

Design of PCI-BUS System  
using SpecC

November. 27, 2000

Tadaaki Tanimoto

Yoichi Kobayashi

IP Technology Center

System LSI Business Division

Semiconductor & Integrated Circuits

Hitachi Ltd.

## Agenda

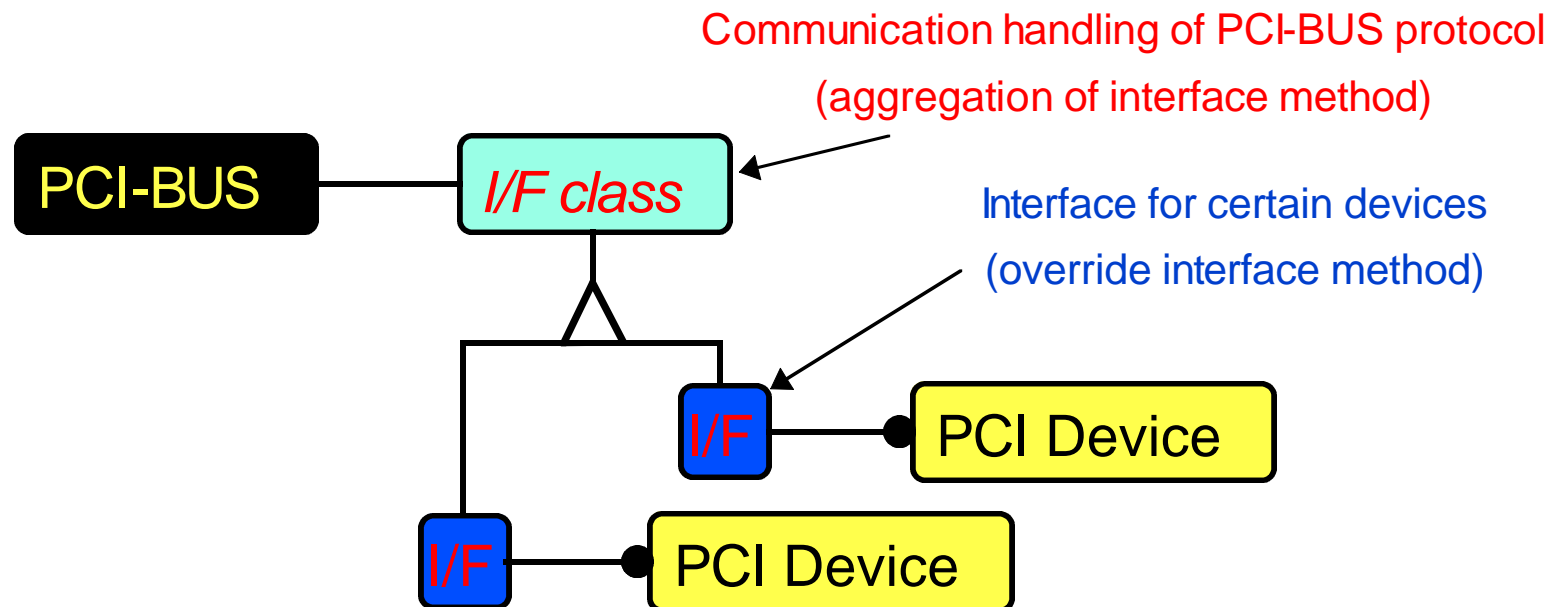
1. Background and Purpose
  
2. Plan for Implementation
  - 2.1 PCI-BUS system diagram
  - 2.2 Implementation Items for Simulator
  - 2.3 Implementation Spec (subset of PCI-BUS)
  - 2.4 Schedule of Implementation
  
3. Current Status
  - 3.1 Implementation Spec
  - 3.2 SpecChart
  - 3.3 State Diagram
  - 3.4 Channel Structure
  - 3.5 Problems in using U.C.I Simulator
  - 3.6 Work Around
  - 3.7 Timing Diagram
  
4. Conclusion

## 1. Background and Purpose (1)

(1) トランスデューサ部のSpecC記述を行うことでSpecC言語の表現能力を評価。

→ PCI-BUSシステムを対象に記述。

(2) UML (class diagram) とSpecChartの対比。



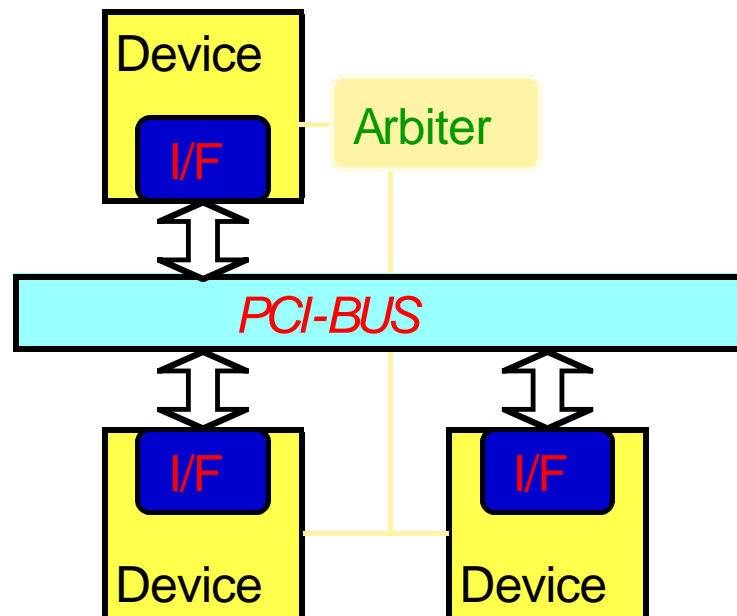
Class Diagram of PCI-BUS system (Just Image)

## 1. Background and Purpose (2)

### (3) 性能検証と機能検証の分離。

性能評価はバスプロトコル設計やシステム仕様決定に不可欠。

→ SpecCによる性能評価シミュレータの作成。



No contradiction in both

- ◆ Bus protocol handling
- ◆ Arbitration handling

But,

Dead Lock is occurred!!  
Violate real time constraint.

Thus,

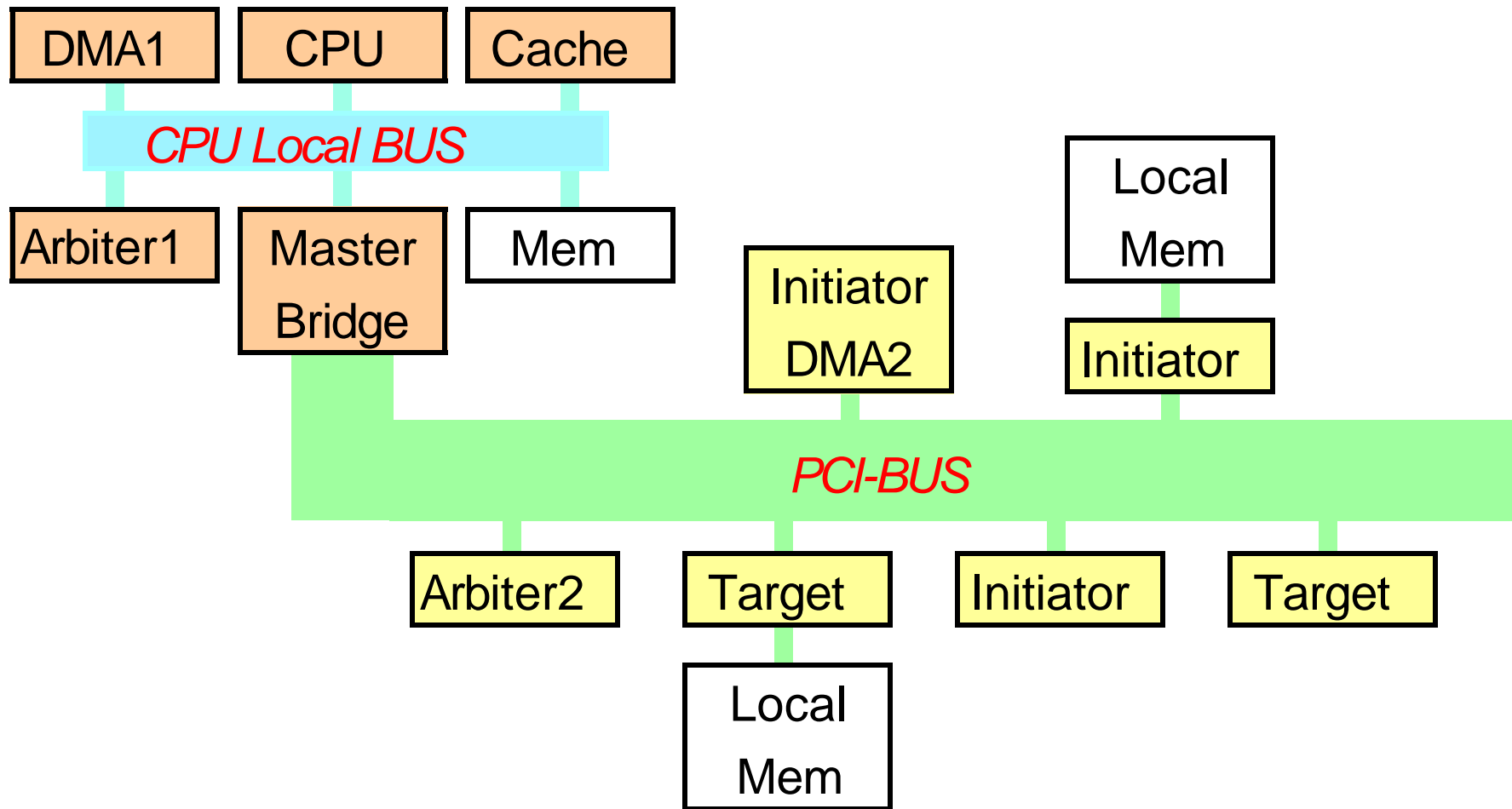
Need performance simulation!!

