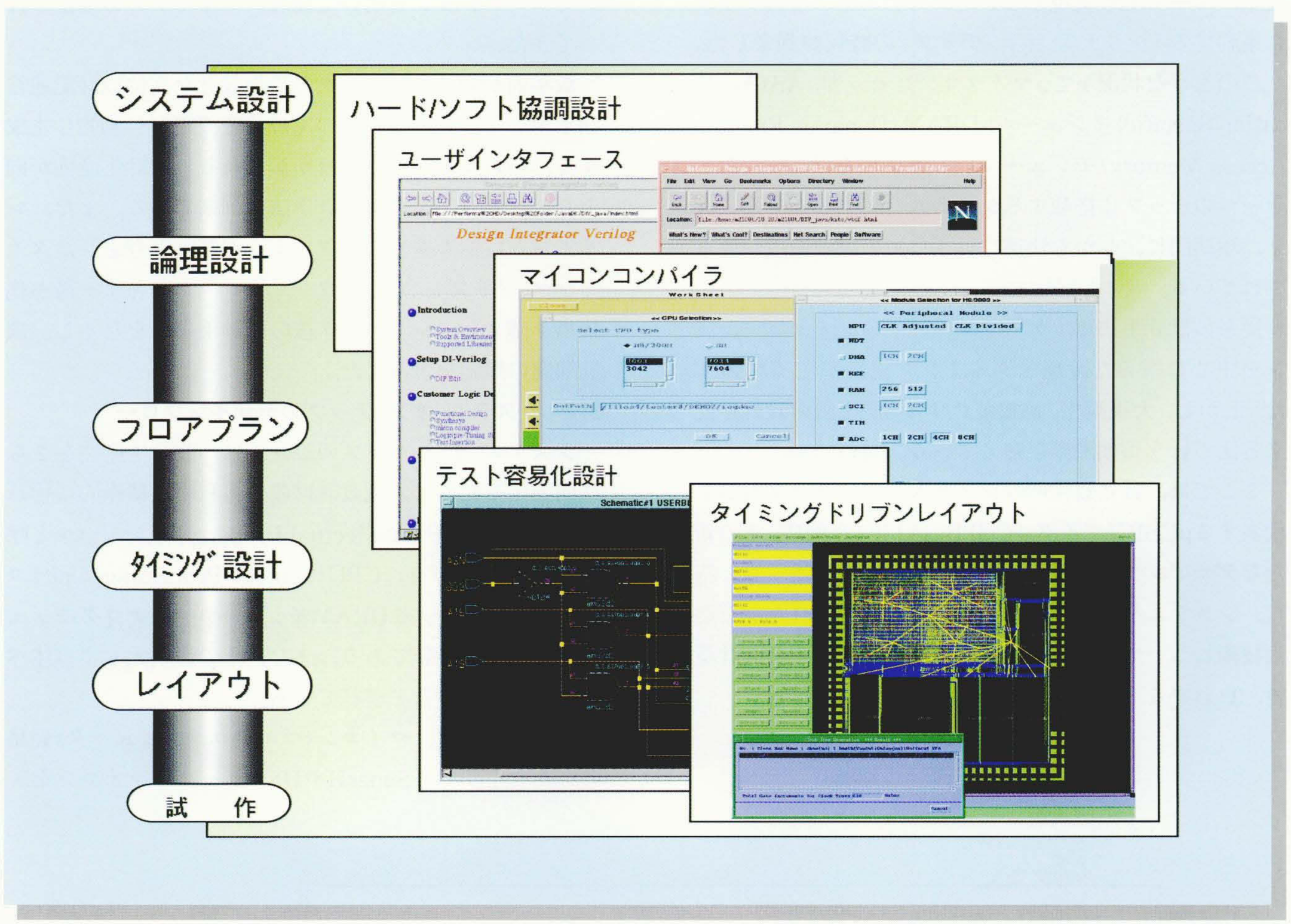
システム オン チップを実現するデザインフローと設計環境

Design Flow and Design Environment for the System-on-a-Chip

大野康宣 *Yasunori Ôno* 野本和之 *Kazuyuki Nomoto*
塩月八宏 *Yahiro Shiotsuki* 日下憲二 *Kenji Kusaka*



注：Netscapeは，米国，日本およびその他の国における米国Netscape Communications Corp.の商標である。

システム オン チップの設計環境

デザインフローの概要を示す。システム オン チップの設計では，ハードウェア・ソフトウェア協調設計環境，テスト設計環境などが重要な要素技術となる。

プロセス技術の進歩によってLSIの集積度が飛躍的に増大し，これまでボードで実現していたシステムを，1チップの上に搭載することができるようになった。チップに搭載する部品も，スタンダードセルやメモリモジュールに加え，CPU(Central Processing Unit)コア，AS(Application Specific)モジュール，DRAM(Dynamic Random Access Memory)モジュール，アナログモジュールなどと多様化している。

