JEITA

EDAアニュアルレポート　2007

Annual Report on Electronic Design Automation

－　65nmから45nmテクノロジ世代のEDA技術の進展に向けて　－

2008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

【巻頭言】

「65nmから45nm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　2007年度　委員長　　齋藤 茂美

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 SI (Signal Integrity) や 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。

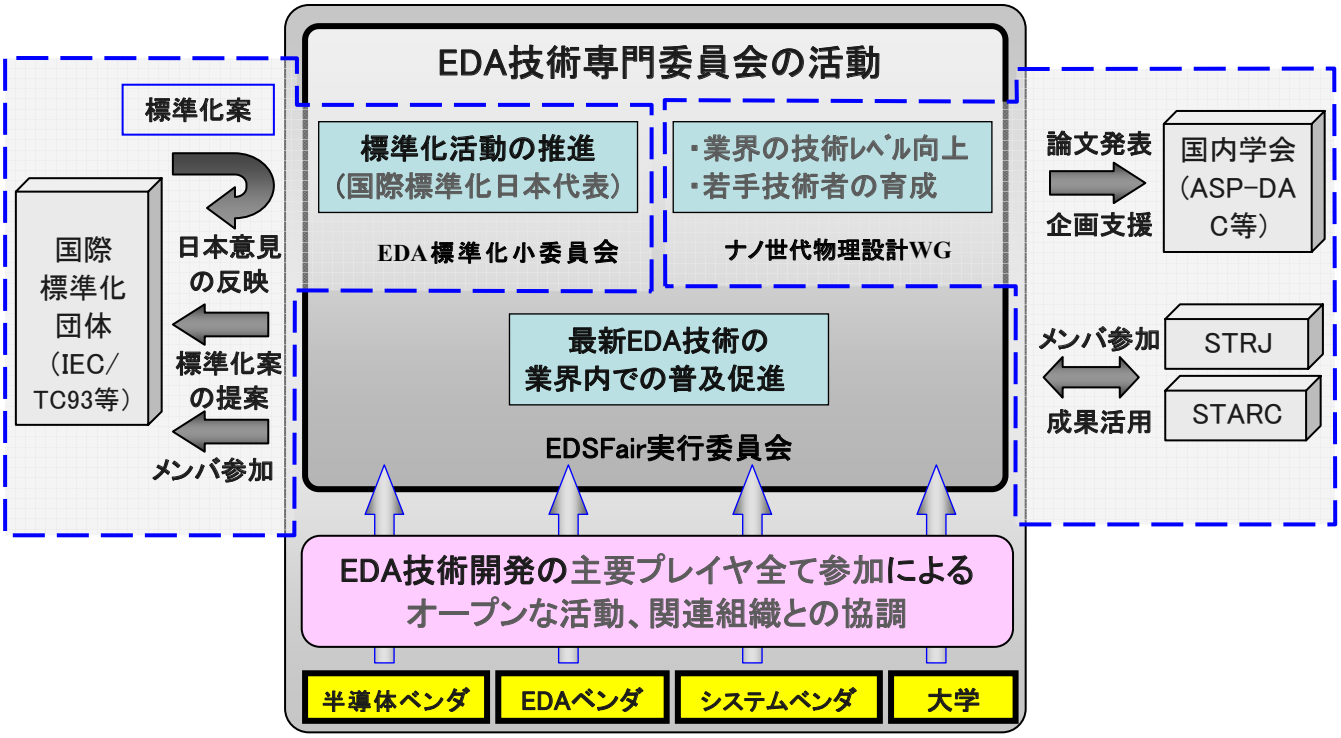
EDA 技術専門委員会は、電子情報技術産業協会(JEITA)における業界活動組織の一つとして、電子機器の設計自動化(EDA : Electronic Design Automation)に関わる様々な活動を行っている。特に、電子機器の機能・性能を決定するシステム LSI 設計技術に関して、以下の 3 つの領域で活動している。

第 1 の活動の領域は、システム LSI 設計技術に関する調査、検討と課題解決手法の提案である。本年度はプロセスばらつきと配線モデル抽出 などのプロセス微細化に伴う設計課題とその課題解決手法につき、新発足のナノ世代物理設計ワーキンググループが検討を行った。

第２の活動領域は、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。IEC、IEEE 等の国際的な標準化活動に対し EDA 標準化小委員会を中心に、標準化提案の検証、技術提案・交流会議などを行なっている。設計記述言語 SystemC および SystemVerilog の IEEE 標準化では、本委員会の各タスクグループが、2005 年末の標準化に貢献し、その後も両標準の普及推進活動と、追加標準仕様の検討、提案を行っている。今年度は新たに、二つの標準化グループが存在し混迷する Power Format 検討ワーキンググループを発足し、業界としての検討を行った。また、EDA 標準化小委員会は電子情報通信学会内の IEC/TC93 国内委員会/WG2(ハードウェア設計記述言語)としての標準化活動も行っており、2007 年 9 月には IEC/TC93 国際会議(カイザーバーグ、米国)に参加し、国際標準化の課題を議論してきた。

最後に、第 3 の活動領域は、EDA 技術および標準化の普及、推進のためのイベント開催、支援である。本年度も、2008 年 1 月にパシフィコ横浜にて EDSFair2008 (Electronic Design and Solution Fair 2008) を開催した。EDSFair は、電子機器の設計技術に関する、国内唯一の総合的な展示会であり、同時開催の FPGA コンファレンスおよびシステム・デザイン・フォーラム 2008 と合わせ、特設ステージでの当委員会企画イベントも含めて積極的・精力的に、企画、出展者誘致、来場者誘致活動を行った。その結果、過去最大数の出展者からの展示があり、最新の EDA 技術と関連情報との出会いの場として、LSI 設計者・電子機器設計者をはじめとした 1 万人以上の方に参加頂いた。

次の図でこれらの委員会活動と関係する団体の関係を示す。



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、上図の関連組織・団体との密な連携のもと、技術検討、標準化、そしてそれらの普及促進という 3 つの活動領域の活動を通じ、65nm から 45nm テクノロジー世代の「システム・オン・チップ時代」の電子情報機器業界に対し、さらには地球温暖化などの世界規模での課題に対し、設計技術の面で貢献し、「システム・オン・チップ時代」がもたらす産業界の変革を乗り越えて、日本の電子情報機器業界の発展に寄与すべく、本年度 19 社約 60 名の業界各社・有志メンバーの参画で運営してきた。

今年度取り組んだ委員会活動の活性化の為の新規会員勧誘が実を結び、来年度は新会員会社3社を加え、22社での活動となる予定である。「システム・オン・チップ時代」に入り、半導体および電子機器が切り拓く素晴らしい未来が今後も広がることを確信しつつ、2008年度も積極的な活動を展開していきたい。

本冊子「ＥＤＡアニュアルレポート2007」は、EDA技術専門委員会の2007年度年次報告として、上記３つの活動領域について、活動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)

2007 年度 JEITA/EDA 技術専門委員会 委員一覧

委員長	齋藤　茂美	ソニー㈱	半導体事業本部 設計基盤技術部門 企画部担当部長
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)
副委員長	太田　光保	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 設計第三開発グループ　参事
監事	河村　薫	富士通㈱	テクノロジ開発統括部 主席部長
幹事	藤波　義忠	NEC エレクトロニクス㈱	基盤技術開発事業本部 技術企画室 グループマネージャー
	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 担当部長
	西本　猛史	シャープ㈱	電子デバイス開発本部 第３開発室 副参事
	熊谷　敬	セイコーエプソン㈱	半導体事業部 IC 設計部 部長
	南　文裕	㈱東芝	システム LSI 設計技術部 設計メソドロジ技術開発 担当主査
	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
	江田　努	ローム㈱	YTC 開発システムユニット LSI 企画推進 G 次席技術員

委員	下出　隆文	三洋半導体㈱	SS 事業本部 SS 事業推進統括部 設計技術部課長
	佐々木則之	㈱ジーダット	EDA 営業技術部 シニアマネージャ
	横川　隆	㈱図研	SoC 事業部 デザインセンター部長
	塩谷　俊人	凸版印刷㈱	半導体ソリューション事業本部 企画部課長
	安井　孝史	日本ケイデンス・デザイン・システムズ社	TFO　Director
	飯島　一彦	日本シノプシス㈱	技術本部 本部長
	瀬谷　和宏	丸紅情報システムズ㈱	半導体システム事業部 副事業部長
	三橋明城男	メンター・グラフィックス・ジャパン㈱	ストラテジック・ビジネス・ディベロップメント部 テクニカルディレクター
	前野　酉治	㈱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 シニアスペシャリスト
	浅井　健史	ローム㈱	開発システムユニット IP 開発チーム課長
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
	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 設計第三開発グループ チームリーダー
	長谷川　隆	富士通㈱	設計技術統括部 第 3 設計部部長
	小島　智	NEC システムテクノロジー㈱	PF 統括本部 CWB 事業推進室 テクニカルマーケティング ディレクター

客員	神戸　尚志　近畿大学	理工学部　電気電子工学科 教授
	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
	岡村　芳雄　㈱半導体理工学研究センター	執行役員・開発第２部長
	若林　一敏　日本電気㈱	中央研究所 ビジネスイノベーションセンター EDA 開発センター研究部長

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
社団法人日本プリント回路工業会

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

CPF Common Power Format
Si2 で標準化されている SOC内の電源情報を記述するフォーマット

DCL Delay Calculation Language
遅延計算のための記述言語

DPCS　Delay and Power Calculation System
IEEE1481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

UPF　Unified Power Format
IEEE/Accellera で標準化されている SOC内の電源情報を記述するフォーマット

VHDL　VHSIC（Very High Speed Integrated Circuit）Hardware Description Language
IEEE1076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

1.1　2007 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 に関連する技術およ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	5 名
客員	4 名

<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
3.2007 年度の予算運用および検討項目
　年度末の適切な繰越金確保。
　メンバ会社増など収入増の検討

委員会の役員

委員長：	ソニー	齋藤	茂美
副委員長（正）：	三洋電機	山田	節
副委員長（代行）：	松下電器産業	太田	光保
監事：	富士通	河村	薫

下部組織の役員

EDA 標準化小委員会	主査：	三洋電機	山田　節
SystemC　WG	主査：	富士通	長谷川　隆
SystemVerilog　WG	主査：	松下電器産業	浜口加寿美
Power Format 検討 WG	主査：	富士通	中森　勉
ナノ世代物理設計 WG	主査：	富士通	奥村　隆昌
EDSFair 実行委員会	委員長：	ルネサス	秋山　俊恭

EDA 技術専門委員会メンバと担当　（2007 年 4 月 11　日現在の委員交替情報を反映）（敬称略）

委員長　：ソニー	齋藤　茂美	EDSFair/ASP-DAC 小委員会主査、ASP-DAC 支援
副委員長：三洋電機	山田　節	EDA標準化小委員会主査、国際対応支援委員会委員
副委員長：松下電器産業	太田　光保	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、内規改訂標準化小委員会副主査、EDSF 実行委員
監事　：富士通	河村　薫	EDSFair 実行委員、EDSFair 企画 WG 主査
幹事　：ルネサス	秋山　俊恭	EDSFair 実行委員長 システム・デザイン・フォーラム実行委員
同　：NEC ｴﾚｸﾄﾛﾆｸｽ	藤波　義忠	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
同　：シャープ	西本　猛史	ホームページ、メールシステム、予算実績管理
同　：東芝	南　文裕	EDSFair 実行委員、広報パンフレット
同　：ローム	浅井　健史	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　：セイコーエプソン	熊谷　敬	
同　：沖	灘岡　満	EDSFair 企画 WG メンバ （輪番では EDSF 実行委員。藤波委員と交替）

委員

同　：ジーダット	佐々木則之	
同　：凸版印刷	稲垣　賢	（EDSFair 企画 WG メンバ）
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	三橋明城男	
同　：リコー	前野　酉治	
同　：図研	横川　隆	

特別委員			
同	: 富士通 VLSI	奥村　隆昌	ナノ世代物理設計ワーキンググループ主査
同	: 松下電器産業	浜口加寿美	SystemVerilog ワーキンググループ主査
同	: 富士通	長谷川　隆	SystemC ワーキンググループ主査
同	: 富士通	中森　　勉	Power Format 検討 WG 主査
同	: NEC　ST	小島　　智	ＴC93/ＷＧ2 コンベナ
客員	: 近畿大学	神戸　尚志	元委員長、TC93 国内委員長
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/電子ﾃﾞﾊﾞｲｽ部	古川　　昇	
同	: 同	岩渕　幸吾	

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) ワーキンググループによる技術動向、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
→ ナノ世代物理設計ワーキンググループ

(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、イベントへの参加と連携
ASP-DAC2008, DAC2007, 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 設計言語およ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 EDA 標準化小委員会下のワーキング
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 などに対しては
→ EDA 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・連携と標準化活動への協力
・IEC/TC93（および WG2）国際会議(2007/9 米国)への参加
・IEEE/DASC,IEEE-SA, Accellera, Si2 等への参加および連携
→ EDA 標準化小委員会にて検討し、対応

3. EDA 技術および標準化の普及、推進のためのイベント実施、支援

(1) 「EDSFair2008」（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 2008」を EDSFair2008 と同時開催

(3) 「ASP-DAC2008」（Seoul）
→ Designer's Forum との連携

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) 2007.04 に EDSFair 臨時検討会にて、ASP-DAC2008　Designers Forum とシステムデザイン
フォーラム 2008、EDSFair の EDA-TC 企画イベントとの連携・整合につき協議。

＜別紙 - ２＞

2007 年度　　JEITA/EDA 技術専門委員会 会合予定

年/月	技術専門委員会	懇親会	関連イベント
2007/4	4/20(金) (東京地区)　議事録　凸版 ・07 年度事業計画説明、承認 ・07 年度プロジェクト/研究会計画説明、承認 ・06 年度会計収支と 07 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	○	・DATE2007（4/16-4/19）　@ Nice ・軽井沢 WS (4/23-24)　　@軽井沢
2007/5			
2007/6	6/15(金) (東京地区)　議事録　ジーダット ・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		・DAC2007　(6/4-6/8) @San　Diego, California
2007/7			
2007/8			・DA シンポジウム 2007 (8/29-30) @浜松
2007/9	9/21(金)　(関西地区) 議事録　日本ケーデンス ・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	○	・ＴＣ93国際会議2007（9/5-9/8） @Gaithersburg,Maryland, U.S
2007/10			
2007/11	11/16(金) (東京地区)　議事録　日本シノプシス ・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		・ICCAD2007 (11/5-11/8) @San Jose, California
2007/12			
2008/1	1/18(金)(関西地区) 議事録　丸紅ソリューション ・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 ・08 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-DAC2008 (1/21-1/24) @Seoul ・EDSFair2008 (1/24-1/25)　@横浜
2008/2			
2008/3	3/14(金) (東京地区)　議事録　メンタージャパン ・半導体部会/半導体技術委員会報告内容説明 ・07 年度プロジェクト/研究会の年間活動報告 ・07 年度予算消費状況 ・08 年度事業計画説明 ・08 年度プロジェクト/研究会の年間活動計画説明		・DATE2008 (3/10-3/14)@ Munich, 独

1.2　2007 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

2006 年度より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、メイテナンスについてもご協力を頂いている。

（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
- ナノ世代物理設計 WG
- 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。（英語版は一部）

- EDA 技術専門委員会（過去のアニュアルレポート）
- EDA 標準化小委員会（SystemC ユーザフォーラム資料）
- ナノ世代物理設計 WG（過去の資料。物理設計標準化研究会時代を含む）
- EDSFair 実行委員会（システムデザインフォーラムの紹介）
- システムレベル設計研究会（旧サイトへのリンク）

(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、イベントの紹介およ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　2007 年度 JEITA/EDA 技術専門委員会 年間実績・予定表

月	EDA 技術専門委員会	
	幹事会	委員会
2007 年 4 月		4/20（金）　14-17　JEITA　306 会議室
5 月		
6 月		6/15（金）　14-17　JEITA　601 会議室
7 月		
8 月		
9 月		9/21（金）　14-17　JEITA 関西支部　8 階会議室
10 月		
11 月		11/16（金）　14-17　JEITA　301 会議室
12 月		
2008 年 1 月		1/18（金）　14-17　中央電気倶楽部　207 会議室
2 月		
3 月		3/14（金）　14-17　JEITA　404 会議室

月	EDA 標準化小委員会関連
2007 年 4 月	4/25(水) 15:00-18:00　第 1 回 SystemC-WG　富士通株式会社川崎工場 本館 2F 第 10 応接室
5 月	5/11(金) 11:00-13:00　第 1 回 EDA 標準化小委員会　電子情報通信学会 69 会議室 5/11(金)-5/12(土)　11:00-17:00　第 1 回 SystemVerilog-WG　NTT 5/18(金) 13:00-17:00　第 2 回 SystemC-WG　JEITA　315 会議室
6 月	6/15(金) 13:00-17:00　第 3 回 SystemC-WG　JEITA　315 会議室
7 月	7/ 5(木) 11:00-13:00　第 2 回 EDA 標準化小委員会　電子情報通信学会 69 会議室 7/20(金) 13:00-17:00　第 4 回 SystemC-WG　JEITA　310 会議室
8 月	8/24(金) 11:00-13:00　第 3 回 EDA 標準化小委員会　電子情報通信学会 69 会議室
9 月	9/14(金) 14:00-17:00　第 5 回 SystemC-WG　JEITA　311 会議室
10 月	10/11(木) 17:30-19:30　第 1 回 PowerFormat 検討 WG　日本ｴﾚｸﾄﾛﾆｸｽｼｮｰ協会　会議室 10/12(金) 11:00-13:30　第 4 回 EDA 標準化小委員会　電子情報通信学会 69 会議室 10/19(金) 13:00-17:00　第 6 回 SystemC-WG　JEITA　関西支部　第 1 会議室
11 月	11/ 7(水) 13:30-17:00　第 2 回 Power Format 検討 WG JEITA　313 会議室 11/12(月) 13:30-17:00　第 2 回 SystemVerilog-WG　松下電器産業　長岡京テクノ棟 11/16(金)-11/17(土)　第 7 回 SystemC-WG(集中審議) 鬼怒川温泉ホテル　第一会議室 11/29(木) 13:30-17:00　第 3 回 PowerFormat 検討 WG　中央大学駿河台記念館　500 会議室
12 月	12/19(水) 14:00-17:30　第 4 回 PowerFormat 検討 WG　フォーサイト　第 3 教室 12/21(金) 13:00-17:00　第 8 回 SystemC-WG 龍名館　2 階　龍の間
2008 年 1 月	1/17(木) 11:00-17:00　第 5 回 PowerFormat 検討 WG　JEITA 関西支部　第 2 会議室 1/18(金) 13:00-17:00　第 9 回 SystemC-WG　TKP 九段下　第 3 会議室
2 月	2/ 1(金) 11:00-13:00　第 5 回 EDA 標準化小委員会　電子情報通信学会 69 会議室 2/15(金) 13:00-17:00　第 10 回 SystemC-WG JEITA　503 会議室 2/28(木) 14:00-17:00　第 6 回 PowerFormat 検討 WG　JEITA　501 会議室 2/29(金) 13:00-17:00　第 3 回 SystemVerilog-WG　東京　日本シノプシス会議室
3 月	3/21(金) 13:00-17:00　第 11 回 SystemC-WG JEITA　503 会議

月	ナノ世代物理設計 WG 関連
2007 年 4 月	
5 月	5/25(金) 13:00-17:00　第 1 回ナノ世代物理設計 WG　ハーモニーホール 3 階小会議室
6 月	6/21(金) 12:00-17:00　第 2 回ナノ世代物理設計 WG　JEITA　306 会議室
7 月	7/27(金) 12:00-17:00　第 3 回ナノ世代物理設計 WG　JEITA 関西支部　第 1 会議室
8 月	8/24(金) 12:00-17:00　第 4 回ナノ世代物理設計 WG　東京工業大学　すずかけホール 2 階
9 月	9/28(金) 11:00-17:00　第 5 回ナノ世代物理設計 WG　JEITA 関西支部　第 1 会議室
10 月	10/26(金) 11:00-17:00　第 6 回ナノ世代物理設計 WG　JEITA　306 会議室
11 月	11/30(金)-12/1(土)　第 7 回ナノ世代物理設計 WG　ひょうご共済会館　会議室スミレ
12 月	12/20(木) 11:00-17:00　第 8 回ナノ世代物理設計 WG　JEITA　313 会議室
2008 年 1 月	1/17(木) 13:00-17:00　第 9 回ナノ世代物理設計 WG　JEITA 関西支部　第 1 会議室
2 月	2/22(金) 13:00-17:00　第 10 回ナノ世代物理設計 WG JEITA　503 会議室
3 月	3/21(金)　9:00-12:30、15:15-17:15 第 11 回ナノ世代物理設計 WG　JEITA　405、502 会議室 3/21(金) 13:00-15:00　ナノ世代物理設計 WG/成果報告会　JEITA　506 会議室

月	EDS Fair 実行委員会関係
2007 年 4 月	4/11(金) 15:00-17:00　第 2 回実行委員会　芝パークホテル別館「アイリス」
5 月	5/11(金) 15:00-17:00　第 3 回実行委員会　JESA 会議室
6 月	6/14(木) 15:00-17:00　第 4 回実行委員会　JESA 会議室
7 月	7/20(金) 14:00-17:00　第 5 回実行委員会　JESA 会議室
8 月	8/ 9(木) 14:00-17:00　第 6 回実行委員会　JESA 会議室
9 月	9/13(木) 14:00-17:00　第 7 回実行委員会　JESA 会議室
10 月	10/11(木) 14:00-17:00　第 8 回実行委員会　JESA 会議室
11 月	11/ 8(木) 14:00-17:00　第 9 回実行委員会　JESA 会議室
12 月	12/13(木) 14:00-17:00　第 10 回実行委員会 JESA 会議室　17:30～懇親会
2008 年 1 月	1/10(木) 14:00-17:00　第 11 回実行委員会 JESA 会議室
2 月	2/29(金) 14:00-17:00　第 12 回実行委員会 ホテル日航プリンセス京都　会議室ローズ A
3 月	

月	関連行事
2007 年 4 月	
5 月	
6 月	
7 月	
8 月	
9 月	9/3(月)-9/7(金)　IEC/TC93国際会議　NIST(Gaithersburg, MD,US)
10 月	10/12(金)　IEEE P.1801 WG との会議　電子情報通信学会　69 会議室
11 月	
12 月	
2008 年 1 月	1/23(水)　Si2 との会議　イノテックビル 1/24(木)　IEEE-DASC との会議　パシフィコ横浜　202 会議室 1/23(水)　IEC/TC93 JPN NC Extraordinary Meeting　パシフィコ横浜　202 会議室
2 月	2/19(火)- 2/21(木)　IEC/TC93/WG2 会議 サンノゼ 2/19(火)- 2/21(木)　IEEE-DASC との会議 サンノゼ
3 月	

２．各技術委員会の活動報告

2.1　ナノ世代物理設計ワーキンググループ(Nano Scale Physical Design Working 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、本ワーキンググループでは、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(45 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

2007 年 5 月から活動を開始し、今年度は、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。

- 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
- 局所温度ばらつきによる配線抵抗変動の回路特性へのインパクト調査に向けた検討

今年度の活動の成果として、統計的静的タイミング解析に関する以下の２項目について調査を行い、有用な知見を得た。

(1)Vth ばらつきに拠る出力遷移時間ばらつきの解析

65nm CMOS インバータを用いてトランジスタ閾値電圧ばらつきに対する出力遷移時間ばらつきを調査した。その結果、遅延時間と出力遷移時間の相関性が低下する入力遷移時間と出力負荷の組み合わせにおいては、従来のばらつき感度を用いた解析手法では、出力遷移時間のばらつきを必ずしも正しく表現できないことを明らかにした。さらに、電源電圧から PMOS 閾値の絶対値と NMOS の閾値を差し引いた電圧を Vm と定義し、従来のばらつき感度で正しく表現できない出力遷移時間のばらつきが Vm に対する二次の相関特性として観測され、この相関特性を加味することで出力遷移時間ばらつきの計算誤差が改善できることを示した。

⑵チップ内システマティックばらつきと回路スキュー特性相関

従来必ずしも明確になっていなかったチップ内システマティックばらつきが、回路特性ばらつきに与える影響を定量的に解析した。解析には、システマティックばらつきをランダム曲面でモデル化し、モンテカルロ解析に取り込む手法を用いた。さらに、その結果を距離相関によるシステマティックばらつきのモデルによる結果と比較、検証した。

その結果、システマティック成分の空間分布が低次でさまざまな曲面を生じると仮定すると、距離相関によるシステマティックばらつきのモデルでは非常に大きな誤差を生じ得ることを示した。

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、今年度の成果の一部を以下の学会、にて発表した。

[1] 「Vth ばらつきに拠る出力遷移時間ばらつきの解析」, 回路とシステム軽井沢ワークショップ, 2008 年 4 月.

[2] 「チップ内システマティックばらつきと回路スキュー特性相関」, 回路とシステム軽井沢ワークショップ, 2008 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、EDA データベース「OpenAccess」、RTL（Register Transfer Level）から GDSII までの設計フローにおける消費電力に関する記述仕様 Common Power Format（CPF）等の標準化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/）は、SystemVerilog の策定など、Verilog-HDL に重点を置いて活動しているが、消費電力関連記述仕様 Unified Power Format（UPF）の標準化推進もおこなっている。

一方日本では、半導体 MIRAI(Millennium Research for Advanced Information Technology)プロジェクトが、ASRC（産総研次世代半導体研究センター）、ASET（技術研究組合 超先端電子技術開発機構）、Selete（株式会社 半導体先端テクノロジーズ）を中心とする産官学で共同し、半導体最先端技術の「壁」を克服する共通基盤としての役割を担っている。具体的には、半導体の微細加工において、45nm の技術世代以降の LSI の消費電力や処理速度といった基本的な性能を格段に向上させる技術を開発している。また、半導体理工学研究センターSTARC（http://www.starc.or.jp/）では、SoC 設計技術等の先導開発(あすか 2)が行われている。

2.1.4　参加メンバー

主　　査	奥 村 隆 昌	富士通 VLSI(株)
副 主 査	金 本 俊 幾	(株)ルネサステクノロジ
委　　員	中 島 英 斉	NEC エレクトロニクス(株)
同	黒 川　　敦	三洋半導体(株)
同	小 野 信 任	(株)ジーダット・イノベーション
同	森　　茂 貴	ソニー(株)
同	高 藤 浩 資	(株)リコー
同	増 田 弘 生	(株)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　EDA 標準化小委員会

2.2.1　EDA 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。

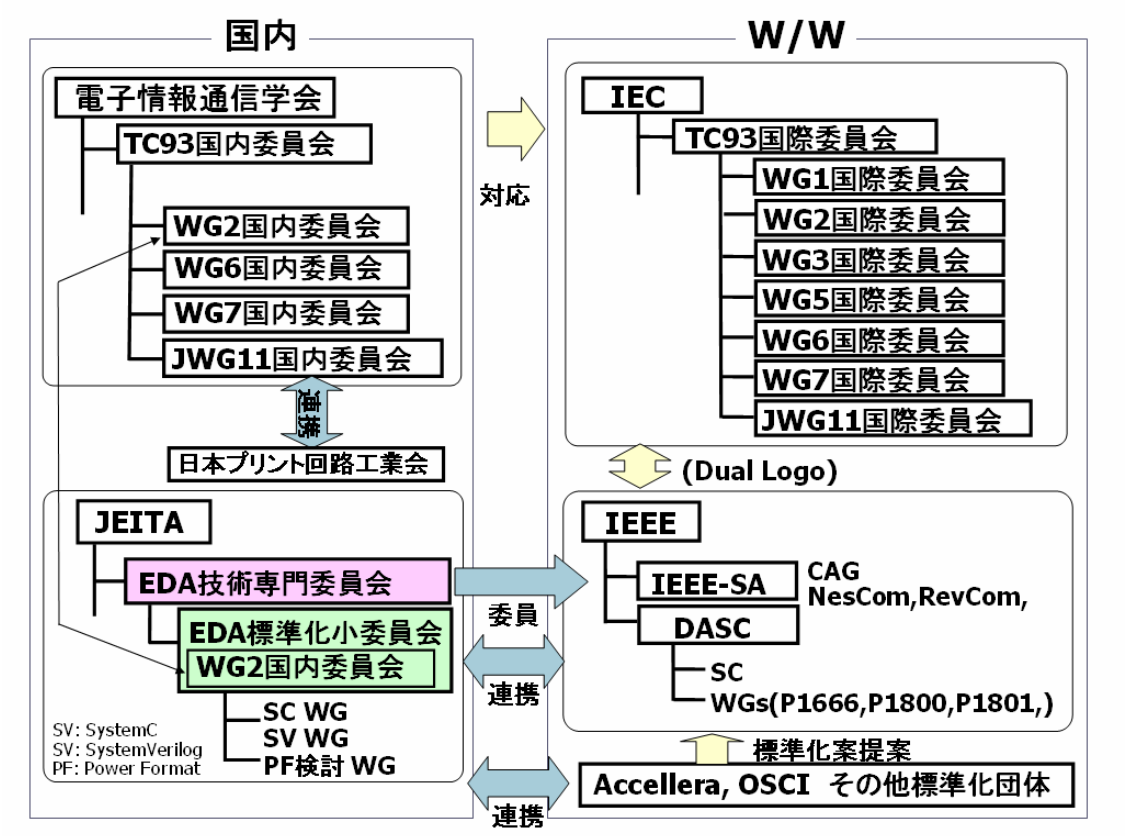
システム設計メソドロジの革新が進展する中で、設計技術言語の国際標準化は依然として重要なテーマである。標準化関連の活動をより明確に位置づけるため、2000 年 11 月に本小委員会が設立された。

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)、および 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、EDA 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。

2003 年度には、SystemC および SystemVerilog の標準化を業界として検討・推進する目的で、それぞれワーキンググループを発足させた。本度は、CPF(Common Power Format)と UPF(Unified Fower Format)の二つの Power Format の標準化案の議論と統一を目的に、検討ワーキング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ひとつであり、SystemVerilog は IEEE1364(VerilogHDL)の後継・検証技術の拡張である。CPF/UPF の Power Format は、主にシステム LSI の低消費電力化設計の効率化を目的とした設計言語である。これらワーキンググループは、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。



図－１　EDA 標準化小委員会と他の標準化組織との関係

本 EDA 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 および関連技術の標準化に関して、

・内外の動向を調査、検討し、

・技術およ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)　2008 年度標準化小委員会メンバー（2008 年 3 月現在 敬称略）

主査	山田　節	三洋電機(株)
副主査	太田　光保	松下電器産業(株)
委員	齋藤　茂美	ソニー㈱
委員	河村　薫	富士通(株)
委員	秋山　俊恭	(株)ルネサステクノロジ
委員	藤波　義忠	NEC エレクトロニクス(株)
委員	西本　猛史	シャープ(株)
委員	南　文裕	(株)東芝
委員	浅井　健史	ローム(株)
委員	熊谷　　敬	セイコーエプソン(株)
委員	灘岡　満	沖電気工業(株)
特別委員	小島　智	NEC システムテクノロジー(株) （TC93/WG2 ココンベナ）
特別委員	相京　隆	(株)半導体理工学研究センター （TC93/WG2　Member）
特別委員	長谷川　隆	富士通(株) （SystemC WG 主査、TC93/WG2　Member）
特別委員	今井　浩史	(株)東芝 （SystemC WG 副主査）
特別委員	浜口　加寿美	松下電器産業(株) （SystemVerilog WG 主査 TC93/WG2　Member）
特別委員	明石　貴昭	日本シノプシス(株) （SystemVerilog WG 副主査）
特別委員	後藤　謙治	日本ケイデンス・デザイン・システムズ社 （SystemVerilog WG 副主査）
特別委員	中森　勉	富士通(株) （Power Format 検討 WG 主査）
特別委員	奥村　隆昌	富士通(株) （ナノ世代物理設計 WG 主査）
特別委員	星野　民夫	(株)アプリスター （TC93/WG2　Member）
特別委員	石河久美子	富士通マイクロソリューションズ(株) （TC93/WG2　Member）
客員	今井　正治	大阪大学
客員	神戸　尚志	近畿大学 （TC93　国内委員長）

(3) 2007 年度活動

EDA標準化小委員会としては、年5回の会合を行い、傘下のSystemCワーキンググループ、SystemVerilogワーキンググループ、Power Format検討ワーキング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。SysyemC/SystemVerilogの両ワーキンググループは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。具体的には、SystemCワーキンググループは、TLM2.0およびサブセットを、SystemVerilogワーキンググループは、IEEE1364(VerilogHDL)とIEEE1800 (SystemVerilog)の統合ドラフトを、国内半導体業界の意向を反映させるために夫々レビューを行った。本年活動を開始したPower Format検討ワーキンググループは、CPFとUPFの二つのPower Formatについての調査と検討を行い、両言語のインターオペラビリティの必要性を確認した。

また、SystemC ワーキンググループは、2008 年 1 月のシステム・デザイン・フォーラム 2008 で設計記述言語のユーザ・フォーラムを主催すると共に、Power Format 検討 WG は、同フォーラムにて活動成果を発表した。

2008 年 9 月の IEC/TC93 ガイザーズバーグ国際会議には、本委員会から TC93 国内委員会委員長である神戸客員、WG2 国際コ・コンベナの小島特別委員、および小委員会主査の山田の 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を行った。

Low Power Format に関し、UPF の標準化を推進している IEEE.P1801 と、同じく CPF の標準化を推進している Si2(Silicon Integration Initiative)と、夫々Power Format に関する技術討論、および標準化スケジュールについての打合せを本年度初めて実施した。

昨年度までの Accellera と OSCI に加えて、本年度は Si2 にも、2008 年 1 月のシステム・デザイン・フォーラム 2008 での後援と本フォーラムでのＳｙｓｔｅｍC, UPF, CPF に関する最新標準化状況および今後の計画などの講演を頂いた。 IEEE DASC とは、2008 年 1 月の EDSFair2008、および同 2 月の DVcon に合わせて合同会議を開催した。

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 のメンバーとして関連する 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 の正式メンバーにもなり、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。今年度も昨年に続き EDSFair2008 合わせてに IEEE DASC との合同会議を開催した（2008. 1.24）。

会議には米国から DASC 議長の Berman 氏(08 年から IEC/TC93 国際幹事も兼務)、ＤＡＳＣ幹事 Bailey 氏(IEEE.P1801 WG Chair も兼務)、Accellera Vice Chair の Denis 氏などが参加した。EDA 技術専門委員会からは、齋藤氏、太田氏、小島氏、中森氏、山田が参加した。ＴＣ９３国内委員会からは、唐津国際会議議長、神戸国内委員長にも出席を頂いた。会議では、EDA 技術専門委員会の活動状況を紹介し、この中で IEEE.P1801 に対して、Si2 による CPF(Common Power Format)との二つの標準化案に対しての懸念を改めて説明し、一本化の要求を行った。Berman 氏からは DASC の主な WG の簡単な最新情報、Denis 氏からは Accellera の最新情報の紹介が行われた。

(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

2008 年 3 月現在の DASC および SA とその傘下の Working Group と Study Group は以下のとおり。 P1801 WG が今年発足した Power Format 対応の新規 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) (IEEE1481R)
- 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)
- SystemVerilog Working Group
    - P1800 SystemVerilog: Unified Hardware Design, Specification and Verification Language (SV-IEEE1800) [cosponsored with IEEE-SA CAG]
    - P1364 Standard for Verilog Hardware Description Language (IEEEVerilog)
- P1850 Standard for PSL: Property Specification Language (IEEE-1850)
- P1685 SPIRIT XML Standard for IP Description (IEEE-1685)
- P1801 Standard for the Design & Verification of Low Power ICs

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、昨年度 100 周年を迎え、本年が 101 年目ににあたる。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。最近は、2002 年 10 月に中国･北京市、2003 年 11 月・2004 年 10 月は米国・Piscataway 市、2005 年 9 月は日本・京都（奈良）、200 6 年 9 月はドイツ・ベルリンで開催され、今年度は米国・ガイザーズバーグの NIST(National Institute of Standards and technology)で開催さ

れた。

(2) TC93 の組織と参加国

2008 年 3 月現在 IEC の Web サイト(www.iec.ch)によれば、24 カ国が TC93 のメンバーとなっている。IEC のメンバー資格には、P(Participating) と O(Observer) の二種類があるが、P メンバーは 3 カ国、O メンバーが 21 カ国である。P メンバーとしては、日本、中国、米国が登録されており、O メンバーとして、オーストラリア、ベルギー、チェコ共和国、デンマーク、エジプト、フィンランド、フランス、ドイツ、ハンガリ、インド、アイルランド、イタリア、韓国、オランダ、ロシア、セルビア、シンガポール、スペイン、スウェーデン、ウクライナ、イギリスが登録されている。幹事国は米国（国際幹事 : Victor Berman 氏）が、国際会議議長は日本(唐津氏)が担当している。

(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、および 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ー（2008 年 3 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: 山田 節(三洋) *、柴田 明一(JPCA) 、高橋 満(日立)、山下 寛巳(SML)

・ WG2:(ハードウェア設計記述言語)　主査:　山田 節（三洋）　*

国際コ・コンベナ、小島 智(NEC システムテクノロジー) *

委員:　長谷川 隆(富士通) *、浜口 加寿美(松下) *、齋藤 茂美(ソニー) *

・ 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(日立、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

(5)TC93 ガイザーズバーグ会議の報告

2007 年の国際会議は､9 月に米国メリーランド州ガイザーズバーグ､NIST(National Institute of Standards and technology)で開催された。

昨年に続き P メンバー国の参加国は、米国と日本のみであったが、O メンバー国の韓国の参加があった。日本からは米国を上回る７名が参加した。米国からは TC93 メンバー６名に加え、IEC からも１名の参加があり、デュアルロゴ関係での議論が進んだ。また韓国からは、１名の参加があった。

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。

WG２では、国際コ・コンベナの Dennis 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、Power Format に関する二つの標準化の動向に懸念を示し、一本化を強く要求した。

TC93ガイザーズバーグ会議WG2の纏め

・IEEE とのデュアルロゴでの処理案件（WG2 単独２件、WG7 共同 4 件）を確認

・新規案件候補（IEEE1666 System C）とメンテナンス案件候補（IEEE1364 Verilog-2005、IEEE1076c VHDL-2007）を選定

・IEC 登録番号の体系化の提案を決定

・２つの Power フォーマット業界標準の課題に対して問題の認識合せを行い、Unified 標準を目標にすることを決めた。

・Feeder 組織 Accellera で進めている UCIS（Unified Coverage Interoperability Standard）が、EDA ベンダー間の調整が難航している事を確認

写真１．TC93 ガイザーズバーグ会議
(会議後、NIST 内のニュートンが万有引力を発見したリンゴの木の子孫の下にて)

(6)　IEC 規格投票について

7 月に以下 7 件の IEEE 標準をデュアルロゴとして、IEC 標準とするための FDIS(Final Draft International Standard)投票が行われ、EDA 標準化小委員会（TC93/WG2）は 7 件全てに賛成を行い、デュアルロゴとして成立した。

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 SystemC ワーキンググループ報告

(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ベンダーの賛同を得て、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

上記の通り、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。

本ワーキンググループは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。

(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のメンバーとして活動を行い、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 OSCI よりリリースされた TLM(トランザクションレベルモデリング) 2.0 ドラフト１、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それぞれ４件、５７件 OSCI に伝えた。
- 2007 年度は OSCI よりリリースされた TLM 要求仕様､用語集について分析を行い、また 11 月にリリースされた TLM 2.0 ドラフト２についてレビューを行い、問題点や要望事項を 10 件 OSCI に伝えた。これらが 2008 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。

② SystemC 技術調査

- 2003 年 11 月度に集中審議を行い、本ワーキング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。
- SystemCを用いた高位合成を効果的に行うためには記述スタイルの標準化が望ましいため、スタイルガイドの骨子を検討し、「合成スタイルガイド構成要件」としてまとめ、アニュアルレポートにて公開した。
- 2007年度はSystemCを用いた推奨設計メソドロジについて検討し、合成編について審議を完了し、本アニュアルレポートにて公開した。

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。
- 2008 年 1 月 25 日に SystemC ユーザ・フォーラム 2008 を開催。今回は TLM に関するトピックを多くしたせいか、聴講者は 166 名と増加した。(前年比 12%増) また、アンケートは 144 名の方に記入いただけた。(内容等の詳細については 4.2.6 を参照)
- アンケート調査結果より、次のような事が読み取れた。
  - 今回実施した TLM 2.0 ドラフト 2 チュートリアルはかなり好評であったが、講演時間が短すぎたようで、不満を感じる人が多かった。また、Q&A の回答(英語)を通訳して欲しいという要望が 5 件もあった。
  - 動作合成サブセットと TLM の標準化を期待する人が多い。
  - OSCI や JEITA には、日本語による情報発信が期待されている。
- 2003 年～2008 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。近年は、SystemC のソフトウェア開発の利用が増加している。また、TLM と合成の標準化は依然期待が高い。

(4) 参加メンバー

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㈱
	小島　　智	NEC システムテクノロジ㈱
	清水　靖介	沖電気工業㈱
	長尾　文昭	三洋半導体㈱
	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
	西園寺 修	日本シノプシス㈱
	竹村　和祥	松下電器産業㈱
	牧野　　潔	メンターグラフィクスジャパン㈱
	渡邊　政志	㈱ルネサステクノロジ
客員	今井　正治	大阪大学
		（計 13 名）

2.5.5　SystemVerilog ワーキンググループ

(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 IEEE1364 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、IEEE1364-2001 として改訂された。JEITA（旧 EIAJ）の EDA 技術専門委員会では、IEEE1364 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 IEEE1800 として標準化された。

2008 年には、Verilog HDL の IEEE1364 と、SystemVerilog の IEEE1800 がひとつの標準言語として統合される予定である。本ワーキンググループは、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) 活動の概要

①グループ結成

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。07 年度 4 月に「SystemVerilog ワーキンググループ」に名前を変更し現在に至る。

②国際的な情報収集・標準化組織との連携(03～07 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おこなった。

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、IEEE P1800 の投票グループにはいり、05 年度の最終投票に参加、標準化推進に貢献した。

③IEEE1800-2005、IEEEp1800-2008 言語仕様検討と Issue List の提出(03～07 年度)

IEEE1800-2005 の標準化に対し、04 年 8 月に改善提案を含む 32 件の Issue List をまとめ Accellera と IEEE に提出、IEEEp1800-2008 の標準化に対し、07 年 5 月に新たに 35 件の Issue List を IEEE に提出した。

④SystemVerilog/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 共通の対訳表をひとつにまとめた。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、2006 年度版アニュアルレポートに「SyetemC/SystemVerilog 専門用語対訳表」として掲載した。

⑤SystemVerilog ユーザフォーラムの開催(04～06 年度)

05 年、06 年、07 年の 1 月に、EDS フェア併設の「システム・デザイン・フォーラム」のカリキュラムのひとつとして「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。

(4) 関連機関の動向

IEEE では、2005 年に IEEE1800 制定、次期改訂として 2006 年 6 月に 1800PAR(Project Authorization Request)が承認され、新たな改訂作業が正式にスタートした。本改訂で、IEEE Std.1364(Verilog HDL)と IEEE Std. 1800(SystemVerilog)を Std. 1800 として統合する。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無くなる。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。

2008年5月末　事前投票用のドラフト完成

2008年8月　投票

(5) 07年度の主な活動内容

○07年5月　IEEEp1800/Draft3（英語1162ページ)のレビュー

35件の指摘･要望をIssue Listとしてまとめ、IEEEp1800-WGに送付

○07年11月　Issue List 35件の対応状況を確認、CLOSE Issueの内容確認と

OPEN Issueのフォローを IEEEp1800-WGに対して行った。

①CLOSE 25件

・登録済 CLOSE issue：3件

・登録無 CLOSE Editorial issue：22件

内容の確認を行った。

②OPEN 10件

・登録済 OPEN issue：8件

7件については、IEEE のプライオリティ付けを了承、継続検討を

お願いした。残り1件は、IEEEに追加説明送付を行った。

・未登録 OPEN Editorial issue：2件

IEEEへ再度の確認を行った。

○08年 2月　上記 OPEN 10件の対応状況を確認、CLOSE Issueの内容確認と

OPEN Issueのフォローを IEEEp1800-WGに対して行った。

①CLOSE 5件

内容確認と CLOSEの了承

②OPEN 5件

・登録済 OPEN issue：3件

IEEEへ追加説明の送付と再度の要求を行った。

・登録無 OPEN　Editorial issue：2件

IEEEへ再度の確認を行った。

本ワーキンググループがIEEEp1800に提出したIssue List(35案件とその最新対応状況)を「SVTG-1　IEEEp1800-2008 Issue List」として付録に添付する。

⑹ 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	富士通マイクロエレクトロニクス㈱
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

2.2.6 PowerFormat 検討グループ報告

(1) 背景

近年、System-LSI に対する低消費電力化の要求はますます強くなってきている。 バッテリー駆動の携帯機器用のみならず、環境保護の面から家電製品やサーバー用途に至るまで全ての領域で低消費電力化はLSI 設計における最大の課題のひとつとなってきた。 一方で、性能向上の要求とプロセスの微細化によるチップあたりの回路規模の増加、さらにはリーク電流の増加が低消費電力化を困難な問題にしている。

その結果、LSI インプリ時において従来から適用されてきたクロックゲーティングやマルチ Vth といった低消費電力技術に加え、マルチ VDD・パワーゲーティング・バックゲートバイアス等の高度な技術が一般に適用される割合が飛躍的に増加している。

しかし、市販のインプリツールではこれらの技術のサポートが遅れており、設計者が既存の機能と独自のワークアラウンド等を組み合わせてなんとかしのいでいる状況が続いた。 特に問題だったのは新しい低消費電力化技術では電源を明確に意識して設計を進める必要があるのにも関わらず、電源の接続や分離を表現する統一した手法が無かったことである。 ツール毎に電源の表現方法が異なるため、フローの途中で電源記述のコンバージョンが必要となり、手間が増えたりミスの原因にもなっていた。

2006 年 6 月、CADENCE 社は 十数社の半導体ベンダー、EDA ベンダーと共に PFI(Power Forward Initiative) を組織し、電源接続やマルチ VDD,パワーゲーティングを表現可能な標準形式 CPF(Common Power Format)を策定した。 一方、SYNOPSYS と MENTOR はこの動きに対抗し、Accellera で UPF(Unified Power Format) を作り、２つの「標準」フォーマットが出来てしまった。

現在、CPF は Si2 にて Rev 1.0 が公開され、CADENCE をはじめとしたインプリツール群で 2007 年初頭からサポートされている。 UPF は Accellera にて Rev 1.0 が公開された後、IEEE に引き継がれ IEEE-P1801 として標準化作業中である。 2007 年終り頃からシノプシス、メンターのツールでのサポートも始まっている。

複数のツール間で電源表現を共通にするべく作成された両フォーマットであるが、ベンダー毎にサポートしている PF（パワーフォーマット）が異なるため、皮肉にもコンバージョンや互換性といった新たな問題を半導体ベンダーにもたらす結果となった。

(2) 目的

日本を始め多くの半導体ベンダーでは、複数の EDA ベンダーのツールを使用して LSI 設計を行っているため、２つの「標準」PF の存在は今後障害となることが予想される。 フローのある部分のツールは CPF, 別の部分のツールは UPF が必要となれば、結局フォーマットのコンバージョンが必要となり、手間が減らず互換性も問題となる。

PF を統一し、全てのツールがそれをサポートするのが理想ではあるが、既に CPF/UPF 共に仕

様が固まっており、Si2, Accellera 両陣営も統一化に対しては後ろ向きであるため、現時点では非常に難しい。 ツールへの CPU/UPF インプリメントも進んでしまっている状況である為、たとえ今すぐフォーマットが統一されたとしても全ツールがそのサポートを完了するには１～２年の時間が必要で、その間はやはり両フォーマット間のコンバージョンが必要となることが予想される。

以上の理由から、早急なフォーマットの統一は目指さず、両フォーマットのインターオペラビリティの確保を第一の目的とする。 また、2008 年に予定されている IEEE P1801 への提言、投票を行う。

(3) 活動内容

ワーキンググループの結成と PF 比較表の作成(07 年度 下期)

07 年 10 月に CPF/UPF のインターオペラビリティを確保する事を目的に組織された。
まず、現状の把握をするため、両フォーマットを実際の設計者の立場から比較し、表にまとめた。（本アニュアルレポートに添付）

設計フローを「消費電力計算」「論理検証」「論理合成」「ライブラリ」「DFT」「LowPower 検証」「P&R」「解析」の８つに分解し、それぞれの場面で多電源とパワーゲーティングの２つの想定デザインに対する CPF,UPF 記述を比較した。 （時間と工数の関係で、今回は詳細なコマンドオプションまでの検討はしていない）

結論としては、現時点では両フローとも致命的な問題や大きな相違は見られなかったが、CPF では Power Switch の ON 時間を表現できない、 UPF では Always On Buffer を扱えないなど小さな課題が明らかになった。

主な活動は以下のミーティングとメーリングリスト上にて実施した。

2007 年 10 月 11 日　第一回ミーティング　活動期間、方針の決定
2007 年 11 月 7 日　第二回ミーティング　主査、副主査決定。 比較表フォーマット検討
2007 年 11 月 29 日　第三回ミーティング　優先度１部分の検討
2007 年 12 月 19 日　第四回ミーティング　優先度２部分の検討
システムデザインフォーラム関連の検討
2008 年 1 月 17 日　第五回ミーティング　優先度３、４部分の検討
システムデザインフォーラム関連の検討
2008 年 1 月 23 日　Si2 ミーティング　Si2 ステータス確認。
JEITA PF 活動紹介。 インターオペラビリティの重要性をアピール
2008 年 1 月 24 日　IEEE/DASC ミーティング　IEEE/DASC ステータス確認。
JEITA PF 活動紹介。 インターオペラビリティの重要性をアピール
2008 年 2 月 28 日　第六回ミーティング　比較表最終確認。 2008 年度活動方針検討。

(4) 関連機関の動向

IEEE では、2008 年 3 月末に 1801 (UPF)の最終 Draft が作成され、４月に投票開始、６月に投票終了、2008 年末に規格化という予定である。

(5) 4.3 章の添付資料について

PFWG-1　PowerFormat 比較表

(6) 参加メンバ

主　　査	中 森　　勉	富士通㈱
副 主 査	南　　文 裕	㈱東芝
委　　員	安 井 卓 也	松下電器産業㈱
同	井 上 善 雄	㈱ルネサステクノロジ
同	中 村 正 昭	三洋半導体㈱
同	山 田 陽 一	セイコーエプソン㈱
同	山 縣 暢 英	ソニー㈱
同	北 原　　健	㈱東芝
客　　員	神 戸 尚 志	IEC/TC93 国内委員長　(近畿大学)
特 別 委 員	小 島　　智	IEC/TC93/WG2 ココンベナ　(NEC システムテクノロジー㈱)

３．各種イベント(主催／協賛)報告

3.1 Electronic Design and Solution Fair 2008 (EDSFair2008)

社団法人 電子情報技術産業協会（JEITA） 半導体部会主催により、2008 年 1 月 24 日（木）～25 日（金）の 2 日間、パシフィコ横浜にて、半導体に関連する設計、製造技術の専門性の高い情報を一堂に集めた展示会「Electronic Design and Solution Fair 2008 (略称:EDSFair2008)」を開催した。

当日の出展社数は、169社と過去最高となり、来場者も 非常に低温であったにもかかわらず10、431人と1万人を超え、会場内非常に盛況であった。

今回は、キャッチフレーズを 「設計を極める！ ～さらなる進化への二日間～」とし、EDA 技術者だけでなく半導体開発に携わる技術者の技術向上を図るため、EDA 技術、IP 技術、設計技術等ベンダーの展示に加え、いくつかの半導体設計者に役立つ新しい企画を実施した。

キーノートスピーチでは、神戸大の永田真准教授が 最近特に重要になっているノイズの問題について、概念から実戦まで幅広く講演し、約400名が熱心に聴講した。

展示会場内の特設ステージでは、一日目にDFMとESLのテーマで日本では初めてEDAベンダー数社の幹部クラスと日本のユーザとのパネルディスカッションを行い、開発現場が抱えている課題とEDAベンダーの解決案をぶつけて議論した。聴衆には、各 EDA ベンダーの課題解決に対する方針の違いが理解できた。二日目は、昨年に引続き若手技術者の育成を目指した「いまさら聞けないことがわかる」シリーズを、ローパワー、高位設計、DFT の三つのテーマについて、実施した。いずれのセッションも約300名の聴衆を集め 立ち見がでるほど盛況であった。

海外の新興ベンダーの技術を早く日本の技術者に伝えるため、新興ベンダーツアーを実施した。日本を代表する技術者 東芝/樋渡氏、D2S/吉田氏にツアーガイドをお願いし、各新興ベンダーの技術、製品の概要を日本語にて紹介した後、ブースに引率して各ブースにおいて説明及び質疑応答の橋渡しを行った。各ツアーとも20～30名の参加者がおり、訪問した新興ベンダーからは非常によい企画と好評であり、参加者からも新興ベンダーへアクセスするきっかけができたと大変好評であった。

3.1.1 EDSFair2008 の概要

(1) 開催期間:2008 年 1 月 24 日(木)～1 月 25 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)

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:10,431(前年 11,136 名)

② 出展社数:169 社/339 小間(前年 154 社／348 小間)

③ 出展社セミナー:101 セッション、延べ聴講者数 3,225 名

④ スイートルーム:6 社

⑤ ユニバーシティ・プラザ:21 ブース、21 大学研究室

⑥ キーノートスピーチ:聴講者数 355 名

⑦併催

第 15 回 FPGA/PLD Design Conference:8 セッション、聴講者数 515 名

・ユーザ・プレゼンテーション/IP フリーマーケット 聴講者数 27 名

⑨同時開催

システム・デザイン・フォーラム 2008:2 セッション、聴講者数 282 名

来場者業種内訳

	2008 年	2007 年	偏差
半導体・電子部品	4736	4722	14
機器メーカ	2660	2884	－224
ツールベンダー	563	590	－27
設計関連サービス	1095	1169	－74
商社・営業	428	668	－240
出版・マスコミ	115	111	3
その他	834	991	－157
計	10431	11136	－705

来場者数の減少の大きい業種は、機器メーカ、商社・営業、その他である。

機器メーカ関連者数の減少は、大手 FPGA メーカの出展中止、商社・営業関連者数の減少は、大手商社の出展中止が影響しているものと思われる。

いずれにしても、大手出展者の出展取りやめは、小間数の減少だけでなく、関係者が来場しないことによる来場者数の減少にもなってしまう。

今後の活動では、新規の出展社の獲得とともに大手ﾍﾞﾝﾀﾞｰの出展を継続させることが必要である。

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)
　①LSI 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
　②PCB 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
　③SiP 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 24 日(木)午前 9 時 45 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　住田孝之様

アメリカ合衆国大使館上席商務官　ジョンフレミング様

横浜市経済観光局長　塚原良一様

・主催者挨拶/テープカット:

社団法人　電子情報技術産業協会　専務理事　半田力

開会式終了後、登壇者および関係者による会場視察が行われ、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おこなった。

ユニバーシティ・プラザ、(株)半導体理工学研究センター、新興ベンダーエリア、メンター・グラフィックス・ジャパン(株)、JEVeC ビレッジ、(株)シルバコ・ジャパン(以上、見学順)

3.1.4 出展社一覧

アーム(株)
(株)アイヴィス
アジレント・テクノロジー(株)
アットデザインリンクス(株)
　アクセリコンテクノロジーズインク
アトレンタ(株)
Anova / CWS / ATopTech
アパッチデザインソリューションズ(株)
(株)アプリスター
アルティウムジャパン(株)
アルデック・ジャパン(株)
アンソフト・ジャパン(株)
EDA Consortium (EDAC)
E2 パブリッシング(株)
伊藤忠テクノソリューションズ(株)
　ANOVA SOLUTIONS INC.
　Blaze DFM, Inc.
　日本イヴ(株)
　REAL INTENT, INC.
　ENOVIA メイトリックスワン(株)
　EMC ジャパン(株)
　日本ヒューレット・パッカード(株)
　日本アイー・ビー・エム(株)
　マクニカネットワークス(株)
　日本ネットワーク・アプライアンス(株)
イノテック(株)
　Arteris Inc.
　Beach Solutions Inc.
　ChipVision Design Systems AG
　eASIC Corporation
　Jazz Semiconductor
　Novelics LLC
　Rapid Bridge LLC
　Target Compiler Technologies N.V.
(株)エーイーティー
ATE サービス(株)
特定非営利活動法人 FPGA コンソーシアム
(株)沖ネットワークエルエスアイ
カーボン・デザイン・システムズ・ジャパン(株)
(株)ガイア・システム・ソリューション
兼松エレクトロニクス(株)/プラットフォームコンピューティング(株)
カリプト・デザイン・システムズ(株)
(株)キー・ブリッジ
(株)クレディスト
コーウェア(株)
CoFluent Design (コ・フルーエントデザイン)
CyberTec(株)
　ジャスパー・デザイン・オートメーション
サイバネットシステム(株)
サガンテック・ノース・アメリカ・インク
札幌市役所
CQ 出版(株)
シーケンスデザイン(株)
(株)シルバコ・ジャパン
シンプリシティ(株)
(株)図研/NEC システムテクノロジー(株)
StarNet Communications
(株)スピナカー・システムズ
　Verific Design Automation, Inc.
ソニックス
(有)ソネット技研
　エイ・ダブリュー・アール ジャパン(株)
ダイキン工業(株)
タナーリサーチジャパン(株)
Chipworks Inc.
ティーモステクノロジック(株)
　サン・マイクロシステムズ株式会社
　日本アイ・ビー・エム株式会社
　富士通株式会社
DSM ソリューションズ(株)
デナリソフトウエア(株)
(株)電波新聞社
日経 BP 社

