SOC設計技術ロードマップ

－論理検証と物理設計の生産性向上の課題と解決策－

2008年　3月　6日

JEITA半導体技術ロードマップ専門委員会（STRJ）
設計ワーキンググループ （WG1）

設計WGメンバ

隅谷 三喜夫(リーダ)
松下電器産業
松崎　正己(サブリーダ)
富士通
樋渡 有(国際担当)
東芝
柏木 治久(国際担当)
半導体理工学研究センター
富重　了一(幹事)
ﾄｯﾊﾟﾝ･ﾃｸﾆｶﾙ･ﾃﾞｻﾞｲﾝｾﾝﾀｰ
中山　勝敏
ルネサステクノロジ
藤波　義忠
NECエレクトロニクス
山本　一郎
沖電気工業
向野　守
三洋電機

豊田　忠雄
シャープ
唐澤　純一
セイコーエプソン
柿本　勝
ソニー
浅井　健史
ローム
塩月 八宏
半導体理工学研究センター
小野　信任
ジーダット･イノベーション
今井　正治
大阪大学

計　16名

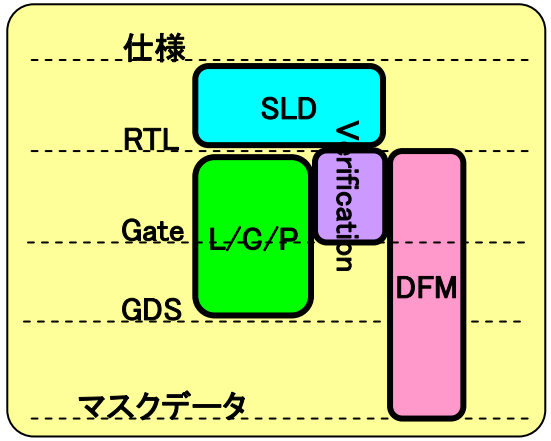
用語集

- RTL：Register Transfer Levelの略。回路をフリップフロップ＋組み合わせ論理回路で表現したレベルのこと. 現在の論理回路設計はおもにこのレベルの記述を使用する.
- SLD：System Level Designの略
- L/C/P：Logic/Circuit/Physical Designの略
- DFM：Design For Manufacturabilityの略で、歩留まり等の製造性考慮設計のこと。
- DFT：Design For Testの略で、テスト容易化設計のこと。
- SOC：System On Chipの略
- EDA：Electronic Design Automationの略
- IP：Intellectual Propertyの略で、半導体の設計データやシミュレーションモデルなどの設計資産のこと。シミュレーションモデルを検証IPともいう。
- プロトタイピング：FPGA を用いてSOCのプロトタイプを作成することで検証時間を短縮する手法。
- アサーション：論理的に成立すべき関係や条件を記述したもの。それをチェッカとして論理検証で活用。
- フォーマル（形式的）検証：設計と仕様を数学的モデルで表現し数学的推論により正しさを検証する静的検証手法。等価性検証とプロパティ検証がある。
- 高位合成・動作合成：C/C++などのアルゴリズム記述から、RTLを自動的に合成すること。
- DR：Design Reviewの略
- CTS：Clock Tree Synthesisの略
- ATPG：Automatic Test Pattern Generatorの略で、LSIテスタで使うパターンを自動生成するEDAツール
- DVFS：Dynamic Voltage and Frequency Scalingの略で、動作中のLSIの電源電圧やクロック周波数を最適な値に制御して消費電力低減を実現する技術
- EM：Electro Migrationの略で電流が過度に流れることで配線に使う金属の原子配列が乱れ断線する現象。
- CMP：Chemical Mechanical Polishingの略で、化学機械研磨のこと

設計WGのミッション

- ◆国際活動
  - ITRSのSystem Drivers章とDesign章を設計TFと分担して担当
    - System Drivers章
      - ITRSのすべての技術分野をドライブするLSI商品を定義
    - Design章
      - 設計技術に対する将来課題と課題解決策の提示
- ◆国内活動
  - SOC構造・規模を時間軸で定量化し、ロードマップ検討の基礎として提示
  - 設計技術課題を時間軸で定量評価し解決策を提案（設計技術ロードマップ）

設計WGのスコープ



System Level Design

- 仕様から最適なHW/SWに分割し、HWに関してはRTL記述を生成する。

Logic / Circuit / Physical Design

- RTL記述から製造可能な設計品質のレイアウトデータ（GDSⅡ）を生成する。

Design Verification

- RTL記述の機能と性能を仕様に基づき検証する。

Design For Manufacturability

- プロセスの物理現象モデルに基づき、製造可能性/歩留まりを検証/最適化する。

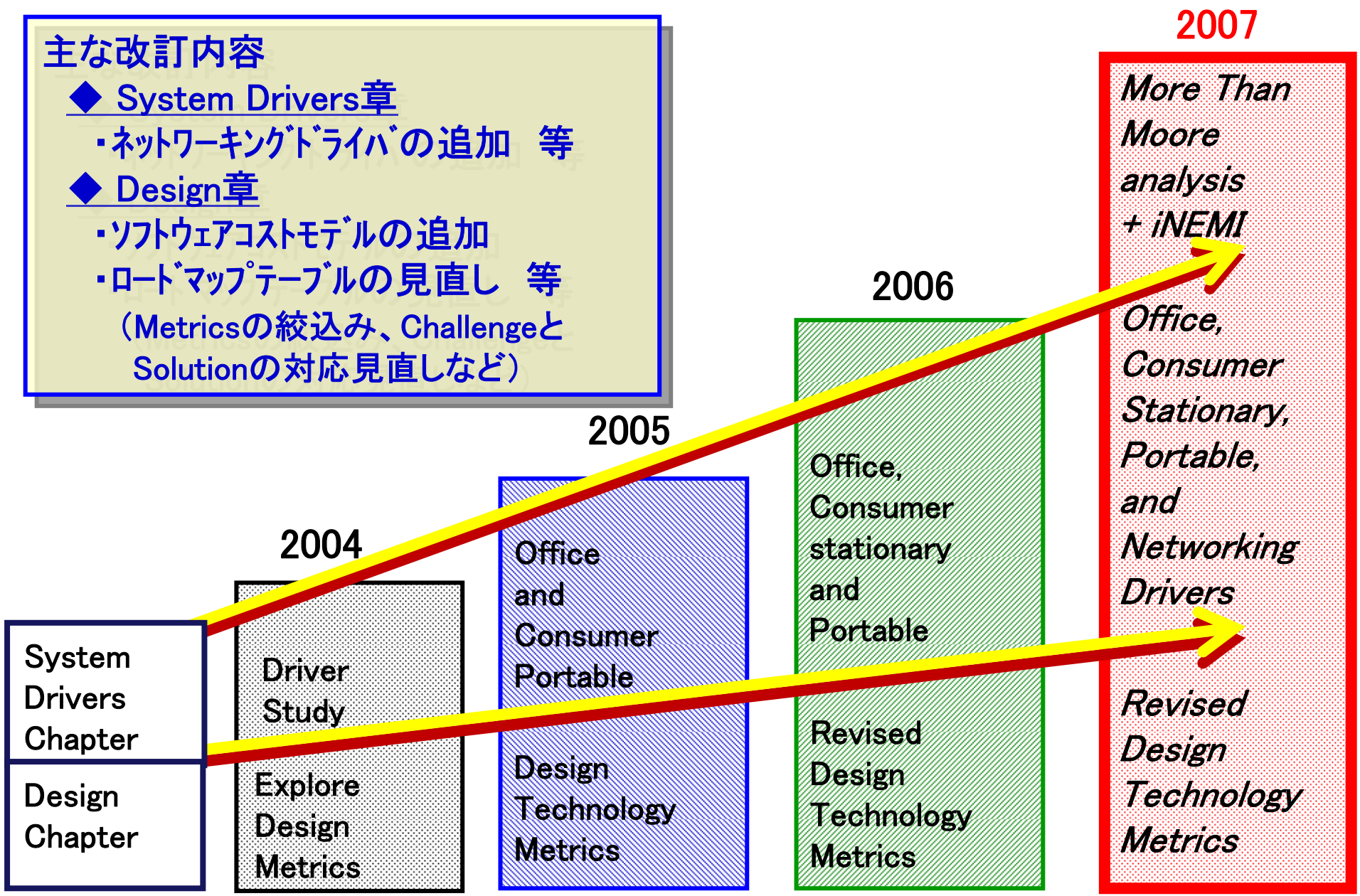
設計WGの活動内容（2004年度以降）

	国際活動（ITRSへの主な貢献）	国内活動
2004年度		**SOC設計生産性ロードマップの策定** ・ロードマップ策定のためのSOCモデルの設定 ・課題抽出と解決策の検討（システム、アーキ設計分野）
2005年度	System Drivers章 Consumer Portable SOCを提案、採択 Design章 課題項目の確認と修正案提示	**SOC設計技術ロードマップの見直し** SLD,L/C/P,Verification,DFMの4つのSWGによる ・メッセージと重要な技術課題の明示 ・課題抽出と解決策の提案
2006年度	System Drivers章 Consumer Stationary SOCを提案、採択 Design章 SLDとVerificationの課題の項目値の確認と修正案提示	**設計遅れ要因変化の分析と提言** 設計遅れ要因変化（3年間）の分析を行い課題解決に向けて提言 **DFM-SWGの深耕** SOC設計におけるプロセスばらつきの影響を定量化するための調査活動(パス遅延ばらつき評価モデルの構築)
2007年度	System Drivers章 Consumer Portable &Stationary SOCの数値の見直し Design章 LCPとDFMの課題の項目値の確認と修正案提示	**SOC設計技術ロードマップの詳細化/定量化** 論理検証と物理設計の2つのSWGによる「設計生産性向上」に向けた課題抽出と解決策のロードマップの詳細化/定量化

