XAPP1260 (v1.0) 2015 年 7 月 8 日

# デバイス プログラマを使用した eFUSE のプログラム

著者 : Randal Kuramoto

## 概要

このアプリケーション ノートは、デバイス プログラマで 7 シリーズ FPGA の eFUSE ビットをプログラムする際のガイドラインを提供します。デバイス プログラマは、次の場合に効果的です。

- 量産製品のプログラムする
- ボード製造サイトでアセンブリを行う前に、機密データをデバイス内の eFUSE にプログラムする
- プログラムされたデバイスの DNA (固有のシリアル番号) 情報を記録する

**重要** : 7 シリーズ FPGA eFUSE ビットは OTP (ワンタイム プログラマブル) であり、一度プログラムすると変更できません。

eFUSE ビットを正しくプログラムするには、正確な設定と検証方法を指定することが重要です。このアプリケーション ノートは、eFUSE ビットの設定についてのガイドラインを提供します。デバイス プログラマ用の暗号化された検証ビットストリームを生成するためのリファレンス デザインも提供されています。

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。

# プログラム可能な eFUSE の概要

図 1 に、eFUSE ビットをプログラム/検証する際のデバイス プログラマの入力、および 7 シリーズ FPGA における eFUSE ビットの使用方法を示します。

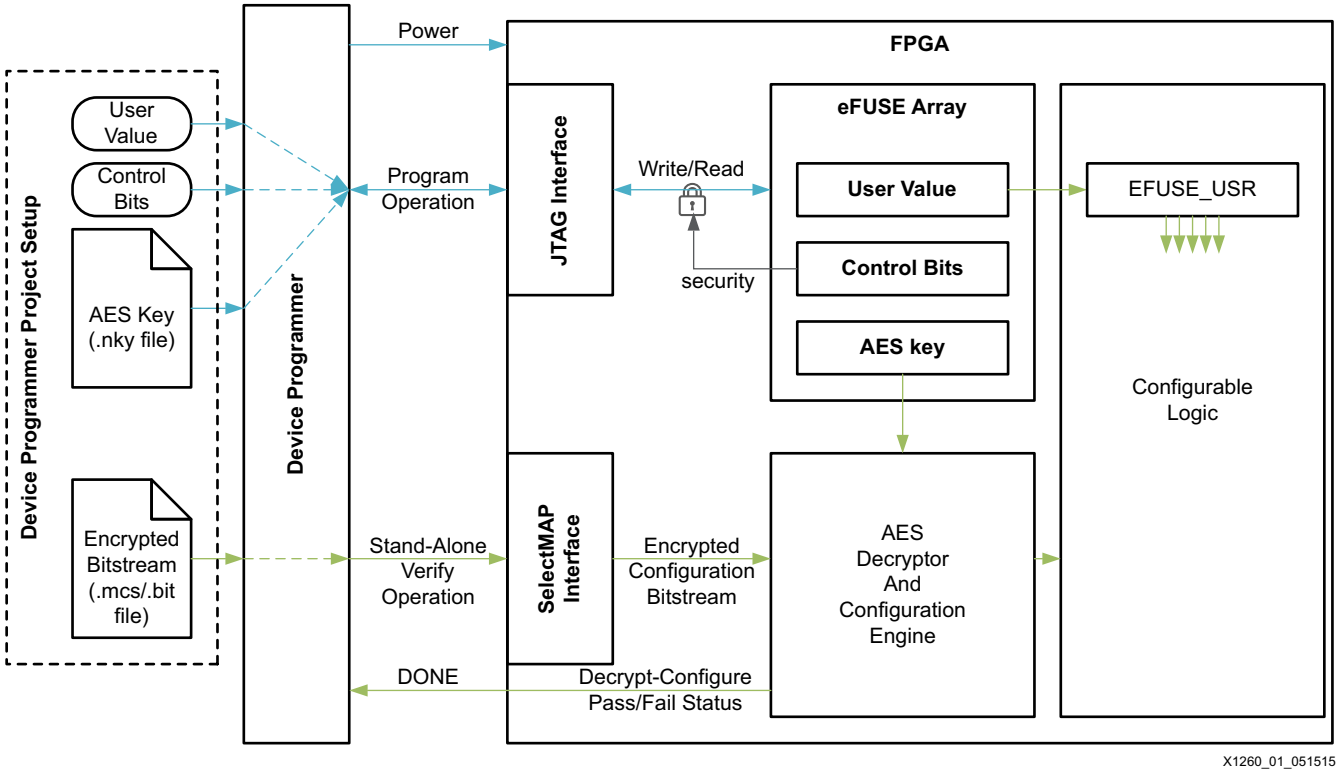



図 1 : eFUSE アレイおよび関連機能のブロック図

7 シリーズ FPGA では、ビットストリームを暗号化することで、複製デザインの不正使用を防ぐことができます。7 シリーズ FPGA は、NIST (National Institute of Standards and Technology) によって FIPS PUB 197 (Federal Information Processing Standards Publication 197) として公表されている AES (advanced encryption standard) を使用しています。AES は、ビットストリームの暗号化と復号化に同じ秘密キーを用いる対称キー アルゴリズムを採用します。Vivado® デザイン ツールは、ビットストリーム ファイルへの書き込み時にこのキーを使用してビットストリームを暗号化します。FPGA は、コンフィギュレーション時にこのキーを使用してビットストリームを復号化します。FPGA での AES 復号化用に、バックアップ バッテリ付きの RAM 内、または FPGA 内の不揮発性 OTP (ワンタイム プログラマブル) eFUSE ビットに AES キーを格納できます。eFUSE ビットは、デバイス プログラマでプログラム可能です。バックアップ バッテリ付きの RAM は、プログラマからデバイスが切断されると設定値を失ってしまうため、事実上はデバイス プログラマではプログラムできません。

