XAPP1223 (v1.0) 2014 年 11 月 6 日

Zynq-7000 AP SoC における暗号化キーの変更

著者 : Lester Sanders

概要

セキュア エンベデッド システムの暗号化キーは、セキュリティ レベルを向上させるために変更する必要があります。システム設計者は、まず最初に暗号化キーを変更すべきか否かなどの暗号化要件を定義する必要があります。暗号化要件の定義には、情報の重要度、エンベデッド システムの使用、エンベデッド システムに起こり得る脅威を考慮します。このアプリケーション ノートでは、エンベデッド システムの暗号化キー要件について簡単に説明します。リファレンス デザインを使用して、Zynq®-7000 All Programmable (AP) SoC デバイスで AES キーを変更する方法について解説します。

このアプリケーション ノートのリファレンス デザインは、ザイリンクスのウェブサイトからダウンロードできます。デザイン ファイルの詳細は、12 ページの「リファレンス デザイン」を参照してください。

含まれるシステム

リファレンス デザイン (zed_key_change) には、次のソフトウェア プロジェクトが含まれています。

- ファースト ステージ ブートローダー (FSBL)
- BBRAM キー ローダー (BKL)
- パーティション ローダー
- hello_world

bbram_key_loader_src および partition_loader_src ディレクトリにソース コードがあります。completed ディレクトリには、コンパイル済みのプロジェクトおよび zed_key_change.bif、zed_key_change.mcs、zed_bbram1.nky、zed_bbram2.nky ファイルが含まれています。

はじめに

AES (Advanced Encryption Standard) などの最新の暗号化方式の場合、暗号化アルゴリズムは公開されているため、システムのセキュリティは暗号化キーのセキュリティに依存します。Zynq-7000 AP SoC デバイスでは、AES キーをワンタイム プログラマブル (OTP) eFUSE アレイまたはバックアップ バッテリ付き RAM (BBRAM) のいずれかに格納できます。BBRAM を使用する場合は、暗号化キーを変更できます。キーを変更することで、攻撃者がキーを解読するために費やす時間が増大し、またキーで保護することで情報量を削減できます。

Zynq-7000 AP SoC デバイスより前のデバイスでは、AES 暗号化キーは iMPACT またはサードパーティのソフトウェアを使用して FPGA にプログラムされます (通常、システムが実際に現場で展開される前)。そして、デバイスが存続する限り同じ暗号化キーが使用されます。Zynq-7000 AP SoC デバイスの場合は、Zynq-7000 デバイス CPU で動作するセキュア キーで暗号化キーを変更できます。このアプリケーション ノートでは、異なるパーティションに対して BBRAM キーを変更する方法について説明します。このキーは、ブート時またはランタイム時に変更できます。

「暗号化キーの要件を定義」で、一般的なセキュリティについて説明し、Zynq-7000 AP SoC デバイスでキーを変更する理由を示します。リファレンス デザインは、Zed または MicroZed 評価プラットフォーム上で実行されます。JTAG ピンの外部接続の要件を含むボードのセットアップについては、3 ページの「評価ボード」で説明します。プロジェクト開発の概要は、4 ページの「リファレンス デザインの開発」で説明します。Bootgen を使用したソフトウェア プロジェクトの開発およびそれらを統合してイメージを生成するプロセスについて説明します。Zed 評価ボードにおけるキー変更のハードウェア検証については、11 ページの「システム テスト」で説明します。

注記 : Zed とは、Zynq 評価および開発を意味します [参照 1]。

このアプリケーションでは、ザイリンクスのソフトウェア開発キット (SDK) に関する知識があることを前提として説明しています。セキュア アプリケーションに SDK を使用するための入門ガイドは、『Zynq-7000 All Programmable SoC のセキュア ブート』(XAPP1175) [参照 2] を参照してください。

必要なハードウェアおよびソフトウェア

Zynq-7000 デバイスで暗号化キーを変更する際のハードウェアおよびソフトウェア要件は次のとおりです。

- Zed または MicroZed 評価ボード
- AC 電源アダプター (12VDC)
- USB Type-A/Micro-B ケーブル
- ザイリンクス プラットフォーム ケーブル USB II
- JTAG–MIO ワイヤ コネクタ (x 4)
- Vivado® Design Suite 2014.3
- ザイリンクス ソフトウェア開発キット (SDK) 2014.3

