マルチコア向けモデルベース開発のための モデル分割とその設計検証 ～ Model Splitter ～

2023年11月

名古屋大学大学院情報学研究科

枝廣

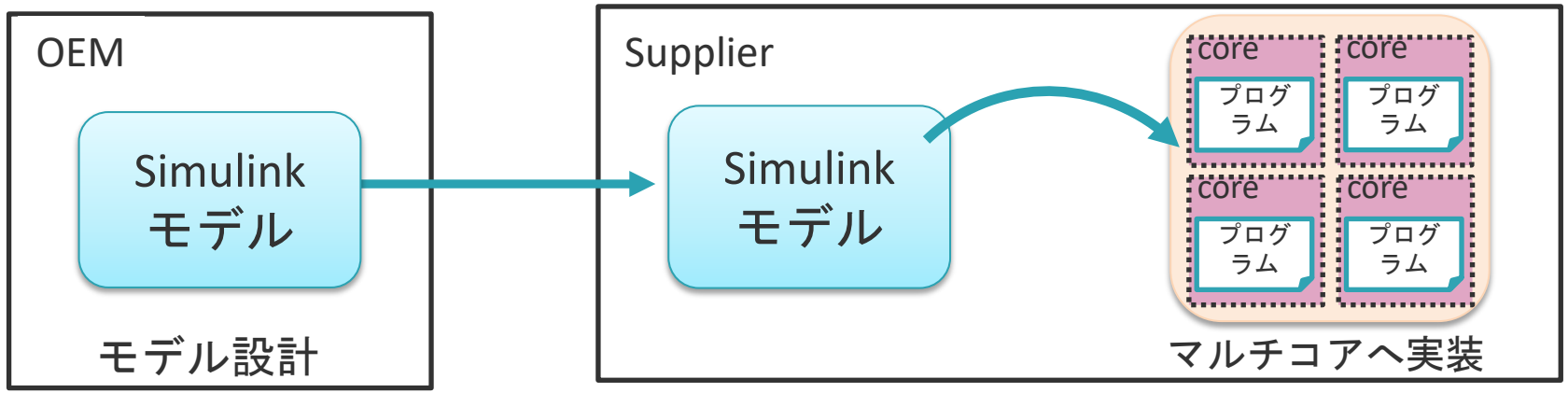
目次

- 概要

- 設計検証
  - モデル分割
  - 分割統合検証（単体・統合テスト）
  - 性能検証

動機

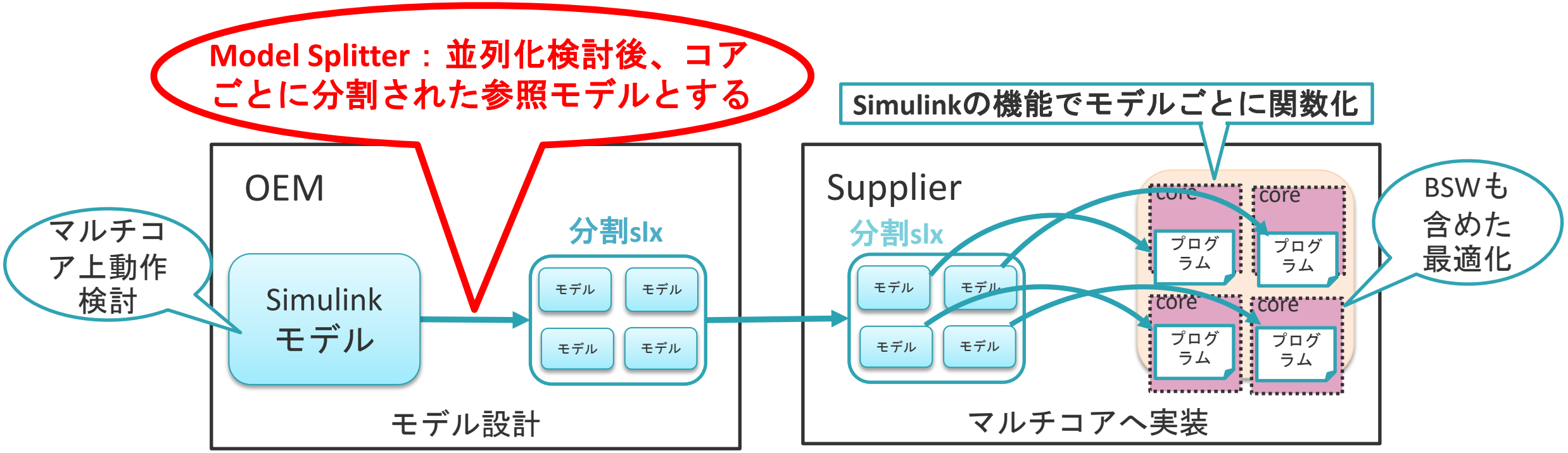
- マルチコアが前提の設計において、OEM-サプライヤ間のインターフェースは従来のSimulinkモデルでよいのか？
  - OEMとサプライヤでマルチコア上での動作状況等を共有できないと、開発における意思疎通が難しい
  - 単なるSimulinkではなく何かしらの情報付加が必要



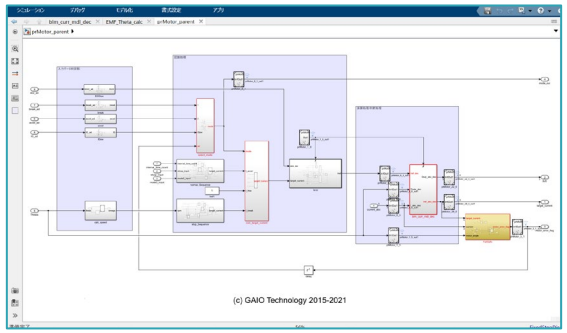
Model Splitterの提案

提案

マルチコア上実装を考慮した分割Simulinkモデル（以下、分割slx）も合わせて受け渡す



Model Splitterにより得られること



SLX

eSOL

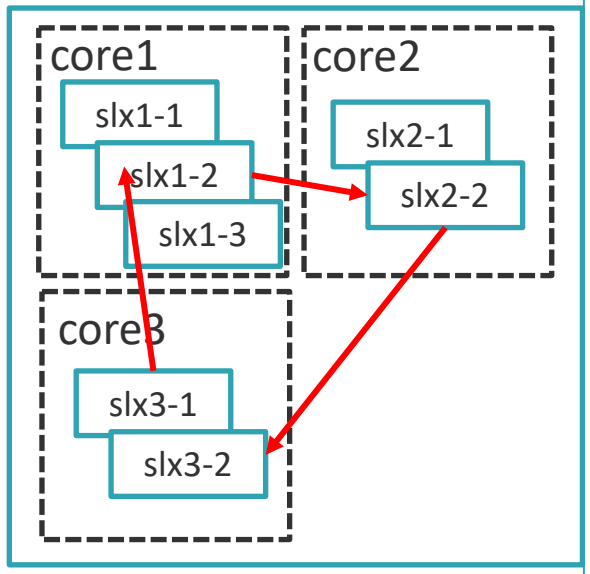
MBP

GAIO TECHNOLOGY

Model Splitter

（注：単純に分割・参照化すると、参照化によるブラックボックス化のためデッドロック、代数ループ等が発生。例：右上図で各コアー一つのモデルにするとループ発生）

分割SLX

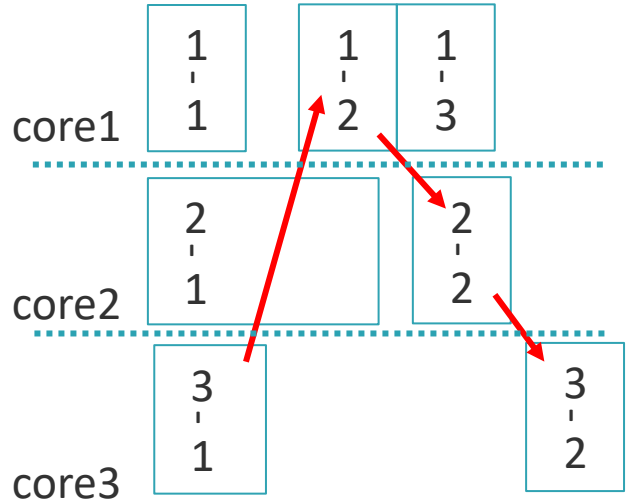


(Embedded Coder, TargetLink)

分割モデルごとの関数化

分割モデル(関数)単位性能見積

GAIO TECHNOLOGY



性能検証

分割モデル(関数)単位・統合モデルテスト生成

GAIO TECHNOLOGY

単体・統合テスト

Model Splitterのメリット

- OEMとサプライヤの間で、(並列)実行可能、可視化可能な仕様書としてのモデル
  →Simulink（とTargetLink）のみで機能検証、関数化コード生成などができる点において「slxと分割slx」によるインターフェースが有利

	モデル上での並列可視化	並列化後のモデル上検証（機能要件）	並列化コード生成	並列化コード検証（機能・非機能要件）	備考（OEMとサプライヤが両者で並列動作確認するために）
slxのみ（現状）	×（並列情報なし）	×（並列情報なし）	OSCARなどを利用	○（ツール利用）	両者で同じ並列化コード生成ツール(OSCAR等)を保有する必要あり
slxとBLXML (MBP内部データ)	○（スクリプトによる色づけ）	×	eMBPなどを利用	○（ツール利用）	両者で同一ツール(eMBP)を保有している必要あり
slxと分割slx（分割とコアの対応表含む）	○（スクリプトによる色づけ）	○（sdi等のSimulink機能が利用可能）	関数化まではEmbedded CoderやTargetLinkで可能	◎（ツール利用。◎は左記理由による）	関数化コードから並列実行コード生成ツールの共通化

GAIO TECHNOLOGY Co., Ltd. & NAGOYA Univ. Confidential

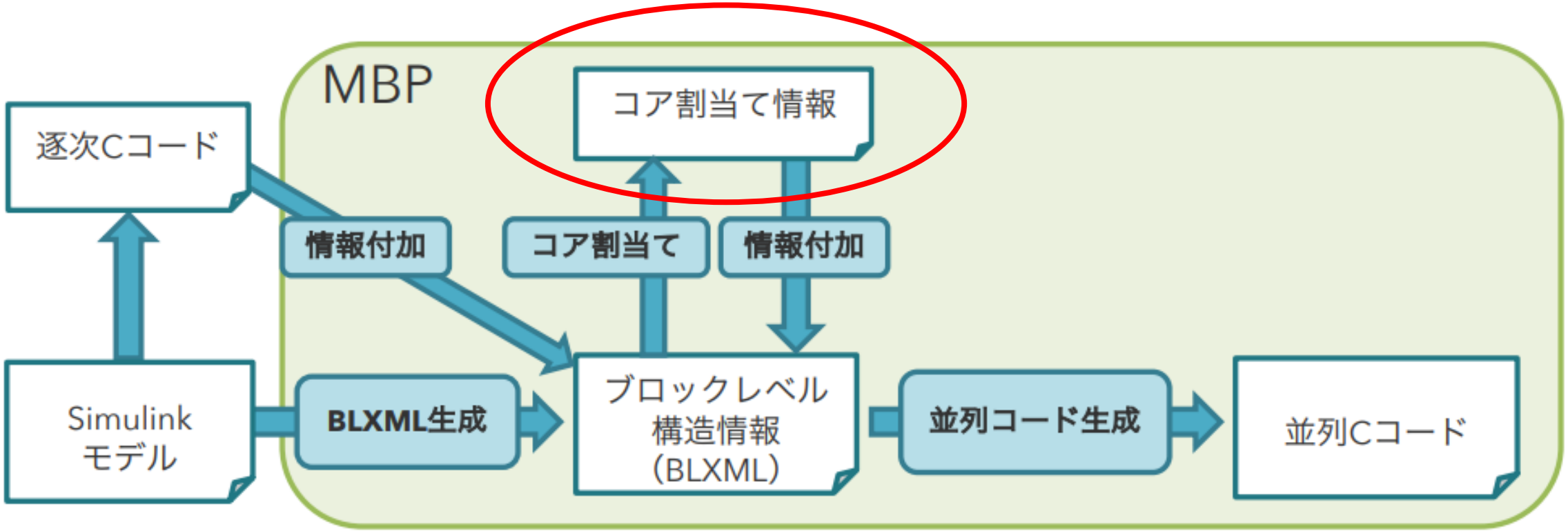
目次

- 概要

- 設計検証
  - モデル分割（Model Splitter）
  - 分割統合検証（単体・統合テスト）
  - 性能検証

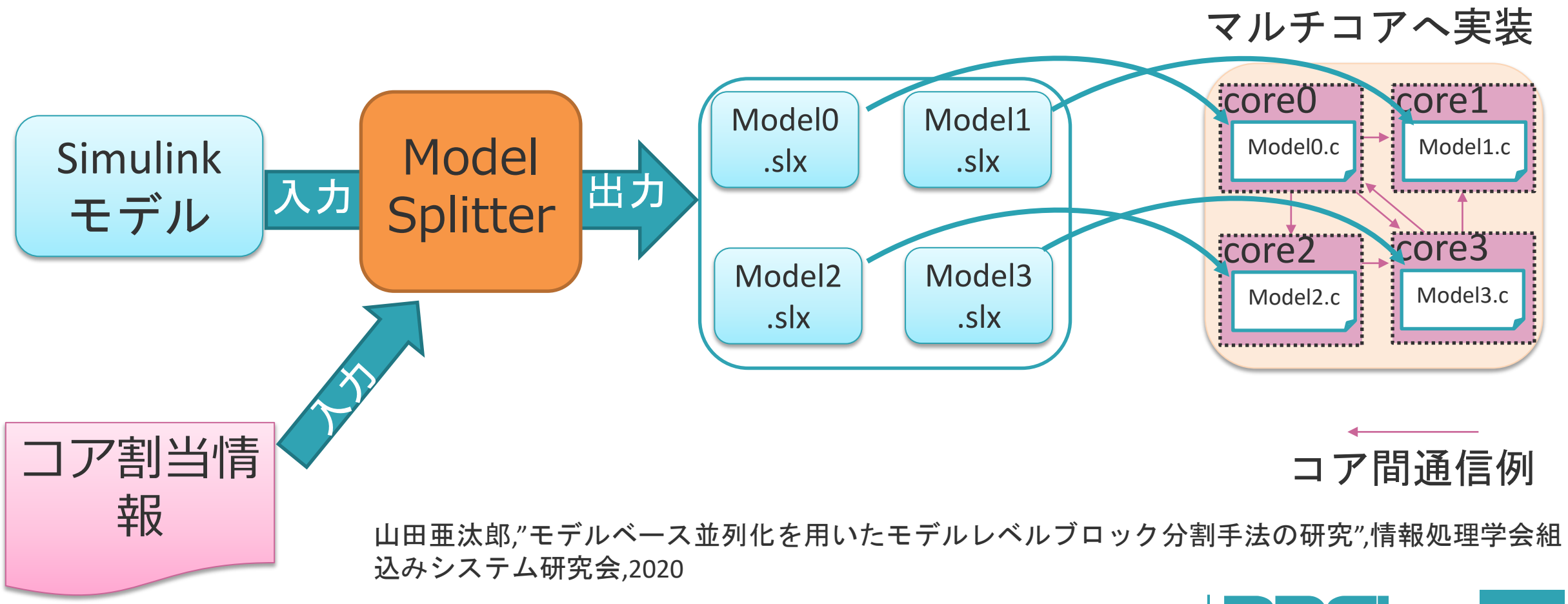
モデルベース並列化 (MBP)

