JEITA

ＥＤＡアニュアルレポート　２００６

Annual Report on Electronic Design Automation

－　65 nm から 45 nm テクノロジ世代の EDA 技術の進展に向けて　－

２００７年５月発行

作　　成

ＥＤＡ技術専門委員会

EDA Technical Committee

発　　行

社団法人　電子情報技術産業協会

Japan Electronics and Information Technology Industries Association

目　次

【巻頭言】

「65 nm から 45 nm テクノロジ世代の EDA 技術の進展に向けて」

EDA 技術専門委員会　2006 年度 委員長　杉山 八六

現在及び近い将来のユビキタス社会を構築していく上で、基幹となる半導体産業への期待が高まっている。なかでも、半導体の設計に関する技術は、LSI の高密度化、高集積化、高性能化への要求を支える重要な役割を担っており、より一層の技術向上を図る必要がある。

設計上流では超大規模システム LSI の機能・論理の設計・検証問題、設計下流ではいわゆる DFM（Design For Manufacturing）問題、そしてこれをつなぐインプリメンテーションの難易度の飛躍的増大などへの対応である。

EDA 技術専門委員会は、電子情報技術産業協会（JEITA）における業界活動組織の 1 つとして、電子機器の設計自動化（EDA : Electronic Design Automation）に関わる様々な活動を行っている。特に、電子機器の機能・性能を決定するシステム LSI 設計技術に係わる活動を、その中心に置いている。

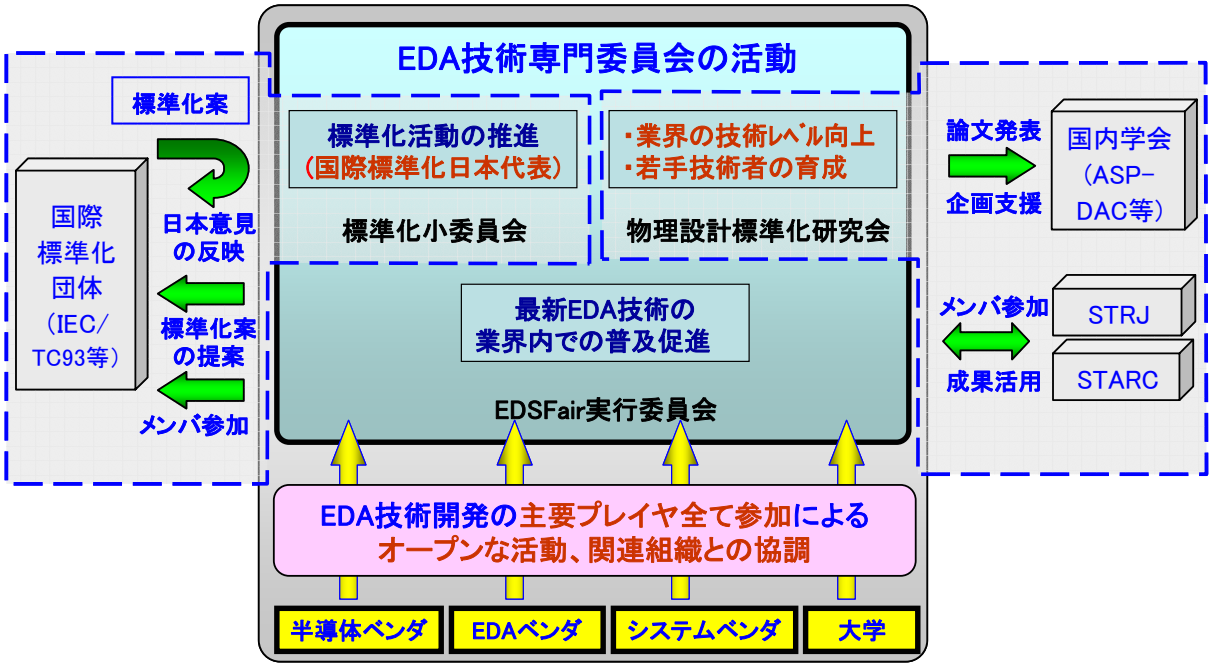
活動の第 1 のテーマは、システム LSI 設計技術に関する動向、関連情報についての調査、検討と課題解決への提案でである。2006 年度はプロセスばらつき、配線モデル抽出などの設計課題の検討と解決策の提示を物理設計標準化研究会にて取り組んだ。

活動の第 2 のテーマは、EDA 技術に関する標準化活動と関連機関、団体への協力と貢献である。IEEE、IEC 等の国際的な標準化活動に対し、標準化小委員会を中心に、標準化提案の検証、技術提案・交流会議などを行った。これまで、SystemC 及び SystemVerilog という、システムないし LSI の設計記述言語の標準化に具体的に取り組むタスクグループを発足させて活動をしている。2006 年度は、IEEE における SystemC 及び SystemVerilog の言語仕様標準化後のフォローアップ、並びに追加標準仕様の検討・提案を行い、国際標準制定に貢献した。

また、電子情報通信学会内に設置されている IEC/TC93 国内委員会/WG2（ハードウェア設計記述言語）委員として活動を推進しており、2006 年 9 月には IEC/TC93 の国際会議（ベルリン）に参加した。

活動の第 3 のテーマは、EDA 技術及び標準化の普及、推進のためのイベント開催、支援である。EDSFair2007（Electronic Design and Solution Fair 2007）は、この最大のイベントとして取り組み、昨年度に引き続きパシフィコ横浜で開催した。EDSFair2007 は、国内唯一の電子機器の設計技術の総合的な展示会であり、ASP-DAC（Asia and South Pacific Design Automation Conference）や FPGA コンファレンスと連携し、システム・デザイン・フォーラム 2007 の継続開催や当委員会企画の特設ステージなど、出展者誘致、来場者増加に向けて積極的に活動した結果、過去最多の出展を記録し、大勢の来場者に来ていただいた。

以下に EDA 技術専門委員会の活動と関連団体との関係を示す。



IEC/TC93：国際電気標準会議/設計自動化
ASP-DAC ：Asia South Pacific-Design Automation Conference
STRJ ：半導体技術ロードマップ委員会
STARC ：半導体理工学研究センター

図 1　EDA 技術専門委員会と関連組織との関係

EDA 技術専門委員会はこれら様々な活動を通じ、65nm から 45nm テクノロジ世代の「システム・オン・チップ時代」の電子機器業界に対し設計技術の面で貢献し、さらには「システム・オン・チップ時代」がもたらす世界規模の産業界の変革を乗り越えて、日本と世界の電子機器業界のさらなる発展に寄与すべく、19 社約 50 名の業界各社有志・メンバの自主的な参画により運営実行してきた。

本冊子「EDA アニュアルレポート 2006」は、EDA 技術専門委員会の 2006 年度の活動の年次報告として、上記 3 つの活動テーマそれぞれについて、2006 年度の成果をまとめたものである。Web にも各種報告を掲載しているので、ご覧いただきたい。

（http://eda.ics.es.osaka-u.ac.jp/jeita/eda/index-jp.html）

「システム・オン・チップ時代」に入り、電子機器が切り拓く素晴らしい未来が広がることを確信しつつ、2007年度も積極的な活動を展開していく。

2006 年度 JEITA/EDA 技術専門委員会 委員一覧

委員長	杉山八六	富士通㈱	電子デバイス事業本部 シニアスタッフ
副委員長	齋藤茂美	ソニー㈱	半導体事業グループ システム LSI 事業本部 設計基盤技術部門 5 部 ライブラリ担当部長
副委員長	山田　節	三洋電機㈱	研究開発本部 デジタルシステム研究所 担当課長（LSI 設計技術担当）
監事	秋山俊恭	㈱ルネサステクノロジ	製品技術本部　設計技術統括部 副統括部長
幹事	灘岡　満	沖電気工業㈱	シリコンソリューションカンパニー 共通技術本部　設計推進部 部長
同	西本猛史	シャープ㈱	IC 事業本部 先端技術開発センター　第 1 開発室 副参事
同	宮山芳幸	セイコーエプソン㈱	半導体事業部　IC 基盤技術部 部長
同	藤波義忠	NEC エレクトロニクス㈱	基盤技術開発事業本部 開発サポートセンター グループマネージャ
同	田口浩文	松下電器産業㈱	半導体社　システム LSI 事業本部 商品開発センター 設計第 4 開発グループ グループマネージャ
同	中尾　徹	ローム㈱	LSI 開発システム本部 LSI デザインクオリティ開発部 技術主任
同	南　文裕	㈱東芝	デザインソリューション統括部 設計自動化技術開発部 設計メソドロジ担当 グループマネージャ

委　　員	藤 岡 督 也	㈱ジーダット	EDA 営業技術部 部長
同	山 田 明 宏	凸版印刷㈱	設計本部 ソリューション部長
同	安 井 孝 史	日本ケイデンス・ デザイン・システムズ社	CSSC AE ディレクター
同	飯 島 一 彦	日本シノプシス㈱	技術本部　本部長
	瀬 谷 和 宏	丸紅ソリューション㈱	半導体システム事業部
同			副事業部長
同	小 島　　智	メンター・グラフィックス・ ジャパン㈱	フィールドマーケティング部 ディレクター
同	藤 井 浩 一	㈱リコー	電子デバイスカンパニー 画像 LSI 開発センター　CAD 技術室 課長代理
同	横 川　　隆	㈱図研	SoC 事業部　デザインセンター 部長
特別委員	奥 村 隆 昌	富士通 VLSI㈱	テクノロジ開発統括部　第一開発部 プロジェクト課長
同	浜口 加寿美	松下電器産業㈱	半導体社　システム LSI 事業本部 商品開発センター 設計第 4 開発グループ チームリーダー
同	長谷川　　隆	富士通㈱	LSI 事業本部 第 2 システム LSI 事業部 設計技術部 プロジェクト課長
客　　員	神 戸 尚 志	近畿大学	理工学部　電気電子工学科 教授
同	今 井 正 治	大阪大学	大学院　情報科学研究科 情報システム工学専攻 教授
同	岡 村 芳 雄	STARC	開発第 2 部　部長
同	若 林 一 敏	日本電気㈱	システムデバイス研究所 システム CAD 研究部長

略　語　一　覧

[1]　団体・組織の名称

Accellera
　VI と OVI を統合した、設計記述言語の標準化に関連する活動機関

ANSI　American National Standards Institute
　米国の標準化国家機関

ASP - DAC　Asia and South Pacific Design Automation Conference
　アジア・太平洋地域での EDA 関連の国際学会（1995 年に始まる）

CENELEC　European Committee for Electrotechnical Standardization
　EC（欧州委員会）の電気電子分野に関する標準化機関

DAC　Design Automation Conference
　米国で行われる EDA 関連の国際学会

DASC　Design Automation Standardization Committee
　IEEE の下部組織で設計自動化に関する標準化委員会

ECSI　European CAD Standardization Initiative
　欧州の設計自動化に関する標準化機関

EDIF Div.　Electronic Design Interchange Format Division
　EIA の下部組織で電子系の惜報データ交換規格の検討機関

EIA　Electronic Industries Alliance
　米国の電子業界団体（Association を Alliance に改称）

JEITA　Japan Electronics and Information Technology Industries Association
　社団法人 電子情報技術産業協会（電子業界団体）

ICCAD　International Conference on Computer Aided Design
　CAD に関する国際学会

IEC　International Electrotechnical Commission
　電気電子分野に関する国際標準化機関

IEEE　Institute of Electrical and Electronics Engineers,Inc.
　米国の電気電子分野の国際的な学会組織

IPC　Institute for Interconnecting and
　Packaging Electronic Circuits Industry Association
　米国のプリント回路に関する業界組織

ISO　International Organization for Standardization
　国際標準化機関

IVC　International Verilog Conference
　OVI が主催する Verilog HDL 国際学会

JPCA	Japan Printed Circuit Association 社団法人 日本電子回路工業会
OSCI	Open SystemC Initiative SystemC の標準化団体
OVI	Open Verilog International Verilog - HDL に関連する技術の標準化と普及推進組織
SEMATECH	Semiconductor Manufacturing Technology Initiative (Consortium) 半導体技術を向上するために始まった米国の官民プロジェクト
Si2	Silicon Integration Initiative 設計環境の整備促進を支援する米国の非営利法人(旧 CFI)
VASG	VHDL Analysis and Standards Group DASC 傘下の VHDL 標準化に関するワーキンググループ
VITAL	VHDL Initiative Toward ASIC Libraries VHDL ライブラリ標準化団体
VSIA	Virtual Socket Interface Alliance LSI の機能ブロックの I/F 標準化を目指している業界団体

[2] 標準化・規格に関する技術用語

ALF	Advanced Library Format OVI で検討された IP をも含む ASIC ライブラリのフォーマット
ALR	ASIC Library Representation ASIC ライブラリ表現
CALS	Computer Aided Logistics Support (1985) Commerce At Light Speed (1995)
CHDS	Chip Hierarchical Design System SEMATECH が要求仕様を作成した 0.25-0.18um 世代設計システム
CHDStd	Chip Hierarchical Design System technical data CHDS で使用するデータモデルの標準化
DCL	Delay Calculation Language 遅延計算のための記述言語
DPCS	Delay and Power Calculation System IEEE 1481 として標準化推進されている遅延と消費電力の計算機構仕様
EDI	Electronic Data Interchange 電子データ交換
EDIF	Electronic Design Interchange Format EIA の下部組織で検討されている電子系の情報データ交換規格

ESPUT	European Strategic Program for Research and Development in Information Technology 欧州情報技術研究開発戦略計画
HDL	Hardware Description Language ハードウェア記述言語
IP	Intellectual Property 流通/再利用可能な LSI 設計資産(本来は知的財産権の意)
JIS	Japanese Industrial Standard 日本工業規格
SDF	Standard Delay Format 遅延時間を表記するフォーマット
SLDL	System Level Design Language システム仕様記述言語
STEP	Standard for the Exchange of Product Model Data CAD の製品データ交換のための国際規格
VHDL	VHSIC (Very High Speed Integrated Circuit) Hardware Description Language IEEE 1076 仕様に基づくハードウェア記述言語
VHDL-AMS	VHDL-Analog and Mixed-Signal (Extensions) DASC の中で進められている VHDL のアナログ及びミックストシグナルシステムへの拡張

1.　EDA 技術専門委員会の活動

2006.05.19　第 3 版

1.1　2006 年度 JEITA/EDA 技術専門委員会 事業計画

委員会の名称　EDA 技術専門委員会（Electronic Design Automation Technical Committee）

委員会の目的　EDA に関連する技術及びその標準化の動向を調査し、その発展、推進を図り、もって国内外の関係業界の発展に寄与する

委員会の構成
- 会員会社/委員　：19 社/19 名
- 特別委員　：3 名
- 客　員　：4 名

委員会の役員
- 委員長　：富士通　杉 山 八 六
- 副委員長（正）：ソニー　齋 藤 茂 美
- 副委員長（代行）：三洋電機　山 田　節
- 監　事　：ルネサス　秋 山 俊 恭

＜方針＞
委員会活動活性化を重視し、
1. 主体は委員会、幹事会は臨時のみ
2. EDSFair, ASP-DAC との連携強化
3. 2006 年度の予算運用及び検討項目
（繰越金の有効活用。メンバ会社増を検討）

下部組織の役員

標準化小委員会	主　査：ソニー	齋 藤 茂 美
SystemC タスクグループ	主　査：富士通	長谷川　隆
SystemVerilog タスクグループ	主　査：松下	浜口 加寿美
物理設計標準化研究会	主　査：富士通	奥 村 隆 昌
EDSFair 実行委員会	委員長：沖	灘 岡　満

EDA 技術専門委員会メンバと担当　（2006 年 5 月現在）（敬称略）

委 員 長：富士通	杉 山 八 六	EDSFair/ASP-DAC 小委員会主査、ASP-DAC 支援
副委員長：ソニー	齋 藤 茂 美	標準化小委員会主査
副委員長：三洋電機	山 田　節	システム・デザイン・フォーラム実行委員長、ASP-DAC2007 Designer's Forum 委員、内規改訂
監　事：ルネサス	秋 山 俊 恭	EDSFair 実行委員、EDSFair 企画 WG 主査
幹　事：沖	灘 岡　満	EDSFair 実行委員長、システム・デザイン・フォーラム実行委員
同　：シャープ	西 本 猛 史	EDSFair 企画 WG メンバ
同　：セイコーエプソン	宮 山 芳 幸	ホームページ、メールシステム、費用消化集計
同　：NECエレクトロニクス	藤 波 義 忠	EDSFair 実行委員、広報パンフレット
同　：松下電器産業	田 口 浩 文	アニュアルレポート、EDSFair 企画 WG メンバ、システム・デザイン・フォーラム実行委員
同　：ローム	中 尾　徹	
同　：東芝	南　文 裕	EDSFair 実行委員、システム・デザイン・フォーラム実行委員、ASP-DAC2007 Designer's Forum 委員

委　　員	ジーダット	藤岡　督也	
同	凸版印刷	山 田 明 宏	
同	日本ケイデンス	安 井 孝 史	
同	日本シノプシス	飯 島 一 彦	
同	丸紅ソリューション	瀬 谷 和 宏	
同	メンタージャパン	小 島　　智	TC93/WG2 ココンビナー
同	リコー	藤 井 浩 一	
同	図研	横 川　　隆	
特別委員	富士通	奥 村 隆 昌	物理設計標準化研究会主査
同	松下電器産業	浜口 加寿美	SystemVerilog タスクグループ主査
同	富士通	長谷川　　隆	SystemC タスクグループ主査
客　　員	近畿大学	神 戸 尚 志	元委員長
同	大阪大学	今 井 正 治	上流設計識者、ASP-DAC リエゾン
同	STARC	岡 村 芳 雄	元委員長
同	日本電気	若 林 一 敏	ASP-DAC リエゾン
事 務 局	JEITA/電子デバイス部	恩 田　　豊	
同	同	木 暮 英 男	

活動計画の概要　＜別紙－1 参照＞

委員会の予算　会費 20,000 円 * 12 ケ月 * 19 社＝4,560,000 円

委員会の開催　年 6 回程度(予定日：**別紙－2** 参照)

必要に応じて幹事会を開催する

担当事務局　JEITA/電子デバイス部

＜別紙－1＞
活動計画の概要

1. EDA 技術に関する動向、関連情報についての調査、検討と課題解決への提案

(1) 研究会による技術動向、ニーズ調査
DSM に関する技術課題の明確化と EDA 技術による解決策の検討
→ 物理設計標準化研究会

(2) 関連機関、団体、キーパーソン等との合同会議、意見交換、交流
STARC、STRJ 等
特に、STRJ/WG1 との具体的な連携・協同の検討

(3) 国内外の学会、研究会、イベントへの参加
ASP-DAC2007、DAC2006、IEICE、IPSJ 等

2. EDA に関する標準化活動への貢献と関連機関、団体への対応

(1) EDA 設計言語及びモデル標準化のための技術的検討と提案
・SystemC、SystemVerilog をそれぞれ 03 年度に新設した標準化小委員会下のタスクグループにて継続して検討、提案活動を継続
・Verilog-HDL、VHDL、アナログ HDL、DPC 等必要に応じて
→ 標準化小委員会で必要に応じて検討

(2) 国際的な関連機関、団体への参画と標準化活動への協力
・IEC/TC93(及び WG2)国際会議(2006/9 ベルリン)への参加
・IEEE/DASC、IEEE-SA、Accellera、Si2 等への参加、協調
→ 標準化小委員会にて検討し、対応

3. EDA 技術及び標準化の普及、推進のためのイベント実施、支援

(1) 「EDSFair2007」(横浜)
日本エレクトロニクスショー協会へ運営委託
→ EDSFair 実行委員会、企画 WG(イベント開催)

(2) 各種ワークショップ、講演会の開催
「システム・デザイン・フォーラム 2007」を EDSFair2007 と同時開催

(3) 「ASP-DAC2007」(横浜)
→ Designer's Forum への協力

4. 委員会活動の広報

(1) 活動成果等の各種技術報告書の発行
プロジェクト、研究会による技術報告書、調査報告書等

(2) 広報パンフレット、アニュアルレポートの発行

(3) WWW ホームページの公開

5. その他

(1) EDSFair/ASP-DAC 小委員会は 2006 年度開催予定なし。関係委員会より要請があれば開催。

＜別紙－2＞

2006 年度 JEITA/EDA 技術専門委員会 会合

年/月	技術専門委員会	関連イベント
2006/4	4/21(金)　東芝　(東京地区) ・06 年度事業計画説明、承認 ・06 年度プロジェクト/研究会計画説明、承認 ・05 年度会計収支と 06 年度会計予算説明、承認 ・アニュアルレポート作成状況報告	・軽井沢 WS (4/24-25)　@軽井沢
2006/5		
2006/6	6/16(金)　シャープ　(東京地区) ・プロジェクト/研究会進捗報告 ・半導体部会/半導体技術委員会報告内容説明 ・委員名簿更新内容確認 ・予算消費状況	
2006/7		・DA シンポジウム 2006 (7/12-13) @浜松 ・DAC2006 (7/24-28) @SanFrancisco, California
2006/8		
2006/9	9/15(金)　ローム　(関西地区) ・プロジェクト/研究会進捗報告 ・半導体部会/半導体技術委員会報告内容説明 ・予算消費状況	
2006/10		
2006/11	11/17(金)　NECエレクトロニクス　(東京地区) ・プロジェクト/研究会進捗報告 ・半導体部会/半導体技術委員会報告内容説明 ・予算消費状況 ・EDSFair 用パンフレット作成手順説明	・ICCAD2006 (11/5-9) @San Jose, California
2006/12		
2007/1	1/19(金)　松下　(関西地区) ・プロジェクト/研究会進捗報告 ・半導体部会/半導体技術委員会報告内容説明 ・07 年度体制協議 ・EDSFair 用パンフレット内容確認 ・アニュアルレポート作成分担・手順説明 ・予算消費状況	・ASP-DAC2007 (1/23-26) @横浜 ・EDSFair2007 (1/25-26) @横浜
2007/2		
2007/3	3/16(金)　セイコーエプソン　(東京地区) ・半導体部会/半導体技術委員会報告内容説明 ・06 年度プロジェクト/研究会の年間活動報告 ・06 年度予算消費状況 ・07 年度事業計画説明 ・07 年度プロジェクト/研究会の年間活動計画説明	・DATE2007 (4/16-20) @ Nice, France

(各開催日の社名は、議事録担当を示す)

1.2 2006 年度 JEITA/EDA 技術専門委員会 ホームページ

1.2.1 目的

電子情報技術産業協会(JEITA)の EDA 技術専門委員会の活動状況を公開し、EDA 技術の標準化や技術調査に関する理解とご協力をいただくことを目的とする。

1.2.2 ホームページの詳細

本年度は、このホームページを一新し、よりわかりやすく、また、欲しい情報に簡易にアクセスできるような構成に変更を行った。ホームページは英語版からスタートし海外からの利用者の利便性を考慮している。簡易に日本語版への切り替えも可能である。

(1) URL : http://eda.ics.es.osaka-u.ac.jp/jeita/eda/

大阪大学のご協力を頂き、大阪大学のサーバーにホームページを設置させていただいている。また、データの更新など、メンテナンスについてもご協力をいただいている。

(2) エントリーページの構成

日本語版、英語版はそれぞれ次のエントリーで構成されている。

日本語版：	英語版：
委員会のご紹介	Introduction of committee
委員会活動	Committee activity
公開資料ライブラリ	Open data library
イベント・関連機関	Event/related organization
お問合せ	Inquiry
サイトマップ	Site map

(3) 委員会の紹介／Introduction of committee

委員長挨拶、活動と成果、メンバー一覧をサブエントリとする。本委員会の概要、前年度の活動内容・結果、本年度の活動計画を紹介している。

(4) 委員会活動／Committee activity

下記の研究会・小委員会等の活動状況が紹介されている(英語版は一部)。

・標準化小委員会
・物理設計標準化研究会
・EDSFair 実行委員会

(5) 公開資料ライブラリ／Open data library

この「公開資料ライブラリ」ページでは、EDA 技術専門委員会内の各委員会の活動報告や各委員からの発表資料等を適宜掲載している。前年度のアニュアルレポートや、研究会の発表資料などを公開している。

また、旧ホームページのサイトにもリンクがあり、過去に終了した下記のプロジェクトの記録もたどることができる。

・SLD 研究会
・TAB(Technical Advisory Board)
・DPC プロジェクト

・HDL 技術小委員会
・HDL-IOP プロジェクト
・EDA ビジョン研究会
・1364HDL プロジェクト
・VHDL プロジェクト
・アナログ HDL プロジェクト

(6) イベント・関連機関／Events/related organization

関連の会議としては、次の関係の深い EDA 関連技術委員会、関連機関、イベントの紹介及びリンクが行われている。

・IEEE/DASC(電気電子学会／設計自動化標準化委員会)
・IEC/TC93(国際電気標準会議／デザインオートメーション標準化技術委員会)

また、関連機関として、本委員会に関連のある 17 機関の紹介があり、また、それぞれのホームページへのリンクが行われている。

(7) EDA 技術専門委員会のコンタクトアドレス

EDA 技術専門委員会、標準化小委員会などのコンタクトアドレスが記載されている。

1.3 2006 年度 JEITA/EDA 技術専門委員会 会合実績

月	EDA 技術専門委員会	
	幹事会	委員会
2006 年 4 月		4/21(金) 14-17　ハーモニーホール 4 階 大会議室
5 月		
6 月		6/16(金) 14-17　ハーモニーホール 6 階 中会議室
7 月		
8 月		
9 月		9/15(金) 14-17　JEITA 関西支部 第 1 会議室
10 月		
11 月		11/17(金) 14-17　ハーモニーホール 6 階 中会議室
12 月		
2007 年 1 月		1/19(金) 14-17　JEITA 関西支部 第 2 会議室
2 月		
3 月		3/16(金) 14-17　JEITA 306

月	小委員会・研究会・関連行事他
2006 年 4 月	4/20(木) 13:00-17:00　第 1 回 PDS 研究会　JEITA 305 会議室 4/21(金) 14:00-17:00　第 1 回 SystemC-TG　ハーモニーホール 4 階 大会議室(奥)
5 月	5/10(水) 11:00-17:00　第 1 回 SystemVerilog-TG　ケイデンス(新横浜) 会議室 5/11(木) 11:00-13:00　第 1 回標準化小委員会　機械振興会館 5 階 516 号会議室 5/26(金) 13:00-17:00　第 2 回 PDS 研究会　電子会館 8F 会議室
6 月	6/2 (金) 13:00-17:00　第 2 回 SystemC-TG　ハーモニーホール 3 階 小会議室 6/23(金) 13:00-17:00　第 3 回 PDS 研究会　JEITA 314 会議室
7 月	7/14(金) 13:00-17:00　第 4 回 PDS 研究会　電子会館 8F 会議室 7/14(金) 13:00-17:00　第 3 回 SystemC-TG　ハーモニーホール 4 階 大会議室
8 月	8/25(金) 11:00-17:00　第 5 回 PDS 研究会　東京工業大学 すずかけ台キャンパス 8/25(金) 11:00-17:00　第 2 回 SystemVerilog-TG　ケイデンス(新横浜) 会議室
9 月	9/5 (火) 10:30-13:00　第 2 回標準化小委員会 機械振興会館 5 階 516 会議室 9/21(木) 13:00-17:00　第 4 回 SystemC-TG　JEITA(東京) 314 会議室 9/22(金) 11:00-17:00　第 6 回 PDS 研究会　ハーモニーホール 3 階 小会議室
10 月	10/20(金) 13:00-17:30　第 5 回 SystemC-TG　JEITA 関西支部 第 1 会議室 10/27(金) 11:00-17:00　第 7 回 PDS 研究会　JEITA 関西支部 第 1 会議室
11 月	11/ 1(金) 11:00-17:00　第 3 回 SystemVerilog-TG　ケイデンス(新横浜) 会議室 11/16(金) 11:00-13:00　第 3 回標準化小委員会　機械振興会館 5 階 516 会議室 11/17(金)～11/18(土)　第 6 回 SystemC-TG(集中審議)　熱海 志ほみや旅館 会議室 11/29(水)～11/30(木)　第 4 回 SystemVerilog-TG(集中審議)　小倉ワシントンホテル 会議室
12 月	12/1 (金)～12/02(土)　第 8 回 PDS 研究会/集中審議　北九州学研都市 産学連携センター 12/13(水) 13:00-17:00　第 5 回 SystemVerilog-TG　品川メンター 会議室 12/15(金) 13:00-17:00　第 7 回 SystemC-TG　ベルサール神田 会議室 6 12/22(金) 11:00-17:00　第 9 回 PDS 研究会　JEITA 関西支部 第 1 会議室
2007 年 1 月	1/12(金) 11:00-17:00　第 10 回 PDS 研究会　JEITA 314 会議室 1/19(金) 13:00-17:00　第 8 回 SystemC-TG　東京グリーンホテル水道橋 2 階 琥珀の間
2 月	2/16(金) 13:00-17:00　第 9 回 SystemC-TG　JEITA(東京) 303 会議室 2/23(金) 11:00-17:00　第 11 回 PDS 研究会　JEITA 関西支部 第 2 会議室
3 月	3/2 (金) 11:00-13:00　第 4 回標準化小委員会　機械振興会館 5 階 516 会議室 3/9 (金) 13:00-17:00　第 6 回 SystemVerilog-TG　松下電器産業 会議室(長岡京) 3/23(金) 13:00-17:00　第 10 回 SystemC-TG　JEITA(東京) 314 会議室 3/23(金) 11:00-17:00　第 12 回 PDS 研究会/成果報告会　JEITA 305 会議室

2.　各技術委員会の活動報告

2.1 物理設計標準化研究会(Physical Design Standardization Study Group)

2.1.1 目的

半導体デバイス・配線テクノロジの進化に伴い、新たな設計上の課題があらわれてきている。また、これらの課題を解決するため各社が開発した手法やライブラリが、そのテクノロジが一般化した後も標準化されず、設計環境の開発・サポートコスト低減の障害となる事例や、半導体ベンダと顧客との情報授受がスムーズに行えない事例が増えてきている。

上記課題を背景として、本研究会では、次のような調査及び標準化を実施することにより、より効率的な設計環境の実現に貢献することを目的として活動を行っている。

- 次世代(65 ナノメータ)以降のテクノロジ・ノードにおける、LSI の物理設計・検証に関する課題の抽出
- 半導体ベンダとその顧客との間でやり取りするライブラリや設計情報等を規定する、設計ルール・ガイドラインの作成
- LSI の物理設計、検証手法の精度、互換性や効率を向上できるライブラリの標準化
- 各種ライブラリを用いて行う検証が十分な精度で行えるかを判定するための標準ベンチマークデータの作成

2.1.2 活動内容

2005 年 5 月から活動を開始し、2007 年 3 月までの約 2 年間の活動を実施した。今年度は、昨年度に継続して下記のテーマを取り上げて調査やベンチマーク活動を行った。

- 環境温度依存を考慮した配線抵抗ばらつきの回路特性へのインパクトの調査
- 統計的静的タイミング解析での技術的な課題に対する検討と提案

これら 2 つのテーマを集中的に検討するために 2 つのタスクグループを設置し、2006 年度はそれぞれ次のような活動を行った。

- **寄生効果モデリング・タスクグループ(Parasitic extraction modeling Task Group: PEM-TG)**

  微細化に伴い回路動作への影響が深刻化している製造及び環境ばらつきに対し、ばらつきの環境温度依存性に着目し、それらが回路特性へ与える影響について次の調査・検討を行った。1) 65nm プロセスでの素子特性ばらつきと配線抵抗ばらつきの温度依存性を考慮した遅延解析。2) 配線抵抗ばらつきが回路特性ばらつきに与える影響。

  また、昨年度から継続して行っている、標準的なベンチマーク回路を用いた LPE(Layout Parasitic/Parameter Extraction)ツールの精度評価結果についてまとめた。

- **統計的デザインメソドロジ・タスクグループ(Statistical design methodology Task Group: SDM-TG)**

  微細化の進展に伴って顕著となってきたばらつきの影響に対処するため適用が検討され始めている統計的な設計手法について、特に統計的静的タイミング解析技術に関する次の調査・検討を行った。1) 統計的静的タイミング解析技術における誤差要因の検討とその評価指標の提案('05 年度活動からの継続)。2) 統計的静的タイミング解析におけるスルーばらつきの検討とスルー依存性を考慮した遅延ばらつき計算手法の提案。

これらの活動で得られた成果は、次のような形態により無償で一般に公開する。

・アニュアルレポート

・JEITA のホームページ

・関連学会の研究会・学会における発表や論文誌への投稿

成果の詳細は本アニュアルレポートの付録に掲載した。また、昨年度、及び、今年度の成果の一部を以下の学会、論文誌にて発表した。

［1］「統計的 STA の精度検証手法」，回路とシステム軽井沢ワークショップ，2006 年 4 月.

［2］「統計的 STA の精度検証手法とその適用例」，DA シンポジウム 2006, 2006 年 7 月.

［3］「Proposal of metrics for SSTA accuracy evaluation」，IEICE Trans. on Fundamentals, 2007 年 4 月.

［4］「45-65nm ノードにおける遅延ばらつき特性の環境温度依存性」，回路とシステム軽井沢ワークショップ，2007 年 4 月.

［5］「統計的 STA でのスルー依存性を考慮した遅延ばらつき計算手法の提案」，回路とシステム軽井沢ワークショップ，2007 年 4 月.

2.1.3 関連機関の動向

米国の非営利標準化法人 SI2（http://www.si2.org/）は、IEEE 1481 の DPCS（Delay and Power Calculation System）及び DCL を拡張した、OLA（Open Library API）や寄生素子抽出用プロセスパラメータの標準セット（SIPPs: Standard Interconnect Performance Parameters）等、標準化の推進と普及に取り組んでいる。また、OVI（Open Verilog International）と VI（VHDL International）が合併して発足した Accellera（http://www.accllera.org/）では、従来の Verilog や VHDL に加え、上記 DPCS や EDA ライブラリの標準 ALF の IEEE 標準化作業もサポートしている。

一方日本では、あすかプロジェクトが、半導体最先端技術の「壁」を産官学で共同して克服する共通基盤としての役割を担ってきた（2006 年 3 月まで）。具体的には、90nm テクノロジ・ノードにおける、システム・オン・チップ（SoC）設計技術、デバイス・プロセス技術の共同開発が行われた。その実行組織の 1 つである半導体理工学研究センターSTARC（http://www.starc.or.jp/）では、SoC 設計技術開発等、先導開発が行われている。

2.1.4 参加メンバ

主　　査	奥 村 隆 昌	富士通 VLSI㈱
副 主 査	門 脇 匡 志	松下電器産業㈱
委　　員	中 島 英 斉	NEC エレクトロニクス㈱
同	黒 川　　敦	三洋半導体㈱
同	中林 太美世	シャープ㈱
同	小 野 信 任	㈱ジーダット・イノベーション
同	多久島 純之	ソニーLSI デザイン㈱
同	室 本　　栄	日本ケイデンス・デザイン・システムズ社
同	小 林 宏 行	日本シノプシス㈱
同	高 藤 浩 資	㈱リコー
同	増 田 弘 生	㈱ルネサステクノロジ
客　　員	佐 藤 高 史	東京工業大学
客　　員	橋 本 昌 宜	大阪大学

2.2 標準化小委員会

2.2.1 標準化小委員会

(1) 発足の背景とミッション

JEITA/EDA 技術専門委員会の標準化活動は 1990 年の EIAJ/EDIF 研究委員会設立に始まり、当初は EDA に関するグローバルな重要課題に対して日本の業界を代表する唯一の機関として、特に設計記述言語の仕様標準化とその啓蒙等に多大な貢献を果たしてきた。近年は活動の中心が設計記述言語の普及定着と環境変化に応じて、先端的設計技術に関する調査・研究等にシフトしてきている。

システム設計メソドロジの革新が進展する中で、標準化は依然として重要なテーマである。標準化関連の活動をより明確に位置付けるため、2000 年 11 月に本小委員会が設立された。

世界的にみれば EDA 関連の標準は IEC(International Electrotechnical Commission)と IEEE(The Institute of Electrical and Electronics Engineers)で議論、制定されてきた。IEC ではデザインオートメーションを議論する TC(Technical Committee)93、IEEE はコンピュータソサイエティの DASC(Design Automation Standards Committee)と SA(Standards Association)である。これまでは IEEE で定められた標準を IEC でも追認するものも多かった。2003 年より議論は IEEE の DASC/SA のワーキンググループでも、標準の制定は IEC と IEEE で同時にできるようになった(Dual Logo)。

国内では IEC の対応機関は電子情報通信学会である。TC 毎に国内委員会があり、電子情報通信学会や JEITA に組織化されている。TC93 とハードウェア設計記述言語関連のワーキンググループ(WG2)の国内委員会は電子情報通信学会にある。

本小委員会は IEC/TC93/WG2 国内委員会を兼ねて活動するという協調体制を 2002 年度に確立した(図 1 参照)。その結果、標準化小委員会の委員が IEC/TC93/WG2 の各種標準化提案を直接審議することができるようになった。

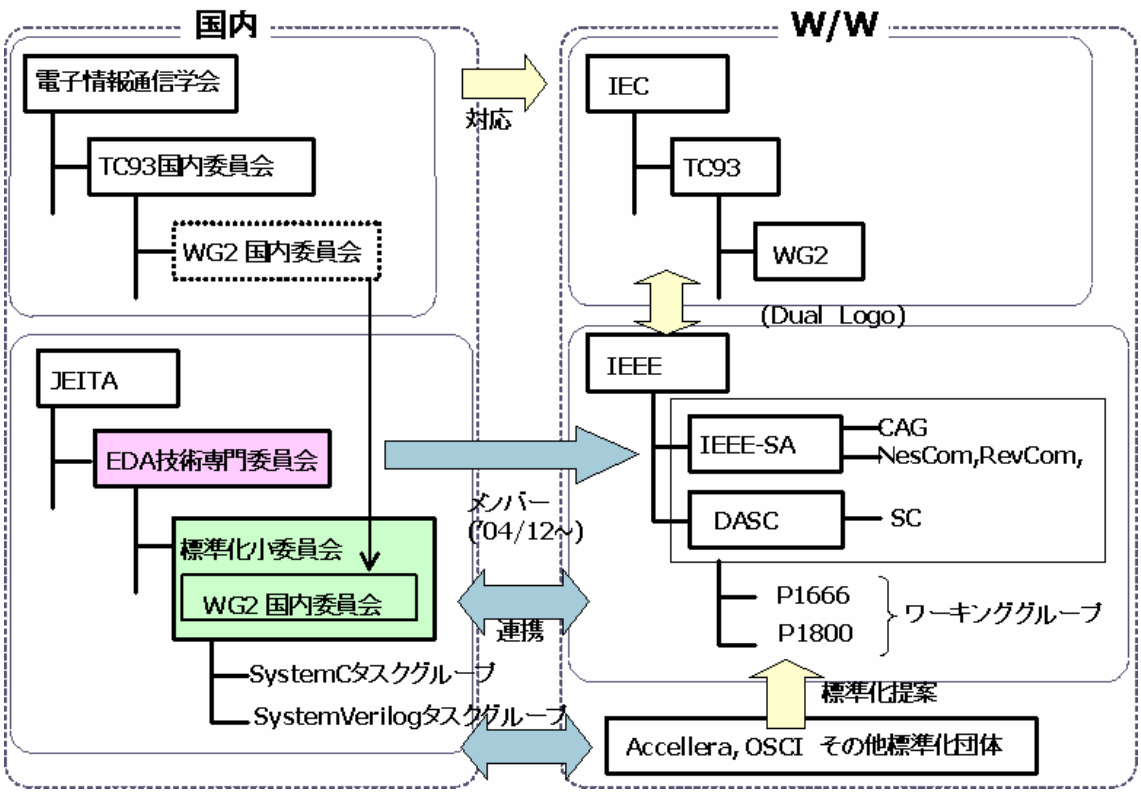


図 1　標準化小委員会と他の標準化組織との関係

2003 年度には、SystemC 及び SystemVerilog の標準化を業界として検討・推進する目的で、それぞれタスクグループを発足させた。SystemC は、ますます重要性が認識されているシステムレベルの設計言語の 1 つであり、SystemVerilog は IEEE 1364(VerilogHDL)の後継・検証技術の拡張である。両タスクグループとも、日本の標準化組織として、海外の関連団体と連携し、言語仕様の専門的な技術検討と改善提案を通じて、標準化へ貢献すること目指して活動を行っている。

本小委員会のミッションは 2002 年度作成の内規では以下のように定義した。

「本小委員会は EDA 及び関連技術の標準化に関して、

・内外の動向を調査、検討し、
・技術及び関連業界の発展に質する提案の必要性を模索し、
・必要かつ可能な場合には、関係機関に対して提案を行い、
・内外の標準化関連機関との連携・協調・協力を推進し、
・特に、デザインオートメーション/設計記述言語(TC93/WG2)WG の活動を支援し、
・また広報活動を行う。」

(2) 2006 年度標準化小委員会メンバ(2007 年 2 月現在。敬称略)

主　　査	齋藤茂美	ソニー㈱
副主査	山田　節	三洋電機㈱
委　　員	小島　智	メンター・グラフィックス・ジャパン㈱
(TC93/WG2 コ・コンベナ)		
委　　員	灘岡　満	沖電気工業㈱
同	南　文裕	㈱東芝
同	藤波義忠	NEC エレクトロニクス㈱
同	杉山八六	富士通㈱
同	中尾　徹	ローム㈱
同	西本猛史	シャープ㈱
同	秋山俊恭	㈱ルネサステクノロジ
同	田口浩文	松下電器産業㈱
同	宮山芳幸	セイコーエプソン㈱
特別委員	長谷川　隆	富士通㈱
同	明石貴昭	日本シノプシス㈱
同	後藤謙治	日本ケイデンス・デザイン・システムズ社
同	星野民夫	㈱アプリスター
同	後藤和永	NEC エレクトロニクス㈱
同	石河 久美子	富士通マイクロソリューションズ㈱
同	奥村隆昌	富士通 VLSI㈱
客　　員	今井正治	大阪大学
同	神戸尚志	近畿大学

(3) 2006 年度活動

標準化小委員会としては、年 4 回の会合を行い、傘下の SystemC タスクグループ、SystemVerilog タスクグループの活動状況の確認、標準化関連課題の議論を行った。両タスクグループは 2005 年末の SystemC、SystemVerilog の IEEE 標準化に引き続き、新規標準化項目の調査検討、設計適用事例の調査などの積極的な活動を継続した。また、2007 年 1 月のシステム・デザイン・フォーラム 2007 で両設計記述言語のユーザ・フォーラムを主催した。(詳細は各 TG 報告参照)

2006 年 9 月の IEC/TC93 ベルリン国際会議には、TC93 国内委員会幹事である神戸氏と本小委員会から浜口委員及び小委員会主査の齋藤の 3 人が参加した。(詳細は 2.2.3 章 IEC/TC93 の(5)項参照)

関連する標準化団体とも活発な交流があった。

3 月に標準化推進室が発足した STARC と、標準化全般に関して、協業と連携の話し合いを行った。4 月初めには SPIRIT コンソーシアムからの説明会が開かれた。Accellera からの Low Power 標準化提案(UPF、Unified Power Format)が 11 月の標準化小委員会及び

EDA 技術専門委員会で紹介された。12 月は TC93/WG7 と WG2 の協業案件であるテスト関係の標準化に関連し、STARC の Test 技術関係者と STIL に関する取り組み状況の情報交換と IEC/TC93 への協力のお願いを行った。Accellera と OSCI の両団体には昨年度に続き、2007 年 1 月のシステム・デザイン・フォーラム 2007 での後援とユーザ・フォーラムでの SystemVerilog と SystemC の最新標準化状況及び今後の計画の講演をお願いした。1 月の EDSFair2007 のおりには、IEEE DASC との合同会議を開催した。(内容は次節に)

2.2.2 IEEE/DASC(電気電子学会／設計自動化標準委員会)・IEEE-SA

(1) 活動の概要

IEEE は米国に本部を置く電気、電子、情報、などの国際的な学会である。また、この分野の標準化活動を長年にわたり、しかも広範囲に実施している。DASC、SA は IEEE の下部組織として、エレクトロニクス産業における設計自動化関連の標準化活動を行っている。

活動の中心は、標準設計記述言語(HDL: Hardware Description Language)の VHDL と Verilog HDL に関連する設計と検証であり、タイミング情報、論理合成、算術関数とテストの標準化に注力している。これら設計言語に関連して、システムレベルまで適用範囲を拡大して、Analog Mixed Signal、ソフトウェアとハードウェア協調設計等の拡張の標準化を検討している。2005 年には SystemC と SystemVerilog という高位設計技術言語、設計と検証を統合した記述言語の標準化作業が完了した。

(2) JEITA/EDA-TC との関連

これまでは EDA 技術専門委員会は IEEE/DASC のメンバとして関連する WG に参加し、標準化案に日本の意見を反映してきた。2004 年 12 月には IEEE-SA の正式メンバにもなり、IEEE の標準化活動にドラフトレビュー、標準化案の改善の提案、投票を通じて積極的に参加している。今年度は 2 年ぶりに EDSFair2007 の時期に IEEE DASC との合同会議を開催した(2007.1.26)。

会議には米国から DASC 議長の Berman 氏、Accellera Vice Chair の Denis 氏、Accellera の Low Power グループ議長の Bailey 氏が参加した。EDA-TC からは齋藤氏、山田氏、小島氏、神戸氏、長谷川氏、浜口氏が参加し、まず EDA-TC の活動状況を紹介した。Accellera から IEEEDASC へ、Low Power 標準化の共同検討提案があった。これに対し、日本からは Low Power に関し、もう 1 つ別の標準化の動き(Si2 による CPF、Common Power Format)があり、懸念を感じており、ぜひ 1 本化を目指してほしいとの意見を伝えた。Berman 氏からは DASC の主な WG の最新情報の簡単な照会があった。

(3) これまでの成果と現在の状況

① これまでの成果

DASC/SA ではこれまでに以下の標準化作業を行っており、そのうちのいくつかは、IEC でも標準として承認されている。

1） VHDL（Std-1076）
2） VHDL Analog Extensions（Std-1076.1）
3） VHDL Math Package（Std-1076.2）
4） VHDL Synthesis Package（Std-1076.3）
5） VHDL Timing（VITAL）（Std-1076.4）
6） Verilog HDL（Std-1364）
7） MVL-9（Std-1164）
8） Waveform and Vector Exchange（WAVES）（Std-1029.1）
9） The RTL Synthesis Interoperability Standard（Std-1076.6）
10） The Delay and Power Calculation Standard（Std-1481）
11） The Open Model Fundations Standard（Std-1499）
12） SystemVerilog（Std 1800-2005）
13） Verilog（Std 1364-2005）
14） SystemC 2.1（Std 1666）

② 現在の状況

2007 年 2 月現在の DASC 及び SA とその傘下の Working Group と Study Group は以下のとおり。P1685 の WG が今年発足した新規 WG である。

・P1076 Standard VHDL Language Reference Manual（VASG）
  ◦ VHDL-200x: the next revision
  ◦ Issues Screening and Analysis Committee（ISAC）
  ◦ VHDL Programming Language Interface Task Force（VHPI）
・P1076.1 Standard VHDL Analog and Mixed-Signal Extensions（VHDL-AMS）
・P1076.1.1 Standard VHDL Analog and Mixed-Signal Extensions - Packages for Multiple Energy Domain Support（StdPkgs）
・P1076.4 Standard VITAL ASIC（Application Specific Integrated Circuit）Modeling Specification（VITAL）
・P1076.6 Standard for VHDL Register Transfer Level（RTL）Synthesis（SIWG）
・P1364.1 Standard for Verilog Register Transfer Level Synthesis（VLOG-Synth）
・P1481 Standard for Integrated Circuit（IC）Open Library Architecture（OLA）（IEEE 1481R）
・P1499 Standard Interface for Hardware Description Models of Electronic Components（OMF）
・P1603 Standard for an Advanced Library Format（ALF）Describing Integrated Circuit（IC）Technology, Cells, and Blocks（ALF）
・P1647 Standard for the Functional Verification Language 'e'（eWG）
・P1666 Standard System C Language Reference Manual（systemc）[cosponsored with IEEE-SA CAG]
・SystemVerilog Working Group

- ◦ P1800 SystemVerilog: Unified Hardware Design, Specification and Verification Language (SV-IEEE 1800) [cosponsored with IEEE-SA CAG]
- ◦ P1364 Standard for Verilog Hardware Description Language (IEEEVerilog)
- ・P1685 SPIRIT XML Standard for IP Description (IEEE-1685)
- ・P1850 Standard for PSL: Property Specification Language (IEEE-1850) [cosponsored with IEEE-SA CAG]

2.2.3 IEC/TC93(国際電気標準会議／デザインオートメーション)

(1) 活動の概要

IECは1906年に設立された国際標準化機関であり、ちょうど本年度が100周年にあたり、9月のIEC総会では100周年の記念イベントが開催された。設計自動化を取り扱うIEC/TC93は1992年に設立された。TC93の全体会議は毎年開催されており、スイス、英、仏、米、デンマーク、日、英、米、独、伊と開催されてきた。最近では2002年10月に中国・北京市、2003年11月・2004年10月は米国・Piscataway市で開催された。2005年9月の日本の京都(奈良)に続き、今年度はドイツ・ベルリン市で開催された。

(2) TC93の組織と参加国

2003年3月現在IECのWebサイト(www.iec.ch)によれば、25カ国がTC93のメンバとなっている。IECのメンバ資格には、P(Participating)とO(Observer)の二種類があるが、Pメンバは8カ国、Oメンバが16カ国である。Pメンバとしては、日本、中国、ロシア、スペイン、フィンランド、イギリス、米国、チェコが登録されており、Oメンバとして、フランス、オーストラリア、イタリア、ベルギー、オランダ、シンガポール、エジプト、スウェーデン、ウクライナ、ハンガリ、インド、アイルランド、デンマーク、ドイツ、韓国、セルビア・モンテネグロが登録されている。

(3) TC93の組織とワーキンググループ(WG)

TC93は7つのWG/JWGから構成されている。特に、WG2、WG3、WG6、及びWG7は日本から提案も含め積極的な貢献をしてきた。今までの各WGの主な活動を示す。

WG1 : モデルのハーモナイゼーション : (a) STEP Electrical(ISO規格)とEDA標準の整合性の検討、(b) EDIFとAP-210との整合性の検討。(C)言語間のInteroperabilityの検討

WG2 : ハードウェア設計記述言語 : (a) VHDL言語仕様、Verilog HDLの整合性等の検討、システム記述言語(SLDL)も議題に取り上げられてきた。(b) IC delay&power calculation systemの検討。日本からの提案 ALR標準化 ; IS(国際規格)化完。現在はSystemC、SystemVerilogが中心。

WG3 : 設計データ交換表現 : PDX(Product data eXchange)によるマテリアルデクラレーション関連への対応の議論。

JWG11 : 記述のXML化の流れへの取り組み方の議論。

WG5　：規格適合性(コンフォマンス)テストの具体的事案の議論。

WG6　：再利用可能部品ライブラリ、日・米・欧の各プロジェクト間の仕様整合と連携の検討、日本からはJEITA/ECALSプロジェクトの成果を提案している。IBISも話題に取り上げられている。最近はマテリアルデクラレーション(MD)に関する規格案が議論の中心となっている。

WG7　：システム　テスト記述言語、ATML(Automatic Test Markup Language)の検討。

(4)　TC93国内委員会と主要メンバ(2007年2月現在。敬称略)

・TC93国際会議

議長：唐津 治夢(SRIインターナショナル)

・国内専門委員会

委員長：神戸 尚志(近畿大学)*

幹　事：古井 芳春(STARC)

委　員：齋藤 茂美(ソニー)*、柴田 明一(JPCA)、高橋 満(PartsWay)、山下 寛巳(SML)

・WG2：(ハードウェア設計記述言語)

主　査：齋藤 茂美(ソニー)*

国際コ・コンベナ、副主査：小島 智(メンター)*

委　員：長谷川 隆(富士通)*、浜口 加寿美(松下)*

・WG3：(設計データ交換表現)

主　査：神戸 尚志(近畿大学)*

・WG6：(再利用可能部品ライブラリ)

主　査：高橋 満(PartsWay、国際コ・コンベナ)

・WG7：(システムテスト記述言語)

主　査：山下 寛巳(SML)

委　員：唐津 治夢(SRIインターナショナル、国際コ・コンベナ)

*印はEDA技術専門委員会からの参加者

4月から故高木前国内委員長の代行を唐津国際会議議長が勤めてこられたが、2007年2月から、STARCの古井氏の参加を得て、上記体制となった。

(5)　TC93ベルリン会議の報告

2006年の国際会議は、9月にドイツのベルリン市内、帝京ドイツホテルで開催された。

参加国は、米国と日本のみで、ここ数年同様、課題を残した。日本からは米国を上回る6名が参加した。米国からはTC93メンバに加え、IEEEから1名の参加があり、デュアルロゴ関係での議論が進んだ。

ベルリン会議では、TC93プレナリー会議、WG1、WG2、WG3、WG5、WG6、WG7、JWG11の8会合が開催された。

写真１　TC93 ベルリン会議風景

WG2 では、まず、国際コ・コンベナの Alex 氏から IEEE DASC、Accellera、SPIRIT など関連する米国の標準化団体の状況報告があった。日本からは EDA 技術専門委員会の活動状況を紹介するとともに SystemVerilog と VerilogHDL の統合化にあたっては設計資産継承のために互換性に配慮が必要と提案した。

TC93 ベルリン会議 WG2 まとめ

① デュアルロゴメンテナンス：IEEE とのデュアルロゴのメンテナンスを IEC トリガ（IEC 61691-4: Verilog）と IEEE トリガ（IEEE 1076：VHDL）の 2 方向で進める。

② 新デュアルロゴ標準化候補：

(ｱ) IEEE 1076.1.1-2004, IEEE VHDL Analog & Mixed-Signal Extensions

(ｲ) IEEE Std 1800-2005, IEEE, SystemVerilog

③ 注目すべき標準化領域：

(ｱ) SOC - SPIRIT：IP_XACT（コンソーシアム, IEEE PAR 1685）

(ｲ) ESL：Esterel　“reactive system”記述用高位言語（コンソーシアム）
Rosetta　マルチドメインモデル用言語（DASC に検討 GP）
AADL　アーキテクチュア記述用標準 SAE

(ｳ) 低消費電力：Accellera、Si2 などで検討中

④ 言語領域を含む Test 標準化案件につき、WG7 と協力していくことを決めた。

(6) IEC 規格投票について

年度末の 3 月に以下 7 件の IEEE 標準をデュアルロゴとして、First Track で IEC 標準とする案件が提案され、標準化小委員会（TC93/WG2）として、国内意見をとりまとめ中である。

このうち、Item 1～5 は Test 関連で前項の WG2 - WG7 の協力案件に相当し、STARC の Test 関係 Expert である相京氏を WG2 Expert として迎えて、対応予定である。また、Item 6 は TC93 ベルリン会議でデュアルロゴ案件としてリストアップ済の案件である。

Item 1: 1450-1999 Standard Test Interface Language (STIL) for Digital Test Vector Data

Item 2: 1450.1-2005 Standard for Extensions to Standard Test Interface Language (STIL) for semiconductor Design Environments

Item 3: 1450.2-2002 Standard for Extensions to Standard Test Interface Language (STIL) for DC Level Specification

Item 4: 1500-2005 Standard Testability Method for Embedded Core-based Integrated Circuits

Item 5: 1641-2004 Standard for Signal and Test Definition

Item 6: 1800-2005 Standard for System Verilog: Unified Hardware Design, Specification and Verification Language

Item 7: 1850-2005 Standard for Property Specification Language (PSL)

2.2.4 Accellera（設計記述言語の標準化機関）

2.2.4.1 活動の概要

設計生産性を改善するには、世界標準とオープンインタフェースに基づいた設計方法論が重要である。Accellera は、Open Verilog International(OVI)と VHDL International(VI)を統合して 2000 年に設立された。設計記述言語を中心に新しい標準の認定、標準の開発及びこれに基づいた新しい設計方法論の普及促進を行っている。

メンバは、半導体メーカ、システムメーカと EDA ベンダから広く参加しており、ハードウェア記述に留まらずシステム記述までを包含する設計記述言語を検討している。

2.2.4.2 現在の活動状況

Technical Committee を設置し、設計記述言語の開発と策定を行っている。Technical Committee は、専門的に検討を進める SubCommittee を下部組織として持つ。また、Accellera で認定した標準は、国際標準として IEEE に提案することが慣例になっており、関連の深い IEEE の Working Group や他の標準団体を支援している。昨年度は、Accellera 標準である SystemVerilog が IEEE 1800 として IEEE の標準となった。

2007 年 2 月時点の Accellera の Technical Committee は以下のとおりである。

Interface Technical Committee

- **Chair**: Brian Bailey
- **Co-Chair**: Damian Denault, Zaiq Technologies
- Committee home page

IEEE P1800 SystemVerilog

- **Chair:** Karen Pieper, Synopsys
- Committee home page
- Check the **SystemVerilog page** for upcoming events and to download the LRM

SystemVerilog - Basic

- **Chair**: Matt Maidment, Intel
- Committee home page

SystemVerilog - Enhancement

- **Chair**: Medhi Mohtashemi, Synopsys
- Committee home page

SystemVerilog - C Interface

- **Chair**: Charles Dawson, Cadence
- Committee home page

SystemVerilog - Assertion

- **Chair**: John Havlicek, Freescale
- Committee home page

SystemVerilog - XC (Cross Hardware Description Language)

- **Chair**: **Somdipta Basu Roy**, TI
- Committee home page (under construction)

Open Compression Interface Technical Commitee

- **Chair**: Bruce Cory, NVidia
- **Vice Chair**: Kee-sup Kim, Intel
- **Secretary**: Mick Tegethoff, Cadence
- Committee home page

Open Verification Library (OVL) Technical Committee

- **Chair:** Mike Turpin, ARM
- **Co-Chair**: Kenneth Larsen, Mentor Graphics
- Committee home page

Unified Coverage Interoperability Standard

- **Chair**: Faisal Haque, Cisco Systems
- Committee home page

Unified Power Format Technical Committee

- **Chair**: Stephen Bailey, Mentor Graphics
- Committee home page

Verilog AMS Technical Committee

- **Chair**: Sri Chandrasekaran, Freescale
- Committee home page

IEEE P1850 PSL

- **Chair**: Harry Foster, Mentor Graphics
- Committee home page
- Download the Accellera Property Specification Language (PSL) LRM
- View the PSL Whiteboard Presentation by Harry Foster
- Conditions of Use for Accellera's PSL LRM

IEEE 1076 VHDL (Accellera Technical Committee)

- Chair: Lance Thompson, IBM
- Committee home page

2.2.4.3 新しい動きと EDA-TC との関係

今年度は、新たに Low Power の標準化検討ワーキング GP、Low Power Format Technical Committee が発足し、EDA-TC に対しても、小島委員を通して、11 月に標準化の背景と今後の IEEE 標準化を目指したスケジュールが紹介された。Accellera は 1 月横浜での IEEE DASC と EDA-TC との合同 MTG において、この標準(UPF)を IEEE へ提案するとともに標準化 Draft を一般公開した。

標準化小委員会は、2001 年 4 月から Associate Member として Accellera に参画しており、継続的に標準活動への参加と情報交換を行っている。本年度も、昨年度に引き続き Accellera の Vice Chirman を招き 2007 年 1 月横浜で開催した EDSFiar2007 と併設のシステム・デザイン・フォーラム 2007 の SystemVerilog ユーザ・フォーラム 2007 において、米国での最新の標準化動向と今後の標準化ロードマップの紹介・発表をお願いした。

2.2.5 SystemC タスクグループ報告

(1) 背景

ハードウェア記述言語によるシステム LSI の設計は、VHDL(IEEE 1076)や Verilog-HDL(IEEE 1364)の標準化への JEITA(旧 EIAJ)の貢献とともに広く普及して、産業界で活用されている。一方、半導体の微細化技術は開発がさらに加速され、既に 1000 万ゲート規模の LSI が開発されるに至り、さらに抽象度の高いレベルからの設計が必須となってきている。1990 年代半ばより複数のシステムレベル設計言語の提案が行われ、標準化推進団体が結成されたものもあった。この中で、C++言語を基本とする SystemC は広く半導体メーカ、システムメーカ、EDA ベンダの賛同を得て、Open SystemC Initiative(OSCI)が結成され、標準化のための言語仕様の策定と整備が進められてきた。

システムレベル設計言語としての要件を備えた SystemC 2.0 のリファレンスシミュレータがまず2001年10月にリリースされ、その後2003年5月に言語参照マニュアル(Language Reference Manual、以下 LRM)が一般公開された。この LRM が 2004 年 11 月に OSCI より IEEE に移管され、IEEE P1666 として正式な標準化プロセスが開始された。並行して OSCI にて開発されていた SystemC 2.1 の言語拡張仕様も IEEE P1666 標準の一部として追加移管され、2005 年 12 月に IEEE Std. 1666-2005 として SystemC のコア言語部分の標準化が完了した。

(2) 目的

上記のような状況の中で、SystemC は SoC(System on Chip)の開発のためのシステムレベル記述言語の 1 つとして既に設計や検証に幅広く使われるようになり、欠くことのできない言語となってきている。設計言語は設計の基本となるもので、この標準化策定に早くから

関わることは、産業界にとって次世代の設計手法を構築する上で非常に重要なことである。

本タスクグループは 2003 年 10 月に設置され、日本国内における唯一の SystemC の標準化関連組織として、IEEE P1666 で進められる SystemC 標準作業に対して日本の産業界として意見を述べ、国内事情・要求事項を取り込んだ形で国際標準化に貢献していく。また、SystemC に関連した調査結果をアニュアルレポートやユーザ・フォーラム等で積極的に情報発信を行うことで、SystemC を利用した設計手法の国内普及を図り、ひいては日本の産業界の国際競争力を高めることを目指す。

具体的には、次の 3 つの項目を柱として活動する。

① SystemC 標準化活動
② SystemC 技術調査
③ SystemC 普及活動

(3) これまでの成果

2003 年 10 月に発足した後、これまでに次のような成果をあげた。

① SystemC 標準化活動

・2003 年度は OSCI より 2003 年 5 月に一般公開された LRM についてレビューを行い、問題点を 62 件抽出し(うち 46 件については 2003 年度の活動報告書に一覧を記載)、IEEE 並びに OSCI に報告した。
・2004 年度は IEEE P1666 のメンバとして活動を行い、IEEE 版の LRM(Draft)をレビューし、43 件の問題点を IEEE に報告した。
・2005 年度は SystemC 2.1 が追加された IEEE P1666 版 LRM についてレビューを行い、19 件の問題点を抽出し IEEE に報告した。2005 年 12 月 5 日に IEEE Std. 1666-2005 として SystemC の基本言語部分の標準化が完了した。プレスリリースも発行され、EDA-TC/JEITA としてもコメントを掲載した。
・2006 年度は Pre-IEEE 標準化作業として、OSCI よりリリースされた TLM 2.0 ドラフト、及び合成サブセットドラフトについてレビューを行い、問題点や要望事項を OSCI に伝えた。これらが 2007 年度中に対処された後に IEEE にドネーションされ、正式な標準化作業が行われる予定である。
  ✧ TLM 2.0 ドラフトに対し 4 件提案(詳細は 4.2.3 を参照)
  ✧ 合成サブセットドラフト(1.1.18)に対し 57 件提案(詳細は 4.2.4 を参照)

② SystemC 技術調査

・2003 年 11 月度に集中審議を行い、本タスクグループ参加各社の SystemC 利用状況について紹介しあい、業界内の現状ステータスについて理解を深めた。
・2004 年度には、過去 5 年間に一般に公開されている SystemC 関連の論文や発表資料等 50 件の調査を行い、報告書を作成した。
・2005 年度は、SystemC 2.1、TLM(トランザクションレベルモデリング)、合成サブセットのテーマを定め、それぞれ分科会形式で掘り下げた調査を行った。

・2006 年度は TLM に関する動向調査を実施し、結果を SystemC ユーザ・フォーラム 2007 にて公表。欧州ユーザと比較し、国内ユーザは低抽象度のモデルを利用する傾向が高いことを掴んだ。TLM の標準化が進んでいないため再利用性があまり高くないため、RTL 設計に近いレベルでの利用に留まっていると予想される。欧州では社内で閉じた使い方ではあるが、一丸となって利用を勧めているようである(添付資料参照)。2006 年度の調査対象は次のとおり。

- ✧ TLM with SystemC(書籍)の分析
- ✧ ECSI TLM Work Shop 2006 の分析
- ✧ SystemC タスクグループメンバ各社での利用状況の分析

③ SystemC 普及活動

・2004 年度より、それまでの OSCI から引き継いで JEITA EDA 技術専門委員会の主催で SystemC ユーザ・フォーラムを開催している。

・2005 年 1 月 27 日に SystemC ユーザ・フォーラム 2005 を開催。受講料は無料。定員 200 名のところ 250 名弱の聴講者が訪れ、立ち見が出るほどの盛況であった。

・2006 年 1 月 27 日に SystemC ユーザ・フォーラム 2006 を開催。今回より、受講料を徴収することにした。(SystemC 単独の場合¥1,600、SystemVerilog と通しの場合¥2,000) 定員 200 名のところ、申し込みは完売したが、実際に会場に訪れた聴講者は 172 名であった(前年比 30%減)。また、アンケートは 134 名の方に記入いただけた。

・2007 年 1 月 26 日に SystemC ユーザ・フォーラム 2007 を開催。前回より値上げした影響か、聴講者は 148 名と減少した(前年比 14%減)。また、アンケートは 132 名の方に記入いただけた(内容等の詳細については 4.2.3 を参照)。

アンケート調査では、次のようなコメントが寄せられた。次回以降開催時の参考としたい。

- ➢ 入場券の半券を残して欲しい
- ➢ 会場前の待機場所があるとよい
- ➢ 発表中にフラッシュを使った写真撮映を止めてほしい
- ➢ 事例紹介を増やして欲しい、TLM の技術的内容、TLM ユーザ事例を聞きたい
- ➢ 日本国内における標準化活動スケジュールの明確化

・2003 年～2007 年開催の際のアンケート調査結果と合わせて聴講者の動向について分析を行った。大まかな傾向としては、主な使用言語は Verilog HDL が相変わらず多数を占めるが、SystemC に関しては様子見の段階から(部分的)使用の段階へ移行しつつあるようだ(ただし、2004 年以降大幅な変化はない)。また、日本国内だけの特色と思われるが、聴講者の一番の興味は SystemC を使った高位合成のようである(欧米とは少々異なる)。2006 年度から特に伸びたのは TLM や等価性検証についてであり、実用レベルで SystemC を使う上で必要な技術が広まってきたことが窺われる。

(4) 参加メンバ

主　　査	長谷川　隆	富士通㈱
副 主 査	今 井 浩 史	㈱東芝
委　　員	中 西 早 苗	NEC エレクトロニクス㈱
同	清 水 靖 介	沖電気工業㈱
同	長 尾 文 昭	三洋半導体㈱
同	山 田 晃 久	シャープ㈱
同	柿 本　勝	ソニー ㈱
同	逢 坂 孝 司	日本ケイデンス・デザイン・システムズ社
同	中 野 淳 二	日本シノプシス㈱
同	竹 村 和 祥	松下電器産業㈱
同	菊 谷　誠	メンターグラフィクスジャパン㈱
同	里 井 朋 樹	㈱リコー
同	渡 邊 政 志	㈱ルネサステクノロジ
客　　員	今 井 正 治	大阪大学

（計 14 名）

2.2.6 SystemVerilog タスクグループ報告

(1) 背景

Verilog HDL（ハードウエア記述言語）は、特定ツールの独自言語として開発され、その後、OVI（Open Verilog International）による言語仕様公開を経て、1995 年に IEEE 1364 として標準化の承認がなされた。さらに 5 年後、ディープサブミクロン対応の機能が盛り込まれ、IEEE 1364-2001 として改訂された。JEITA（旧 EIAJ）の EDA 技術専門委員会では、IEEE 1364 の標準化作業が開始されると同時に Verilog HDL 標準化プロジェクトを設置し、継続的に言語仕様の技術検討・国際標準化に貢献してきた。

その後、半導体の微細化技術はさらに進化し 1000 万ゲート規模の LSI が開発されるに至り、一般的に「論理設計工数の 7～8 割が機能検証に費やされているにも関わらず、6 割近い LSI が機能バグ等の問題によりリ・スピンしている」という報告がなされた（Collett International）。このような状況において、機能検証の網羅性と検証効率を飛躍的に向上させるために、2000 年以降に、新しいテストベンチ記述、アサーション／プロパティ記述など、いくつかの検証用言語が実用化された。

SystemVerilog は、Verilog HDL に、①デザインに対する記述構文 ②検証用言語を追加したものである。デザインの面でも Verilog HDL に比べ記述量の削減や曖昧性の排除といったメリットがあり、設計品質の向上が期待できるが、新たに検証用言語の機能を持たせたことが大きな特徴である。Accellera が SystemVerilog の言語仕様を、SystemVerilog V3.1a としてまとめた。その後、IEEE での標準化作業を経て、2005 年に IEEE 1800 として標準化された。

2008 年には、Verilog HDL の IEEE 1364 と、SystemVerilog の IEEE 1800 が 1 つの標準言語として統合される予定である。本タスクグループは、この統合作業の完了を活動のゴールとしている。

(2) 目的

SystemVerilog 標準化において、日本の半導体産業界の要望に沿った形で言語標準化を進めることが、適用容易性を高め、設計品質の向上につながり、国際的な競争力確保といった結果になる。本タスクグループでは、業界各社から参加したエキスパートにより SystemVerilog 言語仕様の技術的な検討を実施し国際標準化に貢献すること、SystemVerilog に関する最新情報の収集と情報発信、日本国内での SystemVerilog の普及推進を図ることを目的としている。

(3) 活動内容

① グループの結成と言語習得(03 年度)

03 年 10 月に、SystemVerilog 言語仕様の技術検討・標準化を推進するためのタスクグループとして「SystemVerilog タスクグループ」を結成した。03 年度は、SystemVerilog の代表的な言語仕様をまとめた資料を作成した。

② 国際的な情報収集・標準化組織との連携(03～06 年度)

04 年 3 月と 07 年 2 月に、米国で開催された国際学会 DVCon(Design & Verification Conference)にメンバを派遣し、最新技術動向の情報収集を行った。04 年は、Accellera の SystemVerilog 会議に本タスクグループの主査と副主査が参加し JEITA での活動内容を紹介するとともに、今後連携して活動を推進することで合意した。その後、本タスクグループでの活動成果を IEEE の標準化作業に反映することを目的として、提案の優先度アップと投票権の取得を可能にする IEEE の標準化機関 IEEE-SA のメンバになった。そして、05 年 1 月に IEEE P1800 の投票グループにはいり、05 年度の最終投票に参加し標準化推進に貢献した。2007 年 2 月に、3 年ぶりに米国で開催された DVCon にメンバを派遣し、米国での SystemVerilog の実用化動向を調査するとともに、ツールベンダに早急な SystemVerilog の完全なサポートを要求した。また、最新の標準化情報を収集し、2007 年度の活動スケジュールを策定した。

IEEE での標準化の経過・最新状況については、4.3 章に「SVTG-1」として報告資料を掲載する。本資料は、EDS フェア 2007 の併設イベントとして開催した SystemVerilog ユーザ・フォーラムで、Accellera の副主査である Dennis Brophy 氏が使用した発表資料である。

また、2007 年の DVCon への出張報告も 4.3 章に「SVTG-4」として掲載する。

③ 言語仕様検討と IssueReport の提出(03～05 年度)

SystemVerilog 言語仕様検討において、より議論を深めるため、「デザイン」と「検証」の 2 つサブタスクグループにわかれ、専門的な技術ディスカッションを重ねた。そして、04 年 8 月に、改善提案を含む 32 件の IssueReport をまとめ、Accellera と IEEE に提出

した。本案件は、IEEE P1800-WG の審議対象として DB に登録され標準化内容に反映された。

④　SystemVerilog/SystemC の対訳表の作成（05～06 年度）

SystemVerilog の専門用語に関し、多くの翻訳書や EDA ベンダが提供するユーザマニュアル類で多様な訳語が使われると利用者の混乱を引き起こすため、標準的な訳語を定義することを目的として対訳表を作成した。SystemC タスクグループとも連携し、SystemC と SystemVerilog 共通の対訳表を 1 つにまとめた。本対訳表を標準訳語として、出版社・EDA ベンダをはじめ、業界全体に公開し、適用してもらえるよう推進する。対訳表は、4.4 章に「SyetemC/SystemVerilog 専門用語対訳表」として掲載する。

