2014年 12月 18日 (木)

ソフトウェア単体検査の自動化への取り組み

株式会社デンソー

電子基盤システム開発部
モデルベース技術開発室

井原 博之

DENSO

本日の内容

1. 背景
    1. 車載制御ソフトウェア開発の現状
    2. 機能安全（ISO26262）
2. Auto Test Generation（ATG）の導入
    1. 可視性（Observability）
    2. 可視性の実現
    3. 適用結果
3. 今後の取り組みの方向
4. まとめ

1.1. 車載制御ソフトウェア開発の現状

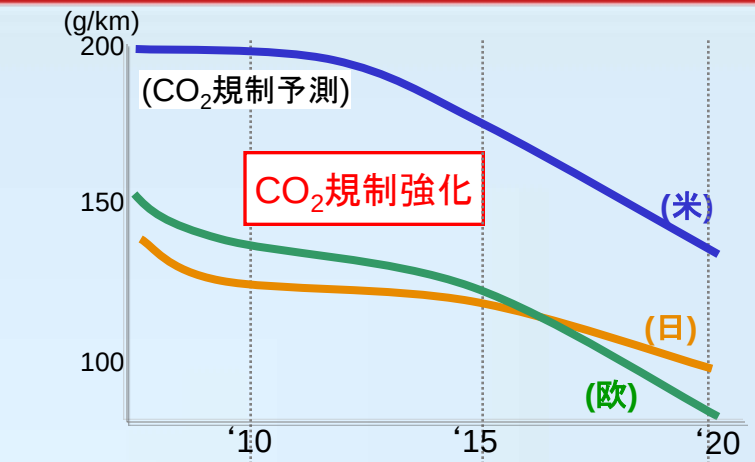
車載制御ソフト開発の動向

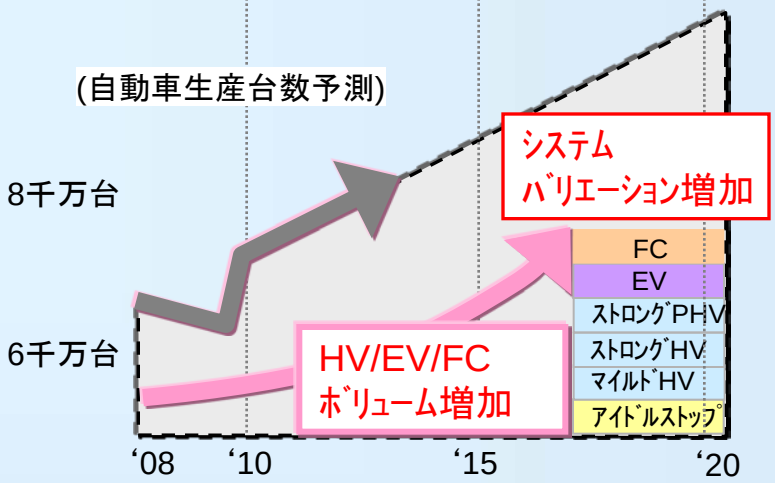
- 法規制対応による緻密化
- システムバリエーション増加
- 電動化による大規模化
- 短期開発競争の激化

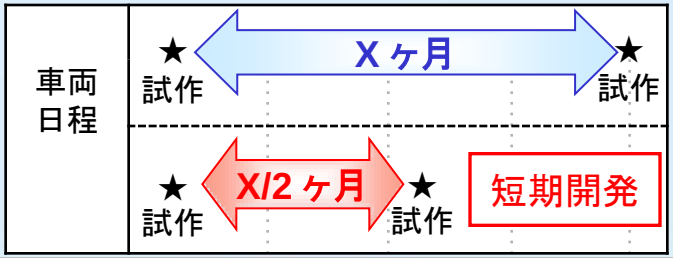
コントローラ開発の動向

- 仕様作成のルールは協調、実装は競争
  → AUTOSAR
- 安全性を確保する設計思想
  → 機能安全(ISO26262)

標準規格に従った
自動化手法の適用が重要







1.2. 機能安全(ISO26262)

■ ISO26262 Part6 9.4.6 NOTE4

For model-based development, software unit testing can be carried out at the model level followed by back-to-back comparison tests between the model and the object code. The back-to-back comparison tests are used to ensure that the behaviour of the models with regard to the test objectives is equivalent to the automatically-generated code.

