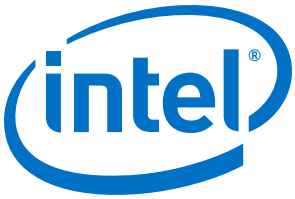


AN 556: インテル FPGA におけるデザイン・セキュリティー機能の使用

この翻訳版は参照用であり、翻訳版と英語版の内容に相違がある場合は、英語版が優先されるものとします。翻訳版は、資料によっては英語版の更新に対応していない場合があります。最新情報につきましては、必ず英語版の最新資料をご確認ください。

更新情報
フィードバック

AN-556 | 2019.11.12
日本語版の最新資料: PDF | HTML

目次

AN-556 | 2019.11.12

フィードバック

インテル® FPGA におけるデザイン・セキュリティー機能の使用

このアプリケーション・ノートでは、デザイン・セキュリティー機能を 40、28、および 20nm インテル® FPGA で使用し、デザインを不正なコピー、リバース・エンジニアリング、およびコンフィグレーション・ファイルの改ざんから保護する方法について説明します。このアプリケーション・ノートでは、40nm、28nm、および 20nm FPGA デザインのセキュリティー機能に対するハードウェア要件およびソフトウェア要件のほか、安全なコンフィグレーション・フローの実装手順を提供します。

注意: このアプリケーション・ノートでは用語として、「40nm」、「28nm」、または「20nm」 FPGA を使用しています。次の表では、サポートされる FPGA とその適用デバイスを一覧表示します。

表 1. サポートされる FPGA

FPGA	デバイス
40nm	Arria® II および Stratix® IV
28nm	Arria V、Cyclone® V、および Stratix V
20nm	インテル® Arria 10 およびインテル Cyclone 10 GX

商業用および軍事用の環境では、デザイン・セキュリティーはデジタル設計者にとって重要な考慮事項です。FPGA が役割を果たすシステム・コンポーネントがより大規模で重要なものになるにつれて、デザインを不正なコピー、リバース・エンジニアリング、および改ざんから保護することが重要です。インテル FPGA では、256 ビットの Advanced Encryption Standard (AES) アルゴリズムを使用してコンフィグレーション・ビットストリームを暗号化することで、こうした問題に対処しています。

表 2. サポートされているインテル FPGA での AES モード

FPGA	AES モード
40nm	カウンターモード (CTR)
28nm	暗号ブロックチェーン (CBC)
20nm	CTR および鍵付きハッシュメッセージ認証コード (HMAC)

デバイスの動作中、FPGA では、コンフィグレーション・データを SRAM コンフィグレーション・セルに格納します。SRAM メモリーは揮発性のため、SRAM セルへのコンフィグレーション・データのロードは、デバイスの電源投入のたびに行う必要があります。コンフィグレーション・データは通常、フラッシュメモリーやコンフィグレーション・デバイスなどの外部メモリーソースから FPGA に送信されます。コンフィグレーション・データは、メモリーソースから FPGA への送信中にインターセプトすることが可能です。コンフィグレーション・データが暗号化されていないと、インターセプトしたコンフィグレーション・データを使用して、他の FPGA をコンフィグレーションすることができます。

インテル FPGA では、揮発性と不揮発性の両方のキーストレージを提供しています。デザイン・セキュリティー機能を使用すると、キーは FPGA に格納されます。セキュリティー・モードに応じて、FPGA のコンフィグレーションには、同じキーで暗号化されたコンフィグレーション・ファイルを使用するか、またはボードテスト用には、通常のコンフィグレーション・ファイルを使用します。

デザイン・セキュリティー機能が使用可能なのは、FPGA のコンフィグレーション時に、ファースト・パッシブ・パラレル (FPP) コンフィグレーション・スキームで、外部ホスト (MAX® II または MAX V デバイスもしくはマイクロプロセッサー) を使用する場合か、アクティブシリアル (AS) またはパッシブシリアル (PS) コンフィグレーション・スキームを使用する場合です。

関連情報

- AN 680: Product Security Features for Altera Devices
  製品セキュリティー機能に関する詳細を提供しています。
- Configuration, Design Security, and Remote System Upgrades in Arria II Devices
  Arria II デバイスのデザイン・セキュリティーに関する詳細を提供しています。
- Configuration, Design Security, and Remote System Upgrades in Stratix IV Devices
  Stratix IV デバイスのデザイン・セキュリティーに関する詳細を提供しています。
- Configuration, Design Security, and Remote System Upgrades in Arria V Devices
  Arria V デバイスのデザイン・セキュリティーに関する詳細を提供しています。
- Configuration, Design Security, and Remote System Upgrades in Cyclone V Devices
  Cyclone V デバイスのデザイン・セキュリティーに関する詳細を提供しています。
- Configuration, Design Security, and Remote System Upgrades in Stratix V Devices
  Stratix V デバイスのデザイン・セキュリティーに関する詳細を提供しています。
- インテル Arria 10 デバイスにおけるコンフィグレーション、デザイン・セキュリティー、およびリモート・システム・アップグレード
  インテル Arria 10 デバイスのデザイン・セキュリティーに関する詳細を提供しています。
- Configuration, Design Security, and Remote System Upgrades in Intel Cyclone 10 GX Devices
  インテル Cyclone 10 デバイスのデザイン・セキュリティーに関する詳細を提供しています。
- インテル MAX 10 FPGA のコンフィグレーション・デザイン・セキュリティー

デザイン・セキュリティー機能の概要

インテル FPGA のデザイン・セキュリティー機能では、不正コピー、リバース・エンジニアリング、および改ざんからの保護を行います。次の表で示すデザイン手法では、ソリューションをセキュリティーで保護します。

20nm FPGA で追加されたセキュリティー機能をイネーブルするには、ヒューズを書き込むか、コンフィグレーション・ビットストリーム内のオプションビットの設定で、スタンドアロンの Qcrypt ツールまたはインテル Quartus® Prime Convert Programming File ツールを使用します。改ざん防止ビットおよび JTAG セキュアモードのイネーブルは、20nm FPGA でのみ、個別に行うことができます。

表 3. 40nm および 28nm FPGA のデザイン・セキュリティー手法

注意: 改ざん防止ビットをイネーブルすると、40nm および 28nm FPGA でテストモードがディスエーブルされます。テストモードのディスエーブルは元に戻すことができないため、インテルによる故障解析の実行ができなくなります。改ざん防止ビットをイネーブルするには、*改ざん防止ビット・プログラミングのイネーブルの手順*の項を参照してください。

デザイン・セキュリティー要素	40nm FPGA	28nm FPGA(1)
不揮発性キー	不揮発性キーは、デバイス内のヒューズに安全に格納されています。独自のセキュリティー機能によりこのキーの特定が困難になります。	
揮発性キー	揮発性キーは、デバイス内のバッテリー・バックアップ RAM に安全に格納されています。独自のセキュリティー機能によりこのキーの特定が困難になります。	
キーの生成	ユーザー提供の 256 ビット文字列 2 つが処理されて、デバイスにプログラムされている 256 ビットキーが生成されます。	ユーザー提供の 256 ビットキーは、デバイスにプログラムされる前に、一方向関数によって処理されます。
キーの選択	ユーザーは、セキュリティー・キー・タイプのうちどちらか 1 つ (不揮発性キーまたは揮発性キー) のみをデバイスに設定します。	
改ざん防止モード	改ざん防止モードでは、暗号化されていないコンフィグレーション・ファイルで FPGA がロードされないようにします。このモードをイネーブルすると、FPGA のロードは、ユーザーのキーで暗号化されたコンフィグレーションでのみ行うことができます。暗号化されていないコンフィグレーションおよび誤ったキーで暗号化されたコンフィグレーションでは、コンフィグレーションは正常に行われません。このモードをイネーブルするには、デバイス内でヒューズを設定してください。	
コンフィグレーションのリードバック	いずれのデバイスでも、コンフィグレーション・リードバック機能はサポートされていません。コンフィグレーション・リードバック機能は、暗号化されていないコンフィグレーション・データのリードバックを実行不可能にします。	

表 4. 20nm FPGA のデザイン・セキュリティー手法

デザイン・セキュリティー要素	説明
不揮発性キー	不揮発性キーは、デバイス内のヒューズに安全に格納されています。独自のセキュリティー機能によりこのキーの特定が困難になります。
揮発性キー	揮発性キーは、デバイス内のバッテリー・バックアップ付き RAM に安全に格納されています。独自のセキュリティー機能によりこのキーの特定が困難になります。
キーの生成	ユーザー提供の 256 ビットキーは、デバイスにプログラムされる前に、一方向関数によって処理されます。
キーの選択	揮発性と不揮発性の両方のキーをデバイス内に存在させることができます。ユーザーは、使用するキーを選択するため、暗号化されたコンフィグレーション・ファイル内のオプションビットを設定します。これは、Convert Programming File ツールまたは Qcrypt ツールを介して行います。
改ざん防止モード	改ざん防止モードでは、暗号化されていないコンフィグレーション・ファイルで FPGA がロードされないようにします。このモードをイネーブルにすると、FPGA のロードは、ユーザーのキーで暗号化されたコンフィグレーションのみで行うことができます。暗号化されていないコンフィグレーションや誤ったキーで暗号化されたコンフィグレーションは、正常に行われません。このモードをイネーブルするには、デバイス内でヒューズを設定してください。

continued...

(1) 28nm FPGA で改ざん防止ビットをイネーブルする場合、デバイスは JTAG セキュアモードです。

デザイン・セキュリティー要素	説明
コンフィグレーションのリードバック	いずれのデバイスでも、コンフィグレーション・リードバック機能はサポートされていません。セキュリティーの観点から、これにより、暗号化されていないコンフィグレーション・データのリードバックが実行不可能になります。
セキュリティー・キーの制御	異なる JTAG 命令およびセキュリティー・オプションを Qcrypt ツールで使用することにより、柔軟に揮発性または不揮発性キーの使用を恒久的または一時的にディスエーブルすることができます。また、揮発性キーをロックすると、その上書きまたは再プログラムを防止できます。
JTAG アクセス制御	JTAG アクセス制御のさまざまなレベルをイネーブルするため、OTP ヒューズまたはオプションビットの設定を、コンフィグレーション・ファイル内で Qcrypt ツールを使用して行います。 1. フル・コンフィグレーションまたはパーシャル・コンフィグレーションは、HPS のみを介して行うように強制します。 2. 外部 JTAG ピンまたは HPS JTAG をバイパスします。この機能によって外部 JTAG または HPS JTAG アクセスがディスエーブルされますが、ロックの解除は、内部コアアクセスを介して行うことができます。(2) 3. 外部 JTAG ピンからのすべての AES キーに関連する JTAG 命令をディスエーブルします。 4. 限られた必須 JTAG 命令セットのみが、外部 JTAG からアクセスできるようにします。JTAG セキュアモードと同様です。

注意: 上記およびその他のセキュリティー機能について詳しくは、インテル FPGA 技術サポートにお問い合わせください。

関連情報

- インテル Arria 10 およびインテル Cyclone 10 GX Qcrypt セキュリティー・ツール (8 ページ)
- 改ざん防止ビット・プログラミングのイネーブルの手順 (23 ページ)
- 安全なコンフィグレーション・フローの実装手順 (13 ページ)
- 28nm および 20nm FPGA の JTAG セキュアモード (31 ページ)

セキュリティー暗号化アルゴリズム

インテル FPGA に備えられている専用の AES 復号化ブロックでは、コンフィグレーション・ビットストリームの解読を FPGA デバイスのコンフィグレーション前に行うことができます。28nm FPGA では、AES ブロックを CBC モードで使用し、40nm および 20nm FPGA では、AES ブロックを CTR モードで使用します。さらに、20nm デバイスでは、標準の NIST CTR モードの暗号化に対するサイドチャネル攻撃を軽減する技術が実装されています。セキュリティー機能を使用しない場合、AES 復号化はバイパスされます。FPGA の AES 実装は、連邦情報処理規格 FIPS-197 に準拠しています。

関連情報

コンピューター・セキュリティー・リソース・センター (CSRC)
AES アルゴリズムについては、*Federal Information Processing Standards Publication FIPS-197* または *AES Algorithm (Rijndael) Information* を参照してください。FPGA の AES 検証については、*Advanced Encryption Standard Algorithm Validation List* を参照してください。

(2) インテル Cyclone 10 GX では、HPS および HPS JTAG Bypass による強制フル・コンフィグレーションまたはパーシャル・コンフィグレーションはサポートしていません。

不揮発性および揮発性キーのストレージ

インテル FPGA では、揮発性と不揮発性の両方のキーストレージを提供しています。揮発性キーのストレージレジスターは、プログラム可能かつ消去可能です。キーレジスターの内容は、電源を入れ直してもコイン型電池を使って保持されます。不揮発性キーレジスターは、ヒューズベースかつワンタイム・プログラマブルです。

注意: コイン型リチウム電池が揮発性キーのストレージに使用される例としては、BR1220 (–30°C から 80°C) や BR2477A (–40°C から 125°C) があります。

表 5. 揮発性キーと不揮発性キーの比較

オプション	揮発性キー	不揮発性キー
キーの長さ	256 ビット	256 ビット
キーのプログラマビリティー	再プログラム可能かつ消去可能なキー	ワンタイム・プログラマブル・キー
外部バッテリー	要	不要
キー・プログラミング方法(3)	オンボード	オンボードとオフボードの両方(4)
デザイン保護(5)	コピー、リバース・エンジニアリング、および改ざんに対して安全	

(3) キー・プログラミングは JTAG インターフェイスを介して実行されます。JTAG インターフェイスには、有効な MSEL ピン設定を使用する必要があります。

(4) 不揮発性キーヒューズのプログラミングでは、通常動作中に FPGA で使用される標準電圧源を使用します。追加の電圧レールは、不揮発性キーのプログラミングには必要ありません。

(5) 揮発性キーの改ざん防止機能が使用可能なのは、Arria II、Arria V、Cyclone V、Stratix V、インテル Arria 10、およびインテル Cyclone 10 GX デバイスのみです。

キー・プログラミング

表 6. キー・プログラミング方法

プログラミング手順	方法	プログラミング・ツール/サポート
オンボード・プログラミング	プロトタイプ作成	インテル FPGA イーサネット・ケーブル [6]、JTAG Technologies [7]、インテル FPGA パラレル・ポート・ケーブル [8]、インテル FPGA ダウンロード・ケーブル [9] 、およ インテル FPGA ダウンロード・ケーブル II [10]。
	量産	JTAG Technologies*
オフボード・プログラミング	プロトタイプ作成	System General*
	量産	System General

