システムＬＳＩ設計技術　103－11
（2001. 11. 29）

システムLSI設計における CADシミュレーション技術

松田 昭信

九州松下電器（株）情報システムセンター
〒812-8531 福岡県福岡市博多区美野島 4-1-62

E-mail: aki@icwh.kme.mei.co.jp

あらまし システム LSI の上流設計時点でシミュレーションを行う目的は設計の検証である．これには静的な論理機能の確認，遅延時間を考慮した動的な動作の確認という異なる目的が含まれている．よって，設計回路が機能仕様と違う論理的なバグが存在する場合には，設計の修正には煩雑な作業が生じる．このようなバグを早期に解決するため，設計の上流工程でのシミュレーション手法の分析をおこない，システム LSI 設計者に効果的なシミュレーションツールの活用技術を考察する．

キーワード システム LSI，シミュレーション，上流工程

CAD Simulation Technology in System LSI Design

Akitoshi MATSUDA

Information System Center, Kyushu Matsushita Electric Co.,Ltd.
4-1-62, Minoshima Hakata-ku, Fukuoka, 812-8531 Japan

E-mail: aki@icwh.kme.mei.co.jp

Abstract The purposes to do simulation at the upper-class design point of system LSI are the verification of design. These include the two different purposes, one is the confirmation of static logic function, the other is the confirmation of dynamic movement with delay time. Therefore, when the logical bugs exist in design circuit with different from function specification, the complicated works occur in the revision of design. We study the technology of effective practical use of simulation tool in upper-class process on system LSI designer by analyzing simulation technique in order to solve bug early.

Key words System LSI, Simulation, Upper-class process

1. はじめに

半導体技術の進歩はとどまることを知らず，半導体集積回路の集積度は 18 ヶ月で 2 倍という高い割合で伸びており，現在では数千万トランジスタ規模の汎用マイクロプロセッサも製造されている．この傾向は今後も継続すると予想されており，2010 年頃には最小線幅 0.07 μ m よりも微細化され，本格的なディープサブミクロン時代が到来することになる．予測によれば，約 7 年後には 10 億トランジスタ規模の集積度を持つシステム LSI が実現可能になると見られている[1],[2].

しかし，ここで深刻な事態が生ずるわけである．米国 SIA の調査によると，1 チップ上に集積される最先端の半導体回路の集積度は年率 58%の割合で増加している．このようなスピードで集積度が増大すると設計生産性が追いつかなくなる．これまでに設計自動化（CAD）ツールの開発と設計レベルの抽象度の向上によっても，図 1 に示すように設計生産性（設計者 1 人月当りの設計可能回路規模）は年率 21%しか向上していない．この状態が続くと生産性のギャップは広がるばかりである．すなわち，設計生産性の向上率が現在のままであると仮定すると，システム LSI の回路規模が増大するのに伴って開発コストが増えるだけでなく，設計開発プロジェクトの規模も肥大化し，遂にはデバイス集積度に設計開発が対応できなくなる[3].

設計生産性を向上するためには，設計のターンアラウンドタイム（TAT）を短縮でき，かつ設計のイタレーション回数が削減できる，新しい設計手法を構築する必要がある．設計記述作成の効率化は TAT の短縮を達成できる．また，上流工程での設計記述の検証はイタレーション回数の削減を達成するために必要である．単に「検証」と言っても，評価項目は機能，性能，信頼性など多数存在する．その中で設計の上流工程で検証すべき最も重要な項目の 1 つはシステムの機能である．

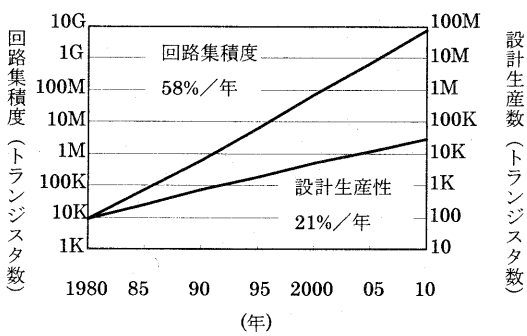

図 1．回路集積度と設計生産性のギャップ

以下では，まず現在の VLSI 設計フローと設計の記述方法について述べる．次に設計の上流工程でのシミュレーション手法を分類する，これらのシミュレーション技術の特徴技術について解説する．さらに，最近 LSI 開発者の関心を集めている C ベース設計手法について述べる．