- MBP(Model Based Parallelizor)
  - Simulinkモデルと逐次コードを用いて並列化



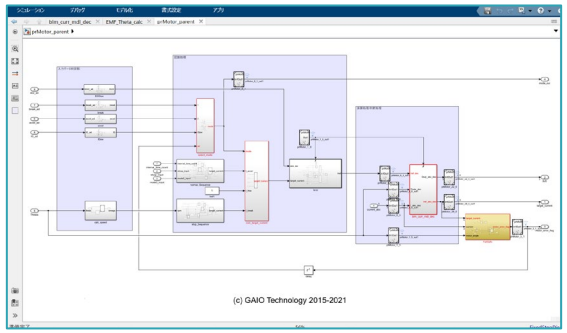
Model Splitter

- MBPが出力したコア割当情報をもとにモデルを分割



山田亜汰郎,”モデルベース並列化を用いたモデルレベルブロック分割手法の研究”,情報処理学会組込みシステム研究会,2020

分割の課題



SLX

MBP

Model Splitter

分割SLX

core1
slx1-1
slx1-2
slx1-3

core2
slx2-1
slx2-2

core3
slx3-1
slx3-2

(Embedded Coder, TargetLink)

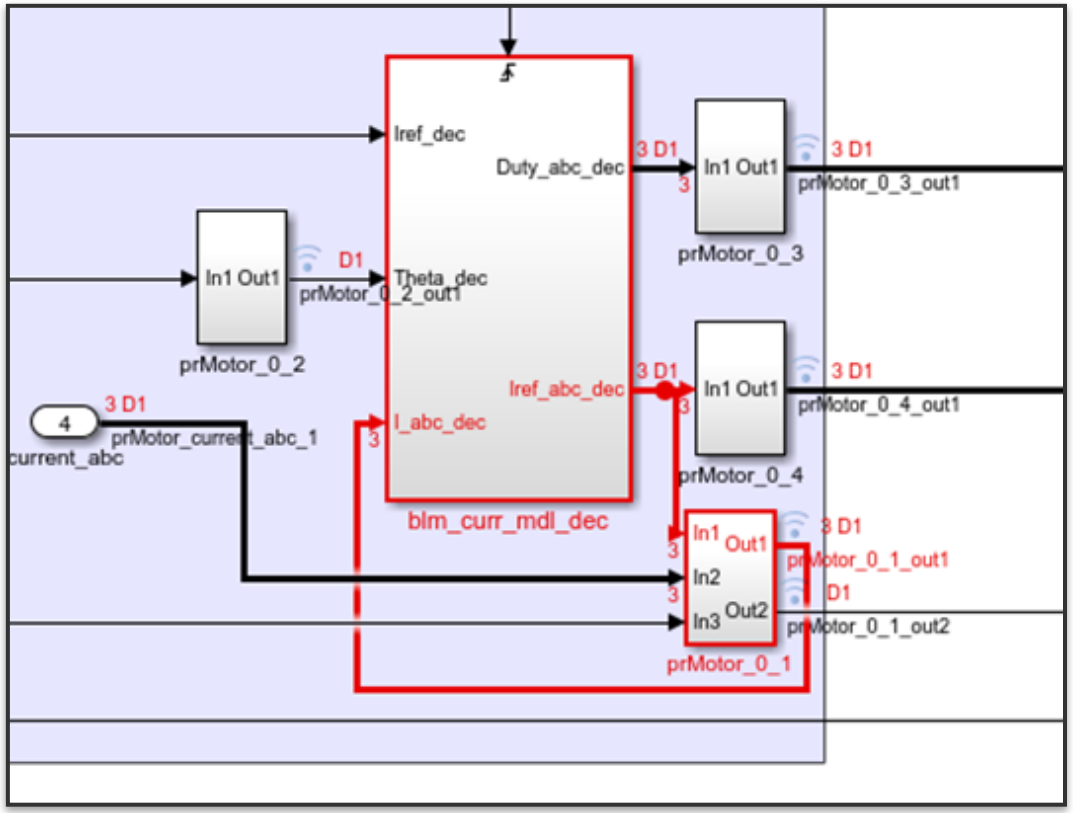
分割モデルごとの関数化

単純に分割・参照化すると、参照化によるブラックボックス化のためデッドロック、代数ループ等が発生。
例：右図で各コア一つのモデルにするとループ発生

代数ループとは

- 同一タイムステップ内でブロックの出力と入力の間に循環依存関係が生じること

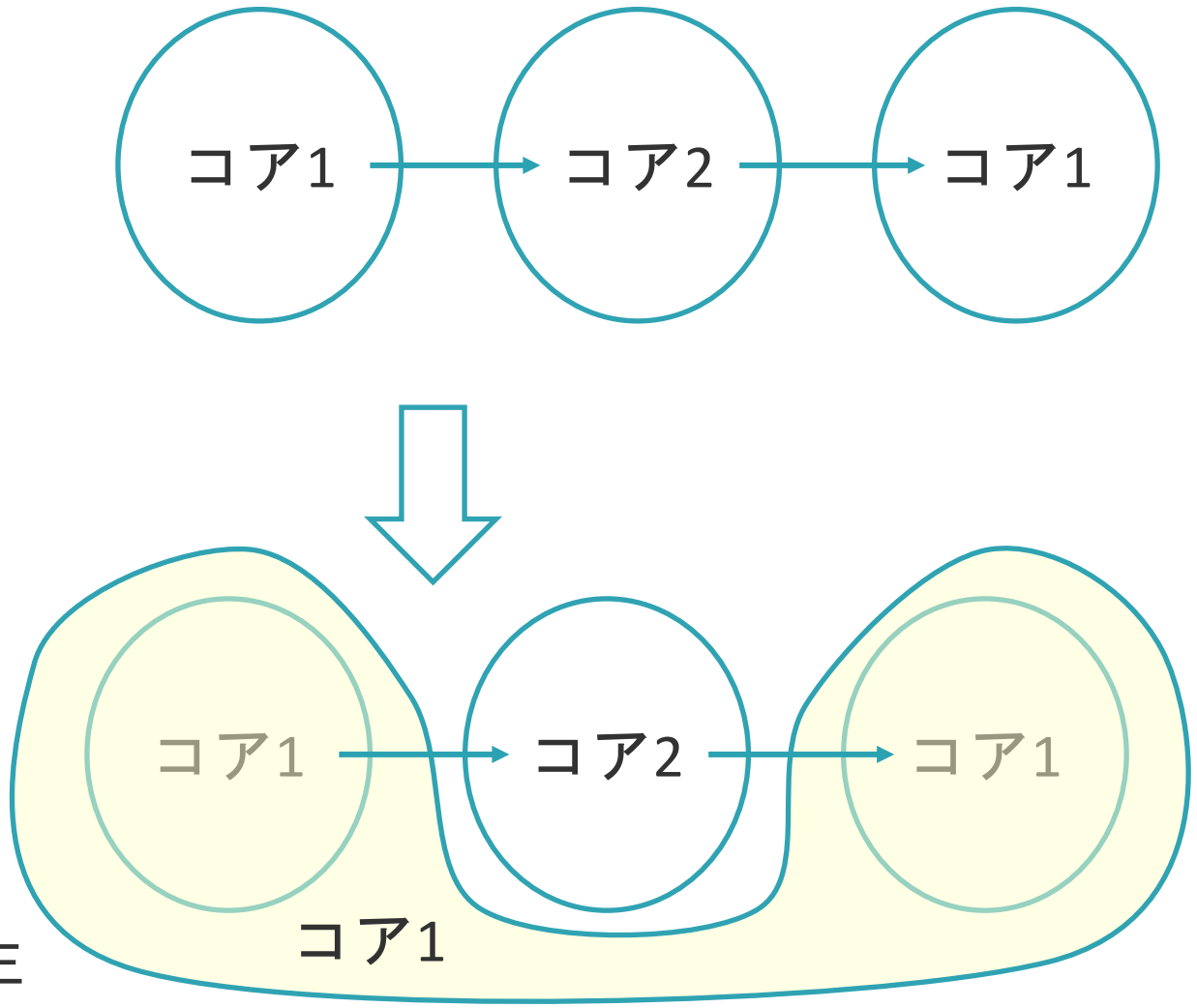
例)
右の図において, 赤線部分で
出力と入力がループしている
(遅延ブロックなし)



