XAPP1267 (v1.0) 2016 年 6 月 2 日

暗号化と認証を使用して UltraScale/UltraScale+ FPGA のビットストリームを保護

著者 : Kyle Wilkinson

概要

このアプリケーションでは、ザイリンクスの Vivado® Design Suite で暗号化ビットストリームおよび暗号化キー (AES-GCM および RSA 認証) を生成する簡単な手順について説明します。AES-GSM 暗号化キーや RSA 公開キーのハッシュ値をプログラムする手順、さらに Vivado Design Suite を使用してザイリンクスの UltraScale™ FPGA に暗号化したビットストリームをロードする手順も説明します。このアプリケーション ノートの内容は、UltraScale FPGA および UltraScale+™ FPGA に適用されます。

はじめに

UltraScale デバイスにはオンチップに搭載された AES –GCM (Advanced Encryption Standard – Galois/Counter Mode) 復号および認証ロジックがあり、デザインの高い安全性が確保されているため、暗号化した UltraScale FPGA デザインのコピーやリバース エンジニアリングは不可能です。UltraScale FPGA の AES システムは、ソフトウェア ベースのビットストリーム暗号化機能と、オンチップのビットストリーム復号化機能、および暗号化キーと暗号化されたビットストリームを格納する専用メモリで構成されます。暗号化キーと暗号化されたビットストリームは、Vivado ツールを使用して生成されます。

UltraScale デバイスでは、この暗号化キーをデバイス内部の専用 RAM (外部の小型バックアップ バッテリに接続された RAM - BBRAM) または eFUSE のいずれかに格納します。RSA 認証を使用する場合は、RSA 公開キーのハッシュ値を eFUSE にプログラムする必要があります。暗号化キーは、JTAG ポートを介してのみデバイスにプログラムできます。BBRAM と eFUSE のいずれもリードバック不可です。コンフィギュレーション実行中、UltraScale デバイスでは反対の処理、つまり取り込んでいるビットストリームの復号化が行われます。UltraScale FPGA の AES 暗号化ロジックは、256 ビットの暗号化キーを使用します。オンチップの AES 復号化ロジックは、ビットストリームの復号化以外の目的には使用できません。つまり、ユーザー デザインで AES 復号化ロジックを使用し、コンフィギュレーション ビットストリーム以外のデータを復号化することはできません。

AES (Advanced Encryption Standard) および認証

UltraScale FPGA の暗号化システムは、AES-GCM 認証付きの暗号化アルゴリズムを使用します。AES は、NIST (National Institute of Standards and Technology) および米国商務省が認証する公式規格です (詳細は AES の資料 [参照 1] および GCM の仕様 [参照 2] を参照)。

AES-GCM は、ビルトイン認証をサポートしている点に特長があります。UltraScale FPGA の AES 暗号化システムは 256 ビットの暗号化キーを使用し (NIST が定める 128 ビットおよび 192 ビットの暗号化キーはインプリメントしない)、一度に 128 ビットのデータ ブロックを暗号化または復号化します。NIST によると、256 ビット キーの場合、キーの組み合わせは $1.1x10^{77}$ 通り考えられます。最も安全な方法として、Vivado の疑似ランダム キー生成機能を使用するのではなく、手動で 256 ビット キーを作成することを推奨しています。

ビットストリーム認証

AES-GCM 暗号化規格はビルトイン認証をサポートしており、7 シリーズ FPGA のように認証用の HMAC キーを指定する必要がないため、セキュリティがさらに強化されています。AES-GCM キーが入手されない限り、ビットストリームの変更または偽造は不可能です。暗号化はデザインを複製やリバース エンジニアリングから保護するための基本的なセキュリティを提供し、認証は FPGA のコンフィギュレーション用に提供されたビットストリームが認証ユーザーによって作成された未修正ビットストリームであるかを確認します。認証は、ビットストリームのデータが本物で改ざんされていないことを検証します。

この認証機能は、あらゆる種類の制御ビットとデータ ビットを含む、ビットストリーム全体に適用されます。シングル ビットの反転を含め、ビットストリームに対するあらゆる改ざんを検出できます。認証に合格すると、コンフィギュレーションはスタートアップ サイクルまで終えて完了します。認証エラーが生じ、AES-GCM エンジンがビットストリームに変更が加えられていることを検出すると、デバイスはスタートアップ プロセスを開始しません。フォールバックが有効であれば、デバイス全体のコンフィギュレーションをクリアした後でフォールバック ビットストリームが読み込まれます。フォールバックが無効の場合は、コンフィギュレーション ロジックがコンフィギュレーション インターフェイスを無効にし、FPGA へのアクセスを完全に遮断します。この場合、PROGRAM_B 信号または POR (パワーオン リセット) 信号をパルスして、コンフィギュレーション インターフェイスをリセットする必要があります。ビットストリームの認証には、次に示す 2 つの方法のいずれか 1 つを選択する必要があります。

1. ビットストリームを暗号化する場合は、AES-GCM 認証を使用できます。
2. ビットストリームの暗号化を使用する場合またはビットストリームを暗号化しない場合は、RSA-2048 認証を使用できます。RSA-2048 については、次の段落で説明しています。

RSA 認証

AES-GCM は対称キーを用いた自己認証アルゴリズムで、暗号化と復号化に同じキーを使用します。したがって、このキーを非公開キーとして保護する必要があり、キーを格納するための領域が内部に必要になります。UltraScale アーキテクチャはもう 1 つの認証手段として RSA-2048 をサポートしています。RSA は非対称アルゴリズムで、署名に使用するキーと検証に使用するキーが異なります。検証には公開キーを用います。この公開キーは保護の必要がないため、セキュアな格納領域は不要です。必要に応じてこの認証方式と暗号化を組み合わせることにより、真正性と機密性の両方を確保できます。RSA-2048 は、暗号化されているビットストリームと暗号化されていないビットストリームのいずれにも使用可能です。RSA のメリットは、公開キーを使用することだけではなく、復号化する前に認証を行うことができることが挙げられます。RSA 公開キーのハッシュ値は eFUSE に格納する必要があります。

UltraScale FPGA が RSA-2048 をサポートする理由は、ビットストリーム データが復号化エンジンへ送信される前に認証を行うためです。この方法では、データの真正性を確認してから復号化を実行することになるため、復号化エンジンへの攻撃を防ぐことができます。RSA 認証はビットストリーム暗号化から切り離して利用できるため、暗号化の有無にかかわらずビットストリームの認証が行えます。RSA コンフィギュレーション制御ロジックは、公開キーとビットストリーム シグネチャを含む暗号化ビットストリームを読み出してデバイス メモリに格納します。次に、RSA コンフィギュレーション制御ロジックからの指示により、RSA エンジンはこの公開キーとシグネチャから予想されるダイジェストを計算します。

ビットストリームがバッファーに格納され、予想されるダイジェスト値が RSA エンジンによって計算されたら、実際のダイジェストと予想値を比較します。コンフィギュレーション データが暗号化されていない場合、RSA 認証に合格すると FPGA の動作が開始します。コンフィギュレーション データが暗号化されている場合は、RSA 認証に合格した後、ビットストリームの復号化が実行されます。RSA 認証に失敗した場合、AES-GCM 認証エラーと同じエラーが生成されます。この場合、デバイスはロック ダウンするか、有効な場合はフォールバックが実行されます。

同じデバイスであれば、RSA 認証を利用したビットストリームは標準の非圧縮ビットストリームの最大 3 倍のコンフィギュレーション時間がかかります。実際に必要な時間は、コンフィギュレーション モードによって異なります。RSA 認証は、ビットストリームの圧縮、パーシャル リコンフィギュレーション、または Tandem ソリューションなどの PCI e インターフェイス経由のコンフィギュレーションと併用できません。

RSA 認証は、特定のコンフィギュレーション モードと幅を使用する UltraScale および UltraScale+ デバイスでサポートされています。RSA 認証をサポートする UltraScale FPGA デバイスとコンフィギュレーション モードの詳細は、『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570) [参照 3] の「RSA 認証」を参照してください。

キー ローリング

UltraScale FPGA では、ビットストリームを複数の AES 暗号化メッセージに分割し、それぞれ固有のキーで暗号化できます。このローリング キーとして知られる機能を利用することで、最初のキーはオンチップに格納され、後続の各メッセージのキーは直前のメッセージ内で暗号化 (ラップ) されます。ローリング キーは差分電力解析 (DPA) などのサイド チャネル攻撃に対して高いセキュリティを提供します。ビットストリーム オプション BITSTREAM.ENCRYPTION.KEYLIFE で、各キーの暗号化ブロック数が定義されます。1 つのキーに対して暗号化ブロック数が少ないほどセキュリティ レベルが向上しますが、これによりビットストリーム サイズが急増し、コンフィギュレーション時間が長くなります。1,024 またはそれ以上の値を選択した場合、コンフィギュレーション サイズは約 15% 増加し、64 の値を選択した場合、ビットストリーム サイズは 50% 増加し、32 の値 (デフォルト) を選択した場合、ビットストリーム サイズは 2 倍以上になります。図 1 に、ビットストリーム サイズの乗数と各キーのブロック数を表したグラフを示します。

