MILSのシミュレーション時間低減による開発効率化への取り組み

大嶋　歩 *　　松永　賢太郎 *　　金　ジェホ *

抄　録

制御ソフトウェア開発では効率的に開発できるようモデルベース開発が一般的になっており，モデル設計段階においてはバーチャル上で制御検証することが重要である．しかし現状，トランスミッションの制御ソフトウェアの検証に用いているMILSはシミュレーション時間が非常に長い．そこでシミュレーション時間を低減できる手法やツールを導入し，これまでより効率的に開発が行えるようにした．本稿ではその取り組み内容について紹介する．

1. まえがき

制御ソフトウェアの開発では，制御の各機能をブロックで表現したモデルベース開発が広く普及してきた．制御仕様をモデルで記述した後，そのままシミュレーションすることで，設計と検証を短いサイクルで回すことができる (Fig. 1)．このようなモデルを用いた検証手法や環境のことをModel In-the Loop Simulation (以下MILS) と呼んでいる．例えば，開発プロセス内の後半にある実車検証フェーズにおいて設計ミスが発覚した場合，設計フェーズに戻ってやり直すといった大きな手戻りが発生してしまう．しかし，設計フェーズの時点であらかじめMILSを使った検証をしていれば，そこでミスが発覚しても小さな手戻りで制御ソフトウェアを修正できる．つまり，効率的な開発が実現できる[(1)]．

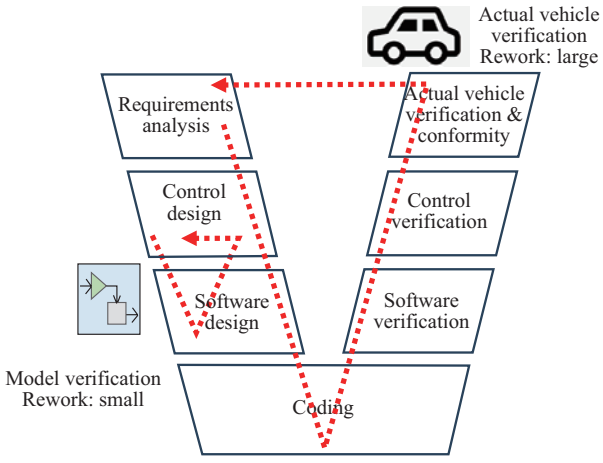

Fig. 1 V process of control development

我々も，制御ソフトウェアをトランスミッションコントロールユニット (以下ATCU) に実装する前に，モデルの段階で検証できるようトランスミッション制御MILS (以下TM制御MILS) を準備している．しかし，このTM制御MILSはシミュレーション時間が非常に長い．そこで，MILSのシミュレーション時間を低減させる取り組みを行った．本稿ではこの取り組みについて紹介する．

2. TM制御MILSの現状と課題

まずTM制御MILSの構成を述べる．実車操作を模擬したドライバモデル，実車やトランスミッション実機を物理式で表現したプラントモデル，トランスミッション制御が入ったコントローラモデルの3つからなり，これらを接続して1つのモデルとして，シミュレーションしている (Fig.2)．

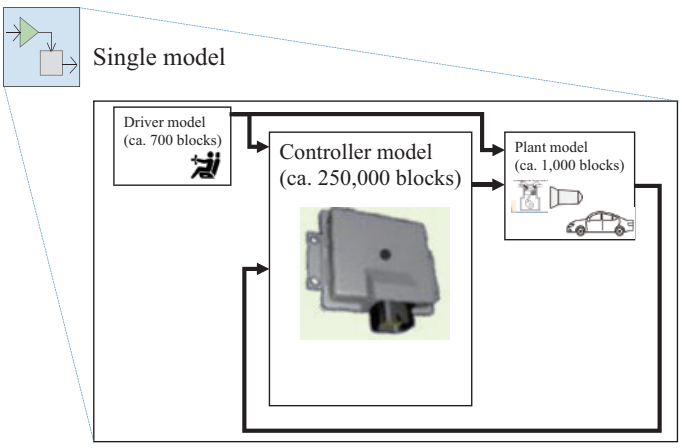

Fig. 2 Overview of the TM-control MILS

* 制御システム開発部

このうち，ドライバモデルやプラントモデルは必要最低限の機能に限定しているのに対し，コントローラモデルはATCUに実装している制御ソフトウェア全てを組み込んでいるため，他のモデルに比べてブロック数が多くボリュームが大きくなっている．それが原因でTM制御MILS全体のシミュレーション時間が長くなっている．具体的には，30sのテストケースを用いてシミュレーションすると，計算が終了するまでに20倍の600s程度の時間が掛かっている．

