SystemCを活用したシステムLSI開発支援

伊藤　徹　　依田　究
飯田　康詞

システムLSIへの要求は，高機能／高性能化，低消費電力化，開発期間の短縮など，ますます高度化している。この要求に応えるため，SystemCを用いたCベース設計が注目を集めている。

本稿では，上流設計において，あるアプリケーションを実現するためのソフトウェアおよびハードウェアの最適な構成の探索（アーキテクチャ探索）に用いられるSystemCによる仮想システムLSIについて述べ，次にSystemCによる開発において新たな課題である等価性検証用に再認識されたプロトタイピングボードを取り上げ，その開発手法について述べる。

システムLSI開発への要求事項

高まるシステムLSIへの高機能化要求を実現するために，1990年代には，設計に用いられる記述レベルがゲートレベルから，より設計生産性の高いRegister Transfer Level（RTL）に移行した。近年，さらなる設計生産性の向上を図るため，さらに記述の抽象度を向上させたCベース設計が注目されている。

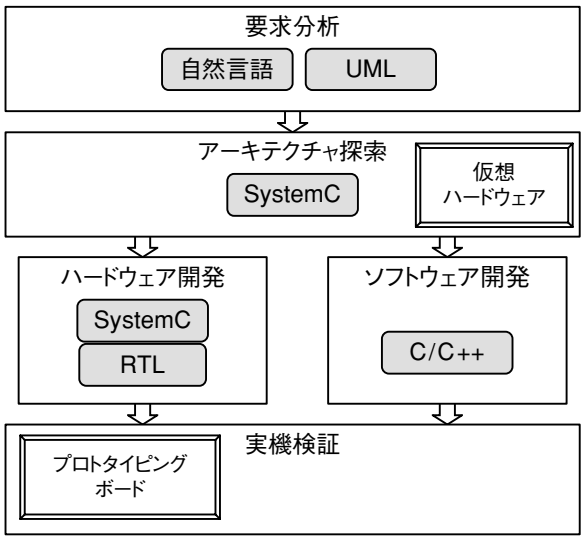


図1　システムLSI開発フロー

当社においても，SystemCによる設計手法を確立し，製品への適用例も増えてきている。

図1はSystemCを用いたシステムLSIの開発フローの概略を示している。図中，網掛けの部分は，主に使用する記述言語を示している。

以下に，開発フローで開発環境に要求される事項について示す。

（1）上流設計でのアーキテクチャ探索環境の提供

システムLSIは，マイクロコントローラ，ソフトウェア，各種I/O，およびアプリケーション固有のアクセラレータなど，さまざまなモジュールから構成される。このため上流設計では，要求される性能，コスト，開発期間などのトレードオフを考慮し，要求仕様に対して最適なシステムLSIの構成を決定するアーキテクチャ探索を行う。

アーキテクチャ探索の際にシステム性能見積りを誤ると，必要以上に機能を盛り込み，過剰スペックとなってコスト増大をもたらし，また逆の場合には必要とする性能を達成できなくなる。このような問題が設計の下流工程やLSIの製造後などで発見されると，作り直しによる多大な追加費用と時間が必要となるために，上流設計での十分な検討が必要となる。

システム性能見積りには，仮想システムLSIの活用が有効である。仮想システムLSIとは，開発対象のLSIをPC上やWS上に構築したものである。この上で実行時に想定されるデータの転送を行って性能のボトルネックを特定したり，ソフトウェアを実行させて処理性能を測定したりして，最適なアーキテクチャを検討する。

このようなアーキテクチャ探索の場面では，仮想システムLSIの構成を変えながら試行を多数繰り返すこととなる。一回の試行を短時間にして開発期間を短縮させるために，仮想システムLSIの動作には高速性が要求される。また，仮想システムLSIを構成する部品であるマイクロコントローラや汎用I/Oなどは，開発の途中で仕様変更が発生する可能性が高い。この仕様変更に即座に対応するた

めに，仮想システムLSIやその部品には，短期間での開発が要求される。

(2) シミュレーションにかわる高速な検証環境の提供

アーキテクチャ決定後は，ハードウェアとソフトウェアとに分かれて開発が並行して行われる。

SystemCを用いたCベースのハードウェア開発では，SystemCで記述した動作モデルを，動作合成，論理合成，配置配線の各工程を通すことにより，順次RTL，ネットリスト，マスクパタンのデータを得る。開発の過程で生成されるRTL，ネットリスト，マスクパタンのデータは，それぞれの前工程のデータとの等価性の検証を行う必要があるが，SystemCによる設計は発展段階にあり，十分な等価性検証のツールがまだない。等価性を検証する方法として，シミュレーションを用いた検証方法があるが，これには時間がかかるという欠点がある[1)]。

