HDL設計

HDL設計ステップアップ講座（最終回）

テストベンチとprocedure

小林 優

筆者はVHDL設計の講習をよく行う．そこで気づくことは，多くの入門者にとってprocedure(Verilog-HDLのtask)は鬼門だということだ．筆者の説明が下手なのかもしれないが，プログラミングの経験の有無が背景にあるように思う．procedureは，プログラミング言語でいうところの「サブルーチン」である．ソフトウェア開発の経験がない方にはわかりにくい概念なのだろう．そこで今回は，Verilog-HDLのtaskより若干複雑なVHDLのprocedureについて解説する．（筆者）

1．テストベンチはプログラミング

●procedureやtaskがわからないのは

筆者が日頃行っているHDL講習では，procedureやtaskの演習を簡単にすませる方もいれば，どうしてもピンとこない方もいます．とくに，

- 「引き数」や「戻り値」の概念
- procedureの中に含めた遅延

などがわかりにくい項目のようです．

HDLによる回路記述は，セレクタやカウンタなどの記述例をまねすればなんとかなりますが，テストベンチはほとんどプログラミングの世界です．

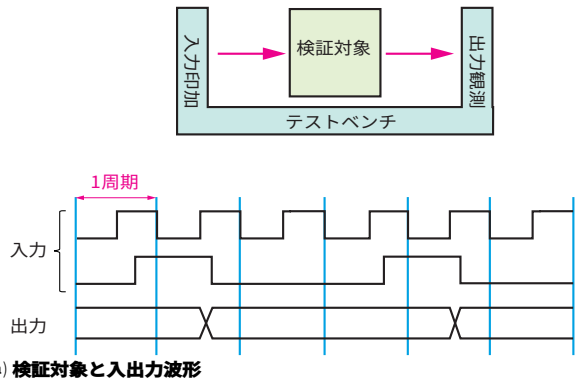
- 「変数」のように値を代入したり演算を行う信号
- if文やforループによる制御構造
- メインルーチンとしてのprocess文
- サブルーチンとしてのprocedure
- procedure呼び出しのネスティング

これらはすべて，デバッグ効率や可読性の高いテストベンチを記述するために使われます．つまり，「テストベンチはプログラミング」なのです．

●テストベンチの役割

回路を記述したのち，シミュレーションにより動作確認を行います．動作確認のために，入力を作り出力を観測する仕組みが**テストベンチ**です．**図1**(a)に示すように，テストベンチは検証対象に対し，上位階層から適切な入力を与え，出力を観測する仕組みになっています．通常1クロック周期単位で変化させた入力を与えます．

基本的なテストベンチのVHDL記述を**図1**(b)に示します．クロックは独立したprocess文で作成し，連続的に印加させます．検証対象への入力はクロックの1周期を単位に変化させて



(a) **検証対象と入出力波形**

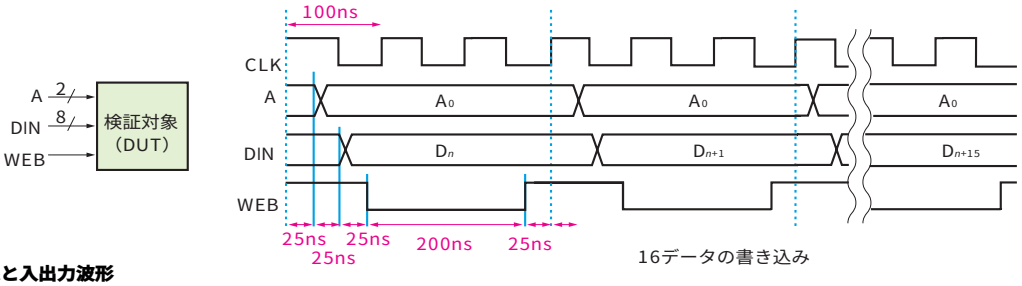
〔図1〕テストベンチの基本

```
architecture SIM of proc is
...
constant STEP: time := 100 ns;
begin
    U1: DUT port map(CLK, DIN, DOUT);

    ーークロックの作成
    process begin
        CLK = '0'; wait for STEP/2;
        CLK = '1'; wait for STEP/2;
    end process;

    ーー検証対象への入力作成
    process begin
                            DIN <= '0';
        wait for 10 ns;
        wait for STEP/2; DIN <= '1';
        wait for STEP;   DIN <= '0';
        wait for STEP*2; DIN <= '1';
        wait for STEP;   DIN <= '0';
        wait;
    end process;
    ...
end SIM;
```

