マルチコア・メニーコア対応
Simulinkモデル自動分割ツール
～Simulinkモデルを並列動作考慮したSimulinkモデルに分割する～

２０２１年４月

名古屋大学大学院情報学研究科

枝廣　正人

この資料は枝廣研WWWサイト https://www.pdsl.jp/other からダウンロードできます。

本資料の一部画像は各製品販売企業WWWサイトから引用させていただきました。

Gサイバーチャンネル on Apr. 8, 2021.

講演概要

- 本セミナーにおいては、前回のセミナーで講演いたしました「モデルレベルでの並列化と関連技術」を用いて「並列動作を考慮したSimulinkモデルに自動で分割するツール」について紹介します。 本ツールは、名古屋大学とガイオ・テクノロジー株式会社様で共同研究を行い開発しました。設計したSimulinkモデルの振る舞いを変えずにモデルレベルで並列化を行い、マルチコア・メニーコアに対応したSimulinkモデルを自動で生成できるのが特徴となっております。

目次

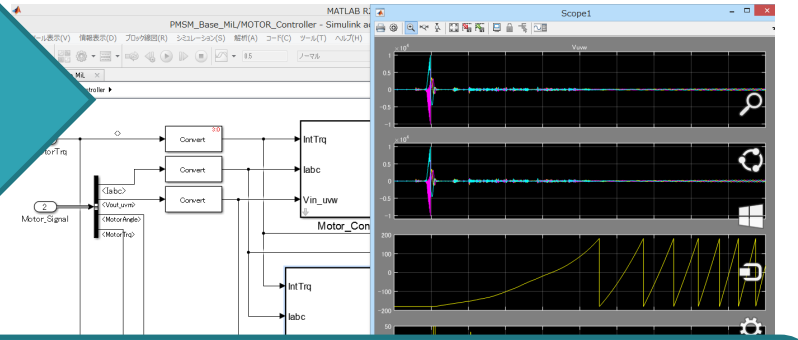
- 背景～どうしてモデルレベルでの並列化なのか～
- モデルベース並列化（前回セミナーの復習）
  - 並列化全体像
  - コア割当て
- マルチコア・メニーコア対応Simulinkモデル自動分割ツール
  - モデルベース並列化を実現するツールを活用
  - 振る舞いを変えずにSimulinkモデルを分割する
  - 適用例
- 適用事例の募集

Gサイバーチャンネル on Apr. 8, 2021.

モデルベース開発(MBD)

制御対象
（プラント）

MATLAB/Simulink



1．数理モデル化

2．制御器の数理モデル設計
（シミュレーションベース）

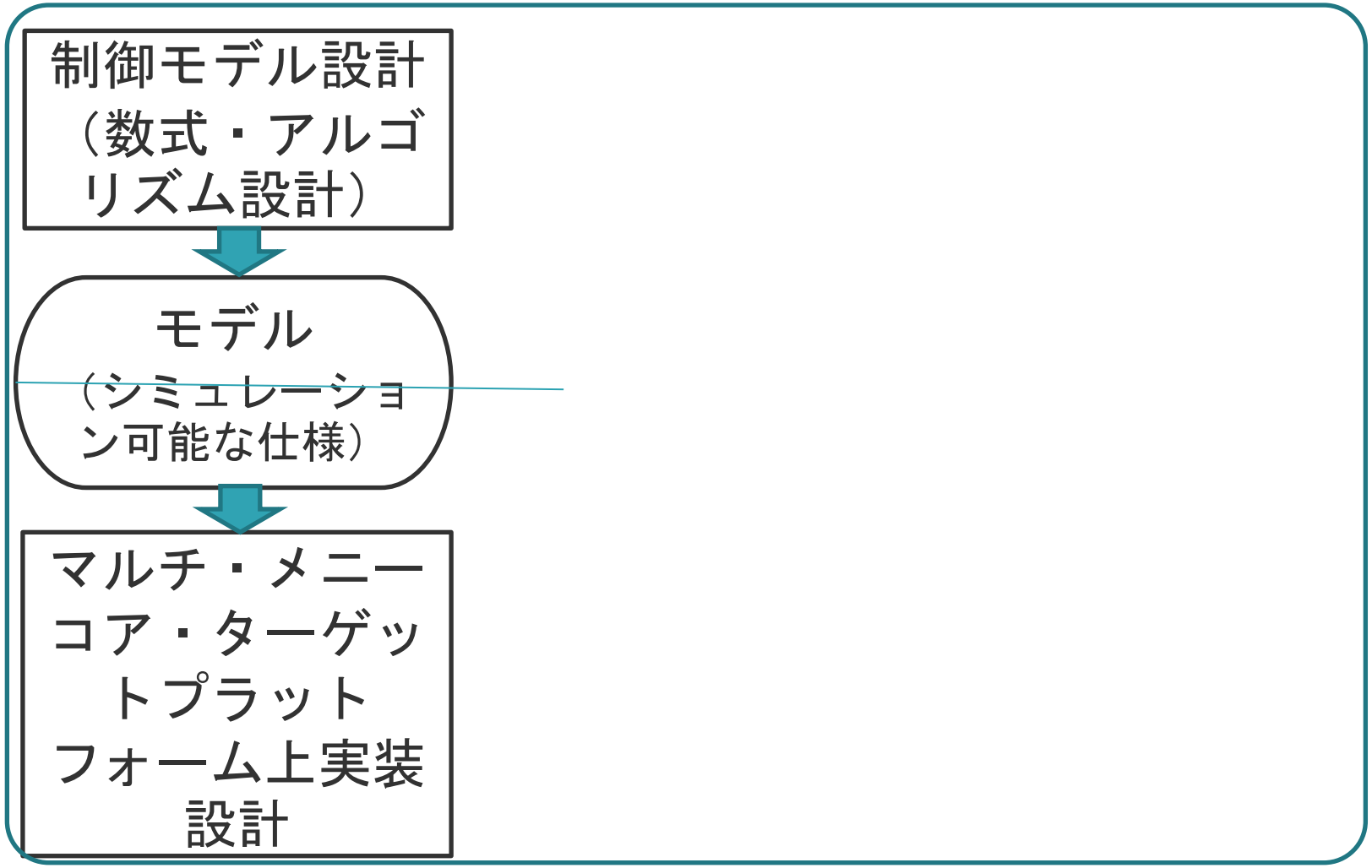
制御器（コントローラ）

3．実装コード化

制御

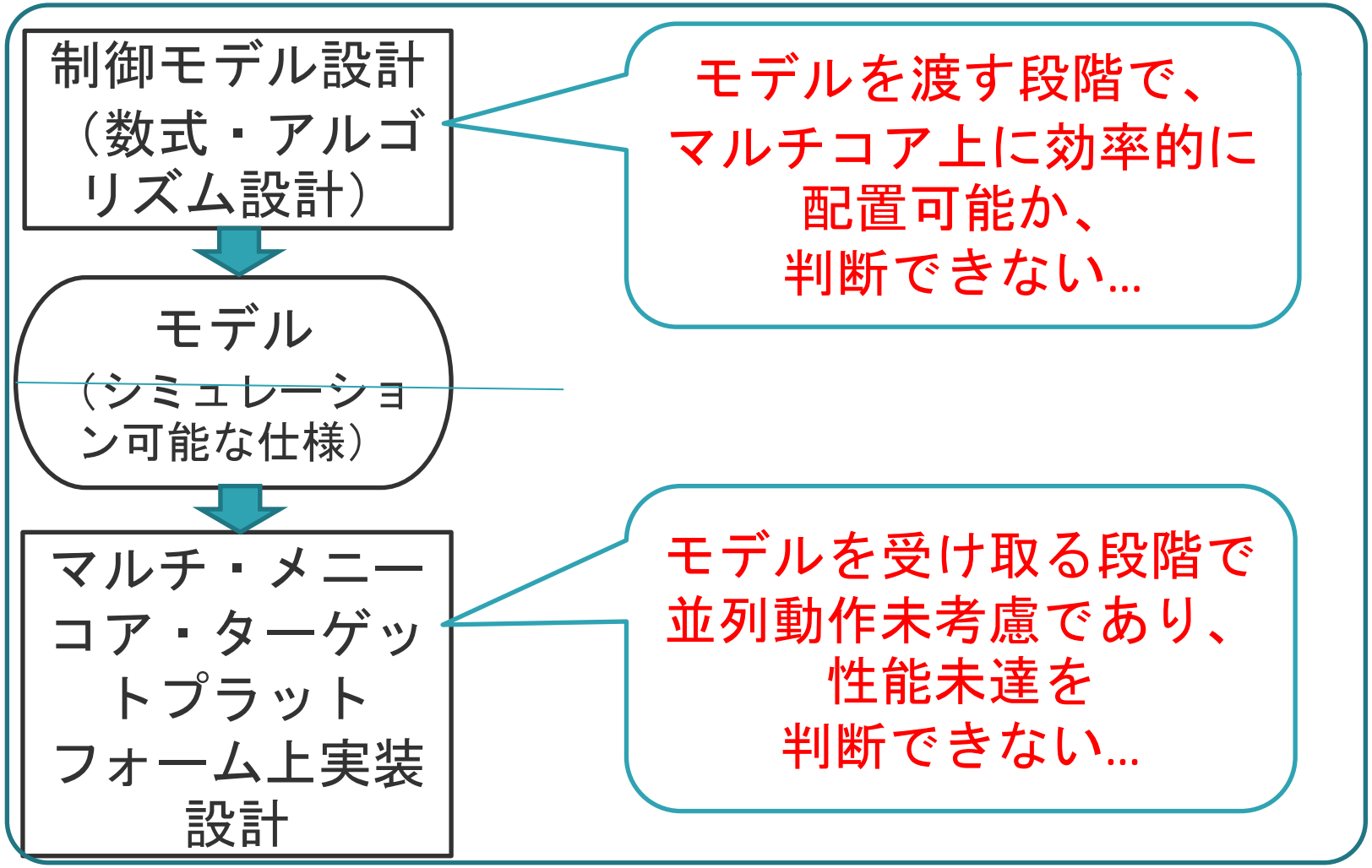
Gサイバーチャンネル on Apr. 8, 2021.

MBDの現状



Gサイバーチャンネル on Apr. 8, 2021.

モデルレベルで考えることができないと、、、

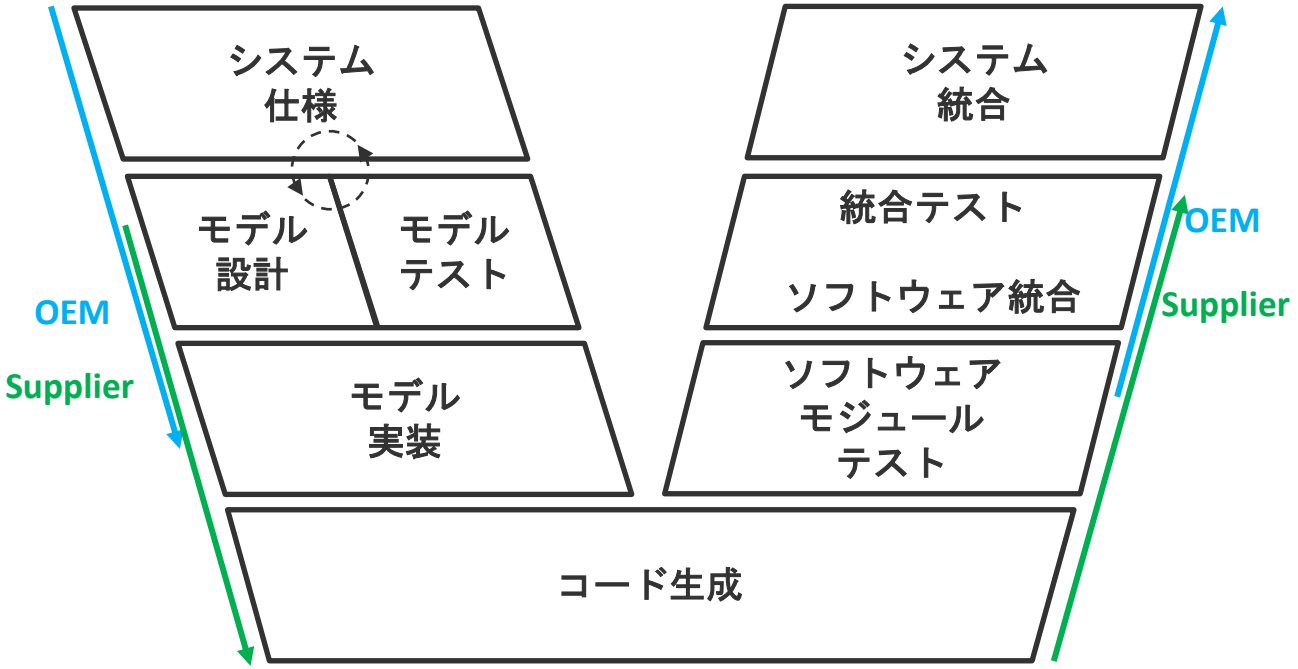


Gサイバーチャンネル on Apr. 8, 2021.