キーの格納に不揮発性の eFUSE ビットを使用するメリットは次のとおりです。

- FPGA の電源が切断された際にキー値を保持するためのバッテリーが不要
- バッテリーを使用する場合よりも BOM コストが低い
- ボード製造サイトでアセンブリを行う前に、デバイス プログラマを使用してデバイスの安全な場所に eFUSE をプログラムできる

不揮発性の 7 シリーズ FPGA eFUSE ビットは、32 ビットの「ユーザー値」を格納することも可能であり、FPGA のコンフィギャラブル ロジック デザインは EFUSE_USR プリミティブからこれらの値を読み出すことができます。また、FPGA JTAG ポートを介して、外部デバイスで読み出すことも可能です。

追加の eFUSE ビットを使用することで、eFUSE の読み出し保護や書き込み保護など FPGA のセキュリティ設定を制御し、より厳しく読み出し/書き込みを保護できます。

**重要** : eFUSE をプログラムしている間、秘密 AES キーは常に読み出し禁止となります。

ザイリンクスから出荷される時点では、すべてのユーザー プログラマブル eFUSE ビットが 0 にプログラムされています。FPGA JTAG ポートを介して選択された eFUSE ビットは 1 にプログラムされます。eFUSE をプログラムする際には、プログラムした eFUSE ビットの検証と選択したセキュリティ オプションの設定を併せて行う必要があります。eFUSE のプログラムが完了すると eFUSE ビットの読み出しが禁止され、これらのビットは直接検証できなくなります。

**推奨** : デバイスをプログラムする場合は、デバイス プログラマ用に作成され、照合キーで暗号化されたサンプル ビットストリームを使用して、デバイス プログラマで正しい eFUSE キーがプログラムされているかを検証することを推奨しています。

また、プログラム済みデバイスのインベントリを再確認するためにデバイス プログラマをスタンドアロン検証モードで使用する場合、このサンプル ビットストリームを用い、照合キーでプログラムされたデバイスを特定できます。スタンドアロン検証動作時、サンプル ビットストリームは、7 シリーズ FPGA SelectMAP コンフィギュレーション インターフェイスに送られます。

7 シリーズ FPGA eFUSE の機能およびデバイス コンフィギュレーションの詳細は、『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。

# eFUSE プログラムにデバイス プログラマを使用

デバイス プログラマの使用用途は次のとおりです。

- FPGA 内の eFUSE ビットをプログラムし、FPGA の DNA 値を記録する
- FPGA が特定 AES キーであらかじめプログラムされていることをスタンドアロンで検証する

## eFUSE のプログラム

このセクションでは、デバイス プログラマを使用して 7 シリーズ FPGA 内の eFUSE をプログラムする手順を示します。

1. プログラミング マネージャーでデバイス プログラマのプロジェクトを作成します。必要事項は次のとおりです。
   a. 入力ファイルの詳細
      - NKY ファイル (AES キーを含む )
      - 暗号化された検証ビットストリーム ファイル
   b. プログラム可能なオプション設定の定義
      - ユーザー指定の 32 ビット値
      - eFUSE 制御ビット (読み出し保護、書き込み保護、およびその他のセキュリティ機能)
2. デバイス プログラマの演算子によってデバイスがプログラムされて、次の動作が実行されます。
   a. デバイスを特定してデバイス DNA を記録する
   b. デバイスの eFUSE ビットをプログラムおよび検証する
   c. 暗号化された検証ビットストリームをデバイスにダウンロードして、二次的な検証を行う

eFUSE のプログラムが完了すると、通常、デバイス内の eFUSE は保護された状態となります。つまり、AES キーが保護されてデバイスから読み出すことはできなくなります。

プログラム完了後、デバイス プログラマは、デバイス内の eFUSE が保護された状態であっても、すでにプログラムされている 7 シリーズ FPGA の eFUSE に正しい AES キーがプログラムされているかを確認/検証するために、スタンドアロン動作を実行できます。このスタンドアロン検証動作では、デバイス プログラマが暗号化されたビットストリームをデバイスにダウンロードします。この暗号化された検証ビットストリームでデバイスが正常にコンフィギュレーションされると、そのデバイスに照合キーが含まれていることが確認できます。

# デバイス プログラマの準備

デバイス プログラマを使用してプログラムを開始する前に、次の準備が必要です。

- デバイス プログラマのサイト/サービスを確立する
- AES キーを含む NKY ファイルの位置を確認する
- 各 eFUSE 制御ビット オプションのセキュリティ設定を明確にする
- 提供されたリファレンス デザインをベースに、NKY ファイルを使用して暗号化された検証ビットストリームを生成する
- デバイス サンプルとデバイス プログラマの設定およびファイルをデバイス プログラマのサイトに提供する