⑤　SystemVerilog ユーザ・フォーラムの開催（04～07 年度）

05 年 1 月、06 年 1 月、07 年 1 月に、EDS フェア併設の「システム・デザイン・フォーラム」のカリキュラムの 1 つとして「SystemVerilog ユーザ・フォーラム」を 3 年連続開催した。

このフォーラムでは、Accellera・IEEE P1800-WG の標準化状況の説明、本タスクグループからは、SystemVerilog の特徴・利点を広く周知させるための「SystemVerilog 言語チュートリアル」を 3 部に分けて実施した。今年度は、テストベンチ記述をテーマとしたチュートリアルを行った。ユーザ・フォーラムのアンケート結果によると、主に使用している言語で SystemVerilog を選択した人が昨年度に比べ 3 倍に増加している。また、SystemVerilog を検証に適用した人も 3 倍に増加していることがわかった。

今年度実施したテストベンチのチュートリアルのテキストは 4.3 章に「SVTG-2」として掲載する。

⑥　デザインガイアでの SystemVerilog チュートリアル開催

06 年 11 月に、九州小倉で開催されたデザインガイアで招待講演として「SystemVerilog チュートリアル」を行った。内容は、05 年・06 年の SystemVerilog ユーザ・フォーラムで実施した 2 回のチュートリアルをまとめたものである。出席者は 100 名程度で好評であった。

このチュートリアルのテキストを 4.3 章に「SVTG-3」として掲載する。

(4)　関連機関の動向

IEEE では、2006 年 6 月 28 日に 1800PAR（Project Authorization Request）が承認され、次の標準改訂が正式にスタートした。本改訂では、並存していた IEEE Std.1364（Verilog HDL）と IEEE Std. 1800（SystemVerilog、現状は Verilog HDL に対する拡張と位置付けられている）について、Std. 1800 を存続させ、両者を統合する。これにより SystemVerilog のユーザである設計者や EDA ツールの開発者は、2 つの言語仕様を参照する必要がなくなる。合わせて本改訂では、誤りの修正、アサーションを始めとする機能の拡張、AMS との統合、SystemC や VHDL 等の言語との相互接続の確立を実施する予定になっている。本改訂に関する今後の主なスケジュールの予定は以下のとおりである。

2007 年 3 月 15 日　統合された最初のドキュメント作成完了
2007 年 4 月 30 日　統合されたドキュメントの内部ドラフト
2007 年 6 月　　統合されたドキュメントの公式リリース
2007 年 10 月 1 日　中間ドラフト
2008 年 1 月 15 日　投票に向けた事前ドラフト
2008 年 3 月 30 日　投票用のドラフト(投票実施、5/15 結果発表)
2008 年 7 月 15 日　P1800 委員会による承認

各ドラフトについて意見を提出する機会があるので、本タスクグループとして、適宜レビューを実施し、ユーザとしての立場で意見・要望を提出することにより、標準改訂に協力する。

(5)　4.3 章の添付資料について

SVTG-1　SystemVerilog IEEE 1800Updates
SVTG-2　SystemVerilog ユーザ・フォーラム発表資料　テストベンチチュートリアル
SVTG-3　デザインガイア発表資料　SystemVerilog(IEEE Std. 1800-2005)チュートリアル
SVTG-4　DVCon2007 出張報告

(6)　参加メンバ

主　査	浜口 加寿美	松下電器産業㈱
副 主 査	明 石 貴 昭	日本シノプシス㈱
委　員	湯 井 丈 晴	㈱沖ネットワークエルエスアイ
同	後 藤 謙 治	日本ケイデンス・デザイン・システムズ社
同	岡 本 実 幸	三洋電機㈱
同	土 屋 丈 彦	㈱東芝
同	千 綿 幸 雄	富士通㈱
同	竹田津 弘州	松下電器産業㈱
同	李　建 道	メンター・グラフィックス・ジャパン㈱
同	高 嶺 美 夫	㈱ルネサステクノロジ
同	杉 浦 正 志	㈱図研

3.　各種イベント(主催／協賛)報告

3.1 Electronic Design and Solution Fair 2007 (EDSFair2007)

情報化・ユビキタス社会の基盤となる情報機器向け電子システムや半導体の開発に不可欠となる、最先端設計ソリューション(EDA 技術、IP 再利用技術、組込みソフトウェア技術、各種設計サービスなど)の情報発信・普及促進を行い、関連業界の発展に寄与することを目的として、2007 年 1 月 25 日(木)～26 日(金)の 2 日間、パシフィコ横浜 展示ホール/アネックスホールにおいて「Electronic Design and Solution Fair 2007(略称 EDSFair 2007)」を開催した。

EDSFair2007 の会場風景

本展示会は、2001 年に米国の Electronic Design Automation Consortium(EDAC)との協力関係を締結して以来、米国の Design Automation Conference(DAC)、欧州の Design, Automation and Test in Europe(DATE)と並ぶ国際コンベンションとして位置付けられている。今後も、電子システム・半導体設計に関する世界 3 大展示会の 1 つとして、最先端の設計ソリューション・設計技術・EDA 技術情報を提供し続けていきたいと考えている。

1993 年から、当時の㈳日本電子機械工業会(EIAJ、現 JEITA)が開催していた「EDA TechnoFair」と、FPGA/PLD ベンダ各社及び大学関係者が組織して実施してきた「FPGA/PLD Design Conference & Exhibit」を 2001 年に統合して以来、今回で第 7 回目の開催を迎えることとなった EDSFair2007 では、キャッチフレーズを“ここにある世界の先端テクノロジ。”として、半導体プロセスの微細化が進み、新しいソリューションが求められる時代に即したブース展示を行うとともに、当該業界の第一人者にご講演いただくキーノートスピーチ、特別講演、展示会場内特設ステージでの「今さら聞けないことがわかる！」と題した若い技術者・設計者向けのセッションを開催した。また、出展各社による出展者セミナーや産学官の技術交流を深める「ユニバーシティ・プラザ」も開催した。

3.1.1 EDSFair2007 の概要

(1) 開催期間：2007 年 1 月 25 日(木)～1 月 26 日(金) 2 日間

(2) 場　　所：パシフィコ横浜(展示ホール、アネックスホール)

(3) 主　　催：社団法人電子情報技術産業協会(JEITA)

協　　力：Electronic Design Automation Consortium(EDAC)

後　　援：経済産業省、アメリカ合衆国大使館、外国系半導体商社協会(DAFS)、横浜市経済観光局(順不同)

協　　賛：社団法人電子情報通信学会(IEICE)、社団法人 情報処理学会(IPSJ)、社団法人日本電子回路工業会(JPCA)（順不同）

特別協賛：サン･マイクロシステムズ㈱

日本ヒューレット･パッカード㈱

運　　営：有限責任中間法人 日本エレクトロニクスショー協会(JESA)

(4) 開催概況

① 入場者数：11,136 名(前年 11,003 名)

② 出展者数：154 社／348 小間(前年 148 社 343 小間)

③ 出展者セミナー：120 セッション、延べ聴講者数 3,042 名

④ スイートルーム：4 社

⑤ ユニバーシティ・プラザ：17 ブース、17 大学研究室

⑥ キーノートスピーチ：聴講者数 343 名

⑦ 特別講演：聴講者数 97 名

⑧ SPIRIT 講演：聴講者数 30 名

⑨ 併催

第 14 回 FPGA/PLD Design Conference：10 セッション、聴講者数 427 名

・ユーザ・プレゼンテーション：5 プレゼンテーション、25 日、聴講者数 22 名

・IP フリーマーケット in EDSFair：12 テーマ、25 日、聴講者数 38 名

・イブニングセッション：聴講者数 32 名

⑩ 同時開催

システム・デザイン・フォーラム 2007：3 セッション、聴講者数 420 名

3.1.2 出展カテゴリー

(1) ハードウェア・ソリューション

システム LSI、ASIC/ASSP、MPU/MCU/DSP、FPGA/PLD デバイス、他

(2) ハードウェア開発環境(EDA)

① LSI 設計関連ツール

システムレベル設計(RTL より高位)、論理設計(RTL～ネットリスト)、論理検証、アナログ設計・検証、レイアウト、レイアウト検証・解析、LSI 信号解析、テスト設計(DFT/BIST/ATPG など)、DFM 関連(OPC/RET/PSM/LRC/TCAD など)、他

② PCB 設計関連ツール

回路図作成、アナログ設計・検証、レイアウト、PCB 信号解析、他

③ SiP 設計関連ツール

(3) ソフトウェア・ソリューション

組込み OS、デバイスドライバ、ファームウェア、ミドルウェア、他

(4) LSI テスト、計測器

LSI テスタ、PCB テスタ、計測器、他

(5) IP コア、マクロ、セルライブラリ

(6) 組込みプロセッサ開発環境

リコンフィギャラブルプロセッサ、ICE、デバッガ、マイコン CASE、コンパイラ/クロスコンパイラ、シミュレータ、ハード/ソフト協調設計環境、他

(7) 設計サービス関連

デザインセンタ、設計サービス、設計コンサルティング、IP 流通サービス、他

(8) 設計インフラ(WS/PC、ネットワーク)

(9) 設計データ管理ツール

設計データ管理、他

(10) マスクメーカ、ファウンダリメーカ

(11) 大学(研究室)、コンソーシアム

(12) PR 関連

出版物、他

3.1.3 開会式

1 月 25 日(木)午前 9 時 50 分より展示会場入口において開会式を執り行った。開会式への登壇者は次のとおりである。

・ご祝辞･テープカット：

経済産業省 商務情報政策局　情報通信機器課長 横尾 英博 様

横浜市 経済観光局ライフサイエンス　新産業誘致担当政策専任部長 長島 敏晴 様

・主催者挨拶：

社団法人 電子情報技術産業協会　半導体部会 部会長 室町 正志 様

・主催者テープカット:

社団法人 電子情報技術産業協会　専務理事 金子 和夫 様

開会式終了後、登壇者及び関係者による会場一巡が行われ、本年は下記のブースを訪問し、新技術・研究開発等の成果の説明を行った。

デナリソフトウエア㈱、ユニバーシティ・プラザ、日本シノプシス㈱、新興ベンダエリア、NEC システムテクノロジー㈱、㈱半導体理工学研究センター、(以上、見学順)

開会式でのテープカット

3.1.4 出展者一覧

会社名	小間数	会社名	小間数
アーム㈱	4	㈱クレディスト	2
アイヴィス	3	コーウェア㈱	10
アジレント・テクノロジー㈱	4	サイバーテック㈲	4
㈱アストロン	2	ジャスパー・デザイン・オートメーション	—
アットデザインリンクス㈱	1	サイバネットシステム㈱	3
Accelicon Technologies, Inc.	—	サガンテック・ノース・アメリカ・インク	2
アトレンタ㈱	6	CQ 出版㈱	1
アパッチデザインソリューションズ㈱	2	シーケンスデザイン㈱	4
㈱アプライド・シミュレーション・テクノロジ	1	㈱ジーダット	6
㈱アプリスター	1	ジェネシス・テクノロジー㈱	1
アンソフト･ジャパン㈱	6	シエラ・デザイン・オートメーション㈱	4
E2 パブリッシング㈱	1	GiDEL	1
㈱礎デザインオートメーション	1	㈱シルバコ・ジャパン	9
伊藤忠テクノソリューションズ㈱	9	シンプリシティ㈱	8
REAL INTENT, INC.	—	㈱図研	8
Prolific Inc.	—	㈱スピナカー・システムズ	1
SARNOFF Corporation	—	ソニックス	2
Xyalis	—	㈱ソリトンシステムズ	2
ダッソー・システムズ㈱	—	Aldec, Inc.	—
サン・マイクロシステムズ㈱	—	GE Fanuc Embedded Systems, Inc.	—
EMC ジャパン㈱	—	MOSAID Technologies Inc.	—
マクニカネットワークス㈱	—	Novocell Semiconductor, Inc.	—
Actis Design, LLC	—	Y Explorations, Inc.	—
ANOVA SOLUTIONS INC	—	ダイキン工業㈱	2
日本セロックシカ㈱	—	巧テクノロジー㈱	2
日本イヴ㈱	—	タナーリサーチジャパン㈱	3
Obsidian Software, Inc.	—	DSM ソリューションズ㈱	1
Blaze DFM, Inc.	—	デジタルテクノロジー㈱	2
DAFCA, Inc.	—	日本アイ・ビー・エム株式会社	—
イノテック㈱	9	デナリソフトウエア㈱	9
アルテリス	—	テンシリカ㈱	1
ビーチソリューションズ	—	㈱電波新聞社	1
チップビジョンデザインシステムズ	—	TOOL㈱	2
イーエイシック	—	日経 BP 社	1
ジャズセミコンダクター	—	日本システムウエア㈱	2
ターゲットコンパイラーテクノロジーズ	—	日本アルテラ㈱	6
トライアントテクノロジーズ	—	日本イヴ㈱	4
エイシップ・ソリューションズ㈱	2	日本ケイデンス･デザイン･システムズ社	20
㈱エーイーティー	1	イノテック㈱	—
AWR Japan ㈱	1	日本シノプシス㈱	20
㈱エスケーエレクトロニクス	2	日本セロックシカ㈱	4
㈱エッチ・ディー・ラボ	1	日本ノーベル㈱	1
㈱エッチ・ディー・ラボ EDA 事業部	1	エース・アソシエイテッド・コンパイラ・エキスパート	—
NEC システムテクノロジー㈱	7	㈱ネットウエル	1
特定非営利活動法人 FPGA コンソーシアム	1	ノバフロー㈱	4
㈱沖ネットワークエルエスアイ	2	Novas Software, Inc	—
カーボン・デザイン・システムズ・ジャパン㈱	2	Silicon Canvas, Inc.	—
㈱ガイア･システム・ソリューション	1	ForteLink, Inc.	—
兼松エレクトロニクス㈱	3	BERKELEY DESIGN AUTOMATION, Inc.	2
カリプト・デザイン・システムズ㈱	2	パルシックジャパンリミテッド	1
㈱キー・ブリッジ	1	㈱PALTEK	2

会社名	小間数
㈱半導体理工学研究センター	10
HANDSHAKE Solutions	1
日立情報通信エンジニアリング㈱	2
㈱日立超 LSI システムズ	1
FISHTAIL DESIGN AUTOMATION	1
フォルテ・デザイン・システムズ㈱	3
㈶福岡県産業・科学技術振興財団	4
㈱セイリング	—
㈲アナロジスト	—
㈱エム ディ アイ LSI開発センター 福岡事業所	—
ブライオンテクノロジーズ㈱	4
㈱プライムゲート	2
プラットフォームコンピューティング㈱	1
プロトタイピング・ジャパン㈱	2
マグマ・デザイン・オートメーション㈱	9
丸紅ソリューション㈱	8
㈱ミッシュインターナショナル	2
三菱電機エンジニアリング㈱	2
三菱電機マイコン機器ソフトウエア㈱	1
メンター・グラフィックス・ジャパン㈱	25
メンター・グラフィックス・ジャパン㈱	4
リード・ビジネス・インフォメーション㈱	1
OneSpin Solutions	1
新興ベンダエリア	
ADVANCED RFIC (S) PTE LTD	1
ADVINNO TECHNOLOGIES PTE LTD	1
APRIO TECHNOLOGIES, INC.	1
AXIOM DESIGN AUTOMATION	1
AZURO, INC.	1
BEACH SOLUTIONS INC.	1
BITROUTER	1
CEBATECH, INC.	1
CHIPVISION DESIGN SYSTEMS AG	1
ENTASYS DESIGN, INC.	1
HELIC S.A.	1
LIBRARY TECHNOLOGIES, INC.	1
MUNEDA GMBH	1
PYXIS TECHNOLOGY, INC	1
RIDGETOP GROUP INC.	1
SIDENSE CORPORATION	1
SOFTJIN TECHNOLOGIES PRIVATE LIMITED	1
SOLIDO DESIGN AUTOMATION INC.	1
TARGET COMPILER TECHNOLOGIES N.V.	1
TENISON DESIGN AUTOMATION	1
THE SPIRIT CONSORTIUM INC.	1
VERIFIC DESIGN AUTOMATION	1
共信テクノソニック㈱	1
㈲シンテスト・ジャパン	1
新興ベンダエリア/JEVeC ビレッジ	
ギガヘルツテクノロジー㈱	1
ケイレックス・テクノロジー㈱	1

会社名	小間数
㈱システム・ジェイディー	—
㈱数理システム	1
日本 EDA ベンチャー連絡会(JEVeC)	1

合計 154 社 348 小間

※会社名 50 音順/共同出展は文字下げ表記

3.1.5 出展傾向

EDSFair2007 は、IT 産業の景気回復に伴い、半導体世界市場も回復傾向となり、前回の開催規模(小間数)を約 1%上回った。

	出展者数	小間数
2007 年	154 社	348 小間
2006 年	148 社	343 小間
2005 年	119 社	336 小間
2004 年	105 社	306 小間
2003 年	99 社	320 小間

3.1.6 出展者セミナー

1 セッション 45 分間で、30～100 名の適正人数のお客様に向けて集中 PR が行える出展者セミナールームを提供した。2007 年は 11 会場にて 120 セッションを開催した。

聴講者数：3,042 名

3.1.7 ユニバーシティ・プラザ

産学の交流を促進するとともに、大学研究機関による研究成果を発表する場とする企画である。設計技術に関する研究成果を発表実演した。

17 大学研究室

・IP 間データ転送量に基づく SoC 設計手法
　大阪大学　大学院情報科学研究科　今井 研究室

・次世代 VLSI 設計技術と高性能アーキテクチャに関する技術開発
　大阪大学　大学院情報科学研究科 情報システム工学専攻　尾上・橋本 研究室

・システム LSI の低消費電力テスト・診断技術
　九州工業大学　情報工学部 電子情報工学科　温・梶原 研究室

・LUT カスケードの応用について
　九州工業大学　情報工学部 電子情報工学科　笹尾 勤 教授 研究室

・CMOS 互換不揮発メモリによるアナログトリミング技術
　九州工業大学　マイクロ化総合技術センター　中村 和之 教授 研究室

・システム LIS の低消費電力化技術及び EDA 技術
　九州大学　システム LSI 研究センター　システム LSI 研究センター

・次世代リコンフィギャラブルロジックと FPGA 遠隔再構成技術
　熊本大学　大学院自然科学研究科　情報電気電子工学専攻　末吉 敏則 教授 研究室

・FPGA を用いた生化学シミュレータ
　慶應義塾大学　理工学部　情報工学科　天野 英晴 教授 研究室

・モバイル応用低消費電力システム LSI
　神戸大学　工学部情報知能工学科　吉本 雅彦 教授　研究室

・協調検証環境と FPGA を用いた技術教育プロジェクト
東海大学　情報理工学部　ソフトウェア開発工学科　清水 尚彦 教授 研究室
・東京大学大規模集積システム設計教育研究センター(VDEC)の活動紹介
東京大学　大規模集積システム設計教育研究センター
・システムレベル設計支援技術に関する研究
東京大学　工学系研究科電子工学専攻　藤田 昌宏 教授 研究室
・協調設計技術(システムレベル設計)
名古屋大学　大学院情報科学研究科　高田 広章 教授 研究室
・リコンフィギャラブルシステムの研究開発
広島市立大学　情報科学部　情報工学科　弘中 哲夫 教授 研究室
・メタヒューリスティック手法に基づく LSI フロアプランニング手法
広島市立大学　情報科学部　情報工学科　若林 真一 教授 研究室
・多層プリント配線基板設計支援システム MULTI-PRIDE
広島大学　大学院工学研究科　情報工学専攻　渡邉 敏正 教授 研究室
・FPGA の組込みメモリを活用した回路実現
明治大学　理工学部　情報科学科　井口 幸洋 助教授 研究室

3.1.8 キーノートスピーチ

「ユビキタス情報社会を創るシステム LSI」
慶應義塾大学 理工学部 教授 黒田 忠広 氏
(1) 日　　時：1 月 25 日(木) 10:30～11:30
(2) 場　　所：アネックスホール
(3) 聴 講 料：無料
(4) 聴講者数：343 名

人間生活を支援するユビキタス情報社会の実現には、システム LSI 技術の進展が不可欠である。しかし、ムーアの法則は減速の兆しを見せている。直面する技術課題を解決するには、デバイスから設計、ビジネスまでの総合的なイノベーションが必要であり、広い教養と深い専門を備えた人材が不可欠である。本講演では、ユビキタス情報社会を創るシステム LSI の課題と可能性について、最新の研究成果も紹介しながら将来を展望した。

3.1.9 特別講演

「Post 45nm 時代における EDA の展望と米国における先端研究」
Lanza Tech Ventures Managing Director カーネギーメロン大学客員教授　Dr. Lucio Lanza
(1) 日　　時：1 月 26 日(金) 11:30-12:50
(2) 場　　所：アネックスホール F202
(3) 聴 講 料：無料
(4) 聴講者数：97 名

45nm 以下の微細ノードにおいて問題となるプロセス変動、予測困難な設計パラメータに対処するためには、設計、リソグラフィー、プロセスモデル、キャラクタライゼーション等の総合的な最適化が必要であり、結果として設計と生産のインタフェースに革新的な変化をもたらす。大学教授として日ごろ先端研究に触れ、さらにそれらの事業化を推進するベンチャーキャピタリストの視点から、今後の EDA の展望を語った。

3.1.10 SPIRIT 講演

「IP-XACT 仕様を活用したマルチベンダ・ツールでの IP ベース SoC 設計」

The SPIRIT Consortium Director [Mentor Graphics] Mr. Bill Chown

(1) 日　　時：1 月 25 日(木)　17:15-17:45

(2) 場　　所：展示フロア内特設ステージ

(3) 聴 講 料：無料

(4) 聴講者数：30 名

複雑な SoC 設計のためにはマルチベンダ・ツールによる設計フローが不可欠である。SPIRIT コンソーシアムの IP-XACT(登録商標)は、IP を設計フローへ組み込み、ツール間で情報共有するための標準仕様である。IP-XACT はすでに公開されており広く利用され始めている。本講演では SPIRIT コンソーシアムと IP-XACT 仕様を紹介し、IP-XACT がもたらす設計者への恩恵を説明した。

3.1.11 第 14 回 FPGA/PLD Design Conference

(1) 日　　時：1 月 25 日(木)・26 日(金)

(2) 場　　所：アネックスホール

(3) 聴 講 料：1 セッション受講券(1 回券)　5,250 円(消費税込み)
3 セッション受講券(回数券)　12,600 円(消費税込み)
5 セッション受講券(回数券)　15,750 円(消費税込み)

(4) 聴講者数：セッション 1　45 名　　セッション 6　55 名
セッション 2　32 名　　セッション 7　35 名
セッション 3　51 名　　セッション 8　44 名
セッション 4　37 名　　セッション 9　51 名
セッション 5　42 名　　セッション 10　35 名

FPGA/PLD Design Conference のセッション

第 14 回 FPGA/PLD Design Conference は、FPGA/PLD をテーマとした日本で唯一のコンファレンスであり、FPGA/PLD に関する最新技術、設計事例、ビジネス及び将来動向を包括的に知ることができるよう企画されている。今回は、「利益を産めるシリコン～FPGA が増殖し続ける理由～」をテーマとして掲げ、変化の激しい市場で競争に打ち勝ち、より高い利益を確保するために、FPGA を上手く活用する設計手法、実装技術、将来の技術動向などを紹介した。また、ユーザによるパネルセッションとして「イブニングセッション」、ユーザの設計事例を中心とした「ユーザ・プレゼンテーション」、各種 IP の発表の場を提供する「IP フリーマーケット」については、プレゼンテーション及びパネル展示を行い、普段発表されることが少ない研究技術等が紹介された。

《1 月 25 日(木)》

・セッション 1【入門編】12:50～14:20
「落とし穴に落ちないための FPGA 設計(初級編)」
～問題点を容易に見つけられる設計フローとは～
NEC マイクロシステム㈱　第一 SoC 開発事業部 SoC 第三グループ主任 古川 寛 氏

・セッション 2【解決編】12:50～14:20
「これで良いのかいまどきの FPGA 設計」
アヴネットジャパン㈱　プロダクトマーケティング本部 エンジニアリング部
アプリケーションスーパーバイザー 井倉 将実 氏

・セッション 3【入門編】15:20～16:50
「HDL によるテストベンチ記述のポイント」
㈱エッチ・ディー・ラボ　テクニカル・グループ 安岡 貴志 氏

・セッション 4【将来編】15:20～16:50
「特許出願から見るリコンフィギャラブル・デバイスの世界」
熊本大学大学院 自然科学研究科 情報電気電子工学専攻 教授 末吉 敏則 氏、
㈱キャップインターナショナル 代表取締役 川合 晶宣 氏

《1 月 26 日(金)》

・セッション 5【解決編】10:30-～12:00
「高速高周波に対応したボード設計の勘所」
㈱トッパン NEC サーキットソリューションズ スーパーバイザー 金子 俊之 氏

・セッション 6【解決編】10:30～12:00
「ASIC と比較した FPGA 設計のポイント」
㈱メディアグローバルリンクス ゼネラルマネージャ 中村 和則 氏、
ヒロコン㈱ 九州開発センター 所長 西川 智洋 氏

・セッション 7【解決編】13:20～14:50
「FPGA の高速 I/O を使いこなす」
東京エレクトロンデバイス㈱ PSD 課長代理 鈴木 正人 氏、
東京エレクトロンデバイス㈱
PLD ソリューション部 システムインターフェイス担当 野口 竜洋 氏

・セッション 8【解決編】13:20～14:50
「プロセッサを搭載する FPGA の活用法とデバッグの実際」
㈱アットマークテクノ代表取締役、実吉 智裕 氏

・セッション 9【将来編】15:50～17:20
「知らないと損をする FPGA 消費電力対策」
熊本大学大学院自然科学研究科 情報電気電子工学専攻 教授 末吉 敏則 氏、
ザイリンクス㈱ テクニカルサポート本部 FAE 部 担当課長 中西 郁雄 氏

・セッション 10【解決編】15:50～17:20
「SystemVerilog によるハード/ソフト設計技術者の協調開発」
富士ゼロックス㈱
オフィスプロダクト事業本部 コントローラーハードウェア開発部 宮下 晴 氏

3.1.11.1 FPGA/PLD Design Conference ユーザ・プレゼンテーション

プレゼンテーション

(1) 日　　時：1 月 25 日(木) 15:30～17:35
(2) 場　　所：アネックスホール F201
(3) 聴 講 料：無料
(4) 事前登録：不要
(5) 聴講者数：22 名

ポスター展示

(1) 日　　時：1 月 26 日(金) 10:00～17:00
(2) 場　　所：展示ホール 1F コンコース
(3) 聴 講 料：無料
(4) 事前登録：不要

ユーザ・プレゼンテーションは、FPGA/PLD に関する設計事例や応用事例、及び FPGA/PLD に関する最新の研究や萌芽的アイデアについての意見交換や議論の場として開催された。

さらに、本年度も聴衆による人気投票により優秀プレゼンテーションを選出し表彰する。

3.1.11.2 FPGA/PLD Design Conference IP（Intellectual Property）フリーマーケット

プレゼンテーション

(1) 日　　時：1 月 25 日（木） 11:30～12:55
(2) 場　　所：アネックスホール F201 号室
(3) 聴 講 料：無料
(4) 事前登録：不要
(5) 聴講者数：38 名

ポスター展示

(1) 日　　時：1 月 26 日（金） 10:00～18:00
(2) 場　　所：展示ホール 1 階 コンコース
(3) 聴 講 料：無料
(4) 事前登録：不要

3.1.11.3 FPGA/PLD Design Conference イブニングセッション

(1) 日　　時：1 月 25 日（木） 17:30～19:30
(2) 場　　所：アネックスホール F203 号室
(3) 聴 講 料：無料
(4) 事前登録：不要
(5) 聴講者数：32 名

『利益を産めるシリコン ～FPGA が増殖し続ける理由～』

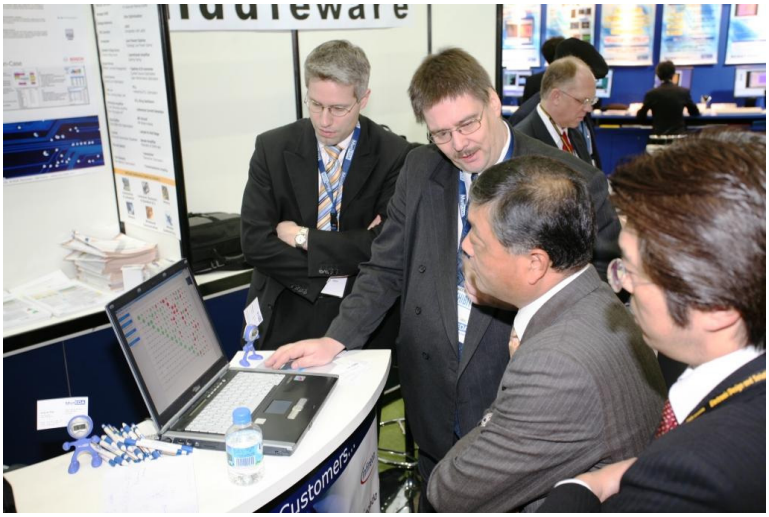
昨年に引き続き、書き換え可能であることを武器に市場で市民権を得た FPGA/PLD。今後益々 FPGA/PLD を活用していく上で、デバイス・設計手法・設計ツールにどんなことが求められるのかを、ユーザ、大学、EDA ツール/デバイスベンダの方々をパネリストに迎え議論した。

司会：三菱電機㈱ コミュニケーション･ネットワーク製作所　松本 仁 氏

パネリスト：㈱アットマークテクノ　実吉 智裕 氏、ザイリンクス㈱　澤田 修 氏、シンプリシティ㈱　藤野 博信 氏、広島市立大学　弘中 哲夫 氏、宮城県産業技術総合センター　小熊 博 氏、メンター・グラフィックス・ジャパン㈱　三橋 明城男 氏

3.1.12 新興ベンダエリア

国内外新興企業 29 社の最新ソリューションが集まり、プレゼンテーションステージや商談コーナを併設した展示ブースで、設計開発者向けに最新情報をいち早く届けた。



新興ベンダエリア内のブース

出展者：

ADVANCED RFIC (S) PTE LTD、ADVINNO TECHNOLOGIES PTE LTD、APRIO TECHNOLOGIES, INC.、AXIOM DESIGN AUTOMATION、AZURO, INC.、BEACH SOLUTIONS INC.、BITROUTER、CEBATECH, INC.、CHIPVISION DESIGN SYSTEMS AG、ENTASYS DESIGN, INC.、HELIC S.A.、LIBRARY TECHNOLOGIES, INC.、MUNEDA GMBH、PYXIS TECHNOLOGY, INC、RIDGETOP GROUP INC.、SIDENSE CORPORATION、SOFTJIN TECHNOLOGIES PRIVATE LIMITED、SOLIDO DESIGN AUTOMATION INC.、TARGET COMPILER TECHNOLOGIES N.V.、TENISON DESIGN AUTOMATION、THE SPIRIT CONSORTIUM INC.、VERIFIC DESIGN AUTOMATION、共信テクノソニック㈱、㈲シンテスト・ジャパン

【JEVeC ビレッジ】 ギガヘルツテクノロジー㈱、ケイレックス・テクノロジー㈱、㈱システム・ジェイディー、㈱数理システム、日本 EDA ベンチャー連絡会(JEVeC)

会場内に設置の特設ステージでは、新興ベンダエリアに参加する国内外新興企業が、次々とプレゼンテーションを行った。

（1 月 25 日（木） 10:30～12:23）

1. ADVANCED RFIC (S) PTE LTD のご紹介
2. ADVINNO TECHNOLOGIES PTE LTD のご紹介
3. APRIO TECHNOLOGIES, INC.のご紹介
4. AXIOM DESIGN AUTOMATION
『Accelerating Functional Verification』
Vice President Sales, Mr. Peter Robinson
5. AZURO, Inc.
『Low Power Clock Implementation』
Vice President of Product Marketing; Marketing department,
Mr. Ashutosh Mauskar
6. BEACH SOLUTIONS INC.のご紹介
7. BITROUTER
『Design Considerations for the U.S. Digital Television Market』
Sales Department, Vice President,
Mr. Douglas Lewis
8. CEBATECH, INC.
『C-to-RTL Compiler Accelerates ASIC/FPGA Design Cycles』
Marketing Department, ESL Product Manager, Mr. John Emmitt
9. CHIPVISION DESIGN SYSTEMS
『Low Power Specialists - Pre-RTL』
VP of Sales and Marketing,
Mr. Tim Herbert

（1 月 25 日（木） 14:20～15:35）

10. ENTASYS DESIGN, INC.
『Power-Centric Design Planning』
President/CEO, Mr. Sung-Hwan Oh
11. LIBRARY TECHNOLOGIES, INC.
『Process variation is here, what do we do about it?』
President, Mr. Mehmet Cirit
12. MUNEDA GMBH
『Design for Manufacturability and Yield with WiCkeD』
Vice President Sales & Marketing, Mr. Andreas Ripp
13. RIDGETOP GROUP INC.のご紹介
14. SOFTJINTECHNOLOGIES PRIVATE LIMITED のご紹介
15. SOLIDO DESIGN AUTOMATION INC.
『Transistor-Level Statistical Design and Verification Technology for Analog/ Mixed-Signal, Custom Digital and Memory Designers』
Engineering, Software Engineer,
Mr. Jeff Dyck

（1 月 26 日（金） 11:40～12:03）

16. TARGET COMPILER TECHNOLOGIES
『Chess/Checkers: a retargetable tool suite for the design of』
Chief Executive Officer,
Mr. Gert Goossens
17. SIDENSE CORPORATION
『128Bit から 8Mbit をサポートする、高速リードアクセスの OTP メモリーIP』
Business Development 大川 和昭 氏

（1 月 26 日（金） 14:30～15:32）

18. ㈲シンテスト・ジャパン
『SynTest 社によるテストコスト低減』
ゼネラルマネージャ 村上 道郎 氏
19. ケイレックス・テクノロジー㈱
『ケイレックス・テクノロジー株式会社 会社紹介』
代表取締役社長 小篠 隆宏 氏
20. ㈱数理システム
『数理システム開発製品のご紹介』
科学技術部 井熊 啓 氏
21. Tenison Design Automation のご紹介
22. VERIFIC DESIGN AUTOMATION
『EDA Component Software』
Spinnaker Systems,
Mr. Yasushi Yamamoto

また、会期前日の 1 月 24 日(水)18 時より、コーウェア㈱代表取締役社長 ジャングットセル 氏を講師とする海外新興ベンダエリア出展者向けのセミナーを開催し、日本進出への足がかりとなる実践的な講演を開催した。

3.1.13 特設ステージ

展示会場内では、特設ステージにて、「今さら聞けないことがわかる！」と題して、若い技術者・設計者向けのセッションを開催した。

場　所：EDSFair2007 会場 特設ステージ

聴講料:無料

特設ステージでのセッション

《1 月 25 日(木)》

3:00～13:10　ごあいさつ「JEITA EDA 技術専門委員会 活動紹介」
電子機器の設計自動化に関わる①国内外の標準化活動、②技術動向調査や課題解決提案を行う研究会活動、③EDSFair を通じた最新 EDA 技術の普及促進活動を行っている社団法人 電子情報技術産業協会(JEITA) EDA 技術専門委員会について紹介した。

13:10～13:15　オープニング ㈱ルネサステクノロジ 秋山 俊恭 氏

13:15～14:00　セッション 1「半導体設計技術最新情報と動向」
㈱半導体理工学研究センター 岡村 芳雄 氏、㈱日立製作所 佐藤 康夫 氏、大阪大学大学院 東野 輝夫 氏、㈳電子情報技術産業協会 EDA 技術専門委員会 委員長 杉山 八六 氏
将来必要となる最新設計技術を紹介。特設ステージのテーマである「今さら聞けないことがわかる！」技術情報が満載であった。若手エンジニア向けに幅広く、最近の話題から数年先のトレンドを見据えた設計技術情報の講演となった。
聴講者数：約 170 名

16:00～17:00　セッション 2「本音で語る動作合成－ここまでできる、ここができない」
日本電気㈱、若林 一敏 氏、メンター・グラフィックス・ジャパン㈱ 小島 智 氏、シャープ㈱ 山田 晃久 氏、㈱半導体理工学研究センター 塩月 八宏 氏、
【司会】シャープ㈱ 西本 猛史 氏
設計サイドと EDA ベンダからパネリストをお招きし、実際に動作合成を使った設計事例を紹介。動作合成を用いたアルゴリズムベース設計の実情と今後の展開の可能性につき、本音の議論を展開した。
聴講者数：約 200 名

《1 月 26 日(金)》

10:15～11:15　セッション 3「誰でも分かる DFM 概論－今さら聞けない DFM がここに」
大日本印刷㈱ 鳴河 照悟 氏、富士通 VLSI㈱ 細野 敏克 氏、
【司会】日経 BP 社 小島 郁太郎 氏
今では日々、DFM という用語を口にし、耳にする。なんとなくは理解しているけれど、DFM の本来の定義って…？　DFM の基礎的概論から、マスクレイアウト、統計的解析まで、その道のスペシャリスト達がわかりやすく解説した。
聴講者数：約 200 名

13:00～14:00　セッション 4 「アナログ難しそう。でも何が難しいの？」
東京工業大学 松澤 昭 教授、
【聞き手】松下電器産業㈱ 大橋 貴子 氏、田口 浩文 氏
知っているようで知らない「今さら聞けない、アナログ設計の世界」を紹介した。先端 SoC におけるアナログ設計の重要性や技術課題、そして設計のツボ(秘訣)を、わかり易く解説した。
聴講者数：約 210 名

16:00～17:00　セッション 5「IC 設計者のための早わかり SiP 設計技術！」
㈱図研 仮屋 和浩 氏、大坪 祐司 氏、日本ケイデンス・デザイン・システムズ社 益子 行雄 氏、アンソフト・ジャパン㈱ 太田 明 氏、柳 孝裕 氏、
【司会】㈱ルネサステクノロジ 秋山 俊恭 氏
SiP 技術は、実装技術ととらえられているが、最適な SiP を開発するためには IC 設計者が全体を理解しておくことが必要である。本セッションでは、IC 設計者に SiP の基本技術、課題を理解する機会を提供した。
聴講者数：約 180 名

3.1.14　来場者数詳細

2007 年来場者数

1 月 25 日(木)	晴れ	4,956 名
1 月 26 日(金)	晴れ	6,180 名
合　計		11,136 名

過去の来場者数

開催年	1 日目	2 日目	合　計
2006 年	5,006 名	5,997 名	11,003 名
2005 年	5,066 名	6,087 名	11,153 名
2004 年	4,764 名	6,023 名	10,787 名
2003 年	5,095 名	6,445 名	11,540 名

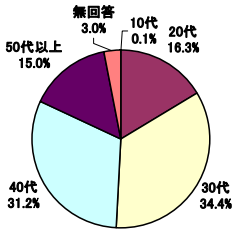
3.1.15　来場者傾向

入場登録票アンケートによる来場者プロファイルを以下に示す。

入場登録票アンケート（属性他）

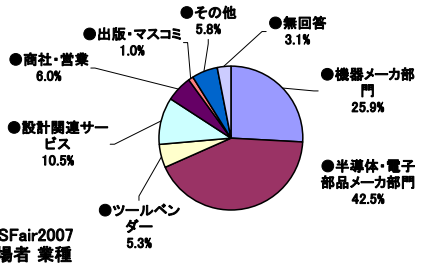
■年代

	2007(今回)	'06(参考)
10代	0.1%	0.1%
20代	16.3%	16.2%
30代	34.4%	32.9%
40代	31.2%	30.6%
50代以上	15.0%	14.4%
無回答	3.0%	5.8%
合計	100.0%	100.0%



■業種

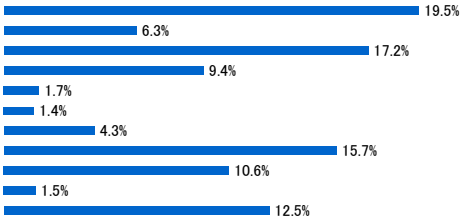
	2007(今回)	'06(参考)
●機器メーカ部門	25.9%	30.2%
●半導体・電子部品メーカ部門	42.5%	37.7%
●ツールベンダー	5.3%	4.6%
●設計関連サービス	10.5%	8.7%
●商社・営業	6.0%	4.6%
●出版・マスコミ	1.0%	1.0%
●その他	5.8%	7.4%
●無回答	3.1%	5.8%



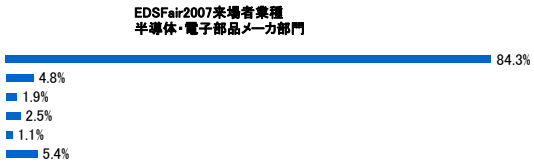
EDSFair2007
来場者 業種

※業種詳細

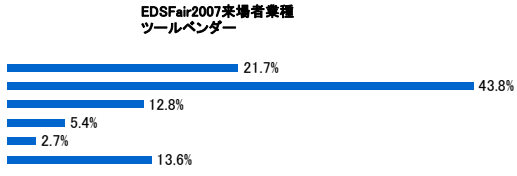
●機器メーカ部門	2007(今回) 25.9%	'06(参考) 30.2%
コンピュータ関連機器	19.5%	25.6%
ネットワーク関連機器	6.3%	6.7%
一般民生機器	17.2%	17.7%
画像処理機器	9.4%	9.3%
医療機器	1.7%	1.7%
アミューズメント	1.4%	0.8%
自動車・輸送機器	4.3%	4.2%
産業機器（機械・精密機器等）	15.7%	13.3%
通信機器	10.6%	7.9%
放送機器	1.5%	1.7%
その他	12.5%	11.1%



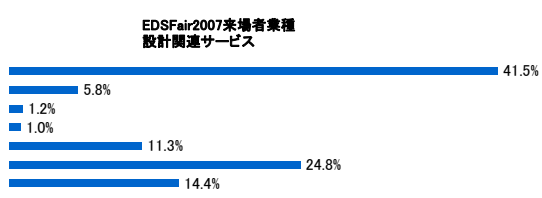
●半導体・電子部品メーカ部門	2007(今回) 42.5%	'06(参考) 37.7%
システムLSI、ASIC、マイコン、メモリ	84.3%	85.4%
FPGA/PLD	4.8%	2.3%
ディスプレイ	1.9%	2.3%
電子コンポーネント	2.5%	3.1%
プリント基板	1.1%	1.1%
その他	5.4%	5.8%



●ツールベンダー	2007(今回) 5.3%	'06(参考) 4.6%
※ツール関連が主要営業品目である商社・代理店も含む		
機能・論理設計ツール	21.7%	21.5%
LSI設計ツール	43.8%	47.1%
プリント基板設計ツール	12.8%	5.8%
マイコンツール	5.4%	6.6%
ハードウェア・ボード機器	2.7%	2.5%
その他	13.6%	16.5%



●設計関連サービス	2007(今回) 10.5%	'06(参考) 8.7%
デザインハウス	41.5%	46.4%
IPプロバイダ	5.8%	6.1%
IP流通サービス	1.2%	0.4%
ネット環境	1.0%	0.4%
教育・コンサルタント	11.3%	11.3%
ソフト開発	24.8%	25.1%
その他	14.4%	10.3%



	2007(今回)	'06(参考)
●商社・営業	6.0%	4.6%

※ツール関連が主要営業品目である商社・代理店は除く

電子機器	14.1%	15.8%
半導体	58.1%	57.3%
電子部品	11.0%	7.5%
ツール	6.2%	4.1%
その他	10.6%	15.3%

EDSFair2007来場者業種
商社・営業

	2007(今回)	'06(参考)
●出版・マスコミ	0.9%	1.0%

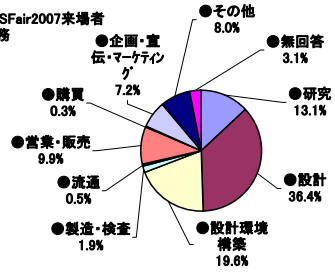
	2007(今回)	'06(参考)
●その他	5.8%	7.4%

	2007(今回)	'06(参考)
●無回答	3.1%	5.8%

■職務

	2007(今回)	'06(参考)
●研究	13.1%	15.1%
●設計	36.4%	42.7%
●設計環境構築	19.6%	15.3%
●製造・検査	1.9%	0.8%
●流通	0.5%	0.2%
●営業・販売	9.9%	7.7%
●購買	0.3%	0.3%
●企画・宣伝・ﾏｰｹﾃｨﾝｸﾞ	7.2%	6.2%
●その他	8.0%	6.0%
●無回答	3.1%	5.7%

EDSFair2007来場者
職務

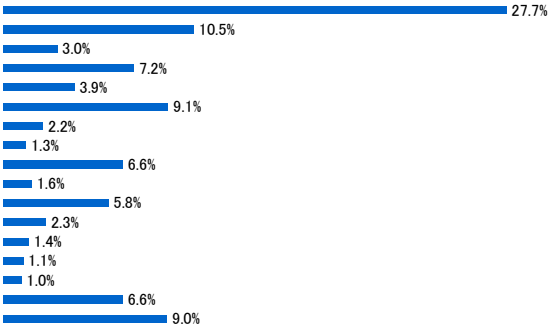


※職務詳細

	2007(今回)	'06(参考)
●研究	13.1%	15.1%

システムレベル	27.7%	–
機能(RTL)	10.5%	–
論理(ゲートレベル)	3.0%	–
レイアウト	7.2%	–
テスト	3.9%	–
アナログ	9.1%	–
カスタム	2.2%	–
IPマクロ	1.3%	–
リソ/マスク/プロセス/製造	6.6%	–
TCAD	1.6%	–
FPGA/PLD	5.8%	–
PCB	2.3%	–
IC Package	1.4%	–
SiP	1.1%	–
装置実装	1.0%	–
ソフトウェア・ファームウェア	6.6%	–
無回答	9.0%	–

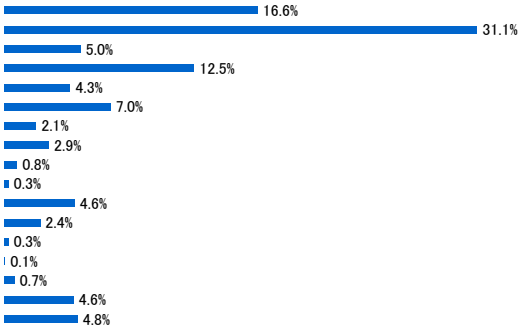
EDSFair2007来場者職務
研究



	2007(今回)	'06(参考)
●設計	36.4%	42.7%

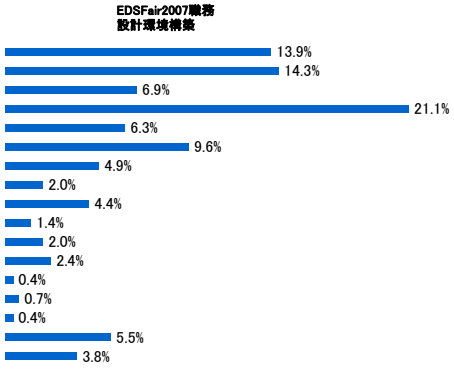
システムレベル	16.6%	27.0%
機能(RTL)	31.1%	20.8%
論理(ゲートレベル)	5.0%	5.9%
レイアウト	12.5%	3.6%
テスト	4.3%	—
アナログ	7.0%	6.7%
カスタム	2.1%	2.8%
IPマクロ	2.9%	2.1%
リソ/マスク/プロセス/製造	0.8%	2.4%
TCAD	0.3%	—
FPGA/PLD	4.6%	4.4%
PCB	2.4%	2.3%
IC Package	0.3%	0.3%
SiP	0.1%	0.1%
装置実装	0.7%	0.3%
ソフトウェア・ファームウェア	4.6%	4.7%
無回答	4.8%	—

EDSFair2007来場者職務
設計



	2007(今回)	'06(参考)
●設計環境構築	19.6%	15.3%

システムレベル	13.9%	—
機能(RTL)	14.3%	—
論理(ゲートレベル)	6.9%	—
レイアウト	21.1%	—
テスト	6.3%	—
アナログ	9.6%	—
カスタム	4.9%	—
IPマクロ	2.0%	—
リソ/マスク/プロセス/製造	4.4%	—
TCAD	1.4%	—
FPGA/PLD	2.0%	5.5%
PCB	2.4%	—
IC Package	0.4%	—
SiP	0.7%	—
装置実装	0.4%	—
ソフトウェア・ファームウェア	5.5%	—
無回答	3.8%	—



	2007(今回)	'06(参考)
●製造・検査	1.9%	0.8%

	2007(今回)	'06(参考)
●流通	0.5%	0.2%

	2007(今回)	'06(参考)
●営業・販売	9.9%	7.7%

	2007(今回)	'06(参考)
●購買	0.3%	0.3%

	2007(今回)	'06(参考)
●企画・宣伝・マーケティング	7.2%	6.2%

	2007(今回)	'06(参考)
●その他	8.0%	6.0%

	2007(今回)	'06(参考)
●無回答	3.1%	5.6%

■ご来場の目的

	2007(今回)	'06(参考)
●展示関連	45.4%	84.1%
●特別展示	21.2%	5.0%
●セミナー／カンファレンス	29.5%	8.9%
●無回答	3.9%	2.0%

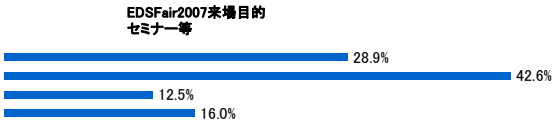
※ご来場の目的詳細

	2007(今回)	'06(参考)
●展示関連	45.4%	84.1%

展示ブース	83.8%	—
スイートデモ	16.2%	—

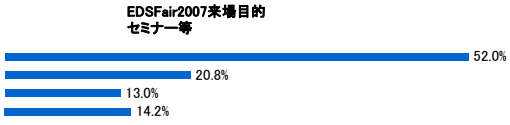
	2007(今回)	'06(参考)
●特別展示	21.2%	5.0%

新興ベンダエリア	28.9%	27.2%
EDSFair特設ステージ	42.6%	40.4%
ユニバーシティ・プラザ	12.5%	16.2%
IPフリーマーケット/ユーザ・プレゼンテーション	16.0%	16.2%



	2007(今回)	'06(参考)
●セミナー／カンファレンス	29.5%	8.9%

出展者セミナー	52.0%	52.7%
キーノートスピーチ	20.8%	21.6%
FPGA/PLD Design Conference	13.0%	14.3%
システム・デザイン・フォーラム	14.2%	11.4%



	2007(今回)	'06(参考)
●無回答	3.9%	2.0%

3.1.16　まとめ

以上述べてきたように、EDSFair2007 の開催を通じ、高度化する電子システム・半導体に関するハードウェア、ソフトウェア設計に関する最新のソリューションを業界一体で展示することにより、①有益な情報の集結と最新動向の情報発信、②重要プレイヤーを集結した業界に向けての情報発信、③有効な産業界コミュニケーションの場の提供、ができたと自負している。今後も EDSFair のより一層の発展に向け、来場者及び出展者双方の満足度向上に向けた各種企画を検討・実現し、開催目的を高いレベルで達成していきたいと考えている。

3.2 システム・デザイン・フォーラム 2007

3.2.1 はじめに

最新の EDA 技術の標準化推進、業界内での普及･促進活動の一環として、EDA 技術専門委員会委員会主催による「システム・デザイン・フォーラム 2007」を、EDSFair2007 と同期して 2007 年 1 月 25～26 日に開催した。

EDA 技術専門委員会は、これまで EDA 標準化活動の発表とその一般への普及を図ることを目的とする“EDA 標準化フォーラム”を 1990 年から 1994 年にかけて 4 回開催し、EDA 専門技術委員会の活動に係る内容の発表、討論の場を目的とする“EDA フォーラム”を 1999 年から 2002 年にかけて 2 回開催してきた。2004 年は、最新の設計技術、課題を設計事例とともに紹介する“システム・デザイン・セミナー”を、2005 年は、“システム・デザイン・フォーラム 2005”として、2 日間の日程で、1 日目に SystemVerilog ユーザ・フォーラムと SystemC ユーザ・フォーラムを、2 日目に SoC に関連した設計技術、課題等を含めた設計事例を紹介する 2 セッションと、LSI、パッケージ、基板を含めた統合設計に関するパネル討論のセッションを開催した。昨年の“システム・デザイン・フォーラム 2006”では、年々増大する設計規模と限られた設計リソースからくる設計クライシス解決の有力手段の 1 つであり、それぞれ 2005 年末に IEEE の標準化がなされた、上流設計記述言語、SystemVerilog と SystemC について、“SystemVerilog ユーザ・フォーラム”と“SystemC ユーザ・フォーラム”の 2 セッション構成で、両設計言語の標準化動向の紹介、チュートリアル、設計適用事例の紹介を行った。

本年の「システム・デザイン・フォーラム 2007」では、フィジカル･デザイン･フォーラムも併設した 3 セッション構成で、2 日間フォーラムを開催した。初日の「フィジカル･デザイン･フォーラム」では、微細化の進展に伴い 65nm 以下のプロセスノードで深刻化するプロセスばらつきを打破する最新の設計技術動向の紹介を、2 日目の「SystemC ユーザ・フォーラム 2007」と「SystemVerilog ユーザ・フォーラム 2007」では、システムレベル設計の設計クライシスを解決する有力手段の 1 つである、上流設計言語の SystemC と SystemVerilog について、その標準化動向の紹介、チュートリアル、及び設計適用事例の発表を行った。

3.2.2 システム・デザイン・フォーラム 2007 概要

・開催日時：1 月 25 日(木) 13:30～17:30 セッション 1　フィジカル・デザイン・フォーラム
1 月 26 日(金) 10:00～12:00 セッション 2　SystemC ユーザ・フォーラム 2007
13:30～15:30 セッション 3　SystemVerilog ユーザ･フォーラム 2007

・開催場所：パシフィコ横浜 アネックスホール　F205、F206

・聴講料金：

	事前申込	当日申込
セッション 1	3,000 円	4,000 円
セッション 2	2,000 円	2,500 円
セッション 3	2,000 円	2,500 円
2､3 セット券	3,000 円	4,000 円

(消費税込み)

・定　　員：200 名/セッション

・主　　催：社団法人 電子情報技術産業協会 EDA 技術専門委員会
・協　　賛：Accellera, OSCI
・受　　付：インターネットで受け付け
http://www.edsfair.com/conference/systemdesign.html
・プログラム

社団法人 電子情報技術産業協会(JEITA) EDA 技術専門委員会は、委員会活動の一環として最新 EDA 技術の業界内への普及促進活動を行っています。本フォーラムは、2 日間で行います。1 日目のフィジカル･デザイン・フォーラムは、65nm 以下の微細化に伴う物理設計の DFM (Design For Manufacturing)問題であるプロセスばらつきを打破する最新の設計技術動向を、2 日目の SystemC ユーザ・フォーラム 2007 と SystemVerilog ユーザ・フォーラム 2007 は、システムレベル設計の設計クライシスを解決する有力手段の 1 つである、上流設計言語の SystemC と SystemVerilog について、その標準化動向の紹介、チュートリアル、設計適用事例の発表を行います。このフォーラムが、システム LSI 設計の最先端の状況把握、さらに議論の場として、お役に立つものと確信いたします。

■セッション1 （1月25日　13:30～15:30、16:00～17:30）
オーガナイザ：JEITA 物理設計標準化研究会 主査　奥村 隆昌（富士通 VLSI）
テーマ：フィジカル・デザイン・フォーラム
概　要：90nm 以降のテクノロジ・ノード、特に直近の 45nm では、WID(Within Die)バラツキの増大が､SoC 設計におけるタイミング設計上の深刻な課題として浮上しています。これに対し、さまざま手法がバラツキに起因する課題を克服する手段として提案されています。本セッションでは、バラツキを考慮した設計手法の現状を以下のトピックを通じてお伝えします。
司　会：増田 弘生 氏(JEITA 物理設計標準化研究会：ルネサステクノロジ)
① バラツキの現状と将来動向　：小野寺 秀俊 氏（京都大学）
② バラツキの計測と解析技術　：中西 甚吾 氏（ルネサステクノロジ）
③ バラツキのモデリング技術　：黒川 敦 氏（三洋半導体）
④ 統計的 STA の実用化技術　：松岡 英俊 氏（富士通）
⑤ バラツキを許容する回路技術：萩原 靖彦 氏（NEC 中央研究所）

■セッション 2 （1 月 26 日　10:00～12:00）
オーガナイザ：JEITA SystemC タスクグループ　逢坂 孝司（日本ケイデンス）
テーマ：SystemC ユーザ・フォーラム 2007
概　要：SystemC は、2005 年 12 月に IEEE において、SystemC の標準 IEEE 1666-2005 が承認され、標準化作業が完了しました。そして現在も C 言語ベースのシステムレベル設計言語の業界標準として、検証、設計分野で幅広く利用されています。本セッション

では、1）OSCI による SystemC の現状とロードマップ、2）JEITA SystemC タスクグループによる SystemC ベースのトランザクション・レベル・モデリングと動作合成に関する取り組みの紹介、3）SystemC を用いた設計事例の発表を行います。

司　会：長谷川 隆 氏（JEITA SystemC タスクグループ 主査：富士通）

① SystemC アップデート　：Patrick Sheridan 氏（OSCI）

② SystemC ベースの TLM と動作合成に関する取り組み：SystemC タスクグループ

③ TLM 標準化の動向について　：武井 勉 氏（半導体理工学研究センター）

④ ソニーにおける動作合成の活用と課題：旦木 秀和 氏（ソニー）

■セッション 3 （1 月 26 日　13:30～15:30）

オーガナイザ：JEITA SystemVerilog タスクグループ　明石 貴昭（日本シノプシス）

テーマ：SystemVerilog ユーザ・フォーラム 2007

概　要：Verilog HDL（IEEE Std. 1364）の次世代言語として、2005 年 11 月に標準化完了した SystemVerilog（IEEE Std. 1800-2005）は、LSI 設計者や検証エンジニアの間で急速に適用が広がっています。本セッションでは、1）Accellera による次の SystemVerilog 改定に向けた取り組みの紹介、2）JEITA SystemVerilog タスクグループによる SystemVerilog テストベンチ・チュートリアルと技術動向紹介、3）日本の SystemVerilog ユーザによる、SystemVerilog 検証事例発表、を行います。

司　会：浜口 加寿美 氏（JEITA SystemVerilog タスクグループ 主査：松下電器産業）

① SystemVerilog 標準化アップデート　：Dennis Brophy 氏（Accellera）

② SystemVerilog テストベンチ言語チュートリアル：SystemVerilog タスクグループ

③ 検証言語としての SystemVerilog 適用事例：鎌田 丈良夫 氏（ルネサステクノロジ）

④ SystemVerilog で構築したアレイプロセッサ検証環境とその効果：清水 圭典 氏（ソナック）

3.2.3 開催結果

各セッションのインターネットでの事前聴講登録数、当時申込数、有料の参加者数、及びアンケート回答数は以下であった。

表 1　システム・デザイン・フォーラム 2007 の参加者数

	セッション 1 （人）	セッション 2 （人）	セッション 3 （人）
事前申込者	134	154	142
当日申込者	10	7	9
申込者数合計	144	161	151
参加者数（有料）	137	148	135
（アンケート回答数）	（114）	（132）	（117）

今年は、フィジカル・デザイン・フォーラムを併設したこともあり、昨年のシステム・デザイン・フォーラム 2006 と比較して、延べ 84 人の参加数増であった。特に、昨年は、一日の開催であったのが、本年は二日の開催とすることにより、EDSFair2007 への集客への貢献も行った。

・各セッションの状況

各セッションとも熱の入った発表、討論となり盛況であった。

■セッション 1

小野寺 秀俊 氏(京都大学)からは、「バラツキ評価 TEG」を用いた観察結果に基づき、0.35μm 世代では、チップ間バラツキが支配的であったものが、0.18μm 世代での、レイアウトに依存したシステマティックなチップ内バラツキの増大を経て、0.13μm、90nm 世代にいたっては、チップ内バラツキが、チップ間ばらつきを上回るに至った様子が示された。さらに、今後のテクノロジ・ノードでは、この傾向が更に深刻化することから、バラツキに対する対応の重要性が訴えられた。

中西 甚吾 氏(ルネサステクノロジ)からは、65nm 世代でのバラツキ解析結果が報告された。報告では、グローバル成分、ランダム成分、他のバラツキ分類のもと、グローバル成分は、ゲート長バラツキが支配的であること、ランダム成分は、1/√LW に比例する Vth のランダムバラツキで説明できることが示された。

黒川 敦 氏(三洋半導体)からは、バラツキのモデリングに際して、チップ内バラツキをシステマティック成分とランダム成分に分離することで、統計理論に基づいた段数効果、サイズ効果が表現可能となる利点が示され、この考えに基づいた SPICE を用いた統計的なシミュレーションの手法が紹介された。通常、SPICE での Tr パラメータの変動と相関を精度よく求めることは難しいため、提案手法では、実測の Vth を基準として、Ion の変動を L で調整する方法を採る。

松岡 英俊 氏(富士通)からは、プロセスノードの進展に伴い、従来の Min-Max コーナ解析ではタイミング制約が悲観的となり過ぎた結果、現実的な解として SSTA が必要とされるに至った経緯の説明に続き、富士通における設計フローへの SSTA 導入事例が報告された。導入に際しては、内製開発によるコスト負担と市販 EDA ツール導入による環境整備、ノウハウの蓄積のコストを考慮した結果、導入当初は、内製ツールにより設計フローを早期に立ち上げるとともに、実測データの蓄積に拠り精度の向上を図り、市販 EDA ツールが整備されるのを待って移行するという段階的、実践的なアプローチを踏んだことが語られた。

萩原 靖彦 氏(NEC 中央研究所)からは、SRAM においては、ランダムばらつきの増大と搭載 Bit 数の増大に伴い、その存在意義(ロジックプロセスとの互換性、ロジック同等の面積スケーリング、速度スケーリング、低消費電力、安定動作、冗長設計による高歩留まり)を同時に満たせなくたってきた現状が語られ、この課題に対する回路的な工夫が語られた。また、65nm 世代までのランダムばらつき増加傾向が 22nm 世代まで続けば CMOS は終焉を迎えるとし、業界あげての対策の必要性を訴えた。

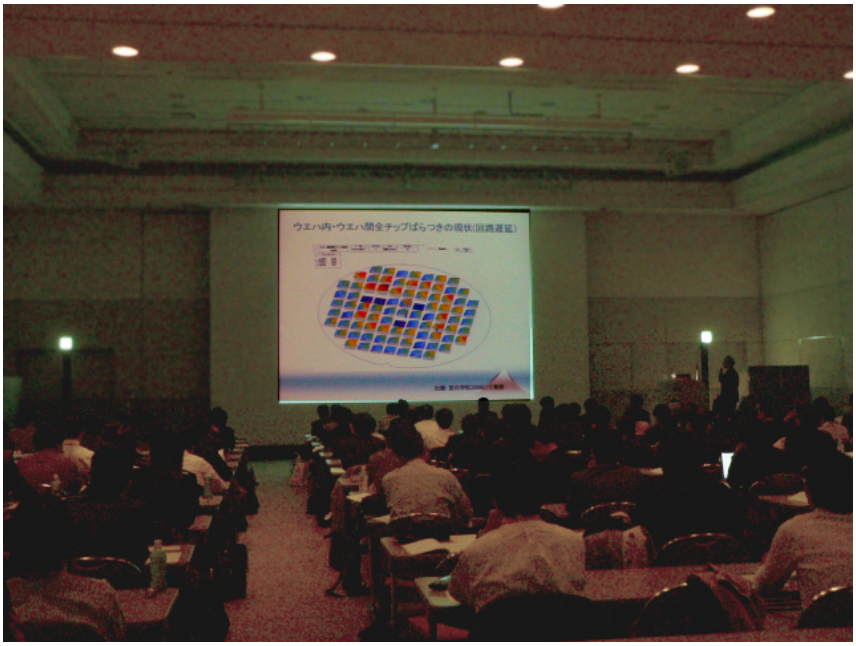


図１　セッション１：フィジカル・デザイン・フォーラム

■セッション 2

昨年同様有料セッションとして開催した本フォーラムであるが、昨年より聴講者はやや減少したものの、事前、当日申し込み者数 161 名、実参加者数 148 名とほぼ目標としていた 160 名に近い人数となった。内容は 2 時間 4 講演を行い、昨年に引き続き OSCI からのアップデートと SystemC タスクグループから研究成果の発表を行った。また、本年はユーザ事例 1 件に加え STARC からの発表も行った。今年はユーザの関心が高い「トランザクション・レベル・モデリング(TLM)」と「合成」を中心とした内容となっており、講演内容の割合としてはこれらテーマが 2 本柱となる構成となった。

最初にタスクグループ主査、長谷川 隆 氏(富士通)の開会挨拶から始まり、まず OSCI からは Patrick Sheridan 氏(コ・ウェア社)を迎えて主に TLM2.0 の開発状況やロードマップについて講演を行った。TLM2.0 はドラフトが OSCI より公開されており、ユーザに向けて今後の開発スケジュールとその概念を発表した。次に SystemC タスクグループより TLM と合成の各サブグループが発表を行い、TLM グループでは国内外における TLM 使用状況の調査を行った結果を発表した。ヨーロッパと日本の TLM 適用範囲の違いが浮き彫りになった。合成グループでは OSCI 合成サブセットのレビュー結果と合成スタイルガイドの作業状況について解説した。既に使われ始めている SystemC ベースの動作合成のユーザに向けて一石を投じる内容であった。STARC からは武井 勉氏を招いて現在の TLM 標準化動向を中心に、TLM 普及と STARC の活動について発表を頂いた。TLM2.0 に合わせてその有用性と TLM にとって重要な標準化活動における各団体の取り組みの紹介は、順調に TLM の業界推進が成されていることを印象付けた。

本年のユーザ事例はソニーの旦木 秀和 氏より、ソニー社内における動作合成の現状とその取り組みを発表頂いた。動作合成ユーザや現在検討しているユーザに向けて今後の課題も含め貴重

な情報を伝えることができた。フォーラム全体を通してほぼ予定通りの時間配分で進み、また、質疑応答時間内も多くの質問がユーザからあがった。

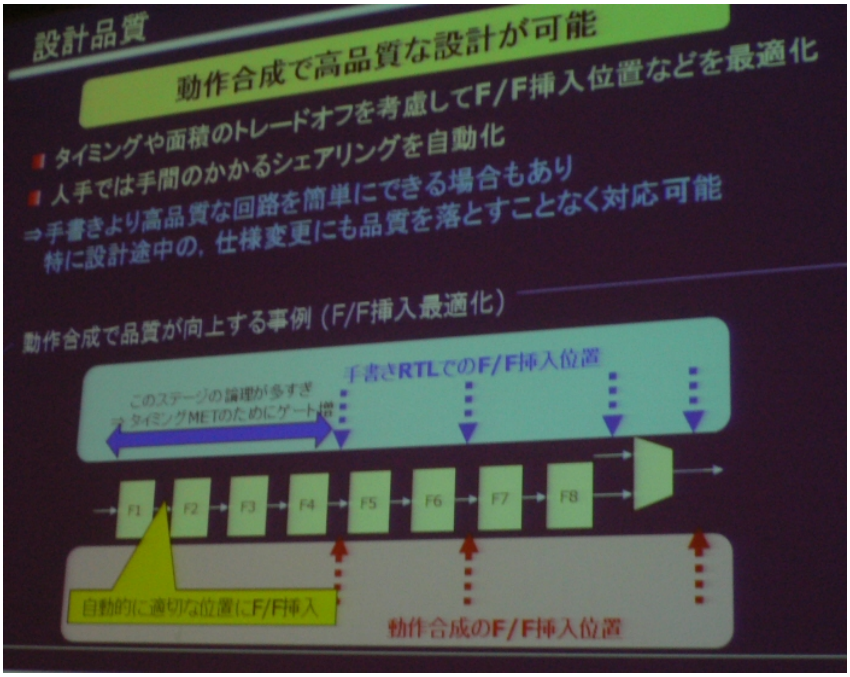


図 2　セッション 2：SystemC ユーザ・フォーラム 2007

■セッション 3

申し込み数 151 名(当日含む)に対して、結果 135 名と昨年と 89%の出席率であった。昨年よりも若干値上げとなったことが要因かは不明だが、結果昨年と比較して 8%と僅かではあるが出席者は減少となった。

プログラムはまず、Accellera Vice Chairman である Dennis Brophy 氏より「The next SystemVerilog Standard」と題した講演で始まった。最新の SystemVerilog 標準化活動が説明され、現在 IEEE std.1364-2005 と IEEE std.1800-2005 の LRM 統合作業が 2008 年標準を目標に進められていることが明らかにされた。

次に JEITA SystemVerilog タスクグループを代表して、李 建道 氏が「SystemVerilog テストベンチ言語チュートリアル」を実施した。前年の「SystemVerilog アサーション」、前々年の「SystemVerilog デザイン」と継続されてきたチュートリアルである。チュートリアルの最後には当タスクグループにて毎年調査、公表している SystemVerilog 構文の EDA ツール対応状況を千綿 幸雄 氏から説明した。

続いてルネサステクノロジの鎌田 丈良夫 氏から、同社がこれまで実施した SystemVerilog 適用事例の中から IP 単体検証環境の改善について講演いただいた。同プロジェクトでは、従来の Verilog HDL を用いた検証環境から SystemVerilog のテストベンチ及びアサーションを用いた環境へ約 1 人月で SystemVerilog を用いた検証環境に移行し、ランダムパラメータが増え、多様なテストシナリオのバリエーション作成効率が可能かつ再利用可能な検証環境を実現できたところ

がメリットとして説明された。

最後のソナック 清水 圭典 氏の講演は、検証の効率化を定量的に見せるという、非常に難しい課題にチャレンジしており、「大変参考になった」との声が多くの参加者から聞かれた。 ソナックでは多彩なコンフィグレーションと複雑な命令の組合せに対応した検証環境を構築するために、SystemVerilog をベースとしたメソドロジである VMM（Verification Methodology Manual）を用い、カバレッジ・ドリブン検証スタイルで行われた。検証の進捗や機能品質の指針として「機能カバレッジ」を使用しており、そのカバレッジ推移を中心にその効果が説明された。この説明は、多くの聴講者の興味を引きつけたようだった。過去 2 回で行われた「RTL 記述への適用」、「海外ユーザの事例」等からさらに踏み込んだ内容となり、SVTB が実際の検証で使われ成果を上げてきていることが明らかとなったフォーラムであった。

図 3　セッション 3：SystemVerilog ユーザ・フォーラム 2007

3.2.4 まとめ

・聴講料の値上げで集客数を心配したが、当日の有料参加者実績で 420 人(3 セッション)と、昨年の 336 人(2 セッション)より増加しており、フィジカル設計分野での“バラツキ問題”、システム設計分野での SystemC と SystemVerilog の両設計記述言語への期待と関心が高い。これは、アンケート結果において、「内容は期待通りでしたか」、「予稿集の内容は期待通りでしたか」、及び「担当業務に役立ちますか」の項目において、3 セッションともに、「満足」及び「やや満足」の合計が約 80%程度に達していることからも伺える。3 セッションとも聴講者の約 80%が、研究、設計、設計環境構築に係わる業務担当であった。

・フィジカル・デザイン・フォーラムの聴講者からは、「デバイスから設計手法まで体系的に学ぶことができた」、「役に立った」等の意見が聞かれ、今後もフィジカル・デザイン・フォーラム開催に対する強い期待が感じられる。

・SystemC、SystemVerilog ユーザ・フォーラムの聴講者中で、SystemC、SystemVerilog での設計・検証環境構築(一部を含む)を既に行っている割合は、各々50%と 20%であり、昨年から着実に増加している。また、その使用目的は、モデリング、テストベンチ・検証、アサーションなどの検証目的が中心であると思われる。

導入検討中、あるいは導入予定なしの理由として、対応ツールが少ないをあげる人が SystemVerilog では多かったが、SystemC では、言語の完成度が不十分とする人が昨年から増加している。

