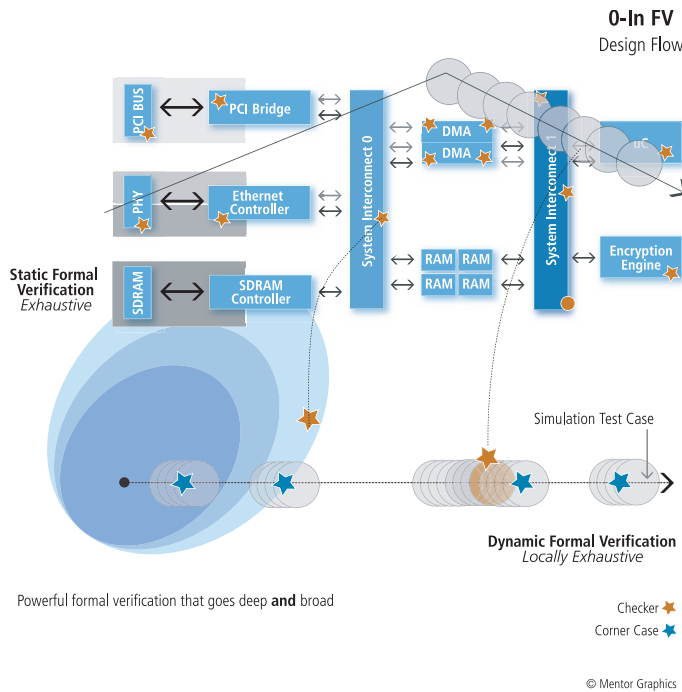


0-In Formal Verification



0-In Formal Verification はスタティックならびにダイナミックなフォーマル検証技術とシミュレーションを組み合わせることにより、業界で最も強力なフォーマル解析を提供します。

特長:

- ・業界で最も強力、大容量なフォーマル検証テクノロジー
- ・スタティック・フォーマル検証によりシミュレーションを行なうことなくブロックレベルのデザインを徹底的に検証
- ・ダイナミック・フォーマル検証によりチップレベル・シミュレーションを活用してコーナーケース・バグを発見
- ・Verilog、0-In CheckerWare、PSL、SVA、OVLを含む標準アサーション言語をサポート
- ・包括的なフォーマル・カバレッジ指標のサポートにより検証を支援
- ・検証結果、デバッグ・エラーを解析し、進捗を管理するための完成されたアサーション管理システム
- ・入力仮定の指定のための使いやすいアサーション・ベースの手法を提供
- ・マルチクロック、ゲーテッド・クロック、ラッチ、Xセマンティクス、合成不可能な構文をサポート
- ・既存のシミュレータを使ってアサーション違反を自動的に再現

メンターの0-In Formal Verificationソリューションは業界で最も強力なスタティックならびにダイナミックなフォーマル検証技術と最も包括的なABV (Assertion Based Verification) ソリューションを統合したもので、機能検証クロージャを迅速かつ高い予測性で達成します。0-In Formal Verificationソリューションは使い方も簡単で、ブロックレベル検証では従来のブロックレベル・シミュレーション・テストベンチの代わりとして、またチップレベル検証では詳細な擬似ランダム・シミュレーションの代わりとして、設計サイクルの全ての段階で効果を発揮します。

困難な検証課題に対応

0-In Formal Verificationソリューションは、業界で最も強力、最も包括的かつ最も幅広く適用されているプロパティ・チェックング・エンジンを提供しています。ユーザは複雑なロジック検証に適用し、徹底的な解析を行なわせることにより、コーナー・ケースの検証が十分に行なわれているとの確信を得ることができます。従来の手法では見逃されていたバグを迅速かつ簡単に発見することができます。

0-In Formal Verificationソリューションの適用が理想的な、困難な検証問題の例：

- ・標準インタフェース・プロトコル
- ・データ・インテグリティ
- ・アービトレーション・ロジック
- ・徹底したブロックレベルの解析
- ・バッファやフロー制御ロジック

使いやすいABV手法との統合

0-In Formal Verificationソリューションは0-Inの包括的ABVソリューションである0-In AssertionSynthesisとタイトに統合されています。設計者あるいは検証エンジニアはアサーションを1度指定するだけで、シミュレーション、フォーマル検証、エミュレーションで変更無しにこれを使用することができます。

