VMM Verifivation Methodology Manual 活用テクニック

デバイスの記事

赤星博輝

第2回 テストベンチの作成にVMMの部品を利用する

検証ライブラリとその利用ガイドラインである“Verification Methodology Manual for System Verilog（VMM）”の活用法を解説する連載の第2回である．今回は実際にVMMの部品を用いてテストベンチを作成する．また，vmm_data，vmm_channel，vmm_atomic_gen，vmm_xactor，vmm_envの使いかたを説明する．検証のエキスパートが作成したVMMのエッセンスを理解することで，検証の再利用性や効率を引き上げることができる．（編集部）

前回は，SystemVerilogのオブジェクト指向に関する部分と，VMMを使う際にいろいろな場面で出てくるvmm_logについて説明しました．今回は，簡単な設計に対してVMMを使ってテストベンチを作成してみます．

VMMでテストベンチを作成するため，今回は五つの部品，すなわちデータを扱うvmm_data，テストベンチ間でデータのやり取りを行うvmm_channel，ランダム・パターン生成（以下，ランダム生成）を行うvmm_atomic_gen，処理を記述するvmm_xactor，テストベンチ環境を構築するvmm_envについて紹介します．

ところで，なぜVMMにはいろいろな構成要素があるのでしょうか？ それはテストベンチを効率的に設計するためです．前回説明したvmm_logを毎回作ると，かなりの工数を必要としますし，再利用しやすく作るのはさらにたいへんです．日常の検証作業で，次の開発プロジェクトのことを考えて行動する人はどのくらいいるのでしょうか？

VMMでは基本となる部品をあらかじめ定義しており，その部品をカスタマイズして使うという手法をとります．ただし，カスタマイズするときにコードをカット・アンド・ペーストして変更を加えると，だんだんわけがわからなくなってきます．VMMでは，オブジェクト指向言語の

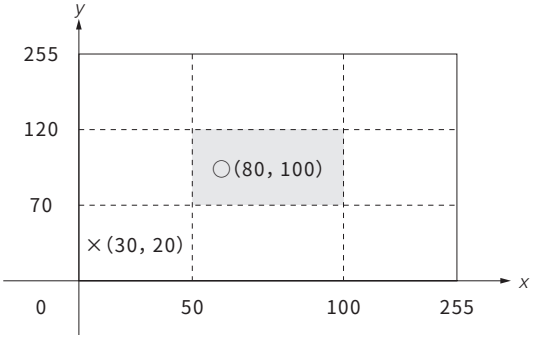


図1 ターゲットの判定回路の動き

256×256の中に(50，70)～(100，120)の長方形があり，その領域内であれば‘1’，領域外であれば‘0’と判定する．

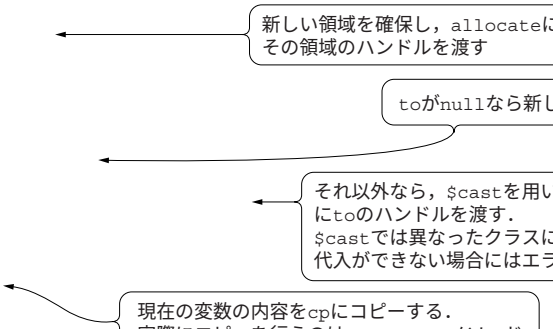


図2 ターゲットの入出力とそのタイミング

データを入力するときはienを‘1’にし，出力するときはoenが‘1’になる．

KeyWord SystemVerilog，VMM，vmm_log，vmm_data，vmm_channel，vmm_atomic_gen，vmm_xactor，vmm_env，ランダム・パターン生成，テストベンチ

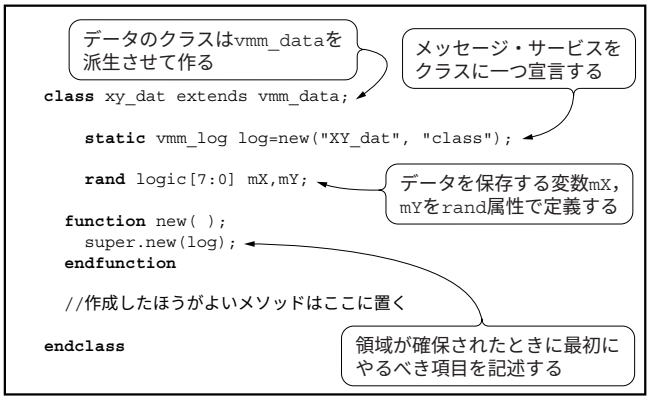
機能を使うことで，変更点だけを追加・変更できるようになっており，検証の再利用を促進します（下掲のコラム「VMMの歴史」を参照）．

