Vol.2014-SLDM-165 No.5
Vol.2014-EMB-32 No.5
2014/3/15

モデル駆動開発を支援するための シミュレーション高速化に関する検討

大原貴都[†1]　藤平達[†1]　茂岡知彦[†1]　新田泰広[†2]　岩崎力[†2]　杉山達也[†2]

組込みシステムの開発効率向上のため，MATLAB [注1]/Simulink [注2]を用いたモデル駆動開発の適用が進められている．モデル駆動開発の一つである SILS では，S-Function ブロックを用いてソフトウェアをモデル上に取り込む．しかし，S-Function ブロックを用いた際，ブロック間の信号量に比例してシミュレーション速度が低下する課題がある．本稿では，共有ライブラリを用いることで Simulink ブロック間の信号量を削減し，高速なシミュレーションを可能とするモデル構成を提案する．提案手法を業務用空調機器に適用し，暖房のシミュレーション時間を約 90%削減し，本提案手法の有効性を確認する．

A Study of Simulation Acceleration Method for Supporting Model Based Development.

TAKATOSHI OHARA[†1]　TORU FUJIHIRA[†1]　TOMOHIKO SHIGEOKA[†1]
YASUHIRO NITTA[†2]　CHIKARA IWAZAKI[†2]　TATSUYA SUGIYAMA[†2]

Recently, embedded system development becomes complicated and the development costs increase. For development efficiency improvement, MBD (Model Based Development) is widely prevalent. MBD has HILS (Hardware In the Loop Simulation) and SILS (Software In the Loop Simulation). In this report, we focus on SILS. Improvement of the development efficiency is possible by applying SILS. However, Simulation speed may decrease by applying SILS. Therefore, we propose a novel model structure. The proposed model structure utilizes shared library for reducing quantity of data communication between the S-Function block. We clarify the effectiveness of the proposed model structure by an experiment. By the experiment, we apply the structure to the packaged air conditioning system. As a result, we realize about 90% reduction of the simulation time.

1. はじめに

携帯電話，カメラ，自動車等，身近な製品のエレクトロニクス化に伴い，組込みソフトウェアが複雑化，大規模化している．図 1，図 2 に組込みソフトウェアに係る企業を対象とした実態調査報告 1)の一部を示す．ソフトウェア開発費が 4 割強を占めており，開発コスト増大の要因となっている．また，不具合内訳をみると，ソフトウェア起因の不具合が半数を超えており，組込みソフトウェアの信頼性確保が難しくなっている．

このような状況を鑑みて，自動車等に代表される制御系組込みソフトウェア開発では開発効率向上や信頼性向上のため, MATLAB [注1]/Simulink [注2] 2)を用いたモデルベース駆動開発(Model Based Development．以降，MBD)が普及している．MATLAB [注1]/Simulink [注2]ではブロック線図形式のモデル(以降, Simulink モデル)で制御ロジックを記述する．作成した Simulink モデルを用いてシミュレーションを行うことで制御ロジックの誤りを発見できる．シミュレーションは HILS(Hardware in the Loop Simulation)，SILS(Software in the Loop Simulation)がある 3)．HILS は制御コントローラとして実機を用いるため実利用形態に近い環境でシミュレーション可能である．これにより，大幅に検証にかかる工数が削減され，机上で効率の良い検証が可能である．さらなる開発期間短縮のため，すべてをソフトウェアでシミュレーションする SILS が普及しつつある．しかし，SILS を用いた際，シミュレーションに時間が掛かり，検証効率があがらない場合がある．

例えば，文献 4)では，ECU（Engine Control Unit）に搭載する車両制御ソフトウェアの検査品質向上のため MBD を適用している．SILS による上流工程のロジック検証の際，シミュレーション速度が十分でなく，実用化への課題である．シミュレーション対象を車両制御ソフトウェア動作に必要なものに限定することで, HILS と同等の検査精度を保ち，高速動作可能な SILS を実現している．しかし，文献 4)の手法では，検証対象に合わせて，限定するシミュレーション対象を検討しなければならない．また，マイコン内部のソフトウェア処理を割込み制御方式からイベント駆動方式に変更を加えている．このため，修正工数および修正内容の検証工数が必要となる．

そこで我々は, 検証対象のソースコードの変更を行わず，シミュレーションの高速化を実現する手法を提案する．

†1 株式会社日立製作所　横浜研究所
Hitachi Ltd., Yokohama Research Laboratory
†2 日立アプライアンス株式会社
Hitachi Appliances, Inc.
注 1 MATLAB は The MathWorks, Inc.の登録商標.
注 2 Simulink は The MathWorks, Inc.の登録商標.

