XAPP1176 (v1.0) 2013 年 7 月 26 日

Vivado IP インテグレーターで AXI Quad SPI を XIP (eXecute-in-Place) モードに設定して使用

著者：Sanjay Kulkarni、Prasad Gutti

概要

このアプリケーション ノートでは、Vivado® Design Suite v2013.2 でリリースされた AXI Quad SPI v3.0 IP コアの新機能、XIP (eXecute-in-Place) について説明します。ここでは、FPGA を SPI シリアル フラッシュ デバイスからコンフィギュレーションするために必要な接続、および SPI モードのコンフィギュレーション フローについて説明します。このアプリケーション ノートでは Kintex®-7 (KC705) ボードで Numonyx 社製 SPI フラッシュ メモリを使用していますが、ソフトウェア サンプル ファイルを編集すれば任意のザイリンクス ボードが使用可能です。

エンベデッド システムでどのコード実行方法を選択するかは、部材コスト、プロセッサの機能、オペレーティング システムの性能など、多数の要因を考慮して決めることになります。エンベデッド アプリケーションの多くは性能が最優先事項ではないため、そのデザインにメモリ管理装置 (MMU) は必要ありません。XIP は、プログラムをブロック RAM にコピーせずに長期保存ストレージ (LTS) から直接実行する方法です。この方法では、共有メモリの使用を拡張することで全体的なメモリ要件が抑えられます。

このメカニズムを利用するには、いくつかの条件を満たす必要があります。プロセッサからは、このストレージは動作速度が遅い点を除いて通常のメモリと同じように見えなければなりません。プログラムは場所に依存しないこと、そしてメモリ内に格納されたイメージを変更しないことが条件となります。プログラムを格納するメモリにはさまざまな種類がありますが、エンベデッド システムでは DDR (Double Data Rate) や SRAM メモリよりもフラッシュ メモリの方が適しています。フラッシュ メモリには、低消費電力でピン インターフェイスの数が少ないという利点もあります。

前述の条件を満たすデバイスとして、パラレル/シリアル フラッシュ メモリがあります。SPI ベースのフラッシュ メモリは、SPI フラッシュ テクノロジの進歩、より高い集積度と動作速度、その他のベンダー ベースのサポートなどの点により、フラッシュからブートするという動作に適しています。このアプリケーション ノートでは、SPI フラッシュに格納した実行可能コードを XIP モードに設定した Quad SPI IP コアを介して DDR メモリに読み込み、インプリメントした XIP モードのストア/ロード機能を実際に使用してみます。また、XIP の 2 つのモード (デュアル/クワッド) を実際に用いて、クワッド モードの方が高いデータ転送レートが得られることを確認します。

本書の構成

このアプリケーション ノートの目的は、AXI Quad SPI コアを XIP モードで実際に使用し、その機能を示すことです。XIP モードのシステムは、Vivado® Design Suite に含まれるザイリンクス Vivado IP インテグレーター バージョン 2013.2 を使用して構築します。Vivado IP インテグレーターでは、プロセッサ、インターコネクト、割り込みコントローラー、ペリフェラル IP、メモリ コントローラー、UART をインスタンシエートして相互に接続することによってシステムを構築できます。Vivado IP インテグレーターの詳細は、『Vivado Design Suite ユーザー ガイド : IP インテグレーターを使用した IP サブシステムの設計』(UG994) [参照 1] を参照してください。

デザインには、ザイリンクスのソフトウェア開発キット (SDK) で構築されたソフトウェアも含まれます。このソフトウェアは、MicroBlaze™ プロセッサ サブシステム上で実行され、制御、ステータス、モニターの各機能をインプリメントします。このアプリケーション ノートには、Vivado Design Suite と SDK の完全なプロジェクト ファイルを含むデザイン ファイルが付属しており、次のリンクからダウンロードできます。

- デュアル XIP モード
- クワッド XIP モード

これらをデザインの詳しい検証や再構築に活用したり、新規デザインのテンプレートとして使用することが可能です。

推奨される設計知識

このアプリケーション ノートは、ユーザーがザイリンクス Vivado Design Suite に関する一般的知識を持っていることを前提としています。Vivado Design Suite の詳細は、『Vivado Design Suite クイック リファレンス ガイド』(UG975) [参照 2] を参照してください。EDK の詳細は、『EDK コンセプト、ツール、テクニック : 効率的なエンベデッド システム構築をサポートするハンディガイド』(UG683) [参照 3] を参照してください。

はじめに

AXI Quad SPI IP コアは、レガシー、エンハンスト、そして XIP モードをサポートするように機能が拡張されています。これら 3 つのモードはさらに、スタンダード、デュアル、クワッドという 3 つの SPI モードに分類されます。スタンダード モードのコマンドは 1 本のライン (IO1) を使用してデータを交換し、デュアル モードでは 2 本のライン (IO0、IO1) を使用します。クワッド モードでは 4 本のライン インターフェイス (IO0、IO1、IO2、IO3) を用いてデータを交換します。レガシー モードは、以前のバージョン (v2.0) のコアをベースとするアプリケーションをサポートします。エンハンスト モードはメモリ マップ方式の AXI4 インターフェイスをサポートし、送信および受信 FIFO での固定長バースト機能もサポートします。エンハンスト モードは、DTR または DRR FIFO のフィルまたは読み出しに必要な AXI インターフェイス時間を削減します。これらの FIFO はコンパイル時に設定を変更でき、深さは 16 または 256 が可能です。

XIP モードは新しい機能で、フラッシュからブートするタイプのアプリケーションで使用します。このモードでは、従来のように DDR/SRAM などのメモリからではなく、シリアル SPI フラッシュからコードを実行できます。このアプリケーション ノートでは、ザイリンクス AXI Quad SPI IP コアの動作モードの 1 つを使用した XIP 機能について実例を示しながら説明します。

表 1 に、サポートされる各モードで使用する AXI4 インターフェイスをまとめます。

表 1 : AXI Quad SPI の各コンフィギュレーション モードで使用する AXI4 インターフェイス

モード	AXI4-Lite インターフェイス	AXI4 Full インターフェイス
レガシー モード	Yes	–
エンハンスト モード	–	Yes
XIP モード	Yes	Yes

使用する SPI スレーブの種類に応じて、コアの SPI モードはさらに 3 つに分類されます。表 2 に、動作モードとサポートされる SPI クロック周波数および I/O インターフェイスを示します。

表 2 : SPI モードと SCK 分周比および I/O インターフェイス

SPI モード	SPI クロック分周比	I/O インターフェイス (CS と SCK は常に存在)
スタンダード	2、4、8、16xn (ただし n = 1 … 128)	IO0、IO1
デュアル	2	IO0、IO1
クワッド	2	IO0、IO1、IO2、IO3

SPI 側の帯域幅を最大にするには、4 本のラインすべてでデータ トランザクションが行われるクワッド モードでコアを使用することを推奨します。クワッド モードでは Fast Read Quad I/O (0xEB h) コマンドがサポートされ、すべての I/O ラインで SPI フラッシュを読み出すことができるため、SPI の帯域幅が最大限に利用できます。

XIP の基本知識

XIP モードでは、実行可能コードを SPI フラッシュ メモリに格納します。FPGA ベースのシステムでは、コンフィギュレーション可能なビットストリームが SPI フラッシュの異なる領域に連続したメモリ アドレスで格納されます。FPGA は、POR (パワー オン リセット) 後にコンフィギュレーション可能なビットストリームを読み込みます。完全なビットストリームが FPGA にダウンロードされ、システムがアクティブ ステートに移行すると、ブートループ プログラムの実行が開始します。ローカル ブロック RAM メモリに小さなブートループ コードが格納され、このコードの働きによってプロセッサは SPI フラッシュ メモリの別の領域へジャンプして次の実行へと移ります。

この初期化中は書き込み可能メモリは使用できず、すべての計算はプロセッサのレジスタで実行する必要があります。このため、第 1 段階のブートローダーはアセンブラー言語で記述し、次のプログラムを通常実行する環境を提供するために必要な最小限の処理を行うのが一般的です。SPI フラッシュ メモリはバイト幅でアドレス指定されます。ELF (Executable Link Format) ファイルは連続したメモリ領域に格納されるため、プロセッサは SPI フラッシュを連続的に読み出します。ブートループ プログラムによって SPI フラッシュから完全な ELF ファイルを読み出して外部メモリ (DDR またはローカル オンチップ メモリ) に格納した後、プロセッサが実行を開始します。エンベデッド システム RTOS のほとんどはデマンド ページングをサポートしていないため、利用できるのは完全なシャドウ メモリまたは XIP メカニズムに限られます。コード実行方法は、XIP、完全シャドウ メモリ、デマンド ページング、バランス XIP の各モードにさらに分類されます。また、プログラムはフラッシュから直接実行することもできます。

