AN61345

EZ-USB® FX2LP™ スレーブ FIFO インターフェースを使った設計

作成者: Rama Sai Krishna V
関連プロジェクト: あり
関連製品ファミリー: CY7C68013A/CY7C68014A/CY7C68015A
ソフトウェア バージョン: なし
関連アプリケーション ノート: AN65209、AN63620

さらにサンプルコードをお求めでしょうか? 以下の通りご対応いたします。

USB ハイスピード コード例の統合リストについては、こちらをクリックしてください。

AN61345 は、スレーブ FIFO インターフェースを使用して FX2LP™を FPGA にインターフェースするためのサンプル プロジェクトを用意しています。実装例で説明したこのインターフェースは、データ収集、産業用の制御と監視、および画像処理などのアプリケーションに高速 USB 接続を追加します。このアプリケーション ノートで提供されたプロジェクトは、Xilinx®社の Spartan®-6 FPGA を用いて実行され、テストされています。

目次

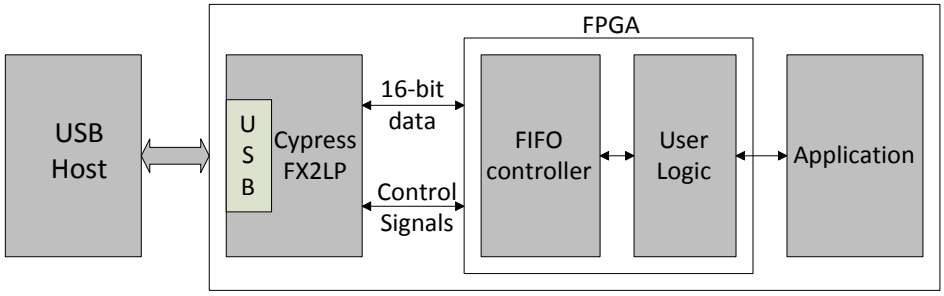
はじめに 1
ハードウェア接続 2
ファームウェア実装 3
FX2LP コードのアーキテクチャ 3
FPGA コード アーキテクチャ 4
データ ループバック 4
ストリーム IN 転送 4
ストリーム OUT 転送 5
シミュレーション波形 6
設計例 7
ZTEX ハードウェアのセットアップ 7
ファームウェアとソフトウェアのコンポーネント 10
操作手順 13
スループット測定 20
関連プロジェクト ファイル 21
Altera® FPGA を使用するためにこの設計を移植する方法 . 21
まとめ 21
ワールドワイドな販売と設計サポート 23

1. はじめに

サイプレスの EZ-USB FX2LP は、USB 2.0 の最大帯域幅まで対応するように設計された柔軟な USB 2.0 の周辺機器コントローラーです。USB 2.0 の 480M ビット/秒の信号レートを最大限に活用するために、FX2LP には、特定のハードウェアが含まれていて USB データをバッファし、MCU、ASIC、FPGA など高帯域幅の様々な内部デバイスに途切れなく接続します。

FX2LP-FPGA インターフェースはデータ収集、産業用の制御と監視、および画像処理などの FPGA ベースのアプリケーションに高速 USB 接続を追加するために実装されます。FX2LP は同期スレーブ FIFO モードで機能し、FPGA はマスターとして動作します。このアプリケーション ノートは、スレーブ FIFO 実装用にサンプルの FX2LP ファームウェアおよび FPGA 実装用にサンプルの VHDL と Verilog のプロジェクトも用意しています。

図 1. FX2LP-FPGA のシステム



FX2LP は 2 つの異なるモードを介して FPGA にインターフェースできます。それは汎用プログラマブル インターフェース (GPIF) モードとスレーブ FIFO モードです。

GPIF モード: このモードでは、FX2LP は外部システムに対してマスターとして動作し、外部システムからデータを読み書きするのに必要な制御信号をすべて生成します。外部システムが FX2LP に対するマスターとして動作するのに十分な高度な処理能力を持っていない場合 (例えば、画像センサーが FX2LP にインターフェースされた USB カメラ アプリケーション) には、GPIF モードのほうが通常は選ばれます。この場合では、インターフェース実装のほとんどの複雑さは FX2LP ファームウェアにあります。

スレーブ FIFO モード: このモードでは、FX2LP にインターフェースする外部システムが必要な読み書きの制御信号を生成する十分な高度な処理能力を持っており、FX2LP に対するマスターとして動作できます。このアプリケーション ノートでは、FX2LP はスレーブ FIFO モードで動作するよう設定されます。

本アプリケーション ノートは、FX2LP における同期の 16 ビット スレーブ FIFO の実装について説明し、FX2LP のスレーブ FIFO のインターフェースに外部の FPGA をインターフェースする方法を示している Verilog と VHDL のサンプル プロジェクトを含んでいます。

注: サンプル プロジェクトは Xilinx 社の Spartan-6 FPGA で実装し試験されています。しかし、本アプリケーション ノートで用意されたコードは標準 Verilog／VHDL コードです。したがって、任意の FPGA に実装するために、これらのファイルを参照して使用できます。プロジェクトを統合し、実装する際に適切な FPGA デバイスを選択する必要があります。

ここでは、スレーブ FIFO のインターフェース、Verilog／VHDL のコーディング、FPGA の統合と実装ツールに慣れていることを前提としています。EZ-USB テクニカル リファレンス マニュアルの第 9 章 (スレーブ FIFO) を参照してください。

2. ハードウェア接続

次の図には、FX2LP を FPGA にインターフェースするために必要なハードウェア接続を示します。

図 2. ハードウェア接続図

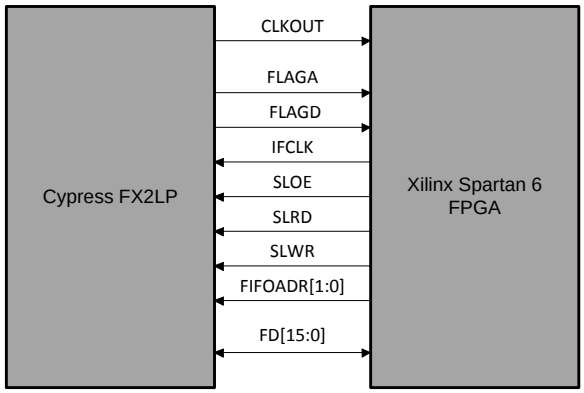


表 1 は図 2 に示したスレーブ FIFO インターフェースの信号を説明します。

表 1. FX2LP と FPGA 間のインターフェース信号

ピン名	説明
SLRD	FIFO からデータを読み出すために、マスターが SLRD ピンをアサートする必要がある
SLWR	FIFO にデータを書き込むために、マスターが SLWR ピンをアサートする必要がある
SLOE	これは FIFO の出力ドライバのイネーブル信号
FIFOADR[1:0]	これらの信号はアクティブなエンドポイントを選択
FD[15:0]	16 ビットのデータ バス
FLAGA/FLAGB/ FLAGC/FLAGD	FIFO がこれらのフラグを用いて状態 (満杯、空、プログラマブル) を示す
IFCLK	同期スレーブ FIFO インターフェースのクロック。本アプリケーション ノートに添付されている設計では、このクロックは 48MHz に設定され、FX2LP にインターフェースする FPGA によって生成
CLKOUT	FX2LP には、12MHz、24MHz、48MHz のクロックを提供する CLKOUT ピンがある

3. ファームウェア実装

FX2LP ファームウェアは、IDE の評価バージョンは FX2LP DVK (CY3684) の内にある Keil 社の uVision 2.0 IDE を使用して開発されました。この節では、スレーブ側 (FX2LP) とマスター側 (FPGA) でのスレーブ FIFO インターフェース実装に必要なコンフィギュレーションについて説明します。

3.1. FX2LP コードのアーキテクチャ

ファームウェアは IN と OUT エンドポイント FIFO の両方用に自動モードを設定します。つまり、IN 転送の場合はパケットが自動的に外部ペリフェラルから USB ドメインに転送され、OUT 転送の場合はその逆です。8051 CPU はパケットの転送を行いません。自動モードまたは手動モードでのエンドポイント FIFO のコンフィギュレーションについての詳細は、EZ-USB テクニカル リファレンス マニュアルのスレーブ FIFO の章を参照してください。バルク転送がこのアプリケーションに使用されているため、エンドポイントをバルクとして設定する必要があります。しかし、最終アプリケーションに基づいて、USB のディスクリプタ ファイルで割り込み、制御、またはアイソクロナスとしてエンドポイント タイプを設定することができます。

スレーブは AUTO モードで動作するので、以下のレジスタ (表 2 に示される) の初期化用のコードを除き、マスターからとマスターへのデータ転送に必要なコードがありません。

表 2. スレーブ FIFO のコンフィギュレーション レジスタ

レジスタ名	レジスタの説明
IFCONFIG	FX2LP をスレーブ FIFO モードに以降させるように IFCONFIG レジスタを設定
PINFLAGSAB/ PINFLAGSCD	FIFO フラグを設定。FIFO フラグは固定モードまたはインデックス モードのいずれかで動作するように設定可能。インデックス モードでは、FLAGA、FLAGB、FLAGC はそれぞれ空、一杯、プログラマブル フラグとして自動的に設定。どれも、FIFOADR [1:0]ラインが特定の瞬間に指すエンドポイントに対応。固定モードでは、プログラム可能なフラグ レジスタに書き込むことにより、任意のエンドポイントに対して 4 つのフラグはそれぞれ空、一杯、プログラマブル フラグとして動作するように設定可能。この設計では、FLAGA は EP2 OUT FIFO に対して空フラグとして、FLAGD は EP6 IN FIFO に対して満杯フラグとして設定
EP2CFG/ EP6CFG	EP2 を OUT、512 バイト、4 重バッファのエンドポイントとして、EP6 を IN、512 バイト、4 重バッファのエンドポイントとして設定
EP2FIFOCFG/ EP6FIFOCFG	FIFO をバイト幅で、AUTO モードで動作するように設定
EP6AUTOINLENH/ EP6AUTOINLENL	EP6 FIFO のバイト数がこれらのレジスタによって指定された値に等しくなると、パケットは自動的に転送。指定された値はエンドポイント ディスクリプタで指定されたパケットの最大サイズ以下

