FPGA および HardCopy デバイスとの RGMII インタフェースの設計

2007 年 11 月 ver. 1.0

Application Note 477

はじめに

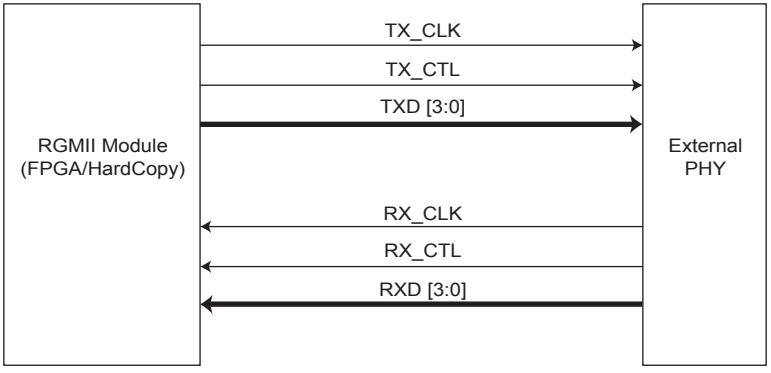
RGMII（Reduced Gigabit Media Independent Interface）は、IEEE 802.3z GMII に代わるもので、ピン数の削減が図られています。ピン数の削減は、クロックの立ち上がりと立ち下がりの両エッジでデータをやりとりし、さらにコントロール信号をマルチプレクスすることによって実現されています。このアプリケーション・ノートでは、Stratix® II、Cyclone®II、および HardCopy®デバイスとの RGMII インタフェースの設計方法を示します。

このアプリケーション・ノートを読む前に、RGMII、SDC（Synopsys Design Constraints）、および TimeQuest タイミング・アナライザに精通しておく必要があります。

システム・レベル図

図 1 に、RGMII の実装のブロック図を示します。RGMII インタフェース・モジュールは、FPGA または HardCopy デバイスの内部に実装されており、外部 RGMII PHY に接続されています。信号はすべて 125 MHz クロック信号と同期します。

図 1. RGMII の信号図



RGMII データは、クロックの両エッジでサンプリングされます。表 1 に、信号の説明を示します。通常、RGMII PHY からのクロックとデータは同時に、つまりエッジ・アラインメントされて生成されるため、クロックは PCB 上ではトレース遅延を追加して配線する必要があります。

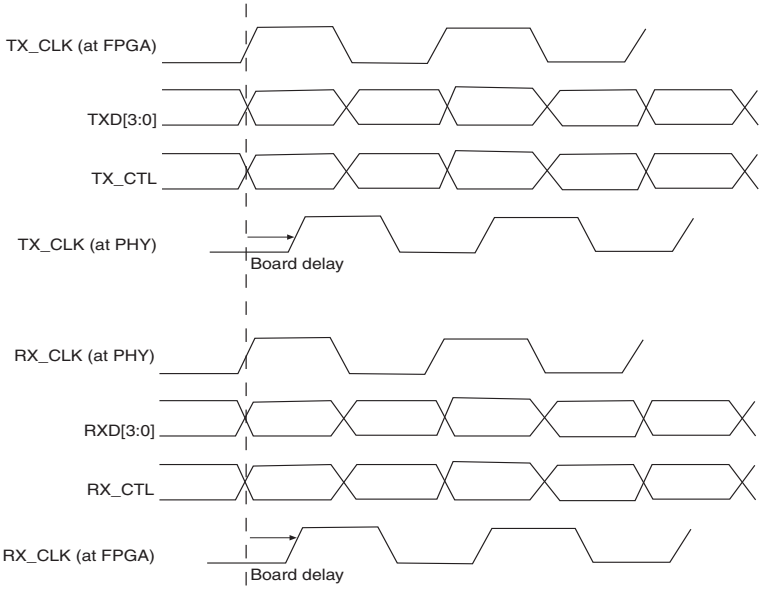
表 1. RGMII の信号の説明

信号	I/O タイプ	説明
TX_CLK	出力	FPGA および HardCopy デバイスからの送信クロック
TXD	出力	TX_CLK のポジティブ・エッジでビット 3 ～ 0、および TX_CLK のネガティブ・エッジでビット 7 ～ 4
TX_CTL	出力	TX_CLK のポジティブ・エッジで TXEN、および TX_CLK のネガティブ・エッジで TXEN と TXERR の論理導関数
RX_CLK	入力	外部 PHY からの受信リファレンス・クロック
RXD	入力	RX_CLK のポジティブ・エッジでビット 3 ～ 0、および RX_CLK のネガティブ・エッジでビット 7 ～ 4
RX_CTL	入力	RX_CLK のポジティブ・エッジで RXDV、および RXC のネガティブ・エッジで RXDV と RXERR の導関数

システム・タイミング

図2に、エッジ・アラインメントされたデータとクロックの図を示します。

図 2. RGMII のタイミング図



データのエッジ・アラインメントが要求される場合、PCB デザインが複雑になります。そのため、RGMII 外部 PHY の最近のリビジョンでは動作時の内部遅延の有無を設定するオプションが提供されます。本資料では、RGMII PHY との送信および受信インタフェースの実装について検討します。

