特集

カバレッジ指向IPマクロインタフェース検証手法

Coverage-Driven IP Macro Interface Verification Method

あらまし

近年，大規模化・複雑化するLSI設計の加速化のために，既設計回路モジュール（IPマクロ）を積極的に再利用することで設計効率を上げる設計手法が注目されている。しかし，実際には「仕様のあいまいさ」「検証の不十分さ」などにより，思ったほど設計効率が上がっていない。とくにモジュール間インタフェースに絡む障害はファームウェアやIP利用方法の制限などでは回避不可能であり，深刻な問題としてとらえられている。そこで，著者らは，インタフェース仕様のあいまいさからくる検証の不十分さを解決する検証手法を開発した。

本稿で紹介する検証手法は，形式的インタフェース仕様記述言語Component Wrapper Language（CWL）でインタフェース仕様を記述することで仕様のあいまいさをなくし，そこから「何を検証すべきか」「どれだけ検証すべきか」といった検証用情報を自動的かつ網羅的に抽出し，形式的検証技術とカバレッジ測定技術の連携により網羅度の高い検証を実現する，という特徴を持っている。

Abstract

LSI design has become a larger and more complex task in recent years. As a result, an IP-based design approach has been attracting attention as a way to accelerate design work. In this approach, existing design circuit modules (IP macros) are reused to increase design efficiency. However, ambiguous specifications and insufficient verifications prevent the design efficiency from being increased as much as expected. In particular, errors caused by interface mismatch are considered to be a serious problem because they cannot be prevented simply by modifying the firmware or restricting IP usage. To solve this problem, we have developed a new verification method that resolves verification insufficiencies due to ambiguities in the IP macro's interface specifications. The verification method features the following: 1) clarification of the interface specifications by describing them in a formal interface specification language called Component Wrapper Language (CWL); 2) automatic, exhaustive extraction of information for the verification process such as the verification properties and degree of verification; and 3) linkage with formal verification techniques and coverage measurement techniques to achieve a highly comprehensive level of verification. This paper introduces the new verification method.

古渡　聡（こわたり　さとし）

デザインプラットフォーム統括部第一設計技術部 所属
現在，LSI設計/検証CADの開発に従事。

木村雅春（きむら　まさはる）

デザインプラットフォーム統括部第一設計技術部 所属
現在，マイクロプロセッサの開発を経て，LSI設計評価技術の開発に従事。

ま え が き

近年，大規模化・複雑化するLSI設計の加速化のために，既設計回路モジュール（IPマクロ）を積極的に再利用することで設計効率を上げる設計手法（IPベース設計）が注目されている。しかし実際には採用したIPマクロが，要求仕様を満足していない（機能障害），接続プロトコルが仕様どおりでない（インタフェース障害），などの問題により設計効率が思ったほど上がっていない。とくにインタフェース障害については深刻であり，ほぼ100％の確率でLSIのリメイクにつながってしまう。

図-1に示される障害を例にとってみる。この障害ケースは，転送データの正当性を示す信号val_oが転送完了後に「0」にセットされていないというものである。対象回路は送信するデータがないにもかかわらず信号val_oが「1」にセットされているために，接続されているもう一方の回路が「正しくない値」を取り込んでしまう。事前に検証でこの状況がチェックされていないと，結果としてシステム全体の動作が正常に行われなくなり，深刻なインタフェース障害になってしまう。著者らは，このような深刻なインタフェース障害をなくすため，回路がインタフェース仕様どおりに実装されているかどうかを漏れなく検証することを目的とした新しい検証手法を開発した。

本稿で紹介するこの新しい検証手法は，形式的なインタフェース仕様記述言語でインタフェース仕様を記述することで仕様のあいまいさを除去し，その記述から「なにを検証すべきか」を表す検証プロパティや「どれだけ検証すべきか」を表す検証網羅度（検証カバレッジ）測定用情報といった検証用情報を自動的かつ網羅的に抽出し，形式的検証技術と検証カバレッジ測定技術を連携させることでより網羅度の高い検証を実現する，という特徴を有する。

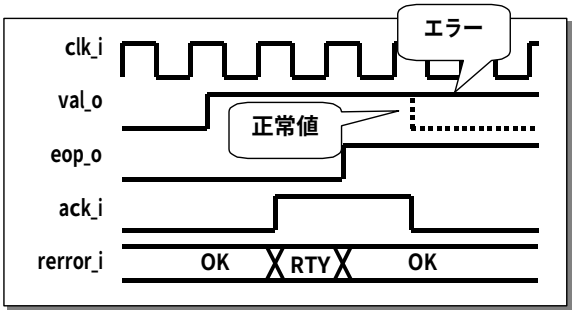


図-1　インタフェース障害例
Fig.1-Example of interface bug.