今後の両設計言語の活用のために充実が必要なものとの質問への回答としてあがった、

SystemC : 高位(動作)合成ツール、等価性・プロパティチェックツール、コーディングスタイル・ガイドライン

SystemVerilog : 論理合成ツール、等価性・プロパティチェックツール、デバッグツール・環境、コーディングスタイル・ガイドライン

を中心に、標準化推進団体及び、EDA-TC を通しての Vendor への対応ツール拡充と、コーディングスタイル・ガイドライン作成の働きかけが必要である。

・次年度以降の開催に関して

今後のフォーラムの開催に関し、フィジカル・デザイン・フォーラムでは 70%が積極的に聴講するとのアンケート回答であり、SystemC と SystemVerilog のアンケートでも、セミナー・ワークショップの定期的な開催(40%)と日本語での情報発信(40%)が今後の JEITA への期待としてあがっており、フィジカル・デザイン・フォーラムを含めた本ユーザ・フォーラムの継続的な開催が期待されている。

また、聴講増との観点では、今回も、大学からの参加が少なかったので、参加費用への配慮なども含めて、「学」からの集客増を検討する必要がある。

また、本フォーラムは独立採算で運営している。今年度も単年度では、赤字であった。次回以降、収入増(聴講費、協賛金)への施策が必要である。

3.2.5　システム・デザイン・フォーラム 2007WG 委員

主　　査	山田　　節	三洋電機㈱
委　　員	灘岡　　満	沖電気工業㈱
同	奥村隆昌	富士通 VLSI㈱
同	長谷川　隆	富士通㈱
同	逢坂孝司	日本ケイデンス・デザイン・システムズ社
同	浜口加寿美	松下電器産業㈱
同	明石貴昭	日本シノプシス㈱
同	田口浩文	松下電器産業㈱
同	南　　文裕	㈱東芝
同	今井正治	大阪大学
同	若林一敏	日本電気㈱
アドバイザ	齋藤茂美	ソニー㈱
オブザーバ	杉山八六	富士通㈱
事務局	石崎芳典	日本エレクトロニクスショー協会
事務局	小田佳代子	日本エレクトロニクスショー協会

3.3 ASP-DAC 2007

3.3.1 はじめに

ASP-DAC(Asia and South Pacific Design Automation Conference)は、VLSI 及びシステム LSI の設計技術や設計自動化技術をテーマにしたアジア太平洋地域での最大規模の国際会議である。ASP-DAC は米国で開催されるこの分野のトップ・コンファレンスである DAC(Design Automation Conference)、ICCAD(International Conference on Computer Aided Design)や欧州で開催される DATE(Design, Automation and Test in Europe)とはシスター・コンファレンスの関係にあり、お互いにリエゾンを交換して協力関係を持っている。

ASP-DAC は、電子情報通信学会や情報処理学会などの学会だけでなく、電機メーカ及び半導体メーカの業界団体である JEITA(会議開始当時は EIAJ)と EDSF(会議開始当時は EDAT)の支援のもとで 1995 年に開始された。業界団体である JEITA が ASP-DAC のような国際会議の支援を行っているのは、次のような理由による。電機メーカや半導体メーカが国際競争力のある電子製品の開発を行うためには、マーケッティングや製品企画だけでなく、大規模・高機能・低消費電力のシステム LSI の最適設計を短期間で行える設計力を持つ必要がある。そのためには、最新の設計自動化技術についての情報収集と研究開発を行う必要がある。一流の国際会議を国内で開催することにより、わが国からより多くの技術者と研究者が参加して最先端の設計技術及び設計自動化技術についての情報収集、情報交換などを行うことが可能になる。

3.3.2 会議の開催経緯

ASP-DAC の第 1 回目の会議は 1995 年 8 月 30 日から 9 月 1 日にかけて幕張メッセの日本コンベンションセンターで、情報処理分野の国際学会である IFIP(International Federation on Information Processing)の TC10 WG10.2 及び WG10.5 に属する CHDL 及び VLSI という名称の 2 つの国際会議と並列開催の形で開催された。第 2 回目は 1997 年 1 月に開催され、それ以降毎年 1 月に開催されてきた。この間、1999 年には香港(中国)で、2002 年にはバンガロール(インド)でそれぞれ開催された。1999 年以降は、日本で 2 年間開催したあと国外で 1 回開催するというローテーションで運営されている。今回の会議(ASP-DAC 2007)は 12 回目で、パシフィコ横浜で 1 月 23 日(火)から 26 日(金)の日程で開催された。

3.3.3 ASP-DAC 2007 の概要

ASP-DAC 2007 の概要を表 1 に示す。一般講演としては、30 カ国から投稿された 408 編の論文の中から 131 編が採択され、3 日間にわたって並列の 4 つのトラック、27 のセッションで発表された。表 1 からもわかるように、論文の投稿数については前回日本で開催された ASP-DAC 2005 とほぼ同数の 408 件で、論文の採択率は前年に引き続き 32%となり、この分野での他の国際会議(DAC、ICCAD、DATE)とほぼ同じ水準を維持している。これにより、ASP-DAC は名実ともに、一流の国際会議になったと評価できる。

基調講演のタイトルと講演者を表 2 に、特別セッションのタイトルを表 3 に示す。また、表 4 には、昨年に引き続き実施されたデザイナーズ・フォーラムのセッション・タイトルを示す。表 5 には、有料チュートリアルのタイトルを示す。

発表された論文の中から、表 6 に示す 2 本の論文が選ばれ、ベストペーパー賞が授与された。また、デザイン・コンテストに応募した作品の中から、表 7 に示すように、ベストデザイン 3 件が選ばれて表彰された。

前回まで、Ph.D.フォーラムのセッションを実施していたが、今回は修士の学生も参加できるよう、学生フォーラムと改称した。学生フォーラムは、電子情報通信学会 VLSI 設計技術研究会及び同学会の東京支部であるが、ASP-DAC が財政的なスポンサー団体の 1 つとなっている。

表 1　ASP-DAC 2005、2006、2007 の比較

開催年	2005 年		2006 年	2007 年
日時	2005 年 1 月 18 日(火) ～21 日(金)		2006 年 1 月 24 日(火) ～27 日(金)	2007 年 1 月 23 日(火) ～26 日(金)
会場	上海市(中国) Hotel Equatorial		横浜市(日本) パシフィコ横浜	横浜市(日本) パシフィコ横浜
併設展示会	小規模な展示会(12 社)		EDSF 2006	EDSF 2007
論文投稿数	692		424	408
論文投稿国 (地域)数	32		27	30
論文採択数 (採択率)	Regular	99(14.3%)	135(32%)	131(32%)
	Short	86(12.4%)		
	Poster	95(13.7%)		
	合計	280(40.5%)		
キーノート アドレス	3 件		3 件	3 件 (表 2 参照)
一般講演	35 セッション(185 編)		27 セッション(135 編)	27 セッション(131 編)
特別セッション (招待講演等)	内部チュートリアル	3 セッション	5 セッション	5 セッション (表 3 参照)
	パネル討論	3 セッション		
	招待講演	1 セッション		
デザイン・ コンテスト	1 セッション		1 セッション	1 セッション
学生フォーラム	1 セッション (Ph.D フォーラム)		昼休みに実施 (Ph.D フォーラム)	昼休みに実施 (Student Forum と改称)
ポスターボード	4 セッション		―	―
有料 チュートリアル	6 件 (全日 2 件、半日 4 件)		6 件 (全日 2 件、半日 4 件)	6 件(表 5 参照) (全日 2 件、半日 4 件)
デザイナーズ・ フォーラム	―		4 セッション (招待講演 2、 パネル討論 2)	4 セッション(表 4 参照) (招待講演 2、 パネル討論 2)

表 2　基調講演

講演タイトル	講演者
Next-Generation Design and EDA Challenges: Small Physics, Big Systems, and Tall Tool-Chains	Rob Rutenbar (Carnegie Mellon University, USA)
Meeting with the Forthcoming IC Design - The Era of Power, Variability and NRE Explosion and a Bit of the Future	Takayasu Sakurai (University of Tokyo, Japan)
How Foundry can Help Improve your Bottom-Line? Accuracy Matters!	Fu-Chieh Hsu (TSMC, Taiwan)

表 3　特別セッションのタイトル

種類	セッション・タイトル
プレゼンテーション＋ポスターボード	セッション 1D：University Design Contest
招待講演	セッション 2D：Design for Manufacturability
	セッション 3D：Embedded Software for Multiprocessor Systems-on-Chip
	セッション 4D：EDA Challenges for Analog/RF
	セッション 7D：Multi-Processor Platforms for Next Generation Embedded Systems

表 4　デザイナーズ・フォーラムのタイトル

種類	セッション・タイトル
招待講演	セッション 6D：Low-power SoC Technologies
	セッション 8D：High-speed Chip to Chip Signaling Solutions
パネル討論	セッション 5D：Presilicon SoC HW/SW Verification
	セッショ:ン 9D：Top 10 Design Issues

表 5　Tutorial のタイトル

トラック	種類	タイトル
1	全日	DFM Tools and Methodologies at 65nm and Beyond
2	全日	Functional Verification Planning and Management - The Road to Verification Closure is Paved with Good Intentions
3	半日	Low Power CMOS Design: The Fabrics: Research Front-End
4	半日	Low Power CMOS Design: The Applications: State-of-the-Art Practice
5	半日	Fast Physical Synthesis for Multi-Million Gate ASIC Designs
6	半日	Concepts and Tools for Practical Embedded System Design

表 6　ベストペーパー賞が授与された論文

論文タイトル・著者
3B-2: “Protocol Transducer Synthesis using Divide and Conquer Approach”, Shota Watanabe, Kenshu Seto, Yuji Ishikawa, Satoshi Komatsu, Masahiro Fujita (Univ. of Tokyo, Japan)
5A-1: “A New Methodology for Interconnect Parasitics Extraction Considering Photo-Lithography Effects”, Ying Zhou (Texas A&M Univ., United States), Zhuo Li (Pextra Corp., United States), Yuxin Tian, Weiping Shi (Texas A&M Univ., United States), Frank Liu (IBM, United States)

表 7　ベストデザイン賞が授与された設計

種類	論文タイトル・著者
Outstanding Design Award	1D-1: "A 1Tb/s 3W Inductive-Coupling Transceiver Chip", Noriyuki Miura, Tadahiro Kuroda (Keio University)
Special Feature Award	1D-9: "Improving Execution Speed of FPGA using Dynamically Reconfigurable Technique", Roel Pantonial, Md. Ashfaquzzaman Khan, Naoto Miyamoto, Koji Kotani, Shigetoshi Sugawa, Tadahiro Ohmi (Tohoku University)
	1D-13: "Implementation of a Standby-Power-Free CAM Based on Complementary Ferroelectric-Capacitor Logic", Shoun Matsunaga, Takahiro Hanyu, Hiromitsu Kimura, Takashi Nakamura, Hidemi Takasu (Tohoku Univewrsity, ROHM)

3.3.4　論文の投稿状況

1997 年から 2007 年の、ASP-DAC への論文投稿数の地域別の推移を図 1 に示す。ASP-DAC 2002 はインドのバンガロールで同時開催された VLSI Design 2002 への投稿論文を含んでいるため、その前後の年よりも投稿数が多い。図 1 に示すように、1999 年(香港開催)以降は投稿論文数が毎年増加傾向にある。海外からの投稿も多くなってきており、名実ともに設計自動化の分野の国際会議として定着したといってよいであろう。

表 8 に、日本からの論文投稿数の推移と全世界から投稿された論文に占める割合を示す。日本からの論文投稿数は、徐々に増えつつあるが、全体に占める割合の点では、2000 年をピークにして、10%台に低下している。今回論文投稿数が多かったのは、米国の 129 編(前回は 155 編)、中国の 61 編(前回は 57 編)、日本の 44 編(前回は 51 編)、台湾の 40 編(前回は 42 編)、韓国の 23 編(前回は 21 編)であった。欧州からは、英国の 11 編をはじめとして、40 編(前回は 47 編)の投稿があった。

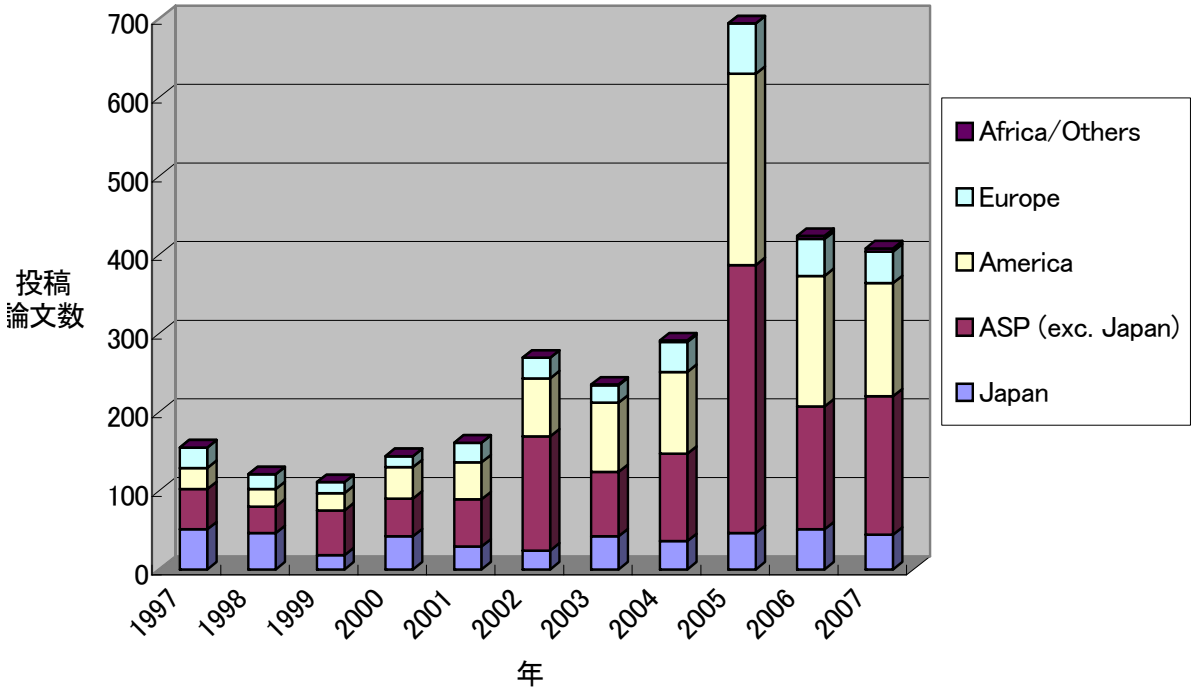


図 1　地域別論文投稿数

表 8　日本からの論文投稿数と全体に占める割合

年 / 地域	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007
日本 (割合)	51 (33%)	46 (38%)	18 (16%)	42 (29%)	29 (18%)	24 (9%)	42 (18%)	36 (12%)	46 (7%)	51 (12%)	44 (11%)
全体	155	121	111	144	161	269	235	291	692	424	408

次に、研究分野別の論文投稿数及び採択論文数を表 9 に示す。ASP-DAC 2007 では、ASP-DAC 2004～2006 と同様に、研究分野を 11 種類に分類して論文の査読と採否の決定を行った。今回論文投稿数が多かった分野は、分野 7 のタイミング・消費電力関係のセッション、分野 1 のシステムレベル設計のセッション、分野 2 の組込み及び実時間システムのセッション、分野 3 の動作・論理合成及び最適化のセッションであった。

表 9　分野別の論文投稿数と採択論文数

分野	研究分野	投稿数	採択数	採択率
1	System Level Design	54	15	28.8%
2	Embedded and Real-Time Systems	45	12	26.7%
3	Behavioral/Logic Synthesis and Optimization	45	14	31.1%
4	Validation/Verification for Behavioral/Logic Design	28	9	32.1%
5	Physical Design (Routing)	23	10	35.7%
6	Physical Design (Placement)	31	11	35.5%
7	Timing, Power, Signal/Power Integrity Analysis and Optimization	63	17	32.1%
8	Interconnect, Device and Circuit Modeling and Optimization	39	13	33.3%
9	Test and Design for Testability	32	10	31.3%
10	Analog, RF and Mixed Signal Design and CAD	22	8	38.1%
11	Leading Edge Design Methodology for SoC's and SiP's	31	12	35.3%
	合　　計	408	131	32.1%

3.3.5　参加者の内訳

ASP-DAC への地域別の参加者数の推移を図 2 に示す。また、日本からの参加者の推移を表 10 に示す。今回の参加者数は 685 名であった。前回と比べると、参加者が若干減少している。日本からの参加者数は全体の 66%の 450 名であった。

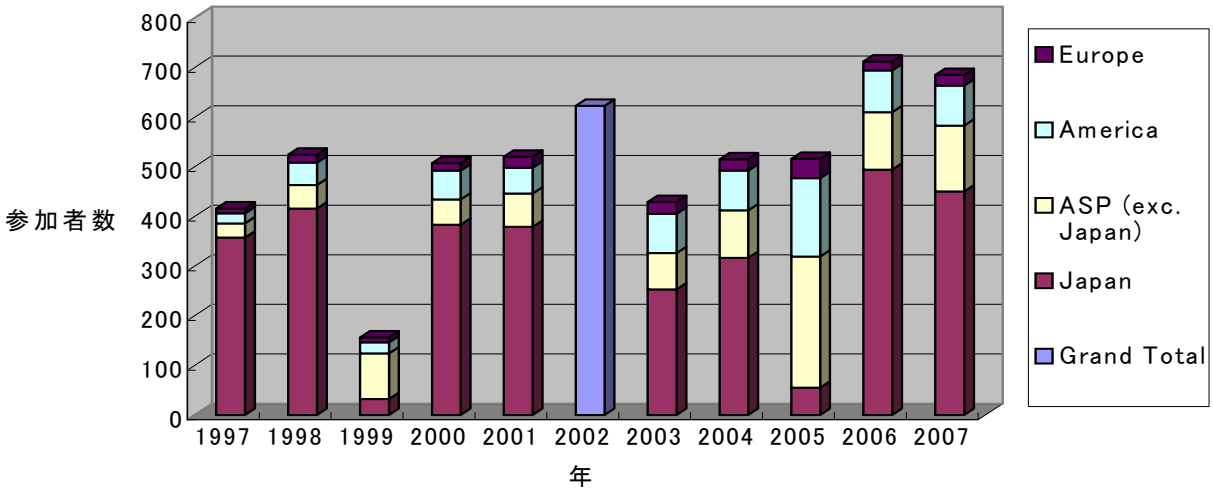


図 2　地域別参加者の推移

表 10　日本からの参加者数と全体に占める割合（チュートリアルのみの参加者を除く）

年 / 地域	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007
日本 （割合）	357 （86%）	416 （79%）	32 （21%）	383 （76%）	379 （73%）	N/A	253 （59%）	316 （61%）	55 （11%）	494 （70%）	450 （66%）
全体	416	524	156	507	520	623	429	515	516	708	685

3.3.6　今後の展望

ASP-DAC の今後の開催予定を表 11 に示す。これまで、3 年に 2 回の割合で日本において開催していたが、今後は、隔年で日本開催になる模様である。

表 11　ASP-DAC の今後の開催予定

年	開催予定地	開催時期	実行委員長
2008 年	ソウル市（韓国）	2008 年 1 月 21 日（月）～24 日（木）	Prof. Kyung（KAIST）
2009 年	パシフィコ横浜（日本）	2009 年 1 月 19 日（火）～22 日（金）	若林 一敏 氏（NEC）

4.　添付資料

4.1.1 寄生効果モデリングタスクグループ

中林　太美世（シャープ）
門脇　匡志　（松下）
多久島　純之（ソニーLSIデザイン）
室本　栄　　（日本ケイデンス）
黒川　敦　　（三洋半導体）
増田　弘生　（ルネサステクノロジ）
佐藤　高史　（東京工業大学）*
橋本　昌宜　（大阪大学）*
金本　俊機　（ルネサステクノロジ）**

*）　客員
**）オブザーバ

JEITA Physical Design Standardization Study Group　1

報告内容

1. 活動目的
2. 活動内容
3. 温度依存抵抗変動の回路特性へのインパクト
4. 遅延ばらつき特性の環境温度依存性
5. LPEツールのベンチマーク結果（2005年度活動）
6. まとめと今後の課題

JEITA Physical Design Standardization Study Group　2

1. 活動目的

- 温度依存抵抗ばらつきがもたらす回路特性への影響調査
- 遅延ばらつき特性の環境温度依存性調査
- 標準ベンチマークデータを用いたLPEツールの精度評価

➡ 配線ばらつき考慮の重要性

2. 2006年度の活動内容

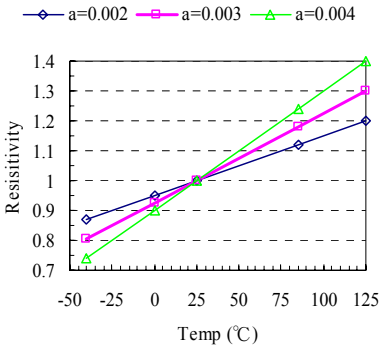
- 温度依存抵抗変動による回路特性へのインパクト調査
- 遅延ばらつき特性の環境温度依存性評価
- 市販されているLPEツールのベンチマーク結果まとめ

3.1 温度依存抵抗変動による回路特性へのインパクト

配線抵抗の温度依存性

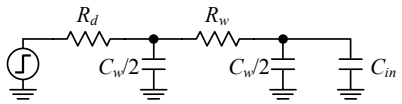
$$R = R_0\left(1 + a\left(T - T_0\right)\right)$$

Cuの温度係数≒0.003



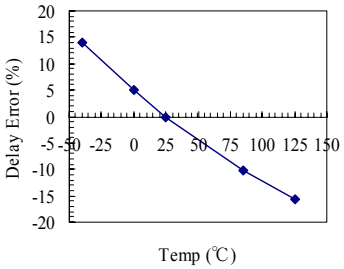
図PEM-1(a) 配線抵抗の温度依存性

遅延への影響



$@25°C:$

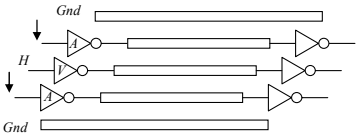
$R_d = 237(\Omega),\quad R_w = 684(\Omega),\quad C_w = 212(fF),\quad C_{in} = 32(fF)$



図PEM-1(b) 温度依存抵抗変動の遅延への影響

[1] T. Sakurai, Trans. ED, 1983.

3.2 クロストーク・ノイズへの影響



1mm並行配線(Near=Far=500um)

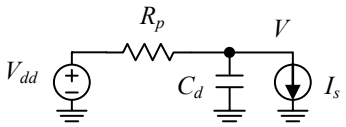
Condition	Victim Temp (°C)			Aggressor Temp (°C)			Vpeak (V)	Ratio
	Driver	Near	Far	Driver	Near	Far		
Same V=A	-40	-40	-40	-40	-40	-40	0.200	1.00
	25	25	25	25	25	25	0.200	1.00
	125	125	125	125	125	125	0.200	1.00
	125	25	25	125	25	25	0.201	1.00
V != A	25	25	25	85	85	85	0.183	0.91
	85	85	85	25	25	25	0.218	1.09
Near != Far	25	25	25	85	85	25	0.183	0.91
	25	25	25	85	25	85	0.191	0.95
	85	85	25	25	25	25	0.218	1.09
	85	25	85	25	25	25	0.210	1.05

図PEM-2 温度依存抵抗変動のクロストークのイズへの影響

- **VictimとAggressorの温度に差がなければ、ピーク電圧はほぼ同じ**
- **60℃の温度差があると仮定した場合、ピーク電圧に約9%の差が生じる**
- **配線の近端と遠端で60℃の温度差があると仮定した場合、約4%の差が生じる**

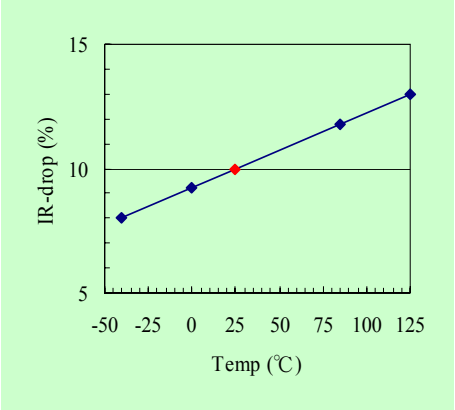
[2] M.Hashimoto, ISPD, 2002.

3.3 IR-dropへの影響



- 簡易モデルの計算で、
  温度が60℃異なると、
  IR-dropは約20%変動

25℃でIR-drop=10%を仮定した特性



図PEM-3 温度依存抵抗変動のIR-dropへの影響

3.4 インパクトのまとめ

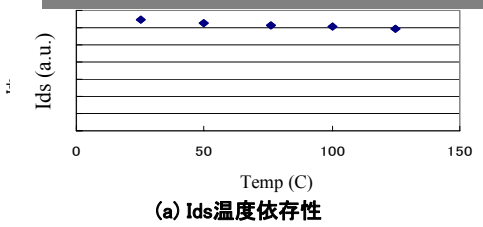
- 報告されているチップ内温度差は、～60℃
- 配線抵抗は、+60℃で約20%増加
- 遅延時間はロング配線で、～約13%程度変動
- クロストークによるピーク・ノイズは、～約10%変動
- IR-dropは、～約20%変動

4. 遅延ばらつき特性の環境温度依存性評価

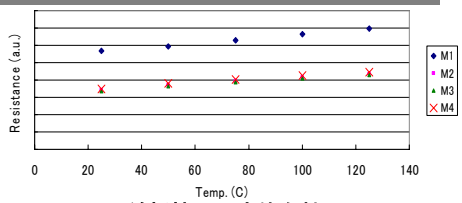
一内容

- LSI回路構成素子の特性およびばらつきの環境温度依存性 実測結果
- 回路遅延ばらつきの解析（モンテカルロ）
  - セル遅延、パス遅延
  - Trばらつき/配線抵抗ばらつきの成分分離
- Cu配線抵抗ばらつきのメカニズム評価
- 45nmプロセスでの回路遅延ばらつき予測
- まとめ

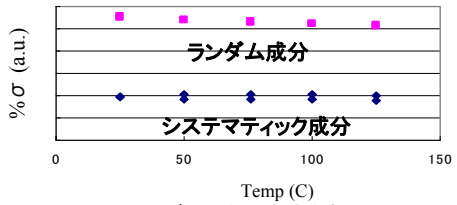
4.1 素子ばらつきの温度依存性



(a) Ids温度依存性

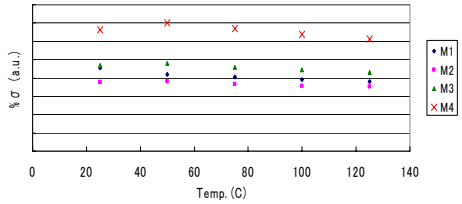


(a)配線抵抗の温度依存性



(b) Idsばらつき温度依存性

図PEM-4 NMOSドレイン電流およびばらつきの温度依存性



(b) 配線抵抗のランダムばらつきの温度依存性

図PEM-5 Cu配線抵抗およびばらつきの温度依存性

Idsの温度依存性：高温でIds小
σ(Ids)の温度依存性：低温でばらつき大
Idsばらつきは低温でWorst

配線抵抗Rの温度依存性：高温でR大
σ(R)の温度依存性：低温でRばらつき大
Rばらつきは低温でWorst

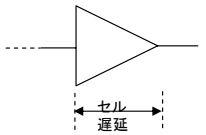
[12] STARC, 90nm TEG

4.2チップ内配線遅延ばらつきの温度依存性 (1)

●基本デジタル回路特性のばらつき評価
(1)無負荷のセル遅延評価
(2)配線RCを含むパス遅延評価

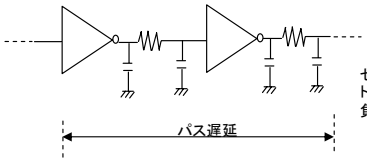
(1) Tech Node=65nm
NMOS: L=60nm, W=300nm, Vthn0=0.4V,
σ(Vthn0)=30mV
PMOS: L=60nm, W=500nm, Vthp0=-0.4V,
σ(Vthp0)=25mV
Vdd=1.0V
SPICE パラメータ=Typical
(2) Simulation Method
Monte Carlo (2000 Runs)
Monte Carlo Parameters:
Vtn0 & Vtp0 (random) of all MOS Transistors
Ln & Lp (systematic) of all MOS Transistors
Wire Resistance (random & systematic)
(3) Simulator:
Synopsys 社製HSPICE

図PEM-6 解析に使用したパラメータ



セル種類:バッファセル
(インバータ2段構成)
トランジスタサイズ:x1

(1)無負荷のセル遅延

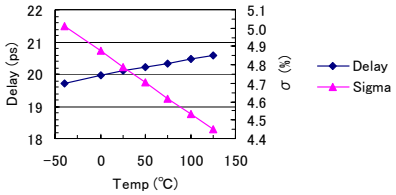


セル種類:インバータセル
トランジスタサイズ:x64
負荷配線長:1mm

(2)配線RCを含むパス遅延

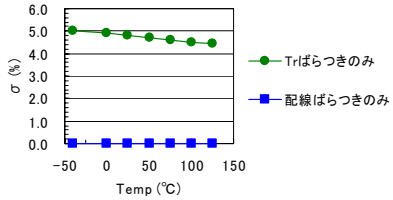
図PEM-7 評価に用いた回路

4.2チップ内配線遅延ばらつきの温度依存性 (2)
無負荷のセル遅延ばらつき解析結果



(a) 無負荷のセル遅延の平均値と相対ばらつき

セル遅延:高温で遅延大
セル遅延ばらつき:低温でばらつき大

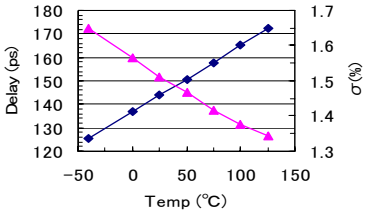


(b) 無負荷のセル遅延の相対ばらつきの成分分離

Trばらつきの効果が支配的
配線ばらつきの効果は無視できる

図PEM-8 無負荷のセル遅延ばらつき

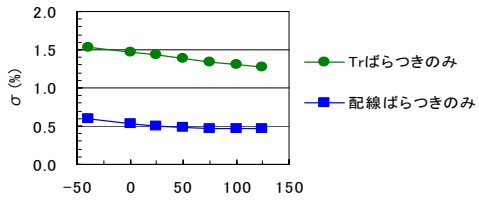
4.2チップ内配線遅延ばらつきの温度依存性 (3)
配線RCを含むパス遅延ばらつき解析結果



(a) 配線RCを含むパス遅延の平均値と相対ばらつき

パス遅延:高温で遅延大

パス遅延ばらつき:低温でばらつき大。セル遅延と比較して大きなトランジスタを使うのでばらつきの絶対値は小さい。



(b) 配線RCを含むパス遅延の相対ばらつきの成分分離

パス遅延のばらつき:Tr.ばらつきの方が大きいが、配線特性ばらつきの影響も顕著になる

図PEM-9 配線RCを含むパス遅延ばらつき

4.3 45nmプロセスでの回路ばらつきの予測 (1)

- 45nmプロセスでの予測:
  - Trばらつき、配線ばらつきが共に増大

  ⇒
  - 素子特性ばらつきの予測
  - 回路特性ばらつきの予測

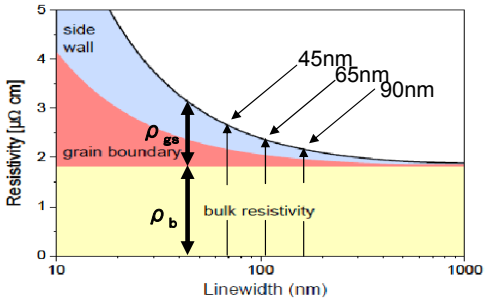
4.3 45nmプロセスでの回路ばらつきの予測 (2)
解析条件(各技術ノード)

	L(um)	W(um)	Vth0(V)	Sigma Vth(mV)	Sigma %σ :配線抵抗ばらつき
90nm(N)	0.1	0.5	0.4	25	0.75
65nm(N)	0.06	0.3	0.4	30	1.05
45nm(N)	0.032	0.2	0.4	35	1.46
90nm(P)	0.1	0.8	−0.4	20	0.75
65nm(P)	0.06	0.5	−0.4	25	1.05
45nm(P)	0.032	0.3	−0.4	30	1.46

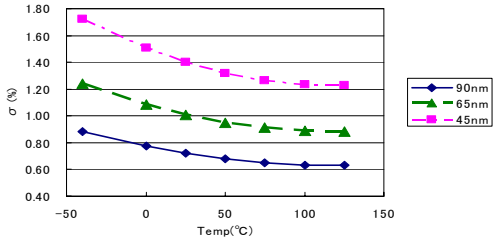
図PEM−10 45nm−90nmプロセスでの解析条件一覧

[7] ITRS2006
[10] W. Steinhoegl, SISPAD 2003
[11] Semiconductor International;

4.3 45nmプロセスでの回路ばらつきの予測 (3)
Cu配線ばらつきの推定



図PEM−11 配線抵抗率の配線幅依存性



図PEM−12 45nm−90nmプロセスCu配線抵抗ばらつきの温度依存性計算結果

Cu配線ばらつき推定方法

$$\rho = \rho_{gs} + \rho_b$$

$$\frac{\delta\rho}{\rho} = \frac{\delta\rho_{gs} + \delta\rho_b}{\rho} \approx \frac{\delta\rho_{gs}}{\rho}$$

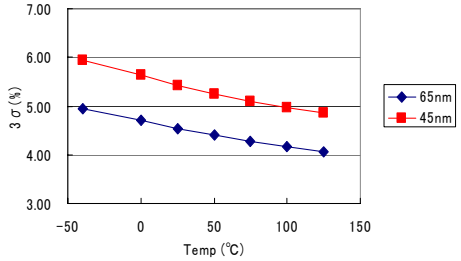
	配線幅*	%ρgs	%σ(R)
90nm	0.15um	17%	0.75% (Exp.)
65nm	0.11um	24%	1.05%
45nm	0.07um	33%	1.46%

*)ITRS2006; http://www.itrs.net/

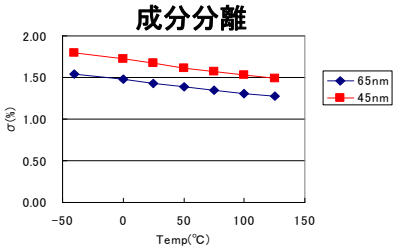
[10] W. Steinhoegl, SISPAD 2003
[11] Semiconductor International;

4.3 45nmプロセスでの回路ばらつきの予測 (4)
45nm/65nmパス遅延ばらつき解析結果

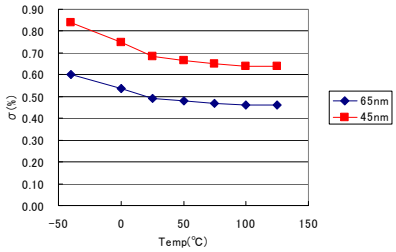
全体



成分分離



(a) Tr.抵抗ばらつき成分



(b) 配線ばらつき成分

図PEM-13 45nmプロセスのパス遅延ばらつき予測:

●45nmプロセスでは低温(−40℃)における遅延ばらつき(3σ)が65nmに比べて約1%増大し、6.0%となる
●45nmプロセスでは配線抵抗ばらつきの影響が増大し、遅延ばらつき全体に対する寄与率が32%となる

4.4 まとめ

- 遅延ばらつき特性の環境温度依存性に関して、はじめて定量的な評価を行った
- 成果内容
  - Cu配線抵抗のばらつきが低温で増大することを指摘し、その物理的メカニズムがCu Grain特性に拠ることを示した
  - Trと配線抵抗ばらつきのセル遅延・パス遅延に与える影響を、モンテカルロ回路解析で定量的に示した
  - 45nmプロセスにおいて、パス遅延ばらつきに対する配線抵抗ばらつきの寄与が32%になることを予測した
- 結論
  - 今後のLSI設計において、低温でのばらつきマージンの考察、MOS電流ばらつきだけでなく配線抵抗ばらつきの考慮が必須になってくる

参考文献 [1]

[1] T. Sakurai, “Closed-form expressions for interconnection delay, coupling, and crosstalk in VLSI’s,” IEEE Trans. Electron Devices, vol.40, no.1, pp.118-124, Jan. 1983.
[2] M. Hashimoto, M. Takahashi, and H. Onodera, “Crosstalk noise optimization by post-layout transistor sizing,” Proc. ISPD, pp.126-130, 2002.
[3]S. B. Samaan, “The impact of device parameter variations on the frequency and performance of microprocessor circuits,” ISSCC 2004, Microprocessor Circuit Design Forum Digest, p.29, Feb. 2004.
[4] H. Masuda et al., “Challenge: variability characterization and modeling for 65- to 90-nm processes,” CICC 2005, pp.593-595, Sep. 2005.
[5] M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers, “Matching properties of MOS transistors,” IEEE J. Solid-State Circuits, vol. 24, no. 5, pp.1433-1440, Oct. 1989.
[6] 増田弘生, “MOS トランジスタの特性ばらつき,” VLSI 夏の学校, 8月, 2006年
[7] ITRS2006; http://www.itrs.net/
[8] S. Ohkawa, M. Aoki, and H. Masuda, “Analysis and characterization of device variations in an LSI chip using an integrated device matrix array,” IEEE Trans. Semiconduct. Manufact., vol. 17, no. 2, pp. 155-165, May 2004
[9] M. Aoki, S. Ohkawa, and H. Masuda, “Design guidelines and process quality improvement for treatment of device variations in an LSI chip,” IEICE Trans. Electronics, vol. E88C, no. 5, pp.788-795, May 2005.
[10] W. Steinhoegl et al., “Scaling laws for the resistivity increase of sub-100 nm interconnects,” SISPAD 2003, pp.27-30, Sept. 2003.
[11] Semiconductor International; http://www.sijapan.com/content/0507vol2/cover/cover_0507.html
[12]半導体理工学研究センター,90nm TEG データ

5. Benchmark results of LPE tools

Purposes of Activity

- Evaluation of accuracy of LPE (Layout Parasitic / Parameter Extraction) tools
- Clarification of accuracy evaluation criteria required in transistor-level parasitic effect modeling
- Development of a set of standard benchmark data

Contents

- We studied Tr. Level LPE tools
  - Survey of LPE tools' functions
  - Investigation and evaluation of the specification for 2D/3D basic benchmark data
  - Investigation and evaluation of specification for the parasitic extraction of contacts and vias.

Background

- With the scaling of process technology, LPE accuracy is becoming increasingly important.
- Interconnect parasitic devices have a big influence on timing, power supply, reliability, noise, etc. in LSI.
- In particular extraction accuracy for contact/via capacitance and non-Manhattan patterns is required for recent technologies.

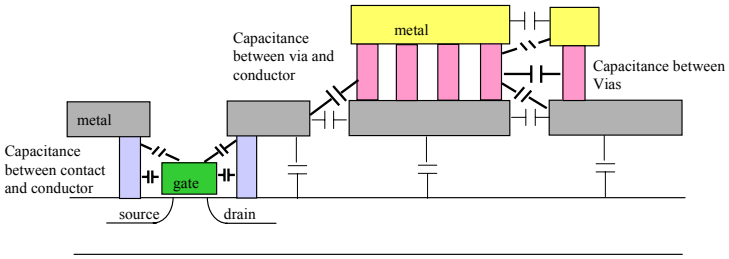


Fig. PEM-E1 Parasitic capacitances of contact/via which become important in next generation process nodes.

Survey of LPE Tools' Functions

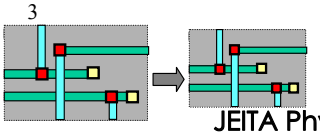
- Target tools:
  - Synopsys Star-RCXT
  - Mentor Graphics Calibre-xRC
  - Sequence Design Columbus-AMS
  - Cadence Assura-RCX
- Outline:
  - Modeling of the interconnect cross-section and dependence of process
  - Extraction
  - Output format and back-annotation

Survey of LPE Tools' Function (1)

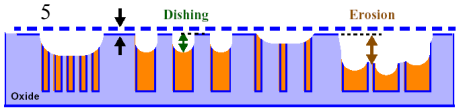
(A) Process Dependence

	Functional Specification	Results
1	For planar process, is it possible to define the non-linear quantity of etching of wire width which is dependent on wire spacing in order to achieve accurate C extraction ?	4/4
2	For planar process, is it possible to define the non-linear quantity of etching of wire width which is dependent on wire spacing in order to achieve accurate R extraction ?	4/4
3	Is process-shift possible ? Is it possible to define the layout shrink ratio ?	3/4
4	As for the cross-sectional shape of wire, is it possible to define it as a trapezoid (trapezium)?	4/4
5	As for the cross-sectional shape of multiple parallel wires, is it possible to define it as a dishing shape ?	2/4
6	Is it possible to define a Void between wires on a same layer ? (Void, also called Air-Gap, is a area which arises in dielectric layer with different permittivity.) How to define the width and height Void ?	4/4
7	Is it possible to define a non-planer wiring structure ?	4/4
8	Is it possible to define a conformal conductor ?	4/4

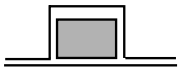
3



5



8



Investigation of LPE Tool's Function (2)

(B) Extraction

	Functional Specification	Results
1	Is it possible to specify the location of the parasitic capacitance of the Dummy Metal for planarization ?	4/4
2	Is it possible to extract RC parasitic elements only from user-specified net ?	4/4
3	Is it possible to extract RC parasitic elements between user-specified two nets without specifying all nets which exist the two nets ?	2/4
4	Is it possible to define the probe point or the external pin by specifying the X-Y coordinate ?	4/4
5	Is it possible to handle aslant wire ? Is it possible to handle any angle ?	4/4
6	Is it possible to extract the parasitic C of Vias ? And which part of C ? (eg.)VIAx-VIAx, VIAx-VIAx-1, VIAx-VIAx+1, VIAx-METALx, VIAx-METALx+1 etc	3/3
7	Is it possible to extract the parasitic C of Contacts ? And which part of C ? (eg.) CON-CON, CON-VIA1, CON-POLY, CON-METAL1, CON-sub etc	3/3
8	Is it possible to switch the output from RC to R only, or C only ?	4/4
9	Is it possible to extract parasitic element while handling middle-hierarchy as a Macro (also called Black Box or Gray Box) and while considering the effect of the conductor in lower-hierarchy ?	3/4
10	Is it possible to extract a Macro (specified hierarchy) while considering the effect of the conductor in upper-hierarchy ?	2/4

26

Investigation of LPE tool's function (3)

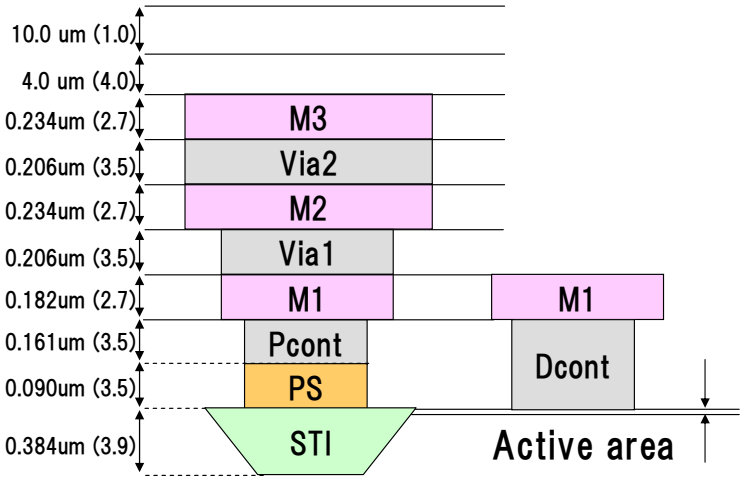
(C) Output formatter and back annotation

	Functional Specification	Results
1	Is it possible to generate DSPF/SPICE/SPEF format netlist ?	4/4
2	Is it possible to generate hierarchical netlist ?	1/4
3	Is it possible to restart the netlister while holding the extract database ?	4/4
4	Is it possible to transform the coupling capacitance to earth capacity according to the threshold level ?	4/4
5	Is reduction of output possible ? Any reduction techniques, control techniques, control parameters ?	4/4
1	Is it possible to back-annotate the extraction results to the schematics ? Can layout-viewer display the parasitic elements ?	4/4
2	Is it possible to perform cross-reference between DSPF/SPICE netlist and schematics ?	4/4

Summary

- Modeling of deep sub micron process effects (copper or LowK wire) is almost taken care. However, some tools don't support dishing shape or shrink the mask layers.
- Only one tool supports the generation of hierarchical netlists.
- Two tools don't support path extraction or macro cell extraction.
- Any tool does not support the extraction of inductance and substrate networks.
- We consider that LPE tools must have not only good accuracy but also individual support for customer requests for specialized device category.

Target Process Structure



	Min L/S [um]
M3	0.138 / 0.138
M2	0.138 / 0.138
M1	0.107 / 0.107
PS	0.053 / 0.053
Diff	-

	Area [um2]
Via2	0.138^2
Via1	0.107^2
Pcont	0. 053^2
Dcont	0. 053^2

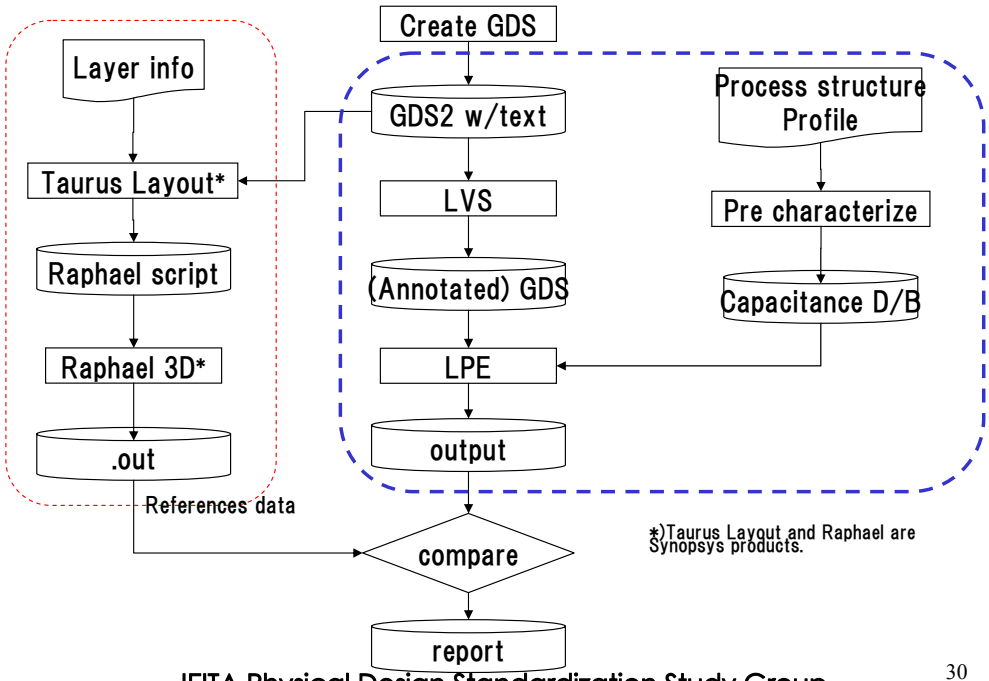
Fig. PEM-E2 Interconnect cross-section

International Technology Roadmap of Semiconductors
2004 UPDATE. Technology node: hp90@2004

JEITA Physical Design Standardization Study Group

29

Evaluation Flow



JEITA Physical Design Standardization Study Group

30

Benchmark. No.1 - 2D/3D Basic

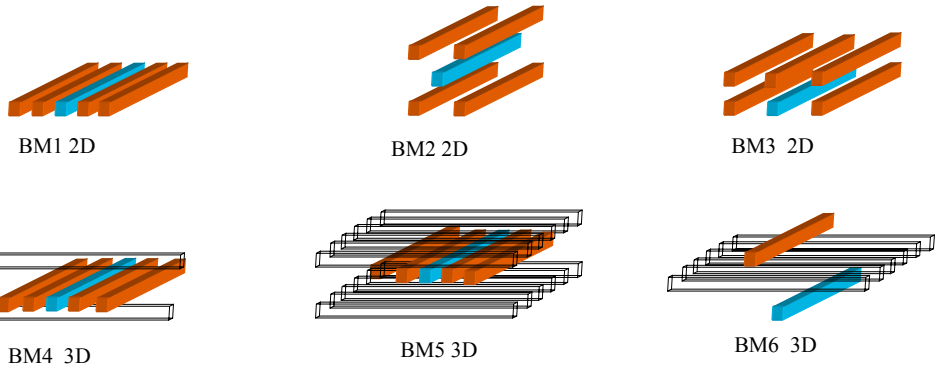
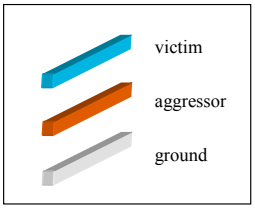


Fig. PEM-E3 No1. 2D/3D Structure



Details of No.1 2D/3D Basic Patterns - Parameter Variations -

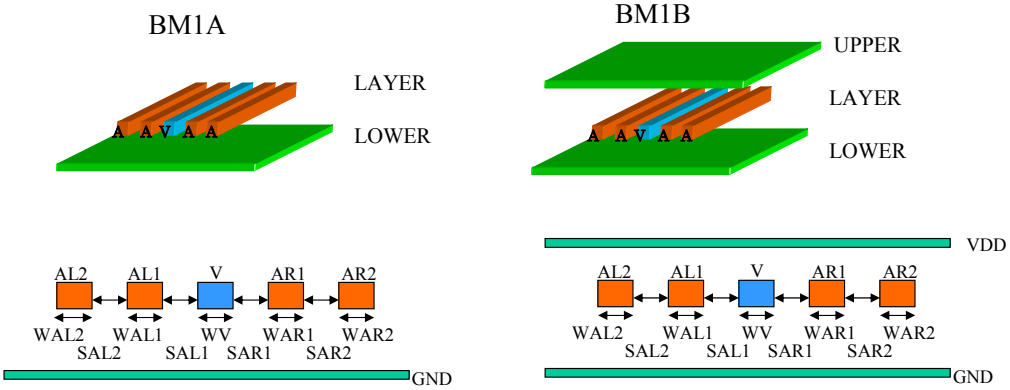
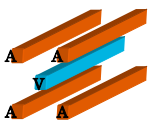


Fig. PEM-E4 Coupling Cap. between V (victim) and each aggressor

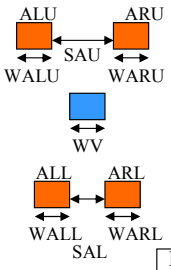
LAYER=PS,M1,M2,M3
Width and spacing of wires are varied as ×1, ×2, ×3 of the minimum size
Width parameters: WV,WAL1,WAL2,WAR1,WAR2
Spacing parameters: SAL1,SAL2,SAR1,SAR2
Here,
BM1A: equal pitch and width for all wires
BM1B: use random parameters for all wire width and spaces using the 9-factor 3-level design table (L27).

Details of No.1 2D/3D Basic Patterns
- Parameter Variations -

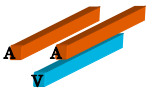
BM2-1



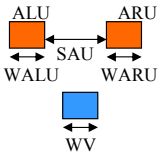
UPPER
LAYER
LOWER



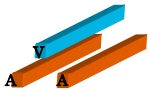
BM2-2



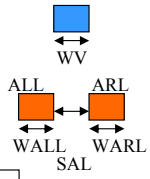
UPPER
LAYER



BM2-3



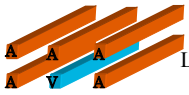
LAYER
LOWER



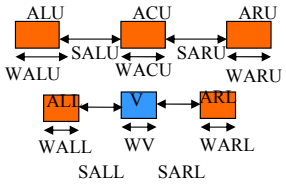
LAYER=PS,M1,M2,M3
Width and spacing = ×1, ×2, ×3 of the minimum size
equal pitch and width for all wires

Details of No.1 2D/3D Basic Patterns
- Parameter Variations -

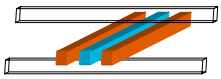
BM3



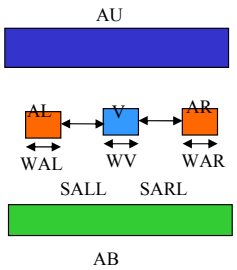
UPPER
LAYER



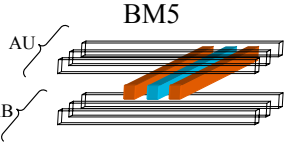
BM4

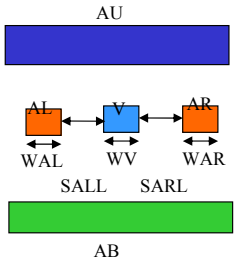


UPPER
LAYER
LOWER



BM5

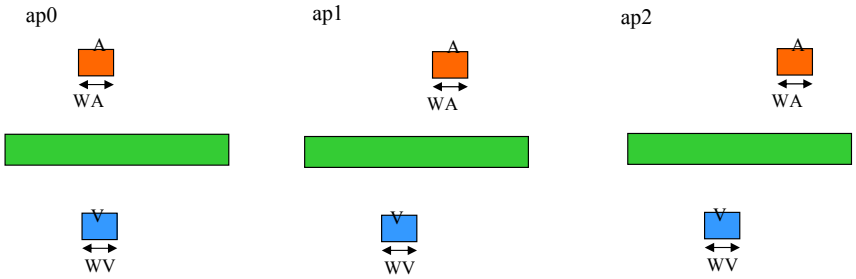




LAYER=PS,M1,M2,M3
Width and spacing = ×1, ×2, ×3 of the minimum size
equal pitch and width for all wires

Details of No.1 2D/3D Basic Patterns -The Variations of Parameters-

BM6



Benchmark. No.2 – non-Manhattan

Rotated pattern of No.1 (BM1A and BM2-1).

Rotation angle θ = { 30, 45, 90 }

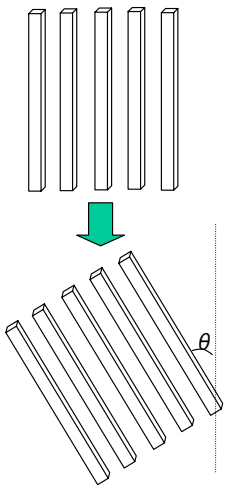


Fig. PEM-E5 No2. non-Manhattan Structure

Benchmark No.3 (1)
- Investigation of the Specification for Benchmark Data that are Evaluated Accuracy of Parasitic Capacitance Extraction of Contacts

Purpose: see if extraction results are consistent over range of gate – contact distance.

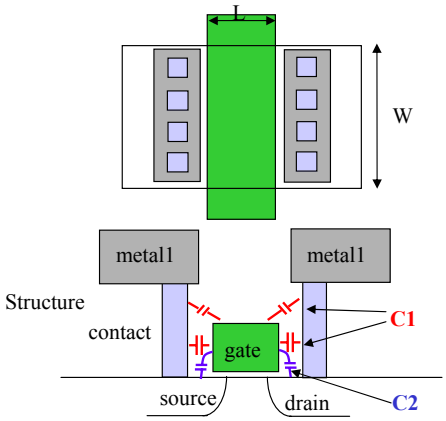


Fig. PEM-E6 Coupling capacitance between gate and contact

C1 :Coupling capacitance between Gate and Contact ⇒ Extraction with LPE
C2 :Sidewall capacitance Between S/D area and Gate ⇒ SPICE

Benchmark No.3 (2)
- Investigation of the Specification for Benchmark Data that are Evaluated Accuracy of Parasitic Capacitance Extraction of Contacts

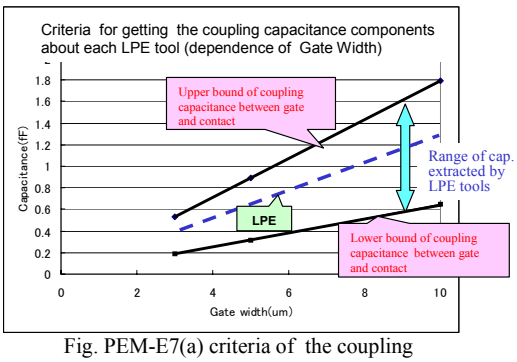


Fig. PEM-E7(a) criteria of the coupling capacitance between gate and contact (dependence of gate width)

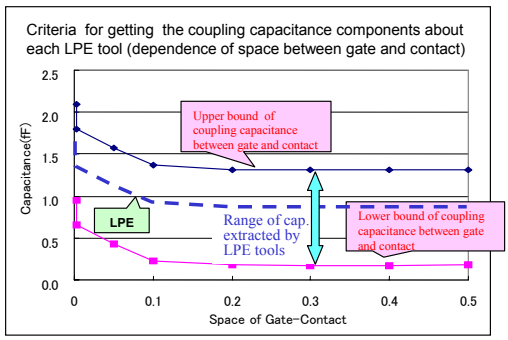
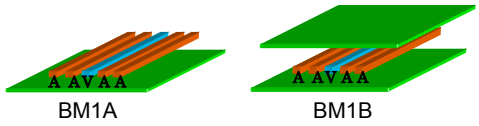


Fig. PEM-E7(b) criteria of the coupling capacitance between gate and contact (dependence of space between gate and contact)

Benchmark Results No.1 BM1A & B (2D) Total Capacitance

One tool is not so accurate in no upper layer cases.
σ of relative error of the best tool is 1.4% .



BM1A BM1B

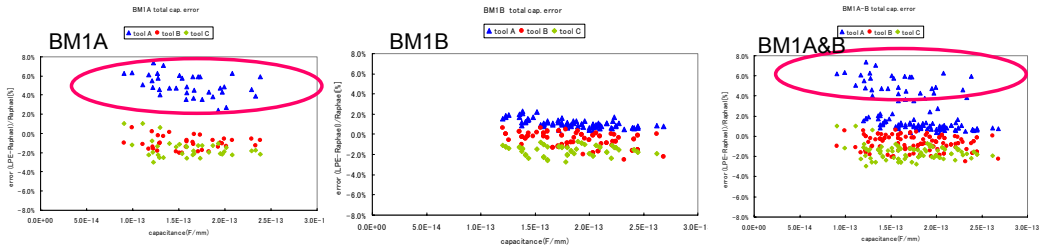


Fig. PEM-E8 NO1. BM1 results(1)

Relative error

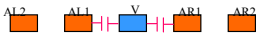
BM1A	tool A	tool B	tool C	ALL
ave	5.0%	−1.0%	−1.6%	0.8%
sigma	1.2%	0.7%	0.9%	3.1%
Min	2.4%	−2.0%	−3.0%	−3.0%
Max	7.4%	0.6%	1.0%	7.4%
ave+sigma	6.2%	1.7%	2.5%	3.9%
3(ave+sigma)	18.5%	5.0%	7.5%	11.8%

BM1B	tool A	tool B	tool C	ALL
ave	1.1%	−0.5%	−1.6%	−0.2%
sigma	0.4%	0.7%	0.5%	1.2%
Min	0.3%	−2.5%	−2.7%	−2.7%
Max	2.3%	0.6%	−0.7%	2.3%
ave+sigma	1.5%	1.2%	2.0%	1.4%
3(ave+sigma)	4.5%	3.6%	6.1%	4.2%

BM1A,B	tool A	tool B	tool C	ALL
ave	2.3%	−0.7%	−1.6%	0.1%
sigma	2.0%	0.7%	0.7%	2.1%
Min	0.3%	−2.5%	−3.0%	−3.0%
Max	7.4%	0.6%	1.0%	7.4%
ave+sigma	4.2%	1.4%	2.3%	2.2%
3(ave+sigma)	12.7%	4.2%	6.8%	6.7%

Benchmark Results No.1 BM1A & B (2D) Coupling Capacitance [AL1 & AR1]

Tool A is very accurate for BM1B pattern ,but not so for BM1A. Tool A has the best accuracy as a whole.
σ of relative error is 4.6% compared with Raphael.



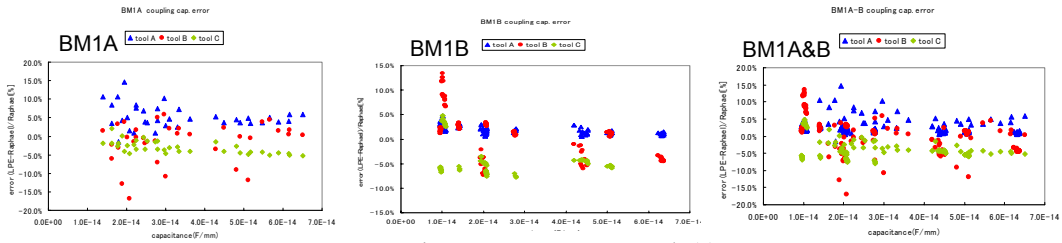


Fig. PEM-E9 NO1. BM1 results(2)

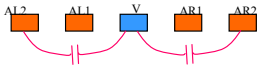
Relative error

BM1A	tool A	tool B	tool C	ALL
ave	5.2%	−1.4%	−3.0%	0.3%
sigma	3.1%	5.3%	1.7%	5.1%
Min	−1.5%	−16.9%	−5.2%	−16.9%
Max	14.6%	5.8%	2.1%	14.6%
ave+sigma	8.3%	6.7%	4.7%	5.4%
3(ave+sigma)	24.9%	20.1%	14.1%	16.1%

BM1B	tool A	tool B	tool C	ALL
ave	2.0%	0.6%	−4.2%	0.5%
sigma	0.9%	4.5%	3.5%	3.9%
Min	0.5%	−7.4%	−7.9%	−7.6%
Max	4.8%	13.4%	4.8%	13.4%
ave+sigma	2.9%	5.1%	7.7%	4.4%
3(ave+sigma)	8.8%	15.4%	23.1%	13.1%

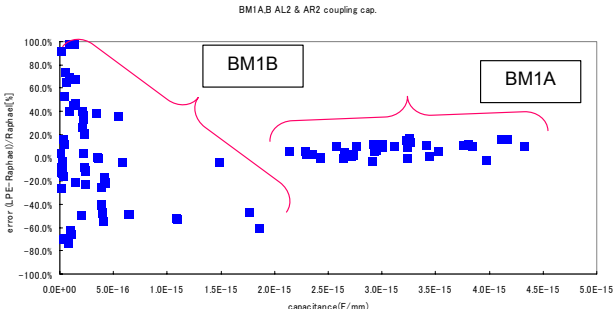
BM1A,B	tool A	tool B	tool C	ALL
ave	2.6%	0.3%	−3.9%	−0.1%
sigma	2.0%	4.7%	3.2%	4.4%
Min	−1.5%	−16.9%	−7.9%	−16.9%
Max	14.6%	13.4%	4.8%	14.6%
ave+sigma	4.6%	5.0%	7.1%	4.4%
3(ave+sigma)	13.7%	15.0%	21.4%	13.2%

Benchmark Results No.1 BM1A & B (2D) Coupling Capacitance [AL2 & AR2]



effect of the second neighbors

Results of a tool which has the best accuracy for this pattern



Relative error

BM1A	
ave	7.4%
sigma	5.4%
Min	-2.6%
Max	17.1%
ave+sigma	12.8%
3(ave+sign	38.4%

Fig. PEM-E10 NO1. BM1 results(3)

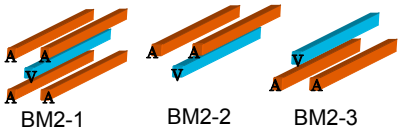
If width & spacing are equal pitch (case BM1A), the coupling capacitance between the second neighbors was extracted accurately.
If width & spacing are varied randomly (case BM1B), error becomes large.
Absolute value is small, but we consider that it becomes more important for analog circuit which has patterns used relative capacitance.

Summary of Benchmark Results No.1 BM1A & B

- Total capacitance: The σ of relative error of the best tool is 1.4% compared with Raphael.
- Coupling capacitance: one of the tools is very accurate about BM1B pattern, but not so accurate about BM1A. However, it has the best accuracy as a whole. The σ of relative error for this tool is 4.6% compared with Raphael.
- Coupling capacitance between the second neighbors: If width & spacing are equal pitch (case BM1A), the coupling capacitance between the second neighbors was extracted accurately.
- If width & spacing are varied randomly (case BM1B), error becomes large. We consider that this is more important for analog circuits which have patterns utilizing relative capacitance.

Benchmark Results No.1 BM2 (2D)

The error occurs by patterns not included in IPL or GRD library.
Tool B has the best of accuracy about total error, but the error of each coupling capacitance is large.



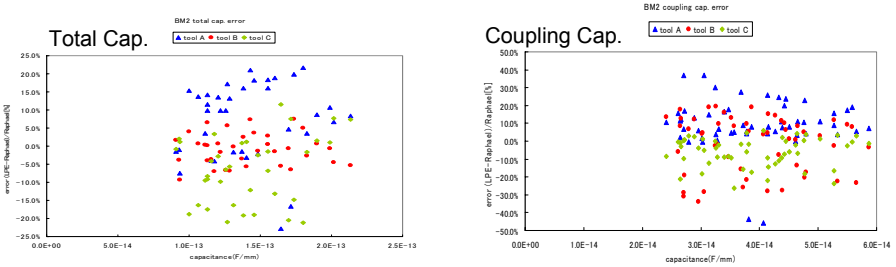


Fig. PEM-E11 NO1. BM2 results

Relative error

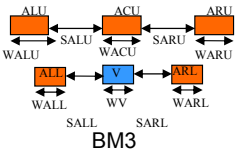
BM2 total				
	tool A	tool B	tool C	ALL
ave	7.2%	−0.9%	−6.5%	−0.1%
sigma	10.6%	4.4%	9.4%	10.2%
Min	−22.8%	−9.3%	−21.2%	−22.8%
Max	21.7%	7.4%	11.5%	21.7%
ave+sigma	17.8%	5.3%	15.9%	10.2%
3(ave+sigma)	53.4%	15.8%	47.6%	30.7%

Relative error

BM2 coupling				
	tool A	tool B	tool C	ALL
ave	9.5%	−1.4%	−5.9%	1.3%
sigma	14.0%	15.7%	8.1%	13.6%
Min	−45.9%	−33.9%	−26.3%	−33.9%
Max	37.1%	19.7%	6.1%	37.1%
ave+sigma	23.5%	17.1%	14.0%	14.9%
3(ave+sigma)	70.6%	51.2%	42.0%	44.8%

Benchmark Results No1. BM3 (2D)

The error occurs by patterns not included in IPL or GRD library.
The relative error of coupling capacitance between V and ALU,ARU is large, but their absolute values are very small.
On the other hand the accuracy of coupling capacitance between same layers, or overlap capacitance is good.
Tool A's variability of relative error is larger than other tools.



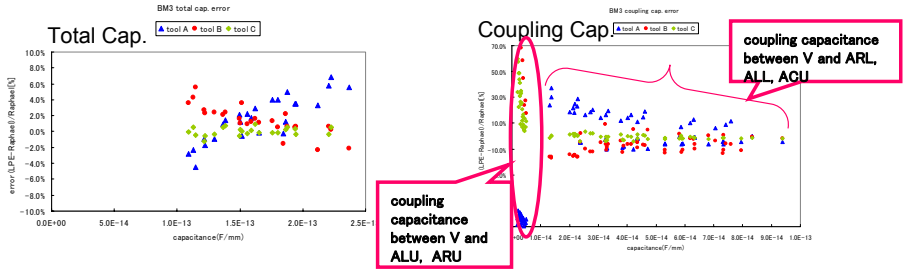


Fig. PEM-E12 NO1. BM3 results

Relative error

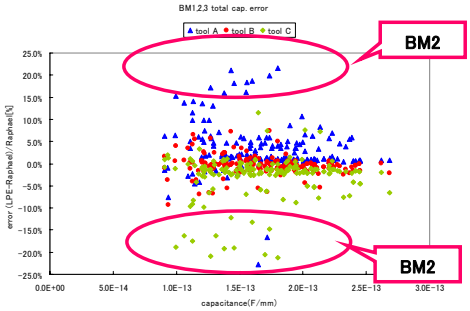
BM3 total				
	tool A	tool B	tool C	ALL
ave	1.6%	1.4%	0.0%	1.0%
sigma	2.9%	1.8%	0.5%	2.1%
Min	−4.4%	−2.3%	−1.2%	−4.4%
Max	6.9%	5.5%	0.9%	6.9%
ave+sigma	4.4%	3.2%	0.5%	3.1%
3(ave+sign	13.3%	9.7%	1.5%	9.4%

Relative error

BM3 Coupling Cap.				
	tool A	tool B	tool C	ALL
ave	6.7%	−7.2%	−0.7%	−0.4%
sigma	12.8%	6.2%	1.8%	10.2%
Min	−10.2%	−16.9%	−4.0%	−16.9%
Max	37.3%	9.2%	3.5%	37.3%
ave+sigma	19.5%	13.5%	2.5%	10.6%
3(ave+sign	58.4%	40.4%	7.5%	31.8%

Benchmark Results BM1-3 (2D) ALL

BM1,2,3 Total capacitance



Relative error

BM1,2,3 (2D) Total cap. Error				
	tool A	tool B	tool C	ALL
ave	3.2%	−0.4%	−2.8%	0.0%
sigma	5.5%	2.3%	4.8%	5.0%
Min	−22.8%	−9.3%	−21.2%	−22.8%
Max	21.7%	7.4%	11.5%	21.7%
ave+sigma	8.6%	2.7%	7.6%	5.1%
3(ave+sign	25.9%	8.0%	22.9%	15.2%

Fig. PEM-E13 NO1. BM1-3 results

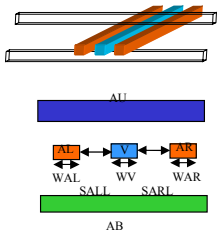
Tool B has the best accuracy for all 2D patterns. The σ of relative error of this tool is 2.7% compared with Raphael.
Variability of the relative error of tool A and tool C is larger than tool B.

Summary of Benchmark Results - No.1 2D patterns BM1-3 -

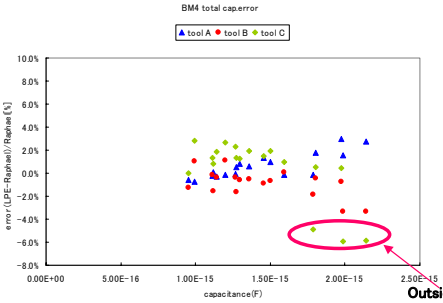
- Total capacitance:
  - All tools have quite good accuracy for the 2D basic patterns.
  - In case the patterns don't be included in IPL or GRD library, the relative error is over 10% compared Raphael.
- Coupling capacitance between same layers:
  - All tools have quite good accuracy.
  - One of the tools has large error in cases with no upper layer.
  - If width & spacing are equal pitch (case BM1A), the coupling capacitance between the second neighbors was extracted accurately.
  - If width & spacing are varied randomly (case BM1B), error becomes large. We consider that this is more important for analog circuits which have patterns utilizing relative capacitance.
- Coupling capacitance between different layers:
  - The error occurs by patterns not included in IPL or GRD library.

Benchmark Results No.1 BM4 (3D) Total Capacitance

Every tool has good accuracy.
σ(relative error) is in a range from 1.7% to 3.1 % compared with Raphael.



BM4



Outside extraction range of capacitance primitive library.

Fig. PEM-E14 BM4 Total capacitance results

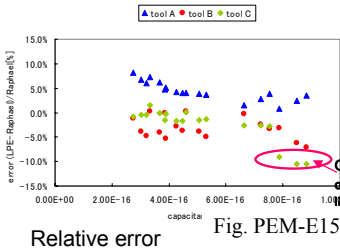
Relative error

BM4 (3D) Total cap. Error				
	tool A	tool B	tool C	ALL
ave	0.6%	−0.9%	0.3%	0.0%
sigma	1.1%	1.2%	2.8%	1.9%
Min	−0.7%	−3.4%	−6.0%	−6.0%
Max	3.0%	1.1%	2.8%	3.0%
ave+sigma	1.7%	2.1%	3.1%	1.9%
3(ave+sign	5.1%	6.2%	9.2%	5.8%

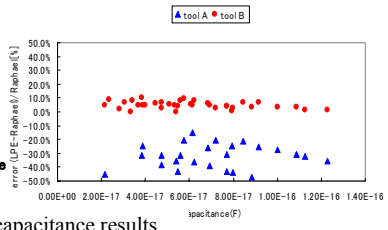
Benchmark Results No.1 BM4 (3D) Coupling Capacitance

Tool B has good accuracy about both Lateral coupling capacitance and Cross coupling capacitance.
Other tools' error becomes large about cross coupling.