そこで，シミュレーション時間を低減できるか検討を始めた．低減後の目標値は，シミュレーション時間をリアルタイムの2倍以内に収めることとした．

3. 課題解決の手法

TM制御MILSのシミュレーション時間を低減させる手法として，以下2つの手法に取り組んだ．

(1) モデルを実行形式に変換してシミュレーションする手法

モデルを実行形式に変換する手法を説明する．通常，モデルは記述されている演算を1か所ずつ解釈し，機械語に翻訳しながらシミュレーションしている．演算を1か所ずつ解釈していくため，例えば，どの演算結果を出力するか条件分岐によって切り替えるモデルだと，全演算を行った上で最終的に条件に沿った結果が出力される．また同時に，都度機械語に翻訳する手間が発生している．これらが原因となり，シミュレーション時間が長くなっている．

それに対して，モデルを実行形式に変換してシミュレーションする手法の場合，まず演算全体を機械語に変換する．その際，演算全体を見て条件分岐に該当する部分などを最適化して必要最低限の演算のみを行う．こうすることで余計な手間が省かれ，結果シミュレーション時間が低減できる (Fig.3).

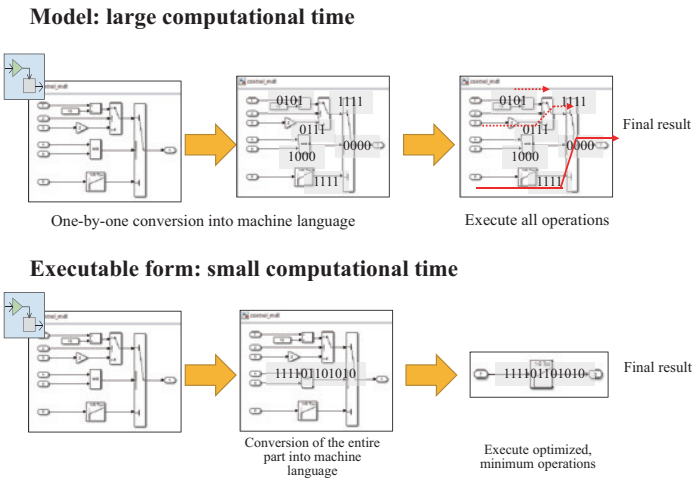

Fig. 3 Difference between a model and an executable form

この手法をTM制御MILSに対して適用するためにdSPACE社のソフトウェアツールVEOSを用いた (Fig. 4). VEOSは，モデルの取り込み，モデルから実行形式に変換するためのビルド，実行形式ファイルでのシミュレーション，という機能が備わったツールである．

なお，ボリュームの大きいモデルをVEOSに取り込むと，モデルをビルドする際にエラーが発生してしまう．このエラーを回避するには，VEOSを使う前に，1つのモデルを複数のボリュームが小さなモデルに分割してからVEOSに取り込む必要がある．TM制御MILSもボリュームが大きいモデルになるため，複数のモデルに分割することが考えられる．

