XAPP1247 (v1.0) 2015 年 8 月 13 日

7 シリーズ FPGA および SPI を使用するマルチブート

著者: Bryan Li、Kiran K Gakhal

概要

このアプリケーション ノートでは、SPI (シリアル ペリフェラル インターフェイス) コンフィギュレーション モードで、7 シリーズ FPGA を使用して有効なマルチブート デザインを構築する場合の主な概念について説明します。7 シリーズのマルチブート機能は、FPGA アプリケーションの制御下で複数の FPGA ビットストリームの読み込みを可能にします。ここでは、FPGA ビットストリーム設定を使用してマルチブート機能をインプリメントする手順、フォールバックを開始する方法、および FPGA ステータス レジスタを使用してフォールバック動作をデバッグおよび検証する方法について説明します。このアプリケーション ノートには、SPI コンフィギュレーション モードを使用する 7 シリーズ FPGA のマルチブート機能とフォールバック機能を検証するためのリファレンス デザインが含まれています。

ここでは、Kintex®-7 FPGA および N25Q128 Micron Quad SPI シリアル フラッシュを搭載した KC705 評価ボードと、Vivado® Design Suite 2015.1 を使用してデザイン フローを示します。マルチブート機能に関する追加情報および SPI コンフィギュレーション モードの詳細は、『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470) [参照 1] を参照してください。

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。

はじめに

7 シリーズ FPGA のマルチブートおよびフォールバックは、フィールドでのシステム アップデートをサポートする機能です。ビットストリーム イメージは、フィールドで動的にアップグレードできます。FPGA のマルチブートは、オンザフライでイメージを切り替える機能です。マルチブート コンフィギュレーション プロセス中にエラーが検出されると、FPGA はフォールバック機能を開始して、検証済みのデザイン (ゴールデン イメージ) をデバイスに読み込むことができます。

フォールバックが実行されると、内部生成されたパルスによって、専用のマルチブート ロジック、ウォーム ブート開始アドレス レジスタ (WBSTAR)、およびブート ステータス (BOOTSTS) レジスタを除くコンフィギュレーション ロジック全体がリセットされます。このリセット パルスで INIT_B および DONE が Low になり、コンフィギュレーション メモリがクリアされ、フラッシュ メモリ デバイスのアドレス 0 からコンフィギュレーション プロセスが再開します。

この資料は表記のバージョンの英語版を翻訳したもので、内容に相違が生じる場合には原文を優先します。資料によっては英語版の更新に対応していないものがあります。日本語版は参考用としてご使用の上、最新情報につきましては、必ず最新英語版をご参照ください。

このアプリケーション ノートは、主に次のセクションで構成されています。

マルチブートの基本

FPGA のマルチブート機能は、リモート アップデート用のイメージをオンザフライで切り替えることができます。マルチブート コンフィギュレーション プロセス中にエラーが検出されると、FPGA はフォールバックを開始して、検証済みのデザイン (ゴールデン イメージ) をデバイスに読み込むことができます。このソリューションでは、次のコンポーネントを格納するための予約領域を持つフラッシュ メモリを使用します。

- フォールバック、または「ゴールデン ビットストリーム」
- マルチブート、または「アップデート ビットストリーム」

リモート プログラム方法をインプリメントし、これを用いて新規または改良したビットストリームをアップデート ビットストリーム領域に書き込みます。FPGA は、このアップデート ビットストリームでコンフィギュレーションを試みます。

リモート アップデートがエラーになるか中断した場合は、フォールバックしてゴールデン ビットストリームから確実にコンフィギュレーションする必要があります。

マルチブートは、フラッシュ メモリ内の特定のアドレスを指定してビットストリームを選択的に読み込む機能です。内部生成パルス (IPROG) によりコンフィギュレーション ロジックが開始され、ゴールデン ビットストリーム WBSTAR (ウォーム ブート開始アドレス) レジスタで指定されたアドレス位置のアップデート ビットストリームにジャンプし、アップデート ビットストリームを読み込みます。アップデート ビットストリームの読み込み中にコンフィギュレーション エラーが検出されると、フォールバックがトリガーされてゴールデン ビットストリームが読み込まれます。図 1 は、このアプリケーション ノートで例示する一般的なマルチブートおよびフォールバックのフローを示しています。

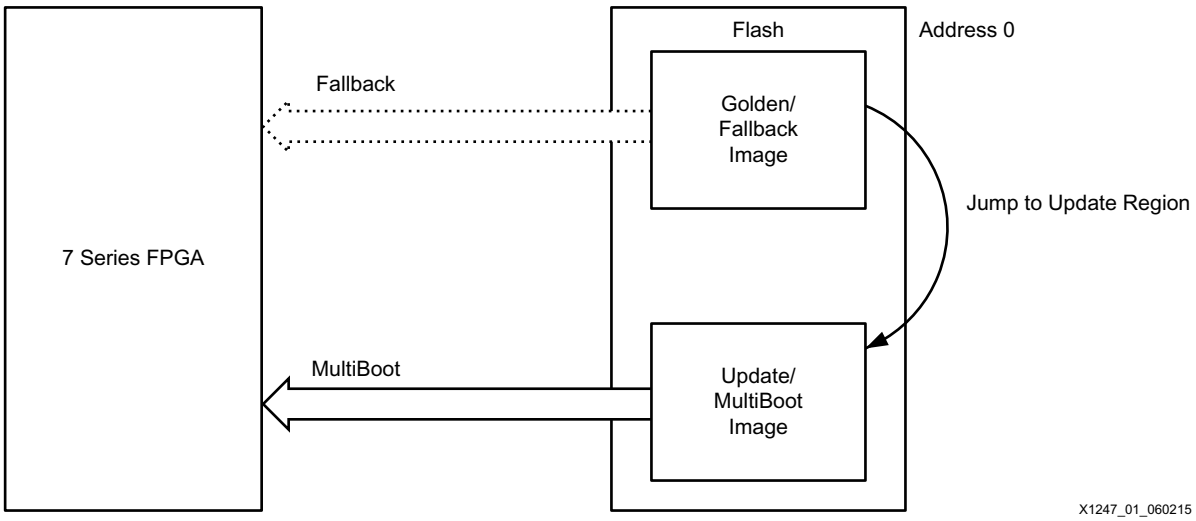

図 1: マルチブートおよびフォールバックのフロー

マルチブート イベントの準備として、内部プログラム (IPROG) コマンドおよび次のイメージ開始アドレス (WBSTAR) コマンドが、ゴールデン ビットストリーム内に設定されます。

マルチブートをインプリメントする方法としては、次の 2 つがあります。

1. ビットストリームへの PROG の組み込み (このアプリケーション ノートで取り上げる)。IPROG は、NEXT_CONFIG_ADDR ビットストリーム オプションを使用して有効にします。
2. ICAPE2 を使用した IPROG (このアプリケーション ノートでは取り上げない)。レジスタ書き込みコマンドを ICAPE2 プリミティブに適用します。