# Design of PCI-BUS System using SpecC



## 2. Plan for Implementation (1)

PCI-BUS system diagram



## 2. Plan for Implementation (2)

### Implementation Items for Simulator

#### Modules

- ◆ CPU Local BUS
  - CPU (Test Bench)
  - Arbitor1
  - Cache (4-Way, Write Back, Blocking)
  - DMA1
- ◆ Master Bridge
- ◆ PCI-BUS
  - Initiator with DMA function
  - Customizable Devices
  - User designed Devices

#### Customizable Items

- ◆ Device type
  - Initiator
  - Target
- ◆ Latency
  - Fixed cycle
  - Random cycle with probability transition
- ◆ Communication method
  - Selection of methods
  - Selection of termination method
  - Active probability of methods
  - Active probability of abnormal termination

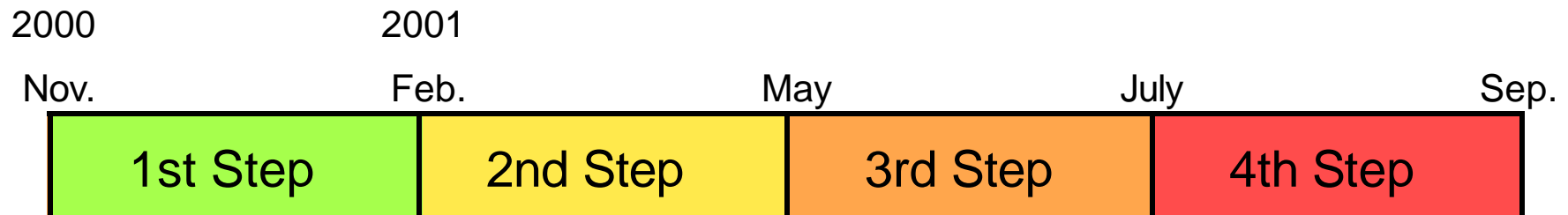
## 2. Plan for Implementation (3)

### Implementation Spec (subset of PCI)

種別	略号	名称	方向		SpecC 評価用	備考
			マスタ	ターゲット		
システム・ピン	CLK	クロック	in	in		
	RST_	リセット	in	in		
アドレスと データ・ピン	AD[31:00]	アドレスとデータ	in/out	in/out		
	C/BE_[3:0]	バス・コマンドとバイトイネーブル	out	in		
	PAR	パリティ	in/out	in/out		
インタフェース 制御信号ピン	FRAME_	サイクル・フレーム	out	in		
	IRDY_	イニシエータ・レディ	out	in		
	TRDY_	ターゲット・レディ	in	out		
	STOP_	ストップ	in	out		
	LOCK_	ロック	out	in		
	IDSEL_	イニシャリゼーション・デバイス セレクト	in	in		
	DEVSEL_	デバイス・セレクト	in	out		
アービトレーション・ ピン	REQ_	リクエスト	out	-		
	GNT_	グラント	in	-		
エラー報告ピン	PERR_	パリティ・エラー	in/out	out		
	SERR_	システム・エラー	out	out		
インタラプト・ピン	INTA_	インタプリタ	out	out		
	INTB_	インタプリタB	out	out		
	INTC_	インタプリタC	out	out		
	INTD_	インタプリタD	out	out		
キャッシュ・ サポート・ピン	SBO_	スヌープ・バックオフ	in/out	in/out		
	SDONE	スヌープ完了	in/out	in/out		
64ビット・ バス拡張ピン	AD[63:32]	アドレスとデータ	in/out	in/out	x	
	C/BE_[7:4]	バス・コマンドとバイトイネーブル	out	in	x	
	REQ64_	64ビット転送要求	out	in	x	
	ACK64_	64ビット転送アクノリッジ	in	out	x	
	PAR64_	パリティ	in/out	in/out	x	
JTAG/バウンダリ・ スキャン・	TCK	テスト・クロック	in	in	x	
	TDI	テスト・データ入力	in	in	x	
	TDO	テスト・データ出力	out	out	x	
	TMS	テスト・モード・セレクト	in	in	x	
	TRST_	テスト・リセット	in	in	x	

## 2. Plan for Implementation (4)

### Schedule of Implementation



*Maybe Release*

1st Step. Support Basic Data Transfer

- Without Read / Write configuration registers  
(use user defined files for configuration)
- Support N devices