日本イヴ(株)
日本ケイデンス・デザイン・システムズ社
　イノテック(株)
日本シノプシス(株)
日本セロックシカ(株)
日本ノーベル(株)
　　エース・アソシエイテッド・コンパイラ・エキスパーツ(株)
ノバフロー(株)
　　Novas Software, Inc
　SpringSoft, Inc.
　　ForteLink, Inc.
　　Silicon Canvas, Inc.
BERKELEY DESIGN AUTOMATION, INC.
パルシックジャパンリミテッド
(株)半導体理工学研究センター
Handshake Solutions
　IBM Service Company Japan
(株)日立超 LSI システムズ
ファラッド(株)
フォルテ・デザイン・システムズ(株)
(財)福岡県産業・科学技術振興財団
ブライオンテクノロジーズ(株)
(株)プライムゲート
Blaze DFM, Inc.
　Itochu Techno-Solutions Corporation
PROGATE GROUP CORPORATION
プロトタイピング・ジャパン(株)
　ProDesign Electronic GmbH
　Temento Systems S.A
　CAST Inc.
　SystemCrafter Ltd.
HELIC SA.
ポーラー・インストルメンツ日本支社
マグマ・デザイン・オートメーション(株)
マクロヴィジョン(株)
三菱電機エンジニアリング(株)
三菱電機マイコン機器ソフトウエア(株)
メンター・グラフィックス・ジャパン(株)
リード・ビジネス・インフォメーション(株)
(株)ロイノス
OneSpin Solutions

3.1.5 出展傾向

EDSFair2008 は、出展社数が過去最高となり、169 社(新規 29 社)となった。

	出展社数	小間数
2008 年	169 社	339 小間
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

	2008 年		2007 年	
	社数	小間数	社数	小間数
通常出展	115 社	294 小間	130 社	324 小間
新興ﾍﾞﾝﾀﾞｰ(JeVEC 対象含む)	54 社	45 小間	24 社	24 小間

出展社数が過去最高であったにもかかわらず、小間数が減少したのは、通常枠で複数小間出展していた大手 FPGA ベンダー、大手商社の撤退、及び 共同出展のベンダーが増加したことによると思われる。

3.1.6 出展者セミナー

1 セッション 45 分間で、30～100 名の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。2008 年は 9 会場にて 101 セッションを開催した。

聴講者数:3,225 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共に、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。

21 大学研究室

- **VLSI における新しい故障モデルに基づく故障検査法の開発**
  愛媛大学 理工学研究科 電子情報工学専攻 高松研究室
- **高機能画像処理システムと次世代タイミング設計技術に関する研究開発**
  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
- **IP 間のデータ転送解析による最適 SoC 設計手法**
  大阪大学大学院 情報科学研究科 今井研究室
- **社会基盤を支える次世代システム LSI の設計手法とその支援技術**
  九州大学 システム情報科学研究院 安浦・松永・吉村研究室
- **動的再構成可能 ASIP 技術「Redefis」を核とする SoC プラットフォーム**
  九州大学 システム情報科学研究院 情報理学部門 村上・井上・馬場研究室
- **高性能・低消費電力システム LSI のためのマルチコア SoC システムアーキテクチャと設計教育**

九州大学 システム LSI 研究センター

■ **パターンマッチング回路の合成と応用について**

九州工業大学 情報工学部 電子情報工学科 笹尾研究室

■ **システム LSI の低消費電力テスト技術**

九州工業大学 情報工学部 電子情報工学科 梶原・温研究室

■ **やわらかいハードウェアとリコンフィギャラブルシステム技術**

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 天野研究室

■ **SoC のインテグリティ:オンチップ・モニタとノイズ解析技術**

神戸大学 大学院工学研究科 情報知能学専攻 永田研究室

■ **ユビキタス応用低消費電力システム LSI**

神戸大学 工学研究科 情報知能学専攻 吉本研究室

■ **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 藤田研究室

■ **動的な仮想回路による HW/SW 複合システム**

東京農工大学 工学部電気電子工学科 関根研究室

■ **システムレベル設計に関する研究報告 ～効率的な設計探索と検証～**

名古屋大学 大学院情報科学研究科 情報システム学専攻 高田・冨山研究室

■ **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 渡邉研究室

■ **結線論理マイクロコードマシンとエネルギー効率**

広島市立大学 情報科学部 高橋研究室

■ **メモリを活用した回路実現について**

明治大学 理工学部 電子情報工学科 井口研究室

■ **超高信頼アナログ LSI 設計環境の構築**

九州工業大学 マイクロ化総合技術センター 中村研究室

■ **組み込み技術者向け教材開発と UML を用いたハードウェア設計プロジェクト**

東海大学 情報理工学部 ソフトウェア開発工学科 清水研究室

3.1.8 キーノートスピーチ

LSI を高性能化する設計技術の挑戦

– インテグリティを指向する設計へ

神戸大学大学院 工学研究科 情報知能学専攻

永田 真 准教授

(1) 日　時: 1 月 24 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:355 名

デジタル・システムの高速・低消費電力化に向けて、低電圧動作、細粒度な電源ドメイン管理、動的な電圧・周波数制御の導入が進むが、その実装にはダイナミック電源ノイズの考慮が欠かせない。高精度・低電圧なアナログ IP の開発においては、基板ノイズにさらされる SoC 環境下で性能を発揮する設計が求められる。LSI 搭載システムの EMC 性能を高めるためには、チップとボードで連携した低ノイズ化設計が必須になる。

本講演では、「LSI におけるノイズ問題」を通して、LSI 設計を良くするためのノイズの知識獲得、およびインテグリティを指向する設計技術の現状とチャレンジについて述べた。

3.1.9 第 15 回 FPGA/PLD Design Conference

(1) 日　時:1 月 24 日(木)・25 日(金)

(2) 場　所:アネックスホール

(3) 聴講料:事前申込:10,500 円(1 日券)※消費税込

当日申込:12,600 円(1 日券)※消費税込

聴講者数:

セッション 1　66 名

セッション 2　59 名

セッション 3　62 名

セッション 4　55 名

セッション 5　72 名

セッション 6　59 名

セッション 7　76 名

セッション 8　66 名

EDSFair と併設の第 15 回 FPGA/PLD Design Conference が 1 月 24 日(木)と 25 日(金)の両日、アネックスホールにて開催され、全 8 セッションで合計 515 名の聴講者を集めた。

今回で第 15 回を迎えたこのコンファレンスは、FPGA/PLD をテーマとした日本で随一のコンファレンスであり、FPGA/PLD に関する最新情報、設計手法、ビジネスおよ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 FPGA を用いるシステム設計者に有用であり、FPGA を活用し、より高い付加価値を創出するための、設計手法、実装技術、ホットトピックスなどを密度濃く選定した。

これに加え、ユーザ・プレゼンテーションおよび各種 IP の発表の場を提供する IP フリーマーケットも開催した。

コンファレンス実施にあたり、実行委員会は、従来の2トラック・セッションごとのチケット販売を改め、1 トラック・1 日券の販売に変更した。また、1 トラックでの体裁を整えるため、実施時間帯を調整し、1 日 4 セッションを詰め込むこととした。セッションあたりの平均聴講者数は、64 名であり、2007 年の 42 名、2006 年の 33 名と比較し、大幅増を達成できた。また、アネックスホールの定員 100 人の部屋での実施において、聴講者は席を探さないと見つからない状況であり、盛況感を出すことができたと考えている。

セッションの実施アンケート結果より、満足度が 5 割を超えるセッションが 8 セッション中 6 セッションと大部分を占め、単に動因増にとどまらず、聴講者に対して有益な情報発信となった。

1 月 24 日(木)

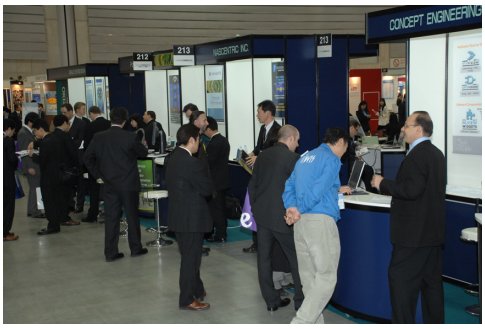
セッション 1(10:00～11:30) 機能満載のFPGAを初心・初級者はどのように設計するか 松本 仁 氏 [三菱電機(株) コミュニケーションネットワーク製作所　光トランスポート部　光技術 G　専任]
セッション 2 (11:45～13:15) 設計効率および、品質向上のためのフィジカルシンセシスの活用 吉谷 裕二 氏 [富士通九州ネットワークテクノロジーズ(株)　第一開発統括部　ユビキタス研究部]
セッション 3(13:45～15:15) 多発する仕様変更と切迫するローパワー要求への対応 西川 智洋 氏[ヒロコン(株)] 三上 廉司 氏[ミカミ設計コンサルティング]
セッション 4 (15:30～17:00) 基板設計(構造)で FPGA を動かすノウハウ！ 尾松 智裕 氏 [シャープ(株)　技術本部　プラットフォーム開発センター　モバイルプラットフォーム開発室　主任研究員]

1 月 25 日(金)

セッション 5(10:00～11:30) FPGA に搭載されるプロセッサの実性能とその活用方法 実吉 智裕 氏 [(株)アットマークテクノ　代表取締役]
セッション 6(11:45～13:15) 最新の FPGA プロトタイプボードを用いた設計検証 星野 民夫 氏 [(株)アプリスター] Mr.Mike Dini [The Dini Group] 藤永 康博 氏 [アルデック・ジャパン(株)]
セッション 7(13:45～15:15) C 言語による FPGA 設計の有効性とハード・ソフト協調設計の進め方 鳥海 佳孝 氏 [設計アナリスト]
セッション 8(15:30～17:00) FPGA の部分再構成を活用したシステムの現状と開発例 堀 洋平 氏 [独立行政法人 産業技術総合研究所　情報技術研究部門　実時間組込システム研究班]

3.1.10 新興ベンダエリア

国内外新興企業 34 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ー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。



出展者:

AXIOM DESIGN AUTOMATION
AZURO, INC.
Incentia Design Systems Inc.
EDXACT SA
ENTASYS DESIGN, INC.
KILOPASS TECHNOLOGY INC.
CONCEPT ENGINEERING GMBH
SIDENSE CORPORATION
Silterra Malaysia SDN BHD
GiDEL, LTD
　立野電脳(株)
JAVELIN DESIGN AUTOMATION
シンテスト・ジャパン
SOFTJIN TECHNOLOGIES PVT. LIMITED
Solido Design Automation
DASSAULT SYSTEMES (ENOVIA)
ChipVision AG
　INNOTECH CORPORATION
Terasic Technologies,Inc.
　立野電脳(株)
NASCENTRIC INC.
Nangate Inc.
BEACH SOLUTIONS INC
PHYSWARE,INC
Ponte Solutions Inc.
　Cubic Micro Inc.
MICROLOGIC DESIGN AUTOMATION
マセマテック(株)
MunEDA GmbH
メガシス(株)

【JEVeC ビレッジ】

(株)アストロン
エイシップ・ソリューションズ(株)
(株)エッチ・ディー・ラボ
ギガヘルツテクノロジー(株)
ケイレックス・テクノロジー(株)
(株)ジーダット
　Takumi Technology Corporation
　(株)A－ソリューション
　Avertec
　大日本印刷(株)
　Daou Xilicon Technology Co., Ltd
　FishTail Automation, Inc.
　Legend Design Technology, Inc.
　(株)数理システム
　Orora Design Technologies, Inc.
　Predictions Software Ltd.
　Pulsic Ltd.
(株)システム・ジェイディー
(株)数理システム
TOOL(株)
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おこなった。

◆1 月 24 日

1.Improve Design Performance and Yield
(113)MunEDA GmbH
Mr. Andreas Ripp
Sales & Marketing, Vice President

2.Verification Closure
(106)AXIOM DESIGN AUTOMATION
Mr. Ravi Shankar
Managing Director of AXIOM Indian Operations

3.Clock-Tree Synthesis for Nanometer Design
(219)AZURO, INC.
Mr. Marc Swinnen
Director of Product Marketing

4.Variation Robustness for Analog/Mixed-Signal, Custom Digital and Memory Design
(116)Solido Design Automation
Mr. Patrick Drennan, Ph.D.
Chief Technology Officer

◆1 月 25 日

5.Next-Generation Multithreaded Mixed A/D Fast-SPICE
(213)NASCENTRIC INC.
Mr. Dino Caporossi
Marketing, VP Marketing

6.Planning for Success: System Physical Prototyping, Solving implementation issues early
(104)JAVELIN DESIGN AUTOMATION
Mr.Matthew Raggett
Chairman & Cofounder

7.Terasic の FPGA 製品について
(115)Terasic Technologies,Inc.

8.System-level power simulation, analysis and modeling
(108)ChipVision AG
Mr. Tom West
Application Engineer

9.新製品発表～世界最小の演算回路 IP
(210)マセマテック(株)
渡 雅男氏
代表取締役社長

10.ESL-IP/SOC register design environment for reducing design iteration due to spec miscommunication
(214)BEACH SOLUTIONS INC
Mr. Simon Rance
Director of Applications and Services

11.Stratix III PCI Express x8 ボードの詳細
(114)GiDEL,LTD.

12.Addressing the Impact of Process Variability on Your Design
(211)Ponte Solutions Inc.
Mr. Richard Nakajima
President of Cubic Micro Japan (Japan Representative Company for Ponte Solutions.)

13.Rapid electromagnetic simulation for advanced electronic packaging
(220)PHYSWARE,INC
Mr. Vikram Jandhyala
CEO and Founder, Physware, Inc. & Associate Professor, Department of Electrical Engineering and Director, Applied Computational Engineering Lab, University of Washington, Seattle

14.Micrologic's EDA Technology
(218)MICROLOGIC DESIGN AUTOMATION
Mr. Shlomo Corem
Sales and Marketing - Regional Sales Director

海外新興ベンダ出展者向けセミナー

会期前日の 1 月 23 日(水)18 時より、コーウェア(株)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。

3.1.11 新興ベンダー・ガイド・ツアー

今年初めての試みとして、新興ベンダー・エリアに出展した海外ベンダーの紹介と、ベンダー・ブースをツアーで訪問する、「新興ベンダー・ガイド・ツアー」を実施した。

新興ベンダー・ツアー参加企業：19社
内訳：米国企業：9社、独企業：3社、カナダ企業：2社、英国企業：1
インド企業：1社、韓国企業：1社、仏企業：1社、台湾企業：1社
実施回数：1月24日 2 回、1月25日2回
ツアー参加人数：各ツアーとも 20－30 名

EDSFair では、2006 年より会場内に新興ベンダー・エリアを設け、国内外の新興企業が EDSFair に出展し、来場者に独自の技術を紹介できるよう、パッケージ・ブースとビレッジ化したエリアを提供してきた。しかし、言葉の壁により来場者と海外新興ベンダーとのコミュニケーションが難しかったり、また、新興ベンダーがまだ十分な展示ができないため、どのようなテクノロジを提供しているのかが分かりにくかったりと、必ずしも双方にとって満足のいくものではなかった。

EDSFair2008 では、ツアー方式のブース訪問を企画。ツアー・ガイドが引率して海外新興ベンダーのブースを訪問、コミュニケーションのサポートを日本語で行うことにより、来場者が新興ベンダーの技術を理解しやすいように支援を行った。

新興ベンダー・ガイド・ツアーでは、二人の設計技術のエキスパートがツアー・ガイドを努めた。1 月 24 日は、東芝のシステム LSI 設計技術部長の樋渡有氏、25 日は、米国 D2S, Inc の日本代表の吉田憲司氏である。二人とも、LSI 設計技術や EDA に精通し、講演経験も豊富なベテランである。

新興ベンダー・ガイド・ツアーは、まずブース訪問に先立って特設ステージにおいて、ツアー・ガイドによりツアーで訪問する海外新興ベンダーの企業紹介が行われた。各ベンダーの代表者もステージ上で紹介され、ベンダーがどのような会社か、特徴とする製品や技術は何かを事前に理解してもらった。 企業紹介のスライドも日本語で準備されていたので、参加者にとっては、ブース訪問前にベンダーについてある程度の知識を得ることができた。 ステージでの企業紹介が一通り終わると、新興ベンダー・ガイド・ツアーの旗のもと、ツアー・ガイドを先頭に紹介されたベンダーのブースを訪問した。

各ブースでは、最初にそのベンダーの代表がパネルや、PC を使って製品や技術をさらに紹介。中には、日本語デモや、パンフレットなどの資料を準備していた企業もあり、出展者としてもこのツアーによるブース来場者を期待していた様子が見受けられた。 ツアー・ガイドは、必要に応じてベンダーの説明を日本語に訳したり、また Q&A では参加者が聞きたいであろう質問を代わって行ったり、日本語でなされた質問を英語で伝えたり、また、回答を日本語で伝えたりと、ツアー参加者の技術の理解のためのコミュニーケションのサポートを行っていた。
各ツアーは、大まかに設計分野ごとでまとめられ、1 日 2 回、1 回につき 5 社を紹介、ブース訪問した。全体で 1 時間程度のツアーであったが、各ツアーとも毎回 20－30 名の参加者があり、盛況であった。

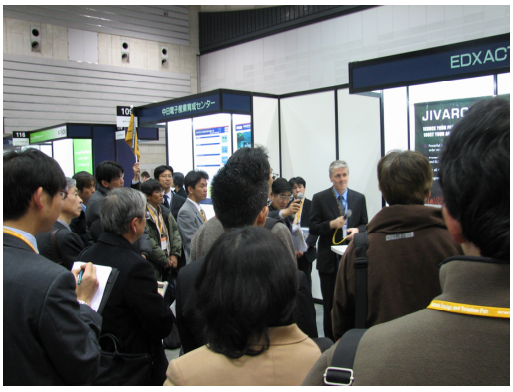
EDSFair2008 の新企画、新興ベンダー・ガイド・ツアーは、ツアー・ガイドによる参加者へのサポートによって、従来以上に海外新興ベンダーの製品や技術が、来場者に身近に感じられたと思う。また、出展した新興ベンダー側

からすると、一度にこれほどの来場者がブースを訪問したのは初めてという結果になり、出展者側からも歓迎されていた。EDSFair2008 への出展申込が遅れ、このツアーに参加できなかったベンダーからは、会期中にツアーの数を増やしてもらえないかとの声も上がっていた。

今までで、なかなか訪問し辛かった海外の新興ベンダーのブースも、このようなツアーでは安心して参加でき、新興ベンダーへのアクセスのきっかけを作ることができた。今回の新興ベンダー・ガイド・ツアーの評価を踏まえ、来年度も引き続きこのツアーを実施していきたい。

EDSF2009に向けての活動と検討事項

- EDSF 後に行われた来場者向けアンケート、新興ベンダー・ツアーに参加した企業向けアンケートを事務局より入手後、ツアー・ガイドからの改善提案、および EDSF 実行委員会での反省事項をまとめて改善案を作成、実行委員会に提出予定。
- EDSF2009の実行委員会にて、新興ベンダー・ツアー実施の検討がされ、実施が継続されることが決定された時点でその改善案を委員会にて検討。
- Nikkei Electronics Asia 4 月号に、日経マイクロデバイス小島編集委員が新興ベンダー・ツアーの記事を掲載予定。 掲載後記事の転用許可の交渉。
- DAC での新興ベンダー出展誘致の材料として、EDSF2008における新興ベンダー・ツアーの紹介資料の作成。
- EDSF2009 における新興ベンダー・ツアーの実施に向けて企画書、実施要領の作成。



3.1.12 特設ステージ

場　所:展示フロア内　参加無料

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。

1 月 24 日(木)

■EDA ベンダーエグゼクティブが語る！

□13:00～14:30 セッション 1 日英同時通訳付き

「DFM の真実 ― 今できること、これからやるべきこと」

Dr. Antun Domic［Synopsys, Inc., Sr. VP and GM, Implementation Group］
Mr. Joe Sawicki［Mentor Graphics Corp., VP, Design-to-Silicon Division］
Mr. John Lee［MAGMA DESIGN AUTOMATION, Inc., GM, Physical Verification Business Unit］
Mr. Nitin Deo［Cadence Design Systems, Inc., Group Director, DFM Marketing］
南 文裕 氏［東芝マイクロエレクトロニクス㈱ 設計自動化技術開発部］
【司 会】 小島 郁太郎 氏［㈱日経 BP 社］

聴講者数約 260 名

まだまだホットな話題である DFM に関して、EDA ツール大手ベンダのエグゼクティブ4名と設計ユーザ代表1名を招いて、設計者の視点での主要な問題(製造性困難・ばらつき増大)を挙げた上で、EDA や設計環境の現状ならびに将来動向について、解説と討論をした。

□15:45～17:15 セッション 2 日英同時通訳付き

「RTL から ESL へのパラダイム・シフトは本当に起こるのか？～誰が，起こすのか？
そして，それはいつ？ ～」

Mr. Eshel Haritan［CoWare, Inc., VP, Engineering］
Mr. Joachim Kunkel［Synopsys, Inc., VP and GM, Solutions Group］
Mr. Simon Bloch［Mentor Graphics Corp., GM, Design Creation and Synthesis Division］
Mr. Steven Wang［Cadence Design Systems, Inc., VP & GM, Incubation(ESL)］
古山 透 氏［東芝 セミコンダクター社 半導体研究開発センター］
柿本 勝 氏［ソニー㈱ 半導体事業本部 設計基盤技術部門］
【司 会】 中山 俊一 氏［CQ 出版㈱］

聴講者数約 240 名

ESLツール大手4社のエグゼクティブと日本側ユーザー代表2名を招いて、現状の各ベンダーのツール実績やユーザーの活用状況の紹介を行った。また今後のESL設計の必要性と普及の可能性について議論した。

1 月 25 日(金)

■今さら聞けないことがわかる！リターンズ

□10:30～11:30 セッション 3

「今さら聞けないローパワー～教えます。現場で使えるローパワー設計～」

石原 亨 准教授［九州大学システムLSI研究センター 設計技術研究部門］
武内 良祐 氏［三菱電機㈱ モバイルターミナル製作所 基盤技術部］
北原 健 氏［東芝 セミコンダクター社 システム LSI 設計技術部 設計メソドロジー技術開発担当］

【司　会】井上 善雄 氏[㈱ルネサステクノロジ 製品技術本部
設計技術統括部 DFM・ディジタル EDA 技術開発部]

聴講者数約 330 名

研究、設計、アプリのそれぞれの立場からローパワー設計の現状と問題点を提起、これに基づき議論を行った。　現場でおきている身近な問題点を多く取り上げ、設計者の目線での現実的な議論を展開した。

□14:00～15:00 セッション 4

「早わかりシステム設計 ―今さら聞けない 5 つのポイント」
近藤　洋 氏［㈱エッチ・ディー・ラボ テクニカルグループ］
【聞き手】牛島 真由美 氏［㈱エッチ・ディー・ラボ テクニカルグループ］
【司　会】柏木 治久 氏［㈱半導体理工学研究センター 開発第2部高位設計開発室］

聴講者数約 290 名

知っているようで実はよく知らない TLM(トランザクション・レベル・モデル)。システム設計の必須モデルである TLM について、"抽象モデル"や"通信と計算の分離"といった特徴を、従来の RTL 設計と比較しながら初心者にもわかりやすく解説した。

□16:15～17:15 セッション 5

「今さら聞けないDFT―テストクライシスって本当？ 今再び注目されるスキャン設計―」
岩崎 一彦 教授［首都大学東京 システムデザイン学部 情報通信システム工学コース］
【聞き手】岩本 尚美 氏［富士通㈱ 電子デバイス事業本部教育部］
【司　会】相京 隆 氏［㈱半導体理工学研究センター 開発第2部テスト＆故障解析開発室］

聴講者数約 280 名

テストクライシスが現実になろうとしているLIテストでは、DFT による様々な取り組みが行われている。その中で、枯れた技術と思われていたスキャン設計の重要性が浮かび上がって来たことを紹介した。本セッションでは、スキャン設計技術を軸にして、DFT の過去・現在・未来を分かりやすく解説した。

3.1.13 来場者数詳細

2008 年来場者数

1 月 24 日(木)	晴れ	4,604
1 月 25 日(金)	晴れ	5,827
合　　計		10,431

過去の来場者数

開催年	1日目	2日目	合 計
2007 年	4,956 名	6,180 名	11,136 名
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

3.1.14 全来場者入場登録票アンケート回答 集計結果

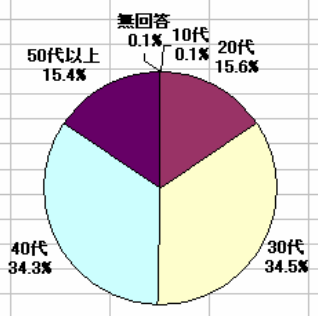
入場登録票アンケートによる来場者プロファイルを以下に示す。

Electronic Design and Solution Fair 2008 来場者アンケート集計結果

入場登録票アンケート(属性他)

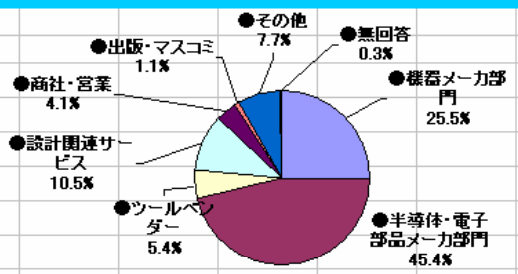
■年代

	2008(今回)	2007(参考)
10代	0.1%	0.1%
20代	15.6%	16.3%
30代	34.5%	34.4%
40代	34.3%	31.2%
50代以上	15.4%	15.0%
無回答	0.1%	3.0%
合計	100.0%	100.0%



■業種

	2008(今回)	2007(参考)
●機器メーカ力部門	25.5%	25.9%
●半導体・電子部品メーカ力部門	45.4%	42.5%
●ツールベンダー	5.4%	5.3%
●設計関連サービス	10.5%	10.5%
●商社・営業	4.1%	6.0%
●出版・マスコミ	1.1%	1.0%
●その他	7.7%	5.8%
●無回答	0.3%	3.1%

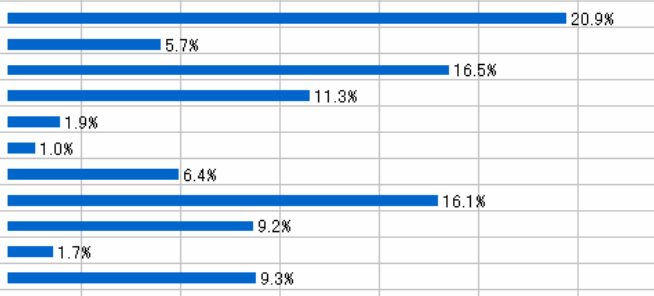


EDSFair2008
来場者 業種

※業種詳細

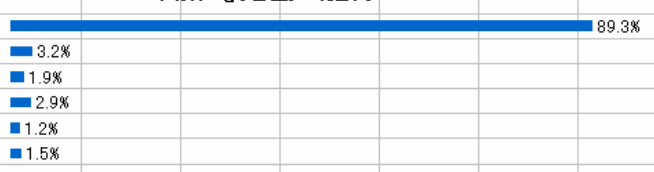
	2008(今回)	2007(参考)
●機器メーカ力部門	25.5%	25.9%
コンピュータ関連機器	20.9%	19.5%
ネットワーク関連機器	5.7%	6.3%
一般民生機器	16.5%	17.2%
画像処理機器	11.3%	9.4%
医療機器	1.9%	1.7%
アミューズメント	1.0%	1.4%
自動車・輸送機器	6.4%	4.3%
産業機器(機械・精密機器等)	16.1%	15.7%
通信機器	9.2%	10.6%
放送機器	1.7%	1.5%
その他	9.3%	12.5%

EDSFair2008来場者業種
機器メーカ力部門



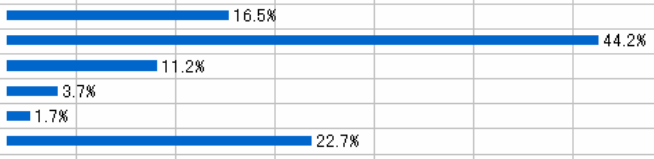
	2008(今回)	2007(参考)
●半導体・電子部品メーカ力部門	45.4%	42.5%
システムLSI、ASIC、マイコン、メモリ	89.3%	84.3%
FPGA/PLD	3.2%	4.8%
ディスプレイ	1.9%	1.9%
電子コンポーネント	2.9%	2.5%
プリント基板	1.2%	1.1%
その他	1.5%	5.4%

EDSFair2008来場者業種
半導体・電子部品メーカ力部門



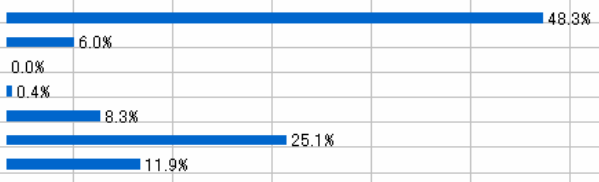
	2008(今回)	2007(参考)
●ツールベンダー	5.4%	5.3%
※ツール関連が主要営業品目である商社・代理店も含む		
機能・論理設計ツール	16.5%	21.7%
LSI設計ツール	44.2%	43.8%
プリント基板設計ツール	11.2%	12.8%
マイコンツール	3.7%	5.4%
ハードウェア・ボード機器	1.7%	2.7%
その他	22.7%	13.6%

EDSFair2008来場者業種
ツールベンダー



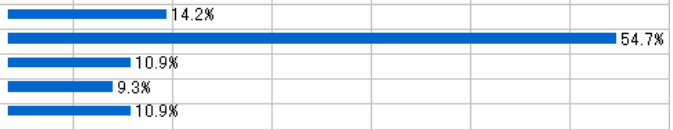
	2008(今回)	2007(参考)
●設計関連サービス	10.5%	10.5%
デザインハウス	48.3%	41.5%
IPプロバイダ	6.0%	5.8%
IP流通サービス	0.0%	1.2%
ネット環境	0.4%	1.0%
教育・コンサルタント	8.3%	11.3%
ソフト開発	25.1%	24.8%
その他	11.9%	14.4%

EDSFair2008来場者業種
設計関連サービス



	2008(今回)	2007(参考)
●商社・営業	4.1%	6.0%
※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	14.2%	14.1%
半導体	54.7%	58.1%
電子部品	10.9%	11.0%
ツール	9.3%	6.2%
その他	10.9%	10.6%

EDSFair2008来場者業種
商社・営業



	2008(今回)	2007(参考)
●出版・マスコミ	1.1%	0.9%

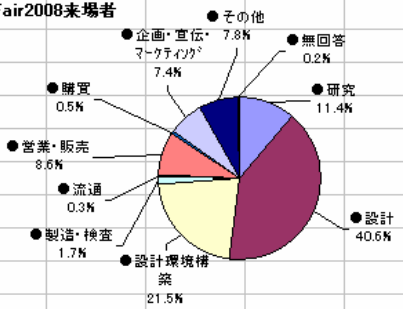
	2008(今回)	2007(参考)
●その他	7.7%	5.8%

	2008(今回)	2007(参考)
●無回答	0.3%	3.1%

■職務

	2008(今回)	2007(参考)
●研究	11.4%	13.1%
●設計	40.6%	36.4%
●設計環境構築	21.5%	19.6%
●製造・検査	1.7%	1.9%
●流通	0.3%	0.5%
●営業・販売	8.6%	9.9%
●購買	0.5%	0.3%
●企画・宣伝・ﾏｰｹﾃｨﾝｸﾞ	7.4%	7.2%
●その他	7.8%	8.0%
●無回答	0.2%	3.1%

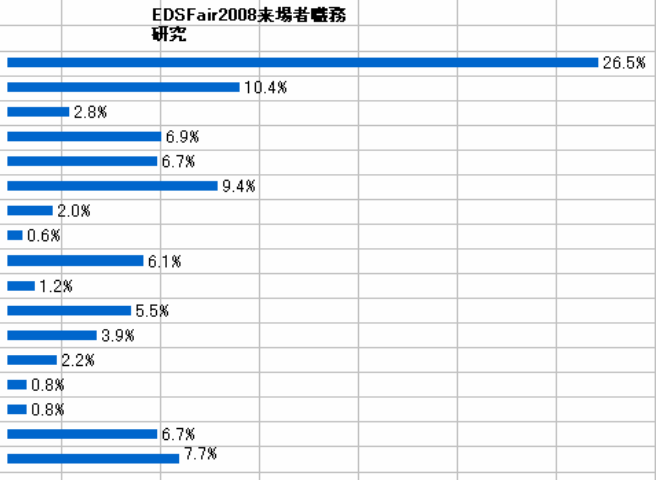
EDSFair2008来場者
職務



※職務詳細

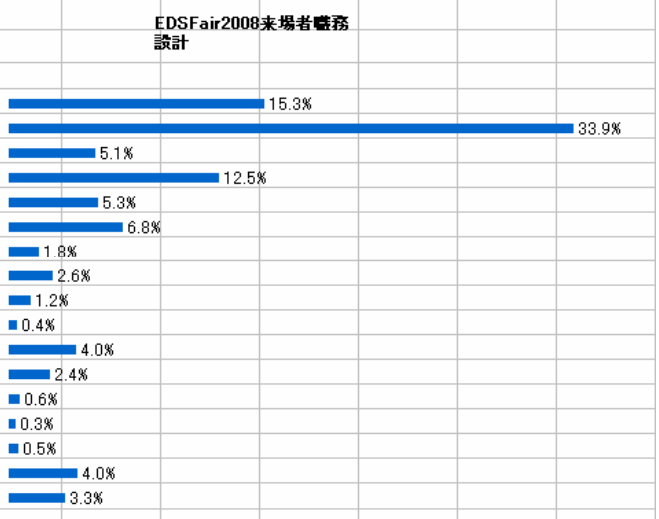
●研究

	2008(今回)	2007(参考)
●研究	**11.4%**	13.1%
システムレベル	26.5%	27.7%
機能(RTL)	10.4%	10.5%
論理(ゲートレベル)	2.8%	3.0%
レイアウト	6.9%	7.2%
テスト	6.7%	3.9%
アナログ	9.4%	9.1%
カスタム	2.0%	2.2%
IPマクロ	0.6%	1.3%
リソ/マスク/プロセス/製造	6.1%	6.6%
TCAD	1.2%	1.6%
FPGA/PLD	5.5%	5.8%
PCB	3.9%	2.3%
IC Package	2.2%	1.4%
SiP	0.8%	1.1%
装置実装	0.8%	1.0%
ソフトウェア・ファームウェア	6.7%	6.6%
無回答	7.7%	9.0%



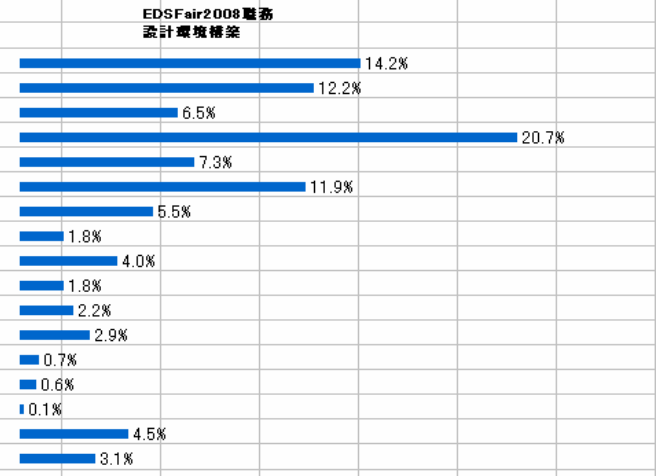
●設計

	2008(今回)	2007(参考)
●設計	**40.6%**	36.4%
システムレベル	15.3%	16.6%
機能(RTL)	33.9%	31.1%
論理(ゲートレベル)	5.1%	5.0%
レイアウト	12.5%	12.5%
テスト	5.3%	4.3%
アナログ	6.8%	7.0%
カスタム	1.8%	2.1%
IPマクロ	2.6%	2.9%
リソ/マスク/プロセス/製造	1.2%	0.8%
TCAD	0.4%	0.3%
FPGA/PLD	4.0%	4.6%
PCB	2.4%	2.4%
IC Package	0.6%	0.3%
SiP	0.3%	0.1%
装置実装	0.5%	0.7%
ソフトウェア・ファームウェア	4.0%	4.6%
無回答	3.3%	4.8%



●設計環境構築

	2008(今回)	2007(参考)
●設計環境構築	**21.5%**	19.6%
システムレベル	14.2%	13.9%
機能(RTL)	12.2%	14.3%
論理(ゲートレベル)	6.5%	6.9%
レイアウト	20.7%	21.1%
テスト	7.3%	6.3%
アナログ	11.9%	9.6%
カスタム	5.5%	4.9%
IPマクロ	1.8%	2.0%
リソ/マスク/プロセス/製造	4.0%	4.4%
TCAD	1.8%	1.4%
FPGA/PLD	2.2%	2.0%
PCB	2.9%	2.4%
IC Package	0.7%	0.4%
SiP	0.6%	0.7%
装置実装	0.1%	0.4%
ソフトウェア・ファームウェア	4.5%	5.5%
無回答	3.1%	3.8%

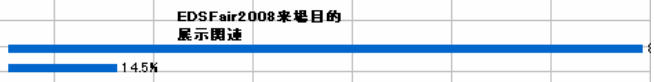


■ご来場の目的

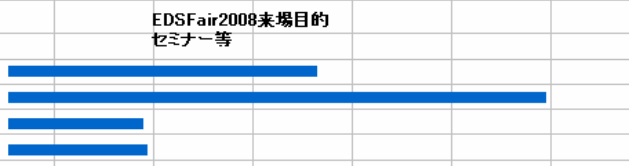
	2008(今回)	2007(参考)
●展示関連	43.0%	45.4%
●特別展示	24.8%	21.2%
●セミナー／カンファレンス	30.8%	29.5%
●無回答	1.4%	3.9%

※ご来場の目的詳細

	2008(今回)	2007(参考)
●展示関連	43.0%	45.4%
展示ブース	85.5%	83.8%
スイートデモ	14.5%	16.2%



	2008(今回)	2007(参考)
●特別展示	24.8%	21.2%
新興ベンダ・エリア	27.5%	28.9%
EDSFair特設ステージ	48.1%	42.6%
ユニバーシティ・プラザ	12.0%	12.5%
IPフリーマーケット/ユーザ・プレゼンテー	12.4%	16.0%

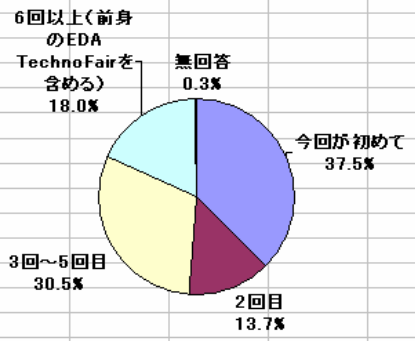


	2008(今回)	2007(参考)
●セミナー／カンファレンス	30.8%	29.5%
出展者セミナー	54.8%	52.0%
キーノートスピーチ	20.8%	20.8%
FPGA/PLD Design Conference	11.1%	13.0%
システム・デザイン・フォーラム	13.3%	14.2%

	2008(今回)	2007(参考)
●無回答	1.4%	3.9%

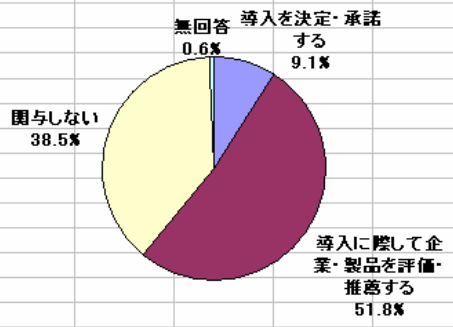
■来場頻度を教えてください

	2008(今回)	2007(参考)
今回が初めて	37.5%	37.8%
2回目	13.7%	13.9%
3回～5回目	30.5%	29.0%
6回以上（前身のEDA TechnoFairを含める）	18.0%	15.9%
無回答	0.3%	3.5%
合計	100.0%	100.0%



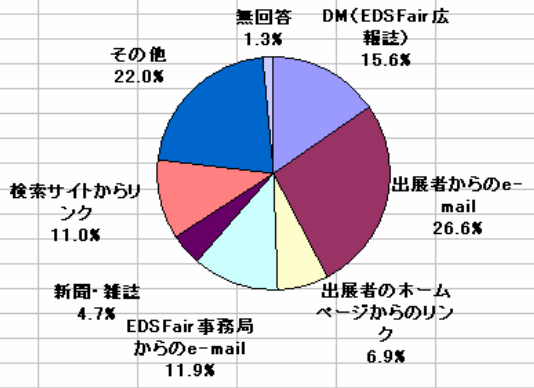
■あなたの製品導入権限について教えてください

	2008(今回)	2007(参考)
導入を決定・承諾する	9.1%	8.9%
導入に際して企業・製品を評価・推薦する	51.8%	49.1%
関与しない	38.5%	37.9%
無回答	0.6%	4.1%
合計	100.0%	100.0%



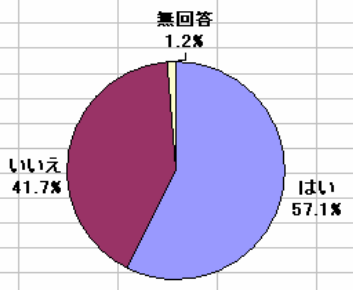
■Electronic Design and Solution Fairをどちらでお知りになりましたか？

	2008(今回)	2007(参考)
DM(EDSFair広報誌)	15.6%	13.2%
出展者からのe-mail	26.6%	25.1%
出展者のホームページからのリンク	6.9%	7.5%
EDSFair事務局からのe-mail	11.9%	10.7%
新聞・雑誌	4.7%	4.8%
検索サイトからリンク	11.0%	11.0%
その他	22.0%	22.9%
無回答	1.3%	4.9%
合計	100.0%	100.0%



■今後、Electronic Design and Solution Fairからの情報配信を希望しますか？

	2008(今回)	2007(参考)
はい	57.1%	52.9%
いいえ	41.7%	42.3%
無回答	1.2%	4.8%
合計	100.0%	100.0%



3.1.15 出展社アンケート回答 集計結果

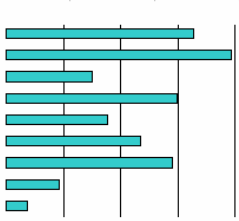
出展社アンケートの回答結果を示す。

出展社 169 社の内、回答 51 社のため、下記結果により　傾向を把握するのは困難である。来年以降　回収率を向上させる施策が必要である。

出展社アンケート/回答数51件

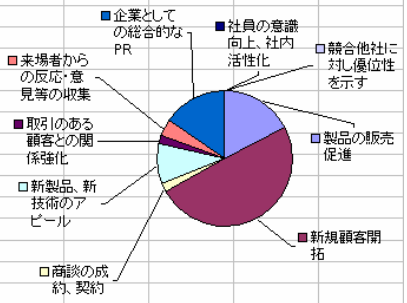
Q1-1. 今回の出展にあたっての目的について該当するものをお選びください。(いくつでも)

	(今回回答数)	2008年度	2007年度(参考)
製品の販売促進	35名	16.4%	17.7%
新規顧客開拓	42名	19.6%	17.7%
商談の成約、契約	16名	7.5%	4.2%
新製品、新技術のアピール	32名	15.0%	14.8%
取引のある顧客との関係強化	19名	8.9%	11.8%
来場者からの反応・意見等の収集	25名	11.7%	11.4%
企業としての総合的なPR	31名	14.5%	12.7%
競合他社に対し優位性を示す	10名	4.7%	5.5%
社員の意識向上、社内活性化	4名	1.9%	4.2%
合計	**214名**	**100.0%**	**100.0%**



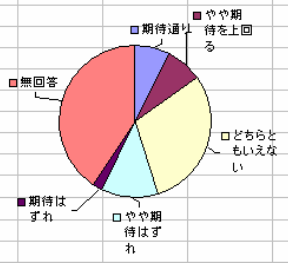
Q1-2. なかでも今回の出展にあたってもっとも重視した目的一箇所をお選びください。

	(今回回答数)	2008年度	2007年度(参考)
製品の販売促進	9名	17.6%	16.0%
新規顧客開拓	25名	49.0%	48.0%
商談の成約、契約	1名	2.0%	0.0%
新製品、新技術のアピール	5名	9.8%	20.0%
取引のある顧客との関係強化	1名	2.0%	4.0%
来場者からの反応・意見等の収集	2名	3.9%	4.0%
企業としての総合的なPR	8名	15.7%	8.0%
競合他社に対し優位性を示す	0名	0.0%	0.0%
社員の意識向上、社内活性化	0名	0.0%	0.0%
合計	**51名**	**100.0%**	**100.0%**



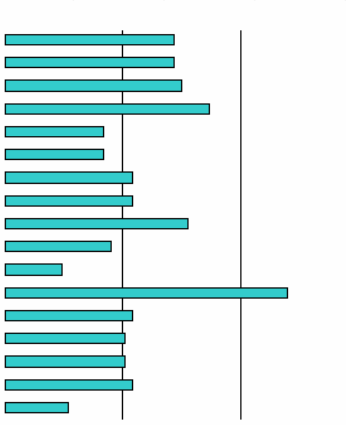
Q1-1で「製品の販売促進」を目的とされていた場合、当項目の評価をお聞かせください。

	(今回回答数)	2008年度	2007年度(参考)
期待通り	4名	7.8%	2.4%
やや期待を上回る	4名	7.8%	7.1%
どちらともいえない	15名	29.4%	40.5%
やや期待はずれ	6名	11.8%	14.3%
期待はずれ	1名	2.0%	4.8%
無回答	21名	41.2%	30.9%
合計	**51名**	**100.0%**	**100.0%**
加重平均値		2.0%	-9.6%



Q2. 出展にあたってターゲットとした来場者の業種について該当するものをお選びください。(いくつでも)

	(今回回答数)	2008年度	2007年度(参考)
【機器メーカ・部門】コンピュータ関連機器	24名	7.2%	6.9%
【機器メーカ・部門】ネットワーク関連機器	24名	7.2%	6.9%
【機器メーカ・部門】一般民生機器	25名	7.5%	8.2%
【機器メーカ・部門】画像処理機器	29名	8.7%	8.8%
【機器メーカ・部門】医療機器	14名	4.2%	3.5%
【機器メーカ・部門】アミューズメント	14名	4.2%	4.1%
【機器メーカ・部門】自動車・輸送機器	18名	5.4%	6.0%
【機器メーカ・部門】産業機器(機械・精密機器等)	18名	5.4%	6.0%
【機器メーカ・部門】通信機器	26名	7.8%	7.9%
【機器メーカ・部門】放送機器	15名	4.5%	4.4%
【機器メーカ・部門】その他	8名	2.4%	2.8%
【半導体・電子部品メーカ部門】システムLSI、ASIC、マイコン、メモリ	40名	12.0%	12.9%
【半導体・電子部品メーカ部門】FPGA/PLD	18名	5.4%	5.0%
【半導体・電子部品メーカ部門】ディスプレイ	17名	5.1%	4.4%
【半導体・電子部品メーカ部門】電子コンポーネント	17名	5.1%	4.1%
【半導体・電子部品メーカ部門】プリント基板	18名	5.4%	4.4%
【半導体・電子部品メーカ部門】その他	9名	2.7%	3.7%
合計	**334名**	**100.0%**	**100.0%**



なかでも出展にあたって最も重視したターゲット一箇所をお選びください。

	(今回回答数)	2008年度	2007年度(参考)
【機器メーカ・部門】コンピュータ関連機器	0名	0.0%	2.0%
【機器メーカ・部門】ネットワーク関連機器	1名	2.0%	2.0%
【機器メーカ・部門】一般民生機器	6名	11.8%	12.0%
【機器メーカ・部門】画像処理機器	6名	11.8%	6.0%
【機器メーカ・部門】医療機器	0名	0.0%	0.0%
【機器メーカ・部門】アミューズメント	0名	0.0%	0.0%
【機器メーカ・部門】自動車・輸送機器	0名	0.0%	2.0%
【機器メーカ・部門】産業機器(機械・精密機器等)	1名	2.0%	0.0%
【機器メーカ・部門】通信機器	1名	2.0%	4.0%
【機器メーカ・部門】放送機器	0名	0.0%	0.0%
【機器メーカ・部門】その他	1名	2.0%	0.0%
【半導体・電子部品メーカ部門】システムLSI、ASIC、マイコン、メモリ	22名	43.1%	50.0%
【半導体・電子部品メーカ部門】FPGA/PLD	3名	5.9%	8.0%
【半導体・電子部品メーカ部門】ディスプレイ	0名	0.0%	4.0%
【半導体・電子部品メーカ部門】電子コンポーネント	0名	0.0%	2.0%
【半導体・電子部品メーカ部門】プリント基板	4名	7.8%	2.0%
【半導体・電子部品メーカ部門】その他	1名	2.0%	0.0%
無回答	5名	9.8%	6.0%
合計	**51名**	**100.0%**	**100.0%**

3.1.16 まとめ

EDSFair2008 では、EDA ベンダーの展示会から技術者への情報提供の場という意味を強めるため、特設ステージ、新興ベンダーツアー、JEVeC ビレッジと新しい企画を実施し、EDA ベンダーのプライベートセミナー、他展示会では得られない情報の提供を行った。いずれの企画も 予想以上の参加者を集めることができ、最新情報発信という目的は十分達成できた。

一方で、来場者数総数が減少してしまった。特別企画には、来場者の多くが興味を持っており、集客を見込めるが、全体の来場者数を増加させるまでには至らなかった。

また、出展社数については、過去最高の 169 社であったが、出展小間数は 昨年より 9 小間減少してしまった。出展小間数増大のため、振興ベンダーの出展誘致活動を推進する必要があるが、既出展社(特に大手)の継続出展および小間数確保に向けた取り組みが必要もある。

今回、小間数減少により 単年度収入は減少し、反面展示会場拡大、特別企画(同時通訳等)により支出増加が見込まれたが、個々の支出項目を精査し費用の最適化を実施し、単年度でも黒字化できた。当面、各項目の費用については、今年度を基準に予算化すれば、大きな無駄な支出を抑えることができる、収支面では適切な運営になる。

今後も EDSFair を維持、発展させるためには、出展社、来場者を増加させることが重要である。そのためには、EDSFair のステータスをより高め、他の展示会、プライベートセミナーとの違いを鮮明にする必要がある。その上で、EDSFair より さらに強烈なプロモーションワードを発信し、出展社にも他展示会と異なる出展のコンセプトを打ち出してもらう必要がある。

3.2 システム・デザイン・フォーラム 2008

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 技術専門委員会主催による“システム・デザイン・フォーラム 2008”を、EDSFair2008 と同期して 2008 年 1 月 25 日に開催した。

EDA 技術専門委員会は、今回と同様の目的とするフォーラムを、ほぼ毎年継続して開催してきた。まず、1990 年から 1994 年にかけて 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した“EDA 標準化フォーラム”を 4 回開催し、つづいて、EDA 技術専門委員会の活動に係る内容の発表、討論の場を目的として“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。また、2004 年には、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 年には “システム・デザイン・フォーラム 2005”を、2 日間の日程で開催した。この“システム・デザイン・フォーラム 2005”では、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行っている。また、2006 年には、“システム・デザイン・フォーラム 2006”として、“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。更に、昨年は“システム・デザイン・フォーラム 2007”として、“SystemVerilog ユーザ・フォーラム”と“SystemC ユーザ・フォーラム”の 2 セッションに、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を紹介するフィジカル･デザイン･フォーラムを新たに加え、計 2 日間 3 セッションを開催した。

そして、本年は“システム・デザイン・フォーラム 2008”として、“SystemC ユーザ・フォーラム”と“Power Format フォーラム”の 2 セッションを開催した。“SystemC ユーザ・フォーラム”では、最新の SystemC 標準化動向、トランザクション・レベル・モデリング向けライブラリである TLM2.0 のチュートリアル、JEITA SystemC ワーキング・グループの TLM2.0 に関する取り組みの報告と､ユーザからの設計適用事例の紹介を行った｡“Power Format フォーラム”では、システム LSI の低消費電力設計向け言語の標準化動向の紹介と論議を行うために、最新の低消費電力設計技術の紹介と、個々に Power Format 標準化を目指す二つの団体 Accellera Organization, Inc.、Si2(Silicon Initiative, Inc.双方からの標準化活動の最新状況や設計適用事例の紹介と、JEITA Power Format 検討ワーキング・グループの Power Format の標準化に対する検討状況の報告を行った。

3.2.2　システム・デザイン・フォーラム 2008 概要

- 開催日時

1 月 25 日(金)	10:00～12:00	**SystemC ユーザ・フォーラム 2008**
	12:45～17:15	**Power Format フォーラム**

- 開催場所

  パシフィコ横浜 アネックスホール

- 聴講料（消費税込）

	事前申込	当日申込
SystemC ユーザ・フォーラム 2008	2,100 円	2,625 円
Power Format フォーラム	3,150 円	4,200 円

- 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 Organization, Inc.

  OSCI(Open SystemC Initiative)

  Si2(Silicon Initiative, Inc.)