そこで当社では，プロトタイピングボードによる検証に再度注目している。プロトタイピングボードの開発においても，最終製品であるシステムLSIへの要求事項に関連して，いくつかの工夫が必要となっている。具体的な開発手法について，「プロトタイピングボードによる検証環境の提供」で説明する。

仮想ハードウェアを用いた検証環境

(1) モデルの説明

上記で示した通り，システムLSIの設計にあたって，SystemCを用いてアーキテクチャ探索が行われ始めている。アーキテクチャ探索をするための一般的なモデルを図2に示す。

このモデルでは，ソフトウェアとハードウェアの両方から構成される仮想的なシステムLSIが，PC上に構築される。ユーザーは実際のハードウェアを用いること無くシステムLSIのシミュレーションが可能である。

図2の説明を以下に示す。

- ソフトウェア：アプリケーションを実現するために必要なプログラムである。プログラムはARM*[1)] などのターゲットチップ向けにクロスコンパイルされ，モデルに実装される。
- 命令セットシミュレータ（Instruction Set Simulator，以下ISSと略記）：ターゲットチップのモデル用にクロスコンパイルされたソフトウェアを，PC上で実行するためのインタプリタである（Just In Timeコンパイラとして実装される場合もある）。CPUごとに用意される。パフォーマンス上の理由から，C/C++で記述されることが多い。
- 仮想ハードウェア：タイマなどのペリフェラルを仮想的に実装した部位である。システムLSIごとに開発する必要がある。当社は主業務の一環として，SystemCによるこの部位の開発受託を行っている。

(2) 業務フロー

仮想ハードウェアの受託開発を請け負う際の，ユーザーおよび当社の業務フローを図3に示す。

ユーザーは，まずシステムLSIのソフトウェア／ハードウェアの仕様を策定し，仮想ハードウェアの開発に関し

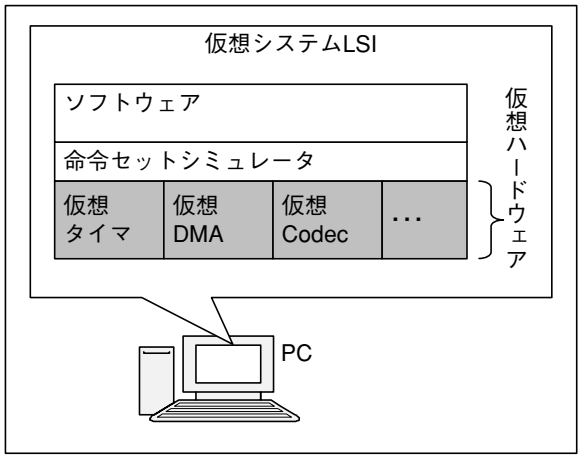


図2　モデル

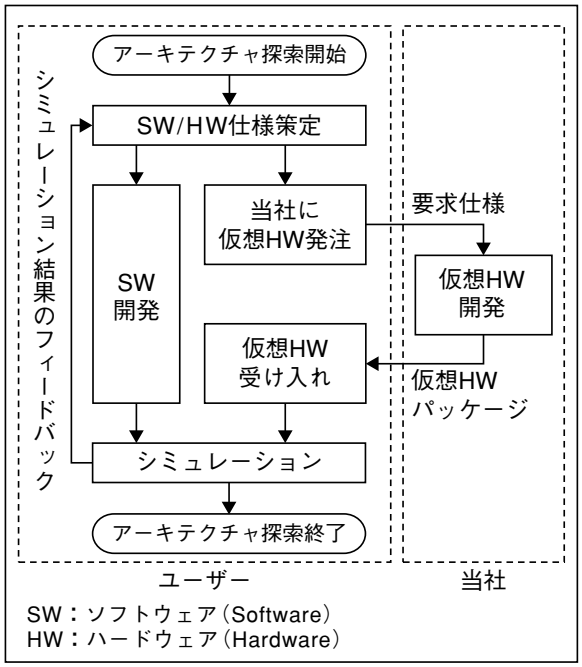


図3　業務フロー

＊1）ARMはARM Ltd.の登録商標です。

ては当社に発注する。当社における仮想ハードウェアの開発と平行して，ユーザーはソフトウェアの開発を進めることが可能である。その後，ユーザーは，開発したソフトウェアと，当社が提供した仮想ハードウェアとを用いてシミュレーションを行い，その結果を仕様に反映する，というループを繰り返すことにより，実際のハードウェアを用いないアーキテクチャ探索が可能である。