(a) Lateral coupling capacitance



(b) Cross coupling capacitance

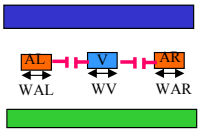


Outside extraction range of capacitance primitive library.

Fig. PEM-E15 BM4 Coupling capacitance results

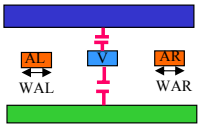
Relative error

BM4 lateral coupling				
	tool A	tool B	tool C	ALL
ave	4.4%	−3.1%	−2.6%	−0.4%
sigma	1.9%	2.2%	3.6%	4.3%
Min	0.9%	−7.0%	−10.5%	−10.5%
Max	8.2%	0.3%	1.6%	8.2%
ave+sigma	6.3%	5.3%	6.2%	4.8%
3(ave+sign	19.0%	15.9%	18.5%	14.3%



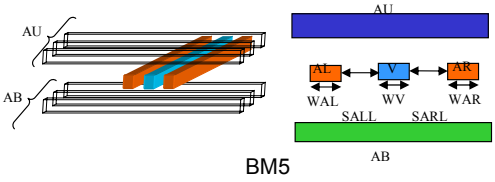
Relative error

BM4 cross coupling			
	tool A	tool B	ALL
ave	−40.3%	4.8%	−17.7%
sigma	14.8%	2.7%	25.0%
Min	−67.1%	−0.1%	−67.1%
Max	−15.2%	10.6%	10.6%
ave+sigma	55.1%	7.5%	42.8%
3(ave+sign	165.2%	22.5%	128.3%

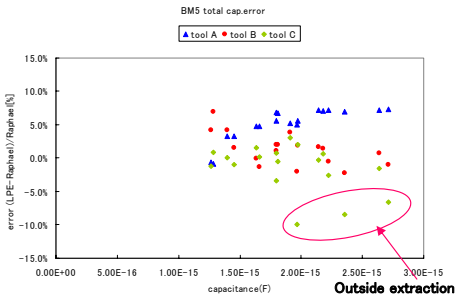


Benchmark Results No.1 BM5 (3D) Total Capacitance

Every tool has almost good accuracy.
σ(relative error) is less than 10%.



BM5



Outside extraction range of capacitance primitive library.

Relative error

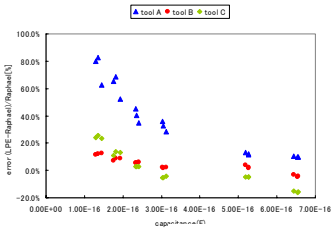
BM5 (3D) Total cap. Error				
	tool A	tool B	tool C	ALL
ave	5.1%	1.4%	−1.5%	1.7%
sigma	2.5%	2.4%	3.6%	3.9%
Min	−0.7%	−3.4%	−6.0%	−6.0%
Max	−0.8%	−2.3%	−10.0%	−10.0%
ave+sigma	7.6%	3.8%	5.0%	5.6%
3(ave+sign	22.9%	11.3%	15.1%	16.8%

Fig. PEM-E16 BM5 Total capacitance results

Benchmark Results No.1 BM5 (3D) Coupling Capacitance

Same as BM4 tool B has good accuracy about both Lateral coupling capacitance and Cross coupling capacitance.
Other tools' error becomes large about cross coupling.

(a)Lateral Coupling Cap.



(b)Cross Coupling Cap.

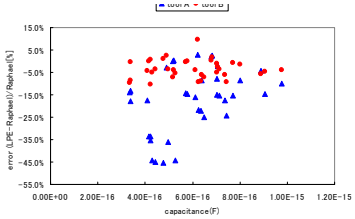


Fig. PEM-E17 BM5 Coupling capacitance results

Relative error

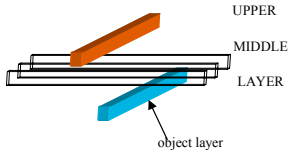
BM5 lateral coupling				
	tool A	tool B	tool C	ALL
ave	38.6%	4.5%	2.3%	15.1%
sigma	25.1%	5.3%	13.4%	23.4%
Min	9.7%	−4.6%	−16.2%	−16.2%
Max	82.6%	12.6%	25.3%	82.6%
ave+sigma	63.7%	9.7%	15.8%	38.6%
3(ave+sign	191.1%	29.2%	47.3%	115.7%

Relative error

BM5 lateral coupling			
	tool A	tool B	ALL
ave	−17.6%	−3.5%	−10.5%
sigma	14.3%	4.2%	12.6%
Min	−45.3%	−10.4%	−45.3%
Max	3.0%	9.8%	9.8%
ave+sigma	31.8%	7.7%	23.1%
3(ave+sign	95.5%	23.1%	69.4%

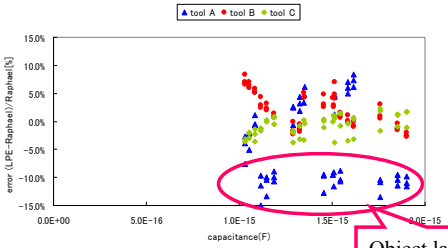
Benchmark Results No.1 BM6 (3D) Total Capacitance

The relative error of every tool is almost less than 10%.
When an object layer is Poly, the relative error about tool A becomes over 10%.



BM6

BM6 Total capacitance



Object layer is Poly.

Fig. PEM-E18 BM6 Total capacitance results

Relative error

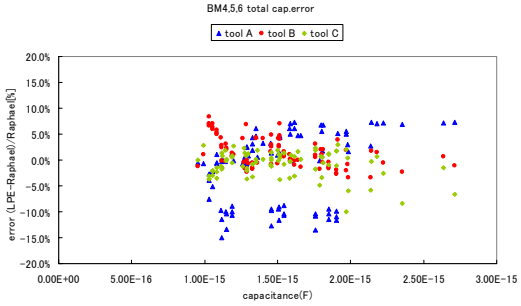
BM6 (3D) Total cap. Error				
	tool A	tool B	tool C	ALL
ave	−4.4%	2.1%	−0.8%	−1.1%
sigma	7.1%	2.9%	1.7%	5.3%
Min	−15.1%	−2.8%	−3.8%	−15.1%
Max	8.4%	8.3%	2.3%	8.3%
ave+sigma	11.5%	5.1%	2.5%	6.4%
3(ave+sign	34.5%	15.3%	7.6%	19.2%

Benchmark Results No.1 BM6 (3D) Coupling capacitance

- The relative error of all tools is too large (> 100%).
- The absolute value of coupling capacitance is very small compared with total capacitance (1e-18(F) - 1e-17(F)).
- We consider that these small capacitances will become more important for analog circuits where they have an influence on circuit. To handle this, LPE tools should be improved in future.

Benchmark Results BM4-6 (3D) ALL

Every tool has almost good accuracy.
σ(relative error) is less than 10%.
When an object layer is Poly in BM6 , the relative error about tool A becomes over 10%.



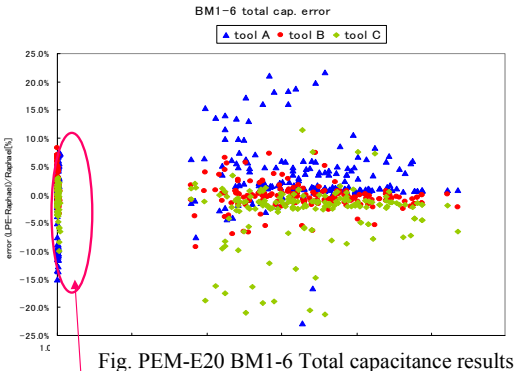
Relative error

BM4,5,6 (3D) Total cap. Error				
	tool A	tool B	tool C	ALL
ave	−1.6%	1.4%	−0.7%	−0.3%
sigma	6.8%	2.8%	2.5%	4.6%
Min	−15.1%	−3.4%	−10.0%	−15.1%
Max	7.3%	8.3%	3.0%	8.3%
ave+sigma	8.4%	4.2%	3.2%	5.0%
3(ave+sign	25.3%	12.5%	9.5%	14.9%

Fig. PEM-E19 BM4,5,6 Total capacitance results

Benchmark Results No1. BM1-6 (2D,3D) ALL

σ (relative error) which has the best accuracy is 2.7% compared with Raphael and each relative error is less than 10%.



Relative error

	Tool A	Tool B	Tool C
average	1.6%	0.1%	-2.1%
σ	6.3%	2.6%	4.3%
Min	-22.8%	-9.3%	-21.2%
Max	21.8%	8.4%	11.5%
ave + σ	7.9%	2.7%	6.4%
3(ave + σ)	23.7%	8.1%	19.2%
#pattern	270 pattern	270 pattern	270 pattern

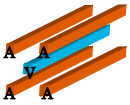
Fig. PEM-E20 BM1-6 Total capacitance results

Cpacitance of BM4,5,6 (3D) are very small compared BM1,2,3 (2D).

Benchmark Results No.2 (non-Manhattan patterns) - (1)

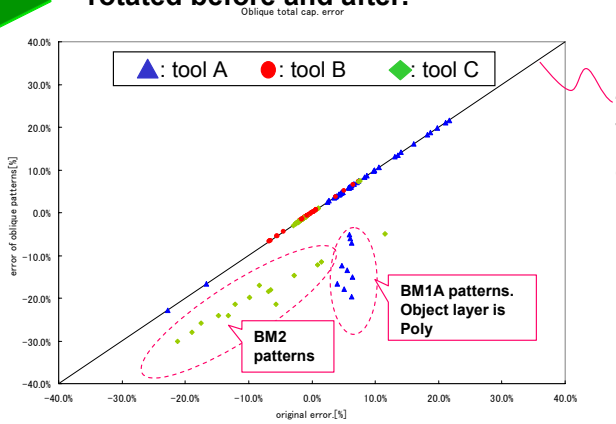
No.1 (BM1A and BM2-1) patterns are rotated.

BM1A



BM2

Rotation angle=90°
Comparison of Total capacitance error rotated before and after.



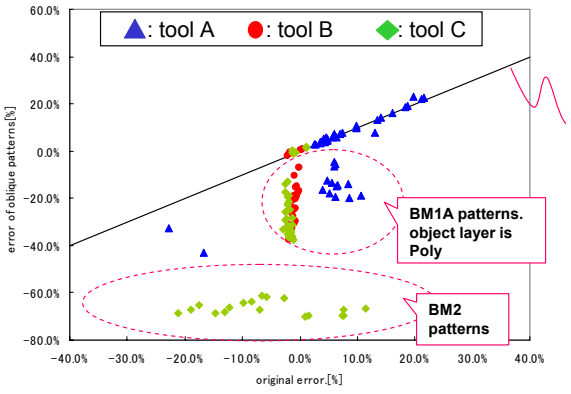
This line shows that the error rotated before and after are equal.

Fig. PEM-E21(a) NO2. non-Manhattan pattern results

Rotation angle = 90°

Benchmark Results No.2 (non-Manhattan patterns) - (2)

Rotation angle=45°
Comparison of Total capacitance error rotated before and after.



This line shows that the error rotated before and after are equal.

Fig. PEM-E21(b) NO2. non-Manhattan pattern results

Rotation angle = 45°

Benchmark Results No.2 (non-Manhattan patterns) - (3)

One of the tools' Results

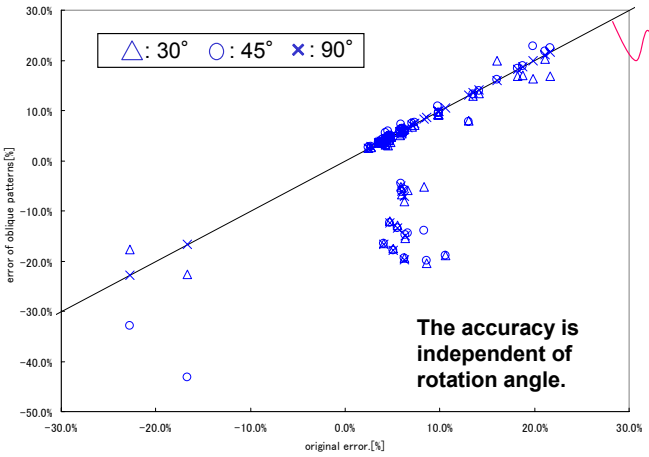


Fig. PEM-E21(c) NO2. non-Manhattan pattern results

Tool A

Summary of Benchmark Results No.2 (non-Manhattan patterns)

- One tool cannot extract the patterns with rotation angle of 30 degree.
- Other tools can extract for any rotation angle.
- All tools can accurately extract for rotation angle of 90 degree.
- Errors in non-Manhattan patterns are larger than in Manhattan patterns.

Benchmark Results No.3
Coupling Capacitance Between Contact and Gate (1)

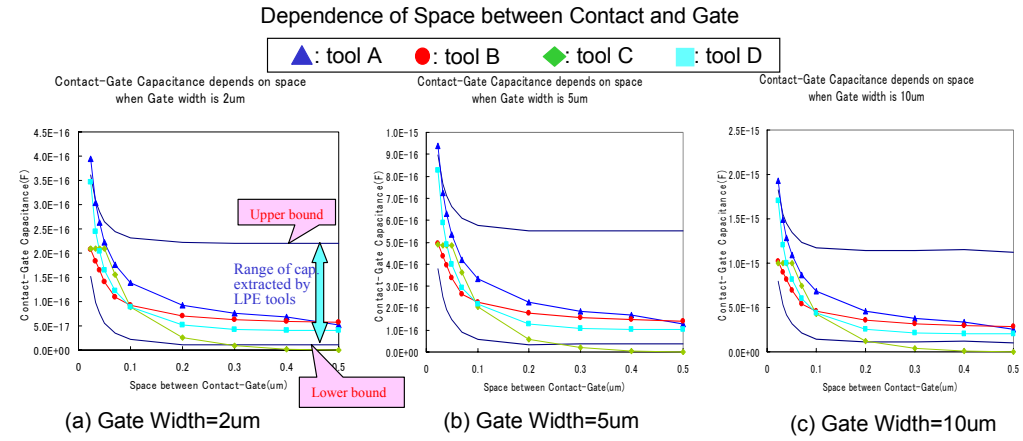
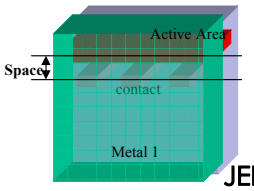


Fig. PEM-E22 NO3. results(1)



Almost of all tools satisfied target criteria.
One of the tools is out of range at space > 0.2um.

Benchmark Results No.3
Coupling Capacitance Between Contact and Gate (3)

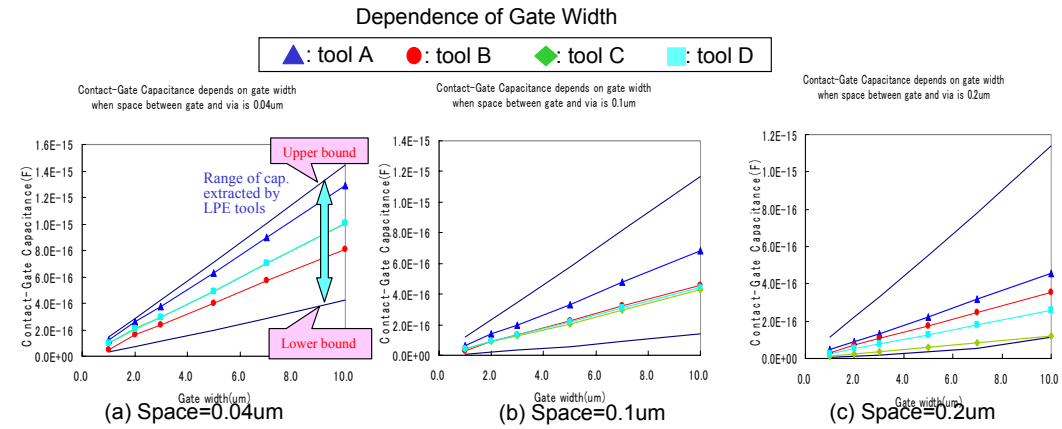
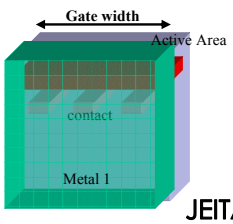


Fig. PEM-E23 NO3. results(2)



All tools satisfied target criteria.

Summary of Benchmark Results No.3

- We investigated of the specification for benchmark required to evaluate accuracy of contact capacitance extraction.
- All the tools mostly satisfied target criteria.
- All tools are able to extract coupling capacitance between gate and contact.

Summary of Benchmark Results

- Summary
  - We compared the function of commercial LPE tools.
  - We can know the accuracy of them by evaluating of 2D/3D basic benchmark data. As a result, σ for total capacitance that has best accuracy is 2.7%.
  - We investigated of specification for the parasitic extraction of contact, and we found that all tools are almost satisfied target criteria.
- Future Work
  - Formulations of LPE errors on circuit design.
  - Evaluation of the consistency of accuracy, performance and design environment.

References

[E1] T. Kanamoto, et al., “Impact of intrinsic parasitic extraction errors on timing and noise estimation,” *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol.E89-A, no.12, pp.3666-3670, Dec. 2006.

[E2] N. D. Arora, K. V. Raol, R. Schumann, and L. M. Richardson, “Modeling and extraction of interconnect capacitances for multilayer VLSI circuits,” *IEEE Trans. Computer-Aided Design*, vol.15, no.1, pp.58-67, Jan. 1996.

[E3] International Technology Roadmap for Semiconductors 2004 Update.

[E4] Synopsys, Inc., Raphael Reference Manual, Raphael V-2004.06 ed., 2004.

[E5] N. D. Arora, “Modeling and characterization of copper interconnects for SoC design,” *Proceeding of International Conference on Simulation of Semiconductor Processes and Devices 2003*, pp.1-6, Sept. 2003.

6. 全体のまとめと課題

- まとめ
  - 温度依存抵抗変動による回路特性へのインパクト調査
  - 遅延ばらつき特性の環境温度依存性評価
  - 市販されているLPEツールの精度評価
- 課題
  - チップ内温度ばらつきを考慮した設計手法の確立
  - LPEツールの精度ばらつきの設計への反映方法の明確化
  - 精度/速度/設計環境との整合性を鑑みたLPEツールの有効性の評価

4.1.2 統計的デザインメソドロジー タスクグループ

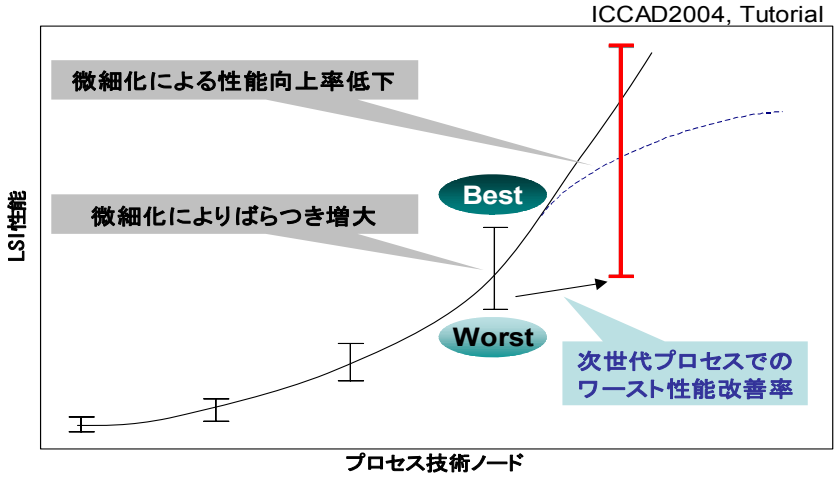
日本シノプシス　小林　宏行
ジーダット・イノベーション　小野　信任
富士通VLSI　奥村　隆昌
ルネサステクノロジ　増田　弘生
NECエレクトロニクス　中島　英斉
リコー　高藤　浩資
[客員] 東京工業大学　佐藤　高史
[客員] 大阪大学　橋本　昌宜

報告内容

- はじめに
- 従来の静的タイミング解析
- スルー依存性を考慮した遅延ばらつきの計算手法提案
- 提案手法の検証
  - キャラクタライズ
  - モンテカルロ解析
  - 比較検証結果
- まとめと今後の課題

微細化によるLSI性能ばらつきの増大

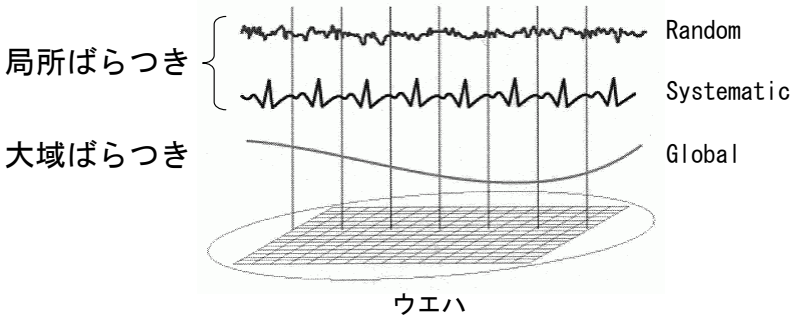
- 微細化に伴い製造、設計ばらつきの増加
  - コーナーベース解析の限界



図SDM-1　LSI性能のトレンド

ばらつきの分類

- 大域ばらつき (Inter-die)
  - ロット間(L2L), ウエハ間(W2W), チップ間(D2D)ばらつき
- 局所ばらつき (Intra-die)
  - チップ内(WID, OCV, etc), セル内ばらつき

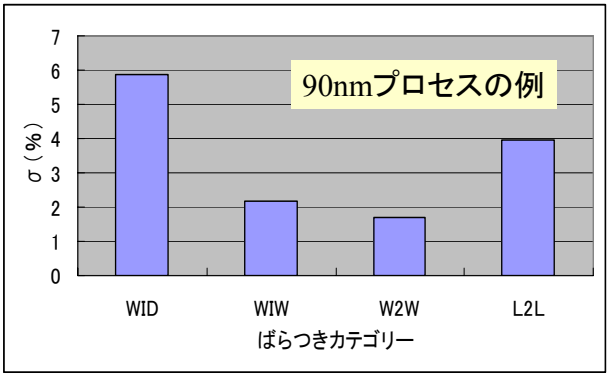


図SDM-2　ばらつきの分類

出典　システム・デザイン・フォーラム2007

ばらつきの要因分析

- 130nm以降でWIDの影響が顕著となる



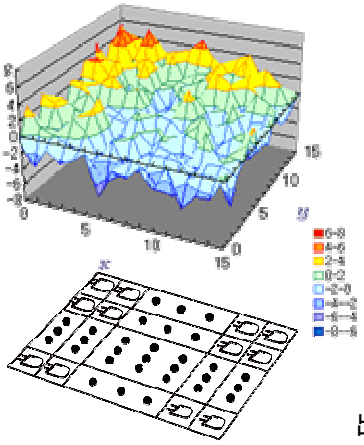
WID:チップ内ばらつき
WIW:ウエハ内ばらつき
W2W:ウエハ間ばらつき
L2L:ロット間ばらつき

図SDM-3　ばらつきの要因分析

出典　VLSI夏の学校, 8月, 2006年 [1]

チップ内回路特性ばらつきの計測結果

- 回路特性(遅延)ばらつき



リングオシレータ遅延ばらつき

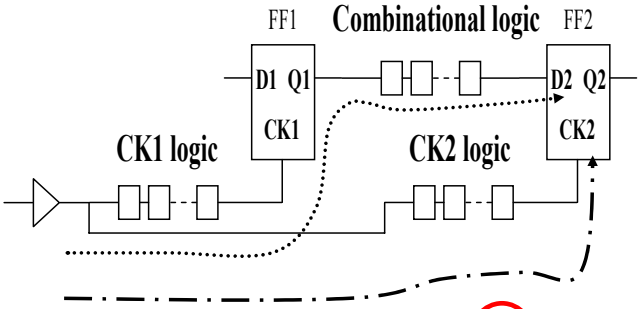
平均値:%σ＝3-4%

従来コーナーモデルではチップ内ミスマッチ余裕を3σ値で10-15%とらなければならない

出典　VLSI夏の学校, 8月, 2006年 [1] [2]

図SDM-4　回路特性ばらつき

従来の静的タイミング解析（セットアップ･ホールド）



$$t_{CK1} - t_{CK2} + t_{CK1\text{-}Q1} + t_{logic} + t_{setup} + \delta_{WID} \le t_{clk}$$

$$t_{CK1} - t_{CK2} + t_{CK1\text{-}Q1} + t_{logic} - t_{hold} - \delta_{WID} \ge 0$$

チップ内ばらつきの増大：

タイミング設計における：
・クロックの高速化、
・ホールド違反マージン確保、
が難しくなる [3]

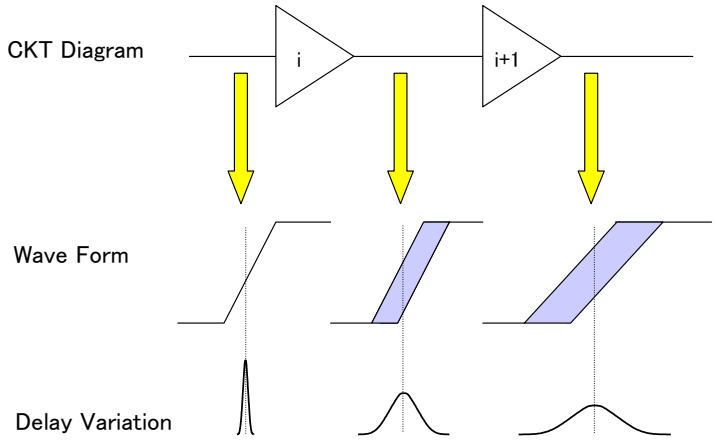
図SDM-5　従来のSTA

「ばらつき」を統計的に扱う統計的STA

- 従来のSTAではマージンが大きくなりすぎて
  タイミング設計が収束しない
- ばらつきを統計分布で計算する新しいツール
  「統計的STA＝SSTA」がベンダーから提供
- SSTAの計算アルゴリズム [4] [5] [6]
  - 遅延時間分布がほぼ正規分布で伝播
    - ベンダーの統計的STAツールでモデル化
  - スルー分布も伝播してゆく [7]
    - 現状のベンダーツールで正確にモデル化されていない
- 遅延・スルーばらつきを統計的に表現する
  計算アルゴリズムを開発

WID遅延ばらつきの伝播イメージ

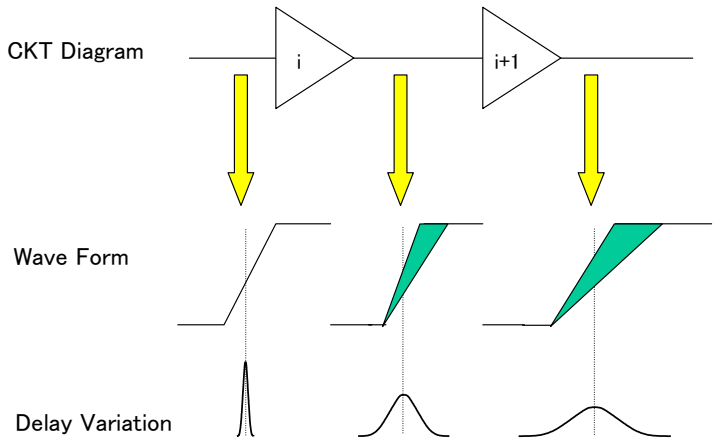
- 遅延ばらつきの伝播



図SDM-6　遅延ばらつきの伝搬

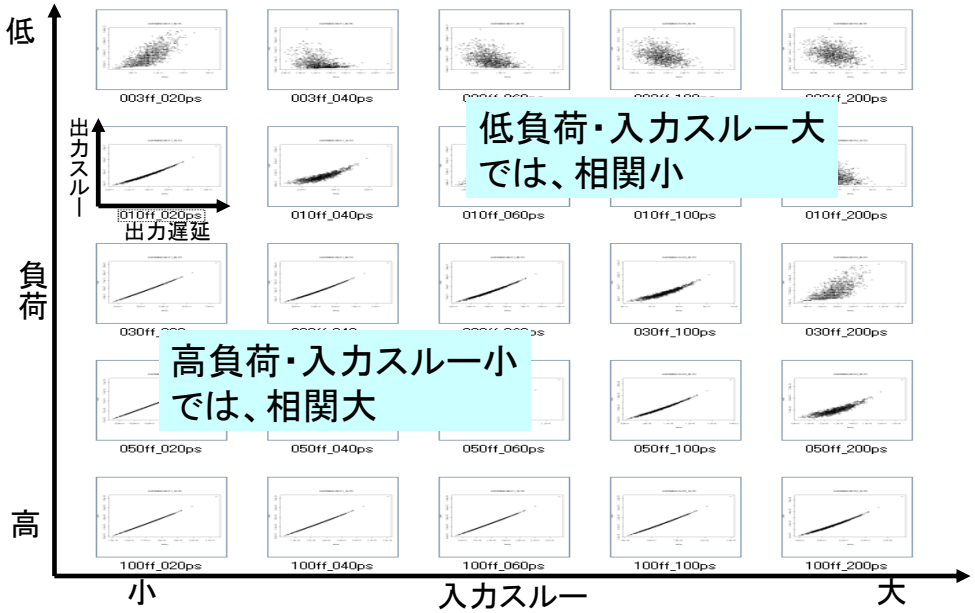
スルー遅延ばらつきの伝播イメージ

- スルーばらつきの伝播



図SDM-7　スルーばらつきの伝搬

遅延・スルーのばらつきに相関はあるのか？
INVでの解析例



図SDM-8 INVでのばらつき相関解析例

統計的ばらつき計算手法

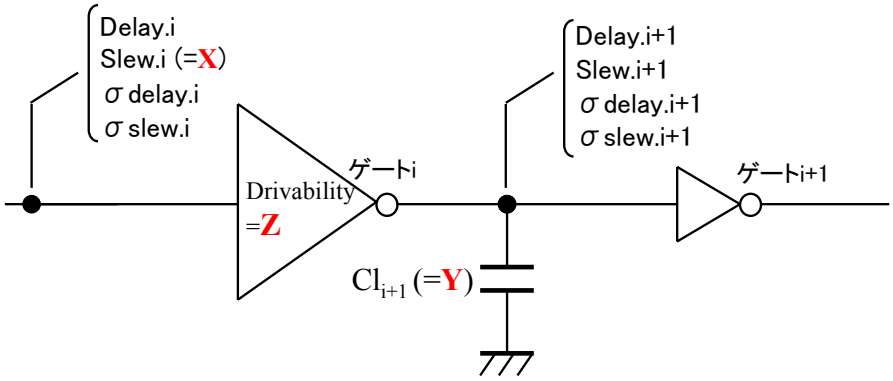
- ベンダーSSTA手法
  - 統計的遅延ばらつき(分布)の伝播
- 実際の回路遅延ばらつきのメカニズム
  - 統計的遅延＆スルーばらつきの伝播
  - 課題
    - スルーばらつきの統計的モデル化
    - 遅延＆スルーばらつきの統計的相関

提案アプローチ

- 精度の保障・従来設計資産の継承
  - テーブル方式の採用
- ばらつきテーブルの相関表現

遅延・スルーばらつきの考え方

- ゲートの電気的特性(出力遅延・スルー)は
  以下の3変数で決定される
  - 入力スルー(X)、出力負荷(Y)、ゲートのドライバビリティ(Z)



図SDM-9 ばらつきの考え方

遅延・スルーばらつきの定式化

- スルー(S)ばらつき
  - *S=g(x,y,z)* とすると

$$\delta S = \frac{\partial g}{\partial x}\delta x + \frac{\partial g}{\partial y}\delta y + \frac{\partial g}{\partial z}\delta z \approx \frac{\partial g}{\partial x}\delta x + \frac{\partial g}{\partial z}\delta z$$

$$Var(S) = Var(X) + Var(Z) + 2Cov(X,Z) \quad \text{ここで: } \delta X = \frac{\partial g}{\partial x}\delta x \quad \delta Z = \frac{\partial g}{\partial z}\delta z$$

- 同様に、遅延(D)ばらつき
  - *D=f(x,y,z)* とすると

$$\delta D = \frac{\partial f}{\partial x}\delta x + \frac{\partial f}{\partial y}\delta y + \frac{\partial f}{\partial z}\delta z \approx \frac{\partial f}{\partial x}\delta x + \frac{\partial f}{\partial z}\delta z$$

$$Var(D) = Var(X) + Var(Z) + 2Cov(X,Z) \quad \text{ここで: } \delta X = \frac{\partial f}{\partial x}\delta x \quad \delta Z = \frac{\partial f}{\partial z}\delta z$$

遅延・スルーばらつきの定式化 (続き)

- 定式化の理解
  - スルー(S)ばらつきの場合

$$\delta S = \frac{\partial g}{\partial x}\delta x + \frac{\partial g}{\partial y}\delta y + \frac{\partial g}{\partial z}\delta z \approx \frac{\partial g}{\partial x}\delta x + \frac{\partial g}{\partial z}\delta z$$

スルーばらつき / 入力スルーばらつき効果 / 負荷容量ばらつき効果 / ゲートTRSばらつき効果 / 〔負荷容量のばらつき小の為無視〕

$$Var(S) = Var(X) + Var(Z) + 2Cov(X,Z)$$

スルーばらつき分散 / 入力スルーばらつき分散 / ゲートTRSばらつき分散 / 2つの要因のばらつき共分散

ばらつき相関＝*Cov(X,Z)*をどう取り扱うかが重要課題

遅延・スルーばらつきの定式化（続き）

- 各項のテーブル計算化
  - スルー(S)ばらつきの場合

$$Var(S) = Var(X) + Var(Z) + 2Cov(X, Z)$$

スルー ばらつき分散 / 入力スルー ばらつき分散 / ゲートTRS ばらつき分散 / 2つの要因の ばらつき共分散

$$(\sigma s_{i+1})^2 = \left(\frac{\partial g_s}{\partial s} \times \sigma s_i\right)^2 + g_{\sigma s}{}^2(cl_{i+1}, s_i) + 2 \times \rho \times \left(\frac{\partial g_s}{\partial s} \times \sigma s_i\right) \times g_{\sigma s}(cl_{i+1}, s_i)$$

σ：標準偏差

ρ＝相関係数

提案アルゴリズム

ゲートiまでの到達遅延

$$arr_{i+1} = arr_i + f_d(cl_{i+1}, s_i)$$

テーブル参照

ゲートiの出力スルー

$$s_{i+1} = g_s(cl_{i+1}, s_i)$$

スルーばらつき考慮の項

ゲートiの出力遅延ばらつき

$$(\sigma d_{i+1})^2 = f_{\sigma d}{}^2(cl_{i+1}, s_i) + \left(\frac{\partial f_d}{\partial s} \times \sigma s_i\right)^2 + 2 \times \rho \times f_{\sigma d}(cl_{i+1}, s_i) \times \left(\frac{\partial f_d}{\partial s} \times \sigma s_i\right)$$

ゲートiの出力スルーばらつき

$$(\sigma s_{i+1})^2 = g_{\sigma s}{}^2(cl_{i+1}, s_i) + \left(\frac{\partial g_s}{\partial s} \times \sigma s_i\right)^2 + 2 \times \rho \times g_{\sigma s}(cl_{i+1}, s_i) \times \left(\frac{\partial g_s}{\partial s} \times \sigma s_i\right)$$

今回は相関係数=1.0固定

ゲートiまでの到達遅延ばらつき

$$(\sigma arr_{i+1})^2 = (\sigma arr_i)^2 + (\sigma d_i)^2$$

※係数：$-1 \le \rho \le 1$

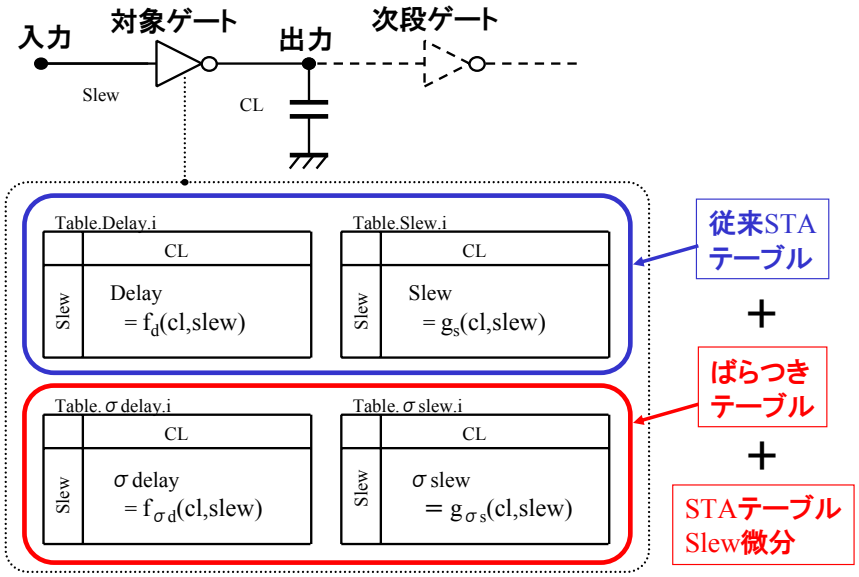
モデル検証の手順

- 条件の決定(65nm、Vthばらつき)
- 実験回路の選定(インバータチェイン)
- ゲートのキャラクタライズ
- SSTA計算(Excel, Perl)
- モンテカルロ回路解析
- 精度検証(比較)

解析条件

- Tech Node=65nm
- NMOS: L=60nm, W=300nm, Vthn0=0.423V
- PMOS: L=60nm, W=500nm, Vthp0=−0.365V
- σ(Vthn0)=30mV
- σ(Vthp0)=25mV
- Vdd=1.0V
- Temp=25℃
- SPICEパラメータ＝Typical [8]

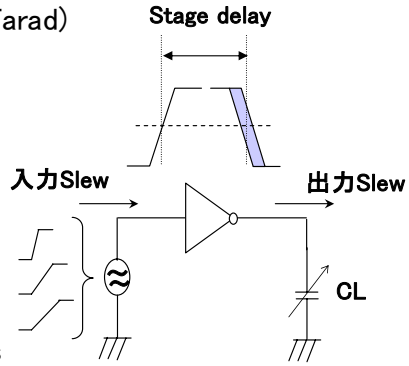
ライブラリ構成



図SDM-10 ライブラリ構成

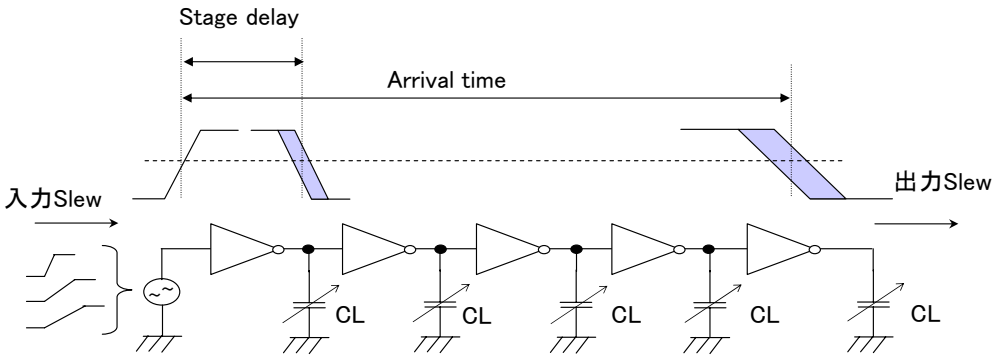
キャラクタライズ条件

- Table Specification parameters
  - Slew Rate=20p, 40p, 60p, 100p, 200p (Sec)
  - Load Capa.=3f, 10f, 30f, 50f, 100f (Farad)
- Characteristic Values
  - Delay: 50%–50% Delay
  - Slew: 20%–80% Slew
- Simulation Method
  - Monte Carlo (2000 Runs)
  - Monte Carlo Parameters:
  - Vtn0 & Vtp0 of all MOS Transistors



図SDM-11 キャラクタライズ条件

実験回路



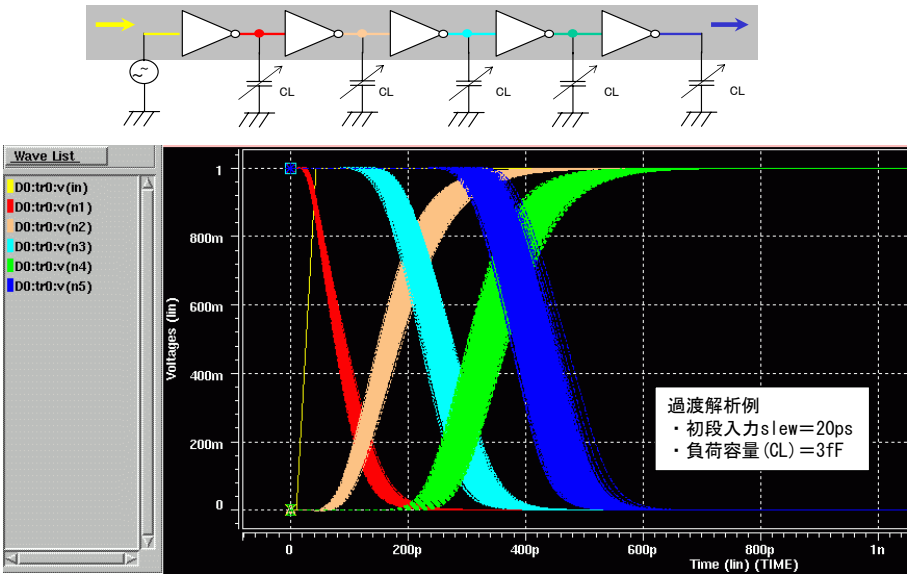
図SDM-12　実験回路

上記の回路に対して以下を実施

・アルゴリズム計算(Perl，Excelで手計算)

・SPICEモンテカルロ計算(HSPICE)

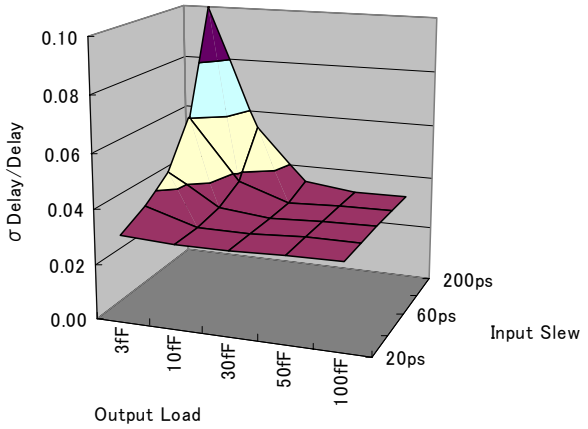
モンテカルロ試行結果



図SDM-13　モンテカルロ試行結果

5段INV出力遅延ばらつき（遅延%σ）
本アルゴリズム計算結果

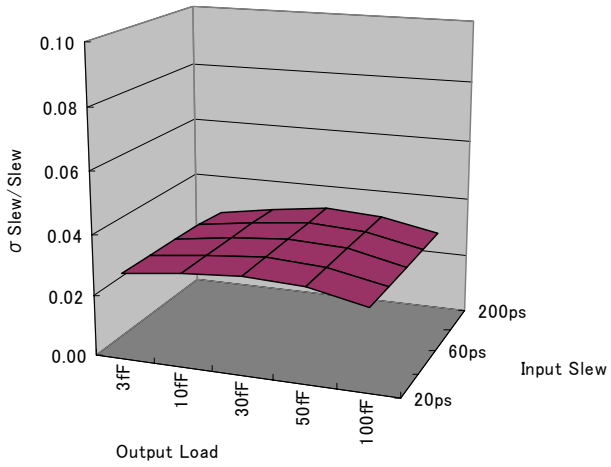
- 高負荷・入力スルー小での遅延σ＝3%以下
- 低負荷・入力スルー大での遅延σ＝Max　10%



図SDM-14　5段INV出力遅延ばらつき結果

5段INV出力スルーばらつき（スルー%σ）
本アルゴリズム計算結果

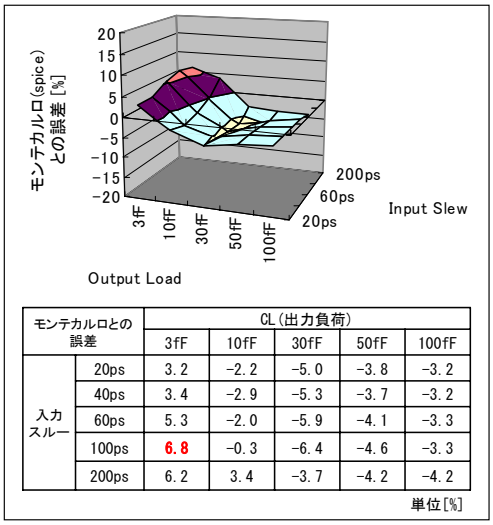
- 負荷・入力スルーすべての条件でスルーσ＝3%以下



図SDM-15　5段INV出力スルーばらつき結果

精度検証結果

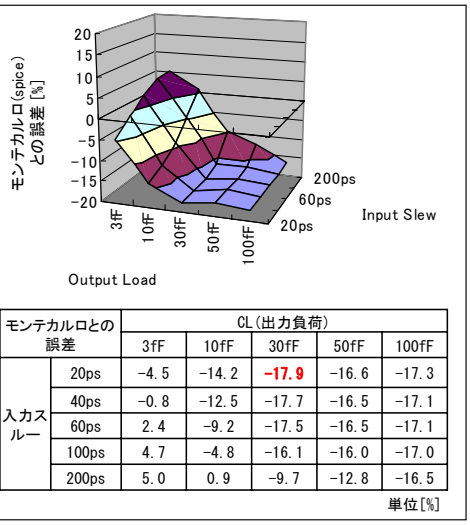
Slewばらつき項を考慮



モンテカルロとの誤差		CL(出力負荷)				
		3fF	10fF	30fF	50fF	100fF
入力スルー	20ps	3.2	−2.2	−5.0	−3.8	−3.2
	40ps	3.4	−2.9	−5.3	−3.7	−3.2
	60ps	5.3	−2.0	−5.9	−4.1	−3.3
	100ps	**6.8**	−0.3	−6.4	−4.6	−3.3
	200ps	6.2	3.4	−3.7	−4.2	−4.2

単位[%]

Slewばらつき項を無視



モンテカルロとの誤差		CL(出力負荷)				
		3fF	10fF	30fF	50fF	100fF
入力スルー	20ps	−4.5	−14.2	**−17.9**	−16.6	−17.3
	40ps	−0.8	−12.5	−17.7	−16.5	−17.1
	60ps	2.4	−9.2	−17.5	−16.5	−17.1
	100ps	4.7	−4.8	−16.1	−16.0	−17.0
	200ps	5.0	0.9	−9.7	−12.8	−16.5

単位[%]

図SDM−16　精度検証結果

まとめ

- 統計的STA のランダム遅延ばらつきの精度を向上する計算手法を検討
- スルー時間ばらつきを遅延ばらつきの計算項として取り入れた
- この提案手法で、スルーばらつき項を無視した場合に比べ、解析平均誤差17.9%が6.8%に低減できることを、モンテカルロ解析で検証した

今後の課題

- 誤差低減の検討
  - 特に低負荷、入力スルーが大きい場合の誤差要因の解明
- 解析パターンの追加
  - 2 入力ゲート等の場合での評価
  - WID(チップ内)の物理パラメータ依存性
- D2D(チップ間)の物理パラメータ依存性の考慮(σ(L)/L など)

参考文献

- [1] 増田弘生、“MOSトランジスタの特性ばらつき”、VLSI 夏の学校, 8月, 2006年
- [2] H. Masuda et al.“Challenge: Variability Characterization and Modeling for 65- to 90-nm Processes”, CICC 2005 Abstract, pp.594-600, Oct., 2005.
- [3] A. Devgan and C. Kashyap, “Block-based statictiming analysis with uncertainty,” in ICCAD, 2003.
- [4] S. Raj, S. Vrudhula, and J. Wang, “A methodology to improve timing yield in the presence of process variations,” in DAC, 2004, pp. 448・ 453.

参考文献（続き）

- [5] S. B. Samaan,; The impact of device parameter variations on the frequency and performance of microprocessor circuits, ISSCC 2004, Microprocessor Circuit Design Forum Digest, p.29, Feb. 2004.
- [6] U. Fassnacht,; A robust ASIC design and IP integration methodology for 65nm and beyond, ICCAD 2004, Sunday Workshop Digest, p3, Nov. 2004.
- [7] H. Chang and S. Sapatnekar, "Statistical Timing Analysis under Spatial Correlations," IEEE Trans. CAD, Vol. 24, No. 9, pp. 1467.1482, Sep. 2005.
- [8] W. Zhao, Y. Cao, "New generation of Predictive Technology Model for sub-45nm design exploration," pp. 585-590, ISQED, 2006.

4.2 SystemCタスクグループ 2006年度活動報告

JEITA EDA技術専門委員会

標準化小委員会

SystemCタスクグループ

JEITA

SystemCタスクグループメンバー

主査	長谷川　隆	富士通㈱
副主査	今井　浩史	東芝㈱
委員	中西　早苗	ＮＥＣエレクトロニクス㈱
	清水　靖介	沖電気工業㈱
	長尾　文昭	三洋半導体㈱
	山田　晃久	シャープ㈱
	柿本　勝	ソニー ㈱
	逢坂　孝司	日本ケイデンスデザインシステムズ社
	中野　淳二	日本シノプシス㈱
	竹村　和祥	松下電器産業㈱
	菊谷　誠	メンターグラフィクスジャパン㈱
	里井　朋樹	㈱リコー
	渡邊　政志	㈱ルネサステクノロジ
客員	今井　正治	大阪大学

（計14名）

JEITA

2

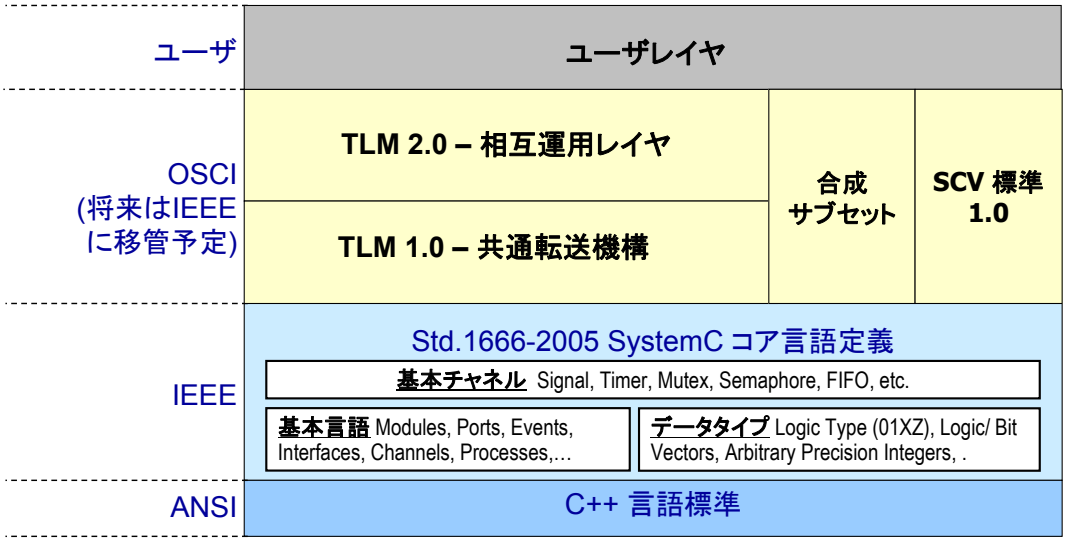
目次

4.2.1 SystemCタスクグループ概要

SystemC とは

- C++言語をベースとした、システムレベル設計言語の代表的な言語である
  - Open SystemC Initiative (OSCI)という標準化組織により、言語仕様(LRM)とリファレンスシミュレータが策定され、無償提供されている
    http://www.systemc.org/
  - 2005年12月に、SystemC 2.1のLRMがIEEE Std. 1666として標準化された
  - 現在は合成サブセットやTLM(Transaction Level Modeling)の標準化案が検討されている
- C++の文法を保持したまま、クラスライブラリの形で以下のような言語拡張がなされている
  - 並列動作を可能とするシミュレーションエンジン(クロック、イベント、等)
  - 抽象化された通信手段(Channels, Interfaces)
  - ハード実装に必要なデータタイプ (固定小数点、固定長ビット、等)
  - 0, 1, Z, X 等の信号値等

SystemCの標準化の階層



OSCI資料(2007)に加筆

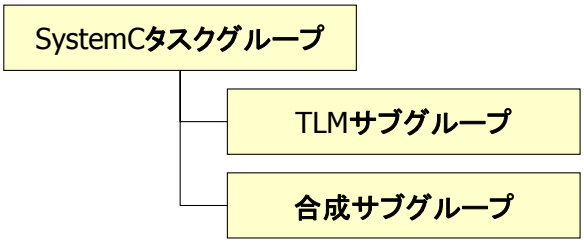
SystemCタスクグループ設立の背景と目的

- 背景
  - SystemCは、SoC(System on Chip)の開発のためのシステムレベル記述言語のひとつとして、既に幅広く使われてきている。また2005年12月にその基本言語仕様がIEEE Std. 1666として国際標準化され、普及が進んでいる。
  - 今後、拡張言語仕様としてTLM(Transaction Level Modeling)や合成サブセットの標準化も予定されている。
- 目的
  - 日本国内におけるSystemCの標準化関連組織として日本国内の事情・要求事項を取り込むべく、OSCIやIEEE P1666ワーキンググループと連携しつつ、SystemCの国際標準化を継続的に進めていく。
  - 国内外におけるSystemCの利用状況の調査や、SystemCの各種拡張仕様についての調査検討、及び動作合成スタイルガイドの提案等を行うことで、国内普及を図る。

JEITA

SystemCタスクグループの体制と歩み

- 2005年度以降の体制



2003年度 2004年度 2005年度

- これまでの歩み
  - OSCI LRMレビュー
  - IEEE P1666 WG
  - SystemC動向調査
  - SystemC 2.1調査
  - SystemC-SystemVerilog共通辞書
  - SystemCユーザ・フォーラム

2005 2006

JEITA

SystemCタスクグループ2006年度の成果

- SystemC標準化活動
  - OSCIよりリリースされたTLM 2.0、及び合成サブセットの各ドラフト仕様についてレビューを行い、問題点や要望事項をOSCIに伝えた
    - TLM 2.0ドラフトに対し4件提案(詳細は4.2.3を参照)
    - 合成サブセットドラフト(1.1.18)に対し57件提案(詳細は4.2.4を参照)
- SystemC技術調査
  - TLMに関する動向調査を実施し、結果をSystemCユーザ・フォーラム2007にて公表
    - 欧州ユーザーと比較し、現状の国内ユーザーは低抽象度のモデルを利用している傾向があることを掴んだ(詳細は4.2.3を参照)
- SystemC普及活動
  - SystemC-SystemVerilog共通辞書の更新を行った(詳細は4.4を参照)
  - 動作合成スタイルガイドの策定に向けて、構成要件を定義(詳細は4.2.4を参照)
  - 2007年1月26日に、EDSF併設のシステムデザインフォーラムの1プログラムとしてSystemCユーザ・フォーラム2007を開催した。(詳細は4.2.2を参照)

JEITA

SystemCワーキンググループ2007年度計画

- 2007年度よりSystemCワーキンググループに名称変更
- SystemC標準化活動
  - OSCIよりTLM 2.0や合成サブセットのIEEE移管が予定されており、事前レビューの実施とIEEE標準化作業に参加する
  - STARCとも連携し、国内での標準化要望をIEEE等に伝える
- SystemC技術調査
  - TLMに関する技術動向調査を継続して実施する
  - 今後TLMからの動作合成技術が進むと予想されており、標準化すべき項目について技術調査、検討を行う
- SystemC普及活動
  - 動作合成スタイルガイド構成要件に従ったスタイルガイドが作成されるよう業界に働きかける
  - SystemCユーザ・フォーラムやEDA-TCのWEBを活用して、積極的に情報発信を行う

JEITA

4.2.2 SystemCユーザ・フォーラム 2007 開催報告

 JEITA

SystemCユーザ・フォーラム2007概要

- 主催：JEITA EDA技術専門委員会
- 協賛：OSCI
- 日時：2007年1月26日　10:00～12:00
- 会場：パシフィコ横浜アネックスホールF205+F206 (定員200名)
- 講演内容：
  - 司会
    - 長谷川 隆氏(SC-TG主査／富士通)
  - 『SystemC Community Update』
    - Patrick Sheridan氏(OSCI／CoWare)
  - 『SystemCベースのTLMと動作合成に関する取り組み』
    - 里井 朋樹氏(リコー)、長尾 文昭氏(三洋半導体)
  - 『TLM標準化の動向について』
    - 武井 勉氏(STARC)
  - 『ソニーにおける動作合成の活用と課題』
    - 旦木 秀和氏(ソニー)

 JEITA

SystemCユーザ・フォーラム2007開催の意図

- 本年度のSystemCユーザ・フォーラムは、SystemCの普及を目的として以下の方針の下実施された。
  - SystemCユーザに対して、OSCIの活動状況とその方針を伝える。
  - JEITA SystemCタスクグループの活動をユーザへ報告する。
  - TLM動向と実設計における事例を紹介し、ユーザへ普及の現状を伝える。

 JEITA

SystemCユーザ・フォーラム2007サマリー

- 昨年より聴講者はやや減少し実参加者数148名とほぼ目標としていた160名に近いものの未達となり、聴講者数の増員は次回の課題として残った。
- 本年はユーザの関心が高い『トランザクション・レベル・モデリング（TLM）』と『合成』を中心とした構成。
- 講演内容：
  - OSCIからはPatrick Sheridan氏を向かえ、主にTLM2.0の開発状況やロードマップについて講演を行った。
  - SystemCタスクグループでは、国内外におけるTLM使用状況の調査結果発表と、OSCI合成サブセットのレビュー結果と合成スタイルガイドの作業状況について解説した。
  - STARCからは武井氏を招いて現在のTLM標準化動向を中心に、TLM普及とSTARCの活動について発表を頂いた。
  - ユーザ事例はソニーの旦木氏より、ソニー社内における動作合成の現状とその取り組み、また今後の普及の為の課題を発表頂き、SystemC活用に対する貴重な情報を伝える事が出来た。

 JEITA

SystemCユーザ・フォーラム2007サマリー

- フォーラム全体を通してほぼ予定通りの時間配分で進み、また質疑応答時間内も多くの質問がユーザから上がった。
- 昨年は予稿集と当日スライドとの差異を指摘されたが、本年は差異が小さく、またOSCIの英文プレゼン資料は日本語版を作成して掲載した事で参加者にとって良かったように思う。
- またアンケートから意見として次のようなものがあった。
  - 入場券の半券を残して欲しい
  - 会場前の待機場所の設置
  - 発表中にフラッシュを使った写真撮映を止めてほしい
  - 事例紹介を増やして欲しい
  - TLMの技術的内容、TLMユーザ事例を聞きたい
  - 日本国内における標準化活動スケジュールの明確化
  - パネル討論会

 JEITA

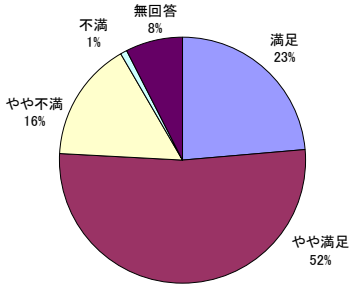
アンケート調査集計結果

- 昨年度とほぼ同様の内容でアンケートを実施し、昨年度までの結果と合わせて聴講者の動向を分析。
  - SystemCユーザは着実であるが増加している。
  - 今後の標準化活動や動向に多くのユーザが注目している。
  - 高位（動作）合成の興味が依然高く、ユーザの興味は使う上でのインフラに移行しつつある。
  - TLMの興味は非常に伸びており、TLM2.0が注目されていると考えられる。
  - 有料であっても関係なく、参加者は内容を重視している。

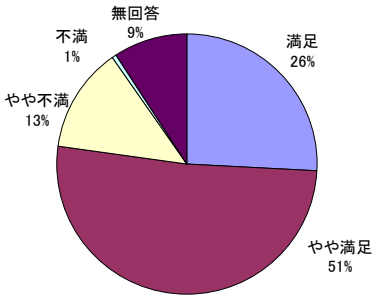
 JEITA

0. 本セッションの内容について

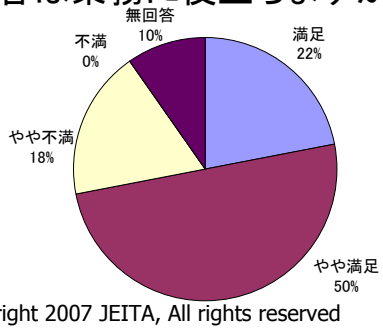
■内容は期待通りでしたか。



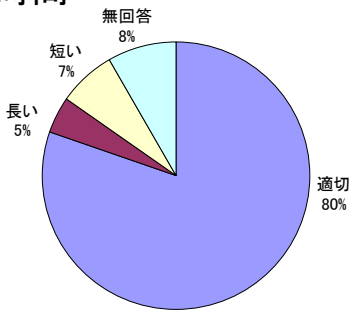
■予稿集の内容は期待通りでしたか



■内容は業務に役立ちますか

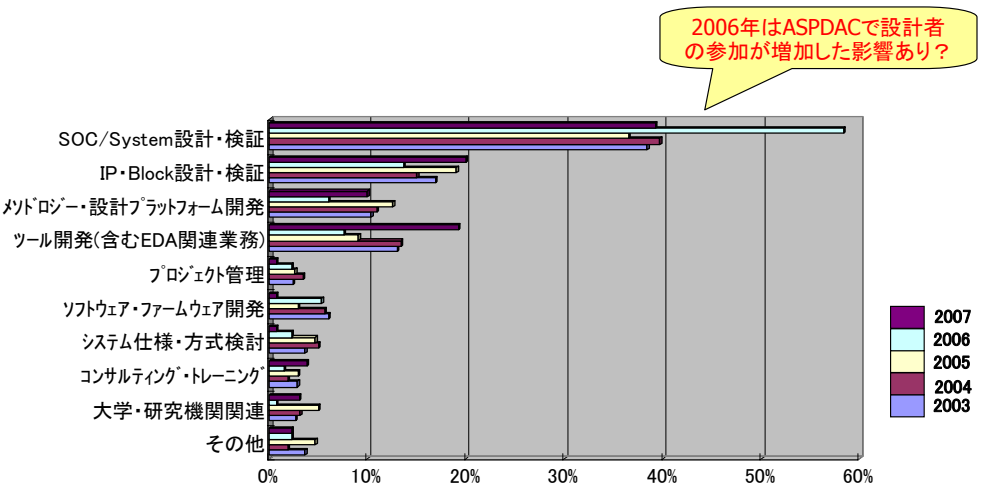


■講演時間



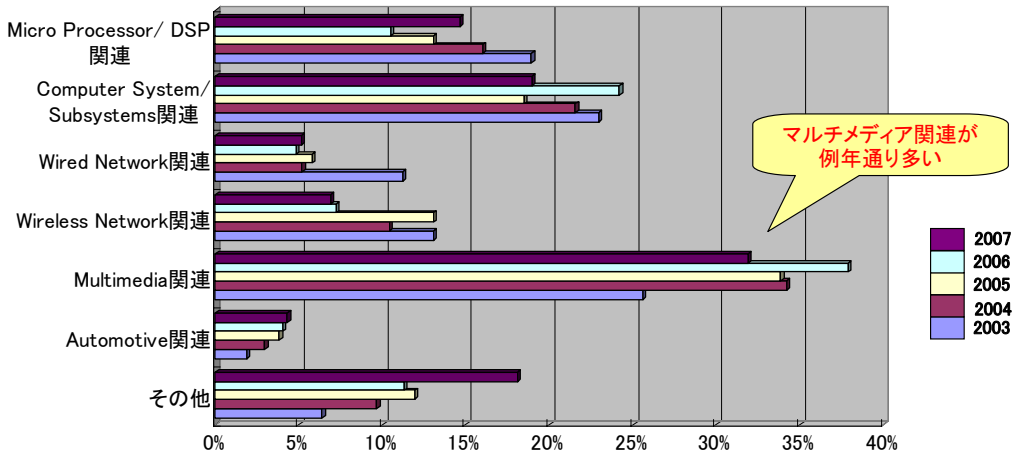
JEITA

1. ご担当業務またはビジネスは?



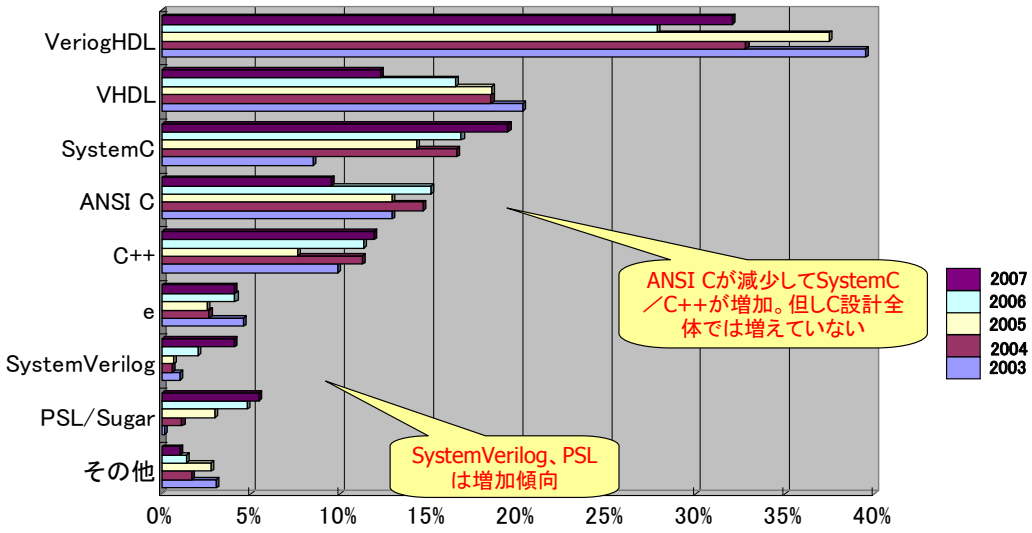
JEITA

2. ご担当製品アプリケーションは?



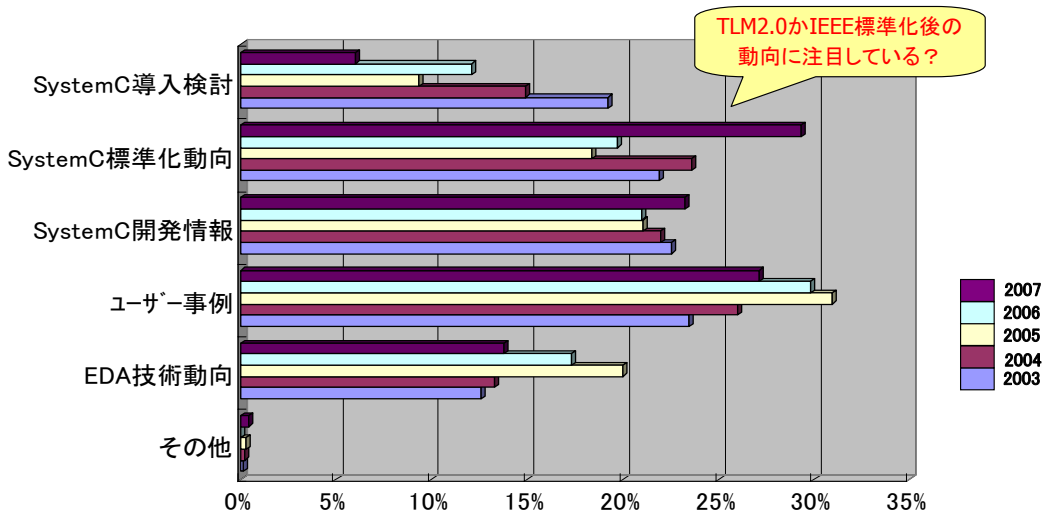
 JEITA

3. 現在主に使用している言語は?



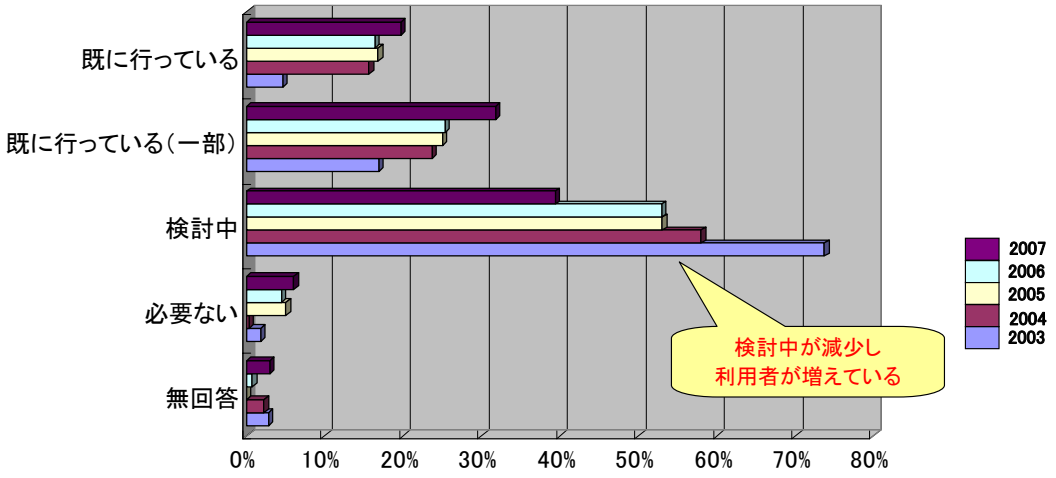
 JEITA

4. SystemCユーザ・フォーラムに参加された目的は?



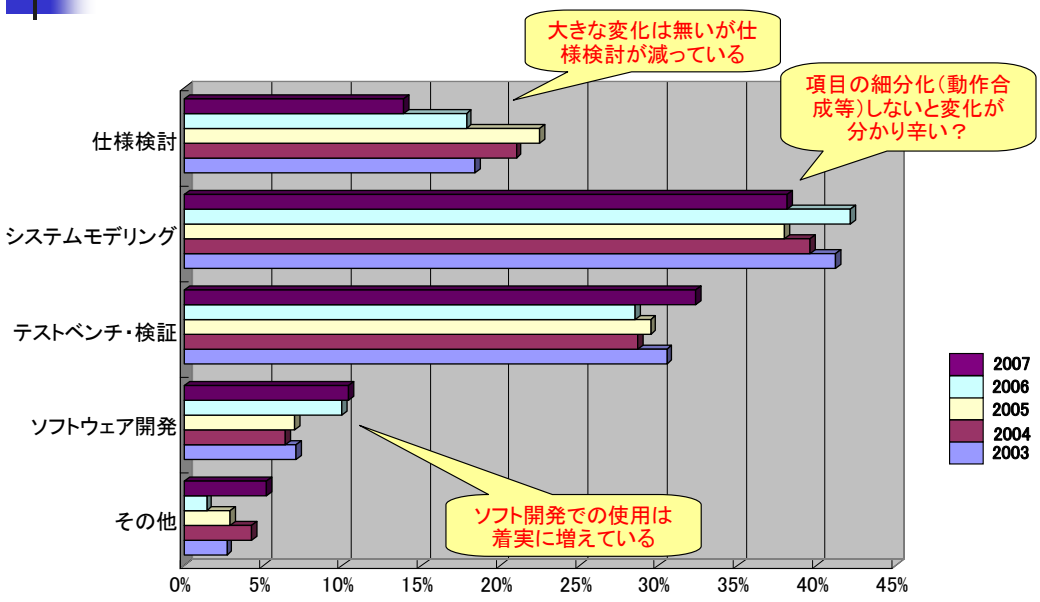
 JEITA

5. SystemCでの設計･検証環境構築について



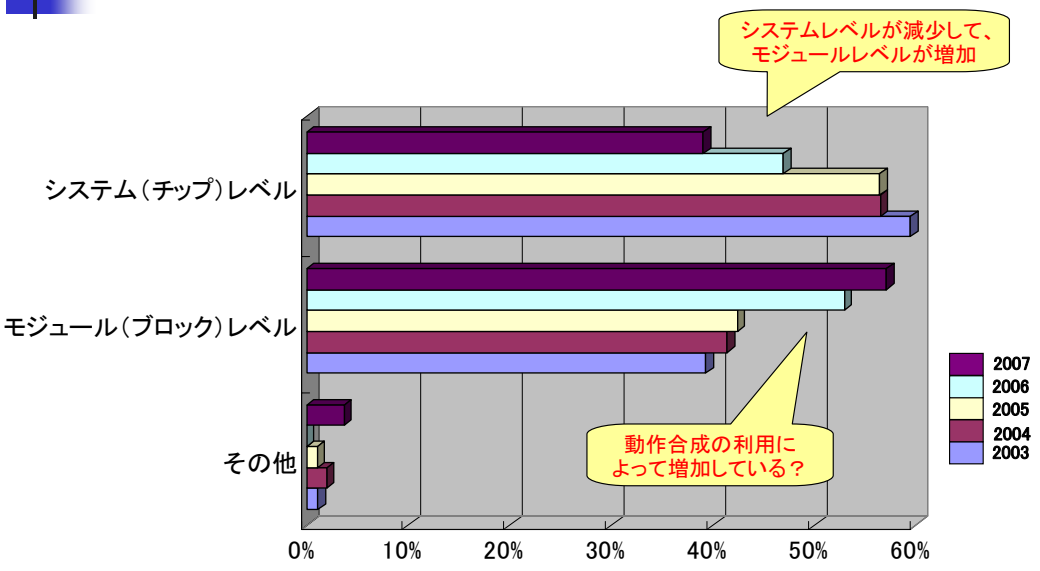
 JEITA

6.「5」で「既に行っている」または「検討中」と回答された方
a) SystemCの使用目的は?



