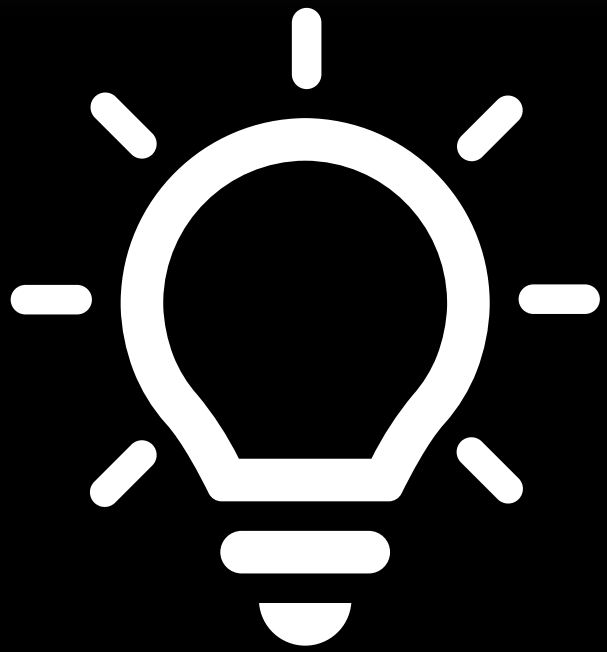
Artgraphics

なぜ SystemVerilog?

篠塚一也
アートグラフィックス
Document Revision: 1.0,2024.12.21

www.artgraphics.co.jp



注意事項
（Caveat）

SystemVerilogの知識を個人的に習得する目的として本資料を活用して下さい。本資料を通して、業務（実践）で必要となるSystemVerilogに関する知識を習得して頂くのが本来の目的です。

転用目的（本来の目的と違った他の用途に使う事）で本資料を使用する事はご遠慮下さい。また、本資料から学んだ知識を転載する場合等は出典が本資料である事を明記して下さい。但し、他の著者の文書にも書かれている内容は、この限りではありません。本注意事項は現在及び過去に於ける弊社からの全てのフリーダウンロード資料に適用されます。

本注意事項に合意出来ない場合には、本資料を速やかに抹消して下さい。尚、ダウンロード記録は、依然として残ります。

はじめに

- これからハードウェア記述言語を学び始める人にとって、Verilogではなく、なぜSystemVerilogなのかを解説します。
- 現在Verilogを使用している人は、SystemVerilogとの相違を把握し、SystemVerilogへの移行が必然的である事を理解できます。
- 本資料は、VerilogとSystemVerilogでは、シミュレーションの実行動作が異なる事を注意しています。同じVerilogコードをVerilog環境で実行した結果とSystemVerilogの環境で実行した結果は、一般に、一致しません。この事実は、最初に理解しておかなければならない相違点です。
- 本資料では、要点だけをかいつまんで解説しているので、熟読が必要です。

規格としてのSystemVerilogの変遷

- Verilogと云えば、正式には、IEEE Std 1364-2005 を意味します。仕様規格から分かるように、2005年に公開されています。しかし、それ以降は仕様が凍結され、SystemVerilogに統合されます。十年一昔と云われますが、Verilogは二昔前の言語仕様です。時代に即したSystemVerilogへの移行が必要です。
- 2005年にIEEE Std 1800-2005が公開され、SystemVerilogはVerilog HDLの拡張言語であると正式に宣言されています。
- Verilog HDL(IEEE Std 1364-2005)とSystemVerilog(IEEE Std 1800-2005)を統合した言語としてIEEE Std 1800-2009が公開されました。この仕様書は、Verilog HDLの仕様を包含したSystemVerilogの最初の仕様書と言えます。
- その後、IEEE Std 1800-2012 → IEEE Std 1800-2017 → IEEE Std 1800-2023等の改訂を経て、現在に至っています。
- 最新仕様はIEEE Std 1800-2023であり、本資料は最新仕様に準拠しています。

SystemVerilog規格のまとめ

- SystemVerilog規格は、以下のような変遷を辿って来ています。

Verilog HDLとの主な差異
変数の初期化順序

- Verilogでは、宣言時に指定した変数の初期化はシミュレーションの開始と同時に行われます。例えば、右記の①と②が同時に実行されるため、実行順序により結果が異なります。
- SystemVerilogでは、変数の宣言時に指定した初期化処理はシミュレーション開始前に行われるため、右記の例は正しく動作します。即ち、①の文が実行してから、②の文が実行します。

```
module test;
integer  width = 16;        ①
integer  nbits;

initial begin
         nbits = width;     ②
// ...
end
endmodule
```

