HDLによる設計法実践講座

VHDL編 5

FPGAによるサイモン・ゲーム機の実現(4)

シミュレーションの効率化

長谷川裕恭

シミュレーション結果のテキスト出力

シミュレーションでの機能検証は，波形出力によって行うのが基本です．最近のHDLシミュレータには，C言語などのソフトウェア開発で用いられるソース・コード・デバッギングの機能がついているものもあります．しかし， ソフトウェアは基本的に順番に実行されますが，ハードウェアは同時に複数の回路を動作させることができるので，実際にはあまり利用されていません．

VHDLのprocess文の中身は順番に実行されますが，複数のprocess文，同時処理文は同時に動作します．実際にソース・コード・デバッギングした場合，あるprocess文を1行ずつ解析していたつもりが，時間の移動とともにまったく異なるprocessの中身の実行に移動してしまい，何を実行していたのかわからなくなってしまうこともあります．

シミュレーションの初期段階では，文法的な誤りを解析するためにソースにブレーク・ポイントを設定し，そのときの値を観測することがあります．しかし，一度それなりに動作した後は，波形出力による解析が中心となります．誤った値を発見したときは，そのノードのドライバの値を調べていきます．

波形出力による機能検証が一段落すると，今度はテスト・パターンをテキストで出力する必要が生じます．HDL設計は，RTLでシミュレーション検証し，論理合成により回路を生成させれば終了というわけにはいきません．例えば，LSIを開発する場合では，LSIの出荷検査の際にテスト・パターンが必要となります．この場合，シミュレーション結果から出荷検査用テスト・パターンを作成します．

そのほかにも，シミュレーション結果をほかの環境へとインターフェースをとる必要が多々あります．インターフェースをとるためには，ほとんどの場合，テキストによる出力データを使用することになります．

今回は，PLDにて回路を作成するため，LSI出荷検査用のテスト・パターンは必要となりません．また，単なる実験的(趣味としての)設計であるため，ほかのシステムへのインターフェースは必要となりませんでした．

しかし，テキストでの出力は，外部とのインターフェース以外にも，シミュレーションの効率化という観点でも有効な手法となります．

シミュレーションによる検証は，一度全パターンを検証すれば終わりというわけにはいきません．開発終了直前まで何度も何度も仕様変更があったりします．仕様変更があるたびに，すべての項目を波形により確認するのはたいへんです．この場合，波形により一度慎重に検証した後は，そのシミュレーション結果をセーブし，その後は，セーブしたテスト・パターンとの一致性を検証すると効率化が図れます．

リスト5.1が，前号で紹介したSIMONのテスト・パターン記述にテキスト出力を追加したものです．テキスト出力をするためには，テキスト出力用のパッケージを呼び出す必要があります．

(1)では，STD.textioのパッケージとIEEE.std_logic_textioのパッケージを呼び出しています．std_logic_textioは，Synopsys社より提供されているパッケージです．

(2)では，出力するファイルをファイル宣言にて定義しています．方向は，outが出力，inが入力となります．

(4)のプロセス文では，実際に値を出力させています．値を出力する場合，まずヘッダ・ファイルを作成し，出力した値が何の値なのかを表示させる必要があります．

(5)では，ヘッダ・ファイルの文字列を，定数で定義しています．定義した文字列は，最初に(6)のwrite関数によって，ラインLoに書き込みます．1行に複数の値を出力させる場合は，write関数を再度実行することでラインLoに付け加えていきます．作成されたラインLoは，writeline関数でファイルに出力します．

今回は省略しましたが，通常，出力するテキスト・ファイルにはシミュレーション時刻も出力します．この場合，信号値の書き込みの前に，

```
write(Lo, NOW, LEFT, 6);
        -- NOWは，現在時間を表す関数
```

と書き込みます．

LEFTは，左詰めで表示させるという意味です．右詰めで表示させる場合は，RIGHTにします．通常，std_logicなどのビット・データを出力させる場合どちらでもたいした違いがありませんが，integerを出力する場合は幅が変化する

ので有用です.

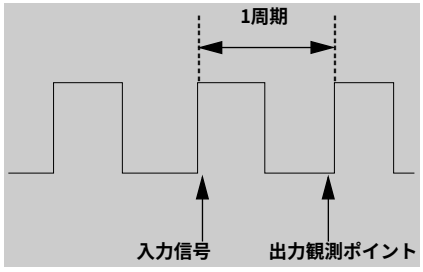
write関数の最後の数字は，表示させる幅を指定しています．実際に表示させるビット長よりも長い場合は，その分だけ空白を出力します．ビット長と同じかそれより短い場合は，そのビット長の分を出力します.

writeコマンドにより作成されたラインLoは，(7)のwritelineコマンドにより1cycleにつき1行ずつファイルに出力されていきます.

