FUJITSU

shaping tomorrow with you

SystemVerilogで ”まとめる” 検証環境

2012.09.28 Verify2012

富士通マイクロソリューションズ株式会社
技術開発統括部　設計技術開発部
鈴木　晃一（suzuki.koich-01@jp.fujitsu.com）

Copyright 2012 FUJITSU Microelectronics Solutions Limited

会社紹介

- 概要
- 当社デザインサービス

Copyright 2012 FUJITSU Microelectronics Solutions Limited

会社概要

社名	**富士通マイクロソリューションズ株式会社（FMSL）** （FUJITSU Microelectronics Solutions Limited） http://jp.fujitsu.com/group/fmsl/
本社	横浜市港北区新横浜2-10-23（野村不動産新横浜ビル)
事業所	仙台事業所、大阪事業所
設立	2003年10月1日
事業内容	最先端半導体テクノロジーをベースとした、大規模システムLSIのハードウェアとソフトウェアの開発及びLSIを搭載したシステムソリューションの提供。 ・デジタルAV用LSI・CPU内蔵SoC・車載システム用LSI ・アナログLSI・無線系LSI・ワイヤレスデバイス・ネットワークソリューション
社員数	597名（2012年3月20日現在）

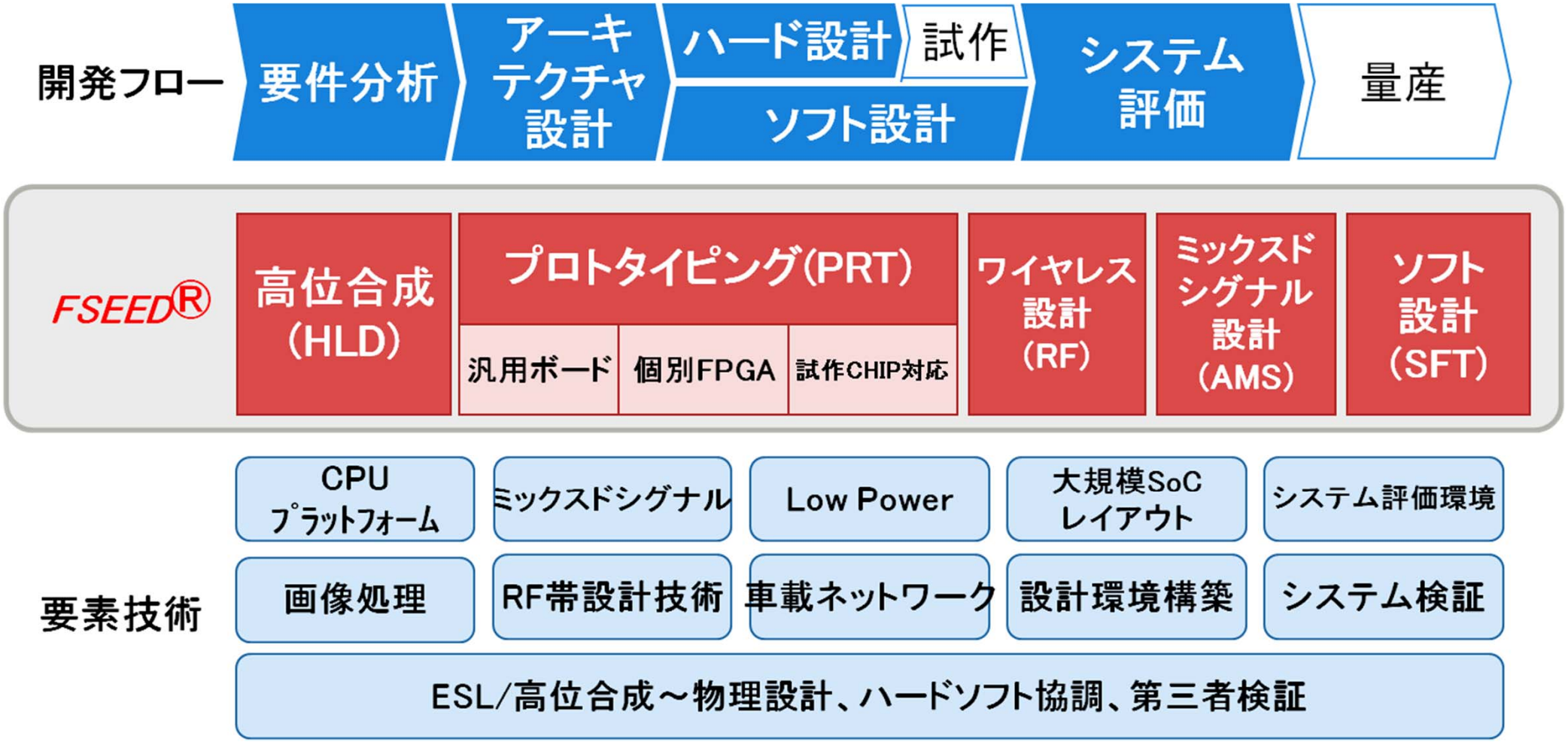
富士通セミコンダクター株式会社グループの
設計・開発専門の会社

Copyright 2012 FUJITSU Microelectronics Solutions Limited

デザインサービス FSEED®

FUJITSU

ハードウェア（ディジタル、アナログ、検証環境）からソフトウェア（ファーム、ミドルウェア）まで
最先端SoC設計ソリューションでお客様の製品開発を加速します



問い合わせは弊社HomePageへどうぞ
http://jp.fujitsu.com/group/fmsl/contact/

Copyright 2012 FUJITSU Microelectronics Solutions Limited

FUJITSU

検証

(Verification)

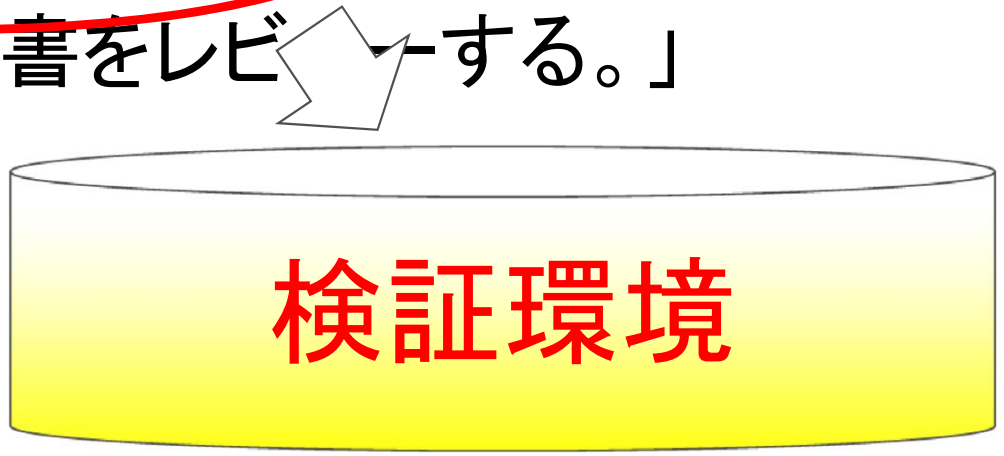
Copyright 2012 FUJITSU Microelectronics Solutions Limited

検証（Verification） - ISO9000:2005より

■「客観的証拠を提示することによって、
規定要求事項が満たされていること。

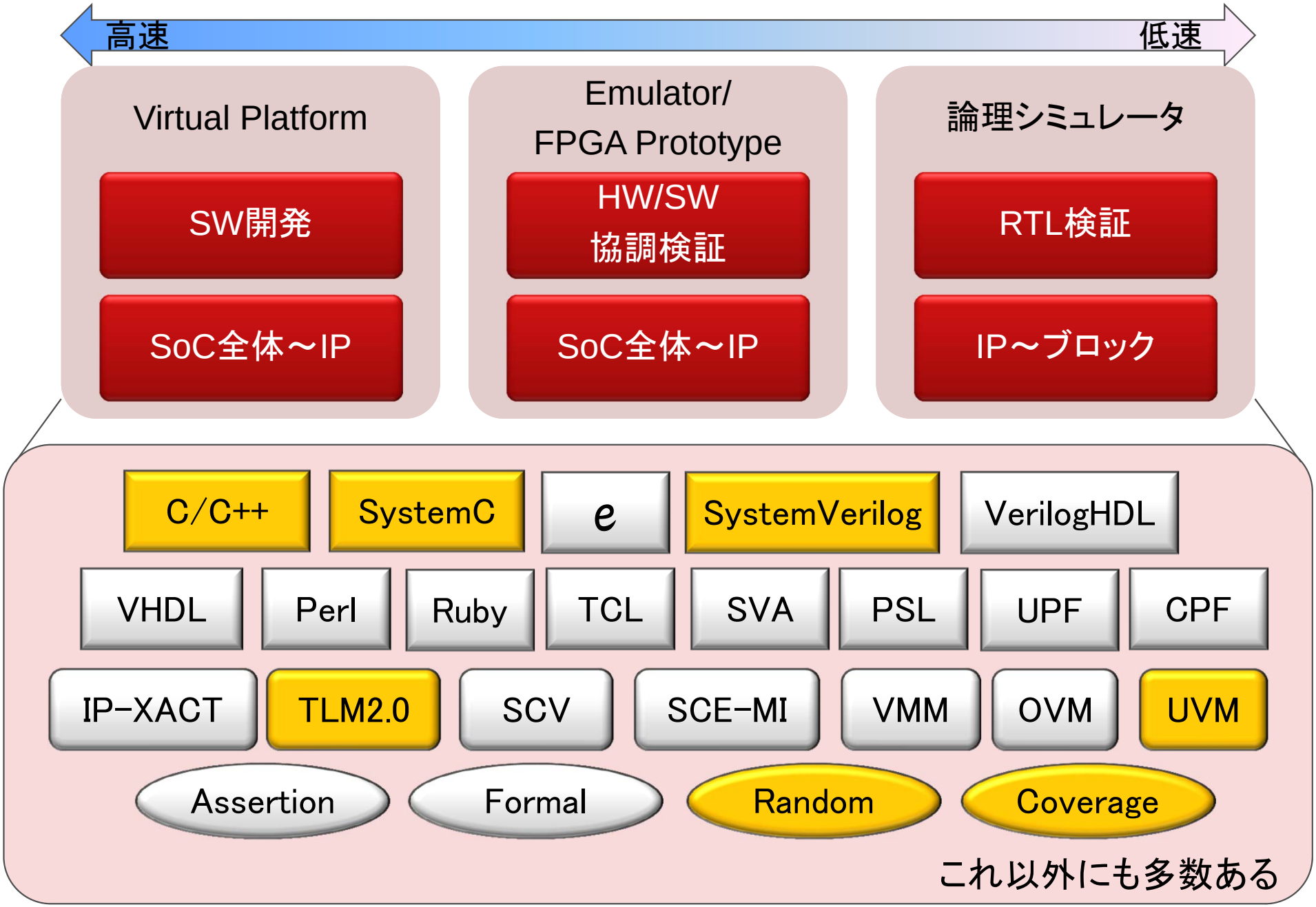
注記2：確認には、次のような活動があり得る。

- 別法によって計算を実施する。
- 新しい設計仕様書を類似の証明済みの設計仕様書と比較する。
- 試験及び実証を行う。
- 発行前に文書をレビューする。」



Copyright 2012 FUJITSU Microelectronics Solutions Limited

目的や規模に応じて様々な検証環境が存在



Copyright 2012 FUJITSU Microelectronics Solutions Limited

目的や規模に応じて様々な検証環境が存在

FUJITSU

- 環境構築にあたって様々な手段（ツールや言語）がある
  - 目的に合わせて最適な選択を行う
- 検証技術は日々進化している
  - トレンド把握や今ある検証環境に改善を考えてみる