Vol.2014-SLDM-165 No.5
Vol.2014-EMB-32 No.5
2014/3/15

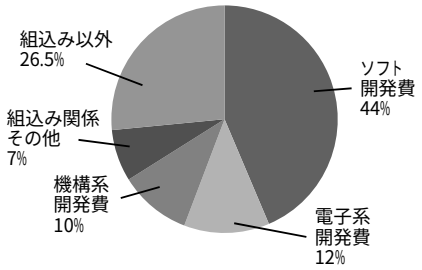


図 1　組込みソフトウェア開発費に関する調査結果
Figure 1　Development cost of embedded system.

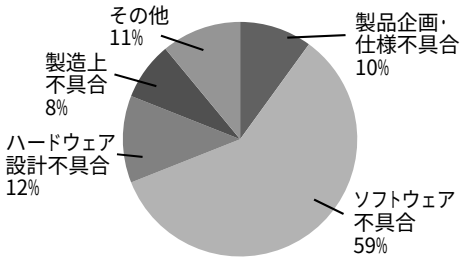


図 2　組込みソフトウェア不具合に関する調査結果
Figure 2　Failure of embedded system.

以下，2 章では SILS におけるシミュレーション速度の低下原因について述べる．3 章でモデル構成を提案し，4 章で提案手法を実製品に適用した結果を述べる．5 章で考察を行い，最後に 6 章で本稿についてまとめる．

2. シミュレーション速度低下に関する検討

2.1 SILS における Simulink モデル

一般に，制御系組込みソフトウェア開発は図 3 に示すように，仕様設計，実装，検証という工程を繰り返し行って完成度を高めていく．

検証の変遷に着目すると，シミュレーションツールが存在しない時代には実際の製品に搭載，動作させて検証を行っており，検証の準備および実施に時間が必要である．

この課題を解決するために，実機相当のシミュレーションを PC 上で行う HILS が登場した．HILS では，検証対象の制御仕様を実装した制御コントローラの実機と制御対象を模擬する専用ハードウェアを接続し，シミュレーションを行う．実機と同等の状態を再現することが出来るが，ソフトウェアの仕様設計局面では制御コントローラの実機が存在しないことが多い．このため，従来は類似システムを改造して使用しているが，検証準備工数を押し上げる要因となっている．

このような状況の中，さらなる開発期間短縮が求められ，すべてをソフトウェアでシミュレーションする SILS が普及しつつある．SILS は制御コントローラおよび制御対象を

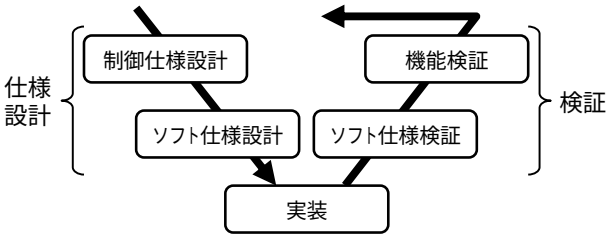


図 3　仕様開発および検証プロセス
Figure 3　Process of specification development and verification.

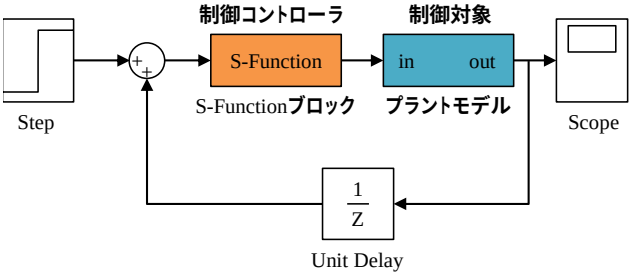


図 4　SILS における Simulink モデル
Figure 4　Simulink model for SILS

C 言語などのソフトウェアで模擬し，シミュレーションする．SILS はソフトウェアのみで構成されるため HILS より安価であり，一人一台のシステム利用環境を可能とする．また, HILS で必要であった実機の準備工数を短縮することが可能である．具体的には，図 4 に示すような Simulink モデルを用いてシミュレーションを行う．制御コントローラおよび制御対象をそれぞれ，S-Function ブロックおよびプラントモデルとして取り込む．取り込んだモデルをブロック線で結合し，フィードバックループなどを形成してシミュレーションを実現する．