同期型ロジック回路のテスト・パターンでは，通常外部からの入力信号はクロックの立ち上がりエッジの直後に入力し，出力信号は，クロックの立ち上がりエッジの直前に観測します(**図**5.1).

今回の入力パターンは，HDLで作成しています．いっぽう，入力パターンをファイルから読み込むには，HDL記述の時と同じくクロックの立ち上がりエッジの直後に読み込んでください．出力データは⑻でクロック・エッジの立ち上がりエッジの直前に出力させています．HDL(RTL)シミュレーションでは，遅延時間はありません．したがって，RTLシミュレーションだけであればどのタイミングで書き込んでもかまいません.

しかし， RTLで使用したテスト・パターンは，ゲート・レベルでのシミュレーションでも流用します．ゲート・レベルでのシミュレーションでは遅延が発生します．したがって，ゲートでの遅延シミュレーションの検証に必要な時刻に書き込むようにしました.



〔図5.1〕信号の入力と観測

```
library IEEE,STD;                          ①テキスト出力のためのパッケージ呼び出し
use IEEE.std_logic_1164.all;
use STD.textio.all;
use IEEE.std_logic_textio.all;
entity simon_vt is end;
architecture SIM of simon_vt is                          ② ファイル宣言
constant cycle2bun : time := 50 ns;
constant cycle : time := 100 ns;
file Onv : TEXT is out "clock24pattern";
signal   CLK, SYSRESET, RESETSW, STARTSW, LEVELSW :
std_logic;
signal   KEYSW  : std_logic_vector(6 downto 1);
signal   EASY,NORMAL,HARD  : std_logic;

-----------------------------------------
省略
-----------------------------------------
file ov : TEXT is out "simon.ptn";

signal endflag : boolean;

begin

U1: SIMON  port map (  CLK, SYSRESET, RESETSW, STARTSW ,
LEVELSW,
          KEYSW, LED, EASY,NORMAL,HARD );

process begin
              CLK <= '1';
  wait for cycle2bun; CLK <= '0';
  wait for cycle2bun ;
  if(endflag) then wait; end if;
end process;

process begin
  endflag <= FALSE;
  setup(SYSRESET,RESETSW,STARTSW,LEVELSW,KEYSW);
  starton(STARTSW,LEVELSW);
  keyin(KEYSW,0);
  keyin(KEYSW,0);
  keyin(KEYSW,0);
  starton(STARTSW,LEVELSW);
  keyin(KEYSW,0);
  keyin(KEYSW,4);                          ③ endflagによる終了検出
  starton(STARTSW,LEVELSW);
  keyin(KEYSW,2);
  endflag <= TRUE;
  wait for cycle ;
  wait;
end process;

process                                    ④テキスト出力
constant HD1 : string := "CSRSLK     L     ENH";
constant HD2 : string := "LYETEE     E     AOA";
constant HD3 : string := "KSSAVY     D      SRR".
constant HD4 : string := " RERES      543210YMD";
constant HD5 : string := " ETTLW             A ";
constant HD6 : string := " SSSS654321          L ";
constant HD7 : string := " EWWW                 ";
constant HD8 : string := " T                    ";
variable Lo : line;                        ⑤ ヘッダ･ファイル用の
begin                                        定数宣言
  write(Lo,HD1,LEFT,21); writeline(ov,Lo);
  write(Lo,HD2,LEFT,21); writeline(ov,Lo);
  write(Lo,HD3,LEFT,21); writeline(ov,Lo);
  write(Lo,HD4,LEFT,21); writeline(ov,Lo);
  write(Lo,HD5,LEFT,21); writeline(ov,Lo);
  write(Lo,HD6,LEFT,21); writeline(ov,Lo);
  write(Lo,HD7,LEFT,21); writeline(ov,Lo);
  write(Lo,HD8,LEFT,21); writeline(ov,Lo);
  wait for cycle - 1 ns;
  while(endflag = FALSE ) loop             ⑥ ヘッダ･ファイルの書き込み
    write(Lo,CLK,LEFT,1);
    write(Lo,SYSRESET,LEFT,1);
    write(Lo,RESETSW, LEFT,1);
    write(Lo,STARTSW, LEFT,1);
    write(Lo,LEVELSW, LEFT,1);
    write(Lo,KEYSW,   LEFT,6);
    write(Lo,LED,     LEFT,6);
    write(Lo,EASY,    LEFT,1);
    write(Lo,NORMAL,  LEFT,1);
    write(Lo,HARD,    LEFT,1);               ⑦ ラインの書き込み
    writeline(ov,Lo);
    wait for cycle;
  end loop;
  wait;
end process;
end SIM;
configuration CFG_TEST of simon_vt is
 for SIM
 end for;
end;
```

〔リスト5.1〕SIMONのテスト・パターンにテキスト出力を追加したリスト