CertusPro-NX
G8B10B Serdes 4Byte Mode
リファレンスデザインユーザーマニュアル

June-2022

免責事項

本ドキュメントに含まれる情報、及び本ドキュメントの対象であるリファレンスデザインの内容、動作、特性、品質に対して、マクニカはいかなる保証も行いません。また、本ドキュメントに含まれる情報、及び本ドキュメントの対象であるリファレンスデザインは全て現状有姿にて提供され、これに対する改版や技術サポートのご依頼に関しては理由の如何を問わずお控え頂くようお願いしております。お客様ご用途における使用可否の判断、使用の際の動作確認、お客様製品への実装における適合性や安全性の確認、法的要件の確認はお客様にて実施頂きますようお願いいたします。これらに対してもマクニカは一切の責任を負うことが難しく、いかなる保証もいたしかねます。また、本ドキュメントの情報、及びドキュメントの対象であるリファレンスデザインはマクニカの所有物であり、予告なしに変更を加えることがございますので予めご了承ください。

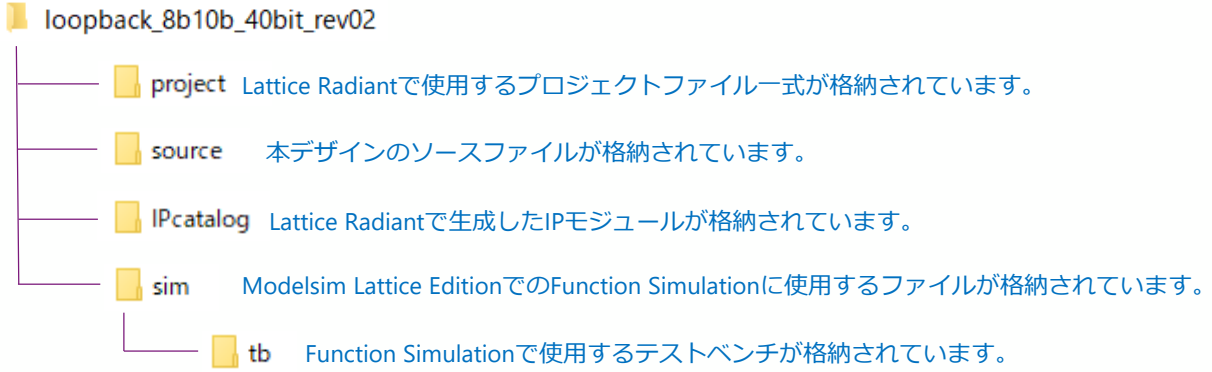
Table of Contents

1. デザイン概要

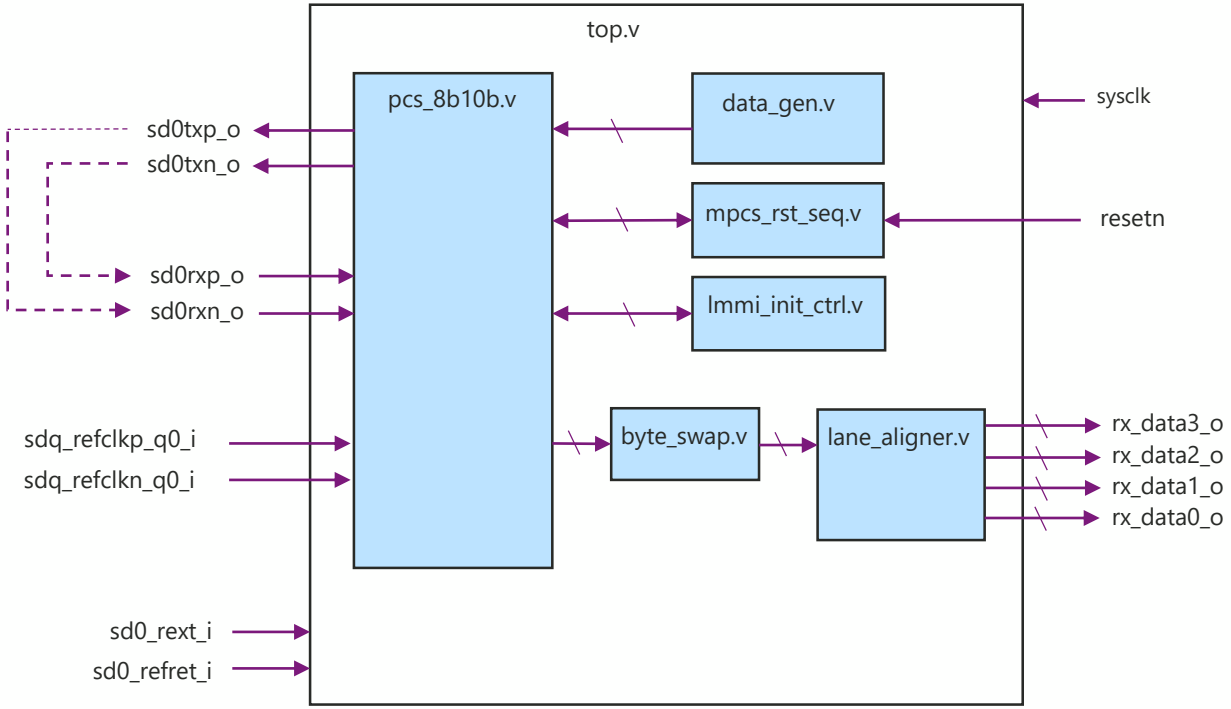
- **本デザインはGeneric 8B10Bプロトコルを使用したCertusPro-NXのSerdesループバックリファレンスデザインです。**
- **簡易的なデータジェネレータにて32bitデータを生成し、Generic 8B10B 4Byte Modeで出力してループバックします。**
  1Byte Mode, 2Byte Mode, 4Byte Modeの詳細は以下のテクニカルノートを参照してください。
  https://www.latticesemi.com/view_document?document_id=53257
- **8B10B符号化に用いるComma CharacterはK28.5のコードを使用しています。**
- **MPCSリファレンスクロック周波数は156.25MHz、シリアルデータレートは3.125Gbpsに設定されています。また、リファレンスクロック入力はQuad内チャネル共用の専用クロックSDQ_REFCLKを使用しています。**