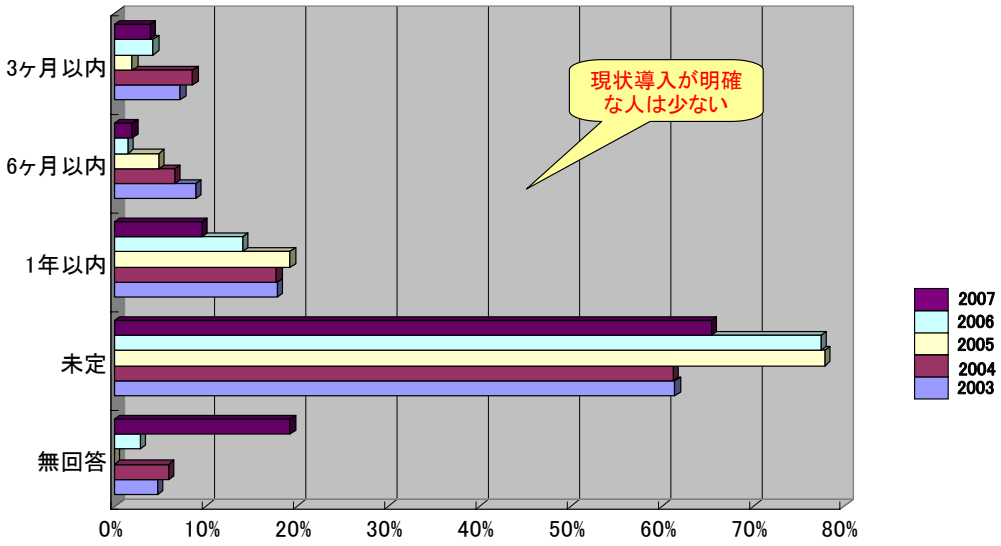
 JEITA

b) SystemCの活用範囲は?



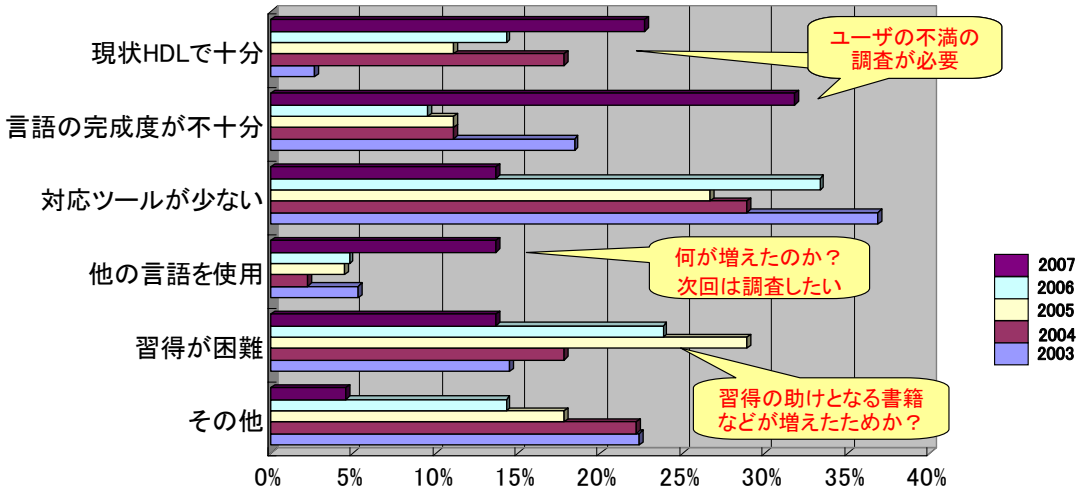
 JEITA

7.「5」で「検討中」と回答された方へ 導入予定時期は?



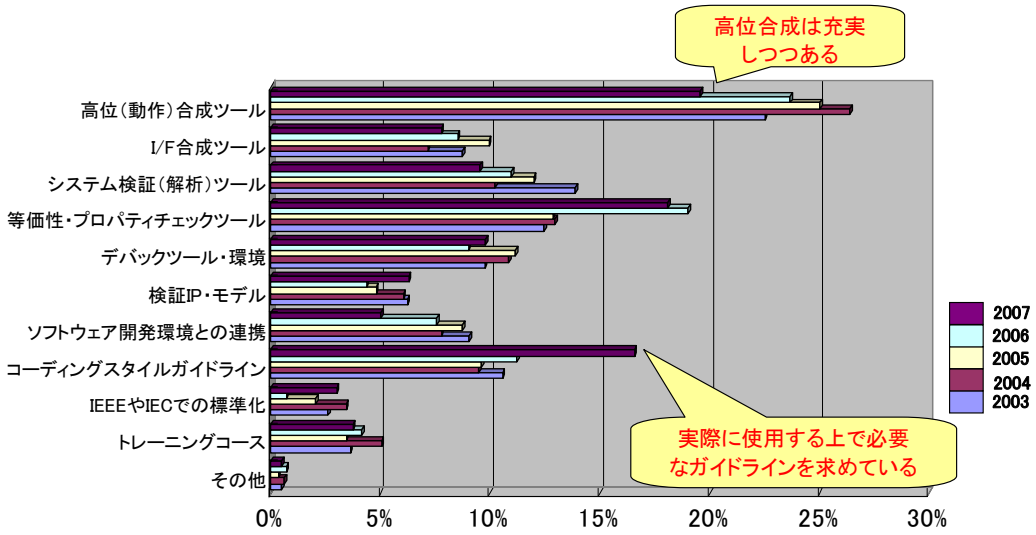
 JEITA

8.「5」で「必要ない」、「検討中」と回答された方へ 導入の障害となっている理由は?



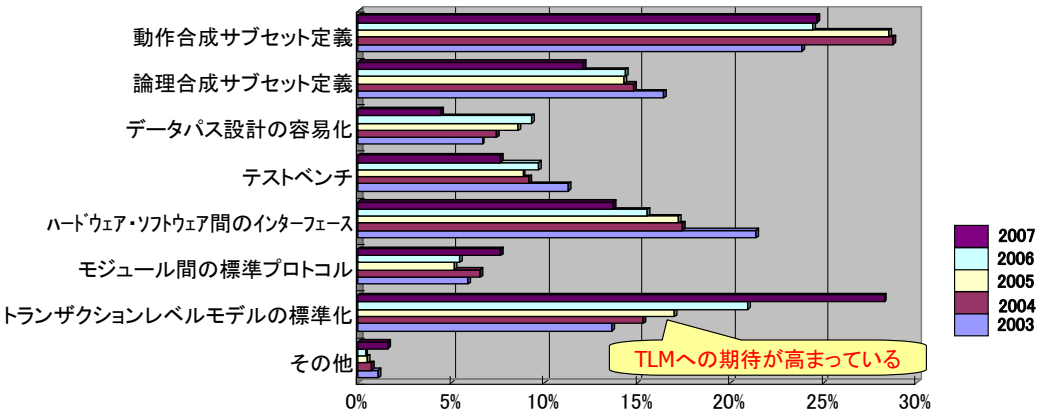
 JEITA

9. SystemCをより活用する為に充実が必要なものは？



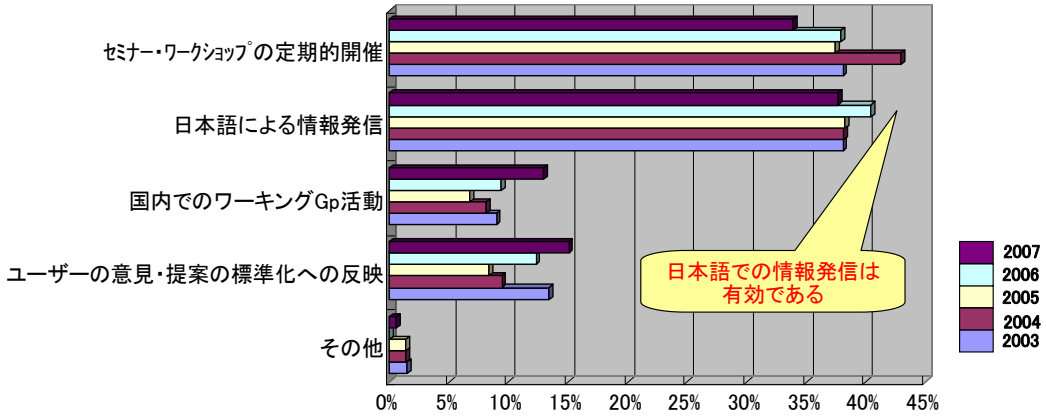
 JEITA

11. 今後SystemCの言語拡張･標準化で期待することは?



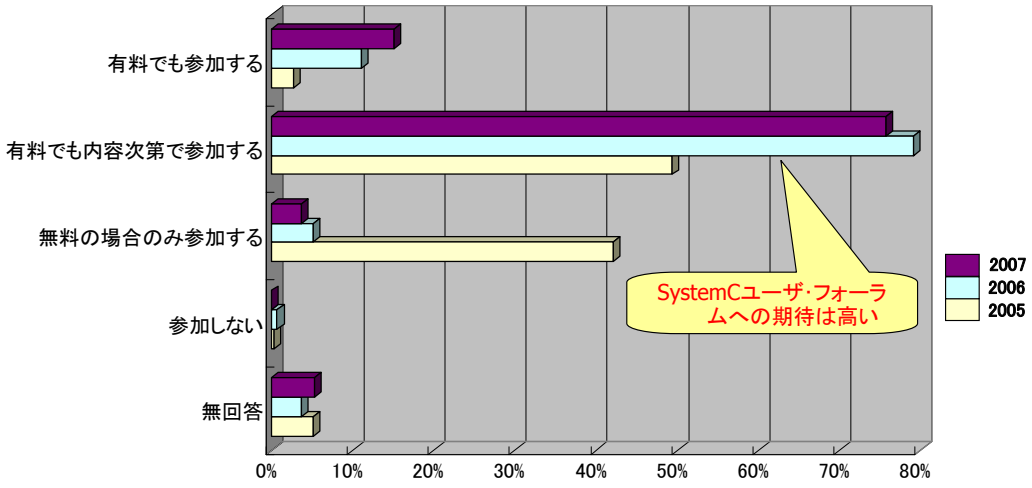
 JEITA

12. 今後Open SystemC Initiative（OSCI）やJEITAに期待する活動は？



 JEITA

13.有料開催についてはどう思われますか？



 JEITA

4.2.3 TLMサブグループ活動報告

TLM活動結果

- TLM動向調査
  - TLM技術に関する書籍「TRANSACTION-LEVEL MODELING WITH SYSTEMC」(2006年)の内容調査。
  - ECSI Institute Workshop on Users Experience with TLM (2006年6月開催)での発表内容調査。
  - JEITA参加企業内でのTLM利用状況を調査し、海外での利用状況との差異をまとめた。
- TLM標準化活動
  - OSCIから公開されたTLM2.0 Public Review版へのフィードバックを実施。

4.2.3.1 ECSI Workshop 2006
調査まとめ

ECSI : European Electronic Chips & Systems Design Initiative
http://www.ecsi.org/

JEITA

調査内容一覧

- 1: Brief Introduction to TLM
- 2.1: OSCI SystemC TLM
- 2.2: SPIRIT for TLM
- 2.3: OCP Modelling in TLM
- 3.1: STMicroelectronics Modelling Experience
- 3.2: FireWire and USB IP Blocks Modelled with SystemC TLM
- 3.3: TLM Modelling Experiences for Early Analysis in the ICODES Project
- 3.4: New Silicomp Activities and Services in SystemC/TLM
- 3.5: System Validation Based on TLM Models
- 4.1: Performance Estimation with SystemC TLM
- 4.2: Cycle-Accurate TLM Using the ARM RealView ESL APIs
- 4.3: Timing, Analysis, and the OSCI TLM
- 4.4: Performance Analysis & Automated Refinement of TL SystemC Models
- 4.5: TLM Peripheral Modeling for Platform-driven ESL Design
- 5.1: Separating Functional and Timed Aspects in Transactional Abstraction Level
- 5.2: OSSS-Channels: Synthesisable TLM Concepts
- 5.3: Software (RTOS) Modeling in TLM
- 5.4: TLM Behavioural Modelling for NOC (Network on Chip) Architectures
- 6.1: Summit Design Vista
- 6.2: VaST Systems Technology
- 6.3: SpiraTech
- 6.4: CoWare
- [参考1]：Pinnapa
- [参考2]：ARM RealView ESL APIs
- [参考3]：SpiraTechのTLMサンプル例

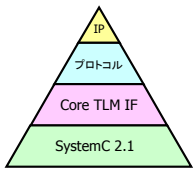
JEITA

1: Brief Introduction to TLM

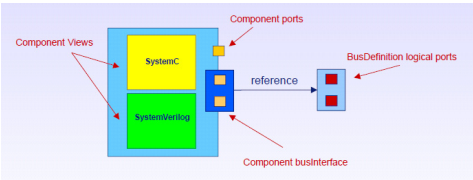
タイトル	Introduction to TLM
著者	Laurent Maillet-Contoz
所属	System Platforms Group, Crolles, STMicroelectronics
概要	TLMの役割を解説
内容	■TLMの役割 ●機能検証のリファレンスモデル ●SW設計者のためのVirtual Prototype ●アーキテクトのためのアーキテクチャモデル ■TLMがサポートする抽象度 ●bit-true、サイクル精度モデル ～ アンタイムドアーキテクチャモデル ●PV（post HW/SW分割、bit-true動作、レジスタバンク、データ転送、システム同期をモデル化） ●PVT（時間情報、タイムドIPモデル、リファインド通信モデル） ■SystemCを採用する理由 ●インダストリ標準な言語 ●HWとSW両方をモデリング可能（並列性、反応性、モジュール、タイミング、データ型） ●C++ベースのオブジェクト指向アプローチ ●シミュレーションカーネルにHWプリミティブが組み込まれている ●様々な抽象度をサポート 出展：ECSI Workshop 8 June 2006 Proceeding

2.1: OSCI SystemC TLM

タイトル	OSCI SystemC TLM Evolution
著者	Laurent Maillet-Contoz
所属	System Platforms Group, Crolles, STMicroelectronics
概要	OSCI TLM 1.0を使ったプロトコル定義方法とOSCI TLM SWGでの検討内容
内容	■TLM 1.0を使ったTLMプロトコル定義の手順 1.適切なCore TLM IFを選ぶ。（双方向IF or 単方向IF） 2.トランザクションペイロードを定義する。（データ、アドレス、ステータス、・・・） 3.プロトコルIFを定義する。（read、write、・・・） 4.通信ポートを定義する。（イニシエータポート、ターゲットポート） ■OSCI TLM SWGでの検討内容 ●TLM 1.0のIEEE標準化 ●スタンダードバスAPI（Generic PV、Generic PVT、割込み、メモリマップ、・・・） ●スタンダードコンフィギュレーション・制御API（コンフィギュレーションIF、デバッグIF、モニタIF） ●ペイロード（PV：トランザクションの抽象モデルのため、PVT：通信の性能モデリング・パイプライン動作を考慮） ●タイムドモデリングのためのCore IFのアップデート（sc_timeパラメータの追加） ■TLM 2.0：モデルのInteroperabilityへ向けて ●合意が必要な項目：Core TLM IFの選択、トランザクションペイロードの定義（リクエスト：アドレス・データ・・・、レスポンス：ステータス・データ・・・） ●推薦する項目：プロトコルIF、イニシエータポートとターゲットポートの実装（Core TLM IFとペイロードに合った） 出展：ECSI Workshop 8 June 2006 Proceeding



2.2: SPIRIT for TLM

タイトル	The SPIRIT Consortium Meta-Data for the Multi-Abstraction Design Flow
著者	Jean-Michel Fernandez
所属	Cadence Design Systems
概要	SPIRITコンソーシアムのアップデート情報
内容	■ SPIRITコンソーシアムの概要 ■ コンソーシアムの紹介 ■ SPIRITの概要 ■ 標準化動向 ■ SPIRIT1.4の概要 ■ 1.4のモデリング概要 ■ 1.4で予定される提供物 ■ SPIRIT1.4ロードマップ  出展：ECSI Workshop 8 June 2006 Proceeding

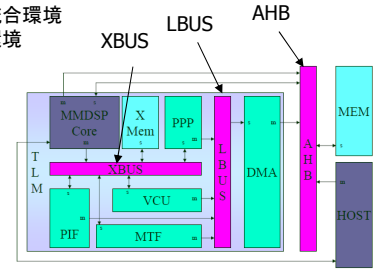
JEITA

2.3: OCP Modelling in TLM

タイトル	OCP-IP provides the standard TLM Socket
著者	Mark Burton
所属	GreenSocs
概要	OCP-IPとOCPにおけるTLMモデリング概要
内容	■ OPC-IP概要 ■ 標準化の重要性 ■ 標準化の現状 →各団体との繋がり。 ■ OCP-IP2.1.2モデリングキット ■ 活動動向 ■ TLMモデリング概要 →OCPにおけるTLM抽象度 ■ モデリングキット概要 ■ OCP-IP SLD WGロードマップ ■ インターオペラビリティに着目した標準化 ■ モデリングキットの技術的向上  出展：ECSI Workshop 8 June 2006 Proceeding

JEITA

3.1: STMicroelectronics Modelling Experience

タイトル	STMicroelectronics Modeling Experience
著者	Andrea Battistella
所属	STMicroelectronics
概要	マルチメディア用SOCの一部をTLMで検証
内容	▪ Video Smart Accelerator部でTLM検証を実施 ▪ 三つの大きな目的 • ファームウェア開発チームのための開発環境 • 検証エンジニアのためのHW/SF/FW統合環境 • システム・アーキテクトのための評価環境 ▪ 結果 TLMプラットフォームは、テープアウトの前後で、ビデオ用ファームウェアの開発に実際使われた。すぐに使用可能になったこと、シミュレーションの速さ、デバッグ効率の良さにより、サブシステムの開発期間を劇的に減らせた  出展：ECSI Workshop 8 June 2006 Proceeding

3.2: FireWire and USB IP Blocks Modelled with SystemC TLM

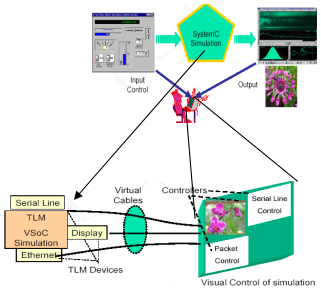
タイトル	FireWire and USB IP Blocks Modelled with SystemC TLM
著者	Ireneusz Sobanski, Adam Bitniok
所属	Evatronix
概要	IPコア開発におけるSystemC-TLMの利用
内容	▪ なぜSystemCとTLMモデルが必要か • 今日、アプリケーションがいっそうソフトウェアに関連するため ➢ IPコアは、ハードウェアのコントローラからソフトウェアのドライバまでの範囲にわたる、広いソリューションとともに提供されるべき ➢ ハード屋とソフト屋という両極端の人々が協力できるメソドロジが必要 • SystemCの利点 ➢ ハード開発屋とソフト開発屋の共通の基盤 ➢ ソフトウェアの検証を、ハードウェア試作ができるはるか前から開始できる ▪ 結論 ・今のTLMスタンダードは、内部で使用する分には問題ない ・他の会社へ TLMのIPを供給する場合に問題がある ▹ どのようにTLMを書くか、スタンダードもガイドラインもない ▹ 内部モジュール間のインターフェースをどう書くべきか ▹ 外部とのインターフェースをどう書くべきか ▹ 時間に依存する部分をどうモデル化するべきか 出展：ECSI Workshop 8 June 2006 Proceeding

3.3: TLM Modelling Experiences for Early Analysis in the ICODES Project

タイトル	TLM Modelling Experiences for Early Analysis in the ICODES Project
著者	Francesca Tonetta
所属	Siemens
概要	エンベデッド・システムのモデリングと合成にTLMを応用する
内容	・ **I**nterface and **CO**mmunication based **D**esign of **E**mbedded **S**ystems (ICODES) http://icodes.offis.de/ ・エンベデッド・システム(ハード/ソフト)用のモデリングと合成のメソドロジを開発する目的のプロジェクト ・使用言語：OSCI TLM SystemC ・ 結論 ・TLMモデリングによって期待される効果 ▹表現性 ▹再利用性 ▹コンパクトなコード ▹バイアスのかからない記述 出展：ECSI Workshop 8 June 2006 Proceeding

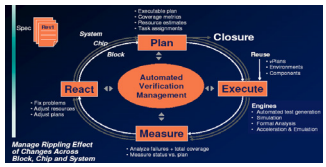
JEITA

3.4: New Silicomp Activities and Services in SystemC/TLM

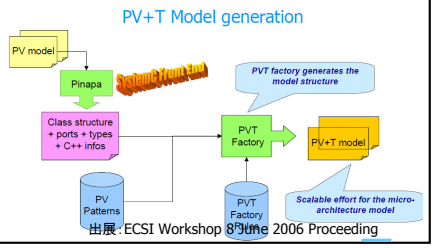
タイトル	New Silicomp Activities and Services in SystemC/TLM
著者	Vania Joloboff
所属	Silicomp
概要	Silicompが提供する仮想ダッシュボードのSystemCを用いたソリューションの紹介
内容	■SilicompのTLMモデルによる設計活動 ・対象アプリケーション 計器関連 ：ダッシュボード、フライトシミュレータ (Cockpit visual application) ・各抽象度で行う設計活動と課題、提供するソリューション 設計活動：HW/SW検証、HW検証、SW検証 課題：現在シミュレーション実行後、結果を解析。 →リアルタイムな検証のニーズ。 提供するソリューション： TLM Virtual Dash Board → リアルタイムにシミュレーション解析できる。 → 同じシナリオを何度でもレプレイできる。 データを可視化して解析。 → TLM Virtual Dash Boardのコンテンツ I/Oコネクタ(SystemC I/F)、I/Oデータのプロトコル、I/Oデータをドライブまたは解析するコントローラI/F、visual applicationにコントローラを実装するためのサーバーフレームワーク。  出展：ECSI Workshop 8 June 2006 Proceeding

JEITA

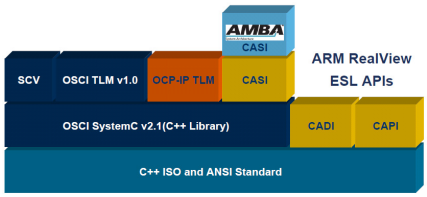
3.5: System Validation Based on TLM Models

タイトル	System Validation Based on TLM Models
著者	Jean-Yves Pateyron
所属	Cadence
概要	Cadenceが提供するTLMベースのシステム検証ソリューションの紹介
内容	■CadenceのTLMモデルによる設計抽象度と設計活動 ・HW設計/検証ソリューションを提供。 ・抽象度の定義とモデル手法 ：バス、Comunicationモデルの抽象度 → PV, PVT, CA, RT ・各抽象度で行う設計活動と課題、提供するソリューション 設計活動: HW検証 提供するソリューション: ・TLMデバッグ環境→ エラーメッセージ、ソースコードビューワー、ラインデバッグ、スレッド ・SCVのトランザクションの記録 SCV Transaction Recording Callback (scv_stream,scv_tr_generator,scv_tr_handle)を提供。 Comparescan-TX を用いてPV、PVT、CA間の比較が可能。 ・NC-SCプロパティチェック ：PSLアサーションが使用可能。 ・NC-SC CVE 機能カバレッジ ： scv_coverpoint, scv_covercross, scv_covergroupをサポート。 ■Cadenceの検証メソドロジ ・Verification Process Automation Plan→Execute→Measure→Reactを循環させる環境。 Plan: 仕様をTLMシミュレーション実行可能なモデルで作成。 TLMをリファレンスとした検証フロー。 仕様書の電子化(vPlanを活用)。 Measure:カバレッジ結果はvPlanにまとめる。  出展:ECSI Workshop 8 June 2006 Proceeding

4.1: Performance Estimation with SystemC TLM

タイトル	Performance Estimation with SystemC TLM
著者	Laurent Maillet-Contoz
所属	ST Micro, System Platforms Group Crolles(France)
概要	PVからPV+Tの生成フロー及び性能推定ツールの紹介
内容	STでは、SystemCのPV+Tをアーキ性能解析と組み込みSWの最適化に活用している。 PV+TはPV（機能）とT（タイミング）に分けてモデル化している。 PV+Tモデルの生成フローは下記の図 ・Pinapa[参考1]というフリーのSystemCパーサーで ポートや各種情報を収集 ・PVパターンとPVT生成ルール に従って PVTの雛形を生成 (PVT Factory) また、性能解析ツールとしてIPTG解析GUIツールを利用している TLM/RTL/BCAを扱える SpiritとPlatform Builder(?)に対応  出展:ECSI Workshop 8 June 2006 Proceeding

4.2: Cycle-Accurate TLM Using the ARM RealView ESL APIs

タイトル	RealViewESL APIs
著者	Nizar ROMDHANE
所属	ARM, ESL Technical Marketing Manager, DSMK
概要	ARM RealViewでのESLツールとのCycle-AccurateのAPIの説明
内容	ARM RealViewで一般公開しているCycle-AccurateのAPIとして ・CASI:サイクルベースのシミュレーション用API ・CADI:SWデバッガとの接続用API ・CAPI:プロファイリング用API(ラベル情報と生データを分けて収集) ・AXI/AHB/APBサポート OSCI TLMとは互換性なし(TLM PVとのアダプタは2006/10リリース?) 現在OSCIにCASIを提案中? 2006年10月にはOSCI TLM へ組み込むための技術 第一弾ができる(未確認) 06年8月Mentor/Seamless に組み込み済 ※[参考2]  出展:ECSI Workshop 8 June 2006 Proceeding

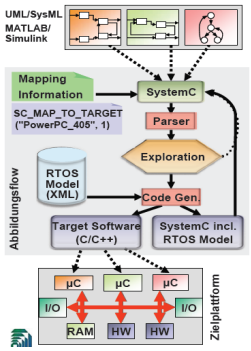
JEITA

4.3: Timing, Analysis, and the OSCI TLM

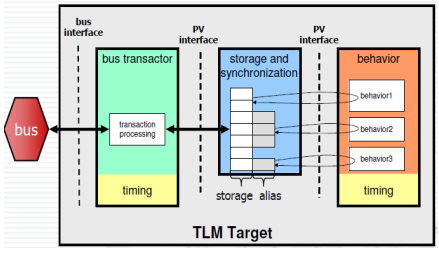
タイトル	TLM WG Status (Personal View)
著者	Adam Rose
所属	Mentor Graphics
概要	OSCI TLM WGでのディスカッション内容の紹介
内容	検討中の主な課題 1) IEEEにおけるTLM 1.0の標準化 2) 標準バスAPIの定義 Generic PV Generic PVT Interrupt Modeling Memory Map Services Memory / Register Modeling 3) Configuration、Control用のAPIの追加 Configuration Interface Debug Interface Analysis Interface 4) その他 non-blocking APIに対する時間パラメタの追加 大規模データにおけるポインタ利用ガイドラインの検討 Timescaleの扱いの検討 出展:ECSI Workshop 8 June 2006 Proceeding

JEITA

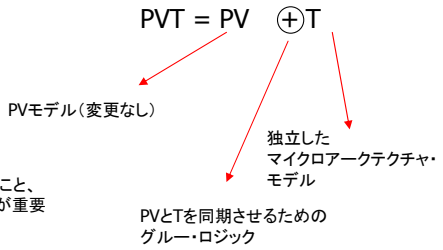
4.4: Performance Analysis & Automated Refinement of TL SystemC Models

タイトル	Performance Analysis and Automated Refinement of Transaction-Level SystemC Models
著者	Olivier Bringmann
所属	Forschungszentrum Informatik
概要	システムアーキテクチャ探索手法及び、自動詳細化フローの紹介
内容	■プラットフォームベース設計 • プラットフォーム非依存の組み込みSW開発 • バーチャルプロトタイプを利用した、初期システム検討実施 • UML/SysML, MATLAB/Simulink等、既存技術との連携 • 自動コード生成(SW/HW)による、シームレスなモデル変換 • XML SPIRIT / FIBEXを利用したSystemC TLMモデルの生成 ■TLMによる性能解析 • 自動化されたボトルネック解析、システム性能解析 • トランザクタ、自動コード生成を利用した、詳細化フロー • 複数の抽象度のTLMモデルの利用 ■組み込みSWの詳細化フロー • HW依存のSWを、プラットフォーム非依存環境で開発 •SystemC RTOSモデルの提供 • VirtualOS/ RTOS / middlewareを含むSW性能解析の実施 • アーキテクチャ情報のマッピングによるターゲットSWの自動生成  出展:ECSI Workshop 8 June 2006 Proceeding

4.5: TLM Peripheral Modeling for Platform-driven ESL Design

タイトル	TLM Peripheral Modeling for Platform-driven ESL design
著者	Serge Goossens
所属	CoWare
概要	SCMLによるペリフェラルモデリング手法の紹介
内容	• 目的 • 標準仕様ベースのTLMメソドロジの確立 • ビヘイビアモデルの再利用性向上 • モデリングの容易化 • シミュレーションの高速化 • 特徴 • 通信部とビヘイビアの分離 • 自動アドレスデコード機能を備えたstorage objectを提供 • 複雑な機能はstorage objectアクセスに対するcallbackとして定義 • タイミングの局所化(機能部またはトランザクタ内に定義)  出展:ECSI Workshop 8 June 2006 Proceeding

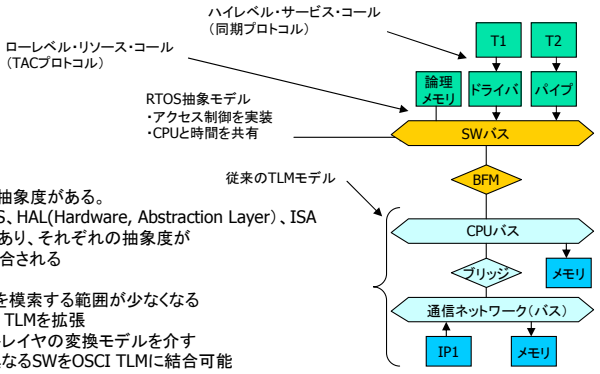
5.1: Separating Functional and Timed Aspects in Transactional Abstraction Level

タイトル	Separating Functional and Timed Aspects in Transactional Abstraction Levels
著者	Florence Maraninchi, Jerome Cornet, Laurent Maillet-Contoz
所属	Verimag, CNRS, STMicroelectronics
概要	TLMによる機能と通信の分離について紹介
内容	■TLMの特徴 ●RTLより早い磁気に利用可能 ●高速シミュレーション ●モデリング工数が軽い ■TLM目的 ●組み込みソフトウェア開発 ●仮想プロトタイプ ●IP検証 ●アーキテクチャ評価 ●組み込みソフトウェアの最終調整 ■TLMの目的に応じてPV/PVTを使い分けること、PVの機能部分がそのままPVTで使えることが重要  出展：ECSI Workshop 8 June 2006 Proceeding

5.2: OSSS-Channels: Synthesisable TLM Concepts

タイトル	OSSS-Channel:Synthesisable TLM concepts
著者	Cornelia Grabbe
所属	OFFIS Research Institute
概要	OSSSを用いたアーキテクチャ検証と合成フローの紹介
内容	■OSSS(Oldenburg System Synthesis Subset) ■C++のサブセット ■SystemCのサブセット ■＋OSSSの独自ライブラリ ■アプリケーション・レイヤ ●ポートを介した時間の概念のないデータの受け渡し ●アプリケーションの検証 ●仮想ターゲット・アーキテクチャ・レイヤ ●Osss_channelライブラリによりアーキテクチャにマッピング ●通信プロトコルを含めたシステム検証 ●インターフェース合成 ●仮想ターゲット・アーキテクチャ・レイヤにマッピングされたシステムをピン・レベルのRTLに合成 出展：ECSI Workshop 8 June 2006 Proceeding

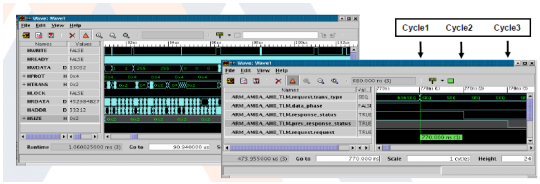
5.3: Software (RTOS) Modeling in TLM

タイトル	Embedded SW Modeling at TLM
著者	Aimen Bouchhima
所属	TIMA Laboratory
概要	抽象度の高いソフトウェアを用いたシステム検証の提案
内容	 ■HWと同じようにSWにも抽象度がある。 ■Functional、COM、OS、HAL(Hardware, Abstraction Layer)、ISA ■HW/SWともに抽象度があり、それぞれの抽象度が落ちたところではじめて統合される ■統合検証が遅くなる ■最適なデザイン空間を模索する範囲が少なくなる ■SWモデリング用にOSCI TLMを拡張 ■SWをレイヤ構成とし、各レイヤの変換モデルを介することによって抽象度の異なるSWをOSCI TLMに結合可能 出展：ECSI Workshop 8 June 2006 Proceeding

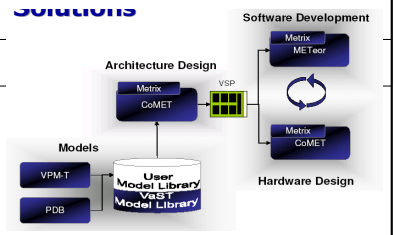
5.4: TLM Behavioural Modelling for NOC (Network on Chip) Architectures

タイトル	TLM Modeling for Network-On-Chip Architectures
著者	CEA-LETI/DCIS/SCME/LIAN
所属	Pascal Vivet
概要	ANOC※アーキテクチャへのTLM適用
内容	※ANOC：Asynchronous Network On Chip ▪ バスを基本とするアーキテクチャ ▪問題点：パフォーマンスに限界がある ▪ NOCアーキテクチャ ▪バスバンド幅、チップパフォーマンス、アーキテクチャのスケーラビリティ、構成の柔軟性 ▪ ANOCアーキテクチャ ▪低電力化、通信にかかわる 領域だけクロックを供給 時代の流れ ANOCをベースとするシステムを評価する手段として OSCI/TLMを採用 ⇒ 選定理由 ▪ OSCIにより標準化されている ▪ 様々な立場の人が利用できる ▪ Systemアーキテクト：アーキテクチャ検証 ▪ HWエンジニア：SWを考慮したHW論理検証 ▪ SWエンジニア：ソフト検証（高速なHWが必要） ▪ 様々な抽象度で検証環境を変更不要 ▪ TLM NOC node(完全な動作モデル) ▪ ハンドシェイクレベル(sc_fifo) ▪ 実装レベル(ネットリスト) 出展：ECSI Workshop 8 June 2006 Proceeding

6.1: Summit Design Vista

タイトル	Vista Effective SystemC Design Flows
著者	Zvika Amir
所属	Summit Design, Inc.
概要	VistaによるSystemCデザインフローの紹介
内容	■ Summit Vistaの概要 ■ Summit Vistaの特長 ■ OCP-IP TL2/TL1による例題 ■ マスタ/スレーブ/OCPプロトコル ■ 各機能またはウィンドウの説明 ■ 他社ツールとのコンパチビリティ  出展：ECSI Workshop 8 June 2006 Proceeding

6.2: VaST Systems Technology

タイトル	VaST Systems Technology
著者	Jean-Marc TALBOT
所属	VaST
概要	VaSTが提供するHW/SW協調設計環境の紹介
内容	 ■VaSTの設計抽象度と設計活動 ・対象アプリケーション ワイヤレス、民生機器、自動車関連 ・抽象度の定義とモデル手法 -バス、Comunicationモデルの抽象度 → CA -SW/HWモデルのモデルの抽象度 → ISS ・各抽象度で行う設計活動と課題、提供するソリューション 設計活動： HW/SW検証、アーキテクチャ探索 提供するソリューション： ・Virtual System Prototype(VSP) and SW Development Environment → HW/SW協調検証、早期のSW開発環境。 → VaSTはLibrary（CAレベルのプロセッサーモデル、バスモデル）を提供、ユーザーはこれを用いて アーキテクチャ検討、HW設計、SW開発を行う。ユーザー独自ブロックはユーザーがC/SystemCで記述。 → SW IDEコンパイラーの3rdパーティのサポート,デバッグ機能,Co-Sim I/Fを提供。 ・適応事例 ： Infineon GSM/GPRS/EDGE baseband IC （ARM、DSP、70 peripherals...） → SWの早期開発、アーキテクチャ探索、RTL評価 UMTS baseband Chip → 従来フローに比べ5ヶ月削減。 SystemCユーザーペリフェラルモデル、VaST プロセッサー/バスモデル、 ARMベースプラットフォーム提供。 出展：ECSI Workshop 8 June 2006 Proceeding

6.3: SpiraTech

タイトル	SpiraTech TLM
著者	Kamal Hashmi
所属	SpiraTech
概要	トランザクタの意味/役割りとAXIによる紹介
内容	**TLMの意味、メリット** **▪様々なレベル(TLM-RTL/Sim-Emulation)間の通信が可能** **▪高いレベルのテストが再利用可能** **▪全てのレベルでプロトコル検証が可能** **▪全てのレベルでデバッグが可能** TLMの役割り **もしI/Fがデータだけしか管理しないと、各ユニット側で余分な情報を持つ必要が出る。I/Fのコミュニケーションに関する時間情報** **各ユニットはそれぞれのレベル(抽象度)に適した、出来るだけ多くの機能を実装するだけにすべき。またI/Fがタイミングをチェックすべき** SpiraTechのTLMサンプル例[参考3] 出展:ECSI Workshop 8 June 2006 Proceeding

6.4: CoWare

タイトル	CoWare solutions
著者	Xavier Buisson
所属	CoWare
概要	CoWareが提供するSystemC TLMメソドロジ
内容	要求 ▪ **簡単にモデリングができること** **▪便利な関数の提供** **▪自動化されたデバッグ・解析サポート** ▪ **高いパフォーマンス** **▪同期・イベントを減らしSimオーバーヘッドを低減するコーディングスタイル** **▪最適化されたSimエンジン** ▪ **様々な抽象度間での再利用が出来ること** **▪通信と機能の分離** **▪プラットフォーム構築におけるモデル選択の自動化** ⇒ これらを実現するツールを提供 出展:ECSI Workshop 8 June 2006 Proceeding

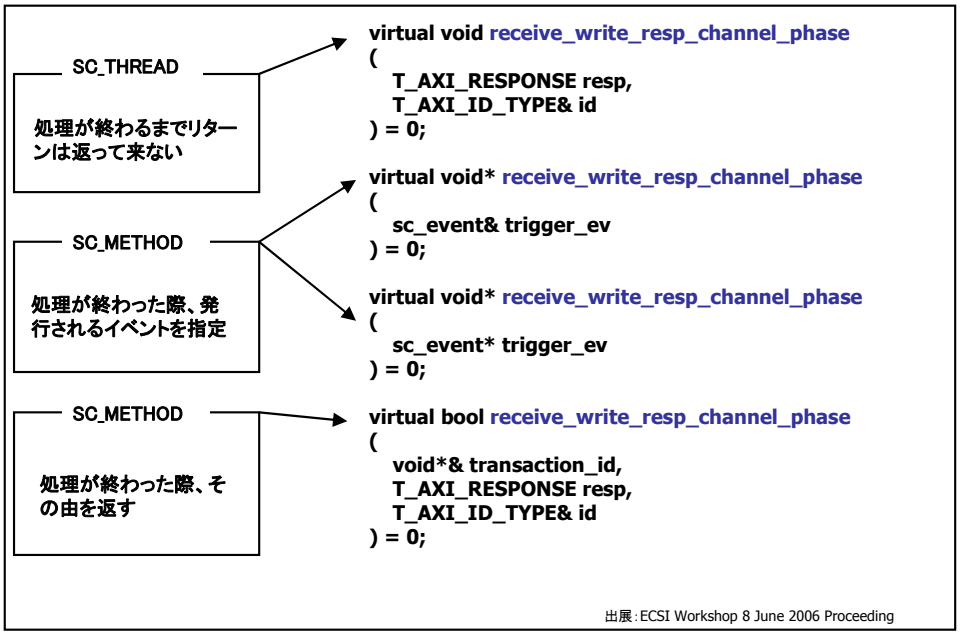
[参考1]：Pinnapa

- オープンソースのSystemCパーサー
  http://greensocs.sourceforge.net/pinapa/
- STとVERIMAGの成果物
  - VERIMAG：フランス Grenobleにある組み込みシステム研究機関
    (http://www-verimag.imag.fr/)
- GreenSoCsプロジェクトの一部

[参考2]：ARM RealView ESL APIs

- 3つのAPIを含む
  - CASI:サイクルベースのシミュレーションAPI
  - CADI:SWデバッガとの接続用API
  - CAPI:プロファイルデータ収集のためのAPI
- 下記URLで一般公開されている（登録必要）
  - http://www.arm.com/products/DevTools/RealViewESLAPIs.html
- 現在2006年8月にVer1.1リリース
- OSCI Simulator上で動くAXI/AHBサンプルコードあり

[参考3]：TLMサンプル例(SpiraTech)



JEITA
27

4.2.3.2 「TRANSACTION-LEVEL MODELING WITH SYSTEMC」の調査結果

Chapter 1　TLM: An Overview and Brief History

■　著者

Frank Ghenassia and Alain Clouard
STMicroelectronics, France

■　概要

大規模・複雑化する SoC を短期間に開発するためには、現在の設計フローを変革する必要がある。現在、システム設計と RTL 設計のギャップを埋める設計手法として TLM(Transaction Level Modeling)が注目されている。TLM で設計することによってソフトウェア設計者とハードウェア設計者が共通のリファレンスモデルをベースに協調して設計することが可能となり、開発期間の短縮及び設計品質の改善が可能である。本章では TLM の概要と簡単な歴史について説明する。

■　内容

□　従来の設計フローの三大課題

図 1 に従来の設計フローを示す。従来の設計フローはハードウェアとソフトウェアの協調検証をプロトタイプ完成後に実施していたため、後戻りロスが大きいなど下記の 3 つの課題があった。

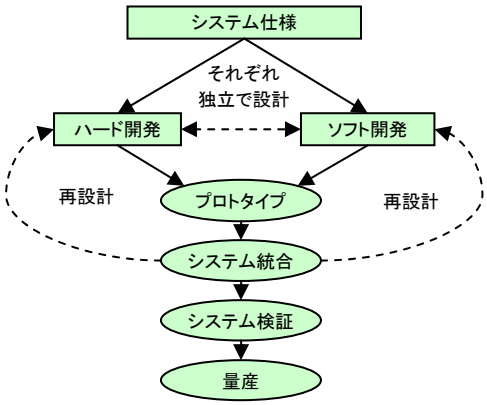


図 1　従来フロー

従来の設計フローの三大課題

- Time-to-Market：プロトタイプ完成後までシステム検証が実施できない
- コスト増：複雑化に伴い、再設計による設計工数の増加、マスク費増など、ナノメータ時代では、一発動作が必要不可欠

上記を解決するためには、設計の早い段階から RTL より高い抽象度で、ハードウェアを含めたソフトウェアの検証を実施する必要がある。この RTL より抽象度の高いレベル（＝トランザクションレベル）での設計手法を TLM(Transaction Level Modeling)と総称し、この TLM を活用した新しい設計フローの構築が必要不可欠である。

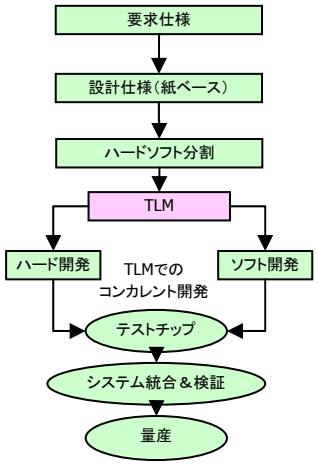


図 2　新設計フロー

□ 新しい設計フロー

新しい設計フローを図 2 に示す。新しいフローでは、従来の RTL 設計からインプリ設計のフローに加えて、システム設計から RTL 設計のフローが追加されている。要求仕様に従ってシステム設計者がハードウェアとソフトウェアを分割するとすぐにトランザクションレベルの開発環境(TLM プラットフォーム)を開発しハードウェアとソフトウェアの協調設計フェーズになる。TLM プラットフォームは、ソフトウェア設計者とシステム設計者との共通環境であり、検証担当者の検証環境開発にも役立つ。その後ハードウェア設計者は RTL を開発し、RTL プラットフォームが完成する。検証パターンは TLM プラットフォームのものを利用する。ハードウェアとソフトウェアをコンカレントに設計することにより SoC 全体の完成度が上がり、設計期間も短縮可能である。

新しい設計フローでは速度と精度のバランスを考慮して下記 3 つの抽象度を定義している。

- SoC 機能ビュー：もっとも抽象度が高く、システム動作をモデル化する。実装の詳細は意識しないため、アーキテクチャもアドレスマップ情報もない。性能は紙の仕様書で別途定義されている
- SoC アーキテクチャビュー：ハードウェアに依存したソフトウェアを開発・検討できるまで SoC 機能ビューを詳細化したもの。システムアーキテクチャ設計者が要求仕様に最適なアーキテクチャも決定する。さらにハードウェア検証者にとっては、リファレンスモデルとなる。
- SoC マイクロアーキテクチャビュー：SoC アーキテクチャビューに詳細な時間概念、サイクル精度を持たせたもので 2 つの役割がある、1 つは実チップや HW エミュレータ前に、組み込みソフトのデバッグと検証を行い、完成度を上げること。もう 1 つは、マイクロアーキテクチャの検証として、リアルタイム性や性能検証を行うことである。

□ TLM の歴史

1990 年代後半には、C ベース言語を拡張した SpecC、CoWareC や、Verilog の拡張である Superlog などモデリング言語が乱立した。ST 社では当初、RTL より一桁早いサイクル精度の C/C++モデルを作っていたが、下記の課題に直面した。

- RTL を作るのと同じ労力がかかる
- 当初の目論見速度より 10 倍遅かった(RTL が数 Hz に対して数 KHz 程度)
- 高速化のためにはツールに依存してしまう
- RTL の更新に追随するには時間が厳しい

これらの理由から、サイクル精度より抽象度が高く、モデル化が簡単だが、ソフト設計者が組み込みソフトを検証するに十分な精度と速度を確保でき、再利用のため統一されたモデリング言語を必要としていた。そこで 1999 年に CoWare 社と Synopsys 社と共同で C++ベースのモデリング言語 SystemC を提案した。2001 年にはトランザクション技術を H.263 ビデオコーデック開発に適用しその有効性を確認し、2005 年 4 月 21 日には SystemC の標準化団体 OSCI から TLM の 1st ドラフトが完成した。

■ まとめ

TLM を活用してソフトウェアとハードウェアの協調検証を早期に実現し、設計品質向上、設計期間の短縮を図ることができる。今後は、TLM から RTL への合成技術、TLM と RTL の等価検証技術が必要になってくる。さらに、ハードウェアとソフトウェアの最適分割などの協調設計に向けて、仕様から TLM への設計フローを確立していく必要がある。

Chapter 2　Transaction Level Modeling

■　著者

Laurent Maillet-Contoz and Frank Ghenassia

■　概要

トランザクションレベルモデリング(以下 TLM)は、SoC 設計フローにおいて RTL よりも上位に位置するソリューションとして提言されている。本章では、早期の SW 開発からアーキテクチャ解析、機能検証までの SoC 設計を活発に取組めるよう、アンタイムドとタイムドモデルを提案し TLM 抽象度を形式化する。TLM により得られる最大の利益は、共通リファレンスを用いて、市場要求に応えられる、広範囲に渡るクロスチーム設計の手法を行える真の SW/HW 協調設計である。

■　内容

TLM が求められる性能として以下があげられる。

- スピード　……　モデルの速度として Mcycle/s 程度
- 精度　……　組み込みソフトを開発するのに必要となる精度
- モデルのし易さ　……　RTL 開発を実施することを考えると素早く開発

様々なモデルが提案されているが、一長一短がありこれら全てを満足することは難しい。そこで SoC TLM プラットフォームと SoC RTL プラットフォームの両方をうまく使い分けた分割統治法を実施し要求に応える。

TLM には、Untimed と Timed がある。以下、それぞれについて詳しく解説する。

Untimed TLM とは一般的に以下のような特徴を持つ。

- 早期 SW/HW 機能検証に活用
- マイクロアーキテクチャの情報は含まない
- 内部構造やアービトレーションを含まない
- API を提供し内部情報へのアクセスを許す
- データ転送粒度はターゲットアプリに従う(cf. フレーム単位)

この Untimed モデルでは、時間情報はないが複数のプロセスが同時実行するシステムをサポートする必要があり、システム同期を使って常に正しい順序を保証しなければならない。図 3 の例のように 3 つのプロセスが条件の下、同時実行している。その順番は順序 A と順序 B などがある。

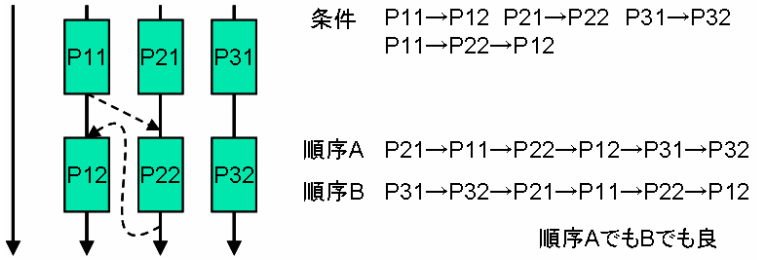


図 3　Untimed モデルの実行順序

注意点としては、デバッグ容易化の観点で再現性は必要であり、また違う順序でバグとならないよう開発となるようにしなければならない。

このような Untimed モデルの推奨モデリングをあげる。

- 最終的にどのような環境で使われるのか考慮する
- 簡単にアルゴリズムを更新でき、また単独で実行可能にする
- 取扱うデータ粒度は正確にアルゴリズムを定義し期待される精度を満たすレベル
- I/F、動作ともビット精度
- 機能だけに限定しマイクロアーキテクチャまで踏込まない
- IP の振る舞いに影響する同期は明確に定義する
- 必要な時のみモデル内部のプロセス間同期にイベントを使う
- 同期プロトコルのような特定の同期手段を利用すべき
- プロセス-アクティベーションの実装は避けるべき
- グローバル変数は禁止
- よいソフトウェアのコーディングスタイルに従う

Timed モデルの目的は以下のとおりである。

- 与えられたマイクロアーキテクチャのパフォーマンスに関するベンチマーク
- マイクロアーキテクチャの明快なチューニング
- 与えられたマイクロアーキテクチャで時間制約に関するソフト最適化

Timed モデルのアプローチとして以下にあげる 2 つが考えられる。

- 遅延付加モデル
  - ・タイミング遅延を挿入する形で指定する
  - ・マイクロアーキテクチャを考慮する点で遅延付き Untimed モデルと異なる
  - ・Untimed モデルがマイクロアーキテクチャ構造に近い場合に有用
  - ・(しかし実際は Untimed モデルに手を入れなければならないケースが多い)
- 単独遅延モデル
  - ・タイミング遅延を独立して計算
  - ・Untimed モデルが実物の構造とかけ離れていても適用が可能
  - ・機能毎にタイミングが違う場合、その情報を Untimed モデルから受取り計算

■ まとめ

TLM は SoC 設計サイクルを通して異なるチームの唯一のリファレンスと手の役割を果たす。TLM は、SoC 設計のボトルネックに立ち向かう信頼の置けるメソドロジであり、また、異なるチーム間で以下にあげる 3 つの目標を達成するための唯一のリファレンスとなる。

- 様々なチーム間を通して一貫した働き
- モデリング効果の合理化
- クロスチームのコミュニケーションとインタラクション

Chapter 3　TLM Modeling Techniques

■　著者

Laurent Maillet-Contoz and Jean-Philippe Strassen

■　概要

TLM の概念や手法は適切なシステム・レベル・モデリング言語による実装を行うことによって実現される。そのために SystemC を選択した。本章は、TLM モデリング技術を完結に示すための開発と特定の SoC 通信とを紹介する。

■　内容

□　モデリング環境

システムレベル設計のためのモデリング言語には、大きく分けてハードウェア設計言語・一般的なプログラム言語・独自開発された C ベース言語がある。これらの言語を使用するにはいくつかの問題がある。それは、ツール群を覚えるために長い学習期間が必要となること、ライセンスやインストールの問題によりモデルなどの共有が難しくなること、ライセンス代が高価なこと、研究プロジェクトから発生した言語のサポートが不確かなことなどである。

このため ST では、一つの言語で HW/SW 設計ができること、並列動作がモデリングできること、学習曲線が短いことなど、業界認定されるために必要となると思われる 9 つの項目を定義した。

そして、これまで ST では C/C++、Esterel、SystemVerilog など様々な言語を研究してきたが、システムレベルのモデリング言語には以下の 3 つの重要な条件があり、SystemC がこれらすべてを満たしているものと考えた。

1. ハードウェア・ソフトウェアのモデリング
2. シミュレーション・カーネルの存在
3. 標準 C++コンパイラの使用

□　開発環境

TLM 開発環境は、TLM SoC モデルの開発・デバッグ・統合の過程でモデリング・エンジニアを効果的にサポートできるべきで、どんな独自言語やツールにも依存しない環境を推奨する。この点では、最小のツール群は単にテキストエディタ、C++コンパイラ、デバッガとなる。

また、TLM モデルはツールから独立していて、異なる環境でリコンパイルできる。これにより、異なる環境における異なる機能を使って検証できることからテストされない機能や見落とされた機能が少なくなる。これは SOC の検証過程を向上する実質的な利点である。

より高い効果を得るため、TLM モデルは、SystemC をサポートするどんな EDA ツールでも開発できるよう設計される。EDA ツール機能の中で、プラットフォームの構築やデバッグを向上する要素として以下のものがある。

- システム・ネットリストの自動生成
- システムレベル・デバッガ
- トランザクション表示

□ モデリング・インフラストラクチャ

TLM 抽象度で新しいモデルを開発することは、システムレベル設計と検証フローの最初のステップである。自然と、そのようなモデルの再利用性が TLM 手法を認知するための重要な要素となる。

それゆえ、TLM モデリング・インフラストラクチャはモデルの再利用性を増加するように開発されてきた。このインフラストラクチャはソースコードやモデル・ライブラリの管理の役目を持つ。ユーザはインフラストラクチャが指示する同じ構成でモデルを作ることを促され、コードの重複を避け、モデルの再利用を促進する。

□ レイヤード・アプローチ

TLM レイヤード・アプローチは、モデリング・エンジニアが必要とする高いレベルの特殊なモデル管理手法を提供する。多くのシステム・モデリングにおいては、複数の異なるセマンティックや要素を持つ通信プロトコルが必要とされる。ある場合には、データをイニシエータからターゲットにポイント・ツー・ポイントで渡す単純な通信プロトコルでもよいが、場合によっては以下のような機能をサポートする複雑なプロトコルが必要となる。

1. アドレス、データ、バイト・イネーブル情報を含む複雑なデータ構造
2. 通信媒体を解したトランザクション・ルーティング
3. 検証目的のスコアボード能力

このようないろいろなモデリング要求に対応するため、TLM レイヤード・アプローチでは以下の 3 つの要素レイヤを定義した。

1. コア TLM インタフェース・レイヤ

   トランザクションレベルで SoC をモデリングするために必要な最低限のインタフェースを定義。トランザクションをイニシエータからターゲットに転送する能力を持つ通信 API が準備される。
2. プロトコル・レイヤー

   コア TLM インタフェース・レイヤ上にプロトコルを定義。
3. IP レイヤ

   プロトコル・レイヤー上に機能モジュールとして IP をモデル化。TLM IP 間の通信は、プロトコル・インタフェースに定義される通信 API を介して確立される。

□ モデリング

ここではインターコネクトと MPEG4 Codec をターゲットとした、イニシエータ・ターゲット・ルータの実装の構造と手順について紹介した。

■ まとめ

SystemC に基づく TLM アプローチのための基盤的モデリング技術について紹介した。

2 章で説明されたシステムレベル言語からモデリング環境やインフラストラクチャまでの TLM モデリング環境の簡単な概要から始め、TLM モデリング API についての拡張的論議を行った。TLM API の基盤について、エンドユーザからモデリングの複雑さを見えなくするレイヤード構造として紹介した。レイヤード構造では、トランザクションをイニシエータからターゲットへ転送するための通信 API を用いたトランザクション・レベル・モデリングのための最低限のインタフェースを定義するコア TLM インタフェースを実装し、その上のレイヤでは、トランザクション・ペイロードとブロッキング・ノンブロッキング転送を用いたトランザクション転送の TLM プロトコルを実装する。

Chapter 4　Embedded Software Development

■ 著者

Eric Paire

■ 概要

早めの組み込みソフト開発はTLMプラットフォーム方法論の最も重要な目標の1つである。本章は TLM プラットフォームとそれで動くソフトウェア(デバイスドライバ、OS/FW、アプリケーション SW)との密接な関係について説明する。

■ 内容

□ TLM プラットフォームを想定したソフトウェア

TLM によるソフトウェア開発は、対応する TLM の精度を反映する。このため、ソフトウェアは、TLM に対応するように並行して開発が行われるように管理する必要がある。その一方で、アンタイムドそしてタイムドの両 TLM プラットフォームにおいて、コードの変更なしで、正しく動作するソフトウェアを書くのは、よい習慣であり、TLM によって強化される。

TLM プラットフォーム上のソフトウェアの実行は、ハードウェア開発者にとって、ソフトウェアがどのようにハードウェア I/F を活用しているかにより、IP の機能仕様を改良する助けになる。ソフトウェア開発者にとっては、ソフトウェア実行中にハードウェア IP がどのように反応するかにより、実装を改良する手助けになる。TLM の使用により、リソースオーバーフローやハードウェアの活用度やパフォーマンスプロファイリングを早期に抽出することが可能になる。

TLM プラットフォームでは、CPU をモデル化する手法として、ネイティブコンパイル方式と、ISS などをベースとしたクロスコンパイル方式がある。

□ TLM レベルのデバイスドライバ

デバイスドライバとは、低レベルペリフェラルへのアクセスを関数群からなる一般的なプログラムインタフェースを提供するものであり、低レベルの IP マネジメントをモジュール化することを目的としている。

デバイスドライバレベルのテストの基本的な目的の 1 つは、IP が適切に実装されていることを確認することである。それは、後で HW の機能追加を行っても、本来の機能が正しく実行されることを確認する(リグレッションテスト)にも有用である。また、同じテストは、ベースとなる I/O アドレスと割り込みマッピングを変更すれば、再利用できる利点がある。

デバイスドライバレベルの実行の、その他の目的として、プログラミングの容易性やパフォーマンスの改善の開発、プログラム例の開発があるが、TLM で実装するメリットは、HW を変更する前、簡単にセットアップしてこれらの確認が行えることである。また、実チップでは(高速性ゆえに)検出が難しい問題を、TLM での検証では、その異常な応答時間で見つけることもできるメリットもある。

デバイスドライバのような低レベルのコードの品質は、通常の動作にあるのではなく、エラーマネジメントとリカバリーにある。ハードウェア関連するエラーを扱うのは、ソフトウェア開発者にとって再現が難しい問題であるが、TLM は、容易に誤り挿入し、デバイスドライバの振舞いをテストすることが可能である。

□ TLM レベルの OS/FW

デバイスドライバの上位に位置付けられるのが、OS とファームウェアである。OS/ファームウェアの開発では、低レベルデバイスドライバを並行して実行し、データフローの相互作用の問題(潜在的にハードウェア)を検出することが可能になる。

OS/ファームウェアの開発では、低級ソフトウェア層で定義されたメカニズムをどう効率的に動かせるか(性能)にフォーカスする。アクティブ Waiting ループの回避や、入出力を処理中の他のタスクの実行や割り込み管理である。アンタイムド TLM プラットフォームでは、割り込みは瞬時に起こるため、ポーリングでのモデリングがよい方法である。

OS/ファームウェアレベルでは、実行環境におけるネイティブ方式とクロスのコンパイル方式の違いを慎重に考えなければならない。パフォーマンスは、ネイティブの方がかなり早いが、OS/ファームウェア層では、管理すべき仮想記憶管理、キャッシュ管理、実行と割り込み経路のためにその高速性の実現はそれほど容易ではない。ソフトウェアデバッガの戦略も重要となる。クロスコンパイル環境では、組み込みデバッガがサポート可能であるが、ネイティブプラットフォームでは、唯一のデバッグ対策は、ホストデバッガの使用である。

□ TLM レベルのアプリケーション SW

ソフトウェア開発者にとって、最終的なゴールは、アプリケーション SW を TLM プラットフォーム上実行することである。

OS/ファームウェアが機能的に正しいことを示すだけ十分である一方で、アプリケーション・ソフトでは追加の情報を期待される。それは、タイムドプロファイリング(遅延、速度、スループット、デッドライン)、アンタイムドプロファイリング（カウンタ、競合、ボトルネック）や電力見積もりなどのパフォーマンス結果である。TLM プラットフォームがなければ、全体のアプリケーションを、これらの性能要素で正確に評価するのは、ほとんど不可能である。トランザクションの分析は SoC 開発における重要な意志決定のための情報となる。

また、ハイレベルのソフトウェアを動かすことはハードウェアとソフトウェア開発者の役に立つだけではなく、マーケティング担当にとっても価値があるものである。印象的なデモは、コミュニケーションの基礎となるものであり、要件へのコンプライアンスを立証するだけでなく、要求をすべていれた場合のインパクトも例証することが可能である。

組み込み型システムでは、コンピューティング資源、境界がある記憶容量、及びそれほど強力でない CPU など多くの規制がある。TLM 環境を使うことで、開発から実行とシミュレーションまで及ぶアプリケーション・ソフトデザインの早めのフェーズでそのような問題を検出できる。実際のプラットフォームから得るのがほとんど不可能な、内部の可観測性が得られるというメリットもある。

アプリケーションレベルで実行されるべきソフトウェアの量は巨大であり、パフォーマン

スを考えた場合、TLM の精度レベルを慎重に選ぶことが、非常に重要になる。

■ まとめ

TLM プラットフォームに基づくシステムを開発すると、ハードウェアとソフトウェアデザインの方法論とスケジュールをかなり改良できる。本当のハードウェアプロトタイプが準備できるかなり前に、ソフトウェア開発者はシミュレーションが可能になる。また、実際のハードウェアができたあとも、ソフトウェア開発者にとって重要な情報を提供可能である。そして、SoC プロジェクトの総合的な品質を上げ、性能問題に取り組む時間をソフトウェア開発者に与え、土壇場でのパッチを避けることが可能になる。

Chapter 5　Functional Verification

■　著者

T. Bultiaux, S. Guenot, S. Hustin, A. Blampey, J. Bulone, M. Moy
STMicroelectronics Belgium, STMicroelectronics France

■　概要

本章では、TLM を使った検証方法、検証環境について解説し、あわせて TLM を使用するメリットを説明する。

■　内容

TLM での機能検証では、TLM と RTL の動作の等価性の保障は重要である。検証は、(1) TLM DUT とテストベンチを作成、(2) テストシナリオで期待値を作成、(3) TLM テストベンチを使って RTL を検証、という順で行われる。

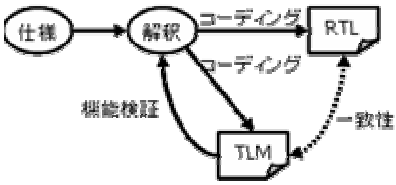


図 4　同一設計者が設計するケース

□　検証フロー

典型的な検証フローを右図に 2 つ示す。図 4 は同一設計者が TLM と RTL を設計するケースで、図 5 は TLM と RTL は別設計者が設計するケースである。後者では、TLM と RTL の並行開発が可能である。TLM 作成というオーバーヘッドはあるが、SW 開発、アーキテクチャ解析にも使用できるという、再利用性のメリットが大きい。また、TLM と RTL の動作の一致性もこのフローで保障している。

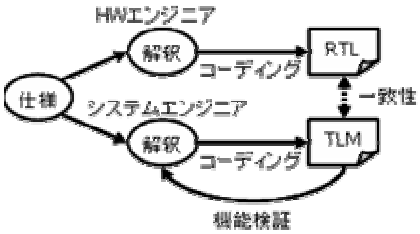


図 5　別設計者が設計するケース

□　検証環境

図 6、図 7 に示すように単体テストとシステムテストの環境を用意している。TLM は Untimed レベルのモデルを使用している。

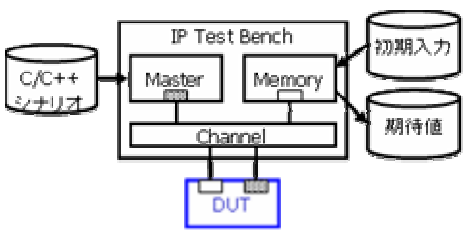


図 6　単体テスト用

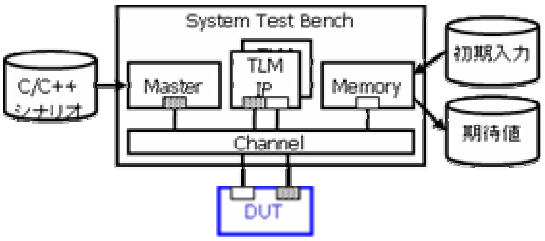


図 7　システムテスト用

DUT を RTL に変えるときは図 8 に示す「トランザクション ⇔ シグナル変換のアダプタ」を使用する。

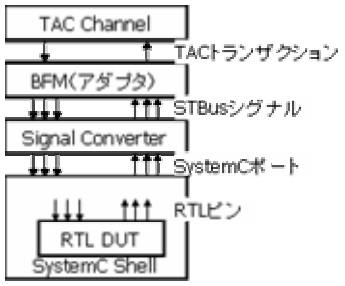


図 8　変換アダプタ

□　**検証ストラテジとカテゴリ**

この検証環境を使用して、(1) テストシナリオ(ディレクテブとランダム)と入力ベクタ(入力データと FW)の作成、(2) TLM テストベンチと SystemC DUT でテストシナリオ実行、期待値取得、(3) TLM テストベンチと RTL DUT でテストシナリオ実行、(4) 出力と期待値比較、テスト結果解析、を実施する。また、検証を「アーキテクチャテスト」と「マイクロアーキテクチャテスト」に分類し、前者ではタイミングに無関係な動作のテスト、後者ではタイミング動作(サイクルレベル)のテストを行う。

□　**エミュレーション**

TLM テストベンチを利用したエミュレーション環境を構築。ベストケースで full speed（10MHz）な検証が可能。再利用可能な高速 C/C++テストベンチの開発が容易。検証をシステムレベルで可能といったメリットがある。構成を図 9 に示す。

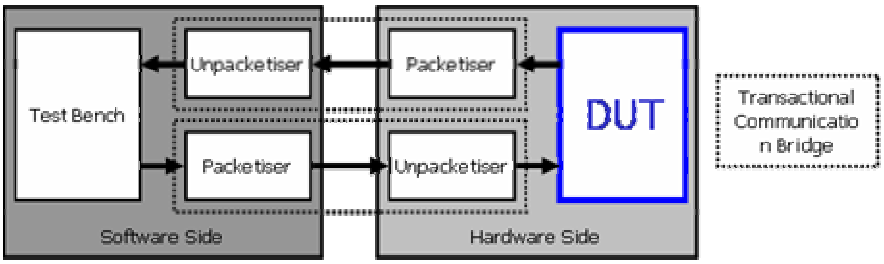


図 9　TLM テストベンチを利用したエミュレーション環境

□　**形式検証**

ある特定のプロパティをチェックする内製ツールを作成している。ただし、適用可能な回路規模はまだ小さい。チェック可能なプロパティは、「デッドロック」、「プロセスは決して終了しないこと」、「同期シグナルは同一デルタサイクルで 2 回書かれない」、「TLM ポートへの同時アクセスがないこと」である。

■　まとめ

TLM を使った機能検証では、「入力ベクタを与えて期待値と一致することを検証」、「ゴールデンモデルと RTL の動作が等価なことを検証」、「プロパティが“真”に保たれることを検証」を実施し、実装と機能の整合性を保証する。TLM を使うことのメリットは、「生産性の向上」、「モデルと実装の一貫性」、「検証と実装の並列化」である。

Chapter 6　Architecture Analysis and System Debugging

■　著者

Antoine Perrin, Gregory Poivre

■　概要

IP を多数含んだ大規模 SOC の検証に、TLM デバッグが有効であることを示す。

■　内容

□　SOC アーキテクチャ特有の検証

- 目的
  - ・IP を多数含んだ大規模 SOC の検証
- 各 IP が実時間制約を持っているとき、アーキテクチャの決定と検証は以下の 2 点に焦点をあてる必要がある
  - ・通信構造
  - ・シェアドメモリ・コントローラ
- SOC のアーキテクトが受け入れ易い単純な環境
  - ・SPIRIT を利用する

□　設計フェーズ

- 初期のマイクロアーキテクチャの決定
  - ・IPTG、COM、MEM によって行われる。
  - ・トポロジー、FIFO サイズ、アービトレーション、IP クラスタリング等の決定
- RTL 性能の検証
  - ・IPTG と、検証対象の RTL によって行われる
  - ・前のフェーズの IPTG 構成ファイルは再利用
- 実チップでの性能検証
  - ・AT に入力し、IPTG 構成ファイルの正確さを評価

□　TLM デバッグ

- 既存環境での問題点
  - ・SoC には多数のマスタと多数のスレーブが存在
    → バスや信号を追うのは困難もしくは不可能
  - ・複数の周波数
  - ・多くの IP のインスタンス
  - ・複数のプロトコル
- これらの問題に対する答が TLM デバッグ
  - ・デバッグ機能
  - ・解析機能

□ Traffic Generator

- 本システムの基幹
- SystemC(+SCV)で書かれている
- IP トラフィックの 2 種類の見方
  ・Traffic Mode：外部から見えるバスの動作で規定
  ・IP Modeling ：内部のトラフィックで規定
- IPTG 構成ファイル・テキストファイル
  ・パラメータ
  ・ヘッダと動作記述部よりなる

□ IPTG 構成ファイルによる入力

- 詳しい統計的トラフィック情報
- リファイメント情報(optional)
- オペコードのシーケンス・リスト
- IP の特徴を示すパラメータ・周波数
  ・データのサイズ
  ・その他
- 構成ファイルは解析ツール(SysProbe)が生成することもある

□ 設計例

- 通信モデルをスプレッドシートによる解析からスタート
- いろいろなシナリオを解析し、メモリバッファの場所や、メモリインタフェースのサイズを選ぶ
- 各 IP 間の接続はすべて SystemC の TLM でモデリング
- CPU 群は第 2 段階で ISS にリプレース
- CPU は traffic modeling mode、他は IP modeling mode
- 性能評価は、各 IPTG の FIFO をモニタすることによって容易
- バンド幅がワーストケースの IPTG 構成ファイルを作り、スプレッドシート解析に再利用
- 性能がミートしないときは、SysProbe で通信モデルの内部ノードまで観察する