2nd Step. Support DMA

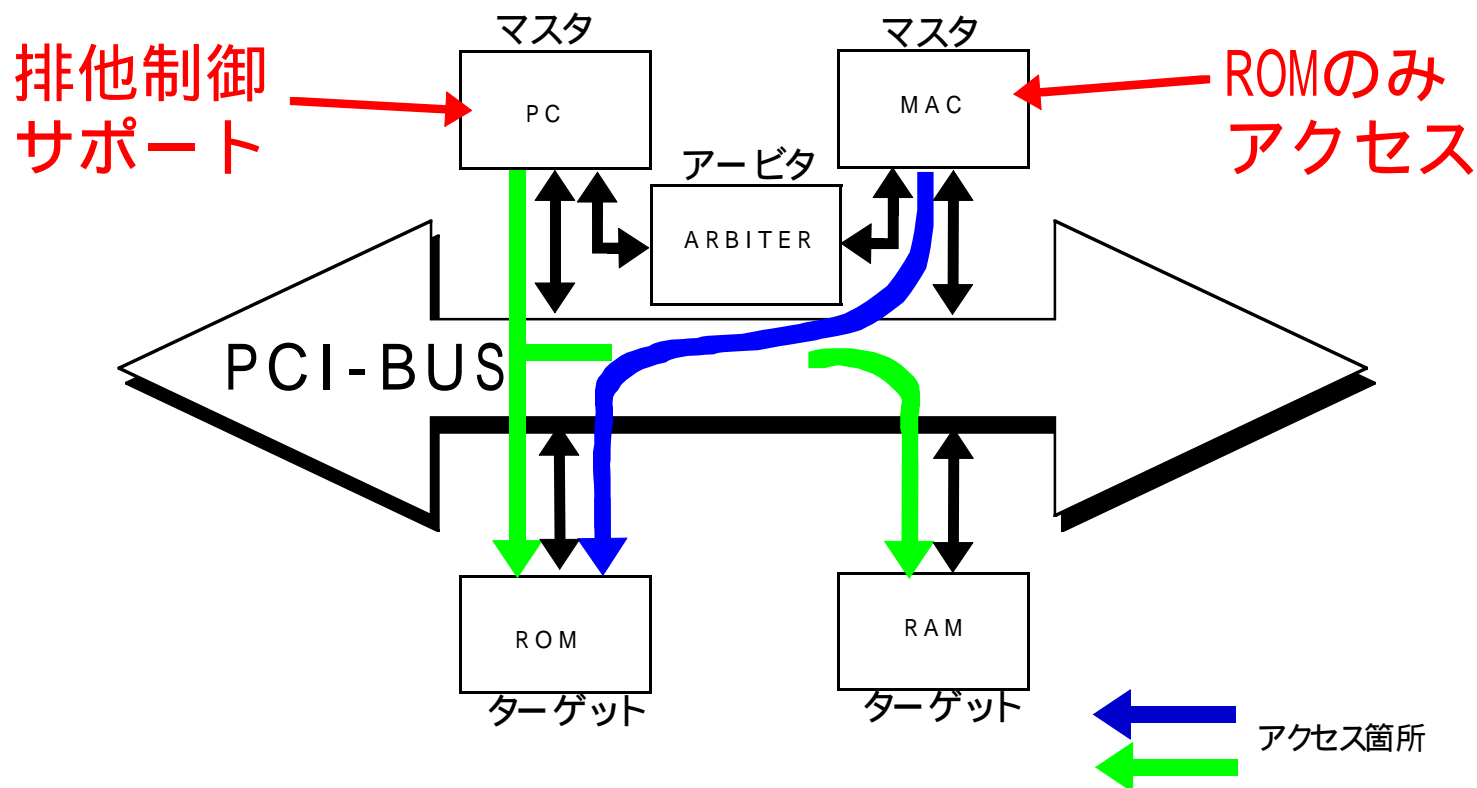
3rd Step. Support ALL Data Transfer

4th Step. Support Configuration Cycle

## 3. Current Status (1)

### Implementation Spec

JTAG・バウンダリスキャン、64ビット拡張、キャッシュ・メモリ、インタラプト、コンフィギュレーション・レジスタ以外をサポート



SpecCはV205ソラリス版を使用

## 3. Current Status (2)

### Implementation Spec (PCI-BUSの信号構成)

種別	略号	名称	方向		SpecC 評価用	備考
			マスタ	ターゲット		
システム・ピン	CLK	クロック	in	in		
	RST	リセット	in	in		
アドレスとデータ・ピン	AD[31:00]	アドレスとデータ	in/out	in/out		
	C/BE_[3:0]	バス・コマンドとバイトイネーブル	out	in		
	PAR	パリティ	in/out	in/out		
インタフェース制御信号ピン	FRAME_	サイクル・フレーム	out	in		
	IRDY_	イニシエータ・レディ	out	in		
	TRDY_	ターゲット・レディ	in	out		
	STOP_	ストップ	in	out		
	LOCK_	ロック	out	in		
	IDSEL_	イニシャリゼーション・デバイス・セレクト	in	in	x	
	DFVSEI	デバイス・セレクト	in	out		
アービトレーション・ピン	REQ_	リクエスト	out	-		
	GNT_	グラント	in	-		
エラー報告ピン	PERK_	ハリアイ・エラー	in/out	out		
	SERR_	システム・エラー	out	out		
インタラプト・ピン	INTA_	インタプリタ	out	out	x	
	INTB_	インタプリタB	out	out	x	
	INTC_	インタプリタC	out	out	x	
	INTD_	インタプリタD	out	out	x	
キャッシュ・サポート・ピン	SBO_	スヌープ・バックオフ	in/out	in/out	x	
	SDONE	スヌープ完了	in/out	in/out	x	
64ビット・バス拡張ピン	AD[63:32]	アドレスとデータ	in/out	in/out	x	
	C/BE_[7:4]	バス・コマンドとバイトイネーブル	out	in	x	
	REQ64_	64ビット転送要求	out	in	x	
	ACK64_	64ビット転送アクノリッジ	in	out	x	
	PAR64_	パリティ	in/out	in/out	x	
JTAG/バウンダリ・スキャン・	TCK	テスト・クロック	in	in	x	
	TDI	テスト・データ入力	in	in	x	
	TDO	テスト・データ出力	out	out	x	
	TMS	テスト・モード・セレクト	in	in	x	
	TRST_	テスト・リセット	in	in	x	

