連 載

SystemC言語入門（第５回）

EDAツール

初めてでも使えるSystemC文法ガイド

湯山洋一，小林和淑

システムLSIと組み込みソフトウェアの両方をモデリングできるC/C++ベース言語「SystemC」に関する連載の５回目である．今回はSystemCの文法について，あらためて解説する．まず，SystemCにおけるシステムの構成とファイルの構成について説明する．SystemCの文法要素は，回路機能向けの記述，通信機能向けの記述，シミュレーション向けの記述に分かれる．（編集部）

前回までの記事ではSystemCの基本的な使いかたや，システムのモデル化，性能評価について説明してきました．今回はこれまでのおさらいとして，システムのモデル化に使用するSystemCの文法要素について解説します．

1. システム構成とファイル構成

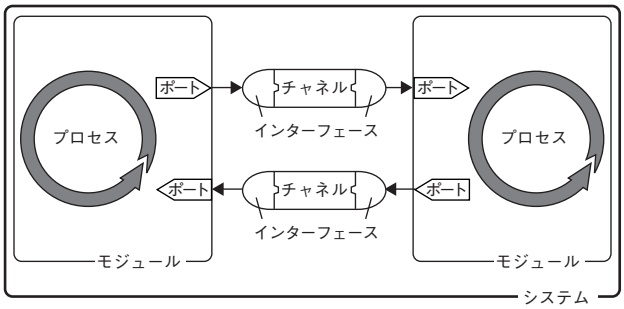
SystemCは，C++のクラスを利用して，ソフトウェアだけでなくハードウェアも記述できるようにしたシステム設計言語です．みなさんの中には，「Cなら知っているけど，C++はちょっと」という方も多いと思います．しかしSystemCでは，C++のキーワードである「クラス」，「コンストラクタ」，「継承」，「メンバ関数」，「テンプレート」がどのようなものであるかさえ知っていれば，後の難しいことは知らなくても記述できます．

幸い，C++の参考書は書店に行けば山のようにあります．その中のどれにも上記のことばの説明が載っているはずです．今回の連載でも，三つのキーワードについては何の断りもなく使っています．C++をまったく知らないという人は，最低限これらのキーワードがどういう概念であるかを理解しておいてください．ただし，詳細な使いかたまで覚える必要はありません．

まずSystemCにおけるシステムの構成要素とそのファイル構成について説明します．

1）SystemCのシステム構成

SystemCでは，システムはいくつかの「モジュール」と，モジュールどうしをつなぐ「チャネル」から構成されます（**図1**）．モジュールの機能は「プロセス」と呼ばれる同時並行に実行される関数の中に記述します．回路の処理機能をモジュールに記述し，モジュール間の通信をチャネルに記述することで，機能と通信を完全に分離することができます．このため，システムを構築する段階で通信方式に変更を加えても，機能部分を変更する必要はありません．また，機能と通信の抽象度を個別に変更することが非常に容易です．



〔図1〕SystemCによるシステム構成

SystemCでは，いくつかの「モジュール」と，モジュールどうしをつなぐ「チャネル」によってシステムを構成する．モジュールの機能は「プロセス」として記述され，モジュールを階層化することも可能である．回路の処理機能と通信を分離することでモジュールの独立性を高め，プロトコルの変更にも柔軟に対応することができる．

2）SystemCのファイル構成

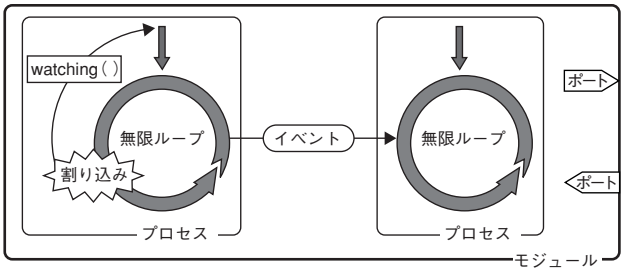
モジュール記述は通常，ヘッダ・ファイルと実装ファイルで構成されます．ヘッダ・ファイルでは，モジュール定義，ポート/チャネル/ほかのモジュールなどのインスタンス化，プロセスの宣言/登録などが行われます．実装ファイ

