初歩からのHDLテストベンチ

第10回　ランダム検証とスコアボード

安岡貴志

デバイスの記事　ビギナーズ

HDLで回路を記述できるようになったばかりで，これからテストベンチを書こうとしてる方を対象とした連載の第10回である．Verilog HDLとVHDLの両方のテストベンチの書き方を解説する．前回（本誌2008年12月号，pp.93-103の第9回）は，検証者がテスト入力を意図的に作り出した．しかし，想定外のバグに対しては，まったくケアできていない．検証者が意識し得ないバグまで，完全に取り去る方法は確立されていないが，そこへのアプローチの一つであるランダム検証について解説する．（筆者）

1．ランダム検証のための基礎知識

● 意識していないバグを炙り出すランダム検証

Verilog HDL・VHDL共通

ランダム検証とは，検証対象への入力をランダムに変化させ，その結果を確認するものです．この検証方法は検証者が意識していないコーナ・ケースのバグを発見できる可能性があります．

ランダム検証はテストの最初から行うものではありません．必ず検証者が意識できるバグを潰した後で，つまり単機能検証が終わってから行います．ランダム検証とは，いわば「当てずっぽう検証」です．最初から行うと，本来確認しなければいけないと分かっている項目を確認できないことがあります．

また，ランダム検証とはいっても，すべての値やタイミングを完全にランダムに変化させることはありません．完全なランダムに入力を与えてしまうと，仕様から外れていたり，そもそもその回路の使用目的から外れたりしてしまうためです．値やタイミングをランダムに振る場合でも，仕様から外れない範囲で制約をかけます．

さらにいえば，ランダム検証の際，本当のランダム，つまりシミュレーションを行うたびに値が変わるような使い方は，基本的にはしません．なぜならそのテストでバグが発見され，回路を修正してもう一度そのテストをしようと思っても，同じ状況を再現できないからです．多くの場合，疑似ランダムといわれる，ランダムっぽく値は変化するが，もう一度同じことをしたときに同じ値が出る方法を使います．

図1(a)は，シフト・レジスタを利用して疑似ランダム値を生成する回路です．この回路は，**図1**(b)のようにランダムっぽく値が変化した後，ちょうど255回目（8クロックのシフトで1回とする）で同じ値に戻ります．ただし，すべてのフリップフロップが0の状態から始めると，値がまったく変わらなくなります．このような構成で，8ビットのランダムな値を得られます[注1]．

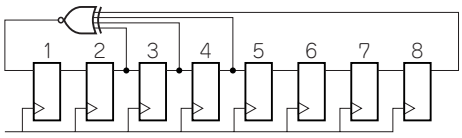
● 検証結果をファイルに書き込むスコアボード

Verilog HDL・VHDL共通

スポーツの試合などでは，その試合をすべて見ていなくても，スコアボード，つまり得点板を見ることで試合の状況や結果が分かります．テストにおけるスコアボードとは，期待値検証の結果をファイルに書き込むものと考えてください（本誌2008年9月号，pp.126-137の連載第8回を参照）．

注1：「原始多項式」について調べてみると，ほかのビット幅や疑似ランダム値が生成される理論について知ることができる．

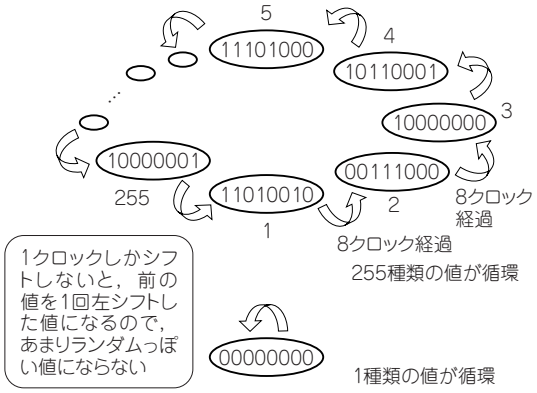
Keyword　Verilog HDL，VHDL，テストベンチ，ランダム検証，スコアボード，FIFO