ビットストリームへの IPROG 組み込みは自動化されたオプションであり、マルチブート設定はビットストリームに組み込まれます。それに対して ICAPE2 を使用する 2 番目のオプションはユーザー アプリケーション固有であり、発生するイベントに応じてデザインがマルチブートをトリガーします。

ビットストリームに組み込まれる内部プログラム (IPROG) コマンド

ウォーム ブート開始アドレス (WBSTAR) および内部プログラム (IPROG) コマンドは、ビットストリームに組み込むことができます。

ゴールデン ビットストリームは、フラッシュ アドレス 0 に格納されます。アップデート ビットストリームは、ゴールデン ビットストリームの WBSTAR (next_config_addr) レジスタで指定されたフラッシュ アドレスに格納されます。WBSTAR にデフォルト以外のアドレス値が設定されている場合は、IPROG は自動的にビットストリームに組み込まれます。

コンフィギュレーション ロジックは、フラッシュ アドレス 0 に格納されているゴールデン ビットストリーム内のコマンドの実行を正常に開始します。制御は IPROG コマンドに達すると、ゴールデン ビットストリームの WBSTAR レジスタで指定されたフラッシュのアドレス 位置にジャンプし、コンフィギュレーション ロジックはアップデート ストリームを読み込みます。エラー条件が原因でコンフィギュレーション ロジックがアップデート イメージの読み込みに失敗した場合は、フォールバックが実行され、コンフィギュレーション ロジックにより INIT_B および DONE が Low になり、コンフィギュレーション メモリがクリアされ、フラッシュ アドレス 0 でゴールデン ビットストリームが読み込まれてコンフィギュレーション プロセスが再開します。フォールバック中、FPGA は WBSTAR コマンドおよび IPROG コマンドを無視します。詳細は、『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470) [参照 1] の第 7 章「リコンフィギュレーションおよびマルチブート」を参照してください。フラッシュ メモリのコンポーネントおよびコンフィギュレーション手順は、図 2 を参照してください。

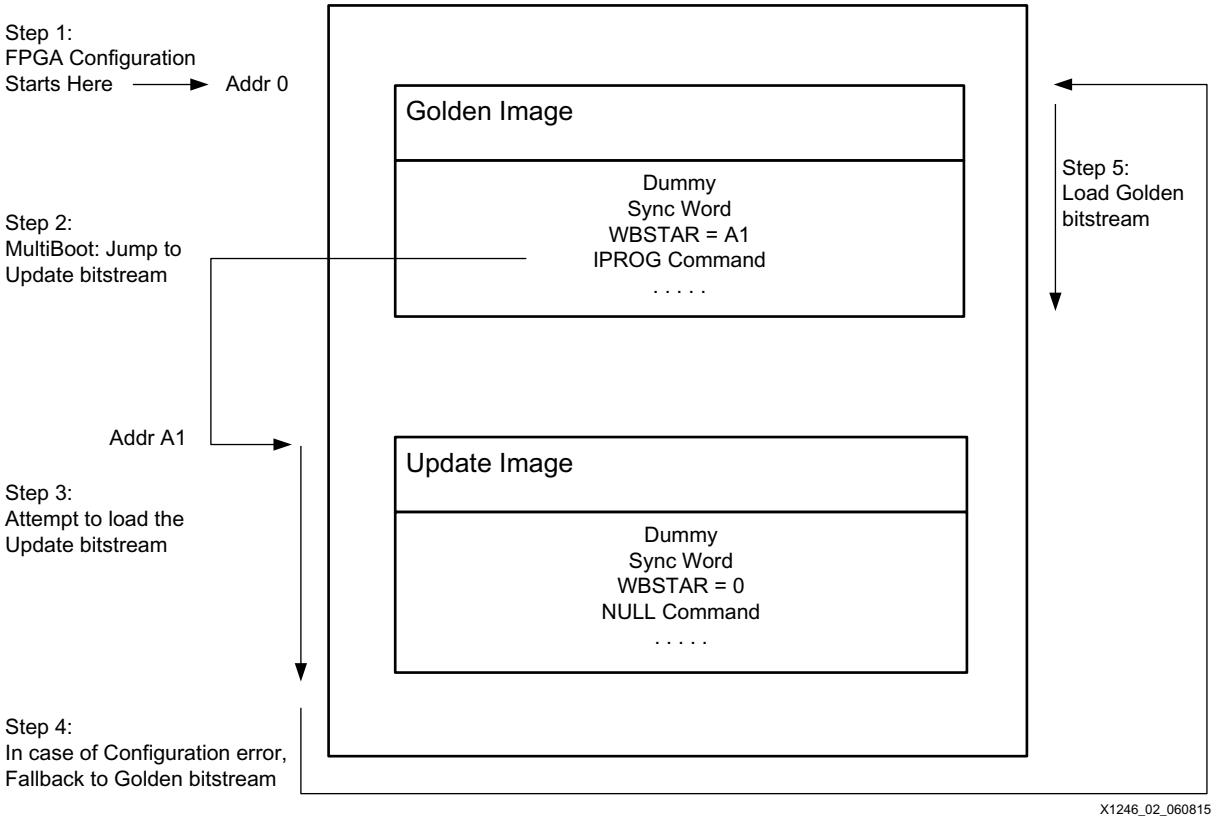

図 2: マルチブートとフォールバックのフラッシュ メモリ コンポーネントおよびコンフィギュレーション手順

FPGA SPI フラッシュ コンフィギュレーションのインターフェイス

図 3 は、7 シリーズ FPGA と SPI フラッシュ間の x1 データ幅での基本的な接続を示しています。読み出しおよびアドレス命令は、MOSI (Master-Out-Slave-In) ピンを介して FPGA から SPI フラッシュへ送られます。その後、データが MISO (Master-In-Slave-Out) ピンを介して SPI フラッシュから戻ります。SCK はクロック ピンで、SS はアクティブ Low のスレーブ セレクト ピンです。

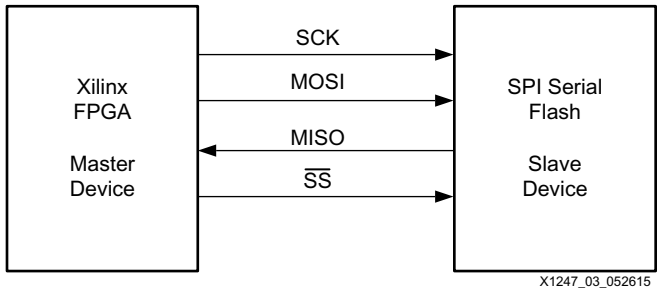

図 3: FPGA フラッシュ インターフェイス

詳細は、『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470) [参照 1] または『SPI フラッシュを使用した 7 シリーズ FPGA のコンフィギュレーション』(XAPP586) [参照 2] を参照してください。