- 受 付

  インターネットで受け付け

  http://www.edsfair.com/conference/systemdesign.html

- 開催案内プログラム

社団法人電子情報技術産業協会（JEITA）EDA 技術専門委員会では、毎年、最新 EDA 技術の普及促進を目的としてシステム・デザイン・フォーラムを開催しております。その中で今年は、「SystemC ユーザ・フォーラム 2008」に加えて、最近、話題となっておりますシステム LSI の低消費電力設計向け言語の標準化動向の紹介と論議を行う「Power Format フォーラム」を新たに開催します。

SystemC ユーザ・フォーラム 2008 では、OSCI(Open SystemC Initiative)によります最新の SystemC 標準化動向とトランザクション・レベル・モデリング向けライブラリである TLM2.0 のチュートリアル、JEITA SystemC ワーキング・グループによります TLM2.0 に関する取り組みの報告、およびユーザーからの設計適用事例のご紹介をいたします。是非、システム LSI 設計の最先端状況の把握と、論議の場としてお役立てください。

Power Format には、Unified Power Format（UPF）と Common Power Format (CPF)

の二つの標準化活動が進められているところから、今後の動向が非常に懸念されております。そこで、Power Format フォーラムでは、半導体理工学研究センター（STARC)からの最新の低消費電力設計技術と標準化への期待に関する講演に加えて、UPF：Accellera Organization, Inc.、CPF:Si2(Silicon Initiative, Inc.)の双方から、標準化活動の最新状況や設計適用事例のご紹介をいたします。このフォーラムは、UPF と CPF の現状に関して一度に知る機会でもあり、また、今後の低消費電力設計のあるべき姿を考え論議する良い場になると確信しております。

● **SystemC ユーザ・フォーラム 2008 (1 月 25 日 10:00～12:00)**

司会：長谷川 隆 氏 JEITA SystemC ワーキング・グループ 主査［富士通］

SystemC は、2005 年 12 月に IEEE において、SystemC の標準 IEEE 1666-2005 が承認された後も、OSCI SystemC2.2 や TLM2.0 のリリース等が行われ、さらに活動の範囲は広がりつつあります。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、今後もさらなる期待が寄せられています。本セッションでは、1) OSCI による SystemC アップデートと TLM2.0 チュートリアル、2) JEITA SystemC ワーキング・グループによる SystemC ベースの TLM2.0 に関する取り組みの報告、3) SystemC を用いた設計事例の発表を行います。

① OSCI アップデート：Patric Sheridan 氏（OSCI)
② TLM2.0 チュートリアル：Thorsten Grötker 氏（OSCI)
③ TLM2.0 に関する取り組み：SystemC ワーキング・グループ
④ TLM を活用したソフトウェア早期開発：中村 和正 氏［富士通］

● **Power Format フォーラム（1 月 25 日 12:45～17:15)**

総合司会／フォーラム全体紹介：太田 光保 氏
JEITA システム・デザイン・フォーラム・ワーキング・グループ主査
［松下電器産業］

♦ 最新のシステム LSI 低消費電力設計技術 －標準化に期待するところー：
西口 信行 氏［STARC 執行役員 開発第 1 部長］

最新のシステム LSI の低消費電力設計では、マルチ電源、マルチ閾値電圧の使用、クロックゲーティング、パワーゲーティング、ベクターレス電力計算、トランジスタゲート長調節など、さまざまな技術を組み合わせて使うことが必要となっています。本講演では、この低消費電力設計を可能にする技術の現状を解説いたします。また、これらの技術の組み合わせを間違いなく効率よくコントロールするには、標準化された方式を使う必要があり、その重要性や期待するところについても述べます。

♦ Unified Power Format（UPF）フォーラム（同時通訳付）

司会/Introduction：Dennis Brophy 氏 ［Accellera, Vice-Chair］

今日、消費電力は、システム LSI のアーキテクチャ担当や設計担当にとって最重要課題の一つになっています。これまで、各 EDA ベンダーは各々のユーザーの低消費電力設計へのニーズに対応して独自仕様のフォーマットを定義し、またユーザーは、それらを用いて設計をしていました。しかし、この複数のフォーマットは、設計フローの一貫性の喪失や、記述が冗長になることから来るミスや仕様変更への追従漏れなどを引き起こしていました。

UPF は、DAC 2006 で多くのユーザから出された、RTL から GDSII までの低消費電力設計をカバーする標準 Power Format の定義要望に対して、Accellera（アクセレラ）が 2007 年 2 月に策定したものです。現在では複数のベンダーが、その低消費電力設計のフローの中で、UPF をサポートしています。

このフォーラムでは、UPF を基軸とするアーキテクチャ仕様から実装までのトータルな低消費電力設計ソリューションと、その事例のご紹介をいたします。

① Architecting a low power design with UPF :
Tom Chau 氏 ［Synopsys, Sr. Director CAE］

② Functional verification of a low power design:
Stephen Bailey 氏［Mentor Graphics, Product Marketing Manager］

③ Synthesizing and implementing the low power design：
Arvind Narayanan 氏 ［Magma Design Automation, Product Director］

④ 事例：UPF Experience & Conclusion:
Dennis Brophy 氏 ［Accellera, Vice-Chair］

♦ Common Power Format（CPF）フォーラム

司会/Introduction： Jake Buurma 氏 ［Si2, VP］

CPF は、Cadence 社が提唱し設立された標準化団体 Power Forward Initiative によって策定され、現在は Si2 の標準として認定されている Low Power 設計フォーマットです。本フォーラムでは、CPF 標準化の背景及び最近の Si2 Low Power Coalition グループによる活動状況について述べると共に、CPF の活用実例をご紹介いたします。電源遮断や多電源設計など複雑化する最先端 Low Power 設計に対する、論理検証・テスト設計・合成・配置配線・サインオフ解析までの CPF を基軸とするトータルな設計ソリューション、及び STARC と NEC マイクロシステムによる CPF の具体的な適用事例をご紹介する事で、標準フォーマット CPF を設計フローの中で共有、活用する事の利点をご説明いたします。

① CPF の概要と CPF を活用した Low Power 設計環境ソリューション：
鈴木 貞雄 氏 ［日本ケイデンス・デザイン・システムズ社
シニア テクニカル マーケティング マネージャー］

② 事例 1：STARCAD-CEL PRIDE V1.5 ローパワー設計フロー：
杉岡 俊明 氏 ［STARC 開発第 1 部 チームリーダー］

③ 事例 2：65nm/55nm 世代以降の究極の低消費電力ＳｏＣ設計技術を目指して：
菊池 洋 氏 ［NEC マイクロシステム 基盤コア開発事業部 主任］

♦ Power Format への JEITA の取り組み：
山田 節 氏 JEITA EDA 標準化小委員会 主査［三洋電機］

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 2008 の参加者数

	SystemC ユーザ・フォーラム	Power Format フォーラム
事前申込者	160	113
当日申込者	12	9
申込者数合計	172	122
参加者数（有料）	165	117
アンケート回答数	145	98

今年は、SystemC ユーザ・フォーラムで、申込者数が 172 名と昨年に対し 24 名増と大きく増加した。また、PowerFormat フォーラム新たに開催したことで、これまでとは異なる分野の技術者に対して、標準化状況の普及と JEITA EDA 技術専門委員会の活動のアピールができた。

● 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。

♦ SystemC ユーザ・フォーラム 2008

本年の SystemC ユーザ・フォーラムでは、参加申込者が昨年より 24 名多い 172 名となり、目標としていた 150 名を上回る事が出来て盛況であった。講演は 2 時間で 4 講演を実施し、例年通り OSCI からのアップデートに加え、本年は現在 OSCI で策定中の OSCI TLM2.0 に関するチュートリアルを実施した。講演全体通して主に TLM2.0 を中心とした構成となった。

図 1　SystemC ユーザ・フォーラム 2008

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 OSCI のロードマップと TLM ワーキングにおける TLM2.0 Draft2 の最新状況の説明があった。次に今回のフォーラムの目玉である OSCI TLM2.0 Draft2 チュートリアルを OSCI から Throsten Grotker 氏を招いて行った。ユーザーに OSCI TLM2.0 の特徴を伝える非常に良い機会となったが、内容の濃さから時間配分が４０分では不足していたように思う。SystemC-WG からは、OSCI TLM2.0 の要求仕様をレビューした結果の報告を行った。TLM チュートリアルでは説明されないユースケースの分析、いくつかの機能等の背景や注意点をユーザーに伝える事が出来た。また、SystemC-WG における SystemC を使用する際に推奨される設計メソドロジの構築作業の現状報告も行った。ユーザー事例は富士通の中村氏より、富士通社内における TLM を用いた検証事例について講演頂き、OSCI TLM2.0 Draft2 を適用した際の特徴や問題点をユーザーに説明頂いた。ユーザーにと

って実際に TLM を使用する際の注意点が浮き彫りになる興味深い内容であった。

TLM チュートリアルがやや時間不足で予定より１０分程度オーバーしたが、フォーラム全体を通してはほぼ予定通りの時間で終了。大きな混乱もなく順調に開催する事が出来た。

♦ Power Format フォーラム

本年は、PowerFormat フォーラム新たに開催した。新フォーラムで認知度がまだ低いと思われたことで参加者数が心配されたが、122 名の参加申し込みがあり、盛況に開催することが出来た。

本フォーラムは、大きく次の四つのパートで構成した。

① EDA 技術専門委員会からのフォーラム開催の狙い説明と、半導体理工学研究センター (STARC)からの最新の低消費電力設計技術と標準化への期待に関する講演。

② Accellera Organization, Inc からの Unified Power Format（UPF）標準化活動の最新状況や設計適用事例を紹介する UPF フォーラム。

③ Si2(Silicon Initiative, Inc.)からの Common Power Format（CPF）標準化活動の最新状況や設計適用事例を紹介する CPF フォーラム。

④ EDA 標準化小委員会からの Power Format への JEITA の取り組みに関する講演。

図２　Power Format フォーラム

最初にシステム・デザイン・フォーラム主査 太田 光保 氏(松下電器産業)から Power Format フォーラム開会の趣旨説明として、Power Format 標準化の必要性と標準化活動への懸念事項に関する背景説明とフォーラム内での論議への期待が述べられた。続いて、STARC の西口 信行 氏が登壇し、最新のシステム LSI の低消費電力設計技術と、標準化の重要性と期待が解説された。また、質疑の中で、システム LSI 設計者の標準化への積極的な貢献に対する期待も述べられた。

次に、UPF フォーラムが、Accellera Organization, Inc の Dennis Brophy 氏の司会で行われた。まず、Dennis Brophy 氏がフォーラム紹介を行い、それに続いて Tom Chau 氏(Synopsys)、Stephen Bailey 氏(Mentor Graphics)、Arvind Narayanan 氏(Magma Design Automation)が、UPF を用いた設計方法および UPF の言語としての解説を行った。最後に再度 Dennis Brophy 氏が登壇し、UPF の実績について状況説明があった。また、質疑としては、会場から二つの標準化活動の統合の可能性についての質問があった。

続いて、CPF フォーラムが、Silicon Initiative, Inc.(Si2)の Jake Buurma 氏 の司会で行われた。まず、Jake Buurma 氏がフォーラム紹介を行い、それに続いて鈴木 貞雄 氏（日本ケイデンス）が、CPF の概要と CPF を活用した Low Power 設計環境について解説した。次に、杉岡 俊明 氏（STARC）と菊池 洋 氏（NEC マイクロシステム）が、CPF を用いた設計事例を紹介した。質疑では、事例紹介に対して、具体的な活用形態について活発な論議があった。

最後に、JEITA の EDA 標準化小委員会の山田 節 氏が登壇し、Power Format への JEITA の取り組みとして、国際標準化活動への貢献と Power Format 標準化に向けてのアクションの説明と、UPF・CPF 両フォーマットに関する比較調査の状況について解説を行った。

3.2.4 まとめ

フォーラム終了後のアンケートでは、SystemC ユーザ・フォーラム 2008 の満足度（満足＋まあ満足）が 81%、Power Format フォーラムの満足度（満足＋まあ満足）が 78%となっており、両フォーラムとも聴講者の期待に答えることができた。また、今後の参加希望についての設問でも、SystemC ユーザ・フォーラム 2008 の「希望する」（参加する＋内容次第で参加する）が 79%、Power Format フォーラムの「希望する」が 57%、未定（内容次第）を加えると 93%と、フォーラムの内容の充実と開催への期待が確認された。

SystemC ユーザ・フォーラム 2008 は、特に TLM2.0 の最新状況が重要なトピックスであり高い満足度が得られたものと考えられる。しかしながら、内容に対して時間が短すぎるとの意見が 23%と、昨年の 7%から大きく増加しており、改善が期待されるところである。

Power Format フォーラムは、CPF と UPF の両標準化陣営を招いての開催であったこと

から、一度に双方の内容の理解と、両者の状況の違いの認識できたことから、満足度が高まったものと考えられる。特に、CPF フォーラムでは実例が充実していたことから、満足度（満足＋まあ満足）が 82%と UPF フォーラムの 63%を大きく引き離す結果となった。実例発表については、今後のフォーラムのプログラム検討において、重要なポイントであることが改めて確認された。逆に CPF フォーラム・UPF フォーラムを同時に開催したことから、開催時間については「長すぎる」が 33%あり、今後、考慮が必要な点となった。また、両フォーマットへの期待の設問（自由記述）に対し、統合化やインタオペラビリティに対する要望が多かったことに対しては、EDA 技術専門委員会の標準化に対するアクティビティを検討する上で、今後の考慮も期待される。

なお、会計の面では、昨年度まで単年度赤字が続いていたが、今年は、費用見直し、一日集中開催、協賛団体の追加で、ほぼ収支を合わせることができた。しかし、財政面の厳しい状況は続いており、システム・デザイン・フォーラムのアクティビティを継続するために、費用バランスを安定させる追加施策が期待される。

3.2.5　システム・デザイン・フォーラム 2008WG 委員

主　査	太 田 光 保	松下電器産業㈱
委　員	秋 山 俊 恭	㈱ルネサステクノロジ
同	藤 波 義 忠	NEC エレクトロニクス㈱
同	浅 井 健 史	ローム㈱
同	奥 村 隆 昌	富士通 VLSI㈱
同	長谷川　隆	富士通㈱
同	中 西 早 苗	NEC エレクトロニクス㈱
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	三橋 明城男	メンター・グラフィックス・ジャパン㈱
同	今 井 正 治	大阪大学
同	若 林 一 敏	日本電気㈱
アドバイザ	山 田　節	三洋電機㈱
オブザーバ	齋 藤 茂 美	ソニー㈱
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 2008

3.3.1 はじめに

ASP-DAC (Asia and South Pacific Design Automation Conference)は、VLSI およびシステム 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、電機メーカおよ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およ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 および WG10.5 に属する CHDL および 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 2008）は 13 回目で、ソウル市（韓国）COEX で 1 月 21 日（月）から 24 日（木）の日程で開催された。

3.3.3 ASP-DAC 2008 の概要

ASP-DAC 2008 の概要を表 1 に示す。一般講演としては、27 カ国から投稿された 350 編の論文の中から 122 編が採択され、3 日間にわたって並列の 4 つのトラック、24 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 2007 に近い 350 件で、論文の採択率は前年と同等の 35%であり、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。これまでどおり、ASP-DAC は名実ともに一流の国際会議であると評価できる。

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、昨年に引き続いて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 に示すベストデザイン 2 件が選ばれて表彰された。

前回と同じく学生フォーラムも実施された。

表 1: ASP-DAC 2006、2007、2008 の比較

開催年	2006 年	2007 年	2008 年
日時	2006 年 1 月 24 日（火） ～27 日（金）	2007 年 1 月 23 日(火) ～26 日（金）	2008 年 1 月 21 日（月） ～24 日（木）
会場	横浜市（日本) パシフィコ横浜	横浜市（日本) パシフィコ横浜	ソウル (韓国) COEX
併設展示会	EDSF 2006	EDSF 2007	---
論文投稿数	424	408	350
論文投稿国 （地域）数	27	30	27
論文採択数 （採択率）	135 (32%)	131（32%）	122（35%）
キーノート アドレス	3 件	3 件	3 件 （表 2 参照）
一般講演	27 セッション（135 編）	27 セッション(131 編）	24 セッション（122 編）
特別セッション （招待講演等）	5 セッション	5 セッション	4 セッション （表 3 参照）
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	昼休みに実施 （Ph.D フォーラム）	昼休みに実施 （Student Forum と 改称）	昼休みに実施 （Student Forum）
ポスターボード	―	―	―
有料チュートリアル	6 件 （全日 2 件、半日 4 件）	6 件　（表 5 参照） （全日 2 件、半日 4 件）	5 件　（表 5 参照） （全日 2 件、半日 3 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）	4 セッション(表 4 参照） （パネル討論 2）

表 2: 基調講演

講演タイトル	講演者
A Brand New Wireless Day	Jan M. Rabaey （ University of California, USA）
The Evolution of SoC Platform According to the New Mobile Paradigm	Ki-Soo Hwang (Core Logic, Korea)
The Future of Semiconductor Industry - A Foundry's Perspective	F. C. Tseng (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
パネル討論	セッション 3D: The Tears and Joy of Sowing and Reaping Complex SoC's
	セッション 6D: How to Design Cool Chips for Hot Products
	セッション 7D: Concurrent SoC and SiP Designs
招待講演	セッション 2D: Tackling Manufacturability/Variability for 45nm and Below

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
パネル討論	セッション 5D: Are System Level EDA Tools/Methodologies Coming?
	セッション 9D: Best Ways to Use Billions of Devices on a Chip
一般講演	セッション 4D: New Emerging Application Areas for Future SoC
	セッショ:ン 8D: Low Power Chips

表 5： Tutorial のタイトル

トラック	種類	タイトル
1	全日	System-Level Synthesis: Functions, Architectures, and Communications
2	全日	Cross-Layer Approaches to Designing Reliable Systems Using Unreliable Chips
3	半日	Latest Advances and Future Opportunities on CAD for FPGAs
4	半日	Improvements in 65/45nm Physical Implementation Flow and Methodology
5	半日	On-Chip Network: State-of-the-Art Industrial Solution and Academic Research

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
1A-1: “Variability-Driven Module Selection with Joint Design Time Optimization and Post-Silicon Tuning”, Feng Wang, Xiaoxia Wu, and Yuan Xie (Pennsylvania State Univ., USA)
9A-1: “An Efficient, Fully Nonlinear, Variability-Aware Non-Morte-Carlo Yield Estimation Procedure with Applications to SRAM Cells and Ring Oscillators”, Chenjie Gu and Jaijeet Roychowdhury (Univ. of Minnesota, USA)

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
Best Design Award	1D-1: "A 1.2GHz Delayed Clock Generator for High-Speed Microprocessors", Inhwa Jung, Moo-Young Kim, and Chulwoo Kim (Korea Univ., Korea)
Special Feature Award	1D-7: "Dynamic Supply Noise Measurement Circuit Composed of Standard Cells Suitable for In-Site SoC Power Integrity Verification", Yasuhiro Ogasahara, Masanori Hashimoto, and Takao Onoye (Osaka Univ., Japan)

3.3.4 論文の投稿状況

1997 年から 2008 年の、ASP-DAC への論文投稿数の地域別の推移を図 1 に示す。ただし、ASP-DAC 2002 については、インドのバンガロールで同時開催された VLSI Design 2002 への投稿論文を含む。図 1 に示すように、1999 年（香港開催）以降は投稿論文数が増加し、安定的に 300～400 件の投稿がある。名実ともに、ASP-DAC は設計自動化分野の国際会議として定着したと言ってよいであろう。

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が全体に占める割合は、2000 年をピークにして、10%前後に低下している。今回は、中国と韓国以外の地域からの投稿がやや減少した。論文投稿数が多かったのは、米国の 92 編（前回は 129 編）、中国の 67 編（前回は 61 編）、日本の 31 編（前回は 44 編）、台湾の 34 編（前回は 40 編）、韓国の 33 編（前回は 23 編）であった。欧州からは 34 編（前回は 40 編）の投稿があった。

図 1　地域別論文投稿数

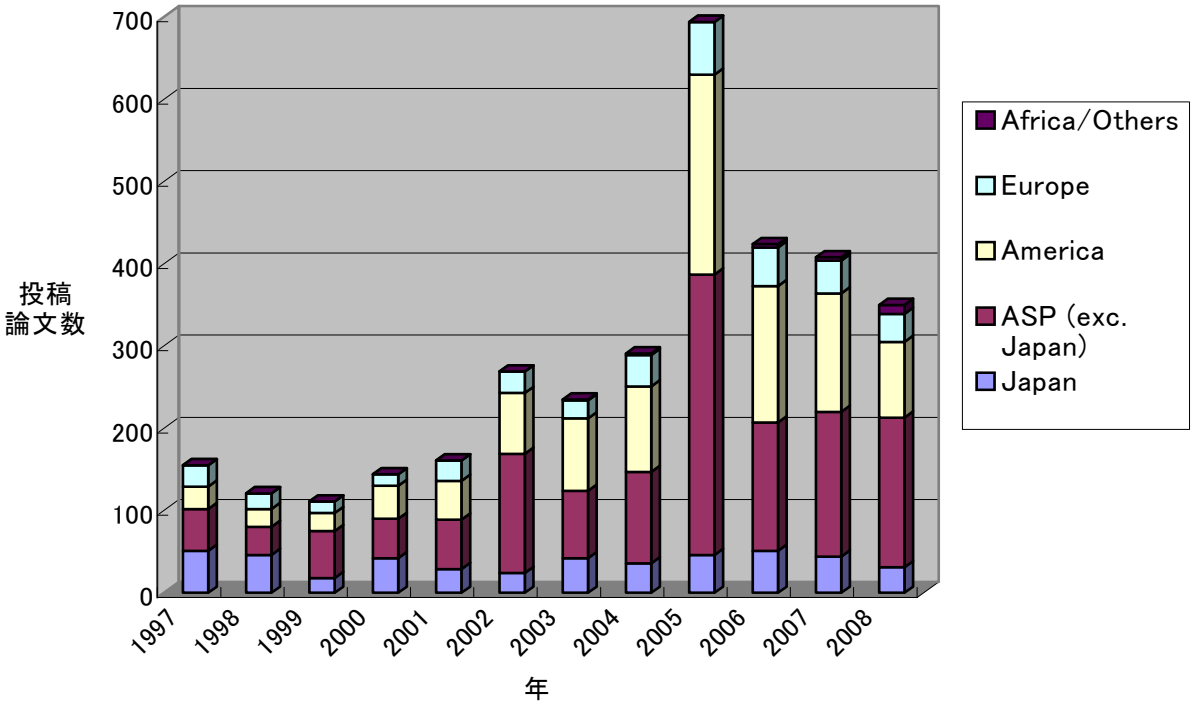


表 8 日本からの論文投稿数と全体に占める割合

年 / 地域	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008
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%)	31 (9%)
全体	155	121	111	144	161	269	235	291	692	424	408	350

次に、研究分野別の論文投稿数および採択論文数を表 9 に示す。ASP-DAC 2008 では、ASP-DAC 2004～2007 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 1 のシステムレベル設計のセッション、分野 7 のタイミング・消費電力関係のセッション、分野 2 の組込みおよび実時間システムのセッション、分野 8 のインターコネクト・デバイス・回路のモデリングとシミュレーション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	55	18	32.7%
2	Embedded and Real-Time Systems	40	13	32.5%
3	Behavioral/Logic Synthesis and Optimization	34	12	35.3%
4	Validation and Verification for Behavioral/Logic Design	15	4	26.7%
5	Physical Design (Routing)	17	7	41.2%
6	Physical Design (Placement)	21	9	42.9%
7	Timing, Power, Signal/Power Integrity Analysis and Optimization	50	17	34.0%
8	Interconnect, Device and Circuit Modeling and Simulation	41	15	36.6%
9	Test and Design for Testability	27	10	37.0%
10	Analog, RF and Mixed Signal Design and CAD	16	5	31.3%
11	Leading Edge Design Methodologies	34	12	35.3%
	合　　計	350	122	34.9%

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。今回の全参加者数は 412 名であった。前回（685 名）と比べると、参加者がやや減少している。日本からの参加者数は全体の 18.7%の 77 名であった。

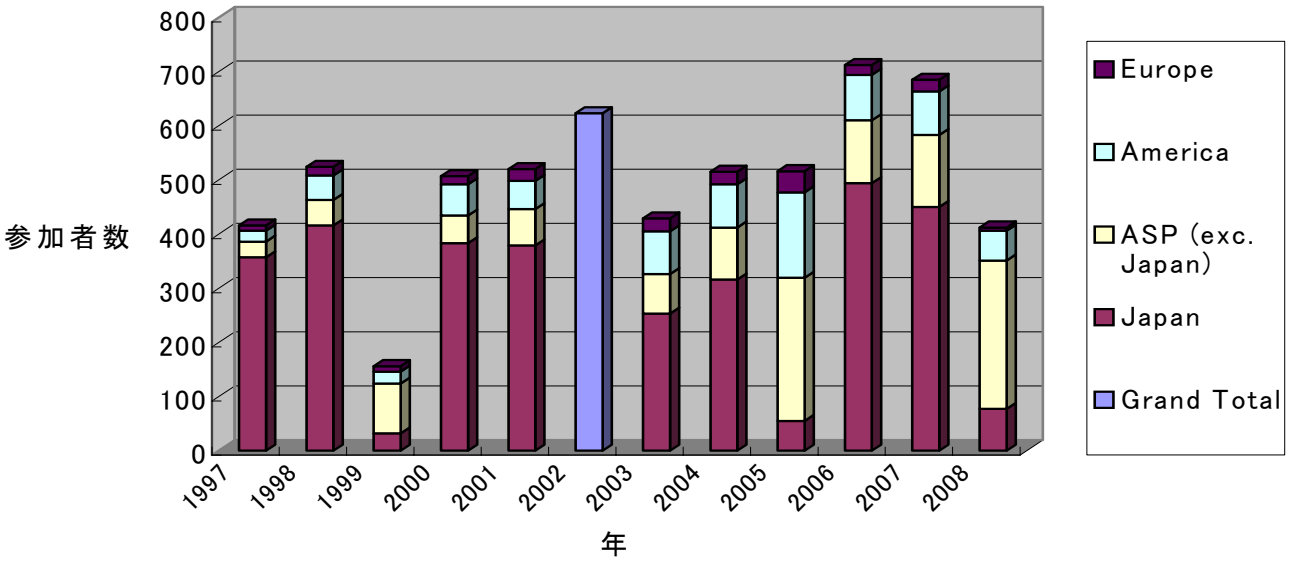


図 2 地域別参加者の推移

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	2008
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%)	450 (66%)
全体	416	524	156	507	520	623	429	515	516	708	685	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。引き続き、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
2009 年	パシフィコ横浜（日本）	2009 年 1 月 19 日（月）～22 日（木）	若林一敏　氏　（NEC）
2010 年	Taipei（台湾）	2010 年 1 月 18 日（月）～21 日（木）	Y.L.Lin 氏　（National Tsing Hua Univ.）

４．添付資料

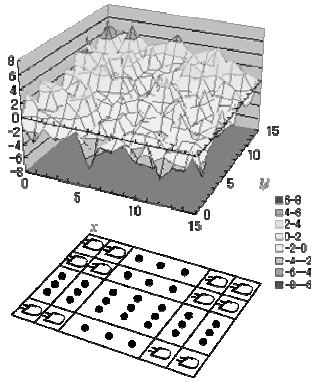
Vthばらつきに拠る出力遷移時間ばらつきの解析

(株)リコー　高藤 浩資
富士通VLSI(株)　奥村 隆昌
三洋半導体(株)　黒川 敦
(株)ルネサステクノロジ　増田 弘生
東京工業大学　佐藤 高史
(株)ルネサス テクノロジ　金本俊幾
大阪大学　橋本 昌宜
NECエレクトロニクス(株)　中島 英斉
(株)ジーダット・イノベーション　小野 信任

報告内容

- 微細化プロセスでのばらつき課題とSSTA
- 出力遷移時間解析手法調査結果
- 感度ベース遅延ばらつき計算手法レビュー
- 感度ベース出力遷移時間計算手法の課題
- 低相関条件での出力遷移時間ばらつき解析
- 出力遷移時間ばらつき計算精度向上の検討
- まとめ

課題：微細化によるLSI性能ばらつきの増大

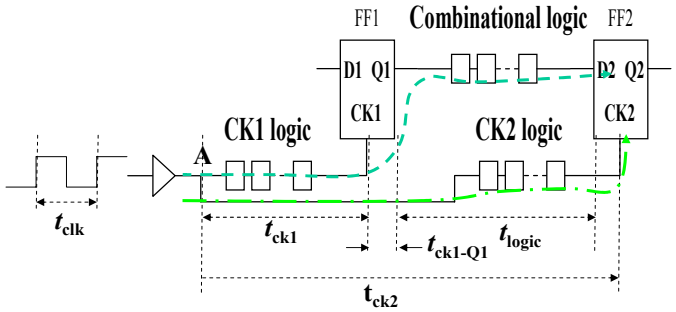


90nm世代遅延ばらつきでの例[9]

リングオシレータ遅延ばらつきの標準偏差(σ)は、平均値(μ)に対して3%～4%

故に、従来のMin, Maxコーナーモデルではチップ内ミスマッチ余裕度を3σ値で10%～15%とらなければならない

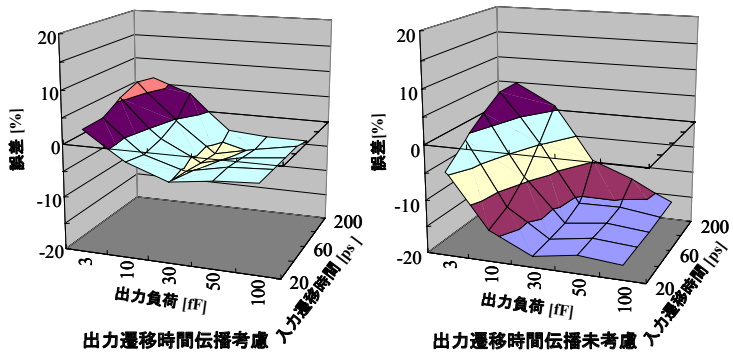
ばらつきを統計的に扱うSSTA [8]



$$t_{CK1}(\mu,\sigma)-t_{CK2}(\mu,\sigma)+t_{CK1\text{-}Q1}(\mu,\sigma)+t_{logic}(\mu,\sigma)+t_{setup}+\delta_{WID}\le t_{clk}$$

$$t_{CK1}(\mu,\sigma)-t_{CK2}(\mu,\sigma)+t_{CK1\text{-}Q1}(\mu,\sigma)+t_{logic}(\mu,\sigma)-t_{hold}-\delta_{WID}\ge 0$$

昨年度活動（PDS研究会）での検討内容［10］



SSTAでの出力遷移時間の取り扱いを検討、出力遷移時間伝播モデルを提案
– vthばらつきに対する出力遷移時間ばらつき感度を考慮
– 前段、自段の2段に着目した解析的出力遷移時間ばらつき伝播モデル
– 伝播項の付加による計算精度の向上を示した
– ただし、負荷小、入力遷移時間大の領域において相対的に精度改善量が少ない

従来技術1：感度ベース手法［1］

遅延時間に対するばらつき量{P_i}の感度に基づき、遅延ばらつきdを表現する。
[1]では、ばらつき量{Pi}が正規分布をとること、{Pi}のばらつき感度が一次の特性を持つことを前提とする。

$$d = d_0 + \sum_{i=1}^{n} \frac{\partial F}{\partial P_i} \Delta P_i$$

感度　ばらつき

$$\mu_d = d_0$$

$$\sigma_d^2 = \sum_{i=1}^{n} \frac{\partial F}{\partial P_i}^2 \sigma_{P_i}^2 + 2\sum_{i \neq j} \frac{\partial F}{\partial P_i} \frac{\partial F}{\partial P_j} \mathrm{cov}(P_i, P_j)$$

演算負荷大

更に、[1]では、主成分分析(PCA: Principle Components Analysis)を利用して{P_i}を主成分{P'_i}に写像し、統計的演算量の削減を図る。

$$d = d_0 + \sum_{i=1}^{m} \frac{\partial F}{\partial P_i'} \Delta P_i'$$

$$\mu_d = d_0$$

$$\sigma_d^2 = \sum_{i=1}^{m} \frac{\partial F}{\partial P_i'}^2 \sigma_{P_i'}^2 + 0$$

演算量削減

出力遷移時間に関しても、遅延時間と同様の考え方で取り扱い可能との記載

従来技術2：出力遷移時間1 [2]

$$\frac{\partial delay_j}{\partial p} = \frac{\partial D_j}{\partial p} + \frac{\partial Tran_{in}}{\partial p}\frac{\partial D_j}{\partial Tran_{in}} + \frac{\partial C_{in}}{\partial p}\frac{\partial D_j}{\partial C_{in}} \quad (2)$$

$$\frac{\partial Tran_j}{\partial p} = \frac{\partial T_j}{\partial p} + \frac{\partial T_{in}}{\partial p}\frac{\partial T_j}{\partial T_{in}} + \frac{\partial C_{in}}{\partial p}\frac{\partial T_j}{\partial C_{in}} \quad (3)$$

感度　　入力遷移時間項　　出力負荷項

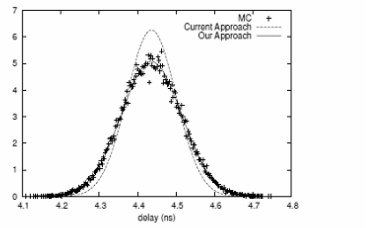


Fig. 4. PDF for 20 buffer chain

感度ベースの手法を出力遷移時間ばらつきに拡張
- [1]に対し、入力遷移時間、出力負荷を陽に含めて定式化
- 出力遷移時間については、遅延パス内各ステージでの伝播を考慮
- ベンチマーク回路での出力遷移時間精度改善を報告

従来技術3：出力遷移時間2 [3]

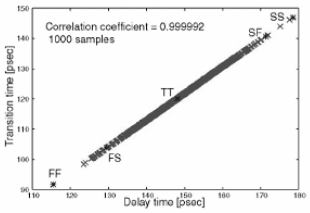


Fig. 3. An example of the relationship between delay variability and transition time variability.

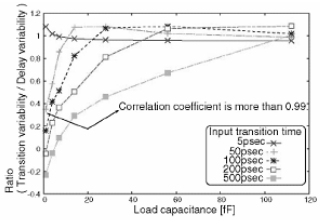


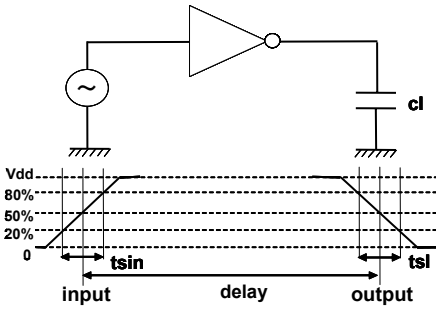
Fig. 4. The values of transition time variability / delay variability in various input transition time and load capacitance.

遅延時間ばらつきと出力遷移時間ばらつきの相関性を利用し精度を向上
- 遅延時間ばらつきと上記相関係数から出力遷移時間ばらつきを計算
- 負荷小、入力遷移時間大の遅延時間と出力遷移時間の相関性が低い条件については、出力遷移時間の値自体が小さく誤差は無視出来るとの前提
- [2] でも、網羅的な負荷と入力遷移時間の組み合わせについての精度評価結果については触れていない

出力遷移時間ばらつきでの検討課題

- 遅延時間ばらつきと出力遷移時間ばらつきの間に相関性が高い条件と、低い条件が存在する
- 相関性が高いとは、vthばらつきに対する出力遷移時間のばらつきが一次の相関特性を持つことと等価であり、感度ベースの手法が有効であることを意味する
- では、相関性が低い条件での出力遷移時間ばらつきの解析はどのように行えば良いのか？

解析：回路、シミュレーション条件

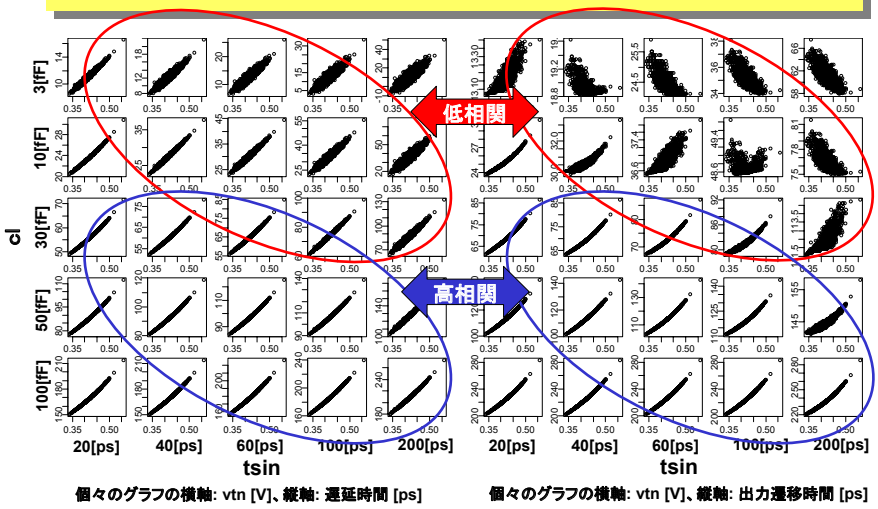


解析回路

Tr.	Model	65nm, Typical [7]
	NMOS	L=60[nm], W=300[nm] vth=N(0.423, 0.03^2) [V]
	PMOS	L=60[nm], W=500[nm] vth=N(-0.365, 0.025^2) [V]
vdd		1.0 [V]
temp.		25 [℃]
cl		3, 10, 30, 50, 100 [fF]
tsin		20, 40, 60, 100 , 200 [ps]
解析手法		モンテカルロ解析
解析回数		1000回

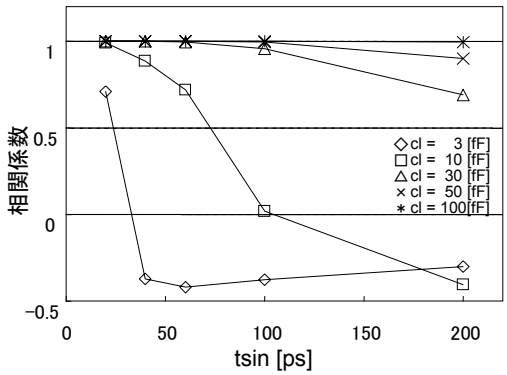
解析条件

遅延、出力遷移時間ばらつきとvthばらつきの関係



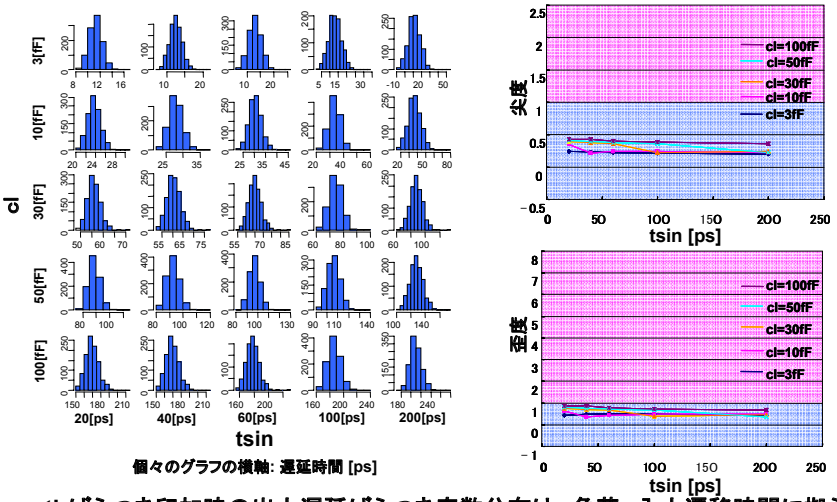
遅延とvthは一次の相関特性。遅延時間と出力遷移時間の相関は条件に依存

遅延時間ばらつきと出力遷移時間ばらつきの相関



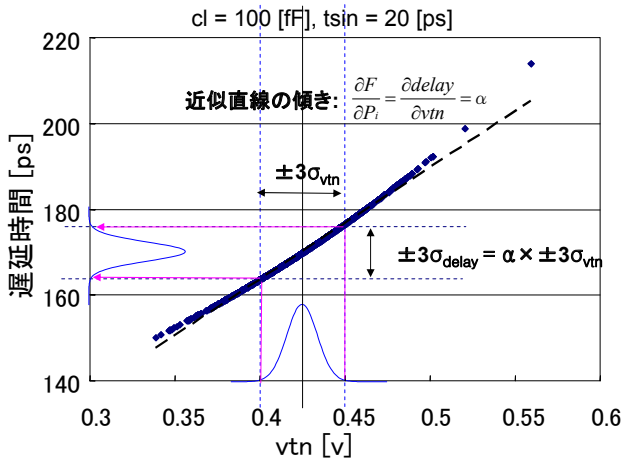
遅延と遷移時間の相関性は、入力遷移時間の増加、または、出力負荷の減少に伴い、低下する([3]での報告の通り)

vthばらつきに対する遅延時間ばらつき



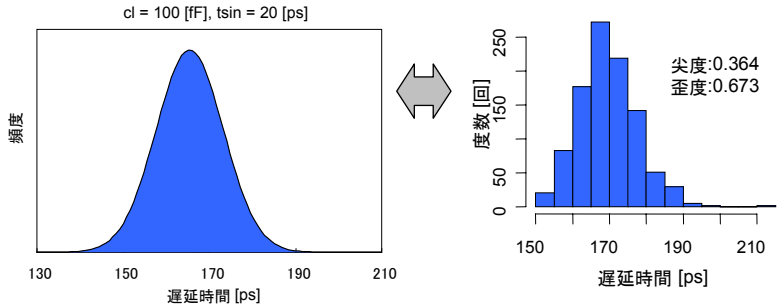
vthばらつき印加時の出力遅延ばらつき度数分布は、負荷、入力遷移時間に拠らず正規分布形状

レビュー：vthばらつきに対する遅延時間ばらつきの計算



感度ベースの考え方で、vthばらつきに対する遅延時間ばらつきを計算。vthのばらつき幅を近似一次式で定まる遅延のばらつき幅とする。

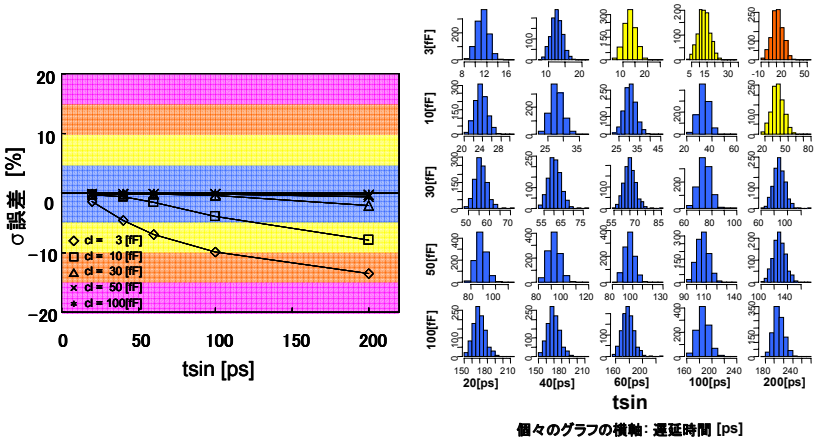
vthばらつきに対する遅延時間ばらつき計算結果



遅延時間ばらつき計算結果　　モンテカルロ解析結果

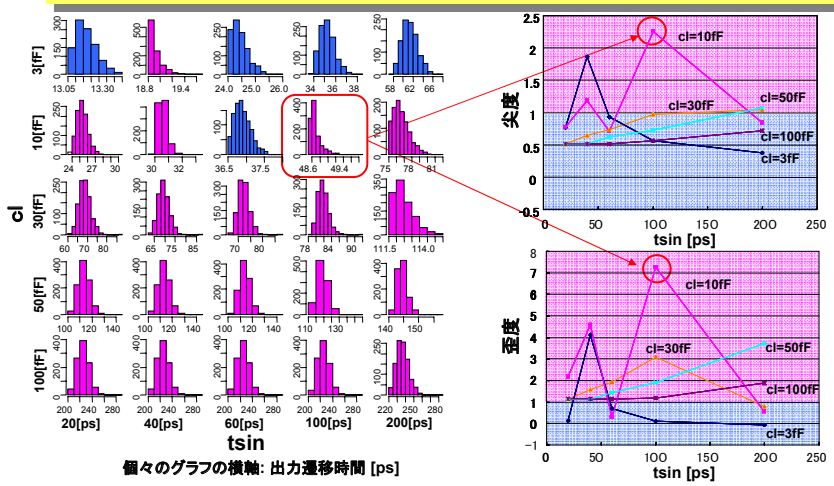
計算した遅延時間ばらつきの分布形状は、モンテカルロ解析結果と一致！

感度ベースでの遅延時間ばらつき計算誤差



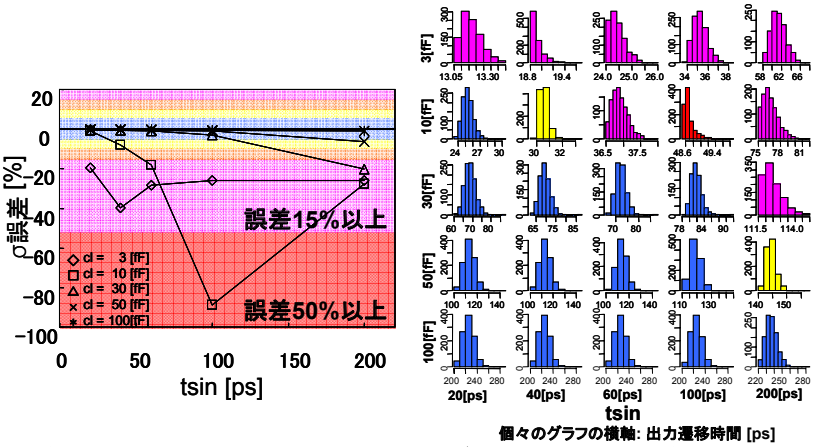
感度ベースに基づく遅延時間ばらつきの標準偏差計算誤差は、最大で約15%

vthばらつきに対する出力遷移時間ばらつき



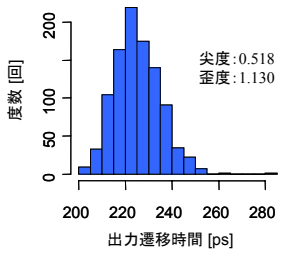
vthばらつき印加時の、出力遷移時間ばらつき度数分布は、分布中心での頻度大きく(尖度正より) 、分布中心が左側に偏在(歪度正より)する形状で、いずれも、正規分布に対し乖離。

感度ベースでの出力遷移時間ばらつき計算誤差

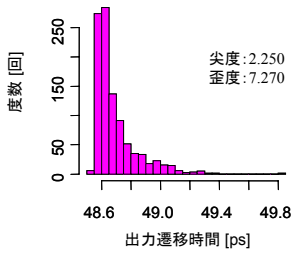


vtnばらつきに対する出力遷移時間感度に基づき、出力遷移時間ばらつきを計算。モンテカルロ解析に対する標準偏差の誤差は、80%を超える！負荷小、入力遷移時間大の領域で誤差が著しく、これは遅延と出力遷移の低相関領域と一致する。

分析：代表点での出力遷移時間度数分布



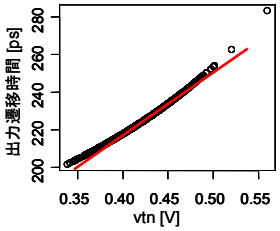
高相関条件代表点
(cl = 100 [fF], tsin = 20 [ps])

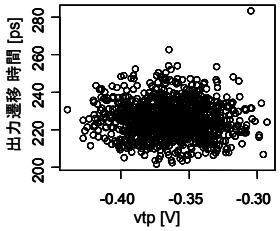


低相関条件代表点
(cl = 10 [fF], tsin = 100 [ps])

遅延時間と出力遷移時間の相関性が対照的な二条件を代表点として選択。高相関条件、低相関条件、各々での出力遷移時間ばらつき特性の支配要因を調査。

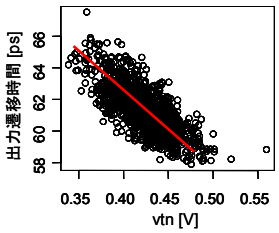
高相関条件での出力遷移時間とvthのばらつき相関

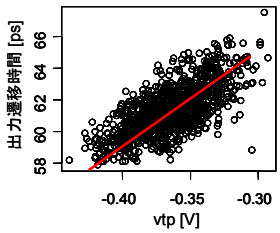




- 遅延と出力遷移時間の相関性が高い条件では、出力遷移時間のばらつきに対して、vtnのばらつきが支配的
- vtpのばらつきに対する出力遷移時間のばらつきは、無相関
- vtnのばらつきと出力遷移時間のばらつきは一次の相関特性を持つ。従って、一次の相関特性に基づき感度ベースの考え方で対応が可能。

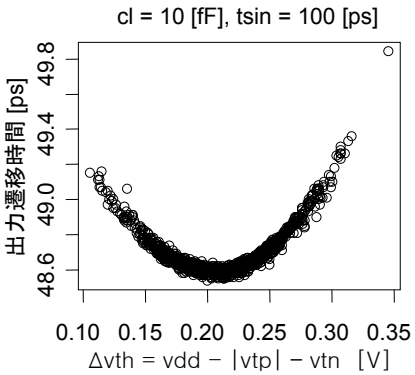
低相関条件での出力遷移時間とvthのばらつき相関





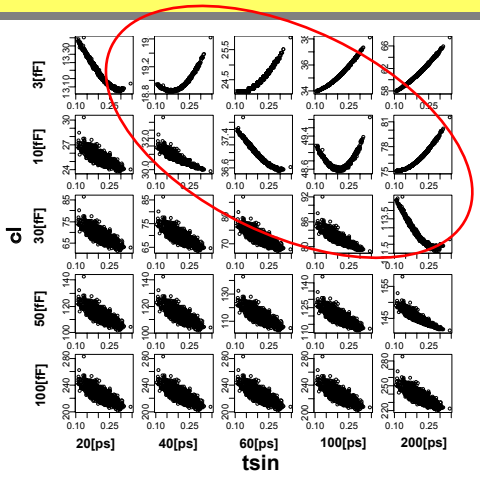
- 遅延と出力遷移時間の相関性が低い条件では、vtnに対して弱い負の相関、vtpに対して弱い正の相関。従って、vtn、vtp双方のばらつきが出力遷移時間のばらつきに影響を与える
- 出力遷移時間大となる条件は、vtn小、vtp大。これは、Low vthセルの構成に等しい
- 出力遷移時間小となる条件は、vtn大、vtp小。これは、High vthセルの構成に等しい
- 出力遷移時間のばらつきは、vtn、vtp電圧の差に依存しているのか？

Δvthばらつきに対する出力遷移時間ばらつき



遅延時間と出力遷移時間の相関性が低い条件で、vtn、vtpの差電位であるΔvthを用いて出力遷移時間のばらつきを集計。この結果、Δvthばらつきに対して、二次状の出力遷移時間相関特性が明らかとなった。

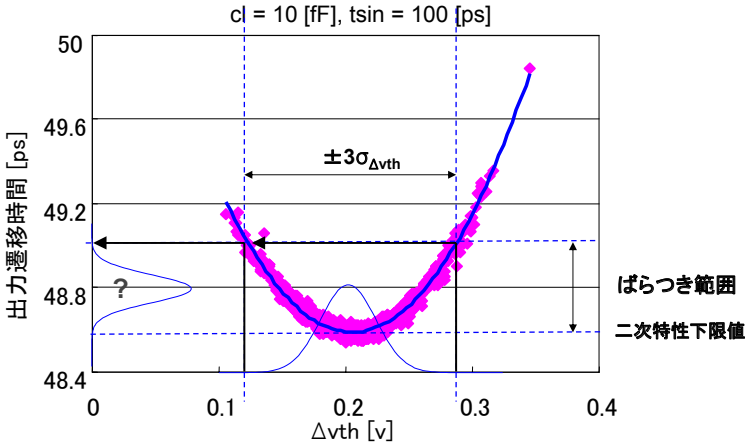
Δvthに対する出力遷移時間ばらつき（全条件）



個々のグラフの横軸: Δvth [V]、縦軸: 出力遷移時間 [ps]

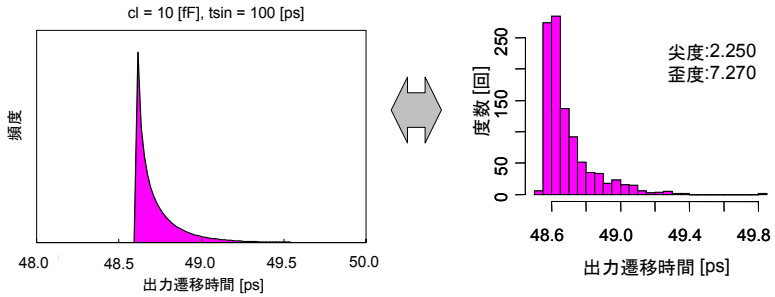
出力遷移時間ばらつきのΔvthばらつきに対する二次状の相関特性は、遅延時間と出力遷移時間の相関性が低下する条件で顕著に観測される

vthばらつきに対する出力遷移時間ばらつきの計算



二次状の相関特性を二次式で近似して、Δvthばらつきに対する出力遷移時間ばらつきを計算。得られる分布形状は、下限値の存在により歪度が正に、同一遷移時間を取るΔvthの値が二個あるため尖度が正に偏移する。

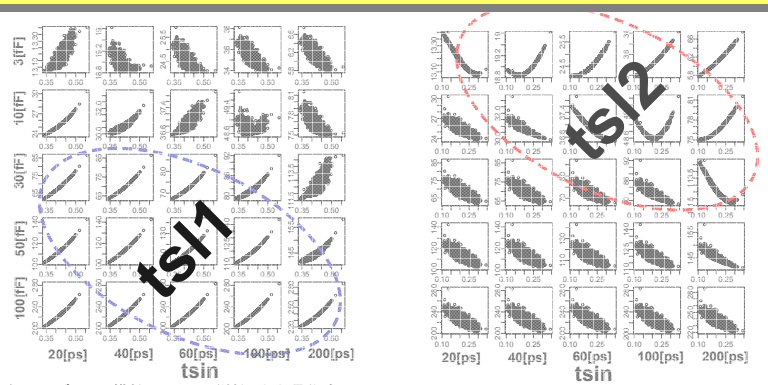
vthばらつきに対する出力遷移時間ばらつき計算結果



出力遷移時間ばらつき計算結果　　モンテカルロ解析結果

計算した出力遷移時間ばらつきの分布形状は、モンテカルロ解析結果と一致！

出力遷移時間ばらつき精度改善の検討

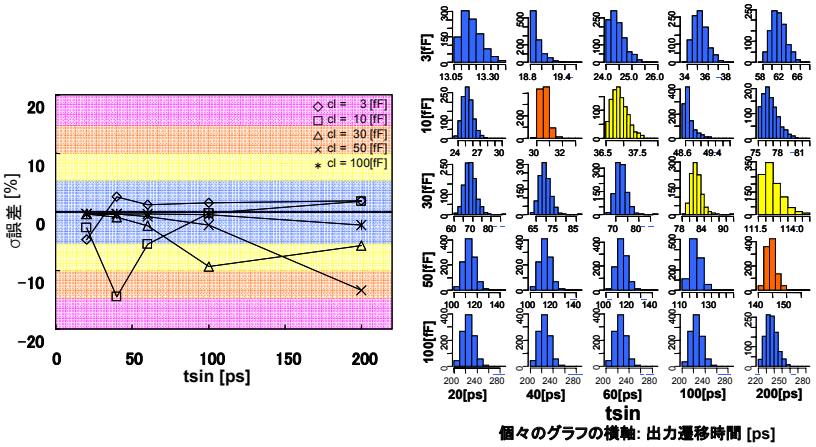


個々のグラフの横軸: vtn [V]、縦軸: 出力遷移時間 [ps]　個々のグラフの横軸: Δvth [V]、縦軸: 出力遷移時間 [ps]

$$tsl = (1 - r) \times tsl1 + r \times tsl2 \quad (0 \leqq r \leqq 1)$$

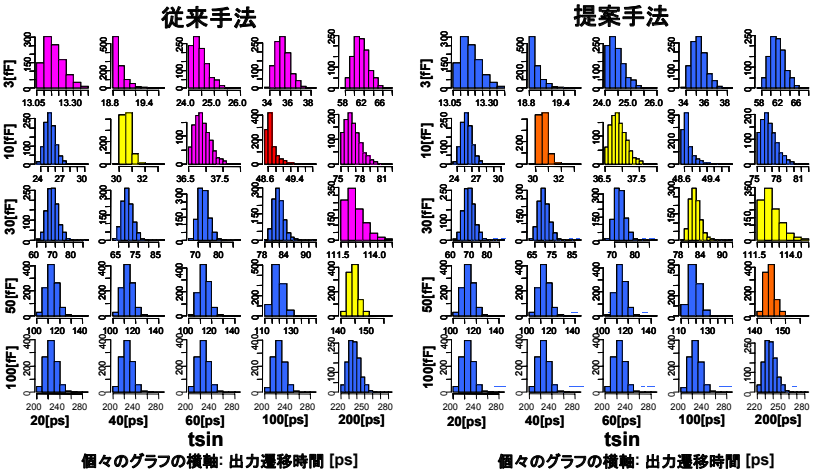
vthに対する一次の相関特性と、Δvthに対する二次の相関特性を組み合わせて、出力遷移時間ばらつきの計算精度改善を図る。rは、二次特性の寄与率で二次特性が支配的な割合を示す係数。

提案手法による出力遷移時間ばらつき計算結果



vtnに対する一次の相関特性、Δvthに対する二次の相関特性を組み合わせることにより、一次特性のみを用いたときの最大誤差約80%が、感度ベースによる遅延時間ばらつき誤差相当の15%程度まで改善。

新旧出力遷移時間ばらつき計算結果比較



提案手法では、一次特性、二次特性の境界領域で計算精度の劣化が認められるが、全体的な計算精度のばらつきは改善。

まとめ

- 65nm CMOSインバータを用いてトランジスタ閾値電圧ばらつきに対する出力遷移時間のばらつきを解析した。
- 遅延時間と出力遷移時間の相関性が低下する「入力遷移時間、出力負荷」の組み合わせにおいては、NMOS、PMOS双方のトランジスタ閾値電圧差(Δvth)の考慮が必要であることを明らかにした。
- vthばらつきに対する一次の相関特性とΔvthに対する二次の相関特性を組み合わせることにより、「入力遷移時間、出力負荷」の組合せに拠らず、出力遷移時間の計算精度が改善されることを示した。

参考文献

[1] H.Chang and S.Sapatnekar, "Statistical timing analysis under spatial correlations," IEEE Trans. CAD, vol.24, no.9, pp.1467-1482, Sep. 2005.
[2] R.Goyal, S.Shrivastava, H.Parameswaran, and P.Khurana, "Improved first-order parameterized statistical timing analysis for handling slew and capacitance variation," Proc. of International Conference on VLSI Design, pp.278-282, Jan. 2007.
[3] T.Kouno and H.Onodera, "Consideration of transition-time variability in statistical timing analysis," Proc. of IEEE International SOC Conference, pp.207-210, Sep. 2006.
[4] S.Raj, S.Vrudhula, and J.Wang, "A methodology to improve timing yield in the presence of process variations," Proc. of DAC, pp.448-453, June 2004.
[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, p.3, Nov. 2004.
[7] http://www.eas.asu.edu/~ptm/
[8] A.Devgan and C.Kashyap, "Block-based static timing analysis with uncertainty," Proc. of ICCAD, pp.607-614, Nov. 2003.
[9] H.Masuda, S.Ohkawa, A.Kurokawa, and M.Aoki, "Challenge: variability characterization and modeling for 65- to 90-nm processes," Proc. of CICC, pp.593-599, Sep. 2005.
[10]高藤, 小林, 小野, 増田, 中島, 奥村, 橋本, 佐藤, "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," 電子情報通信学会 回路とシステム軽井沢ワークショップ, pp.709-714, 2007年4月.