検証手法概要

● 従来手法の問題点

従来のインタフェース検証の問題点は以下の2点に集約される。

(1) 検証プロパティの十分性判定があいまい

従来から，インタフェース仕様書より「何を検証すべきか？」を表す検証プロパティの抽出を手作業で行い，検証ツールでのチェックはされてきている。しかし，抽出作業が手作業であるため十分な検証プロパティが抽出できたかどうかの判定ができないという問題がある。

(2) 検証のゴールがあいまい

インタフェースのチェックがどれだけ完了しているかという検証網羅性の判定基準がないため，コードカバレッジ(注1)や発見バグ数の傾向などにより検証の完了時期を決定している。コードカバレッジはインタフェース動作がどれくらい正しいかの基準にはならず，また，発見バグ数が減少し，たとえ収束したとしても，テストパターン不足が原因である場合もあり，検証十分性を必ずしも意味していない。

● カバレッジ指向IPマクロインタフェース検証手法の特徴

本稿で紹介するカバレッジ指向IPマクロインタフェース検証手法の特徴は以下の3点である。

(1) 形式的インタフェース仕様記述言語の採用

後述する形式的なインタフェース仕様記述言語（Component Wrapper Language：CWL[1),(2)]）によりインタフェース仕様を記述することで仕様のあいまいさを除去する。また，このインタフェース仕様記述は計算機で処理可能であることから自動的に検証プロパティやプロトコルチェッカを生成することで人的ミスを回避することが可能になる。

(2) 形式的検証とシミュレーション検証の連携

検証の網羅性は高いが検証可能回路規模が小さい形式的検証技術と，検証の網羅度は低いが大規模回

(注1) 回路の動作を記述したHDL（ハードウェア記述言語）コードのどの部分が何回実行されたかを表す検証の網羅性判定基準。すべてのコードが最低1回実行された場合を100％とする。

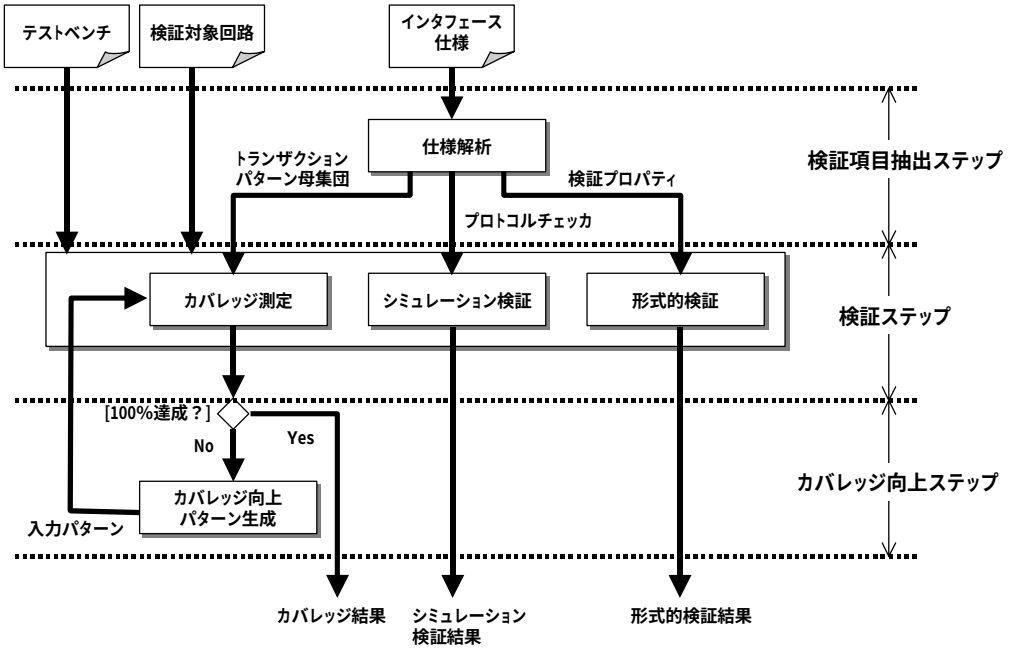


図-2　インタフェース検証フロー
Fig.2-Interface verification flow.