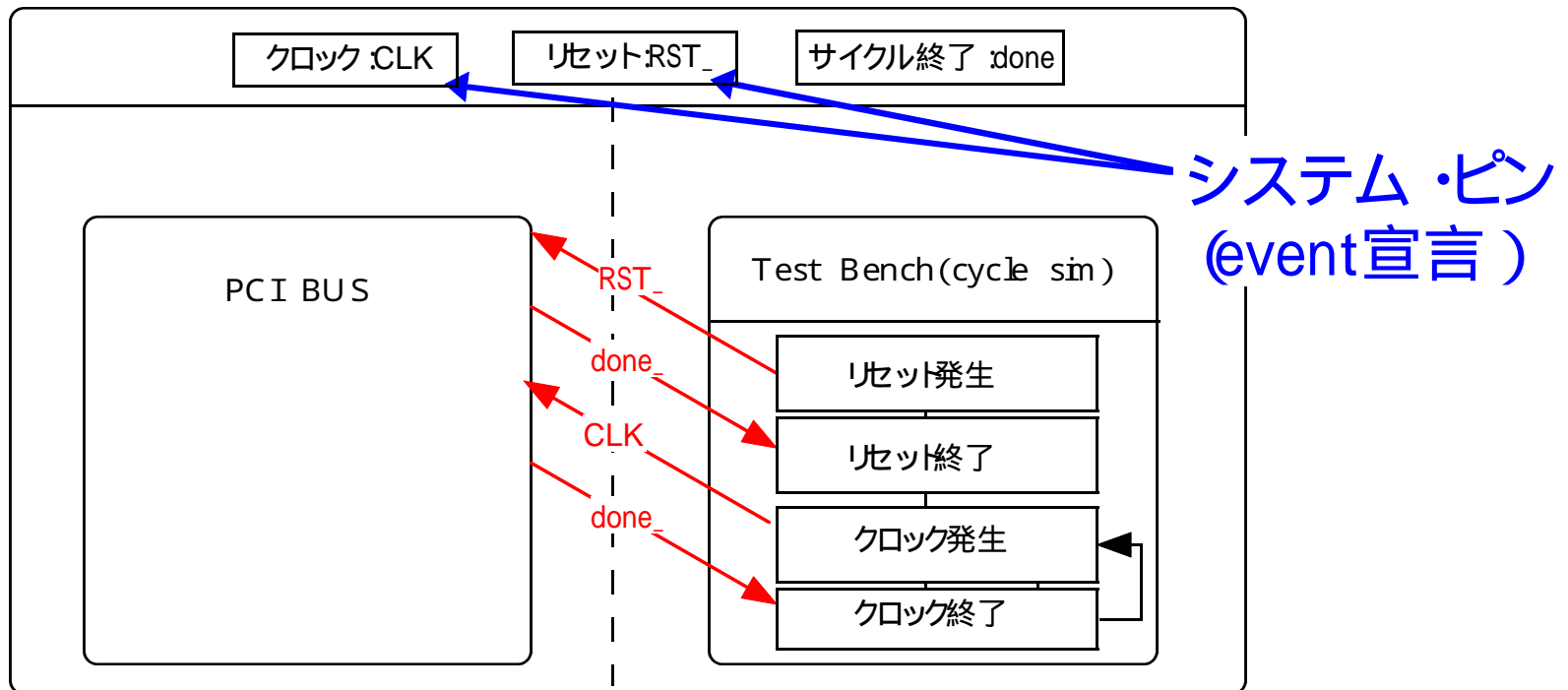
システムの  
チャンネル対象

PCIバス内の  
インタフェース  
信号として  
チャンネル対象

アービタ信号  
としてチャンネル化

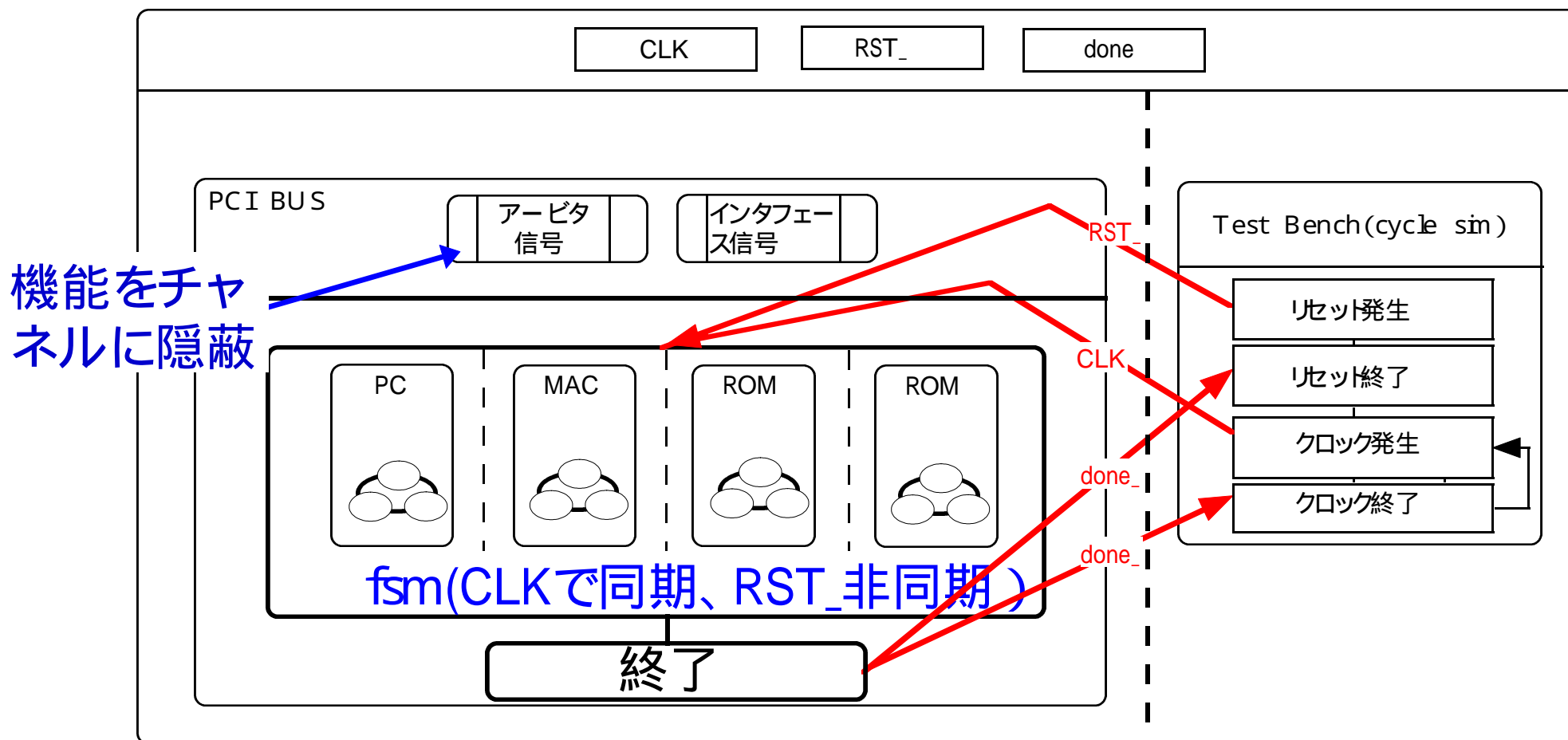
## 3. Current Status (3)

### SpecChart (評価モデル全体構成とテストベクタ)



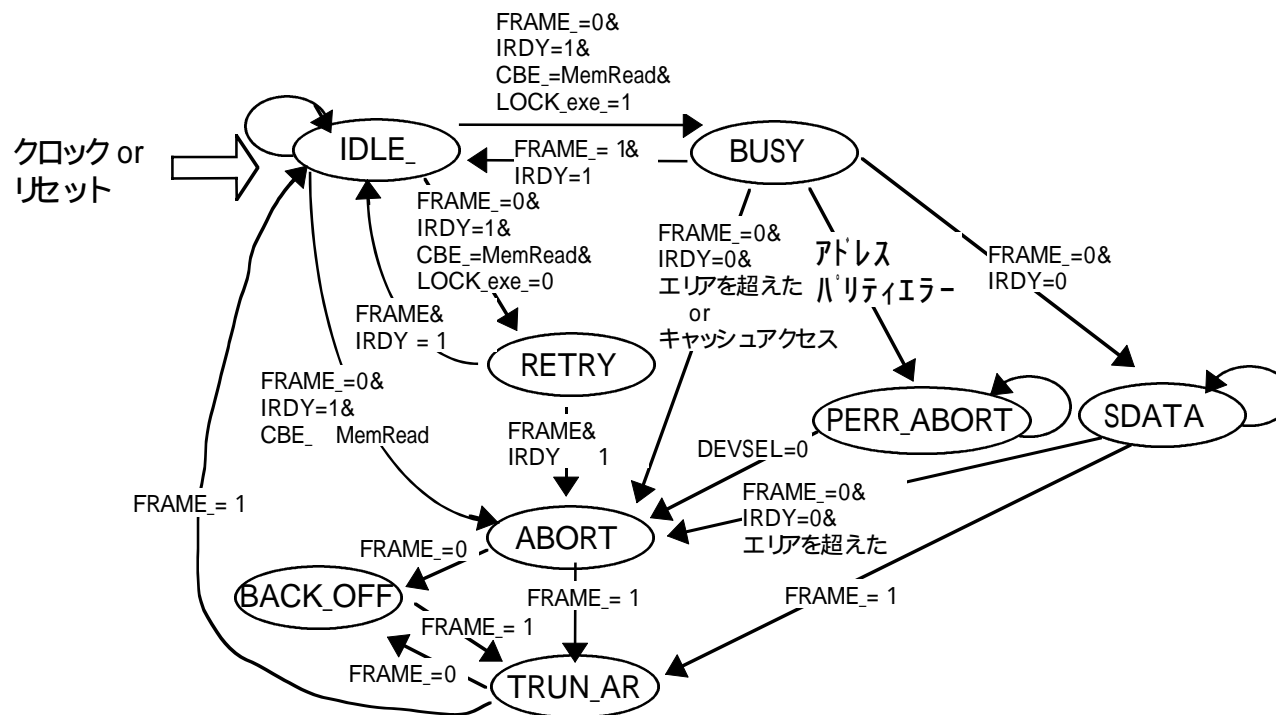
## 3. Current Status (4)