4. FPGA コード アーキテクチャ

FPGA コードの主な機能は、スレーブ FIFO の満杯フラグと空フラグを監視して、それに応じて FIFO に読み書きすることです。

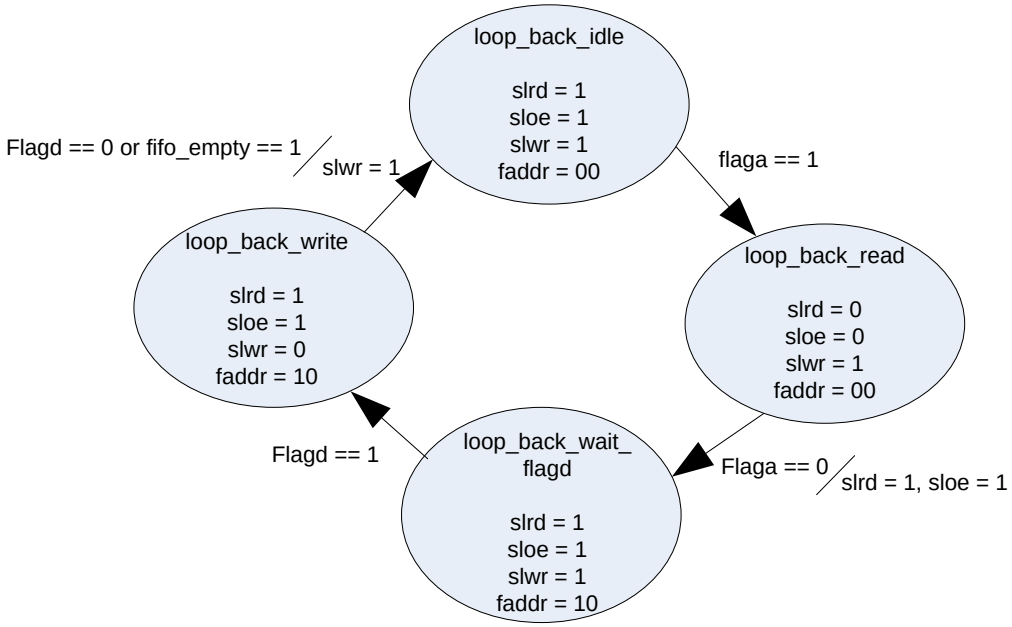
FPGA からのインターフェース クロック (IFCLK) は、FX2LP のスレーブ FIFO インターフェースのセットアップ時間要件を満たすために 180 度シフトしています。

Verilog と VHDL の両プロジェクトは、FX2LP スレーブ FIFO に対するマスターとして動作するよう FPGA の設定方法を示すために提供されています。Xilinx ISE Design Suite はコードの開発に使用されます。

4.1. データ ループバック

FX2LP の EP2 FIFO に書き込まれたデータは FPGA によって読み出され、EP6 FIFO に書き込まれます。関連するプロジェクト フォルダ Loopback (本アプリケーション ノートに添付) はこのプロジェクトを含みます。

図 3. ループバックの状態図

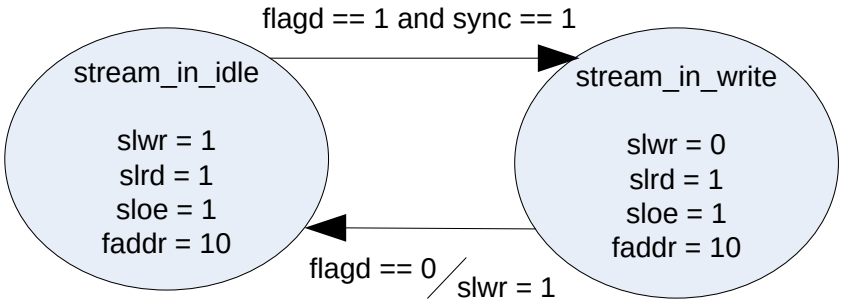


loop_back_idle 状態は SLRD、SLOE、SLWR 信号が HIGH (デアサート) のアイドル状態です。EP2 FIFO の空フラグ (FLAGA) が HIGH になると、ステート マシンは loop_back_read 状態に移行します。loop_back_read 状態では、FPGA は SLRD と SLOE をアサートすることでデータを読み出します。EP2 FIFO の空フラグが LOW になると、ステート マシンは loop_back_wait_flagd 状態に移行します。この遷移中に、SLRD と SLOE 信号はデアサートされます。loop_back_wait_flagd 状態では、FIFO のアドレス ラインは EP6 をアドレス指定するために駆動されます。EP6 の満杯フラグ (FLAGD) が LOW になるまでこの状態のままです。FLAGD が HIGH になると、ステート マシンは loop_back_write 状態に移行します。この状態では、FPGA は SLWR 信号をアサートすることで EP6 FIFO に同じデータを書き込みます。

4.2. ストリーム IN 転送

FPGA は EP6 の満杯フラグ (FLAGD) と同期 (FX2LP の PC0) 信号を監視します。FLAGD と同期信号の両方が HIGH の場合、FPGA は FIFO 内に増分データを書き込み続けます。EP6 FIFO にデータを書き込んでいる間、FPGA は満杯フラグがアサートされると直ぐに書き込みを一時停止し、そのフラグがデアサートされると書き込みを再開します。関連するプロジェクト フォルダ Stream IN はこのプロジェクトを含みます。

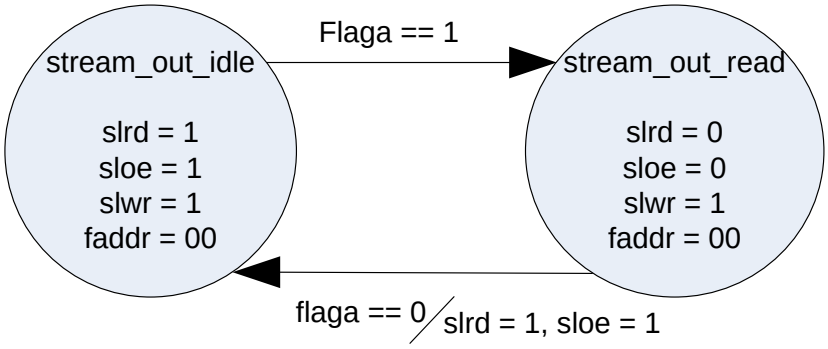
図 4. データ ストリーム IN の状態図



このステート マシンには 2 種類の状態があります: stream_in_idle と stream_in_write。stream_in_idle 状態は SLWR が HIGH であるアイドル状態です。EP6 の満杯フラグ (FLAGD) が LOW (アサート) である限り、ステート マシンは stream_in_idle 状態のままです。一杯フラグと同期 (FX2LP の PC0) 信号が HIGH になると、state stream_in_idle 状態から stream_in_write 状態に移行します。stream_in_write 状態では、FPGA は継続的に EP6 FIFO に増分データを書き込みます。FLAGD が LOW になると、ステート マシンは stream_in_idle 状態に戻ります。SLWR 信号はこの遷移中にデアサートされます。

4.3. ストリーム OUT 転送

図 5. ストリーム OUT の状態図



このステート マシンには 2 種類の状態があります: stream_out_idle と stream_out_read。stream_out_idle 状態は SLRD と SLOE が HIGH であるアイドル状態です。EP2 の空フラグ (FLAGA) が LOW (アサート) である限り、ステート マシンは stream_out_idle 状態のままです。EP2 の空フラグが HIGH になると、stream_out_idle 状態から stream_out_read 状態に移行します。stream_out_read 状態では、FPGA は EP2 FIFO からデータを読み出し続けます。FLAGA が LOW になると、ステート マシンは stream_in_idle 状態に戻ります。SLRD と SLOE 信号はこの遷移中にデアサートされます。関連するプロジェクト フォルダ Stream OUT はこのプロジェクトを含みます。

5. シミュレーション波形

この節では、異なるモード (ストリーム IN とストリーム OUT) の下でスレーブ FIFO インターフェース信号のシミュレーション波形を示します。これらの波形は Xilinx 社の ISim ツールを使って取り込まれています。