今回トライアルした検証環境を紹介します

Copyright 2012 FUJITSU Microelectronics Solutions Limited

今回紹介する検証環境

アルゴリズム（C/C++）との検証

- DPI-C+RTL検証環境

TLMモデルとの検証

- TLM/RTL検証環境

ソフトウェアとの検証

- Virtual Platform+RTL検証環境

Copyright 2012 FUJITSU Microelectronics Solutions Limited

FUJITSU

Direct Programming Interface（DPI/DPI-C）検証環境

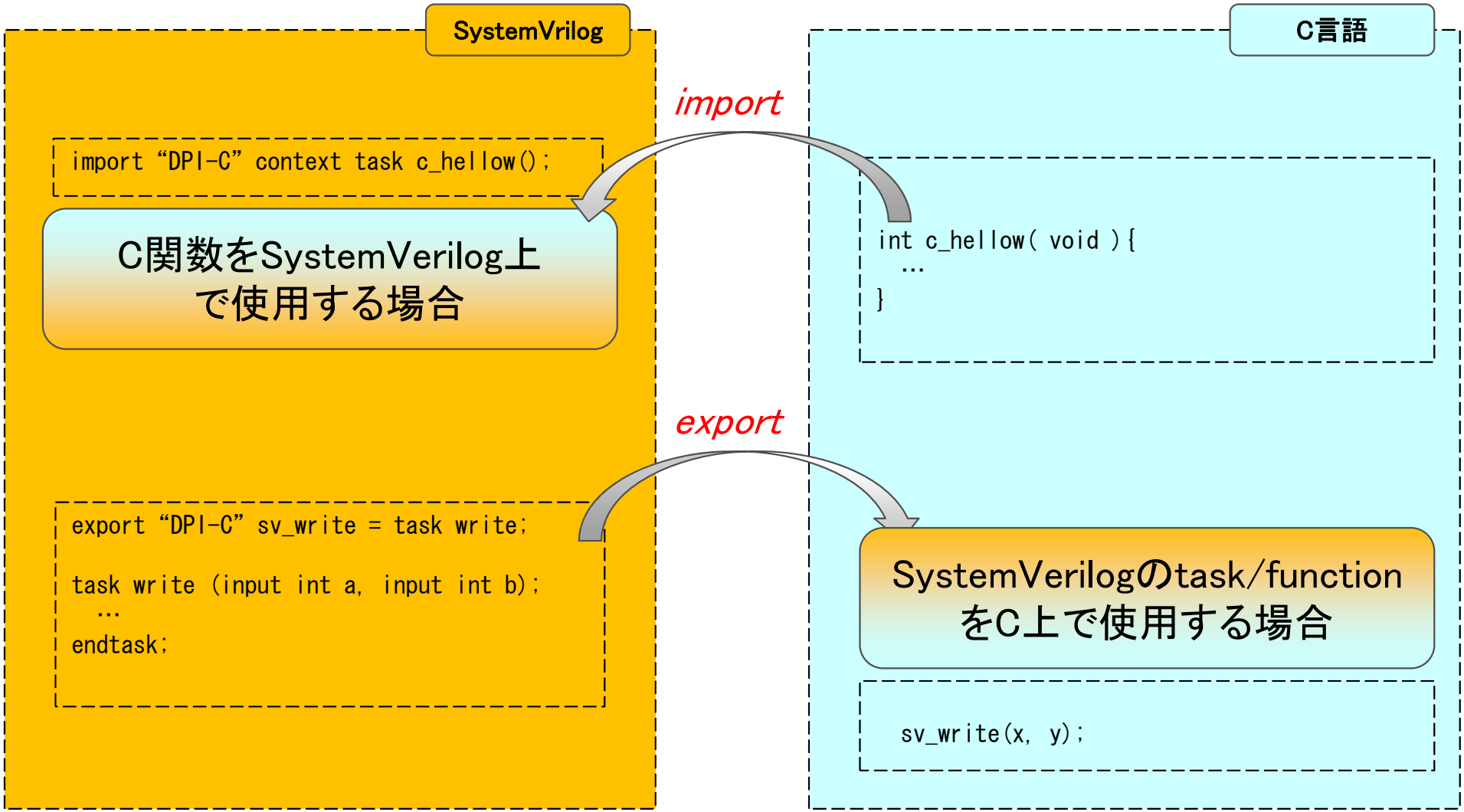
- DPI（DPI-C）とは
- DPIを使った利点
- DPI-Cを使った検証環境紹介

Copyright 2012 FUJITSU Microelectronics Solutions Limited

Direct Programming Interface（DPI）とは

FUJITSU

- DPIを介して他言語との接続が可能
  - DPI-C、（C++、SystemC）

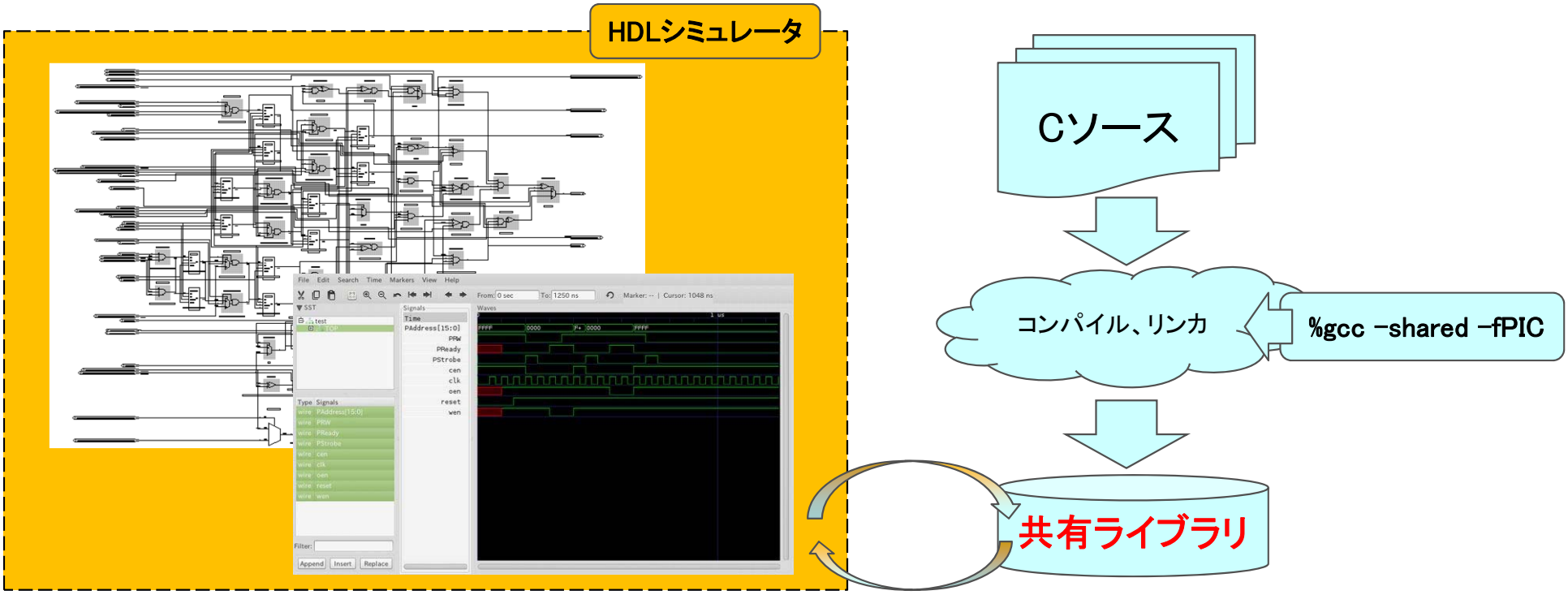


Copyright 2012 FUJITSU Microelectronics Solutions Limited

DPI-Cの動作イメージ

FUJITSU

■ 共有ライブラリ化して、HDLシミュレータが呼び出す。



■ 参考：シミュレーション時間（簡易テストベンチ）

環境	SystemVerilogのみ	SystemVerilog+DPI-C
Sim時間	126.6秒	93.3秒

Copyright 2012 FUJITSU Microelectronics Solutions Limited

DPI-Cを使う利点

利点1:テストプログラムを”まとめる”