### SpecChart (PCI-BUSモデル全体構成)



## 3. Current Status (5)

### State Diagram (ターゲット・デバイスROM)

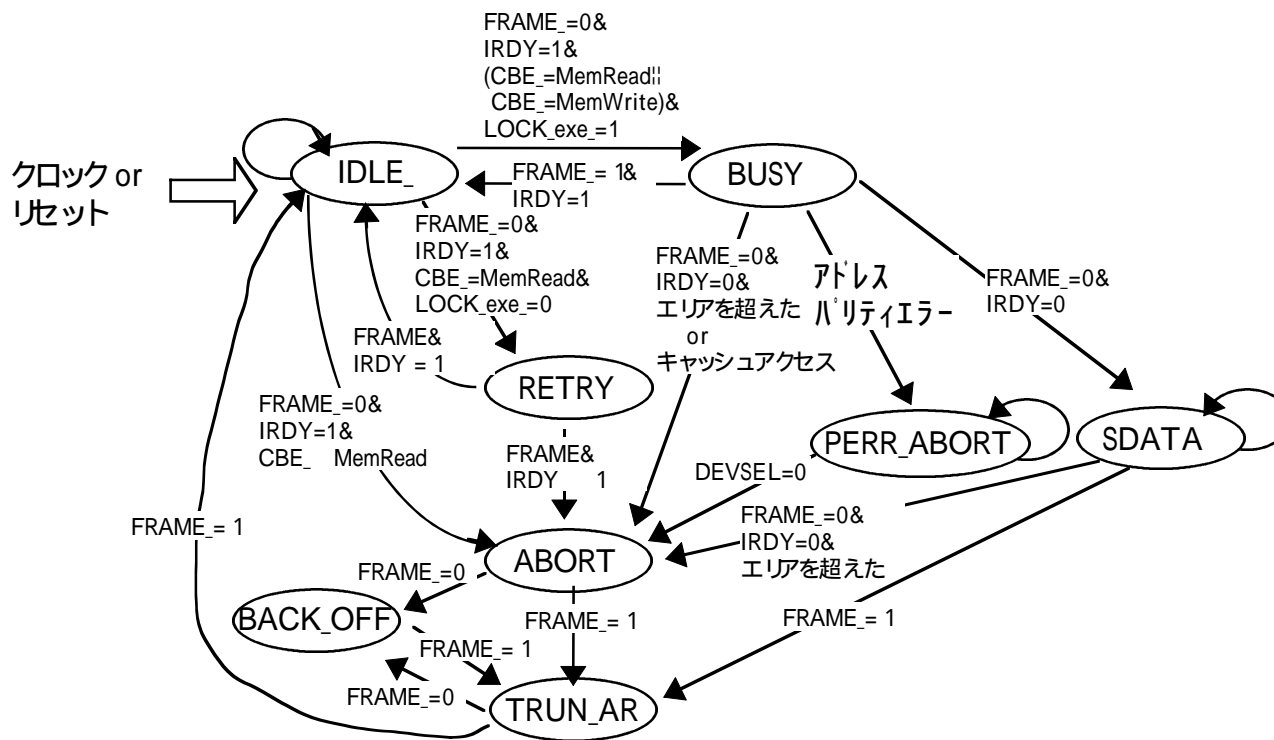


IDLE:アイドル状態  
 BUSY :エージェント切り替え  
 RETRY :リトライ処理  
 PERR\_ABORT :アドレスパリティエラー処理

SDATA:データ転送処理  
 ABORT :ターゲット・イニシエーテッド・ターミネーション処理  
 BACK\_OFF :マスタ終了待ち  
 TRUN\_AR 終了処理

## 3. Current Status (6)

### State Diagram (ターゲット・デバイスRAM)



IDLE: アイドル状態

BUSY : エージェント切り替え

RETRY : リトライ処理

PERR\_ABORT : アドレスパリティエラー処理

SDATA: データ転送処理

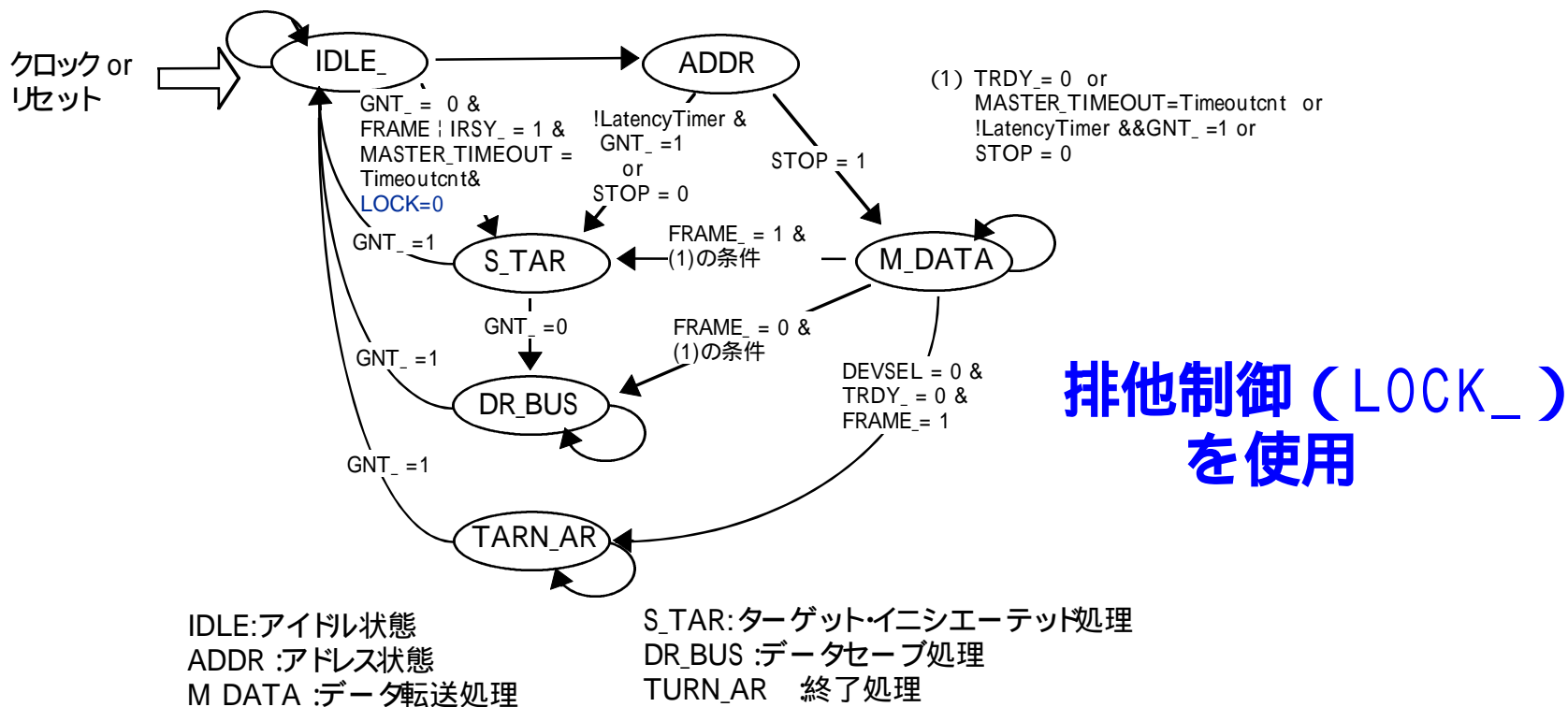
ABORT : ターゲット・イニシエーテッド・ターミネーション処理

BACK\_OFF : マスタ終了待ち

TRUN\_AR 終了処理

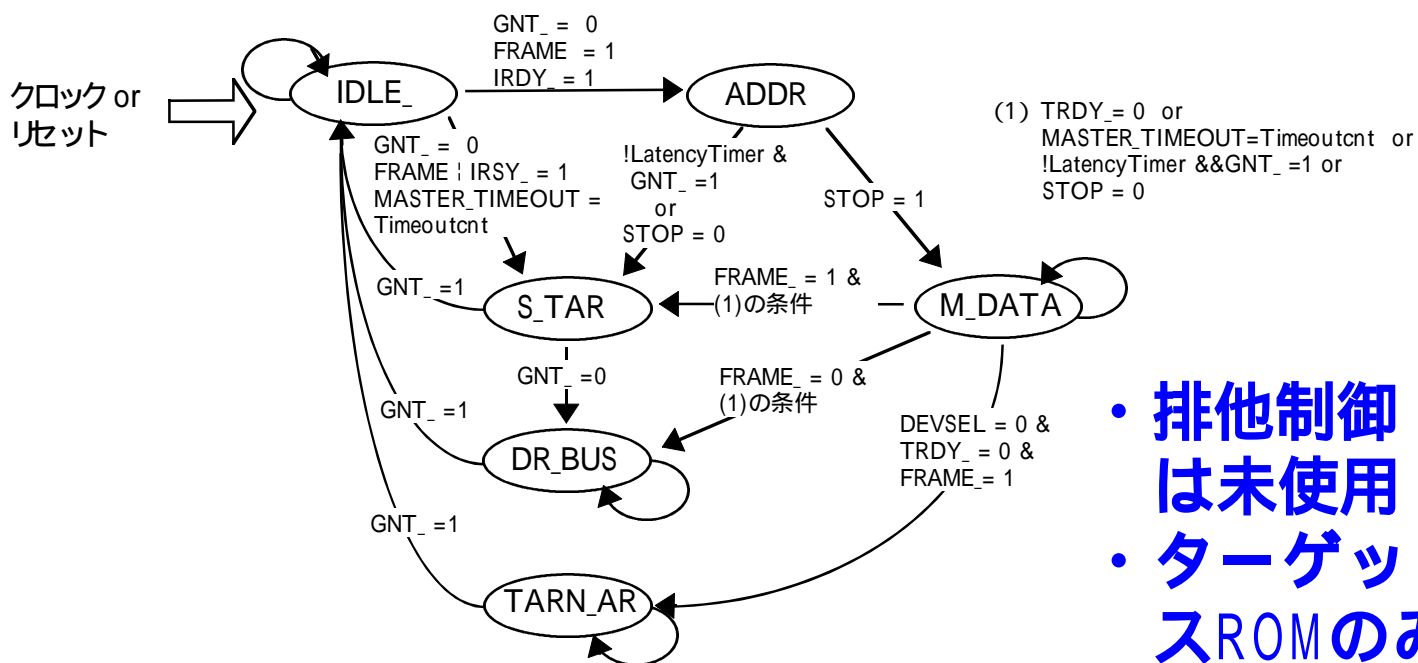
## 3. Current Status (7)

