C-026

車載バッテリコントローラの故障時向け実機レス検証環境の開発
Development of Virtual Prototyping for Hardware Failure Verification Environment for Automotive Battery Control System

黒川 能毅‡　勝 康夫‡
Yoshiki Kurokawa　Yasuo Sugure

1. はじめに

車載制御のソフトウェア分野においては、電動化や予防安全など自動車の高機能化により、プログラムの大規模化、複雑化が進んでいる。さらに 2011 年度に策定された自動車の機能安全国際規格 ISO26262 に基づいて、安全性、信頼性の高いソフトウェアを高効率に検証する技術が求められている。

この検証技術として、実機あるいは実機相当の動作をリアルタイムに模擬可能な専用装置を用いて車載システム全体を検証する Hardware In the Loop Simulation (以下 HILS と略す)がある。しかし、検証に実機や HILS のような環境を用いる場合、検証環境の準備に時間を要する、不具合発生時に不具合再現が困難なケースが存在する、機能安全性確認の為の故障注入が難しいなどの課題がある。

この課題を解決するため、PC 上でのシミュレーションによる実機レスで車載システムの検証を行う取り組みが研究されてきた[1][2][3][4]。

実機レス検証環境を適用することで、ソフトウェア出荷時の回帰試験において、実機の台数に制約されず、並列に検証を行うことで検証期間の短縮が見込める。また、ISO26262 が求める故障時の確認においても、実機では再現が難しい電子デバイスの故障、例えばマイコンやメモリの故障も容易に再現できる。特にハイブリッド車や電気自動車で使用されるリチウムイオンバッテリシステムの安全度水準 ASIL(Automotive Safety Integrity Level)は最も厳しい D レベルの対応が必要なケースがある。ASIL-D 製品の故障時検証は ISO26262 の規格では強く推奨されている。

以上の背景により、本報告では、リチウムイオンバッテリコントローラ（以下バッテリコントローラと称す）において、実機では起こしにくい搭載ソフトウェアのハードウェア故障時のソフトウェア動作検証を検証可能にする実機レス検証環境を構築した。本環境ではマイコンモデルを用いて実機で使用するオブジェクトコードが実行可能であるという特徴を持ち、オブジェクトコードに手を加えず直接検証ができる。この環境上でバッテリ制御部ハードウェアモデルの試作と評価を検討、実証した。

2. バッテリコントローラ概要と検証環境

2.1 バッテリコントローラ構成

バッテリコントローラは、複数の直列接続されたリチウムイオンバッテリの電圧のセンシングおよび、バランスコントロールを行うユニットである。複数のリチウムイオンバッテリ毎にセンシングおよびバランシングを行うためのコントローラチップ（以下セルコンと称する）を 1 個配置し、各々が配下のリチウムイオンバッテリの管理を行う。複数個のセルコンが配置され、セルコン同士はリングバスに接続される。後述するリングバスーSPI（Serial Peripheral Interface、以下 SPI と略す)通信チップを介してマイコンからセルコンへのコマンドの授受を行い、セルコン内のコントロールレジスタを操作し、センシングやバランシングを行う。リングバス上には、セルコン以外にマイコンからのコントロールを行うためのリングバスーSPI 通信チップがあり、SPI でマイコンと接続されている。リングバスはリングバスーSPI 通信チップを起点としてセルコンを順にデイジーチェーンで接続して、最後のセルコンの出力をリングバスーSPI 通信チップに入力し、終点としている。リングバスーSPI 通信チップは SPI から受信したリングバスへのリード・ライトのコマンドを出力し、結果をリングバス終点から受け取り、これを SPI 経由でマイコンに戻す。バッテリコントローラは、これ以外に外付け EEPROM、RTC、CAN バス、電源 IC で構成されている。

