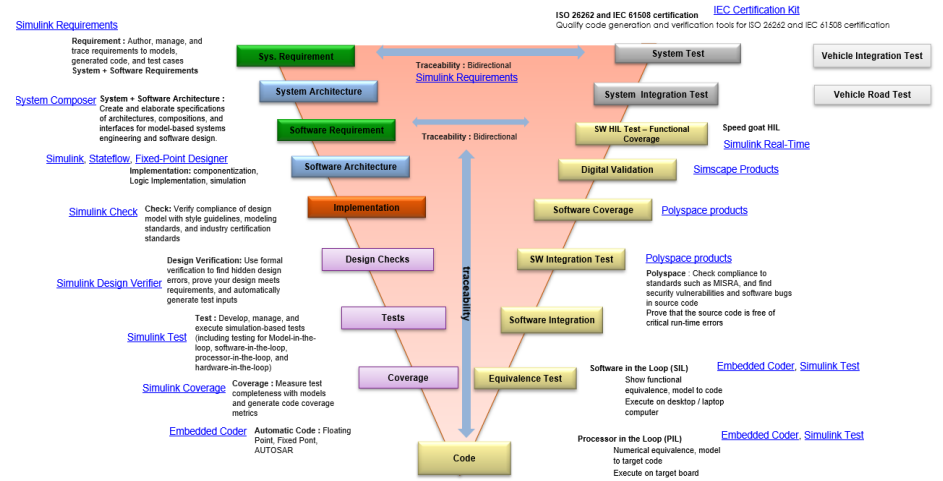
Gain insight into Architecture Development

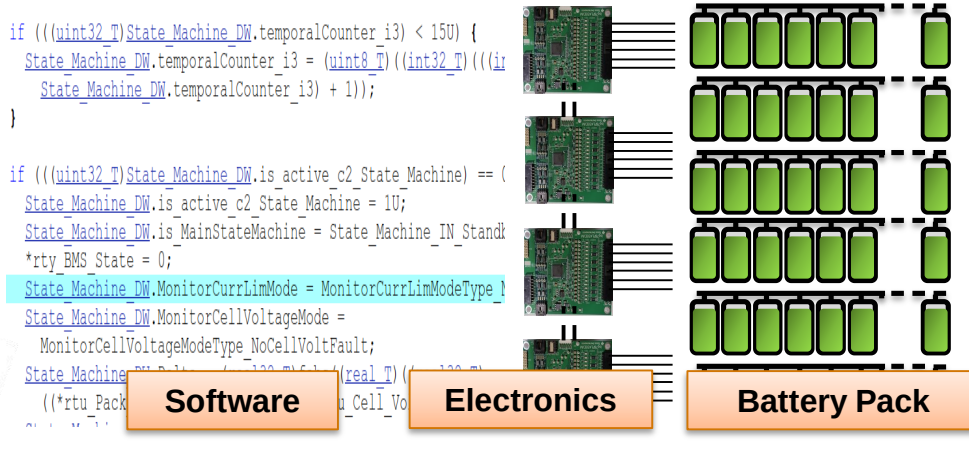
Gain insight into cell behavior and model it

An overview on Integrated Software Development

Agenda

- What is BMS and what engineers worry about?
- **Developing the architecture**
- Developing battery models
- Design, Verify and Deploy BMS algorithms
- Hardware-in-Loop testing
- Summary - Q&A

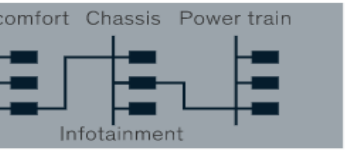
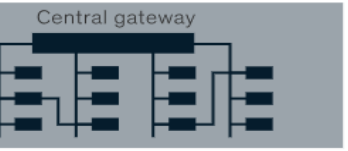
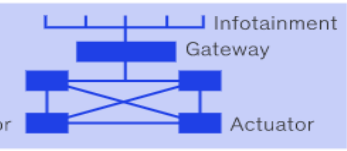




Software | Electronics | Battery Pack

Vehicle EE Architecture

Electrical/electronic architecture is evolving toward a centralized setup.

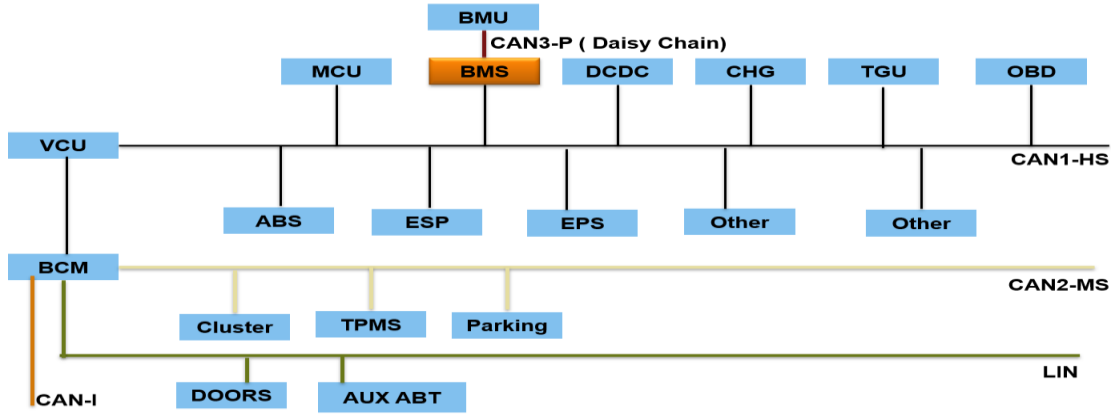
Architecture type	Generation	High-level architecture	Main features
Distributed	1		● Independent engine-control units (ECUs) ● Isolated functions ● Each function has its own ECU (1:1 connection)
	2		● Collaboration of ECUs within 1 domain ● Domains: body/comfort, chassis, power train, and infotainment ● 3 or 4 independent networks ● Limited communication among domains
	3 Today		● Stronger collaboration via central gateway ● Cross-functional connection ● Ability to handle complex functions (eg, adaptive cruise control)
Domain centralized	4		● Central domain controller ● Ability to handle more complex functions ● Consolidation of functions (cost optimization)
Vehicle centralized	5		● Virtual domain ● Limited dedicated hardware ● Ethernet backbone ● High-complexity, high-computing functions

https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/automotive-software-and-electrical-electronic-architecture-implications-for-oems

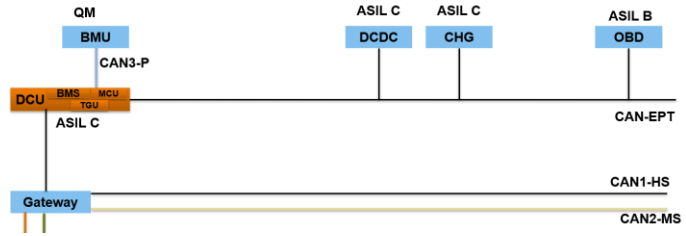
April 25, 2019 | Article

mckinsey.com

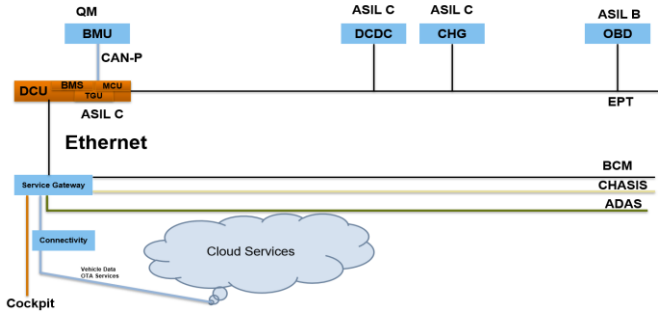
Central Gateway architecture
100s of ECUS



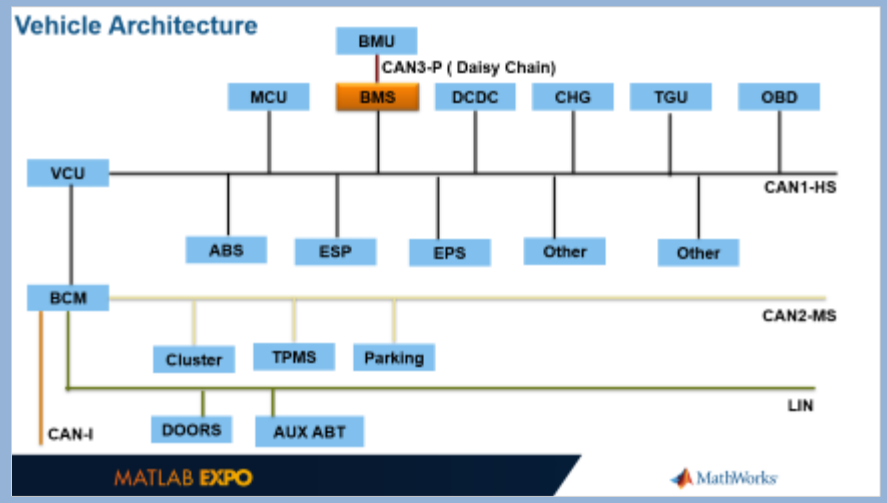
Domain Controller Architecture
Reduced ECU's & Combined Functionality

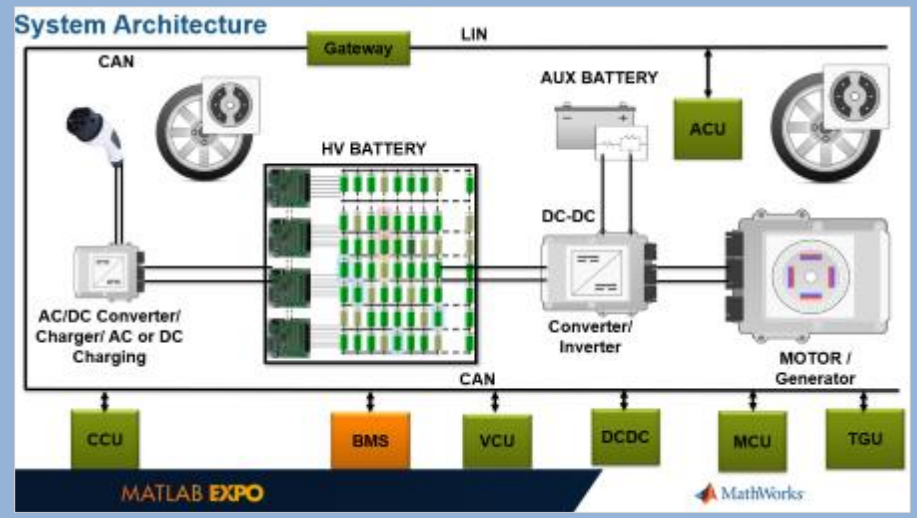


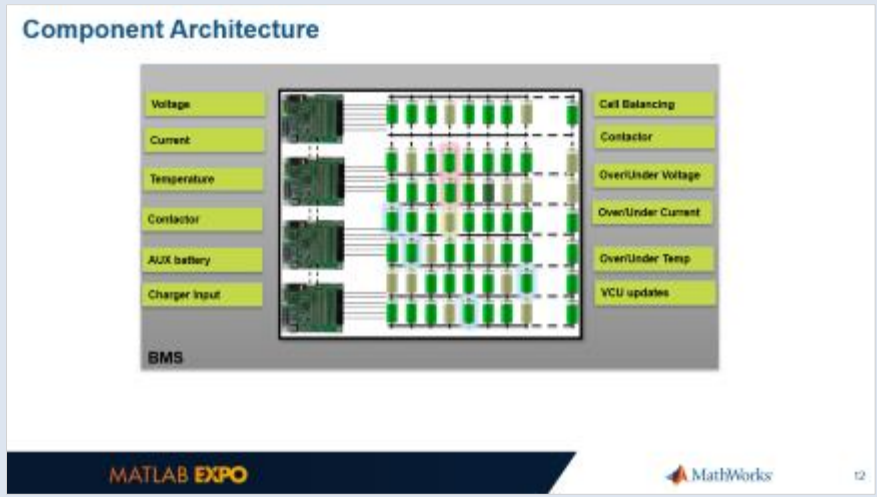
Service-oriented architectures
Cloud Based, Ethernet

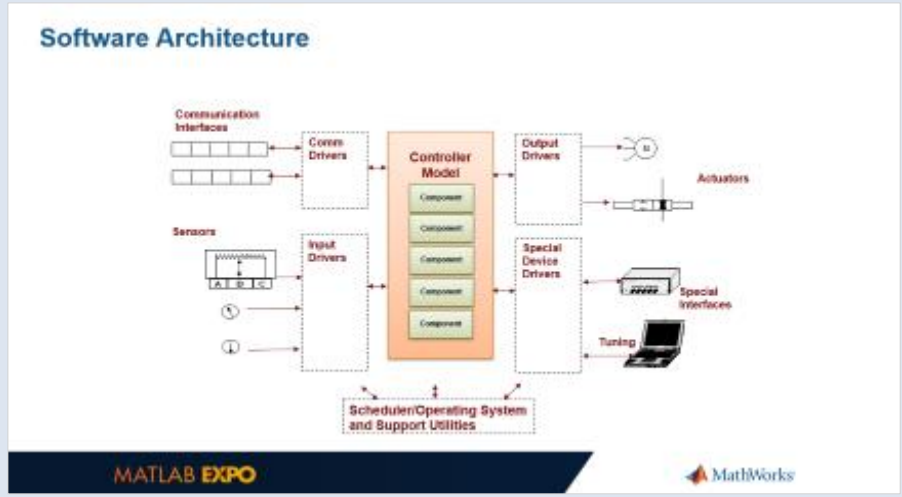


BMS System Architecture : An EV Perspective





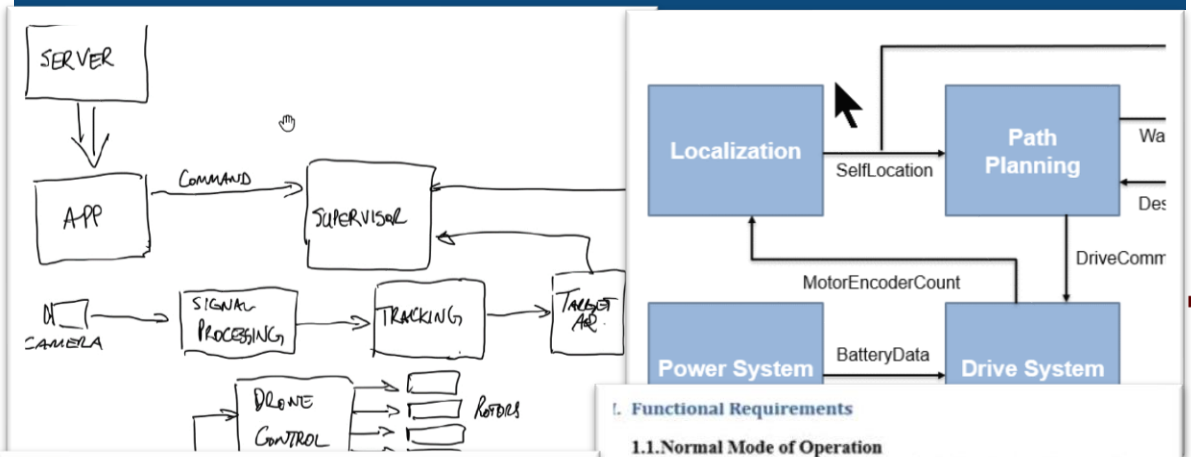


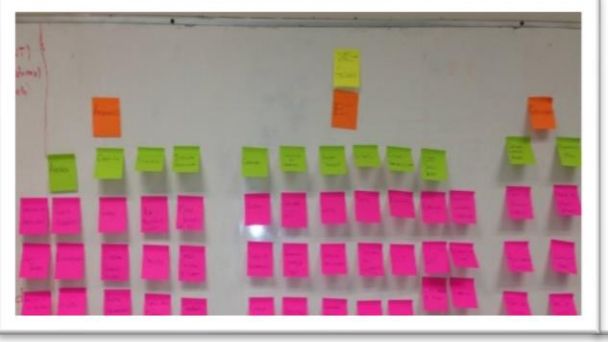


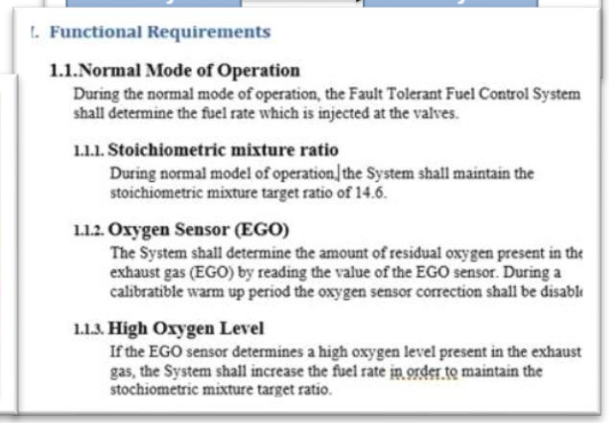
Challenges in Architecture development

The Gap between System Concept and Implementation

Early in the Process
Concepts / Descriptions







1. Functional Requirements

1.1. Normal Mode of Operation

During the normal mode of operation, the Fault Tolerant Fuel Control System shall determine the fuel rate which is injected at the valves.

1.1.1. Stoichiometric mixture ratio

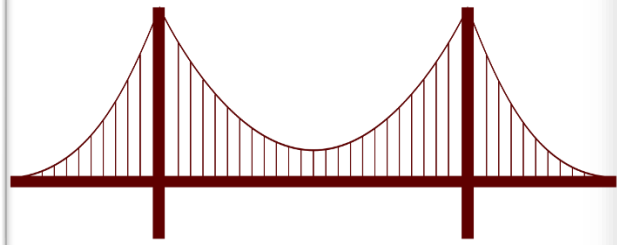
During normal model of operation, the System shall maintain the stoichiometric mixture target ratio of 14.6.

1.1.2. Oxygen Sensor (EGO)

The System shall determine the amount of residual oxygen present in the exhaust gas (EGO) by reading the value of the EGO sensor. During a calibratible warm up period the oxygen sensor correction shall be disabl

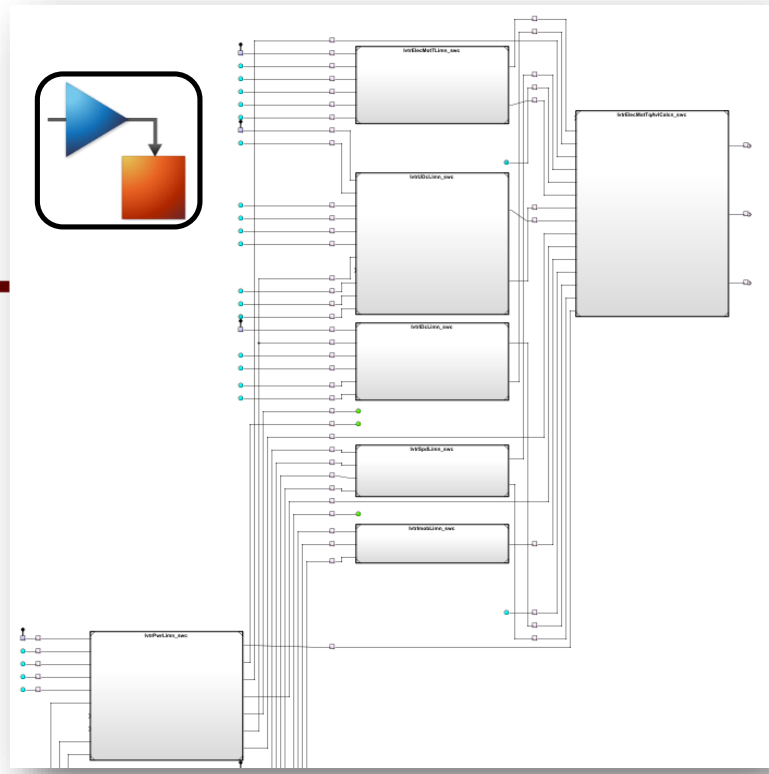
1.1.3. High Oxygen Level

If the EGO sensor determines a high oxygen level present in the exhaust gas, the System shall increase the fuel rate in order to maintain the stochiometric mixture target ratio.



NEED
Traceability
Synchronization
Analysis & Simulation

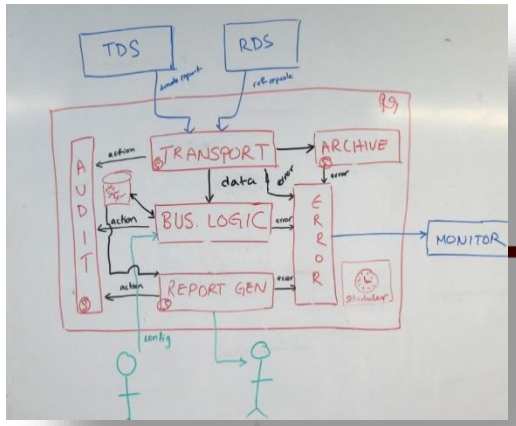
Later in the Process
Models



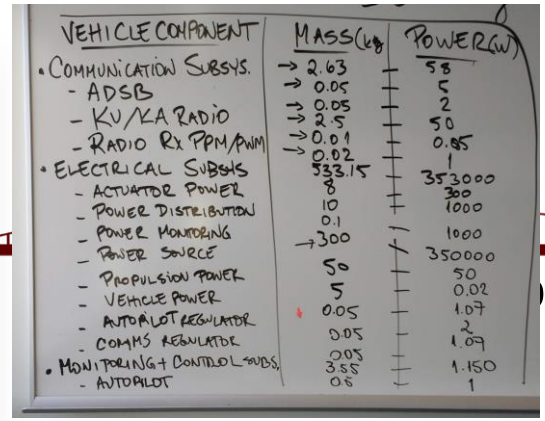
Challenges in Architecture development

Expectations from the Architecture models

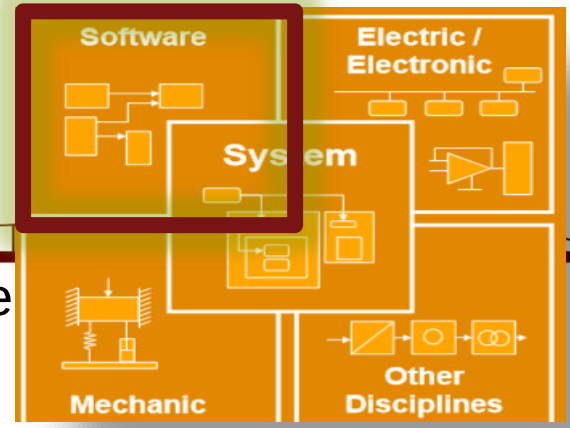
Be Intuitive



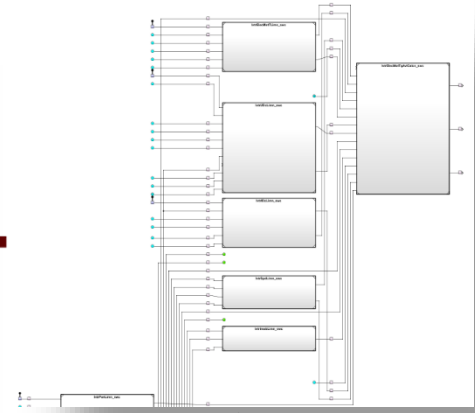
Facilitate Analysis



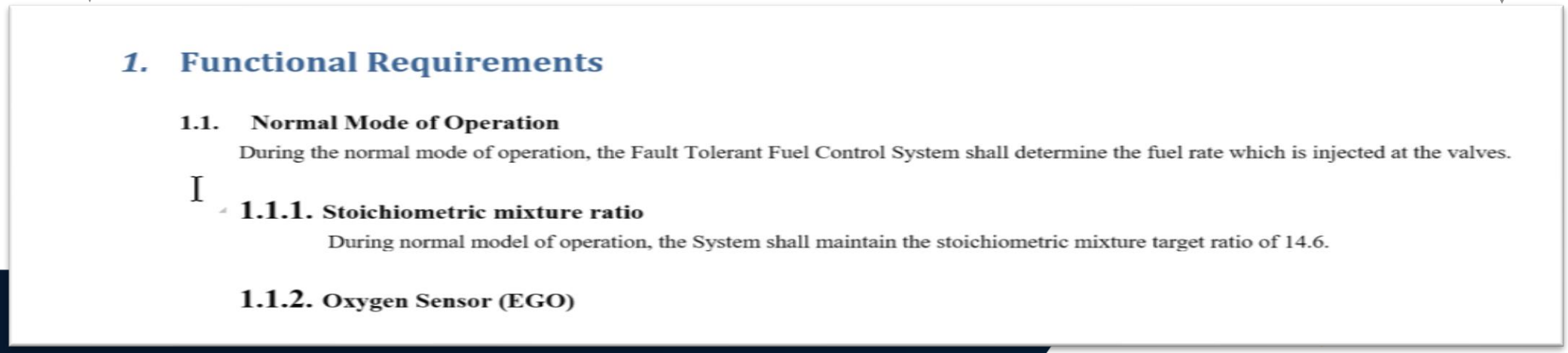
Tackle Complexity



Enable Implementation



Traceability



1. Functional Requirements

1.1. Normal Mode of Operation

During the normal mode of operation, the Fault Tolerant Fuel Control System shall determine the fuel rate which is injected at the valves.

1.1.1. Stoichiometric mixture ratio

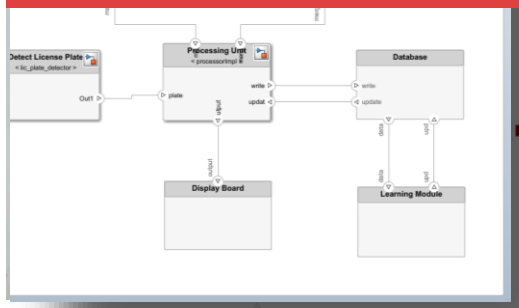
During normal model of operation, the System shall maintain the stoichiometric mixture target ratio of 14.6.

