AN65974

EZ-USB FX3 スレーブ FIFO インターフェースを使った設計

著者: Rama Sai Krishna V

ソフトウェア バージョン: EZ-USB FX3 SDK1.3.4

関連アプリケーション ノート: AN75705, AN68829

さらにサンプルコードをお求めでしょうか？以下を参照してください。

豊富な USB Super Speed サンプルコードのリストについては、www.cypress.com/101781 をご覧ください。

本アプリケーション ノート (AN65974) は、EZ-USB® FX3™の同期スレーブ FIFO インターフェースにつて説明します。ハードウェア インターフェースと FLAG のコンフィギュレーション設定について詳細に説明し、例を示します。本資料は、スレーブ FIFO インターフェースを容易に設計できるように GPIF™ II Designer へのリファレンスを含んでいます。FPGA を FX3 と通信させるために同期スレーブ FIFO をどのように使用するかを示した 2 つの完成された設計例が用意されています。

目次

1. はじめに 2
2. 詳細情報 2
3. GPIF II 3
4. 同期スレーブ FIFO インターフェース 4
4.1. 2 本と 5 本のアドレス ラインを持つスレーブ FIFO の違い 5
4.2. スレーブ FIFO インターフェースのピン マップ 5
5. スレーブ FIFO のアクセス シーケンスとインターフェースのタイミング 6
5.1. 同期スレーブ FIFO インターフェースのタイミング .7
5.2. 同期スレーブ FIFO 読み出しシーケンス 7
5.3. 同期スレーブ FIFO 書き込みシーケンス 9
6. スレッドとソケット 10
7. DMA チャネルのコンフィギュレーション 11
8. フラグのコンフィギュレーション 12
8.1. 専用のスレッド フラグ 12
8.2. 現時点のスレッド フラグ 12
9. GPIF II Designer 15
9.1. 同期スレーブ FIFO インターフェースの実装 15
9.2. 部分的フラグの設定 15
9.3. 部分的フラグの一般式 18
9.4. CyU3PgpifSocketConfigure() API の使用例 18
9.5. 部分的フラグを使用する際の他の注意事項 21
9.6. フラグ違反によるエラー状態 21
10. SDK におけるスレーブ FIFO ファームウェアの例 22
11. 設計例 1: Xilinx FPGA を FX3 の同期スレーブ FIFO インターフェースと連結 23
11.1. ハードウェアのセットアップ 23
11.2. ファームウェアとソフトウェア コンポーネント 24
11.3. FX3 ファームウェアの詳細 25
11.4. FPGA 実装の詳細 28
11.5. プロジェクトの動作 36
12. 設計例 2: Altera FPGA を FX3 の同期スレーブ FIFO インターフェースと連結 43
12.1. ハードウェアのセットアップ 43
12.2. ファームウェアとソフトウェア コンポーネント 44
12.3. FX3 ファームウェアの詳細 45
12.4. FPGA 実装の詳細 48
12.5. プロジェクトの動作 55
13. 関連プロジェクト ファイル 61
14. まとめ 62
付録 A: トラブルシューティング 63
付録 B: FX3 DVK (CYUSB3KIT-001) を用いたハードウェア セットアップ 65
B.1 ジャンパおよびスイッチの設定 66
付録 C 67
C.1 ショートパケット例 67
C.2 長さゼロのパケット (ZLP) 例 68
改訂履歴 71
ワールドワイドな販売と設計サポート 72

1. はじめに

サイプレス EZ-USB FX3 は次世代 USB 3.0 ペリフェラル コントローラーであり、開発者はどんなシステムにも USB 3.0 機能を追加できます。このコントローラーは、画像装置やビデオ装置、プリンター、スキャナーなどのアプリケーションに適用できます。

EZ-USB FX3 は、完全に設定可能なパラレルかつ汎用プログラマブルなインターフェースを備えています。この GPIF II と呼ばれるインターフェースは外部プロセッサ、ASIC、FPGA などに接続できます。GPIF II は、サイプレスの主力 USB 2.0 製品である FX2LP の GPIF 拡張バージョンです。GPIF II は同期アドレス データの多重化インターフェースのようなインターフェースを介して、FPGA や画像センサー、プロセッサなど一般に使用されているデバイスへの密着した接続を提供します。

GPIF II がよく使用される実装例には、同期スレーブ FIFO インターフェースがあります。このインターフェースは、EZ-USB FX3 に接続した外部デバイスが FX3 FIFO にアクセスしてデータを読み書きするアプリケーションに使用されます。スレーブ FIFO インターフェースを介してレジスタへは、直接アクセスできません。

本アプリケーション ノートでは、まず GPIF II を紹介してから、同期スレーブ FIFO インターフェースについて詳しく説明します。本書には、FPGA 上で同期スレーブ FIFO と互換性があるマスター インターフェースを実装する方法についての 2 つの完全な設計例も示します。Xilinx Spartan 6 FPGA と Altera Cyclone III FPGA 用の Verilog と VHDL ファイルを提供しています。同期スレーブ FIFO に対応する FX3 ファームウェア プロジェクトが設計例に含まれています。これらの例では、Spartan 6 FPGA 用の Xilinx SP601 評価キットまたは Cyclone III FPGA 用の Altera Cyclone III スターター ボード, FX3 開発キット (DVK), および FX3 ソフトウェア開発キット (SDK) を使用して開発します。

2. 詳細情報

サイプレスは www.cypress.com で豊富なデータを提供しており、お客様の設計に適したデバイスを選択したり、そのデバイスを迅速かつ効果的にデザインに統合できます。

- 概要: USB Portfolio, USB Roadmap
- USB 3.0 製品選択: FX3, FX3S, CX3, HX3
- アプリケーションノート: サイプレスは基本から上級レベルまで幅広いトピックを網羅する多数のアプリケーションノートを提供しています。FX3 を使い始めるための推奨アプリケーションノートは次のとおりです。
  - AN75705 – EZ-USB FX3 入門
  - AN70707 – EZ-USB FX3/FX3S ハードウェア設計ガイドラインおよび回路図チェックリスト
  - AN65974 – EZ-USB FX3 スレーブ FIFO インターフェースを使った設計
  - AN75779 – USB ビデオクラス(UVC)フレームワーク内で EZ-USB FX3 を使用してイメージセンサーインターフェースを実装する方法
  - AN86947 –EZ-USB FX3 による USB 3.0 スループットの最適化
  - AN84868 –サイプレスの EZ-USB FX3 を使用した USB 経由での Xilinx FPGA コンフィギュレーション
  - AN68829 – Slave FIFO Interface for EZ-USB FX3: 5-Bit Address Mode
  - AN76348 – Differences in Implementation of EZ-USB FX2LP and EZ-USB FX3 Applications
  - AN89661 – USB RAID 1 Disk Design Using EZ-USB FX3S
- Code Examples:
  - USB Hi-Speed
  - USB Full-Speed
  - USB SuperSpeed
- Technical Reference Manual (TRM):
  - EZ-USB FX3 Technical Reference Manual
- 開発キット:
  - CYUSB3KIT-003, EZ-USB FX3 SuperSpeed Explorer Kit
  - CYUSB3KIT-001, EZ-USB FX3 Development Kit
- モデル: IBIS

2.1. EZ-USB FX3 ソフトウェア開発キット

サイプレスは、SuperSpeed USB をあらゆる組込みアプリケーションに簡単に統合するための FX3 用の完全なソフトウェアおよびファームウェアスタックを提供します。Software Development Kit (SDK) には、アプリケーション開発を加速するのに役立つツール, ドライバ, およびアプリケーションの例が付属しています。

2.2. GPIF™ II Designer

GPIF II Designer は、設計者が EZ-USB FX3 USB 3.0 デバイスコントローラの GPIF II インターフェースを構成できるグラフィカルソフトウェアです。

このツールを使用すると、サイプレスが提供する 5 つのインターフェースから 1 つを選択することも、独自の GPIF II インターフェースを最初から作成することもできます。サイプレスは、非同期および同期スレーブ FIFO、非同期および同期 SRAM などの業界標準インターフェースを提供しています。システムにこれらの定義済みインターフェースのいずれかをすでに用意されている場合は、バス幅 (x8, 16, x32) エンディアン、クロック設定などの標準パラメーターのセットから選択して、インターフェースをコンパイルできます。このツールには、カスタマイズされたインターフェースを必要とするユーザーのために合理化した 3 段階の GPIF インターフェース開発プロセスがあります。ユーザーはまず自分のピン構成と標準パラメーターを選択できます。次に、構成可能なアクションを使用して仮想ステートマシンを設計できます。最後に、ユーザーは出力タイミングを確認することで、予想されているタイミングと一致していることを確認できます。この 3 ステップのプロセスが完了した後、インターフェースをコンパイルして FX3 と統合できます。

3. GPIF II

GPIF II は、業界標準または独自のインターフェースのフレキシブルな実装を可能にする、プログラム可能なステートマシンです。これはマスターまたはスレーブのいずれとしても機能できます。

GPIF II の特長は以下のとおりです。

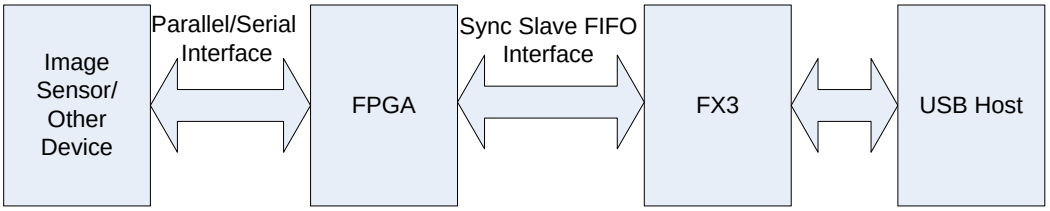
- マスターまたはスレーブとして動作
- ファームウェアでプログラム可能な 256 の状態を提供
- 8 ビット、16 ビット、32 ビットのパラレル データ バスをサポート
- 最大 100MHz までのインターフェース周波数に対応
- 32 ビット データ バスを使用する場合は、14 本の設定可能な制御ピンをサポート。すべての制御ピンは入力／出力または双方向ピンのどちらとしても利用可能
- 16／8 ビット データ バスを使用する場合は、16 本の設定可能な制御ピンをサポート。すべての制御ピンは入力／出力または双方向ピンのどちらとしても利用可能

GPIF II の状態遷移は制御入力信号に基づいて発生します。制御出力信号は GPIF II の状態遷移により駆動されます。ステートマシンの動作は、必要なインターフェース仕様を満たすように設計されたディスクリプタで定義されます。GPIF II ディスクリプタは、基本的にプログラム可能なレジスタ コンフィギュレーションの一式です。EZ-USB FX3 レジスタ空間の 8KB メモリは、GPIF II 波形メモリ専用であり、ここに GPIF II ディスクリプタが格納されます。

GPIF II がよく使用される実装例は同期スレーブ FIFO インターフェースであり、以下の節で詳細に説明します。

図 1 に、同期スレーブ FIFO インターフェースを使用したアプリケーション例を示します。

図 1. アプリケーション例の図



4. 同期スレーブ FIFO インターフェース

同期スレーブ FIFO インターフェースは、外部プロセッサやデバイスが EZ-USB FX3 の内蔵 FIFO バッファに対してデータ読み書きアクセスを実行する必要があるアプリケーションに最適です。レジスタへのアクセスは、スレーブ FIFO インターフェースを介しては実行されません。普通は、高い処理能力の要件に対応できるために同期スレーブ FIFO インターフェースは USB アプリケーションのインターフェースとして選択されます。

図 2 に、同期スレーブ FIFO インターフェースの図を示します。表 1 は、図 2 に示した信号を説明します。

図 2. 同期スレーブ FIFO インターフェースの図

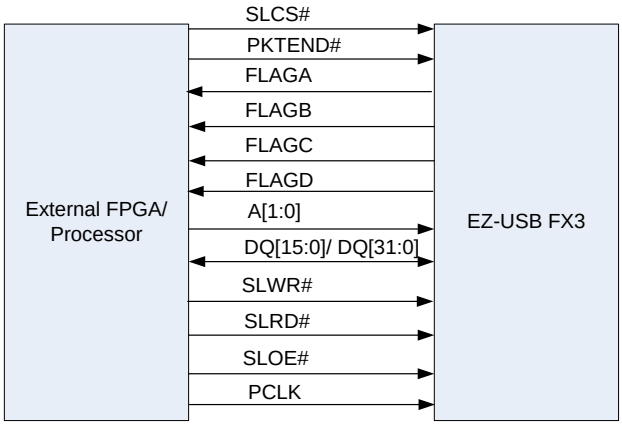


表 1. 同期スレーブ FIFO インターフェースの信号

信号名	説明
SLCS#	スレーブ FIFO インターフェースのチップ選択信号。スレーブ FIFO にアクセスするためにアサート
SLWR#	スレーブ FIFO インターフェースの書き込みストローブ信号。スレーブ FIFO への書き込み転送を実行するためにアサート
SLRD#	スレーブ FIFO インターフェースの読み出しストローブ信号。スレーブ FIFO からの読み出し転送を実行するためにアサート
SLOE#	出力イネーブル信号。FX3 にスレーブ FIFO インターフェースのデータ バスを駆動させる。スレーブ FIFO からの読み出し転送を実行するためにアサート
FLAGA/ FLAGB/ FLAGC/ FLAGD	FX3 からのフラグ出力。FLAG は FX3 ソケットの状態を示す。[1] 本資料に含まれるサンプル プロジェクトでは、FLAGA と FLAGB をスレーブ FIFO の書き込み動作に、FLAGC と FLAGD をスレーブ FIFO の読み出し動作に使用
A[1:0]	スレーブ FIFO の 2 ビット アドレス バス
DQ[15:0]/ DQ[31:0]	スレーブ FIFO の 16 ビット／32 ビット データ バス
PKTEND#	ショート パケットまたは長さゼロのパケットをスレーブ FIFO に書き込むためにアサート
PCLK	スレーブ FIFO インターフェース クロック

[1]「スレッドとソケット」節では、データ転送用のソケットの概念について説明します。FLAG については、「フラグのコンフィギュレーション」節で詳細に説明します。

4.1. 2 本と 5 本のアドレス ラインを持つスレーブ FIFO の違い

アドレス ラインを 2 本持つ同期スレーブ FIFO インターフェースは最大 4 個のソケットに対応しています。4 個以上のソケットにアクセスするには、アドレス ラインを 5 本持つ同期スレーブ FIFO インターフェースをご使用ください。追加のアドレス ラインに加えて、このインターフェースは EPSWITCH#信号も持っています。それらの増加されたピンのため、FLAG として使用できるピン数が少なくなるので、フラグを current_thread FLAG として設定する必要があります。

アドレス ラインを 5 本持つ同期スレーブ FIFO インターフェースを使用する場合、追加の遅延が生じます。

- アドレスからフラグが有効になるまでの 2 サイクルの遅延が各転送の開始時に生じます。
- ソケット アドレスが切り替わるたびに、ソケットの切り替えを完成するために数サイクルの遅延が生じます。

増加した遅延と追加のインターフェース プロトコルの要件のため、アプリケーションが 4 個以上の GPIF II ソケットへのアクセスを必要とする場合にのみ、アドレス ラインを 5 本持つ同期スレーブ FIFO インターフェースを使用することをお勧めします。このインターフェースの詳細については、アプリケーション ノート AN68829 – Slave FIFO Interface for EZ-USB FX3: 5-Bit Address Mode を参照してください。

本アプリケーション ノートの以降の節では、アドレス ラインを 2 本持つ同期スレーブ FIFO インターフェースについて説明します。

4.2. スレーブ FIFO インターフェースのピン マップ

表 2 に、スレーブ FIFO インターフェースの初期設定のピン マップを示します。この表に、GPIF II をスレーブ FIFO インターフェースに設定した場合に使用可能な GPIO ピンおよびその他のシリアル インターフェース (UART/SPI/I²S) も示します。

ピン マップは必要に応じて変更でき、FLAG は GPIF II Designer ツールを使用して追加するか再設定もできます。詳細については、フラグのコンフィギュレーションを参照してください。

表 2. スレーブ FIFO インターフェースのピン マップ

EZ-USB FX3 のピン	SuperSpeed explorer kit/ Interconnect ボードでのピン名	8 ビット データ バスを使った同期スレーブ FIFO インターフェース	16 ビット データ バスを使った同期スレーブ FIFO インターフェース	24 ビット データ バスを使った同期スレーブ FIFO インターフェース	32 ビット データ バスを使った同期スレーブ FIFO インターフェース
GPIO[17]	CTL[0]	SLCS#	SLCS#	SLCS#	SLCS#
GPIO[18]	CTL[1]	SLWR#	SLWR#	SLWR#	SLWR#
GPIO[19]	CTL[2]	SLOE#	SLOE#	SLOE#	SLOE#
GPIO[20]	CTL[3]	SLRD#	SLRD#	SLRD#	SLRD#
GPIO[21]	CTL[4]	FLAGA	FLAGA	FLAGA	FLAGA
GPIO[22]	CTL[5]	FLAGB	FLAGB	FLAGB	FLAGB
GPIO[23]	CTL[6]	FLAGC	FLAGC	FLAGC	FLAGC
GPIO[24]	CTL[7]	PKTEND#	PKTEND#	PKTEND#	PKTEND#
GPIO[25]	CTL[8]	FLAGD	FLAGD	FLAGD	FLAGD
GPIO[28]	CTL[11]	GPIO	A1	A1	A1
GPIO[29]	CTL[12]	GPIO	A0	A0	A0
GPIO[0:7]	DQ[0:7]	DQ[0:7]	DQ[0:7]	DQ[0:7]	DQ[0:7]
GPIO[8]	DQ[8]	A0	DQ[8]	DQ[8]	DQ[8]
GPIO[9]	DQ[9]	A1	DQ[9]	DQ[9]	DQ[9]
GPIO[10:15]	DQ[10:15]	GPIO として使用可能	DQ[10:15]	DQ[10:15]	DQ[10:15]

EZ-USB FX3 のピン	SuperSpeed explorer kit/ Interconnect ボードでのピン名	8 ビット データ バスを使った同期スレーブ FIFO インターフェース	16 ビット データ バスを使った同期スレーブ FIFO インターフェース	24 ビット データ バスを使った同期スレーブ FIFO インターフェース	32 ビット データ バスを使った同期スレーブ FIFO インターフェース
GPIO[16]	PCLK	PCLK	PCLK	PCLK	PCLK
GPIO[33:41]	DQ[16:24]	GPIO として使用可能	GPIO として使用可能	DQ[16:24]	DQ[16:24]
GPIO[42:44]	DQ[25:27]	GPIO として使用可能	GPIO として使用可能	GPIO として使用可能	DQ[25:27]
GPIO[45]	IO45	GPIO	GPIO	GPIO	GPIO
GPIO[46]	DQ[28]	GPIO/UART_RTS	GPIO/UART_RTS	GPIO/UART_RTS	DQ[28]
GPIO[47]	DQ[29]	GPIO/UART_CTS	GPIO/UART_CTS	GPIO/UART_CTS	DQ[29]
GPIO[48]	DQ[30]	GPIO/UART_TX	GPIO/UART_TX	GPIO/UART_TX	DQ[30]
GPIO[49]	DQ[31]	GPIO/UART_RX	GPIO/UART_RX	GPIO/UART_RX	DQ[31]
GPIO[50]	I2S_CLK	GPIO/I2S_CLK	GPIO/I2S_CLK	GPIO/I2S_CLK	GPIO/I2S_CLK
GPIO[51]	I2S_SD	GPIO/I2S_SD	GPIO/I2S_SD	GPIO/I2S_SD	GPIO/I2S_SD
GPIO[52]	I2S_WS	GPIO/I2S_WS	GPIO/I2S_WS	GPIO/I2S_WS	GPIO/I2S_WS
GPIO[53]	RTS/SCK	GPIO/SPI_SCK /UART_RTS	GPIO/SPI_SCK /UART_RTS	GPIO/SPI_SCK /UART_RTS	GPIO/UART_RTS
GPIO[54]	CTS/SSN	GPIO/SPI_SSN /UART_CTS	GPIO/SPI_SSN /UART_CTS	GPIO/SPI_SSN /UART_CTS	GPIO/UART_CTS
GPIO[55]	TX/MOSI	GPIO/SPI_MISO /UART_TX	GPIO/SPI_MISO /UART_TX	GPIO/SPI_MISO /UART_TX	GPIO/UART_TX
GPIO[56]	RX/MISO	GPIO/SPI_MOSI /UART_RX	GPIO/SPI_MOSI /UART_RX	GPIO/SPI_MOSI /UART_RX	GPIO/UART_RX
GPIO[57]	I2S_MCLK	GPIO/I2S_MCLK	GPIO/I2S_MCLK	GPIO/I2S_MCLK	GPIO/I2S_MCLK

注: EZ-USB FX3 の完全なピン マップは、EZ-USB FX3 スーパースピード USB コントローラーのデータシートを参照してください。

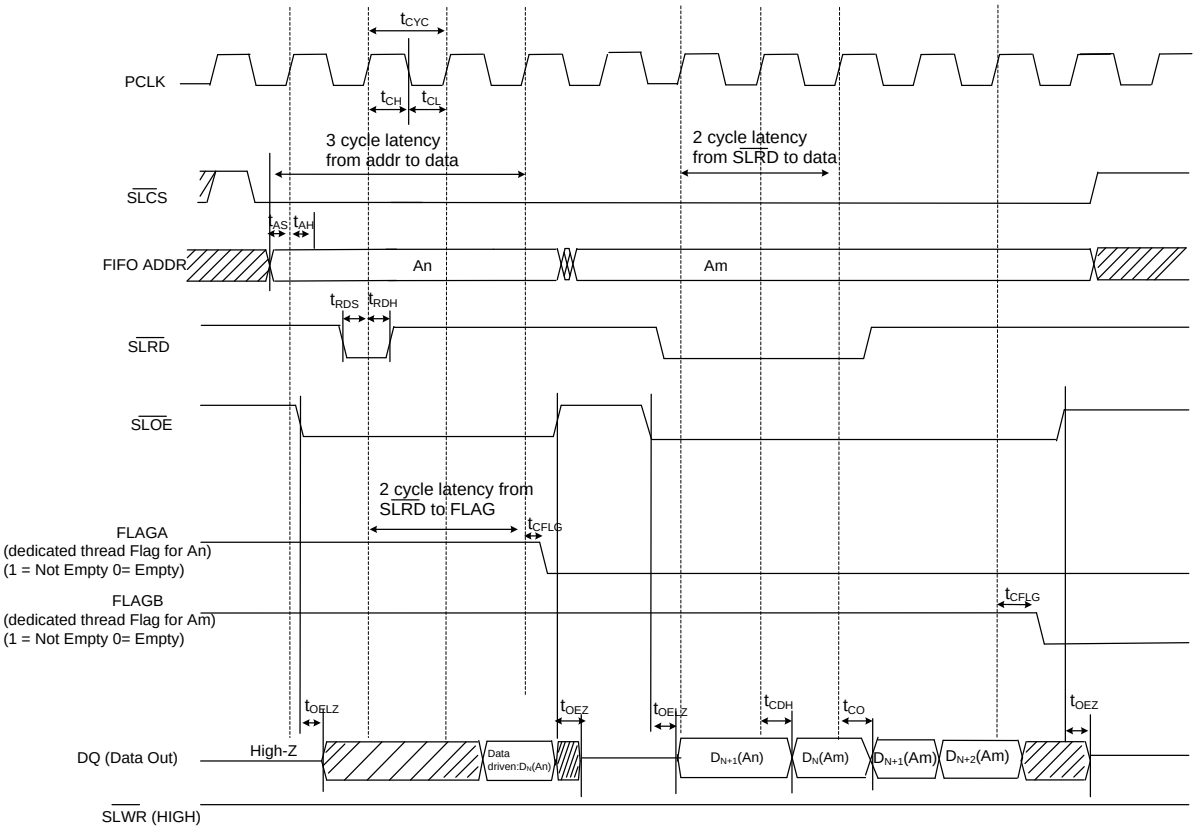
5. スレーブ FIFO のアクセス シーケンスとインターフェースのタイミング

ここでは、同期スレーブ FIFO インターフェースのアクセス シーケンスとタイミングについて説明します。

外部プロセッサまたはデバイス (インターフェースのマスターとして機能する) は、EZ-USB FX3 の内部 FIFO バッファに対してシングル サイクルまたはバースト データ アクセスを実行できます。外部マスターは 2 ビット アドレスを ADDR ライン上で駆動し、読み出しか書き込みストローブをアサートします。EZ-USB FX3 はフラグ信号をアサートしてバッファの空または一杯の状態を示します。

5.1. 同期スレーブ FIFO インターフェースのタイミング

図 3. 同期スレーブ FIFO 読み出しシーケンス



5.2. 同期スレーブ FIFO 読み出しシーケンス

同期スレーブ FIFO インターフェースからの読み出しを実行するシーケンスは以下の通りです。

1. FIFO アドレスが安定し、SLCS#がアサートされます。
2. SLOE#がアサートされます。SLOE#は出力イネーブル専用で、その唯一の機能はデータ バスを駆動することです。
3. SLRD#がアサートされます。

SLRD#のアサート中、FIFO ポインタは PCLK の立ち上がりエッジで更新されます。この動作は、データ バスへの新たにアドレス指定された FIFO からのデータ伝播を開始します。t_{CO} の伝播遅延 (PCLK の立ち上がりエッジから測定される) の後、新しいデータ値を得ます。N は、FIFO から読み出される最初のデータ値です。データ バスを駆動するために、SLOE#をアサートする必要もあります。

バースト読み出しの場合でも同じ一連のイベントが示されます。

注: バースト モードでは、読み出し期間中、SLRD#と SLOE#がアサートされたままです。SLOE#がアサートされると、以前にアドレス指定された FIFO からのデータを持つデータ バスが駆動されます。SLRD#がアサートされている間、PCLK の後続の各立ち上がりエッジで、FIFO ポインタがインクリメントされ、次のデータ値がデータ バスに置かれます。

フラグの使用: フロー制御用の外部プロセッサはフラグ信号を監視します。フラグ信号は EZ-USB FX3 からの出力であり、特定のスレッドまたは現時点でアドレス指定されているスレッドの空／満杯／部分的状態を示すように設定されます。

図 4. 同期スレーブ FIFO の書き込みシーケンス

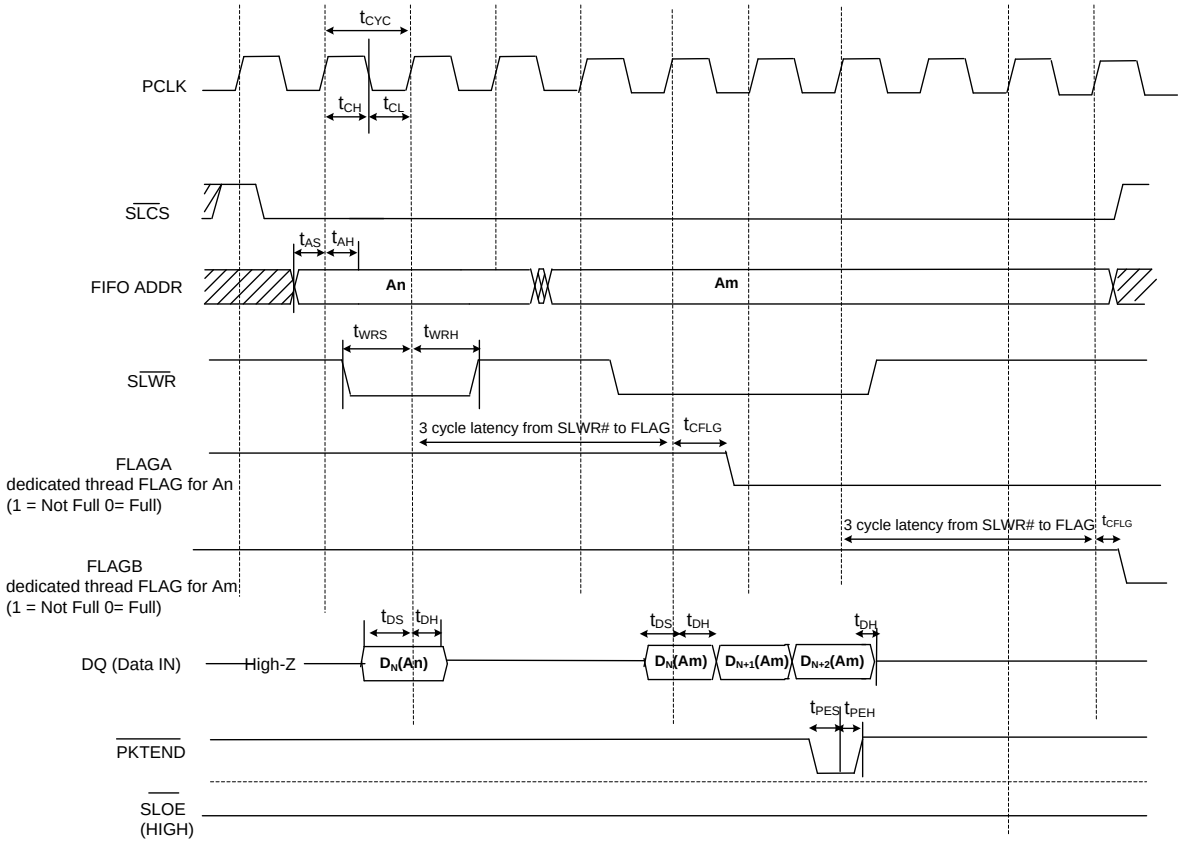
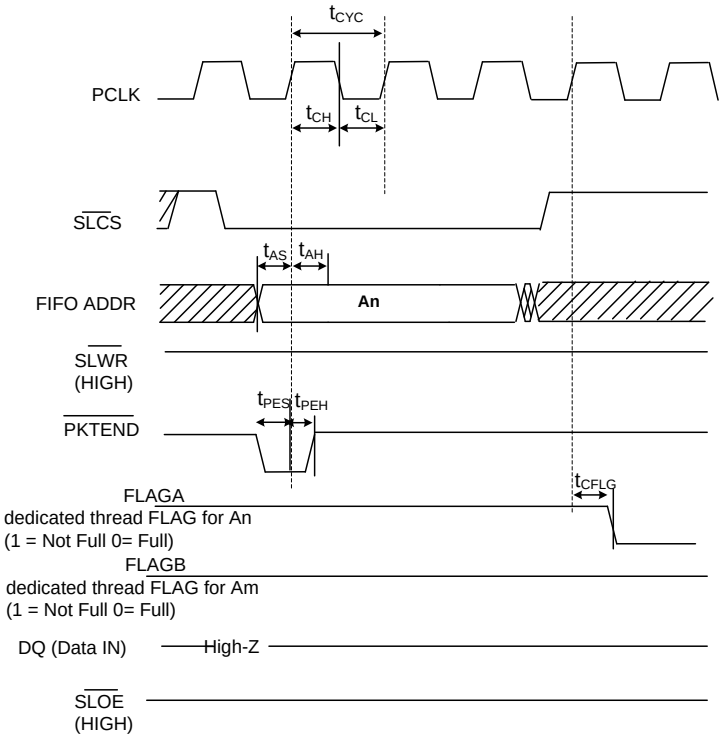


図 5. 同期 ZLP 書き込みサイクルのタイミング



5.3. 同期スレーブ FIFO 書き込みシーケンス

同期スレーブ FIFO インターフェースへの書き込みを実行するシーケンスは以下のとおりです。

1. FIFO アドレスが安定し、SLCS#がアサートされます。
2. 外部マスター／ペリフェラルがデータ バス上にデータを出力します。
3. SLWR#がアサートされます。
4. SLWR#がアサートされている間、データが FIFO に書き込まれ、PCLK の立ち上がりエッジで FIFO ポインタがインクリメントされます。
5. FIFO フラグは、クロックの立ち上がりエッジから遅延時間 t_{CFLG} 後に更新されます。

