連載 ASIC化のためのHDL設計手法講座

(6)RTLとゲート・レベル検証の不一致問題対策

● 長谷川 裕恭 ●

前号では，“RTL”と“ゲート・レベル”の2段階の検証が必要だと解説しました．これは，ゲート・レベルでないと遅延が考慮されないという問題（**コラム**参照）よりも，シミュレーション結果の不一致という問題で必要となってくるのです．今回は，この不一致の問題がなぜ起きるかについてと，その対応策について解説します．

Xの不一致

RTLとゲート・レベル検証での，結果不一致の原因の一つは「Xの不一致」にあります．Xはいうまでもなく，値の「不定」を示します．このXの不一致は，RTL記述と論理ゲートでの，Xの伝播のふるまいが異なることにより生じます．

■RTL記述でXを入力すると…

例を示しましょう．**リスト1**のRTL記述で，Aに'1'を入力すれば，条件項と一致するので，Yからは'0'が出力されます．Aに'0'を入力すれば，条件項と異なるのでelse項が実行され，Yからは'1'が出力されます．ここまでは，問題ありません．

ここでAに'X'を入力すると，どうなるでしょうか．'X'は不定であって'1'ではないので，else項が実行され，Yは'0'を出力してしまいます．Aには，'X'という間違いの値が伝播してきたのですが，このif文を通過すると'1'という値に変化してしまい，場合によっては結果が正しく見えてしまうことがあるということになります．

■RTLで'X'の伝播をするように記述してみる…

'X'の伝播をゲート・レベルと同一とするには，**リスト2**のように記述を変更するという方法もあります．この記述では，Aに'X'が入力されると'1'でも'0'でもないのでelse項が実行され，Yは'X'を出力することになります．

しかし，この方法はそう簡単にはいきません．たとえば，**リスト3**のように条件項がA =“010”のときだけ，Yに'1'を出力し，そうでないとき'0'を出力する記述をしたとします．これを'X'を考慮して完全に記述するとなると，**リスト4**のように記述量がかなり増えてしまいます．

もともと，RTL記述とその記述から生

リスト1　RTL記述で'X'が正しく伝搬しない記述例1

[Verilog-HDL]
```
if(A==1'b1)
  Y <= 1'b0;
else
  Y <= 1'b1;
```

[VHDL]
```
if(A='1') then
  Y <= '0';
else
  Y <= '1';
end if;
```

リスト2　'X'が正しく伝播するようにリスト1を書き直す（これはうまくいくが）

[Verilog-HDL]
```
if(A==1'b1)
  Y <= 1'b0;
else if(A==1'b0)
  Y <= 1'b1;
 else
  Y <= 1'bx;
```

[VHDL]
```
if(A='1') then
  Y <= '0';
elsif(A='0') then
  Y <= '1';
else
  Y <= 'X';
end if;
```

リスト3　RTLで'X'が正しく伝播しない記述例2

[Verilog-HDL]
```
if(A==1'b010)
  Y <= 1'b1;
else
  Y <= 1'b0;
```

[VHDL]
```
if(A="010") then
  Y <= '1';
else
  Y <= '0';
end if;
```

リスト4　'X'が正しく伝播するようにリスト3を書き直す（これは面倒！！）

[Verilog-HDL]
```
if(A==1'b010)
  Y <= 1'b1;
else if(A==1'b000 ||
     A==1'b001 ||
     A==1'b011 ||
     A==1'b100 ||
     A==1'b101 ||
     A==1'b110 ||
     A==1'b111)
  Y <= 1'b0;
```

[VHDL]
```
if(A="010") then
  Y <= '1';
elsif(A="000" or
    A="001" or
    A="011" or
    A="100" or
    A="101" or
    A="110" or
    A="111") then
  Y <= '0';
end if;
```

