Artgraphics

UVM 入門

Document Revision: 1.9, 2019.04.06

アートグラフィックス
篠塚一也

UVM 入門

〒124-0012 東京都葛飾区立石 8-14-1
www.artgraphics.co.jp

UVM Primer

8-14-1, Tateishi, Katsushika-ku, Tokyo, 124-0012 Japan
www.artgraphics.co.jp

はじめに

この入門書は UVM の概要を簡単に纏めた学習用の素材です。UVM の基礎を把握する事を主眼にしている為、詳細の機能については触れていません。この入門書を読了後に、UVM User's Guide（[2]）を熟読する事を薦めます。或いは、UVM の文献（例えば[4]）を読む事を薦めます。

誤字訂正、及び、説明の補足を随時行います。従って、常に本書の内容は更新を余儀なくされます。ご了承ください。

尚、本書の記述は UVM 1.2 をベースにしています。

アートグラフィックス
篠塚一也

変更履歴

日付	Revision	変更点
2018.03.10	1.0	第一版。
2018.04.05	1.1	① 第二章の構成を変更、及び、節の追加。 ② 第五章を追加。
2018.05.31	1.2	① 第一章の内容に補足。
2018.07.25	1.3	① 1.7 を追加。 ② 他の節に於いて一部記述を変更していますが、読み直す必要はありません。
2018.07.28	1.4	① 随所で表現を正確にしました。 ② 「補足」の章を追加しました。
2018.08.07	1.5	① 第一章を再編成しました。 ② フィールド・マクロ・フラグの記述を追加しました。 ③ virtual sequencer 及び virtual sequence の記述を追加しました。
2018.10.20	1.6	① 第二章に「クラスの記述法」の節を追加しました。
2019.02.19	1.7	① 2.1.4 ファクトリの記述に説明を補足。
2019.03.10	1.8	① 1.1.4 節に説明補足。 ② 1.8 節を追加。 ③ 3.3.1 節に説明補足。 ④ 4.2 節に説明補足。
2019.04.06	1.9	① 誤字訂正
2019.06.16	2.0	① 誤字訂正

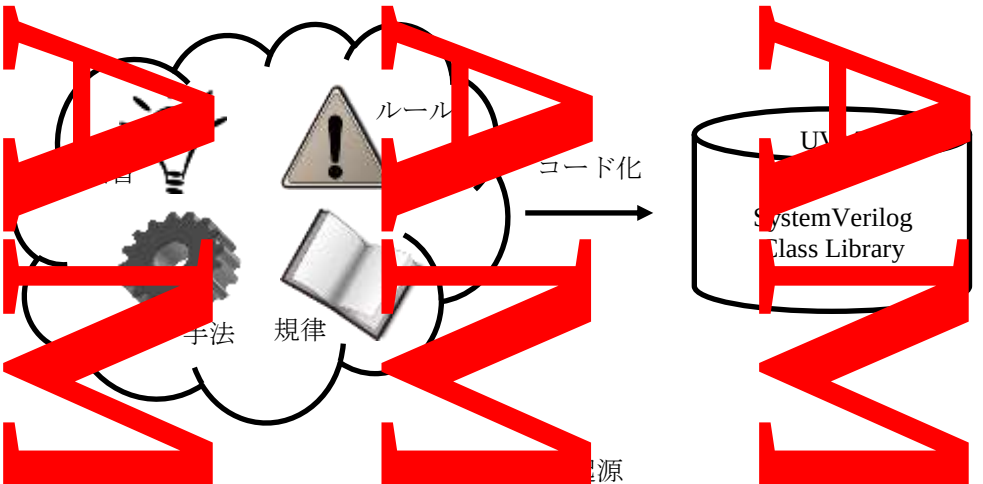
目次

1 概要

1.1 UVM 概要

1.1.1 UVM とは何か？

Univers Verific ethodology (UVM)とは、 野で推奨されて る技術 、慣習、規律等をコー として具体化し、検証技術 利用性を促進して生産性 をさせる為の System erilog クラス ライブラリー です。 M は ccellera System Initia により開発されました。



源

UVM は SystemVerilog をベースにして記述されているので、SystemVerilog をサポートしている検証ツールの環境で使用する事が出来ます。

1.1.2 on-Level Model

UVM は TLM を採用し シグナル・レベルよりも高位の記述法を用いて検証タスクを実現します。このアプロー テムの動作を考察する 方法です。

UVM ではトランザクションはオブジェクトであり、UVM コンポーネントがトランザクションを操作し 特別なコンポー イバー（uvm_ ス）が存在します。ドライバーはトランザクションをシグナル レベルに変換して DUT を操作する役目を持ちます。

