VMM Verification Methodology Manual 活用テクニック

デバイスの記事

赤星博輝

第5回（最終回）　大規模回路のための検証環境を作成する

今回は，VMM（Verification Methodology Manual）に関する総仕上げを行います．これまでの検証環境では，入力ベクタを作成する部分と，ライブラリの使用法について説明してきましたが，チェックに関してはほとんど触れてきませんでした．そこで，これまでに紹介した機能を使って，連載第2回（本誌2006年10月号，pp.139-147）で示した領域判定回路をチェックする検証環境を作成します．　（筆者）

大規模・複雑化が進む最近の設計では，検証が大変になっています．このような状況について，ほかの分野でどうなっているかを考えてみるのも時には重要だと思います．

例えば，穴を掘るという作業を考えてみましょう（**図1**）．人はシャベルを持って穴を掘ることができます．しかし，これは掘れる穴の大きさに実質的な制限があります．大きな穴をシャベルだけで掘ることはできるのですが，効率の悪い作業になります．大きな穴を掘る場合に効率などを考えると，ショベルカーなどを使うことになります．このときに考えないといけないのは，ショベルカーを使うには免許が必要だし，ショベルカーを所持またはレンタルするコストもかかることです．

これを，検証に当てはめてみます．検証対象が小規模で，個人でできるうちはコストは大してかかりません．その代わり，できる作業が限られます．大規模な開発に対応するためには，ツールの導入が必要であり，そのために教育を行い，ツールや人員のレベル維持に費用が発生するということになります．

ツールなどは必要となった時に購入すればよいのですが，教育に関しては地道な取り組みをしておかないと，そう簡単にレベルアップできないところに注意しておく必要があります．

1．チェックの自動化を検討する

連載第2回（本誌2006年10月号，pp.139-147）の領域判定回路は，**図2**のように2次元（256 × 256）の領域に対して，X軸が50～100，Y軸が70～120の間に入っていれば‘1’と判定し，それ以外は‘0’と判定する回路でした．

これまでは，ランダム生成を用いて入力パターンを発生させて，波形ビューワで確認するところで終わっていました．しかし，現実にランダム生成で大量のパターンを自動

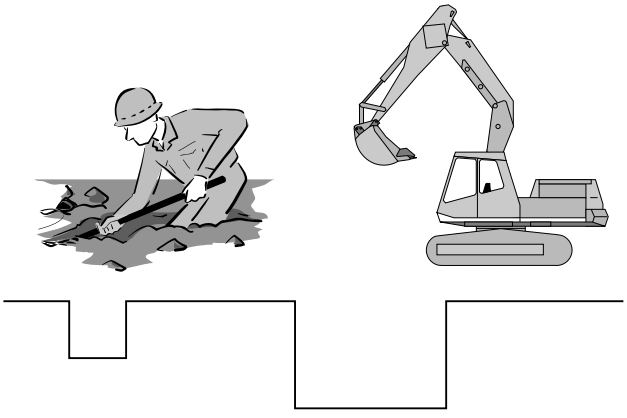

図1　世の中の動きと検証を対比して考える
やることが大きくなってくると，設備の導入とそれに伴う教育が重要になってくる．

KeyWord　VMM，検証，スコアボード，リファレンス・モデル，ランダム生成，テストベンチ，キュー，コールバック，レポート

発生させると，設計者が波形を見て動作確認を行うのは実質的に不可能です．

それでは，ランダム生成で作成した入力パターンをどのようにチェックすればよいでしょうか．ランダム生成を使うことが前提のVMMではスコアボードというチェックの自動化が必須になってきます．

● リファレンス・モデルを利用する

チェックの自動化は，どのように実現すればよいでしょうか．

これは，状況によって変わってきます．例えば，C/C++言語で詳細にアルゴリズムなどの検討を行っている場合には，**図3**のように，C/C++言語のモデルをリファレンスとして使用することができます．このように，あらかじめ検証済みのリファレンス・モデルがあればそれを利用できます．