ITRS2007 Overview

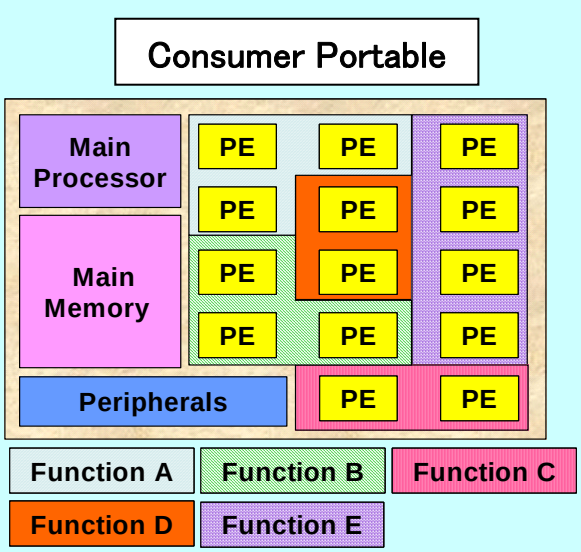
主な改訂内容

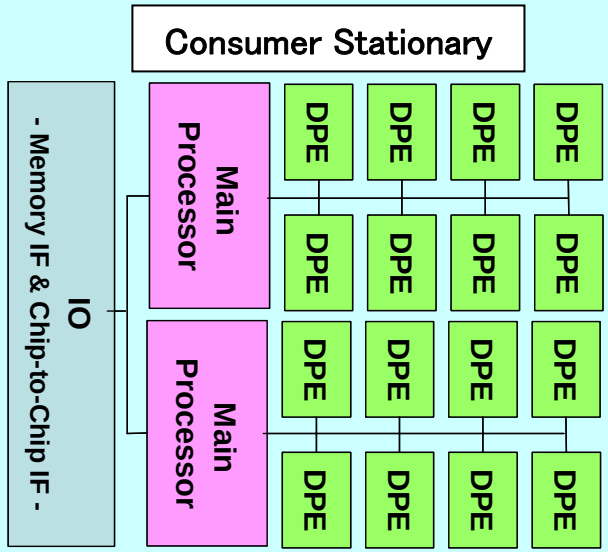
- ◆ System Drivers章
  - ・ﾈｯﾄﾜｰｷﾝｸﾞﾄﾞﾗｲﾊﾞの追加　等
- ◆ Design章
  - ・ｿﾌﾄｳｪｱｺｽﾄﾓﾃﾞﾙの追加
  - ・ﾛｰﾄﾞﾏｯﾌﾟﾃｰﾌﾞﾙの見直し　等
    （Metricsの絞込み、Challengeと Solutionの対応見直しなど）

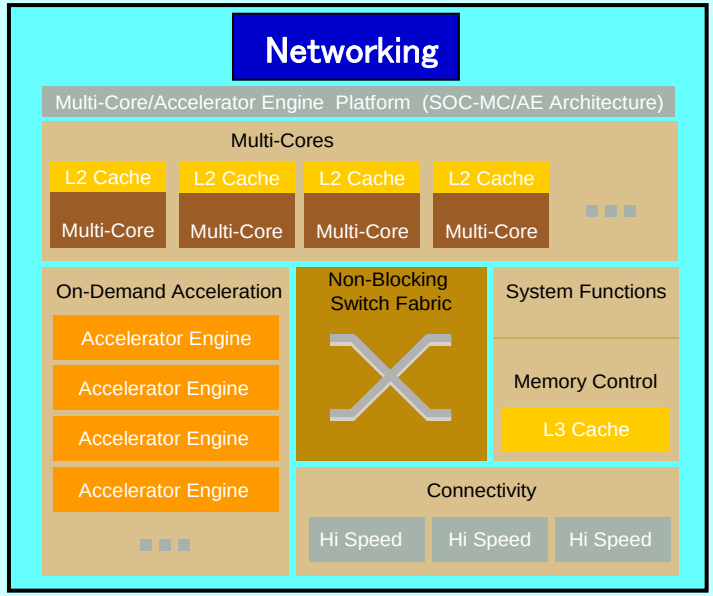


ITRS2007改訂内容の抜粋

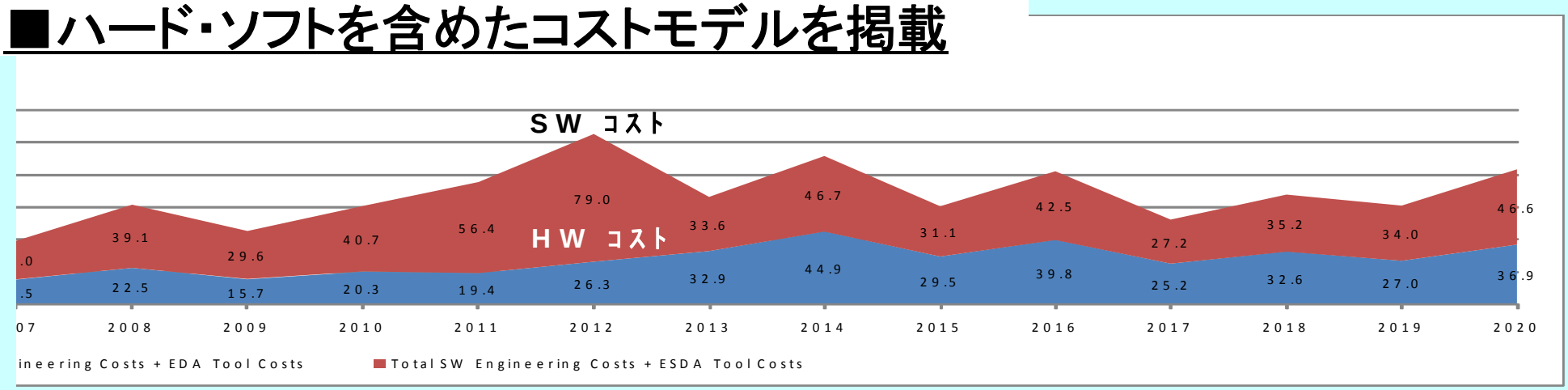
■新たに「Networking System Driver」を追加







■ハード・ソフトを含めたコストモデルを掲載

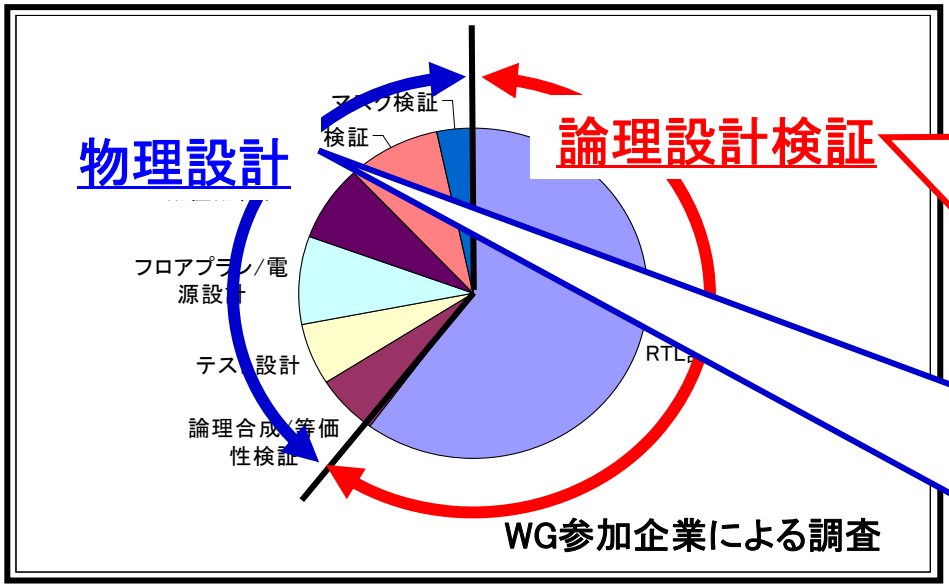


2007年度国内活動

SOC設計技術ロードマップの詳細化・定量化

SOC設計を「論理設計検証」と「物理設計」工程に分割し、
2つのSWG活動で、設計生産性の観点から詳細化、定量化を実施

SOC設計工程別の工数分布



設計生産性＝ システム複雑度 / 設計工数

・システム複雑度は回路規模で定量化済み
・重要課題を抽出し技術要求/解決策を詳細化
・解決策実現による生産性向上率を定量化

設計生産性＝ システム複雑度 ＋シリコン複雑度 / 設計工数

・シリコン複雑度を導入し、複雑度を詳細化
　複雑度要因を定義し、ITRS指標と紐付け
　設計生産性への影響を定量化
・生産性向上の技術要求と解決策を提示

2007年度国内活動の結果

論理検証SWG活動

- 重要課題を抽出し、中期的な技術要求と解決策を詳細検討
- 検証戦略ガイド整備、仕様記述標準化、検証EDA技術向上が必要
- 上記解決策の2010年時点の設計生産性向上率を定量化

  2.3倍の生産性向上の見込み ＞ ITRS要求（1.96）

物理設計SWG活動

- シリコン複雑度の影響を加えた設計生産性要求を定量化

  Near Term（2015年）で1.7倍（従来比）

  Long Term（2022年）で2.9倍（従来比）
- 設計生産性向上のための課題解決策を提示
- 高精度な見積もり/解析技術と各種最適化の自動化技術が必要

論理検証SWG

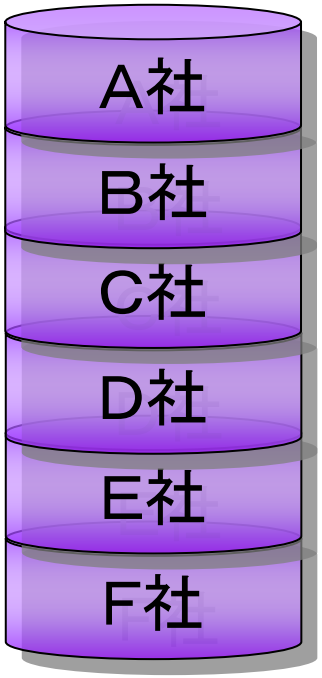
◆目標
- ー論理検証の課題を分析し、実現すべき技術要求と解決案を提示
- ー2010年の技術要求が実現された場合の生産性向上率を定量化

◆スコープ
- ー HW/SW分割後の機能仕様からRTLまでのハードウェア機能検証
- ー 高位レベルの性能検証も検討

◆検討手順
- ー 2007年時点の機能（論理）検証の課題を洗い出し、現状を整理
- ー 重要課題について2010年に実現すべき技術要求と解決案を提示
- ー 2011年以降で実現されるべき技術要求を提示
- ー ITRS設計生産性要求ロードマップに対して、2010年の技術要求が実現されたときの生産性向上率を定量化

論理検証の課題の抽出・分類

各社から集めた75項目の課題 ➡ 15項目の課題を抽出 ➡ 中長期での重要課題



明確で誤解がない仕様の作成
検証対象の状況に応じた最適な検証戦略の策定
最適な検証のマネジメント
網羅的な検証項目の抽出と検証順序の最適化、および仕様変更への対応
IPモデルの整備と検証品質
シミュレーションの高速化
クロックドメイン検証
デバッグの効率化
ランダム検証
設計レベル間検証
機能カバレッジの統一
設計初期段階での性能検証
高精度時間モデルが必要となる回路の検証
機能検証スキルを持つ人材／専門家の育成
検証事例／トラブル情報の共有