### State Diagram (マスタ・デバイスPC)



## 3. Current Status (8)

### State Diagram (マスタ・デバイスMAC)



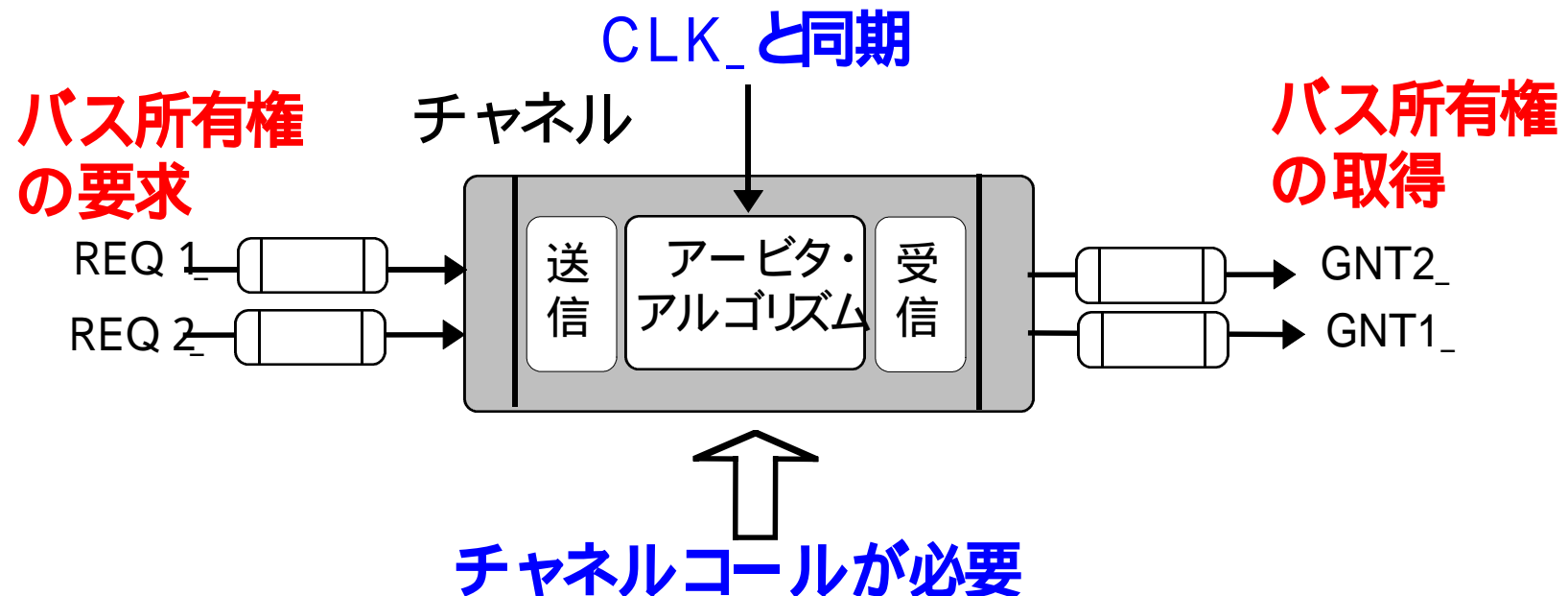
IDLE\_: アイドル状態  
 ADDR : アドレス状態  
 M\_DATA : データ転送処理

S\_TAR: ターゲット・イニシエーテッド処理  
 DR\_BUS : データセーブ処理  
 TARN\_AR : 終了処理

## 3. Current Status (9)

### Channel Structure

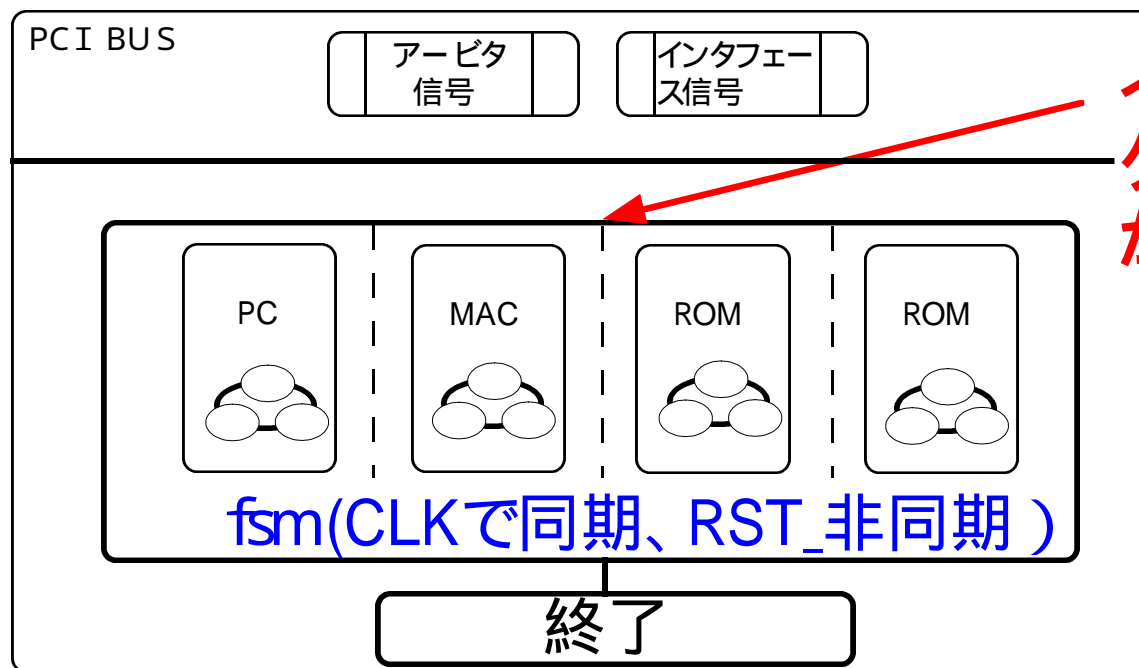
アービタの機能 (アービタアルゴリズム) をチャンネルに隠蔽



## 3. Current Status (10)

Problems in using U.C.I Simulator (同期処理が出来ない)

( SpecChart )



イベント宣言された  
クロックCLKで同期  
が出来ない!!

実行すると . . . . .

同期せずに終了!!

