FPGAでシリアルATAコントローラを設計する

第4回 シリアルATAコアとシミュレーション

菅原 博英

前回までに設計したシリアルATAコントローラのコア（SATA_CORE）のシミュレーションを行う．
（編集部）

1．コア全体シミュレーション

コードの解説を読むだけでは，SATAの動作を理解するのはなかなか難しいと思います．今回はRTL設計に伴うシミュレーション環境を設計します．

このうちのリンク層の確認を実行させます．まずPHYの初期化完了後，最初に送信されるRegD2Hを試してみます．最も簡単な制御FIS（Frame Information Structure）なので，リンク層のHOLD/HOLDAによるフロー制御は発生しません．

● テスト・コード

テスト全体のファイル構成を**リスト1**に，test_top配下にある全体の概要を**図1**に示します．xxx_cmd_inc.vはテスト動作をなるべく抽象的に記述したタスク群です．これらが，xxx_internal_task_inc.vを経由してinternal_core_itf.v内の内部バス動作を実行させます．

(1) xxx_cmd_inc.v

DEV_INITのタスクを**リスト2**に示します．

DEV_INITは，デバイス側のSATA_COREへレジス

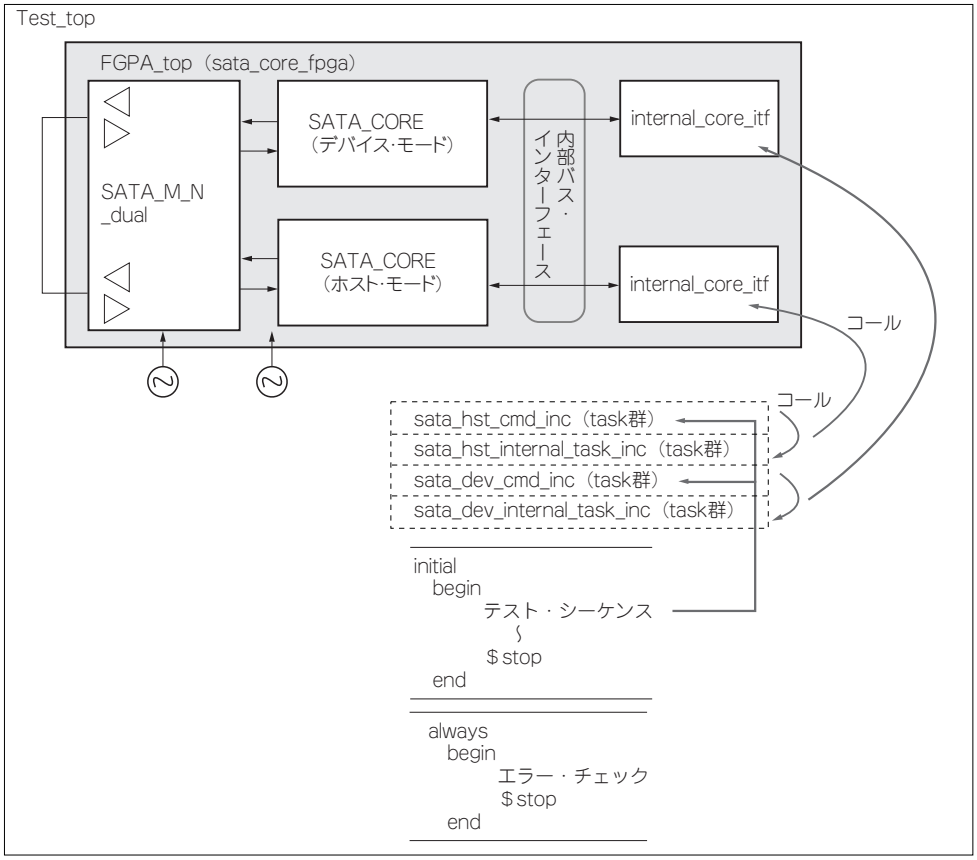

図1
test_top配下の全体概要

KEYWORD ── シリアルATAコア，リンク層，FIS転送，8B/10B変換，スクランブラ

リスト1
テスト全体の
ファイル構成

```
\verilog\test\sata_core\test_top.v                              :最上位モジュール
 +-- \verilog\test\sata_core\sata_core_ports_inc.v              :ポート定義ファイル
 |    +- \verilog\rtl\fpga_top\sata_core_fpga.v                 :FPGAのトップ
 |        +-- \verilog\rtl\bhv_test\internal_core_itf.v         :各サブモジュール
 |        +-- \verilog\rtl\sata_core\全部
 |        +-- \verilog\core\bhv\SATA_PHY_dual_bhv.v
 |        +-- \verilog\core\bhv\DCM_CLK_bhv.v
 |        +-- \verilog\core\bhv\RAM32bX64w_bhv.v
 +-- \verilog\test\sata_core\sata_core_clock_inc.v    :クロック定義ファイル
 +-- \verilog\rtl/bhv_test/sata_task_reg_inc.v"       :パラメータ定義
 +-- \verilog\rtl/bhv_test/sata_hst_cmd_inc.v"        :各ドライバ
 +--  \verilog\rtl/bhv_test/sata_dev_cmd_inc.v"
 +-- \verilog\rtl/bhv_test/sata_hst_internal_task_inc.v"
 +-- \verilog\rtl/bhv_test/sata_dev_internal_task_inc.v"
```