● 今回の検証ターゲットはx，y平面の判定回路

VMMは，検証するための道具とそのガイドです．検証するものがないと説明しにくいので，検証のターゲットとしてここでは単純な判定回路を使用します．その回路とは，**図1**に示す領域に対して，「入力された点（x，y）が四角形の中にあるのか，または外にあるのか」を判定するものです．回路の入出力とデータの入力/出力タイミングを**図2**に示します．

この回路を検証するためにどのようなデータが必要かを考えます．機能としては*X*座標と*Y*座標を与えて領域外/領域内の判定を行うものなので，*X*座標と*Y*座標の二つの数値が必要です．ほんとうはienも必要なのですが，これはあとで考えます．

リスト1　データのクラスの基本
メッセージ・サービス（vmm_log），変数，newの三つが最低限必要．



● 検証で扱うデータはすべてvmm_dataから派生させる

VMMでは，検証で扱うデータをvmm_dataから派生させて作ります．今回は，二つの数値を記憶するために，vmm_dataから派生させたxy_datというクラスを作ります．このvmm_dataから派生したクラスを構成するうえで最低限必要なものは，**リスト1**に示すようにvmm_log，必要な変数の定義，newメソッドの三つになります．

VMMのメリットはこれまでの検証の知識がその部品の中に埋め込まれているところで，vmm_logはVMMの「Rule 4-58」にガイドとして示されているように，静的（static）なメッセージ・サービスとして宣言します．これは，多くのデータが処理されるときに，個別にメッセージ・サービスを作ると負荷が重いことなどが理由です．

コラム　VMMの歴史

VMMの歴史は，著者のひとりであるJanick Bergeron氏の検証への取り組みの歴史ともいえます．Janick Bergeron氏からVMMの歴史について紹介していただいたので，同氏のコメントを以下に紹介します．

「まずは1991年にセルフ・チェック機構をもつ，トランザクション・レベルのテストベンチを使い始めました．この段階ではVerilog HDLでモデルを記述し，第1世代のvmm_log，vmm_data，vmm_env相当のものが作成されました．このときはダイレクト・テストで物理レベルのトランザクタを使用し，バックグラウンドのノイズにはランダム生成も使いました．1994年にSONET（synchronous optical network）伝送の検証でより高い抽象度のトランザクタが必要となったのですが，Verilog HDLでは不十分なものしか記述できませんでした．

その後，Veraやe言語の登場によりオブジェクト指向的な実装が可能となり，1998年に第1世代のvmm_channelを作成しました．

2001年に（Janick氏がChief Technical Officerを務めていた）米国Qualis Design社が検証IPを提供したときは，すべてのトランザクタがvmm_xactorをベースにしており，vmm_logもほぼ現在の形になっていました．トランザクタもコールバック・メソッドを提供するようになり，ユーザによる機能の拡張や修正が可能となりました．VMMで重要なランダム生成におけるファクトリ・パターンも実装されました．

2002年の終わりには，ほぼ今の形のvmm_envを作りました．これは，検証環境を構築するために明確なガイドラインが欲しいということから作成しました．ほかにも，顧客からの要求に基づいてvmm_broadcastやvmm_schedulerなどが作成されました．

2003年に米国Synopsys社に所属することになり（Qualis Design社がSynopsysに買収された），VeraでRVM（Reference Verification Methodology）の実装を行い，vmm_notify，vmm_atomic_gen，vmm_scenario_genなどを追加していきました．2004年にVeraのRVMをSystemVerilogで実装し直し，2005年にVMMを書籍として出版しました」．

このように，Janick Bergeron氏は15年以上もの間，検証にかかわってきました．検証に対して積極的に環境を改善し，その活動の集大成がVMMということになります．とても参考になる部分が多いので，みなさんもぜひ，その知識に触れていただきたいと思います．