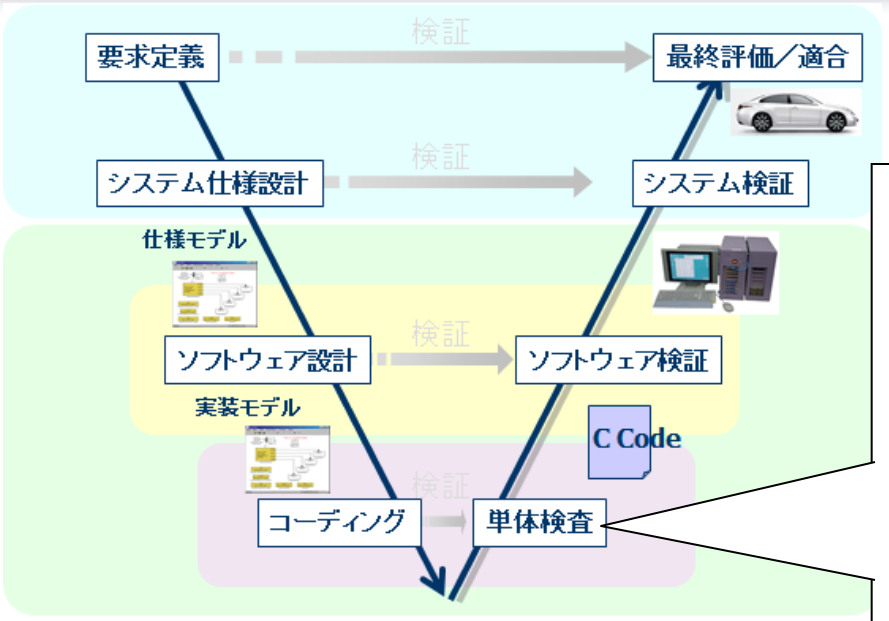
機能安全規格では、ソフトウェアユニットテストにおいて、コードに対して、要求ベースのテストが要求されているが、モデルベース開発の場合、以下の方法も許容されている。

・要求ベースのテストをモデルで実施し
・モデルーコードー致性検証(B to B比較テスト)

※B to B比較テストの役割をテスト対象の振る舞いが、モデルーコードで一致することの保障と位置付け

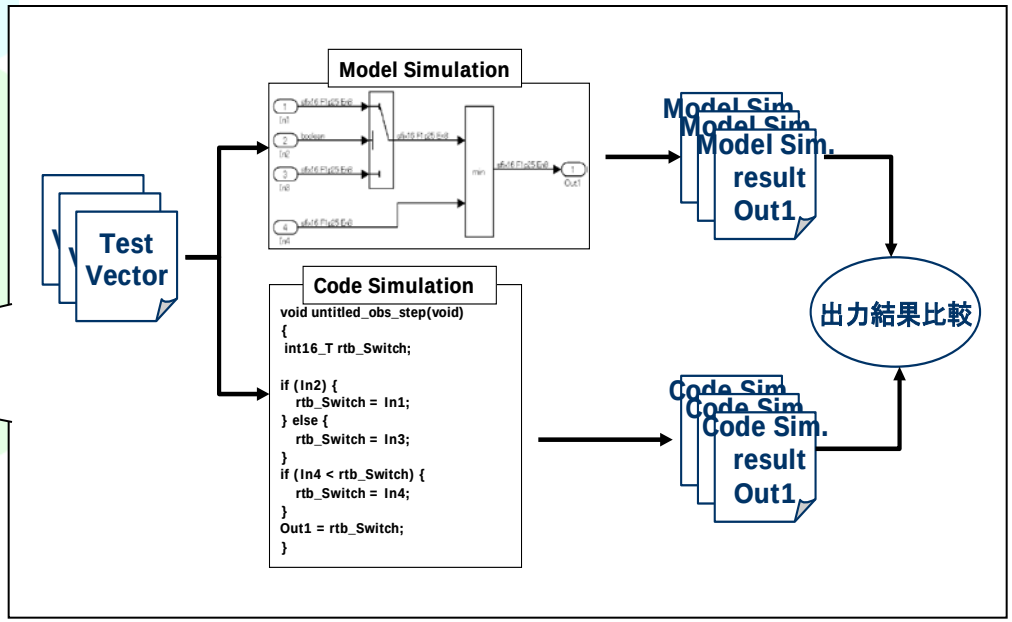
ISO26262で求める
B to B比較テストを満足する自動化を実現する

2. Auto Test Generation（ATG）の導入



単体検査

B to B比較テスト（一致性検証）



ATG技術を導入

ISO26262で推奨されているカバレッジ（MC/DC、DCなど）を満たすテストベクタの自動生成を実現

導入のポイント

可視性の実現

可視性：Observability

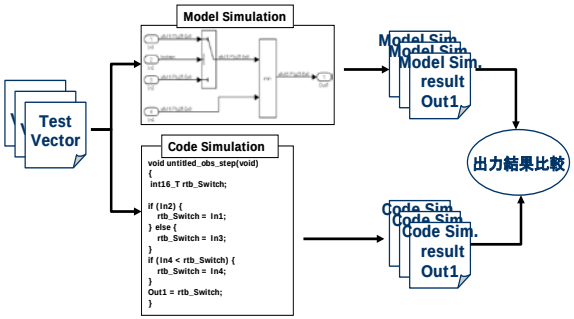
Simulink® Design Verifier ™ を活用※し、B to B比較テスト環境を構築

