FPGA/マルチコアCPUを活用したプラントモデル並列化技術について

＜概要＞
近年、Model-Based Design活用の拡大や開発のフロントローディングが進むなかで、制御対象であるプラントモデルは大規模化・詳細化する傾向にあり、シミュレーション時間の増加が課題となっている。本講演では、高速化の手法として、Simulink®およびSimscape™で作成したプラントモデルの物理的構造に着目した並列化方法（マルチコアCPU向け、および、FPGA実装向けのモデリング）について紹介する。事例として、FPGAにプラントモデルを実装（HDL Coder™使用）した場合、CPU実行に比べ数百倍の高速化が可能となった例を紹介する。

2017.10.31
トヨタテクニカルディベロップメント（株）
シミュレーション事業部
川口　晃

TTDC会社紹介

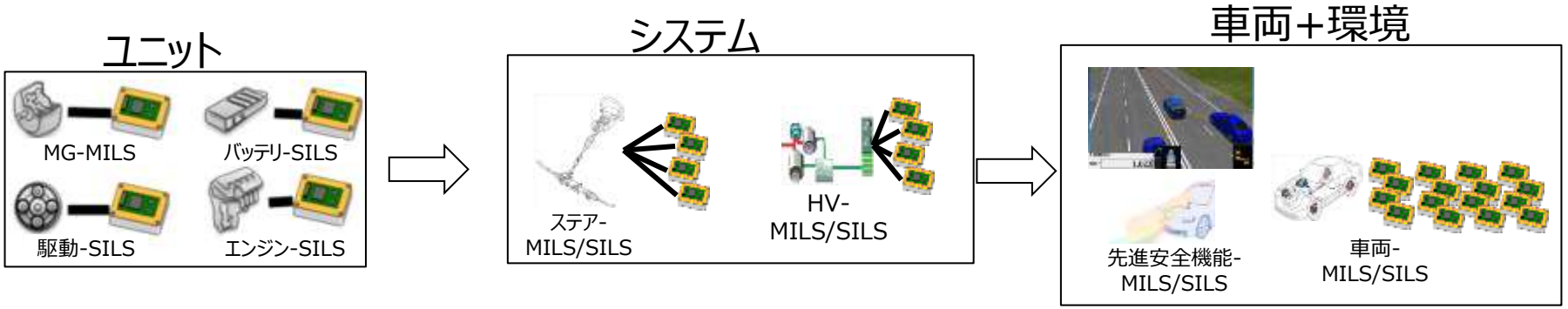
会社名	トヨタテクニカルディベロップメント株式会社 (TTDC)
会社名(英文)	Toyota Technical Development Corporation
設立	2006年4月1日設立
従業員数	950名　※2017年4月1日時
本社	愛知県豊田市花本町井前１番地９
事業内容	知的財産（IP）事業 ・調査・技術動向解析 ・外国出願・権利化（特許・意匠・商標） ・翻訳/通訳 計測制御事業 ・計測機器・装置の開発/製作 ・装置・設備の企画/計画立案 ・モデルベース開発ソリューションの提案/提供 ・計測機器の校正/検査/修理 ・次世代事業の開発支援

目次

1. 背景

■ 車両制御機能の大規模化…

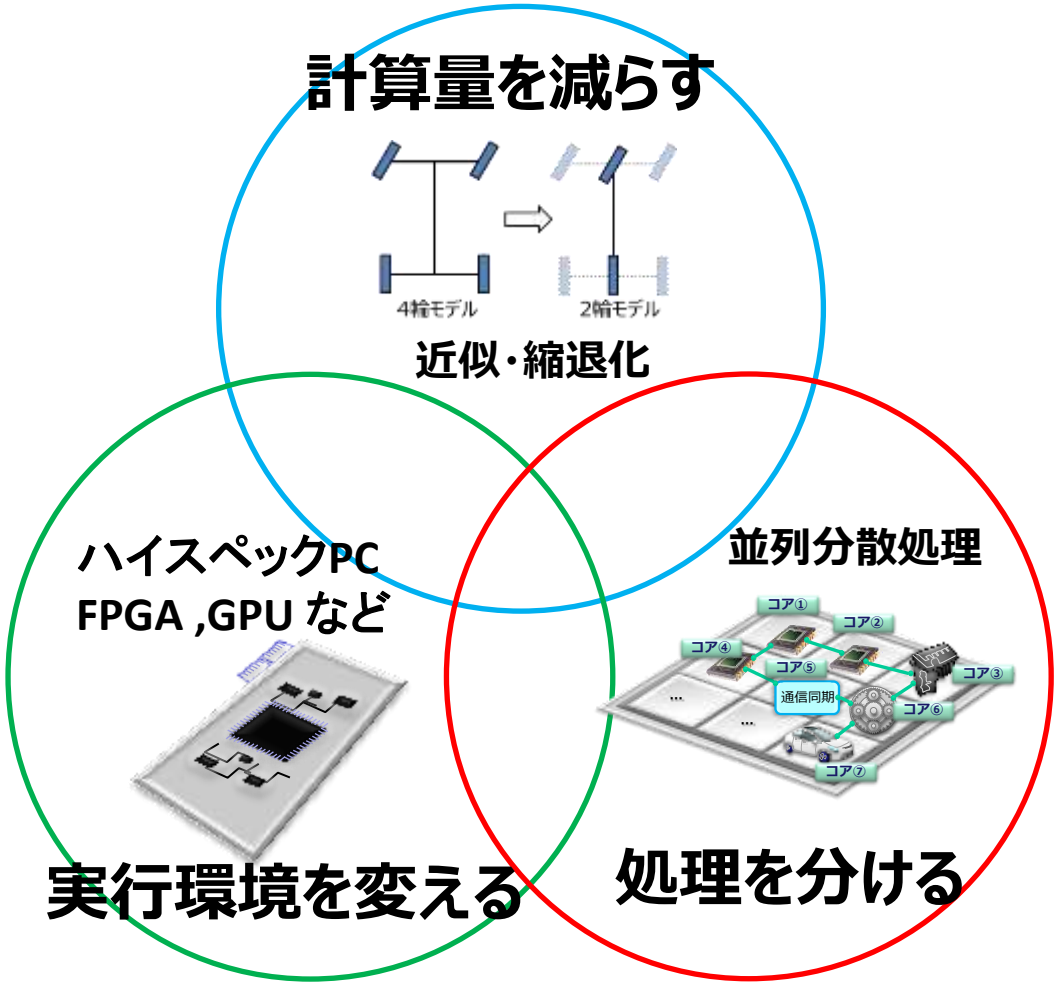
自動運転制御、ADASなど車両制御機能が増大しており、
システム単位から車両全系へのシミュレーションニーズが増加
⇒ モデルが大規模化し、シミュレーション負荷が重く、
車両開発での実用は困難



