Artgraphics

# SystemVerilog基礎講座
# アサーションの基礎知識

篠塚一也
アートグラフィックス
Document Revision: 1.1,2025.11.07

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。

Copyright © 2025 Artgraphics. All rights reserved.

# アサーションとは？

- アサーションはシステムの動作（すなわち、仕様）を記述するための手段です。主として、アサーションはデザインの検証に使用されます。その他、入力となるスティムラスの検証にも使用されます。
- アサーションではシーケンス、および、プロパーティを用いて仕様を記述します。ただし、それらの記述自身だけでは起動しないため、assert、cover、assume等のアサーション文を使用して検証を行ないます。検証とは、仕様とデザイン（実装）が一致する事を確認する作業です。
- アサーションにおいて、検証者は仕様、および、合否処理（pass_statementとfail_statement）を記述します。その他の全ての検証タスクはシミュレータが担当します。例えば、デザインをシミュレーションし、（仕様 != デザイン）となる状況が発生するとfail_statementが実行します。シーケンス（およびプロパーティ）は複数のクロッキングイベントを経過して評価を終了します。評価終了時に真となれば、仕様とデザインが一致すると判断されます。途中で評価が偽となれば、仕様とデザインが一致しないと判定されます。

Copyright © 2025 Artgraphics. All rights reserved.

# アサーションの種類

- アサーションには、即時実行型と並列型の二種類があります。
- 即時実行型アサーションは、通常の実行文と同じように動作し、即時に実行が終了します。したがって、時間を消費する概念がありません。以下のような使用法があります。

```
initial begin
        sample = new;
        assert( sample.randomize() );
        ...
end
```

- 一方、並列型アサーションは、デザインのシミュレーションと並行して実行し、検証仕様が成立、または不成立するまで実行を続ける機能です。並列型アサーションでは、クロッキングイベント時に信号値をサンプリングして、Observed領域で検証仕様の評価を行います。
- 以降では、並列型アサーションのみを扱います。

Copyright © 2025 Artgraphics. All rights reserved.

# シーケンスとプロパーティ

- 並列型アサーションは、一回の評価では結論を出す事ができない複雑な検証仕様から構成されます。仕様記述のために、シーケンスとプロパーティを使用します。
- 簡単に言えば、シーケンスはブーリアン式をサイクルディレー（## オペレータ）で結合した正則表現です。

```
sequence unlock;
        @(posedge clk) a ##1 b ##1 c;
endsequence
```

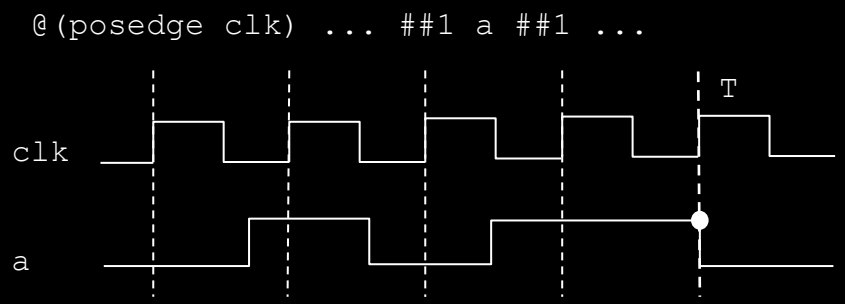
- プロパーティはシーケンスを結合して構築される検証仕様です。

```
property p;
        @(posedge clk) $rose(check) |->
                x ##[1:2] unlock.triggered() ##1 y;
endproperty
```

Copyright © 2025 Artgraphics. All rights reserved.

# 重要
# サンプリングと式の評価

- 評価式はシーケンスで記述されます。評価式に使用される変数の値は、原則として、Preponed領域でサンプリングされます。つまり、クロッキングイベントが起きた時（$time==T）の最初の領域で変数の値が決定されます。時刻Tにおいて変数の値が変化する可能性がありますが、その値は評価式で参照されません。




時刻Tにおけるaのサンプル値は、1'b1であり1'b0ではない。

- 例外は、automatic変数やローカル変数です。これらの変数の値は、式が評価される時に決定されます。
- 検証条件の評価をObserved領域で行います。

Copyright © 2025 Artgraphics. All rights reserved.