DUT 側 シグナルの変化を検知する役 を持つ UVM コンポーネント 必要になります。そのコンポーネントは、一般的には、コレクター（collector）と呼ばれます。

ドライ ーの存在により ンザクション ムを記述する事が出来る様になります。尚 UVM コンポーネントと DUT 間のデータ受受には System erilog virtual nterface が使用されます。

1.1.3 代表的な UVM クラス

UVM には多くのクラスが定義されていますが、ユーザが直接使用するのはその内の一部のクラスで、メソドロジー・クラス（methodology class）と呼ばれます。代表的なメソドロジー・クラスとトランザクション関連のクラスとしては、以下のクラスが存在します。

UVM クラス	種別
uvm_sequence_item	トランザクション関連
uvm_sequence	トランザクション関連

uvm_driver	メソドロジー・クラス
uvm_sequencer	メソドロジー・クラス
uvm_env	メソドロジー・クラス
uvm_agent	メソドロジー・クラス
uvm_test	メソドロジー・クラス
uvm_monitor	メソドロジー・クラス
uvm_scoreboard	メソドロジー・クラス

uvm_sequence_item はトランザクションを記述する為のクラスで、データ・オブジェクトになります。uvm_sequence もデータ・オブジェクトですが、トランザクションの生成を行います。

上記の表の uvm_driver 以降はコンポーネントで、データ・オブジェクトではありません。コンポーネントとは、シミュレーションの実行エントリーで、コンポーネントが実行する事によりシミュレーションが進行して行きます。この為、UVM はデータ・オブジェクトとコンポーネントを明確に分けています。

UVM コンポーネントはシミュレーションの対象となるので、デザインにおける module の様な役目を果たします。即ち、UVM において、コンポーネントは自然にコンポーネント・インスタンスの階層構造を構成します。階層のトップには、uvm_top と呼ばれるインスタンスが存在します。UVM はその階層構造を使用してダイナミックなシミュレーションを制御します。具体的には、階層構造を使用してテストベンチの構成をシミュレーション開始直前に変える事が出来ます。この機能により、一度のコンパイルで複数のテスト・ケースを実行する事が出来ます。階層構造は、シミュレーション開始直前に決定し、一度シミュレーションが開始（コンポーネントのタスク run_phase が呼ばれた時点）するとコンポーネント・インスタンスの階層構造は変化しません。

実行エンティティーであるドライバー（uvm_driver のサブクラス）は DUT を制御します。DUT はシグナル・レベルの為、ドライバーはトランザクション・レベルからシグナル・レベルへの変換を行い、DUT をドライブします。ドライバーはトランザクションをシーケンサー（uvm_sequencer のサブクラス）から受け取ります。ユーザはこれらの機能を実現する必要があります。ユーザはメソドロジー・クラスから継承したクラスの内部に、これらの変換手順、及び、制御処理を定義します。

モニター（uvm_monitor のサブクラス）は DUT の信号変化を監視する機能を持ちます。モニター（通常はコレクター）は DUT の信号値をトランザクションに変換して、他のコンポーネントに引き渡します。それらのコンポーネントの一つとしてスコアボード（uvm_scoreboard のサブクラス）が存在します。スコアボードは DUT の出力が期待する結果と一致するか否かのテストを行います。その他、カバレッジ情報等も収集します。ユーザはメソドロジー・クラスから継承したクラスの内部に、これらの変換、及び、チェック手順を定義します。

シーケンサー、ドライバー、及び、モニターはエージェント（uvm_agent のサブクラス）として纏められます。エンバイロンメント（uvm_env のサブクラス）はシーケンサー、スコアボード、エージェント、及び、その他のエンバイロンメント（uvm_env のサブクラス）から構成されます。

1.1.4 Virtual Interface

DUT とのデータ授受は virtual interface を介して行われます。virtual interface は interface のインスタンスへのポインターです。interface のインスタンスはテストベンチ（トップ・モジュール）で作られます。そのインスタンスが DUT と関連するコンポーネントの virtual interface に引き渡されなければなりません。その割り当てをハード・コーディングすると検証技術の再利用性に反する為、実行時（通常は、build_phase、又は、connect_phase）に virtual interface の情報を設定します。

