

HDLによる 設計法 実践講座

Verilog-HDL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Verilog-HDL記述

小林 優

Appendixで紹介しているサイモンの記述をこれから数回にわたって解説します。まずAppendixをご覧ください。

今回はサイモンの基本ブロック中の分周器(divider), キー入力部(keyin), 間隔カウンタ(intval)のVerilog-HDL記述を解説します。また, 本機で用いる以外の回路や記述スタイルも紹介します。

分周器 (divider)

基本クロック32kHzを分周して, 回路全体を駆動するシステム・クロック512Hzを得る回路です。

入出力概要

分周回路のVerilog-HDL記述をリスト2.1に示します。同期カウンタですので, 簡潔な記述となっています。入力は, ・CLK(32kHz)

```
module divider ( CLK, SYSRESET, clkin, hz32 );
input          CLK, SYSRESET;
output         clkin, hz32;
reg            [9:0] cnt;

always @( posedge CLK )
    if( SYSRESET )
        cnt <= 10'b0;
    else
        cnt <= cnt + 10'b1;

assign clkin = ~cnt[5];
assign hz32  = (&cnt[9:6]);

endmodule
```

〔リスト2.1〕分周器(divider)のVerilog-HDL記述

・SYSRESET(分周器専用のリセット入力)

です。いっぽう, 出力は,

・clkin(システム・クロック, 512Hz)
・hz32(スイッチのチャタリング除去用)

となっています。

カウンタの本体は10ビットのレジスタ,
reg [9:0] cnt;

です。クロックCLKの立ち上がりで動作し, SYSRESET入力があればリセットし, なければカウント・アップします。

出力のシステム・クロックclkinは, 基本クロックを64分周した信号ですから, カウンタの6ビット目のcnt[5]をそのまま用います。ただし, リセット時の初期値を'1'の波形にしたかったため, cnt[5]の反転信号を用いています。

もう一つの出力hz32は, 32Hzの信号です。システム・クロックclkinの1周期分だけ'1'とするため, レジスタcntの上

位4ビットcnt[9:6]を用いて作成します。

同期式と非同期式の分周回路

リスト2.1の分周器の記述は単純ですが, 論理合成によって同期式カウンタになります。図2.1に示すように10ビットのフリップフロップをベースにインクリメンタ, セレクタなどから構成されています。カウンタとしては大規模です。

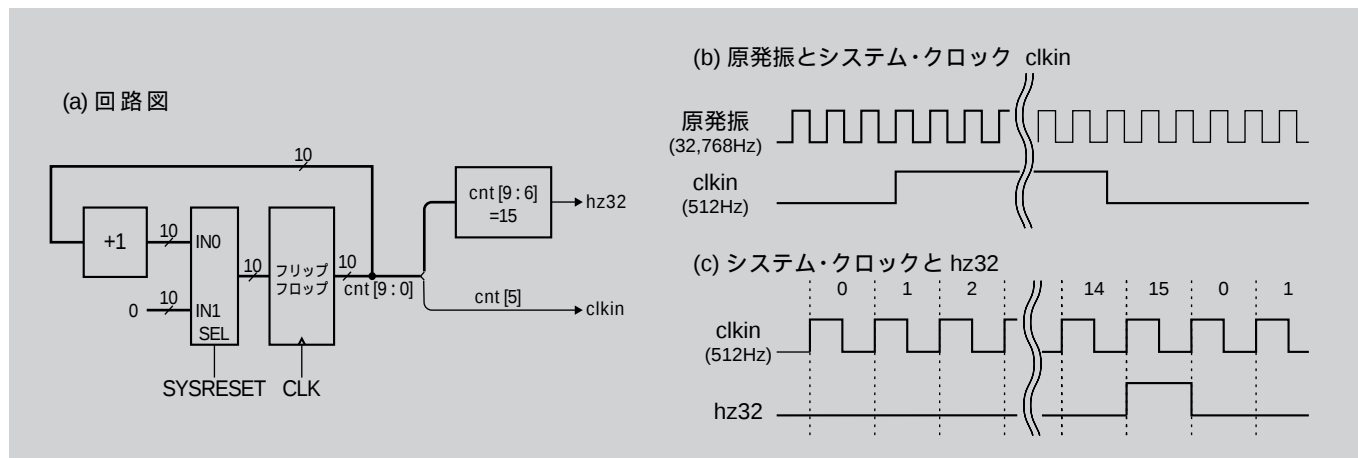
一方, 従来からの非同期式分周回路は, 図2.2に示すように, トグル式のカウンタを並べたものです。前段の出力を後段のクロックに入力し, クロックの立ち下がりで出力が反転します。リセット入力は, 全ビットに入力します。リセット動作はクロックに対して非同期動作です。

