初歩からのHDLテストベンチ

第12回　非同期検証と応用的検証

安岡貴志，平郡政幸

デバイスの記事　ビギナーズ

HDLで回路を記述できるようになったばかりで，これからテストベンチを書こうとしてる方を対象とした連載の最終回である．本稿ではVerilog HDLとVHDL両方のテストベンチの書き方を解説する．今回は非同期回路の問題点と，そのRTL（Register Transfer Lenel）シミュレーションにおける解決方法，タスク/プロシージャの応用的な使い方，シミュレーション以外の検証方法について解説する．（筆者）

1．非同期検証

● HDL設計フロー

Verilog HDL，VHDL共通

図1はHDL設計の作業工程を示しています．

図1の①は，RTLの検証対象回路を検証する工程です．この工程ではテストベンチが作られ，RTLシミュレーションが行われます．RTLシミュレーションではゲート遅延は考慮されません．このため，検証対象の組み合わせ回路は，入力信号が変化すると遅延時間0で出力信号が変化します．このシミュレーションでは各サイクルで行われる処理が，仕様と合っているかどうかを確認します．組み合わせ回路の遅延が指定のクロック周期に収まるかどうかはチェックしません．

図1の②は，論理合成を行う工程です．RTLシミュレーションが完了した後，つまり検証対象のRTLコードがゲート遅延を除いて仕様通り動くという確認が取れた後，この工程を実施します．この工程では，RTLコードを論理合成ツールに読み込ませて，ゲート回路の接続図，いわゆるネットリストを出力させます．

図1の③は，ネットリストに対してシミュレーションを行う工程です．ネットリストにはRTLの情報に加えて，ゲート遅延の情報が付加されています．RTLのときに一つのバス信号として表現されていた複数ビットの信号が，1本1本別々の遅延情報を持ち，かつRTLのときにはなかったゲート間の中間信号が生成されます．このため，ゲート・レベル・シミュレーションはデータ量が多く，シミュレーションの所要時間は，RTLシミュレーションの数十倍以上になります．

たいていの場合，ゲート・レベル・シミュレーション工程では，RTLシミュレーションで実施したすべてのテストを実施するだけの時間を取ることができません．そこで，できるだけ機能を組み合わせて，最少のパターン数で機能を網羅できるリグレッション・テスト（本誌2008年12月号，pp.93-103の連載第9回を参照）と，ゲート・レベル・

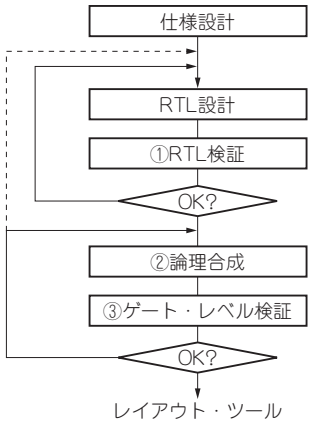


図1
HDL設計の作業工程

Keyword　非同期回路，RTLシミュレーション，ゲート・レベル・シミュレーション，ゲート遅延，クロック・スキュー，配線遅延，ジッタ

シミュレーションでしか検証できない非同期部分のテストだけを行います．

● 非同期対策

Verilog HDL，VHDL共通

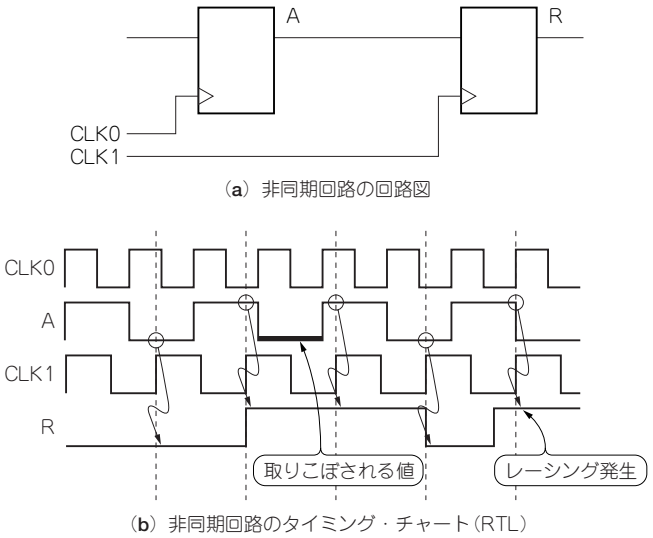
ゲート・レベル・シミュレーションでは，RTLシミュレーションと違い，ゲート遅延が発生します．同期回路，つまり一つのクロックで動いている分には，遅延による問題は論理合成時に調整できます．しかし，非同期回路，つまり**図2**(**a**)のように二つのクロックの間で，データの載せ替えが発生する回路では，ゲート遅延によってRTLシミュレーションとゲート・レベル・シミュレーションで，載せ替えられるデータが変わることがあります．