(3) 仮想ハードウェアの開発に際して

ユーザーのアーキテクチャ探索TAT（Turnaround Time）の短縮を支援するため，当社では仮想ハードウェア開発に際して以下の取り組みを行っている。

(a) シミュレーション時間の短縮

シミュレーション時間の短縮のため，仮想ハードウェアのパフォーマンス向上に取り組んでいる。

たとえば，仮想ハードウェアの設計時に，極力スレッドの数を少なくした設計をユーザーに提案する。一般に，仮想システムLSIが動作するWindowsなどの汎用OSは，コンテキストスイッチに必要な時間が大変長い（状況によるが10ms程度は必要）。よってスレッドを多用した設計はパフォーマンス劣化の大きな原因になり得る。

また，UTF（Untimed Functional）レベルはBCA（Bus Cycle Accurate）レベルに比べシミュレーションの精度が低いが，シミュレーション時の処理が比較的少なく高速化を期待できる。よって，要求仕様のヒアリングを行う際に，ユーザーのアーキテクチャ探索の状況を鑑み，可能であればUTFレベルを用いてTATの短縮を図ることを提案する。

(b) テスト時間の短縮

仮想ハードウェアのテストに際し，テストプログラム（Test Program，以下TPと略記）を用いた自動テストを実施している。TPも，仮想ハードウェアと同様SystemCで記述される。構成を**図4**に示す。

図のように，テスト対象となる仮想ハードウェアとTPとをSystemCのポートにて接続する。TPが仮想ハードウェアに対するテストドライバおよびテストスタブとして動作することとなる。

このように，テストを自動化すると，当社において開発手戻りが発生した際の，再テストに必要な時間を事実上ゼロにできるため，開発期間の短縮に有効である。また，当社はTPをもユーザーに提供しているため，ユーザーにおける受け入れテスト時間の短縮にも有効である（事実上，受け入れテストの工数がゼロになる）。

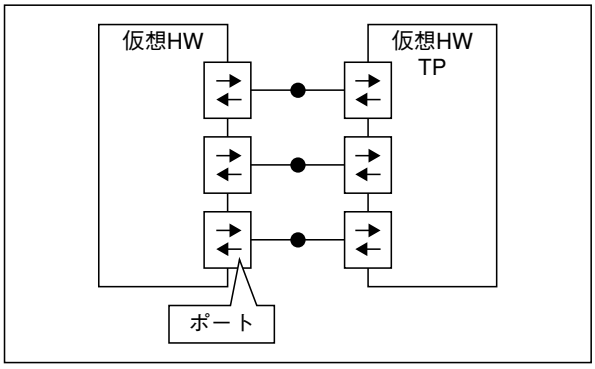


図4　仮想ハードウェアテスト構成

(4) シミュレーションの更なる高速化を目指して

図2に示したように，一般に，システムLSIのシミュレーションを行う際には，ソフトウェアをターゲットチップ向けにクロスコンパイルして生成したコードを使用する。よって，クロスコンパイルされたソフトウェアを実行するためにはISSが必要となる。この場合，PC向けにネイティブコンパイルされたソフトウェアを実行する場合に比べて，約10倍～100倍程度の速度の低下が発生する（PCとターゲットチップが同一クロック周波数の場合）。

よって，当社は現在，以下のような技術的な課題に取り組んでいる。

- **図5**に示すように，ソフトウェアをPC用にネイティブコンパイルする。
- そのソフトウェアと仮想ハードウェアを直接リンクする。
- リンクされたオブジェクトを，ISSを介さずにPC上で直接実行する。こうすることによって，シミュレーション速度の約10倍～100倍の向上を図ることができる。