キー・プログラミングでは、次の定義を使用します。

- **オンボード:** デバイスがボード上でプログラムされる手順
- **オフボード:** デバイスが別のプログラミング・システム上でプログラムされる手順
- **プロトタイプ作成:** 特定の方法が正しく動作するかどうかを検証するために初期に使用される方法
- **量産:** 大量生産に使用される方法

注意: 他のサードパーティーの不揮発性キー・プログラミングでは、JTAG TCK パルス幅 (期間) を調整して、適切なポリフューズ・プログラミングを行ってください。詳細は 表 10 (11 ページ) を参照してください。

関連情報

インテル FPGA 技術サポート
プログラミング・サポートに関する情報を提供しています。

インテル Arria 10 およびインテル Cyclone 10 GX Qcrypt セキュリティー・ツール

Qcrypt ツールは、スタンドアロンの暗号化ツールとして、インテル Arria 10 およびインテル Cyclone 10 GX FPGA コンフィグレーション・ビットストリーム・ファイルの暗号化および復号化に使用できます。Qcrypt ツールはまた、スクリプトを介して HPS ブートイメージの暗号化に使用することもできます。さまざまな種類のセキュリティー設定で、インテル Quartus Prime グラフィカル・ユーザー・インターフェイスから現在アクセスできないものは、Qcrypt ツールを通して設定することができます。

---

(6) インテル FPGA イーサネット・ケーブルでは、揮発性キーと不揮発性キー両方のプログラミングをサポートしています。

(7) JTAG Technologies*では、揮発性キーと不揮発性キー両方のプログラミングをサポートしています。

(8) インテル FPGA パラレル・ポート・ケーブルでは、揮発性キーのプログラミングのみをサポートしています。

(9) インテル FPGA ダウンロード・ケーブルでは、揮発性キーのプログラミングのみをサポートしています。ただし例外として、20nm FPGA では、揮発性キーと不揮発性キーの両方のプログラミングをサポートしています。

(10) インテル FPGA ダウンロード・ケーブル II では、揮発性キーと不揮発性キーの両方のプログラミングをサポートしています。

Qcrypt ツールでは、ロウ・バイナリー・ファイル (.rbf) のみが暗号化および復号化しますが、.sof および .pof ファイルなどのその他のコンフィグレーション・ファイルの暗号化および復号化はされません。暗号化フローの間、Qcrypt ツールでは認証タグを生成し、同時に .rbf ファイルを暗号化します。認証タグによって、コンフィグレーション・ビットストリームの変更や改ざんが防止されます。Qcrypt ツールを使用すると、暗号化と復号化の他にもさまざまなセキュリティーの機能や設定のイネーブルと設定ができるようになります。セキュリティーの機能や設定を .rbf ファイルに組み込むことで、さまざまな種類のセキュリティー機能をインテル Arria 10 およびインテル Cyclone 10 GX デバイスで柔軟に使用することができます。セキュリティー・ヒューズを恒久的に書き込む必要はありません。.ekb ファイルまたは暗号化されたコンフィグレーション・ファイル (.rbf 以外) を生成するには、インテル Quartus Prime Convert Programming File ツールを使用してください。

注意: Qcrypt ツールはライセンス保護されていないので、すべての インテル Quartus Prime 開発ソフトウェアのユーザーに使用いただけます。

関連情報

- デザイン・セキュリティー機能の概要 (4 ページ)
- 改ざん防止ビット・プログラミングのイネーブルの手順 (23 ページ)
- 安全なコンフィグレーション・フローの実装手順 (13 ページ)
- 28nm および 20nm FPGA の JTAG セキュアモード (31 ページ)
- Qcrypt ツールのオプション (9 ページ)
  Qcrypt ツールの機能に関する詳細情報を提供しています。
- AN 759: Intel Arria 10 SoC Secure Boot User Guide
  HPS ブートイメージの暗号化に関する詳細情報を提供しています。

Qcrypt ツールの使用

次のコマンドを使用して、.rbf ファイルを暗号化および復号化します。このコマンドは、高度なセキュリティー・オプションを設定する唯一の方法です。

```
qcrypt [options] <input_file.rbf> <output_file.rbf>
```

Qcrypt ツールのオプション

表 7. Qcrypt ツールの基本オプション

基本オプション	説明
--encrypt	デフォルト動作で input_file.rbf を暗号化します。
--decrypt	input_file.rbf を復号化して、元のビットストリームを取得します。復号化後の .rbf が元のビットストリームと異なるのは、セキュリティー・オプションがイネーブルされていた場合です。明示的にこれらのセキュリティー・オプションをレベル 0 にリセットすれば、復号化された .rbf を暗号化前の元の .rbf と一致させることができます。元の .rbf ファイルは、復号化後のファイルとわずかに異なりますが、その違いは無視できます。
--keyfile=<KEY_FILE>	このキーファイルのデフォルト名は keyfile.key です。このオプションを使用すると、keyfile.key に別の名前を指定できます。キーファイルは、現在のプロジェクト・ディレクトリーにあり、そこには input_file.rbf も格納されています。キーファイルの例は、インテル Quartus Prime 開発ソフトウェアを使用したシングルデバイス .ekp ファイルの生成およびコンフィグレーション・ファイルの暗号化 (14 ページ) を参照してください。

continued...

基本オプション	説明
--keyname=<KEY_NAME>	使用する名前付きキーをキーファイルから指定します。デフォルトで、ツールではキーファイルの最初のキーを使用します。
--keystore=<types of key>	使用するセキュリティー・キーを指定します。 • otp (不揮発性キー) • battery (揮発性キー) One-time programmable (otp) がデフォルト値です。
--iv=<HEX_VALUE>	オプションのシード値。非ランダム初期化ベクトル (IV) が作成されます。デフォルトでは、.rbf が暗号化されるたびに暗号化された異なる.rbf が生成されます。このオプションでは、シード値を指定して、同じ .rbf の生成を確実に行うことができます。これには同じ --iv 値を使用します。HEX_VALUE には、任意の 32 ビット 16 進値を指定できます。

表 8. Qcrypt ツールのセキュリティー・オプション

セキュリティー・オプション	説明
--lockto=<FILE_NAME.qlk>	認証を以前の対応ベース・ビットストリームにロックします。 .qlk ファイルが自動作成されるのは、CvP コア・イメージ・ビットストリームなどの基本コンフィグレーション・ファイルが暗号化されるときです。このオプションを使用して、後続のコア CvP またはパーシャル・リコンフィグレーション・イメージを基本コンフィグレーションのみで使用可能にします。これにより、後続のビットストリームが、誤った (しかしそうでなければ認証されている) ベース・ビットストリームを介してロードされるのを防ぐことができます。
--no-lockto	必須の --lockto 要件をオーバーライドします。
--epof-only=[0:3]	暗号化および認証されたビットストリームのみを外部コンフィグレーションに使用できるようにします。
--no-config=[0:3]	外部ピンからのコンフィグレーションをディスエーブルします。このオプションを設定すると、コンフィグレーションの制御は内部 HPS によってのみ行うことができます。 *注意:* このセキュリティー・オプションは、インテル Cyclone 10 GX ではサポートされていません。
--no-pr=[0:3]	外部パーシャル・コンフィグレーションをディスエーブルします。
--no-jtag-key=[0:3]	キー関連の JTAG 命令をディスエーブルします。
--no-jtag-ext=[0:3]	JTAG セキュアモードをイネーブルします。
--no-jtag=[0:3]	外部 JTAG ピンを BYPASS モードに強制します。
--no-hps-jtag=[0:3]	内部 HPS JTAG を BYPASS モードに強制します。 *注意:* このセキュリティー・オプションは、インテル Cyclone 10 GX ではサポートされていません。
--no-otp-key=[0:3]	不揮発性 OTP ヒューズキーの使用をディスエーブルします。
--no-battery-key=[0:3]	バッテリー・バックアップ付きキーの使用をディスエーブルします。
--lock-battery-key=[0:3]	バッテリー・バックアップ付き揮発性キーの変更または上書きを防止します。
--secure=[2:3]	テストモード<default=2> をディスエーブルします。

関連情報

- インテル Arria 10 およびインテル Cyclone 10 GX Qcrypt セキュリティー・ツール (8 ページ)
- AN 759: Intel Arria 10 SoC Secure Boot User Guide
  HPS ブートイメージの暗号化に関する詳細情報を提供しています。

Qcrypt ツール・セキュリティー・オプションのセキュリティー・レベル

Qcrypt ツールを使用すると、表 8 (10 ページ) にあるセキュリティー・オプションのセキュリティー・レベルを柔軟に決定することができます。最小または最大要件は、セキュリティー・レベルの 0 から 3 の範囲で指定できます。

表 9. Qcrypt ツール・セキュリティー・オプションのセキュリティー・レベル

セキュリティー・レベル	説明
0	セキュリティー機能はイネーブルされません。対応する OTP ヒューズによる場合は例外です。
1	セキュリティー機能がイネーブルされるのは、現在のフル・コンフィグレーションまたはパーシャル・リコンフィグレーションの開始から次のフル・コンフィグレーションの開始までです。
2	セキュリティー機能がイネーブルされるのは、次回のパワーオンリセットまでです。さらに、セキュリティー機能によって通常は防止されている操作が前回のパワーオンリセット以降に行われた場合は、コンフィグレーションは進行しません。
3	セキュリティー機能の恒久的有効化のために、デバイス内の対応ヒューズを燃焼していない限り、コンフィグレーションは進行しません。

セキュリティー・レベル 2 によって提供されるセキュリティーのレベルは、対応 OTP セキュリティー・ヒューズを設定するのとほぼ同じくらい強力ですが、ある程度の柔軟性があります。たとえば、JTAG の使用が製造テストやデバッグに必要な場合、JTAG を完全にディスエーブルしても、安全な (暗号化された) ビットストリームをデバイスにロードすることができます。さらに、保護されたビットストリームのロードは、以前に何らかの種類の JTAG コマンドでプローブされたデバイスには行わなくても良いです。

インテルでは、最も厳格なセキュリティー・レベルを各オプションに対して、デザイン要件と矛盾しない形で使用することをお勧めしています。

注意: Qcrypt ツールに関する情報は、`--help` オプションを使用して参照してください。

ハードウェアおよびソフトウェア要件

デザイン・セキュリティー機能を使用すると、揮発性または不揮発性のキーが FPGA に格納されます。このキーのプログラムは、暗号化されたコンフィグレーション・ファイルを使用して FPGA をコンフィグレーションする前に行う必要があります。

ハードウェア要件

次の表で示す仕様は、キー・プログラミングを正常に行うために従う必要があります。

表 10. キー・プログラミングの仕様

パラメーター	キー・プログラミング・モード	
	不揮発性キー	揮発性キー
TCK 周期	10µs ± 1µs (11)	—
周囲温度	25°C ± 5°C	25°C ± 5°C
電圧 (VCCBAT)	—	(12)

(11) 40nm および 28nm FPGA のみに適用されます。

(12) 揮発性キーを使用しない場合、VCCBAT 接続については、それぞれのデバイスファミリーのピン接続ガイドラインを参照してください。

V_{CCBAT} は揮発性キーのストレージ専用電源で、V_{CCIO} や V_{CC} などの他のオンチップ電源との共有はしません。V_{CCBAT} は、オンチップ電源の状態にかかわらず、揮発性レジスターに電力を継続的に供給します。

注意: 電源投入後、デバイスによるパワーオンリセット (POR) の終了を待ってから、キー・プログラミングを開始してください。揮発性 Encryption Key Programming (**.ekp**) ファイルをプログラムするときに検証エラーが発生する可能性があるのは、VCCBAT ピンが GND に接続されている場合です。VCCBAT ピンは、V_{CCBAT} 電圧に接続してください。詳しくは、各デバイスファミリーのピン接続ガイドラインを参照し、適切な動作を確保してください。

関連情報

- Device Datasheet for Arria II Devices
  JTAG、POR、および電圧仕様に関する詳細情報を提供しています。
- DC and Switching Characteristics for Stratix IV Devices
  JTAG、POR、および電圧仕様に関する詳細情報を提供しています。
- Arria V Device Datasheet
  JTAG、POR、および電圧仕様に関する詳細情報を提供しています。
- Cyclone V Device Datasheet
  JTAG、POR、および電圧仕様に関する詳細情報を提供しています。.
- Stratix V Device Datasheet
  JTAG、POR、および電圧仕様に関する詳細情報を提供しています。
- インテル Arria 10 デバイス・データシート
  JTAG、POR、および電圧仕様に関する詳細情報を提供しています。
- インテル Cyclone 10 GX デバイス・データシート
  JTAG、POR、および電圧仕様に関する詳細情報を提供しています。
- Stratix V E, GS, and GX Device Family Pin Connection Guidelines
- Stratix V GT Device Family Pin Connection Guidelines
- Arria V GT, GX, ST and SX Device Family Pin Connection Guidelines
- Arria V GZ Device Family Pin Connection Guidelines
- Stratix IV GX and E Device Family Pin Connection Guidelines
- Stratix IV GT Device Family Pin Connection Guidelines
- Arria II Device Family Pin Connection Guidelines
- インテル Arria 10 GX、GT、および SX デバイスファミリー ピン接続ガイドライン

ソフトウェア要件

表 11. サポートされている Quartus のバージョン (インテル FPGA)

下記に示すサポートされている Quartus ソフトウェアのバージョンを使用し、FPGA のタイプに基づいて、デザイン・セキュリティー機能をイネーブルしてください。

デバイス	サポートされている Quartus のバージョン
40nm FPGA	Quartus II ソフトウェア v9.0 以降
28nm FPGA	Quartus II ソフトウェア v11.0 以降
20nm FPGA	インテル Quartus Prime 開発ソフトウェアのバージョン 15.1 以降(13)

注意: インテル Quartus Prime ライト・エディションでデザイン・セキュリティー機能をイネーブルするには、インテル FPGA 技術サポートからライセンスファイルを入手してください。