バースト読み出しの場合でも同じ一連のイベントが示されます。

注: バースト モードでは、バースト書き込み期間中、SLWR#と SLCS#がアサートされたままです。バースト書き込みモードでは、SLWR#がアサートされた後、データ バス上の値が PCLK の各立ち上がりエッジで FIFO に書き込まれます。FIFO ポインタは PCLK の各立ち上がりエッジで更新されます。

ショート パケット: ショート パケットは、PKTEND#信号を使用して USB ホストに転送します。外部デバイス／プロセッサは、最後のデータ ワードとそれに対応する SLWR#パルスと共に PKTEND#をアサートするよう設計する必要があります。FIFOADDR ラインは、PKTEND#のアサート中に一定に保持しなければなりません。SLWR#と PKTEND#のアサート時に、GPIF II ステートマシンはパケットをショート パケットとして解釈し、USB インターフェースに転送します。プロトコルがショート パケットを転送する必要がない場合、PKTEND#信号が HIGH にプルされることがあります。

読み出し方向では、ショート パケットの送信元が USB であることを示す特定の信号がないため、ご注意ください。外部マスターは、すべてのデータが読み出された時点を判定するために空のフラグを監視する必要があります。

長さゼロのパケット: 外部デバイス／プロセッサは、SLWR#をアサートせずに PKTEND#をアサートすることで長さゼロのパケット (ZLP) を転送することができます。SLCS#とアドレスは、図 5 に示すように駆動する必要があります。

フラグの使用: 外部プロセッサはフロー制御のためにフラグ信号を監視します。フラグ信号は、EZ-USB FX3 デバイスからの出力であり、特定のスレッドまたは現時点でアドレス指定されているスレッドの空／一杯／部分的状態を示すように設定されます。

表 3. 同期スレーブ FIFO のタイミング パラメーター

パラメーター	説明	Min	Max	単位
FREQ	インターフェース クロック周波数	–	100	MHz
t_{CYC}	クロック周期	10	–	ns
t_{CH}	クロック HIGH 時間	4	–	ns
t_{CL}	クロック LOW 時間	4	–	ns
t_{RDS}	SLRD#から CLK までのセットアップ時間	2	–	ns
t_{RDH}	SLRD#から CLK までのホールド時間	0.5	–	ns
t_{WRS}	SLWR#から CLK までのセットアップ時間	2	–	ns
t_{WRH}	SLWR#から CLK までのホールド時間	0.5	–	ns
t_{CO}	クロックからデータ有効までの時間	–	7	ns
t_{DS}	データ入力セットアップ時間	2	–	ns
t_{DH}	CLK からデータ入力までのホールド時間	0	–	ns
t_{AS}	アドレスから CLK までのセットアップ時間	2	–	ns
t_{AH}	CLK からアドレスまでのホールド時間	0.5	–	ns
t_{OELZ}	SLOE#からデータ LOW-Z までの時間	0	–	ns
t_{CFLG}	CLK からフラグ出力までの伝播遅延	–	8	ns
t_{OEZ}	SLOE#アサート停止からデータ HI-Z までの時間	–	8	ns
t_{PES}	PKTEND#から CLK までのセットアップ時間	2	–	ns
t_{PEH}	CLK から PKTEND#までのホールド時間	0.5	–	ns
t_{CDH}	CLK からデータ出力までのホールド時間	2	–	ns
注: ADDR から DATA までの 3 サイクル遅延				

以下の節で、GPIF II Designer と EZ-USB FX3 SDK を使用したフラグ信号のコンフィギュレーションについて説明します。様々なフラグ コンフィギュレーションを説明する前に、スレッド, ソケット, および DMA チャネルの概念を紹介するのが重要です。

6. スレッドとソケット

ここでは、FX3 の入出力データ転送に必要な概念を簡単に説明します。

- ソケット
- DMA ディスクリプタ
- DMA バッファ
- GPIF スレッド

ソケットとは、ペリフェラルハードウェアブロックと FX3 RAM の接続点です。USB、GPIF、UART、SPI など FX3 上のペリフェラルハードウェアブロックには、それぞれに対応する一定数のソケットがあります。各ペリフェラルを介した独立のデータフロー

数は、それに対応するソケット数に等しいです。ソケットの実装には、アクティブな DMA ディスクリプタを指し、ソケットに対応する割り込みを有効にする、または通知するレジスター式が含まれます。

DMA ディスクリプタとは、FX3 RAM に割り付けられたレジスター式です。DMA バッファのアドレスとサイズの情報および次の DMA ディスクリプタへのポインタを格納しています。これらのポインタは DMA ディスクリプタ チェーンを作ります。

DMA バッファとは、RAM の一部であり、FX3 デバイスを介して転送されるデータの中間記憶領域として使用されます。DMA バッファは、FX3 ファームウェアによって RAM から割り付けられ、それらのアドレスが DMA ディスクリプタに格納されています。

GPIF スレッドとは、GPIF II ブロック内の専用データ経路であり、外部データ ピンをソケットに接続します。ソケットは、イベントにより互いに直接通知するか、または割り込みにより FX3 CPU に通知します。ファームウェアは信号方式を設定します。例えば、GPIF II ブロックから USB ブロックまでのデータ ストリームを例にとります。GPIF ソケットはデータで DMA バッファが一杯になったことを USB ソケットに通知でき、USB ソケットは DMA バッファが空であることを GPIF ソケットに通知できます。この実装は、自動 DMA チャネルと呼ばれています。自動 DMA チャネルの実装は、FX3 CPU がデータ ストリームでデータを修正する必要ない場合に使用されています。

あるいは、GPIF ソケットは FX3 CPU に割り込みを送信することで、GPIF ソケットが DMA バッファを一杯にしたことの通知もできます。FX3 CPU はこの情報を USB ソケットに伝達します。USB ソケットは FX3 CPU に割り込みを送信することで、USB ソケットが DMA バッファを空にしたことを通知できます。その後、FX3 CPU はこの情報を GPIF ソケットに伝達します。これは、手動 DMA チャネルの実装と呼ばれています。この実装は、FX3 CPU がデータ ストリームでデータを追加／除去／修正する必要がある場合に使用されています。

DMA バッファにデータを書き込むソケットは、プロデューサ ソケット (producer socket) と呼ばれています。DMA バッファからデータを読み出すソケットは、コンシューマ ソケット (consumer socket) と呼ばれています。ソケットはデータ管理のために、DMA ディスクリプタに格納された DMA バッファ アドレス、DMA バッファ サイズおよび DMA ディスクリプタ チェーンを使用します。ソケットは DMA バッファを一杯にしたか空にした後、ある DMA ディスクリプタから他の DMA ディスクリプタに切り替えるのに限られた時間 (数マイクロ秒まで) かかります。切り替え中、ソケットはデータを転送できません。

EZ-USB FX3 は、GPIF II を介したデータ転送用に 4 つの物理的なハードウェア スレッドを提供します。1 度に 1 つのソケットが物理的なスレッドにマッピングされます。初期設定では、PIB ソケット 0 はスレッド 0 に、PIB ソケット 1 はスレッド 1 に、PIB ソケット 2 はスレッド 2 に、PIB ソケット 3 はスレッド 3 にマッピングされます。

インターフェースのアドレス信号 A1:A0 は、アクセスされるスレッドを示していることにご注意ください。次に FX3 の DMA ファブリックは、そのスレッドにマッピングされたソケットにデータを送信します。したがって、A1:A0 = 0 の時、スレッド 0 がアクセスされ、スレッド 0 を介して転送されたすべてのデータはソケット 0 に送信されます。同様に、A1:A0 = 1 の時、データはソケット 1 に入力および出力転送されます。

注: スレーブ FIFO インターフェースはアドレス ラインを 2 本のみ持っているため、最大 4 個のソケットにアクセスできます。4 個以上のソケットにアクセスするには、アドレス ラインを 5 本持つスレーブ FIFO インターフェースをご使用ください。アプリケーション ノート AN68829 – Slave FIFO Interface for EZ-USB FX3: 5-Bit Address Mode を参照してください。

アクセスされるソケットは DMA チャネルを設定することで指定する必要があります。

注: FX3 のスレッドとソケットの詳細については、EZ-USB FX3 Technical Reference Manual のセクション 7.4.6 を参照してください。

7. DMA チャネルのコンフィギュレーション

ファームウェアは、要求されたプロデューサとコンシューマ ソケットで DMA チャネルを設定する必要があります。

データをスレーブ FIFO インターフェースから USB インターフェースへ転送する場合、P ポートはプロデューサとなり、USB はコンシューマとなります。その逆の場合も同様です。 したがって、データをスレーブ FIFO インターフェースで双方向に転送する場合は、2 つの DMA チャネルを設定する必要があります。1 つはプロデューサとして P ポートを使用し、もう 1 つはコンシューマとして P ポートを使用します。

P ポート プロデューサ ソケットは外部デバイスがスレーブ FIFO インターフェースを介して書き込むソケットであり、P ポート コンシューマ ソケットは外部デバイスがスレーブ FIFO インターフェースを介して読み出すソケットです。

DMA チャネル内の P ポート ソケット番号は、A1:A0 でアドレス指定されるソケット番号です。

特定の DMA チャネルの設定中に、複数のバッファをそのチャネルに割り当てられます。FLAG がバッファごとに一杯／空の状態を示すため、ご注意ください。(すべてのバッファは、最大サイズは 64KB -16 です。)

例えば、1024 バイトの 2 つのバッファが DMA チャネルに割り当てられた場合、一杯フラグは、1024 バイトが最初のバッファに書き込まれた時に一杯の状態を示します。それは、DMA チャネルが 2 番目のバッファに切り替えるまで一杯の状態を示したままです。DMA チャネルが次のバッファに切り替えるのに要する時間は、普通は数マイクロ秒ですが、予測できません。外部マスターは、この切り替えが完了して次のバッファがデータ アクセス用に使用可能になる時点を判定するために、フラグを監視する必要があります。

次の節で、異なるスレッドの状態を示すように FLAG をどのように設定するかについて説明します。

8. フラグのコンフィギュレーション

フラグは、空, 一杯, 部分的に空, または部分的に一杯の信号として設定されます。これらは GPIF II ステートマシンによって制御されず、EZ-USB FX3 の内蔵 DMA ハードウェア エンジンによって制御されます。フラグは特定のスレッドまたは現時点でアドレス指定されているスレッドに対応し、そのスレッドにマッピングされたソケットの状態を示します。

フラグはソケットの方向 (ソケットの初期化時に設定された) に基づいて、空または一杯の状態を示します。そのためフラグは、データがソケットから読み出されている時に空／空ではない状態を示し、データがソケットに書き込まれている時に一杯／一杯ではない状態を示します。

使用可能な各種フラグは以下のとおりです。

- 専用のスレッド フラグ (空／一杯、または部分的に空／一杯)
- 現時点のスレッド フラグ (空／一杯、または部分的に空／一杯)

各種フラグは以降の節で説明します。各種フラグ コンフィギュレーションは結果的に異なる遅延となり、表 4 でまとめられています。

8.1. 専用のスレッド フラグ

フラグは、特定のスレッドの状態を示すように設定できます。この場合、そのフラグはそのスレッド専用となり、どのスレッドがアドレス バス上でアドレス指定されているかに関わらず、その特定のスレッドにマッピングされたソケットの状態をのみ示します。

ここで、外部プロセッサ／デバイスは、どのフラグがどのスレッド専用であるかを追跡し、他のスレッドがアドレス指定されるたびに正しいフラグを監視する必要があります。

例えば、FLAGA がスレッド 0 専用で、FLAGB がスレッド 1 専用である場合、外部プロセッサはスレッド 0 にアクセスする際、FLAGA を監視する必要があります。外部プロセッサがスレッド 1 にアクセスする際、FLAGB を監視する必要があります。

フラグは、アクセスされるすべてのスレッド専用にできます。アプリケーションが 4 つのスレッドにアクセスする必要がある場合は、4 つの対応するフラグがあります。

書き込み転送を実行する時、常にフラグ用の 3 サイクル遅延は転送の終了時に生じるため、ご注意ください。3 サイクルの遅延は、バッファが一杯になる書き込みサイクルからフラグが LOW にアサートされる時点までの時間です。4 番目のクロック エッジで、外部マスターは LOW になっているフラグをサンプリングできます。これを図 4 に示します。

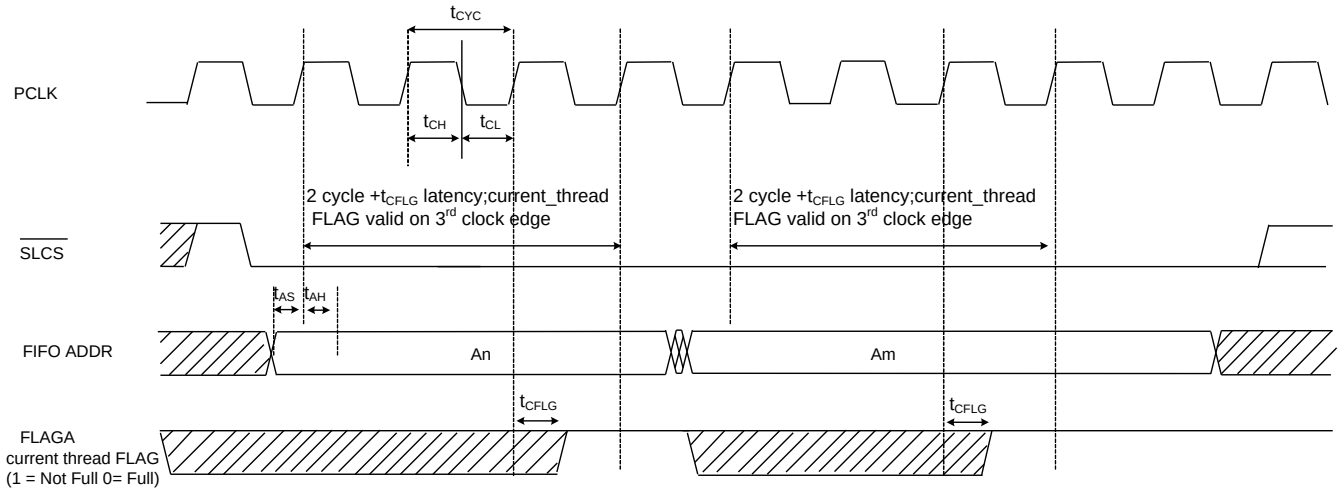
読み出し転送を実行する時、常にフラグ用の 2 サイクル遅延は転送の終了時に生じます。2 サイクルの遅延は、バッファが空になる読み出し (最後の SLRD#アサート) サイクルからフラグが LOW にアサートされる時点までの時間です。3 番目のクロック エッジで、外部マスターは LOW になっているフラグをサンプリングできます。これを図 3 に示します。

8.2. 現時点のスレッド フラグ

フラグは、現時点でアドレス指定されているスレッドの状態を示すように設定できます。この場合、GPIF II ステートマシンは、アドレス バス上のアドレスをサンプリングしてから、そのスレッドの状態を示すようにフラグを更新します。1 つの「current_thread」フラグはすべての 4 つのスレッドの状態を示すために使用できるので、このコンフィギュレーションに必要なピンが少なくなります。ただし、current_thread フラグが同期スレーブ FIFO インターフェース用に使用される場合、GPIF II がまずアドレスをサンプリングしてからフラグを更新する必要があるので、2 サイクルの遅延が生じます。有効なアドレスがインターフェースでサンプリングされると、2 サイクルの遅延が始まります。この後の 3 番目のクロック エッジで、新たにアドレス指定されたスレッドのフラ

グの有効な状態をサンプリングできます。(SDK に同梱されているスレーブ FIFO ディスクリプタは「current_thread」フラグのコンフィギュレーションを使用します。)

図 6. 現時点のスレッド フラグを使う場合の転送開始時に生じる追加の遅延



注: 書き込み転送を実行する時、常に 3 サイクル遅延は転送の終了時に生じます。3 サイクルの遅延は、バッファが一杯になる書き込みサイクルからフラグが LOW にアサートされる時点までの時間です。4 番目のクロック エッジで、外部マスターは LOW になっているフラグをサンプリングできます。これを図 4 に示します。

読み出し転送を実行する時、常にフラグ用の 2 サイクル遅延は転送の終了時に生じます。2 サイクルの遅延は、バッファが空になる読み出し (最後の SLRD#アサート)サイクルからフラグが LOW にアサートされる時点までの時間です。3 番目のクロックエッジで、外部マスターは LOW になっているフラグをサンプリングできます。これを図 3 に示します。

8.2.1. 部分的フラグ

フラグは、ソケットの部分的に空／一杯の状態を示すように設定できます。ウォーターマーク値は、読み出しまたは書き込み可能な 32 ビット ワード数がウォーターマーク値以下の場合にはフラグがアサートされるように選択する必要があります。

注: 部分的フラグ用の遅延は、そのフラグ用に指定されたウォーターマーク値に依存します。

表 4 で、異なるフラグ コンフィギュレーションを使用する時の遅延をまとめます。またこの表は、特定のフラグ用に GPIF II Designer で選択する設定も示します。 フラグの GPIF II Designer での設定の例と画面をフラグのコンフィギュレーションに示します。

表 4. 異なるフラグのコンフィギュレーションに対応する遅延

フラグのコンフィギュレーション	GPIF II Designer でのフラグ設定の選択	転送開始時のアドレスからフラグまでの遅延	転送終了時のフラグ遅延		必要な追加の API 呼び出し
			スレーブ FIFO への書き込み転送の場合 (最後の SLWR# アサートから一杯フラグ アサートまでの遅延)	スレーブ FIFO からの読み出し転送の場合 (最後の SLRD# アサートから空フラグ アサートまでの遅延)	
特定のスレッド「n」専用の一杯／空フラグ	Thread_n_DMA_Ready	0 サイクル	3 サイクル + t_{CFLG} (外部デバイスは 4 番目のクロック エッジで有効なフラグをサンプリング可能)	2 サイクル + t_{CFLG} (外部デバイスは 3 番目のクロック エッジで有効なフラグをサンプリング可能)	該当なし
現時点でアドレス指定されているスレッドの一杯／空フラグ	Current_thread_DMA_Ready	2 サイクル + t_{CFLG} (外部デバイスは 3 番目のクロック エッジで有効なフラグをサンプリング可能)	3 サイクル + t_{CFLG} (外部デバイスは 4 番目のクロック エッジで有効なフラグをサンプリング可能)	2 サイクル + t_{CFLG} (外部デバイスは 3 番目のクロック エッジで有効なフラグをサンプリング可能)	該当なし
特定のスレッド「n」専用の部分的に一杯／空フラグ	Thread_n_DMA_Watermark	0 サイクル	ウォーターマーク レベルに依存	ウォーターマーク レベルに依存	CyU3PGpifSocketConfigure () API を呼び出すことでウォーターマーク レベルを設定 **注:** ウォーターマーク値は 32 ビット データ ワードの単位 例: CyU3PGpifSocketConfigure (0,PIB_SOCKET_0,4,CyFalse,1) はスレッド 0 のウォーターマーク レベルを 4 に設定 CyU3PGpifSocketConfigure (3,PIB_SOCKET_3,4,CyFalse,1) はスレッド 3 のウォーターマーク レベルを 4 に設定
現時点でアドレス指定されているスレッドの部分的に一杯／空フラグ	Current_thread_DMA_Watermark	2 サイクル + t_{CFLG} (外部デバイスは 3 番目のクロック エッジで有効なフラグをサンプリング可能)	ウォーターマーク レベルに依存	ウォーターマーク レベルに依存	CyU3PGpifSocketConfigure() API を呼び出すことでウォーターマーク レベルを設定 **注:** ウォーターマーク値は 32 ビット データ ワードの単位 例: CyU3PGpifSocketConfigure (0,PIB_SOCKET_0,4,CyFalse,1) はスレッド 0 のウォーターマーク レベルを 4 に設定 CyU3PGpifSocketConfigure (3,PIB_SOCKET_3,4,CyFalse,1) はスレッド 3 のウォーターマーク レベルを 4 に設定

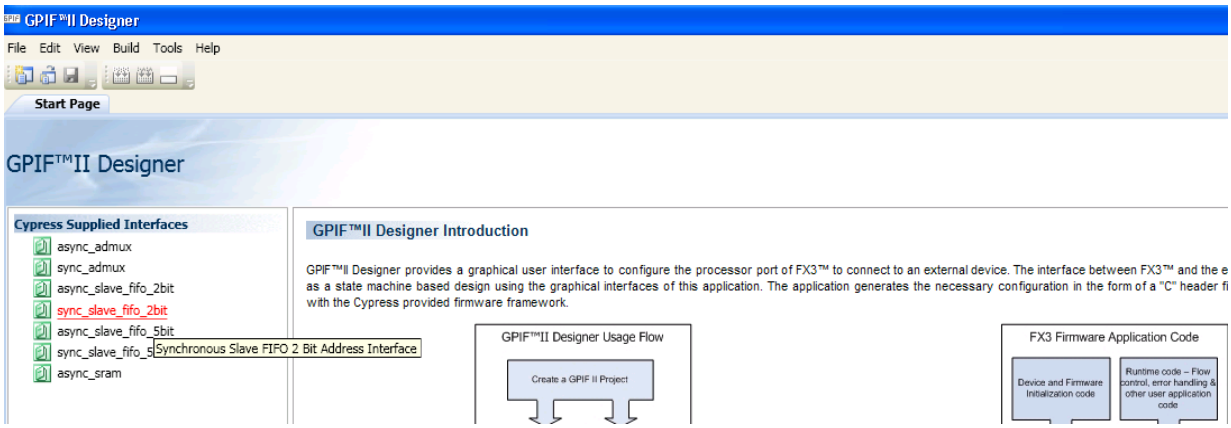
以下の節で、GPIF II Designer ツールと EZ-USB FX3 SDK を用いてフラグを設定する方法を説明します。

9. GPIF II Designer

9.1. 同期スレーブ FIFO インターフェースの実装

GPIF II Designer ツールをインストールした後、GPIF II によるスレーブ FIFO インターフェースの実装例を使用できます。GPIF II Designer ツールをサイプレスのウェブサイト www.cypress.com からインストールできます。GPIF II Designer を起動すると、スタート ページに「**Cypress Supplied Interfaces**」があります。

図 7. GPIF II Designer にあるスレーブ FIFO プロジェクトーCypress Supplied Interfaces



sync_slave_fifo_2bit プロジェクトは、2 ビット アドレスを使った同期スレーブ FIFO インターフェースの GPIF II による実装です。次の節で、GPIF II Designer を用いて部分的フラグを設定する方法を説明します。

9.2. 部分的フラグの設定

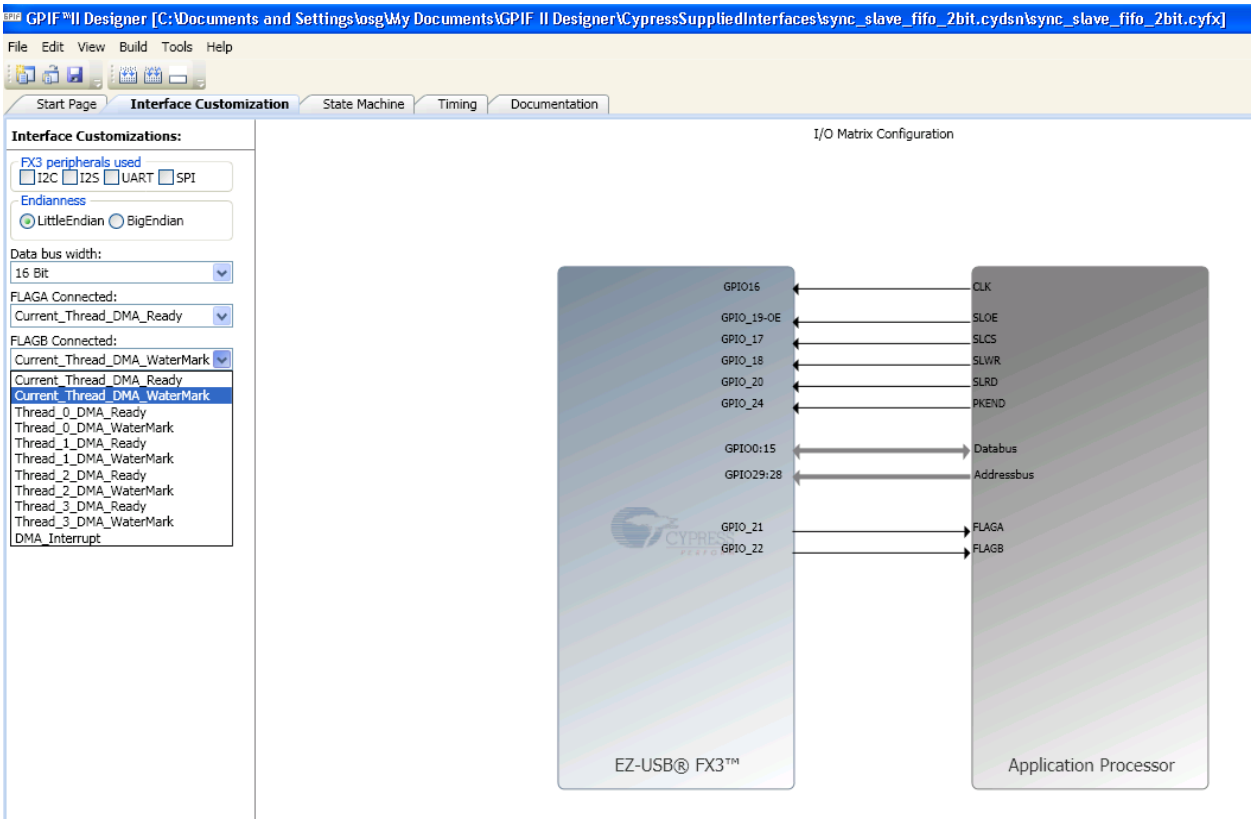
部分的フラグの設定手順は以下のとおりです。

- GPIF II Designer ツールで部分的フラグの設定を選択します。
- ファームウェアで CyU3PGpifSocketConfigure() API を使用して部分的フラグのウォーターマーク レベルを設定します。

GPIF II Designer で、「Cypress Supplied Interfaces」から **sync_slave_fifo_2bit** プロジェクトを開きます。フラグを現時点のスレッド用の部分的フラグに設定するために、「FLAGA Connected」または「FLAGB Connected」の下で **Current_Thread_DMA_Watermark** を選択します。

あるいは、FLAG をスレッド「n」専用の部分的 FLAG に設定するために、**Thread_n_DMA_Watermark** を選択します。

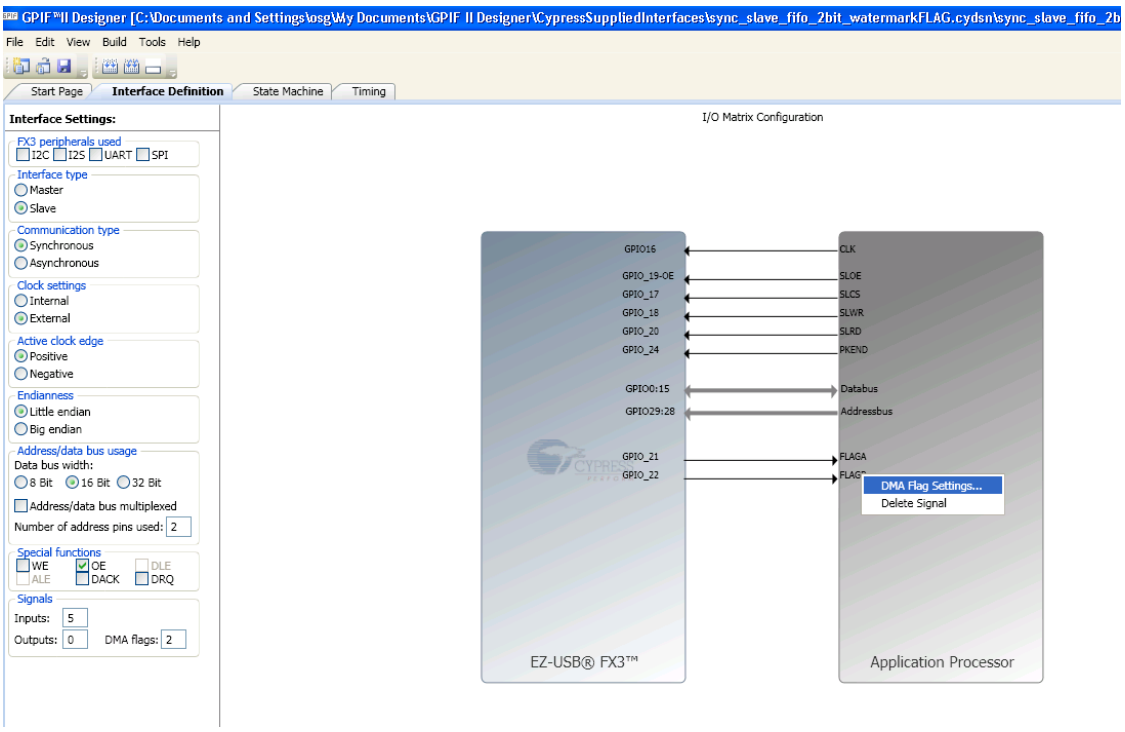
図 8. GPIF II Designer でのフラグ設定—Cypress Supplied Interfaces の sync_slave_fifo_2bit



sync_slave_fifo_2bit.cyfx プロジェクトでのもの以外、フラグをさらに追加するか変更するために、**File** > **Save Project as Editable** をクリックしてください。これにより、プロジェクトを別の名前で保存できます。この後、新たに保存されたプロジェクトを変更することができます。

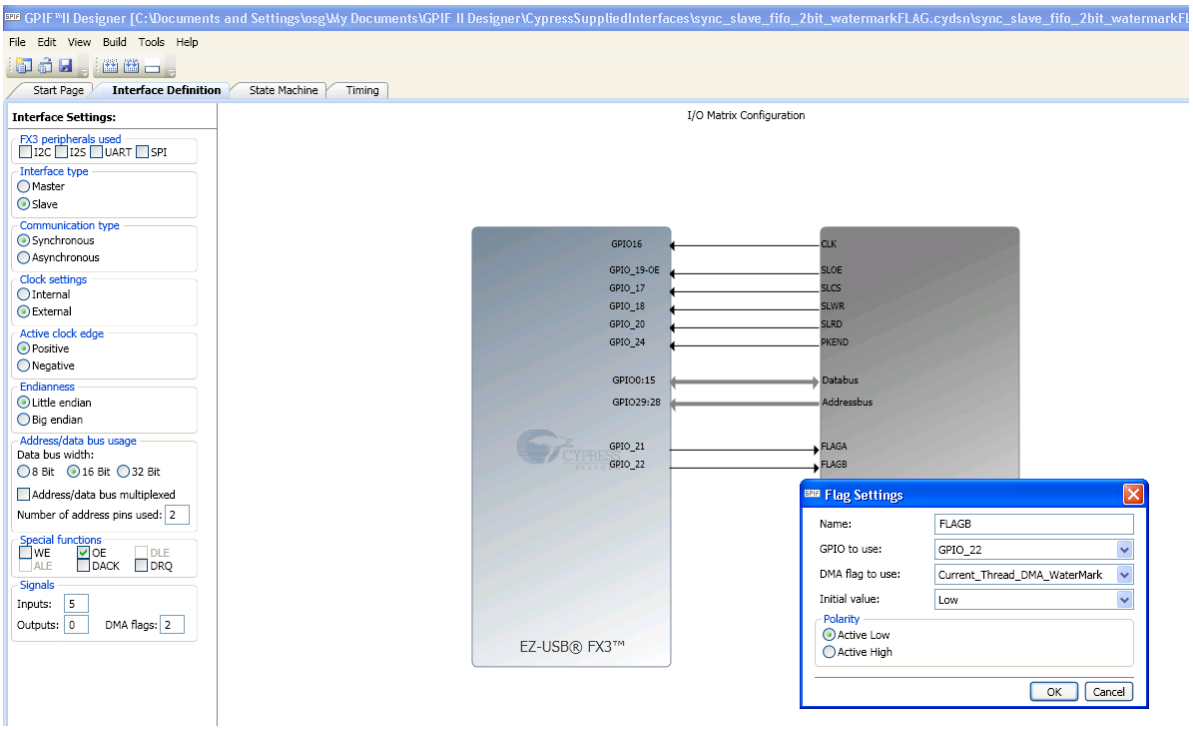
この場合、フラグを部分的フラグに設定するために、「I/O Matrix Configuration」図でフラグを右クリックします。下図のように **DMA Flag Settings** をクリックします。