このアプリケーション ノートでは、このモードを「ストア アンド ダウンロード」と分類します。これは、実行可能コードを不揮発性メモリ (NVM) に格納し、ブートアップ時に RAM にコピーするメモリ システムのことをいいます。ブートアップ時に、(多くの場合マイクロプロセッサ内の) NVM に格納された小さなコードを実行し、NVM メモリ (NAND または NOR) の内容を RAM にコピーしてその RAM の位置へジャンプします。これ以降、コードおよびデータは RAM から実行されます。

XIP モードでの AXI Quad SPI コアの使用

このコアは、XIP モードでは完全に読み出し専用モードで動作します。このコアには 2 つの AXI インターフェイスがあり、ローカル レジスタの設定には AXI4-Lite を使用し、メモリ アクセスにはメモリ マップ方式の AXI4 を使用します。プロセッサから見ると、AXI Quad SPI のメモリ マップされたアドレス範囲は通常のメモリと同じであるため、プロセッサはデータのアドレス、長さ、サイズを指定するだけでメモリからデータを読み出すことができます。SPI フラッシュには、FPGA がビットストリームでコンフィギュレーションされる前に実行可能コードを読み込んでおく必要があります。

このモードの使用法には、次の 2 つがあります。

- コンフィギュレーション ビットストリームと実行ファイルの両方を SPI フラッシュに格納する。
- 実行ファイルのみを SPI フラッシュに格納し、コアは iMPACT ツールでコンフィギュレーションする。

1 番目の方法では、SPI フラッシュ メモリのアドレス範囲をビットストリーム用と実行ファイル用の 2 つのセクションに分割します。通常、FPGA コンフィギュレーション ファイルは先頭アドレス 0x000000 からの位置に格納されます (SPI フラッシュが 24 ビット アドレスのメモリの場合)。これは、SPI フラッシュからブートする場合に FPGA によって与えられるデフォルトのアドレスです。FPGA のコンフィギュレーションが完了したら、実行ファイルが格納されている SPI メモリの位置へプロセッサがジャンプするようにソフトウェアを設定しておく必要があります。ブート コードは、フラッシュに格納するコンフィギュレーション ビットストリームに含めておきます。フラッシュから FPGA にコンフィギュレーション メモリが読み込まれると、プロセッサはオンチップ ブロック RAM に格納されたブート コードを実行します。ブート コードもコンフィギュレーション ビットストリームの一部に含まれます。このブート コードには、フラッシュ内の ELF ファイルの位置に関する情報が含まれます。プロセッサは SPI フラッシュ内のこの位置へジャンプして次に実行するコマンドを取得します。

プロセッサがコードを実行する方法には 2 つあります。1 つは、プロセッサ ブート コードが SPI フラッシュの ELF 領域全体を読み出し、データをローカル DDR メモリにコピーしてから実行を開始する方法

です。もう 1 つは、プロセッサがオンチップ ブロック RAM をスクラッチ パッド メモリとして使用しながら、メイン コードをフラッシュから直接実行する方法です。フラッシュへのアクセスは低速 (50MHz 未満) であるため、この方法はシステムの動作速度に対する要求がそれほど高くない場合に適しています。

2 番目の方法では、実行ファイルのみを SPI フラッシュに格納し、FPGA は iMPACT ツールでコンフィギュレーションします。MicroBlaze™ ブートループ アプリケーションは、プロセッサが実行ファイルの格納されているメモリ位置へジャンプするように記述する必要があります。実行ファイルは連続フォーマットまたはページ フォーマットで格納されます。SPI フラッシュはバイト単位でアクセスできるため、実行可能コードの格納には連続アドレス フォーマットの方が適しています。図 1 に、SPI フラッシュ メモリの構造例を示します。アドレスは必要に応じて変更できます。

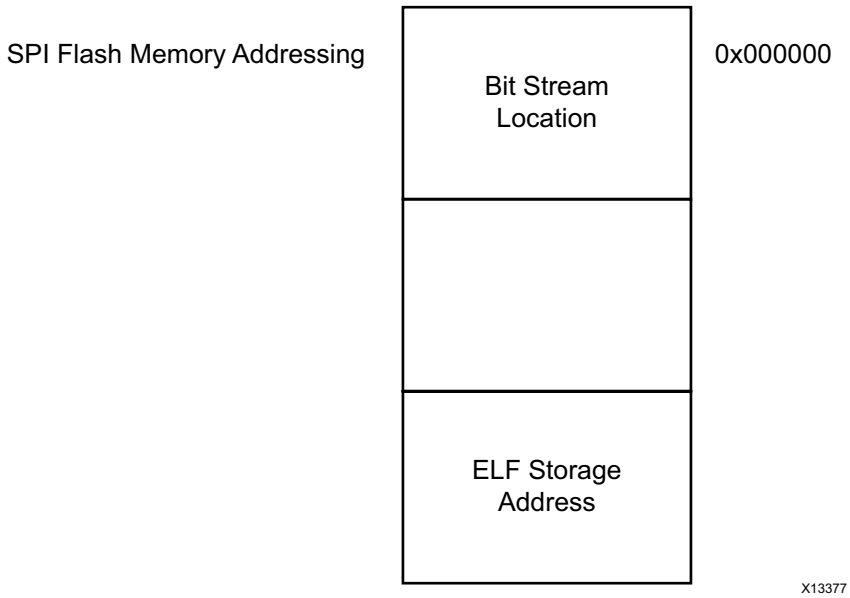


図 1 : SPI フラッシュ メモリのアドレス空間

表 3 に、さまざまな設定に使用する AXI Quad SPI コアのパラメーターを示します。

表 3 : コアのパラメーター

機能/説明	コアの設定		XIP テストで必要な設定
	コアのパラメーター	XGUI パラメーター	
AXI4 Memory Mapped ベース アドレス	C_S_AXI4_BASEADDR	これらのパラメーターの設定はシステム構成によります	キャッシュ可能アドレス
AXI4 Memory Mapped 上位アドレス	C_S_AXI4_HIGHADDR		キャッシュ可能アドレス
AXI4-Lite ベース アドレス	C_BASEADDR		AXI Lite アドレス
AXI4-Lite 上位アドレス	C_HIGHADDR		AXI Lite アドレス
AXI インターフェイスの選択	C_TYPE_OF_AXI4_INTERFACE	[Enable Performance Mode]	1
XIP モードかどうかの選択	C_XIP_MODE	[Enable XIP Mode]	1
SPI クロック周波数比	C_SCK_RATIO	[SPI Options] の [Frequency Ratio]	2 (デフォルト)

表 3 : コアのパラメーター (続き)

機能/説明	コアの設定		XIP テストで必要な設定
	コアのパラメーター	XGUI パラメーター	
SPI モード	C_SPI_MODE	[SPI Options] の [Modes] 0 = Standard 1 = Dual 2 = Quad	2
SPI スレーブとして使用する SPI メモリ デバイス	C_SPI_MEMORY 1= Winbond 2 = Numonyx	[SPI Options] の [Slave Device]	2

ハードウェア要件

このシステムには、次のハードウェア ボードが必要です。

- ザイリンクス KC705 評価ボード (Rev. C、D、1.0、または 1.1)