Verilog HDLとの主な差異
シミュレーション実行モデル

- 例えば、右記の例では、変数aに1、2,3を割り当てていますが、Verilogでは変数aに値が設定されるタイミングを理解し難い問題があります。
- SystemVerilogはシミュレーションに関して厳密にタイミングを定義しているので、変数aに値が設定される時期を正確に把握できます。

```
logic [3:0]
          a;

initial  a = 1;
initial  #0 a = 2;
initial  a <= 3;
// ...
```

実行命令	実行領域	実行順序
a = 1;	Active	①
#0 a = 2;	Inactive	②
a <= 3;	NBA	③

Verilog HDLとの主な差異
イベント制御の問題

- エッジセンシティブなイベント待ちは、クリティカルなタイミング状況ではイベント解除を見逃し易い傾向があります。例えば、Verilogでは右のような記述をすると、どちらのinitialプロシージャが先に実行しても、@ev1または@ev2の何れかが解除されません。
- SystemVerilogでは、triggeredメソッドを使用する事により、クリティカルなタイミング状況においても、イベント解除を見逃す事はありません。つまり、@ev1および@ev2の代わりにレベルセンシティブに待つ事が可能です。

```
event     ev1, ev2;
initial begin
          ->ev2;
          @ev1;
end
initial begin
          ->ev1;
          @ev2;
end
```

```
event     ev1, ev2;
initial begin
          ->ev2;
          wait( ev1.triggered );
end
initial begin
          ->ev1;
          wait( ev2.triggered );
end
```

Verilog HDLとの主な差異
アレイ

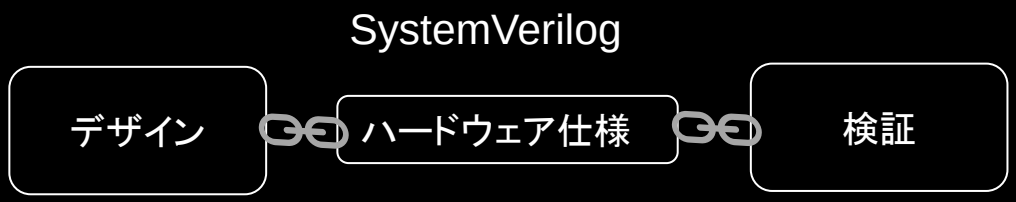
- Verilogでは一次元のアレイしか定義できませんが、SystemVerilogでは、多次元のアレイを定義する事ができます。
- しかも、SystemVerilog では、アレイには固定サイズのアレイだけでなく、ダイナミックアレイ、任意のデータ型のキーを持つアレイ、キュー等を定義する事ができます。

```
module test;
int                 a1[10],             // 固定サイズのアレイ
                    a2[],               // ダイナミックアレイ
                    a3[string],         // 文字列をキーに持つアレイ
                    a4[$],              // キュー
                    a5[5][8];           // 二次元のアレイ

// ...
endmodule
```

SystemVerilogの特徴

- SystemVerilogはハードウェア設計、仕様、および検証を統一的に記述できる言語であり、単なる検証言語ではありません。



SystemVerilogの利点

- SystemVerilogは、IEEE Std 1800-2023として規格化されているので、EDAツール間での相互互換性が保証されています。
- SystemVerilogは上位互換性の意味において、Verilogのスーパセットです。従って、従来のVerilogユーザは、そのままSystemVerilogのツール環境で作業を進める事ができます。
- RTL論理合成可能性を促進するための機能が充実しています。
- 検証に必要となる様々なデータ構造を支援するためにSystemVerilogは豊富なデータタイプを備えています。
- 記述能力が高いプログラミング機能が豊富です。
- SystemVerilogはシミュレーション実行論理の厳密な規約を定義しています。
- 検証に必要なアサーション、ファンクショナルカバレッジ、ランダムスティミュラス 生成等の機能をSystemVerilogは標準的に備えています。

設計言語としてのSystemVerilog

- Verilog HDLと比較して多くの機能がSystemVerilogに追加されました。特にRTL記述をより効率良く正確に行なうための機能が備えられています。例えば、以下のような機能が追加されています。
- モジュール間、およびテストベンチとDUT間の通信を簡素化するためのインターフェース機能
- enum型データタイプ
- ビット数に制限のない可変長リテラル（'0、'1、'x、'z）
- C/C++のようなストラクチャおよびユニオン等のデータタイプ
- 値を戻さないファンクション
- 便利な複合オペレータ（++、--、+=、&=、等）
- 論理合成可能性を促進する機能（always_comb、always_latch、always_ff、等）
- プログラミング機能の強化（foreach, continue、break、return文等）