車載システム開発

車載システム開発にモデルベース開発（MBD）が普及

- MATLAB/SimulinkがMBDに用いられるツールの代表例
- 自動車メーカ（OEM）と自動車部品メーカ（Supplier）の協業開発
- MBDのV字モデルに則って開発が行われている



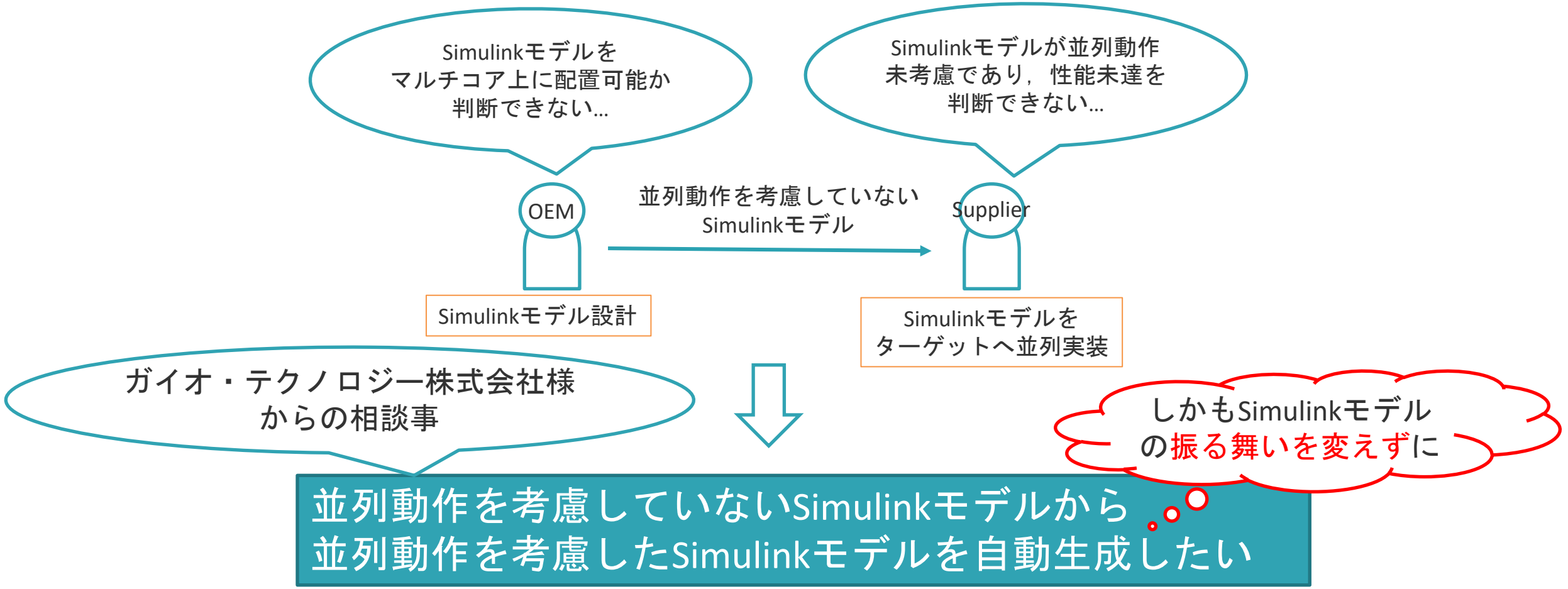
MBDのV字モデル例

V字モデル図はMichailidis, A., Spieth, U., Ringler, T., Hedenetz, B.and Kowalewski, S.
Test front loading in early stages of automotive software development based on AUTOSARより引用

Gサイバーチャンネル on Apr. 8, 2021.

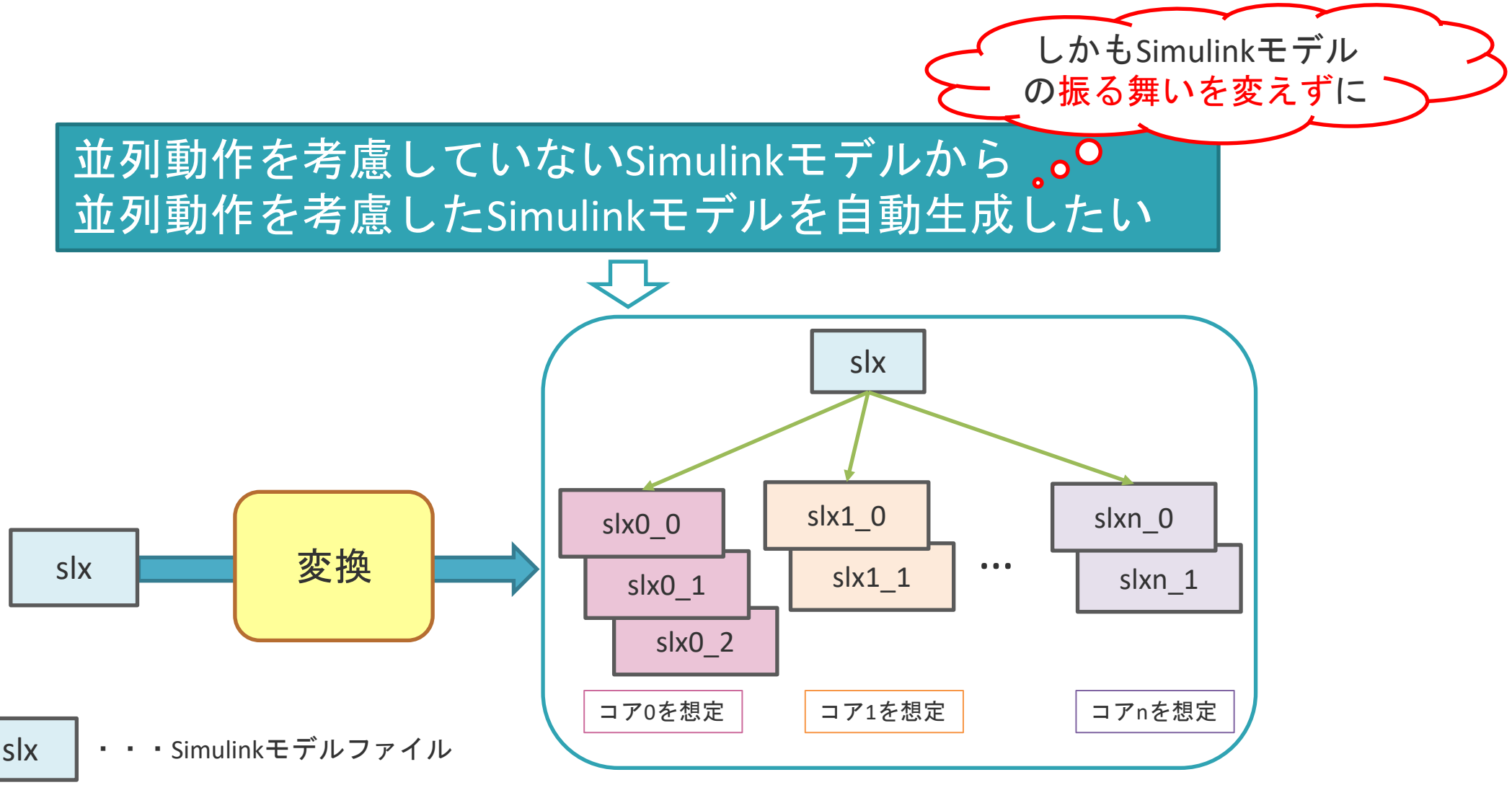
車載システム開発のモデルベース開発(MBD)

OEM とSupplier間はSimulinkモデルでやり取り



Gサイバーチャンネル on Apr. 8, 2021.

ガイオ・テクノロジー株式会社様からの相談事イメージ図



Gサイバーチャンネル on Apr. 8, 2021.

目次

- 背景～どうしてモデルレベルでの並列化なのか～
- モデルベース並列化（前回セミナーの復習）
  - 並列化全体像
  - コア割当て
- マルチコア・メニーコア対応Simulinkモデル自動分割ツール
  - モデルベース並列化を実現するツールを活用
  - 振る舞いを変えずにSimulinkモデルを分割する
  - 適用例
- 適用事例の募集

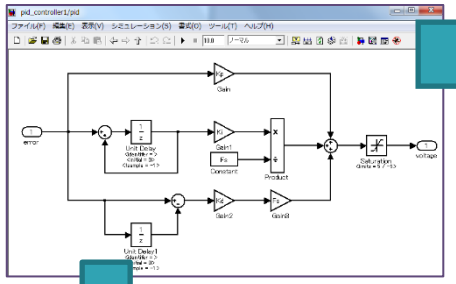
モデルレベルでどう並列化するのか？

- モデル（ブロック線図）のレベルで、ブロックをコアに割当
  - すべてのコアにおける負荷を同一にしつつコア間通信を最小化（いわゆるmin-cut手法）
- 重要：各ブロックに性能情報が付いていること
  - 性能見積の話題で
- 重要：すべての依存関係がブロック線図上の線として表現されていること
  - 実際には例外あり。SimulinkではData Store Memory
    →現状では依存関係をつけるか、同一Data Store Memoryに対するすべてのアクセスブロックを同じコアに配置して生成コードの順序を変えないことで対応
- 重要：並列動作時に逐次動作とふるまいを変えないこと
  - 検証の話題で

Gサイバーチャンネル on Apr. 8, 2021.

モデルからの情報抽出

Simulinkモデル



②ブロックレベル構造抽出 (CSPグラフ構造)

①コード生成

```
/* Saturate: '<S1>/Saturation' */
if (pid_controller1_pid_Sum2_1 >= p
  pid_controller1_pid_Saturation_1
} else if (pid_controller1_pid_Sum2
{
  pid_controller1_pid_Saturation_1
} else {
  pid_controller1_pid_Saturation_1
}

/* End of Saturate: '<S1>/Saturatio

/* Sum: '<S1>/Sum' incorporates:
 *  Inport: '<Root>/error'
```

③ブロック処理コード抽出

```
<block blocktype="Saturate" name="pid_controller1_pid
  <input line="pid_controller1_pid_Sum2_1" port="pid_
    <connect block="pid_controller1_pid_Sum2" port="p
  </input>
  <output line="pid_controller1_pid_Saturation_1" por
    <connect block="pid_controller1_pid_voltage" port
  </output>
  <var line="pid_controller1_pid_Sum2_1" mode="input"
  <var line="pid_controller1_pid_Saturation_1" mode="
  <param name="Saturation_UpperSat" storage="pid_cont
  <param name="Saturation_LowerSat" storage="pid_cont
  <code file="models/pid/pid_controller1_ert_rtw/pid_
    /* Saturate: '&lt;S1&gt;/Saturation' */
    if (pid_controller1_pid_Sum2_1 &gt;= pid_controll
      pid_controller1_pid_Saturation_1 = pid_controll
    } else if (pid_controller1_pid_Sum2_1 &lt;= pid_c
    {
      pid_controller1_pid_Saturation_1 = pid_controll
    } else {
      pid_controller1_pid_Saturation_1 = pid_controll
    }
    /* End of Saturate: '&lt;S1&gt;/Saturation' */
  </code>
  <code file="models/pid/pid_controller1_ert_rtw/pid_
    pid_controller1_pid_Saturation_1 = 0.0F;
  </code>
  <performance best="26" type="task" typical="31" wor
  <performance best="15" type="init" typical="15" wor
  <forward block="pid_controller1_voltage" type="port
    <var line="pid_controller1_pid_1" mode="input" na
  </forward>
  <backword block="pid_controller1_pid_Sum2" type="da
    <var line="pid_controller1_pid_Sum2_1" mode="outp
  </backword>
</block>
```

④抽出コード処理量見積

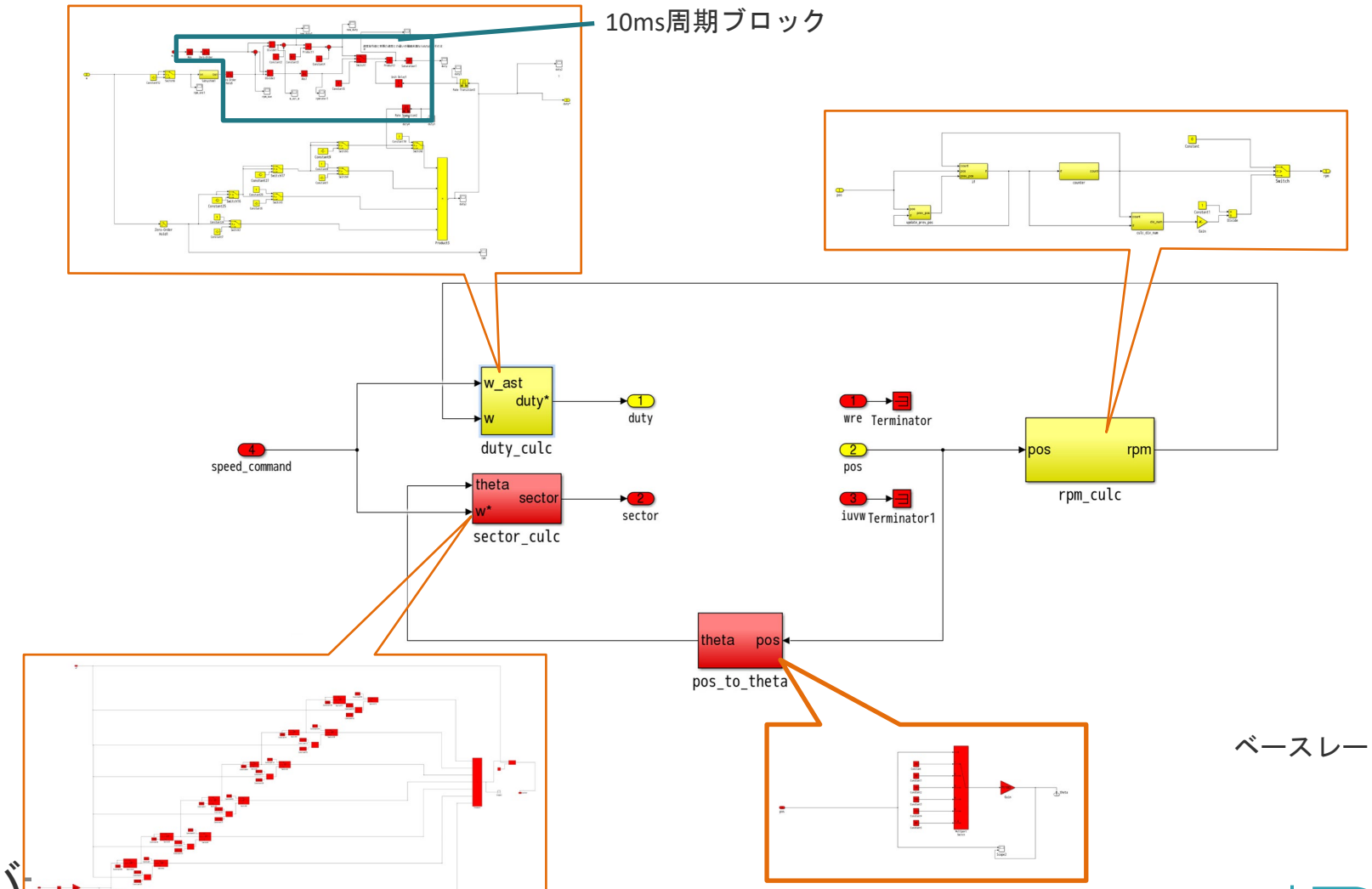
```
<ComponentSet name="Board_B0">↓
    <ComponentSet name="Cluster_B0C0">↓
        <ComponentSet name="PE_B0C0P1">↓
            <SlaveComponent name="LRAM_B0C0P1"
            <MasterComponent name="CPU_B0C0P1"
                <CommonInstructionSet name="
                    <Instruction name="ret">
                        <Latency best="10.0" typical="10.
                        <Pitch best="10.0" typical="10.0"
                    </Instruction>↓
                    <Instruction name="br">↓
```

SHIM

ブロックレベル構造XML (BLXML)

Gサイバーチャンネル on Apr. 8, 2021.