利点2:アルゴリズムからの一貫フロー

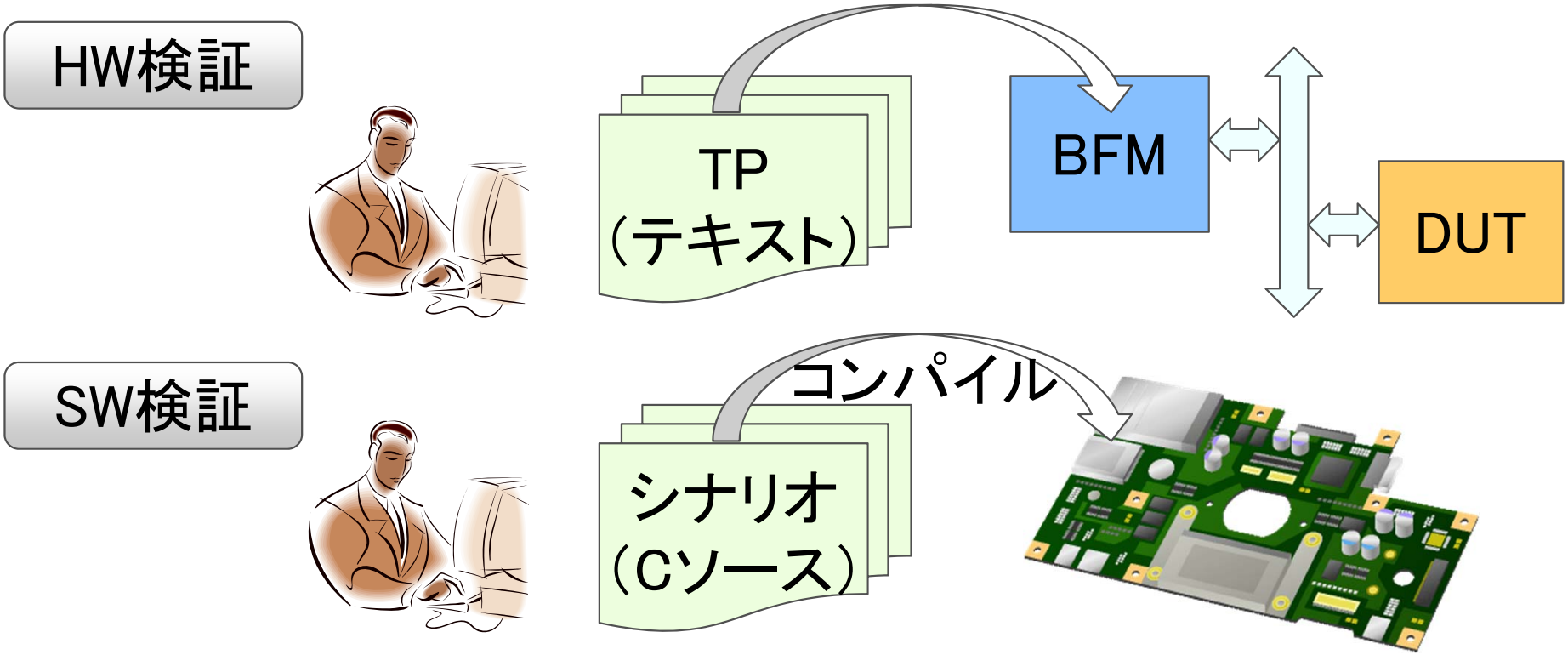
Copyright 2012 FUJITSU Microelectronics Solutions Limited

利点1：テストプログラムを”まとめる”（1/3）

FUJITSU

■Case1：BFM（Bus Function Model）で検証

- ○ HW検証速度が速い
- × ソフト開発で流用できない



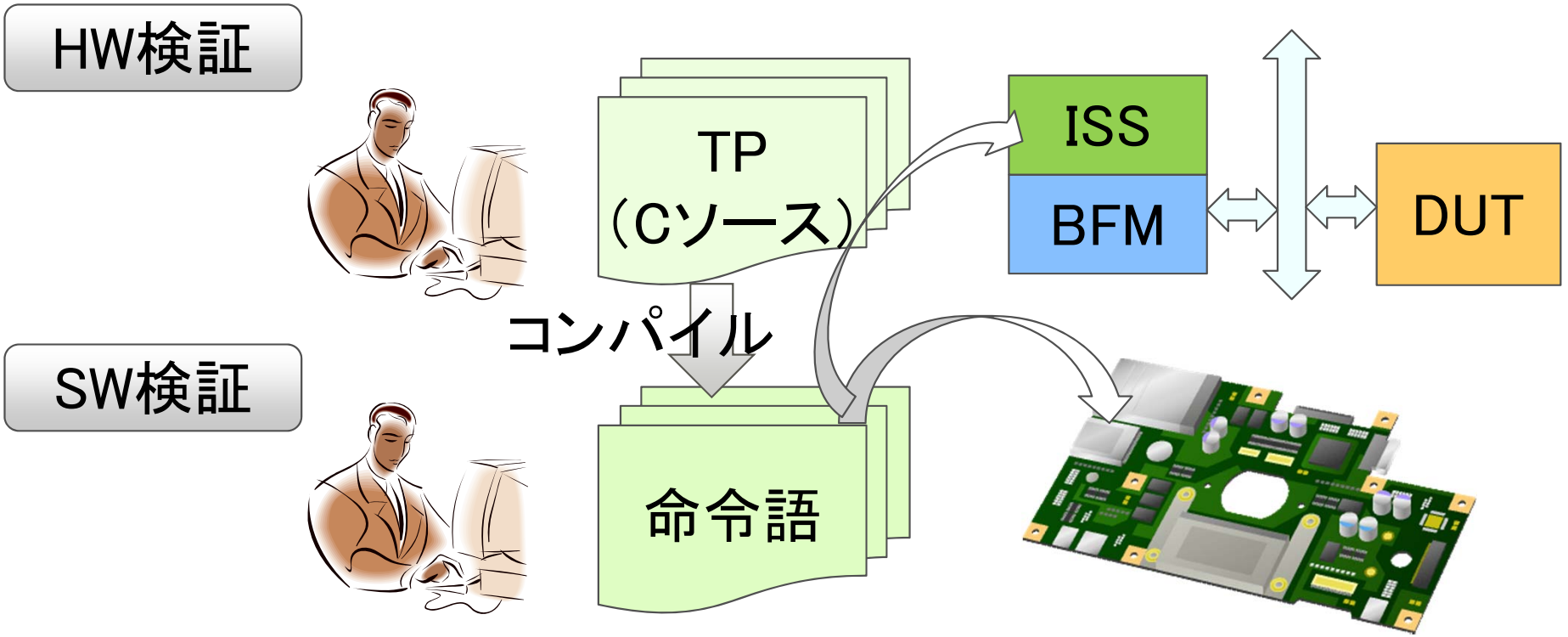
Copyright 2012 FUJITSU Microelectronics Solutions Limited

利点1：テストプログラムを”まとめる”（2/3）

FUJITSU

■Case2：ISS（Instrucstion Set Simulator）で検証

- × HW検証速度が遅い
- ○ ソフト開発で流用できる



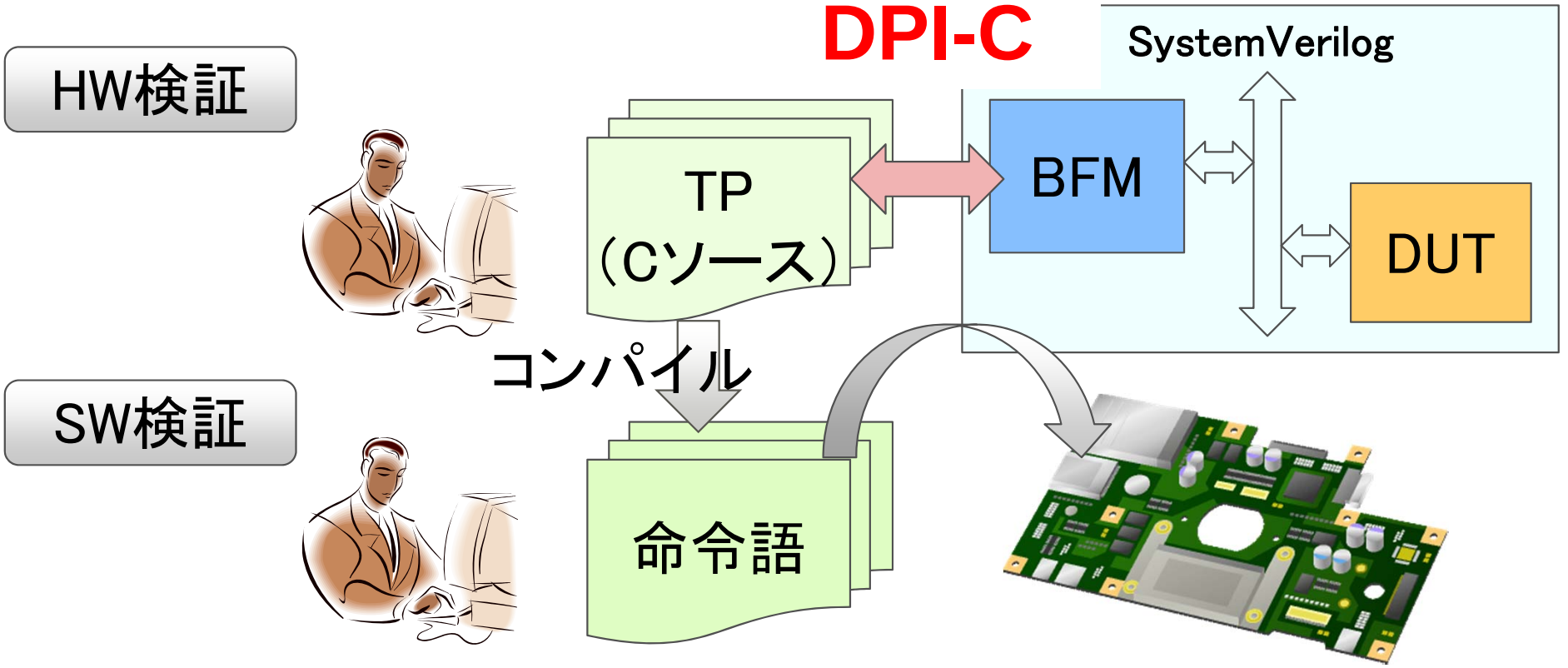
Copyright 2012 FUJITSU Microelectronics Solutions Limited

利点1：テストプログラムを”まとめる”（3/3）

FUJITSU

■Cソース（テストプログラム）とBFMをDPIで接続

- ○ HW検証速度が早い
- ○ ソフト開発で流用できる

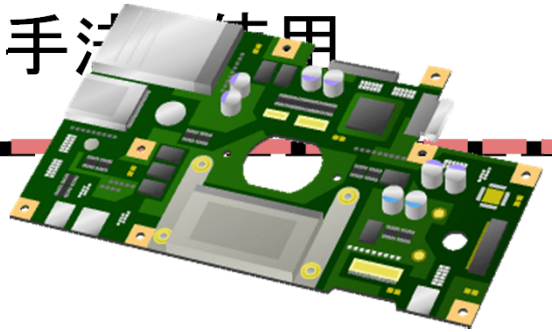


Copyright 2012 FUJITSU Microelectronics Solutions Limited

利点2:アルゴリズムからの一貫フロー

FUJITSU

- RTL検証環境からEmulator検証環境が”まとまる”
  - ゴリズムの複雑化により高位合成手法を使用



C/C++

SystemVerilog

DUT (RTL)