このリファレンス システムを構築してダウンロードするには、次のソフトウェア デザイン ツールが必要です。

- Vivado 2013.2
- SDK 2013.2

システム図

図 2 に、AXI Quad SPI コアを XIP モードで使用するシステムを示します。

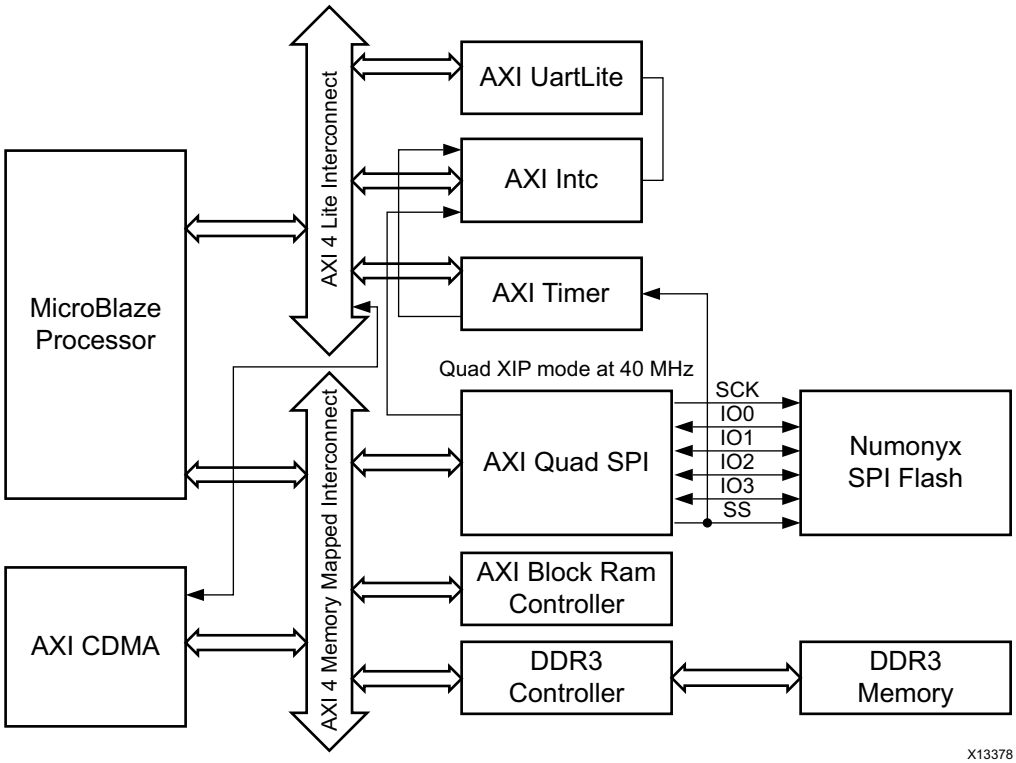


図 2 : XIP モードの代表的なシステム

システムの詳細

このシステムは、AXI インターフェイスをベースにしています。AXI インターフェイスは、AMBA® (Advanced Micro-controller Bus Architecture) 規格に基づいて標準化された IP インターフェイス プロトコルです。リファレンス デザインで使用している AXI インターフェイスは、メモリ マップ方式の AXI4 および AXI4-Lite インターフェイスで構成されています。クロック ジェネレーターとプロセッサ システムのリセットブロックは、システム全体にクロックおよびリセットを供給します。I/O ペリフェラルおよびプロセッサ サポート IP を含むシステムの高度な制御は、エンベデッド MicroBlaze プロセッサが担います。

性能とエリアのバランスをとるようにシステムを最適化するには、複数の AXI インターコネクト ブロックを使用し、AXI インターコネクト ブロックを個別に調整および最適化して、セグメント化/階層化された AXI インターコネクト ネットワークをインプリメントします。表 4 に、XIP モードのデモ システムで使用しているコアとバージョン、アドレスを示します。

表 4 : デモ システムで使用しているコアとアドレス

IP コア	バージョン	ベース アドレス	上位アドレス
MicroBlaze	9.0	N/A	N/A
MIG 7 Series	2.0	0x80000000	0xBFFFFFFF
AXI Quad SPI (AXI4 Full)	3.0	0xC4000000	0xC4FFFFFF
AXI Quad SPI (AXI4 Lite)	3.0	0x44A10000	0x44A1FFFF
AXI BRAM Controller	3.0	0xC0000000	0xC001FFFF
AXI UARTLITE	2.0	0x40600000	0x4060FFFF
LMB BRAM IF Controller	4.0	0x00000000	0x0000FFFF
AXI Interrupt Controller (INTC)	3.0	0x41200000	0x4120FFFF
AXI BRAM Controller	3.0	0xC2000000	0xC201FFFF
AXI Timer	2.0	0x41C00000	0x41C0FFFF
Processor System Reset	5.0	N/A	N/A
Clock Generator	5.0	N/A	N/A
AXI Interconnect - Lite	2.0	N/A	N/A
AXI Interconnect - Full	2.0	N/A	N/A
AXI CDMA	4.0	0x44A00000	0x44A0FFFF
UTIL Reduced Logic	1.0	N/A	N/A

Vivado IP インテグレーターでの IP コアの設定

Vivado IP インテグレーターでは、目的のコアをシステムに追加した後、XGUI 画面で設定オプションを変更できます。次に、これらオプションの設定画面のスクリーンショットを示し、各オプションについて説明します。