図 6. データ ストリーム IN – バースト書き込み開始からの波形

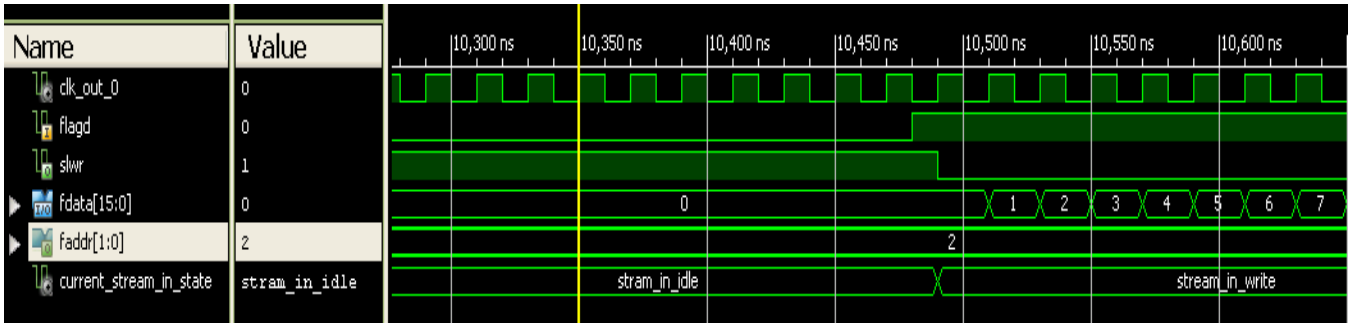


図 6 には、FLAGD または満杯フラグがデアサートされた時の SLWR アサートを示します。

図 7. データ ストリーム IN、バースト書き込み終了

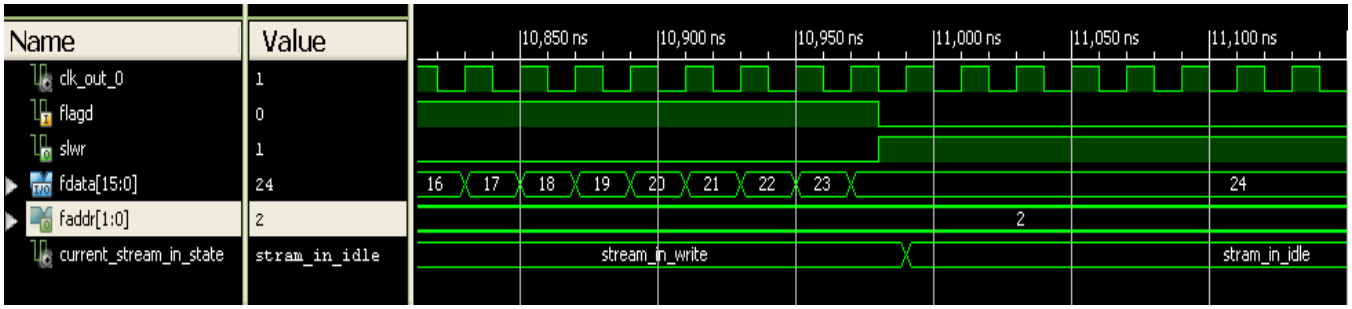


図 7 に示すように、満杯フラグがアサートされた後、SLWR はデアサートされます。

図 8. データ ストリーム OUT、バースト読み出し開始

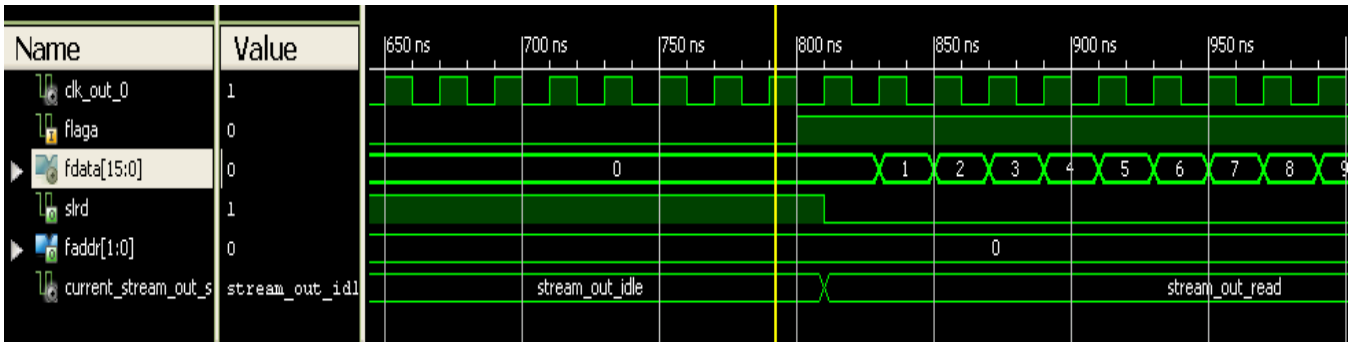


図 8 には、FLAGA (EP2 EF) がデアサートされた時の SLRD アサートを示します。

図 9. データ ストリーム OUT、バースト読み出し終了

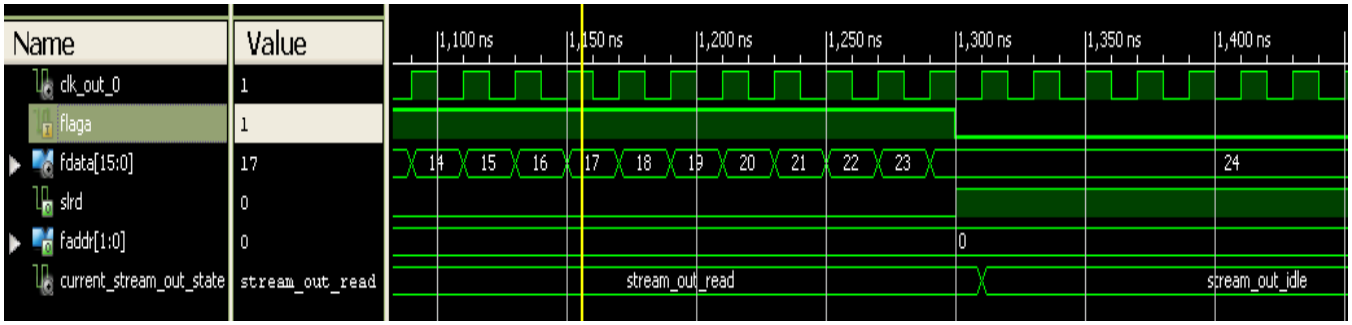


図 9 に示すように、FLAGA がアサートされると、SLRD 信号はデアサートされます。

6. 設計例

本節では、Xilinx 社の Spartan-6 FPGA が同期スレーブ FIFO インターフェースを介して FX2LP に接続する完全な設計例を提供します。

Spartan-6 準拠の例が本アプリケーション ノートに提供されています。この例はストリーム IN、ストリーム OUT、ループバック転送を実行します。FPGA ビット ファイルの動作は「FPGA コード アーキテクチャ」節で説明されています。

この設計を実現するために使ったハードウェア、ファームウェア、ソフトウェアのコンポーネントは次の節で説明します。

6.1. ZTEX ハードウェアのセットアップ

ZTEX FX2LP - FPGA モジュール 1.11 (図 11 に提示) は実験用基板 1.3 (図 12 に提示) と共に使用されます。実験用の基板は電源供給と JTAG ケーブルを必要とします (FPGA を設定するためです) 。CON1 (実験用基板 1.3 上) は、4.5V～16V の電源電圧に対して直径 2.1mm のセンター ピン (+) と直径 5.5mm のバーレル (-) のある標準 DC 電源ジャックです。CON9 (実験用基板 1.3 上) は、Xilinx 社によって標準化された 14 ピン、2.0mm ピッチの JTAG コネクターです。「1」と呼ばれる極性キー(ホール) は両方のモジュールと実験用基板に存在します。その極性キーが同じコーナーに位置するように ZTEX FX2LP - FPGA モジュール 1.11 は実験用基板 1.3 に搭載される必要があります。両方の基板上にある極性ホールを識別するために、上記のリンクにあるレイアウト図を参照してください。

図 13 に示すように FPGA モジュール 1.11 を実験用基板 1.3 に取り付けます。5V または 12V 電源で基板を電源投入し、ミニ USB ケーブルを使用してホスト PC に接続します。

プラットフォーム USB ケーブル II (JTAG アダプター) は ZTEX FX2LP - FPGA モジュール 1.11 に存在する Xilinx 社の Spartan-6 FPGA を設定するために使用されます。「ChipScope Pro」 (次のステップで説明) はこの JTAG アダプターと互換性のあるソフトウェアです。

ハードウェア接続図は図 10 に示します:

図 10. ハードウェア接続 (ZTEX 基板)