関連情報

インテル FPGA 技術サポート

安全なコンフィグレーション・フローの実装手順

安全なコンフィグレーション・フローを実装するには、次の手順を実行します。

1. **.ekp** ファイルを生成して、コンフィグレーション・データの暗号化を行います。

   コンフィグレーション・ソフトウェアでは、ユーザー定義の 256 ビットキーを使用してキー・プログラミング・ファイルおよび暗号化されたコンフィグレーション・ファイルを生成します。暗号化されたコンフィグレーション・ファイルは、フラッシュメモリーやコンフィグレーション・デバイスなどの外部メモリーに格納されます。詳しくは、ステップ 1: .ekp ファイルの生成およびコンフィグレーション・ファイルの暗号化 (14 ページ) を参照してください。

   注意: 20nm FPGA の場合、.rbf の暗号化は、拡張セキュリティー・オプションを持つスタンドアロンの Qcrypt ツールを使用して行うこともできます。

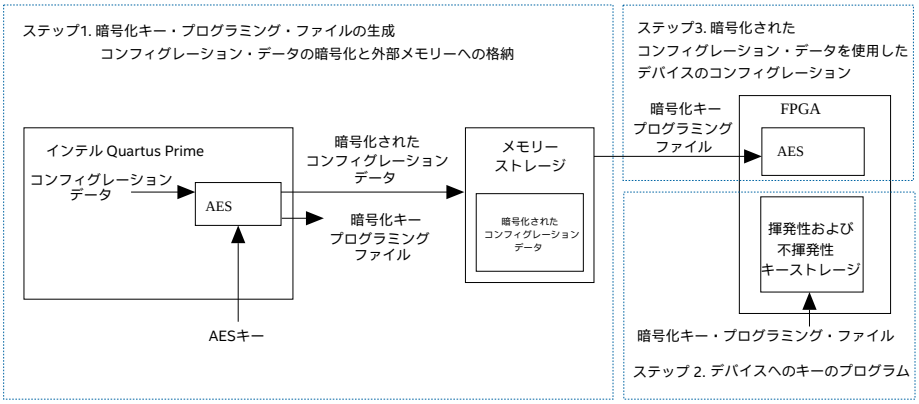
2. ユーザー定義の 256 ビットキーを FPGA にプログラムします。

   詳細については、ステップ 2a: 揮発性キーの FPGA へのプログラム (18 ページ) およびステップ 2b: 不揮発性キーの FPGA へのプログラム (19 ページ) を参照してください。

3. 40nm、28nm または 20nm FPGA デバイスをコンフィグレーションします。

   電源投入時、外部メモリーソースでは、暗号化されたコンフィグレーション・ファイルが FPGA に送信されます。デバイスでは、保存されたキーを使用してファイルを復号化し、デバイス自体をコンフィグレーションします。暗号化されたコンフィグレーション・データを使用した FPGA のコンフィグレーション方法については、暗号化されたコンフィグレーション・データを使用した 40nm、28nm または 20nm FPGA のコンフィグレーション (23 ページ) を参照してください。

図 -1: 安全なコンフィグレーション・フロー



(13) 20nm FPGA の場合、デザイン・セキュリティー機能をイネーブルするには、インテル Quartus Prime 開発ソフトウェアで用意されているスタンドアロンの Qcrypt ツールを使用することもできます。

ステップ 1: .ekp ファイルの生成およびコンフィグレーション・ファイルの暗号化

FPGA のデザイン・セキュリティー機能を使用するには、Qcrypt ツールで 20nm デザインを暗号化するか、または、インテル Quartus Prime 開発ソフトウェアで **.ekp** ファイルを生成し、コンフィグレーション・ファイルを暗号化してください。キーは、インテル Quartus Prime 生成のコンフィグレーション・ファイルには保存されません。実際の 256 ビットキーの生成は、ビットシーケンスから行われます。

デザイン・セキュリティー機能をイネーブルするには、インテル FPGA 技術サポートからライセンスファイルを入手してください。

.ekp ファイルの形式は、プログラミングに使用するハードウェアおよびシステムによって異なります。次の 3 つのファイル形式が、インテル Quartus Prime 開発ソフトウェアでサポートされています。

- JAM Byte Code (**.jbc**) ファイル
- JAM™ Standard Test and Programming Language (STAPL) Format (**.jam**) ファイル
- Serial Vector Format (**.svf**) ファイル

.ekp ファイルタイプのみが、インテル Quartus Prime 開発ソフトウェアから自動生成されます。**.jam** ファイルと **.svf** ファイルを インテル Quartus Prime を使用して作成しなければならないのは、これらのファイルがキー・プログラミングに必要な場合です。インテル Quartus Prime 開発ソフトウェアでは、JBC 形式の **.ekp** ファイルを同じプロジェクト・ディレクトリーで生成します。

注意: インテルでは、**.ekp** ファイルを社外秘とすることを推奨しています。

.ekp ファイルは、インテル FPGA イーサネット・ケーブル通信ケーブルまたは インテル FPGA ダウンロード・ケーブルおよび インテル Quartus Prime 開発ソフトウェアと併用してください。インテル FPGA イーサネット・ケーブル通信ケーブルでは、揮発性キーと不揮発性キー両方のプログラミングをサポートすることができます。インテル FPGA ダウンロード・ケーブルは、揮発性キーのプログラミングのみに使用されます。**.jam** ファイル形式は、一般に、サードパーティーのプログラミング・ベンダーおよび JTAG プログラマー・ベンダーで使用されています。**.svf** ファイル形式は、JTAG プログラマー・ベンダーで使用されています。

インテル Quartus Prime 開発ソフトウェアを使用したシングルデバイス .ekp ファイルの生成およびコンフィグレーション・ファイルの暗号化

シングルデバイス**.ekp** ファイルを生成し、コンフィグレーション・ファイルを暗号化するには、次の手順に従います。

1. デザイン・セキュリティー機能をイネーブルするには、インテル FPGA 技術サポートからライセンスファイルを入手してください。
2. インテル Quartus Prime 開発ソフトウェアを起動します。
3. Tools メニューの **License Setup** をクリックします。**Options** ダイアログボックスに **License Setup** オプションが表示されます。
4. **License file** フィールドに、ライセンスファイルの位置およびファイル名を入力するか、あるいはライセンスファイルを探して選択します。
5. **OK** をクリックします。
6. 次のいずれかのオプションを使用してデザインをコンパイルします。
   a. Processing メニューで、**Start Compilation** をクリックします。
   b. Processing メニューで、**Start** にカーソルを合わせ、**Start Assembler** をクリックします。

   暗号化されていない SRAM Object File (**.sof**) が生成されます。

7. File メニューの **Convert Programming Files** をクリックします。**Convert Programming Files** ダイアログボックスが表示されます。
   a. **Convert Programming Files** ダイアログボックスで、プログラミング・ファイル・タイプを **Programming file type** の一覧から選択します。
   b. 該当する場合、適切なコンフィグレーション・デバイスを **Configuration device** の一覧から選択します。
   c. **Mode** の一覧からモードを選択します。
   d. ファイル名を **File name** フィールドに入力するか、あるいはファイルを探して選択します。
   e. **Input files to convert** セクションの下の **SOF Data** をクリックします。
   f. **Add File** をクリックし、**Select Input File** ダイアログボックスを開きます。
   g. 暗号化されていない SOF ファイルを探して、**Open** をクリックします。
   h. **Input files to convert** セクションの下の SOF ファイル名をクリックします。フィールドが強調表示されます。
   i. **Properties** をクリックすると、**SOF Files Properties: Bitstream Encryption** ダイアログボックスが表示されます。
   j. **SOF Files Properties: Bitstream Encryption** ダイアログボックスで、**Generate encrypted bitstream** をオンにします。
   k. **Generate key programming file** をオンにし、**.ekp** のファイルパスとファイル名をテキストエリアに入力するか、あるいは `<filename>.ekp` を探して選択します。
   l. 20nm FPGA のみの追加手順: **Enable volatile security key check** ボックスをオンにし、`.sof` ファイルを揮発性セキュリティー・キーで暗号化するか、オフにして不揮発性セキュリティー・キーを使用します。
   m. 20nm FPGA のみの追加手順: **Generate encryption lock file** をオンにし、`.qlk` ファイルパスとファイル名を挿入するか、目的の `<filename>.qlk` を探します。
   n. **.key** ファイルまたは **Add** ボタンを使用して、プルダウンリストにキーを追加します。**Add** ボタンと **Edit** ボタンをクリックすると、**Key Entry** ダイアログボックスが表示されます。**Delete** ボタンをクリックすると、現在選択されているキーがプルダウンリストから削除されます。

      注意: 40nm FPGA では、2 つの 256 ビットキーを入力する必要があります。暗号化は、2 つの 256 ビットキーの組み合わせから派生しています。28nm および 20nm FPGA では、単一の 256 ビットキーを入力する必要があります。最終的な暗号化キーは、一方向関数を使用して派生します。

      .key ファイルオプションを使用すると、1 つまたは 2 つのキーファイルを、対応するドロップダウンボックスで指定できます。異なるファイルを **Key 1** フィールドと **Key 2** フィールドに使用することも、1 つの**.key** ファイルを両方のフィールドに使用することもできます。

      .key ファイルはプレーン・テキスト・ファイルで、その各行はキーを表しています。ただし、「#」から始まる行の場合は除きます。「#」記号は、コメントを示すために使用されます。それぞれの有効キーの行には、次のフォーマットが含まれています。*<key identity><white space><256-bit hexadecimal key>*

```
# This is an example key file
key1 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
key2 ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789
```

キー ID は英数字からなる名前で、キーの識別のために使用されます (キーファイルのエントリーに類似しています) 。またこのキーは、**Show entered keys** ボタンがオフになっているときにテキスト表示されます。このキーが完全なキーと一緒に表示されるのは、**Show entered keys** がオンになっているときです。

プルダウンリストのキーは、**.key** ファイルに保存できます。対応する **Save** ボタンをクリックしてキーを保存し、標準の **File** ダイアログボックスを表示してください。プルダウンリスト内のすべてのキーは、選択または作成された **.key** ファイルに保存されます。

Key Entry Method を選択して、オンスクリーン・キーパッドまたはキーボードで暗号化キーを入力してください。

オンスクリーン・キーパッドを使用すると、キーパッドを使用したキーの入力ができます。キーを選択し、オンスクリーン・キーパッドをクリックして値を入力します 。オプションで、キーを入力時に表示することもできます。このオプションを使用すると、キーを確認する必要がありません。

オンスクリーン・キーパッドの使用中は、キーボードでキーを入力しようとすると、ポップアップが表示され、入力したキーは無視されます。あるいは、キーボードから暗号化キーを入力することもできます。

i. デザイン・セキュリティー機能の免責事項をお読みください。デザイン・セキュリティー機能の免責事項に同意・承認する場合は、**acknowledgment** ボックスをオンにしてください。

ii. **OK** をクリックします。

o. 20nm FPGA のみの追加手順: **Security Options** の下の、**Disable external partial reconfiguration** の一覧からレベルを選択します。

p. 20nm FPGA のみの追加手順: **Security Options** の下の、**Disable key-related JTAG instructions** の一覧からレベルを選択します。

q. 20nm FPGA のみの追加手順: **Security Options** の下の、**Disable other extended JTAG instructions** の一覧からレベルを選択します。

8. **Convert Programming Files** ダイアログボックスで、**OK** をクリックします。<*filename*>**.ekp** が、暗号化されたコンフィグレーション・ファイルと同じプロジェクト・ディレクトリーに生成されます。
9. Tools メニューで **Programmer** をクリックし、**Programmer** ダイアログボックスを開きます。
10. **Mode** の一覧で、プログラミング・モードに **JTAG** を選択します。
11. **Hardware Setup** をクリックし、**Hardware Setup** ダイアログボックスを開きます。
    a. 現在選択されているハードウェアの一覧から、**Intel FPGA Ethernet Cable** をプログラミング・ハードウェアとして選択します。
    b. **Done** をクリックします。
12. **Add File** をクリックし、**Select Programmer File** ダイアログボックスを開きます。
    a. <*filename*>**.ekp** をタイプして **File name** フィールドに入力します。
    b. **Open** をクリックします。
13. 追加した**.ekp** ファイルを強調表示して、**Program/Configure** をクリックします。
14. File メニューで、**Create/Update** にカーソルを合わせ、**Create JAM, SVF, or ISC File** をクリックします。**Create JAM, SVF, or ISC File** ダイアログボックスが表示されます。
15. **File format** フィールドの**.ekp** ファイルに対して、必要なファイル形式 (JEDEC STAPL Format [**.jam**]) を選択します。
16. ファイル名を **File name** フィールドに入力するか、もしくはファイルを探して選択します。

17. **OK** をクリックして**.jam** ファイルを生成します。
18. Tools メニューで **Programmer Options** をクリックし、**Programmer Options** ダイアログボックスを開きます。

    注意: 不揮発性の安全なデザイン機能の場合は、**Configure volatile design security key** をオフにして、**.ekp** ファイルの不揮発性**.svf** ファイルを生成してください。
19. OK をクリックします。
20. 15 (16 ページ) から 17 (17 ページ) を繰り返して、**.ekp** ファイルの**.svf** ファイルを生成します。**Create JAM, SVF, or ISC File** ダイアログボックスのデフォルト設定を使用し、**.ekp ファイル**の**.svf** ファイルを生成します。

インテル Quartus Prime 開発ソフトウェアのコマンドライン・インターフェイスを使用したシングルデバイス .ekp ファイルの生成およびコンフィグレーション・ファイルの暗号化

あるコマンドライン・インターフェイスを使用すると、シングルデバイスの**.ekp** を生成し、Raw Binary File (**.rbf**) を暗号化することができます。このコマンドライン・インターフェイスでは、インテル Quartus Prime 開発ソフトウェアのコマンドライン実行可能ファイルである `quartus_cpf` を使用し、次の構文またはオプションが必要です。

- --key/-k <*path to key file*>:<*key identity*>
- **.sof** ファイル (ユーザーによるデザイン)
- **.ekp** ファイル (必要な暗号化キー・プログラミング・ファイル名)

圧縮および非圧縮の**.rbf** をコンフィグレーション用に作成するために、次のコマンドを使用します。コマンドのオプションファイルには、文字列圧縮 =on を含めます。

```
quartus_cpf -c --option=<option file> --key
<keyfile>:<keyid1>:<keyid2> <input_sof_file> <output_rbf_file>
```

