特集

システムLSI設計検証技術

Verification Technology for a Complex System-on-a-Chip

あらまし

近年の複雑なシステムLSIの開発では，設計後期のシステム全体の検証段階で初めて設計仕様の致命的な不具合が発見されるケースが多く，開発期間やコストの増加が大きな問題になっている。短期間で効率良く開発するためには，開発の早い段階から，ハードウェア/ソフトウェアを含めたシステムレベルの検証を行う必要がある。

この課題の解決へ向けたソリューションとして，C言語によるモデリング技術をベースに仮想的なシステムプロトタイピングを行いハードウェア/ソフトウェアを同時開発する手法，およびエミュレーション技術をベースに短TATでプロトタイピングを構築して高速にシステム検証する手法を開発し，システムLSIの開発へ適用を進めている。

検証戦略に応じ，これらの技術を組み合わせてフローを構築することにより，完成度の高いシステムLSIを短期間で開発することが可能になった。

Abstract

The developers of a complex System-on-a-Chip (SOC) are often unaware of fatal specification problems until the final stages of overall system verification. This prolongs the SOC development period and significantly increases the development cost. Therefore, to develop an SOC efficiently in a short time, it is necessary to verify the system hardware/software design at the early development stage. We have developed a virtual system-prototyping technology based on a C-language modeling approach to enable concurrent hardware/software development from an early stage. We have also developed a rapid prototyping technology based on emulation to rapidly verify an entire system. Fujitsu has used both of these verification technologies to develop various SOCs. We can combine these technologies and determine the optimal sequence of the verification flow according to the SOC to be developed. Our verification methodology enables us to shorten the development period for an SOC and reduce a project's risk.

東　明浩（ひがし　あきひろ）

システムLSI開発研究所LSIデザイン研究部 所属
現在，システムLSI設計手法の研究に従事。

藤田隆司（ふじた　りゅうじ）

システムLSI開発研究所LSIデザイン研究部 所属
現在，システムLSI設計手法の研究に従事。

佐々木貴行（ささき　たかゆき）

システムLSI開発研究所LSIデザイン研究部 所属
現在，システムLSI設計手法の研究に従事。

ま え が き

今日，システムLSIにおけるソフトウェアの比重が大きくなり，従来のハードウェア中心の検証方法ではバグをすべて洗い出すことが困難な状況となってきている。システムLSIの検証においては，実際に使用するソフトウェアをハードウェアに搭載して実動作に近い状態でシステム検証を行うことが必須である。

従来の設計フローを**図**-1に示す。仕様，アーキテクチャ設計後，ハードウェア，ソフトウェアの詳細設計を行う。ソフトウェアの検証は，命令セットシミュレータ（ISS）を使用したソフト単体デバッグが主である。ハードウェア（RTL：Register Transfer Level）が完成した段階で論理シミュレータと協調検証を行う。

従来手法での問題点として，以下が挙げられる。

・RTL完成まで精度の高いソフトウェアデバッグができない。
・シミュレーション速度が遅く，現実的な時間内でシステムレベルの検証ができない。

本稿では，システムLSI開発における検証の現実的な解へ向けて，ソフトウェアをハードウェアに搭載した検証での課題を挙げ，課題を解決するための手法，適用例，考察を述べる。

品質の高いソフトウェアの早期開発

従来，ソフトウェア開発はハードウェア開発と独立に行っていたため，ハードウェアインタフェース部分の検証を十分デバックすることができず，ハードウェアが完成してから本格的なデバッグを開始していた。そのため，ハードウェアの検証のためにソフトウェアを使用しようとしてもソフトウェア品質が低いため，ソフトウェアが安定するまで時間がかかっていた。ソフトウェアをハードウェアに搭載する検証において，ソフトウェアの品質が高くない場合，バグの切り分けに時間がかかり，ソフトウェアのデバッグにほとんどの時間を取られ，本来のハードウェア検証時間が少なくなる問題がある。それを回避するためには，ハードウェアが完成していない状態で品質の高いソフトウェアが必要となる。そこで，実ハードウェアに近いソフトウェアデバッグ環境をソフトウェア開発者に提供し，品質の高いソフトウェアを早期に開発する必要がある。