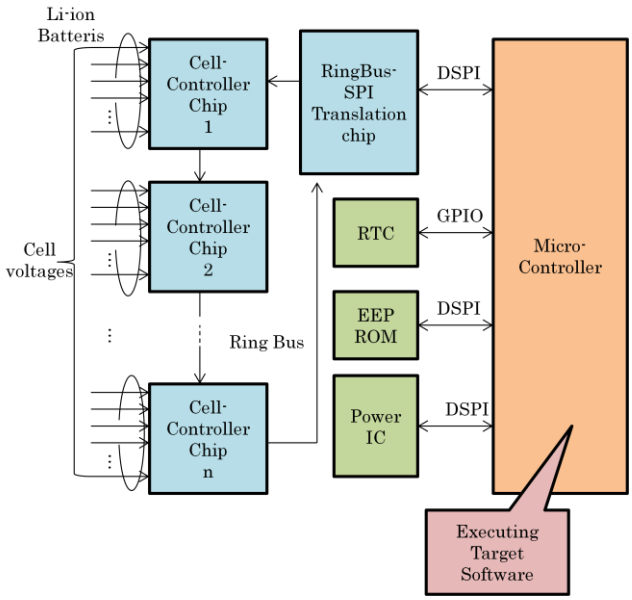


図 2-1　バッテリコントローラ構成

2.2 バッテリコントローラ検証の課題

現在、バッテリコントローラのソフトウェア出荷前検証はソフトウェア単体での検証と、HILS により検証を行っている。表 2-1 にバッテリコントローラソフトウェアの回帰試験の内訳を示す。出荷に向けて行う回帰試験は大きく分けて、以下の 2 種類に大別される。

Virtualizer ™は Synopsys 社の商標です。

(1) バッテリコントローラソフトウェアの最終オブジェクトコードを用いる HILS による試験
(2) ハードウェアで直接故障状態を起こすのが困難な項目について、ソフトウェアにテストソフトウェアを追加し、テストソフトウェアで疑似的に故障状態を起こす試験

表 2-1　バッテリコントローラソフトウェア回帰試験

#	検証種類	回帰試験に占める割合
1	HILSによる直接検証	80.8%
2	故障関連：テストソフト挿入による間接検証	19.2%

上記 2 種の回帰試験に占める内訳は、(1)が 80.8%、(2)が 19.2%である。このうち(2)については、テストソフトウェアを追加する性質上、バッテリコントローラソフトウェアの最終オブジェクトコードを用いてのテストができない。特にセルコンやリングバスーSPI 通信チップ内部の異常については、チップ内部のため、HILS を用いても意図した故障状態をコントロールできる状態で発生させることは困難である。表 2-2 に内容を示す。

表 2-2　HILS で困難な故障内容

故障種別	内容
回路周辺の異常	セル異常電圧の検知
	ヒューズ異常の検知
チップ内部の異常	チップ内部動作の異常検知
	バージョン等の確認、初期化の確認
通信の異常	リングバス通信異常検知

3. 実機レス検証環境の適用

3.1 実機レス検証環境概要

図 3-1 に構築する実機レス検証環境の概要を示す。実機レス検証環境とは、マイコンの実物の代わりに CPU、メモリ、バス、周辺回路のモデルを用いて、実 ECU に搭載するソフトウェアを実行し、ソフトウェアの動作の正しさを検証する手法である。実機レス検証環境は、マイコンシミュレータを中心に周辺回路のモデルを接続して仮想 ECU を構築し、これに実機で使用可能なオブジェクトコードを搭載して動作させる。仮想環境であるため、ソフトウェアの挙動を詳細に解析することが可能である。

実機レス検証環境は実機を用いないため、以下の点の特徴を持つ。
(1) 対象ハードウェアが存在しない状態でも先行してソフトウェアの動作検証が行える。
(2) 実行環境（シミュレーションエンジン）を複数準備することで、試験が並列に行える。
(3) 現象の再現やブレークポイントを指定することが実機と比較し容易である。

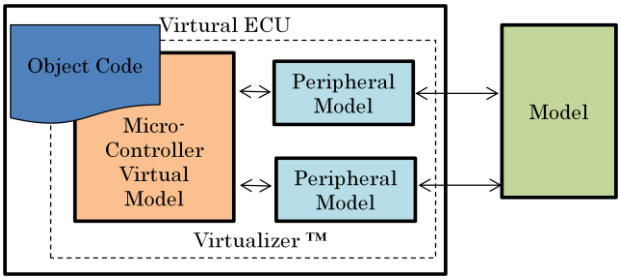


図 3-1　実機レス検証環境

3.2 バッテリコントローラへの適用の課題

実機レス検証環境では、セルコンはソフトウェアモデルであるため、チップ内部のエラーをソフトウェアモデルの改造で容易に生成でき、実行時間がかかっても今まで最終オブジェクトコードでできなかった試験ができると見込める。よって、これらの試験を実機レス検証環境で行い、最終オブジェクトコードで直接出荷時検証ができる範囲を広げることとした。

