デバイスの記事

SystemVerilog アサーション入門（後編）

静的に解析するプロパティ検証にも利用可能

赤星博輝

本稿では，あらかじめ内部信号のふるまいを定義し，回路がそのとおりに動作しているかどうかをシミュレータに自動監視させるSystemVerilogアサーションについて解説する．前編（本誌2005年9月号，pp.83-94）ではアサーションを使う利点や基本構文，繰り返し記述，シーケンスなどについて解説した．後編ではアサーションを利用する際の注意点やハフマン符号デコーダの検証記述例などを紹介する．（編集部）

● 信号のふるまいに対応した関数が定義されている

アサーションでよく使われる関数のうちのいくつかを説明します（**表1**）．

1）信号変化を調べる関数（$rose，$fell，$stable）

信号の変化状態を調べる関数として，$rose，$fell，$stableの三つがあります．**図1**で示すように，信号aの値が‘0’から次のサイクルで‘1’になったときに成立するのが$rose(a)，信号aの値が‘1’から次のサイクルで‘0’になったときに成立するのが$fell(a)，信号aの値が前のサイクルと同じ場合に成立するのが$stable(a)です．

$rose，$fellはよく使う関数です．例えば，「要求reqを出すと，2～4サイクル以降いつかackが返ってくる」というプロパティについて，二つの記述を作成しました．CHECK1はreqが条件ですが，CHECK2は$rose(req)が条件という点が異なります．この差を**図2**で見てみます．

```
property CHECK1;
  @(posedge clk)   req |->  ##[*2:4] ack;
endproperty
```

表1　アサーションで使われる関数の例

関　数	動　作
$rose	立ち上がりの判定
$fell	立ち下がりの判定
$statble	無変化の判定
$past	過去の信号値を参照
$onehot	信号がワンホットか判定
$onehot0	信号がワンホットもしくはALL0か判定
$countones	2進数表記で‘1’の数を返す
$isunknown	信号にX，Zを含むかを判定
$sampled	アサーションで使用する値を取得

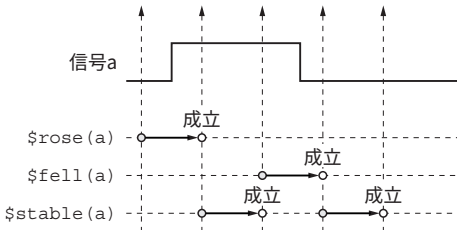

図1　関数$rose，$fell，$stableの動作
$rose，$fell，$stableはそれぞれ，信号の立ち上がり，立ち下がり，無変化の判定に使用する．

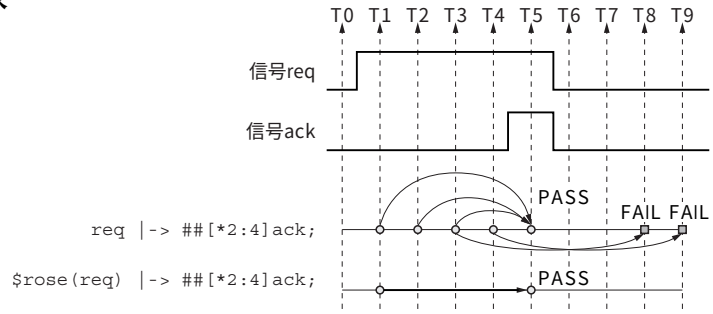

図2　立ち上がり判定$roseを使うと便利な場合
$roseを使わないで記述すると，reqが‘1’になった回数だけ応答するackが必要になる．$roseを使うことで，立ち上がりに対して1回だけackが必要となる．

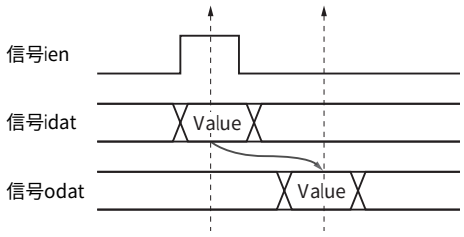

図3　$pastを使う必要がある例
「ienが‘1’ならば，そのときのidatと次のサイクルのodatが等しい」というアサーションを書く場合，$pastが必要になる．

```
property CHECK2;
  @(posedge clk)    $rose(req) |->
                              ##[*2:4] ack;
endproperty
```

CHECK1はreqが‘1’となった場合にチェックを開始するので，reqが‘1’である5サイクルの間（T1～T5）に五つのアサーションが起動されます．このとき，前半の3サイクル（T1～T3）で起動されたCHECK1のアサーションはT5で成立しますが，後半の2サイクル（T4～T5）で起動されたCHECK1のアサーションはT8，T9で非成立となります．

これに対してCHECK2では$rose(req)の場合にチェックを開始するので，T1で起動されるだけになります．

2）過去の値を見るには$past

プロパティによっては，過去の信号の値と現在の信号の値を比較したいことがあります．例えば，「1ビットの信号ienが‘1’ならば，このときの4ビットの信号idatの値と次のサイクルのodatの値が同じである」というプロパティです（**図3**）．これまでの記法では，異なった時刻の信号値を比較することができませんが，$pastを使うことでこの問題を解消できます．$pastの使いかたを以下に示します．

- $past(x)：信号xの1サイクル前の値
- $past(x, 5)：信号xの5サイクル前の値

この$pastを使って先ほどのプロパティを記述してみると，以下のように書けます．

```
ien |=>  odat == $past(idat)
```

3）内部変数を利用したアサーション

SystemVerilogの特徴として，アサーションを記述するときに内部変数を使えることが挙げられます．この機能がアサーションの記述力を大幅に向上させています．

リスト1　内部変数を利用したアサーション

```
property example;
  内部変数定義;
  クロック指定;
  disable iff定義;
  プロパティ定義;
endproperty
```

（**a**）内部変数定義の位置

```
property pDAT;
  reg[3:0] v_dat;
  @(posedge clk)
  ($rose(a), v_dat=idat) |=> odat == v_dat;
endproperty
```

（**b**）内部変数定義を使った記述例1

```
property pDAT2;
  reg[3:0] v_dat;
  @(posedge clk)
  (ien, v_dat=idat) |-> ##[2:3] oen &&(v_dat==odat);
endproperty
```

（**c**）内部変数定義を使った記述例2

まず，プロパティで内部変数を定義する場所は，クロック指定の前になります（**リスト1（a）**）．内部変数の定義方法はVerilog HDLの場合と同じで，Verilog HDLおよびSystemVerilogのデータ型を使用できます．

内部変数に値を代入するためには，,を使って記述します．「1ビットの信号aが‘1’ならば，そのときの信号xの値を内部変数vに代入する」というのは，以下のように記述できます．

```
(a, v=x)
```

その内部変数の値を参照するには，通常の信号と同じように変数名を使用します．

$pastのところで示した「1ビットの信号aが立ち上がりならば，このときの4ビットの信号idatの値と次のサイクルのodatの値が同じである」というプロパティに対して，内部変数を使った記述例を**リスト1（b）**に示します．

$pastだけでは記述しにくい場合でも，内部変数を使うと記述しやすくなることがあります．例えば，「1ビットの信号ienが‘1’ならば，2～3サイクル後に1ビットの信号oenが‘1’になり，そのとき（oenが‘1’になったとき）のodatの値はienが‘1’のときのidatの値である」というプロパティは**リスト1（c）**のように書けます．

● 信号値の観測では実行順序に注意

カウンタの例で実験してみます．例えば，「信号count