ここで S-Function ブロックとは MATLAB [注1]/Simulink [注2] に含まれる基本機能である．C 言語や FORTRAN 言語などで記述されたソフトウェアを Simulink モデル上に取り込み，動作させることが可能である．SILS を実現する際，一般的に用いられるブロックであり，Simulink モデルから自動生成されたソースコードやレガシーシステムのソースコードを取り込み，シミュレーションを行う．

2.2 S-Function ブロックにおけるシミュレーション速度の低下原因調査

S-Function ブロックを用いてシミュレーションを行う際，シミュレーション速度が低下する場合がある．原因を調査するため，実験を行う．実験では，S-Function ブロック数，ブロックの演算処理量，ファイル入出力，信号量をモデル構造上の影響因子候補として設定する．図 5 に実験に用いる Simulink モデル，表 1 に影響因子の設定値を示す．シミュレーションのサンプル時間は 1 ミリ秒，シミュレーション時間は 1500 秒とし，シミュレーションを実施する．

Vol.2014-SLDM-165 No.5
Vol.2014-EMB-32 No.5
2014/3/15

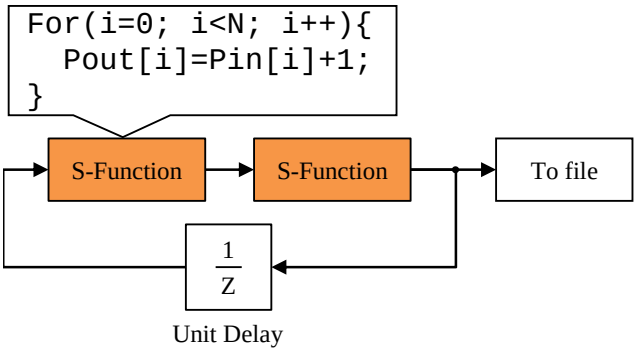


図 5　シミュレーション時間測定に用いた Simulink モデル
Figure 5 The Simulink model for simulation time measurement.

表 1 シミュレーション時間測定に用いた設定値
Table 1　The setting parameters for simulation time measurement.

#	処理条件	信号量	ブロック数
1	演算処理無 ファイル出力無	1,10,100, 3000,5000	1,2,3,5
2	演算処理無 ファイル出力有	1,10,100, 3000,5000	1,2,3,5
3	演算処理有 ファイル出力無	1,10,100, 3000,5000	1,2,3,5

測定結果を図 6 および図 7 に示す．まず，図 6 では，信号量に比例してシミュレーション時間が増加し，処理内容を変えた場合の処理速度は変化が少ないことが分かった．また，ファイル出力の有無については処理速度の変化は少なく，ボトルネックではないことが分かった．次に，図 7 では，ブロック数の増加と共に，処理時間が増加することが分かった．

以上の実験結果より，S-Function ブロックを用いた際にブロックの数およびブロック間の信号量がシミュレーション速度低下要因であることが分かった．

3. 提案手法

3.1 共有ライブラリを用いた高速化

2.2 節で述べたように，S-Function ブロックはブロック数およびブロック間の信号量に比例してシミュレーション時間が増加する．シミュレーション時間短縮のためにはブロック数の削減，もしくはブロック間の信号量削減が必要である．

そこで提案手法では，S-Function ブロック間の信号量削減を行う．図 8 に，提案する Simulink モデル構成を示す．これまで，シミュレーション対象のソースコードは S-Function ブロックを用いて参照し，モデル上に取り込んでいる．S-Function ブロック間で共有する必要があるデータは信号線を経由して転送している．これは，異なる S-Function ブロックから同一のソースコードを参照した場合，ソースコードにおける各変数の値はブロック間で共有さないためである．

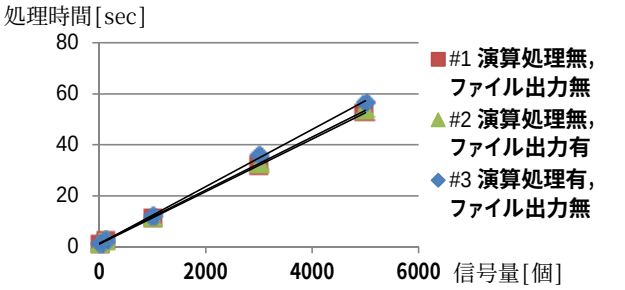


図 6　信号量および演算処理を変更した際の処理時間
Figure 6　The simulation time of changing quantity of signal and processing contents.