バッテリコントローラに実機レス検証環境を適用する際の課題 2 点を表 3-1 に挙げる。

表 3-1　実機レス検証環境を適用する際の課題点

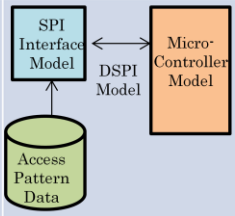
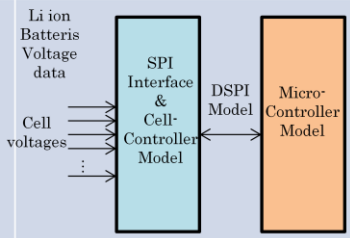
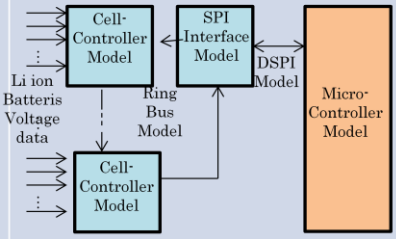
#	問題	課題
1	実行速度が遅い	実行速度と故障模擬の両立
2	チップ内部のエラーに関連する処理の検証が実機では難しい	故障注入機能のモデル化

(1) チップのモデルにおいても高速化が要求されるが、故障模擬にはチップを詳細にモデリングする必要があり、トレードオフとなる。そのため、実行速度と故障模擬の両立が課題となる。
(2) 実機では難しいチップ内部のエラーに関連する処理を直接試験するためには、故障を注入するための機能を作成するチップのモデルで実現する必要がある。

4. バッテリコントローラモデル化検討

本章ではバッテリコントローラハードウェアのモデル化に当たり次節から、実行速度と故障模擬の両立と、故障注入機能のモデル化の 2 点の課題の対策について検討をする。

表4-1　モデル化案

	案1	案2	案3
ブロック図			
モデル複雑度	◎	○	△
オペレーション再現性	△	○	◎
エラー再現性	△	△	◎
実行負荷	○	△	△

4.1 実行速度と故障模擬の両立

4.1.1 バッテリコントローラモデル構成の最適化

新規開発に必要なバッテリコントローラチップをどのようにモデル化するかを検討した。表 4-1 にモデル化案の比較を示す。

案 1 は SPI の受け口のみ作成し、パターンリストをテストベクタとして用意し、パターンリストに沿ってデータをマイコンに返す方式である。モデル開発工数は最小であるが、あらかじめ通信パターンを予測し、その通り通信する必要があるため、複雑な動作をする回帰試験等最終製品に近いオブジェクトコードの検証には向かない。また、パターン作成工数が大きく、複数のテストベクタを作成するのには向かない。

案 2 は SPI 受け口を用意し、モデル内部でセルコン搭載個数分の内部情報を持ち、SPI でのみ通信を行う方式である。UART 通信部分を作成しなくてよいため、その分モデル開発工数は小さくなる。ただし、セルコンには内部の異常を通知する機構があり、これを新たに追加しなくてはならないため、工数は増加する。また、将来的にセルコンの数量を変更した製品に対応するためには、コードに手を入れる必要がある。

案 3 はセルコンおよびリングバスーSPI 通信チップを別々にモデル化し、リングバスによる通信を行う方式である。モデル開発工数は 3 案の中で最大となるが、リングバスによる通信を模擬するため、通信時の異常を起こすテストベクタの作成が容易である。また、将来的にセルコンの数量を変更した製品への対応についても容易に行える。

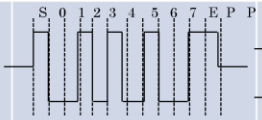
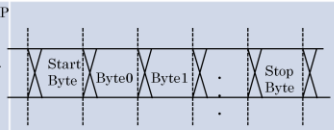
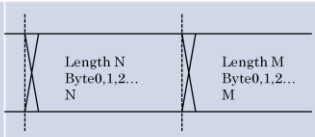
案 1、2、3 の比較において、モデル化の工数はかかるが、異常を起こすテストベクタの作成が容易となり、将来的にセルコンの数量を変更可能である案 3 での開発を行うこととした。

4.1.2 バス通信の抽象度の最適化

セルコンおよびリングバスーSPI 通信チップを別々にモデル化することにより、リングバスによるバス通信をモデル化することとなった。通信方式として、以下の表 4-2 に示す 3 案を検討した。

案 1 は通信をクロック単位で再現する方式である。リングバスで発生する全ての状況を生成可能であるが、モデル化の工数が大きく、シミュレーション実行速度も非常に遅くなると予想される。