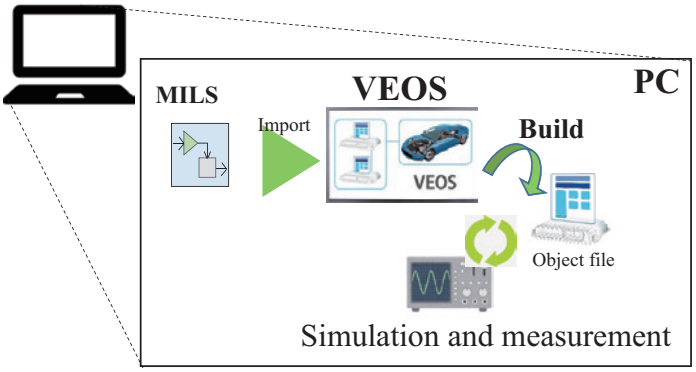

Fig. 4 Simulation environment using VEOS

(2) モデルを専用マシン上でシミュレーションする手法

モデルを専用マシン上でシミュレーションする手法を説明する．パソコンとは別に，モデルのシミュレーションに特化したハードウェアを用いる．専用マシン内のプロセッサコアにモデルをダウンロードしてから，シミュレーショ

ンを行う．この専用マシンは，高い演算能力を持つプロセッサコアとリアルタイムOSを備えたハードウェアツールであるdSPACE社のSCALEXIOプロセッサユニット (以下SCALEXIO) を用いた (Fig. 5).

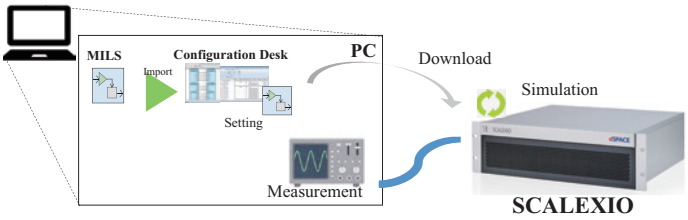

Fig. 5 Simulation environment using SCALEXIO

4. シミュレーション時間短縮の結果

ここでは各ツールを用いたシミュレーション時間短縮の結果を述べる.

(1) モデルを実行形式に変換したときの結果

TM制御MILSをVEOSで実行形式に変換してシミュレーションを行った結果，シミュレーション時間がリアルタイムの4倍程度の時間まで短縮された.

3章で述べたようにTM制御MILSそのままでは，モデルブロック数が多いため，VEOS上でビルドエラーが発生する．そのため，1つのモデルで作られているTM制御MILSを，それぞれのモデルブロック数が均等になるよう複数のモデルに分割した．分割した後に各モデルをVEOSに取り込むと，ビルド時にエラーなく実行形式ファイルに変換することができた (Fig. 6).

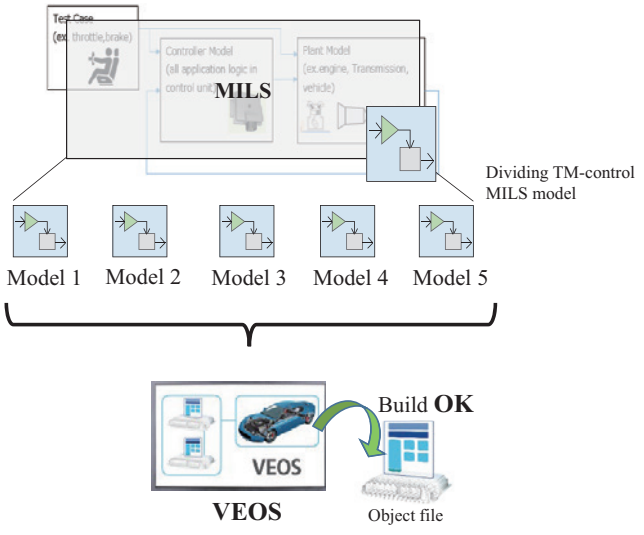

Fig. 6 Model division for MILS

VEOSを使う中で，モデルの分割数とシミュレーション時間には相関があることが分かった．分割数とシミュレーション時間の関係を Fig. 7に示す．こちらは30sテストケースを例にとっている．今回10分割まで試したが，その結果，分割数に応じて120s~300s (4倍から10倍) とシミュレーション時間が長くなることが分かった.