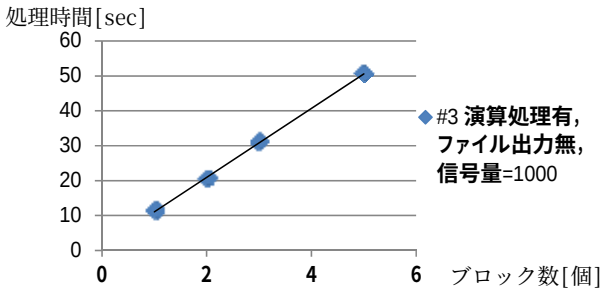


図 7 ブロック数を変更した際の処理時間
Figure 7　The simulation time of changing number of S-Function block.

提案手法では，S-Function ブロックが参照する対象を共有ライブラリとする．この時，参照している共有ライブラリ上で各種変数値を保持することが可能であり，異なる S-Function ブロック間でデータを共有することが出来る．この性質を利用することで，信号線による値の転送が不要となり，信号量を削除することが可能である．

3.2 その他の高速化方法の検討

3.1 節により，S-Function ブロックの特性に起因したシミュレーション速度低下は回避可能である．これ以外にもシミュレーション速度向上が可能であるか調査した．結果を表 2 に示す．表 2 の中から，3.1 節で示した手法と併用可能な項目を選定する．

#2，#4，#5 はシミュレーション対象とする制御システムの特性に合わせて設定する項目である．本稿で対象とする SILS では，シミュレーション対象の制御システムの形態はソースコードである．このため，MATLAB [注1]/Simulink [注2] 固有のソルバを用いずに計算を実施する．よって，これらの項目は対象外とする．

#6 は，例えば画像処理のような反復処理が多い場合に高い効果を発揮する．しかし，適用するにあたり，既存のソースコードを並列化可能なように変更を加える必要があるため，実施にコストが掛かってしまう．また変更を加えた際に不具合が混入する可能性があるため，検証として望ましくない．よって，本稿では検討しない．

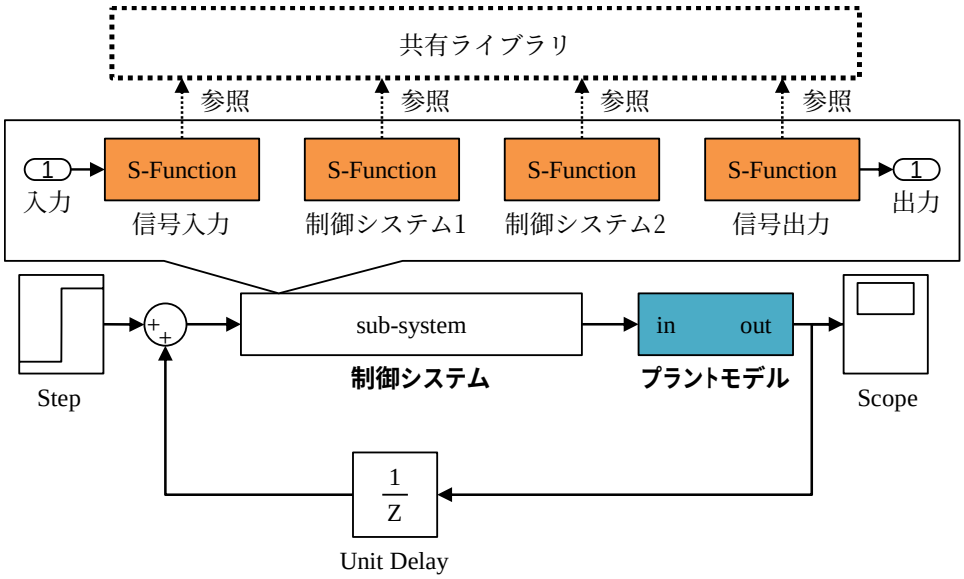


図 8 提案する Simulink モデル構成

Figure 8 The proposed Simulink model structure.