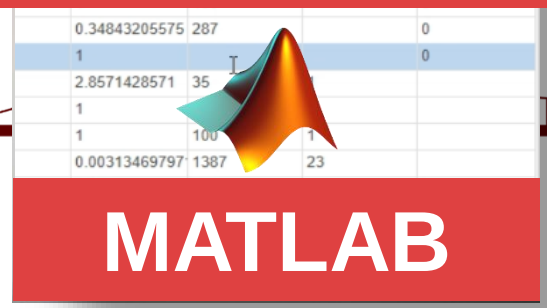
1.1.2. Oxygen Sensor (EGO)

Solution : System Composer Ecosystem

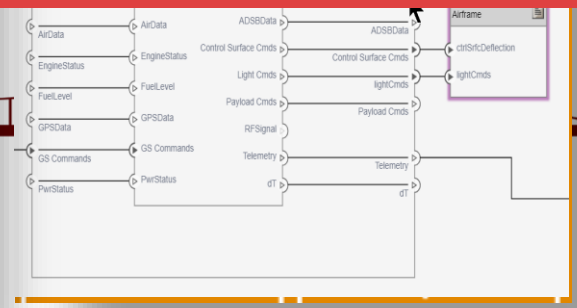
Be Intuitive Facilitate Analysis Tackle Complexity Enable Implementation

System Composer

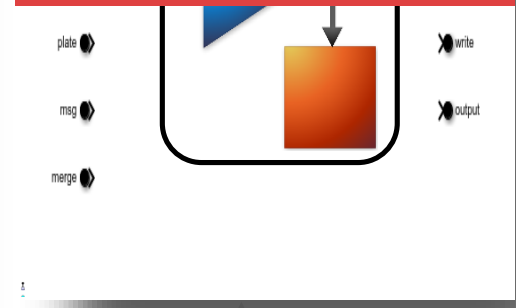




MATLAB



Simulink



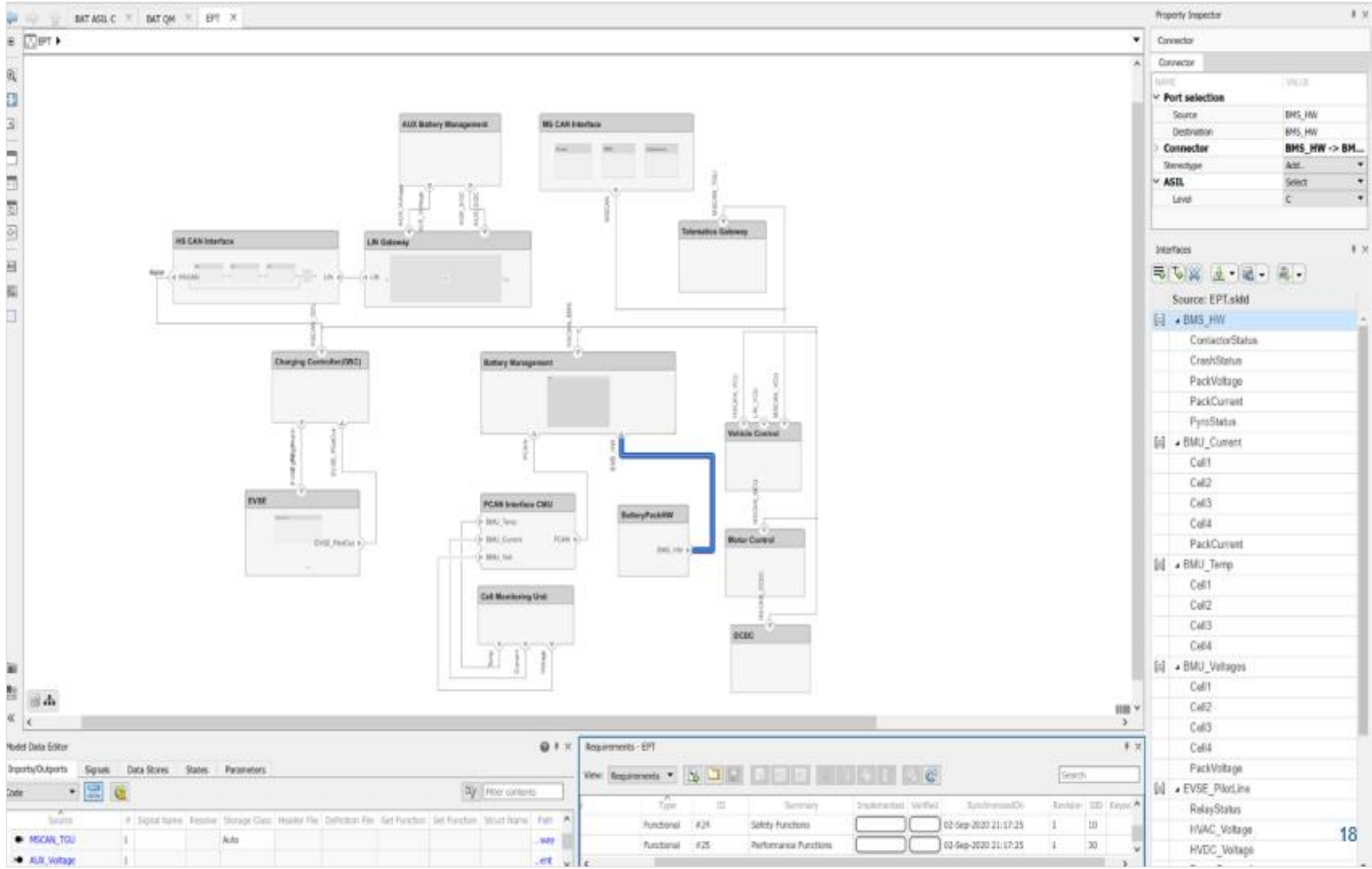
Requirements Coverage Reporting and Impact Analysis

Simulink Requirements

Developing the Architecture : BMS System

System Architecture
- HW and SW Interface
- Feature Allocation
- Analysis

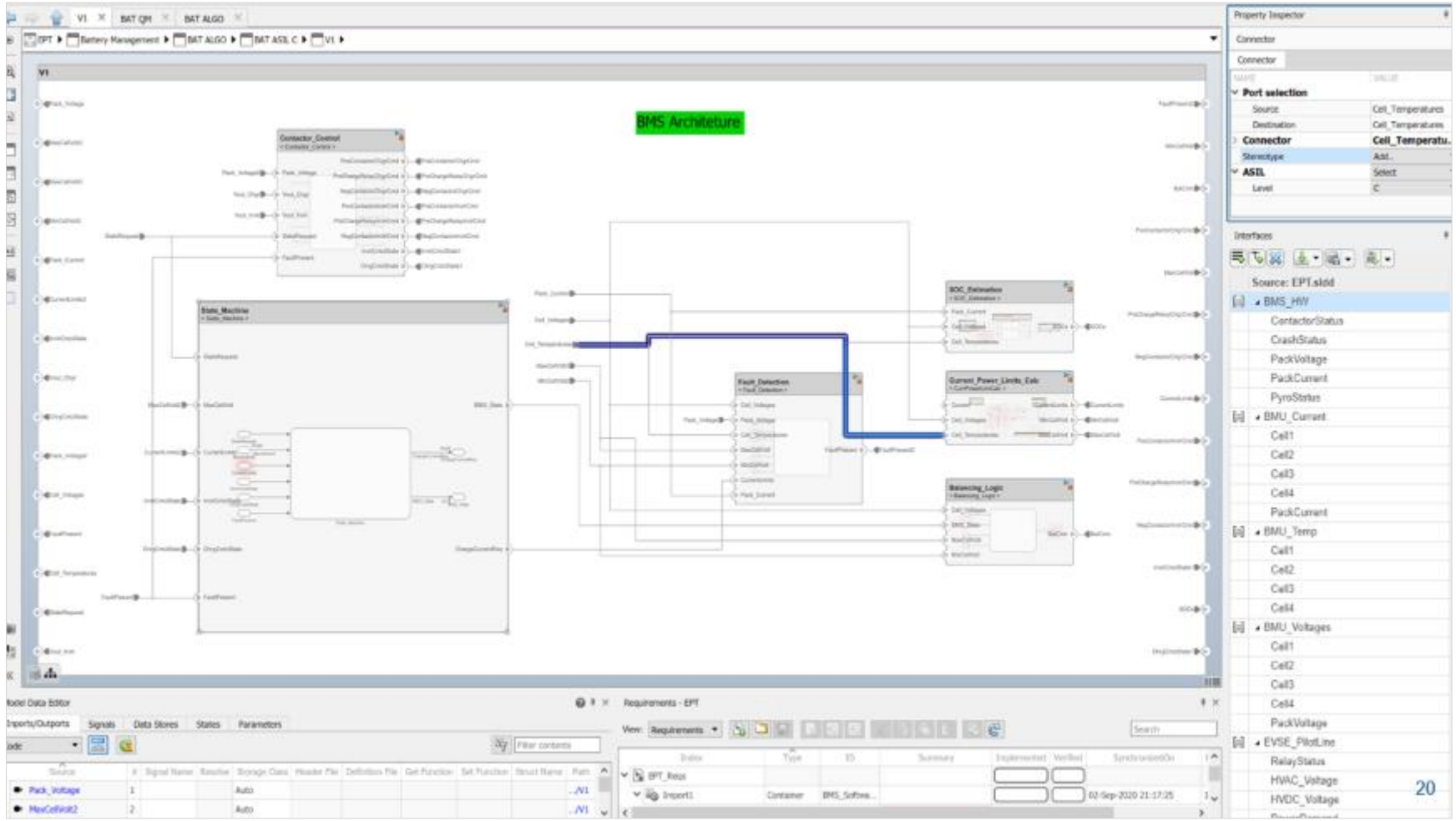
System Composer
- Define Interfaces
- Different architectural Views
- Components Definitions
- BUS and Signal Definitions
- Requirement traceability



Developing the Architecture : BMS Software

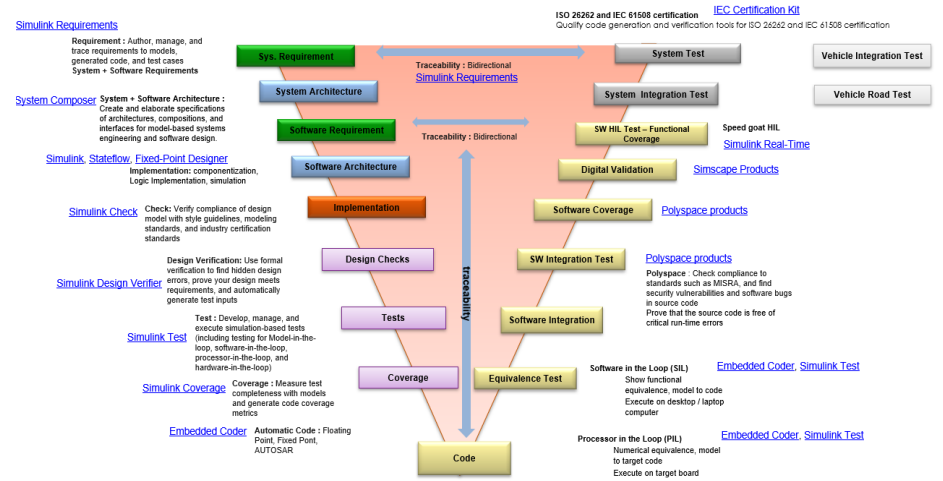
System Composer

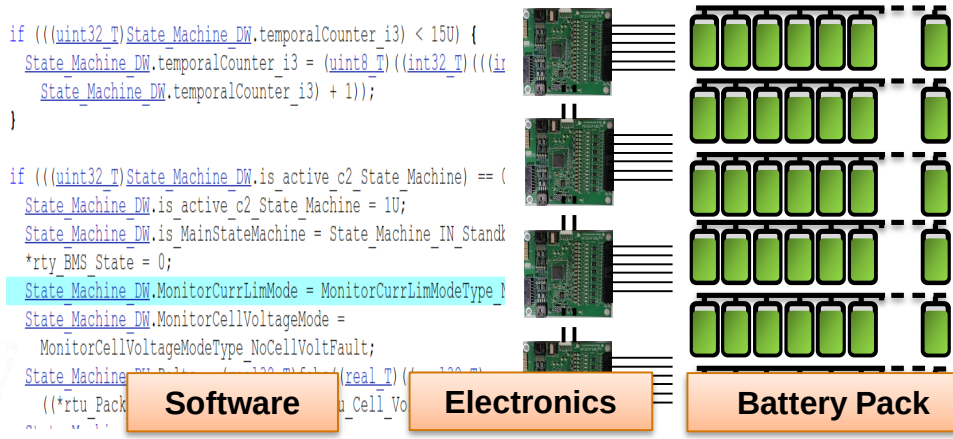
- Software Interfaces
- Shared signals
- SWC Components
- Data Dictionary
- Requirement Traceability



Agenda

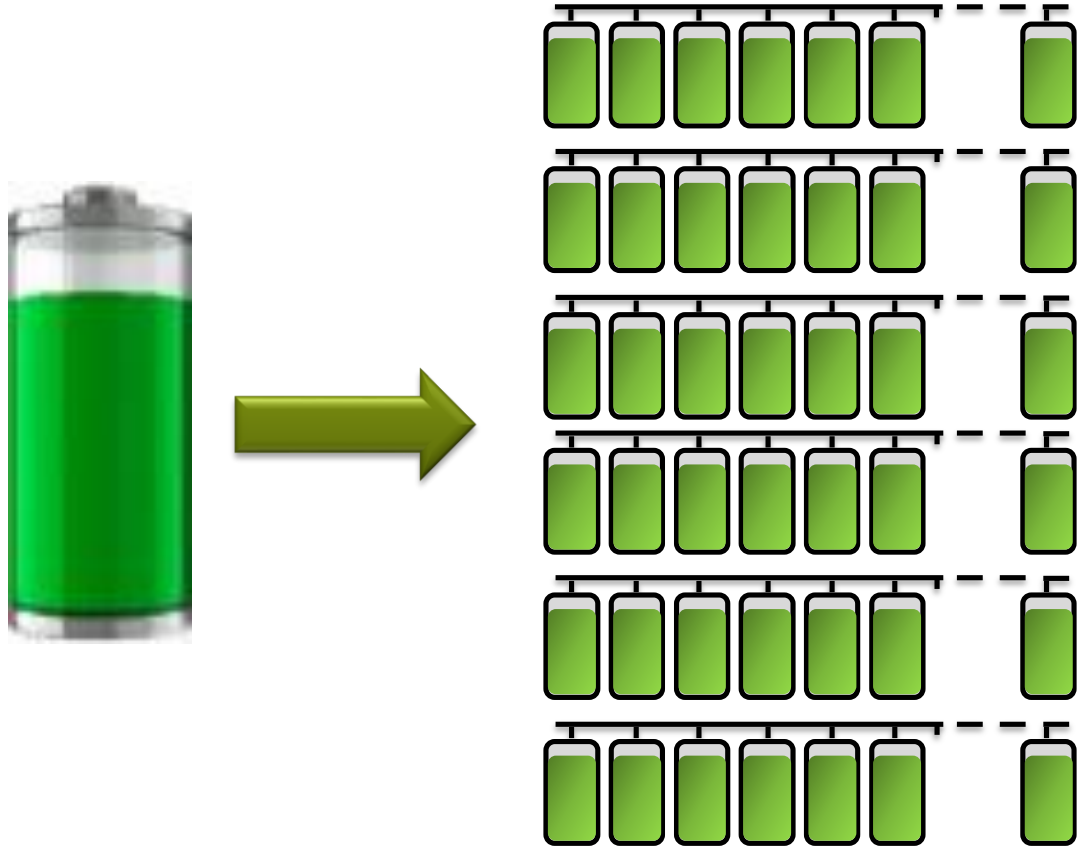
- What is BMS and what engineers worry about?
- Developing the architecture
- **Developing battery models**
- Design, Verify and Deploy BMS algorithms
- Hardware-in-Loop testing
- Summary - Q&A





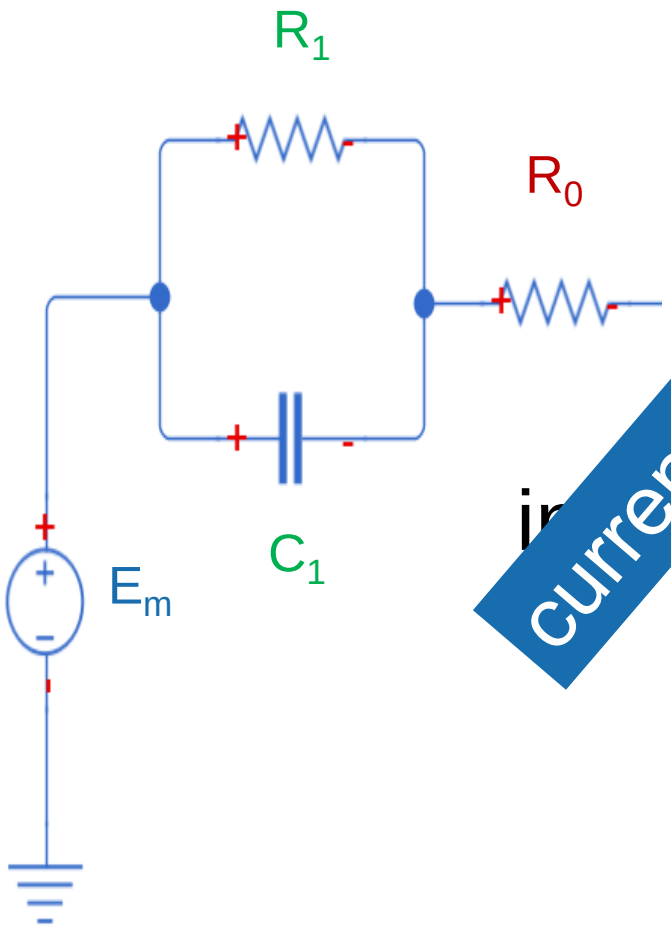
Software | Electronics | Battery Pack

Model the cell behavior

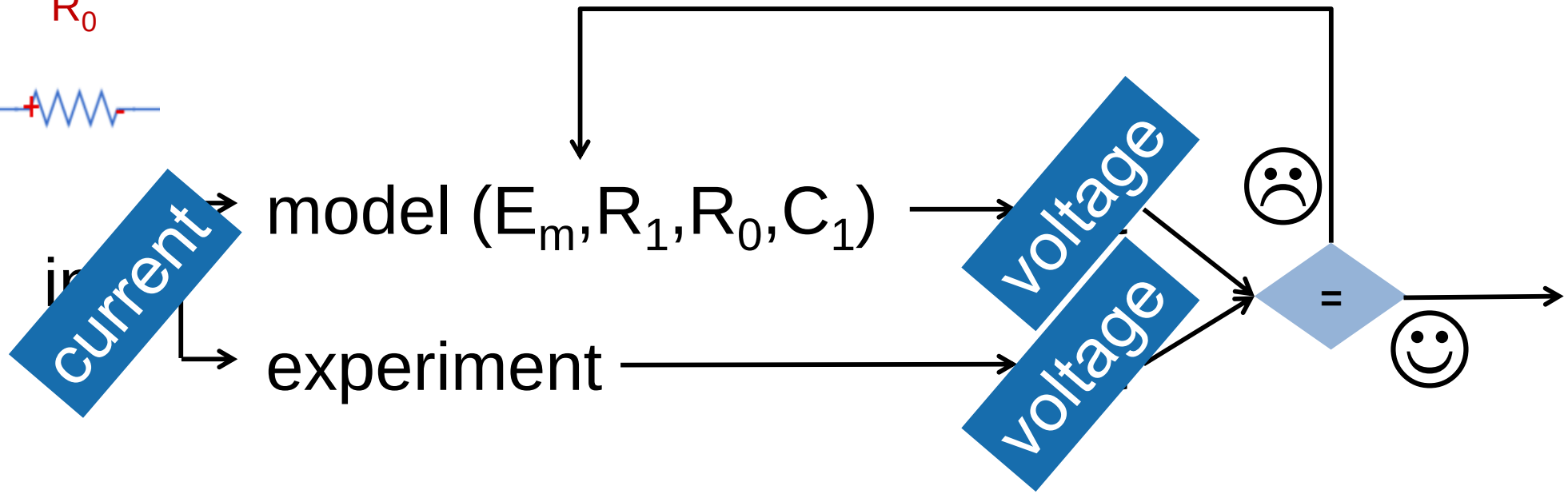


Gain insight into cell behavior and model it

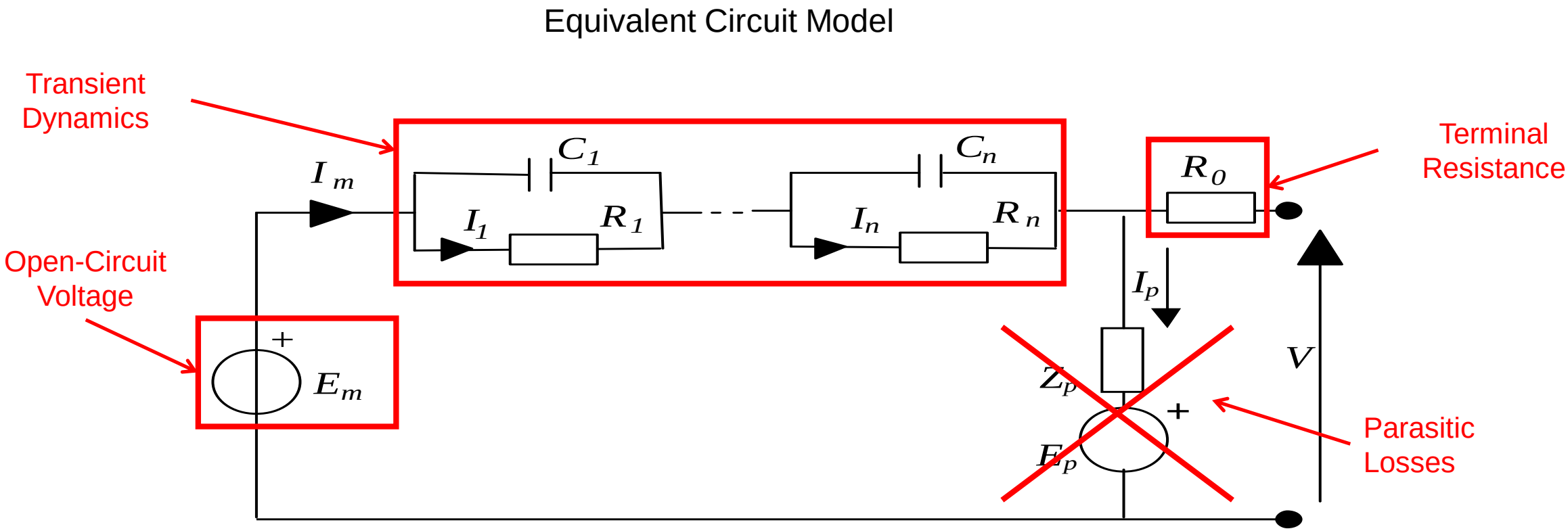
Why do we need to model a battery ?



