■10 群（集積回路）－ 3 編（システムオンチップ技術）

5 章 設計検証技術

（執筆者：枝廣正人）［2009 年 9 月 受領］

■概要■

本章ではシステムオンチップ（SoC）における設計技術を扱う．ただしシステムオンチップといっても大規模 LSI の設計方法論と大きく変わることはない．そのため，基本的な設計検証技術に関しては 10 群 1 編などを参照されたい．

システムオンチップに特有の問題として，複数 IP コア間の検証の問題がある．IP コアは，それぞれ異なるベンダで設計開発されていることも多く，同じバスインタフェースを持っていたとしても仕様細部の解釈の違いから簡単につながらないことも多い．複数 LSI の場合にはプリント配線板上においてロジックアナライザなどを用いて不具合解析を行うことが可能であるが，LSI の内部に配線が入ってしまう SoC の場合には解析は容易ではない．そのため，SoC 製造前の事前設計検証が重要となる．

また，SoC の性能を十分発揮させるため，SiP などの高度な実装技術を適用する場合が多いため，その目的を達成するために実装と LSI との協調設計も重要である．更に同じ LSI 上に様々な機能をもつ IP が混在することによる，テスト，ノイズ問題の考慮なども不可欠である．

【本章の構成】

本章では IP ベース設計（5-1 節），実装・LSI 協調設計（5-2 節），デバッグ（5-3 節），テスト（5-4 節），ノイズ・オンチップ波形（5-5 節）について述べる．ただし，テストに関しては 1 編 3 章 3-7-5「システムオンチップのテスト」において説明されているため，1 編 3 章 3-7-5 にリンクする．

■10 群 － 3 編 － 5 章

5-1 IP ベース設計

（執筆者：枝廣正人）［2009 年 9 月 受領］

システムオンチップ（SoC）は IP コア（プロセッサ，メモリ，周辺デバイスなどが LSI 上の回路モジュールになったもの）をバスなどによって接続した構成と考えることができるが，IP コア自体の設計と，SoC 全体の統合設計の階層構造を利用して設計する方法を IP ベース設計という．IP コアは，IP コアベンダから購入することも多く，その場合ベンダのノウハウを流出させないため，SoC 設計開発者にとってはブラックボックスであることが多いため，IP ベース設計は重要である．本節では IP ベース設計の流れについて述べる．

5-1-1 SoC のモデルと設計検証

SoC の簡単なモデルは本編の概要にて示したとおりである（**図 1**）．各 IP コアの内部は単体で設計検証されるとすると，SoC 全体から見て IP コアはバスに対する入出力関係として見える．特に IP コアは機能ブロックであるので，SoC のなかで定められた機能を実現する．したがって，単なる入出力関係だけでなく，定められた機能に対する設計検証，すなわち性能や SoC 内外のほかのリソースに対する要求（バスのバンド幅，メモリ容量など）も重要になってくる．

最初に述べたように，IP ベース設計の特徴の一つは，ノウハウ保護のため IP コアがブラックボックスとして与えられることである．しかしながら，ブラックボックスだからといって事前検証を怠り，SoC 開発後に不具合がわかり，再設計することは多額の費用が必要となるため可能な限り避ける必要がある．そのため事前検証のためのモデル（シミュレーションモデルとも呼ばれる）が与えられることが多い．言い換えれば，十分な事前検証を可能にするような IP コアが IP ベース設計としては望まれるということである．

事前検証は，大きく分けて機能要件，非機能要件，バスインタフェースがある．当然ながら IP コア内部に関しても事前単体検証は必要であるが，一般の集積回路に対する設計検証と同様であるため，10 群 1 編などを参照されたい．

5-1-2 機能要件の事前設計検証

まずは機能要件に対する設計検証について述べる．事前検証のために与えられるモデルとしては次のような種類がある．それぞれのモデルに対する設計検証手法については本章 5-3 節を参照されたい．

(1)　プログラム言語レベル（例えばＣ言語レベル）

近年の SoC では，設計の初期段階において C 言語などを用いて論理的な動作を表現し，検証することが多く行われるため，IP コアに対しても，その論理的な動作と等価な C 言語モデルが与えられることが理想的である．

IP コアが実現する個々の機能を関数インタフェース（API）としてもつようなライブラリとして提供し，C 言語などから関数呼び出しとして IP コアの論理的な動作を表現するような方法もある．

例えばプロセッサの命令セットシミュレータ（ISS: Instruction Set Simulator）やサイクルレ

ベルシミュレータなどがある．

(2) トランザクションレベル

プログラム言語レベルよりもハードウェアに近い表現としてトランザクションレベル（TLM（Transaction-Level Modeling）とも呼ばれる）がある．これは IP コアに対する論理的な入出力関係を一つの「トランザクション」とし，回路的な信号レベルの入出力や内部動作はブラックボックス（正確に内部状態をモデル化していないかも知れない）として，機能レベルの入出力関係のみを正確に表現することにより，IP ベース設計を容易にする．

(3) レジスタトランスファレベル

IP コア自体はレジスタトランスファレベルで与えられることが多いので，これはシミュレーションモデルというよりは「そのまま」である．

プログラムレベルやトランザクションレベルのモデルは，IP コアの内部が抽象化されているため動作速度も速く，SoC の機能を検証するためには好適である．しかしながらこれらのモデルから自動的に合成されたハードウェアでない限り，ハードウェアの機能・動作とは完全に一致しないリスクを常に抱えている．また，内部が抽象化されているため，タイミングなどは保証されておらず，それに対しては別途検証が必要となる．

これに対してレジスタトランスファレベルはハードウェアと完全に一致していることが多く正確性は期待できるが，動作速度が遅く複雑な機能，多数の状態をもつような SoC の機能検証を行うためには膨大な時間を要する．

5-1-3 非機能要件の事前設計検証

機能要件は満足する場合にも，バス混雑やメモリ使用量，リアルタイム性能などの非機能要件についても設計検証を行う必要がある．これに対しても，上記の機能レベルの設計検証において，バスへの入出力の監視，メモリ間のデータ移動の監視，プロセッサにおける実行命令数（実行サイクル数）の監視などによって行われる．