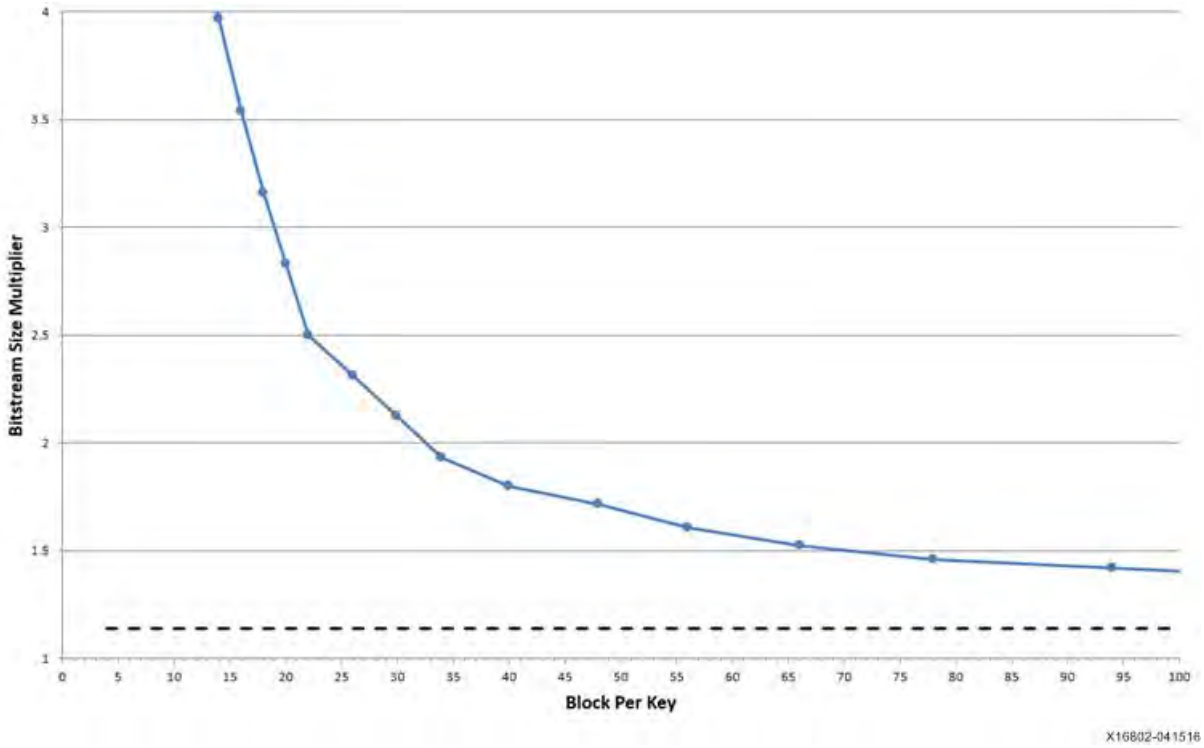


図 1 : ビットストリーム サイズの乗数と各キーのブロック数

ザイリンクスでは、独自の AES キーを作成することを強く推奨していますが、Vivado ソフトウェアを使用して疑似ランダム キーを生成する場合は、出力される NKY ファイルに含まれているキーの数 (Key0、Key1、Keyn) を確認できます。複数のカスタム キーを定義する場合は、それらを NKY ファイルに提供して、BITSTREAM.ENCRYPTION.KEYFILE write_bitstream プロパティを使用する必要があります。この write_bitstream プロパティの詳細は、10 ページの表 6または『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 4] を参照してください。

RSA 認証を使用する場合は、暫定的なローリング キーを保持するために特定のブロック RAM を使用する可能性があり、これらのブロックを初期化する際に影響を及ぼします。いかなるブロック RAM カラムの場合でも、クロック領域の最下部に位置する各 36K ブロックがその影響を受けます。つまり基本的には、デバイス下部で開始する最初の 36K ブロック、それ以降はカラム内の 12 番目ごとの 36K ブロック RAM (BRAM36_X*Y0、BRAM36_X*Y12、BRAM36_X*Y24 など) となります。これらのブロック RAM は、RSA 認証を使用した場合にユーザーが指定する値に初期化されず、コンフィギュレーション後は常に 0 へ初期化されます。この RSA ブロック RAM の使用による影響を受けるデザインでブロック RAM を使用している場合は、DRC がトリガーされます。

より安全な方法としてユーザーが独自キーを定義している場合でかつ、BITSTREAM.ENCRYPTION.KEYLIFE オプションで判断される正しいキー数 (数千個になる可能性がある) を提供していない場合は、ユーザーに代わって Vivado が必要なキーを生成して NKY ファイルに含めます。ザイリンクスでは、独自キー セットを常にユーザーが生成することを推奨しています。

暗号化ビットストリームのインプリメンテーションの概要

UltraScale FPGA に暗号化されたデザインをインプリメントする場合の 7 つの基本手順を次に示します。

1. AES キーの格納場所(BBRAM または eFUSE) および対応するセキュリティ オプションを選択します。(BBRAM と eFUSE のトレードオフに関しては、『UltraScale FPGA での不正操作防止デザインの開発』(XAPP1098) [参照 5] を参照)。
2. 認証方法 (AES-GCM および RSA) を選択します。(AES-GCM と RSA 認証のトレードオフに関しては、(XAPP1098) [参照 5] を参照)。
3. 指定した AES キーの格納場所に基づいて、ボード デザインのハードウェア要件を満たします。
4. Vivado Design Suite ツールを使用して AES キーを生成するか、またはソフトウェアにユーザー独自の AES キーを提供 (常にこちらの方法が高いセキュリティを提供) してビットストリームを暗号化します。

a. AES キーを生成します。

b. 認証方法として RSA が選択されている場合は、OpenSSL [参照 6] またはその他のキー生成ソフトウエアを使用して、RSA の公開キーと非公開キーを生成します。

5. JTAG インターフェイスを介して、FPGA に AES キーをプログラムします。
6. JTAG またはその他のコンフィギュレーション モード (SPI、BPI など) を使用して、FPGA に暗号化されたビット ファイルをプログラムします。

   注記 : RSA 認証をサポートする UltraScale FPGA デバイスとコンフィギュレーション モードの詳細は、『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570) [参照 3] の「RSA 認定」を参照してください。
7. ハードウェア検証を実行して正しく動作していることを確認します。

ハードウェア ボード要件

暗号化デザイン フローを実行するにあたって、いくつか基本的なハードウェア要件があります。

- プログラム機能およびデバッグ機能 : FPGA へは JTAG 経由で接続。
- BBRAM キー ストレージ : V_{BATT} へのバッテリ (バッテリ電圧要件は各データシートを参照)。
- eFUSE キーストレージ : OTP (ワンタイム プログラマブル) の eFUSE にプログラムする前に、BBRAM でテストできるように V_{CCBATT} または V_{CCAUX} に接続することを推奨。

ソフトウェア要件

Vivado Design Suite 2016.1 またはそれ以降のバージョンが必要です。

AES キーのストレージ

AES-GCM キーの格納場所には、バッテリ バックアップ RAM (BBRAM) と eFUSE の 2 つの選択肢があります。

推奨 : ストレージ オプションとして BBRAM または eFUSE のいずれかを選択するときには、各オプションの長所/短所をよく理解した上でデザイン要件に最適なオプションを判断してください。

それぞれの長所と短所の詳細は、この後のセクションで説明しています。これらのストレージ オプションに関するその他の情報は、『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570) [参照 3] を参照してください。

BBRAM ストレージ

暗号化キーを FPGA のバックアップ バッテリ付き RAM に格納した場合、暗号化キーのメモリ セルは揮発性であるため、継続的に電力を供給する必要があります。正常に動作している場合、これらのメモリ セルには補助電圧入力 (V_{CCAUX}) から電源が供給されますが、

推奨 : V_{CCAUX} が切断されたときにキーを維持できるように、独立した V_{BATT} 電源入力を推奨しています。このため、AES キーは、バックアップ バッテリのあるボードでインシステム プログラムすることを推奨します。そうしないと、電源/バッテリが切断されるとキーを失うことになります。

重要 : BBRAM を既知のステートにプログラムしてから、キー ソースとして BBRAM を使用する暗号化されたビットストリームを用いてコンフィギュレーションしてください。BBRAM キーがプログラムされる前に、電源投入時に暗号化されたビットストリームをダウンロードしようとすると、FPGA デバイスはロックアップする可能性があります。その際に

は、デバイスの電源を入れ直し、BBRAM キーを読み込んだ後に暗号化されたビットストリームでコンフィギュレーションする必要があります。

表 1 に、格納場所としての BBRAM の長所/短所について説明しています。

表 1 : 格納場所としての BBRAM の長所/短所

長所	短所
• 揮発性、再プログラム可能 • 受動的および能動的にキーを消去可能 (つまり、証拠を削除できる) • 不正操作防止(1) • BBRAM は RSA 認証またはコンフィギュレーション カウント機能を使用して DPA 攻撃から保護できる • リードバック パスがないため、BBRAM キーのリードバックは不可	• 外部バッテリが必要 • 多くのバッテリ ベンダーは高温/長期利用における動作仕様を定義していない

注記 :