路に適用可能なシミュレーション検証技術を連携させることで検証規模に応じて可能な限り網羅度の高い検証を実現する。この連携は形式的なインタフェース仕様記述言語で記述されたインタフェース仕様から各検証技術に合った検証用情報（プロトコルチェッカや，検証プロパティ）を自動生成することで実現している。

(3)　明確な検証十分性判定基準

コードカバレッジのようなあいまいな基準ではなく，検証内容に直結したカバレッジ測定指標で検証十分性の判定を行う。具体的には「対象インタフェースで発生するトランザクション(注2)がどれだけチェックされたか」を表すトランザクションカバレッジを用いて十分性判定を行う。このトランザクションカバレッジの母集団は形式的なインタフェース仕様記述から自動生成する。

本検証手法のキーとなる形式的なインタフェース仕様記述言語とカバレッジ測定指標，また形式的検証技術については次章以降でより詳細に述べる。

●　検証フロー

本稿で紹介する検証手法のフローを**図-2**に示す。以下に示す三つのステップから構成されている。

(1)　検証項目抽出ステップ

インタフェース仕様から次の検証ステップ用の検証情報を生成するステップである。インタフェース仕様は形式的インタフェース仕様記述言語で記述し，シミュレーション検証にはプロトコルチェッカを，そして形式的検証には検証プロパティおよび入力制約をそれぞれ生成する。

(2)　検証ステップ

前ステップで生成された検証用情報を利用してシミュレーション検証および形式的検証を行うステップである。

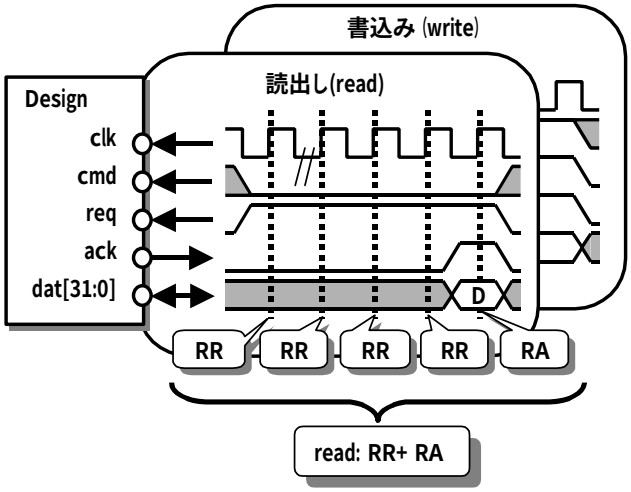
シミュレーション検証では検証以外に「どれだけのトランザクションパターンをチェックしたか？」を表すトランザクションカバレッジの測定も併せて行う。トランザクションカバレッジについては次章で述べる。

(3)　カバレッジ向上ステップ

検証ステップのトランザクションカバレッジの測定によりカバレッジ100%を達成していないインタフェースに対し，未検証のトランザクションを発生させるためのシミュレーション用入力パターンを自動生成するステップである。

検証ステップの形式的検証ですべての検証プロパティについて違反がないことが証明されたインタ

(注2)　回路がほかの回路からデータを取り込んだり送ったりするための一連の信号変化。



（a）タイミングチャート

```
interface FullHandshake;
  port;
    input.clock       clk;
    input.control     cmd;
    input.control     req;
    output.control    ack;
    inout.data[31:0] dat;
  endport
  word;
    read : RR+ RA;
    write: WR+ WA;
  endword
  sentence ;
   FOREGROUND:
     read ;
     write ;
  endsentence
endinterface
```

（b）CWL定義（一部）

図-3　タイミングチャートとCWL記述例
Fig.3-Example of timing chart and CWL description.