チップ内システマティックばらつきと回路スキュー特性相関

NECエレクトロニクス(株) 中島英斉
(株)ルネサス テクノロジ 増田弘生
富士通VLSI(株) 奥村隆昌
三洋半導体(株) 黒川敦
(株)ルネサス テクノロジ 金本俊幾
東京工業大学 佐藤高史
大阪大学 橋本昌宜
(株)リコー 高藤浩資
(株)ジーダット・イノベーション 小野信任

目次

背景

- MOS素子が微細化してくると、チップ内(WID)ばらつきが無視できなくなる
- WIDランダムばらつきσ_{rnd}は、ゲート面積を大きくする、または、論理段数を多くすることで、回路遅延ばらつきへの影響を、ある程度緩和できることが分かっている
- WIDシステマティックばらつきσ_{sys}の回路特性ばらつきに与える影響は、十分モデル化されているとは言えない

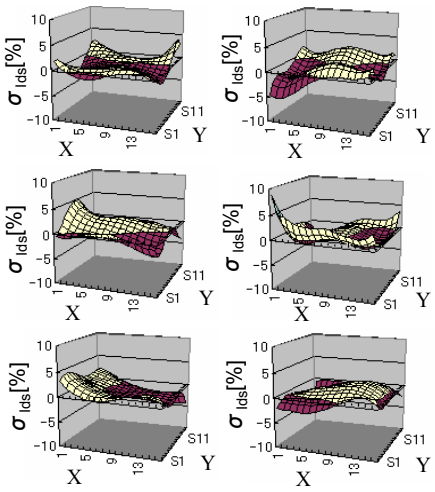
目的

- WIDシステマティックばらつきσ_{sys}モデルの提案
- WID σ_{sys}が回路スキューに与える影響の評価方法提案
- WID σ_{sys}と回路スキューばらつき分布の関係に対する定量的解析と解釈

報告内容要旨

- チップ内システマティックばらつきが、回路特性ばらつきに与える影響を、新しい手法で定量的に解析する
- 解析方法は、システマティックばらつきをランダム曲面でモデル化し、その結果をモンテカルロ解析に取り込む手法である
- その結果と距離相関によるシステマティックばらつきモデル化手法とを比較、検証する
- 結論：
  - 回路スキュー最大値はσ=20.1ps(パス遅延相対値9.0%)レベルで非常に大きくなることがわかった
  - 距離相関によるシステマティックばらつきのモデルと比較した結果、距離相関を用いる方法では、スキュー値を小さく計算する等の問題があることを指摘する
  - 配線をクロスした点対称ドライバでスキューを小さく抑えられることを初めて定量的に確認した

90nm TEG: Ids WIDシステマティック成分の測定結果例



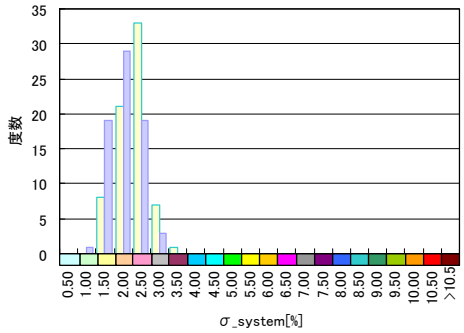
成分抽出は、実測IdsのWIDばらつきに4次式近似でフィッティングすることで行った [6]

観測結果：

・システマティック成分の空間分布は多種多様に変化。空間形状は、ほぼランダムにばらついていると考えられる

・システマティック成分の滑らかな空間分布は、ほぼXY2変数の2次式程度のオーダーで近似できる

チップ内ldsシステマティックばらつきσの分布測定結果例



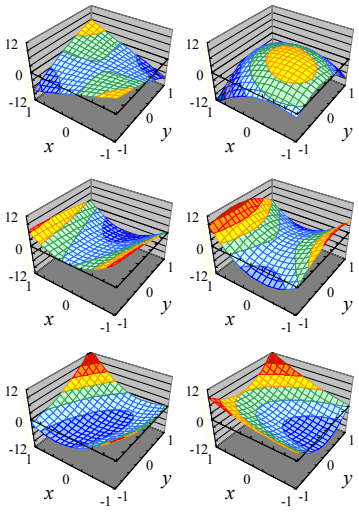
	0° 配置	90° 配置
μ[%]	1.697	1.872
σ[%]	0.382	0.429
min[%]	0.982	1.079
max[%]	3.023	2.761
skew	0.489	0.388
excess	0.690	−0.782

σ_{sys}値ヒストグラム(1ウエハ内)を示す測定結果

観測結果:
システマティックばらつきの大きさは2%程度、1～3%の範囲でばらつく

ウエハ内でのσ_{sys}分布はほぼ正規分布となる

チップ内システマティックばらつきの表現:
ランダム曲面のモンテカルロ生成



2次のLegendreランダム曲面の生成例(規格化)[8]

ランダム曲面の特徴:
- 2次元に拡張した直交Legendre多項式による表現
- モンテカルロで規定のσ_{sys}値をもつランダム平面が簡易に生成できる
- 多項式の次数の選択で、任意の複雑度の平面を記述できる

ランダム曲面の生成アルゴリズム［8］

- $Z_2(x,y) = a_0R_0(x,y) + a_1R_1(x,y) + a_2R_2(x,y) + a_3R_3(x,y) + a_4R_4(x,y) + a_5R_5(x,y)$

$R_0(x,y) = \sqrt{1}\cdot 1$
$R_1(x,y) = \sqrt{3}\cdot x$
$R_2(x,y) = \sqrt{3}\cdot y$
$R_3(x,y) = \sqrt{5}\cdot (3x^2-1)/2$
$R_4(x,y) = \sqrt{9}\cdot xy$
$R_5(x,y) = \sqrt{5}\cdot (3y^2-1)/2$

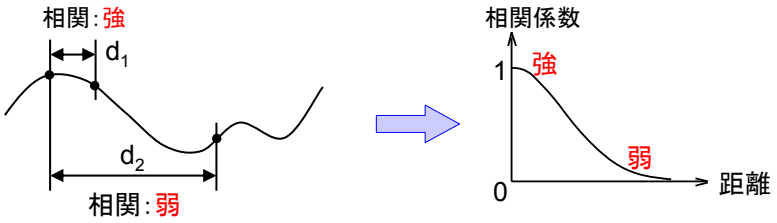
a_0-a_5：正規分布（σ=1）に従うランダム変数

$RMS_{Z2} = \sqrt{6}$

$\Delta Lg(x,y) = Lg_{Typ}\, Z_2(x,y)/\sqrt{6} \times \sigma_{sys}$

WIDのシステマティックバラツキはMOSのLgのばらつきの影響が支配的である[3]

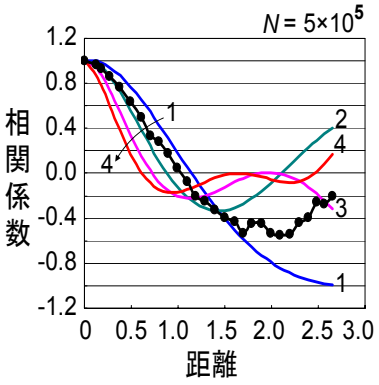
システマティックばらつきの特徴化
相関係数の距離依存という扱い



システマティックに変化する曲面上の2点。
その2点間の相関係数は距離に依存する。

これが現在の標準的な設計原理

ランダム曲面の「相関(平均) vs. 距離特性」へのマッピングと実測比較

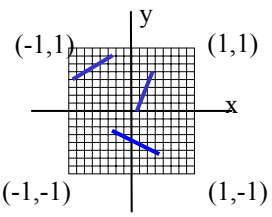


リングOSCの1段辺りの遅延時間のシステマテック成分の測定結果

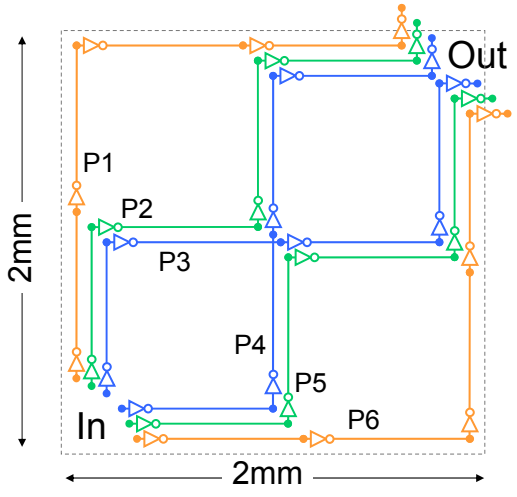
RingOSC: 2in-NAND, wire load, 7-stage.

観測結果:
実測値は、1～2次のランダム曲面(距離相関)とよく一致する

システマティック成分のリングOSCばらつきとPoly抵抗ばらつきの相関(≒0.8)は大きく、ばらつき要因はチップ内のLgのチップ内ばらつきによるものと推定される[3]



ベンチマーク回路



Wire
0.12pF/mm

Inverter
Wn/Ln = 1.8um/65nm
Wp/Lp = 3.0um/65nm

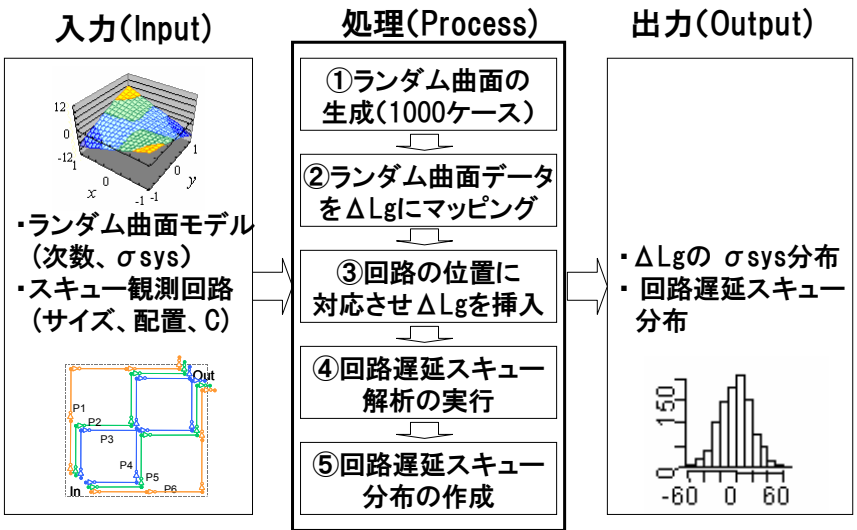
6種のパス
組み合わせは15通り
サイズ: 2mm□

% σ_{sys}(L)=2%

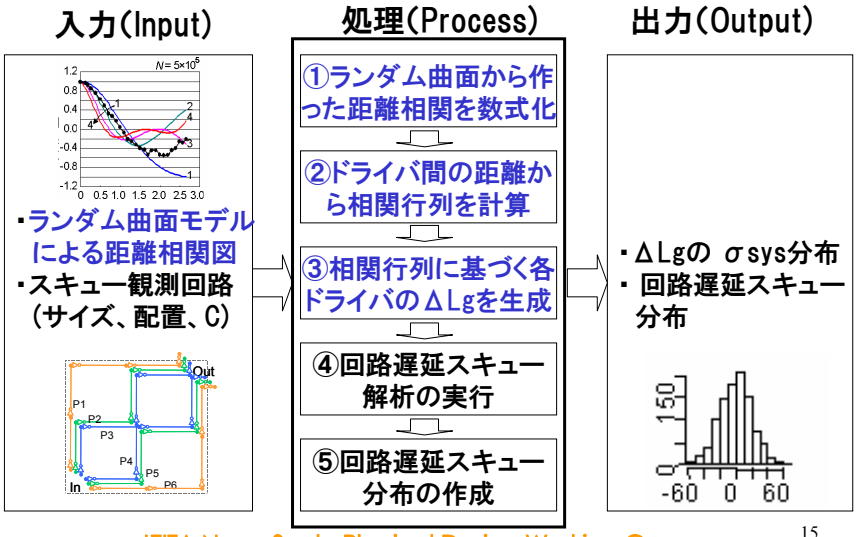
Public MOSモデル [9] および解析条件

Tr	Model	65nm, Typical, BSIM4[9]
	NMOS	Lg_{typ}=65nm, W=1.8um, Tox=1.84nm Vtn=0.47V
	PMOS	Lg_{typ}=65nm, W=3.0um, Tox=1.85nm Vtp=−0.46V
Vdd		1.2V
Temp.		25℃
Cl		0.12pF/mm
解析方法		モンテカルロ

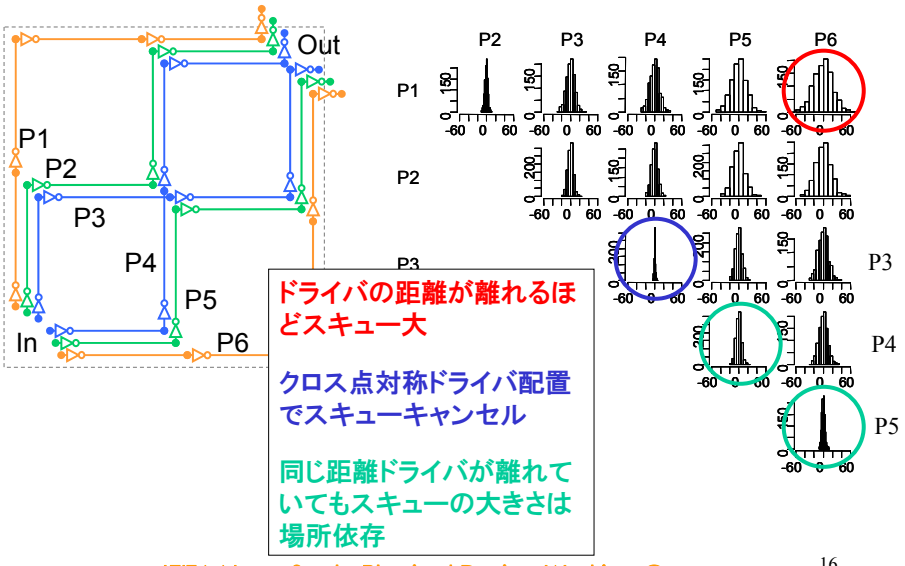
ランダム曲面を使った回路遅延スキュー解析：モンテカルロ解析フロー



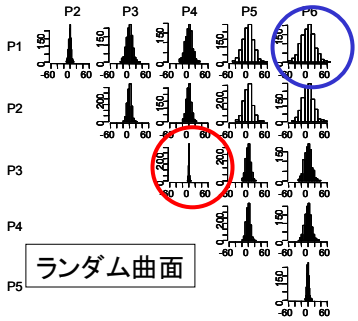
距離相関モデルによる回路遅延スキュー解析：モンテカルロ解析フロー

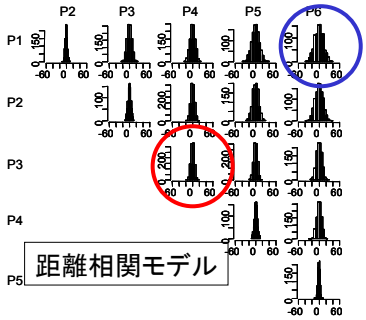


ランダム曲面(1次)モンテカルロ解析



1次のランダム曲面と距離相関モデルの比較

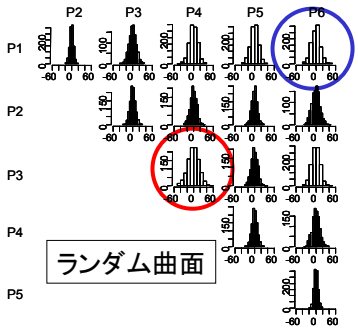


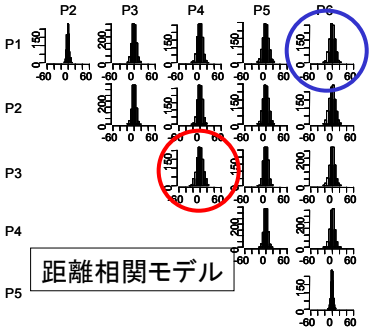


誤差[ps]：距離相関モデルσ － ランダム曲面σ

	P1	P2	P3	P4	P5	P6
P1		−0.3	−2.6	−2.9	−5.4	−7.4
P2	−0.3		−1.2	−1.4	−3.9	−5.4
P3	−2.6	−1.2		3.9	−1.4	−2.9
P4	−2.9	−1.4	3.9		−1.2	−2.6
P5	−5.4	−3.9	−1.4	−1.2		−0.3
P6	−7.4	−5.4	−2.9	−2.6	−0.3	

2次のランダム曲面と距離相関モデルの比較





誤差[ps]：距離相関モデルσ － ランダム曲面σ

	P1	P2	P3	P4	P5	P6
P1		−2.6	−5.0	−5.6	−3.9	−5.4
P2	−2.6		−3.3	−3.7	−0.8	−3.8
P3	−5.0	−3.3		−7.9	−3.7	−5.3
P4	−5.6	−3.7	−7.9		−3.3	−5.2
P5	−3.9	−0.8	−3.7	−3.3		−2.6
P6	−5.4	−3.8	−5.3	−5.2	−2.6	

解析結果の分析

- ランダム曲面によるスキュー解析：
  - 回路スキュー最大値はσ=20.1ps(パス遅延相対値9.0%)レベルで大きく、設計で無視できない
  - 配線をクロスした点対称ドライバ配置でスキューを小さく抑えられる事を指摘(アナログ回路でのコモンセントロイドレイアウト効果に相当)
  - スキュー値はドライバ間距離依存だけでなく、チップ内位置依存を考慮する必要がある
- 距離相関によるシステマティックばらつきのモデルの評価：
  - 距離相関モデルでは殆どのケースでスキューを小さく計算する(よく見積もってしまう)
  - 配線をクロスした点対称ドライバ配置で回路スキューをキャンセルできることを示したが、従来の距離相関モデルでは解析できない
  - 回路遅延スキューの位置依存は距離相関モデルで誤差が大きい
  - ランダム曲面の次数が高くなると、距離相関モデルの誤差が大きくなる傾向がある

まとめ

- チップ内システマティックばらつきが回路スキューに与える影響は最大値はσ=20.1ps(パス遅延相対値9.0%)レベルで非常に大きくなることを示した
- 距離相関によるシステマティックばらつきのモデルと比較した結果、距離相関を用いる方法では、約230psのパス遅延に対して、回路スキュー標準偏差として7.4psの誤差が生じることがわかった
- 配線をクロスした点対称ドライバ配置でスキューを小さく抑えられることを初めて定量的に確認した
- 本開発を通して、チップ内のシステマティックなばらつきを定量的に評価する手法を提供した。今後、設計の場での活用に努力する

参考文献

[1] U.Fassnacht, “A robust ASIC design and IP integration methodology for 65nm and beyond,” ICCAD Workshop 2004.

[2] 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.1433-1439, Oct. 1989.

[3] M.Aoki, S.Ohkawa, and H.Masuda, “Design guidelines and process quality improvement for treatment of device variations in an LSI chip,” IEICE Trans. Electron, vol.E88-C, pp.788-795, May 2005.

[4] A.Agarwal, D.Blaauw, V.Zolotov, S.Sundareswaran, M.Zhao, K.Gala, and R.Panda, “Statistical delay computation considering spatial correlations,” Proc. ASP-DAC, pp.271-276, Jan. 2003.

[5] EDSフェアー '06展示;「物理設計技術」(株)半導体理工学研究センター, 2006年1月.

[6] 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. Semiconductor Manufacturing, vol.17, no.2, pp.155-165, May 2004.

[7] S.Ohkawa, M.Aoki, and H.Masuda, “A novel expression of spatial correlation with random curved surface,” IEICE Technical Report, vol.106, no.425, pp.91-96, Dec. 2006.

[8] S.Ohkawa, H.Masuda, and Y.Inoue, “A novel expression of spatial correlation by a random curved surface model and its application to LSI design,” IEICE Trans. Fundamentals, (to be published).

[9] Predictive Technology Model (PTM), http://www.eas.asu.edu/~ptm/, Arizona State University.

[10] Numerical Technologies Random Generator for Excel (NtRand), http://www.numtech.co.jp/NtRand/#NtRandMultiNorm, Numerical Technologies Incorporated.

4.2 SystemCワーキンググループ 2007年度活動報告

JEITA EDA技術専門委員会

標準化小委員会

SystemCワーキンググループ

JEITA

SystemCワーキンググループメンバー

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）
	小島　智	（NECシステムテクノロジ）
	清水　靖介	（OKI）
	逢坂　孝司	（ケイデンス）
	長尾　文昭	（三洋半導体）
	西園寺　修	（シノプシス）
	柿本　勝	（ソニー ）
	竹村　和祥	（松下電器）
	牧野　潔	（メンター）
	渡邊　政志	（ルネサステクノロジ）
客員	今井　正治	（大阪大学）
		（計13名）

JEITA

目次

 JEITA

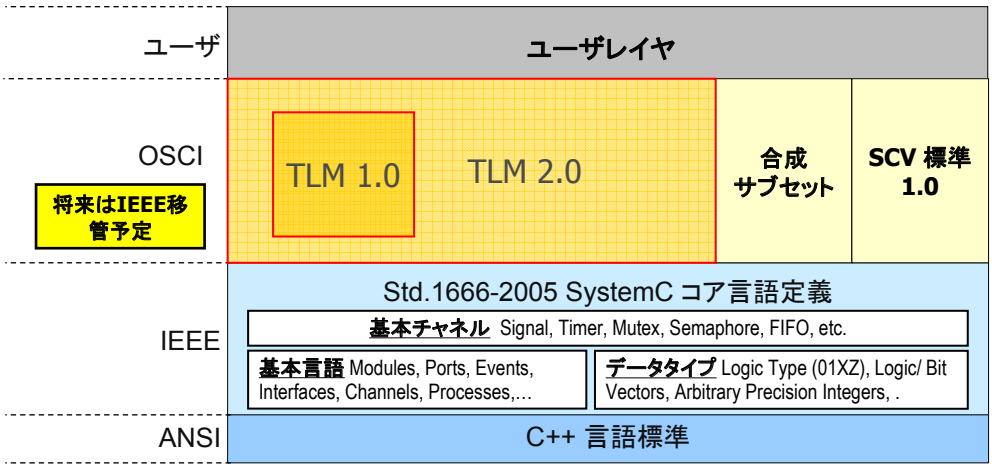
4.2.1 SystemCワーキンググループ概要

 JEITA

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

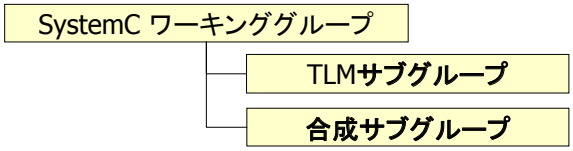


OSCI資料(2007)に加筆

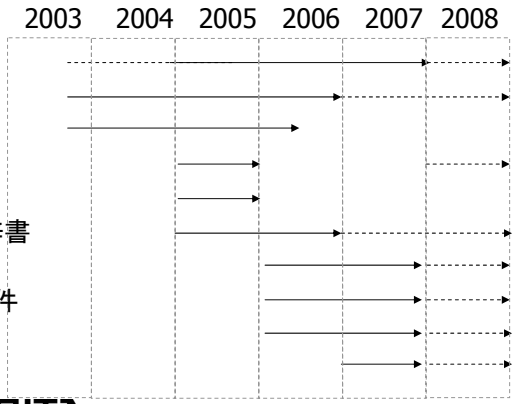
SystemCワーキンググループについて

- 設立と名称変更
  - 2003年10月に、JEITA EDA技術専門委員会 標準化小委員会内にSystemCタスクグループとして設置(SystemVerilogタスクグループと同時)
  - 2007年4月度よりSystemCワーキンググループに名称変更して活動継続
- 目的
  - 日本国内における唯一のSystemCの標準化関連組織として、OSCIやIEEE P1666ワーキンググループと連携しつつ、日本国内の事情・要求事項を取り込むべくSystemCの国際標準化を進めていく。
  - SystemCに関連した調査結果を積極的に情報発信を行うことで、国内普及を図る。これらにより日本の産業界の国際競争力を高めることを目指す。

SystemCワーキンググループの活動



- これまでの歩み
  - SystemCユーザ・フォーラム
  - SystemC動向調査
  - OSCI LRMレビュー
  - IEEE P1666 WG 標準化活動
  - SystemC 2.1調査
  - SystemC-SystemVerilog共通辞書
  - 合成サブセットレビュー
  - 動作合成スタイルガイド構成要件
  - TLM標準化
  - SystemC推奨メソドロジ



これまでの活動内容と主な成果

- SystemC標準化活動
  - IEEE P1666 WGに投票権のあるメンバーとして参加し、SystemC言語仕様書のレビューにて50件以上の改善提案によりIEEE Std. 1666-2005の策定に貢献。(2005年度)
  - OSCI動作合成サブセットdraft、TLM2.0 draft1に対するレビューを実施し、OSCIへフィードバックを提出。
  - TLM 2.0 draft2のレビューを実施し、OSCIへ10件のフィードバックを提出した。(説明不足7件、例題追加要望3件)
- SystemC技術調査
  - 2000年～2005年度における国内外でのSystemCの利用状況について調査を実施した。
  - SystemC 2.1について調査を行い、その特長を日本語で紹介した。(2005年度)
  - 国内外におけるTLM(Transaction Level Modeling)の利用状況調査を実施した。(2006-2007年度)
  - OSCIより公開されている合成サブセットのドキュメントのレビュー及び抄訳の作成を実施した。(2006年度)
  - SystemC動作合成ガイドライン構成要件の検討と、ガイド準拠のドキュメントのレビューを行った。(2007年度)
  - SystemC推奨設計メソドロジの検討を行い、合成編について審議完了した。(2007年度)
- SystemC普及活動
  - SystemCを利用した設計の普及をはかるため、SystemCユーザフォーラムを開催し積極的に情報発信を行った。
  - SystemC Japan 2007にてこれまでの活動成果を講演した。(2007年度)
  - JEITA EDA-TCのWEBページを活用し、これまでの活動成果を一挙掲載した。(2007年度)
    http://eda.ics.es.osaka-u.ac.jp/jeita/eda/index-jp.html

JEITA

2007年度報告資料について

4.2.2 TLM 2.0 要求仕様の日本語要約
- Requirements specification for TLM 2.0 Ver. 1.1の日本語要約

4.2.3 TLM 2.0 用語集
- OSCI TLM 2.0 Glossary Ver. 0.2の日本語訳

4.2.4 TLM 2.0 draft2ユーザマニュアルの日本語要約
- OSCI TLM2.0 User Manual Ver. 1.1.0の日本語要約

4.2.5 SystemC推奨設計メソドロジ 合成編
- 推奨設計メソドロジとしてSystemCの効果的な使い方を提案

4.2.6 SystemCユーザ・フォーラム2008開催報告
- SystemCユーザ・フォーラムの開催報告とアンケート結果

JEITA

4.2.2 TLM 2.0 要求仕様の日本語要約

本節は、OSCI の Requirements specification for TLM 2.0 Ver. 1.1 を日本語要約したものである。

JEITA SystemC-WG によるまとめ

1. ユースケースと必要条件

	2.0 対象	ユースケースの必要条件	プラットフォームの必要条件
SW アプリケーション開発	○	実 SW を修正なしで実行 SW デバッガのサポート	レジスタ、エンディアン等 SW 実行に関わる HW 機能を正確に実装、高速性
SW の性能解析	○	実 SW を修正なしで実行 SW プロファイリング情報の抽出	上記（スピードは劣ってよい）＋精度（シリコン比±10%）、性能解析機能
SW アーキテクチャ解析	×	SW/HW とも実際のものは不要 性能データの抽出、設定変更の容易性	HW に非依存で SW アプリケーションを表現するメカニズム、解析機能
HW アーキテクチャ解析	○	汎用 HW モデルと SW 負荷再現の仕組み、性能データの抽出、設定変更の容易性	プロトコル非依存の HW モデルとアプリの負荷モデル、解析機能
HW 性能検証	×	高精度な ISS、TLM と RTL（アクセラレータ、エミュレータを含む）の混在検証	構成・設定変更の容易性、高精度（±1%以内）、バス、サブシステムの性能解析機能
HW 詳細化と実装	×	動作合成サブセットと TLM 標準に準拠（切り替え、Untimed モデルの再利用のためのコーディングスタイル）	ピンレベルモデルとの通信、クロックの定義
HW 機能検証	○	TLM と RTL（アクセラレータ、エミュレータを含む）の混在検証、代表的な SW が実行可能	レジスタ、エンディアン等を正確に実装、割り込みタイミングがほぼ正確、プロファイリング機能、RTLX100 倍のスピード

2. TLM2.0 Draft1 と Draft2 の違い

	TLM2.0 draft1 （TLM-2006-11-29）	TLM2.0 draft2 （TLM-2007-11-29）
I/F	（TLM1.0 使用） PV tlm_transport_if PVT tlm_(non_)blocking_put/get_if	（TLM1.0 未使用） UT tlm_b_fw/bw_transport_if LT tlm_nb_fw/bw_transport_if AT tlm_nb_fw/bw_transport_if
データ構造	Data,Address の template のペア（例:int,int） 又は、 tlm_request,tlm_response の別々の構造体	tlm_generic_payload のみ Extension で拡張可能
データタイプ	data,address とも template で指定 PASS_BY_VALUE/PASS_BY_POINTER の両方サポート	data: char* address: sc_dt::uint64 PASS_BY_POINTER のみ

本文の要約

序文

- TLM ワーキンググループの目的
  - SoC 設計における仕様検討、シミュレーション、検証、実装、評価を目的として、RTL より高い抽象度で相互接続可能なモデリング、通信手法を実現するための技術と SystemC の基本クラスライブラリを開発すること
- TLM2.0 の目的
  - メモリマップドバス（以降 MMB）を利用した SoC プラットフォームにおいて、モデルの相互利用性を実現すること
  - 任意の IP やシステムを開発するための基本プリミティブを提供すること
- 対象アプリケーション
  - メモリマップドバスを利用したシステム
    - Loosely-timed（以降 LT）システム（PV）
    - Approximately-timed（以降 AT）システム（PVT）
- 提供物
  - ユーザ定義テンプレートを基本とした汎用 TLM API
  - ユーザ定義テンプレート用の低レベルの推奨データタイプ
  - TLM API 用のデータ構造
- 汎用 MMB について
  - 汎用 MMB は LT でモデリングした機能レベルにおいては十分であるかもしれないが、必ずしも特定のプロトコルの機能の詳細を満たさない場合もある。この場合は、相互利用性は保証されず、ブリッジが必要となる。

Part1：TLM のユースケースと必要条件

1. OSCI TLM の対象外のユースケース

- 要求仕様のモデリング
  → UML で扱う
- アルゴリズム開発とアルゴリズムの性能解析
  → ドメイン依存の計算モデル（MoC）で扱う（KPN、SDF、CSP,同期型言語等）
- 論理合成と動作合成
  → OSCI 合成ワーキンググループで検討中

2. OSCI TLM の対象となるユースケース

2.1. SW アプリケーション開発と HW/SW の統合

- ユースケース： SW 実装フェーズ
  - TLM の HW プラットフォームを SW アプリケーション（ドライバ、OS、アプリケーションフレームワーク等）の開発に利用
  - 実際の組み込み SW をターゲットプラットフォーム上で開発、デバッグ
- ユースケースの必要条件
  - 実際の SW を修正なしで使えること
  - SW デバッガが利用できること
  - 超高速に実行できること
- プラットフォームモデルの必要条件
  - 高速である（OS を数秒内にブート可能）
  - レジスタが正確に実装されていること
  - 実 SW の実行に必要な HW の機能が完全に実装されていること
  - LT （OS ブートに必要なタイマ割り込みを実現できる程度）
  - エンディアンが正確に反映されていること

2.2. SW の性能解析

- ユースケース：SW 検証フェーズ
  - TLM の HW プラットフォームと実際の組み込み SW を使用し、 SW アプリケーションの性能を計測する

■ ユースケースの必要条件
  - 実際の SW を修正なしで使えること
  - SW のプロファイリング、最適化のための性能情報が抽出できること
  - 2.1 ほどのスピードは不要だが、インタラクティブモードで大量の SW 実行ができる程度のスピードは必要
■ プラットフォームモデルの必要条件
  - レジスタが正確に実装されていること
  - 実 SW の実行に必要な HW の機能が完全に実装されていること
  - 性能解析データを取れる程度の精度 （シリコン比±10%程度）
  - 解析機能
  - LT と AT を、実行時に切り替えるスイッチ

2.3. SW アーキテクチャ解析（TLM2.x の対象外）

■ ユースケース：初期 SW 仕様検討フェーズ
  - TLM の HW プラットフォームと SW の負荷モデルを利用し、SW の分割や HW へのマッピングを検討する
■ ユースケースの必要条件
  - SW は負荷モデル、HW はリソースモデルで実行
  - アプリケーションの分割やリソースマッピングの検討のために、適切な精度で性能データを抽出できる
  - 探索時間の短縮のため、高い柔軟性、設定変更の容易性が必要
■ プラットフォームモデルの必要条件
  - HW に非依存で SW アプリケーションを表現するメカニズム
  - AT プロセッサモデルか、負荷モデルを実行可能な AT リソースモデル
  - AT の相互接続用モデルと周辺モデル
  - 解析機能

2.4. HW アーキテクチャ解析

■ ユースケース：初期 HW 仕様検討フェーズ
  - TLM の HW プラットフォームと SW の負荷モデルを利用し、HW のリソースの探索を行う（相互接続、メモリサブシステム、高バンド幅アクセラレータ）
■ ユースケースの必要条件
  - 汎用的な HW のリソースモデルを簡単に構築すること
  - アプリケーションの分割やリソースマッピングの検討のために、適切な精度で性能データを抽出できる
  - 探索時間の短縮のため、高い柔軟性、設定変更の容易性が必要
  - アプリケーションの負荷を再現する効果的な仕組み
■ プラットフォームモデルの必要条件
  - プロトコル非依存の HW モデル
  - 適切なタイミング予測結果を得るため、代表的なプロトコルの機能を表現すること（latency、バンド幅、パイプライン、スレッド化）
  - AT の相互接続用モデルと周辺モデル
  - アプリケーションの負荷モデル（トラフィック生成、タスクグラフ、実 SW）
  - 解析機能

2.5. HW 性能検証（TLM2.x の対象外）

■ ユースケース：HW 性能検証フェーズ
  - 実際の SW を利用して、HW プラットフォーム全体または一部の性能を検討する
■ ユースケースの必要条件
  - トランザクションの発生タイミングが正しいプロセッサモデル（ISS）
  - TLM と RTL の混在 （タイミングをバックアノテートした PVT モデル、RTL、RTL アクセラレータ、RTL エミュレータ）
■ プラットフォームモデルの必要条件
  - 設定変更が容易で、トランザクションの発生タイミングが正しい HW モデル
    - レジスタが正確に実装されていること
    - バス、サブシステムの性能に関する情報の取得、可視化が可能

- 適切な性能情報が得られる精度（±1%以内）
- 性能の最適化を検討するための柔軟性
- シミュレーション速度よりは、精度と設定変更容易性が優先される

2.6. HW 詳細化と実装（TLM2.x の対象外）

■ ユースケースの説明：ピンレベルの動作モデルの検証
- 動作合成入力モデルの検証
- 想定するプラットフォーム上の通信方法
  - 全ての通信は 1 つ以上のバス I/F を介して行う
  - 1 つ以上のバス I/F とピンレベル接続の組み合わせで行う

■ ユースケースの必要条件
- OSCI 動作合成標準と、TLM 標準の両方に準拠すること
- Untimed シミュレーションと動作合成の両方に、最小限の変更で再利用できること
- TLM ポートとピンレベルポートとの切り替えのため、コーディングスタイルが提供されること

■ プラットフォームモデルの必要条件
- TLM によるバス接続と、TLM またはピンにより他のモジュールと通信可能
- TLM をバイパスして高速化を図る仕組みが標準化されれば、Untimed モデルにおいて、バス I/F を経由するモデルとバスをバイパスするモードの切り替えが可能
- プロセスを持つ Untimed モデルが構築でき、クロックを定義できること

2.7. HW 機能検証

■ ユースケースの説明
- 代表的な SW と、単体テストに十分な範囲の HW を含む TLM モデルを利用し、 HW の実装が、システムの要求仕様を満足することを確認する

■ ユースケースの必要条件
- TLM と RTL の混在 （タイミングをバックアノテートした PVT モデル、RTL、RTL アクセラレータ、RTL エミュレータ、RTL テストベンチ）
- SW は完璧なものではなく、タイミング精度がある場合もあれば、全くない場合がありうる

■ プラットフォームモデルの必要条件
- レジスタが正確に実装されていること（SW がアクセスする箇所のみ）
- 割り込みがおおよそ期待するタイミングで発生できること
- バス性能の抽出、可視化、SW 性能のプロファイリングが可能
- エンディアンが正確に反映されていること
- タイミングをランダムに変更可能であること
- RTL の 100 倍以上のスピード（柔軟性が優先）

Part2：実装のコンセプトと要求仕様

3. SW 開発と SW 性能解析を目的とした、TLM 実装に対する要求仕様

3.1. 相互利用性

- ソフトウエア開発のために必要となるモデルを、OSCI の仕様に基づいて開発するだけで、相互に運用できるようにすることが目的である

3.1.1. 相互利用性のための要求仕様

3.1.1.1. MMB 用の相互運用可能な API

- MMB を利用した SoC プラットフォームにおいて、LT の TLM モデルの API を定義することを目的とする
  - ほとんどの MMB をサポートする単一のファンクションレベルの API を用意する
  - 直接 API と定義されたデータ構造を使用する
  - API を継承したモデルを利用する
  - 独自の TLM-API と、TLM2.0 間のアダプタを用意することにより相互利用性を確保する

3.1.1.2. LT のための汎用データ構造

- トランザクションのための汎用データ構造を定義する

- ソフトウエア開発用
- read、write、bus locking、 マスタースレーブ接続等の一般的な MMB プロトコルのための機能をサポート
- TLM 2.0 #Draft1 を議論のスタートポイントとする

3.1.1.2.1. ランタイムに動的にデータのサイズを変更すること

- 訳注 例：AMBA の HSIZE

3.1.1.2.2. データ構造の中の属性の意味について完全に定義

- マスタとスレーブ間において属性の取り扱いをどのように解釈するかを厳密に定義
- 訳注： 入れ物だけを用意するわけではない

3.1.1.2.3. なるべく簡単な構造にする

- 簡単なマスタ/スレーブの場合には、全ての属性を使用しなくてもいいようにする
- 訳注:バースト転送をサポートしていないバスのために、バーストサイズ=1 を定義しなくてもいいようにするということ

3.1.1.3. モジュール間のアダプタはなるべく避ける

- できれば、異なるプロトコル間のアダプタを使用しなくてもいいようにする
  - AXI – OCP
  - マスタとスレーブ間で同じ機能だけを使用する場合の話（片方がバーストサポートしてなければ、仕方がない）
- 属性の定義だけでは不十分で、汎用 MMB データ構造（payload）から継承された新しいクラスを用意して必要な拡張をできるようにしておく
- 拡張されたプロトコル間では相互利用性は確保できないが、その場合にはアダプタを用いる
- 汎用 MMB プロトコルのセマンティックス、属性、デフォルト値は、アダプタの使用をなるべく避ける方向で開発されなければならない

3.1.1.4. 互換性のないモデルを接続できないようにする

- 互換性のあるモデルだけを接続できるようにする
- そうでない場合には、コンパイル時か実行時にエラーを出すようにする
- トランザクションのデータ構造を定義する際には、アダプタの使用をなるべく避けるようにする

3.1.1.5. AT との互換性

- AT と、LT の間で同じデータを使用する
- ただし、LT だけのモジュールも作成できるようにする
- 全てのモジュールが AT と LT の両方をサポートしなければならないわけではない
- シミュレーション時に、動的に LT と AT を変更できるようにする
- LT→AT は必須
- LT で OS をブートして、AT に切り替える

3.1.1.6. Endian

- Big-Endian と Little-Endian 間で接続できるようにする
- Endian が混在したプラットフォームのサポート

3.1.1.7. Accurate Timed との互換性

- なんらかの Accurate Timed レベルのバスを接続したときのシミュレーションオーバヘッドをなるべく少なくなるような方法を考える
- ただし、これを実現するのはそういった API サンプルを用意することができないので難しい何らかの方法を検討するべき

3.1.2. 相互利用性を保証しないケース

3.1.2.1. ブリッジが無い場合の相互利用性

- 互換性のないモデルを接続できないような API にすることに合意した
- 互換性のない任意のモデルを接続できてしまうような不完全な API では、危険すぎるため

3.1.3. 検討された実装方法オプション

3.1.3.1. 一般的なトランザクションの属性

- OCP、PCI、AXI 等の実際の MMB にある機能のサブセットをサポートする汎用 MMB とする
- 一般的に、現実の MMB プロトコルは基本的に同一の機能の別の実装にしかすぎない
- 属性の例としては、バイトイネーブル、Endian 等である

3.1.3.2. テンプレートを使用しないデータタイプ

- テンプレートを使用しない
- データ　：unsigned char アレイ
- アドレス ：uint64
    - 訳注： 2.0 draft1 との大きな違い。テンプレートを使用すると、相互利用性がなくなってしまうため。

3.1.3.3. 双方向の transport API

- 高速な機能モデル作成のためにこれを採用した
- Transport では、データ構造へのポインタを使う。ただし、オーナーシップを明確に定義する。
- 訳注：TLM 1.0 では安全性のために、データ渡しを使用。TLM 2.0 Draft1 では、データ渡しとポインタ渡しの両方を使用。Draft2 ではポインタ渡しのみを使用

3.1.3.4. データのサイズを定義する

- 相互利用性のためには無いほうがいいが、3.1.1.4 の互換性チェックのためには必要になってしまう

3.1.4. 却下された実装方法オプション

- 最小限の属性+拡張機能という手法は却下
- スピードが遅くなる
- 危険なトランザクションへの対応ができない
- 拡張メカニズムは提供しない。その代わり、C++の継承を利用してデータ構造を拡張する例を提示するにとどめる。
    - 訳注：delay の拡張手段は提供される。ここ（４章）で言っているのは、データ構造の話だと思います（JEITA）

3.2. タイミング精度

3.2.1. Timing 精度に関する要求

3.2.1.1. SW 開発向けの TLM プラットフォームは、LT でなければならない

- システムの他のモジュールとの関連において、タイミング情報を軽く保持して、システムを予期することによりシステムを先に進められることが必要である
- LT の MMB コンポーネントを用いることによって、OS をブートできるようにするための必要最低限のタイミング情報を持つ必要がある

3.2.1.2. モデル間の同期

- 同期メカニズムは、高速に複雑なシステムモデリングに対応できる必要がある
- 同期メカニズムは、基本的にはマスタ側によって調整される
- モデル同期は、インタラプトの取り扱いについても取り決めなければならない

3.2.2. 却下されたタイミング精度の要求仕様

- タイミングパラメーターを Transport 関数の signature（引数）に与える
- タイミング情報は、sc_time 型にて与える

3.2.3. 検討された実装方法オプション

- タイミングパラメーターを Transport 関数に渡すデータ構造のメンバーに与える
- タイミング情報は、unsigned int 型にする

3.2.4. 却下された実装方法オプション

- なし

3.3. 実行速度

3.3.1. 実行速度に関する要求

- 実行速度はモデルに依存するが、常に精度とのトレードオフになる

3.3.1.1. LT TLM-API における実行速度の要求

- HW/SW インテグレーションのための SW アプリケーション開発　：50MT/sec
- SW パフォーマンス解析　：10MT/sec
  MT：Mege Transaction
- 中程度の複雑さのシステムではこのくらいが実現できる程度の TLM API でないといけない

3.3.1.2. ミックスモードにおけるスピード

- LT 精度のバスと AT 精度のバスに接続したときのオーバヘッドは最小限にするべきである

3.3.2. 検討された実装方法の選択肢（却下された）

- 高速化を求めると他の物が犠牲になる
    - 安全性→ポインタ渡し、Direct Memory Interface（DMI）によって解決
    - 相互利用性→ テンポラルデカップリングによって解決
- 高速性と相互利用性を両立するためには、それを実現する（新しい）標準 API が必要となる

3.3.2.1. ポインタ渡し（Pass by Pointer）

- 訳注: 遅くなるためデータ渡し（Pass by value）は使用しない

3.3.2.2. DMI（検討課題）

- マスタ側において、必要なメモリ内の情報を直接取得する手段の提供

3.3.2.2.1. Direct Memory のポインタの無効化

- メモリマップが変更されたときなど、DMI によって取得したメモリのポインタを無効にする手段が必要

3.3.2.2.2. この機能はオプションとする

3.3.2.3. 抽象レベル

- バス幅、バースト転送手法の詳細など、タイミングにのみ影響し、トランザクションの機能には関係ない情報は排除する

3.3.2.4. テンポラル：デカップリング

- 高速性を実現するために、マスタにおいてテンポラルデカップリング手法を採用する
- これは、マスタの時間の進行を止めるのではなく、一つのマスタの実行を一時的に先に進めておくということである
    - 訳注：インタラプトが発生するのを待つためだと思います（JEITA）
- 先に進める時間は、コンフィギャーできる必要がある
    - 実行スピードと正確性のバランスを取るため
    - デバッグ時には、最小値にする必要もある

3.3.2.4.3. テンポラル：デカップリングは、PV の同期手法に則らないといけない

3.3.2.4.4. SystemC カーネルとマスタの間にて進める時間（slack）を制御できるようにする必要がある

3.3.2.4.5. スレーブ側にて、機能を実行する前に、SystemC カーネルに対して同期のトリガを欠ける機能が必要（同期 On demand）

3.3.2.4.6. スレーブ側にて、機能を実行した後に、SystemC カーネルに対して同期のトリガを欠ける機能が必要（同期 On demand）

3.3.2.4.7. テンポラル：デカップリングを使用したマスタと使用していないマスタの混在ができなければいけない

3.3.3. 拡張性

- 汎用 MMB を拡張する際には、常に高速性を実現できなければいけない

3.4. 完全性

3.4.1. 完全性への要求

3.4.1.1. Non-intrusive（他に影響を与えない）トランザクション

- マスタにデバッガを接続し、他に影響を与えないデバッグ用のトランザクションをシステムにリクエストできるようにしなければいけない
- このトランザクションは、すでに定義されたデータ構造で影響を与えないものに対してのみ対応する
- さらに拡張したデバッグ情報の転送についてはここでは定義しない

3.4.1.2. Visibility

- API は、解析用ポートをサポートしなければいけない

3.4.1.2.1. 解析メカニズムはユーザの視点において、他に影響を与えるものではいけない

3.4.1.2.2. 解析メカニズムは、機能カバレージやスコアボーディング等にとって有用なものである必要がある

3.4.1.2.3. 解析メカニズムは、トランザクションレコーディング、パフォーマンス解析、トランザクションレベルアサーションにとって有用なものである必要がある

- 機能カバレージのためには、トランザクションそのものだけでなく、開始時刻、終了時刻も必要である

3.4.1.3. メモリマップ情報の転送機能

- 2.0 仕様ではなく、2.1 仕様のために調査を開始
- 時には、メモリマップ情報をコンポーネント間で転送できるようにしておく必要がある
- メモリマップ情報の転送は、シミュレーション実行時（runtime）にできるようにする
    - 訳注： Constructor 時ではないということ

3.4.2. 検討された実装オプション

- MMB 転送のデバッグモード
- 解析用のポート

4. アーキテクチャ解析サポートのための TLM 実装要求

4.1. 相互利用性（2.0）

- OSCI ドキュメントだけでプラットフォーム非依存で接続性の高い、アーキテクチャ探索用モデルを開発出来ること

4.1.1. 相互利用性の必要条件

- 汎用的且つ高い接続性の MMB の API
- ブリッジ（プロトコル/機能変換など）は可能な限り使用しない
- 多くの AT MMB で動作可能なトランザクションデータ構造
- AT モデルと LT モデルとの互換性
    - ポインタ渡しの構造体のコピー
    - 可能な限り同じデータ構造
    - On-the-fly 切換え
- AT モデルと Accurate Timed モデルとの互換性（少ないオーバヘッド）

4.1.2. 却下した相互利用性必要条件

- AT TLM API は 100%のサイクル精度である必要は無い
- これはサイクル精度 TLM API の範囲であり TLM 2.x の対象外

4.1.3. 実装において考慮すべき点

- AT TLM API はノンブロッキング転送 API をベースとする
- 64bit int、データは unsigned char*を用いる

4.1.4. 実装において採用されなかった項目

- AT Timed TLM API における（TLM1.0 仕様の）put/get API
- データ構造はデータとアドレス型を含むテンプレート型

4.2. タイミング精度

- LT モデルに比べより多くのチャレンジが求められ、十分なスピードとデザインを決定するに値するほどの精度が要求される

4.2.1. タイミング精度の必要条件

- 各種性能測定対象（レイテンシ、スループット等）の精度が x%となること
- ブリッジ（プロトコル/機能変換など）は可能な限り使用しない
- バースト構造をモデリングできること
- 多重 Outstanding（Split）トランザクションをモデリングできること
- Out-of-order トランザクションをモデリングできること

4.2.2. 却下したタイミング精度の必要条件

- 高速化を考慮し、以下の要求は対象外
    - AHB のアドレス/データフェーズのパイプライン
    - OCP のデータハンドシェイクパイプライン

4.2.3. 実装で考慮すべき項目

- AT TLM API は高い精度を保証するため、トランザクション実行時に以下のようなタイミングが必要
    - バスで初期化が開始されたタイミング
    - バスで初期化が完了したタイミング

- バスで初めのリクエストがアサートしたタイミング
- バスで最後のリクエストがアサートしたタイミング
- バスで最初のレスポンスがアサートしたタイミング
- バスで最後のレスポンスが受理されたタイミング

4.2.4. 実装において採用されなかった項目
- なし

4.3. スピード
- 様々なユースケースで利用し易くなる TLM モデルの処理速度（2.0）
  - SW アーキテクチャ解析　　：1MT/sec
  - HW アーキテクチャ解析　　：1MT/sec

4.4. 完成度
- LT の一般的な MMB の全機能をサポート
- パフォーマンス解析用測定機能
- AT から LT へモード切換が可能（not 2.0）

5. モデリング効率（2.0）
5.1. 概念的な簡易性
- ワーキンググループは根底の概念が明確でそれを簡単に説明できるべき

5.2. コミュニケーションの一貫性
- モジュール間通信数（HW におけるポート数）ha 出来る限り少なくする

5.3. 利用しやすさ
5.4. コンパクトなコード、表現
5.5. 安全とエラー耐性

6. 一般的 TLM API（2.0）
- ユーザ定義テンプレートをベースとする一般的な TLM API

7. 推奨する低レベルデータ型（2.0）
- ユーザ定義テンプレートが必要な時に使われる低レベルデータ型のリスト

8. "Proof-of-Concept"実装の品質
8.1. 全ソースコードに copyright 注意書きを付ける
8.2. ヘッダファイルと例題のコードには意味のあるコメントを付ける
8.3. doxygen スタイルのコメントを利用する（必須ではない）
8.4. それぞれの機能に関してリグレッションテストを用意する
8.5. 例題とリグレッションは Purify で問題ないようにする
8.6. コンパイル時のエラーとワーニングは対策する
8.7. 利用者が十分に機能を理解できるよう例題（デモ）を用意する
8.8. ベンチマークの例を用意する
8.9. 一貫して UNIX ファイルフォーマットを使用する
8.10. namespace をヘッダファイルの中で開かない
8.11. 無駄なファイルは成果物に含めない
8.12. 全てのヘッダファイルで#include のガード（二重読み込み?）を設ける

9. ドキュメント（2.0）

9.1. LRM は通常のリリースで必要

9.2. 全主要 API の機能を説明する技術的 Whitepaper はコミュニティレビューパッケージに必要

9.3. （UML 付き）設計と原理的説明

9.4. 例題とウォークスルー

10. ポーティング

10.1. コミュニティレビューパッケージは Windows と Linux で動作すべき

10.2. 通常リリースは OSCI カーネルでサポートしている OS/コンパイラでポーティングすべき

10.3. OSCI SystemC 2.2 ライブラリでコンパイル出来るべき

11. OSCI TLM 1.0 IP との互換性（2.0）

12. off-chip 通信のモデリング障害を作ってはならない（2.0）

13. マルチポートターゲットのサポート（2.0）

14. 階層的モデルのサポート（2.0）

15. LT スタイルで書かれたモデルで AT の考えを混ぜない（2.0）

16. PV と PVT 両方の体系的なサポートは不要（2.0）

17. ディストリビューションはオブジェクトコードとヘッダファイルだけでも配布可能なようにする（2.0）

18. 相互利用性要求の明確化

18.1. 標準的な I/F

- Untimed/LT/AT に関して標準的な I/F クラス、転送/タイミングメカニズム
- 指定されたペイロードの詳細な独立性
- 汎用ペイロードとユーザ定義ペイロード両方の利用可能性

18.2. 汎用ペイロード

- クラスとドキュメントからアダプタ無しで相互利用性のあるモデルを開発できるよう、正確なセマンティックスの汎用ペイロードを定義すべき
- 汎用ペイロードに十分性は無いため、必ずしも指定された MMB プロトコル間の相互利用性を与えるものではない
- TLM2.0 Draft1 にある属性は利用されるべき

18.3. 拡張メカニズム

- 特定のプロトコルに対応できるよう汎用ペイロードの拡張メカニズムを定義すべき
- 継承や組合せなどを用いて拡張されたモジュールがアダプタなどを介して接続できるよう、拡張に関するガイドライン（推奨/非推奨）を与えるべき
- 特定のプロトコルを表現するための拡張メカニズムと無視可能な属性を与えるメカニズムの区別

18.4. 特定のプロトコル

- AXI や PLB など特定プロトコルのモデリングは OSCI ではなくオーナの責任とする

18.5. ガイドライン

- TLM2.x では、共通バス属性をモデル化するために与えられたメカニズムの使用方法についてのガイドは与えるが、モデリングするベストなアプローチを与えるガイドは含めない

18.6. 汎用ペイロードとアダプタ

- 汎用ペイロードを使用した TLM2.0 のコア I/F を使っている限りアダプタ無しで接続できる
- 特定のプロトコルが汎用ペイロードの属性のみ使用する場合を除き、アダプタ無しで接続することは要求しない

18.7. 特定プロトコルのための相互利用性

- 特定プロトコル（AXI や PCI など）に関する相互利用性の容易化は OSCI の責任対象としない
- 出来る限り汎用ペイロードをベースに、特定プロトコルのモデルを開発すべきである
- 拡張に関するレポジトリは要求仕様に含めない

18.8. AT モデル

- AT モデル用の汎用ペイロードは、Untimed や LT と同じ属性にすべきである

18.9. サイクル精度モデル

- 将来的にはサイクル精度モデルの I/F やメカニズムを定義すべきである

18.10. TLM1.0

- TLM 2.X のインプリが TLM 1.0 や TLM 2.0 Draft1 とそれている場合、互換性や相互利用性を考慮した解決策が必要である
- TLM1.0 標準が引き続き維持されることは想定するが、自動的に互換性があることは想定しない

4.2.3 TLM2.0用語集

TLM2.0 Glossary Ver0.2日本語訳

JEITA

Words	訳語	説明
adaptor	アダプタ	トランザクション・レベル・インタフェースとピン・レベル・インタフェース間、もしくは、異なる抽象度のトランザクション・レベル・インタフェース間を接続するモジュール。異なるタイプの2つのトランザクション・レベル・インタフェースを変換するために用いられる。
approximately timed (AT)	アプロキシメイトリー・タイムド	モデルの外部観測可能な状態とそのモデルの詳細リファレンス・モデルの状態に一対一対応が付き、その対応が状態遷移を保つ（正確なタイミングは保たない）ようなモデリング・スタイル。タイミング精度の程度は定義されない。
attribute (of a transaction)	アトリビュート	トランザクションで運ばれるトランザクション・オブジェクトのメンバーとして実装されるデータ。モデル化するバスもしくはプロトコルに固有なアトリビュート、タイムスタンプ等のシミュレーション・モデルの産物も含む。
bidirectional interface	双方向インタフェース	TLM 1.0のトランザクション・レベル・インタフェース。トランザクション・オブジェクトのペア（リクエストとレスポンス）が、それぞれ反対方向に、単方向インタフェースのルールに従って送信される。各トランザクション・オブジェクトのトランザクション・アトリビュートはトランザクション存続期間中は厳密に読出しのみ許される。 注）ハンドシェークの意味で使われている。
blocking	ブロッキング	wait関数を呼出すことが許されている。ブロッキング関数はシミュレーション時間を消費、または、コンテクスト・スイッチする。したがってメソッド・プロセスから呼出してはいけない。ブロッキング・インタフェースはブロッキング関数のみ定義する。
bridge	ブリッジ	2つのトランザクション・レベル・インタフェース間を接続するモジュール。それぞれはメモリ・マップド・バスもしくは他のプロトコルを表すもので、通常は同じ抽象度のもの。バス・ブリッジは2つのバスを接続するデバイス。コミュニケーション・ブリッジはネットワークのデータ・リンク層のネットワーク・セグメントを接続するデバイス。
caller	呼び出し元関数	ある関数を呼出している関数、プロセス、モジュール。
callee	呼び出し先関数	呼び出し元関数から呼出される関数。
channel	チャネル	1つ以上のインタフェースを実装したクラス、もしくは、そのクラスのインスタンス。チャネルは階層チャネル、プリミティブ・チャネル、そうでなければ、sc_objectクラスを継承することが強く推奨される。チャネルはコミュニケーション機構もしくはプロトコルの定義を隠蔽するのに使われる。（SystemC用語）
core interface	コア・インタフェース	この標準で定義されるトランザクション・レベル・インタフェースで、他のいかなるインタフェースも継承しておらず、トランザクション・タイプをテンプレート・パラメータとして持つ。コア・インタフェースはインタフェース・プロパである。コア・インタフェースは汎用ペイロードAPIとは違う。