シミュレーション規模が増大し、
高速化のニーズが高まってきた。

2.シミュレーション高速化について

シミュレーションを高速化したい…



＜本講演の範囲＞

①実行環境
⇒マルチコアCPUと
FPGA

②処理の対象
⇒プラントモデルを
並列処理
（並列モデリング手法）

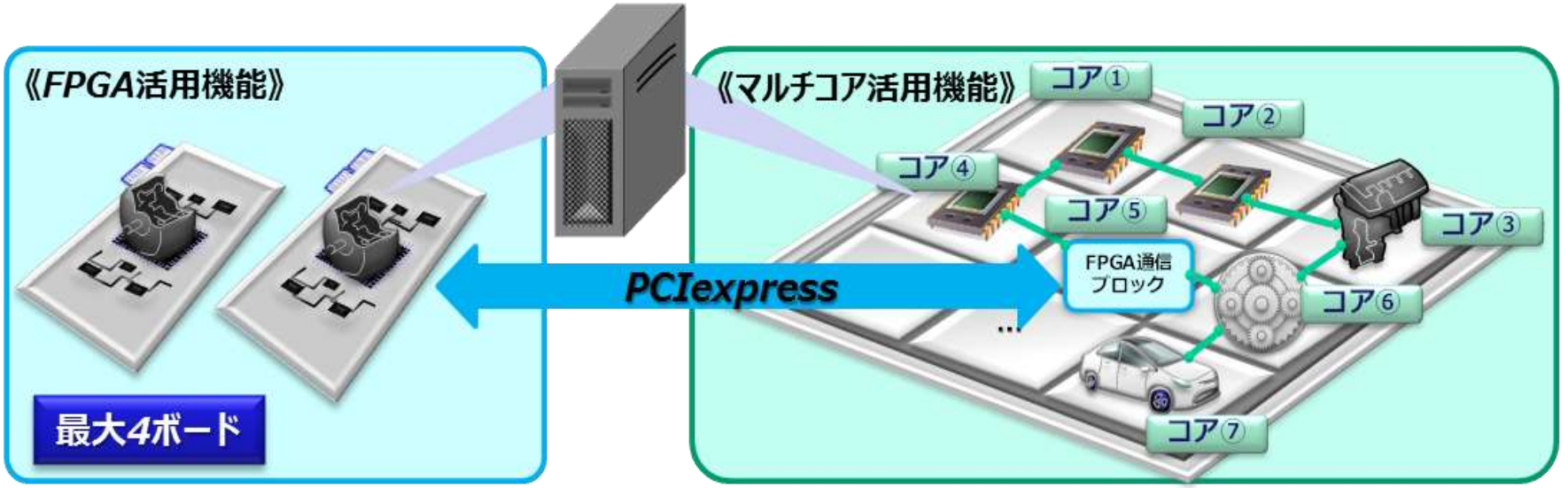
について紹介

FPGA (Field programmable gate array)
GPU (Graphics processing unit)
ベクトルプロセッサ

2.シミュレーション高速化について

■実行環境の概要

FPGAとマルチコアCPUを活用した並列シミュレーション システム



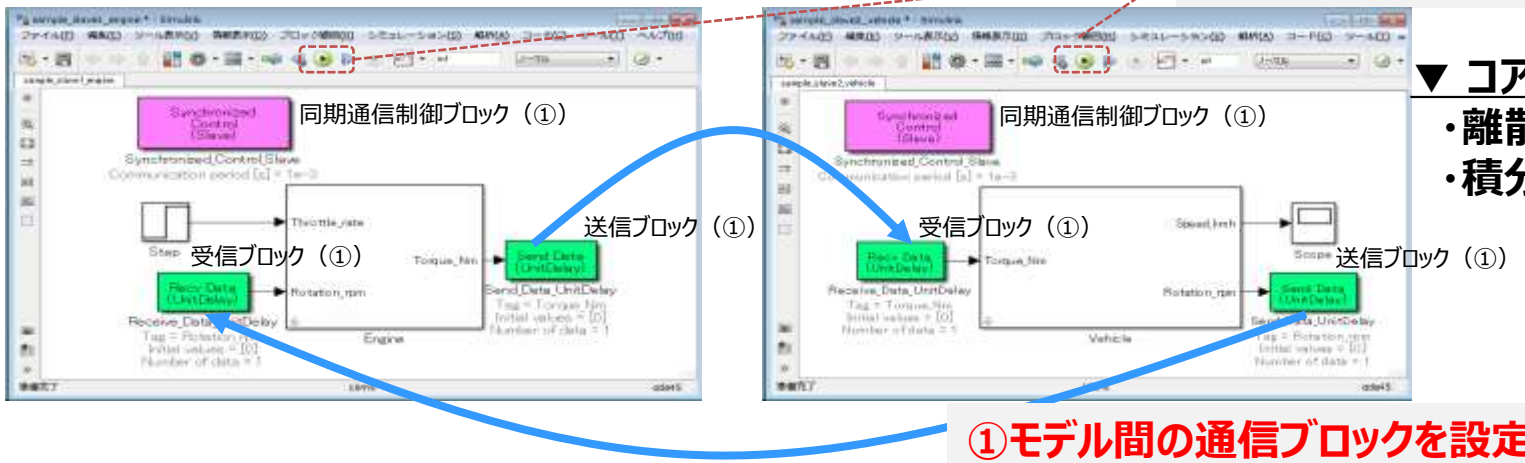
高速処理必要なモデルを**FPGAで演算**

大規模化するシステムを**並列演算で高速処理**

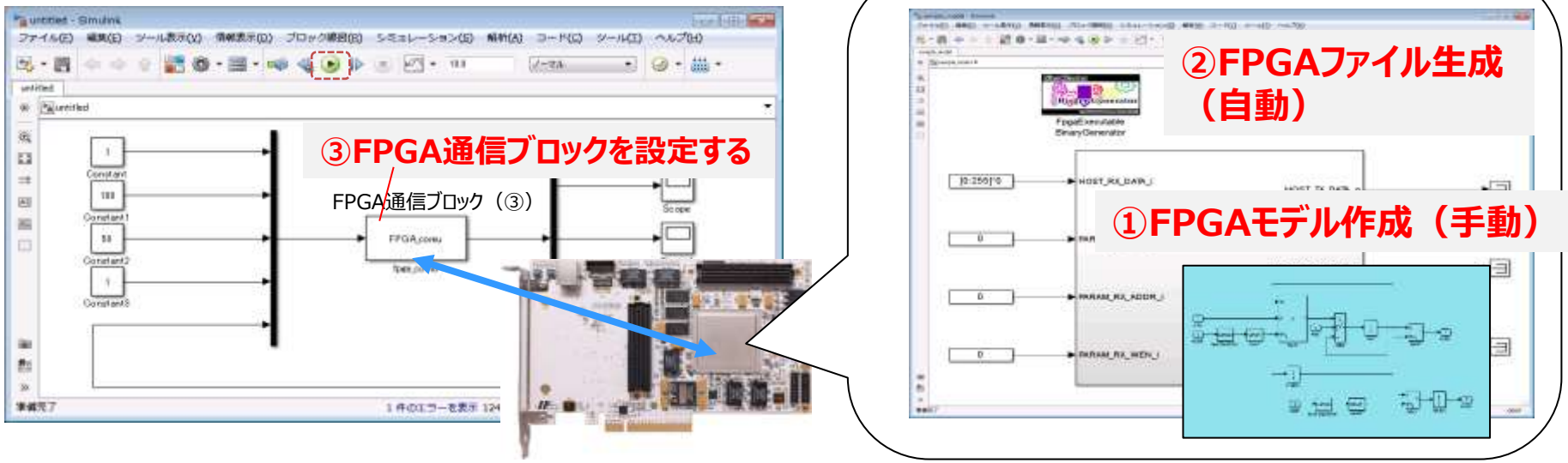
CPUコア⇔CPUコア、CPUコア⇔FPGAの通信は、
低レイテンシかつステップ同期通信