Goal: Model should match real data as closely as possible



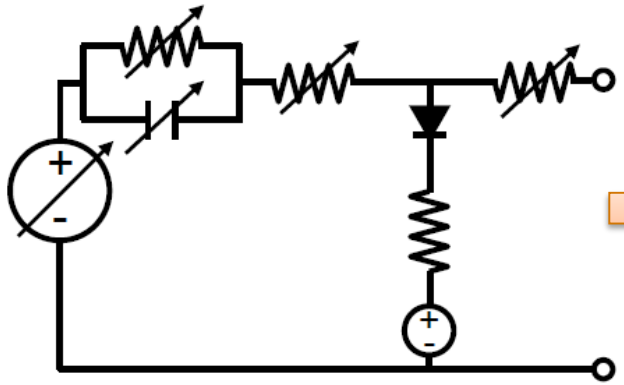
Battery Model



$$[E_m\ R_x\ C_x] = f(SOC, Temperature\ldots)$$

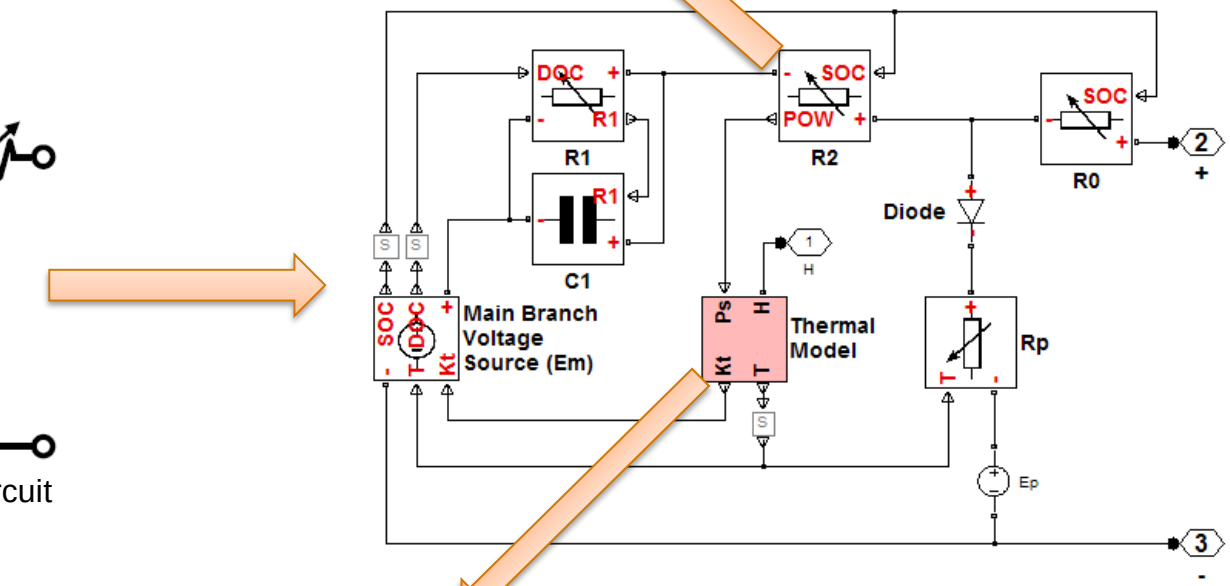
Battery cell modeling as RC equivalent circuit

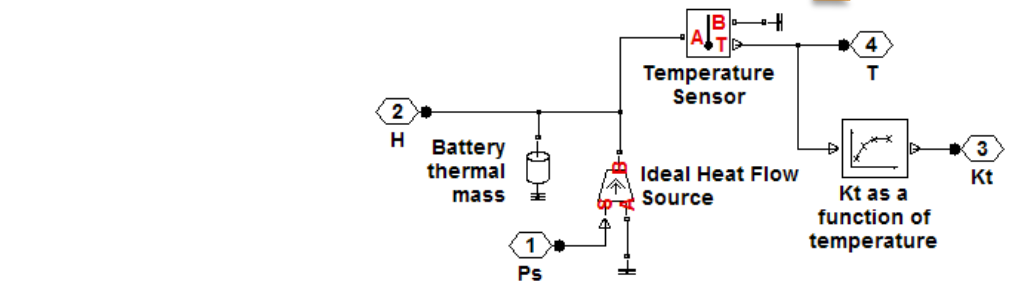
- 1RC Equivalent circuit representation of Battery cell
- Resistors, capacitor, and voltage source are dependent upon SOC and temperature



Battery cell equivalent discharge circuit

```
equations
  v == i*R20*exp(A21*(1-
  pow == v*i;
end
```

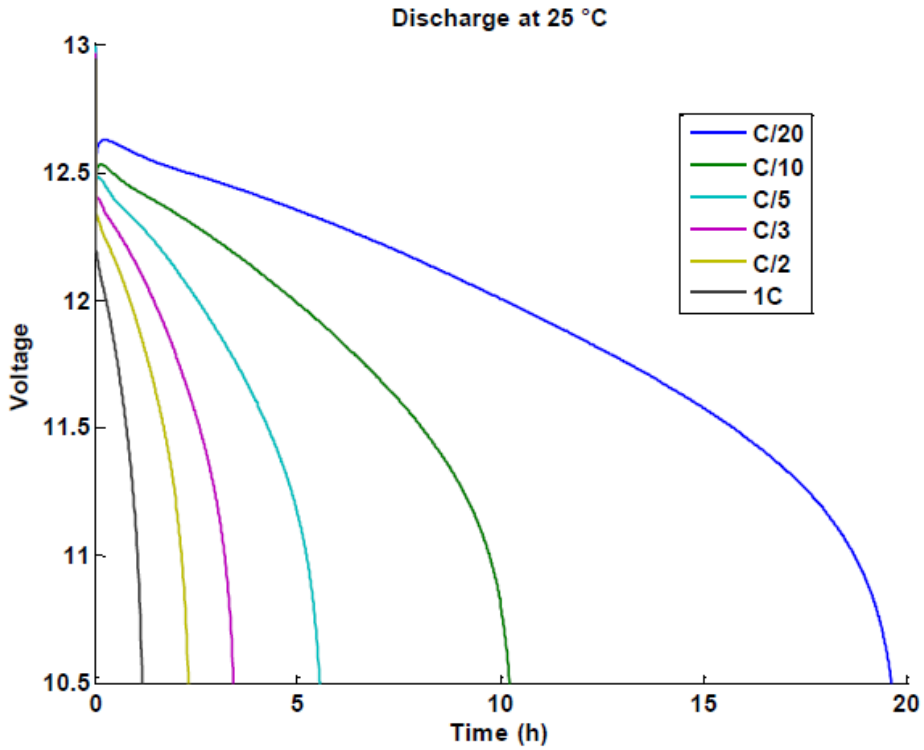




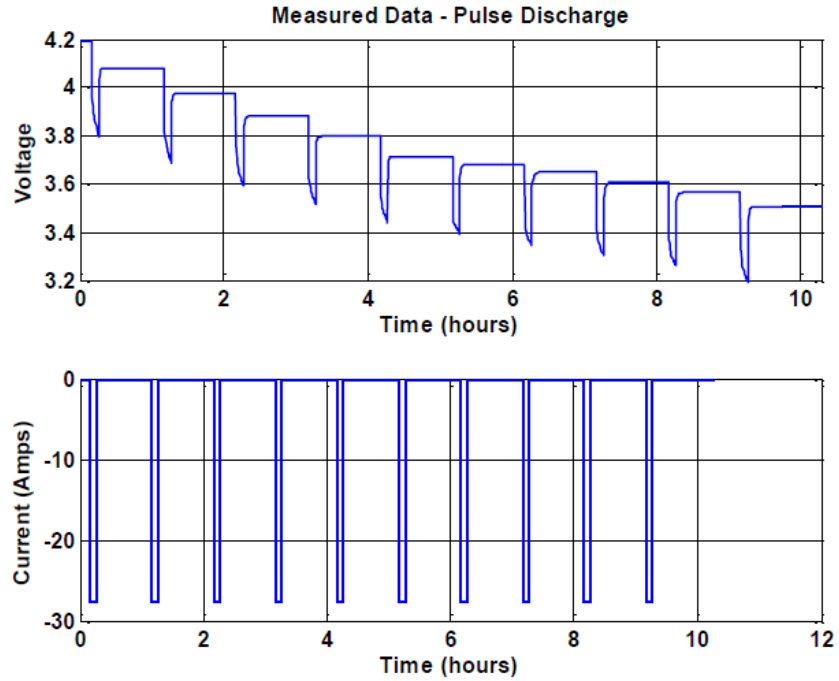
Experimental Data

Battery data is collected by conducting a series of tests with the battery

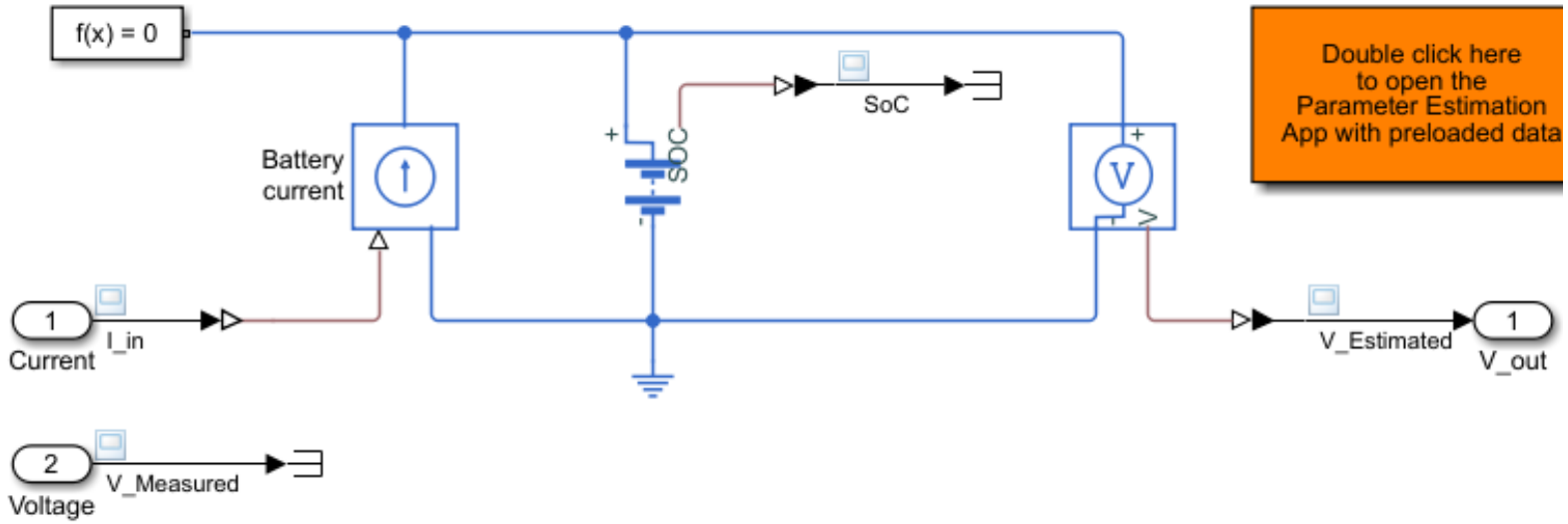
- Used to determine battery capacity
  - Multiple Temperatures
  - Multiple Currents

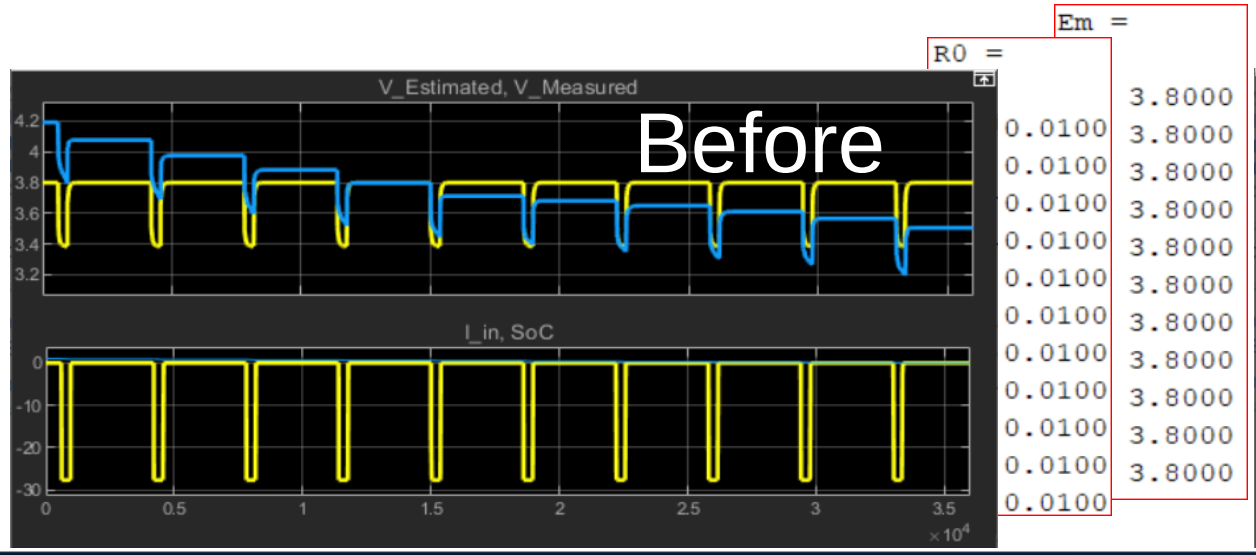


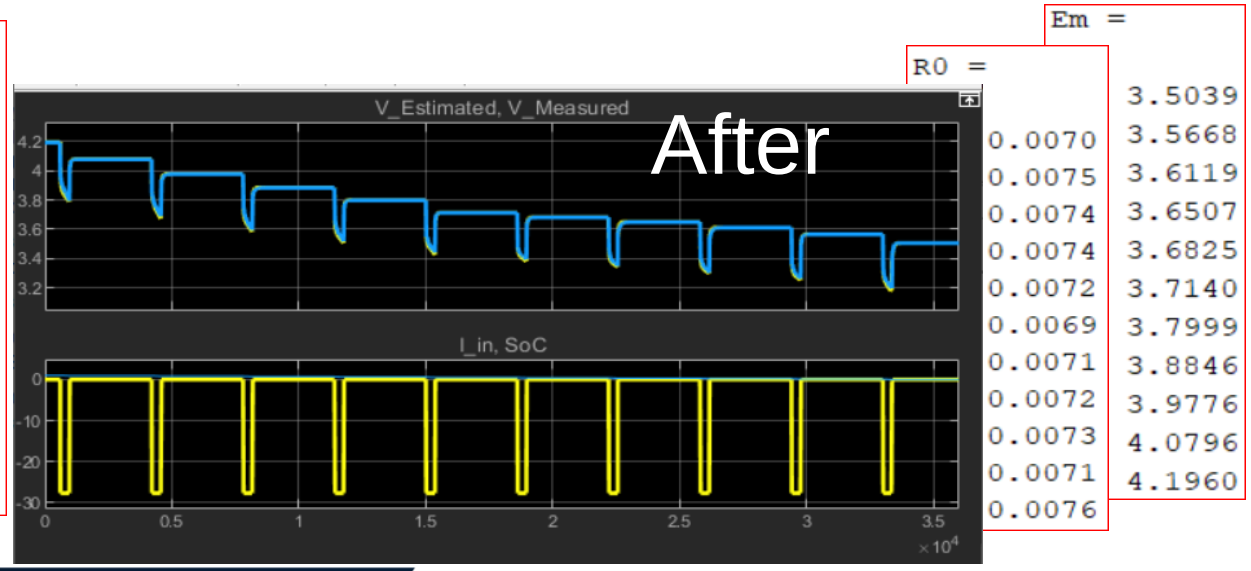
- Used to determine battery dynamics
  - Range of SOC
  - Multiple Temperatures
  - Multiple Currents
  - Discharge and Charge Curves



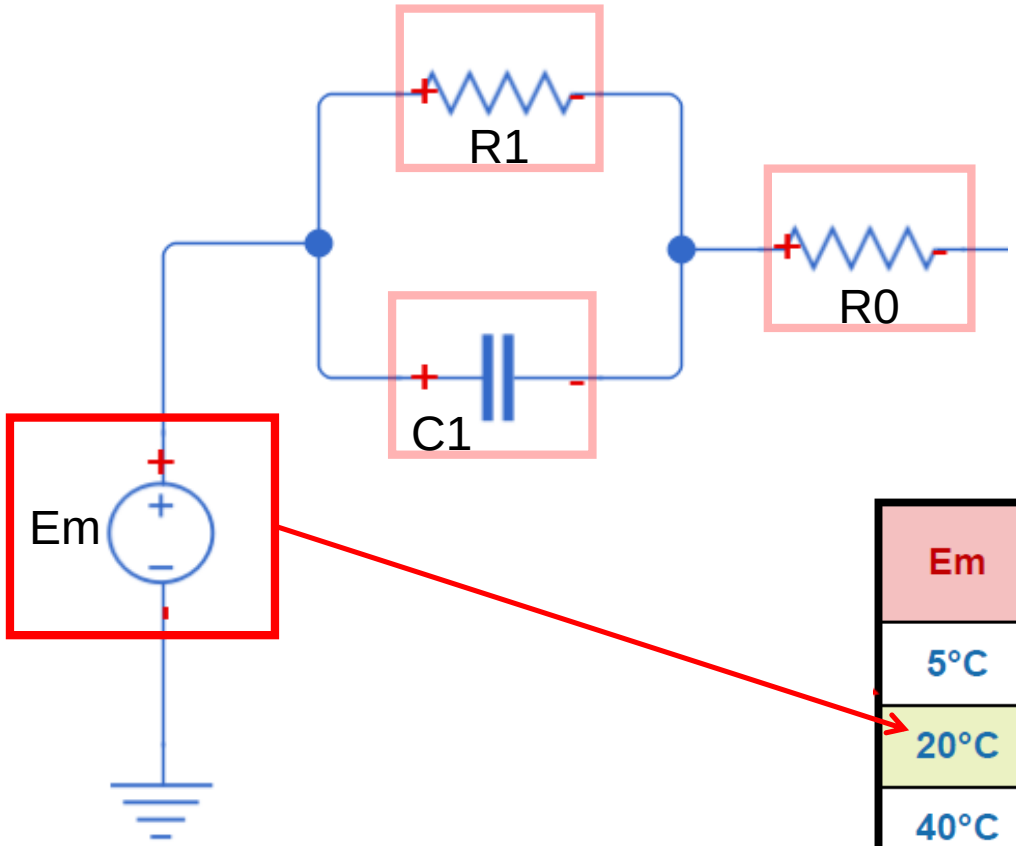
Parameter Estimation Process







Look-up Tables



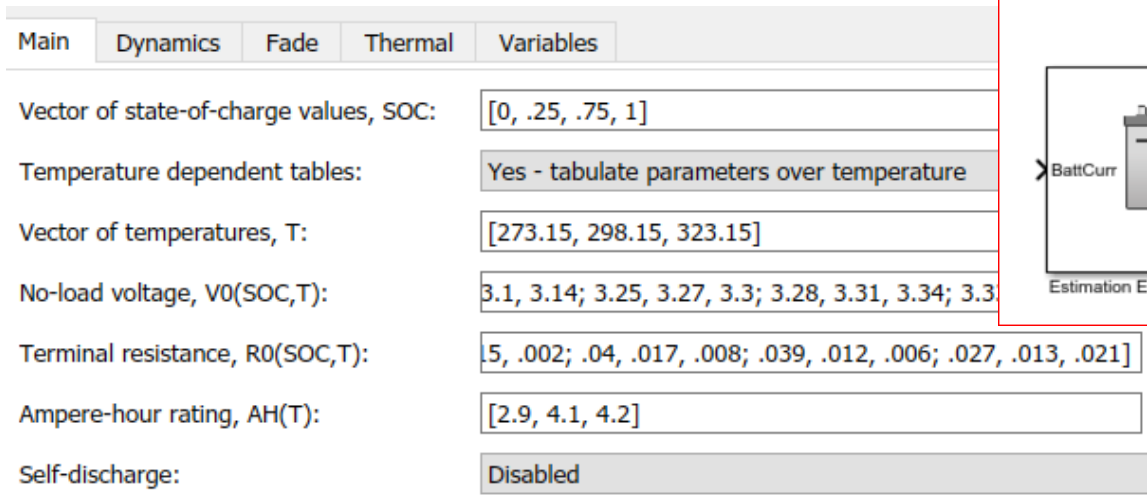
Repeat parameter estimation for each Temperature break-point in LUT

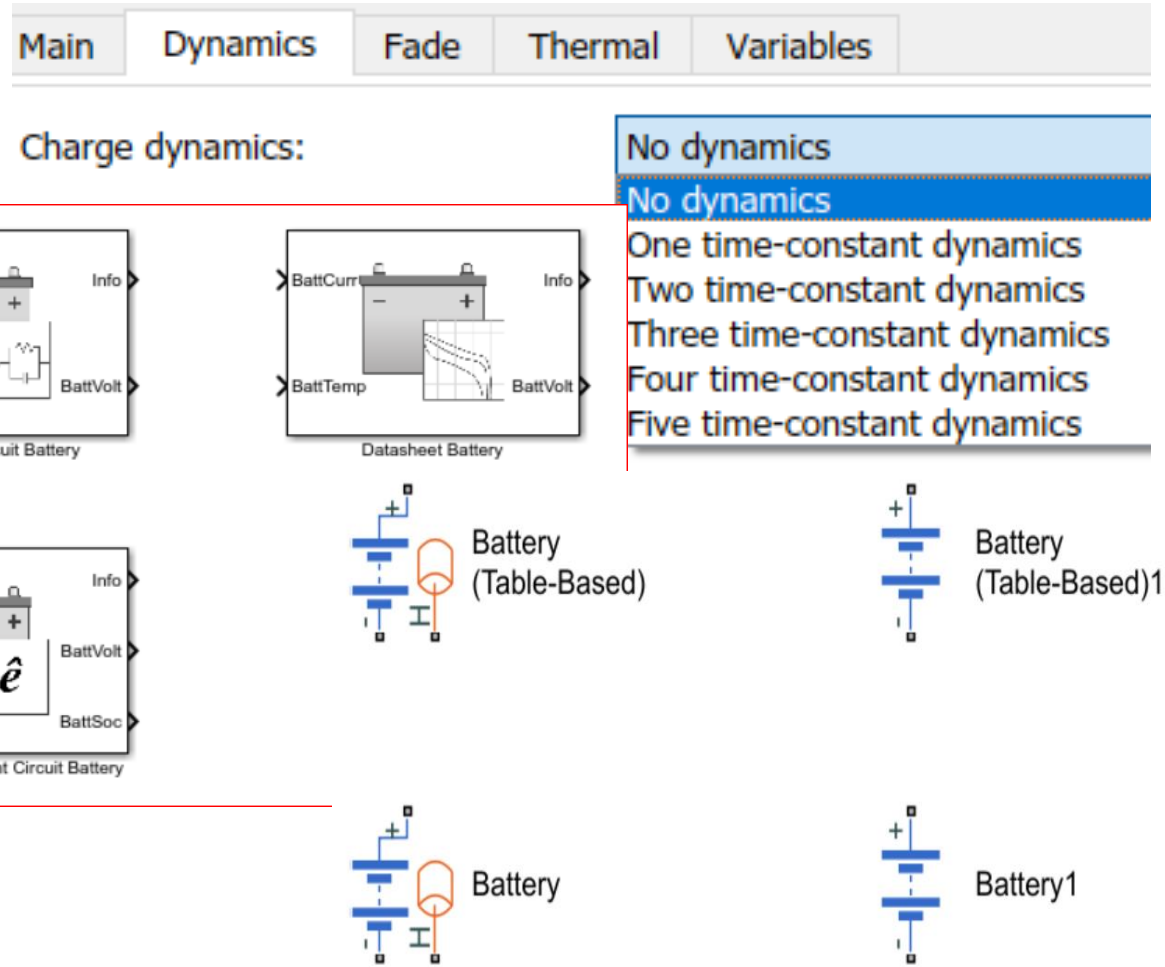
- Values will characterize the battery performance

Em	SOC 1	SOC 0.9	SOC 0.8	…	SOC 0
5°C	4.20 V	4.10 V	4.05 V	…	3.50 V
20°C	4.18 V	4.07 V	4.02 V	…	3.49 V
40°C	4.15 V	4.02 V	3.97 V	…	3.43 V

Battery Cell Blocks in Simulink and Simscape

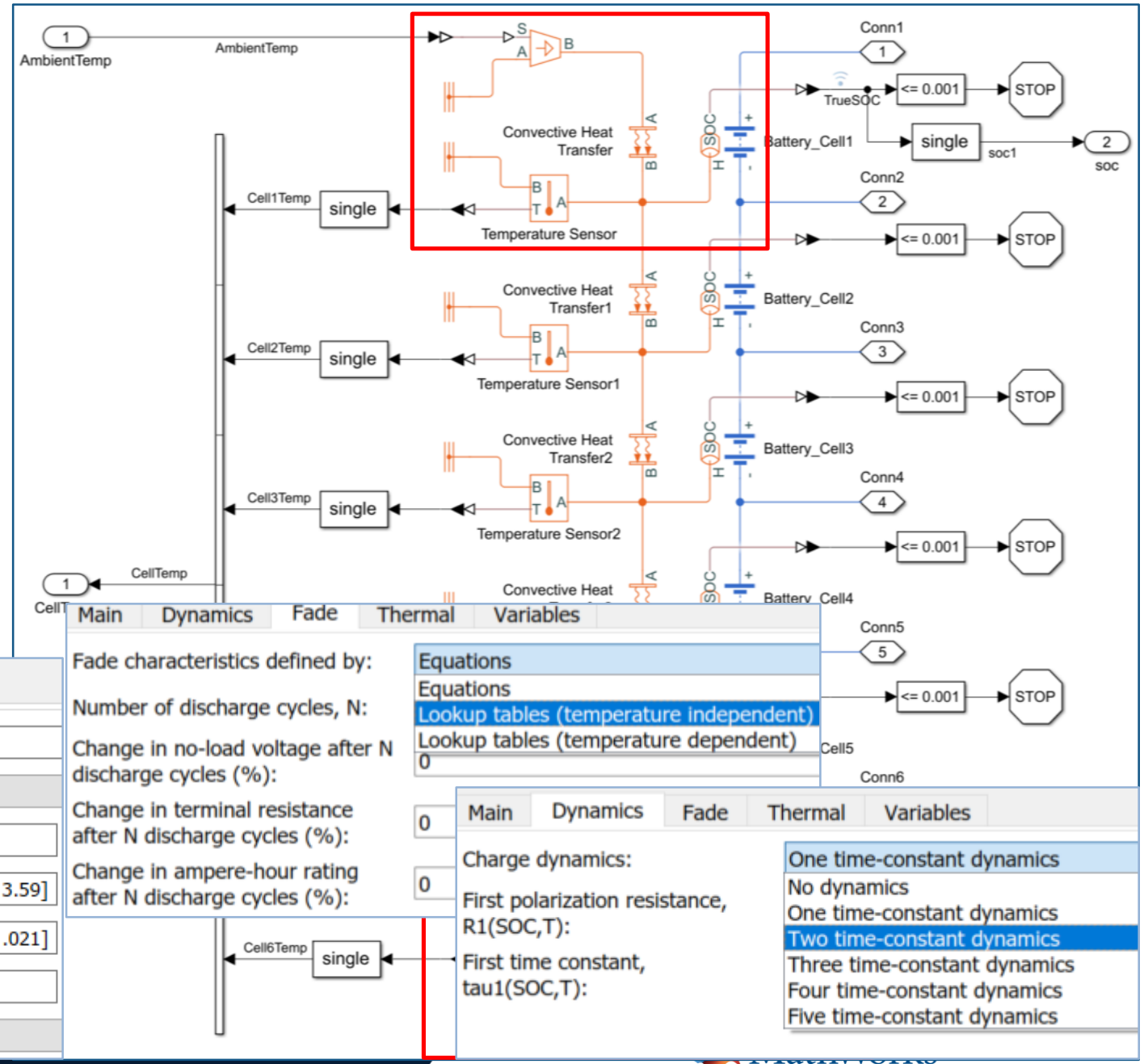
- Chose block for fidelity and simulation speed
- Parameterize as function of SOC & Temperature
- Add thermal and fade effects
- Create custom battery blocks using Simscape language or Simulink