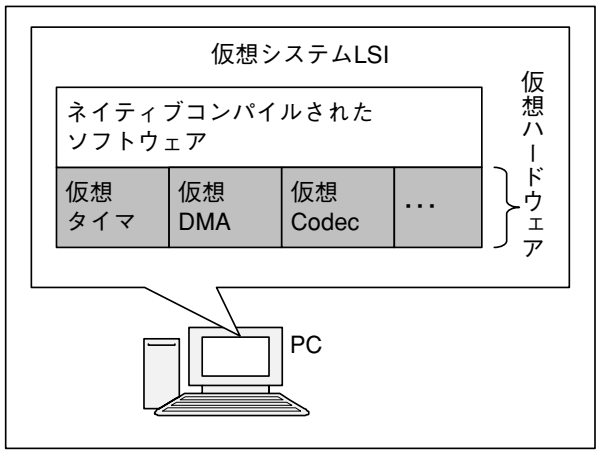


図5　シミュレーション高速化の取り組み

将来的に，このようなシステム構成をユーザーに提案することにより，ユーザーのアーキテクチャ探索の効率上昇に貢献してゆきたいと考えている。

プロトタイピングボードによる検証環境

(1) 業務フロー

当社でのプロトタイピングボードの開発フローを**図6**に示す。

ユーザーから提示されたLSIの構成を受けて，プロトタイピングボードの構成を検討し，プロトタイピングボードに使用する最適なFPGAを選定し，さらに，そのFPGAに適合するようにLSI用RTLを変更する。また，平行してボードを設計，作成する。

こうして作成したプロトタイピングボードを使用してユーザーはIP検証を行う。

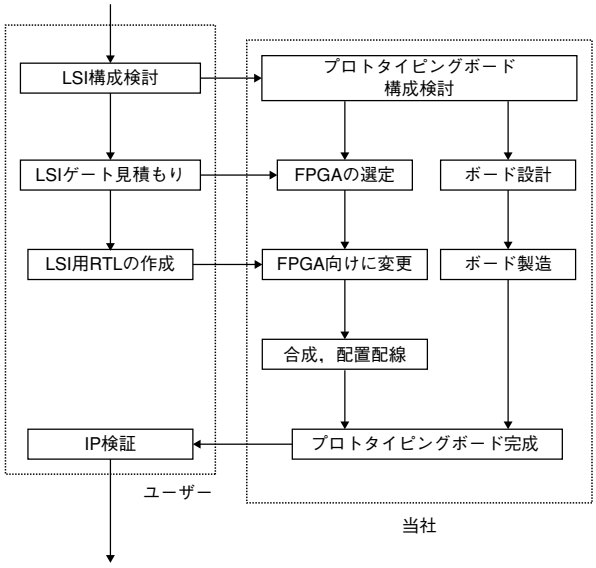


図6　プロトタイピングボードの開発フロー

LSIが高機能化する中で以下のような要求が高くなっている。

- LSIが高機能化するのにともないゲート規模が大きくなるため，より大きなゲートを実装できるプロトタイピングボードが要求される。
- 低消費電力化のため，ゲーテッドクロックを含むRTLのFPGAへのインプリメントが要求される。
- IP検証またはLSI完成前のソフトウェア開発環境として使用するため，プロトタイピングボードの開発期間短縮が要求される。

＊2）ARM7TDMI，ARM946EJ-S，ARM926EJ-SはARM Ltd.の商標です。

そこで，本項では上記の要求に対する当社の対応について述べる。

(2) ゲート規模の増加への対応

当社のプロトタイピングボードで使用しているCPUコアとFPGAのゲート数（slice数）の関係を**図7**に示す。

図7に示すようにCPUコアがARM7TDMI→ARM946EJ-S→ARM926EJ-Sと高性能化し，またLSIの微細加工も進化するのに対応してLSIに実装するIPのゲート規模も大きくなっている。そのため，プロトボード上のFPGAも容量が大きく高速のものを使用するようになっている。

現在開発中のプロトタイピングボードでは，この要求を満たすため，収容可能なゲート数および動作速度ともに最上クラスのFPGAを使用している。

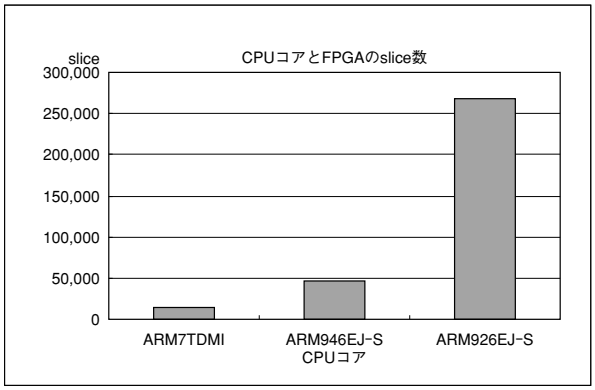


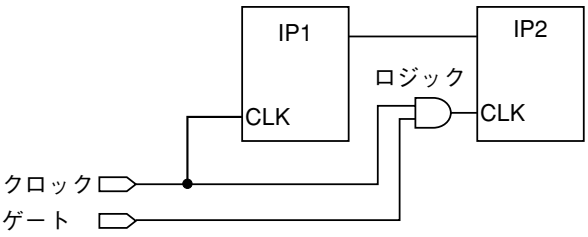
図7　ゲート規模の増加＊2)