数多くのユーザに採用されているこのABVソリューションは、改良を重ねることにより、高速かつ徹底した機能検証を実現しています。0-In Formal Verificationソリューションを含む0-In ABVソリューションは、アサーションの使用をより効率的かつ効果的なものとするための以下のような機能を備えています。

- ・徹底したフォーマル検証を行なうための強力なエンジン
 - スタティック・フォーマル解析—徹底した証明と反例を提供
 - ダイナミック・フォーマル解析—シミュレーションの活用によりクリティカルなコーナー・ケースをフォーマル解析
 - 業界で最も大容量なデータを扱う事が可能なフォーマル解析エンジン
 - 幅広い設計スタイルのサポート
- ・標準的アサーション言語の完全なサポート
 - Verilog
 - Property Specification Language (PSL)
 - System Verilog Assertions (SVA)

-Open Verification Library (OVL)

-0-In CheckerWare

・検証の進捗を記録する完成されたアサーション管理システム

-アサーション・ブラウザ

-アサーション違反ブラウザ

-トータル・カバレッジ・ビューワ

-リグレッション解析

・使い方はシンプル

-アサーションとCheckerWareを使用した使いやすい制約（入力動作の仮定）適用手法

-標準バス・インタフェースに対応した事前検証済みフォーマル制約条件

-シミュレーションからフォーマル検証までの詳細なカバレッジ情報

-パワーユーザ向けフォーマル・エンジンの詳細調整機能

0-In Formal Verificationソリューションは、フォーマル検証ツールによりバグを迅速に発見し、またシミュレーションだけでは見逃されてしまうバグを発見可能にすることにより、あらゆるエンジニアの生産性を向上させます。

業界で最も先進的なフォーマル・エンジン

0-In Formal Verificationソリューションには、高度にチューニングされた最新のフォーマル検証アルゴリズムが複数搭載されており、これにはスタティック／ダイナミック・フォーマル検証が含まれています。これらのエンジンにより実行されるフォーマル解析は業界で最も高速かつ徹底的です。スタティック・フォーマル解析は多くの場合ブロック・レベルで徹底したプロパティ検証に使用されます。ダイナミック・フォーマル解析は通常、チップレベルで使用されます。ダイナミック・フォーマル解析は、デザインに組み込まれたアサーションならびに0-In CheckerWareエレメントによりハイライトされたコーナーケース動作を監視

することにより、シミュレーションを活用します。そして、これらのコーナーケース・ステートに焦点を絞って徹底したフォーマル解析を行い、従来の手法では見逃されていたであろうバグを発見していきます。ダイナミック・フォーマル検証とスタティック・フォーマル検証を組み合わせるDirectedあるいはConstrained-randomシミュレーション手法を補完することにより、より効果的な検証を可能にし、検証クロージャまでの期間を短縮します。

0-Inのフォーマル検証エンジンは今日の設計に一般に含まれる幅広い構文を自動的に扱うことができ、これには複数クロック、ゲートド・クロック、メモリ、ラッチ、X/Zセマンティクスも含まれます。

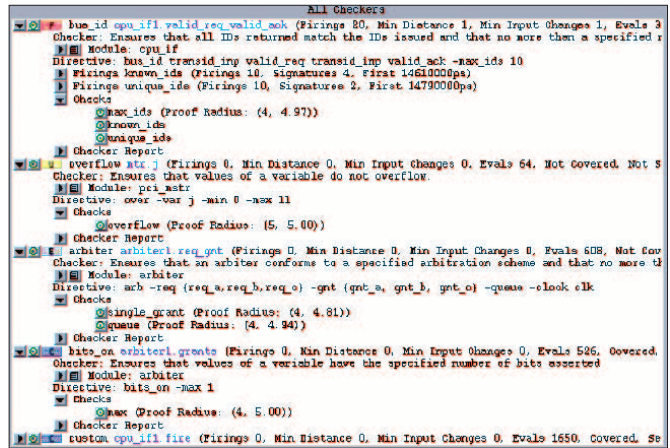
クラス最高の標準規格サポートとインターオペラビリティ

0-In Formal Verificationソリューションは、業界で最も幅広い標準アサーション言語ならびにライブラリのサポートを特長としています。0-In Formal Verificationソリューションは0-In Assertion Synthesisを活用してアサーション検証に優れたインターオペラビリティを提供します。PSLやSVAのような言語や、検証済みアサーション・ライブラリ、あるいはその組み合わせ等、アサーション形式は自由に選択できます。シミュレーションとフォーマル検証で全く同じアサーションを使用することにより優れた検証生産性が達成されます。