対応中の課題、2次的な課題
および機能検証の範囲外の項目

明確で誤解がない仕様の作成
検証対象の状況に応じた最適な検証戦略の策定
最適な検証のマネジメント
網羅的な検証項目の抽出と検証順序の最適化、および仕様変更への対応
IPモデルの整備と検証品質
シミュレーションの高速化
デバッグの効率化
設計レベル間検証
設計初期段階での性能検証
機能検証スキルを持つ人材／専門家の育成

3つの設計レベルに分類

戦略策定　仕様・設計　検証実行

論理検証の課題と技術要求（抜粋）

■戦略策定

重要課題	課題は何か	今どうしているか	中期に解決すべき要求と解決案	将来の要求
検証対象の状況に応じた最適な検証戦略の策定 **最適な検証戦略**	検[illegible]計品質や適用可能な設計手法に応じて、最適な検証戦略を立案することが難しい。さらに、設計品質や検証レベルの異なったブロックが混在する場合、個々に **設計対象によって求められる品質などが異なり、最適な戦略策定が困難**	[illegible]証指針を示して、それに基づいてトップはトップ検証担当がサブブロックは各サブブロック・リーダがそれまでの経験に基づいて、検証対象の特性を考慮した検証[illegible]ビューの中で確認して、その防止策や効率的な検証方法を検証手法や環境に取り入れている。 **過去の経験を元に立案・DRで確認**	【技[illegible]個々の経験やスキルのばらつきによらず高品質な検証環境が構築できる仕組みの実用化 【解決策案】 標準となる検証手法が確立する(種々[illegible]な検[illegible] **標準的な検証手法のマニュアル化**	検証ターゲットに応じた検証手法が用意され、検証環境が容易に構築できる **検証ターゲットに最適な検証手法が策定できる**
最適な検証のマネジメント **マネジメント**	利用可能なリソースを有効活用し、検証過程での種々の問題解決して、リスク管理を行いながら最適なQCDのバランスの取れたマネジメントを実行することが難しく、プロジェクトリーダとプロジェクトメンバのスキルに依存している。 **与えられたリソースに対してリスク管理を行いつつ期限内に求められる品質を達成する方法が一義的に決められない**	・過去の事例から問題点、対処を考え、検証戦略を改良していく。 ・プロジェクトによっては、独立した検証技術チームが検証を実施したり、コンサルティングを行い検証戦略や検証環境構築を支援することもある。 **過去の経験を元にプロジェクト管理**	【技術要求】 ・プロジェクトリーダ/メンバが目指すべきベストプラクティス(やるべき項目、手順、リスク管理など)を明確するためのガイドラインの策定。 ・プラットフォームベースの検証では、プロジェクトリーダ/メンバのスキルに依存せず最適な検証環境が構築[illegible]。 【[illegible] 様[illegible]レートとして用意して、各項目の記載内容の詳細定義と事例を示したガイドを作成する。 **ベストプラクティスに基づくガイドラインの策定**	【技術要求】 プロジェクトリーダ/メンバのスキルに依存せず、最適な検証のマネジメントができる。 【解決策】 マネジメントガイドに事例等の知識ベースを追[illegible] **様々な状況に応じたマネジメント・コンサルティング・システム**

国内活動

論理検証の課題と技術要求(抜粋)

■仕様・設計

重要課題	課題説明	2007年時点での対応策(取り組み)	2010年に実現すべき技術要求／解決策の案	2011年以降の技術要求／解決策の案
網羅的な検証項目の抽出と検証順序の最適化、および仕様変更への対応	仕様から検証項目を網羅的に抽出する手法や適切な検証順序を決めることが定式化されておらず、仕様と検証項目の対応を確認出来ないため、プロジェクトメンバのスキルに依存している。 このため、仕様変更時に仕様の変更内容と検証項目との対応が個別対応となり、全体の整合	・設計レビューで検証項目の確認を行い、抽出の妥当性や網羅性を確認している。 ・網羅性を上げるために、過去の不具合事例から検証項目を追加する。 ・一部のツールには、検証回路にランダムなバグを埋め込み、その検出結果からテストベンチの検証項目網羅性を評価するものがある。 ・仕様変更では、担当者が変更内容をそれぞれ確認して、必要に応じて検証項目を修正した上で検証	【技術要求】 仕様と検証項目の対応が確認できる仕組みを用意する。 【解決案】 検証項目毎に仕様との対応をひも付けできる仕組みを作り、検証項目に漏れが出ない仕様表記方法を確立する。仕様変更があった場合に、変更履歴をたどることで、変更が必要な検証項目を見つけられる仕組みも用意する。	【技術要求】 検証項目の漏れを自動チェックできること 【解決案】 IPベース及プラットフォームベース設計の進展で、多くは検証項目が各IPとリンク付けされたデータとして整備可能である。IPの組み合わせについては、各IPの接続情報を元に、検証項目の候補をリストアップする。

検証項目の網羅的抽出

仕様から検証項目を網羅的に抽出する手法と最適な検証手順を決めることが難しい

ボトムアップで検証リストを作成してDRで確認

仕様と検証項目の対応が確認できる

検証項目の漏れの自動確認

国内活動

論理検証の課題と技術要求（抜粋）

■検証実行

重要課題	課題説明	2007年時点での対応策(取り組み)	2010年に実現すべき技術要求／解決策の案	2011年以降の技術要求／解決策の案
設計レベル間検証	高位設計(Cレベル)の導入により検証を効率化しようとしているが、高位レベルとRTLレベルの等価性検証ができないことや、高位のテストベンチを、RTLではそのまま利用出来ないため、RTLのための修正が必要となり、効率化	設計レベル間の検証は主として動的シミュレーションに頼っている。テストベンチの流用については、高位テストベンチを修正することで、RTLレベルで使えるようにしたり、ベクタをファイルダンプしてテスト系列として利用している。	【技術要求】 等価検証が実用的に使える 【解決案】 動作合成が回路タイプによらず処理できて、動作合成の単位では等価検証を実用化 そのためには、以下のことが必要となる。	【技術要求】 機能仕様と制約条件からアーキテクチャが合成され、通信部分はインタフェース合成で生成される。 機能仕様の正当性確認とゲートレベル以降のタイミング検証のみで実装可能
設計初期段階での性能見積もり	設計初期段階（上流）での性能見積もりとその検証手法が未確立であり、特に、電力、システム性能、動作周波数等は下流工程で初めて精度高い値が算出（抽出）されるため、設計手直し等の設計	上流では、設計経験、ノウハウに基づく人手による電力見積もりや高位（C言語など）モデルによるシミュレーションでシステム性能検証を実施している。最終確認はゲートレベルの検証で行っている。	【技術要求】 RTLで消費電力や性能の確認が高精度に出来ること 【解決案】 論理合成アルゴリズムを内蔵するなどで、RTLレベルでゲートレベル精度の消費電力検証	【技術要求】 仕様書の非機能要求として指定した電力、システム性能記述の制約に基づいて高位合成(アーキテクチャ、動作、論理)で回路を生成する

設計レベル間の検証

高位設計が導入される中で、設計階層毎に検証が必要で効率化が阻害されている

論理シミュレーションによる確認

等価検証の実用化

アーキテクチャ合成、動作合成の実用化

性能見積もり

設計初期段階での性能見積もりの方法が確立していない

経験に基づく見積もり

RTLでの高精度な見積もり方法の確立

非機能要件として指定した制約に基づくアーキ／動作合成

2010年の技術要求/解決策

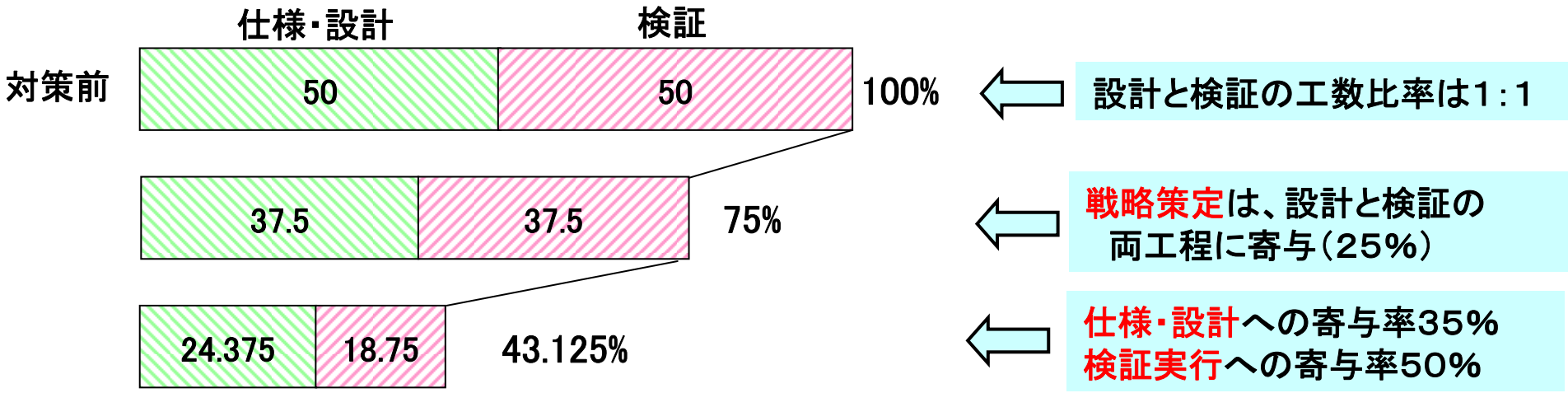
設計レベル	重要課題	2010年の技術要求／解決策
戦略策定	検証対象の状況に応じた最適な検証戦略の策定	最適な戦略を作るための共通な手順が確立される
	最適な検証のマネジメント	ベストプラクティスに基づいたマネジメント・ガイドラインが策定される
	IPモデルの整備と検証品質	IP利用に必要な各種モデルが用意され、その品質が保証されている
	機能検証スキルを持つ人材／専門家の育成	各エンジニアのスキル要件が明確になり、教育プログラムが用意される
仕様設計	明確で誤解がない仕様の作成	共通な仕様表現方法が確立される
	網羅的な検証項目の抽出と検証順序の最適化、および仕様変更への対応	仕様と検証項目の対応が確認できるようになる
検証実行	シミュレーションの高速化	2桁以上の高速化が実現される
	デバッグの効率化	アサーションが自動生成され、プロトタイプの内部状態が容易に観測できる
	設計レベル間検証	等価検証が実用化される
	設計初期段階での性能検証	RTLで性能や消費電力の確認が出来る