これに伴い，こうした「システム オン チップ」を設計するための新しいデザインフロー，および設計環境が必要である。

その主なものとして，(1) CPUコアを搭載する場合のハードウェア・ソフトウェア協調設計環境，(2) 大規模LSIを高速に論理・タイミング検証するために，従来のシミュレーション中心でなく，形式論理検証や静的タイミング検証を中心に据えた設計環境，(3) 大規模でかつ多種多様な部品を搭載する，LSIのテスト設計のための環境がある。

従来の論理合成，シミュレーション，およびレイアウトツールを核にした設計環境に，これらの新しい技術を盛り込み，快適に，かつ短期間に設計ができるような新しい統合設計環境を構築していく。

1. はじめに

プロセス技術の発達によって論理LSIもハーフミクロンからクォータミクロンの世界に突入し，チップに搭載できるゲート規模も飛躍的に増大した。これに呼応し，従来，ボード上で実現していたシステムを一つのチップに集約する「システム オン チップ」の時代が到来した。

このような状況下で，マイクロプロセッサ，AS(Application Specific)モジュール，DRAM(Dynamic Random Access Memory)モジュール，アナログモジュールなどを一つのチップに搭載するシステム オン チップの設計を，短期間に，しかも快適に行うための設計環境が要求されている。

従来の設計手法は，ゲートレベルでのシミュレーションを中心に据えた論理・タイミング検証が主流であった。しかし，大規模化，高機能化する論理LSIの設計を従来方法で行うには限界を迎えている。

ここでは，日立製作所のシステム オン チップ製品を代表するμCBIC(マイクロCell Based IC)を例に，その新たなデザインフローおよび設計環境について述べる。特に，システムを設計するために新しく取り入れた主な要素技術について詳述し，最後にそれらを統合した設計環境，いわゆる「デザインキット」について述べる。

2. システム オン チップのデザインフロー

2.1 従来のデザインフロー

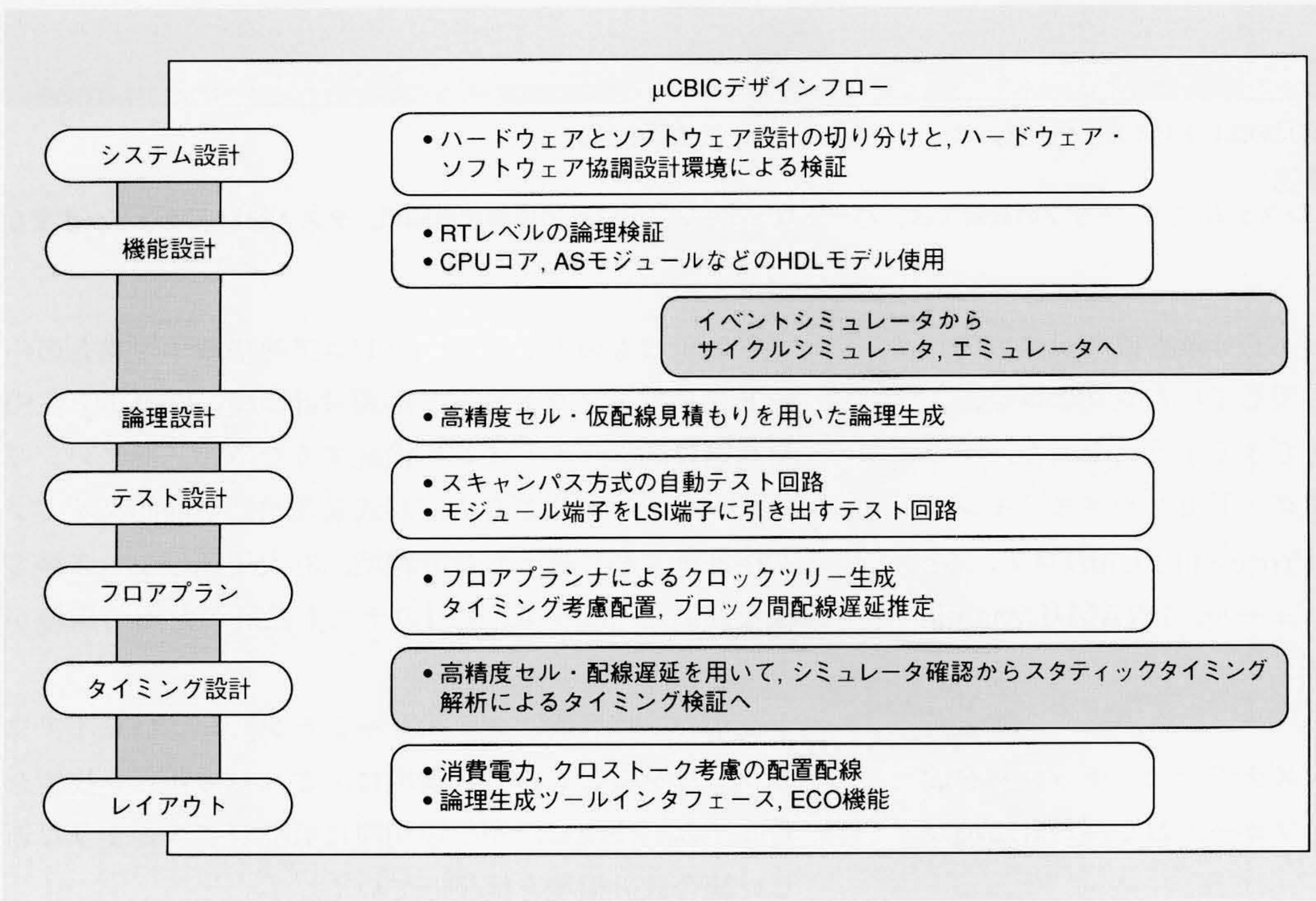
従来，論理LSIの設計では，ユーザーが設計対象とするブロックの論理を，論理図面上での接続で設計し，論理シミュレーションを用いて論理・タイミングの検証を行ってきた。