2.シミュレーション高速化について

マルチコア並列イメージ



FPGA動作イメージ

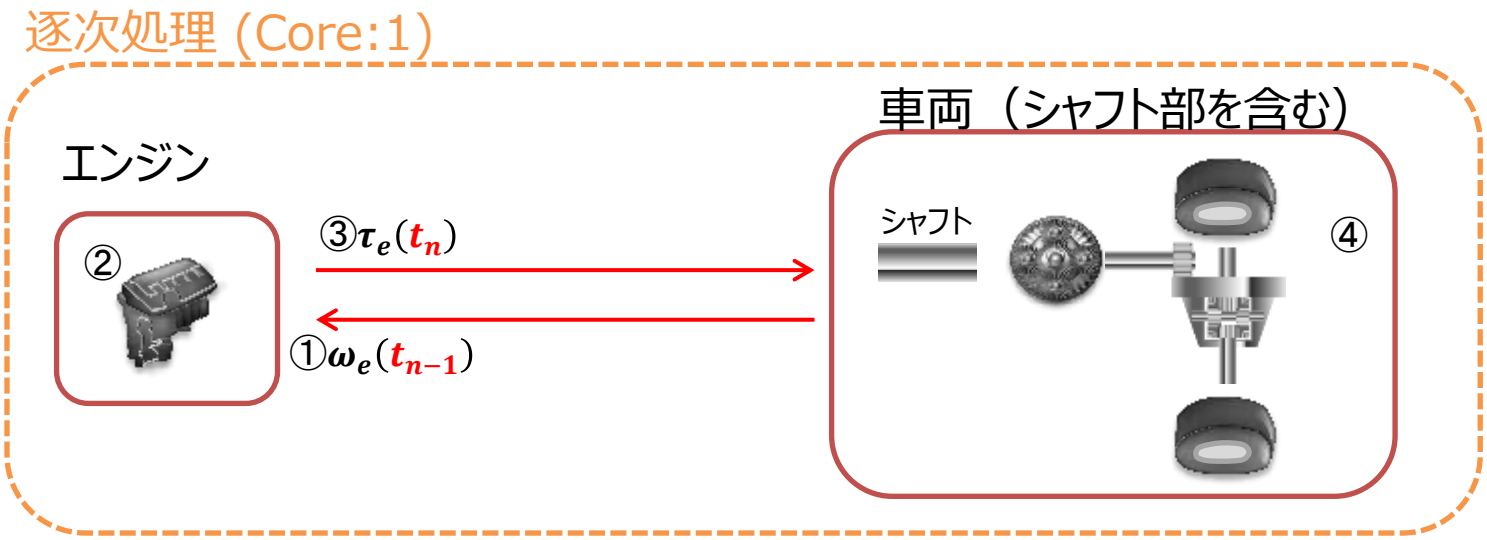


3. 物理モデルのマルチコア並列化

3.1　物理モデル並列化の注意点

- データ授受のタイミング

ユニット間（エンジン・車両）で並列化すると通信による
遅延（1ステップ）の影響が発生

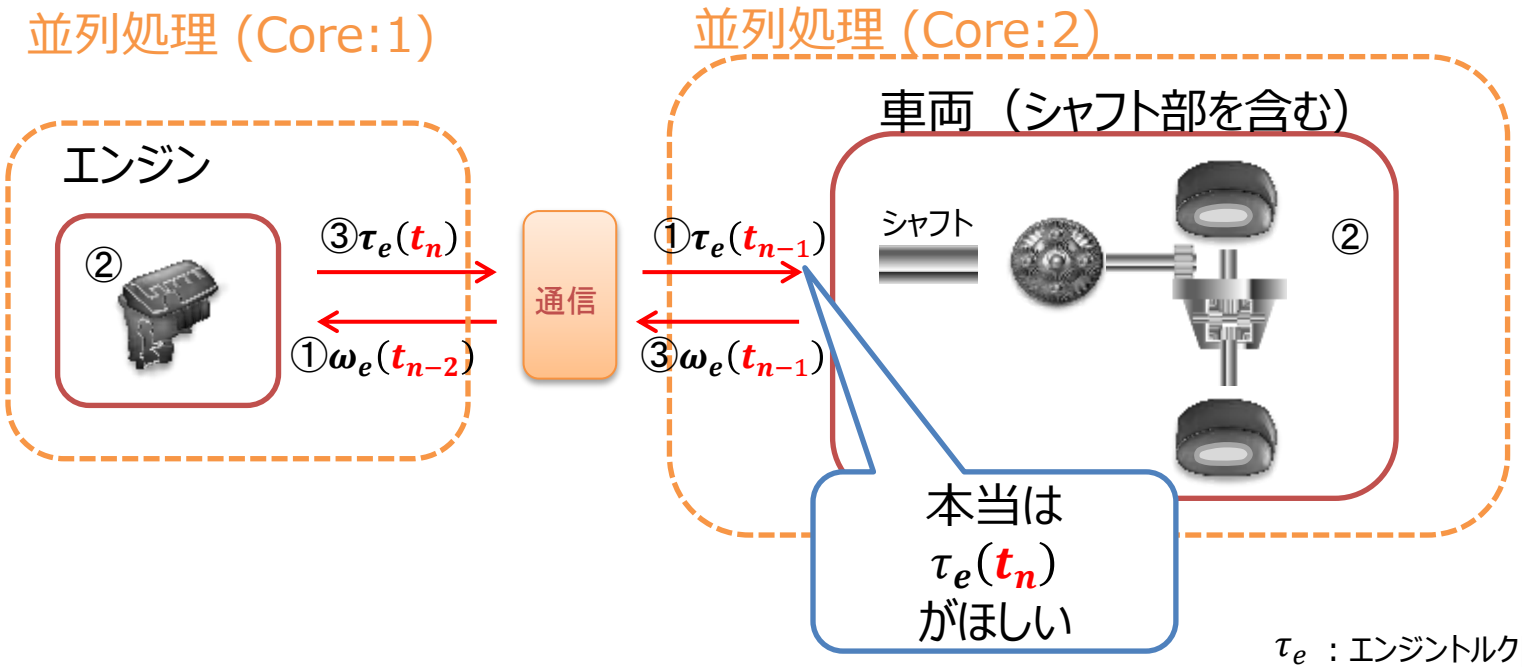


τ_e ：エンジントルク[Nm]
τ_i ：TM負荷トルク[Nm]
ω_e ：エンジン角速度[rad/s]
J_e ：エンジンイナーシャ[rad/s]

3.1 物理モデル並列化の注意点

- データ授受のタイミング

ユニット間（エンジン・車両）で並列化すると通信による
遅延（1ステップ）の影響が発生

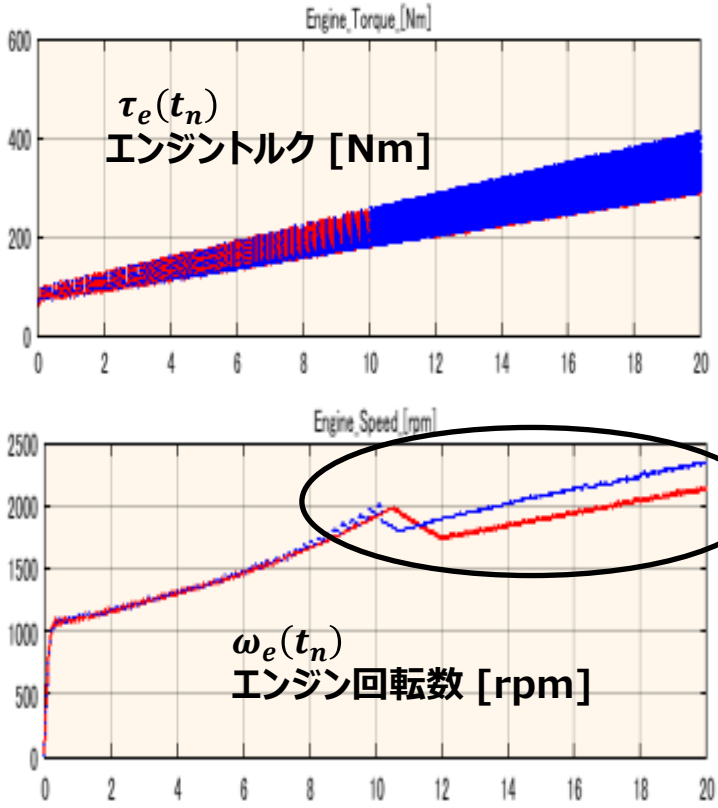


逐次処理と比較すると
データ授受に1ステップのズレ発生

τ_e ：エンジントルク[Nm]
τ_i ：TM負荷トルク[Nm]
ω_e ：エンジン角速度[rad/s]
J_e ：エンジンイナーシャ[rad/s]

3.1　物理モデル並列化の注意点

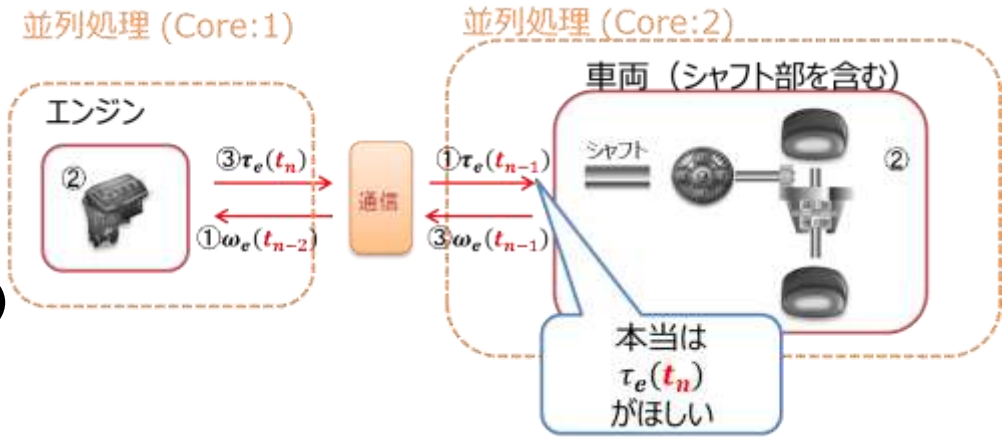
簡易のコンベ車にてシミュレーション精度の影響を比較　（ｽﾃｯﾌﾟｻｲｽﾞ：0.5ms / ｼﾐｭﾚｰｼｮﾝ時間：0 - 20s）



