Artgraphics

SystemVerilog基礎講座

ブロッキングとノンブロッキング代入文の相違

篠塚一也
アートグラフィックス
Document Revision: 1.1,2025.07.03

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
ブロッキングとノンブロッキングに関する問題

- 下記の記述において、変数aに対してブロッキングとノンブロッキングで値が設定されています。どのような値がプリントされると思いますか？

```
module test;
logic [1:0]       a;

always @(a) begin
        $display("@%0t: a = %0d",$time,a);
end

initial begin
        #10;
        a <= 1;
        a = 2;
end

endmodule
```

Copyright © 2025 Artgraphics. All rights reserved.

例題の解答と解説

- 以下のようにプリントされます。

```
@10: a = 2
@10: a = 1
```

- a <= 1の方が先に書かれていますが、直ぐには実行されません。この命令はスケジューリングされて後続の命令に実行が移ります。なお、この命令はNBA領域で実行されます。一方、a = 2の方が後に書かれていますが直ぐ実行されます。したがって、aには2が先に割り当てられ、次に1が割り当てられます。

実行順序	実行命令	実行領域
1	`tmp = 1`	Active
2	`a = 2`	Active
3	`a = tmp`	NBA

Copyright © 2025 Artgraphics. All rights reserved.

例題－2
ブロッキングとノンブロッキングに関する問題

- 次の記述に問題があれば指摘して下さい。また、問題点を解消する方法があれば示して下さい。
- ただし、記述に使用されている変数の初期化は、どこかで正しく行われていると仮定します。

```
logic             clk;
logic [3:0]       a, b;

always @(posedge clk)
         a = a + 1;

always @(posedge clk)
         b = a + b;
```

Copyright © 2025 Artgraphics. All rights reserved.

例題の解答と解説

- 二つのalwaysプロシージャの実行順序により結果が異なる問題が存在します。二つのプロシージャをそれぞれp1とp2と呼び、p1とp2の実行順序により結果が異なる事を示します。以下では、現在、aの値は5、bの値は7であると仮定します。

p1がp2よりも先に実行する場合	p2がp1よりも先に実行する場合
この場合にはaが先に計算されるので**a==6**となり、その後bが計算されるので**b==13**となります。	この場合には、bが先に計算されるので**b==12**となり、その後にaが計算されるので**a==6**となります。

- bはaに依存するので、p1とp2の実行順序により結果が異なることが分かります。この問題を回避するためには、以下のようにノンブロッキング代入文を使用しなければなりません。コメントにあるように同じa(t)が右辺に使用されています。ここで、a(t)は現在の値を、a(t+1)は新しい値を示します。b(t)に関しても同様です。

```
always @(posedge clk)
        a <= a + 1;       // a(t+1) = a(t) + 1

always @(posedge clk)
        b <= a + b;       // b(t+1) = a(t) + b(t)
```

Copyright © 2025 Artgraphics. All rights reserved.

例題ー3　代入内タイミング制御に関する問題

- 代入文の内部にタイミング制御がある場合、ブロッキング代入文とノンブロッキング代入文では式の評価法が異なります。
- 以下にはブロッキング代入文とノンブロッキング代入文にディレーが指定されています。アレイaの要素を参照するために使用されているindexの値は幾つですか？

ブロッキング代入文	ノンブロッキング代入文

initial begin
index = 1;
a[index] = #50 20;
end
initial begin
#25 index = 0;
end
	initial begin
	index = 1;
	a[index] <= #50 20;
	end
	initial begin
	#25 index = 0;
	end

Copyright © 2025 Artgraphics. All rights reserved.

例題の解答と解説

- ブロッキング代入文では、イベントが起きた後に左辺の参照が評価されます。しかし、ノンブロッキング代入文では、文が処理される時に左辺の参照が評価されます。

ブロッキング代入文	ノンブロッキング代入文

initial begin
index = 1;
a[**index**] = #50 20;
end
initial begin
#25 index = 0;
end
	initial begin
	index = 1;
	a[**index**] <= #50 20;
	end
	initial begin
	#25 index = 0;
	end

a[index]のindexは**0**です。	a[index]のindexは**1**です。

- 理解が不十分な場合は、文献[4]の5.2.2項および6.7節を参照して下さい。

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。文献[5]は、ブロッキングとノンブロッキング代入文の相違を非常に詳しく解説した技術資料です。

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

Copyright © 2025 Artgraphics. All rights reserved.