■ まとめ

- 新しい言語を修得する必要もなく、プラットフォーム上の組み立てによるプラグ&プレイの環境を作ることができた
- 実システムを正確にはシミュレートできない、紙上の解析を補完するソリューション
- IP デザイナが直接 IP 構成ファイルを書き、様々なプロジェクト、様々な製品に再利用できる
- 強力で保守しやすい解析環境

Chapter 7　Design Automation

■　著者

Christophe Amerijckx, Stephane Guenot, Amine Kerkeni, Serge Hustin
STMicroelectronics Belgium, STMicroelectronics France, STMicroelectronics Tunisia

■　概要

本章では、SoC デザインサイクルの中で最良の TLM 使用法の確立に必要な実装の詳細について述べる。

■　内容

SoC デザインフローで TLM メソドロジを統合するために重要なのは、一般的に実現可能な『自動化』手法を確立して実施することにある。そのために標準フォーマットの SPIRIT を使用した TLM デザインデータから、それに準拠したエディタ、コンフィグレータ、チェッカ、ネットリスタに連なる自動化アプローチを取り入れる手法を紹介する。

□　SPIRIT 概要

SPIRIT コンソーシアムは、デザインのコンフィグレーションとインテグレーションの自動化による大規模な SoC 設計を加速させることを目的とした IP の記述とハンドリングの標準的なメカニズムを開発しており、各コンポーネントや IP 向けに定義された XML ファイルを SPIRIT 準拠のデザインツールが使用する。そのためのルールと制限も自動化ツールのユーザ･インタフェース実装のために SPIRIT によって定められている。

- SPIRIT XML のスキーム概要
- SPLIT 準拠ツールの動作概要

□　プラットフォーム構築自動化要求

ミリオンゲート規模の SoC 設計において、できる限りプラットフォームレベルで SoC フローのステップを実行すべきであるが、TLM でシミュレーション後は必ず RTL シミュレーションに進むことからエラーが発生しやすい傾向にある。そのため、プラットフォーム組み立ての自動化が要求されている。

- SPIRIT 準拠のジェネレータを使用することによる一般的な作業の自動化
- SPIRIT 準拠のユーザ入力が必要とされる作業向けツールの提供

□　自動化ツールの開発構想

SPIRIT スタンダード上での自動化ツール開発で採用される構想について。

1. SPIRIT メタレベル記述

あらゆるメソドロジやデータモデルの記述言語と構造は形式的なルールに則って書かれなければならない(メタレベル・ルール)。SPIRIT 標準において XML ドキュメントのメタレベル記述は XSD（XML Schema Definition）で与えられている。

2. SPIRIT コンテントレベル記述

メタレベル記述元に必要な情報を SPIRIT XML ドキュメントとして保持し、ジェネレータのようなツールセット向けの入力を供給する。W3C コンソーシアムでは SPIRIT オブジェクトの作成、変更、削除するために固有の API、Xerces を供給する。

3. API 生成

SPIRIT スペシフィックな API 構造を生成するために、SPIRIT スキーマをベースとして作る方法によって、異なるアプリケーションであっても SPIRIT XML フォーマットをベースに変更ができる。例え SPIRIT バージョンがアップデートや変更が行われても API ソースコードの再生成を可能にする。

4. 開発環境と相互運用性

SPIRIT 開発環境はユーザ固有の設計環境を作成するために必要なツールとファシリティを SoC 開発者に供給する。また、SPIRIT は異なる EDA ベンダや IP ベンダによって提供される異なるツール間の相互運用性についてもソリューションを供給する。フローのあらゆるステップでエンドユーザが SPIRIT 準拠のツールから他のツールに交換することが可能になる。

□ **SPIRIT 準拠の自動化ツール**

SPIRIT 準拠自動化ツールは次の 5 つに分類される。

1. パッケージャ

SPIRIT 準拠のパッケージャは全ての入力データを電子コンポーネントの XML 記述にパッケージするための自動化ツールである。

2. エディタ

ASCI フォーマットである XML ドキュメントはあらゆるテキストエディタで編集可能であるが、エンドユーザはより XML フォーマットに即した GUI などのエディタを期待する。

2 種類の GUI エディタが利用できる。

1. XML ジェネリックエディタ
2. IRIT スペシフィックエディタ(SPIRIT の文法やルールに即したエディタ)

3. チェッカ

SPIRIT 準拠のチェッカは SPIRIT ユーザガイドの中に書かれる文法のルールを検証するために特化したツールである。

4. コンフィグレータ

SPIRIT 準拠のコンフィグレータは、デザインの内容に基づいて SPIRIT データをコンフィグレーションするためのツールである。このコンフィグレーションは次のいずれかがベースとなる。

1. テンプレート(インタフェースを複製するコンフィグレーション)
2. デザインにユーザが入力

SPIRIT コンフィグレータは XML を入力し、コンフィグレーションされた XML を出力するツールである。

5. ジェネレータ

SPIRIT 準拠のジェネレータは重要なツールファミリである。デザインフローにおいて、各々ジェネレータは特定の作業を目的とし、ジェネレータが連続するチェインを構成することができる。SoC にフローにおいて重要なジェネレータはネットリスタである。

・ネットリスタ

- コ・シミュレーション・ネットリスト
- コ・エミュレーション・ネットリスト

・その他ジェネレータ例

- リグレッション・ジェネレータ
- レジスタ・アクセス・テスト・ジェネレータ

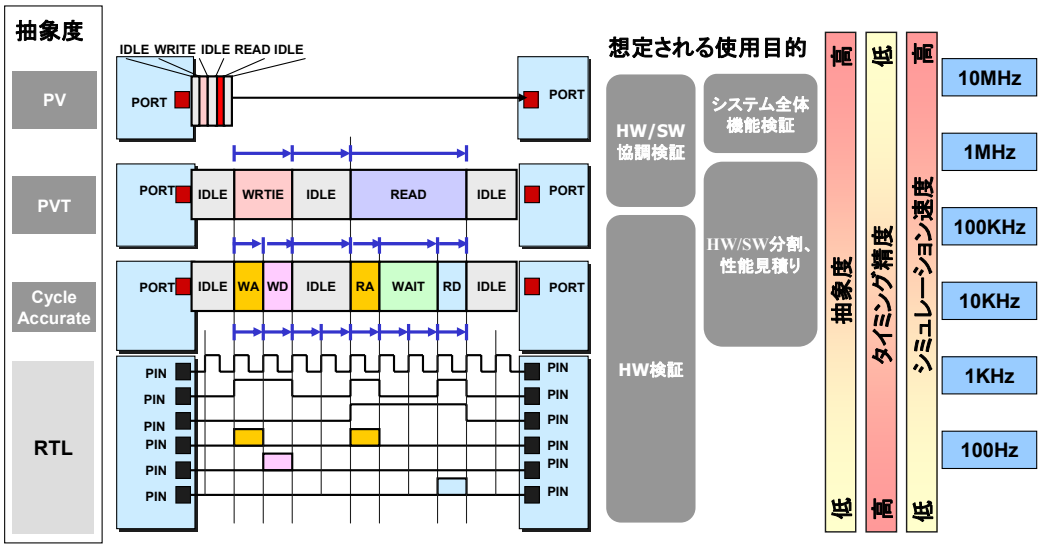
■ まとめ

TLM を用いた設計フローにおける自動化構想を標準的な SPIRIT 開発環境を用いた例を紹介している。この章におけるポイントは標準フォーマットによる相互運用性も考慮したデータベースと API 生成。そしてそのフォーマットに準拠した必要なツール群とその概念である。

4.2.3.3 TLM動向調査

JEITA

TLMの抽象度定義と想定される使用目的



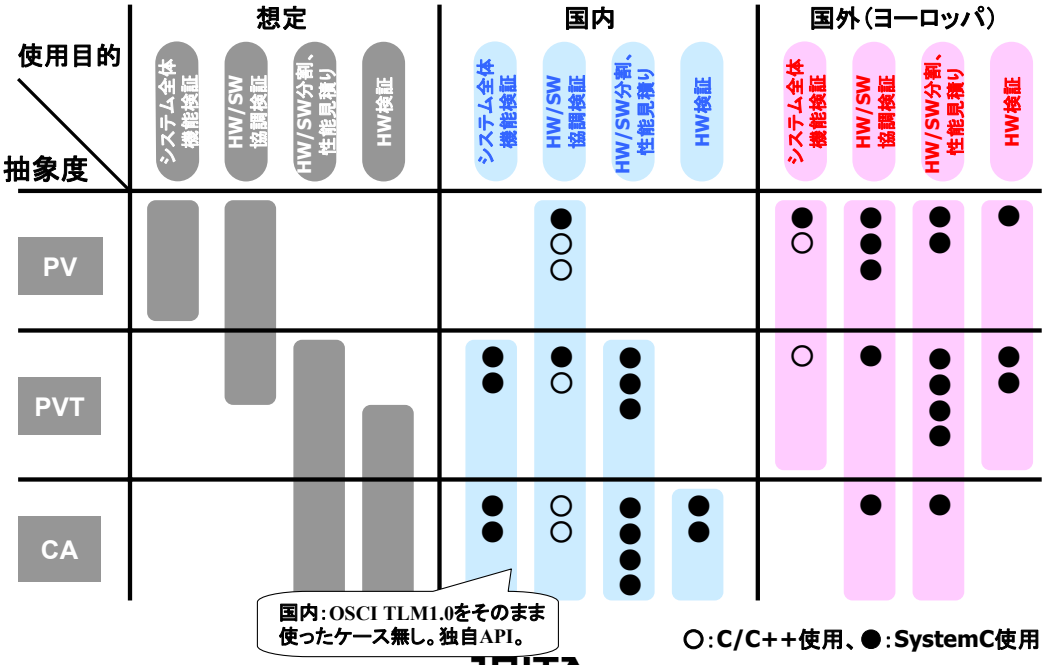
JEITA

国内外各社TLM利用状況調査

- 国内・国外のSystemCユーザーがTLMをどの抽象度（PV、PVT、CA）で何を目的に使用しているかを調査。
- 調査対象
  - 国内：JEITA SystemC TG参加企業
  - 国外：ECSI Institute Workshop on Users Experience with TLM, June 8, 2006 - Paris, France
- TLM使用目的の分類
  - システム全体機能検証
  - HW/SW分割、性能見積り
    - HW/SW分割、アーキテクチャ検討・生成、性能見積り
  - HW検証
    - HWモジュールの検証、例えば、RTL、高位合成用記述をTLMテストベンチで検証
  - HW/SW協調検証
    - HWモデル上でのデバイスドライバ、アプリケーションSWの設計・検証

JEITA

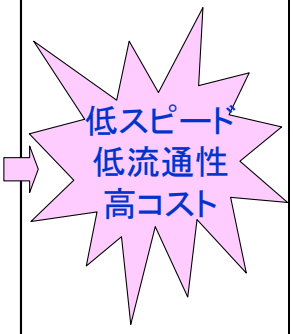
国内外各社TLM利用状況調査（結果）



JEITA

国内でのTLM使用に関する現状の課題

- CAレベル主体の低抽象度のTLMの使用
  - CAモデルはシミュレーション速度が不十分。生産性低い。
  - 生産性向上のために、より高抽象度のTLM使用が必要。
  - しかし、現状PV、PVTモデル利用のメソドロジの確立が不十分で高抽象度への移行ができていない。
- 独自APIの多用
  - OSCI TLM1.0をそのまま使った例は無し。独自API多数。
  - モデルの再利用性への障害。



解決策はPV、PVTのメソドロジの確立、TLM APIの標準化！

JEITA

TLMメソドロジ成功の鍵

- PV、PVTのメソドロジ確立・普及のために
  - 成功（失敗）事例、適用事例などの積極的な情報発信
  - ユーザフォーラムの開催
- TLM APIの標準化への寄与
  - OSCI TLM2.0調査とフィードバック
  - OSCI TLM2.0でPV/PVTのレベルは共有可能になることを期待。（OSCI TLM1.0では低抽象度トランザクションレベルの基本I/Fのみ標準化。もっと高抽象度の標準が欲しい。）

JEITA

4.2.3.4 TLM2.0へのフィードバック

JEITA

OSCI TLM2.0へのフィードバック

- ドキュメント（LRM）への追加項目
  - 抽象度
    - 抽象度の定義
    - TLM2標準でカバーされる抽象度
    - 各抽象度の利用目的
  - クラス
    - クラスの構造図
    - すべてのクラスのサンプルコード
- クラスの名前付け
  - すべてのクラス名を「tlm_」で開始
- 質問事項
  - 「timed unidirectional blocking IF」が無い理由

JEITA

4.2.4 合成サブワーキンググループ活動報告

4.2.4.1 合成サブセットレビュー

「SystemC Synthesizable Subset Draft 1.1.18」(以下「Synthesizable Subset」)は OSCI(Open SystemC Initiative)の Synthesis Working Group が作成したドキュメントであり、SystemC の言語仕様の中から動作合成／論理合成ツールがサポートすべきサブセットを定義したものである。今回取り上げたバージョンは 2004 年 12 月 23 日に作成され、2006 年 7 月 13 日に Public Preview として公開された Draft 1.1.18 である。

「Synthesizable Subset」に記載されている内容は「合成ツールが扱うべき構文の定義」「モジュールの記載方法(フォーマット)」「合成ツールが扱うデータタイプと演算の定義」等であり、章立ては ISO C++の LRM に準拠している。表記方法としては BNF を基本しており、それを文章、表等で解説している。「Synthesizable Subset」では、各構文がどのような回路にインプリメントされるべきかについては定義していない。また、スタイルガイド的な記述ではなく、あくまでツール実装者向けのドキュメントとなっている。

SystemC タスクグループでは「Synthesizable Subset」の内容を確認し、レビューを行った。そしてレビューの結果、合計 57 件の不具合・改善要望を OSCI Synthesis-WG へ Issue List として提出し、フィードバックを行った。カテゴリ別に見た Issue の件数は、誤記の指摘 8 件、IEEE 1666-2005 準拠対応の修正要求 11 件、ドキュメントの記載内容についての改善要望 24 件、技術的な改善要望 14 件となっている。不具合・改善要望の内訳を見ると、記載内容に関する指摘が約半数あることからも、LRM としては説明不足な部分がまだ多く残っており、ブラッシュアップが必要である。また、技術的な改善点や IEEE 1666-2005 対応の指摘もかなり存在し、定義内容としても見直すべき所がある。特に IEEE 1666-2005 対応の指摘については、「Synthesizable Subset」が作成された時期が IEEE 標準化前ということもあり、ベースになっている SystemC LRM が OSCI v2.0.1 であるため、現在の LRM に準拠していない箇所が存在する。また、一部には IEEE 1666-2005 自体に不明確な箇所も存在しており、IEEE 1666-2005 自体の改定も必要だと考える。今後の標準化を考えた場合は、このあたりの見直しがキーになると考えられる。

今回のレビュー作業に伴い、「Synthesizable Subset」の抄訳として「OSCI-SystemC 合成サブセット Draft 1.1.18 の概要」を作成した。今回「OSCI-SystemC 合成サブセット Draft 1.1.18 の概要」も一緒に添付するので、「Synthesizable Subset」と一緒に参照していただきたい。

また、今回 OSCI に提出した Issue List については、現在 OSCI 側で内容の検討中である。SystemC タスクグループでは引き続き OSCI への改善要望や「Synthesizable Subset」標準化へのフォローを行っていく予定である。

4.2.4.1.1 OSCI へ提出した SystemC 合成サブセット Draft に対する不具合・改善要望リスト

アニュアルレポート向けに追記
↓

JEITA ISSUE LIST for “SystemC Synthesizable Subset Draft 1.1.18” (Released on December 23, 2004)							
JEITA Issue #	Clause/ Subclaus	Title	Type of Comment (Technical/Editorial)	COMMENTS (Justification)	Proposed change	Resolution	不具合・改善要望の分類
1	1.3	Terminology	Technical	Missing description of 'no difference with C++'.	Add explanation of 'no difference with C++'.		記述の改善要望
2	1.3	Terminology	Technical	Missing description of 'supported but not recognize'.	Add explanation of 'supported but not recognize'. Or, map to the other category.		記述の改善要望
3	1.4	Conventions	Editorial	Improper use of conventions : boldface (reserved words), fixed-width font(SystemC examples and code fragments), struck-through (not supported), underscored (ignored), shadowed (extended)	Unnecessary boldface in 1.3 Synthesis		誤記訂正
4	–	–	Editorial	In some chapter/section, there is not clearly described supported or not.	Add status of supported or not, into such as Chapter 4, 8.		記述の改善要望
5	3.1.2.2	Module constructor	Technical	Better to use 'int' rather than 'unsigned char', in template part of example 2.	Change 'unsigned char' to 'int'.		記述の改善要望
6	3.1.2.2	Module constructor	Technical	Should use 'inc' rather than 'increment', in the code of example 3.	Change 'increment' to 'inc'.		記述の改善要望
7	3.1.2.2	Module constructor	Technical	In example 3 (constructor part), 'sc_module_name&' and 'sc_module_name const' is inconsistent with the syntax summary.	Change example to follow the syntax summary.		記述の改善要望
8	3.2	Deriving modules	Technical	In example (constructor part), 'sc_module_name&' and 'sc_module_name const' is inconsistent with the syntax summary.	Change example to follow the syntax summary.		記述の改善要望
9	4.4.1	Integer Types	Editorial	Inadequate note about 'compile flag _32BIT_'.	Remove the sentence of "The compile flag _32BIT_ is not supported as it is not even mentioned in the LRM.".		記述の改善要望
10	4.4.1	Integer Types	Editorial	Inadequate note about 'compile flag MAX_NBITS '.	Remove the sentence of "(note: LRM does not mention this limit).".		記述の改善要望
11	4.4.1.4	Shift Operators	Technical	Regarding the 'negative shift value', there is no consistency between integer type and fixed point data type. (even not described in LRM) integer : ?? fixed point : negative shift value is allowed (in reference simulator)	IEEE 1666 has no description about the behavior with negative shift value. Need to revise LRM first. Then synthesizable subset follows it.		IEEE 1666対応
12	4.4.1.4	Shift Operators	Technical	Also, the negative shift value in sc_bigint/sc_biguint is not defined in IEEE 1666. (Inconsistency to sc_int/sc_uint.)	IEEE 1666 has no description about the behavior with negative shift value for sc_bigint/sc_biguint.. Need to revise LRM first. Then synthesizable subset follows it.		IEEE 1666対応
13	4.4.1.6	Bit Select Operator	Technical	Inconsistent behaviors when out of range value is specified. ([0,W-1]) sc_bigint/sc_biguint : accept out of range value (in simulator) sc_int/sc_uint : not accept	IEEE 1666 defines out of range value is not allowed. 7.2.5 Bit-select It shall be an error if the specified bit position is outside the bounds of its numeric type or vector object. So that, synthesizable subset should follow it.		IEEE 1666対応
14	4.4.1.7	Part Select Operator	Technical	Inconsistent behaviors when reverse order value is specified. sc_bigint/sc_biguint : accept reverse order (in simulator) sc_int/sc_uint : doesn't accept	IEEE 1666 defines reverse order is not allowed. 7.2.6 Part Select It shall be an error if the left-hand index position or right-hand index position lies outside the bounds of the object. NOTE 1-A part-select cannot be used to reverse the bit-order of a limited-precision integer type. So that, synthesizable subset should follow it.		IEEE 1666対応
15	4.4.1.9	Unsupported Methods	Technical	Outdated description of ″Otherwise they should probably be supported (?). These are being added in 2.1. I think we should say they are supported.″, since reduce method is already supported in IEEE 1666.	Remove description for 'reduce methods'.		IEEE 1666対応
16	5.1.1	Specifiers	Technical	'friend' should not be ignored.	'friend' should be treated same as in C++.		技術的改善要望
17	5.1.1.9	SystemC type specifiers	Technical	'sc_bit' is deprecated in IEEE 1666.	Delete 'sc_bit'.		IEEE 1666対応
18	5.1.1.9	SystemC type specifiers	Technical	Fixed point data type should be supported.	TBD		技術的改善要望
19	5.1.1.9	SystemC type specifiers	Technical	Unnecessary description (...) in the bottom of Page 30.	Remove (…) in the bottom of Page 30.		記述の改善要望
20	5.2	Declarators	Technical	Pointer for static memory should be supported.	TBD		技術的改善要望
21	5.2.1	Type names	Technical	Pointer for static memory should be supported.	TBD		技術的改善要望
22	5.2.5	Initializers	Technical	It is unclear whether the other classes than module class are supported or not.	Clearly state whether the other classes than module class are supported or not.		記述の改善要望
23	5.2.5.3	5.2.5.3 References	Editorial	-->	Clearly state that this section is referencing the C++.		記述の改善要望
24	6	Expressions	Technical	Regarding i = x[i++]; only the ISO C++ document's note is described, but no description for the synthesis. (The result may be depend on the compiler system.)	Add description for synthesis.		技術的改善要望

JEITA Issue #	Clause/ Subclaus	Title	Type of Comment (Technical/Editorial)	COMMENTS (Justification)	Proposed change	Resolution	不具合・改善要望の分類
25	8	Statements	Editorial	-->	Syntax summary should be like this. statement ::= labeledstatement | expression_statement | compound-statement | wait-statement | signal-assignment-statement | selection-statement | iteration-statement | jump-statement | declaration-statement		記述の改善要望
26	8.3	Wait statement	Technical	Incompatible with BNF, as wait(constant-expression).	Change 'wait(constant-expression)' to 'wait(expression)'.		誤記訂正
27	8.3	Wait statement	Technical	Thread process SC_THREAD is not supported, as described in 9.3 SC_THREAD.	Remove description.		誤記訂正
28	8.3	Wait statement	Technical	wait_until' is deprecated in IEEE 1666.	Delete 'wait_until'.		IEEE 1666対応
29	8.5	Variable assignment	Technical	This one is included in C++ 'expression statement'. So that this clause is not necessary.	Remove this clause.		記述の改善要望
30	8.6	Selection statements	Technical	Inadequate structure of section 8.6, 8.6.1, 8.6.2.	In section 8.6, change 'if (condition) statement' to 'if statement', and detail should be described in 8.6.1.		記述の改善要望
31	8.6.1	If statement	Technical	Inadequate structure of section 8.6, 8.6.1, 8.6.2.	Describe in same style as '8.6.2 switch statement'.		記述の改善要望
32	8.6.2	Switch statement	Technical	Restrictions for 'switch' statement is not reasonable. Should be same as in C++.	Remove restriction.		技術的改善要望
33	8.7	Procedure call statement	Technical	This one is included in C++ 'expression statement'. So that this clause is not necessary.	Remove this clause.		技術的改善要望
34	8.9	Jump statement	Technical	Missing 'goto' clause.	Add 'goto' clause.		記述の改善要望
35	8.9.1	Break	Editorial	Title should be 'Break Statements'	Change 'Break' to 'Break Statements'.		記述の改善要望
36	8.9.2	Continue statement	Editorial	Incorrect descriptions for 'Break' and 'Continue'.	Exchange descriptions between the two.		記述の改善要望
37	8.10	Declaration statement	Technical	Missing description.	Add description as 'supported'.		記述の改善要望
38	9	Processes	Technical	Using 'sc_bit' in the example. But sc_bit is deprecated in IEEE 1666. Need to use 'bool'.	Change 'sc_bit' to 'bool'.		IEEE 1666対応
39	9	Processes	Editorial	Since 'process' is extended, needs to be 'Shadowed'.	Add 'Shadow' to 'process'.		誤記訂正
40	9	Processes	Technical	sensitive_pos/sensitive_neg' are deprecated in IEEE 1666.	Delete 'sensitive_pos/sensitive_neg'.		IEEE 1666対応
41	9.2	SC_CTHREAD	Technical	'watching' is deprecated in IEEE 1666. Need to use 'reset_signal_is'.	Change 'watching' to 'reset_signal_is'.		IEEE 1666対応
42	9.3	SC_THREAD	Technical	'SC_THREAD' should be supported.	TBD		技術的改善要望
43	10	Submodule instantiation	Editorial	Missing 'Shadow' in BNF.	Add 'Shadow'.		誤記訂正
44	12.1.2	Class members	Technical	Missing description.	Add description.		記述の改善要望
45	12.1.4.1	Static members	Editorial	Should refer chapter 7, static member function.	Add reference to chapter 7, static member function.		記述の改善要望
46	12.4.7	Construction and destruction	Technical	Unclear description for 'destruction'.	Improve description for 'destruction'.		記述の改善要望
47	Annex A	Syntax summary	Editorial	'space take the place of underlines' is incorrect. It should be 'space take the place of hyphens'.	Change 'space take the place of underlines' to 'space take the place of hyphens'.		誤記訂正
48	A	Syntax summary	Technical	Missing 'Ignore'(with underline) description.	Add 'Ignore'(with underline) description.		誤記訂正
49	A.2	Lexical conventions	Technical	All the sc_object should accept the 'string' as a name. In other way, 'string' can be converted to the constant.	Remove 'struck-through' on 'string'. (Make 'string' usable in literal.)		技術的改善要望
50		Lexical conventions	Technical	'string' should be treated as an array of 'char' type.	Remove 'struck-through' on 'string'. (Make 'string' usable in literal.)		技術的改善要望
51	A.3	Basic concepts	Editorial	Missing 'Shadow' for 'sc-main-definition'.	Add 'Shadow' for 'sc-main-definition'.		誤記訂正
52	A.4	Expressions	Technical	Destructor should be supported.	TBD		技術的改善要望
53	A.4	Expressions	Technical	Member of static memory should be specified by pointer.	Add : (Make these usable) postfix-expression -> [template] id-expression postfix-expression -> pseudo-destructor-name		技術的改善要望
54	A.5	Statements	Technical	'variable-assignment-statement' and 'procedure-call-statement' are not added in SystemC.	Remove 'Shadow' for 'variable-assignment-statement' and 'procedure-call-statement'.		記述の改善要望
55	A.5	Statements	Technical	Restrictions for 'switch' statement is not reasonable. Should be same as in C++.	Remove restriction.		技術的改善要望
56	A.6-1	SystemC Type Specifiers	Technical	'sc_bit' is deprecated in IEEE 1666.	Delete 'sc_bit'.		IEEE 1666対応
57	A.7	Declarators	Technical	Declaration of the pointer for static memory should be supported.	Add : (Make these usable) * [cv-qualifier-seq] [::] [nested-name-specifier] * [cv-qualifier-seq]		技術的改善要望

誤記訂正	8
記述の改善要望	24
技術的改善要望	14
IEEE 1666対応	11
計	57

4.2.4.1.2　OSCI-SystemC 合成サブセット Draft 1.1.18 の概要

本ドキュメントは OSCI の Synthesis WG が作成した「SystemC Synthesizable Subset Draft 1.1.18」について、概要を説明したものである。内容については特に重要な点のみ記載しているので、完全な翻訳書ではない。したがって、詳細については、原文を参照のこと。本文中の章番号は、原文と同じである。

また、文中の「下線＋太字」で書かれている箇所は、SystemC タスクグループ内でレビューした結果であり、OSCI Synthesis WG にフィードバックを行った。各レビューについている番号は、レビューの Issue List である「4.2.4.1.1 OSCI へ提出した SystemC 合成サブセット Draft に対する不具合・改善要望リスト」との対応を示す。

1. 概要

1.2 目的

- 合成可能な SystemC 構文(サブセット)を規定している。
- 動作合成&論理合成を対象としている。
- コーディングガイドラインではない。
- Basic Requirements of SystemC Subset for Synthesis(V2.0.1 準拠)と Advanced Requirements of SystemC Subset for Synthesis(V2.0.1 非準拠)を作成して、これらを元にして本ドキュメントを作成している。

1.3 用語

- 本ドキュメント内の SystemC 構文の合成ツールでの扱い方は以下のとおり。
  - ✓ Supported：コンストラクト(構成)をそのとおりに処理(解釈)する。
  - ✓ Extended：C++構文を拡張して処理(解釈)する。
  - ✓ Ignored：コンストラクト(構文)を無視する。処理(解釈)しない。
  - ✓ Not Supported：コンストラクト(構文)が入力されることを想定していない。エラーになる場合もあるし無視される場合もある。(実装依存)

- **(1) no difference with C++の説明が必要である。もしくは、最初の分類に合わせた表現にするべきである。**
- **(2) Supported but not recognize の表現があるので、最初に規定したカテゴリに分けるべきである。**
- **(3) ボールド(予約語)、下線(無視)、取り消し線(サポートしない)、影付き(拡張)、固定幅フォント：例などの表記は統一されていないので統一すべきである。**
- **(4) 章によって、Support が明示されているところとされていないところがあるので、明記すべきである。**

1.4 表記法

- g) 例題は、推奨するコーディングスタイルを示すものではない。

1.5 参照

- ISO/IEC 14882: Programming languages - C++, 1998
- ISO/IEC 9899: Programming languages - C, 1999
- SystemC 2.0.1 Language Reference Manual, Revision 1.0, Open SystemC Initiative, 2003

2. 合成単位とその解析

- 合成ツールにとって main 関数は合成対象外。
- 合成ツールは sc_main を無視してもよい。
- 合成ツールは sc_main を認識して、sc_main からテンプレート引数やコンストラクタ引数の情報を取得してもよい。

3. モジュール

3.1 モジュール定義

- モジュール定義は以下のいずれかである。
  - ✓ SC_MODULE マクロの使用
  - ✓ sc_module クラスからの直接派生
  - ✓ sc_module クラスからの間接派生
  - ✓ template 引数を持つモジュール定義もサポート

3.1.2 モジュール宣言

- 共有メンバ変数(信号やポートではない)は合成不可。
- モジュール内で宣言された変数は同一モジュール内のただ 1 つのプロセスからのみアクセス可能。
- コンストラクタは SC_CTOR と普通の関数形式のどちらもサポートする。
- SC_CTOR を使用する場合、SC_HAS_PROCESS を使用してはいけない。

3.1.2.1 ポートと信号

- サポートしているものは以下のとおり。
  - ✓ sc_signal, sc_in, sc_out, sc_inout, sc_in_clk, sc_out_clk, sc_inout_clk
  - ✓ sc_signal_resolved, sc_in_resolved, sc_out_resolved, sc_inout_resolved
  - ✓ sc_signal_rv, sc_in_rv, sc_out_rv, sc_inout_rv

3.1.2.2 モジュール・コンストラクタ

- 例にあげるような、以下のいずれかである。
  - ✓ SC_CTOR マクロを使って記述した場合
  - ✓ SC_CTOR マクロを使わずに記述した場合
    - ■ この場合、少なくともモジュール名は引数として渡す必要あり

例 1

```
SC_MODULE( FullAdder ) {
          sc_in< bool> X, Y, Cin;
          sc_out< bool > Cout, Sum;
          SC_CTOR( FullAdder ) {}
};
```

例 2

```
template< unsigned char N = 2 >
SC_MODULE( AndGate ) {
          sc_in< sc_uint< N > > Inputs;
          sc_out< bool> Result;
          SC_CTOR( AndGate ) {}
};
```

例 3

```
SC_MODULE(Increment) {
          sc_in<sc_int<16> > A;
          sc_out<sc_int<16> > X;
          const int m_delta; // initialized by the constructor
          SC_HAS_PROCESS(Increment);

Increment(sc_module_name& name_, int delta)
          : sc_module(name_), m_delta(delta)
          {
              SC_METHOD(proc);
              sensitive << A;
          }

          void proc() {
              X = A.read() + m_delta;
          }
};
```

- (5) 例 2 のテンプレート部分の unsigned char は int にすべきである。
- (6) 例 3 のコード中「Increment」は「inc」にすべきである。
- (7) 例 3 の記述例にあるコンストラクタの部分で sc_module_name 後の&や sc_module_name の const は、Syntax summary に従うと記述できないので修正が必要である。

3.2 派生モジュール

- 記述例は以下のとおり。

記述例

```
SC_MODULE( BaseModule ) {
          sc_in< bool > reset;
          sc_in_clk clock;
          BaseModule ( const sc_module_name& name_  )
              : sc_module( name_ )
              {}
};
class DerivedModule : public BaseModule {
          void newProcess();
          SC_HAS_PROCESS( DerivedModule );
          DerivedModule( sc_module_name name_ )
          : BaseModule( name_ ) {
              SC_CTHREAD( newProcess, clock.pos() );
              watching(reset.delayed() == true);
          }
};
```

- 派生モジュールを定義するキーワード class を使用する。
  - ✓ SC_MODULE を使用してはいけない。
- 派生モジュールがプロセスを持つ場合は、SC_HAS_PROCESS を記述する。
- SC_CTOR は使用不可。

➢ (8) 記述例にあるコンストラクタの部分で sc_module_name 後の&や sc_module_name の const は、Syntax summary に従うと記述できないので修正が必要である。

4. データタイプ

4.1.1 整数型

- サポートするのは以下のとおり。
  - ✓ unsigned int, signed int
  - ✓ unsigned short, signed short
  - ✓ unsinged char, signed char, char
  - ✓ unsigned long, signed long
  - ✓ unsigned long long, signed long long
  - ✓ Bool
- wchar_t は未サポート
- 合成ツールは char（signed でも unsigned でもない）は signed char として扱う。（ISO C++では実装依存）

4.1.1.1 表現

- 合成ツールは、整数型を 2 の補数として実現する。

4.1.1.2 ビット幅

- シミュレーションと合成とで一貫性を持たす。（同一プラットフォームにおいて）

4.1.2 型変換

- ISO C++に準拠。
- （注） ISO C++に含まれていない long long と unsigned long long を追加。

4.1.3.1 シフト演算（a<<b, a>>b）

- a が符号付きの場合、右シフトすると合成ツールでは符号ビットの値がシフトインする。
- ISO C++では a が負の場合の右シフト動作は未定義である。

4.1.3.2 単項+演算子（+a）

- ISO C++に準拠。

4.1.3.3　単項-演算子（-a）

- ISO C++に準拠。

4.1.3.4　+演算子（a+b）

- ISO C++に準拠。

4.1.3.5　*演算子（a*b）

- ISO C++に準拠。

4.1.3.6　/演算子（a/b）、%演算子（a%b）

- 0 での除算（合成では、ドントケアとし論理最適化に利用）
- 0 への丸め（合成では、ISO C ではなく、ISO C++に準拠）

4.1.3.7　~演算子（~a）

- ISO C++に準拠。

4.1.3.8　ビットワイズ演算子（a&b, a|b, a^b）

- ISO C++に準拠。

4.1.3.9　比較演算子（<, >, <=, >=, ==, !=）

- ISO C++に準拠。

4.1.3.10　条件演算子（a?b:c)

- ISO C++に準拠。

4.1.3.11　インクリメント、デクリメント演算子（a++, ++a, a--, --a）

- 合成ツールでは、bool 型変数のインクリメント、デクリメントはサポートしない。
  - ✓　ISO C++ではサポートされている。

4.1.4　浮動小数点型

- 合成ツールは浮動小数点型をサポートしない。

4.2.1　配列

- ポインタの配列以外はサポートする。
- 信号及びポートの配列もサポートする。
- 初期化子がない場合、配列宣言時には配列範囲の記述が必要である。
- 初期化子がある場合、配列の範囲は初期化子から導出される。

4.2.2 ポインタ

- コンパイル時にアドレスが決定できるポインタについてはサポートする。
  - ✓ ポインタが配列をさす場合、配列サイズが決定している必要がある。
  - ✓ ポインタの演算は許されていない。
  - ✓ 「ポインタが 0 であるか？」というテストは許されていない。
  - ✓ データとしてのポインタ値の使用は許されていない。

4.2.3 参照

- 参照はサポートする。

4.2.4 列挙型

- 列挙型はサポートする。

4.2.5 クラスメンバとしてのポインタ

- コンパイル時にポインタの先に実体がある場合はサポートする。

4.3 CV 修飾子（型修飾子）

- const 修飾子と volatile 修飾子は、合成結果には反映されない。
  - ✓ あってもなくても合成結果は同じになる。

4.4 SystemC データタイプ

- 以下のものをサポートする。
  - ✓ 整数型：sc_int, sc_uint, sc_bigint, sc_biguint
  - ✓ 固定小数点型：sc_fixed, sc_ufixed
  - ✓ ビットベクタ型：sc_bv
  - ✓ ロジック型（4 値）：sc_logic, sc_lv
- ベクタ長や精度がコンパイル時に決定できる場合にサポートする。

4.4.1 整数型

- 以下のものをサポートする。
  - ✓ sc_bigint<W>, sc_biguint<W>
  - ✓ sc_int<W>, sc_uint<W>（W≦64）
- 符号付き整数は 2 の補数で表現する。（演算結果も同様である）
- SystemC の整数型に対してサポートする演算は以下のとおり。

Op Category	Operators/Methods					
Arithmetic	+	-	*	/	%	
Assign	+=	-=	*=	/=	%=	
Unary	+	-				
Auto Incr/dec	++ prefix	++ postfix	-- prefix	-- postfix		
Bitwise	&	|	^			
Assign	&=	|=	^=			
Unary	~					
Relational	==	!=	<	<=	>	>=
Shift	>>	<<	>>=	<<=		
Bit Select	[x]					
Part Select	(i,j)					
Concatenation	(,)					
Conv to C integral	to_int	to_long	to_int64	to_uint	to_uint64	to_ulong
Assignment	=					

- 合成可能なメソッドは以下のとおり。

Methods	Alternatives
iszero	x == 0
sign	x < 0
bit	x[i]
range	x(i,j)
reverse	x = x(0, W-1)　(**??**see part select)
test	x[i]
set	x[i] = 1
clear	x[i] = 0
invert	x[i] = !x[i]
length	(template parameter)

- コンパイルフラグ_32BIT_についての解釈は以下のとおりである。
  - ✓ sc_int の最大ビット幅はデフォルトでは 64bit であるが、SystemC インストール時(コンパイル時)に–D_32BIT_を指定すれば、sc_int の最大ビット幅を 32bit にすることができる。しかし、このケースは LRM では言及されていないから、合成ツールではサポートしない。
- コンパイルフラグ MAX_NBITS についての解釈は以下のとおりである。
  - ✓ sc_bigint,sc_biguint のビット幅は任意であるが、SystemC インストール時(コンパイル時)に–DMAX_NBITS を設定すればビット幅を有限にでき、シミュレーション速度が向上する。MAX_NBITS がどのオペランドよりも大きなビット幅であれば、合成ツールは実装上の制限を考慮する必要はない。なお、LRM では MAX_NBITS については触れていない。

- (9) コンパイルフラグ_32BIT_の説明中の「LRM では言及されていない」という文章は削除すべきである。
- (10) コンパイルフラグ MAX_NBITS の説明中の「LRM では MAX_NBITS については触れていない」という文章は削除すべきである。

4.4.1.1 算術演算

- 減算、単項減算の演算結果は signed になる。
- 他の演算はオペランドの中で 1 つでも符号付き整数があれば、演算結果は signed になる。
- 他の場合は、unsigned になる。

4.4.1.2 ビットワイズ演算

- ~(補数演算)：ビット反転はサポートされる。
  - ✓ 例：~((sc_biguint<8>)128) = ~((sc_bigint<9>)128)=-129
- 論理積(&)、論理和(|)、排他的論理和(^)はサポートされる。
  - ✓ オペランドのどれかが signed の場合、他の unsigned の数値は符号ビットを追加して signed として扱ってから演算を行う。
  - ✓ 演算する値のビット長が違う場合、符号を含めて長いほうに合わせる。
- 結果は、両方のオペランドが unsigned なら unsigned、それ以外は signed になる。

4.4.1.3 比較演算

- 結果は bool 型になる。

4.4.1.4 シフト演算

- シフト値は C++のデータタイプになる。
- シフトは算術シフトである。
- sc_int/sc_uint に対するシフト値は 0 以上でなければならない。
- これは固定小数点型とは一貫性がない。
  - ✓ 固定小数点型では負のシフト値、つまり反対側にシフトも許されている。
- C の 64 ビット整数のシフトと同じ動作である。
  - ✓ シフト値が負または 64 以上の場合、C 言語での整数シフトに準拠、つまり実装依存ということになる。
- sc_bigint/sc_biguint に対する負のシフト値は、ゼロシフトになる。
  - ✓ 結局、sc_bigint/sc_biguint と sc_int/sc_uint では仕様に矛盾が存在する。
  - ✓ シフトが+/-が明確でない場合には、双方向シフトをサポートする回路を生成する場合がある。最小回路生成を意図するときは、unsigned を使用してシフトが一方向であることを示した方がよい。

- (11) sc_int/sc_uint 及び固定小数点型の負のシフト値の振る舞いについては、IEEE 1666-2005 でも未定義であった。まず SystemC の LRM 自体の修正を行い、その上で本ドキュメントもそれに合わせるべきである。
- (12) sc_bigint/sc_biguint に対する負のシフト値の振る舞いについては、IEEE 1666-2005 でも未定義であった。まず SystemC の LRM 自体の修正を行い、その上で本ドキュメントもそれに合わせるべきである。

4.4.1.5 代入演算

- 全てサポートされる

4.4.1.6 ビットセレクタ

- サポートされる。
  - ✓ 例：X[3] = y[2];
- 演算結果に対して（宣言した変数に代入しないで、テンポラリな変数としての値)のビットセレクトの使用した場合は以下のとおりになる。
  - ✓ sc_int/sc_uint はサポートしない。
  - ✓ sc_bigint/sc_biguint は算術演算子など一部サポートする。
- ビット範囲が[0,W-1]を超えた場合は以下のとおりになる。
  - ✓ sc_bigint/sc_biguint は超えてもよい。
  - ✓ sc_int/sc_uint は超えてはいけない。
  - ✓ 一貫性がない。LRM ではこの理由が明記されていないので解決すべき問題。

- (13) IEEE 1666-2005 では 7.2.5 Bit-Select の章で「超えてはいけない」と定義されているので、本ドキュメントもそれに合わせるべきである。

4.4.1.7 パートセレクト

- サポートされる。
  - ✓ x(5,3) = y(4,2);
- 演算結果に対して(宣言した変数に代入しないで、テンポラリな変数としての値)のビットセレクトの使用した場合は以下のとおりになる。
  - ✓ sc_int/sc_uint はサポートしない
  - ✓ sc_bigint/sc_biguint は算術演算子など一部サポートする
- 反転アクセス(例：x(0,3))は以下のとおりになる。
  - ✓ sc_bigint/sc_biguint はサポートされる。
  - ✓ sc_int/sc_uint はサポートされない。
  - ✓ 一貫性がない。LRM では明記されていないので解決すべき問題。
- ビット範囲が[0,W-1]を超えた場合は以下のとおりになる。
  - ✓ sc_bigint/sc_biguint は超えてもよい。
  - ✓ sc_int/sc_uint は超えてはいけない。

- ✓ 一貫性がない。LRM ではこの理由が明記されていないので解決すべき問題。
- 動的範囲の使用は、合成のために範囲の長さを静的に確定する必要がある合成ツールもある。

➢ (14) IEEE 1666-2005 では 7.2.6 Part-Select の章及びその NOTE 1-A で定義されているので、本ドキュメントもそれに合わせるべきである。

4.4.1.8 連結

- サポートされる。
  - ✓ 例：(x,y) = (z,w);

4.4.1.9 合成未サポートのメソッド

- 合成未サポートのメソッドは以下のとおりである。
  - ✓ リデュース・メソッド
    - ■ and_reduce,or_reduce.xor_reduce,nand_reduce,nor_reduce,xnor_reduce
    - ■ 任意精度データタイプに対してサポートされていないから。
    - ■ 本来はサポートされるべきである。
  - ✓ to_double()
  - ✓ set_packed_rep, get_packed_rep

➢ (15) リデュース・メソッドについては IEEE 1666-2005 ではサポートされたので、削除すべきである。

4.4.2 固定小数点型

- すべてのオーバーフロー・モードと量子化モードは合成対象である。

Overflow Mode	Parameters
Wrap-around Basic (*default*)	o_mode = SC_WRAP, n_bits = 0
Saturation	o_mode = SC_SAT
Symmetrical Saturation	o_mode = SC_SAT_SYM
Saturation to Zero	o_mode = SC_SAT_ZERO
Wrap-around Advanced	o_mode = SC_WRAP, n_bits > 0
Sign Magnitude Wrap-Around	o_mode = SC_WRAP_SM, n_bits ≥ 0

Quantization Mode	Parameter
Truncation (*default*)	q_mode = SC_TRN
Rounding to plus Infinity	q_mode = SC_RND
Truncation to zero	q_mode = SC_TRN_ZERO
Rounding to zero	q_mode = SC_RND_ZERO
Rounding to minus infinity	q_mode = SC_RND_MIN_INF
Rounding to infinity	q_mode = SC_RND_INF
Convergent rounding	q_mode = SC_RND_CONV

- 合成対象メソッドは以下のとおりである。

Methods	Alternatives
is_zero	x == 0
is_neg	x < 0
b_not	~
b_and	&
b_or	|
b_xor	^
Lshift	<<
Rshift	>>
Neg	unary -
Bit	x[i]
Range	x(i,j)
WI	template argument 1
Iwl	template argument 2
o_mode	template argument 3
q_mode	template argument 4
n_bits	template argument 5

- 合成対象演算子は以下のとおりである。

Op Category	Operators/Methods					
Arithmetic	+	-	*	/		
Assign	+=	-=	*=	/=		
Unary	+	-				
Auto Incr/dec	++ prefix	++ postfix	-- prefix	-- postfix		
Bitwise	&	|	^			
Assign	&=	|=	^=			
Unary	~					
Relational	==	!=	<	<=	>	>=
Shift	>>	<<	>>=	<<=		
Bit Select	[x]					
Part Select	(i,j)					
Conv to C integral	to_int	to_long	to_int64	to_uint	to_uint64	to_ulong
Assignment	=					

4.4.2.1 算術演算子

- 減算、単項減算の結果は常に signed になる。
- その他の演算では以下のとおりになる。
  - ✓ オペランドに 1 つでも signed があれば演算結果は signed になる。
  - ✓ オペランドに signed がなければ演算結果は unsigned になる。
- 割り算(/)について、演算結果として必要な精度についてオペランドの精度からは合成ツールにはわからない。
  - ✓ 明示的なキャストや代入の場合は必要な精度はわかる。

- ✓ それ以外の場合は、本当に必要な精度がわからないので、合成ツールが生成するハードウェアは必要以上に大きくなる場合がある。
- ✓ A/=B の場合、演算結果は A の精度となる

4.4.2.2 ビットワイズ演算子

- ~(補数演算)：仮数部をビット反転する。
  - ✓ 仮数部をビット反転するので SystemC の整数型とは結果が異なる。
  - ✓ 例：~((sc_ufixed<8,8>)128) != ~((sc_biguint<8>)128)
- 論理積(&)、論理和(｜)、排他的論理和(＾)は、signed、unsigned の混在は許されない。

4.4.2.3 比較演算

- 結果は bool 型になる。

4.4.2.4 シフト演算

- 負のシフト値は、逆方向にシフトとして扱う。
- 算術シフトとして扱う。
  - ✓ Note：シフト値の正負について
  - ✓ シフト値は、単方向シフト回路が生成されるため、unsigned にすべきである。
  - ✓ シフト値が signed の場合、双方向シフト回路を生成するか、単方向シフト回路で済ますかは合成ツールの解析能力に依存する。

4.4.2.5 ビットセレクト

- 制限付きでサポートする。
  - ✓ 例：x[3] = y[2];
- 演算結果に対して(宣言した変数に代入しないで、テンポラリな変数としての値)のビットセレクトの使用の場合、sc_fixed/sc_ufixed はサポートしない。

4.4.2.6 パートセレクト

- 制限付きでサポートする。
  - ✓ 例：x(5,3)=y(4,2);
- 演算結果に対して(宣言した変数に代入しないで、テンポラリな変数としての値)のビットセレクトの場合は、sc_fixed/sc_ufixed はサポートしない。
- パートセレクトの結果を固定小数点型の変数に直接代入はできない。(SystemC の仕様)
  - ✓ x = x(0, W-1); // サポートしていない
  - ✓ x(W-1, 0) = x(0,W-1); // サポートしている

- 反転レンジでのアクセスの際、レンジ幅[0,W-1]外にアクセスした場合、例外が発生する。
  - ✓ sc_bigint/sc_ubigint での仕様と異なる。(sc_bigint/sc_ubigint はレンジ幅を超えてもよい。)
- 「レンジ幅は静的に決定できる必要がある」という制約を合成ツールが持っていてもよい。

4.4.2.7 代入演算子

- 全ての定義済み代入演算子はサポートされる。

4.4.2.8 合成ツールがサポートしないメソッド及びオプション

- 合成ツールがサポートしないメソッド及びオプションは以下のとおりである。
  - ✓ overflow_flag(), quantization_flag(), type_param(), get_bit(), set_bit(), get_slice(), get_rep(), lock_observer(), unlock_observer(), observer_read(), value(), is_normal()
  - ✓ cast(), cast_switch(), observer()
  - ✓ キャストをオフするための SC_OFF オプション
  - ✓ SystemC コンパイル時に設定する SC_FXMAX_WL, SC_FIXDIV_WL, SC_FXCTE_WL
  - ✓ to_double(), to_float()

4.4.2.9 合成ツールではサポートしない型

- 合成ツールがサポートしない型は以下のとおりである。
  - ✓ sc_fixed_fast, sc_ufixed_fast
  - ✓ sc_fix, sc_ufix, sc_fix_fast, sc_ufix_fast, sc_fxcast_context, sc_fxcasr_switch
  - ✓ sc_fxval, sc_fxval_fast
  - ✓ sc_fxnum_observer, sc_fxnum_fast_observer, sc_fxval_observer, sc_fxval_fast_observer

4.4.3 ビットベクタ

- sc_bv<W>がサポートする演算子は以下のとおりである。

Op Category	Operators/Methods					
Bitwise	&	|	^			
Assign	&=	|=	^=			
Unary	~					
Relational	==	!=				
Shift/Rotate	>>	<<	>>=	<<=	lrotate(i)	rrotate(i)
Bit Select	[x]					
Part Select	(i,j)					
Concatenation	(,)					
Conv to C integral	to_int	to_long	to_int64	to_uint	to_uint64	to_ulong
Assignment	=					
Reduce	and_reduce	or_reduce	xor_reduce	nand_reduce	nor_reduce	xnor_reduce

- sc_bv<W>がサポートするメソッドは以下のとおりである。

Methods	Alternatives
length	template parameter
bit, get_bit	x[i]
set_bit	x[i]
range	x(i,j)
reverse	x = x(0, W-1)
b_not	~

4.4.3.1 ビットワイズ演算子

- ~単項演算子：各ビットを反転する。
- 2 項演算子(&、|、^)はサポートする。

4.4.3.2 比較演算子

- bool 型を返す。

4.4.3.3 シフト演算子と rotate メソッド

- 負のシフト値は禁止である。
  - ✓ 実行時に例外が発生する。
- 算術シフトではない。

4.4.3.4 ビットセレクタ

- 制限付きでサポートする。
  - ✓ 例：x[3]=y[2];
- ビット幅[0,W-1]を超えてのアクセスは禁止である。
  - ✓ 実行時に例外が発生する。

4.4.3.5 パートセレクト

- 制限付きでサポートする。
  - ✓ 例：x(5,3) = y(4,2);
- 反転レンジ指定もサポートする。
- ビット幅[0,W-1]を超えてのアクセスは禁止である。
  - ✓ 実行時に例外が発生する。

4.4.3.6 連接

- サポートする。
  - ✓ 例：(x,y) = (z,w);

4.4.3.7　代入演算子

- 全てサポートする。

4.4.3.8 リデュース・メソッド

- サポートするメソッドは以下のとおりである。
  - ✓ and_reduce(), or_redice(), xor_reduce(), nand_reduce(), nor_reduce(), xnor_reduce()
- 演算結果は bool 型になる。

4.4.4.1　sc_logic

- サポートする演算子は以下のとおりである。
  - ✓ ビットワイズ：&,|,^,~
  - ✓ 代入：=, &=, |=, ^=
  - ✓ 比較：==, !=

4.4.4.2　サポートされない論理定数

- 不定値“X”は合成ツールではサポートしない。
  - ✓ 例：sc_logic<W>(“X”)
- ただし、合成ツールは“X”を don't care として扱い、論理の最適化に使用してもよい。
- sc_logic<W>(“Z”)は port に直接代入されたときのみ使用可能。

don't care の記述例

```
if (x == 0x00)
          y = sc_logic<1>("0");
else if(x == 0x01)
          y = sc_logic<1>("1");
else if (x == 0x10)
          y = sc_logic<1>("0");
else
          y = sc_logic<1>("X"); //don't-care condition
```

4.4.4.3　未サポートメソッド

- サポートしないメソッドは以下のとおりである。
  - ✓ is_01(),to_bool(),value()
- b_not メソッドは~演算子があるので対応不要である。

4.4.4.4　任意長の論理ベクタ

- sc_lv<W>の可能な演算は sc_bv と同じである。

5.　宣言

5.1　宣言

- C++での宣言文はほとんどサポートする。
- extern 宣言はサポートしない。
- asm（アセンブラ記述ブロック）はサポートしない。

5.1.1 指定子

- friend は無視される。
- ➢ (16) friend は C++と同じ扱いにすべきである。
- typedef はサポートする。

5.1.1.1 記憶クラス指定子

- auto、regisiter、extern、mutable は無視される。
- static は制限付きでサポートする。（12.1.4.2 参照）

5.1.1.2 関数指定子

- virtual、explicit は無視される。
- inline はサポートする。

5.1.1.3 typedef 指定子

- サポートする。

5.1.1.4 friend 指定子

- ISO C++に準拠。

5.1.1.6 CV-qualifier(型修飾子）

- const はサポートする。
- volatile は合成時には無視される。
  - ✓ あってもなくても結果は同じである。

5.1.1.7 型指定子

- wchar_t、float、double はサポートしない。

5.1.1.8 エラボレートタイプ指定子

- サポートする。

5.1.1.9 SystemC タイプ指定子

- SystemC 拡張タイプ指示子はサポートする。

- ➢ (17) sc_bit は IEEE-1666 でなくなったため、削除すべきである。
- ➢ (18) 固定小数点はサポートすべきである。
- ➢ (19) P30 下（）を削除すべきである。

5.1.2 列挙型宣言

- サポートする。

5.1.3 asm 宣言

- サポートしない。

5.1.4 外部宣言

- サポートしない。

5.2 宣言子

- 制限付きでサポートする。
- 例外処理はサポートしない。
- ポインタは例外を除いてサポートしない。

➢ (20) 静的メモリのポインタはサポートすべきである。

5.2.1 タイプ名

- ポインタは例外を除いてサポートしない。

➢ (21) 静的メモリのポインタはサポートすべきである。

5.2.2 あいまい性の解決

- ISO C++に準拠。

5.2.3 パラメータ

- サポートする。

5.2.4 デフォルト引数

- サポートする。

5.2.5 イニシャライザ

- 制限付きでサポートする。
- モジュール内の変数は初期化子を持ってはいけない。
- モジュール・コンストラクタのメンバ初期化子によって初期化されてはいけない。
- モジュール内にクラス変数がある場合、基底クラスはデフォルトコンストラクタを宣言あるいは継承してはいけない。または、空のデフォルトコンストラクタを宣言あるいは継承する必要がある。

➢ (22) モジュール以外のクラスでのサポートを明確に記載すべきである。

5.2.5.3 参照

- ISO C++に準拠。

➢ (23) C++を参照していることを明記すべきである。

6. 式

- オペランド評価順序における副作用(ISO C++の 5 章)は以下のとおりである。
  - ✓ i = x[i++] の振る舞いについて、ISO C++では記載されていない。
- sizeof、new、delete 演算子はサポートしない。
- キャストは制限付きサポートである。
- 関数 typeid(ISO C++ 5.2.8 節)はサポートしない。

➢ (24) ISO C++ドキュメントの注が書かれているだけで、合成ツールでの扱いは規定していない。合成での扱い方を明記すべきである。(結果はコンパイラの実装依存になる)

7.1 関数定義

- 例外処理(try)はサポートしない。

7.2 関数本体

- wait()関数は SC_CTHREAD からコールする。
- wait()関数は SC_METHOD からはコールしてはいけない。

8. 文

- try block はサポートしない。

➢ (25) Statement の Syntax summary は次の形がよい。

```
statement ::=
        labeled-statement
      | expression-statement
      | compound-statement
      | wait-statement
      | signal-assignment-statement
      | selection-statement
      | iteration-statement
      | jump-statement
      | declaration-statement
```

8.1 ラベル構文

- ラベルは合成ツールに参照されるかもしれないが、デザインの機能には影響を与えない。

8.3 Wait 文

- 整数引数の wait 構文のみを使用しなければならない。(デフォルトは 1)

➢ (26) BNF では wait(constant-expression)となっており、上記文章と矛盾している。

➢ (27) スレッドプロセス SC_THREAD は、9.3 SC_THREAD でサポートしないと述べているので、ここで記述するのは間違いである。

➢ (28) wait_until は IEEE 1666-2005 で削除されているので、削除すべきである。

8.4 信号アサイン文

- ノート：合成ツールは、いくつかのエラーと一貫性をチェックする。
- 信号アサイメント文は、同じ信号やポートへ、1 つのプロセスからしか書き込みができない。ただし、resolved ベクタ型(sc_signal_rv、sc_in_rv、sc_out_rv、sc_inout_rv)は、複数プロセスから書き込みできる。
- 信号は 2 つのイベント間で 1 つの値のみアサインできる。

8.5 変数アサイン文

- Postfix 式は、lvalue が MUST である。
- 現在の値を新しい値に置き換える。
- 右側の式のタイプは、ターゲットタイプにマッチ or 含まれる必要あり。

➢ (29) C++ expression statement に含まれるので、本節自体を削除すべきである。

8.6.1 IF 文

- 条件によって文を選択する。

8.6.2 スイッチ文

- 式は整数型または列挙(enumeration)型である。
- 合成結果は、スイッチ文の記述スタイルで異なる。
- すべての取りうる選択がカバーされる場合（ディフォルトブランチを使含む)、スイッチ文は Full Case として扱う。
- すべての選択がブレーク文を含む場合、スイッチ文は Parallel Case として扱う。

➢ (30) 8.6、8.6.1、8.6.2 の各レベルがおかしいので修正すべきである。
➢ (31) 8.6.1 は 8.6.2 switch 文と同様の記載の仕方をすべきだし、8.6 は if (condition) statement と書くべきではなく、if statement とし、8.6.1 で詳細に記載すべきである。
➢ (32) switch 文の扱いは C++と同じにすべきであり、制約をつけるべきではない。

8.7 プロシジャーコール文

➢ (33) C++ expression statement に含まれるので、本節自体を削除するべきである。

8.8 繰り返し文

- C++の do と for の繰り返しは、C++と同じふるまいでサポートされる。

8.9 ジャンプ文

- 無条件で転送する。

➢ (34) goto の節が存在しないので、記載すべきである。

8.9.1 ブレーク

- ループやスイッチ文の完了を表す。
- ループやスイッチ文に続く文を続けて実行する。

➢ (35) タイトルを Break Statements とするべきである。

8.9.2 継続文

- ループの実行を完了する。

➢ (36) Break と Continue の説明が逆である。

8.9.3 復帰文

- ファンクション、またはプロシジャーの実行を完了する。
- 実行は、サブプログラムがコールされるところから、コールするサイトに戻される。
- プロセスの中ではなく、サブプログラムの中で起きなければならない。

8.10 宣言文

➢ (37) 説明がないので、Support を明記すべきである。

9. プロセス

- プロセス文の識別子は、同じ範囲(スコープ)でのプロセス定義を意味する。

➢ (38) 例に sc_bit を使っているが、IEEE 1666-2005 で削除されているので、bool を使用すべきである。
➢ (39) BNF の process は拡張であるので、網がけで表示すべきである。

- モジュールの各プロセスは、メンバ関数の宣言が最初に必要。
- プロセス定義の範囲は、モジュールでなければならない。
- プロセス定義は、プロセス定義があるモジュールのコンストラクタの中に、1 つ適切なプロセス文が必要である。
- 逆に、モジュールに属し、コンストラクタ本体内に関係するプロセス文を持つ、それぞれの関数定義は、プロセス定義と呼ばれ、この章にリストされた規則を守らなければいけない。

9.1 SC_METHOD

- SC_METHOD プロセス本体は、wait 文や、wait 文を実行する関数の呼び出しを含んではいけない。
- 展開できないループを含んではいけない。

- (40) sensitive_pos/sensitive_neg は IEEE 1666-2005 では Deprecated features になっているので削除すべきである。

9.1.1 合成セマンティク

- メソッドプロセス本体内で使用されるコーディングスタイルとセンシティビティリストによって、純粋な組合せ回路と同様に順序回路を記述可能である。

組合せ回路の記述例

```
SC_MODULE( L_AND ) {
           sc_in< sc_logic > in_a, in_b;
           sc_out< sc_logic > output;

           void comb() { // This process describes purely combinational logic.
                  // It is "virtually" level-triggered
           output.write( in_a.read() & in_b.read() );
           }

           SC_CTOR( L_AND ) {
               SC_METHOD( comb );
               sensitive << in_a << in_b;
           }
};
```

順序回路の記述例

```
SC_MODULE( Counter ) {
           sc_in< bool > clk;
           sc_in< bool > rst_n;
           sc_in< bool > enable;
           sc_out< unsigned int > val;

           unsigned int countVar;

           void seq() {    // This process describes sequential logic.
               if ( !rst_n.read() ) {
                   countVar = 0;
               }
               else if ( enable.read() )
               {
                   countVar += 1;
                   val.write( countVar );
               }
           }

           SC_CTOR( Counter ) {
               SC_METHOD( seq );
               sensitive_neg << rst_n;
               sensitive_pos << clk;
           }
};
```

9.2 SC_CTHREAD

- SC_CTHHREAD の構文は、プロセスの終了をしないようにする必要がある。
- これを達成する一般的な方法は、無限ループを使用する。
- 各無限ループは、少なくとも 1 つコントロールパス内に wait()が必要である。
- 最初の wait()の前の動作は、リセット動作とみなす。

- 次の構文は SC_CTHREAD に対して提案され、合成ツールは、少なくともこれらの構文をサポートする必要がある。上記要求を満たす他の構文もサポートすべきである。

➢ **(41) watching 文は IEEE-1666 では Deprecated features なので、reset_signal_is に変更すべきである。**

記述例 1：シンプルな記述例

```
// Simple SC_CTHREAD structure for synthesis
void process() {
           // reset
           reset_behavior(); // must be executable in a single cycle
           wait();          // first wait implies end of reset

           // infinite loop
           while (true) {
               rest_of_behavior(); // must contain 1 wait per control path
           }
}
```

記述例 2：無限ループ内にリセットがある場合

```
// Reset reaches into infinite loop
void process() {
           // reset
           reset_behavior(); // must be executable in a single cycle

           // infinite loop
           while (true) {
               // everything located here is also executed during a reset
               wait();         // first wait() in process
               rest_of_behavior(); // must contain 1 wait per control path
           }
}
```

記述例 3：初期化部も合成する場合

```
// SC_CTHREAD structure for synthesis, with ""initialization part""
void process() {
           // reset
           reset_behavior(); // must be executable in a single cycle
           wait();
           initialization(); // may contain any number of wait()s. This part
                    // is only executed once, after a reset.
           // infinite loop
           while (true) {
               rest_of_behavior(); // must contain 1 wait per control path
           }
}
```

9.3 SC_THREAD

- サポートされない。

➢ **(42) サポートすべきである。**

10. サブモジュールインスタンス

- モジュールインスタンスを示す文字列は、モジュール・コンストラクタには必要であるが、合成ツールに対しては意味がなく、任意に選ぶことができる。

➢ (43) BNF の網かけが必要である。

記述例

```
SC_MODULE(MyModule) {
            sc_in_clk    CLK;
            sc_in<bool>  RST;
            sc_in<int>  a;
            sc_in<int>  b;
            sc_out<int> c;
            sc_out<bool> RDY;
            sc_signal<int> tmp;

            Adder add;
            GCD *gcd;

            SC_CTOR(MyModule): add("add") {
                add(a,b,tmp);
                gcd = new GCD("GCD");
                gcd->CLK(CLK);
                gcd->RST(RST);
                gcd->x(tmp);
                gcd->y(b);
                gcd->z(c)
                gcd->RDY(RDY);
            }
};
```

11. ネームスペース

- 非 const の global/shared 変数は合成ではサポートされない。
- Function body の中では、あらかじめ定義された変数やパラメータとして渡された変数しか使用できない。
- Global constant はサポートされる。

11.1.1 ネームスペース定義

- サポートされる。

11.1.2 Unnamed なネームスペース

- サポートされる。

11.1.3 ネームスペースのメンバ定義

- サポートされる。

11.1.4 ネームスペースのエイリアス

- サポートされる。

11.1.5 using による定義

- サポートされる。

11.1.6 using ディレクティブ

- サポートされる。

12. クラス

- 制限付きのサポート。union はサポートされない。（class と struct のみ）

12.1.1 クラスの名前

- ISO C++に準拠。

12.1.2 クラスメンバ

- （44）説明がないので、説明を記載すべきである。

12.1.3 メンバ関数

- サポートされる。（ただし class あるいは struct のメンバとして定義されたものに限る。）

12.1.3.1 非スタティックメンバ関数

- サポートされる。

12.1.3.2 this ポインタ

- this ポインタのみがサポートされる。

12.1.4 スタティックメンバ

- 制限付きでサポートされる。

12.1.4.1 スタティックメンバ関数

- 他の関数と同じ制限とコーディングガイドラインが適用される。

- （45）static member functions 7 章を参照する形で記載すべきである。

12.1.4.2 スタティックデータメンバ

- const static データメンバのみがサポートされる。

12.1.5 union

- サポートされない。

12.1.6　ビットフィールド

- サポートされる。

12.1.7　ネストされたクラス定義

- サポートされる。

12.1.8　ローカルクラス定義

- サポートされる。

12.1.9　ネストされたタイプ名

- ISO C++に準拠。

12.2　派生クラス

- サポートされる。

12.2.1　多重基底クラス

- サポートされる。

12.2.2　メンバ名ルックアップ

- ISO C++に準拠。

12.2.3　仮想関数

- サポートされる。

12.2.4　抽象クラス

- サポートされる。

12.3　メンバアクセス制御

- ISO C++に準拠。

12.3.1　アクセス制御指定

ISO C++に準拠。

12.3.2　基底クラス及び基底クラスメンバのアクセス制御

- ISO C++に準拠。

12.3.3　アクセス定義

- ISO C++に準拠。

12.3.4　フレンド

- ISO C++に準拠。

12.3.5　保護されたメンバのアクセス

- ISO C++に準拠。

12.3.6　仮想関数へのアクセス

- ISO C++に準拠。

12.3.7　多重アクセス

- ISO C++に準拠。

12.3.8　ネストされたクラス

- ISO C++に準拠。

12.4　スペシャルメンバ関数

- 制限付きでサポートされる。

12.4.1　コンストラクタ

- signal、port もしくは module のデータメンバに関して constructor は無視する。
- これらに関しては reset で初期化すべきで、そうでない場合は結果を保証しない。

12.4.2　テンポラリオブジェクト

- ISO C++に準拠。

12.4.3　コンバージョン

- ISO C++に準拠。

12.4.3.1　コンストラクタによるコンバージョン

- ISO C++に準拠。

12.4.3.2　コンバージョン関数

- ISO C++に準拠。

12.4.4　デストラクタ

- サポートされる。

12.4.5　フリーストア

- サポートされない。

12.4.6　初期化

- 制限付きでサポートされる。
- 非 const のモジュールメンバは、mem-initializers で初期化されてはいけない。
- RESET に入れる。

12.4.6.1　明示的初期化

- サポートされる。

12.4.6.2　基底とメンバの初期化

- サポートされる。

12.4.7　コンストラクションとデストラクション

- コンストラクションについては C++と同じルールが適用される。
- デストラクションはサポートされないので、デストラクションオブジェクト、すなわちテンポラリやローカルなクラスインスタンスは、目に見える効果はない。

➢ (46)「Without support for destructors, destruction of objects, i.e. temporaries and local class instances, does not have any visible effect.」の箇所のデストラクションの説明が不明であるので、詳細を記載すべきである。

12.4.8　クラスのコピー

- ISO C++に準拠。

13.　オーバーロード

- 制限付きのサポート。(いくつかの演算子はオーバーロードできない。また、コーディングスタイルに制限があるものもある)

13.1.1　オーバーロード定義

- ISO C++に準拠。

13.1.2　定義マッチング

- ISO C++に準拠。

13.1.3　オーバーロードリゾリューション

- ISO C++に準拠。

13.1.3.1 候補関数と引数リスト

- 以下のものはサポートされる。(ISO C++に準拠。)
  - ✓ Function call syntax, Call to named function, Call to object of class type, Operators in expressions, Initialization by constructor, Copy-initialization of class by user-defined conversion, Initialization by conversion function, Initialization by conversion function for direct reference binding

13.1.3.2 二次候補関数

- ISO C++に準拠。

13.1.3.3 最良二次候補関数

- 以下のものはサポートされる。(ISO C++に準拠。)
  - ✓ Implicit conversion sequences, Standard conversion sequences, User-defined conversion sequences, Reference Binding, Ranking implicit conversion sequences
- 以下のものはサポートされない。
  - ✓ Ellipsis conversion sequences

13.1.4 オーバーロード関数のアドレス

- アドレス演算はサポートされない。

13.1.5 オーバーロードされた演算子

- 次の演算子はサポートされない。
  - ✓ new | delete| new[] | delete[]
- signal 及び port タイプのためのクラスでは operator== を定義すること。
- operator== は、ターゲットとソースのタイプが異なる場合には false を返す。

13.1.5.1 単項演算子

- ISO C++に準拠。

13.1.5.2 2 項演算子

- ISO C++に準拠。

13.1.5.3 代入

- ISO C++に準拠。

13.1.5.4 関数呼び出し

- ISO C++に準拠。

13.1.5.5 添字付け

- ISO C++に準拠。

13.1.5.6　クラスメンバ・アクセス

- ISO C++に準拠。

13.1.5.7　インクリメント、デクリメント

- ISO C++に準拠。

13.1.6　組み込み演算子

- ISO C++に準拠。

14.　テンプレート

- 全てサポートする。

15.　プリプロセッサ・ディレクティブ

- 全てサポートする。
- マクロを定義可能である。
- _STDC_、_cplusplus はツール実装依存になる。
- SC_SYNTHESIS マクロに合成サブセットのバージョンを設定する。

16.　字句要素

- 制限なし。（ISO C++に準拠。）

17.　SCOPE AND VISIBILITY

- 制限なし。（ISO C++に準拠。）
- 合成ツールは access specifier を認識できる必要がある。ただし、無視してよい。

18.　その他

18.1　トレース

- 合成ツールは以下を認識できる必要がある。ただし、無視してよい。
  - ✓ sc_trace_file*宣言、sc_trace_file*への代入
  - ✓ sc_close_vcd_trace_file、sc_close_isdb_trace_file、sc_close_wif_trace_file
  - ✓ sc_close

18.2　標準出力へのメッセージ出力

- 合成ツールは以下を認識できる必要がある。ただし、無視してよい。
  - ✓ printf
  - ✓ cout <<
- printf、cout 中で値を変更してはいけない。
  - ✓ 例：printf(“y = %d, ++y)は禁止である。

18.3 例外処理

- サポートされない。

ANNEX A 構文規則サマリ

- C++の IEEE 規格の構文規則に対し、合成用に使用できるものとできないものを区別し、さらに合成可能な SystemC 固有のライブラリを構文規則の中に組み入れている。
- 構文規則(Syntax)の記法は以下のとおり。

a) Roman フォントの小文字とハイフォンを用いた単語は構文要素(syntax category)を示す。
b) ボールドフォントを用いた単語は予約語である。
c) 記号::=の左側は右側で置き換えることができることを示す。
d) 記号｜はその前後のどちらか 1 つを選択することを示す。
e) 記号[ ] で囲まれた項目は、1 回現れるか、なくてもよいことを示す。
f) 網掛け文字の単語は追加された SystemC 固有の規則を構成する。
g) 斜体の単語は構文規則そのものではない。(コメント)
h) 横線で消されている単語は合成でサポートされない(ツール依存である)ことを示す。

- (47) a)の中の“space take the place of underlines”は、“space take the place of hyphens”の誤りである。
- (48) 無視 Ignore(下線)を説明がないので追加すべきである。

A.1 キーワード

- ISO C++に準拠。

A.2 字句規則

- 文字(literal)として特殊記号を使うことはできない。英数字とアンダーバーのみ使うことができる。
- 文字列(string literal)は、サブモジュールの名前としてだけ使われる。