SpecC warning: dead lock in the program, abort

## 3. Current Status (11)

### Problems in using U.C.I Simulator (同期処理が出来ない)

PCI\_BUSメイン

```
behavior PCI_BUS( in event CLK,
                 FromSystem RST_,
                 out event done
                 ) {
    :
    par {
        ROM.main();
        RAM1.main();
        PC.main();
        MAC.main();
    }
}
```

並列実行

クロックで同期

MATSTER\_DEVICE3\_PC

```
behavior MASTER_DEVICE3_PC( in event CLK, //syncro clock
                             FromSystem RST_, //async reset
                             :
                             :
                             while( 1 ) {
                                 wait( CLK );
                                 if ( OwnMaster ) {
                                     if ( LatencyTimer > 0 ) LatencyTimer--;
                                     printf(" LatencyTimer = %d¥n", LatencyTimer );
                                 }
                                 if ( !OE_PAR ) {

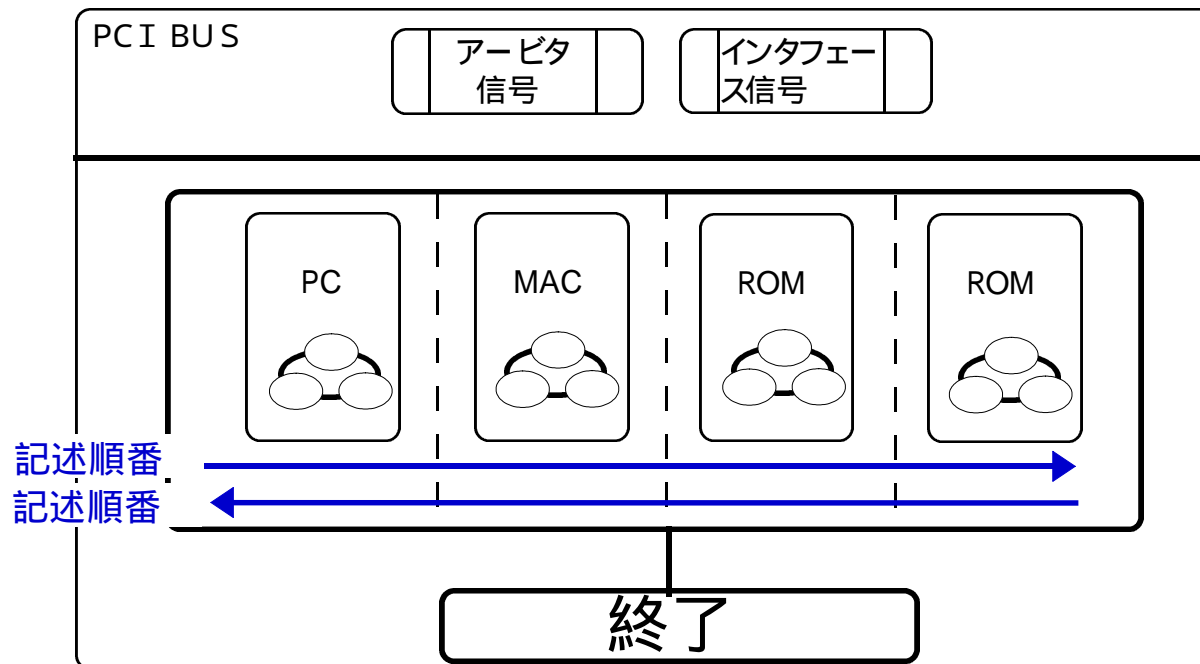
```

( SpecC記述 )

- 同期しない理由は、以下が考えられる
- ( 1 ) 記述誤り 仕様を勘違いしている
  - ( 2 ) SpecCの不具合

## 3. Current Status (12)

### Problems in using U.C.I Simulator (並列動作できない)



**記述順番、 を実行すると . . . . 結果が異なる！！**  
**並列処理を行うときを考えると致命的！！**  
**並列実行の時には、全てのBehaviorの処理完了後、**  
**各Behaviorの出力値を決定してほしい**