2. VLSI 設計フローと CAD 技術

まず，VLSI 設計フローを説明する．VLSI 設計の各ステップを図 2 に示す．同図で，システムレベル設計から論理設計までを「上流設計」，レイアウト設計からマスクパターンまでを「下流設計」と呼ぶ．ハードウエア記述言語（HDL）を用いて設計を行う場合の上流設計の一般的な手順は，次のようになる．

(1) システムレベル設計

与えられた設計仕様（機能仕様）に基づいて，システムレベルでの設計記述が行われる．また，システムの仕様を VLSI で実現するために，要求される

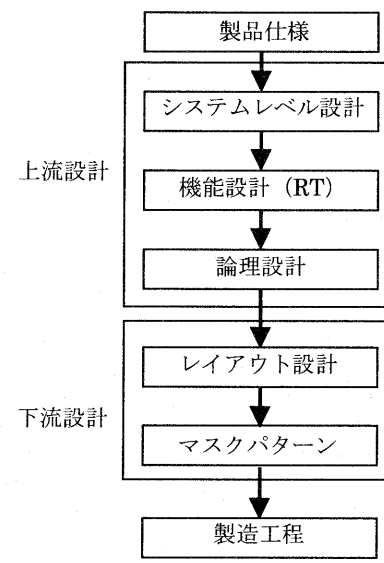

図 2．VLSI 設計フロー

動作や構成を決定する工程である．ここで，システムLSIは一般にCPUやDSP，メモリ回路を内蔵するので，ソフトウエアで機能を実現できる．そこで，ハードウエア／ソフトウエア各々による分担を決定するシステム分割が行われる．この設計レベルでは，システムは複数のコンポーネントに分割され，これらのコンポーネントおよびコンポーネント間の接続（ネットリスト）が記述される．各コンポーネントのシステムレベル動作は，ビヘイビア記述を用いて記述できる．この記述に対して機能シミュレーションを実行することにより，システムの機能確認が行われる．

(2) 機能設計（RTL設計）

ここでは，レジスタおよび組合せ回路でシステムを表現する．これをレジスタトランスファ（RT）レベル記述という．上記のシステムレベルでの設計記述に基づいて，RTレベルの機能設計が行われる．RTレベルでの設計記述では，コンポーネントの接続記述とコンポーネントの動作記述から構成される．しかし，システムレベルでの記述と異なるのは，それぞれのコンポーネントの記述レベルがビヘイビアではなくRTである点である．RT記述では，レジスタおよびレジスタ間での演算とデータフローが記述され，設計が具体化され始める．

(3) 論理設計

ここでは，RTレベル記述をさらに詳細化し，ゲートレベルの論理回路を構成する．上記のRTレベルの設計記述を論理合成系に入力し論理合成を行うことによって，ゲートレベルの設計を得ることが可能である．論理合成の作業は，図2の「論理設計」のステップで実行される．論理合成を行うことにより，コンポーネント間の配線（ネットリスト）が生成される．

(4) レイアウト設計

ここでは，セルの設計仕様，回路やレイアウト情報，電気的特性などが登録されたデータベースであるライブラリーからゲートや論理セルのレイアウト情報を取り出し，チップ上の配置・配線を決定する．

(5) マスクパターン

VLSIの回路を形成するために必要なフォトマスクのマスクパターンデータは，CADで設計された図面をマスクの製造データに変換する過程と変換結果を自己検証する過程を経て作成される．近年，設計データ量の増加に加え，より微細な配線パターンが要求されており，このCAD図面からのデータ変換ならびにフォトマスクの信頼性を保証するためのデータ検証の処理時間が増大する傾向にある．

