XAPP1191 (v1.0) 2016 年 7 月 19 日

ビットストリームのリビジョン選択を含む SPI フラッシュ プログラミング

著者 : Ralf Krueger

概要

図 1 にアーキテクチャを示したシステムは、リモートにある FPGA のビットストリームのアップデート、JTAG を介したビットストリームのアップデート、各種ビットストリーム リビジョンによる SPI フラッシュからの FPGA のコンフィギュレーションをサポートします。ビットストリームまたはデザイン固有データは、レジスタ インターフェイスを介して SPI フラッシュ上の指定場所に格納されます。ビットストリームが破損した場合は、ファクトリで組み込まれたゴールデン ビットストリームによって安全にフォールバックされます。SPI デバイスのサイズに応じて複数のビットストリーム リビジョンをフラッシュ デバイスに格納し、ローカル イベントまたはリモート イベントに基づいて FPGA をプログラムできます。このアプリケーション ノートでは、SPI デバイスとのレジスタ インターフェイスについて詳しく説明します。また、ビットストリームおよびその他のデザイン データを Vivado Design Suite から JTAG 経由でフラッシュにダウンロードする方法も説明します。サンプル デザインでは、KCU105 ボードとザイリンクスのダウンロード ケーブルを使用しています。

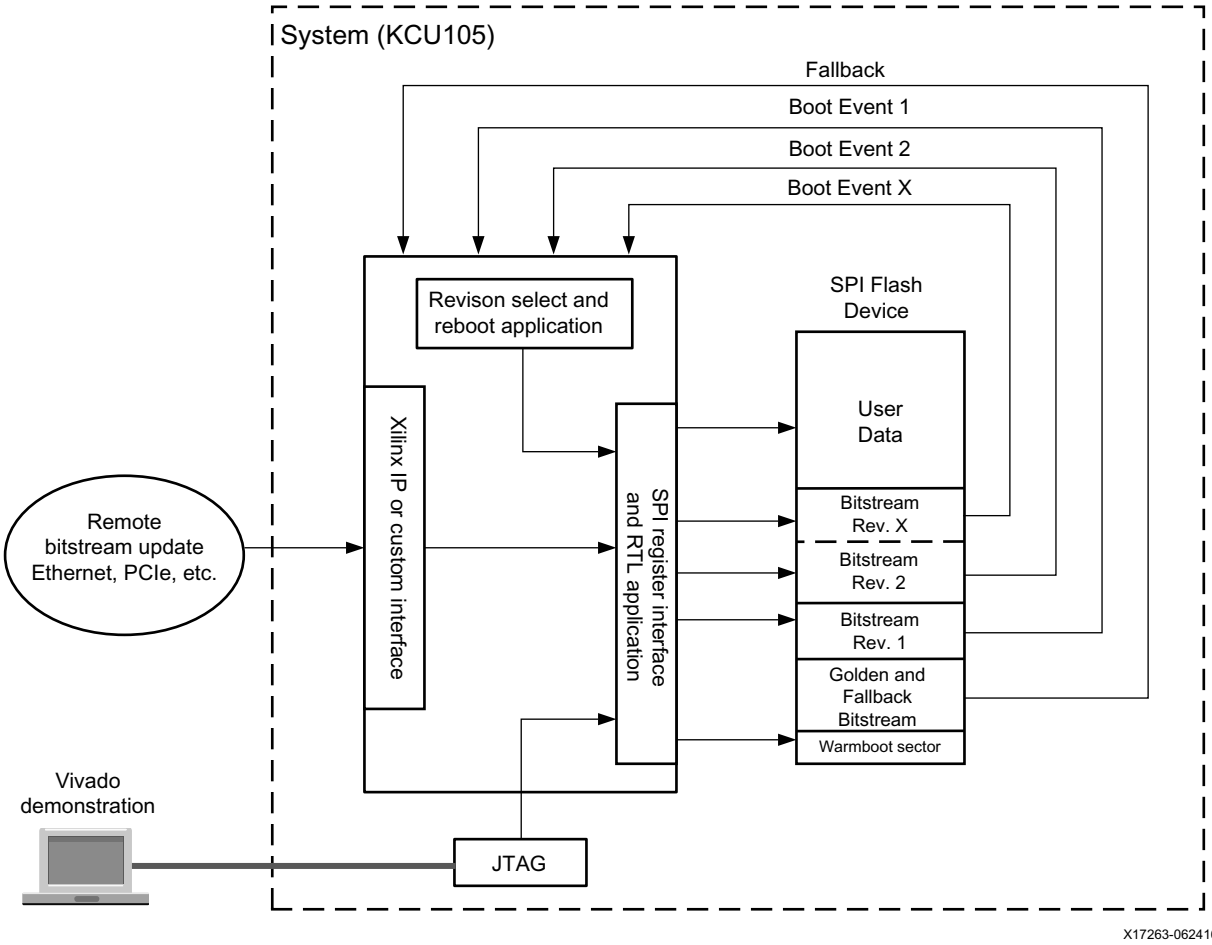


図 1 : システム レベルのブロック図

このアプリケーション ノートのリファレンス デザイン ファイルは、ザイリンクスのウェブサイトからダウンロードできます。デザイン ファイルの詳細は、「リファレンス デザイン」を参照してください。

特長

- すべての UltraScale および UltraScale+ FPGA をサポート
- レジスタ インターフェイスを含む HDL ベースのフラッシュ プログラマ リファレンス デザイン
- イベント ベースのビットストリーム リビジョン選択と自動リブートの HDL サンプル
- 正常動作が確認済みのゴールデン ビットストリームへのエラー時のリカバリ/フォールバック

はじめに

最初のフラッシュ プログラマのリファレンス デザインでは、STARTUPE3 プリミティブを使用した SPI デバイスへの書き込みインターフェイスと、レジスタと制御信号に基づく最上位のインターフェイス デザインを実装しています。また、SPI データのロード アドレス (サブセクターの開始アドレス)、サブセクターの消去数 (基本的にはデータ長の総数)、アプリケーションのプログラミング データが含まれています。リファレンス デザインに含まれる FIFO は、フラッシュ デバイスに書き込むデータをバッファーします。ハンドシェイク信号は、消去およびデータ プログラミング手順全体を制御します。ビットストリーム長とデザイン固有データは SPI ページ サイズ転送に基づいており、最終ページまで転送するには NOOP が必要になる場合があります。

2 番目のリファレンス デザインでは、フラッシュ プログラマ アプリケーションを利用して、フラッシュに格納された各種ビットストリーム バージョンで FPGA を再プログラムし、新しく読み込まれたビットストリームを自動的に起動する機能をデモします。

