クラスの概念を利用して SystemVerilogモデルの再利用性を向上

―― 簡易CPUバス・モデルのクラス記述

宮下晴信

デバイスの記事

ここではクラスを利用したSystemVerilogの記述例を紹介する．SystemVerilogは，C++やSystemCと同じようにクラスの概念に対応している．これを利用すると，シミュレーション・モデルやテストベンチの記述量が減る．また，再利用しやすいモデルを作成することができる．（編集部）

SystemVerilogでは，クラス（class）の概念が導入されました．このクラスは，米国Synopsys社の機能検証ツール（テストベンチ開発環境）である「Vera」(1)から取り入れた機能です．クラスの概念は機能の実装（設計記述）には使うことができませんが，モデリングや検証には非常に役に立ちます．すなわち，シミュレーション・モデルやテストベンチの記述量が減ったり，再利用性が向上したりします．

本稿では，前半でSystemVerilogのクラスの基本的な内容について説明し，後半では本誌2004年8月号，pp.80-90の解説記事「SystemVerilogで簡易CPUバス・モデルを記述」で紹介した簡易CPUバス・モデルの一部をクラスを使って実装してみます．

❶ SystemVerilogのクラス

まず，SystemVerilogのクラスについて説明していきます．

● C++やSystemCのクラスと若干異なる

SystemVerilogのクラスは，C++やSystemCのクラスとほとんど同じです．すでにこれらの言語を使っていれば，それほど抵抗なく使えると思います．ただし，SystemVerilogのクラスはC++やSystemCのクラスと次の点で異なります．

- **ガベージ・コレクションに対応している**
- **モジュールの中でのみ定義する**
- **メンバ・メソッドで関数（function）とタスク（task）をサポートしている**
- **多重継承をサポートしていない**

ガベージ・コレクションはJavaでもサポートされている機能で，メモリ管理を自動的に行ってくれます．つまり，動的に生成したクラスのインスタンスが不必要になった場合，SystemCのようにdelete関数で明示的にオブジェクト用のメモリを解放する必要がありません．

SystemVerilogのクラスは単独では存在できません．あくまでもモジュール内の関数やタスクと同じように，モジュール内で定義し，定義されたモジュール内でのみ使用できます．ただし，SystemVerilogのクラスを別ファイルで定義し，そのクラスを使うモジュールにおいて，クラスを定義したファイルをインクルードすることができます．こうすることでクラスの再利用を行えます．

SystemVerilogのクラスは，モジュールの関数やタスクと同じように，クラスに関数やタスクをメンバ・メソッドとして定義することができます．また，多重継承は行えませんが，モデリングや検証においてこれはとくに問題となりません．

● new関数でクラスを生成・初期化

SystemVerilogのクラスの簡単な例を**リスト1**に示します．**リスト1**では，Basketクラスを定義しています．Basketクラスには，3種類のフルーツ，つまりりんご，バナナ，オレンジを入れることができ，三つのメンバ変数（apple，banana，orange）と四つのメンバ関数（new，add，del，num）を持っています．クラスのメンバ・メソッドは関数とタスクを持てると説明しましたが，ここでは関数についてのみ説明します．タスクについては，簡易CPUバス・モデルのところで説明します．

SystemVerilogにおいて，クラスの生成および初期化はnew関数で行うので，すべてのクラスにnew関数が必要で

リスト1　SystemVerilogのクラス記述

```
typedef enum {APPLE, BANANA, ORANGE} TYPE;

Class Basket;

  int apple;
  int banana;
  int orange;

  function new;
    apple  = 0;
    banana = 0;
    orange = 0;
  endfunction : new

  function void add( TYPE index, int no );
    case( index )
      APPLE  :  apple  += no;
      BANANA :  banana += no;
      ORANGE :  orange += no;
    endcase
  endfunction : add

  function void del( TYPE index, int no );
    case( index )
      APPLE  :  apple  -= no;
      BANANA :  banana -= no;
      ORANGE :  orange -= no;
    endcase
  endfunction : del

  function int num( TYPE index );
    case( index )
      APPLE  :  return apple;
      BANANA :  return banana;
      ORANGE :  return orange;
    endcase
  endfunction : num

endclass : Basket
```

リスト2　Basketクラスの使用例

```
module Test;

`include "Basket.sv"

  Basket basket;

  initial begin

    basket = new;

    if( basket != null ) begin
      $display("APPLE  = %d", basket.num( APPLE  ) );
      $display("BANANA = %d", basket.num( BANANA ) );
      $display("ORANGE = %d", basket.num( ORANGE ) );
      $display(" ");

      basket.add( APPLE,  5 );
      basket.add( BANANA, 4 );
      basket.add( ORANGE, 3 );

      $display("APPLE  = %d", basket.num( APPLE  ) );
      $display("BANANA = %d", basket.num( BANANA ) );
      $display("ORANGE = %d", basket.num( ORANGE ) );
      $display(" ");

      basket.del( APPLE,  4 );
      basket.del( BANANA, 3 );
      basket.del( ORANGE, 2 );

      $display("APPLE  = %d", basket.num( APPLE  ) );
      $display("BANANA = %d", basket.num( BANANA ) );
      $display("ORANGE = %d", basket.num( ORANGE ) );
      $display(" ");
    end

  end
```

す．Basketクラスのnew関数では，三つのメンバ変数を0にしています．SystemVerilogのnew関数は，SystemCのコンストラクタに相当します．

モジュールの関数やタスクと同様に，クラスの関数やタスクにもendfunctionやendtaskの後にラベルを付けることができます．**リスト1**のBasketクラスでも各関数のendfunctionの後にラベルを付けています．

クラスの関数は，モジュールの関数と同じように引き数および戻り値を持つことができます．戻り値には，戻り値がないvoidを使うこともできます．

リスト1のadd，del，num関数の第1引き数のように，typedefで定義した型を使うこともできます[注1]．add，del，num関数の第1引き数の型は，列挙型をtypedefで定義したタイプになり，三つの値（APPLE，BANANA，ORANGE）のみ指定できます．

Basketクラスの各関数は次のようなことをします．add関数は，第1引き数で指定したフルーツを第2引き数で指定した個数分入れます．del関数は，第1引き数で指定したフルーツを第2引き数で指定した個数分取り出します．num関数は，第1引き数で指定したフルーツがいくつあるかの値を返します（ただし，add，del関数については，エラー処理を行っていない）．

● includeディレクティブを使ってクラスを読み込む

リスト2は，Basketクラスの使用例です．**リスト1**のコードをBasket.svというファイルにし，Testモジュール内でincludeディレクティブを使って読み込んでいます．

次にBasketクラスのインスタンスbasketを宣言しています．この時点では，インスタンスbasketはBasketクラスのインスタンスとしては初期化されていません．初期化されるのは，new関数を実行したときです．その代わりに，インスタンスbasketの値はnullに初期化されます．

SystemVerilogでは，クラスのインスタンスを宣言したとき，そのインスタンスはnullに初期化されます．インスタンスをnullと比較することで，new関数が実行され

注1：SystemVerilogでは，typedefによって新しいタイプを定義できるようになった．