5-1-4 バスインタフェースの事前設計検証

機能要件，非機能要件の設計検証において抽象化されたモデルを用いている場合は特に，回路的な信号レベルの設計検証が必要である．特に標準的なバスを利用している IP コアだとしても，一般にバスの仕様はオプションも多く，コスト面の理由などから SoC 内で利用する場合にすべての仕様を盛り込むことは少ないため，IP コアが仮定しているバスの仕様と SoC で実装されるバスの仕様に不一致や仕様解釈上の差異が起こる可能性がある．

そのための検証モデルとしては，レジスタトランスファレベルが多く用いられるが，バスシミュレータなども使われる

■10 群 － 3 編 － 5 章

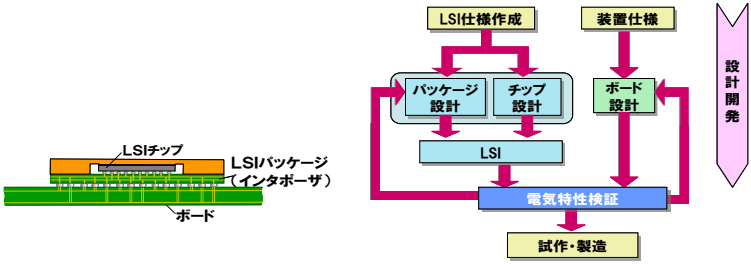
5-2 実装･LSI 協調設計

（執筆者：原田高志）［2009 年 9 月 受領］

コンピュータや携帯機器などの電子機器の性能向上を技術的に支えているのが LSI の高速化，LSI パッケージやボード（プリント配線板）における実装の高密度化である．しかしながら，こうした技術の発展に反比例するように実装レイアウトの自由度は低下し，設計は難しいものになっている．例えば，10 Gbps の信号伝送では，信号の立ち上がり時間は数十 ps（ピコ秒）程度になる．電気信号の伝搬には 1 mm あたり約 5~7 ps を要するので，信号配線長をミリ以下の単位で管理しないと信号のタイミングがずれ，正常な回路動作を妨げる．ボード上の配線は勿論のこと，LSI パッケージ（インタポーザ）の配線レイアウトやピンアサインを考慮した設計が重要となるゆえんである．また，ディジタル回路のスイッチングはチップへの電荷供給によって行われる．この電荷はチップ内部や LSI 周辺のキャパシタから供給されるため，スイッチング速度の速い回路ではボード上の LSI の極近傍に十分な容量のキャパシタを配置する必要がある．しかし，この領域は信号配線も集中し，思うようにキャパシタを配置できないため，近年ではボードに合せて LSI のピン配置を決定するようなケースも生じている．こうした課題を解決するため，LSI やボードの最適なレイアウトを設計自由度の高い開発初期段階で求める必要がある．この要求を実現するのが LSI チップとパッケージ，そして LSI を実装するボードのすべての特性を考慮した協調設計である．

5-2-1 協調設計の課題

従来，電子機器の開発においては，**図 5･1** に示すように機器の性能や配線の数，消費電力などの特性を考慮して，チップ設計とパッケージ（インタポーザ）設計が別々に行われ，両方の設計が完了した時点でチップとパッケージを組み合わせて LSI 仕様を決定し，更にこの仕様を考慮してボード設計を行った後，電気的特性などに問題がないかの検証・評価がなされてきた．この流れではそれぞれの設計が個別に施されるため，LSI とボードを組み合わせる最終段階まで特性を確認できず，この段階で問題が発見されると，パッケージやボードのレイアウト，もしくは LSI の仕様まで遡って設計のやり直し（Rework）をしなければならない．この Rework が電子機器の開発期間を長期化させる原因の一つとなっている．年々向上する半導体チップの性能を最大限に発揮し，かつ開発期間を短縮して最先端の製品を早く市場に提供していくためには，LSI パッケージやボードを統合した設計をどのように展開していくかが重要な課題である．



（a）代表的な実装構造　　（b）設計フロー

図 5･1　LSI チップ，LSI パッケージ，ボードで構成される実装構造との設計フロー

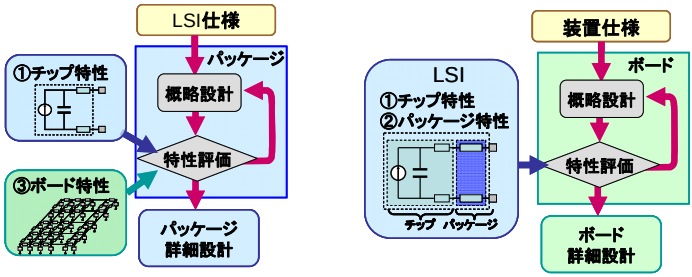
5-2-2 シミュレーション技術の活用

設計上流段階では LSI パッケージ，ボードとも，そのレイアウトの詳細は決まっていないため，設計自由度が大きい反面，部品の配置や配線などを一つずつ決める作業と，その都度，特性を把握する作業が必要である．この段階で電気的な特性を知るための手段として，有限要素法や FDTD（Finite Difference Time Domain）法などの電磁界シミュレータや SPICE に代表される回路シミュレータが用いられる．電磁界シミュレータはパッケージやボードの物理的な構造を直接モデル化し，Maxwell の方程式を適用する手法であり，精度が高い反面，計算時間が長くなることが欠点である．近年，コンピュータ性能の向上やパッケージやボードの構造に合わせた 2.5 次元解析手法の適用などにより，解析の短時間化が計られている．

一方，回路シミュレータは物理的な構造を PEEC（Partial Element Equivalent Circuit）[1] 法などを用いて等価回路として表現し，回路解析を行う．計算規模は電磁界シミュレータほど大きくなく，計算時間が短いが，3 次元構造を等価回路に変換するため，誤差が発生しやすい．解析時間を優先するか，解析精度を優先するかはどのような目的（what if 解析のように設計の上流工程に適用するか，設計の最終段階で仕様の確認に用いるかなど）によって異なる．