注意: 暗号化と圧縮を 20nm FPGA で同時に使用することはできません。

オプションファイルについて詳しくは、インテル Quartus Prime 開発ソフトウェアのコマンドライン・ヘルプを参照してください。使用可能なオプションについて詳しくは、`quartus_cpf --help=option` を実行してください。20nm FPGA の場合は、Qcrypt ツールのコマンドラインを使用して、`.rbf` ファイルを暗号化または復号化します。`.ekp` または `.rbf` 以外の暗号化された設定ファイルを生成するには、quartus_cpf を実行する必要があります。

例-1:

次の例では、キーのセット 2 つが、2 つの異なるキーファイルに格納されています。key1 は **key1.key** に、key2 は **key2.key** に格納されています。

```
quartus_cpf --key D: \SIV_DS\key1.key:key1 --key

D:\SIV_DS\key2.key:key2 D:\SIV_DS\test.sof D:\SIV_DS\test.ekp
```

例-2:

次の例では、キーのセット 2 つが同じキーファイルに格納されています。key1 と key2 が **key12.key** に格納されています。

```
quartus_cpf --key
```

```
D:\SIV_DS\key12.key:key1:key2 D:\SIV_DS\test.sof D:\SIV_DS
\test.ekp
```

インテル Quartus Prime 開発ソフトウェアを使用したマルチデバイス .ekp ファイルの生成およびコンフィグレーション・ファイルの暗号化

マルチデバイス **.ekp** ファイルを生成し、コンフィグレーション・ファイルを暗号化するには、次の手順に従います。

1. インテル Quartus Prime 開発ソフトウェアを起動します。
2. インテル Quartus Prime 開発ソフトウェアを使用したシングルデバイス .ekp ファイルの生成およびコンフィグレーション・ファイルの暗号化 (14 ページ) のステップ 9 (16 ページ) からステップ 11 (16 ページ) を繰り返します。
3. **Add File** をクリックし、**Select Programmer File** ダイアログボックスを開きます。
   a. シングルデバイス**.ekp** ファイルを選択し、<*single_ekp*>**.ekp** を **File name** フィールドに入力してください。
   b. **Open** をクリックします。

   注意: 同じ JTAG チェーン内のデバイスの正しいシーケンスには、インテル Quartus Prime プログラマーの **Auto-Detect** オプションを使用することができます。FPGA のいずれかで key-programmed が必要ない場合は、デバイスを インテル Quartus Prime プログラマーの <*single_ekp*>**.ekp** ファイルに置き換える必要はありません。
4. 3 (18 ページ) を同じチェーン内の各デバイスについて繰り返します。適切なデバイスシーケンスを使用して、**.ekp** ファイルがプログラマー・ウィンドウに追加されていることを確認します。
5. 追加したすべての**.ekp** ファイルを強調表示し、**Program/Configure** をクリックします。
6. File メニューで **Create/Update** にカーソルを合わせ、**Create JAM, SVF, or ISC File** をクリックします。**Create JAM, SVF, or ISC File** ダイアログボックスが表示されます。
7. 必要なファイル形式 (**.jam**) の選択を、すべての**.ekp** ファイルに対して、**File format** フィールドで行います。
8. ファイル名を **File name** フィールドに入力するか、あるいはファイルを探して選択します。
9. **OK** をクリックして、**.jam** ファイルを生成します。
10. Tools メニューで **Programmer Options** をクリックし、**Programmer Options** ダイアログボックスを開きます。

    注意: **Configure volatile design security key** をオフにして、**.ekp** ファイルの non-volatile **.svf** ファイルを生成してください。
11. **OK** をクリックします。
12. 7 (18 ページ) から 9 (18 ページ) を繰り返して、**.svf** ファイルをすべての**.ekp** ファイルに対して生成します。**Create JAM, SVF, or ISC File** ダイアログボックスのデフォルト設定を使用し、**.ekp** ファイルの**.svf** ファイルを生成します。

ステップ 2a: 揮発性キーの FPGA へのプログラム

揮発性キーを FPGA にプログラムする前に、FPGA の正常なコンフィグレーションが、暗号化されていないコンフィグレーション・ファイルを使用して確実に行われるようにしてください。揮発性キーは、再プログラム可能かつ消去可能なキーです。揮発性キーを使用して FPGA をプログラムする前に、揮発性キー保持のための外部バッテリーの用意が必要です。FPGA では、揮発性キーが正常にプログラムされる

と、暗号化および非暗号化の両方のコンフィグレーション・ビットストリームを受け入れることができます。これにより、暗号化されていないコンフィグレーション・ビットストリームをボードレベルのテストで使用できるようになります。

FPGA をコンフィグレーションする場合、その FPGA に含まれる揮発性キーのコンフィグレーション・ファイルが誤ったキーで暗号化されていると、コンフィグレーションは正常に行われません。この現象が発生すると、FPGA からの nSTATUS 信号は low にパルスされ、自動的にリセットされます。これは、**Auto-restart configuration after error** オプションを インテル Quartus Prime 開発ソフトウェアでイネーブルした場合です。

キーを FPGA にプログラムするには、オンボードのプロトタイプ作成を使用します。 これについては、キー・プログラミング (8 ページ) の表に記載されています。

ステップ 2b: 不揮発性キーの FPGA へのプログラム

不揮発性キーをデバイスにプログラムする前に、FPGA の正常なコンフィグレーションが、暗号化されていないコンフィグレーション・ファイルを使用して確実に行われるようにしてください。不揮発性キーでは、JTAG インターフェイスを介したワンタイムプログラムが可能です。不揮発性キーのデバイスへのプログラムは、外部バッテリーなしで行うことができます。デバイスでは、不揮発性キーが正常にプログラムされると、暗号化および非暗号化の両方のコンフィグレーション・ビットストリームを受け入れることができます。改ざん防止ビットを設定した場合は、暗号化されたコンフィグレーション・ビットストリームだけを受け入れます。これにより、暗号化されていないコンフィグレーション・ビットストリームが、ボードレベルのテストで使用できるようになります。

FPGA のコンフィグレーションの試行の際に、誤ったキーで暗号化されたコンフィグレーション・ファイルを持つ揮発性キーが含まれていると、正常なコンフィグレーションが行われません。この現象が発生すると、FPGA からの nSTATUS 信号は low にパルスされ、自動的にリセットされます。これは、**Auto-restart configuration after error** オプションを インテル Quartus Prime 開発ソフトウェアでイネーブルした場合です。

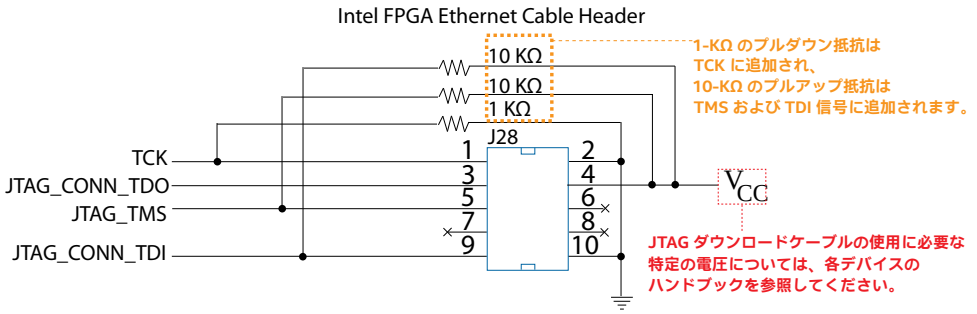
不揮発性キーを FPGA へプログラムするには、オンボードのプロトタイプ作成と量産、およびオフボードのプロトタイプ作成と量産を使用します。これについては、キー・プログラミング (8 ページ) の表に記載されています。

および インテル Quartus Prime 開発ソフトウェアを使用した揮発性および不揮発性キーのプログラミング

インテル FPGA イーサネット・ケーブル通信ケーブルを インテル FPGA イーサネット・ケーブル・ヘッダー (次の図を参照) に接続します。

図 -2: インテル FPGA イーサネット・ケーブル・ヘッダー

インテル FPGA イーサネット・ケーブル・ヘッダーと インテル FPGA ダウンロード・ケーブル・ヘッダーは、同一のものとしてキー・プログラミングに使用できます。



注意: インテル FPGA イーサネット・ケーブルの場合は、TCK 速度を必要な TCK 期間に設定します。

関連情報

- EthernetBlaster Communications Cable User Guide
- EthernetBlaster II Communications Cable User Guide
  インテル FPGA イーサネット・ケーブル の TCK クロック速度変更に関する詳細情報を提供しています。
- Device Datasheet for Arria II Devices
  JTAG ダウンロード・ケーブルの使用に必要な特定の電圧に関する詳細情報を提供しています。
- DC and Switching Characteristics for Stratix IV Devices
  JTAG ダウンロード・ケーブルの使用に必要な特定の電圧に関する詳細情報を提供しています。
- Arria V Device Datasheet
  JTAG ダウンロード・ケーブルの使用に必要な特定の電圧に関する詳細情報を提供しています。
- Cyclone V Device Datasheet
  JTAG ダウンロード・ケーブルの使用に必要な特定の電圧に関する詳細情報を提供しています。
- Stratix V Device Datasheet
  JTAG ダウンロード・ケーブルの使用に必要な特定の電圧に関する詳細情報を提供しています。
- インテル Arria 10 デバイス・データシート
  JTAG ダウンロード・ケーブルの使用に必要な特定の電圧に関する詳細情報を提供しています。
- インテル Cyclone 10 GX デバイス・データシート
  JTAG ダウンロード・ケーブルの使用に必要な特定の電圧に関する詳細情報を提供しています。

インテル Quartus Prime 開発ソフトウェアを使用したシングルデバイスの揮発性または不揮発性キーのプログラミング

シングルデバイスの揮発性および不揮発性キーのプログラミングを、インテル Quartus Prime 開発ソフトウェアを使用し、インテル FPGA イーサネット・ケーブルを介して実行するには、次の手順を実行します。

1. インテル FPGA イーサネット・ケーブルのファームウェアのバージョンを確認してください。JTAG ファームウェアのビルドが最新であることを確認してください。

   注意: Cable and Adapter driver のページを参照して、最新バージョンの インテル FPGA イーサネット・ケーブルのファームウェアを入手してください。
2. インテル Quartus Prime 開発ソフトウェアを起動します。
3. Tools メニューで **Programmer** をクリックし、**Programmer** ダイアログボックスを開きます。
4. **Mode** の一覧で、**JTAG** をプログラミング・モードとして選択します。
5. **Hardware Setup** をクリックし、**Hardware Setup** ダイアログボックスを開きます。
   a. **Currently selected hardware** 一覧で、**Intel FPGA Ethernet Cable** をプログラミング・ハードウェアとして選択します。
   b. **Done** をクリックします。
6. **Add File** をクリックし、**Select Programmer File** ダイアログボックスを開きます。
   a. *filename*>**.ekp** をタイプして、**File name** フィールドに入力します。
   b. **Open** をクリックします。
7. 追加した**.ekp** ファイルを強調表示して、**Program/Configure** をクリックします。

8. Tools メニューで **Options** をクリックし、**Options** ダイアログボックスを開きます。
9. **Category** 一覧で、**Programmer** をクリックします。**Configure volatile design security key** オプションをオンまたはオフにして、揮発性または不揮発性キーのプログラミングを実行します。
10. **OK** をクリックして **Options** ダイアログボックスを閉じます。
11. **Start** をクリックしてキーをプログラムします。

注意: インテル Quartus Prime 開発ソフトウェアのメッセージウィンドウには、キー・プログラミング動作が正常に行われたかどうかの情報が表示されます。

関連情報

EthernetBlaster Communications Cable User Guide
JTAG ファームウェアに関する詳細情報を提供しています。

インテル Quartus Prime 開発ソフトウェアのコマンドライン・インターフェイスを使用したシングルデバイスの揮発性または不揮発性キーのプログラミング

シングルデバイスの揮発性または不揮発性キーのプログラミングを、インテル Quartus Prime 開発ソフトウェアのコマンドライン・インターフェイスを使用し、インテル FPGA イーサネット・ケーブルを介して実行するには、次の手順を実行します。

1. インテル Quartus Prime 開発ソフトウェアを使用したシングルデバイスの揮発性または不揮発性キーのプログラミング (20 ページ) のステップ 1 (20 ページ) を実行します。
2. JTAG サーバーに接続された インテル FPGA イーサネット・ケーブルのポート番号を決定するには、command-line プロンプトで `quartus_jli -n` を入力します。
3. ステップ 1: .ekp ファイルの生成およびコンフィグレーション・ファイルの暗号化 (14 ページ) で生成された **single_ekp.jam** ファイルを使用して、揮発性または不揮発性キーのプログラミングを 1 つの FPGA に対して実行します。次のコマンドラインを使用します。
   - 揮発性キーのプログラミング:

     ```
     quartus_jli -c<n> single_ekp.jam -aKEY_CONFIGURE
     ```
   - 不揮発性キーのプログラミング:

     ```
     quartus_jli -c<n> single_ekp.jam -aKEY_PROGRAM
     ```

     <n> は、`-n` オプションによって返されたポート番号です。

   注意: インテル Quartus Prime 開発ソフトウェアのコマンドラインには、キー・プログラミング動作が正常に行われたかどうかの情報が表示されます。

関連情報

AN 425: Using the Command-Line Jam STAPL Solution for Device Programming
`quartus_jli` に関する詳細情報を提供しています。

インテル Quartus Prime 開発ソフトウェアを使用したマルチデバイスの揮発性または不揮発性キーのプログラミング

マルチデバイスの揮発性または不揮発性キーのプログラミングを インテル Quartus Prime 開発ソフトウェアを使用し、インテル FPGA イーサネット・ケーブルを介して実行するには、次の手順を実行します。

1. インテル Quartus Prime 開発ソフトウェアを使用したシングルデバイスの揮発性または不揮発性キーのプログラミング (20 ページ) のステップ 1 (20 ページ) からステップ 5 (20 ページ) を繰り返します。
2. **Add File** をクリックし、**Select Programmer File** ダイアログボックスを開きます。
   a. シングルデバイス**.ekp** ファイルを使用したプログラミング
      i. <*single_device*>**.ekp** をタイプして、**File name** フィールドに入力します。
      ii. **Open** をクリックします。
      iii. ステップ 2.a.i (22 ページ) からステップ 2.a.ii (22 ページ) を同じチェーン内のデバイスの数だけ繰り返します。
      iv. 追加した**.ekp** ファイルを強調表示し、**Program/Configure** をクリックします。

