Artgraphics

# SystemVerilog基礎講座
# プロセス実行順序

篠塚一也
アートグラフィックス
Document Revision: 1.1,2025.10.22

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.

# 例題－1　プロセス実行順序に関する問題

- 以下の記述において、$time==0で@(negedge a)のイベントは起きますか。

```
module test;
logic    a;

initial begin
         a = 0;
end

always @(negedge a)
         $display("@%0t: @(negedge a)",$time);

endmodule
```

Copyright © 2025 Artgraphics. All rights reserved.

# 例題の解答と解説

- @(negedge a)のイベントが起こるとは限りません。
- initialプロシージャがalwaysプロシージャよりも先に実行すれば、イベントは起きません。一方、alwaysプロシージャがinitialプロシージャよりも先に実行すればイベントは起きます。

Copyright © 2025 Artgraphics. All rights reserved.

# 例題－2　プロセス実行順序に関する問題

- 以下の記述において、$time==0で@(negedge a)のイベントは起きますか。

```
module test;
logic    a;

initial begin
         a <= 0;
end

always @(negedge a)
         $display("@%0t: @(negedge a)",$time);

endmodule
```

Copyright © 2025 Artgraphics. All rights reserved.

# 例題の解答と解説

- イベントは起きます。
- @(negedge a)のイベント待ちが有効になってから、aに0がセットされるので、イベントが起きます。

Copyright © 2025 Artgraphics. All rights reserved.

# 例題－3　プロセス実行順序に関する問題

- SystemVerilogでは、initialとalwaysプロシージャの実行開始順序は規定されていないので、どのプロシージャが最初に実行を開始するかわかりません。以下のように3つのinitialプロシージャが定義されているとします。

```
initial begin
          $display("@%0t: p1 started",$time);
end
initial begin
          $display("@%0t: p2 started",$time);
end
initial begin
          $display("@%0t: p3 started",$time);
end
```

```
@0: p3 started
@0: p2 started
@0: p1 started
```

- 3つのinitialプロシージャがどのような順序で実行を開始しても右上のようにメッセージがプリントされるようにしたいと思います。以下の問に答えてください。

① eventを使用して、上記のようにプリントするようにしてください。

② eventを使用せずに、上記のようにプリントするようにしてください。

Copyright © 2025 Artgraphics. All rights reserved.

# 例題の解答と解説ーeventを使用

- eventを使用すると以下のようになります。レベルセンシティブにイベント待ちをしなければなりません。

```
event     p3_started, p2_started;

initial begin
          wait( p2_started.triggered );
          $display("@%0t: p1 started",$time);
end

initial begin
          wait( p3_started.triggered);
          $display("@%0t: p2 started",$time);
          ->p2_started;
end

initial begin
          $display("@%0t: p3 started",$time);
          ->p3_started;
end
```

Copyright © 2025 Artgraphics. All rights reserved.

# 例題の解答と解説ー変数を使用

- 変数を使用すると以下のようになります。変数の場合でも、レベルセンシティブにイベント待ちをしなければなりません。ここで紹介する方法は汎用的な解法です。

```
typedef enum { P1=0, P2, P3 } run_id_e;
logic [2:0]         proc_ready;

initial begin
          wait( proc_ready[P2] ) ;
          $display("@%0t: p1 started",$time);
          proc_ready[P1] = 1'b1;
end

initial begin
          wait( proc_ready[P3] ) ;
          $display("@%0t: p2 started",$time);
          proc_ready[P2] = 1'b1;
end

initial begin
          $display("@%0t: p3 started",$time);
          proc_ready[P3] = 1'b1;
end
```

Copyright © 2025 Artgraphics. All rights reserved.

# 例題ー4　イベント発生時に関する問題

- 以下の記述においてどのような値がプリントされますか？

```
assign p = q;

initial begin
          q = 1;
          #1 q = 0;
          $display(p);
end
```

Copyright © 2025 Artgraphics. All rights reserved.

# 例題の解答と解説

- この場合、0または1がプリントされます。どちらの値でも正解になります。

```
assign p = q;

initial begin
          q = 1;
          #1 q = 0;
          $display(p);
end
```

qに0をセットした後、そのまま続行すれば1がプリントされ、
続行する前にpの値を更新すれば0がプリントされます。

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。本資料で紹介した技術を詳しく学ぶためには、文献[5]をすすめます。SystemVerilogの検証機能全体を体系的に習得するためには、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.

[5]篠塚一也、検証のためのSystemVerilogプログラミング、森北出版 2022.


Copyright © 2025 Artgraphics. All rights reserved.