第 3 章 WG1 設計

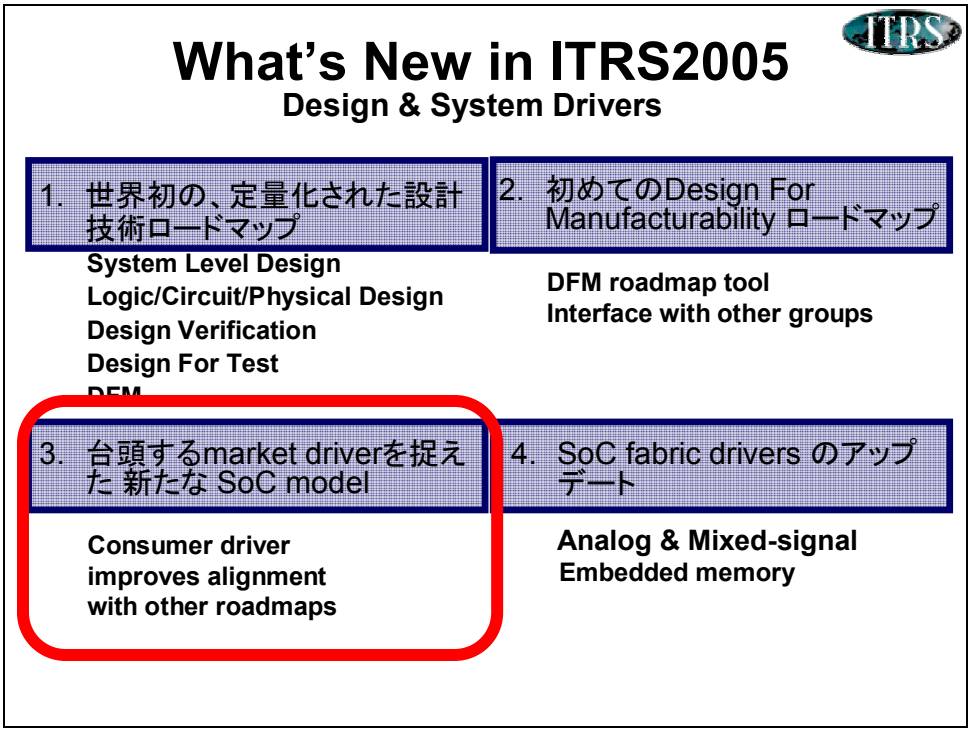
3-1 概要

国際活動は、国際担当を中心に活動を行い、ITRS2005 に対して貢献を行った。国内活動は、設計枠組み(工程)別のサブワーキンググループを中心に活動を行い、1999 年度に作成した SoC 設計技術ロードマップの見直しを行った。

3-2 国際活動

4 月、7 月、12 月、2 月にそれぞれ欧州(ドイツ/ミュンヘン)、米国(サンフランシスコ)、韓国(ソウル)、米国(サンフランシスコ)で行われた ITRS の Design ITWG に出席した。国際担当を中心とした活動では、ITRS2005 に対して 3 つの貢献を行った。①2004 年度国内活動成果を System Drivers 章の SoC Driver として提案し、採択された。②Design 章の新たな章構成を提案し、採択された。③Design 章の定量化されたテーブルについてレビューを行い、フィードバックをかけた。この中で、Design for Test のテーブルは、WG2 の DFT-SWG にご協力いただいた。

特に①については、ITRS2005 の What's New のひとつ(図表 3-1)にも記載され、大きな貢献となった。



図表 3-1 What's New in ITRS2005

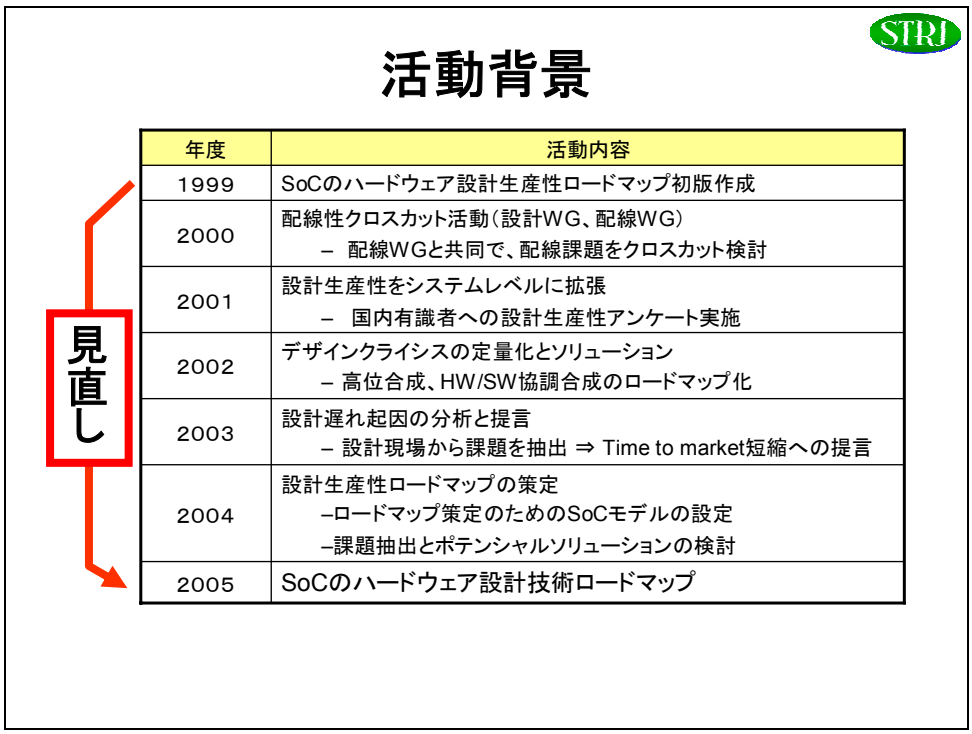
3-3 国内活動

設計枠組み(工程)別のサブワーキンググループを中心に活動を行い、1999 年度に作成した SoC 設計技術ロードマップの見直しを行った。①2004 年度国内活動および ITRS2005 の SoC Driver を用いて、設計生産性向上要求の見直しを行った。②設計技術の変遷に影響を受けにくい設計の枠組みを策定した。③設計の枠組み毎に技術課題とその課題解決策の見直しを行った。④設計の枠組み毎にメッセージと重要な技術課題を明示した。

3-3-1 背景

WG1 は発足以来、設計技術のロードマップ活動を行っており、1999 年には SoC のプロファイリングによる設

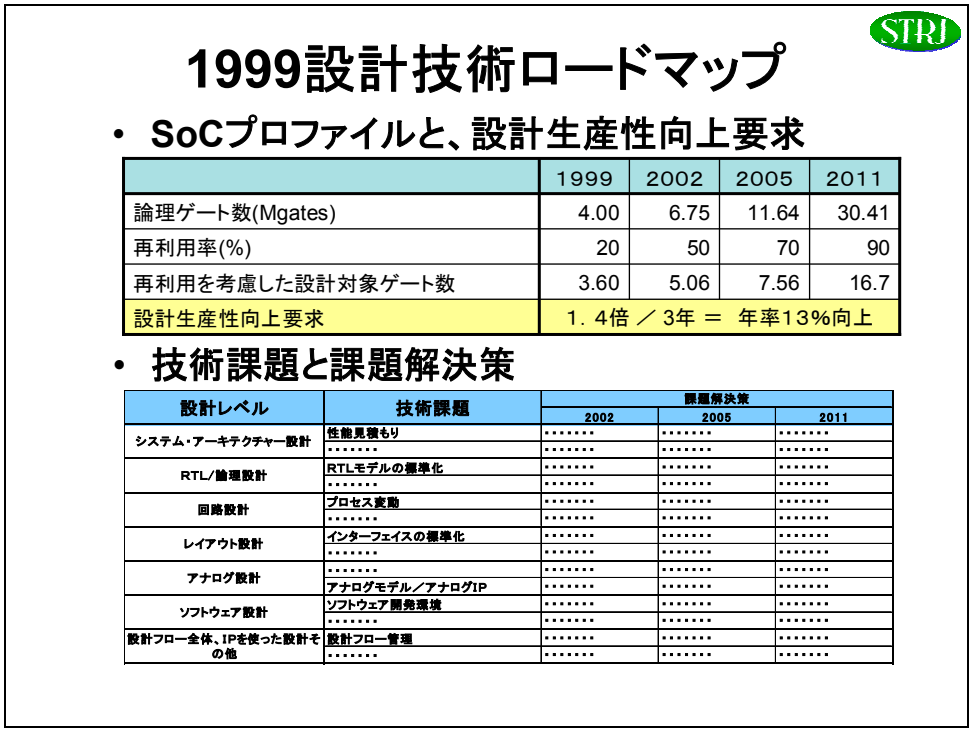
計生産性向上の要求を定量化し、これを実現するにあたっての技術課題と課題解決策を提言した。この 5 年間は様々なアプローチから設計生産性の向上について検討を重ねてきたが、ここで、6 年ぶりに SoC 設計生産性のロードマップの改版をすることにした。