2010年の生産性向上率の見積り

設計レベル	重要課題の解決策	生産性向上率の見積もり内容	向上割合
戦略策定	最適な戦略を作るための共通な手順が確立される ベストプラクティスに基づいたマネジメント・ガイドラインが策定される IP利用に必要な各種モデルが用意され、その品質が保証されている 各エンジニアのスキル要件が明確になり、教育プログラムが用意される	・戦略策定とマネジメント改善 →10%向上 ・人材育成 →10%向上 ・IPモデル整備 →5%向上	25%
仕様・設計	共通な仕様表現方法が確立される 仕様と検証項目の対応が確認できるようになる	・仕様の明確によるミス削減 →50%向上 ・網羅性向上で検証作業増 →25%悪化 ・仕様変更対応効率化 →10%向上	35%
検証実行	2桁以上の高速化が実現される アサーションが自動生成され、プロトタイプの内部状態が容易に観測できる 等価検証が実用化される RTLで性能や消費電力の確認が出来る	・検証実行の高速化、デバッグ効率化 →50%向上 ・手変換ミスによる手戻りが5回→2.5回でシミュレーション回数半減 →50%向上 ・見積もり誤差による設計手戻りが2回→1回に半減 →50%向上	50%

2010年の設計検証の生産性向上率

設計生産性向上率は2.3倍（100／43.125）
ITRSで記載されている生産性要求値(1.96)をクリア



ITRS2007の設計生産性要求テーブル

	2007	2008	2009	2010	2011	2012	2013	2014
Trend: SOC total logic size (normalized to 2007)	1.00	1.29	1.62	2.12	2.64	3.24	4.07	5.29
Requirement: % of reused design	38%	42%	46%	50%	54%	58%	62%	66%
Requirement: Productivity for new designs （normalized to 2007)	1.00	1.25	1.54	1.96	2.38	2.84	3.47	4.37

論理検証SWGのまとめ

- 直面している技術課題から、設計生産性に影響が大きい10項目の重要課題を抽出
- 解決のための中期的な技術要求と解決策を提示
  - 戦略策定：戦略策定手順の確立、ガイド整備 など
  - 仕様・設計：表記の標準化、検証項目との対応など
  - 検証実行：検証速度向上、等価性検証実用化など
- 解決策の実現で、2010年ではITRSの要求値を上回る2．3倍の設計生産性向上が見込まれる
- 解決策の実現には、「設計技術開発」の強化・加速が重要

物理設計SWG

◆目標

－シリコン複雑度の影響を加えた物理設計の生産性要求を定量化

－設計生産性向上を実現するための解決策を提示

◆背景

－ 物理設計においては、システム複雑度（回路規模）に加えて、シリコン複雑度を織り込んだ設計生産性要求が必要

新規追加

$$\text{設計生産性} = \frac{\text{システム複雑度} + \text{シリコン複雑度}}{\text{設計工数}}$$

－ シリコン複雑度要因例：

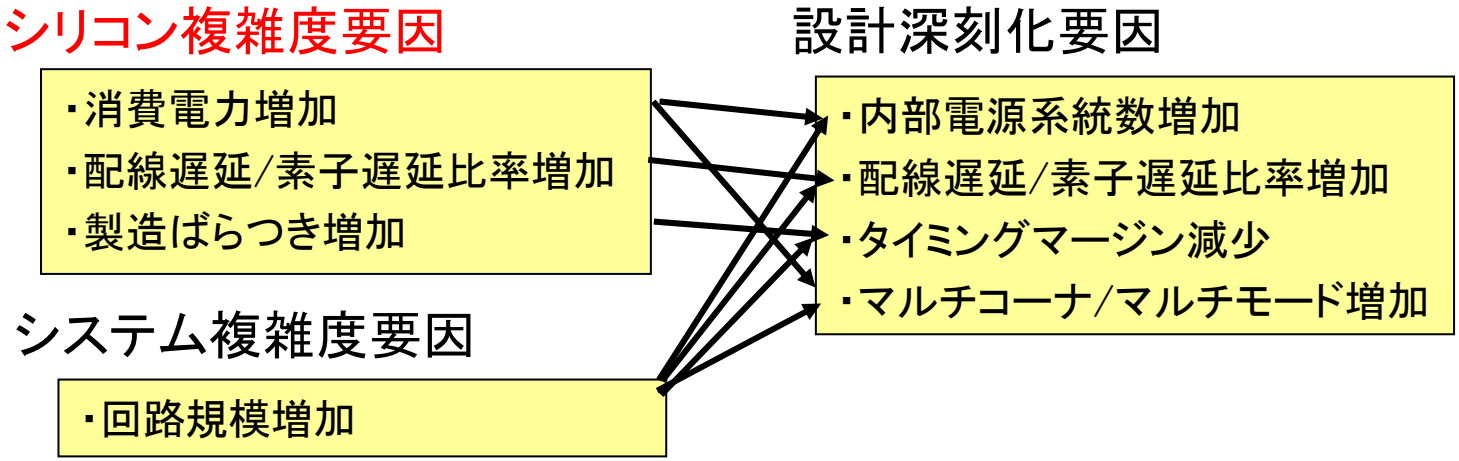
・消費電力対策、製造ばらつき対策　など

◆スコープ

－ RTL記述からレイアウトデータ（GDSⅡ）を生成するまでの工程

設計生産性要求の定量化手順

① シリコン複雑度要因と設計深刻化要因の抽出



② 各複雑度を設計深刻化要因を介して定量化

ITRS指標から各要因を定量化

例：・「タイミングマージン減少による設計深刻化」

∝「製造ばらつき増を起因とする遅延変動増加」×「回路規模増加」

・「内部電源系統数増加による設計深刻化」

∝「消費電力増加」×「回路規模増加」

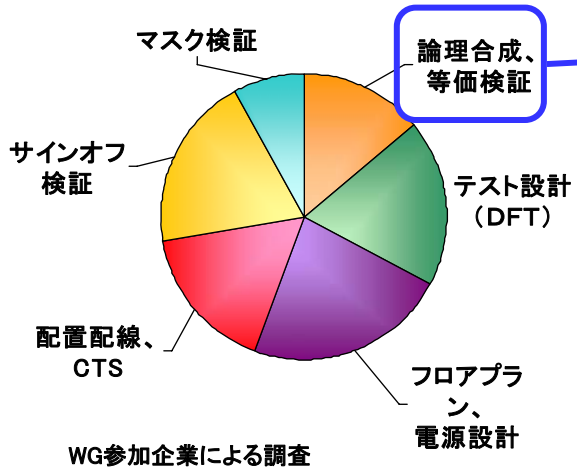
設計生産性要求の定量化手順

③ 物理設計への影響を定量化

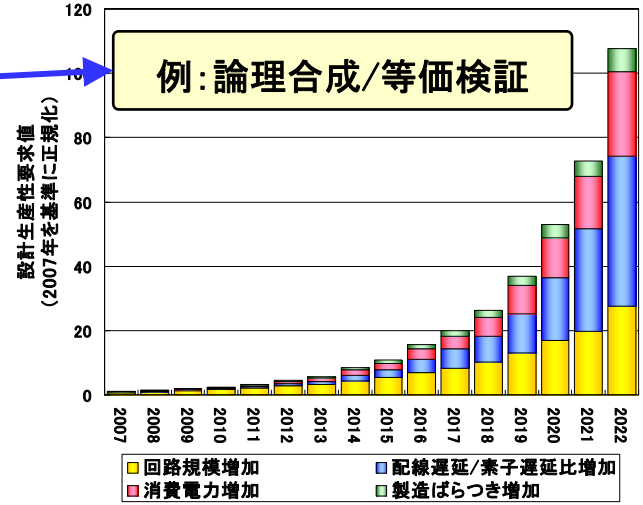
- 物理設計の6つの工程へ分割（A）
- 各工程の工数分布を調査（B）
- 各工程への複雑度（システム＋シリコン）影響率を算出（C）
- （B）の分布を2007年度の初期値として（C）を合算し、物理設計全体への影響を定量化