Emulator
トランザクタ

DPI-C

Emulator
トランザクタ

一致比較

C結果

RTL
結果

➢ DPI-C部分変更で簡単にEmulatorへ実装

Copyright 2012 FUJITSU Microelectronics Solutions Limited

FUJITSU

TLM/RTL検証環境

- 概要
- TLMモデルにUVMを適用する利点
- UVMを使ったTLM/RTL検証環境紹介

Copyright 2012 FUJITSU Microelectronics Solutions Limited

デザイン大規模化・アルゴリズム複雑化

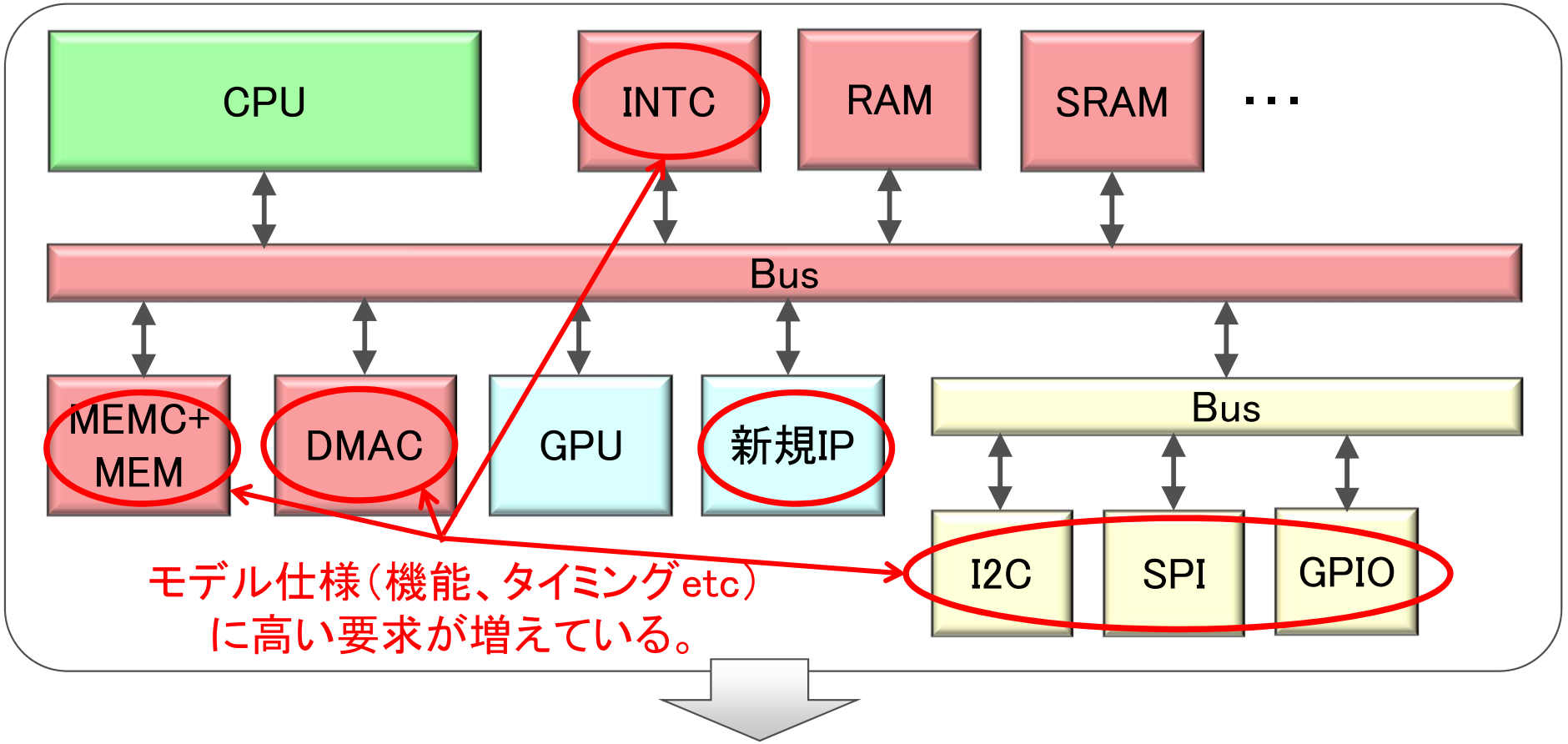
FUJITSU

Copyright 2012 FUJITSU Microelectronics Solutions Limited

Virtual Platform環境の構築

FUJITSU

■ 先行ソフトウェア開発、アーキテクチャ評価



従来行なっていたTLMモデルの検証環境を改善したい

RTL検証工程まで考慮した検証環境が出来ないか？

TLM：Transaction Level Modeling

Copyright 2012 FUJITSU Microelectronics Solutions Limited

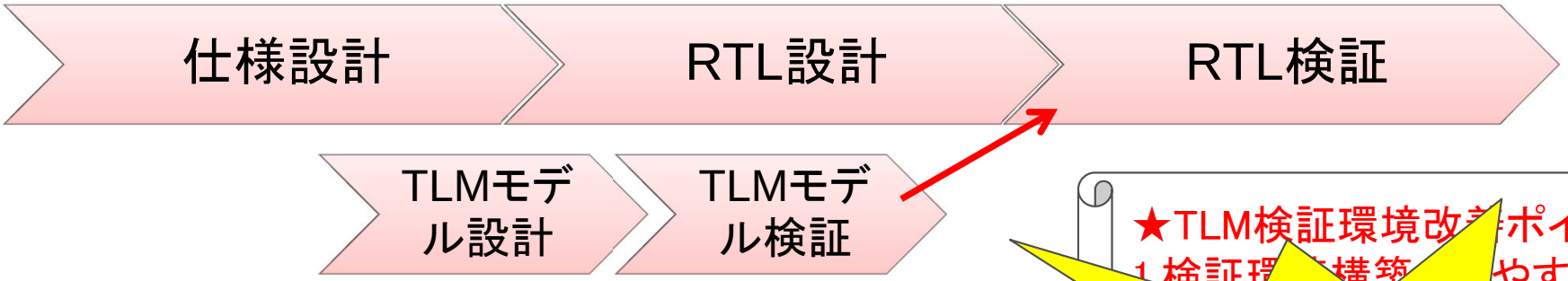
これまでのTLM/RTL検証環境

FUJITSU

■それぞれ個別の環境を作成

	TLM検証環境	RTL検証環境
主言語	SystemC（C/C++）	VHDL/VerilogHDL/ SystemVerilog/e

■TLM/RTL開発パターン①



■TLM/RTL開発パターン②

Copyright 2012 FUJITSU Microelectronics Solutions Limited

Universal Verification Methodology（UVM）

FUJITSU

- ツールに依存しない業界標準の検証メソドロジ
  - オープンソースで無償で利用可能
  - 検証環境、検証IP（VIP）の再利用が容易
  - カバレッジドリブン検証環境構築のしやすさ
    - rand,制約（costraint）
    - アサーション、プロパティ
    - 機能カバレッジ

TLMモデル検証環境にも適用出来ないか？

http://www.uvmworld.org/

Copyright 2012 FUJITSU Microelectronics Solutions Limited

UVMでTLMモデルを検証できるのか？

FUJITSU

- UVMはTLMの基本構成が含まれています。

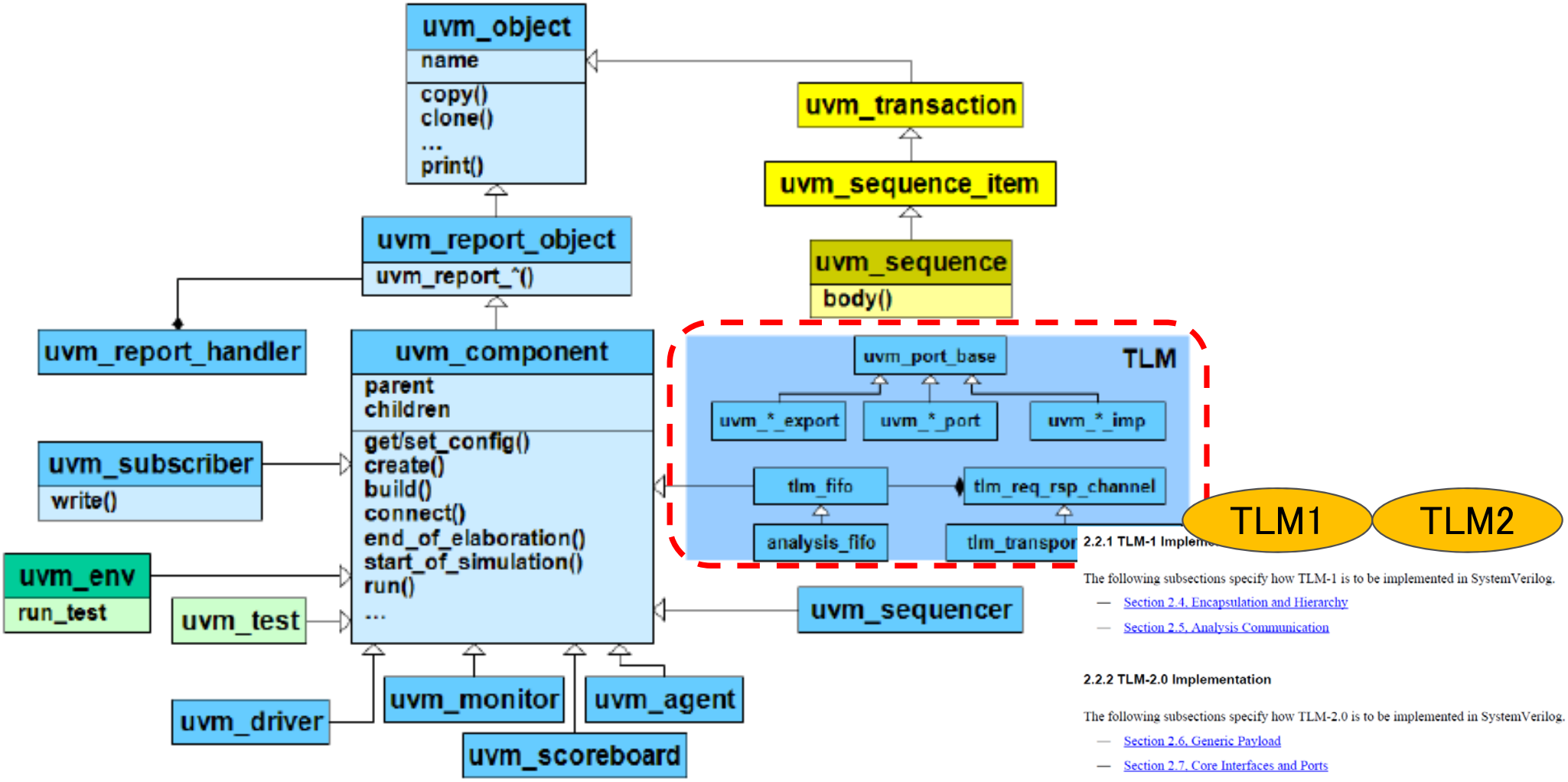


Figure 3—(Partial) UVM Class Hierarchy

2.2.1 TLM-1 Implem

The following subsections specify how TLM-1 is to be implemented in SystemVerilog.