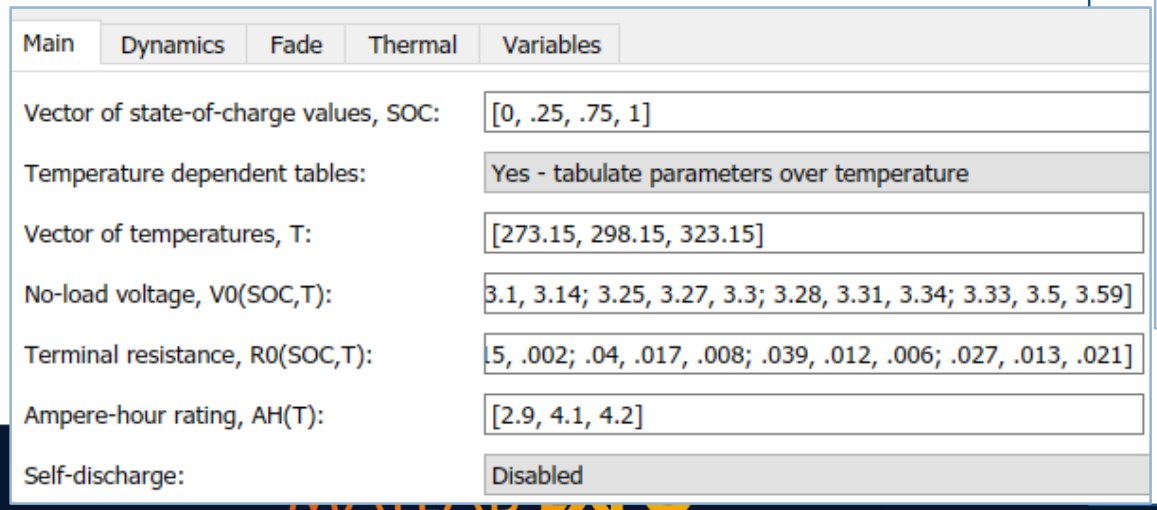




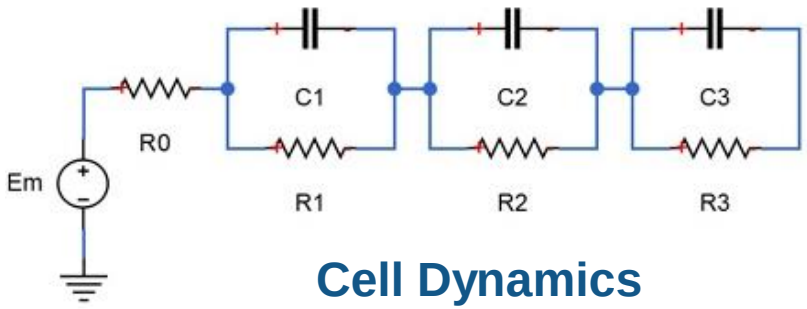
Model Battery Pack

- Connect cells in series to build battery pack models
- Simulate electrical and thermal behavior of battery pack
- Parameterize as function of SOC & Temperature
- Simulate capacity fade effects
- Choose model fidelity

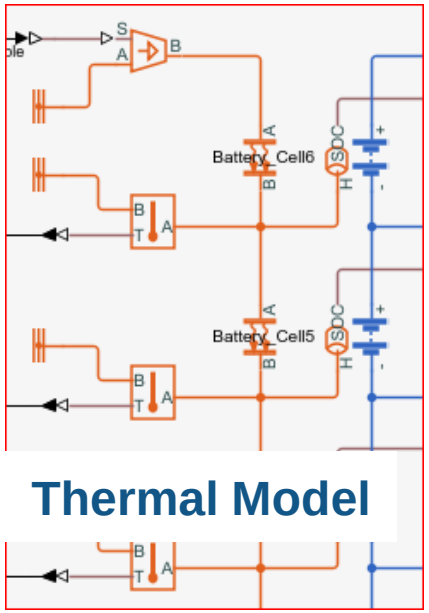




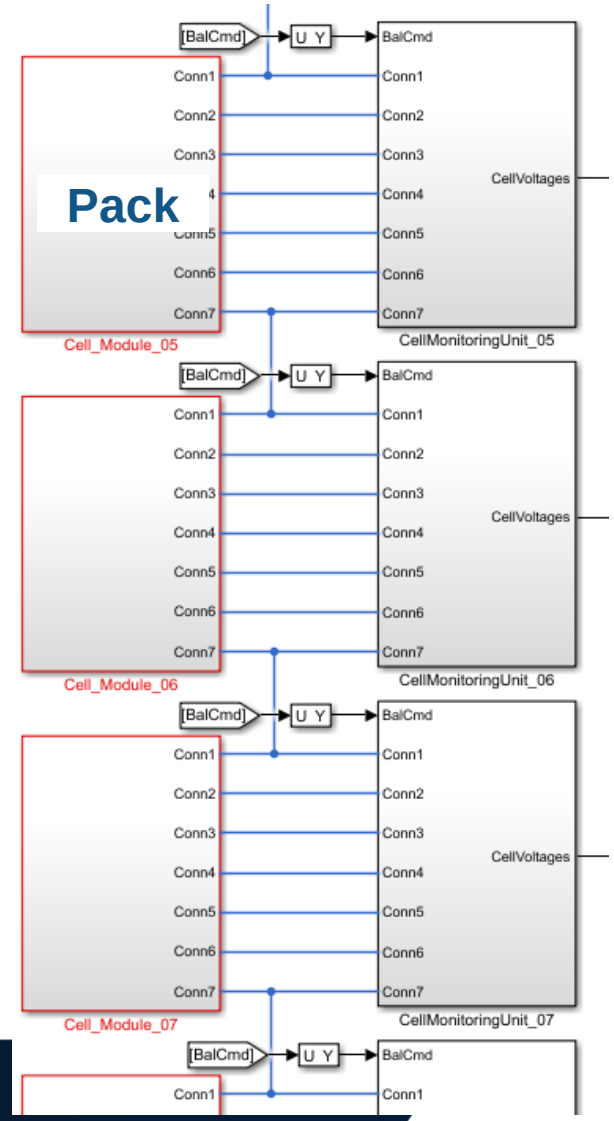
Start with Simulation
Battery Cell ←→ Large Battery Pack

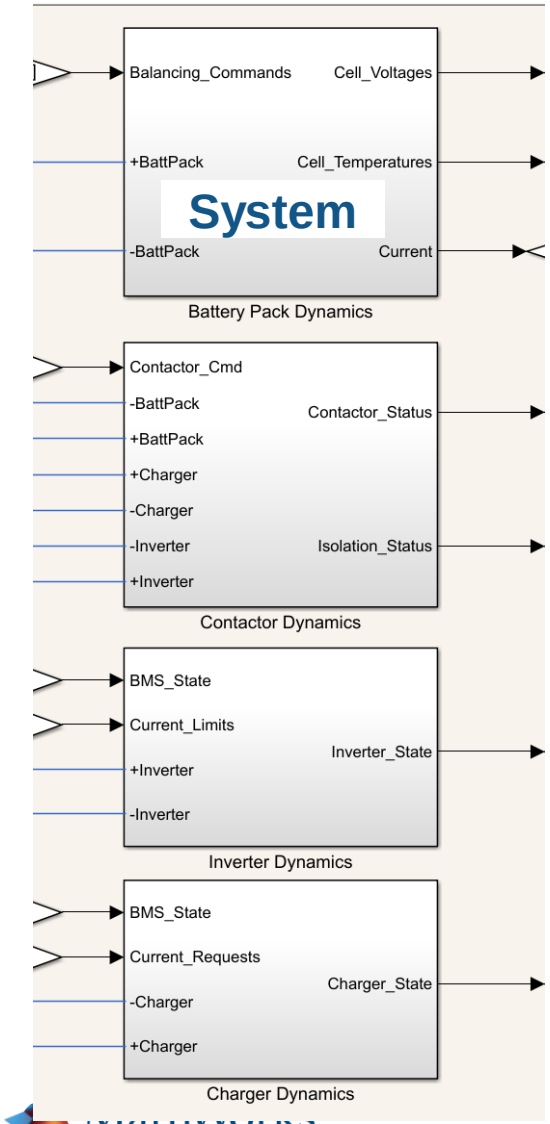


Cell Dynamics



Thermal Model

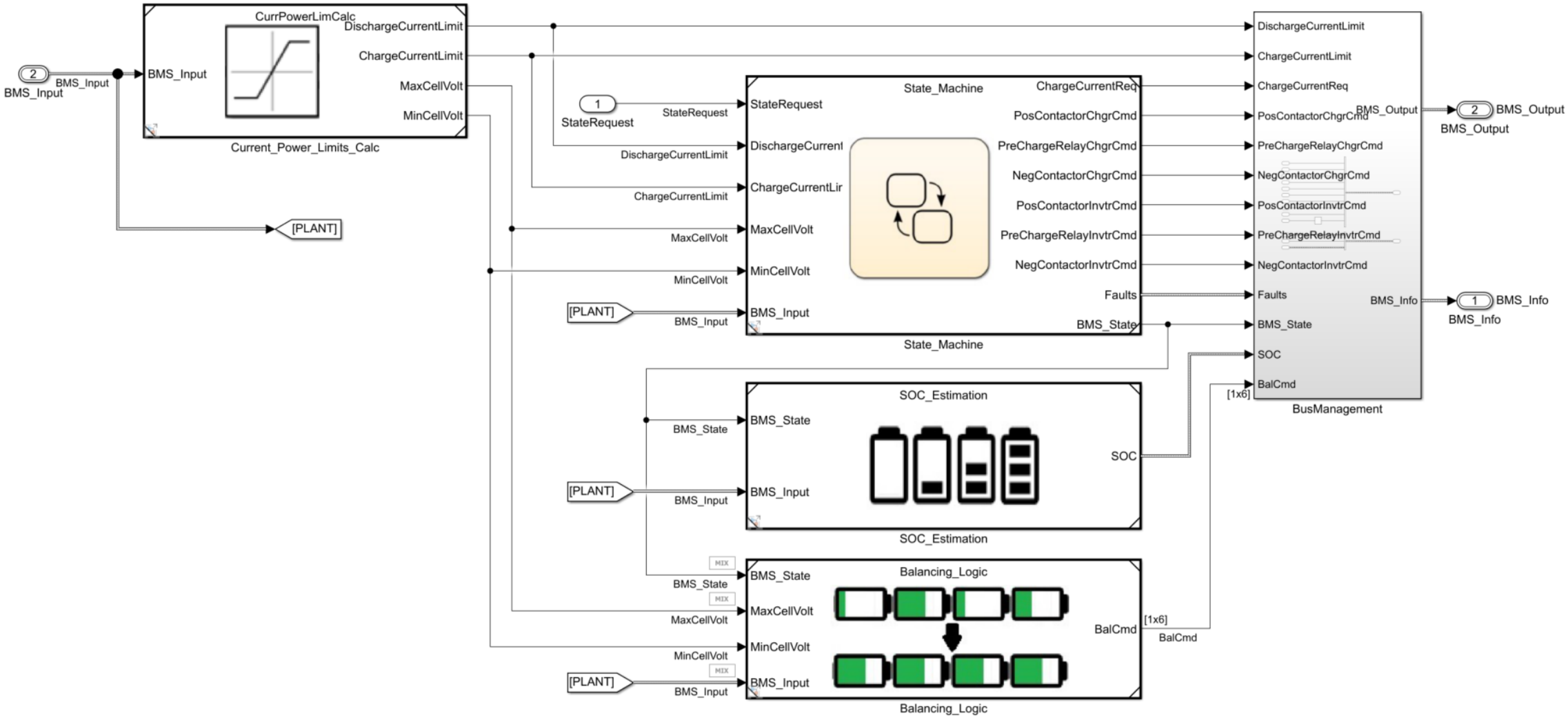




Agenda

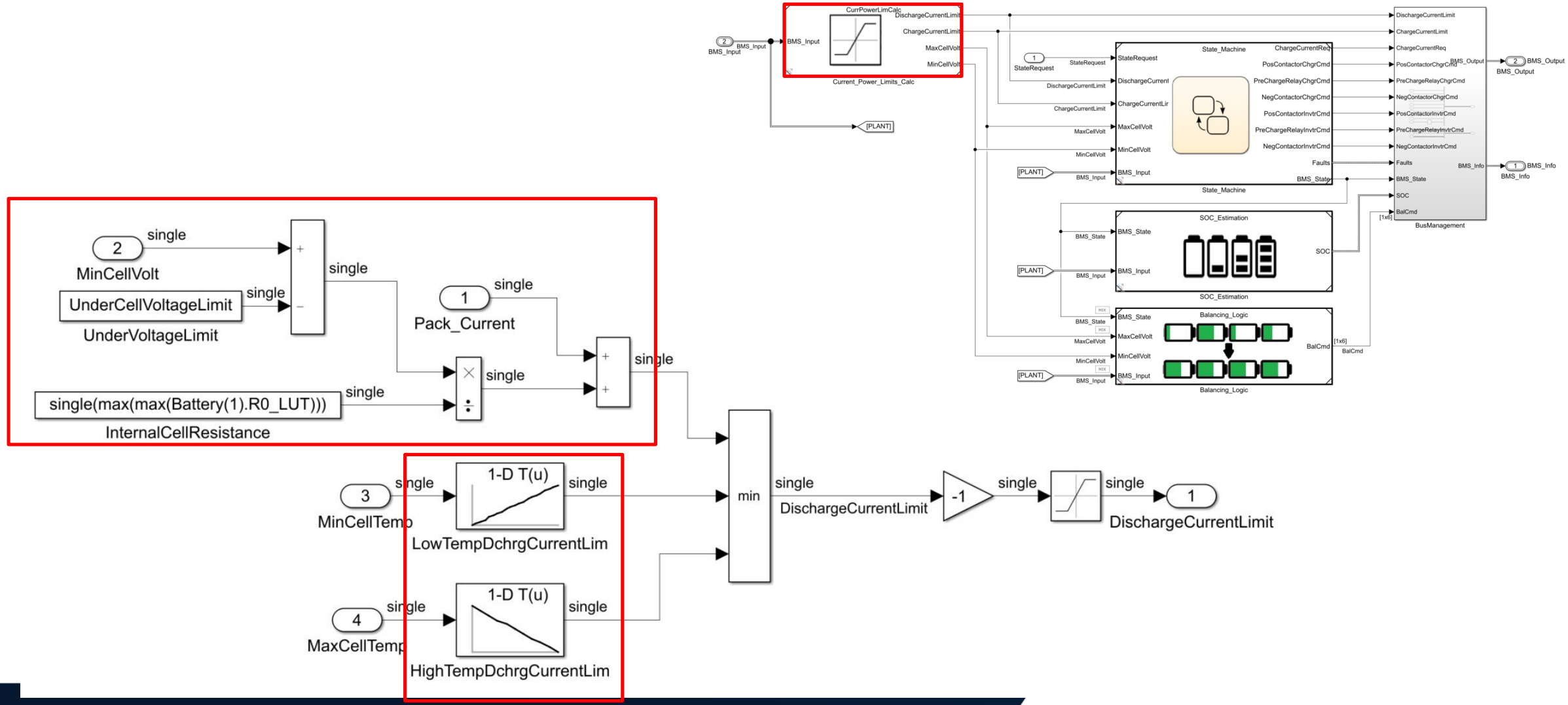
- What is BMS and what engineers worry about?
- Developing the architecture
- Developing battery models
- **Design,** Verify and Deploy **BMS algorithms**
- Hardware-in-Loop testing
- Summary - Q&A