  リファレンスクロックの詳細は以下のテクニカルノートを参照してください。
  https://www.latticesemi.com/view_document?document_id=53257
- **デザイン動作はModelSim Lattice Editionでのファンクションシミュレーションにて確認しています。実機上での確認はしておりません。**
- **本デザインはRadiant3.1.1でコンパイルされており、論理合成ツールはSynplify Proを使用しています。**
- **MPCS IPはVersion 1.3.0を使用しています。**

2. デザインフォルダ構成

本デザインのフォルダ構成を以下に示します。



3. 回路ブロック図



4. デザインポート説明

ポート名	入出力方向	説明
sysclk	入力	システムクロック入力。本デザインではリセットシーケンサやMPCSの制御クロックとして使用
resetn	入力	リセット入力（Active-Low）
sdq0_refclkp_i	入力	MPCSリファレンスクロック（ポジティブ）
sdq0_refclkn_i	入力	MPCSリファレンスクロック（ネガティブ）
sd0rxp_i	入力	Serdesシリアルデータ入力（ポジティブ）
sd0rxn_i	入力	Serdesシリアルデータ入力（ネガティブ）
sd0_rext_i	入力	PMA PLL用のアナログリファレンスリターン信号への外部抵抗接続信号です。基板上で処理を行います
sd0_refret_i	入力	PMA PLL用のアナログリファレンスリターン信号です。基板上で処理を行います
sd0txp_o	出力	Serdesシリアルデータ出力（ポジティブ）
sd0txn_o	出力	Serdesシリアルデータ出力（ネガティブ）
rx_data3_o	出力	MPCS入力Byte3データループバック出力
rx_data2_o	出力	MPCS入力Byte2データループバック出力
rx_data1_o	出力	MPCS入力Byte1データループバック出力
rx_data0_o	出力	MPCS入力Byte0データループバック出力