図表 3-2 活動背景

3-3-2 1999 年度の設計技術ロードマップ

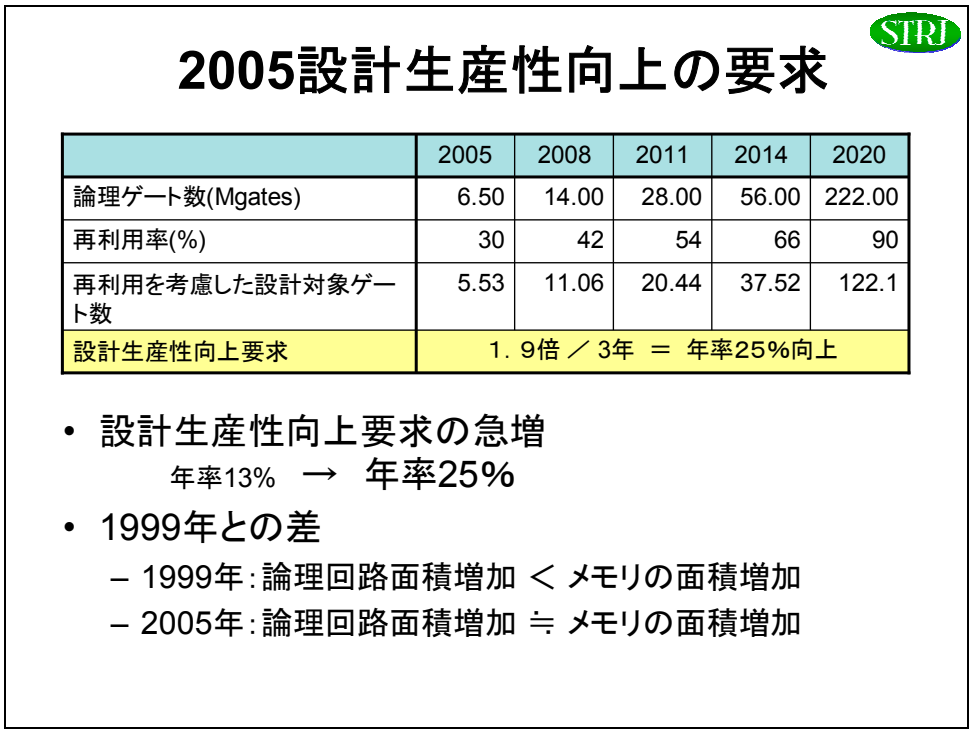
1999 年のロードマップは、SoC プロファイルによる設計生産性要求ロードマップと、技術課題ごとの課題解決策ロードマップの、二つ（図表 3-3）で構成されている。



図表 3-3 1999 年度の設計技術ロードマップ

3-3-3　設計生産性向上要求の見直し

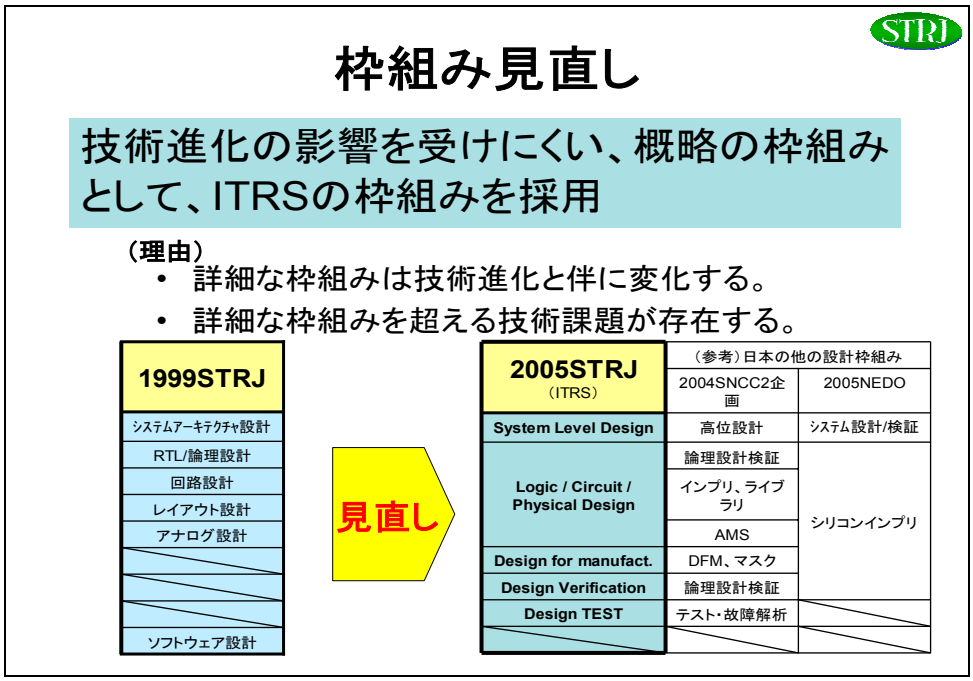
2004 年度国内活動および ITRS2005 の SoC Driver を用いて SoC のプロファイリングを行い、これに基づいて設計生産性向上要求を見直した。1999 年では論理ゲートの増加がメモリの面積増加により抑えられていたのが、2005 年ではより現実的な SoC 構造から論理ゲートの増加とメモリの面積増加がほぼ等価と算出され、1999 年には年率 13%であった設計生産性向上要求が、年率 25%ときわめて高い要求になった。



図表 3-4 設計生産性向上要求の見直し

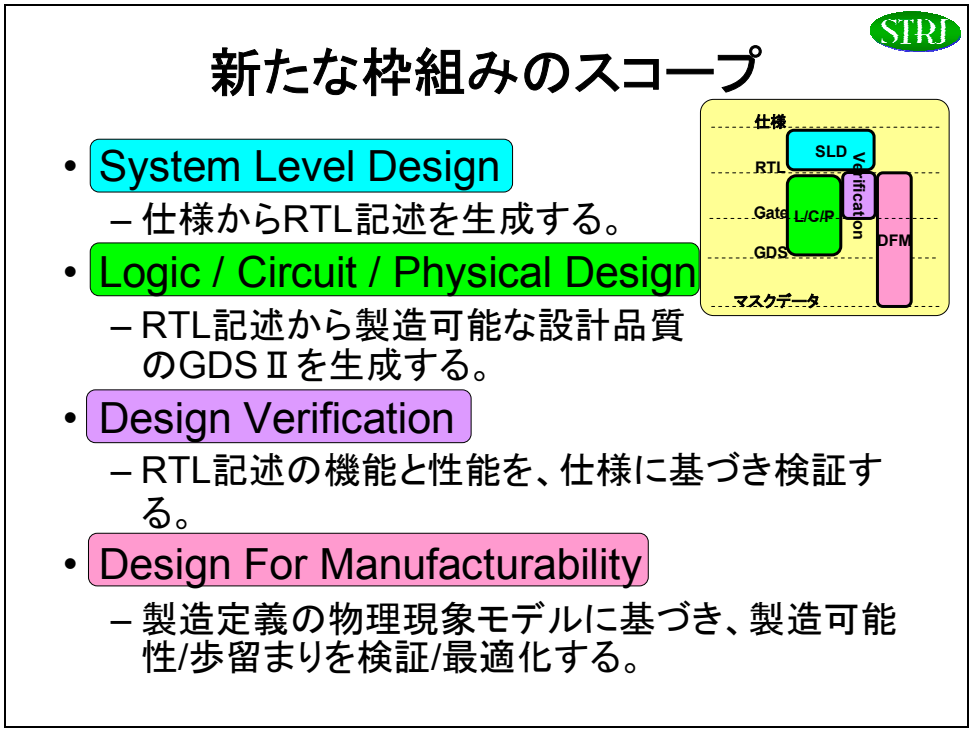
3-3-4　枠組みの見直し

技術課題、課題解決策の見直しには、設計工程全般にわたる広範囲の作業が必要なため、設計工程ごとの分担作業を進めることにした。



図表 3-5　枠組みの見直し

しかし、1999 年の設計工程は当時の設計技術を正確に反映しており、この粒度では設計技術の変遷と伴に見直しが必要となる。このため、設計技術変遷の影響を受けにくい、普遍的な枠組みに変更することにした。様々なロードマップの枠組みを検討した結果、ITRS の枠組みを採用(図表 3-5)することとした。さらにここで、分担作業での重複や漏れをなくすため、それぞれのスコープを明確化(図表 3-6)することにした。



図表 3-6 スコープの明確化

3-3-5 メッセージと重要な技術課題

1999 年のように網羅的に技術課題を抽出し、課題解決策を提示することも重要だが、2005 年では、WG 内の有識者がもつ危機感を、メッセージとして述べることにした。以下に、各枠組みのメッセージのサマリを説明する。下線部分は各枠組の有識者が考える最も重要な技術課題である。

[System Level Design]

SoC 開発の成否は、「新規仕様の取り込み、追加仕様への対応」と「設計/製造コストの制約実現」の高度なトレードオフを正確に判断することにかかっている。正確な判断をするには、仕様追加/変更が設計/製造コストに与える影響の定量的な把握が課題である。技術課題として「仕様変更がプロジェクト(PJ)マネジメント(コスト・期間)に与える影響の定量的な予測」が最重要と考えた。

[Logic / Circuit / Physical Design]

プロセスの微細化に伴うゲート数増加、電流密度増大など、消費電力が増大する要因が急激に増加している。消費電力の増大を抑えるための低消費電力設計が非常に複雑化し、設計工程の後戻り、設計作業の繰り返しが増大している。技術課題として「イタレーションのない低消費電力設計」が最重要と考えた。

