Spartan-3 ジェネレーション FPGA で効率的な PCB レイアウトを達成するための LVDS 信号の反転

XAPP491 (v1.0) 2006 年 10 月 4 日

著者 : Nick Sawyer、Gary Lawman

本資料は英語版 (v1.0) を翻訳したものです。英語の更新バージョンがリリースされている場合には、最新の英語版を必ずご参照ください。

概要

LVDS や LVPECL などの差動信号は、単純な 4 層または 6 層の PCB に配線するのが困難で、ビアを多用する結果となります。これは、ドライバの正のピンでレシーバの対応する正のピンを、ドライバの負のピンでレシーバの負のピンを駆動する必要があるからです。このため、トレースの向きが不正となり、回路にインバータを追加したような状態になることがあります。このアプリケーション ノートでは、Spartan™-3 ジェネレーション FPGA でレシーバのデータパスにインバータを追加することにより、ビアの多用を回避し、PCB をリスピンすることなく PCB トレースのスワップを修正する方法を示します。ここに示す手法は FPGA がドライバとなる場合にも適用可能であり、トレースのスワップにより PCB を別のドライバまたはコネクタに配線しやすくなります。

はじめに

図 1 に、正のピンでレシーバの正のピンを駆動し、負のピンでレシーバの負のピンを駆動する PCB レイアウトの例を示します。ピンが誤ってスワップされると、PCB トレースがインバータとなり、ボードのリスピンが必要となる可能性があります。

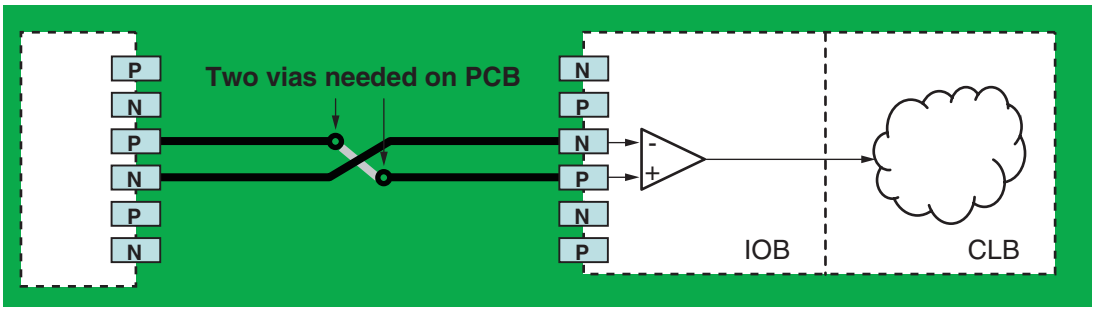

図 1 : ビアを使用したトレースのスワップが必要な PCB レイアウト

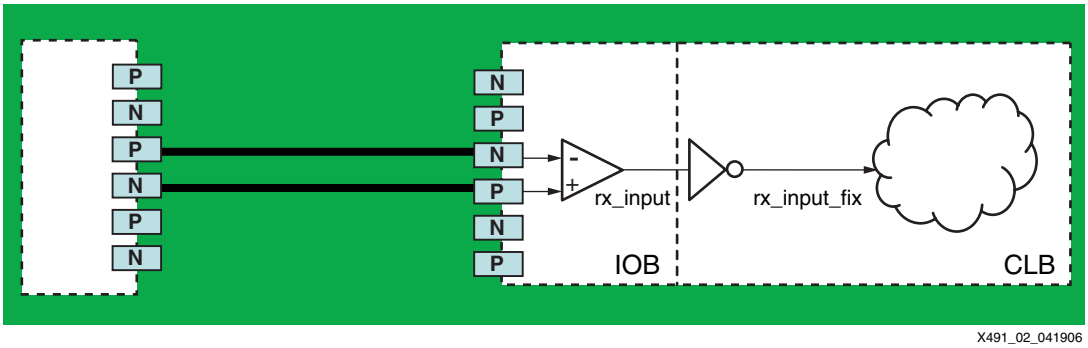

図 2 : ビアを使用したトレースのスワップが不要な PCB レイアウト

図 2 に、Spartan-3 ジェネレーション FPGA でレシーバのデータパスにインバータを追加することにより、この問題を解決したレイアウトを示します。この手法を使用すると、トレースを効果的にスワップすることにより配線を単純化できるので、最高のシグナル インテグリティを達成する差動ペアのレイアウトが可能となります。適用したスワップは、FPGA 内で修正できます。DCM が使用されている場合 (「非同期入力」を参照)、このスワップはデータ ラインにのみ適用され、クロック ラインには適用されません。ラインをスワップしてもデバイスが破損することはありません。