## デバイス プログラマのサイト/サービスの確立

デバイス プログラミング サービスの詳細は、ザイリンクス販売代理店アヴネット社へお問い合わせください。また、ザイリンクス プログラマのオプションについては、FAE へお問い合わせください。

**重要** : デバイス プログラマのサイトまたはサービスはできるだけ早期に確立することが重要です。事前構築されていないデバイス/パッケージのデバイス プログラマ ソケットを開発するには、数週間かかる場合があります。

### デバイス プログラマの設定および入力ファイルを定義する

デバイス プログラマには、明確な設定と必要なファイルを提供する必要があります。デバイス プログラマの設定と入力ファイルのチェックリストおよびそれらの説明は、表 1 を参照してください。最初の試験デバイス向けにデバイス サンプルを備えたデバイス プログラミング サイトに、必要なファイルを含むすべての設定 (表 1) を提供します。

**表 1 : デバイス プログラマの設定**

| プログラムされた場合の機能の説明 | 設定の種類 | デバイス プログラマの入力ファイルまたは設定 (設定値とファイル名をここに記入) |
|---|---|---|
| **256 ビットの AES キー**<br>このキーで、FPGA コンフィギュレーション用の暗号化されたビットストリームを復号できます。 | NKY ファイル名<br>NKY ファイルには、256 ビットの AES キー値が含まれます。 | |
| **32 ビットのユーザー定義値**<br>FPGA デザイン内の EFUSE_US プリミティブから、または JTAG (R_EN_B_User がプログラムされていない場合) からアクセスできるオプションのユーザー定義のデバイス識別子の値です。<br>使用しない場合は、空のままにしておきます。 | 8 文字の 16 進数値 `[31:0]` | |

表 1 : デバイス プログラマの設定 (続き)

| プログラムされた場合の機能の説明 | 設定の種類 | デバイス プログラマの入力<br>ファイルまたは設定<br>(設定値とファイル名をここに記入) |
|---|---|---|
| **CFG_AES_Only** (FUSE_CNTL[0])<br>eFUSE に格納されている AES キーの使用を強制し、照合キーがないビットストリームで FPGA をプログラムすることを防ぎます。(このビットがプログラムされていない場合、FPGA は暗号化されていないビットストリーム、またはバックアップ バッテリ付きの RAM に格納されたキーの値で暗号化されたビットストリームを使用して FPGA をコンフィギュレーションできます。)<br><br>**注意** : このビットがプログラムされている場合、エラー解析のためにザイリンクス テストビットストリームをデバイスにロード できないため、RMA (Return Material Authorization) による返却は受け付けられません。 | Yes = 1 にプログラム<br>No = プログラムしない (0)<br><br>通常設定 = No | |
| **AES_Exclusive** (FUSE_CNTL[1])<br>外部コンフィギュレーション インターフェイスからのパーシャル リコンフィギュレーションを無効にします。ただし、ICAPE2 を使用することでパーシャル リコンフィギュレーションが可能になります。<br><br>**注意** : このビットがプログラムされている場合、RMA (Return Material Authorization) による返却は、デバイス解析およびデバッグの面で制限があります。オプションの推奨方法として、ビットストリームを生成する前に Vivado デザイン ツールで BITSTREAM.READBACK.SECURITY プロパティを Level2 に設定することで、同じセキュリティ機能を実現できます。 | Yes = 1 にプログラム<br>No = プログラムしない (0)<br><br>通常設定 = No | |
| **W_EN_B_Key_User** (FUSE_CNTL[2])<br>AES キーと FUSE_USER のプログラムを無効にします。 | Yes = 1 にプログラム<br>No = プログラムしない (0)<br><br>通常設定 = Yes | |
| **R_EN_B_Key** (FUSE_CNTL[3])<br>AES キーの読み出し、および AES キーと 32 ビットのユーザー定義値のプログラムを無効にします。<br><br>**重要** : JTAG ポートを介して秘密の AES キーが読み出されることを防ぐには、R_EN_B_Key ビットをプログラムする必要があります。 | Yes = 1 にプログラム<br>No = プログラムしない (0)<br><br>AES キーがプログラムされている場合は、REQUIRED = Yes | Yes<br>AES キーをプログラムしない限りは変更しないでください。 |

表 1 : デバイス プログラマの設定 (続き)

| プログラムされた場合の機能の説明 | 設定の種類 | デバイス プログラマの入力<br>ファイルまたは設定<br>(設定値とファイル名をここに記入) |
|---|---|---|
| **R_EN_B_User** (FUSE_CNTL[4])<br>JTAG を介す 32 ビットのユーザー定義値の読み出し、および AES キーとユーザー定義値のプログラムを無効にします。JTAG ポートを介すユーザー定義値の読み出しは無効になりますが、EFUSE_USR コンポーネントを使用するユーザー定義値の読み出しは無効になりません。 | Yes = 1 にプログラム<br>No = プログラムしない (0)<br><br>通常設定 = No | |
| **W_EN_B_Cntl** (FUSE_CNTL[5])<br>制御ビットのプログラムを無効にします。 | Yes = 1 にプログラム<br>No = プログラムしない (0)<br><br>通常設定 = Yes | |
| **暗号化された検証ビットストリーム**<br>ビットストリームは、NKY ファイルにある AES キーを使用して暗号化され、MCS フォーマットで提供されます。<br><br>**注意** : 実際のデザインは使用しないでください。ビットストリーム デザインの制約については、「暗号化された検証ビットストリーム」を参照してください。 | MSC ファイル名 | |