フェースについては完全に検証が完了しているため，このステップは不要である。

検証用情報の網羅的生成

ここでは本検証手法の特徴である検証網羅性の保証を実現する検証用情報の生成について述べる。

● インタフェース仕様記述言語CWL[(1),(2)]

従来からトランザクションパターンの表現としてタイミングチャートが用いられてきたが，この表現方法の欠点は大きく2点考えられる。一つ目は，表現したいトランザクション仕様の一例しか図示できない点である。この欠点により，タイミングチャートに記述されていないほかのパターンについては，そのタイミングチャートを見た人の解釈に依存し，仕様の誤解が発生するもととなっている。二つ目は，タイミングチャートが紙で書かれていることによりCADツールを利用する際に人手が介在してしまうというものである。これは，余計な工数の発生とともに記述ミスや漏れといった人的ミスが混入する危険があることを意味する。

CWLは，IPマクロのインタフェース仕様をコンパクトかつ正確に記述することを可能とする言語である。ここで「インタフェース仕様」と呼んでいるのは，I/Oピン情報などの構造情報だけでなく，I/Oピンにおける論理的な信号変化の仕様も含んでいる。

CWLにおけるトランザクションパターンの表現方法は「正規表現」をベースにしており，この記述方法がCWLのキーとなっている。正規表現は，文字列の集合をコンパクトに表現することに適しており，この表現方法を用いることでコンパクトかつ網羅的なトランザクションパターンの表現力を獲得している。CWLでは記述対象がトランザクションであるため「文字」の代わりに「1サイクル分の波形パターン」を用いている。

タイミングチャートとCWL記述の例を**図**-3に示す。図中の“RR”,“RA”が「1サイクル分の波形パターン」であり,“read”トランザクションを“RR”,“RA”の正規表現で表現しているのが分かる。

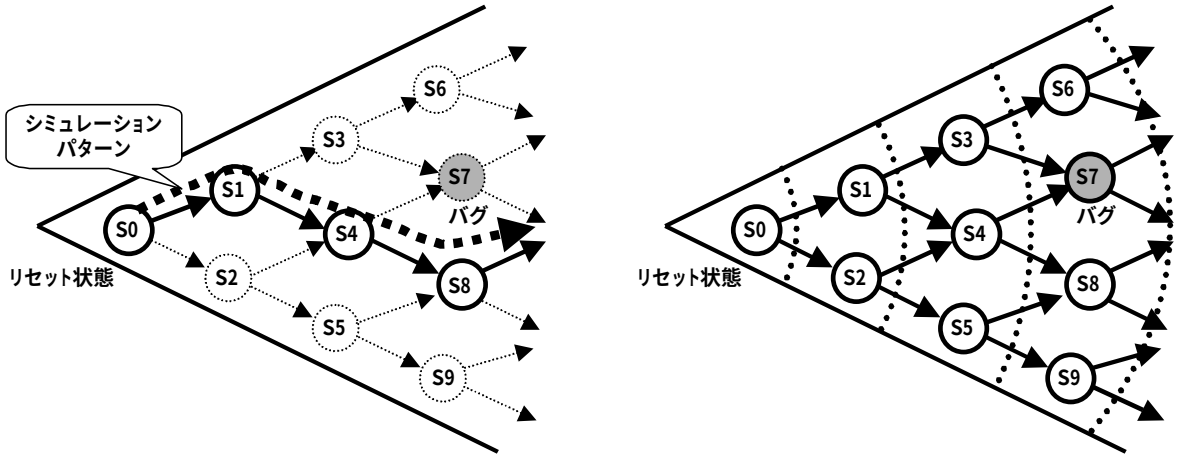
● CWLと検証用情報の生成

CWL記述は計算機で読込み可能であり，またインタフェース仕様のどの情報を記述しているかが明確であることから，検証ステップの検証ツール用に様々な検証用情報を抽出することが可能である。

(1) プロトコルチェッカ

CWLのベースになっている正規表現は歴史が古く，文字列が与えられた正規表現にマッチするかどうかをチェックする有限状態機械を作る手法は確立している。同じ原理で，対象インタフェースに発生する信号系列を監視し，仕様どおりに検証対象回路がプロトコルを制御しているかどうかをチェックするモジュールである「プロトコルチェッカ」を生成することができる。

このプロトコルチェッカをシミュレーション検証



（a）シミュレーション検証の探索空間　　（b）形式的検証の探索空間

図-4　シミュレーション検証と形式的検証の探索空間の違い
Fig.4-Difference between simulation verification and formal verification with respect to search space.