インバータのロジックへの組み込み

インバータは、次の 2 つの場合に前方のロジックに組み込まれます。

1. フリップフロップ入力を直接駆動する場合
2. ロジック ファンクションを駆動する場合

1 の場合、Spartan-3 ジェネレーション FPGA では、図 3 に示すように CLB フリップフロップの直接入力 (D) 上にマルチプレクサがあります。このマルチプレクサは、入力信号またはその反転入力信号のどちらかを選択します。このマルチプレクサはコンフィギュレーション セルからコンフィギュレーションされ、デバイスに読み込まれるビットストリームで初期化されるので、動作中にアクセスすることはできません。

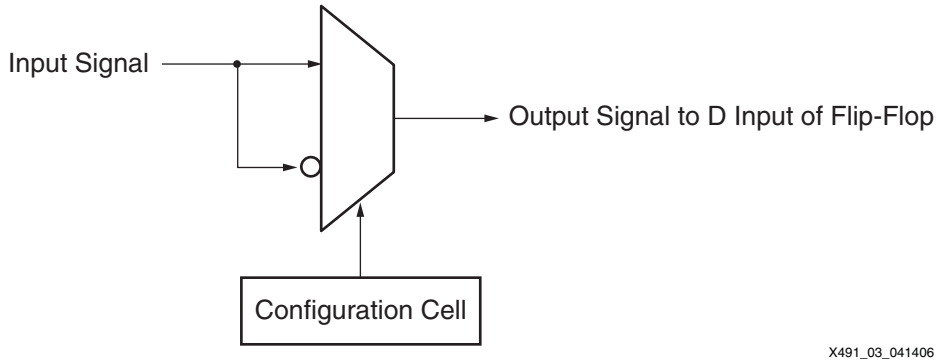

図 3 : CLB フリップフロップの前にあるプログラマブル インバータ

2 の場合、インバータは単純に組み込まれます。たとえば、B = ~A を実行するインバータの後に D = B AND C を実行する AND ゲートがある場合、D = ~A AND C という AND ゲートに変換され、インバータは使用されません。インバータが組み込まれることにより、追加のロジックが使用されることも、余分な遅延が発生することもありません。

このインバータの組み込みは、IOB 出力フリップフロップにも適用されます。この場合も追加のロジックは不要で、遅延も発生しません。この機能は、FPGA で N および P の LVDS 出力に直接一致する定義済みのピンを持つコネクタを駆動する場合に役立ちます。

非同期入力

図 2 は、最も単純な例を示しています。スワップされた受信 LVDS 信号が、FPGA 内の組み合わせロジックで使用されます。この例では、コードに単純なインバータを追加するだけです。このインバータの Verilog および VHDL のコードは、次のとおりです。

Verilog :
```
assign rx_input_fix = ~rx_input;
```

VHDL :
```
rx_input_fix <= not rx_input;
```

このインバータは、入力信号で駆動される組み合わせロジックまたは FPGA 内のフリップフロップの D 入力に組み込むことはできますが、FPGA の IOB のフリップフロップ、DCM、または BUFGMUX クロック バッファに組み込むことはできません。そのため、データの入力に使用するクロック信号にピンのスワップを適用することはできません。クロックが単にシステムのオシレータである場合は、悪影響なしでラインをスワップし、再反転せずにおくことができます。

図 4 に、入力が n 個の信号ペアのバスである例を示します。一部の信号ペアがスワップされています。この例では、デザインで n 入力に対応するマスクを定義するのが理にかなっています。マスクは、修正が必要なビットを選択的に反転 (XOR) するために使用します。図 4 では、ビット 0 と 2 は正しく、ビット 1 は反転が必要です。この反転をコードで記述するには、generate loop を使用して入力バッファをインスタンシエートし、必要なビットを反転するのが最適な方法です。