[Design Verification]

開発コストの増大を抑えるには、設計工数の 70%を占める検証工程の効率化が必須となっている。リスピンをなくす検証品質の向上を、効率化とともに実現するには、適切な検証戦略の作成と検証実行の自動化が課題である。技術課題として、「検証項目の抽出・検証モデル/テストベンチの作成」の効率化・完成度向上が最重要と考えた。

[Design For Manufacturability]

従来の一義的なデザインルールやコーナー条件によるデバイスモデルだけでは目標歩留まりの達成が困

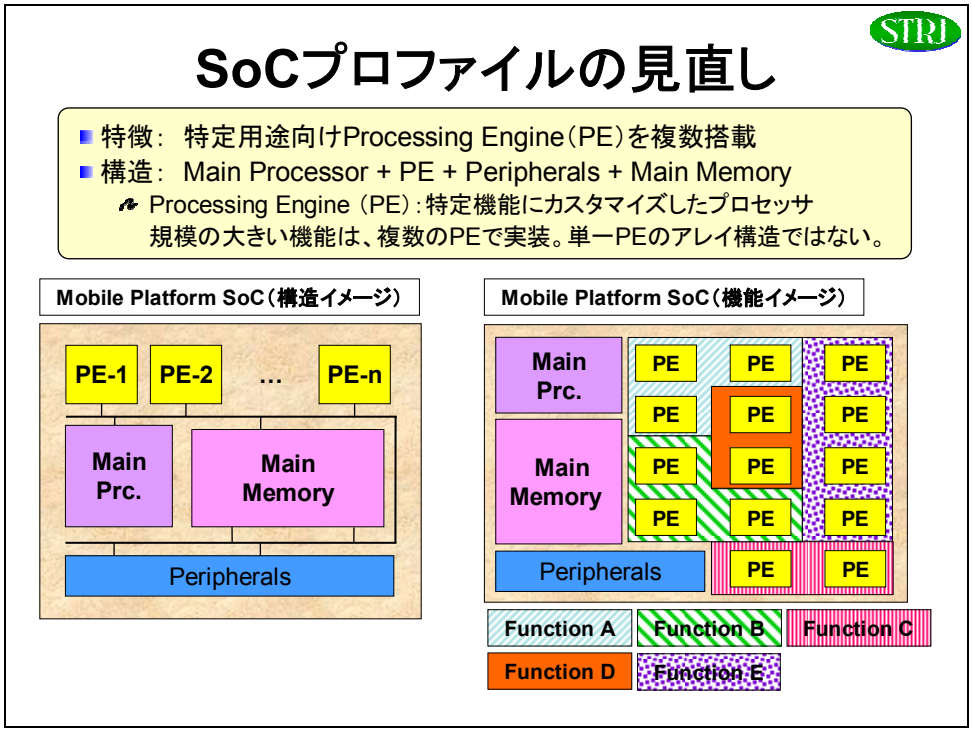
難になってきた。微細化に伴う製造困難性を設計工程で改善することにより、歩留まりを改善したい。技術課題として、「設計目標を製品固有の設計属性とプロセス固有の故障可能性の両者の関数として表現すること」が最重要と考えた。

3-4 （国内活動成果）SoC 設計技術ロードマップ

以下に 2005 年度国内活動の成果である、SoC 設計技術ロードマップを説明する。

3-4-1 SoC プロファイルと設計生産性向上要求

2004 年度国内活動および ITRS2005 の SoC Driver を用いて SoC のプロファイリングを行った。



図表 3-7　SoC プロファイリング

図表 3-7 のアーキテクチャをもとに、図表 3-8 に示す定量化を行った。定量化の基本的考え方を次に説明するが、詳細は昨年度の報告書（半導体技術ロードマップ専門委員会　平成 16 年度報告　第 3 章　WG1 設計 P39～42）を参照いただきたい。

まず、図表 3-8 の設計規模の項目を参照いただきたい。ここでは、SoC 全体に搭載される、論理回路規模（ゲート数）とメモリ容量を求めている。面積見積りの基礎データであるレイアウト密度は、ITRS2005 から引用している。設計規模が拡大するにもかかわらず、市場から要求される設計工期短縮要求とコスト要求は厳しいものがあり、許される設計工数と許される SoC のダイサイズは 1999 年同様、一定と考える。このほか、アナログ、I/O 等のオーバヘッドの面積比率、Main Processor および PE（Processing Engine）の論理ゲート数と内蔵メモリ容量は、技術の進化に関わらず一定と仮定する。微細化にともない、Main Processor と PE の 1 個あたりの面積は小さくなり、より多くの PE を搭載することができる。なお、Main Memory は PE の数に比例して容量が増えるものとした。PE が増えるにつれて対象とする処理のデータ量が増えるためである。

次に、図表 3-8 の演算能力の項目を参照いただきたい。ここでは、SoC 全体の演算能力を求めている。性能の基礎データは ITRS2005 の LOP（Low Operating Power）トランジスタを用いている。SoC 全体の演算方力は、〔動作周波数の向上〕×〔PE 数の向上〕 に比例する。

最後に、図表 3-8 の設計エフォートの項目を参照いただきたい。ここでは、設計工数が、搭載される論理回

路規模に比例するものとし、増大していく設計工数を回路再利用率の向上と、設計生産性の向上で、克服していく必要があることを示している。回路再利用率は、2005 年に現状の SoC 開発での平均値と予想される 30%を、2020 年にあるべき回路再利用率 90%を置き、その間は一次関数的に単調増加すると仮定する。再利用にあたっては、1999 年と同様、新規設計の半分の工数が必要と仮定した。再利用設計においてもテクノロジが変わればライブラリが変わり、インプリ工程が必要となり、As-Is で使えるものではないと考えたからである。これらの結果、2020 年に 90%という高い回路再利用率をもってしても、3 年ごとに 1.9 倍の設計生産性向上が要求されることとなった。

	Year	単位	2005	2008	2011	2014	2020
	Metal-1 Half Pitch	nm	90	58	40	28	14
設計規模	論理回路の割合(対チップ面積)	%	28%	21%	19%	19%	19%
	メモリの割合(対チップ面積)	%	45%	52%	53%	53%	53%
	オーバヘッドの割合(対チップ面積)	%	27%	28%	28%	28%	28%
	論理回路のレイアウト密度	M gates/cm^2	36	104	223	456	1,822
	搭載論理回路規模	M gates	6.5	14	28	56	222
	メモリのレイアウト密度	M bits/cm^2	60	151	325	681	2,817
	搭載メモリ容量	M bits	18	51	110	231	956
演算能力	電源電圧	V	0.9	0.8	0.7	0.6	0.5
	動作周波数	MHz	390	577	859	1,253	2,766
	Processing Engine(PE)数	個	16	46	101	212	878
	演算能力	相対値	1.0	4.3	14	43	392
設計エフォート	設計期間(改革無し)	相対値	1.0	2.2	4.3	8.5	34.2
	回路再利用率	%	30%	42%	54%	66%	90%
	新規設計	M gates	4.6	8.1	13	19	22
	新規設計工数(1M gates 当たり)	相対値	1.00	0.50	0.27	0.15	0.05
	再利用工数(1M gates 当たり)	相対値	0.50	0.25	0.14	0.07	0.02
	設計生産性向上要求		**1.9 倍 / 3 年 ＝ 年率 25%向上**				

図表 3-8 設計生産性向上要求

3-4-2 設計技術課題と課題解決策

以上のような高い設計生産性向上を実現するために、各設計工程ではいかなる技術課題があり、それを時系列にいかにして解決していくべきかを以下に、提言する。

3-4-2-1 System Level Design

〔**スコープ**〕仕様から実装可能な設計記述 RTL を生成するまでを基本対象とする。しかし、高位合成技術については、現状の論理合成技術の普及範囲(フィジカル合成等)まで浸透しないと、現状の RTL 設計者への普及は見込めない。

〔**前提条件**〕SoC のプロファイリング(図表 3-7)によれば、性能を引き出すにはソフトウェア(SW)の大幅な変更が必要になるが、PE 上での SW 含めたシステム検証等に関しては今回触れない。

〔**メッセージ**〕

- システムの肥大化に伴い仕様設計の重要性は増大するもこの分野の設計環境の整備は未だ進んでいない。システム仕様として定義すべき物の明確化と仕様記述の標準化(人の為と EDA ツールの為の 2 方向)が必要である。一般に機能要件だけが仕様として取り扱われがちであるが、後述する非機能要件も重要な要求仕様の一部であることを忘れてはならない。
- SoC 開発に関わる顧客、マネージメント、マーケティング、セット設計者、ハード設計者、ソフト設計者達の間で、お互いにインプリする仕様共有化できる仕様記述言語が切望されていると共にとその仕様に対するコスト・期間も共有化することが SoC 開発の上で重要である。
- 上記から「**仕様変更が PJ マネジメント(コスト・期間)に与える影響の定量的な予測**」を重要な技術課題とする。
  - システム仕様は、機能要件と非機能要件に分類できる。
    - 機能要件 :システムの動作を定義。フォーマルなものが良いが厳密すぎると可読性が落ちる。ハード設計の流れから SystemC 等、ソフトウェア設計の流れから UML 等のアプローチがあるが、記述容易性や視認性などの面で満足できる完成度ではない。
    - 非機能要件: 目標チップサイズ、動作電源・電圧、動作周波数、消費電力、動作温度範囲、電源分離方式、電源遮断方式、DFT 手順、ピン配置、パッケージ等

