第5章

SystemVerilog アサーション入門(前編)

内部信号のふるまいをツールが自動監視するデバッグ手法

赤星博輝

SystemVerilogの新しい特徴の一つとしてアサーション(assertion ;「表明」,「主張」を意味することば)への対応がある.アサーションでは,あらかじめ内部信号のふるまいを定義し,回路がそのとおりに動作しているかどうかをシミュレータに自動監視させる.デバッグの効率化や検証環境の再利用などに効果がある.また,アサーションの記述は形式的検証ツールの一種であるプロパティ検証ツールの入力データとしても利用できる.(編集部)

SystemVerilogでは,これまでVerilog HDLが弱かった検証機能を大幅に強化しました.例えば,制約付きランダム・テスト生成,機能カバレッジ,インターフェース,アサーションなどの機能が追加されています(p.84のコラム「SystemVerilogのここが良い,ここが悪い」を参照).

現在のLSI設計は,**図1**のように設計チームと検証チームが共同でボートをこぐ状況にたとえることができます.検証側が右側を,設計側が左側をこぐとすると,短時間でゴールするためには,検証と設計のこぎ手が協調してバランスよくこぐ必要があります.もし,右側と左側のこぎ手のバランスがとれていないとボートはまっすぐに進みません.現在は,どちらかというと設計に比重が置かれているため,ボートはまっすぐに進まず,蛇行しがちに見えます.

もし,検証力を高めることができれば,設計期間だけでなく設計品質などにも良い影響を与えることでしょう.しかし,ただ「検証力を強化しろ」と言われても,現実にどうやってよいのかと悩むかと思います.検証のレベル・アップに特効薬はなく,ひとつひとつ積み上げていくしかないと考えます.まず何をやってよいのかわからないときには,ここでご紹介する「アサーション」を導入してみることをお勧めします.

● アサーションを使う四つのメリット

LSIを設計するとき,設計した回路が正しく仕様を満たしているかどうかをチェックする必要があります.例えばカウンタを設計している場合,1サイクル進むと値がインクリメント(+1)されているかどうかをチェックする必要があります.このチェック項目が,カウンタという回路が満たすべき条件(仕様)になります.このような条件を「プロパティ」と呼び,このプロパティを検証言語(アサーション言語)などで記述したものをここでは「アサーション」と呼ぶことにします(p.86のコラム「SystemVerilogアサーションとPSLの比較」を参照).

検証で重要なことは,このチェック項目をはっきりさせることです.アサーションの導入でうまくいかない原因の一つに,チェック項目がはっきりしないためにアサーションが書けない,ということがあります.そのため,「アサーションを書く」という作業を通じて,設計や検証のやりかたを見直すことが重要です.

図1 設計と検証はバランスが重要

検証期間が延びているのは,設計が複雑になったからというよりも,設計と検証のバランスが悪くなったからだろう.設計者と検証エンジニアが力を合わせて,最速かつ最短時間でゴールに着きたいものである.

Column　SystemVerilogのここが良い，ここが悪い

筆者がSystemVerilogに注目し始めたのは2002年の初めごろです．世の中はC言語をベースとしたシステムLSI設計手法が立ち上がろうとしていたときですが，どうしてもCベース設計に向いていない領域があることを感じていました．その一方で，SystemVerilogは，ハードウェア設計者にとってたいへん魅力のある言語と感じられました．

SystemVerilogの言語仕様はどんどん改訂されています．Version 3.0からVersion 3.1aになり，LRM（language reference manual）は140ページから584ページへと厚くなりました．筆者が満足している点は，これまでVerilog HDLの弱かったところや問題があったところが大幅に改善されたことです．これにより，いろいろな言語を併用する必要がなくなり，実質的にはSystemVerilogとC言語でほとんどのことができるようになりました（とはいえ，C言語はやっぱり必要なのだが…）．

不満な点については，実はあまりないのですが，SystemVerilogで追加された機能の多くはこれまでほかの言語（検証言語など）でサポートされてきたものであり，それほど革新的な機能は含まれていません．設計者の側の習得の努力も必要ではありますが，EDAツールや言語の側についてもさらなる進歩を期待します．

また，日本語（おそらく英語でもそうなのだが…）で書いた文章にはあいまいさが残る場合があります．例えば，「処理Aから3クロック以内に処理Bを実行する」という仕様は，設計グループ内で誤解なく理解できるでしょうか？ 文章だけではできないはずです．例えば，「処理Aが開始されてから3クロック以内に処理Bが完了する」と解釈する人もいれば，「処理Aが完了して3クロック以内に処理Bを開始する」と解釈する人もいるかもしれません．これで誤解が発生しない場合は，その設計グループの中に暗黙の了解事項があるということになります．こうした暗黙の了解事項の量が多いと，新しいメンバが入ってきたときに問題が発生しやすくなります．このときに，検証言語などで仕様を記述していれば，仕様を明確に伝えられ，さらに，その仕様をツールによってチェックできます．

昨今の検証がたいへんなのは，設計する回路の規模が大きくかつ複雑になっていることが原因です．設計サイズが大きくなると，与えた入力に対して出力ポートの値を観測するだけでは，バグを発見できる可能性が少なくなってきています．せっかく内部でバグが発生する入力パターンを与えても，バグの値が出力ポートに出てこなければ，バグを発見できないからです．

もちろん気長に出力ポートまで値を導き出すテストベンチを作成すればよいのですが，検証をすばやく完了するためには，内部の値を確認することが必要になります．また，大規模な設計では，出力ポートの値がまちがっていた場合にどこが原因なのかを突き止めるデバッグ作業が必要になり，内部の信号を確認しながらバグの原因を特定していきます．

また，検証の課題の一つとして，作業効率の問題が挙げられます．例えば，シミュレーション結果を波形で確認する作業を行っている場合がありますが，ほんとうに人が画面に張り付いて，目視で確認する必要があるのでしょうか？ 問題点の一つは，人間はミスをするということです．大規模な設計では，人為的なミスが入る可能性をできる限り減らす必要があります．また，設計修正やシミュレーション・パターンの追加が生じることは珍しくないのですが，そのたびに設計者が波形を確認していたのでは，効率が悪すぎます．

アサーションを使うことによるメリットには，検証項目の明確化，検証環境の再利用，内部信号の観測によるバグの早期発見，検証の効率化といった四つのポイントがあります．今回は，このアサーションをシミュレータで利用する方法を中心に説明します．

● 即時アサーションはinitialなどのブロックで使う

SystemVerilogでは，即時アサーション（immediate assertion）と並列アサーション（concurrent assertion）の2種類のアサーションが導入されました．この二つの違いですが，**図2**に示すようにある時点の信号の値だけを見るのが即時アサーションで，複数サイクルにまたがる信号の値を見ることができるのが並列アサーションになります．

まず，即時アサーションについて説明します．

即時アサーションは，呼び出されたときにプロパティを満たしているかどうかをチェックします．構文は，以下のようになります．

```
assert （ <論理式> ） [<成立時に実行する文>]
                    [else <不成立時に実行する文>];
```

呼び出されたときにチェックするので，いつ呼び出すかをinitialブロックやalwaysブロックで記述する必要