      注意: 同じ JTAG チェーン内のデバイスの正しいシーケンスについては、インテル Quartus Prime プログラマーの **Auto-Detect** オプションを使用します。
   b. マルチデバイス **.jam** ファイルを使用したプログラミング
      i. <*multi_device*>**.jam** をタイプして **File name** フィールドに入力します。
      ii. **Open** をクリックします。
      iii. 追加した**.jam** ファイルを強調表示して、**Program/Configure** をクリックします。
3. インテル Quartus Prime 開発ソフトウェアを使用したシングルデバイスの揮発性または不揮発性キーのプログラミング (20 ページ) のステップ 8 (21 ページ) からステップ 10 (21 ページ)を繰り返して、揮発性または不揮発性キーのプログラミングを実行します。
4. **Start** をクリックしてキーをプログラムします。

   注意: インテル Quartus Prime 開発ソフトウェアのメッセージウィンドウには、キー・プログラミング動作が正常に行われたかどうかの情報が表示されます。

インテル Quartus Prime 開発ソフトウェアのコマンドライン・インターフェイスを使用したマルチデバイスの揮発性または不揮発性キーのプログラミング

マルチデバイスの揮発性または不揮発性キーのプログラミングを、インテル Quartus Prime 開発ソフトウェアのコマンドライン・インターフェイスを使用し、インテル FPGA イーサネット・ケーブルを介して実行するには、次の手順を実行します。

1. インテル Quartus Prime 開発ソフトウェアを使用したシングルデバイスの揮発性または不揮発性キーのプログラミング (20 ページ) のステップ 1 (20 ページ) を実行します。
2. JTAG サーバーに接続された インテル FPGA イーサネット・ケーブルのケーブルポートの番号を決定するには、command-line プロンプトで `quartus_jli -n` を入力します。
3. ステップ 1: .ekp ファイルの生成およびコンフィグレーション・ファイルの暗号化 (14 ページ) で生成された **multi_ekp.jam** ファイルを使用して、揮発性または不揮発性キーのプログラミングを複数の FPGA に対して実行します。次のコマンドラインを使用します。
   - 揮発性キーのプログラミング:

     ```
     quartus_jli -c<n> multi_ekp.jam -aKEY_CONFIGURE
     ```
   - 不揮発性キーのプログラミング

     ```
     quartus_jli -c<n> multi_ekp.jam -aKEY_PROGRAM
     ```

     <n> は、-n オプションによって返されたポート番号です。

注意: インテル Quartus Prime 開発ソフトウェアのコマンドラインには、キー・プログラミング動作が正常に行われたかどうかの情報が表示されます。

JTAG Technologies を使用したキー・プログラミング

デザインに対するキー・プログラミングの実行には、**.svf** ファイル (**.svf** フォーマットの**.ekp** ファイル) および JT 37xx バウンダリー・スキャン・コントローラーを JT2147 QuadPod システムと組み合わせて使用します。

マルチデバイスのプログラミングをサポートするための .svf ファイルの作成に関する情報については、インテル Quartus Prime 開発ソフトウェアを使用したマルチデバイス .ekp ファイルの生成およびコンフィグレーション・ファイルの暗号化 (18 ページ) で説明しています。

関連情報

JTAG Technologies
JTAG プログラミングの手順に関する詳細情報を提供しています。

暗号化されたコンフィグレーション・データを使用した 40nm、28nm または 20nm FPGA のコンフィグレーション

最後のステップとして、保護された 40nm、28nm または 20nm FPGA のコンフィグレーションを、暗号化されたコンフィグレーション・ファイルを使用して行います。

コンフィグレーション中、暗号化されたコンフィグレーション・データは、40nm、28nm、または 20nm の FPGA に送信されます。FPGA では、以前に保存したキーを使用してコンフィグレーション・データを復号化し、暗号化されていないデータを使用して FPGA 自体のコンフィグレーションを行います。コンフィグレーション・ファイルは、正しいキーを使用して暗号化された場合にのみ、FPGA によって受け入れられ、正常なコンフィグレーションが行われます。正しいキーがなければ、盗まれた暗号化ファイルは無用です。

改ざん防止ビット・プログラミングのイネーブルの手順

*安全なコンフィグレーション・フローの実装手順*で生成されたデフォルトの `.ekp` ファイルに含まれるのは、揮発性キーまたは不揮発性キーのプログラミングのみです。改ざん防止ビット・プログラミングをイネーブルするには、次の手順に従います。

1. テキストエディターを使用して、`quartus.ini` ファイルを作成します。次のキーと値のペアを使用します。`PGM_GEN_KEY_SECURE_EKP=ON`
2. `quartus.ini` を次のいずれかのフォルダーに保存します。
   - プロジェクト・フォルダー
   - <*Quartus installation folder*>\bin64 フォルダー (Windows OS 用)
   - <*Quartus installation folder*>/linux64 フォルダー (Linux OS 用)
3. Intel Quartus Prime Convert Programming File ツールによる `quartus.ini` の読み出しが、`.ekp` ファイルの生成プロセス中に行われた場合、追加の改ざん防止ビット・プログラミング命令が、生成された `.ekp` ファイルに挿入されます。

   注意: この `quartus.ini` で生成された `.ekp` ファイルには、改ざん保護ビット・プログラミングが含まれています。`.ekp` ファイルを使用してデバイスにプログラムすると、改ざん防止ビットがプログラムされます。このプログラムは、元に戻すことはできません。`.ekp` ファイルを管理して、改ざん防止ビットがデバイスに意図せずにプログラミングされるのを回避してください。

.ekp ファイルには改ざんビット・プログラミング命令が含まれています。したがって、この .ekp ファイルから .jam ファイルまたは .svf ファイルをキー・プログラミング用に生成した場合、.jam ファイルまたは .svf ファイルによる改ざん保護ビットのプログラミングには、キーと値の指定ペアを持つ quartus.ini は不要です。

サポートされているコンフィグレーション・スキーム

デザイン・セキュリティー機能は、JTAG ベースのコンフィグレーションを除くすべてのコンフィグレーション・スキームで使用できます。

表 12. 各コンフィグレーション手法に対するデザイン・セキュリティーのサポート

コンフィグレーション・スキーム	コンフィグレーション方法	デザイン・セキュリティー	注記
FPP	MAX II または MAX V デバイス/マイクロプロセッサーおよびフラッシュメモリー	あり	このモードでは、ホスト・システムで 4 倍のデータレートの DCLK 信号を送信する必要があります。
AS	シリアル・コンフィグレーション・デバイス	あり	—
PS	MAX II または MAX V デバイス/マイクロプロセッサーおよびフラッシュメモリー	あり	—
	インテル FPGA ダウンロード・ケーブルおよび インテル FPGA ダウンロード・ケーブル II	あり	暗号化された .rbf の FPGA へのコンフィグレーションを、インテル Quartus Prime Programmer の PS モードを使用して実行してください。
JTAG	インテル FPGA ダウンロード・ケーブルおよび インテル FPGA ダウンロード・ケーブル II	—	キー・プログラミング用

システムに Common Flash Interface (CFI) フラッシュメモリーが含まれている場合は、それを FPGA コンフィグレーションに対しても使用することができます。MAX II および MAX V を Parallel Flash Loader Intel FPGA IP コアと併用することで、効率的な CFI フラッシュメモリーのプログラムを JTAG インターフェイスを介して行うことができます。

デザイン・セキュリティー機能は、圧縮やリモート・システム・アップグレード機能などの他のコンフィグレーション機能と併用できます。デザイン・セキュリティー機能で圧縮を使用している場合、コンフィグレーション・ファイルは、インテル Quartus Prime 開発ソフトウェアで圧縮されてから暗号化されます。コンフィグレーション中、FPGA では、コンフィグレーション・ファイルを復号化してから解凍します。

注意: 暗号化と圧縮を 20nm FPGA で同時に使用することはできません。

バウンダリー・スキャン・テスト (BST) または Signal Tap ロジック・アナライザーを使用して、FPGA 内の機能データを解析してください。ただし、JTAG コンフィグレーションの実行は、改ざん防止ビットが設定されたキーを 40nm、28nm、または 20nm FPGA にプログラムした後ではできません。

Signal Tap ロジック・アナライザーを使用する場合は、まず、暗号化されたコンフィグレーション・ファイルを持つデバイスのコンフィグレーションを行ってください。このとき使用するのは、PS、FPP、または AS コンフィグレーション手法です。デザインには、Signal Tap ロジック・アナライザーのインスタンスが、少なくとも 1 つ含まれている必要があります。FPGA のコンフィグレーションをデザイン内の Signal Tap ロジック・アナライザーのインスタンスを使用して行った後、インテル Quartus Prime 開発ソフトウェアで Signal Tap ロジック・アナライザー・ウィンドウを開き、**Scan Chain** をクリックします。スキャンが完了すると、Signal Tap ロジック・アナライザーは、JTAG インターフェイスを使用したデータ収集の準備が整った状態になります。

関連情報

- Configuration, Design Security, and Remote System Upgrades in Arria II Devices
  Arria II デバイスのデザイン・セキュリティーに関する詳細を提供しています。
- Configuration, Design Security, and Remote System Upgrades in Stratix IV Devices
  Stratix IV デバイスのデザイン・セキュリティーに関する詳細を提供しています。
- Configuration, Design Security, and Remote System Upgrades in Arria V Devices
  Arria V デバイスのデザイン・セキュリティーに関する詳細を提供しています。
- Configuration, Design Security, and Remote System Upgrades in Cyclone V Devices
  Cyclone V デバイスのデザイン・セキュリティーに関する詳細を提供しています。
- Configuration, Design Security, and Remote System Upgrades in Stratix V Devices
  Stratix V デバイスのデザイン・セキュリティーに関する詳細を提供しています。
- インテル Arria 10 デバイスにおけるコンフィグレーション、デザイン・セキュリティー、およびリモート・システム・アップグレード
  インテル Arria 10 デバイスのデザイン・セキュリティーに関する詳細を提供しています。
- Configuration, Design Security, and Remote System Upgrades in Intel Cyclone 10 GX Devices
  インテル Cyclone 10 デバイスのデザイン・セキュリティーに関する詳細を提供しています。

セキュリティー・モードの検証

インテル FPGA では、KEY_VERIFY JTAG 命令をサポートしています。これによってデバイスの既存のセキュリティー・モードを検証できます。揮発性キーのプログラムが成功したかどうかを確認するには、.jam ファイルを使用してセキュリティー・モードの確認手順を自動化します。

表 13. KEY_VERIFY JTAG 命令

JTAG 命令	命令コード	説明
KEY_VERIFY	00 0001 0011	TDI と TDO 間のキー検証スキャンレジスターを接続します。

KEY_VERIFY JTAG 命令によって、チップ上でイネーブルしたセキュリティー機能に関する情報を読み出すことができます。

表 14. 40nm FPGA のセキュリティー・モードの検証

セキュリティー・モード	サポートされるデバイス	ビット 0	ビット 1	ビット 2	ビット 3	ビット 4	ビット 5
No key	Arria II GX	0	0	0	0	0	0
	• Arria II GZ • Stratix IV	0	0	0	0	X	X
Volatile key	Arria II GX	1	0	0	0	0	0
	• Arria II GZ • Stratix IV	1	0	0	0	X	X
Volatile key with tamper protection	Arria II GX	1	0	0	0	1	0
	• Arria II GZ • Stratix IV	X	X	X	X	X	X
Non-volatile key	Arria II GX	0	1	0	1	0	0

continued...

セキュリティー・モード	サポートされるデバイス	ビット 0	ビット 1	ビット 2	ビット 3	ビット 4	ビット 5
	• Arria II GZ • Stratix IV	0	1	0	1	X	X
Non-volatile key with tamper protection bit	Arria II GX	0	1	1	1	0	0
	• Arria II GZ • Stratix IV	0	1	1	1	X	X

表 15. 28nm FPGA のセキュリティー・モードの検証

セキュリティー・モード	ビット 0	ビット 1	ビット 2	ビット 3	ビット 4	ビット 5	ビット 6	ビット 7	ビット 8
No key	0	0	0	0	0	X	X	X	X
Volatile key	1	0	0	0	0	X	X	X	X
Volatile key with tamper protection (14)	1	0	0	0	1	X	X	X	X
Non-volatile key	0	1	0	1	0	X	X	X	X
Non-volatile key with tamper protection bit (14)	0	1	1	1	0	X	X	X	X

表 16. 20nm FPGA のセキュリティー・モードの検証

ビット	セキュリティー機能または設定	説明	アクティブ値
0	Volatile Key	このビットが設定されるのは、揮発性キーがデバイスに正常にプログラムされたときです。	1
1	Attempt Non-volatile Key Programming	このビットが設定されると、何者かにより不揮発性キーの OTP ヒューズへの書き込みが試行されたことを示します。	1
2	Disable Non-volatile Key	このビットが設定されると、揮発性キーの使用をディスエーブルします。	1
3	Non-volatile Key	このビットが設定されると、何者かにより不揮発性キーの OTP ヒューズの書き込みが行われたことを示します。	1
4	Tamper Protection	このビットが設定されると、FPGA は不揮発性または揮発性キーのいずれかを使用した改ざん保護モードになります。	1
5	Don't Care	ドントケア	X
6	Volatile Key Lock	このビットが設定されると、揮発性キーの外部 JTAG からの再プログラムが防止されます。	1
7 - 10	Don't Care	ドントケア	X
11(15)	Force Configuration from HPS only	このビットが設定されるのは、コンフィグレーションが HPS からのみ可能な場合です。	1
12	External JTAG Bypass	このビットが設定されると、外部 JTAG がディスエーブルであることを示します。	1
13(16)	HPS JTAG Bypass	このビットが設定されると、外部 JTAG がディスエーブルであることを示します。	1

continued...