数年前からは，ブロックの論理設計を高位設計記述言語を用いて行い，セルレベルの論理接続を自動的に生成する，いわゆる論理生成技術が普及し，設計生産性が向上した。トップダウン設計手法の到来である。また，レイアウト技術も進歩し，タイミング制約を考慮したタイミング ドリブン レイアウト，クロックスキュー最小化技術を確立し，レイアウト後の設計手戻りをなくし，設計期間の短縮を実現した。

2.2 システム オン チップのデザインフロー

システム オン チップの設計では，設計生産性をさらに向上するために，既設計財産の再利用が積極的に行われる。これが，IP(Intellectual Property)モジュールと呼ばれるものであり，CPU(Central Processing Unit)コア，ASモジュール，DRAMモジュール，アナログモジュールなど多種多彩である。また，その提供元も半導体ベンダ，IPベンダなど多様である。

日立製作所は，マイコン(マイクロコンピュータ)製品の強みを生かしてSuperHやH8シリーズのマイコンを，



注：略語説明
RT(Register Transfer)
HDL(Hardware Description Language)
ECO(Engineering Change Order)

図1　システム オン チップのデザインフロー
μCBICのデザインフローの例を示す。システム設計支援と論理検証の高速化が今後の課題である。

さらに，DRAMモジュールをIPモジュールとして用意している。

μCBICのデザインフローの例を**図1**に示す。

2.2.1　システム設計

システム オン チップ，特にCPUコアを搭載するLSIでは，システム設計段階で，LSIのソフトウェアの設計が必要である。ハードウェア，ソフトウェアの切り分けをした後，ハードウェア・ソフトウェア同時シミュレーションにより，システム検証を行う。

2.2.2　論理検証

システム設計の段階，あるいはシステム設計完了後の論理設計の段階では，高速論理検証ツールが要求される。この段階ではタイミングまで考慮しない，論理確認を主眼にした検証を行う。論理エミュレータ，サイクル ベース シミュレータなどを用いる。

さらに，論理生成前後の論理等価性の確認には，形式論理等価性比較ツールなども使用し始めている。

2.2.3　タイミング検証

タイミング検証では，静的タイミング検証ツールを用いて，全パスのタイミング検証を高速に行う。論理の大規模化に伴い，シミュレータによるタイミング検証のためのテストパターン設計は不可能になっている。

この手法は，マイクロプロセッサの設計にすでに適用しており，μCBICへの展開を始めたところである。

2.2.4　テスト設計

多種多様なモジュールを搭載するLSIのテスト設計では，それぞれのモジュールを単体でテストできるような回路設計が要求される。こうしたテスト回路の生成とテストパターンの生成を自動的に行うツールを開発し，適用している。