タ・アクセスを実行するDEV_REG_RとDEV_REG_Wを用います．最初にStatusレジスタを確認します．PHYがReadyになるまでは7Fhが読めるので，それを繰り返し，80hが読めたら各レジスタを初期化します．最後にDEV_REGDHを呼び出してRegD2H送信を起動します．

(2) xxx_internal_task_inc.v

sata_dev_～.v内にDEV_REG_Rのタスクがあります．それを**リスト3**に示します．

上位からのアドレス(adr)をTRNS_TFRの要求するアドレス(IOadr)に変換して，内部バス動作(DEV_REG_R)を実行させています．完了後，internal_core_itf.vでリード値(reg_rd)の再代入を行います(dev_reg_rd)．

● internal_core_itf.v

ITF_REG_Rのソースを**リスト4**に示します．

リスト2　DEV_INITタスク

```
task DEV_INIT;
//-----------------------------------------------------
   begin
    //- - - - - - - - - - - - - - - - - - - - - - -
    DEV_REG_R(StatusReg)              ;
//  while(dev_reg_rd==16'h007F) begin
    while((dev_reg_rd==16'h007F)
         |(dev_reg_rd==16'h0000)) begin
        DEV_REG_R(StatusReg)      ;
    end
    //- - - - - - - - - - - - - - - - - - - - - - - - -
    DEV_REG_W(FeatureReg     ,16'h0000)     ;
    DEV_REG_W(ErrorReg       ,16'h0000)     ;
    DEV_REG_W(StatusReg      ,16'h0050)     ;
    DEV_REG_W(DevHeadReg     ,16'h0000)     ;
    //- - - - - - - - - - - - - - - - - - - - - - - - -
    DEV_REG_W(CylHighReg     ,16'h0012)     ;
    DEV_REG_W(CylLowReg      ,16'h0034)     ;
    DEV_REG_W(SctNumReg      ,16'h0056)     ;
    DEV_REG_W(SctCntReg      ,16'h0078)     ;
    DEV_REG_W(CylHighReg     ,16'h009A)     ;
    DEV_REG_W(CylLowReg      ,16'h00BC)     ;
    DEV_REG_W(SctNumReg      ,16'h00DE)     ;
    DEV_REG_W(SctCntReg      ,16'h00F0)     ;
    //- - - - - - - - - - - - - - - - - - - - - - - - -
    DEV_REGDH                ;
    //- - - - - - - - - - - - - - - - - - - - - - - - -
   end
endtask
```

アドレス(adr=DEV_REG_RではIOadr)をO_CMD_xxxとして出力し，アクセス待ち(I_CMD_NRDY)がないことを確認します．ただし，DATAレジスタへのアクセス以外はI_CMD_NRDYは常にネゲートです．

出力信号の更新は，@(posedge CLK)でクロックに同期させつつ，少し遅延させて(#1)行います．一方の入力信号は，@(negedge CLK)や遅延(#1)などを用いて，安定した時点で確認します(#1だけの文はエラーになることがあるので，dummyへ代入)．ITF_REG_Rではリードした値を変数(reg_rd)へ取り込みます．

最後に，アドレスなどを0に戻して終了です．O_CMD_REとO_CMD_WE以外はそのままでもレジスタ・バス仕様上は大丈夫ですが，シミュレーション波形を見やすくするために0へ戻します．

● リンク上位層確認

シミュレーションに必要なモジュールをコンパイルし，波形ファイルを開いて20μs程度実行すると，最初のRegD2Hの送信が完了します．

リスト3　DEV_REG_Rタスク

```
task DEV_REG_R;
//-----------------------------------------------------
input    [6:0]     adr  ;
reg [6:0]      IOadr     ;
   begin
    //- - - - - - - - - - - - - - - - - - - - - - -
    casex({adr[6:3]})
    4'b0010:IOadr={3'b01_1,1'b0,adr[2:0]};//Normal
    4'b1011:IOadr={3'b01_0,1'b0,adr[2:0]};//Read to Wonly
    4'b1010:IOadr={3'b01_1,1'b0,adr[2:0]};//Read to Ronly
    4'b0100:IOadr={3'b10_1,1'b1,adr[2:0]};//Contol
    4'b1101:IOadr={3'b10_0,1'b1,adr[2:0]};//Read to Wonly
    4'b1100:IOadr={3'b10_1,1'b1,adr[2:0]};//Read to Ronly
    4'bx11x:IOadr={3'b11_0,     adr[3:0]};//Other Control
    endcase
    //- - - - - - - - - - - - - - - - - - - - - - - - -
    `FPGA_TOP.dev_sml.ITF_REG_R(IOadr) ;
    dev_reg_rd =`FPGA_TOP.dev_sml.reg_rd    ;
    //- - - - - - - - - - - - - - - - - - - - - - - - -
   end
endtask
```