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2回　HDLシミュレータの動作

前回(2006年6月号)，HDL記述はハードウェア記述とテストベンチ記述に分かれる，と書きました．

しかし，HDLそのものには，ハードウェアとテストベンチという分類はありません．

HDLシミュレータは，実際にハードウェアになる記述か，それともテスト用の仮想的な記述かは，通常は認識しません．HDLシミュレータは，言語規格に則って，記述を順番に実行するだけなのです．

● HDLシミュレータの動作原理を知っていると効率良く設計できる

HDLシミュレータは，実際の設計場面で，もっとも頻繁に利用するツールです．

これからHDLで設計される人は，デバッグ時に多くのエラーに悩まされると思います．その中には，HDLシミュレータの内部の動きを知ることで，解決できることも少なくないと思います．

シミュレータをブラック・ボックスとして考えてしまうと，シミュレータの結果と期待値が違ったとき，手がかりがないので原因を見つけるのに苦労します．

シミュレータの内部動作を知り，何が行われるか理解しておくことは，デバッグ効率を高めます．また，トラブルを未然に防ぐことにも通じるかと思います．

● HDLシミュレータは二つの動作をもつ

HDLシミュレータは大きく分けて次の二つの動作をもっています．

①プログラムを実行するソフトウェア

②ネットリストを解析する回路シミュレータ

①はソフトウェアのような手続き言語で動作を記述します．②はネットリストによる記述です．

言い換えると，HDL記述の中には①の記述と②の記述の両方があるということになります．

ただし，HDLで記述されるのはハードウェアです．ソフトウェアのような記述でも，ハードウェアに変換されることが前提です．

ハードウェアは複数の回路が同時に動く

HDLで実際に実現するデバイスは，ハードウェアです．現実のハードウェアは，同じタイミングで多くの回路が動作しています．

ところがHDLシミュレータは，CやC++で書かれたソフトウェア・プログラムなので，一度に一つの処理しか実行できません．

そこで，HDLシミュレータでは，時間的に並行して動作するHDL記述を，プロセスという細かい実行単位に分けて解析します．

Keyword 1　サイクル・ベース・シミュレータ

サイクル・ベース・シミュレータは，イベント・ドリブンとはアルゴリズムが異なります．イベント・ドリブンでは，変化のあったところだけを演算するのですが，サイクル・ベースでは，同期化したクロック・エッジでのみ，変化の有無に係わらず毎回演算します(図2-A)．基本的には，非同期回路はシミュレーションできず，同期回路しかシミュレーションできません．

そのかわり，イベント・ドリブン・ベースのシミュレータと比較して数倍以上速くシミュレーションできます．それは，イベント・ドリブンのような動的なイベント管理をなくしているからです．フリーのサイクル・ベース・シミュレータとしては，Verilator(http://www.veripool.com/verilator.html)があります．Verilogソースを2値化し，C++ソースに変換するトランスレータになっています．

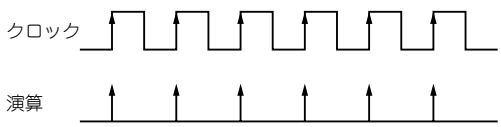

図2-A　サイクル・ベース・シミュレータの動作イメージ

リスト2−1　二つのプロセスをもつHDL

```
module test1;
  integer i;
  reg a,b;


  initial begin                          プロセス1

    $display("プロセス1：HDLシミュレーションは、プログラム言語のように順番に実行されます。");
    $display("プロセス1：サブルーチンを3回呼び出してみましょう。");
    for (i=0;i< 3;i=i+1) begin
      sub_routine;//サブルーチンの呼び出し
    end

    $display("プロセス1：サブルーチンの呼び出しが終わりました。現在の時刻は=%3dです。¥n",$time);

  end

  initial begin                          プロセス2

    $display("プロセス2：HDLシミュレーションは、プログラム言語のように順番に実行されます。");
    $display("プロセス2：サブルーチンを3回呼び出してみましょう。");

    for (i=0;i< 3;i=i+1) begin
      sub_routine;//サブルーチンの呼び出し
    end

    $display("プロセス2：サブルーチンの呼び出しが終わりました。現在の時刻は=%3dです。¥n",$time);
  end


  task sub_routine;                      ┐
    begin                                │
      $display("    サブルーチンが呼び出されました。");  ├ サブルーチン
    end                                  │
  endtask                                ┘
endmodule
```

順番に処理した複数のプロセスを，現実の回路では同時に実行されたとみなすことで，同時に複数の回路が動作することをシミュレーションしています．

現実の回路で時間差がある部分には，必ずディレイ文が挿入され，そこで実行順が変わります．

● 二つのプロセスがあるHDLを動かしてみる

リスト2−1は，initial　beginからendまでという二つのプロセスをもつHDL記述です．

実行結果を**リスト2−2**に示します．

これを見ると，プロセス1を始めから終わりまで実行したあとに，プロセス2を始めから終わりまで実行していることがわかります．しかし，この二つのプロセスは，同時に実行されたとみなされています．時刻が0なのがその証拠です．

このプロセス中にディレイ文を挿入してみます．

● HDLシミュレータはディレイ文を見つけるとプロセス実行順を変える

リスト2−3(p.198)は，サブルーチン呼び出しの後

Keyword 2　同期回路/非同期回路

同期回路とは，クロック・エッジで一斉に動く回路です(**図2−B**)．CPLD/FPGAでは，同期回路で設計するのが基本になります．

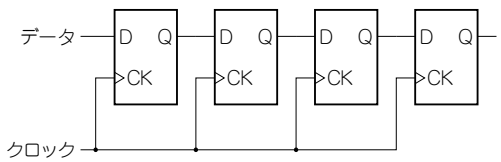

図2−B　同期回路の例(4ビットのシフトレジスタ)

非同期回路とは，共通のクロックをもたずに動く回路のことです．非同期回路の例として，**図2−C**のようなリプル・カウンタがあります．HDLを使った設計ではほとんど使われることはありません．

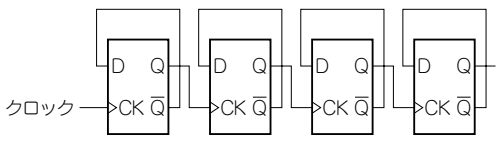

図2−C　非同期回路の例(4ビットのリプル・カウンタ)