5. 各モジュール概要

■top.v

本デザインのトップモジュールです。

■mpcs_rst_gen.v

MPCS (pcs_8b10b.v) が要求するリセットシーケンス生成モジュールです。resetn信号をトリガとして以下のシーケンスを生成しています。

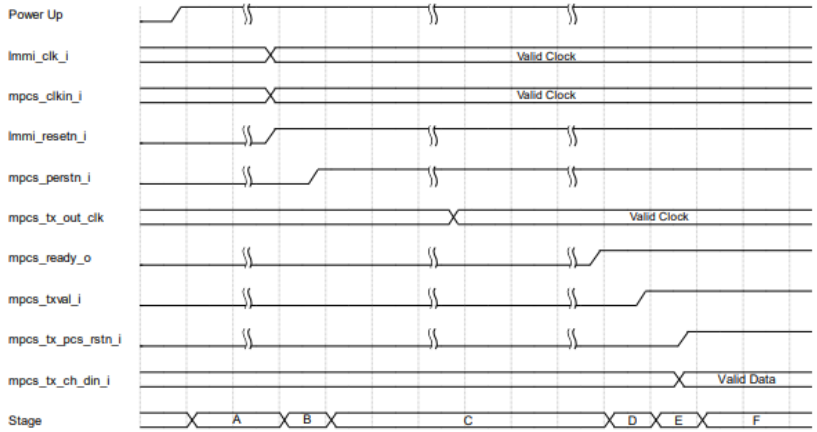


Figure 7.18. MPCS Mode Reset Sequence (Tx Path)

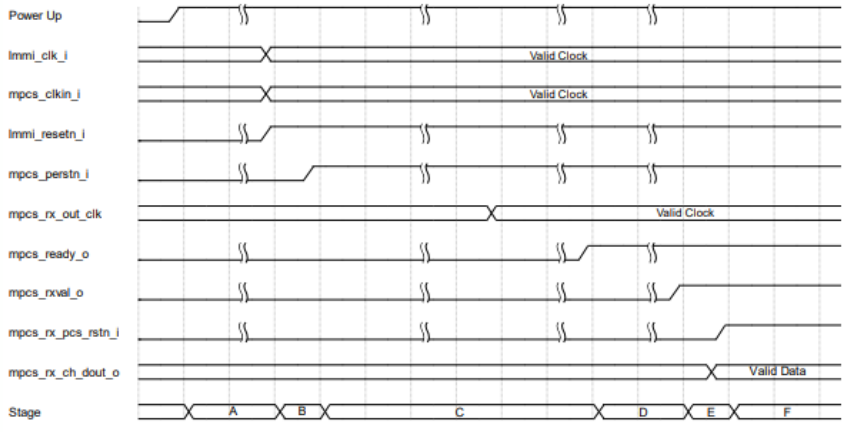
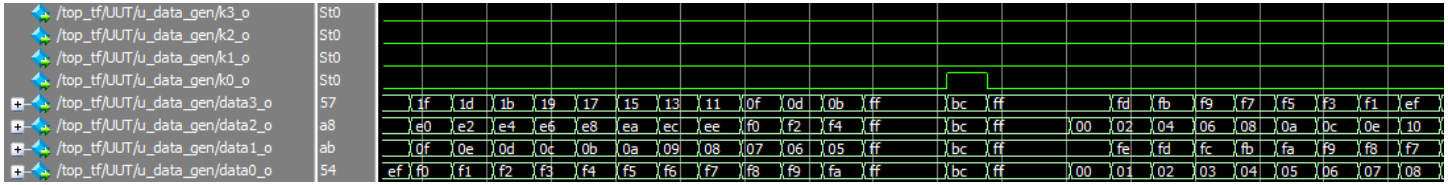


Figure 7.19. MPCS Mode Reset Sequence (Rx Path)

5. 各モジュール概要

■data_gen.v

MPCS (pcs_8b10b.v) への入力データを生成するモジュールです。データは8bitのアップカウンタとダウンカウンタで生成しており、Byte0データにはカウント値0xFDのタイミングでK bitにHighを出力しつつComma Characterを出力します。Byte1～Byte3においてもデータにカウント値0xFDのタイミングでComma Characterを出力しますが、K bitにはHighを出力せず、Byte0でアラインを取る仕様となっています。また、ループバックしたバイトデータのアライメントのためComma Characterを中心とした両側2Byteに0xFFを出力しています。