Vivado ツール ベースのサンプルは、HEX 形式のビットストリームまたは任意の ASCII データをザイリンクス JTAG ダウンロード ケーブル経由で SPI にダウンロードします。hw_jtag コマンドを使用した Tcl スクリプトは、タスクを実行するためのユーザー定義コマンド ライン インターフェイスを提供します。

アプリケーション リファレンス デザイン

このアプリケーション ノートの中心になるのは SPI フラッシュ プログラマです。このプログラマはフラッシュを消去して中間 FIFO に 32 ビット データを書き込み、4 ビット データを FIFO からフラッシュに転送します。リファレンス デザインが受け取るクロック入力は 2 つあり、第 1 のクロックは SPI、FIFO の読み出し側、ステート マシンを駆動します。第 2 のクロックは FIFO の書き込み側を駆動します。双方のクロックは同期または非同期にできます。

データ向けに、最上位アプリケーションのインターフェイスは 3 つの基本レジスタを使用します。データの Valid 信号は、これらのレジスタに有効なデータが含まれることを示します。データ レジスタは、実際の SPI プログラミング データを SPI フラッシュ プログラマに転送するのに使用します (図 2)。プログラミング サイクルの最初に、SPI ロード アドレス (開始アドレス)、SPI データのページ数、SPI サブセクター数を読み込みます。これら 3 つのレジスタが有効になると、最上位アプリケーションは消去プロセスが開始できることを知らせます。プログラマは消去中であることを示す信号をアサートし、SPI の適切な数のサブセクターを消去します。この信号のディアサートにより、消去サイクルの終了が最上位アプリケーションに伝えられます。これで最上位アプリケーションは、FIFO ライト イネーブル信号でデータのダウンロードを開始できます。ダウンロードするデータ量の合計は常に、フラッシュに書き込む全ページ数 (モジュロ 256) と等しくする必要があります。ほとんどの FIFO 関連フラグは、必要に応じて最上位アプリケーションを監視するために使用できます。最後に、データ書き込みサイクルの完了を示す信号が有効になります。

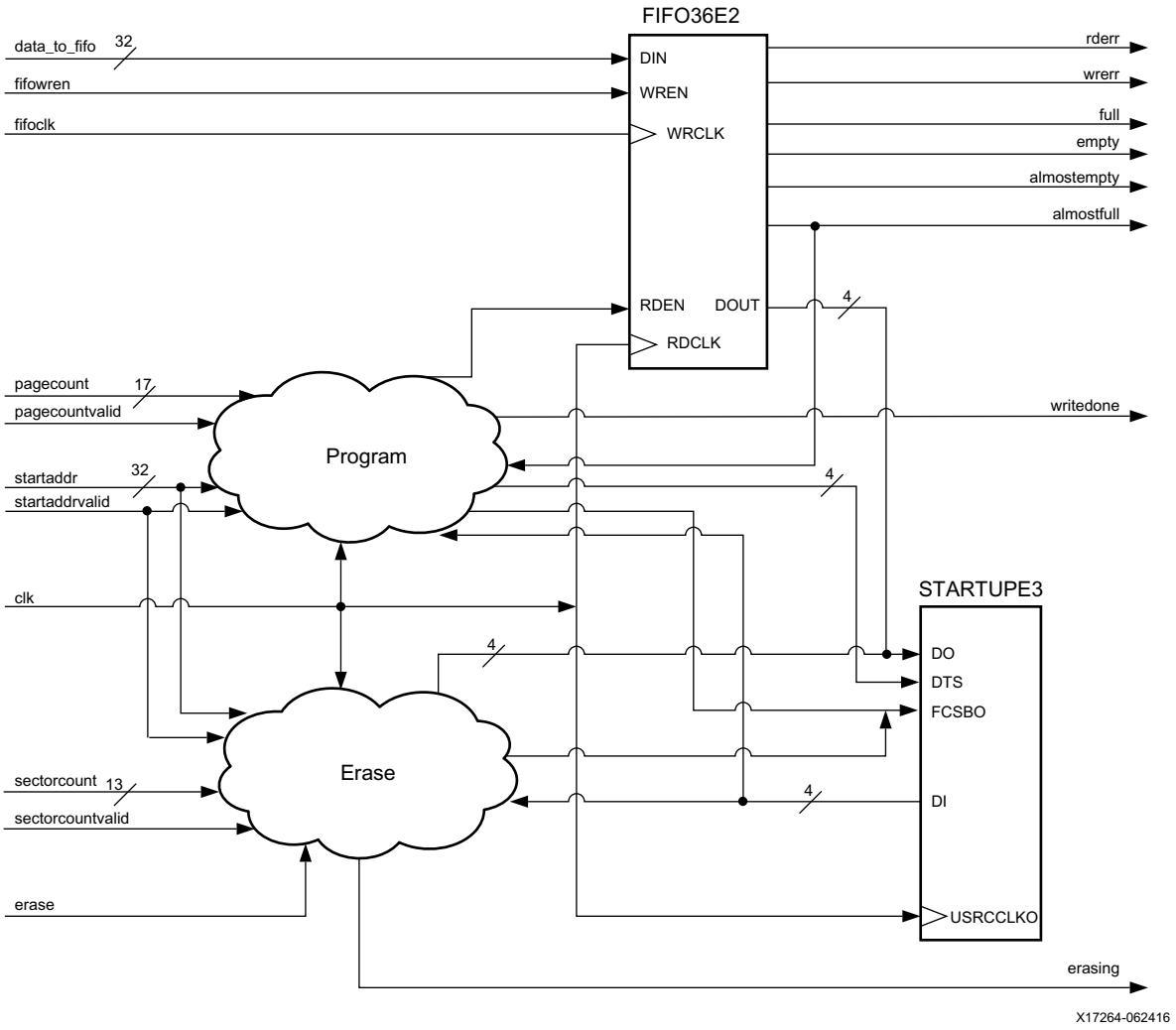


図 2 : SPI フラッシュ プログラマのブロック図

表 1 に、SPI フラッシュ プログラマのポートを示します。オプション ポートを使用すると、潜在的な FIFO の問題を監視、書き込みプロセスの完了を通知、または消去プロセスまたは転送プロセスでエラーを引き起こすコード変更があった場合にリセットできます。

表 1 : 最上位への SPI フラッシュ プログラマのインターフェイス ポート

ポート	方向	幅	説明
Clk	入力	1	SPI フラッシュ、ステート マシン、FIFO RDCLK を駆動するクロックです。
fifoclk	入力	1	FIFO の書き込みクロックです。
data_to_fifo	入力	32	フラッシュに読み込むビットストリームまたはユーザー データです。
startaddr	入力	32	データ読み込み用のビットストリーム/データのロード アドレス (開始アドレス) です。SPI ページにアラインする必要があります。
startaddrvalid	入力	1	読み込まれて有効になっている開始アドレス レジスタ内のロード アドレスです。
pagecount	入力	17	データの合計ページ数です。256 バイトにアラインする必要があります。
pagecountvalid	入力	1	読み込まれて有効になっているデータのページ数です。
sectorcount	入力	13	データ ロード前に消去されるサブセクターの数です。
sectorcountvalid	入力	1	読み込まれて有効になっている消去と書き込みのサブセクターの総数です。

