特集

論理等価性検証システム：ASSURE

Logic Equivalence Verification System: ASSURE

あらまし

本稿では，米国富士通研究所(FLA)で開発した，二つの回路の論理的な等価性を判定する論理等価性検証システム“ ASSURE ”について紹介する。

ASSUREは，複数の検証アルゴリズムを独自の発見的手法で組み合わせることにより，市販ツールと比較しても高速で，かつ高検証成功率を達成し，数百万ゲート規模の大規模な実設計にも適用可能な検証ツールである。ASSUREは，このほか，フリップフロップの自動対応付け機能や，誤設計診断機能といった，FLAで開発した最新の検証技術に基づく実用的な機能を備えており，検証時間の増大や網羅性の低下が問題になってきている従来の論理シミュレーションによる設計検証を補完し，100%の検証カバー率を保証する優れた設計検証環境を提供する。

Abstract

This paper introduces a logic equivalence verification system called“ ASSURE ”developed by Fujitsu Laboratories of America (FLA) for verifying logic equivalence in two circuits. ASSURE combines multiple verification algorithms with its unique heuristic method to realize faster and more successful verification than other tools in the market. This verification tool can therefore even be applied to designs as large as several million gates.

ASSURE has practical functions based on the latest verification technologies developed by FLA, for example, automatic flip-flop matching and diagnosis for design errors. It also provides a superior environment for design verification that guarantees 100% verification coverage by supplementing the conventional logic simulation used for design verification, which has been causing problems such as increasing verification times and decreasing coverage rates.

金持知己（かねもち　ともき）

1986年神戸大学大学院工学研究科電子工学専攻修士課程了。同年富士通入社。以来ディジタルシステム設計用CADの研究開発に従事。1996年から米国富士通研究所。
Advanced CAD Development Department

高山浩一郎（たかやま こういちろう）

1987年大阪大学大学院工学研究科電子工学専攻修士課程了。同年富士通入社。以来VLSI設計CADの研究開発に従事。1998年から米国富士通研究所。
Advanced CAD Research Department

論理等価性検証システム：ASSURE

ま え が き

論理設計の正しさを検証する工数は，今や設計工数全体の半分に達すると言われる。現在，論理シミュレータを用いた検証が主に行われているが，設計の大規模化に伴ってシステム検証に要する時間の増大が問題となってきている。一方，設計の正しさを数学的に完全に証明する形式的検証ツールが実用化され始めており，論理シミュレーションを補完する技術として注目を浴びている。米国富士通研究所(FLA：Fujitsu Laboratories of America, Inc.)で開発した論理等価性検証システム“ASSURE”は，二つの回路の論理的な等価性を判定する形式的検証ツールの一つである。

論理設計の流れの中でスキャンの挿入を行ったり，レイアウト後のタイミング調整のために全体の論理を保持したまま部分的な回路変更を行った場合，これまでは，論理シミュレーションを再実行して論理が正しく保存されていることを確認していた。しかし，設計が大規模になるほど論理シミュレーションでの検証の処理時間が増大し，網羅性は相対的に低下するので，誤りの混入を見逃す確率が大きくなる。一方，ASSUREを用いれば実用的な時間内で100%の検証を行うことが可能となる。例えば，70万ゲート規模の設計の検証を55分(メモリ750 Mバイト使用)で完了しており，市販ツールと比較してもトップクラスの性能を達成している。ASSUREは以下の特徴を備えている。

(1) BDD(Binary Decision Diagram),[1]およびATPG(Automatic Test Pattern Generation)[2]を独自の発見的手法で組み合わせることにより，高速性と高検証成功率(robustness)を両立。

(2) 市販論理合成ツールと独立したVerilog/VHDLパーサの採用により，論理合成ツールのバグも検出可能。

(3) 回路規模増加に対する線形な処理性能(実行時間，および使用メモリサイズ)により，数百万ゲート規模の実設計に適用可能。

(4) 新しいコマンド追加などの拡張や，他のツールとの統合が容易なユーザインタフェース(Tcl/Tkベースのコマンドラインインタフェース，およびGUI：Graphical User Interface)を採用。

以下，本稿では，まずASSUREのシステム構成について説明し，つぎにASSUREの検証コアで用いられている検証アルゴリズムについて解説する。続いて，ASSUREの実設計への適用を可能とした技術について述べる。