本デザインで使用しているMPCSモジュールを流用してオリジナル回路を設計される場合、このモジュールをユーザー独自の回路に置き換えて使用してください。

5. 各モジュール概要

■lmmi_init_ctrl.v

LMMI（Lattice Memory Mapped Interface）を介したPMA/PCSのレジスタ初期設定用のモジュールです。本デザインではLattice RadiantでのMPCS生成時に設定したGeneric 8B10Bのワードアライメントコードの読み出しを行っていますが、特にPMA/PCSへの設定は行っておらずデザイン動作に関与していません。LMMIやレジスタマップについての詳細は次のテクニカルノートを参照してください。https://www.latticesemi.com/view_document?document_id=53257

本モジュールを使用してPMA/PCSのレジスタ初期設定を行う場合のソース改版例は以下を参照してください。

```
// ------------------------------------------------------------------↓
// PMA/PCS register access sequence↓
// ------------------------------------------------------------------↓

always@(posedge lmmi_clk_i or negedge lmmi_resetn_i)begin↓
    if(!lmmi_resetn_i)begin↓
        pc_ff <= {CNT_WIDTH{1'b0}};↓
        request_ff <= 1'b0;↓
        wr_rdn_ff <= 1'b0;↓
        offset_ff <= 9'h000;↓
        wdata_ff <= 8'h00;↓
        lmmi_active <= 1'b0;↓
        init_done <= 1'b0;↓
    end↓
    else begin↓
        if(lmmi_ready_i && lmmi_state == IDLE)begin↓
            if(pc_ff == {CNT_WIDTH{1'b1}})begin↓
                pc_ff <= pc_ff;↓
            end↓
            else if(pc_ff == 0) begin↓
                lmmi_idle ();↓
            end↓
        //------------------------------------------------------------↓
            else if(pc_ff == 1)begin    // 1st access↓
                if(lmmi_active)begin↓
                    lmmi_idle ();↓
                end↓
                else begin↓
                    lmmi_write ( PMA, 8'h64, 8'h41 );   // PMA Reg 64 : PRBS control register↓
                end                                     // 8'h41(8'b010
            end↓
        //------------------------------------------------------------↓
            else if(pc_ff == 2)begin    // 2nd access↓
                if(lmmi_active)begin↓
                    lmmi_idle ();↓
                end↓
                else begin↓
                    lmmi_read ( PMA, 8'h64 );   // PMA Reg 64 : PRBS co
                end↓
            end↓
        //------------------------------------------------------------↓
                ⋮
```

“PMA/PCS register access sequence”のコメント以下のalways文でこの部分より下を改版します。

アクセス順に沿ってpc_ffの条件式に1から順に設定します。

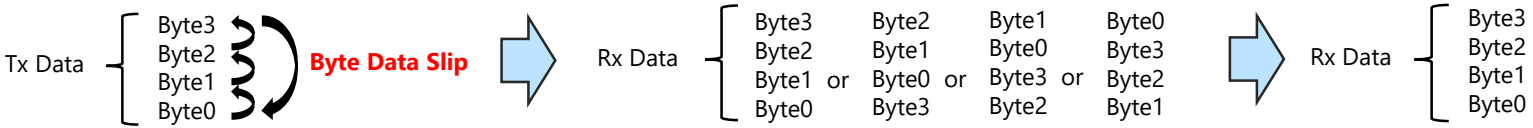
この構文は変えずコピー&ペースト。

レジスタへのライトはlmmi_write(), リードはlmmi_read()のタスクを使用します。
構文は以下の通りです。
lmmi_write (レジスタ領域, アドレス, ライトデータ)
lmmi_read (レジスタ領域, アドレス)
レジスタ領域はPMAレジスタ、PCSレジスタに分かれており、PMAレジスタへのアクセスの場合は”PMA”、PCSレジスタへのアクセスの場合は”PCS”と記載します。
ライトデータはlmmi_wdata_oから出力されます。リードデータはlmmi_rdata_valid_oがHighのタイミングでlmmi_rdata_oから出力されます。

5. 各モジュール概要

■byte_swap.v

MPCSが受信したデータは送信時のバイトデータの並びからスリップして一致しないことがあります。本デザインではByte 0のみK bitにHighが与えられていることを利用して、このモジュールでバイトデータを送信時の並びと同じになるように並び替えています。バイトデータスリップと並び替えのイメージは以下の通りです。