表 1 : 最上位への SPI フラッシュ プログラマのインターフェイス ポート (続き)

ポート	方向	幅	説明
fifowren	入力	1	FIFO ライト イネーブルです。データが data_to_fifo レジスタを介して FIFO に転送されると、この信号がアサートされます。
fifofull	出力	1	オプション : FIFO Full フラグです。
fifoempty	出力	1	オプション : FIFO Empty フラグです。
fifowrerr	出力	1	オプション : FIFO 書き込みエラー フラグです。
fiforderr	出力	1	オプション : FIFO 読み出しエラー フラグです。
writedone	出力	1	オプション : データ書き込みプロセスの完了を知らせます。
reset	入力	1	オプション : デザインのリセットです。ステート マシンは常にアイドルに戻るため不要です。
erase	入力	1	消去プロセスの開始できる状態になるとアサートされます。
erasing	出力	1	消去プロセスの完了を知らせます。

サンプル デザイン

このアプリケーション ノートでは、2 つのサンプル デザインを使用して SPI フラッシュ プログラマの全機能を説明します。

- Vivado ツール ベースの Tcl スクリプトから、ダウンロード ケーブルと JTAG を使用してビットストリームをダウンロードします。BSCAN プリミティブを介してデータがフラッシュ プログラマ モジュールに転送され、ビットストリームまたはデザイン データが SPI デバイスに書き込まれます。
- SPI デバイスのユーザー指定場所にリビジョン選択したアプリケーションが格納され、KCU105 のスイッチを押すと読み込まれます。

フラッシュ プログラマのサンプル

図 3 のブロック図に、最上位のデザインと SPI フラッシュの構成を示します。

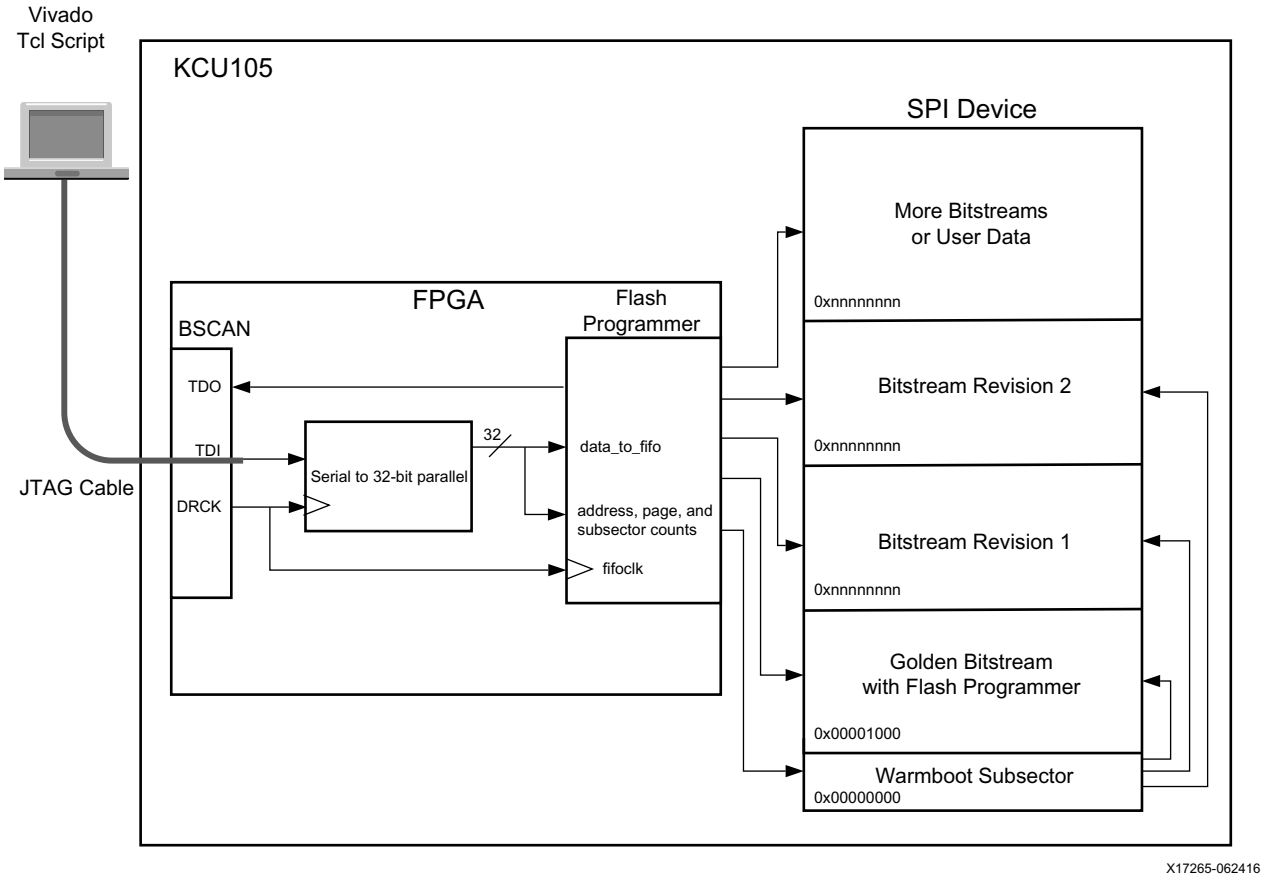


図 3 : SPI フラッシュ デバイスへのビットストリームとデータのダウンロード

このサンプル デザインではまず、アドレス `0x00000000` にあり非常に小さいビットストリームを格納するウォームブート SPI サブセクターを用いることを念頭に置いています。このビットストリームを有効なビットストリームが格納された SPI アドレス空間内のアドレスに対して使用し、ウォームブートの IPROG 命令を実行します。このウォームブート セクターは、(必要な場合) Tcl スクリプトとリビジョン制御どちらかのビットストリームによってアップデートされます。どちらかのビットストリームが破損した場合、システムは自動的にゴールデン ビットストリームにフォールバックします。ウォームブート ビットストリームは通常、ゴールデン ビットストリームの上位にあるサブセクター (アドレス `0x00000000`) にあります。Tcl スクリプトによってロードされるウォームブート ビットストリームの例を次に示します。

```
AA995566  -- Sync Word
20000000  -- NOP
20000000  -- NOP
3003E001  -- BPI/SPI configuration option register
0000066C  -- SPI Read Opcode (Quad Fast Read 32 bit, X4, 32 bit Addr)
30008001  -- Write 1 word to CMD register (flip the 3 to a 0)
00000012  -- BPI/SPI restart bitstream read
20000000  -- NOP
20000000  -- NOP
30020001  -- Write to WBSTAR
XXXXXXXX  -- Warmboot start address
30008001  -- Write 1 word to CMD register
0000000F  -- IPROG
20000000  -- NOP
20000000  -- NOP
```