表4-2　モデル通信方式案

	案1	案2	案3
タイミング図			
アクセス単位	ビット毎	バイト毎	コマンド毎
オペレーション再現性	◎	○	△
実行負荷	×	○	○

案 2 は通信をバイト単位で通信する方式である。一部リングバスで発生する異常を再現することはできないが、セルコンが異常フラグを持っているため、その異常フラグの操作によって状況再現を代用可能である。また、シミュレーション速度も高速であると予想される。

案 3 は通信を 1 リード、ライト単位でデータ授受を行う方式である。シミュレーション速度は高速であると考えられる反面、授受するデータ長が変化するため、データのハンドリングが複雑になると考えられる。

案 1、2、3 の比較において、通信の異常状態を再現しやすく、また、シミュレーションも高速であると予想される、案 2 のバイト単位で通信する方式を採用した。

4.2 故障注入機能のモデル化

4.2.1 データ通信部故障モデル化

リングバスの通信のエラー状態は、バスプロトコル内のエラーフラッグバイトと送信データ全体の CRC（Cycle Redundancy Check）チェックを行うバイトの 2 つのデータで検出、伝播させる。

このうち、エラーフラッグバイトは通信エラーだけでなく、セルコン内部のレジスタエラーについてもチェックビットを持ち、このエラーフラッグバイトをセルコンチップ間に伝播させることで、最終的にマイコンに通知している。エラーフラッグバイトについてセルコンモデルに省略することなく実装することで、全てのエラーについて起きたことを伝播してマイコンに伝達し、マイコンがどのセルコンチップで起きたエラーかを解析し、エラーとして所定のチップに対応する変数に登録するまでの一連の動作を検証可能とした。

また、CRC エラーについても、各チップで受信し CRC チェックを行うことで、エラーを検出している。また、次チップに対してもチェックが可能なように CRC を計算しなおして次チップに送信している。CRC データは最終的にマイコンに戻り、そこでも CRC チェックが行われるため、本機能に関しても省略することなく実装することで、CRC チェックが正常動作することを確認できるようにした。また、CRC エラー検知に関しても試験を行うために、モデル内部で意図的に CRC エラーを引き起こすよう、算出した CRC 値を変更する実装を行った。CRC 値を変更するタイミングを制御するために、チップには存在しない仮想入力ピンをモデルに追加し、このピンの値を論理的に真に設定すると、前記 CRC 値を変更するよう動作するようにモデルを改造した。このようにすることで、シミュレーション実行時の任意の時間で任意のセルコンモデル内で CRC エラーを引き起こすことが可能となった。

4.2.2 AD 変換パス故障モデル化

セルコンチップには、自身の AD 変換器の動作検証を行うためのテストレジスタおよび機能が備わっており、バッテリコントローラソフトウェアにおいてもこの機能を使用して自身の AD 変換器の定期的な検証を行っている。本機能のエラー検出をテスト可能とするためには、次の実装を行った。まず、テストレジスタの値から期待値が算出、出力されるように構成することによって、バッテリコントローラソフトウェアによる定期的テストがパスすることを確認した。その後、仮想入力ピンをセルコンモデルに追加し、仮想ピンに理論値の真が入力された場合に期待値がずれるようモデルを改造した。これにより、シミュレーション実行時に任意の時間で AD 変換器のテストエラーを起こすことが可能となった。

5. バッテリコントローラソフトウェア検証環境

前章の検討結果を反映して、シリアル通信チップモデルおよびセルコンチップモデルを SystemC を使用して実装を行った。プラットフォームに Virtualizer ™を使用しており、Virtualizer ™がサポートする記述方式に則りモデルの実装を行った。また、これら実装したモデルに加え、マイコンモデル、EEPROM モデル、RTC モデル、電源モデルを配置、接続し、実機レス検証環境でシミュレーションを行う環境を構築した。本環境において、ターゲットとなるバッテリコントローラソフトウェアを最終オブジェクトの形式でマイコンモデルを使用して動作させ、周辺回路である Direct Memory Access（以下 DMA と略す）によるリングバスーSPI 通信を行わせ、マイコンモデルと実装したリングバスーSPI 通信チップモデルおよびセルコンチップモデルがシミュレーション時間で 2 秒以上安定して協調動作を行っていることを確認し、故障時検証に十分なシミュレーション期間（800ms 以上）を得て、これにより作成したテストが完了した。