《凡例》

① 逐次処理　（左グラフ：　　）

② 並列処理　（左グラフ：　　）



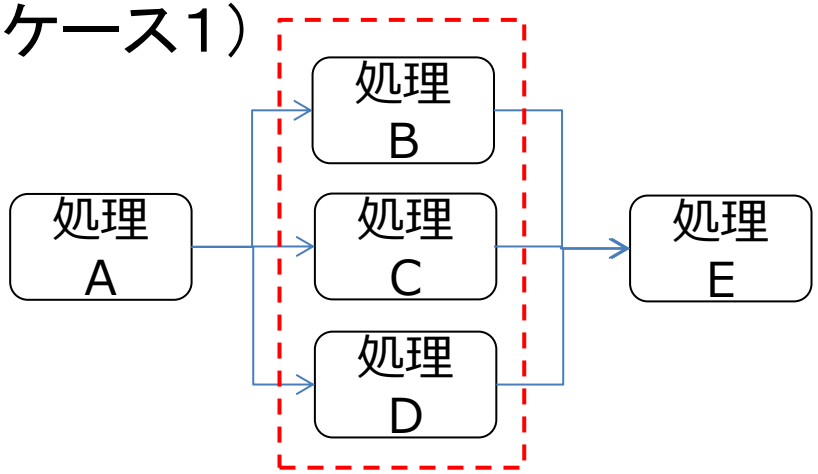
・ステップずれの影響 ⇒ 演算が進む毎に蓄積

・高周波成分（左グラフ：エンジントルク）が含まれると
　ステップずれの影響が大きい

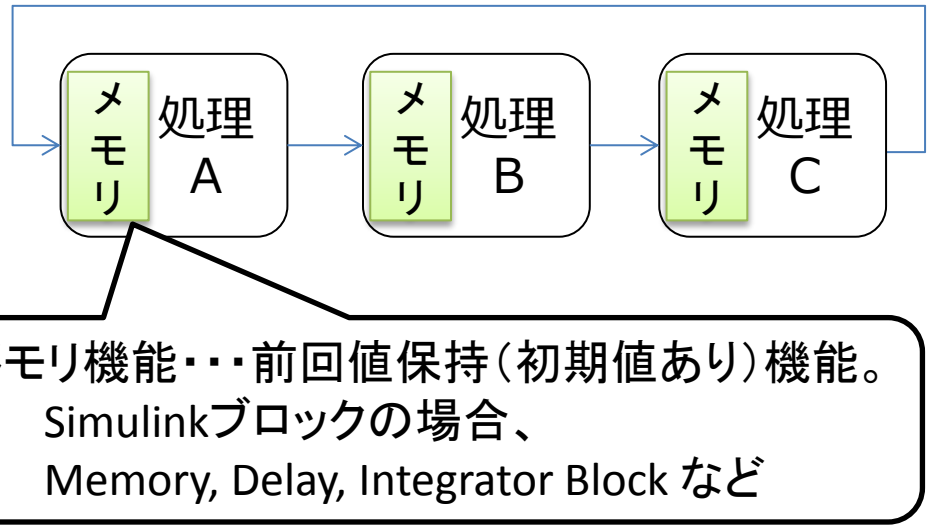
3.2 並列処理のポイント

■タイミングのズレなく並列処理できるケース

（ケース1）



（ケース2）



＜並列化のポイント＞

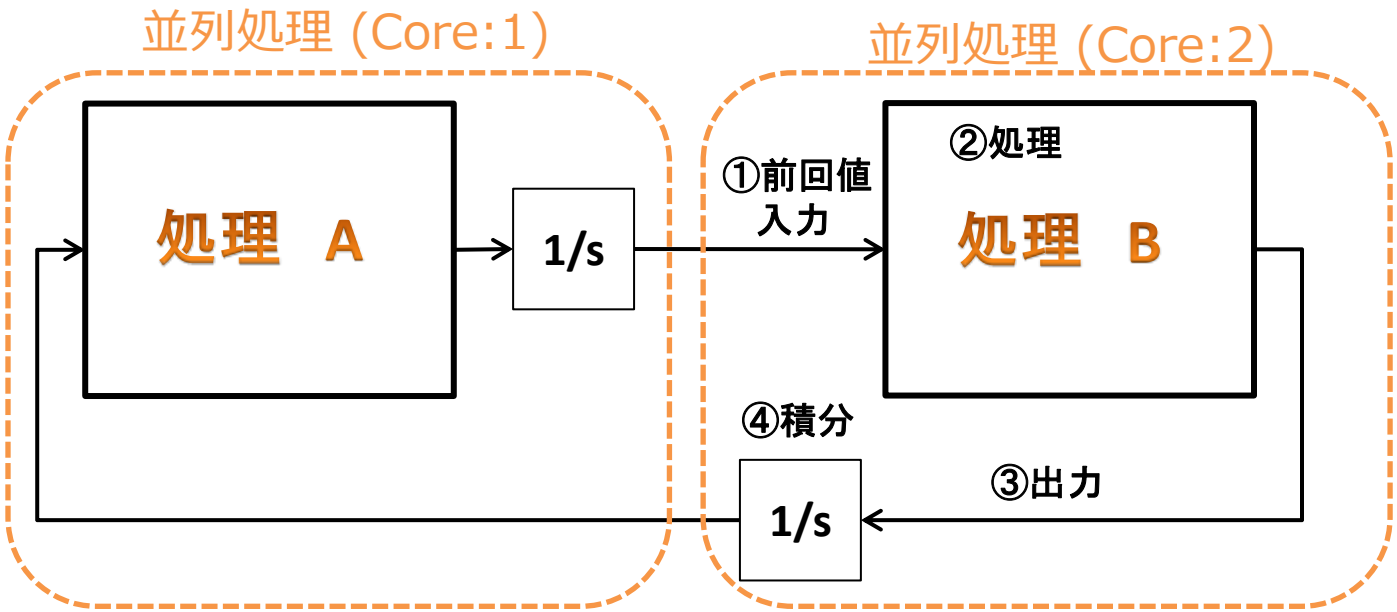
①処理間で直接依存関係がない。（ケース1の赤枠）

or

②入力が前ステップの値である。同一ステップ内で
依存関係がない（ケース2の処理A,B,C）

3.2 並列処理のポイント

■Simulinkモデルの分割例



＜並列化のポイント＞

SimulinkモデルのUnit Delay や Integrator がある箇所で分割できる。

↓

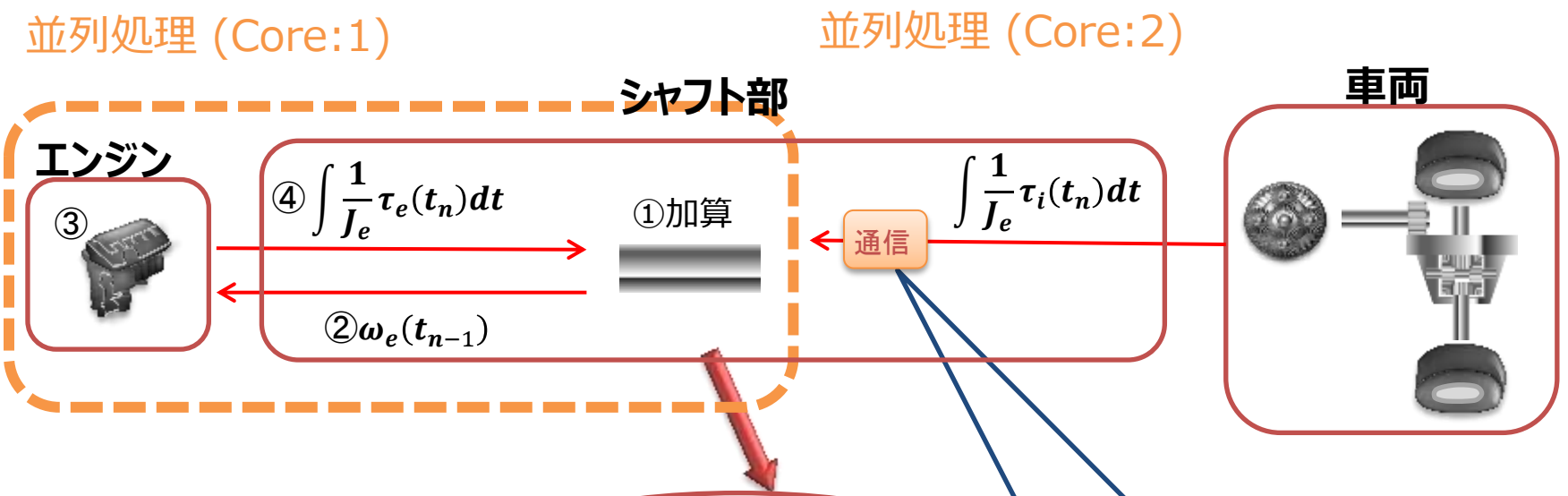
回転運動系の場合、シャフト（イナーシャ）部の積分部で分割できる。

$$\omega_e(t_n) = \int \frac{1}{J_e}\tau_e(t_n)dt + \int \frac{1}{J_e}\tau_i(t_n)dt$$

3.3 物理構造を活用した並列演算手法

特許出願中

車両モデルの動力伝達箇所であるシャフト部の積分処理を活用



$$\omega_e(t_n) = \int \frac{1}{J_e}(\tau_e(t_n) + \tau_i(t_n))dt = \int \frac{1}{J_e}\tau_e(t_n)dt + \int \frac{1}{J_e}\tau_i(t_n)dt$$

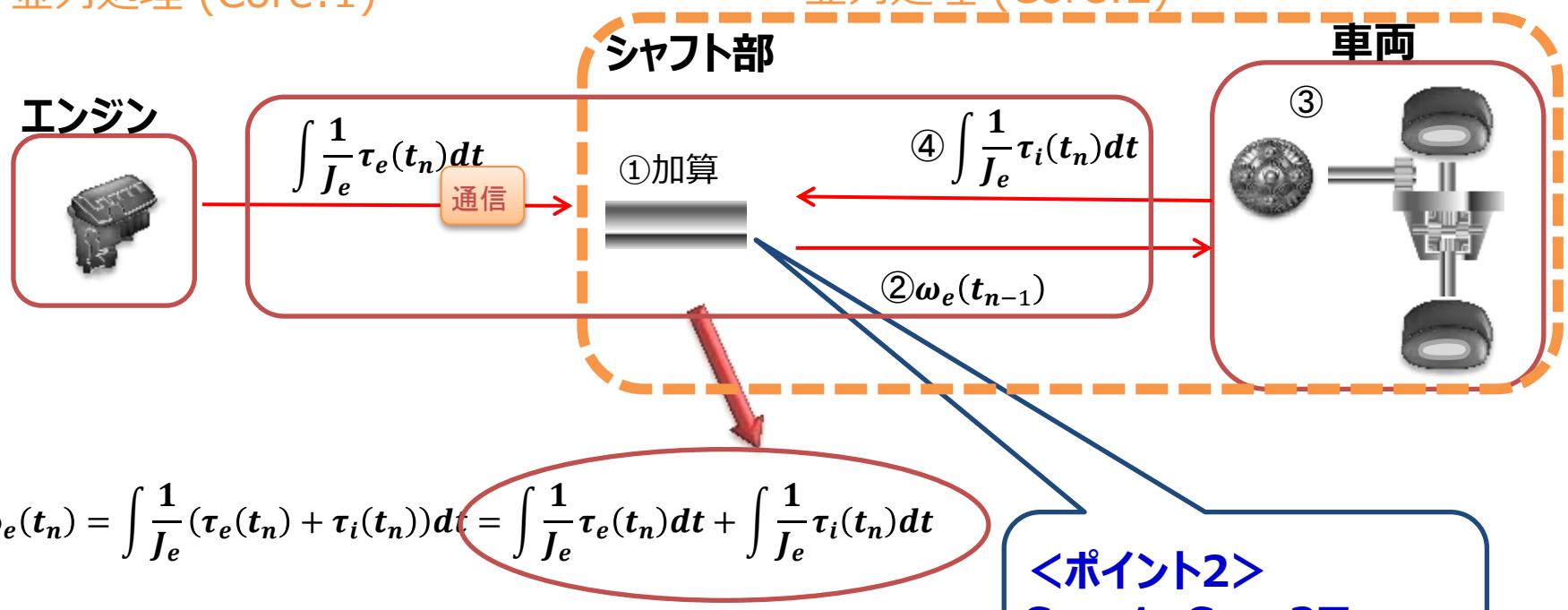
τ_e ：エンジントルク[Nm]
τ_i ：TM負荷トルク[Nm]
ω_e ：エンジン角速度[rad/s]
J_e ：エンジンイナーシャ[rad/s]

<ポイント1>
トルクの積分値を通信
⇒初期値（前回値保持）を持つので、タイミングずれなし

3.3 物理構造を活用した並列演算手法

特許出願中

車両モデルの動力伝達箇所であるシャフト部の積分処理を活用



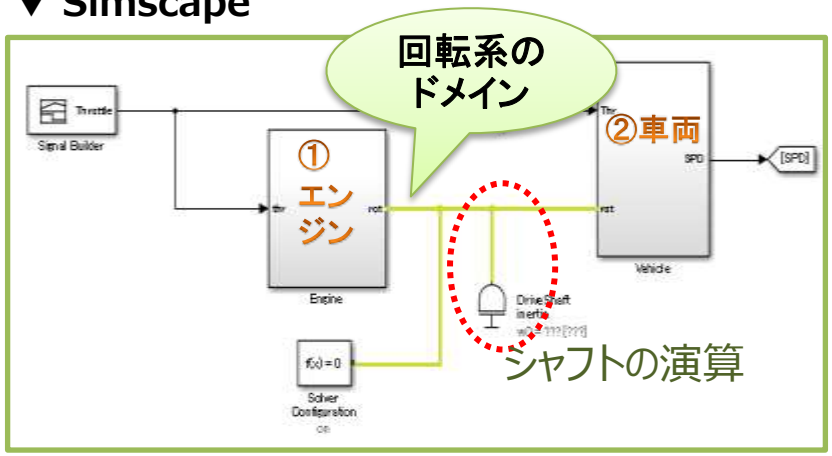
$$\omega_e(t_n) = \int \frac{1}{J_e}(\tau_e(t_n) + \tau_i(t_n))dt = \int \frac{1}{J_e}\tau_e(t_n)dt + \int \frac{1}{J_e}\tau_i(t_n)dt$$