暗号化キーの要件を定義

エンベデッド システム設計者は、システムの暗号化キーの要件を定義する必要があり、キーの変更が必要か否かを判断します。暗号化キーの要件は、エンベデッド システムで使用される情報の重要度、エンベデッド システムの使用モデル、およびシステムへの脅威によって決定されます。

エンベデッド システムで使用される情報の重要度

Zynq-7000 AP SoC デバイスで使用されるパーティションは、ソフトウェア、データ、およびハードウェア (ビットストリーム) の 3 種類あります。ハードウェアやソフトウェアでは、競合性の高い IP が含まれる場合に重要と判断されます。データ パーティションでは、患者カルテなど機密情報として保護する必要がある情報が含まれる場合に重要と判断されます。

エンベデッド システムの使用モデル

エンベデッド システムの使用形態によって、攻撃のリスクや確率が左右されます。セキュリティ ガードで保護されたセキュリティ範囲内でエンベデッド システムが使用される場合、内部攻撃の対象となりますが、盗用や物理的な攻撃にさらされる可能性が低下します。デバイスが現場で使用され、有線/無線通信を介してインターネット接続される場合は、デバイスが盗用されたり、中間者攻撃でデータが傍受される可能性があります。フィールド エンベデッド システムがリモート ソフトウェア アップデートをサポートする場合は、通常、暗号化キーの変更が必要です。

エンベデッド システムへの脅威

エンベデッド システムへの脅威の例として、物理的攻撃とリモート ハッキングの 2 つを紹介します。本来、脅威は情報の重要度やエンベデッド システムの使用モデルと深く関係します。使用モデルを用いて、サイドチャネル DPA (差分電力解析) 攻撃などの手元のシステムを対象とした物理的攻撃、またはサービス妨害などを引き起こすインターネット経由のリモート攻撃を受け易いか否かを判断できます。

情報の重要度は、必要な保護レベルの判断に役立ちます。必要な保護レベルは、機密情報への不正アクセス、IP の盗用、あるいはシステムの機能障害 (サービス妨害) を引き起こす原因などによって異なります。

モバイル アプリケーションでエンベデッド システムが使用される場合は、デバイスが盗用されているかを所有者が認識できることがあります。手頃な価格でデバイスを置き換えることができ、暗号化キーをゼロ化すれば、物理的な攻撃を受けにくくなります。

デバイスのリモート ハッキングによる情報流出が脅威の対象となる場合には、キーを変更して攻撃から保護することが可能です。

暗号化キーを変更する理由

暗号化キーを変更する理由は次のとおりです。

- 最良のセキュリティ対策 ー クローズド システムの場合でも、定期的にキーを変更する必要があります。
- オープン システム ー エンベデッド システムのソフトウェア サプライヤーはそれぞれ異なる暗号化キーを使用する必要があります。
- 連結システム ー エンベデッド システムは、有線および無線イーサネットを使用して連結されます。連結されると、優れたシステムになりますが、リモート ハッカーへアクセス ポイントを提供することになります。
- 複数の非公開のピア間通信 ー 複数のリモート エンティティへ機密情報を送信する場合は、異なる暗号化キーを使用する必要があります。

評価ボード

リファレンス デザインは、ザイリンクスと Avnet 社の評価ボードを使用します。ザイリンクスの Zynq-7000 AP SoC デバイス評価プラットフォームは ZC702 と ZC706 の 2 つで、Avnet 社の Zynq-7000 デバイス評価プラットフォームは、Zed と MicroZed ボードの 2 つです。早期にリリースされた Zed ボードは、セキュリティをサポートしていません。ZC702 と ZC706 ボードには、BBRAM 用のバッテリが実装されています。

ボードのセットアップ要件は、GPIO ピンから JTAG ポートへ外部接続を行うことです。この接続は、最新のボード上で簡単に定義および実装できます。リファレンス デザインでは、安価な Avnet 社製ボードを使用します。

図 1 に Zed ボードのセットアップを示します。JTAG ポートは J15 コネクタへ接続します。JTAG TMS、TCK、TDO、TDI ポートは、ボード上に表記されています。MIO ポートは、JE1 コネクタへ接続します。Zed および MicroZed ボード上の MIO (GPIO) と JTAG の接続は比較的単純です。