２コア配置の例（色分けはコア割当）



ベースレート：100us

Gサイバー

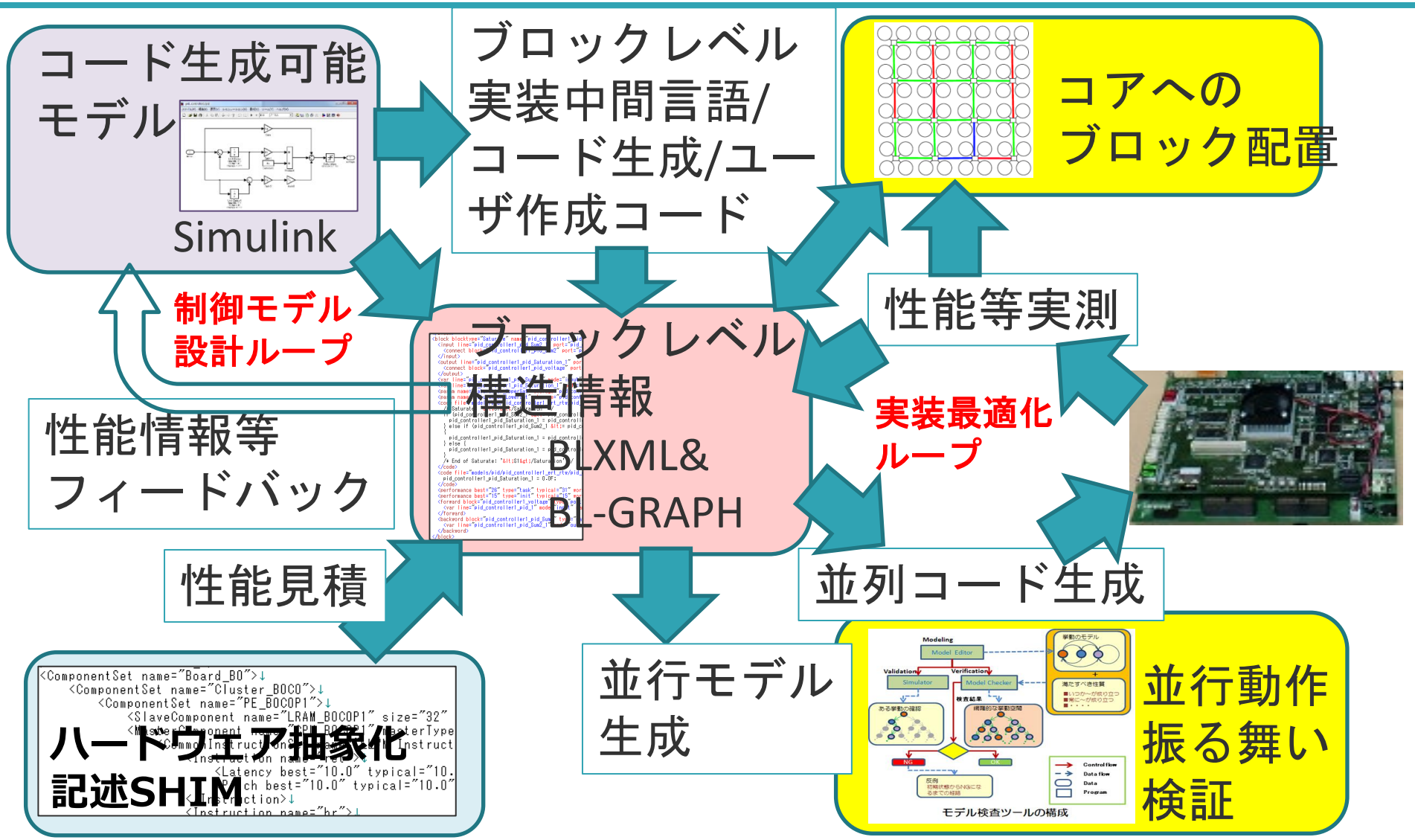
自動コード生成

- 同一コアに割り当てられたブロックに対応するコードを順に並べ、必要に応じてコード間にコア間通信を配置

```
/* Block: pid_controller1_pid_Saturation */
agent sc_task_0010 () {
  interface {
    in<real32_T> CH_0013_0010;

    out<real32_T> CH_0010_I00002;

    spec {
      CH_0013_0010;
      CH_0010_I00002;
    };
  }

  map {
    SigmaC_agent_setUnitType(SigmaC_agent_self(),"k1-cluster");
  }

    /* params */
    struct {
      real32_T Saturation_UpperSat;
      real32_T Saturation_LowerSat;
    } pid_controller1_P = {
  5.0F,                                 /* Computed Parameter: S
                                         * Referenced by: '<S1>/
                                         */

  -5.0F                                 /* Computed Parameter: S
                                         * Referenced by: '<S1>/
                                         */

    };

    /* input variables */
    real32_T pid_controller1_pid_Sum2_1;

    /* output variables */
    real32_T pid_controller1_pid_Saturation_1;

  init{
  /* initialize task context */
  pid_controller1_pid_Saturation_1 = 0.0F;
  }

  void start ()
    exchange (CH_0013_0010 ch_0013_0010,
              CH_0010_I00002 ch_0010_I00002) {

    /* input */
    pid_controller1_pid_Sum2_1 = ch_0013_0010;

    /* C code */
  /* Saturate: '<S1>/Saturation' */
  if (pid_controller1_pid_Sum2_1 >= pid_controller1_P.Saturat
    pid_controller1_pid_Saturation_1 = pid_controller1_P.Satu
  } else if (pid_controller1_pid_Sum2_1 <= pid_controller1_P.
  {
    pid_controller1_pid_Saturation_1 = pid_controller1_P.Satu
  } else {
    pid_controller1_pid_Saturation_1 = pid_controller1_pid_Su
  }

  /* End of Saturate: '<S1>/Saturation' */


    /* output */
    ch_0010_I00002 = pid_controller1_pid_Saturation_1;
  }
}
```

Gサイバーチャンネル on Apr. 8, 2021.

並列化全体像



Gサイバーチャンネル on Apr. 8, 2021.

並列化全体像

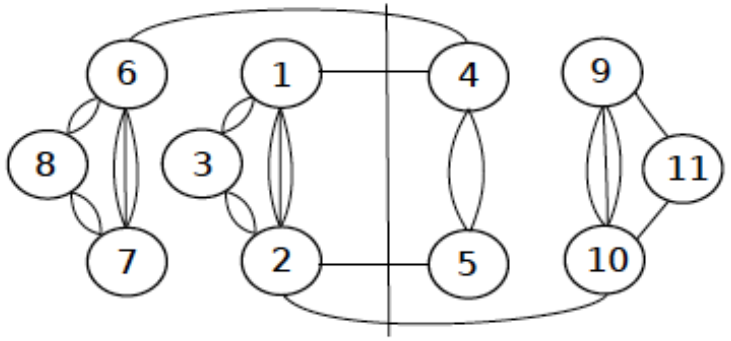
Gサイバーチャンネル on Apr. 8, 2021.

目次

- 背景～どうしてモデルレベルでの並列化なのか～
- モデルベース並列化（前回セミナーの復習）
  - 並列化全体像
  - コア割当て
- マルチコア・メニーコア対応Simulinkモデル自動分割ツール
  - モデルベース並列化を実現するツールを活用
  - 振る舞いを変えずにSimulinkモデルを分割する
  - 適用例
- 適用事例の募集

Min-Cut法

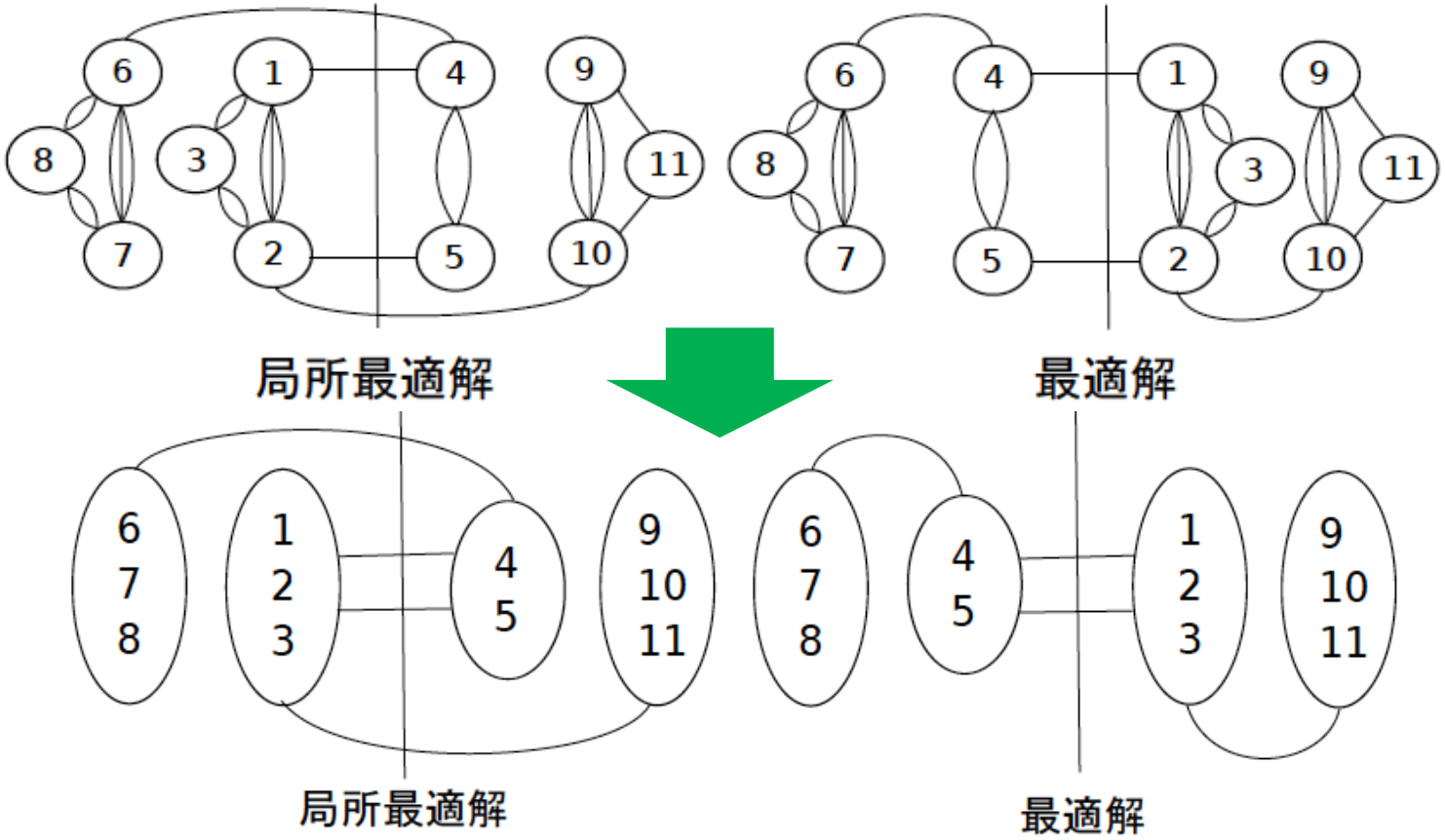
- 負荷を同一にしてコア間通信を最小化
  - 典型的には２段階
    - 初期解・・・問題に応じて良さそうな解を高速に生成
    - 反復・・・両側から選び、解が良くなる方向で交換
  - 局所最適解に陥りやすく、脱出のための工夫がポイント
    - 局所最適解・・・最適解に到達していないのに、どれを交換しても解は良くならない
  - 脱出アルゴリズム
    - Kernighan-Lin法
    - メタヒューリスティック
      - Simulated Annealingなど
    - 階層クラスタリング
    - など数多く提案されている



局所最適解

階層クラスタリング手法(*)

- 階層的にクラスタリングし、崩しながらペア交換アルゴリズムを適用すると局所最適解から脱出しやすい



(*) M. Edahiro and T. Yoshimura: New Placement and Global Routing Algorithms for Standard Cell Layouts. DAC 1990: pp. 642-645