5-2-3 協調設計システムの例

図 5･2 にディジタル回路の電源系の設計に利用されている協調設計システムの例を示す[2]．このシステムは①LSI チップの電気的な特性，②LSI パッケージの電気的な特性，及び③ボードの電気的な特性をそれぞれ，等価回路モデルに変換し解析するモジュールで構成される．製品開発のスタート段階では LSI チップの機能や性能はおおよそ決まっており，本システムは主に LSI パッケージ，及びボードの設計への適用を目指している．LSI パッケージの設計は①から出力されるチップの情報（等価回路モデル）と③から出力されるボードの情報を用いて設計する．また，ボードの設計は①と②の出力の組み合せである LSI 等価回路モデルを用いて行う．ここで等価回路を用いる理由は，電気特性の解析に広く普及している回路シミュレータ（SPICE）を利用するためである．

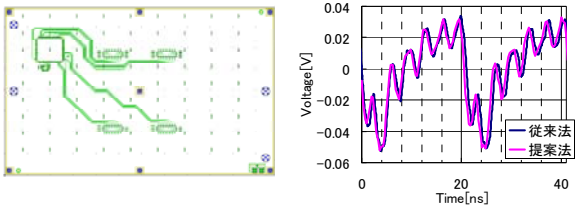


（a）LSI パッケージ設計向け特性評価　　（b）ボード設計向け特性評価

図 5･2　LSI チップ－パッケージ－ボード協調設計システムの構成

以下に，LSI のスイッチングに必要な電荷を供給するためのボードの電源系設計に対し，前述の協調設計システムを適用した例を紹介する．前述のようにボード電源供給系の電源電圧の安定性は回路の正常動作における重要なポイントである．ボード設計の初期段階では，電源形状やキャパシタの容量，配置などの決定のため，パラメータ変更の結果を短時間で解析する手法が求められ，この解析に本システムが用いられる．LSI（LSI チップ，パッケージ）はチップのスイッチング時の電流の振る舞いを表現した電流源とチップ，パッケージ内部の特性を表した等価回路で示される [3]．ボードは 2 次元的な広がりをもつため，そのモデルはインピーダンス Z 及びアドミタンス Y の各要素を組み合わせた 2 次元回路ネットワークとなる [4]．

解析時間の短縮も，本システムに要求される項目である．シミュレーションに掛かる時間を短縮するため，本システムでは従来の時間軸解析に代わって，周波数解析を採用している．従来の解析は LSI のスイッチング時の電流の時間的変化を時系列で逐次計算していた．しかし，この方法では回路動作が安定するまで計算を繰り返すため（約 100 万回以上），1 回の解析に数時間を要する．それに対し，本システムでは周波数軸解析を採用している．これはディジタル回路がクロック周波数に相当する一定の周期で動作を繰り返す特徴を利用したもので，クロックの基本周波数とその 20 倍（20 次高調波）程度までのポイント周波数のみを計算すればよいので，解析時間を大幅に短縮できる．**図 5･3** はボードの電源電圧の解析例である．本手法はチップの電流波形を時間軸から周波数軸に変換するなどのプロセスが入るにもかかわらず，解析に要する時間は数十秒程度であり，大幅な時間短縮を実現している．



（a）評価ボードのレイアウト　　（b）電源電圧変動

図 5･3　評価ボードと電源電圧変動の解析結果の例

■参考文献

1) A.E. Ruehli, “Equivalent Circuit Models for Three-Dimensional Multiconductor Systems,” IEEE Trans on MTT, vol.22, no.3, pp.216-221, 1974.
2) 原田高志・和深裕・楠本学・小川雅寿・佐藤俊二・小菅淳二・島先敏貴, “LSI電源モデルを用いた多層プリント回路基板の高速電源電圧変動解析,” 電子情報通信学会和文論文誌Ｃ, vol.J89-C, no.11, pp.843-853, 2006.
3) N. Matsui, N. Orhanovic, H. Wabuka, “FDTD_SPICE Analysis of EMI and SSO of LSI ICs Using a Full Chip Macro Model,” IEEE Int. Symp. on EMC, pp.99-104, 2002.
4) J. Kim, M. Swaminathan, “Modeling of Multilayered Power Planes Using Effects Using Transmission Matrix Method,” IEEE Trans. on AP, vol.25, no.2, pp.189-199, 2002.

■10 群 － 3 編 － 5 章

5-3 デバッグ

（執筆者：枝廣正人）［2009 年 9 月 受領］

本章 5-1 節で述べたようにシステムオンチップ（SoC）では事前検証が重要である．SoC 製造後も不具合があればデバッグしていくことになるが，この方法には大きく分けて二種類ある．実際の SoC を搭載したボード（評価ボードや実機ボードと呼ばれることが多い）を用いる場合と，事前検証と同じ環境を使う場合である．基本的なデバッグ手法についてはシステムオンチップにおいても通常の LSI と違いはないが，ここでは検証環境に関して IP ベース設計の特徴的な部分について述べる．

IP ベース設計において重要な特徴は，

- ✓ IP コアのなかにはブラックボックスのものがあり，SoC チップにおいても IP コアの入出力のみを監視することになる（この入出力には，ソフトウェアにおいて API として定義されている入出力関係も含む）．
- ✓ IP コアのなかには，ハードウェアとしてはブラックボックスであるが，ソフトウェアのシミュレータ（シミュレーションモデル）が提供されているものがある．プロセッサなどではデバッガなどのツールも提供される場合がある．

といったことである．

このような IP コアを含む SoC に対するデバッグ手法は大きな課題である．本節では，事前検証も含めいくつかの検証環境，検証手法について説明する．

SoC の簡単なデバッグ手法を**図 5･4** に示す．SoC のシミュレーションモデルもしくは SoC チップ（ターゲットと呼ぶ）にデバッガを接続してデバッグを行う．このとき最初に述べたように IP コアによって提供される環境が異なるため，それに応じたデバッグを行う必要がある．多くの場合には，SoC 全体のなかで制御の中心となるホストプロセッサがあるので，デバッグもホストプロセッサを中心として行われることが多い．