# アサーションとスケジューリング領域

- アサーションでは、以下に示す3つのシミュレーション領域が使用されます。

| 領域 | 実行内容 |
|---|---|
| Preponed | アサーションの式に使用されている変数のサンプリング |
| Observed | アサーションの式の評価 |
| Reactive | action_blockの実行 |

```
assert property (@(posedge clk) p1)
        $display("@%3t: PASS",$time);
        else $display("@%3t: FAIL",$time);
```

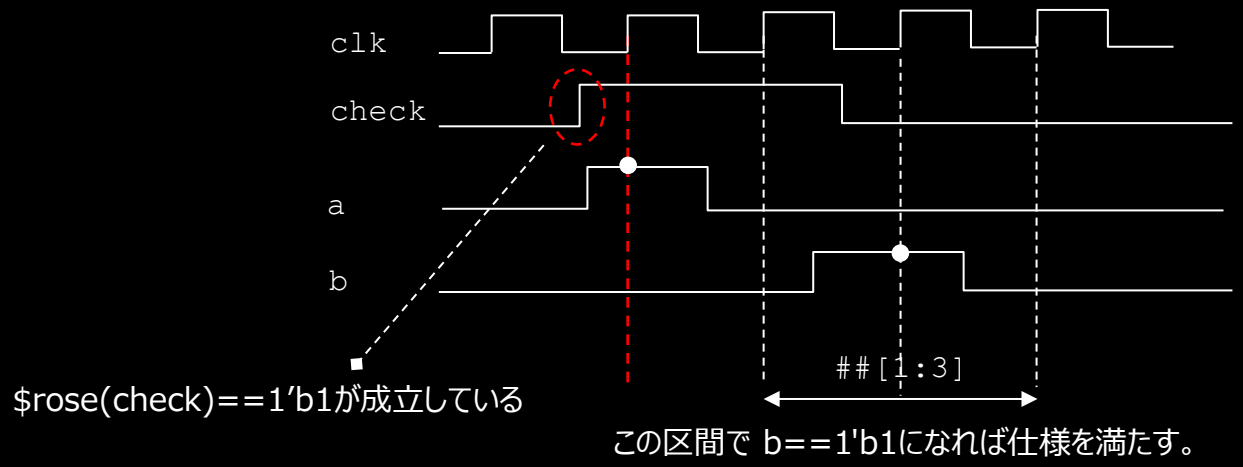
action_block

Copyright © 2025 Artgraphics. All rights reserved.

# アサーションの使用例

- この検証仕様は、クロッキングイベント発生時に、$rose(check)==1'b1 であれば、a==1'b1 で、しかも、1~3クロックサイクルの間に b==1'b1 が成立しなければならない条件を記述しています。

```
property check_a_b;
        $rose(check) |-> a ##[1:3] b;
endproperty
assert property (@(posedge clk) check_a_b)
        else $display("@%0t: FAIL",$time);
```




Copyright © 2025 Artgraphics. All rights reserved.

# implication (|->, |=>)

- implicationオペレータはプロパーティーオペレータで、前提条件（オペレータの左辺）が成立すれば、結果（オペレータの右辺）が成立しなければならない仕様を定義します。オペレータの左辺はantecedent、右辺はconsequentと呼ばれます。
- 以下のような二つの形式があります。

```
sequence_expr |-> property_expr
sequence_expr |=> property_expr
```

- **|->**は左辺が成立すれば、即座に右辺の仕様を評価しますが、**|=>**は1クロックサイクル後に右辺の評価を開始します。つまり、|=>は以下の仕様と同じです。

```
sequence_expr |-> 1'b1 ##1 property_expr
```

- オペレータ|=>を|->で表現できるので、一般的には|->を使用して仕様を表現します。
- 左辺が成立しない場合、右辺に関心がないため全体の仕様はpassしますが、このpassは真の意味ではなく**vacuous pass**と呼ばれます。

Copyright © 2025 Artgraphics. All rights reserved.

# 参考文献

文献[1]は最新版の仕様書です。是非一読下さい。SystemVerilogに関する知識の確認には、文献[2-4]を参照して下さい。文献[4]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。アサーションを本格的に学習するためには文献[2]をすすめます。


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

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

[3] 篠塚一也、SystemVerilog入門, 共立出版 2020.

[4] 篠塚一也、SystemVerilog超入門, 共立出版 2023.


Copyright © 2025 Artgraphics. All rights reserved.