二重階層クラスタリング法

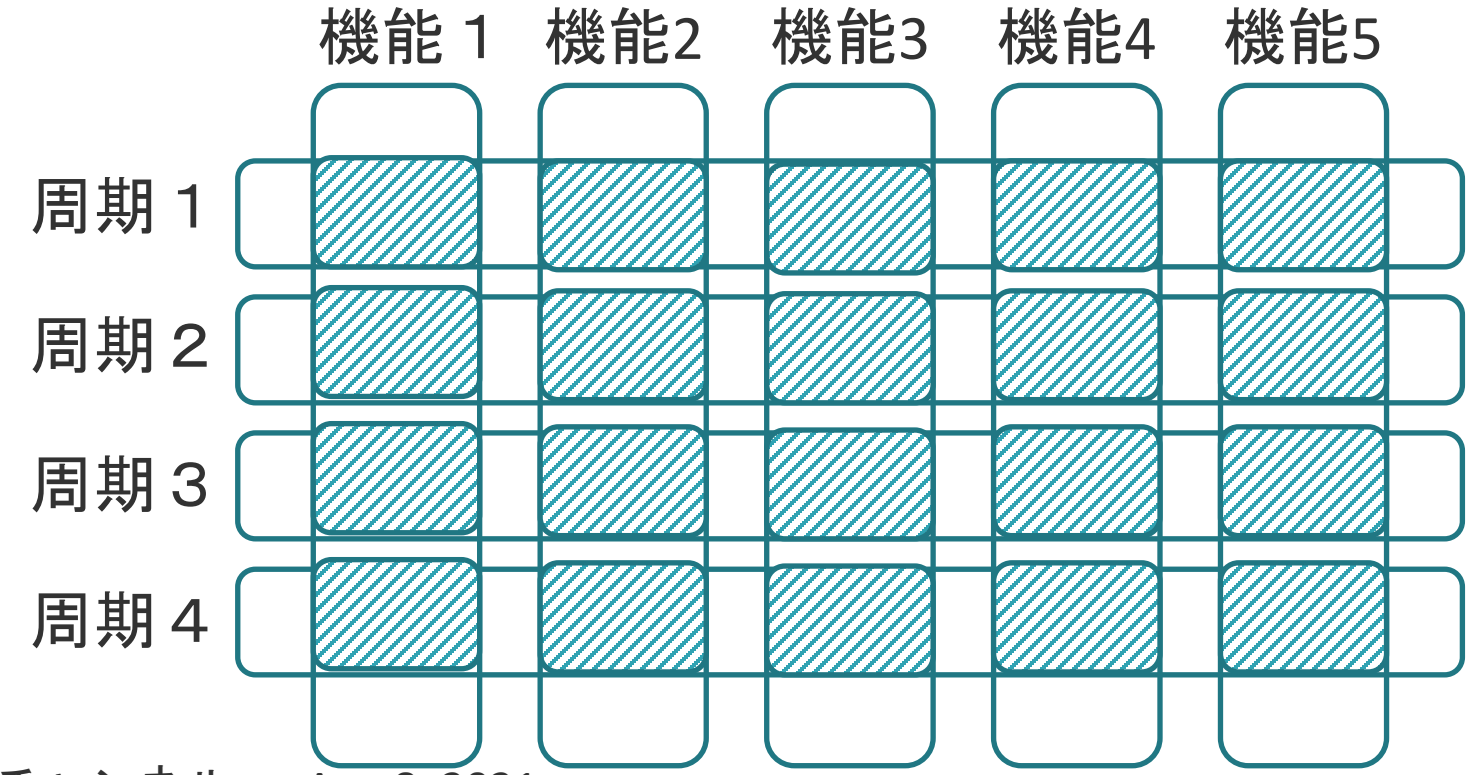
- 設計制約等により同じプロセッサに配置したいグループなどの考慮を入れる
- 二重階層クラスタリング法（全体）
  1. 第ゼロ段階のクラスタリング（ユーザ強制配置）
  2. 第一段階のクラスタリング
  3. 第二段階のクラスタリング
  4. 第二段階クラスタのコア配置（グローバル配置）
  5. 第一段階クラスタのコア配置（ローカル配置）
     - ローカル配置に階層クラスタリング法を適用
  6. 第一段階および第ゼロ段階クラスタリングを展開し、並列化完了

第一段階のクラスタリング

- メモリアクセスクラスタリング
  - 同じ共有メモリ資源に読み書きする処理ブロックをグループ化
  - 検証に有利
- 一括コードブロッククラスタリング
  - 複数ブロックに対し、一括してコード生成されるブロック群のグループ化
- サブシステム関連クラスタリング
  - 特殊ポート付や、標準ライブラリなど分割しない方が良いサブシステムのグループ化

第二段階のクラスタリング

- 例えば（機能 (Atomic Subsystem)×周期）でグループ化
- グローバル配置
  - 個々の第二段階クラスタをコア（群）へ割当
  - クラスタ数が多くないため様々な手法が可能



Gサイバーチャンネル on Apr. 8, 2021.

グローバル配置アルゴリズム

- 混合整数線形計画を利用
  - 鍾，枝廣. "組込み制御システムに対するマルチコア向けモデルレベル自動並列化手法", 情報処理学会論文誌, Vol.59, pp. 735-747, 2018.
  - 性能ヘテロジニアス対応
    - コアに性能倍率属性を対応させる
    - Z. Zhong and M. Edahiro. "Model-Based Parallelizer for Embedded Control Systems on Single-ISA Heterogeneous Multicore Processors," International Journal of Computer & Technology, Vol. 19, pp. 7470-7484, 2019.
  - 一般のヘテロジニアス対応
    - コアにコア種類属性を対応させ、ブロックにコア種類ごとの性能情報を対応させる（コア種類: CPU, GPU, etc.)
    - 一般には非線形問題になるが、パスに注目することにより、混合整数線形計画問題に帰着
    - Z. Zhong and M. Edahiro. “Model-Based Parallelization for Simulink Models on Multicore CPUs and GPUs," International Journal of Computer & Technology, Vol. 20, pp. 1-13, 2020.

Gサイバーチャンネル on Apr. 8, 2021.

目次

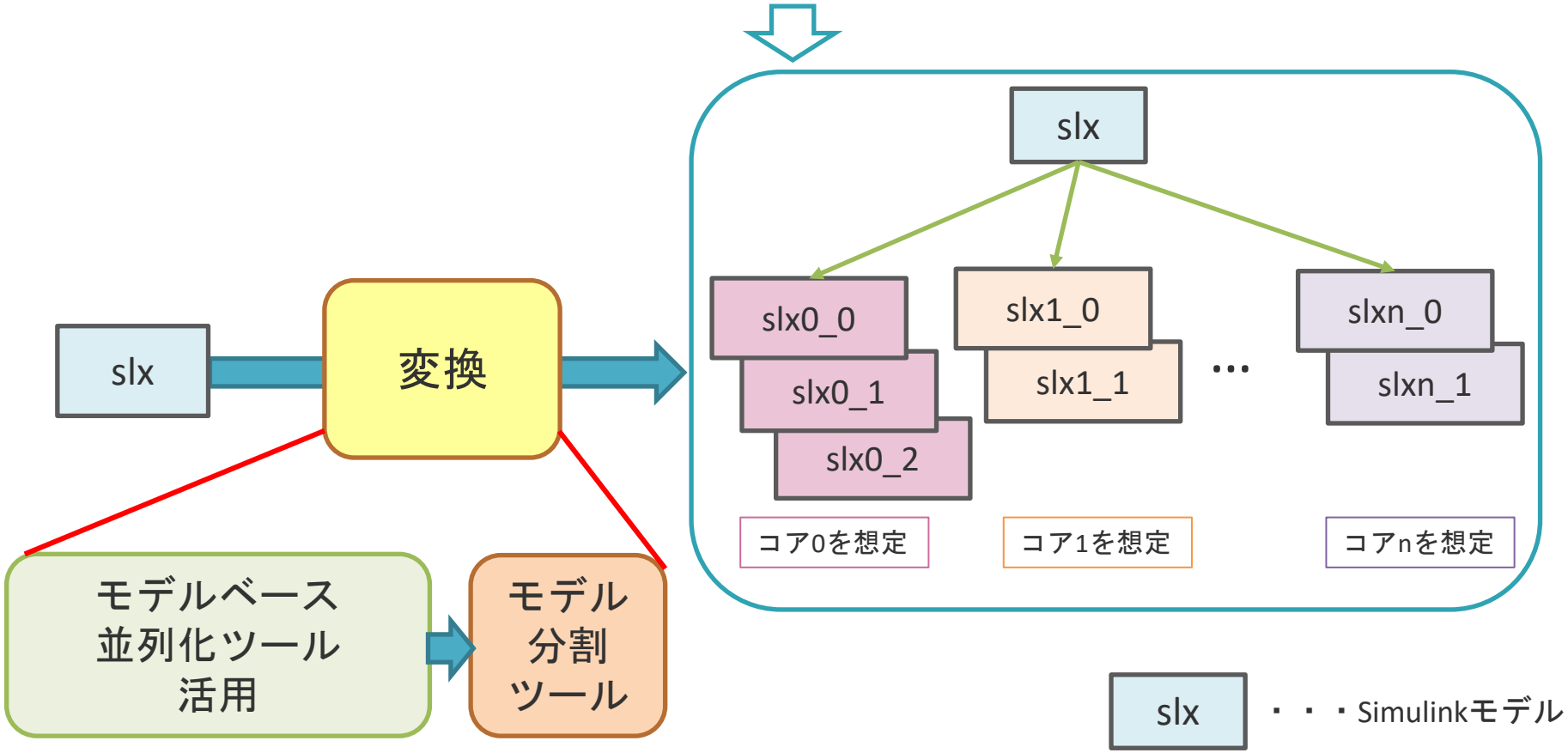
- 背景～どうしてモデルレベルでの並列化なのか～
- モデルベース並列化（前回セミナーの復習）
  - 並列化全体像
  - コア割当て
- マルチコア・メニーコア対応Simulinkモデル自動分割ツール
  - モデルベース並列化を実現するツールを活用
  - 振る舞いを変えずにSimulinkモデルを分割する
  - 適用例
- 適用事例の募集

Gサイバーチャンネル on Apr. 8, 2021.

相談事イメージ図再掲

並列動作を考慮していないSimulinkモデルから
並列動作を考慮したSimulinkモデルを自動生成したい

しかもSimulinkモデル
の振る舞いを変えずに

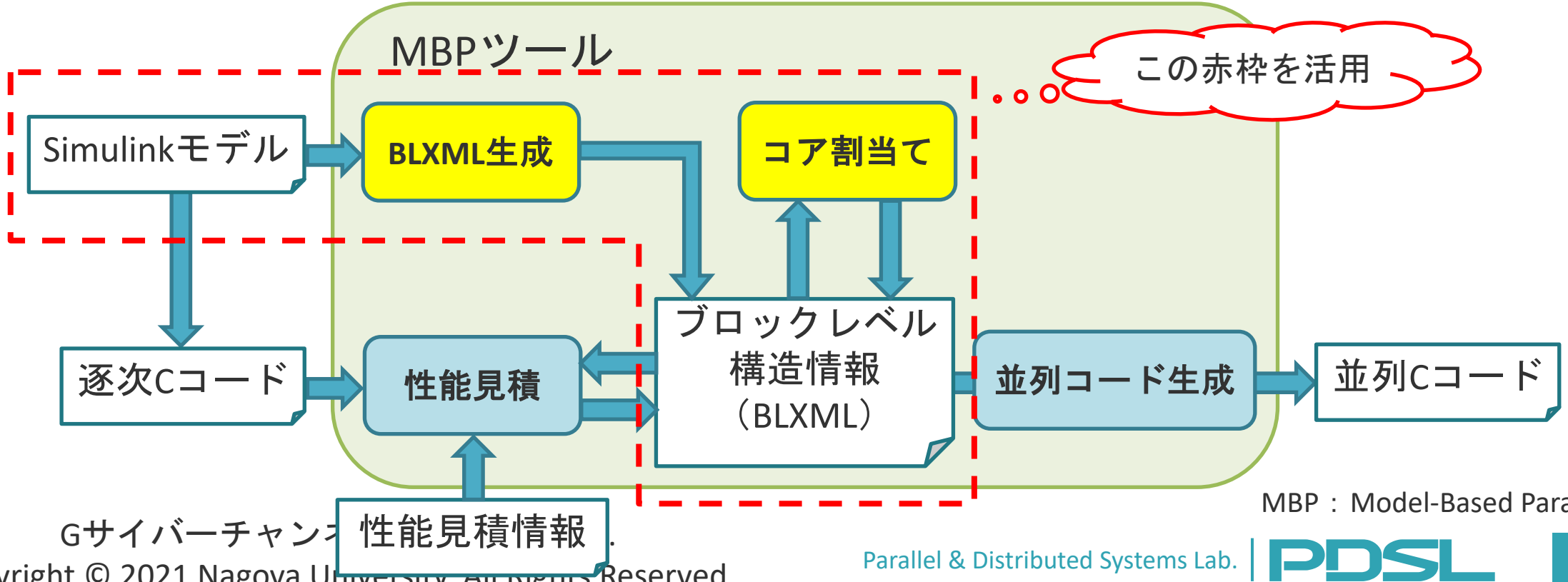


slx ・・・Simulinkモデルファイル

Gサイバーチャンネル on Apr. 8, 2021.