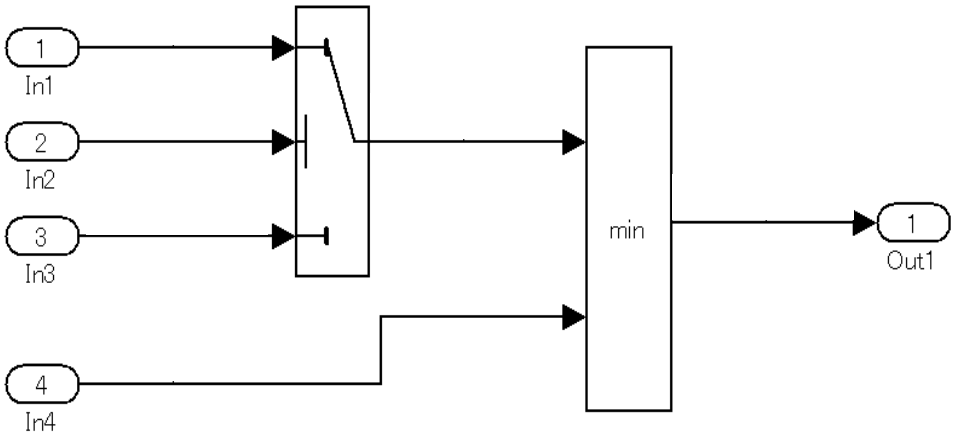
※Simulink Design Verifierの機能を利用し可視性を実現

Simulink Design Verifier：
形式的手法を使用し、テストケースの生成、各種検証を行うSimulink® の検証ツール

DENSO

2.1. 可視性（Observability）



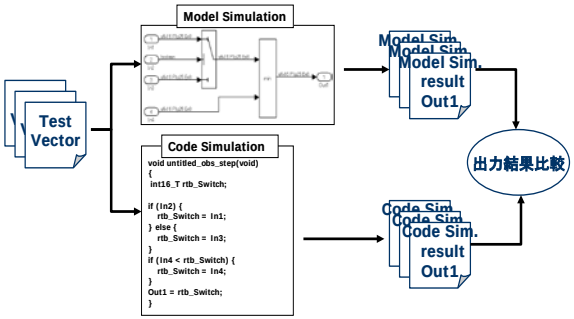


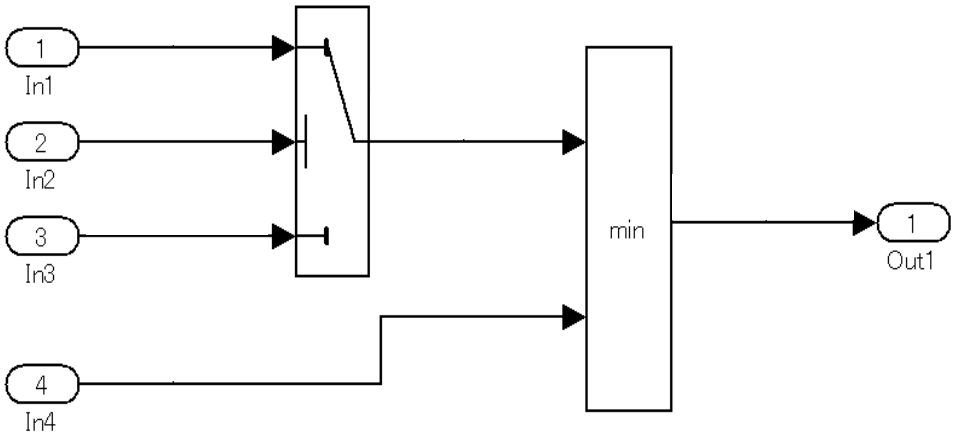
テストベクタ					Out1
No.	In1	In2	In3	In4	
1	12	False	10	11	10
2	12	True	10	11	11

```
void untitled_obs_step(void)
{
 int16_T rtb_Switch;

if (In2) {
   rtb_Switch = In1;
} else {
   rtb_Switch = In3;
}
if (In4 < rtb_Switch) {
   rtb_Switch = In4;
}
Out1 = rtb_Switch;
}
```