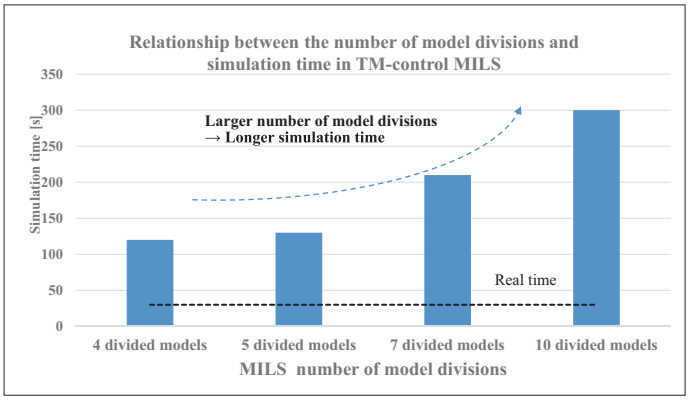

Fig. 7 Relationship between the number of model divisions and simulation time

分割数とシミュレーション時間に相関が有るのは，1つのモデル演算が終わってから次のモデルの演算を行うといった演算順序の指定が影響している (Fig. 8). 分割数が多くなるほど演算順序の指定箇所が増えてしまうため，それだけ演算の待ち時間が発生し，結果シミュレーション時間が長くなってしまう．この演算順序の指定がないと，各モデル間の演算順序が不定となってしまう．この場合，分割する前のTM制御MILSとは異なる演算順序になってしまい，狙いの結果が出力されないこととなる．しかし，モデル間で演算順序を指定することで，分割前のMILSと同じ演算となり狙いの演算結果が得られる.

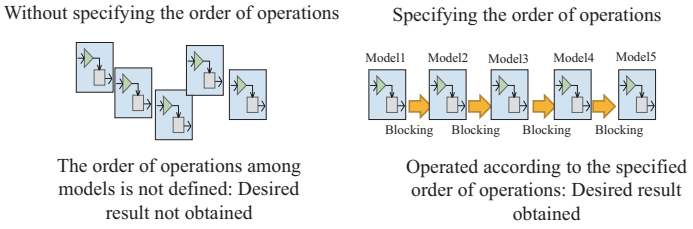

Fig. 8 Specification of the order of operations

また，MILSを用いた検証はTM制御全体だけでなく，ボリュームの小さい単一機能MILSという使い方もある．これは，ある制御機能のみを検証したい場合，その制御機能のコントローラモデルとプラントモデル部分に限定した

MILSとなっている．一例として，変速制御機能のみに限定した単一機能MILSであれば，リアルタイムに対して半分程度の時間にまでシミュレーション時間が短縮されている (Fig. 9)．実際の開発においては，ある制御機能だけを検証する際は，単一機能MILSで必要な検証ができるケースもあり，より短いシミュレーション時間でMILSを使った検証が可能となる．

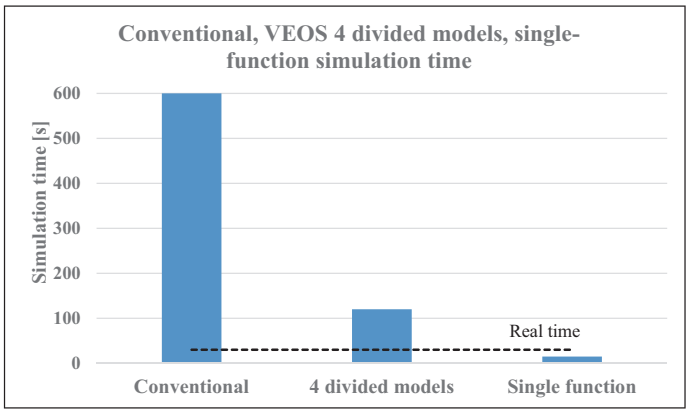

Fig. 9 Comparison of simulation time between conventional MILS and VEOS

(2) 専用マシンでの結果

SCALEXIOを用いると，TM制御MILSのシミュレーションを行う場合でも，VEOS使用時のようにモデルを分割することなく，リアルタイムと同等の時間でシミュレーションすることができた (Fig. 10)．これはMILSのシミュレーション実行をSCALEXIOという高性能な専用マシンで行っていることと，SCALEXIOが一般のパソコンで使われているOSと異なるリアルタイムOSを有しているためである．