物理設計の工程分割（A）

各工程の工数分布調査（B）

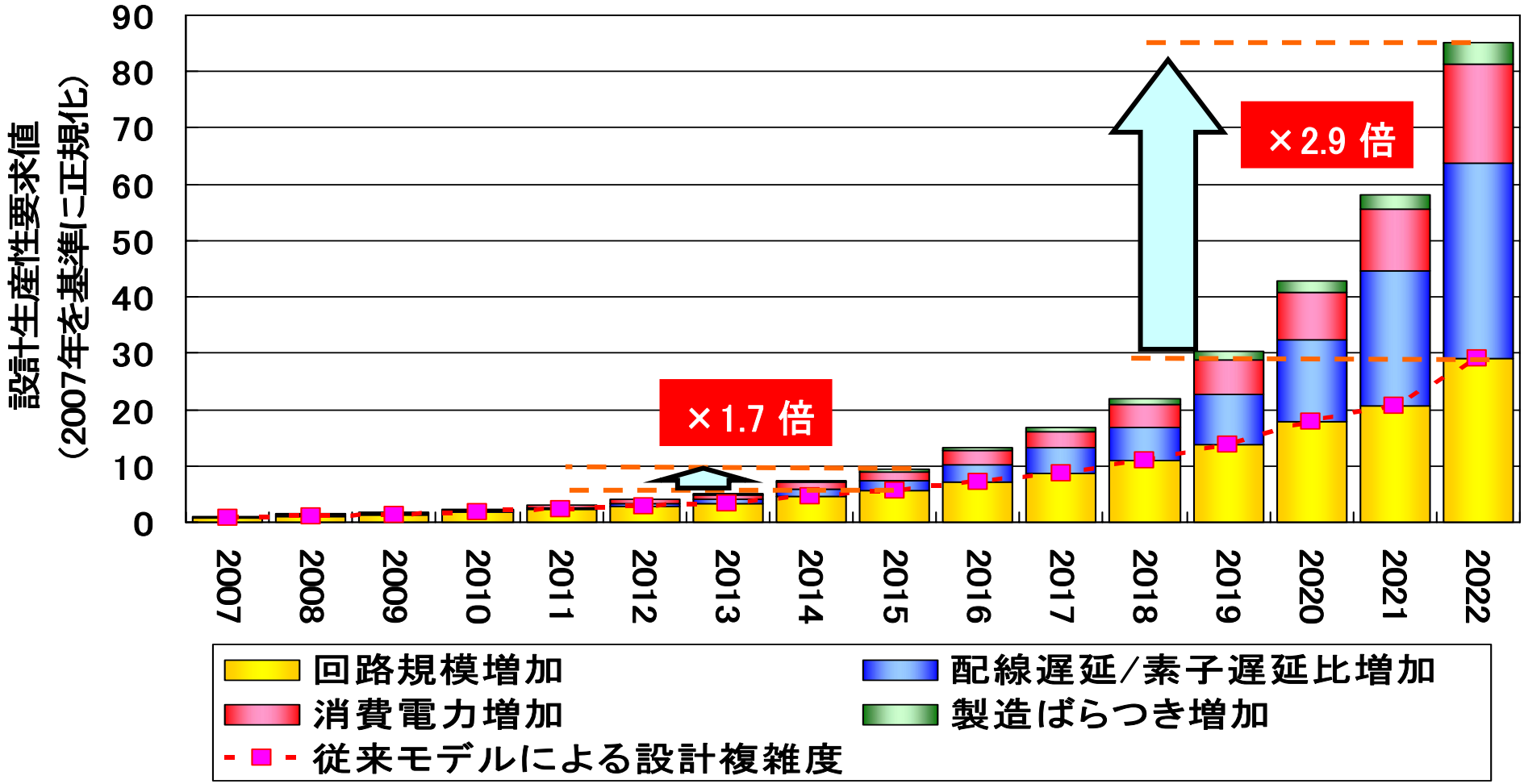


各工程への影響度グラフ（C）



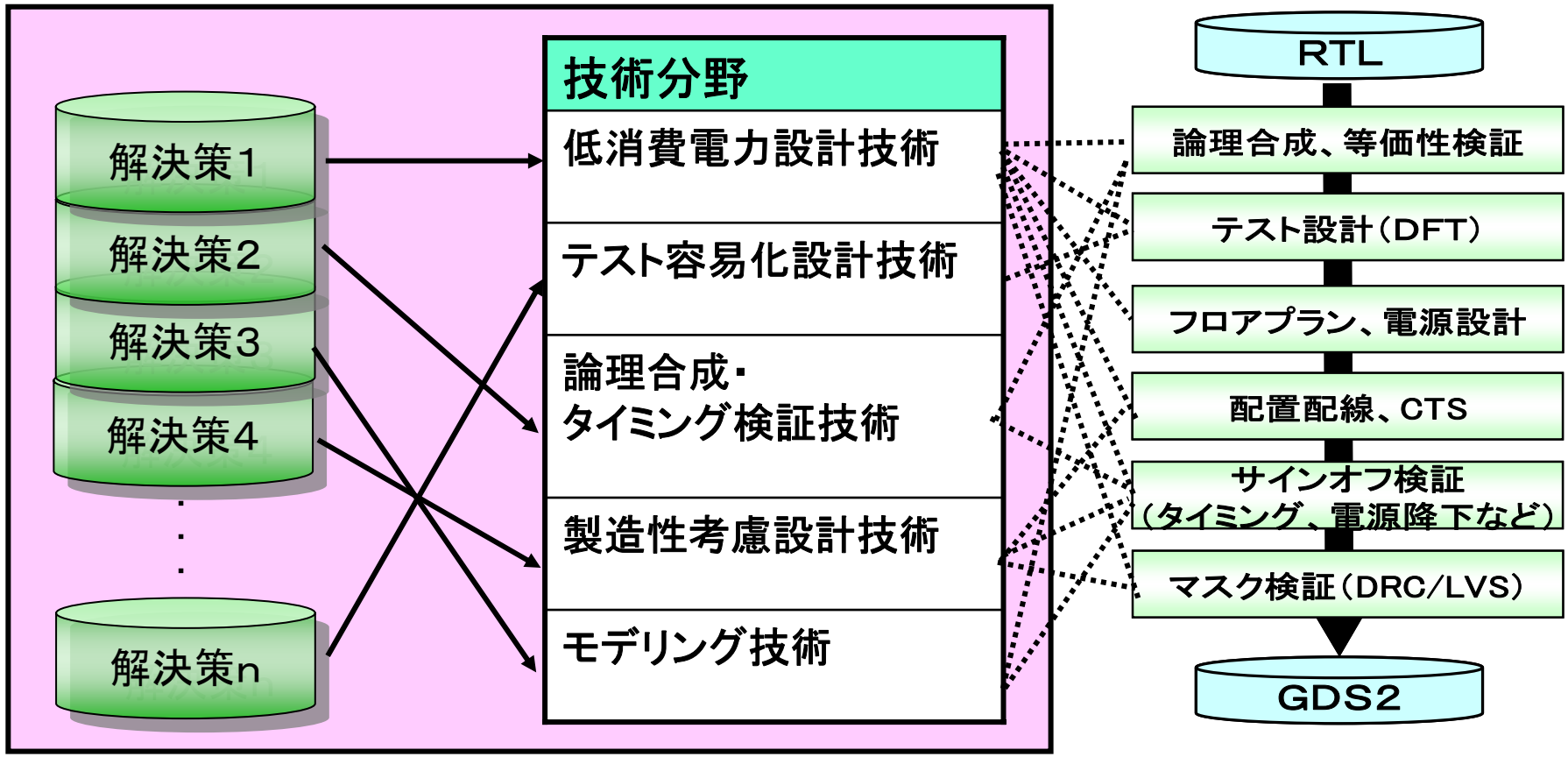
物理設計の生産性要求の定量化

従来（システム複雑度のみ）に比べて、生産性向上の要求値が
1.7倍(2015年)、**2.9倍**(2022年)にアップ



生産性向上の解決策検討

物理設計工程の生産性向上のための課題解決策を検討し、設計工程と関連付けて「5つの技術分野」で整理



国内活動

物理設計の課題解決策(1/2)

技術分野	課題解決策	
	～2010	2011～
低消費電力設計技術	・クロックゲーティッドCTS、活性化率を考慮した消費電力見積もり技術 ・マルチVDDを考慮した消費電力見積もり技術 ・RTLレベルでの消費電力見積もりの高精度 ・パワー記述の標準化 ・電源系統考慮の論理合成技術 ・マルチVth/VDD考慮の物理合成技術 ・DVFS(ダイナミックな電圧と周波数のスケーリング)の自動化技術	・IR drop対応の電源ラインを考慮した見積もり技術 ・電源内蔵Chipによるバイアス可変考慮した消費電力見積もり技術 ・バイアス可変ブロック割り出し自動化技術 ・非同期回路適用の自動化技術 ・小振幅クロック考慮論理合成技術 ・Vth可変制御回路考慮の物理合成技術 ・上流言語(C等)からの多電源動作合成技術
テスト容易化設計技術	・テスタビリティ解析技術 ・低故障検出率の論理抽出機能の精度向上 ・高速遅延故障検出ATPG技術 ・故障パターン増加を抑える高速圧縮技術	・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自動修正技術 ・上位言語->RTL時の低検出率回路回避技術 ・故障検出向上回路の自動生成技術 ・故障検出向上回路と論理合成/フロアプランとの連携

物理設計の課題解決策(2/2)

技術分類	課題解決策	
	～2010	2011～
論理合成・タイミング検証技術	・配置考慮の論理合成技術 ・論理合成と物理合成の融合技術 ・CTS考慮の合成技術 ・クロックメッシュ対応技術 ・マルチコーナマルチモードのタイミング同時最適化技術 ・統計的タイミング解析技術	・仕様からの合成制約自動生成技術 ・制約条件の等価検証技術 ・複数電圧間におけるタイミング自動最適化技術 ・可変VDD/Vthにおけるクロック自動最適化技術 ・高位言語での動作合成からの配置合成技術 ・小振幅クロック考慮技術 ・非同期設計考慮技術
製造性考慮設計技術	・容量セル挿入最適化技術 ・パーティクル/リソ/CMP考慮最適化技術 ・シグナルEM対応クロック配線技術 ・電源、信号EM解析・検証技術 ・マルチモードIRdrop解析技術 ・歩留り考慮ライブラリキャラクタライズ技術	・IR drop考慮の電源配線自動最適化技術 ・シグナルEM対応バス配置配線技術 ・ダイナミックIRdrop解析技術 ・歩留まり考慮論理合成技術 ・製造インタフェイス考慮技術
モデリング技術	・電源記述仕様の標準化モデル ・マルチVDD/Vth/Ta及び可変電圧考慮タイミングモデル ・状態依存、ばらつき考慮消費電力ライブラリ生成技術	・インスタンス毎の電源認識対応のモデル ・可変VDD/Vth及びばらつき考慮のタイミングモデル適用 ・小振幅クロック対応のタイミング、消費電力モデル

物理設計SWGのまとめ

- 物理設計工程の設計生産性要求に「シリコン複雑度」を新たに導入
- 設計生産性要求はシステム複雑度（＝回路規模）にシリコン複雑度を加えた結果、
  - Near Term（2015年）で1.7倍(従来比)
  - Long Term（2022年）で2.9倍(従来比)
- 生産性向上のための課題解決策を提示
- 高精度な見積もり/解析技術と各種最適化の自動化技術の開発加速が必要

2007年度国内活動のまとめ

論理検証SWG活動

- 重要課題を抽出し、中期的な技術要求と解決策を詳細検討
- 検証戦略ガイド整備、仕様記述標準化、検証EDA技術向上が必要
- 上記解決策の2010年時点の設計生産性向上率を定量化

