設計ツール

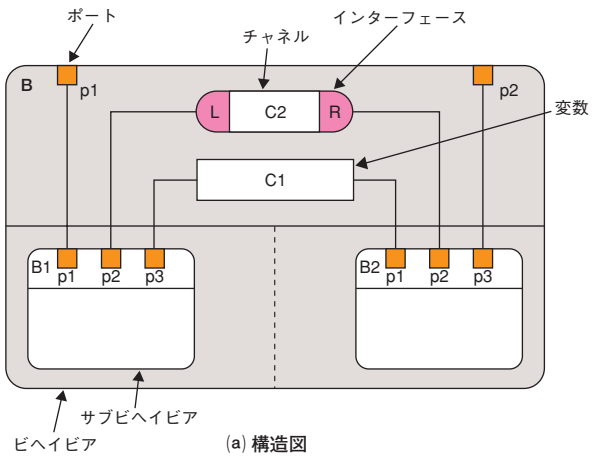
第2章

SpecC言語の文法概要

木下常雄

ここでは仕様記述のためのシステム・レベル言語「SpecC」の文法を紹介します．SpecCは，ANSI Cに，ハードウェアやリアルタイム処理をともなうソフトウェアの動作を表現するための記述を追加した言語です．たとえば，並行実行や同期処理，例外処理，ブーリアン型などの文法を備えています．SpecC言語の記述をC++に変換するプリプロセッサは，米国University of California, Irvine のSpecC言語のホームページ(http://www.ics.uci.edu/~specc/)から無償でダウンロードできます．現在，Solaris版とLinux版が公開されています．同大学の許可が得られたので，本誌付属のCD-ROMにも収録しています．(編集部)

SpecC言語は，ANSI Cの構文に仕様記述のための文法を追加した，システムを記述・設計するための言語です．SpecCを使えば，ソフトウェアとハードウェア（電子回路）を分けへだてなく記述できます．ご存じのように，C言語はソフトウェア開発やハードウェアのアルゴリズム設計の世界で普及が進んでおり，そのコード資産は，膨大なものがあります．SpecCにはハードウェアのモデル化に必要な文法が盛り込まれているので，ソフトウェア開発者だけでなく，ハードウェア設計者にとっても受け入れやすいと思われます．



〔図1〕SpecCモデルの構造

ビヘイビアBはサブビヘイビアB1とB2をもち，サブビヘイビア同士は並列に動作する．二つのサブビヘイビア同士は，変数C1とチャネルC2を経由してコミュニケーションし合う．

1 システム・レベル言語「SpecC」の特徴

ここでは，SpecCの特徴について説明します．

システム・レベル言語では，一つの構文でモデル全体の全階層を記述できるのが好ましく(これをホモジニアスと呼ぶ)，

```
interface L { void Write ( int x ) ; } ;                    インターフェースを定義する
interface R { int Read ( void ) ; } ;

channel C implements L ,  R                                  チャネルCはインターフェースLとRを実装している
{
  int Data ;  bool Valid
  void Write ( int x )
       { Data = x ; Valid = true ; }
  int Read ( void )
       { while ( ! Valid ) waitfor ( 10 )
         return (  Data  ) ; }
} ;

behavior B1 ( in int p1 , L p2 , in int p3 )                 サブビヘイビアB1の定義
{
  void main ( void )
       { /* ...  */ p2 . Write ( p1 ) ; }
} ;

behavior B2 ( out int p1 , R p2 , out int p3 )               サブビヘイビアB2の定義
{
  void main ( void )
       {  /* ...  */ p3 = p2 . Read () ; }
} ;

behavior B ( in int p1 , out int p2 )                        ビヘイビアBの定義
{
  int c1 ;
  C c2 ;
  B1 b1 ( p1 , c2 , c1 ) ;
  B2 b2 ( c1 , c2 , p2 ) ;
  void main ( void )
       { par { b1 . main () ; b2 . main () ; } }
 };
```

(b) 記述例

「ある部分はこの言語で，別の部分はあの言語で」，というように，異なる言語を併用する（ヘテロジニアスと呼ぶ）方法はとりません．ホモジニアスにしておくと，シミュレータ間のインターフェースや言語から言語への翻訳などにわずらわされずにすむからです．ハードウェアとして実装するために詳細設計を行う場合にも，一つの記述を詳細な表現（RTL とかゲートとか）に変換するだけですみます．ライブラリに登録しておいた設計データをプラグ・アンド・プレイで新しい設計の一部に利用できるので，再利用も行いやすくなります．

