LatticeECP2/M
暗号化機能 使用ガイド

はじめに

すべてのラティスFPGAがコンフィグレーション・データの読み出しに対するセキュリティを提供します。即ちデバイスがリードバックされるとき、実際のコンフィグレーション・データの代わりにすべてゼロ出力になるようにヒューズを設定することができることを意味します。この種類の保護は業界では一般的であり、LatticeXP™ やMachXO™ デバイスファミリなどのように、コンフィグレーション・データがオンチップに保存されているなら、非常に良いセキュリティを提供します。しかしながら、コンフィグレーション・ビットストリームが外部のブートデバイスから来る場合、コンフィギュレーション・データを読むのはかなり簡単ですから、FPGAデザインへのアクセスを許容することになります。

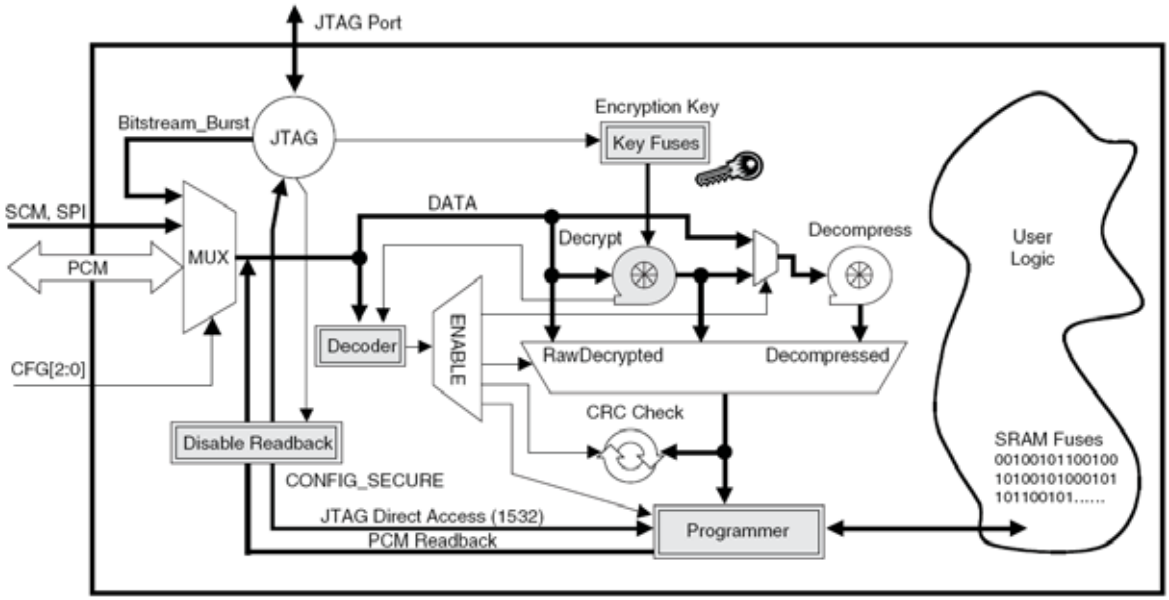
この理由で、ラティスからの最新のFPGA、LatticeECP2M™ とLatticeECP2™ "S" バージョンは、ビットストリームを保護するために128ビットのAES暗号方式(Advanced Encryption Standard)を提供します。ユーザは128ビット暗号鍵を選択し、その全くの管理制御ができ、そしてFPGA内で暗号鍵を保持する為に如何なる特別な電圧も不要です。

このドキュメントはこの新しいセキュリティ機能と、どのようにそれを利用するかについて説明します。

通常のコンフィグレーション手順

図16-1はLatticeECP2/M Sバージョンのビットストリーム暗号化データパスについて説明するブロック図です。以下のセクションを読む際にはこの図を参照してください。

図16-1 LatticeECP2/M Sバージョンのビットストリーム暗号化ブロック図



ラティスFPGAはsysCONFIG™ インターフェイスかJTAGインターフェイスを用いることによって、コンフィグレーションされます(表16-1を参照)。

表16-1 コンフィグレーション・ポートのインターフェイス

インターフェイス	ポート
sysCONFIG	SPI
	SPIm
	スレーブシリアル(SCM)
	スレーブパラレル(PCM)
ispJTAG™	IEEE 1532 (消去、プログラム、ベリファイ)
	ビットストリーム・バースト(高速プログラム)