その際、前述したコンポーネント・インスタンスの階層構造が使用されます。

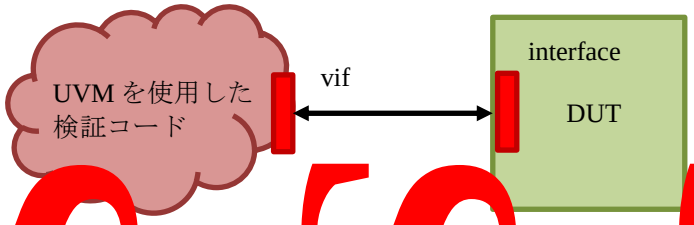


図 1-2　Virtual Interface を介した UVM と DUT の信号授受

virtual interface を使用する UVM コンポーネントは driver、及び、collector です。テストベンチには interface インスタンスが実在するので、テストベンチではそのインスタンスと UVM 側の virtual interface のコンフィグvif との対応表を準備しておきます。この準備を run_test() を呼ぶ前に完了しなければなりません。UVM 側は build_phase() 又は connect_phase 実行中に、その対応表を使用して vif に実インスタンスを割り当てます。従って、vif に関する設定がなされていないと実行時エラー終了します。一般に、この時点でのエラー解析は複雑なので、vif の設定には注意が必要です。詳細は、3.2 節を参照して下さい。

1.1.5　UVM テストベンチの構造

下図において、図の包含関係はクラス、又は、module のインスタンスを持つという関係を意味します。例えば、テストベンチは DUT のインスタンスを有します。

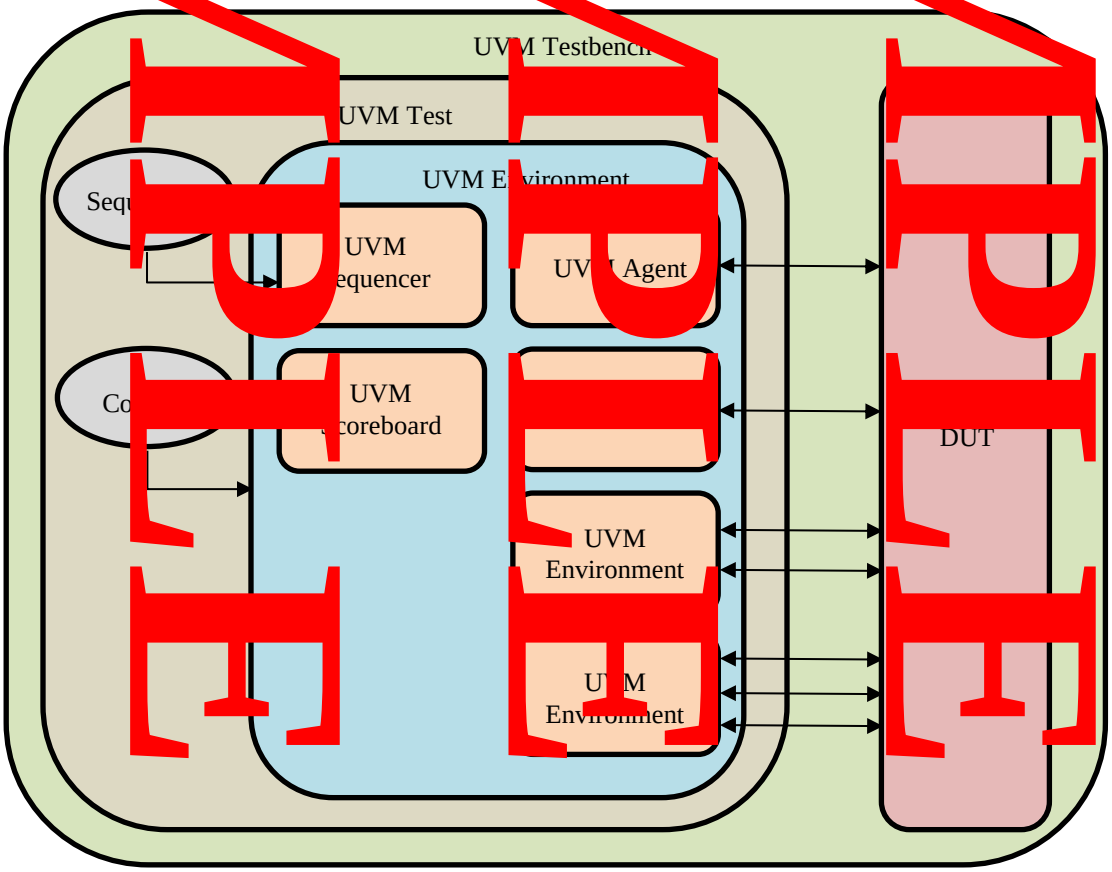


図 1-3　UVM テストベンチの構造（[2]）