Verilog-AMS 入門講座

桜井 至

第4回

Verilog-AMSの基本構文③

――システム・タスク，コンパイラ指示子，ディスプリン，ネイチャ

アナログ/ミックスト・シグナル対応のハードウェア記述言語である「Verilog-AMS」の入門講座の第4回をお届けします．今回は，構文紹介の最終回として，システム・タスク，コンパイラ指示子，ディスプリン，ネイチャなどを紹介します．ディスプリンとネイチャは，アナログ回路，ディジタル回路，電子機械部品（たとえばモータ）など，さまざまなドメインのモデルを表現できるVerilog-AMSならではの考え方です．また最後に，Verilog-AMSがサポートするSPICEプリミティブにも触れます．（編集部）

前回は，Verilog-AMSの演算子，関数，および階層構造について紹介しました．今回は，構文紹介の最終回として，システム・タスク，コンパイラ指示子，およびVerilog-AMS特有の構文であるディスプリンとネイチャを紹介します．

1．システム・タスク

システム・タスクはVerilog-AMS言語の拡張と位置付けられます．Verilog-AMSでは，次に示す3種類のシステム・タスクが存在します．

- 標準アナログ・システム・タスク
- ランダム発生システム・タスク
- 入出力操作システム・タスク

●標準アナログ・システム・タスク

標準アナログ・システム・タスクは，現在のシミュレーション時間や温度のようなシステム・レベルの値を返す関数です．**表1**に標準アナログ・システム・タスクの一覧を示します．

制限された指数関数`$limexp`は，実際はシステム・タスクではなく，アナログ演算子として動作します．前回はアナログ演算子として紹介しています．

シミュレーション時間システム・タスク`$realtime`は，現在のシミュレーション時間を秒単位で返します．`$realtime`は，時間に関連した動作を記述する場合や，`$strobe`において実行時間を表示させるために使用することができます．

周囲温度システム・タスク`$temperature`は，周囲温度をケルビン単位で返します．また，熱電圧システム・タスク`$vt`は熱電圧kT/qを返します．この`$vt`は，オプションとして，温度を引き数として定義することができます．`$vt`（温度）は指定された温度での熱電圧を返します．

〔表1〕標準アナログ・システム・タスク

システム・タスク	引き数	戻り値
`$limexp( )`注1	式	式の指数．
`$realtime`	なし注2	現在のシミュレーション時間（s）．
`$temperature`	なし	周囲温度（ケルビン）．
`$vt( )`	《温度》	熱電圧(kT/q)．オプションで`temperature`での熱電圧を返す．

《 》ではさまれた部分はオプション．

注1：`$limexp`はVerilog-AMS LRM v1.5で`limexp`に変更される．
注2：Verilog-AMS LRM v1.5では引き数としてスケール・ファクタを定義可能．

●ランダム生成システム・タスク

Verilog-AMSはVerilog-Dと同様に，複数のランダム生成システム・タスクを用意しています．各システム・タスクは実行するごとに整数のランダム値を返しますが，その値の分布は実行したシステム・タスクによる確率分布として生成されます（**表2**）．それぞれのシステム・タスクに同じ値の`seed`を指定した場合には，システム・タスクごとに同じラン

〔表2〕ランダム発生システム・タスク

システム・タスク	引き数	説明
`$ramdom( )`	`seed`	32ビットのランダム数．
`$dist_chi_square( )`	`seed, freedom`	ランダム数のc^2分布．`freedom`引き数は整数で，分布に対する自由度の数値を示し，密度関数の形を決めるのに役立つ．
`$dist_exponential( )`	`seed, mean`	平均が`mean`値に近づくランダム数の指数分布．
`$dist_normal( )`	`seed, mean, std_dev`	平均が`mean`値に近づくランダム数の正規分布．`std_dev`は密度の形を決める．
`$dist_poisson( )`	`seed, mean`	平均が`mean`値に近づくランダム数のポアソン分布．
`$dist_t( )`	`seed, freedom`	ランダム数のスチューデント式t分布．`freedom`は密度関数の形を決めるのに役立つ．
`$dist_uniform( )`	`seed, start, end`	`start`と`end`で指定された範囲での均一分布ランダム数．

注：`$dist_*`システム・タスクは，Verilog-AMS LRM v1.5から`$rdist_*`に変更される．

〔リスト1〕seedへの不正な代入の例

```
module top;
   integer seed;
   ...
   analog begin
      @(initial_step)
         seed = 86;  // seedの初期化(正常な代入)
      y = $dist_uniform(seed, 1, 100);
      seed = seed + 1; // seedへの不正な代入
      ...
   end
endmodule
```