- Section 2.4, Encapsulation and Hierarchy
- Section 2.5, Analysis Communication

2.2.2 TLM-2.0 Implementation

The following subsections specify how TLM-2.0 is to be implemented in SystemVerilog.

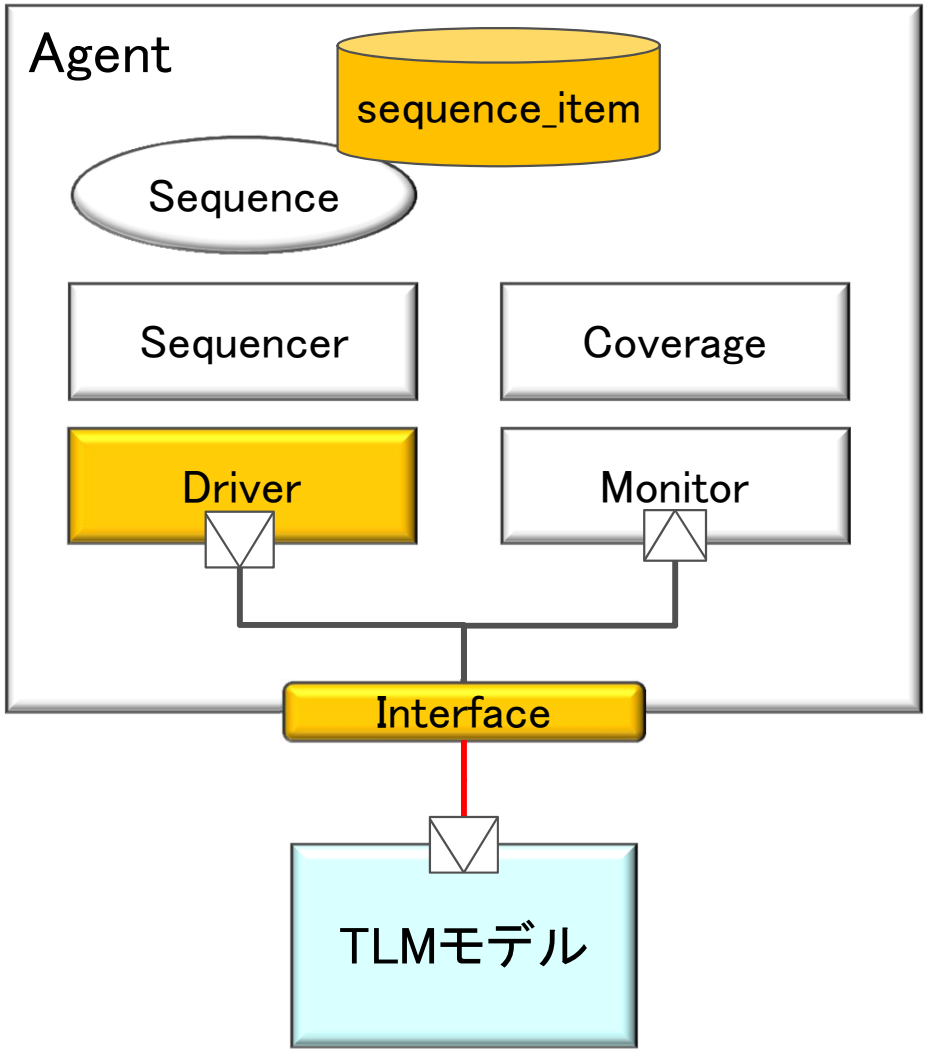
- Section 2.6, Generic Payload
- Section 2.7, Core Interfaces and Ports
- Section 2.8, Blocking Transport
- Section 2.9, Nonblocking Transport
- Section 2.10, Sockets
- Section 2.11, Time
- Section 2.12, Use Models

Copyright 2012 FUJITSU Microelectronics Solutions Limited

UVMの構成要素（Agent）

FUJITSU

■TLM検証環境において、キーとなる部分



■sequence_item

■検証データの中身

➢効率的な検証できる？

■Driver（Interface）

■DUT（検証対象物）との接続

➢TLMモデルとどう接続する？

Copyright 2012 FUJITSU Microelectronics Solutions Limited

UVM（TLM2）を使うと効率的か？

FUJITSU

■TLMのデータ型は予めライブラリで「rand」宣言済み

2.6 Generic Payload

TLM-2.0 defines a base object, called the *generic payload*, for moving data between components. In SystemC, this is the primary transaction vehicle. In SystemVerilog, this is the default transaction type, but it is not the only type that can be used (as will be explained more fully in Section 2.7).

簡単にランダム検証

2.6.1 Attributes ※一部誤りがありますので、ご注意ください

Each attribute in the SystemC version has a corresponding member

```
protected rand bit [63:0]   m_address;
protected rand uvm_tlm_command_e          m_com
protected rand byte                    m_data[];
protected rand int unsigned            m_length;
protected rand uvm_tlm_response_status_e  m_res
protected rand bit
protected rand byt
protected rand int
protected rand int
```

```
import uvm_pkg::*;
`include "uvm_macros.svh"

module test();

  uvm_tlm_gp gp = new;

  initial begin
    for(int i=1; i<2; i++) begin
      assert(gp.randomize() with {
          gp.m_length  inside {[1:10]};
          gp.m_byte_enable_length == 0;
          gp.m_data.size()        == gp.m_length;
          gp.m_streaming_width    == gp.m_length;
      });
      `uvm_info("Generate",{"¥n",gp.sprint()},UVM_MEDIUM)
    end
    $finish();
  end

endmodule: test
```

```
UVM_INFO test.sv(17) @ 0: reporter [Generate]
----------------------------------------------------------------------
Name                  Type                      Size  Value
----------------------------------------------------------------------
<unnamed>             uvm_tlm_generic_payload   -     @4921
  address             integral                  64    'h4b3a2417a0eb0075
  command             uvm_tlm_command_e         32    UVM_TLM_WRITE_COMMAND
  response_status     uvm_tlm_response_status_e 32    UVM_TLM_INCOMPLETE_RESPONSE
  streaming_width     integral                  32    'h4
  data                darray(byte)              4     -
    [0]               byte                      8     'h1f
    [1]               byte                      8     'hee
    [2]               byte                      8     'hc0
    [3]               byte                      8     'h55
----------------------------------------------------------------------
```

メッセージも簡単に表示！

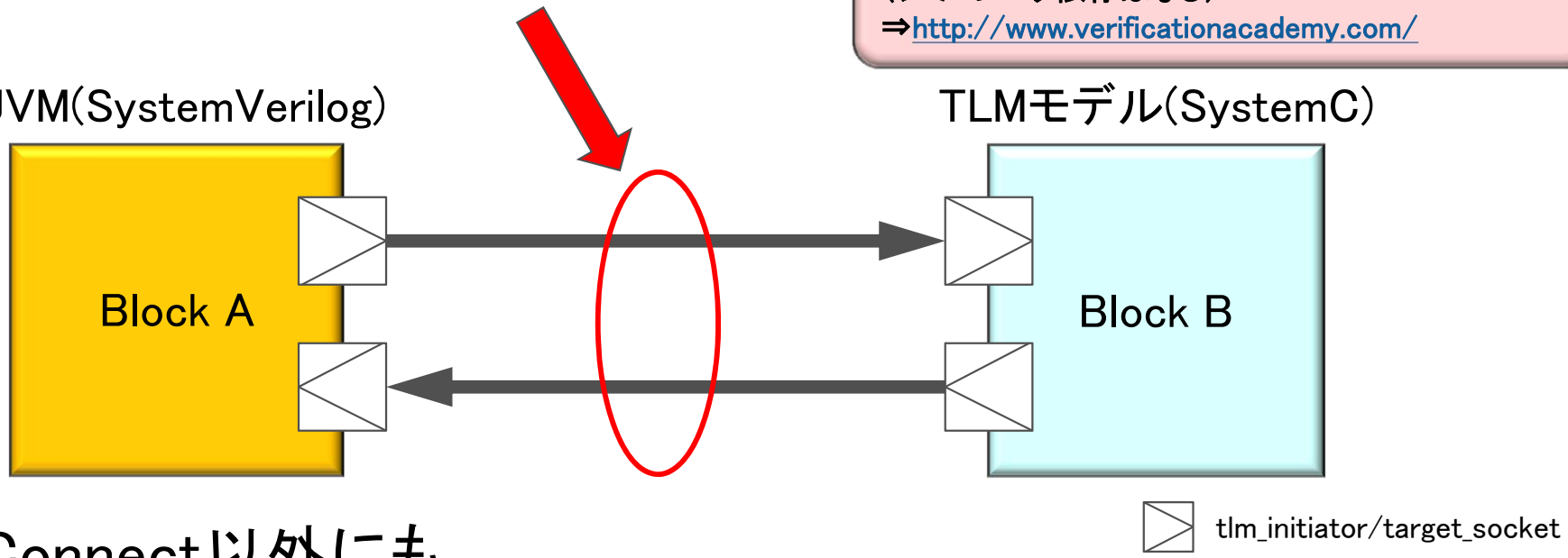
Copyright 2012 FUJITSU Microelectronics Solutions Limited

TLMモデルとどう接続するか？

FUJITSU

- UVM（SystemVerilog）とTLMモデル（SystemC）接続は
  - 今回は UVM Connect を使用

Mentor Graphics社からオープンソースとして提供
（シミュレータ依存はなし）
⇒http://www.verificationacademy.com/



UVM Connect以外にも

- VCS Built-in TLI connectivity for UVM to SystemC TLM 2.0

  http://www.vmmcentral.org/vmartialarts/2012/09/vcs-built-in-tli-connectivity-for-uvm-to-systemc-tlm-2-0/