■lane_aligner

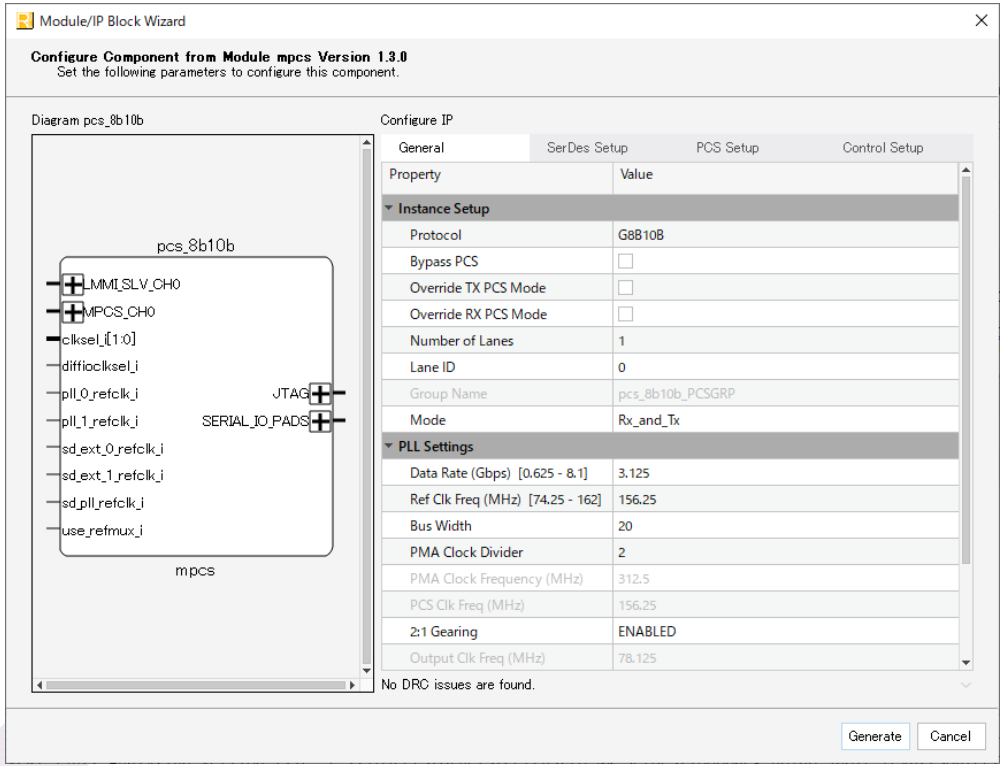
MPCSが受信したデータは最大1クロック分のスキューが発生します。本デザインではComma Characterの両側2Byteに0xFFが挿入されていることを利用して、このモジュールで0xFF,0xFF,0xBC,0xFF,0xFFとなるパターンを検出し、アライメントを取っています。