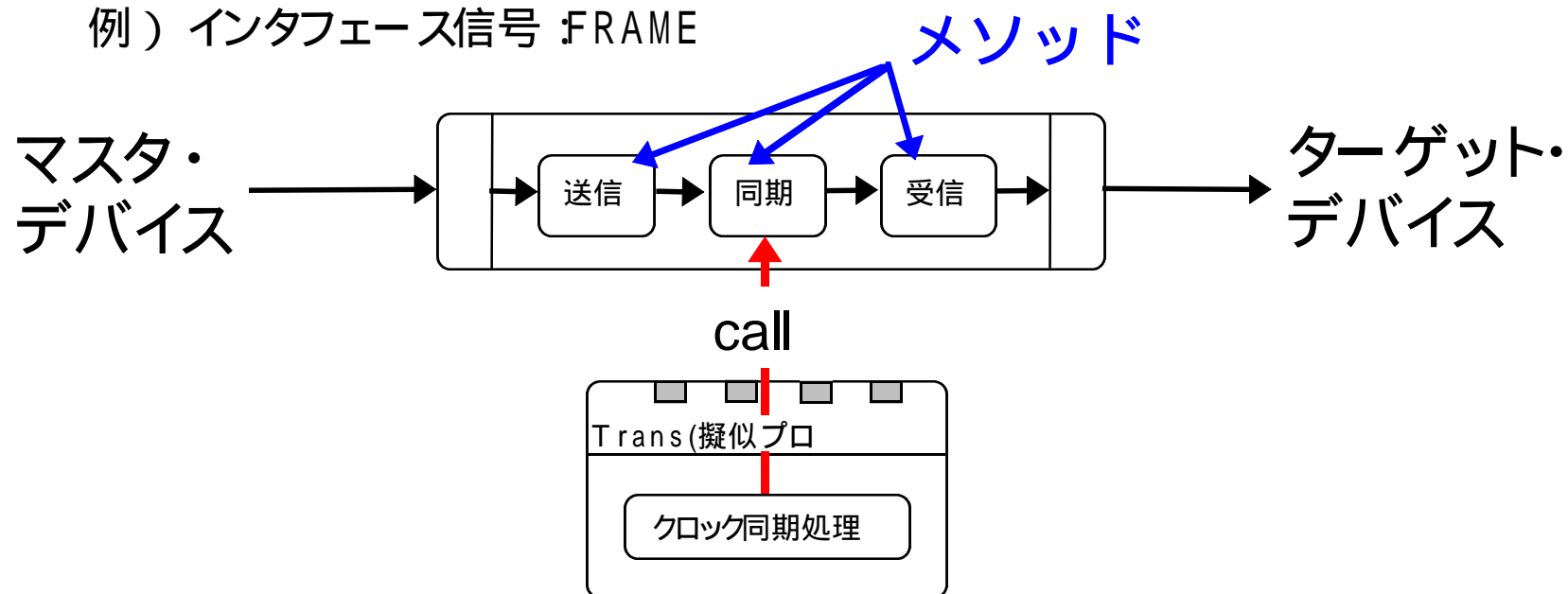


## 3. Current Status (14)

### Work Around (並列動作対策)

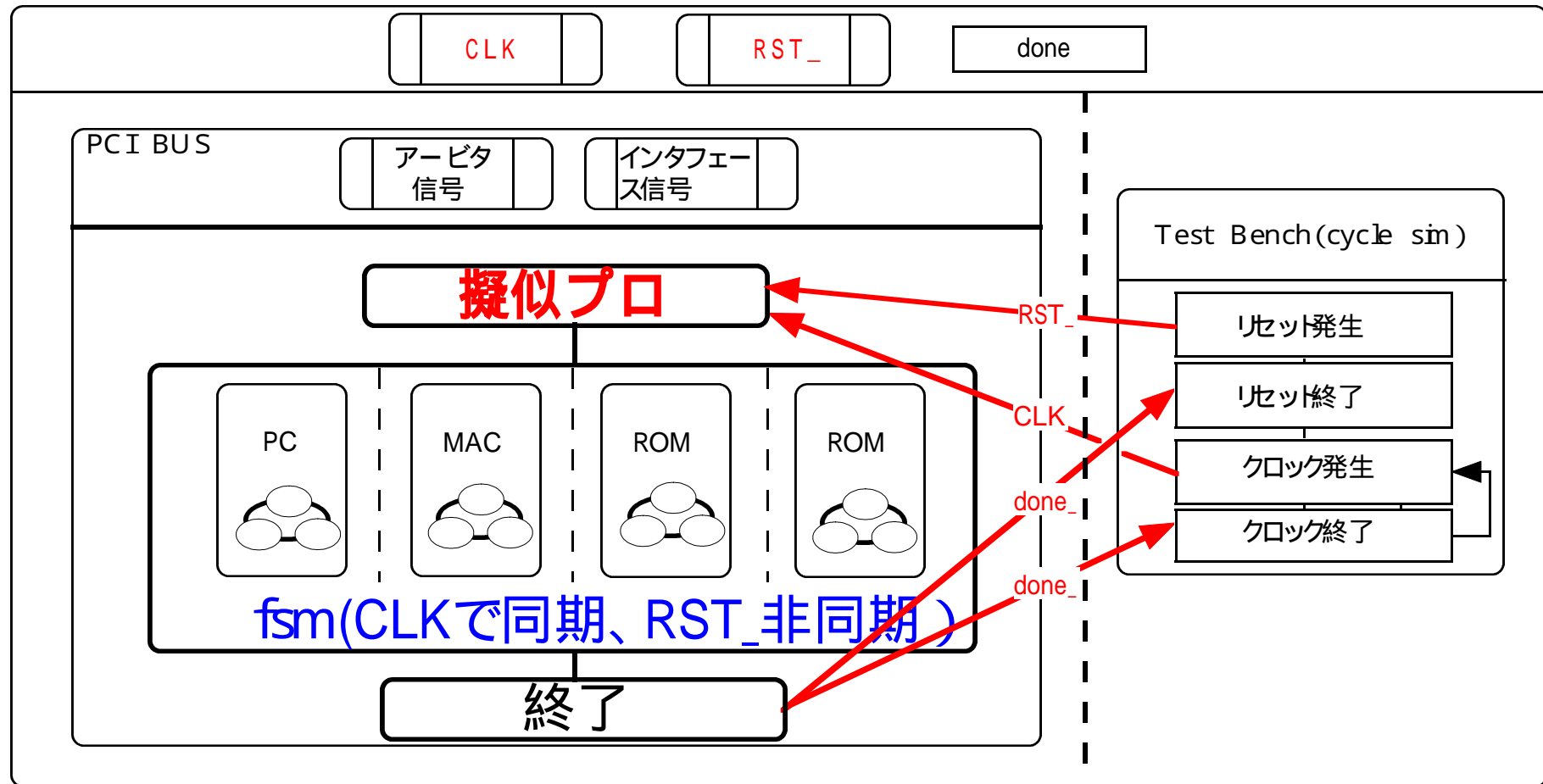
チャンネル内に同期をとるためのメソッドと擬似プロを追加し、  
擬似プロ内で同期メソッドをコールする  
PCIバス上の各信号線をクロックと同期して値を伝播する  
ようにした

例) インタフェース信号 : FRAME



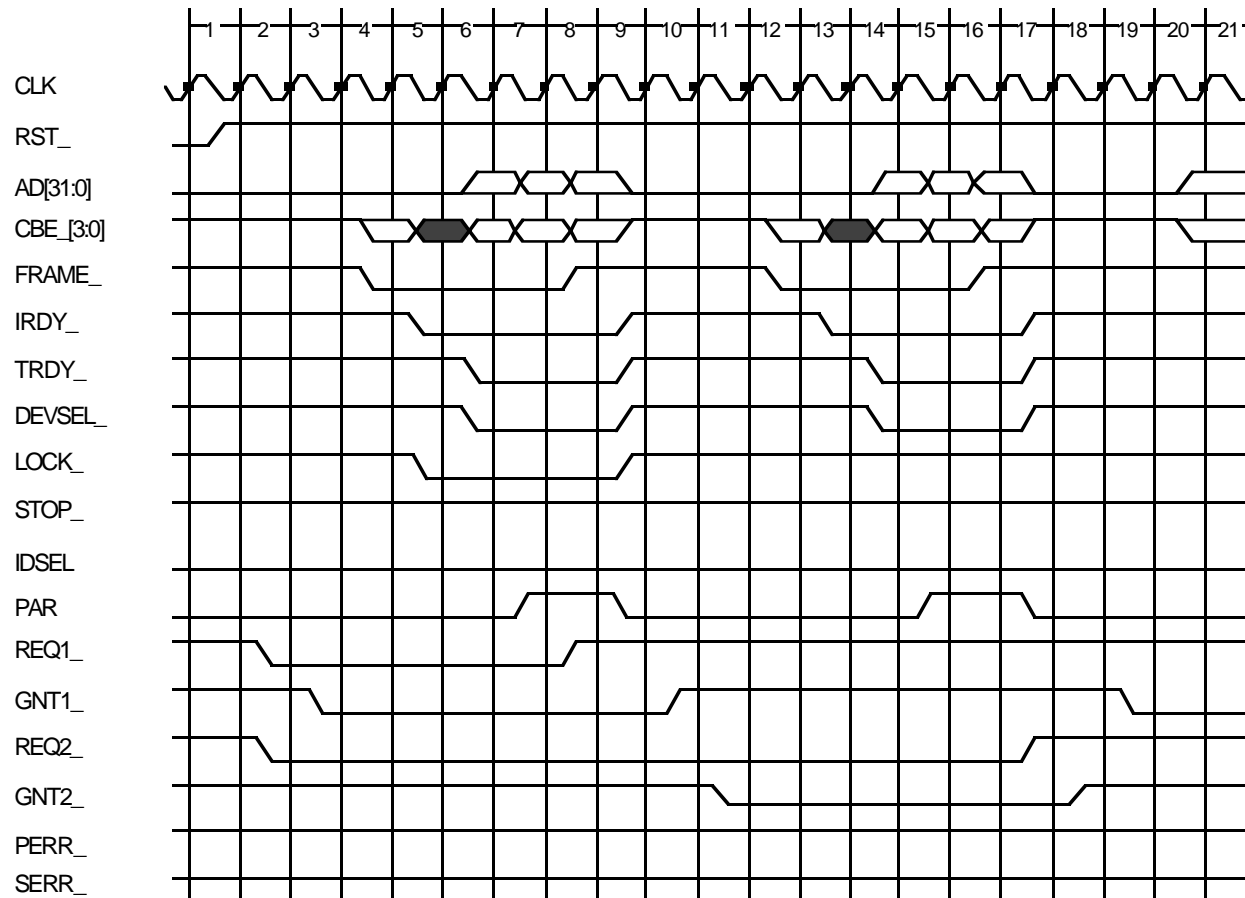
## 3. Current Status (15)

### Work Around (PCI-BUSモデル全体構成 :対策後)



## 3. Current Status (16)

### Timing Diagram (リード・バースト転送)



## 4. Conclusion (1)

### PCI-BUSによるSpecC評価まとめ

#### (1) 同期 (イベント制御) について

SpecCでは、notify, waitというイベント制御文があり、Verilogで言えば、以下の対応する。

(SpecC)	(Verilog)
notify	"->" or "信号の変化"
wait	@

Verilogでは、1 イベントに対し複数の文で拾いだす事ができるが、SpecCは、1対1である。

現在のSpecC (V205)では、PCI-BUSのような同期バスの場合には、エッジトリガを認識するような記述が必要。

ハードウェア設計者が好んで用いる、信号エッジをトリガとする動作を如何に記述すべきかを調査中。

#### (2) 並列動作について

今回、並列動作の命令語par文を用いている。しかし、並列動作は、行っていない。

現在のSpecC (V205)では、並列処理はないと思ったほうが無難。

## 4. Conclusion (2)

### SpecCを使用して見て

#### (1) 記述について

ハードウェア設計者がSpecC記述を作成するのは困難だと考えられます。

ハードウェア設計者は、データパスとステートマシン、及び回路ブロック図、タイミングダイアグラムから設計をスタートさせるため、より上位の概念でのuntimedな状態での並列処理の通信機構を意識する事に不慣れです。また、C++やC言語にも不慣れです。従って、C++の知識とSpecC特有の処理をmixさせるのは、時間がかかると思います。

SpecCを始める前の教育マニュアル(例えば、並列処理の通信機構の分類とその記述例)などが必要だと感じます。

ただ、高位レベルという概念から、今後、もっと充実していけば面白いと感じました。

#### (2) シンタクス・セマンティクス

シンタクスエラーチェックが大変弱いです。

bit宣言など、C++に付け加えた命令語は、使用法を誤ると、c言語に変換したときエラーとなる場合があります。変換されたc言語を解釈しないと、判らないときがあります。

また、暗黙のキャスト変換が未定義の様であり、その対策が困難です。

現在のSpecCでは、ソフトウェアの経験と感が必要です。

#### (3) デバック

SpecCは、ANSI Cをベースに作成されていますが、所詮はc言語です。

配列の使用誤りなど、メモリ破壊を起したとき、設計者は、デバック出来ないのではないのでしょうか？