(14) 改ざん防止機能がイネーブルの場合、デバイスは電源投入後に JTAG セキュアモードになります。JTAG セキュアモードをディスエーブルするには、UNLOCK を発行する必要があります。

(15) ビット 11 はインテル Cyclone 10 GX デバイスには適用されません。インテル Cyclone 10 GX デバイスでは、このビットは「ドントケア」状態にあります。

ビット	セキュリティー機能または設定	説明	アクティブ値
14[17]	Disable Partial Reconfiguration and Scrubbing	このビットが設定されると、外部 PR および外部スクラビング (HPS PR および HPS スクラビングを含む) がディスエーブルであることを示します。	1
15	Disable Volatile Key	このビットが設定されると、揮発性キーがディスエーブルであることを示します。	1
16	Don't Care	ドントケア	X
17	Disable Key Related JTAG Instructions	このビットが設定されると、キーに関連するすべての JTAG 命令への外部 JTAG アクセスがディスエーブルであることを示します。	1
18	JTAG Secure Mode	このビットが設定されると、必須の JTAG 命令のみが外部からのアクセスを許可されていることを示します。	1
19	Don't Care	ドントケア	X
20	Volatile Key Clear	このビットが設定されるのは、揮発性キーがデバイスから正常にクリアされたときです。	1

次の例で示すのは、FPGA のセキュリティー・モードを検証するための .jam ファイルです。サンプルの .jam ファイルが適用できるのは、JTAG チェーン内の単一の FPGA デバイスのみです。SoC デバイスの場合は、次のステートメントを IRSCAN コマンドの前に追加してください。

```
PREIR 4;
PREDR 1;
```

例-3: 40nm FPGA の JAM ファイル (Arria II GX デバイス)

```
STATE RESET;

STATE IDLE;

'Security Mode Identification

BOOLEAN verify_reg[6];

IRSCAN 10, $013;

WAIT 100 USEC;

DRSCAN 6, $0, CAPTURE verify_reg[5..0];
```

例-4: 40nm FPGA の JAM ファイル (Arria II GZ および Stratix IV デバイス)

```
STATE RESET;

STATE IDLE;

‘Key Verification

BOOLEAN verify_reg[4];
```

(16) ビット 13 はインテル Cyclone 10 GX デバイスには適用されません。インテル Cyclone 10 GX デバイスでは、このビットは「ドントケア」状態にあります。

(17) ビット 14 はインテル Cyclone 10 GX デバイスには適用されません。インテル Cyclone 10 GX デバイスでは、このビットは「ドントケア」状態にあります。

```
IRSCAN 10, $013;

WAIT 100 USEC;

DRSCAN 4, $0, CAPTURE verify_reg[3..0];
```

例-5: 28nm FPGA の JAM ファイル

```
STATE RESET;

STATE IDLE;

'Key Verification in JAM format

BOOLEAN verify_reg[9];

IRSCAN 10, $013;

WAIT 100 USEC;

DRSCAN 9, $0, CAPTURE verify_reg[8..0];
```

例-6: 20nm FPGA の JAM ファイル

```
STATE RESET;

STATE IDLE;

'Key Verification in JAM format

BOOLEAN verify_reg[21];

IRSCAN 10, $013;

WAIT 100 USEC;

DRSCAN 21, $0, CAPTURE verify_reg[20..0];
```

関連情報

- Configuration, Design Security, and Remote System Upgrades in Arria II Devices
  Arria II デバイスのデザイン・セキュリティーに関する詳細を提供しています。
- Configuration, Design Security, and Remote System Upgrades in Stratix IV Devices
  Stratix IV デバイスのデザイン・セキュリティーに関する詳細を提供しています。
- Configuration, Design Security, and Remote System Upgrades in Arria V Devices
  Arria V デバイスのデザイン・セキュリティーに関する詳細を提供しています。
- Configuration, Design Security, and Remote System Upgrades in Cyclone V Devices
  Cyclone V デバイスのデザイン・セキュリティーに関する詳細を提供しています。
- Configuration, Design Security, and Remote System Upgrades in Stratix V Devices
  Stratix V デバイスのデザイン・セキュリティーに関する詳細を提供しています。
- インテル Arria 10 デバイスにおけるコンフィグレーション、デザイン・セキュリティー、およびリモート・システム・アップグレード
  インテル Arria 10 デバイスのデザイン・セキュリティーに関する詳細を提供しています。

- Configuration, Design Security, and Remote System Upgrades in Intel Cyclone 10 GX Devices
  インテル Cyclone 10 デバイスのデザイン・セキュリティーに関する詳細を提供しています。

JTAG セキュアモードでの検証

28nm FPGA で改ざん防止ビットがイネーブルされていると、非必須の JTAG 命令はディスエーブルされます。KEY_VERIFY を改ざん防止ビットのプログラム中に実行すると、TDI では BYPASS レジスターを指します。このため、KEY_VERIFY 命令を改ざん防止ビットが設定されているときに実行すると、0x0 (hex) が返されます。

デバイス内の改ざん防止プログラムの有無を確認するには、ユーザー定義パターンを KEY_VERIFY 命令実行時にシフトし、受信した TDO パターンに 0 がシフトしていることを確認します。

20nm FPGA では、KEY_VERIFY 命令を JTAG セキュアモードで実行することができます。20nm FPGA の JTAG セキュアモード中に検証を実行するには、USERCODE 命令の実行時に 0x0 (hex) の値が返されることが予想されます。

例-7: JTAG セキュアモードでの検証例

0x15A (バイナリーでは 1 0101 1010) にシフトします。改ざん防止ビットがプログラムされている場合、KEY_VERIFY=BYPASS なので、0 1011 0100 で最後の 0 は BYPASS レジスターの内容であると予想してください。

暗号化機能がイネーブルされたシリアル・フラッシュ・ローダーのサポート

インテルでは、インシステム・プログラミング (ISP) ソリューションの提供をシリアル・コンフィグレーション・デバイスである Serial Flash Loader Intel FPGA IP コアに対して行っています。デザインのシリアル・フラッシュ・ローダー (SFL) ブロックをインスタンス化すると、シリアル・コンフィグレーション・デバイスに保存されているデザインの更新が柔軟になります。これには、コンフィグレーション・デバイスの再プログラムを AS インターフェイスを介して行う必要はありません。

FPGA の JTAG インターフェイスがアクセス可能な限り、SFL ソリューションをアプリケーションに対して使用することができます。デザイン・セキュリティー機能に改ざん防止ビットが設定されていると、SFL ソリューションは機能しません。改ざん防止ビットが設定されている場合、JTAG プログラミングはサポートされませんが、デザインで Serial Flash Loader IP コアをインスタンス化し、SFL プログラミングを初回実行してから、不揮発性キー・プログラミングを改ざん保護ビット付きでの設定を FPGA 内で行うことができます。

単一の FPGA デバイスチェーンに対して暗号化がイネーブルされたシリアル・フラッシュ・ローダーのサポート

Serial Flash Loader IP コアを、暗号化機能がイネーブルされた状態で単一の FPGA デバイスチェーン内で使用するには、次の手順を実行します。

1. インテル Quartus Prime 開発ソフトウェアを起動します。
2. Serial Flash Loader IP コアを FPGA のトップレベルデザインでインスタンスします。
3. デザインのコンパイルを次のいずれかのオプションを用いて実行します。暗号化されていない**.sof** が生成されます。
   a. Processing メニューで、**Start Compilation** をクリックします。

b. Processing メニューの **Start** にカーソルを合わせて、**Start Assembler** をクリックします。

4. 次の手順に従って、**.sof** ファイルを**.jic** ファイルに変換します。
   a. File メニューで **Convert Programming Files** を選択します。
   b. **Convert Programming Files** ダイアログボックスから **Programming file type** フィールドでスクロールして、**JTAG Indirect Configuration File (.jic)** を選択します。
   c. **Configuration device** フィールドでシリアル・コンフィグレーション・デバイスを指定します。
   d. **File name** フィールドでターゲット・ディレクトリーを探して、出力ファイル名を指定します。
   e. **Input files to convert** セクションで**.sof** データを強調表示します。
   f. **Add File** をクリックします。
   g. **.jic** ファイルに変換するファイルに変換する**.sof** ファイルを選択します。
   h. **OK** をクリックします。
   i. **.sof** ファイル名をクリックして**.sof** ファイルを暗号化します。

      注意: .sof ファイルを暗号化するには、インテル Quartus Prime 開発ソフトウェアを使用したシングルデバイス .ekp ファイルの生成およびコンフィグレーション・ファイルの暗号化 (14 ページ)のステップ 7 (15 ページ) を参照してください。

   j. Flash Loader を強調表示し、**Add Device** をクリックします。
   k. **OK** をクリックして、**Select Devices** ページを表示します。
   l. シリアル・コンフィグレーション・デバイスのプログラムに使用するターゲットの FPGA を選択します。
   m. **OK** をクリックします。
5. シリアル・コンフィグレーション・デバイスのプログラムを、暗号化された**.jic** ファイルを使用して実行します。
6. キーを FPGA デバイス内にプログラムします。

   注意: キーを単一の FPGA デバイスにプログラムするには、インテル Quartus Prime 開発ソフトウェアを使用したシングルデバイスの揮発性または不揮発性キーのプログラミング (20 ページ) の手順に従います。

7. 暗号化された FPGA のコンフィグレーションが、プログラムされたシリアル・コンフィグレーション・デバイスを使用して行われます。

   注意: キーのプログラムに**.jam** ファイルを使用するには、**.jic** ファイルを**.jam** ファイルに変換してください。

関連情報

- AN 370: Using the Intel FPGA Serial Flash Loader IP Core with the Intel Quartus Prime Software
- Device Datasheet for Arria II Devices
  PS および FPP コンフィグレーション手法のタイミング・パラメーターに関する詳細情報を提供しています。
- DC and Switching Characteristics for Stratix IV Devices
  PS および FPP コンフィグレーション手法のタイミング・パラメーターに関する詳細情報を提供しています。

- Arria V Device Datasheet
  PS および FPP コンフィグレーション手法のタイミング・パラメーターに関する詳細情報を提供しています。
- Cyclone V Device Datasheet
  PS および FPP コンフィグレーション手法のタイミング・パラメーターに関する詳細情報を提供しています。
- Stratix V Device Datasheet
  PS および FPP コンフィグレーション手法のタイミング・パラメーターに関する詳細情報を提供しています。
- インテル Arria 10 デバイス・データシート
  PS および FPP コンフィグレーション手法のタイミング・パラメーターに関する詳細情報を提供しています。
- インテル Cyclone 10 GX デバイス・データシート
  PS および FPP コンフィグレーション手法のタイミング・パラメーターに関する詳細情報を提供しています。

28nm および 20nm FPGA の JTAG セキュアモード

FPGA は、次の場合に電源投入時に JTAG セキュアモードになります。

- 28nm FPGA の改ざん防止ビットをイネーブルしたとき
- 20nm FPGA の JTAG セキュア設定をイネーブルしたとき

注意: 20nm FPGA では、LOCK および UNLOCK の JTAG 命令はサポートしていません。外部 JTAG をロック解除して非必須の JTAG 命令にアクセスすることはできません。

JTAG セキュアモードでは、多くの JTAG 命令がディスエーブルされます。JTAG セキュアモードの 28nm および 20nm FPGA では、必須の IEEE Std. 1149.1 および IEEE Std. 1149.6 BST JTAG 命令のみを使用できます。非必須の JTAG 命令を、FPGA が JTAG セキュアモードのときに実行しようとすると、BYPASS JTAG 命令チェーンが選択され、命令は実行されません。

表 17. 必須および非必須の IEEE Std. 1149.1 および IEEE Std. 1149.6 BST JTAG 命令

必須の IEEE Std. 1149.1 および IEEE Std. 1149.6 BST JTAG 命令	非必須の IEEE Std. 1149.1 および IEEE Std. 1149.6 BST JTAG 命令
• BYPASS • EXTEST • IDCODE • LOCK • UNLOCK • SAMPLE/PRELOAD • SHIFT_EDERROR_REG	• CONFIG_IO • CLAMP • EXTEST_PULSE (18) • EXTEST_TRAIN (18) • HIGHZ • KEY_CLR_VREG • KEY_VERIFY (18) • PULSE_NCONFIG • USERCODE

(18) これらの JTAG 命令は、JTAG セキュアモード時に 20nm FPGA に対して実行することができます。

28nm FPGA の場合、非必須の JTAG 命令のアクセスをイネーブルするには、UNLOCK JTAG 命令を発行して、JTAG セキュアモードを非アクティブ化してください。LOCK 命令を発行して、デバイスを JTAG セキュアモードに戻します。LOCK と UNLOCK JTAG 命令の両方を発行できるのは、ユーザーモード中で内部 JTAG インターフェイスを使用した場合のみです。これら 2 つの命令の発行に外部 JTAG ピンを使用しても、JTAG セキュアモードのアクティブ化または非アクティブ化はできません。

LOCK および UNLOCK JTAG 命令によって JTAG セキュアモードをアクティブ化または非アクティブ化できるのは、改ざん防止ビットがイネーブルされている FPGA でのみです。この 2 つの命令を改ざん防止ビットがディスエーブルされているデバイスで発行しても、JTAG セキュアモードはオンまたはオフになりません。

内部 JTAG インターフェイス

28nm および 20nm FPGA の JTAG 制御ブロックにアクセスするには、外部 JTAG インターフェイスと内部 JTAG インターフェイスの 2 つのインターフェイスがあります。

外部 JTAG インターフェイスの JTAG 制御ブロックへのアクセスは、物理 JTAG ピン (TCK 、TDI 、TDO および TMS) を介して行います。FPGA コンフィグレーションに外部 JTAG インターフェイスを使用するのは、JTAG コンフィグレーション手法をプログラミング・ケーブルを介して使用する場合、または JTAG 命令を外部プレーヤーまたは JAM プレーヤーや JTAG チェーン・デバッガー・ツールなどのプロセッサーを使用して行う場合です。内部 JTAG インターフェイスとは、内部 FPGA コア・ファブリックからの TCK 、TDI 、TDO 、および TMS 信号と JTAG コントロール・ブロックの間の接続を指します。

JTAG コントロール・ブロックにアクセスするには、このインターフェイスのいずれかを一度に 1 つずつ使用します。たとえば、内部 JTAG インターフェイスを使用すると、JTAG 制御ブロックへの外部 JTAG インターフェイスはディスエーブルされます。内部 JTAG インターフェイスにアクセスするには、WYSIWYG アトムを インテル Quartus Prime デザインに含めてください。