デバッガは，ブレークやトレースの機能があり，SoC 内のレジスタやメモリ，入出力の値を読み書きできる．上述したように様々なモデルが考えられるが，どのようなモデルでも同じユーザインタフェースで使えるように，デバッガとシミュレーションモデルの間のインタフェースを共通化するような活動もある．

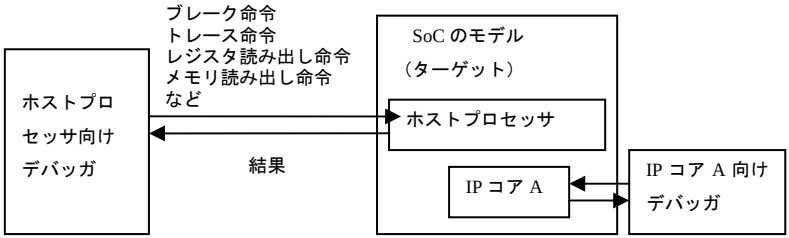


図 5･4　SoC デバッグモデル

以下では，設計段階に応じたデバッグの特徴を述べる．

5-3-1 設計初期段階でのデバッグ

SoC 設計の初期段階では，多くの場合には C 言語などを用いて SoC のシミュレーションモデルをソフトウェアで構築し，機能を検証する．この場合，図 5･4 においてターゲットのモデルはソフトウェア，デバッガもソフトウェアとなる．

いずれもソフトウェアであるため，ソースコードがあれば基本的にはすべての変数等にアクセスすることは可能となるが，IP コアによってはソースコードとして提供されない場合もあり，それぞれの対応が必要である．

(1) ソフトウェアシミュレーションモデルのライブラリによる提供

ソフトウェアシミュレーションモデルがライブラリとして提供される場合，ライブラリの API に従ったソースコードを書き，リンクして使うことになる．あるいは，CPU モデルのような場合，ライブラリとともに CPU のデバッガも同時に提供されることがあり，そのような場合には CPU のソフトウェアモデルが中心となり，SoC のほかの IP コアモデルは CPU の周辺回路として，ライブラリに定められた記述方法で記載する必要がある場合もある．しかしながらマルチコアの時代になり，SoC 内に複数のプロセッサが用いられる時代には，一つのプロセッサのソフトウェアモデルを中心として記載することが難しくなってきている．そのため，SoC のソフトウェアモデルとしてはバスなどの接続を中心として記載されることが多くなり，IP コアのソフトウェアモデルもバスに対する読み書きのトランザクションをモデル化したものが多くなってきている．これをトランザクションモデル（TLM: Transaction Level Modeling）と呼ぶ．

ライブラリとして提供されるソフトウェアシミュレーションモデルとしては，CPU の命令セットシミュレータ（ISS: Instruction Set Simulator），サイクルベースシミュレータ，トランザクションモデルなどが代表的である．ISS は命令カウント（各命令に要するクロック数を含む場合もある），サイクルベースシミュレータはクロック数の情報が提供されるため，ある程度の性能検証が行える．命令セットシミュレータよりもさらに高速にするためにプログラマーズ・ビュー（PV）モデルとよばれるシミュレーションモデルもある．

(2) アクセス関数ライブラリによる提供

シミュレーションモデルではなく，SoC 上で動作するソフトウェアから IP コアへアクセスするためのアクセス関数のみが与えられる場合もある．この場合には SoC のシミュレーションモデルのなかには IP コアのモデルを入れることはできず，SoC 上で動作するソフトウェアからアクセス関数を呼び出すことになる．そのため，命令数やクロック数などの情報がなく，機能面の検証のみ可能である場合が多い．

(3) ハードウェア記述言語（RTL モデルなど）による提供

IP コアによってはソフトウェアのモデルが提供されないものもある．ハードウェア記述言語として与えられた場合には，ハードウェア記述言語に対応したシミュレータと接続してデバッグ環境を動作させることができるが，一般にハードウェア記述言語のシミュレータはソフトウェアモデルと比較して非常に遅く，SoC 全体の機能検証などの目的には現実的でない場合もある．そのような場合には下で説明するハイブリッドエミュレーションなどが使われる．

また，ハードウェア記述言語も与えられない場合には，デバッグ環境を動作させるための最低限のソフトウェアモデルが必要となる．その場合，IP コアへの入力に対して何らかの出力をするようなシミュレーションモデルを構築することになるが，一般に正確性は劣り，限定的な使用方法となる．

5-3-2 設計中後期段階

設計の中後期段階では，ソフトウェアモデルで記載された部分がハードウェア記述言語（例えばレジスタトランスファレベル言語（RTL: Register Transfer Level））によって記載される．これにより，実際にチップレイアウトまで考えることができるようになり，タイミング検証もできるようになる．

ハードウェア記述言語に対応したシミュレータを用いることにより，論理的な検証やタイミングの検証を行うことができる．また，SoC としては IP コアとバスの論理的な関係（バスプロトコルなど）についての検証も行う必要があり，バスシミュレータと呼ばれるツールが使われることもある．

更に，非機能要件の検証も行う必要があり，バスの混雑度などの性能なども検証していく必要がある．ただし，SoC 全体の動作検証や性能検証を行うためには，OS を起動し，想定された利用シーンに合わせてアプリケーションを実行させる必要がある場合がある場合が多く，ハードウェア記述言語レベルのシミュレーションでは現実的な時間内には終わらない可能性がある．そのような場合には，ハードウェア記述言語を FPGA 上に実装し，動作を高速化する方法があり，エミュレーションと呼ばれる．設計の中期段階ではすべての IP コアがハードウェア記述言語で記載されておらず，一部がソフトウェアモデルとして残っている状況がある．そのような場合に，ソフトウェアシミュレータとエミュレータを協調動作させる手法があり，ハイブリッドエミュレータなどと呼ばれている．

5-3-3 SoC 製造後

既に SoC チップが存在する場合には，SoC チップを搭載した評価ボード（実機ボードなどとも呼ばれる）などが存在する．したがって，その上に実際のソフトウェアを搭載し，動作させることにより機能や性能の検証を行うことができる．SoC チップ上で実際のソフトウェアを実行させる時間と比較し，エミュレーション環境においては数十倍以上遅く，ソフトウェア環境では更に遅くなるため，実機ボード上で可能な限りデバッグすることが効率が良い．