システム構成

ASSUREでは，論理シミュレーションなどにより，すでに正しいことが検証されている仕様設計(以下，Spec)に対して，論理合成や最適化，スキャン・クロック合成，レイアウト設計，ECO(Engineering Change Order)などを実施した後のインプリメンテーション設計(以下，Impl)が論理的に等価であるかどうかを，おおよそ以下の手順に従って数学的に証明する。

(1) SpecとImplの二つの設計を読み込む。

(2) 外部入出力ポート，階層インスタンスとその入出力ポート，フリップフロップ(以下，FF)などのステートポイントを対応付ける。これら入出力ポートやステートポイントを，検証ポイント，またはキーシグナルと呼ぶ。

(3) BDD法やATPG法を組み合わせて適用し，これら検証ポイントのペアが論理的に等価であるかどうかを検証していく。

(4) もしも等価でない(不一致)ペアが見つかった場合，

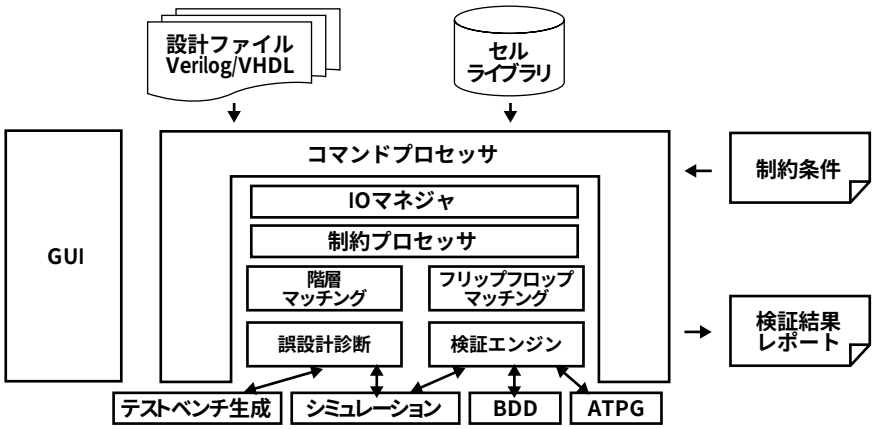


図-1 ASSUREのシステム構成
Fig.1-ASSURE system architecture.

以下の機能(i)，および(ii)を用いてこの原因(設計誤り)を調べる。

(i)　論理的な違いを観測できる論理シミュレーションパターン(識別ベクタ)を出力し，これを用いた論理シミュレーション結果を，GUIの回路図ウィンドウに表示する機能。

(ii)　論理シミュレーションに基づき，不一致ペアの原因である可能性が高いゲートやネットを捜し出し，その結果をGUIの回路図ウィンドウに表示，解析するエラー診断機能。

以上に述べた手順(1)～(4)，および機能(i)～(ii)を実現するASSUREのシステム構成を図-1に示す。ASSUREの核となる検証エンジンは，BDD，ATPG，シミュレーション，およびテストベンチ生成の各パッケージから構成されている。設計ファイルやテクノロジセルライブラリは，IOマネジャにより読み込まれ，回路モデルに変換される。検証ポイントの対応付け，論理定数値でのクリップ処理，ドントケア条件の指定など，論理検証の際にASSUREへのガイドとして与えられる様々な制約条件は，制約プロセッサを介して取り込まれる。階層マッチングでは，インスタンスやポートの名前に基づく自動対応付けが行われる。フリップフロップマッチング(以下，FFマッチング)では，名前によらないFFの自動対応付けが行われる。誤設計診断では，論理検証の結果，不一致となった検証ポイントのペアの解析機能を提供する。これらの各構成要素とユーザとのインタラクションは，Tcl/Tkベースのコマンドプロセッサ，およびGUIを介して行われる。

検証アルゴリズム

●　階層マッチング

階層マッチングでは，名前に基づく設計階層の対応付けを行う。対応付けの対象となるのは，外部入出力ポート，ブラックボックスや階層インスタンスとこれらの入出力ポート，FFなどの順序回路素子とこれらの入出力ポートである。ほとんどの場合，SpecとImplの間で，階層構造，および階層インスタンスや順序回路素子の名前が同じである，という仮定は成り立つが，論理合成ツールやレイアウトツールによっては，出力されるネットリストでは階層が平坦化されていたり，元の名前を含んではいるがプレフィックスやサフィックスが付加された名前に変更されていたりする，など，単純な名前による対応付けができない場合がある。そこで，ASSUREの階層マッチングでは，階層構造が一致していない場合に，これが一致するように階層の自動平坦化を行う，名前が完全に一致していなくても，一方が他方の部分文字列である場合には一致しているものと見做す，などの強力なマッチング機能が提供されている。さらに，一方が階層平坦化されており，そこに含まれる順序回路素子のインスタンスの名前が，階層平坦化前の階層パス名の区切り子として /(スラッシュ)以外の特殊文字(例えば$)を用いた連結名になっている場合には，一つのコマンドのみで対応付けができるようになっている。