<ポイント2>
Core1, Core2で
両方で同じ加算を実施

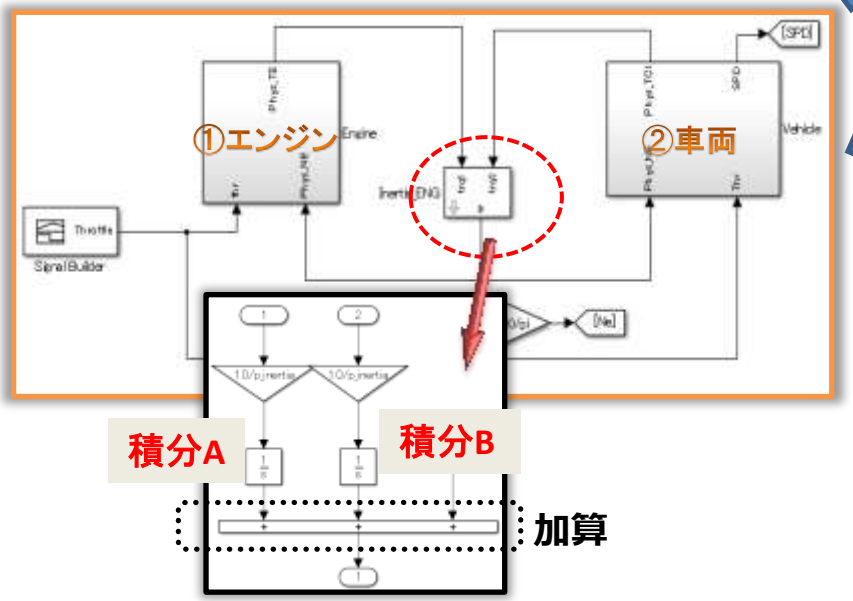
τ_e ：エンジントルク[Nm]
τ_i ：TM負荷トルク[Nm]
ω_e ：エンジン角速度[rad/s]
J_e ：エンジンイナーシャ[rad/s]