代数ループにならない分割アルゴリズム

1. 連結成分による分割
   - 右図

2. 再収れん箇所で分割

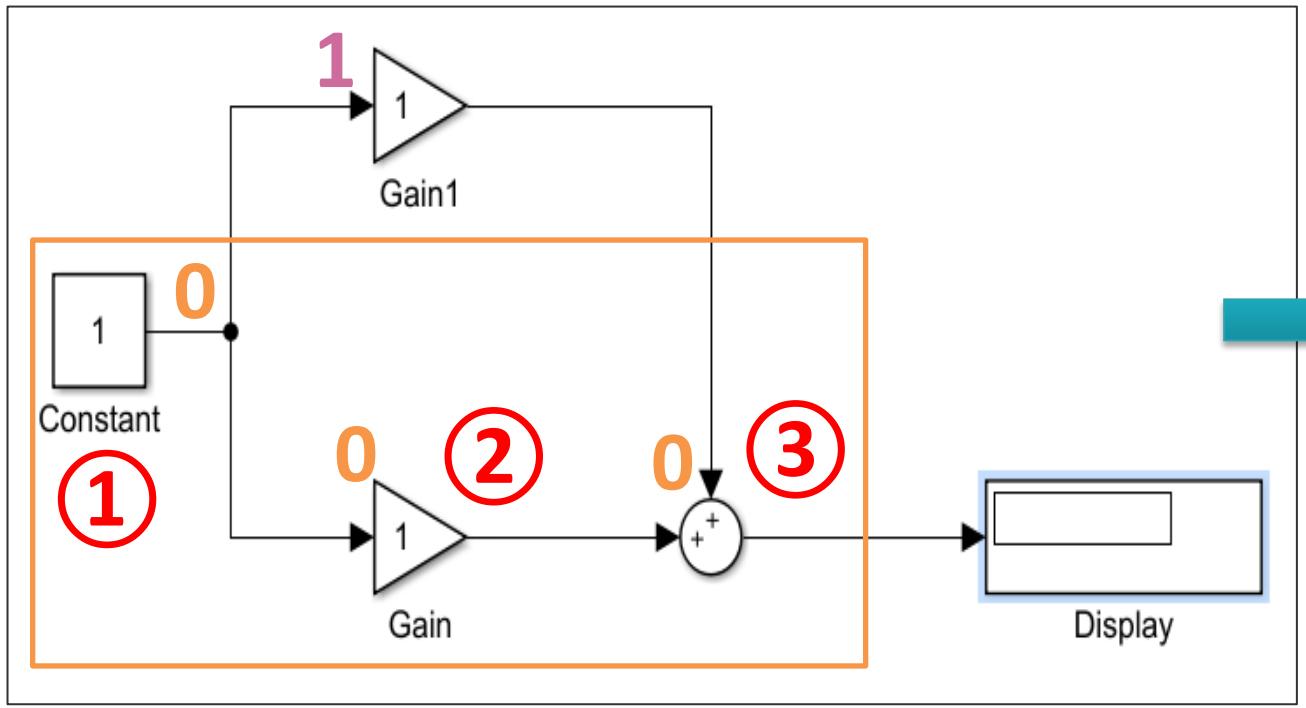


非連結成分を
統合すると
代数ループ発生

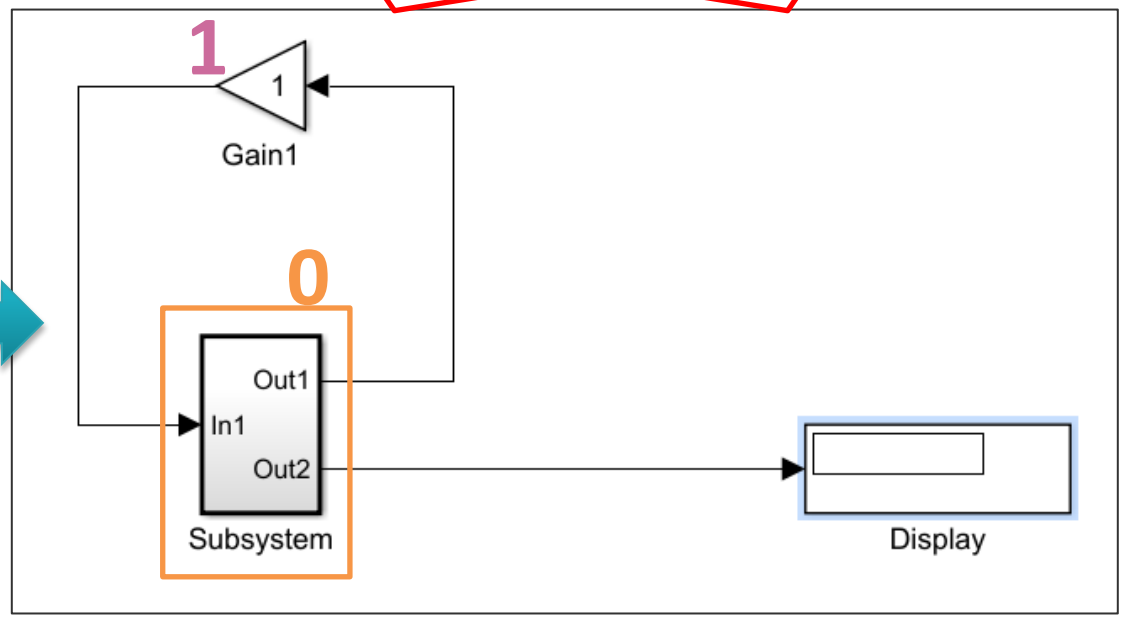
連結成分による分割の課題

: サブシステムとし割当

0,1: コア番号

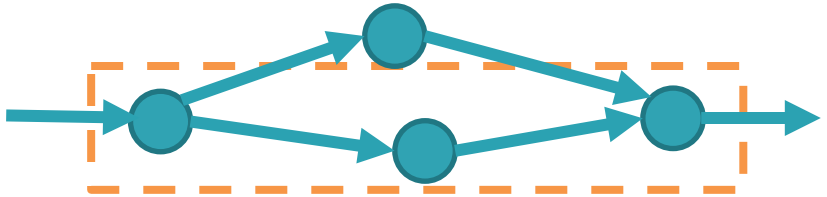


代数ループ
発生!!

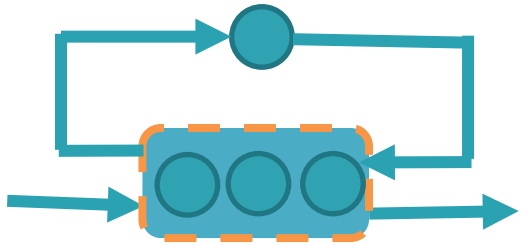


※見やすさのためGain1を反転

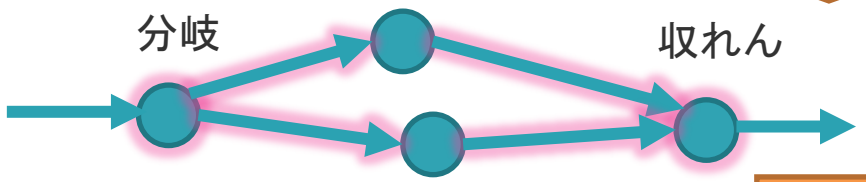
代数ループ発生ケース



元のグラフにループはないが、
無向グラフとして見るとループがある



連結部を縮退した際にループが発生



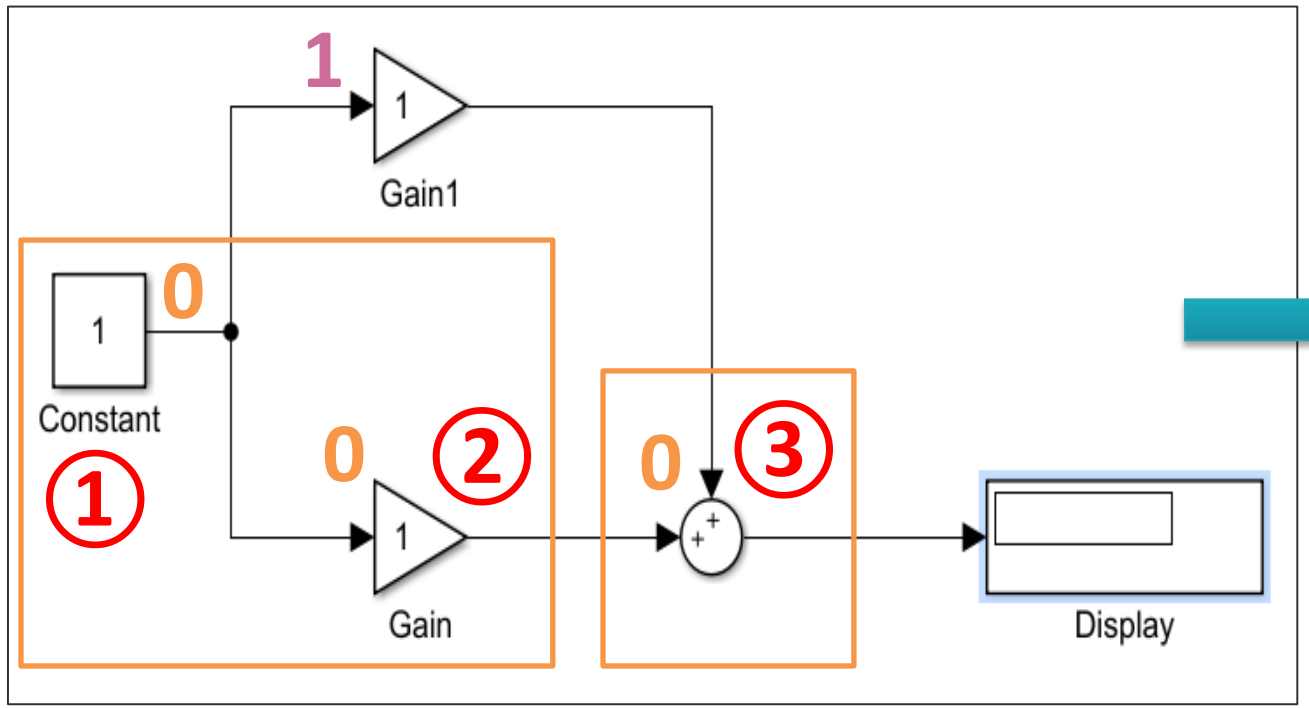
元のグラフで再収れん構造がある

再収れん構造を発見し代数ループを解消する.