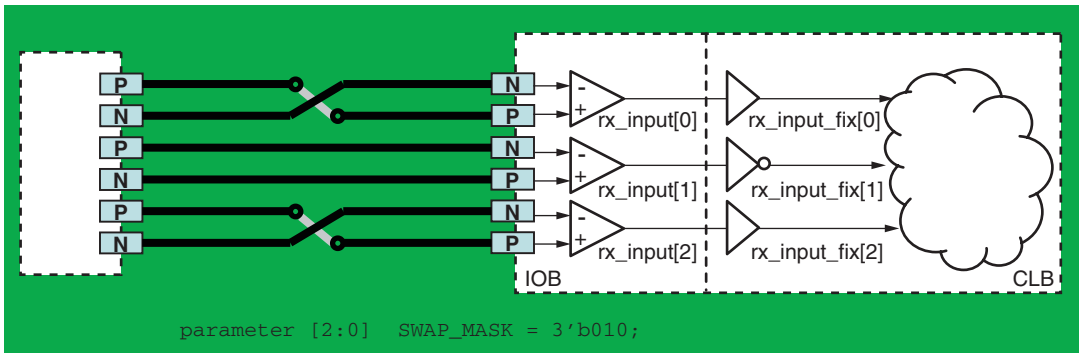

図 4 : 配線を単純化するため一部のバス トレースの極性をスワップ

次に、generate loop を使用して受信信号を反転する Verilog コードを示します。

```
.
parameter [2:0] SWAP_MASK = 3'b010;
.
.
genvar i;
generate
for (i = 0; i <= 2; i = i + 1)
begin: loop0
IBUFDS
#(.IOSTANDARD("LVDS_25"), .IBUF_DELAY_VALUE("0"), .DIFF_TERM("FALSE"))
ibuf_d   (.I(datain_p[i]), .IB(datain_n[i]), .O(rx_input[i]));
assign rx_input_fix[i] = rx_input[i] ^ SWAP_MASK[i];
end
endgenerate
```

次に、generate loop を使用して受信信号を反転する VHDL コードを示します。

```
.
constant SWAP_MASK : std_logic_vector(2 downto 0):= "010";
.
.
loop0: for i in 0 to 2 generate
ibuf_d: ibufds generic map
(IOSTANDARD => "LVDS_25", IBUF_DELAY_VALUE => "0", DIFF_TERM => FALSE)
port map
(i => datain_p(i), iB => datain_n(i), o => rx_input(i));
rx_input_fix(i) <= rx_input(i) xor SWAP_MASK(i);
end generate;
```

上記のコードで赤の太字で示された数値を変更すると、さまざまなビット幅に簡単に応用できます。

IOB 入力フリップフロップの同期的な使用

LVDS は高速信号伝送に使用されることがほとんどなので、入力信号は通常 IOB フリップフロップに取り込まれます。データは、次のいずれかの方法で取り込みます。

- シングル データ レート (SDR) : IOB 内の 1 つの (通常立ち上がりエッジでトリガされる) フリップフロップを使用します。
- ダブル データ レート (DDR) : 立ち上がりエッジおよび立ち下がりエッジの両方でトリガされるフリップフロップを使用して、入力データ ラインを取り込みます。

どちらの場合も、IOB ブロック内のフリップフロップには反転入力がないので、入力増幅器とフリップフロップの間で入力信号を反転することはできません。インバータは、IOB 入力フリップフロップの後に追加する必要があり、その後のレジスタ付きロジックまたは組み合わせロジックに組み込むことができます。

SDR の例

図 5 に、IOB 内のフリップフロップを 1 つ使用した SDR の例を示します。

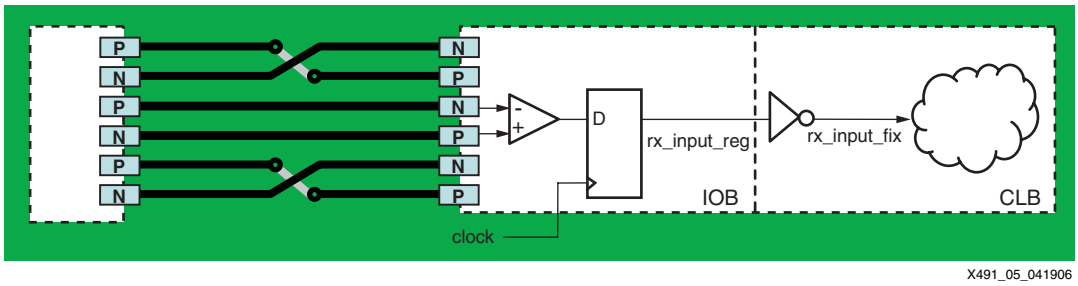

図 5 : SDR レジスタ付きレシーバ

次に、先ほどの generate loop の例で 1 つのレジスタを使用した SDR の場合のコードを示します。フリップフロップのインスタンシエーションが追加されていることだけが異なります。