 JEITA

Words	訳語	説明
cycle accurate	サイクル精度	モデルの外部境界で、あるサイクルでのモデルの状態の予測ができる、従って、モデルの状態とそれに対応するRTLモデルの外部観測可能な状態との対応付けが毎サイクルできるモデリング・スタイル。ただし、毎サイクルでモデル全体の状態の再評価、全ての外部ピン、内部レジスタの状態を表すことは要求されない。この用語は、サイクルの概念を持つモデルにのみ使われる。
cycle approximate	サイクル近似	モデルの外部観測可能な状態とそのモデルに対応するサイクル精度モデルの状態間に1対1の対応付けが可能なモデル。その対応付けは、状態遷移を保持するが、正確なタイミングは保持しない。タイミング精度の程度は定義されない。この用語は、サイクルの概念を持つモデルにのみ使われる。
cycle count accurate, cycle count accurate at transaction boundaries	サイクル数精度、トランザクション境界でのサイクル数精度	モデルの状態とそのモデルに対応するRTLモデルの外部観測可能な状態との間で、トランザクション境界を表すタイミング・ポイントで状態をサンプルすることで1対1の対応付けが可能なモデリング・スタイル。サイクル数精度モデルは全てのサイクルでサイクル精度である必要はないが、機能的状態とトランザクション境界で定義されるキーとなるタイミング・ポイントでサイクル数が正確に予測されることが求められる。
declaration	宣言	C++プログラムに、ある名前を導入し、C++コンパイラがその名前をいかに解釈すべきかを指定するC++の構成概念。宣言は定義とは限らない。例えば、クラス宣言はクラス名を規定するが、クラス・メンバーは規定しない。一方、関数宣言は、関数のパラメータは規定するが、関数本体は規定しない。(C++用語)
definition	定義	変数、関数、型、テンプレートの完全な規定。例えば、クラス定義はクラス名とクラス・メンバを規定する。関数定義は、関数のパラメータと関数本体を規定する。(C++用語)
generic payload	汎用ペイロード	トランザクション・アトリビュートのセットとメモリ・マップド・バスを介して通信するコンポーネントのアンタイムド、ルーズリー・タイムド、アプロキシメイトリー・タイムド・モデル間のある程度の相互利用を達成するためのセマンティクス。アトリビュートは、全てのモデルに使えるものと近似タイムド・モデルにのみ使えるものとに分けられる。 注)拡張して使用可能。
generic payload API	汎用ペイロードAPI	汎用ペイロードにアクセスするためのクラス・インタフェース。汎用ペイロードAPIはコア・インタフェースとは別物。
initiator	イニシエータ	この標準では正確な技術的定義は与えない。メモリ・マップド・バス、NoC上のトランザクションを開始するコンポーネントを意味するのに使われる。イニシエータは通常マスタであり、マスタは通常イニシエータである。ただし、イニシエータという用語はトランザクションを開始することのできるコンポーネントを意味し、マスタという用語はバスの制御を取ることのできるコンポーネントを意味する。

JEITA

Words	訳語	説明
interface	インタフェース	sc_interfaceを継承したクラス。インタフェース・プロパはインタフェース。オブジェクト指向の意味では、チャネルもインタフェース。ただし、チャネルはインタフェース・プロパではない。(SystemC用語)
Interface Method Call (IMC)	インタフェース・メソッド・コール	インタフェース・メソッドの呼出し。インタフェース・メソッドとはインタフェース内で宣言されたメンバ関数。IMCの枠組みは、メソッド呼出しとメソッドの実装の分離を提供する。例えば、呼び出し元に影響を与えることなく、チャネルの置換えが可能となる。(SystemC用語)
interface proper	インタフェース・プロパ	sc_interfaceを継承した抽象クラス(ただし、sc_objectは継承しない)。インタフェース・プロパはチャネル内で実装されるメソッドを宣言し、それらはポートを介して呼出される。インタフェース・プロパは純粋バーチャル関数の宣言を持つが、通常は、関数定義、データ・メンバは持たない。(SystemC用語)
interoperability	相互利用性	この標準で定義するインタフェースを使うことにより、異なった出所のトランザクション・レベル・モデルで情報を交換できるようになること。この意味は、PVユースケースで共通メモリ・マップド・バス・プロトコルを実装したモデルは、アダプタ無しで相互利用できるはずだということ。さらに、異なったプロトコル、ユースケースのモデルについても、一般にはアダプタが要求されるが、相互利用のための手間の削減を意味する。
lifetime (of an object)	(オブジェクトの)ライフタイム	オブジェクトのライフタイムは、記憶領域が確保され、もしあれば、コンストラクタ呼出が完了した時に始まる。確保されていた記憶領域が開放された時、もしくは、デストラクタが呼出される直前で、オブジェクトのライフタイムは終わる。(C++用語)
lifetime (of a transaction)	(トランザクションの)ライフタイム	トランザクションのライフタイムは、トランザクションが有効になった時に始まり、無効になった時に終わる。トランザクション・オブジェクトはプールまたは再利用される可能性がある為、そのトランザクション・オブジェクトのライフタイムは、対応するトランザクション自身のライフタイムより長い場合がある。例えば、トランザクション・オブジェクトは、TLM1.0インタフェースの複数のput関数呼出しで引数となるケースがある。
loosely timed (LT)	ルーズリー・タイムド	オペレーティング・システムのブートとスレッド間の明示的同期無しで複数スレッドを管理する機構をサポートするに十分な最小の時間情報を表現するモデリング・スタイル。タイマ・モデル、概念的な調停間隔、実行スロット長もルーズリー・タイムド・モデルでモデリングされるかもしれない。ユーザによっては、プロトコルの頑健性をテストするため、ルーズリー・タイムドの記述にランダム遅延を入れる場合があるが、この方法によってモデリング・スタイルの基本的な性質が変わることはない。

JEITA

Words	訳語	説明
master	マスタ	この標準では、明確な技術的定義は無いが、バス・トラフィックを開始するためにメモリ・マップド・バスを制御できるモジュールまたはポート。もしくは、自立的にソフトウェア・スレッドを実行できる、すなわち、他のシステム動作を開始できるコンポーネント。一般には、バス・マスタはイニシエータになりうる。
method	メソッド	クラスの動作を実装した関数。この用語はC++用語のメンバ関数と同義。SystemCでは、メソッドはインタフェース・メソッド・コールの文脈中で使われる。この標準中では、メンバ関数という用語はC++クラスを定義するときに使い、メソッドという用語は形式的でない文脈もしくはインタフェース・メソッド・コールについて議論している際に使う。（SystemC用語）
non-blocking	ノンブロッキング	waitの呼出しが許されない。ノンブロッキング関数はシミュレーション時間の消費、コンテキスト・スイッチ無しでリターンすることが保障されている。そのため、ノンブロッキング関数はスレッド・プロセスもしくはメソッド・プロセスから呼出される。ノンブロッキング・インタフェースはノンブロッキング関数のみ定義する。
object	オブジェクト	記憶領域の区画。あらゆるオブジェクトは型とライフタイムを持つ。定義＊)によって作成されたオブジェクトは名前を持つ、new構文によって作成されたオブジェクトは名前を持たない。（C++用語）＊）definitionの項を参照。
OSCI initiator	OSCIイニシエータ	OSCIトランザクションを開始するモジュール、ポート、またはプロセス。イニシエータは、トランザクション・オブジェクトを初期化する責任があり、トランザクション・オブジェクトのライフタイムの終了時にそのトランザクション・オブジェクトの削除もしくは再使用の責任がある。TLM1.0では、getインタフェースとtransportインタフェースからの応答についてイニシエータという用語は適切でない。その代わりに、呼び出し元関数と呼び出し先関数という用語を用いるべき。
OSCI interconnect component	OSCIインターコネクト・コンポーネント	トランザクション・オブジェクトにアクセスするモジュールまたはプロセス。そのトランザクションに対しては、OSCIイニシエータもしくはOSCIターゲットとして動作する。OSCIインターコネクト・コンポーネントは汎用ペイロードの役割に応じて、トランザクション・オブジェクトのアトリビュートの変更を許可される場合も許可されない場合もある。アービタやルータは、通常OSCIインターコネクト・コンポーネントとしてモデリングされるが、代わりにOSCIターゲットとOSCIイニシエータの両方でそれらをモデルする事もある。
OSCI target	OSCIターゲット	リードまたはライトOSCIトランザクションの最終目的地のモジュール、エクスポートまたはチャネル。ライトでは、データはイニシエータから一つ以上のターゲットにコピーされる。リードでは、データは一つのターゲットからイニシエータにコピーされる。ターゲットはトランザクション・オブジェクトの状態を読み出すか、変更を行う場合がある。
OSCI transaction	OSCIトランザクション	この標準特有の仕組みで定義されたトランザクション。システム・トランザクションは、一つまたは複数のOSCIトランザクションによって表現される。OSCIトランザクションは、一つのトランザクション・オブジェクトによって表される。

Words	訳語	説明
phase	フェーズ	トランザクションのライフタイム中で連続するタイミング・ポイント間の期間。
programmers view (PV)	プログラマーズ・ビュー（PV）	機能的な正確さ、OSのブートとアプリケーション・ソフトウェアが動作するハードウェア・プラットフォームのルーズリー・タイムド・モデルを要求するソフトウェア・プログラマーのユースケース。
slave	スレーブ	この標準では、明確な技術的定義は無いが、他律モジュール、バス・マスタからのコマンドに応答する事が出来るポート。しかしそれ自身は、バス・トラフィックを起こす事が出来ない。一般的にスレーブは、OSCIターゲットとしてモデリングされる。
system initiator	システム・イニシエータ	イニシエータであるが、OSCIイニシエータである必要はない。この用語は、OSCIイニシエータと明確に区別したいときに用いる。
system transaction	システム・トランザクション	トランザクションであるが、OSCIトランザクションである必要はない。この用語は、OSCIトランザクションと明確に区別したいときに用いる。
target	ターゲット	この標準では、明確な技術的定義は無いが、イニシエータが生成したトランザクションに応答できるが、新しいトランザクションを開始することはできないコンポーネント。
temporal decoupling	テンポラル・デカップリング	コンテキストスイッチを抑えるため、複数マスタがシミュレーションの現在時刻に先行して実行できることを許す仕組み。シミュレーション速度が向上する。
timing point	タイミング・ポイント	プロセスが、トランザクションを通じて相互作用する、制御を転送もしくは同期するポイント。タイミング・ポイントは関数呼出し、関数からのリターン、イベント通知で実現される。タイミング・ポイントはトランザクションのフェーズの境界を示す。
transaction	トランザクション	並列動作している二つ以上のプロセス間の相互作用もしくは通信の抽象概念。トランザクションは幾つかのアトリビュートを持ち、それらのアトリビュートは特定の時間範囲内で有効。トランザクションに関連したタイミングは、トランザクションのタイプに依存して、ある特定のタイミング・ポイントに限定される。プロセスは、トランザクションのアトリビュートの読出し、書込み、もしくはアトリビュートにセンシティブにすることができる。トランザクションは単トランザクションもしくは複合トランザクション。

Words	訳語	説明
transaction object	トランザクション・オブジェクト	OSCIトランザクションに含まれるアトリビュートを保持するオブジェクト。
transaction level (TL)	トランザクション・レベル（TL）	並列プロセス間通信がピンレベルの信号動作からトランザクションへ抽象化される抽象度。この用語は時間、構造、動作については、ある特定の抽象度を意味するものではない。
transaction level model, transaction level modeling (TLM)	トランザクション・レベル・モデル、トランザクション・レベル・モデリング（TLM）	前者はトランザクション・レベルのモデル。後者はそのモデルを作成する行為を意味する。トランザクション・レベル・モデルは一般的に、RTLモデルによって使用されるような各ピンやネット上にイベントを設定するようなスタイルとは異なり、関数呼び出しを使用して通信を行う。
transactor	トランザクタ	トランザクション・レベル・インタフェースとピン・レベル・インタフェース、二つ以上のトランザクション・レベル・インタフェース間（しばしば異なる抽象度の）を接続するモジュール。典型的なケースでは、トランザクション・レベル・インタフェースはメモリ・マップド・バスやその他プロトコルを表し、一方のインタフェースは、低抽象度のプロトコルの実装を表す。一つのトランザクタは複数のトランザクション・レベル・インタフェース、ピン・レベル・インタフェースを持つケースもある。
transport interface	トランスポート・インタフェース	この標準で一つしかない双方向のコア・インタフェース。トランスポート・インタフェースは呼び出し元関数から呼び出し先関数へリクエスト・トランザクション・オブジェクトを送り、呼び出し先関数から呼び出し元関数へリクエスト・トランザクション・オブジェクトを返す。
unidirectional interface	単方向インタフェース	トランザクションのアトリビュートがトランザクション・ライフタイムの最初のタイミング・ポイントから最後までの期間で厳密に読出しのみに限定されたTLM1.0のトランザクション・レベル・インタフェース。トランザクション・オブジェクトで表される情報は、呼び出し元関数から呼び出し先関数、または呼び出し先関数から呼び出し元関数へ送られる。void put(const T&t)の場合では、最初のタイミング・ポイントは関数呼び出しによって表される。T valid get(T& t)の場合では、この関数からのリターンによって表される。T get()の場合、厳密に言うと二つのトランザクション・オブジェクトが関わっている。この関数からのリターンは1つ目のトランザクション・オブジェクトの終わりと2つ目のトランザクション・オブジェクトの最初のタイミング・ポイントを表す。

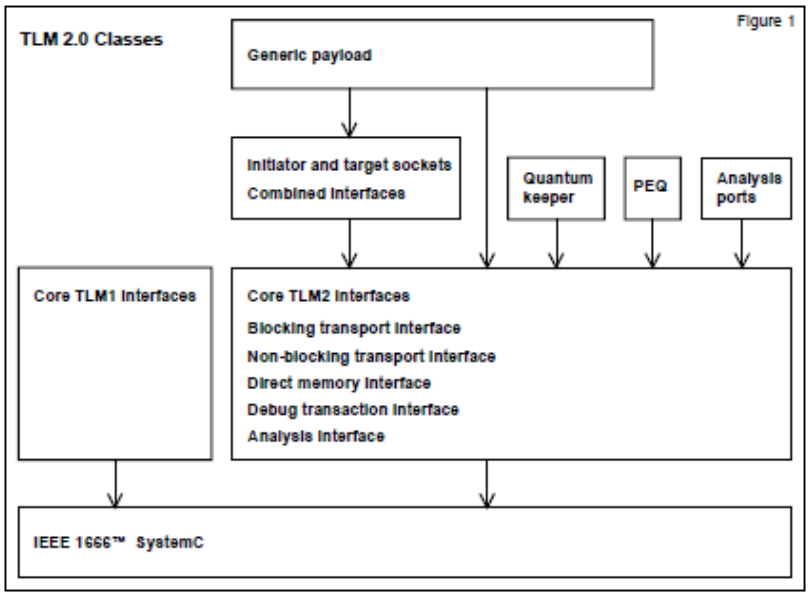
Words	訳語	説明
untimed (UT)	アンタイムド	時間やサイクルを明示的に言及しないモデリングスタイル。しかしオペレーションの並列性や順序は含まれる。時間概念は無いが、複数並列スレッド間のオペレーションの順序性は、イベント、ミューテックス、ブロッキングFIFO等の同期プリミティブを使って、複数並列スレッド間のオペレーションの順序性を実現しなければならない。ユーザによっては、プロトコルの頑健性をテストするため、アンタイムドな記述にランダム遅延を入れる場合があるが、この方法によってモデリング・スタイルの基本的な性質が変わることはない。
valid	バリッド	ポインタまたは参照で関数からリターンされるオブジェクトの状態で、そのオブジェクトが削除されておらず、その値や動作をアプリケーションからアクセスが可能な状態である期間。（SystemC用語）

4.2.4 TLM 2.0 draft2 ユーザマニュアルの日本語要約

本節は、OSCI の TLM2 USER MANUAL Ver. 1.1.0 を日本語要約したものである。

1. あらまし

- TLM2 はコア・インタフェース、アナリシス・ポート、イニシエータ・ソケットとターゲット・ソケット、汎用ペイロードから構成される。
- コア・インタフェースは以下から構成される。
  - TLM1 コア・インタフェース
  - ブロッキングとノン・ブロッキング・トランスポート・インタフェース
  - ダイレクト・メモリ・インタフェース（DMI）
  - デバッグ・トランザクション・インタフェース
  - ライト・インタフェース
  - アナリシス・インタフェース
  - コーディングスタイルをサポート：アンタイムド、ルーズリー・タイムド、アプロキシメイトリー・タイムド
- 汎用ペイロードの特徴は以下の通り。
  - メモリ・マップ・バスのモデリングをサポート。
  - 拡張機能により特定のバス・プロトコルに対応。
- ノン・ブロッキング・トランスポート・インタフェースのための便利クラス。
  - クォンタム・キーパー
  - ペイロード・イベント・キュー（PEQ）
- 言語の構成



1.1. スコープ

- このドキュメントは TLM2.0 標準の内容を記述する。
- このドキュメントはキー・コンセプトと TLM2 コア・インタフェースとクラスにフォーカスする。
- 説明用コード、例題、ユニット・テストについては説明しない。
- TLM1 コア・インタフェースをリストアップするが、semantics は定義しない。
- このドキュメントは言語のリファレンス・マニュアルではない。

2. リファレンス

- ISO/IEC 14882:2003, Programming Languages—C++
- IEEE Std 1666-2005, SystemC Language Reference Manual
- Requirements Specification for TLM 2.0, Version 1.1, September 16, 2007

2.1. 参考文献

- Transaction-Level Modeling with SystemC, TLM Concepts and Applications for Embedded Systems,

edited by Frank Ghenassia, published by Springer 2005, ISBN 10 0 387-26232-6(HB), ISBN 13 978-0-387-26232-1(HB)

- Integrated System-Level Modeling of Network-on-Chip enabled Multi-Processor Platforms, by Tim Kogel, Rainer Leupers, and Heinrich Meyr, published by Springer 2006, ISBN 10 1-4020-4825-4(HB), ISBN 13978-1-4020-4825-4(HB)
- ESL Design and Verification, by Brian Bailey, Grant Martin and Andrew Piziali, published by Morgan Kaufmann/Elsevier 2007, ISBN 10 0 12 373551-3, ISBN 13 978 0 12 373551-5

3. イントロダクション

3.1. 背景

- TLM1 には 3 つの（相互利用性低下とシミュレーション速度の低下をもたらす）欠点がある。

欠点	TLM2 での解決
トランザクション・クラスの標準が無い。	汎用ペイロード
タイミング・アノテーションをサポートしない。(モデル間でタイミング情報を交換する標準的方法がない。Wait で遅延を実装する。)	ノン・ブロッキング・トランスポート・インタフェース
トランザクションを値もしくは参照渡しする。	トランザクションは幾つかのトランスポート・コールに渡ってライフタイムを持つ。

3.2. トランザクション・レベル・モデリング、ユース・ケース、抽象度

- 分類法を変更
  - 抽象度ではなく、インタフェース（API）とコーディングスタイルを区別するアプローチをとる。
  - TLM2 標準はトランザクション・レベル・モデルを実装するための低レベルのプログラミング体系と考えるべきインタフェースのセットを定義している。
  - 様々なユース・ケースに適切ないくつかのコーディングスタイルを説明している。それぞれのコーディングスタイルは機能、タイミング、コミュニケーションに渡る様々な抽象度をサポートする。
- 1 スレッドからなるアンタイムド機能モデルはアルゴリズム・モデルと呼ばれ、トランザクション・レベルではない（プロセス間のコミュニケーションが無いので)。
- 複数プロセスからなるトランザクション・レベル・モデルは、1 つのプロセスの制御を他のプロセスに譲るメカニズムを要求する。次の 2 つの方法がある。
  ① コード中に明示的にコミュニケーションと同期ポイントを記述する。
    - FIFO、セマフォ等で実現可能。
    - 完璧なアンタイムド・モデリング・スタイルが可能（シミュレーション時間を進める必要無し、すべての同期ポイントは事前に決定でき、明示的にコーディングされている)。
  ② ルーズリー・タイムドと呼ばれるコーディングスタイル。
    - 複数の SW スレッドが歩調を合わせて動作しているわけではなく､不規則にコミュニケーションするような環境で動作しているようなケースに適する。おそらく、事前に完全にタイミング・ポイントを決めることができないケース。

3.3. コーディングスタイル

3.3.1. アンタイムド・コーディング・スタイル

- ブロッキング・トランスポート・インタフェースを使用。
- タイミング情報のアノテーションの仕組み無し。
- レイテンシをモデル化するために wait を使用。
- コード中に明示的に同期構文を呼ぶことにより、プロセス間の同期を実装する。
  - ブロッキング・トランスポート呼出しで、別プロセスにより通知されるイベントにウエイトする。または、wait 文を呼出すよう指示するフラグを返す。

3.3.2. ルーズリー・タイムド・コーディング・スタイルとテンポラル・デカップリング

- ノン・ブロッキング・トランスポート・インタフェースを使用。
  - タイミング・アノテーション、トランザクションの複数フェーズおよび複数タイミング・ポイントをサポート。
  - 一般ペイロードでは、2 つのタイミング・ポイント（リクエスト開始とレスポンス開始）を持つ。これらのタイミング・ポイントは同じシミュレーション時間でも、異なるシミュレーション時間でも生じえる。
- ユース・ケース：SW 開発
  - OS ブートと OS 実行をモデル化するに十分なタイマーと粗いプロセス・スケジューリングのモデリングをサポート。
- テンポラル・デカップリング
  - 高速シミュレーションを可能にする。
  - プロセスは、他プロセスによりアップデートされる変数に出会うまで、もしくは、他プロセスと情報

交換する必要があるまで、シミュレーション時間中前に進むことができる。
- バックトラックは許されない。個々のプロセスはモデルの機能を破壊しないでシミュレーション時間中前に進めるか決定する責任を負う。プロセスが外部依存に出会ったとき 2 つの選択がある。
  ① シミュレーション時間に追いつくまで他のプロセスに実行を譲る（強制同期）。
  ② 現在の値を受入れ、実行継続（値の早期サンプリングがダメージを与えない、以降の値変化は以降のプロセス実行中で取込まれると仮定）。

■ クォンタム・キーパー
- プロセスが先行実行してよい上限を課す（もし、上限が無ければ、SystemC スケジューラは他のプロセスを実行させることができない）。
- アプリケーションがクォンタムを設定。
- クォンタム値にはシミュレーション速度と精度のトレードオフがある。
  - 小さい値 = 遅い
  - 大きい値 = 低いタイミング精度（重要なイベントを取りそこなう、モデルが機能しなくなる）

■ 例
- プロセッサ、メモリ、タイマー、低速なペリフェラルから構成されるシステム。
- プロセッサ上で動作する SW はその実行時間の大半をメモリからの命令のフェッチと実行に費やし、システムの残りとはタイマーからの割込み（例えば、1 ms 毎）が発生したときのみ情報交換する。
- ISS モデルは 1 ms のクォンタムまで先行実行できる。メモリ・モデルには直接アクセスし、タイマー割込みの割合でペリフェラル・モデルとのみ同期する。

■ アンタイムド・モデル
- イニシエータは明示的な同期ポイントまでシミュレーション時間中を先行実行できる。
- あらかじめ決まったポイントでイニシエータ間で制御を譲るため、システム・モデル中の明示的同期ポイントの存在を当てにしている。

■ ルーズリー・タイムド・モデル
- 明示的同期ポイントがなくても OK。
- クォンタムに到達する前に制御を譲る（要求ベース同期）により、タイミング精度を上げることが可能。
- 要求ベース同期はアンタイムド・モデルの明示的同期を同等。

3.3.3. アンタイムド対ルーズリー・タイムド・モデリング

■ アンタイムド・モデル
- イニシエータは明示的な同期ポイントまでシミュレーション時間中を先行実行できる。
- あらかじめ決まったポイントでイニシエータ間に制御を譲るため、システム・モデル中の明示的同期ポイントの存在を当てにしている。

■ ルーズリー・タイムド・モデル
- 明示的同期ポイントがなくても OK。
- クォンタムに到達する前に制御を譲る（要求ベース同期）により、タイミング精度を上げることが可能。
- 要求ベース同期はアンタイムド・モデルの明示的同期を同等。

3.3.4. アプロキシメイトリー・タイムド・コーディング・スタイル

■ ノン・ブロッキング・トランスポート・インタフェースを使用。

■ ユース・ケース：アーキテクチャ解析、パフォーマンス解析
- トランザクションを複数のフェーズに分解。
  - フェーズ間は明示的なタイミング・ポイント。
- 汎用ペイロードでは、4 つのタイミング・ポイントが存在。
  - リクエストの開始と終了、レスポンスの開始と終了。

■ テンポラル・デカップリングは使用しない。
- プロセス間通信にディレイがアノテートされる。
- 2 種類のディレイで十分。ターゲットのレイテンシとアクセプト・ディレイ(initiation interval)。
  - wait(delay)または notify(delay)で実現。

3.3.5. アンタイムド、ルーズリー・タイムド、アプロキシメイトリー・タイムド・コーディング・スタイルの特徴

■ シミュレーション時間の使い方、いつ SystemC スケジューラへ譲るかで特徴付けられる。

コーディングスタイル	時間概念	同期
アンタイムド	必要無し。	明示的同期ポイントまで実行。
ルーズリー・タイムド	あり。プロセスは一時的にその時間から切り離されている。プロセスは消費した時間値を保持。	明示的同期ポイントに到達したとき、もしくは、その時間クォンタムを消費したとき、他のプロセスへ実行権を譲る。
アプロキシメイトリー タイムド	プロセスはシミュレーション時間に合わせて動作。	プロセス間相互作用のディレイは、wait かタイムド・イベント通知で実装。

3.3.6. ルーズリー・タイムドとアプロキシメイトリー・タイムド・モデリング間の切替え

- 詳細は TLM 2.0 の最終版には含まれる。
- シミュレーション中にルーズリー・タイムドからアプロキシメイトリー・タイムドへ切替えて良い。
- 利用例
  - ルーズリー・タイムド・レベルでリセットとブートを高速に実行し、興味のあるステージでより詳細な解析をするためアプロキシメイトリー・タイムド・モデリングへ切替える。

3.3.7. サイクル精度モデリング

- TLM 2 のスコープ外。
- TLM 1 そのままでモデリング可能。
  - 原理的にアプロキシメイトリー・タイムド・コーディング・スタイルでモデリング可能。
  - 適切なフェーズとトランザクションのどのアトリビュートを誰が変更し、誰が読むのかのルールを定義する。
  - 各フェーズは 1 サイクルを表す。

3.3.8. ブロッキング対ノンブロッキング・トランスポート・インタフェース

- 使用するコーディングスタイルに応じてインタフェースを選ぶ。
- インタフェースの混在、コーディングスタイルの混在はコスト（シミュレーション速度とコーディングの複雑さ）を負うので、避ける。
- ブロッキング・トランスポート・インタフェース
  - プロセス間の明示的同期を持つモデル（同期ポイントの順序が決定的で実装の詳細に依存しない）に適する。
  - アンタイムド・モデルから直接 RTLI へ変換するケースにも適する。
  - 明示的な詳細なタイミング無しで単純なコーディングスタイルを可能とする。
- ノン・ブロッキング・トランスポート・インタフェース
  - タイミング情報を持つ。
  - ルーズリー・タイムドからアプロキシメイトリー・タイムド・モデルへリファインするケース、プロセス間の同期ポイントの順序が非決定的もしくはハードウェア実装・OS スケジューラに依存するようなモデルに適する。
  - ルーズリー・タイムドコーディング・スタイルでのテンポラル・デカップリングとアプロキシメイトリー・タイムド・モデリングのタイミング・アノテーションを可能とする。

3.3.9. ユース・ケースとコーディングスタイル

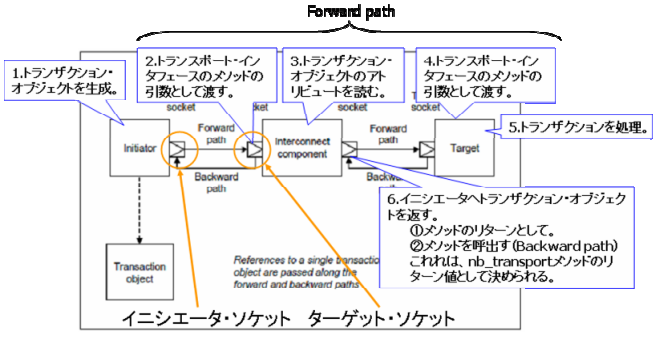
ユース・ケース	コーディングスタイル
アプリケーション SW 開発	アンタイムド、ルーズリー・タイムド
SW パフォーマンス解析	ルーズリー・タイムド
HW アーキテクチャ解析	ルーズリー・タイムド、アプロキシメイトリー・タイムド
HW パフォーマンス検証	アプロキシメイトリー・タイムド、サイクル精度
HW 機能検証	アンタイムド（検証環境）、ルーズリー・タイムド、アプロキシメイトリー・タイムド

3.4. イニシエータ、ターゲット、ソケット、ブリッジ

- TLM2 トランスポート・インタフェースはイニシエータとターゲット間でトランザクションを受け渡す。

コンポーネント	役割・動作
イニシエータ	トランザクションを生成。 コア・インタフェースのメソッドを呼ぶことによりトランザクションを渡す。
ターゲット	トランザクションの最終目的地。
接続コンポーネント （アービター、ラウター）	イニシエータもしくはターゲットとして動作。

- トランザクション・オブジェクトのライフタイム



- フォワード・パスとバックワード・パスをサポートする仕組み
  - イニシエータ・ソケット（port）とターゲット・ソケット（export）。
- イニシエータは少なくとも 1 つのイニシエータ・ソケットを持つ。ターゲットは少なくとも 1 つのターゲ

ット・ソケットを持つ。接続コンポーネントは少なくとも 1 つずつイニシエータ・ソケットとターゲット・ソケットを持つ。

- 異なるトランザクションを転送するいくつかの（イニシエータもしくはターゲット）ソケットを持っても良い。
  - TLM2 トランザクション間のブリッジ・コンポーネント。
- バスブリッジのモデリング方法

モデリング方法	利点
接続コンポーネント	高速なシミュレーション ポインタ渡し
TLM2 トランザクション間のブリッジ	柔軟性（トランザクションが異なるアトリビュートを持てる） コピー渡し

- ソケットは利便性のため提供。最大の相互運用性と一貫したコーディングスタイルのため推奨。
  - TLM2 コア・インタフェースを直接使用しても良い。

3.5. DMI、デバッグ・トランザクション・インタフェース

- トランスポート・インタフェースにより使用される接続コンポーネントを通る通常のパスをバイパスする。
- イニシエータ・ソケット、ターゲット・ソケットに含まれる。

インタフェース	目的	アクセス先	アクセス方向
DMI （ダイレクト・メモリ・インタフェース）	アンタイムド、ルーズリー・タイムド・シミュレーション時における通常のメモリ・トランザクションを高速化する。	ターゲットの持つメモリ	フォーワード・インタフェース (イニシエータからターゲットへ) バックワード・インタフェース（ターゲットからイニシエータへ)
デバッグ・トランザクション・インタフェース	通常のトランザクションに付随したディレイ、副作用の無いデバッグ・アクセス。	ターゲットの持つメモリ	フォーワード・インタフェース

3.6. ネームスペース

- すべての TLM2 クラスは「tlm」名前空間内で定義されている。
- TLM2 クラスの実装は「tlm」名前空間内の名前空間にあるかもしれないが、その名前空間はアプリケーションで使うべきではない。

3.7. ヘッダー・ファイル

- ユーザ・コードは「tlm.h」をインクルードすること。

4. コア TLM2インタフェース

TLM1 からのコア・インタフェースへの追加で、TLM2 はブロッキングとノン・ブロッキングのトランスポート・インタフェース、ダイレクト・メモリ・インタフェース（DMI）とデバッグ・トランザクション・インタフェースを追加する。

4.1. ブロッキング・トランスポート・インタフェース

4.1.1. 概要

新しい TLM 2 のブロッキング・インタフェースは TLM1 のトランスポート・インタフェースと似せているが全く同一のものではない。TLM1 は、別々の要求と応答オブジェクトが値(または、コンスト・リファレンス)を渡すと仮定するが、ブロッキングを使用するか、ノンブロッキング TLM2 インタフェースであることにかかわらず、TLM2 は、単一トランザクション・オブジェクトが参照で渡されると仮定している。

4.1.2. TLM1 からのマイグレーション・パス

古い TLM1 と新しい TLM2 のインタフェースは共に TLM2 の一部である。新旧ブロッキング・トランスポート・インタフェース間の類似性が、TLM 1 インタフェースを使用しているレガシーモデルと新しい TLM2 インタフェース間のアダプタ構築の作業を容易にする。

4.1.3. クラス定義

```
namespace tlm {
  template <typename TRANS = tlm_generic_payload>
  class tlm_blocking_transport_if : public virtual sc_core::sc_interface {
  public:
    virtual void b_transport(TRANS& trans) = 0;
  };
} // namespace tlm
```

4.1.4. TRANS テンプレート引数

このコア・インタフェースがあらゆる型のトランスポート・トランザクションに使用される可能性を意図しているが、正確なトランザクション属性が重要でないモデル間の相互利用性向上の為、汎用ペイロードのトランザクション・タイプ(tlm_generic_payload)を提供する。相互利用性を最大限にするには、デフォルトトランザクション型の tlm_generic_payload を使用すべきである。

4.1.5. ルール

a) コーラーにはストレージのアロケートとトランザクション・オブジェクト構築の役割がある。b_transport メソッドは完全に初期化されたトランザクション・オブジェクトと共に呼び出される。

b) 汎用ペイロードの場合では、b_transport の呼び出しは、最初のトランザクションのタイミング・ポイントをマークする。b_transport からの戻りは、トランザクションの最後のタイミング・ポイントをマークする。

c) コーリーはトランザクションのクラス TRANS によって課されたあらゆる制約を条件としてトランザクション・オブジェクトを変更または更新する可能性がある。

d) コーリーは呼び出した後、トランザクション・オブジェクトの消去とプールを行う役割がある。

e) コーリーは、コーラーが b_transport から戻った際、そのトランザクション・オブジェクトを無効にすると仮定する。

f) f) b_transport メソッドは、タイミング・アノテーションとテンポラル・デカップリングをサポートしない。

g) トランザクション・オブジェクトはタイミング情報を含まない事を推奨する。タイミングはノンブロッキング・トランスポート・インタフェースを使用してアノテートされる。

h) b_transport は一つまたはいくつかの nb_transport の呼び出しを作る可能性がある。それはアンタイムド・イニシエータと、ルーズリー・タイムドまたはアプロキシメイトリー・タイムドとの間のアダプタの作成を容易にする。

4.2. ノンブロッキング・トランスポート・インタフェース

4.2.1. 概要

新しいノンブロッキング・トランスポート・インタフェースは、ルーズリー・タイムドとアプロキシメイトリー・タイムドの両方がサポートされ、nb_transport メソッドはトランザクション・オブジェクトの参照を渡す事以外はブロッキングと異なる。nb_transport メソッドはステート表示のフェーズと、次のフェーズのトランザクションの遅延をアノテートする時間を渡し、トランザクション・オブジェクトのライフタイム、タイミング・アノテーションや高速シミュレーションの為のテンポラル・デカップリングと共に複数のフェーズをサポートしている。

ノンブロッキング・トランスポート・インタフェースは、アンタイムド・コーディング・スタイルもサポートするが簡略化される可能性がある。ノンブロッキング・トランスポート・インタフェースは特に、ブロッキング・トランスポートの使用が不便なパイプラインされたトランザクションをモデルするのに適している。ノンブロッキング・トランスポート・インタフェースと汎用ペイロードは、メモリ・マップド・バスの抽象モデリング高速化の為にデザインされ、トランザクション型とフェーズ型のテンプレート引数で提供される為、汎用ペイロードと分けて使用出来る。

4.2.2. クラス定義

```
namespace tlm {
  enum tlm_phase { BEGIN_REQ, END_REQ, BEGIN_RESP, END_RESP };
  enum tlm_sync_enum { TLM_REJECTED = 0, TLM_ACCEPTED = 1,
  TLM_UPDATED = 2, TLM_COMPLETED = 3 };
  template <typename TRANS = tlm_generic_payload, typename PHASE = tlm_phase>
  class tlm_nonblocking_transport_if : public virtual sc_core::sc_interface {
  public:
    virtual tlm_sync_enum nb_transport(TRANS& trans, PHASE& phase, sc_core::sc_time& t) = 0;
  };
} // namespace tlm
```

4.2.3. TRANS テンプレート引数

このコア・インタフェースがあらゆる型のトランスポート・トランザクションに使用される可能性を意図しているが、正確なトランザクション属性が重要でないモデル間の相互利用性向上の為、汎用ペイロードのトランザクション・タイプ(tlm_generic_payload)を提供する。相互利用性を最大限にするには、デフォルトトランザクション型の tlm_generic_payload を使用すべきである。

4.2.4. PHASE テンプレート引数

このコア・インタフェースはあらゆる型フェーズとタイミング・ポイントと共にトランスポート・トランザクションに使用される可能性を意図しているが、汎用ペイロードと共に使用される特別な tlm_phase 型が提供される。相互利用性を最大限にするには、ルーズリーまたはアプロキシメイトリー・タイムド・コーディングにおいて、デフォルト引数の tlm_phase 型と tlm_generic_payload を使用すべきである。

4.2.5. nb_transport メソッド呼び出し

a) nb_transport メソッドは直接、間接的に wait を呼ばない。

b) nb_transport メソッドはスレッドプロセスまたはメソッドプロセスから呼ばれない。

c) ブロッキングとノンブロッキングのトランスポートの両方が共に使われる場合、nb_transport は b_transport を呼ぶ必要性がある。

d) ルーズリーとアプロキシメイトリーの両方で、nb_transport はイニシエータとターゲット間のフォワードとバックワード・パスの両方に従って呼び出される。汎用ペイロードの場合、二つのパスは反転した同じコンポーネントとソケットのシーケンスを通す。

e) イニシエータは最終タイミング・ポイントの後、トランザクション・オブジェクトの削除とプールの役割

を持ち、nb_transport の最終呼び出しがフォワード・パスの場合、イニシエータは nb_transport から戻るトランザクション・オブジェクトを削除する可能性があり、nb_transport の最終呼び出しがバックワード・パスの場合、イニシエータは、そのプロセスがリジュームする次の時間でトランザクション・オブジェクトを削除する可能性がある。いずれのケースでもトランザクション・オブジェクトは、ターゲットが SystemC スケジューラに制御を渡すまで、ターゲットによるバリッドとアクセス権が残り、ターゲットはその制御を渡す前にトランザクション・オブジェクトのステートを確認する機会をある。その後、いかなるインターコネクト・コンポーネントや、ターゲットはトランザクション・オブジェクトがインバリッドであると仮定しなければならない。

f) インターコネクト・コンポーネントやターゲットが、各々のトランザクション・インスタンスの nb_transport を最終呼び出しした後、トランザクションのステートにアクセスする必要がある場合、それはトランザクション・オブジェクトのコピーを作る必要がある。

g) フォワード・パスにおける nb_transport 呼び出しは、バックワード・パスに関連する nb_transport の呼び出しを部分的に直接または間接的に行わず、逆もまた同様である。

4.2.6. trans 引数

a) イニシエータは、最初の引数が渡されるトランザクション・オブジェクトがアロケートされているストレージと構造の役割を持つ。nb_transport メソッドは、十分に初期化されたトランザクション・オブジェクトと共に呼び出される。

b) 与えられたトランザクション・オブジェクトの寿命は、nb_transport を連続呼び出すイニシエータとターゲットの間を一つのトランスポート・オブジェクトのフォワードとバックワードを渡すような性質の nb_transport からの戻りを超えて延長される可能性がある。

c) nb_transport のコーラーはイニシエータである必要はない。特に、ターゲットはイニシエータにタイミング・ポイントを戻す信号を送る為に nb_transport を呼び出す可能性がある。

d) トランザクション・オブジェクトのライフタイムは、nb_transport の各々の呼び出しを超えて延長される可能性がある為、コーラーまたはコーリーのいずれかは、TRANS クラスのトランザクションによって課せられるあらゆる制約を条件として、トランザクション・オブジェクトの変更やアップデートが起こる可能性がある。

e) イニシエータは最後のタイミング・ポイントの後、トランザクション・オブジェクトの削除とプールの役割を持つ。イニシエータは、イニシエータのみがトランザクション・オブジェクトを削除する可能性がある。

4.2.7. phase 引数

a) イニシエータは、最初の引数が渡されるトランザクション・オブジェクトがアロケートされているストレージと構造の役割を持つ。nb_transport メソッドは、十分に初期化されたトランザクション・オブジェクトと共に呼び出される。

b) トランザクションのアトリビュートはフェーズ・トランザクションをマークするタイミング・ポイントの時に変更するのみである為、概念的には各フェーズにおいてステーブルである。

c) フェーズ引数は参照渡しされる。コーラーまたはコーリーのいずれかはフェーズを変更する可能性がある。

d) トランザクションのステートへのいずれの変化も、コーラーまたはコーリーのいずれか一つの呼び出しから次までのフェーズ引数の値を比較する事によって変化を検出する事が出来るように、フェーズ引数の変化と合わせるべきである。

e) フェーズ引数の値は、コーリーとコーラー間の通信を行う為のプロトコル・ステートマシンの現在のステートを表す。単一のトランザクション・オブジェクトが二つ以上のコンポーネント間をパスされるいずれかで、各コーリー／コーラー接続は、異なるプロトコル・ステートマシンを（少なくとも概念的に）要求する。

f) トランザクション・オブジェクトは、フェーズが一般的なそのコーラーに対してローカルである nb_transport の単体呼び出しを超えて延長するであろうライフタイムとスコープを持つ。与えられたトランザクション用の各々nb_transport 呼び出しには、別々のフェーズ・オブジェクトを持つ可能性がある。

g) デフォルト・フェーズ引数 tlm_phase の最終タイミング・ポイントは、採用されるコーディングスタイルに依存する。ルーズリー・タイムド・コーディング・スタイルでは、フェーズ BEGIN_RESP にトランザクションは、最終タイミング・ポイントとしてマークする。アプロキシメイトリー・タイムド・コーディング・スタイルでは、フェーズ END_RESP にトランザクションは、最終タイミング・ポイントをマークする。

h) enum の tlm_phase は、明確な汎用ペイロードである。その他のプロトコルはこの同じフェーズ型を使用するか、それ自体のフェーズ型（相互利用性の通信ロスをと共に）を変わりに用いる可能性がある。

4.2.8. sc_time 引数

a) トランザクション・オブジェクトは、タイミング情報を含まない事が推奨される。タイミングは nb_transport の sc_time 引数を使用してアノテートされるべきである。

b) 時間引数はマイナスではなく、常にカレント・シミュレーション時間に関連している。

c) ルーズリー・タイムド・コーディング・スタイルを使用した時、コーラーは sc_time_stamp()の時間に受け取られたトランザクションであるかのように、反応すべきコーリーを指定する為の時間引数の正の値を渡す可能性がある。

d) テンポラル・デカップリングの汎用的な記述については、3.3.2 節のルーズリー・タイムド・コーディングスタイルとテンポラル・デカップリングを参照。
e) クォンタムの記述については、7.3 節のテンポラル・デカップリングを使用したプロセスのルールを参照。
f) もしコーラーが、時間内の未来の時点（sc_time_stamp() + t）に起こるような、いずれか必要とされるステート情報の予測に基づいたトランザクションをどう処理するか決定する事が出来ない場合、コーリーは TLM_ACCEPTED の値を戻すべきである。
g) 時間引数の正の値と共に、コーリーは、シミュレーション時間から、また“タイムワープ内で生きている”まで、テンポラル・デカップリングされるようになる事を仕向ける。
h) コーラーに時間引数の正の数を nb_transport にパスさせるのは、一般的にルーズリー・タイムド・コード・スタイルが連想されられるが、アプロキシメイトリー・モデルであっても技術的に可能である。アプロキシメイトリー・タイムド・コード・スタイルを使用した場合、コーリーがリーズナブルな動作な時だけ、与えられた遅延（恐らくペイロード・イベント・キューを使用する）と共に時間通知が作成され、そして TLM_ACCEPTED の値を返す。
i) nb_transport がシミュレーション時に連続して、かつ２回目の呼び出しが、最初の呼び出し以前に事実上処理されるような時間引数と共に呼び出される場合、コーリーには、現在の時間までトランザクションの到達時間の遅延をペイロード・イベント・キューに入れるか、または nb_transport 呼び出し中にトランザクションを処理して、システムの設計がアウトオブオーダーの実行を許容することができると仮定するという、二つの選択肢がある。
j) nb_transport から戻る上で、それは nb_transport の sc_time 引数を t とする、sc_time_stamp() + t の時間にステートが変化するであろうトランザクションの通知を受け取るかのように動く事がコーラーの役割である。
k) nb_transport から戻る上で、コーラーはアノテートされるレイテンシの実装に、テンポラル・デカップリング・モード、ペイロード・イベント・キューにトランザクションを置く、または、wait(t)を呼ぶ、３つの選択肢がある。
l) このノンブロッキング・トランスポート・インターフェースは明らかにパイプラインされたトランザクションをサポートする事が意図されている。
m) コーリーは時間引数の値を増加させる可能性があるが、しかし値を減少させない。このルールは SystemC シミュレータを逆時間に走らせられない事からの制約である。

4.2.9. tlm_sync_enum の戻り値

a) 同期する事はプロセスの動作時に SystemC スケジューラを制御する事であるが、テンポラル・デカップリングも含まれる。
b) 原則同期はいつでも実行可能(スレッドプロセスの場合で wait を呼ぶか、またはメソッドプロセスでカーネルに戻る)で、ルーズリー・タイムド・コーディング・スタイルを使用する時、プロセスは tlm_quantum_keeper クラスの同期メソッドを呼ぶことによって同期する。
c) プロセスが同期するルールのステートがある時、実際のコントロールが実施される前に、プロセスは先のステートを実行する可能性がある。
d) 下記のルールはフォワードとバックワード・パスの両方に適用される。
e) 戻り値の意味は固定されており、トランザクション型やフェーズ型によっては変わらない。
f) TLM_REJECTED：コーリーはトランザクションをリジェクトした。
g) TLM_ACCEPTED：コーリーはトランザクションを受け取った。
h) TLM_UPDATE：コーリーはトランザクションを受け取りと更新した。
i) TLM_COMPLETED:コーリーはトランザクションを受け取り、そしてそのトランザクションを完了した。

4.2.10. コーディングスタイルと tlm_sync_enum

tlm_sync_num のコードスタイルに関する戻り値の推奨される使用は次の通り。
LT = Loosely-timed, AT = Approximately-timed, CA = Cycle accurate.

tlm_sync_enum	Coding style	Transaction and phase arguments	Timing annotation
TLM_REJECTED	AT, CA	Unmodified	No
TLM_ACCEPTED	LT, AT	Unmodified	No
TLM_UPDATED	AT	Updated	Yes
TLM_COMPLETED	LT, AT	Updated, but caller may ignore phase	Yes

4.2.11. コーディングスタイルと tlm_phase

下記のテーブルは、コーディングスタイルによる tlm_pahse の使用方法がまとめられている。

tlm_phase	Coding style	Path
BEGIN_REQ	Loosely-timed and approximately-timed	Forward (call)
END_REQ	Approximately-timed only	Forward (return) Backward (call)
BEGIN_RESP	Loosely-timed and approximately-timed	Forward (return) Backward (call)
END_RESP	Approximately-timed only	Forward (call) Backward (return)

4.2.12. ルーズリー・タイムドとアプロキシメイトリー・タイムド・コーディング・スタイル向けフェーズ・シーケンス

汎用ペイロードによって使用されるフェーズ型の tlm_phase に着目する。

ルーズリー・タイムド・コーディングスタイル向けフェーズ・トランザクションのフルシーケンスは、

BEGIN_REQ → BEGIN_RESP

アプロキシメイトリー・タイムド・コーディングスタイル向けフェーズ・トランザクションのフルシーケンスは、

BEGIN_REQ → END_REQ → BEGIN_RESP → END_RESP

しかしながら、ノンブロッキング・トランスポート・インタフェース自体の 2 つのコーディングスタイルの間には大きな区別はなく、TLM_COMPLETED の値を返す nb_transport によりこれらのシーケンスの両方ともに先取りが可能である。

TLM_COMPLETED：戻り値は、フォワードまたはバックワード・パス上に送られた END_RESP に関わらず、TLM_ACCEPTED、TLM_UPDATED、または TLM_COMPLETED のいずれかである。

もし、アプロキシメイトリー・タイムドのイニシエータが END_REQ の最初に受け取りを除き、BEGIN_RESP をターゲットから受け取る場合、BEGIN_RESP に先立って暗黙な END_REQ が来ている事を仮定する。

BEGIN_REQ (→ BEGIN_RESP)
BEGIN_REQ → BEGIN_RESP
BEGIN_REQ → END_REQ (→ BEGIN_RESP → END_RESP)
BEGIN_REQ → END_REQ → BEGIN_RESP (→ END_RESP)
BEGIN_REQ (→ END_REQ) → BEGIN_RESP → END_RESP
BEGIN_REQ → END_REQ → BEGIN_RESP → END_RESP

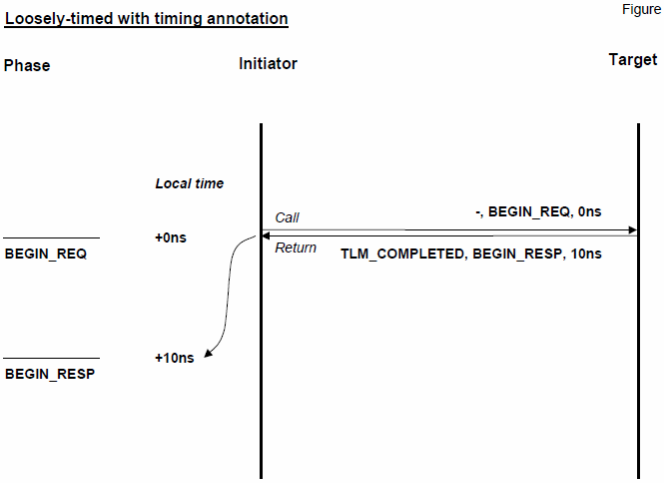
4.2.13. Message sequence charts for nb_transport

- 一連のメッセージ手順図が、LT/AT 記述スタイルのための nb_transport コールとタイミングの認められた手順で図解入りで以下に示されています。
- nb_transport への/からの引数と戻り値は、表記法 return, phase, delay を使って示しています。
- return の箇所はファンクションコールからの返り値が、phase の箇所はフェーズ引数の値が、delay の箇所には sc_time 引数の値になります。
- 表記に '-' が使われている際は、値が未使用であるという事です。

4.2.13.1. Loosely-timed with timing annotation

- LT 記述は 2 相の BIGIN_REQ と BIGIN_RESP に制限されます。(順番も)
- initiator から target に BEGIN_REQ を送り、target から initiator に BEGIN_RESP を戻す。
- もし、target がすぐに応答時間とトランザクションの次の状態を計算できるなら、以下に示すように BEGIN_RESP フェーズに、トランザクションの遅延をアノテート。
- nb_transport が TLM_COMPLETED を返すとき、callee が実際に引数 phase をアップデートする義務は無い。したがって以下の図で、call はまだ BEGIN_REQ に設定された phase のまま返ったかも知れません。
- トランザクションはパイプライン化されるかも知れません。
- Initiator は最初の call からの遅延した return を得る前に、別のトランザククションを target に nb_transport を call することができます。どのような遅延であるかを説明するのは initiator の義務です。

Figure 4

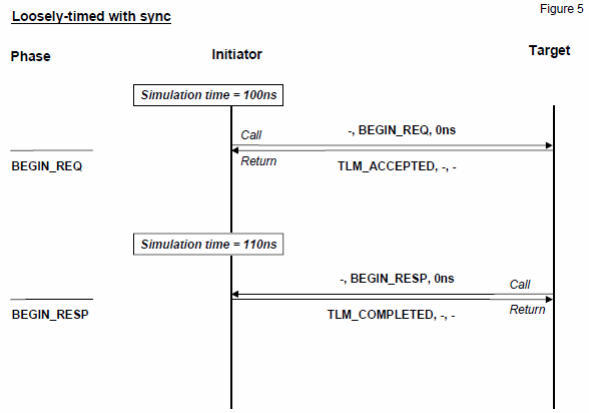


4.2.13.2. Loosely-timed with sync

- Terget が応答時間やトランザクションの次の状態について計算できないのであれば、TLM_ACCEPTED で返すべきです。(以下のように)
- これは、intiator に後で後方のパスで応答を期待していることを知らせています。しかし、initiator が SystemC スケジューラに制御を譲った後にだけです。
- 次に、terget は、後方のパスで phase を BEGIN_RESP にセットした nb_transport を call します。
- このトランザクションの最終的なタイミング先は terget による nb_transport への call で示されます。
- nb_transport からの return では、terget が SystemC スケジューラに (call 待ち、sync、method プロセ

スからの return で）制御を譲り返すまでしか、トランザクションは有効なままで残りません。その時、initiator プロセスは再開して、トランザクション・オブジェクトを削除するかも知れません。

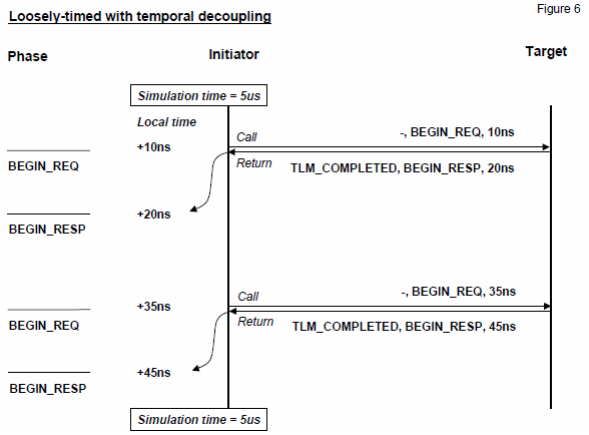
Figure 5



4.2.13.3. Loosely-timed with temporal decoupling

- 時間的に分断された initiator は、現在のシミュレーション時間の前に抽象的なローカル時間で実行されるかも知れません。その場合、引数 time に 0 以外の値を nb_transport に渡すべきです。（以下のように）
- Terget はその後の nb_transport の call で引数 time の値を増加させることによってローカル時間をさらに進めるかも知れません。
- 呼び出しからシミュレーション時間まで返された追加の時間は、トランザクションが完了する抽象的な時間を与えます。しかし、initiator が譲るまでシミュレーション時間事態が進むことはできません。

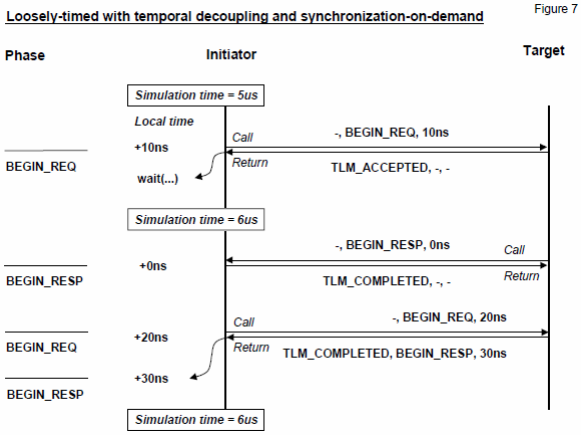
Figure 6



4.2.13.4. Loosely-timed with temporal decoupling and synchronization-on-demand

- 時間的に分断された initiator がシミュレーション時間を跨いで、トランザクションを target にパスするとき、事実上、それは未来について予測しているか、「タイムワープ」で実行するように target に依頼しています。
- もし、target が現在のシミュレーション時に持っている情報に基づくトランザクタの次の状態について計算できないなら、TLM_COMPLETED を返す代わりに、target はこのとき TLM_ACCEPTED を返すことによりトランザクションの完了を拒否するかも知れません。
- これは、「synchronization-on-demand」です。そうする前に、更なるステートメントを作成するかも知れませんが、initiator はどこかのポイントでスケジューラに制御を譲るはずです。
- その後、シミュレーション時間が進んだ後に、target はトランザクションの次の状態について計算できるだけの情報を持って、後方のパスで nb_transport に call することができます。