1. BBRAM からキーを読み出すための物理的パスはありません (書き込み専用パスはある)。

eFUSE ストレージ

eFUSE は、コンフィギュレーション設定に使用する不揮発性のワンタイム プログラマブル技術です。ヒューズ リンクは、一定期間に大量の電流を流すことでプログラム (バーン、ブローとも呼ぶ) します。ユーザー プログラム可能な eFUSE のプログラムにはザイリンクス コンフィギュレーション ツールが使用できます。

重要 : eFUSE ビットは OTP (ワンタイム プログラマブル) です。一度プログラムすると再プログラムはできません。

たとえばレジスタへのアクセスを無効にした場合、後から有効に変更できません。FPGA ロジックからアクセスできるのは FUSE_USER レジスタのみで、それ以外の eFUSE ビットには FPGA ロジックからはアクセスできません。表 2 に、格納場所としての eFUSEの長所/短所について説明しています。

表 2 : 格納場所としての eFUSE の長所/短所

長所	短所
• 外部バッテリが不要 • eFUSE キーで暗号化されたビットストリームのみ FPGA へロード可能 • リードバック パスがないため、eFUSE キーのリードバックは不可 • EFUSE は RSA 認証を使用して DPA 攻撃から保護できる	• 恒久的 : キーの消去または変更が不可 • キー値のゼロ化や変更ができないため、BBRAM ソリューションよりセキュリティ レベルが低い

難読化キー

UltraScale FPGA では、AES キーを難読化した形式でデバイスにロードできます。これにより、委託製造会社には難読化したキーを提示できるため、実際の AES-256 キーを露呈することがなくなります。BITSTREAM.ENCRYPTION.OBFUSCATEKEY プロパティを設定した場合、Vivado の write_bitstream ソフトウェアは出力ファイル NKY に新しいキー (ObfuscateKey) を生成します。この難読化キーは、シリコンに格納されているメタライズされたファミリ キーを使用して AES-256 キーを暗号化することで生成できます。すべての UltraScale デバイスで同じキーが使用されます。

ユーザーは、実際の AES-256 キーの代わりに難読化キーを委託製造会社に提示できます。eFUSE または BBRAM のいずれかにキーが格納されている場合、NKY ファイルに KeyObfuscate フィールドが含まれていると、格納場所で自動的にフラグがセットされて、このキーが難読化されていることを示します。また、最終的なビットストリームには、キーを使用して残りのビットストリームを復号化する前に、適切な AES-256 キーの格納場所を復号化するようチップに伝える命令が含まれます。ビットストリームで選択される場所にある難読化キーの設定は、ビットストリームの難読化キーの設定と

一致する必要があります。BITSTREAM.ENCRYPTION.OBFUSCATEKEY プロパティは、BBRAM キー ストレージのコンフィギュレーション カウント機能による DPA 対抗措置と併用できません。

eFUSE レジスタ

UltraScale FPGA には、全部で 6 つの eFUSE レジスタ (FUSE_RSA、FUSE_KEY、FUSE_DNA、FUSE_USER、FUSE_CNTL、および FUSE_SEC) があります。このアプリケーション ノートでは、FUSE_DNA レジスタに焦点を当てて説明していません。表 3 では、すべての UltraScale eFUSE レジスタについて説明しています。

表 3 : eFUSE レジスタの説明

レジスタ名	サイズ (ビット)	内容	説明
FUSE_RSA	384	ビットストリーム認証キー [0:383] (ビット 383 が最初にシフト)	RSA ビットストリーム認証に使用する公開キーのハッシュ値を格納します。
FUSE_KEY	256	ビットストリーム暗号化キー [0:255] (ビット 255 が最初にシフト)	AES-GCM ビットストリームの復号化と認証で使用するキーを格納します。バックアップ バッテリ付きの SRAM (BBRAM) にキーを格納する代わりに、eFUSE にキーを格納して使用できます。UltraScale FPGA の復号化エンジンは、この AES キーを使用して暗号化ビットストリームを読み込みます。AES キーは、FUSE_CNTL レジスタの読み出し/書き込みアクセス ビットの設定に基づき、JTAG ポートを介して読み出しまたはプログラムが可能です。
FUSE_DNA	96	ザイリンクスがプログラムするデバイス ID [95:0] (ビット 0 が最初にシフト)	デバイス ID ビット [95:0] です。Device DNA として知られる 96 ビットの読み出し専用 DNA_PORTE2 プリミティブ値に対応します。
FUSE_USER	32 または 128	ユーザー定義 [31:0] または [128:0] (ビット 0 が最初にシフト)	32 ビットまたは 128 ビットのユーザー定義コードを格納します。このレジスタは、eFUSE_USR プリミティブを使用して FPGA ロジックから読み出し可能です(eFUSE_USR プリミティブの詳細は、『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570) [参照 3] の第 7 章「デザイン入力」を参照)。このコードは、FUSE_CNTL レジスタの読み出し/書き込みアクセス ビットの設定に基づき、JTAG ポートを介して読み出しまたはプログラムが可能です。
FUSE_CNTL	21	制御ビット CNTL [20:0] (ビット 0 が最初にシフト)	キーの使用や eFUSE レジスタへの読み出し/書き込みアクセスを制御します。このレジスタは、JTAG ポートを介して読み出しまたはプログラムが可能です。
FUSE_SEC	32	セキュリティ制御ビット [31:0] (ビット 0 が最初にシフト)	暗号化および認証に関するオプションを設定します。このレジスタは、FUSE_CNTL レジスタの読み出し/書き込みアクセス ビットの設定に基づき、JTAG ポートを介して読み出しまたはプログラムが可能です。

eFUSE 制御レジスタ (FUSE_CNTL) ビットの説明

このレジスタに含まれるユーザー プログラマブルなビットを使用して、AES キーの使用法やその他の eFUSE レジスタの読み出し/書き込み保護を設定します。表 4 に、ビットの説明と推奨設定を示します。

表 4 : eFUSE 制御レジスタ (FUSE_CNTL) ビットの説明

ビット	ビット名	説明	推奨設定
0	R_DIS_Key	• FUSE_KEY 暗号化キーの読み出しとプログラムを無効にします。 ◦ 1 にプログラムされている場合、AES キーの読み出しと、AES キーおよびユーザー定義値のビットのプログラムが無効になります。	あり (1 にプログラム)
1	R_DIS_USER	• FUSE_USER ユーザー コードの読み出しとプログラムを無効にします。JTAG ポートを介したユーザー コード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 コードの読み出しは無効になりません。 ◦ 1 にプログラムされている場合、JTAG を経由したユーザー定義値の読み出し/プログラムが無効になります。 **注記** : ユーザー定義値は、eFUSE_USR プリミティブを介して FPGA デザインから常時アクセス可能です。	なし (0 を保持)
2	R_DIS_SEC	• FUSE_SEC セキュリティ設定の読み出しとプログラムを無効にします。FUSE_SEC eFUSE ビットの書き込みイネーブルです (アクティブ Low)。 ◦ 1 にプログラムされている場合、FUSE_SEC ビットのプログラムが無効になります。 **推奨** : FUSE_SEC レジスタのビットをプログラムした後に、このビットを 1 にプログラムすることによって、FUSE_SEC eFUSE ビットへの意図しない変更を防ぐことができます。	あり (1 にプログラム)
3 ～ 4	予約	予約	–
5	W_DIS_CNTL	• FUSE_CNTL 制御設定のプログラムを無効にします。FUSE_CNTL eFUSE ビットの書き込みイネーブルです (アクティブ Low)。 ◦ 1 にプログラムされている場合、FUSE_CNTL ビットのプログラムが無効になります。 **推奨** : FUSE_CNTL レジスタのビットをプログラムした後に、このビットを 1 にプログラムすることによって、FUSE_CNTL eFUSE ビットへの意図しない変更を防ぐことができます。	あり (1 にプログラム)
6	R_DIS_RSA	FUSE_RSA 認証キーの読み出しとプログラムを無効にします。	あり (1 にプログラム)
7	W_DIS_KEY	• FUSE_KEY 暗号化キーのプログラムを無効にします。キーおよびユーザー定義の eFUSE 値への書き込みイネーブルです (アクティブ Low)。 ◦ 1 にプログラムされている場合、AES キーのプログラムが無効になります。 **推奨** : キーをプログラムした後にこのビットプログラムすることで、eFUSE AES キー値への意図しない変更/破損を防止します。	あり (1 にプログラム)

表 4 : eFUSE 制御レジスタ (FUSE_CNTL) ビットの説明 (続き)

ビット	ビット名	説明	推奨設定
8	W_DIS_USER	FUSE_USER ユーザー コードのプログラムを無効にします。	なし (0 を保持)
9	W_DIS_SEC	• FUSE_SEC セキュリティ設定のプログラムを無効にします。セキュリティ レジスタの書き込みイネーブルです (アクティブ Low)。 。1 にプログラムされている場合、FUSE_SEC レジスタのビットのプログラムが無効になります。  **推奨** : FUSE_SEC レジスタのビットをプログラムした後に、このビットをプログラムすることによって、FUSE_SEC レジスタへの意図しない変更を防ぐことができます。	あり (1 にプログラム)
10 ～ 14	予約	予約	–
15	W_DIS_RSA	FUSE_RSA 認証キーのプログラムを無効にします。	顧客のセキュリティ要件を未決定にしておく
16 ～ 20	予約	予約	–

eFUSE セキュリティ レジスタ (FUSE_SEC) の説明

このレジスタに含まれるユーザー プログラム可能なビットを使用して、eFUSE セキュリティの設定を選択したり、必要に応じて RSA 認証を有効にできます。表 5 に、ビットの説明と推奨設定を示します。