FPGA および HardCopy 送信インタフェースの実装

送信インタフェースの実装は簡単です。図 3 に送信インタフェースのブロック図を示します。

図 3. 送信インタフェースのブロック図

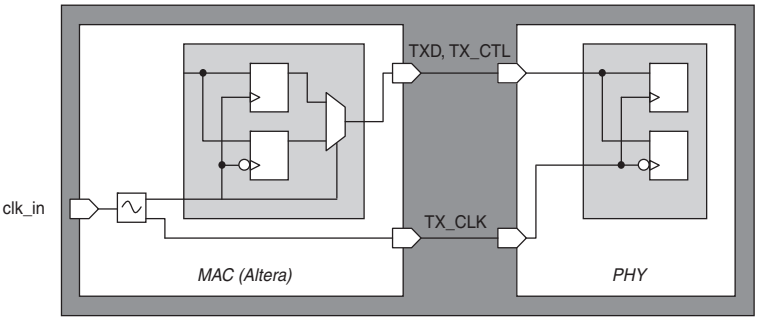
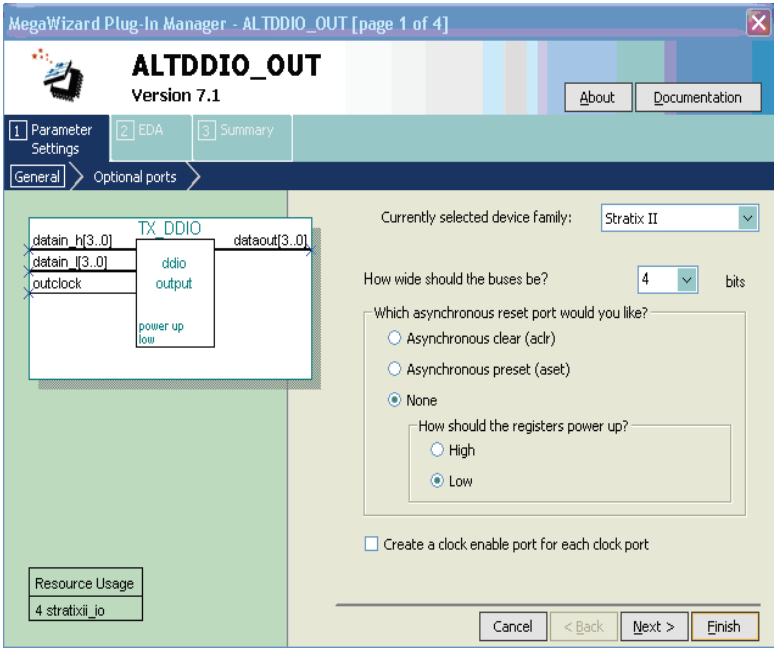


図 4 に示すように、アルテラの DDIO メガファンクションを使用して、任意の DDIO（Double Data Input/Output）I/O レジスタにインタフェースを配置することができます。

図 4. 送信インタフェース用 DDIO メガファンクション



データおよびクロック送信の場合、RGMII をサポートするほとんどの PHY デバイスには、送信または受信クロックに遅延を追加するオプションがあります。このオプションは、デザイン要求に応じてイネーブル／ディセーブルすることができます。PHY デバイス内部でこのオプションをイネーブルして TX_CLK を遅延させる場合、図 5 に示す波形のように、FPGA はエッジ・アラインメントされたクロックをデータと共に生成します。

この場合、PHY デバイスはデータを取り込むために必要に応じてクロックをシフトします。このオプションをディセーブルした場合、図 6 に示す波形のように、FPGA はデータに対してシフトされた（通常はデータの中央にアラインメントされた）クロックを生成しなければなりません。PHY デバイスは、このシフトされたクロックを使用してデータを取り込みます。

図 5. FPGA が生成する TX_CLK（PHY で TX_CLK 遅延をイネーブルした場合）

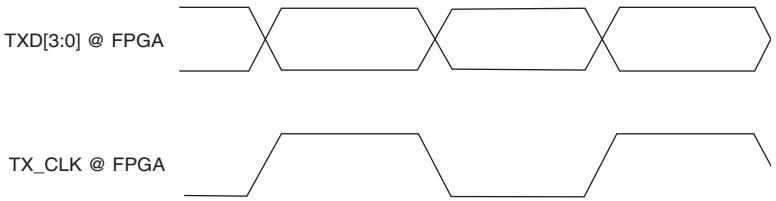
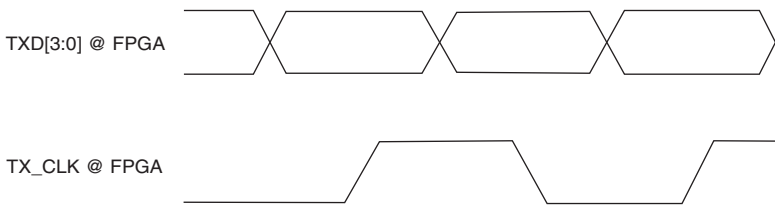