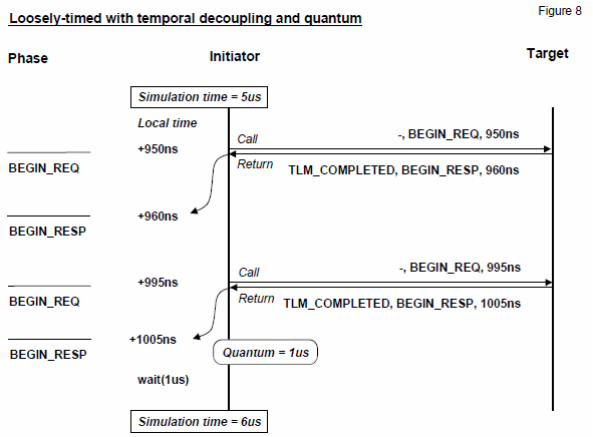
Figure 7



4.2.13.5. Loosely-timed with temporal decoupling and quantum

- 時間的に分断された initiator は、時間量を超えるまで、ローカル時間を進め続けます。その時、initiator が次の量子境界線までサスペンドすることによって、同期せざるを得なくなります。これは、モデルの中の他の initiator の実行とキャッチアップを可能にします。この意味は、事実上 initiator がそれ自身のローカル時間を記録し制御を返すので順番に実行することになります。オリジナルの initiator はシミュレーション時間が次の境界に達した後に再び実行されるだけであるべきです。
- LT コーディングスタイルの遅延の第一の目的は、各 initiator がいつ制御を返すかを決定することを可能にすることです。モデルが正しく機能する手段としてタイミングの詳細を当てにしないのが、最も良いことです。
- 各境界の中で initiator によって発生したトランザクションは厳格な連続したオーダーで起きます。しかし、シミュレーション時間を進めること無しにです。ローカル時間は SystemC のスケジューラによって監視されません。

Figure 8



4.2.13.6. Approximately-timed timing parameters

A) 一般的なペイロードがある AT コーディングスタイルは 4 つの phase を使用します。LT と同様に BEGIN_REQ と BEGIN_RESP を持っていますが、END_REQ と END_RESP フェーズが追加されています。BEGIN_REQ と END_RESP を initiator から target に送り、 END_REQ と BEGIN_RESP を target から initiator まで送ります。

B) LT target は、BEGIN_RESP で BEGIN_REQ に応答しますが、AT target の場合は、BEGIN_REQ に対して、END_REQ または BIGIN_RESP のどちらかで応答するかも知れないということに注意してください。BEGIN_RESP のケースでは、END_REQ は暗黙です。

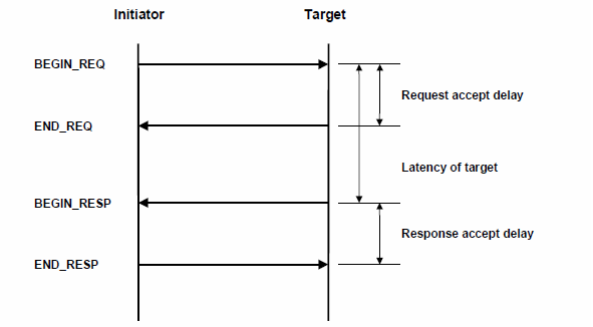
C) 4 つのフェーズで、それぞれ request accept delay（または、連続したトランザクションを送る際の最小開始間隔）、 latency of target と response accept delay をモデル化できます。

D) 連続したトランザクションをパイプライン化できます。

E) initiatoer は、前のトランザクションの target からの END_REQ を受け取るか、または target が TLM_COMPLETED で前のトランザクションを完了するまで BEGIN_REQ フェーズの新しいトランザクションを開始しないものとします。

F) target は、前のトランザクションのために initiator からの END_RESP を受け取るまで、または、TLM_COMPLETED で前のトランザクションが完了するまで、BEGIN_RESP フェーズの新しいトランザクションに応答しないものとします。

Figure 9

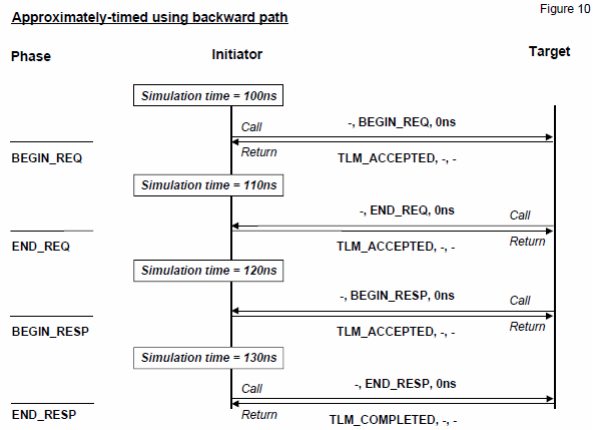


4.2.13.7. Approximately-timed using backward path

- 一般的なペイロードのために AT コーディングスタイルでは、トランザクションは initiator と target の間で 2 度前後渡されます。（他のプロトコルにおいては、この数は大小するかも知れません。）LT であれば、target はすぐにトランザクションに応じることが出来るかもしれません。そして、これが nb_transport のファンクションコールから返された値に反映されます。

- もし、nb_transport コールの受取人がすぐにトランザクションの次の状態について計算できない、または次のフェーズのトランザクションの遅延を計算できないのであれば、LT と同様に TLM_ACCEPTED を返すべきである。Caller は、スケジューラに制御を譲って、callee が応じる準備が出来ているとき、反対のパスで呼び出しを nb_transport で受け取ると期待するはずです。Callee がこの場合、LT のケースと違い、initiator か target であるかもしれないというところに注意してください。
- プロセスは通常、シミュレーション時間を進めることを許容する手段として、スケジューラに制御するので、AT コーディングスタイルは LT コーディングスタイルよりも、より多くゆっくりとシミュレートされることが予想されます。
- Callee は、それが、他の phase を先取りして、最終 phase までジャンプしたことを caller に示すために、どの phase でも TLM_COMPRETED を返してトランザクションを完了することができます。これは、initiator と target に同じく適用されます。

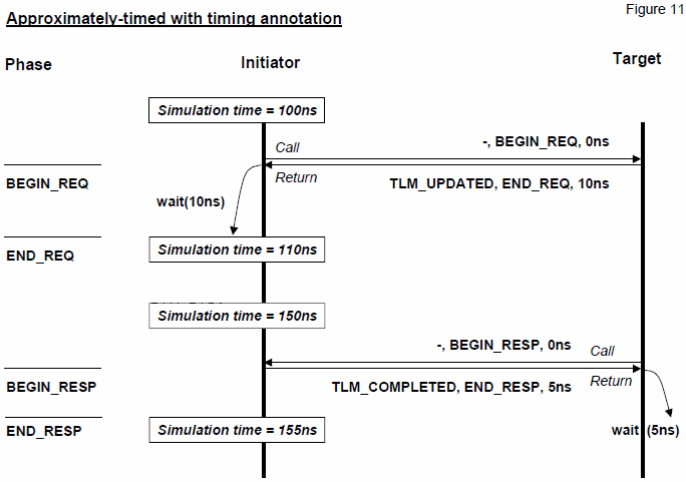
Figure 10



4.2.13.8. Approximately-timed with timing annotation

- nb_transport の呼び出しの受取人が次の状態のトランザクション、時間、および次のフェーズのトランザクションへの遅延をすぐに計算できるなら、反対の経路を使用するよりも nb_transport からの新しい状態をむしろ返すかもしれない。 loosely-timed コーディングスタイルにおいて、TLM_COMPLETED によって行われた。 approximately-timed コーディングスタイルでトランザクションが完了するときでも、これは可能です。しかし、 callee は、TLM_UPDATED と共に、トランザクションとフェーズをアップデートして、 phase transition に遅延をアノテートするべきである。

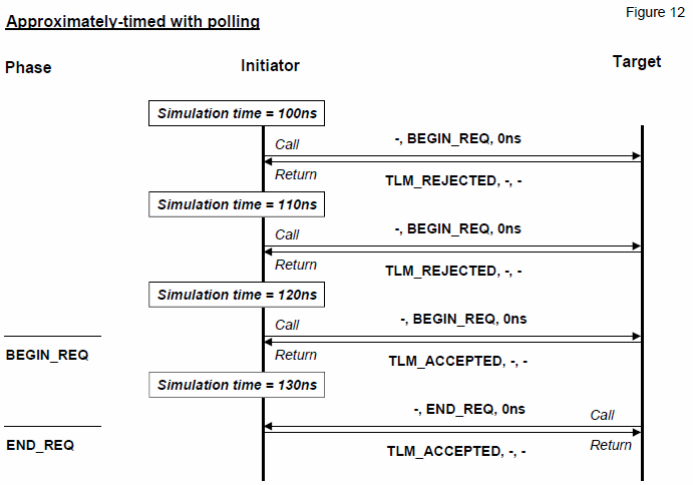
Figure 11



4.2.13.9. Approximately-timed with polling

- ターゲットは nb_transport で TLM_REJECTED の値を返すことによって incoming transaction を跳ね返らせる事が許される。 TLM_REJECTED は、ターゲットが、与えられたトランザクションの存在を示すのさえ拒否したことを意味する。どんな代替手段にも initiator には無いが、柔軟なコントロールと後で再試行する。
- TLM_REJECTED は全く TLM_ACCEPTED と異なっている。 TLM_ACCEPTED は、immediate response を提供できないが、ターゲットはトランザクションを受け入れる。
- 通信のポーリング・スタイルをモデル化するのに TLM_REJECTED を使用できる。

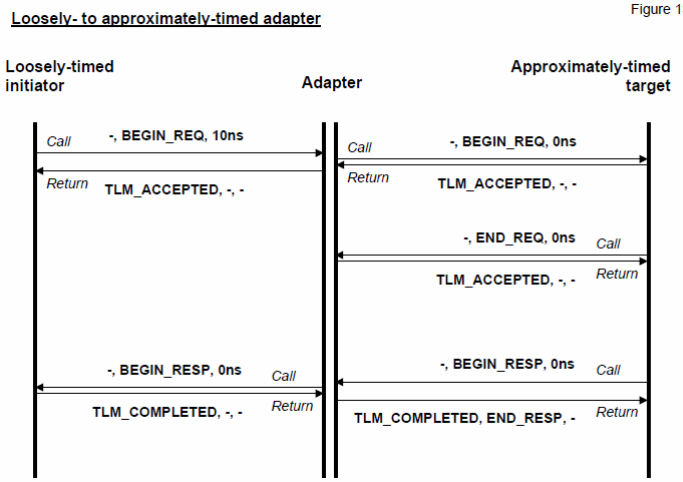
Figure 12



4.2.13.10. Loosely- to approximately-timed adapter

- loosely-timed initiator は approximately-timed target と 2 つある方法の 1 つで働くことができる。Initiator が approximately-timed phase transitions の完全なセットを実装していてそれに応じるか、それらの間にアダプタを挿入することによって。マイナス面は、initiator の temporal decoupling から得られるシミュレーション速度利点も、target に同期する必要性によって無くなるということである。アダプタを使用する通信のための message sequence chart を以下に示します。
- このとき、 loosely-timed の initiator は temporally decoupled であり、最初のコールで local time を nb_transport に渡すことに注意してください。 Target は local time を無視して、initiator はすぐに、やむを得ず同期します。 phase transition を END_REQ への変更は initiator で必要とされません。BEGIN_RESP は initiator に戻されます。(initiator は TLM_COMPLETED 受け取って、トランザクションが完了していると見なします)。アダプタは、END_RESP フェーズへの変遷を発生させて、リターンのときにそれをファンクションコールからターゲットに戻すことができます。 approximately-timed target は nb_transport コールの間にシミュレーション時間を進めるのを待っているかもしれません。
- もし、アダプタが省略されて、initiator が直接 target に接続したなら、loosely-timed initiator は target からの END_REQ コールを無視できなければなりません。このすべてが、generic payload が使用されていると仮定します。追加フェーズがある他のプロトコルはそれ自身のルールを必要とするだろう。
- loosely-timed initiator は backward path の上の adapter から initiator までの nb_transport に最後のコールに続いて、実行を再開するとすぐに、トランザクション・オブジェクトを自由に削除するか、プールする。Adapter または target が、このポイント後にトランザクションの状態を保有する必要があるなら、それはトランザクション・オブジェクトのコピーを取らなければならない

Figure 13

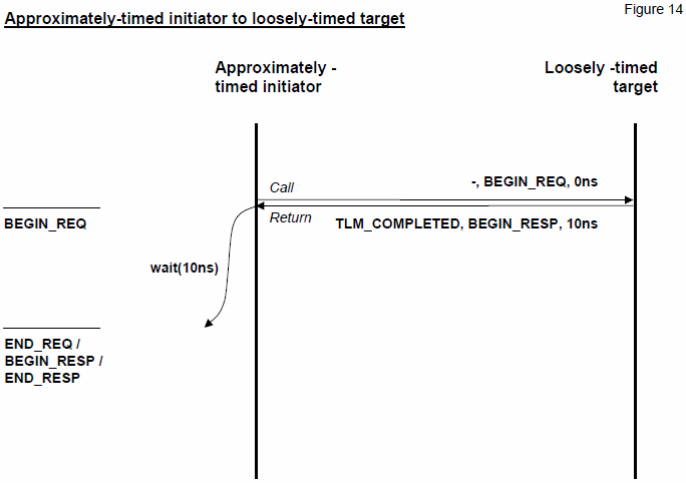


4.2.13.11. Approximately-timed initiator to loosely-timed target

- tlm_phase と一緒に使用される non-blocking transport interface は、直接、または、アダプタを使用することで approximately-timed initiator が loosely-timed target に接続されるのを可能にします。これが実際に役に立つかどうかは、initiator モデルの内容によるが、メカニズムは原則として動作する。
- loosely-timed target はリターンのときに BEGIN_REQ から BEGIN_RESP フェーズまでのトランザクションのためにことによると遅延をアノテートして nb_transport からトランザクションを終了するかもしれません。nb_transport からのリターンのときに、initiator は与えられた遅延を中断して、BEGIN_RESP へのトランザクションと同時に起こる暗黙の END_REQ フェーズを仮定するはずである。 target は、トランザクションが完了しているのを示したので、 initiator は、END_RESP を target に送るために再び nb_transport にコールをしてはいけないので、それで、END_RESP へのトランザクションも暗黙である。

initiator は 3 タイミング・ポイントを同時に扱えるほど強健でなければならない。なぜなら、 target が loosely-timed にすぎないので、 initiator のために、起こりうるタイミング不正確に通じる遅延の受け入れとターゲットのレイテンシ間を見分けるための方法が全くない。

Figure 14



4.2.14. Transaction lifetime example

- 通常、トランザクションのライフタイムは以下のようになります。
- トランザクションは、initiator によって作成され、nb_transport の引数としてバスブリッジを表す相互接続コンポーネントにパスされます。そして、TLM_ACCEPTED として返ってきます。
- そのブリッジ・コンポーネントは nb_transport の引数として順番にトランザクションを target に渡します。そして、また TLM_ACCEPTED として返ってきます。
- その後、target はトランザクションを完了でき、TLM_COMPLETED を返す nb_transport を target と initiator 間の後方のパスを通して nb_transport の引数としてトランザクションをブリッジに戻します。
- ブリッジは、ブリッジと initiator 間を後方のパスを通す nb_transport の引数として、TLM_COMPLETED のトランザクションを initiator に返します。
- 最終的に、initiator はその後の再利用のためにメモリ・プールの中のトランザクション・オブジェクトをメンテします。

4.3. ダイレク・メモリ・インタフェース

4.3.1. イントロダクション

ダイレクト・メモリ・インターフェース（DMI）は、イニシエータがターゲット内のメモリ領域に直接アクセスできる手段を提供する。つまりトランスポート・インタフェースを使わずに直接ポインタを使ってメモリアクセスが可能である。DMI によって、イニシエータからインターコネクト・コンポーネントに接続されたターゲットへの複数の transport 関数や nb_transport 関数の呼び出しがバイパスされるため、イニシエータとターゲット間のメモリアクセスを高速化することが可能となる。

ダイレクト・メモリ・インタフェースには２つのインタフェースがあり、イニシエータからターゲットへのフォワード・パス呼び出しとターゲットからイニシエータへのバックワード・パス呼び出しである。フォワード・パスは、ある与えられたアドレスに対する DMI アクセス（読み込みまたは書き込み）の特定モードを要求するために利用され、DMI 領域の境界を含む tlm_dmi 型の DMI 記述子へのリファレンスを返す。バックワード・パスは、フォワード・パスによって確立されたＤＭＩポインタが無効な場合にターゲットから呼び出される。

DMI 記述子はイニシエータによって使われるレイテンシ値を返し、ルーズリー・タイムド・モデリングに十分なタイミング精度を持つ。

DMI ポインタはデバッグ用途にも使えるが、通常デバッグのトラフィックよりもメモリアクセスのトラフィックの方が支配的であるため、DMI ではなくデバッグ・トランザクション・インタフェースで十分である。DMI ポインタをデバッグ用に使う場合、レイテンシ値は無視すべきである。

4.3.2. クラス定義

```
namespace tlm {
        // Defined in tlm_helpers.h
        enum tlm_endianness {TLM_UNKNOWN_ENDIAN, TLM_LITTLE_ENDIAN, TLM_BIG_ENDIAN};

class tlm_dmi_mode
{
public:
        enum Type { READ = 0x1, WRITE = 0x2, READ_WRITE = READ|WRITE };
        Type type;
};

class tlm_dmi
{
```

```
public:
        tlm_dmi() { init(); }
        void init();
        unsigned char* dmi_ptr;
        sc_dt::uint64 dmi_start_address;
        sc_dt::uint64 dmi_end_address;
        sc_core::sc_time read_latency;
        sc_core::sc_time write_latency;
        tlm_endianness endianness;
};

template <typename DMI_MODE = tlm_dmi_mode>
class tlm_fw_direct_mem_if : public virtual sc_core::sc_interface
{
public:
        virtual bool get_direct_mem_ptr( const sc_dt::uint64& address,
        DMI_MODE& dmi_mode,
        tlm_dmi& dmi_data) = 0;
};

class tlm_bw_direct_mem_if : public virtual sc_core::sc_interface
{
public:
        virtual void invalidate_direct_mem_ptr(sc_dt::uint64 start_range, sc_dt::uint64 end_range) = 0;
};
} // namespace tlm
```

4.3.3. get_direct_mem_ptr メソッド

a) get_direct_mem_ptr メソッドの呼び出しは、イニシエータまたはインターコネクト・コンポーネントからのみであるべき。
b) address 引数は DMI アクセスを要求するアドレスであるべき。
c) dmi_mode 引数は、イニシエータによって作成された DMI モードオブジェクトをリファレンスすべき。DMI モードオブジェクトは要求されている DMI アクセスのモードを示すべき。この引数のデフォルトは、イニシエータが要求しているのは読み込みアクセスか、それとも書き込みアクセスかを示すメンバー変数を含む tlm_dmi_mode クラスである。
d) dmi_data 引数は、イニシエータによって作成された DMI 記述子へのリファレンスであるべき。
e) インターコネクト・コンポーネントではイニシエータからターゲットへのフォワード・パスに沿って get_direct_mem_ptr メソッド呼び出しを通過させるべきである。言い換えれば、インターコネクト・コンポーネントのターゲット・ソケットに対する get_direct_mem_ptr メソッドの実装は、アドレスでコードや必要に応じてアドレスを修正するイニシエータ・ソケットの get_direct_mem_ptr メソッドを呼び出してもよい。例えば、インターコネクト・コンポーネントアドレスでは、ターゲットのアドレス幅やメモリマップの場所によって、アドレスをマスク（ビット幅の削減など）が必要。
f) アドレッシングエラーが発生したときには、インターコネクト・コンポーネントは get_direct_mem_ptr メソッドを通過させる必要はない。
g) インターコネクト・コンポーネントは dmi_mode と dmi_data 引数はフォワード方向において書き換えなしで通過すべき。ただしバックワード方向においては DMI モード及び DMI descriptor は修正してもよい。
h) ターゲットが与えられたアドレスへのＤＭＩアクセスが可能であれば、DMI descriptor のメンバー及び true を関数の返り値としてセットすべき
i) ターゲットが与えられたアドレスへのＤＭＩアクセスが不可能であれば、アドレス範囲とＤＭＩ descriptor の型のメンバーだけをセットすべきであり、false を関数の返り値としてセットすべき。

4.3.4. DMI_MODE テンプレート引数と tlm_dmi_mode クラス

a) tlm_fw_direct_mem_if テンプレートは DMI モードクラスの型でパラメタライズされる。
b) DMI モードクラスは、ターゲットに対して要求されている DMI アクセスのモードに伝えることを意図している。
c) イニシエータは DMI モードオブジェクトを管理する。
d) DMI_MODE テンプレート引数のデフォルト値は、tlm_dmi_mode クラスであり、これにはイニシエータが読み込みアクセスか書き込みアクセスを要求しているのかが示されている。
e) type 変数はイニシエータによって設定され要求されている DMI アクセスの種類を示している。ターゲットによって READ または WRITE アクセスから READ_WRITE アクセスに変更するために type 変数は修正してもより。イニシエータでは、READ_WRITE アクセスに設定できない。インターコネクト・コンポーネントは、バックワード・パスの場合のみ、READ_WRITE を READ または WRITE に上書きすることでアクセス種類を制限することができる。

f) tlm_dmi_mode のかわりに別のクラスを使う場合がある。例えば、DMI モードが要求を生成する CPU の種類に依存した DMI 要求を扱うために CPU ID を含むかもしれない。
g) あるアプリケーションでは他の DMI モードクラスを使う必要があっても DMI が読み込みなのか書き込みなのかを区別する必要があるので、新しく DMI モードクラスを作る際には tlm_dmi_mode を継承することを推奨する。
h) イニシエータは DMI モードオブジェクトを使ってターゲットによってアクセスが許可された DMI アクセスのモードを使うことだけ責任がある。

4.3.5. tlm_dmi クラス

a) DMI descriptor は tlm_dmi クラスのオブジェクト。DMI descriptor は、イニシエータで生成されるが、そのメンバーはインターコネクト・コンポーネントまたはターゲットで設定される。イニシエータは DMI descriptor を再利用してもよい。その場合イニシエータは get_direct_mem_ptr 関数を呼び出す間に init メソッドを呼び出し再度初期化する。
b) インターコネクト・コンポーネントは DMI descriptor をフォワード・パスで修正してはいけないので、DMI descriptor はターゲットで受け取ったときに初期状態にあるべき。
c) init メソッドはメンバーを初期化する。

4.3.6. invalidate_direct_mem_ptr メソッド

a) invalidate_direct_mem_ptr メソッドはターゲットまたはインターコネクト・コンポーネントだけににによって呼び出される。
b) ターゲットはＤＭＩ領域またはアクセスタイムの有効無効を修正する変更の前に invalidate_direct_mem_ptr メソッドを呼び出す必要がある。例えば、あるＤＭＩのアドレス領域が制限される前または、READ_WRITE から READ へのアクセスタイプの変更前または、アドレスのリマップ前に呼び出される必要がある。
c) start_range と end_range の引数は無効なＤＭＩ領域の最初と最後のアドレスをそれぞれ示す。
d) イニシエータが invalidate_direct_mem_ptr を受け取ったらすぐに与えられたアドレス領域とオーバーラップするＤＭＩ領域（get_direct_mem_ptr で事前に獲得していたもの）を無効にして廃棄する
e) 部分的にオーバーラップしている場合、つまり存在するＤＭＩ領域の一部分だけ無効な場合には、イニシエータは存在する領域の境界を調整するまたは全体を無効化する
f) インターコネクト･コンポーネントは invalidate_direct_mem_ptr 呼び出しをバックワード・パスに沿って通過させる義務がある。
g) 複数のイニシエータが同一ターゲットに対してダイレクト・メモリ・アクセスを実施ので、インターコネクト・コンポーネントがそれぞれのイニシエータに対して、invalidate_direct_mem_ptr を呼び出す実装が安全である。
h) start_range を 0､end_range を sc_dt::uint64 の最大値に設定するとイニシエータのすべてのダイレクト･メモリのポインタがインターコネクト・コンポーネントで無効にできる。
i) invalidate_direct_ptr は get_direct_mem_ptr を直接または間接的に呼び出すことはない実装である。

4.3.7. DMI Hint を利用した最適化

a) DMI hint は DMI アクセスの繰り返しポーリングをなくすことによってシミュレーションスピードを最適化するメカニズムである。DMI ポインタが使える状態か確認する際、get_direct_mem_ptr を呼び出すかわりに、イニシエータで DMI hint を確認することができる。
b) 汎用ペイロードによって DMI hint は提供される。ユーザ定義のトランザクションによって同じメカニズムを実装できる。その場合ターゲットによって DMI hint の値を適切にセットすべきである。
c) DMI hint を使うかどうかオプションである｡イニシエータは汎用ペイロードの DMI hint を無視することは自由である。
d) DMI をイニシエータが使う場合の推奨手順は下記
   - イニシエータが有効な DMI 領域のキャッシュに対するアドレスをチェックする。
   - DM ポインタが存在しないなら、イニシエータはトランスポート・インタフェースを通して通常のトランザクションを実行する
   - その後、イニシエータはトランザクションの DMI hint をチェックする。
   - hint が DMI を許可されていることを示していれば、イニシエータは get_direct_mem_ptr メソッドを呼び出す。

4.4. デバッグ･トランザクション･インターフェース

4.4.1. イントロダクション

デバッグ・トランザクション・インターフェースによって、遅延や wait,event notification または通常のトランザクションに対する副作用を一切発生させないでターゲットのストレージに対して読み込み及び書き込みが可能である。デバッグ・トランザクション・インターフェースは、トランスポート・インタフェースと同じパスを通るため、デバッグ・トランザクション・インターフェースの実装は通常のトランザクションと同じアドレス変換を実行する。

例えば、デバッグ・トランザクション・インターフェースによって ISS に接続されたソフトウェアデバッガがシミュレーションされるシステム内のメモリ内のデータを読み出したり、書き込んだりできる。またデバッグ・トランザクション・インターフェースによってイニシエータがシステムメモリの内容をシミュレーション中に解

析目的でスナップショットを取ることやシステムメモリのある領域を初期化することも可能である。

4.4.2. クラス定義

```
namespace tlm {
class tlm_debug_payload
{
        public:
        sc_dt::uint64 address;
        bool do_read;
        unsigned int num_bytes;
        unsigned char* data;
};
class tlm_transport_dbg_if : public virtual sc_core::sc_interface
{
        public:
        virtual unsigned int transport_dbg(tlm_debug_payload& r) = 0;
};
} // namespace tlm
```

4.4.3. ルール

a) transport_dbg 関数は、通常のトランザクションで使われるトランスポート・インタフェースと同じパスに続いて呼び出す。

b) イニシエータはデバッグペイロードオブジェクトを生成し、transport_dbg メソッドの引数で渡す前にオブジェクトのメンバーをセットしておく。

c) メンバー変数の address は読み込みまたは書き込みされる領域内の最初のアドレスをセットする。address 変数はインターコネクト・コンポーネントによって修正してもよい例えば、インターコネクト・コンポーネントがターゲットのアドレス幅に合わせるためにビットマスクする場合など。

d) メンバー変数は複数のインターコネクトを通過する際にその都度修正可能。

e) do_read 変数はターゲットから読み込む場合は true で、書き込む場合は false

f) num_bytes 変数は、読み込みまたは書き込む時のバイト数がセットされ、読み込みも書き込みもしない場合は 0 がセットされる。

g) data 変数は、読み込まれたデータ及び書き込むデータをコピーした配列のアドレス。この配列はイニシエータで宣言され、transport_dbg 関数が完了する前には delete してはいけない。

h) ターゲットの transport_dbg 関数は与えられアドレスを使って与えられたバイト数だけ読み込みまたは書き込みをする実装。

i) data 配列は通常はホストマシンのエンディアン。transport_dbg 関数の実装でターゲットとホストマシンのエンディアンの整合性を保障しなければならない。

j) transport_dbg 関数は実際に読み書きしたバイト数の合計を返り値として返す。その場合、num_bytes 変数よりも少なくなる。読み込み及び書き込みできなかった場合は、0 を返す。

k) transport_dbg 関数は wait 関数、event notification、他のいかなる影響もターゲットまたはインターコネクト・コンポーネントに対して影響を与えない。

5. ソケットと結合されたインタフェース

5.1. イントロダクション

■ ソケットとは：port と export を組み合わせたもの

イニシエータ・ソケット	Forwardpath 用 port Backward path 用 export
ターゲット・ソケット	Forwardpath 用 export Backward path 用 port

■ 特徴
- SystemC のポートバインド用のオペレータをオーバーロード
  - 階層的にバインドする場合は、順番に注意が必要

■ 機能
- TLM2.0 が提供する以下のコア・インタフェースを提供
  - Transport
  - DMI
  - デバッグトランザクション I/F
- TLM1.0 の機能は含まない

■ ソケットの利点

a) forward パス/backward パスそれぞれの、トランスポート、ダイレクト・メモリ I/F、デバッグ I/F を、1 つのオブジェクトにグループ化している。

b) 単一の関数呼び出しにより、forward パス/backward パスの port と export をそれぞれバインド可能。

c) 強固な型チェック機能を提供し、ソケットをバインドする際の型の不一致をチェック可能。

d) バス幅のパラメタを提供しており、トランザクションを中断する目的で利用可能。

5.2. クラス定義

■ I/F クラス

− forward/backward I/F、blocking/non-blocking I/F でグルーピングした以下の通信クラスを提供

I/F クラス名	forward/backward	blocking/ non-blocking	グループ化される I/F
tlm_fw_nb_transport_if	forward	non-blocking	tlm_nonblocking_transport_if tlm_fw_direct_mem_if tlm_transport_dbg_if
tlm_bw_nb_transport_if	backward	non-blocking	tlm_nonblocking_transport_if tlm_bw_direct_mem_if
tlm_fw_b_transport_if	forward	blocking	tlm_blocking_transport_if tlm_fw_direct_mem_if tlm_transport_dbg_if
tlm_bw_b_transport_if	backward	blocking	tlm_bw_direct_mem_if

■ プロトコルタイプクラス

− 各 I/F はプロトコルタイプクラスにより、トランザクション・オブジェクトの型を定義する。

− プロトコルタイプクラスは以下の 3 つのパラメタを持つ。

tlm_generic_payload	ペイロードの型
tlm_phase	転送フェーズ
tlm_dmi_mode	DMI モード

− デフォルトプロトコルタイプとして、tlm_generic_payload_types が提供されている。

− 新規にプロトコルを定義する場合には、各 I/F を新しいプロトコルタイプクラスによりパラメタライズする必要がある(汎用ペイロードを利用するかどうかに関わらず)。

■ ソケットクラス

− forward/backward I/F、blocking/non-blocking I/F でグルーピングした通信クラスを提供

tlm_initiator_socket tlm_target_socket	イニシエータ用ソケット ターゲット用ソケット

− さらに、テンプレート引数の組み合わせでインタフェースを明確化した、補助クラスも提供。

tlm_nb_initiator_socket	イニシエータ用ノンブロッキングソケット
tlm_b_initiator_socket	イニシエータ用ブロッキングソケット
tlm_nb_target_socket	ターゲット用ノンブロッキングソケット
tlm_b_target_socket	ターゲット用ブロッキングソケット

■ テンプレート引数

− tlm_initiator_socket、tlm_target_socket ともに、以下のテンプレート引数を持つ。

引数	デフォルト値	内容
BUSWIDTH	32	転送データ幅のワード長をビットで示す
FW_IF	tlm_fw_nb_transport_if <tlm_generic_payload_types>	Forward path に用いる I/F tlm_initiator_socket から tlm_target_socket に対してアクセスするための API を提供。
BW_IF	tlm_bw_nb_transport_if <tlm_generic_payload_types>	Backward path に用いる I/F tlm_initiator_socket から tlm_target_socket に対してアクセスするための API を提供。

■ API

− tlm_initiator_socket、tlm_target_socket は、それぞれ以下の API を持つ。

tlm_initiator_socket	tlm_target_socket

unsigned int get_bus_width() const;
void bind(target_socket_type&);
void operator() (target_socket_type&);
void bind(tlm_initiator_socket&);
void operator() (tlm_initiator_socket&);
void bind(BW_IF &);
void operator() (BW_IF&);
	unsigned int get_bus_width() const;
	void bind(initiator_socket_type&);
	void operator() (initiator_socket_type&);
	void bind(tlm_target_socket&);
	void operator() (tlm_target_socket&);
	void bind(FW_IF&);
	void operator() (FW_IF&);
	BW_IF* operator->();

5.3. ルール

a) tlm_initiator_socket クラスでは、コンストラクタで指定されたキャラクタストリングを、ベースクラスである sc_port のコンストラクタに対して引き渡し、インスタンス名として設定する。また、同じストリングに”_export”を付加した名前を、backward パスの sc_export に対してインスタンス名として設定する。これらは sc_gen_unique_name の呼び出しにより設定される。

→ sc_port : foo

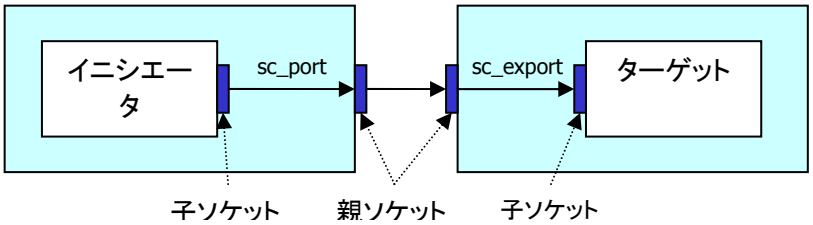
→ sc_export : foo_export

b) tlm_target_socket クラスでは、コンストラクタで指定されたキャラクタストリングは、ベース

クラスである sc_export のコンストラクタに対して引き渡し、インスタンス名として設定する。また、同じストリングに”_port”を付加した名前を、sc_port に対してインスタンス名として設定する。これらは sc_gen_unique_name により設定される。

→ sc_export : foo
→ sc_port : foo_port

c) get_bus_width は、テンプレート引数の BUSWIDTH 値を返す。
d) BUSWIDTH は、ソケットを介して転送されるデータワード長をビット単位で表す。バースト転送においては、バースト転送における各ビートのビット数を示す。BUSWIDTH の解釈は、トランザクション型に依存する。
e) ソケットは、同じ BUSWIDTH を持つソケットとのみバインドできる。
f) bind と()オペレータは、ともにソケットを引数にとり、指定されたソケットインスタンスを指定元のソケットとバインドする。
g) bind と()オペレータは、ともにソケットを引数にとり、指定元のソケットインスタンスの export を、引数で指定されたチャネルインスタンスとバインドする。
h) イニシエータ・ソケットをターゲット・ソケットにバインドする際、bind メソッドと()オペレータは、イニシエータ・ソケットの port をターゲット・ソケットの export へ、ターゲット・ソケットの port をイニシエータ・ソケットの export へバインドする(同一階層の場合)。
i) イニシエータ・ソケットとターゲット・ソケットは、どちらかの bind または()オペレータにより、相互にバインドすることができる。
j) イニシエータ・ソケット間、またはターゲット・ソケット間でバインドする場合、bind メソッドと()オペレータは、port と port、export と export をそれぞれバインドする。この機能は、階層間の接続に利用する。すなわち、階層の下位から上位(子ソケットから親ソケット)、上位から下位(親ソケットから子ソケット)へトランザクションを渡すことができる。
k) 階層間のソケットのバインドは、正しい順番で行われなければならない。イニシエータ・ソケット間のバインドを行う場合、子ソケットから親ソケットに対してバインドしなければならない。ターゲット・ソケット間でバインドする場合、親ソケットから子ソケットに対してバインドしなければならない。このルールは、tlm_initiator_socket が sc_port を継承しており、tlm_target_socket が sc_export を継承していることによる。port は階層の下位から上位に対してバインドされる必要があり、トップ階層では port から export へ、export から export へは階層の上位から下位にバインドされなければならない。



l) 二つのソケットをバインドするには、forward path、backward path ともに、同じインタフェース型、バス幅、プロトコルタイプクラスを持つ必要がある(デフォルトは tlm_generic_payload_types)。ソケット間の型チェックを強化するには、プロトコルが generic payload 型かどうかに関わらず、新しいプロトコル型のクラスを定義する必要がある。
m) ターゲット・ソケットの->オペレータは、ターゲット・ソケットの backward path のポートの->オペレータを呼び出し、その->オペレータの戻り値を返す。
n) 以下のソケットは、tlm_initiator_socket, tlm_target_socket に優先して使われるべきである。
  - tlm_nb_initiator_socket, tlm_b_initiator_socket
  - tlm_nb_target_socket, tlm_b_target_socket

6. アナリシス・インタフェースとアナリシス・ポート

アナリシス・ポートは、複数のコンポーネントにまたがるトランザクションに対する解析機能、例えば、機能の正確性をチェックするとか機能カバレージ情報を取得するなどを実現するために使用される。

アナリシス・ポートのもっとも大切な点は、一つのアナリシス・ポートが複数のチャネル(サブスクライバー)にバインド可能であり、アナリシス・ポートに対して write()をコールしたときにそれぞれのサブスクライバーに対して同一の write()をコールさせるようにすることである。アナリシス・ポートは 0 以上複数のサブスクライバーに対してバインドすることができ、バインドしなくても構わない。

それぞれのサブスクライバーは、tlm_analysis_if の write()メソッドを実装する。このメソッドは const reference によってトランザクションに渡され、サブスクライバーはただちに処理を実行する。もし、サブスクライバーがトランザクションのライフタイムを延長しておきたいような場合には、自分でトランザクション(generic_payload)に対する deep_copy()メソッドを行い、サブスクライバー自身がコピーして作成したトランザク

ションの持ち主になり自分でメモリ管理を行うことになる。

アナリシス・ポートはモデルの動作そのものに関わるようなことに対して用いてはならず、その他の解析等の側面的な解析機能のみで使用しなければならない。 tlm_analysis_if インタフェースは、tlm_write_if を継承したクラスである。この tlm_write_if は、アナリシスの目的だけではなく、他の目的にも使用可能である。例えば、第 8 章の Payload Event Queue においても用いられている。

TLM2 キットには、tlm_analysis_fifo という、段数が無限の tlm_fifo に対して t l m_alansysis_if を用いてトランザクションを write できるようにするものが含まれている。tlm_fifo は、tlm_analysis_triple というトランザクションと、開始時間、終了時間を含むものもサポートしている。

6.1. クラス定義

＜省略＞

6.2. ルール

a) tlm_write_if と tlm_analysis_if は、単方向、ネゴシエーション無し、ノンブロッキングのインタフェースであり、即座に応答しなければならない。

b) tlm_analysis_port のコンストラクタは、ポートのインスタンス名の文字列を引数として使用する。これは、ベースクラスの sc_object に渡される。

c) bind メソッドは、サブスクライバーのアナリシス・ポートのインスタンスを登録し、write メソッドが呼ばれたときに登録されたサブスクライバー内の write メソッドが呼ばれる。複数のサブスクライバーを単一のアナリシスポートインスタンスに登録することも可能。

d) オペレータ () は、bind()メソッドと同一。

e) サブスクライバーの数が 0 であっても構わない。その場合には、write()メソッドは伝達されない。

f) unbind()メソッドは、bind()の逆であり、サブスクライバーのリストから削除される。

g) tlm_analysys_port の write()メソッドが呼ばれた場合、そのポートに登録されたすべてのサブスクライバーに対して、write()メソッドが const リファレンスの引数と共に呼ばれる。

h) write()メソッドはノンブロッキングであり、その実装の中に wait()を含んではいけない。

i) write()メソッドは、const reference によって渡されたトランザクションの内容や、その中のポインタ先のデータ(data,バイトイネーブル配列)を書き換えてはいけない。

j) もし、write()メソッドが return する前にそのトランザクション・オブジェクトに対しての処理を完了することができない場合には、新しいトランザクション・オブジェクトを deep_copy()メソッドによってコピーしてから実行する。コピーされたオブジェクトのメモリ管理はサブスクライバーが責任を負う。

k) tlm_analysys_fifo クラスのコンストラクタは、bound されてない tlm_fifo を構成する

l) tlm_analysys_fifo クラスの write()メソッドは tlm_fifo ベースクラスの nb_put()を同一の引数とともに呼び出す。

7. クォンタム・キーパー

7.1. イントロダクション

テンポラル・デカップリング機能は、SystemC のプロセスに対して、クォンタムと呼ばれる時間を実際のシミュレーション時刻より先に進めるものである。LT のコーディングスタイルにおいて、nb_transport()の引数によって渡される時間は、sc_time_stamp()による実際の現在時刻に対する相対時間として定義される。

テンポラル・デカップリング機能を用いると、スレッドのコンテキストスイッチとイベントを減らすことにより非常に高速なシミュレーション速度が実現できる。

tlm_quantumkeeper クラスには、クォンタム時間に対して制御するためのメソッドが用意されている。

テンポラル・デカップリング機能を用いる時には、一定のコーディングスタイルを保つために、あらかじめ定義されている quantum keeper クラスを用いることを強く推奨する。しかしながら、SystemC カーネルそのものにこの機能を実装するということも可能である。

テンポラル・デカップリングの詳細については、3.3.2 章 LT スタイルとテンポラル・デカップリングも参照のこと。

タイミング･アノテーションとノンブロッキングトランスポートインターフェースに関しては､4.2.8 章 sc_time 引数の項目も参照のこと。

7.2. クラス定義

<省略>

7.3. ルール

a) tlm_quantumkeeper クラスによって単一のグローバルなクォンタム時間(同期間隔)が定義される。これがクォンタム時間のデフォルトとなる。テンポラル・デカップリングをサポートしているイニシエータモデルは、おおよそこのクォンタム時間が来るごとに実際のシミュレーション時間との間で同期が図られる。ただし、ターゲットの要求によってはそれよりも頻繁に同期が実行される場合もある。

b) それぞれのイニシエータモデルはそれぞれ別のクォンタム時間を持つことも可能であるが、一般的には全てのイニシエータモデルに同一のクォンタム時間を設定する。同期の回数が少なくてもすむイニシエータは他よりクォンタム時間を長く設定できるわけではあるが、一般的には、同期の回数が多く必要なモデルがもっともシミュレーション時間に大きく影響してしまうからである。

c) 最大のシミュレーション速度を実現するためには、全てのイニシエータモデルがテンポラル・デカップリングを用い、他の SystemC プロセスは全く存在しないか、必要最小限に押さえる必要がある。

d) 理想的には、テンポラル・デカップリングを持つイニシエータモデルのみに、SystemC プロセスがあり、それぞれの SystemC プロセスにおいては、時間を先に進めておき、クォンタム量を超えた時のみシミュレーションカーネルに戻る(Yield)ようにする。
e) Yield とは、SC_（C)Thread の場合には、wait()を実行すること、SC_Method の場合には、関数から return することを意味する。
f) テンポラル・デカップリングといえども、あくまでも標準的な SystemC カーネル上で動作するものであるので、イベントはスケジュールできるし、プロセスは停止、再開することもでき、LT のコーディングスタイルは他のコーディングスタイルと混在できる。
g) すべてのイニシエータがテンポラル・デカップリングを使用しなければいけないというわけではない。使用しているプロセスと使用していないプロセスは混在可能である。しかし、そうすると、スピードが落ちてしまう。
h) それぞれのプロセスは、ローカルな処理時間や通信にかかる時間を加算していくが、この時間のことをローカルオフセット時間と呼ぶ。
i) sc_time_stamp()を呼んだときにはあくまでも現在のクォンタムが開始した実シミュレーション時間を返す。
j) SystemC カーネルは、ローカルオフセット時間については関知しない。nb_transport()を用いた時には、ローカルオフセット時間は、引数によって書き戻される。
k) ローカルでないオブジェクトの値を読み出した場合には自他のプロセスで変更しない限り、あくまでもクォンタムの開始した実際の時間における値を返す。とくに、sc_signal の値はアップデートされない。
l) クォンタム・キーパーのコンストラクタは、ローカルオフセット時間を 0 に設定する。また、compute_local_quantum メソッドを呼び出す。
m) set_global_quantum()メソッドは、グローバルクォンタム時間を引数によって設定する。しかしローカルのクォンタム時間は変更されない。get_global_quantum()メソッドはグローバルクォンタム時間を返す。set_global_quantum()を呼んだ後には、reset()メソッドを呼び出し、ローカルクォンタム時間を再計算させることが望ましい。
n) get_local_time()メソッドはローカルオフセット時間を返す。
o) get_current_time()メソッドは、ローカル時間、すなわち、sc_time_stamp()+ローカルオフセット時間を返す。
p) inc()メソッドは、ローカルオフセット時間を引数によって渡しローカル時間を実時間よりも先に進める。
q) need_sync()メソッドは、現在のローカルオフセット時間がローカルクォンタムより大きい時に true を返す。
r) ローカルクォンタムは、イニシエータが同期する前にどれだけの時間が残っているかの時間を意味する。つまり、クォンタム量より、最後に同期したあとから、inc()によって渡された時間の総和を引いた物となる。訳注 つまり、inc()を呼び出すとごとに、減っていく
s) sync()メソッドは、wait(ローカルオフセット時間)を呼び出し、実際のシミュレーション時間とローカルの時間との同期をとり、reset()メソッドを呼び出す。
t) reset()メソッドは、compute_local_quantum()を呼び出し、ローカルオフセット時間を 0 に戻す。
u) compute_local_quantum()は、ローカルクォンタムの値を再計算してその値を返す。デフォルトの実装は、実シミュレーション時間より、グローバルクォンタムの倍数の実シミュレーション時間より大きい値となる。この関数はユーザが変更可能である。
   - 訳注:グローバルクォンタムを 1000 ns ごとに同期するとした場合、ロ-カル時間が 3900ns、実時間が 3000ns のとき、inc(200ns)をおこなうと,4100ns となり、4000ns を超えてしまうので、need_sync()は、true を返す。ここで sync()を実行すると、実時間が 4100ns になり、ローカルクォンタムは、5000 ns -4100 ns = 900 ns となる。つまり 900 ns が次に同期するまでの残った時間になる。
v) tlm_quantum_keeper クラスはクォンタム・キーパーのデフォルトの実装となり、このクラスを継承したクラスを作り、compute_local_quantum メソッドを書き換えることが可能ではあるが、普通ではない。
w) ローカルオフセット時間がローカルクォンタム以上になった場合には、プロセスを yield させる必要があるが、自分で wait()を呼び出さないで、sync()メソッドを呼び出すことを強く推奨する。
x) ローカルクォンタム量を超えた場合に同期を自動的に実行するようなしかけはない。したがって、イニシエータは自分で need_sync メソッドを呼び出し、必要に応じて sync()を呼び出す必要がある。
y) イニシエータがローカルクォンタム量を超えるよりも前に実行をサスペンドして、実時間を先に進んでしまっているローカル時間に追いつかせる必要が生じた場合、sync()を呼び出すか、あるいは明示的に別のイベントを wait させてもいい。途中で同期させるこの方法を、 sync-on-demand(要求に応じた同期)と呼ぶ。
   - 訳注：nb_transport()の戻り値が TLM_ACCEPTED の場合には、この sync-on-demand を必ず実行しなければならない。なぜなら、この場合、wait()をコールし、ターゲットに処理を移させて、ターゲットが、別途 nb_transport()をイニシエータに対して呼ぶようにしなければいけないからである。いいかえれば、TLM_ACCEPTED をターゲットが返すということは、イニシエータに対して、仮の

時間だけを進めることをあきらめさせ、無理矢理 wait()を実行させることによってイニシエータを実時間に同期させ、自分の方に処理を戻して、自分も wait()なりを実行したいと通知することを意味する。

z) クォンタム量として設定する時間、つまり同期する時間間隔は、複数のイニシエータ同士において通信しあう必要のある間隔よりも短くなければいけない。そうしないと大切なプロセス間の相互作用(訳注：インタラプト等)が行えず、モデルが動作しなくなってしまう。

aa) ブロッキング・インタフェースとノンブロッキングインターフェースが混在している時、テンポラル・デカップリングを使用しているイニシエータは、直接又は間接的に b_transport()を呼び出す必要が生じる場合がある。とくにアダプタを用いる場合である。もし、ブロッキング・インタフェースが wait()を使用しているのであれば、それを呼び出す側は、当然ながら、SC_(C)Thread でなければならない。(SC_Method ではいけない)

bb) イニシエータが b_transport()を呼び出す場合、あるいはテンポラル・デカップリングに対応できない何らかのブロッキングメソッドを呼び出す場合には、事前に必ず同期させなければならない。つまり、時間を先に進めておくことをこの時点であきらめて、実時間の中で処理を進めていくことになる。

cc) イニシエータが頻繁にブロッキングコールを呼び出すような場合には、テンポラル・デカップリングを有効に活用することはできない。なぜなら、すべてのブロッキングコールの前には、かならず同期させないといけないからである。

8. Peyload event queue

ペイロードイベントキュー(PEQ)は、トランザクション・オブジェクトに関連付けられた SystemC のイベント通知用のキューである。それぞれのトランザクションは遅延と共に PEQ 内に格納され、現在のシミュレーション時間と設定された遅延時間の合計時間になった時点で PEQ から出力される。

tlm_peq クラスは TLM2.0 標準として決定していない。

PEQ は、ノンブロッキング転送 I/F の意味を理解するのに助けとなる概念的な重要性を目的としてこのドキュメントに含まれている。AT コーディングスタイルでノンブロッキング I/F を組上げる際、特に重要になってくる。しかしながら、ここで与えた特定の Payload event queue を使わずしても nb_transport を実現することは可能である。

トランザクションは、遅延時間を引数に delayed_write_export ライトメソッドにより PEQ へ挿入される。遅延時間は現在のシミュレーション時間との加算に使われ、トランザクションが PEQ の最後から出力する時間となる。トランザクションはら write_port ライトメソッドをコールすることで PEQ から出力される。

もしトランザクションが実行されるそれぞれのタイミングでプロセスを起動させたいのであれば、そのシンプルな解決策は、tlm_fifo に write_port をバインドすることである。

トランザクションは PEQ に挿入された順序ではなく、引数の遅延時間から計算された時間順に出力される。

もしいくつかのトランザクションが同時に出力されるようスケジューリングされた場合、それらはシングルデルタサイクル内で出力される。トランザクションを消失させたり、取り消すことは出来ない。

PEQ は tlm_write_if と tlm_delayed_write_if を使用しており、これらは TLM2 標準の一部である。ライト I/F は tlm_analysis_if をベースクラスとしている。

6 章の Analysis I/F や解析ポートを参照のこと(PEQ が例として含まれている)。

9. Generic Payload

9.1. Introduction

Generic Payload は MMB モデルの相互利用性を高めることを目的に、モデル間の通信に必要となる属性(コマンド、アドレス、データなど）を定義したクラスである。Generic Payload は全ての MMB プロトコルで必要となる属性を含んでいるわけではない。しかし、そのための拡張メカニズムが用意されている。拡張されたとしても Generic Payload は共通のベース型となり、異なるプロトコルを繋ぐブリッジ開発の工数低減に寄与する。

Generic Payload はイニシエータ・ソケット/ターゲット・ソケットの両方で使用されることが奨められる。Generic Payload をベースとしているか否かで異なるプトロコル型のチェックを行うことが出来る。

9.2. 拡張性と相互利用性

3 つの推奨する拡張手法がある。これらは blocking、non-blocking transport I/F のトランザクションテンプレート引数 TRANS と複数の I/F が組合されて定義された TYPES を置き換えることで実現するものである。

実現方法	TRANS トランザクション型	TYPES プロトコル型
A	tlm_generic_payload	tlm_genetric_payload_types
B	tlm_generic_payload	ユーザ定義型
C	ユーザ定義型	ユーザ定義型

A→B→C の順に相互利用性が損なわれる。これら 3 つの方法で実現したモデルは、一つのシステムで混在されることも考えられている。

9.2.1. 無視可能な拡張を含む、Generic Payload を直接利用

- トランザクション型は tlm_generic_palyload を、プロトコル型は tlm_generic_payload_types を指定する
- 拡張部分は"無視可能"とする。"無視可能"とはインターコネクト/ターゲット共この拡張に関してエラーや転送失敗などを起こさないこと
- コンパイル時に型のチェックを行うことが出来る

9.2.2. tlm_generic_payload の型を含む新しいプロトコル型の定義

- トランザクション型は tlm_generic_palyload であるが、プロトコル型は tlm_generic_payload_types を用いず、独自で定義した型を利用する
- コンパイル時の型チェックは行える
- 拡張された属性を"無視可能"と実現/義務化することは出来るが意味定義には十分な注意が必要
- 推奨される 2 つの利用パターンがある
- イニシエータ/インターコネクト/ターゲット全てが共通に同じ新しいプロトコル型を使用
- 末端だけが新しいプトロコルで、その間のモデルは generic_payload_types を使用
- この拡張は、各モデルがそれを知る/知らないに関わらず正しく転送が行えるよう考慮が必要

9.2.3. 新しいプロトコル型とトランザクション型の定義

- トランザクション型、プロトコル型共新しい型を定義/利用する
- この手法は実現するプロトコルと Generic Palyload が著しくことなる場合に利用する
- 相互利用性を考えるとこの手法よりは、前 2 つの手法を推奨する

9.3. Generic Payload 属性とメソッド

Generic Payload クラスはプライベートな属性とそれにアクセスするパブリックはメソッドを持つ。各属性の変更は原則イニシエータが行うが、アドレスやステータスなどはインターコネクトやターゲットが変更する場合がある。リードコマンドにおけるデータ配列はターゲットが更新する。

9.4. ソースコード

(省略)

9.5. メモリ管理

イニシエータはトランザクションオブヘクトのメモリ確保/管理する必要がある。これはスタティック/オート/ダイナミックアロケーションが可能である。イニシエータはトランザクションが完全に終了するまでそのオブジェクトを消去してはならない。このことはデータポインタやバイトイネーブルポインタ属性についても同様。tlm_generic_payload クラスは shallow コピー(ポインタのみコピー)と deep コピー(値全てをコピー)を持つ。後者に関してはそれをコールしたモジュールがメモリ管理を行う必要がある。

9.6. コンストラクタ、代入、デストラクタ

- コンストラクタ
  - デフォルトコンストラクタは、Generic Payload 属性にデフォルト値を設定する。
  - コピーコンストラクタは、データのコピーは行わずポインタのコピーを行う shallow コピーを行う
- 代入
  - コピーコンストラクタと同様代入は shallow コピーを行う
  - deep_copy メソッドは完全は複製を行う。
- デストラクタ
  - 仮想デストラクタではデータ配列や倍等ネーブル配列などの削除は行わない。またデストラクタがオーバーロードされることは想定していない。

9.7. デフォルト値と標準的な属性

属性	デフォルト値	インターコネクトでの変更	ターゲットでの変更
コマンド	TLM_IGNORE_COMMAND	No	No
アドレス	0	Yes	No
データポインタ	0	No	No
データ配列	-	No	Yes(read)
データ長	1	No	No
バイトイネーブルポインタ	0	No	No
バイトイネーブル配列	-	No	No
バイトイネーブル長	1	No	No
ストリーミング幅	0	No	No
DMI 許可	false	Yes	Yes
レスポンス・ステータス	TLM_INCOMPLETE_RESPONSE	No	Yes
拡張ポインタ	0	Yes	Yes

9.8. コマンド属性

- set_command メソッドは値渡しの引数でコマンド属性を設定し、get_command メソッドは現在のコマンド属性を値で返す。
- set_read メソッドと set_write メソッドはそれぞれ TLM_READ_COMMAND 、TLM_WRITE_COMMAND をコマンド属性に設定する。is_read メソッドと is_write メソッドはそれぞれ現在のコマンド属性の値が TLM_READ_COMMAND、TLM_WRITE_COMMAND かどうかを判定しそれを返す。
- リードコマンドはコマンド属性が TLM_READ_COMMAND と等しい Generic Payload トランザクションであり、ライトコマンドはコマンド属性が TLM_WRITE_COMMAND と等しい Generic Payload トランザクションである。

- リードコマンドを受取った時、ターゲットはローカル配列の内容をデータポインタ属性となるよう配列ポインタにコピーする。
- ライトコマンドを受取った時、ターゲットはデータポインタ属性により指定されている配列をターゲット内のローカル配列にコピーする。
- もしターゲットがリードまたはライトコマンドを実行することが出来ない場合は、標準的なエラーレスポンスを生成する。推奨するレスポンス・ステータスは TLM_COMMAND_ERROR_RESPONSE。
- Generic Payload トランザクションのコマンド属性が TLM_IGNORE_COMMAND であった場合、ターゲットはライトまたはリードコマンドを実行してはならない。
- コマンド属性はイニシエータにより設定され、インターコネクトやターゲットは書き換えない。
- コマンド属性のデフォルト値は TLM_IGNORE_COMMAND である。

9.9. Address Attribute

- set_address, get_address でアドレスの設定、取得する
- ターゲットは address_attribute の現在値が read/write の最初のバイトを示しているものと解釈する
- データ配列の指定されたインデックスのバイトデータが address_attribute + (array_index % streaming_width)で示されるアドレスへ／から転送される
- 転送できない場合はエラーを出す（推奨：TLM_ADDRESS_ERROR_RESPONSE）
- イニシエータで設定されたアドレスが上書きされた場合、前の値は失われる（アドレスオフセット計算などによる上書き）
- デフォルトは‘0’