Design BMS algorithms in Simulink



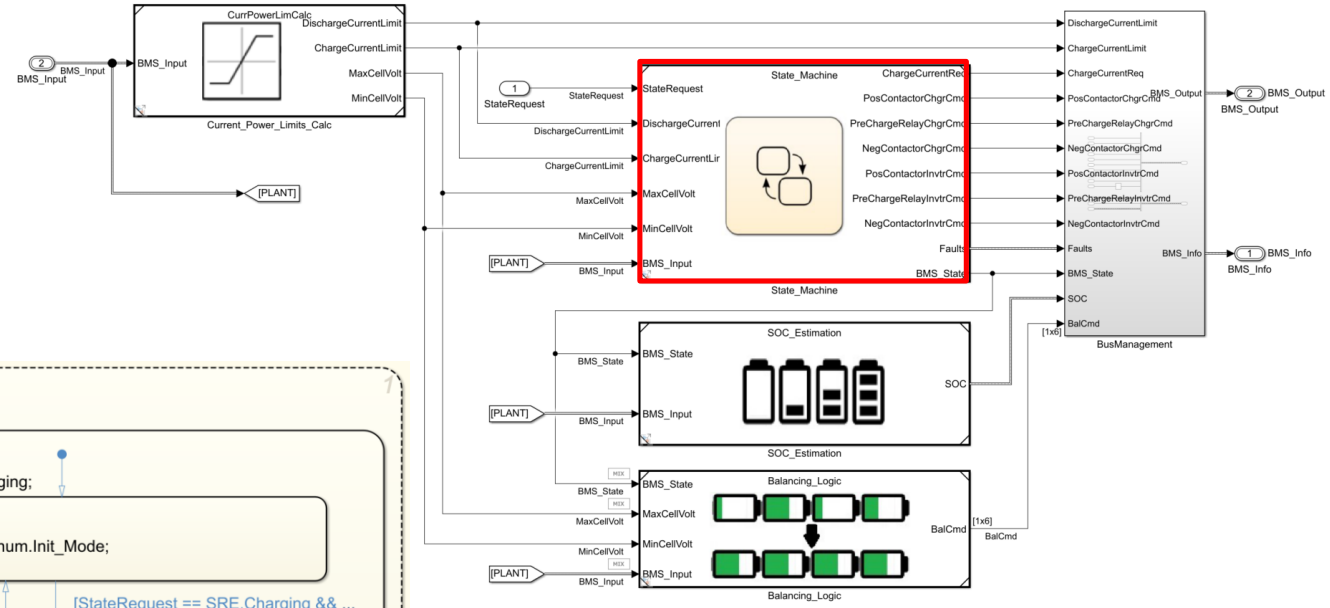
Design BMS algorithms in Simulink

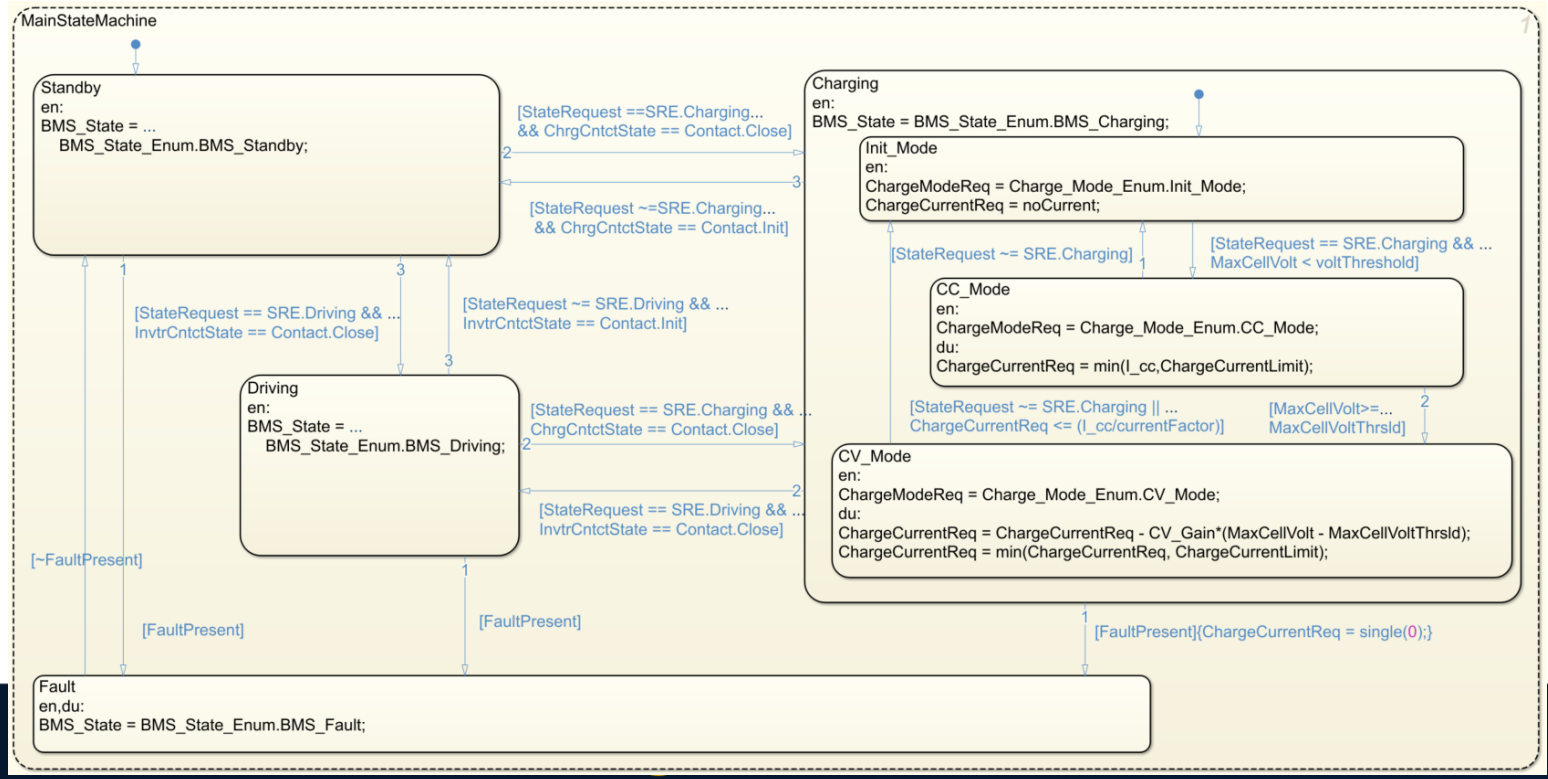
Current Limits



Design BMS algorithms in Simulink

State Machine

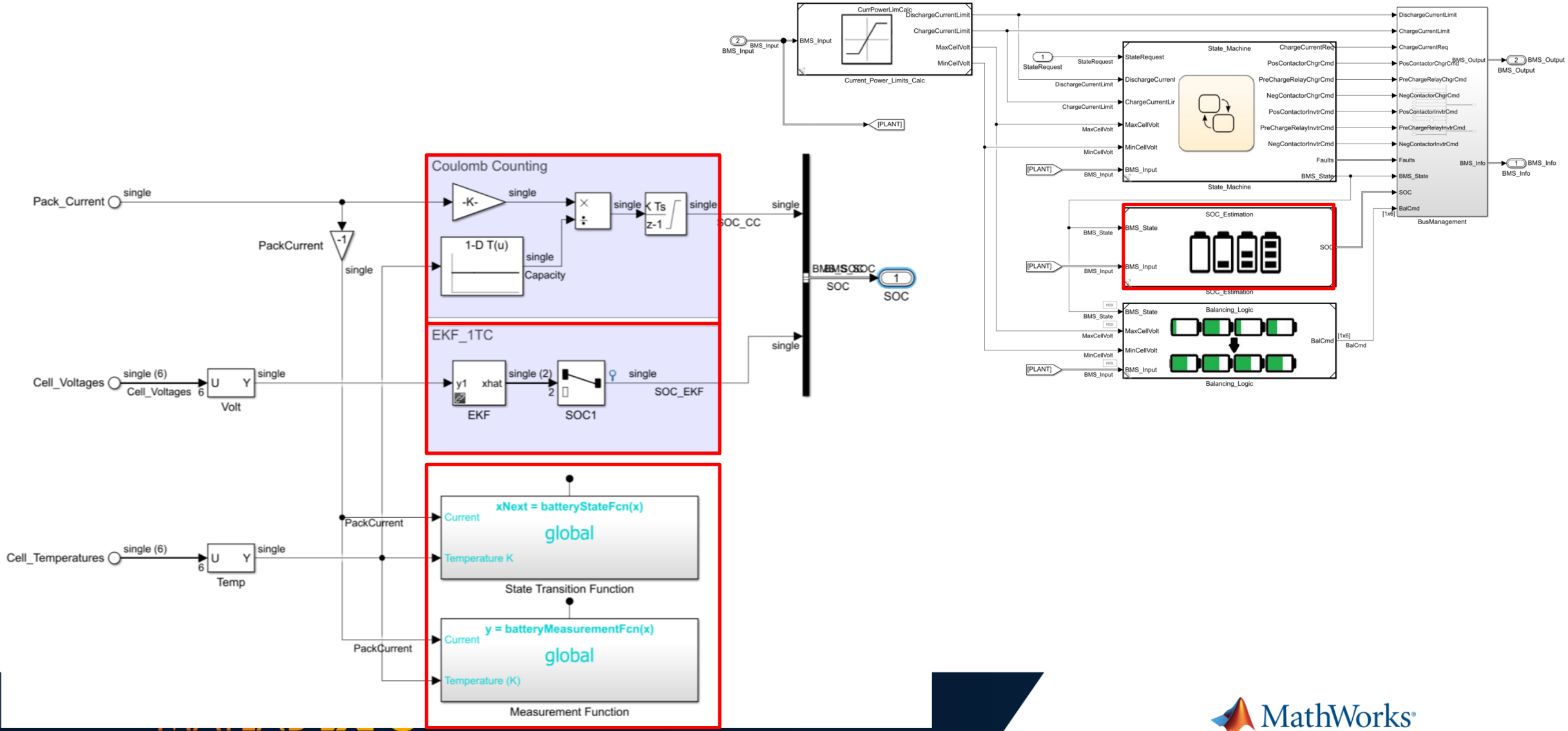




MathWorks®

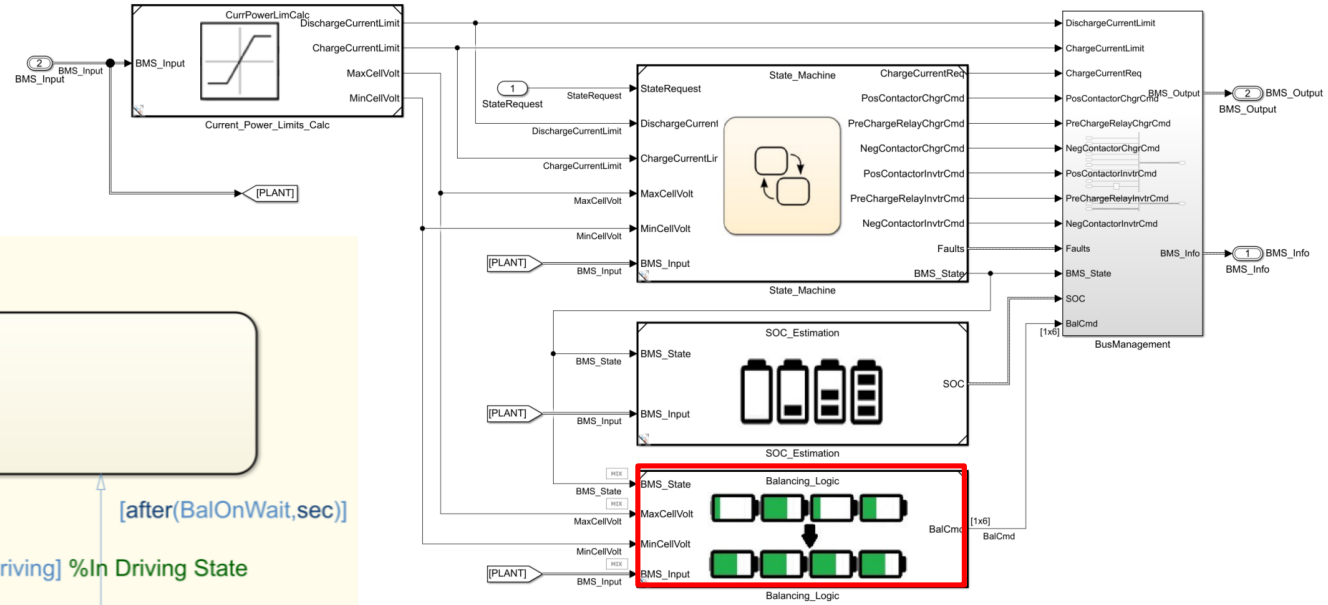
Design BMS algorithms in Simulink

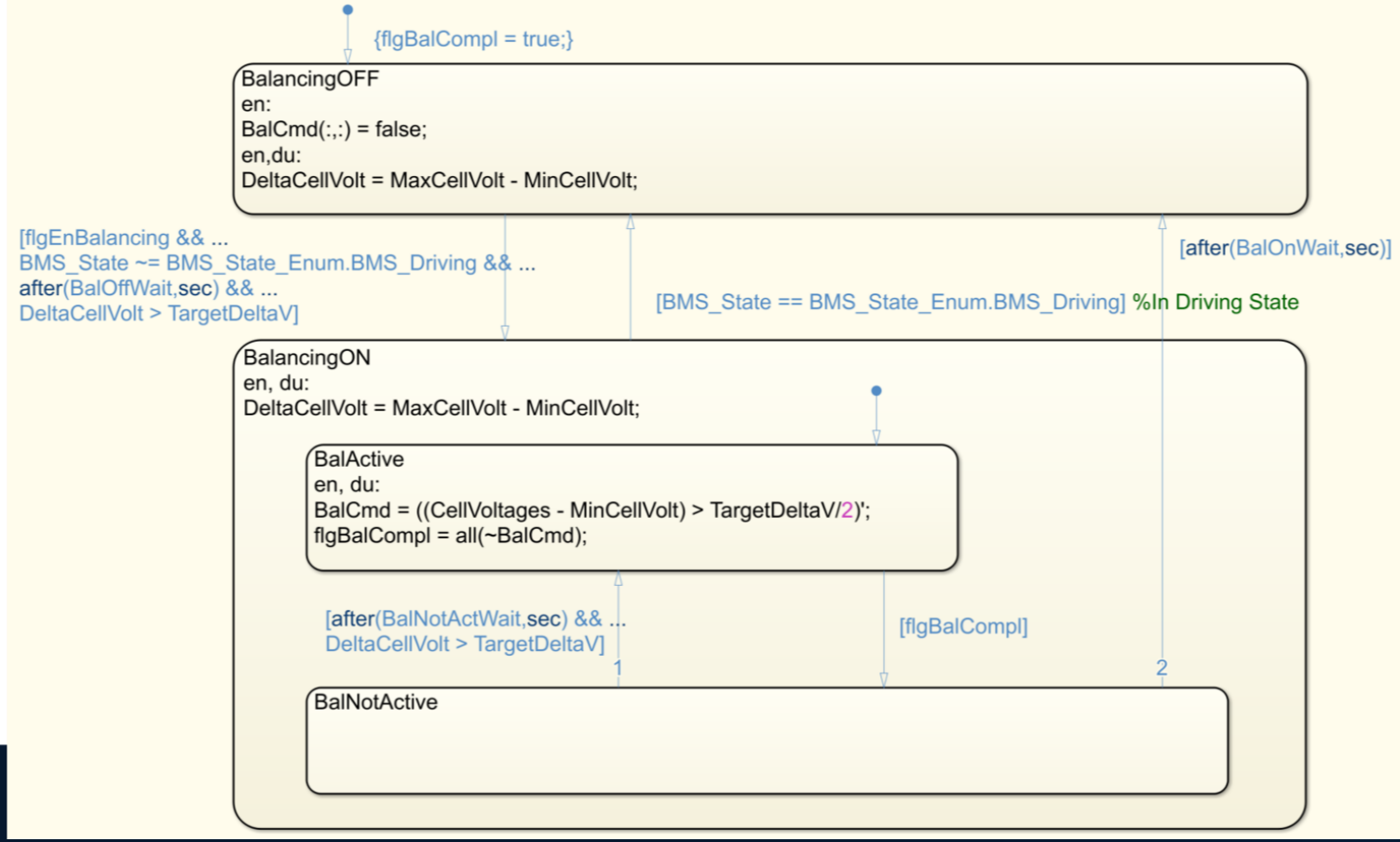
State of Charge



Design BMS algorithms in Simulink

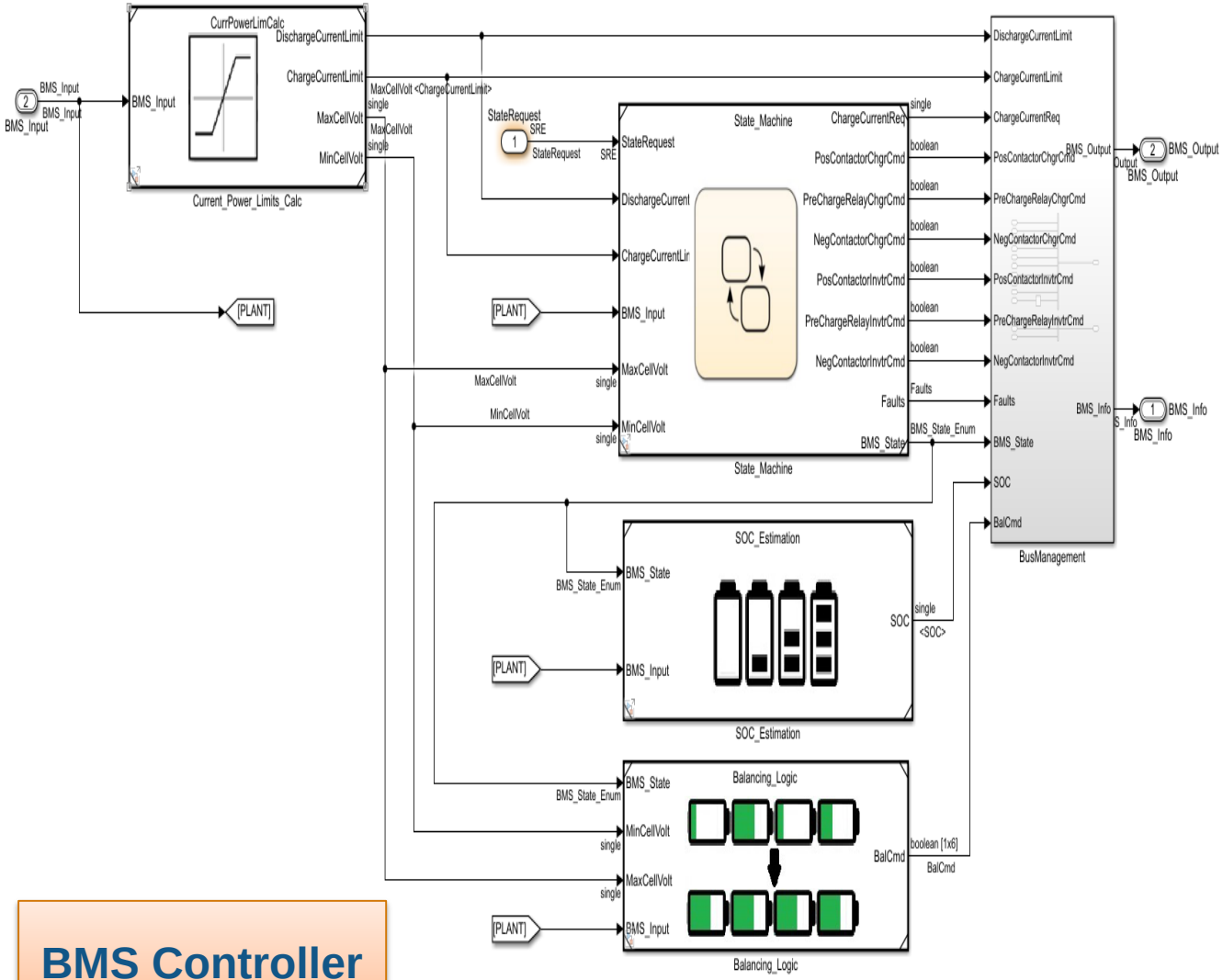
Cell Balancing





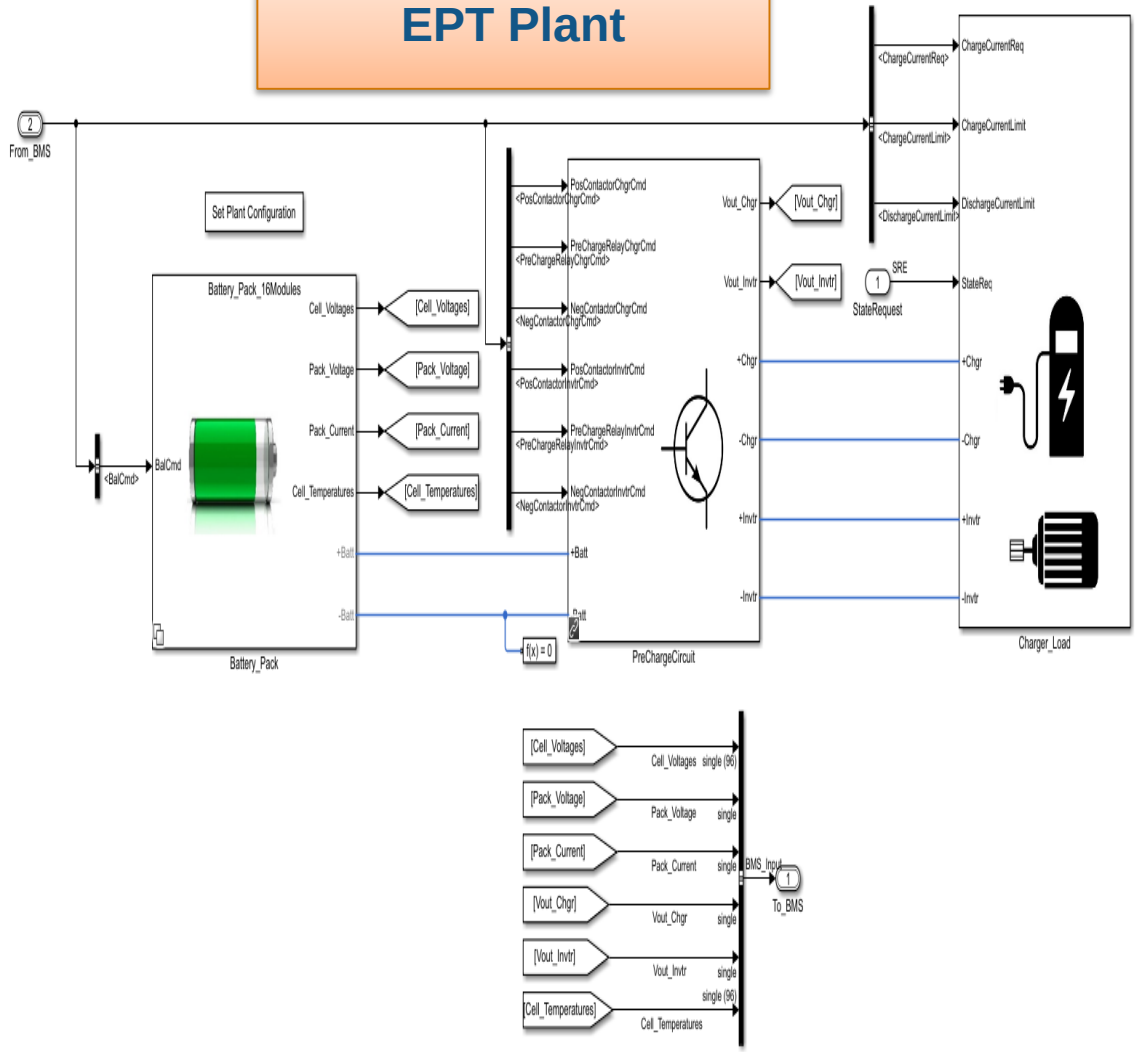
MathWorks

Simulation System



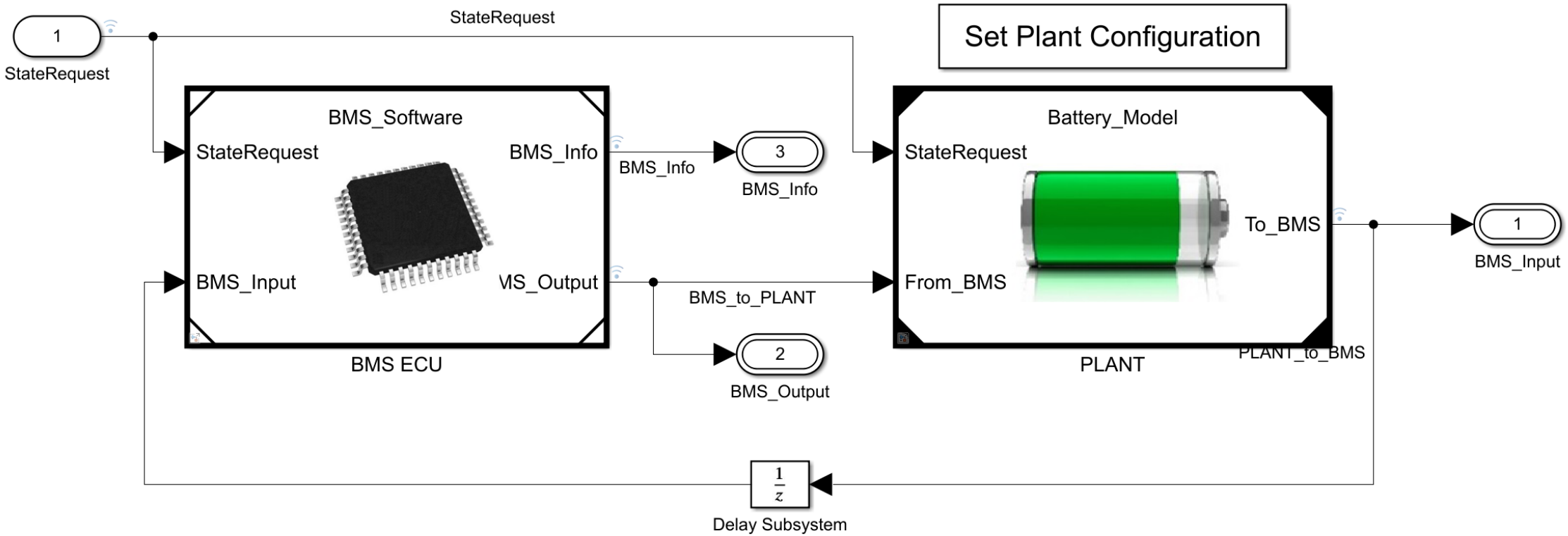
BMS Controller

EPT Plant



Design BMS algorithms in Simulink

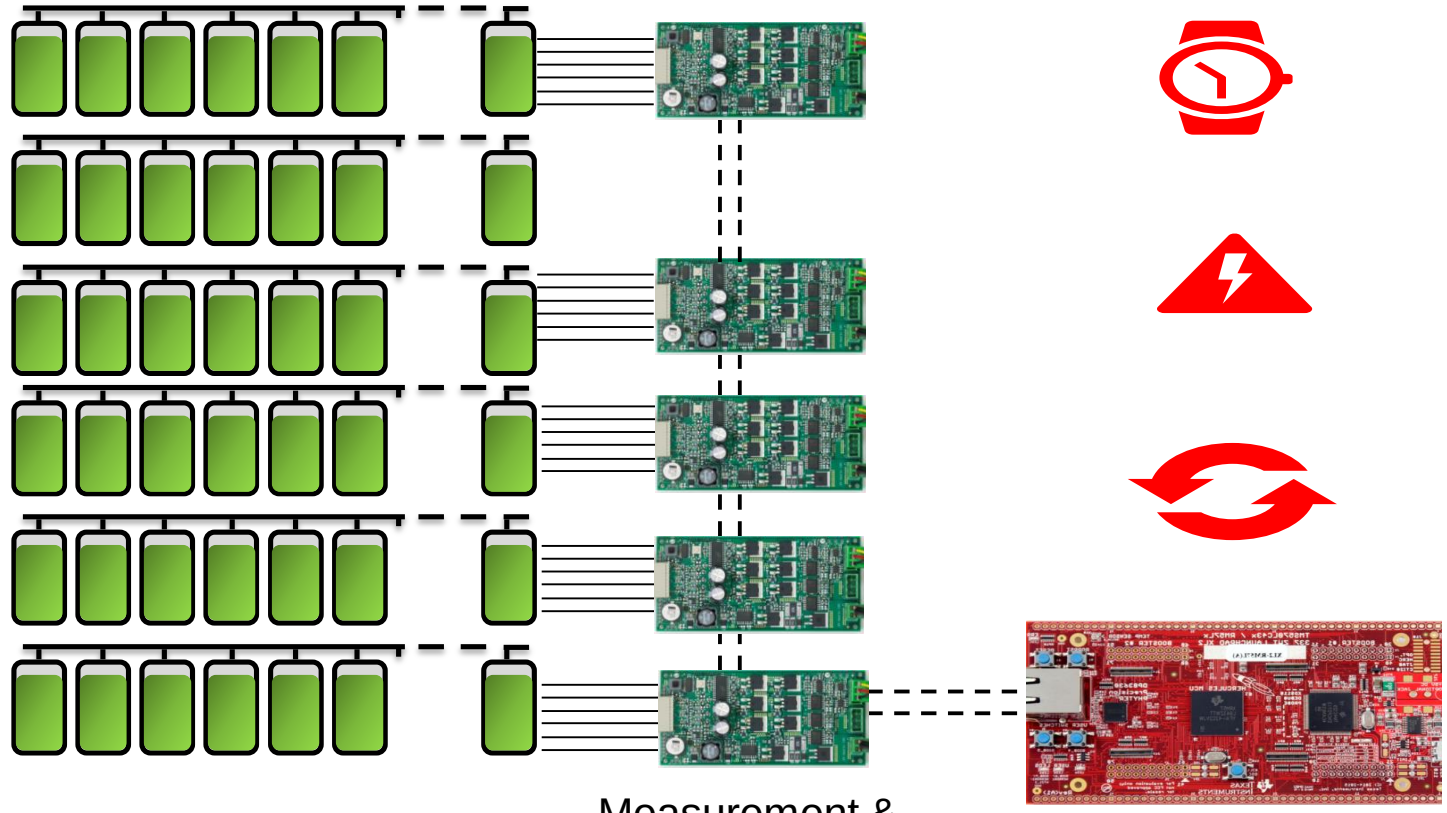
Battery Pack + Algorithm



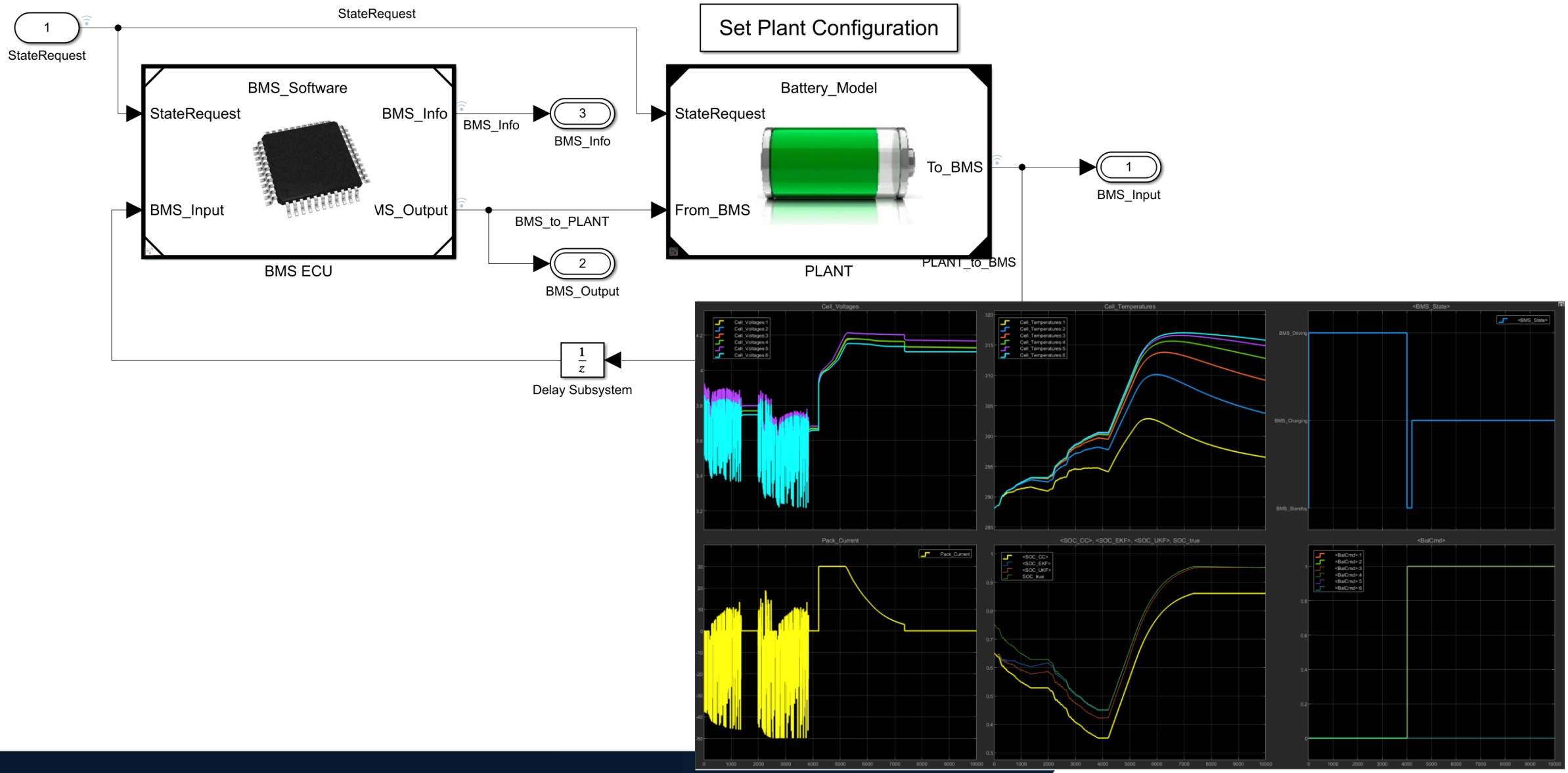
Real-Time Testing of Battery Management System

Testing BMS with Real Battery Cells/Pack

- Longer test cycles
- Difficult to reproduce results
- Difficult to test fault conditions
- Limited test automation