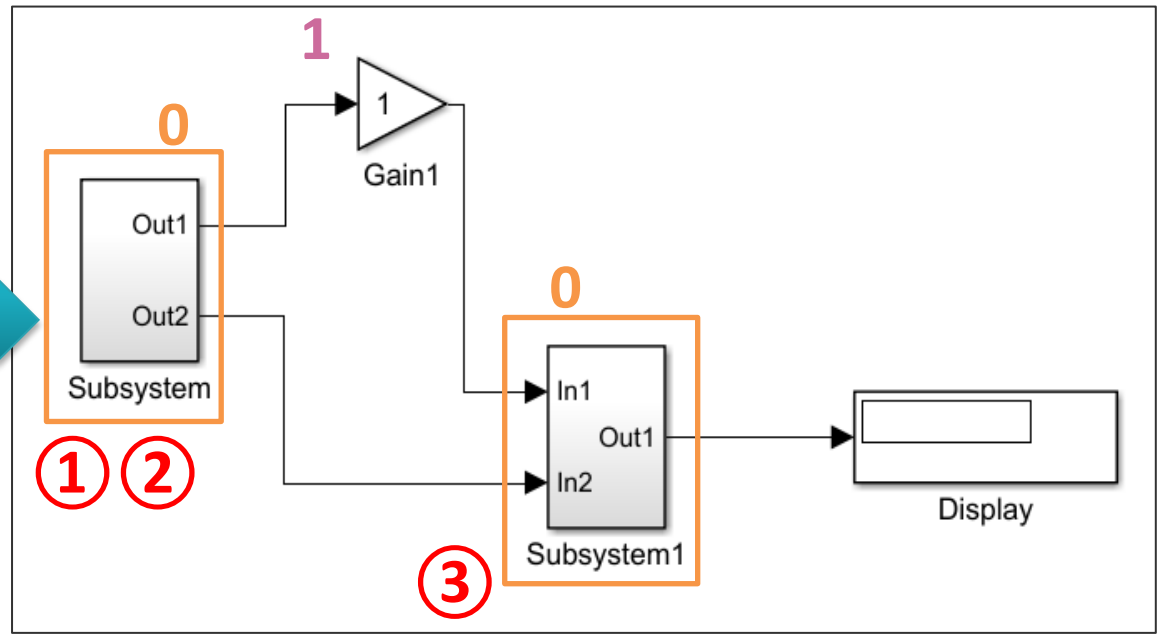
再収れん箇所で分割

: サブシステムとし割当

0,1: コア番号

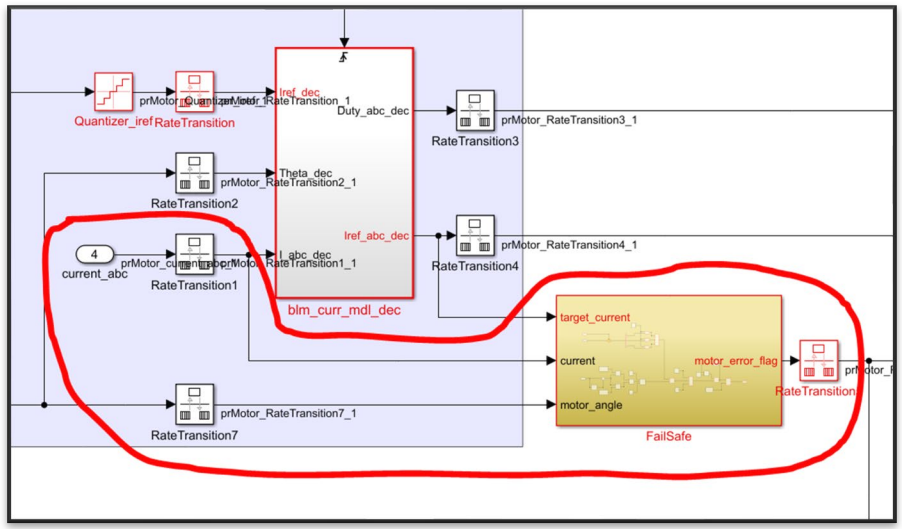


正常に分割



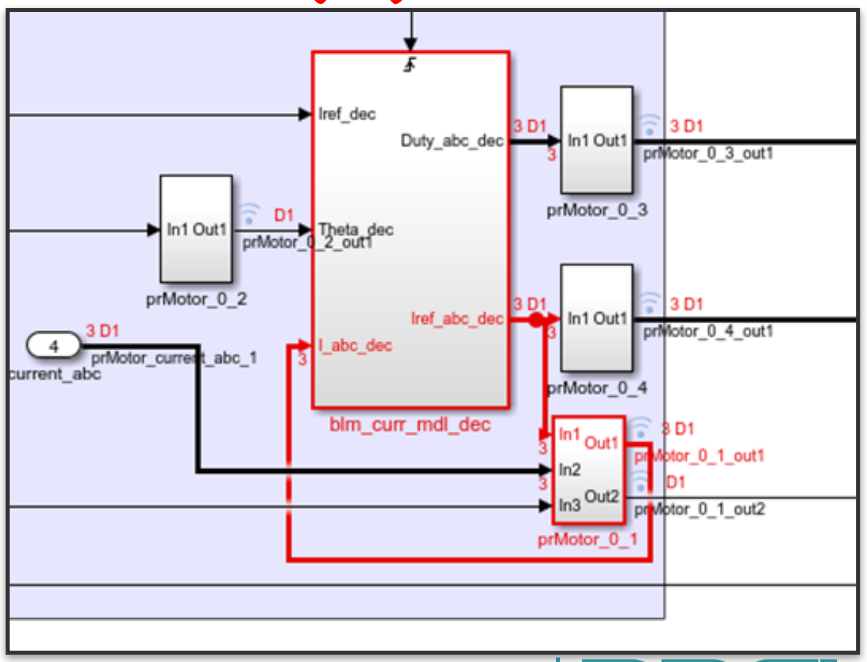
実際の例

連結成分のみ・・・元のモデルにはなかった代数ループ発生



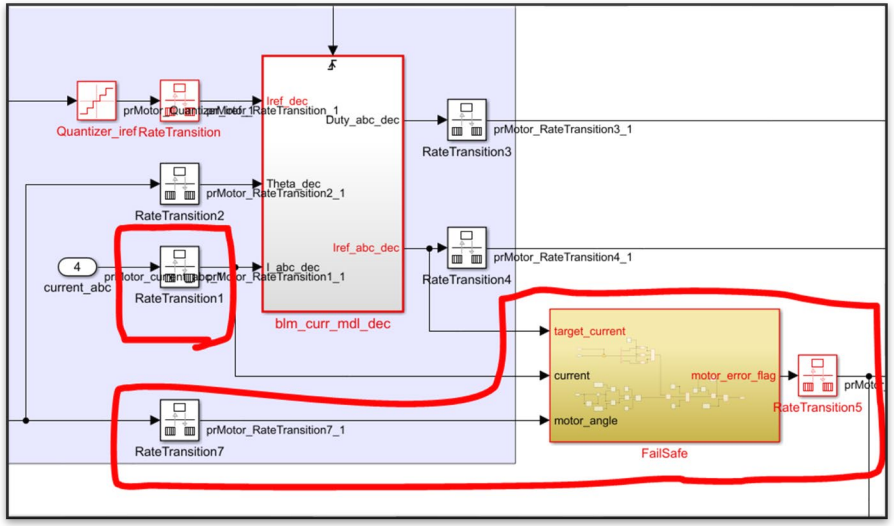
分割後

代数ループ発生



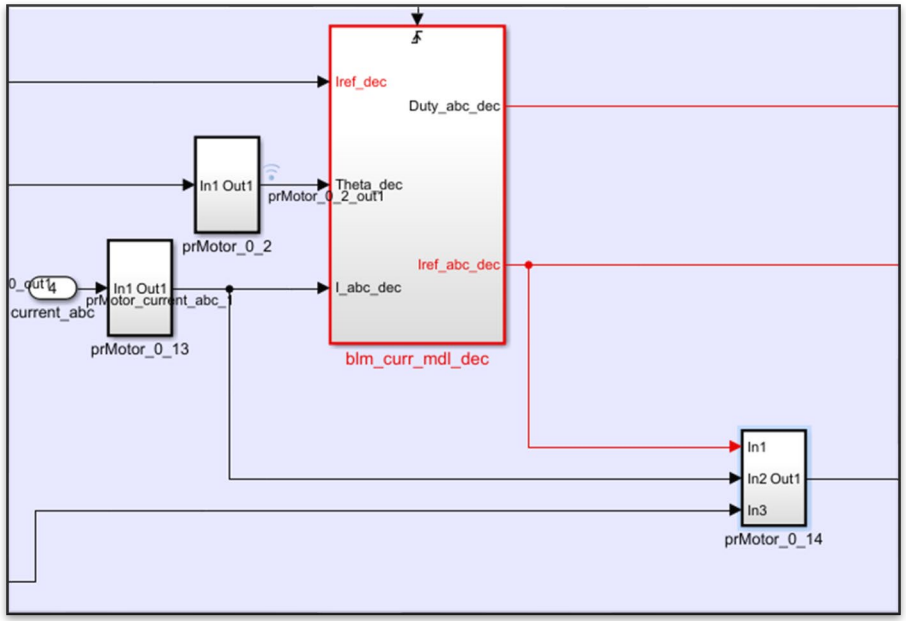
実際の例

再収れん箇所で分割・・・正常に分割



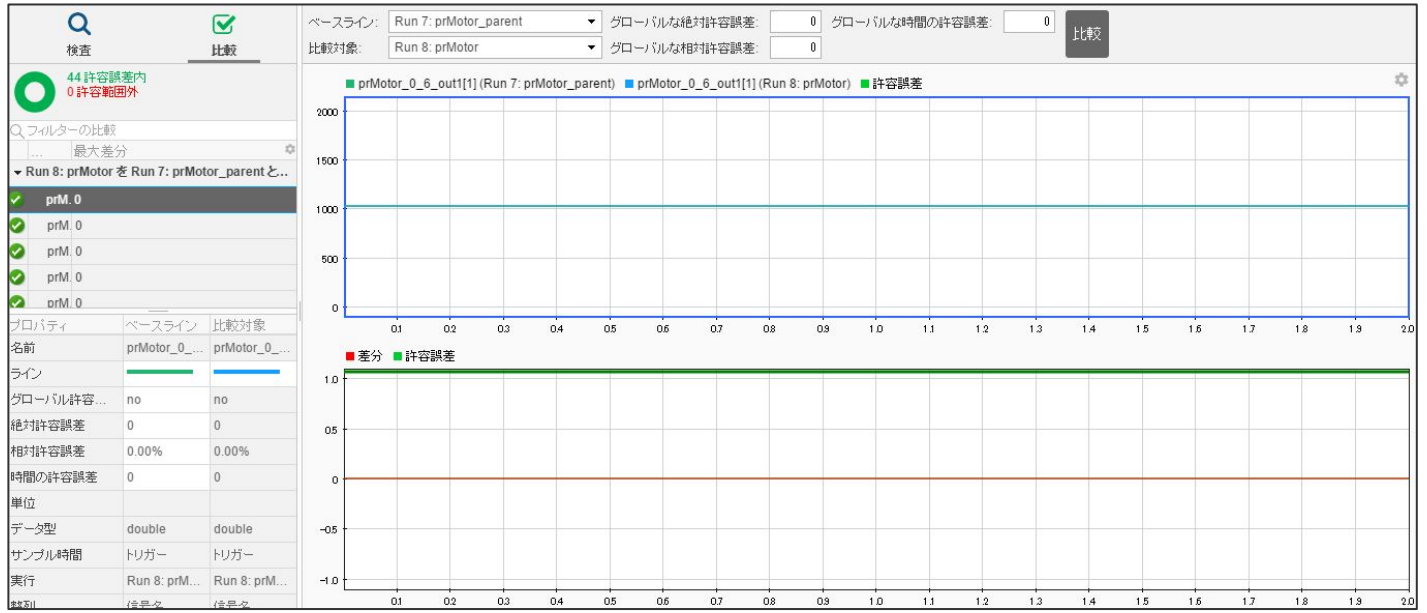
分割後

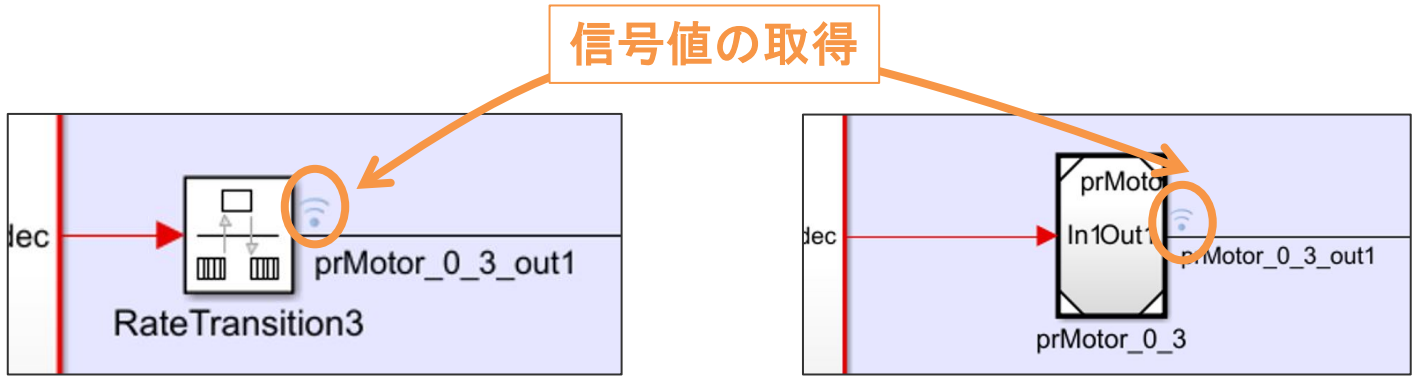
代数ループ回避



実際の例

[シミュレーション データ インスペクターによる比較結果]





分割前モデル

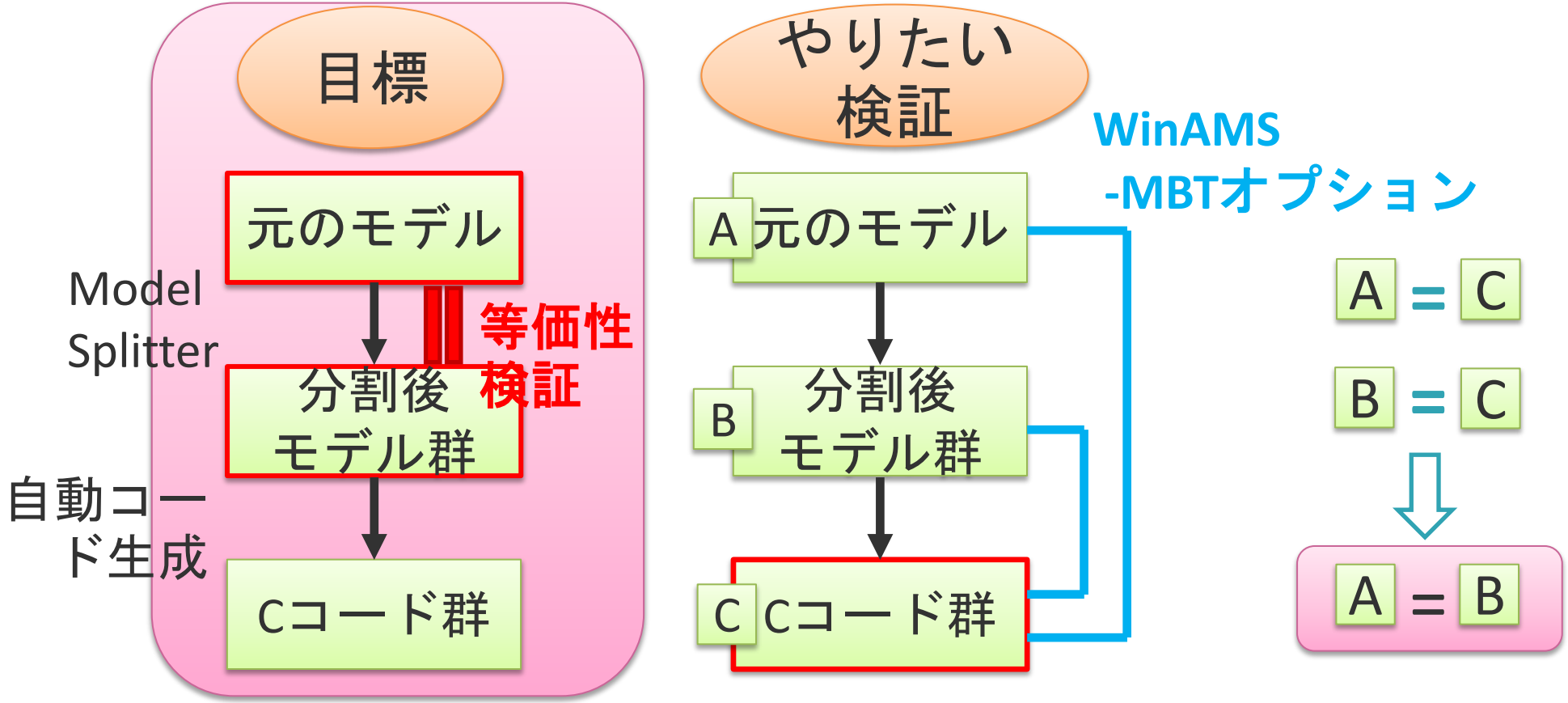
分割後モデル

目次

- 概要

- 設計検証
  - モデル分割
  - 分割統合検証（単体・統合テスト）
  - 性能検証

検証方法



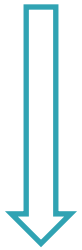
各モデルとコードが一致することを示すことで
モデルの等価性を証明する

使用ツール (いずれもガイオ社製ツール)

- CasePlayer2

ソースコードを基に
仕様書作成/静的解析を自動実行

- カバレッジマスター



CasePlayer2の結果を基に
ソースコードに対し単体テストを実行

- カバレッジマスターWinAMS -MBTオプション

以後、「MBTオプション」と呼称

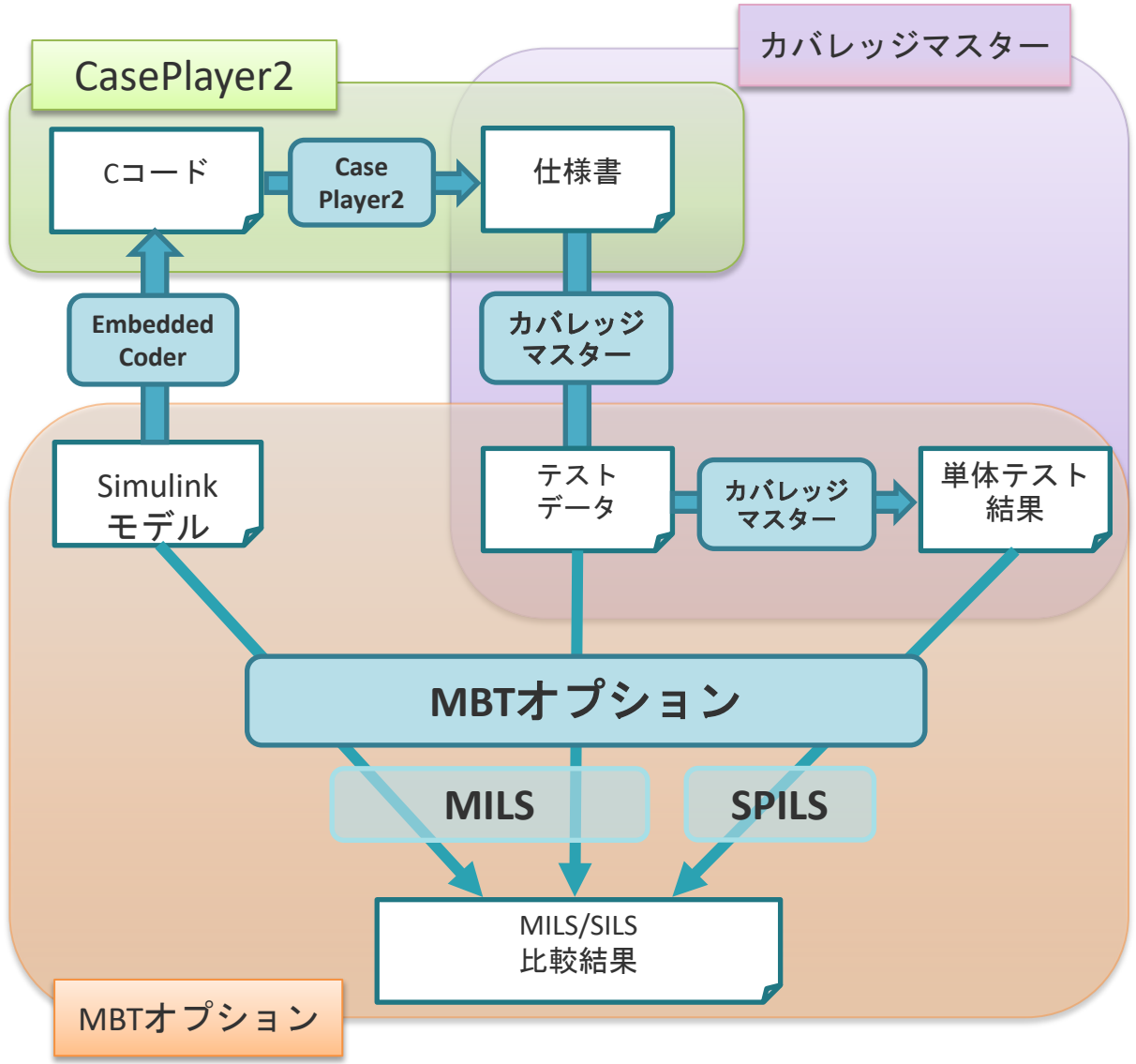
MATLAB/Simulink上でMILS実行/SPILS実行。
その結果を比較検証。
このツールはMATLABのコマンドライン上で実行する。