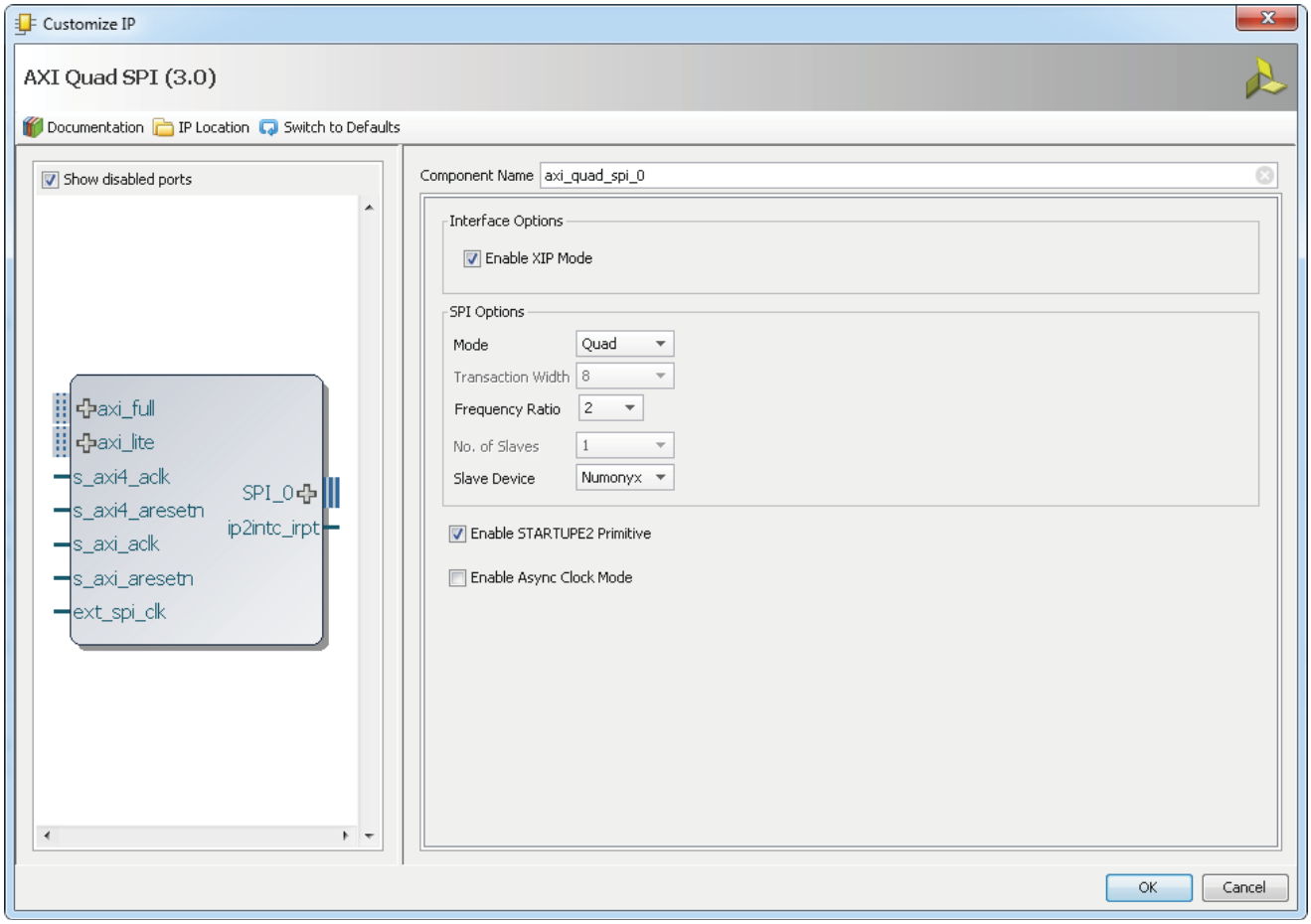


図 3 : Vivado IP インテグレーターの IP コア設定画面

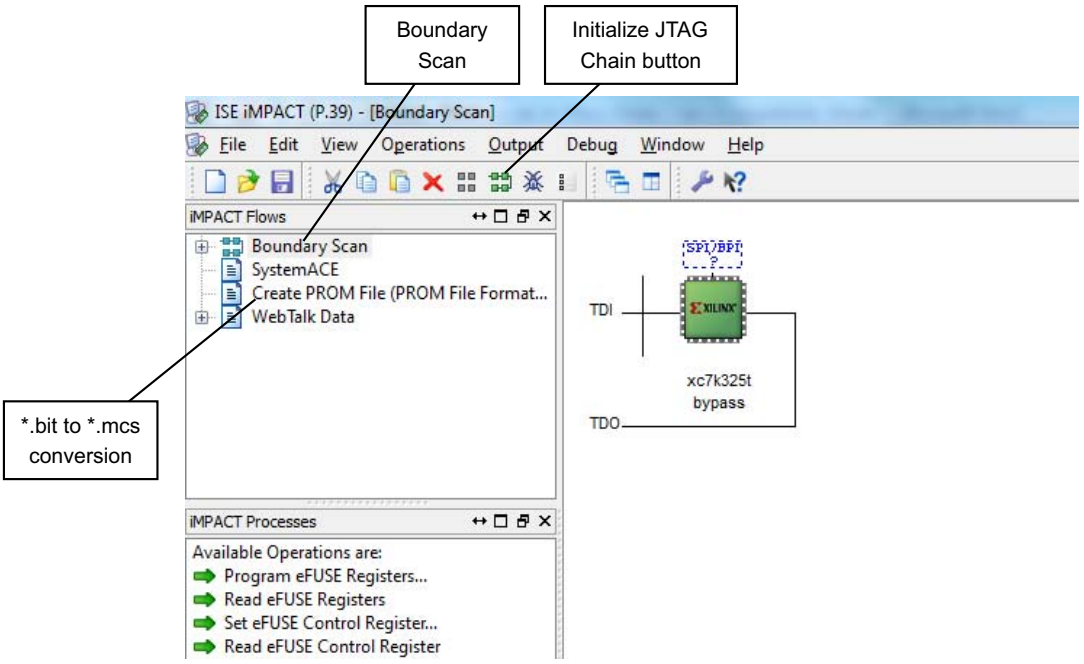
iMPACT および SPI フラッシュ プログラム

ザイリンクス デバイスは SPI フラッシュからブートできます。ブートの前に、SPI フラッシュにビットストリームおよびブートループ コードを書き込んでおく必要があります。そのためには、`top.bit` ファイル (ハードウェア情報) とブートループ ファイル (ソフトウェア情報) を 1 つの `download.bit` ファイルにマージします。iMPACT ツールには、BIT ファイルを MCS フォーマットのファイルに変換する機能があります。変換方法は、「ビットストリームから MCS ファイルへの変換」で説明します。iMPACT はこの MCS ファイルを使用してダウンストリームの SPI フラッシュ デバイスをプログラムします。

ビットストリームから MCS ファイルへの変換

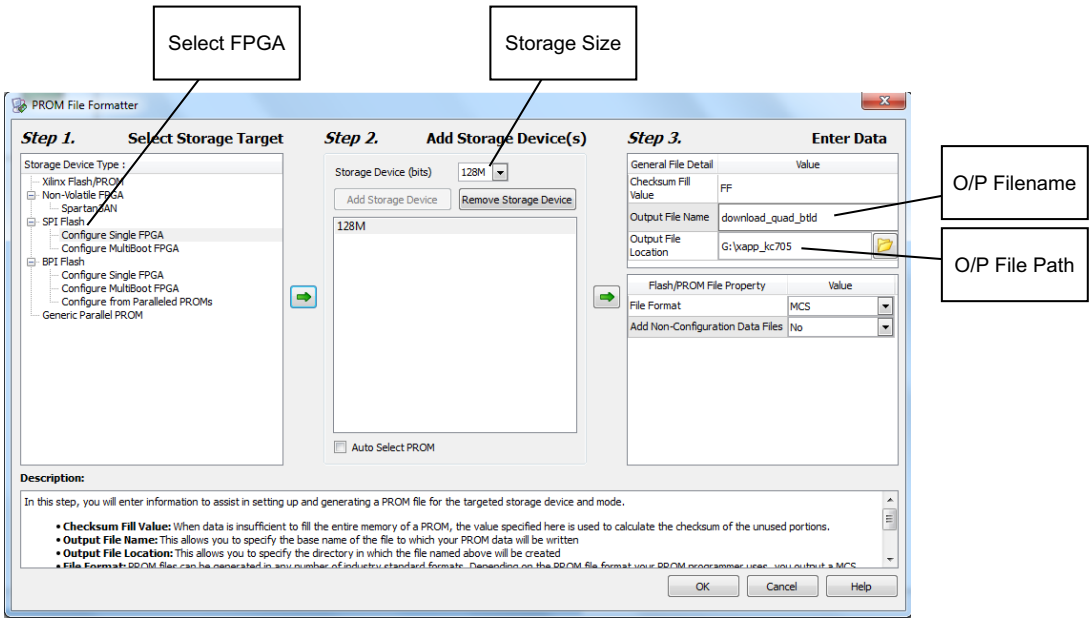
ここでは、ビットストリームを MCS ファイルに変換し、2 つの MCS ファイルをマージして SPI フラッシュへダウンロードする手順について説明します。

1. iMPACT ツールを起動し、バウンダリ スキャンを実行して JTAG チェーンを初期化し、デバイスを選択します。



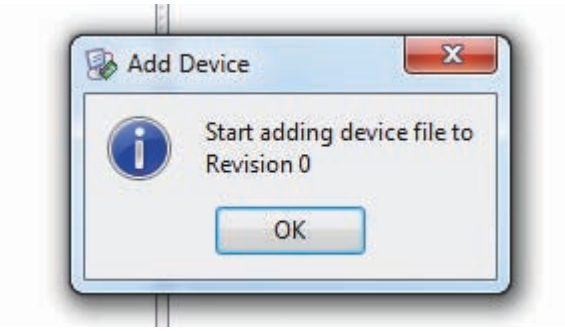
X13379

2. [Create PROM File] をダブルクリックします。
   a. [Configure Single FPGA] をクリックし、[Step 1.] と [Step 2.] の間の矢印をクリックします。
   b. [Add Storage Device(s)] でストレージ デバイスの容量を設定し、[Step 2.] と [Step 3.] の間の矢印をクリックします。
   c. ストレージ デバイスの容量は、そのデバイスのデータシートに記載されています。
   d. [Output File Name] に出力ファイル名を入力し、[Output File Location] に `application.elf` ファイルの場所を選択して [OK] をクリックします。

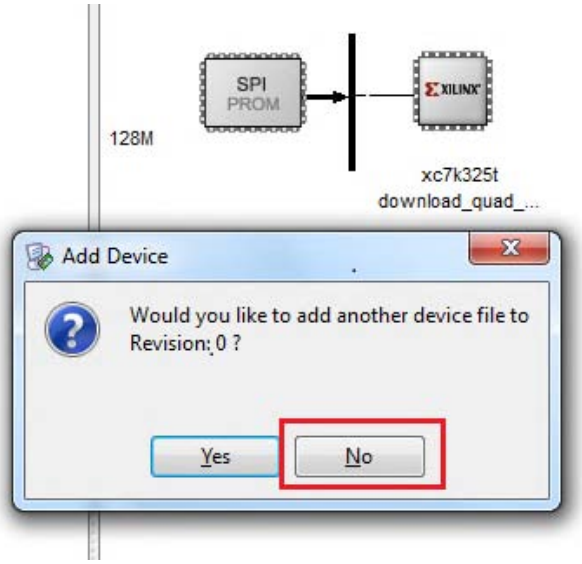


X13380

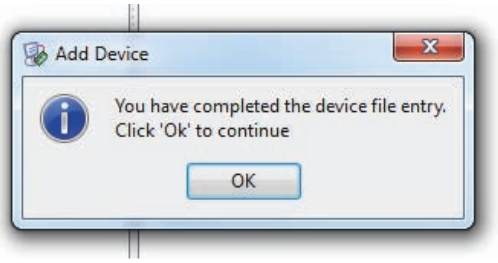
3. 次のダイアログ ボックスで [OK] をクリックし、変換するビットファイルを選択します。



