狙い通りの機能を実現するために

ロジック回路設計の手ほどき

菅原 孝幸
Takayuki Sugawara

第6回　最終確認のシミュレーションを行う

今回は論理合成を行い，実際のCPLD/FPGAの特性を含めたシミュレーションを行ってみましょう．シミュレーションの最終段階です．

Quartus Ⅱを使って，2006年4月号付録CPLD基板に搭載されたCPLDに回路を作り込むつもりで論理合成を行います．

論理合成後のデータを使ったシミュレーション結果は，実際にCPLDにハードウェアを書き込んで動作させたときにより近くなります．このシミュレーションもVeritak-CQ版(トランジスタ技術2006年8月号付録CD-ROMに収録)で行います．

連載第1回(2006年6月号)で，シミュレーションには論理合成前と論理合成後の二つがあると紹介しました．二つのシミュレーションの違いも解説します．

8ビットのカウンタ回路をHDLで記述する

● ハードウェア記述とテスト・ベンチを分ける

論理合成するハードウェア記述の中にはテスト・ベンチが混じっていてはいけません．ハードウェア記述とテスト・ベンチはそれぞれ別のHDLソース・ファイルにします．

ハードウェア記述は**リスト6-1**のモジュールcounterで，counter.vというファイルを作ります．

テスト・ベンチは**リスト6-2**のcounter_testというモジュールで，counter_test.vというファイルを作ります．これらは作業しやすいように同じフォルダに入れておきます．

```
module counter (input clock,reset,          //モジュールポートの記述、クロックとリセットが入力
         output reg [7:0] counter_out);     //モジュールポートの記述  counter_out がカウンタの出力

 always @(posedge clock,posedge reset) begin//クロックの立ち上がりで,またはresetの立ち上がりで
  if (reset) counter_out<=0;                //リセットがハイ(H)なら0にする
  else counter_out<=counter_out+1;          //リセットがロー(L)ならカウンタの値をインクリメント
 end

endmodule
```

リスト6-1　カウンタ回路のハードウェア記述
ファイル名counter.vとして保存する

Keyword 1　インスタンス化

インスタンス化とは，記述でしかないモジュールを実体のある存在にすることです．

例えば，**リスト6-1**のカウンタ回路を二つ呼び出したとします(**リスト6-A**)．これを論理合成すると，二つの8ビット・カウンタ回路が合成されます(**図6-A**)．カウンタの記述は一つですが，カウンタ回路は二つできるわけで

リスト6-A　同じモジュールを二つ呼び出す例

```
module w_counter;//二つのカウンタを呼び出した上位モジュール
    reg clock,reset;
    wire [7:0] counter_out1,counter_out2;
    counter  c1(.clock(clock),.reset(reset),.counter_out(counter_out1));//カウンタ1
    counter  c2(.clock(clock),.reset(reset),.counter_out(counter_out2));//カウンタ2
endmodule
```

リスト6-2　カウンタ回路のテスト・ベンチ
ファイル名 counter_test.v として保存する

```
`timescale 1ns/1ps
`define CYCLE (10)
module counter_test;//カウンタのテスト・ベンチ
        reg reset;//リセット信号
        reg clock=0;//クロック
        wire [7:0] counter_out;//カウンタ出力

        always #(`CYCLE) clock=~clock;//発振器の生成

        initial begin
          reset=1;
          #(10*`CYCLE);//10サイクルHの後、
          reset=0;//リセットをLにする
          #(1000*`CYCLE)                $finish;//1000サイクルでシミュレーション終わり
        end
                          counterというモジュールを組み込む
        counter  dut(.clock(clock),.reset(reset),.counter_out(counter_out));
endmodule
          サブモジュールとしての名前   テスト・ベンチの中での信号名
```

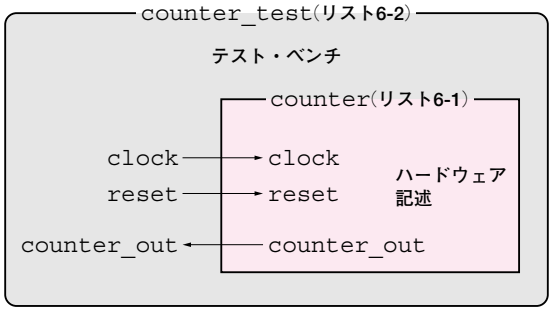

図6-1　テスト・ベンチのモジュールの中にハードウェア記述のモジュールがある
このような構成にするとハードウェア記述にテスト・ベンチがまざらない

● テスト・ベンチからハードウェア記述を呼び出す

テスト・ベンチのモジュール counter_test の中で，ハードウェア記述のモジュール counter を呼び出し，ハードウェア記述とテスト・ベンチの信号を接続しています(**図6-1**)．

このように，モジュールの中に他のモジュールを組み込むことをインスタンス化と呼んでいます．

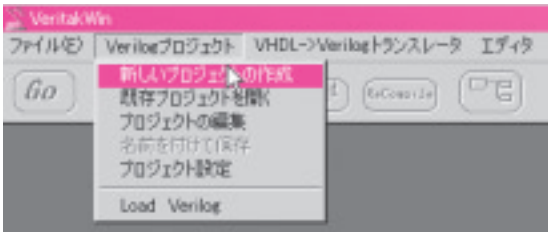

図6-2　複数のHDLソースをまとめて扱うプロジェクトを作成する
［Verilog プロジェクト］－［新しいプロジェクト］を選ぶ

作成したカウンタの機能を確認する

● 複数のHDLソースをまとめるプロジェクト

設計の規模が大きくなってくると，ファイルの数も増えてきます．Veritak では，一つのプロジェクト・ファイルを呼び出すことで，複数のファイルをまとめてコンパイルできるようになっています．

プロジェクト・ファイルを作成してみましょう．

Veritak のメニューから［Verilog プロジェクト］-

Keyword 1　**インスタンス化**（つづき）

す．.counter_out（counter_out1）という記述は，呼び出されてインスタンス化されるモジュール（counter）のポートにある信号 counter_out を上位モジュールにある信号 counter_out1 に接続する，という意味です．

このように，任意の数だけモジュールの階層を重ねていくことができますが，HDLの場合のインスタンス化は，最終的に物理的な論理回路になるので現実味があります．

作りたいロジック回路が使用予定のFPGAに収まるかどうかは，設計の最終段階で，最上位階層の論理合成後に初めてわかります．

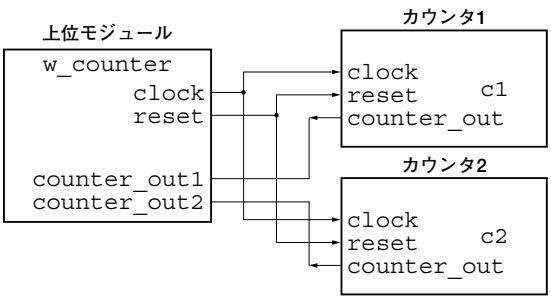

図6-A　リスト6-Aの配線