図 -3: 内部および外部 JTAG インターフェイスの接続

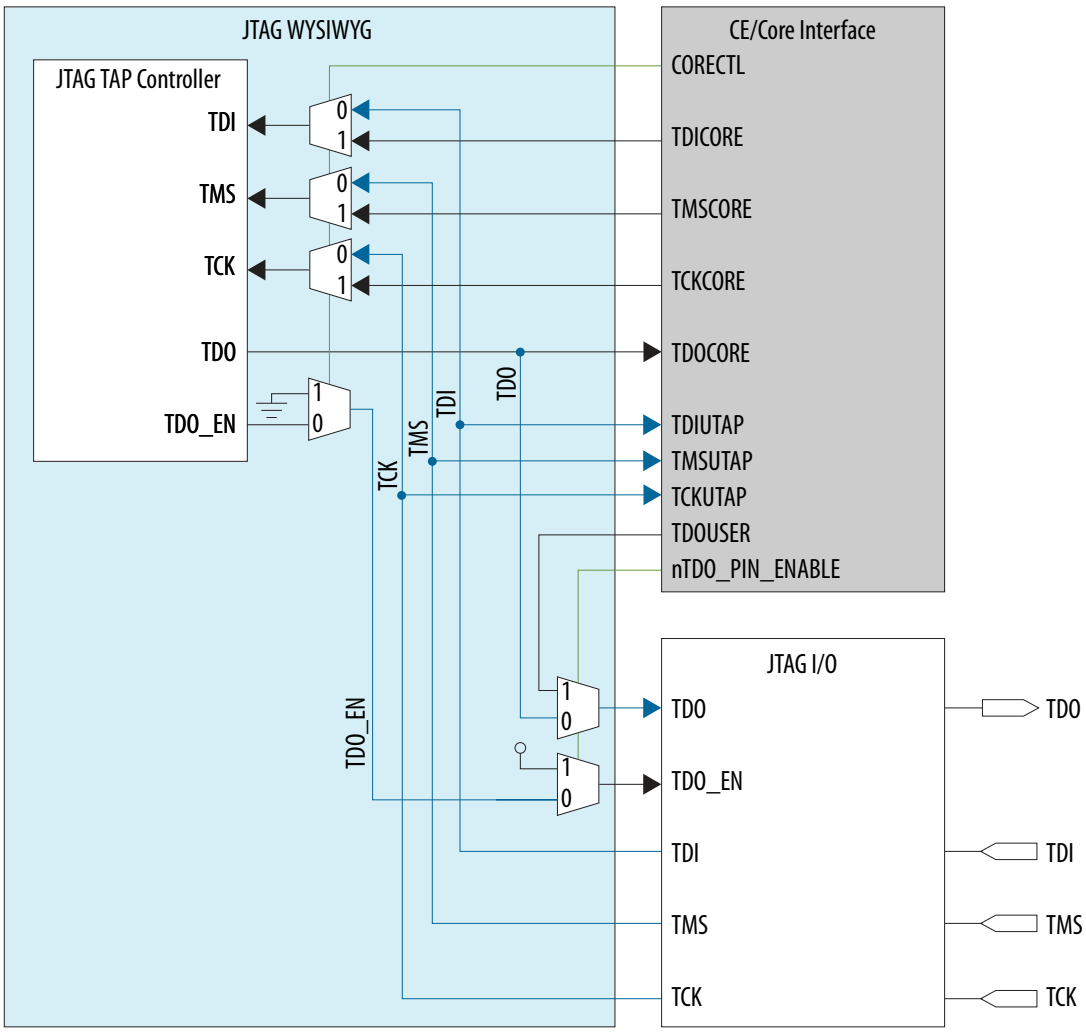


表 18. 28nm および 20nm FPGA の WYSIWYG アトム

デバイスファミリー	JTAG WYSIWYG アトム
Arria V	arriav_jtag <jtagblock_name> (.clkdruser(), .corectl(), .runidleuser(), .shiftuser(), .tck(), .tckcore(), .tckutap(), .tdi(), .tdicore(), .tdiutap(), .tdo(), .tdocore(), .tdouser(), .tdoutap(), .tms(), .tmscore(), .tmsutap(),

continued...

デバイスファミリー	JTAG WYSIWYG アトム
	.updateuser(), .usr1user());
Cyclone V	cyclonev_jtag <jtagblock_name> (.clkdruser(), .corectl(), .runidleuser(), .shiftuser(), .tck(), .tckcore(), .tckutap(), .tdi(), .tdicore(), .tdiutap(), .tdo(), .tdocore(), .tdouser(), .tdoutap(), .tms(), .tmscore(), .tmsutap(), .updateuser(), .usr1user());
Stratix V	stratixv_jtag <jtagblock_name> (.clkdruser(), .corectl(), .runidleuser(), .shiftuser(), .tck(), .tckcore(), .tckutap(), .tdi(), .tdicore(), .tdiutap(), .tdo(), .tdocore(), .tdouser(), .tdoutap(), .tms(), .tmscore(), .tmsutap(), .updateuser(), .usr1user());
インテル Arria 10	twentynm_jtag <jtagblock_name> (.tms(), .tck(), .tdi(), .ntrst(), .tdoutap(), .tdouser(), .tmscore(), .tckcore(), .tdicore(), .ntrstcore(), .tmscorehps(), .tckcorehps(), .tdicorehps(), .ntrstcorehps(), .tdocorefrwl(), .corectl(), .ntdopinena(), .tdo(), .tmsutap(), .tckutap(), .tdiutap(), .ntrstutap(), .tmsuhps(), .tckuhps(), .tdiuhps(), .ntrstuhps(), .tmscoreout(), .tckcoreout(), .tdocorehps(), .ntrstcoreout(),

continued...

デバイスファミリー	JTAG WYSIWYG アトム
	.shiftuser(), .clkdruser(), .updateuser(), .runidleuser(), .usr1user(), .tdocore(),);
インテル Cyclone 10 GX	twentynm_jtag <jtagblock_name> (.tms(), .tck(), .tdi(), .ntrst(), .tdoutap(), .tdouser(), .tmscore(), .tckcore(), .tdicore(), .ntrstcore(), .tdocorefrwl(), .corectl(), .ntdopinena(), .tdo(), .tmsutap(), .tckutap(), .tdiutap(), .ntrstutap(), .tmscoreout(), .tckcoreout(), .tdocorehps(), .ntrstcoreout(), .shiftuser(), .clkdruser(), .updateuser(), .runidleuser(), .usr1user(), .tdocore(),);

表 19. WYSIWYG アトムのポートの機能

ポート	入力/出力	機能
<jtagblock_name>	—	*Arriaii_jtag* WYSIWYG アトムの識別子。Verilog HDL、VHDL、および AHDL など指定の記述言語に対して有効な識別子名を表します。
.corectl()	入力	JTAG コントロール・ブロックへのアクティブ high 入力。コア・インターフェイスからの内部 JTAG アクセスをイネーブルします。コンフィグレーション後に FPGA がユーザーモードに入る時、このポートはデフォルトで low になっています。このポートをロジック high にすると、内部 JTAG インターフェイスがイネーブルされ (同時に外部 JTAG インターフェイスがディスエーブルされ) 、ロジック low にすると、内部 JTAG インターフェイスがディスエーブルされます (同時に外部 JTAG インターフェイスがイネーブルされます) 。
.tckcore()	入力	コア TCK 信号。(19)
.tdicore()	入力	コア TDI 信号(19)
.tdocore()	出力	コア TDO 信号(19)
.tmscore()	入力	コア TMS 信号(19)
.clkdruser()	入力/出力	これらのポートは、内部 JTAG インターフェイスを使用した JTAG セキュアモードをイネーブルするためには使用されていないため、未接続のままにできます。
.runidleuser()		
.shiftuser(),		

continued...

(19) 外部 JTAG インターフェイスについては、各デバイスのデータシートで JTAG コンフィグレーションのタイミング仕様を参照してください。内部 JTAG インターフェイスの場合は、タイミング制約およびタイミング・クロージャーの解析をこれらのパスに対して実行し、セットアップまたはホールド時間の要件を満たす必要があります。

ポート	入力/出力	機能
.tck()		
.tckutap()		
.tdi()		
.tdiutap()		
.tdo()		
.tdouser()		
.tdoutap()		
.tms()		
.tmsutap()		
.updateuser()		
.usr1user()		

関連情報

- EthernetBlaster Communications Cable User Guide
- EthernetBlaster II Communications Cable User Guide
  インテル FPGA イーサネット・ケーブル の TCK クロック速度変更に関する詳細情報を提供しています。
- Device Datasheet for Arria II Devices
  JTAG ダウンロード・ケーブルの使用に必要な特定の電圧に関する詳細情報を提供しています。
- DC and Switching Characteristics for Stratix IV Devices
  JTAG ダウンロード・ケーブルの使用に必要な特定の電圧に関する詳細情報を提供しています。
- Arria V Device Datasheet
  JTAG ダウンロード・ケーブルの使用に必要な特定の電圧に関する詳細情報を提供しています。
- Cyclone V Device Datasheet
  JTAG ダウンロード・ケーブルの使用に必要な特定の電圧に関する詳細情報を提供しています。
- Stratix V Device Datasheet
  JTAG ダウンロード・ケーブルの使用に必要な特定の電圧に関する詳細情報を提供しています。
- インテル Arria 10 デバイス・データシート
  JTAG ダウンロード・ケーブルの使用に必要な特定の電圧に関する詳細情報を提供しています。
- インテル Cyclone 10 GX デバイス・データシート
  JTAG ダウンロード・ケーブルの使用に必要な特定の電圧に関する詳細情報を提供しています。
- Stratix V E, GS, and GX Device Family Pin Connection Guidelines
- Stratix V GT Device Family Pin Connection Guidelines
- Arria V GT, GX, ST and SX Device Family Pin Connection Guidelines
- Arria V GZ Device Family Pin Connection Guidelines
- Stratix IV GX and E Device Family Pin Connection Guidelines
- Stratix IV GT Device Family Pin Connection Guidelines
- Arria II Device Family Pin Connection Guidelines
- インテル Arria 10 GX、GT、および SX デバイスファミリー ピン接続ガイドライン

JTAG セキュアモードのデザイン例

このデザイン例で説明する内容は次の通りです。

- 内部 JTAG WYSIWYG アトムのインスタンス化。
- インテル Quartus Prime 開発ソフトウェアでのユーザーロジック実装による LOCK および UNLOCK JTAG 命令の実行。

このリファレンス・デザインのターゲットは、改ざん防止ビットがイネーブルされている Arria V デバイスです。このデザイン例は、他の 28nm FPGA にも適用できます。

関連情報

AN 556 Design Files

デザイン例の インテル Quartus Prime デザイン・コンポーネント

表 20. インテル Quartus Prime Arria V デバイスのデザイン・コンポーネント

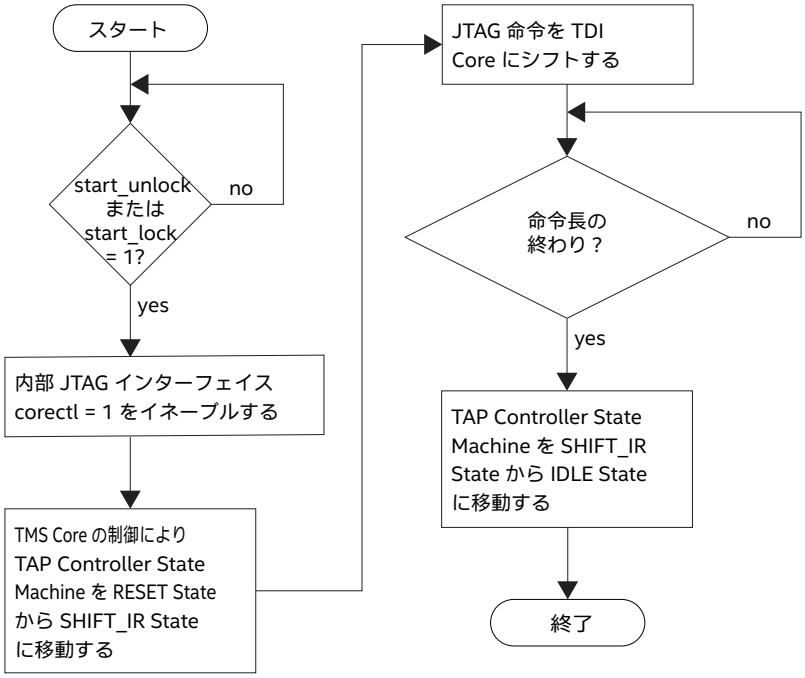
コンポーネント	機能および説明
JTAG_Lock_Unlock.bdf	リファレンス・デザインのトップ・エンティティー。
JTAG_Lock_Unlock_wysiwyg.v	Arria V デバイスの WYSIWYG アトムのインスタンス化用 Verilog コード。このコードを表 18 (33 ページ) に従って修正し、他の 28nm との迎合性を確保してください。
ALTINT_OSC.v	内部オシレーター・クロックソースの IP コアのインスタンス化。このリファレンス・デザインでは、内部オシレーターからのクロックソースを使用してユーザーロジックが駆動され、外部クロックソースが不要になります。
User_logic_control_block.v	Arria V デバイスの WYSIWYG アトムによって JTAG 命令を実行する Verilog ファイルの例。このコードは、修正してデザインの要件や制限に合わせたり、他の同様の実装と置き換えたりすることができます。
Pulse_nconfig.jam	この JAM ファイルを使用して PULSE_NCONFIG JTAG 命令を実行し、JTAG セキュアモードを確認します。詳細は JTAG セキュアモードの検証 (39 ページ) を参照してください。このファイルはオプションであり、他の方法と置き換えて JTAG セキュアモードを検証することができます。

LOCK および UNLOCK JTAG 命令

このリファレンス・デザインを Arria V デバイスにコンフィグレーションする際に改ざん防止ビットがイネーブルされていると、Arria V デバイスは、電源投入およびコンフィグレーション後に JTAG セキュアモードになるため、必須の JTAG 命令しか実行できません。

JTAG セキュアモードをディスエーブルするには、ユーザーロジックの start_unlock ポートをトリガーして UNLOCK JTAG 命令を発行します。start_unlock ポートが high になった後、UNLOCK JTAG 命令が発行されます。UNLOCK JTAG 命令が発行された後、デバイスは JTAG セキュアモードを終了します。これにより、必須と非必須の両方の JTAG 命令が許可されます。

図 -4: LOCK または UNLOCK JTAG 命令の実行



ユーザーロジックの `start_lock` ポートでは、LOCK JTAG 命令の実行をトリガーします。LOCK JTAG 命令の機能は、デバイスを JTAG セキュアモードに戻すことです。