1. 暗号化されたビットストリームはサポートしません。

sysCONFIGインターフェイスではシリアル・コンフィグレーション・モード (SCM)かSPIシリアル・フラッシュを用いてシリアルにデータを入力する、またはパラレル・コンフィグレーション・モード (PCM)を用いて並列に入力することが可能です。FPGAとコンフィグレーションデバイスとの接続は、一般にクロック、チップセレクト、ライト制御信号 (PCM 時)、およびデータより成ります。コンフィグレーションの間、特別なプリアンブルがビットストリーム内に検出されるまで、FPGAに書かれたすべてのデータが無視されます。プリアンブル後のすべてがコンフィグレーション・データです。通常はプリアンブルがBDB3(16進)ですが、暗号化されたビットストリームは異なったプリアンブルを含みます。

JTAGポート(IEEE1149.1とIEEEに1532の標準に適合する)は、Bitstream-Burstモード(Fast Program)または1532モード(消去、プログラム、ベリファイ)のデータを入力することができます。Bitstream-Burstモード(Fast Program)のために生成されたコンフィグレーション・ビットストリームはsysCONFIGモード用に生成されたビットストリームと同一です。ビットストリームはヘッダー、プリアンブル、コンフィグレーション・データ、およびフレームデータCRCを含んでいます。しかし、1532モードはデバイスをコンフィグレーションするために標準のJTAG命令を利用します。言い換えれば、コンフィグレーション・データファイルはコンフィグレーション・データだけを含んでいます。1532モード・データファイルはプリアンブルを含んでいないので、暗号化されたコンフィグレーション・ファイルを入力するためにこれを用いることはできません。

また、コンフィグレーション・インターフェイスであることに加えて、JTAGはユーザが128ビット暗号鍵のプログラムすることが可能です。実際は、JTAGは暗号鍵をプログラムする唯一の方法です。

LatticeECP2/Mコンフィグレーションの詳細な情報については、www.latticesemi.comのラティスウェブサイトで入手できるテクニカルノートTN1108 (LatticeECP2/M sysCONFIG Usage Guide)を参照してください。

ビットストリーム暗号化 / 復号化フロー

LatticeECP2/M Sバージョンは暗号化、非暗号化のビットストリームを共にサポートします。非暗号化フローはテクニカルノートTN1108でカバーされているので、本ドキュメントは暗号化フローに必要な追加ステップに的を絞ります。暗号化フローはLatticeECP2/Mの通常のFPGA設計フローに2ステップだけ、即ちコンフィグレーション・ビットストリームの暗号化と暗号鍵のプログラミングが加わります。

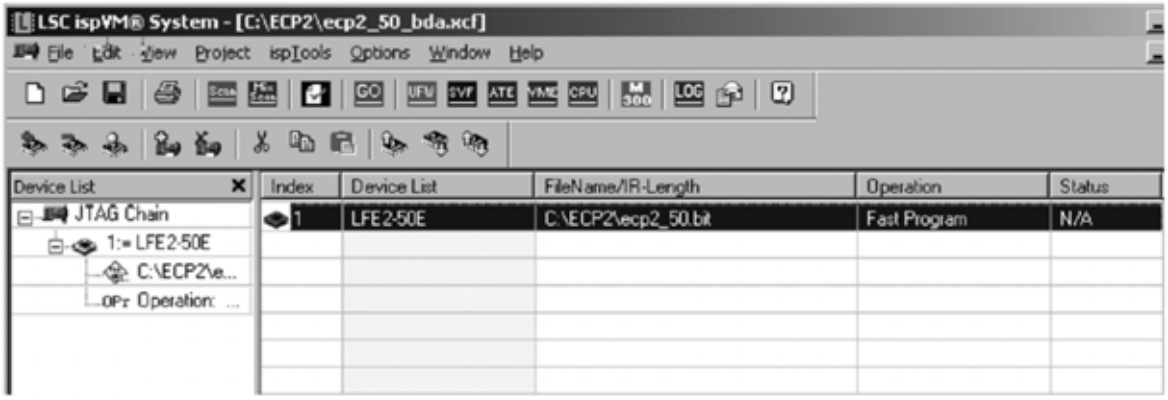
ビットストリームの暗号化

いかなる他のラティスFPGA設計フローのように、技術者は最初に暗号化機能をサポートするispLEVER® のバージョンを用いることでデザインを生成しなければなりません。デザインは、論理合成されて、マッピングされ、配置され、配線されて、そしてベリファイされます。技術者が生成されビットストリームのデザインに満足できるようになると、最終的なデバッグのためにFPGAにロードされます。デザインがデバッグされた後は、デザインにセキュリティを施す時です。