3.3 物理構造を活用した並列演算手法

▼ Simscape

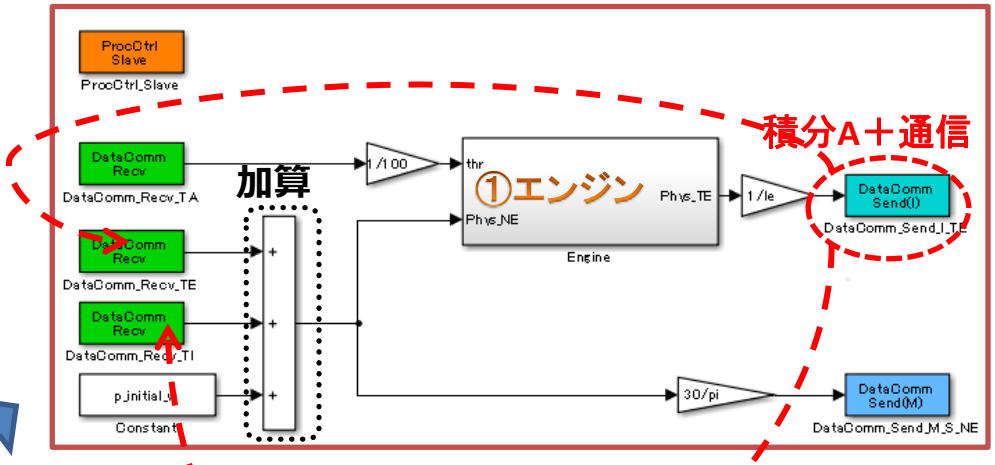


▼ Simulink

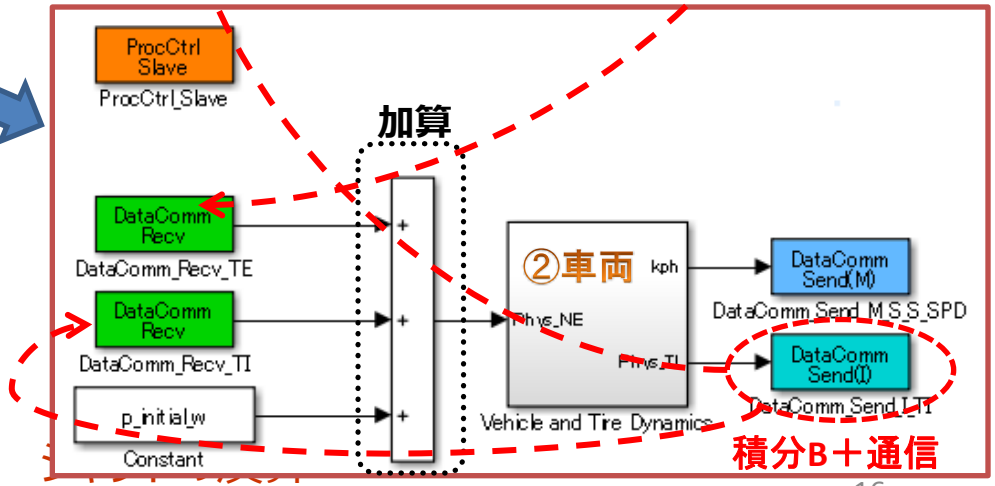


▼ 並列モデル

① エンジン（Core:1）



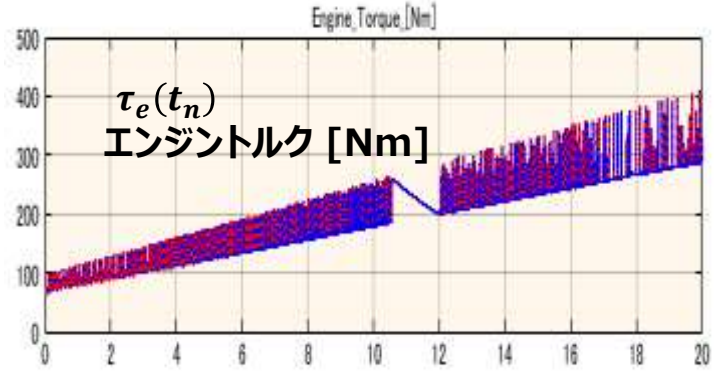
②車両（Core:2）

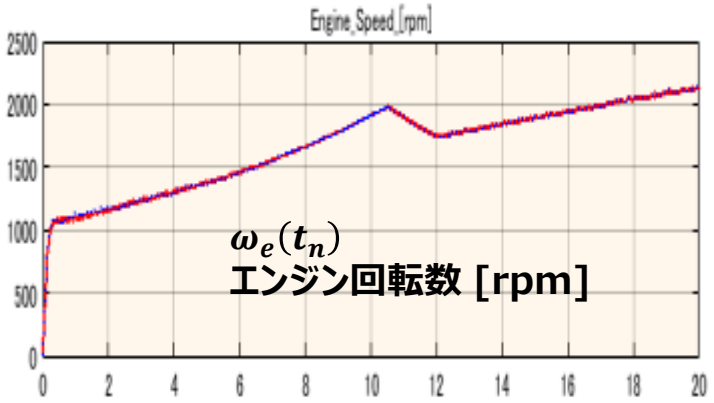


3.3 物理構造を活用した並列演算手法

■ 物理構造を活用した並列モデリング

簡易のコンベ車にてシミュレーション精度の影響を比較　（ｽﾃｯﾌﾟｻｲｽﾞ：0.5ms / ｼﾐｭﾚｰｼｮﾝ時間：0 - 20s）

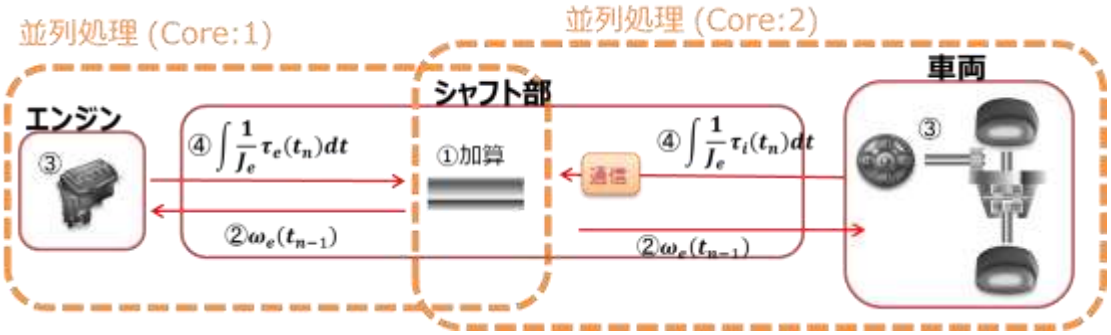




《凡例》

① 逐次処理　（左グラフ：　　）

② 並列処理　（左グラフ：　　）



並列前後で 結果一致。
通信によるシミュレーション精度の
影響を解消

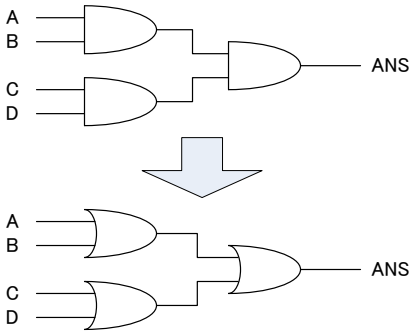
4. FPGAでの並列モデリング

4.1 FPGAとは

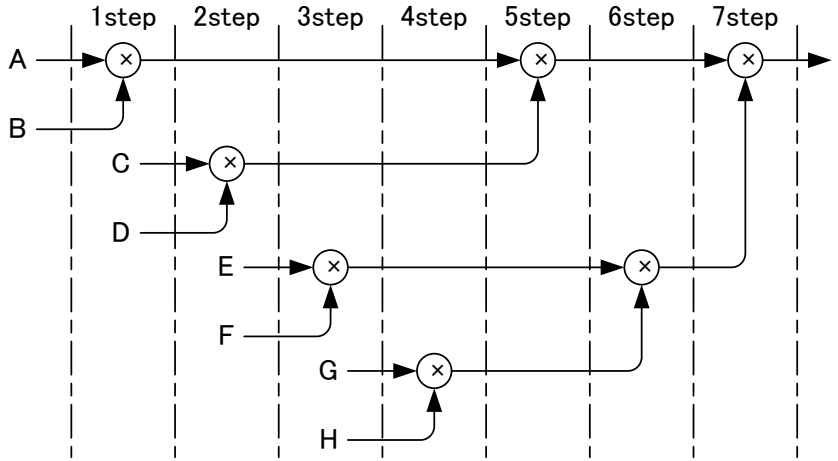
- Field Programmable Gate Array の略
- 現場で　書き変えられる　カスタムIC
- ハードウェア（集積回路の一つ）
- **内部回路を変更できる**

  集積回路（IC）：複数の部品で構成される電子部品

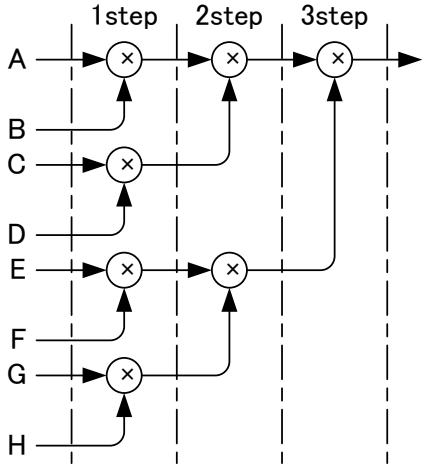
内部回路変更のイメージ



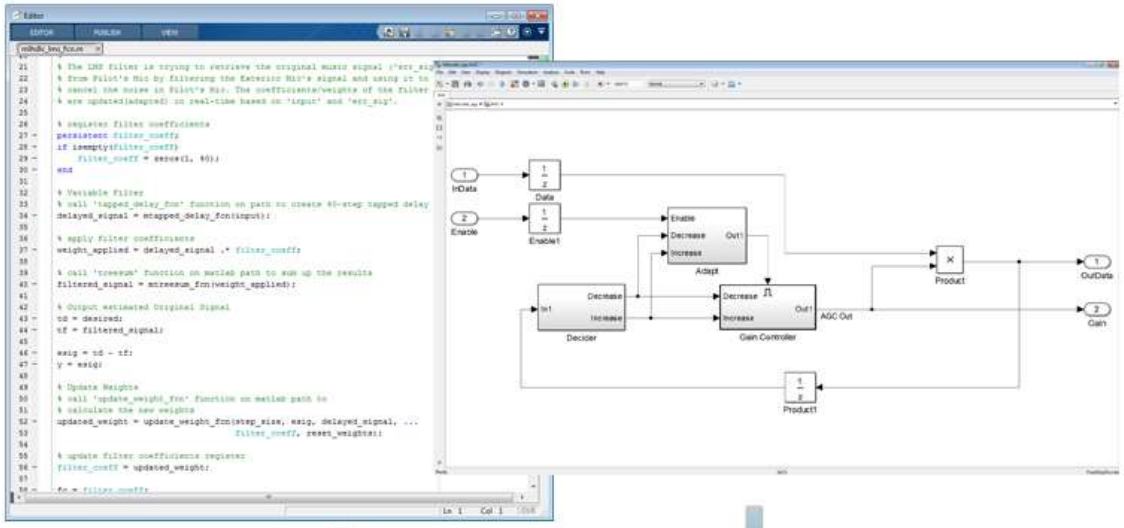
CPU：順序実行のため
1stepで1命令を実行

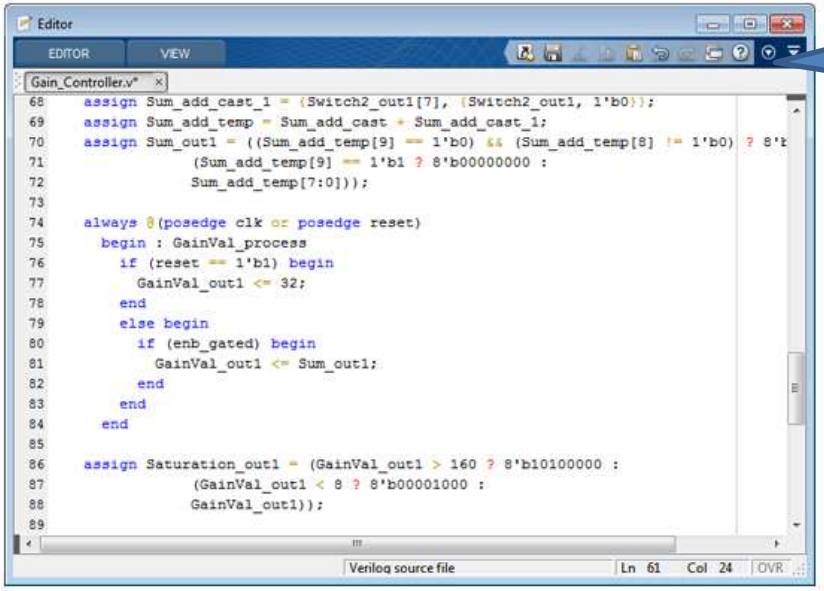


FPGA：並列実行のため
1stepで複数の処理を実行



4.1 FPGAとは





HDL Coder™ でSimulink からHDLコードを生成

出展：https://jp.mathworks.com/products/hdl-coder.html

4.2　物理モデルのFPGA実装

■FPGA実装用Simulinkモデル作成のフロー

① 方程式 / 構造

② Simscapeモデル
(物理モデリングツール)

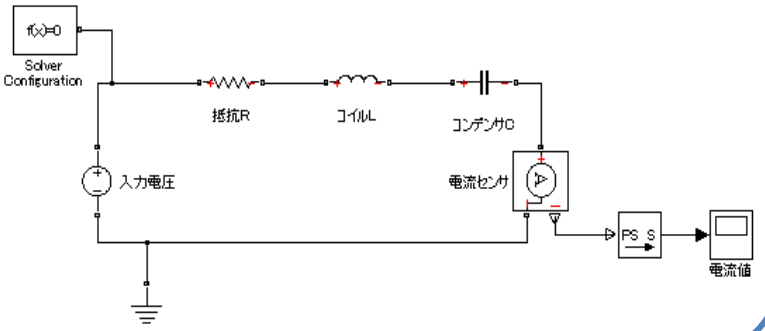
③ 微分代数方程式を
常微分方程式へ

④ Simulink モデル

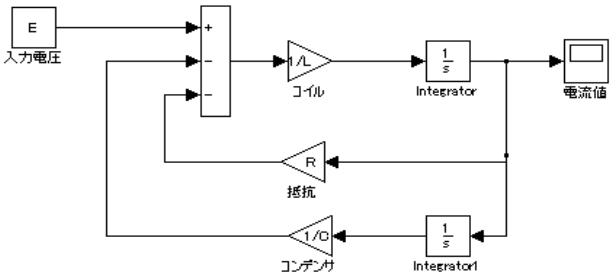
FPGA用Simulinkモデル実装

物理モデル実装

② Simscapeモデル



④ Simulinkモデル



4.2　物理モデルのFPGA実装

■FPGA実装用Simulinkモデル作成のフロー

④　Simulink モデル

↓

⑤　HDL Coder Block
置き換えモデル

平方根や割り算などを
マップ置き換え

↓

⑥　Timing design

はじめは、
余裕をもった
タイミング設計

↓

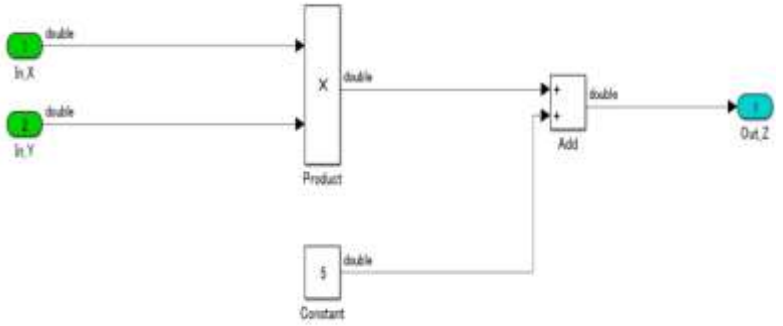
⑦ Fixed Point design

自動固定小数点
化機能を活用

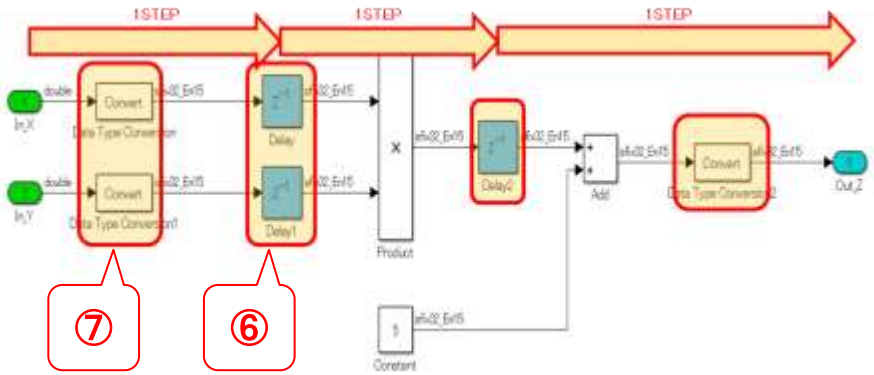
↓

⑧　FPGA実装用
Simulinkモデル

↓ Simulinkモデル



↓ FPGA実装用Simulinkモデル



【ポイント】
ハードウェア(FPGA)上の処理タイミングを考慮したタイミング設計

4.3　並列化を考慮したFPGA実装

■FPGA実装用Simulinkモデル作成のフロー

① 方程式

↓

② Simscapeモデル
(物理モデリングツール)