Vivado ツールのフロー

マルチブート アプリケーション用のビットストリームの準備

このセクションでは、マルチブート アプリケーション用のゴールデン ビットストリームとアップデート ビットストリームを作成するために必要なビットストリーム プロパティについて概説します。示されていないビットストリーム オプションには、デフォルトの設定を使用してください。

表 1 に、ゴールデン ビットストリームとアップデート ビットストリームを生成するための基本的なマルチブート ビットストリーム プロパティと、各プロパティの説明を示します。これらのプロパティの詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) [参照 3] を参照してください。

表 1: マルチブート ビットストリーム プロパティ

ビットストリーム プロパティ	値	ゴールデン	アップデート	説明
BITSTREAM.CONFIG.CONFIGFALLBACK	Enable	√	√	コンフィギュレーションが正常に完了しなかった場合、デフォルトのビットストリームの読み込みを有効にします。
BITSTREAM.CONFIG.NEXT_CONFIG_ADDR	Addr A1	√	N/A	次のコンフィギュレーション イメージの開始アドレスを、ウォーム ブート開始アドレス (WBSTAR[28:0] ビット) レジスタに設定します。
BITSTREAM.GENERAL.COMPRESS(1)	Enable	√	√	FPGA ビットストリーム ファイルの圧縮の有効/無効を指定します。

注記:

1. BITSTREAM.GENERAL.COMPRESS はオプションですが、プログラム時間やコンフィギュレーション時間が短縮されるため選択しています。

推奨: Vivado グラフィカル ユーザー インターフェイス (GUI) フローを使用する場合は、Vivado Design Suite でデザインの合成およびインプリメンテーションを実行する前に、ゴールデンおよびアップデート デザインの XDC ファイル内にマルチブート プロパティを含めます。これにより、合成とインプリメンテーションは 1 回のみ実行すればよいので、時間を節約できます。これに対し、プロパティをデザインのインプリメンテーション後に Vivado ビットストリーム設定 GUI で追加した場合は、インプリメンテーション フロー全体を再実行する必要があります。

Vivado でゴールデン デザイン インプリメンテーション用の制約ファイル (.xdc) を開きます。次の内容を制約ファイルにコピー アンド ペーストして、.xdc ファイルに加えた変更を保存します。

```
set_property BITSTREAM.CONFIG.CONFIGFALLBACK ENABLE [current_design]
set_property BITSTREAM.CONFIG.NEXT_CONFIG_ADDR 0x0400000 [current_design]
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.SPI_BUSWIDTH 1 [current_design]
```

注記: BITSTREAM.CONFIG.NEXT_CONFIG_ADDR `0x0400000` は、リファレンス デザインからの例です。このアドレスは、SPI フラッシュ サイズに応じてほかの値に設定できます。

注記: デフォルトでは SPI_BUSWIDTH は x1 です。デフォルトの x1 モードを使用しない場合は、必ずこのプロパティをほかの値に設定してください。

次に、アップデート デザインの制約ファイル (.xdc) を開いて、次のビットストリーム プロパティを制約ファイルに追加し、保存できます。

```
set_property BITSTREAM.CONFIG.CONFIGFALLBACK ENABLE [current_design]
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.SPI_BUSWIDTH 1 [current_design]
```

注記: デフォルトでは SPI_BUSWIDTH は x1 です。デフォルトの x1 モードを使用しない場合は、必ずこのプロパティをほかの値に設定してください。

マルチブート プロパティを追加したら、ゴールデン デザインとアップデート デザインのビットストリームを、`write_bitstream` Tcl コマンドを使用して生成します。ビットストリームを生成するには、プロジェクトでインプリメント済みのデザインを開いておく必要があります。`-verbose` スイッチを `write_bitstream` と共に用いると、使用されているすべてのビットストリーム オプションの要約を出力できます。詳細なヘルプを参照するには、`write_bitstream -help` コマンドを使用してください。

```
write_bitstream  -verbose <file_name>
```

SPI フラッシュ プログラム ファイルの生成

フラッシュ プログラム ファイル (.mcs) を作成するには、`write_cfgmem` Tcl コマンドを使用します。`write_cfgmem` は、FPGA ビットストリーム (.bit) を使用し、SPI フラッシュのプログラムに使用できるフラッシュ ファイル (.mcs) を生成します。

たとえば、次のように 2 つの FPGA ビットストリーム (.bit ファイル) で、1 つのフラッシュ プログラミング ファイル (.mcs) を生成します。

```
write_cfgmem -format mcs -interface SPIX1 -size 16 -loadbit "up 0 <path>/golden.bit up
0x0400000 <path>/update.bit"  <path>/filename.mcs
```

注記: アドレス値 `0x0400000` は、リファレンス デザインで使用しているサンプル値です。ゴールデン イメージに設定された値 Addr A1 (アップデート イメージの開始アドレス) を使用する必要があります (表 1 を参照)。

各 write_cfgmem コマンド オプションの詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) [参照 3] を参照するか、または Vivado で -help コマンドを使用してください。

```
write_cfgmem -help
```

ハードウェア検証

リファレンス デザインには、クイック検証に使用できるフラッシュ プログラミング ファイル (.mcs) が含まれています。

FPGA ブートがゴールデン イメージからなのかアップデート イメージからなのかを確認するには、GPIO (汎用入出力) の LED のステータスを確認します。GPIO の LED は、KC705 ボード上の電源スイッチの近くにあります。

- ゴールデン イメージの場合、GPIO の LED [7:0] は右から左に点滅します。
- アップデート イメージの場合、GPIO の LED [7:0] は左から右に点滅します。

1. KC705 ボード上のスイッチ SW13 で適切なコンフィギュレーション モード (マスター SPI) が選択されていることを確認します (図 4 を参照)。SPI コンフィギュレーション モードの場合、モード ピン M[2:0] は 001 に設定されます。

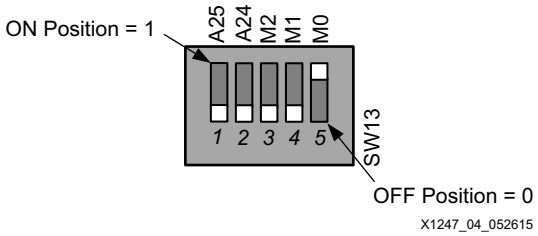

図 4: KC705 SW13: モード ピンの設定

2. JTAG ケーブルを接続し、KC705 ボードの電源を投入します。
3. Vivado ハードウェア マネージャーを起動して、デモ ボードに接続します。次に、前のセクションで生成したフラッシュ イメージ (.mcs) をプログラムするか、またはリファレンス デザインと共に提供される ready_to_download ファイルに含まれている kc705_multiboot_spi.mcs ファイルを使用します。KC705 ボード上で Quad SPI フラッシュをプログラムする手順については、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) [参照 3] を参照してください。
4. SPI フラッシュのプログラムが正常に完了したら、PROGRAM_B ピン (PROG_B ボタン、SW14) をパルスして、FPGA をフラッシュからブートします。別の方法として、ハードウェア マネージャーでデバイスを右クリックして、Vivado の GUI オプション [Boot from Configuration Memory Device] を使用することもできます (図 5 を参照)。