Verilog :

```
.
parameter [2:0] SWAP_MASK = 3'b010;
.
.
genvar i;
generate
for (i = 0; i <= 2; i = i + 1)
begin: loop0
IBUFDS#(.IOSTANDARD("LVDS_25"), .IFD_DELAY_VALUE("0"), .DIFF_TERM("FALSE"))
ibuf_d   (.I(datain_p[i]), .IB(datain_n[i]), .O(rx_input[i]));
FD fd_d (.C(clkin), .D(rx_input[i]), .Q(rx_input_reg[i]));
assign rx_input_fix[i] = rx_input_reg[i] ^ SWAP_MASK[i];
end
endgenerate
```

VHDL :

```
.
constant SWAP_MASK : std_logic_vector(2 downto 0):= "010";
.
.
loop0: for i in 0 to 2 generate
ibuf_d: ibufds
generic map (IOSTANDARD => "LVDS_25", IFD_DELAY_VALUE => "0", DIFF_TERM => FALSE)
port map    (i => datain_p(i), iB => datain_n(i), o => rx_input(i));
fd_d: fd     port map    (c => clkin, d => rx_input(i), q => rx_input_reg(i));
rx_input_fix(i) <= rx_input_reg(i) xor SWAP_MASK(i);
end generate;
```

上記のコードで赤の太字で示された数値を変更すると、さまざまなビット幅に簡単に応用できます。

入力 DDR の例

図 6 に、各入力ラインで反転が必要な内部データ ラインが 2 つ生成されるレシーバ DDR の例を示します。Spartan-3E FPGA の DDR 入力には、入力フリップフロップの新しい IDDR2 構造を使用することをお勧めします。この構造では、立ち下がりエッジから次の立ち上がりエッジまでのパスを削除することにより、内部ロジックが設計しやすくなっています。IDDR2 の詳細は、DS312 『Spartan-3 FPGA ファミリ : データシート』を参照してください。

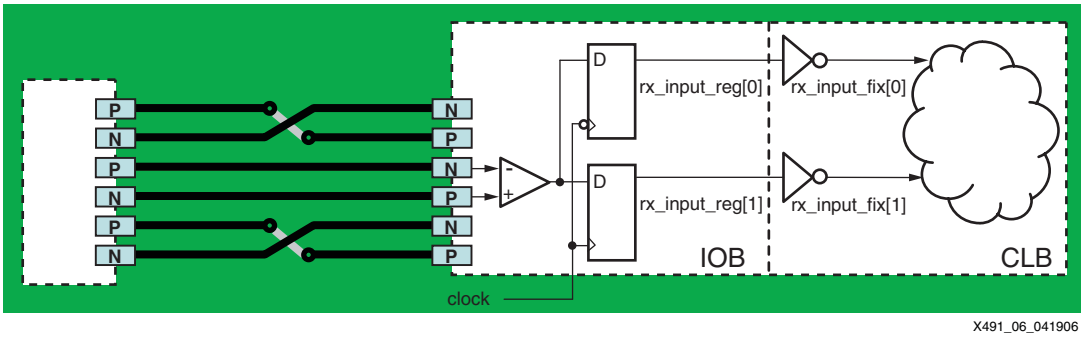

図 6 : DDR レジスタ付きレシーバ

次に、先ほどの generate loop の例で DDR レジスタ付きレシーバの場合のコードを示します。Spartan-3E FPGA の IDDR2 のインスタンシエーションが追加されていることだけが異なります。Spartan-3 デバイスには IDDR2 構造は含まれていないので、コードは多少異なります。詳細は、付属の ZIP ファイルを参照してください (「デザイン ファイル」を参照)。

Verilog :

```
.
parameter [2:0] SWAP_MASK = 3'b010;
.
.
genvar i;
generate
for (i = 0; i <= 2; i = i + 1)
begin: loop0
IBUFDS#(.IOSTANDARD("LVDS_25"), .IFD_DELAY_VALUE("0"), .DIFF_TERM("FALSE"))
ibuf_d   (.I(datain_p[i]), .IB(datain_n[i]), .O(rx_input[i]));
IDDR2 #(.DDR_ALIGNMENT("C0")) fd_ioc(.C0(clkin), .C1(notclk), .D(rx_input[i]),
        .CE(1'b1), .R(1'b0), .S(1'b0), .Q0(rx_input_reg[i+3]),
        .Q1(rx_input_reg[i]));
assign rx_input_fix[i] = rx_input_reg[i] ^ SWAP_MASK[i];
assign rx_input_fix[i+3] = rx_input_reg[i+3] ^ SWAP_MASK[i];
end
endgenerate
```