図 9. デフォルトの sync_slave_fifo_2bit での「Save Project as Editable」による新規プロジェクトのフラグ設定



下図にフラグ設定の選択方法を示します。

図 10. 特定のフラグ設定の選択



部分的フラグを設定する第 2 ステップは、ファームウェア プロジェクトでそのフラグのウォーターマーク値を指定することです。ウォーターマーク値を指定するために、*cyfxslfifo.c* ファイルに CyU3PGpifSocketConfigure() API への呼び出しを追加します。この呼び出しは、CyU3PGPIFLoad() API への呼び出しの直後に追加できます。CyU3PGpifSocketConfigure() API の完全な説明については、EZ-USB FX3 SDK API ガイドをご参照ください。この API へのパラメーター入力の 1 つはウォーターマーク値です。

ウォーターマーク値は部分的フラグがアサートされる時点を決めます。次の節で、部分的フラグがアサートされた後に読み書き可能なデータ ワード数の計算式について説明します。

9.3. 部分的フラグの一般式

上記では、フラグがとりえる設定および部分的フラグの設定手順について説明しました。ここでは、部分的フラグのウォーターマーク値を決定する方法を説明します。

次の式は、部分的フラグがアサートされた後に読み書きデータ ワード数を計算するために使用します。

注: CyU3PGpifSocketConfigure() API で指定したウォーターマーク値は 32 ビット データ ワードの単位です。

1. 外部マスターから同期スレーブ FIFO へ書き込む場合:

   (a) LOW になっている部分的フラグがサンプリングされるクロック エッジの後に書き込み可能なデータ ワード数 = ウォーターマーク x (32/バス幅) – 4

2. 外部マスターへ同期スレーブ FIFO から読み出す場合:

   (a) アサートされている部分的フラグがサンプリングされるクロック エッジの後に読み出し可能なデータ ワード数 (SLOE#がアサートされたまま) = ウォーターマーク x (32/バス幅) – 1
   (b) SLRD#からデータまでの 2 サイクルの遅延が既にあります。従って、アサートされている部分的フラグがサンプリングされるクロック エッジの後に SLRD#がアサートされたままのサイクル数 = ウォーターマーク x (32/バス幅) – 3

9.4. CyU3PgpifSocketConfigure() API の使用例

本節に、CyU3PGpifSocketConfigure() API を使って指定したウォーターマーク値の影響の例を示します。異なるウォーターマーク値に対応する部分的フラグの動作を明確に示すために画面を提供します。

注: これらの例は、フラグの極性がアクティブ LOW にセットされている場合のものです。そのため、フラグは一杯／空または部分的に一杯／空を示すために LOW になります。

9.4.1. 例 1

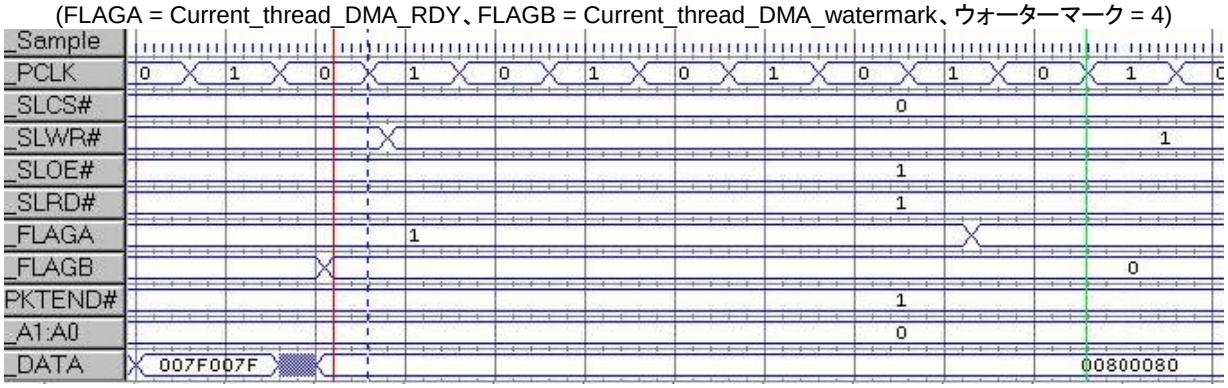
32 ビット データ バスを使うスレーブ FIFO

- GPIF II Designer で、FLAGA は Current_thread_DMA_RDY に、FLAGB は Current_thread_DMA_watermark に設定されています。
- CyU3PGpifSocketConfigure (0, PIB_SOCKET_0, 4, CyFalse, 1)。
- バースト書き込みは、外部 FPGA からスレーブ FIFO を介して EZ-USB FX3 へ実行されます (最後に書き込まれるデータは 0x00800080 です)。

下図は、フラグが転送終了時にどのように 0 になるかを示すロジック アナライザの画面です。図を見ると、FLAGB (部分的フラグ) は最後のデータ ワードが書き込まれるサイクルと同じサイクルで LOW になることが分かります。

図 11. 32 ビット データ バス幅のバースト書き込み転送

(FLAGA = Current_thread_DMA_RDY、FLAGB = Current_thread_DMA_watermark、ウォーターマーク = 4)



9.4.2. 例 2

32 ビット データ バスを使うスレーブ FIFO

- GPIF II Designer で、FLAGA は Current_thread_DMA_RDY に、FLAGB は Current_thread_DMA_watermark に設定されています。
- CyU3PgpifSocketConfigure (3, PIB_SOCKET_3, 4, CyFalse, 1)。
- バースト読み出しは、外部 FPGA によって EZ-USB FX3 からスレーブ FIFO を介して実行されます (最後に読み出されるデータは 0x00000080 です)。

下図は、フラグが転送終了時にどのように 0 になるかを示すロジック アナライザの画面です。図を見ると、最後のデータが読み出される前に FLAGB (部分的フラグ) が 4 サイクルの間 LOW になることが分かります。つまり、FLAGB が LOW になるサイクルの後、3 つのデータ ワードが読み出し可能です。

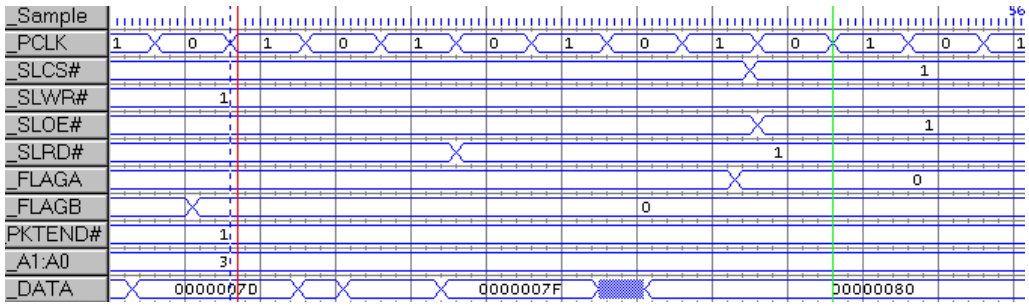
部分的フラグの一般式からの式 2(a) を次のように本例に適用できます。

ウォーターマーク値 = 4、バス幅 = 32

したがって、アサートされている部分的フラグがサンプリングされるクロック エッジの後に読み出し可能な 32 ビット データ ワード数 = 4 x (32/32) – 1 = 3

図 12. 32 ビット データ バス幅のバースト読み出し転送

(FLAGA = Current_thread_DMA_RDY、FLAGB = Current_thread_DMA_watermark、ウォーターマーク = 4)



9.4.3. 例 3

16 ビット データ バスを使うスレーブ FIFO

- GPIF II Designer で、FLAGA は Current_thread_DMA_RDY に、FLAGB は Current_thread_DMA_watermark に設定されています。
- CyU3PgpifSocketConfigure (0, PIB_SOCKET_3, 3, CyFalse, 1)。

- バースト書き込みは、外部 FPGA からスレーブ FIFO を介して EZ-USB FX3 へ実行されます (最後に書き込まれるデータは 0x0200 です)。

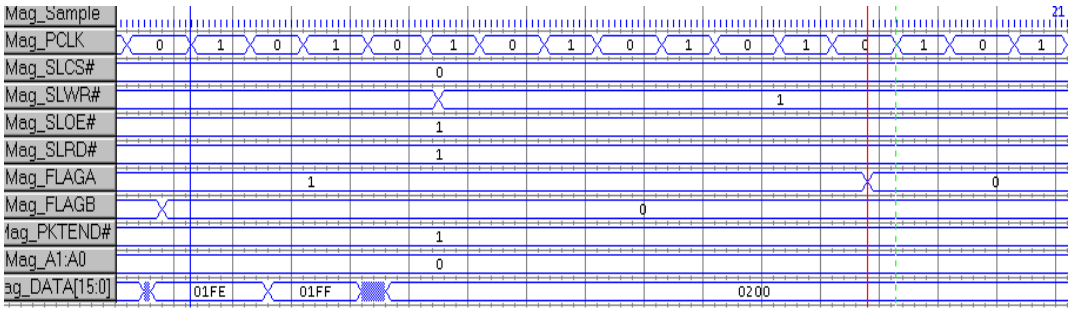
下図は、フラグが転送終了時にどのように 0 になるかを示すロジック アナライザの画面です。図を見ると、最後のデータの前に FLAGB (部分的 FLAG) が 3 サイクルの間 LOW になることが分かります。つまり、FLAGB が LOW になるサイクルの後、2 つのデータ ワードが書き込み可能です。

部分的フラグの一般式からの式 1 を次のように本例に適用できます。

ウォーターマーク値 = 3、バス幅 = 16

したがって、アサートされている部分的フラグがサンプリングされるクロック エッジの後に書き込み可能な 16 ビット データ ワード数 = 3 x (32/16) – 4 = 2

図 13. 16 ビット データ バス幅のバースト書き込み転送
(FLAGA = Current_thread_DMA_RDY、FLAGB = Current_thread_DMA_watermark、ウォーターマーク = 3)



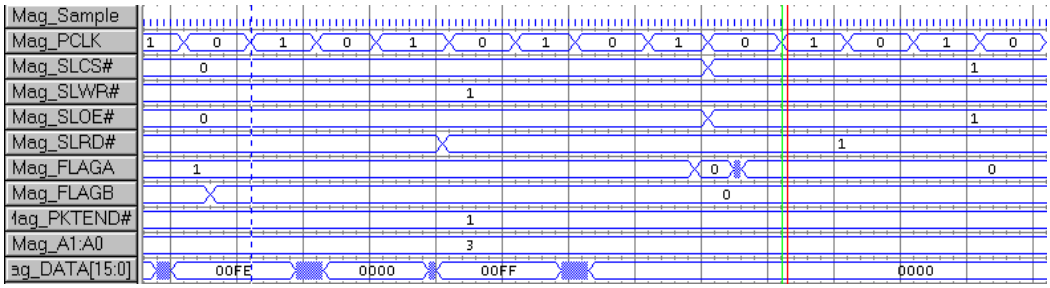
9.4.4. 例 4

16 ビット データ バスを使うスレーブ FIFO

- GPIF II Designer で、FLAGA は Current_thread_DMA_RDY に、FLAGB は Current_thread_DMA_watermark に設定されています。
- CyU3PGpifSocketConfigure (3, PIB_SOCKET_3, 2, CyFalse, 1)。
- バースト読み出しは、外部 FPGA によって EZ-USB FX3 からスレーブ FIFO を介して実行されます (最後に読み出されるデータは 0x0000 です)。

下図は、フラグが転送終了時にどのように 0 になるかを示すロジック アナライザの画面です。図を見ると、最後のデータの前に FLAGB (部分的 FLAG) が 4 サイクルの間 LOW になることが分かります。つまり、FLAGB が LOW になるサイクルの後、3 つのデータ ワードが読み出し可能です。

図 14. 16 ビット データ バス幅のバースト読み出し転送
(FLAGA = Current_thread_DMA_RDY、FLAGB = Current_thread_DMA_watermark、ウォーターマーク = 2)



9.5. 部分的フラグを使用する際の他の注意事項

- 部分的フラグは、転送を終了させる時点を決めるためにのみ使用される場合があります。一杯／空フラグは、ソケットの使用可能性を確かめるために転送の開始時に監視しなければなりません。つまり、部分的フラグそのものだけ使用できず、一杯／空フラグと共に使用する必要があります。
- 外部マスターがカウント メカニズムを実装し、常に EZ-USB FX3 の DMA バッファのサイズに等しいデータ量を書き込む場合、全く部分的フラグを使用しない場合でも問題ありません。外部マスターは、読み書きされているデータをカウントし、DMA チャネルを作成する時にセットアップしたバッファ サイズを超えないようにする必要があります。この場合、転送の開始時点を決めるために一杯／空フラグを監視する必要があります。
- 上記のステップのようにカウント メカニズムが実装されておらず、部分的フラグが使用されている場合は、以下のステップのいずれかを行ってください。
  - 外部マスターが常に一定のデータ量をバーストで転送する場合、このバースト サイズはウォーターマーク値を選択する際に考慮しなければなりません。バースト値は、最後のパラメーターとして CyU3PgpifSocketConfigure()API で設定できます。バースト値を設定することにより、DMA ハードウェアは、1 ワードごとにではなく、バーストごとにウォーターマーク値をチェックします。例えば、外部マスターが常に 8 ワードのバーストで書き込む場合、ウォーターマーク値は、EZ-USB FX3 の DMA バッファに 8 ワードのバースト用の空間があると部分的フラグが LOW になるよう選択します。外部マスターは部分的フラグが LOW になることを確認したら、完全な 8 ワードのバーストを書き込むことができます。これを実現するに,16 ビット モードでの書き込み方向 (スレーブ FIFO へ) では、「9.3」節で説明した式に従って、CyU3PgpifSocketConfigure()API でウォーターマーク値を 6 にセットする必要があります。
  - 上記のステップに代わるものとして、部分的フラグが 0 になった後、外部マスターはバースト アクセスを行わず、シングル サイクル モードに切り替えることができます。外部マスターは各サイクルで、書き込みを行う前に、一杯／空フラグをチェックしてバッファにはまだ空き容量があることを確かめられます。

9.6. フラグ違反によるエラー状態

部分的フラグまたは一杯／空フラグでバッファが使用不可能であることを示す場合、スレーブ FIFO インターフェースに対してデータ読み書きアクセスを行わないでください。

空のバッファに対し読み出しアクセスを実行すると、バッファ アンダーラン エラーが生じます。一杯のバッファに対して書き込みアクセスを実行すると、バッファ オーバーラン エラーが生じます。これらのエラーは、バッファ境界でデータの破損を引き起こすことがあります。スレーブ FIFO インターフェースは FX3 の PIB ブロック領域にあるので、そのインターフェースに関連するすべてのエラーは PIB エラーにより示されます。

PIB エラーが生じると、PIB 割り込みがトリガーされます。FX3 SDK では、コールバック関数を PIB 割り込み用に登録できます。PIB 割り込み用のコールバック関数は PIB エラー コードをチェックできます。次のコードは、コールバック関数を登録する方法、およびアンダーラン／オーバーラン エラーをチェックしてデバッグ メッセージをプリントアウトする実際のコールバック関数の例です。

表 5 に示すに、エラー コードはエラーがどのスレッドに発生したかを示します。これらのエラーのいずれかが発生したら、ロジック アナライザを使用してインターフェースのタイミングを慎重に分析し、部分的フラグが 0 になった後に読み書きが実行されるサイクル数に注意を払うことをお勧めします。例は図 11～図 14 を参照してください。

```
/* Register a callback for notification of PIB interrupts*/
CyU3PpibRegisterCallback(gpif_error_cb,intMask);

/* Callback function to check for PIB ERROR*/
void gpif_error_cb(CyU3PpibIntrType cbType, uint16_t cbArg)
{
        if(cbType==CYU3P_PIB_INTR_ERROR)
        {
                switch(CYU3P_GET_PIB_ERROR_TYPE(cbArg))
                 {
                        case CYU3P_PIB_ERR_THR0_WR_OVERRUN:
                                CyU3PdebugPrint (4, "CYU3P_PIB_ERR_THR0_WR_OVERRUN");
                       break;
                       case CYU3P_PIB_ERR_THR1_WR_OVERRUN:
                                CyU3PdebugPrint (4, "CYU3P_PIB_ERR_THR1_WR_OVERRUN");
```

```
                    break;
                     case CYU3P_PIB_ERR_THR2_WR_OVERRUN:
                           CyU3PdebugPrint (4, "CYU3P_PIB_ERR_THR2_WR_OVERRUN");
                    break;
                    case CYU3P_PIB_ERR_THR3_WR_OVERRUN:
                           CyU3PdebugPrint (4, "CYU3P_PIB_ERR_THR3_WR_OVERRUN");
                    break;

                    case CYU3P_PIB_ERR_THR0_RD_UNDERRUN:
                           CyU3PdebugPrint (4, "CYU3P_PIB_ERR_THR0_RD_UNDERRUN");
                    break;
                    case CYU3P_PIB_ERR_THR1_RD_UNDERRUN:
                           CyU3PdebugPrint (4, "CYU3P_PIB_ERR_THR1_RD_UNDERRUN");
                    break;
                    case CYU3P_PIB_ERR_THR2_RD_UNDERRUN:
                           CyU3PdebugPrint (4, "CYU3P_PIB_ERR_THR2_RD_UNDERRUN");
                    break;
                    case CYU3P_PIB_ERR_THR3_RD_UNDERRUN:
                           CyU3PdebugPrint (4, "CYU3P_PIB_ERR_THR3_RD_UNDERRUN");
                    break;

                    default:
                           CyU3PdebugPrint (4, "No Underrun/Overrun Error");
                    break;
             }
        }
}
```

表 5. オーバーラン／アンダーラン状態用の PIB エラー コード

PIB エラー コード	説明
CYU3P_PIB_ERR_THR0_WR_OVERRUN	スレッド 0 バッファでの書き込みオーバーラン
CYU3P_PIB_ERR_THR1_WR_OVERRUN	スレッド 1 バッファでの書き込みオーバーラン
CYU3P_PIB_ERR_THR2_WR_OVERRUN	スレッド 2 バッファでの書き込みオーバーラン
CYU3P_PIB_ERR_THR3_WR_OVERRUN	スレッド 3 バッファでの書き込みオーバーラン
CYU3P_PIB_ERR_THR0_RD_UNDERRUN	スレッド 0 バッファでの読み出しアンダーラン
CYU3P_PIB_ERR_THR1_RD_UNDERRUN	スレッド 1 バッファでの読み出しアンダーラン
CYU3P_PIB_ERR_THR2_RD_UNDERRUN	スレッド 2 バッファでの読み出しアンダーラン
CYU3P_PIB_ERR_THR3_RD_UNDERRUN	スレッド 3 バッファでの読み出しアンダーラン

10. SDK におけるスレーブ FIFO ファームウェアの例

本アプリケーション ノートは、同期スレーブ FIFO インターフェースについて、およびフラグの設定方法を説明しました。GPIF II Designer ツールで必要な設定を行った後、更新された設定はファームウェアに統合する必要があります。GPIF II Designer でプロジェクトをビルドした後、ヘッダ ファイル *cyfxgpifconfig.h* が生成されます。このヘッダ ファイルはファームウェア プロジェクトに含める必要があります。EZ-USB FX3 SDK は、スレーブ FIFO インターフェースを統合したファームウェアの例を含んでいます。

EZ-USB FX3 SDK をインストールした後、同期スレーブ FIFO インターフェースを統合したファームウェアの例は次のディレクトリにあります。*[FX3 SDK Install Path]\EZ-USB FX3 SDK\1.3\firmware\slavefifo_examples\slfifosync*。

このファームウェア例は、16 ビットと 32 ビット データ バス幅をサポートしています。定数 CY_FX_SLFIFO_GPIF_16_32BIT_CONF_SELECT はヘッダ ファイル *cyfxslfifosync.h* で定義されています。32 ビット データ バス幅を選択するには、この定数を 1 にセットします。16 ビット データ バス幅を選択するには、この定数を 0 にセットします。

注: スレーブ FIFO が 32 ビット バス幅を使って 100MHz で動作するには、PLL 周波数を 400MHz に設定する必要があります。このために、setSysClk400 パラメーターを CyU3PDeviceInit() 関数の入力にセットしてください。詳細については、FX3 SDK で提供された API ガイドをご参照ください。

11. 設計例 1: Xilinx FPGA を FX3 の同期スレーブ FIFO インターフェースと連結

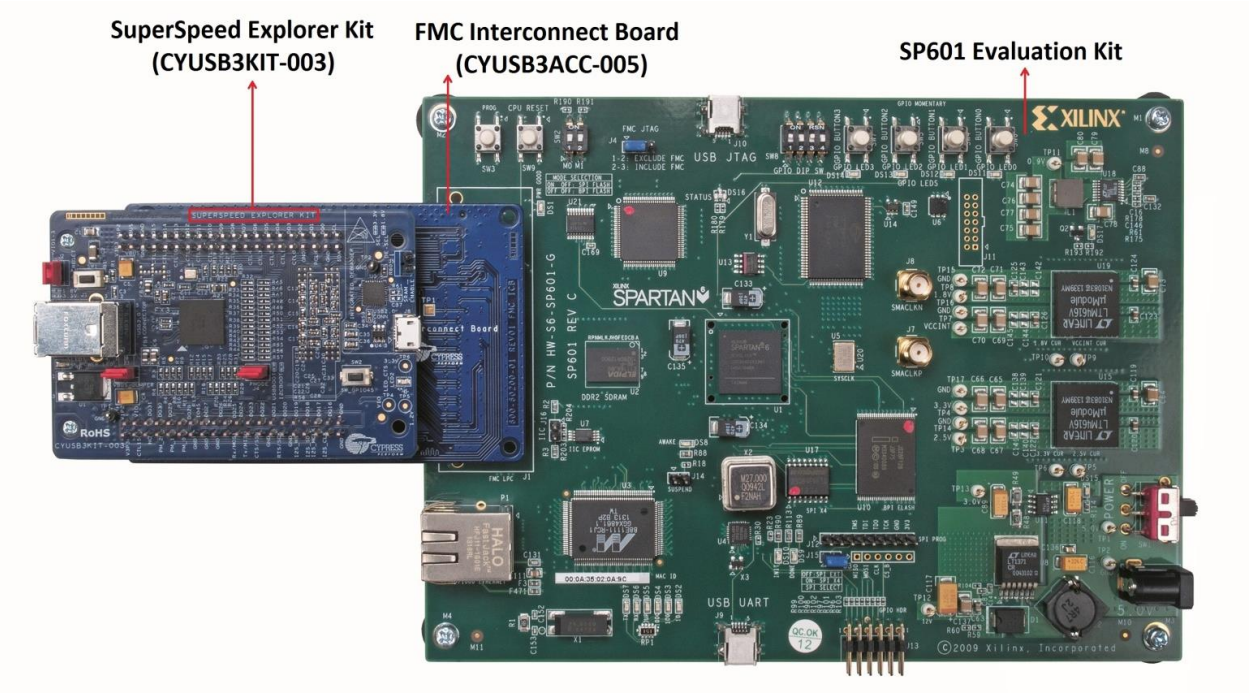
本節で、Xilinx Spartan 6 FPGA が同期スレーブ FIFO インターフェースを介して FX3 に接続する完全な設計例を提供します。この設計を実装するために使うハードウェア、ファームウェアおよびソフトウェアのコンポーネントについて説明します。

11.1. ハードウェアのセットアップ

本例で提供されたプロジェクトは、Xilinx Spartan 6 SP601 評価キットと相互接続しているサイプレス FX3 開発キット (CYUSB3KIT-001) または SuperSpeed Explorer Kit (CYUSB3KIT-003) から構成されるハードウェアのセットアップで実行されます。FX3 基板と Xilinx 基板は、Samtec—FMC 相互接続基板を使用して接続されています。CYUSB3ACC-002 相互接続基板は、サイプレス FX3 開発キット (CYUSB3KIT-001)上の Samtec コネクタと Xilinx 基板上の FMC コネクタを接続します。

CYUSB3ACC-005 相互接続基板は、SuperSpeed Explorer Kit (CYUSB3KIT-003) 上のヘッダと Xilinx 基板上の FMC コネクタを接続します。図 15 に、SuperSpeed Explorer Kit を使用したハードウェア セットアップを示します。付録 B: FX3 DVK (CYUSB3KIT-001) を用いたハードウェア セットアップを参照してください。ハードウェア セットアップ以外に、以下のステップはご使用の FX3 基板にかかわらず同じです。

図 15. FMC 相互接続基板を用いて Xilinx SP601 基板に接続したサイプレス SuperSpeed Explorer Kit



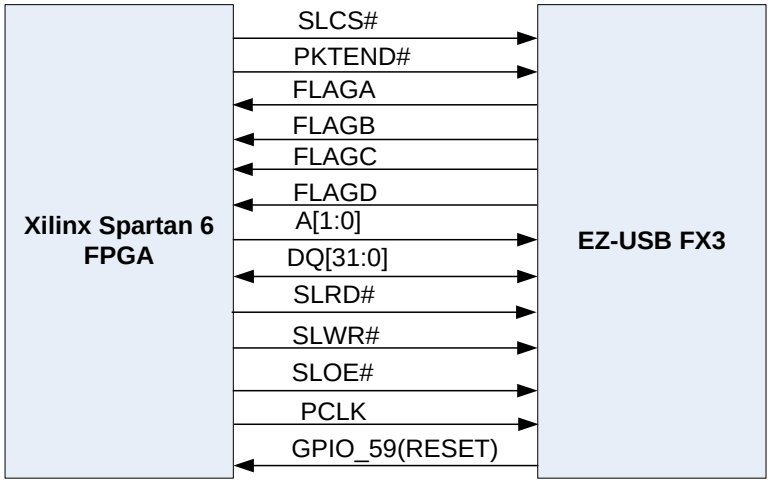
注: FPGA がスレーブ FIFO インターフェースを介して FX3 とインターフェースするすべてのアプリケーションで、SuperSpeed Explorer Kit 上のジャンパ J5 が開いていることを確かめてください。

11.2. ファームウェアとソフトウェア コンポーネント

- FX3 SDK 同梱の FX3 同期スレーブ FIFO ファームウェア プロジェクト
- FX3 SDK 同梱の Control Center と Streamer ソフトウェア ユーティリティ

下図に、FPGA と FX3 の相互接続図を示します。

図 16. FPGA と FX3 の相互接続図



この例には、以下のコンポーネントがあります。

- ループバック転送: このコンポーネントでは、FPGA は FX3 から完全なバッファを読み出してから、その内容を FX3 に書き戻します。USB ホストはこのデータを送受信するために、OUT／IN トークンを発行する必要があります。この目的には、EZ-USB FX3 SDK 同梱の Control Center ユーティリティを使用できます。
- ショートパケット: このコンポーネントでは、FPGA はフルパケットに続いてショートパケットを FX3 へ転送します。USB ホストはこのデータを受信するために、IN トークンを発行する必要があります。
- 長さゼロのパケット (ZLP) 転送: このコンポーネントでは、FPGA はフルパケットに続いて長さゼロのパケットを FX3 へ転送します。USB ホストはこのデータを受信するために、IN トークンを発行する必要があります。
- ストリーム (IN) データ転送: このコンポーネントでは、FPGA は単方向転送を行う、即ち、同期スレーブ FIFO を介して FX3 に連続的にデータを書き込みます。USB ホストはこのデータを受信するために、IN トークンを発行する必要があります。この目的には、EZ-USB FX3 SDK 同梱の Control Center または Streamer ユーティリティを使用できます。
- ストリーム (OUT) データ転送: このコンポーネントでは、FPGA は単方向転送を行う、即ち、同期スレーブ FIFO を介して FX3 から連続的にデータを読み出します。USB ホストはこのデータを提供するために、OUT トークンを発行する必要があります。この目的には、EZ-USB FX3 SDK 同梱の Control Center または Streamer ユーティリティを使用できます。

11.3. FX3 ファームウェアの詳細

FX3 ファームウェアは、FX3 SDK 同梱のサンプル プロジェクトに基づきます。ファームウェアの主な特長は以下のとおりです。

- USB 3.0 と USB 2.0 の両方に対応しています。
- サイプレスの VID/PID (0x04B4/0x00F1) でエニュメレーションされます。これにより、USB 転送を開始するためにサイプレスの Control Center と Streamer ユーティリティの使用が可能です。
- 次の特長を持つ同期スレーブ FIFO ディスクリプタを統合します。
  - 最大 4 個のソケットまでサポート
  - 最大 32 ビットまでのデータ バスを設定可能
  - 100MHz の PCLK 入力クロックで動作
  - 以下の 4 つのフラグを設定:
    - i. FLAGA: スレッド 0 専用の一杯フラグ
    - ii. FLAGB: スレッド 0 専用の、ウォーターマーク値 6 を持つ部分的フラグ
    - iii. FLAGC: スレッド 3 専用の空フラグ
    - iv. FLAGD: スレッド 3 専用の、ウォーターマーク値 6 を持つ部分的フラグ

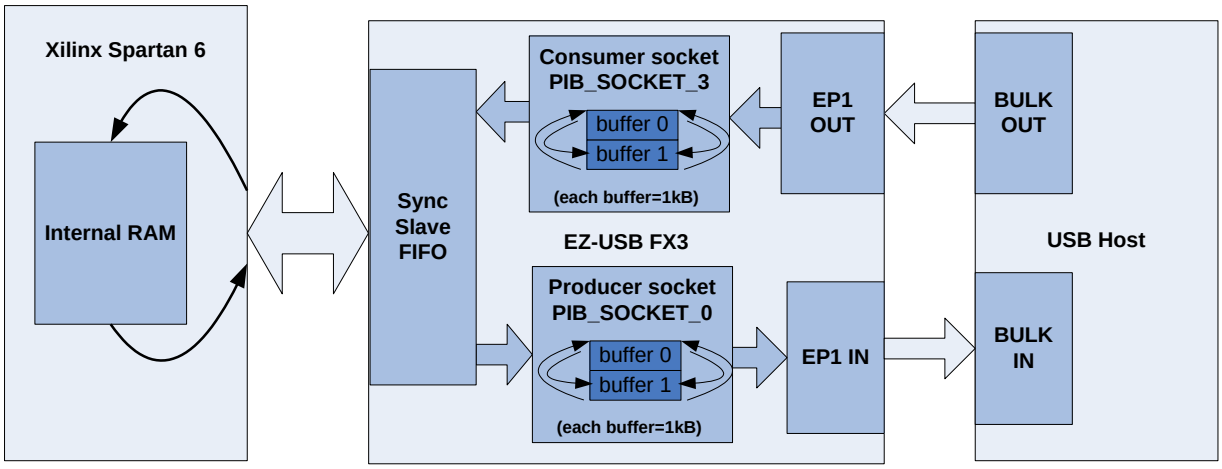
注: 本アプリケーション ノートで提供された GPIF II Designer プロジェクトはこれら設定をデモします。さらにファームウェア プロジェクトは、ウォーターマーク値を設定するために CyU3PGpifSocketConfigure() API を使用する方法を示します。