Vivado ツール ベースの Tcl スクリプトは、ビットストリームまたはユーザー データをダウンロードするためのコマンド ライン インターフェイスを提供します。このスクリプトはダウンロード ケーブルを JTAG モードに設定し、*HW_JTAG Tcl コマンドを使用して BSCAN プリミティブと通信します。BSCAN プリミティブの DRCK クロック出力は、データをデシリアライズして適切なレジスタを読み込むシフト レジスタをクロック駆動します。終了時に消去信号がアサートされ、消去プロセスが開始されます。フラッシュ プログラマ モジュールが消去フラグをアサートし、Tcl スクリプトはこの消去フラグがディアサートされるまで待機します。次に、あらかじめファイルからバッファーに読み込まれていたデータが、バッファーが空になるまでダウンロードされます。バッファー サイズが常に SPI ページ サイズの倍数になっていることが重要です。ファイルの最終ページのデータが足りない場合 (最も可能性の高いシナリオ)、このスクリプト サンプルはバッファー内のデータを自動的に調整 (充填) します。

ケーブル ダウンロード プロセスと効果的なケーブル クロック周波数は、ダウンロードごとに大きく異なる場合があります。BSCAN プリミティブから出力される DRCK クロック周波数は、SPI クロックよりも大幅に低速であるということを除いて予測不可能です。このため、このフラッシュ プログラマ サンプルでは、2 つのクロック ドメイン間で信号を同期する必要はありません。

このフラッシュ プログラマ サンプルとアプリケーション ノートの説明は、ザイリンクス KCU105 評価ボードで使用するためのものです。このボードの 256Mb SPI フラッシュはボード上面に実装されており、STARTUPE3 プリミティブと Bank0 の関連ピンを介して FPGA に接続されています。このサンプルではボード裏面の SPI フラッシュは使用しません。通常、KCU105 ボード上の Micron N25Q256A SPI フラッシュ デバイスに 2 つの非圧縮ビットストリームを格納できます。リビジョンを選択するには 3 つのビットストリームを格納するストレージ容量が必要になるため、このサンプル デザインでは圧縮されたビットストリームのみを使用します。比較的小さいデザイン変更でもビットストリーム サイズが大幅に増加する場合があるため、圧縮後のビットストリーム サイズに基づいて SPI フラッシュを選択することは推奨しません。このサンプル デザインのサイズはかなり小さくリソースの 1% 未満しか利用しないため、どのようなデザイン変更があってもビットストリームのサイズが 256Mb SPI フラッシュの容量を超えることはないはずです。実際のアプリケーションでは、フル サイズのビットストリームをちょうど処理できる SPI フラッシュ デバイスを選択することを推奨します。

リファレンス デザインのダウンロードとディレクトリ構造

「リファレンス デザイン」をダウンロードして選択したディレクトリにインストール (解凍) します。サンプル デザインのルート名は xapp1191 です。`spiflashprogrammer` と `rev_sel` という 2 つのデザイン ディレクトリがあります。`spiflashprogrammer` ディレクトリにはフラッシュ プログラマ デザインと最上位デザインが含まれます。最上位デザインは Vivado ツールとフラッシュ プログラマ デザイン間をインターフェイスします。`rev_sel` ディレクトリに含まれる 2 つのデザインは、このアプリケーション ノートのリビジョン選択部分をデモするもので、`spiflashprogrammer` モジュールを利用します。`spiflashprogrammer` モジュールのサンプル アプリケーションと見なすことができます。xapp1191 ディレクトリには、Vivado ツールから KCU105 上の FPGA で実行中のアプリケーションにビットストリームをダウンロードする Tcl スクリプト (`write_to_spi.tcl`) と、このサンプルで用いる多数の HEX ファイル (プリコンパイル済みビットストリーム) も含まれます。

HEX ビットストリーム ファイルを生成するには、Vivado ツールの `write_cfgmem` コマンドを使用します。Tcl スクリプトの `write_bitstream` と `write_cfgmem` は、xapp1191 ディレクトリ (`gen_bitstream.tcl`) に含まれています。デザインの再コンパイル後にサンプルを実行するには、HEX ファイルを <アプリケーション名> ディレクトリから xapp1191 ルート ディレクトリにコピーするか、または write_cfgmem コマンドを変更してルート ディレクトリに直接書き込みます。

注記 : write_cfgmem コマンドの -loadbit アドレス オプションは影響を及ぼしません。

次のコマンドを実行する前にすべての適切なプロパティと変数を設定します。

```
write_bitstream $impl_path/${top_name}_compressed.bit -force -verbose
write_cfgmem -format hex -interface SPIx4 -size 32 -loadbit "up 0x0
$impl_path/${top_name}_compressed.bit" -file ${top_name}_compressed -force
```

すべてのビットストリームは圧縮形式で生成されます (圧縮形式であることはファイル名に反映される)。消去プロセスと読み込み時間の違いを示すため、非圧縮バージョンの `flashprogrammer` サンプルもあります。

手順

ゴールデン ビットストリームの読み込み、ビットストリームのアップデート、フォールバック

write_spi_tcl スクリプトには多数のプロシージャが含まれています。今回直接利用するメイン プロシージャは wr_data です。wr_data プロシージャを実行するには、ロード アドレス (SPI 開始アドレス)、ウォームブート アドレス (ほとんどの場合 0)、ファイル名、ビットストリーム タイプ (アップデート、ウォーム、その他) を含む複数のコマンド ライン引数を使用します。すべての引数は省略なしで明示的に指定するか、またはコマンドの最初の文字を使用します。-h 引数を使用すると、すべてのオプションが表示されます。このスクリプトはエラー チェックを実行し、ハードウェア ケーブルを JTAG モードで開き、適切なファイルを読み込み、さまざまなデータ操作を実行した後で、SPI フラッシュ デバイスの消去およびプログラムを実行します。wr_data コマンドの実行中、ステータス メッセージなどの情報が Td コンソール ウィンドウに表示されます。

注意 : 実行中のダウンロード プロセスを取り消すと、アプリケーションのステート マシンが不定な状態のままになります。先に進む前に、ハードウェア サーバーを切断し、ターゲットを再度開きし、spiflashprogrammmer_top.bit ファイルを使用してデバイスを再プログラムします。

1. KCU105 ボードにケーブルを接続して電源が入っていることを確認します。
2. spiflashprogrammer ディレクトリに移動し、Vivado ツールで spiflashprogrammer (spiflashprogrammer.xpr) デザインを開きます。