ispLEVERの適切なバージョンを用いて**Tools**プルダウン・メニューから**Security**機能を選択するか、または

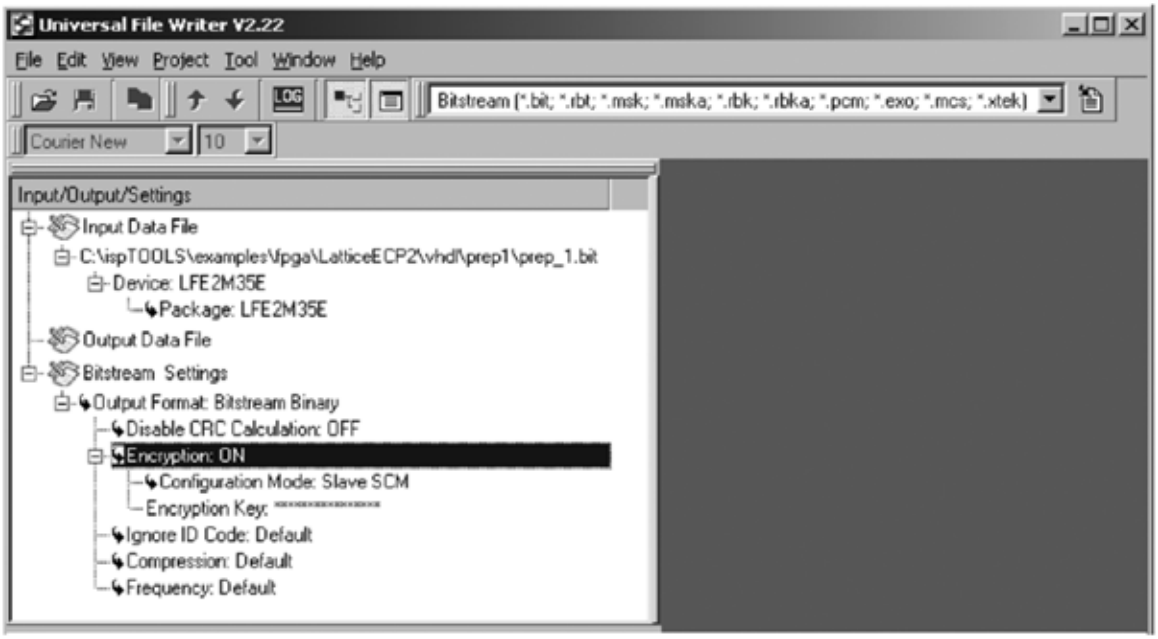
ラティスispVM® システム・ツールスイートの一部である**Universal File Writer** (UFW)を用いることによって、ビットストリームを暗号化することができます。ファイルはispVMを用いることで以下のように暗号化されます。

図16-2 ispVMメインウィンドウ



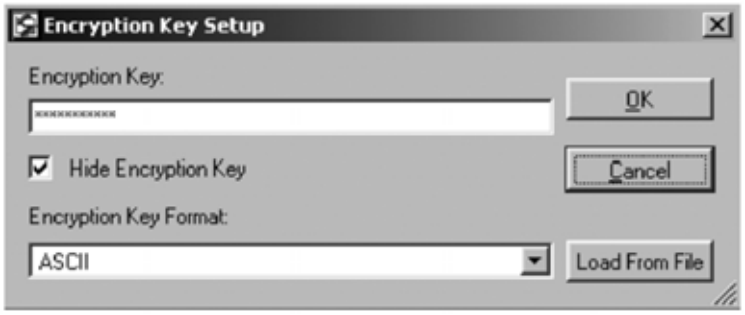
1. ispVMを起動します。ispLEVERから、或いはWindowsの**Start → Programs**メニューからispVMを起動することができます。図16-2と同様に見えるウィンドウが表示されます。ツールバーで**UFW**ボタンをクリックします。図16-3と同様のウィンドウが表示されます。

図16-3 Universal File Writer (暗号化オプション)



2. **Input Data File**をダブルクリックし、ispLEVERを用いて生成された暗号化されていないビットストリームにブラウズします。**Output Data File**をダブルクリックし、出力ファイル名を選択します。**Encryption**を右クリックしてONを選択します。**Configuration Mode**を右クリックして、SPIシリアル・フラッシュなどFPGAをコンフィグレーションするデバイスのタイプを選びます。**Encryption Key**を右クリックして**Edit Encryption Key**を選択します。図16-4と同様なウィンドウが表示されます。

図16-4 Encryption Key (暗号化鍵)ダイアログ・ウィンドウ



3. 所望の128ビット暗号鍵を入力します。暗号鍵は16進かASCIIで入力することができます。16進では0からFをサポートし、大文字と小文字を区別しません。ASCIIはすべての英数字およびスペースをサポートして、大文字と小文字を区別します。**注；** この暗号鍵を必ず覚えておいてください。暗号鍵を無くすると、ラティスは暗号化されたファイルを回復することができません。UFWメインウィンドウに戻るためにOKをクリックします。