表 2 シミュレーション速度改善対策リスト

Table 2 List of simulation speed improvement methods

#	方法	内容
1	アクセラレータモード	通常インタプリタ方式．本モードを設定するとバイナリコードを出力・実行し高速化される．
2	適切なソルバ利用	stiff なシステムの陰的解法．連続系にのみ適用可
3	データログの削除	スコープやディスプレイなどのログデータ出力により速度低下するので削除
4	適応ゼロクロッシング	連続系システムでゼロクロッシングを回避して速度向上
5	代数ループの回避	遅延ブロック挿入して代数ループを回避し，速度低下を防ぐ
6	並列化	反復処理の並列化で高速化
7	ビルドの高速化	並行モデルのビルドを高速化する

#7 に関してはビルドの高速化であり，シミュレーション速度の向上には直接影響を与えるものではないため，本稿では検討しない．

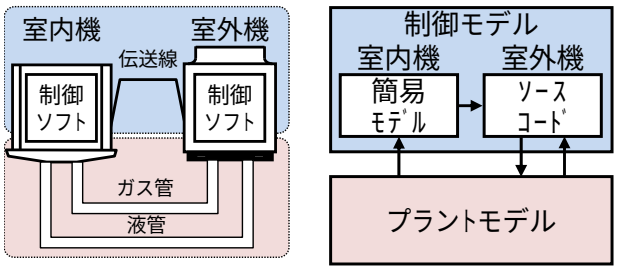
#1, #3 はシミュレーション対象の特性によらず適用が可能である．#1 は速度向上に高い効果があることが知られている．また，細かい設定を必要としないため導入が容易であり，本提案手法に導入する．#3 に関しては，出力するデータは大まかにシミュレーション結果とデバッグ用の中間データに分かれる．デバッグ用の中間データ出力はシミュレータ構築時に活用されるデータであり，構築後は不要となる場合が多い．提案手法ではデバッグ用中間データの出力を切換え可能とする．

以上より，提案手法では共有ライブラリおよびアクセラレータモードを用いてシミュレーションの高速化を行う．

表 3 実験環境

Table 3 Measuring condition

CPU	Inte Core 注3 7-3820 3.60GHz,quad-core,8-thread
Memory	PC3-12800 32GB （available area = 3GB due to 32bit OS）
OS	Windows 注4 7 Enterprise SP1 32bit
MATLAB 注1/Simulink 注2 Version	R2010b SP1
Simulation Time	1500sec. from system start.



(a)業務用空調機器 (b)モデル化

図 9 業務用空調機器およびそのモデル化の概要

Figure 9 Packaged air conditioning system and the modeling.

注 3 Intel Core は Intel Corporation の登録商標.
注 4 Windows は Microsoft Corporation の登録商標.

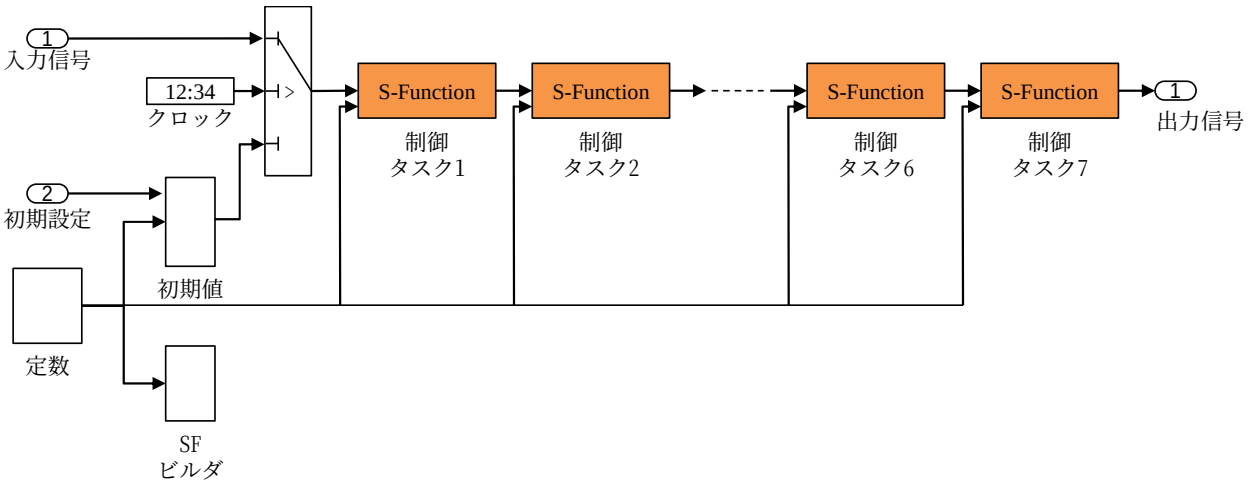


図 10　従来の室外機モデル
Figure 10　Conventional compressor unit model.