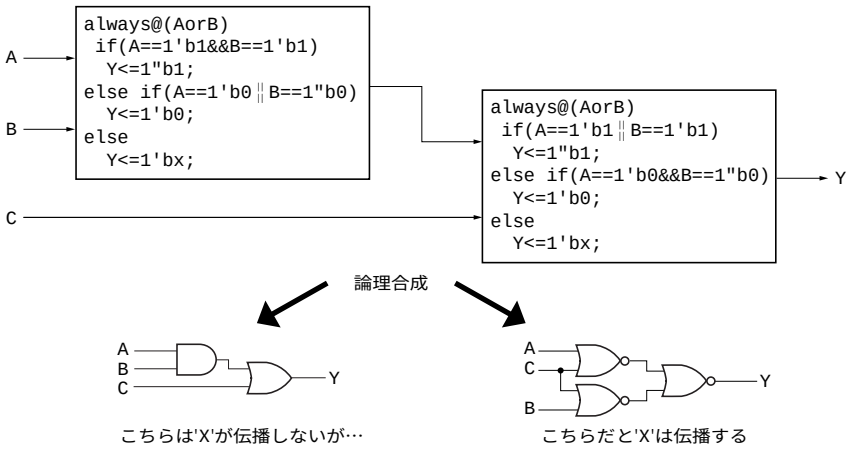

図1　論理合成で生成される回路構造は一定ではない

成した論理ゲートは，回路構造が必ずしも一致しません．したがって，いくらRTL記述で'X'の伝播を記述したとしても完全に一致させることはできないのです．

■回路構造の変化 ── always文の場合

次の例です．**図1**のように，RTL記述で二つのalways文(process文)を記述したとします．右側のalways文では，AとBの二つの入力があります．このとき，Aにだけ入力信号が'X'になる可能性があるとします．設計者は，Aが'X'になる可能性があるとき，Bにつねに'0'が入力されるように設計したとします．このようにBでA信号をマスクすれば，RTLシミュレーションでは，'X'は発生しなくなります．

この記述に対し，論理合成を実行し，論理ゲートを生成させると**図1**右下の回路が生成されたとします．左の回路はRTL記述の回路構造と等価なので，設計者の意図どおり，'X'はこれ以降伝播しなくなります．

しかし，論理合成ツールは，回路の品質を高めるために，最適化などが行われ，回路の構造を変えてきてしまうことが多々あります．この記述も，場合によっては右側の回路が生成されることもありえるのです．右側の回路と左側の回路では，論理的には等価なのですが，右側の回路では，Aに入力された'X'は，Bでマスクされることなく後段に伝わってしまうのです．

このように，'X'の不一致の問題を解決するために，いくらRTL記述で'X'の伝播を同じにしようとしても，完全に同一にすることはできないのです．

'X'伝播の対応方法とその問題点

■すべてのFFに初期リセットを行い 'X'を発生させなければ解決か？

じつは，'X'の不一致の問題を解決するためにもっとも効果的な方法は，'X'を発生させないことなのです．つまり，すべてのFFに対して初期リセットをかけてしまえば，'X'状態のFFがなくなり，'X'が発生しなくなります．

'X'が発生しなければ，当然，'X'の不一致の問題は起きなくなるはずです．しかし，ことはそう簡単に運びません．

■設計の周辺部分から 'X'が発生する…

最近のASICは，搭載可能なゲートが飛躍的に向上しました．昔なら，ASICは設計者が新たに設計した論理ゲートのみをASICに搭載していたのが，最近では，周辺のデバイス，RAMなどを搭載するのが当たり前になりました．RAMを使用すると，たとえ設計者が新たに設計した部分のFFをすべてリセットしたとしても，RAMから'X'が発生してしまいます．また，RAM以外にも双方向バス

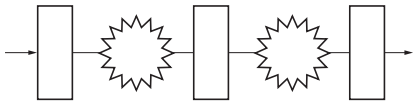

図2　初期リセットをかけないでも，3クロックでFFの値が確定する

を使用していれば，そこから発生することもあります．

さらに，ASICベンダが供給する論理ライブラリが，FFのホールド・エラー時や，信号が遷移する間の短い期間に'X'を発生させるものもあるのです．

■基本的な対策はある

Xの不一致の問題を完全に解決することはなかなかできません．しかしながら，RTLでのシミュレーションで'X'の発生を抑えておけば，それだけゲート・レベルでのシミュレーションで問題が起きにくくなることは間違いありません．

できるだけすべてのFFに対して初期リセットをかけることをお勧めします．もちろん，すべてのFFに初期リセットをかけると，それだけゲート数が増えてしまいます．**図2**のようにデータが一方向に流れて行く回路であれば，いずれは値が確定します．この部分のFFには，初期リセットを省略してもかまいませんが，少なくてもリセット信号が立ち上がり，リセット解除となった段階では'X'状態のFFをなくし，シミュレーションを実行してください．

初期リセットは非同期リセット

■非同期リセットを使用すると…

RTL設計だからというよりも，大規模な論理回路設計では，非同期リセットは使用せず，すべてを同期で実現すべきです．

非同期リセットを使用すると，**図3**のようにBからCのリセット端子をすり抜けDまで到達するパスが存在することになります．大規模設計では，タイミング解析の終点をFFにしないと，遅延値が把握できなくなってきます．