〔重要な技術課題と課題解決策〕

- 重要な技術課題
  - 仕様変更が PJ マネジメント(コスト・期間)に与える影響の定量的な予測
- 着目した理由
  - 高位レベル設計工程での効率化を実際の設計生産性の向上として反映する事が必要であり、頻繁な仕様変更に起因する開発遅延を削減し市場要求に応じた的確な製品開発を行いたい
- 詳細な説明
  - SoC 開発の成否は、「差異化仕様を最大限盛り込むこと」と「開発工期(コスト)、製造コストの制約を実現すること」の高度なトレードオフを正確に判断することにかかっている。トレードオフを正確に判断するためには、仕様を追加/削除した時に影響を受ける設計/検証作業が明確になり、開発工期(コスト)、製造コストの増減が定量的に把握できる必要がある。現在は、仕様変更の影響の定量的把握に基づくトレードオフの正確な判断ができないため、意欲的過ぎる仕様の盛り込みによる開発遅れや、保守的過ぎる開発計画による製品企画の失敗がしばしば起きている。一方、差異化のためには、設計途中であっても仕様変更が必要なことがあり、設計が進むにつれて仕様変更の影響が大きくなる為、その定量的把握がますます大切になる。
- 実現時の効果
  - 仕様の追加/削減を行うと、開発工期(コスト)、製造コストへの影響が容易に・自動的に・正確に見積れるようになる。その結果、様々な仕様の追加/削減を探索し、最も効果的(差異化/影響)な仕様を検討できるようになる。また、無理のない開発計画を立てられる為、開発遅れが激減する。更に、開発途中での仕様変更に対しても、変更の影響度を確実に把握できる事から開発期間を含めた見直しが的確に行える。
- 時間軸を伴った解決策の提示
  - 2005～2008 年: 仕様変更の影響度を把握するために、仕様の構成要素と対応する各設計工程の分類が行われ、設計工程毎の基本データを蓄積するためのインフラ技術が確立される。IP を載せかえることで仕様変更を行うような設計(派生設計)に関して、基本データの蓄積が進む。蓄積した基本データから、限られた範囲内ではあるが、仕様変更が設計工程へ及ぼす

影響を測ることが可能となる。

- 2009～2011 年:機能要件の検証環境に加え、非機能要件の実現性に関して動的検証環境が整備され、事前に各要求値の妥当性が評価出来るようになり、確認された要求値に従った設計が出来るようになる。IP の新規設計により仕様変更を行うような設計(新規設計)においても基本データの蓄積が進み、更に広い範囲内で、仕様変更が設計工程へ及ぼす影響を測ることが可能となる。
- 2012～2014 年:仕様変更の影響度を把握できるようになる。仕様記述と各設計工程での設計工数面・コスト面での見積もりがリンクされる。

〔その他の技術課題と課題解決策〕

以下に、System Level Design の設計工程を、仕様設計、アーキテクチャ設計、実装設計に分け、分析する。

①仕様設計における技術課題と課題解決策

- 現状、仕様は記述方法・記載内容も統一されずに自然言語で記述されるため、記載内容の曖昧さによる誤解や記載抜け・記載内容の矛盾や EDA ツール毎の入力への人手変換指示ミスなどの問題が顕在化しており下記を技術課題として挙げた。
  - 仕様の明確化
  - 仕様の妥当性の検証
  - アーキテクチャ設計への接続
- 実現時の効果
  - 機能要件と非機能要件を含んだ仕様記述の標準化とともに、仕様ベースでの設計環境の整備が進み、仕様記述による標準規格のモデル化が普及する。
- 時間軸を伴った解決策の提示
  - 2005～2008 年
    - 静的検証による仕様の矛盾検証や機能面での仕様のコーナーケースを抽出できるようになり、機能要件の仕様記述の標準化が進む。
  - 2009～2011 年
    - 性能、電力、リアルタイム性など非機能要件の検証手法や非機能要件のバジェッティングが可能となり、非機能要件を含めた仕様記述の標準化が進む
  - 2012～2014 年
    - 仕様変更へのリスク分析が出来るようになり、仕様記述による標準規格のモデル化が普及する

②アーキテクチャ設計における技術課題と課題解決策

- アーキテクチャ設計は、現状アーキテクチャ設計者の経験と勘に頼らざるを得ず、SW 含めた性能検証が出来るのは設計後期である。このため性能未達の場合、アーキテクチャ見直しとなり大きなイタレーションが発生するなどの問題が顕在化している。下記を技術課題として挙げた。
  - 最適な HW/SW 分割
  - 高精度な HW アーキテクチャ設計
  - 容易な HW アーキテクチャ探索
- 実現時の効果
  - 容易にアーキテクチャ探索出来るようになり、アルゴリズム記述から最適な HW/SW 分割ができ、かつ確認取れたバジェッティングによるトップダウン設計ができる様になる。
- 時間軸を伴った解決策の提示
  - 2005～2008 年

- 高位記述モデルで機能の並列性など各種分析機能が充実し、高位モデルでの記述の標準化・整備・流通が進む。

- 2009～2011 年
  - 消費電力・DFT・テスト時間など高位モデルと実装レベルの対応付けが出来るようになると共に、機能の取捨選択が容易に行える設計環境が整備される。
- 2012～2014 年
  - アルゴリズム記述からの最適なアーキテクチャ選択が容易に出来るようになる。

③実装設計における技術課題と課題解決策

- 実装設計では、規模増大に伴い開発遅延や設計工数増大が問題となっているが、現状の高位レベルの実装設計技術はメソドロジとして確立していないため下記を技術課題として挙げた。
  - 設計規模拡大を凌駕する検証効率の向上
  - 高位合成技術高度化
  - 高位レベルからの再利用設計の浸透
  - 既存 RTL 設計と連携した高位設計
- 実現時の効果
  - 設計進度に応じた階層的な検証技術が確立され検証効率が飛躍的に向上すると共に高位レベルでの Plug&Play の実現で再利用設計がようやく浸透する。
- 時間軸を伴った解決策の提示
  - 2005～2008 年
    - インターフェース含めた合成が出来る様になると共にプロパティベースの検証 IP の整備・流通が進む。
  - 2009～2011 年
    - アプリケーション分野別のプラットフォームが高位レベルに拡張され、高位レベルでの Plug&Play が容易に出来るようになりようやく再利用設計が浸透する。設計レベルを統合した検証環境が整備される。
  - 2012～2014 年
    - レイアウト情報を加味した最適化を行うまで高位合成技術が拡大すると共に、高位レベルでの等価性検証が実現される。仕様レベルで接続チェックが可能となる。

技術課題

設計ﾚﾍﾞﾙ	技術課題	説　明
仕様設計	仕様の明確化	SOC開発に携わる顧客、ﾏﾈｼﾞﾒﾝﾄ、各設計者が共通理解できる仕様記述
	仕様妥当性の検証	仕様設計段階での仕様の妥当性を詳細レベルで検証
	仕様変更がコスト/期間に与える影響の定量的な予測	仕様の追加・変更に対する定量的なコストを予測し、SOC開発に携わる顧客、マネジメント、マーケティング、セット設計者、ハード設計者、ソフト設計者達の間で共有する
	アーキ設計連携	仕様記述からアーキテクチャ設計言語への自動的変換
アーキテクチャ設計	最適なHW/SW分割	求められる性能/コスト条件を満足する機能のHW/SW分割
	高精度HWｱｰｷﾃｸﾁｬ設計	高位モデルでの精度の高い性能検証
	容易なHWｱｰｷﾃｸﾁｬ探索	高位モデルの取捨選択、結合が容易にできる
実装設計	設計規模拡大を凌駕する検証効率の向上	検証速度と網羅性の向上のための検証抽象度の向上と、下位での検証を省略するための等価性検証技術の確立
	高位合成技術高度化	現在の論理合成並みに手軽に扱え、精度の高い高位合成
	高位ﾚﾍﾞﾙからの再利用設計の浸透	I/F含めた高位モデルでのPlug&Playの実現
	既存RTL設計と連携した高位設計	既存の設計資産（RTL, チップ）を有効活用できる高位設計

図表 3-9　System Level Design の技術課題

仕様設計における課題解決策

設計レベル	技術課題	課題解決策		
		2005～2008	2009～2011	2012～2014
仕様設計	表記方法	・MinSetを決めて誤解しない範囲 ・分野別PF化の為の部品整備 ・機能要件のみの仕様記述の標準化	・平行・並列性記述の拡充 ・非機能要件の記述の妥当性　（共通認識として使えるレベルで） ・非機能要件含めた仕様記述の標準化	・仕様レベル（機能要件＋非機能要件）で製品の複雑度 ・実現の難しさを導出 ・仕様記述による標準規格のモデル化普及
	仕様の妥当性の検証	・設計者のための仕様記述言語制定 ・数学的な検証による仕様の矛盾確認 ・仕様記述チェッカ ・仕様のコーナケース抽出（機能）	・非機能要件仕様の検証手法の確立（テストベクタ生成等） ◎：性能、電力 △：ﾘｱﾙﾀｲﾑ性 ×：その他 ・仕様のコーナケース抽出　（非機能要件）	・ドメイン固有の知識に基づく　仕様の矛盾確認 ・非機能要件仕様のカバレジ　手法の確立
	仕様変更がPJマネジメント（コスト/期間）に与える影響の定量的な予測	・仕様の構成要素と各設計工程の分類 ・基本データ蓄積のためのインフラ技術確立 ・IP主体の基本データ蓄積	・非機能要件の動的検証 →確認済み要求値による設計 ・新規設計の基本データ蓄積も浸透	・仕様変更の影響度把握 ・仕様記述と工数・コスト見積りとのリンク
	アーキ設計との繋ぎ （モデルドリブンの設計）	アーキテクチャPFへのマッピング （機能：０８～０９）	非機能要件のバジェッティング機能	仕様変更へのリスク分析とSW/CPEへのマッピング CPE: Configurable Processing Engine

