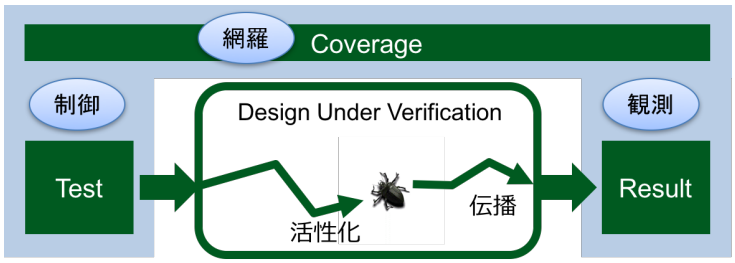
Functional Verification Technology Overview

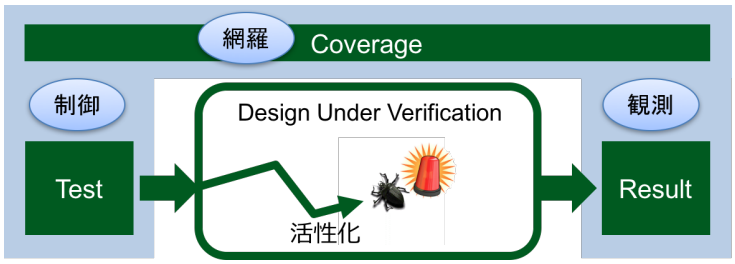
ASSERTION BASED VERIFICATION

アサーションベース検証とは

検証対象となるRTL設計の内部や入出力インタフェースに対し、仕様で定められている振舞いや、発生してはいけない振舞いを、複数信号の時相的な関係によって簡潔に表現したプロパティと検証ツールへのコマンドを組み合わせて記述し、RTLの振舞いとの比較によって検証する手法を指します。HDLシミュレーションやフォーマル検証がアサーションベース検証のツールの代表例です。VerilogやVHDLなど設計言語がIEEEで標準化されているように、アサーション言語もIEEE企画として標準化されています。SVA – SystemVerilog Assertions、PSL – Property Specification Languageがその代表的な言語です。2016年にワールドワイドで行われたWilson Research Groupの市場調査によれば、FPGAプロジェクトの47%においてアサーションが採用されており、その大部分はSVAを採用しています。



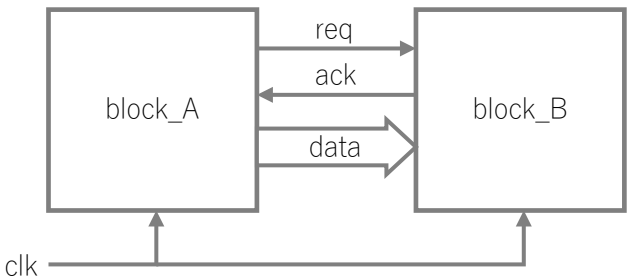
検証対象（Design Under Verification）にバグが混入している場合、テストパターンの役割は大きく2つに分けられます。1つめはバグを活性化させること、そして2つめはバグに起因する不正な振舞いを外部まで伝搬させることです。設計規模や複雑度が増すと、テストの制御性と観測性の維持が困難になります。



そこで多くのFPGAプロジェクトで採用しているのがアサーションベース検証です。アサーションを検証対象に注入することで観測性が上がり、バグ発生時の不正な振舞いを出力段まで伝搬させる必要がありません。バグが発生したその箇所、その時間でフェイルのメッセージが出力されるため、デバッグの高効率化が期待できます。

アサーション記述例

SystemVerilog Assertionsによるアサーション記述例を見てみます。対象はblock_Aとblock_Bのハンドシェーク部分です。アサーション記述はあくまでも仕様から起こすことが重要で、RTL実装の記述をベースにすると、同じ内容の異なる表現となってしまい、検証の意味がなくなってしまいます。