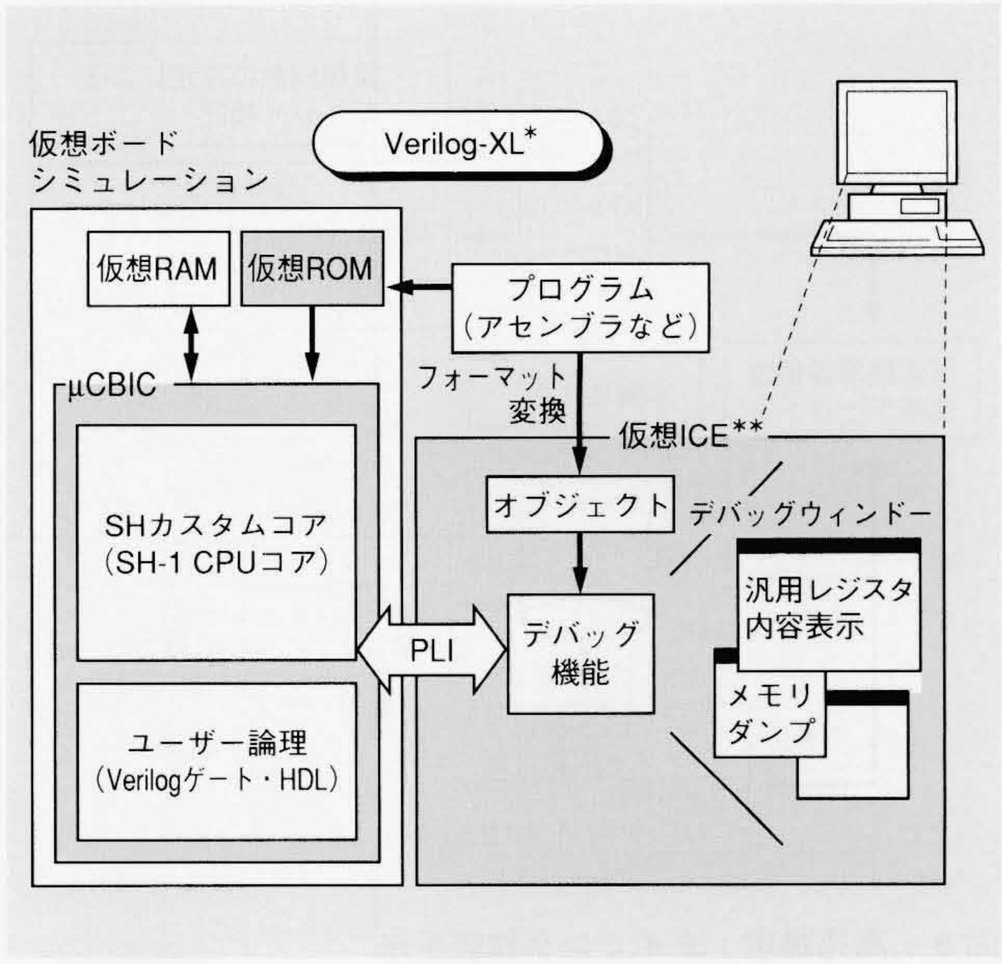
2.2.5　統合設計環境

これらの一連の設計作業を簡単にできるようにするため，一つのユーザーインタフェースの下に各設計ツールを統合した設計環境を構築している。

3. ハードウェア・ソフトウェア協調設計環境

CPUコアを搭載するLSIのシステム検証のために，同一のシミュレーション環境の中でハードウェアのシミュレーションとソフトウェアのシミュレーションを協調して行えるツールが，EDA(Electronic Design Automation)ベンダから市場投入されている。

ハードウェア・ソフトウェア協調設計ツールを**図2**に示す。



注：略語説明ほか
RAM(Random Access Memory)
ROM(Read-Only Memory)
PLI(Programmable Language Interface)
＊Verilog-XLは，米国Cadence Design Systems, Inc.の登録商標である。
＊＊仮想ICEは，横河電機株式会社の商標である。

図2　ハードウェア・ソフトウェア協調設計ツール
同一の市販EDAシミュレータの環境上で，ハードウェアとソフトウェアのシミュレーションを可能にした。

このツールでは，CPUコアの機能を記述言語でモデル化し，このモデルとアプリケーションプログラムをリンクすることによってハードウェア・ソフトウェアの同時シミュレーションを実現している。この環境を使用することにより，シミュレーション中に汎用レジスタやメモリの内容などのダンプが可能になるなど，ソフトウェアのデバッグに威力を発揮する。