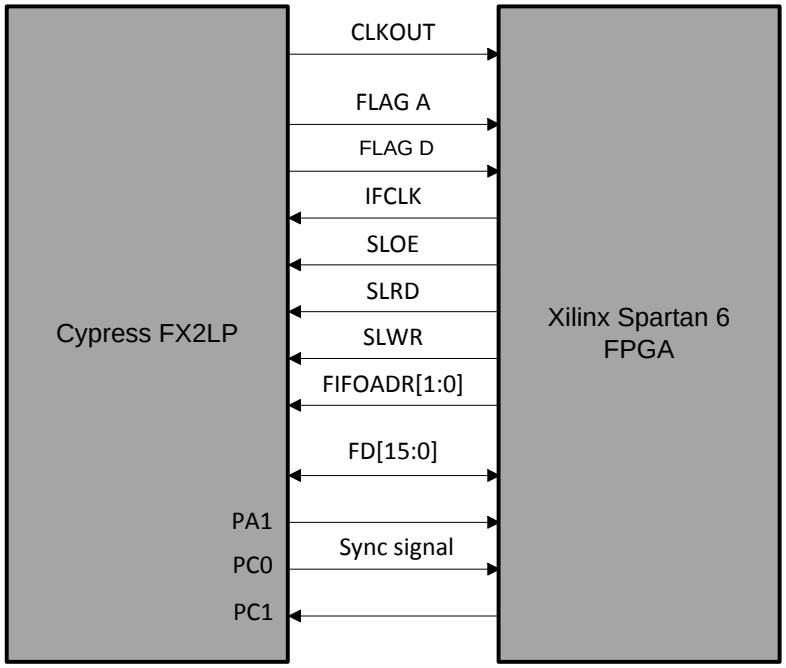


図 11. ZTEX FX2LP – FPGA モジュール 1.11



図 12. ZTEX 実験用基板 1.3

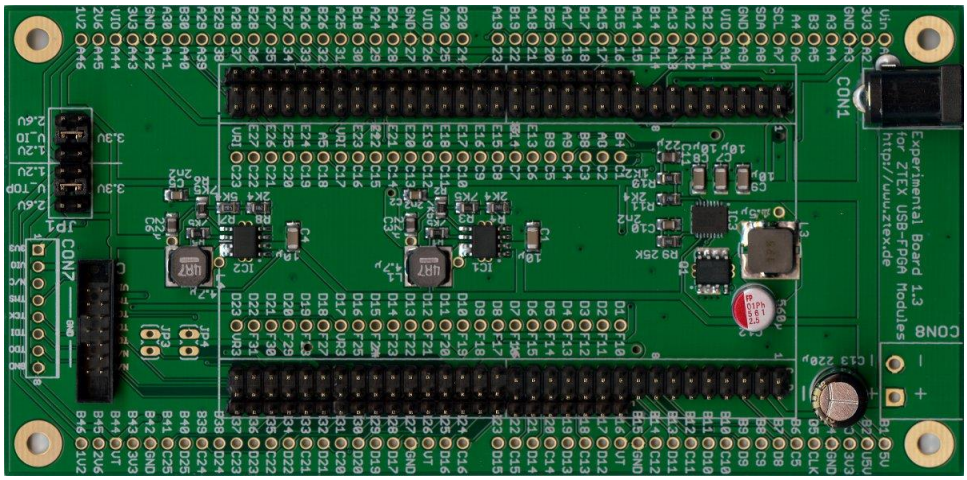
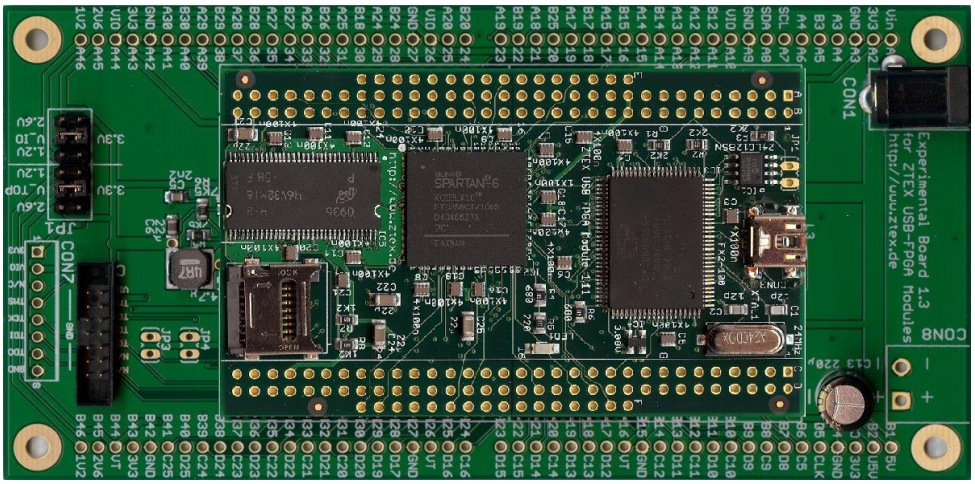


図 13. 実験用基板の上部にある ZTEX FX2LP-FPGA モジュール

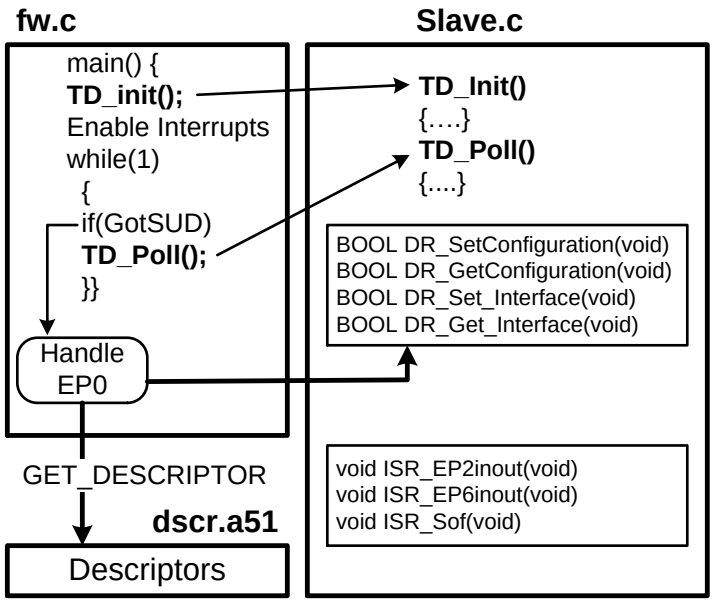


6.2. ファームウェアとソフトウェアのコンポーネント

6.2.1. FX2LP のファームウェア

図 14 は、FX2LP のファームウェア モジュール間のやり取りを示しています。

図 14. スレーブ FIFO インターフェース用の FX2LP ファームウェア



Fw.c ファイルには **main** 関数が含まれています。エニュメレーションなどの USB メンテナンスの多くを実行し、カスタマイズが必要なたびにアプリケーション コード (Slave.c) 内の具体的に名付けられた外部関数を呼び出します。Fw.c ファイルは殆どユーザーによる変更を必要としません。色々な準備手順を整えた後、Slave.c に用意してある TD_init と呼ばれる外部関数を呼び出します(接頭辞 TD は「タスク ディスパッチャ」の略です) 。次に、CONTROL エンドポイント 0 に対する SETUP パケットの着信を確認する無限ループに移行します。ループは USB 中断イベントに応じても確認しますが、これはスレーブ FIFO アプリケーションでは使用されません。ループを通り抜ける度に、Slave.c ファイルに用意してある外部 TD_Poll 関数を呼び出します。 このアプリケーションでは、TD_Poll 関数は FPGA と FX2LP 間のデータ転送の同期化を処理します。エンドポイント FIFO が自動モードに設定されたため、ひとたびデータ転送が開始されると、この関数は何もしません。

全ての USB の周辺装置はその CONTROL エンドポイントを介して 2 種類の要求を受信します: エニュメレーションとオペレーショナル。

エニュメレーション

USB デバイスが接続されると、ホスト PC は複数の GET_DESCRIPTOR 要求を送信して、エニュメレーションと呼ばれるプロセスの一部としてデバイスの種類とその要件を見つけます。fw.c コードはこれらの要求を捉えて、dscr.a51 ファイルに格納された値を使用してそれらを処理します。

USB フレームワークを使用する利点は、コードがテスト済みであり、USB の「第 9 章」の要件を満たすことが確認されていることです。第 9 章はデバイスの要求 (EP0 を介する) とそれらの適切な応答を説明する USB 仕様の章です。

オペレーショナル

ユーザー コードを必要とするどんな場合でも、fw.c はユーザーが Slave.c ファイルに用意している DR 接頭辞 (デバイス リクエスト) 付きの具体的に名付けられた外部関数を呼び出します。スレーブ FIFO のような簡単なアプリケーション用には、1 つのコンフィギュレーションと 1 つのインターフェースのみがあります。それで、図 14 の 2 つの DR_Set-Get 関数のペアは単にホストに送信された SET 値を格納し、ホストが GET 要求を発行するとそれらの値をエコーバックします。より複雑なコンフィギュレーションの場合は、これらの DR 呼び出し (「フック」) を使用してカメラの解像度を変更したり、2 つの異なるインターフェースに要求を送信したりすることができます。

この節の残りの部分は、スレーブ FIFO アプリケーションを実行するためにユーザー コードを必要とするファイルの 3 つの部分について説明します。

TD_Init

この関数は次のことを行います:

- 8051 クロックを 48MHz に設定します。
- 内部 48MHz クロックを使うようにスレーブ FIFO インターフェースを設定します。

```
IFCONFIG = 0xE3; //Internal clock, 48 MHz, Slave FIFO interface
SYNCDELAY;
```

- EP2 を BULK-OUT エンドポイントとして、EP6 を BULK-IN エンドポイントとして設定します。両方は 4 重バッファであり、512 バイトの FIFO を使用します。EP4 と EP8 はこの設計に使用されないため無効にされます。

```
EP2CFG = 0xA0; //out 512 bytes, 4x, bulk
SYNCDELAY;
EP6CFG = 0xE0; //in 512 bytes, 4x, bulk
SYNCDELAY;
EP4CFG = 0x02; //clear valid bit
SYNCDELAY;
EP8CFG = 0x02; //clear valid bit
SYNCDELAY;
```

- FIFO をリセットします。
- 16 ビットのインターフェースで自動 OUT モードにエンドポイント 2 FIFO を設定し、16 ビットのインターフェースで自動 IN モードにエンドポイント 6 を設定します。

```
EP2FIFOCFG = 0x00;// AUTOOUT=0, WORDWIDE=1
// core needs to see AUTOOUT=0 to AUTOOUT=1 switch to arm endpoints
SYNCDELAY;
EP2FIFOCFG = 0x11;// AUTOOUT=1, WORDWIDE=1
SYNCDELAY;
EP6FIFOCFG = 0x0D;// AUTOIN=1, ZEROLENIN=1, WORDWIDE=1
SYNCDELAY;
```

- FIFO フラグ出力を設定する。FLAGA は EP2 OUT FIFO に対して空フラグとして、FLAGD は EP6 IN FIFO に対して満杯フラグとして設定されます。

```
PINFLAGSAB = 0x08;    // FLAGA - EP2EF
SYNCDELAY;
PINFLAGSCD = 0xE0;    // FLAGD - EP6FF
SYNCDELAY;
```

- PA1 ピン (FPGA の PROG_B ピンに接続) を HIGH に設定します。これは FPGA の JTAG コンフィギュレーションを有効にするには必要です。ZTEX ハードウェアのセットアップで新しい FX2LP ファームウェアを使用するなら、FX2LP の PA1 ピンを HIGH に設定することを確認してください。

```
OEA|=0x02; //Declare PA.1 as output
SYNCDELAY;
IOA|=0x02; //output 1 on PA.1
SYNCDELAY;
```

- PC0 ピンを出力ピンとして、PC1 ピンを入力ピンとして設定します。PC0 は FPGA と FX2LP 間のデータ転送を同期化するために使用されます。PC1 は、スレーブ FIFO インターフェース クロック (IFCLK) を提供する FPGA の用意ができている事を知るのに使用されます。

```
OEC|=0x01; //PC.0 as output (SYNC signal)
SYNCDELAY;
IOC|=0x00; //output 0 on PC.0...SYNC signal is LOW
SYNCDELAY;
OEC&=0xFD; //PC.1 as input (Clock changing signal)
SYNCDELAY;
```

TD_Poll

TD_Poll は fw.c ファイルに存在する無限ループで呼び出されます (図 14 を参照) 。このスレーブ FIFO の設計では、TD_Poll は単にインターフェース クロック (IFCLK) のソースを変更するために使用されます。