VHDL :

```
.
constant SWAP_MASK : std_logic_vector(2 downto 0):= "010";
.
.
loop0: for i in 0 to 2 generate
ibuf_d : ibufds
generic map (IOSTANDARD => "LVDS_25", IFD_DELAY_VALUE => "0", DIFF_TERM => FALSE)
port map    (i => datain_p(i), iB => datain_n(i), o => rx_input(i));
fd_d : iddr2
generic map (DDR_ALIGNMENT => "C0")
port map (c0 => clkin, c1 => notclock, d => rx_input(i), ce => '1', r => '0',
          s => '0', q0 => rx_input_reg(i+3), q1 => rx_input_reg(i));
rx_input_fix(i)   <= rx_input_reg(i) xor SWAP_MASK(i);
rx_input_fix(i+3) <= rx_input_reg(i+3) xor SWAP_MASK(i);
```

上記のコードで赤の太字で示された数値を変更すると、さまざまなビット幅に簡単に応用できます。

DDR を使用する際、ビット操作が重要になることがあります。generate loop の DDR の例では、下位ビットはクロックの立ち下がりエッジで取り込まれ、上位ビットが次の立ち上がりエッジで取り込まれるバスが生成されます。図 7 に、DDR デザインをシミュレーションした際のビット取り込みの波形を示します。このシミュレーションでは、トレースはすべてピン スワップされておらず、各ビット値が明確に示されています。

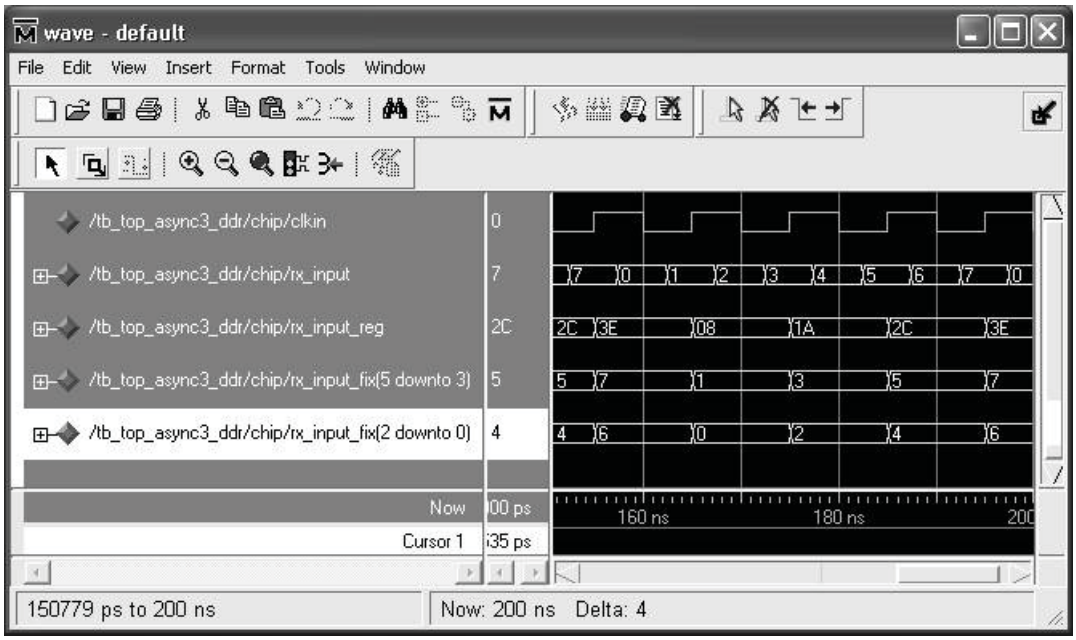

図 7 : 入力 DDR のシミュレーション

出力 DDR の例

図 8 に、送信データ ラインの各ペアを Spartan-3E FPGA の ODDR2 構造 (Spartan-3 FPGA では FDDRRSE) を使用して選択するトランスミッタ DDR の例を示します。この場合、極性を反転する必要のある LVDS 出力に接続されている各ラインにインバータが追加されます。これらのインバータは、前述のようにインプリメンテーション中に出力フリップフロップに組み込まれるので、回路のタイミングは変わりません。