↓

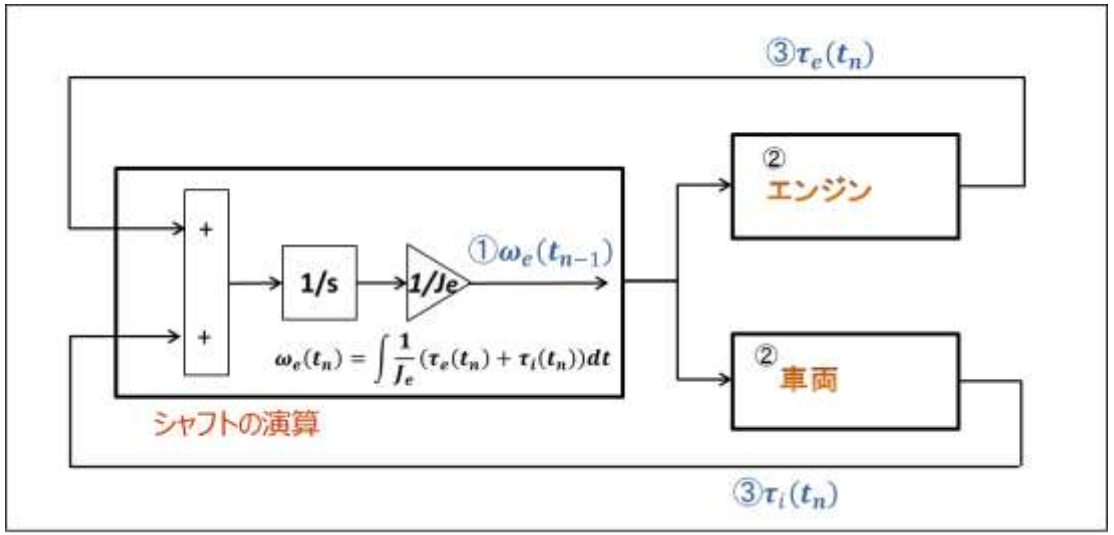
並列化を考慮した
③ 微分代数方程式を
常微分方程式へ

↓

④ Simulink モデル

↓

FPGA用Simulinkモデル実装

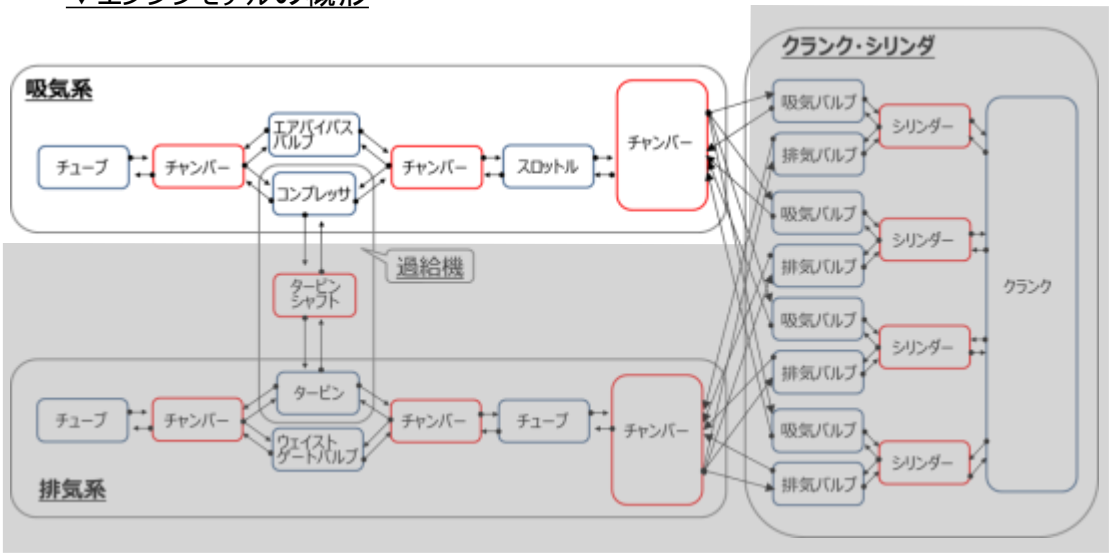


FPGAに実装するモデルも
マルチコア並列化と同じく物理構造を
考慮して並列化

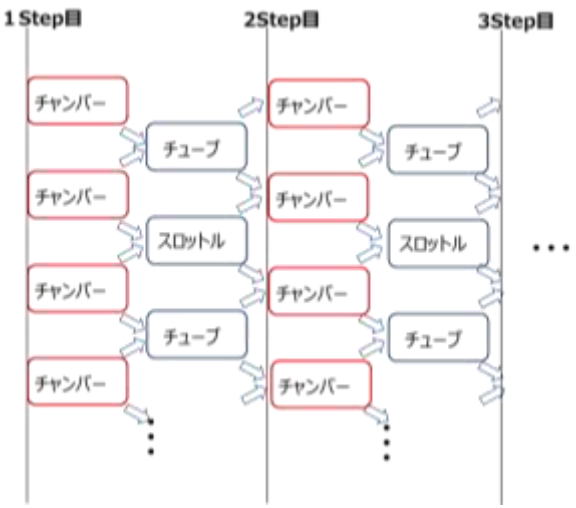
4.4 応用事例 –エンジンモデル–

モデルを部品単位で粗粒度並列

▼エンジンモデルの概形

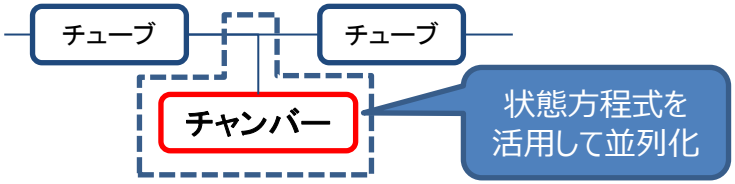


▼ 並列処理フロー



※**赤色**の部品が積分器を持っている

▼ ブロック図



質量流量[kg/s]： $\dot{n_A}, \dot{n_B}, \ldots$ 熱流量[J/s]： $\dot{u_A}, \dot{u_B}, \ldots$	圧力[Pa]： P_0 温度[K]： T_0	気体の状態方程式： $N_0 = \int (\dot{n_A} + \dot{n_B} + \cdots)dt \, , U_0 = \int (\dot{u_A} + \dot{u_B} + \cdots)dt$ $P_0 = \frac{U_0}{V} \cdot \frac{2}{f}, \quad T_0 = \frac{V}{N_0} \cdot \frac{P_0}{R_{univ}}$

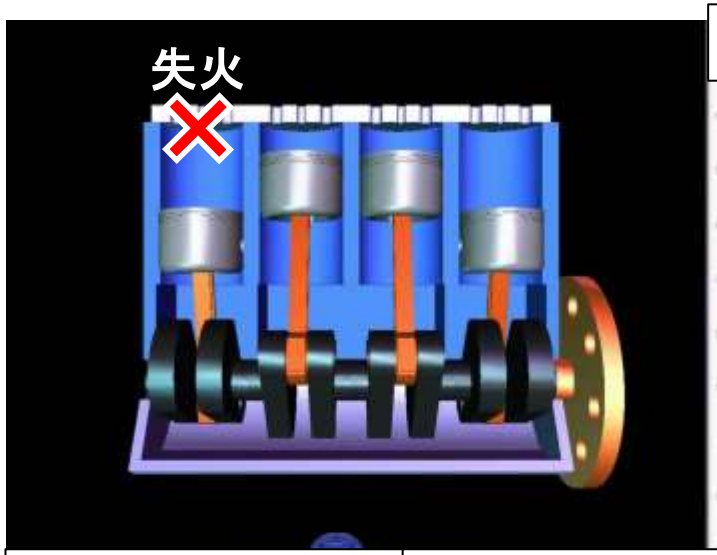
【ポイント】

積分器を持つ物理的部品の活用

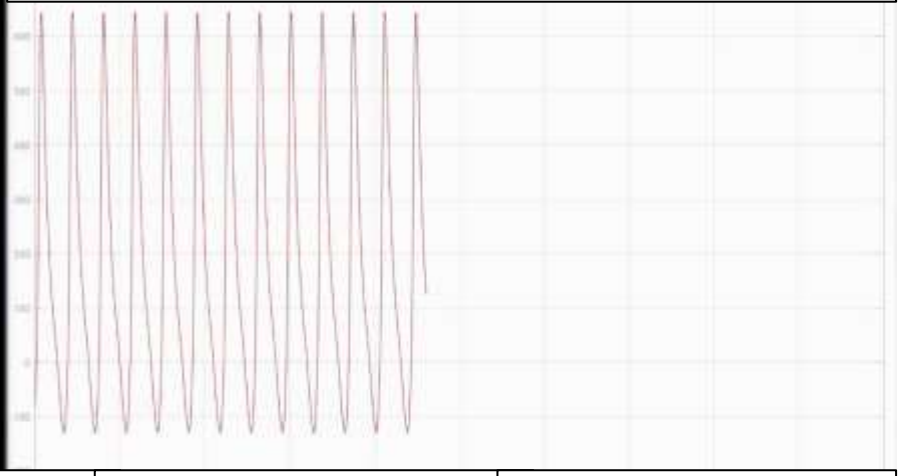
チャンバー、シャフト、etc

4.4 応用事例 –エンジンモデル–

1気筒失火シミュレーション ⇒600[rpm]でアイドリング中に失火



エンジントルク[N*m]



