HDLによる設計法実践講座 VHDL編 2

FPGAによるサイモン・ゲーム機の実現(1)

分周器，キー入力，間隔カウンタのVHDL記述

長谷川裕恭

Appendixで解説しているサイモンの概要にしたがって具体的に各ブロックをRTL記述していきます．大規模な設計では，上位の記述(ビヘービァ・レベル)でシステム全体の検証を行ってから具体的な記述に入っていくこともありますが，2～3万ゲート以下のあまり規模の大きくない設計ではいきなりRTLで記述します．

各ブロックはブロックごとに検証を行い，その後システム全体の検証を行います．ただし，今回設計するサイモンはゲート数(2NAND換算)が1700程度の規模の小さい設計なので，HDLに慣れた人ならば1日か2日で記述を書き終えてしまえます．この規模であれば一つ一つのブロックに対していちいちシミュレーションするようなことはせず，全体を記述し終えてからシミュレーションするほうが効率的です．

今回は，まずサイモンの回路設計のうち分周器(DIVIDER)，キー入力部(KEYIN)，間隔カウンタ(INTVAL)のVHDL記述を解説します．

分周回路の記述(DIVIDER)

このブロックでは32kHzの外部クロックを512Hzに分周し，32Hzのイネーブル信号を作り出しています．**リスト**2.1が分周器のVHDLの記述です．(1)が分周用10ビット・カウンタの記述で，同期式バイナリ・カウンタです．(2)のSYSRESETが'0'の時，CNTの値は0に確定し，カウントを開始します．

SYSRESETの信号が入力されないとCNTの値は確定しないのでシミュレーションができなくなります．逆に言うとSYSRESETはシミュレーションでカウント値を確定させるためだけの信号です．実際に基板上で配線する際は，'1'で固定します．

◎リプル・カウンタを使用する方法

分周器であれば**図**2.1，**リスト**2.2のように非同期式リプル・カウンタを使用する方法もあります．

リプル・カウンタは下位ビットの出力を次段のフリップフロップのクロック信号に使用するので上位ビットになればなるほど入力クロックに対する遅延が蓄積されてしまいます．大規模なロジック回路設計では，この遅延を考慮するのが難しく，使用するのはあまり好ましくありません．しかし，分周器の場合は最終段の出力のみクロックとして使用するため，この遅延は問題になりません．

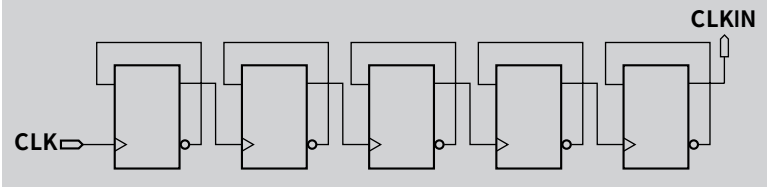
ゲートアレイやASIC設計では，分周器であれば非同期式リプル・カウンタを使用してロジック面積を減らすべきです．

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
entity DIVIDER is
   port  (CLK, SYSRESET : in std_logic;
          CLKIN, HZ32   : out std_logic);
end DIVIDER;
architecture RTL of DIVIDER is
signal CNT : std_logic_vector(9 downto 0);
begin

CLKIN <= not CNT(5);
HZ32  <= CNT(9) and CNT(8) and CNT(7) and CNT(6);

process (CLK,SYSRESET) begin
   if(SYSRESET='0') then
      CNT <= ( others => '0' );
   elsif(CLK'event and CLK='1') then
      CNT <= CNT + 1;
   end if;
end process;