表 5 : eFUSE セキュリティ レジスタ (FUSE_SEC) のビットの説明

ビット	ビット名	説明	推奨設定
0	FUSE_SHAD_SEC[0] (CFG_AES_Only)	暗号化したビットストリームのみを許可します。	なし (0 を保持)
1	FUSE_SHAD_SEC[1]	eFUSE に格納された AES キーの使用を強制します (BBRAM キーは無効)。このビットがプログラムされていない場合、ビットストリームを暗号化するかどうか、およびキーをバックアップ バッテリ付きの RAM (BBRAM) と eFUSE のどちらに格納するかをビットストリーム オプションで選択できます。 **注意** : このビットが 1 にプログラムされている場合、AES キーが確認されない限りデバイスは使用できません。このビットがプログラムされている場合は、RMA (Return Material Authorization) による返品を受け付けていません。また Vivado による SPI/BPI フラッシュの間接プログラミング フローも使用できません。Vivado を使用してプログラムする場合は、この FUSE の前に外部コンフィギュレーション メモリをプログラムする必要があります。	なし **推奨** : 0 を保持し、顧客のセキュリティ要件を未決定にしておくことを推奨
2	RSA_AUTH	RSA 認証を強制します。 **注意** : このビットが 1 にプログラムされている場合、AES キーが確認されない限りデバイスは使用できません。このビットがプログラムされている場合は、RMA (Return Material Authorization) による返品を受け付けていません。また Vivado による SPI/BPI フラッシュの間接プログラミング フローも使用できません。Vivado を使用してプログラムする場合は、この FUSE の前に外部コンフィギュレーション メモリをプログラムする必要があります。	顧客のセキュリティ要件を未決定にしておく
4	SCAN_DISABLE	ザイリンクス テスト アクセスを無効にします。	なし (0 を保持)
5	CRYPT_DISABLE	復号化を永久的に無効にします。	なし (0 を保持)

- FUSE_SHAD_SEC[0:1] をプログラムしない場合 :
  - 暗号化は、ビットストリーム オプションで無効/有効にできる
  - eFUSE に格納されている AES キーを使用するか、バックアップ バッテリ付きの SRAM (BBRAM) に格納されている AES キーを使用するかをビットストリーム オプションで選択できる
- FUSE_SHAD_SEC[1:0] をプログラムする場合 :
  - 外部コンフィギュレーション ポート経由で FPGA をコンフィギュレーションできるのは eFUSE キーで暗号化したビットストリームのみとなります。

注意 : FUSE_SHAD_SEC[0] または RSA_AUTH をプログラムした場合、それぞれについて外部コンフィギュレーション ポート経由で FPGA をコンフィギュレーションできるのは、AES 暗号化されたビットストリームまたは RSA 認証を使用したビットストリームのみとなります。これにより、ザイリンクスのテスト ビットストリームや構築済みビットストリームを使用したデバイス コンフィギュレーションはできなくなります。このため、ザイリンクスは FUSE_SHAD_SEC[0] ビットまたは RSA_AUTH ビットがプログラムされているデバイスに対する RMA (Return Material Authorization) 要求は受け付けておらず、フラッシュの間接プログラムもサポートしていません。

暗号化キーと暗号化ビットストリームの生成

Vivado ツールで提供されているビットストリーム ジェネレーターの write_bitstream は、暗号化したビットストリームと暗号化していないビットストリームのどちらも作成できます。ビットストリームを AES で暗号化するには、write_bitstream のプロパティでビットストリーム暗号化を有効にするオプションを選択します。ユーザーは、独自の 256 ビットのキーをビットストリーム ジェネレーターに入力 (最も安全な方法であり、ザイリンクスで推奨) できますが、Vivado ツールにランダム キーを生成させることも可能です (非推奨)。これにより、ビットストリーム ジェネレーターで暗号化ビットストリーム ファイル (BIT) および暗号化キーファイル (NKY) が生成されます。表 6 に、XDC ファイルで定義可能な write_bitstream のプロパティおよびそれらの説明を示します。キー生成およびビットストリーム暗号化で使用する Vivado GUI の例については、『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 4] を参照してください。

表 6 : Write_bitstream プロパティ

Write_bitstream プロパティ	デフォルト値	設定可能な値	説明
BITSTREAM.ENCRYPTION.ENCRYPT	No	No または Yes	ビットストリームを暗号化します。
BITSTREAM.ENCRYPTION.ENCRYPTKEYSELECT	bbram	bbram または eFUSE	使用されている AES 暗号化キーの格納場所 (バックアップ バッテリ付き RAM (BBRAM) または eFUSE レジスタ) を示します。 **注記** : このプロパティは、Encrypt オプションが True に設定されている場合のみ有効です。
BITSTREAM.ENCRYPTION.KEYLIFE	32	4 ～ 2,147,483,647	AES-GCM 認証ビットストリームに使用すべき単一のキーに対する 128 ビット暗号化ブロックの数を示しています。デフォルトの 32 の場合、ビットストリーム サイズは 2 倍になります。
BITSTREAM.ENCRYPTION.KEY0	Pick	Pick または <256 ビット 16 進文字列>	Key0 は、ビットストリーム暗号化の AES 認証キーをセットします。このオプションを使用するには、最初に ENCRYPT オプションを Yes に設定する必要があります。
BITSTREAM.ENCRYPTION.KEYFILE	None	<文字列>	入力暗号化ファイル名 (ファイル拡張子 .nky) を指定します。このオプションを使用するには、最初に ENCRYPT オプションを Yes に設定する必要があります。
BITSTREAM.ENCRYPTION.RSAKEYLIFEFRAMES	8	8 ～ 1,247,483,647	RSA 公開キー認証する場合に、該当する AES-256 キーに使用すべきコンフィギュレーション フレーム数を指定します。コンフィギュレーション フレーム値に 8 を指定するのは、246 の暗号化ブロックのキーを使用するのと同じことです。
BITSTREAM.ENCRYPTION.STARTIV0	Pick	Pick または <32 ビット 16 進文字列>	最初の AES-GCM メッセージの最初の GCM カウント値を指定するために使用する初期化ベクター (IV) で、32 ビット 16 進数です。
BITSTREAM.ENCRYPTION.STARTIVOBFUSCATE	Pick	Pick または <128 ビット 16 進文字列>	難読初期化ベクター値 (Obfuscate IV0) を開始します。
BITSTREAM.AUTHENTICATION.AUTHENTICATE	No	Yes または No	RSA 認証の使用有無を示します。No の場合、AES-GCM が使用されます。

表 6 : Write_bitstream プロパティ (続き)

Write_bitstream プロパティ	デフォルト値	設定可能な値	説明
BITSTREAM.AUTHENTICATION.RSAPRIVATEKEYFILE	None	<文字列>	RSA-2048 暗号化されたビットストリームにサインするために使用される 2 つのキーを含んでいる OpenSSL .pem ファイルを指定します。
BITSTREAM.ENCRYPTION.OBFUSCATEKEY	Disable	Enable または Disable	ビットストリームを生成し、それらが eFuse または BBRAM (バックアップバッテリ付きの RAM (BBRAM) に書き込まれる前に、ビットストリームの暗号化に使用されるキーを難読化します。したがって、ユーザーは実際のカスタマー キーの代わりに難読化したキーをデバイス プログラマに提供することが可能になります。その後、デバイス プログラマは、その難読化されたキーを eFUSE または BBRAM に書き込み、指定した格納場所で obfuscated-key フラグを使用して難読化されていることを示します。

次の XDC ファイルのコードは、BBRAM キー ストレージと定義されたカスタム キーを使用した場合を示しています。ツールを使用して疑似ランダムのキーを生成する場合は、BITSTREAM.ENCRYPTION.KEY0 プロパティ (13 行目) をコメントアウトする必要があります。また、RSA 認証は有効に設定されており、RSA キーのフレーム数がデフォルト値の 8 に設定されていることを確認できます。RSA 認証機能を無効化する場合は、これらをコメント アウトしてください。ここにあるすべてのプロパティは、[Edit Device Properties] から選択/変更可能です。

```
 7 #Encryption settings
 8 set_property BITSTREAM.ENCRYPTION.ENCRYPT YES [current_design]
 9 #set_property BITSTREAM.ENCRYPTION.ENCRYPTKEYSELECT EFUSE [current_design]
10 set_property BITSTREAM.ENCRYPTION.ENCRYPTKEYSELECT BBRAM [current_design]
11 set_property BITSTREAM.ENCRYPTION.OBFUSCATEKEY Enable [current_design]
12 set_property BITSTREAM.ENCRYPTION.KEYLIFE 32 [current_design]
13 set_property BITSTREAM.ENCRYPTION.KEY0
256'h123456781234567812345678123456781234567812345678123456781234567812345678 [current_design]
14 set_property BITSTREAM.ENCRYPTION.STARTIV0 32'h876543218765432187654321876543321
[current_design]
15 set_property BITSTREAM.ENCRYPTION.STARTIVOBFUSCATE
32'hABCDABCDABCDABCDABCDABCDABCDABCD[current_design]
16
17 #Authentication settings
18 set_property BITSTREAM.AUTHENTICATION.AUTHENTICATE YES [current_design]
19 set_property BITSTREAM.ENCRYPTION.RSAKEYLIFEFRAMES 8 [current_design]
20
21
22 set_property BITSTREAM.AUTHENTICATION.RSAPRIVATEKEYFILE
{C:config/ultrascale/encryption/encryption_test/encryption_test.runs/impl_1/rsa.pem}
[current_design]
23
```