各ツールを用いた検証概要

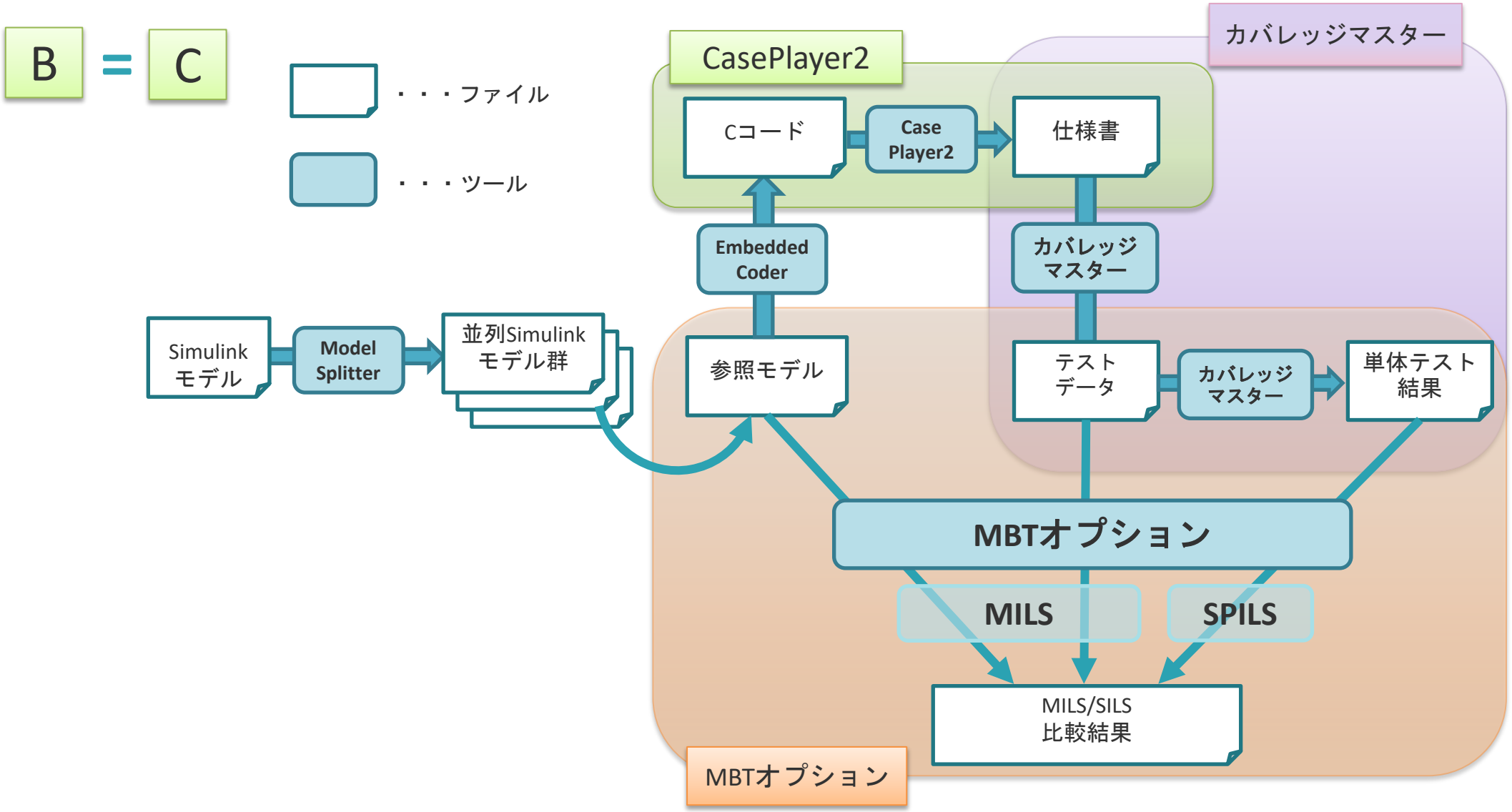
A = C

・・・ファイル

・・・ツール



各ツールを用いた検証概要



CasePlayer2での実行結果

モジュール仕様書の例

モジュール仕様書

名称	prMotor_0_7_step	種別	記憶クラス
		Normal	extern

定義位置	prMotor_0_7.c　52行目

概要説明
Model step function

戻り値	
型	コメント
void	

参照している外部変数一覧

変数名	定義ファイル名	行番号
rtDW	prMotor_0_7.c	25
rtU	prMotor_0_7.c	28

代入している外部変数一覧

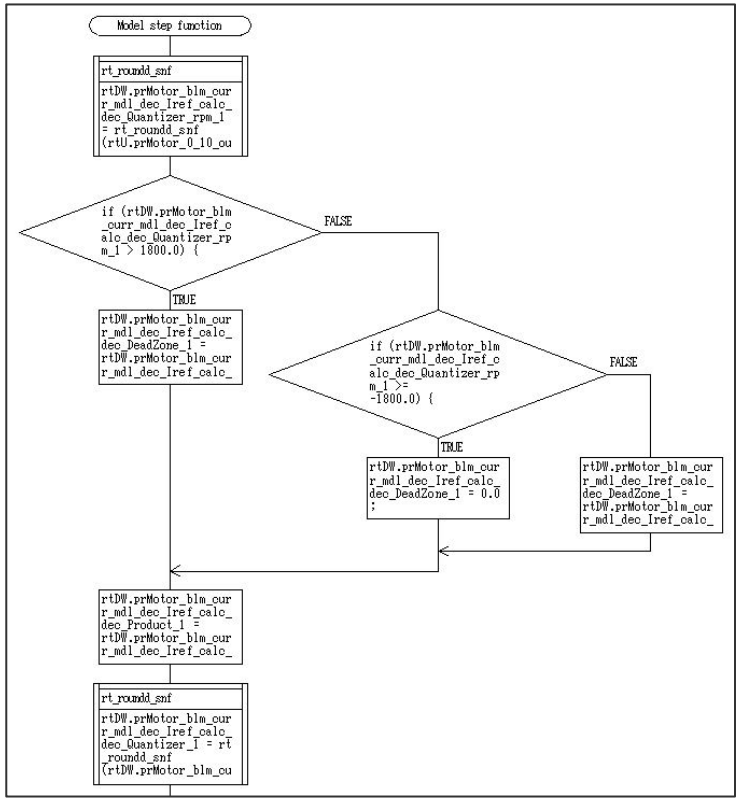
変数名	定義ファイル名	行番号
rtDW	prMotor_0_7.c	25
rtY	prMotor_0_7.c	31