実機ボード上でのデバッグは，アプリケーションソフトウェアであれば，SoC 上のソフトウェア環境のデバッガ（例えば gdb）などでデバッグを行う．OS やデバイスドライバなどの基本ソフトウェアであれば ICE（In-Circuit Emulator）などを用いてデバッグする．これらのデバッグにより，不具合原因がソフトウェアではなくハードウェアであることが想定される場合には，ボード上でのロジックアナライザなどを用いたデバッグや，上述のシミュレーション環境を用いたデバッグを行う．

■10 群 － 3 編 － 5 章

5-4 テスト

（※10 群 1 編 3 章 3-7-5『システムオンチップのテスト』より）

（執筆者：米田友和）［2008 年 6 月 受領］

システムオンチップ設計では，IP コアベース設計が用いられることが多い．IP コアは，プロセッサ回路，メモリ回路，アナログ回路など様々である．それゆえに，用いられるテスト容易化設計方式も IP コア毎に異なり，テストパターンは各 IP コアごとに用意される．このような IP コアが組み込まれた IP コアベースシステムオンチップでは，新たなテスト問題に対処する必要が生ずる．それらは主に，テストアクセス機構の設計，コアラッパの設計，及び，テストスケジューリングである．

(1) テストアクセス機構設計

IP コアがシステムオンチップに組み込まれると，システムオンチップの外部入力から，IP コアに用意されたテストパターンを IP コアの入力まで伝搬し，更に，そのテストパターンに対する IP コアのテスト応答をシステムオンチップの外部出力に伝搬することは困難となる．そこで，各 IP コアに対して，用意されたテストパターン，及び，そのテスト応答の伝搬を実現するテストアクセス機構の設計が必要となる．テストアクセス機構の実現方式としては，テスト用に付加したハードウェアのみで実現するテストバス方式，テストレイル方式，境界スキャン方式，既存のハードウェアの一部をテスト時に再利用して実現する透明経路方式などが提案されている．

(2) コアラッパ設計

テストアクセス機構と IP コアのインタフェース回路となるのがコアラッパであり，IEEE Standard 1500 として標準化されている．コアラッパは IP コアを包み込むように設計され，通常動作とテスト動作で IP コアの入出力の接続先を切り替える機能をもつ．標準化では，通常動作（BYPASS），コアテスト（INTEST），コア外部テスト（EXTEST）の三つが必須動作として規定されている．

(3) テストスケジューリング

各 IP コアに対して，テストアクセス機構とコアラッパを設計することでシステムオンチップのテストは可能となる．しかしながら，これらの設計とシステムオンチップのテスト時間には密接な関係があり，テスト時間を考慮してテストアクセス機構とコアラッパを設計することが重要となる．例えば，システムオンチップの外部入力から同時に転送できるテストデータのビット幅はテストアクセス機構に依存する．また，同時にテストする IP コアのラッパに入力されるテストデータのビット幅の総和は，テストアクセス機構のビット幅を超えることはできない．これらの制約を考慮し，テスト時間の最小化を目的として，同時にテストする IP コアの組合せや順序を決定することをテストスケジューリングという．また，テスト時の消費電力は，通常動作時の 2〜3 倍であることが知られており，テストスケジューリングでは，テスト時の消費電力制約を考慮することも重要である．

■10 群 － 3 編 － 5 章

5-5 ノイズ

（執筆者：帰山隼一）［2009 年 9 月 受領］

SoC の設計では，LSI 内部で発生する電源ノイズやデバイスノイズなど，様々なノイズの影響を考慮しなければならない．これらのノイズは論理回路の誤動作，通信回路のビットエラーレートの悪化，映像や音声などのアナログ情報の劣化などを引き起こす．本節では，様々なノイズの種類について概略を説明し，そのなかで特に大きな影響を及ぼす電源ノイズについて，その発生過程と論理回路への影響を説明する．また，電源ノイズへの対策，電源ノイズの観測について，具体的な方法を紹介する．

5-5-1 ノイズの種類

LSI 内部で発生するノイズには様々なものがあり，発生源や回路動作への影響，対策の方法が異なる（**表 5･1**）．電源ノイズは LSI 内部で消費される電流の変化によって生ずる電源電圧の変動のことで，近年のシステム LSI では最も大きなノイズ源である．電源ノイズはディジタル回路とアナログ回路の両方の動作に影響を及ぼすので，本節で詳細を説明する．

表 5･1　ノイズの種類

ノイズの種類	主な発生源	主な影響
電源ノイズ	LSI が発生する電流変動 スイッチング電流のリップル	遅延変動 発振器・クロック分配ツリーが発生するジッタ ADC などの量子化雑音（画像・音声などのノイズ）
デバイスノイズ	熱ノイズ フリッカーノイズ	発振器が発生するジッタ ADC などの量子化雑音（画像・音声などのノイズ） 無線などの隣接チャネル干渉
クロストークノイズ 反射ノイズ	配　線	論理値の誤判定
放射電磁ノイズ（EMI）	周辺部品・周辺機器	論理値の誤判定 画像・音声などのノイズ

熱ノイズやフリッカーノイズなどのデバイスが発生するノイズは，ガウス分布を有する不規則なランダム雑音である [1]．このランダム雑音の振幅は論理振幅に比べて十分に小さいので，ディジタル回路の動作には影響しないが，発振器や ADC・DAC などのアナログ回路には無視できない影響を及ぼす．デバイスノイズがアナログ回路に及ぼす影響は Spice などの回路シミュレータで検証可能であり，ADC や発振器などのアナログ回路を設計する際には通常この検証を行わなければならない．発振器におけるデバイス雑音の影響は位相ノイズ（ランダムジッタのパワースペクトル）として現れる [2]．位相ノイズは SerDes などの高速 I/O ではタイミングマージンを減少させ，無線では隣接チャネル干渉の原因になる．デバイスノイズやその影響を抑制するためには，受動素子やトランジスタのインピーダンスを低く設計する，トランジスタのチャネルを大きく作る，アンプのゲインを最適化する，不要な周波数帯を遮断する，などの対策を講じる．