●システム・レベル言語に対する三つの要件

システム・レベル言語と名乗るからには，以下の三つの要件を満足する必要があります．

(1) 実行可能

記述をそのままコンパイルして実行できるのであれば，そのコードはシミュレーションに利用できます（特別なシミュレーション・ソフトウェアなどは必要ない）．SpecC の記述を実行する際には，まずプリプロセッサ（前処理のソフトウェア）によってSpecC 記述をC++ 記述に変換し，さらにC++ コンパイラでオブジェクト・コードに変換します．

(2) モジュラリティ

システムの動作を記述するときには，機能とコミュニケーション（接続）を完全に分けて記述しておく必要があります．こうした特徴をモジュラリティと呼びます．モジュラリティがあると，設計で使用する部品を階層的に記述できます．ビヘイビアも階層化でき，システムをシリアル，あるいはパラレルに動作するサブビヘイビア（子供のビヘイビア）の集まりとして記述できます．また，部品間の接続構造も記述できます．

(3) 完全性

システム・レベル言語として「完全である」とは，構造的階層，ビヘイビア，並列性，同期，例外処理，タイミング，状態遷移をすべて扱えることを意味します．また，これらはお互いに独立（直交）の関係にあり，必要な構文の種類がもっとも少なくなるようになっています．

●構造的階層

システム機能とは，階層のあるチャネルで相互に接続されたビヘイビア階層のネットワークのことです．SpecC によるシステム機能の記述は，ビヘイビア，チャネル，インターフェースからなります．

図1(a)はSpecC 記述の基本構造です．また，**図1**(a)の構造をSpecC で記述したものが**図1**(b)です．この例ではビヘイビアB は二つのサブビヘイビアB1 とB2 をもっており，これらは並列に実行されます．また，B1 とB2 は変数C1 とチャネルC2 を経由してコミュニケーション（ハードウェアの信号の伝搬，ソフトウェアのデータの受け渡しに相当）し合います．機能の動作とコミュニケーションにかかる時間はビヘイビアで決まり，コミュニケーションの仕方はチャネルで決まります．インターフェースはビヘイビアとチャネルの間をつなぎます．

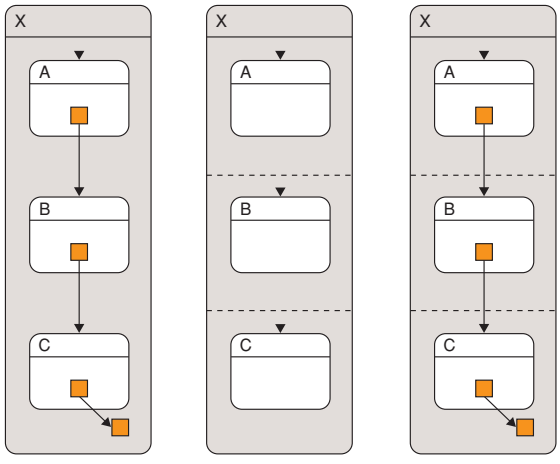
●ビヘイビアの階層化

あるビヘイビアに含まれるサブビヘイビアの構成をビヘイビア階層と呼びます．サブビヘイビアにはシリアルとパラレルの二つの形式があります．シリアル型の動作は，標準的な命令型文（手続き型プログラミング言語で見られる代入文やif 文などのこと）やステートマシン（有限状態機械）で表現されます．パラレル型には並行実行とパイプラインの2 種類があります．

図2(a)は，シリアル型のサブビヘイビアを示しています．最後のサブビヘイビアC が終了したとき，ビヘイビアX が終了します．**図2**(b)は，par 構文で記述した並行実行の例です．すべてのサブビヘイビアA，B，C が終了したときに，ビヘイビアX が終了します．**図2**(c)は，pipe 構文で記述したパイプラインの例です．最初にA を，2 回目にA とB を，3 回目にA とB とC を実行し，これ以後は，A とB とC をエンドレスで実行します．

並行実行の宣言は，以下のように記述します．

```
par{  a.main();  b.main();  c.main();  }
```



(a) シリアル型　(b) パラレル型（並行実行）　(c) パラレル型（パイプライン）

〔図2〕ビヘイビアの階層化

シリアル型では，サブビヘイビアA，B，C の順に実行し，C が終わるとビヘイビアX が終了する．パラレル型の並行実行では，サブビヘイビアA，B，C が並列に実行される．パラレル型のパイプラインでは，まずA が，次にA とB が，その次にA とB とC が実行され，以降はA とB とC の実行が繰り返される．