時に検証対象回路内に挿入するだけで，自動的にプロトコルチェックを行うことができる。

（2） 検証プロパティ

プロトコルチェッカ作成の過程で「Aが発生した後にBが発生することはない」などの情報が生成される。この情報をPSL[3]などのプロパティ記述言語で表現することで形式的検証ツール用の検証プロパティを生成することができる。形式的検証とCWLの関係については次章でより詳しく述べる。

（3） カバレッジ測定用検証情報

コードカバレッジのようなカバレッジ測定指標は，検証対象回路の設計仕様に深い知識を要しなくてもよいということから手軽に適用可能であり，広く用いられている。このような回路記述構造に対するカバレッジ測定指標は「回路の記述を網羅的にテストしているか？」を測ることに関しては有用であるが，「実装が仕様を満足しているか」を検証するために必要な「回路の動作を網羅的にテストしているか？」を測ることに関しては十分とは言えない。

インタフェース仕様どおりに回路が動作しているかを十分検証するには，やはり，「仕様で定義されているトランザクションパターンのどれだけが検証されているか？」を測る必要がある。このカバレッジ測定指標を本検証手法では「トランザクションカバレッジ」と呼び，この測定指標でカバレッジを測定するための母集団であるトランザクションパターンの総集合を，CWLで記述されたインタフェース仕様から自動的に生成する。トランザクションカバレッジの母集団は市販のコードカバレッジ測定ツールで測定可能な形で出力し，プロトコルチェックを含めたシミュレーション検証においてトランザクションカバレッジが測定可能となっている。

インタフェースの形式的検証技術

● 形式的検証技術の特徴

今日，形式的検証技術は大きく分けて二つのカテゴリが存在する。一つ目は二つの回路が等価かどうかを証明する「等価性検証」技術であり，もう一つは与えられた検証プロパティ（「AならばBになる」などの設計意図）どおりに回路が動作するかどうかを検査する「モデル検査[4]」技術である。本稿で形式的検証技術と呼んでいるのはこのモデル検査技術のことである。

シミュレーション検証では，まず検証者が設計誤りを起こしそうな部分を活性化させる入力列を作成して，その入力列で回路を動作させ期待された動きをしているかどうかを検証する。つまり，1サイクルにチェック可能な回路の状態は1状態となり，したがって，この入力列の質が重要となり，不注意な入力列生成を行うと検証漏れが発生する。**図**-4（a）は状態遷移図をベースにシミュレーション検証のイメージを表している。状態遷移図におけるS0，S1などのノードはその時点での回路の状態（回路内レジスタの値の組合せ）を表しており，入力により次のサイクルに移行する状態が異なる。シミュレーション検証はこの状態をテストパターンでなぞって

いくことに相当する。パターンが十分でなければ図-4（a）のようにS7のノード（バグ状態）に近いところまでチェックしていても障害を見落としてしまう。

これに対して形式的検証では，1サイクルごとに現在の状態集合と入力集合から到達し得るすべての次サイクル状態の集合をチェックする。したがって，シミュレーション検証のような検証漏れは発生しない｛図-4（b)｝。

● **形式的検証の問題点と解決法**

このように形式的検証は非常に強力な検証技術であるが，つぎのような問題がある。

(1) 検証可能回路規模が小さい。
(2) 検証可能回路の記述スタイルに制限がある。
(3) 入力制約を正しく与えないと擬似エラーを頻発する。

前2点の問題は満足いくレベルとは言えないが最近の技術の進歩により徐々に解決されてきている。しかし最後の問題については検証者が仕様書からインタフェース仕様を読み取って検証対象回路の入力列のルールを手作業で与えるしか解決法がなかった。

本検証手法では，前章で述べたCWLを用いて記述されたインタフェース仕様から，次サイクルにどのような入力がないとインタフェース仕様違反になるという情報を自動抽出し，形式的検証の入力制約として与えることで前述の問題を解決している。また同様に，次サイクルにどのような出力がなければならないという情報を検証プロパティとして自動抽出することで，人手作業では漏れがちな検証プロパティの網羅的な自動抽出を実現している。

カバレッジの自動向上

すべて新規の設計ならばそれほど問題にはならないが，少しでも既存回路を利用した設計を行うと問題になるのがカバレッジを上げるためのパターンの生成である。新規設計ならば設計者も回路のことをよく理解しており回路に目的の動作をさせるためにどのようなパターンを入力すればよいかが比較的容易に分かるが，過去に設計された回路は設計者が不在もしくは複数の設計者により変更されたなどの理由により新規設計回路のようには入力パターンの作成ができないことが多いからである。