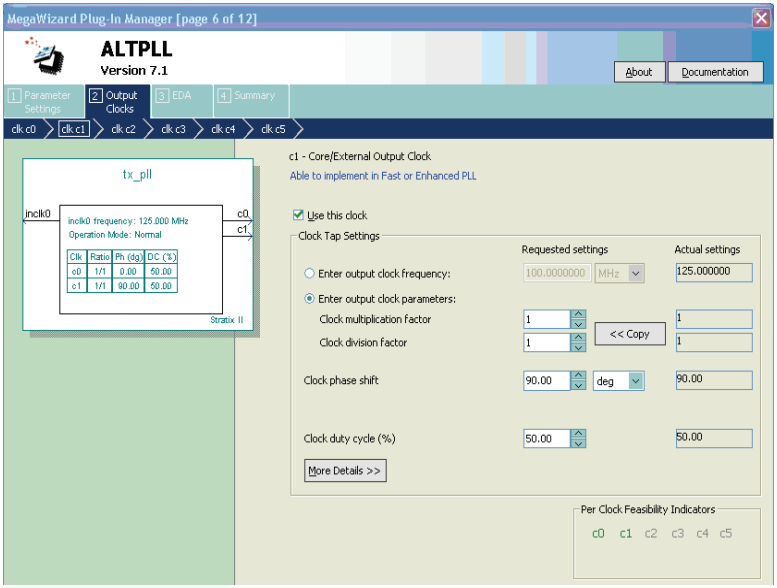
図 6. FPGA が生成する TX_CLK（PHY で TX_CLK 遅延をディセーブルした場合）



データと共に供給されるクロックのアラインメントは、さまざまな手法を使用して行えます。データをラッチするのと同じクロックをドライブしたり、altddio_out メガファンクションなどでトグル・クロック出力レジスタによってクロックを生成することができます。データ出力レジスタ・クロックとは別に出力クロックを生成する場合（例えば 2 個の PLL タップ）、クロック間の関係の調整（PLL 位相の調整など）によってクロックとデータのタイミング関係を変更することができます。これは、アルテラの PLL メガファンクションを使用して行えます。

図 7 に、TXD と TX_CLK 用の 2 つのクロックを個別に生成する PLL メガファンクションを示します。

図 7. TXD および TX_CLK 用のクロックを生成するためのアルテラの PLL メガファンクション



PHY 内部の TX_CLK 遅延オプションをイネーブルすると、データを中央でアラインメントするためのロジックを FPGA に追加する必要がないため、送信インタフェースが簡素化されます。

ソース・シンクロナス・インタフェースの実装について詳しくは、「AN 433: Constraining and Analyzing Source-Synchronous Interfaces」を参照してください。

FPGA および HardCopy 受信インタフェースの実装

図 8 に示すように、受信インタフェースは任意の DDIO I/O レジスタに実装することができます。

図 8. 受信インタフェースのブロック図

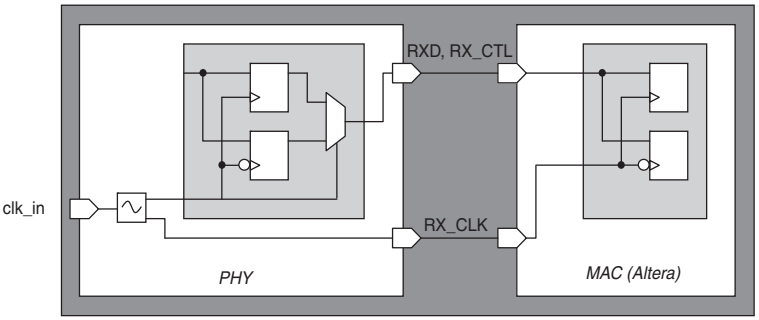
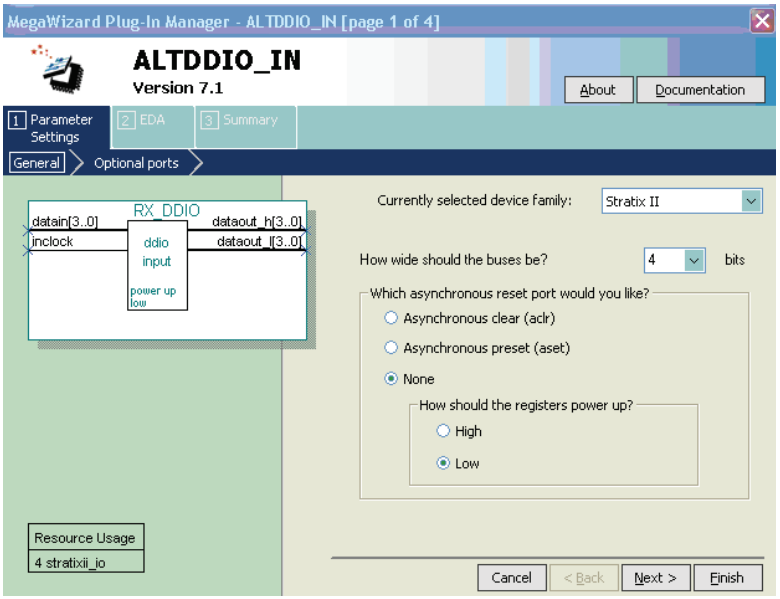