  2.3倍の生産性向上の見込み ＞ ITRS要求（1.96）

物理設計SWG活動

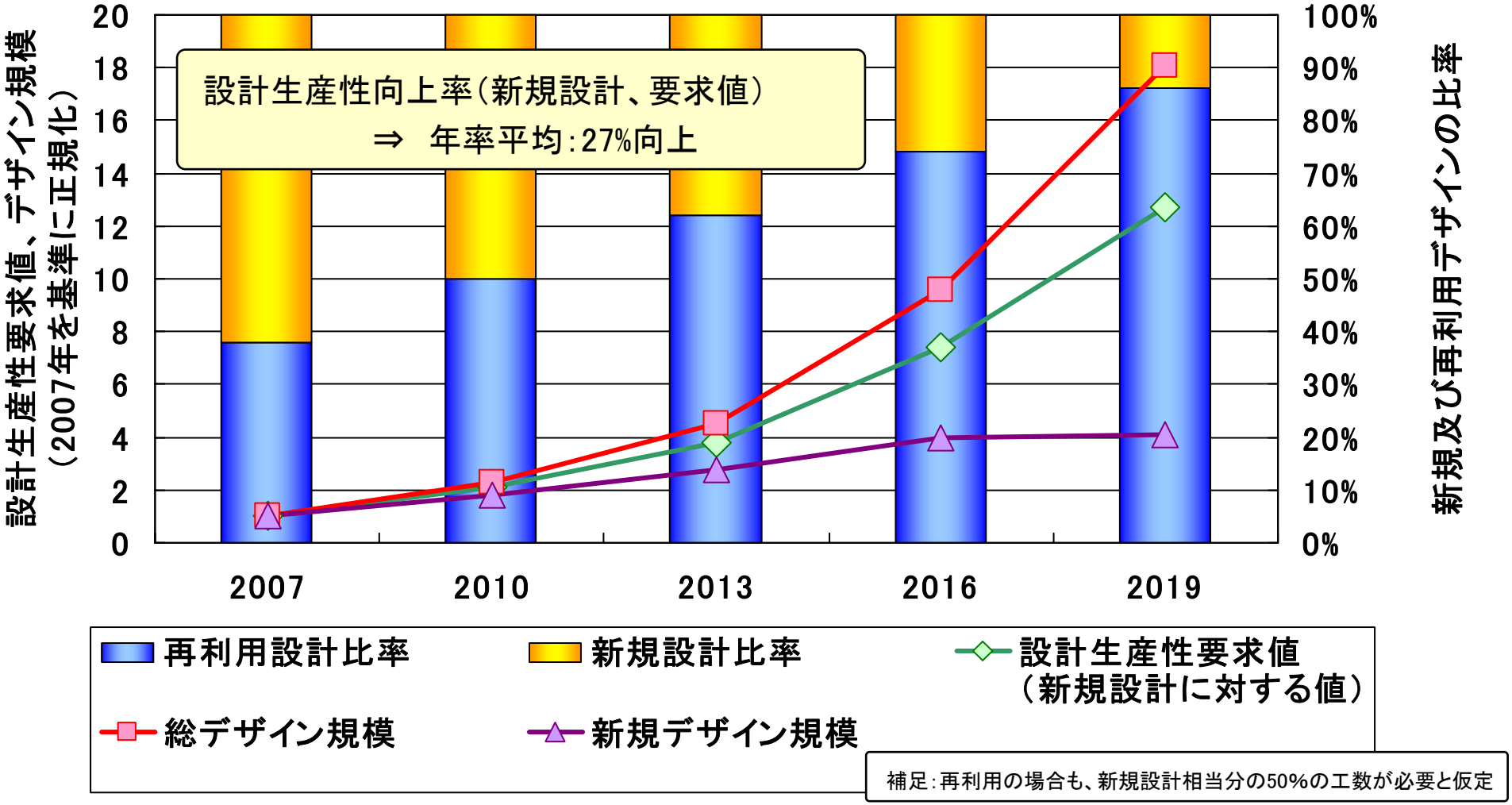
- シリコン複雑度の影響を加えた設計生産性要求を定量化

  Near Term（2015年）で1.7倍（従来比）

  Long Term（2022年）で2.9倍（従来比）
- 設計生産性向上のための課題解決策を提示
- 高精度な見積もり/解析技術と各種最適化の自動化技術が必要

補足資料

ITRS2007掲載の設計生産性要求モデル



論理検証の課題と技術要求

設計レベル	重要課題	課題説明	2007年時点での対応策(取り組み)	2010年に実現すべき技術要求／解決策の案	2011年以降の技術要求／解決策の案
戦略策定	検証対象の状況に応じた最適な検証戦略の策定	検証対象に要求される設計品質や適用可能な設計手法に応じて、最適な検証戦略を立案することが難しい。さらに、設計品質や検証レベルの異なったブロックが混在する場合、個々に検証ゴールや手法の選択が必要となり、全体最適な検証戦略を立案することが難しい。 仕様変更時に変更内容の影響を正確に把握して、適切な検証戦略に修正することが難しい。	・プロジェクトリーダが全体の検証指針を示して、それに基づいてトップはトップ検証担当がサブブロックは各サブブロック・リーダがそれまでの経験に基づいて、検証対象の特性を考慮した検証環境を構築している。 ・過去の経験や他の設計事例から起こりうる問題点を設計レビューの中で確認して、その防止策や効率的な検証方法を検証手法や環境に取り入れている。	【技術要求】 個々の経験やスキルのばらつきによらず高品質な検証環境が構築できる仕組みの実用化 【解決策案】 標準となる検証手法が確立する(種々の検証事例に基づいた網羅的な検証手法マニュアルの作成)	【技術要求】 検証ターゲットに応じた検証手法が用意され、検証環境が容易に構築できる
戦略策定	最適な検証のマネジメント	利用可能なリソースを有効活用し、検証過程での種々の問題解決して、リスク管理を行いながら最適なQCDのバランスの取れたマネジメントを実行することが難しく、プロジェクトリーダとプロジェクトメンバのスキルに依存している。	・過去の事例から問題点、対処を考え、検証戦略を改良していく。 ・プロジェクトによっては、独立した検証技術チームが検証を実施したり、コンサルティングを行い検証戦略や検証環境構築を支援することもある。	【技術要求】 ・プロジェクトリーダ/メンバが目指すべきベストプラクティス(やるべき項目、手順、リスク管理など)を明確するためのガイドラインの策定。 ・プラットフォームベースの検証では、プロジェクトリーダ/メンバのスキルに依存せず最適な検証環境が構築できるよう、変更追加が起こる部分の修正方法が定型化できていること。 【解決案】 検証マネジメントに必要な検討項目をプラットフォームに対するテンプレートとして用意して、各項目の記載内容の詳細定義と事例を示したガイドを作成する。	【技術要求】 プロジェクトリーダ/メンバのスキルに依存せず、最適な検証のマネジメントができる。 【解決策】 マネジメントガイドに事例等の知識ベースを追加し、様々な条件を入力することで最適なマネジメント手法を生成するシステムを構築する。

論理検証の課題と技術要求

設計レベル	重要課題	課題説明	2007年時点での対応策（取り組み）	2010年に実現すべき技術要求／解決策の案	2011年以降の技術要求／解決策の案
戦略策定	IPモデルの整備と検証品質	使用しているIPのモデル(ex.高位動作モデル)や検証IPが不足していたり、提供されるIPや検証IPの品質が十分でないことにより、検証が効率的に行えない。	実績があるIPを、その使用条件のもとで情報共有して使っている。 高位モデルなど不足しているモデルは自前で開発することもある	【技術要求】 高位モデルとRTLモデルがセットで販売され、SoC開発に十分なIP品種が流通する。 検証IPは利用環境によらず網羅的に検証され、利用者の環境で品質確認が容易 【解決案】 IP業界として品質を示すための基準を定めて、定量的な品質表示を可能とする。例えば、標準となる検証手法（Verification Methodology Manual（VMM） など）を決めて、これに基いた検証環境をIPと合わせて提供し、検証レベルがユーザー側でも確認できるようにする。	【技術要求】 完全に検証され、仕様通りに動作するIPが流通する
戦略策定	機能検証スキルを持つ人材／専門家の育成	機能検証にはダイナミック検証やスタティック検証など複数の技術が必要で、検証用言語も複数存在するため、設計者が検証技術もカバーするのは困難で、専門家の育成が必要になっている	機能検証ガイドの作成や、検証技術講座などで基本知識を教えて、実務では検証技術担当がプロジェクトに参加して、OJTで対応したり、テストベンチやアサーションのテンプレートを作成して、設計者に引き次ぐことで、検証ノウハウを伝えている。	【解決案】 検証エンジニアのスキル要件を明確にする体系的な教育プログラムを整備する。 各社からの知識、ノウハウを持ち寄り、共同でガイドライン作成、教育講座開設を行う。 受講の容易さのためにWeb教育による基本レベル裾野展開を行う。 トランジスタや論理回路の物理現象理解の基礎からの教育も実施する（現在、論理合成等EDAツールの利用により、電気の基本的な振る舞いを理解する部分が空洞化している）	【一般的要求】 知識ベースを利用したコンサルティングシステムや専門家による支援を充実させ、個々のスキルのばらつきを補う仕組みを用意する

論理検証の課題と技術要求

設計レベル	重要課題	課題説明	2007年時点での対応策(取り組み)	2010年に実現すべき技術要求／解決策の案	2011年以降の技術要求／解決策の案
設計	明確で誤解がない仕様の作成	SOC規模拡大により様々な機能が組合され、仕様が複雑になってきた。このため、個々の機能の仕様は正しくても全体としては矛盾している「仕様の誤り」や、条件が網羅されていなかったり、暗黙の前提に基づいた記述による「仕様の漏れ」などが生じている。「仕様の誤り」がある場合は、仕様通りに設計してもSOCは意図通りの動作をしない。「仕様の漏れ」がある場合は、設計者の解釈が入り込むことにより、SOCは意図通りの動作をしない。	・仕様を設計と検証の別々の立場から読み下し、相互確認をすることで仕様の妥当性を検証している。 ・シミュレーションによる機能検証とともに、プロトタイピングによる実機評価をおこなうことで、二重確認を行っている。 ・システムレベル検証ツールで、シミュレーションによる動作確認で、設計の初期段階で仕様を検証している。	【技術要求】 機能仕様を誰が表現しても画一的な表現になるような仕組みを用意する ・仕様記述の標準化により理解が共通になり、誤解による設計誤りを防ぐとともに、相互確認が容易となる。 ・仕様記述を設計資産化出来るので、再利用により記述ミスが軽減され仕様の完成度が上がる。 【解決案】 機能仕様については、機能動作とその入出力で定義する。機能カテゴリ(制御(状態遷移系)、信号処理(データパス系)など)に対して仕様テンプレートが用意され、入出力についても、画像、音声、制御などのライブラリから選択して機能仕様を表現する。制約条件として必要な情報のみ追記する。機能はキーワード毎に動作を記述する。キーワード毎に各機能モジュール間で相互チェックを支援する仕組みが提供される。	【技術要求】 仕様の矛盾、網羅性が容易にチェックできること 例えば、キーワード毎に矛盾がないことが言語解析により確認される。 非機能仕様も、機能仕様に対応する形で、フォーマルに記述できること。