- Cadence社から、e-TLM2接続に関するホワイトペーパー

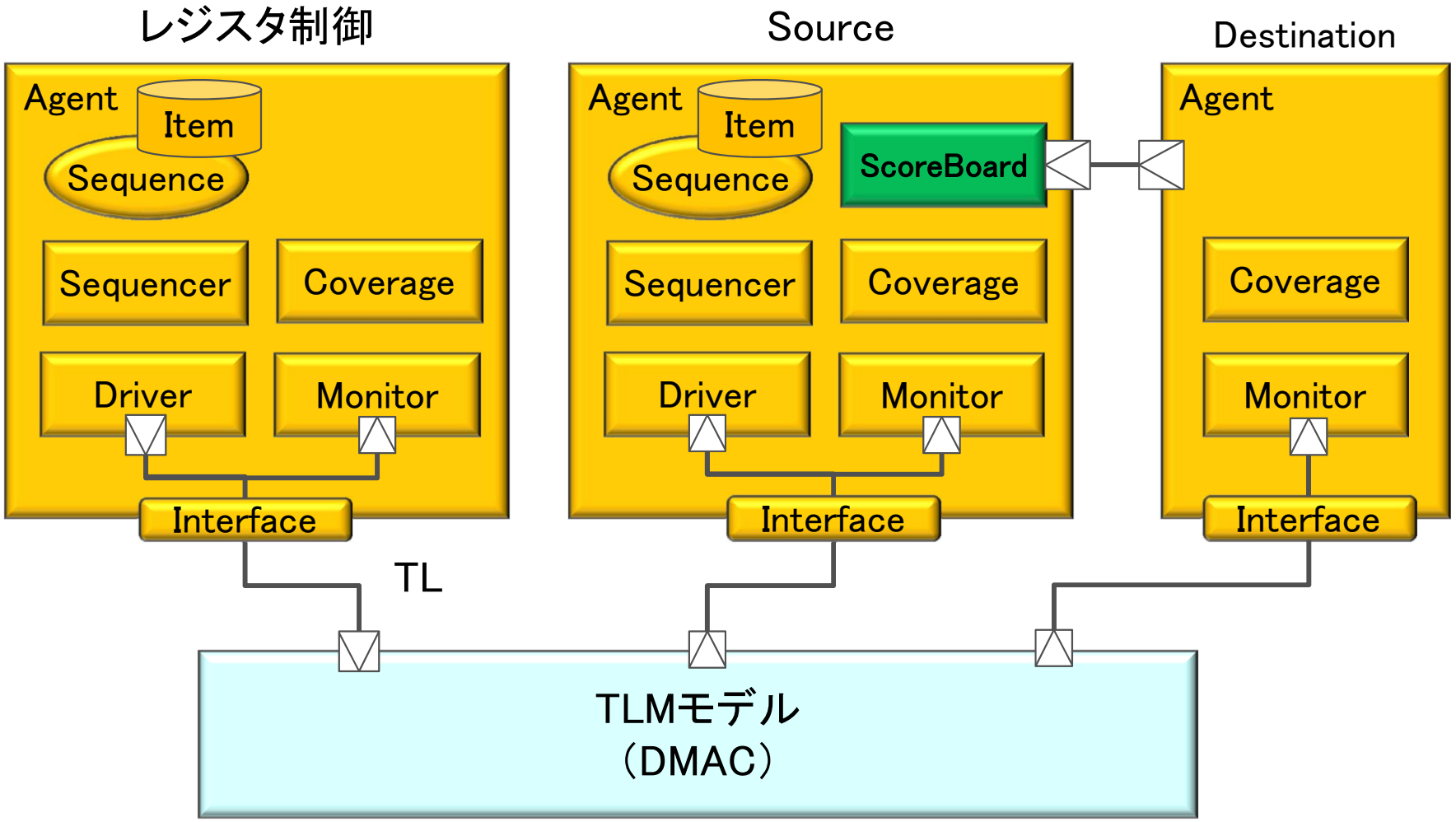
  https://docs.google.com/viewer?url=http%3A%2F%2Fwww.cadence.com%2Frl%2FResources%2Fwhite_papers%2FeTLM2_Interface_wp.pdf

Copyright 2012 FUJITSU Microelectronics Solutions Limited

TLM/RTL検証環境例（DMAC）

FUJITSU

■UVMに沿って検証構築

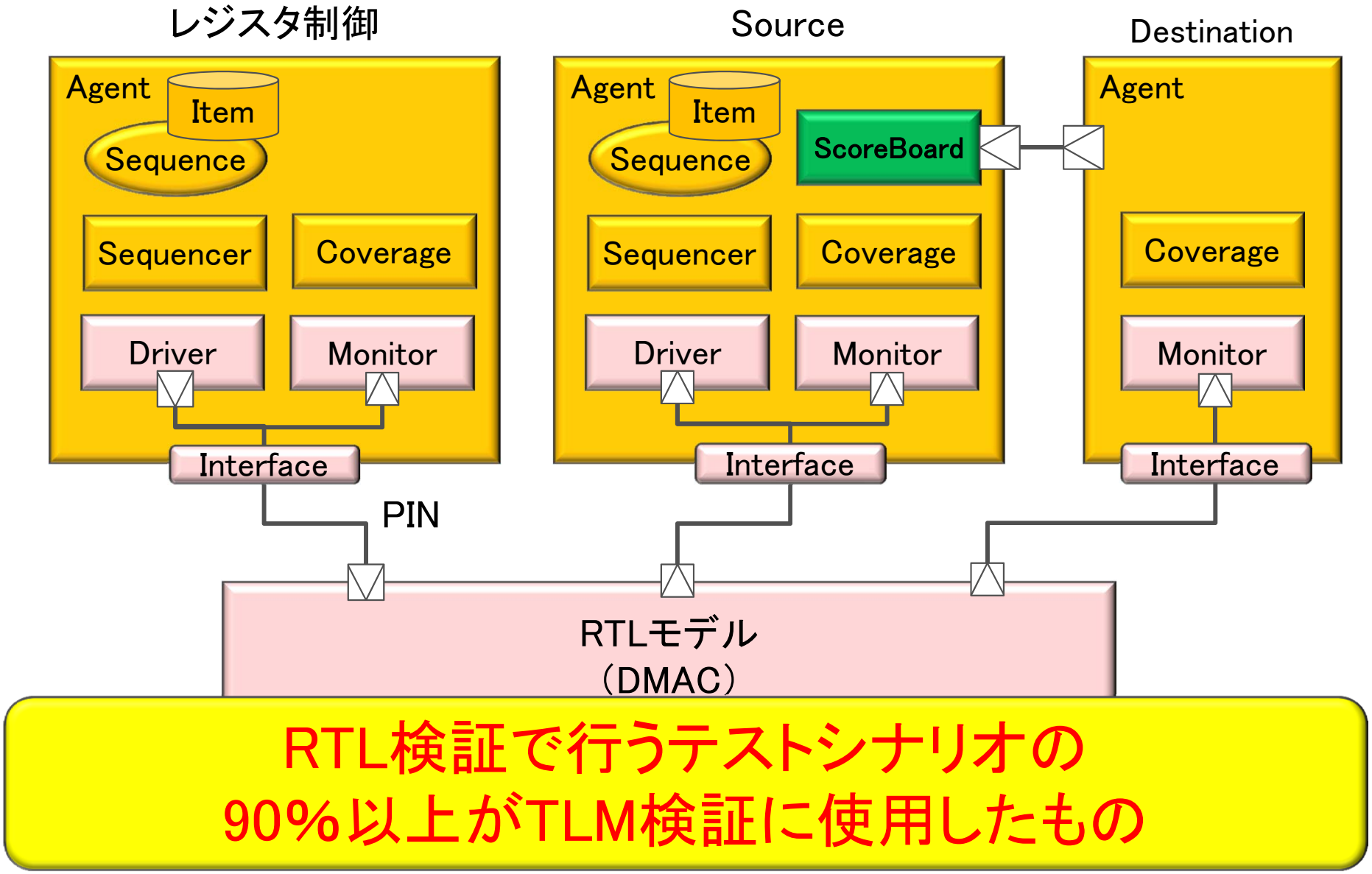


Copyright 2012 FUJITSU Microelectronics Solutions Limited

TLM/RTL検証環境例（DMAC）

FUJITSU

■TLM検証環境からRTL検証環境への変換も一部



Copyright 2012 FUJITSU Microelectronics Solutions Limited

TLM/RTL検証環境トライアル結果

FUJITSU

良かった点

- TLM/RTL検証環境の共通化
- テストシナリオが容易にランダム化
- デバッグ効率化（TL/RTLが同波形上に表示）
- HW技術者が容易に導入可能（SystemCより慣れているHDL）

もう一歩な点

- HDLシミュレータでSystemCのコンパイルが遅い気がする
- シミュレーションエラー時にSEGVが多かった。
- tlm_generic_payload「extension」の使用は非推奨
- uvm_sequence_itemの継承は注意が必要