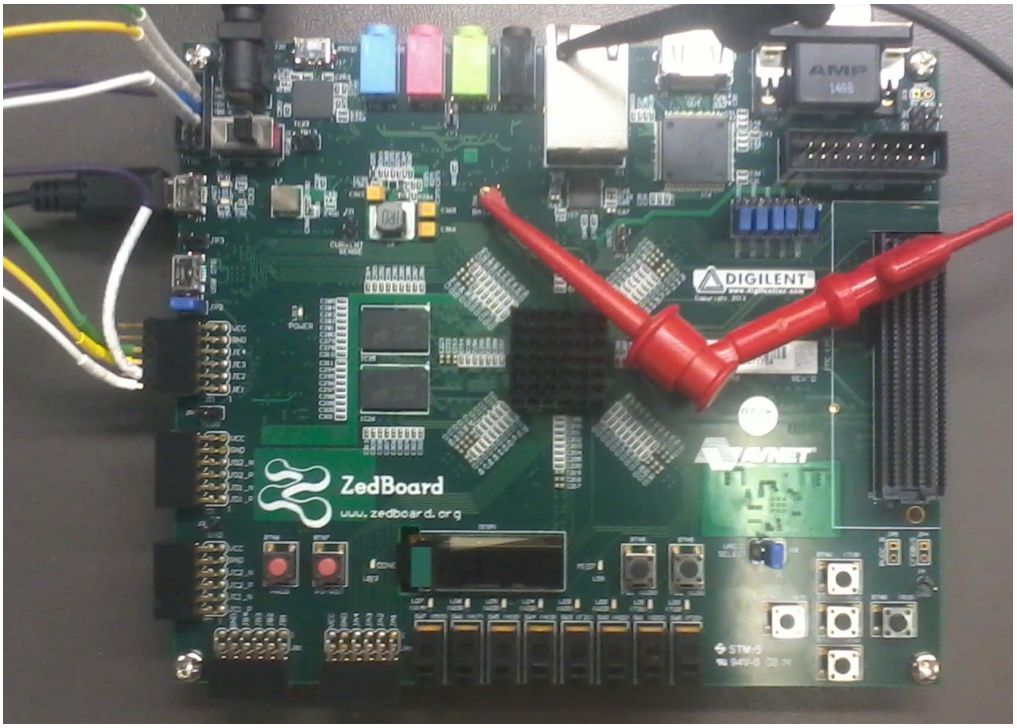

図 1 : Zed ボード上における MIO と JTAG の接続

Zed および MicroZed ボードは、BBRAM 用のバッテリを備えていません。プロトタイプのテストでは、BBRAM のバッテリの代わりに電源を使用できます。Zed ボードでは、電源を J16 の BATT/GND 端子に接続します。MicroZed ボードでは、電源を S12 に接続します。図 1 は、リファレンス デザインを使用した MicroZed の接続の写真です。

評価ボードを使用してプロトタイプ化した後、ボード トレースを用いて GPIO と JTAG ポートを接続したカスタム ボードを開発できます。

リファレンス デザインの開発

BBRAM キーを変更するシステム開発では、4 つのソフトウェア プロジェクト (FSBL (ファースト ステージ ブートローダー)、BBRAM キー ローダー (BKL)、パーティション ローダー、hello_world) を使用します。リファレンス デザインでは、FSBL、BBRAM キー ローダー、およびパーティション ローダーがオリジナルの暗号化キーを使用し、hello_world パーティションが 2 つ目の暗号化キーを使用します。

このアプリケーション ノートでは、hello_world パーティションのロード/復号化の前に暗号化キー ロード ソフトウェアが実行されるように、デバイス上でロード/実行の順序を変更する方法について説明します。これは、BBRAM キー ローダーとパーティション ローダーを使用して実行します。

キーの変更には、パーティションのロード/実行の順序を変更する必要があるため、パーティションをロードする標準フローを変更しなければなりません。変更されない場合は、FSBL がすべてのパーティションをロードして、ロードされた最後のパーティションに実行を移します。すでに述べたように、hello_world パーティションは新しい暗号化キーを使用して暗号化されています。BBRAM キー ローダーがロードされても実行されなければ、hello_world パーティションの復号化で使用される BBRAM キーがアップデートされません。