2.1. 可視性（Observability）



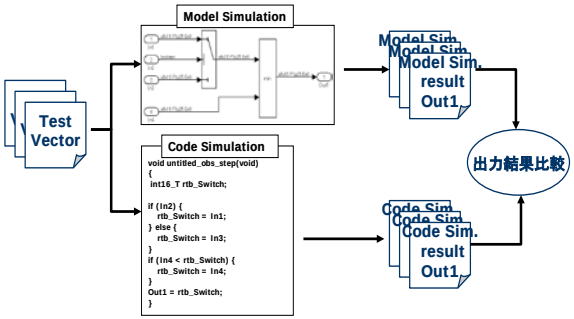


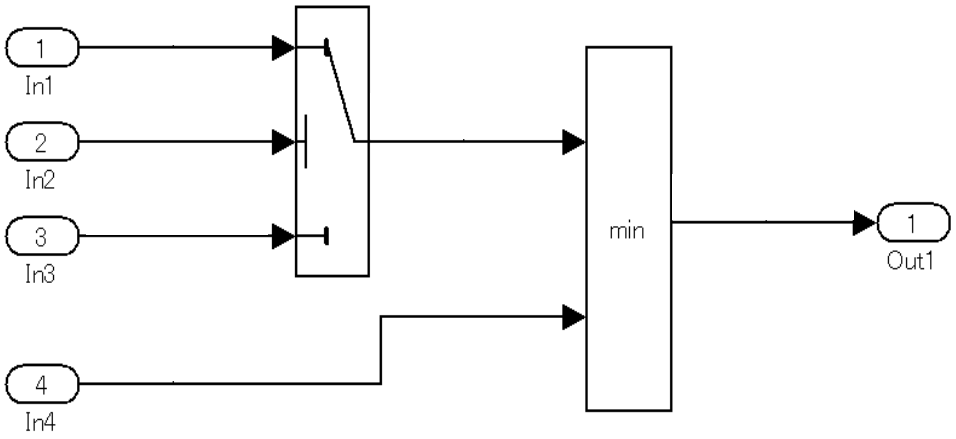
テストベクタ					Out1
No.	In1	In2	In3	In4	
1	12	False	10	11	10
2	12	True	10	11	11

```
void untitled_obs_step(void)
{
 int16_T rtb_Switch;

if (In2) {
   rtb_Switch = In1;
} else {
   rtb_Switch = In3;
}
if (In4 < rtb_Switch) {
   rtb_Switch = In4;
}
Out1 = rtb_Switch;
}
```