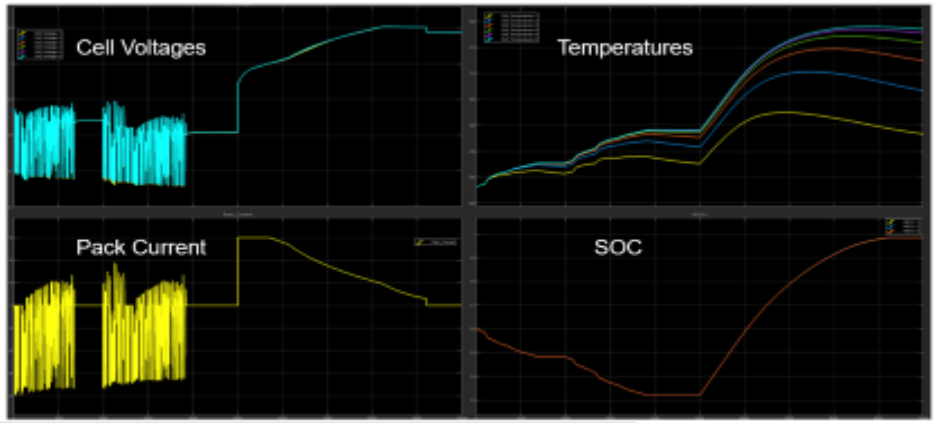


Evaluate System Behavior with CoSimulation

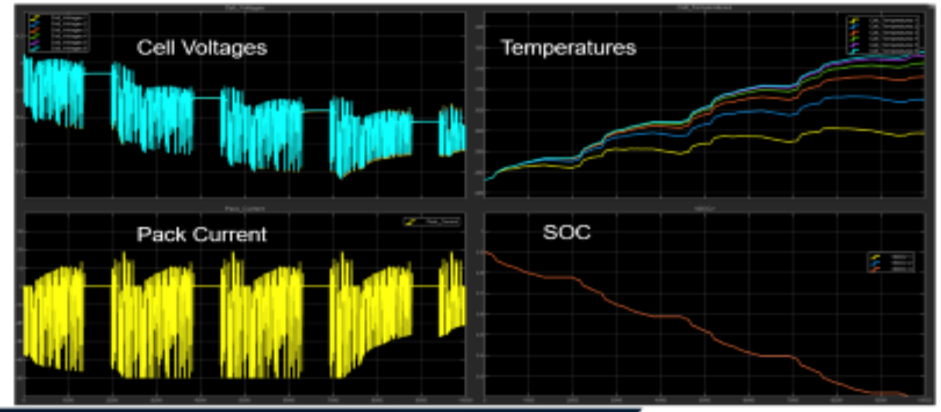


Performance Analysis

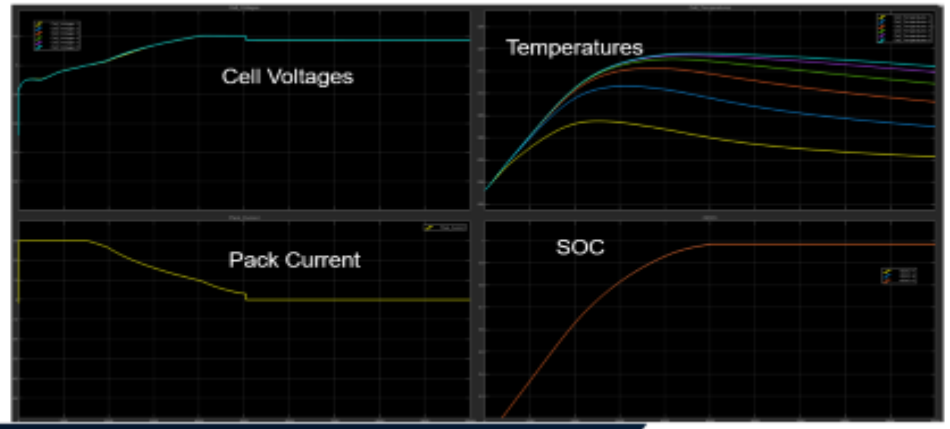
Discharge + Charge



Discharge Only



Charge Only

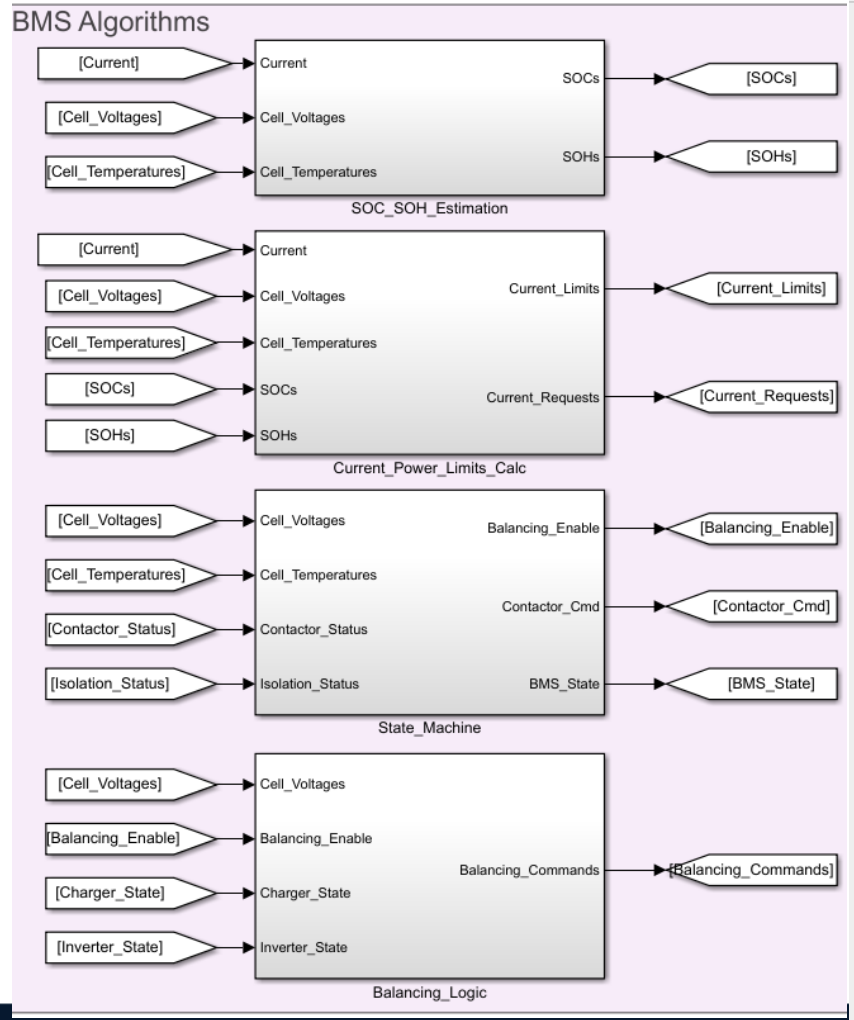


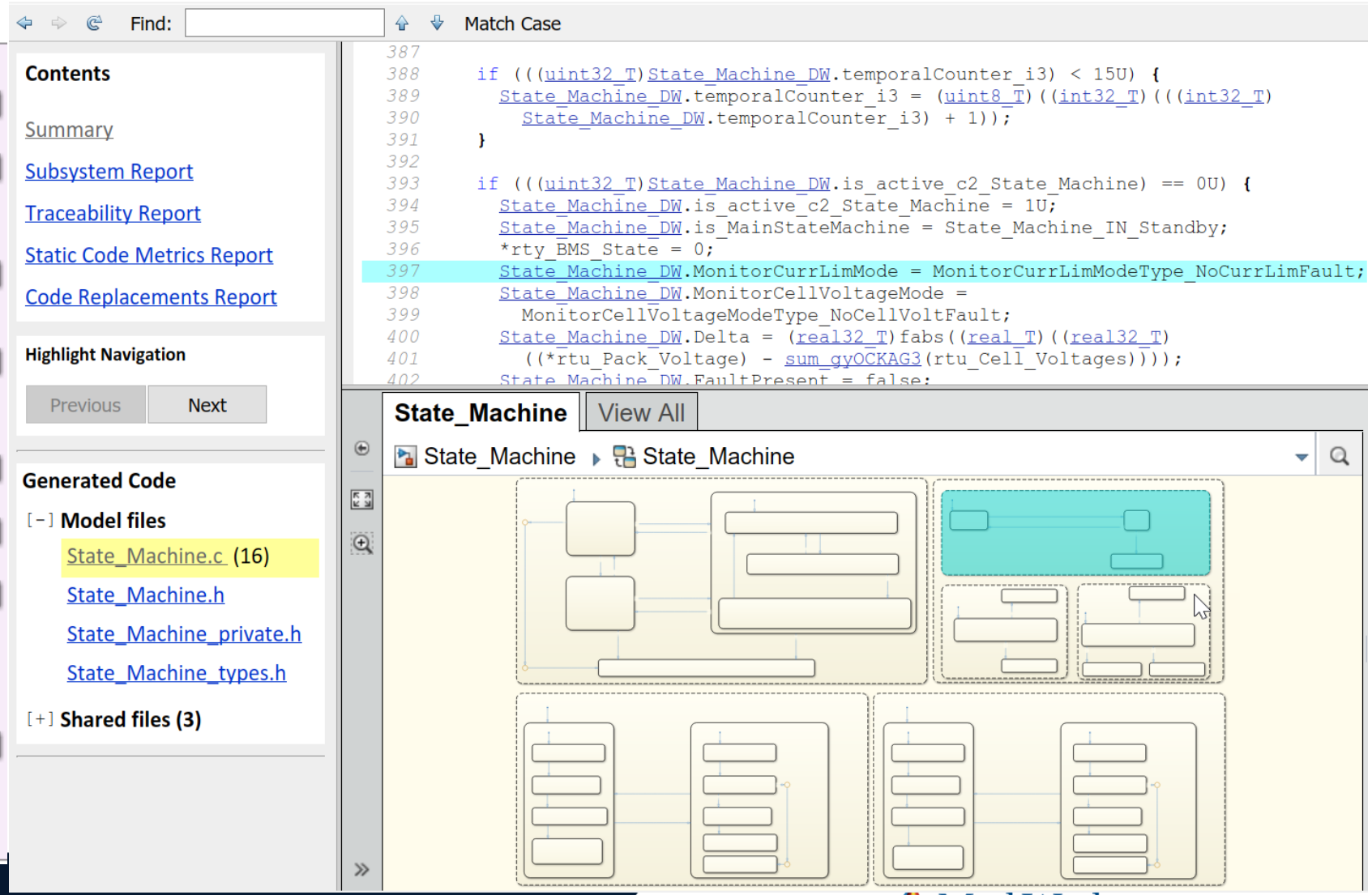
Evaluate SOC Estimation

Agenda

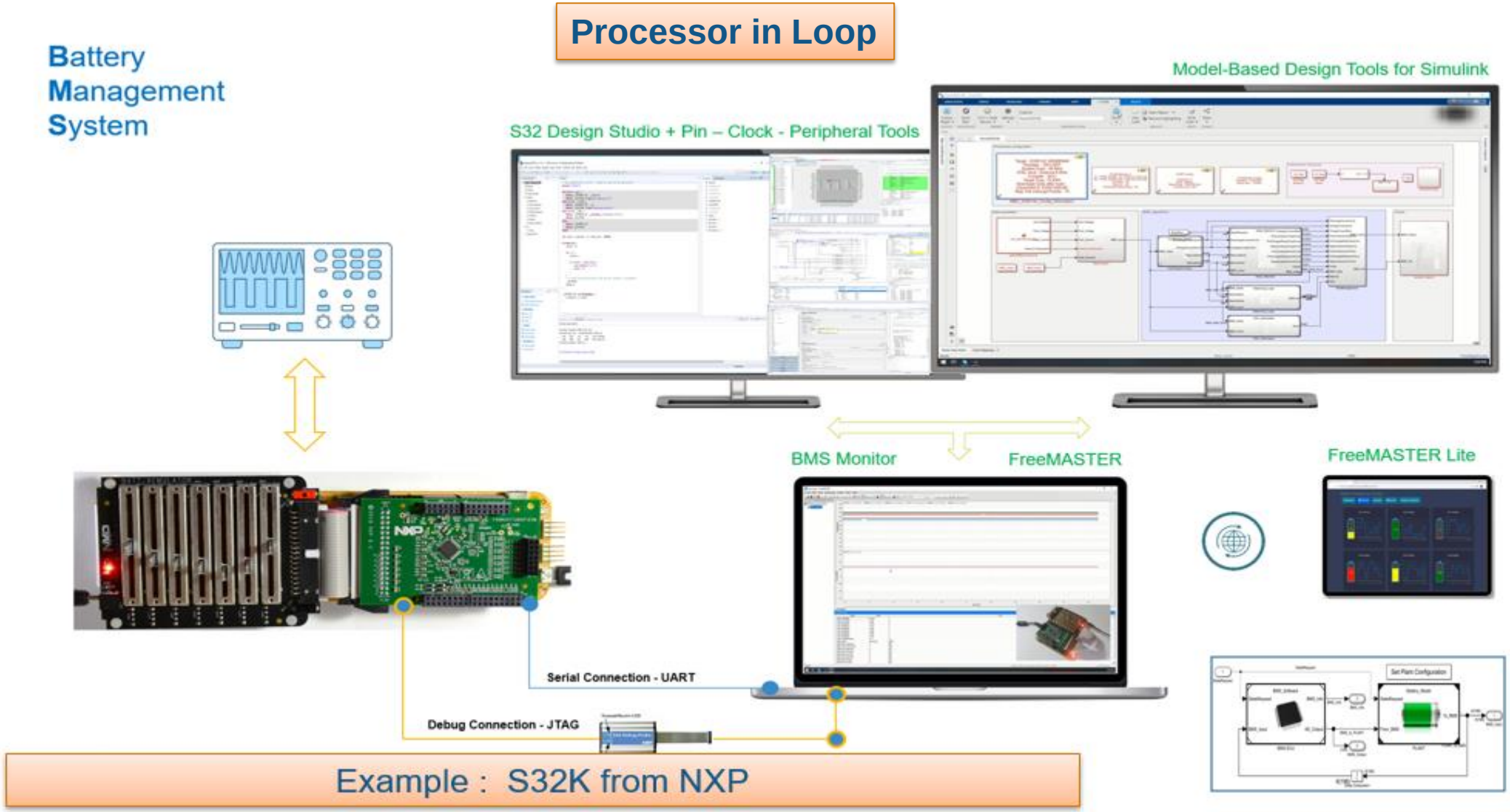
- What is BMS and what engineers worry about?
- Developing the architecture
- Developing battery models
- Design**,** Verify and **Deploy BMS algorithms**
- Hardware-in-Loop testing
- Summary - Q&A

Generate C/C++ Code From BMS Algorithm Models





Deploy algorithms on Target Controller



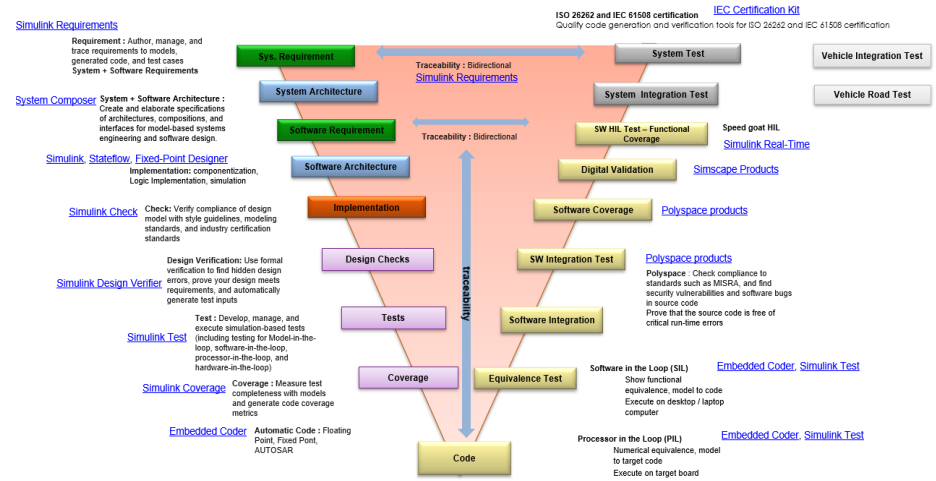
Did we generate code too early?
Is this ready to ship?

What if there are bugs?

Where are they? How do we find them?

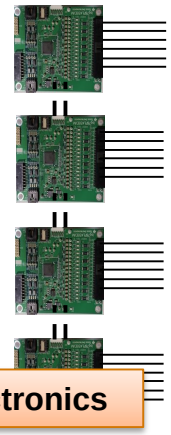
Agenda

- What is BMS and what engineers worry about?
- Developing the architecture
- Developing battery models
- Design, **Verify** and Deploy **BMS algorithms**
- Hardware-in-Loop testing
- Summary - Q&A




```
if (((uint32_T)State_Machine_DW.temporalCounter_i3) < 15U) {
  State_Machine_DW.temporalCounter_i3 = (uint8_T)((int32_T)(((ir
    State_Machine_DW.temporalCounter_i3) + 1));
}

if (((uint32_T)State_Machine_DW.is_active_c2_State_Machine) == (
  State_Machine_DW.is_active_c2_State_Machine = 1U;
  State_Machine_DW.is_MainStateMachine = State_Machine_IN_Standb
  *rty_BMS_State = 0;
  State_Machine_DW.MonitorCurrLimMode = MonitorCurrLimModeType_N
  State_Machine_DW.MonitorCellVoltageMode =
    MonitorCellVoltageModeType_NoCellVoltFault;
  State_Machine
    ((*rtu_Pack                    u_Cell_Vo
```



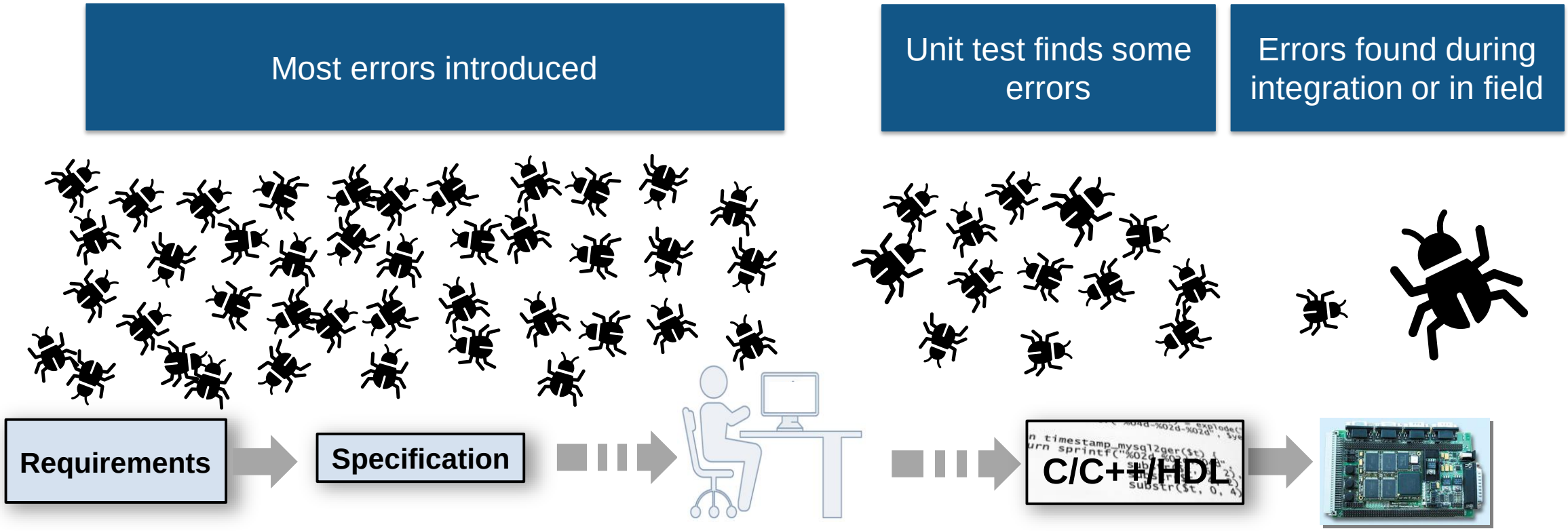
Software **Electronics** **Battery Pack**

Why Testing, Verification and Validation

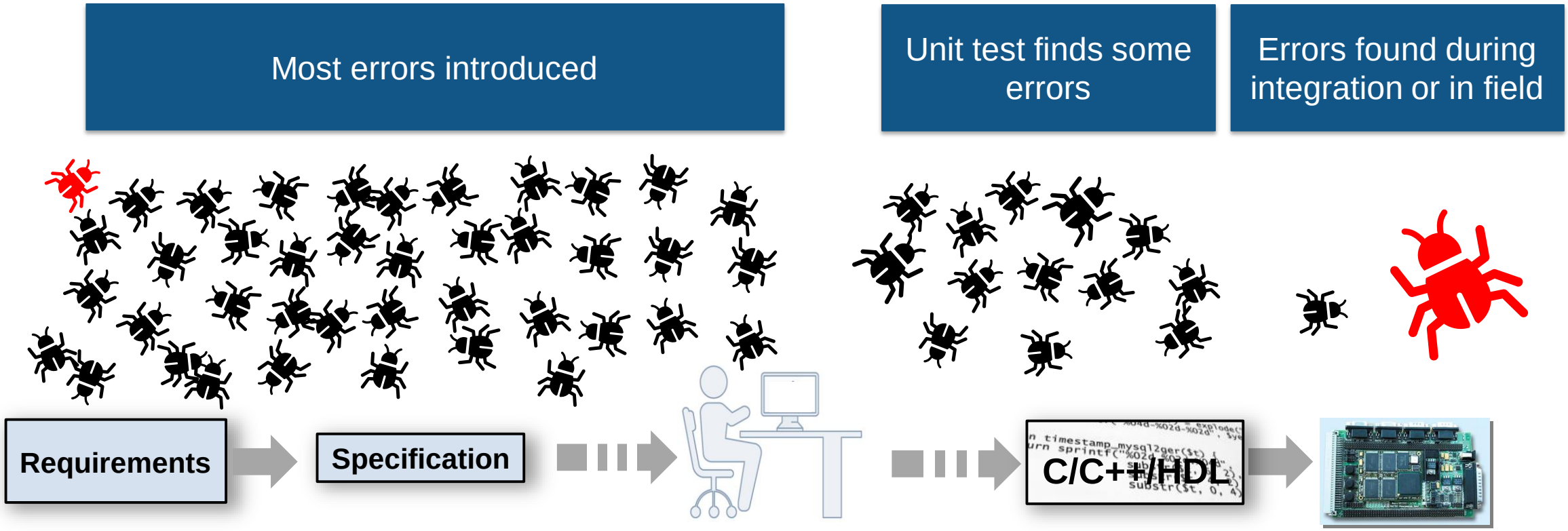
Safety Critical System

Functional Safety Certification

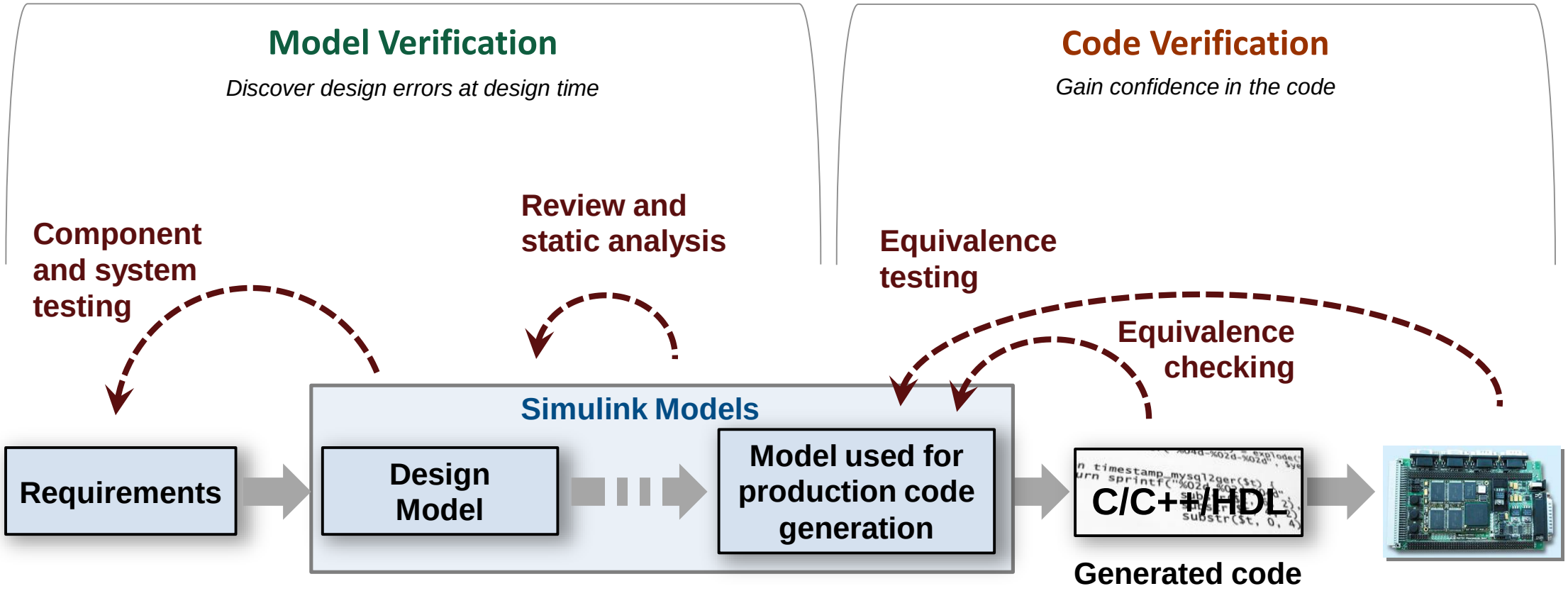
Typical Development Workflow



Challenge: Errors introduced early but found late



Model-Based Design Verification Workflow



- ✓ Faster development
- ✓ Reduced cost
- ✓ Less hassle
- ✓ **More Engineering**

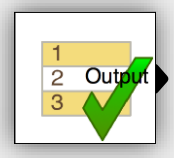
Systematic Functional Testing with Simulink Test

Test Case

Inputs

MAT file (input)

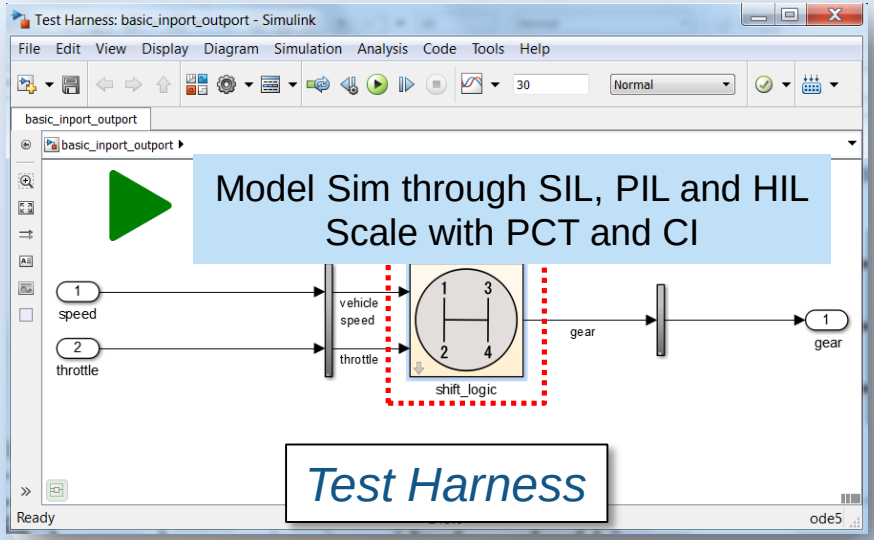
Signal Editor



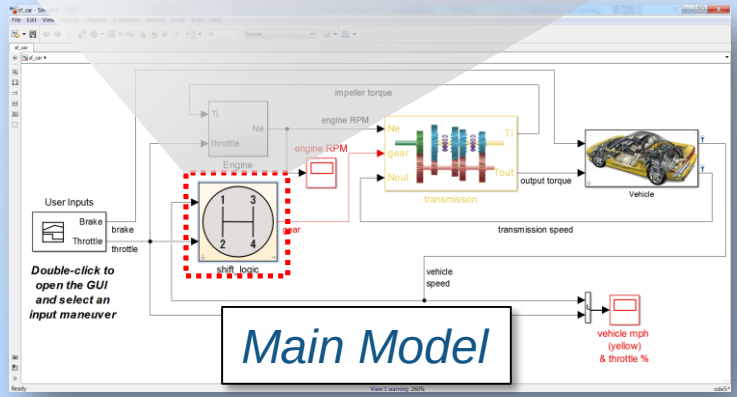
Test Sequence

Excel file (input)

and more!