仮想プロトタイプを使用したソフトウェアデバッグ環境を**図**-2に示す。CPUモデルとしてISSとバスインタフェース，およびハードウェアモデルとして機能，レジスタ，バスインタフェースを仕様に基づきC言語で記述してある。機能は時間概念のない動作レベル，バスインタフェースはサイクルレベルの各抽象レベルである。

ソフトウェア開発初期段階では，機能とレジスタだけのモデルを使用してデバッグを行う。完成度の向上に伴い，クロック単位のディレイ情報を付加したバスインタフェースモデルを使用して，実動作に近い精度でデバッグを行う。本デバッグの結果は，ソフトウェアとハードウェアの協調動作，データ転送，割込み処理，バス調停などに消費するサイクルを反映している。また，モデル精度に応じてシステムのチューニングを行い，性能の確認，およびハードウェア/ソフトウェアの分割ポイントなどアーキテクチャの見直しを行うことでソフトウェアとハードウェアの完成度を高めることができる。

これらの環境は，仮想的なシステムプロトタイプとして，プラットフォームの形態でソフトウェア開発者へ提

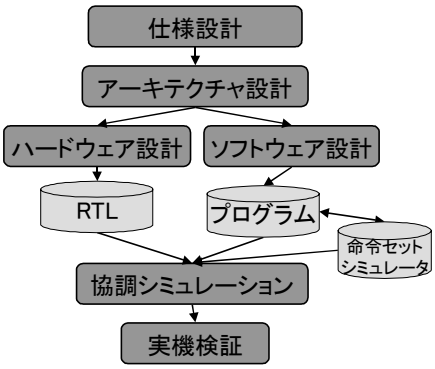

図-1　従来手法による設計フロー
Fig.1-Traditional design flow.

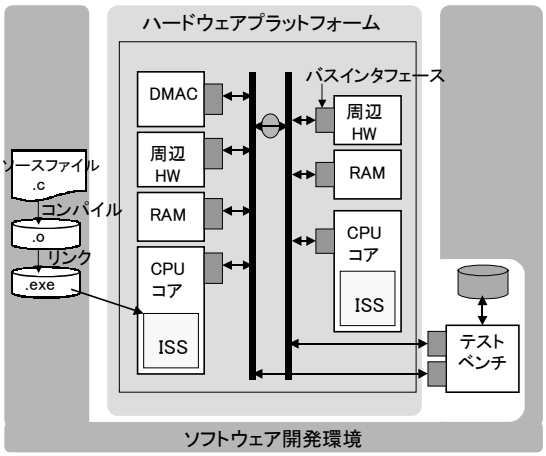

図-2　仮想プロトタイプ
Fig.2-Virtual prototype of an SOC.

供される。プラットフォームは，CPU，メモリ，周辺回路を含み，複数の抽象度の設計データから構成されており，検証の目的や，必要とする精度とパフォーマンスのレベルに合わせてモデルを選択できる。本環境を使用することによりソフトウェア開発者は，ほぼ実回路と同じ環境でデバッグができる。このような環境をソフトウェア開発者に提供することにより，ハードウェア検証に使用できる高品質のソフトウェアを早期に開発することが可能となった。

システム検証の高速化

提案する検証フローを**図-3**に示す。ソフトウェアを搭載したハードウェア検証を論理シミュレータで行うとシミュレータ自身の負荷と，ツール間の接続にPLI（Programmable Language Interface）などを使用するため低速になる。さらに，ソフトウェアを走行させるためには大量のクロックを入力する必要があり，検証時間が増大する。検証時間を短縮するためには，論理回路をエミュレーション専用FPGA（Field Programmable Gate Array）などにマッピングし，高速にシミュレーションを行うエミュレータ装置や，汎用FPGAを使用してシステムを構築し，高速に動作させるプロトタイピング装置を使用する必要がある。