モデルベース並列化を実現するツールを活用

- Simulinkモデルからブロックレベル構造情報（BLXML）を生成
- 生成したBLXMLに並列化後のコア割当て情報のみを付加
- モデル分割ツール(Model Splitter)に以下の情報を入力
  - Simulinkモデル
  - コア割当て情報付きBLXML

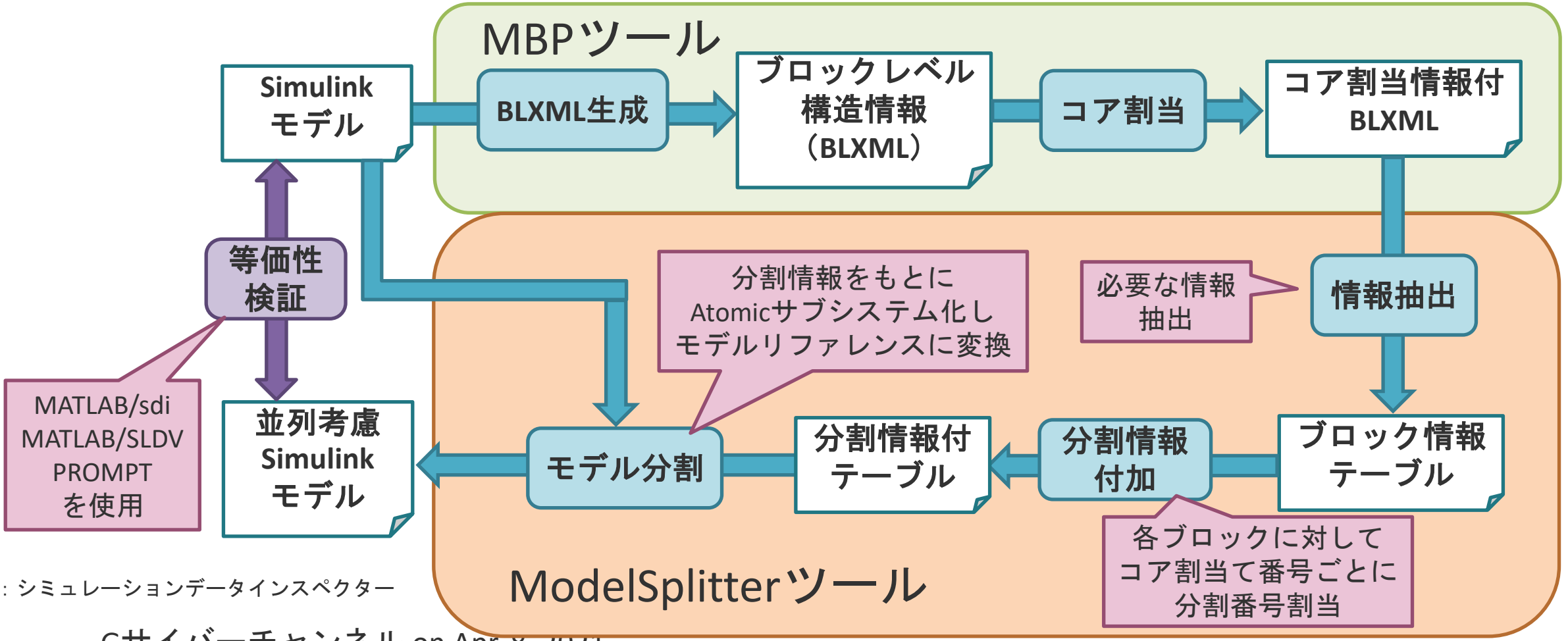


MBP： Model-Based Parallelizer

Gサイバーチャン

振る舞いを変えずにSimulinkモデルを分割する

- Simulinkモデルから並列動作を考慮した複数のSimulinkモデルに分割するモデルレベルブロック分割手法の全体像



sdi：シミュレーションデータインスペクター

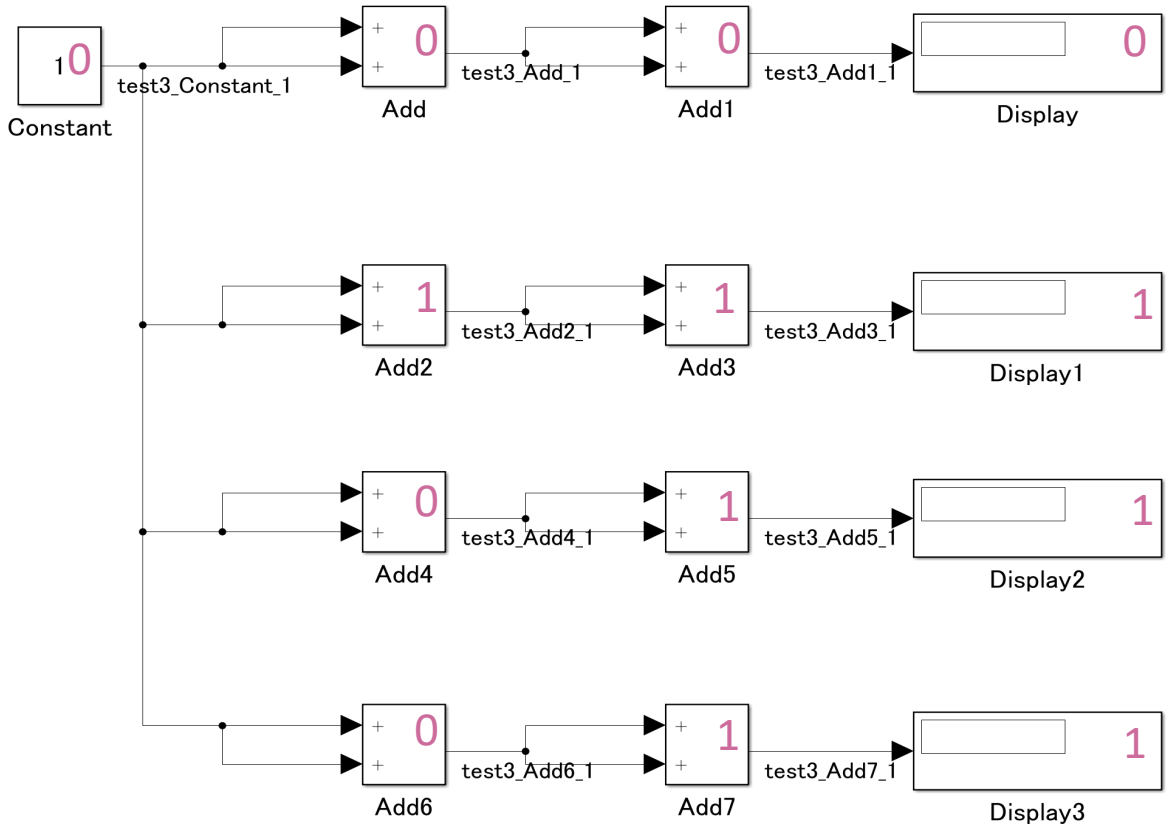
分割情報付加～分割番号を割り当てる～

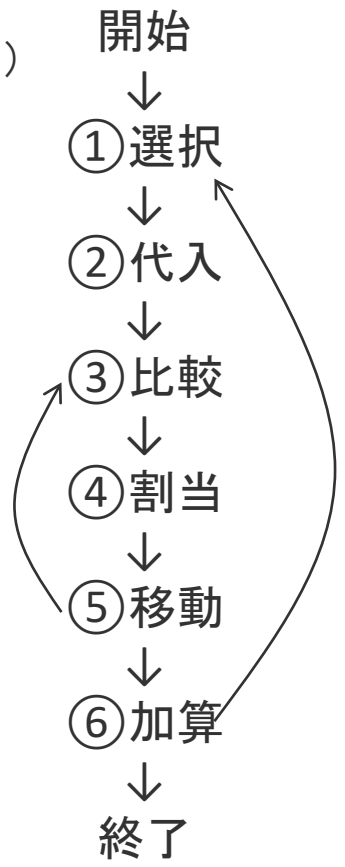
- 簡単なSimulinkモデルを例に分割情報付加の流れを説明

コア0の分割番号 0
コア1の分割番号 0

基準コア番号 □

0 … コア割当番号
（MBPツールで取得）

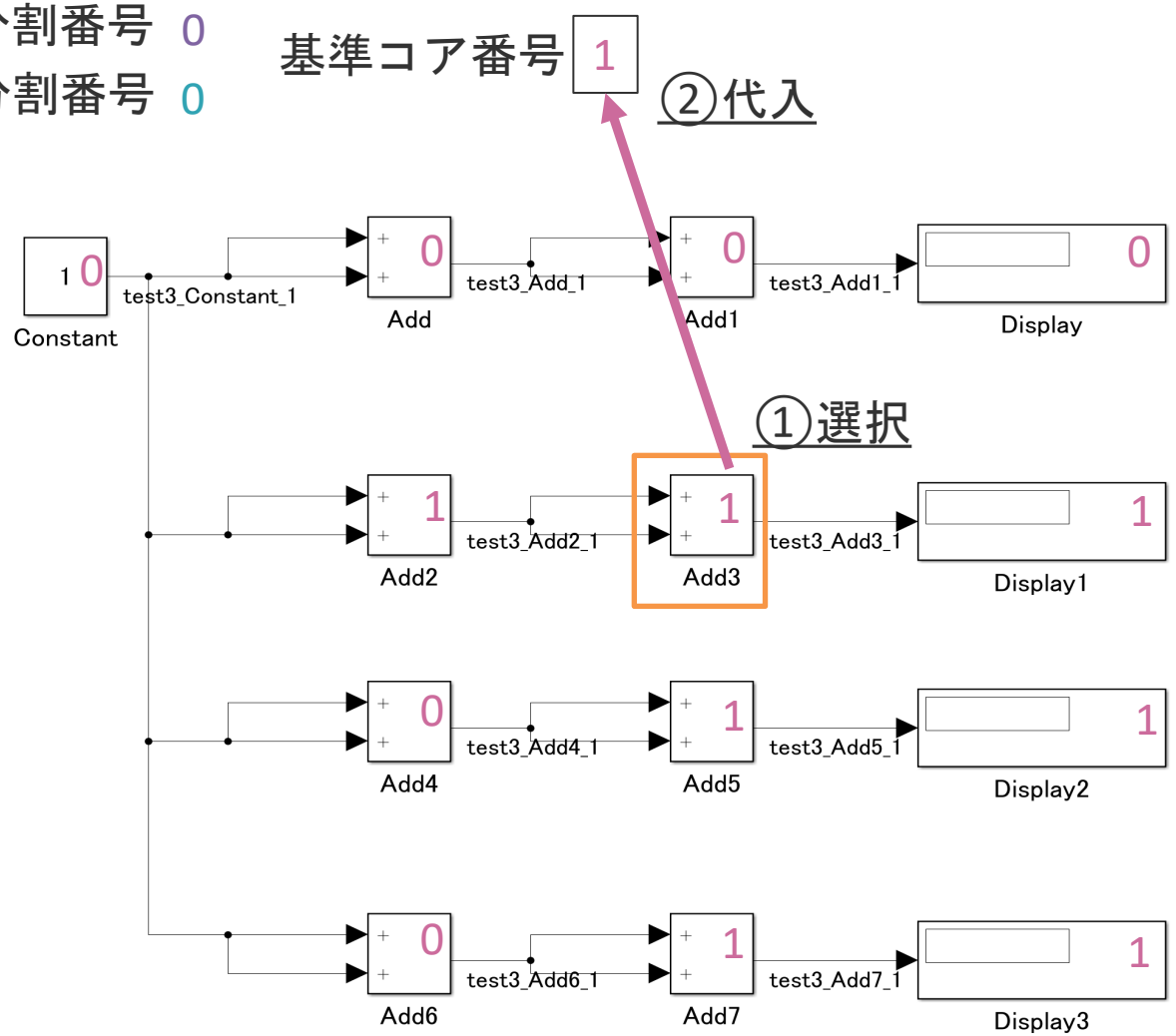


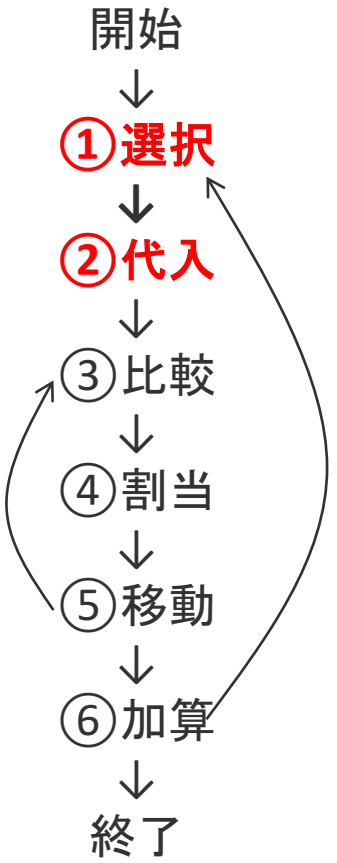


Gサイバーチャンネル on Apr. 8, 2021.