## AES キーと NKY ファイル

通常、ビットストリームの暗号化には、256 ビットの AES キーが定義されて Vivado デザイン ツールに与えられます。`write_bitstream` コマンドが実行されると、暗号化されたビットストリームが生成されます。Vivado デザイン ツールは、AES キーの値を NKY ファイルに保存します。この NKY ファイルをデバイス プログラマの入力として保持し、FPGA eFUSE に AES キー値をプログラムする場合、さらには `write_bitstream` コマンドの連続実行において同じキーを使用してビットストリームを暗号化 (または `write_bitstream` の実行で検証ビットストリームを暗号化) する場合に使用します。

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 2] の「暗号化済みビットストリームの生成」および『暗号化を使用して 7 シリーズ FPGA ビット スト リームを保護』(XAPP1239) [参照 3] を参照してください。Vivado デザイン ツールの暗号化ビットストリーム フローで、暗号化キーの格納場所として「eFUSE」が選択されていることを確認してください。

NKY ファイルの例 :

```
Device xc7a200tfbg676;
Key 0 0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef;
Key StartCBC 9966d617cc42731291374e58239a823e;
Key HMAC cd552f296733ac7abec573bf72e1b2d91d8c5ee603857a62f743ab8229fcbb25;
```

デバイス プログラマは、64 文字の 16 進数値のみ使用し、その後に Key 0 が続き、Key 0 値のビット順は KEY[0:255] です。StartCBC 値と HMAC 値はビットストリーム生成中に使用され、eFUSE にはプログラムされません。

# ユーザー定義の 32 ビット値

これらの値は、eFUSE の 32 ビットのユーザー定義値フィールドをプログラムするために、デバイス プログラマの入力としてオプションで指定できます。

32 ビット値へアクセスする方法は次のとおりです。

- JTAG FUSE_USER 命令 (R_EN_B_User [読み出し保護機能] eFUSE ビットがプログラムされていない場合)
- FPGA デザインの EFUSE_USR プリミティブ

通常、この値は 8 文字の 16 進数値 [31:0] としてデバイス プログラマに指定されます。

**注記** : eFUSE アレイにおけるユーザー定義ビット値の物理的な割り当てにより、Vivado ハードウェア マネージャーなど一部のプログラマ ソリューションでは、ユーザー値を user value[31:8] と user value[7:0] に分割します。プログラマ演算子は、完全な値 [31:0] からこのように分割できることが必要です。

# eFUSE 制御 (FUSE_CNTL) ビット

eFUSE 制御ビットは、FUSE_CNTL eFUSE レジスタに格納され、各ビットによって特定のセキュリティ設定が永久的に有効になります。通常、選択される eFUSE 制御ビットの設定は、ユーザー アプリケーションの要件に基づきます。各 FUSE_CNTL ビットの説明および標準的なアプリケーションでの FUSE_CNTL の設定例は、表 1 を参照してください。標準的なアプリケーションとは、ビットストリーム内容を保護し、その使用を照合 AES キーを用いてプログラムされた FPGA に限定するアプリケーションのことを指します。

追加で eFUSE 制御ビットをプログラムすることで、標準的なアプリケーションを上回るセキュリティ要件に対応できます。標準設定以上に eFUSE 制御ビットをプログラムした場合に RMA 解析へ与える影響については、表 1 の注意欄を参照してください。

# 暗号化された検証ビットストリーム

特別に設計されて暗号化されたビットストリームをデバイス プログラマへ提供することで、「eFUSE プログラムにデバイス プログラマを使用」で説明されている検証プロセスを実行できます。

このセクションでは、デバイス プログラマ向けに、暗号化された検証ビットストリームを構築するためのリファレンス デザインと手順を提供します。

**注意** : デバイス プログラマ ソケットのピン配置および制約により、デバイス プログラマでの検証に実際のデザイン ビットストリームは使用しないでください。

## リファレンス デザイン

デバイス プログラマによる制約 (「暗号化された検証ビットストリーム デザインの変更と制約」で説明) があるため、リファレンス デザインの大半は空です。

このアプリケーション ノートのリファレンス デザインは、次のリンクからダウンロードできます。

https://secure.xilinx.com/webreg/clickthrough.do?cid=393635

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 |
|---|---|
| **一般** | |
| 開発者 | Randal Kuramoto |
| 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 |

表 2 : リファレンス デザインの詳細 (続き)

| パラメーター | 説明 |
|---|---|
| ソース コードの形式 | VHDL/Verilog |
| 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 |
| **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 2014.4 合成 |
| 使用したインプリメンテーション ツール/バージョン | Vivado 2014.4 インプリメンテーション |
| 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 | AC701 評価キット |