エミュレータ装置は，高速かつデバッグ機能が豊富であるが高価である。プロトタイピング装置は，エミュレータ装置より高速に動作し比較的安価であるがデバッグ性が悪い。それぞれの特徴を理解し適材適所に使用することにより，検証を効率的に行うことができる。また，大規模な回路になると，エミュレータ装置にすべての回路を搭載できなくなり，回路を分割して検証を行わなければならない場合がある。分割された回路を検証するためにはエミュレータ装置とプロトタイピング装置に，分割された回路を搭載して並列に検証を行う。

シミュレータ，エミュレータ装置，プロトタイピング装置をすべて使用すると，ソフトウェア，ハードウェアを含めたLSIの完成度は向上するが，それぞれの検証環境の立上げに工数がかかるため，極力環境の共通化を行い，効率的に環境を立上げることが重要である。検証環境の立上げ効率化のため，エミュレータ装置用/プロトタイピング装置用メモリ自動生成ツール，ライブラリ変換ツール，論理分割を効率的に行うためのツール，自動化スクリプトを作成した。また，プロトタイピング装置では汎用のFPGAを使用しているため，検証対象の回路に対して種々の制約が発生する。そのため，FPGAの仕様を理解してシステム構築を行う必要がある。

以下にメモリ自動生成ツール，プロトタイピング装置，短期間での環境立上げについて説明する。

● メモリ自動生成ツール

シミュレータ，エミュレータ装置，プロトタイピング装置の環境における最大の差異はメモリの実現方法である。各環境において使用可能なメモリを以下に示す。

・シミュレータ：各LSIテクノロジの内蔵メモリ

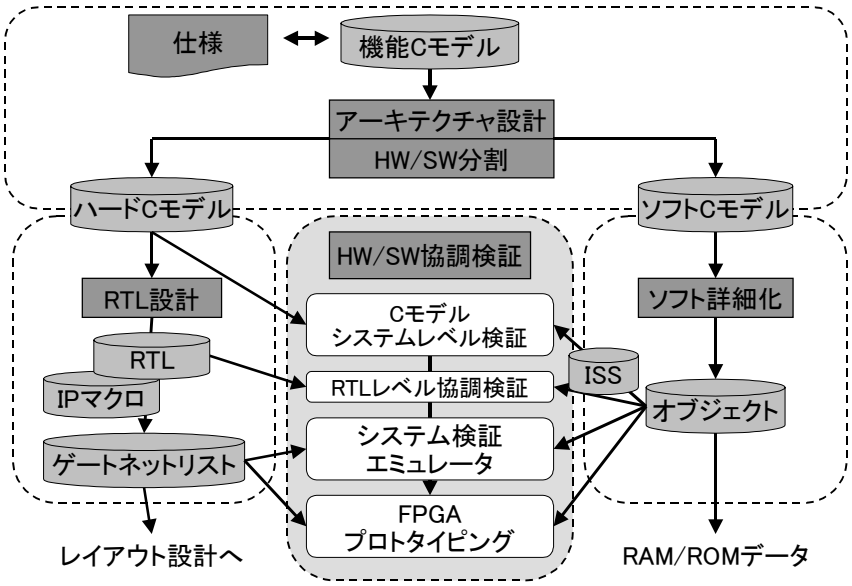

図-3　システムLSI設計フロー
Fig.3-New design flow for an SOC.

・エミュレータ装置：エミュレータ固有のメモリ
・プロトタイピング装置：FPGAのブロックメモリ

上記差異のため，シミュレータで使用したメモリをエミュレータ装置用，プロトタイピング装置用にそれぞれ変換する必要があり，多数のメモリを使用するLSIにおいては作業に時間がかかってしまう。そこでメモリ仕様を入力するとエミュレータ用メモリ，FPGAのブロックメモリを自動的に生成するツールを作成した。本ツールを使用することにより，環境ごとに検証済のメモリを自動的に生成することができ，短期間で環境を立上げることが可能となる。