9.10. Data pointer attribute

- set_data_ptr, get_data_ptr でデータポインタの設定／取得が出来る
- トランスポート・インタフェースで、オブジェクトなし（null）はエラー
- データの read/write は汎用ペイロードの他のアトリビュートに従ってターゲットがデータ配列からコピーする
- データ配列長はデータ長アトリビュート以上であること（単位はバイト）
- 値はイニシエータが設定して、他のコンポーネントやターゲットによって上書きされない
- データ配列の保存場所はイニシエータが確保する（レジスタファイル、キャッシュメモリ、バッファなど）
- write コマンドと TLM_IGNORE_COMMAND では、データ配列はイニシエータが書き込み、他から上書きされないこと
- read コマンドでは、データ配列はターゲットで上書きされるが、他のコンポーネントから上書きはできない
- データポインタ属性のデフォルト値は‘0’（null）

9.11. Data length attribute

- set_data_length, get_data_length でデータ長を設定、取得する
- ターゲットは、read/write コマンドではデータ配列へ／からコピーする byte enable attribute でディスエーブルされたバイトも含んだ長さバイト数として解釈する
- 値はイニシエータが設定して、他のコンポーネントやターゲットによって上書きされない
- ‘０’に設定してはいけない。ゼロバイト転送の場合は、コマンド・アトリビュートを’TLM_IGNORE_COMMAND’とする
- tlm_initiator_socket, tlm_target_socket クラスを使って、バースト転送をする場合、転送長はソケットのＢＵＳＷＩＤＴＨテンプレート・パラメータで指定されたビット長になる。
- データ長がＢＵＳＷＩＤＴＨ／８と同じか小さいとシングルワード転送、以上だとバースト転送となる。
- ターゲットはトランザクションが実行できなかった場合は、エラーメッセージ’TLM_BURST_ERROR_RESPONSE’を出してデータ配列の内容を変更してはいけない。
- デフォルト値は‘1’

9.12. Byte enable pointer attribute

- set_byte_enable, get_byte_enable で値を設定、取得する
- 各転送単位（beat）の有効バイト数より各転送単位のアドレス増分が大きい場合や、バスの指定されたバイトレーンに複数ワードデータの設定を行うようなバースト転送を行う場合に使用する。また高抽象レベルでは、データ配列に歯抜けがある状態でバースト転送を行う時に使用する
- ある決まったパターンを繰り返し使用する場合や、データ配列全体でパターンがある場合に、byte enable mask が定義される
- byte enable array の項目数は、byte enable length attribute で与えられる
- byte enable pointer が‘0’(null)に設定されている場合は、その転送では byte enable は使用しない
- byte enable attribute はイニシエータが設定する。enable 配列とその内容もイニシエータが確保して設定する。enable 配列の内容は他のコンポーネントやターゲットで上書きされない。
- byte enable pointer が null でないときは、ターゲットは対応する実装を持つか、もしくはエラーを出す。推奨エラー’TLM_BYTE_ENABLE_ERROR_RESPONSE’
- read/write コマンドでは、データ配列のディスエーブルされたデータは接続コンポーネントやターゲットでは無視されなければならない。
- ターゲットが転送データとローカル配列間でバイト単位のコピーをする場合、ディスエーブルされたデータを変更してはいけないし、その値を使っても行けない。

- 上記のルールを守ることができない場合は、新しくプロトコル・クラスを作ることを推奨する
- ターゲットが read コマンドで、byte enable を使わずにローカル配列から転送データに連続したブロックデータのコピーが可能であるが、イニシエータの管理領域にターゲットがデータを書き込むことになり危険である。
- デフォルト値は‘0’。ヌルポインタ

9.13. Byte enable length attribute

- set_byte_enable_length,get_byte_enable_length で値を設定、取得する
- read/write コマンドでは、ターゲットは、byte enable 配列の要素数として解釈する
- 値はイニシエータが設定して、接続コンポーネントやターゲットは上書きしない
- データ配列のある要素に適用される byte enable は
  byte_enable_array_index = data_array_index%byte_enable_length
  で与えられる
- byte enable length が data array length より大きい場合、余分な byte enable は read/write に影響しない
- byte enable pointer が null の場合、byte enable length は無視される
- ターゲットが、指定された byte enable length を使ってトランザクションが実行できないときは、エラーを出す。推奨は、TLM_BYTE_ENABLE_ERROR_RESPONSE
- デフォルト値は、‘1’

9.14. Streaming width attribute

- read/write に対して、ターゲットは現在の streaming width に従って動作する
- 各転送単位(Beat)のバイト数を指定する
- データをコピーするターゲットのアドレスはそれぞれの転送単位(Beat)のアドレス値に毎回リセットされる
- 非ゼロ‘0’のストリーム幅のトランザクションはオリジナルのトランザクションと同じアドレスを持つトランザクションと機能的に等価である。それぞれがオリジナルストリームと同じデータ長で同じバイトシーケンスを持つ。
- ストリーム幅が‘0’の場合はストリーム転送ではなく通常のデータ転送として扱われる
- データ配列長やデータ配列に格納されているバイト数には無関係である
- ターゲットが、指定された Streaming width を使ってトランザクションが実行できないときは、エラーを出す。推奨は、TLM_BURST_ERROR_RESPONSE
- Byte enable と組み合わせて使われるときがあり、その場合通常は byte enable length と同じ値を持つ
- デフォルト値は、‘0’

9.15. Endianness

- Generic payload を使うときのイニシエータとターゲットの互換性のルールを規定する
- tlm_initiator_socket と tlm_target_socket を使う場合、データ配列、バイトイネーブル配列の有効ワード長 W は(BUSWIDTH+7)/8 バイトで求められる
- データ配列のバイトオーダはホストのエンディアンを使うリトルエンディアンの場合はデータ配列 data[0]は LSB ワードでビックエンディアンホストの場合は、data[0]は MSB ワード
- 例えばデータ{a,b,c,d}が配列に入っているとき、リトルエンディアンホストでは、{LSB0,,, MSB0、LSB1,,, MSB1、LSB2,,, MSB2,,, }で、ビッグエンディアンでは、 {MSB0,,, LSB0、 MSB1,,, LSB1、MSB2,,, LSB2 ,,, }
- イニシエータ、ターゲットはホストと同じエンディアンを使うことを想定しており、シミュレーション速度に影響する
- アプリケーションはエンディアンに非依存であることを強く推奨する。
- もしイニシエータやターゲットがホストと異なるエンディアンを持つ場合は、変換する必要があり、ヘルパー関数が用意される(開発中)
- データ配列長、アドレス長は、有効ワード長の整数倍である必要はない
- データ配列アクセスにワード境界に合わせたワード単位での転送ができない場合はヘルパー関数で隠蔽する
- 部分ワードアクセスの場合はエンディアンを考慮して byte enable を設定する必要がある
  リトルエンディアン　address=0, W=4, data={1,2,3,4,5}
  ビッグエンディアン　address=0, W=4, data={4,3,2,1,x,x,x,5}, be={1111,0001}
- SOCKET を接続する場合、BUSWIDTH は合わせる必要があるが、異なる場合はエンディアンを考慮したバス幅変換を行う必要がある

9.16. Atomic Transactions

- Read-modify-write を実現するために、取り込むことを検討中

9.17. DMI allowed attribute

- G set_dmi_allowed, get_dmi_allowed で値を設定取得する
- 直接メモリ・ポインタを取得できる可能性をイニシエータに示すもので、DMI 経由で現在のトランザクションが実行できた場合は、ターゲットが’true’にセットする。4.3.7 参照
- デフォルト値は、false

9.18. Response status attribute

set_response_status, get_response_status で値を設定、取得する

- resoponse status=TLM_OK_RESPONSE の場合にのみ、is_response_ok が’true’を返す。response status ≠TLM_OK_RESPONSE の場合、is_response_error が’ true’を返す
- get_response_string は現在の状態を文字列で返す
- 原則としてターゲットは generic payload の全ての機能を実装すべきだが、そうでない場合はエラーを返すようにする
- イニシエータが TLM_INCOMPLETE_RESPONSE を設定して、ターゲットが上書きする。他のコンポーネントは上書きしてはならない。
- ターゲットは処理が成功した場合は、TLM_OK_RESPONSE を設定して、エラーの場合は以下の適切なものを設定する
- 適切なエラーが判断できない場合は、TLM_GENERIC_ERROR_RESPONSE を設定する
- デフォルト値は、TLM_INCOMPLETE_RESPONSE
- イニシエータはレスポンス・ステータスをいつも確認すべきである

Error response	Interpretation
TLM_ADDRESS_ERROR_RESPONSE	Unable to act upon the address attribute, or address out-of-range
TLM_COMMAND_ERROR_RESPONSE	Unable to execute the command
TLM_BURST_ERROR_RESPONSE	Unable to act upon the data length or streaming width
TLM_BYTE_ENABLE_ERROR_RESPONSE	Unable to act upon the byte enable
TLM_GENERIC_ERROR_RESPONSE	Any other error

9.18.1. The standard error response

- ターゲットが汎用ペイロードのトランザクションを受け取ったときには、次のいずれか一つの動作をすべきである
  - a) 汎用ペイロードのアトリビュートやモデル化されたコンポーネントのセマンティックスに従ったトランザクションのコマンドを実行して、レスポンス・ステータスを TLM_OK_RESPONSE に設定する
  - b) 先に示したエラーメッセージのうちの一つをレスポンス・ステータスとして設定する
  - c) SystemC レポートハンドラを使って SystemC の４つの重要度レベルでレポートを生成して、レスポンス・ステータスを TLM_OK_RESPONSE に設定する
- 上記は推奨であり、実装は上記に制約されない。
- 汎用ペイロード以外のトランザクション・タイプに対しても、上記と同様の応答を推奨するが、応答詳細はこの標準の範囲外である。
- 汎用ペイロードのアトリビュートは従来のメモリ・マップド・バスを想定してセマンティックスが定義されているが、ターゲットが RAM のように動作することを必ずしも想定しているわけではない。次のようなコーナケースが考えられる。
  - i) ターゲットがメモリマップドレジスタを持っていて、write/read 両方をサポートしているが、write の後の read が書き込まれた値ではなく、ターゲットのステータスによるある値を返すことが正常な動作であれば、a)でカバーされる
  - ii) ターゲットの write コマンドがデータアトリビュートを無視してビットを設定するように実装されている場合は、a)でカバーされる
  - iii) ROM への write コマンドはイニシエータにエラーを出すことなく無視されることがあるが、ターゲットは、少なくとも SC_INFO か SC_WARNING のレポートを生成すべきである
  - iv) ターゲットは read コマンドを実行する write コマンドや write コマンドを実行する read コマンドを実装してはならない
  - v) ターゲットは、汎用ペイロードの意図に対応しているが付加機能がある read コマンドを実装するかもしれないが、これは a)でカバーされる
  - vi) アドレス指定可能なレジスタファイルを構成するメモリマップドレジスタを持っているターゲットが、アドレス範囲外を指定された場合には、トランザクションのステータス・アトリビュートに TLM_ADDRESS_ERROR_RESPONSE を設定するか、SystemC レポートを生成すべきである
  - vii) シミュレーションのバスモニター・ターゲットが、バスの物理範囲を越えるアドレスのトランザクションを受け取った時には、a)の対応として、エラーの可能性があるトランザクションを後処理のためにログ記録して、b)や c)によるエラー生成はしないのがよい。代替として、c)に基づくレポートを生成しても良い。
- a),b),c)の選択は、ケースバイケースであるが、汎用ペイロードに対しては、いずれかの対応をすることが明確なルールとなっている。

9.19. 拡張メカニズム

9.19.1. はじめに

拡張メカニズムは一般ペイロードの不可欠な部分であり、一般ペイロードから切り離しては使用できない。これは、一般ペイロードにアトリビュートを追加することを目的とする。

拡張は無視可能なもの、又は必須なものがある。無視可能な拡張は、インターコネクト・コンポーネントの全てか、どれか、または一般ペイロード・トランザクションを受取るターゲットによって無視される。無視可能な拡張の主旨は、補助情報、シミュレーション工芸品、サイドバンド情報、またはメタデータをモデル化することであり、下流コンポーネントの機能へは直接影響しない。必須な拡張は、インターコネクト・コンポーネントまたは、トランザクションを受取るターゲットの何れかが検査し作用する義務を有する拡張である。必須な拡張の主旨は、一般ペイロードをプロトコルの個別詳細をモデル化するため特定化する時に使用することにある。

9.19.2. 理論的根拠

拡張メカニズムの背後にある理論的根拠は、同じトランザクション型と共に特定化する一般ペイロードのコア・アトリビュート集合のバリエーションを与えるＴＬＭポートまたは、ソケットを許可することにある。このようにアダプタまたはブリッジ無しで直接接続することができる。拡張メカニズム無しに、一般ペイロードへの新しいアトリビュート追加は、コア・インタフェース・クラスの新しいテンプレート特殊化に先立ち、新しいプロトコル・クラスの定義が要求される。これは、一般ペイロードと他の如何なる特殊化とはタイプ互換性はない。拡張メカニズムは、TLM ポートのタイプ互換性を破ること無しに一般ペイロードへ導入されるという小規模のバリエーションを認める。これは、ほんの僅か異なる情報を伝えるポートへの接続に必要はコーディング作業を軽減する。

9.19.3. 拡張、安全資料の規則とブリッジの管理

拡張は、クラス tlm_extension から派生するタイプのオブジェクトである。一般ペイロードは拡張オブジェクトへのポインタ配列を含む。拡張オブジェクトはそれ自身、普通トランザクションのイニシエータによって管理される。それは、拡張を創造し、トランザクションへの拡張を追加し、トランザクションが終了するとき拡張を消去するか保持することに責任を持つ。一般ペイロード・オブジェクトは全て、拡張の型を伝えることが出来る。

拡張に対するポインタ配列は登録した拡張に対するスロットを持つ。メソッド set_extension は簡単にポインタを上書きできる。そして、原則としてイニシエータ、インターコネクト・コンポーネント、又はターゲットから呼び出すことができる。これは非常に汎用性の高い低レベルのメカニズムであるが、逆に使用ミスを招く。そこで安全使用のガイドラインを下に示す。ユーザは十分注意を払い拡張オブジェクトの管理と消去を検討すべきである。

a) 一般原則は、一般ペイロードで拡張オブジェクトを作り、ポインタをセットするどんなプロセスもポインタをクリアし、拡張オブジェクトを消去すべきである。

b) 推奨アプローチは、イニシエータを、しかも唯一のイニシエータを持ち、set_extension を呼び出すことである。もしインターコネクト・コンポーネント、またはターゲットが拡張を使用するイニシエータに対し値を戻すならば、イニシエータは拡張を作り追加すべきである。さらに、安全第一のためには、イニシエータは拡張ポインタが未だトランザクション完了時に有効であると仮定すべきでない。そこでトランザクションを保持するとき、イニシエータは各トランスポート呼び出し前に set_extension を呼び出すべきである。

c) もし書き手がイニシエータのソースコードにアクセスできないなら、インターコネクト・コンポーネントは、set_extension の呼び出しは正当である。この場合、インターコネクト・コンポーネントは如何なる存在する拡張ポインタを上書きすべきではないし、トランザクション完了前に拡張を消去すべきでない。

d) もしインターコネクト・コンポーネントが、ターゲットへ順方向トランザクションを引き渡したとき、実際存在する拡張ポインタを上書きするなら、(推奨の手本に反するが)、同じインターコネクト・コンポーネントはイニシエータへ逆方向トランザクションを引き渡す前に元の拡張ポインタを回復すべきである。

e) 二つの独立した一般ペイロード・トランザクション間のブリッジを生成するとき、必要に応じて入って来るトランザクション・オブジェクトから出て行くトランザクション・オブジェクトに如何なる拡張をコピーするための責任、又出て行くトランザクションとその拡張を所有し管理するための責任を持つ。(例としては、データアレイとバイト・イネイブルアレイ) ブリッジは拡張オブジェクトの深いコピー、または拡張アレイの浅いコピーを成し遂げる。後者の場合、拡張オブジェクトがまだ入って来るトランザクションのイニシエータにより所有される。もしブリッジが出て行くトランザクションに更なる拡張を追加するために set_extension を呼び出すなら、これらの拡張はブリッジにより所有される。

f) ９．５章の一般ペイロードメモリ管理を参照。

9.19.4. ルール

a) 拡張はイニシエータ、インターコネクトまたはターゲット・コンポーネントにより追加することができる。特に、拡張の生成はイニシエータに対する制約はない。

b) 拡張はいくつでも一般ペイロードのインスタンスに追加される。

c) 無視可能な拡張に関して、どんなインターコネクトないしターゲット・コンポーネントは与えられた拡張を自由に無視できるが、実装により強制されるべきではないことを推奨する。拡張されていないのでインターコネクト、またはターゲット・コンポーネントに標準エラーを出力することは可能ですが、推奨できない。

d) 無視可能な拡張に関して、与えられた拡張があってもなくてもコンポーネントの主要機能に影響を与えるべきでない。しかし、例えば、診断レポート、デバグないし最適化に影響を与えるかもしれない。

e) 与えられた拡張を強制するビルトイン・メカニズムはない。
f) 拡張の意味付けはアプリケーション定義であり、予め定義した拡張はない。
g) 拡張は、クラス tlm_extension からユーザ定義したクラスを引用すること、並びに tlm_extension に対するテンプレート引数としてユーザ定義したクラスの名前を引き渡すこと、さらにそのクラスオブジェクト生成により生成される。ユーザ定義拡張クラスは、一般ペイロードの拡張アトリビュートを表すメンバーを含む。
h) クラス tlm_extension の純粋仮想機能 clone は、如何なる拡張アトリビュートを含み、拡張オブジェクトのクローンを生成するユーザ定義拡張クラスで定義される。この clone 手法は、一般ペイロードの deep_copy 手法で使用されることを意図している。　コピーは見えない副作用なしに元のオブジェクトの破壊を延命できるように拡張オブジェクトの完全な深いコピーを生成する。
i) クラス・テンプレート tlm_extension をインスタンシエーションすることは、初期化された公的データ番号 ID を生成する。そしてこれは、一般ペイロード・オブジェクトを伴う拡張を登録、ならびに拡張にユニークな ID を指定することになる。ID は全体の実行プログラムで唯一である。
j) 一般ペイロードは、あたかも再サイズ変更可能アレイで拡張に対するポインタを格納する。ここで、拡張の ID はアレイにおける拡張ポインタのインデックスを与える。一般ペイロードを伴う拡張を登録することは、その拡張に対するアレイ・インデックスを予約する。各一般ペイロード・オブジェクトは現在実行中のプログラムにおいて登録された全ての拡張に対して格納ポインタ可能なアレイを含む。
k) 拡張アレイにポインタはトランザクションが構築されているとき null である。
l) 一般ペイロード・オブジェクトは与えられた拡張型の少なくとも一つのオブジェクトを示すポインタを格納することができる。
m) メソッド set_extension、get_extension と clear_extension は夫々、二つの定義を持つ。一つは関数テンプレートであり、または ID 引数を伴うものである。関数テンプレートはテンプレート引数を使用して、どの拡張が Set、Get または Clear するかを決定する。テンプレートでない関数は ID 引数を使用して、どの拡張が Set、Get または Clear するかを決定する。
n) これらの３つのメソッドの関数テンプレートフォームは使用すべきである。ID 引数を持つ関数は一般ペイロード・オブジェトのクローン生成時のように、低レベルプログラミングを意図している。
o) メンバー関数 template<typename T>T* set_extension(T*)は、引数値を伴うポインタ・アレイで型 T の拡張オブジェクトに対してポインタを置換する。引数は登録された拡張に対するポインタである。関数の戻り値はこの呼び出しで置換した一般ペイロードにおけるポインタの前回値（多分 null）である。
p) メンバー関数 tlm_extension_base* set_extension(unsighned int, tlm_extension_base*)は 2 番目の引数値を持つ初めの引数によって与えられるアレイ・インデックスにおけるポインタ・アレイでの型 T の拡張オブジェクトへのポインタを置換する。与えられたインデックスは拡張 ID として登録される。さもなければ、関数の動作が未定義となる。関数の戻り値は与えられたアレイ・インデックスにおけるポインタの前回値（多分 null）である。
q) もし一般ペイロード･オブジェクトが既に設定された型の拡張に対する null でないをポインタを含んでいれば、古いポインタは上書きされる。
r) メンバー関数 template<typename T> void get_extension(T*&)は、与えられた型の拡張オブジェクトへのポインタを戻す。もし存在しなければ、null ポインタを戻す。引数は、tlm_exstension から引用される型のオブジェクトに対するポインタである。この関数テンプレートを使用して存在しない拡張を回復すれことはエラーではない。
s) メンバー関数 tlm_extension_base* get_extension(unsigned int)は、引数から与えられる ID を伴う拡張オブジェクトへのポインタを戻す。与えられたインデックスは拡張 ID として登録されている。さもなければ、関数の動作は未定義となる。もし与えられたインデックスにおけるポインタが拡張オブジェクトに対するポインタではないなら、関数は null ポインタを戻す。
t) メンバー関数 template<typename T> void clear_extension(const T*)は、一般ペイロード・オブジェクトから引数によって与えられた拡張を取り除く。引数は、tlm_extension から引用される型のオブジェクトへのポインタである。
u) メンバー関数 void clear_extension(unsigned int)は、引数から与えられるアレイ索引における拡張での一般ペイロード・オブジェクトから取り除く。与えられたインデックスは拡張 ID として登録されている。さもなけらべ、関数の動作は未定義となる。
v) 一般ペイロード・トランザクションは全ての登録された拡張に対するポインタを格納するための十分なスペースを確保する。これは二つの方法の何れかで実現できる。すなわち、C++静的初期化後のトランザクション・オブジェクトを構築すること、または、静的初期化後で、初めてトランザクション・オブジェクトを使用する前にメソッド re_size_extensions を呼び出すことである。前者では、拡張アレイのサイズを設定するための一般ペイロード・コンストラクターの責任を負う。後者では、初めて拡張をアクセスする前に resize_extensions を呼び出す手続きに責任を負う。
w) メソッド resize_extensions は、全ての登録された拡張に便宜をはかるために一般ペイロードにおける拡張アレイのサイズを増加させる。

10. TLM1 Legacy

以下の TLM1 のコア･インタフェース及び tlm_fifo チャネルは依然として TLM 2.0 ドラフト 2 標準であるが、このドキュメントでは詳しく触れない。

10.1. TLM1 コア・インタフェース

シグネチャ付きのトランスポートメソッドである transport(const REQ& , RSP&) は TLM 1.0 に含まれていなかったが、TLM 2.0 において付け加えられた。

10.2. TLM1 fifo インタフェース

（省略）

10.3. tlm_fifo

（省略）

11. Glossary

（省略）

4.2.5 SystemC推奨設計メソドロジ 合成編

JEITA

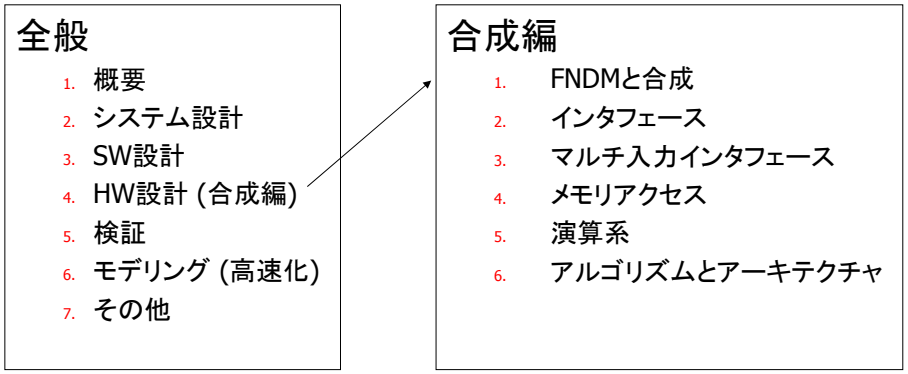
SystemC推奨設計メソドロジの目的と計画

- 目的
  - SystemCの特徴を生かした設計メソドロジ（スタイル）の共通基盤を作る。
  - 設計メソドロジの議論の土台を提案する。
  - 設計メソドロジ案を公開し、各種ツール開発のリファレンスとなることで、ツール間の親和性を高める。
- 計画
  - 2008年3月　合成編完成
  - 2008年夏　　合成編公開（公開方法は検討中）
  - 2008年度　　設計メソドロジ全般を検討予定

 JEITA 2

SystemC推奨設計メソドロジの範囲

- 主にデジタル信号処理系を対象とする。
- アルゴリズムをHWとSWに分割し、コンパイル可能なレベルに実装するまでの手法を検討する。



JEITA

概要

SystemCは、LSIのシステムレベルの設計に使われます。LSIシステム全体を高速にシミュレーションし、システムアーキテクチャ(CPUやメモリ、周辺ハードウエア、CPUバスなどの配置や接続)の最適解を探索します。または、周辺ハードウエアの詳細アーキテクチャを決め、ハードウエアに実装する(合成する)ために使われています。

2005年にSystemCはIEEE-1666として標準化されました。2006年にTLM-1.0がOSCIより公開され、また合成サブセットがレビューのため公開されています。2008年にはTLM-2.0が正式リリースされる予定です。EDAツールもこれらの標準に対応していくと思われます。

ハードウエアの設計においても、これら標準化の流れに従って方法論を構築できれば、システムLSIの設計全体の流れをスムースにします。しかしながら、これら標準においてTLMと合成の橋渡しに関する議論がまだ十分行われていません。OSCIの合成サブセットにおいてSystemCの特徴を生かす合成方法について十分に述べられていません。

ここでは、システムLSI設計の方法論について、主に合成の立場から1つの筋の通る骨組みを提案します。

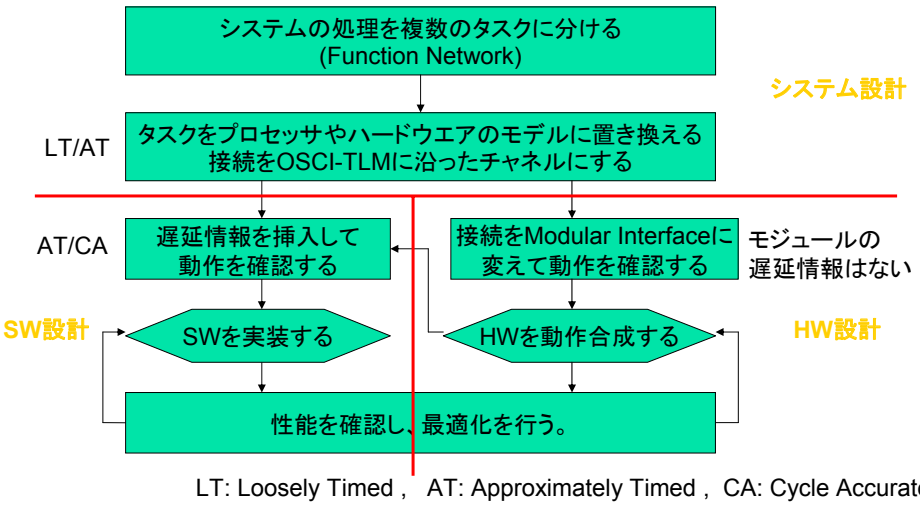
JEITA

FNDMと合成

FNDM：Function Network Design Methodology

JEITA

システムLSIの開発ステップ

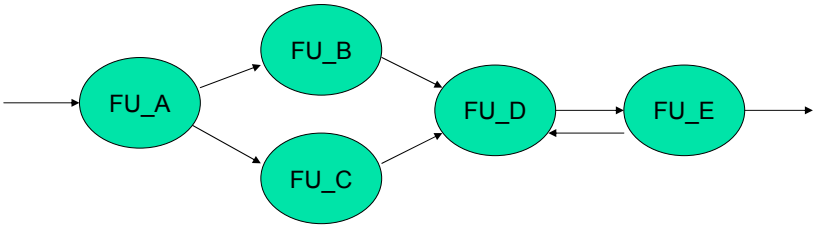


LT: Loosely Timed， AT: Approximately Timed， CA: Cycle Accurate

JEITA
6

Function Network

- 機能ユニット(Function Unit)がデータの流れに応じて接続(Network)される。

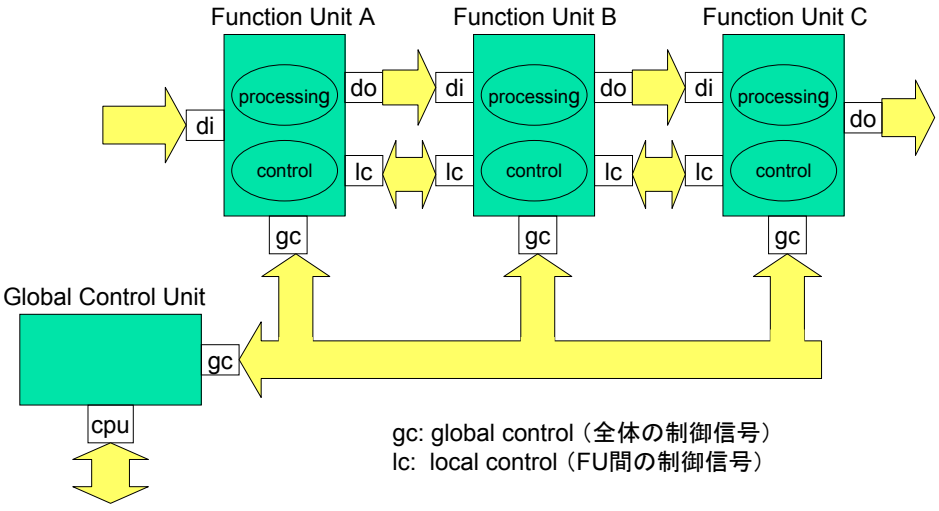


図からわかること

- FU_BとFU_Cは並列に動作可能。
- FU_A->FU_B/FU_C->FU_Dの処理はパイプライン化することもシーケンシャル処理することもできる。
- FU_D->FU_Eの処理はフィードバックがあるためにシーケンシャルにしか処理できない。

JEITA

Function Network を実装するシステムの構成



JEITA

信号の分類

- データ転送
  片方向のハンドシェークを使ったデータ転送が多い。
  モジュラーインタフェースを使う。
- ローカル制御
  データ転送の相手との制御情報の交換する（データのハンドシェーク信号はデータ転送に含まれる）
- グローバル制御
  GCU (Global Control Unit)と制御信号を交換する。
  パラメータのような静的な制御信号が多い。

Function Unit

- データの処理を行うユニット
- データの演算、加工はprocessingプロセスで行う。
- データに依存して処理が変わる場合、processingプロセスから取り出されたその制御信号はcontrolプロセスで扱う。
- プロセスは、processingプロセスかcontrolプロセスに分類できる。その数に制約はない。
- データの入出力ポート数に制約はない。
- gcポートは1つでなければならない。

Global Control Unit

- FU (Function Unit)の制御を行う。
- 通常はシステムに1つ。
- 将棋倒し型の場合は、CPUインタフェースと制御データの保持を行う。
- 中央制御型の場合は、将棋倒し型の機能に加えて各FUの動作タイミングを制御する。

FUの動作タイミング

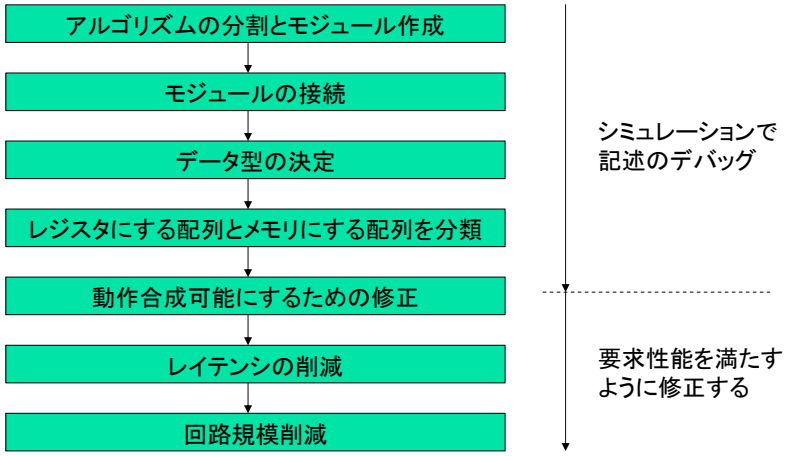
- 将棋倒し型

  データの入力側のモジュールから順に入力データを受け取り準備が整ったところから処理を始める。

  出力が滞り、内部バッファに余裕がなくなった場合は、データ入力を停止する。
- 中央制御型

  GCUの合図に基づいて各モジュールは動作を行う。

HW設計の流れ



JEITA

動作記述の特徴を生かす

下記の4つを独立して設計を行う。

ライブラリ

- インタフェース （モジュラーインタフェース）
- メモリアクセス （モジュラーインタフェース）
- 演算系 （ユーザ定義型）

システム

- アルゴリズムとアーキテクチャ
  （モジュールと階層構造）

JEITA

基本ルール

- モジュールの役割をFU(Function Unit)かGCU(Global Control Unit)のどちらかに定める。
- モジュールはそれぞれの用意された雛形を拡張して記述する。
- インタフェース、メモリアクセス、演算系はライブラリを利用する。
- ライブラリを拡張したり、新規に作成する場合は、メンバ関数名、引数の統一化を行う。

インタフェース

出力インタフェースの公開メンバ関数

- ブロッキング関数（データの送受信が終了するまで戻らない）
  void put(const TD &data, const bool &vld=true)
- ブロッキング関数（データの送受信を1回だけ試み成否を返す）
  bool ns_put(const TD &data, const bool &vld=true)
- ノンブロッキング関数（毎サイクル実行される必要がある）
  前準備
  bool nb_can_put(const bool &vld=true)
  実行
  bool nb_put(const TD &data)
  後処理
  bool nb_clr_put()
- ポート接続
  void bind(T channel)
  void operator ()(T channel)
- インタフェースポートの初期化
  void reset()

シングル入力インタフェースの公開メンバ関数

- ブロッキング関数（データの受信が終了するまで戻らない）
  void get(TD &data, const bool &en=true)
  T get(const bool &en=true)
- ブロッキング関数（データの受信を1回だけ試み成否を返す）
  bool ns_get(TD &data, const bool &en=true)
- ノンブロッキング関数（毎サイクル実行される必要がある）
  前準備
  bool nb_can_get(const bool &en=true)
  実行
  bool nb_get(TD &data)
  後処理
  bool nb_clr_get()
- ポート接続
  void bind(T channel)
  void operator ()(T channel)
- インタフェースポートの初期化
  void reset()

インタフェース・チャネルの公開メンバ関数とポート

- チャネルのトレース
  friend sc_trace(sc_trace_file*, const T&, const std::string&)
- FIFOなどの内部にレジスタを持つチャネルでは、クロックやリセットが必要になるが、その場合は、以下のポート名を使うものとする。

  clk　クロック
  rst　リセット（負論理）

  ※解析用に信号が必要な場合については検討中である。

put関数

void put(const TD &data, const bool &vld=true)

vldが真のときにdataを送信する。送信が終了するまで次の処理に進まない。

vldが偽のときは、送信を行わない。vldが真のときに必要な最小クロック数だけ待って次の処理に進む。

ns_put関数

bool ns_put(const TD &data, const bool &vld=true)

- vldが真のときにdataの送信を1回試みる。成功した場合はtrueが、失敗した場合はfalseが戻る。
- vldが偽のときは、送信を行わない。vldが真のときに必要な最小クロック数だけ待って次の処理に進む。

nb_can_put関数

bool nb_can_put(const bool &vld=true)

- ノンブロッキング関数であり、クロックを含まずに処理する。
- vldが偽の場合は常に偽が戻される。
- vldが真でかつ送信がすぐ可能な場合に真が戻り、それ以外の場合は偽が戻る。

nb_put関数

bool nb_put(const TD &data)

- ノンブロッキング関数であり、クロックを含まずに処理する。
- データ送信が行われる場合に真が戻り、行われないと判断できる場合に偽が戻る。

nb_clr_put関数

bool nb_clr_put()

- ノンブロッキング関数であり、クロックを含まずに処理する。
- nb_put()を実行した次のクロックサイクルでデータ送信を終了する場合に必ず実行する。nb_can_put(false)で代用させる事ができる。

put()のnon-blocking関数による定義

non-blocking関数は、以下の使用方法でblocking関数が作れるように実装すること。

```
void put(const TD &data, const bool &vld=true){
  if(vld){
    while(!nb_can_put()) wait();
    while(!nb_put(data)) wait();
    do{wait();}
    while(nb_clr_put());
  }else
    wait();
}
```


JEITA

get関数

void get(TD &data, const bool &en=true)

T get(const bool &en=true)

- enが真のときにデータを受信する。受信が終了するまで次の処理に進まない。
- enが偽のときは、受信を行わない。enが真のときに必要な最小クロック数だけ待って次の処理に進む。
- get()関数には2通りの形態があり、dataを引数として渡してそこに受信値を受け取るか、戻り値として受信値として受け取る。

JEITA

ns_get関数

bool ns_get(TD &data, const bool &en=true)

- enが真のときにdataの受信を1回試みる。成功した場合はtrueが、失敗した場合はfalseが戻る。
- enが偽のときは、受信を行わない。enが真のときに必要な最小クロック数だけ待って次の処理に進む。

nb_can_get関数

bool nb_can_get(const bool &en=true)

- ノンブロッキング関数であり、クロックを含まずに処理する。
- enが偽の場合は常に偽が戻される。
- enが真でかつ受信の予約ができた場合に真が戻り、それ以外の場合は偽が戻る。

nb_get関数

bool nb_get(TD &data)

- ノンブロッキング関数であり、クロックを含まずに処理する。
- データ受信が行われる場合に真が戻り、行われない場合に偽が戻る。nb_can_get()が真になったあと一定数のクロック後に1クロックだけ真になる。このタイミングを逃すとnb_get()で真の戻り値を得ることができない。

nb_clr_get関数

bool nb_clr_get()

- ノンブロッキング関数であり、クロックを含まずに処理する。
- nb_can_get()を実行した次のクロックサイクルで次のデータ受信を続けない場合に必ず実行する。nb_can_get(false)で代用させる事ができる。

get()のnon-blocking関数による定義

non-blocking関数は、以下の使用方法でblocking関数が作れるように実装すること。

```
void get(TD &data, const bool &en=true){
  if(en){
    while(!nb_can_get()) wait();
    do{wait();}
    while(!nb_get(data));
    while(!nb_clr_get()) wait();
  }else
    wait();
}
```


bind関数

void bind(T channel)

void operator ()(T channel)

モジュールのポートとそのモジュールの外部のチャネルとを接続する。そのモジュールの上位モジュールのポートとの接続も可能。

reset関数

void reset()

インタフェースポートに関連したレジスタを初期化する。

sc_trace関数

friend sc_trace(sc_trace_file*, const T&, const std::string&)

vcdファイルにトレースを記録する場合に使用する。

blocking関数の使い方

blocking関数の記述は簡単。

```
while(1){
  input.get(d_in);
  d_out = calc(d_in);
  output.put(d_dout);
}
```


Non-blocking関数の使い方

nb_clr_get()とnb_clr_putをnb_can_get()とnb_can_put()に続くwait()の直後におく。条件判定に直接記述するとどちらか一方のみしか起動できない場合があるので、check_stall()関数を用意する。

```
bool en、vld, is_ds;  T data;
en=true;  vld=false;  is_ds=false;
while(1){
  input.nb_can_get(en);
  if(is_ds)
    is_ds = false;
  else
    vld = input.nb_get(data);

  if(vld)  d_out = calc(d_in);

  if(vld)
    if(en=output.nb_can_put())  output.nb_put(d_dout);

  do{wait();}
  while(check_stall(data, is_ds));
}
```

check_stall関数

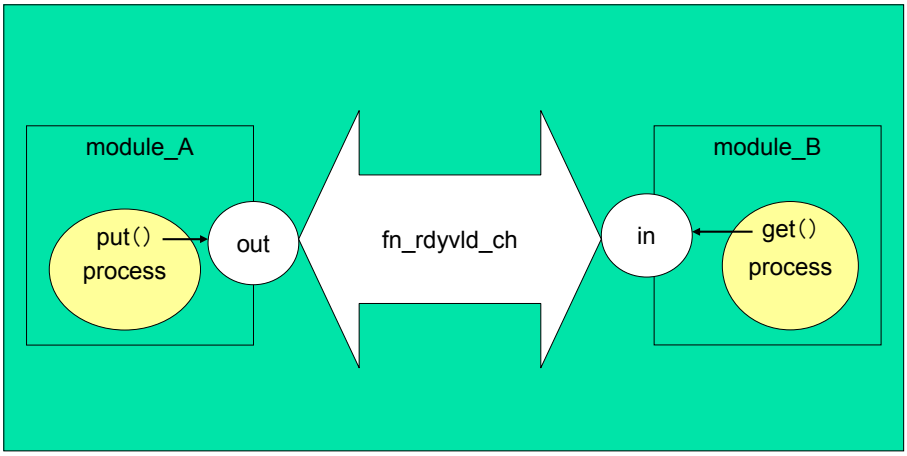
- ノンブロッキング関数を利用するとき、複数のclr関数がある場合に用意する。
- 出力の都合でストールする場合、入力データを取りこぼす可能性があるので対処をしておく。

```
bool check_stall(T &ds, bool &is_ds ){
    bool stall = !input.nb_clr_get();
    stall = stall || !output.nb_clr_put();
    if(stall&&!is_ds)
      is_ds = input.nb_get(ds);
    return stall;
  }
```


インタフェース・ライブラリ

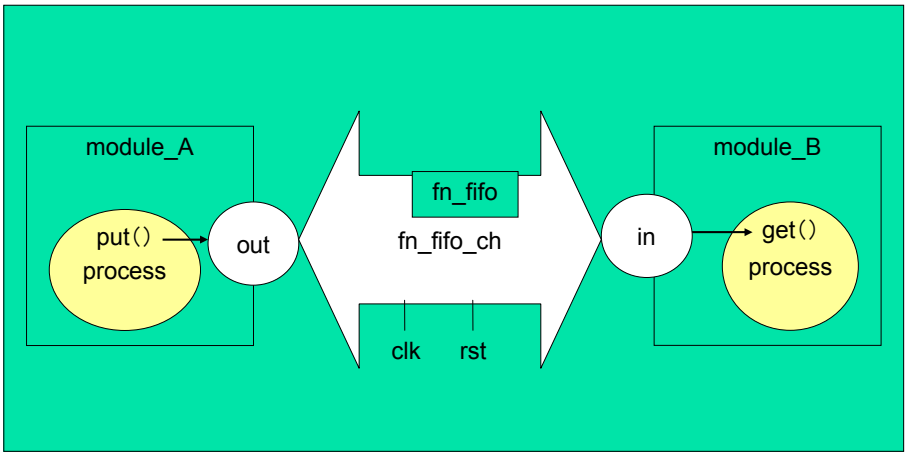
- モジュラーインタフェースで用意される。
- mi::mi_get_if<TD>クラスかmi::mi_put_if<TD>クラスを継承している。
- ハンドシェークのみでバッファされないデータ転送
  fn_rdyvld_ch<TD>　チャネル
  fn_rdyvld_ch<TD>::in　データ入力ポート
  fn_rduvld_ch<TD>::out　データ出力ポート
- レジスタで構成されたFIFOを持つデータ転送
  fn_fifo_ch<TD,FL>　チャネル
  fn_fifo_ch<TD,FL>::in　データ入力ポート
  fn_fifo_ch<TD,FL>::out　データ出力ポート
- SRAMで構成されたFIFOを持つデータ転送
  fn_fifo_sram_ch<TMW>　チャネル
  fn_fifo_sram_ch<TMW>::in　データ入力ポート
  fn_fifo_sram_ch<TMW>::out　データ出力ポート
- FIFOのチャネルにはclk（立ち上がり動作）とrst(負論理）のポートがあり、クロックとリセットの供給が必要である。

ハンドシェーク (fn_rdyvld_ch)



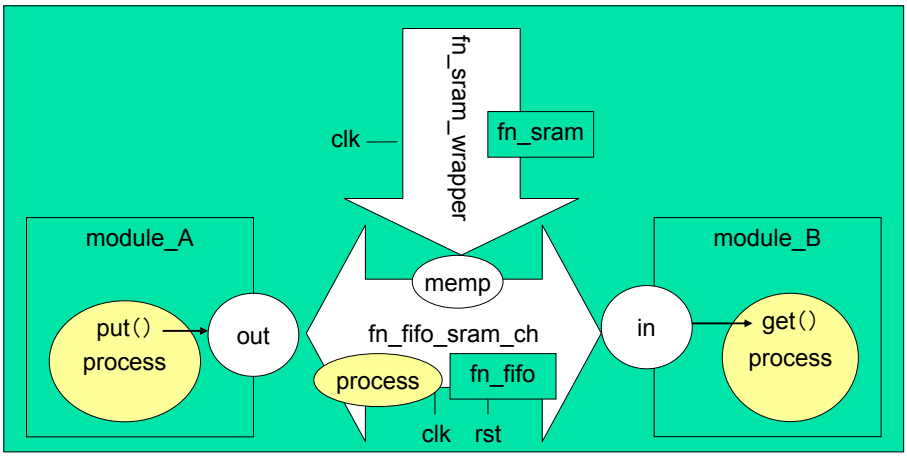
JEITA

FIFO (fn_fifo_ch)



JEITA

SRAMを用いたFIFO (fn_fifo_sram_ch)



JEITA

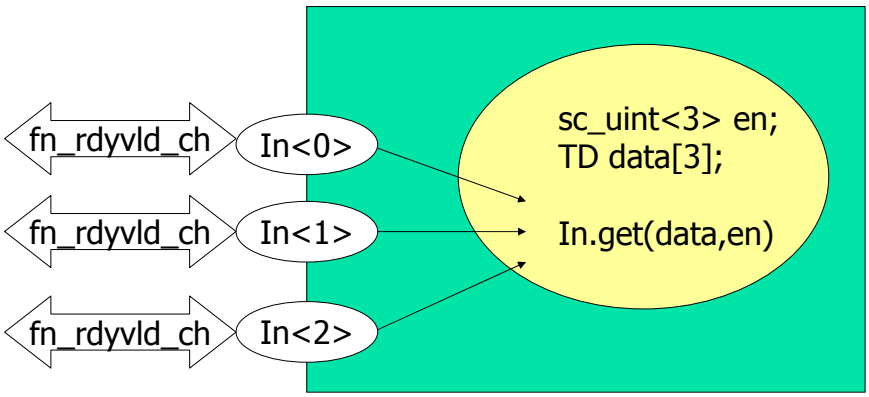
マルチ入力インタフェース

JEITA

マルチ入力インタフェースの概要

- 1つのモジュールに同一種類の入力インタフェースが複数あるときに使用することができる。
- 複数の入力を一度にアクセスするためのメンバ関数が用意されている。
- 同一種類の入力インタフェースが複数ある場合でも、独立して入力する場合は、シングル入力インタフェースを用いた方が良い。
- ポートは入力の数だけ存在し、シングル入力インタフェースと同じチャネルを接続する。

マルチ入力インタフェースの例



マルチ入力インタフェースの公開メンバ関数

- ブロッキング関数（データの受信が終了するまで戻らない）
  void get(TD data[num_of_port], const sc_uint<num_of_port> &en)
- ブロッキング関数（データの受信を1回だけ試み成否を返す）
  sc_uint<num_of_port> ns_get(TD data[num_of_port], const sc_uint<num_of_port> &en)
- ノンブロッキング関数（毎サイクル実行される必要がある）
  前準備
  sc_uint<num_of_port> nb_can_get(const sc_uint<num_of_port> &en)
  実行
  sc_uint<num_of_port> nb_get(TD &data)
  後処理
  sc_uint<num_of_port> nb_clr_get()
- ポート接続
  void bind<num>(T channel)
- 全インタフェースポートの初期化
  void reset()

get関数

void get(TD data[num_of_port], const sc_uint<num_of_port> &en)

enのビットが1のポートのデータを受信する。指定した全受信が終了するまで次の処理に進まない。

enが偽のときは、受信を行わない。enが真のときに必要な最小クロック数だけ待って次の処理に進む。

num_of_portは、ポートの数。

ns_get関数

sc_uint<num_of_port> ns_get(TD data[num_of_port], const sc_uint<num_of_port> &en)

- enと戻り値は各ビットが各ポートの動作を示す。
- enのビットが1のポートのdataの受信を1回試みる。成功したポートのビットは1が、失敗したポートのビットは0が戻る。
- enのビットが0のポートは、受信を行わない。enのビットが1のときに必要な最小クロック数だけ待って次の処理に進む。

nb_can_get関数

sc_uint<num_of_port> nb_can_get(const sc_uint<num_of_port> &en)

- ノンブロッキング関数であり、クロックを含まずに処理する。
- enが偽の場合は常に偽が戻される。
- enが真でかつ受信の予約ができた場合に真が戻り、それ以外の場合は偽が戻る。

nb_get関数

sc_uint<num_of_port> nb_get(TD &data)

- ノンブロッキング関数であり、クロックを含まずに処理する。
- データ受信が行われる場合に真が戻り、行われない場合に偽が戻る。nb_can_get()が真になったあと一定数のクロック後に1クロックだけ真になる。このタイミングを逃すとnb_get()で真の戻り値を得ることができない。

nb_clr_get関数

sc_uint<num_of_port> nb_clr_get()

- ノンブロッキング関数であり、クロックを含まずに処理する。
- nb_can_get()を実行した次のクロックサイクルで次のデータ受信を続けない場合に必ず実行する。nb_can_get(false)で代用させる事ができる。

get()のnon-blocking関数による定義

non-blocking関数は、以下の使用方法でblocking関数が作れるように実装すること。

```
void get(TD &data, const bool &en=true){
  if(en){
    while(!nb_can_get()) wait();
    do{wait();}
    while(!nb_get(data));
    while(!nb_clr_get()) wait();
  }else
    wait();
}
```


bind関数

template<unsigned int num>

void bind(T channel)

モジュールのポートとそのモジュールの外部のチャネルとを接続する。そのモジュールの上位モジュールのポートとの接続も可能。numはポート番号。

reset関数

void reset()

インタフェースポートに関連したレジスタを初期化する。

メモリアクセス

メモリ書き込みの公開メンバ関数

- ブロッキング関数（メモリの書き込みが終了するまで戻らない）
  void put(const TA &address, const TD &data, const bool &vld=true)
- ブロッキング関数（メモリの書き込みを1回だけ試み成否を返す）
  bool ns_put(const TD &data, const bool &vld=true)
- ノンブロッキング関数（毎サイクル実行される必要がある）
  前準備
  bool nb_can_put(const TA &address, const bool &vld=true)
  実行
  bool nb_put(const TD &data)
  後処理
  bool nb_clr_put()
- ポート接続
  void bind(T channel)
  void operator ()(T channel)
- インタフェースポートの初期化
  void reset()

メモリ読み出しの公開メンバ関数

- ブロッキング関数（メモリの読み出しが終了するまで戻らない）
  void get(const TA &address, TD &data, const bool &en=true)
  T get(const TA &address, const bool &en=true)
- ブロッキング関数（メモリの読み出しを1回だけ試み成否を返す）
  bool ns_get(TD &data, const bool &en=true)
- ノンブロッキング関数（毎サイクル実行される必要がある）
  前準備
  bool nb_can_get(const TA &address, const bool &en=true)
  実行
  bool nb_get(TD &data)
  後処理
  bool nb_clr_get()
- ポート接続
  void bind(T channel)
  void operator ()(T channel)
- インタフェースポートの初期化
  void reset()

メモリアクセス・チャネルの公開メンバ関数とポート

- チャネルのトレース

  friend sc_trace(sc_trace_file*, const T&, const std::string&)

- ポート

  clk　クロック

  rst　リセット（負論理）

  // 解析用信号（検討中）

put関数

void put(const TA &address, const TD &data, const bool &vld=true)

vldが真のときにdataを書き込む。書き込みが終了するまで次の処理に進まない。

vldが偽のときは、書き込みを行わない。vldが真のときに必要な最小クロック数だけ待って次の処理に進む。

ns_put関数

bool ns_put(const TA &address, const TD &data, const bool &vld=true)

- vldが真のときにdataの書き込みを1回試みる。成功した場合はtrueが、失敗した場合はfalseが戻る。
- vldが偽のときは、書き込みを行わない。vldが真のときに必要な最小クロック数だけ待って次の処理に進む。

nb_can_put関数

bool nb_can_put(const TA &address, const bool &vld=true)

- ノンブロッキング関数であり、クロックを含まずに処理する。
- vldが偽の場合は常に偽が戻される。
- vldが真でかつ書き込みがすぐ可能な場合に真が戻り、それ以外の場合は偽が戻る。

nb_put関数

bool nb_put(const TD &data)

ノンブロッキング関数であり、クロックを含まずに処理する。

データ書き込みが行われる場合に真が戻り、行われないと判断できる場合に偽が戻る。

nb_clr_put関数

bool nb_clr_put()

ノンブロッキング関数であり、クロックを含まずに処理する。

nb_put()を実行した次のクロックサイクルでデータ書き込みを終了する場合に必ず実行する。nb_can_put(false)で代用させる事ができる。

put()のnon-blocking関数による定義

non-blocking関数は、以下の使用方法でblocking関数が作れるように実装すること。

```
void put( const TA &address, const TD &data, const
   bool &vld=true){
  if(vld){
    while(!nb_can_put(address)) wait();
    while(!nb_put(data)) wait();
    do{wait():}
    while(nb_clr_put());
  }else
    wait();
}
```


get関数

void get(const TA &address, TD &data, const bool &en=true)

T get(const TA &address, const bool &en=true)

- enが真のときにデータを読み出す。読み出しが終了するまで次の処理に進まない。
- enが偽のときは、読み出しを行わない。enが真のときに必要な最小クロック数だけ待って次の処理に進む。
- get()関数には2通りの形態があり、dataを引数として渡してそこに受信値を受け取るか、戻り値として受信値として受け取る。

ns_get関数

bool ns_get(const TA &address, TD &data, const bool &en=true)

- enが真のときにdataの読み出しを1回試みる。成功した場合はtrueが、失敗した場合はfalseが戻る。
- enが偽のときは、読み出しを行わない。enが真のときに必要な最小クロック数だけ待って次の処理に進む。

nb_can_get関数

bool nb_can_get(const bool &en=true)

- ノンブロッキング関数であり、クロックを含まずに処理する。
- enが偽の場合は常に偽が戻される。
- enが真でかつデータ読み出しの予約ができた場合に真が戻り、それ以外の場合は偽が戻る。

nb_get関数

bool nb_get(TD &data)

- ノンブロッキング関数であり、クロックを含まずに処理する。
- データ読み出しが行われる場合に真が戻り、行われない場合に偽が戻る。nb_can_get()が真になったあと一定数のクロック後に1クロックだけ真になる。このタイミングを逃すとnb_get()で真の戻り値を得ることができない。

nb_clr_get関数

bool nb_clr_get()

- ノンブロッキング関数であり、クロックを含まずに処理する。
- nb_can_get()を実行した次のクロックサイクルで次のデータ読み出しを続けない場合に必ず実行する。nb_can_get(false)で代用させる事ができる。

get()のnon-blocking関数による定義

non-blocking関数は、以下の使用方法でblocking関数が作れるように実装すること。

```
void get(const TA &address, TD &data, const bool
   &en=true){
  if(en){
    while(!nb_can_get(address)) wait();
   do{wait();}
    while(!nb_clr_get());
    while(!nb_get(data)) wait();
  }else
    wait();
}
```


JEITA

bind関数

void bind(T channel)

void operator ()(T channel)

モジュールのポートとそのモジュールの外部のチャネルとを接続する。そのモジュールの上位モジュールのポートとの接続も可能。

JEITA

reset関数

void reset()

メモリアクセスポートに関連したレジスタを初期化する。

sc_trace関数

friend sc_trace(sc_trace_file*, const T&, const std::string&)

vcdファイルにトレースを記録する場合に使用する。

blocking関数の使い方

blocking関数の記述は簡単。
forループの中で使われる場合が多い

```
sum = 0;
for(address = address_min; address <
   address_max; ++address){
  input.get(address, d_in);
  sum+=din;
}
```


Non-blocking関数の使い方

Non-blocking関数は毎サイクル実行されなければならない。
メモリアクセスのnb_can_getとnb_getは並列に実行する。
forループの中で使われる場合が多い。

```
sum=0;  addr=addr_min;
for(sc_uint<4> i=0; i<8;){
  if(mem.nb_can_get(addr, addr<addr_max)) addr++;
  vld = mem.nb_get(din);
  if(vld){
    ++i;
    sum+= din;
  }
  wait();
}
```

Non-blocking関数の使い方2

nb_can_getの第2引数enableを使用するのではなく、nb_clr_get関数によって処理を継続しないことを示した方がわかりやすいかもしれない。

```
sum=0;  addr=addr_min;
for(sc_uint<4> i=0; i<8;){
  if(addr<addr_max)
    if(mem.nb_can_get(addr)) addr++;
  vld = mem.nb_get(din);
  if(vld){
    ++i;  sum+= din;
  }
  wait();
  mem.nb_clr_get();
}
```