チップ内に長距離の信号配線を用いる場合や，長距離配線に高速信号を伝播させる場合には，クロストークや反射によるノイズ，信号の減衰によるシンボル間干渉も考慮しなければならない [3]．また，周辺回路から飛来する放射電磁ノイズ（EMI）も誤動作を引き起こす場合がある．EMI は主にシールドを用いて緩和する．

5-5-2 ジッタ

ノイズには，信号の電圧・電流の揺らぎのほかに，タイミングの揺らぎも含まれる．このタイミングの揺らぎをジッタという．前述のノイズは様々なジッタを引き起こす原因になっている．**図 5･5** はジッタの分類と，それぞれの発生原因をまとめたものである．ランダムジッタは挙動の予測が不可能なジッタで，統計力学的な確率分布しか求められない．デターミニスティックジッタは，Spice などの過渡解析で挙動を予測することが可能なジッタである．

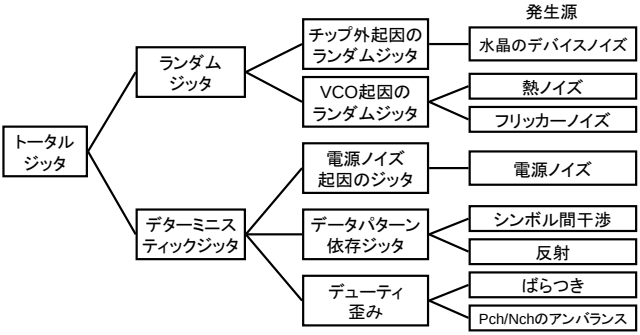


図 5･5　ジッタの分類・ノイズとの関係

ジッタの定義は**図 5･6** に示すように，観測方法の違いによって大きく二つに分けられる．ピリオドジッタは理想的なクロック周期に対する実際のクロック周期の誤差である．論理回路では，クロック周期の変動がセットアップマージンに影響を与えるので，その設計・検証時にはピリオドジッタを考慮しなければならない．クロック周期のジッタにはこのほかに，サイクルトゥーサイクルジッタがある．これは前のクロック周期との差分で，図 5･6 では T_n-$T_{(n-1)}$で表される．タイミングジッタは，理想的なクロックエッジのタイミングに対する，実際のクロックエッジのタイミングのずれの大きさである．タイミングジッタが大きければ，ほかの LSI と接続する I/O 回路や，LSI 内部のクロックドメイン間での通信で信号のタイミングが合わずに誤動作を起こすので，これらの回路を設計する場合には考慮しなければならない．図 5･6 のなかのタイミングジッタの式で，特に n が小さいものをショートタームジッタ，n が大きいものをロングタームジッタと呼ぶ．一般に n が大きくなるとタイミングジッタも大きくなるので，タイミングジッタの大きさを比較する場合にはこの点を注意しなければならない．

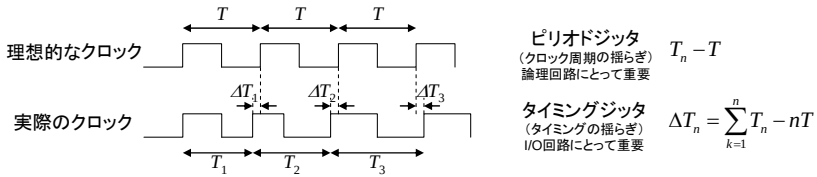


図 5･6　観測方法の違いによるジッタの定義

5-5-3 電源ノイズの発生過程

電源ノイズは**図 5･7** に示すように，LSI 内部の回路が消費する負荷電流 $i(t)$が変化することで生ずる．負荷電流の変動によって生ずる電源電圧の変動は，抵抗による IR ドロップと，インダクタンスによる $L･di/dt$ によるものがある．一般に電源配線の直流抵抗は低く設計されるので，電源電圧の変動は $L･di/dt$ による電圧降下が支配的である．電源電圧が降下すると，電源配線の LRC によって減衰振動を生ずる．LSI 内部の電源配線の LC は一般に数百 MHz の共振周波数を有する．また，LSI が消費する電流の変化によって，プリント基板上の LC も数 MHz 程度の減衰振動を生ずる．

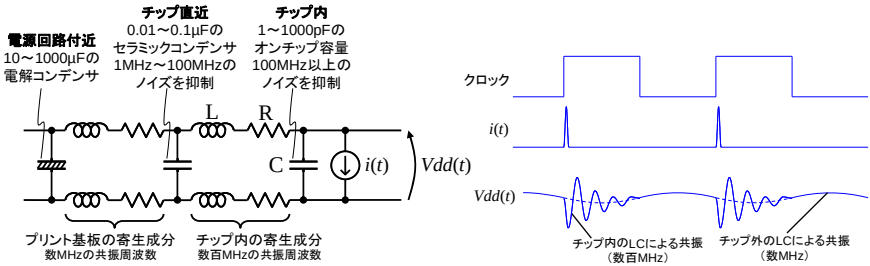


図 5･7　電源ノイズの発生過程

5-5-4 電源ノイズの影響

ディジタル回路が電源ノイズを受けると，論理遅延の変動を生ずる．電源電圧が低下すると論理遅延が大きくなる．フリップフロップのセットアップマージンを満たさない程度にまで遅延が大きく（電源電圧が低く）なると誤動作を起こす（**図 5･8**）．