本アプリケーション ノートに添付されているサンプル プロジェクトは FPGA からインターフェース クロック (IFCLK) を取得するために設計されています。ZTEX ハードウェア基板を使用している場合、FPGA の PROG_B ピン (PA1 に接続) を HIGH にセットしてそれを設定するために、まず FX2LP をプログラムする必要があります。しかし FX2LP ファームウェアは、インターフェース クロックを供給するように FPGA を設定する前に IFCONFIG レジスタを設定して外部クロックで動作することはできません (ファームウェアが IFCONFIG.7=0 (IFCLK は外部デバイスから提供) に設定する前に、外部 IFCLK ソースが存在する必要があることに注意してください)。それで、この条件を満たすために、最初に IFCONFIG レジスタが内部 IFCLK を使用できるように設定され、その後 IFCONFIG.7 は、FPGA が起動してビットストリームで走ると、FPGA から IFCLK を取得するために変更されます。FX2LP の PC1 はこの変更に使用されます。FX2LP の PC0 は FX2LP と FPGA 間の同期信号として使用されます。以下のコードは、クロック ソースを内部から外部に変更するために使用されます。

```
if(!(IOC & 0x02))
        {
        done_frm_fpga = 1;
        }
if((done_frm_fpga) && (IOC & 0x02))
        {
        IFCONFIG = 0x03; //external clock input, Slave FIFO interface
        SYNCDELAY;

        IOC|=0x01; //output 1 on PC.0...SYNC signal is HIGH
        SYNCDELAY;
        done_frm_fpga = 0;
        }
```

FX2LP ファームウェアは本アプリケーション ノートの添付物の一部です。ビットストリームを FPGA にダウンロードする前に、コントロール センター ユーティリティを使用してそのファームウェアを構築し、FX2LP にダウンロードします。これらの手順は「操作手順」節で説明されています。

6.2.2. コントロール センター ユーティリティ

サイプレスのコントロール センター ユーティリティを使用してファームウェアをダウンロードし、FX2LP への BULK 転送を実行します。サイプレスの SuiteUSB 開発ツールをインストールすることで、コントロール センター ユーティリティを利用できます。

6.2.3. ChipScope Pro

「ChipScope Pro」ソフトウェア (Xilinx ISE Design Suite 14.1 と共に提供) は Xilinx 社の Spartan-6 FPGA を設定するために使用されます。Xilinx ISE Design Suite 14.1 の評価バージョンはライセンス認証なしに 30 日間使用できます。または、Xilinx 社の Spartan-6 FPGA を設定する他の適切な方法が採用されます。ChipScope Pro を使用して Xilinx 社の Spartan-6 FPGA を設定する手順については次の節で説明します。

「ChipScope Pro」ソフトウェアの他のバージョンが使用されている場合、そのバージョンが Xilinx 社の Spartan-6 FPGA に対応するかを確認してください。

6.3. 操作手順

6.3.1. CYUSB ドライバの割り当て

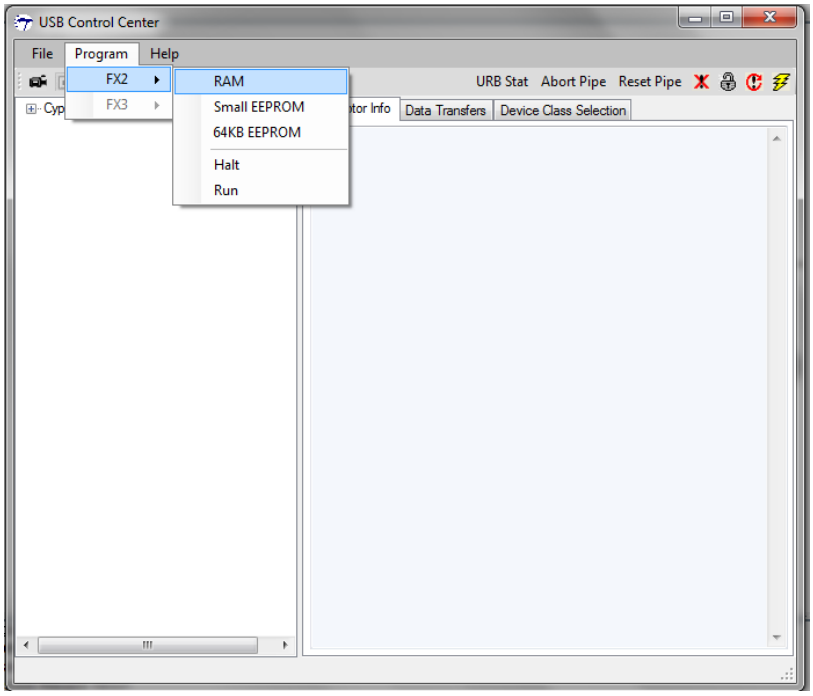
ZTEX FPGA モジュールが USB を介してホストに接続された後、デバイスの VID と PID を確認してください。ファームウェアが基板上にある EEPROM にフラッシュされていない場合、デバイスの VID と PID は、それぞれ 0x04b4 と 0x8613 です。ZTEX SDK で構築されたファームウェアがフラッシュされた場合、それらは 0x2214 と 0x0100 です。これら VID と PID の値を CYUSB.inf ファイルに追加してください。「CYUSB.sys」ドライバをこのデバイスに割り当てます。これはデバイスがコントロール センター ユーティリティに表示されるには必要です。

カスタム VID と PID を持つデバイス用に CYUSB.sys を割り当てる詳細については、CyUSB.pdf の文書を参照してください。サイプレスの SuiteUSB 開発ツールをインストールすると、CyUSB.pdf は次のパスにあります: C:\Cypress\Cypress Suite USB 3.4.7\Driver (インストール パスで異なる場合があります)。

6.3.2. FX2LP ファームウェアのダウンロード

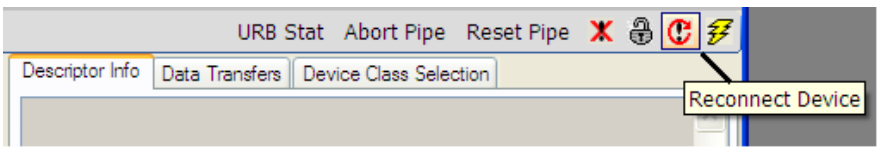
コントロール センター ユーティリティを開き、FX2LP ファームウェアをダウンロードします (**Program>FX2>RAM** をクリックし、関連するプロジェクト フォルダにある slave.hex ファイルに移動します)。

図 15. ファームウェア イメージを FX2LP RAM にダウンロード



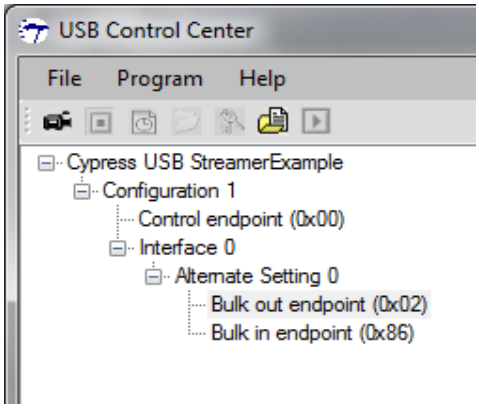
デバイスの初期設定の VID と PID 値が 0x2214 と 0x0100 である場合、slave.hex をダウンロードした後、次の図に示されるボタンを押してデバイスを再接続する必要があります。

図 16. コントロール センターの再接続ボタン



FX2LP は図 17 に示すようにエニュメレートします。エンドポイント 2 はバルク OUT エンドポイントとして、エンドポイント 6 はバルク IN エンドポイントとして設定されます。

図 17. slave.hex をダウンロードした後のコントロールセンターでの FX2LP デバイス表示



6.3.3. FPGA ビットストリームのダウンロード

JTAG ケーブルを ZTEX 実験用基板 1.3 上に用意された JTAG コネクターに接続します。前の手順後に、FPGA の PROG_B ピン (FX2LP の PA1 ピンに接続) は HIGH になり、FPGA の JTAG コンフィギュレーションを有効にするために必要とされます。次に、以下の図に示すように、Spartan-6 準拠のビットストリーム (本アプリケーション ノートで提供されたストリーム IN、ストリーム OUT またはループバック ビットストリーム) で FPGA を設定します。