豊富な検証IPライブラリ

0-In CheckerWare Libraryを使用することにより、フォーマル検証を簡単に導入し、その効果を発揮させることができます。CheckerWare Libraryには一般的なRTL構造（ハンドシェイク、FIFO、アービタ、FSM）ならびに標準インタフェース（DDR-SDRAM、AMBA、PCI Express）等の様々な事前検証済み検証IPライブラリが含まれて

おり、0-In Formal Verificationソリューションに完全に統合されています。各CheckerWare Libraryエレメントはアサーションとして使用することも、入力の変数としても使用できます。



```
All Checkers
[+] bus_id cpu_ifi valid_req_valid_ack (Firings 80, Min Distance 1, Min Input Changes 1, Evals 3)
  Checker: Ensures that all IDs returned match the IDs issued and that no more than a specified...
  Module: cpu_ifi
  Directive: bus_id transid_imp valid_req transid_imp valid_ack --max_ids 10
  Firings known_ids (Firings 10, Signatures 4, First 1463000ps)
  Firings unique_ids (Firings 10, Signatures 2, First 1479000ps)
  Checks
    @max_ids (Proof Radius: (4, 4.97))
    @known_ids
    @unique_ids
  Checker Report
[+] overflow_mtr_j (Firings 0, Min Distance 0, Min Input Changes 0, Evals 64, Not Covered, Not 5)
  Checker: Ensures that values of a variable do not overflow.
  Module: pci_mstr
  Directive: over --var j --min 0 --max 11
  Checks
    @overflow (Proof Radius: (5, 5.00))
  Checker Report
[+] arbiter_arbiter_req_gnt (Firings 0, Min Distance 0, Min Input Changes 0, Evals 508, Not Cov)
  Checker: Ensures that an arbiter conforms to a specified arbitration scheme and that no more th...
  Module: arbiter
  Directive: arb --req (req_a, req_b, req_c) --gnt (gnt_a, gnt_b, gnt_c) --queue --clock clk
  Checks
    @single_gnt (Proof Radius: (4, 4.81))
    @queue (Proof Radius: (4, 4.94))
  Checker Report
[+] bits_on_arbiter1_gnts (Firings 0, Min Distance 0, Min Input Changes 0, Evals 526, Covered)
  Checker: Ensures that values of a variable have the specified number of bits asserted
  Module: bits_on
  Directive: bits_on --max 1
  Checks
    @max (Proof Radius: (4, 5.00))
  Checker Report
[+] custom_cpu_ifi_fire (Firings 0, Min Distance 0, Min Input Changes 0, Evals 1650, Covered, Se
```

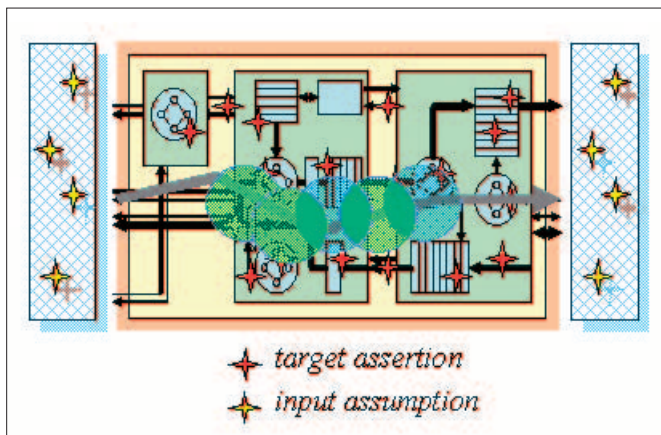
レポート環境0-In Viewにより各種カバレッジ指標を確認できます。

豊富なフォーマル・カバレッジ指標

フォーマル検証による解析は徹底していますが、フォーマル検証を使用するユーザとしては、「検証が十分かどうか」、また「次に何をすべきか？」という問いに答えるためのフィードバックが必要です。0-In Formal Verificationソリューションは業界で最も包括的なフォーマル・カバレッジ指標ならびに手法を提供することによりこれらの問いに対する答えを見出す助けとなります。