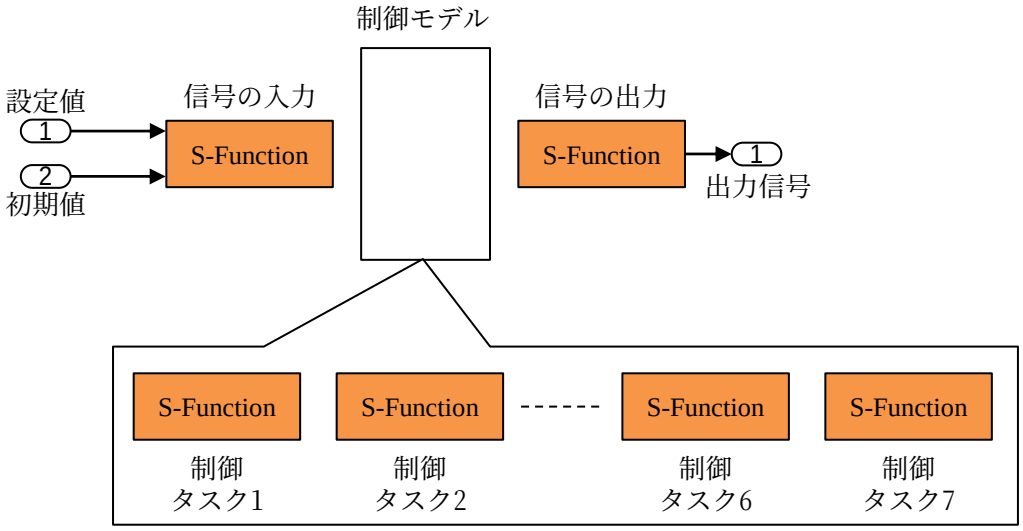


図 11　提案手法を適用した室外機モデル
Figure 11　The compressor unit model that applied the proposed method.

4. 効果検証

提案手法の効果を確認するために，実製品のソースコードへ適用する．今回は業務用空調機器を対象に取り上げる．業務用空調機器は図 9(a)に示すように，室外機および室内機がガス管，液管，伝送線で連結されている．ガス管および液管には冷媒が充填されており，冷媒の気化熱を利用して冷房および暖房機能を提供する．空調機の制御は室外機に搭載されている圧縮機などのアクチュエータの動作と室内温度や外気温度などの状態量およびその時間変化が密接に影響しあうため，制御仕様検討時点での検証が難しい．本実験では室外機の制御コントローラに搭載されている制御ソフトを対象に検証を行う．実験には表 3 に示す環境を用いた．MATLAB [注1]/Simulink [注2] への取り込みは，図 9(b)に従って実施した．ガス管や液管，外気環境をプラントモデルとしてモデル化し，室外機のソースコードをモデル上に取り込んだ．室内機は検証対象ではないため簡易的なモデルとした．

図 10 に提案手法を適用する前の室外機モデル構成，図 11 に提案手法を適用したモデル構成を示す．図 10 では，7 つある制御タスクが直列に信号線で連結されている．各制御タスクの実行時に必要なデータを信号線により転送している．一方，図 11 では，信号線が連結されている S-Function ブロックは信号の入力および出力部のみである．7 つある制御タスクの S-Function ブロックは互いに連結されていない．提案手法を用いることで S-Function ブロック間の連結を削除することが出来た．

シミュレーション時間の測定結果を表 4 に示す．冷房時では，シミュレーションに 860 秒掛かっていたが，提案手法を適用することで，280 秒で終了出来た．削減率は 67%

表 4　シミュレーション実行時間

Table 4　Simulation run time.

対象	適用前	適用後	削減率
冷房	860[sec]	280[sec]	-67%
暖房	650[sec]	90[sec]	-86%

表 5 提案手法適用工数

Table 5　The applied effort of the proposed method.