図表 3-10　仕様設計における課題解決策

アーキテクチャ設計における課題解決策

設計レベル	技術課題	課題解決策		
		2005～2008	2009～2011	2012～2014
アーキテクチャ設計	最適なHW/SW分割	・機能の並列性分析ツール ・高速、高精度(実行性能)なCPUモデル ・アルゴリズム記述からアーキテクチャ最適化　(SW/HW分割など人手)	・機能のＨＷ／ＳＷ分割指定を容易にできるＧＵＩ ・アルゴリズム記述からのSW/HW　スケジューリング最適化（SW/HW分割などGUIで人手）	・アルゴリズム記述からのアーキテクチャ最適化（SW/HW分割自動） ・電力最適化スケジューリング
	精度の良いＨＷアーキテクチャ設計	・高位モデルの記述標準化/整備/流通 ・高位モデルの性能パラメータの変更（パラメタライズ化） ・高位モデルと実装レベルの対応付け(面積・性能) ・高精度の高位タイミングモデル、ライブラリ（プロセッサモデル、他） ・プロファイリング（面積、性能(粗い)）	・非機能要件仕様は人手設定 ・高位モデルと実装レベルの対応付け(消費電力・DFT・テスト時間) ・高位レベルのテスト(DFT)用ライブラリ ・プロファイリング（面積、性能、電力(粗い)、テストコスト）	・非機能要件仕様も自動最適化 ・高精度の高位電力モデル、ライブラリ ・高位レベルの電力制御用ライブラリ ・プロファイリング（面積、性能、　テストコスト、電力、歩留り）
	容易なＨＷアーキテクチャ探索	・アーキテクチャ記述言語の標準化 ・高位モデルの自動結合	・取捨選択、結合が容易にできるＧＵＩ ・高位モデルライブラリの整備/流通	・最適アーキテクチャの自動選択 ・アーキテクチャライブラリの整備/流通

図表 3-11　アーキテクチャ設計における課題解決策

実装設計における課題解決策

設計レベル	技術課題	課題解決策		
		2005～2008	2009～2011	2012～2014
実装設計	設計規模拡大を凌駕する検証効率の向上	・高位合成の黎明期に対応した、C対RTL等価検証 ・プロパティ検証IPの整備/流通	・高位合成の成熟期に対応した、C対RTL等価検証 ・等価性検証（仕様記述, C, RTL）基礎 ・各設計レベルを統合した検証環境	・等価性検証（仕様記述, C, RTL）実現 ・設計進度に応じた階層的な検証技術
	高位合成技術高度化	・高位合成の黎明期（動作記述に対する人間による判断の追加が必要） ・インターフェース合成	・高位合成の成熟期（目標速度 ・面積の指定のみで、最適な合成を実現） ・高位テスト(DFT)合成 ・動作合成の適用範囲拡大 ・インクリメンタル動作合成（C→RTL）	・レイアウトまで、合成・最適化の範囲を拡大（動作記述＋目標性能・面積から、レイアウトを生成） ・フィジカル高位合成
	高位レベルからの再利用設計の浸透	・アプリケーション分野別プラットフォームの高位拡張（骨組みのみ）プラットホーム設計(高位検証)	・アプリケーション分野別プラットフォームの高位拡張（plug & playの仕組み組み込み）	・仕様レベルでの接続可能性チェック
	既存RTL設計と連携した高位設計	・RTLの高位モデルへの引き上げ（言語変換） ・FPGA, 実チップと高位検証との連携	・RTLの高位モデルへの引き上げ（抽象度引き上げ）	RTLは品質保証済みのIPを利用。

図表 3-12　実装設計における課題解決策

3-4-2-2 Logic / Circuit / Physical Design

〔スコープ〕

- 実装可能な設計記述(現在は RTL)から製造可能な設計品質のデータ(現在は GDS Ⅱ)を生成する工程を対象とする。
- ここで製造可能な設計品質のデータとは、リソグラフィを考慮したマスク設計以降のプロセス制約・機能/特性を実現するための制約(タイミング・消費電力・信頼性・製造性など)において問題の無いデータということである。

〔メッセージ〕

- プロセスの微細化に伴うゲート数の大規模化・電流密度の増大、トランジスタリーク電流の増大による消費電力が増大する要因が急激に増加している。
- 消費電力の増大を抑えるための低消費電力設計が非常に複雑化し、設計工程の後戻り、設計作業の繰り返しが増大している。
- 設計課題として「イタレーションのない低消費電力設計」が最重要と考えた。

〔重要な技術課題と課題解決策〕

- 重要な技術課題
  - イタレーションのない低消費電力設計
- 着目した理由
  - 微細化によりゲート数の増加・大規模化によるダイナミックパワーの増加ばかりでなく、トランジスタリークの増加によるリーク電流の増加が問題となってきている。消費電力の増大を抑えるための低消費電力設計は複雑化しており設計工程のイタレーション(後戻り、設計作業の繰り返し)が多くなってきているので設計工程のイタレーションの低減は非常に重要である。
- 詳細な説明
  - 微細化によるトランジスタのリーク電流上昇及び、微細化・高周波数化による、単位面積あたりの電流密度が上昇し、消費電力が急激に増大する。消費電力は、パッケージの熱対策、携帯システムの動作時間に影響し、SoC・システム性能に大きく影響するため、低消費電力設計は非常に重要である。
  - 低消費電力設計の施策としてマルチ電源、電源遮断、マルチ Vt、基板制御などが行われ設計が複雑化しており、L/C/P 設計工程のイタレーションが増大している。設計工程のイタレーション削減には、上位で複数のパラメータを同時に考慮して最適化し、上位で考慮した各パラメータを製造可能な最終データ(GDS Ⅱ)の検証まで一貫して取り扱う事が重要である。しかしながら、低消費電力削減の施策を RTL 上で記述する方法がなく、適切な論理合成手法が存在しない。
  - 低消費電力設計の施策としての各パラメータを考慮しつつタイミング・シグナルインテグリティ(SI)を考慮し最適な消費電力を実現する複雑な工程でのイタレーションを低減することは非常に重要と考える。

〔その他の技術課題と課題解決策〕

以下に、Logic / Circuit / Physical Design の設計工程を、RTL 設計、RTL→GDS Ⅱ に分け、分析する。

①RTL 設計における技術課題と課題解決策

- L/C/P 工程の最上位である「RTL」の設計において合成制約・テスト回路を含めた品質向上、及び GDS Ⅱを作成するまでの各種見積りの精度向上は、設計工程のイタレーション低減に非常に重要である。以下を技術課題とした。
  - RTL 記述

- 合成制約
- 見積り精度向上
- テスト容易化回路挿入

- 実現時の効果
  - RTL 段階での RTL・合成制約・テスト回路の品質向上、見積り精度向上が図れる。
- 時間軸を伴った解決策の提示
  - 2005～2008 年
    - RTL プロトタイピング精度向上
    - 合成制約チェック標準化
    - 電源系考慮の RTL シミュレーション
  - 2009～2011 年
    - RTL スタイルエラーの自動修正
    - 低故障検出率論理の RTL 自動修正
  - 2012～2014 年
    - 高位合成とリンクした合成制約自動生成
    - 上位言語とリンクした性能見積り

②RTL→GDS Ⅱ における技術課題と課題解決策

- 「RTL～GDS Ⅱ」の設計においては低消費電力設計以外にも、タイミング、信頼性、製造性において問題のない GDS Ⅱ を設計する必要がある。プロセスの微細化により考慮すべき項目が増加しており設計が複雑化し、設計工程のイタレーションが増大している。設計工程のイタレーションを低減する事は重要であり以下の項目における最適化、自動化を技術課題とした。
  - モデリング
  - タイミング設計
  - SI 考慮設計
  - 製造性考慮設計
  - テスト容易化設計
- 実現時の効果
  - 品質、製造性を考慮した論理合成と物理合成の融合、しいては上位言語の動作合成との融合により全体の最適化が図れるとともに、各工程での自動化により設計生産性の向上に寄与する。
- 時間軸を伴った解決策の提示
  - 2005～2008 年
    - 配置考慮の論理合成
    - マルチコーナ/モードのタイミング最適化
  - 2009～2011 年
    - 論理合成技術と物理合成技術の融合
    - 歩留まり考慮の論理合成
    - 信号線のエレクトロマイグレーション(EM)に対応したバスの配置配線自動化
  - 2012～2014 年
    - 上位言語での動作合成から配置合成
    - 故障リペア回路の自動生成