NKY ファイルの生成は、ビットストリーム生成と同時に実行されます。NKY ファイルは、ビット ファイルと同じ top_level 名を使用し、暗号化されたビットストリームと同じインプリメンテーション ディレクトリに配置されます。NKY ファイルのフォーマットは次のとおりです。

```
Device: <type>
EncryptKeySelect <type>;
StartIvObfuscate 32'h hex 00000000000000000000000000000000;
RsaPublicKeyDigest 384'h
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000;
Key0 256'h 0000000000000000000000000000000000000000000000000000000000000000;
StartIV0 32'h 00000000000000000000000000000000;
```

例 : (top.nky)

```
Device xcvu095;
EncryptKeySelect BBRAM;
StartIvObfuscate abcdabcdabcdabcdabcdabcdabcdabcd;
RsaPublicKeyDigest
6DF61AA1636F1FF8A35D064CED96918A6D4820D3CAEEFFA47F64F5C58AC2E539CA3A53103C86450EC576AD
595AB3A528;
Key0 123456781234567812345678123456781234567812345678123456781234567812345678;
StartIV0 876543218765432187654321000003f8;
Key1 44a619e399910767e68cb81bcbae831bd5d10a96e5a348420e9384eb0df06111;
StartIV1 b152ef23b9481138af45a21a000003d8;
.
.
.
Key4063 ddd9955fae4847a54d10a1c06a52171cf69e3593220018c2cac4ca56bf359f5c;
StartIV4063 7d94e42bf79ab77cce1a1d8300000382;
```

RSA 認証の PEM ファイル、構文例 :

```
-----BEGIN RSA PRIVATE KEY-----
```

```
MIIEpQIBAAKCAQEAvCMmT6/MM9LxXs7ZxybE4wKACvp0S2EpWy/q+wFkjeev/oT1EZkyRkeCLWKwLaTUeGxFYe
WCVFhpHH7PU9d/5HudIsVr/uJ8k/V7GASsj/8EL3O+RFoMdpsv6AFFD8desse3svR2d3yWlnrWLKfSd25DLqOg
5fHMauV5DwDpsrbUvBf/ZOW5JWd4iyi0oeK1/Dw/91AYiJoRWmKt6s3IH1ZkX4OfoXMBJ+SnVgV9NIm59lOb0v
d0ZZtNOqo1oX/Ekn93jwoD1UbHAWN90TfZSIAqsv2c4aeC342jKrHUq4cykK
.
.
.
xuTbhBadZaq8u8TGsXO3oPvI+p2tee5sNNoleJj3/gnkPtF9od5bqo8=
-----END RSA PRIVATE KEY-----
```

暗号化キーの読み込み

BBRAM および eFUSE 256 ビットの対称キーは、Vivado デバイス プログラマ ツールを使用して JTAG 経由でのみデバイスに読み込むことができます。UltraScale デバイスの場合、このキーを読み込むパスはデバイスに対して書き込み専用となります。いずれのキーもリード バックするための物理的なパスはありません。JTAG 経由でデバイスにキーを書き込んだ場合、キーの整合性チェックは JTAG 経由でデバイスに CRC32 の想定値を書き込むことで開始されます。デバイスは、実際の CRC32 整合性チェックは格納されたキーに対してデバイスによって (内部で) 算出され、JTAG ポートから受信した CRC32 の想定値と比較されます。デバイスは、実際のキー情報ではなく、合/否の結果を JTAG ポートへ出力して整合性ステータスを示します。キー用の物理的なリードバック パスを排除することで、格納されたキーのセキュリティが強化されます。

BBRAM キーのプログラミング ソリューション :

- Vivado デバイス プログラマ ツールと JTAG ケーブルを使用

  注記 : BBRAM ベースのキーの場合は、キーを書き込む前に、BBRAM にある既存のキーをゼロ化 (消去および検証) してください。

eFUSE キーのプログラミング ソリューション :

- Vivado デバイス プログラマ ツールと JTAG ケーブルを使用
- 『デバイス プログラマ向けの eFUSE プログラム』(XAPP1245) [参照 7]
- 『デバイス プログラマを使用した eFUSE のプログラム』(XAPP1260) [参照 8]
- デバイスのプリプログラミング サービスについては、Avnet 社へお問い合わせください。

推奨 : eFUSE ソリューションの場合、AES キーのインシステム プログラミング時に次の事項にも注意が必要です。

- FPGA 内の電源ノイズを最小限に抑えるため、コンフィギュレーション済みデザインの FPGA は避ける、あるいはクリアしてください。
- システムの電源ノイズを最小限に抑えるため、可能な場合にはボードレベルのシステム クロックを停止してください。

Vivado ハードウェア マネージャを使用して有効なハードウェア ターゲットへ接続した後、UltraScale FPGA を右クリックすると、設定した格納場所オプションに基づいて [Program BBR Key] または [Program eFUSE Registers] が選択できます (図 2 参照)。

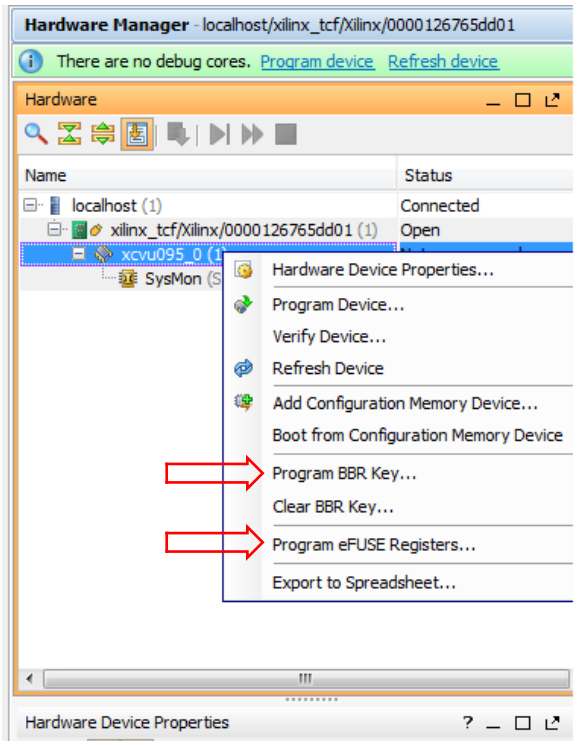


図 2 : Vivado ハードウェア マネージャでキーのプログラムを選択

BBRAM

[Program BBR Key] を選択すると、プロジェクト ディレクトリに最近生成された NKY ファイルを参照できます。NKY ファイルを追加した後には、キーの値を再確認して、デバイスにプログラムしようとしている適切な AES キー値であるかを検証可能です。(図 3 参照)。

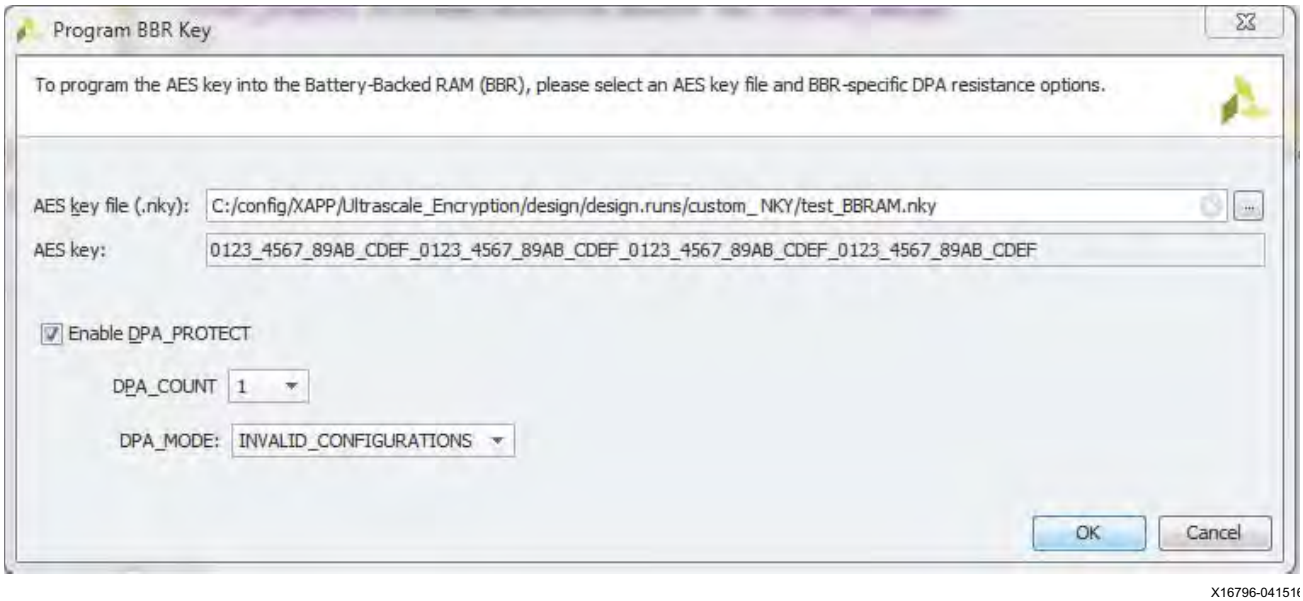


図 3 : BBRAM プログラミング GUI

注記 : NKY ファイルに KeyObfuscate フィールドが含まれている場合は、write_bitstream より先に BITSTREAM.ENCRYPTION.OBFUSCATEKEY プロパティが有効になっているため、ES-256 キーをプログラムする間に Vivado ソフトウェアによって eFUSE または BBRAM の難読化キー フラグが自動的にセットされます。