- PLL 周波数を 400MHz に設定します。これは、setSysClk400 パラメーターを CyU3PDeviceInit() 関数の入力にセットすることで行われます。

注: この設定は、スレーブ FIFO が 32 ビット データを 100MHz で動作するのに必須です。

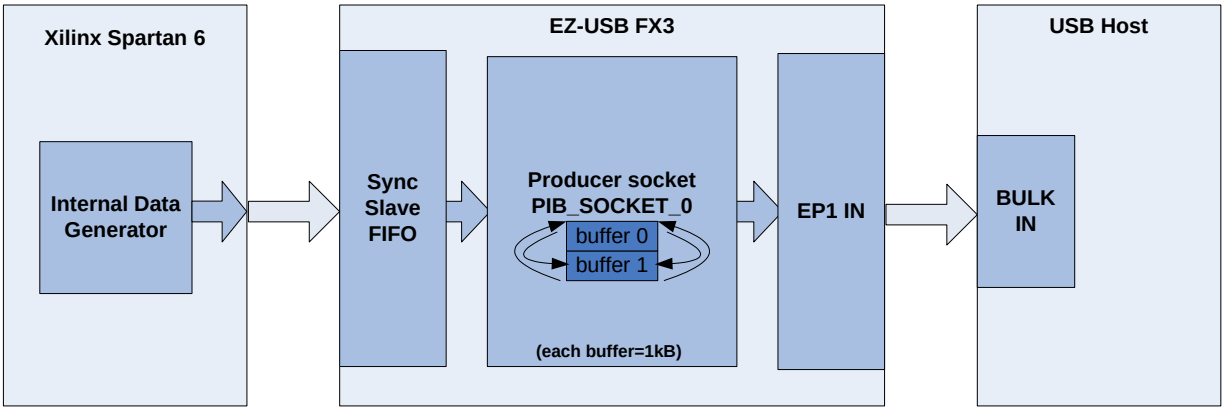
- DMA チャネルをセットアップします。
  - ループバック転送、ショートパケットおよび ZLP 転送の場合、次の 2 本の DMA チャネルが作成されます。
    - PIB_SOCKET_0 がプロデューサであり、UIB_SOCKET_1 がコンシューマである P2U チャネル。USB 接続が USB 2.0 接続か USB 3.0 接続かに応じて、DMA バッファ サイズは 512 か 1024 です。DMA バッファ数は 2 個です。
    - PIB_SOCKET_3 がコンシューマであり、UIB_SOCKET_1 がプロデューサである U2P チャネル。USB 接続が USB 2.0 接続か USB 3.0 接続かに応じて、DMA バッファ サイズは 512 か 1024 です。DMA バッファ数は 2 個です。

図 17. ループバック転送のセットアップ



注: ショートパケットと ZLP には、P2U チャネルをのみご使用ください。

図 18. ショートパケットと ZLP 転送のセットアップ

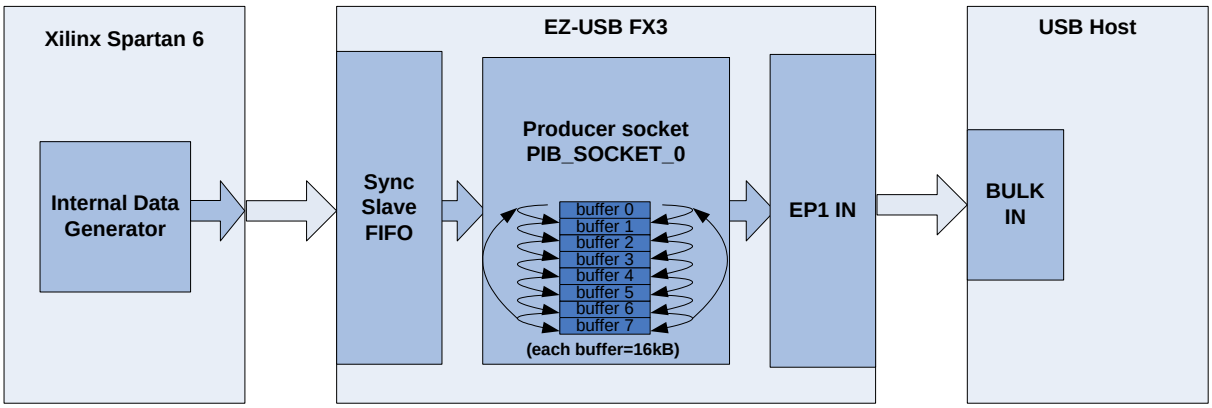


本アプリケーション ノートで提供した FX3 ファームウェア プロジェクトの *cyfxslfifosync.h* ファイルで以下の#define が有効の場合、本節で説明した DMA チャネルはセットアップされます。

```
/* set up DMA channel for loopback/short packet/ZLP transfers */
#define LOOPBACK_SHRT_ZLP
```

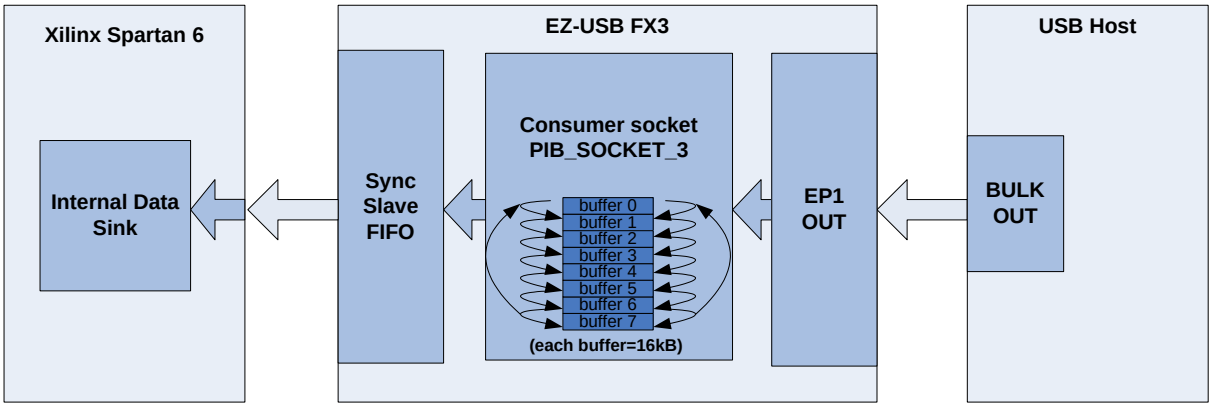
- ストリーム転送の場合、次の 2 本の DMA チャネルが作成されます。
  - PIB_SOCKET_0 がプロデューサであり、UIB_SOCKET_1 がコンシューマである P2U チャネル。DMA バッファ サイズは 16×1024 (USB 3.0 接続の場合) または 16×512 (USB 2.0 接続の場合) です。DMA バッファ数は 8 個です。このバッファのサイズと数は、高スループット性能を達成することを目的として選択されます。

図 19. ストリーム IN 転送用のセットアップ – 性能に最適化されたバッファの数とサイズ



  - PIB_SOCKET_3 がコンシューマであり、UIB_SOCKET_1 がプロデューサである U2P チャネル。DMA バッファ サイズは 16×1024 (USB 3.0 接続の場合) または 16×512 (USB 2.0 接続の場合) です。DMA バッファ数は 4 個です。性能を向上するためにバッファ数を増加できますが、P2U チャネルのバッファ数は減らす必要があります。これは、FX3 SDK は両方のチャネルが 8 個の 16×1024 バッファを持つようの十分なバッファ メモリを提供していないからです。

図 20. ストリーム OUT 転送用のセットアップ – 性能に最適化されたバッファの数とサイズ



本アプリケーション ノートで提供した FX3 ファームウェア プロジェクトの *cyfxslfifosync.h* ファイルで以下の#define が有効の場合、本節で説明した DMA チャネルはセットアップされます。

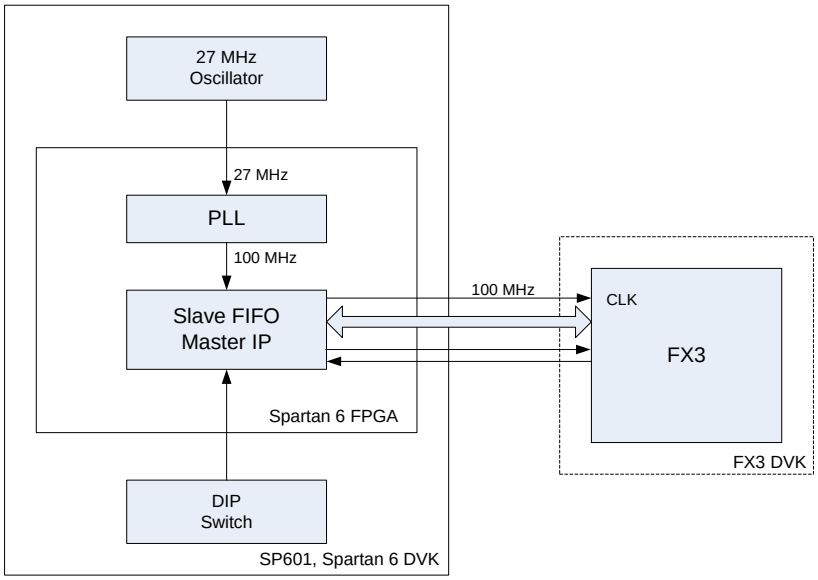
```
/* set up DMA channel for stream IN/OUT transfers */
#define STREAM_IN_OUT
```

P2U と U2P DMA チャネルに割り当てられたバッファ数は、以下の#define を使用して制御できます (これらの定義は *cyfxslfifosync.h* ファイルにもあります)。

```
/* Slave FIFO P_2_U channel buffer count */
#define CY_FX_SLFIFO_DMA_BUF_COUNT_P_2_U      (4)
/* Slave FIFO U_2_P channel buffer count */
#define CY_FX_SLFIFO_DMA_BUF_COUNT_U_2_P      (8)
```

11.4. FPGA 実装の詳細

図 21. SP601 評価キットを使用した Xilinx Spartan 6 (XC6SLX16) FPGA 実装



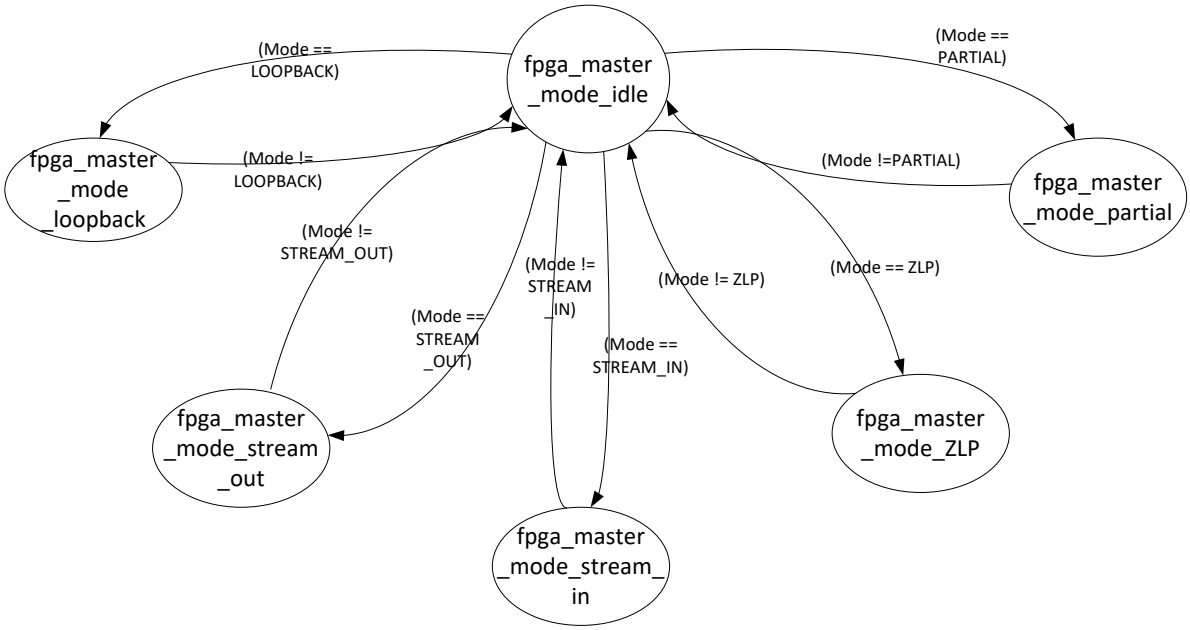
FX3 の最高性能をデモするために、GPIF インターフェースを 100MHz で動作させます。SP601 は、27MHz シングルエンド発振器を搭載しています。FPGA は、PLL を使用して 27MHz クロックから 100MHz クロックを生成します。

各種の転送のステートマシン実装は以下の節で説明します。

11.4.1. FPGA マスター モードのステートマシン

ステートマシンは、FPGA マスターの転送モードを選択するために実装されます。5 つの転送モードは、ループバック, ショートパケット (部分的), 長さゼロのパケット, ストリーム IN とストリーム OUT 転送です。

図 22. モード選択用の FPGA ステートマシン



fpga_master_mode_idle 状態:

転送モードが選択されていない場合、FPGA マスターはこの状態のままです。

fpga_master_mode_partial 状態:

mode = PARTIAL の場合、ステートマシンはこの状態に入ります。mode ! = PARTIAL の場合、ステートマシンはこの状態から fpga_master_mode_idle 状態に入ります。

fpga_master_mode_zlp 状態:

mode = ZLP の場合、ステートマシンはこの状態に入ります。mode ! = ZLP の場合、ステートマシンはこの状態から fpga_master_mode_idle 状態に入ります。

fpga_master_mode_stream_in 状態:

mode = STREAM_IN の場合、ステートマシンはこの状態に入ります。mode ! = STREAM_IN の場合、ステートマシンはこの状態から fpga_master_mode_idle 状態に入ります。

fpga_master_mode_stream_out 状態:

mode = STREAM_OUT の場合、ステートマシンはこの状態に入ります。mode ! = STREAM_OUT の場合、ステートマシンはこの状態から fpga_master_mode_idle 状態に入ります。

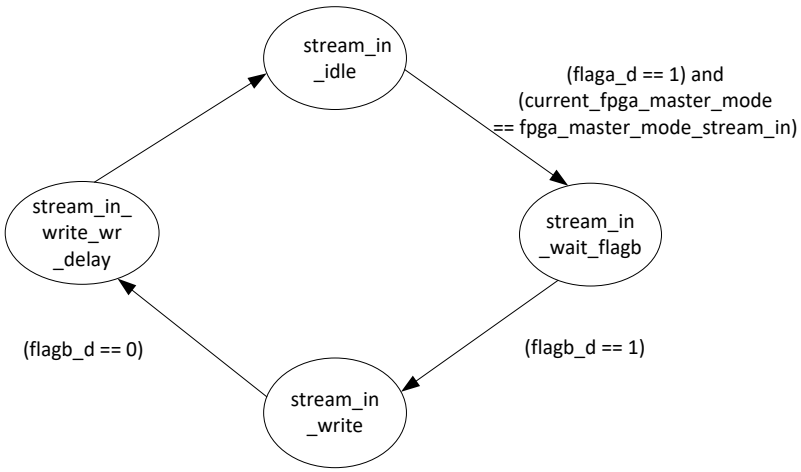
fpga_master_mode_loop_back 状態:

mode = LOOPBACK の場合、ステートマシンはこの状態に入ります。mode ! = LOOPBACK の場合、ステートマシンはこの状態から fpga_master_mode_idle 状態に入ります。

11.4.2. ストリーム IN の例: FPGA がスレーブ FIFO に書き込む

Verilog RTL で実装されたストリーム IN 転送用のステートマシンは下図に示します。

図 23. ストリーム IN 用の FPGA ステートマシン



stream_in_idle 状態:

この状態では、ステートマシンで使用するすべてのレジスタと信号を初期化します。スレーブ FIFO 制御ラインの状態は下記になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

stream_in_wait_flagb 状態:

flaga_d = 1 で、FPGA マスター モードが stream_in になると、ステートマシンはこの状態に入り、flagb_d を待ちます。

stream_in_write 状態:

flagb_d = 1 になると、ステートマシンはこの状態に入り、スレーブ FIFO インターフェースへ書き込み始めます。スレーブ FIFO 制御ラインの状態は下記になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 0 ; A[1:0] = 0

11.4.3. stream_in_write_wr_delay 状態:

flagb_d = 0 になると、ステートマシンはこの状態に入ります。スレーブ FIFO 制御ラインの状態は下記になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

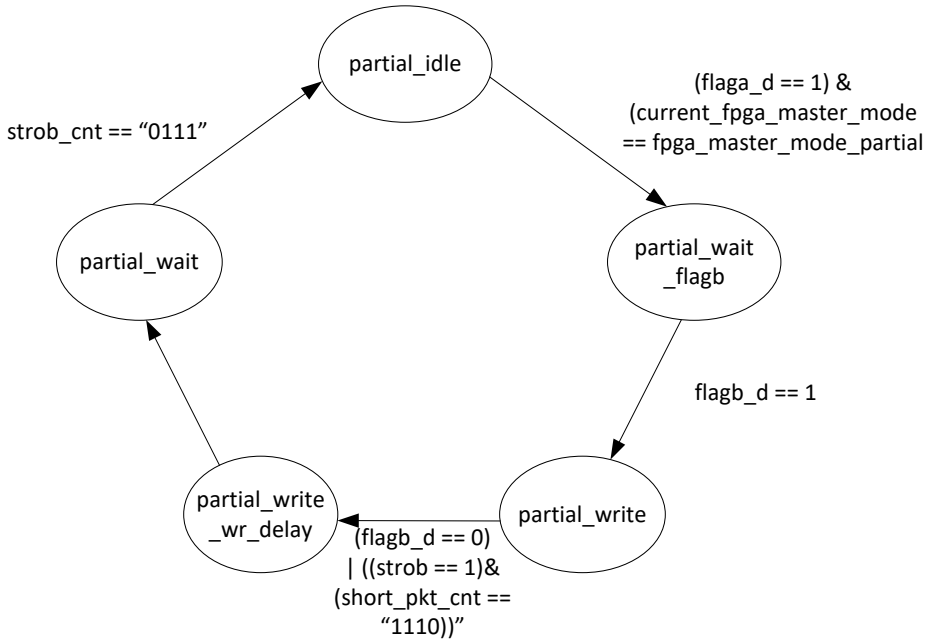
1 クロック サイクル後、ステートマシンは stream_in_idle 状態に入ります。

部分的フラグの一般式節の式 (1) によると、部分的フラグ (flagb) が 0 になった後、FX3 はアサートされている SLWR#を 2 サイクルの間サンプリングする必要があります。FPGA からインターフェースまでの伝播遅延が 1 サイクルであることを前提にすると、FPGA は flagb_d (flagb のフリップフロップされた出力) を 0 としてサンプリングした後に 1 サイクルの間 SLWR#をアサートします。

11.4.4. ショートパケットの例: FPGA がフルパケットに続いてショートパケットをスレーブ FIFO に書き込む

この例では、PKTEND#を使用したショートパケットの転送手順をデモします。Verilog RTL で実装されたショートパケット例用のステートマシンは下図に示します。

図 24. ショートパケット転送用のステートマシン



partial_idle 状態:

この状態では、ステートマシンで使用するすべてのレジスタと信号を初期化します。スレーブ FIFO 制御ラインの状態は下記になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

partial_wait_flagb 状態:

flaga_d = 1 で、FPGA マスター モードが partial になると、ステートマシンはこの状態に入ります。

partial_write 状態:

flagb_d = 1 になると、ステートマシンはこの状態に入ります。スレーブ FIFO 制御ラインの状態は下記になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 0 ; A[1:0] = 0

partial_write_wr_delay 状態:

flagb_d = 0、または strob = 1 かつ short_pkt_cnt = 「1110」になると、ステートマシンはこの状態に入ります。strob = 1 の場合、FPGA マスターはショートパケットを転送します。

スレーブ FIFO 制御ラインの状態は下記になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

1 クロック サイクル後、ステートマシンは partial_wait 状態に入ります。

部分的フラグの一般式節の式 (1) によると、部分的フラグ (flagb) が 0 になった後、FX3 はアサートされている SLWR#を 2 サイクルの間サンプリングする必要があります。FPGA からインターフェースまでの伝播遅延が 1 サイクルであることを前提にすると、FPGA は部分的 flagb_d (flagb のフリップフロップされた出力) を 0 としてサンプリングした後に 1 サイクルの間 SLWR# をアサートします。

partial_wait 状態:

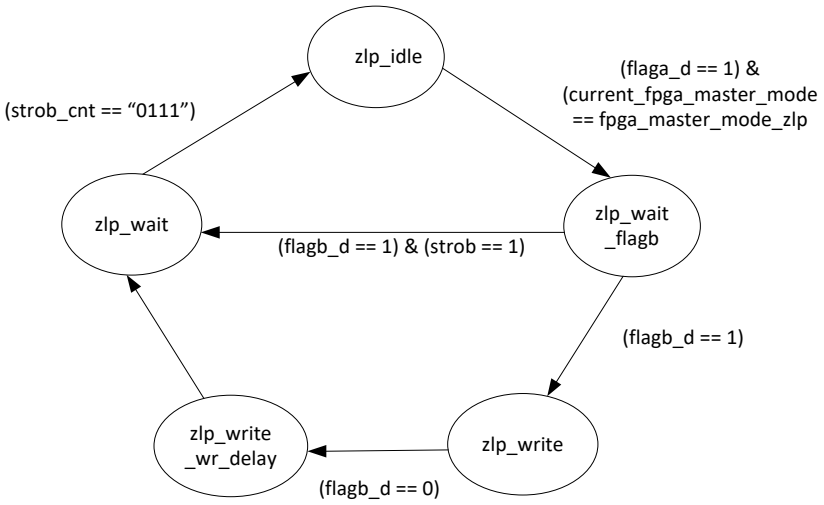
strob_cnt = 0111 になると、ステートマシンは partial_idle 状態に入ります。

ウォーターマーク値が 6 であるため、flaga は部分的フラグ (flagb) から 6 クロック サイクル後にのみ 0 になることを期待されます。この状態は flaga ステータスが確かに有効であることを保証するために、少なくとも 4 クロック サイクル以上実行されます。

11.4.5. 長さゼロのパケットの例: FPGA がフルパケットに続いて ZLP をスレーブ FIFO に書き込む

この例では、PKTEND#を使用した ZLP の転送手順をデモします。Verilog RTL で実装された ZLP 例用のステートマシンは下図に示します。

図 25. ZLP 転送用のステートマシン



zlp_idle 状態:

この状態では、ステートマシンで使用するすべてのレジスタと信号を初期化します。スレーブ FIFO 制御ラインの状態は下記になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

zlp_wait_flagb 状態:

flaga_d = 1 で、FPGA マスター モードが zlp になると、ステートマシンはこの状態に入ります。

zlp_write 状態:

flagb_d = 1 になると、ステートマシンはこの状態に入ります。スレーブ FIFO 制御ラインの状態は下記になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 0 ; A[1:0] = 0

zlp_write_wr_delay 状態:

flagb_d = 0 になると、ステートマシンはこの状態に入ります。スレーブ FIFO 制御ラインの状態は下記になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

1 クロック サイクル後、ステートマシンは zlp_wait 状態に入ります。

「部分的フラグの一般式」節の式 (1) によると、部分的フラグ (flagb) が 0 になった後、FX3 はアサートされている SLWR#を 2 サイクルの間サンプリングする必要があります。FPGA からインターフェースまでの伝播遅延が 1 サイクルであることを前提にすると、FPGA は部分的 flagb_d (flagb のフリップフロップされた出力) を 0 としてサンプリングした後に 1 サイクルの間 SLWR#をアサートします。

zlp_wait 状態:

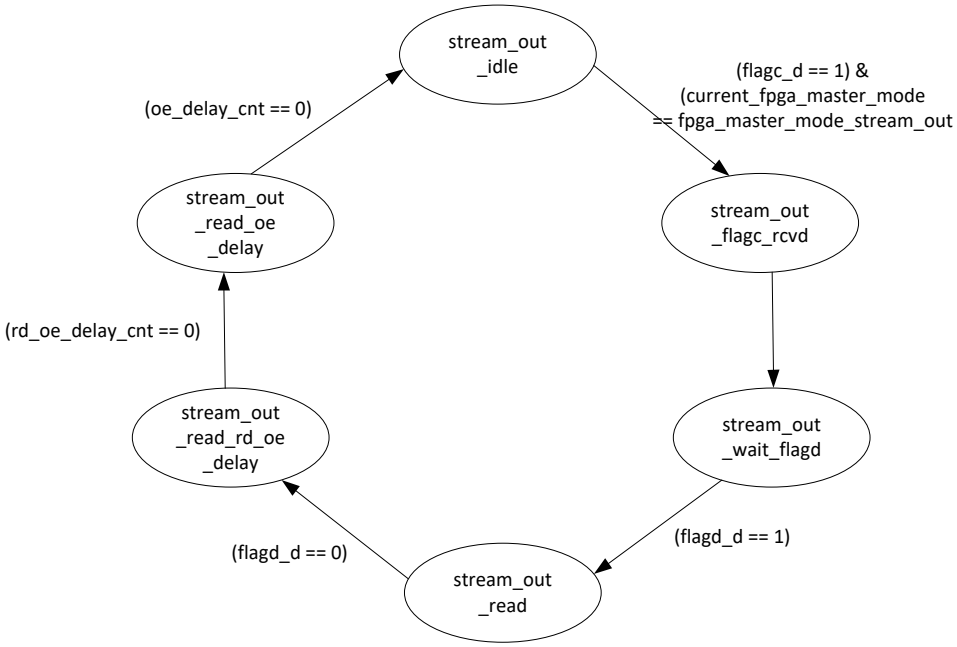
flagb_d = 1 で、strob = 1 になると、ステートマシンは zlp_wait_flagb 状態からこの状態に入り、zlp パケットを slavefifo に転送します。

ウォーターマーク値が 6 であるため、flaga は部分的フラグ (flagb) から 6 クロック サイクル後にのみ 0 になることを期待されます。この状態は flaga ステータスが確かに有効であることを保証するために、少なくとも 4 クロック サイクル以上実行されます。

strob_cnt = 0111 になると、ステートマシンは zlp_idle 状態に入ります。

11.4.6. Verilog RTL で実装されるストリーム OUT 例用のステートマシン

図 26. ストリーム OUT 転送用のステートマシン



stream_out_idle 状態:

この状態では、ステートマシンで使用するすべてのレジスタと信号を初期化します。スレーブ FIFO 制御ラインの状態は下記になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 3

stream_out_flagc_rcvd 状態:

flagc_d = 1 で、FPGA マスター モードが stream out になると、ステートマシンはこの状態に入ります。

stream_out_wait_flagd 状態:

1 クロック サイクル後、ステートマシンはこの状態に入ります。

stream_out_read 状態:

flagc_d = 1 になると、ステートマシンはこの状態に入ります。ここで、ステートマシンは以下のように読み出し制御信号をアサートします。

PKTEND# = 1 ; SLOE# = 0 ; SLRD# = 0 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 3

stream_out_read_rd_oe_delay 状態:

flagc_d = 0 になると、ステートマシンはこの状態に入ります。スレーブ FIFO 制御ラインの状態は下記になります。

PKTEND# = 1 ; SLOE# = 0 ; SLRD# = 0 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 3

「部分的フラグの一般式」節の式 (2b) によると、部分的フラグ (flagd) が 0 になった後、FX3 はアサートされている SLRD#を 3 サイクルの間サンプリングする必要があります。FPGA からインターフェースまでの伝播遅延が 1 サイクルであることを前提にすると、FPGA は flagd_d (flagd のフリップフロップされた出力) を 0 としてサンプリングした後に 1 サイクルの間 SLRD#をアサートします。

stream_out_read_oe_delay 状態:

rd_oe_delay_cnt = 0 になると、ステートマシンはこの状態に入ります。スレーブ FIFO 制御ラインの状態は下記になります。

PKTEND# = 1 ; SLOE# = 0 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 3

oe_delay_cnt = 0 の場合、ステートマシンはこの状態から stream_out_idle 状態に入ります。

11.4.7. ループバックの例: FPGA がスレーブ FIFO からデータを読み出し、そのデータをスレーブ FIFO に書き戻す

ステートマシンは、1 ループバック サイクルの間 6 つの状態を通ります。ステートマシンおよび対応する動作を下図に示します。

図 27. ループバック転送用のステートマシン

loop_back_idle 状態:

この状態では、ステートマシンで使用するすべてのレジスタと信号を初期化します。スレーブ FIFO 制御ラインの状態は下記になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 3

loop_back_flagc_rcvd 状態:

flagc_d = 1、FPGA マスター モードが loop back になると、ステートマシンはこの状態に入ります。

loop_back_wait_flagd 状態:

1 クロック サイクル後、ステートマシンはこの状態に入り、flagd を待ちます。

loop_back_read 状態:

flagd_d = 1 になると、ステートマシンはこの状態に入ります。ここで、ステートマシンは以下のように読み出し制御信号をアサートします。

PKTEND# = 1 ; SLOE# = 0 ; SLRD# = 0 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 3

loop_back_read_rd_oe_delay 状態:

flagc_d = 0 になると、ステートマシンはこの状態に入ります。スレーブ FIFO 制御ラインの状態は下記になります。

PKTEND# = 1 ; SLOE# = 0 ; SLRD# = 0 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 3

部分的フラグの一般式節の式 (2b) によると、部分的フラグ (flagd) が 0 になった後、FX3 はアサートされている SLRD#を 3 サイクルの間サンプリングする必要があります。FPGA からインターフェースまでの伝播遅延が 1 サイクルであることを前提にすると、FPGA は flagd_d (flagd のフリップフロップされた出力) を 0 としてサンプリングした後に 1 サイクルの間 SLRD#をアサートします。

loop_back_read_oe_delay 状態:

rd_oe_delay_cnt = 0 になると、ステートマシンはこの状態に入ります。スレーブ FIFO 制御ラインの状態は下記になります。

PKTEND# = 1 ; SLOE# = 0 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 3

loop_back_wait_flaga 状態:

oe_delay_cnt = 0 になると、ステートマシンは loop_back_
wait_flaga 状態に入ります。スレーブ FIFO 制御ラインの状態は下記になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

loop_back_wait_flagb の状態:

flaga_d = 1 になると、ステートマシンは loop_back_wait_
flaga 状態に入ります。スレーブ FIFO 制御ラインの状態は下記になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

loop_back_write 状態:

flagb_d = 1 になると、ステートマシンは loop_back_wait_
flaga 状態に入ります。スレーブ FIFO 制御ラインの状態は下記になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 0 ; A[1:0] = 0

loop_back_write_wr_delay 状態:

flagb_d = 0 になると、ステートマシンは loop_back_wait_
flaga 状態に入ります。スレーブ FIFO 制御ラインの状態は下記になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

「部分的フラグの一般式」節の式 (1) によると、部分的フラグ (flagb) が 0 になった後、FX3 はアサートされている SLWR#を 2 サイクルの間サンプリングする必要があります。FPGA からインターフェースまでの伝播遅延が 1 サイクルであることを前提にすると、FPGA は部分的 flagb_d (flagb のフリップフロップされた出力) を 0 としてサンプリングした後に 1 サイクルの間 SLWR#をアサートします。

loop_back_flush_fifo 状態:

1 クロック サイクル後、ステートマシンはこの状態に入り、内部 FIFO をフラッシュします。スレーブ FIFO 制御ラインの状態は下記になります。

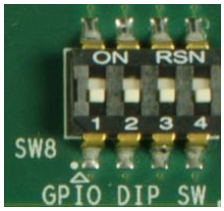
PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

11.5. プロジェクトの動作

11.5.1. ループバック転送のテスト手順

1. SuperSpeed Explorer Kit を使用する場合、ジャンパ J5 が開いていることを確かめてください。FX3 開発キット (CYUSB3KIT-001) を使用する場合、表 7 に基づいてジャンパとスイッチをセットアップしてください。FMC で FX3 DVK 基板を Xilinx SP601 DVK 基板と接続し、FX3 DVK 基板に電源投入してから Xilinx SP601 DVK に電源投入します。
2. FPGA がトランザクションを開始する前に、FPGA および FX3 を設定する必要があります。FPGA コードは、GPIO 入力を使用してどのモードを開始するかを決めます。

図 28. Xilinx SP601 基板上の SW8 – FPGA および FX3 設定の前にすべてオフ



3. FX3 デバイスにファームウェア イメージ ファイル *SF_loopback.img* をロードします。FX3 SDK 同梱の Control Center ユーティリティを使って USB から FX3 をプログラムできます。
4. *slavefifo2b_fpga_top.bit* ファイルを Xilinx Spartan 6 FPGA にロードします。FPGA は、Xilinx ISE 設計スイト 14.7 同梱の iMPACT アプリケーションなどどの標準プログラマでもプログラムできます。ザイリンクス FPGA ボードをプログラミングする前に、FX3 DVK をプログラムする必要があります。 FPGA がすでにプログラムされている場合、PMODE ラインを駆動するため、FX3 DVK はホスト PC でエニュメレートされません。

図 29. Control Center ユーティリティを使って FX3 ファームウェアをプログラム

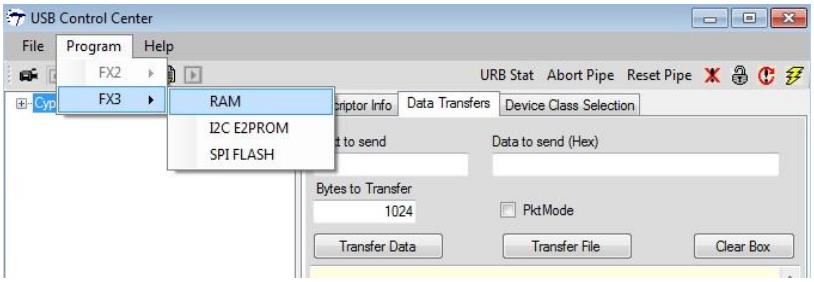
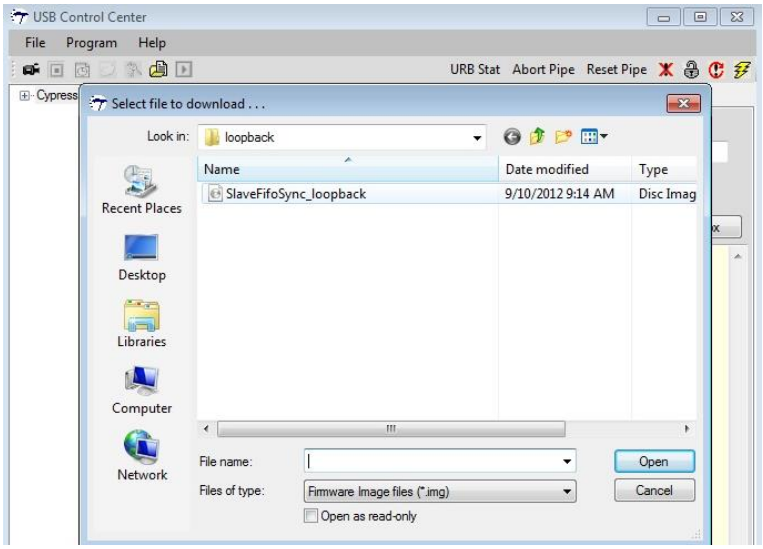


図 30. Control Center ユーティリティを使ってループバック テスト用の FX3 ファームウェアをプログラム



ファームウェアをダウンロードしてから、FX3 デバイスは (USB 3.0 ポートに接続された場合) データ転送を開始する前に、SuperSpeed デバイスとしてエニュメレートします。FPGA は GPIO 入力を使ってどの転送を開始するかを決めます。この目的には、SP601 DVK 基板上のスイッチ SW8 が使用されます。異なる転送用のスイッチ設定は表 6 に示します。

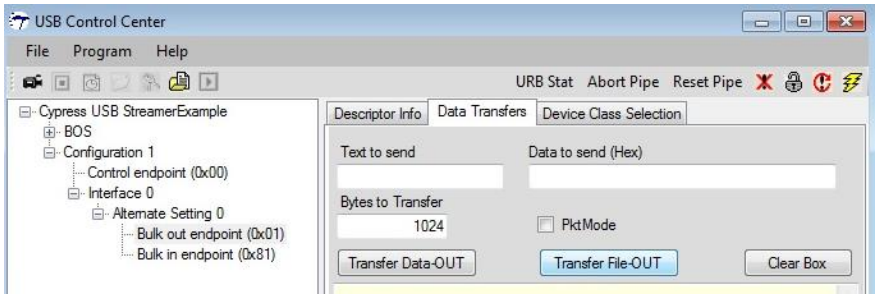
5. Xilinx SP601 基板上の SW8 の 1 と 3 番位置は、ループバック転送を実行するためにオンにする必要があります。

表 6.*slavefifo2b_fpga_top.bit* での FPGA 転送モードの設定

SW8[4]	SW8[3]	SW8[2]	SW8[1]	FPGA 転送モード
オフ	オフ	オフ	オン	FPGA はフルパケットに続いてショートパケットを連続的に書き込む
オフ	オフ	オン	オフ	FPGA はフルパケットに続いて ZLP を連続的に書き込む
オフ	オフ	オン	オン	FPGA はフルパケットを連続的に書き込む (ホストから Bulk IN パケットをストリーム)
オフ	オン	オフ	オフ	FPGA はフルパケットを連続的に読み出す (ホストから Bulk OUT パレットをストリーム)
オフ	オン	オフ	オン	ループバック転送モード
オフ	X	X	X	無効

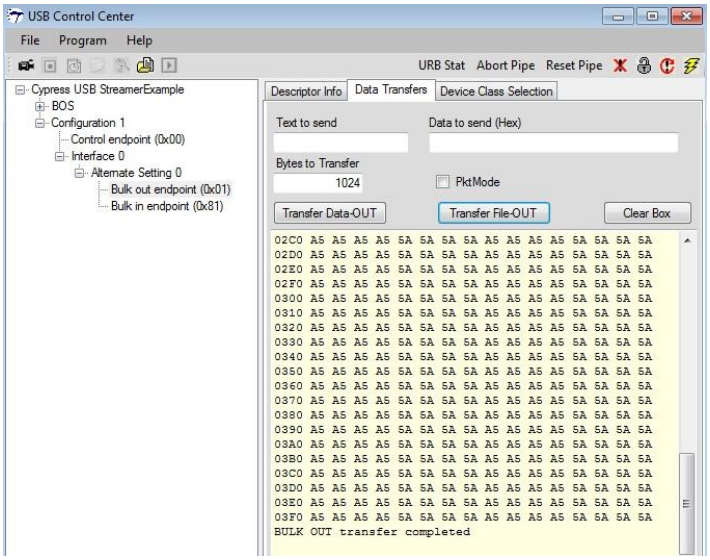
6. この時、転送は Control Center ユーティリティから起動できます。まず、Bulk OUT 転送を USB ホストから起動します。Control Center 内の Bulk OUT エンドポイントを選択して **Transfer File-OUT** ボタンをクリックします。

図 31. Transfer File-OUT でバルク OUT 転送を起動



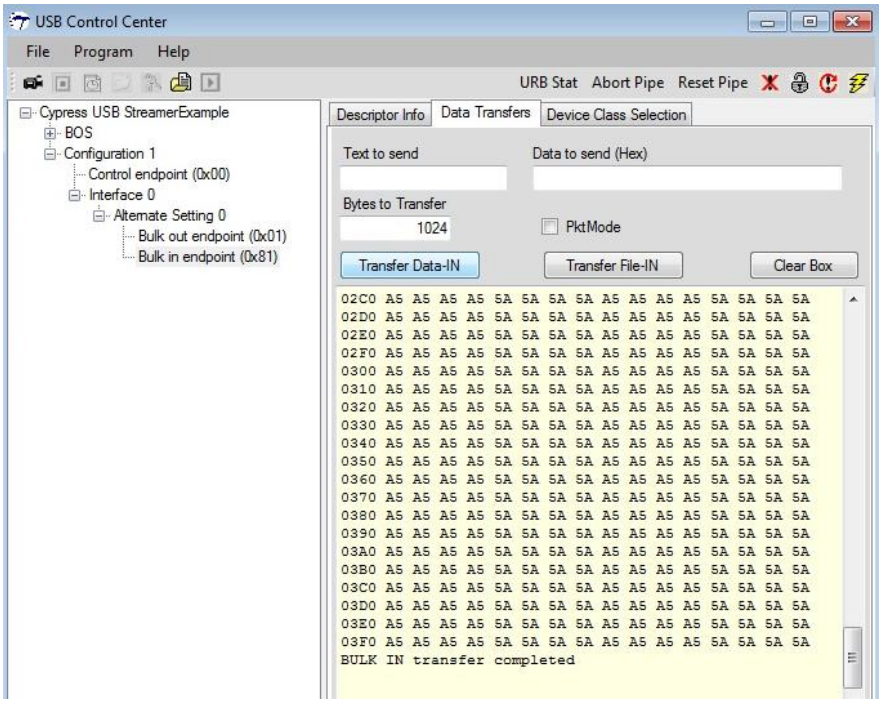
7. これにより、転送するデータを保存するファイルを閲覧し選択できます。本アプリケーション ノートの添付内の Loopback フォルダには *TEST.txt* ファイルがあります。このファイルには、交互方式で「0xA5A5A5A5 0x5A5A5A5A」を送信するデータ パターンが含まれます。ダブル クリックしてファイルを選択しデータを送信します。

図 32. Transfer File-OUT 用に TEST.txt ファイルを選択することでデータ パターンを転送



8. FPGA は、FLAGA が 1 になることを待つ状態に入りました。FPGA はデータが PIB_SOCKET_0 のバッファに格納されるとすぐに、そのデータを読み出します。その後、FPGA は同じデータをループバックして FX3 の PIB_SOCKET_3 に書き込みます。
9. USB ホストから Bulk IN 転送を発行できます。Control Center 内の Bulk IN エンドポイントを選択して **Transfer File-IN** ボタンをクリックします。この時、以前に書き込まれたデータは読み戻されます。

図 33. Transfer Data-IN を使って Bulk IN 転送を起動することでループバック テストを実行



11.5.2. ストリーム転送のテスト手順

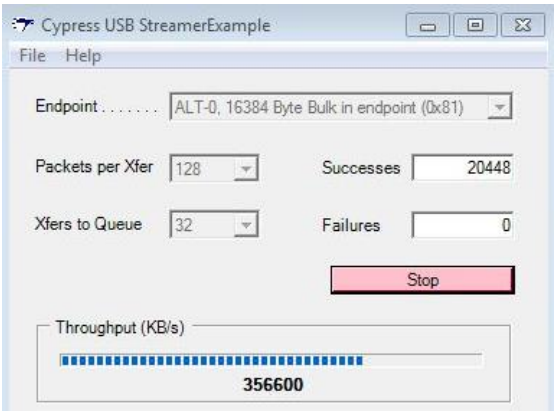
注: 以下のパスにある FX3 SDK 同梱の C++ Streamer ユーティリティを使用してください。

<FX3_SDK_installation_path>*EZ-USB FX3 SDK\1.3\application\cpp\streamer\x86*

このパスのリリース 1.3 は FX3 SDK バージョン番号です。FX3 SDK の将来のリリースではバージョンアップできます。

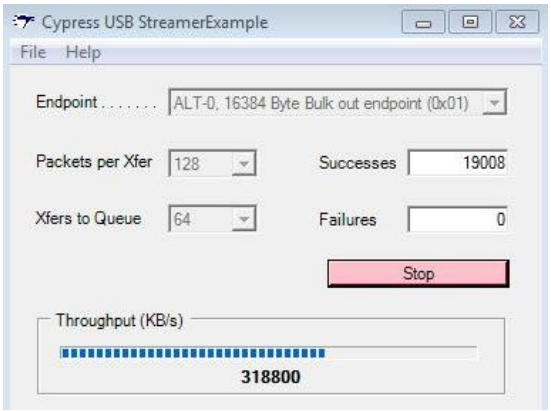
1. ストリーム転送を実行するには、前述した同じ.bit ファイルを FPGA にロードができます。表 6 に示すように、ユーザーは切り替えの設定をのみ行う必要があります。
2. ストリーム IN／OUT 転送を実行する場合、FX3 デバイスにファームウェア イメージ ファイル *SF_streamIN.img* をロードします。FX3 SDK に同梱された Control Center を使って USB から FX3 をプログラムできます。
3. ファームウェアをダウンロードした後、FX3 デバイスは (USB 3.0 ポートに接続すれば) SuperSpeed デバイスとしてエニュメレーションします。この時、データ転送は FX3 SDK に同梱された Control Center と Streamer ユーティリティから開始できます。
4. *slavefifo2b_fpga_top.bit* ファイルを Xilinx Spartan 6 FPGA にロードします。
5. 表 6 に示すように、必要な転送に応じて SP601 基板に搭載されたスイッチ SW8 をセットします。
6. ストリーム IN の場合、FPGA は FLAGA が 1 になることを待つ状態に入りました。FPGA はバッファが使用可能になるとすぐに、FX3 の PIB_SOCKET_0 に連続的に書き込みます。USB ホストから連続 Bulk IN 転送を発行できます。Cypress Streamer ユーティリティ内の Bulk IN エンドポイントを選択して **Start** ボタンをクリックします。転送速度が表示されます。図 34 に示される転送速度は、Intel Z77 Express チップセットを搭載した Win7 64 ビット PC で観察されるものです。

図 34. Cypress Streamer ユーティリティで表示されるストリーム IN の転送速度



7. ストリーム OUT の場合、FPGA は FLAGC が 1 になることを待つ状態に入りました。FPGA はデータが使用可能になるとすぐに、FX3 の PIB_SOCKET_3 から連続的に読み出します。USB ホストから連続 Bulk OUT 転送を発行できます。Cypress Streamer ユーティリティ内の Bulk OUT エンドポイントを選択して **Start** ボタンをクリックします。転送速度が表示されます。図 35 に示される転送速度は、Intel Z77 Express チップセットを搭載した Win7 64 ビット PC で観察されるものです。

図 35. Cypress Streamer ユーティリティで表示されるストリーム OUT の転送速度



SF_streamIN.img はストリーム IN とストリーム OUT 両方に使用できます。*SF_streamIN.img* では、8 個のバッファが P2U DMA チャネルに割り当てられ、もう 4 個のバッファが U2P チャネルに割り当てられます。*SF_streamOUT.img* では、8 個のバッファが U2P DMA チャネルに割り当てられ、もう 4 個のバッファが P2U チャネルに割り当てられます。そのため、*SF_streamIN.img* ファームウェア ファイルで P2U チャネルのより高い性能を実現でき、*SF_streamOUT.img* ファームウェア ファイルで U2P チャネルのより高い性能を実現できます。

11.5.3. ショートパケットと ZLP 転送のテスト手順

1. ストリーム転送を実行するには、前述した同じ.bit ファイルを FPGA にロードができます。表 6 に示すように、ユーザーは切り替えの設定をのみ行う必要があります。
2. ファームウェア イメージ ファイル *SF_shrt_ZLP.img* を FX3 デバイスにロードします。FX3 SDK に同梱された Control Center を使って USB から FX3 をプログラムできます。
3. *slavefifo2b_fpga_top.bit* ファイルを Xilinx Spartan 6 FPGA にロードします。
4. FX3 ファームウェアがプログラムされるとすぐに、PIB_SOCKET_0 に割り当てられるバッファは使用可能になります。FLAGA 信号を監視することで、FPGA はこの条件を待機する状態に入りました。FPGA はフラグが 1 になるとすぐに、FX3 に書き込みはじめます。
5. スイッチがショートパケット転送用に設定された場合、FPGA はフルパケット (1024 バイト) に続いてショートパケットを書き込みます。スイッチが ZLP 転送用に設定された場合、FPGA はフルパケット (1024 バイト) に続いて ZLP を書き込みます*。

6. ここで、USB ホストは Bulk IN トークンを発行できます。Control Center ユーティリティ内の Bulk IN エンドポイントを選択して **Transfer Data-IN** ボタンをクリックします。まず、フルパケットが受信されます。**Transfer Data-IN** をもう 1 度クリックします。この時、ショートパケットあるいは ZLP が受信されます。

***注:** ショートパケット転送の直後に FX3 をリセットせずに ZLP 転送が完了された場合、最初の転送はショートパケット転送です。これは、ZLP 転送の直後のショートパケット転送に当てはまります。理由は、Control Center でホストがデータを要求するまでは FX3 バッファはフラッシュされなかったためです。

図 36.連続した Tranfer Data-IN 動作でフルパケットに続いてショートパケットが受信される

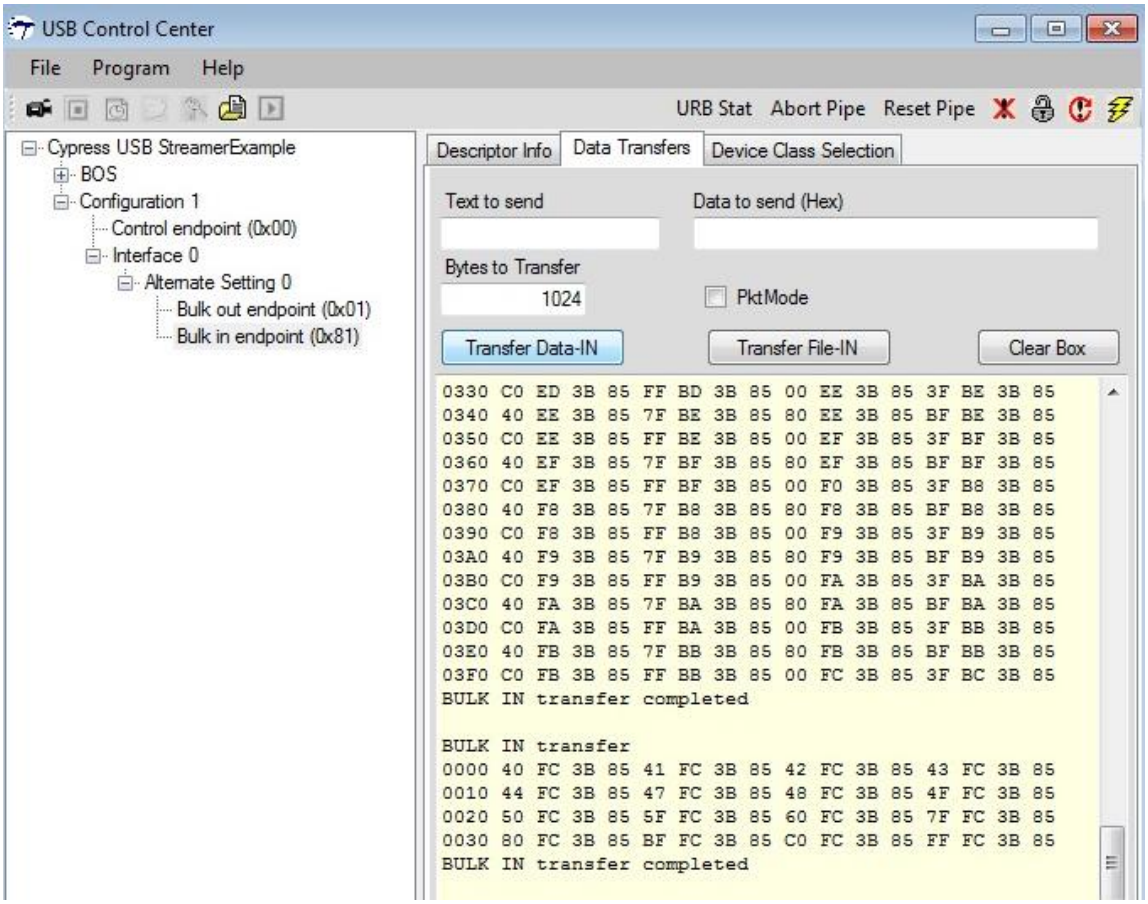
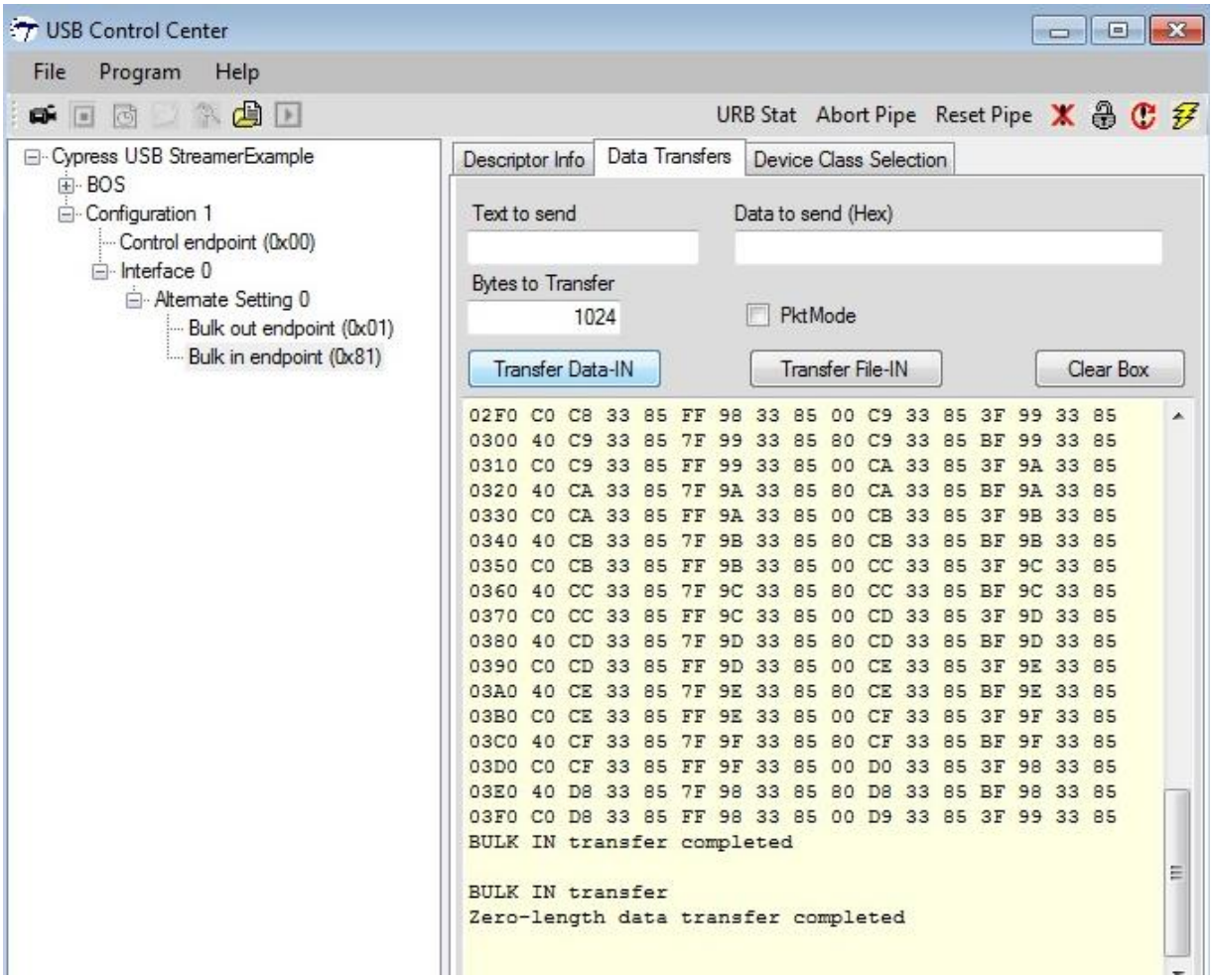


図 37. 連続した Transfer Data-IN 動作でフルパケットに続いて長さゼロのパケット (ZLP) が受信される



7. FPGA が連続的にデータを書き込んでいるため、複数の BULK IN 転送は行われます。

12. 設計例 2: Altera FPGA を FX3 の同期スレーブ FIFO インターフェースと連結

ここでは、Altera Cyclone 3 FPGA が同期スレーブ FIFO インターフェースを介して FX3 に接続される完全な設計例を提供します。以下で本設計を実装するために使うハードウェア、ファームウェアおよびソフトウェアのコンポーネントについて説明します。

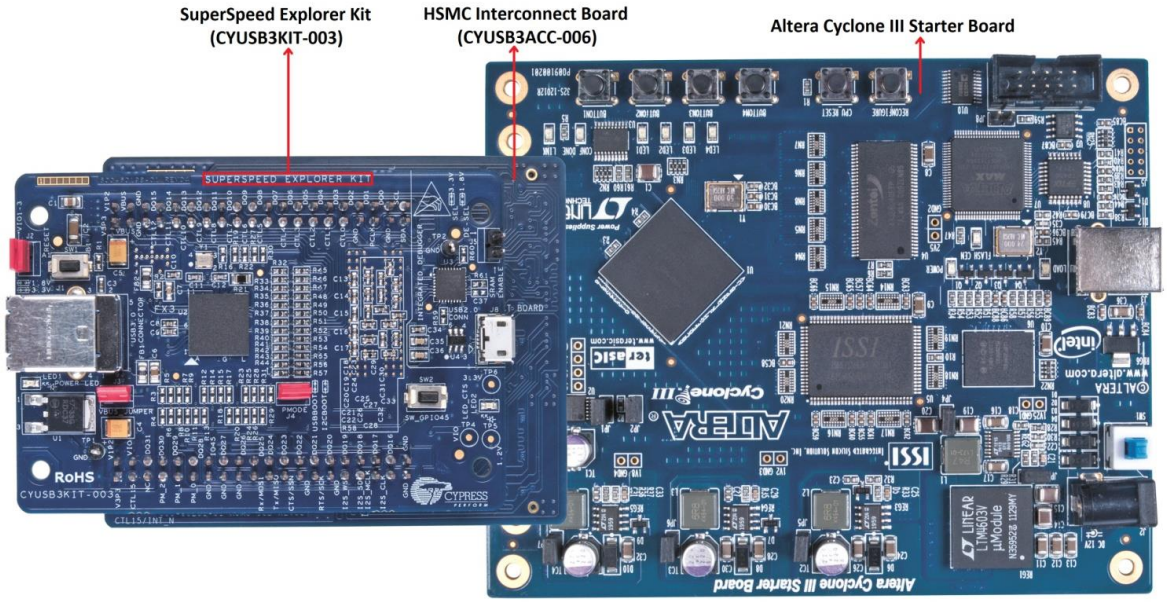
12.1. ハードウェアのセットアップ

本例で提供されたプロジェクトは、Cyclone III FPGA Starter 基板と相互接続しているサイプレス FX3 開発キット (CYUSB3KIT-001) または SuperSpeed Explorer Kit (CYUSB3KIT-003) から構成されるハードウェアのセットアップで実行されます。FX3 基板と Altera 基板は、Samtec—HSMC 相互接続基板を使用して接続されています。CYUSB3ACC-003 相互接続基板は、サイプレス FX3 開発キット (CYUSB3KIT-001) 上の Samtec コネクタと Altera 基板上の HSMC コネクタを接続します。

CYUSB3ACC-006 相互接続基板は、SuperSpeed Explorer Kit (CYUSB3KIT-003) 上のヘッダと Xilinx 基板上の HSMC コネクタを接続します。

図 38 に、SuperSpeed Explorer Kit を使用したハードウェア セットアップを示します。付録 B: FX3 DVK (CYUSB3KIT-001) を用いたハードウェア セットアップを参照してください。ハードウェア セットアップ以外に、以下のステップはご使用の FX3 基板にかかわらず同じです。

図 38. HSMC 相互接続基板を用いて Altera Cyclone III Starter 基板に接続したサイプレス SuperSpeed Explorer Kit



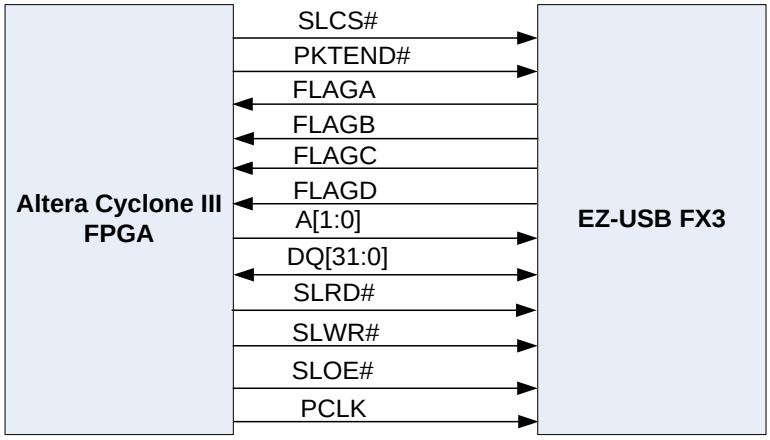
注: FPGA がスレーブ FIFO インターフェースを介して FX3 とインターフェースするすべてのアプリケーションで、SuperSpeed Explorer Kit 上のジャンパ J5 が開いていることを確かめてください。

12.2. ファームウェアとソフトウェア コンポーネント

- FX3 SDK 同梱の FX3 同期スレーブ FIFO ファームウェア プロジェクト
- FX3 SDK 同梱の Control Center と Streamer ソフトウェア ユーティリティ

下図に、FPGA と FX3 の相互接続図を示します。

図 39. FPGA と FX3 の相互接続図



本例には以下のコンポーネントがあります。

- ループバック転送: このコンポーネントでは、FPGA は FX3 から完全なバッファを読み出してから、その内容を FX3 に書き戻します。USB ホストはこのデータを送受信するために、OUT／IN トークンを発行する必要があります。EZ-USB FX3 SDK 同梱の Control Center ユーティリティはこの目的に使用できます。
- ショートパケット: このコンポーネントでは、FPGA はフルパケットに続いてショートパケットを FX3 へ転送します。USB ホストはこのデータを受信するために、IN トークンを発行する必要があります。
- 長さゼロのパケット (ZLP) 転送: このコンポーネントでは、FPGA はフルパケットに続いて長さゼロのパケットを FX3 へ転送します。USB ホストはこのデータを受信するために、IN トークンを発行する必要があります。
- ストリーム (IN) データ転送: このコンポーネントでは、FPGA は単方向転送を行う、すなわち、同期スレーブ FIFO を介して FX3 に連続的にデータを書き込みます。USB ホストはこのデータを受信するために、IN トークンを発行する必要があります。EZ-USB FX3 SDK 同梱の Control Center または Streamer ユーティリティはこの目的に使用できます。
- ストリーム (OUT) データ転送: このコンポーネントでは、FPGA は単方向転送を行う、すなわち、同期スレーブ FIFO を介して FX3 から連続的にデータを読み出します。USB ホストはこのデータを提供するために、OUT トークンを発行する必要があります。EZ-USB FX3 SDK 同梱の Control Center または Streamer ユーティリティはこの目的に使用できます。