RTL設計の技術課題

	技術課題	説　明
RTL設計	RTL記述	イタレーション削減のため、インプットであるRTLの品質向上が必須。また、消費電力低減のための施策においてマルチ電源/Vtなどの設計技術に使われ、上位のRTLにて物理に対応した記述の導入がRTL-フィジカルの工程の品質向上・生産性向上には不可欠である。
	合成制約	微細化、高速化にともない、タイミング制約が複雑化、困難化し合成制約作成の繰り返し作成作業が多くなってきており、高い精度のタイミング制約を短期に開発する必要がある。また、レイアウト工程で論理が変更されることにより制約変更が必要になるため、制約の等価検証が不可欠である。
	見積もり精度向上	RTLの段階で、性能（動作周波数、消費電力、面積）、歩留まりなどを高精度に見積もることはRTL記述やゲートレベルの回路構造の最適化、量産性向上を促し不可欠である。
	テスト容易化回路挿入	微細化による素子数の増大、故障モードが多様化する。一方市場故障率の低減が要求され、故障検出率向上が必要である。RTLの記述、論理合成により故障検出率が上がらないことがあり、上位のRTLにて故障検出を上げるためのDFT設計を支援する環境の構築が必要である。

図表 3-13　RTL 設計の技術課題

RTL→GDS Ⅱの技術課題

	技術課題	説　明
RTL →GDS Ⅱ	モデリング	微細化に伴い、ばらつきが無視できない。また消費電流力低減のための施策に対応したモデリングが各種見積もり、合成、検証のために不可欠である。
	タイミング設計	タイミング制約を満たす論理合成を実施する際に、遅延計算に大きな影響を与える配線負荷容量はWire Load Modelと呼ばれるモデルを用いて行われている。しかし微細化とともに実際の配置配線を行った後の配線負荷との乖離が大きくなり、配置配線後にタイミングエラーが発生し、大きな設計手戻りが問題となっている。また微細化によるSOCの複雑化、ばらつきの増加により、複雑化するコーナ条件、動作モードを考慮したタイミング最適化技術が不可欠である。
	低消費電力設計	消費電力低減の施策として多電源、電源遮断、マルチVt、基板制御などが行われ設計の各工程が非常に複雑化してきている。SOC設計では、設計工程を通じて各パラメータを一貫して行う必要がある。しかしながらRTLで各パラメータを記述する方法がなく、適切な合成手法が存在しない。また、各パラメータを考慮しつつタイミング、SI制約を考慮し、最適な消費電力を実現する上でのイタレーションを低減する事が不可欠である。
	SI考慮設計	動作モードごとにIR-drop、Xtalkなどが変化するため、すべての条件を満たすための最適化技術が必要であるが、最適化する上で考慮する負荷を減らす技術開発が要求される。また、高周波数化により電源ばかりでなく、シグナルにもEMを考慮する事が不可欠である。
	製造性考慮設計	製造ばらつきにより製造歩留り・性能歩留りの悪化が課題であり、対策が必要である。ばらつくことを前提とした設計技術や、ばらつきを押さえ込む(小さく)設計技術が必要である。
	テスト容易化設計	微細化による故障モードが多様化する一方市場故障率の低減が要求され、故障検出率向上が必要である。縮退故障、遅延故障、ブリッジ故障を効率よく検出するための回路の合成、テストパターン生成のための施策が必要。

図表 3-14　RTL→GDS Ⅱの技術課題

RTL設計の課題解決策

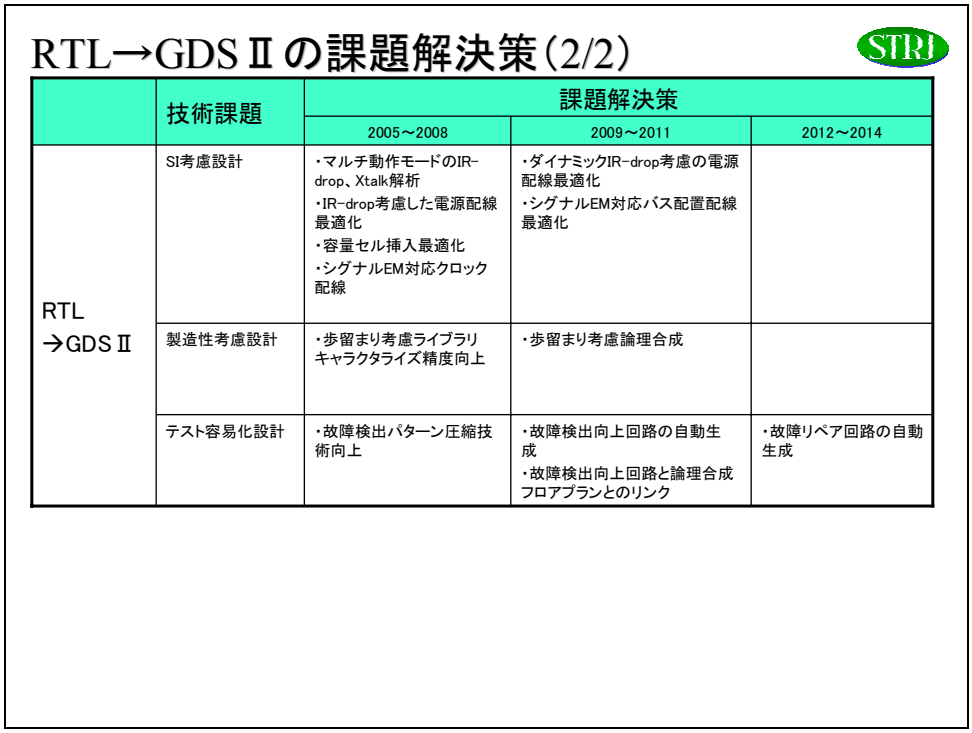
	技術課題	課題解決策		
		2005～2008	2009～2011	2012～2014
RTL設計	RTL記述	・RTL電源記述標準化 ・RTLスタイルチェック標準化	・可変電源認識・記述標準化 ・RTLスタイルエラーの自動修正技術実用化	・基板制御記述標準化
	合成制約	・合成制約チェック標準化	・制約条件等価検証	・高位合成とのリンクによる合成制約の自動生成
	見積もり精度向上	・RTLプロトタイピングの精度向上 ・CG,CTS,DFT考慮の性能（動作周波数、面積、消費電力）の見積もり ・電源系統考慮RTLシミュレーション	・マルチ電源考慮の性能見積り ・基板制御ブロック割り出し機能実用化 ・IR-drop対応の電源ラインを考慮した面積見積もり	・上位言語とリンクした性能見積り ・基板制御考慮の消費電力見積り ・冗長、リペア回路を考慮した面積見積もり ・RTLにおける考慮すべき見積もり項目のターゲット値を考慮したRTL記述改訂
	テスト容易化回路挿入	・低検出率の論理抽出の精度向上	・高効率テストポイント挿入技術の向上 ・低故障検出率論理のRTL自動修正	・上位言語とのリンクによるRTL時の低検出率回路回避

図表 3-15　RTL 設計の課題解決策

RTL→GDSⅡの課題解決策（1/2）

	技術課題	課題解決策		
		2005～2008	2009～2011	2012～2014
RTL →GDSⅡ	モデリング	・電源系統の電源記述仕様の開発 ・マルチ電源/Vt考慮のタイミングモデル ・状態依存、ばらつきを考慮の消費電力ライブラリ生成	・電源記述仕様の標準化 ・インスタンス毎の電源認識考慮のモデル ・可変電源、基板制御、及びばらつき考慮のタイミングモデル適用 ・可変電源、基板制御考慮の消費電力ライブラリ生成	・小振幅クロック考慮のタイミング、消費電力モデル
	タイミング設計	・配置考慮の論理合成 ・マルチコーナ/モードのタイミング最適化 ・論理合成時の想定配置と実配置とのリンク強化 ・クロックメッシュ技術実用化	・論理合成技術と物理合成技術の融合 ・ダイナミックIR-drop考慮のSTA ・複数電圧間におけるタイミング最適化・検証技術の実用化 ・可変電源/基板制御におけるクロック最適化	・上流言語（C等）での動作合成からの配置合成技術開発 ・小振幅クロック、非同期設計考慮のタイミング最適化
	低消費電力設計	・電源系統考慮　論理合成ツールの実用化 ・マルチ電源/Vt考慮の物理合成 ・アクティビティプロパゲーションとリンクした消費電力解析	・基板制御考慮の物理合成 ・可変電源、基板制御を考慮した消費電力解析	・上流言語（C等）からのマルチ電源動作合成 ・小振幅クロック考慮の論理合成 ・基板制御考慮の論理合成

図表 3-16　RTL→GDSⅡの課題解決策（1/2）



	技術課題	課題解決策		
		2005～2008	2009～2011	2012～2014
RTL →GDSⅡ	SI考慮設計	・マルチ動作モードのIR-drop、Xtalk解析 ・IR-drop考慮した電源配線最適化 ・容量セル挿入最適化 ・シグナルEM対応クロック配線	・ダイナミックIR-drop考慮の電源配線最適化 ・シグナルEM対応バス配置配線最適化	
	製造性考慮設計	・歩留まり考慮ライブラリキャラクタライズ精度向上	・歩留まり考慮論理合成	
	テスト容易化設計	・故障検出パターン圧縮技術向上	・故障検出向上回路の自動生成 ・故障検出向上回路と論理合成フロアプランとのリンク	・故障リペア回路の自動生成