呼び出し関数一覧

関数名	定義ファイル名	行番号
rt_roundd_snf	prMotor_0_7.c	33

補足説明

フローチャートの例

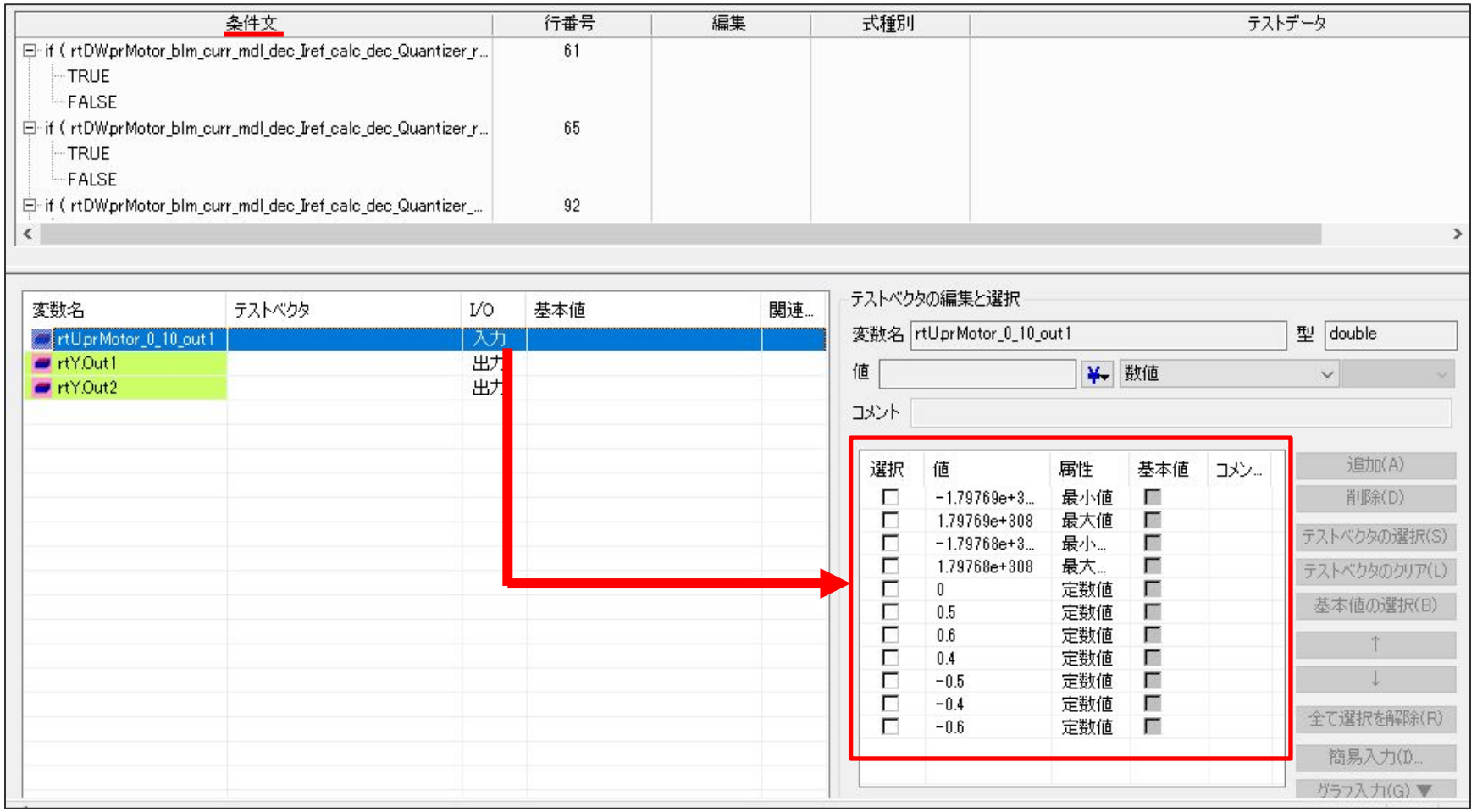


上記以外にも以下のような仕様書が出力される

- 構造体仕様書
- #define一覧
- インクルード関係図　…etc

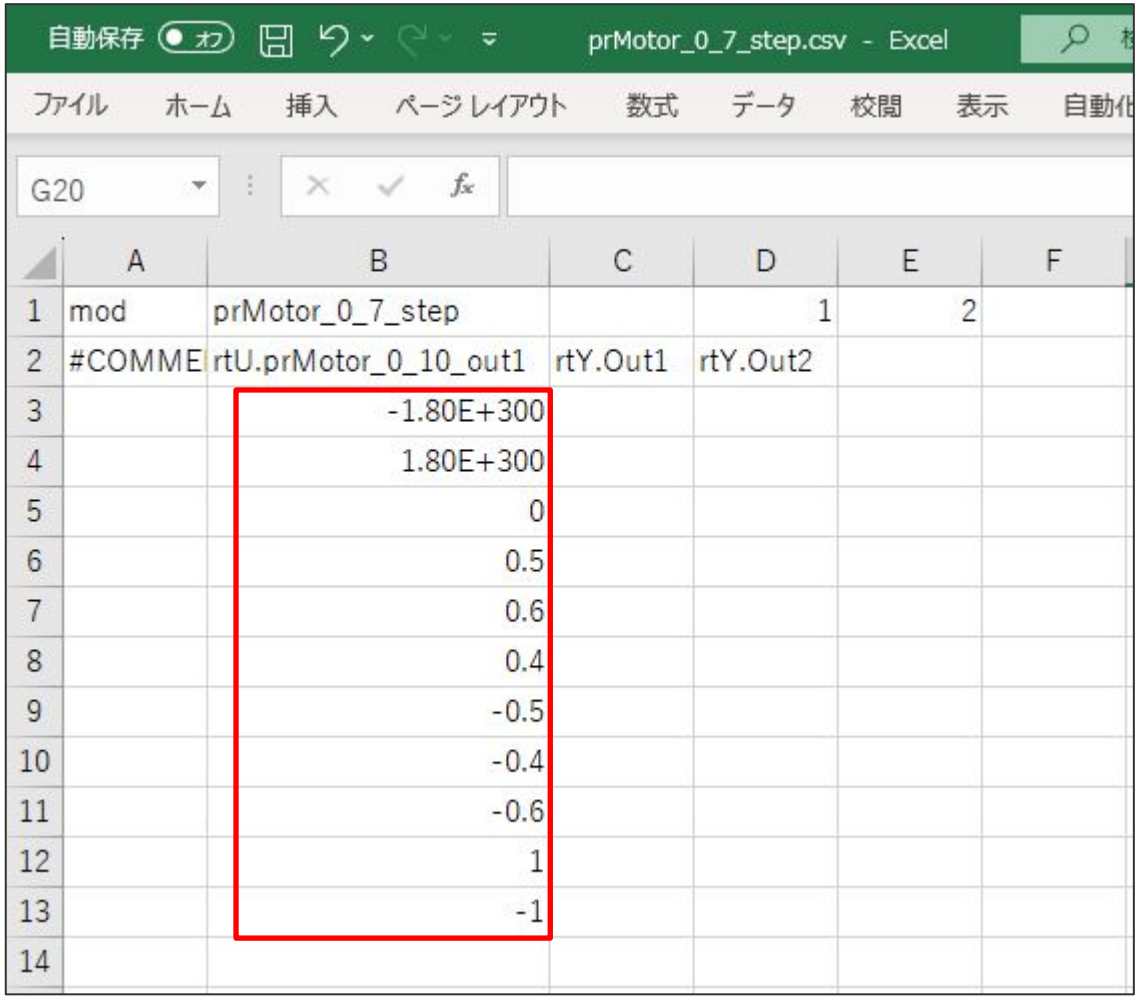
カバレッジマスターでの実行内容(1/3)

[特徴] 条件式を網羅できるようテストデータ値候補を自動算出



カバレッジマスターでの実行内容(2/3)

- 先ほど自動算出されたテストデータから入力データCSVを生成



	A	B	C	D	E	F
1	mod	prMotor_0_7_step		1	2	
2	#COMME	rtU.prMotor_0_10_out1	rtY.Out1	rtY.Out2		
3		-1.80E+300				
4		1.80E+300				
5		0				
6		0.5				
7		0.6				
8		0.4				
9		-0.5				
10		-0.4				
11		-0.6				
12		1				
13		-1				
14						

カバレッジマスターでの実行内容(3/3)

- 入力データCSVをCコードに入力し単体テストを行なった結果

SPILS結果レポート

テスト結果報告書

テスト全体の情報

トップのCSVファイル	C:¥Users¥imaip¥tools¥CoverageMaster¥prMotor_0_7¥TestCsv¥prMotor_0_7_step.csv
テストタイトル	
全体のC0網羅率	100%
全体のC1網羅率	100%
テスト日時	2023/08/09 18:08:44
全体の合否	No Check
テストCSVファイル数	1
総テストベクタ数	11
スタートアップコマンドファイル	C:¥Users¥imaip¥tools¥MBT¥prMotor_0_7¥Work_0_7¥sil¥SS_STARTUP.txt

テスト結果情報

CSVファイル	種別	テストタイトル	関数名	実行された関数	CALLしたスタブ関数/元の関数	テストベクタ数	テスト日時	実行時間	仮想時間	合否
prMotor_0_7_step.csv	mod		prMotor_0_7_step	rt_roundd_snf		11	2023/08/09 18:08:56	141 ms	11.15 ms	No Check

カバレッジ情報

関数	C0網羅率	C1網羅率	カバレッジログファイル
prMotor_0_7_step	100%	100%	TestCoverLog¥prMotor_0_7.c¥prMotor_0_7_step&0001.txt
rt_roundd_snf	100%	100%	TestCoverLog¥prMotor_0_7.c¥rt_roundd_snf&0002.txt

MBTオプションでの実行内容(1/2)

MATLAB上でコマンドを実行

- MILS実行コマンド

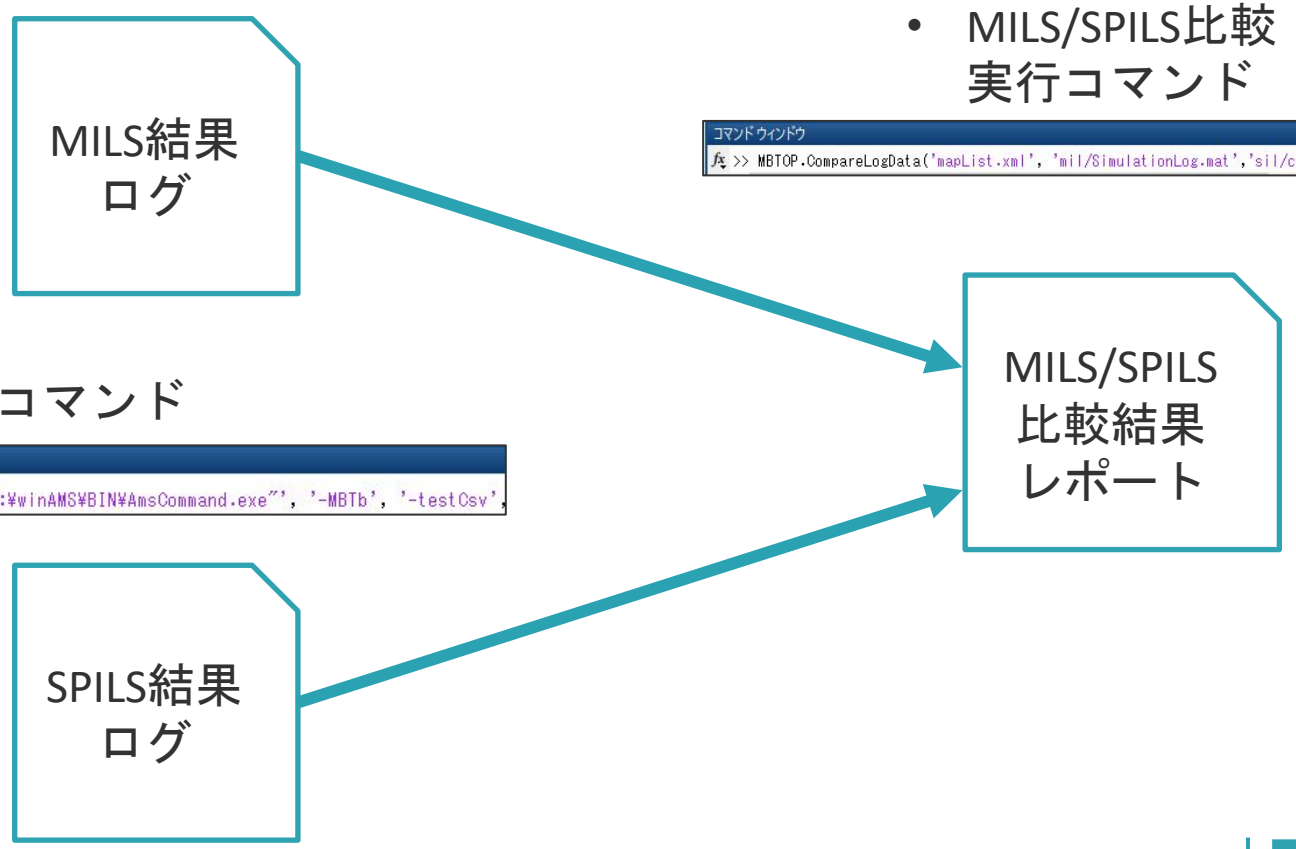
```
コマンド ウィンドウ
>> MBTOP.RunSimulation( 'prMotor_0_7.slx', 'mapList.xml','modelData.csv', 'mil');
```

- SPILS実行コマンド

```
コマンド ウィンドウ
fx >> status = system(['"C:¥winAMS¥BIN¥AmsCommand.exe"', '-MBTb', '-testCsv',
```

- MILS/SPILS比較
  実行コマンド

```
コマンド ウィンドウ
fx >> MBTOP.CompareLogData('mapList.xml', 'mil/SimulationLog.mat','sil/codeData.csv', 'errorResult.mat');
```



MBTオプションでの実行内容(2/2)

- MILS/SPILS実行比較結果レポート例
  （例：出力信号Out1）

信号データ

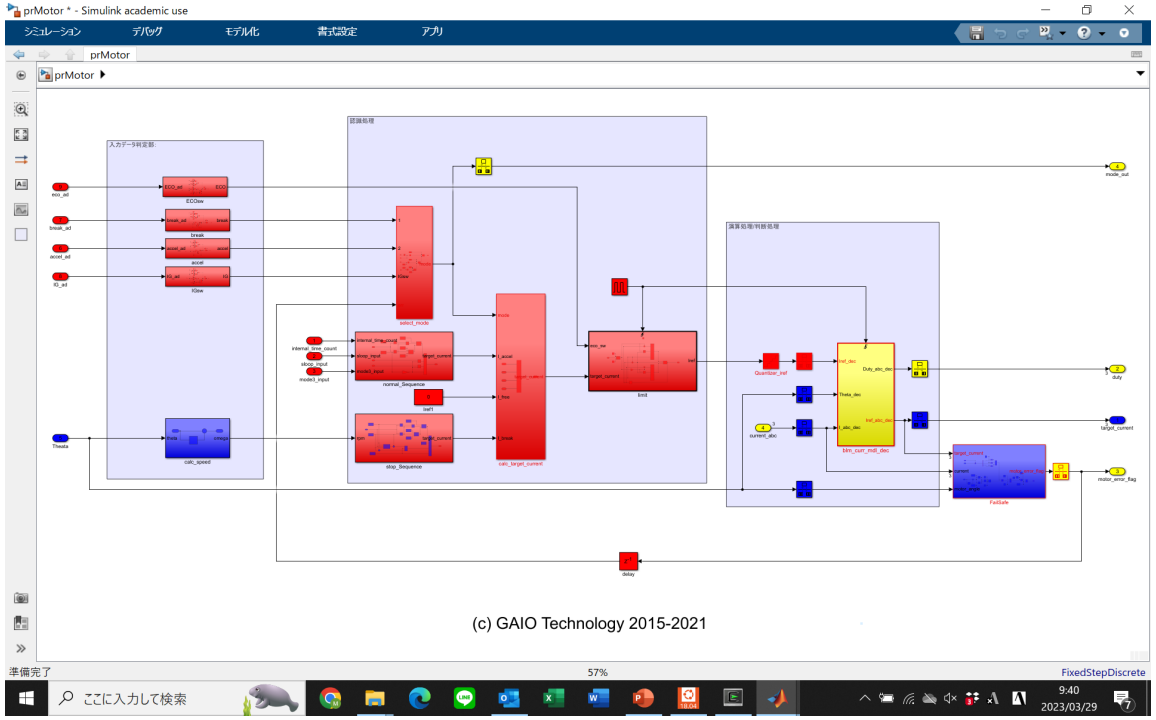
時間	モデルデータ	コードデータ	差分	誤差
0	-6.1523437500000004E+299	-6.1523437500000004E+299	0	0
0.20000000000000001	6.1523437500000004E+299	6.1523437500000004E+299	0	0
0.40000000000000002	0	0	0	0
0.60000000000000009	0	0	0	0
0.80000000000000004	0	0	0	0
1	0	0	0	0
1.2000000000000002	0	0	0	0
1.4000000000000001	0	0	0	0
1.6000000000000001	0	0	0	0
1.8	0	0	0	0
2	0	0	0	0