## リファレンス デザインで暗号化された検証ビットストリームを構築

リファレンス デザイン ファイルをダウンロードした後、次の手順に従って、暗号化された検証ビットストリームを構築します。

1. リファレンス デザイン ディレクトリ ツリーとファイルをコンピューターに解凍します。
2. リファレンス デザイン ディレクトリ ツリー内の `verilog` または `vhdl` ディレクトリに、ユーザーの NKY ファイルをコピーします。
3. `verilog` または `vhdl` ディレクトリ内の `XAPP1260_make_encrypted_verification_bitstream.tcl` ファイルを編集し、Tcl スクリプト ファイル内の使用手順に従って、各デバイスや NKY ファイルのスクリプトをカスタマイズします。次の各手順で、Tcl スクリプトの編集部分を示します。
   a. Tcl スクリプトの次の行にある xc7a200tfbg676-1 を置き換えて、デバイスを指定します。
      ```
      set part xc7a200tfbg676-1
      ```
   b. Tcl スクリプトの次の行にある `XAPP1260_example.nky` を置き換えて、NKY ファイルを指定します。
      ```
      set nkyFileName XAPP1260_example.nky
      ```
4. Vivado デザイン ツールを起動して、`verilog` または `vhdl` ディレクトリに移動 (cd) し、`source` コマンドで `XAPP1260_make_encrypted_verification_bitstream.tcl` ファイルを読み込みます。つまり、次のコマンドラインを使用します。
   ```
   vivado -mode batch -source XAPP1260_make_encrypted_verification_bitstream.tcl
   ```
5. ログ内の write_bitstream オプションのサマリに「暗号化された検証ビットストリームのチェックリスト」の設定が含まれていることを確認します。

Tcl スクリプトによって、暗号化された検証ビットストリームが生成され、MCS ファイル形式で出力されます。MCS ファイルは、デバイス プログラマで使用できます。生成されたファイル名は、次のように表わされます。

```
XAPP1260_<part>_encrypted_verification_bitstream.mcs
```

<part> には、「リファレンス デザインで暗号化された検証ビットストリームを構築」の手順 3 で指定したデバイス名が入ります。

デザインを検証またはデバッグする目的として、暗号化せずに FPGA でデザインが機能するかを確認するために、Tcl は暗号化されていないビットストリームも生成できます。暗号化されていないビットストリーム ファイル名は、次のように表されます。

```
XAPP1260_<part>_unencrypted_bitstream.bit
```

<part> には、「リファレンス デザインで暗号化された検証ビットストリームを構築」の手順 3 で指定したデバイス名が入ります。

**注記** : XAPP1260_example.nky ファイルには、プログラムされていない eFUSE を備えた FPGA のキー値と一致する、すべて 0 のキー値が含まれています。XAPP1260_example.nky ファイルを用いて生成された暗号化テスト ビットストリームを正しく復号化し、eFUSE がプログラムされていない FPGA のコンフィギュレーションに使用できます。これは、デフォルトのすべて 0 の eFUSE キーがすべて 0 の暗号化キーと一致するため可能です。この方法を利用すると、FPGA の eFUSE をプログラムしなくても Vivado デザイン ツールの暗号化フローを検証できます。

## ビルド エラーの場合

Vivado デザイン ツールの Tcl スクリプト実行で生成された vivado.log ファイルを確認します。デバイス名や NKY ファイルに関する問題が生じると、Tcl スクリプトがエラーをレポートします。エラーがレポートされた場合は、プロジェクト サブディレクトリ内のログ ファイルでエラー内容を確認してください。

```
ERROR:[Common 17-69] Command failed:Run 'impl_1' has not been launched.
```

詳細は、XAPP1260_encrypted_verification_bitstream.runs\synth_1\runme.log ファイルで、合成中のエラーを確認できます。

## 暗号化された検証ビットストリーム デザインの変更と制約

基本のリファレンス デザインから生成される暗号化されたコンテンツを変更するため、必要に応じてリファレンス デザインを変更できます。ただし、最終的な FPGA デザインは次の制約事項に従う必要があります。

- 検証ビットストリームは、電源投入時の最小電流を超えてはいけません。通常、デバイス プログラマへの電力供給は制限されています。デバイス プログラマは、デバイスに電源を投入し、eFUSE をプログラムするだけの電力しか備えていないことに注意してください。
- ユーザー指定の I/O は使用しないでください。デバイス プログラマのピン接続は固定されています。デバイス プログラマは、専用の JTAG インターフェイス ピンとのみ適切に接続できることに注意してください。ユーザー指定のピンの接続および電力は定義されていません。
- トランシーバーをインスタンシエートまたは有効化しないでください。デバイス　プログラマのソケットは、トランシーバーを使用しない、または有効にしないことを前提に設計されていることに注意してください。

## 暗号化された検証ビットストリームのチェックリスト

暗号化された検証ビットストリームに対して、プロパティが次のように適切に指定されている必要があります。

```
BITSTREAM.ENCRYPTION.ENCRYPT = YES
BITSTREAM.ENCRYPTION.ENCRYPTKEYSELECT = EFUSE
BITSTREAM.ENCRYPTION.KEYFILE = <NKY file name that contains your AES key>
```