図 2 に、エンベデッド システムをブートする際の標準フローを示します。

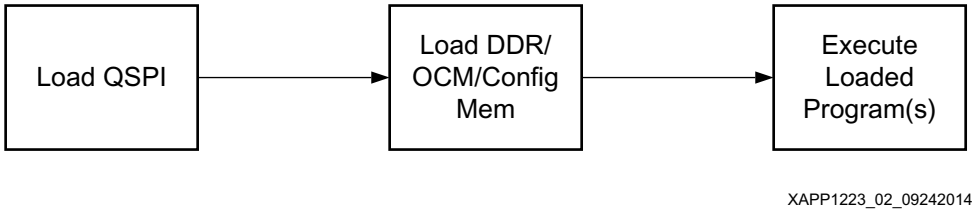

図 2 : システム開発の標準フロー

図 3 に、暗号化キーが変更されるリファレンス デザインのフローを示します。新しいキーで暗号化されたパーティションを復号化できるようにパーティションのロード/実行の順序が変更されています。

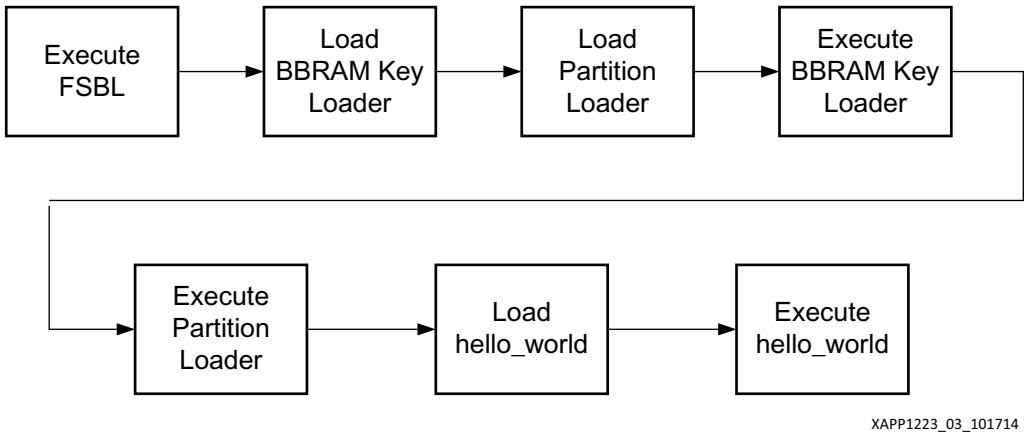

図 3 : 暗号化キーを変更する場合のロード/実行の順序

リファレンス デザインでは、次のタスクを使用して AES キーを変更します。

- ブロック RAM キー ローダーおよびパーティション ローダー ソフトウェア プロジェクト
- JTAG ポートの有効化/無効化
- すべてのソフトウェア プロジェクトを実行するためのハンドオフ コード
- Bootgen イメージ フォーマット (BIF) ファイルの変更

アドレス マップと実行ハンドオフ

リファレンス デザインでは、FSBL、BBRAM キー ローダー、パーティション ローダー、および hello_world が連続して実行される必要があります。標準的な FSBL フローでは、最後のパーティションのみ実行されます。1 つのソフトウェア パーティションから次のパーティションへ実行を移すために、ソフトウェア プロジェクトには HandoffExit 関数が含まれます。HandoffExit 関数は、次に実行するパーティションのロード アドレスを使用します。表 1 に、リファレンス デザインで使用されるソフトウェア プロジェクトのアドレス マップを示します。

表 1 : アドレス マップ

ソフトウェア	アドレス
ブロック RAM キー ローダー	0xFFFFC000
パーティション ローダー	0x200000
hello_world	0x300000

このアドレス マップ、HandoffExit 関数、および BIF partition_owner 属性 (後で説明) を使用して、実行順序は FSBL → BBRAM キー ローダー → パーティション ローダー → hello_world となります。

FSBL (ファースト ステージ ブートローダー) と Hello_World プロジェクト

SDK の新しいプロジェクト テンプレートを使用して、FSBL と hello_world プロジェクトをコンパイルします。hello_world プロジェクトは、変更なしでコンパイルされます。最初のコンパイルの後に FSBL プロジェクトにわずかな変更が必要です。