4. メニューバーから **Project → Generate** をクリックして、暗号化されたビットストリーム・ファイルを生成します。

5. これで、ラティスispDOWNLOAD® ケーブル、サードパーティ・プログラマ、または通常は非暗号化ビットストリームをプログラムするのに用いられるいかなる他の方法も用いることで、直接不揮発コンフィグレーション記憶素子(SPIシリアル・フラッシュなど)にビットストリームをロードすることができます。しかしながら、LatticeECP2/Mが暗号化されたファイルからコンフィグレーションすることができるようになる前に、ファイルを暗号化するのに用いられた128ビット暗号鍵をFPGA内の一度のみプログラマムできる(OTP)ヒューズにプログラムしなければなりません。

128ビット暗号鍵のプログラミング

次のステップはLatticeECP2/MのOTPヒューズに128ビットの暗号鍵をプログラムすることです。このステップは製造フローでの柔軟性を許容するためにファイル暗号化と切り離されていることに留意してください。例えば、ボードメーカがSPIシリアル・フラッシュに暗号化されたファイルをプログラムし、暗号鍵はユーザの施設でプログラムされるかもしれません。このフローはデザインにセキュリティを与え、そしてユーザがデザインの過剰生産を制御することを可能にします。過剰生産は、サードパーティが認可されているより多くのボードを製造し、それらをグレーマーケットの顧客に販売するために起こります。暗号鍵が工場でプログラムされることで、工場が市場に投入される動作ボード数を制御します。LatticeECP2/M Sバージョンは FPGAにプログラムされたものと同じ128ビット暗号鍵で暗号化されたファイルからのみコンフィグレーションされます。

LatticeECP2/M Sバージョンに暗号鍵をプログラムするには、以下のように進めます。

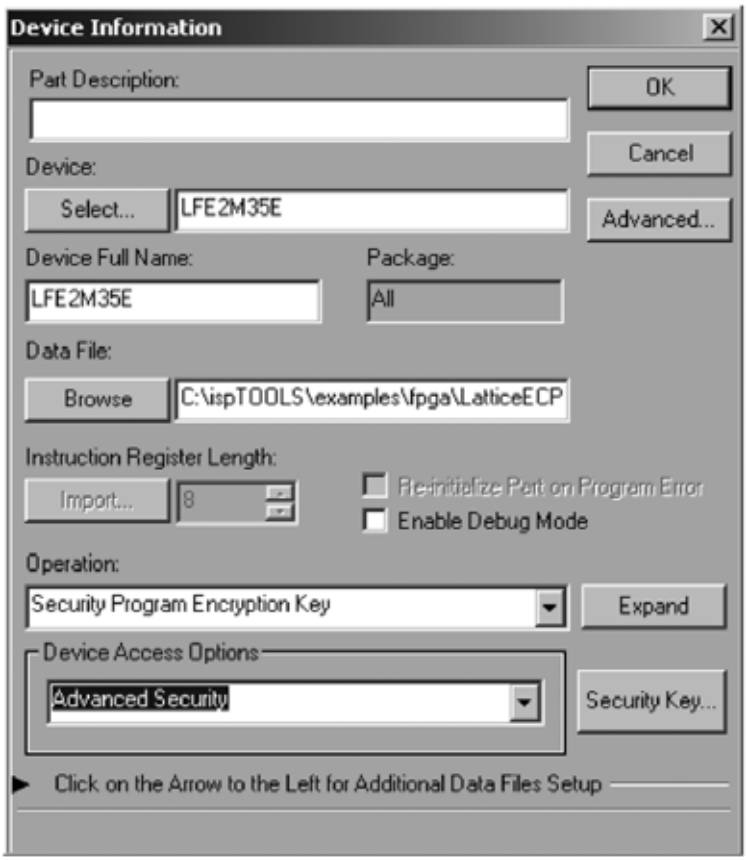
1. ラティスispDOWNLOADケーブルをPCからLatticeECP2/Mに配線されたJTAGコネクタに取り付けます。(JTAGポートを用いることでのみLatticeECP2/Mに128ビット暗号鍵をプログラムすることができることに留意します。) 電源をボードに供給します。

2. ispVMシステム・ソフトウェアを起動します。ispLEVERデザインツール内かWindowsの **Start → Program** メニューからispVMを起動できます。図16-2と同様なウィンドウが表示されます。ウィンドウがボードのJTAGチェインを表示しない場合、以下の通り進めます。さもなければ、ステップ3に進みます。