各設計工程のCADツールは，設計合成するものと結果検証するものと2つに大別できる．検証ツールは早くから実用化されたが，合成ツールは不完全であった．近年のVLSI大規模化に伴い，トップダウン設計手法が標準となり，上流設計での合成ツールの開発が進んでいる．そこでのCADツールは，フロアプランナがチップ面積最小化のためのレイアウト設計ツールであったが，配線遅延解析のために論理設計や機能設計で利用される．さらに上流設計でのハードウエア／ソフトウエアのシステム分割では，システムの性能，消費電力，コスト，テスト易化などを考慮する必要がある．このように，システムLSI設計におけるCADは，システムレベル設計，論理合成，レイアウト設計の各設計技術の統合化が必要である．

3. 設計検証ツール

各設計工程を通して設計検証をVLSIの上流設計においてシミュレーションによって行うことは不可欠である．これには論理回路の検証，スイッチングおよび配線の遅延時間を考慮した動作の確認というような目的が含まれている．しかし，最近のVLSI設計での設計検証シミュレーションは，その大規模・複雑さから，通常のシミュレーションだけでは，設計の誤りをすべて削除することは不可能になりつつある．事実，全設計期間の内，70～80%は設計ではなく，設計検証作業に要する期間となる傾向である．このため，設計の正確さを数学的に証明する形式的検証（Formal Verification）技術が重要となることが予想される．以下に設計の各ステップで行われる検証を述べる．

(1) システムレベル設計検証

システムレベルでの設計検証は，遅延時間を考慮しない設計対象のスタティックな論理機能の確認である．設計生産性を向上させるためには，設計のイタレーション回数を減らすことが必須である．設計のイタレーションが生ずる原因はさまざまであるが，設計された回路機能が与えられた機能仕様と違う場合には，設計の初期段階まで戻って設計の修正を行う必要が発生する．このイタレーションを防ぐには，設計の上流での機能検証が必須である．

(2) 機能設計検証

最近，ESDA（電子システム設計自動化）ツール

と呼ばれる RT レベルデータ生成用のグラフィカル入力ツールが出てきており，機能仕様から RT レベルへ落とす際，とくにフォーマットなど熟知していなくとも，ステート・マシンのグラフやフローチャートを入力してやれば，論理合成可能な RT レベルソースに落としてくれる．また，そのツール上でシミュレーションも行え，テスト・パターンをグラフィカルに作成し，下流工程で利用することも可能である．

RT レベルのソースが得られると，チップ上で占める面積の見積りが得られる．RT レベルで仮想遅延時間を想定するためには，仮想配線モデルを用いるのが一般的である．しかし一般に設計の下流工程でレイアウト設計を行うことにより，配線遅延に関する高精度の情報が得られる．そこで，この遅延時間に関する情報を RT レベルにバックアノテーションすることにより，より実遅延時間に近い値を用いた解析が可能になる．

(3) 論理設計検証

論理設計検証の目的は，主に遅延時間を考慮した機能の確認である．RT レベルでの遅延時間の解析と同様に，下流工程でのレイアウト設計の結果から得られる配線遅延に関するより高精度の情報を反映させることにより，ゲートレベルにおいても遅延時間を考慮した動的な機能確認を行うことができる．

(4) レイアウト設計検証

論理，回路シミュレーションを行い検証した回路接続情報に従い VLSI 回路パターン（シリコンウェハ上に配置するトランジスタ，抵抗，コンデンサ等）を配置し，VLSI 回路パターン間を配線処理しレイアウトを設計規則に従って作成する．この設計規則は，ツールに予め設定することにより，特別意識しなくてもレイアウト設計が可能になっている場合が多い．近年，プロセスの微細化などに伴って，付加的なレイヤー（位相シフトマスクのためのレイヤーなど）が必要となる場合もある．

4. CAD シミュレーション技術

4.1 仕様シミュレーション

システム LSI のシステムレベルでの仕様シミュレーションは，システム全体の仕様を確認することを目的としている．よって，システム内部の記憶素子（レジスタ，メモリなど）に入力される信号が，これらの記憶素子のセットアップ時間とホールド時間の制約を満たしてさえいれば，各コンポーネント内部の信号値の変化を細かく解析する必要はない．