図2(**b**)は，**図2**(**a**)の回路のRTLシミュレーションのタイミング・チャートです．ここで図の中央付近の信号Aの値が，信号Rには載せ替えられず，取りこぼされています．しかし，ゲート・レベル・シミュレーションでは，信号Aに遅延が付くので，信号Aとクロック信号CLK1の位相が変わり，取りこぼされる値の位置が変わってきます．このようなとき，載せ替えるデータが変わっても，回路の動作に問題ないことを，何らかの方法で確認しなければいけません．

この方法として，ゲート・レベル・シミュレーションで確認すると，RTLシミュレーションより精度の高いテストができます．しかし，所要時間の長いゲート・レベル・シミュレーションでは，万が一不具合が見つかったときに，やり直しに時間がかかります．そこで，RTLシミュレーション工程で，遅延パラメータを利用した不具合の確認をしておくと，ゲート・レベル・シミュレーションで発見される不具合の数が減り，検証の時間を減らすことができます．

図3(**a**)と**図3**(**b**)はそれぞれVerilog HDLとVHDLによるフリップフロップの記述です．ここではパラメータDLY0を使って，クロック信号CLK0の立ち上がりから，信号Aへの代入の時間に遅延を付けています．**図3**(**c**)と**図3**(**d**)はそれぞれDLY0が1nsと9nsのときのタイミング・チャートです．ここでは，1nsが設計ルール上の最小遅延，9nsが最大遅延という前提でパラメータを操作しています．

このように，最小遅延と最大遅延の両方のシミュレーションをRTLのときにしておけば，非同期の問題もだいたい発見できるので，ゲート・レベル・シミュレーションのやり直しの回数を減らすことができます．ただし，パラメータはモデルとは別ファイルに記述しておく必要があります．なぜなら，パラメータの値を直接モデルと同じファイルに書いた場合，この最小遅延・最大遅延のシミュレーションを切り替える際，RTLのファイル自体を書き直さなければいけないからです．手で修正を行うと，それだけ作業量も増え，同時にミスを持ち込む危険があります(本誌2008年7月号，pp.113-121の連載第7回を参照)



(**a**) 非同期回路の回路図

(**b**) 非同期回路のタイミング・チャート(RTL)

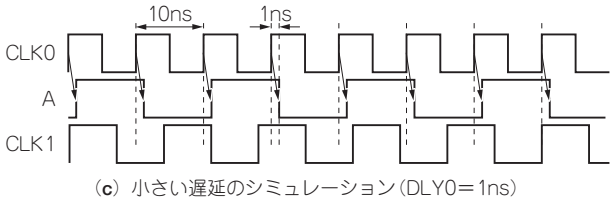
図2　クロックの乗せ換え
レーシングについては，連載第2回を参照．

```
always @(posedge CLK0)
  A <= #DLY0 IN;
```

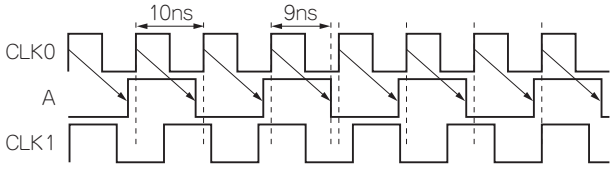
(**a**) FFの記述(Verilog HDL)

```
process(CLK0)begin
  if(CLK0'event and CLK0='1')then
    A <= IN after DLY0;
  end if;
end process;
```

(**b**) FFの記述(VHDL)



(**c**) 小さい遅延のシミュレーション(DLY0＝1ns)



(**d**) 大きい遅延のシミュレーション(DLY0＝9ns)

図3　非同期対策
(c)と(d)の両方で回路の仕様上動作に問題がなければよい．