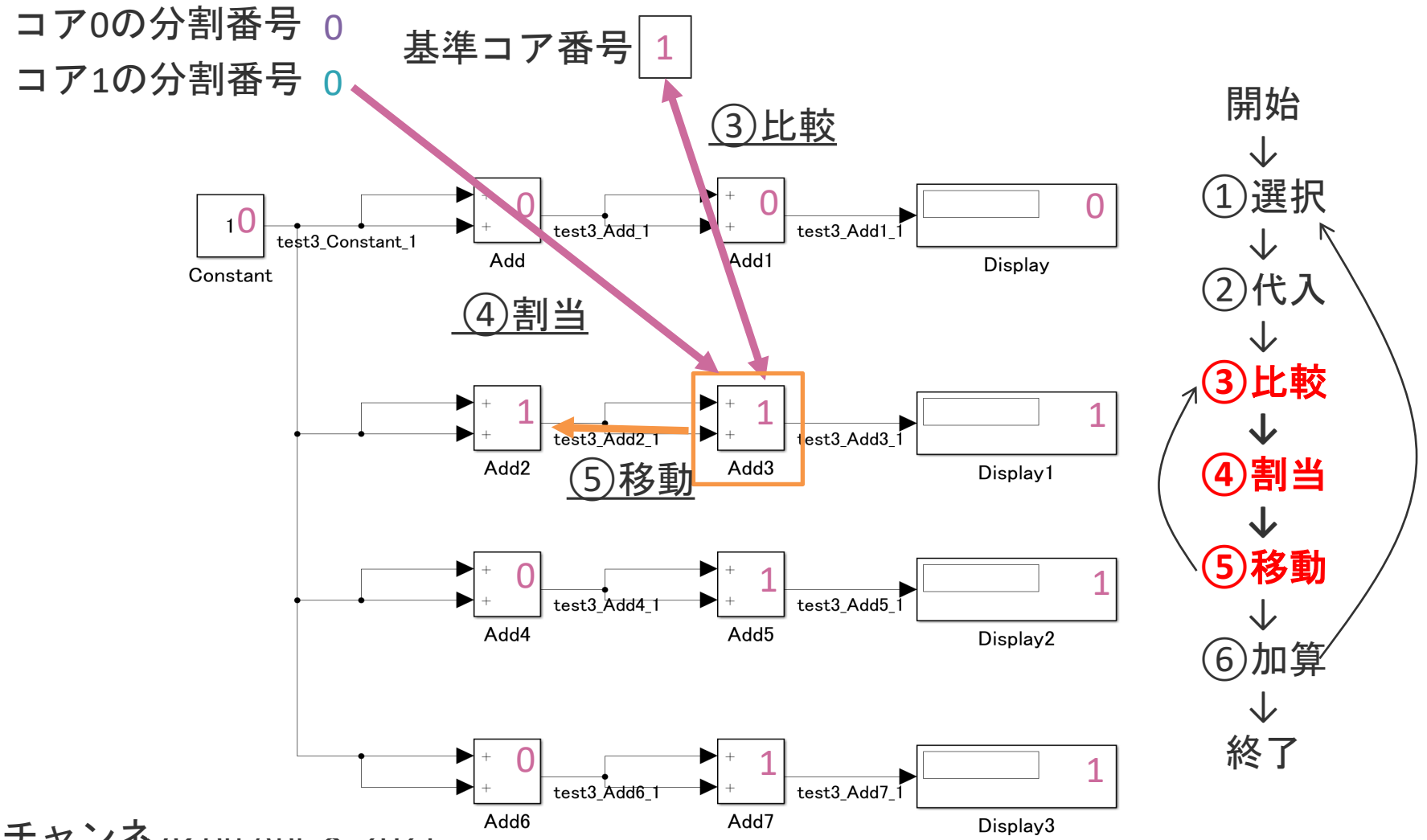
分割番号割当の例（1/6）

コア0の分割番号 0
コア1の分割番号 0

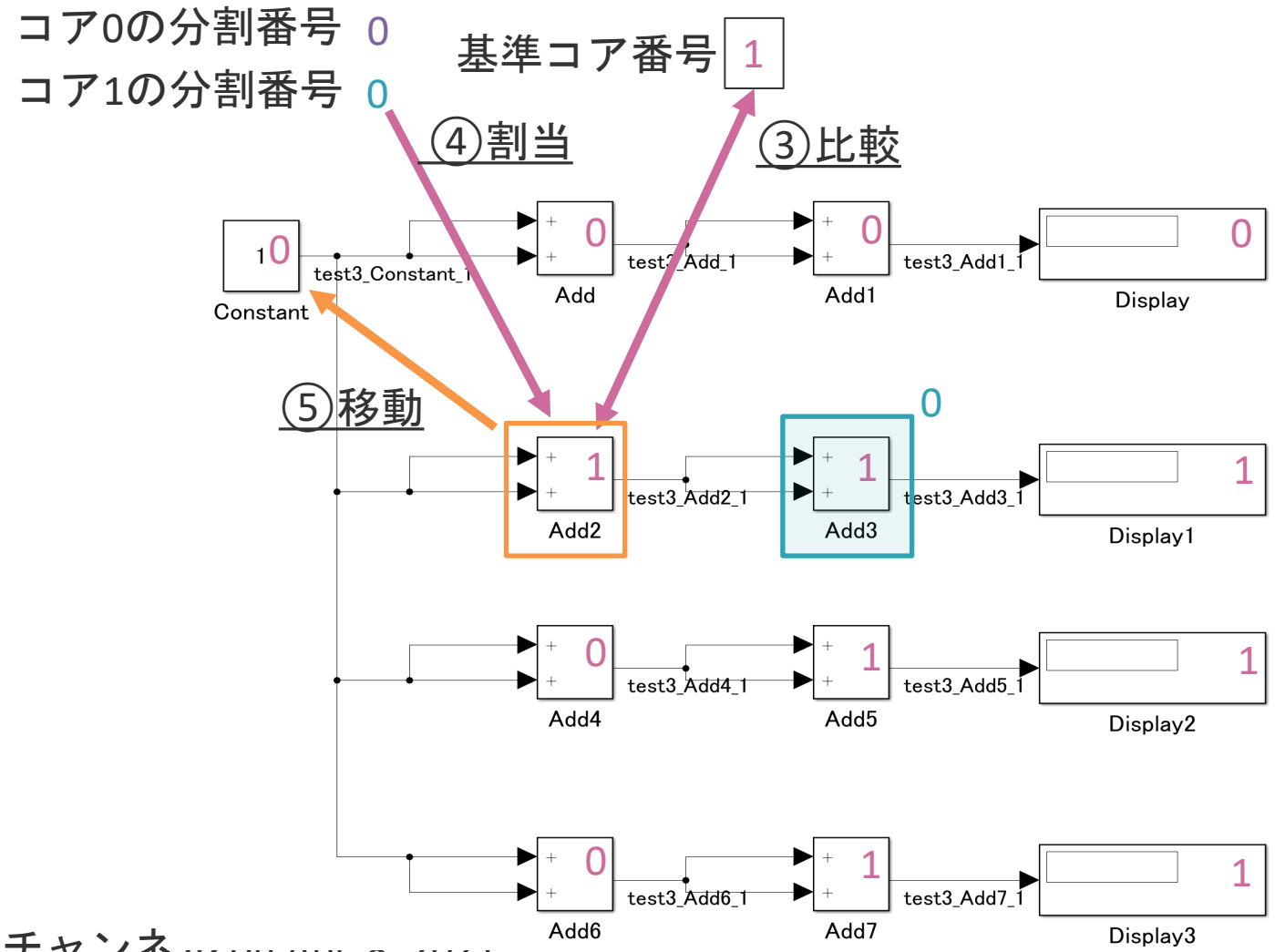


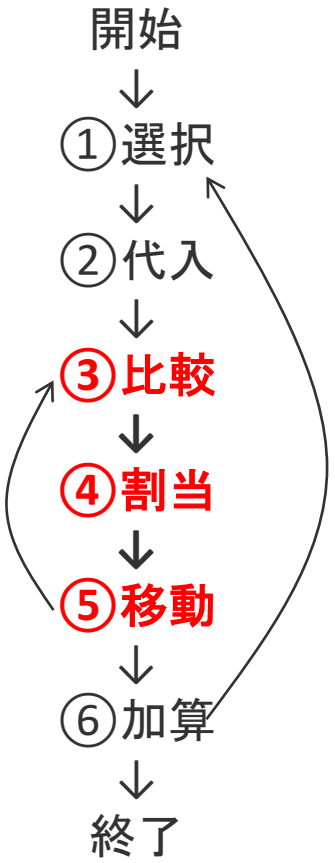


分割番号割当の例（2/6）

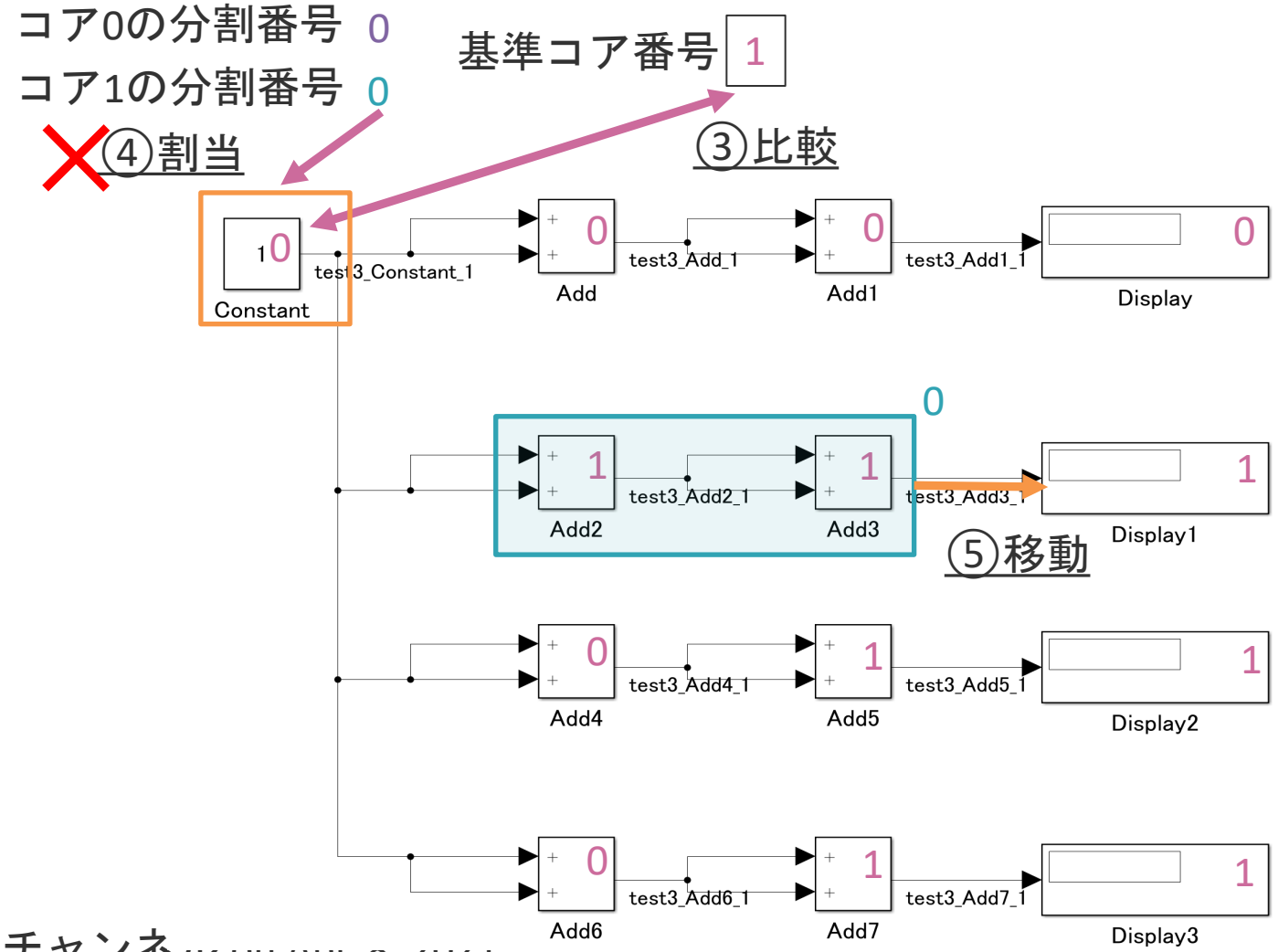


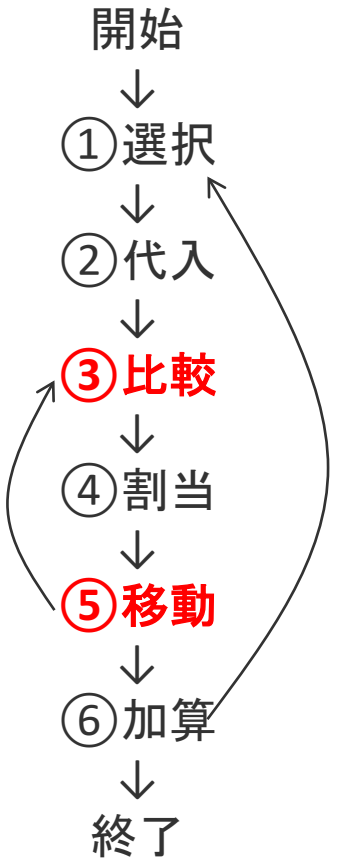
分割番号割当の例（3/6）



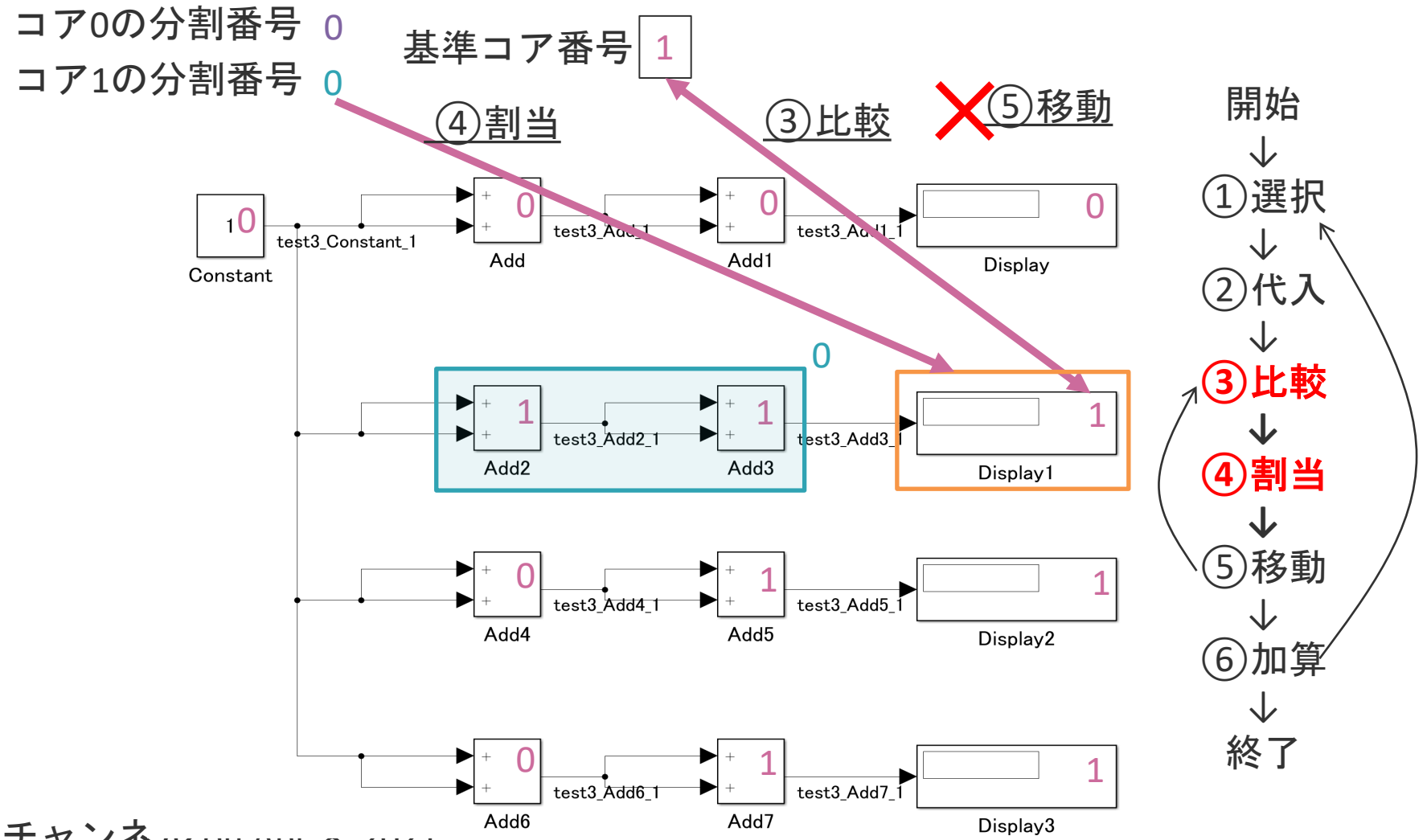


分割番号割当の例（4/6）





分割番号割当の例（5/6）



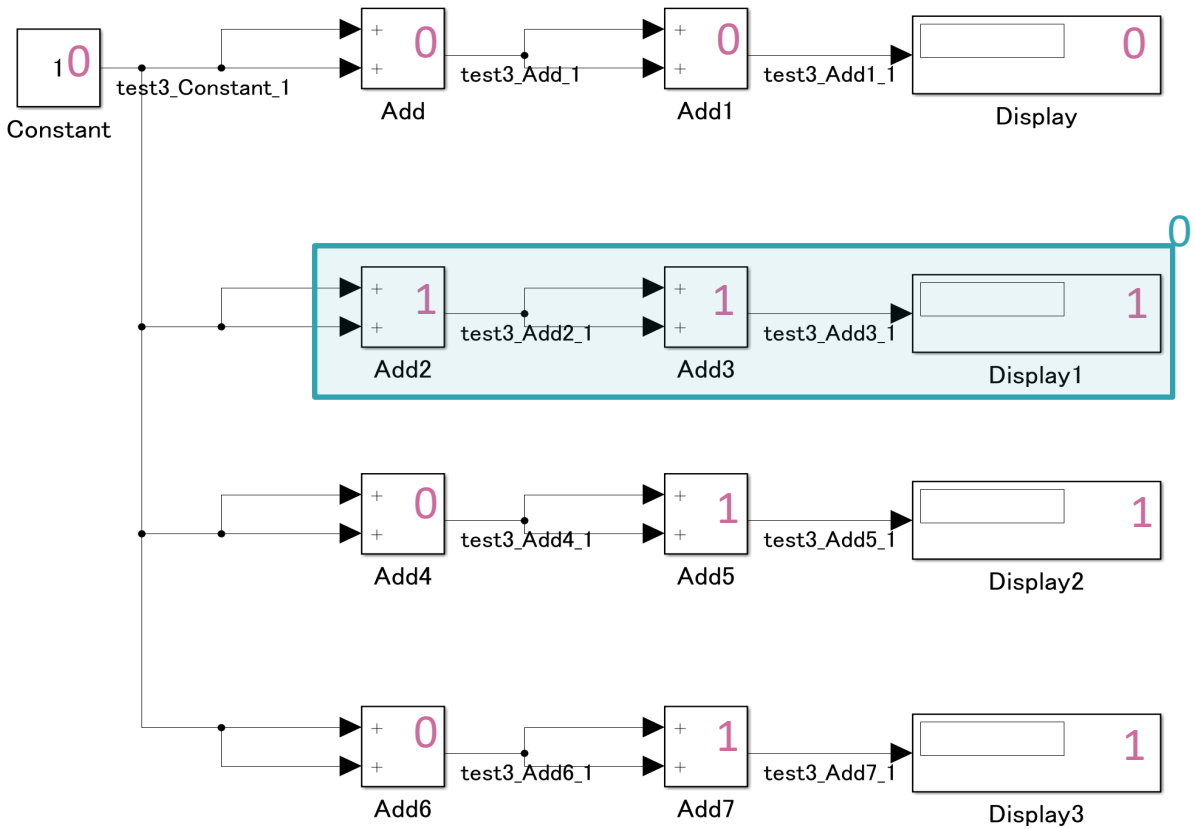
分割番号割当の例（6/6）

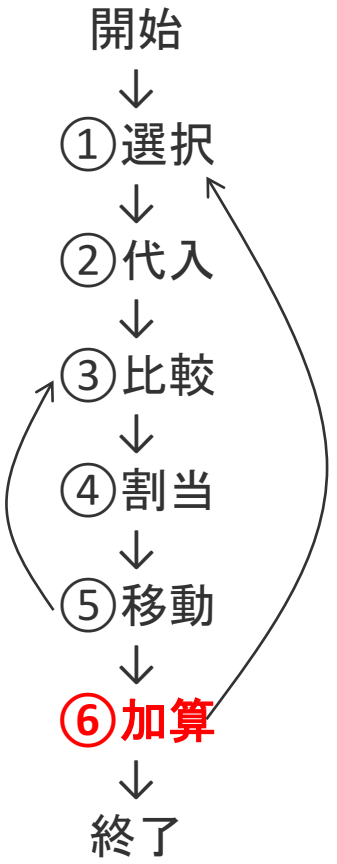
コア0の分割番号 0

コア1の分割番号 1

基準コア番号 1

⑥加算

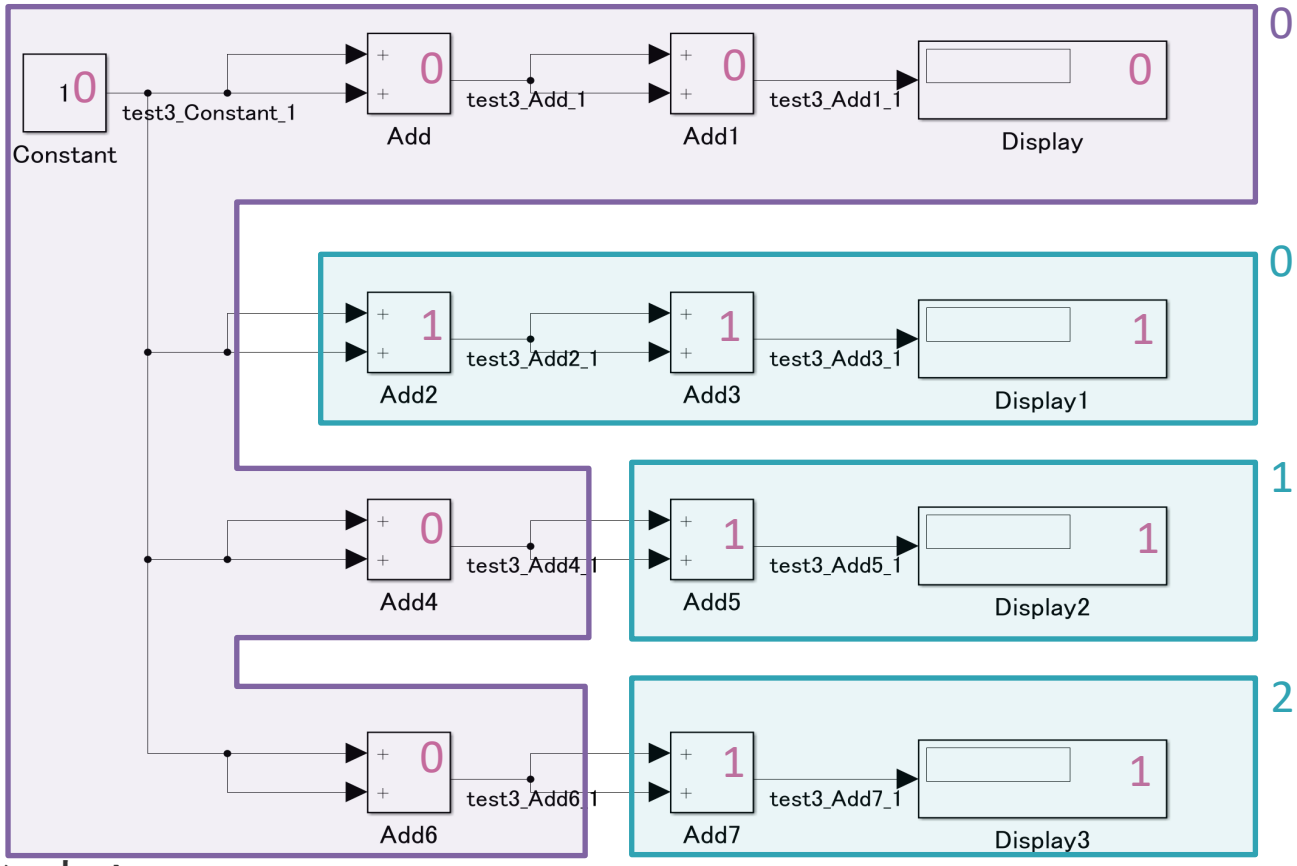


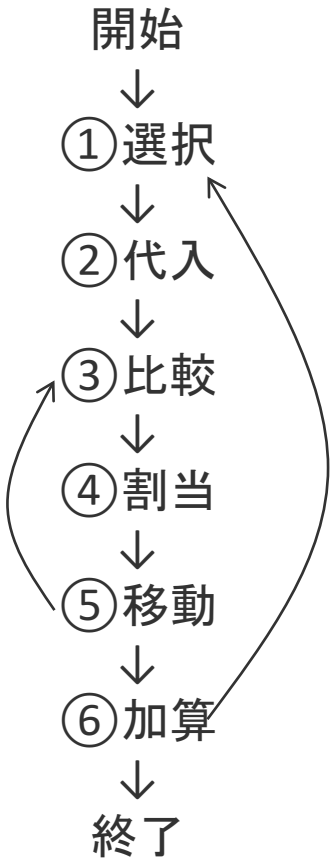


分割番号割当の結果

コア0の分割番号 1
コア1の分割番号 2

基準コア番号

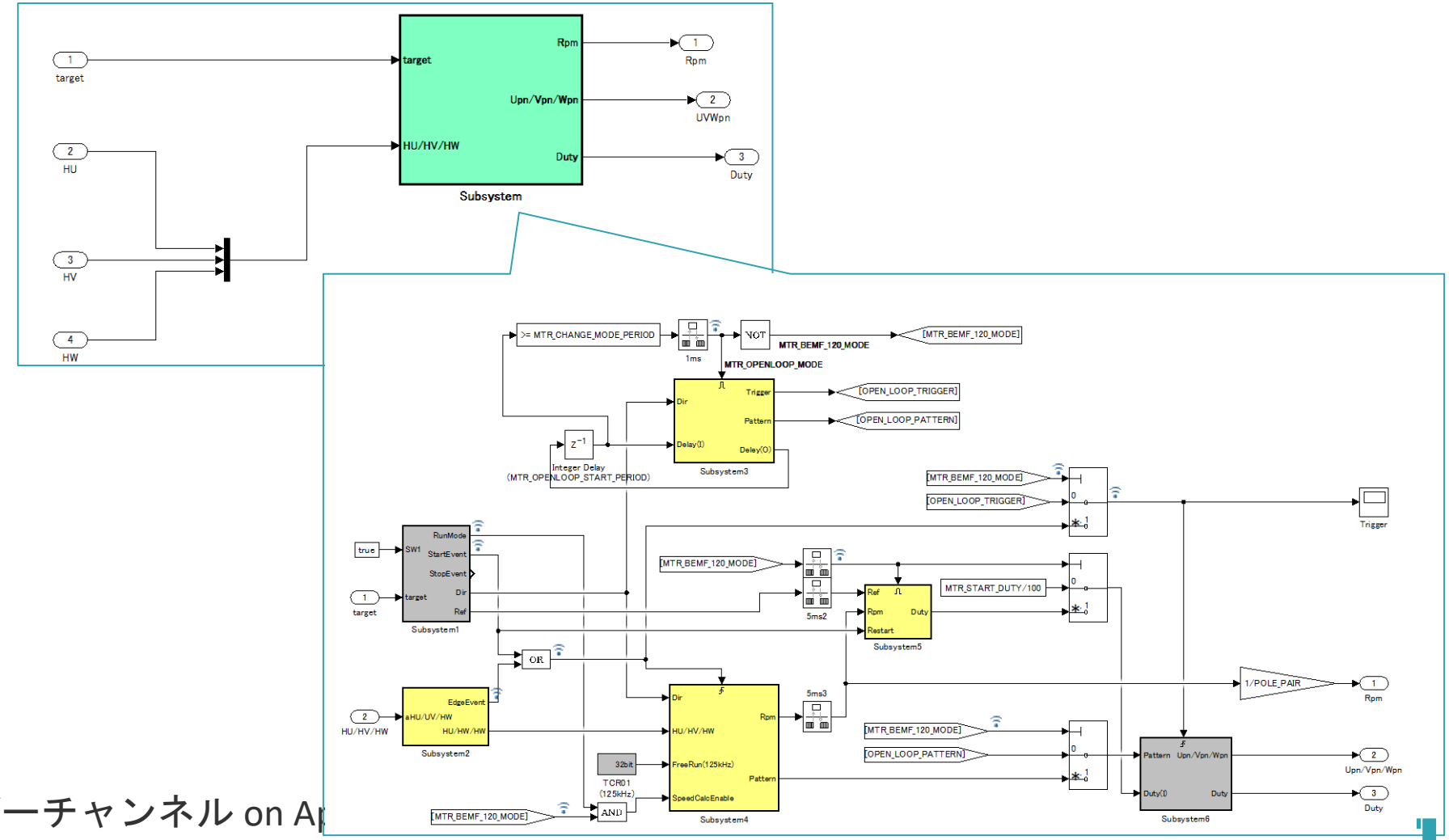