論理検証の課題と技術要求

設計レベル	重要課題	課題説明	2007年時点での対応策(取り組み)	2010年に実現すべき技術要求／解決策の案	2011年以降の技術要求／解決策の案
設計	網羅的な検証項目の抽出と検証順序の最適化、および仕様変更への対応	仕様から検証項目を網羅的に抽出する手法や適切な検証順序を決めることが定式化されておらず、仕様と検証項目の対応を確認出来ないため、プロジェクトメンバのスキルに依存している。 このため、仕様変更時に仕様の変更内容と検証項目との対応が個別対応となり、全体の整合性を確認することが困難となっている。	・設計レビューで検証項目の確認を行い、抽出の妥当性や網羅性を確認している。 ・網羅性を上げるために、過去の不具合事例から検証項目を追加する。 ・一部のツールには、検証回路にランダムなバグを埋め込み、その検出結果からテストベンチの検証項目網羅性を評価するものがある。 ・仕様変更では、担当者が変更内容をそれぞれ確認して、必要に応じて検証項目を修正した上で検証を行う。検証項目の妥当性や、検証環境の変更箇所は、設計レビューで確認し共有化している。	【技術要求】 仕様と検証項目の対応が確認できる仕組みを用意する。 【解決案】 検証項目毎に仕様との対応をひも付けできる仕組みを作り、検証項目に漏れが出ない仕様表記方法を確立する。仕様変更があった場合に、変更履歴をたどることで、変更が必要な検証項目を見つけられる仕組みも用意する。	【技術要求】 検証項目の漏れを自動チェックできること 【解決案】 IPベース及プラットフォームベース設計の進展で、多くは検証項目が各IPとリンク付けされたデータとして整備可能である。IPの組み合わせについては、各IPの接続情報を元に、検証項目の候補をリストアップする。

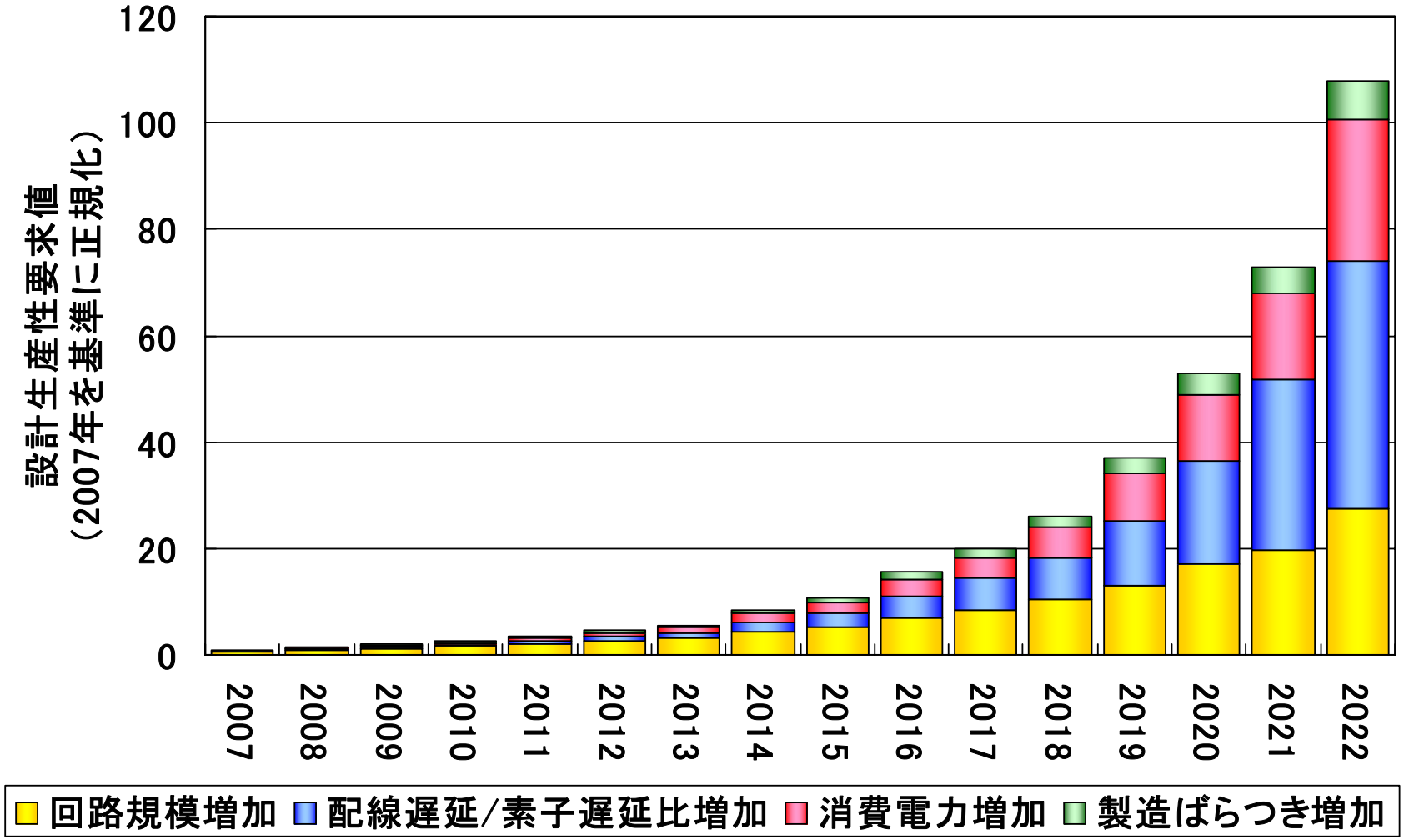
論理検証の課題と技術要求

設計レベル	重要課題	課題説明	2007年時点での対応策(取り組み)	2010年に実現すべき技術要求／解決策の案	2011年以降の技術要求／解決策の案
検証実行	シミュレーションの高速化	大規模化と機能の複合化により、検証項目が膨大になり、シミュレーションが検証期限までに終わらない状況にある。高位検証やプロトタイピング等の検証手段はあるが、(RTL)サインオフとしての機能検証はRTLダイナミックシミュレーションに基づいたものとなるため、検証時間の増大を抑制する決め手にはなっていない。	シミュレーションの高速化としてエミュレータやアクセラレータを利用しているが、エミュレータやアクセラレータで回路を動作させるまでの立ち上げに時間がかかったり、回路によっては修正が必要になり、高速化が十分行えないこともある。 マルチCPUの計算機や複数台の計算機を利用して、複数CPUでの分散実行による高速化が、比較的低コストで実現可能となってきている。また、マルチコア計算機をターゲットとした論理シミュレータが商品化されている。	【技術要求】 2桁以上の高速化 【解決案】 マルチCPU分散実行対応のシミュレータとその低コスト化。 大規模高速でプログラミングが容易なプロトタイピングの実現	【技術要求】 リアルタイムシミュレーションの実現
検証実行	デバッグの効率化	問題の場所の特定は、シミュレーション結果などの目視確認が主となるため、時間がかかり、効率化が難しい	目視作業を軽減するために、アサーションの利用やフォーマル検証などを使う試みをしている。	【技術要求】 アサーションの生成やプロトタイプ・プロービングの容易化 【解決案】 擬似エラーの解析支援技術の向上 アサーション記述を統一して、ツールの使いやすさを向上させる 波形表示ツールのトレーサビリティー強化などのデバッグ機能の向上 FPGAプロトタイプの内部ノードのモニタ機能やデバッグ機能の向上	【技術要求】 仕様と検証項目から不良箇所を特定する技術の開発