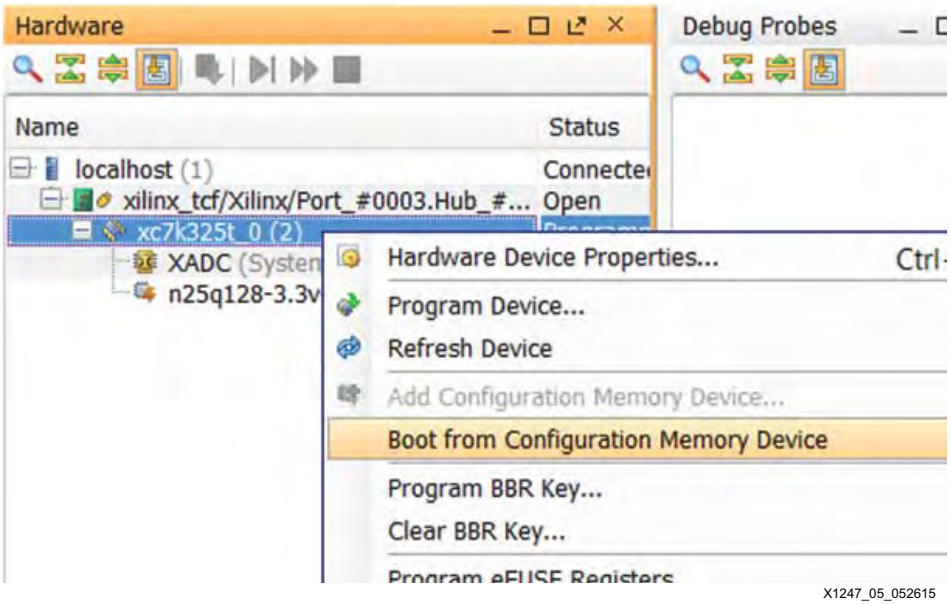

図 5: Vivado ハードウェア マネージャー - コンフィギュレーション メモリ デバイスからのブート

5. ボード上で DONE ピンのステータスを確認します。SPI フラッシュから FPGA のコンフィギュレーションが正常に完了すると、DONE は High になります。
6. KC705 ボード上の GPIO の LED [7:0] が、左から右に連続して点滅していることを確認します。LED[7] > LED[6] > LED[5] > LED[4] > LED[3] > LED[2] > LED[1] > LED[0] という連続の点滅は、アップデート イメージが正常に読み込まれたことを示します。

   さらに、[Hardware Device Properties] から [REGISTER] 下の [BOOT_STATUS] レジスタを調べて、マルチブートを開始した INTERNAL_PROG (IPROG) フラグがアップデート ビットストリームにジャンプすることを確認できます (図 6 を参照)。BOOTSTS レジスタの詳細は、『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470) [参照 1] を参照してください。

   注記: デバイスのプロパティを確認する前に、ウィンドウで FPGA を右クリックして、デバイスをリフレッシュします。

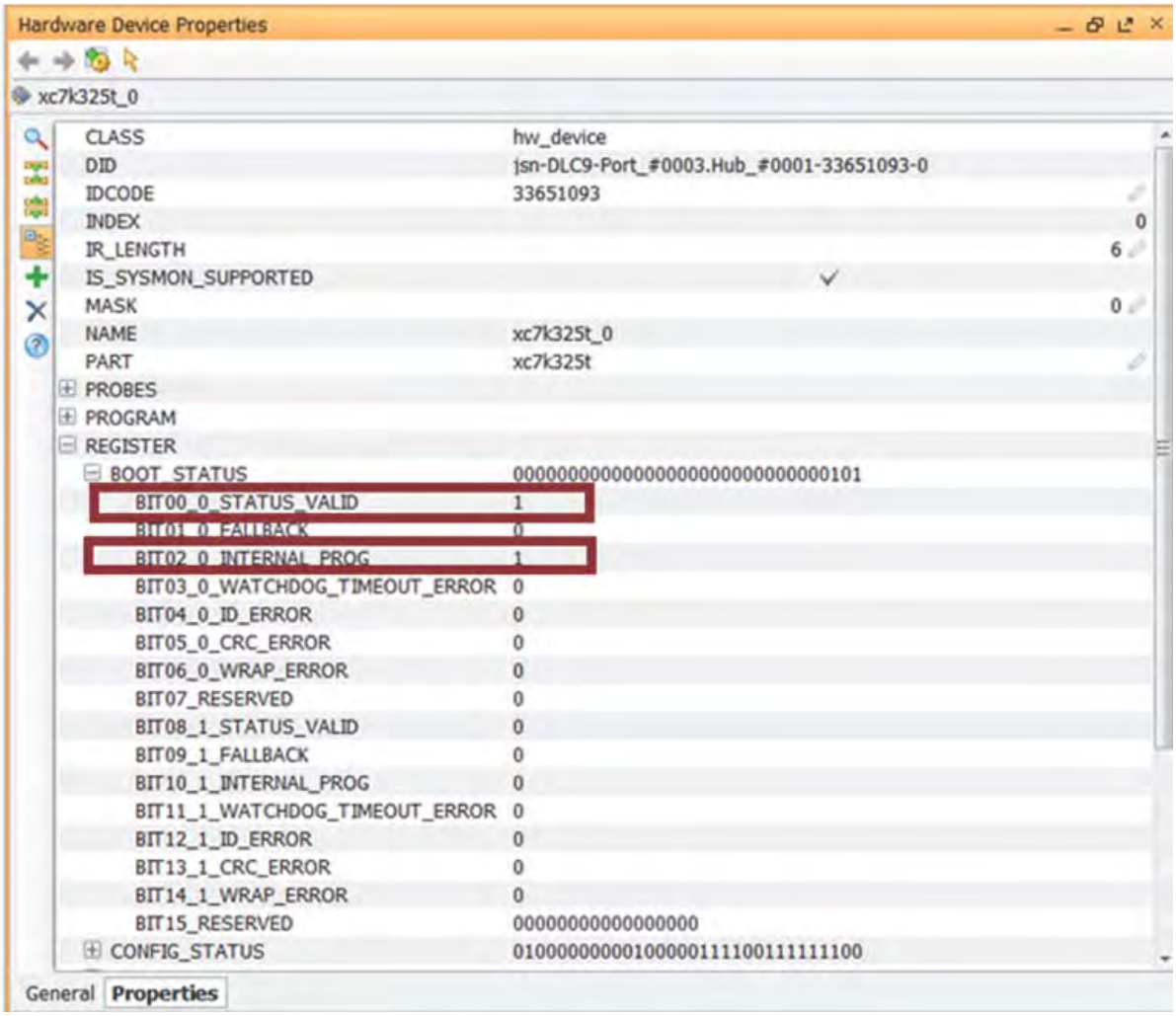

図 6: STATUS_VALID フラグおよび INTERNAL_PROG フラグを示すアップデート イメージ読み込み後の BOOT_STATUS

CRC エラーによりトリガーされるフォールバック

ゴールデン イメージへのフォールバックは、さまざまな方法でトリガーされます。詳細は、UG470 [参照 1] の「リコンフィギュレーションおよびマルチブート」の章を参照してください。