●　FFマッチング

一般に，FFなどの記憶素子を含む順序回路の等価性検証は，組合せ回路のそれに比べて非常に多くの計算コストを要する。しかし，二つの論理回路中のFFが対応付けされる場合，順序回路の等価性検証は，対応するFFに挟まれた組合せ回路の等価性検証に置き換えることができる。これは，順序回路の故障テストにおいてフルスキャンによって組合せ回路用のATPGが利用できることと同様の考え方である。ASSUREは，名前や回路構造を用いた対応付け手法に加えて，BDDを応用したアルゴリズムを実装している。

【BDDベースのアルゴリズム】

与えられた二つの回路に同じ入力ベクタを適用して論理シミュレーションを行った場合，二つの回路中の対応するFFは各時刻において常に同じ値を取る。著者らはこの性質を利用して，BDDを用いて信号の一部を変数のまま扱った記号シミュレーションを行うことで，高精度な対応付けを高速に行うアルゴリズムを考案した。あるシミュレーション時刻で，k個の入力について変数のまま処理することで2^kパターンの論理シミュレーションを行ったのと等価な精度を得ることができる。

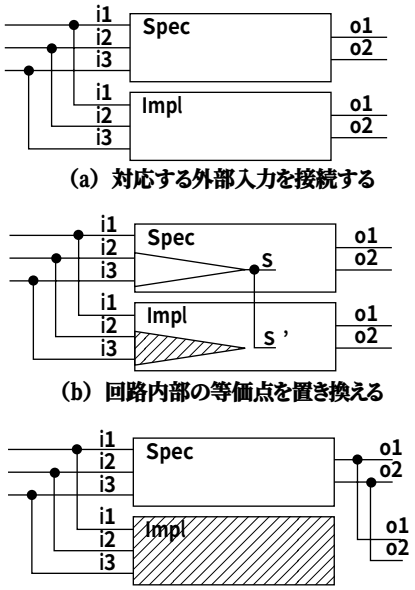
●　等価性検証アルゴリズム

ASSUREが実装している検証アルゴリズムは，組合せ回路を対象としている。概略アルゴリズムを(1)～(3)に示す。

(1)　与えられたSpecとImplの対応する外部入力を接続する {図-2(a)}。

(2)　外部入力から外部出力に向かって二つの回路の内部に論理的に等価な信号ペアを順次求めていく。等価な信号ペアが求まった場合には，一方の信号を他方の信号で置き換える {図-2(b)}。置き換えられた信号の入力側の論理コーンは，検証対象から除かれる(図中網掛け部分)。

(3)　対応する外部出力の等価性が証明されれば処理を終了する {図-2(c)}。



(a) 対応する外部入力を接続する

(b) 回路内部の等価点を置き換える

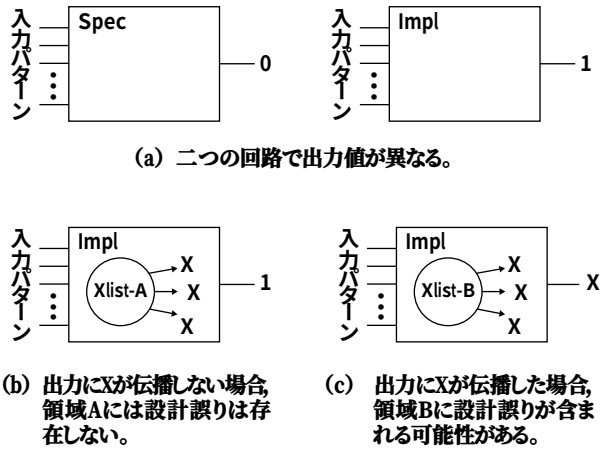
(c) 外部出力が置き換えられれば検証終了

図-2 検証の概略アルゴリズム
Fig.2-Verification algorithm.