   a. ツールバーの**SCAN**ボタンをクリックして、JTAGチェーン内のラティスデバイスを検出します。図

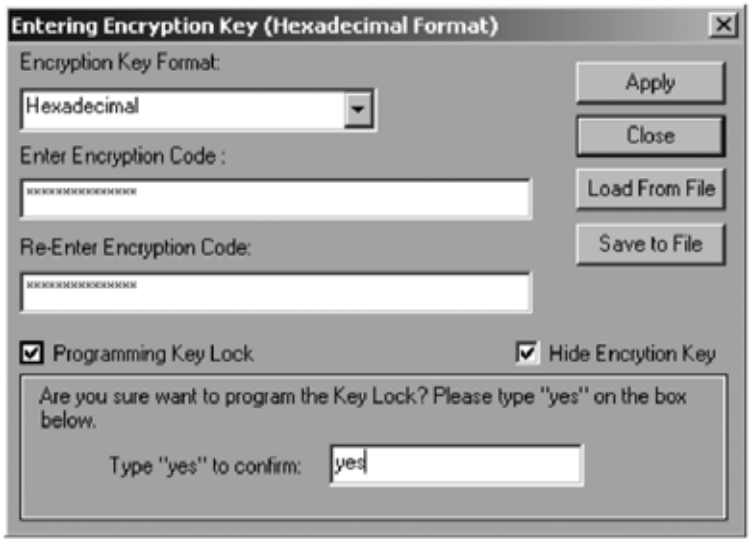
16-2で示すチェインには1つのデバイス、LatticeECP2のみがあります。

図16-5 デバイス情報ウィンドウ(暗号化オプション)



3. チェーン内のLatticeECP2を含む行の上をダブルクリックします。これは**Device Information**ウィンドウを開きます(図5を参照)。**Device Access Options**ドロップダウン・ボックスから**Security Mode**を選び、続いて**Security Key**ボタンを右クリックします。図16-6と同様なウィンドウが表示されます。

図16-6 暗号鍵の入力



4. 所望の128ビット暗号鍵を入力します。暗号鍵は16進かASCIIで入力することができます。16進では0からFをサポートし、大文字と小文字を区別しません。ASCIIはすべての英数字およびスペースをサポートし、大文字と小文字を区別します。この鍵はビットストリームを暗号化した鍵と同じでなければなりません。LatticeECP2/MはFPGAのOTPフューズと一致する鍵で暗号化してあるファイルからのみコンフィギュレーションできます。**注；**この暗号鍵を必ず覚えておいてください。**Key Lock**がいったんプログラムされると、ラティスセミコンダクターはOTP暗号鍵をリードバックすることができません。

a. **Save to File** ボタンを用いることでファイルに鍵を保存することができます。暗号鍵は、ユーザが選択する8文字のパスワードを用いることで暗号化されるでしょう。ファイルの名前は <プロジェクト名>.bek になります。将来、128ビット鍵を入力することの代わりに、単に **Load from File** をクリックしてパスワードを与えます。

5. Key Lockをプログラムすることは、128ビットの暗号鍵を安全にします。いったんKey Lockがプログラムされて、デバイスの電源サイクル後（オフしてオン）かPROGRAMNピンがトグルされると、デバイスから128ビットの暗号鍵を読み出すことはできません。満足であれば、確認のために **Yes** と入力し、次いで **Apply** をクリックします。

6. ispVMメインウィンドウ(図16-2)から、ツールバーで緑色の **GO**ボタンをクリックして、LatticeECP2/MのOTPヒューズに暗号鍵をプログラムします。完了すると、LatticeECP2/Mは丁度プログラムしたものと完全に一致する鍵で暗号化してあるファイルからのみコンフィギュレーションできます。

コンフィグレーションのベリファイ

暗号化されたビットストリームが使用されているときの付加的なセキュリティ・ステップは、SRAMファブリックからのリードバック・パスの自動的なブロックです。この場合、すべてのポートで読み出し動作はすべてのゼロを出力するでしょう。しかしながら、コンフィグレーション・ビットストリームを暗号化してあり、リードバックが禁止されたときさえ、それでも首尾よくビットストリームがFPGAにダウンロードされたことをベリファイする方法があります。

SRAMファブリックが直接プログラムされる場合、データは最初に復号され、そして次にFPGAはデータのCRC計算を実行します。すべてのCRCが一致すると、コンフィグレーションは成功です。CRCが一致しないと、DONEピンはLowのであるままで、INITNはHighからLowになります(この種の誤りに関する詳しい情報については、テクニカルノートTN1108、LatticeECP2/M sysCONFIGユーザガイドを参照してください)。

暗号化されたデータがSPIシリアル・フラッシュなどの不揮発コンフィグレーション・メモリに格納されるなら、データは暗号化されている状態で格納されます。暗号化されたコンフィグレーション・ファイルと格納されたデータの間で、ビット毎のベリファイを実行することができます。

ファイル形式

ベースバイナリ・ファイル形式は、すべての非暗号化された非1532コンフィグレーションモードと同じです。異なるファイルの種類(16進、バイナリ、ASCIIなど)は、結果としてデバイスをコンフィグレーションするのに用いられるかもしれませんが、ファイル中のデータは同じです。表16-2は非暗号化されたビットストリームの書式を示します。ビットストリームはコメント・フィールド、ヘッダー、プリアンブル、コンフィグレーション・セットアップ、およびデータから成ります。