リファレンスになるものがない場合には，どうするか考える必要があります．

ひとつのやり方として，設計とは別にリファレンス・モデルを開発する方法があります．このリファレンス・モデルの作成は，C言語でもHDLでも開発可能です．さらに合成を意識する必要がないので，合成を行うHDL記述よりも短い時間で作成できます．

しかし，いくつか問題点があります．

まず，同時に二つのモデル（設計とリファレンス）を開発するため，多くの開発リソースを必要とするという問題があります．最近の開発では，特に人的リソースが不足気味なので，なかなかこの手法が取り難い面があります（開発プランをうまく立て，うまくスケジューリングするなどの必要がある）．

また，リファレンス・モデルの作り方が問題になることがあります．例えば，検証用のリファレンス・モデル作成時に，設計側のコードと共用にすることがあります． 領域判定回路の検証で考えてみると，**図3**で判定する関数を共用しているような場合です．これもリファレンス・モデルだけで検証が済んでいればよいのですが，そうでないと問題が残っている場合があります．共用した場所にバグがある場合には，リファレンス・モデルとDUT（device under test）とを比較していても，バグを検出できません．

● ランダム生成を活用する

こういった時にはアイデアが必要です．実はランダム生成を工夫すると検証がやりやすくなる場合もあります．これまで領域判定の検証では，2次元の座標をランダム生成し，その座標値をドライバがDUTに投入していました．これを**図4**のように，領域内（HIT）と領域外（MISS）という二つの状態についてランダム生成を行い，次に，そのHITなら領域内の値でランダム生成し，MISSなら領域外の値でランダム生成します．このHITかMISSという情報をスコアボードに期待値として渡すことで，スコアボードには

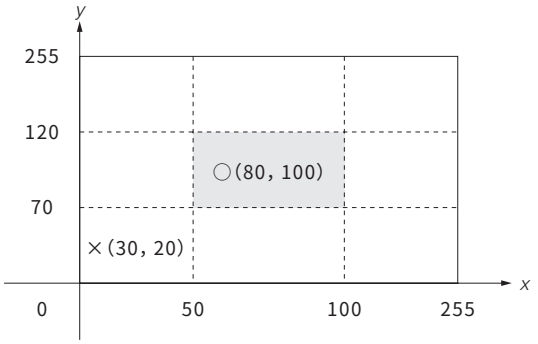

図2　領域判定問題

256×256の領域で，50≦x≦100，70≦y≦120に入っているか，入っていないかを判定する．

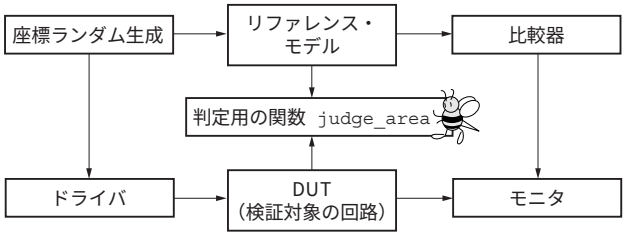

図3　リファレンス・モデルが使用できる場合

ランダム生成した値をリファレンス・モデルを使い期待値を作成し，その結果を比較することが可能になる．

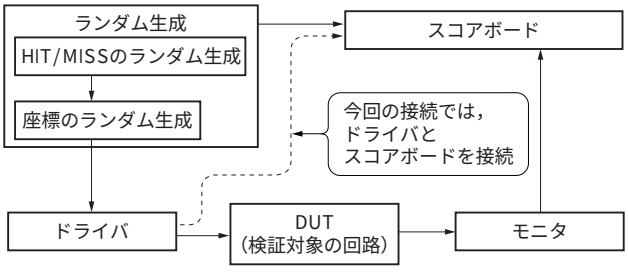

図4　ランダム生成を活用し期待値比較を容易に

ランダム生成も発想を変えると，スコアボードで簡単に出力値のチェックが可能となる．