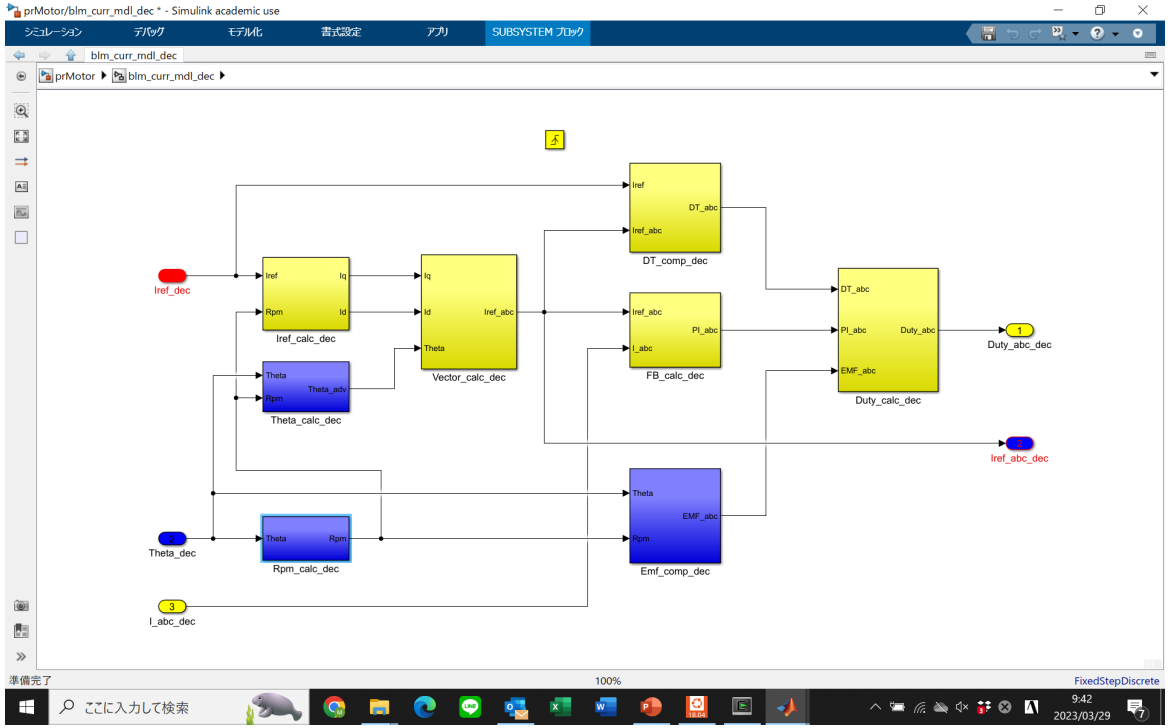
＝振る舞いに差がない

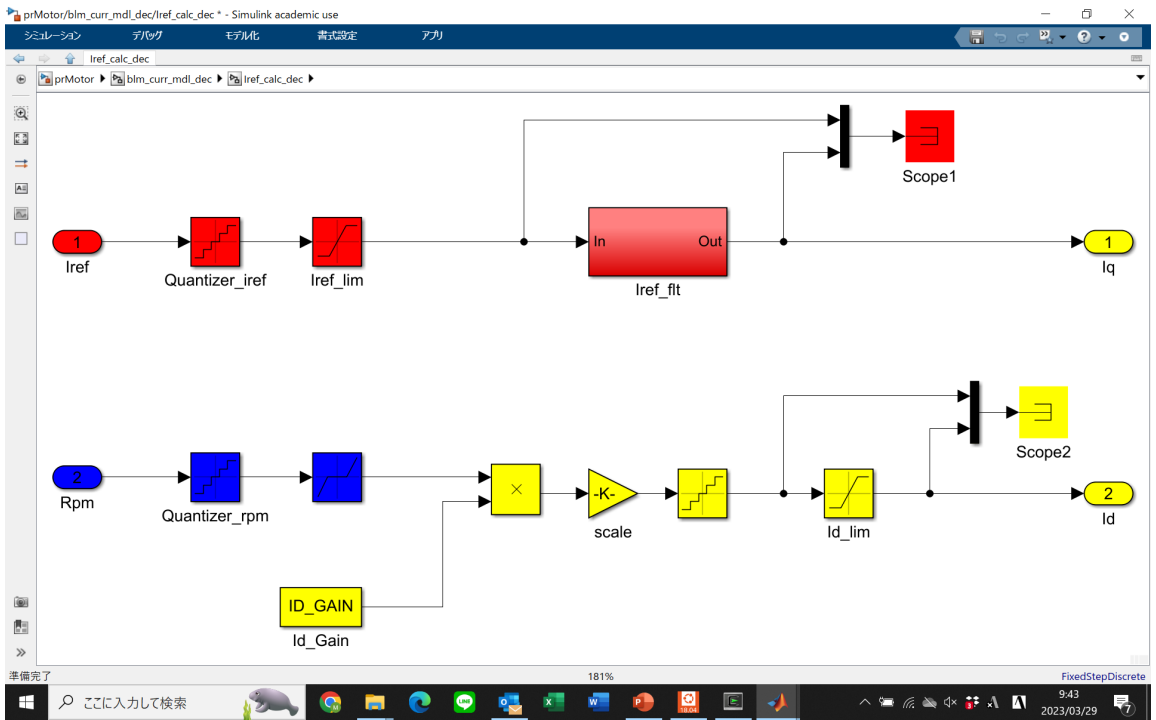
目次

- 概要

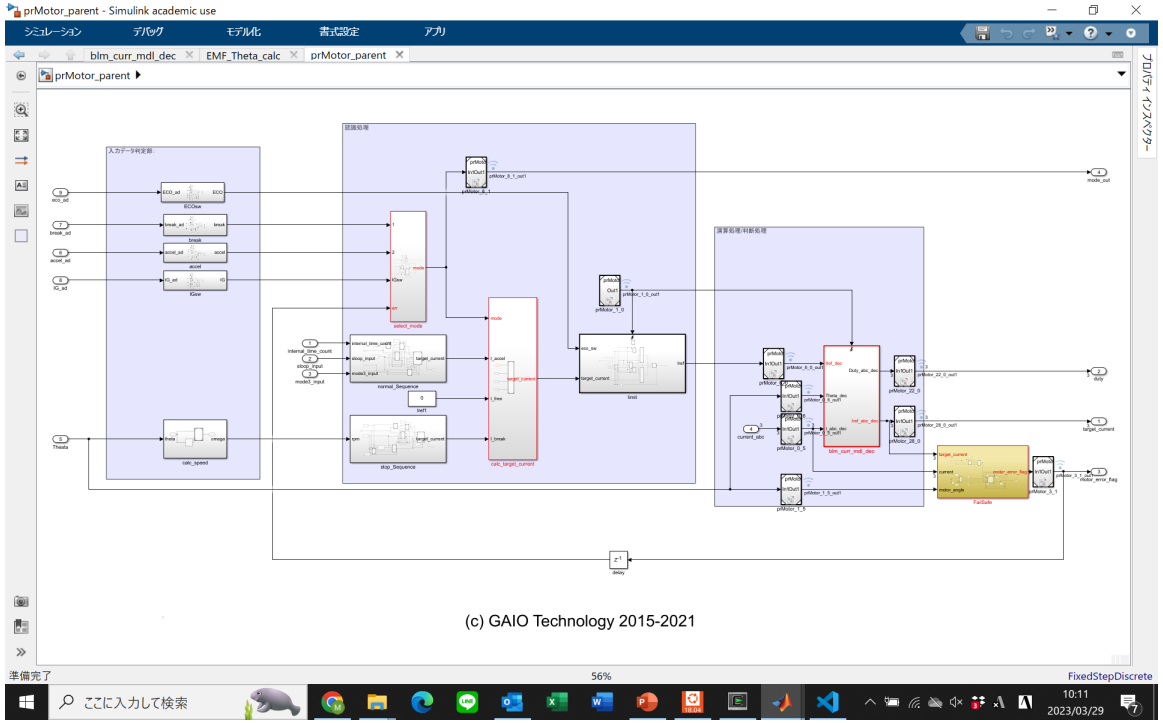
- 設計検証
  - モデル分割
  - 分割統合検証（単体・統合テスト）
  - 性能検証