(2)において，回路内部に論理的な等価点を求めていくことで検証問題を小さく分割している。したがって，アルゴリズムの面から見た検証の難しさは，回路規模ではなく，二つの回路内部にいかに多くの等価点が存在するかで決まる。つまり，小さい回路でも内部構造が全く異なる回路同士の比較は難しい問題で，大きい回路でも内部構造が類似しており内部等価点を多く含む回路同士の比較は容易な問題であると言える。これまで，(2)を効率よく処理するためのアルゴリズムの研究が広く行われており，ATPGベース[3]やBDDベース[4]のアルゴリズムが考案されている。経験的に一方のアルゴリズムを用いて検証できない場合にも，他方のアルゴリズムを用いて検証できる事例が少なくない。しかし，ある事例に対してどのアルゴリズムを適用すればより効率的な検証を行うことができるかは事前に判定できない。そこで，著者らは，いくつかの基本的な検証アルゴリズムを組み合わせて，問題の性質(難しさ)に合わせて順次強力なアルゴリズムを適用する検証フレームワーク[5]を考案しASSURE上に実装した。

● 誤設計(エラー)診断アルゴリズム

検証の結果，ある出力ペアが不一致であることが判明した場合，二つの出力の値を異ならせる入力パターンが生成される。設計者はこの入力パターンを用いて論理シミュレーションを行い，誤りのか所を特定する。従来は，シミュレーションの結果から誤りを特定する作業のほとんどは人手で行われていた。ここでは著者らが考案した多重の設計誤りを対象とした自動診断アルゴリズムを示す。



(a) 二つの回路で出力値が異なる。

(b) 出力にXが伝播しない場合，領域Aには設計誤りは存在しない。

(c) 出力にXが伝播した場合，領域Bに設計誤りが含まれる可能性がある。

図-3 Xlistに基づく誤設計診断
Fig.3-Error diagnosis based on Xlist.

現在，ASSUREの誤設計診断機能では，単一誤りのみを対象としているが，今後，多重誤りについてもサポートする予定である。

【設計誤りモデル】

本手法では設計誤りモデルとしてXlist[6]を用いる。Xlistは回路中の信号の集合として定義され，3値シミュレーションを行う際には，Xlist中の信号上に論理値X(不定値)が割り付けられる。

Xlistを用いた診断の様子を図-3に示す。(a)に示すように，ある入力パターンの下でシミュレーションを行った結果，SpecとImplの対応する出力の値がそれぞれ0および1であったとする。また，(b)に示すように，Aで示されるXlistを考慮した場合に当該出力値が1のままであったとすると，Aの中には設計誤りは存在しない。一方，(c)に示すように，Bで示されるXlistを考慮した場合に出力値がXになったとすると，Bの中もしくはその入力側に設計誤りが存在する可能性があることが分かる。

【診断アルゴリズム】

Xlistに基づく自動診断アルゴリズム[7]の概略を以下に示す。

(1) ある診断用入力パターンとXlistを用いて論理シミュレーションを行う。

(2) 外部出力に現れた値(0，1，X)をもとに，Xlistのスコアを計算する。いずれの非等価な出力にもXが伝播しなかったXlistは候補集合から削除される。より多くの非等価な出力にXが伝播したXlistがより大きなスコアを取る。

より大きいスコアを持つXlistが設計誤りを示している

可能性が高いことから，設計者はスコアの降順に実際にその中の信号の論理が誤っているかどうかをチェックする。

実設計への適用

● 大規模実設計の検証

ASSUREは，数百万ゲート規模の設計にも適用可能である。ピークメモリサイズは，70万ゲート程度の大規模設計検証の場合に約750 Mバイトであり，200万ゲート程度の超大規模設計の場合には約２Gバイトである。また，処理時間の目安としては，数万ゲート程度の小規模設計は数分以内，20万ゲート程度の中規模設計だと30分程度，70万ゲート程度の大規模設計は１～２時間程度，200万ゲート程度の超大規模設計は６時間程度で検証できる。いくつかの大規模実設計のASSUREでの検証結果を表-1に示す。