バイトデータの並び替えとスキューアライメントの動作波形については「6. ファンクションシミュレーション」の項を参照してください。

5. 各モジュール概要

■pcs_8b10b.v

Generic 8B10Bを使用するMPCSモジュールです。以下にMPCS設定の詳細をモジュール生成GUIの図と合わせて説明しています。



Generalタブの設定は以下の通りです。

■Protocol
G8B10Bを選択しています。

■Number of Lane
1Laneを選択しています。

■Lane ID
0を選択しています。Quad0の1Lane目にアサインされます。

■Data Rate
シリアルデータレートの設定です。3.125Gbpsに設定しています。

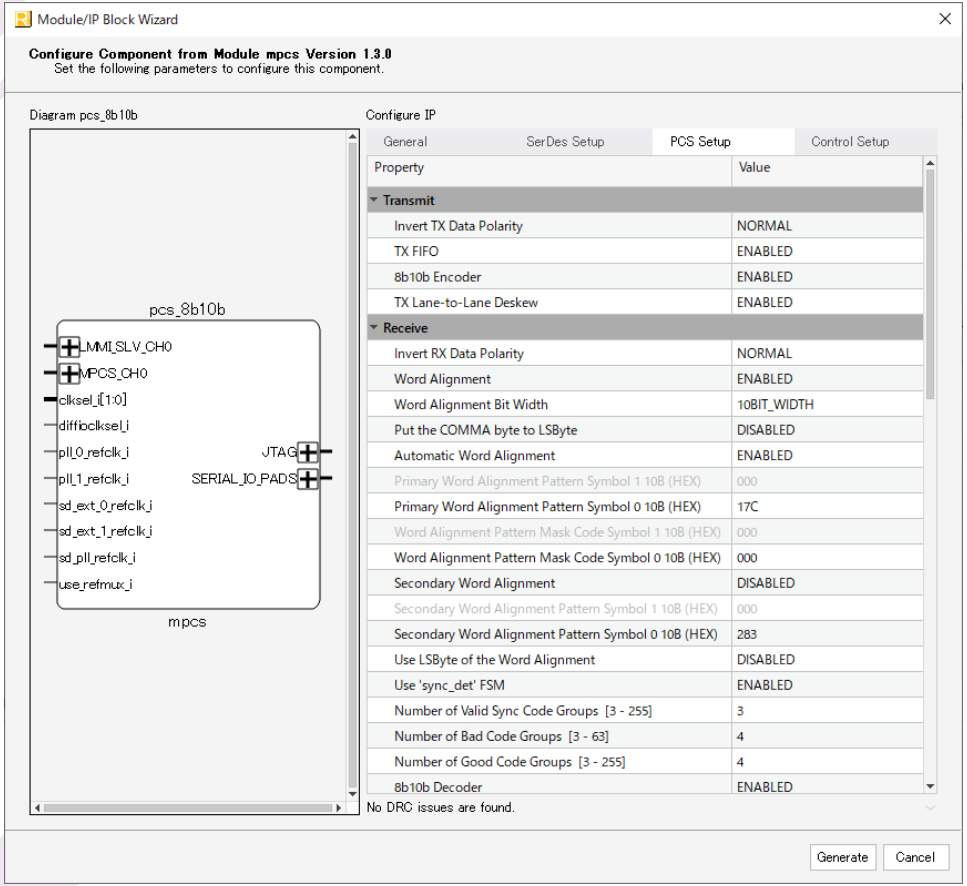
■Ref Clk Freq
MPCSリファレンスクロック周波数の設定です。156.25MHzに設定しています。

■Bus Width
MPCSからの出力データバス幅です。20bitに設定しています。

■2:1 Gearing
Bus Widthで設定した出力バス幅を倍にしてバスクロック周波数を半減させるかどうかの設定です。ENABLEに設定しています。

※ Bus Width = 20bit, 2:1 Gearing = ENABLEのため、MPCSはデータ入力バス、及びデータ出力バス幅が40bitとなり、4Byte Modeの設定となります。

5. 各モジュール概要



PCS Setupタブはデフォルト設定です。主な設定は以下の通りです。

■TX FIFO
ファブリックからMPCSに入力する送信データのクロック同期用FIFO設定です。ENABLEに設定しています。

■Word Alignment Bit Width
Comma Characterのバス幅設定です。10bitに設定しています。

■Primary Word Alignment Pattern Symbol 0 10B
Comma Characterを設定します。10bitコードをLSB側から読んだ値を設定します。本デザインではK28.5のRD-の値を選択しており、10'b00_1111_1010をLSBから読んだ値、10'b01_0111_1100 = 17Cと設定しています。

■Secondary Word Alignment
2つ目のWord Alignment Patternを使用するかどうかの設定です。Word Alignment Patternは上記K28.5のRD-のみ使用するためDisableに設定しています。

■RX FIFO
MPCSからファブリックに出力する受信データのクロック同期用FIFO設定です。ENABLEに設定しています。

（参考）MPCSクロック配線

7.1.1. 8B/10B PCS Clock

Figure 7.1 shows the 8B/10B PCS channel clock diagram.