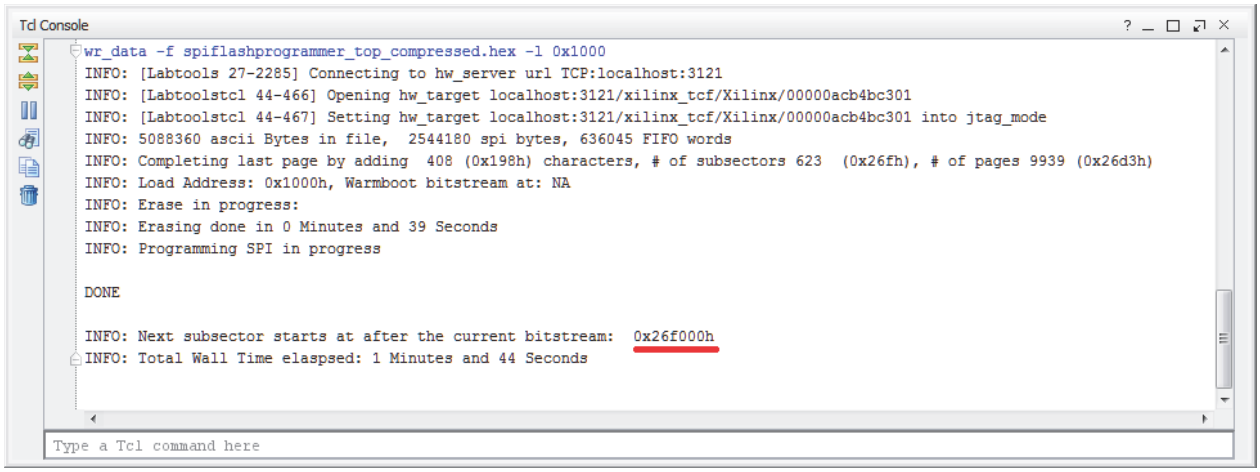
これ以降の手順の大半は Vivado ツールの Td コンソール ウィンドウで実行するので、メッセージを読みやすくするためにこのウィンドウを浮動させて拡大します。画面上の空白エリアにウィンドウを動かして、このウィンドウが Vivado ツールのメイン画面によって隠されることなく、常に表示されるようにします。

3. Td コンソール ウィンドウのソースで、write_to_spi.tcl ソース <path_to_xapp1191_dir>/write_to_spi.tcl を見つけます。
4. ハードウェア マネージャを開いてターゲットに接続し、デザイン ビットストリーム (spiflashprogrammmer_top.bit) を使用してデバイスをプログラムします。ボードの LED が LED0 から LED7 まで順に点灯します。これは、このデザイン固有のサインです。
5. この手順では最上位デザインであるゴールデン ビットストリームが読み込まれます。すべてのビットストリームが、xapp1191 ディレクトリのルート ディレクトリに格納されている必要があります。コンソールのコマンド ライン ウィンドウに次のコマンドを入力します。

```
wr_data -f spiflashprogrammer_top_compressed.hex -l 0x1000
```

ヒント : *wr_* と入力するだけで、手順 3 で読み込んだ Tcl コマンドをオートコンプリートと一意に識別します。また、ヘルプ コマンドとして wr_data -h または wr_data -help も使用できます。

標準 Tcl コマンド ラインのポップアップ ウィンドウが開いて Running scan_dr_hw_jtag というメッセージが表示され、JTAG 操作と SPI プログラミングが実行中であることを示します。コンソール ウィンドウのメッセージ部分 (図 4) には、最初にハードウェア マネージャからの INFO メッセージが、次に SPI ロード アドレスなどのファイルに関する情報、続いて消去およびプログラミングの進行中メッセージが表示されます。ホスト マシンとその負荷によって異なりますが、プロセス全体の処理に約 1 ～ 2 分かかります。プログラミングが完了すると、SPI 内で次に空いているサブセクター アドレスが経過時間と共に表示されます。この空白サブセクター アドレス (図 4 の例では `0x26f000`) が次のビットストリームを読み込むアドレスになるので、これを書き留めておきます。



X17259-062416

図 4 : 手順 5 の Td コンソール

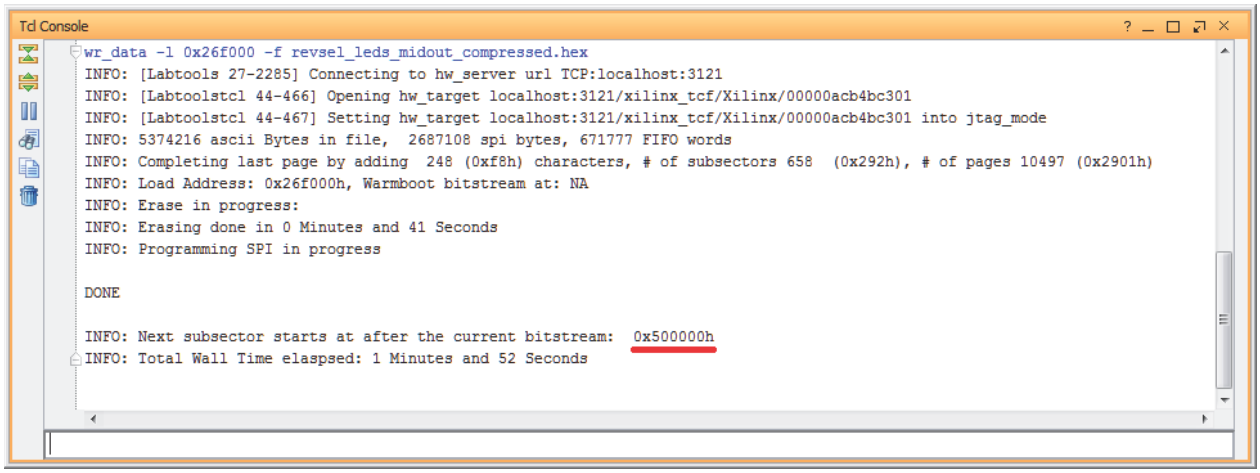
ゴールデン ビットストリームが読み込まれて、アドレス `1000h` に格納されました。

6. このステップでは、アップデートされたビットストリームがウォームブート アドレスと一緒にサブセクター 0 に読み込まれます。このサンプルでは、アップデート ビットストリームの読み込みとフォールバック機能を示すために、`revsel` ビットストリームの 1 つが使用されています。また、フォールバックをデモするために、まったく同じビットストリームの破損バージョンが使用されています。

ヒント : Td コンソール ウィンドウ (図 5) が表示されていることを確認します。

   a. コンソール コマンド ウィンドウ (図 5) に、次のコマンドを入力します (引数の順序は関係なし)。

```
wr_data -l 0x26f000 -f revsel_leds_midout_compressed.hex
```