(3) ゲーテッドクロックへの対応

プロトタイピングボードではLSIのRTLを検証するため，LSI用RTLをそのままインプリメントする必要がある。しかし，LSIのRTLは低消費電力化のため，各IPのクロックを個別に停止できるようなゲーテッドクロックが使われている。

図8（次ページ）に，ゲーテッドクロックを含むRTLをFPGAにインプリメントした，回路の一例を示す。IP1で生成されたデータをIP2に正しく入力するためには，IP2に入力されたクロックの変化の後，一定期間データの状態を保持する（ホールドタイム）必要がある。しかし，図中のロジック部分のクロックの伝搬遅延によって，この保持期間を確保できなくなってしまう場合がある。このような状態は，ホールドタイム違反と呼ばれ，FPGAの異常動作の原因となる。

そこで，ゲーテッドクロックのゲート信号を，FPGA内のフリップフロップのクロックイネーブル（CE）に自動



IP2を使用しないときはゲートをLowとしてIP2へのクロックを停止する。

図8　ゲーテッドクロック

変換可能なツールを採用して，この問題を回避している。**図9**に，自動変換されたゲーテッドクロックの回路を示す。

変換後はロジックが削除されるためクロックの伝搬遅延が無くなり，ホールドタイム違反が発生しなくなる。

(4) 開発期間短縮への対応

プロトタイピングボードの開発期間を短縮するため，当社はプロトタイピングボードをCPUコア，メモリコントローラのように各LSIで共通となる部分と各LSIで固有の機能となる部分を分けて開発を行っている。

写真1はCPUコアにARM946EJ-Sを使用したプロトタイピングボードである。

各LSI固有の機能は**写真2**の拡張ボードにインプリメントし，**写真1**の拡張スロットの部分に**写真2**の拡張ボードを挿入して使用する。

拡張ボードは，プロトタイピングボード上に無いアナログ回路，プロトタイピングボード上のFPGAに実装できないIPをインプリメントするFPGAを持つ。

このように開発することで，新規に設計する部分を最小限とし，プロトタイピングボードの開発期間短縮を実現している。

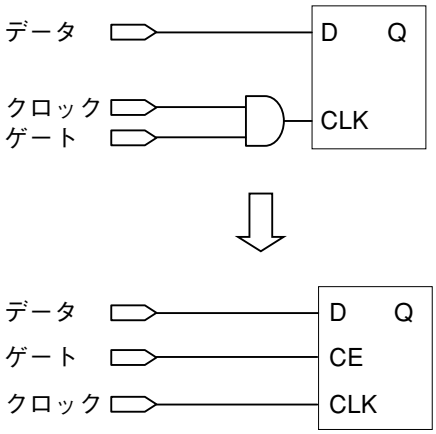


図9　ゲーテッドクロックの変換

また，FPGAに実装する回路のシミュレーションを実施する場合，プロトタイピングボード上のFPGAに実装される回路以外の外部回路をシミュレーションするモデル化を作成しなければならないが，時間がかかる。

そこで，当社では当該のLSI用の回路をシミュレーションするためのモデルを流用してプロトタイピングボード用のシミュレーション環境を実現することによって開発期間の短縮を実現している。

LSI用シミュレーション環境は，プロトタイピングボード用のシミュレーション環境とは外部回路の構成が違っているためそのまま接続して使用することができないが，これらの回路の構成の違いを吸収する回路（**図10**ではラッパと記述）を設計することによって解決している。