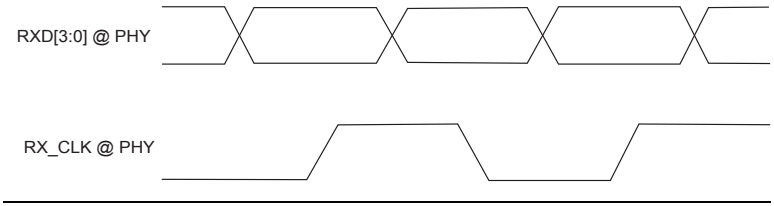
図 9 に、受信インタフェース用 DDIO メガファンクションを示します。

図 9. アルテラの受信インタフェース用 DDIO メガファンクション



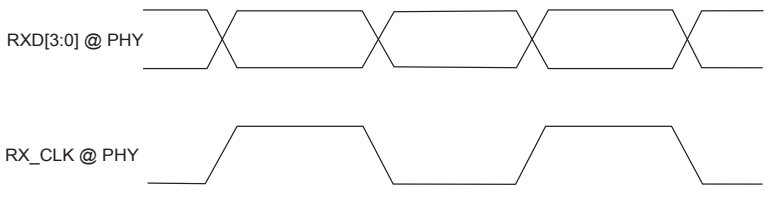
RGMII 外部 PHY には、データを取り込むために RX_CLK に遅延を追加するオプションがあります。外部 PHY デバイスでこのオプションをイネーブルして RX_CLK を遅延させる場合、図 10 に示す波形のように、PHYデバイスはデータの中央でアラインメントされたクロックを送信します。したがって、FPGA および HardCopy デバイスは RX_CLK を使用して受信データを直接取り込むことができ、FPGA または HardCopy デバイスでボード遅延や内部遅延を追加する必要はありません。

図 10. PHY が生成する RX_CLK（PHY で RX_CLK 遅延をイネーブルした場合）



一方、この遅延オプションをディセーブルした場合（タイミング図は図 11 を参照）、FPGA および HardCopy デバイスはデータを取り込むために必要に応じてクロックをシフトしなければなりません。これは、DLL を使用して RX_CLK を DQS ピンに割り当てるか、あるいはクロック・レイテンシを低く抑える必要がある場合は、RX_CLK をグローバルまたはリージョナル・ネットに設定することによって実現できます。

図 11. PHY が生成する RX_CLK（PHY で RX_CLK 遅延をディセーブルした場合）



一般に、FPGA デバイスの受信ロジックの簡素化とタイミング・マージンの改善という理由から、PHY 内部で RX_CLK 遅延オプションをイネーブルすることが推奨されます。

アルテラの推奨事項

クロック・ピンの配置

アルテラでは、デザインを HardCopy に移行する計画がある設計者の方に対し、可能な限り RX_CLK に FPGA のプライマリ・クロック入力を使用することを強く推奨しています。RX_CLK にクロック入力ピン以外のピンを選択した場合、クロック・レイテンシが（したがって、TimeQuest によってレポートされるタイミング結果も）、FPGA デバイスと HardCopy デバイスの間で異なる可能性があります。プライマリ・クロック入力が使用できない場合は、Pin Planner でクロック・ピンの近くに I/O を配置すればこの相違を小さくすることができます。

データおよびクロック・アラインメント

アルテラでは、外部 PHY デバイスに内部遅延オプションがある場合、受信インタフェースに対してこのオプションをイネーブルすることを強く推奨しています。この遅延追加によって設計が大幅に簡素化されることに加え、HardCopy に移行する際のタイミング・マージンが改善します。送信インタフェースについては、送信インタフェースの時間スラックが厳しい場合は遅延オプションをディセーブルし、FPGA または HardCopy デバイス内部の PLL を使用してデータを中央にアラインメントすることを推奨しています。これは、PLL の方が内部補償機能によって優れたタイミング精度を提供するためです。
表 2 に、データおよびクロック・アラインメントに関する推奨事項を要約します。

表 2. データおよびクロック・アラインメントに関する推奨事項

インタフェース	クロック・アラインメント	PHY での遅延設定	アラインメントの実行場所
送信	中央	ディセーブル	FPGA ／ HardCopy
受信	中央	イネーブル	PHY

クロック不確実性の値

また、HardCopy デザインでは、設計者がクロック不確実性の値（18 ページの「クロック不確実性の制約」を参照）を挿入する必要があります。

遅延オプションをディセーブルした場合の FPGA と外部 PHY デバイスの接続

このセクションでは、RGMII のタイミング制約の例を示します。

以下の例では、TX_CLK と RX_CLK の両方について、外部 PHY デバイスの遅延オプションをディセーブルしています。10 ページの図 12 に、外部 PHY デバイスでのタイミング波形を示します。したがって、外部デバイスのタイミング要求を満たすには、11 ページの図 13 および図 14 に示すように、TX_CLK と RX_CLK の両方を FPGA 内部の PLL によって遅延させます。

外部 PHY I/O の想定タイミング要求：

- TXD のセットアップ時間 = 1.0 ns
- TXD のホールド時間 = 0.8 ns
- 入力クロックーデータ間のスキュー（RX_CLK-RXD）= ± 500 ps