このリファレンス デザインはセキュア モードで実行されるため、JTAG ポートは無効となります。BBRAM キー ローダーで使用されるセキュア キー ドライバーは JTAG ポートを使用します。JTAG ポートを有効にする場合、`fsbl_src/main.c` コードを変更します。DAP/JTAG ポートを有効にするには、`main.c` 内のコードに CTRL(23) を書き込みます。

標準的な FSBL は、FSBL の直後にビットストリームがパーティションとしてロードされるようにロード順序を強制的に実行します。システムによっては、FSBL コードを変更してこの要件を緩和する必要があります。

1. コマンド プロンプトに「xsdk &」と入力します。
2. [Create Application Project] をクリックします。
3. [New Project Templates] ダイアログ ボックスで [Project Name] に「fsbl」と入力して [Hardware Platform] を選択して、[Next] をクリックします。
4. [New Project Templates] で [Zynq FSBL] を選択して [Finish] をクリックします。リファレンス デザインの `fsbl/src` で、`main.c` を fsbl_src/main.c に置き換えます。
5. [Project Explorer] ペインで、[**fsbl**] を右クリックして [**C/C++** Build Settings] → [Symbols] → [Defined Symbols +] をクリックします。テキスト ボックスに「FSBL_DEBUG_INFO 」と入力して、[**OK**] をクリックします。
6. これらの手順を繰り返して、hello_world ソフトウェア プロジェクトを作成します。hello_world ソフトウェア プロジェクトのコード変更は不要です。
7. [Project Explorer] ペインで、[hello_world] をダブルクリックして [src] → [lscript.ld] をクリックし、[ps7_ddr_0_S_AXI_BASEADDR] のベース アドレスに「**`0x00300000`**」と入力します。

BBRAM キー ローダー

BBRAM キー ローダーは、SDK の `lib/sw_services` ディレクトリのセキュア キー ドライバー (SKD) ライブラリを使用します。この SKD を用いて、eFUSE および BBRAM キーをプログラムできます。以前は、iMPACT または BPM Microsystems 社製デバイス プログラマを使用して暗号化キーをプログラムしていました。

BBRAM キー ローダー ソフトウェア プロジェクトを作成するには、まず次のコマンドを使用して、BBRAM キー ローダー ボード サポート パッケージを作成します。

1. [File] → [New] → [Board Support Package] をクリックします。
2. [Project Name] に「bbram_key_loader_bsp_0」と入力します。[Hardware Platform] で [zed_hw_platform] を選択して、[Finish] をクリックします。
3. [Board Support Package Settings] ウィンドウで [xilskey] をオンにして、[Finish] をクリックします。
4. [File] → [New] → [Application Project] をクリックします。
5. [Project Name] に「bbram_key_loader」と入力して、[Hardware Platform] で [zed_hw_platform] を選択します。[Use Existing] をオンにして [bbbram_key_loader_bsp_0] BSP を選択して、[Next] をクリックします。
6. [New Project Templates] で [Empty Application] を選択します。
7. [Project Explorer] ペインで [bbram_key_loader] を右クリックして [src] をクリックし、[Import] → [General] → [File System] をクリックして、[Next] をクリックします。
8. [Browse] で、リファレンス デザインの xilskey_v2_0/examples ディレクトリを選択して、[OK] をクリックします。
9. `xilskey_bbram_example.c`、`xilskey_input.h`、および `handoff.S` ファイルを選択します。

   xilskey ソース ファイルも SDK のインストール エリア (`lib/sw_services` の下) にあります。xilskey_v2_0/examples ディレクトリのリファレンス デザインのコピーには、`xilskey_input.h` にサンプルの AES キーが追加され、

xilskey_bbram_example.c にハンドオフ コードが追加されています。ソース ファイルを追加した後、[Finish] をクリックします。

10. [Project Explorer] ペインで、[bbram_key_loader] をダブルクリックして [src] → [lscript.ld] をクリックし、[ps7_ram_0_S_AXI_BASEADDR] のベース アドレスに「0xFFFFC000 」と入力します。

表 2 に、ZC702 および Zed/MicroZed ボードで使用される接続を示します。

表 2 : MIO と JTAG の接続

ピン名	ZC702 MIO	Zed MIO	MicroZed MIO
TDI	17	13	13
TDO	18	10	10
TCK	19	11	11
TMS	20	12	12