また，検証が完了した後には，ハードウェア，つまりLSIの論理の部分をそのまま，以後の設計工程へ移行することができる。

幾つかのEDAベンダのツールに対して，SuperHマイコンコアのモデルがサポートされている。

また，μCBIC設計の場合には，このツールをソフトウェアのデバッグのためだけでなく，テストパターン設計用のツールとしても活用を試みている。

4. 高速論理・タイミング検証設計環境

論理の大規模化に伴い，論理・タイミング検証の工数は増大の一途をたどっている。また，検証漏れに伴う論理不良を発生させる可能性が高まっている。

システム オン チップの設計を短期間に行うために

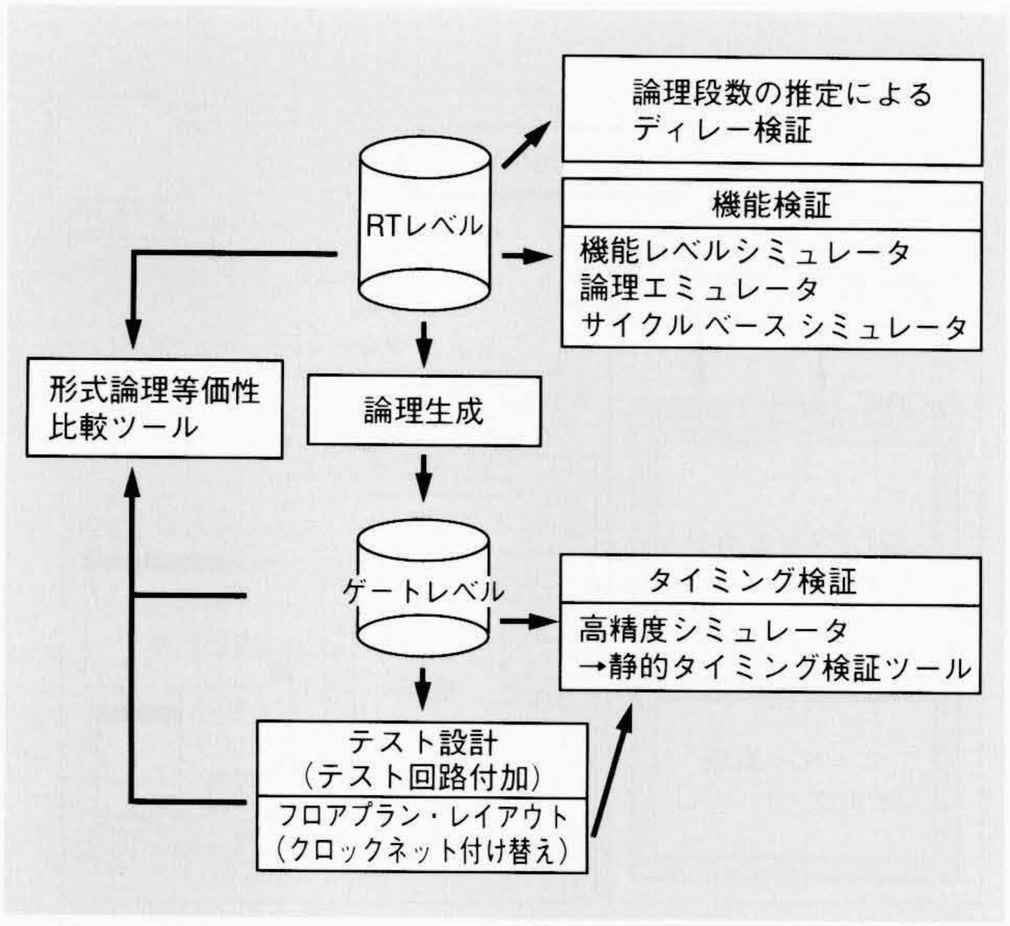


図3　高速論理・タイミング検証手法

RTレベルの機能検証をシミュレータで，ゲートレベルのタイミング検証を静的タイミング検証ツールで，さらに両レベルの論理等価性検証を形式検証ツールでそれぞれ実施する新しい検証手法を示す。

は，論理検証の効率をいかに上げるかがポイントとなる。ここでは，マイクロプロセッサの手法について述べ，システム オン チップの設計での手法を考察する。

高速論理・タイミング検証手法の例を**図3**に示す。

4.1　論理検証

マイクロプロセッサの設計では，数年前から，トップダウン設計を導入してきている。ブロックの論理設計を高位設計記述言語を用いて行い，セルレベルの論理接続を自動的に生成する論理生成技術を活用することにより，設計生産性が向上した。論理の検証には機能レベルシミュレータを活用しているが，機能検証用のテストパターン量は，一般に論理規模の1.5～2乗に比例し，このため，高速論理検証ツールが要求されている。マイクロプロセッサの設計では，大規模論理を高速に検証するため，論理エミュレータあるいはサイクル ベース シミュレータを利用し始めている。この段階では，タイミングまで考慮しない，論理確認を主眼にした検証を行っている。

さらに，論理等価性の確認には，従来のシミュレーションによる結果比較という方法ではなく，形式論理等価性比較ツールなども使用し始めている。論理生成前後，テスト回路付加前後，クロックネット付け替え前後など，RTレベルとゲートレベル間，ゲートレベル間どうしの検証に活用し，シミュレーション結果比較では1週間近く要したものを半日程度の検証で済ませている。

4.2　タイミング検証

論理検証完了後のタイミング検証は，静的タイミング検証ツールを用いてディレー・タイミング検証を高速に行っている。

静的タイミング検証ツールは，論理生成ツールの浸透に伴って普及してきている。ディレー検証では精度が重要であり，高精度セルライブラリ，レイアウトツールの配線特性を反映したライブラリを構築し，レイアウト前の検証を行うとともに，レイアウト後にも実配線長に基づくディレー計算を行って検証している。

動作周波数の向上に伴ってタイミング設計がますます重要になってきており，最近では，機能レベル設計時に高位設計記述の論理段数を推定することにより，ディレー検証を行っている。設計のより上流の段階で，タイミングを考慮した設計を進めている。

以上のように，マイクロプロセッサの設計では，機能検証は高速論理検証ツールを，タイミングの検証は静的検証ツールをそれぞれ活用しており，μCBICへの展開を始めたところである。