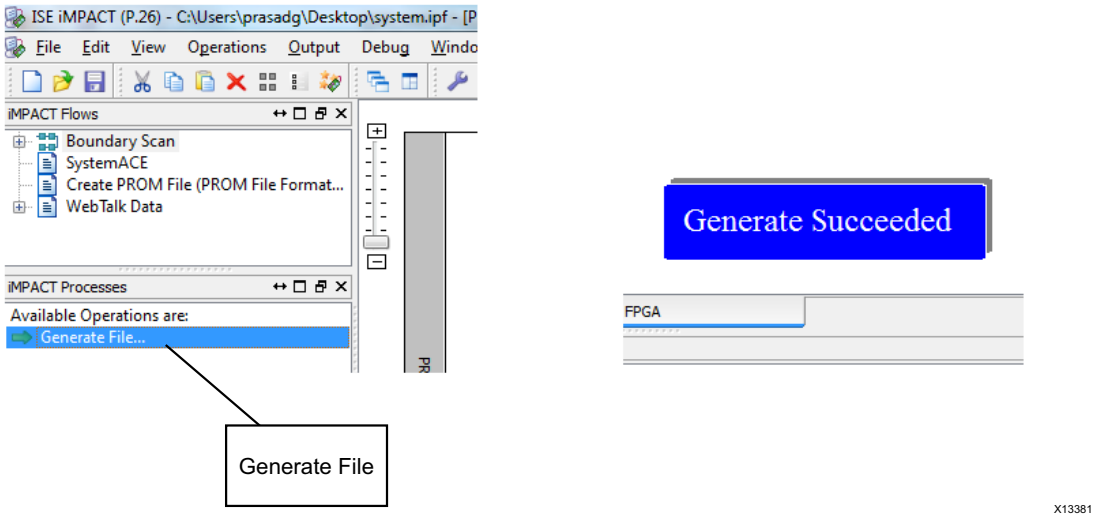
4. iMPACT でビットファイルが追加されます。次のダイアログ ボックスで [No] をクリックして完了します。



5. 次のメッセージが表示されます。



6. [OK] をクリックして次へ進み、[Generate File] をダブルクリックして MCS ファイルを生成します。



7. MCS ファイルの生成が完了すると、[Generate Succeeded] というメッセージが表示されます。

2 つの MCS ファイルのマージ

iMPACT で SPI フラッシュをプログラムするには、MCS フォーマットのファイルを使用する必要があります。ELF ファイルを MCS ファイルに変換するユーティリティがいくつかあります。ザイリンクスも MCS ユーティリティを提供しています。まず、こちらのウェブ ページから `xapp1053.zip` をダウンロードしてください。次にこの zip ファイルを展開し、FLASH_BURN フォルダーを開きます。このフォルダーには、Windows 用のフラッシュ プログラミング ユーティリティ XIP.exe と XMCSUTIL.exe が含まれます。これらを、`application.elf` ファイルと同じフォルダーにコピーしてください。

これらのユーティリティ以外に、このアプリケーション ノート XAPP1176 の zip ファイルにはバッチ ファイル `bit_and_appl_to_mcs.bat` が含まれます。このファイルを、`application.elf` ファイルと同じフォルダーにコピーしてください。ELF ファイルを MCS フォーマットに変換するには、次の手順を実行します。

1. `bitfile.mcs`、`application.elf`、バッチ ファイル、XIP.exe、XMCSUTIL.exe が同じフォルダーにあることを確認します。
2. XMD プロンプトで次のコマンドを実行して `application.elf` ファイルをバイナリに変換します。

   ```
   mb-objcopy -O binary -R .vectors.reset -R .vectors.sw_exception -R
   .vectors.interrupt -R .vectors.debug_sw_break -R .vectors.hw_exception
   <application_name.elf> <application_name.b>
   ```

   次に例を示します。

```
XMD%
XMD% mb-objcopy -O binary -R .vectors.reset -R .vectors.sw_exception -R
.vectors.interrupt -R .vectors.debug_sw_break -R .vectors.hw_exception
xip_led_app.elf xip_led_app.b
XMD%
```

   注記：このコマンドで -R .vectors.reset -R .vectors.sw_exception -R .vectors.interrupt -R .vectors.debug_sw_break -R .vectors.hw_exception を使用しないと、バイナリ ファイルのサイズが大きくなりすぎます。

3. 上記のコマンドでバイナリ ファイルが生成されます。このバイナリ ファイルを、次のコマンドを実行して MCS ファイルに変換します。

   ```
   XMCSUTIL -accept_notice -i <application_name.b> -o <application_name.mcs> -29
   ```

   このコマンドの実行例を示します。

```
XMD%
XMD% XMCSUTL -accept_notice -i xip_led_app.b -o xip_led_app.mcs -29

=> Checking filenames
     - input/output/log filenames valid
xmcsutil<tm> Version 1.24
Copyright (c) 2001-2007 Xilinx, Inc. All rights reserved
Xilinx MCS/HEX Data Processing Utility
**********************************************************************
**//==================================================================\\**
**||  NOTICE:FOR XILINX PROTOTYPE USE ONLY                            ||**
**||                                                                  ||**
**||  SOFTWARE PROVIDED “AS IS”.ALL WARRANTIES, EXPRESS OR IMPLIED,   ||**
**||  ARE HEREBY DISCLAIMED.SOFTWARE NOT AUTHORIZED FOR USE IN        ||**
**||  PRODUCTION ENVIRONMENTS OR FOR USE IN OR WITH LIFE-SUPPORT OR   ||**
**||  MISSION-CRITICAL APPLIANCES, SYSTEMS, OR DEVICES.               ||**
**||                                                                  ||**
**||  This software is for use with Xilinx devices only.Please send   ||**
**||  all technical questions and comments to:                        ||**
**||                                                                  ||**
**||  xspi@xilinx.com                                                 ||**
**||                                                                  ||**
**\\==================================================================//**
**********************************************************************
===[Program notice/license accepted via -accept_notice command option]==
**********************************************************************
Mon May 20 17:10:53 2013
Converting bin file [xip_led_app.b] to MCS format [xip_led_app.mcs]
Converted [7900] bytes

- max MCS byte addr [0x00001EDB] in file [xip_led_app.mcs]

Elapsed clock time = 0 seconds
XMD%
```

4. これで、application.mcs と bitfile.mcs の 2 つのファイルが作成されます。次のコマンドを実行して、これら 2 つの MCS ファイルを 1 つの MCS ファイルにマージします。

   ```
   XMCSUTIL -accept_notice -i <system_bitstream.mcs> <application_name.mcs>
   -o <combined.mcs> -16 -segaddr 0x00 <start of SPI flash address where elf
   to be stored> -usedataaddr -padff
   ```

   次に例を示します。

   ```
   XMCSUTIL -accept_notice -i download_btld_quad.mcs xip_led_app.mcs -o
   combined.mcs -16 -segaddr 0x00 0xC00000 -usedataaddr -padff
   ```

   この例では、xip_led_app.mcs ファイルは SPI フラッシュのアドレス 0xC00000 に格納されます。格納先のアドレスは必要に応じて変更できます。SPI アドレスの設定方法の詳細は、アプリケーション ノート『Spartan®-3A DSP 1800A スターター プラットフォームで SPI を利用したフラッシュ メモリからのブートロード』[参照 5] を参照してください。

5. ここまでの手順は、このアプリケーション ノートの zip ファイルに含まれるバッチ ファイルでも実行できます。バッチ ファイルとして実行すると、コマンド実行中のエラーを防ぐことができます。

   バッチ ファイル内のビットファイル名、アプリケーション名、SPI フラッシュ内の ELF ファイル格納場所の開始アドレスを編集し、XMD プロンプトで次のコマンドを実行します。