図 18. ChipScope Pro

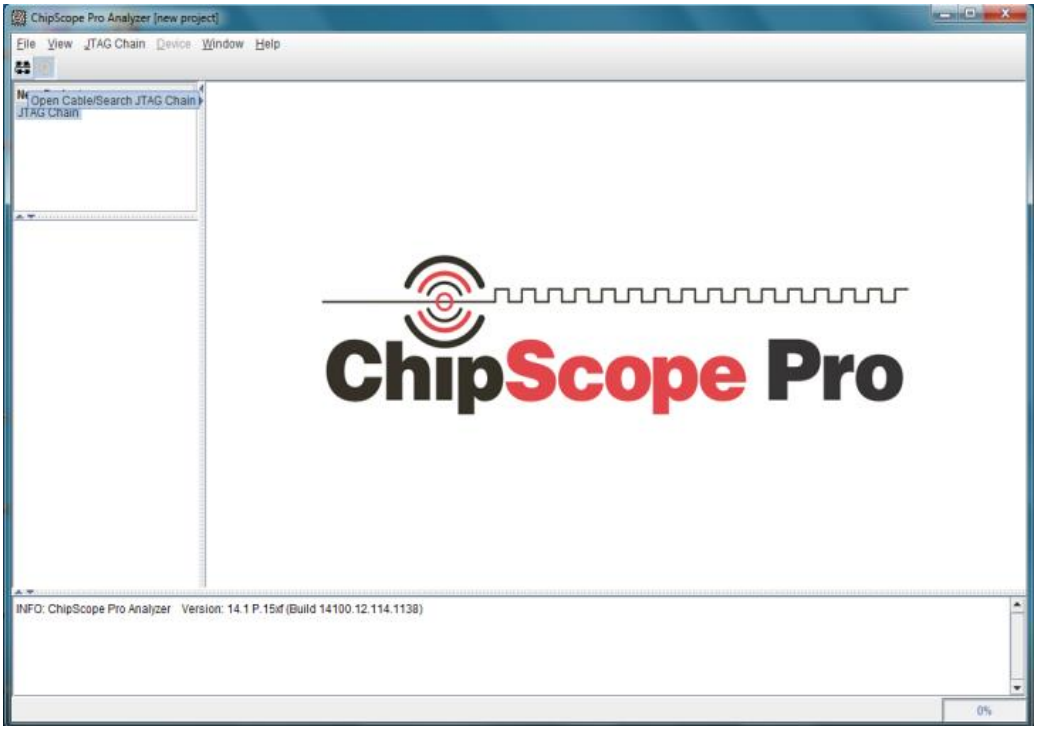


図 19. ChipScope Pro を使用して FPGA を設定

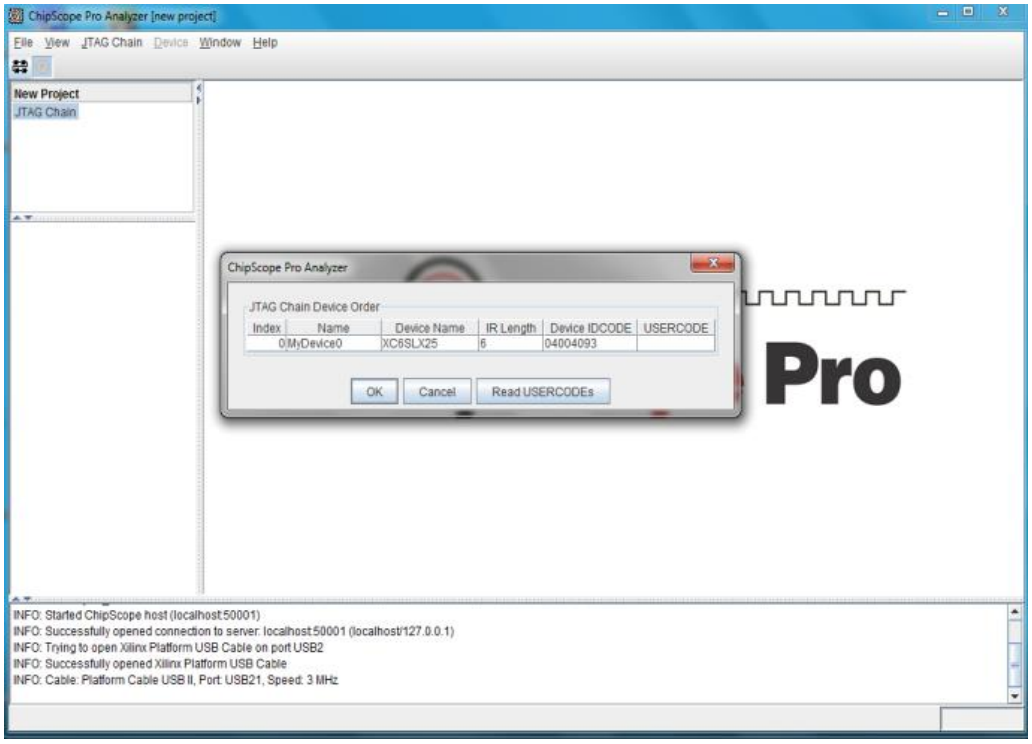


図 20. ChipScope Pro を使用して FPGA を設定

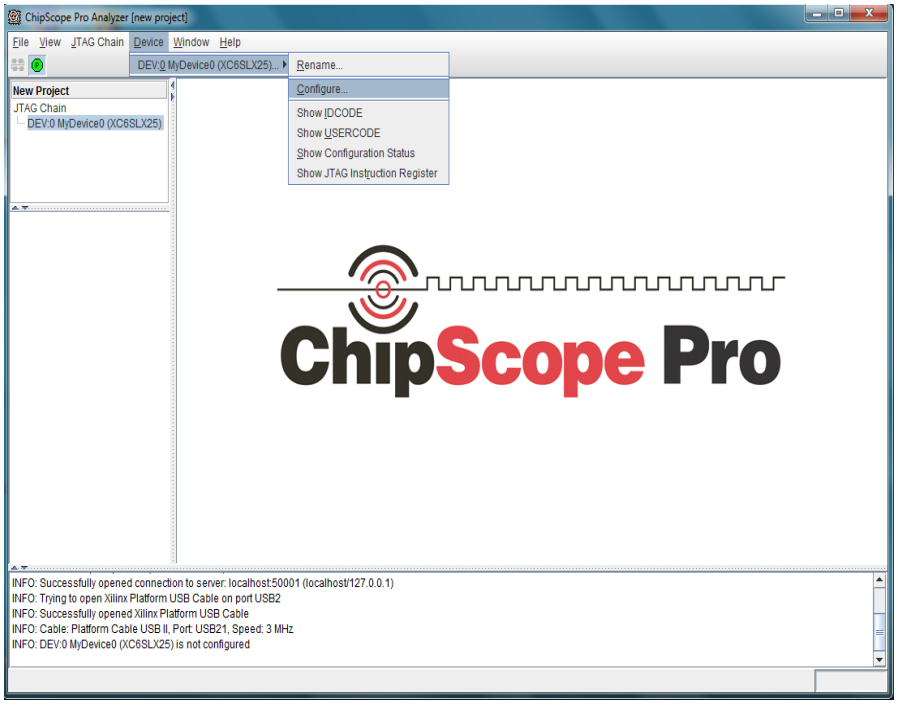
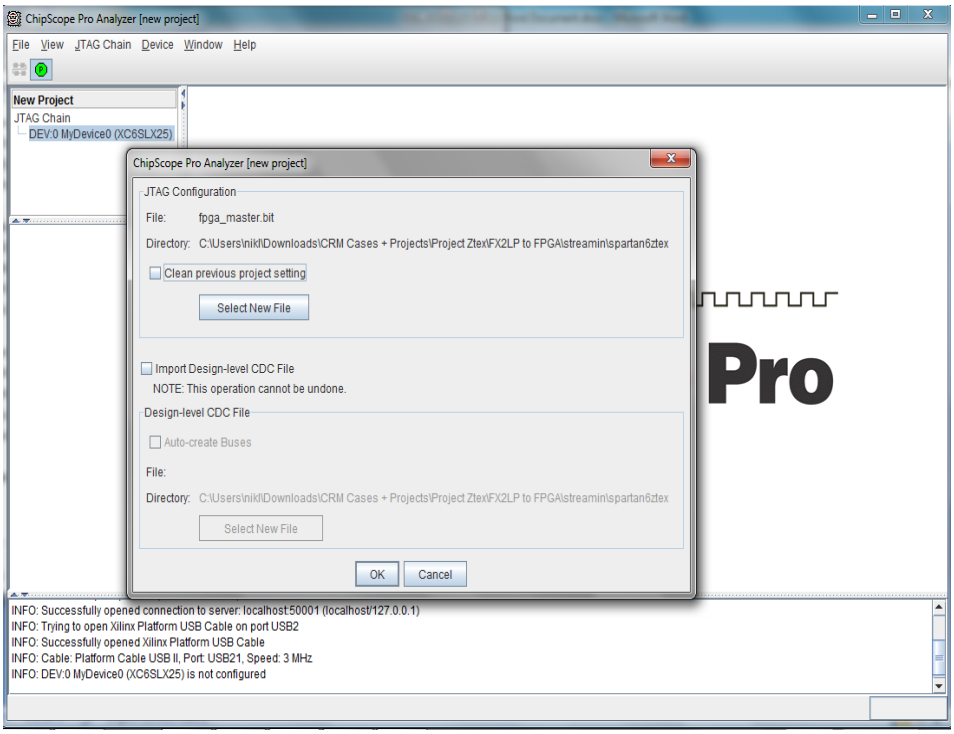


図 21. ChipScope Pro を使用して FPGA を設定



6.3.4. ループバック ビットストリームの結果を検証

FPGA がループバック ビットストリームで設定された場合 (ステップ 3)、エンドポイント EP2OUT と EP6IN にループバックを実行します。ホストから EP2 に送信されたデータは FPGA によって読み出され、EP6 エンドポイント FIFO に書き込まれます。

ループバック動作を検証するために、コントロール センターのウィンドウで **Bulk Out Endpoint (0x02)**を選択し、**Transfer File-OUT** ボタンをクリックし、512_count.hex (添付ファイルにある) を閲覧してデータ転送を実行します。そして、**Transfer Data-IN** をクリックすることで EP6IN バッファを読み出して、書き込まれたデータを検証します。これらの手順は次の図に示します。