0-In Formal Verificationソリューションは各アサーションに対して「Proof Radius」を提供します。これは、「Proof Radius」により示されたサイクル数内ではアサーションに違反することは不可能だという指標です。通常、多くのアサーションに対して無限大の「Proof Radius」がレポートされます。それ以外のアサーションに対しては、「Proof Radius」をもとに、追加的フォーマル検証を行なう必要があるかどうかを判断することができます。

フォーマル・ツールにより全てのコーナー・ケースならびにアサーションの前提条件がカバーされるまで、フォーマル解析は終わりません。0-In



アサーションでは通常設計入力動作を指定します。

Formal Verificationソリューションではアサーションのコーナー・ケースならびに前提条件をフォーマル解析を使って直接ターゲットとすることができます。また、便利なテキスト形式のレポートや統一された0-In Viewレポート環境により、カバレッジのレベルがレポートされます。これらの構造カバレッジ・レポートはフォーマル検証テスト計画に穴がないかどうかを調べ（コンフィギュレーション・レジスタに追加的値が必要である、等）、テスト計画の改良に役立てることができます。

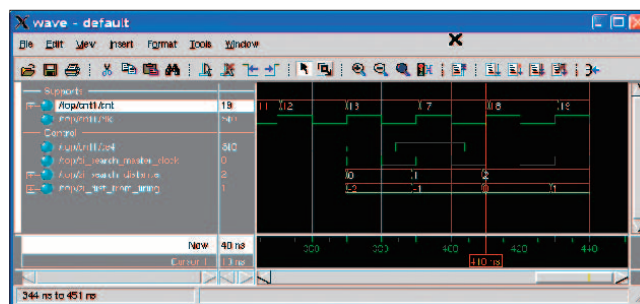
入力仮定の簡単な指定

全てのフォーマル検証ツールでは、ユーザがデザイン・バウンダリでの入力動作に対する仮定を指定することが必要です。そうしないと、フォーマル解析の発見したアサーション違反が本当のバグなのか、不正な入力動作に対応したものかわかりません。0-In Formal Verificationソリューションは、デザイン・バウンダリでの仮定の指定とその管理に対して業界で最も簡単な手法を提供しています。デザイン・バウンダリでの仮定はブロックレベルあるいはクラスタレベルで指定でき、0-In CheckerWare Libraryを含む任意の標準形式のア

サーションならびに言語が使用できます。これらのバウンダリ・アサーションは通常のシミュレーション実行時に適用され、アサーションに関する問題はシミュレーションを使って簡単に診断できます。ブロックあるいはクラスタを統合する場合にも、バウンダリ・アサーションはブロック間の通信に関わる問題を直接検出し、検証プロセスを大幅に迅速化します。バウンダリ・アサーションはフォーマル検証においてアサーション違反が本当のバグであることを保証するのに使用されます。

使いやすい、シミュレーションを中心としたデバッグ

0-In Formal Verificationソリューションでアサーション違反が発見されると、デバッグ用の波形が作成されます。また、既存のシミュレータを使って違反を自動的に再現しますので、使いなれたシミュレーション・デバッグ・ツールを使って問題を診断することができます。サポートされるシミュレータにはModelSim、VCS、NC-Simが含まれます。デバッグ・オプションには詳細なテキスト形式のレポート、0-In Viewレポート環境を使った色分け表示によるグラフィカルなデバッグ、ならびにModelSim、Verdi、Debussy等の波形ビューワが含まれます。



シミュレーション波形を使用することによりアサーション違反を簡単に診断することができます。

製品の仕様は予告なく変更されることがありますのでご了承ください。
Mentor Graphicsはメンター・グラフィックス・コーポレーションの登録商標です。
その他記載されている製品名はすべて各社の登録商標または商標です。

メンター・グラフィックス・ジャパン株式会社

本 社 〒140-0001 東京都品川区北品川4丁目7番35号 御殿山ヒルズ
電話 (03) 5488-3030 (営業代表)
大阪支店 〒532-0004 大阪市淀川区西宮原2丁目1番3号 SORA新大阪21
電話 (06) 6399-9521
名古屋支店 〒460-0008 名古屋市中区栄3丁目18番1号ナディアパークビジネスセンタービル
電話 (052) 249-2101
URL <http://www.mentorg.co.jp>

**Mentor
Graphics**

05/03-R1-1000-SI