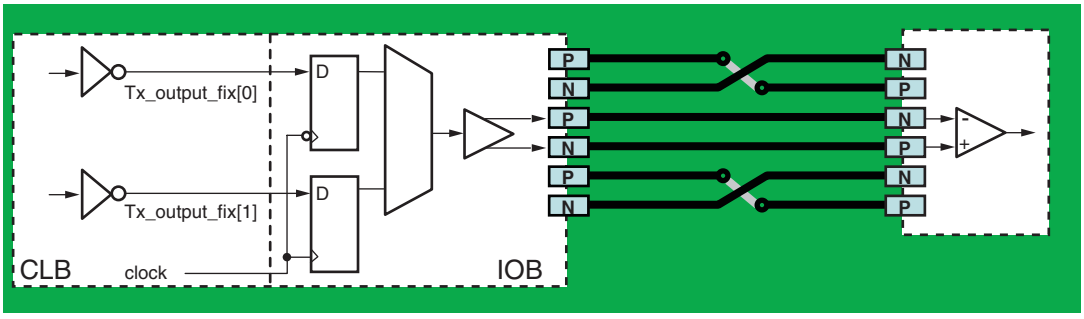

図 8 : DDR レジスタ付きトランスミッタ

次に、先ほどの generate loop の例で DDR レジスタ付きトランスミッタの場合のコードを示します。Spartan-3 デバイスには ODDR2 構造は含まれていないので、コードは多少異なります。詳細は、付属の ZIP ファイルを参照してください (「デザイン ファイル」を参照)。

Verilog :

```
parameter [2:0] SWAP_MASK = 3'b010 ;

genvar i ;
generate
for (i = 0 ; i <= 2 ; i = i + 1)
begin : loop0
OBUFDS #(.IOSTANDARD("LVDS_25"))
obuf_d   (.I(tx_output_reg[i]), .O(dataout_p[i]), .OB(dataout_n[i]));
ODDR2  #(.DDR_ALIGNMENT("NONE")) fd_ioc  (.C0(clkin), .C1(notclk),
.D0(tx_output_fix[i+3]), .D1(tx_output_fix[i]), .CE(1'b1), .R(1'b0),
.S(1'b0), .Q(tx_output_reg[i])) ;
assign tx_output_fix[i]   = tx_output[i]   ^ SWAP_MASK[i] ;
assign tx_output_fix[i+3] = tx_output[i+3] ^ SWAP_MASK[i] ;
end
endgenerate
```

VHDL :

```
constant SWAP_MASK  : std_logic_vector(2 downto 0) := "010" ;

loop0 : for i in 0 to 2 generate
ibuf_d : obufds generic map (IOSTANDARD => "LVDS_25")
            port map    (i => tx_output_reg(i), o => dataout_p(i), oB =>
dataout_n(i));
fd_d  : oddr2  generic map (DDR_ALIGNMENT => "NONE")
           port map (c0 => clkin, c1 => notclock, d0 => tx_output_fix(i),
d1 => tx_output_fix(i+3), ce => '1', r => '0', s => '0', q =>
tx_output_reg(i));
tx_output_fix(i)   <= tx_output(i)   xor SWAP_MASK(i) ;
tx_output_fix(i+3) <= tx_output(i+3) xor SWAP_MASK(i) ;
end generate ;
```

上記のコードで赤の太字で示された数値を変更すると、さまざまなビット幅に簡単に応用できます。

前述のように、DDR を使用する際、ビット操作が重要になることがあります。generate loop の DDR の例では、下位ビットはクロックの立ち下がりエッジで送信され、上位ビットが次の立ち上がりエッジで送信されるバスが生成されます。

デザイン ファイル

このアプリケーション ノートに付属しているさまざまなレシーバおよびトランスミッタのデザイン ファイル例は、すべて Spartan-3 および Spartan-3E デバイス用に記述されています。ザイリンクス Web サイト (xapp491.zip) から、Verilog および VHDL の両方のデザイン ファイルをダウンロードできます。ZIP ファイルに含まれる readme.txt ファイルに、最新の情報が記載されています。

まとめ

LVDS を使用するデザインの設計では、Spartan-3 ジェネレーション FPGA リソースの使用を注意深く計画することにより、PCB のレイアウトが簡潔になり、ボードのシグナル インテグリティが向上します。ここに示す手法は、デバイスに組み込む LVDS レシーバおよび LVDS トランスミッタに適用できます。ただし、入力クロック ピンには適用できないので、極性を正しくしておく必要があります。

改訂履歴

次の表に、この文書の改訂履歴を示します。

日付	バージョン	改訂内容
2006 年 10 月 4 日	1.0	初版リリース