5. テスト容易化設計環境

完成したLSIは，LSIテスタ上でテストする。テスタではLSIの入力ピンから信号パターンを印加し，出力ピンから期待値どおりの信号が現れることを確認することによってLSIの良・不良を判定する。この印加パターンと期待値パターンを合わせて「テストパターン」と呼んでいる。製造上の不良品を取り除くには，あらゆる製造不良を検出できる高品質なテストパターンが必要である。

従来，テストパターンの設計は人手で行ってきた。しかし，大規模論理を1チップに搭載し，さらにIPモジュールを搭載するシステム オン チップでは，何らかのテスト容易化設計が必須となる。

日立製作所のμCBIC製品で採用しているテスト容易化設計環境について以下に述べる。

5.1　スキャンパス方式自動テスト設計

テストの自動化を困難にしている要素の第1は，論理回路がFF(Flip-Flop)のような内部状態を持つことにある。テスト時の内部状態の違いにより，同じ印加パターンから別の期待値が生じるので，テストにあたってはすべての内部状態を正確に把握し，制御する必要がある。特に，大規模で複雑な論理回路では，制御はいっそう困難になる。

スキャンパス方式自動テスト設計では，テスト時に内

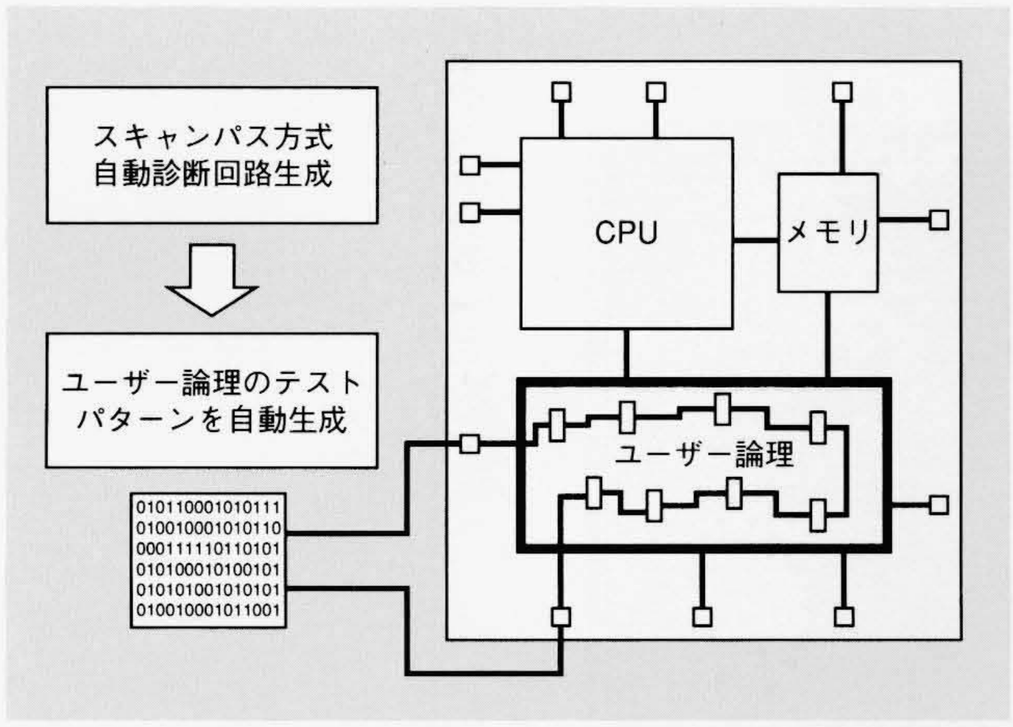


図4　スキャンパス方式自動テスト設計

スキャンパス方式自動テスト設計では，テスト時に内部状態を持つFFをテスト専用回路でシリアルに接続し，このパスを使用してFFの保持値を設定，観測することができるようにする。