[Enable DPA_PROTECT] をオンにすると、BBRAM のコンフィギュレーション カウント機能を使用して DPA 攻撃から保護できます。

- [DPA_COUNT] は、コンフィギュレーション カウンターの最初のロード値を指定します。カウント値が 0 になると、BBRAM は消去されます。
- [DPA_MODE] は、[DPA_COUNT] をデクリメントする条件を指定します。選択肢は 2 つあり、[INVALID_CONFIGURATIONS] は標準的な DPA 設定で、[ALL_CONFIGURATIONS] はコンフィギュレーションごとにデクリメントするため、デバイスで使用するコンフィギュレーション数が固定されます。

eFUSE

[Program eFUSE Registers] を選択すると、ウィザードが表示されて、プログラムする NKY ファイルおよびさまざまな eFUSE レジスタの選択プロセスが指示されます。NKY または PEM ファイルを追加した後には、キーの値を再確認して、デバイスにプログラムしようとしている適切な AES キーおよび RSA キーの値であるかを検証可能です。(図 4 参照)。

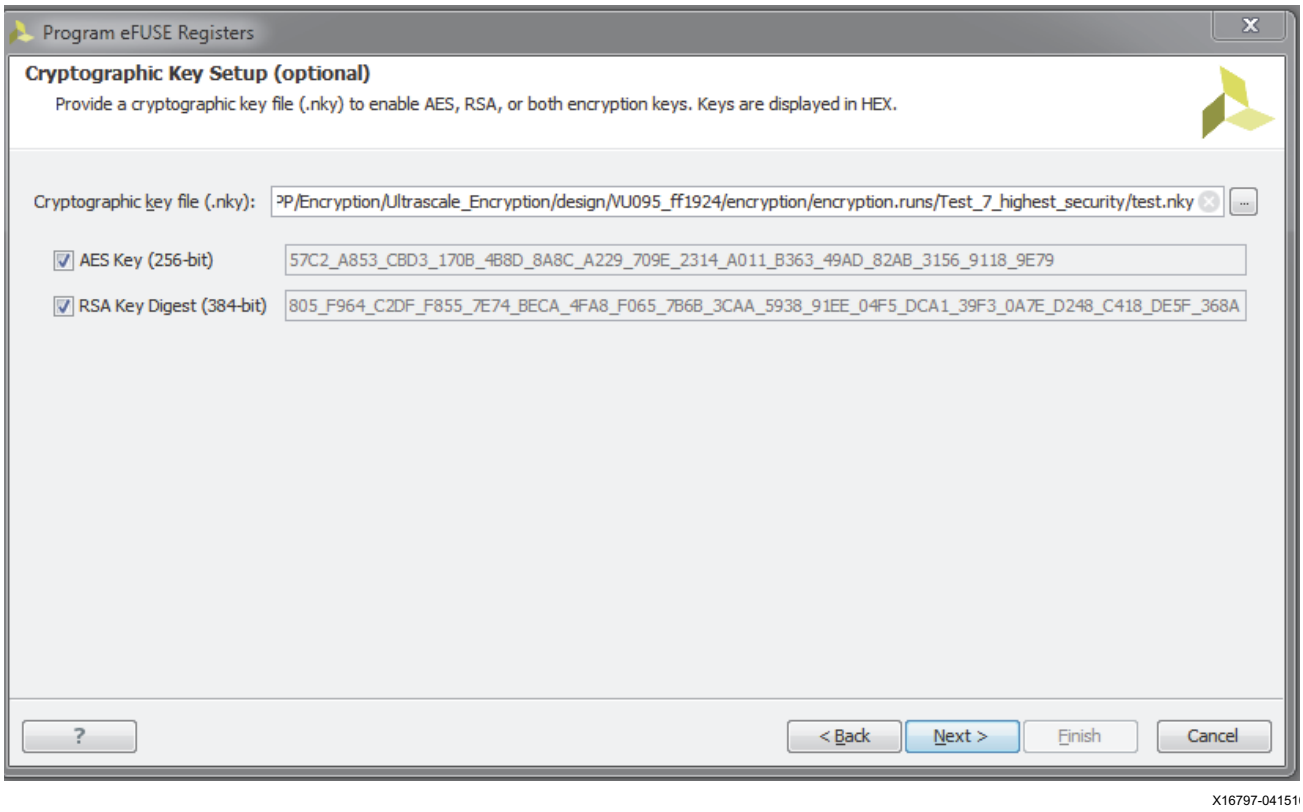
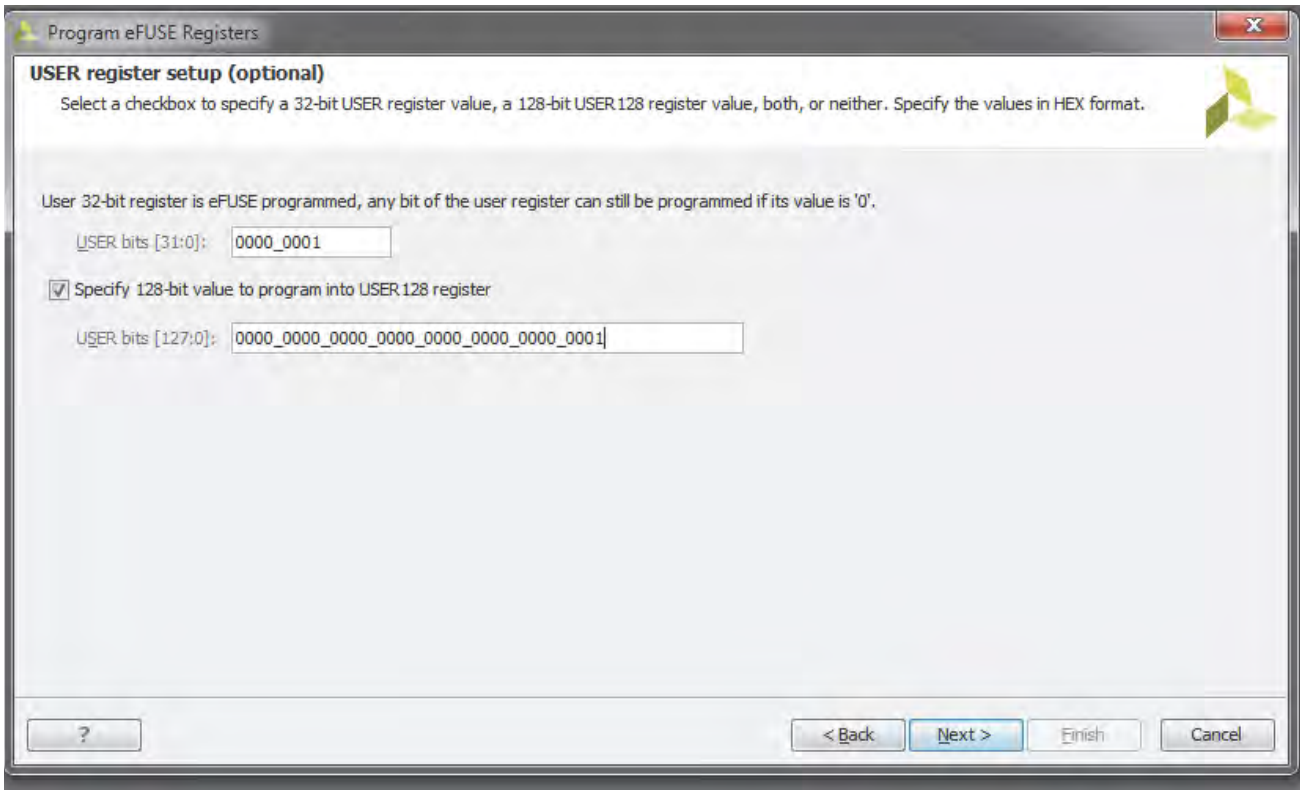