デバイス プログラマでの検証時間を最小にするには、次のプロパティ設定が重要です。

```
BITSTREAM.GENERAL.COMPRESS = TRUE
```

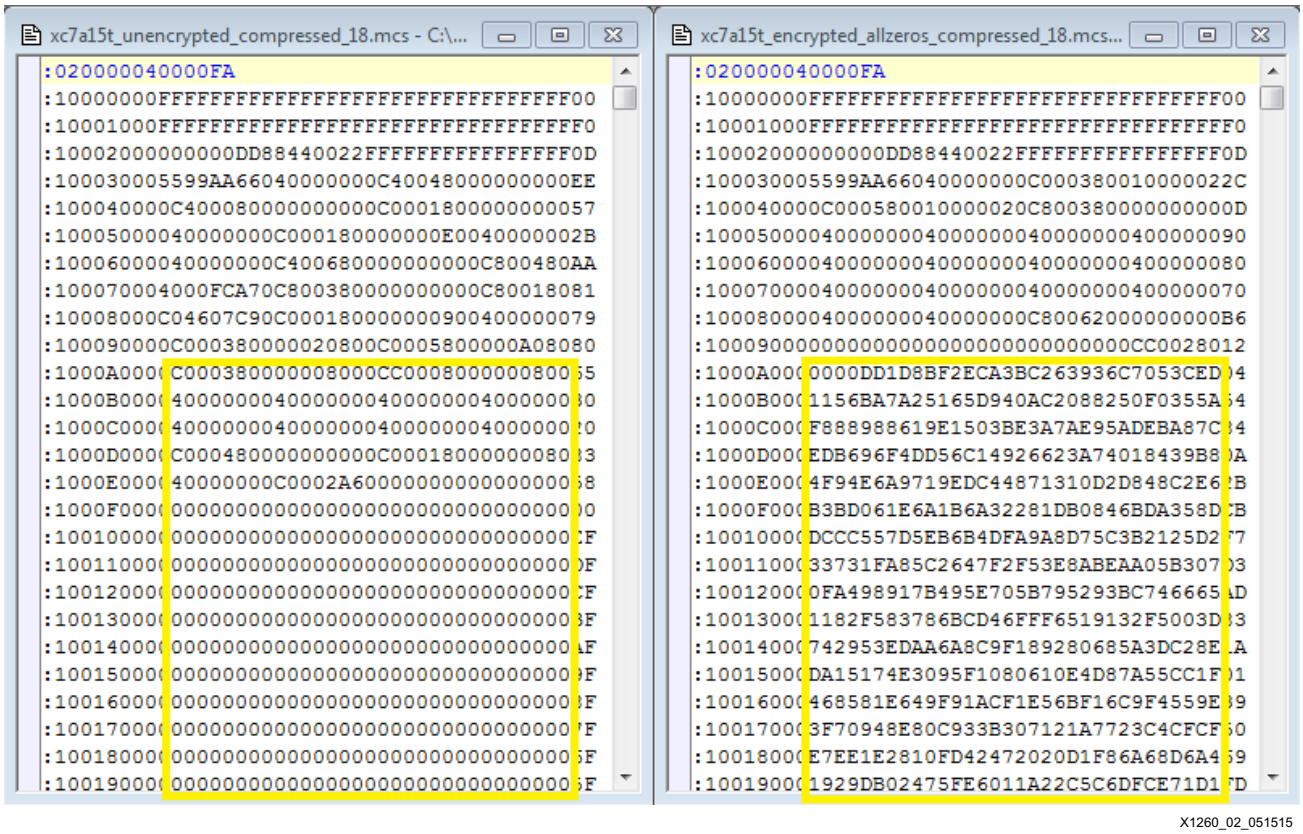
ここにリストしたプロパティ設定は、-verbose オプションを使用して write_bitstream コマンドを実行することで検証できます。リファレンス デザインの Tcl スクリプトには、-verbose オプションと write_bitstream コマンドが含まれています。-verbose オプションを使用して write_bitstream が実行されると、ビットストリームの書き込み時にプロパティ設定がログ ファイルに記録されます。サマリ内の重要な部分を次に示します。

```
Summary of write_bitstream Options:
+-------------------------+-------------------------+
| Option Name             | Current Setting         |
+-------------------------+-------------------------+
...
+-------------------------+-------------------------+
| KEYFILE                 | your_key_file_name.nky  |
+-------------------------+-------------------------+
...
```

```
+-------------------------+-------------------------+
| ENCRYPTKEYSELECT        | EFUSE                   |
+-------------------------+-------------------------+
...
+-------------------------+-------------------------+
| ENCRYPT                 | YES                     |
+-------------------------+-------------------------+
...
+-------------------------+-------------------------+
| COMPRESS                | TRUE                    |
+-------------------------+-------------------------+
...
```