図表 3-17 RTL→GDSⅡの課題解決策(2/2)

3-4-2-3 Design Verification

〔スコープ〕

- アーキテクチャ(HW/SW)が決定した後の設計記述の機能と性能の検証を対象とする。
- 検証対象としては、基本は RTL 記述の設計データとし、高位レベルは対象としない。
- SoC の高機能化に伴い、プロセッサー搭載が普通になり、検証のためにはプロセッサーを動作させるためのソフトウェア(検証ソフトウェア)も必須であるが、今回は敢えてソフトウェアとして明確に区分しない。(検証モデル、テストベンチに含める)
- 検証に際して、完成した仕様書(Executable なものを含む)を前提とする。それに基づき検証仕様書を作成し、検証戦略を立案した上で、「検証項目抽出から検証モデル・テストベンチの作成、検証実行、解析・デバッグまで」を範囲とする。

〔メッセージ〕

- 開発コストの増大を抑えるには、設計工数の 70%程度を占める検証工程の効率化が重要となっている。
- リスピン(SoC の作り直し)をなくす検証品質の向上を、効率化とともに実現するには、適切な検証戦略の作成と検証実行の自動化が課題であるが、今回は、以下の分類で詳細検討をした。
  - 検証戦略:検証項目の抽出、検証モデル/テストベンチの作成
  - 検証実行:実行スピード、デバッグ効率化など
  - その他
- その中で「検証項目の抽出・検証モデル/テストベンチの作成」の効率化・完成度向上が最重要と考えた。

〔重要な技術課題と課題解決策〕

- 重要な技術課題
  - 検証項目の抽出と検証モデル・テストベンチの生成
- 着目した理由

- SoC の大規模化・複雑化に伴い、開発工程の 70%程度を占める機能検証の効率化が開発TAT短縮に不可欠である。また、リスピン(SoC の作り直し)を無くし、増大するマスクコストやリスピンに伴う開発費用を抑えるためには検証完成度(検証品質)の向上も重要である。
- 検証完成度向上のためには、いかに網羅性の高い検証項目を抽出できるかがキーである。
- 機能検証の効率化(TAT短縮)のためには、検証の実行時間の短縮も重要であるが、実行のための検証モデルやテストベンチの作成工数の削減がより重要であると考える。

■ 詳細な説明

- 網羅性の高い検証項目の抽出に向けての課題
  - 検証する内容が、「機能、ブロック・インタフェース(タイミング)、性能(スループット/レイテンシ)、消費電力」と多種にわたる。
  - 機能検証項目の抽出の完全性をどうようにして保証するかという明確な手法・指標がない。
  - ブロックレベルでは、個々のブロックの単体動作の検証項目に加えて、システムとして動作を保証する検証項目をいかに早期に抽出できるかが重要である。
  - システムレベルでは、現状のボトムアップの設計手法の中で、全体システム(アーキテクチャ)の機能、性能を保証する検証項目をいかに抽出するかも課題である。 など
- 検証モデルやテストベンチの作成工数の削減に向けての課題
  - 目的や用途毎に多種のモデル(抽象度毎、HW 用/SW用、機能用/性能用など)を準備する必要があり、また、SoC 周辺モデル(テストベンチ、ランダムパターン生成、ストリーム生成)の準備も必要。
  - アサーションなどの検証記述言語(HVL)が各種存在する上に、検証者間でも表現方法/レベルに差がある。(標準の検証ガイドラインがない) ※HVL : Hardware Verification Language
  - 標準IP(バス、インタフェースコアなど)に対する検証IPが不足。検証 IP の整備(品質認証・保証を含む)も不十分。 ※検証 IP(Verification IP) : IP を検証するための検証環境で検証シナリオ、検証モデルなどを含んだテストベンチ。

■ 実現時の効果

- 要求仕様から検証項目が自動的に抽出できるようになり、検証項目漏れによるバグの撲滅が図れるようになる。
- 検証手法の標準化やそれに基づいた完成度の高い品質保証済みの検証IPが整備されていき、検証工数の大幅削減が図れる。
- 高位レベルでのシステム検証により、システム全体の機能・性能が早期に正確に検証できることになり、開発効率の大幅向上が図れる。

■ 時間軸を伴った解決策の提示

- 検証項目の抽出に関しては、
  - 2005～2008 年 : ファンクションカバレッジをベースにした網羅性の算出技術でカバー(ブロックレベル)
  - 2009～2011 年 : ファンクションカバレッジをベースにした網羅性の算出技術でカバー(サブシステムレベル)
  - 2012～2014 年 : 機能仕様をフォーマルな形式(UMLなど)で表現し、要求仕様からそれに対応した検証項目を抽出する技術ができる
- 検証モデル・テストベンチに関しては

- 2005～2008 年 ： 検証手法の標準化（ガイドライン）が整備され、それに基づいた完成度の高い検証IPが標準IPを手始めに整備されていく。
- 2009～2011 年 ： 完成度の高い品質保証済みの検証IPが充実していき、検証工数の削減が図れる。
- 2012～2014 年 ： システムレベルの検証は高位(SLD)で実施し、既存ブロックは整備された検証IPを再利用し、新規ブロックは高位合成＋高位レベルとの等価性検証で、検証実行を含めた効率化が図られる。

〔その他の技術課題と課題解決策〕

上記で挙げた検証戦略・手法を除き、残された項目を、検証実行とアナログ設計等その他に分けて分析する。

①検証実行時における技術課題と課題解決策

- 技術課題の説明：SoC の大規模化・複雑化に伴い
  - 検証の実行速度の大幅向上が必要である。
  - 検証結果の確認が目視によるチェックが多いことや、擬似エラーと真正エラーの切り分けが難しい、エラー（不具合）個所の特定が難しいなどで、解析工数が増大している。
  - ボトムアップの設計の中でトップ回路の検証環境の組上げ工数も増大している。
  - クロックドメイン検証などの非同期回路の検証にも有効な手法がなく、チェック抜けが発生。
- 実現時の効果
  - 検証実行時の速度を向上し、かつ、デバッグ・解析の効率化が図れることにより、検証の効率化が図れる。同時に、検証完成度の向上にも寄与する。
- 時間軸を伴った解決策の提示
  - 2005～2008 年
    - 高速で安価な FPGA ベースのボードプロトタイピング技術
    - アサーション検証やフォーマル検証（サブブロック）での検証効率化
  - 2009～2011 年
    - 高位レベルとの等価性検証技術の本格活用
    - フォーマル検証技術の適用範囲の拡大
    - RTL 構造検証技術(SuperLint)の高度化で、非同期検証などの擬似エラーを撲滅
  - 2012～2014 年
    - システム検証は高位レベル設計(SLD)で完全実施することにより高速化し、RTLは高位レベルとの等価性検証で保証
    - 品質保証済みの検証IPの利用で検証を効率化し、IP間接続はフォーマル検証で自動化

②アナログ設計等その他の技術課題と課題解決策

- 技術課題の説明
  - SoC の大規模化の中で、アナログ回路の統合化も進展していき、デジアナ混在検証もますます重要になってくる。
- 実現時の効果
  - 高速・高精度なデジアナ混在検証技術により、検証の効率化と検証完成度の向上が図れる。
- 時間軸を伴った解決策の提示
  - 2005～2008 年
    - アナログ機能記述を用いたデジアナ混在シミュレーション技術

- 2009～2011 年
  - デジアナ混在シミュレーションの高速化技術
- 2012～2014 年
  - 高位レベルでデジアナ混在シミュレーション

- 技術以外の課題:SoC の大規模化に伴い、多拠点で多部署のメンバーが参画した検証業務となり、現状では検証手法などが不統一といった課題がある。また、全体の検証工数の見積もりも困難で、設計と検証間の時間マネージメントにも課題が多い。従って、技術面だけではなく、プロジェクトマネージメント面での取り組みも重要となってくる。