図 22. OUT エンドポイント 2 にファイルを転送

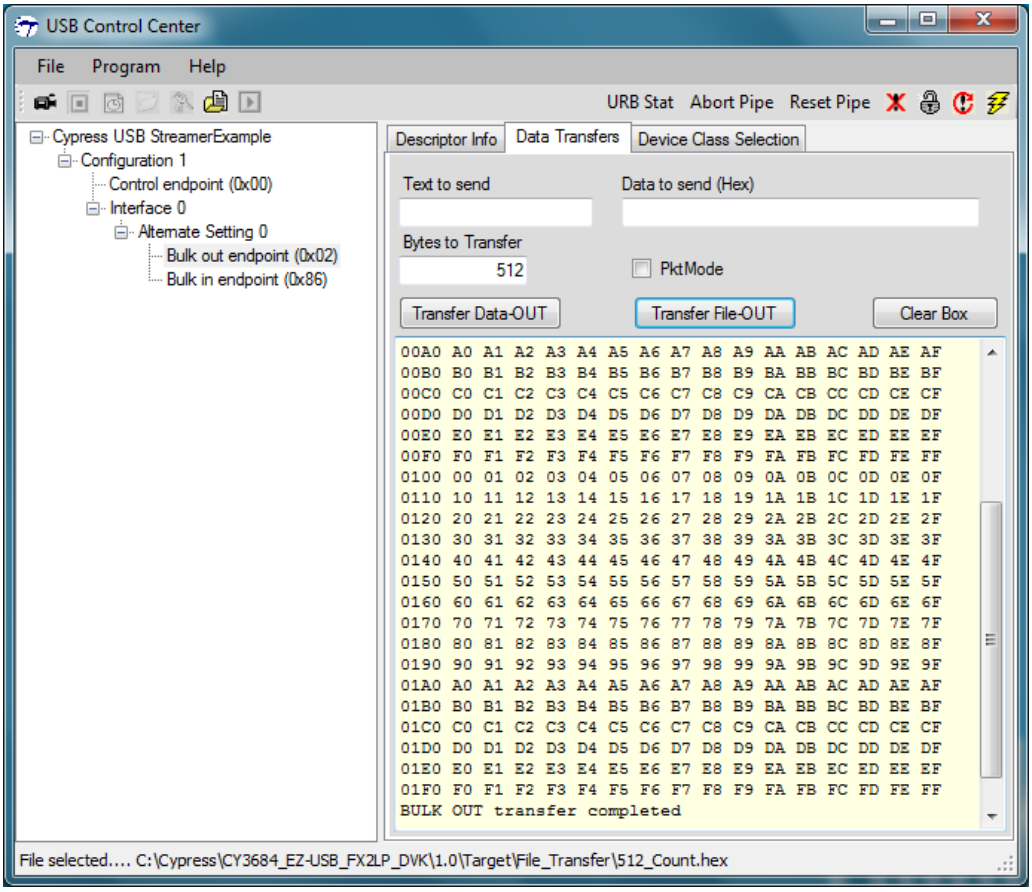
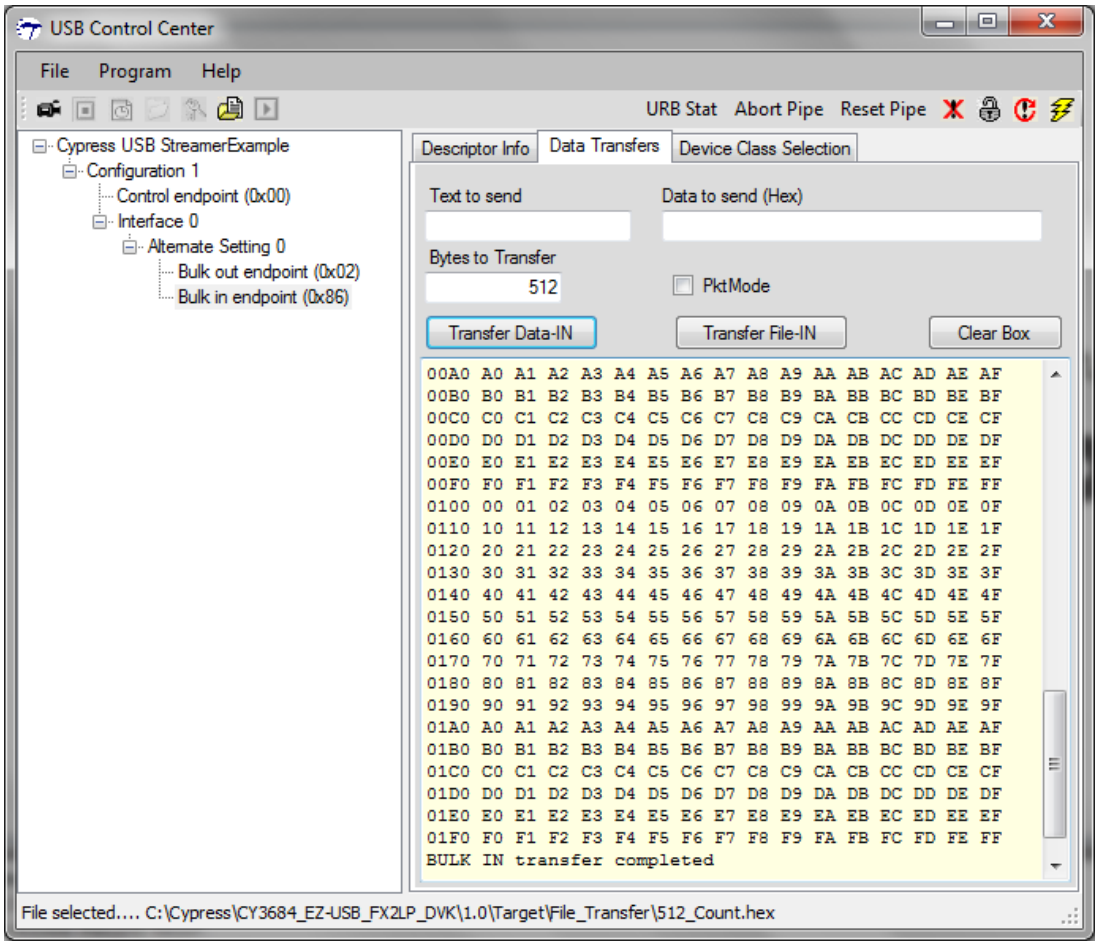


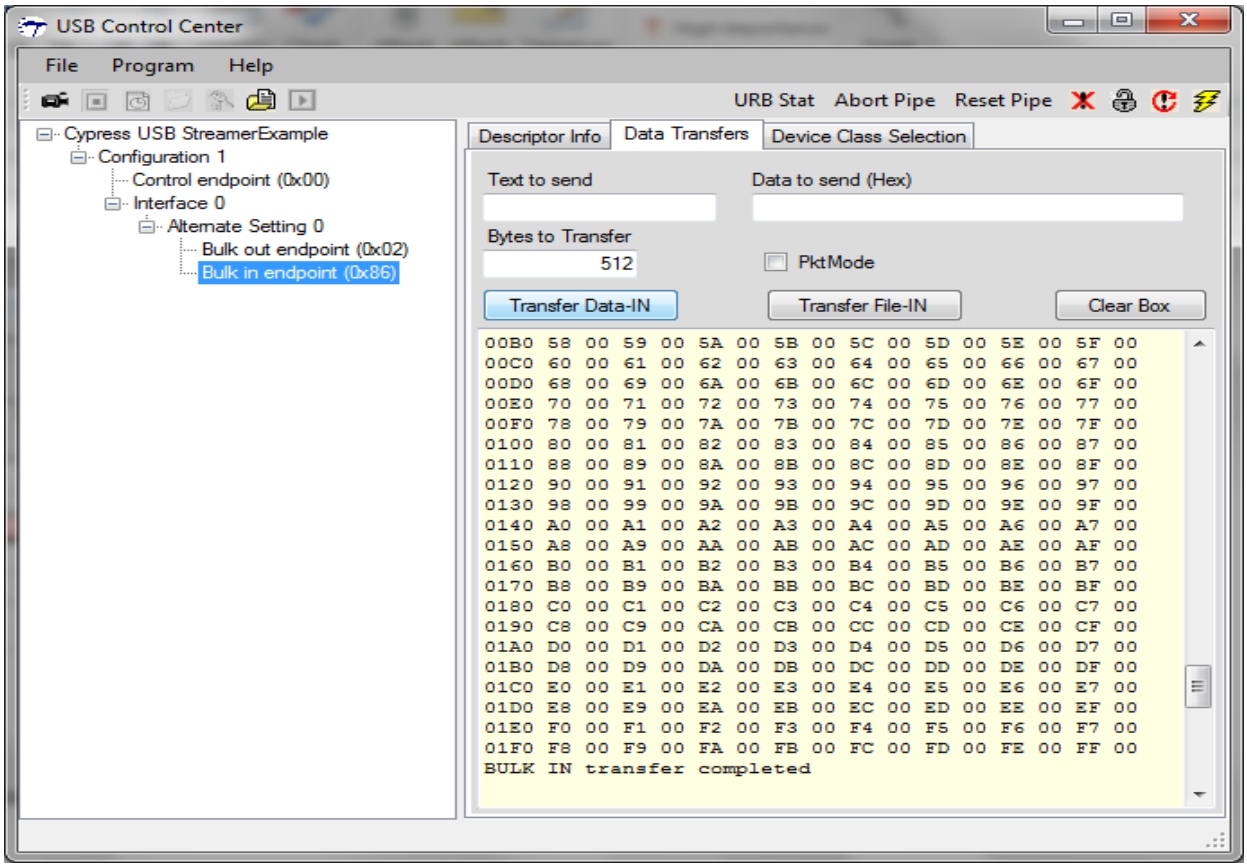
図 23. IN エンドポイント 6 から同じデータを読み出す



6.3.5. ストリーム IN ビットストリームの結果を検証

FPGA がストリーム IN ビットストリーム (ステップ 4) で設定された場合、FPGA をマスター モードにします。FPGA はその後、増分データを FX2LP のエンドポイント 6 IN に送信します。ストリーム IN 動作を検証するために、コントロール センターのウィンドウ内で **Bulk in endpoint (0x86)**を選択し、**Transfer Data-IN** ボタンをクリックします。次の図に示すようにデータを見ることができます。

図 24. IN エンドポイント 6 からデータを読み出す



6.3.6. ストリーム OUT ビットストリームの結果を検証

FPGA がストリーム OUT ビットストリーム (ステップ 4) で設定された場合、FPGA をマスター モードにします。FPGA はその後 FX2LP の OUT エンドポイント 2 からデータを読み出します。USB コントロール センターを使用してエンドポイント 2 に、任意の数の OUT 転送を実行するか、またはエンドポイント 2 にある OUT 転送を検証するためにストリーマ アプリケーションを走らせることができます。FPGA は FX2LP からのフラグに基づいて OUT エンドポイント 2 からデータを読み出します。FPGA は単に受信したデータを無視し、FX2LP の OUT エンドポイント 2 からの更なるデータのために待機します。

6.4. スループット測定

FPGA はデータ ストリーミング用にコードで設定され、インターフェースのスループットはサイプレスの SuiteUSB 3.4 に含まれるストリーマ アプリケーションで測定されます。約 39MB/s のスループットは、CyUSB.sys のドライバ (バージョン 3.4.7) に結合されたデバイスを使って Windows 7 (64 ビット) を走らせる Intel 7 シリーズ/c216 のチップセット ファミリーで測定されます。ストリーミング デバイスのスループット評価の詳細は、「AN4053 - Streaming Data Through Isochronous/Bulk Endpoints on EZ-USB FX2™ and EZ-USB FX2LP™」に記載されています。