先にリストしたプロパティ設定は、ビットストリームが復号化の実行や AES 暗号化キーの格納場所を指示するという理由から重要です。これらのプロパティを設定せずに生成されたビットストリームは、eFUSE に格納された AES キー値をテストせずに、FPGA をコンフィギュレーションできます。たとえば、標準の FPGA eFUSE 設定では、暗号化されていないビットストリームからのコンフィギュレーションが可能です。したがって、格納された AES キーをテストせずに、暗号化されていないビットストリームで標準の FPGA をプログラムできることになります。

暗号化された検証ビットストリーム (MCS) ファイルのビットストリーム ヘッダー以降のデータ (12 行目以降) は、暗号化されていることを視覚的に確認できます。暗号化されていないビットストリーム データは、ほとんど 0 ビット値となります (FPGA デザイン内の大半のリソースが未使用なため)。暗号化されたビットストリームは、よりランダムなデータ パターンとなります。図 2 に、暗号化されていないビットストリームと暗号化されたビットストリームのデータ パターンの比較を示します。黄色の枠でハイライトした部分がビットストリーム データです。




X1260_02_051515

図 2 : 暗号化されていないビットストリーム (左) と暗号化されたビットストリーム (右)

# デバイス プログラマのセットアップと動作を検証するガイドライン

前のセクションでは、デバイス プログラマの設定を定義し、デバイス プログラマの入力ファイルを構築する方法について説明しました。

このセクションでは、次のガイドラインを提供します。

- デバイス プログラマへ提供する前に、デバイス プログラマの設定と入力ファイルが予定されていたものであるかを確認する。
- デバイス プログラマの設定および入力ファイルを確認する。
- デバイス プログラマでプログラムされた最初のデバイスが、その eFUSE 設定で、アセンブリ済みボード上で正しく動作するかを検証する。

これらのガイドラインをチェックリストとして使用することで、デバイス プログラマで 7 シリーズ FPGA eFUSE を確実にプログラムできます。

## デバイス プログラマ ファイルの事前確認

このセクションでは、デバイス プログラマに提供する前に、デバイス プログラマの設定と入力ファイルが予定されていたものであるかを確認するためのガイドラインを示します。

1. プロトタイプ ボード上で、暗号化を行わずに、基本のリファレンス デザインがユーザー デバイスで機能することを確認します。

   Vivado ハードウェア マネージャーを使用して、プロトタイプ ボード上にある、eFUSE 内の AES キーを用いてプログラムされていない FPGA に `XAPP1260_<part>_unencrypted_bitstream.bit` ファイルをダウンロードします。この FPGA はコンフィギュレーションを正常に完了するはずです。

2. プロトタイプ ボード上で、暗号化されたビットストリームのセキュリティを検証します。

   Vivado ハードウェア マネージャーを使用して、プロトタイプ ボード上にある、eFUSE 内の AES キーを用いてプログラムされていない FPGA に暗号化された検証ビットストリームをダウンロードします。この FPGA はコンフィギュレーションを正常に完了できないはずです。

3. 暗号化された検証ビットストリームと実際の暗号化されたデザイン ビットストリームがプロトタイプ ボード上のデバイスで有効であることを確認します。

   a. Vivado ハードウェア マネージャーを使用して、ユーザーが記入した表 1 の設定で eFUSE をプログラムします。Vivado ツールを使用してデバイス eFUSE へアクセスする際の手順については、『暗号化を使用して 7 シリーズ FPGA ビットストリームを保護』(XAPP1239) [参照 3] を参照してください。

      - Vivado ハードウェア マネージャーを使用して、デバイスの [Properties] → [Register] → [eFUSE] → [FUSE_CNTL] で、FUSE_CNTL の値を確認します。Vivado ハードウェア マネージャーの eFUSE プログラミングから得た FUSE_CNTL 値のコピーを保存します。この値を使用して、デバイス プログラマから得られる FUSE_CNTL 値が同じであるかを後に検証できます。

   b. Vivado ハードウェア マネージャーを使用して、暗号化された検証ビットストリームをプロトタイプ ボード上のデバイスにダウンロードします。FPGA はコンフィギュレーションを正常に完了するはずです。

   c. Vivado ハードウェア マネージャーを使用して、実際の暗号化されたデザイン ビットストリームをプロトタイプ ボード上のデバイスにダウンロードします。FPGA はコンフィギュレーションを正常に完了するはずです。

# デバイス プログラマの設定検証

このセクションでは、デバイス プログラマの設定と入力ファイルを確認するためのガイドラインを示します。

4. 照合キーでデバイスがプログラムされていない場合、デバイス　プログラマのスタンドアロン検証動作でエラーがレポートされることを確認します。
   a. ファクトリからの新品デバイスをデバイス プログラマ ソケットに挿入します。
   b. 暗号化された検証ビットストリームでスタンドアロン検証動作を実行します。
   c. デバイス プログラマがエラーをレポートするはずです。

      スタンドアロン検証がエラーをレポートしなかった場合 :

      - デバイスが空であることを再確認します。
      - 暗号化された検証ビットストリームを再確認します。Vivado ハードウェア マネージャーを使用して、暗号化された検証ビットストリームで、ボード上の eFUSE がプログラムされていないデバイスに対してコンフィギュレーションを実行します。Vivado ハードウェア マネージャーは、コンフィギュレーション エラーをレポートするはずです。
5. 設定を使用して、1 つまたは 2 つのテスト デバイスをプログラムして検証します。その後これらのプログラム済みテスト デバイスは、アセンブリ済みボード上での検証プロセスへ渡されます。