図 4 : eFUSE プログラミング暗号キーのセットアップ

図 5 に、ユーザー レジスタのセットアップ画面を示します。この画面では、FUSE_USER レジスタ ビットにプログラムする 32 ビットまたは 128 ビットの固有値を指定できます。これらのレジスタは、eFUSE_USR プリミティブを使用して FPGA ロジックから読み出し可能です



X16798-0415

図 5 : eFUSE プログラミング USER レジスタのセットアップ

図 6 に、制御レジスタのセットアップ画面を示します。この画面では、プログラムする FUSE_CNT レジスタ ビットを選択できます。これらのビットで、さまざまな eFUSE 制御レジスタの読み出し動作や書き込み動作を無効にすることでセキュリティ機能を備えることができます。

注記 : 制御レジスタ ビットの説明および推奨値は、7 ページの表 4 を参照してください。

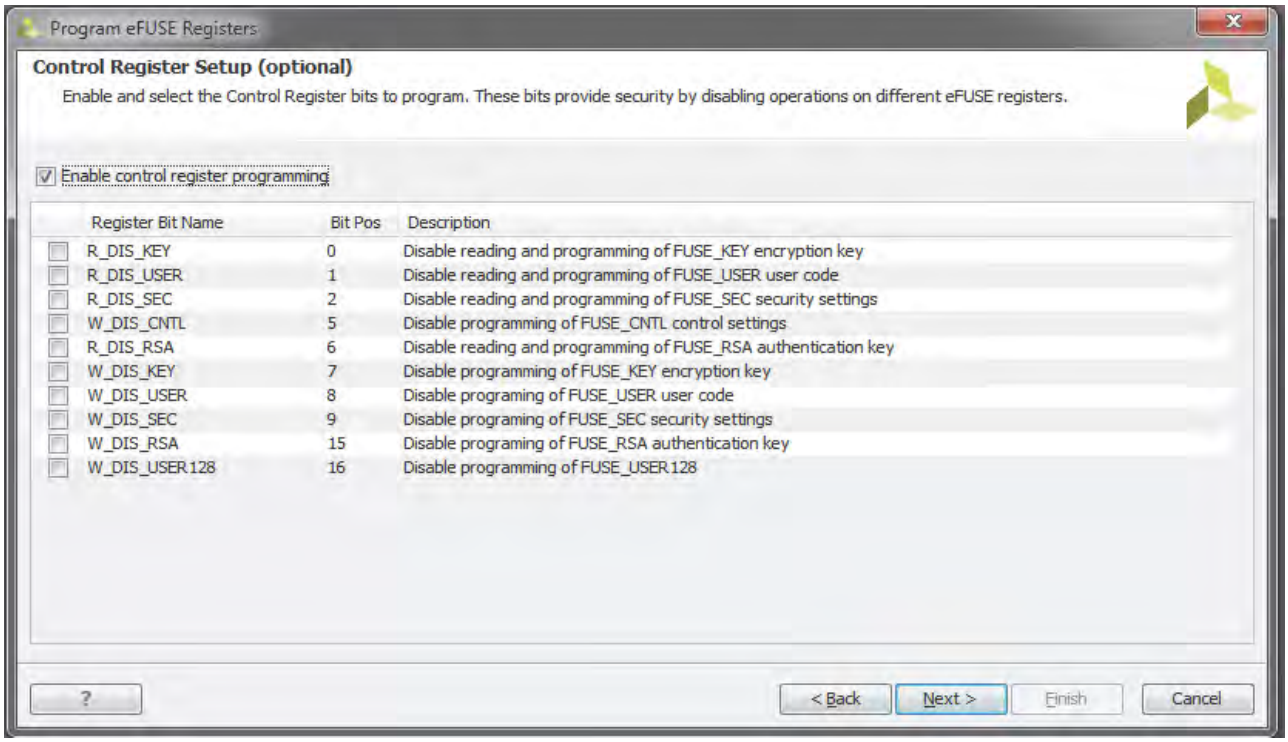
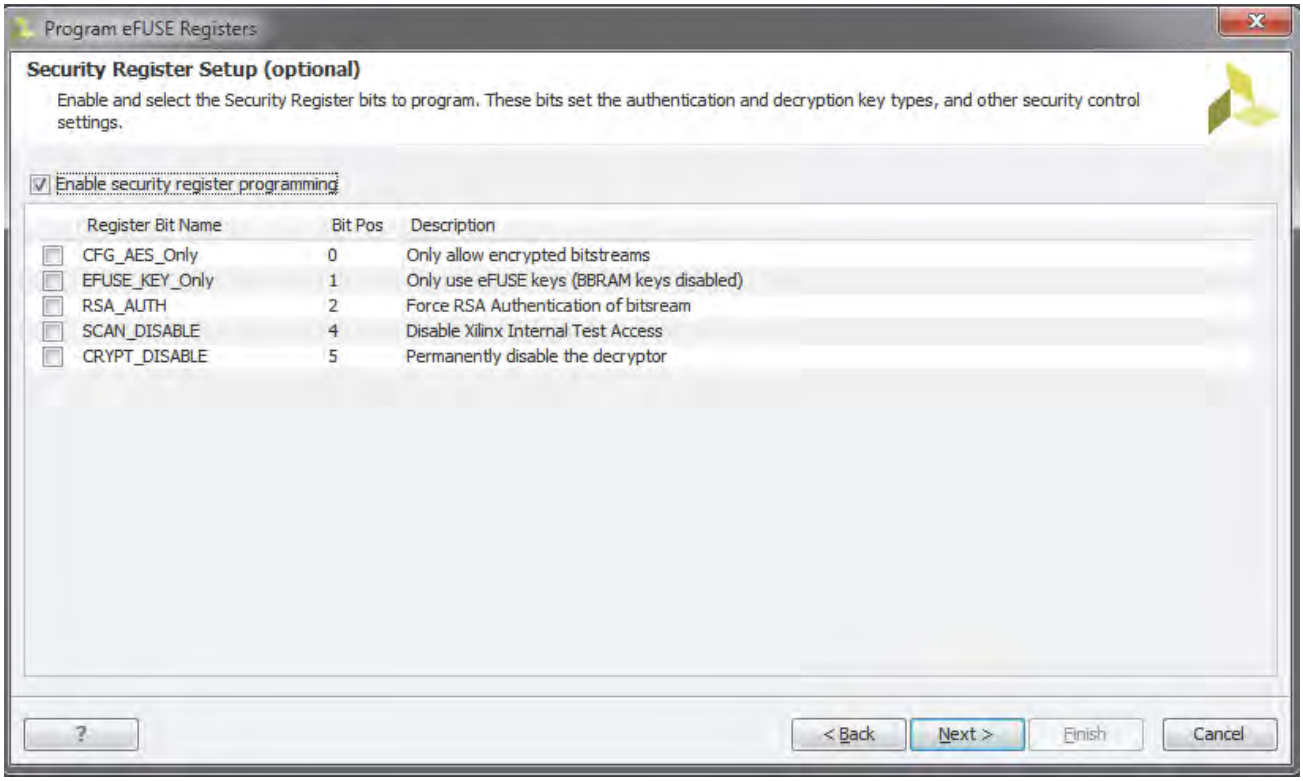


図 6 : eFUSE プログラミング制御レジスタのセットアップ

図 7 に、セキュリティ レジスタのセットアップ画面を示します。この画面では、プログラムする FUSE_SEC レジスタ ビットを選択できます。これらのビットでは、暗号化されたビットストリームのみを許可したり、RSA 認証を有効にすることでセキュリティを強化できます。