表 21. ユーザーロジックの入力および出力ポート

ポート	入力/出力	機能
`clk_in`	入力	ユーザーロジックのクロックソース。ユーザーロジックの f_{MAX} は、タイミング・クロージャー解析に依存します。タイミング制約を適用し、パスでタイミング解析を実行して、f_{MAX} を決定してください。
`start_lock`	入力	ロジック high。LOCK JTAG 命令の内部 JTAG インターフェイスに対する実行をトリガーします。
`start_unlock`	入力	ロジック high。UNLOCK JTAG 命令の内部 JTAG インターフェイスに対する実行をトリガーします。
`jtag_core_en_out`	出力	`User_logic_control_block` の出力。このポートは、JTAG WYSIWYG アトムの `corectl` ポートに接続され、内部 JTAG インターフェイスをイネーブルします。
`tck_out`	出力	`User_logic_control_block` の出力。このポートは、JTAG WYSIWYG アトムの `tck_core` ポートに接続されています。
`tdi_out`	出力	`User_logic_control_block` の出力。このポートは、JTAG WYSIWYG アトムの `tdi_core` ポートに接続されています。
`tms_out`	出力	`User_logic_control_block` の出力。このポートは JTAG WYSIWYG アトムの `tms_core` ポートに接続されています。
`indicator`	出力	この出力ピンのロジック high は、LOCK または UNLOCK JTAG 命令の実行の完了を示しています。

関連情報

AN 39: IEEE 1149.1 JTAG Boundary-Scan Testing in Altera Devices

JTAG セキュアモードの検証

インテルでは、デバイスが正常に JTAG セキュアモードに入ったかまたは終了したかを確認するのに、非必須の JTAG 命令を実行することをお勧めしています。JTAG セキュアモードの検証にリファレンス・デザインを使用する場合[20]、次の手順を実行します:

1. FPGA の電源投入

   FPGA の電源投入後は、改ざん防止ビットがイネーブルされているため、FPGA は JTAG セキュアモードになります。

2. FPGA コンフィグレーション

   リファレンス・デザインを FPGA にコンフィグレーションします。FPGA は耐改ざん性があり、暗号化されたコンフィグレーション・ファイルのみを受け入れるため、リファレンス・デザインを暗号化ファイルでコンフィグレーションする必要があります。これについては、暗号化されたコンフィグレーション・データを使用した 40nm、28nm または 20nm FPGA のコンフィグレーション (23 ページ) で説明しています。デバイスが確実にユーザーモードに入るようにするには、CONFDONE ピンをチェックするか、counter_output ピンを観察します。デバイスが正常にユーザーモードに入ると、CONFDONE ピンが high になり、counter_output ピンがトグルします。

3. JTAG セキュアモードの確認

   デバイスがユーザーモードに入ったら、PULSE_NCONFIG JTAG 命令を JTAG ピンを使用して発行します。デザインに添付されている pulse_nconfig.jam ファイルを使用します。pulse_nconfig.jam ファイルを実行するには、quartus_jli または JAM プレーヤーを使用します。PULSE_NCONFIG JTAG 命令は、デバイスのリコンフィグレーションをトリガーします。デバイスが JTAG セキュアモードの場合、リコンフィグレーションは行われません。これは、PULSE_NCONFIG JTAG 命令は、非必須 JTAG 命令だからです。これを確認するには、CONFDONE ピンと counter_output ピンを観察します。リコンフィグレーションが行われなかった場合、CONFDONE ピンは high のままで、counter_output ピンはトグルを続けます。

4. UNLOCK JTAG 命令の実行

   ユーザーロジックの start_unlock ポートをロジック high にします。UNLOCK JTAG 命令が完了すると、indicator ポートは high になります。

5. JTAG セキュアモードの確認

   UNLOCK JTAG 命令が完了したら、PULSE_NCONFIG JTAG 命令の再発行を外部 JTAG ピンを使用して行います。デバイスが JTAG セキュアモードになっていない場合は、PULSE_NCONFIG JTAG 命令によってデバイスのリコンフィグレーションがトリガーされます。CONFDONE ピンと counter_output ピンを観察して、デバイスのリコンフィグレーションを監視します。CONFDONE ピンは high から low になり、counter_output ピンはデバイスのリコンフィグレーション中にトグルを停止します。

関連情報

AN 425: Using the Command-Line Jam STAPL Solution for Device Programming
quartus_jli に関する詳細情報を提供しています。

[20] これらの手順は、改ざん防止ビットがイネーブルされている FPGA に対してのみ適用してください。

AN 556 の改訂履歴: インテル FPGA におけるデザイン・セキュリティー機能の使用

ドキュメント・バージョン	変更内容
2019.11.12	*キー・プログラミング*の項を次のとおり更新しました。 • インテル FPGA ダウンロード・ケーブル II の脚注を更新しました。 • 注記を追加し、他のサードパーティー製の不揮発性キーのプログラミングの場合、JTAG TCK パルス幅 (期間) を規制して適切なポリフューズ・プログラミングを行う必要があることを示しました。
2018.12.11	*インテル Quartus Prime 開発ソフトウェアを使用したシングルデバイス .ekp ファイルの生成およびコンフィグレーション・ファイルの暗号化*の項を更新して、重要ファイルの例を修正しました。
2018.06.15	• *デザイン・セキュリティー機能の概要*の項の説明を更新しました。 • *40nm および 28nm FPGA デザインのセキュリティー手法*の表の説明を更新しました。 • *揮発性キーと不揮発性キーの比較*の表のデザイン保護オプションに関する説明を削除しました。 • *キー・プログラミング方法*の表の インテル FPGA パラレル・ポート・ケーブルに関する注記を削除しました。 • *Qcrypt ツールの基本オプション*の表の `--decrypt` に関する説明を更新しました。 • *安全なコンフィグレーション・フローの実装手順*の項のスタンドアロン Qcrypt ツールを使用した `.rbf` の暗号化に関する情報を更新しました。 • *インテル Quartus Prime 開発ソフトウェアを使用したシングルデバイス .ekp ファイルの生成およびコンフィグレーション・ファイルの暗号化*の項にステップを追加しました。 • *改ざん防止ビット・プログラミングのイネーブルの手順*の項を追加しました。 • *各コンフィグレーション・スキームに対するデザイン・セキュリティーのサポート*の表にある PS および JTAG コンフィグレーション手法を更新しました。 • *セキュリティー・モードの検証*の項で参考例への説明を追加しました。 • *内部および外部 JTAG インターフェイスの接続*の図を更新しました。 • *ユーザーロジックの入力ポートと出力ポート*の表で、`jtag_core_en_out` 、`tck_out` 、`tdi_out` 、および `tms_out` ポートの機能を修正しました。 • インテルへのブランド変更に従って、次の IP コア名を変更しました。 — Intel FPGA Parallel Flash Loader IP コアの名前を Parallel Flash Loader Intel FPGA IP コアに変更しました。 — Intel FPGA Serial Flash Loader IP コアの名前を Serial Flash Loader Intel FPGA IP コアに変更しました。

日付	バージョン	変更内容
2017 年 12 月	2017.12.18	• インテル Cyclone 10 GX デバイスファミリーのサポートを追加しました。 • 「キー・プログラミングの仕様」を更新しました。TCK 期間の揮発性および不揮発性キーを更新しました。 • 「20nm FPGA のセキュリティー・モードの検証」の表を更新しました。ビット 11 および 13 の脚注を追加して、これらのビットがインテル Cyclone 10 GX デバイスに適用されないことを明確にしました。 • 最新のインテルブランド規格に更新しました。 • *Qcrypt ツールのセキュリティー・オプション*の表の `--lockto=<FILE_NAME.qlk>` セキュリティー・オプションの説明を更新しました。 • ドキュメント全体への軽微なテキストの変更を行いました。
2016 年 6 月	2016.06.01	• *Arria 10 Qcrypt ツール*の情報を追加しました。 • 改ざん防止ビットと JTAG セキュアに関する情報 (20nm FPGA で個別にイネーブル可能) を追加しました。 • 20nm FPGA のソフトウェア要件を追加しました。 • *インテル Quartus Prime 開発ソフトウェアを使用したシングルデバイス .ekp ファイルの生成およびコンフィグレーション・ファイルの暗号化*の項で 20nm FPGA に関する 2 つのステップを追加しました。 • 20nm FPGA の JTAG セキュア検証方法を追加しました。 • EXTEST_PULSE、EXTEST_TRAIN、KEY_VERIFY に関する注記(JTAG セキュアモード中の JTAG 命令の使用が可能)を追加しました。 • Arria 10 の JTAG アトムを更新しました。

continued...

日付	バージョン	変更内容
		• 20nm FPGA のデザイン・セキュリティー手法についての注記を追加しました。 • 暗号化と圧縮についての注記(20nm FPGA で同時に使用は不可)を追加しました。 • 20nm FPGA の TCK 期間の不揮発性キーの仕様を更新しました。 • USB-Blaster の 20nm FPGA の揮発性および不揮発性キーに対するサポートに関する注記が追加されました。
2015 年 11 月	2015.11.02	• 注記 (ユーザーは EthernetBlaster II の TCK 速度を必要な TCK 期間に設定する必要があること) を追加し、「EthernetBlaster II Communications Cable ユーザーガイド」のリンクを追加しました。 • *Quartus II* のインスタンスを *Quartus Prime* に変更しました。
2015 年 6 月	2015.06.15	JTAG セキュアモードのデザイン例へのリンクを追加しました。
2015 年 5 月	2015.05.04	.key ファイルの例の合計文字数を修正しました。
2015 年 1 月	2015.01..23	• 20nm FPGA (Arria 10) のサポートを追加しました。 • 20nm の JAM ファイルの例を追加しました。 • 20nm のセキュリティー・モードの検証の表を追加しました。 • Arria 10 の JTAG WYSIWYG アトムを追加しました。 • アルテラ FPGA の AES モードを追加しました。
2014 年 12 月	2014.12.15	不揮発性セキュリティー・キー・プログラミングの USB-Blaster II サポートを追加しました。
2014 年 9 月	2014.09.30	• 「Quartus II ソフトウェアを使用したシングルデバイス .ekp ファイルの生成およびコンフィグレーション・ファイルの暗号化」で .key ファイルの例を追加しました。 • V_{CCBAT} 電圧ガイドラインを削除し、「ハードウェア要件」の値を更新したデバイスファミリーのピン接続ガイドラインのリンクを追加しました。 • 「28nm FPGA のセキュリティー・モード検証」に改ざん防止機能を持つモードに関する注記を追加しました。 • 「JTAG セキュアモードでの検証」のサブセクションを JTAG セキュアモード中の改ざん防止設定に追加しました。
2014 年 5 月	2014.05.19	不揮発性および揮発性キーのストレージの項を更新して、有効な MSEL ピン設定の使用に関する情報を含めました。
2013 年 6 月	2013.06.19	• 「FPGA のデザイン・セキュリティー手法」の表を更新して、追加のデザイン・セキュリティー機能を含めました。 • 「不揮発性および揮発性キーストレージ」の項を更新して、不揮発性および揮発性キーストレージの詳細を含めました。 • 「キー・プログラミング」の項を更新して、System General プログラミング・ツールを使用した 28nm および 40nm の両方の FPGA のサポートを含めました。 • 「ハードウェア要件」の項を更新して、「キー・プログラミングの仕様」の表を更新しました。 • 「安全なコンフィグレーション・フローの実装手順」を更新しました。 • 「ステップ 1: .ekp ファイルの生成およびコンフィグレーション・ファイルの暗号化」、「ステップ 2a: 揮発性キーの FPGA へのプログラム」、および「ステップ 2b: 揮発性キーの FPGA へのプログラム」の項を更新しました。 • 「Quartus II ソフトウェアを使用したシングルデバイスの .ekp ファイルの生成およびコンフィグレーション・ファイルの暗号化方法」を更新し、28nm および 40nm FPGA の暗号化キーに関する情報を含めました。 • 「セキュリティー・モードの検証」の項を更新して、28nm および 40nm FPGA のセキュリティー・モードとそれに関連するビット値を更新しました。 • 「28nm FPGA の JTAG セキュアモード」の項を更新し、必須および非必須 JTAG 命令、内部 JTAG インターフェイスと外部 JTAG インターフェイス、WYSIWYG アトムの機能、および JTAG セキュアモードのデザイン例に関する詳細を含めました。 • すべてのトピック内のすべてのリンクを「関連情報」の項に移動し、簡単に参照できるようにしました。

continued...

日付	バージョン	変更内容
2012 年 6 月	2.1	• 表 1 および表 3 を更新しました。 • **.ekp** ファイル検証エラー情報を更新しました。 • 「ハードウェア要件」の項を更新しました。
2011 年 6 月	2.0	• Quartus II ソフトウェア・バージョン 11.0 リリース用にアプリケーション・ノートを更新しました。 • 特定のデバイス名を 40nm または 28nm FPGA に変更しました。 • 「セキュリティー・モードの検証」および「28nm FPGA の JTAG セキュアモード」の項を追加しました。 • 表 1 を追加しました。 • 表 5 を追加しました。 • 例 3、例 4、および例 5 を追加しました。 • 図 1 を更新しました。 • 軽微なテキストの編集を行いました。
2009 年 6 月	1.1	• 1 ページの「はじめに」を更新しました。 • 2 ページの「デザイン・セキュリティー機能の概要」を更新しました。 • 2 ページの「セキュリティー暗号化アルゴリズム」を更新しました。 • 3 ページの「不揮発性および揮発性キーストレージ」を更新しました。 • 4 ページの表 2 の注記 3 を更新しました。 • 4 ページの「ハードウェアおよびソフトウェア要件」を更新しました。 • 5 ページの表 3 の注記 1 を更新しました。 • 5 ページの「安全なコンフィグレーション・フローの実装手順」を更新しました。 • 17 ページの「ステップ 2a: 揮発性キーの Arria II GX または Stratix IV デバイスへのプログラム」を更新しました。 • 18 ページの「ステップ 2b: 不揮発性キーの Arria II GX または Stratix IV デバイスへのプログラム」を更新しました。 • 24 ページの「ステップ 3:暗号化されたコンフィグレーション・データによる Arria II GX または Stratix IV デバイスのコンフィグレーション」を更新しました。 • 28 ページの表 3 を追加しました。 • 6 ページの表 1 および 29 ページの図 26 を更新しました。
2009 年 3 月	1.0	初版