実践的
VHDL設計
セミナ2

CD-ROM
設計データ
EDAツール

竹本　悟

ビギナーズ・スクエア

続 RTLって何？
――記述できること，できないこと

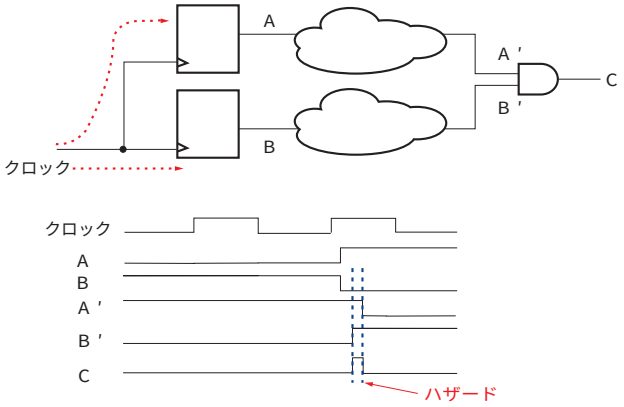
前回に引き続いて，RTL記述について解説する．RTL記述で重要なことは，論理合成ツールにうまく仕事をさせるように書くこと，そしてバグが少なく自分がデバッグしやすいように書くことである．今回は，ハザード対策，シミュレーションの入出力記述，デバッグを考慮した記述スタイルなどについて説明する．　（編集部）

1　論理合成はハザードに無力

ディジタル回路を設計し，デバッグするときに実際よく悩まされるのが「ハザード（髭のように短いパルス）の発生」である．回路中に，ラッチのような非同期的な要素があったり，あるいはこのハザードがクロック系に入ってしまうと，回路が誤動作を起こす．

◆TTL時代のハザード問題

TTLでカウンタを設計していた時代，LS163という同期式のカウンタをよく使用した．ただし，このキャリ出力にはハザードが混じっており，それを安易にクロック等にもっていくと回路を誤動作させてしまうという欠点があった．というのは，このカウンタのキャリ出力は，フリップフロップ（以下FFと記述する）の出力をたんに組み合わせ回路でゲーティングしていたのである．このとき，必ずといっていいほどハザードが発生する（**図1**）．



〔図1〕ハザードの発生の仕組み

◎同じクロックで駆動されるフリップフロップから出た信号でも，組み合わせ回路を通過する時間が違うため，ゲートの入力タイミングが前後する．その結果，不要なパルスが発生する．これがハザードである．

◎ASICでは，ゲートの遅延のほかに配線の抵抗/容量による遅延が加わり，さらにクロックのスキューが入る可能性があるので，ハザードの発生する確率が非常に高い．「組み合わせ回路はハザードを発生するものだ」という認識で設計しなければならない．論理合成はスタティックなタイミング解析しか行わないので，ハザードを予測できないと考える．

慣れた回路設計者は，それを承知しており，間にフリップフロップを入れるなどして防衛策をとったものである．

◆VHDLで記述してもハザードは発生する

HDLでカウンタを記述するときは，このようなハザード対策は考える必要がない，といいたいが，そうではない．ハザードが発生する記述がある．

リスト1にカウンタのVHDLによるRTL記述例を示す．算術演算ライブラリを使用していないので，やや長い記述になっているが，キャリの部分に注目してほしい．ここはコンカレント文で記述されている．この記述を論理合成すると，**図2**(**a**)に示すように論理合成ツールは4入力AND+2入力ANDで構成しようとするため，**図1**と同様，伝達時間の差でハザードが出る．

ハザードの発生原因は，このように回路内にあるゲートの遅延差や配線遅延差である．じつは，論理合成後にゲート・レベルのシミュレーションを行い，注意深く見れば，ハザード（動作不安定箇所）の発見は可能である．しかし，実際にはなかなか難しい（**図2**(**b**)）．

論理合成では，FF間の伝達時間の計算はするが，それがハザードを発生するかどうかの判断はできない，と考えたほうがいい．

◆RTLの場合はハザードの発見は不可能

RTL記述の段階でシミュレーションしてハザードを見つけられるだろうか．残念ながら，RTLはゲート遅延を表現しない．つまり，RTLでシミュレーションしてもハザードは発見できない．そして，論理合成時にも何の警告も出ない．

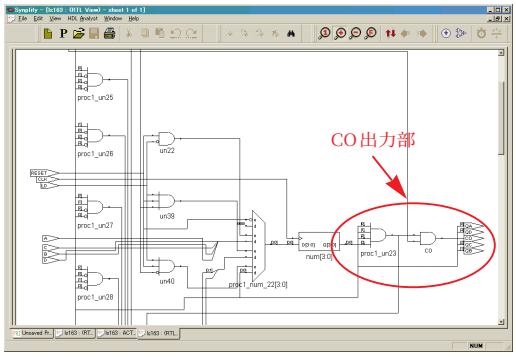
とくに問題になるのは，組み合わせ回路を通過した信号をフリップフロップのセット，リセット，クロック端子に入れる場合である．RTL記述段階で何も問題なく，論理合成も正常にパスして，ゲート・レベルのシミュレーションでもOKだったのに，実際にASICを作ってみたら動かない…調べていくと，信号にヒゲが乗っていた，というケースである．厄介なことに，こうした場合はテスト・パターンを入れても発見できない場合がある．