2.1. 可視性（Observability）



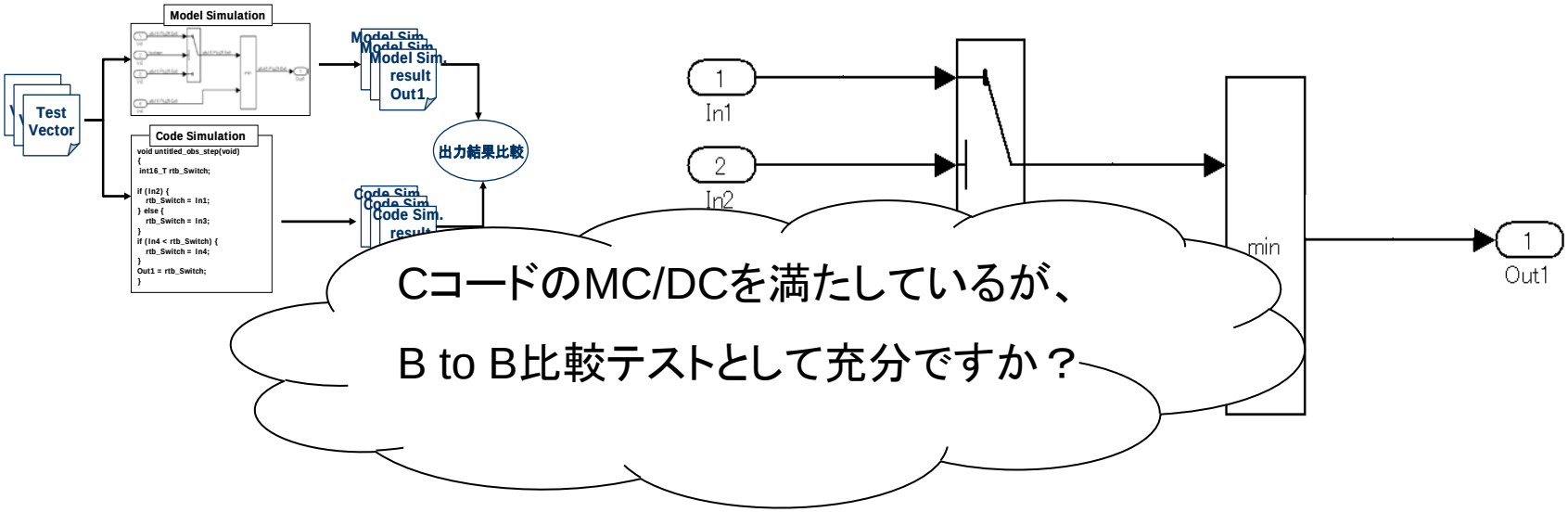


テストベクタ					Out1
No.	In1	In2	In3	In4	
1	12	False	10	11	10
2	12	True	10	11	11

```
void untitled_obs_step(void)
{
 int16_T rtb_Switch;

if (In2) {
   rtb_Switch = In1;
} else {
   rtb_Switch = In3;
}
if (In4 < rtb_Switch) {
   rtb_Switch = In4;
}
Out1 = rtb_Switch;
}
```

2.1. 可視性（Observability）



テストベクタ					Out1
No.	In1	In2	In3	In4	
1	12	False	10	11	10
2	12	True	10	11	11

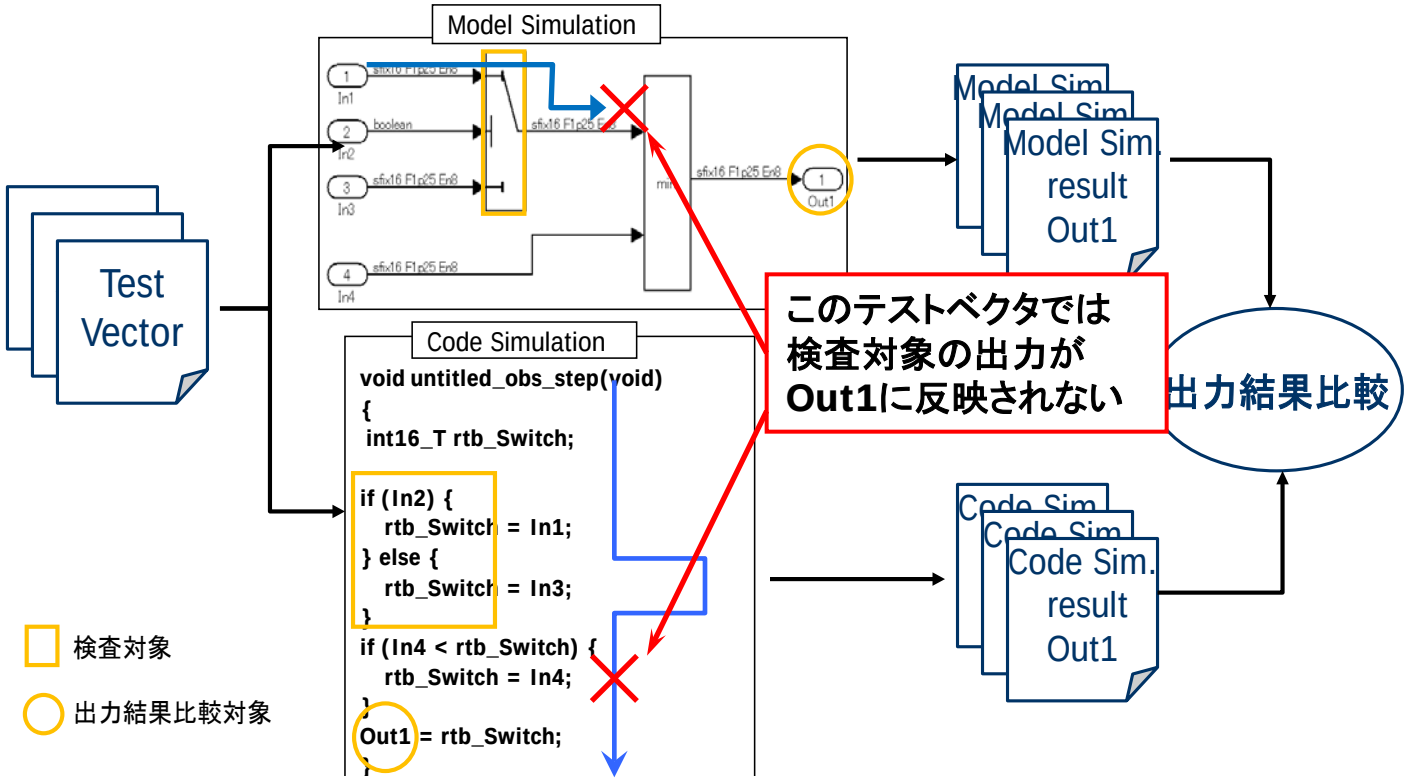
```
void untitled_obs_step(void)
{
 int16_T rtb_Switch;

if (In2) {
   rtb_Switch = In1;
} else {
   rtb_Switch = In3;
}
if (In4 < rtb_Switch) {
   rtb_Switch = In4;
}
Out1 = rtb_Switch;
}
```