この回路を記述したものがリスト2.2です。同期式のリスト2.1に比べると量も複雑さも増しています。カウンタを構成する個々のTフリップフロップは,
reg d0, d1, ..., d9;
のように1ビットずつレジスタ宣言します。そして, 各々のTフリップフロップごとにalways文を記述します。

非同期式の利点

非同期式のカウンタの利点は, まず第一に回路規模が小さくなることです。同期式ではインクリメンタやセレクタの分, どうしても規模が大きくなります。

第二に消費電流です。低消費電流が特徴のCMOSロジック回路でも, 回路が頻繁に動作していれば消費電流も大きくなります。電池で動作する携帯機器では, 待機時の動作周波数を低速化するなどして低消費電流化を図っています。



〔図2.1〕分周器(divider)の概要

```

module divider_async ( CLK, SYSRESET, clkkin, hz32 );
input    CLK,  SYSRESET;
output   clkkin, hz32;
reg      d0, d1, d2, d3, d4, d5, d6, d7, d8, d9;

always @( posedge d8 or posedge SYSRESET )
  if( SYSRESET )
    d9 <= 1'b0;
  else
    d9 <= ~d9;

always @( posedge d7 or posedge SYSRESET )
  if( SYSRESET )
    d8 <= 1'b0;
  else
    d8 <= ~d8;
...

```

```

always @( posedge CLK or posedge SYSRESET )
  if( SYSRESET )
    d0 <= 1'b0;
  else
    d0 <= ~d0;

assign clkkin = ~d5;
assign hz32   = d9;

endmodule

```

〔リスト2.2〕非同期式分周器のVerilog-HDL記述(抜粋)

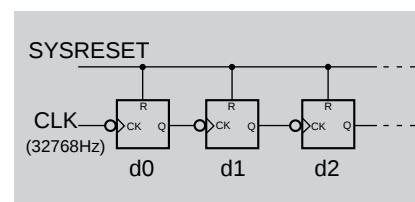
本回路の場合、大きな差はないと思いますが、システム・クロックの512Hzを作り出す際、基本クロックの32kHzで動作する部分が少ないほうが有利です。

同期式の利点

いっぽう、同期式カウンタの利点は、回路記述のシンプルさとタイミング設計

の容易さです。これは大規模な回路を設計する時には最大の利点になります。

非同期式回路を含んだ回路では、わずか数100ゲート規模でもタイミング設計に苦勞し、ミスを犯すことがあります。また、現状の論理合成ツールが同期設計向きであることを考慮すると同期設計が有利です。したがってサイモンは同期式



〔図2.2〕非同期式回路で分周した場合

同期か非同期か、それが問題だ

筆者は、HDL設計の伝道者(?)であると同時に同期設計の信奉者でもあります。

完全非同期設計(!)の中で育った筆者は、画像処理関連の設計に関わるようになって初めて同期設計を知りました。クロック・ライン設計の苦勞から開放された感激で、ことあるごとに同期設計のメリットを語ってきました。HDL設計に手

を染めてから、さらに拍車がかかったことは言うまでもありません。

しかし最近、携帯情報機器開発に関わるようになって状況が一変しました。電池寿命と高速処理という相反する条件を満たすためには、少しでも消費電流を減らすため、非同期式回路を使用せざるを得ないのです。実際どれだけ少ないのか、

具体的数値があるわけではありません。しかし、あえて危険を冒せないのが、商品開発に関わる設計者の悲しい性です。

非同期設計の記述は、リスト2.2のように、クロックの本数が多く、とてもわかりにくいものです。いっそのこと回路図のほうが理解しやすいのでは、と思うことさえあります。非同期設計の回路を、いかに理解しやすくHDLで表現するか。これが筆者の次なるテーマです。