トレース遅延、ピン・キャパシタンス、およびデータとクロック間の立ち上がり／立ち下がり時間の違いは無視できるものと仮定しています。

FPGAおよびHardCopyのタイミング制約は､以下のように計算されます。

- 出力最大遅延 = 外部レジスタの tsu = 1.0 ns
- 出力最小遅延 = 外部レジスタの –th = –0.8 ns
- 入力最大遅延 = 外部デバイスの tco = 0.5 ns
- 入力最小遅延 = 外部デバイスの tco = –0.5 ns

ソース・シンクロナス・インタフェースの制約について詳しくは、「AN 433: Constraining and Analyzing Source-Synchronous Interfaces」を参照してください。

図 12 に、外部 PHY デバイスでのタイミング波形を示します。

図 12. 外部 PHY で遅延オプションをディセーブルした場合のタイミング波形

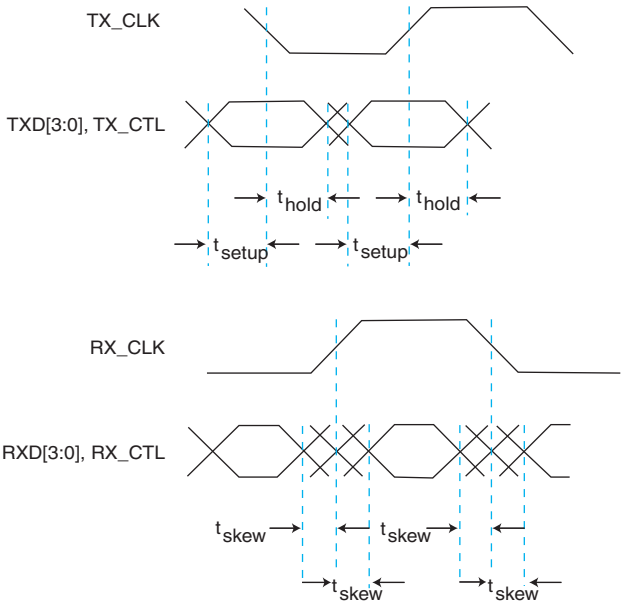


図 13 および図 14 に、FPGA 内部の PLL によって遅延させた TX_CLK および RX_CLK を示します。

図 13. 遅延オプションをディセーブルした場合の RGMII 送信インタフェースの実装

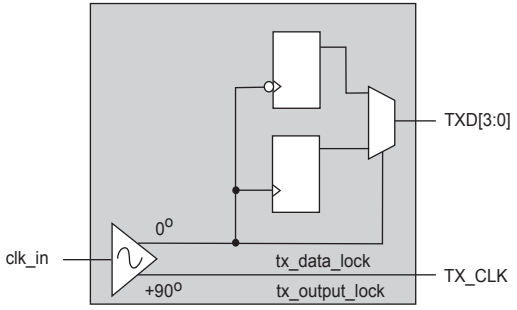
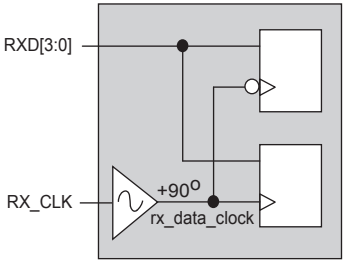


図 14. 遅延オプションをディセーブルした場合の RGMII 受信インタフェースの実装



FPGA（送信）側

送信側には以下のコードを使用します。

```
# Create a 125MHz clock and apply generated clocks to the PLL outputs
# pll|clk[0]: clock for TXD registers
# pll|clk[1]: TX_CLK with 90 degree phase shift--this delay is design-dependent
create_clock -name input_clock -period 8 [get_ports clk_in]

# Below are the pll derived clocks, which can also be generated by typing
# derive_pll_clocks in the TCL console
# You can refer to The Quartus II TimeQuest Timing Analyzer chapter in Quartus II handbook
create_generated_clock -name tx_data_clock -source [get_pins \
{tx_pll|altpll_component|pll|inclk[0]}] [get_pins {tx_pll|altpll_component|pll|clk[0]}]
create_generated_clock -name pll_output -phase 90 -source [get_pins \
{tx_pll|altpll_component|pll|inclk[0]}] [get_pins {tx_pll|altpll_component|pll|clk[1]}]

# Apply a generated clock to the clk_out port
create_generated_clock -name tx_output_clock -source [get_pins \
{tx_pll|altpll_component|pll|clk[1]}] [get_ports {TX_CLK}]

# Set output delay based on the requirements mentioned previously
set_output_delay -clock tx_output_clock -max 1.0 [get_ports TXD*]
set_output_delay -clock tx_output_clock -min -0.8 [get_ports TXD*] -add_delay
set_output_delay -clock tx_output_clock -clock_fall -max 1.0 [get_ports TXD*] -add_delay
set_output_delay -clock tx_output_clock -clock_fall -min -0.8 [get_ports TXD*] -add_delay
set_output_delay -clock tx_output_clock -max 1.0 [get_ports {TX_CTL}]
set_output_delay -clock tx_output_clock -min -0.8 [get_ports {TX_CTL}] -add_delay
set_output_delay -clock tx_output_clock -clock_fall -max 1.0 [get_ports {TX_CTL}] \
-add_delay
set_output_delay -clock tx_output_clock -clock_fall -min -0.8 [get_ports {TX_CTL}] \
-add_delay

# Set false paths to remove irrelevant setup and hold analysis
set_false_path -fall_from [get_clocks tx_data_clock] -rise_to [get_clocks \
tx_output_clock] -setup
set_false_path -rise_from [get_clocks tx_data_clock] -fall_to [get_clocks \
tx_output_clock] -setup
set_false_path -fall_from [get_clocks tx_data_clock] -fall_to [get_clocks \
tx_output_clock] -hold
set_false_path -rise_from [get_clocks tx_data_clock] -rise_to [get_clocks \
tx_output_clock] hold
```

