第2章

初めてでも使える Verilog HDL文法ガイド《改訂版》

Verilog HDL 2001仕様に合わせて改訂

小林 優

SystemVerilogの詳細な説明に入る前に，従来のVerilog HDLの文法についておさらいする．本稿は，本誌2002年5月号の別冊付録に収録された「初めてでも使えるHDL文法ガイド Verilog HDL編」をVerilog HDL 2001仕様に合わせて改訂したものである．Verilog HDL 2001では，それまで文法的にあいまいとされてきた部分などが修正されている．（編集部）

Verilog HDLは，1995年にIEEE 1364として標準化されましたが，その後さまざまな修正と拡張が行われ，2001年にIEEE 1364-2001として新しい標準になりました．既存の機能もそのまま使えるように，仕様追加の形で改訂されています．

そこで，新たにHDL設計を始める方でもわかるように，Verilog HDL 2001の文法を「記述スタイル編」と「文法ガイド編」に分けて解説します．なお，記述スタイル編についてですが，新旧両方のスタイルが可能な場合には併記せず，新スタイルのみを紹介しています．文法ガイド編では両方を併記しています．

記述スタイル編

1 モジュール構造

● 基本構造

回路を記述する基本構造がモジュールです（図1）．モジュールは予約語のmoduleとendmoduleで囲まれ，回路表現からテストベンチ（検証用の記述）まで，すべてこの中で記述されます．moduleに続きモジュール名，ポート・リストを記述します．モジュール名やポート名などの識別子には英数字と_（アンダ・スコア）が使え，大文字小文字を区別します．

```
module　モジュール名(
ポート・リスト
);

    変数宣言
    ネット宣言
    パラメータ宣言

    モジュール構成要素
        assign
        function
        always
        generate
        下位モジュール接続など

endmodule
```

図1 モジュール構造
回路記述もテストベンチも，すべてこのモジュール構造で記述する．さらに1モジュールは1ファイルで記述し，ファイル名とモジュール名を一致させておくとよい．

ポート・リストでは，入力/出力の方向，データ型，ビット幅，端子名を定義します（**リスト1(a)**）．双方向ポートはinoutを用います．ビット幅の表現は[0:31]や[32:1]も可能ですが，[MSB:LSB]として扱います．ポートの宣言をかっこの外で定義するスタイルもあります（2.6節の「generateによる回路の繰り返し」のところで説明）．

● 宣言部でネット信号や変数などを宣言

モジュールの最初に，内部で使用する信号の宣言を行います．

ネット宣言（**リスト1(b)**）では，回路記述で使用するネットをあらかじめ宣言します．ネット宣言では信号強度や

リスト1　各種宣言例

```
module MODULE_NAME (
input  wire  CLK, RST,           // 入力
input  wire  [15:0] BUS1, BUS2,  // 入力バス信号
output reg   BUSY,               // 出力
inout  wire  [31:0] DBUS         // 双方向バス信号
);
```

(a) ポート・リスト

```
wire  ENBL;
wire  [15:0] BUS;          // バス信号
wire  signed [31:0] DOUT; // 符号付き32ビット
wire  #3000 TRIG;          // 遅延付加ネット
wire  (strong0, pull1) #100 OC_OUT = TRIG;
                                // オープン・コレクタ
```

(b) ネット宣言

```
reg  FF1, FF2;            // フリップフロップ
reg  [3:0] CNT4;          // 4ビット・カウンタ
reg  [7:0] MEM [0:1023]; // 1Kバイト・メモリ
```

(c) 変数宣言

```
parameter  STEP=1000;           // 1クロック周期
parameter  WAIT=2'b00, INIT=2'b01, ACTION=2'b10;
                                // ステート名
parameter  MEMSIZE=1024;        // メモリ・サイズ
reg  [7:0] MEM [0:MEMSIZE-1];  // 1Kバイト・メモリ
```

(d) パラメータ宣言

遅延を付加することもできます．ネット型にはwireのほかに**表3**(本稿の文法ガイド編，p.51)に示すようなものがあります．

多ビットの信号は，デフォルトで符号なしとして扱います．符号付きの信号にする場合には，予約語のsignedを付加します．これにより，演算時に符号拡張などが自動で行われます．

変数宣言(**リスト1(c)**)では，フリップフロップやラッチなどの値を保持する信号を定義します．メモリは配列として定義します．メモリをビット単位でアクセス(代入や参照)する場合は，mem[0][5]のようにします．これは，0番地のビット5です．変数には**表2**(文法ガイド編，p.51)に示すタイプがあります．integerはテストベンチの中でよく用いられます．

パラメータは定数の代わりの識別子として用います(**リスト1(d)**)．例えば1クロックの周期やメモリ・サイズなどです．値を変えてシミュレーションし直すときに重宝です．また，ステート・マシンの状態名に使うことで記述の可読性が向上します．

● コメントは2種，フリー・フォーマット

コメントは，

- /* ～ */　　…　複数行コメント
- //で始まり行末まで　…　1行コメント

があります．フリー・フォーマットなので，記述を見やすくするためにスペースやタブ，改行を自由に挿入できます．

2 RTL記述

RTLとは，意訳すれば「詳細なブロック図レベル」です．ANDゲートやORゲートを記述したりフリップフロップを並べるのではなく，セレクタやカウンタのレベルで記述します．これがRTLです．

RTLの記述には，以下の五つの記述スタイルがあります．

- assignによる組み合わせ回路
- functionによる組み合わせ回路
- alwaysによる組み合わせ回路
- alwaysによる順序回路
- 下位モジュール接続

2.1　assignによる組み合わせ回路

論理式1行で記述できる組み合わせ回路は，assignで記述できます．**図2**に例を示します．信号が1ビットか多ビットかは宣言で決まります．したがってセレクタや加算回路などの記述はビット数に影響されません．

assignにより，AND，ORなどのゲート回路から，セレクタ，加算回路なども記述できます．関係演算子(==や<=など)は真のとき‘1’，偽のとき‘0’となるので，**図2**のけた上がり信号(CARRY)のようなこともできます．また，ネット型の信号宣言と，この信号に対する代入を一度に記述することができます(**リスト2**)．

各代入文は同時に実行されます．記述が前後しても動作に違いはありません．「代入」というより「接続」と考えたほうがわかりやすいでしょう．