   ```
   bit_and_appl_to_mcs.bat
   ```

   これによって、combined.mcs ファイルが生成されます。

```
XMD%
XMD% XMCSUTL -accept_notice -i download_btld_quad.mcs xip_led_app.mcs -o
combined_quad.mcs -16 -segaddr 0x00 0xC00000 -usedataaddr -padff

=> Checking filenames
     - input/output/log filenames valid
xmcsutil<tm> Version 1.24
Copyright (c) 2001-2007 Xilinx, Inc. All rights reserved
Xilinx MCS/HEX Data Processing Utility
**********************************************************************
**//==================================================================\\**
**|| NOTICE:FOR XILINX PROTOTYPE USE ONLY                           ||**
**||                                                                ||**
**||  SOFTWARE PROVIDED “AS IS”.ALL WARRANTIES, EXPRESS OR IMPLIED,||**
**||  ARE HEREBY DISCLAIMED.SOFTWARE NOT AUTHORIZED FOR USE IN     ||**
**||  PRODUCTION ENVIRONMENTS OR FOR USE IN OR WITH LIFE-SUPPORT OR ||**
**||  MISSION-CRITICAL APPLIANCES, SYSTEMS, OR DEVICES.             ||**
**||                                                                ||**
**||  This software is for use with Xilinx devices only.Please send||**
**||  all technical questions and comments to:                      ||**
**||                                                                ||**
**||  xspi@xilinx.com                                               ||**
**||                                                                ||**
**\\==================================================================//**
**********************************************************************
===[Program notice/license accepted via -accept_notice command option]===
**********************************************************************
Mon May 20 17:11:06 2013

Combining [2] MCS files
-[2] MCS files will be combined into a single MCS file [combined_quad.mcs]
-> adding bytes from MCS files [download_btld_quad.mcs]; relocating to addr
[0x0]

    =>added [download_btld_quad.mcs] = 11443612 bytes | 91548896 bits
-> adding bytes from MCS file [xip_led_app.mcs]; relocating to addr [0xC00000]
   * added [1139300] 0xFF bytes at addr range [0x00AE9D9C :0x00BFFFFF]
   => added [    xip_led_app.mcs] = 1147200 bytes | 9177600 bits

=> Total bytes written = [12590812 | 100726496 bits]
   - to output file [combined_quad.mcs]
   - max hex byte addr [0x00C01EDC]

Elapsed clock time = 183 seconds
XMD%
```

6. MCS ファイルのバイト数を調べるために、次のコマンドを実行します。

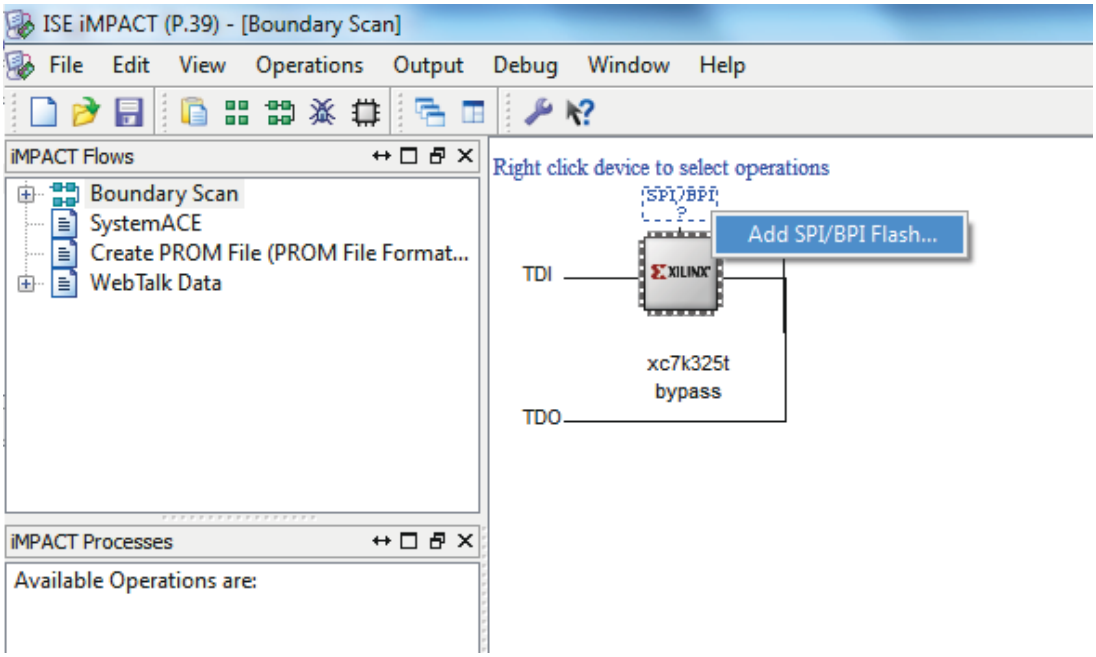
   ```
   XMCSUTIL -bytecount -i <application_name.mcs> -o count.log
   ```

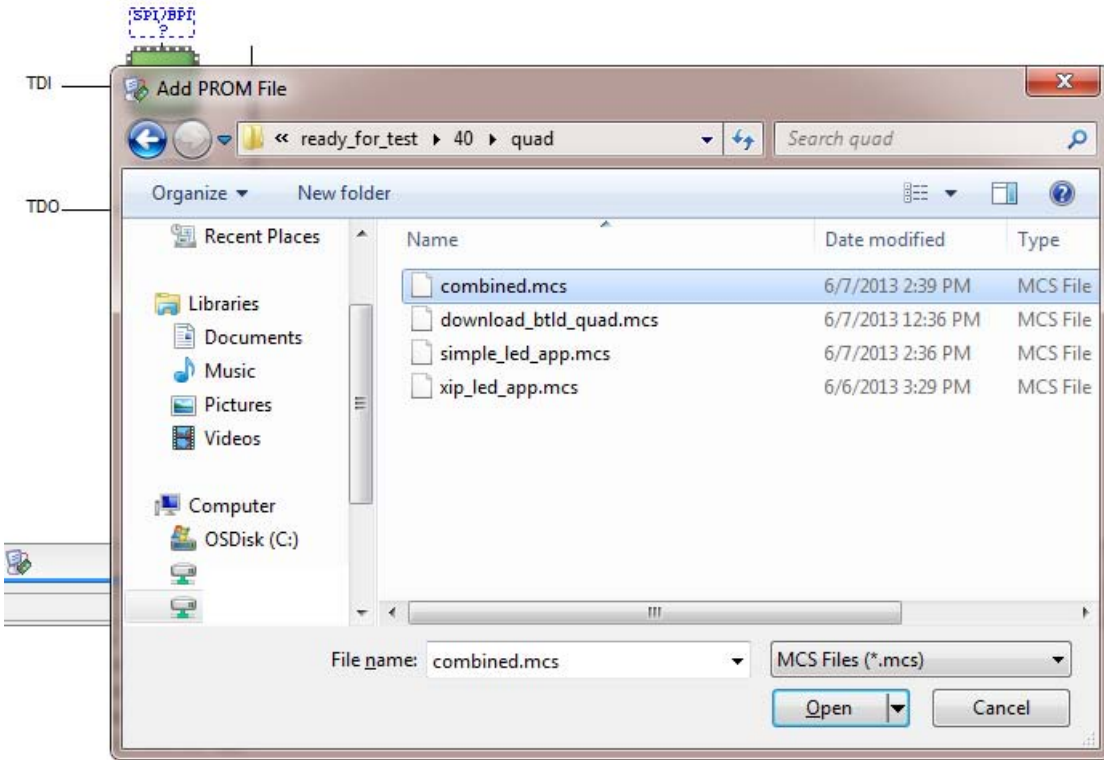
このファイルのバイト数は、count.log に記録されます。このコマンドは、application.elf (xip_led_app.elf) と 2 つのユーティリティが格納されているディレクトリから実行します。

```
XMD%
XMD% XMCSUTL -byte_count -i xip_led_app.mcs -o count.log

=> Checking filenames
     - input/output/log filenames valid
xmcsutil<tm> Version 1.24
Copyright (c) 2001-2007 Xilinx, Inc. All rights reserved
Xilinx MCS/HEX Data Processing Utility
**********************************************************************
**//==================================================================\\**
**|| NOTICE:FOR XILINX PROTOTYPE USE ONLY                           ||**
**||                                                                ||**
**||  SOFTWARE PROVIDED “AS IS”.ALL WARRANTIES, EXPRESS OR IMPLIED,||**
**||  ARE HEREBY DISCLAIMED.SOFTWARE NOT AUTHORIZED FOR USE IN     ||**
**||  PRODUCTION ENVIRONMENTS OR FOR USE IN OR WITH LIFE-SUPPORT OR ||**
**||  MISSION-CRITICAL APPLIANCES, SYSTEMS, OR DEVICES.             ||**
**||                                                                ||**
**||  This software is for use with Xilinx devices only.Please send||**
**||  all technical questions and comments to:                      ||**
**||                                                                ||**
**||  xspi@xilinx.com                                               ||**
**||                                                                ||**
**\\==================================================================//**
**********************************************************************
==> Use “-accept_notice” option to accept notice automatically
-++< Press ENTER to accecpt notice and continue >==-
**********************************************************************
Mon May 20 17:30:18 2013
==> MCS file 1 [xip_led_app.mcs] has [7900] bytes <63200 bits.

Elapsed clock time = 0 seconds
XMD%
```