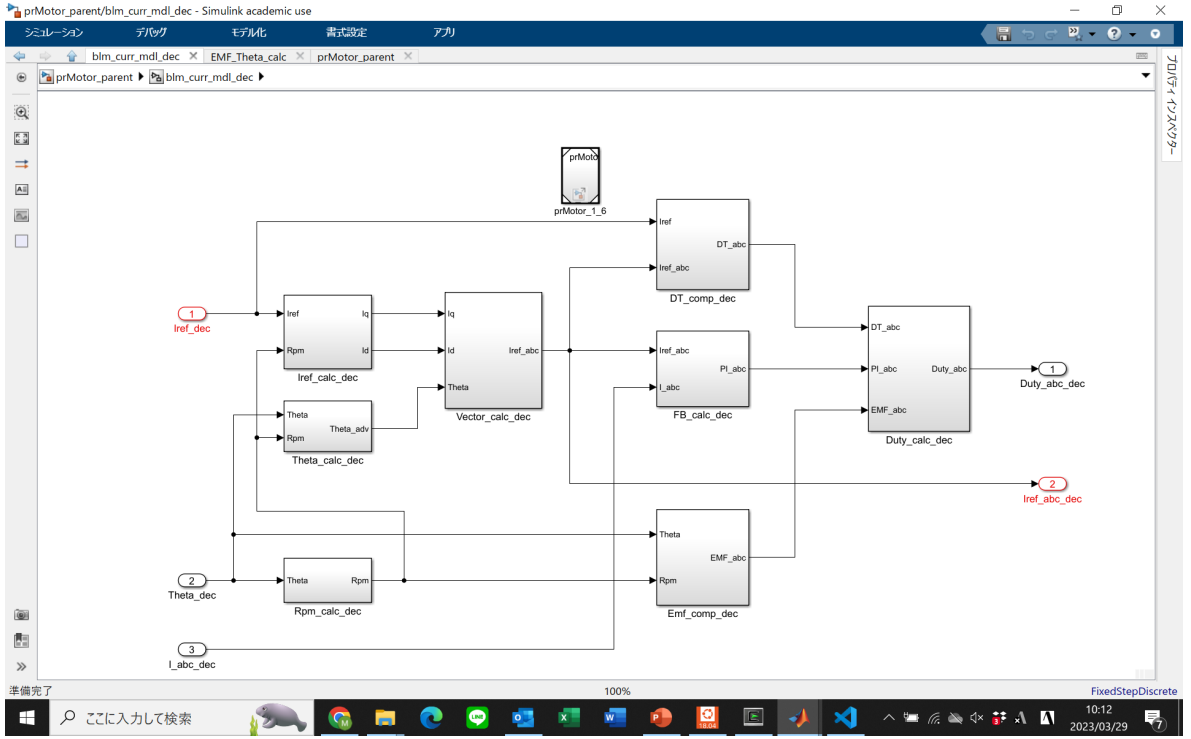


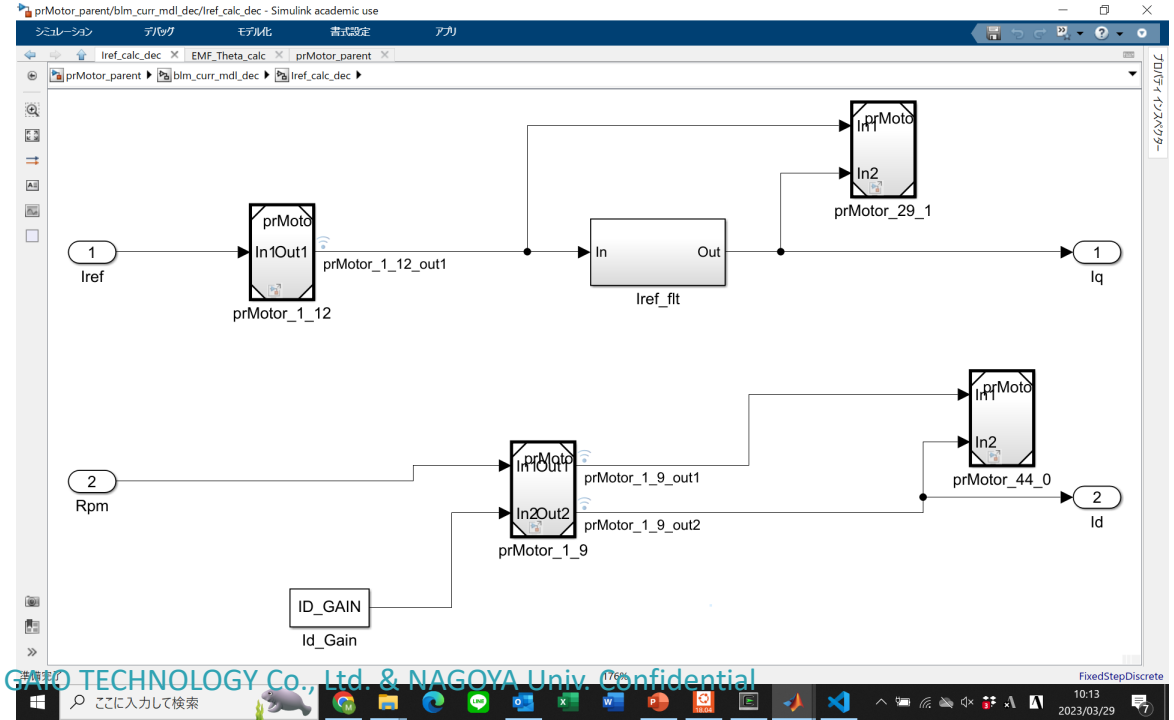




MBP結果
（コア割り当てにもとづく色分け）
左上：最上位階層
右上：第２階層
左下：第３階層







Model_Splitterにより参照モデル化した結果

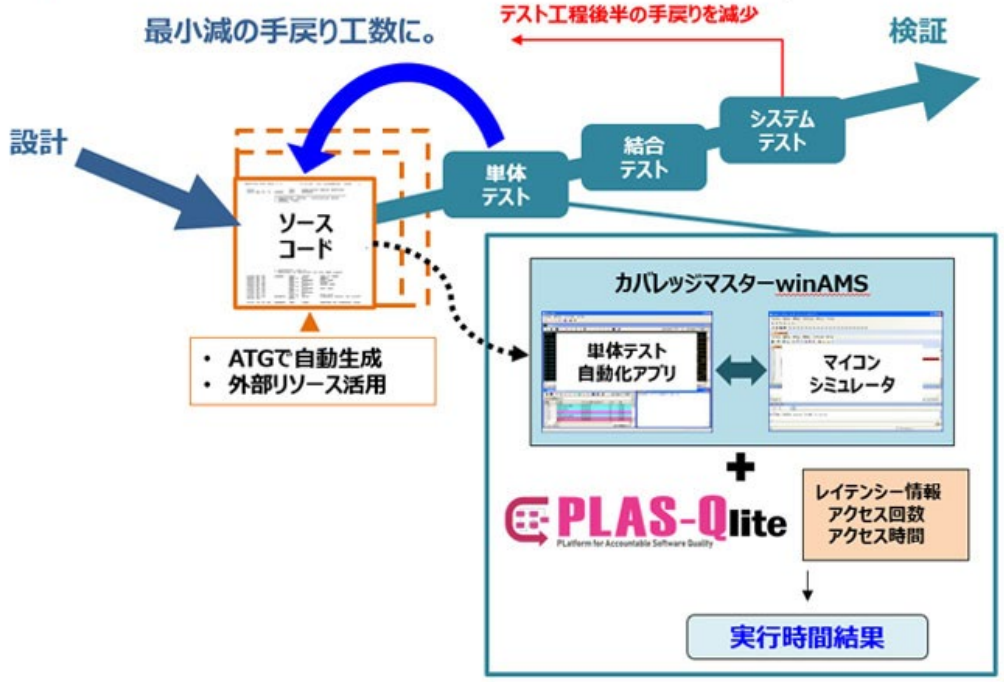
左上：最上位階層
右上：第２階層
左下：第３階層

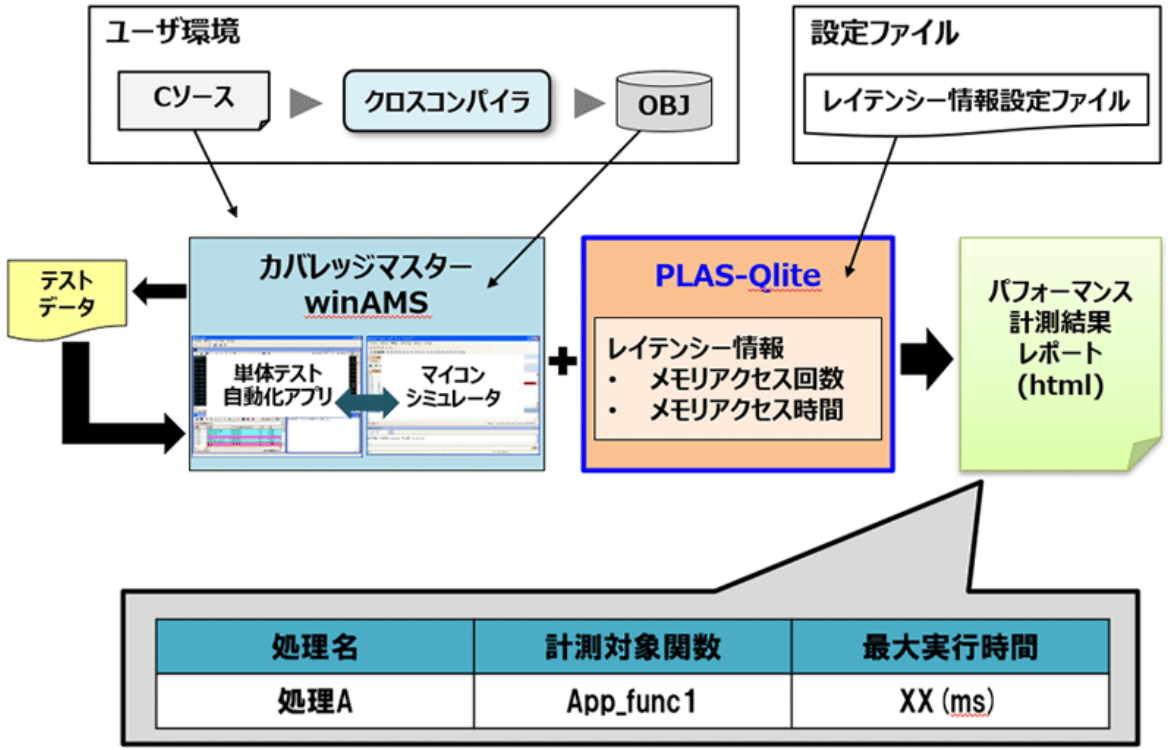
GAIO TECHNOLOGY Co., Ltd. & NAGOYA Univ. Confidential

ガイオ社性能計測ツール

CasePlayer2
カバレッジマスターwinAMS
PLAS-Qlite

PLAS-Qlite
PLatform for Accountable Software Quality





処理名	計測対象関数	最大実行時間
処理A	App_func1	XX (ms)

関数パフォーマンス計測結果の一部

テストデータごとのPF計測結果

テストデータ情報

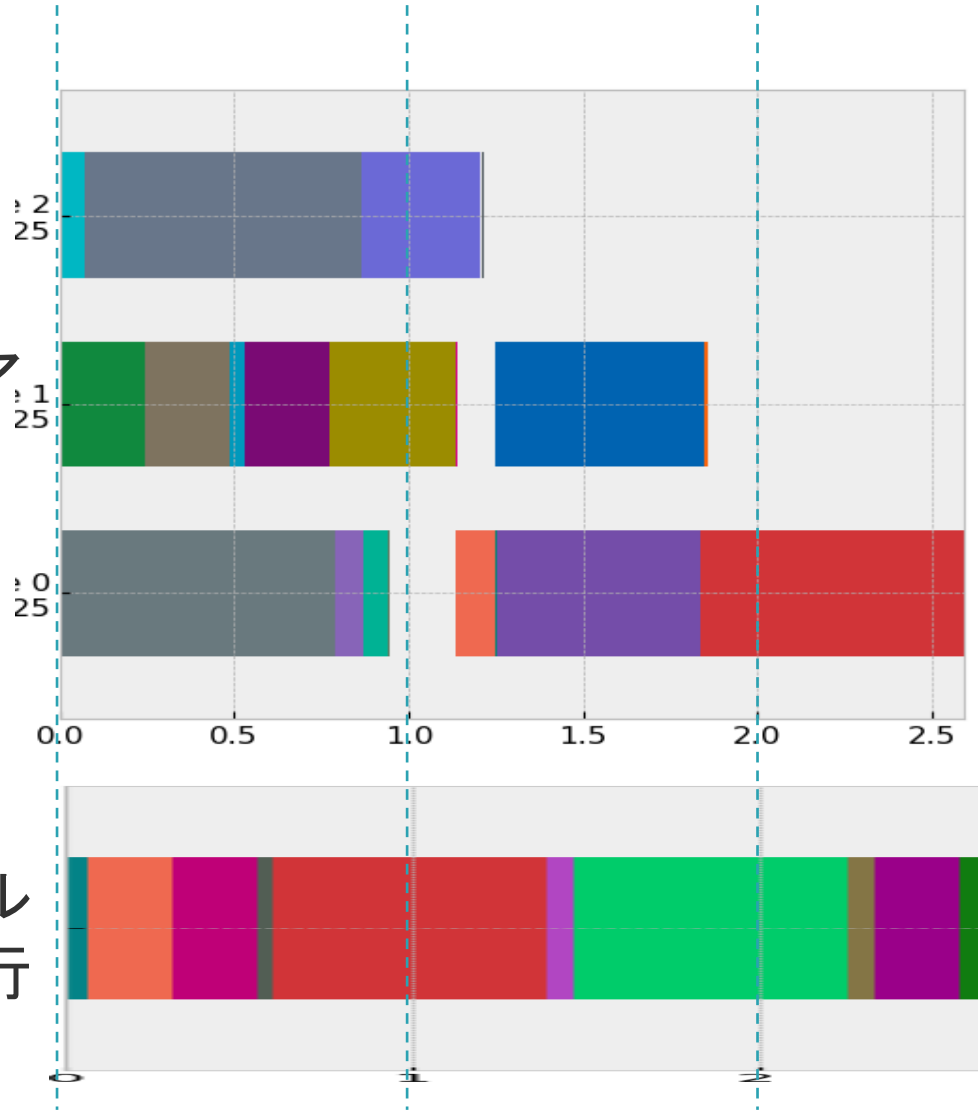
テストCSVファイル	prMotor_parent_run.csv
計測対象関数名	prMotor_parent_run
テストデータNo	1

関数呼び出し情報

関数名	実行時間 [ms]
prMotor_parent_run	364.251410
-prMotor_parent_initialize	0.308300
--prMotor_2_1_initialize	0.009150
--prMotor_3_0_initialize	0.009150
--prMotor_12_0_initialize	0.009150
--prMotor_0_0_initialize	0.009150

計測結果を用いた並列化のツール内部見積結果

３コア実行



コア間通信は高速通信
（20サイクル程度）を仮定
並列性能向上　約1.95倍

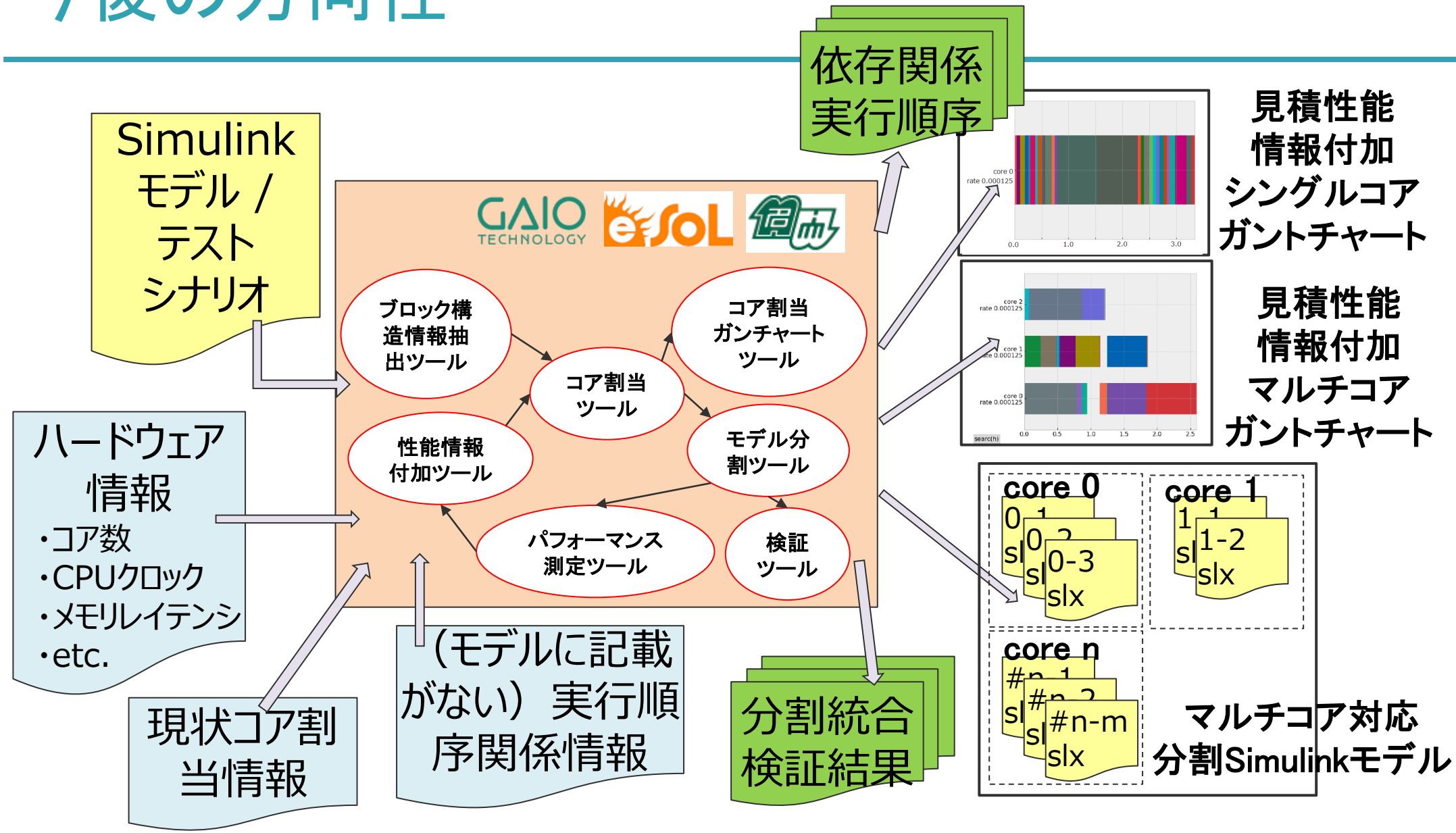
（注：上下の図で色の対応は取れていません）

シングルコア実行

まとめ

- マルチコア向けモデルベース開発のためのモデル分割とその設計検証
  - 仕様としてのモデルに並列動作情報を入れ、それにもとづく設計検証を構築したい
  - MBPのコア割当情報をもとにモデルを分割
    - 代数ループの回避
  - モデルからのテストケース生成を用いた分割統合検証手法
  - マイコンシミュレータを用いた性能検証手法

今後の方向性


Simulink
モデル /
テスト
シナリオ
ハードウェア
情報
・コア数
・CPUクロック
・メモリレイテンシ
・etc.
現状コア割
当情報
（モデルに記載
がない）実行順
序関係情報
GAIO TECHNOLOGY
ブロック構
造情報抽
出ツール
コア割当
ツール
コア割当
ガンチャート
ツール
性能情報
付加ツール
モデル分
割ツール
パフォーマンス
測定ツール
検証
ツール
依存関係
実行順序
分割統合
検証結果
見積性能
情報付加
シングルコア
ガントチャート
見積性能
情報付加
マルチコア
ガントチャート
core 0
0-3
slx
core 1
1-2
slx
core n
#n-m
slx
マルチコア対応
分割Simulinkモデル