一般的には，電源電圧が下限に達したときに論理遅延が最も大きくなるので誤動作を起こしやすい．電源ノイズの周波数が高く，クロック周波数も高い場合には，電源電圧が下限から回復する過程で最も誤動作を起こしやすくなる [4]．これはクロック分配ツリーも電源ノイズによって遅延が変動する（ピリオドジッタが発生する）ためである．電源電圧が上昇する過程では，クロック分配ツリーの遅延が徐々に小さくなるので，クロック分配ツリーの末端ではクロックエッジの間隔が短縮する．電源電圧が下限から回復する過程では，論理遅延も大きく，一方でクロック周期が短縮されるので，誤動作を起こしやすくなる．一般に，クロック分配ツリーの遅延は 1 ns～2 ns 程度なので，電源ノイズによるクロック周期の変動は数十 ps～100 ps 程度である．したがって，このクロック周期の変動の影響を受けるのは，主にクロック周波数が 1 GHz 以上の論理回路である．電源ノイズの周波数がクロック周波数と同等かそれ以上に高い場合には，1 クロック周期の間に遅延の大小が平均化されるので，クロック周期の変動や論理遅延の変動は小さくなる．ディジタル回路の設計時には，電源の LRC を加味したシミュレーションを行い，各フリップフロップのセットアップ・ホールドマージンを検証する必要がある．

アナログ回路を設計する場合には，電源電圧変動除去比（Power Supply Rejection Ratio: PSRR）[1]を Spice の AC 解析などを用いて検証する．

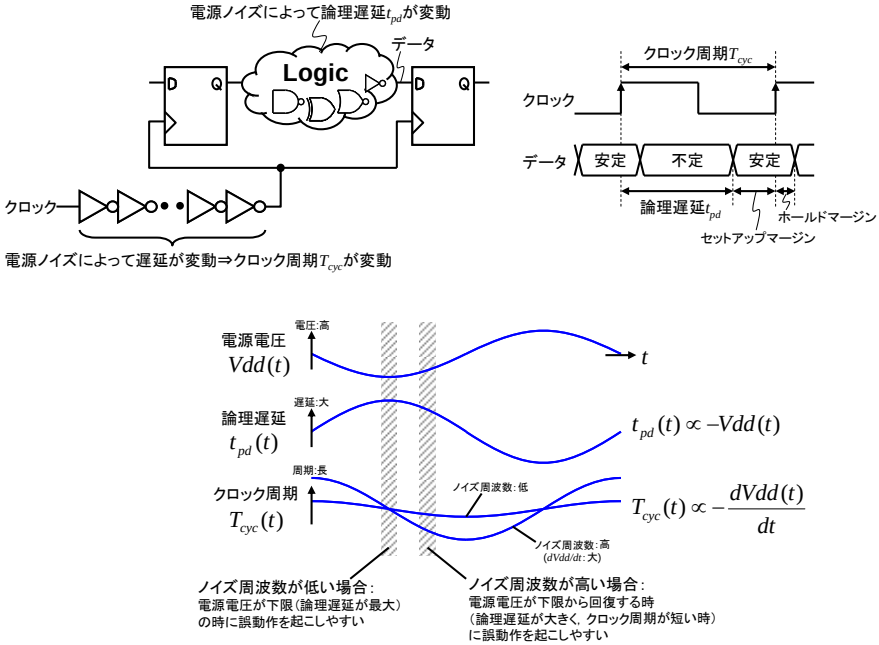


図 5･8　電源ノイズの論理回路への影響

5-5-5 ノイズへの対策

(1)　デカップリング容量

LSI 内部の負荷電流の変化による電源電圧の変動を抑制するためには，電源とグラウンドの間のインピーダンスを下げる必要がある．一般に，電源配線の直流抵抗は十分に低く設計されており，またインダクタンスは主に配線長に依存しているので大幅に削減することが困難である．したがって，電源とグラウンドの間に容量を配置して，交流インピーダンスを下げることが一般に行われる．この容量をデカップリング容量またはバイパス容量と呼ぶ．デカップリング容量は，LSI の直近に配置するセラミックコンデンサと，LSI 内部に配置するオンチップ容量があり，抑制できるノイズ周波数が異なる（図 5･7，**図 5･9**）．容量は近くのインダクタと LC 共振を起こすので，特定の共振周波数でインピーダンスのピークができる．容量が大きくなるとインピーダンスが下がると共に，共振周波数も低下する．オンチップ容量は図 5･9 に示すように，N-well 容量，nMOS 容量，pMOS 容量などで構成される．一般に，容量を大きくするほど電源ノイズの抑制効果は大きくなるが，LSI 内部で容量が大きな面積を占めてしまう．アクティブデカップリングは，ミラー効果 [1)]を用いて，より少ない容量で大きなデカップリング効果を得るものである．例えば 10 pF のデカップリング容量 C_d を配置し，バイアス点付近で-100 倍のゲインを有するアンプを接続することで，実効的に 1010 pF 相当の容量 C_{eff} を得ることができる．

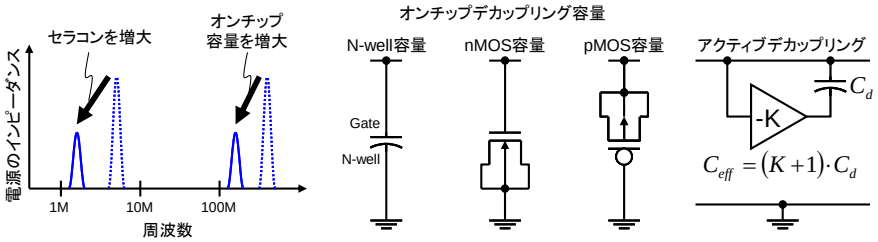


図 5･9　デカップリング容量

(2)　基板ノイズ対策

電源ノイズは LSI の基板を通じても伝播する [5]．例えば，ディジタル回路とアナログ回路の電源を分離して，アナログ回路にはノイズのない電源を供給しても，ディジタル回路が発生するノイズが基板を介してアナログ回路に伝わる．この基板ノイズの影響を緩和するためにガードリング [1] が一般的に用いられる（**図 5･10(a)**）．ガードリングとは，基板ノイズから保護したい回路を取り囲む電源またはグラウンドに接続された拡散層で，基板ノイズの進入を減少させる．また，トリプルウェルプロセスが使用可能な場合には，保護したい回路の下に Deep N-well を配置して，ノイズ源回路と基板を分離する方法もある（**図 5･10(b)**）．ただし，P-sub と N-well 間の接合容量を介して伝播する高周波ノイズも存在するので注意が必要である．