このアプリケーション ノートでは、CRC エラーによってトリガーされるフォールバックを示します。CRC エラーは、アップデート ビットストリームを手動で破損させて誘発できます。RESET CRC コマンドと CRC コマンド間の多くの位置で、ビットを反転させることができます。図 7 と図 8 に例を示します。ビットストリーム形式の詳細は、UG470 [参照 1] の「ビットストリームの構成」セクションを参照してください。

1. アップデート ビットストリーム (.bit) を任意の HEX エディターで開いて、ビットストリームの中ほどで一部のデータ バイトを反転させます。たとえば、図 7 と図 8 で強調表示しているように `00` から `11` にします。

Top_MultiBoot_Module_B.bit

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000540   00 00 00 00 00 00 00 00 00 00 00 00 00 30 00 20  .............0.
00000550   01 00 00 00 12 30 01 40 04 00 00 00 00 00 00 00  .....0.@........
00000560   00 00 00 00 00 00 00 00 00 30 00 20 01 00 00 00  .........0. ....
00000570   13 30 01 40 04 00 00 00 00 00 00 00 00 00 00 00  .0.@............
00000580   00 00 00 00 00 30 00 20 01 00 00 00 14 30 01 40  .....0. .....0.@
00000590   04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000005A0   00 30 00 20 01 00 00 00 15 30 01 40 04 00 00 00  .0. .....0.@....
000005B0   00 00 00 00 00 00 00 00 00 00 00 00 00 30 00 20  .............0.
000005C0   01 00 00 00 16 30 01 40 04 00 00 00 00 00 00 00  .....0.@........
```

X1247_07_052615

図 7: 元のアップデート イメージ (Top_MultiBoot_Module_B.bit)

Top_MultiBoot_Module_B_corrupted.bit

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000540   00 00 00 00 00 00 00 00 00 00 00 00 00 30 00 20  .............0.
00000550   01 00 00 00 12 30 01 40 04 00 00 00 00 00 00 00  .....0.@........
00000560   00 00 00 00 00 00 00 00 00 30 00 20 01 00 00 00  .........0. ....
00000570   13 30 01 40 04 00 00 00 00 00 00 00 00 00 00 00  .0.@............
00000580   00 00 00 00 00 30 00 20 01 00 00 00 14 30 01 40  .....0. .....0.@
00000590   04 00 00 00 00 11 00 00 00 00 00 00 00 00 00 00  ................
000005A0   00 30 00 20 01 00 00 00 15 30 01 40 04 00 00 00  .0. .....0.@....
000005B0   00 00 00 00 00 00 00 00 00 00 00 00 00 30 00 20  .............0.
000005C0   01 00 00 00 16 30 01 40 04 00 00 00 00 00 00 00  .....0.@........
```

X1247_08_060215

図 8: 破損させたアップデート イメージ (Top_MultiBoot_Module_B_corrupted.bit)

2. 破損させたアップデート ビットストリームを保存して、新しいフラッシュ プログラム ファイル (.mcs) を、この破損させたビットストリームで生成します (`write_cfgmem` コマンドの説明は、6 ページの「SPI フラッシュ プログラム ファイルの生成」を参照)。

   ```
   write_cfgmem -format mcs -interface SPIX1 -size 16 -loadbit "up 0 <path>/golden.bit up 0x0400000 <path>/corruputed_update.bit" <path>/filename.mcs
   ```

   注記: CRC は、Reset CRC (RCRC) コマンドと EOS (スタートアップ終了) 前の最終 CRC チェック間のビットストリーム内のデータとコマンドをチェックします。詳細は、UG470 [参照 1] の「ビットストリームの構成」セクションを参照してください。したがって、コンフィギュレーション ロジックで CRC エラーを検出できるのは、ビットストリーム内の RCRC コマンドと CRC コマンド間でデータまたはコマンドが破損している場合のみです。

3. 新しいフラッシュ イメージ ファイル (.mcs) を作成した場合は、7 ページの「ハードウェア検証」の手順 1 ～ 6 を繰り返して、破損させたイメージで SPI フラッシュをプログラムします。

   手順 3 では、リファレンス デザインの ready_to_download ディレクトリ内に含まれている `kc705_multiboot_corrupted_spi.mcs` ファイルを検証用に使用できます。

4. フォールバック操作が正常に実行され、ゴールデン イメージが FPGA に読み込まれたことは GPIO の LED [7:0] で確認できます。LED は、LED[0] → LED[1] → LED[2] → LED[3] → LED[4] → LED[5] → LED[6] → LED[7] の順で右から左に連続的に点滅します。