本検証手法ではこの問題を形式的検証技術を利用することで解決している。今日市販されている形式的検証ツールでは与えられた検証プロパティが成立しない場合に反例パターン(注3)を生成する。この反例パターン生成機能を用いることでカバレッジを向上させるパターンの生成が可能となる。

本検証手法においてシミュレーション検証が終了した時点でトランザクションカバレッジの測定も終わっており，未検証のトランザクションパターンが明確になっている。この情報を用いて，形式的検証ツールに「トランザクションパターンAが発生することはない」というプロパティを与えることで，トランザクションパターンAを発生するための入力パターンを反例パターンとして生成することが可能となる。

このカバレッジ測定結果とトランザクションカバレッジの母集団から「トランザクションパターン発生用」のプロパティを生成するプロセスを自動化することで検証者の負担なしにカバレッジを向上させるパターンの生成を実現している。

表-1　インタフェース検証例

インタフェース	CWL行数	プロパティ数	障害数（仕様ミス / 設計障害）	
			シミュレーション	形式的検証
外部1	116	23	1 / 0	0 / 0
外部2	144	30	0 / 0	1 / 3
内部1	181	—	0 / 4	—
内部2	142	—	0 / 0	—

適　用　事　例

本検証手法の適用事例として，通信回路モジュールでのインタフェース検証事例を紹介する。

検証対象回路はホストCPU側に二つの外部インタフェースがあり，モジュール内部にも同様なインタフェースが二つあるというものである。CWLによる仕様記述としては4インタフェースすべてについて記述し，CPU側の二つのインタフェースのみ形式的検証を適用し，シミュレーション検証は4インタフェースすべてに対して適用した。

本検証手法の適用の結果，内部インタフェースにおいて4件，外部インタフェースにおいて5件のインタフェース障害が発見された（**表**-1）。また，修

（注3） 検証プロパティが成立しないことを具体的に示すための検証対象回路への入力パターン。

正回路に修正不備により同様の障害が混入したケースがあったが，それらについても同じ検証プロパティで検出できたという効果も得られた。このような修正後回路への障害混入はシミュレーション検証で発見することは非常に困難であり，形式的検証の有効性を示す結果であると言える。

また，トランザクションカバレッジの向上についても，設計側作成のテストベンチにおけるシミュレーションでは50%以下であったカバレッジ結果が，本手法の適用により60%～100%へと向上することができた。100%へ至るパターンの生成が不可能であったのは，対象インタフェースに関連する論理回路の規模が大きいというケースと目的としたトランザクションの発生に必要な初期化シーケンスが長いというケースがあったためである。これらはいずれも処理規模増大につながり，パターン生成のコアに用いている形式的検証技術の処理規模を超えてしまっていた。この処理規模の限界によるパターン生成不可の問題は形式的検証技術を利用する限り逃れられない問題であり，今後の検討課題である。

むすび

本稿では，IPマクロを利用する上で重要なインタフェースに関する検証に対し，検証プロパティの網羅性保証と明確な検証十分性の測定を行う検証手法とその有用性について紹介した。今回紹介した検証手法のポイントはやはり形式的なインタフェース仕様記述言語の採用にあると言える。これなくしては検証プロパティやトランザクションカバレッジの母集団の自動生成は実現できない。

しかし，今回紹介した検証手法にもいくつか課題が残されている。それは言語およびツール利用に対するスキルの問題である。言語に関する問題にはよく利用するインタフェース仕様記述のライブラリ化およびテンプレート化で対処し，形式的検証ツールなどのツール利用に対する問題には統合環境の整備という形で対処することで，より利用しやすい検証環境を構築していきたい。

参 考 文 献

(1) 鈴木敬ほか：インタフェースに着目したIP再利用設計手法“インタフェース・セントリック・デザイン”の提案．第15回 回路とシステム（軽井沢）ワークショップ，2002.04.22，p.299-304.
(2) 岩下洋哲ほか：インタフェース仕様記述言語CWLとその応用．第15回 回路とシステム（軽井沢）ワークショップ，2002.04.22，p.305-310.
(3) Property Specification Language Reference Manual Version 1.1，June 9，2004. http://www.eda.org/vfv/docs/PSL-v1.1.pdf
(4) K. L. McMillan：Symbolic Model Checking. Kluwer Academic Publishers，1993.