● プロトタイピング装置

プロトタイピング装置は，エミュレータ装置よりも動作速度が速く，モジュール化されているため，外部装置との接続やIPモジュールの作成が行いやすい。デバッグ性ではエミュレータ装置に劣るが，ある程度の品質の回路検証に使用すると効果的である。また，ハードウェア検証後にソフトウェア開発用として使用することにより，早期にソフトウェア開発のための高速なデバッグ環境を提供することが可能となる。プロトタイピング装置の各モジュールを接続する方法として以下のものがある。

・接続用FPGAによるダイレクト接続方式
・専用の時分割バスによる接続方式

動作速度はダイレクト接続方式のほうが速くなるが，時分割バス接続方式はトレース機能やメモリとのデータ転送機能などをユーザが特別に回路を付加しなくても容易に使用することができる。デバッグ性およびシステムの構築しやすさは時分割バス接続方式が有利となる。

プロトタイピング装置への搭載フローを**図-4**に示す。プロトタイピング装置は多数の汎用FPGAを接続してシステムを構築するため，エミュレータ装置が自動的に処理していた論理分割やクロックスキューの調整をユーザが行う必要がある。

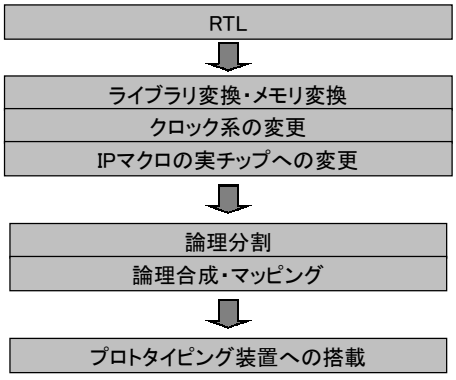

図-4　プロトタイピング装置への搭載フロー
Fig.4-Installing flow to prototyping device.

● 短期間での環境立上げ

エミュレーションやプロトタイピング装置にはメモリやクロック系などシミュレーションにはない制限事項が存在する。また，エミュレーションやプロトタイピング装置で高速にエミュレーションするためには論理合成可能なRTL記述が必須となる。短期間でシミュレーション，エミュレーション，プロトタイピング装置の検証を立上げるためには，それぞれの制限事項を考慮してエミュレーションやプロトタイピング装置で使用可能なシミュレーション用テストベンチの設計を行う必要がある。エミュレーションやプロトタイピング装置へ実装するためのコンパイル時間は非常に長いため，テストベンチのバグによるやり直しは大きな時間的ロスになる。エミュレーションやプロトタイピング装置の検証環境を短期間で立上げるためには，各作業のフェーズでシミュレーションにより正常性を確認しながら作業を進める必要がある。以下に各フェーズと確認環境を示す。

・メモリ/ライブラリ変換：変換後のRTLシミュレーション
・論理分割：分割後のRTLシミュレーション
・論理合成：論理合成後のゲートレベルシミュレーション
・配置配線：配置配線後のゲートレベルシミュレーション

上記のように段階的にシミュレーションを行うことにより，早期に問題を発見することができ，さらに問題が発生した場合の切り分けが容易となる。

システム検証でのデバッグ効率化

ソフトウェアをハードウェアに搭載したシステム検証において期待値と異なる結果が発生した場合，デバッグは困難となる。期待値不一致が発生した場合，バグ切り分けのためにはソフトデバッガが有効である。シミュレータにおいてソフトデバッガをサポートするために，ISSにソフトデバッガを接続できるツールを使用する。また，エミュレータおよびプロトタイピング装置には，ICE（In Circuit Emulator）を接続することによりソフトデバッガを使用可能とする。