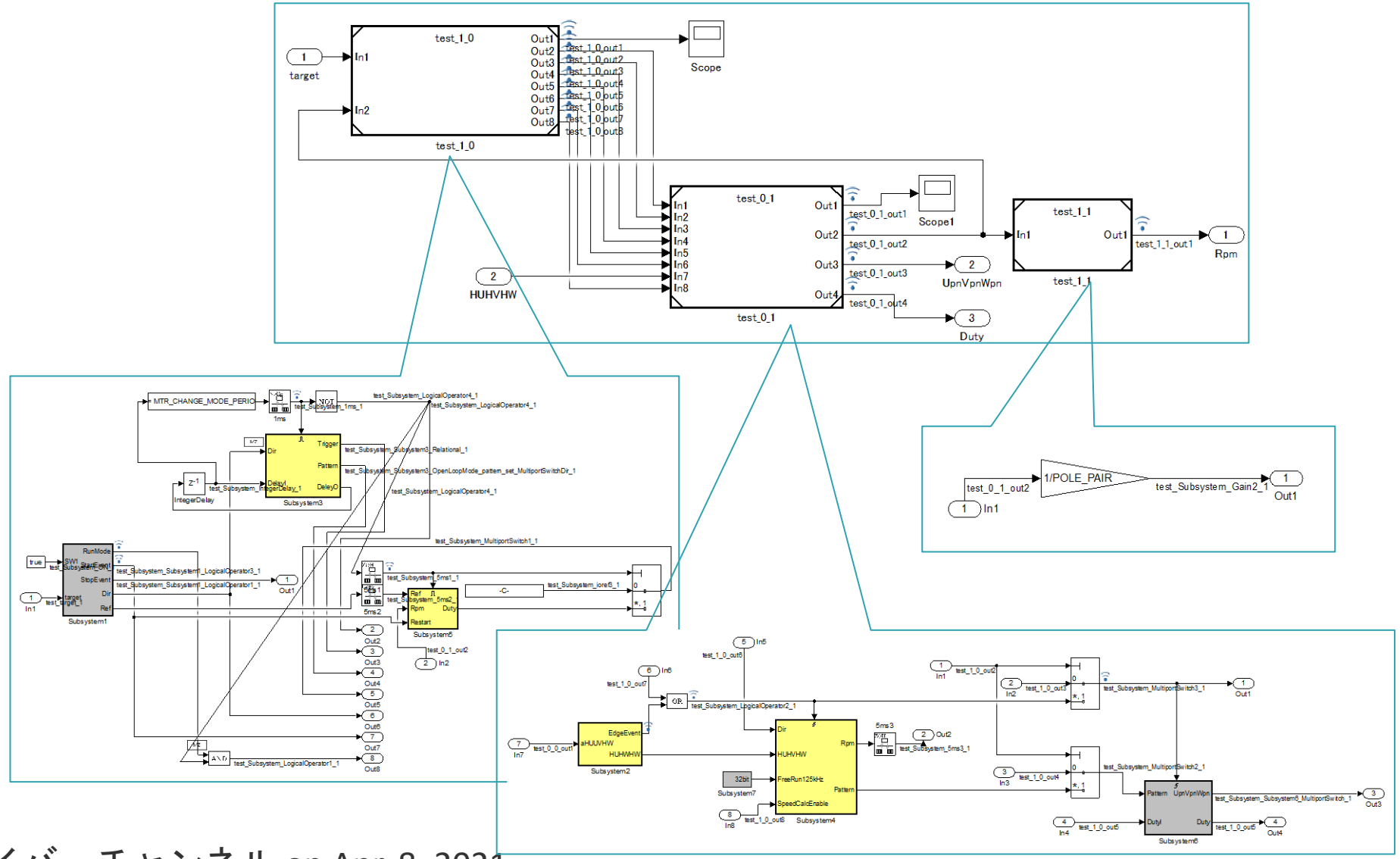
Gサイバーチャンネル on Apr. 8, 2021.

適用例

- 以下のSimulinkモデルでモデルレベルブロック分割手法を適用
- 使用したSimulinkモデルはモータ制御モデル



適用例のモデル分割結果



適用例の等価性検証

適用例における等価性検証:
「モデルの変更前後で同じ入力値に対して出力値が一致するかどうか」

テストケースが満たすべき要件

- 分割前後でモデルの振る舞いに差がないことを確認できる
  →モデル設計時に作成された動作確認用のテストケースを用意したい
- 分割前後でモデルの構造に差がないことを確認できる
  →テストケース生成ツール(*)を用いてテストケースを用意したい

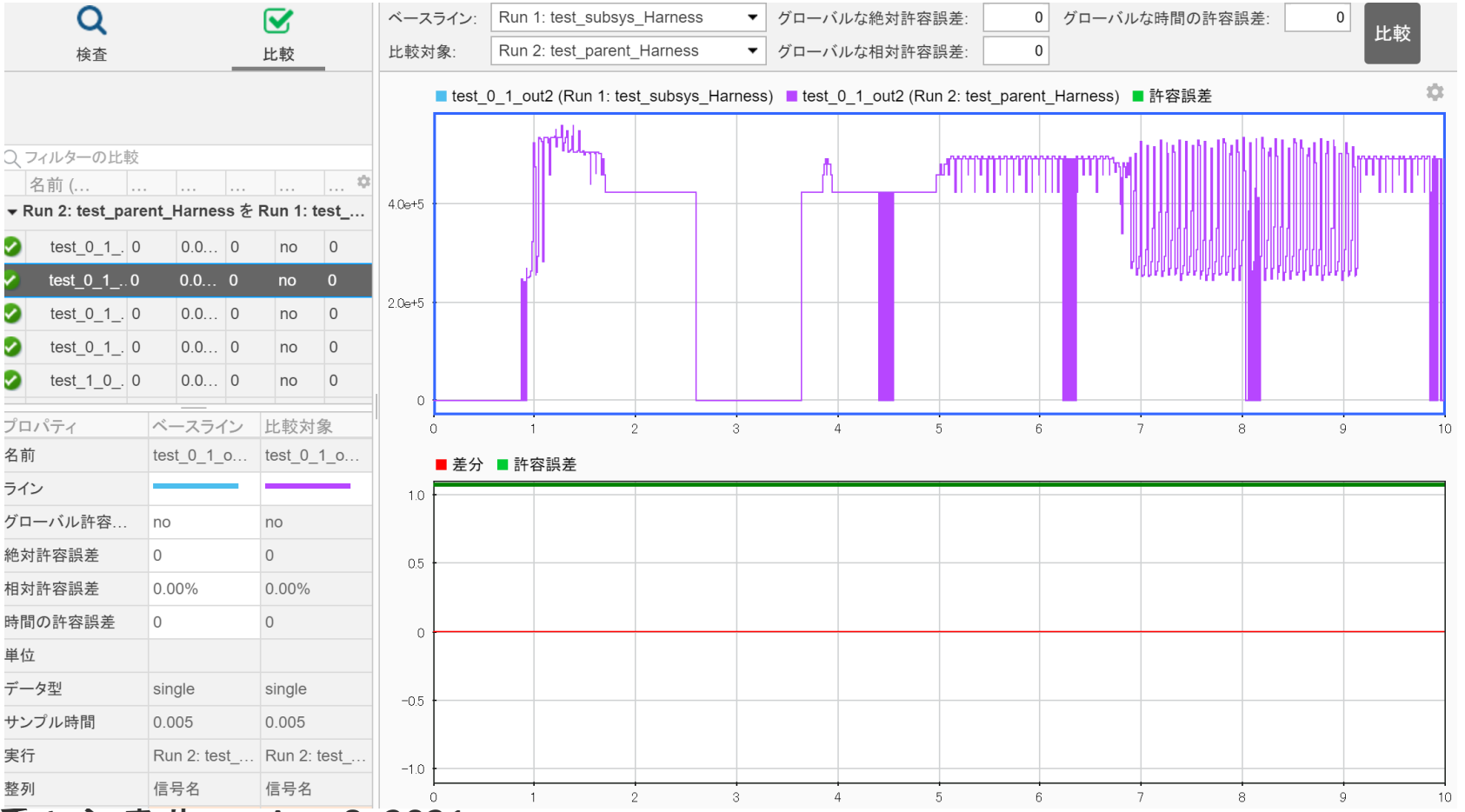
- 生成したテストケース
  - DC（判定カバレッジ） 120/123（3/123はデッドパス）
  - CC（条件カバレッジ） 70/70
  - MC/DC（改良条件判定カバレッジ） 13/13
- 今回のSimulinkモデルでは分割前後で生成したテストケースに対する出力値が一致

(*)ガイオ・テクノロジー株式会社/PROMPT(Performable RatiOnalized Model Perfective Tool)

Gサイバーチャンネル on Apr. 8, 2021.

適用例のMATLAB/sdi表示結果