#	作業	内容	工数 (人日)
1	事前検討	方式検討など	16
2	共有ライブラリ化	対象ソース選択，インタフェース構築など	4
3	デバッグ	-	4
		合計	24

である．次に，暖房時の結果である．適用前は 650 秒，適用後は 90 秒であり，削減率は 86%であった．提案手法を用いることでシミュレーション時間を短縮することが可能であることが分かった．

表 5 に本提案手法の適用に要した工数を示す．提案手法の適用に関し，24 人日を要した．共有ライブラリの実現に関しては 9 個の S-Function ブロックを対象に実施し，4 人日で作業完了した．

5. 考察

まず，共有ライブラリの利用について述べる．従来は，制御タスク間で信号線を用いて値を入出力していた．これは，異なる S-Function ブロックから同一のソースコードを参照した場合，ソースコードにおける各変数の値は共有されないためである．よって，S-Function ブロック間で共有する値は毎周期，転送しなければならない．一方，提案手法を用いて共有ライブラリを参照した場合，異なる S-Function ブロック間で値を共有することが出来る．よって信号線による値の転送が不要となり，通信量を削減することが可能となった．しかし提案手法を用いる場合，信号の入力および出力用にそれぞれ S-Function ブロックを作成する必要がある．提案手法適用前のモデルにおいて S-Function ブロック数が少ない場合，提案手法によるシミュレーション時間の低減効果は低いと考えられる．

また，実験で取り上げたモデルでは，S-Function ブロック間で約 4000 個のデータを転送していた．これを 7 つのブロック間で転送しているため約 28000 個のデータを転送していた．一方，提案手法では，データ転送しているブロックは一つとなるため約 4000 個のデータ転送である．通信量とシミュレーション時間は比例関係にあるため，大幅に通信量を削減することができたといえる．しかし，検証対象によってブロック間のデータ通信量に差異がある．このため，ブロック間データ通信量が少ない場合には提案手法の適用による大幅な実行時間削減は得難いと推測する．

次に，提案手法の適用に要した工数について述べる．提案手法の適用には 24 人日を要した．内訳をみると，事前検討に 67%の工数を費やしており，その半数が提案手法の方式検討であった．本稿により提案手法が確立されたため，他システムへの適用時に方式検討は不要である．よって，事前検討の工数は半減される．

提案手法適用のために実施した実装は，共有ライブラリ化に伴う MATLAB[注1]/Simulink[注2] とのインタフェース部のみであった．制御ロジックに関わるソースコードの変更は無い．

以上より，提案手法を用いることで検証対象のソースコードを変更せずシミュレーションの高速化を実現可能であり，有効性を確認出来た．

シミュレーション実行時間に着目すると，冷房と暖房でシミュレーション時間の削減率に 19%の差があった．提案手法の適用対象によって得られる効果が変動すると考えられるが原因を調査中である．

6. まとめ

制御系組込みソフトウェア開発を対象に，SILS の高速化について検討した．SILS は S-Function ブロックを用いて検証対象システムのソースコードを Simulink モデル上に取り込み，シミュレーションを実現する．この時，シミュレーション速度の低下が課題である．

課題解決のため本稿では，S-Function ブロック間のデータ通信量に着目した．S-Function ブロックのシミュレーション速度はブロック間のデータ通信量の増加と共に低下することが実験により明らかとなった．ブロック間通信量削減のため，共有ライブラリを用いたモデル構成を提案した．提案手法の有効性を確認するため，業務用空調機器における室外機の制御ソフトを対象に提案手法を適用した．結果より，冷房機能に対して 67%，暖房機能に対して 86%シミュレーション時間を削減できることを確認し，本提案手法の有効性を示した．

参考文献

1) 経済産業省， 2010 年度版組込みソフトウェア産業実態調査報告書， 2010.
2) Simulink – シミュレーションおよびモデルベースデザイン (MBD) – Math Works, http://www.mathworks.co.jp/products/simulink/, 2014/01/20.
3) Hanselmann, H., Hardware-in-the-loop simulation testing and its integration into a CACSD toolset, *in Computer-Aided Control System Design, 1996. Proceedings of the 1996 IEEE International Symposium on, pp. 152-156 (1996).*
4) 森山裕，深澤健，前川正博，北村章，VirtualCRAMS(SILS)への ISS レス技術の適用，富士通テン技報，Vol.26，No.2，pp.7-pp.15 (2008/12).