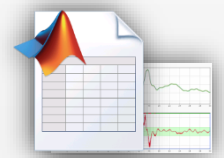


Test Harness

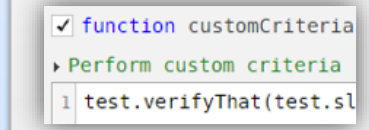


Main Model

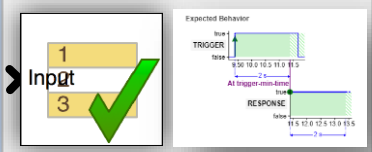
Assessments



MAT file (baseline)



MATLAB Unit Test



Assessments

Excel file (baseline)

and more!

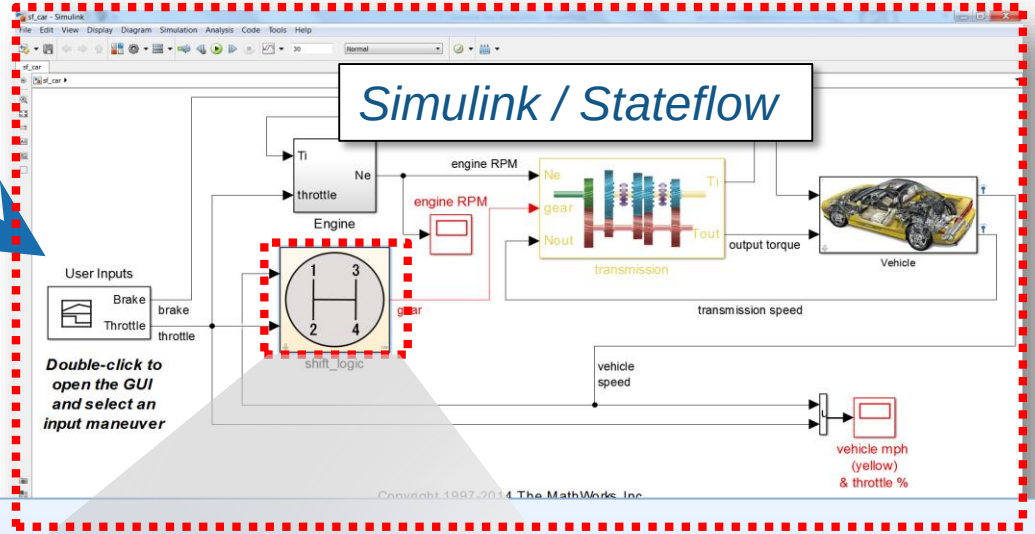
Requirements Verification with Simulink

Requirements

- crs_req_func_spec
  - 1 Driver Switch Request Handling
    - 1.1 Switch precedence
    - 1.2 Avoid repeating commands

Implemented By

Verified By

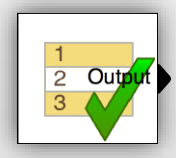


Test Case

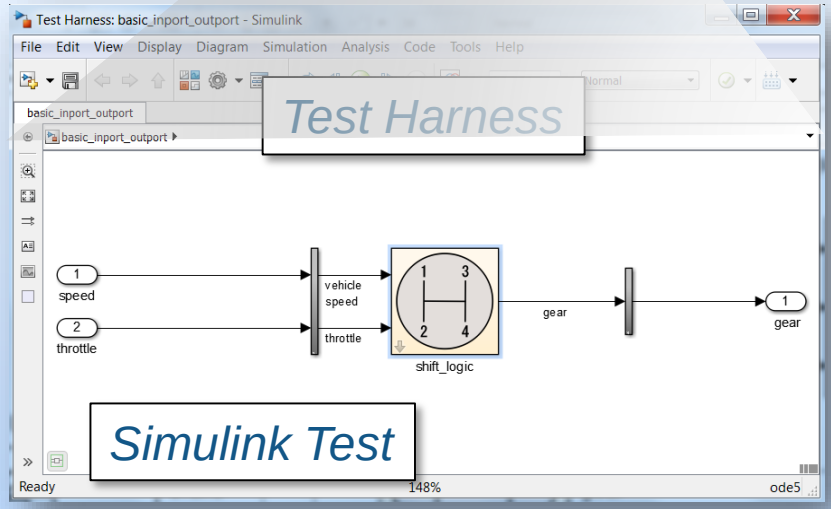
Inputs

MAT / Excel file (input)

Signal Editor

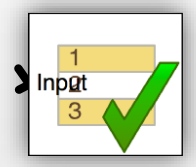


Test Sequence



Assessments

MAT / Excel File (baseline)

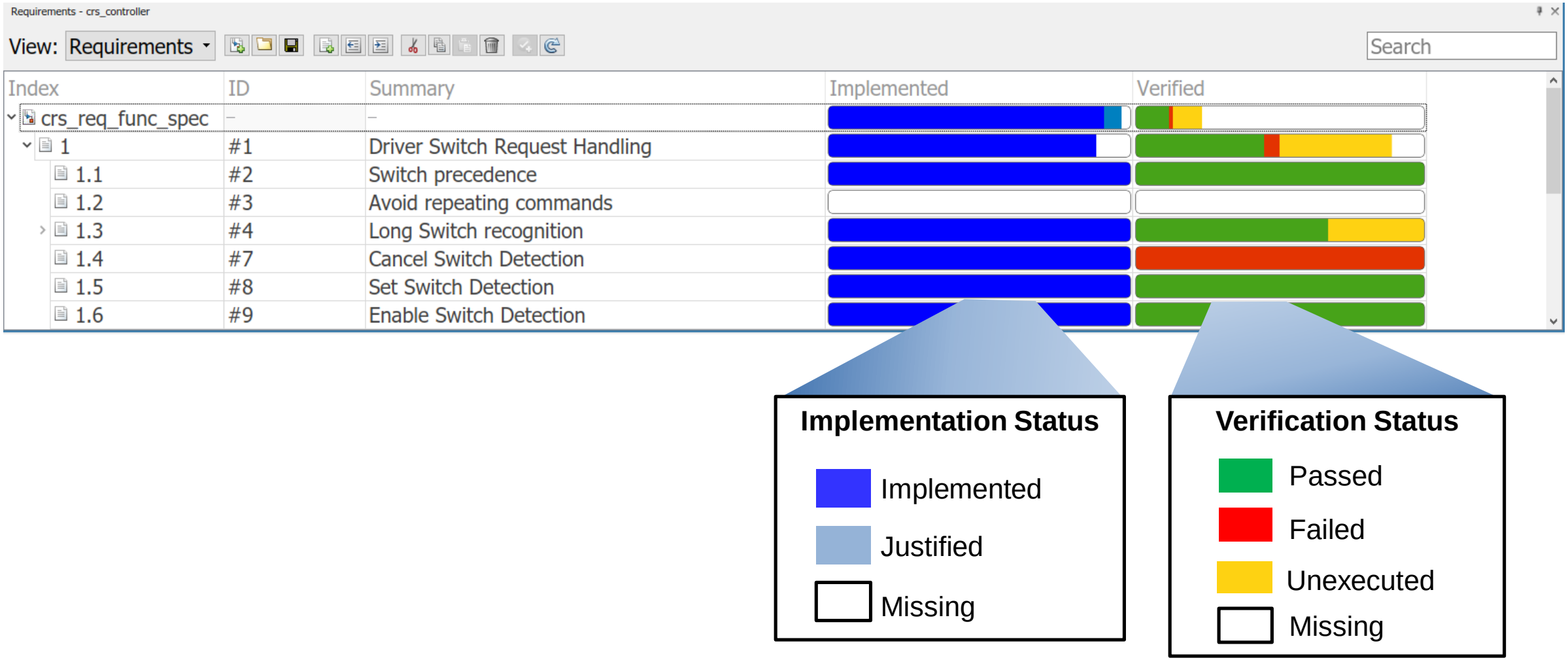


Test Assessments

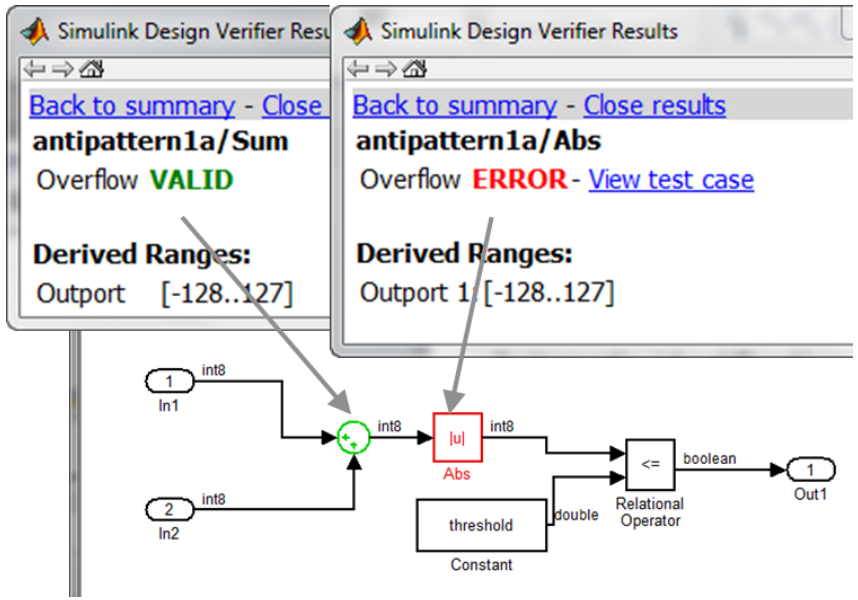
```
function customCriteria
Perform custom criteria
1 test.verifyThat(test.sl
```

MATLAB Unit Test

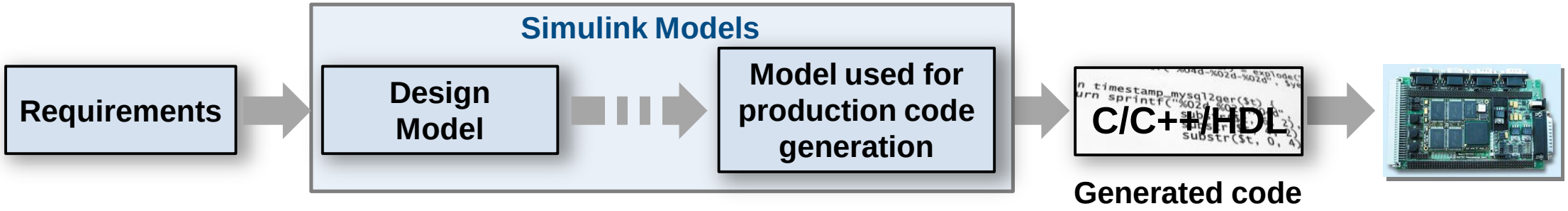
Track Implementation and Verification



Detect Design Errors with Formal Methods

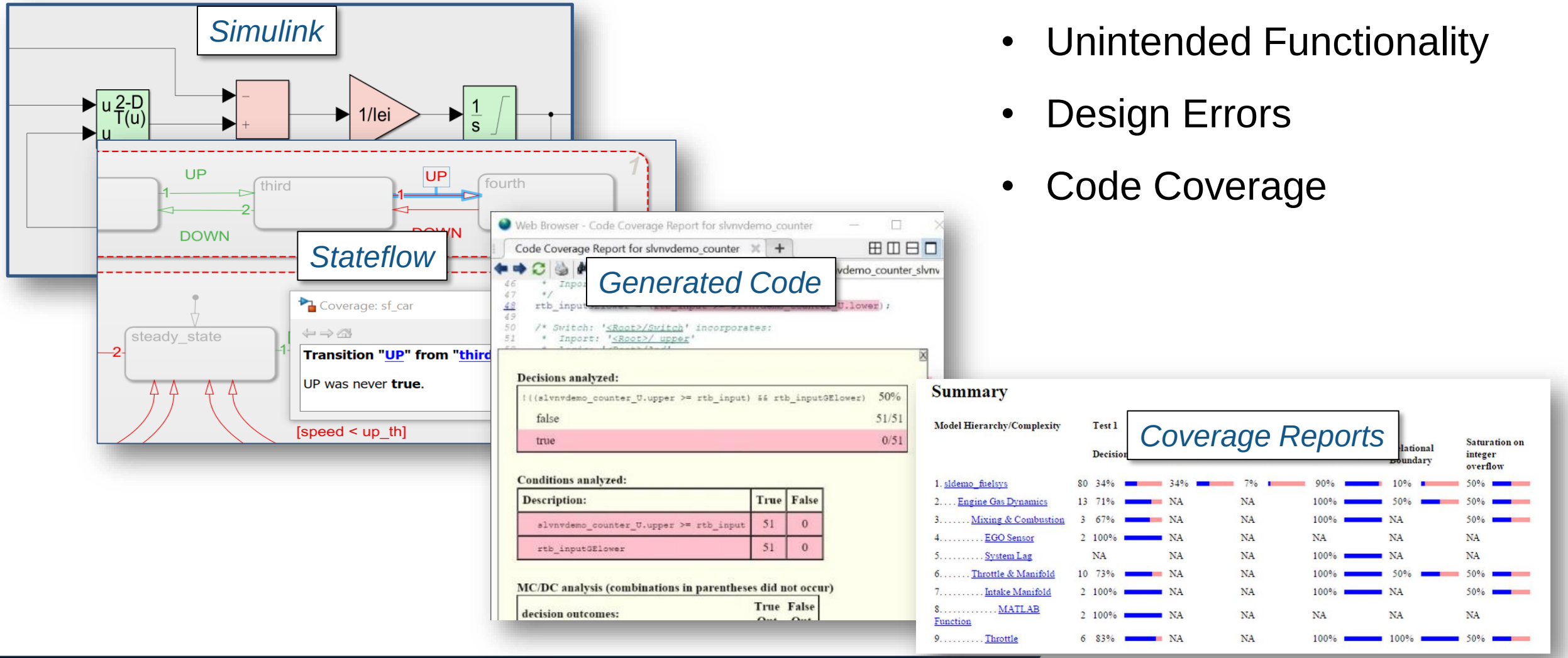


- Find run-time design errors:
  - Integer overflow
  - Dead Logic
  - Division by zero
  - Array out-of-bounds
  - Range violations
- Generate counter example to reproduce error



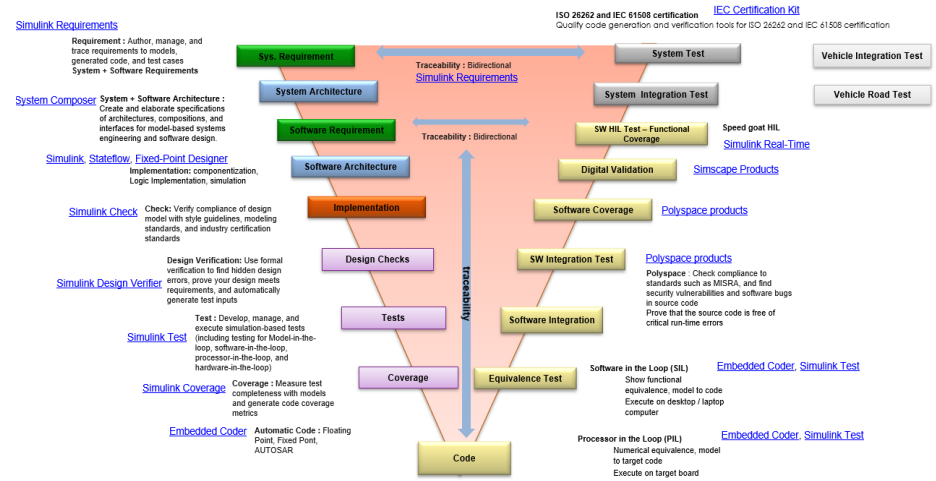
Coverage Analysis to Measure Testing

- Identify testing gaps
- Missing requirements
- Unintended Functionality
- Design Errors
- Code Coverage



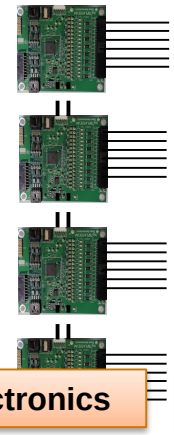
Agenda

- What is BMS and what engineers worry about?
- Developing the architecture
- Developing battery models
- Design, Verify and Deploy BMS algorithms
- **Hardware-in-Loop testing**
- Summary - Q&A




```
if (((uint32_T)State_Machine_DW.temporalCounter_i3) < 15U) {
  State_Machine_DW.temporalCounter_i3 = (uint8_T)((int32_T)(((ir
    State_Machine_DW.temporalCounter_i3) + 1));
}

if (((uint32_T)State_Machine_DW.is_active_c2_State_Machine) == (
  State_Machine_DW.is_active_c2_State_Machine = 1U;
  State_Machine_DW.is_MainStateMachine = State_Machine_IN_Standb
  *rty_BMS_State = 0;
  State_Machine_DW.MonitorCurrLimMode = MonitorCurrLimModeType_N
  State_Machine_DW.MonitorCellVoltageMode =
    MonitorCellVoltageModeType_NoCellVoltFault;
  State_Machine
    ((*rtu_Pack                    u_Cell_Vo
```

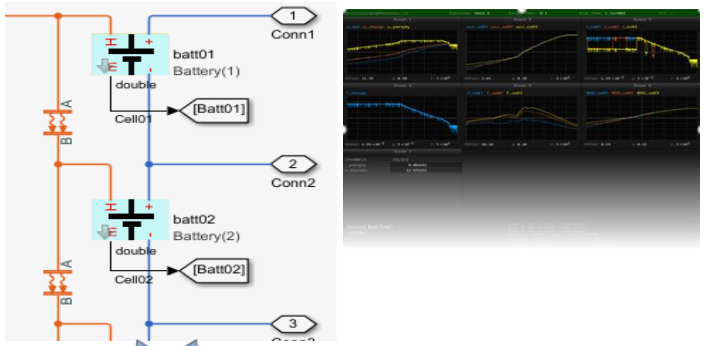


Software **Electronics** **Battery Pack**

Hardware-In-Loop Testing of Battery Management System

Testing BMS with Emulated Battery Cells

- Reduce testing time
- Test fault conditions safely
- Automate testing



Automatic Code Generation

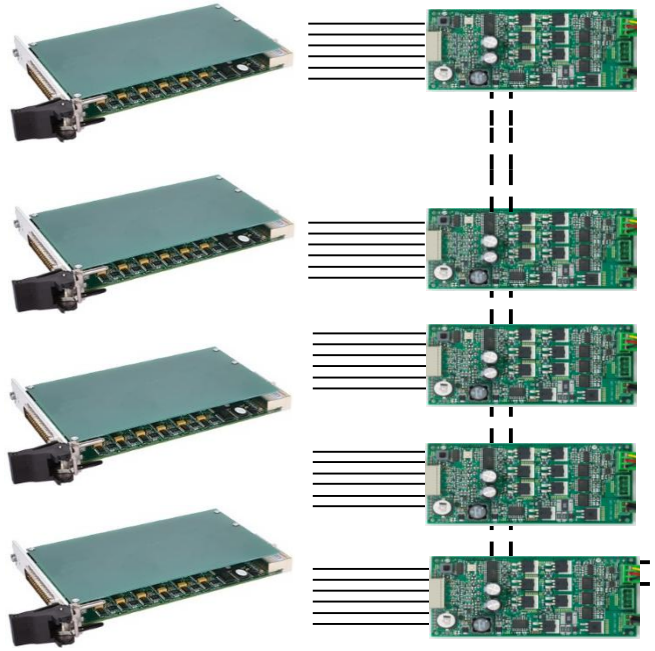
Battery Emulation

1 0 0 1 0 1 1 1 0 0 1 0 1

Wiring and Signal Conditioning

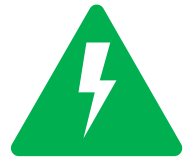
Sensor and Fault Emulation

- Produce Isolated Voltages
- Sink and Source Current
- Support Series and Parallel Configuration
- Temperature simulation



Measurement & Diagnostics

Main Controller



An Integrated Solution for Embedded Development with Simulink

Simulink Requirements

Requirement : Author, manage, and trace requirements to models, generated code, and test cases
System + Software Requirements

System Composer **System + Software Architecture :** Create and elaborate specifications of architectures, compositions, and interfaces for model-based systems engineering and software design.