検証戦略・手法における技術課題

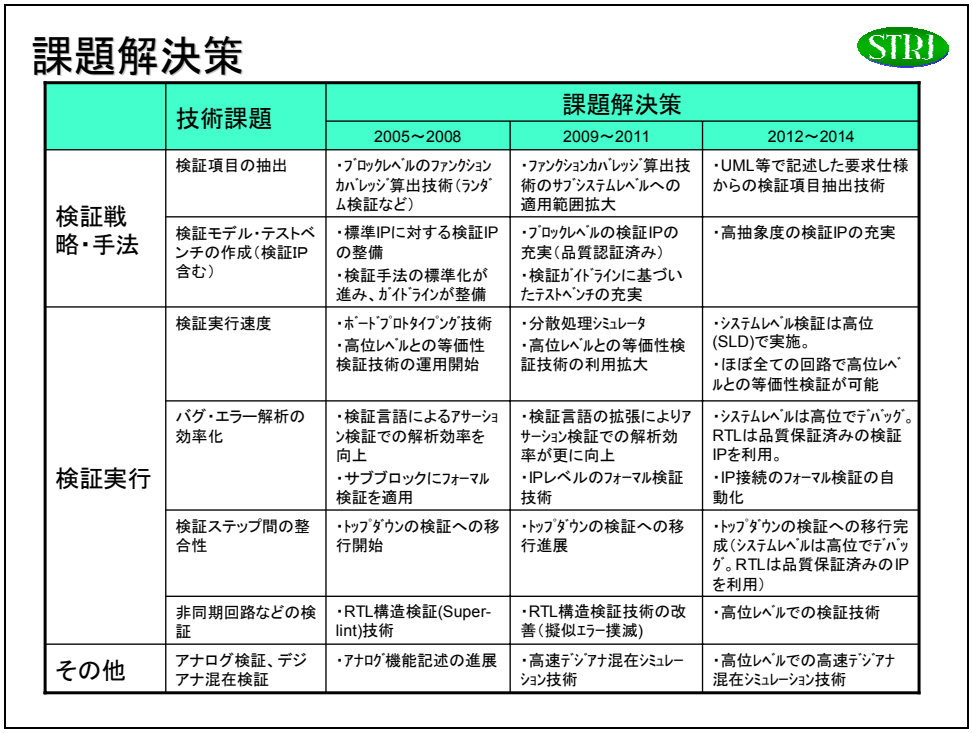
	技術課題	説　明
検証戦略・手法	検証項目の抽出	検証の対象は設計回路(デザイン)であり、検証する内容は、「機能、ブロック・インタフェース(タイミング)、性能(スループット／レイテンシ)、消費電力」と多種にわたる。また、性能検証としては「キャッシュサイズ、最高動作周波数、PE数、ホストCPU選択、メモリサイズ、バス(タイプ選択＆構成)」を抑える必要がある。そのような状況の中、機能検証項目の抽出の完全性をどのようにして保証(必要十分)するか、そして、機能検証の網羅性(定量化)をいかに反映した機能検証リストを作成するかが課題である。 また、ブロックレベルでは、個々のブロックの単体動作の検証項目に加えて、システムとして 動作保証の検証項目をいかに早期に抽出できるかという課題があり、システムレベルでは、現状のボトムアップの設計手法の中で、全体システム(アーキ) の機能、性能を保証する検証項目をどうやって抽出するかという課題がある。
	検証モデル・テストベンチの作成(検証IP含む)	検証モデルとして、目的や用途毎に多種のモデル(抽象度毎、HW用/SW用、機能用/性能用など)を準備する必要があり、また、SoC周辺モデル(テストベンチ、ランダムパターン生成、ストリーム生成)の準備も必要で、その作成に多大な工数／期間がかかる。 アサーションなどの検証記述言語(HVL)が各種存在する上に、標準の検証ガイドラインがないため検証者間でも表現方法/レベルに差がある。 また、標準IP(バス、IFコアなど)に対する検証IPが不足しており、かつ、検証IPの整備(品質認証・保証を含む)も不十分である。

図表 3-18　検証戦略・手法における技術課題

検証実行における技術課題

	技術課題	説　明
検証実行	検証実行速度	SOCの大規模化やテストベンチの長大化でシミュレータの高速化が必要である。 高速化の手段としてハードウェアエミュレータが存在するが、高価でかつスピードも不十分なため、安価で高速なボードプロトタイピング技術の本格化も必要。 また、実用的な高位レベルとの等価性検証技術も現状ない。
	バグ・エラー解析の効率化	シミュレーション結果の確認方法として、依然、目視によるチェックが多く、自動化が困難。また構造検証(HDL-lint/Super-lint)などでは、擬似エラーが多く、真のエラーを見つけることが難しい。 SOCが大規模・複雑化して、不一致が出た時の問題が回路なのかテストベンチ(ソフトなどを含む)なのかといった判断やエラー原因個所の特定が難しい。
	検証ステップ(レベル)間の整合性	トップ回路の仕様決定が遅れることもあり、現状のボトムアップ(ブロック単体→トップ)の設計フローの中で、トップレベルの検証環境の構築が後回しになり、かつ、組上げ工数も増大している。また、ボトムアップの設計手法で、ブロックの設計時に上位で使われる状況を理解せずに設計している場合があり、そのため、ブロック間IF検証で問題が発生することがある。
	非同期回路などの検証	クロックドメイン検証などで構造検証(HDL-lint/Super-lint)以外に適当な手法がないため抜けが発生。 また、非同期割り込み時(ex.モード切替)の動作検証が困難。例えば、キー入力やスイッチ切り替えなどの想定外の処理シーケンスが発生した場合の検証。
その他	アナログ検証、デジアナ混在検証	・アナログとのIFで十分な検証手法がなく、デジアナ混在検証が動的・静的にも困難。
	物理的影響(ノイズ、ソフトエラー)への対応	・プロセルの微細化に伴い、物理的影響による課題が発生する。

図表 3-19　検証実行における技術課題



	技術課題	課題解決策		
		2005～2008	2009～2011	2012～2014
検証戦略・手法	検証項目の抽出	・ﾌﾞﾛｯｸﾚﾍﾞﾙのﾌｧﾝｸｼｮﾝｶﾊﾞﾚｯｼﾞ算出技術（ﾗﾝﾀﾞﾑ検証など）	・ﾌｧﾝｸｼｮﾝｶﾊﾞﾚｯｼﾞ算出技術のｻﾌﾞｼｽﾃﾑﾚﾍﾞﾙへの適用範囲拡大	・UML等で記述した要求仕様からの検証項目抽出技術
	検証モデル・テストベンチの作成（検証IP含む）	・標準IPに対する検証IPの整備 ・検証手法の標準化が進み、ｶﾞｲﾄﾞﾗｲﾝが整備	・ﾌﾞﾛｯｸﾚﾍﾞﾙの検証IPの充実（品質認証済み） ・検証ｶﾞｲﾄﾞﾗｲﾝに基づいたﾃｽﾄﾍﾞﾝﾁの充実	・高抽象度の検証IPの充実
検証実行	検証実行速度	・ﾎﾞｰﾄﾞﾌﾟﾛﾄﾀｲﾌﾟﾝｸﾞ技術 ・高位ﾚﾍﾞﾙとの等価性検証技術の運用開始	・分散処理ｼﾐｭﾚｰﾀ ・高位ﾚﾍﾞﾙとの等価性検証技術の利用拡大	・ｼｽﾃﾑﾚﾍﾞﾙ検証は高位(SLD)で実施。 ・ほぼ全ての回路で高位ﾚﾍﾞﾙとの等価性検証が可能
	バグ・エラー解析の効率化	・検証言語によるｱｻｰｼｮﾝ検証での解析効率を向上 ・サブブロックにﾌｫｰﾏﾙ検証を適用	・検証言語の拡張によりｱｻｰｼｮﾝ検証での解析効率が更に向上 ・IPレベルのﾌｫｰﾏﾙ検証技術	・ｼｽﾃﾑﾚﾍﾞﾙは高位でﾃﾞﾊﾞｯｸﾞ。RTLは品質保証済みの検証IPを利用。 ・IP接続のﾌｫｰﾏﾙ検証の自動化
	検証ステップ間の整合性	・ﾄｯﾌﾟﾀﾞｳﾝの検証への移行開始	・ﾄｯﾌﾟﾀﾞｳﾝの検証への移行進展	・ﾄｯﾌﾟﾀﾞｳﾝの検証への移行完成（ｼｽﾃﾑﾚﾍﾞﾙは高位でﾃﾞﾊﾞｯｸﾞ。RTLは品質保証済みのIPを利用）
	非同期回路などの検証	・RTL構造検証(Super-lint)技術	・RTL構造検証技術の改善（擬似ｴﾗｰ撲滅）	・高位ﾚﾍﾞﾙでの検証技術
その他	アナログ検証、デジアナ混在検証	・ｱﾅﾛｸﾞ機能記述の進展	・高速ﾃﾞｼﾞｱﾅ混在ｼﾐｭﾚｰｼｮﾝ技術	・高位ﾚﾍﾞﾙでの高速ﾃﾞｼﾞｱﾅ混在ｼﾐｭﾚｰｼｮﾝ技術

図表 3-20　Design Verification の課題解決策

3-4-2-4 Design For Manufacturability

〔スコープ〕物理現象モデルに基づき、製造可能性と歩留まりを検証し最適化する設計技術

〔メッセージ〕半導体プロセスの微細化に伴い、マスクコストとデータ量の爆発、そしてリソグラフィ技術の限界が SoC の製造容易性に重大な設計課題を引き起こしている。それは　従来の一義的なデザインルールやコーナー条件によるデバイスモデルだけでは目標歩留まりの達成が困難になってきたことに起因している。この問題を解決する為に、設計が微細化に伴う製造困難性を改善することで歩留まりを改善する技術を検討する。

〔重要な技術課題と課題解決策〕

①設計目標を製品固有の設計属性とプロセス固有の故障可能性の両者の関数として表現すること

＞製造が提示する「物理現象モデル」に基づき、「面積、スピード、消費電力、歩留まり」が最適になるような関数として表す。

②リソグラフィ技術の限界（波長）、ドーピング濃度のバラツキ、それらに伴う閾値電圧のバラツキ

＞ITRS2005　Design 章　表18では　2009 年には Vth のバラツキが　40%以上、回路動作のバラツキが 50%、消費電力のバラツキが60%近くになると示されている

③電源電圧のバラツキ

＞電源電圧変動に対する設計上の工夫が必要。

3-5 今後の計画

国内活動としては Design For Manufacturability における技術課題と課題解決策の検討を引き続きすすめる。国際活動としては、ITRS2006 のマイナーupdateに向けて、ITRS2005 の各表に対するフィードバック等をおこなう。