12.3. FX3 ファームウェアの詳細

FX3 ファームウェアは、FX3 SDK 同梱のサンプル プロジェクトに基づきます。ファームウェアの主な特長は以下のとおりです。

- USB 3.0 と USB 2.0 の両方に対応しています。
- サイプレスの VID/PID (0x04B4/0x00F1) でエニュメレーションされます。これにより、USB 転送を開始するためにサイプレスの Control Center と Streamer ユーティリティの使用が可能です。
- 次の特長を持つ同期スレーブ FIFO ディスクリプタを統合します。
  - 最大 4 個のソケットまでサポート
  - 最大 32 ビットまでのデータ バスを設定可能
  - 100MHz の PCLK 入力クロックで動作
  - 以下の 4 つのフラグを設定:
    - i. FLAGA: スレッド 0 専用の一杯フラグ
    - ii. FLAGB: スレッド 0 専用の、ウォーターマーク値 6 を持つ部分的フラグ
    - iii. FLAGC: スレッド 3 専用の空フラグ
    - iv. FLAGD: スレッド 3 専用の、ウォーターマーク値 6 を持つ部分的フラグ

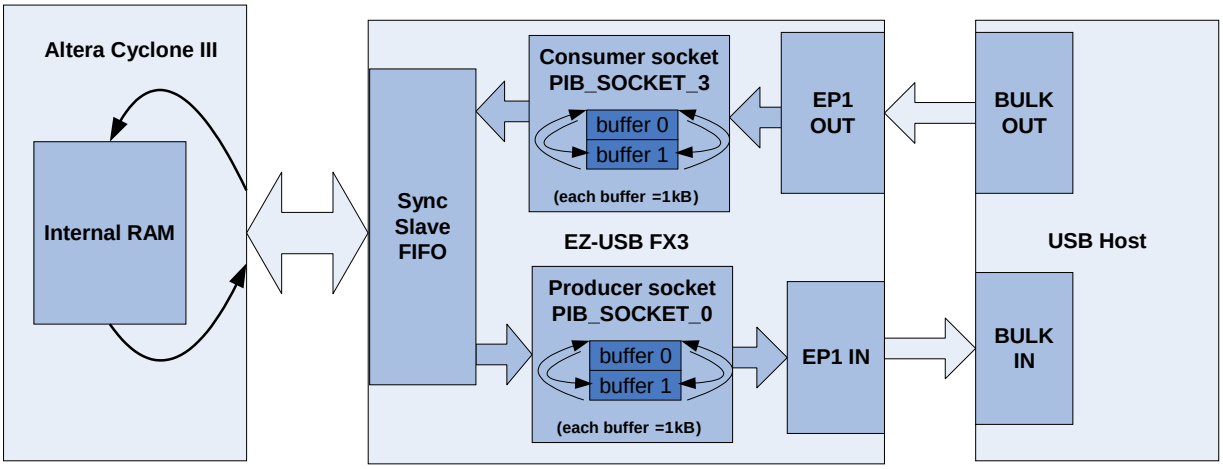
注: 本アプリケーション ノートで提供された GPIF II Designer プロジェクトはこれら設定をデモします。さらにファームウェア プロジェクトは、ウォーターマーク値の設定に CyU3PGpifSocketConfigure() API を使用する方法を示します。

- PLL 周波数を 400MHz に設定します。これは、setSysClk400 パラメーターを CyU3PDeviceInit() 関数の入力にセットすることで行われます。

注: この設定は、スレーブ FIFO が 32 ビット データを 100MHz で動作するのに必須です。

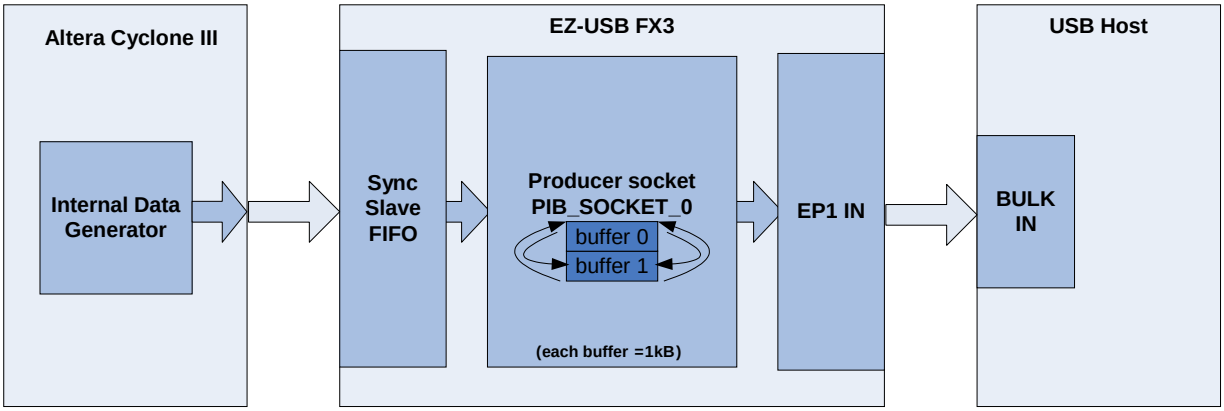
- DMA チャネルをセットアップします。
  - ループバック転送、ショートパケットおよび ZLP 転送の場合、次の 2 本の DMA チャネルが作成されます。
    - i. PIB_SOCKET_0 がプロデューサであり、UIB_SOCKET_1 がコンシューマである P2U チャネル。USB 接続が USB 2.0 接続か USB 3.0 接続かに応じて、DMA バッファ サイズは 512 か 1024 です。DMA バッファ数は 2 個です。
    - ii. PIB_SOCKET_3 がコンシューマであり、UIB_SOCKET_1 がプロデューサである U2P チャネル。USB 接続が USB 2.0 接続か USB 3.0 接続かに応じて、DMA バッファ サイズは 512 か 1024 です。DMA バッファ数は 2 個です。

図 40. ループバック転送のセットアップ



注: ショートパケットと ZLP には、P2U チャネルのみをご使用ください。

図 41. ショートパケットと ZLP 転送のセットアップ

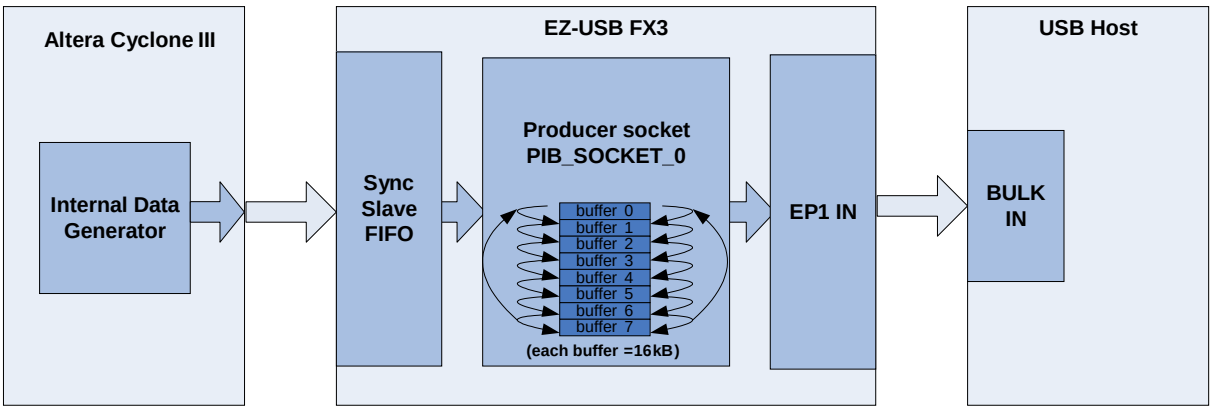


本アプリケーション ノートで提供した FX3 ファームウェア プロジェクトの *cyfxslfifosync.h* ファイルで以下の#define が有効の場合、本節で説明した DMA チャネルはセットアップされます。

```
/* set up DMA channel for loopback/short packet/ZLP transfers */
#define LOOPBACK_SHRT_ZLP
```

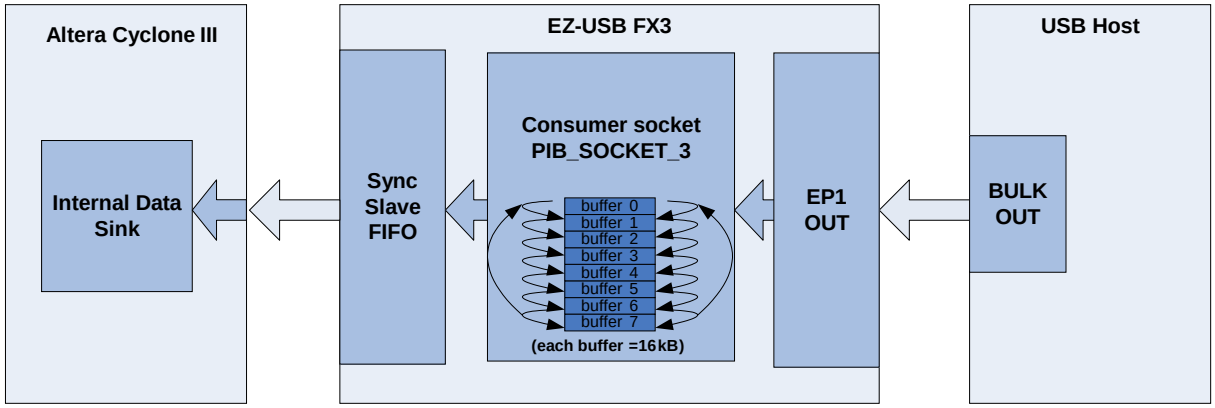
- ストリーム転送の場合、次の 2 本の DMA チャネルが作成されます。
  - PIB_SOCKET_0 がプロデューサであり、UIB_SOCKET_1 がコンシューマである P2U チャネル。DMA バッファ サイズは 16×1024 (USB 3.0 接続の場合) または 16×512 (USB 2.0 接続の場合) です。DMA バッファ数は 8 個です。このバッファのサイズと数は、高スループット性能を達成することを目的として選択されます。

図 42. ストリーム IN 転送用のセットアップ – 性能に最適化されたバッファの数とサイズ



  - PIB_SOCKET_3 がコンシューマであり、UIB_SOCKET_1 がプロデューサである U2P チャネル。DMA バッファ サイズは 16×1024 (USB 3.0 接続の場合) または 16×512 (USB 2.0 接続の場合) です。DMA バッファ数は 4 個です。バッファ数は性能を向上するために増加できますが、P2U チャネルのバッファ数を減らす必要があるため、ご注意ください。これは、FX3 SDK は両方のチャネルが 8 個の 16×1024 バッファを持つようの十分なバッファ メモリを提供していないからです。

図 43. ストリーム OUT 転送用のセットアップ – 性能に最適化されたバッファの数とサイズ



本アプリケーション ノートで提供した FX3 ファームウェア プロジェクトの *cyfxslfifosync.h* ファイルで以下の#define が有効の場合、前述した DMA チャネルはセットアップされます。

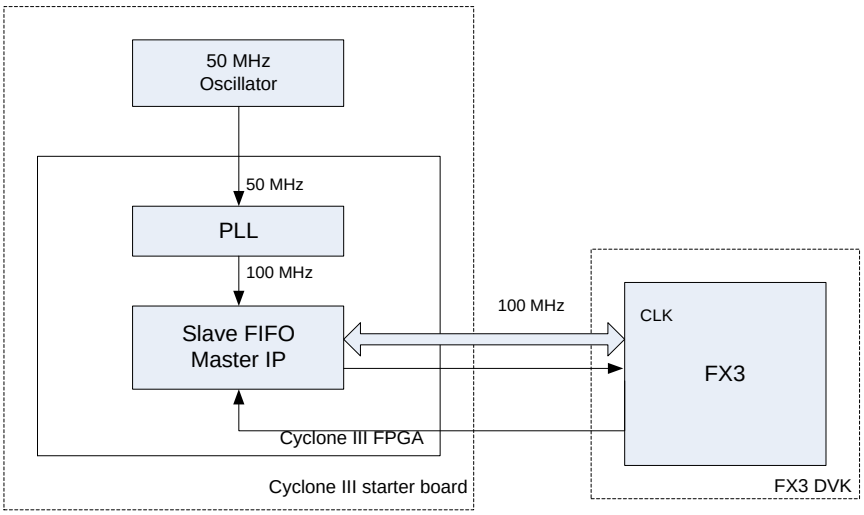
```
/* set up DMA channel for stream IN/OUT transfers */
#define STREAM_IN_OUT
```

P2U と U2P DMA チャネルに割り当てられたバッファ数は、以下の#define を使用して制御できます (これらの定義は *cyfxslfifosync.h* ファイルにもあります)。

```
/* Slave FIFO P_2_U channel buffer count */
#define CY_FX_SLFIFO_DMA_BUF_COUNT_P_2_U      (4)
/* Slave FIFO U_2_P channel buffer count */
#define CY_FX_SLFIFO_DMA_BUF_COUNT_U_2_P      (8)
```

12.4. FPGA 実装の詳細

図 44. SP601 評価キットを使った Altera Cyclone III (EP3C25F324C6) FPGA 実装



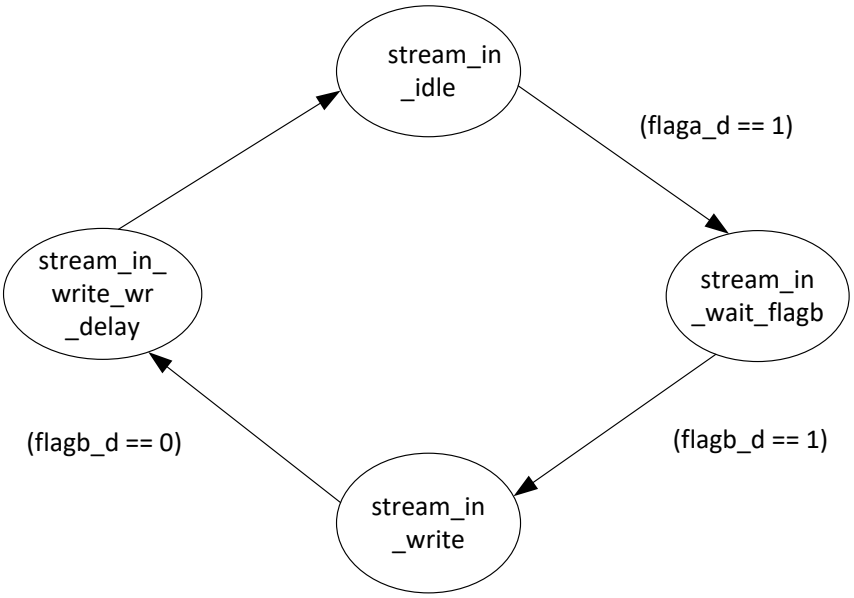
FX3 の最高性能をデモするために、GPIF インターフェースを 100MHz で動作させます。Cyclone III Starter 基板は 50MHz シングルエンドの発振器を搭載しています。FPGA は、PLL を使用して 50MHz クロックから 100MHz のクロックを生成します。

各種の転送の状態は以下のとおりです。

12.4.1. ストリーム IN の例: FPGA がスレーブ FIFO に書き込む

Verilog RTL で実装されたストリーム IN 転送用のステートマシンは下図に示します。

図 45. ストリーム IN 用の FPGA ステートマシン



stream_in_idle 状態:

この状態では、ステートマシンで使用するすべてのレジスタと信号を初期化します。スレーブ FIFO 制御ラインの状態は以下になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

stream_in_wait_flagb 状態:

flaga_d = 1 になると、ステートマシンはこの状態に入り、flagb_d を待ちます。

stream_in_write 状態:

flagb_d = 1 になると、ステートマシンはこの状態に入り、スレーブ FIFO インターフェースへ書き込み始めます。スレーブ FIFO 制御ラインの状態は以下になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 0 ; A[1:0] = 0

12.4.2. stream_in_write_wr_delay 状態:

flagb_d = 0 になると、ステートマシンはこの状態に入ります。スレーブ FIFO 制御ラインの状態は以下になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

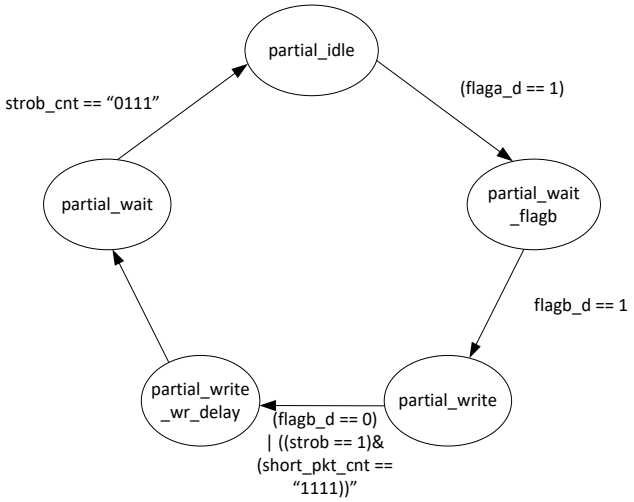
1 クロック サイクル後、ステートマシンは stream_in_idle 状態に入ります。

「部分的フラグの一般式」節の式 (1) によると、部分的フラグ (flagb) が 0 になった後、FX3 はアサートされている SLWR#を 2 サイクルの間サンプリングする必要があります。FPGA からインターフェースまでの伝播遅延が 1 サイクルであることを前提にすると、FPGA は flagb_d (flagb のフリップフロップされた出力) を 0 としてサンプリングした後に 1 サイクルの間 SLWR#をアサートします。

12.4.3. ショートパケットの例: FPGA がフルパケットに続いてショートパケットをスレーブ FIFO に書き込む

この例では、PKTEND#を使用したショートパケットの転送手順をデモします。Verilog RTL で実装された、ショートパケット例のステートマシンは下図に示します。

図 46. ショートパケット転送用のステートマシン



partial_idle 状態:

この状態では、ステートマシンで使用するすべてのレジスタと信号を初期化します。スレーブ FIFO 制御ラインの状態は以下になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

partial_wait_flagb 状態:

flaga_d = 1 になると、ステートマシンはこの状態に入ります。

partial_write 状態:

flagb_d = 1 になると、ステートマシンはこの状態に入ります。スレーブ FIFO 制御ラインの状態は以下になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 0 ; A[1:0] = 0

partial_write_wr_delay 状態:

flagb_d = 0、または strob = 1 かつ short_pkt_cnt = 「1111」になると、ステートマシンはこの状態に入ります。strob = 1 の場合、FPGA マスターはショートパケットを転送します。スレーブ FIFO 制御ラインの状態は以下になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

1 クロック サイクル後、ステートマシンは partial_wait 状態に入ります。

「部分的」節の式 (1) によると、部分的フラグ (flagb) が 0 になった後、FX3 はアサートされている SLWR#を 2 サイクルの間サンプリングする必要があります。FPGA からインターフェースまでの伝播遅延が 1 サイクルであることを前提にすると、FPGA は部分的 flagb_d (flagb のフリップフロップされた出力) を 0 としてサンプリングした後に 1 サイクルの間 SLWR#をアサートします。

partial_wait 状態:

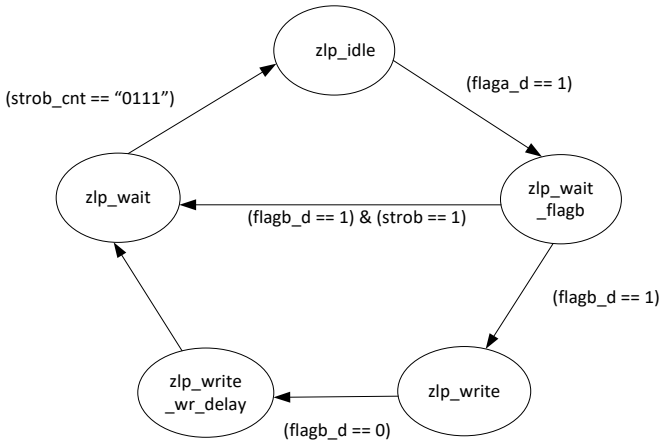
strob_cnt = 0111 になると、ステートマシンは partial_idle 状態に入ります。

ウォーターマーク値が 6 であるため、flaga は部分的フラグ (flagb) から 6 クロック サイクル後にのみ 0 になることを期待されます。この状態は flaga ステータスが確かに有効であることを保証するために、少なくとも 4 クロック サイクル以上実行されます。

12.4.4. 長さゼロのパケットの例: FPGA がフルパケットに続いて ZLP をスレーブ FIFO に書き込む

この例では、PKTEND#を使用した ZLP の転送手順をデモします。Verilog RTL で実装された ZLP 例用のステートマシンは下図に示します。

図 47. ZLP 転送用のステートマシン



zlp_idle 状態:

この状態では、ステートマシンで使用するすべてのレジスタと信号を初期化します。スレーブ FIFO 制御ラインの状態は以下になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

zlp_wait_flagb 状態:

flaga_d = 1 になると、ステートマシンはこの状態に入ります。

zlp_write 状態:

flagb_d = 1 になると、ステートマシンはこの状態に入ります。スレーブ FIFO 制御ラインの状態は以下になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 0 ; A[1:0] = 0

zlp_write_wr_delay 状態:

flagb_d = 0 になると、ステートマシンはこの状態に入ります。スレーブ FIFO 制御ラインの状態は以下になります。
PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

1 クロック サイクル後、ステートマシンは zlp_wait 状態に入ります。

「部分的フラグの一般式」節の式 (1) によると、部分的フラグ (flagb) が 0 になった後、FX3 はアサートされている SLWR#を 2 サイクルの間サンプリングする必要があります。FPGA からインターフェースまでの伝播遅延が 1 サイクルであることを前提にすると、FPGA は部分的 flagb_d (flagb のフリップフロップされた出力) を 0 としてサンプリングした後に 1 サイクルの間 SLWR#をアサートします。

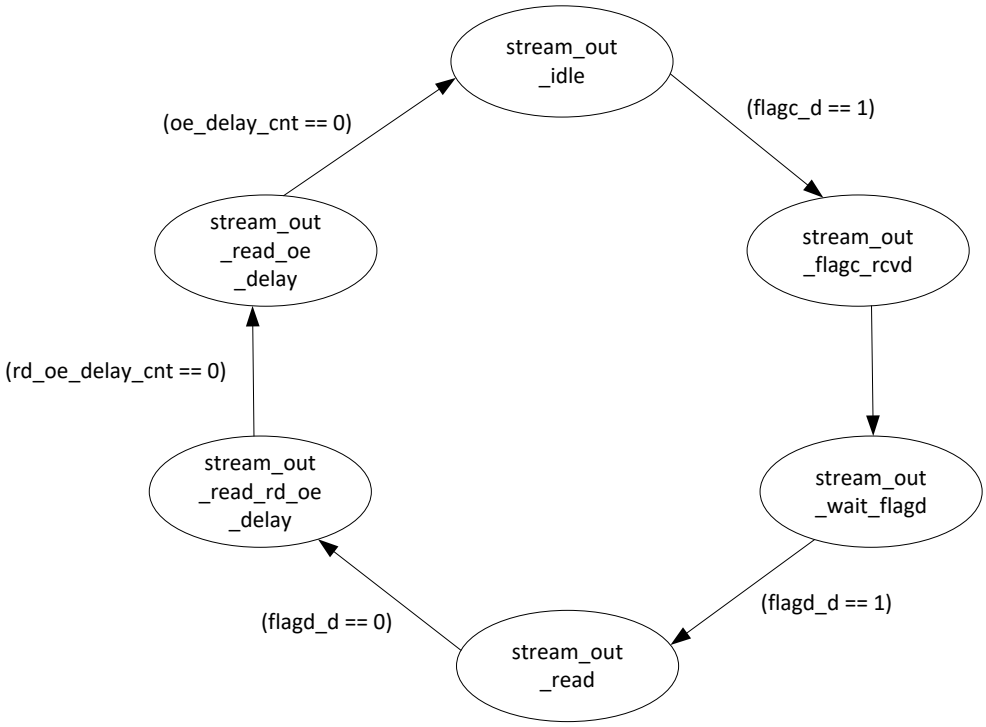
zlp_wait 状態:

flagb_d = 1 で、strob = 1 になると、ステートマシンは zlp_wait_flagb 状態からこの状態に入り、zlp パケットを slavefifo に転送します。 ウォーターマーク値が 6 であるため、flaga は部分的フラグ (flagb) から 6 クロック サイクル後にのみ 0 になることを期待されます。この状態は flaga ステータスが確かに有効であることを保証するために、少なくとも 4 クロック サイクル以上実行されます。

strob_cnt = 0111 になると、ステートマシンは zlp_idle 状態に入ります。

12.4.5. Verilog RTL で実装されるストリーム OUT 例用のステートマシン

図 48. ストリーム OUT 転送用のステートマシン



stream_out_idle 状態:

この状態では、ステートマシンで使用するすべてのレジスタと信号を初期化します。スレーブ FIFO 制御ラインの状態は以下になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 3

stream_out_flagc_rcvd 状態:

flagc_d = 1 になると、ステートマシンはこの状態に入ります。

stream_out_wait_flagd 状態:

1 クロック サイクル後、ステートマシンはこの状態に入ります。

stream_out_read 状態:

flagc_d = 1 になると、ステートマシンはこの状態に入ります。ここで、ステートマシンは以下のように読み出し制御信号をアサートします。

PKTEND# = 1 ; SLOE# = 0 ; SLRD# = 0 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 3

stream_out_read_rd_oe_delay 状態:

flagc_d = 0 になると、ステートマシンはこの状態に入ります。スレーブ FIFO 制御ラインの状態は以下になります。

PKTEND# = 1 ; SLOE# = 0 ; SLRD# = 0 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 3

「部分的フラグの一般式」節の式 (2b) によると、部分的フラグ (flagd) が 0 になった後、FX3 はアサートされている SLRD#を 3 サイクルの間サンプリングする必要があります。FPGA からインターフェースまでの伝播遅延が 1 サイクルであることを前提にすると、FPGA は flagd_d (flagd のフリップフロップされた出力) を 0 としてサンプリングした後に 1 サイクルの間 SLRD#をアサートします。

stream_out_read_oe_delay 状態:

rd_oe_delay_cnt = 0 になると、ステートマシンはこの状態に入ります。スレーブ FIFO 制御ラインの状態は以下になります。

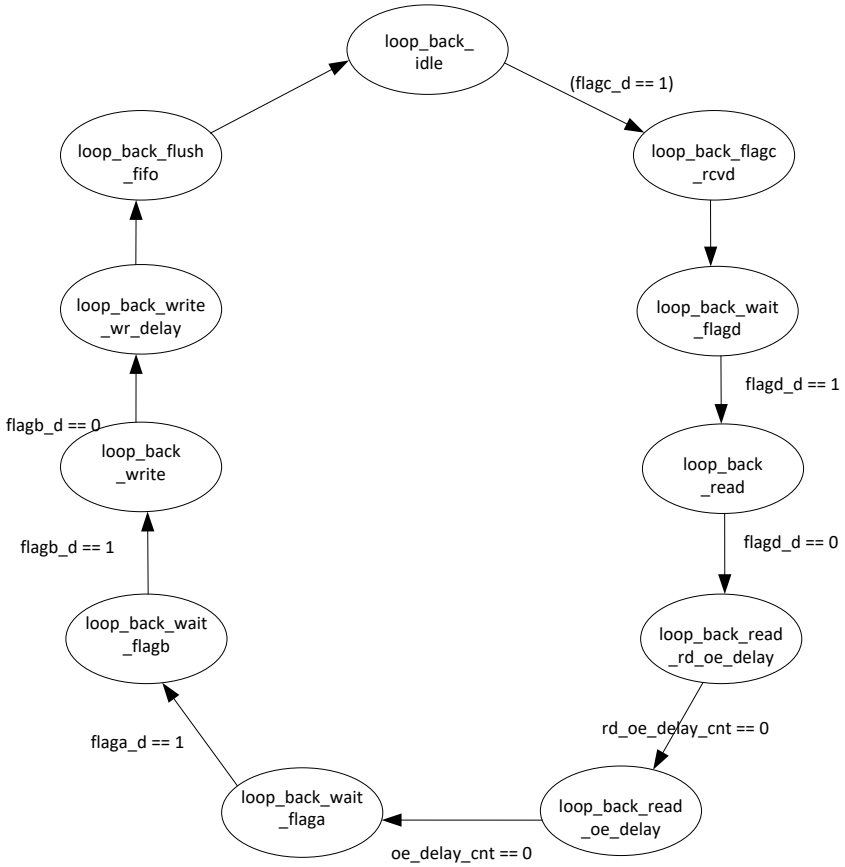
PKTEND# = 1 ; SLOE# = 0 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 3

oe_delay_cnt = 0 の場合、ステートマシンはこの状態から stream_out_idle 状態に入ります。

12.4.6. ループバックの例: FPGA がスレーブ FIFO からデータを読み出し、そのデータをスレーブ FIFO に書き戻す

ステートマシンは、1 ループバック サイクルの間 6 つの状態を通ります。ステートマシンおよび対応する動作を下図に示します。

図 49. ループバック転送用のステートマシン



loop_back_idle 状態:

この状態では、ステートマシンで使用するすべてのレジスタと信号を初期化します。スレーブ FIFO 制御ラインの状態は以下になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 3

loop_back_flagc_rcvd 状態:

flagc_d = 1 になると、ステートマシンはこの状態に入ります。

loop_back_wait_flagd 状態:

1 クロック サイクル後、ステートマシンはこの状態に入り、flagd を待ちます。

loop_back_read 状態:

flagd_d = 1 になると、ステートマシンはこの状態に入ります。ここで、ステートマシンは以下のように読み出し制御信号をアサートします。

PKTEND# = 1 ; SLOE# = 0 ; SLRD# = 0 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 3

loop_back_read_rd_oe_delay 状態:

flagc_d = 0 になると、ステートマシンはこの状態に入ります。スレーブ FIFO 制御ラインの状態は以下になります。

PKTEND# = 1 ; SLOE# = 0 ; SLRD# = 0 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 3

部分的フラグの一般式節の式 (2b) によると、部分的フラグ (flagd) が 0 になった後、FX3 はアサートされている SLRD#を 3 サイクルの間サンプリングする必要があります。FPGA からインターフェースまでの伝播遅延が 1 サイクルであることを前提にすると、FPGA は flagd_d (flagd のフリップフロップされた出力) を 0 としてサンプリングした後に 1 サイクルの間 SLRD#をアサートします。

loop_back_read_oe_delay 状態:

rd_oe_delay_cnt = 0 になると、ステートマシンはこの状態に入ります。スレーブ FIFO 制御ラインの状態は以下になります。

PKTEND# = 1 ; SLOE# = 0 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 3

loop_back_wait_flaga 状態:

oe_delay_cnt = 0 になると、ステートマシンは loop_back_
wait_flaga 状態に入ります。スレーブ FIFO 制御ラインの状態は以下になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

loop_back_wait_flagb の状態:

flaga_d = 1 になると、ステートマシンは loop_back_wait_
flaga 状態に入ります。スレーブ FIFO 制御ラインの状態は以下になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

loop_back_write 状態:

flagb_d = 1 になると、ステートマシンは loop_back_wait_
flaga 状態に入ります。スレーブ FIFO 制御ラインの状態は以下になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 0 ; A[1:0] = 0

loop_back_write_wr_delay 状態:

flagb_d = 0 になると、ステートマシンは loop_back_wait_
flaga 状態に入ります。スレーブ FIFO 制御ラインの状態は以下になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