- (49) sc_object すべてに名前としての文字列を与えられるようにすべきである。また、文字列が定数に変換される用法もあるのでサポートすべきである。
- (50) 文字列は、char 型配列の定数として扱うべきである。

A.3 基本概念

- 構文解析の開始点として sc-main-definition が追加された。

- (51) sc-main-definition が網掛けされていない。

A.4 式

- デストクタ、メンバのポインタ指定、typeid、sizeof、new、delete は使えない。

- (52) デストラクタを使用できるようにすべきである。
- (53) 静的メモリのメンバはポインタ指定可能にすべきである。

A.5 文

- wait-statement と signal-assignment-statement を追加している。

- (54) variable-assignment-statement と procedure-call-statement の追加は不要であるので削除すべきである。
- (55) switch 文の制限は削除すべきである。

A.6 宣言

- wchar_t、float、double の型は宣言できない。
- asm と extern は宣言できない。

A.6-1 SystemC タイプ指示子

- sc-process-definition、sc-type-specifier、sc-module-specifier が追加されている。
- SystemC のデータタイプのうち sc_int、sc_uint、sc_bigint、sc_biguint、sc_logic、sc_lv、sc_bit、sc_bv、sc_fixed、sc_ufixed が宣言できる。

- (56) IEEE 1666-2005 に従えば、sc_bit は不要であるので、削除すべきである。

A.7 宣言子

- 参照は宣言できるが、ポインタが宣言できない。

- (57) 静的メモリのポインタは宣言できるようにすべきである。

A.8 クラス

- class と struct は定義できるが、union は定義できない。

A.8.1 モジュール宣言

- モジュールの定義が追加されている。

A.9 派生クラス

- ISO C++に準拠。

A.10　特別なメンバ関数

- ISO C++に準拠。

A.11　オーバーロード

- new、delete、new[ ]、delete[ ]以外の演算子はオーバーロード可能。

A.12　テンプレート

- ISO C++に準拠。

A.13　例外処理

- まったくサポートされない。

A.14　プリプロセッサ

- ISO C++に準拠。

4.2.4.2 動作合成スタイルガイドの構成要件

1. はじめに

SystemC が IEEE1666 として標準化され、高位設計ツールもこの標準に準拠する形で整備が進みつつある。高位設計の中でも動作合成ツールは、機能のハードウェア化における設計期間の短縮や高位レベルでの設計資産蓄積と流通による設計の高品質化を実現する技術として期待が高い。JEITA 主催の EDS フェア(Electronic Design and Solution Fair)のシステム・デザイン・フォーラムとして開催している SystemC ユーザ・フォーラムの参加者アンケートでは、SystemC 活用への期待の問いに対しては動作合成ツールの充実が、言語拡張／標準化への期待の問いに対しては動作合成サブセット定義が、毎年一番高い票を得ている。特に、2007 年のアンケート結果では、コーディングスタイル・ガイドラインの充実要求が急に高くなっているのが、変化として目立っている。

OSCI(Open SystemC Initiative)からは 2006 年 6 月に合成サブセットのレビューリリースが行われ、2007 年内には標準化が行われるものと期待している。OSCI 合成サブセットのレビューに対しては、SystemC タスクグループからも 57 件の改善・要望事項を提出しており、IEEE1666 に準拠した合成サブセットドラフトが作成されて、早期に標準化に進むよう OSCI 及び IEEE へ協力していくつもりである。OSCI 合成サブセットは、動作合成に限らず合成に使用可能な構文を示したものであり、動作合成を設計に使用するときに必要となる、記述スタイルや設計者の回路合成の意図をツールに指示する仕組みを共通化するものではない。

動作合成ツールは複数社から提供されているが、合成用の記述や合成制約の指示の仕方がそれぞれ異なっており、合成ノウハウが広く共有され効果的な活用方法が確立していくことや、それに伴い設計データの流用性が高まって効率的な設計が進められる環境ができるには、まだ時間がかかる状況にある。

SystemC タスクグループでは、SystemC の普及を推進するために、高位設計において期待が高い動作合成ツールの利用方法を共通化することで設計生産性を高めることを目的として、動作合成スタイルガイドの構成要件を定めることとした。動作合成技術の活用による設計効率化を実現するには、動作合成スタイルガイドそのものを規定することが求められるが、SystemC タスクグループは SystemC の標準化とその普及を主たる目的としていることと、動作合成ツールの特徴がそれぞれ異なっている中で、1 つの方法に決めるにはまだ時期尚早との判断から、各ツール提供会社や設計コンサルティング会社などで作成される動作合成ガイドラインが共通の指針で作成され、結果として動作合成向けの記述や設計制約が共通化されることを期待して、構成要件を規定することにした。

2. スタイルガイド構成要件とは

動作合成を活用するためには、SystemC の合成サブセットを定義するとともに、その活用方法を示すガイドラインが必要である。そのガイドラインが作成者によってばらばらに構成されてしまうと、利用する側は動作合成スタイルガイドを全て読み通さないと、その活用方法を理解できないこととなり、導入時の負担が大きくなる。このため、動作合成スタイルガイドで示すべき

項目とその中で説明すべき内容を示して、スタイルガイドを効果的に活用しやすくすることと、動作合成とその活用方法が設計マネージャや設計者に容易に理解できて、導入の可否や活用方法の検討を進めやすくすることを目的として、スタイルガイドの構成を規定して、各章で説明すべき内容について説明している。

本構成要件の説明の中では明示していないが、動作合成ガイドラインは、動作合成ツールによらず使用できる設計スタイルや記述スタイルを示して共通化することを目指しているので、できる限り動作合成ツール非依存を前提として記述されることを期待している。各動作合成ツールの特徴となっている機能の説明については、付録や別冊の形でまとめられることが望ましい。

また、スタイルガイドの規約では、現実に活用可能なガイドラインとするために、必須、推奨、参考のレベル分けを行い、重要度がわかるように記載することで、スタイルガイドに柔軟性を持たせることが必要である。

3. 動作合成スタイルガイドの構成

3.1 構成の章立てと説明

動作合成スタイルガイドは、ハードウェア設計者で SystemC 初心者や SystemC を機能検証などで使っているが動作合成はこれからという人を主な対象者とする。また、導入部については、動作合成ツールの設計への採用を判断する設計マネージャが読むことも想定している。内容としては、動作合成の基本的な機能や SystemC による高位設計フローの概説と機能検証について説明する。設計フローは、アルゴリズムが C/C++などで用意されているブロックレベルの設計を例として、動作合成の位置付けや使い方がわかるように説明する。動作合成については、ベンダ非依存で必要となる基本的な回路構成方法と SystemC による記述方法や動作合成ツールへの合成制約の指示や合成結果の検証方法を、順を追って説明する。スタイルガイドの前半を読むことで、動作合成の位置付け、機能、使い方、効果などの基本的な処理が理解できて、設計プロジェクトへの適用性を判断するための情報が一通り得られるように作成されるべきである。

構成としては、大きく分けて 3 つのパートで構成する。導入部としては、マネージャや動作合成を未経験の設計者に動作合成の仕組みと効果を説明し、動作合成ツールと設計者の役割分担を正しく認識させる。続く初級編では、動作合成を用いた一通りの設計作業ができるレベルまでの説明を行う。SystemC 記述を例示して、記述スタイルや合成制約の与え方を具体的に説明しながら、動作合成を使った基本処理や合成前後の機能検証方法について説明する。さらに上級編では、実際のチップ設計に動作合成を適用するときに必要となるモジュール構成の考え方や、いろいろなタイプのパイプライン処理の扱い方など動作合成を効果的に使うテクニックの説明を行う。また、合成サブセットの詳細や、再利用性や合成後の性能を向上させる記述スタイルについても解説して、必要に応じてツール依存の記述スタイルも説明する。

具体的な章立てとしては、以下の構成を推奨する。

＜導入部＞

第 1 章　概論

第 2 章　動作合成とは

＜初級編＞

第 3 章　動作合成のための記述スタイル

第 4 章　検証環境

＜上級編＞

第 5 章　動作合成を考慮した SoC 設計

第 6 章　RTL との対応

第 7 章　上級テクニック

第 8 章　SystemC の合成サブセット

付録/別冊　ツールの特徴(ツールに独自な特徴となる機能について説明)

3.2　各章の詳細説明

ここではスタイルガイドの各項目で記載すべき内容について説明する。

3.2.1 「第 1 章　概論」

この章では、高位設計の進展や動作合成が活用されるようになってきた背景とその使い方を、設計者や設計マネージャが理解できることを目的としている。この章を読むことで、動作合成の特徴が把握できて、設計工程に動作合成を取り込むかどうかを適切に判断するポイントがわかるように説明する。

以下の項目が含まれて記載されていることとする。

(1)　高位設計が必要となってきた背景

半導体技術の微細化の進展に伴う SoC 設計の大規模化や機能の複合化により、高い抽象度でシステム全体を見渡した設計が必要になってきていることや、ハードウェアとともにソフトウェアの果たす役割が大きくなっているなかで、SoC を手戻りなく実現するための設計技術、設計手法の進展や変化について説明する。必要に応じて、ツールの紹介や基本的な機能の比較などが含まれてもよい。

(2)　C/C++及びその拡張言語である SystemC を使った高位設計フロー

高位設計の活用には、アルゴリズムからの設計、機能仕様書／設計仕様書からの詳細化、既存 RTL の再設計(改版、作り直し)など、いろいろな形態があるが、ここでは一般的な設計フローとして、C 言語などで記述されたアルゴリズムからハードウェア及び組み込みソフトウェアで実現される実装レベルまでの設計の手順を示す。各設計段階におけるモデリングとしての UT(UnTimed)や TLM(Transaction Level Model)の説明や、各レベルでの設計作業及び機能検証方法について概要が記載されていることが望ましい。アルゴリズムから実装までの設計フローの説明を通して、設計フローにおける動作合成の位置付けと使い方が理解できるようにすることが目的である。

(3) 動作合成の特徴と効果

動作合成については、動作合成そのものの説明に力点を置くのではなく、動作合成により得られるメリットと注意すべき点を説明することで、動作合成の特質(得失)がわかるように説明する。

例えば、動作合成による設計期間や工数の削減、及び再利用性の向上などによる設計効率化が期待できる点だけではなく、動作合成を利用するために準備作業が必要なことや、適用する機能ブロックが動作合成の処理に適切でないと、回路サイズが大きくなったり設計の手間が増える危険性についても説明して、動作合成の採否の判断とその活用方法を検討するのに必要なポイントが理解できるように記述する。

動作合成の詳細は 2 章で説明するので、それぞれの内容についての説明はここでは必要ない。動作合成採用にあたって、マネージャが検討すべき項目が把握できるように記載すること。

3.2.2 「第 2 章　動作合成とは」

この章は、マネージャや動作合成未経験の設計者に

- 動作合成の仕組みと効果
- 動作合成ツールと設計者の役割分担
- 動作合成の設計フローの中での位置づけ

の 3 点を詳細に理解できることが必要である。そのため、簡単な例も入れて具体的にわかりやすく説明していただきたい。記載内容として以下の 3 点が含まれている必要がある。

(1) 動作合成の概要

動作合成ツールに入力する設計データに記載される設計情報は何かと、合成結果として得られるデータが何かについて、説明する。例えば、入力設計データであれば、回路情報としてインタフェースとアルゴリズムがあり、合成制約としてレイテンシやスループットがあることなどを説明する。

動作合成の基本機能としてスケジューリングと資源(演算器とレジスタ)の分配(リソースシェアリング)を行い、結果としてデータパスとコントローラが生成されることを説明する。これらの機能が、従来の RTL 設計に比べてどのような利点を生み出しているか述べる。

(2) 動作合成を使うために設計者がすべきこと

動作合成を使うために設計者がすべきこと(役割)を明確にする。一般に、マネージャや動作合成未経験の設計者は過大にツールに期待する傾向があるので、誤解を少なくするために、動作合成がやってくれることと動作合成ができないこと(設計者がしないといけないこと)をはっきりさせる。

動作合成が C/C++のアルゴリズムからそのまま最適なハードウェアを作成すると考えている読者に、例えば、以下のような設計者の作業は残されていることを指摘する。

- 合成に適したサイズ・独立性・並列性を考慮したモジュールの分割
- 転送速度・配線数などを考慮したモジュール間インタフェースの選択
- 全体最適を考慮した各処理の回路規模とレイテンシの決定
- メモリかレジスタかの選択とメモリのアドレスマップ作成

(3) 設計フローの中での動作合成

設計フローの中で、動作合成に至るまでに設計者が行っておかなければならないこと、動作合成終了後に確認しなければならないこと、動作合成後に必要になる設計作業などを説明する。これらにより、設計フローの中での動作合成の使い方を説明する。

3.2.3 「第 3 章　動作合成のための記述スタイル」

この章では、初めて動作合成を行う設計者が一通りの動作合成作業が作業できるようなレベルを目標として、合成対象回路の SystemC のモデリング手法と最適化のための合成制約について解説を行う。全体的には代表的な回路例や記述例を中心に解説を行い、様々なバリエーションや回路のチューニングテクニック等は上級編にて記載することとする。記載内容としては、以下の 3 点が含まれていることとする。

(1) 基本的な動作合成向き SystemC の記述スタイル

動作合成対象回路のモジュール記述の基本形を解説する。含まれる内容としては、以下の様なものがあげられる。

- モジュール記述／プロセス記述／ポート構成
- SC_CTHREAD プロセスの定義方法とリセット動作の実装方法
- よく使うデータタイプの解説と定数の記載方法

これらはテンプレートとして用いられることを想定した、実際の SystemC 記述例と共に解説することが望ましい。

(2) 動作合成を実施する際の合成制約の設定方法

動作合成対象に対する合成制約について、最低限必要なもの／設定すべきものに対して、その制約が合成実行時に回路に与える意味と、その設定方法について解説する。なお合成制約はツール依存性が強い部分ではあるが、なるべく一般化して解説することが望ましい。

(3) 動作合成対象回路の回路構成

動作合成で回路を生成する場合に、特に考慮すべき回路構成について、その動作と実装記述例の解説を行う。含まれるべき回路構成としては以下のようなものがあげられる。

- パイプライン構造のデータパス回路の構成の実装方法
- 合成対象回路と他のモジュールとのインタフェース回路の実装方法(ピン実装やクラス化)やハンドシェークの例
- 動作合成回路が使用するメモリ(RAM)の実装方法や取り扱い方

なお、これらの回路構造について様々な場合を記述するのではなく、代表的でかつ比較的シンプルな回路構成に絞り本章で解説し、詳細は上級編にて解説すべきである。

3.2.4 「第 4 章　検証環境」

この章は、

- 動作合成の前後で行う検証における考え方と注意すべき点

について述べる。

ここでは検証項目には立ち入らずに、検証の仕組み(テストベンチ、期待値比較、合成前後の対応)などの一般的な説明をする。ただし、従来の RTL 設計におけるブロック単体検証でやっていることは含めない。動作合成を行った場合に特に必要となる項目に重点を置いて説明する。例えば、次のステップで動作合成の前後の検証を進める場合の検証環境の例をつける。

- C/C++でリファレンスモデルを作成し、動作合成対象の SystemC モデルの検証を行う。
- 動作合成対象の SystemC モデルと、動作合成の結果得られた RTL との一致検証を行う。

記載内容としては以下の内容を含めること。

(1)　合成前と合成後で起こりうる不整合の例と、あらかじめ作りこんでおくべき検証環境の仕組みや、レイテンシ、リセット動作、不定の扱いの違いが出る可能性があることを説明し、それらの対策として、FIFO などを活用できることや、期待値比較のタイミングの考慮等を記載する。

(2)　合成前にやっておく検証の例

機能検証は一通りやっておく。(ライン)カバレッジ等で、機能検証の十分性を確認しておく。カバレッジとしては、ラインカバレッジやステートカバレッジを測定するとよいことを紹介する。

(3)　合成後の検証の進め方

合成前の C/C++リファレンスモデルと SystemC モデルの検証環境(テストベンチ)の活用方法について記載する。その他、SystemC と RTL の混在シミュレーション環境(Co-Sim)を活用できることを説明するとよい。(深入りはせず、スクリプト例を示す程度とする。)

3.2.5 「第 5 章　動作合成を考慮した SoC 設計」

この章は、

- 動作合成を考慮したブロック分割の考え方
- 動作合成したブロックを SoC へ組み込むためのテクニック

の 2 点を理解してもらうためにある。

記載内容としては以下の内容が含まれている必要がある。

(1) 動作合成を考慮したブロック分割の考え方

ブロック分割を検討する際に考慮しなければならない動作合成特有の考え方について解説する。以下の内容を考慮した考え方として解説することが望ましい。

- ブロック分割しなければ動作合成できない回路仕様
- ブロック分割による回路の品質への影響
  (プロセス間でのリソース共有の制限事項や注意事項)
- ブロック分割単位の動作合成実行時間への影響
- 設計容易性及び検証容易性を損なわないブロック分割方法
- バックエンドフローとの整合性を考慮したブロック分割方法

(2) 動作合成したブロックを SoC へ組み込むためのテクニック

動作合成するブロックをバスと接続して SoC へ組み込むためのインタフェース設計方法について解説する。

特に、CPU から読み書きされるレジスタインタフェースについては、演算のコアとインタフェースを分けた構成での設計方法及びレジスタを動作合成するブロック内に置く場合とブロック外に置く場合での得失の解説を行う。

3.2.6 「第 6 章　RTL との対応」

SystemC での RTL の記述や階層記述をこの章で説明する。合成向けには Verilog-RTL と同等な注意事項が考えられる。詳細説明を Verilog-RTL 論理合成のスタイルガイドに譲ってもよい。ただし、その場合は SystemC と Verilog-RTL の対応付けについて明確に説明する必要がある。

(1) METHOD プロセスの定義と RTL 記述

METHOD プロセスは、sensitive に並べる信号の選び方によって、組合せ回路または順序回路を意味することを示す。合成はされるけれども LSI 化で問題となることの多いラッチの生成などの注意点や、合成とシミュレーション結果が変わる可能性の高い記述についての注意を行うこと。sc_signal 信号に代入する場合とそうでない変数に値を代入する場合の SystemC での扱いの違いと合成結果についても説明する。

(2) 階層記述

SystemC での階層の記述方法について説明する。HDL では容易に記述できるけれども SystemC では冗長になる、バスの一部の信号のポートへの接続や、複数信号を連接して 1 つのポートに接続する場合の記法についても説明する。

(3) プロセス間のデータの受け渡し

複数プロセスがアクセスする信号はポートか sc_signal でなければならないことを説明する。さらにこれらの信号にアクセスする上での注意点を説明する。

3.2.7 「第 7 章　上級テクニック」

この章ではこれ以前の章で初級向きではないとして説明されていない記述スタイルや設計手法について記載する。例えば、下記のような項目を上級テクニックと見なし、これ以前の章の要件から除いている。これらは、この章で説明されることを推奨する。

(1) パイプラインの詳細説明

パイプラインの実行間隔（インターバル）が 2 クロック以上になる場合があることについての解説を行う。また、パイプラインのデータ入出力インタフェースにおいて、データが転送されないことによりストールが生じて実行間隔が伸びる場合についての注意などの説明を行う。

(2) モジュラ・インタフェース

インタフェースをパッケージ化して扱うことで、インタフェースの共通化をはかり設計ミスを最小化するとともに、設計データの流用性を高める方法について解説する。現時点ではモジュラ・インタフェースが適切な方法と考えているが、これに限定するものではない。

(3) メモリの詳細説明

メモリをモジュール内部おいてそのまま合成する場合と、モジュール外部に出す場合の合成方法の違いを解説する。メモリをモジュール内部に内蔵するか外部に置くかの違いやインタフェースの違いを、クラスを用いるなどしてアルゴリズムから分離する手法を説明する。ROM の利用方法もここで解説する。

3.2.8 「第 8 章　SystemC の合成サブセット」

この章では、動作合成で使用できる SystemC のデータ型や演算、制御などに使用する構文を説明する。

動作合成で使用できる構文と使用できない構文がわかるように説明することが必要で、文法／構文の側面から記載することが望ましい。一般的な構文説明だけではなく、動作合成で使用するときの制約条件や基本的な使用例を交えたわかりやすい説明があるとなおよい。

現在、OSCI で合成サブセットが策定中であるが、正式に公開された場合は、これを参照することや、OSCI 合成サブセットに対して補足説明を付加する形で記載することでもよい。

3.2.9 「付録/別冊　ツールの特徴」

基本的にはツールによらず同じ記述方法で適切な合成結果が得られることで、設計データの蓄積や資産化が可能になり、その記述方法や使い方が設計スキルとして設計者間で共有できることが望ましいが、動作合成ツールには、サポートする機能や扱える構文及びその解釈の仕方がツールによって異なる場合がある。動作合成スタイルガイド構成要件は、ツールに依存しない形で作成することを前提として作成しているが、現実に設計の中で動作合成ツールを使う場合には、そのツールが持っている機能を十分活用することが必要になる。このような、動作合成一般に共通

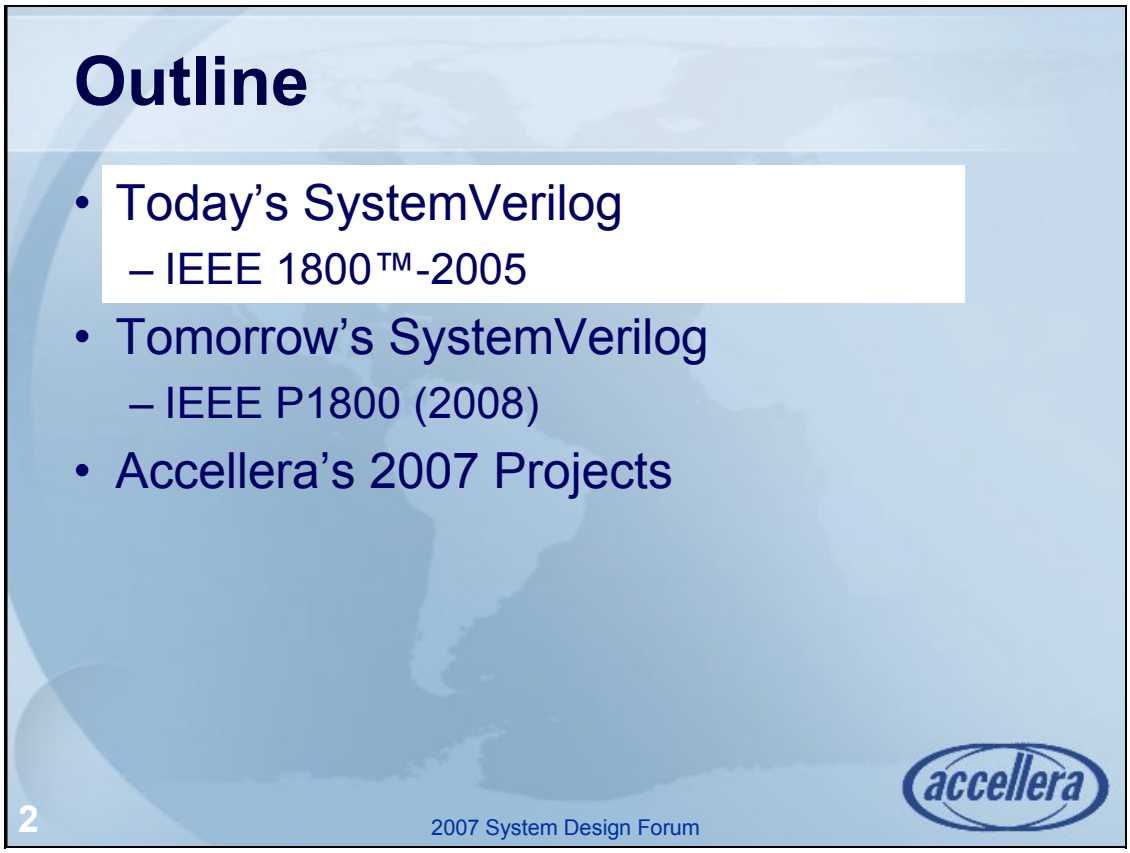
ではないが、使用するツールの特徴となっている機能や記述方法及び使い方などの説明は、付録または別冊として必要に応じて記載する。

4. 最後に

本構成要件は、SystemC 言語を対象として、動作合成を活用するために必要なスタイルガイドの構成要件を説明しているが、利用可能な動作合成ツールには、C/C++など他の言語を入力としているものもある。本書の構成要件はそのような動作合成ツールにおいても有効であると考えているので、SystemC 以外の言語を動作合成の入力としている場合でも、本構成要件に基づいたスタイルガイドが作成されることを期待している。

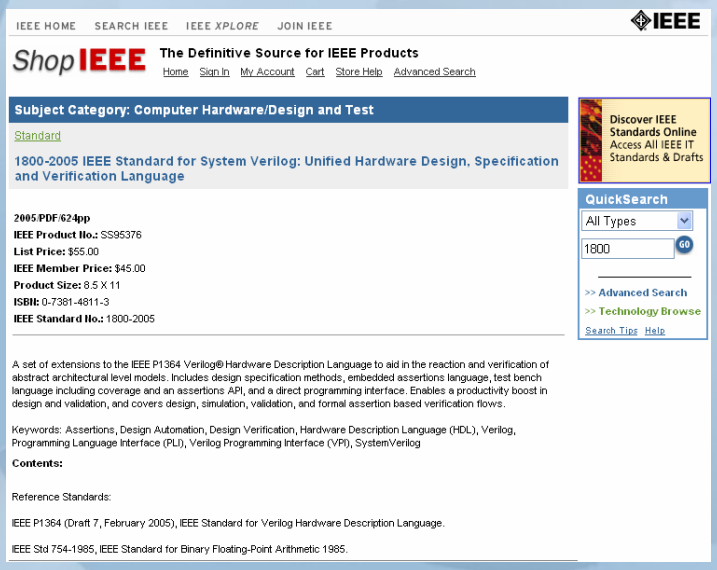
また、合成制約を指示する形式やブロック間インタフェースを共通化することで、データの再利用性が高まり設計資産として蓄積しやすくなるとともに、それをリファレンスとして設計知識の共有化を進めることができるので、今後 OSCI/IEEE などの標準化の場で決められることを期待している。

4.3 SystemVerilog タスクグループ報告



Available Now from IEEE

http://shop.ieee.org/ieeestore/Product.aspx?product_no=SS95376



Price:

USD $55.00

IEEE Member Price:

USD $45.00

SystemVerilog Support

- 2004
  - 6 companies with 9 products（6社 9製品）
- 2005
  - 45 companies with 91 products（45社 91製品）
- 2006
  - 109 companies with more than 280 products（109社 280以上の製品）

- Source
  - `http://www.systemverilog.org/products/products_solu.html`
  - `http://www.synopsys.com/partners/systemverilog/systemverilog_partners.html`
  - `http://www.mentor.com/products/fv/partners/vanguard_program.cfm`

SystemVerilog Suppliers（提供者）

- Ace Verification LTD
- Adveda
- Alatek
- Aldec, Inc.
- Amiq
- ARM
- Atrenta Inc.
- Averant
- Avery Design Systems
- Axiom
- Beach Solutions
- Blue Pearl Software
- Bluespec
- Cadence
- Calypto Design Systems
- CCL Softlabs
- ChipVision Design Systems
- Computer Based Education
- ControlNet India
- Correct Designs
- D&R
- Denali
- Doulos
- eInfochips
- Esterel Technologies
- EVE
- Expert I/O
- Flexody
- FMF
- ForteLink Inc.
- GDA Technologies
- Globetex
- HD Labs
- HDL Design House
- I Technology
- Ingot Systems
- Interra Systems
- Intrinsix
- IPextreme
- J. van der Schoot Design
- Jasper Design Automation
- JEDA Technologies
- Kacper Technologies
- Level 5
- LOA Technology
- Logic Research
- Magma
- Mentor Graphics
- Mirafra Technologies
- MU Electronics
- N Square Corporation
- Nobug
- Novas Software
- nSys
- Oki Network LSI
- Paradigm Works
- Perfectus Technology PerfTrends
- Poly Space
- Posedge Software,
- Productivity Design Tools
- Project VeriPage Inc.
- psi Electronics
- Real Intent, Inc.
- Sasken
- Scaleeo
- SDV
- Sequence
- SiConcepts, Inc.
- Silicomotive Solutions
- Silicon Interfaces
- Silicon Vision
- Simantis
- SK Electronics
- Spike Technologies
- Summit
- Sunburst Design
- Sutherland HDL
- SynaptiCAD
- Synopsys
- Syosil
- Tenesix
- Temento
- Tenison EDA
- Tera Systems
- Test Insight
- Tharas Systems
- TNI Valiosys
- TransEDA
- Trustic
- Vericine
- Veriex
- VeriEZ Solutions, Inc.
- Verific Design Automation
- Verification Consulting
- Verification Technology
- Verilab, Ltd.
- Verirfication General
- Verisure
- Veritools
- VGVP
- VhdlCohen Publishing
- Vtech
- Willamette HDL, Inc.
- WinterLogic Inc.
- WSFDB Consulting
- XtremeEDA Corporation
- Yogitech

Outline

- Today's SystemVerilog
  - IEEE 1800™-2005
- Tomorrow's SystemVerilog
  - IEEE P1800 (2008)
- Accellera's 2007 Projects

New Officers Elected

- Chair – Karen Pieper, Synopsys
- Vice-Chair – Neil Korpusik, Sun Microsystems
- Secretary – Dennis Brophy, Mentor Graphics

- Entity-based IEEE project
  - One company, one vote

新たに選ばれた役員

- 主査 – Karen Pieper, Synopsys
- 副主査 – Neil Korpusik, Sun Microsystems
- 書記 – Dennis Brophy, Mentor Graphics

- 団体ベースによる IEEE プロジェクト
  - 一社につき一票の制度

Standardization Timeline*

- IEEE Approves New PAR
  - June 2006
- 1364 & 1800 Merged Working Draft
  - Before DAC 2007
- First sponsor ballot
  - Before DAC 2008
- Ratified IEEE Standard
  - December 2008

*Timeline subject to change

標準化スケジュール

- IEEE による新規 PAR の承認
  - 2006年 6月
- 1364 & 1800 マージの作業草案
  - 2007年 DAC の前
- 最初のスポンサーによる投票
  - 2008年 DAC の前
- IEEE による標準の承認
  - 2008年 12月

Timeline subject to change

IEEE P1800 Updated Scope

- SystemVerilog 1800 is a Unified Hardware Design, Specification and Verification language. Verilog 1364-2005 is a design language. Both standards were approved by the IEEE-SASB in November 2005.
- This standard creates new revisions of the Verilog 1364 and SystemVerilog 1800 IEEE standards.
- Includes
  - Errata fixes and resolutions
  - General Enhancements
  - SystemVerilog Assertion language enhancements
  - Merge of Verilog 1364 LRM and SystemVerilog 1800 LRM into a single LRM
  - Integration with AMS
  - Insures interoperability with other languages such as SystemC and VHDL.

IEEE P1800 アップデートの範囲

- SystemVerilog 1800 はハード設計、仕様、検証が統一された言語、Verilog　1364－2005 は設計言語。どちらも2005年 11月にIEEE-SASB によって承認されている
- この標準によって Verilog 1364 と SystemVerilog 1800 IEEE標準の新しいバージョンとなる
- 以下の内容が含まれる
  - 正誤表の解決
  - 一般的な改善
  - SystemVerilog Assertion 言語の改善
  - Verilog 1364 と SystemVerilog 1800 をマージしたーつのLRM
  - AMS の統合
  - SystemC や VHDL などの言語との相互接続技術の確立

IEEE P1800 Updated Purpose

- The purpose of this project is to provide the EDA, Semiconductor, and System Design communities with a solid and well-defined IEEE Unified Hardware Design, Specification and Verification standard language, while resolving Errata and developing enhancements to current SystemVerilog 1800 IEEE standard.
- The language is designed to co-exist, be interoperable, possibly merge, and enhance those hardware description languages presently used by designers.

IEEE P1800 アップデートの目的

- このプロジェクトの目的は、EDA、半導体企業、そしてシステム設計の業界に、IEEE による明確なハード設計、仕様、そして検証を統一した明確な標準言語を提供するとともに、既存の SystemVerilog 1800 IEEE 標準の正誤の解決や改善をもたらすことである
- 新言語は、現在設計者に使われているハードウェア設計言語と共存し、相互接続を可能とし、さらに融合する可能性があり、そして既存言語を改善するように開発される

Outline

- Today's SystemVerilog
  - IEEE 1800™-2005
- Tomorrow's SystemVerilog
  - IEEE P1800 (2008)
- Accellera's 2007 Projects

Accellera Members

- Aldec, Inc.
- **ARM Ltd.**
- **Cadence Design Systems**
- **Denali Software Inc.**
- **Freescale Semiconductor**
- **IBM**
- **Intel Corporation**
- Jasper
- JEITA
- L-3 Communications
- **Magma Design Automation**
- **Mentor Graphics**
- **Nokia**
- **Novas**
- Real Intent
- **Rockwell Collins**
- Silicon Canvas Inc.
- Silvaco
- ST Microelectronics
- **Sun Microsystems**
- **Synopsys Inc.**
- **Texas Instruments**
- Tharas Systems
- Toshiba
- Xilinx

… and over 4,000 Designers Forum members
（4,000 以上の設計者によるフォーラムの会員）

BOLD: Accellera Board Member

Accellera Technical Committees（技術委員会）

- SystemVerilog
- Property Specification Language (PSL)
- VHDL
- Open Verification Library (OVL)
- Verilog AMS
  - Compact Modeling Extensions
- Interface Technology (ITC)
- Open Compression Interface (OCI)
- Unified Power Format (UPF)
- UCSI - Unified Coverage Interoperability Standard (*newly formed*)
- PSL – Property Spec Language

Project Status & Roadmap

	2003	2004	2005	2006	2007	2008
SystemVerilog	3.1	*3.1a*	*IEEE 1800*	*NEW PAR*		1800.V07
OVL Verilog/SVA PSL VHDL SystemC			• *OVL 1.0* • *Kicked off* • *Kicked off*	OVL 1.6 – 1.8	OVL 2.0	
VHDL			*Formation* *VHDL 1.0*	VHDL 3.0 VHPI 2.4		
OCI			*Formation*	OCI 1.0		
Verilog-AMS	2.1	•*2.2*	*2.3* *Verilog-AMS*	*3.0* *SystemVerilog* *-AMS*		
ITC (SCE-API)	1.0	*1.1*	2.0	2.0		
UPF				First Draft		
PSL	1.01	*1.1*	*IEEE 1850*	*NEW PAR*		1850.V08


accellera
Thank You
ありがとうございました
www.accellera.org
2007 System Design Forum

IEEE Std.1800-2005
（SystemVerilog)
テストベンチ
チュートリアル

JEITA　SystemVerilog Task Group

Agenda

- SystemVerilog タスク・グループ紹介
- 検証テクノロジの現状
  - 検証テクノロジの乱立
  - 基本的なテストベンチの役割
  - 最近の検証トレンド
  - IEEE Std. 1800-2005（SystemVerilog）概要
- テストベンチの説明
  - コンストレイント・ランダム・スティミュラス
  - カバレッジ
  - レスポンス
  - テストベンチ、検証環境の再利用
  - program & clocking block
  - クラス

SystemVerilogタスク･グループ

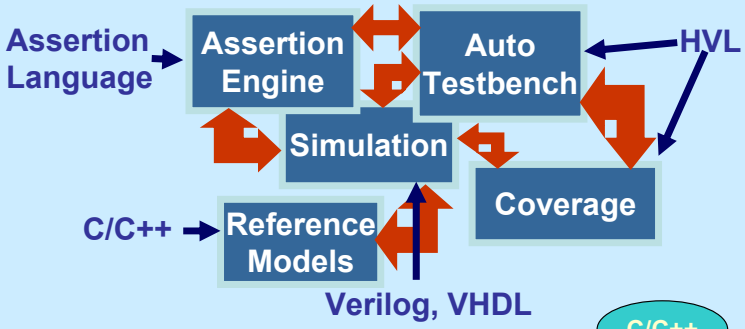
- JEITA：社団法人 電子情報技術産業協会
- 「JEITA EDA技術専門委員会／標準化小委員会傘下の SystemVerilog Task Group (SV-TG)
  - SystemVerilogの国際標準化活動に日本から参画
  - メンバー企業 10社

沖ネットワークエルエスアイ	日本シノプシス
三洋電機	松下電器産業
図研	メンター･グラフィックス･ジャパン
東芝	富士通
日本ケイデンス･デザイン･システムズ社	ルネサステクノロジ

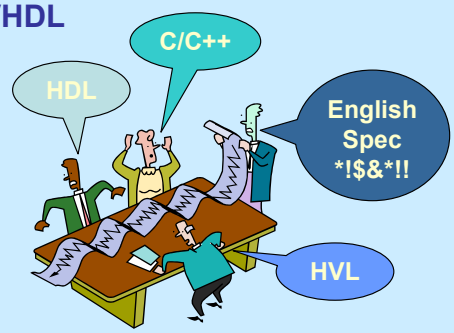
(注)五十音順, 敬称略

JEITA (3) SystemVerilog

検証テクノロジの乱立



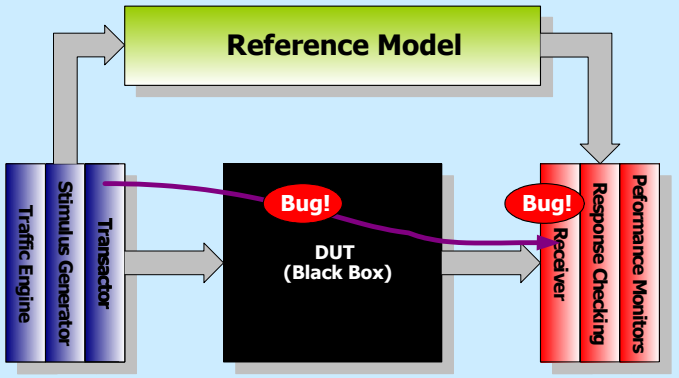
- 検証の早期完了のための新しい手法
- ツール専用の検証言語の乱立
  - 習得が困難
  - ポータビリティの欠如
    - 技術革新の妨げ
    - ツール価格の上昇



JEITA (4) SystemVerilog

基本的なテストベンチの役割

- バグの発見のためには. . .
  1. スティミュラスを加える
  2. 出力で結果を観測する
  3. 不一致があれば、デバッグする
- 検証を完了させるためには. . .
  - 全てのテスト項目について 1.～3. を実施する



JEITA (5) SystemVerilog

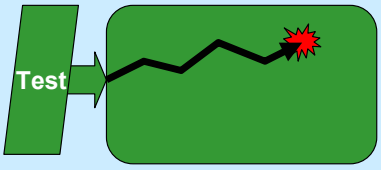
最近の検証トレンド

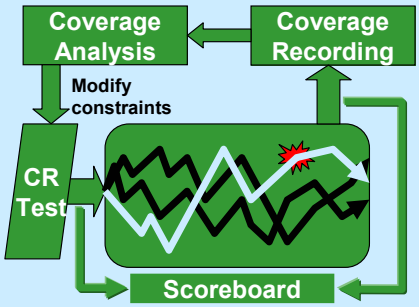
- カバレッジ・ドリブン検証
  - コンストレイント・ランダムスティミュラス生成
  - カバレッジ covergroup/cover
  - レスポンス（目視チェックではありません）
    - アサーション assert
    - スコアボード
- テストベンチ、検証環境の再利用
  - 部品化
  - 再利用

JEITA (6) SystemVerilog

カバレッジ・ドリブン検証

- ダイレクト・テスト
  - テスト作成者による手作業
    - 意図した事象のシナリオを1つ1つコーディング
  - 制御性の問題
    - 入力スティミュラスのみで、深いサイクルでの回路動作を予測しなければならない
  - 調整がきかない
    - 条件の異なるテストの生成にはさらに手作業が必要
- カバレッジ・ドリブン検証
  - コンストレイント・ランダムスティミュラス生成
    - 制約の修正 ＝ テスト生成
  - ダイレクト・スティミュラス
    - 到達しにくいコーナーケース
  - ファンクショナルカバレッジ
    - デザインの動作シナリオは予測不可
    - 意図した事象を捉える仕組みが必要
  - スコアボード
    - 予測できないシナリオの結果をチェック可能

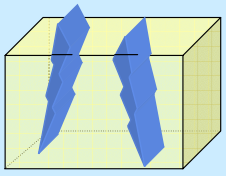


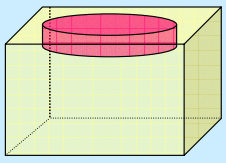


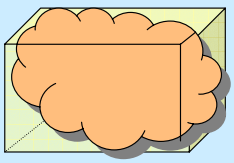
JEITA (7) SystemVerilog

ランダムスティミュラスの導入

- 直接指定（ダイレクト検証）
  - 全て手作業による記述
  - 到達しにくいコーナーケースのカバーが可能
  - 人手によるテストケースの作成には限界がある
- ランダムスティミュラス
  - テストパターン生成の自動化（記述は簡単）
  - 広く浅いカバレッジ
  - 冗長なテストを行う可能性がある
- コンストレイント・ランダムスティミュラス
  - テストパターン生成の自動化
    （記述は比較的簡単、制約を数式で記述）
  - 広く深いカバレッジ
  - 冗長性を排除し、多くの機能をカバー
  - 人手による規則性（偏り）を排除


設計機能領域

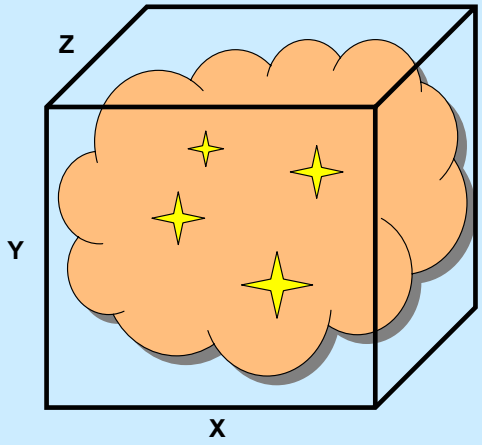

設計機能領域


設計機能領域

JEITA (8) SystemVerilog

Constraint Solver ― ランダム制約の解決

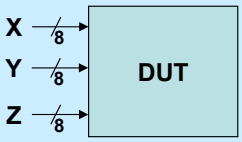
- ランダム変数の宣言 – X, Y, Z
- 制約の宣言 – Y < 42, X ≤ Y ≤ Z
- 与えられた制約を満たす変数値の集合を求める
- 解空間からランダムに値を取出す



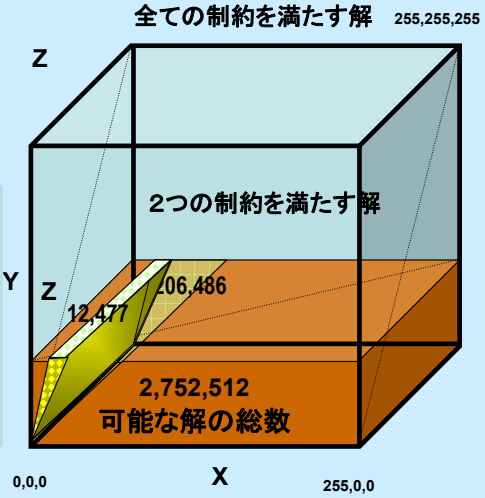
JEITA (9) SystemVerilog

解空間の計算

```
# 21 26 32
# 1d 22 91
# 04 0e d0
# 0f 28 60
# 14 20 d1
# 07 1f 80
```



制約	可能な解の総数
制約なし	$2^{8+8+8} = 2^{24} = 16{,}777{,}216$
Y < 42	$42 \times 2^{8+8} = 2{,}752{,}512$
X ≤ Y ≤ Z	$\sum_{k=1}^{256}(\sum_{n=1}^{k} n) = 2{,}829{,}056$
X[7:4]–Z[3:0]	$2^4 \times 2^{8+4+4} = 1{,}048{,}576$



JEITA (10) SystemVerilog

SystemVerilog の概要

- IEEE Std. 1364-2001 (Verilog HDL)の拡張
- IEEE Std. 1800-2005 標準
- デザインと検証のために言語を統合
  - HDVL (Hardware Description and Verification Language)
- SystemVerilog はVerilog HDL の新バージョン!

JEITA (11) SystemVerilog

スティミュラスのランダム生成

randomize()メソッド、rand、randc宣言でランダム値を生成する

キーワード	機能
randomize()	ランダム値を生成する
rand	変数が通常のランダム値の生成することを宣言する
randc	変数がcyclicなランダム値の生成することを宣言する "cyclic"とは、例えば2ビットの信号なら4サイクルで0,1,2,3を1回づつ発生させること

```
class Transaction;
  rand  bit [31:0]  Addr;
  randc bit [1:0]   BurstType;
  rand  bit [4:0]   DataSize;
endclass
Transaction t1 = new();
initial begin
  repeat(100) begin
    t1.randomize();
  end
end
```

```
Addr=7337dd17 BurstType=0 DataSize=0e
Addr=30d4b41b BurstType=1 DataSize=15
Addr=cac8ccaa BurstType=3 DataSize=14
Addr=05e23536 BurstType=2 DataSize=02
 ・
 ・
```

0～3を1回ずつ発生

JEITA (12) SystemVerilog

ランダム生成の制約

constraintブロックでランダム生成に制約条件をつける

キーワード	機能
>、>=、<、<=、==	値の大小関係、等しいことの指定
inside	生成値の範囲の制約
->	条件判定

```
class Transaction;
  rand  bit [31:0]  Addr;
  randc bit [1:0]   BurstType;
  rand  bit [4:0]   DataSize;
  constraint c0  {
    Addr inside { [32'h0 : 32'hFFF] };
    Addr[1:0] == 2'b00;
    (BurstType != 2'b01) -> (DataSize == 5'h00);
    (BurstType == 2'b01) -> (DataSize >= 5'h10);
  }
endclass
```

```
Addr=00000b98 BurstType=1 DataSize=1f
Addr=00000024 BurstType=3 DataSize=00
Addr=00000e40 BurstType=2 DataSize=00
Addr=00000840 BurstType=1 DataSize=12
Addr=0000057c BurstType=0 DataSize=00
 ・
 ・
```

・アドレス範囲が0～0xFFF
・アドレス下位2ビットが0

BurstTypeが1の時のみデータサイズを0x10以上

JEITA

SystemVerilog

発生確率の重み付け

キーワードdistでランダム発生確率に重み付けを行う

キーワード	機能
dist	生成値の発生確率への重み付けを行う

```
class Transaction;
  rand  bit [31:0]  Addr;
  randc bit [1:0]   BurstType;
  rand  bit [4:0]   DataSize;
  rand  bit         Lock;
  constraint c0  {
    Addr inside { [32'h0 : 32'hFFF] };
    Addr[1:0] == 2'b0;
    (BurstType != 2'b01) -> (DataSize == 5'h00);
    (BurstType == 2'b01) -> (DataSize >= 5'h10);
    Lock dist {0:=80, 1:=20};
  }
endclass
```

```
Lock=0 Addr=0000057c BurstType=0 DataSize=00
Lock=0 Addr=00000b98 BurstType=1 DataSize=1f
Lock=0 Addr=00000024 BurstType=3 DataSize=00
Lock=1 Addr=00000e40 BurstType=2 DataSize=00
Lock=0 Addr=0000092c BurstType=0 DataSize=00
 ・
 ・
 ・
```

Lock=1となる確率が20%

JEITA

SystemVerilog

ランダムシーケンスの生成

randsequenceでランダムシーケンスを生成する

キーワード	機能
:=	リスト内プロダクション(シーケンス要素)の選択確率を重み付ける
if…else/case	プロダクションの選択条件を指定する
repeat(n)	プロダクションをn回繰り返す
rand join	選択したプロダクションの順序を変更しない

基本の記述例

```
randsequence( main )
  main : first second done ;
  first : add := 3 | dec := 2 ;
  second : pop | push ;
  done : { $display("done"); } ;
  add : { $display("add"); } ;
  dec : { $display("dec"); } ;
  pop : { $display("pop"); } ;
  push : { $display("push"); } ;
endsequence
```

3シーケンスがランダムに発生

```
add pop done
add push done
dec push done
dec pop done
```

repeatの例

```
randsequence()
  ...
  PUSH_OPER : repeat( 2 ) PUSH ;
  PUSH : ...
endsequence
```

rand joinの例

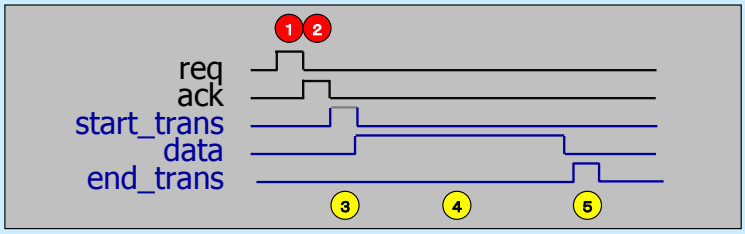
```
randsequence( TOP )
  TOP : rand join S1 S2 ;
  S1 : A B ;
  S2 : C D ;
endsequence
```

"AB"、"CD"の順序はそのまま

```
A B C D
A C B D
A C D B
C D A B
C A B D
C A D B
```

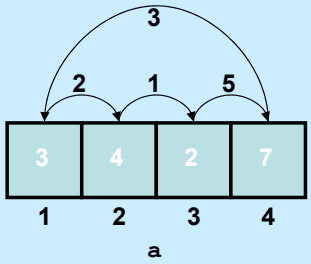

coverステートメントによるカバレッジ

- coverは、シーケンス、プロパティに適用できる
  - assertと同じシーケンスやプロパティの発生を記録
  - 例えば、バスプロトコルの信号シーケンスをカバレッジ計測するのに適している



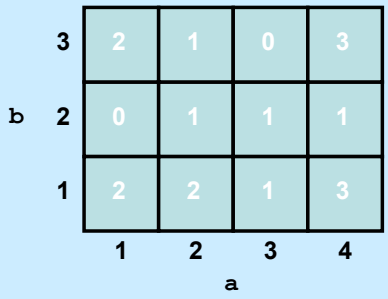
covergroup によるカバレッジ・モデル

- 変数、エクスプレッションの値に対するカバレッジ
  - テストにおいて発生するデータ値に関する情報を記録
  - 値の変化(トランジション)も定義可能
  - 例えば、テストで生成された RAM アドレスのカバレッジなどに適している



単純なカバレッジ: a==1 は何回?

トランジションカバレッジ:
a==1 の後に a==2 となったのは何回?



クロスカバレッジ:
a==1 の時に b==1 は何回?

JEITA (17) SystemVerilog

機能カバレッジ情報の収集

covergroup ブロックにカバレッジポイントを記述する

キーワード	機能
coverpoint	収集する機能カバレッジポイントを指定する
bins	coverpointで指定した情報をどのビン(=容器)に容れるかを指定する 複数のビンが指定可能
iff	coverpointのサンプル条件を指定する

```
covergroup addr_region @(posedge clk);
  coverpoint cp_addr {
    bins    bank0 = {['h0000_0000:'h0000_03FF]} iff(!reset);
    bins    bank1 = {['h0000_0400:'h0000_07FF]} iff(!reset);
    bins    bank2 = {['h0000_0800:'h0000_0BFF]} iff(!reset);
    bins    bank3 = {['h0000_0C00:'h0000_0FFF]} iff(!reset);
  }
  coverpoint cp_burst {
    bins    BurstType0 = {2'b00} iff(!reset);
    bins    BurstType1 = {2'b01} iff(!reset);
    bins    BurstType2 = {2'b10} iff(!reset);
    bins    BurstType3 = {2'b11} iff(!reset);
  }
endgroup
```

JEITA (18) SystemVerilog

クロスカバレッジ情報の収集

キーワードcrossによりクロスカバレッジ情報を収集する

キーワード	機能
cross	クロスカバレッジ情報を収集するカバレッジポイントを指定する

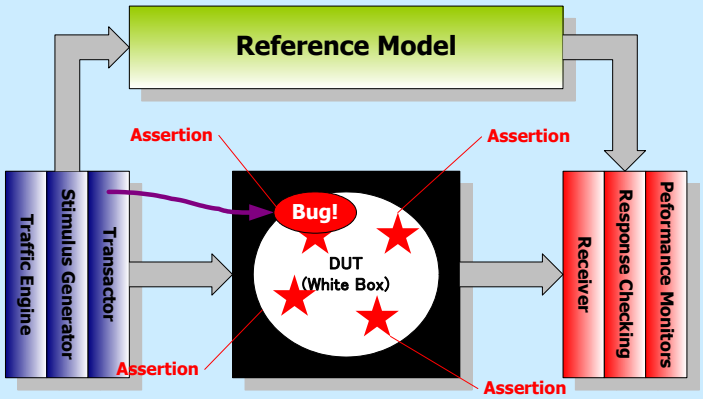
```
covergroup addr_region @(posedge clk);
 coverpoint cp_addr {
    bins    bank0 = {['h0000_0000:'h0000_03FF]} iff(!reset);
    bins    bank1 = {['h0000_0400:'h0000_07FF]} iff(!reset);
    bins    bank2 = {['h0000_0800:'h0000_0BFF]} iff(!reset);
    bins    bank3 = {['h0000_0C00:'h0000_0FFF]} iff(!reset);
  }
  coverpoint cp_burst {
    bins    BurstType0 = {2'b00} iff(!reset);
    bins    BurstType1 = {2'b01} iff(!reset);
    bins    BurstType2 = {2'b10} iff(!reset);
    bins    BurstType3 = {2'b11} iff(!reset);
  }
  addr_burst : cross cp_addr, cp_burst;
endgroup
```

```
cp_addr  cp_burst     # hits
=========================
bank0    BurstType0   4
bank0    BurstType1   3
bank0    BurstType2   5
bank0    BurstType3   2
 .
 .
bank3    BurstType0   2
bank3    BurstType1   3
bank3    BurstType2   3
bank3    BurstType3   3
=========================
```

JEITA (19) SystemVerilog

レスポンスのチェック

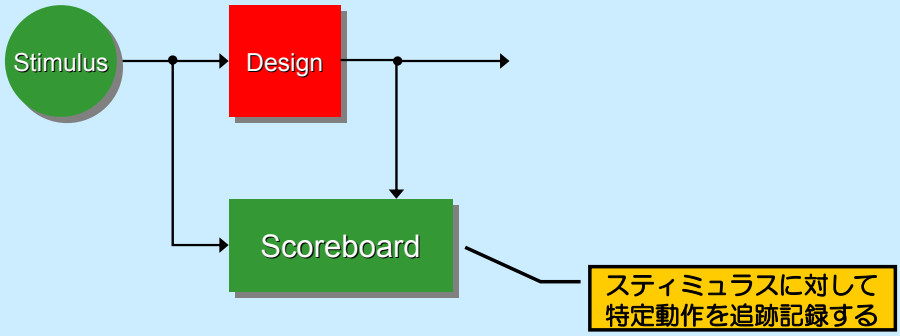
- レスポンスをチェックする目的
  - 不正動作の検出
  - cover ステートメントによるカバレッジ計測
- 実現方法
  - アサーション
  - スコアボード



JEITA (20) SystemVerilog

スコアボード

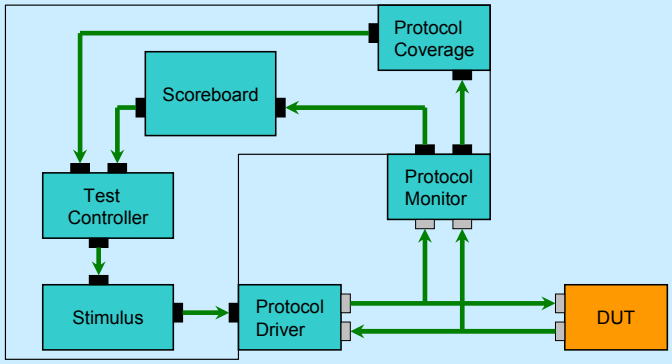
- スティミュラスに対する DUT の入出力を追跡記録
- 実装はアプリケーション固有だが、パターンは一般化が可能
- 記録手段：メールボックス、連想配列、キュー、など



JEITA (21) SystemVerilog

テストベンチ、検証環境の再利用

- 部品化
  - コンポネントのカプセル化
  - プロトコルのカプセル化
- 再利用
  - 一貫したトランザクションインタフェース
  - 構築済みインフラストラクチャー
  - 高い抽象レベル



JEITA (22) SystemVerilog

プログラム・ブロック

- 簡単な記述例

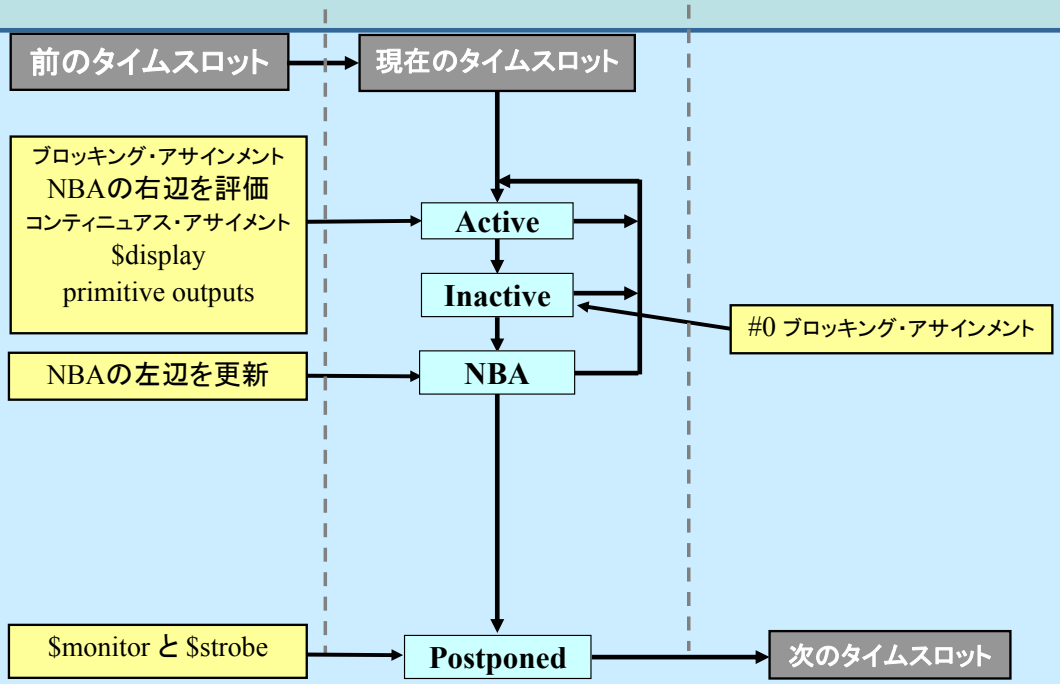
```
program test (input clk, input [16:1] addr, inout [7:0] data);
    initial ...
endprogram
```

  - テストベンチのコードを明確化し、カプセル化する
  - 厳密なモデリング・スタイルを強要する
- module との主な相違点
  - Initial ブロックのみ（always, モジュールインスタンスは不可）
  - プログラム・ブロックの変数は外側から見えない
  - ブロッキングとノンブロッキングのアサインメントに関するルール
    - プログラム・ブロックの変数に対してはブロッキングのみ
    - それ以外に対してはノンブロッキングのみ

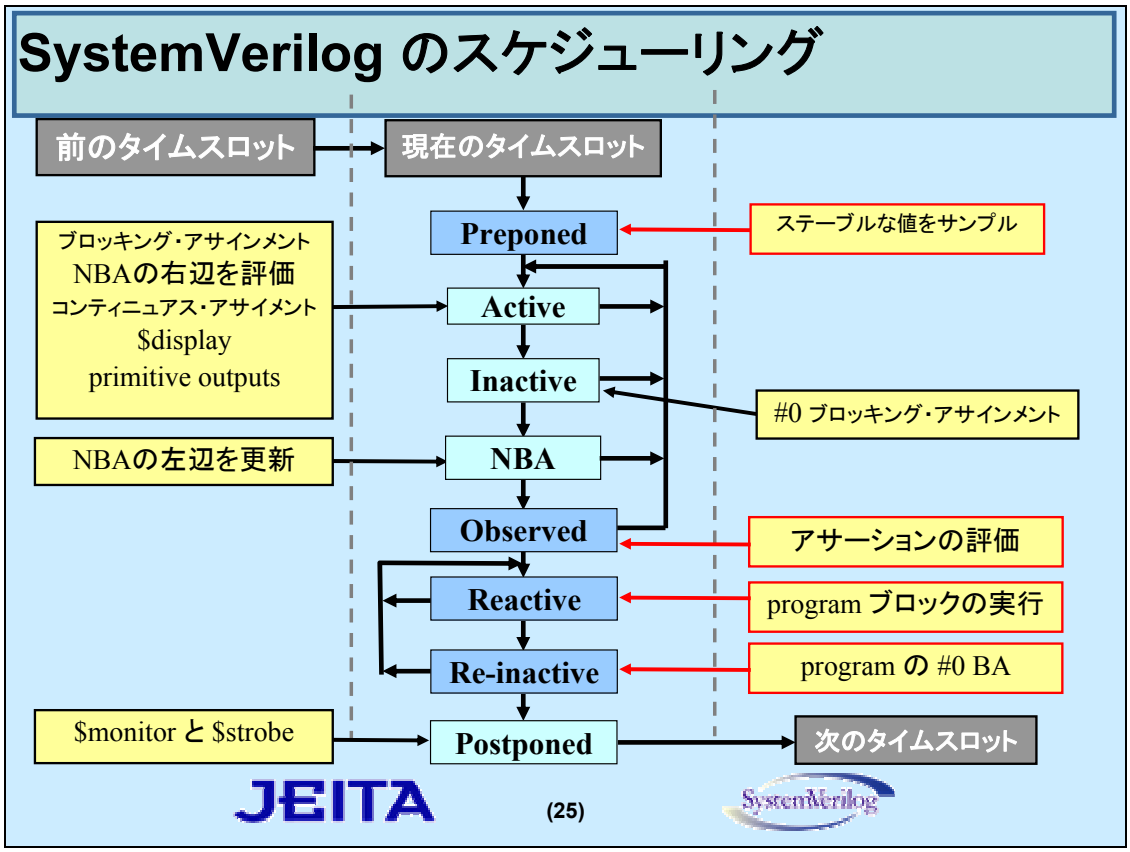
  → VHDLのprocess文と類似
- “*reactive*” 領域において実行される
  - DUTとの通信においてレーシングが起きないことを保証できる

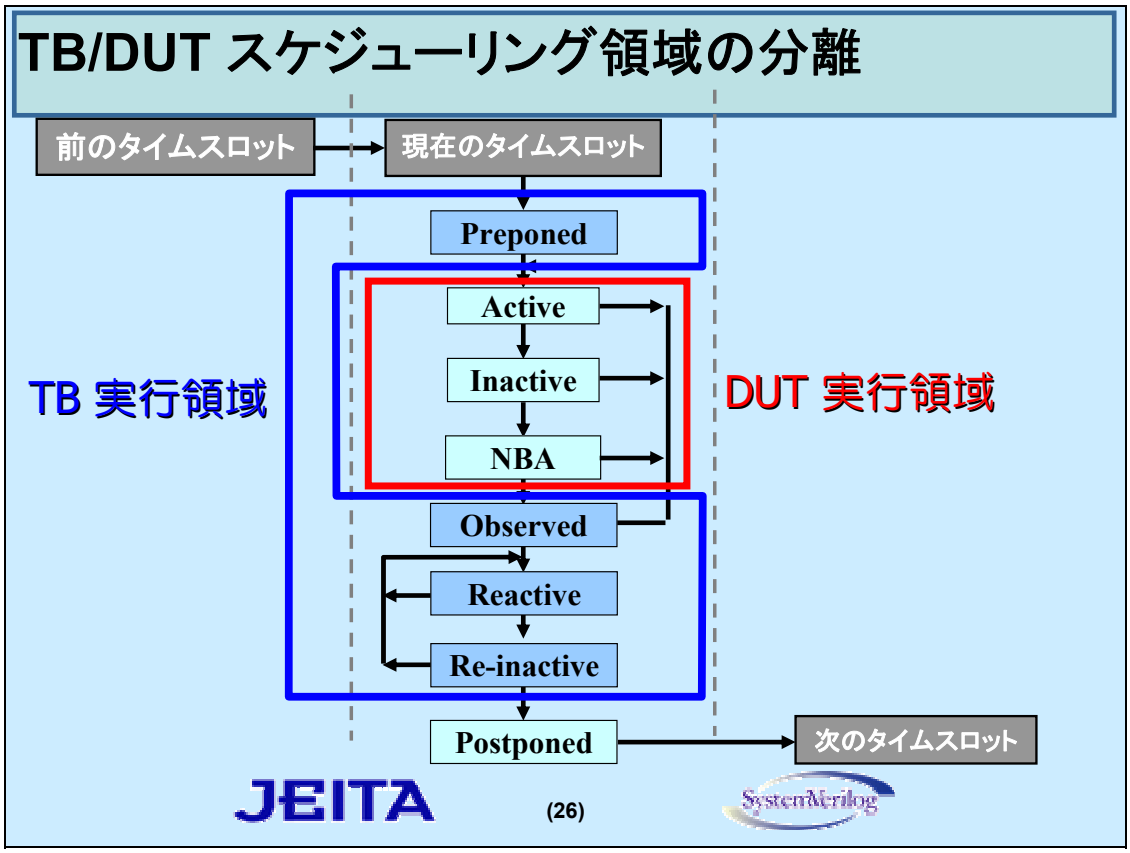
JEITA (23) SystemVerilog

Verilog 1995/2001 のスケジューリング



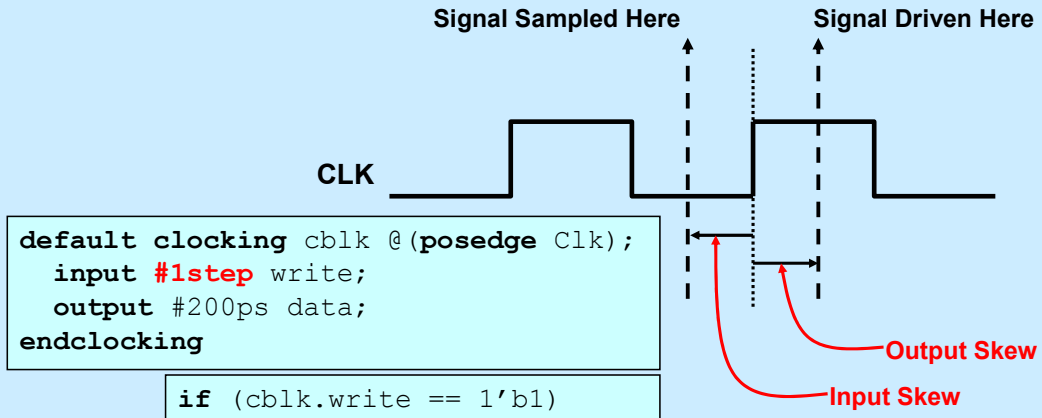
JEITA (24) SystemVerilog


SystemVerilog のスケジューリング
前のタイムスロット
現在のタイムスロット
Preponed
ステーブルな値をサンプル
ブロッキング・アサインメント
NBAの右辺を評価
コンティニュアス・アサイメント
$display
primitive outputs
Active
Inactive
#0 ブロッキング・アサインメント
NBAの左辺を更新
NBA
Observed
アサーションの評価
Reactive
program ブロックの実行
Re-inactive
program の #0 BA
$monitor と $strobe
Postponed
次のタイムスロット
JEITA
(25)
SystemVerilog


TB/DUT スケジューリング領域の分離
前のタイムスロット
現在のタイムスロット
Preponed
Active
Inactive
NBA
Observed
Reactive
Re-inactive
Postponed
次のタイムスロット
TB 実行領域
DUT 実行領域
JEITA
(26)
SystemVerilog

クロッキング・ブロック ― 入力のサンプリング

クロッキング・ブロックを使って安定した値をサンプルする




```
default clocking cblk @(posedge Clk);
  input #1step write;
  output #200ps data;
endclocking
```

```
if (cblk.write == 1'b1)
   cblk.data <= ##2 sample;
 ...
```

JEITA (27) SystemVerilog

クロッキング・ブロック ― 同期出力

- タイミング情報を一箇所で記述
  - パターンを生成する部分と分離
  - 後からタイミングを変更することが容易

```
program tb (output stb);
  clocking slck @(posedge clk);
    output #200ps stb;
  endclocking

  initial begin
      .....
      slck.stb = 1;
      .....
endprogram
```

- プログラム・ブロックのポート stb へのドライブ
  - slck.stb = 1 の代入の後の次のクロックの立ち上がりから、200ps後

JEITA (28) SystemVerilog

クロッキング・ブロック ― クロックへの同期

- サイクル・ディレイ記法: ##
  - default clocking
    - モジュール(あるいはprogram、interface) に1つだけ記述可能
  - default clocking で指定したクロックによるディレイ記述
    - ##5　　5サイクル（固定値）遅らせる
    - ##(j+1)　　変数で指定したサイクル遅らせる
- 同期イベント
  - @(negedge slck.bus[0]);
    bus の bit0 の立ち下りを待つ
    クロッキング・ブロック slck で指定したクロックでサンプルされる bus の値の bit0 の立ち下りを待つ

これまでのVerilog HDLテストベンチ

- デザインのようにIP化し、再利用して行くことが困難
  - 常に「似ている過去のテストベンチ」を「コピー」し、エディット
    - 生産性が低い
    - 完成済みテストベンチを壊す事となり、信頼性が低くなる
  - なぜならmodule, taskベースのテストベンチだから・・・
    - task - 入れ子が困難
    - module - input/output/inoutを通過する0/1/X/Zによる低レベルな情報伝播
  - Verilog HDLでは、デザインのように構造化しにくい

クラスとは？

- 動的オブジェクト
  - 必要な時にだけ使用する
  - moduleは静的オブジェクト – 最初から最後まで
- データ構造を容易に拡張できる
  - 制御信号の追加 – または隠蔽
- 関数(task/function)を容易に拡張できる
  - 関数の追加 – または隠蔽
  - 関数の引数追加 – または引数の隠蔽
- module, taskに比べて柔軟性が高い

SystemVerilogのクラス

基本文法:

```
class name;
    <data_declarations>;
    <task/function_declarations>;
endclass
```

Note:
クラスの宣言だけでは記憶域を取らない。
newでインスタンスを生成してはじめて記憶域を取る

```
class ab_base;
  rand reg [31:0] a, b;
  constraint c0 {
    a[3:0] == 'h0;
    b > a;
  }
  function void disp();
    $display(a,,b);
  endfunction
endclass

ab_base ab_obj = new;
```

クラスの継承

- クラスは他のクラスの特性やメソッドを継承できる
  - サブクラスは親クラスのメソッドを明示的に再定義できる
  - 良い機能はそのままに、カスタマイズが可能

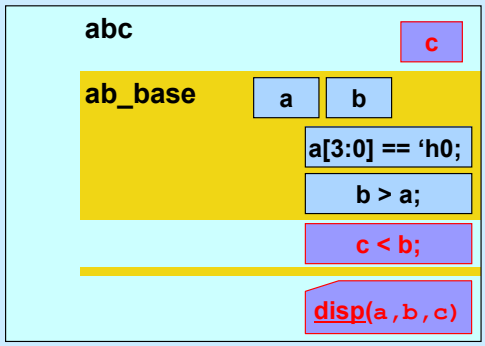
```
class abc extends ab_base;

  rand reg [31:0] c;

  constraint c1 {
    c < b;
  }

  function void disp();
    $display(a,,b,,c);
  endfunction

endclass
```



まとめ

- 最近の検証トレンド
  - カバレッジ・ドリブン検証
  - テストベンチ、検証環境の再利用
- テストベンチの説明
  - コンストレイント・ランダム・スティミュラス
  - カバレッジ
  - レスポンス
  - テストベンチ、検証環境の再利用
  - program & clocking block
  - クラス

EDAツールサポート状況

- 調査対象ツール/Version(2007/1/10時点で公開されているツール)
  - 論理シミュレータ
    - *Incisive Simulator 5.83 (Cadence)*
    - *Questa 6.2e (Mentor)*
    - *VCS 2006.06 (Synopsys)*
  - 論理合成ツール
    - *Encounter RTL Compiler 6.2 (Cadence)*
    - *Design Compiler 2006.06-SP4 (Synopsys)*
  - 等価性検証ツール
    - *Encounter Conformal 6.2 (Cadence)*
    - *Formality 2006.12 (Synopsys)*
  - プロパティチェッカー
    - *0-In V2.3q (Mentor)*
    - *Incisive Formal Verifier 5.7 (Cadence)*
    - *Magellan 2006.06 (Synopsys)*
  - Lintチェッカー
    - *Incisive Simulator 5.83 (Cadence)*
    - *LEDA2006.06 (Synopsys)*