シリンダ内圧力 (1気筒目) [MPa]

シリンダ内圧力 (2気筒目) [MPa]

シリンダ内圧力 (3気筒目) [MPa]

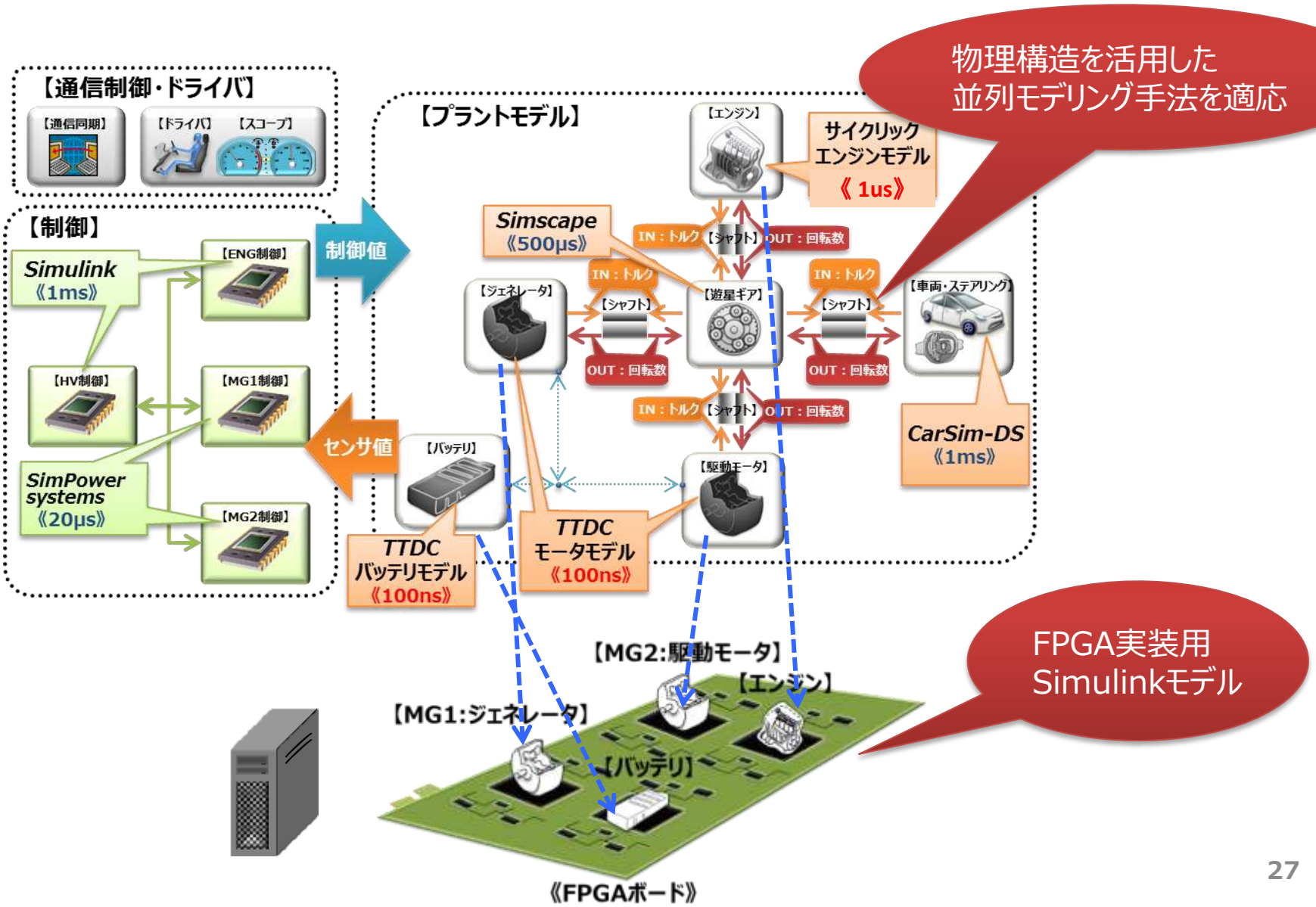
シリンダ内圧力 (4気筒目) [MPa]

1気筒失火時のトルクや筒内圧をシミュレーション可能

5. マルチコア ＋　FPGA　の事例

5. マルチコア+FPGAの事例

- 【応用事例】HV車両シミュレーション モデル構成

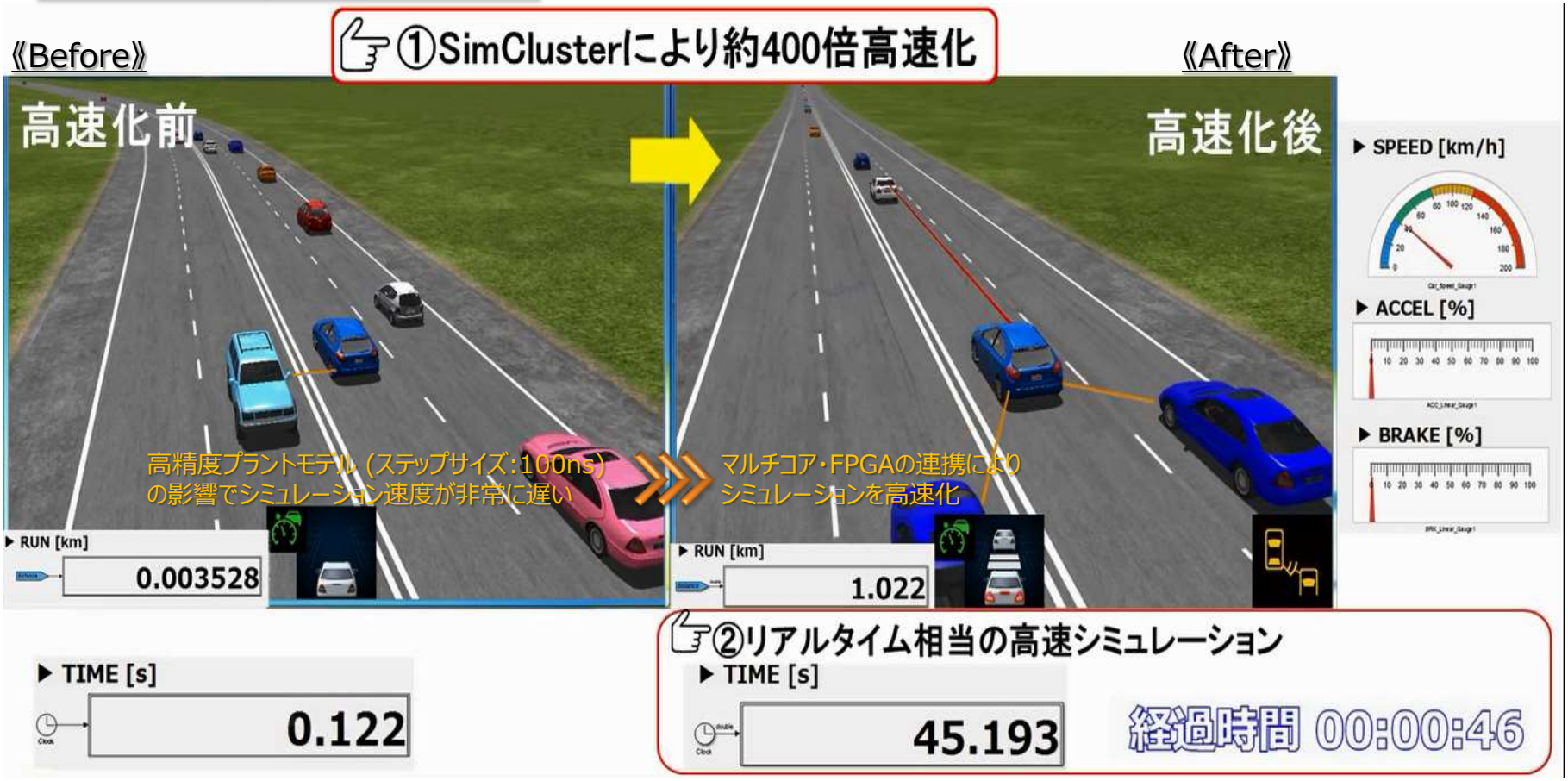


5. マルチコア+FPGAの事例

■ 【応用事例】HV車両シミュレーション 高速化結果

▼ HV車シミュレーション高速化結果（※ シミュレーション1[s]に対する計測時間）

高速化前 (逐次処理)	高速化後 (並列+ FPGA)
約400 [s]	約1.0 [s]



6. まとめ

■ 並列化によるシミュレーション高速化のポイント

・物理構造を考慮した並列モデリング手法

・FPGA実装による高速化

物理構造を考慮した並列モデリングの考え方は、
マルチコアCPUでもFPGAでも共通

■ MathWorks様への期待

・ HDL Coderの浮動小数点機能の発展

・ SimscapeからのHDL Code生成