部分的フラグの一般式の式 (1) によると、部分的フラグ (flagb) が 0 になった後、FX3 はアサートされている SLWR#を 2 サイクルの間サンプリングする必要があります。FPGA からインターフェースまでの伝播遅延が 1 サイクルであることを前提にすると、FPGA は部分的 flagb_d (flagb のフリップフロップされた出力) を 0 としてサンプリングした後に 1 サイクルの間 SLWR#をアサートします。

loop_back_flush_fifo 状態:

1 クロック サイクル後、ステートマシンはこの状態に入り、内部 FIFO をフラッシュします。スレーブ FIFO 制御ラインの状態は以下になります。

PKTEND# = 1 ; SLOE# = 1 ; SLRD# = 1 ; SLCS# = 0 ; SLWR# = 1 ; A[1:0] = 0

12.5. プロジェクトの動作

12.5.1. ループバック転送のテスト手順

1. SuperSpeed Explorer Kit を使用する場合、ジャンパ J5 が開いていることを確かめてください。FX3 開発キット (CYUSB3KIT-001) を使用する場合、表 7 に基づいてジャンパとスイッチをセットアップしてください。HSMC コネクタで FX3 DVK 基板を Altera Cyclone III FPGA Starter 基板と接続し、FX3 DVK 基板に電源投入してから Altera Cyclone III FPGA Starter 基板に電源投入します。
2. ファームウェア イメージ ファイル *SF_loopback.img* を FX3 デバイスにロードします。FX3 SDK に同梱された Control Center を使って USB から FX3 をプログラムできます。Altera Cyclone III FPGA のプログラミングの前に FX3 をプログラムする必要があります。ファームウェアをダウンロードした後、FX3 デバイスは (USB 3.0 ポートに接続すれば) SuperSpeed デバイスとしてエニュメレーションします。

図 50. Control Center を使って FX3 ファームウェアをプログラム

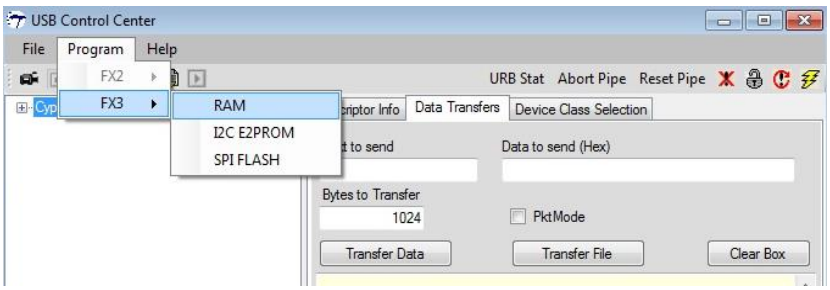
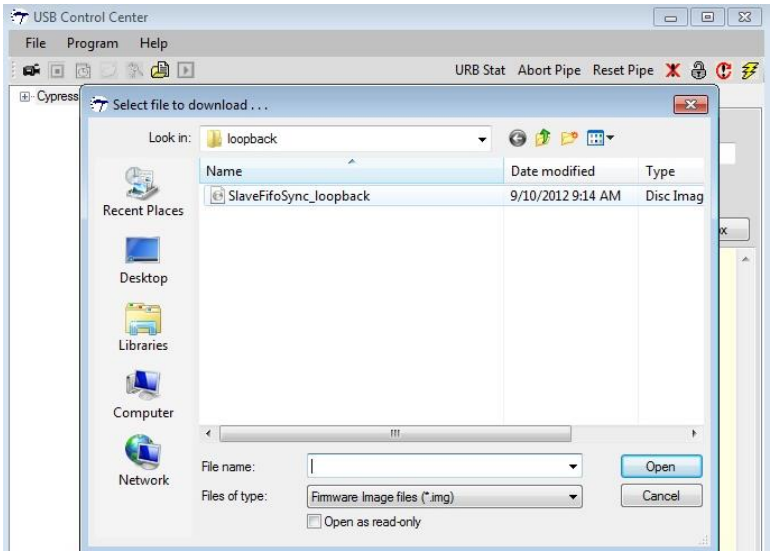
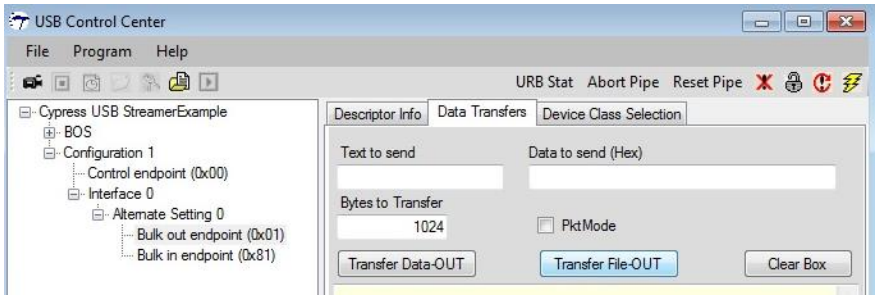


図 51. Control Center を使ってループバック テスト用に FX3 ファームウェアをプログラム



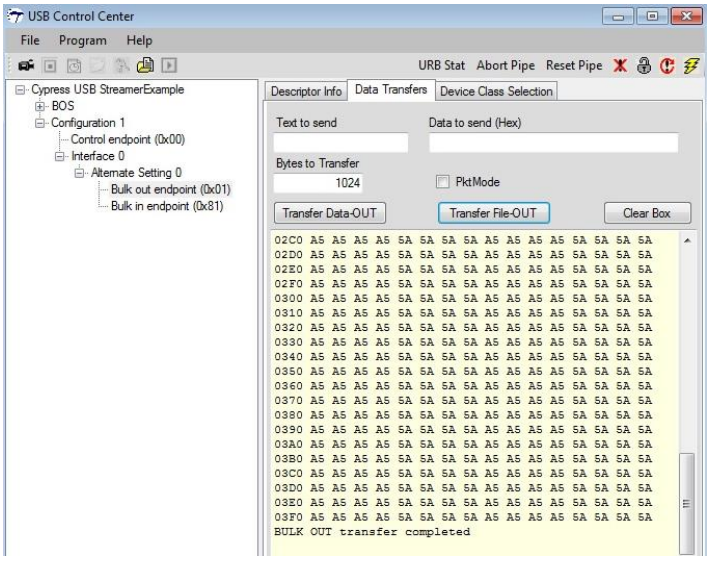
3. *slaveFIFO2b_loopback.sof* ファイルを Altera Cyclone III FPGA にロードします。FPGA は、Quartus II Web Edition 12.1 ソフトウェア同梱の USB-Blaster アプリケーションなどどの標準プログラマでもプログラムできます。
4. この時、転送は Control Center ユーティリティから起動できます。まず、Bulk OUT 転送を USB ホストから起動します。Control Center 内の Bulk OUT エンドポイントを選択して **Transfer File-OUT** ボタンをクリックします。

図 52. Transfer File-OUT でバルク OUT 転送を起動



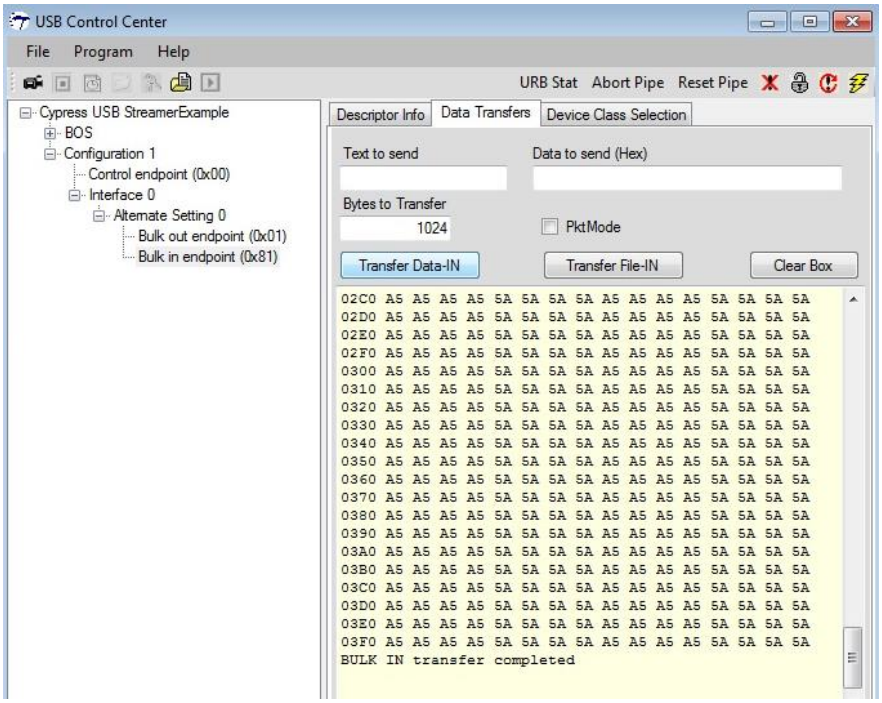
5. これにより、転送するデータを保存するファイルを閲覧し選択できます。本アプリケーション ノートの添付内の Loopback フォルダには *TEST.txt* ファイルがあります。このファイルには、交互方式で「0xA5A5A5A5 0x5A5A5A5A」を送信するデータ パターンが含まれます。ダブル クリックしてファイルを選択しデータを送信します。

図 53. Transfer File-OUT で TEST.txt ファイルを選択することでデータ パターンを転送



6. FPGA は、FLAGA が 1 になることを待つ状態に入りました。FPGA はデータが PIB_SOCKET_0 のバッファに格納されるとすぐに、そのデータを読み出します。その後、FPGA は同じデータをループバックして FX3 の PIB_SOCKET_3 に書き込みます。
7. USB ホストから Bulk IN 転送を発行できます。Control Center 内の Bulk IN エンドポイントを選択して **Transfer File-IN** ボタンをクリックします。この時、以前に書き込まれたデータは読み戻されます。

図 54. Transfer Data-IN を使って Bulk IN 転送を起動することでループバック テストを実行



12.5.2. ストリーム転送のテスト手順

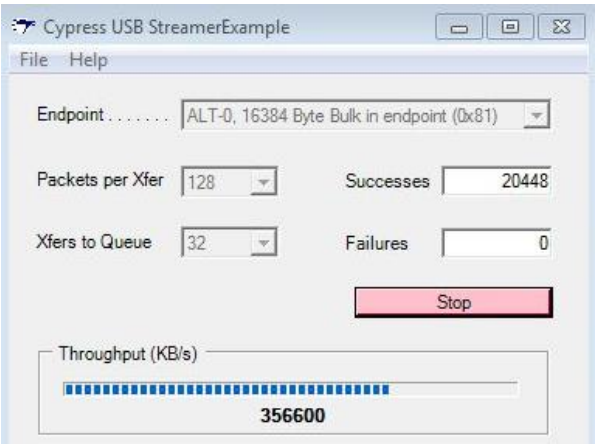
注: 以下のパスにある FX3 SDK 同梱の C++ Streamer ユーティリティをご使用ください。

<FX3_SDK_installation_path>*EZ-USB FX3 SDK\1.3\application\cpp\streamer\x86*

このパスのリリース 1.3 は FX3 SDK バージョン番号です。FX3 SDK の将来のリリースではバージョンアップできます。

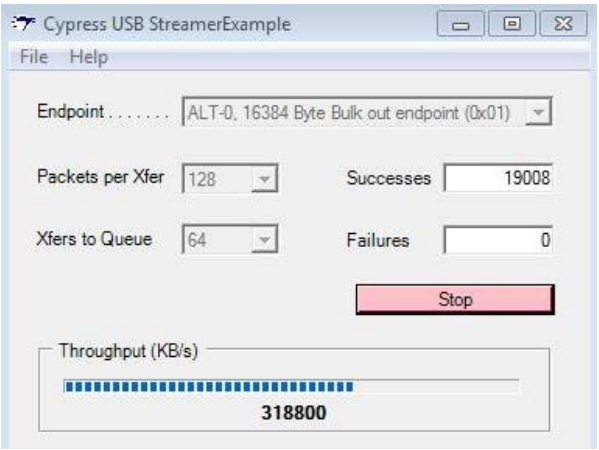
1. HSMC コネクタで FX3 DVK 基板を Altera Cyclone III FPGA Starter 基板と接続し、FX3 DVK 基板と Altera Cyclone III FPGA Starter 基板に電源を投入します。
2. ストリーム IN／OUT 転送を実行する場合、FX3 デバイスにファームウェア イメージ ファイル *SF_streamIN.img* をロードします。FX3 SDK に同梱された Control Center を使って USB から FX3 をプログラムできます。Altera Cyclone III FPGA のプログラミングの前に FX3 をプログラムする必要があります。ファームウェアをダウンロードした後、FX3 デバイスは (USB 3.0 ポートに接続すれば) SuperSpeed デバイスとしてエニュメレーションします。
3. Altera Cyclone III FPGA を *slaveFIFO2b_streamIN.sof* ファイル (ストリーム IN 転送の場合)、または *slaveFIFO2b_streamOUT.sof* ファイル (ストリーム OUT 転送の場合) でプログラムします。FPGA は Quartus II ソフトウェア同梱の USB-Blaster アプリケーションなどどの標準プログラマでもプログラムできます。
4. ストリーム IN の場合、FPGA は FLAGA が 1 になることを待つ状態に入ります。FPGA はバッファが使用可能になるとすぐに、FX3 の PIB_SOCKET_0 に連続的に書き込みます。USB ホストから連続 Bulk IN 転送を発行できます。Cypress Streamer ユーティリティ内の Bulk IN エンドポイントを選択して **Start** ボタンをクリックします。転送速度が表示されます。図 55 に示される転送速度は、Intel Z77 Express チップセットを搭載した Win7 64 ビット PC で観察されるものです。

図 55. Cypress Streamer ユーティリティで表示されるストリーム IN の転送速度



5. ストリーム OUT の場合、FPGA は FLAGC が 1 になることを待つ状態に入ります。FPGA はデータが使用可能になるとすぐに、FX3 の PIB_SOCKET_3 から連続的に読み出します。USB ホストから連続 Bulk OUT 転送を発行できます。Cypress Streamer ユーティリティ内の Bulk OUT エンドポイントを選択して **Start** ボタンをクリックします。転送速度が表示されます。図 56 に示される転送速度は、Intel Z77 Express チップセットを搭載した Win7 64 ビット PC で観察されるものです。

図 56. Cypress Streamer ユーティリティで表示されるストリーム OUT の転送速度

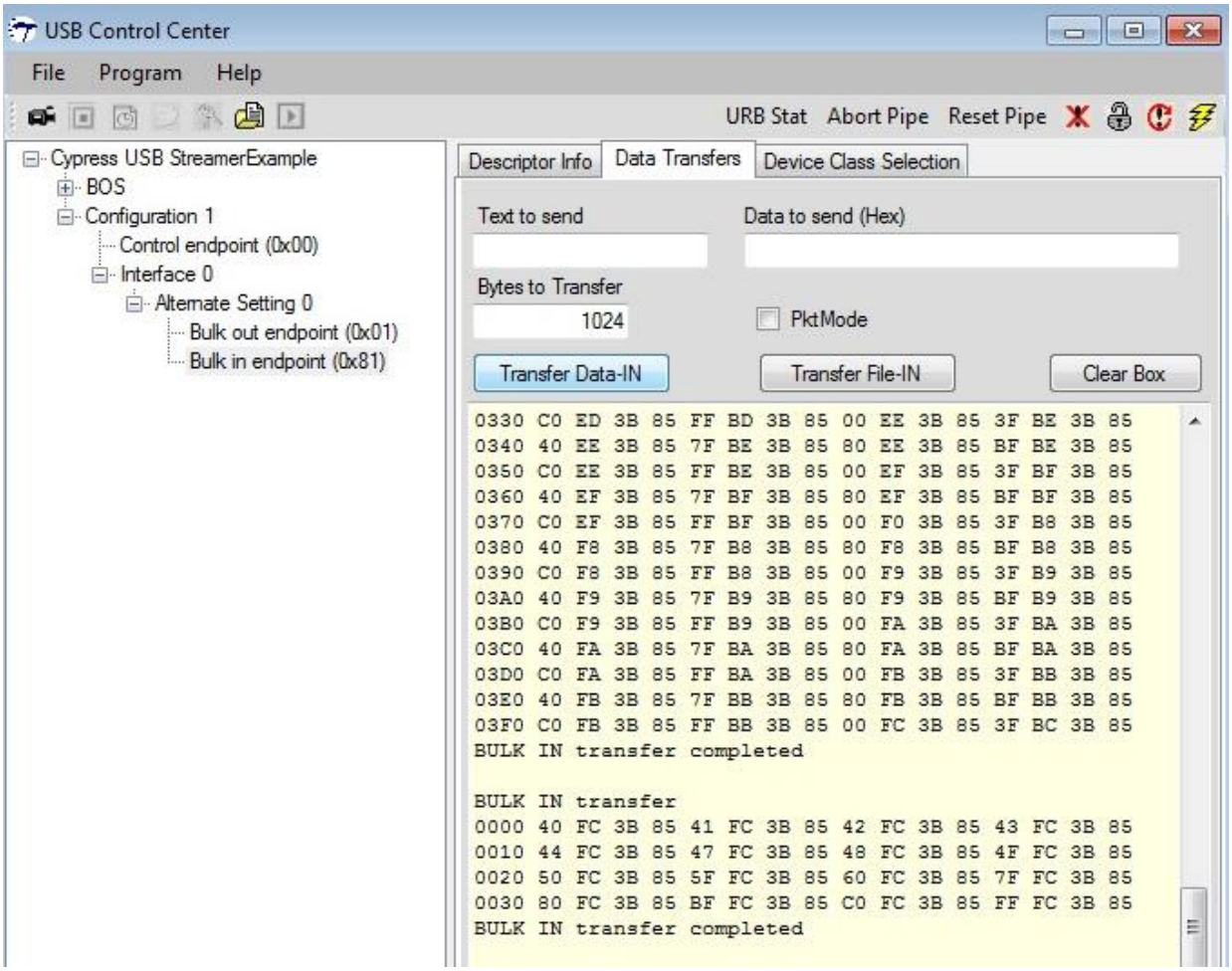


*SF_streamIN.img ファイル*はストリーム IN とストリーム OUT 両方に使用できます。*SF_streamIN.img* では、8 個のバッファが P2U DMA チャネルに割り当てられ、もう 4 個のバッファが U2P チャネルに割り当てられます。*SF_streamOUT.img* では、8 個のバッファが U2P DMA チャネルに割り当てられ、もう 4 個のバッファが P2U チャネルに割り当てられます。そのため、*SF_streamIN.img* ファームウェア ファイルで P2U チャネルのより高い性能を実現でき、*SF_streamOUT.img* ファームウェア ファイルで U2P チャネルのより高い性能を実現できます。

12.5.3. ショートパケット転送のテスト手順

1. HSMC コネクタで FX3 DVK 基板を Altera Cyclone III FPGA Starter 基板と接続し、FX3 DVK 基板と Altera Cyclone III FPGA Starter 基板に電源を投入します。
2. ストリーム IN／OUT 転送を実行する場合、FX3 デバイスにファームウェア イメージ ファイル *SF_shrt_ZLP.img* をロードします。FX3 SDK に同梱された Control Center を使って USB から FX3 をプログラムすることができます。Altera Cyclone III FPGA のプログラミングの前に FX3 をプログラムする必要があります。ファームウェアをダウンロードした後、FX3 デバイスは (USB 3.0 ポートに接続すれば) SuperSpeed デバイスとしてエニュメレーションします。
3. *slaveFIFO2b_partial.sof* ファイルを Altera Cyclone III FPGA にロードします。FPGA は Quartus II ソフトウェア同梱の USB-Blaster アプリケーションなどどの標準プログラマでもプログラムできます。
4. FX3 ファームウェアがプログラムされるとすぐに、PIB_SOCKET_0 に割り当てられるバッファは使用可能になります。FLAGA 信号を監視することで、FPGA はこの条件を待機する状態に入りました。FPGA はフラグが 1 になるとすぐに、FX3 に書き込みはじめます。
5. FPGA はフルパケット (1024 バイト) に続いてショートパケットを書き込みます。
6. ここで、USB ホストは Bulk IN トークンを発行できます。Control Center ユーティリティ内の Bulk IN エンドポイントを選択して **Transfer Data-IN** ボタンをクリックします。まず、フルパケットが受信されます。**Transfer Data-IN** をもう 1 度クリックします。ここで、ショートパケットが受信されます。

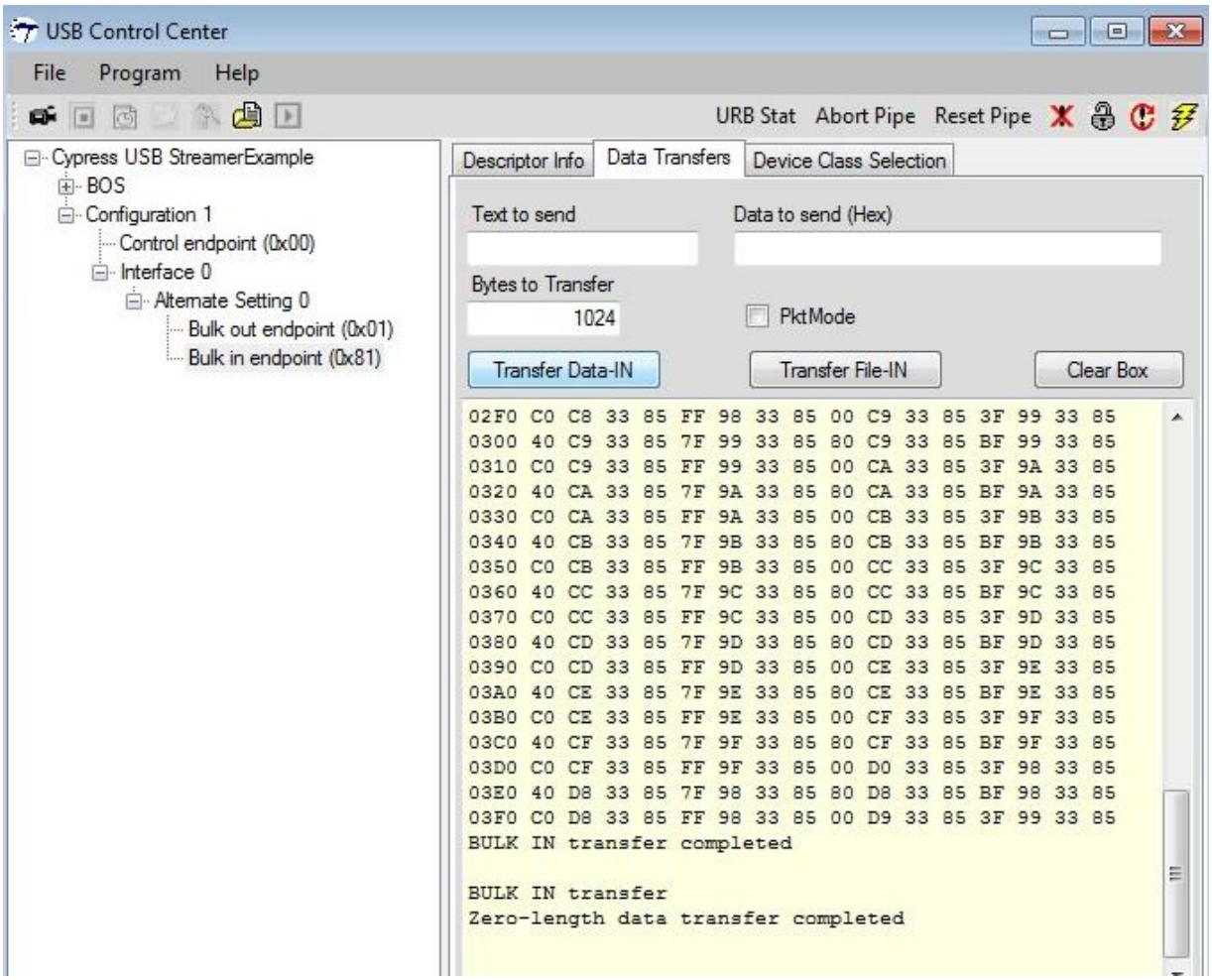
図 57. 連続した Tranfer Data-IN 動作でフルパケットに続いてショートパケットが受信される



12.5.4. ZLP 転送のテスト手順

1. HSMC コネクタで FX3 DVK 基板を Altera Cyclone III FPGA Starter 基板と接続し、FX3 DVK 基板と Altera Cyclone III FPGA Starter 基板に電源を投入します。
2. ストリーム IN／OUT 転送を実行する場合、FX3 デバイスにファームウェア イメージ ファイル *SF_shrt_ZLP.img* をロードします。FX3 SDK に同梱された Control Center を使って USB から FX3 をプログラムできます。Altera Cyclone III FPGA のプログラミングの前に FX3 をプログラムする必要があります。ファームウェアをダウンロードした後、FX3 デバイスは (USB 3.0 ポートに接続すれば) SuperSpeed デバイスとしてエニュメレーションします。
3. *slaveFIFO2b_ZLP.sof* ファイルを Altera Cyclone III FPGA にロードします。FPGA は Quartus II ソフトウェア同梱の USB-Blaster アプリケーションなどどの標準プログラマでもプログラムできます。
4. FX3 ファームウェアがプログラムされるとすぐに、PIB_SOCKET_0 に割り当てられるバッファは使用可能になります。FLAGA 信号を監視することで、FPGA はこの条件を待機する状態に入りました。FPGA はフラグが 1 になるとすぐに、FX3 に書き込みはじめます。
5. FPGA はフルパケット (1024 バイト) に続いて ZLP を書き込みます。
6. ここで、USB ホストは Bulk IN トークンを発行できます。Control Center ユーティリティ内の Bulk IN エンドポイントを選択して **Transfer Data-IN** ボタンをクリックします。まず、フルパケットが受信されます。**Transfer Data-IN** をもう 1 度クリックします。ここで、ZLP が受信されます。

図 58. 連続した Transfer Data-IN 動作でフルパケットに続いて長さゼロのパケット (ZLP) が受信される



7. FPGA が連続的にデータを書き込んでいるため、複数の BULK IN 転送は行われることにご注意ください。

13. 関連プロジェクト ファイル

ファイル名／フォルダ名		説明
FX3 ファームウェア		FX3 ファームウェアのソース
FPGA ソース ファイル	fx3_slaveFIFO2b_xilinx	このフォルダには以下のサブフォルダが含まれます。 *fpga_slavefifo2b_verilog* Xilinx FPGA の Verilog ソース コードはすべての転送形式 (ストリーム IN、ストリーム OUT、ショートパケット、ZLP、ループバック) に対応 *Fpga_slavefifo2b_vhdl* Xilinx FPGA の VHDL ソース コードはすべての転送形式 (ストリーム IN、ストリーム OUT、ショートパケット、ZLP、ループバック) に対応
	Fx3_slaveFIFO2b_altera	Altera FPGA の Verilog と VHDL ソース コード このフォルダには以下の各種転送のサブフォルダが含まれます。各サブフォルダには Verilog と VHDL 両方のプロジェクトが含まれます。 ループバック データ転送用の *fpga_loopback* ショートパケット データ転送用の *fpga_partial* ストリーム IN データ転送用の *fpga_streamIN* ストリーム OUT 転送用の *fpga_streamOUT* ZLP 転送用の *fpga_zlp*
SF_loopback.img		この FX3 ファームウェア イメージはスレーブ FIFO の実装を含み、ループバック転送に必要な DMA チャネルをセットアップします。 **注:** 提供される FX3 ファームウェア イメージ同士の間に、各種転送のデモを簡単化することを目的とする DMA チャネルのセットアップ方法上の相違点があります。しかし、各種転送をデモするのにどのファームウェア イメージでも使用可能です。 このイメージは、ファームウェアのソースコードに次のマクロ定義の変更を加えた後に生成されます。 STREAM_IN_OUT' – Disabled LOOPBACK_SHRT_ZLP – Enabled
SF_streamIN.img		この FX3 ファームウェア イメージはスレーブ FIFO の実装を含み、ストリーム IN 転送を実行する際の性能の最適化に必要な DMA チャネルをセットアップします。 このイメージは、ファームウェアのソースコードに次のマクロ定義の変更を加えた後に生成されます。 STREAM_IN_OUT' – Enabled LOOPBACK_SHRT_ZLP - Disabled
SF_streamOUT.img		この FX3 ファームウェア イメージはスレーブ FIFO の実装を含み、ストリーム OUT 転送を実行する際の性能の最適化に必要な DMA チャネルをセットアップします。 このイメージは、ファームウェアのソースコードに次のマクロ定義の変更を加えた後に生成されます。 STREAM_IN_OUT' – Enabled LOOPBACK_SHRT_ZLP - Disabled
SF_shrt_ZLP.img		この FX3 ファームウェア イメージはスレーブ FIFO の実装を含み、ストリーム ショートパケットまたは長さゼロのパケット (ZLP) を実行する際の性能の最適化に必要な DMA チャネルをセットアップします。 このイメージを使用する際、フルパケットに続いてショートパケットまたは長さゼロのパケットの連続ストリームをご確認ください。 STREAM_IN_OUT' – Disabled LOOPBACK_SHRT_ZLP - Enabled
TEST.txt		このファイルは Control Center ユーティリティ内の **Transfer File-OUT** ボタンを使って送信されるデータ パターンを含む

14. まとめ

スレーブ FIFO インターフェースは、外部 FPGA やプロセッサ、デバイスが EZ-USB FX3 の内蔵 FIFO バッファへのデータ読み書きアクセスを実行する必要があるアプリケーションに最適です。

本アプリケーション ノートは同期スレーブ FIFO インターフェースの詳細を説明しました。また本資料は、各種フラグ コンフィギュレーションおよびフラグ設定に GPIF II Designer ツールを使用する方法について説明しました。2 つの完全な設計例も示しました。

付録 A: トラブルシューティング

状況 1

BULK IN と BULK OUT データ転送が失敗しました。

```
BULK IN transfer
BULK IN transfer failed with Error Code:997
```

```
BULK OUT transfer
BULK OUT transfer failed with Error Code:997
```

原因および解決策

スレーブ FIFO インターフェースに接続された FPGA は連続してデータを送信しない、またはインターフェース クロック (PCLK) はより低い周波数で実行しています。本シナリオでは、USB リンクが低消費電力状態のままになる可能性があります。

以下のように **CyU3PUsbLPMDisable** 関数を使って低消費電力状態を終了します。

```
CyFxSlFifoApplnUSBEventCB (
    CyU3PusbEventType_t evtype,
    uint16_t            evdata
    )
{
    switch (evtype)
    {
        case CY_U3P_USB_EVENT_SETCONF:
            /* Stop the application before re-starting. */
            if (glIsApplnActive)
            {
                CyFxSlFifoApplnStop ();
            }
            CyU3PUsbLPMDisable();
            /* Start the loop back function. */
            CyFxSlFifoApplnStart ();
```

注: この解決策は USB コンプライアンスに合格しないことがあります。SDK に同梱されている **USBBulkSourceSink** の例から別の解決策を実施することができます (「**CyU3PusbSetLinkPowerState (CyU3PUsbLPM_U0)**」をご検索ください)。この解決策を最終のファームウェアに使用する、またはサイプレス テクニカル サポートまでご連絡ください。

この解決策を実施しても FX3 DVK で BULK IN 転送がまだ失敗した場合、ジャンパ J100 の 1 番目のピンと 2 番目のピンが接続されることを確認してください。これらピンを接続すると FLAGA がスレーブ FIFO インターフェースに接続された FPGA に使用できます。

状況 2

SLWR#をアサートせずに PKTEND#信号がアサートされているにもかかわらず、FX3 の DMA バッファ サイズよりも小さくて パケット サイズ (USB3.0 の場合は 1024 バイト、USB2.0 の場合は 512 バイト) の倍数であるデータの読み出し後に ZLP を取得しません。