X17260-062416

図 5 : 手順 6 の Td コンソール

b. 次の空白サブセクター アドレス `0x500000` を書き留めます。

c. サブセクター `0` にウォームブート アドレスを読み込みます。

```
wr_data -w 0x26f000 -t w
```

d. 前回は `-o` で十分でしたが (デフォルト)、今回はファイル タイプを明示的に指定する必要があります。サブセクター 0 には短いウォームブート ビットストリームが格納されています。

e. KCU105 の PROG ボタンを押します。ユーザー LED が中央の LED から左右に順に点灯します。これは、このデザイン/ビットストリーム固有のサインです。

7. この手順ではフォールバック機能をデモします。KCU105 にある 5 つのユーザー スイッチ (SW6 ～ SW10) を見つけます。スイッチ SW9 を押すと FPGA がゴールデン ビットストリームに戻り、LED が 0 から 7 に順に点灯します。詳しい説明は、「ビットストリームのリビジョン選択のサンプル」を参照してください。

   注記 : 誤って SW6 を押した場合、FPGA は SPI 内の上位アドレスから別のビットストリームを読み込もうとします。そのビットストリームはまだ読み込まれていないので、FPGA はプログラムされません。手順 4 に戻ってやり直します。SW8 を押すと現在のビットストリームを再度読み込みます。その他のスイッチは無効です。

   a. HEX ファイルのビット番号 999 が意図的に 0 から 1 に反転されたビットストリームが読み込まれます。これは xapp1191 ディレクトリに含まれる `revsel_leds_midout_compressed_fbbad.hex` ビットストリームで、ビット番号 999 以外は前回読み込まれた最初のビットストリーム `revsel_leds_midout_compressed.hex` とまったく同じです。

   ```
   wr_data -l 0x26f000 -f revsel_leds_midout_compressed_fbbad.hex
   ```

   b. SW9 を押すと、サブセクター `0` のウォームブート アドレスがアドレス `1000h` (ゴールデン ビットストリームのアドレス) で上書きされるため、ウォームブート サブセクターも再度読み込む必要があります。詳細は、「ビットストリームのリビジョン選択のサンプル」を参照してください。

   c. 次のコマンドを入力すると、サブセクター `0` にアドレス `26f000h` を指すウォームブート ビットストリームが読み込まれます。

   ```
   wr_data -w 0x26f000 -t w
   ```

   d. PROG ボタンを押すと、不正なビットストリームの読み込みはできず、FPGA プログラミングはゴールデン ビットストリームにフォールバックします。

   e. 正常なビットストリームを再度読み込むには、次のコマンドを入力します。

   ```
   wr_data -l 0x26f000 -f revsel_leds_midout_compressed.hex
   ```

   f. もう一度 PROG ボタンを押すと、アップデート ビットストリームによって、フォールバックなしで FPGA がコンフィギュレーションされます。

ビットストリームのリビジョン選択のサンプル

ビットストリーム リビジョンの動的選択が可能なこのサンプルでは、IPROG アドレスでウォームブート サブセクター 0 を再プログラムする機能を利用しています。IPRG アドレスは SPI デバイスに既に格納されているブート ビットストリームを指します。このサンプルでは、spiflashprogrammer のわずかに変更したバージョンを使用して、サブセクター 0 を消去してから、ウォームブート ビットストリームをサブセクター 0 に書き込みます。1 つの ブロック RAM に複数のウォームブート ビットストリームが格納されています。システム内のイベント (このサンプルではスイッチ) に基づいて、ブロック RAM 内にプリロードされた適切なウォームブート ビットストリームが選択され、サブセクター 0 に書き込まれます。サブセクターのプログラム後、現在実行中のビットストリーム リビジョンが ICAP プリミティブを介して IPROG コマンドを発行します。FPGA はアドレス 0x00000000 からブートされます。このアドレスには、SPI デバイス内の目的のビットストリーム位置を指すウォームブート ビットストリームが格納されています。このサンプルではブロック RAM の INIT 属性を使用して、目的の位置のブロック RAM を適切に初期化します。これらのビットストリームには、ブート対象として選択されたビットストリームの開始アドレスが格納されています。ブロック RAM 内のウォームブート ビットストリームは、前述したビットストリームとまったく同じ構成です。ウォームブート ビットストリームのロード アドレスは、手順 7 で使用されたロード アドレスによって事前設定されています (例 : 0x1000 と 0x26f000)。

次に示すのは、ブロック RAM 内に格納されたウォームブート ビットストリームの例です。

```
INIT_00 => X"ffffffff_ffffffff_ffffffff_ffffffff_ffffffff_ffffffff_ffffffff_ffffffff",
INIT_01 => X"20000000_00000012_30008001_0000066C_3003E001_20000000_20000000_AA995566",
INIT_02 => X"ffffffff_20000000_20000000_0000000f_30008001_00001000_30020001_20000000",
INIT_03 => X"ffffffff_ffffffff_ffffffff_ffffffff_ffffffff_ffffffff_ffffffff_ffffffff",
```

次に示すのは、ICAP リブートのコマンド シーケンスです。

```
AA995566   -- Sync Word
20000000   -- Type 1 NO OP
30020001   -- Type 1 Write 1 Word to WBSTAR
00000000   -- Warm Boot Start Address
20000000   -- Type 1 NO OP
30008001   -- Type 1 Write 1 Words to CMD
0000000F   -- IPROG Command
```

「ゴールデン ビットストリームの読み込み、ビットストリームのアップデート、フォールバック」から継続している場合、ゴールデン ビットストリームとリビジョン 1 のビットストリームは読み込み済みであると考えられます。

1. リビジョン 2 のビットストリームの読み込みを開始します。
   a. SW9 を押してゴールデン ビットストリームに戻ります。
   b. LED が、ゴールデン ビットストリームがアクティブであるを示します。
2. 正しいアドレス (5 の後に 5 個の 0 が続く) とリビジョン 2 のビットストリームを入力します。

   ```
   wr_data -l 0x500000 -f revsel_leds_7to0_compressed.hex
   ```

   次のコマンドを入力します。

   ```
   wr_data -w 0x500000 -t w
   ```
3. SPI にリビジョン 2 のビットストリームが読み込まれ、実行できる状態になりました。
   a. KCU105 の PROG ボタンを押します。
   b. このデザイン/ビットストリーム固有のサインとして、LED が左から右に順に点灯します。
4. SW8 を押すと、リビジョン 1 のビットストリームがブートされます (LED を確認)。
   a. SW6 を押すと、FPGA がリビジョン 2 のビットストリームに戻ります。
   b. スイッチ (SW8 または SW6) でリビジョン 1 または 2 を選択することで、目的のビットストリームが再度読み込まれます。
5. ゴールデン ビットストリームへのフォールバックが引き続き正しく動作することを示すため、ビット 999 が 0 から 1 に反転された不正なリビジョン 2 のビットストリームを読み込みます。
6. SW9 を押すと、FPGA がゴールデン ビットストリームに戻ります。次のアドレスを読み込みます。