FPGA（受信）側

受信側には以下のコードを使用します。

```
# Create a 125MHz clock and apply generated clocks to the PLL outputs
# virtual_source: an ideal clock in the sourcing device
# RX_CLK: input clock port of the interface
# pll|clk[0]: clock for capturing RXD and RX_CTL with 90 degree phase shift - this delay
# is design-dependent
create_clock -name virtual_source -period 8
create_clock -name RX_CLK -period 8 [get_ports RX_CLK]

# Below is the pll derived clock, which can also be generated by derive_pll_clocks

# Refer to the Quartus II TimeQuest Timing Analyzer chapter in Quartus II Handbook.
create_generated_clock -name rx_data_clk -phase 90 -source [get_pins  \
{rx_pll|altpll_component|pll|inclk[0]}] [get_pins {rx_pll|altpll_component|pll|clk[0]}]

# Set multicycle paths to align the launch edge with the latch edge
set_multicycle_path 0 -setup -end -rise_from [get_clocks virtual_source] -rise_to \
[get_clocks {RX_CLK}]
set_multicycle_path 0 -setup -end -fall_from [get_clocks virtual_source] -fall_to \
[get_clocks {RX_CLK}]

# Set input delay based on the requirements mentioned previously
set_input_delay -max 0.5 -clock [get_clocks virtual_source] -add_delay [get_ports RXD*]
set_input_delay -min -0.5 -clock [get_clocks virtual_source] -add_delay [get_ports RXD*]
set_input_delay -max 0.5 -clock [get_clocks virtual_source] -clock_fall -add_delay \
[get_ports RXD*]
set_input_delay -min -0.5 -clock [get_clocks virtual_source] -clock_fall -add_delay \
[get_ports RXD*]
set_input_delay -max 0.5 -clock [get_clocks virtual_source] -add_delay [get_ports \
{RX_CTL}]
set_input_delay -min -0.5 -clock [get_clocks virtual_source] -add_delay [get_ports \
{RX_CTL}]
set_input_delay -max 0.5 -clock [get_clocks virtual_source] -clock_fall -add_delay \
[get_ports {RX_CTL}]
set_input_delay -min -0.5 -clock [get_clocks virtual_source] -clock_fall -add_delay \
[get_ports {RX_CTL}]

# Set false paths to remove irrelevant setup and hold analysis
set_false_path -fall_from [get_clocks virtual_source] -rise_to [get_clocks rx_data_clk] \
-setup
set_false_path -rise_from [get_clocks virtual_source] -fall_to [get_clocks rx_data_clk] \
-setup
set_false_path -fall_from [get_clocks virtual_source] -fall_to [get_clocks rx_data_clk] \
-hold
set_false_path -rise_from [get_clocks virtual_source] -rise_to [get_clocks rx_data_clk] \
-hold
```

遅延オプションをイネーブルした場合の FPGA と外部 PHY デバイスの接続

外部 PHY デバイスが、内部遅延をサポートしている場合は、追加のロジックは不要です。FPGA は、エッジ・アラインメントされたデータを送信し、中央にアラインメントされたデータを受信します（15 ページの図 15 に、外部 PHY デバイスでのタイミング波形を示します）。したがって、図 16 および 17 ページの図 17 に示すように、送信クロックは送信データと同じクロックでドライブすることができ、受信クロック RX_CLK は FPGA または HardCopy デバイスがデータの取り込みに直接使用することができます。

外部 I/O の想定タイミング要求：

- 最小セットアップ時間 = 0.9 ns
- 最小ホールド時間 = 2.7 ns
- PHY デバイス内部で追加される最小クロック遅延 Tdelay_min = 1.2 ns
- PHY デバイス内部で追加される最大クロック遅延 Tdelay_max = 2.8 ns

トレース遅延、ピン・キャパシタンス、およびデータとクロック間の立ち上がり／立ち下がり時間の違いは無視できるものと仮定しています。

FPGA/HardCopy のタイミング制約は、以下のように計算されます。

- RX_CLK（90 度位相シフト）の立ち上がりエッジ = 2 ns
- 計算された出力最大遅延 = 外部レジスタの tsu = –0.9 ns
- 計算された出力最小遅延 = 外部レジスタの -th = –2.7 ns
- 計算された入力最大遅延 = 外部デバイスの tco = 2.8 ns – 2 ns（RX_CLK の立ち上がりエッジ）= 0.8 ns
- 計算された入力最小遅延 = 外部デバイスの tco = 1.2 ns – 2 ns（RX_CLK の立ち上がりエッジ）= –0.8 ns