2.1. 可視性（Observability）

テストベクタ					Out1
No.	In1	In2	In3	In4	
1	12	False	10	11	10
2	12	True	10	11	11

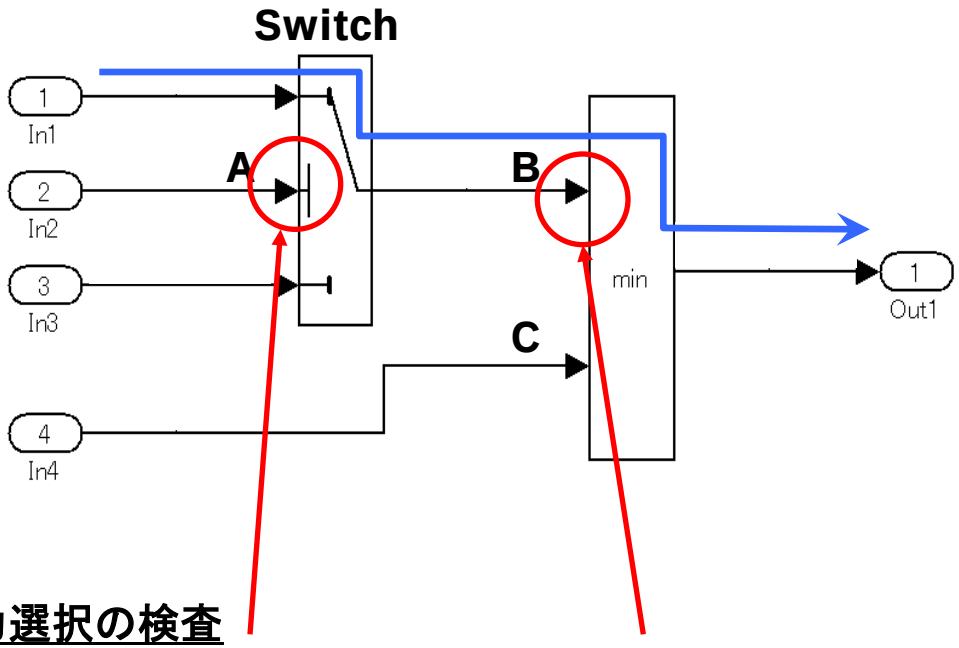
Switch 第1入力選択の検査



テスト対象の出力が、結果比較できる出力に反映（可視化）されなければならない
＝可視性のあるテストベクタが必要

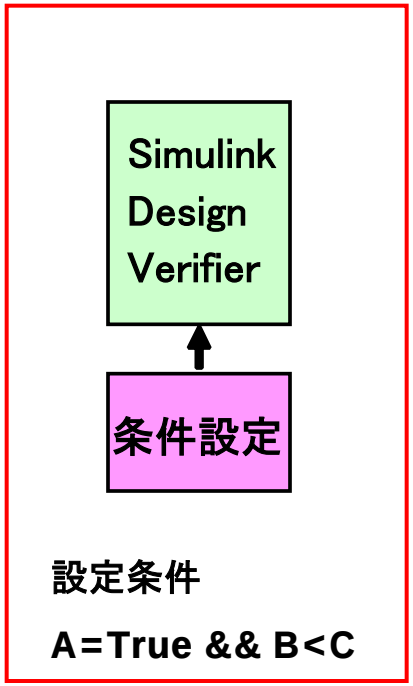
2.2. 可視性の実現

可視性実現のために



Switch 第1入力選択の検査

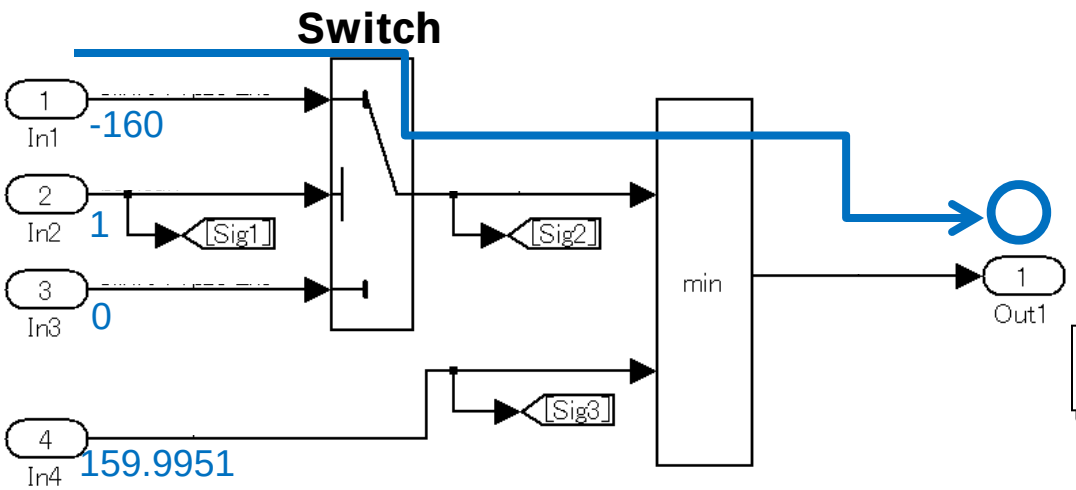
第1入力を選択	第1入力を出力
A = True	B < C



2.2. 可視性の実現

具体的には

Switch 第1入力選択の検査



テスト生成

<Simulink Design Verifier テスト生成レポート>

Test Case 1

Summary

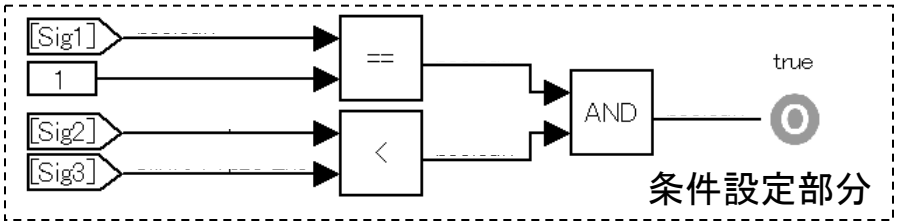
Length: 0 Seconds (1 sample periods)
Objective Count:1

Objectives

Step	Time	Model Item	Objectives