検証言語としてのSystemVerilog

- Verilogには検証機能は皆無ですが、SystemVerilogには多くの検証機能、および検証に必要な機能が付け加わりました。具体的には、以下のような機能が追加されています。
- メモリー使用量と実行効率において優れた2-state型のデータタイプ（bit、byte、shortint、int、longint等）、文字列型データタイプ（string）
- 豊富なアレイ型（ダイナミックアレイ、associativeアレイ、キュー）、多次元アレイ
- クラス、インターフェース、パッケージ
- テストベンチ機能（program）
- プロセス間通信機能（mailbox、semaphore等）
- プロセス制御機能（fork/join_any、fork/join_none）
- クロック信号に同期する信号の管理（clocking block）
- ランダムスティミュラス生成機能
- ファンクショナルカバレッジ
- アサーション

デザインと検証のタイミング

- SystemVerilogの検証機能と厳密なスケジューリングルールにより、DUTからのレスポンスを正しいタイミングで検証できます。
- 組み合わせ回路はActive領域、シーケンシャル回路はActiveとNBA領域で実行すると仮定すれば、下表のようにDUT検証の正しいタイミングを決定できます。

領域	デザイン		検証		検証手段
	組み合わせ回路	シーケンシャル回路	組み合わせ回路	シーケンシャル回路	
Active	○	○			
Inactive			✓		#0
NBA		○			
Observed				✓	クロッキングブロック
Reactive				✓	program, およびchecker

検証のタイミング

- 組み合わせ回路

組み合わせ回路	検証法

`always @(a,b)`
`  z = a&b;`
	`initial forever @(a,b) #0 begin`
	`  if( z != a&b )`
	`  ...`
	`end`

組み合わせ回路の出力はActive領域で決定する。	Inactive領域では組み合わせ回路からの出力が安定しているので検証可能です。

- シーケンシャル回路

シーケンシャル回路	検証法
`always @(posedge clk)` `  q <= d;`	`clocking cb @(posedge clk); endclocking` `initial forever @cb verify();`
シーケンシャル回路の出力はNBA領域で決定する。	Observed領域ではシーケンシャル回路からの出力が安定しているので検証可能です。

まとめ

- 本資料では、SystemVerilogは時代が求めるハードウェア記述言語である事を事実に基づいて概説しました。また、Verilog HDLは、もはや言語としては存在しませんが、SystemVerilogのサブセットとして使用できる事を紹介しました。
- 更に、Verilog HDLが持つ曖昧性をSystemVerilogが解決している事も実例で解説しました。
- SystemVerilogでは、命令の実行領域が明確に定義されているので、DUTを検証するタイミングを理解し易くなっています。
- SystemVerilogは、近代的な検証手法に適した検証機能を備えているので、これからのハードウェア記述言語として、最も適しています。
- 本資料では、SystemVerilog機能の解説を省略しましたので、巻末の文献で知識の習得に努めて下さい。

参考文献

文献[1]は最新版の仕様書です。是非一読下さい。文献[2]は前仕様です。前仕様の知識がある方は文献[3]を読めば、文献[1]に相当する知識を得られます。文献[4]に文献[3]を無償でダウンロードする方法が記されています。SystemVerilogに関する知識の確認には、文献[5-7]を参照して下さい。文献[7]は設計分野で必要とされるSystemVerilogの基礎知識を非常に詳しく解説しています。シミュレーション領域を完全に理解したい方には文献[8]をすすめます。

[1] IEEE Std 1800-2023: IEEE Standard for SystemVerilog – Unified Hardware Design, Specification and Verification Language.

[2] IEEE Std 1800-2017: IEEE Standard for SystemVerilog – Unified Hardware Design, Specification and Verification Language.

[3] SystemVerilog IEEE Std 1800-2023の概要、アートグラフィックス 2024.

[4] IEEE Std 1800-2023の追加機能、アートグラフィックス 2024.

[5] 篠塚一也、SystemVerilogによる検証の基礎、森北出版 2020.

[6] 篠塚一也、SystemVerilog入門, 共立出版 2020.

[7] 篠塚一也、SystemVerilog超入門, 共立出版 2023.

[8] SystemVerilog シミュレーションの論理、アートグラフィックス 2024.