図 15 に、外部 PHY デバイスでのタイミング波形を示します。

図 15. 外部 PHY デバイスで遅延オプションをイネーブルした場合のタイミング波形

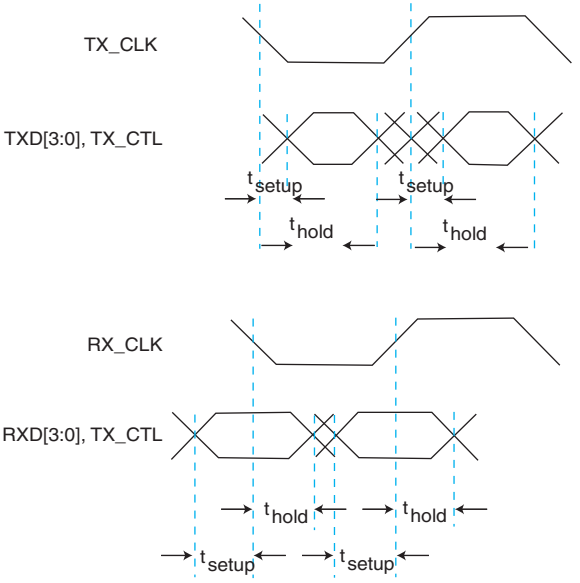
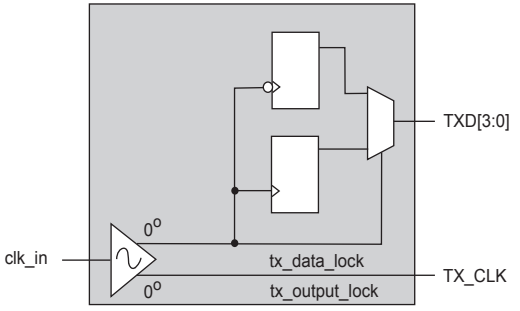


図 16. 遅延オプションをイネーブルした場合の RGMII 送信インタフェースの実装



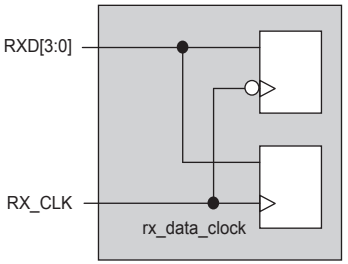
FPGA（送信）側

送信側には以下のコードを使用します。

```
# Create a 125MHz clock and apply generated clocks to # the PLL outputs
# pll|clk[0]: TX_CLK
# pll|clk[1]: clock for TXD registers
create_clock -name input_clock -period 8 [get_ports {clk_in}]

# Below are the pll derived clocks, which can also be generated by typing derive_pll_clocks
# in the TCL console
# Refer to the Quartus II TimeQuest Timing Analyzer chapter in Quartus II Handbook.
create_generated_clock -name tx_data_clock -source [get_pins \
{tx_pll|altpll_component|pll|inclk[0]}] [get_pins {tx_pll|altpll_component|pll|clk[0]}]
create_generated_clock -name tx_output_clock -source [get_pins \
{tx_pll|altpll_component|pll|clk[0]}] [get_ports {TX_CLK}]

# Set output delay based on the requirements mentioned previously
set_output_delay -clock tx_output_clock -max -0.9 [get_ports TXD*]
set_output_delay -clock tx_output_clock -min -2.7 [get_ports TXD*] -add_delay
set_output_delay -clock tx_output_clock -clock_fall -max -0.9 [get_ports TXD*] \
-add_delay
set_output_delay -clock tx_output_clock -clock_fall -min -2.7 [get_ports TXD*] \
-add_delay
set_output_delay -clock tx_output_clock -max -0.9 [get_ports {TX_CTL}]
set_output_delay -clock tx_output_clock -min -2.7 [get_ports {TX_CTL}] -add_delay
set_output_delay -clock tx_output_clock -clock_fall -max -0.9 [get_ports {TX_CTL}] \
-add_delay
set_output_delay -clock tx_output_clock -clock_fall -min -2.7 [get_ports {TX_CTL}] \
-add_delay

# Set multicycle paths to align the launch edge with the latch edge
set_multicycle_path 0 -setup -end -rise_from [get_clocks tx_data_clock] -rise_to \
[get_clocks tx_output_clock]
set_multicycle_path 0 -setup -end -fall_from [get_clocks tx_data_clock] -fall_to \
[get_clocks tx_output_clock]

# Set false paths to remove irrelevant setup and hold analysis
set_false_path -fall_from [get_clocks tx_data_clock] -rise_to [get_clocks \
tx_output_clock] -setup
set_false_path -rise_from [get_clocks tx_data_clock] -fall_to [get_clocks \
tx_output_clock] -setup
set_false_path -fall_from [get_clocks tx_data_clock] -fall_to [get_clocks \
tx_output_clock] -hold
set_false_path -rise_from [get_clocks tx_data_clock] -rise_to [get_clocks \
tx_output_clock] -hold
```