表16-2 非暗号化コンフィグレーション・データ

フレーム	内 容	記 述
Comments	(Comment String)	ASCIIコメント（Argument）ストリングとターミネータ
Header	1111…1111	16ビットダミー
	1011110110110011	16ビット標準ビットストリーム・プリアンブル (0xBDB3)
Verify ID		64ビットのコマンドとデータ
Control Register0		64ビットのコマンドとデータ
Reset Address		32ビットのコマンドとデータ
Write Increment		32ビットのコマンドとデータ
Data 0		データ、16ビットCRC、ストップビット
Data 1		データ、16ビットCRC、ストップビット
・ ・	・ ・	・ ・
Data n-1		データ、16ビットCRC、ストップビット
End	1111….1111	ターミネータと16ビットCRC
Usercode		64ビットのコマンドとデータ
SED CRC		64ビットのコマンドとデータ
Program Security		32ビットのコマンドとデータ
Program Done		32ビットのコマンドとデータ、16ビットCRC
End	1111….1111	32ビット・ターミネータ

注: この表のデータは参照のためのみに意図されています。

表16-3は暗号化のためにビルドされた、しかしまだ暗号化されていないビットストリームを示します。ハイライトされた領域が暗号化されます。表16-2と表16-3の変化は以下を含みます。

・Program Securityフレーム(リードバックは禁止)がファイルの始めに移動されました。これはリードバックがコンフィグレーションの最初にオフになるようにするためです。これは完了前に誰かがコンフィグレーションを中断しまだ安全でないデータを読み出すことができないようにする、重要なセキュリティ機能です。

・usercodeのコピーは非暗号化されたコメント・ストリングに配置されます。ユーザに暗号化されたファイルを特定する方法を与えるためにこれを行いました。例えばファイル・インデックスや "ヒント" としてusercodeを用いることができます。usercode自身はコンフィグレーション・データファイルで暗号化されている一方、デバイス内では暗号化されていない事に留意してください。コンフィグレーション時にusercodeは復号化されて、JTAG usercodeレジスタに配置されます。これはユーザにデバイス内のデータを特定する方法を提供します。JTAG usercodeレジスタはいつでも、また全てのSRAMリードバック・パスがオフにされたときでさえリードバックできます。usercodeはどのような32ビットの値にも設定することができます。どうusercodeを設定するかに関する情報に関してはispLEVERのヘルプ機能を使用してください。

・CONFIG_MODEはispLEVERのグローバル・プリファレンスの1つですが、そのコピーは非暗号化されたコメント・ストリングで配置されます。CONFIG_MODEにはSPI/SPIm、スレーブSCM、またはスレーブPCMがあります。

グローバル・オプションのCOMPRESS_CONFIGがispLEVERのデザインプランナやUFWを用いることでON にされていると、データの圧縮が暗号化の前に実行されることに留意してください。

表16-3 暗号化直前のコンフィグレーション・ファイル

フレーム	内 容	記 述
Comments	(Comment String)	ASCIIコメント（Argument）ストリングとターミネータ
Header	1111…1111	16ビットダミー
		16ビット標準ビットストリーム・プリアンブル
Verify ID		64ビットのコマンドとデータ
Control Register0		64ビットのコマンドとデータ
Program Security		32ビットのコマンドとデータ
Reset Address		32ビットのコマンドとデータ
Write Increment		32ビットのコマンドとデータ
Data 0		データ、16ビットCRC、ストップビット
Data 1		データ、16ビットCRC、ストップビット
・ ・	・ ・	・ ・
Data n-1		データ、16ビットCRC、ストップビット
End	1111….1111	ターミネータと16ビットCRC
Usercode		64ビットのコマンドとデータ
SED CRC		64ビットのコマンドとデータ
Program Done		32ビットのコマンドとデータ、16ビットCRC
End	1111….1111	32ビット・ターミネータ

注:この表のデータは参照のためにのみ意図されています。陰影をつけられた領域は暗号化されます。

データ自体の明白な暗号化と共に、一度暗号化されたファイルには、非暗号化されたファイルからの付加的は違いがあります(表16-4、16-5、および16-6を参照)。

・3つのプリアンブルがあり、それらは暗号化プリアンブル、アライメント・プリアンブル、およびビットストリーム・プリアンブルです。アライメント・プリアンブルは暗号化されたデータの始まりを示します。表16-3によると、ビットストリーム・プリアンブルを含めてオリジナルのビットストリーム全体がすべて暗号化されています。コメント・ストリング、暗号化プリアンブル、ダミーデータ、およびアライメント・プリアンブルは暗号化されていません。