原因および解決策

DMA バッファ サイズが 2KB に設定されていることを前提にします。FPGA は 1KB のデータを DMA バッファに書き込み、そのデータが USB ホストに転送されることを期待します。USB ホストは FX3 から 2KB のデータを要求します。

本シナリオでは、FX3 は USB ホストから要求された転送を終了するために 1KB のデータと共に ZLP を送信する必要があります。

FPGA がスレーブ FIFO インターフェースにデータを書き込んでいる間、GPIF II ステートマシンは「Write」状態に入ります。パケット サイズの倍数であるデータを書き込んだ直後に ZLP を送信するためには、SLWR#信号がアサート停止された後、FPGA は少なくとも 2 クロック サイクルの間 PKTEND#信号をアサートする必要があります。理由は、GPIF II ステートマシンが「Write」状態から「ZLP」状態に遷移するために 2 クロック サイクルを要するためです。

状況 3

ファームウェアを FX3 にダウンロードすると FLAGA は LOW になります。

原因および解決策

DMA チャネルが正常に作成されたかを確認します。GPIF スレッド 0 に対応する DMA チャネルが失敗した場合、そのスレッドに対応する FLAG は誤った状態を反映します。**CyU3PdmaChannelCreate** 関数の返り値をチェックして DMA チャネルが正常に作成されたか確認できます。

CyU3PdmaChannelCreate 関数の失敗の主な原因は以下に示します。両方の場合においては、この関数は CY_U3P_ERROR_MEMORY_ERROR コードを返します。

- DMA バッファの割り当てが失敗: すべての使用可能な (SYS MEM – CODE MEM) メモリを DMA バッファとして割り当てられます。システム メモリ (SYS MEM) はご使用の製品番号に依存し、コード メモリ (CODE MEM) はご開発のアプリケーション コードに依存します。合計 DMA バッファ サイズ (バッファ数 * 各 DMA バッファ サイズ) が使用可能な合計バッファ空間 (SYS MEM – CODE MEM) より小さいことをご確認ください。
- DMA バッファのディスクリプタの割り当てが失敗: バッファ ディスクリプタは各 DMA バッファのサイズと状態を追跡する構造です。システムには 512 個のディスクリプタがあります。それぞれのバッファは AUTO チャネル用に 1 個のディスクリプタと MANUAL チャネル用に 2 個のディスクリプタを必要とします。

DMA バッファの数は両方の条件を満たす必要があります。

状況 4

FPGA が 32 ビット データ バスを介して 100MHz でデータを FX3 のスレーブ FIFO インターフェースに書き込んでいる間、多くの DMA オーバーフロー エラーが生じます。

原因および解決策

19.2MHz の水晶またはクロック ソースを使う時に FX3 マスター クロックが 384MHz に設定される場合、および GPIF II が 32 ビット幅として設定され 100MHz で動作する場合、DMA のオーバーフロー エラーは GPIF II に発生することがあります。

CyU3PsysClockConfig_t 構造の setSysClk400 パラメーターは、FX3 デバイスのマスター クロックが 400MHz より大きい周波数に設定されるかを示します。このパラメーターを設定して、FX3 のマスター クロック周波数を **CyU3PdeviceInit** の呼び出しの間に 403.2MHz に設定します。この構造は **main** 関数の **CyU3PdeviceInit** の呼び出しにパラメーターとして渡されます。

```
/* setSysClk400 clock configurations */
        clkCfg.setSysClk400 = CyTrue;    /* FX3 device's master clock is set to a frequency >
400 MHz */
        clkCfg.cpuClkDiv = 2;            /* CPU clock divider */
        clkCfg.dmaClkDiv = 2;            /* DMA clock divider */
        clkCfg.mmioClkDiv = 2;           /* MMIO clock divider */
        clkCfg.useStandbyClk = CyFalse; /* device has no 32-kHz clock supplied */
        clkCfg.clkSrc = CY_U3P_SYS_CLK; /* Clock source for a peripheral block  */

    /* Initialize the device */
    status = CyU3PdeviceInit (&clkCfg);
    if (status != CY_U3P_SUCCESS)
    {
        goto handle_fatal_error;
    }
```

注: お使いのアプリケーションの問題がまだ解決していない場合、サイプレス テクニカル サポートまでご連絡ください。

付録 B: FX3 DVK (CYUSB3KIT-001) を用いたハードウェア セットアップ

図 59 に、FX3 開発キット (CYUSB3KIT-001) と Xilinx SP601 基板を用いたハードウェア セットアップを示します。

図 59. FMC 相互接続基板を用いて Xilinx SP601 基板に接続したサイプレス FX3 開発キット

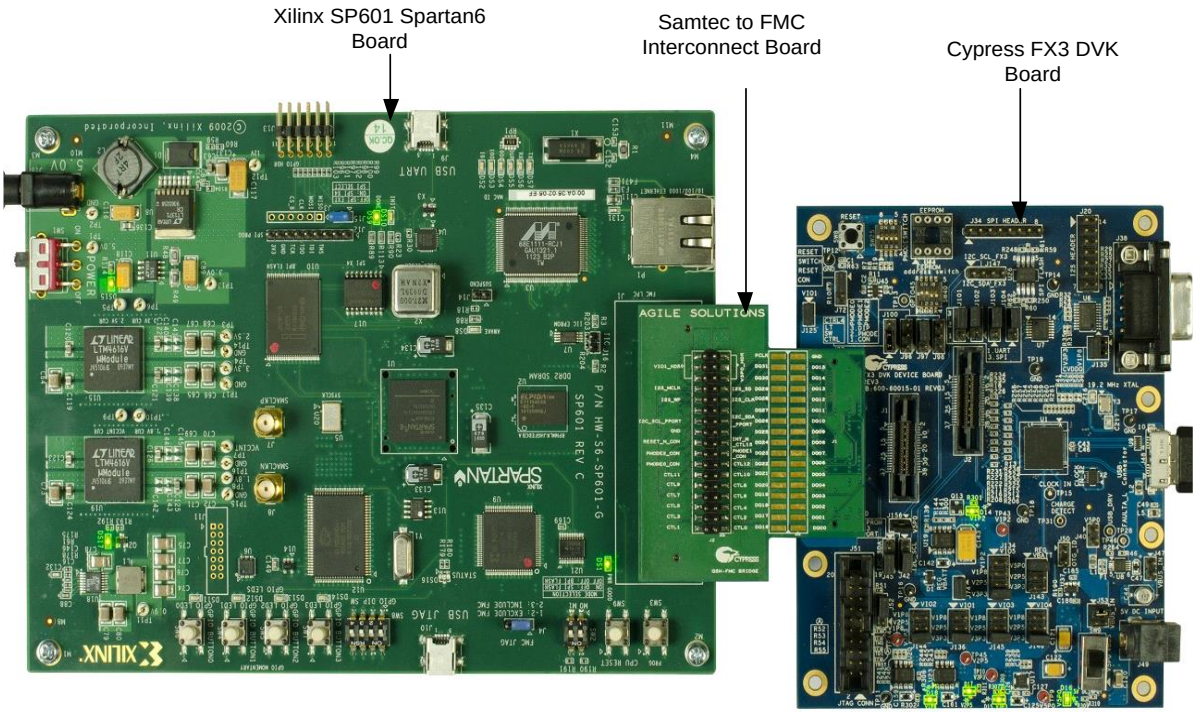
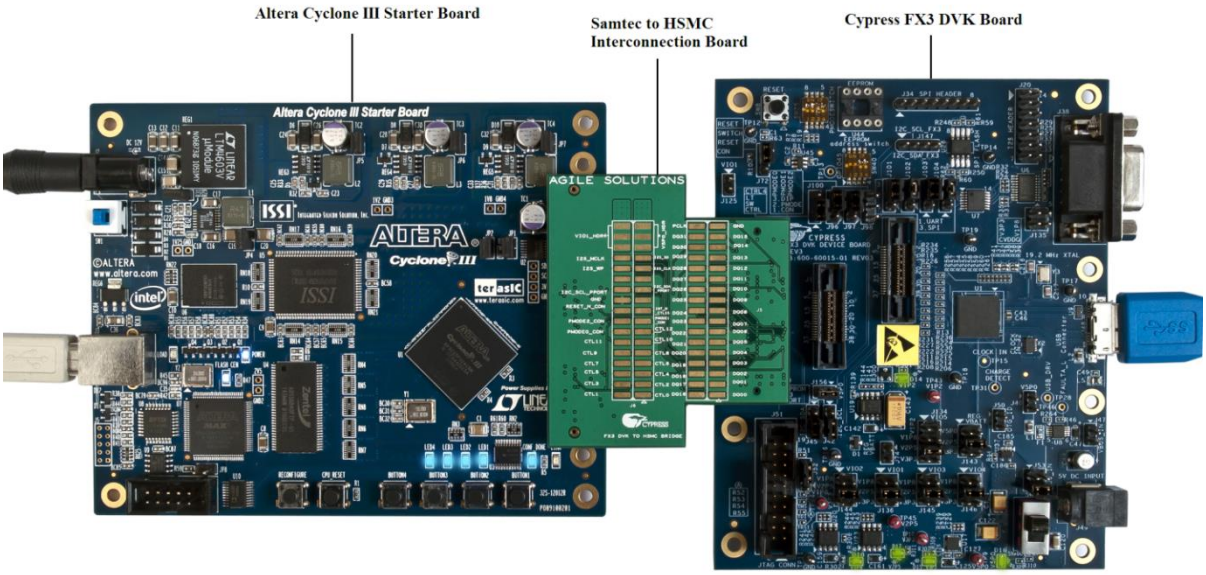


図 60 に、FX3 開発キット (CYUSB3KIT-001) と Altera Cyclone III 基板を用いたハードウェア セットアップを示します。

図 60. HSMC 相互接続基板を用いて Altera Cyclone III 基板に接続したサイプレス FX3 開発キット



B.1 ジャンパおよびスイッチの設定

表 7 は、デモを実行するための FX3 DVK 基板のジャンパおよびスイッチの設定です。FX3 DVK に接続された FPGA にかかわらずこれらの設定は同じです。

表 7. FX3 DVK 上のジャンパおよびスイッチの設定

シリアル番号	ジャンパ／スイッチ	ジャンパで短絡されるピン	機能
1	J100	1 と 2	GPIO[21]/CTL[4] – FLAGA に設定
2	J136	3 と 4	VIO1(3.3V)
3	J144	3 と 4	VIO2(3.3V)
4	J145	3 と 4	VIO3(3.3V)
5	J146	3 と 4	VIO4(3.3V)
6	J134	4 と 5	VIO5(3.3V)
7	J135	2 と 3	CVDDQ(3.3V)
8	J143	3 と 4	VBATT(3.3V)
9	J101	1 と 2	GPIO[53]=UART_RTS
10	J102	1 と 2	GPIO[54] = UART_CTS
11	J103	1 と 2	GPIO[56] = UART_TX
12	J104	1 と 2	GPIO[57] = UART_RX
13	J96 と SW25	2 と 3	SW25 を使って PMODE0 ピンの状態 (オン／オフ) を選択。SW25.1 がオフになるべき
14	J97 と SW25	2 と 3	SW25 を使って PMODE1 ピンの状態 (オン／オフ) を選択。SW25.1 がオフになるべき
15	J98	1 と 2	PMODE2 ピンが開放
16	J72	1 と 2	リセット
17	J53	1～3 または 2～4	バス パワー
18	SW9	スイッチが VBUS_IN という方向へ指すべき	バス パワー
19	J156	短絡用にジャンパを配置	Samtec コネクタに電源を供給
20	J45	2 と 3	GPIO[59] – FX3 から FPGA にリセット

注: PMODE ピンは USB ブート用にセットされます。表に示されないジャンパを開放したままにできます。

付録 C

本節で、FX3 内の DMA バッファ サイズと USB ホスト アプリケーション上の転送バッファ サイズが変更された時にスレーブ FIFO アプリケーションの 2 つのデータ転送モード (ショートパケットと ZLP) がどのように動作するかを示します。

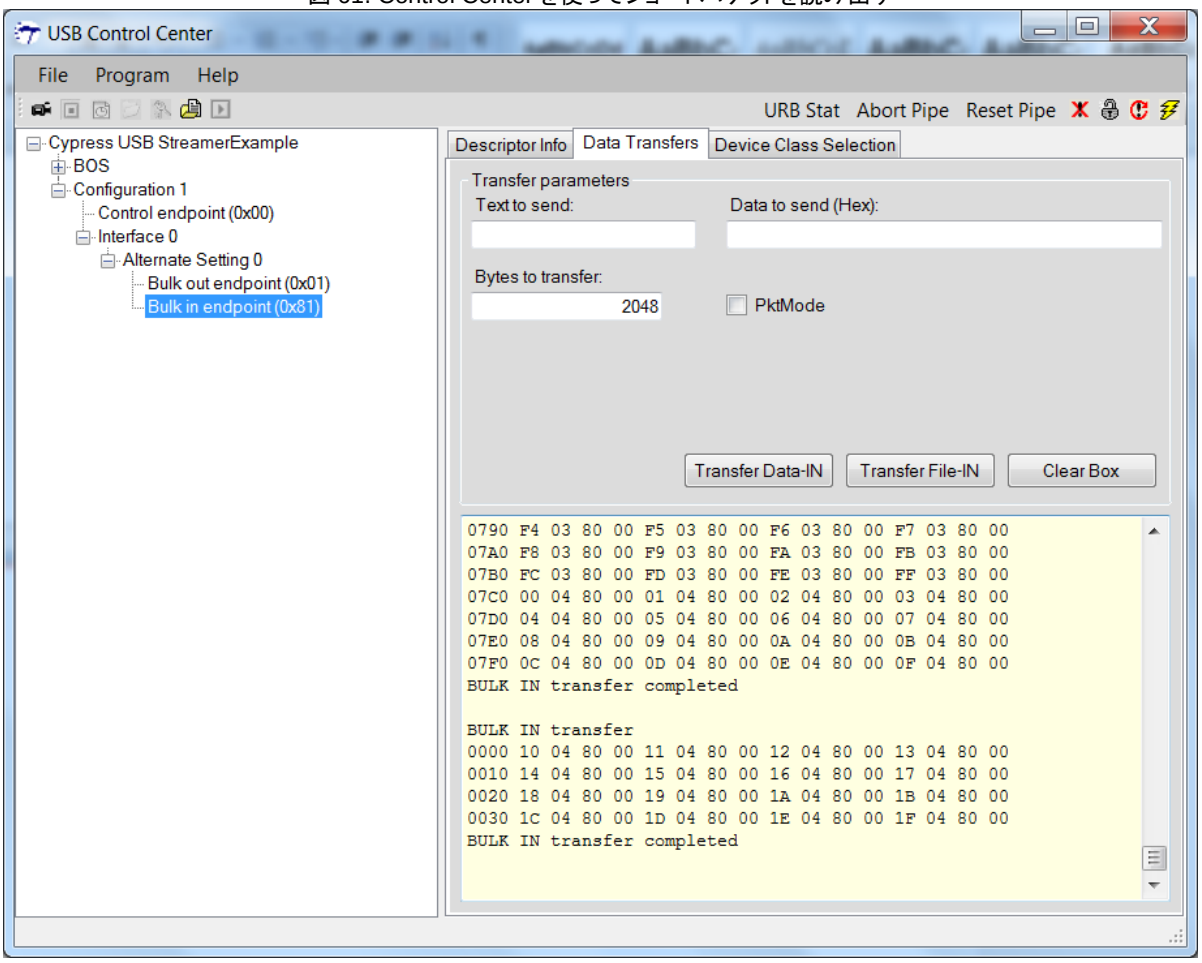
C.1 ショートパケット例

提供された FX3 ファームウェアに使用される DMA バッファ サイズは 1024 バイトで、バッファの数は 2 個です。本例では、FPGA は 1024 バイトのデータを 1 番目の DMA バッファに書き込んでから 64 バイトのデータを 2 番目の DMA バッファに書き込みます。

USB Control Center で「1024」を **Bytes to transfer** フィールド (転送バッファ サイズ) に入力し、**Transfer Data-IN** ボタンをクリックすることによりデータを読み出します。ユーザーは 1024 バイトのデータを取得します。「Transfer Data-IN」ボタンをもう 1 度クリックすると、FPGA によって書き込まれた 64 バイトのデータを取得します。

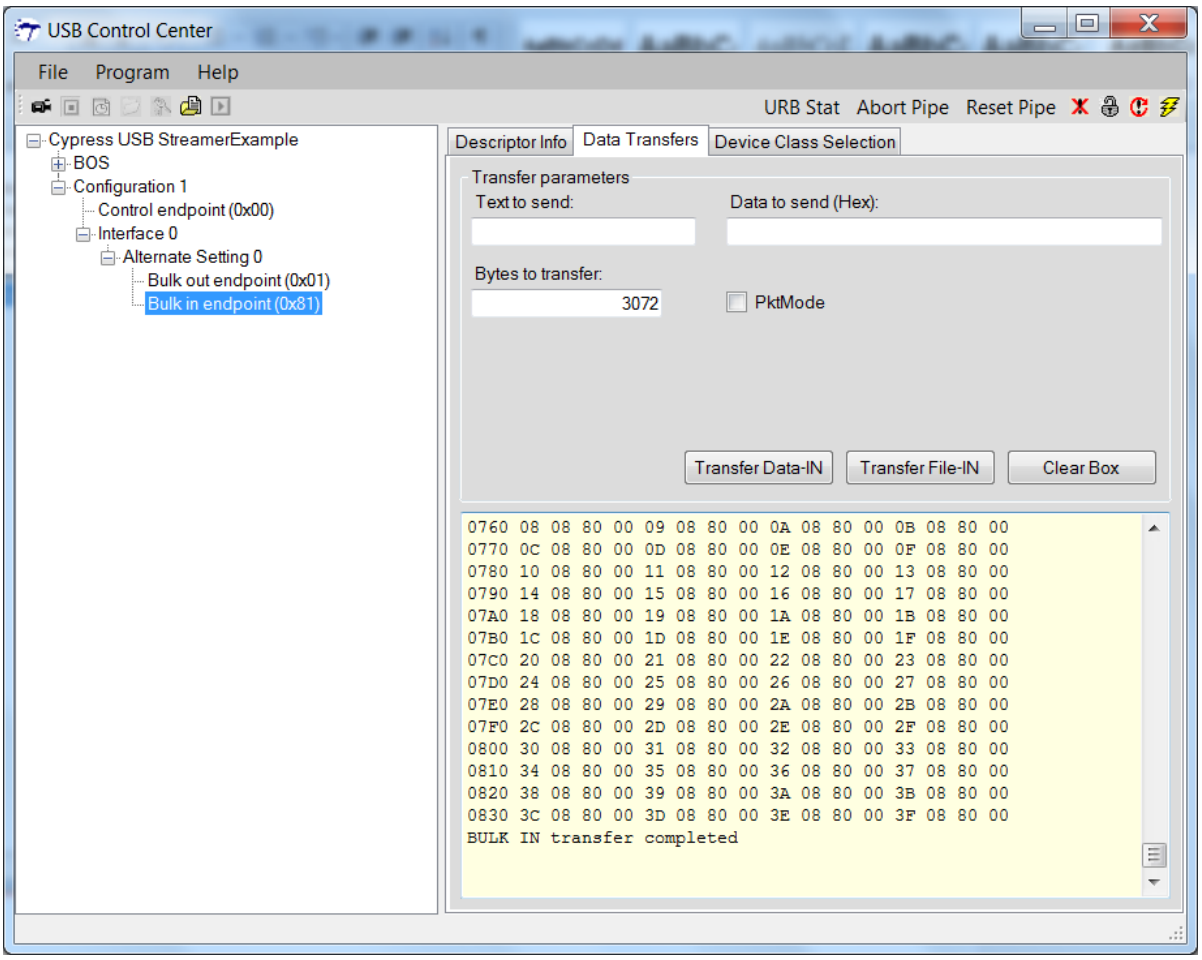
DMA バッファ サイズが 2048 バイトに変更されることを前提にします。FPGA は 2048 バイトのデータを 1 番目の DMA バッファに、64 バイトのデータを 2 番目の DMA バッファに書き込みます。FPGA は 1 番目の DMA バッファを完全に一杯にしてから、ショートパケットを次の使用可能な DMA バッファに書き込みます。さらなる DMA バッファが割り当てられる場合、この転送形式 (1 フルパケットに続いて 1 ショートパケット) が繰り返されます。Control Center ユーティリティでは、「2048」を **Bytes to transfer** フィールドに入力し、**Transfer Data-IN** ボタンをクリックすることによりデータを読み出します。これにより、ユーザーは 2048 バイトのデータを取得します。「Transfer Data-IN」ボタンをもう 1 度クリックすると、FPGA によって書き込まれた 64 バイトのデータを取得します。これを図 61 に示します。

図 61. Control Center を使ってショートパケットを読み出す



転送バッファ サイズを 3072 に増加すると、**Transfer Data-IN** をクリックするたびに 2112 (0x840) バイトのデータを取得します。これを図 62 に示します。

図 62. 要求されたバイト数が DMA バッファ サイズより大きい場合のショートパケット データ転送

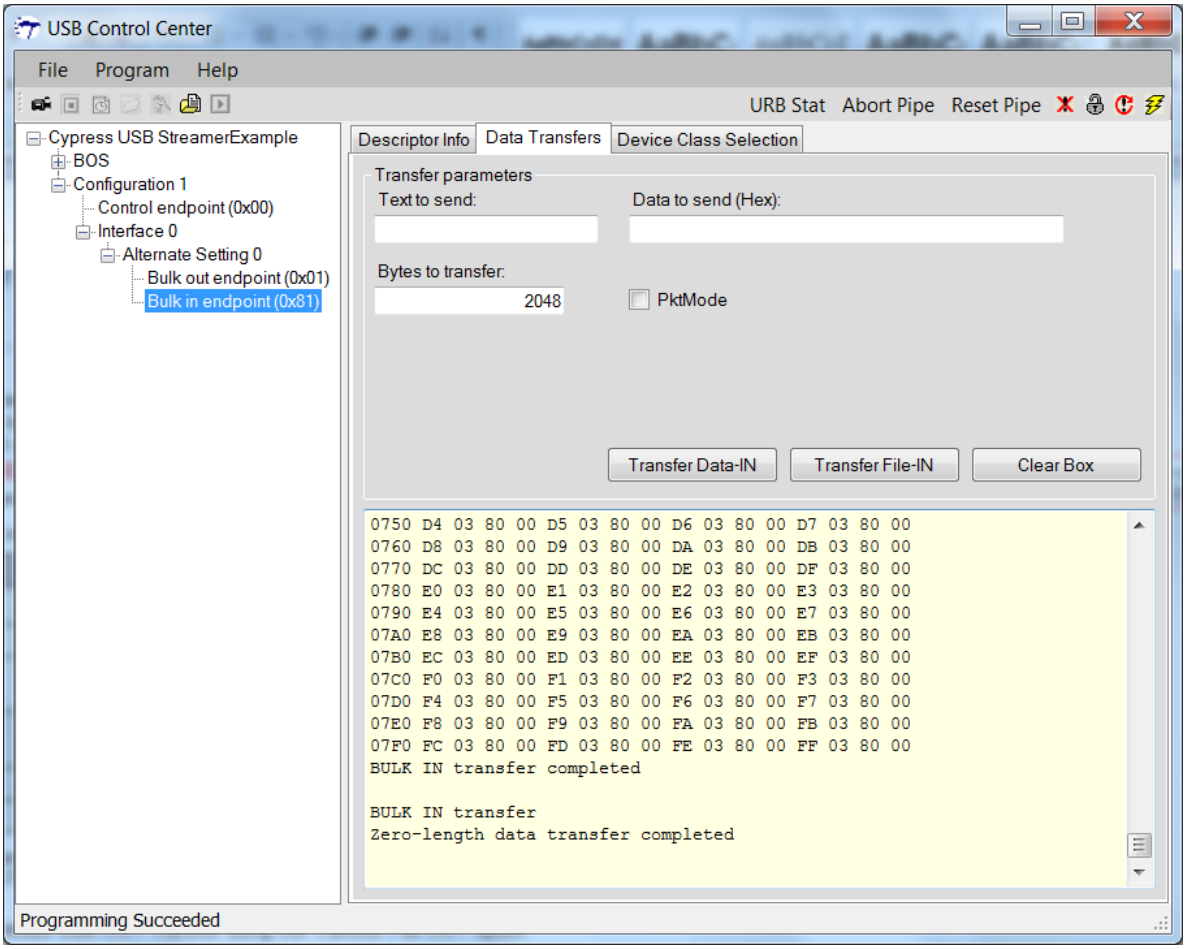


C.2 長さゼロのパケット (ZLP) 例

提供された FX3 ファームウェアに使用される DMA バッファ サイズは 1024 バイトで、バッファの数は 2 個です。本例では、FPGA は 1024 バイトのデータを書き込んで ZLP を生成するために必要な制御信号をアサートします。USB Control Center で、「1024」を **Bytes to transfer** フィールド (転送バッファ サイズ) に入力し、**Transfer Data-IN** ボタンをクリックすることによりデータを読み出します。ユーザーは 1024 バイトのデータを取得します。「Transfer Data-IN」ボタンをもう 1 度クリックすると、FPGA によって書き込まれた ZLP を取得します。

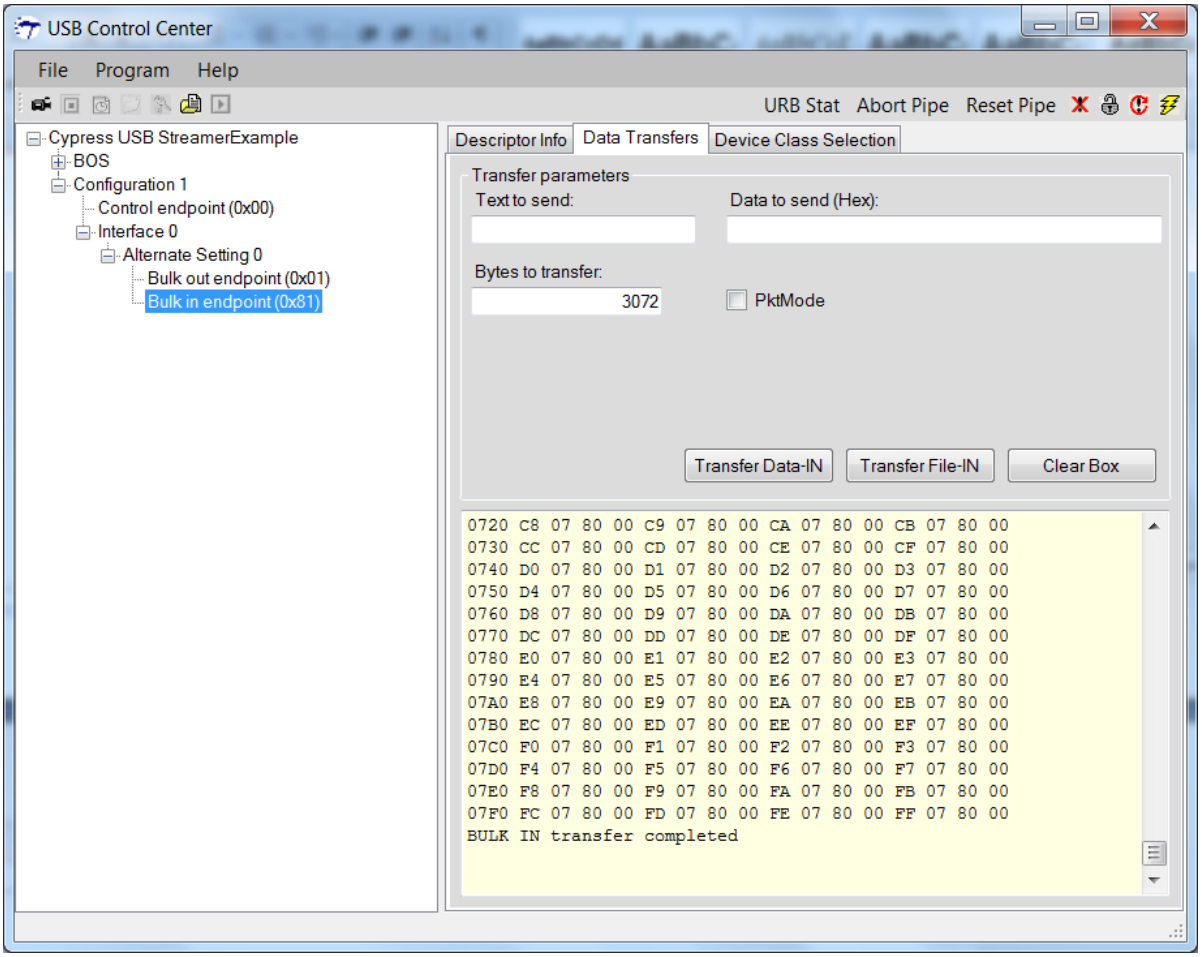
DMA バッファ サイズが 2048 バイトに変更することを前提にします。FPGA は 2048 バイトのデータを書き込んで ZLP を生成するために必要な制御信号をアサートします。FPGA は 1 番目の DMA バッファを完全に一杯にしてから、ZLP を生成します。さらなる DMA バッファが割り当てられる場合、この転送形式 (1 フルパケットに続いて 1 ZLP) が繰り返されます。Control Center ユーティリティでは、「2048」を **Bytes to transfer** フィールドに入力し、**Transfer Data-IN** ボタンをクリックすることによりデータを読み出します。これにより、ユーザーは 2048 バイトのデータを取得します。「Transfer Data-IN」ボタンをもう 1 度クリックすると、FPGA によって書き込まれた ZLP を取得します。これを図 63 に示します。

図 63. Control Center を使って ZLP を読み出す



転送バッファ サイズを 3072 (DMA バッファ サイズより大きい) に増加すると、**Transfer Data-IN** をクリックするたびに 2048 (0x800) バイトのデータを取得します。1 ZLP が 2048 バイトのデータの後に送信されますが、物理 USB バス上に ZLP が存在しても Control Center で表示されません。これを図 64 に示します。

図 64. 要求されたバイト数が DMA バッファ サイズより大きい場合の ZLP 転送



改訂履歴

文書名: AN65974 – EZ-USB FX3 スレーブ FIFO インターフェースを使った設計

文書番号: 001-92695

版	ECN	発行日	変更内容
**	4393307	05/29/2014	これは英語版001-65974 Rev. *Iからを翻訳した日本語版001-92695 Rev. **です。
*A	4928466	10/06/2015	これは英語版001-65974 Rev. *Lからを翻訳した日本語版001-92695 Rev. *Aです。
*B	5716287	04/27/2017	更新されたロゴと著作権。
*C	5894510	09/25/2017	これは英語版001-65974 Rev. *Nを翻訳した日本語版001-92695 Rev. *Cです。
*D	6640397	07/30/2019	これは英語版001-65974 Rev. *Oを翻訳した日本語版001-92695 Rev. *D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	cypress.com/wireless

PSoC® ソリューション

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

サイプレス開発者コミュニティ

コミュニティ | 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, 2010-2019. 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。いかなるコンピューティングデバイスも絶対に安全ということはない。従って，Cypress のハードウェアまたはソフトウェア製品に講じられたセキュリティ対策にもかかわらず，Cypress は，Cypress 製品への権限のないアクセスまたは使用といったセキュリティ違反から生じる一切の責任を負わない。加えて，本書面に記載された製品には，エラッタと呼ばれる設計上の欠陥またはエラーが含まれている可能性があり，公表された仕様とは異なる動作をする場合がある。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，CapSense，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。