●255クロック・サイクル・シフトを続けると8ビットの同じ値が表れる．
●8クロック・サイクルごとに値を見るなら，8クロック・サイクル×255回ごとに同じ値が表れる．
●XORゲートの入力は，原始多項式 $x^8+x^4+x^3+x^2+1$の係数が1である
次数に相当するレジスタ（フリップフロップ）の出力．

そのほかの原始多項式
10次 $x^{10}+x^7+1$
32次 $x^{32}+x^{31}+x^4+1$

（a）回路



（b）出力される値

図1　疑似ランダム生成回路
8ビットのランダムな値を得る例を示す．

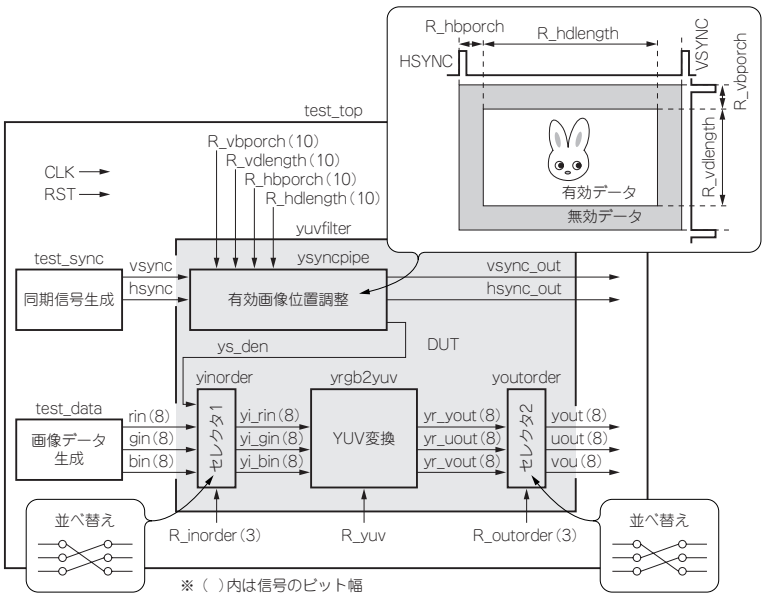


図2　検証対象の回路とテストベンチの概要
前回（本誌2008年12月号，pp.93-103の第9回）と同じもの．

テストにおいて，シミュレーション実行時にずっと人が張り付いていなくても，結果をファイルに書き込んでくれるスコアボード機能を作ることで，複数のテスト・パターン実行後に，結果と不具合の状況を確認できます．

ランダム検証では，検証対象のデータ処理系の機能，つまりデータが正しく変換されているかを確認するために，事前に期待値を作成しておくことが困難です．そこでリファレンス・モデルを，検証対象回路と並行して動作させ，期待値を出力させます．スコアボードは検証対象回路からの出力と，リファレンス・モデルからの出力を比較し，結果をファイルに落とします．

● FIFOの利用

Verilog HDL・VHDL共通

リファレンス・モデルとして利用されるビヘイビア・レベルのモデルは，基本的にタイミングの存在しない，一つの関数として存在します．つまり，入力値を与えた瞬間，クロック1周期の遅れもなく，演算結果が出力されます．

一方，検証対象回路であるRTL（Register Transfer Level）モデルでは，入力値が与えられた後，クロック数周期後に結果が出力されます．

このタイミングの異なる二つのモデルの期待値を比較するためには，出力の早い一方の値を格納しておき，必要なタイミングで読み出す仕組みが必要です．

このような目的のために，しばしばFIFO（First-in First-out）が利用されます．FIFOは，最初に書いたデータ順に読み出せるメモリ回路です．

2．ランダム値生成関数の記述

前回，画像生成回路のテスト環境を作成しました（**図2**）．このとき，画像データの入力値は固定値でしたが，今回はこれをランダムに変化する値に変えます．

Verilog HDL

リスト1は，前回の画像データ生成部分のモジュールにランダム値を生成する記述を加えたものです．`ifdefの文法を**図3**に示します．簡単にいえば，コンパイル時にマクロ名「RANDOM」を指定するとエリアAの記述がコンパイルされ，エリアBの記述はコンパイルされません．何も指定しないとその逆になります．エリアBは前回の記述と全く同じです．エリアAに書かれているファンクションRandomGenは，**図1**の方法を使って疑似ランダム値を生