〔リスト2〕ノイズを考慮したデバイダの記述例

```
module divide(out, in);
   input in;
   output out;
   electrical in, out;

   parameter integer seed = 999; // seed値
   parameter real jitter = 100p; // ジッタのばらつき値
   parameter real mean  = 0;     // 平均値
   parameter real std_dev = 1;   // 標準偏差
   parameter real delay = 2n;    // 出力遅延値

   real delta_t;
   integer count; // 初期値は0

   analog begin
      @(cross(V(in)-2.5, +1) begin // V(in)の2.5Vとの立ち上がり
                                   //          交差の検出
         count = !count
         //ガウス分布ランダム発生の定義
         delta_t = jitter * $dist_normal(seed, mean, std_dev);
      end
      // transitionによる出力遅延
      // delta_tは出力遅延のばらつきを定義
      V(out) <+ transition(count*5, delay+delta_t, 0.1n, 0.1n);
   end
endmodul
```

ダム値がつねに返されます．

ランダム生成システム・タスクでは，引き数seedは整数で，入出力パラメータとして動作します．一度seedに初期値を設定すると，その値はシミュレータ(システム)しか変更できません．初期指定されたseedが同一であれば，同一系列のランダム値が返されます．そのため，シミュレーション結果の再現性は保証されます．

Verilog-AMSソース記述のなかに，seed引き数への別の代入が存在してはいけません．**リスト1**にその例を示します．@initial_stepでseedに初期値を設定してから，引き数seedを指定して$dist_uniformを実行していますが，その後，seedの値を変更することはできません．

ランダム数の正規分布$dist_normalは，過渡解析における大信号ノイズを生成するために使用できます．**リスト2**に示す分周器モジュールは，平均的なジッタのガウス分布を生成するために$dist_normalシステム・タスクを使用しています．出力信号V(out)にはジッタの統計値を使ってノイズが代入され，出力遅延は標準偏差によるばらつきをもって出力されます．

〔表3〕入出力システム・タスク

システム・タスク	引き数	機能
$fopen( )	ファイル名	ファイルを開いて，チャネル番号を戻す．
$fclose( )	チャネル識別子，文字列，変数	一つ以上のチャネルを閉じる．
$strobe( )	文字列，変数	データを標準出力に書き出す．
$fstrobe( )	チャネル識別子，文字列，変数	一つ以上のチャネルを介してデータをファイルに書き出す．
$readmemb( )	ファイル名，配列名	バイナリ・データのASCII・ファイルを配列(メモリ)に読み込む．
$readmemh( )	ファイル名，配列名	16進数データのASCIIファイルを配列(メモリ)に読み込む．
$readmemr( )	ファイル名，配列名	実数データのASCIIファイルを配列(メモリ)に読み込む．

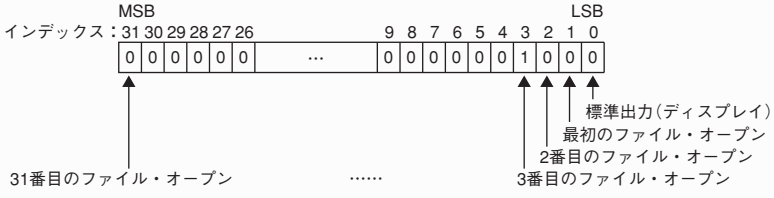
●入出力システム・タスク

Verilog-AMSは入力と出力の処理を行うシステム・タスクを用意しています(**表3**)．出力はASCIIファイルと標準出力へのデータの書き込みに関連したもので，入力はASCIIファイルからメモリ配列へのデータの読み込みに関連したものです．

ファイル・オープン$fopenとファイル・クローズ$fcloseのシステム・タスクは，ファイル名に対応するチャネルに番号を割り付け，ファイルをオープンしたりクローズしたりします．$fopenの構文は次のとおりです．

```
チャネル識別子 = $fopen("ファイル名");
```

ファイル名は$fopenがチャネル識別子に返すチャネル番号と対応付けされます．$fopenシステム・タスクは32ビットに対応するチャネル番号を返します(**図1**)．インデックス0はチャネル番号1(つまり，2^0)で，このチャネルは標準出力として予約されており，普通は0になっています．インデックス1から31は，ファイルがオープンされるたびにインデックス1から順番に1ビットだけを1にし，そのほかのビットはすべて0になる値を返します．つまり，最初の$fopenはチ



〔図1〕$fopenに対するチャネル識別子の値

ファイルをオープンするごとに，インデックス1から順番に，オープン・ファイルに対応するビットが1になる(ほかはすべて0)．最下位ビットは標準出力に対応する．図では，3番目の$fopenによって戻すチャネル番号を示す．