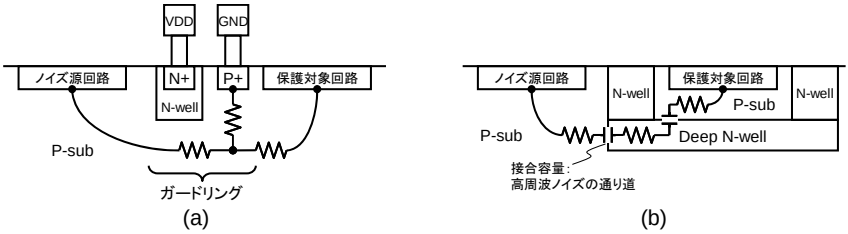


図 5･10　基板ノイズ対策

(3)　オンチップレギュレータ

図 5･11 に示すオンチップレギュレータを用いることで，ディジタル回路で発生する電源ノイズの影響を小さく抑えることができる．ADC や VCO などの電源ノイズに敏感なアナログ回路をディジタル回路などの電源ノイズから保護する場合になどに用いられる．

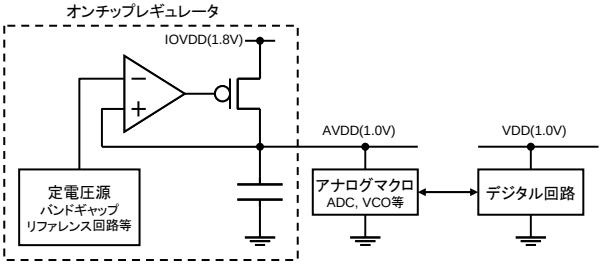


図 5･11　オンチップレギュレータ

(4)　差動回路

上記の（1）～（3）は電源ノイズそのものを抑制する手法を説明したが，電源ノイズがある環境下でもその影響を受け難い回路を設計することも重要である．**図 5･12** に示す差動回路は，電源ノイズを受けることで出力の同相（コモンモード）電圧が変動するが，差動出力の両方が同じように電源ノイズの影響を受けるので，差動出力を差分で観測した場合には電源ノイズの影響を受けにくい．このような差動回路で，差動入力に対するゲインと同相入力に対するゲインの比を表したものが同相信号除去比（Common-Mode Rejection Ratio: CMRR）[1] であり，アナログ回路の評価指標としてよく用いられる．アナログ回路の基本構成ブロックを同相信号除去比の高い差動回路にすることで，電源ノイズの影響を受けにくい回路を構成することができる．また，差動回路は信号振幅や信号の伝播遅延が電源電圧の影響を受けにくいという特徴を有している．このことから，VCO などの発振器や，ジッタに対する要求仕様が厳しいディジタル回路のクロックバッファなどでも差動回路が用いられる場合がある．

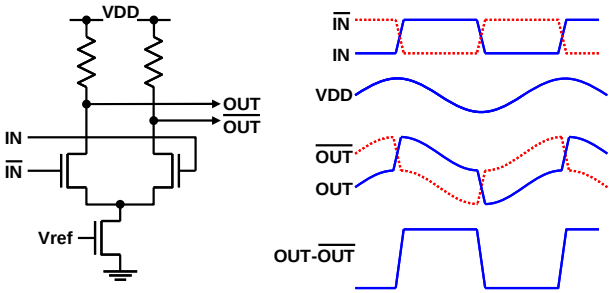


図 5･12　差動回路

5-5-6 ノイズの観測

LSI の設計時には，モデル化した電源系のインピーダンスと消費電流の変動を掛け合わせるなどして電源ノイズを求め，その電源ノイズが回路動作に与える影響を検証する．しかし，一般には LSI 内部の電源ノイズを直接観測することができないので，モデルから求めた電源ノイズが正しいか否かを検証することが困難である．そこで，LSI 内部の電源ノイズを実測評価するために，オンチップオシロスコープ [6]を用いる．**図 5･13** のようなオンチップオシロ

スコープで得られた実測ノイズ波形と，モデルから求めた電源ノイズ波形を比較・検証することで，電源ノイズのモデルなど，シグナルインテグリティ問題の設計精度向上に役立てることができる．

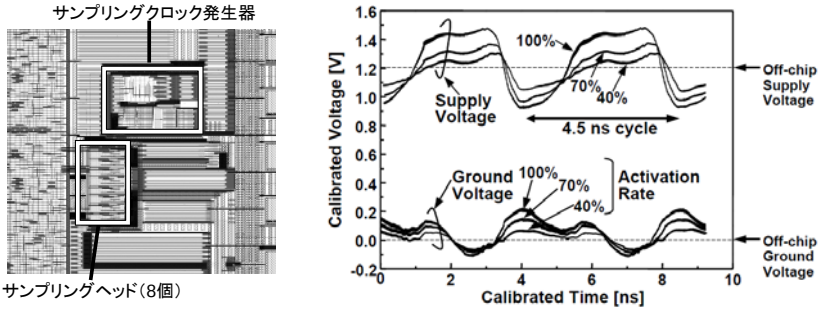


図 5･13　オンチップオシロスコープとその観測波形

■参考文献

1) B. Razavi, “Design of Analog CMOS Integrated Circuits,” McGRAW-HILL, 2001.
2) A. Hajimiri and T. H. Lee, “The Design of Low Noise Oscillators,” Kluwer Academic, 1999.
3) W. J. Dally and J. W. Poulton, “Digital System Engineering,” Cambridge University Press, 1998.
4) T. Rahal-Arabi, et al, “Enhancing Microprocessor Immunity to Power Supply Noise with Clock/Data Compensation,” Symposium on VLSI Circuits, pp.16-19, Jun. 2005.
5) M. Sode, et al, “A method using circuit/substrate macro modeling to analyze substrate noise in a 3.2-GHz 350M-transistor microprocessor,” Custom Integrated Circuit Conference, pp.687-690, Sep. 2008.
6) M. Takamiya, et al, “An on-chip 100GHz-sampling rate 8-channel sampling oscilloscope with embedded sampling clock generator,” IEEE International Solid-State Circuits Conference, pp.182-183, Feb. 2002.