6. 結果

6.1 バッテリコントローラソフトウェア故障時実行時間

構築したバッテリコントローラソフトウェア検証環境において故障時の動作を模擬し、ソフトウェアが故障を検知することを検証した。表 6-1 に検証結果一覧を示す。

表 6-1　故障時実行時間

テストケース	シミュレーション時間[ms]	実行時間[s]	倍率
ケースA	278.3	598	2149
ケースB	361.3	1015	2809
ケースC	760.8	1863	2499
ケースD	278.3	496	1782
ケースE	329.4	601	1852

前出の表 2-2 で選定した項目についてそれぞれの項目のエラーを起こし、バッテリコントローラソフトウェアが検知し、結果をソフトウェア内の所定の変数の変化として検出するようテストベクタを作成した。各テストベクタはシミュレーション実行時間を短縮するため、変化が観測できる最短の時間で判定を行うようにした。テストは全てパス

し、それぞれのエラー発生において所定のメモリの値が期待値通りに変化した。

テストの実行速度に関しては、バーチャルマシン上に環境を構築しているため実行速度にばらつきがあるが、1700 倍から 2800 倍となっている。マイコンモデルがコアのみを単純動作させているときの性能が約 700 倍であり、これに周辺機能である DMA と SPI、SPI 通信チップモデル、セルコンモデル、EEPROM モデルが並行動作するためと考えられる。

今回試験としてはセルコンの 2 番目のチップについてのエラーを起こし検知をテストしたが、テストしたケースでは、セルコンは 6 個搭載され、全体をカバーするテストを行うには、実行時間も 6 倍かかると考えられる。今回作成したテスト全てで 2 時間半程度の実行時間となるため、全部行うには 15 時間かかると推測される。本テストは現在の回帰試験と並行して実行可能なため、現在の回帰試験に対し、回帰試験期間の延長無しに追加できる見通しを得た。

6.2 バッテリコントローラ回帰試験検証カバー率

表 6-2 に、この試験の追加後による、回帰試験での直接検証可能な範囲の検証カバー率を示す。表 2-1 と比較し直接検証が可能な範囲を 80.8%から 89.4%に改善できる見通しを得た。今後は、更なる SystemC によるモデル化を行うことにより、故障検出時の検証が可能となる範囲を広げ、直接検証可能な回帰試験の範囲を広げていけると考えられる。

表 6-2　回帰試験の検証カバー率

#	検証種類	回帰試験に占める割合
1-1	HILSによる直接検証	89.4%
1-2	故障関連 ： 実機レス検証環境による直接検証	
2	故障関連:テストソフト挿入による間接検証	9.6%

7. おわりに

車載リチウムイオンバッテリコントローラ向けソフトウェアに、実機レス検証環境を構築した。課題解決のため、以下の２点を検討した。

(1)　バスアクセス時のエラーおよびアナログ・デジタル変換回路テストのエラー注入モデル化

(2)　疑似エラーを容易に注入可能でかつシミュレーション実行時間に影響が少なくなるモデル構成の最適化とバスモデル抽象度の決定

本モデルを使用し、集積回路部分の故障時応答確認テストについて、これまでソフトウェアで疑似エラーを起こすことで間接的に検証していたテスト項目を、最終製品オブジェクトコードの状態で直接テストし、故障テストの動作確認ができた。テストにかかる時間を評価した結果、回帰試験と並行して行うことで回帰試験期間の延長無しに、直接検証が可能な範囲を 80.8%から 89.4%に改善できる見通しを得た。

参考文献

[1]　Y. Niimi、T. Ono、and N. Tsuchitya、” Virtual Development of Engine ECU by Modeling Technology”、 SAE Technical Paper 2012-01-0007 、(2012)

[2]　新見幸秀、小野 孝幸,、石原 秀昭、” モデリング技術によるエンジン ECU の仮想開発”、 自動車技術会 p7～p9、(2011)

[3]　伊藤康宏、勝康夫、於保茂、”Virtual HILS”、2011 CPSY/DC/EMB/SLDM 研究会、信学技報 Vol.110 No.473、p243-247、(2011)

[4]　S. Nakao、M. Shimozawa、and Y. Sugure、”Virtual FMEA : Simulation-Based ECU Electrical Failure Mode and Effects Analysis,”、SAE Technical Paper 2014-01-0205、(2014)

‡株式会社　日立製作所　研究開発グループ
Hitachi, Ltd., Research & Development Group