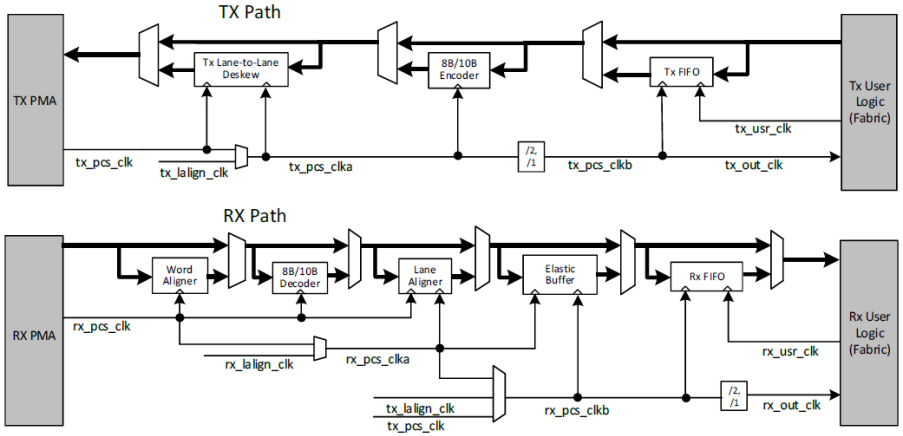


Figure 7.1. 8B/10B PCS Channel Clock Diagram

top.vクロック接続部より抜粋

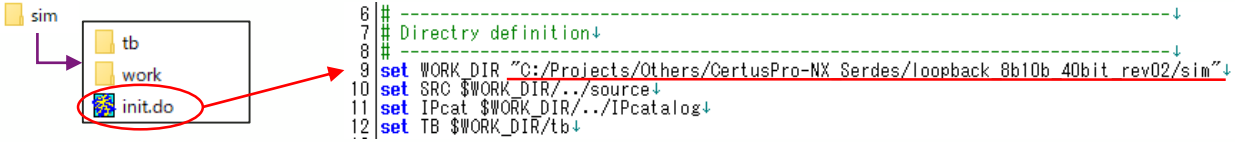
```
123     // ---------------------------------------------------------------------------↓
124     // 8bit Tx data generator↓
125     // ---------------------------------------------------------------------------↓
126     data_gen↓
127         u_data_gen (↓
128             .clk        ( tx_out_clk_0  ),↓
129             .resetn     ( resetn        ),↓
130             .data3_o    ( tx_data3_0    ),↓
131             .data2_o    ( tx_data2_0    ),↓
132             .data1_o    ( tx_data1_0    ),↓
133             .data0_o    ( tx_data0_0    ),↓
134             .k3_o       ( tx_k3_0       ),↓
135             .k2_o       ( tx_k2_0       ),↓
136             .k1_o       ( tx_k1_0       ),↓
137             .k0_o       ( tx_k0_0       )↓
138         );↓
```

```
199             .mpcs_rx_usr_clk_i_0    ( rx_out_clk_0  ),  // User Interface RX Clock Input↓
200             .mpcs_tx_usr_clk_i_0    ( tx_out_clk_0  ),  // User Interface TX Clock Input↓
201             .mpcs_rx_out_clk_o_0    ( rx_out_clk_0  ),  // PCS RX Output Clock↓
202             .mpcs_tx_out_clk_o_0    ( tx_out_clk_0  ),  // PCS TX Output Clock↓
```

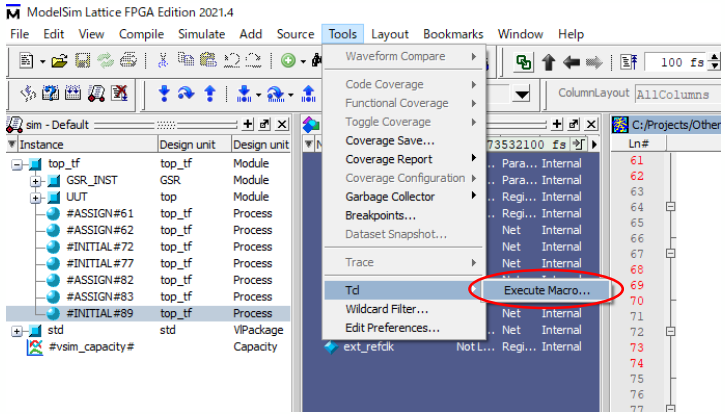
上図の通りMPCSへの入力クロックはTx/Rx共にMPCSから出力されるtx_out_clk, rx_out_clkに同期される必要があります。
また、ファブリック内回路のクロックもこれらのクロックと同期する必要があります。
本デザインを流用して設計される際にはMPCSに接続されるユーザー回路が正しくMPCS出力のtx_out_clk, rx_out_clkで同期するようご注意ください。

6. ファンクションシミュレーション

ファンクションシミュレーションを実施するには、ModelSim Lattice Editionを使用します。
シミュレーション開始前にinit.doファイル内の以下のディレクトリ指定をユーザー環境のsimフォルダのパスに変更する必要があります。
Init.doファイルはsimフォルダの中に格納されています。