したがって，システムレベルまたは RT レベルの設計記述の機能のみを確認する場合には，システムクロックが変化（立ち上がり，立ち下がり）した時点の信号値だけを考慮してシミュレーションを行えばよい．最近では，プロセス逐次的な動作記述をコンピュータのマシン語にコンパイルして実行する，コンパイル方式のシミュレーション技術が用いられるようになった．

4.2 遅延シミュレーション

論理ゲートの出力の変化にはスイッチング遅延が生じる．また，信号が配線を伝播するときにも伝播遅延が生じる．このような遅延時間を伴うイベントの連鎖反応の結果を正しく解析するためには，シミュレータはこれらのイベントをそれらが生起する順に時間軸上にならべてそれらの因果関係を調べる必要がある．

イベントドリブンシミュレータは，その時点でのシミュレーション時刻よりも将来に生起するイベントの時間管理を行っている．そのため，システムに生起するイベント数が多ければ多いほど，また遅延時間の分解能を高くすればするほど，シミュレーションの処理時間が増大することになる．最近，システム LSI の回路規模が大規模・複雑化により，遅延シミュレーション技術が重要となってきている．

4.3 コシミュレーション技術

一般にシステム LSI では，アーキテクチャの自由度が向上する．一方で設計は複雑になり，それに対応した新設計手法が求められており，ハードウエア／ソフトウエアの協調設計・検証が注目されている．

協調設計とは，ハードウエアとソフトウエアの最適化設計手法である．この目的は，設計の早期段階でのハードウエアとソフトウエアの最適分割と TAT の短縮にある．

従来のシステム LSI の開発では，ハードウエア試作後にソフトウエアを含めたシステム検証が行われる．この段階で不具合が検出されても，ハードウエアのフィードバックは困難であり，ソフトウエアの工夫だけでは品質改善に限界がある．

現在，協調設計環境の構築は積極的に取り組まれているが，現状は設計者の経験でハードウエアとソフトウエアの機能分割を行い，コシミュレーションや評価ボードなどの協調検証ツールによって配分を

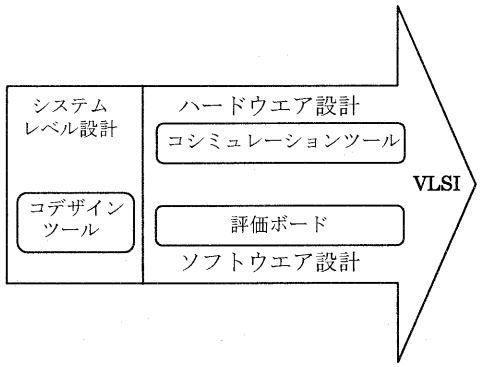

図 3. 協調設計

評価している．ハードウエア製造前にシステム上の不具合が検出できるので，品質改善のみならず TAT の短縮も実現できる（図 3）．

しかし，協調検証ではある程度設計が進んだ状態でないと検証が意味を持たない．システムの仕様を C 言語で記述しておき，インタラクティブに機能分割の結果を評価し，この記述からの HDL とオブジェクトコードを自動生成するといったことを実現する方向に進んでいる．

システム LSI の機能で，ソフトウエアで実現する機能はプロセッサで実行され，ハードウエアで実現する機能は専用ハードウエア回路で実行される[4]．これらの実現方法の異なる機能の確認を行う場合には，従来の論理シミュレーションの手法だけでは不十分である[5]．

なぜなら，ソフトウエアのシミュレーションは高速に実行できるが，専用ハードウエアのシミュレーション速度は，2 桁から 3 桁遅い．システム LSI の機能検証を効率的に行うためには，このようなシミュレーション速度の違いを何らかの方法で埋める必要がある．

これに対して，最近では応用ソフトウエア全体の検証に適したシミュレーションツールが発表されている．それでは，モデル化の抽象度をより高めることにより，20MIPS 程度のシミュレーション速度を達成している．