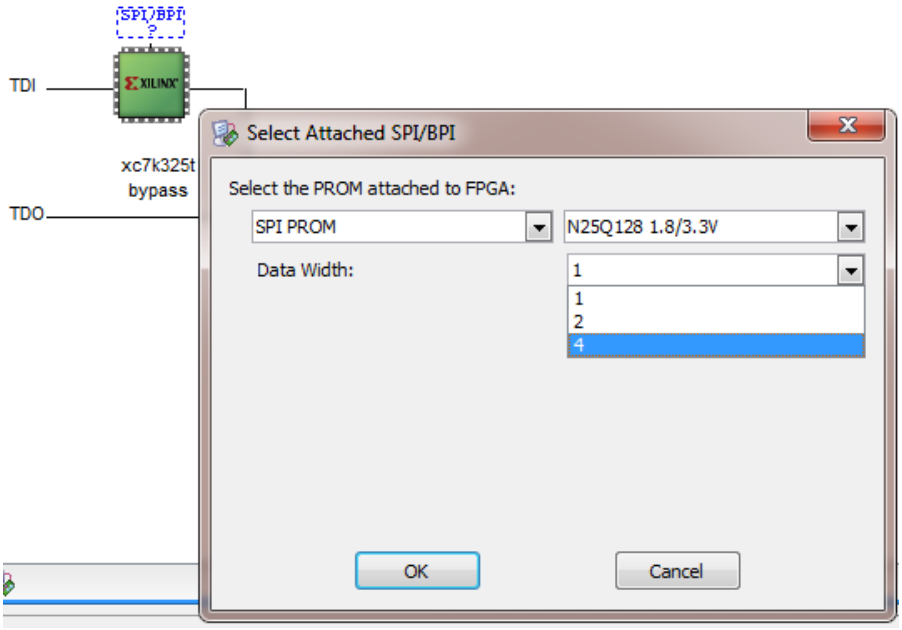
MCS ファイルによるフラッシュ デバイスのプログラムと実行

1. iMPACT ツールで [Boundary Scan] ウィンドウを選択します。[SPI/BPI] を右クリックします。[Add SPI/BPI Flash] をクリックし、プログラムするファイルを選択します。



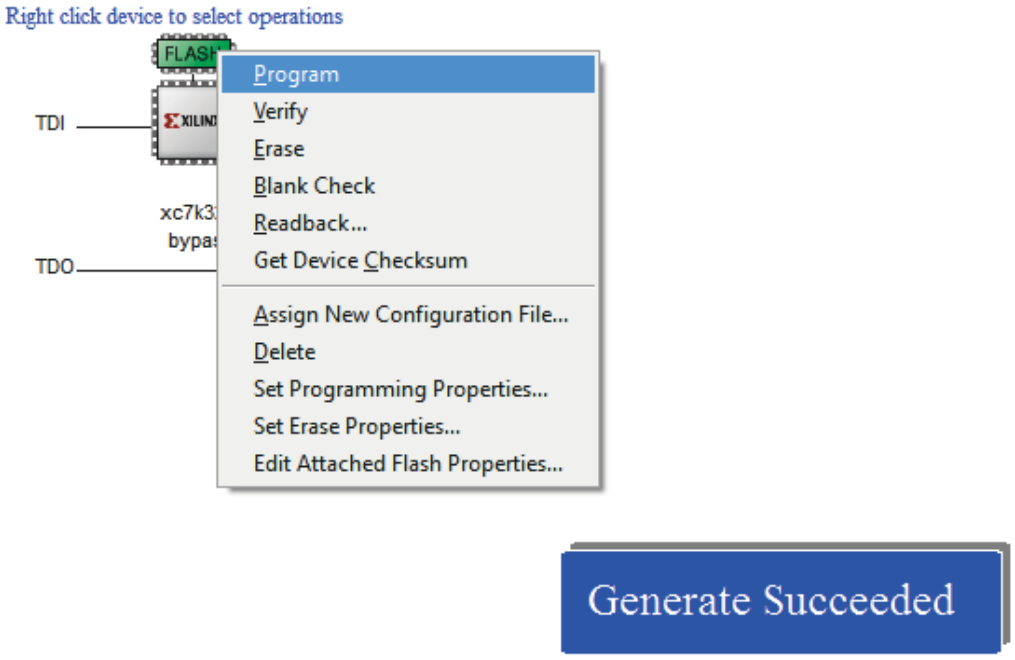


2. 次に表示されるダイアログ ボックスでフラッシュ デバイスを正しく選択し、フラッシュ デバイスがクワッド モードをサポートしている場合は [Data Width] を 4 (クワッド モード) に設定します。[OK] をクリックするとフラッシュが選択されます (XIP モードの場合、[Data Width] の値はクワッド モードなら 4、デュアル モードなら 2、スタンダード モードなら 1 を選択)。



3. [FLASH] を右クリックします。[Program] をクリックします。プログラムが完了すると、「Generate Succeeded」というメッセージが表示されます。

注記：iMPACT による SPI フラッシュのプログラムには 20 分ほどかかります。



4. プログラムが完了したら、いったんボードの電源を切って入れ直します。これで FPGA は SPI フラッシュによってコンフィギュレーションされ、ブートローダーが実行されます。ブートローダーは SPI フラッシュから DDR メモリへ `application.elf` をコピーし、プログラム カウンターには DDR メモリのアドレスがセットされます。
5. ハイパーターミナルを起動します。FPGA のコンフィギュレーション前に起動しておかないと、出力が表示されません。
6. アプリケーションが実行されます。次に、出力例を示します。

```
Booting from SPI Flash in XIP DUAL IO Mode on Jun 7 2013 at 13:41:55

Axi Quad SPI Version:V3.0
Board used          :KC705 Rev 1.0
FPGA Device used    : xc7k325tffg900-2
Tool Used           :Vivado IPI
Tool Version        :2013.2

Clearing the DDR ...@ 0x80000000
Loading ELF from SPI Flash (0xC4C00000) to DDR (0x80000000)

************************************************************************
* Hello World ...Using DUAL IO Mode (‘,’)                               *
* Booted from DDR ...                                                   *
* Time taken for                                                        *
* Appl loading from SPI Flash to DDR ->                  837 us *
* Processor starts execution from DDR ->                463404 us *
* Throughput Measured (Appl Size)/(Appl load time) :      9.9 MB/s *
************************************************************************

Starting LED Test
LEDs glow for 5 times

Memory RD/WR test @ 0x80001F6C
Memory Test PASSED

Execution is Finished
```

7. ボードをリセットせずにブート アプリケーションを実行するには、ボードの CPU RST プッシュボタンを押します。これで同じアプリケーションがもう一度実行され、同じ結果が得られます。

注記：サンプル テストの出力で、「Appl loading from SPI Flash to DDR」にはチップ セレクトがアクティブのときに SPI フラッシュから DDR メモリへアプリケーションを読み出すのに実際にかかった時間が表示されます。「Processor starts execution from DDR」には、ブートローダーがアプリケーションを DDR メモリに読み込み、アプリケーションの実行を開始するのにかかった実際の時間が表示されます。「Throughput Measured」には、QSPI がフラッシュからアプリケーションを読み出す際の実際のスループット (単位 : MB/s) が表示されます。このアプリケーション ノートでは、アプリケーション サイズは約 8KB です。

提供されるハードウェア システム

このセクションでは、このリファレンス デザインとアプリケーション ノートでサポートされるハードウェア システムについて詳しく説明します。

XIP モードのテスト システム - クワッド モード

1. XIP モードのテスト システムでは、「iMPACT および SPI フラッシュ プログラム」で説明した手順に従って XIP モード用に設定が完了したアプリケーションがリファレンスとして提供されています。MCS ファイルをフォルダーから Numonyx 社製 SPI フラッシュへダウンロードし、SPI フラッシュから FPGA をブートします。

```
Booting from SPI Flash in XIP QUAD IO Mode on Jun 7 2013 at 12:32:28

Axi Quad SPI Version:V3.0
Board used          :KC705 Rev 1.0
FPGA Device used    : xc7k325tffg900-2
Tool Used           :Vivado IPI
Tool Version        :2013.2

Clearing the DDR ...@ 0x80000000
Loading ELF from SPI Flash (0xC4C00000) to DDR (0x80000000)

************************************************************************
* Hello World ...Using QUAD IO Mode (‘,’)                               *
* Booted from DDR ...                                                    *
* Time taken for                                                         *
* Appl loading from SPI Flash to DDR ->                    421 us *
* Processor starts execution from DDR ->                 462988 us *
* Throughput Measured (Appl Size)/(Appl load time) :      19.7 MB/s *
************************************************************************

Starting LED Test
LEDs glow for 5 times

Memory RD/WR test @ 0x80001F6C
Memory Test PASSED

Execution is Finished
```