◆論理合成は何もしてくれない

組み合わせ回路を通過した信号は，一般にハザードが乗る可能性が高い，と考えなくてはならない．これは，ゲート間の微妙な遅延時間の違いのため，信号の代わり端にヒゲが乗るからである．このような信号をフリップフロップのクロックやセット/リセット端子に入力

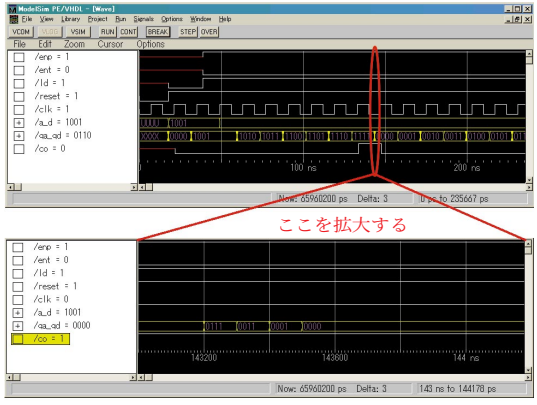
〔リスト1〕16進カウンタの例

```
library ieee;
use ieee.std_logic_1164.all;

 --4 bit binary syncronous counter with sycronous RESET
entity ls163 is
Port (ENP, ENT, LD, RESET, CLK, A, B, C, D: in std_logic;
      QA, QB, QC, QD:out std_logic;
      CO:out std_logic
      );
end ls163;

architecture bit4 _counter_s of ls163 is
  signal Num:std_logic_vector (3 downto 0);
  begin
  proc1: process (CLK, RESET)
    begin
    if CLK'event and CLK='1' then
       if RESET='0' then
             Num <= "0000";
         else
           if LD='0' then
              Num <= D & C & B & A;
           else
             if ENP='1' or ENT='1' then --counter behavior
               case Num is
                when "0000" => Num<="0001";
                when "0001" => Num<="0010";
                when "0010" => Num<="0011";
                when "0011" => Num<="0100";
                when "0100" => Num<="0101";
                when "0101" => Num<="0110";
                when "0110" => Num<="0111";
                when "0111" => Num<="1000";
                when "1000" => Num<="1001";
                when "1001" => Num<="1010";
                when "1010" => Num<="1011";
                when "1011" => Num<="1100";
                when "1100" => Num<="1101";
                when "1101" => Num<="1110";
                when "1110" => Num<="1111";
                when "1111" => Num<="0000";
                when others => Null;     --do nothing
             end case;
           end if;              --end counter behavior
         end if;                --end LD
       end if;                  --end reset
     end if;                    --end clock
  end process;

  CO <= '1' when Num = "1111" and ENP='1' else --carry out is asyncronous
        '0';                                   --この部分でハザードが発生！！

QA <= Num (0);
QB <= Num (1);
QC <= Num (2);
QD <= Num (3);
end bit4 _counter_s;
```



(a)リスト1を論理合成ツールで合成した結果

合成結果を回路図で出力したもの．CO出力部はANDゲートで構成されているのがわかる．フリップフロップが入っていない組み合わせ回路となっている．これではハザードが出る！（Syplify 5.0評価版を使用）



(b)タイミング・シミュレーションを行った結果

合成後に，Altera FLEX8000をターゲットに配置配線を行い，タイミング・シミュレーションをかけると(b)のようになる．QA-QD信号（カウンタ出力信号）の出力結果を見ると，一見正常に動作しているように思えるが，よく見ると不安定箇所が散見される．たとえば，１１１１→０００００に変化するところを拡大すると，１１１１→０１１１→００１１→０００１→０００００と遷移している．この場合は，システムとしての誤動作になっていないが，不安定であることがわかる．(ModelSim PE/VHDL評価版を使用)

〔図2〕 リスト1のカウンタを論理合成にかける

してしまうと，誤動作の原因になる．とくに，カウンタのキャリやセレクタの出力などは，ほとんどの場合ハザードが乗ってくる．初心者は，そこまで予測できないから，こうしたトラブルを招きやすい．また，回路図では，ハザードの影響を防ぐ配慮ができる技術者でも，不慣れなHDL設計では忘れてしまうことも多い．

論理合成ではRTLに記述されている論理にしたがって回路を構成する．このときハザード発生の原因となるゲート遅延などは問題としない(記述できない．記述したとしても論理合成ツールは無視する)．つまり論理合成を行う段階ではなんの対策も取れない．

設計者としてできることは，ハザードが発生しないような，あるいは発生しても問題を引き起こさないようなHDL記述を行うことである．組み合わせ回路を通過してくる信号は，すべてハザードが入るものと見なして設計する必要がある．

たとえば，コンカレント文中の，等式の左辺に入る信号は組み合