1	0	Test Objective	Objective: T

Generated Input Data

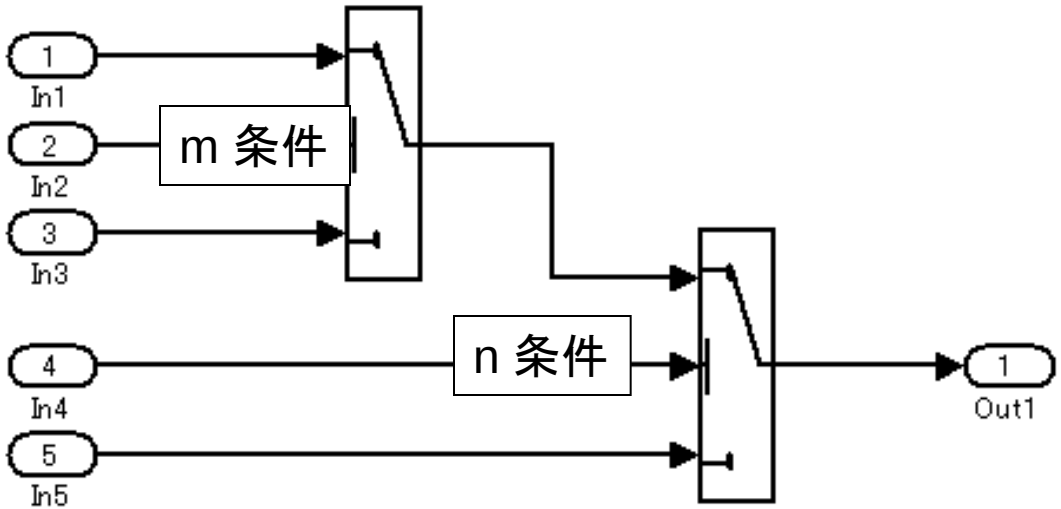
Time	**0**
Step	**1**
In1	-160
In2	1
In3	0
In4	159.9951



※スクリプト形式（Sig1==1 && Sig2<Sig3）での記述も可能

2.2. 可視性の実現

要求されるテストベクタ数



MC/DC

Max (m, n) + 1

MC/DC + Observability

Sum (m, n) + 1

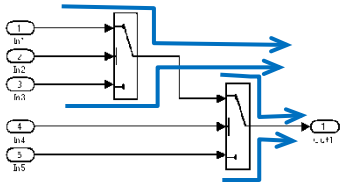
テストベクタ数の増加は許容できるレベル

2.3. 適用結果

Component	Subsystem	カバレッジ率 MC/DC + Observabiliy	
		条件設定なし	条件設定あり
A	Total (19Subsystems)	47％	93％
B	Subsystem1	-	94％
	Subsystem2	-	100％
	Subsystem3	-	97％
	Subsystem4	-	26％
	Subsystem5	-	100％
	Subsystem6	-	74％
	Subsystem7	-	100％
	Subsystem8	-	82％
	Subsystem9	-	95％
Total		-	91％