end RTL;
```

(2)非同期リセットの記述
シミュレーションのためだけに必要

(1)同期式カウンタの記述

〔リスト2.1〕分周器(DIVIDER)のVHDL記述



〔図2.1〕リプル・カウンタ

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
entity DIVIDER is
   port  (CLK, SYSRESET : in std_logic;
          CLKIN, HZ32   : out std_logic);
end DIVIDER;
architecture RTL of DIVIDER is
signal CNT : std_logic_vector(3 downto 0);
signal CLK2,CLK3,CLK4,CLK5,CLK6,CLK7 :
std_logic;
begin

CLKIN <= CLK7;
HZ32  <= CNT(3) and CNT(2) and CNT(1) and
CNT(0);

process( CLK ) begin
   if(CLKÕevent and CLK='1') then
      CLK2 <= not CLK2;
   end if;
end process;

process( CLK2 ) begin
   if(CLK2Õevent and CLK2='1') then
      CLK3 <= not CLK3;
   end if;
end process;

process( CLK3 ) begin
   if(CLK3Õevent and CLK3='1') then
      CLK4 <= not CLK4;
   end if;
end process;
process( CLK4) begin
   if(CLK4Õevent and CLK4='1') then
      CLK5 <= not CLK5;
   end if;
end process;
process( CLK5) begin
   if(CLK5Õevent and CLK5='1') then
      CLK6 <= not CLK6;
   end if;
end process;
process( CLK6) begin
   if(CLK6Õevent and CLK6='1') then
      CLK7 <= not CLK7;
   end if;
end process;
process (CLK7,SYSRESET) begin
   if(SYSRESET='1') then
      CNT <= ( others => '0' );
   elsif(CLK7Õevent and CLK7='1') then
      CNT <= CNT + 1;
   end if;
end process;
end RTL;
```

〔リスト2.2〕リプル・カウンタのVHDL記述

しかし，FPGA/PLDではデバイスの構成上あまり有利になりません．

また，**リスト**2.1と**リスト**2.2を見くらべればわかるように，HDLでは同期式のほうが明らかに解りやすく記述できるので，今回は同期式で分周することにしました．

◎システム・クロック

サイモンの各ブロックには，DIVIDERで分周した512HzのCLKINを単一のシステム・クロックとして供給します．外部から供給される32kHzのクロックは十分に遅い周期なので，システム・クロックとして使用しても回路の遅延値はまったく考慮せずに設計できます．

今回は秒単位の非常に長いシミュレーションが必要となるので，シミュレーションの容易さを考え，512Hzに分割することにしました．

次号で解説する乱数発生部(RANDOM)を除けば，さらに遅いクロックのほうがシミュレーションしやすく，消費電力も少なくなります．しかし，乱数発生部では時系列によって720パターンを発生させ，STARTONを押すタイミングによって値を確定するので，あまり遅いクロックにするとランダム性が失われてしまいます．

したがって，ここではランダム性とシミュレーションしやすさの兼ねあいで512Hzのクロックにしています．

キー入力部(KEYIN)

キー入力部では，チャタリング防止とキー入力の立ち上がり検出を行います．

◎チャタリング防止

リスト2.3の(1)では，STARTSW, LEVELSW, RESETSWのキー入力信号をS，L，Rの配列の0ビット目にセットします．

その後，**リスト**2.3の(2)のfor-loop文で2段シフトさせ，合計で3段のシフトレジスタを構成しています．

このシフトレジスタは，**リスト**2.4のようにprocess文の外側でまず配列の0ビット目に代入し，for-loop文で3段のシフトレジスタとして記述したいところですが，これではシミュレーションでエラーになってしまいます．

なぜかというと，VHDLの文法仕様では，一つのprocess文で出力ドライバが一つ生成されることになっているからです．

S，L，Rの0ビット目は実際にはfor-loop文で代入していませんが，S，L，Rというひとかたまりの信号と判断されてしまいます．その結果，process文ではS，L，Rに代入が行われず，常に初期値'U'が出力されます．この'U'と**リスト**2.4の(1)の出力が衝突することになり，値が確定しなくなってしまうわけです．

リスト2.3のfor-loop文は2段シフトさせているだけなので，実際にはfor-loop文を使用するメリットはあまりありません．for-loop文を使用せずに単純に代入文で記述してもよいでしょう．

HZ32は分周器で作成した32Hz(31ms周期)のパルス信号です．この周期でシフトすることによってチャタリングを防止しています．

キーを押した際，あるいは離した際に生じるチャタリングの長さはせいぜい20msぐらいなので，この周期で入力信号をシフトすればまず大丈夫です．