・FPGA内の復号化エンジンはそのタスクを実行するために幾らかの時間がかかります。2つの方法のうちの1つで付加的な時間を提供します。マスタ・コンフィグレーション・モード(SPIとSPIm)では、FPGAがコンフィグレーション・クロックをドライブします。したがって、付加的時間が必要なときにFPGAはコンフィグレーション・クロックの送出を止めます。スレーブ・コンフィグレーション・モード(Bitstream-Burst、スレーブシリアル、およびスレーブパラレル)においては、付加的時間を生成するためにデータを水増し(padding) しなければなりません。このために、暗号化されたデータ用の幾つかの異なるファイル形式があります(表16-4、表16-5、および表16-6を参照し)。ビットストリームを復号するために必要な時間のために、暗号化されたデータファイルからコンフィグレーションするためには、非暗号化されたファイルからよりは長い時間がかかることに留意してください。平均的に暗号化されたデータファイルからコンフィグレーションするにはおよそ50%長い時間かかりますが、これはスレーブモードで暗号化されたファイルサイズが非暗号化されたバージョンよりおよそ50%大きいことを意味します。

表16-4 マスタモードの暗号化ファイル形式

フレーム	内 容	記 述
Comments	(Comment String)	ASCIIコメント（Argument）ストリングとターミネータ
Header	1111…1111	16ビットダミー
		16ビット暗号化プリアンブル
30,000 Filler Bits		これによりデバイスが128ビット暗号化キーをロードし、ハッシュする時間を与える
Alignment Preamble		16ビットのアライメント・プリアンブル
	1	1ビットのダミーデータ
Data		ビットストリームがマスタ・プログラミング・モードに生成された場合、ダミー・フィラービットはない。復号化する際に時間が必要な場合、マスタデバイスがCCLKを停止する。そして暗号化されたデータを受け入れる準備が整うとクロックを再開する。
Program Done		32ビットのProgram Doneで、暗号化されている
End	1111….1111	32ビットのターミネータ（全て 1）。暗号化されている
Filler Bits		境界要件のためのフィラービット
Dummy Data	1111….1111	200ビット・ダミーデータ（全て 1）。復号化エンジンをオフにする遅延時間を与える

注: この表におけるデータは参照としてのみを意図しています。陰影をつけられた領域は暗号化されたデータです。

表16-5 スレーブシリアル・モードの暗号化ファイル形式

フレーム	内 容	記 述
Comments	(Comment String)	ASCIIコメント（Argument）ストリングとターミネータ
Header	1111…1111	2バイトダミー
		16ビット暗号化プリアンブル
30,000 Filler Bits		これによりデバイスが128ビット暗号化キーをロードし、ハッシュする時間を与える
Alignment Preamble		16ビットのアライメント・プリアンブル
	1	1ビットのダミーデータ
Data		128ビットのコンフィグレーション・データ
		64ビットデータ（全て 1）。復号化エンジンがちょうど受信した128ビットデータを復号する遅延を与える。データを中断している最中にペリフェラル・デバイスが必要な64クロックを供給できるなら64ビットのダミーデータは不要で、ファイルサイズを小さくできる。
	・・・	
		コンフィグレーション・データの最終フレームの最後の128ビット
		64ビットデータ（全て 1）。復号化エンジンがちょうど受信した128ビットデータを復号する遅延を与える。データを中断している最中にペリフェラル・デバイスが必要な64クロックを供給できるなら64ビットのダミーデータは不要で、ファイルサイズを小さくできる。
Program Done		32ビットのProgram Doneで、暗号化されている
End		32ビットのターミネータ（全て 1）。暗号化されている
Filler Bits		境界要件のためのフィラービット
Delay		64ビットデータ（全て 1）。Program Doneとフィラーを復号化するための遅延