期待値不一致になった場合，ソフトデバッガのブレークポイントやレジスタまたはメモリの内容を見ることにより，問題の切り分けを行うことができる。

ソフトデバッガによりハードウェアに問題があると判明した場合，問題のあるモジュールのみをシミュレータ

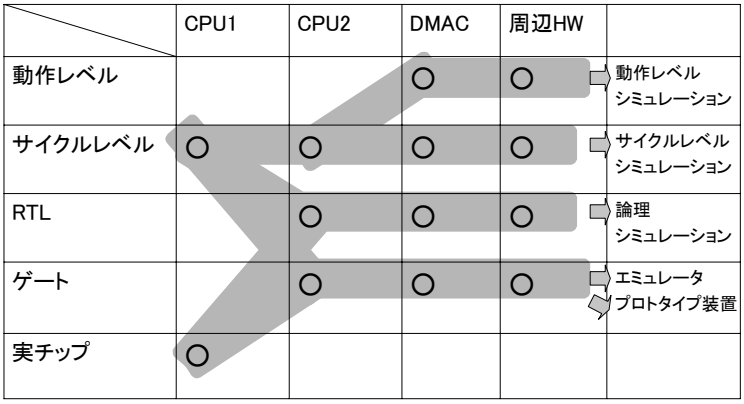

図-5　各抽象レベルと検証方法の組合せ
Fig.5-Abstraction level versus verification method.

で動作させ，ほかの部分はエミュレータ装置またはプロトタイピング装置で動作させる協調シミュレーションを行うことによりデバッグ性を向上させることができる。

適　用　例

図-3に示した検証フローをコンシューマ市場向けシステムLSIに適用した。検証対象には複数のCPUが搭載され，それぞれプログラムを含んでいる。検証に使用したハードウェアモデルの一部と検証方法の組合せを**図-5**に示す。UT（Un Timed：動作レベル），BCA（Bus Cycle Accurate：サイクルレベル）モデルを組み合わせたC言語ベースのプロトタイプでソフトデバッグを行った。RTLの単体検証後，ISSと論理シミュレータで協調シミュレーションを行い，チェックプログラムによる初期動作の検証を行った。アプリケーションプログラムによるシステム検証は，実チップと論理合成後のゲートレベルのデータを使用してエミュレータで実施した。長時間のストリームデータを使用したソフトウェアデバッグは，プロトタイピング装置で実施した。プロトタイピング装置では，Cモデル（BCA），実チップ，ゲートレベルの各モデルを必要に応じ組み合わせて検証を行った。各検証環境での動作時間を**表-1**に示す。実チップの処理時間はクロック100 MHzとして計算を行った。

新検証フロー適用の効果として次の点が挙げられる。

・従来と比較して3か月程度早い段階で，精度の高いデバッグができ，ソフトウェアの完成度が大幅に向上した。
・実行可能なモデルによる検証により，仕様の見落としや過不足が開発の早い段階で明確になった。
・エミュレータおよびプロトタイピング装置による検証の高速化により，システム検証期間を従来に比べて1/2に短縮することが可能となった。

表-1　各環境での動作時間

	動作速度（クロック/秒）	ストリームデータ1秒の処理時間
論理シミュレータによる協調検証	100 Hz	277時間（12日間）
Cモデルプロトタイプ（クロックサイクル精度）	5 kHz	5.5時間
エミュレータ	500 kHz	3.3分
プロトタイピング装置	1 MHz	1.6分

む　す　び

複雑なシステムLSIの開発において，ソフトウェアとハードウェアを効率的に検証する技術として，C言語によるモデリング技術をベースに仮想的なシステムプロトタイピングを行い，品質の高いソフトウェアを早期開発する手法，およびエミュレーション技術をベースに短TAT（Turn Around Time）でプロトタイプを構築して高速にシステム検証する手法について述べた。

これからのシステムLSIの検証は，一つの手法だけではバグをすべて洗い出すことは不可能と思われる。またシミュレーションの速度も重要な要素である。各種ツールおよび各種装置を組み合わせることにより，効果的な検証が可能となる。開発プロジェクトごとに最適化した設計手法をベースとして，各種ツールおよび各種装置の特徴をよく理解し，うまく組み合わせて検証を行う戦略が重要となる。

参 考 文 献

(1)　A. Higashi et al.：Verification methodology for a complex System-on-a-chip. *FUJITSU Sci. Tech. J.* Vol.36, No.1, p.24-30（June 2000）.