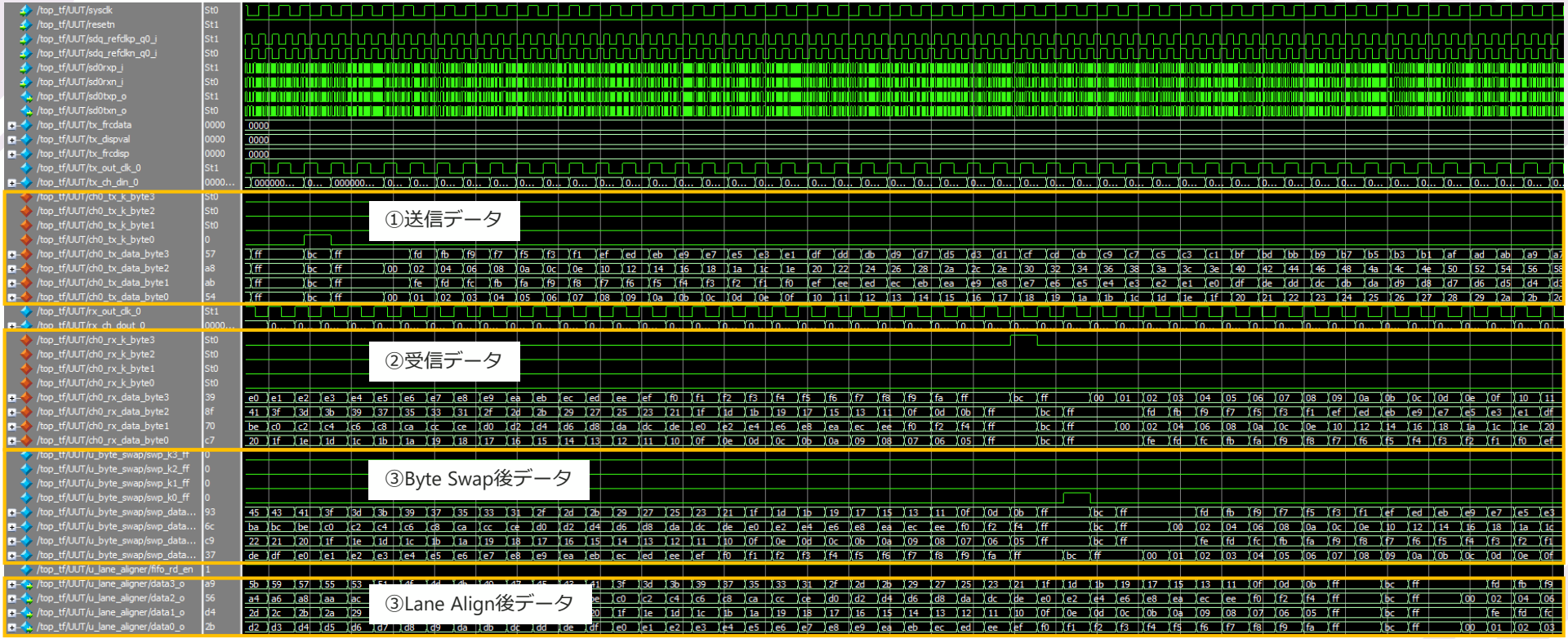


Model Simを起動後、Tools > Tcl > Execute Macro からinit.doファイルを起動します。



Init.doの編集方法、doファイルでのシミュレーション詳細については、以下のページの「ModelSim Lattice Edition DO マクロ ユーザーガイド」を参照してください。
https://www.macnica.co.jp/business/semiconductor/articles/lattice/132003/

6. ファンクションシミュレーション



上図は本デザインのループバックシミュレーション波形です。MPCSに入力した①の送信データがループバック後に受信されると、②のようにバイトデータの並びがスリップし、バイトデータ間で1クロック分のスキューが発生していることが分かります。この受信データをまずはbyte_swap.vにて③のように送信時の並びに修正しています。次にlane_aligner.vにて④のようにスキューを吸収し、送信時のデータと一致させて出力しています。

Revision History

Date	Revision	Page	Change Information
2022/07/13	1.0		First Revision