   検証動作に失敗した場合は、「デバイス プログラマ ファイルの事前確認」のガイドラインに従って、暗号化された検証ビットストリームを再確認します。

# 最初にプログラムされたデバイスをアセンブリ済みボード上で検証

このセクションでは、デバイス プログラマでプログラムされた最初のデバイスが、これらの eFUSE 設定で、アセンブリ済みボード上で正しく機能するかを検証するためのガイドラインを提供します。

6. FUSE_CNTL レジスタで、AES キーの読み出し保護機能およびその他のセキュリティ設定を確認します。
   a. Vivado ハードウェア マネージャーを使用して、JTAG を介してボード上のデバイスに接続します。
   b. デバイスの [Properties] → [Register] → [eFUSE] → [FUSE_CNTL] から、FUSE_CNTL の値を確認します。プロトタイプ ボードでの eFUSE の Vivado ハードウェア マネージャー プログラミングから得た FUSE_CNTL の値を保存している場合は、その値とデバイス プログラマでプログラムされたデバイスの値を比較します。
   c. 少なくとも R_EN_B_Key (FUSE_CNTL[3]) ビットが 1 にプログラムされていることを確認します。たとえば、図 3 では、16 進数値 `0008` (バイナリ値 `00000000001000`) を示しています。この例では、1 にプログラムされた FUSE_CNTL[3] ビットが正しく定義されています。
      - FUSE_CNTL[3] ビットが 1 にプログラムされていない場合、またはデバイス プログラマでプログラムされたデバイスから取得した FUSE_CNTL レジスタ値が Vivado ハードウェア マネージャーでプログラムされたデバイスの値と一致しない場合は、デバイス プログラマのすべての設定を再確認します。
7. 暗号化されたビットストリームを FPGA にロードします。

a. コンフィギュレーションに失敗した場合は、ユーザー デザインとデバイス プログラマの設定が同じ NKY ファイル (AES キー ) を使用してビットストリームを暗号化しているかを再度確認します。

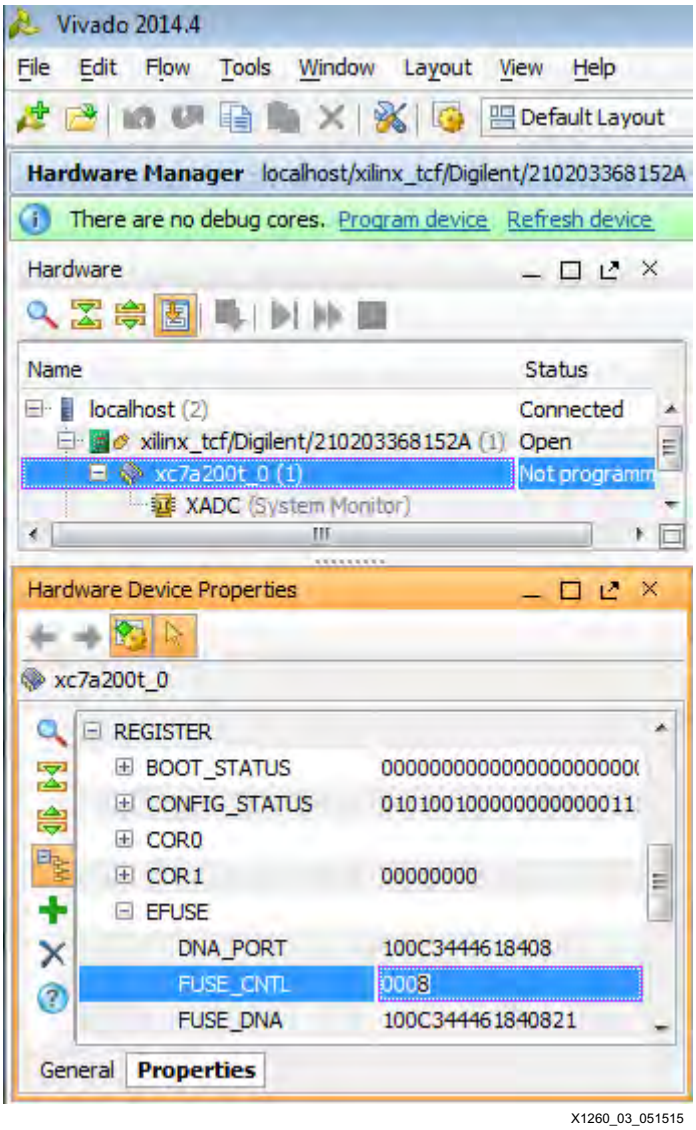



図 3 : Vivado ハードウェア マネージャーの FUSE_CNTL 値

# 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. 『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)
3. 『暗号化を使用して 7 シリーズ FPGA ビットストリームを保護』(XAPP1239 : 英語版、日本語版)

# 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 7 月 8 日 | 1.0 | 初版 |

# 法的通知



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。フィードバックは日本語で入力可能です。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。