Non-blocking関数の使い方3

メモリアクセスのnb_can_putとnb_putの使い方は、インタフェースの場合に似ている。これもforループの中で使われることが多い。

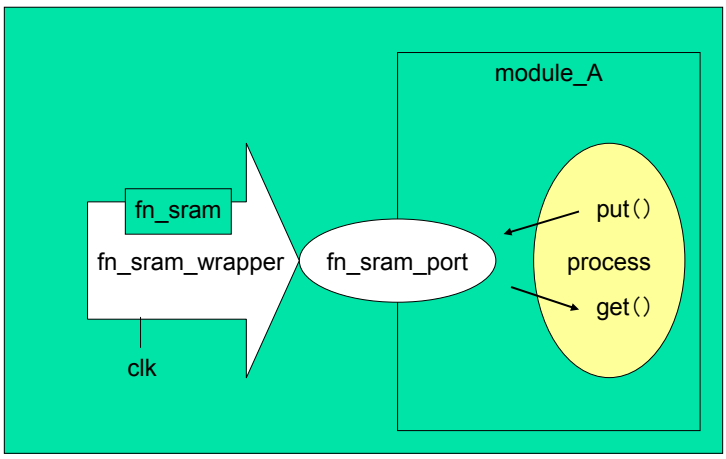
```
en=true;  vld=false;
for(address = address_min; address < address_max; ){
  if(input.nb_can_get(en))  vld = input.nb_get(d_in);
  else  vld = !en;
  if(vld)  d_out = calc(d_in);

  if(mem.nb_can_put(address, vld)){
    en = mem.nb_put(d_dout);
    ++address;
  }else  en = !vld;
  wait();
  mem.nb_clr_put();
}
```

メモリアクセス・ライブラリ

- モジュラーインタフェースで用意される。
- mi::mi_mem_get_if<TA,TD>クラスかmi::mi_mem_put_if<TA,TD>クラスを継承している。
- メモリはモジュールの外部に配置する。
- 1つのプロセスからgetとputをする場合
  fn_sram_port<TMW> SRAMアクセスポート
- getするプロセスとputするプロセスが異なる場合
  fn_sram_ch<TMW>　チャネル
  fn_sram_ch<TMW>::mem　SRAMアクセスポート
- getするモジュールとputするモジュールが異なる場合
  fn_sram_wr_ch<TMW>　チャネル
  fn_sram_wr_ch<TMW>::in　SRAMリードアクセスポート
  fn_sram_wr_ch<TMW>::out　SRAMライトアクセスポート
- getとputが別のチャネルにはclk（立ち上がり動作）とrst(負論理）のポートがあり、クロックとリセットの供給が必要である。

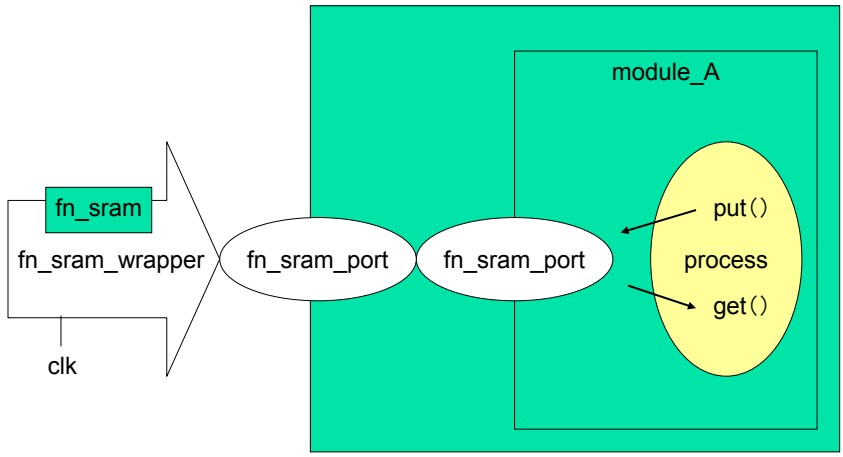
JEITA

メモリアクセス(fn_sram_port)



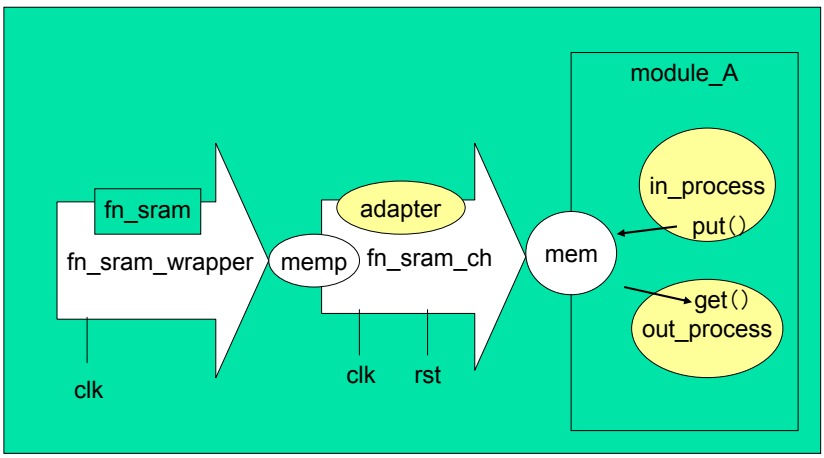
JEITA

メモリアクセス(fn_sram_port)



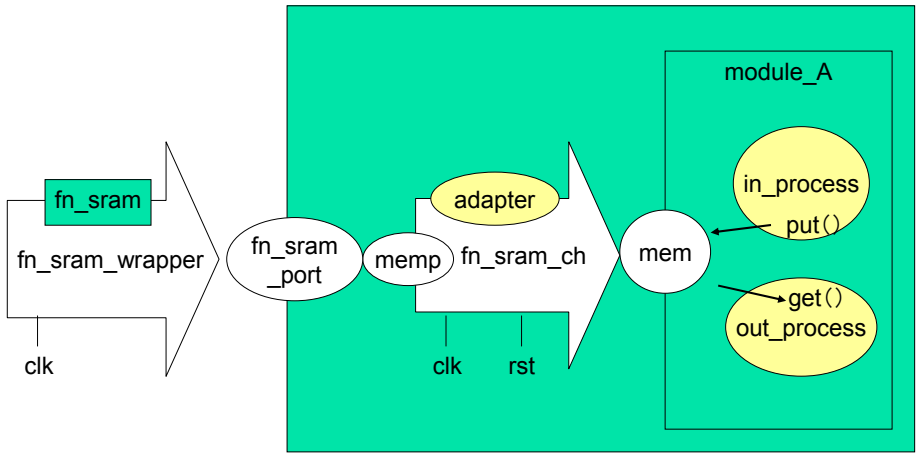
JEITA
79

メモリアクセス(fn_sram_ch)



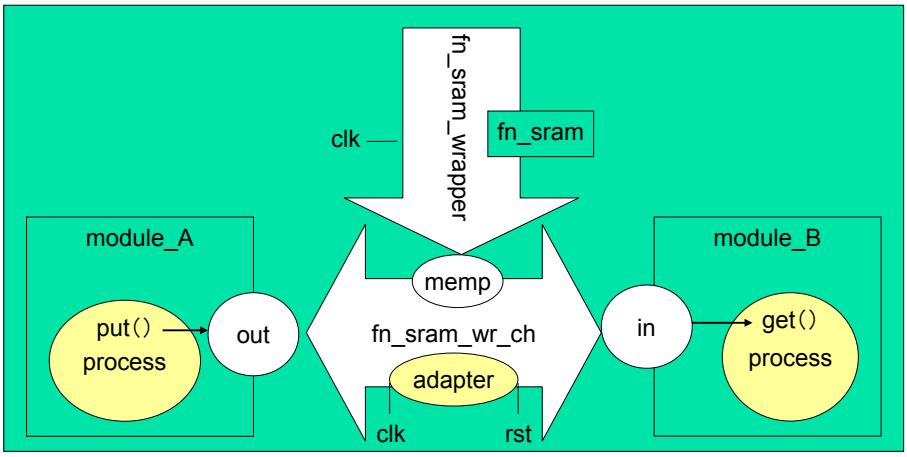
JEITA
80

メモリアクセス(fn_sram_ch)



JEITA

メモリアクセス(fn_sram_wr_ch)



JEITA

演算系

演算系ライブラリ

- 次の定数と演算子と関数は必ず用意する。

static const unsigned int raw_bits_len
T(const sc_uint<raw_bits_len>&)
void set_raw_bits(T&, const sc_uint<raw_bits_len>&)
　ビット列を与えて値を初期化
sc_uint<raw_bits_len> raw_bits(const T&)
　値をビット列として取り出し
// sc_uintを使わずにfn_raw_bits<ram_bits_len>クラスを用意した方が良いが、そうするとクラスの名前まで統一化することになる。
== 等値の評価
<< 外部出力
const std::string to_string()
sc_trace(sc_trace_file*, const T&, const std::string&)

値のビット列による初期化と取り出し

- 演算系の値をsc_uint<raw_bits_len>との間で相互変換可能にする。この値は、演算系がその値を示す場合に最低限必要なビット列である。たとえば、浮動小数点であれば、指数と仮数のデータを並べたものである。並べ方はライブラリ依存とする。
- データ保存などの演算を行わないデータ処理が共有可能になる。
- 合成不可能な型（たとえばdouble型など）の定数は、あらかじめビット列に変換しておいて使用する。

アルゴリズムとアーキテクチャ

モジュールGCUとFUの雛形

- モジュールGCUとFUの接続は、モジュラーインタフェースgcのみとする。
- モジュールFUは、GCUとの接続gcと必要な数だけのデータ入出力とローカルコントロール入出力がある。これらは、すべてモジュラーインタフェースである。基本的にモジュールFUは、これ以外にクロックポートclkとリセットポートrstだけを持つ。

共有ヘッダファイル

```
// common.hh
#ifndef _COMMON_HH_
#define _COMMON_HH_ 1
#if SYSTEMC_VERSION <= 20050714  // 2.1.v1
#include <systemc.h>
#else
#include <systemc>
using namespace sc_core;
using namespace sc_dt;
#endif
#include “fn.h”

typedef sc_int<16> data_t;
typedef fn_rdyvld_ch<data_t> data_ch_t;
typedef data_ch_t::in data_in_if_t;
typedef data_ch_t::out data_out_if_t;
#endif
```

GCUのヘッダファイル

```
// gcu.hh
#ifndef _GCU_HH_
#define _GCU_HH_ 1
class gcu : public sc_module
{
public:
  sc_in<bool> clk;
  sc_in<bool> rst;
  // CPU interface のポートを羅列する。
    ...
  gc_ch::gcu gc;
  void processing1();
  SC_HAS_PROCESS(gcu);
  gcu(const sc_module_name& name=“gcu”);
};
#endif
```


GCUのソースファイル

```
// gcu.cc
#include “common.hh”
#include “gc_ch.hh”
#include “gcu.hh”
void gcu::processing1(){
  if(!rst){
    // 初期化処理
  }else{
    // 動作
  }
}

gcu::gcu(const sc_module_name& name):
 sc_module(name),clk(“clk”),rst(“rst”), ..., gc(“gc”){
  SC_METHOD(processing1); // processing1はRTL
  sensitive << clk.pos() << rst.neg();
}
```

FUのヘッダファイル

```
// fxxx.hh
#ifndef _FXXX_HH_
#define _FXXX_HH_ 1
class fxxx : public sc_module
{
  public:
  sc_in<bool> clk;
  sc_in<bool> rst;
  gc_ch::fu1 gc;
  lc_ch::out lc;
  data_in_if_t din;
  data_out_if_t dout;
  void processing1();
  SC_HAS_PROCESS(fxxx);
  fxxx(const sc_module_name& name=“fxxx”);
};
#endif
```


FUのソースファイル

```
// fxxx.cc
#include “common.hh”
#include “fxxx.hh”
void fxxx::processing1(){
  // 初期化処理
  wait();
  while(true){
    // 動作
  }
}

fxxx::fxxx(const sc_module_name& name) :
 sc_module(name), clk(“clk”), rst(“rst”),
 gc(“gc”), lc(“lc”), din(“din”), dout(“dout”) {
  SC_CTHREAD(processing1, clk.pos()); // processing1はBL
  reset_signal_is(rst, false);
}
```

モジュラインタフェースgcまたはlc

```
// gc_ch.hh
#ifndef _GC_CH_HH_
#define _GC_CH_HH_ 1
#include “common.hh”
class gc_gcu_port;
class gc_ful_port;
class gc_fu2_port;
class gc_ch : public sc_object
{
public:
  sc_signal<ttt1> sss1; // tttnは型名で、sssnは信号名で置き換える。
  sc_signal<ttt2> sss2;
  sc_signal<ttt3> sss3;
  typedef gc_gcu_port gcu;
  typedef gc_ful_port ful;
  typedef gc_fu2_port fu2;
  friend void sc_trace(sc_trace_file* tf, const gc_ch &ch, const std::string& str){
    sc_trace(tf, ch.sss1, str+”.sss1”);
    sc_trace(tf, ch.sss2, str+”.sss2”);
    sc_trace(tf, ch.sss3, str+”.sss3”);
  }
  gc_ch(const char* name=“gc_ch”):sc_object(name),
  sss1(“sss1”),sss2(“sss2”),sss3(“sss3”){};
};
```


モジュラインタフェースgcまたはlc (続き)

```
class gc_gcu_port : public sc_object
{
public:
  sc_out<ttt1> sss1;
  sc_in<ttt2> sss2;
  sc_in<ttt3> sss3;
  void bind(gc_ch &ch){
    sss1(ch.sss1);
    sss2(ch.sss2);
    sss3(ch.sss3);
  }
  void operator ()(original_gc_ch &ch){ bind(ch); }
  gc_gcu_port(const char* name=“gc_gcu_port”) :
   sc_object(name),
    sss1(“sss1”), sss2(“sss2”), sss3(“sss3”){};
};
```

モジュラインタフェースgcまたはlc (続き)

```
class gc_fu1_port : public sc_object
{
public:
  sc_in<ttt1> sss1;
  sc_out<ttt2> sss2;
  void bind(gc_ch &ch){
    sss1(ch.sss1);
    sss2(ch.sss2);
  }
  void operator ()(gc_ch &ch){ bind(ch); }
  gc_fu1_port(const char* name=“gc_fu1_port”) :
    sc_object(name),
    sss1(“sss1”), sss2(“sss2”){};
};
```


モジュラインタフェースgcまたはlc (続き)

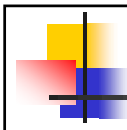
```
class gc_fu2_port : public sc_object
{
public:
  sc_in<ttt1> sss1;
  sc_out<ttt3> sss3;
  void bind(gc_ch &ch){
    sss1(ch.sss1);
    sss3(ch.sss3);
  }
  void operator ()(gc_ch &ch){ bind(ch); }
  gc_fu2_port(const char* name=“gc_fu2_port”) :
    sc_object(name),
    sss1(“sss1”), sss3(“sss3”){};
};
#endif
```

検討課題

- 複数のプロセスやモジュールからSRAMの共通の領域を読み書き可能なアクセス
- 複数のプロセスやモジュールからSRAMの異なる領域を読み書き可能なアクセス
- DRAMのアクセス
- FUとしてのDSPやCPUの組み込み

4.2.6 SystemCユーザフォーラム 2008 開催報告

JEITA



SystemCユーザフォーラム2008概要

- 主催：JEITA EDA技術専門委員会
- 協賛：OSCI
- 日時：2008年1月25日　10:00～12:00
- 会場：パシフィコ横浜アネックスホールF203+F204 (定員200名)
- 講演内容：
  - 司会
    - 長谷川 隆氏(SC-WG主査／富士通)
  - 『SystemC Community Update』
    - Patrick Sheridan氏(OSCI／CoWare)
  - 『OSCI TLM2.0 Draft2 tutorial』
    - Thorsten Grötker氏（OSCI／Synopsys）
  - 『TLM2.0に関する取り組み／SystemC推奨設計メソドロジ』
    - 渡邊　政志氏(ルネサス)／長尾　文昭氏（三洋電機）
  - 『TLMを利用したソフトウェア早期開発』
    - 中村　和正氏(富士通)

 JEITA 2

SystemCユーザフォーラム2008開催方針

- 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ユーザへのOSCI活動状況を報告する事。
  - OSCI TLM2.0のユーザー拡大を目的としたチュートリアルを開催する事。
  - JEITA SystemC WG活動内容を紹介し、活動実績を発表する事。
  - SystemCの適用事例を紹介し、ユーザへ普及の現状をアピールする事。

 JEITA

SystemCユーザフォーラム2008サマリー

- 2時間で4講演を実施。本年は現在OSCIで策定中のOSCI TLM2.0に関する講演を中心とした構成。またSystemCの推奨される設計メソドロジについても触れた。
- 講演内容：
  - OSCIからはPatrick Sheridan氏を向かえ、OSCIに関するアップデートを行った。主にTLMワーキングにおけるTLM2.0 Draft2の最新状況の説明があった。（20分）
  - 今回のフォーラムの目玉であるOSCI TLM2.0 Draft2チュートリアルを、OSCIからThorsten Grötker氏を招いて行った。ユーザーにOSCI TLMの特徴を伝える非常に良い機会となった。（40分）
  - SystemC WGでは、OSCI TLM2.0の要求仕様をレビューした結果の報告を行った。TLMチュートリアルからでは分からないユースケースの分析、いくつかの機能等の背景や注意点をユーザーに伝える事が出来た。また、WGにおけるSystemCを使用する際に推奨される設計メソドロジの構築作業の現状報告を行った。（30分）
  - ユーザ事例は富士通の中村氏より、富士通社内におけるTLMを用いた検証事例について講演頂き、OSCI TLM2.0 Draft2を適用した際の特徴や問題点をユーザーに講演頂いた。ユーザーにとって実際にTLMを使用する際の注意点が浮き彫りになる興味深い内容であった。（30分）

 JEITA

SystemCユーザフォーラム2008サマリー

- 会場状況・予稿集について
  - 本年のフォーラム参加人数は172名となり、昨年の148名と本年目標の150名を上回る事が出来て盛況であった。
  - TLMチュートリアルがやや時間不足で予定より10分程度オーバーしたが、フォーラム全体を通してはほぼ予定通りの時間で終了。今年は質疑応答が少なかった。
  - 昨年好評だったOSCI英文予稿集の日本語訳文を本年も付帯した。発表資料との差異が大きいページも見受けられたが、講演者の方が事前にウェブ上へアップされて居た為、それ程大きな不満にはならなかった。またチュートリアルは時間の関係で若干未使用ページがあった。
- またアンケートからの意見として次のようなものがあった。
  - 富士通中村氏の発表が素晴らしかった。(同意見計2名)
  - OSCI発表は同時通訳か、日本メンバーに日本語で発表して欲しい。
  - 質問に対するOSCIの回答も日本語訳して欲しい。(同意見計3名)
  - 開発者のチュートリアルが受けれて有意義なフォーラムであった。同フォーラムを次年度以降も続けて欲しい。
  - JEITA渡邊さんの解説で、よりTLMの理解が深まった。
  - 合成も早く標準化して欲しい。

 JEITA

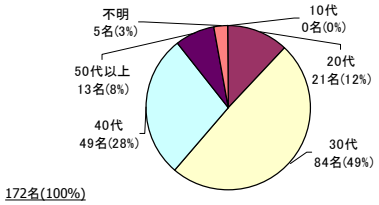
アンケート調査集計結果

- 昨年度とほぼ同様の内容(但し一部新規項目が追加されている)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。
  - 昨年同様にSystemCの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。
  - 年々SystemC導入済みが増えて、導入検討中が減っている。
  - SystemCのソフトウェア開発の利用が増加している。
  - TLMと合成の標準化は依然期待が高い。また標準インタフェースプロトコルの標準化が増加している。
  - 有料でも同フォーラム参加希望は増加している。

 JEITA

1. 申込者属性

■年代



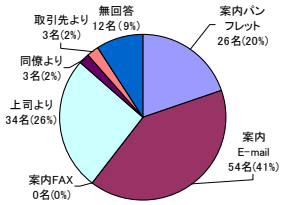
172名(100%)

エンジニアの中心は
30代～40代

案内パンフレットの効果あり
同僚からの勧めが増加

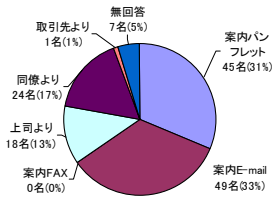
■本フォーラムを何で知りましたか？

2007年



132名(100%)

2008年



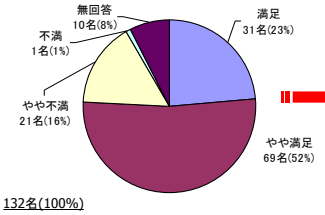
144名(100%)

JEITA

2. 本セッションの内容について

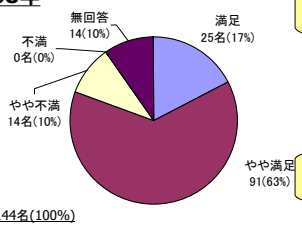
■内容は期待通りでしたか。

2007年



132名(100%)

2008年



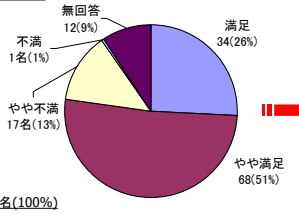
144名(100%)

チュートリアルの時間
不足による消化不良？

不満（やや不満）は昨
年より減っている

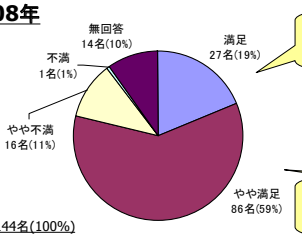
■予稿集の内容は期待通りでしたか？

2007年



132名(100%)

2008年



144名(100%)

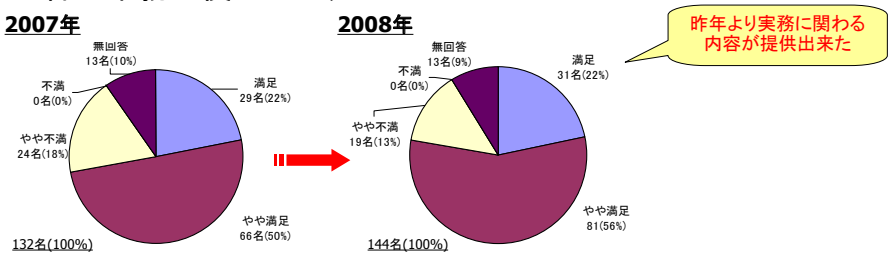
時間の関係で未使用
ページが多かった？

満足（やや満足）は昨
年より増加している

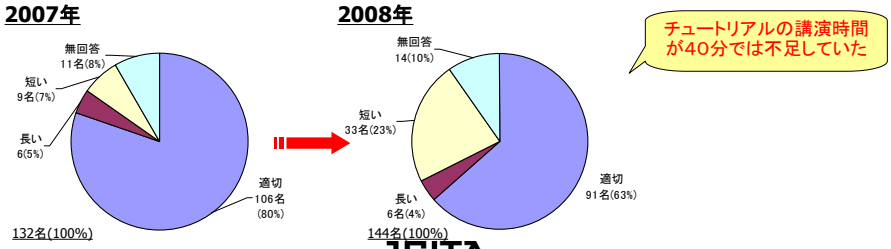
JEITA

2. 本セッションの内容について

■内容は業務に役立ちますか？

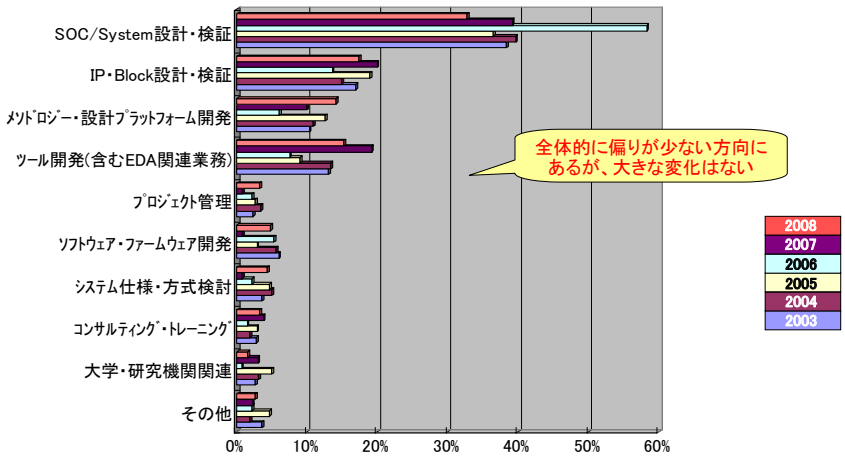


■講演時間



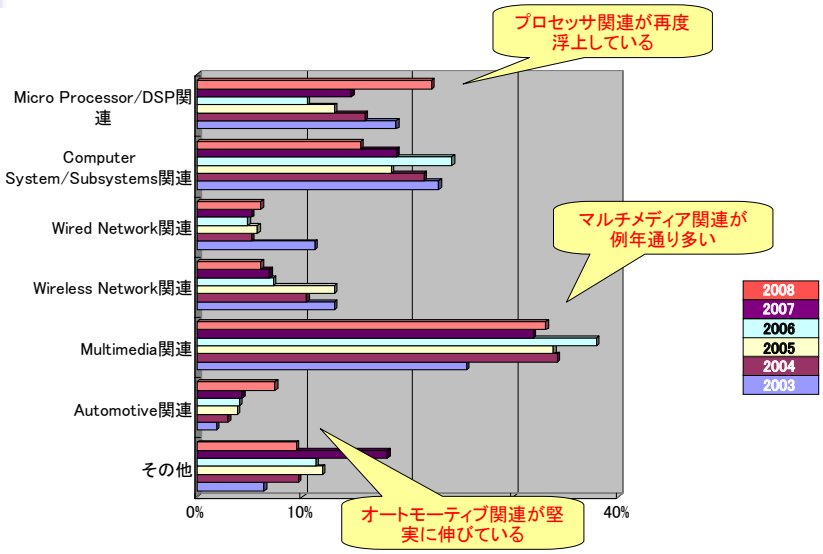
JEITA

3. ご担当業務またはビジネスは?



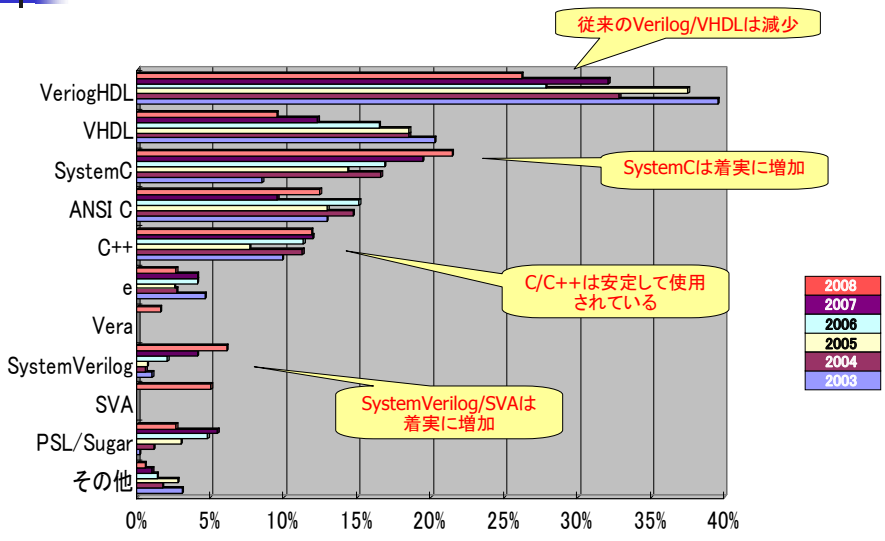
JEITA

4. ご担当製品アプリケーションは?



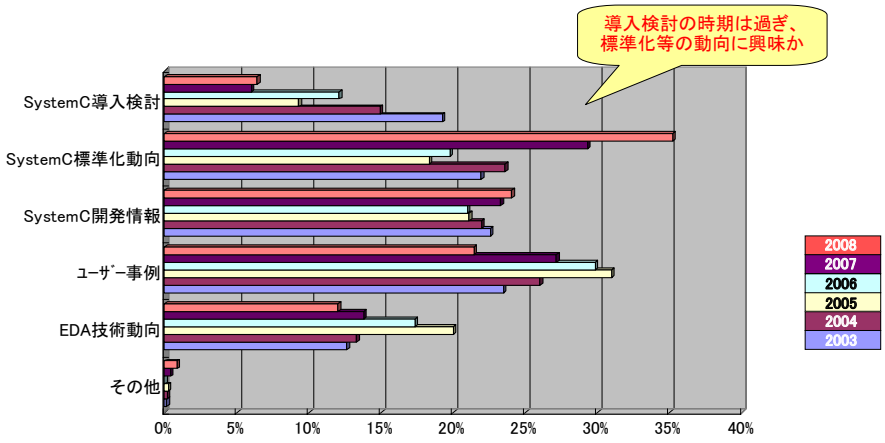
 JEITA

5. 現在主に使用している言語は?



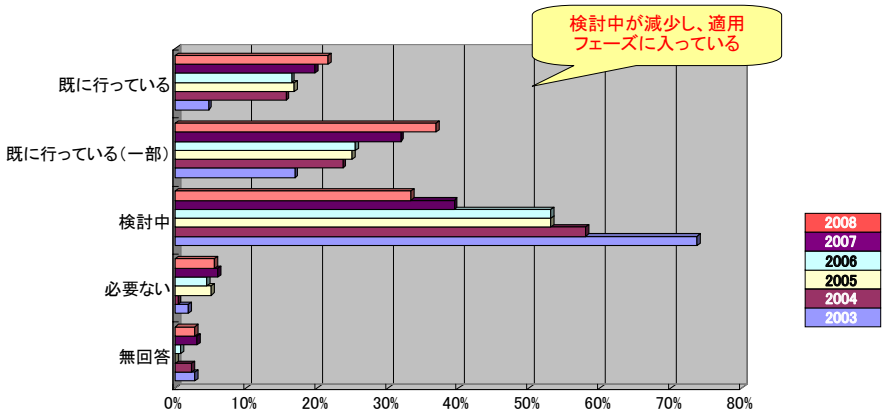
 JEITA

6. SystemCユーザフォーラムに参加された目的は?



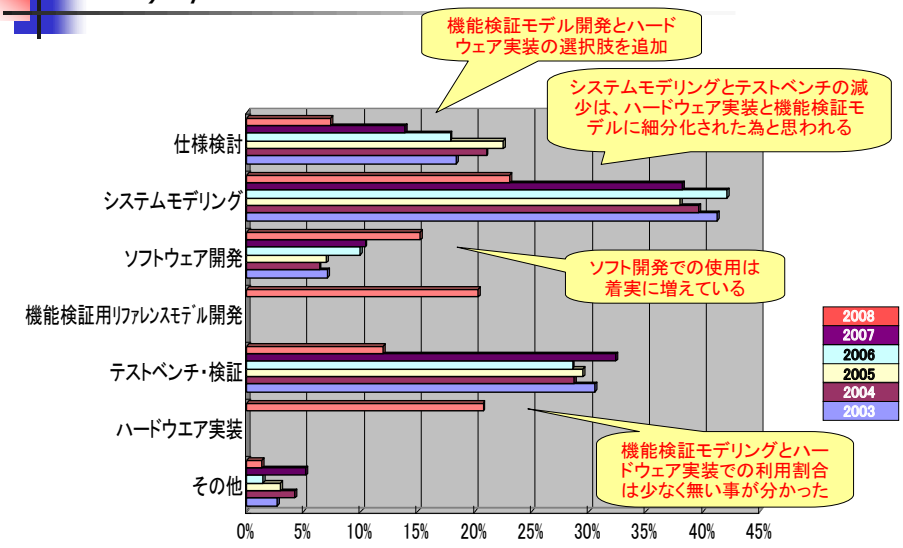
 JEITA

8. SystemCでの設計･検証環境構築について



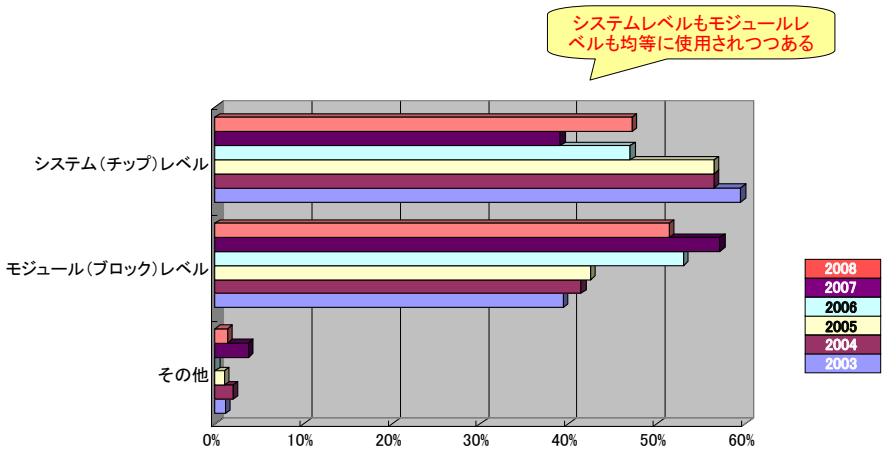
 JEITA

9.「8」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?



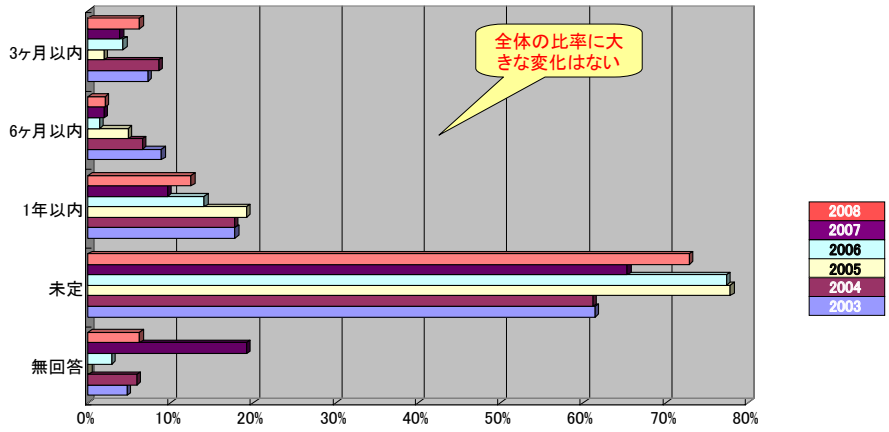
JEITA

9.「8」で「既に行っている」または「検討中」と回答された方
b) SystemCの活用範囲は?



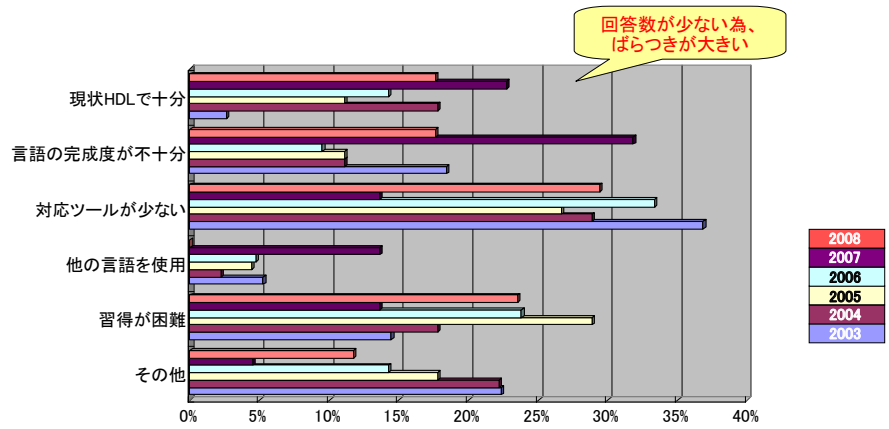
JEITA

10.「8」で「検討中」と回答された方へ 導入予定時期は?



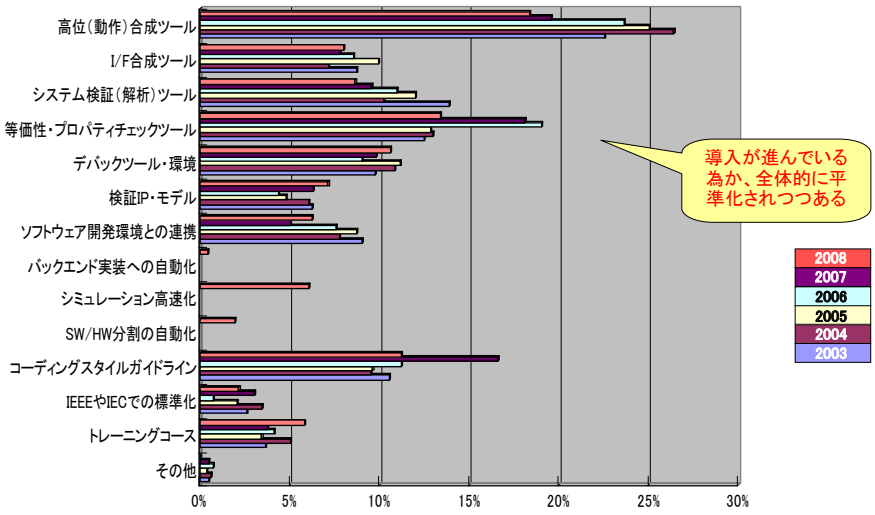
 JEITA

11.「8」で「必要ない」、「検討中」と回答された方へ 導入の弊害となっている理由は?



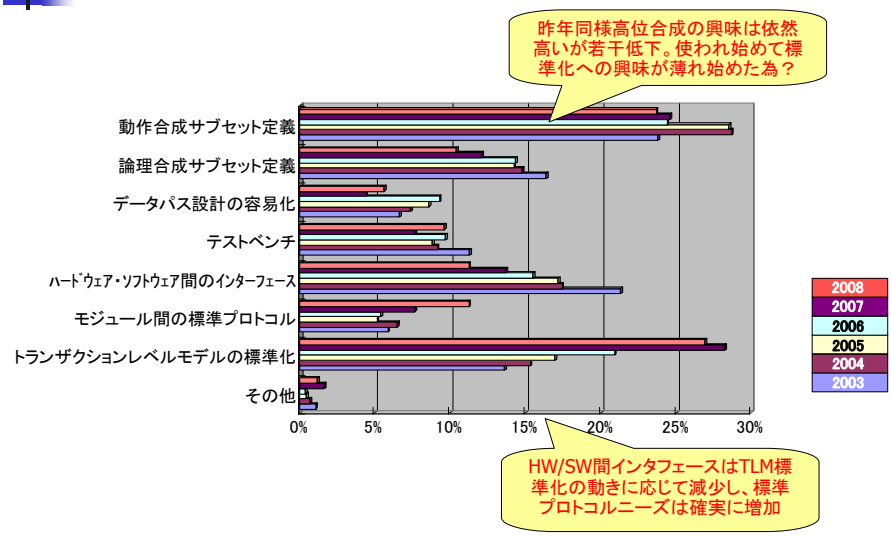
 JEITA

12. SystemCをより活用する為に充実が必要なものは？



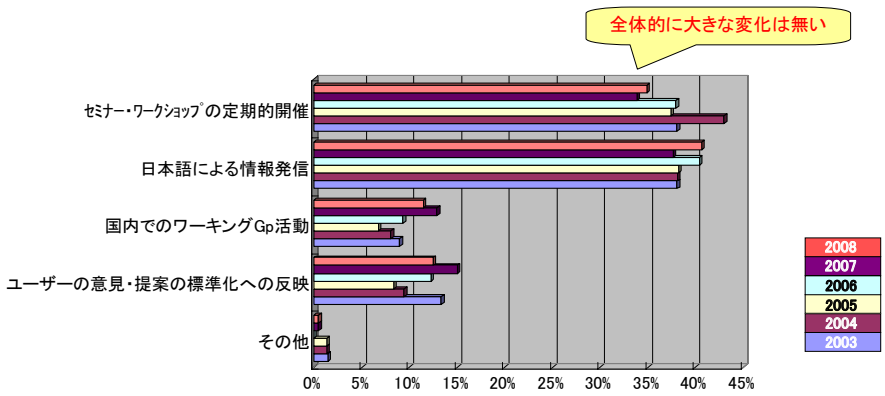
 JEITA

13. 今後SystemCの言語拡張･標準化で期待することは?



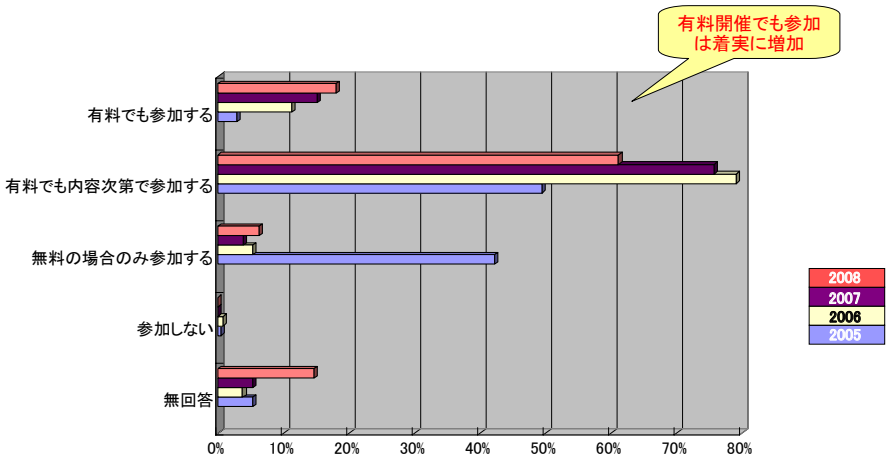
 JEITA

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



 JEITA

15.有料開催についてはどう思われますか？



 JEITA

SystemVerilog IEEEp1800 Draft3 レビュー結果 Issue List

了承	assigned	editorial/not fixed 再アピール	"Fixed" status is checked

Summary	Assigned ID	Status	JEITA ack	Action required(Feb.29, 2008
"HDL" doesn't seem to be a suitable expression for SystemVerilog. In some cases, SystemVerilog is referred as "HDVL.". Please discuss in TWG and define the terminology and keep consistency in the entire document.	NA	fixed		
"Elaboration occurs after parsing the HDL and before simulation" should be revised. The reasons are (1). because Compilation description has been added before this description and compilation has been defined as the pre-process of elaboration (2). not only simulation is the purpose to use SystemVerilog.	1825	not fixed closed	OK as our request makes the scope to change too large	
"IEEE Std 11364-2005" should be "IEEE Std 1364-2005"	NA	fixed		
There is no "'begin_keyword" description in index	NA	fixed		
Want to see the keyword list categorized by approved version. (e.g. 1800-2005 keywordA, keywordB; 1800-2008 keywordC, keywordE…;) Please see the attached file for the example.	1826	fixed	We expect this item will be fixed.	
WindowsNT is shown as the example to describe the difference of binary file operation. Need more updated OS example.	1827	fixed	OK	
"Syntax 20-22—" should be "Syntax 20-22" because the original contains redundant hyphens.	NA	fixed		

"Syntax21-"1 should be "Syntax21-7"	NA	fixed		
"Syntax21-2" should be "Syntax21-8"	NA	fixed		
"Syntax21-3" should be "Syntax21-9"	NA	fixed		
"SDF to Verilog delay value mapping" should be "SDF to SystemVerilog delay value mapping"	NA	fixed		
"SDF constructs mapping to Verilog" should be "SDF constructs mapping to SystemVerilog"	NA	fixed		
"SDF delay constructs mapping to Verilog declarations" should be "SDF delay constructs mapping to SystemVerilog declarations"	NA	fixed		
"Table" are at the top of the charts but "Syntax" are at the bottom of the charts. Using link-jump feature, we can jump to "Table"/"Syntax" but our expectation is to jump to the charts. To remove this inconvenience, please put both at the top of the charts.	NA	fixed		
In the definition of "pass_enable_switch_instance", "inout_terminal ,(black comma) inout_terminalred " should be "inout_terminal ,(red comma) inout_terminalred "	NA	fixed		
Syntax continues to the subject with "--" but Table continues to the subject with ":". Please unify.	NA	**NOT FIXED**	Please fix.	
on Page 668, "//from A.3.2" should be "//from A.5.2"	NA	fixed		
The hyperlink from Syntax29-5 on page 683 is not active.	NA	**NOT FIXED**	Please fix.	

"Software tools can perform" should be "Software tools should perform". Such warnings are mandatory to avoid describing unexpected behavior.	1828	fixed	Our intention was to remove these statements as LRM doesn't have to define how to implement in tools.	
"assert (req1 || req2);" in the example should be "assert (req1 || req2)"	NA	fixed		
"Table 7-2" should be "Table 8-2" on line 2 of page 124.	NA	fixed		
"fileId" should be "fileID" on line 6 of page 126.	NA	fixed		
we believe that usually logic [31:0] would be little endian.	1829	normal	Correction: 6.8 -> 11.5.1 As the previous example shows, using up/down-vect would be better for readers than big/little-vect.	push by e-mail
There are no Sequence methods(ended, triggered, matched) in the BNF	1830	fixed	We strongly expect this item will be fixed.	
The indentation of Yes/No in table18-29 are broken.	NA	fixed		
the title at the article and at the bookmarks in Acrobat is different	NA	fixed		
the title at the article and at the bookmarks in Acrobat is different	NA	fixed		
the title at the article and at the bookmarks in Acrobat is different	NA	fixed		
the title at the article and at the bookmarks in Acrobat is different	NA	fixed		
the title at the article and at the bookmarks in Acrobat is different	NA	fixed		

SystemVerilog-WG 3

"It shall be illegal for a module declaration to mix the port_reference port lists of non- ANSI style module headers with the list_of_port_declarations ports lists of ANSI style headers." should be "It shall be illegal for a module declaration to mix the port_reference port lists of non- ANSI style module headers with the list_of_port_declarations ports lists of ANSI style headers in one module" to explicitly show that the mixture of non-ANSI and ANSI in one module is prohibited.	1831	not fixed close	OK	
path rule/scope rule is not clearly described here. 22.7 definition would be mentioned here.remove "If $root is not specified, a hierarchical path is ambiguous. For example, A.B.C can mean the local A.B.C or the top-level A.B.C (assuming there is an instance A that contains an instance B at both the top level and in the current module). Verilog addresses that ambiguity The ambiguity is resolved by giving priority to the local scope and thereby preventing access to the top-level path. $root allows explicit access to the top level in those cases in which the name of the top-level module is insufficient to uniquely identify the path."	1832	urgent	We expect this item will be fixed.	push by e-mail
"The immediate assertion statement is a test of an expression performed when the statement is executed in the procedural code." should be more precisely defined, i.e. the scheduling should be independently defined to avoid racing. We recommend to execute the immediate assertions at "observed".We don't put this to the situation at "$display" which cause the different behavior for each simulator.	1833	fixed as #2005	We expect this item will be fixed.	
for PLI, tf_ and acc_ should be kept, at least in appendix for legacy code maintenance purpose	1834	not fixed closed	OK	
Give the name of the superset (summation) of Assertion API, Coverage API, Data read API and VPI. For example, SVPLI.	1835	urgent	We expect this item will be fixed.	push by e-mail

SystemVerilog-WG 4

PowerFormat 比較表

2007年度

PowerFormat 検討WG

1

目次

2

比較前提、共通コメント

※設計意図を表現する上で本質的でない次のようなユーティリティコマンドは表には記載していません。

CPF	UPF
set_design	load_upf
end_design	name_format
set_instance	save_upf
set_hierarchy_separator	save_scope
set_time_unit	create_hdl2upf_vct
…	create_upf2hdl_vct
	…

※表に表現不能な差異は以下にまとめます （フリー形式）

CPF	UPF
–	create_hdl2upf_vct
–	create_upf2hdl_vct

その他 CPF/UPF共通のコメント

分類	解説
物理検証(LVS)	CPF/UPFにるレベルシフタ、アイソレータ、パワースイッチ等の挿入は論理データベース（ネットリストまたはRTL）に対してまず行われます。 従って、LVSでは最終的なネットリストとGDSIIを比較すればよく、もはやCPF/UPFを必要としません。 （タイミング調整により挿入されるバッファと同等です） ＃論理データベースへの挿入が意図したとおり行われたかどうかは、LP検証等で確認されます。

3

多電源での比較 (1/4)

LP技術	多電源
前提条件	レイアウトはFLAT設計 DVFSのような動作時に電圧を変化させることは考えない パワードメインのネストはしない VSSは全ドメインで共通 アナログセルやI/Oは対象外とする

4

多電源での比較 (2/4)

設計フェーズ		目的	CPF ステートメント	UPF ステートメント	コメント
論理検証	L-1	信号の電圧を考慮したシミュレーション	create_power_domain update_power_domain	create_power_domain create_pst add_pst_state	※純粋にパワードメインのシミュレーションに必要なステートメントのみ記述
論理合成	S-1	レベルシフタ自動挿入	create_level_shifter_rule update_level_shifter_rules	map_level_shifter_cell set_level_shifter	
	S-2	電圧を考慮した遅延計算	create_nominal_condition update_nominal_condition create_power_mode update_power_mode define_library_set	create_power_domain create_pst add_pst_state	※DVFSを考えずに、電圧毎にセル名がちがっていれば、UPF設定は不要かもしれない。左欄は、電圧によってセル名が同じ時のことを考慮。
	S-3	電源ネットの生成	create_power_nets create_ground_nets create_global_connection	connect_supply_net create_supply_net create_supply_port set_domain_supply_net	
ライブラリ	Y-1	レベルシフタの定義	define_level_shifter_cell	*is_level_shifter*	※UPF自身にはライブラリ定義は無いが、期待されるlibertyのプロパティを記載（ただしセルの種類程度）
DFT	D-1	パワードメイン単位でのテスト回路挿入	create_power_domain update_power_domain	create_power_domain create_pst add_pst_state	

多電源での比較 (3/4)

設計フェーズ		目的	CPF ステートメント	UPF ステートメント	コメント
フォーマル検証	F-1	レベルシフタ挿入前後での等価性検証	(create_level_shifter_rule) (update_level_shifter_rules)	(set_level_shifter)	レベルシフタは等価性検証では元々無視されるので必須ではない
LP検証（VRC）	V-1	レベルシフタの抜け、冗長のチェック	create_level_shifter_rule update_level_shifter_rules	set_level_shifter	
	V-2	レベルシフタの種類の妥当性チェック	create_level_shifter_rule update_level_shifter_rules	map_level_shifter_cell	
P&R	I-1	電圧を考慮した遅延計算	create_nominal_condition update_nominal_condition create_power_mode update_power_mode define_library_set	create_power_domain create_pst add_pst_state	※DVFSを考えずに、電圧毎にセル名がちがっていれば、UPF設定は不要かもしれない。左欄は、電圧によってセル名が同じ時のことを考慮。
	I-2	電源ネットの生成	create_power_nets create_ground_nets create_global_connection	connect_supply_net create_supply_net create_supply_port set_domain_supply_net	
	I-3	電源ドメイン領域の生成（フロアプラン）	なし(*1)	なし(*1)	(*1)領域はインプリツールにて指定
	I-4	電源ドメイン領域を考慮した配置配線	なし(*1)	なし(*1)	(*1)領域はインプリツールにて指定

多電源での比較 (4/4)

設計フェーズ		目的	CPF ステートメント	UPF ステートメント	コメント
解析	A-1	電圧を考慮した遅延計算	create_nominal_condition update_nominal_condition create_power_mode update_power_mode define_library_set	create_power_domain create_pst add_pst_state	※DVFSを考えずに、電圧毎にセル名がちがっていれば、UPF設定は不要かもしれない。左欄は、電圧によってセル名が同じ時のことを考慮。
	A-2	電源ドメイン毎のXTALK解析	create_nominal_condition update_nominal_condition create_power_mode update_power_mode define_library_set	create_power_domain create_pst add_pst_state	
	A-3	電源ドメイン毎のIR-Drop解析	create_power_nets create_ground_nets create_global_connection	connect_supply_net create_supply_net create_supply_port set_domain_supply_net	
遅延計算・STA	T-1	電圧を考慮した遅延計算	create_nominal_condition update_nominal_condition create_power_mode update_power_mode define_library_set	create_power_domain create_pst add_pst_state	※DVFSを考えずに、電圧毎にセル名がちがっていれば、UPF設定は不要かもしれない。左欄は、電圧によってセル名が同じ時のことを考慮。

7

パワーゲーティングでの比較 (1/5)

LP技術	パワーゲーティング
前提条件	レイアウトはFLAT設計 パワードメインのネストはしない VSSは全ドメインで共通 アナログセルやI/Oは対象外とする パワースイッチはチップに搭載

8

パワーゲーティングでの比較 (2/5)

設計フェーズ		目的	CPF ステートメント	UPF ステートメント	コメント
消費電力計算	C-1	消費電力計算	set_switching_activity	create_power_domain create_pst add_pst_state	
論理検証	L-1	電源遮断ドメインからのZ or X出力	create_power_domain - shutoff_condition update_power_domain identify_power_logic	create_power_domain create_pst add_pst_state	※純粋にパワードメインのシミュレーションに必要なステートメントのみ記述
	L-2	リテンションFFのシミュレーション	create_state_retention_rule update_state_retention_rules	map_retention_cell set_retention set_retention_control	
	L-3	電源モード(ONドメイン、OFFドメインの組み合わせ)の検証	create_nominal_condition update_nominal_condition create_power_mode update_power_mode	create_power_domain create_pst add_pst_state	
	L-4	パワースイッチのON時間を考慮したシミュレーション	なし	create_power_switch - ack_delay	
	L-5	アイソレーションの確認	create_isolation_rule update_isolation_rule	map_isolation_cell set_isolation set_isolation_control	
	L-6	オールウェイズオンバッファを含むシミュレーション	(define_always_on_cell)	なし	

パワーゲーティングでの比較 (3/5)

設計フェーズ		目的	CPF ステートメント	UPF ステートメント	コメント
論理合成	S-1	アイソレータ自動挿入	create_isolation_rule update_isolation_rule	map_isolation_cell set_isolation set_isolation_control	
	S-2	電源ネットの生成	create_power_nets create_ground_nets create_global_connection	connect_supply_net create_supply_net create_supply_port set_domain_supply_net	
	S-3	リテンションFFのマッピング	create_state_retention_rule update_state_retention_rules	map_retention_cell set_retention set_retention_control	
ライブラリ	Y-1	アイソレータの定義	define_isolation_cell	*is_isolation_cell*	
	Y-2	パワースイッチの定義	define_power_switch_cell	*switch_cell_type*	
	Y-3	リテンションFFの定義	define_state_retention_cell	*retention_cell*	
	Y-4	オールウエィズオンバッファの定義	define_always_on_cell	*特に必要なし(*1)*	(*1)パワーレールを複数定義する事によりalwayson cellを実現可能 ※UPF自身にはライブラリ定義は無いが、期待されるlibertyのプロパティを記載(ただしセルの種類程度)

パワーゲーティングでの比較 (4/5)

設計フェーズ		目的	CPF ステートメント	UPF ステートメント	コメント
DFT	D-1	パワードメイン単位でのテスト回路挿入	create_isolation_rule update_isolation_rule	map_isolation_cell set_isolation set_isolation_control	
フォーマル検証	F-1	アイソレータ挿入前後での等価性検証	create_isolation_rule update_isolation_rule	set_isolation set_isolation_control	
	F-2	リテンションFFマッピング前後での等価性検証	create_state_retention_rule update_state_retention_rules	map_retention_cell set_retention set_retention_control	
	F-3	パワースイッチ挿入前後での等価性検証	create_power_switch_rule update_power_switch_rule	set_power_switch	
LP検証(VRC)	V-1	アイソレータの抜け、冗長のチェック	create_isolation_rule update_isolation_rule	set_isolation set_isolation_control	
	V-2	アイソレータの種類の妥当性チェック	create_isolation_rule update_isolation_rule	map_isolation_cell	
	V-3	パワースイッチの接続チェック	create_power_switch_rule update_power_switch_rule	set_power_switch	

パワーゲーティングでの比較 (5/5)

設計フェーズ		目的	CPF ステートメント	UPF ステートメント	コメント
P&R	I-1	電源ネットの生成	create_power_nets create_ground_nets create_global_connection	connect_supply_net create_supply_net create_supply_port set_domain_supply_net	
	I-2	電源ドメイン領域の生成（フロアプラン）	なし(*1)	なし(*1)	(*1)領域はインプリツールにて指定
	I-3	電源ドメイン領域を考慮した配置配線	なし(*1)	なし(*1)	(*1)領域はインプリツールにて指定
	I-4	パワースイッチの挿入	create_power_switch_rule update_power_switch_rule	set_power_switch create_power_switch map_power_switch	
	I-5	オールウェイズオンバッファの挿入	なし(*1)	オールウエイズオンバッファ定義がない	(*1)挿入はインプリツールにて指定
解析	A-1	PSWを含んだ Static IR-Drop	必要なし	必要なし	
	A-2	PSW ON時のDynamic IR-Drop	必要なし	必要なし	

EDAアニュアルレポート　2007

2008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
〒101-0065
東京都千代田区西神田3-2-1
千代田ファーストビル南館
電話　03-5275-7258　FAX　03-5212-8121

作　　成　　三協印刷株式会社
〒152-0002
東京都目黒区目黒本町5-20-7
電話　03-3793-5971　FAX　03-3793-6242