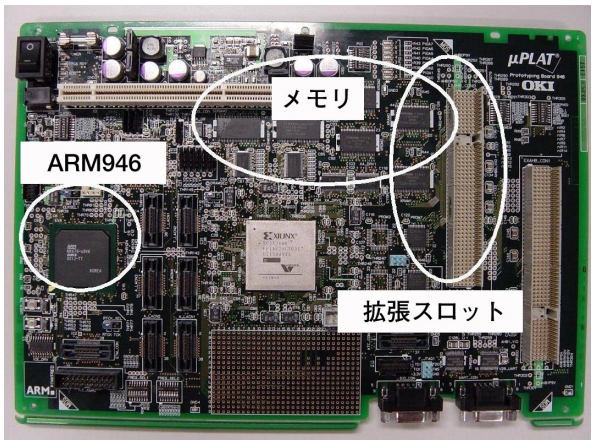


写真1　プロトタイピングボード

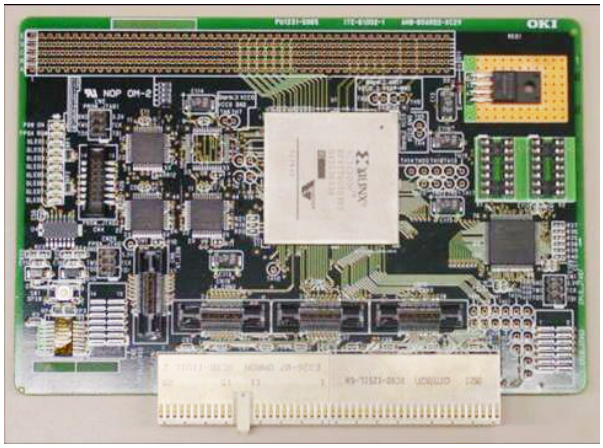


写真2　拡張ボード

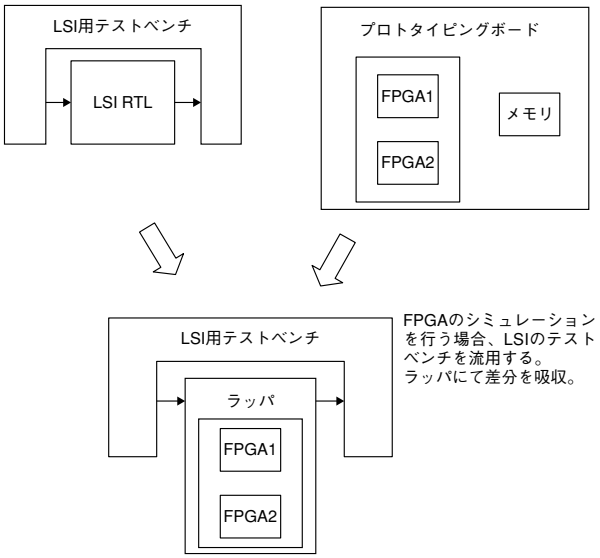


図10　LSI用テスト環境の流用

p

ま と め

SystemCによる仮想ハードウェアモデルの構築とその実行動作の高速化の手法について，またプロトタイピングボードでのLSIの検証を可能とするためにLSIの回路の実装の仕方について述べてきた。

仮想ハードウェアモデルは，スレッド数の削減やSystemCの抽象化レベルの適切な選択によって，要求される高速なシミュレーションを実現することができる。また，テストの自動化によって開発途中に発生する仕様変更などに，迅速に対応することができる。

プロトタイピングボード上のFPGA上にLSIと等価な回路をインプリメントする際に必要な，ゲーテッドクロックの対応は，適切なツールの利用によって効率的に行っている。また，プロトタイピングボードの構成を，各LSIの共通部分と拡張部分に分割することによって，開発期間の短縮を図っている。

ところで，システムの開発期間を短縮し，また開発効率を向上させるためには，LSIのハードウェアの開発とソフトウェアの開発とを並行して進めていく，いわゆるハード・ソフト協調設計が必要となる。しかしハード・ソフト協調設計を実現するためには，当該LSIをシミュレーションする環境を構築するだけでは不十分であり，当該LSIを含むシステム全体をシミュレーションすることが課題となる。同時にシミュレーション時間の短縮が課題となる。われわれはこうした課題にも取り組んでおり，稿をあらためて報告したい。 ◆◆

■参考文献

1）児玉秀賢：SoCをC言語アルゴリズムで受注しアーキテクチャ設計で差異化を図る，日経マイクロデバイス，pp.73-79，2007年2月

●筆者紹介

伊藤徹：Toru Ito. 株式会社沖テクノコラージュ システム本部
依田究：Kiwamu Yoda. 株式会社沖テクノコラージュ システム本部 システム開発第1部 開発第4チーム
飯田康詞：Koji Iida. 株式会社沖テクノコラージュ システム本部 システム開発第2部 開発第1チーム