また、Vivado ツール内で BOOT_STATUS レジスタを表示して、デザインのフォールバックが正常に完了していることを確認できます。

a. BIT10_1 フラグおよび BIT13_1 フラグは、IPROG が実行され、CRC_ERROR が検出されたことを示しています。

b. BIT01_0 フラグは、フォールバック ビットストリームが正常に読み込まれたことを示しています。

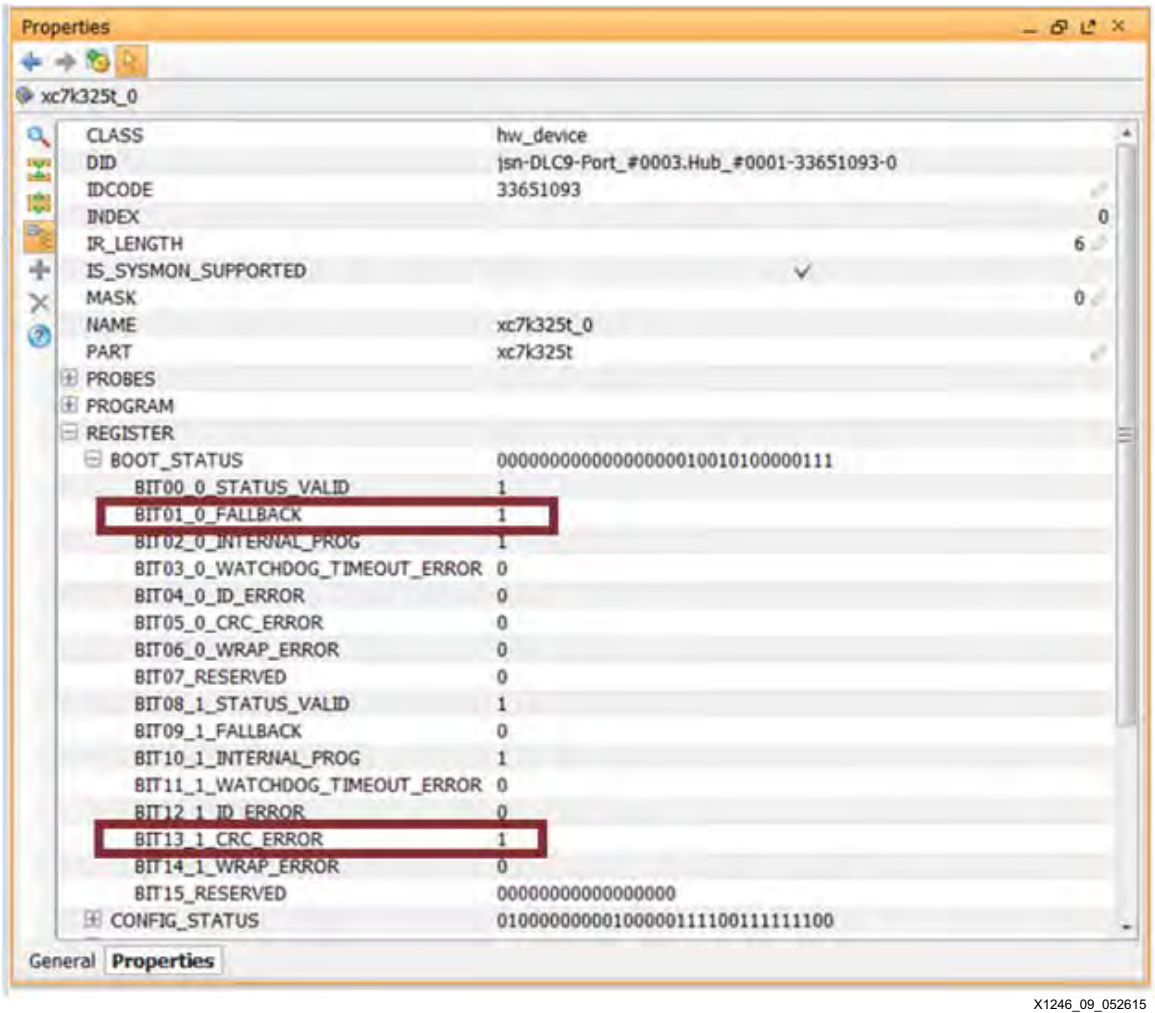

図 9: フォールバックおよび CRC フラグを示す BOOT_STATUS レジスタ

デバッグおよびチェックリスト

ファイル生成

1. Vivado グラフィカル ユーザー インターフェイス (GUI) を使用してマルチブート プロパティを設定する場合は、マルチブート プロパティを適用する前に、ゴールデン デザインとアップデート デザインが正常に合成およびインプリメントされていることを確認します。
2. ゴールデン イメージとアップデート イメージの両方に対して、マルチブート ビットストリーム プロパティが適切に設定されていることを確認します。
3. ConfigRate オプションが、ターゲット フラッシュによりサポートされている最大周波数を超えていないことを確認します。UG470 [参照 1] の「最大コンフィギュレーション クロック周波数の決定」セクションを参照してください。
4. MCS ファイルの生成時に、データ バス幅の `-interface` オプションが正しく設定されていることを確認します。
5. アップデート イメージの開始アドレスが、ゴールデン デザインの NEXT_CONFIG_ADDR プロパティに指定したアドレスと同じであることを確認します。これを下位アドレスに設定すると、FPGA プログラム時間は長くなります。

コンフィギュレーション

1. マルチブート プロパティを追加しなくても、デザインが予期したとおりに動作することを確認します。これは、考えられるすべてのプログラム エラーの問題をデバッグするための有効な確認方法であり、問題がデザイン自体にあるのかマルチブートの設定にあるのか根本原因を判断するのに役立ちます。
2. 新しいフラッシュ プログラム ファイル (.mcs) でプログラムする前に、フラッシュ デバイスを完全に消去します。ブランク チェックを実行して、消去動作を確認します。
3. 追加のデバッグ情報を得るために、BOOT_STATUS と CONFIG_STATUS の レジスタ データをキャプチャします。デバイス プロパティをキャプチャする前に、デバイスをリフレッシュします。
4. フラッシュのゴールデン イメージ領域が常時保護され、変更されることがないようにします。アップデートおよび変更できるのはアップデート イメージ領域のみです

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	
開発元	ザイリンクス
ターゲット デバイス	7 シリーズ 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、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	なし
シミュレーション	
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	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	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	なし
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® Design Suite バージョン 2015.1
使用したインプリメンテーション ソフトウェア ツール/バージョン	Vivado Design Suite バージョン 2015.1
スタティック タイミング解析の実施	なし
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 ボードと 128Mb Micron Quad-SPI フラッシュ

まとめ

このアプリケーション ノートは、7 シリーズ FPGA でマルチブート機能を有効にして、SPI フラッシュに格納されているさまざまなビットストリームで FPGA をコンフィギュレーションする方法を示しています。マルチブート機能とフォールバック機能を示すための詳細なリファレンス デザインとクイック検証ファイルが提供されています。

付録 A: 高度なアプリケーション

タイムアウト エラーを使用したフォールバック

次のエラーによってフォールバックがトリガーされることがあります。

- CRC エラー
- IDCODE エラー
- ウォッチドッグ タイマーのタイムアウト エラー

CRC エラーでトリガーされるフォールバックは、7 ページの「ハードウェア検証」で説明されています。フォールバック エラーの詳細は、『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470) [参照 1] を参照してください。

信頼性の高いインシステム アップデート ソリューションをインプリメントするには、フラッシュ メモリ内でゴールデン イメージは常時保護され、アップデート イメージのみが変更されるようにする必要があります。

マルチブート動作時に、ゴールデン ビットストリームに組み込まれた IPROG コマンドは、WBSTAR レジスタで指定されたアドレス位置へジャンプし、コンフィギュレーション ロジックは、ビットストリームを読み込むための次の同期ワードを検索します。SYNC ワードが検出されると、コンフィギュレーション ロジックは、FPGA をコンフィギュレーションするために同期ワードに続くコマンドとデータを待機します。next_config_addr の位置にアップデート イメージがないか、またはアップデート イメージ内の同期ワードが破損している場合、コンフィギュレーション ロジックはフラッシュ メモリ全体をスキャンして、有効な同期ワードを検索します。

シリアル フラッシュ内でアップデート領域が破損しているというシナリオでは、ゴールデン領域へのフォールバックをトリガーするために、ウォッチドッグ タイマーを設定する手段が必要になります。詳細は、UG470 [参照 1] の「ウォッチドッグ タイマー」セクションを参照してください。