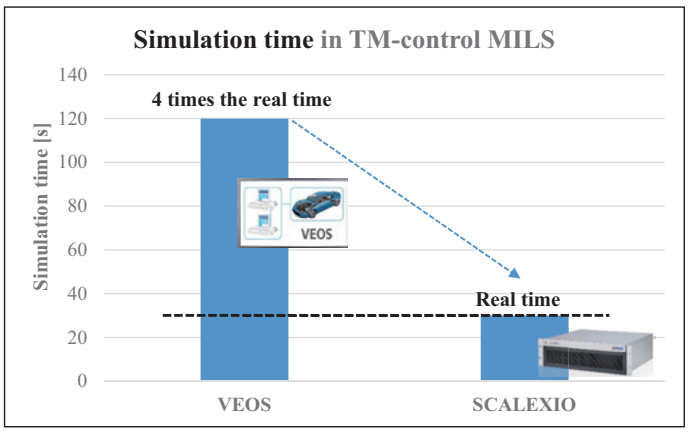

Fig. 10 Simulation time of TM-control MILS

この手法の場合，パソコンだけでなく専用のハードウェアが必要になるため，設計者一人ひとりが個別で行うには多くの設備投資が必要となる．しかし，TM制御MILSのようなボリュームの大きいMILSにおいても，モデルを分割することなくリアルタイムでシミュレーションすることができるのは，使用頻度は高くなくても非常に有益であるため，数人で1台を使うという共有のリソースとして運用するのが理想的と考える．

5. 結論

今回のTM制御MILSのシミュレーション時間低減の取り組みにおいて，目標値と置いたリアルタイムの2倍以内のシミュレーション時間を達成した．

モデルを実行形式に変換したときの結果は，従来リアルタイムの20倍程度掛かっていたシミュレーション時間が4倍から10倍の時間に低減された．単一機能MILSであればリアルタイム以下のシミュレーション時間まで低減できた．専用マシンでのシミュレーション結果は，リアルタイムと同じシミュレーション時間まで低減できた．

6. 考察

このように2つの手法とも，シミュレーション時間短縮ができるため，MILSの使い勝手向上に効果があると言える．このMILSを活用することで，より効率的な制御ソフトウェア開発が出来ると考えている．

今回2つの手法を用いたが，実際にMILSで検証する際は，用途に応じて使い分けするのが望ましい．部分的な検証で良い場合は，単一機能MILSを実行形式に変換することで，リアルタイムよりも短い時間でシミュレーションするのが有効である．一方，制御ソフトウェア全体を検証する場合には，TM制御MILSをモデルの分割をしなくてもシミュレーションできる専用マシンを用いるのが有効である．

7. 今後の課題

今回取り組んだ2つの手法は，どちらもツールが必要となるため，社内で普及させるには必要なツール数を精査し，設備投資計画に落とし込む必要がある．また実際に開発プロセスの中でMILSやツールを使う際に，具体的にどの設計フェーズで使うべきか，開発者内で共通のMILSを使う際の管理，運用ルールを決めることが今後の課題となる．

技術開発としては，ATCUの実機が無い状態でも制御の検証ができるよう，バーチャル検証領域の拡大を検討している．現状のTM制御MILSでは，OS部分を簡易的にモデル化しているが，これを実際のソフトウェアと組み合わせてATCU全体を模擬できる仮想ECUシミュレーション環境構築を検討している．これによって，ATCU実機や実車が必要となるような検証内容でも，仮想ECUがあれば設計フェーズで検証ができるようになる．

このようにモデルベース開発のメリットを有効活用し，更なる開発効率化をこれからも追求していく．

8. 参考文献

(1)藤塚亮平，原田大輔，馬場勇輝：MILS (Model In the Loop Simulation) を用いた制御検証の効率化，JATCO Technical Review No.18, pp.39 - 44, 2019.

(2)山形大輔：実験業務へのMBD活用による開発品質向上～CVT開発におけるVRS適用～，JATCO Technical Review No.21, pp.11 - 14, 2022.

■ 著者 ■

大嶋　歩

松永　賢太郎

金　ジェホ