Dummy Data	1111….1111	200ビット・ダミーデータ（全て 1）。復号化エンジンをオフにする遅延時間を与える

注；この表におけるデータは参照としてのみを意図しています。陰影をつけられた領域は暗号化されたデータです。

表16-6 スレーブパラレル・モードの暗号化ファイル形式

フレーム	内 容	記 述
Comments	(Comment String)	ASCIIコメント（Argument）ストリングとターミネータ
Header	1111…1111	2バイトダミー
		2バイト暗号化プリアンブル
30,000 Filler Bits		これによりデバイスが128ビット暗号化キーをロードし、ハッシュする時間を与える
Alignment Preamble		2バイトのアライメント・プリアンブル
	11111111	1バイトのダミーデータ
Data		16バイトのコンフィグレーション・データ
		64バイトデータ（全て 1）。復号化エンジンがちょうど受信した128ビットデータを復号する遅延を与える。データを中断している最中にペリフェラル・デバイスが必要な64クロックを供給できるなら64ビットのダミーデータは不要で、ファイルサイズを小さくできる。
	・・・	
		16バイトのコンフィグレーション・データ
		64バイトデータ（全て 1）。復号化エンジンがちょうど受信した128ビットデータを復号する遅延を与える。データを中断している最中にペリフェラル・デバイスが必要な64クロックを供給できるなら64バイトのダミーデータは不要で、ファイルサイズを小さくできる。
Program Done		4バイトのProgram Doneで、暗号化されている
End		4バイトのターミネータ（全て 1）。暗号化されている
Filler Bits		境界要件のためのフィラービット
Delay		64バイトデータ（全て 1）。Program Doneとフィラーを復号化するための遅延
Dummy Data	1111….1111	200バイト・ダミーデータ（全て 1）。復号化エンジンをオフにする遅延時間を与える

注；この表におけるデータは参照としてのみを意図しています。陰影をつけられた領域は暗号化されたデータです。

復号化フロー

ユーザの観点からは、ちょうど議論した暗号化フローと比べて、復号化フローははるかに簡単です。

データがFPGAに入力されると、デコーダはプリアンブル(図16-1を参照)探し始め、それ（プリアンブル）までのすべての情報を無視します。プリアンブルは制御レジスタ0 (Control Register 0)の圧縮ビットと共にコンフィグレーション・データのパスを決定します。

デコーダがビットストリームに標準のビットストリーム・プリアンブルを検出すると、それはこれが非暗号化されたデータファイルであることを知ります。そしてファイルが圧縮されてるかどうかを決定するために、デコーダはビットストリームの制御レジスタ0を調べます。ファイルが圧縮されていない場合、生の (Raw) データパスが選択されます(図16-1を参照)。ファイルが圧縮されている場合、Decompressedパスが選択されます。次にCRCがチェックされて、SRAMヒューズがプログラムされます。

デコーダがビットストリームに暗号化プリアンブルを検出すると、これが暗号化されたデータファイルであることを知ります。

暗号鍵がプログラムされていないなら、暗号化データはブロックされ、そしてコンフィグレーションは失敗します(DONEピンはLowのまま)。適切な暗号鍵をプログラムしてあるなら、コンフィグレーションは続行します。次のブロックリードは3万クロックのフィラーデータを含んでいます。この遅延はFPGAが鍵ヒューズを読んで、復号化エンジンを準備する時間を与えます。デコーダはアライメント・プリアンブルを探すためにフィラーデータを読み続けます。いったん見つけられると、それ以降のデータが復号化エンジンに送られる必要があります。そしてデコーダはファイルが圧縮されたかどうか決定するために復号した制御レジスタ0の内容を調べます。ファイルが圧縮されていない場合、Decrypted（復号化）データパスが使用されます。ファイルが圧縮されている場合、復号されたデータは伸張エンジンを通り抜けます。そして、Decompressed（伸張化）パスが選択されます(図16-1のブロック図を参照)。

次に、CRCがチェックされ、そしてビットストリーム・プリアンブルが読まれると、SRAMヒューズがプログラムされます。内部Doneビットがコンフィグレーションの終わりにセットされると、復号化と伸張エンジンはオフにされます。何か (チェインにおける他のデバイスへの) データオーバフローが起きると、チェイン下位のデバイスがコンフィグレーション記憶デバイスから生データを受信するように、これを行います。

しかし、FPGAの暗号鍵がファイルを暗号化するのに使用された暗号鍵に一致しない場合、何が起こるでしょうか? 一度データが復号されると、FPGAはCRCチェックをパスする適切なコマンドとデータと共に有効な標準ビットストリーム・プリアンブル(BDB3)を期待します。暗号鍵が一致しないと、復号化エンジンは適切なコンフィグレーション・ビットストリームを生成しません。プリアンブルが見つからないのでコンフィグレーションは始まらないでしょう (INITNピンはHighのままで、DONEピンはLowのまま)。或いは、CRCエラーが発生して、結果としてINITNピンがエラーを示すためにLowになります。INITNとDONEの詳しい情報に関してはテクニカルノートTN1108 (LatticeECP2/M sysCONFIG Usage Guide) を参照して下さい。

参考文献

・Lattice Technical Note TN1108, LatticeECP2/M sysCONFIG Usage Guide

・Federal Information Processing Standard Publication 197, Nov. 26, 2001.Advanced Encryption Standard (AES)

テクニカル・サポート支援

ホットライン: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
インターネット: http://www.latticesemi.com

変更履歴（日本語版）

Rev.#	日付	変更箇所
1.2J	Feb. 2009	日本語版新規発行
1.2J1	Feb. 2009	マイナータイポ修正