今後改良を重ね，さらに大規模な設計の検証を可能とする予定である。

● 効率良い検証のための設計制約

ASSUREは，主に論理検証処理のスピードアップを図ったり，ユーザが意図しているとおりの正しい論理検証を行うためのガイド情報を設定するために，いくつかの制約コマンドをサポートしている。階層インスタンスやポートの対応付けを指定するcorrespondコマンド，ポートの論理等価，および反転関係を指定するequivalence，およびcomplementコマンド，任意のポートに論理定数値を設定するlogic_0，およびlogic_1コマンド，特定の階層インスタンスやポートを論理検証の対象外とするskipコマンド，設計中に存在するドントケア条件を指定するdont_careコマンド，などである。制約プロセッサは，これらの制約コマンドを解析して制約データベースを生成する。これらの情報は，検証エンジンが論理検証を実行する際に参照される。各制約コマンドには，これと対の形で，すでに設定された制約を取り消すためのコマンドが用意されている。また，設定されているすべての制約条件をレポートするためのコマンドが用意されている。このコマンドは，論理検証の結果レポートされた不一致ペアが，誤った対応付けなどにより引き起こされていないかを確認する場合に有効である。ASSUREの制約プロセッサの特徴は，各制約コマンドの引数である階層インスタンスやポートを，正規表現を用いて指定できる点である。つまり，*(アスタリスク)などのワイルドカードを用いて，同じような名前を一度に指定できる。例えば，“ CLK_01からCLK_09 ”は，“ CLK_0<1-9>* ”と指定するだけでよく，９個の制約コマンドをいちいち指定する必要はない。

表-1　ASSUREの実行結果

データ	サイズ（kGates）	実行時間（分）	メモリサイズ（Mバイト）
DATA1	70	5	120
DATA2	170	12	280
DATA3	200	18	370
DATA4	700	55	750

（注）使用マシン： Ultra-60，２GバイトRAM，６Gバイト swap

● スキャン・クロック合成への対応

ASSUREは，スキャン回路挿入やクロックツリー合成前後の設計検証をサポートしている。スキャン設計手法としては，LSSD，MUXDを始め，JTAGバウンダリスキャンもサポートしている。スキャン回路挿入前後の設計の検証では，スキャン回路挿入前の設計と，通常のファンクションモードでのスキャン挿入後の設計を検証できなければならない。このために，スキャン・テスト制御ポートを定められた論理定数値でクリップし，挿入されたスキャンチェーンやクロックを検証の対象から外す必要がある。また，クロックツリー合成前後の設計検証では，階層インスタンス境界に，クロックポートが追加される場合が多く，これらの追加ポートを認識し，正しい対応付けができなければ，ユーザが期待しない論理不一致を招いてしまう。

ASSUREでは，ユーザによる手作業での制約コマンドを極力減らし，簡単にユーザの期待どおりの検証結果を得ることができるように，階層境界にとらわれずに論理定数値や論理等価，および反転関係を伝播しながら論理検証を進める，ソフトインタフェースモードを開発した。これにより，従来，各階層インスタンスのスキャン関連のポートに必要だった論理定数値やskip指示を不要とし，最上位モジュールのスキャン・テスト制御ポートに対する論理定数値，FFのSI，およびSO端子に対するskipコマンド，を指示するだけで期待通りの論理検証ができるようになった。

今後さらに，セルライブラリを整備することにより，FFのSI，およびSO端子に対するskipコマンドの指示も不要とする予定である。

む　す　び

本稿では，FLAの最新の検証技術に基づく，二つの回路の論理等価性を判定する「論理等価性検証システムASSURE」について，その実用性を示した。ASSUREでは，数百万ゲート程度の超大規模実設計も検証できるよう，複数の検証アルゴリズムを採用し，FFの自動対応付

け機能，誤設計診断機能，スキャン・クロック合成回路の検証機能などを備えている。

今後は，リタイミング設計の検証などの順序回路検証への機能拡張，および性能改善を行う予定である。

参考文献

(1) R. E. Bryant：Graph-based Algorithms for Boolean Function Manipulation. *IEEE Trans. on Computers*, C-35，pp.677-691 (August 1986).

(2) P. R. Stephan，R. K. Brayton，and A. L. Sangiovanni-Vincentelli：Combinational Test Generation Using Satisfiability. *IEEE Trans. on CAD/ICAS*，15，9，pp.1167-1176(Sep. 1996).

(3) D. Brand：Verification of Large Synthesized Design. in Proc. Int. Conf. Computer-Aided Design，1993，pp.534-537.

(4) Y. Matsunaga：An Efficient Equivalence Checker for Combinational Circuits. in Proc. Design Automation Conf.，1996，pp.629-634.

(5) R. Mukherjee et al.：An Efficient Filter-based Approach for Combinational Verification. in Proc. Design Automation and Test in Europe Conference，1999，pp.132-137.

(6) V. Boppana and M. Fujita：Modeling the unknown! towards model-independent fault and error diagnosis. in Proc. Int. Test Conf.，1998，pp.1094-1099.

(7) V. Boppana et al.：Multiple Error Diagnosis based on Xlists. in Proc. Design Automation Conf.，1999，pp.660-665.