   ```
   wr_data -l 0x500000 -f revsel_leds_7to0_compressed_fbbad.hex
   ```

a. 前述のとおり、ウォームブート セクターを再度読み込む必要があります。これは、手順 6 で SW9 を押したことでサブセクター 0 のウォームブート アドレスがアドレス 1000h で上書きされ、ゴールデン ビットストリームがブートされるためです。

b. このサンプルでは、リビジョン 1 のビットストリームがデフォルト リビジョンのビットストリームです。

c. 次のコマンドはサブセクター 0 にウォームブート ビットストリームを読み込み、アドレス 26f000h を指します。

```
wr_data -w 0x26f000 -t w
```

d. PROG ボタンを押すと、リビジョン 1 のビットストリームが読み込まれます。

e. SW6 を押すと、リビジョン 2 のビットストリームを読み込もうとしますが、このビットストリームには不正 (反転) ビットが含まれるため読み込みはできず、FPGA プログラムはゴールデン ビットストリームに戻ります。

7. 正常なリビジョン 2 のビットストリームを再度読み込むには、次の手順を実行します。

a. 次のコマンド ラインを入力します。

```
wr_data -l 0x500000 -f revsel_leds_7to0_compressed.hex
```

b. 続いて、次のコマンド ラインを入力します。

```
wr_data -w 0x26f000 -t w
```

c. PROG ボタンを押してから SW6 を押すと、正常なリビジョン 2 のビットストリームがブートされます。

d. SW8 と SW6 を押すと、もう一度このリビジョンをブートできます。または、PROG ボタンを押した直後に次のコマンドを実行すると、リビジョン 2 が読み込まれます。

```
wr_data -w 0x500000 -t w
```

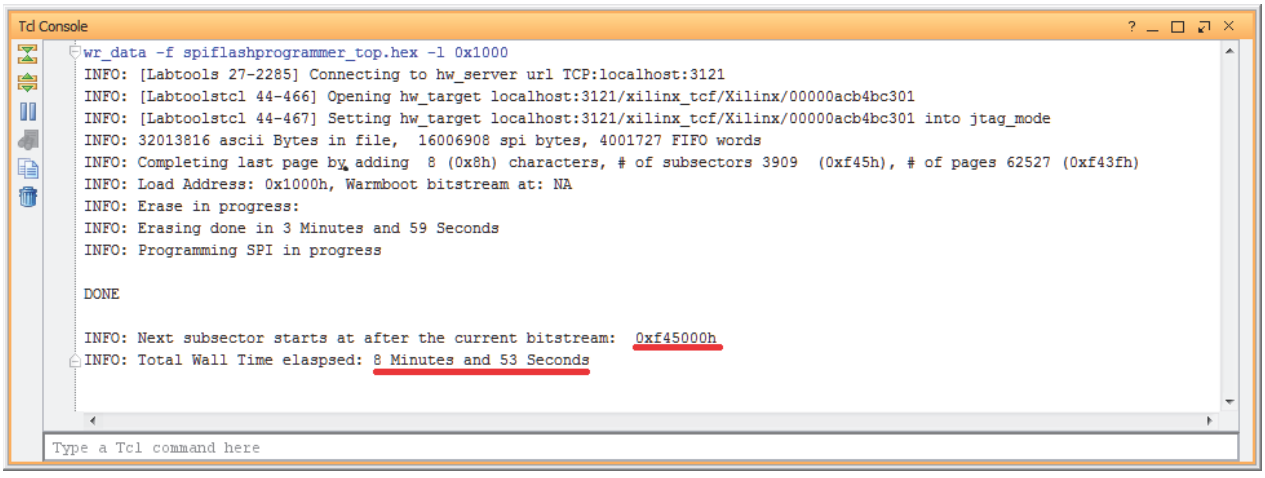
フル サイズのビットストリームのプログラミング

xapp1191 ディレクトリには、フル サイズのビットストリームと spiflashprogrammer_top デザインが含まれています。KCU105 SPI デバイスにはフル サイズの KCU040 ビットストリームを 2 つ格納できます。フル サイズのビットストリームを読み込む時間 (消去時間とプログラミング時間の両方) は、圧縮されたバージョンのビットストリームよりも大幅に長くなります。実際にかかる時間は、ホストマシンの性能とホストマシンの現在の負荷によって異なります。次のサンプルでは、SPI デバイスのアドレス 0x1000 にフル サイズのビットストリームが読み込まれます。標準的な実行時間は 10 分前後です。このフル ビットストリームは 256Mb の SPI デバイスのほぼ 1/2 を必要とします (256Mb のうち約 120Mb)。