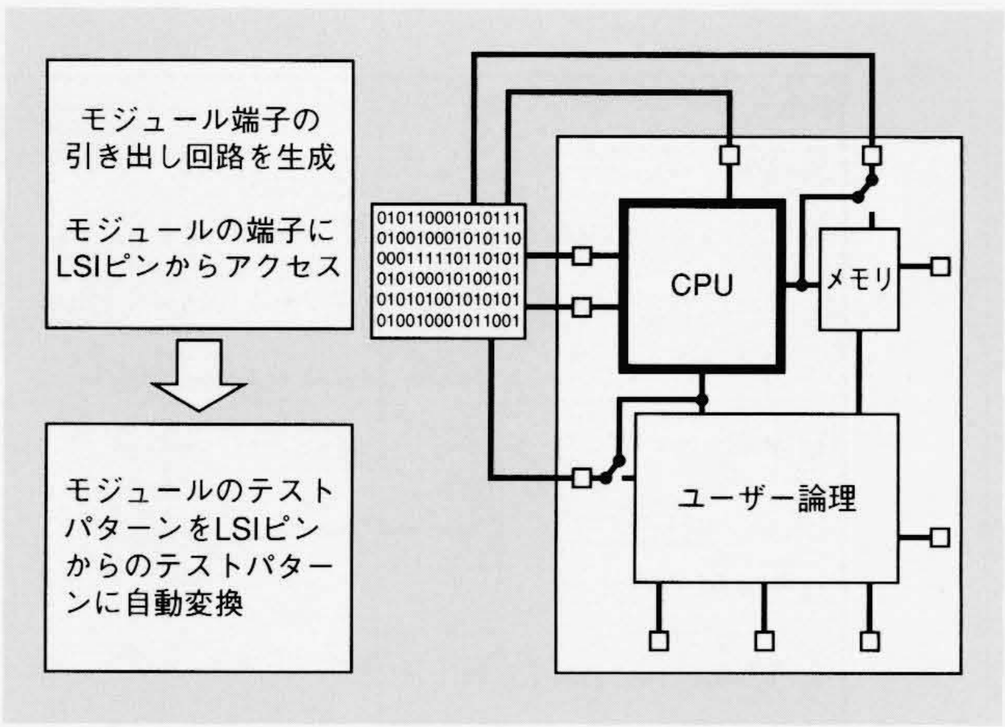


図5　モジュールテスト設計

モジュールテスト設計では，テスト時に内蔵したモジュールの端子をLSIの外部端子に引き出すテスト回路を自動付加し，既設計テストパターンを再利用することができるようにする。

部状態を持つFFをテスト専用回路でシリアルに接続し，このパスを使用してFFの保持値(LSIの内部状態)を設定，観測できるようにする。この状態では，LSIは内部状態がない完全な組合せ回路と見なせるため，DA(Design Automation)プログラムによってテストパターンを自動作成することが可能になる(**図4**参照)。

μCBICのテストでは，このためのテスト専用回路の自動付加・テストパターンの自動生成を行っている。また，独自のテスト回路を組み込んだFFを使用することにより，スキャン方式の制約であるクロック系の設計制約，使用できるFFの種類限定などの各種制約を無くしているのが大きな特徴である。

現在は，テストパターンをLSI内部で発生させ，LSIにみずからテストを行わせるBIST(Built-in Self Test)方式の検討を進めている。

5.2　モジュールテスト設計

μCBICでは，すでに汎用品としてラインアップされているH8SやSH3などのCPUコアを搭載できるが，これらについては，高品質なテストパターンを設計済みである。しかし，μCBICに内蔵されたCPUでは，端子を外部ピンから観測することができず，用意したパターンを使用することができない。

そこで，μCBICのモジュールテスト設計では，テスト時に内蔵したCPUなどのモジュールの端子を，LSIの外部端子に引き出すテスト回路を自動付加し，設計済みのテストパターンを再利用することができるようにしている(**図5**参照)。

6．デザインキット

システム オン チップを設計するために，セルライブラリや市販EDAツールを補完するアプリケーションプログラム群，および日立製作所が開発したDAプログラムなどを統合した設計環境，いわゆる「デザインキット」を開発した。

統一ユーザーインタフェースには，インターネットやイントラネットで強力なナビゲーションとアプリケーション実行機能を実現しているウェブブラウザを採用することで，メニューのカスタマイズやメンテナンスの容易化を実現した。HTML(Hypertext Markup Language)をベースとしているため，プログラム言語や専用のインタフェースファイル仕様を習得する必要がなく，ユーザー自身でのカスタマイズも可能であり，自由度を拡大した。また，アプリケーションやEDAツールの実行パラメータ設定ユーザーインタフェース部分には，仮想マシンJava※)を採用することにより，EWS(Engineering Workstation)やパソコンの多機種多様なプラットフォーム間でも，統一した操作環境を提供することを基本的に可能とした(**図6**参照)。