- 分割前後それぞれのSimulinkモデルに対し、
  同じ入力データを使用したMATLAB/sdiの表示結果



Gサイバーチャンネル on Apr. 8, 2021.

目次

- 背景～どうしてモデルレベルでの並列化なのか～
- モデルベース並列化（前回セミナーの復習）
  - 並列化全体像
  - コア割当て
- マルチコア・メニーコア対応Simulinkモデル自動分割ツール
  - モデルベース並列化を実現するツールを活用
  - 振る舞いを変えずにSimulinkモデルを分割する
  - 適用例
- 適用事例の募集

Gサイバーチャンネル on Apr. 8, 2021.

適用事例の募集

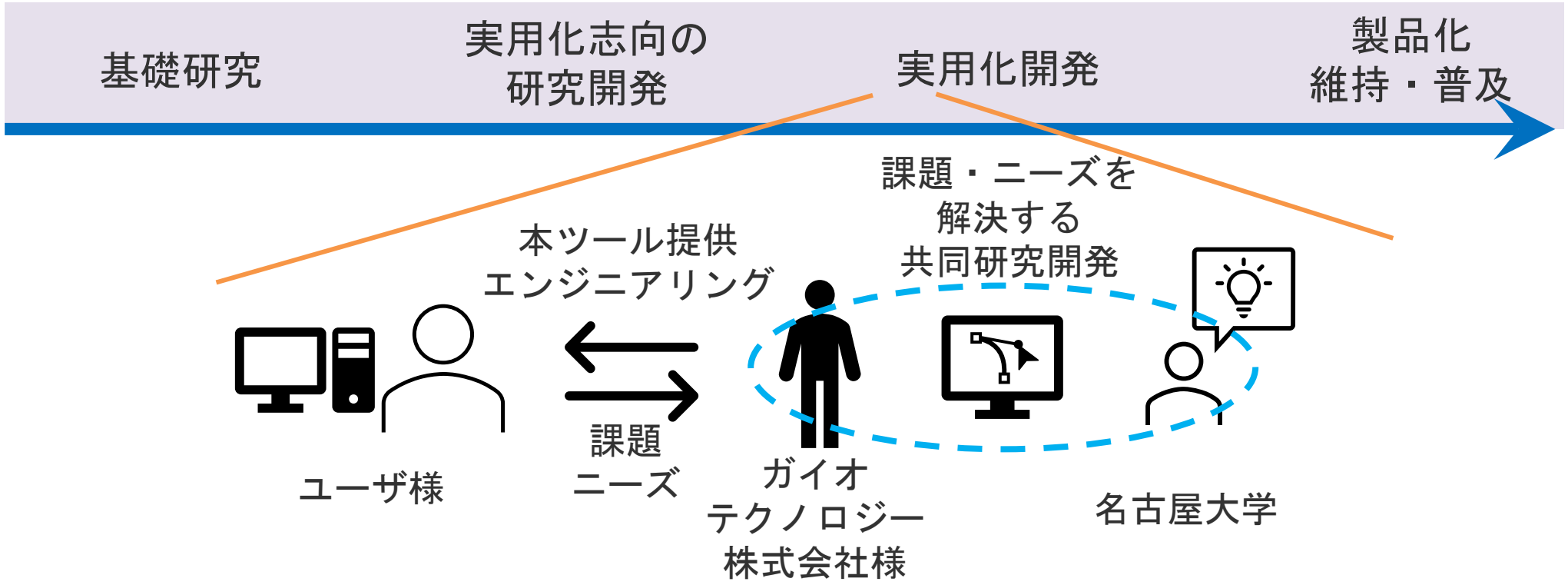
- いよいよ実用化開発の段階に入りつつあると考えております。
- 共同研究先のガイオ・テクノロジー株式会社様よりユーザ様へツール提供を行い、フィードバックによる実用化開発を行っていきたいと考えております。
- 次は第２部として、適用事例募集の詳細について、ガイオテクノロジー社の方からの講演となります。

適用事例募集全体像

Gサイバーチャンネル on Apr. 8, 2021.

適用事例募集の詳細(ガイオ・テクノロジー株式会社から)

- お問い合わせ先：【ヘルプデスク担当　三宮】【Emailアドレス：seminar-help@gaio.co.jp】



適用事例募集全体像

ご清聴ありがとうございました

Gサイバーチャンネル on Apr. 8, 2021.