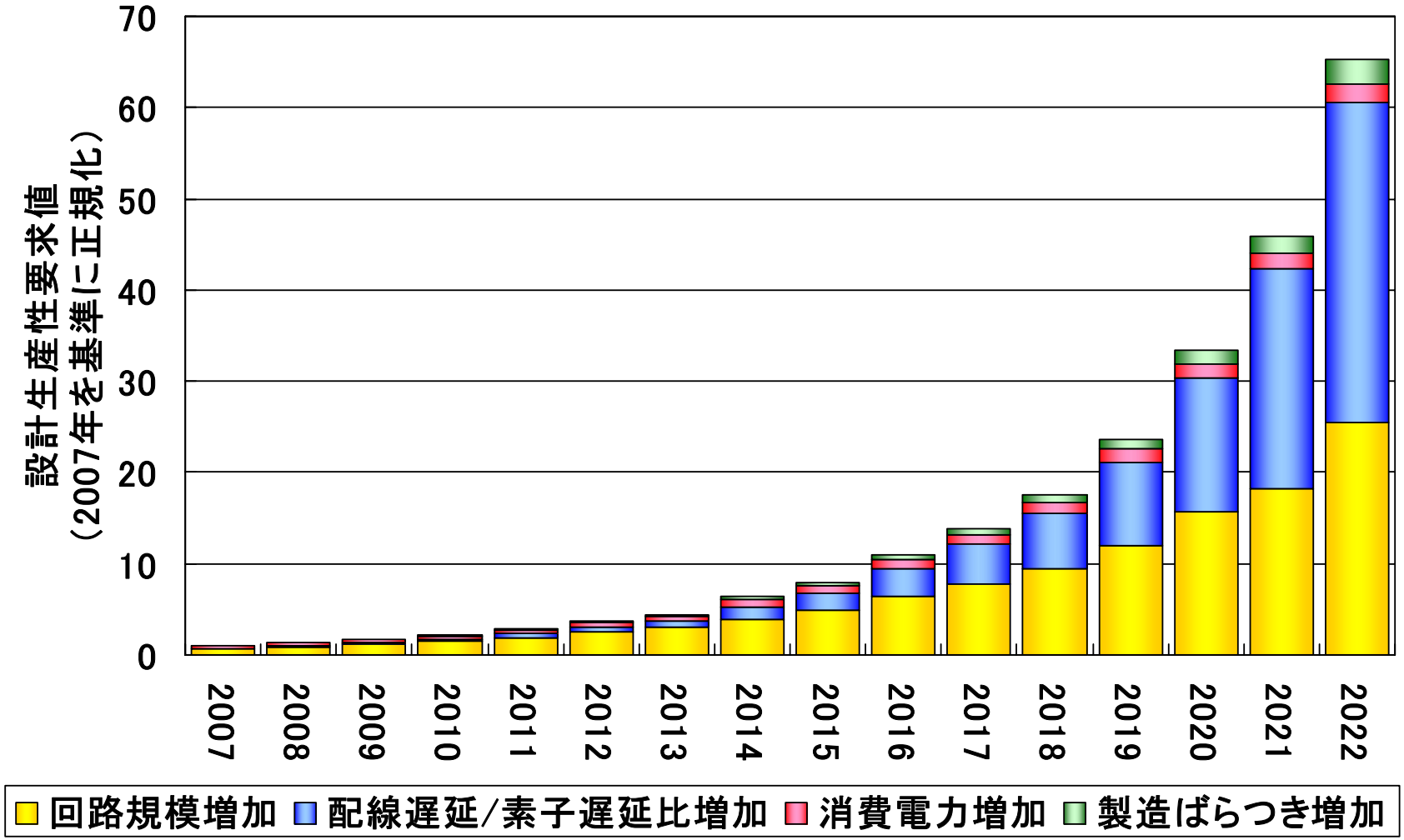
論理検証の課題と技術要求

設計レベル	重要課題	課題説明	2007年時点での対応策(取り組み)	2010年に実現すべき技術要求／解決策の案	2011年以降の技術要求／解決策の案
検証実行	設計レベル間検証	高位設計(Cレベル)の導入により検証を効率化しようとしているが、高位レベルとRTLレベルの等価性検証ができないことや、高位のテストベンチを、RTLではそのまま利用出来ないため、RTLのための修正が必要となり、効率化が阻害されている フォーマル検証技術は制約(回路規模、使い方)があり、効率的な使い方が難しい	設計レベル間の検証は主として動的シミュレーションに頼っている。テストベンチの流用については、高位テストベンチを修正することで、RTLレベルで使えるようにしたり、ベクタをファイルダンプしてテスト系列として利用している。 フォーマル検証、等価検証については、回路規模や使い方の制約を理解した専門家のサポートを得ながら使っている。	【技術要求】 等価検証が実用的に使える 【解決案】 動作合成が回路タイプによらず処理できて、動作合成の単位では等価検証を実用化 そのためには、以下のことが必要となる。 ・抽象レベルの厳密な定義と変換ルールの規程 ・動作合成のレジスタ生成情報などを利用して等価検証の適用範囲を拡大 ・ノウハウ、経験を盛り込んだ知識ベースの自動分割と並列処理による全体検証	【技術要求】 機能仕様と制約条件からアーキテクチャが合成され、通信部分はインタフェース合成で生成される。機能仕様の正当性確認とゲートレベル以降のタイミング検証のみで実装可能となる。 動作合成、論理合成によりゲートレベルまで自動的に生成できること。
検証実行	設計初期段階での性能検証	設計初期段階(上流)での性能見積もりとその検証手法が未確立であり、特に、電力、システム性能、動作周波数等は下流工程で初めて精度高い値が算出(抽出)されるため、設計手直し等の設計の非効率化を招いている。	上流では、設計経験、ノウハウに基づく人手による電力見積もりや高位(C言語など)モデルによるシミュレーションでシステム性能検証を実施している。最終確認はゲートレベルの検証で行っている。	【技術要求】 RTLで消費電力や性能の確認が高精度に出来ること 【解決案】 論理合成アルゴリズムを内蔵するなどで、RTLレベルでゲートレベル精度の消費電力検証(見積もり)や動作周波数の確認が出来るようにする	【技術要求】 仕様書の非機能要求として指定した電力、システム性能記述の制約に基づいて高位合成(アーキテクチャ、動作、論理)で回路を生成する

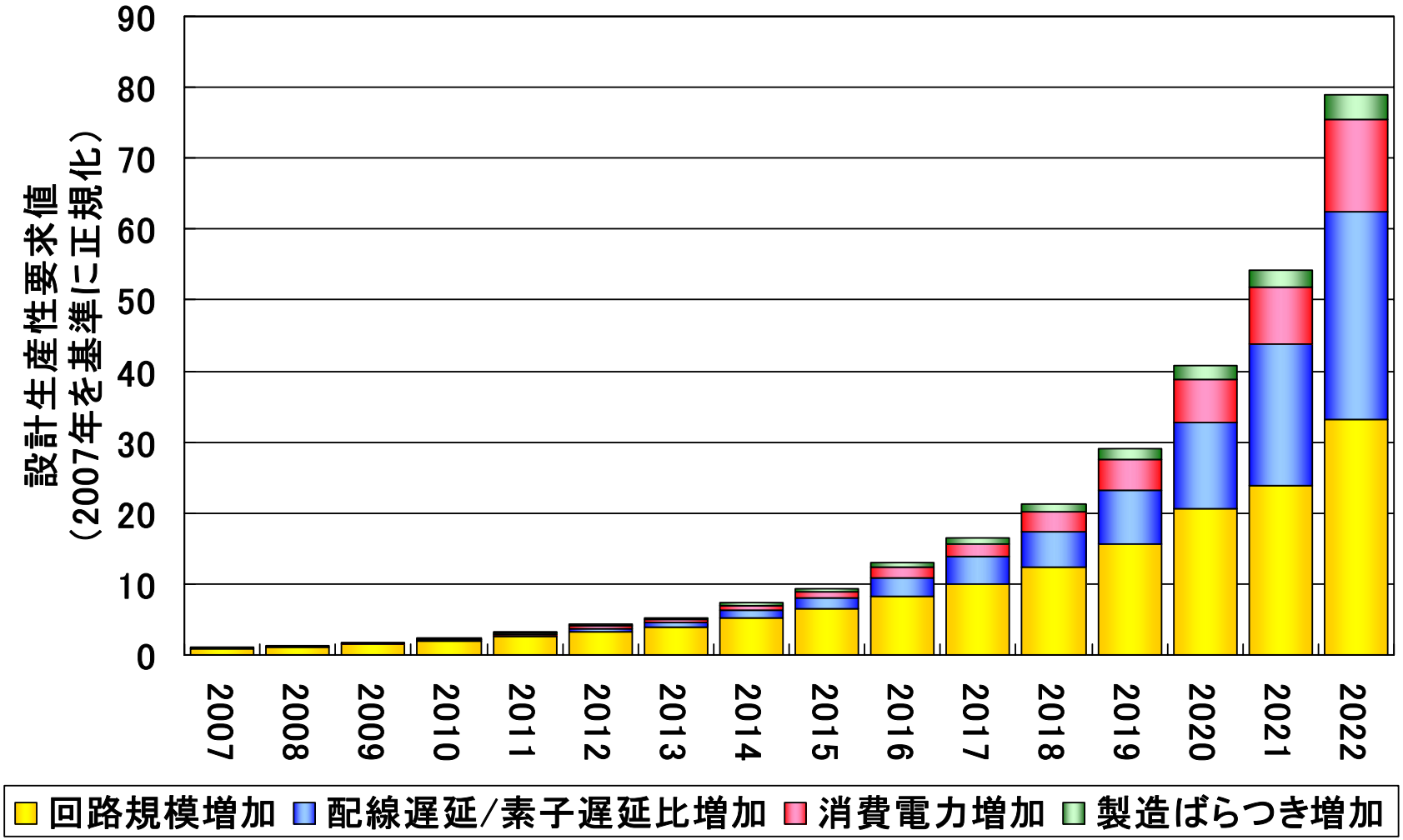
シリコン複雑度を考慮した設計生産性要求
「論理合成、等価性検証」工程



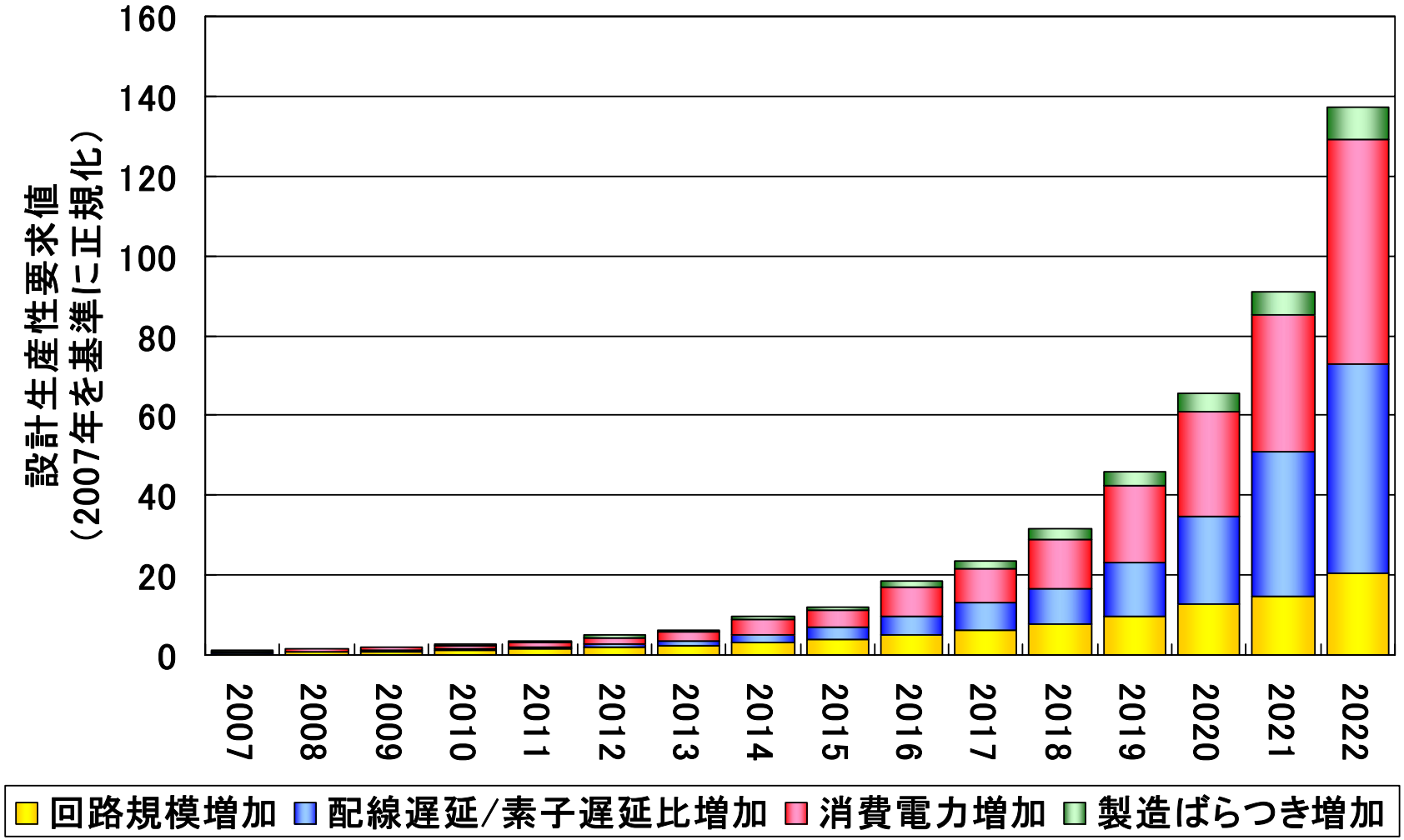
シリコン複雑度を考慮した設計生産性要求
「フロアプラン、電源設計」工程



シリコン複雑度を考慮した設計生産性要求
「配置配線、CTS」工程



シリコン複雑度を考慮した設計生産性要求
「サインオフ検証」工程



シリコン複雑度を考慮した設計生産性要求
「マスク検証」工程