図 17 に、遅延オプションをイネーブルした場合の RGMII 受信インタフェースの実装を示します。

図 17. 遅延オプションをイネーブルした場合の RGMII 受信インタフェースの実装



FPGA（受信）側

受信側には以下のコードを使用します。

```
# Create a 125MHz clock
# virtual_source: an ideal clock in the sourcing device
# RX_CLK: input clock port of the interface; 90 deg phase shifted
create_clock -name virtual_source -period 8
create_clock -name RX_CLK -period 8 -waveform { 2 6 } [get_ports {RX_CLK}]

# Set input delay based on the requirements mentioned previously
# RX_CLK is 90 deg phase shifted
# Input delay is relative to the rising and falling edges of the clock
set_input_delay -max 0.8 -clock [get_clocks virtual_source] -add_delay [get_ports RXD*]
set_input_delay -min -0.8 -clock [get_clocks virtual_source] -add_delay [get_ports RXD*]
set_input_delay -max 0.8 -clock_fall -clock [get_clocks virtual_source] -add_delay \
[get_ports RXD*]
set_input_delay -min -0.8 -clock_fall -clock [get_clocks virtual_source] -add_delay \
[get_ports RXD*]
set_input_delay -max 0.8 -clock [get_clocks virtual_source] -add_delay [get_ports \
{RX_CTL}]
set_input_delay -min -0.8 -clock [get_clocks virtual_source] -add_delay [get_ports \
{RX_CTL}]
set_input_delay -max 0.8 -clock_fall -clock [get_clocks virtual_source] -add_delay \
[get_ports {RX_CTL}]
set_input_delay -min -0.8 -clock_fall -clock [get_clocks virtual_source] -add_delay \
[get_ports {RX_CTL}]

# Set false paths to remove irrelevant setup and hold analysis
set_false_path -fall_from [get_clocks virtual_source] -rise_to [get_clocks {RX_CLK}] \
-setup
set_false_path -rise_from [get_clocks virtual_source] -fall_to [get_clocks {RX_CLK}] \
-setup
set_false_path -fall_from [get_clocks virtual_source] -fall_to [get_clocks {RX_CLK}] \
-hold
set_false_path -rise_from [get_clocks virtual_source] -rise_to [get_clocks {RX_CLK}] \
-hold
```

クロック不確実性の制約

HardCopy デバイスの場合、設計者がタイミング制約ファイルにクロック不確実性の値を挿入する必要があります。これは、TimeQuest コンソールで derive_clock_uncertainty コマンドを入力することによって簡単に行えます。

クロック不確実性について詳しくは、「Quartus II ハンドブック」の「Quartus II TimeQuest タイミング・アナライザ」の章を参照してください。

HardCopy の場合のクロック不確実性

以下は、derive_clock_uncertainty コマンドによって生成されるクロック不確実性の制約です。クロック不確実性の値は、リソース使用量によって異なることがあります。以下の不確実性の値は例示のみを目的としています。

```
set_clock_uncertainty -from tx_data_clock -to tx_output_clock -setup 0.100
set_clock_uncertainty -from tx_data_clock -to tx_output_clock -hold 0.050
set_clock_uncertainty -from tx_data_clock -to tx_data_clock -setup 0.100
set_clock_uncertainty -from tx_data_clock -to tx_data_clock -hold 0.050
set_clock_uncertainty -from virtual_source -to rx_data_clk -setup 0.130
set_clock_uncertainty -from virtual_source -to rx_data_clk -hold 0.130
set_clock_uncertainty -from rx_data_clk -to rx_data_clk -setup 0.100
set_clock_uncertainty -from rx_data_clk -to rx_data_clk -hold 0.050
set_clock_uncertainty -from virtual_source -to RX_CLK -setup 0.100
set_clock_uncertainty -from virtual_source -to RX_CLK -hold 0.100
set_clock_uncertainty -from RX_CLK -to RX_CLK -setup 0.100
```

まとめ

アルテラ FPGA との RGMII インタフェースの設計は簡単なプロセスです。Stratix II、Cyclone II、および HardCopy デバイスは、ハードウェア、ソフトウェア、ドキュメントをはじめ、堅牢な RGMII インタフェースの開発に役立つ完全なソリューションを提供しています。

このデザイン・ガイドラインと推奨事項に従うことにより、特にデザインを HardCopy デバイスに移行する場合において、信頼性の高いインタフェースを簡単に実装することができます。

参考資料

このアプリケーション・ノートでは、以下の資料を参照しています。

- AN 433: Constraining and Analyzing Source-Synchronous Interfaces
- 「Quartus II ハンドブック Volume 3」の「Quartus II TimeQuest タイミング・アナライザ」の章

改訂履歴

表 3 に、本資料の改訂履歴を示します。

表 3. 改訂履歴

日付およびドキュメント・バージョン	変更内容	概要
2007 年 11 月 ver. 1.0	初版	—

101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support:
www.altera.com/support
Literature Services:
www.altera.com/literature