また，μCBIC設計に必要なマイコンモジュールを選択するだけで専用のカスタム マイコン コアを作成するマイコンコンパイラを開発し，適用している。マイコンコ

※)　Javaは，米国およびその他の国における米国Sun Microsystems，Inc.の商標である。

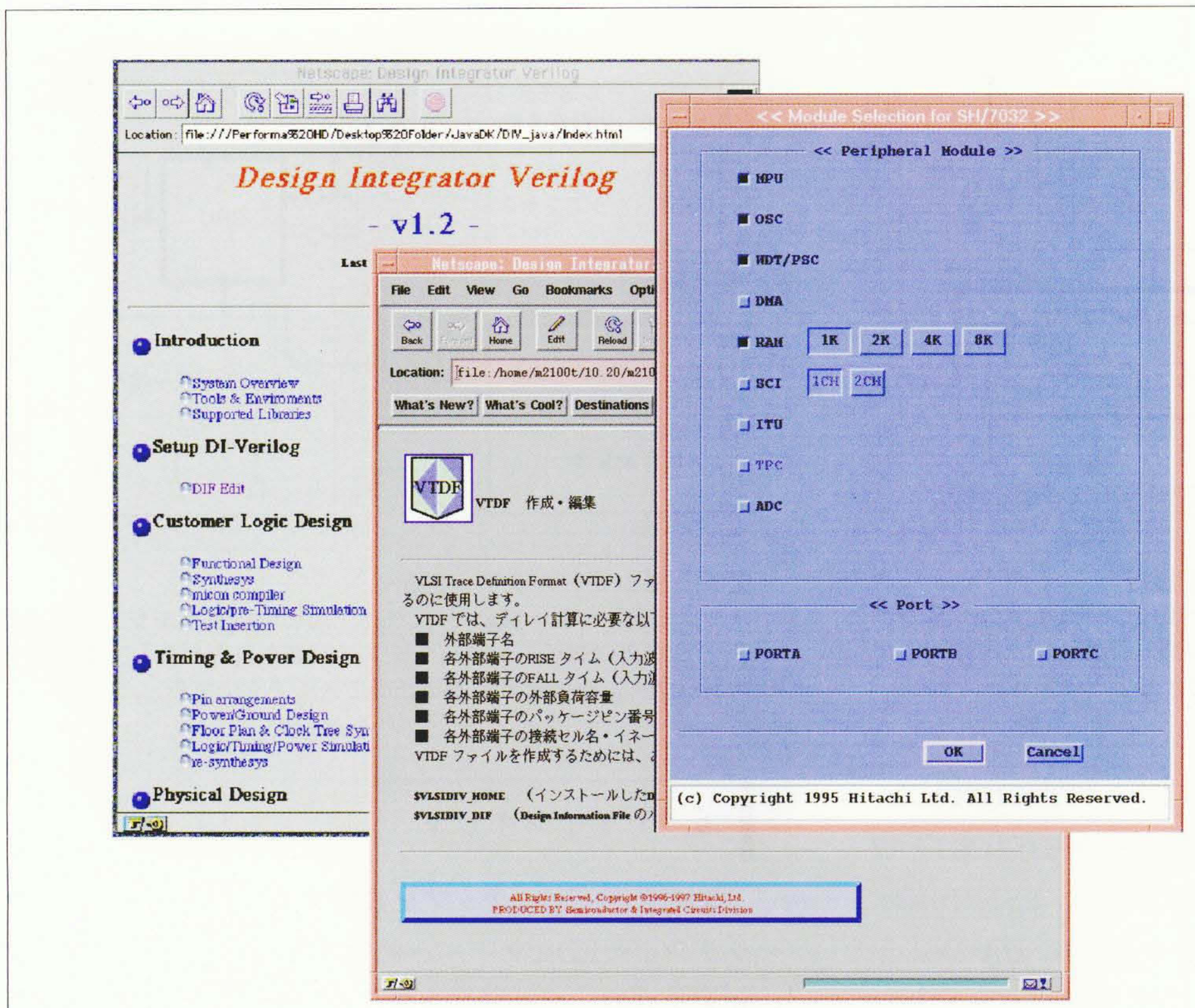


図6　統一ユーザーインタフェース

HTMLベースでカスタマイズが容易である。また，Javaの採用により，多機種多様なプラットフォームで使用することができる。

ンパイラの使用により，カスタム マイコン コアの論理シンボル，論理図面，ネットリスト，レイアウト用のシンボルなどを自動生成することができ，カスタマイズ期間を大幅に短縮できる。

7. おわりに

ここでは，システム オン チップのデザインフローとそれを支える新しい設計手法・環境について述べた。

この設計環境はようやく実際のLSI設計に適用され始めたところである。

微細化の伸展は，そのほかにも新しい課題を生み出している。消費電力の管理やクロストークの影響の考慮など，次の問題解決に向けた設計環境を検討し，またIPモジュールについては，VSIA(Virtual Socket Interface Alliance)準拠の具現化に向けて活動していく考えである。

執筆者紹介

大野康宣
1981年日立製作所入社，半導体事業部 マイコン・ASIC本部 設計技術室 所属
現在，ASIC用設計環境開発・適用サポートに従事
E-mail：ohno@cm.musashi.hitachi.co.jp

塩月八宏
1982年日立製作所入社，半導体事業部 マイコン・ASIC本部 設計技術室 所属
現在，ASIC用設計環境開発・適用サポートに従事
E-mail：shiotsuk@cm.musashi.hitachi.co.jp

野本和之
1980年日立製作所入社，半導体事業部 半導体技術開発センタ DA開発部 所属
現在，ASIC用DA・テスト容易化DAの開発に従事
E-mail：nomotok@cm.musashi.hitachi.co.jp

日下憲二
1981年日立製作所入社，半導体事業部 半導体技術開発センタ DA開発部 所属
現在，論理LSI用設計環境開発・DA適用サポートに従事
E-mail：kkusaka@cm.musashi.hitachi.co.jp