注記 : セキュリティ レジスタ ビットの説明および推奨値は、9 ページの表 5 を参照してください。

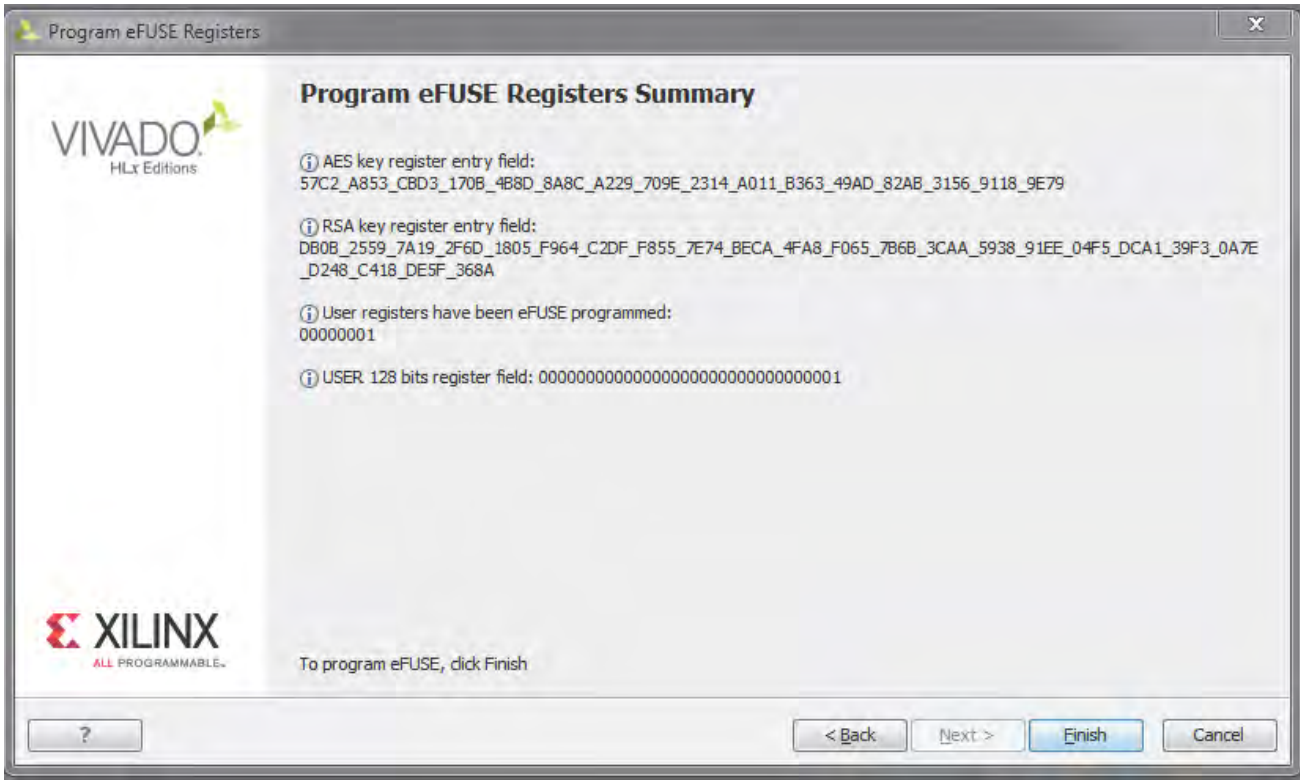


X16800-04151

図 7 : eFUSE プログラミング セキュリティ レジスタのセットアップ

注記 : NKY ファイルに KeyObfuscate フィールドが含まれている場合は、write_bitstream より先に BITSTREAM.ENCRYPTION.OBFUSCATEKEY プロパティが有効になっているため、ES-256 キーをプログラムする間に Vivado ソフトウェアによって eFUSE または BBRAM の難読化キー フラグが自動的にセットされます。

最後の画面 (図 8) は、選択したオプションが意図するものであるかを検証するためのサマリ画面となります。eFUSE レジスタはワンタイム プログラマブルであるため、一度プログラムするとその後の変更が不可であることに留意してください。



X16801-041

図 8 : サマリ画面

暗号化されたビットストリームの読み込み

デバイスに適切な暗号化キーをプログラムすると、暗号化ビットストリームを使用したデバイス コンフィギュレーションが可能になります。暗号化ビットストリームでコンフィギュレーションを行った後は、ビットストリームのセキュリティ設定にかかわらず、JTAG または SelectMAP リードバックによってコンフィギュレーション メモリを読み出すことはできません。デバイスに暗号化キーが含まれている場合でも、暗号化していないビットストリームでデバイスをコンフィギュレーション (FUSE_SHAD_SEC[0] ビットがプログラムされていない場のみ) できますが、その前に INIT_B または PROGRAM_B をアサートしてコンフィギュレーション メモリをクリアする必要があります。この場合、暗号化キーは無視されます。また、暗号化していないビットストリームでコンフィギュレーションを行った後は、リードバックが可能です (リードバックの write_bitstream セキュリティ設定で許可されている場合のみ)。この場合でも、デバイスから暗号化キーを読み出すことはできないため、「トロイの木馬」ビットストリームを使用して UltraScale FPGA の暗号化システムを無効にすることはできません。

サポートされているコンフィギュレーション方法が暗号化によって影響を受けることはありません。UltraScale FPGA では、圧縮と RSA 認証の両方を使用してビットストリームを作成できません。暗号化ビットストリームの読み込みには、JTAG、シリアル、SPI、BPI、SelectMAP、ICAPE3 から任意のコンフィギュレーション インターフェイスを使用できます。コンフィギュレーション後にリコンフィギュレーションするには、PROGRAM_B ピンをトグルする、電源を再投入する、あるいは IPROG または JPROGRAM 命令を与える必要があります。暗号化がオンの場合でも、フォールバック リコンフィギュレーションや IPROG リコンフィギュレーションを有効にできます。また、ICAPE3 プリミティブを使用したリードバックが可能です。V_{BATT} または V_{CCAUX} が維持されている限り、これらのイベントによって BBRAM 内のキーがリセットされることはありません。

暗号化ビットストリーム内のキーとデバイスに格納されたキーが一致しないとコンフィギュレーションがエラーとなり、INIT_B ピンが Low になって (フォールバックが有効の場合は High に戻る)、DONE ピンは Low のままになります。暗号化されたビットストリームでは、Tandem コンフィギュレーションやパーシャル リコンフィギュレーションなどの高度なコンフィギュレーション ソリューションがサポートされています。パーシャル ビットストリームは、暗号化されずに

ICAP へ、または暗号化されて (同じ AES キーを使用) 任意のコンフィギュレーション ポートへ送信できます (後者はユーザーによって明示的に無効化されていない限り可能)。セキュリティ レベル 2 を設定 (set_property BITSTREAM.READBACK.SECURITY Level2 [current_design] を使用) するか、FUSE_SHAD_SEC[0] cfg_aes_only ビットを 1 にプログラムすることで、外部コンフィギュレーション ポートからのパーシャル リコンフィギュレーションを阻止できます。

重要 : RSA 認証の暗号化ビットストリームは、SelectMAP、SPI、または BPI のいずれかのインターフェイスからプログラムされる必要があります。Vivado ハードウェア マネージャを使用して JTAG 経由で直接 RSA 認証のビットストリームをプログラムすることは許可されていません。RSA 認証をサポートする UltraScale FPGA デバイスとコンフィギュレーション モードの詳細は、『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570) [参照 3] の「RSA 認証」を参照してください。

ハードウェア検証

通常、開発者は AES キーが BBRAM または eFUSE ビットのいずれかに正しくプログラムされているかを検証します。検証手順のチェックリストを次に示します。

1. Vivado 2016.1 またはそれ以降のバージョンを使用してビットストリームを生成 : 暗号化されていないビットストリーム、ユーザー指定のキーで暗号化されたビットストリーム、すべて 1 のキーで暗号化されたビットストリーム、すべて 0 のキーで暗号化されたビットストリーム。
2. 生成されたビットストリームを確認して、暗号化を検証します。
3. ハードウェアをチェック : Vivado デバイス プログラマを使用して FPGA へ接続し、JTAG を介して暗号化されていない BIT ファイルをダウンロードします。デザインが予想どおり機能していることを確認します。
4. FPGA の復号化をテスト : すべて 0 のキーで暗号化された BIT ファイルをダウンロードします (eFUSE の場合)。
5. JTAG を介して AES キーをプログラム : eFUSE を使用する場合、まず最初に BBRAM キーを使用して手順 5 および 6 を実行し、有効性をチェックします。予想どおりの動作が確認できたら、最終的なテスト用に eFUSE をプログラムします。
6. キーをテスト : ユーザー指定のキーで暗号化された BIT ファイルをダウンロードします。
7. キーをテスト : すべて 0 のキーで暗号化された BIT ファイルをダウンロードします (エラーが予測される)。
8. キーをテスト : 暗号化されていない BIT ファイルをダウンロードします (セキュリティ設定によって結果は異なる)。
9. キーのセキュリティを確認 : キーが読み出し保護されていることを確認します。

まとめ

このアプリケーション ノートでは、UltraScale FPGA の AES 暗号化と認証規格について説明しています。また、キーを格納できる各オプションのメリット/デメリットも示しています。最も重要なことは、Vivado ソフトウェアして、暗号化ビット ファイルと認証/暗号キーを簡単に生成して、UltraScale FPGA にこれらのファイルをプログラムできることを説明しています。

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. 『Advanced Encryption Standard (AES)』(FIPS PUB 197)
2. 『The Galois/Counter Mode of Operation (GCM) 』(仕様書)
3. 『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570 : 英語版、日本語版)
4. 『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)

5. 『UltraScale FPGA での不正操作防止デザインの開発』(XAPP1098)
(ザイリンクスへお問い合わせください)
6. OpenSSL (www.openssl.org)
7. 『デバイス プログラマの eFUSE プログラム』(XAPP1245)
(ザイリンクスへお問い合わせください)
8. 『デバイス プログラマを使用した eFUSE のプログラム』(XAPP1260 : 英語版、日本語版)
9. 『UltraScale FPGA およびコンフィギュレーション モード別の RSA 認証のサポートの有無』(XCN 15038)
10. 「UltraScale RSA 認証のデザイン アドバイザリ - RSA 認証を使用する UltraScale デバイスでコンフィギュレーション インターフェイスの幅が狭いとビットストリーム認証エラーが発生する」(ザイリンクス アンサー 65792)
11. 『UltraScale FPGA での不正操作防止デザインの開発』(XAPP1098)
(ザイリンクスへお問い合わせください)
12. 『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570 : 英語版、日本語版)
13. 『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)
14. 『デバイス プログラマの eFUSE プログラム』(XAPP1245)
(ザイリンクスへお問い合わせください)
15. 『デバイス プログラマを使用した eFUSE のプログラム』(XAPP1260 : 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
2016 年 6 月 2 日	1.0	初版

お読みください : 重要な法的通知

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。