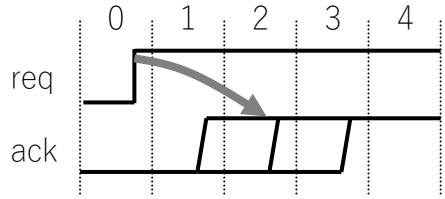


仕様：ハンドシェーク
reqが立ち上がったら1～3サイクル以内にackが返されなくてはならない

仕様は日本語や英語のような自然言語で表現されますが、自然言語には曖昧さが含まれます。この例でも1～3サイクルが、どのサイクルを起点とするかが明確ではありません。

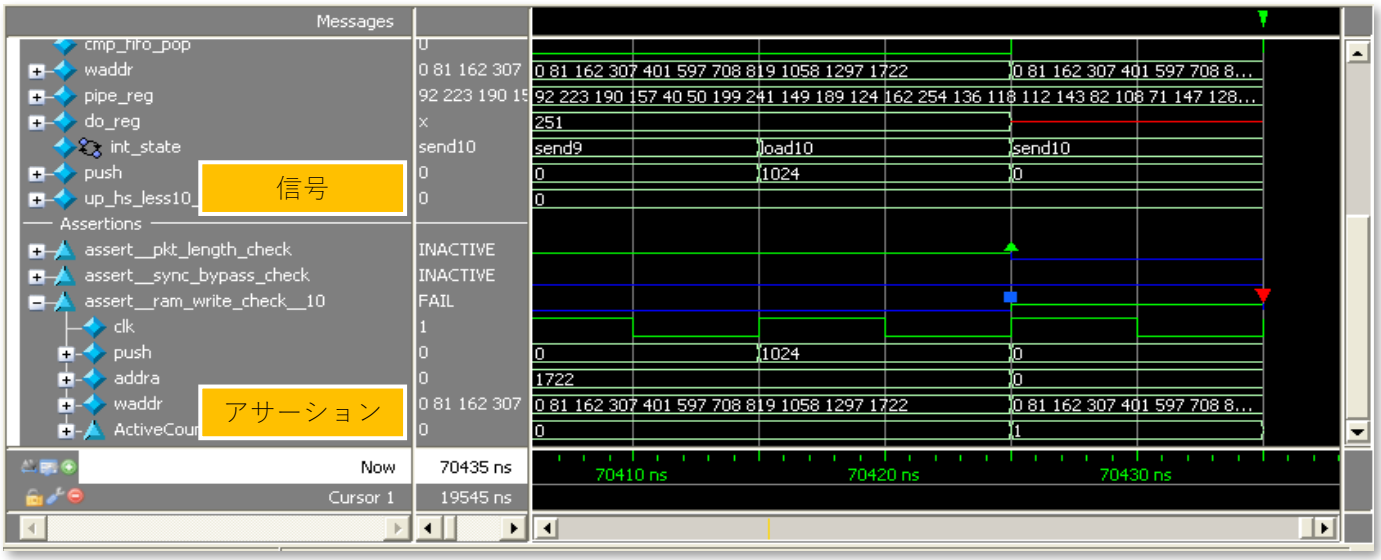
アサーション記述にはこの曖昧さがありません。以下のSVA記述はreqの立ち上がりが検出されたサイクルを0サイクル目とし、1~3サイクル以内にackが1となることを仕様として求める記述です。1サイクル目、2サイクル目、3サイクル目にack信号が1になれば仕様を満たし、a_req_ackと名付けたアサーションはパスし、それ以外の振舞いをした場合、a_req_ackはフェイルします。

```
property req_ack;
 @(posedge clk) $rose(req) |-> ##[1:3] ack;
endproperty
a_req_ack: assert property(req_ack);
```



シミュレーションでの扱い

シミュレーションではアサーションのパス／フェイルの情報を、波形やリスト、スレッドビューなど様々な表現で解析することができます。波形ウィンドウではAssertionsペイン内にアサーションを▲で表示します。波形上ではアサーションがパスすれば▲が、フェイルすれば▼が表示されます。



assert_pkt_lengthは70425nsでパスし、assert_ram_write_check_IOは70435nsでフェイルしていることが確認できます。各アサーション名の[+]をクリックすることで、そのアサーションを構成する信号レベルへと展開できるため、どの信号が原因でフェイルしているかを知ることができます。アサーションによるチェック／監視は、ちょうど波形の目視確認を自動化していると考えることもできます。

アサーションの価値

アサーションをRTL設計に追加することで、バグがあった場合にはその特定もデバッグも容易になります。あるバグをテストパターンが活性化し、その不正な振舞いを出力段まで伝搬するのに最短で10万サイクルかかるとすると、アサーションがこのバグを検出すれば、その10万サイクル分のシミュレーションは不要になります。機能検証のおよそ50%もの時間が費やされるデバッグ工数を削減できるだけでなく、シミュレーションのライセンス数も適正量に抑えられる可能性があります。

またバグがなかった場合であっても、波形を目視確認する労力を削減することができます。この検証労力削減の効果はそのプロジェクトでの省力化に限定されません。アサーションは検証資産として異なるプロジェクトや異なる部門で再利用することができます。さらに再利用性の高いブロックやIPの開発時に、その入出力インタフェース部分にアサーションを追加しておけば、IPとそれに接続する他のブロックとのプロトコルに関する問題が発生した際に、問題の切り分けが容易になります。

フォーマル検証の活用

アサーションはフォーマル検証でも使用できます。シミュレーションではテストパターンにより設計空間内の探索が方向付けられますが、フォーマル検証では初期値から開始して、設計空間が取り得る値を網羅的に探索します。そしてアサーションをターゲットにあらゆるステートの組み合わせを照らし合わせ、違反ないことを証明するか、違反があった場合にはその状態を再現するパターンを最小サイクルで提示します。FPGAプロジェクトでもフォーマル検証の採用が進んでいます。