※ここでのカバレッジ率＝

$$\frac{\text{可視性を満たすMC/DC観点数}}{\text{総MC/DC観点数}}$$



これの出来た率

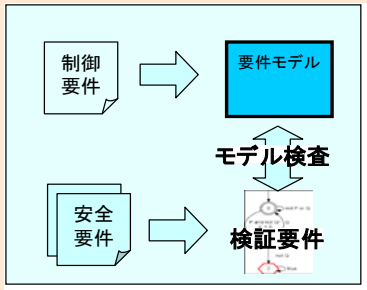
モデル規模、記述内容により、カバレッジが低いケースがある

効果は出ているが、まだまだ問題があり改善が必要

3. 今後の取り組みの方向

②要件検証

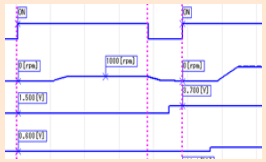
形式的検証技術等を用いた、
要件の妥当性・整合性の検証

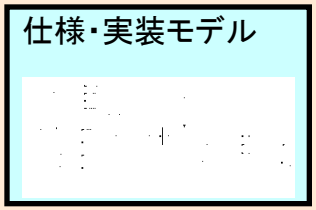


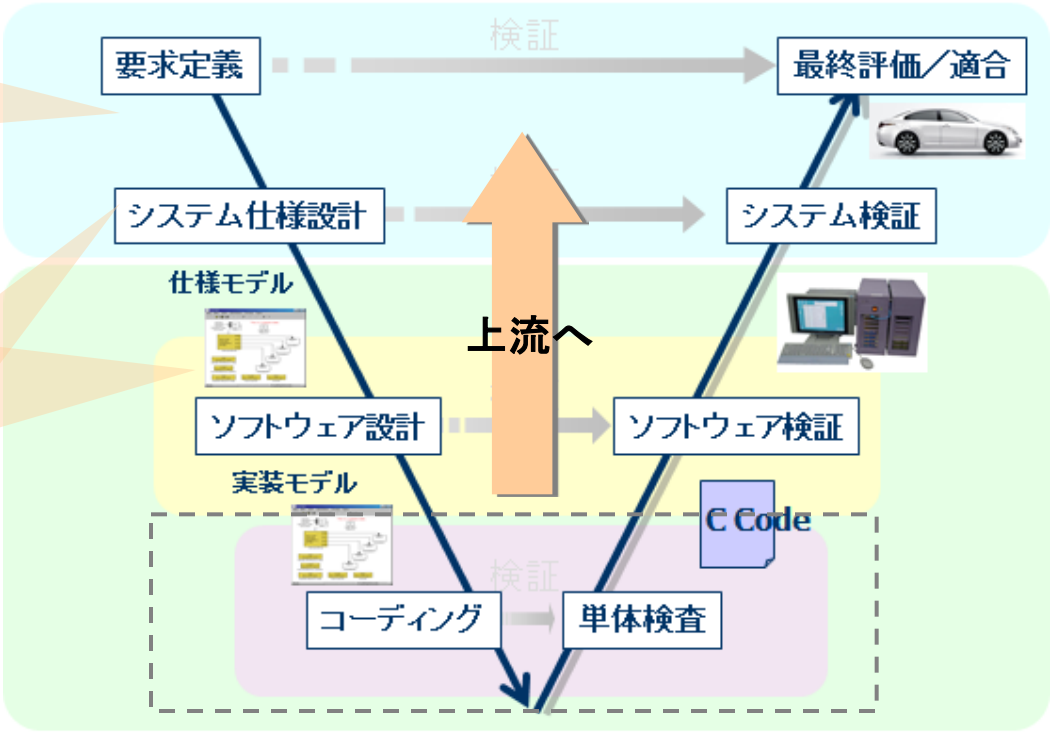
①機能検査データ生成（機能ATG）

機能検査データの自動生成

機能検査データ







開発工程上流からシームレスに最適化されたプロセスを構築

『ソフトウェア単体検査の自動化への取り組み』

- ECUソフトウェア開発では大規模・複雑化／各種ルール・規格への対応が必要

- B to B比較テストの自動化に取り組んでいる
- 可視性（Observabilty）の実現が重要
- Simulink Design Verifier への条件設定により可視性を実現
- 高い効果が得られているが、改善ポイントも残っている

- 今後は、要件検証、機能検査自動化等上流工程へ拡張して行く

今後も継続して、
より洗練された効果的な検証を実現していきたいと考えている

ご清聴ありがとうございました