(b) **テストベンチのVHDL記述**



(a) **検証対象と入出力波形**

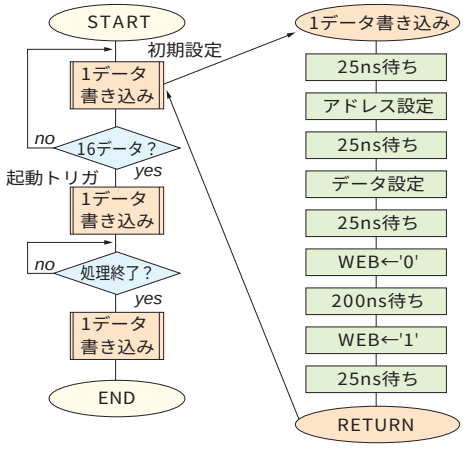
```
architecture SIM of proc is
...
constant DATA_REG:  std_logic_vector(1 downto 0):= "00";
signal    A: std_logic_vector(1 downto 0);
signal  DIN: std_logic_vector(7 downto 0);
signal  WEB: std_logic;

begin
    dut0: DUT port map(A, DIN, WEB);

    process begin
        for i in 0 to 15 loop
            wait for  25 ns; A   <= DATA_REG;
            wait for  25 ns; DIN <= DATA  ;
            wait for  25 ns; WEB <= '0';
            wait for 200 ns; WEB <= '1';
            wait for  25 ns;
        end loop;          1データ書き込みシーケンス
        wait;
    end process;
    ...
end SIM;
```

(b) **テストベンチのVHDL記述**

〔図2〕繰り返しはループ構造を使う



〔図3〕
テスト入力フロー

与えます．最初のステップでクロックの1周期に対し1/10程度の遅延の10nsを与えています．これはクロックの立ち上がりと同時に入力の変化が生じないようにするためです．このような同時変化の対策は，RTL検証時にも必要です．

検証対象の出力は，通常シミュレータに付属の波形観測機能やデバッガなどで確認します．したがって，テストベンチでは印加する入力を記述するだけでも検証は可能です．しかし，大量の出力の波形をながめても，それで期待どおりの動作を判定検証するのは不可能です．本格的な検証を行うには，あらかじめ用意した出力の期待値と自動比較して確認する仕組みなどが必要になります．これをVHDLで行うことができます．

●繰り返しの必要なテストベンチ

テスト入力によっては，数クロックを1シーケンスとして，複数のシーケンスを記述する場合があります．

CPUの周辺チップの検証を想定してみます（**図2**）．内部に複数のレジスタをもち，各種設定を行うためCPUから多数のデータを書き込むこととします．このとき，書き込みアドレスは固定で，連続書き込みできるものとします．**図2**(a)の波形に示すように，アドレスとデータを与え，書き込み信号WEBをアクティブ（'0'）にすることで1データの書き込みが行われます．これらの動作はクロックCLKの3周期で行われます．

この回路の動作設定のために，書き込みシーケンスを16回繰り返すこととします．この入力を実現するには，**図2**(b)のVHDL記述に示すようにループ構造を用います．forループを用いて16回の繰り返しを記述しています．クロックの3周期で1シーケンスとなるように，与えている遅延量の総和が3周期分の300nsになるようにします．

遅延量の総和がクロック周期の整数倍でないと，クロックとの関係が乱れてきますので要注意です．14個目までは正しく書けたけど，15個目以降がNGなんていうことも起こり得ます．

2. procedureの宣言と呼び出し

次に，この書き込みシーケンスが連続的だけでなく，ときどき必要になるようなテストベンチを想定してみます．

●1シーケンスをprocedure化する

前述の検証対象に対して，**図3**に示すように回路の初期設定と起動トリガなど，書き込みシーケンスを数回必要とする動作をシミュレーションしてみます．このフローの中で，「1データ書き込み」が繰り返し登場しますから，これをサブルーチン化します．このルーチンは，アドレス，データ，書き込み信号WEBを設定し，その前後に遅延を与えています．

このサブルーチン「1データ書き込み」をprocedureを使って記述してみます．基本となる記述は，**図2**のVHDL記述の中でアミをかけている部分です．**図2**では，アドレスAは固定値，データDINは別に宣言した配列から与えていましたが，procedure化するにあたって，「引き数」とすることとしました．