〔表1〕SystemC文法要素

用　語	関連するキーワード	説　明
モジュール	SC_MODULE	SystemCにおける設計の基本要素となるクラス．プロセス，ポート，チャネル，ほかのモジュールなどをメンバとして持ち，階層構造にできる．
プロセス	SC_METHOD SC_THREAD SC_CTHREAD	モジュールの動作機能を記述した関数をプロセスと呼ぶ．一つのモジュールに対してプロセスをいくつでも定義でき，プロセスは同時並列に実行される．SC_METHODプロセス，SC_THREADプロセス，SC_CTHREADプロセスの3種類のプロセスが用意されている．
プロセスの中断	wait( )	SC_THREADプロセスとSC_CTHREADプロセスは，実行を一時的に中断することができる．実行の再開は，次に説明するセンシティビティにより決定する．
センシティビティ	sensitive sc_event wait( )	実行を一時中断したプロセスが，いつ実行を再開するかを決定するのがセンシティビティである．プロセスは複数のイベントや信号に対してセンシティビティを持つことができる．
割り込み	watching( )	SC_THREADプロセスやSC_CTHREADプロセスは無限ループとして記述される．これに対して割り込みをかけ，実行の制御をプロセスの先頭に戻すことができる．
イベント	sc_event wait( )	実行を一時的に中断したプロセスに対して，イベント通知を行うことで，プロセスの実行を再開させることができる．
インターフェース	sc_interface	通信用の関数の宣言を行なうためのクラス．ただし，インターフェースでは関数の実装そのものは行われない．
チャネル	sc_channel	インターフェースで宣言した通信関数を実装するためのクラス．複数のインターフェースを実装することもできる．
ポート	sc_port	モジュールからチャネルの中の通信関数にアクセスするために使用されるクラス．

ルではプロセスの定義が行われます．ただし，モジュールをテンプレート・クラスにした場合は，ヘッダ・ファイル内でプロセスを定義しなければなりません．通常，ヘッダ・ファイルは“モジュール名.h”，実装ファイルは“モジュール名.cpp”とし，各モジュールごとに作成します．チャネルについても同様です．クラスの定義や関数の宣言はヘッダ・ファイルで行い，関数の定義は実装ファイルで行います．

また，最上位階層としてシステム・ファイルが必要になります．このファイルではsc_main( )関数の定義を行います．sc_main( )関数ではモジュールやチャネルのインスタンス化と接続を行い，さらにシミュレーションの制御も行います．



〔図2〕SystemCによる回路機能記述

モジュールの機能は同時並列に実行される「プロセス」として記述される．wait( )関数によってプロセスの実行を一時中断したり，watching( )によって割り込みをかけることもできる．「イベント」を用いることで，並列に実行されるプロセス間の通信も行える．

2. 文法ガイド ～回路機能編～

ここからは，SystemCのモデル化に使用する文法要素について解説します．**表1**はSystemCで用いられる主な文法要素をまとめた表です．まずは回路機能の記述に用いる文法要素である「モジュール」と「プロセス」について説明します(**図2**)．また，**リスト1**に回路記述のサンプルを示します．

●モジュール

モジュールは，SystemCのSC_MODULEクラスを継承する設計の基本要素となるクラスです．Verilog HDLにおけるmoduleや，VHDLのentityに相当します．プロセス，ポート，チャネル，ほかのモジュールなどをメンバとして持ち，階層構造を記述することもできます．複雑なシステムも，モジュールに分割することで設計が容易になります．

1）宣言/定義

モジュールの定義(宣言)は以下の形式で記述します(**リスト1**の5，17～22行目)．

```
SC_MODULE(モジュール名){
  // プロセスの宣言を記述
  // ポートやチャネルのインスタンス化
     ・・・・・
  SC_CTOR(モジュール名){
    // コンストラクタ
```