初歩からのHDLテストベンチ

第4回　標準出力の記述方法

安岡貴志

デバイスの記事　ビギナーズ

HDLで回路を記述できるようになったばかりで，これからテストベンチを書こうとしている方を対象とした連載の第4回である．前回までは波形での検証結果確認を前提に，基本的なテストベンチの作成方法を解説した．今回は，波形の目視に頼らない検証結果の確認方法として，標準出力の記述方法を解説する．（筆者）

標準出力とは，処理結果を文字列で出力するものと考えてください．UNIXやLinuxなどのターミナル上でシミュレーションを実行しているのあれば，そのターミナル〔**図1(a)**〕です．シミュレーション・ツールで専用ウィンドウを表示しているのであれば，その中でログなどが表示されるウィンドウ〔**図1(b)**〕です．

1．標準出力の書き方

Verilog HDL

Verilog HDLでテキストを出力するためには，システム・タスク$displayを使います．

図2(a)に$displayの書式を示します．かっこの中に書かれた信号の値や，ダブル・クォーテーション(" ")の中に書かれた文字列を標準出力に表示します．シミュレーション中にこのタスクが実行されると，その時点の信号の値とダブル・クォーテーションの中の文字列が標準出力に表示されます．

図2(b)に$displayの記述例と表示例を示します．

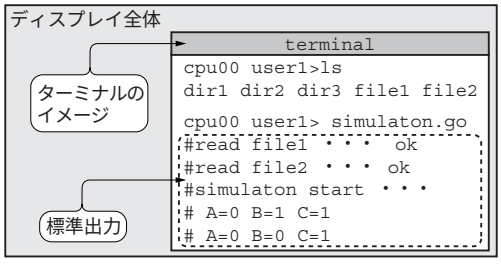
ダブル・クォーテーションの中の文字列の中に%(とそれに続く1文字)が含まれると，文字列に続く信号名の値が，%(とそれに続く1文字)と置き換えられて表示されます〔**図2(c)**〕．

$displayは，initial文やalways文の中で使います〔**図2(d)**〕．また，ダブル・クォーテーションの中では，特殊文字を使うことができます〔**図2(e)**〕．

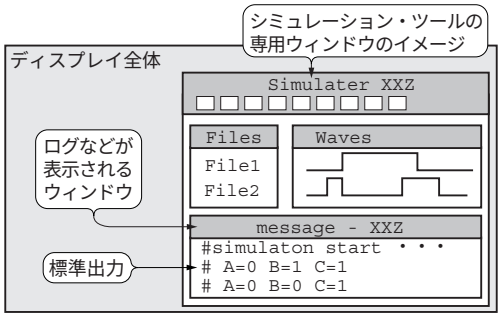
VHDL

●line変数

VHDLでテキストを入力したり，出力したりするために



(a) UNIXやLinuxのターミナルのイメージ



(b) シミュレータ専用ウィンドウのイメージ

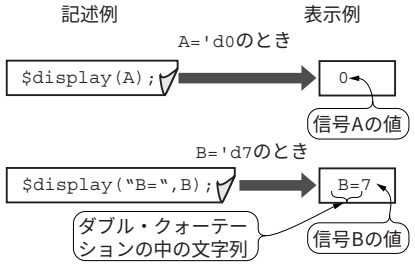
図1　標準出力

標準出力は，UNIXやLinuxなどのターミナル上でシミュレーションを実行していれば，そのターミナル，シミュレーション・ツールで専用ウィンドウを表示していれば，その中でログなどが表示されるウィンドウになる．

KeyWord　テストベンチ，テスト入力，絶対時間，相対時間，fork，wait，after，assert文，エンコーダ，for文

```
$display(信号名);
$display(信号名,信号名,・・・);
$display(“文字列”,信号名);
$display(“文字列”,信号名,“文字列”,信号名,信号名,・・・);
```

（**a**）書式



（**b**）記述例と表示例

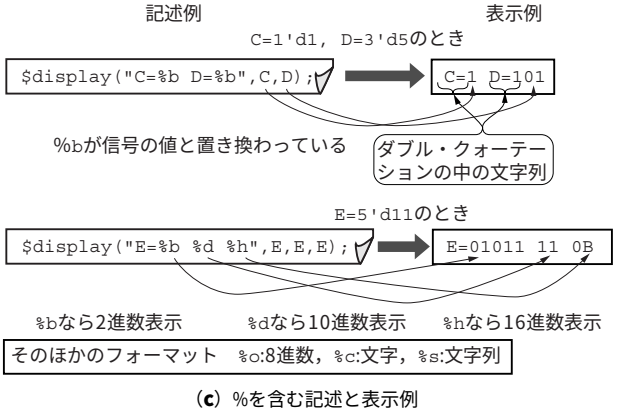
```
module and_comb_tb2();
reg   SA,SB;
wire  SY;

and_comb and_comb(.A(SA), .B(SB), .Y(SY));

initial begin
     SA = 0; SB = 0;
#100 SA = 1; SB = 0;
#100 SA = 0; SB = 1;
#100 SA = 1; SB = 1;
#100 $finish;
end

initial begin
#50  $display("A=%b B=%b Y=%b",SA,SB,SY);
#100 $display("A=%b B=%b Y=%b",SA,SB,SY);
#100 $display("A=%b B=%b Y=%b",SA,SB,SY);
#100 $display("A=%b B=%b Y=%b",SA,SB,SY);
end

endmodule
```

```
module and_comb ( A, B, Y);
input A,B;
output Y;

assign Y = A & B;

endmodule
```

出力結果

```
A=0 B=0 Y=0
A=1 B=0 Y=0
A=0 B=1 Y=0
A=1 B=1 Y=1
```

（**d**）テストベンチ全体の記述例と表示例



（**c**）%を含む記述と表示例

```
特殊文字
\n      改行
\t      タブ
\\      バックスラッシュ（\）
\"      ダブルクォート（"）
```

※ただし、半角文字の“ \ ”は日本語環境では“ \ ”となる

記述例

```
$display("King \nQueen");
$display("White \tRabbit");
$display("\ \(^.^)/");
$display("\"Here! \"cried Alice");
```

表示例

```
King
Queen
White  Rabbit
\(^.^)/
"Here!"cried Alice
```

（**e**）特殊文字

図2　システム・タスク$displayの使い方

Verilog HDLでテキストを入力したり出力したりするためには，システム・タスク$displayを使う．

```
variable 変数名  : line;
```

（**a**）書式

```
variable LO: line;
```

（**b**）記述例

```
library STD;
use STD.TEXTIO.all;
```

（**c**）TEXTIOパッケージの呼び出し

図3
line変数

1行分のテキスト・データを蓄えるための変数である．

は，まず1行ごとのテキスト・データを蓄えます．1行分のテキスト・データを蓄えるための変数をline変数といいます．

図3(a)と**図3(b)**にline変数の宣言の書式と記述例を示します．また，line変数を使うには，STDライブラリのTEXTIOパッケージが必要です［**図3(c)**］．

● プロシージャwrite

writeはline変数に値を代入するプロシージャ（関数，もしくはサブプログラムのようなもの）です．writeはVHDLの標準仕様に組み込まれています．

またwriteでstd_logicやstd_logic_vectorを扱う場合には，std_logic_textioパッケージを呼び出す必要があります（**図4**）．

● プロシージャwriteline

writelineはline変数に格納された値を，標準出力，もしくはファイルに出力するプロシージャです．**図5**に