5. C ベース設計手法

一般に，システムレベル設計者は，C/C++をベースとした環境でシステムの定義と解析を行い，システムの動作を確定後，実行可能なシステム仕様をハードウエア設計者とソフトウエア設計者に渡す．ハードウエア設計者は，C/C++で書かれている実行可能な仕様を HDL に変換して，さらに詳細記述を追加しながら，IC をインプリメントするためのゲートレベル・ネットリストに合成可能な HDL コードを完成させている．しかしこの方法は C/C++コードを同等の HDL 記述に書き換える作業に時間がかかる上，人為ミスも発生しやすいという問題がある．また通常は C/C++で書かれたテストベンチを HDL に書き換える作業も伴うため，作業量が非常に多くなる．さらにハードウエア設計過程でシステム全体の動作に影響するトレードオフを試そうとした場合には，異なる代替デザインを C/C++の実行可能な仕様全体のなかで解析しなければならないため，再びシステムレベル設計者が関与しなければならない．

この問題を解決するためには，システムレベル設計者が作成した C/C++の実行可能な仕様を，HDL に書き換えるのではなく，ハードウエア合成ツールへの入力として利用できる形式までハードウエア設計者自身が詳細化できるスムーズで信頼性の高い手法が必要である．

C/C++から HDL への変換を不要にすることにより，C/C++で書かれたテストベンチを再利用できる．この結果，検証時間が短縮され，元の仕様へ準拠していることが保証される．

よって，C ベース言語でシステム LSI を設計することにより，RT レベルモデルに比べ，1000～100 倍の速さでシミュレーションが可能で検証にかける工数が削減できる．それにより，全体の工数が削減でき，早期のバグ発見，修正，検討が可能となる．

6. おわりに

現在，ハードウエアの記述と応用ソフトウエアの記述を，共通の C ベース言語を用いて行う傾向が進んでいる．コデザイン環境では，ハードウエアとソフトウエアを区別せず C ベース言語でシステムの機能を記述しておき，このシステム記述から論理合成可能な HDL 記述と応用プログラムのオブジェクトコードを生成する．システム仕様記述言語は C ベース言語が主流であり，主な言語に SystemC, SpecC, CycleC, Superlog などがあるが，今のところどのが標準になるかがまだ不明確な段階である．とくに，現在の RT レベル設計からの移行の過渡期においては，仕様記述と RT レベル記述が混在するような設計となることが予想される．このような状況では高位合成ツールは導入しづらいのが事実である．

また，現在 IP（設計資産）の標準化や製品化，流通や販売，再利用のためのメソドロジ確立，そして包括的なシステム LSI 設計環境の構築が求められている．IP を利用する段階において IP 単体では検証済みだが，システム LSI に統合されたときにシステム全体としての動作を検証する必要がある．つまり，IP 再利用に基づく設計手法では，機能検証をシミュレーションするだけでは不十分である．

大規模システム LSI を短期間で開発するには，スタンダードな機能ブロックは IP を有効利用し，製品差別化を決定づけるコア・コンピタンスとなる部分に設計リソースを効率的に投下していく必要がある．IP 再利用に関するノウハウと IP コアの再利用性を評価する高精度の手法の確立が必要がある．

文　献

[1] 日本電子機械工業会 EDA 技術委員会 EDA ビジョン研究会，“2002 年 EDA 技術ロードマップ，” 1998.

[2] Semiconductor Industry Association, “ The National Technology Roadmap for Semiconductors-Technology Needs, ” 3rd Edition, 1998.

[3] 木村晃子，“LSI 設計の生産性を 21%/年から 58/%年へ改善” 日経マイクロデバイス，1999 年 7 月号，p58-63, 1999.

[4] R.Gupta, C.C.Jr., G..De Micheli : “Synthesis and Simulation of Digital Systems Containing Interacting Hardware and Softwarre Components”, in Proceedings of the 29th Design Automation Conference (DAC), pp. 225-230, 1992.

[5] J.Staunstrup, W.Wolf : “Hardware/Software Co-Design: Principles and Practice”, Kluwer Academic Publishers, 1997.