Simulink, Stateflow, Fixed-Point Designer
Implementation: componentization, Logic Implementation, simulation

Simulink Check **Check:** Verify compliance of design model with style guidelines, modeling standards, and industry certification standards

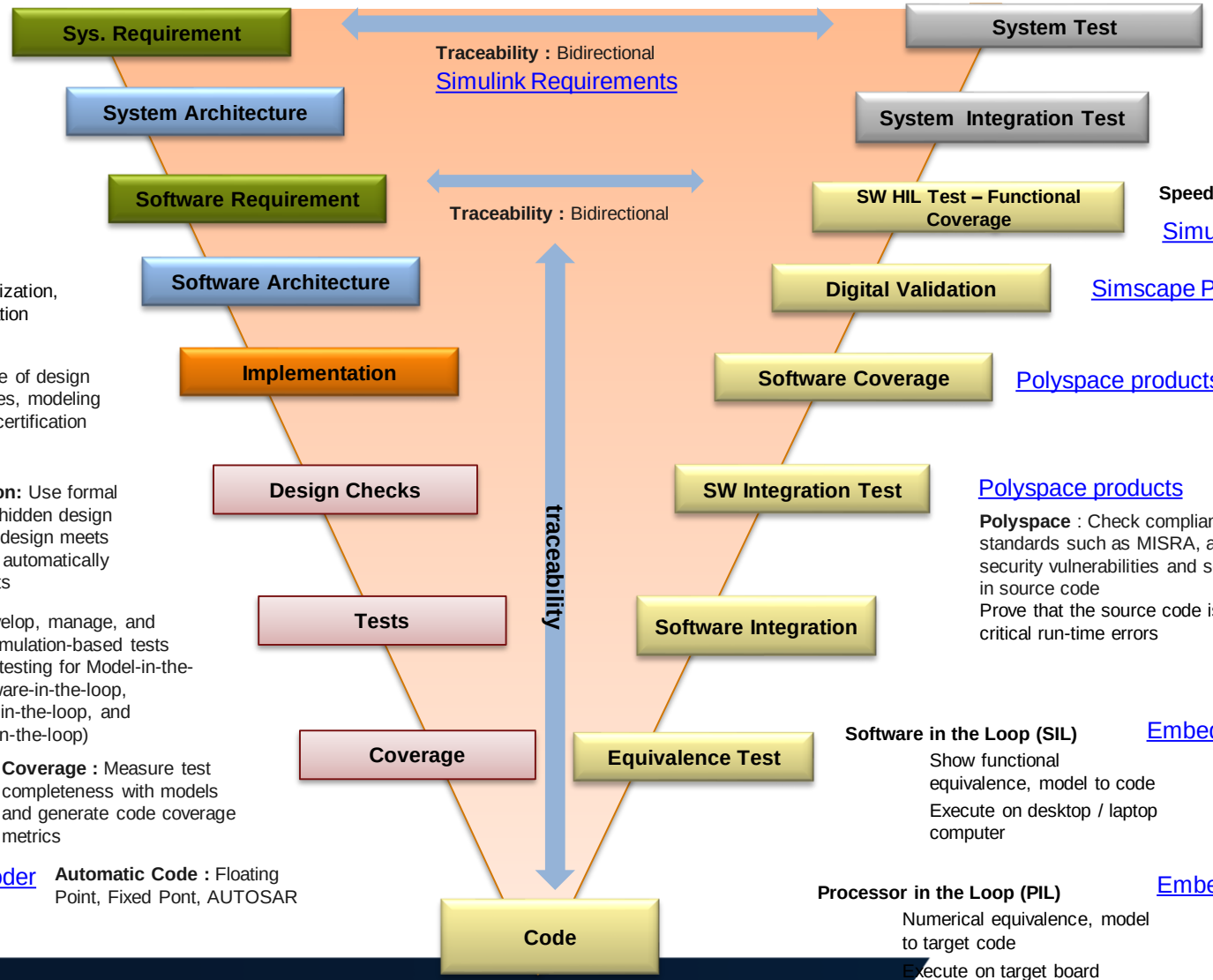
Simulink Design Verifier **Design Verification:** Use formal verification to find hidden design errors, prove your design meets requirements, and automatically generate test inputs

Simulink Test **Test :** Develop, manage, and execute simulation-based tests (including testing for Model-in-the-loop, software-in-the-loop, processor-in-the-loop, and hardware-in-the-loop)

Simulink Coverage **Coverage :** Measure test completeness with models and generate code coverage metrics

Embedded Coder **Automatic Code :** Floating Point, Fixed Pont, AUTOSAR

ISO 26262 and IEC 61508 certification IEC Certification Kit
Qualify code generation and verification tools for ISO 26262 and IEC 61508 certification



Vehicle Integration Test

Vehicle Road Test

Speed goat HIL
Simulink Real-Time

Simscape Products

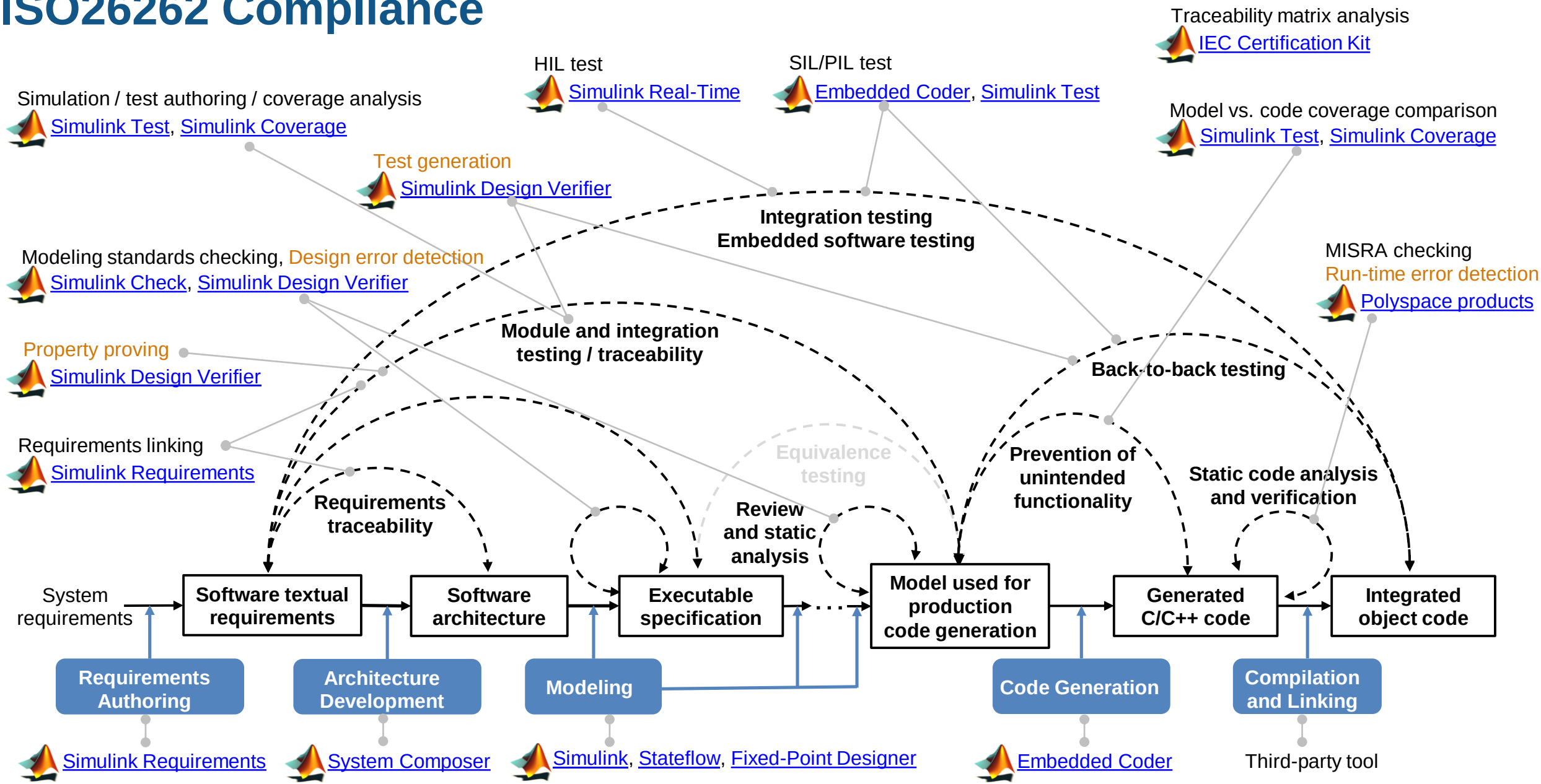
Polyspace products

Polyspace products
Polyspace : Check compliance to standards such as MISRA, and find security vulnerabilities and software bugs in source code
Prove that the source code is free of critical run-time errors

Software in the Loop (SIL) Embedded Coder, Simulink Test
Show functional equivalence, model to code
Execute on desktop / laptop computer

Processor in the Loop (PIL) Embedded Coder, Simulink Test
Numerical equivalence, model to target code
Execute on target board

ISO26262 Compliance



ISO 26262:2018 Structure

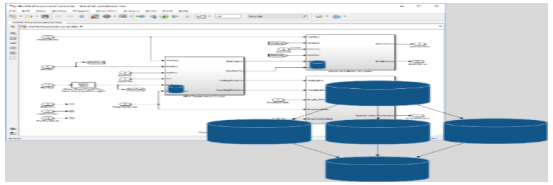
ISO 26262-1	• Vocabulary
ISO 26262-2	• Management of functional safety
ISO 26262-3	• Concept phase
ISO 26262-4	• Product development: system level
ISO 26262-5	• Product development: hardware level
ISO 26262-6	• Product development: software level
ISO 26262-7	• Production, operation, service and decommissioning
ISO 26262-8	• Supporting processes
ISO 26262-9	• ASIL-oriented and safety-oriented analyses
ISO 26262-10	• Guidelines on ISO 26262
ISO 26262-11	• Guidelines on application of ISO 26262 to semiconductors
ISO 26262-12	• Adaptation of ISO 26262 for motorcycles

- Model-Based Systems Engineering
- Design of hardware system
- Model-Based Design
  - Development
  - Verification & Validation
  - Code generation
- Tool classification and qualification
- HDL Code generation

For BMS Applications

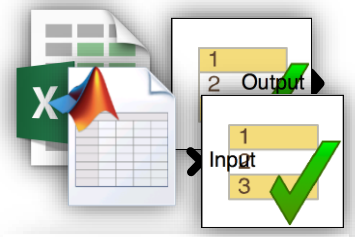
Business Logics and Architectures

- Algorithm Development
- Data management
- Architecture Models
- Share model as executable



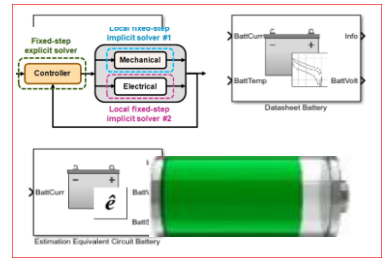
Verification and Validation

- Field Recorded Data Import
- Excel based test cases
- Test Scenarios and Assessment
- Signal builders
- Supported Various Reporting formats
- Model and Code Coverage



Physical Modelling and Co-Simulation

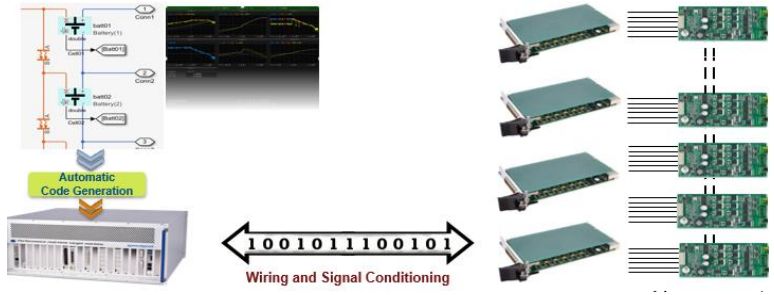
- Battery Pack Modelling
  - Cell Behaviors
- Electric Powertrain
  - Plant modelling
  - Co-Simulations
- Parameter estimations



Automatically generated Production Code

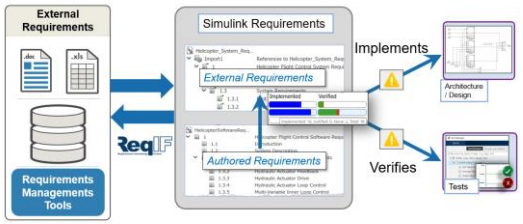
- Generate C/C++ Code
- Easy Integration with Legacy Code
- DLL generation capability
- AUTOSAR Specific Code

HIL and Integration Testing

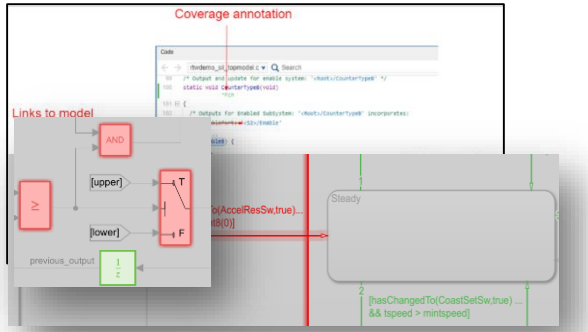


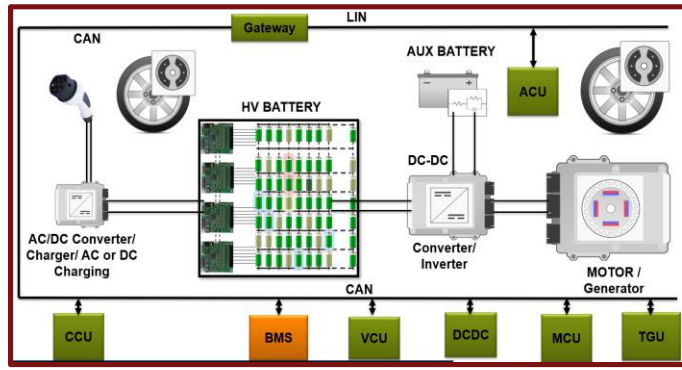
Traceability

- Bidirectional Traceability
- **Reqs. – Architecture – Design – Code – Test Cases – Reports**



Formal Verification &Test case Generation

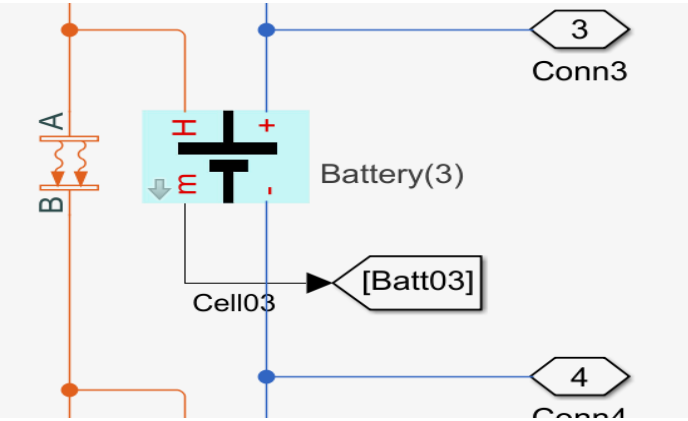


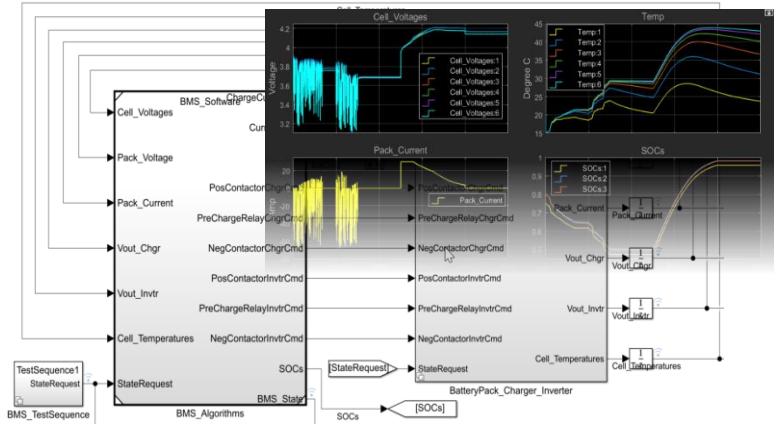


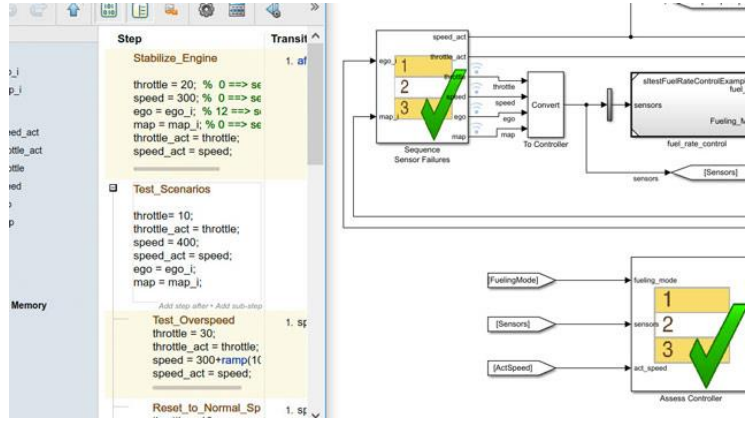
MATLAB EXPO

MathWorks

Conclusion







Leverage models to communicate technical specifications, design implementation, results and maintain traceability

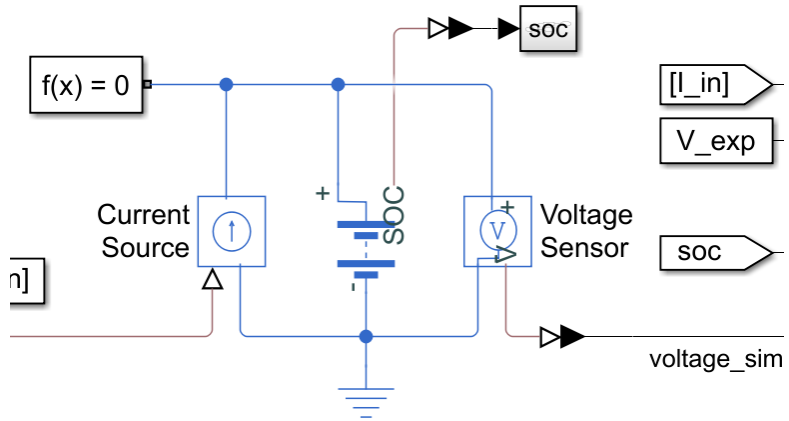
Test your design iterations every step of the way through simulations and Hardware-In-Loop testing

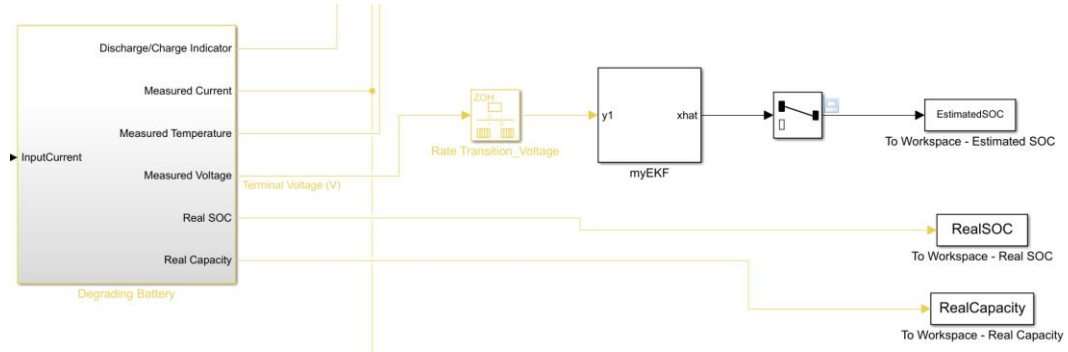
Gain confidence in design and work towards safety certification

Learn More about Battery Management System

A two-day course describes modeling Battery pack for designing and testing Battery Management System in Simulink® using Simscape, Stateflow, and Control System Toolbox. Topics include:

- Creating Physical Models using Simscape
- Cell model and its characterization
- Battery Pack modeling
- SoC Estimation using EKF
- Logic-Driven System Modeling using Stateflow
- Fault-Detection/Cell Balancing using Stateflow
- Harness creation and testing of Battery Management Systems using Simulink Test





Training Services

Exploit the full potential of MathWorks products

Flexible delivery options:

- Public training available in several cities
- Onsite training with standard or customized courses
- Web-based training with live, interactive instructor-led courses

More than 48 course offerings:

- Introductory and intermediate training on MATLAB, Simulink, Stateflow, code generation, and Polyspace products
- Specialized courses in control design, signal processing, parallel computing, code generation, communications, financial analysis, and other areas

MathWorks Consulting | Battery Simulation and Controls

MathWorks Consulting Services leverages industry background and technical expertise gained from hundreds of customer engagements to solve your battery simulation and controls challenges and to bring you the best battery performance.

- Areas we can help you:
  - Develop or assess and improve your **battery management system**
  - Develop and improve **accuracy of your models**
  - Automate **parameter estimation** to a model from your experimental data
  - Develop or assess and improve your **SOC or SOH estimation algorithms**
  - We seek to teach and build in-house competency through project-based coaching sessions and knowledge transfer.

Battery Modeling Development Service

MathWorks Consulting Services works with global companies in a wide range of industries, including aerospace, defense, automotive, and energy production in the development of battery models. Our Consultants have industry experience in developing models of Li-ion, NiMH and lead acid batteries, including automated techniques for fitting the models to measured datasets. Battery modeling can be a complicated and time-consuming task, depending on the level of accuracy required. Applying Model-Based Design to battery models, MathWorks Consultants work with you to establish a well-defined process for model development and parameter estimation which helps manage complexity and reduces development effort.

MathWorks Consulting Services engagements are highly customized to your particular project. We focus on knowledge transfer, collaborative model development and delivering practical, real-world solutions.

We teach

- Simulink® and Simscape™ for modeling battery cell equivalent circuits and battery packs.
- Optimal methods for cell test data acquisition and analysis
- Data processing and optimization
- Parameter estimation techniques using Simulink Design Optimization™ and Parallel Computing Toolbox™ for estimating battery parameters from measured data.

We develop

- Test plans for the battery cell or pack
- Plant models for offline and real-time simulation environments using Simulink®, Simscape™, SimPowerSystems™, and SimElectronics®
  - Multi-RC equivalent circuit cell models
  - Battery pack models from series or parallel cell configurations
  - Thermal models for the battery pack, conditioning system, ambient conditions, etc.
  - System level models, such as detailed plant models for each component of an electric vehicle
  - Creation and validation of system level plant models for detailed grid simulations, including three phase fault injection

We deliver

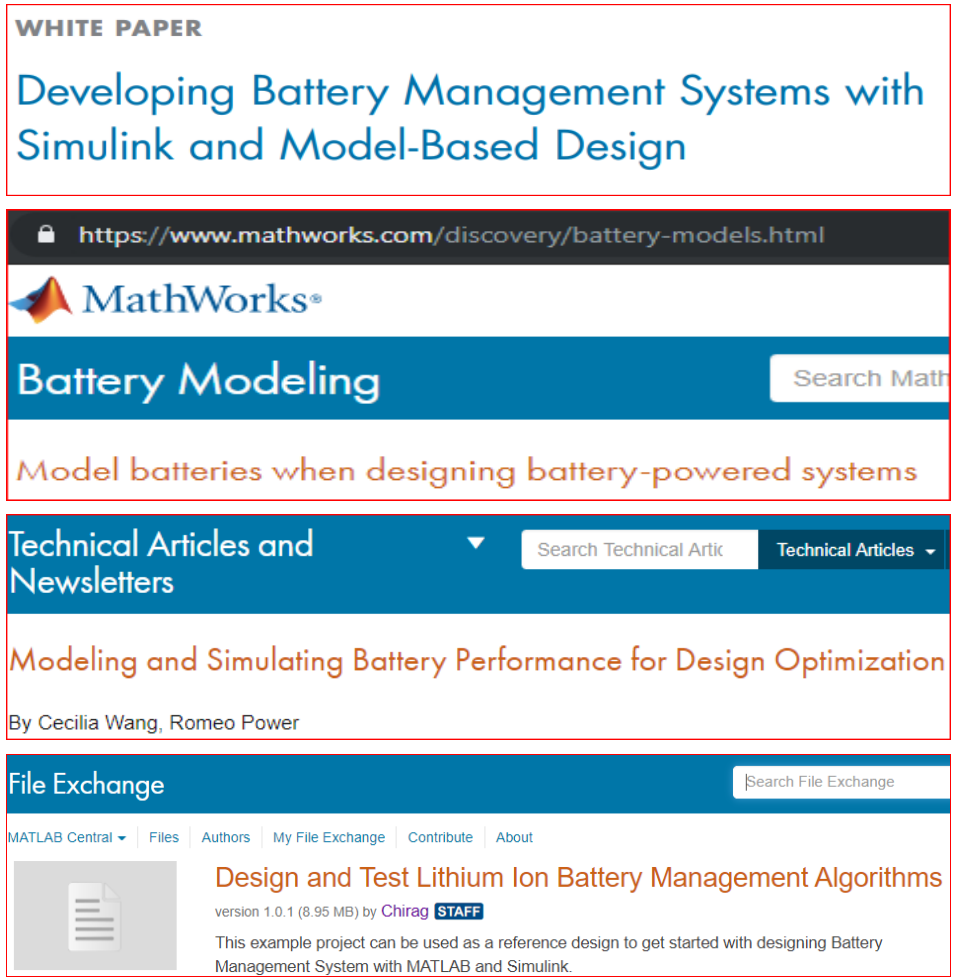
- Detailed models, providing an accurate simulation of your specific battery or complete system
- Automated battery parameter estimation techniques that you can apply and customize to similar data sets

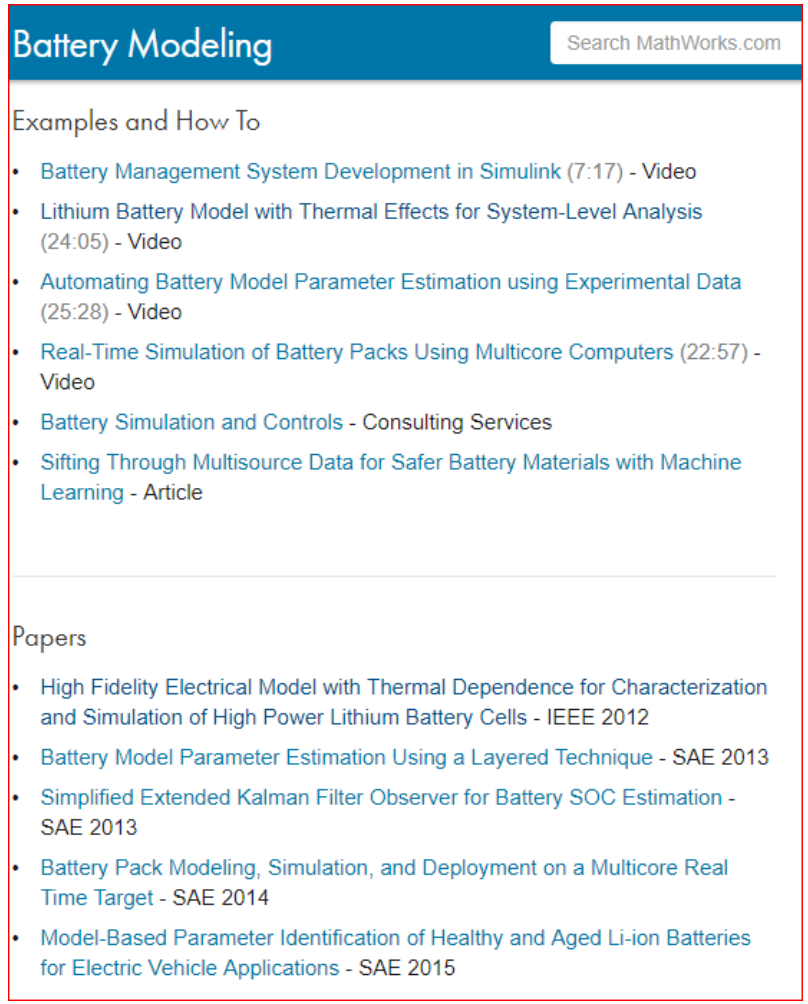
Working side-by-side with you, we help build your skills and experience and leave you independent and in control of your processes, tools and design work. MathWorks Consulting Services is ready to work with you on your battery modeling development project.

FOR MORE INFORMATION:
mathworks.com/battery
Contact MathWorks Consulting

MathWorks
Accelerating the pace of engineering and science
mathworks.com

More about Battery Management System





[Download Link to File Exchange]

For more info:
Prashant Hegde
phegde@mathworks.com

THANK YOU