Copyright 2012 FUJITSU Microelectronics Solutions Limited

Virtual Platform+RTL検証環境

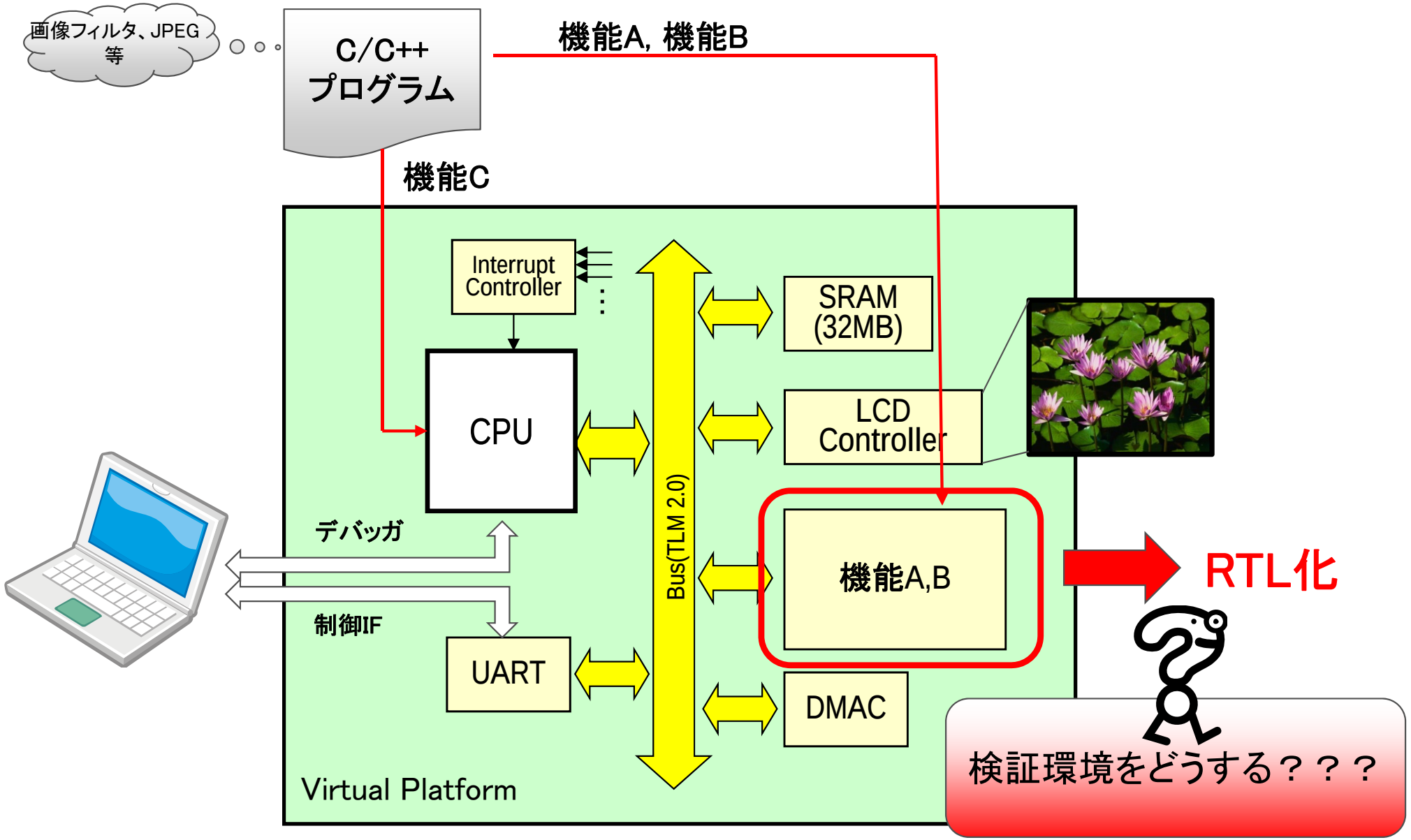
- 評価システム概要
- Virtual Platform+RTL検証環境紹介
- 従来の検証環境との比較結果

Copyright 2012 FUJITSU Microelectronics Solutions Limited

評価システム概要

FUJITSU

Virtual Platformから高位合成しRTLを生成

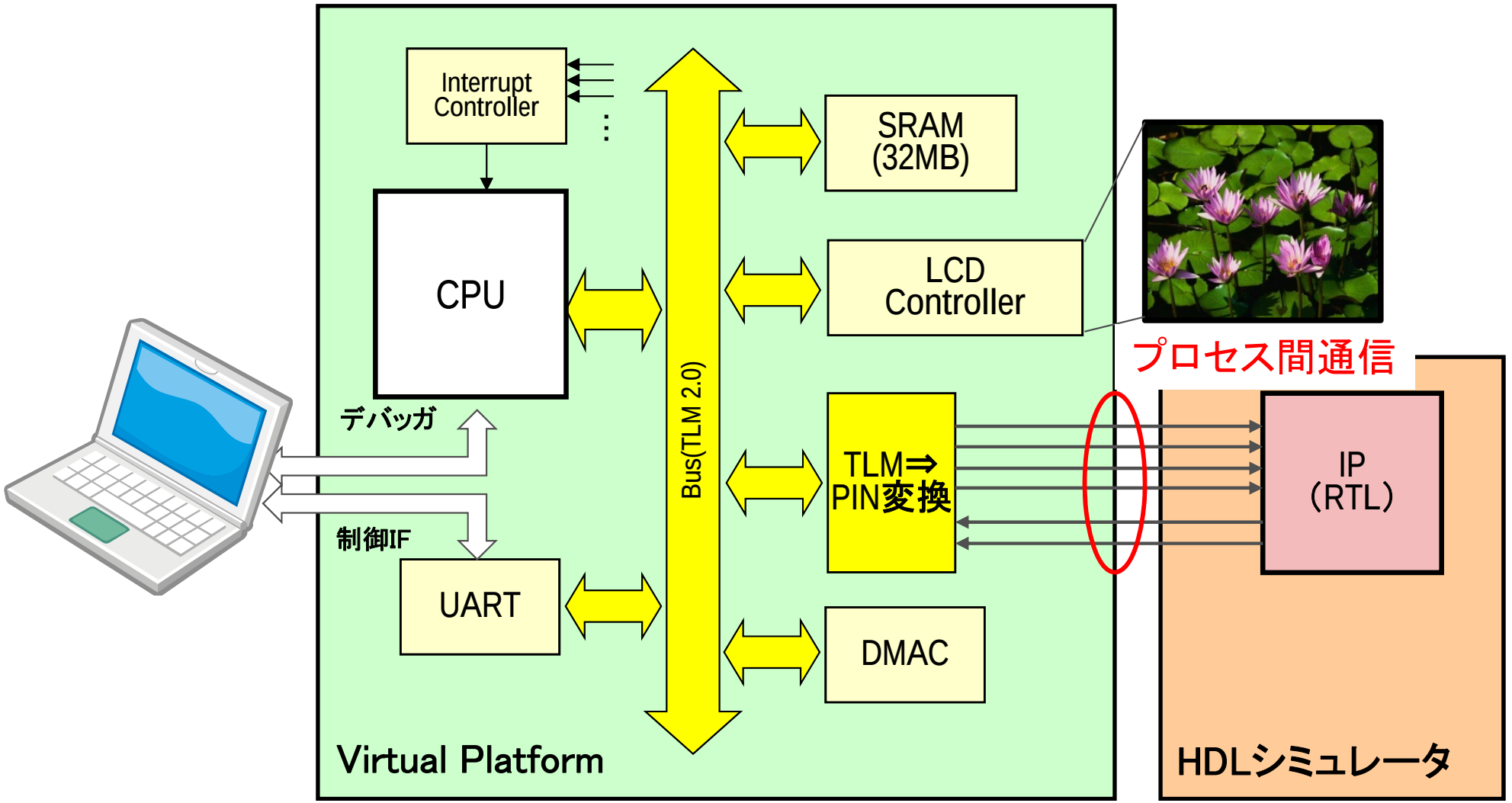


Copyright 2012 FUJITSU Microelectronics Solutions Limited

従来のVirtual Platform+RTL検証環境

FUJITSU

■Virtual PlatformとHDLシミュレータとのCo-Simulation

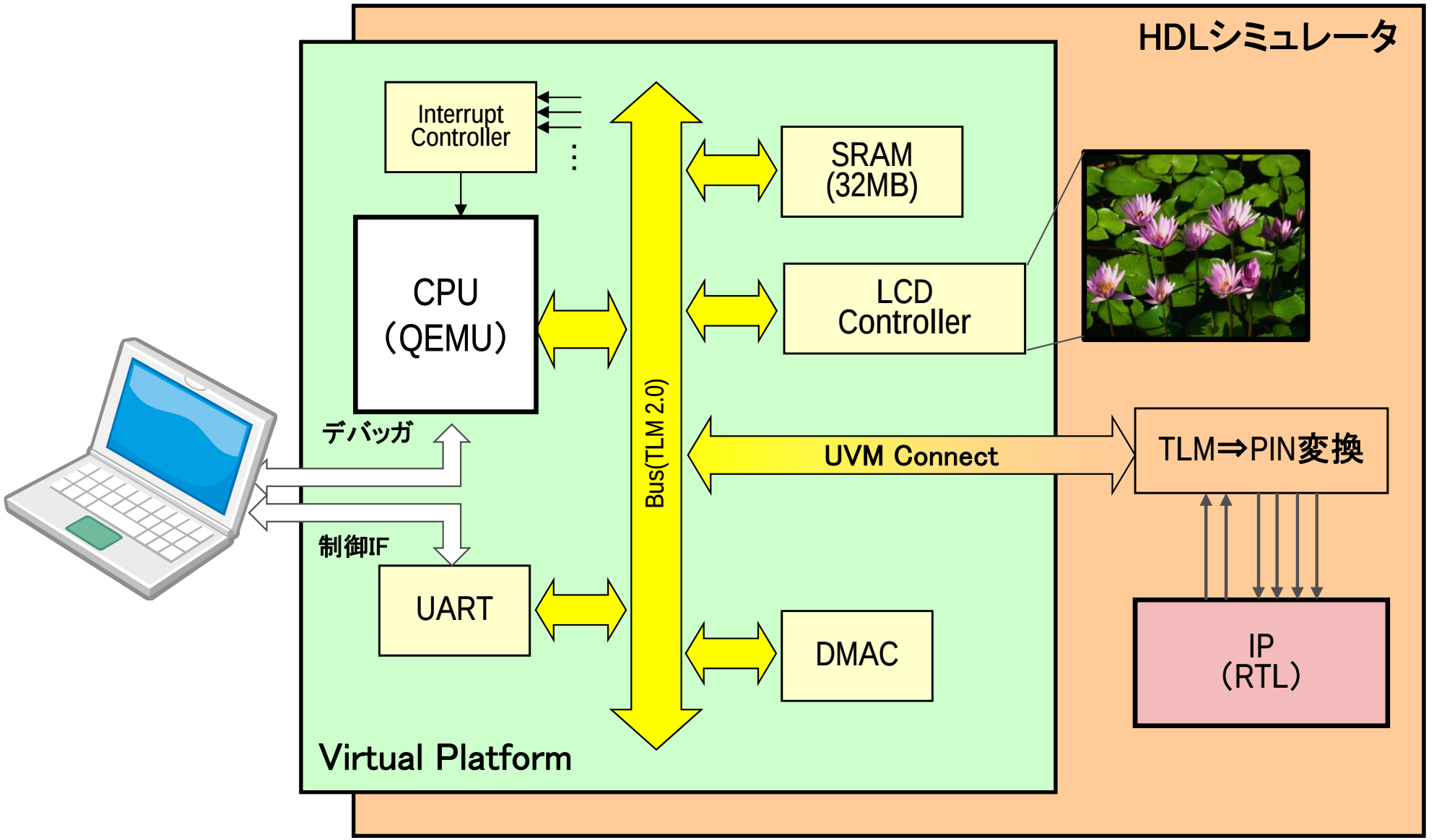


Copyright 2012 FUJITSU Microelectronics Solutions Limited

今回のVirtual Platform+RTL検証環境

FUJITSU

■HDLシミュレータのみで動作する検証環境



Copyright 2012 FUJITSU Microelectronics Solutions Limited

従来の検証環境比較

FUJITSU

■ 環境構築やSim時間を短縮出来た。

環境	従来	今回
ツール	VP+HDLシミュレータ	HDLシミュレータ
接続手法	プロセス間通信	UVM Connect
接続形式	ピン接続	トランザクション接続
環境構築	10日	5日
Sim時間	約90分	約8分
メリット	・デバッグ環境 ・抽象度の可変	・VP部分は無料（オープン） ・TLM I/Fで接続
デメリット	・VP部分が有料 ・ツール依存な環境	・デバッグ環境 ・抽象度：Loosely-Timed（LT）