ウォッチドッグ タイマーの設定

1 つの方法として、BITSTREAM.CONFIG.TIMER_CFG プロパティをゴールデン イメージとアップデート イメージの両方で使用して、ビットストリームでタイマー値を有効にできます。ゴールデン イメージとアップデート イメージの XDC ファイル内で次のビットストリームの Timer プロパティを、5 ページの表 1 にリストされているプロパティと共に設定します。

```
set_property BITSTREAM.CONFIG.TIMER_CFG <Timer Value> [current_design]
```

7 シリーズ FPGA の場合、ウォッチドッグ タイマーは専用内部クロック CFGMCLK を分周したものを使用します。タイマー値には、スタートアップが完了するまでの FPGA コンフィギュレーション全体の時間を十分にカバーできる値を指定する必要があります。スタートアップでの待機時間 (DCI の一致、MMCM のロック、DONE) も含めるよう注意してください。つまり、これらの条件に基づき正しいタイマー値を計算するには、いくらかの労力を必要とします。

このアプリケーション ノートでは、ビットストリーム プロパティを使用する代わりに、独立したバリア イメージまたはタイマー イメージを使用してウォッチドッグ タイマーを有効にするソリューションを提供しています。バリア イメージやタイマー イメージを使用すると、TIMER_CFG ビットストリーム プロパティのタイマー値を計算する必要がなくなります。

リファレンス デザインに付属する Tcl スクリプトを使用して、バリア イメージと、そのイメージをゴールデン イメージやアップデート イメージと共にフラッシュ メモリ内に配置するためのアドレス マップを生成できます。図 10 では、フラッシュ メモリ コンポーネントと、マルチブート アプリケーションでタイマー イメージをインプリメントするコンフィギュレーション手順を示しています。

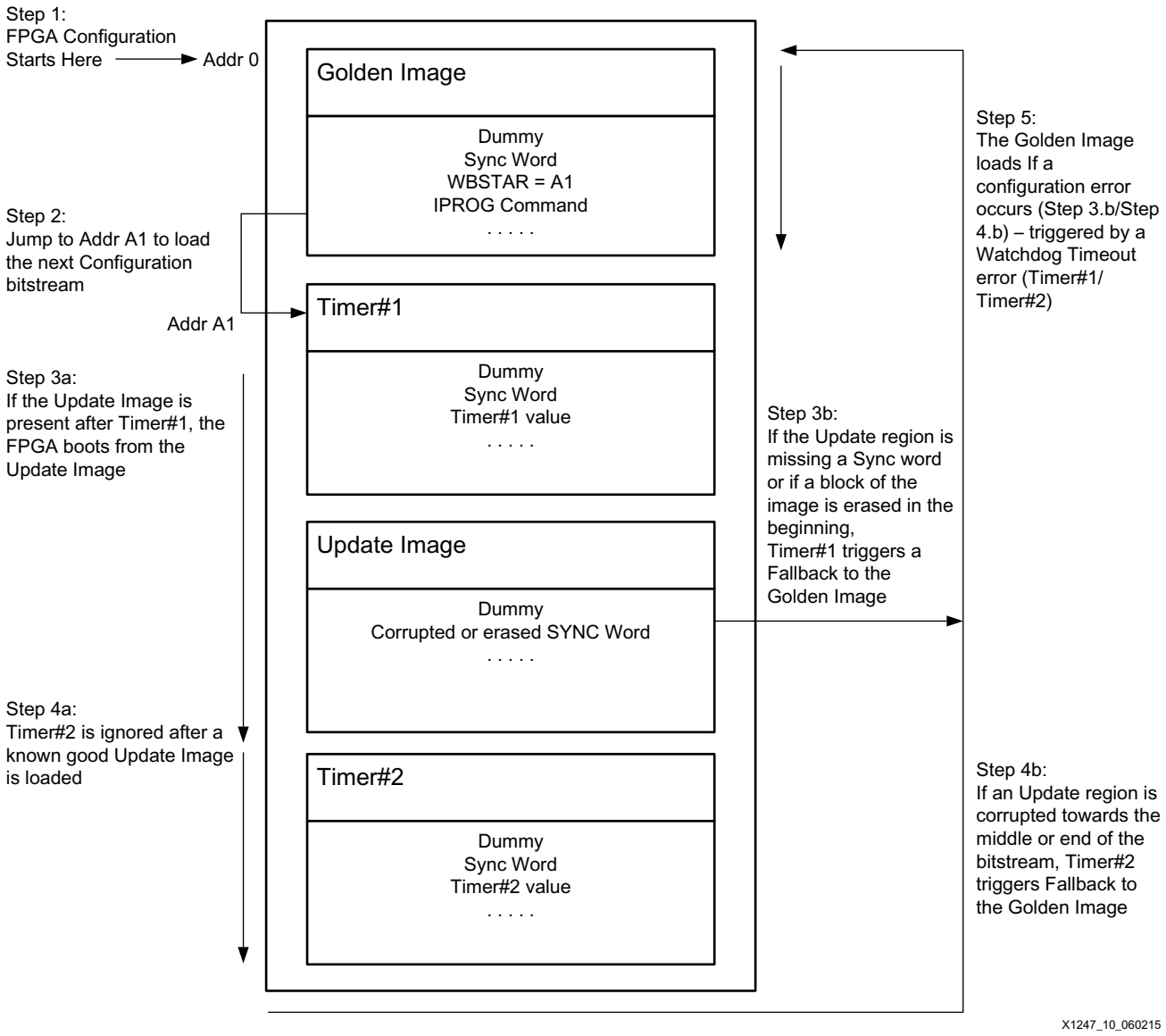

図 10: マルチブート、フォールバック、バリアのフラッシュ メモリ コンポーネント、およびコンフィギュレーション手順

重要: SPI フラッシュ メモリ内のゴールデン、timer1、および timer2 領域は、常時保護しておく必要があります。変更できるのはアップデート イメージ領域のみです。

図 10 に概要が示されている、バリア イメージを使用するデザイン インプリメンテーションの基本プロセスは、次のとおりです。

1. コンフィギュレーション ロジックは、フラッシュのアドレス 0 に格納されている、ゴールデン ビットストリーム内のコマンドの実行を開始します。ゴールデン ビットストリームに組み込まれている IPROG コマンドは、ゴールデンの WBSTAR レジスタに格納されているアドレス位置へのジャンプを制御します。この例では、WBSTAR は timer1 の開始アドレスを指します。