JEITA (35) SystemVerilog

サポート状況の集計方法

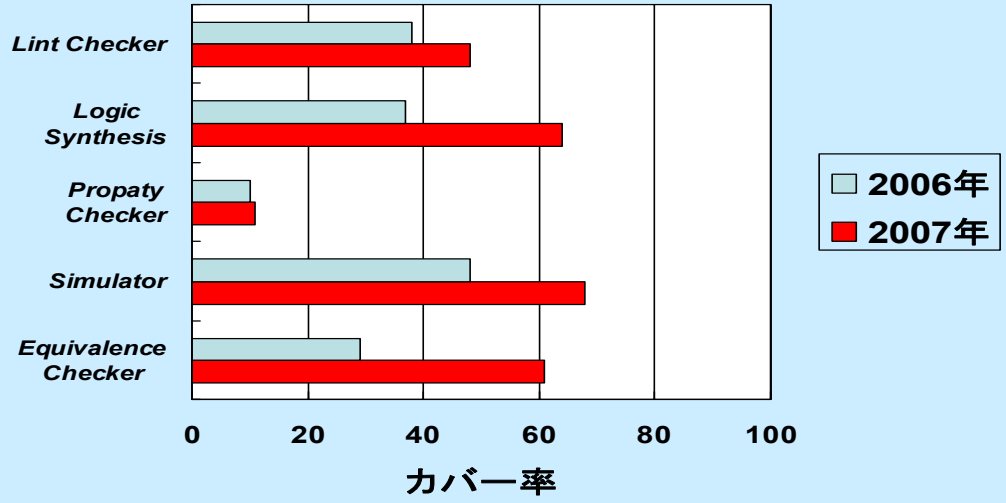
- IEEE Std. 1800-2005 LRMから項目を抜粋
- 各項目についてツールごとにサポート済み/未サポートを調査
- ツールを以下の5つのカテゴリに分類、全ツール結果の*AND*をとる
  - 論理シミュレータ
  - 合成ツール
  - プロパティチェッカー
  - 等価性検証ツール
  - Lintチェッカー
- 標準言語として*ベンダーを気にせず*使用することができる構文カバー率を調査

カバー率はベンダーによりばらつきがあります。詳細は各ベンダーにお問い合わせください。

JEITA (36) SystemVerilog

ツールカテゴリごとのSystemVerilog構文カバー率

- ツールカテゴリごとのLRM全項目に対するカバー率
  - Lintチェッカー、合成ツール及び等価性チェッカーにおいて合成非対象構文(アサーション,TB等)は対象外。



JEITA (37) SystemVerilog

今回紹介した構文のサポート状況

- 論理シミュレータ上ではほぼサポートされている。
  - Randomization / Constraints
  - randsequence
  - cover / covergroup
  - cross
  - Program Block
  - Clocking Block
  - Class
- 使用にあたっては、機能が制限されているものもあります。EDAベンダから情報収集してください。

JEITA (38) SystemVerilog

サポート状況まとめ

- 昨年に比べEDAツールの対応が進んでいることが確認できる。
- 合成/等価検証/シミュレータに関してはベンダーを気にせず使用することが可能な環境が、実用レベルに達しつつある。
- プロパティチェッカーについては、未だサポートが充分とは言えず、早期のサポートが期待される。

JEITA (39) SystemVerilog

SystemVerilog
(IEEE Std. 1800-2005)
チュートリアル

2006/11/29 @デザインガイア2006

JEITA SystemVerilogタスク・グループ

JEITA

SystemVerilog

アジェンダ

- JEITA SystemVerilogタスク･グループ紹介
- SystemVerilog概要
- 言語チュートリアル
  - 設計のための構文
  - 検証のための構文
- まとめ

SystemVerilog Task Groupメンバー

- JEITA：社団法人 電子情報技術産業協会
- 「JEITA EDA技術専門委員会／標準化小委員会傘下の SystemVerilog Task Group (SV-TG)
  - SystemVerilogの国際標準化活動に日本から参画
  - メンバー企業 10社

沖ネットワークエルエスアイ	日本シノプシス
三洋	松下電器産業
図研	メンター・グラフィックス・ジャパン
東芝	富士通
日本ケイデンス・デザイン・システムズ社	ルネサステクノロジ

(注)五十音順, 敬称略

JEITA (3) SystemVerilog

JEITA組織図

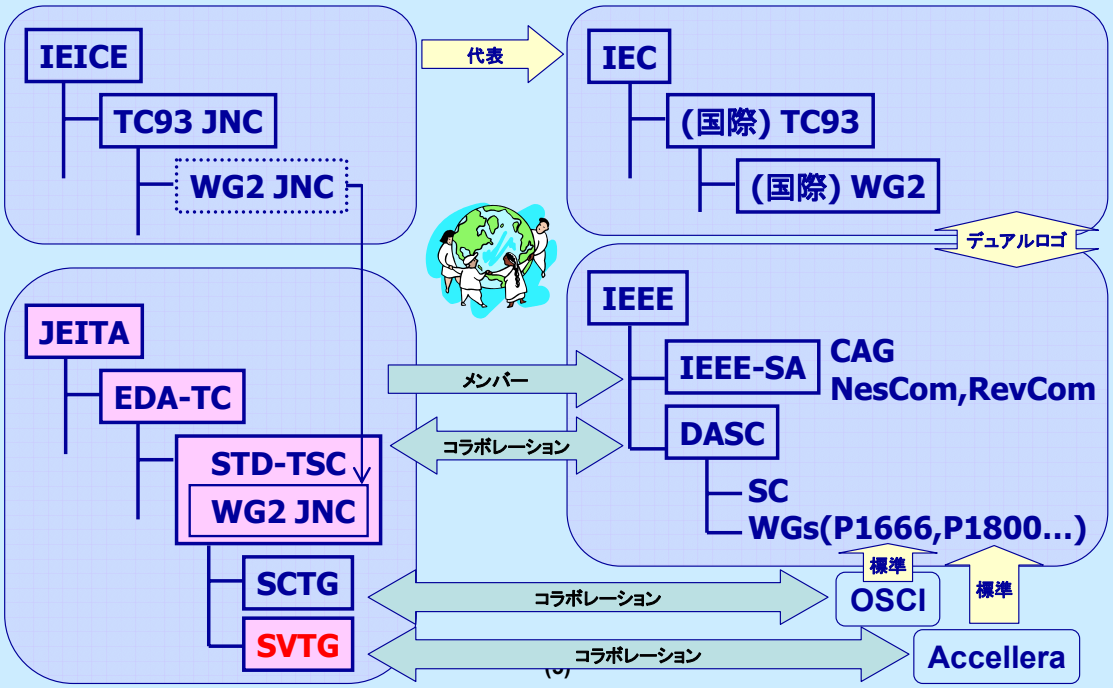
Japan Electronics and Information Technology Industries Association

- **JEITA**
  - Information Systems Board
  - Personal Informatization Board
  - Digital Home Appliances Board
  - Industrial Equipment and Social Systems Board
  - Display Devices Board
  - Electronic Components Board
  - **Semiconductor Board(JEITA-JSIA)**
    - Semiconductor Industrial Affairs Committee
    - Semiconductor International Affairs Committee
    - **Semiconductor Technology Committee**
      - Marketing Committee
      - Road Map Committee
      - ⋮
      - **EDA Technical Committee(EDA-TC)**

◆*EDA Technical Committee was formed to handle EDIF 2.0 standard as one of technical committees in JEITA (former EIAJ) in April 1990 .*

JEITA (4) SystemVerilog

世界の標準機関とコラボレーション



アジェンダ

- JEITA SystemVerilogタスク･グループ紹介
- SystemVerilog概要
- 言語チュートリアル
  - 設計のための構文
  - 検証のための構文
- まとめ

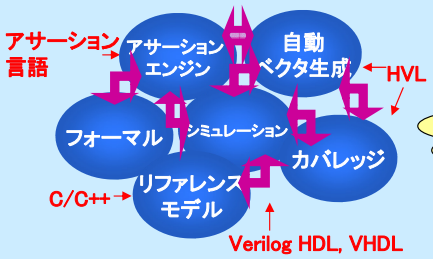
JEITA (6) SystemVerilog

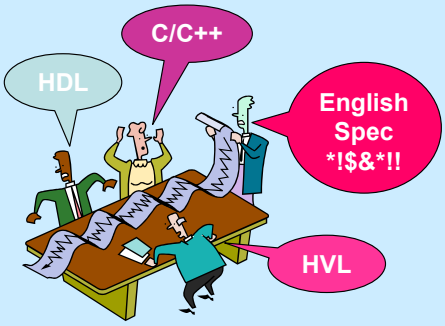
何故SystemVerilogなのか？

- より大きな規模の回路をより短期間で設計しなくてはならない
  - より簡潔な記述で設計する必要 -> 記述量の削減
  - 曖昧の無い言語が必要 -> 不要なイタレーションの削減
  - 新しい検証テクニックが必要 -> より多くのテストを効率良く
  - 高抽象度化のための言語構造が必要 -> TLM構造のサポート
- しかし…
  - Verilog HDL / VHDLでは問題が多い…
    - Verilog HDLには,信号競合という大問題がある
    - HDLのピンモデルでは抽象度は上げられない
  - 過去の資産(デザイン,エンジニアのスキル等)は捨てたくない
  - 等々
- 新しい言語ではなく,Verilog HDLをエンハンスすることで,資産を継承

JEITA (7) SystemVerilog

新しい技術の台頭と共に言語が乱立

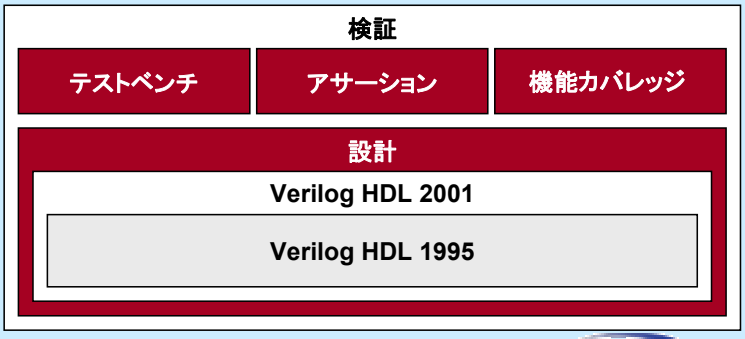




JEITA (8) SystemVerilog

SystemVerilog の概要

- IEEE Std. 1364-2001 (Verilog HDL)の拡張
- IEEE Std. 1800-2005 標準
- デザインと検証のために言語を統合
  - HDVL (Hardware Description and Verification Language)
- SystemVerilog はVerilog HDL の新バージョン!



設計記述面の利点

- Verilog HDL記述の曖昧さの解消
- 可読性, 保守容易性の向上,記述量の削減
  (コンサイスRTL)
  - コード記述量を1/3～1/10 以下へと削減可能
  - 簡潔な記述 = 生産性の向上,バグの減少
- TLM記述を可能とする
  - `interface`構造を追加
  - ピンレベル・インターフェイス以外の`module`接続をサポート
- システムから ゲート・レベルまで
  - ゲート・レベル・ネットリストはVerilog HDLをそのまま継承

検証記述面の利点

- 検証記述の改善
  - 単一環境に組み込まれた検証
    - アドバンスド･テストベンチ構文を追加
    - アサーション構文を追加
    - カバレッジ構文を追加
- DPI(Direct Programing Interface)によるC/C++との親和性強化

JEITA (11) SystemVerilog

他言語と比べるならば･･･

- vs 既存HDL
  - 検証面の充実
    - 制約付きランダム･スティミュラス
    - アサーション
    - 機能カバレッジ
  - 動的オブジェクトの追加
- vs ソフトウェア言語
  - 並列性
    - `always`, `fork-join`, ゲート･プリミティブ
  - 時間の概念
    - #(遅延), ##(サイクル遅延), @(イベント遅延)

JEITA (12) SystemVerilog

アジェンダ

- JEITA SystemVerilogタスク･グループ紹介
- SystemVerilog概要
- 言語チュートリアル
  - 設計のための構文
  - 検証のための構文
- まとめ

言語特徴説明

- 設計のための構文
  - データ型
  - インタフェース
  - デザインインテント手続きブロック
  - デザインインテント条件文
- 検証のための構文
  - アサーション

※ テストベンチ構文とカバレッジ構文は今回は,割愛

設計のための構文

JEITA (15) SystemVerilog

データ型 – 基本

- `logic` 4値 (0/1/X/Z)
  - `reg`, `wire`の区別なく記述可能
  - `wire`として使用する場合，ドライバを一つに限定
  - レース状態が起きないモデリングを可能に
  - リント・ツールでチェック可能（早期検出）
- `bit` 2値 (0/1)
  - シミュレーションの高速化
  - 初期値は0と規定されている
  - 全てのシミュレータの2値モードが整合

JEITA (16) SystemVerilog

データ型 – ユーザ定義

- typedef
  - 可読性の向上
    - ユーザにとって意味のある型を定義できる
  - 保守容易性の向上
    - 記述部を変更せず，宣言部を変更するだけで定義を変更できる
- typedefを用いたワード定義の例

```
// 1ワードを32ビットとして定義
typedef logic [31:0] word_t;
word_t A, B;
```

```
// 1ワードを64ビットとして定義
typedef logic [63:0] word_t;
word_t A, B;
```

JEITA (17) SystemVerilog

データ型 – 構造体

- struct
  - C言語に似た，簡潔な宣言
    - typedef を用いて，データ型名を指定可能
    - 構造体の全体に対し，一括して値を代入可能
  - 可読性，保守容易性の向上，記述量の削減

```
struct {
  bit [7:0]   opcode; // 8bit幅の2値
  bit [23:0]  addr;   // 24bit幅の2値
  data_word   data;   // typedefされたユーザデータ型
} instruction;        // 名前付き構造体の宣言

instruction IR;       // 構造体の配置

IR = {6, 377, 45};    // 構造体への代入
```

JEITA (18) SystemVerilog

データ型 – 列挙型

- enum
  - Verilog HDL 1995/2001
    - parameter定数宣言，もしくは`defineマクロ宣言
    - 値に名前をつけるにとどまる
  - SystemVerilog
    - enum宣言
    - 値の集合を定義，初期値設定，範囲チェック可能
    - 高抽象化モデリング
    - 記述量の削減に加え，可読性，保守容易性の向上

```
parameter RED = 0, YELLOW = 1, GREEN = 2;
reg [1:0] traffic_light;
```

Verilog HDL 1995/2001

```
enum {red, yellow, green} traffic_light;
```

SystemVerilog

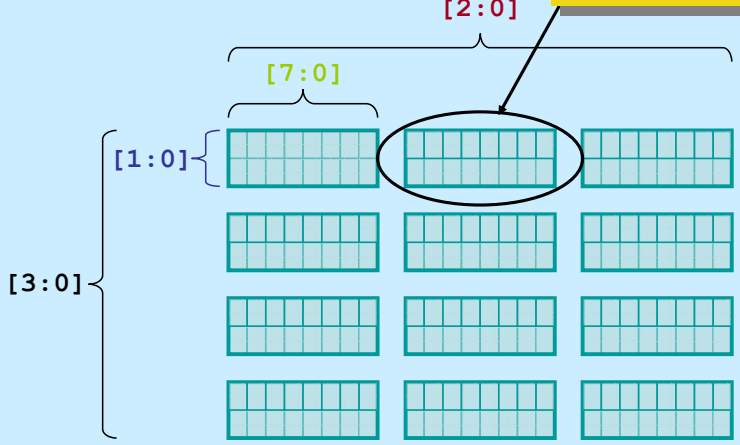
JEITA (19) SystemVerilog

データ型 – 多次元配列

- 2次元パック型 x 2次元アンパック型

```
logic [1:0][7:0] arrayxd [3:0][2:0];
```

この宣言で一度に
アクセスできる最大単位は16bits



JEITA (20) SystemVerilog

データ型 – 多次元配列

- 可読性,保守容易性,記述量の削減

```
reg     [7:0]       mem0[0:1023];
reg     [7:0]       mem1[0:1023];
reg     [7:0]       mem2[0:1023];
reg     [7:0]       mem3[0:1023];
reg     [31:0]      q;

always @(posedge clk) begin
    if(ce) begin
        if(we[3]) mem3[a] <= d[31:24];
        if(we[2]) mem2[a] <= d[23:16];
        if(we[1]) mem1[a] <= d[15:8];
        if(we[0]) mem0[a] <= d[7:0];
    end
end

always @(posedge clk) begin
    if(ce)
        q <= {mem3[a], mem2[a],
              mem1[a], mem0[a]};
end
```

Verilog HDL 1995/2001

```
reg     [3:0][7:0]      mem[0:1023];
reg     [3:0][7:0]      q;

always @(posedge clk) begin
    if(ce) begin
        if(we[3]) mem[a][3] <= d[3];
        if(we[2]) mem[a][2] <= d[2];
        if(we[1]) mem[a][1] <= d[1];
        if(we[0]) mem[a][0] <= d[0];
    end
end

always @(posedge clk) begi
    if(ce)
        q <= mem[a];
end
```

多次元化

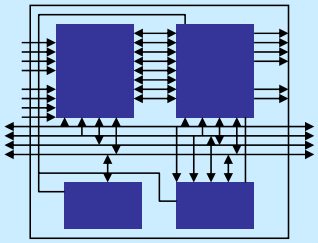
同一変数を
バイト単位で
ハンドリング

SystemVerilog

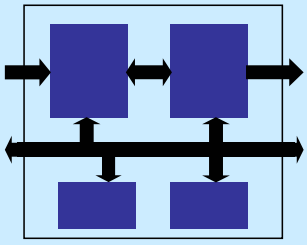
(21)

インタフェース

- Verilog HDL 1995/2001 モジュール・ポート接続
  - モジュール間の詳細な端子接続
    を知る必要がある
  - 入出力の端子群に変更があった場合
    記述の変更が煩雑
  - ポート宣言を多くのモジュールで
    何度も繰り返す必要がある



- SystemVerilog `interface`接続が可能
  - モジュール間の接続を一つに束ねて行う
  - 接続の内容の定義はモジュール定義からは
    独立して行う
  - 記述量の削減に多大な効果
  - ポートの追加・削除に伴うファイル修正作業を軽減



JEITA

(22)

SystemVerilog

インタフェース

```
module memMod(
 input  logic       req,
        bit         clk,
        bit         start,
        logic [1:0] mode,
        logic [7:0] addr,
 inout  wire  [7:0] data,
 output bit         gnt,
        bit         rdy );
        logic       avail;
 always @(posedge clk) gnt <= req & avail;
...
endmodule

module cpuMod(
 input  bit        clk,
        bit        gnt,
        bit        rdy,
 inout  wire [7:0] data,
 output bit        req,
        bit        start,
       logic [7:0] addr,
       logic [1:0] mode );
 ...
endmodule

module top;
 logic req, gnt, start, rdy;
 logic clk = 0;
 logic [1:0] mode;
 logic [7:0] addr;
 wire  [7:0] data;
  memMod mem(req, clk, start, mode, addr, data, gnt, rdy);
  cpuMod cpu(clk, gnt, rdy, data, req, start, addr, mode);
endmodule
```

interfaceを使わない場合

```
interface simple_bus;
 logic       req, gnt;
 logic [7:0] addr, data;
 logic [1:0] mode;
 logic       start, rdy;
endinterface

module memMod(simple_bus a, input bit clk);
 logic       avail;
 always @(posedge clk) a.gnt <= a.req & avail;
  ...
endmodule

module cpuMod(simple_bus b, input bit clk);
 ...
endmodule

module top;
 logic clk = 0;
 simple_bus sb_intf();
 memMod mem(sb_intf, clk);
 cpuMod cpu(.b(sb_intf), .clk(clk));
endmodule
```

interfaceを使う場合

モジュール接続

- Verilog HDL 1995/2001 での接続:
  - 順序接続
  - 名前接続

```
module my_chip (input wire clock, reset)
  wire [63:0] a, b, c, d;
  alu u1 (clock, a, b, reset, c, d);
  alu u2 (.ck(clock), .reset(reset), .a(a), .b(b), .c(c), .d(d));
```

ポート宣言の順序で接続
(誤りやすい記述)

名前同士で接続
(冗長な記述)

- SystemVerilog で拡張された名前接続:
  - .name を用いて同一端子名省略
  - .* ワイルドカード

```
module my_chip (input wire clock, r
  wire [63:0] a, b, c, d;
  alu u1 (.ck(clock), .reset, .a, .b, .c, .d);
  alu u2 (.ck(clock), .*);
```

.name によって接続される端子とネット名が同一のときに記述を削減できる

.* は同一名の端子とネットを自動的に接続する

JEITA

(24)

デザインインテント手続きブロック

- always_comb/latch/ff
  - 設計者が意図する回路を明示的に指定
  - ツール認識に依存しない回路が得られる
  - always_comb/latchはセンシティビティリストが不要
    - ツールは代入の右辺から導き出す
  - 詳細な規定によりツール間の不整合を防ぐ
    - initial, always すべての手続き的ブロックがアクティブになった後,時間ゼロにおいて自動的に1回実行される
    - ⇒ 時間ゼロでの出力値は入力値に一致する
    - Verilog HDL 1995/2001のalways記述で@イベントがトリガされず、シミュレーションがロックする問題を解決

```
always_comb begin
  tmp1 = a & b;
  tmp2 = c & d;
  y = tmp1 ! tmp2;
end
```

JEITA (25)

デザインインテント条件文

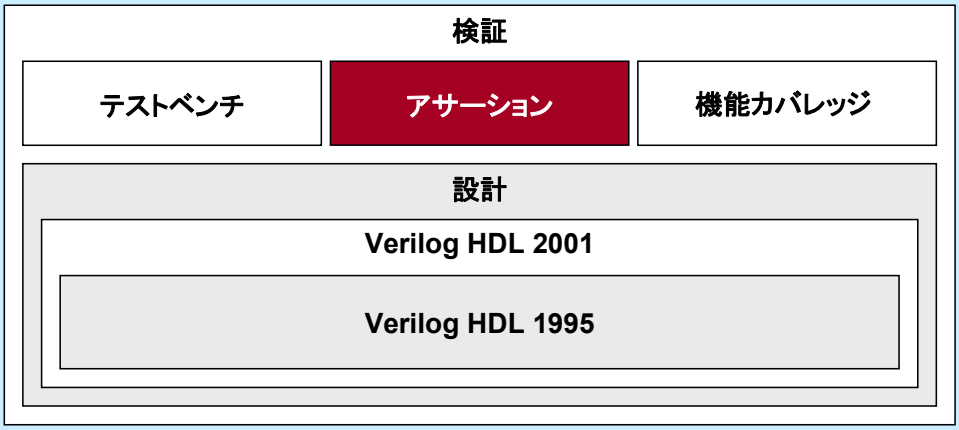
- unique/priority case/if
  - case/if文にて条件処理を順次/並列の何れかの解釈に指定
  - ツールベンダー依存のプラグマを排除
    - // full_case, parallel_case
  - シミュレーション,合成で統一した解釈を実現
    - RTL/ゲート間のシミュレーション・ミスマッチ防止

```
unique case (sel)
  3'b001 : muxo = a;
  3'b010 : muxo = b;
  3'b100 : muxo = c;
endcase
```

```
priority if (sel0) q = 3'b000;
    else if (sel1) q = 3'b011;
    else if (sel2) q = 3'b110;
    else if (sel3) q = 3'b111;
```

JEITA (26) SystemVerilog

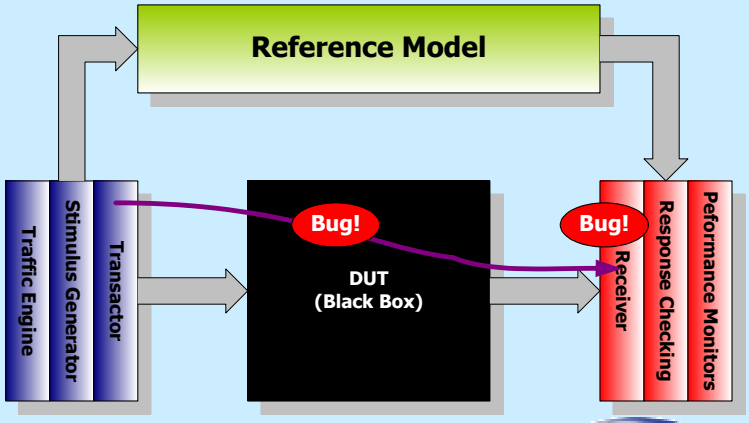
検証のための構文



JEITA (27) SystemVerilog

従来の検証手法

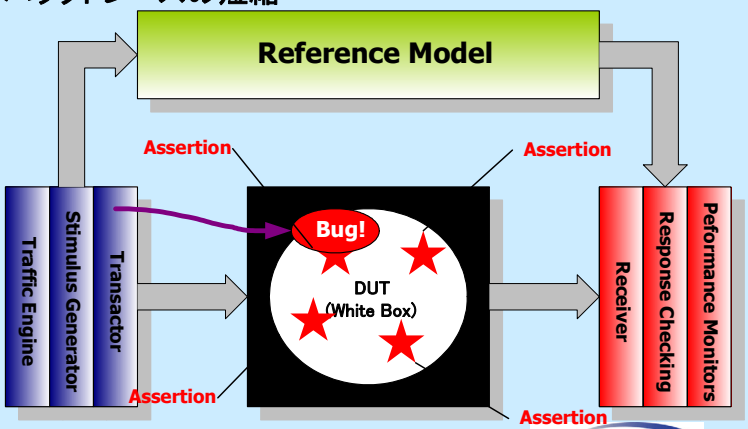
- バグの発見のためには．．．
  1. スティミュラスを加えて不正な動作を活性化
  2. 同じく不正な動作の結果を出力へ伝播させる
  3. 出力波形からバックトレースしてバグの原因箇所を特定



JEITA (28) SystemVerilog

アサーションベース検証

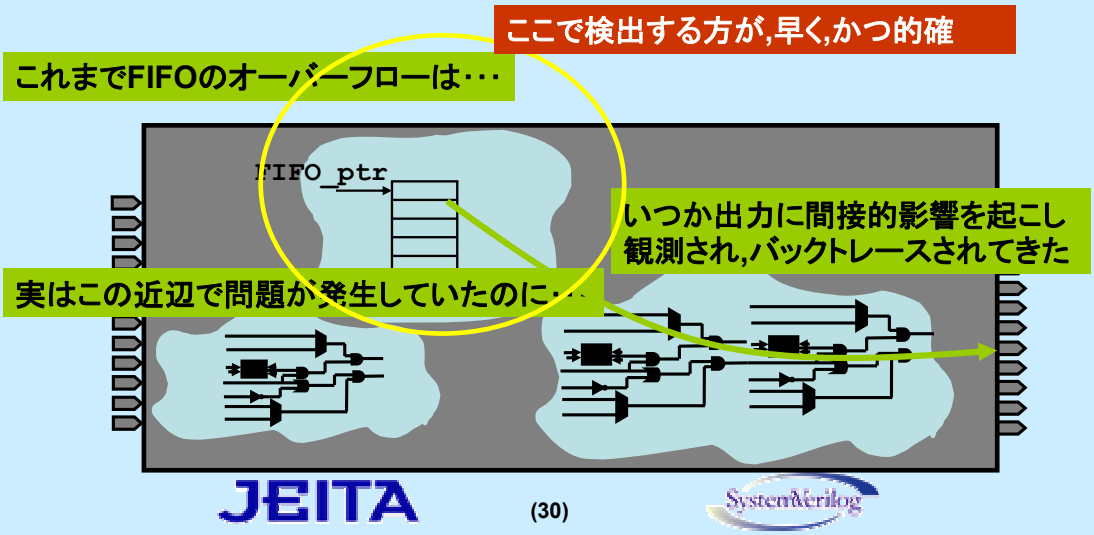
- 観測性の向上
  - バグの早期発見
  - 出力に伝播させることが困難なバグの発見
  - バックトレースの短縮



(29)

アサーションでバグをすぐに見つける

- ブロック内ホワイトボックス検証
  - 設計のバグを,プライマリ出力に到達する前に観測



(30)

アサーションとは？ 何故使用するのか？

- アサーションは検証要素
  - 設計ブロックや, そのインタフェース上の禁止される動作を監視
  - アサーションで記述される, 期待される動作を追跡
- アサーションの利点
  - 観測性の向上
  - デバッグ効率向上
  - 検証期間短縮
  - カバレッジの一指標

JEITA (31) SystemVerilog

アサーション使用例

- 禁止状態のチェック
  - プロパティの定義
  - プロパティの引数による再利用
  - `not` 演算子
- ワンホットのチェック
  - `$onehot` システム・ファンクション
- タイムアウト
  - シーケンスの定義
  - シーケンスの引数による再利用
  - プロパティからのシーケンス呼び出し
- 状態保持のチェック
  - `throughout` 演算子

JEITA (32) SystemVerilog

禁止状態のチェック

- 意図していない状態になっていないか？
  - FIFOにおいて*push*と*pop*は同時に発生しない
  - FIFOにおいて*full*のときに書き込みはできない

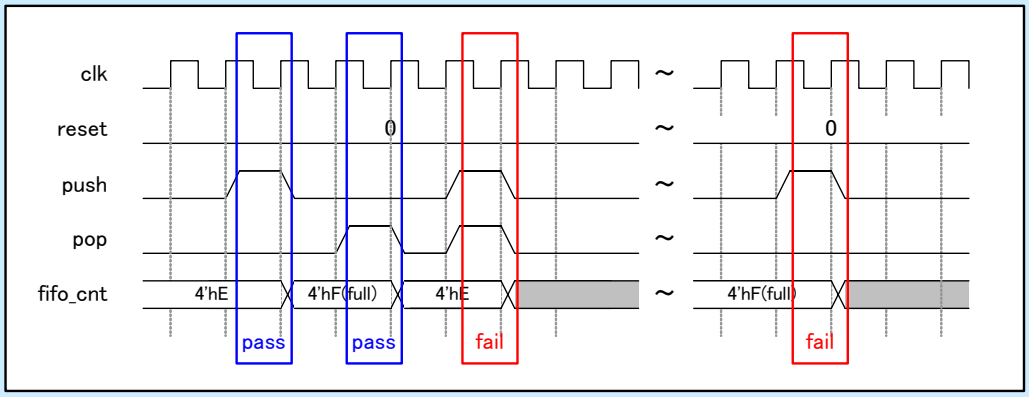
  - property を使用して期待する状態を定義
  - not を使用して禁止状態を指定

```
property p_expected([引数]);
        <期待する状態>
endproperty
property p_illegal([引数]);
        not (<禁止状態>);
endproperty
```

JEITA (33) SystemVerilog

禁止状態の使用例

- FIFOにおいて*push*と*pop*は同時に発生しない
- FIFOにおいて*full*のときに書き込みはできない



JEITA (34) SystemVerilog

禁止状態の記述例

- FIFOにおいて*push*と*pop*は同時に発生しない（禁止状態1）
- FIFOにおいて*full*のときに書き込みはできない（禁止状態2）

```
property p_fifo_push_pop;  // 禁止状態1を定義
       @(posedge clk) disable iff (reset)
                              // サンプルクロックと非同期リセット
       not (push && pop);
endproperty
property p_fifo_full_push; // 禁止状態2を定義
       @(posedge clk) disable iff (reset)
       not (full && push && !pop);
endproperty

// 定義したプロパティをアサート
a_fifo_rule_1 : assert property (p_fifo_push_pop);
a_fifo_rule_2 : assert property (p_fifo_full_push);
```


ワンホットのチェック

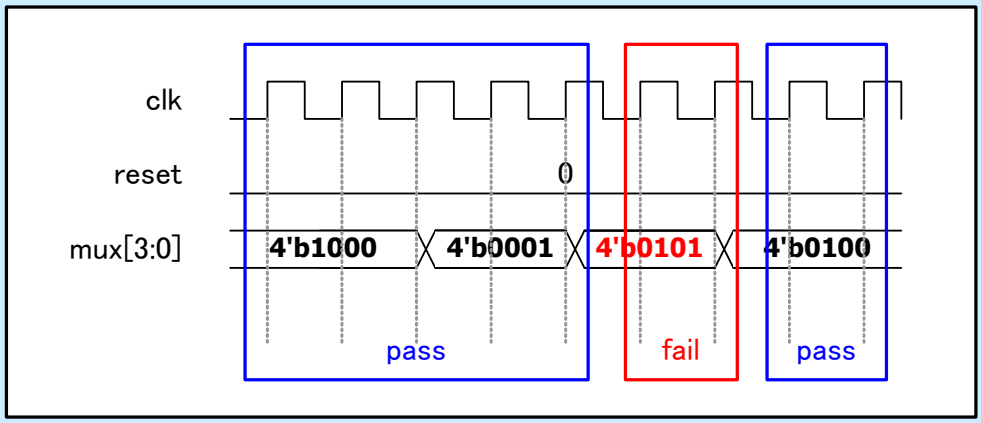
- ワンホットであるべき信号が,不正状態にならないか？
  - 対象とする信号の1ビットのみ”1”である

- $onehotシステム・ファンクションを使用

```
$onehot （<チェック対象信号>）
```

ワンホットの使用例

- 信号中の1ビットのみ1である



ワンホットの記述例

- 信号中の1ビットのみ1である
  - プロパティ記述をassert

```
property p_onehot_check; // ワンホット指定を定義
        @(posedge clk) disable iff (reset)
        $onehot (mux);
endproperty
// 定義したプロパティをアサート
a_onehot_1 : assert property (p_onehot_check);
```

  - assert宣言内にプロパティを記述

```
// もしくはプロパティ定義を省略
a_onehot_2 : assert property (
        @(posedge clk) disable iff (reset)
        $onehot (mux));
```

タイムアウトのチェック

- 規定時間以内に動作が完了するか？
  - *req*のアサート後, 既定サイクル以内に*ack*がアサート
  - *bus_trans*のデアサート後, 次のサイクルで*bus_req*デアサート

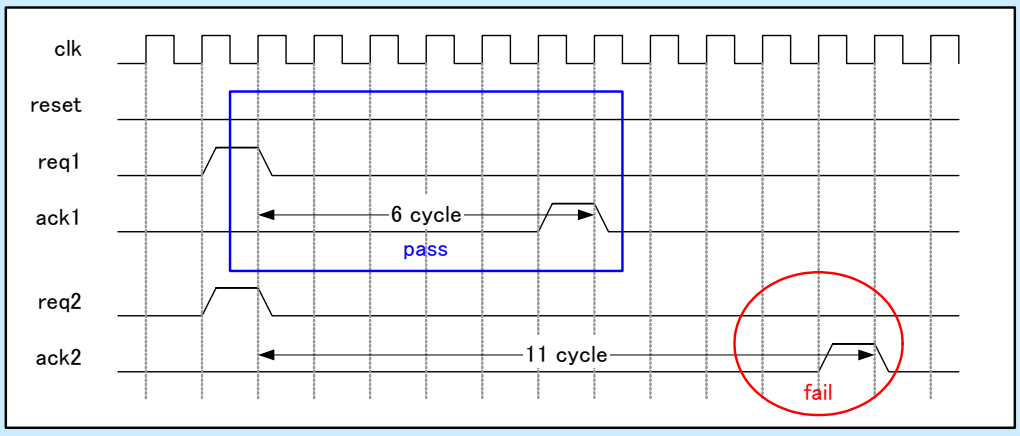
- 規定の動作をsequenceで定義

```
sequence s_sequence([引数]);
        <動作シーケンス>
endsequence
```


タイムアウトの使用例

- *req*のアサート後, 既定サイクル以内に*ack*がアサート

※ *req1-ack1*: 8サイクル以内, *req2-ack2*: 10サイクル以内



タイムアウトの記述例

- *req*のアサート後, 既定サイクル以内に*ack*がアサート

※ *req1-ack1*: 8サイクル以内, *req2-ack2*: 10サイクル以内

```
sequence s_response(ack, maxcycle); // req-ackのタイムアウトを定義
       @(posedge clk) ##[0:maxcycle-1] ack;
                                      // 0~maxcycleの間にackがアサート
endsequence
property p_response(req, ack, maxcycle); // req-ackのプロパティ
       @(posedge clk) disable iff (reset)// シーケンスをreq,ackに適用
       req |=> s_response (ack, maxcycle);
endproperty

// 定義したプロパティをreq1-ack1,req2-ack2それぞれにアサート
// (既定サイクルは異なる)
a_timeout_rule_1 : assert property (p_response(req1, ack1, 8));
a_timeout_rule_2 : assert property (p_response(req2, ack2, 10));
```

sequence,property 再利用

状態保持のチェック

- 特定シーケンス中, 状態が保持されているか？
  - *frame*はバスサイクル期間中アクティブである
  - *ready*が8クロックlowである期間中, *transmit*はアサートされる

- throughoutを使用

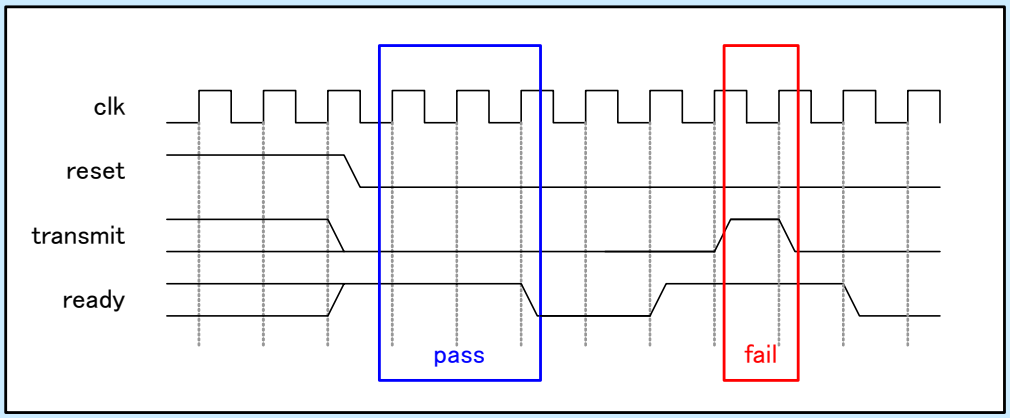
```
sequence s_throughout;
        <保持状態> throughout <保持期間>
endsequence
```

※保持状態はブーリアン表現

状態保持のチェック使用例

- *resetとtransmit*は, readyシーケンス中インアクティブである

  ※ readyシーケンス::= readyが連続的に3サイクルアクティブ



状態保持のチェック記述例

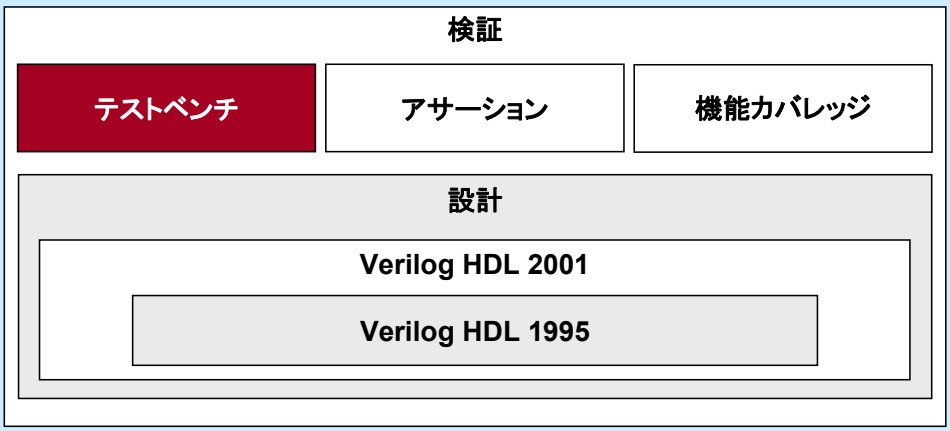
- *resetとtransmit*は, readyシーケンス中インアクティブである

  ※ readyシーケンス::= readyが連続的に3サイクルアクティブ

```
sequence s_ready_3; // 保持すべきシーケンスを定義
       @(posedge clk)
       ( !reset && !transmit ) throughout ready[*3];
endsequence

// シーケンスをclkドメインのプロパティとしてアサート
a_throughout_rule : assert property(
       @(posedge clk) disable iff (reset)
       s_ready_3);
```

テストベンチ・チュートリアルはSV-UF2007で！



JEITA (45)

SystemVerilog ユーザ・フォーラム

「システム・デザイン・フォーラム 2007」

ご挨拶

セッション 3:　SystemVerilog ユーザ・フォーラム 2007
26 日(13:30～15:30)

Verilog HDL (IEEE Std. 1364) の次世代言語として、2005 年 11 月に標準化完了した SystemVerilog (IEEE Std. 1800-2005) は、LSI 設計者や検証エンジニアの間で急速に適用が広がっています。
本セッションでは、1) Accellera による次の SystemVerilog 改定に向けた取り組みの紹介、2) JEITA SystemVerilog タスクグループによる SystemVerilog テストベンチ・チュートリアルと技術動向紹介、3) 日本の SystemVerilog ユーザによる、SystemVerilog 検証事例発表、を行います。

司会 ：浜口 加寿美 氏 (JEITA SystemVerilog タスクグループ 主査/松下電器産業)
① SystemVerilog 標準化アップデート ：Dennis Brophy 氏 (Accellera)
② SystemVerilog テストベンチ言語チュートリアル ：SystemVerilog タスクグループ
③ 検証言語としての SystemVerilog 適用事例 ：鎌田 丈良夫 氏 (ルネサステクノロジ)
④ SystemVerilog で構築したアレイプロセッサ検証環境とその効果 ：清水 圭典 氏 (ソナック)

日　時：1 月 25 日(木) 13:30-17:30　セッション 1　フィジカル・デザイン・フォーラム
1 月 26 日(金) 10:00-12:00　セッション 2　SystemC ユーザ・フォーラム 2007
13:30-15:30　セッション 3　SystemVerilog ユーザ・フォーラム 2007
場　所：パシフィコ横浜 アネックスホール

まとめ

- 設計・検証言語としてSystemVerilogがIEEEで標準化
- SystemVerilogの特徴から
  - 設計のための構文紹介
  - アサーションを構文紹介
- テストベンチと機能カバレッジは,2007年1月26日の「SystemVerilogユーザ・フォーラム」(EDSフェア併設システム・デザイン・フォーラム内)にて紹介予定

JEITA (47) SystemVerilog

予約語一覧(221)

always	for	output	supply0
and	force	parameter	supply1
assign	forever	pmos	table
begin	fork	posedge	task
buf	function	primitive	time
bufif0	highz0	pull0	tran
bufif1	highz1	pull1	tranif0
case	if	pulldown	tranif1
casex	ifnone	pullup	tri
casez	initial	rcmos	tri0
cmos	inout	real	tri1
deassign	input	realtime	triand
default	integer	reg	trior
defparam	join	release	trireg
disable	large	repeat	vectored
edge	macromodule	rnmos	wait
else	medium	rpmos	wand
end	module	rtran	weak0
endcase	nand	rtranif0	weak1
endfunction	negedge	rtranif1	while
endmodule	nmos	scalared	wire
endprimitive	nor	small	wor
endspecify	not	specify	xnor
endtable	notif0	specparam	xor
endtask	notif1	strong0	
event	or	strong1	

Verilog HDL 1995 (102)

automatic
cell
config
design
endconfig
endgenerate
generate
genvar
incdir
include
instance
liblist
library
localparam
noshowcancelled
pulsestyle_ondetect
pulsestyle_onevent
showcancelled
signed
unsigned
use

Verilog HDL 2001 (21)

alias	endclass	join_none	string
always_comb	endclocking	local	struct
always_ff	endgroup	logic	super
always_latch	endinterface	longint	tagged
assert	endpackage	matches	this
assume	endprogram	modport	throughout
before	endproperty	new	timeprecision
bind	endsequence	null	timeunit
bins	enum	package	type
binsof	expect	packed	typedef
bit	export	priority	union
break	extends	program	unique
byte	extern	property	uwire
chandle	final	protected	var
class	first_match	pure	virtual
clocking	foreach	rand	void
const	forkjoin	randc	wait_order
constraint	iff	randcase	wildcard
context	ignore_bins	randsequence	with
continue	illegal_bins	ref	within
cover	import	return	
covergroup	inside	sequence	
coverpoint	int	shortint	
cross	interface	shortreal	
dist	intersect	solve	
do	join_any	static	

SystemVerilog (98)

(48)

DVCon出張報告

（design & verification conference & exhibition）

2007年 03年09日 @SystemVerilog-TG
2007年 03年16日 @EDA-TC 、一部追加

松下電器産業株式会社
竹田津 弘州

Panasonic

出張の概要

DVCon（design & verification conference & exhibition）

■日程　2月21日（水）～2月23日（金）

■場所　アメリカ カリフォルニア州 サンノゼ
(会場) DoubleTree Hotel San Jose

■目的

- DVConは、SystemVerilogを含む上流の設計・検証にフォーカスした学会
- この分野においてはDACより豊富かつ最先端の情報が得られる
- SystemVerilogに関する標準化状況、業界動向や設計事例などの情報収集を目的とする

Panasonic

プログラム一覧

■SystemVerilogをキーワードにセッションを聴講
■ [ ] のセッションについて本資料にて報告する

21日		○AM 9:00～12:30	○PM 13:30～17:00
	Tutrial 1	Practical Deployment of Assertion-Based Verification and Formal Analysis	Tutrial 3 Pragmatic Adoption of Verification Methodology Manual(VMM) for Re-usable Transaction-Based Test benches in Systemverilog
	Tutrial 2	Practical Application of Mentor's Adbanced Verification Methodology(AVM)	Tutrial 4 Using Formal Verification to Attain Competeness and Correctness
	Tutrial 5		SystemC Transaction Level Modeling Standards and Methodology Guidelines

22日	900 -1000	Keynote Address	
	1030 -1200	session 1 SystemVerilog for Design	session 2 SystemC in Action
		Lunch Panel The Lowdown on Low-Power Design and Verification – Views from the Experts	
	1330 -1500	session 3 SystemVerilog Assertions	session 4 Physical Layer Verification
	1530 -1700	Panel session	

23日	900 -1000	Panel session		
	1030 -1200	session 5 SystemVerilog DPI	session 6 Formal Verification	session 7 Advances in Reserch –I
		Lunch Presentation		
	1330 -1500	session 8 Advanced Stimulus Generation	session 9 Real World Verification Applications	session 10 Advances in Reserch II
	1530 -1700	Embedded Tutorial		

Panasonic

参加プログラムと内容

Tutrial

●Practical Deployment of Assertion-Based Verification and Formal Analysis

Organizer: **Michal Siwinski**- *Cadence Design Systems, Inc.*
Presenter: **Lovleen Bhatia** - *Texas Instruments Inc.*
Erich Marschner - *Cadence Design Systems, Inc.*
Faisal Haque - *Verification Central*
Raghavan Menon - *Ingot Systems, Inc.*
Jon Michelson - *Verification Central*
Axel Scherer - *Cadence Design Systems, Inc.*

＜内容＞

●アサーションベース検証による検証サイクル短縮・設計品質向上の事例紹介
●基本から応用まで、効果を出すにはどう適用するかを具体的な回路を用いて事例発表

＜ポイント＞

●FIFO、アービタ、インターフェースなど、汎用的な回路での適用ポイント説明
●検証プランを早期に行い、検証を意識したデザイン設計が重要

- Formal検証ではアルゴリズム的に検証の限界がある
  ⇒ Formal適用を考えたブロック分割が必要
- Formal検証容易化には、インターフェースのプロトコルを複雑にしない
- マネジメントを容易化するブロック分割を検討
- 適用効果例：デッドコードチェックの容易化、複雑な条件にて起こりうるバグを発見

Panasonic

参加プログラムと内容

Session

●SystemVerilog for Design

Chair: Cliff Cummings - *Sunburst Design Inc.*
1.1 Stefan Sandström - *Axis Communications AB*
1.2 Bill Dittenhofer - *AMI Semiconductor*
1.3 Jonathan Bromley - *Doulos Ltd.*

＜内容＞

- **SystemVerilogを用いた設計効率向上事例紹介**
- **言語特長を活かした実用度の高い応用事例が多数あり、SystemVerilogの普及拡大が見られた**

＜ポイント＞

1.1 Designing a System-on-Chip with SystemVerilog In the Real World
- interfaceをはじめ、基本構文の使い方紹介
- SV導入にあたり、導入初期の工数増加を取り戻すだけの効果が得られる

1.2 Through the Looking Glass - A User's Perspective on SystemVerilog, SystemC
- SystemVerilogとSystemC、C++の相関に関しての発表
- SystemVerilogはリファレンスモデル生成、RTL検証に向いており、前者はSystemCも重なるが、後者はSystemCは向いていない
- モデル生成、検証（ランダム含む）双方を組み合わせて使うことで効果up

1.3 Towards a Practical Design Methodology with SystemVerilog Interfaces
- interfaceとmodportsの使用法を具体例とともに発表（構文特化のチュートリアル）
- 応用事例がもっとも多く見られた発表

Panasonic

参加プログラムと内容

Session

●SystemVerilog Assertions

Session Chair: David Lacey - *Hewlett-Packard Co.*
3.1 Dmitry Korchemny - *Intel Corp.*
Eduard Cerny, Angshuman Saha - *Synopsys, Inc.*
3.2 Jiang Long, Andrew Seawright,
Harry Foster - *Mentor Graphics Corp.*
3.3 Roger Sabbagh - *Mentor Graphics Corp.*
Jim O'Connor - *iVivity, Inc.*

＜内容＞

- **SystemVerilogによるアサーションを用いた機能検証事例紹介**
- **大手メーカによる適用事例があり、応用的な利用技術・適用ノウハウが紹介された**

＜ポイント＞

3.1 Using SystemVerilog Assertions for Creating Property-Based Checkers
- 検証環境の再利用性を高めるチェッカライブラリ作成例を紹介
- クロック設定を中心に、define/generete文の応用により、回路にあわせて編集可能なチェッカライブラリ作成が可能

3.2 SVA Local Variable Coding Guidelines for Efficient Use
- ローカル変数を使ったアサーションコーディングガイド
- ローカル変数により、複雑な信号間比較も容易に実行可能、質の向上が可能
- ただし、多用するとシミュレータ等のツールパフォーマンスに影響

3.3 Have I Placed All the Right Assertions
- I2Cを例に、アサーション作成を意識したシグナルフローチャート作成法を紹介

Panasonic

参加プログラムと内容

Session

●**SystemVerilog DPI**

Session Chair: Stuart Sutherland - *Sutherland HDL, Inc.*
5.1 Rich Edelman, Mark Glasser,
Arnab Saha, Hui Yin - *Mentor Graphics Corp.*
5.2 Tom Austin – *Plexus*
5.3 Arthur Freitas - *Hyperstone AG*

＜内容＞

- **SystemVerilogによるDPI（Direct Program Interface）適用事例紹介**
- **C言語、SystemC、SystemVerilogなど、多種の言語で作成した検証コンポーネントを組み上げる手法を適用事例とともに解説**

＜ポイント＞

5.1 Inter-Language Function Calls between SystemC and SystemVerilog
- 主にtaskなどのfunction callを意識した環境構築事例

5.2 A Reusable Real-Time GUI Scoreboard for Your Self-Checking Testbench Using the SystemVerilog DPI
- C++で構築したスコアボードを用いたSystemVerilogによる検証環境の構築事例

5.3 Hardware/Firmware Co-Verification Using ISS Integration and a SystemVerilog DPI
- プロセッサ開発を例に、SystemVerilogのDPI導入のメリットを紹介
- ISSを使用することで、ソフトウェア開発を早期に開始、デバッグ効率を向上

Panasonic

7

参加プログラム

Keynote

●**Taking An Enterprise-Wide Approach to Next-generation System Level Development**

Moshe Gavrielov
- Executive Vice President and General Manager, Verification Division, Cadence Design Systems, Inc.

＜内容＞

- **次世代のシステムレベル開発に向けての施策の発表**
- **A380（エアバス社の世界最大旅客機）の供給問題等を例に、不具合が市場に与える深刻さを発表**
- **設計が複雑化する中で、リスピンや不具合を市場に出してしまうリスクを排除する必要がある**
- **設計スタイルとして、システムレベルから機能検証・物理設計・電力まで統合的なマネージメントが必要**

Panasonic

8

まとめ

- **SystemVerilogの動向調査：**
  - **大手メーカでの応用事例を含み、具体的な利用技術・ノウハウ発表が多数あり、普及が拡大している**
  - **RTLに対するSystemVerilog適用が、国内よりも進んでいる**
    - RTLへの適用は国内では事例が少ないが、今回多くの事例発表があった
- **EDA技術動向**
  - **設計フェーズ個々のEDA技術から、今後は設計全体を最適化するメソドロジ・マネージメントの自動化技術へ移行**
- **全体を通じて：**
  - **ユーザ事例が多かったのが印象的**

Panasonic

4.4 SystemC/SystemVerilog 対訳表

［注意 1］ 訳語間の「・」は単語の切れ目を見やすくするために挿入してあり、使用時には使用者の判断で省略可能

［注意 2］ keyword を翻訳するにあたり、SystemVerilog の観点から大幅な意訳が必要なものには、keyword 欄に(注 SV)を挿入

keyword or phrase	訳語(決定)
!=? （注 SV）	ワイルドカード非等価演算子
##0 （注 SV）	シーケンス・オーバラップ連接
.* port connection	.*ポート接続
.name port connection	.name ポート接続
?: （注 SV）	条件演算子
[=n] （注 SV）	シーケンス非連続的繰り返し
[->n] （注 SV）	シーケンス goto 繰り返し
|=> （注 SV）	非オーバラップ含意演算子
|-> （注 SV）	オーバラップ含意演算子
==? （注 SV）	ワイルドカード等価演算子
abstract base class	抽象基底クラス
abstract class	抽象クラス
active region	アクティブ領域
aggregate expression	集合体表現
aggregate type	集合体型
alias	エイリアス
antecedent	前提部
application	アプリケーション
arc coverage	アーク・カバレッジ
arc transition	アーク遷移
architecture exploration	アーキテクチャ探索
argument	引数
argument mode	引数モード
argument passing	引数渡し
argument passing by name	引数の名前渡し
array literals	配列リテラル
array locator methods	配列ロケータ・メソッド
array ordering methods	配列オーダリング・メソッド
array part selects	配列部分選択
array querying functions	配列クエリ・メソッド
array reduction methods	配列リダクション・メソッド
array slices	配列スライス
assert	アサート
assert quiescent state	静止状態アサート
assertion	アサーション
assertion density	アサーション密度
assertion-based design	アサーションベース設計
assertion-based verification	アサーションベース検証

keyword or phrase	訳語(決定)
assignment compatible type	アサイン可能型(タイプ)
assignment operator	アサイン演算子
associative array methods	連想配列メソッド
associative arrays	連想配列
assume	アシューム
assume-guarantee reasoning	アシュームの整合性保証
assumption	仮定
atomic proposition	原子命題
attribute	属性
automatic task	オートマチック・タスク
back-to-back grant	連続承認通知
base class	基底クラス
base type	基底型(タイプ)
behavior	動作
binary fixed-point representation	固定小数点表現
binding	接続
bins	ビン
bit concatenation	ビット連結
bit vector	ビット・ベクタ
bit-select	ビット選択
bit-stream casting	ビット列型変換
bit-true	ビット精度動作
bitwise	ビット毎
black-box testing	ブラックボックス・テスト
black-box verification	ブラックボックス検証
block name	ブロック名
blocking	ブロッキング
blocking assignment	ブロッキング代入
block-level	ブロックレベル
body	本体
boolean condition	ブール式条件
boolean expression	ブール式
boolean layer	ブーリアン・レイヤ
boundary condition	境界条件
bounded-model checking	境界付きモデル・チェック
branch coverage	ブランチ・カバレッジ
branch unit	ブランチ･ユニット
branching-time temporal logic	分岐時間時相論理
built-in method	組み込みメソッド
built-in package	組み込みパッケージ
c++ exception	C++ 例外
callback	コールバック
cast compatible type	型変換可能型(タイプ)
cast operator	型変換演算子
casting	型変換

keyword or phrase	訳語(決定)
channel	チャネル
class	クラス
class definition	クラス定義
class member	クラス・メンバ
class method	クラス・メソッド
class parent member	親クラスのメンバ
class scope operator	クラス・スコープ演算子
class template	クラス・テンプレート
c-like	C 言語ライク
clock	クロック
clock tick	クロック・ティック
clocked thread	クロック同期スレッド
clocked thread process	クロック同期スレッド・プロセス
clocking block	クロッキング・ブロック
code coverage	コード・カバレッジ
coding restriction	コーディング制約
compilation unit	コンパイル・ユニット
component	コンポーネント
composition operators	構成演算子
compositional reasoning	コンポジショナル・リーズニング
concatenation	連接
concatenation operator	連接演算子
concurrent	コンカレント
concurrent assertion	コンカレント・アサーション
concurrent assertion construct	コンカレント・アサーション構文
conditional expression	条件式
conditional expression pattern	条件式パターン
conditional operator	条件演算子
configuration	コンフィギュレーション
consecutive concatenation	連続的連接
consecutive repetition	連続的繰り返し
consecutive repetition operator	連続的繰り返し演算子
consequence	帰結部
consistency assertion	整合性アサーション
constraint	制約
constraint block	制約ブロック
consumer routine	消費ルーチン
context object	コンテキスト・オブジェクト
continuous assignment	継続代入
continuous invariant	連続時間不変条件
co-operative multitasking	協調的マルチタスク
co-routine semantics	コルーチン・セマンティクス
cover	カバー
cover property	カバー・プロパティ
coverage	カバレッジ

keyword or phrase	訳語(決定)
coverage access function	カバレッジ・アクセス関数
coverage analysis	カバレッジ解析
coverage API	カバレッジ API
coverage density	カバレッジ密度
coverage metrics	カバレッジ・メトリクス
coverage model	カバレッジ・モデル
coverage option	カバレッジ・オプション
cross	クロス(カバレッジ)
cross functional coverage	クロス機能カバレッジ
cycle delay operator	サイクル遅延演算子
cycle-accurate	サイクル精度
cycle-approximate	近似サイクル精度
cycle-based semantics	サイクルベース・セマンティクス
cycle-true	サイクル精度
data member	データ・メンバ
data type	データ型(タイプ)
declaration	宣言
decrement operator	デクリメント演算子
default argument	デフォルト引数
delta cycle	デルタ・サイクル
delta notification	デルタ遅延通知
design assertion	設計アサーション
design insight	設計上の意図
design pattern	デザイン・パターン
Direct Programming Interface (DPI)	ダイレクト・プログラミング・インタフェース
directed test	ディレクティッド・テスト
directive	ディレクティブ
DPI export	DPI エクスポート
DPI import	DPI インポート
dynamic array	動的配列
dynamic array method	動的配列メソッド
dynamic process	動的プロセス
dynamic property checker	動的プロパティ・チェッカ
dynamic sensitivity	動的センシティビティ
dynamic spawned process	動的起動プロセス
dynamic variable	動的変数
elaboration	エラボレーション
elementary channel	基本チャネル
encapsulation	カプセル化
ended （注 SV)	シングルクロック・シーケンス終端
ending event	終端イベント
end-to-end	エンドツーエンド
enumerated type	列挙型
enumerated type method	列挙型メソッド
event	イベント

keyword or phrase	訳語(決定)
event finder	イベント・ファインダ
event list	イベント・リスト
eventualit y	イベンチャリティ
export	エクスポート
export binding	エクスポート接続
expression coverage	エクスプレッション・カバレッジ
expression operator	エクスプレッション演算子
fail	失敗する
failure	失敗
fairness	公平性
false fireing	誤発火
finite-precision fixed-point type	有限精度固定小数点型(タイプ)
finite-precision integer	有限精度整数
finite-precision integer types	有限精度整数型(タイプ)
fire	発火、発火する
fireing	発火
first match	最初の一致
first match operator	ファースト・マッチ演算子
fixed-point classes	固定小数点クラス
fixed-point type	固定小数点型(タイプ)
forbidden sequence	禁止シーケンス
formal verification	フォーマル検証
foundation layer implication operator	基礎層含意演算子
functional coverage	機能カバレッジ
functional coverage group	機能カバレッジ・グループ
functional coverage model	機能カバレッジ・モデル
functional coverage point	機能カバレッジ・ポイント
functional coverage specification	機能カバレッジ仕様
fusion	融合
fusion operator	融合演算子
goto repetition	goto 繰り返し
goto repetition operator	goto 繰り返し演算子
hardware design and verification language	ハードウェア設計検証言語
hierarchical channel	階層チャネル
immediate assertion	イミディエート・アサーション
immediate notification	即時通知
implementation	実装
implementation-defined	実装依存の
implication	含意
implication operator	含意演算子
implicit conversion	暗黙の変換
import	インポート
inactive region	インアクティブ領域
include file	インクルード・ファイル
inclusive operator	包含的演算子

keyword or phrase	訳語(決定)
increment operator	インクリメント演算子
inheritance	継承
initializer list	初期化リスト
initiator	イニシエータ
instance	インスタンス
instantiate	インスタンスする
instantiation	インスタンス生成
integer	整数
interface	インタフェース
Interface Method Call (IMC)	インタフェース・メソッド・コール
interface methods	インタフェース・メソッド
interface proper	インタフェース・プロパー
intersect （注 SV)	シーケンス長一致論理積
intersection	交差
invariance	不変性
invariant property	不変プロパティ
iterative constraint	繰り返し制約
iterator	イタレータ
label	ラベル
length context classes	レングス・コンテキスト・クラス
length-matching sequence connection	長さの一致するシーケンスの結合
library	ライブラリ
library map file	ライブラリ・マップ・ファイル
lifetime	ライフタイム
limited variable-precision fixed-point type	限定任意精度固定小数点型(タイプ)
limited-precision fixed-point classes	限定精度固定小数点クラス
limited-precision fixed-point type	限定精度固定小数点型(タイプ)
limited-precision integer	限定精度整数
limited-precision integer types	限定精度整数型(タイプ)
line coverage	ライン・カバレッジ
linear-time temporal logic	線形時間時相論理
list method	リスト・メソッド
literal value	リテラル値
liveness property	活性プロパティ
lock	ロック
logic vector	論理ベクタ
logical implication operator	論理的含意演算子
longest static prefix	最長静的プリフィックス
macro	マクロ
mailbox	メールボックス
master	マスタ
match	一致
matched (注 SV)	マルチクロック・シーケンス終端
member function	メンバ関数
method	メソッド

keyword or phrase	訳語(決定)
method process	メソッド・プロセス
method process instance	メソッド・プロセス・インスタンス
model checking	モデル・チェック
modeling layer	モデリング・レイヤ
module	モジュール
module hierarchy	モジュール階層
module instance	モジュール・インスタンス
module instantiation	モジュール・インスタンス生成
module name	モジュール名
multiple dimension array	多次元配列
multiple event notification	多重イベント通知
multiple out-of-order	多重アウトオブオーダ(ランダム順序)
multiple request transaction timing	多重要求トランザクション・タイミング
multiport	マルチポート
mutex	ミューテックス
mutual	相互
named assertion	名前付きアサーション
named binding	名前による接続
named block	名前付きブロック
named port connection	名前付きポート接続
native compilation	ネイティブ・コンパイル
NBA region	NBA 領域
nested identifier	ネストした識別子
nested module	入れ子のモジュール
non-abstract class	非抽象クラス
nonblocking	ノンブロッキング
non-blocking assignment (NBA)	ノンブロッキング代入
non-clocked procedural assertion	クロックなし手続きアサーション
nonconsecutive count repetition	非連続的繰り返し
nonconsecutive exact repetition	非連続的終端合致繰り返し
nonconsecutive repetition	非連続的繰り返し
nonconsecutive repetition operator	非連続的繰り返し演算子
non-corrupt data assertion	ノンコラプト・データ・アサーション
non-inclusive operator	非包含的演算子
non-length-matching sequence connection	長さの異なるシーケンスの結合
non-overlapped	非重複
non-overlapped implication	非重複含意
notification	通知
notify	通知する
numeric type	数値型(タイプ)
object	オブジェクト
object hierarchy	オブジェクト階層
observe region	観測領域
operator	演算子
operator associativity	演算子の結合順位

keyword or phrase	訳語(決定)
operator overloading	演算子のオーバロード
operator precedence	演算子の優先順位
oring sequences	シーケンス論理和
out-of-order protocols	アウトオブオーダ・プロトコル
out-of-order request	アウトオブオーダ要求
out-of-order transaction	アウトオブオーダ・トランザクション
over constrained	過制約
over constrained procedural assertion	過制約手続きアサーション
overflow mode	オーバーフロー・モード
overlapped implication	時間重複含意
overlapping implication	時間重複含意
overload	オーバーロード
override	オーバーライド
package	パッケージ
packed array	パック型配列
packed structure	パック型構造体
parameter	引数 または パラメータ
part-select classes	部分選択クラス
part-selects	部分選択
pass statement	パス文
path coverage	パス・カバレッジ
pin-accurate	信号レベル精度
PLI callback	PLI コールバック
polling gets	ポーリング受信
polling peeks	ポーリング参照
polling puts	ポーリング送信
polymorphism	ポリモーフィズム
pop	ポップ
port	ポート
port binding	ポート接続
port instantiation	ポート・インスタンス
port policy	ポート・ポリシー
portless channel access	ポートなしチャネルアクセス
positional binding	位置による接続
positional port binding	位置によるポート接続
post-NBA region	ポスト NBA 領域
post-observed region	ポスト観測領域
postponed region	後処理領域
pragma	プラグマ
pre-active region	プリアクティブ領域
pre-NBA region	プリ NBA 領域
preponed region	前処理領域
prerequisite sequence	前提シーケンス
primitive	プリミティブ
primitive channel	プリミティブ・チャネル

keyword or phrase	訳語(決定)
procedural	手続き的
procedural assinment	手続き的代入
procedural continuous assinment	手続き的継続代入
process	プロセス
process instance	プロセス・インスタンス
process sensitivity	プロセス・センシティビティ
program block	プログラム・ブロック
Programing Language Interface (PLI)	プログラミング・ランゲージ・インタフェース
programmer's view	プログラマーズビュー
property	プロパティ
property checking	プロパティ・チェック
property declaration	プロパティ宣言
property language	プロパティ言語
property structure	プロパティ構造
propositional temporal logics	命題時相論理
protocol inlining	プロトコル・インライン展開
protocol layer	プロトコル・レイヤ
proxy class	代理クラス
pure virtual functions	純粋仮想関数
push	プッシュ
quantization mode	量子化モード
query	クエリ
quiescent state assertions	静的状態アサーション
random constraint	ランダム制約
random distribution	ランダム分布
random number generator (RNG)	乱数発生器
random test	ランダム・テスト
random test generator	ランダム・テスト生成
randomization method	ランダム・メソッド
reactive region	リアクティブ領域
reactive testbench	反応型テストベンチ
real literal	実数リテラル
recursive property	帰納的プロパティ
reduction operator	リダクション演算子
reduction or	リダクション OR
reference model	リファレンス・モデル
regression test	リグレッション・テスト
repetition	繰り返し
request-update method	要求-更新メソッド
resolution operator	リゾリューション演算子
resolved	リゾルブ
resolved signal	リゾルブ信号
restriction	制限
return value	返り値
rounding modes	まるめ処理モード

keyword or phrase	訳語(決定)
safety property	安全性プロパティ
sample point	観測点
scheduling semantics	スケジューリング・セマンティクス
scope	スコープ
scope resolution operator	スコープ・リゾリューション演算子
semaphore	セマフォ
semi-formal	セミフォーマル
sensitivity	センシティビティ
sequence	シーケンス
sequence and operator（注 SV）	シーケンス長不一致論理積
sequence declaration	シーケンス宣言
sequence expression	シーケンス式
sequence fusion	シーケンス融合
sequence fusion operator	シーケンス融合演算子
sequence implication operator	シーケンス含意演算子
sequence intersection	シーケンス交差(演算子)
sequence or operator	シーケンス論理和演算子
sequential assertions	シーケンシャル・アサーション
severity level	重大さ
severity task	重大タスク
sign extension	符号拡張
signal	信号
signature	シグネチャ
signed type	符号付き型(タイプ)
simulation	シミュレーション
simulation time	シミュレーション時間
simulation time resolution	シミュレーション時間精度
singular type	単体型(タイプ)
slave	スレーブ
sparse array	スパース配列
spawn	生成する
spawned process	起動プロセス
spawned process instance	起動プロセス・インスタンス
specification-level	仕様レベル
specify	指定する、仕様記述する
starting event	開始イベント
state sequence coverage	状態系列カバレッジ
state-explosion	状態爆発
static processes	静的プロセス
static sensitivity	静的センシティビティ
static spawned process	静的起動プロセス
static tasks	静的タスク
stimulus	スティミュラス
streaming operator	ストリーム演算子
string literal	文字列リテラル

keyword or phrase	訳語(決定)
strong operator	強演算子
strong suffix implication	強接尾子含意
subordinate Boolean expressions	従属ブール式
suffix implication	接尾子含意
suffix next implication	ネクスト接尾子含意
suffix sequence	帰結シーケンス
syndrome	シンドローム
system function	システム関数
system-level	システムレベル
tag	タグ
tag set	タグ・セット
tag-based	タグベース
tagged	タグ付き
tagged union	タグ付きユニオン
target	ターゲット
task	タスク
template	テンプレート
template parameters	テンプレート・パラメータ
temporal	テンポラル、時相
temporal concatenation	時間的連接
temporal layer	テンポラル・レイヤ
temporal property	テンポラル・プロパティ
terminating property	終了プロパティ
testbench	テストベンチ
thread	スレッド
thread process	スレッド・プロセス
thread process instance	スレッド・プロセス・インスタンス
throughout（注 SV）	シーケンス全域条件
time resolution	時間精度
time slots	タイム・スロット
time unit	時間単位
time window	時間ウィンドウ
time-annotated model	タイミング付加モデル
time-bounded window	時間境界ウィンドウ
timed	タイムド
timed notification	タイムド通知
time-out	タイムアウト
top-level module	最上位モジュール
top-level object	最上位オブジェクト
trace	トレース
trace file	トレース・ファイル
transaction	トランザクション
transaction-level	トランザクションレベル
transactor	トランザクタ
transient behavior	過渡状態

keyword or phrase	訳語(決定)
transition	遷移
transport layer	トランスポート・レイヤ
traverse object	トラバース・オブジェクト
trigger	トリガ
triggering event	トリガとなるイベント
type	型、タイプ
unary delay	単項遅延
union	共用体
unpack	アンパック
unpacked array	アンパック型配列
unsigned type	符号なし型(タイプ)
unsized literal	不定長リテラル
unspawned process	未起動プロセス
unspawned process instance	未起動プロセス・インスタンス
untimed	アンタイムド
update	更新
update request	更新要求
user layer	ユーザ･レイヤ
user-defined	ユーザ定義
user-defined conversion	ユーザ定義変換
user-defined type	ユーザ定義型(タイプ)
variable-precision fixed-point classes	任意精度固定小数点クラス
variable-precision fixed-point type	任意精度固定小数点型(タイプ)
vector	ベクタ
verification layer	検証レイヤ
violation	違反
virtual interface	仮想インタフェース
void return value	void 戻り値
weak operator	弱演算子
weak suffix implication	弱接尾辞含意
weak suffix implication operator	弱接尾辞含意演算子
weight operator	重み付け演算子
white-box test	ホワイトボックス・テスト
wild equality operator	ワイルドカード等価演算子
wild inequality operator	ワイルドカード非等価演算子
wildcard	ワイルドカード
wildcard operator	ワイルドカード演算子
within（注 SV）	シーケンス包含
zero extension	ゼロ拡張

ＥＤＡアニュアルレポート２００６

２００７年５月発行

発　　行　　社団法人 電子情報技術産業協会　電子デバイス部
〒101-0062
東京都千代田区神田駿河台3-11
三井住友海上駿河台別館ビル
電話　03-3518-6430　　FAX　03-3295-8725
印刷・製本　　株式会社 オガタ印刷
〒102-0072
東京都千代田区飯田橋1-5-6
電話　03-3264-3456