2. `kc705_xip_quad_mode.zip` には、次のフォルダーが含まれます。
   - HW – 基本的なシステム ファイルが含まれます。
   - ready_for_download – MCS ファイルと `bit_and_appl_to_mcs.bat` ファイルが含まれます。`bit_and_appl_to_mcs.bat` ファイルを実行して、ビットストリームとアプリケーション ファイルを結合した MCS ファイルを生成します。

     MCS ファイルを結合するには、`download.mcs` および `<application_name>.elf` と同じディレクトリにこの `.bat` ファイルを置きます。

テストにそのまま使用可能な結合済みの MCS ファイル (boot_from_flash_quad_mode.mcs) も用意されています。iMPACT でこのファイルを用いて SPI フラッシュにプログラムします。

- SW – 各アプリケーションのファイルが含まれます。
  - ブート後、bootloader と xip_quad_app が実行されます。

XIP モードのテスト システム - デュアル モード

1. XIP モードのテスト システムでは、「iMPACT および SPI フラッシュ プログラム」で説明した手順に従って XIP モード用の設定が完了したアプリケーションがリファレンスとして提供されています。MCS ファイルをフォルダーから Numonyx 社製 SPI フラッシュへダウンロードし、SPI フラッシュから FPGA をブートします。

```
Booting from SPI Flash in XIP DUAL IO Mode on Jun 7 2013 at 13:41:55

Axi Quad SPI Version:V3.0
Board used          :KC705 Rev 1.0
FPGA Device used    : xc7k325tffg900-2
Tool Used           :Vivado IPI
Tool Version        :2013.2

Clearing the DDR ...@ 0x80000000
Loading ELF from SPI Flash (0xC4C00000) to DDR (0x80000000)

************************************************************************
* Hello World ...Using DUAL IO Mode (‘,’)                               *
* Booted from DDR ...                                                    *
* Time taken for                                                         *
* Appl loading from SPI Flash to DDR ->                    837 us *
* Processor starts execution from DDR ->                 463404 us *
* Throughput Measured (Appl Size)/(Appl load time) :      9.9 MB/s *
************************************************************************

Starting LED Test
LEDs glow for 5 times

Memory RD/WR test @ 0x80001F6C
Memory Test PASSED

Execution is Finished
```

2. kc705_xip_dual_mode.zip には、次のフォルダーが含まれます。
   - HW – 基本的なシステム ファイルが含まれます。
   - ready_for_download – MCS ファイルと bit_and_appl_to_mcs.bat ファイルが含まれます。bit_and_appl_to_mcs.bat ファイルを実行して、ビットストリームとアプリケーション ファイルを結合した MCS ファイルを生成します。

     MCS ファイルを結合するには、download.mcs および <application_name>.elf と同じディレクトリにこの .bat ファイルを置きます。

     テストにそのまま使用可能な結合済みの MCS ファイル (boot_from_flash_dual_mode.mcs) も用意されています。iMPACT でこのファイルを用いて SPI フラッシュにプログラムします
   - SW – 各アプリケーションのファイルが含まれます。
     - ブート後、bootloader と xip_dual_app が実行されます。

まとめ

このアプリケーション ノートでは、実行可能データを SPI フラッシュから DDR メモリへ読み込んでからプロセッサを実行する XIP モードについて説明してきました。AXI Quad SPI コアを XIP モードにコンフィギュレーションすると、この機能を利用できます。SPI モードにはいくつかの種類がありますが、常にクワッド モードのパフォーマンスが最も高くなります。このアプリケーション ノートは、AXI Quad SPI IP の使用法、およびブート可能アプリケーションを作成する際のメカニズムについて実例を交えて説明することを目的としています。

リファレンス デザインの詳細

表 5 に、このアプリケーション ノートに付属するリファレンス デザインの内容をまとめます。

表 5 : リファレンス デザインの内容

パラメーター	説明
全般	
開発者	Sanjay Kulkarni、Prasad Gutti
ターゲット デバイス (ステッピング レベル、ES、プロダクション、スピード グレード)	Kintex-7
ソース コードの提供	あり
ソース コードの形式	VHDL/Verilog (一部コアは暗号化済み)
シミュレーション	
既存のザイリンクス アプリケーション ノート/リファレンス デザイン、CORE Generator ツール、サードパーティからデザインへのコード/IP の使用	Vivado 用に提供されたリファレンス デザインおよび CORE Generator ツール シミュレーションで生成されるビデオ コア
論理シミュレーションの実施	N/A
タイミング シミュレーションの実施	N/A
論理シミュレーションおよびタイミング シミュレーションでのテストベンチの利用	N/A
テストベンチの形式	N/A
使用したシミュレータ/バージョン	N/A
SPICE/IBIS シミュレーションの実施	N/A
インプリメンテーション	
使用した合成ツール/バージョン	
使用したインプリメンテーション ツール/バージョン	Vivado Design Suite 2013.2
スタティック タイミング解析の実施	あり (PAR/TRCE のタイミングにパス)
ハードウェア検証	
ハードウェア検証の実施	あり
使用したハードウェア プラットフォーム	KC705 ボード

デバイスの使用リソースと性能

このセクションでは、このアプリケーション ノートで説明した 2 つのセットアップそれぞれにおけるデバイス リソース使用率を示します。サポートされるデバイスは、表 6 に示したとおりです。

表 6 : デバイスのリソース使用率

デバイス	スピード グレード グレード	パッケージ	スライス レジスタ	配置済みスライス	スライス LUT	I/O	RAMB36/ FIFO36	RAMB18/ FIFO18
xc7l235t	-2	ffg900	15965	7190	19065	130	90	0

XIP システムの使用率

表 7 に、各 IP コアのモジュール レベルの使用率を示します。

表 7 : モジュール レベルのリソース使用率

IP コア	インスタンス名	スライス	スライス レジスタ	LUT	LUT RAM	ブロック RAM	DSP スライス	BUFG CTRL	BUFR
axi_quad_spi	axi_quad_spi_1	1533	577	508	44	N/A	N/A	N/A	N/A
axi_intc	axi_intc_1	352	112	148	N/A	N/A	N/A	N/A	N/A
microblaze	microblaze_1	4467	1407	1700	198	11	3	N/A	N/A
mdm	mdm_1	125	76	31	04	N/A	N/A	1	N/A
mig_7series	mig_1	34380	10931	13183	2325	N/A	N/A	2	N/A
axi_bram_ctrl	axi_bram_ctrl_1	922	316	326	2	N/A	N/A	N/A	N/A
axi_bram_ctrl	axi_bram_ctrl_2	907	316	234	2	N/A	N/A	N/A	N/A
axi_cdma	axi_cdma_1	2361	1040	878	81	N/A	N/A	N/A	N/A
axi_timer	axi_timer_1	659	216	266	N/A	N/A	N/A	N/A	N/A
axi_gpio	axi_gpio_1	201	92	76	N/A	N/A	N/A	N/A	N/A
axi_uartlite	axi_uartlite_1	232	86	96	10	N/A	N/A	N/A	N/A
axi_interconnect_lite	axi_interconnect_1	567	120	231	N/A	N/A	N/A	N/A	N/A
axi_interconnect_full	axi_interconnect_2	2521	639	976	14	N/A	N/A	N/A	N/A
proc_sys_reset	proc_sys_reset_1	52	32	90	1	N/A	N/A	N/A	N/A

参考資料

1. 『Vivado Design Suite ユーザー ガイド : IP インテグレーターを使用した IP サブシステムの設計』(UG994)
2. 『Vivado Design Suite クイック リファレンス ガイド』(UG975)
3. EDK コンセプト、ツール、テクニック : 効率的なエンベデッド システム構築をサポートするハンディガイド (UG683)
4. 『LogiCORE IP AXI Quad Serial Peripheral Interface (SPI) 製品ガイド』(PG153)
5. 『Spartan-3A DSP 1800A スターター プラットフォームで SPI を利用したフラッシュ メモリからのブートロード』(XAPP1053)

ザイリンクスの資料で使用されている技術用語については、次の用語集を参照してください。

japan.xilinx.com/company/terms.htm

改訂履歴

次の表に、この文書の改訂履歴を示します。

日付	バージョン	内容
2013 年 7 月 26 日	1.0	初版

Notice of Disclaimer