2. アップデート イメージの前の timer1 またはバリア イメージでは、コンフィギュレーション ロジックを検証済みの同期ワードに同期するために役立つ短時間のタイマーが有効になっており、破損したアップデート イメージにより正常な同期が損なわれる可能性が小さくなります。
3. a. 検証済みのアップデート イメージが timer1 の後に存在します。この場合、アップデート イメージの読み込みは正常です。

   b. アップデート イメージの先頭で破損があるか、データのセクターが消去されていて同期ワード (AA995566) がありません。コンフィギュレーション ロジックは、同期ワード (AA995566) を確認後にビットストリームの確認を開始します。同期ワードが破損しているかまたは消去されている場合、データまたはコマンドは無視され、コンフィギュレーション ロジックはフラッシュ全体のスキャンを続行して有効な同期ワードを検索します。このシナリオでは、timer1 に設定されているタイマー値がウォッチドッグ タイムアウト エラーによってフォールバックをトリガーし、フェイルセーフ イメージまたはゴールデン イメージが読み込まれます (手順 5 に続きます)。
4. a. 検証済みのアップデート イメージが timer1 の後に存在する場合、timer2 は無視されます。

   b. アップデート領域が破損しているか、アップデート イメージの終わり部分でデータのセクターが消去されています。このシナリオでは、コンフィギュレーション ロジックはコンフィギュレーションを終了するための EOS (スタートアップ終了) を確認できません。timer2 が有効になり、アドレス 0 （ゴールデン イメージの位置）へのフォールバックがウォッチドッグ タイムアウト エラーによってトリガーされます。
5. タイムアウト エラーによりトリガーされるフォールバックの結果として、ゴールデン フェイルセーフ イメージが読み込まれます。

バリアの構成

スクリプトを使用して作成されるバリアまたはタイマー イメージは、コンフィギュレーション ロジックを同期し、コンフィギュレーション タイマー値を設定するための一連のコマンドです。表 3 に、TIMER レジスタを有効にするバリア イメージに含まれている基本コマンドを示します。

表 3: バリア イメージの構成

FFFFFFFF	ダミー パッド ワード
000000BB	バス幅自動検出、ワード 1
11220044	バス幅自動検出、ワード 2
FFFFFFFF	ダミー パッド ワード
FFFFFFFF	ダミー パッド ワード
AA995566	同期ワード
20000000	NOOP
20000000	NOOP
30022001	パケット タイプ 1 コマンド: TIMER レジスタに書き込みます。
xxxxxxxx	TIMER レジスタ値。TIMER_CFG を有効にし、TIMER 値を設定します。
20000000	NOOP
20000000	NOOP

アドレス テーブルとバリア イメージを作成する Tcl スクリプト

Tcl スクリプト `multiboot_address_table.tcl` が、ダウンロード可能な xapp1247.zip ファイルで提供されています。

アドレス テーブルとバリア イメージを作成するには、リファレンス デザイン ファイルを任意のディレクトリに解凍します。Vivado コマンド プロンプトを開いて、ディレクトリを、リファレンス デザイン ディレクトリ内のスクリプトの場所を指すように変更します。リファレンス デザインには、`multiboot_address_table.tcl` スクリプトが格納された multiboot_address_table フォルダーがあります。

次の構文を使用して、スクリプトを実行します。

```
>> tclsh multiboot_address_table.tcl <flash_type> <data_width> <freq_mhz>
< flash_size_mbit> <bitstream_size>
```

説明 :

flash_type: テストされるフラッシュ タイプは spi、bpi

data_width: フラッシュのデータ幅 = 1、2、4、16

freq_mhz: CCLK の周波数 = ConfigRate の設定値

flash_size_mbit: フラッシュ デバイスのサイズ (Mb 単位)

bitstream_size: ビットストリームのサイズ (バイト単位)

注記: ビットストリームのサイズについては、UG470 [参照 1] を参照してください。圧縮が有効である場合は、圧縮後のビットストリーム サイズを入力します。圧縮されたビットストリームの場合は、後続のビルドのサイズが大きく異なる可能性があり、このスクリプトを再実行する必要があることに注意してください。

Tcl スクリプトを実行すると、timer1 と timer2 の 2 つのタイマー イメージが生成されます。これらのファイルは、スクリプトを実行したのと同じディレクトリ内に格納されます。このスクリプトはさらに、シリアル フラッシュ内でのファイルの場所を示すアドレス マップの位置と、4 つのファイルを結合する `write_cfgmem` コマンドも出力します。次に示すのは、スクリプトの実行後にWindows コマンド プロンプトに表示されるサンプル出力です。

```
c:\xapp1247\multiboot_address_table>tclsh multiboot_address_table.tcl spi 1 3 12 8
1132000
Flash type           : SPI
Flash width (bits)   : 1
CCLK frequency (MHz) : 3
Flash density (Mbits) : 128
Bitstream size (B)   : 1132000

Writing Timer: timer1.bin
Writing Timer: timer2.bin

Golden bitstream address :  0x00000000
Timer1 image address     : 0x0013FC00
Multiboot image address  :  0x00140000
Timer2 image address     : 0x00280000

write_cfgmem command:
write_cfgmem -format mcs -size 16 -interface SPIx1 -loadbit “up 0x00000000 <golden>
up 0x00140000 <multiboot>“  -loaddata “up 0x0013FC00 timer1.bin up 0x00280000
timer2.bin“ <output_mcs>
```

ゴールデン デザインのインプリメンテーションで、NEXT_CONFIG_ADDR の設定 (5 ページの表 1 の Addr A1) が、スクリプトの実行後に表示される timer1 イメージの開始アドレス出力を指していることを確認します。たとえば、前のページに示すサンプルのアドレス マップの場合、ゴールデン イメージの WBSTAR レジスタには、ビットストリーム プロパティ NEXT_CONFIG_ADDR で設定される `0x0013FC00` が格納されます。

```
set_property BITSTREAM.CONFIG.NEXT_CONFIG_ADDR 0x0013FC00 [current_design]
```

マルチブートを有効にするアップデート デザインの制約は、未変更のままです。

サンプル出力で、timer1 と timer2 の .bin ファイルには `write_cfgmem` コマンドと共に -loaddata スイッチが使用されており、ゴールデンおよびアップデートの .bit ファイルには -loadbit スイッチが使用されていることに注意してください。シリアル フラッシュ プログラム ファイルの生成中は、各ファイルの開始アドレス シーケンスに注意を払ってください。

`write_cfgmem` コマンドの各オプションの詳細は、Vivado ツールで次のように `-help` コマンドを実行して確認できます。

```
write_cfgmem -help
```

参考資料

注記: 日本語版のバージョンは、英語版より古い場合があります。

1. 『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470: 英語版、日本語版)
2. 『SPI フラッシュを使用した 7 シリーズ FPGA のコンフィギュレーション』(XAPP586: 英語版、日本語版)
3. 『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908: 英語版、日本語版)
4. 『Kintex-7 FPGA データシート: DC 特性およびスイッチ特性』(DS182: 英語版、日本語版)
5. 『Kintex-7 FPGA 用 KC705 評価ボード ユーザー ガイド』(UG810)
6. Kintex-7 FPGA KC705 評価キット ウェブサイト

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
2015 年 8 月 13 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。

© Copyright 2016 Xilinx, Inc. Xilinx、Xilinx のロゴ、Artix、ISE、Kintex、Spartan、Virtex、Vivado、Zynq、およびこの文書に含まれるその他の指定されたブランドは、米国およびその他各国のザイリンクス社の商標です。すべてのその他の商標は、それぞれの保有者に帰属します。