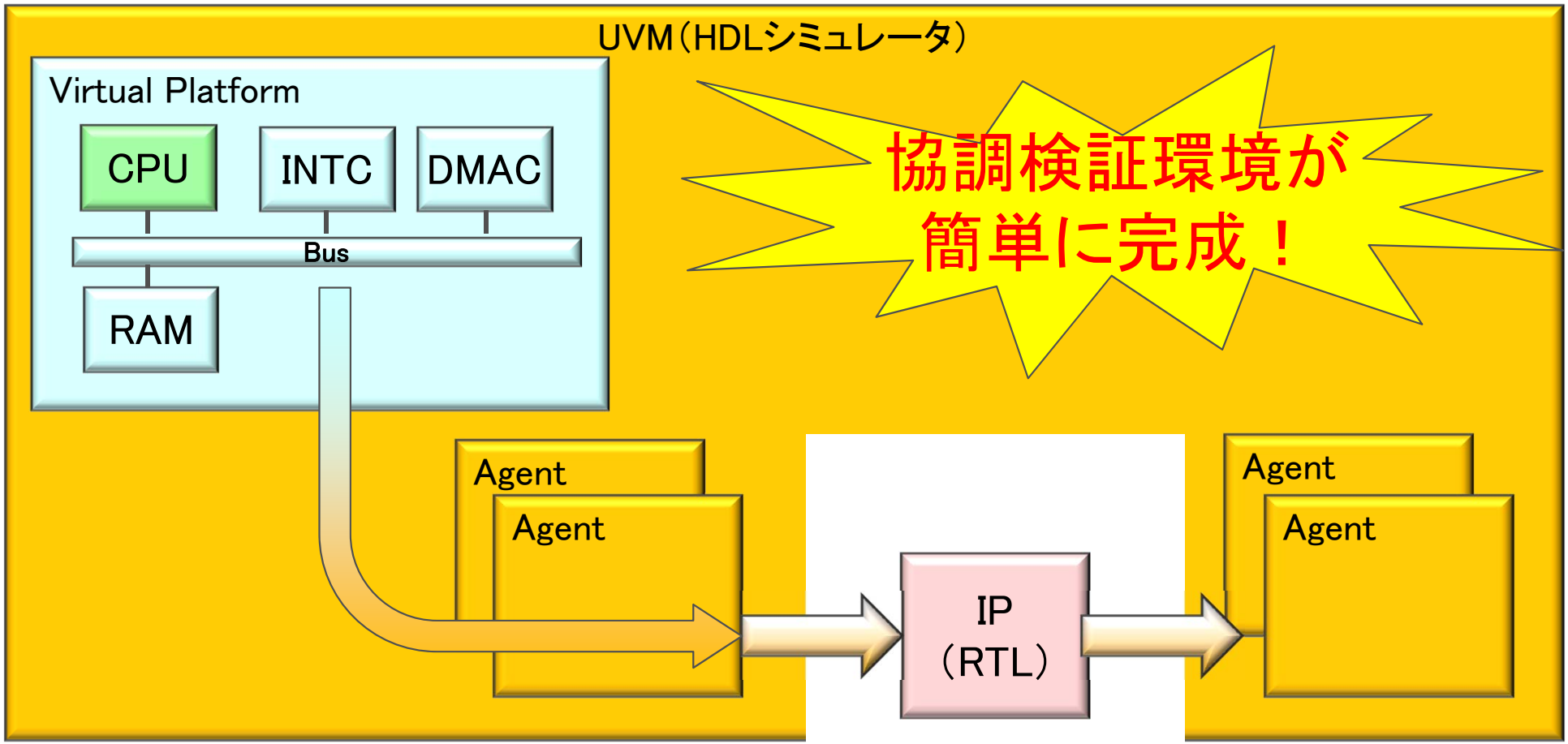
HW/SW協調検証環境と考えるとメリットは大きい

Copyright 2012 FUJITSU Microelectronics Solutions Limited

用途例：IPのドライバ開発（HW/SW協調）

FUJITSU

- RTL検証工程中盤から終盤、そしてシステム検証へ



用途に応じて、最適な検証環境を構築する。

Copyright 2012 FUJITSU Microelectronics Solutions Limited

FUJITSU

まとめ

- 紹介した検証環境
- 最後に

Copyright 2012 FUJITSU Microelectronics Solutions Limited

まとめ：紹介した検証環境

FUJITSU

アルゴリズム（C/C++）との検証

- DPI-C+RTL検証環境

再利用

TLMモデルとの検証

- TLM/RTL検証環境

共通化

ソフトウェアとの検証

- Virtual Platform+RTL検証環境

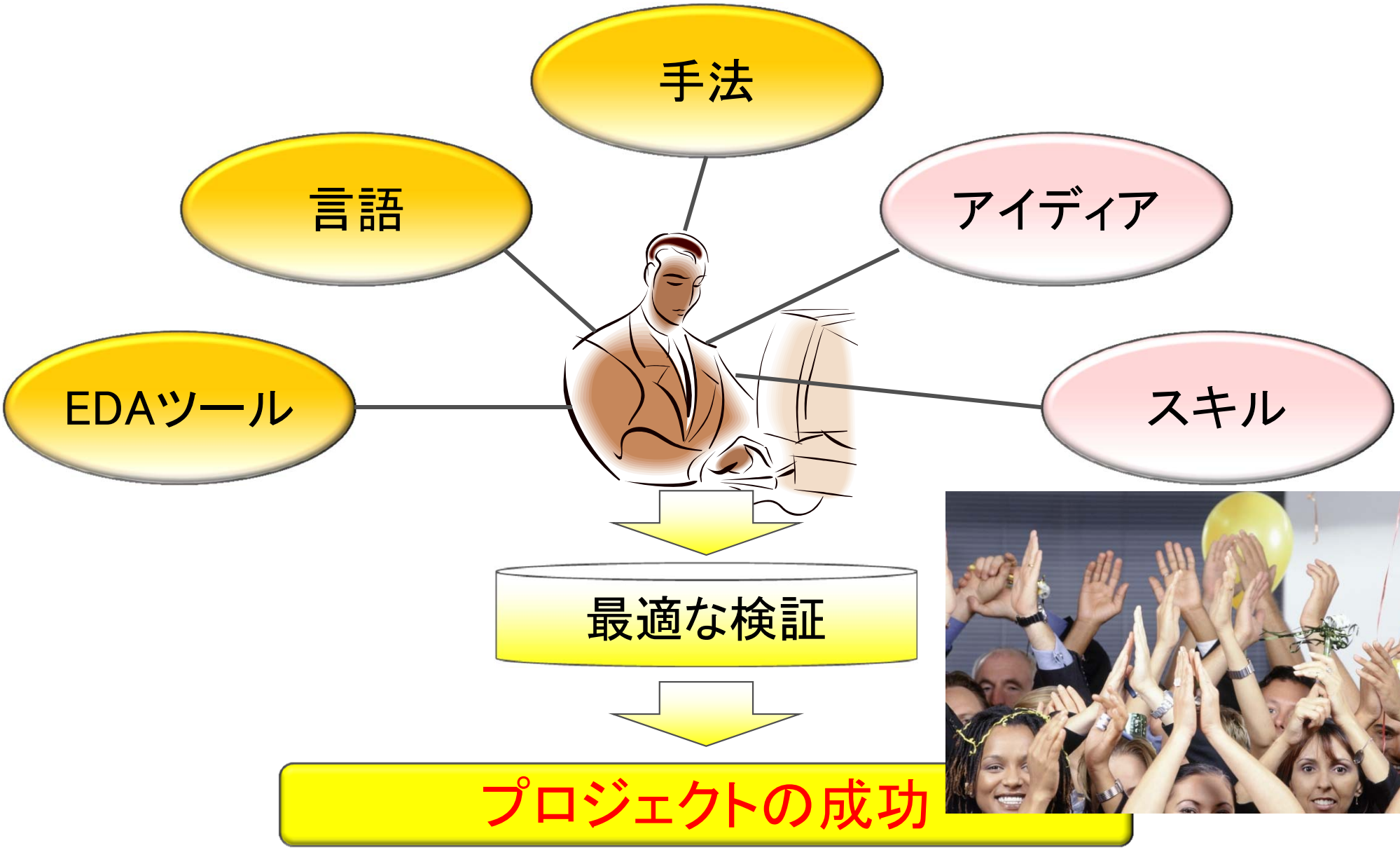
コスト

Copyright 2012 FUJITSU Microelectronics Solutions Limited

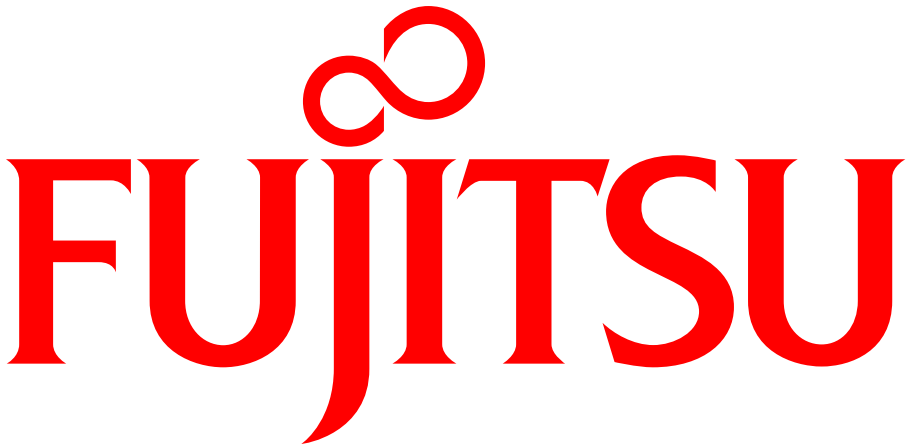
最後に

FUJITSU

■検証環境は十人十色



Copyright 2012 FUJITSU Microelectronics Solutions Limited



shaping tomorrow with you