図 25. ストリーム IN 転送用に測定されたスループット

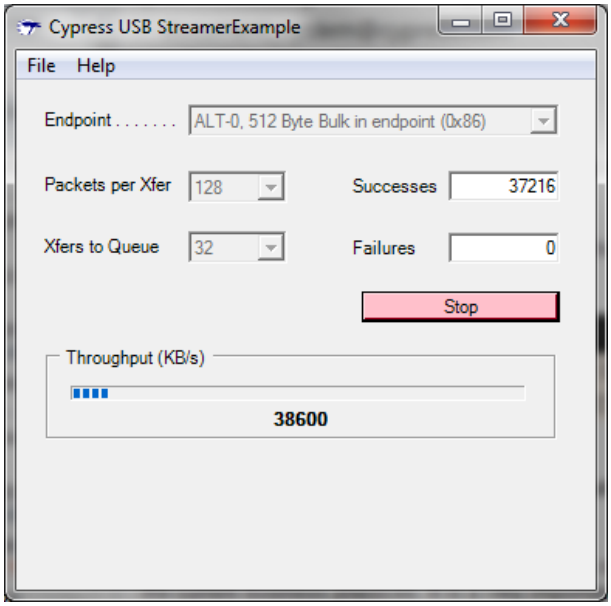
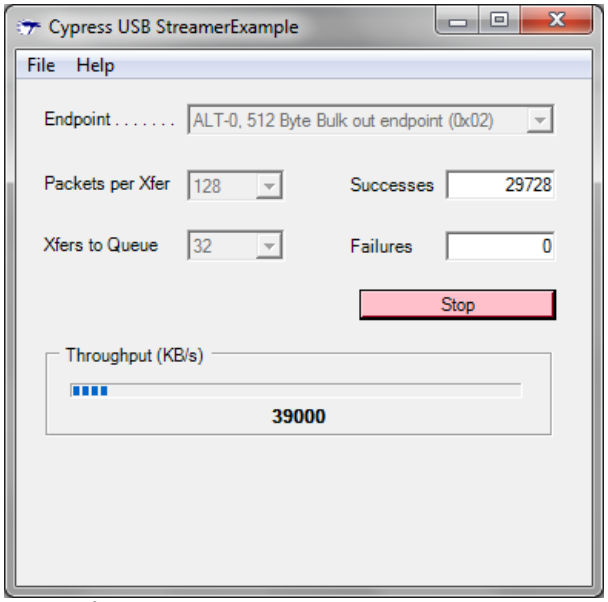


図 26. ストリーム OUT 転送用に測定されたスループット



注: これらのスループット数は、ストリーマ アプリケーションで **256「Packets per Xfer」**と **64「Xfers to Queue」**を選択することで測定されます。

7. 関連プロジェクト ファイル

表3は、本アプリケーション ノートに添付されているファイルについて説明します。

表 3. アプリケーション ノートのファイルの説明

ファイル名／フォルダ名		説明
FX2LP ファームウェア		FX2LP ファームウェア プロジェクトのソースファイル
FPGA Source Code_Verilog	ループバック	データ ループバックを実行するための FPGA Verilog ソースコード
	ストリーム IN	ストリーム IN データ転送を実行するための FPGA Verilog ソースコード
	ストリーム OUT	ストリーム OUT データ転送を実行するための FPGA Verilog ソースコード
FPGA Source Code_VHDL	ループバック	データ ループバックを実行するための FPGA VHDL ソースコード
	ストリーム IN	ストリーム IN データ転送を実行するための FPGA VHDL ソースコード
	ストリーム OUT	ストリーム OUT データ転送を実行するための FPGA VHDL ソースコード
512_count.hex		ファイル転送を実行するための 512 バイトのデータ
Firmware_SDCC		SDCC コンパイラを使用して作成された FX2LP ファームウェア プロジェクトのソースファイル
Firmware_SDCC/Release		SDCC コンパイラを使用して FX2LP ファームウェア プロジェクトを構築した後に生成されたファイル
Readme_SDCC.pdf		SDCC コンパイラを使用してプロジェクトを作成する手順を説明する文書
Sdccman.pdf		SDCC コンパイラについて説明する SDCC マニュアル

8. Altera® FPGA を使用するためにこの設計を移植する方法

Altera FPGA を使用するためにこの設計を移植する際に次の手順を行う必要があります:

1. Altera ツールを使用して生成される基本要素:
   - クロック生成用の PLL
   - FX2LP にインターフェース クロックを提供する DDR
2. FPGA への FX2LP スレーブ FIFO インターフェース信号のピン マッピング。

9. まとめ

本アプリケーション ノートは、スレーブ FIFO モードに設定された FX2LP で FPGA と FX2LP 間のインターフェースをセットアップする方法を説明しています。関連するプロジェクトには、スレーブ FIFO モードで FX2LP を初期化するためのファームウェア、および FPGA を FX2LP に対してマスターとして設定するための Verilog と HDL コードが含まれています。

著者について

氏名: Rama Sai Krishna V

役職: アプリケーション エンジニア担当

変更履歴

版署名: AN61345 - EZ-USB® FX2LP™ スレーブ FIFO インターフェースを使った設計

文書番号: 001-92749

版	ECN	変更者	提出日	変更内容
**	4396302	HZEN	07/17/2014	これは英語版 001-61345 Rev. *J を翻訳した日本語版 Rev. **です。
*A	5894527	HIKA	09/25/2017	これは英語版 001-61345 Rev. *L を翻訳した日本語版 Rev. *A です。

ワールドワイドな販売と設計サポート

サイプレスは、事業所、ソリューション センター、メーカー代理店、および販売代理店の世界的なネットワークを保持しています。お客様の最寄りのオフィスについては、サイプレスのロケーション ページをご覧ください。

製品

ARM® Cortex® Microcontrollers	cypress.com/arm
車載用	cypress.com/automotive
クロック&バッファ	cypress.com/clocks
インターフェース	cypress.com/interface
IoT (モノのインターネット)	cypress.com/iot
メモリ	cypress.com/memory
マイクロコントローラ	cypress.com/mcu
PSoC	cypress.com/psoc
電源用 IC	cypress.com/pmic
タッチ センシング	cypress.com/touch
USB コントローラー	cypress.com/usb
ワイヤレス／RF	cypress.com/wireless

PSoC® ソリューション

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

サイプレス開発者コミュニティ

フォーラム | WICED IOT Forums | Projects | ビデオ | ブログ | トレーニング | Components

テクニカルサポート

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2010-2017. 本書面は, Cypress Semiconductor Corporation 及び Spansion LLC を含むその子会社 (以下「Cypress」という。) に帰属する財産である。本書面 (本書面に含まれ又は言及されているあらゆるソフトウェア若しくはファームウェア (以下「本ソフトウェア」という。) を含む) は, アメリカ合衆国及び世界のその他の国における知的財産法令及び条約に基づき Cypress が所有する。Cypress はこれらの法令及び条約に基づく全ての権利を留保し, 本段落で特に記載されているものを除き, その特許権, 著作権, 商標権又はその他の知的財産権のライセンスを一切許諾しない。本ソフトウェアにライセンス契約書が伴っておらず, かつ Cypress との間で別途本ソフトウェアの使用方法を定める書面による合意がない場合, Cypress は, (1) 本ソフトウェアの著作権に基づき, (a) ソースコード形式で提供されている本ソフトウェアについて, Cypress ハードウェア製品と共に用いるためにのみ, かつ組織内部でのみ, 本ソフトウェアの修正及び複製を行うこと, 並びに (b) Cypress のハードウェア製品ユニットに用いるためにのみ, (直接又は再販売者及び販売代理店を介して間接のいずれかで) 本ソフトウェアをバイナリーコード形式で外部エンドユーザーに配布すること, 並びに (2) 本ソフトウェア (Cypress により提供され, 修正がなされていないもの) が抵触する Cypress の特許権のクレームに基づき, Cypress ハードウェア製品と共に用いるためにのみ, 本ソフトウェアの作成, 利用, 配布及び輸入を行うことについての非独占的で譲渡不能な一身専属的ライセンス (サブライセンスの権利を除く) を付与する。本ソフトウェアのその他の使用, 複製, 修正, 変換又はコンパイルを禁止する。

適用される法律により許される範囲内で, Cypress は, 本書面又はいかなる本ソフトウェア若しくはこれに伴うハードウェアに関しても, 明示又は黙示をとわず, いかなる保証 (商品性及び特定の目的への適合性の黙示の保証を含むがこれらに限られない) も行わない。 適用される法律により許される範囲内で, Cypress は, 別途通知することなく, 本書面を変更する権利を留保する。Cypress は, 本書面に記載のある, いかなる製品若しくは回路の適用又は使用から生じる一切の責任を負わない。本書面で提供されたあらゆる情報 (あらゆるサンプルデザイン情報又はプログラムコードを含む) は, 参照目的のためのみに提供されたものである。この情報で構成するあらゆるアプリケーション及びその結果としてのあらゆる製品の機能性及び安全性を適切に設計, プログラム, かつテストすることは, 本書面のユーザーの責任において行われるものとする。Cypress 製品は, 兵器, 兵器システム, 原子力施設, 生命維持装置若しくは生命維持システム, 蘇生用の設備及び外科的移植を含むその他の医療機器若しくは医療システム, 汚染管理若しくは有害物質管理の運用のために設計され若しくは意図されたシステムの重要な構成部分としての使用, 又は装置若しくはシステムの不具合が人身傷害, 死亡若しくは物的損害を生じさせるようなその他の使用 (以下「本目的外使用」という。) のためには設計, 意図又は承認されていない。重要な構成部分とは, それの不具合が装置若しくはシステムの不具合を生じさせるか又はその安全性若しくは実効性に影響すると合理的に予想できるような装置若しくはシステムのあらゆる構成部分をいう。Cypress 製品のあらゆる本目的外使用から生じ, 若しくは本目的外使用に関連するいかなる請求, 損害又はその他の責任についても, Cypress はその全部又は一部をとわず一切の責任を負わず, かつ Cypress はそれら一切から本書により免除される。Cypress は Cypress 製品の本目的外使用から生じ又は本目的外使用に関連するあらゆる請求, 費用, 損害及びその他の責任 (人身傷害又は死亡に基づく請求を含む) から免責補償される。

Cypress, Cypress のロゴ, Spansion, Spansion のロゴ及びこれらの組み合わせ, WICED, PSoC, CapsSense, EZ-USB, F-RAM, 及び Traveo は, 米国及びその他の国における Cypress の商標又は登録商標である。Cypress のより完全な商標のリストは, cypress.com を参照すること。その他の名称及びブランドは, それぞれの権利者の財産として権利主張がなされている可能性がある。