注記 : このフル サイズ ビットストリームのプログラムを実行すると、前述の例でこれまでに読み込まれたすべてのビットストリームとデータが上書きされます。

1. KCU105 の SW9 を押すとゴールデン ビットストリームに戻ります (図 6)。

```
wr_data -f spiflashprogrammer_top.hex -l 0x1000
```

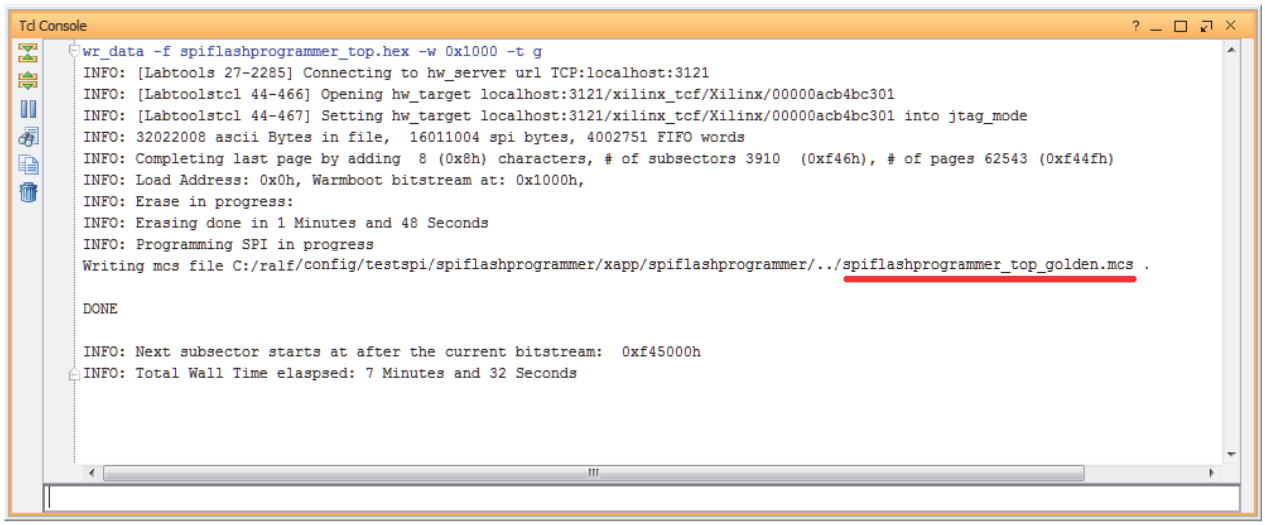


X17261-062416

図 6 : 手順 1 の Td コンソール

2. この Tcl スクリプトは、ウォームブート アドレスとゴールデン ビットストリームの両方を含む単一 MCS ファイルを書き込む追加機能を提供します。この MCS ファイルを使用すると、ハードウェア マネージャーを使用して SPI デバイスをプリロードできます。生成される MCS ファイルの名前の末尾には、_golden が追加されます。次のコマンド (図 7) はウォームブート サブセクターとゴールデン ビットストリームを読み込んで、1 つの MCS ファイルに書き込みます。このウォームブート サブセクターにはゴールデン ビットストリームの開始アドレスがプログラムされているため、(ハードウェア マネージャーを使用して) SPI フラッシュにプログラムすると、FPGA のブートとゴールデン ビットストリームの読み込みが実行されます。ウォームブート サブセクターは特定のビットストリーム リビジョンを指すこともできます。ただし、SPI デバイスから FPGA をブートするには、そのリビジョンが SPI フラッシュ デバイス内に存在して (読み込まれて) いる必要があります。

   ```
   wr_data -f spiflashprogrammer_top.hex -w 0x1000 -t g
   ```



X17262-062416

図 7 : 手順 2 の Td コンソール

3. ダウンロード ケーブルを非 JTAG モードに戻すには、もう一度ハードウェア マネージャーでターゲット デバイスを開きます。

まとめ

このアプリケーション ノートでは、リモートまたは JTAG 経由での SPI フラッシュ デバイスのプログラミング、ビットストリームのアップデート、各種ビットストリームのリビジョン選択の機能を説明しています。リファレンス デザイン ファイルに含まれるサンプルは、SPI フラッシュを使用して FPGA ビットストリームをプログラムする一般的な使用例をデモします。

リファレンス デザイン

このアプリケーション ノートのリファレンス デザイン ファイルは、ザイリンクスのウェブサイトからダウンロードできます。

表 2 に、リファレンス デザインの詳細を示します。

表 2 : リファレンス デザインの詳細

パラメーター	説明
全般	
開発者	Ralf Krueger
ターゲット デバイス	UltraScale および UltraScale+ FPGA
ソース コードの提供	あり
ソース コードの形式	VHDL
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、またはサードパーティからデザインへのコード /IP の使用	なし
シミュレーション	
論理シミュレーションの実施	なし
タイミング シミュレーションの実施	なし
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	なし
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	Vivado シミュレータ バージョン 2015.4 またはそれ以降
SPICE/IBIS シミュレーションの実施	なし
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	Vivado 合成、バージョン 2015.4 またはそれ以降
使用したインプリメンテーション ツール/バージョン	Vivado バージョン 2015.4 またはそれ以降
スタティック タイミング解析の実施	あり
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	KCU105 評価ボード

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
2016 年 7 月 19 日	1.0	初版

法的通知

本通知に基づいて貴殿または貴社 (本通知の被通知者が個人の場合には「貴殿」、法人その他の団体の場合には「貴社」。以下同じ) に開示される情報 (以下「本情報」といいます) は、ザイリンクスの製品を選択および使用することのためにのみ提供されます。適用される法律が許容する最大限の範囲で、(1) 本情報は「現状有姿」、およびすべて受領者の責任で (with all faults) という状態で提供され、ザイリンクスは、本通知をもって、明示、黙示、法定を問わず (商品性、非侵害、特定目的適合性の保証を含みますがこれらに限られません)、すべての保証および条件を負わない (否認する) ものとします。また、(2) ザイリンクスは、本情報 (貴殿または貴社による本情報の使用を含む) に関係し、起因し、関連する、いかなる種類・性質の損失または損害についても、責任を負わない (契約上、不法行為上 (過失の場合を含む)、その他のいかなる責任の法理によるかを問わない) ものとし、当該損失または損害には、直接、間接、特別、付随的、結果的な損失または損害 (第三者が起こした行為の結果被った、データ、利益、業務上の信用の損失、その他あらゆる種類の損失や損害を含みます) が含まれるものとし、それは、たとえ当該損害や損失が合理的に予見可能であったり、ザイリンクスがそれらの可能性について助言を受けていた場合であったとしても同様です。ザイリンクスは、本情報に含まれるいかなる誤りも訂正する義務を負わず、本情報または製品仕様のアップデートを貴殿または貴社に知らせる義務も負いません。事前の書面による同意のない限り、貴殿または貴社は本情報を再生産、変更、頒布、または公に展示してはなりません。一定の製品は、ザイリンクスの限定的保証の諸条件に従うこととなるので、http://japan.xilinx.com/legal.htm#tos で見られるザイリンクスの販売条件を参照してください。IP コアは、ザイリンクスが貴殿または貴社に付与したライセンスに含まれる保証と補助的条件に従うことになります。ザイリンクスの製品は、フェイルセーフとして、または、フェイルセーフの動作を要求するアプリケーションに使用するために、設計されたり意図されたりしていません。そのような重大なアプリケーションにザイリンクスの製品を使用する場合のリスクと責任は、貴殿または貴社が単独で負うものです。http://japan.xilinx.com/legal.htm#tos で見られるザイリンクスの販売条件を参照してください。

自動車用のアプリケーションの免責条項

オートモーティブ製品 (製品番号に「XA」が含まれる) は、ISO 26262 自動車用機能安全規格に従った安全コンセプトまたは余剰性の機能 (「セーフティ設計」) がない限り、エアバッグの展開における使用または車両の制御に影響するアプリケーション (「セーフティ アプリケーション」) における使用は保証されていません。顧客は、製品を組み込むすべてのシステムについて、その使用前または提供前に安全を目的として十分なテストを行うものとします。セーフティ設計なしにセーフティ アプリケーションで製品を使用するリスクはすべて顧客が負い、製品の責任の制限を規定する適用法令および規則にのみ従うものとします。

この資料に関するフィードバックおよびリンクなどの問題につきましては、jpn_trans_feedback@xilinx.com まで、または各ページの右下にある [フィードバック送信] ボタンをクリックすると表示されるフォームからお知らせください。いただきましたご意見を参考に早急に対応させていただきます。なお、このメール アドレスへのお問い合わせは受け付けておりません。あらかじめご了承ください。