MIO と JTAG の接続は、xilskey_input.h ファイルに定義されています。

セキュア キー ドライバーのドキュメントは次のディレクトリにあります。

$ Xilinx/../SDK/2014.3/data/embeddedsw/lib/sw_services/xilskey_v2_0/doc

パーティション ローダー

パーティション ローダーは、ソース アドレスからデスティネーション アドレスへシングル パーティションをコピーする小さなコード セットです。パーティション ローダーは、Zynq-7000 AP SoC 上の 2 つのハードウェア機能 (デバイス コンフィギュレーション ダイレクト メモリ アクセス コントローラー (DevC DMAC) および AES Decryptor) に対して簡単にアクセスできるようにインターフェイスを提供します。devcfg デバイス ドライバーは、SDK/../data/embeddedsw/XilinxProcessorIPLib/drivers/devcfg_v3_1 ディレクトリにあります。パーティション ローダーは、この devcfg デバイス ドライバーの関数を使用して、DevC DMAC や AES Decryptor を制御します。コピーされるパーティションがソフトウェアまたはデータのいずれかの場合は、PcapDataTransfer 関数が使用されます。コピーされるパーティションがハードウェア (ビットストリーム) の場合は、PcapLoadPartition 関数が使用されます。

パーティション ローダーは、5 つの引数 (ソース アドレス、デスティネーション アドレス、ソース サイズ、デスティネーション サイズ、およびコピー時に復号化機能を介すか否かを指定する引数) を使用します。ソース アドレスは、パーティションの Bootgen で生成された Quad SPI (Quad Serial Peripheral Interface) フラッシュ メモリのアドレスです。このアドレスは、SDK Flash Writer によって書き込みが行われます。デスティネーション アドレスは、Zynq-7000 AP SoC デバイス内の DDR、オンチップ メモリ (OCM)、またはコンフィギュレーション メモリ内のアドレスです。暗号化が使用される場合、パーティションのソースの長さはデスティネーションの長さより長くなります。

パーティションのソースおよびデスティネーションのアドレス/サイズを決定するには次の 2 つの方法があります。[FSBL_DEBUG_INFO] シンボルがコンパイルのオプションである場合は、FSBL デバッグ ログ ファイルにアドレスおよびサイズ情報が含まれます。

図 4 の FSBL デバッグ ログ ファイルの最後の部分に、bbram_key_loader パーティションのソースおよびデスティネーション アドレスとサイズを示しています。

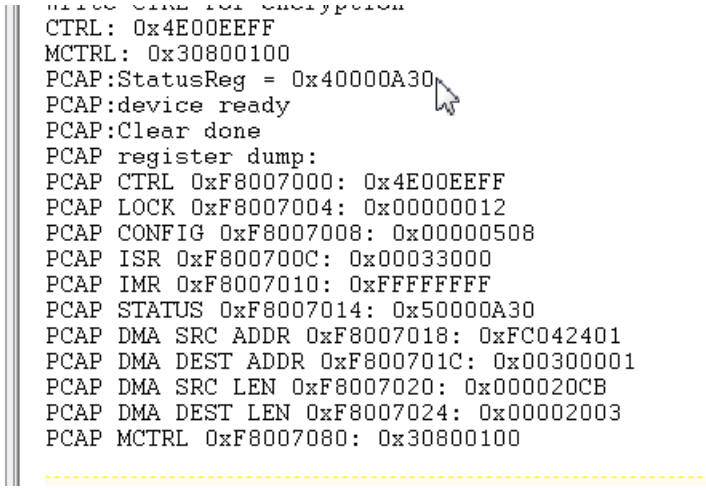

図 4 : bbram_key_loader パーティションのソースおよびデスティネーション アドレスを示すログ ファイル

パーティションのアドレスおよびサイズは、`-debug` コマンド オプションを使用して Bootgen を実行した場合にも提供されます。

パーティション ローダーの実行中に DevC DMAC が停止した場合は、サイズ (SRC LEN、DEST LEN) が適切であるかを確認してください。パーティションのサイズ情報は提供されますが、ない場合はバイト数またはワード数という単位で必ず必要です。ワードとバイト間の変換には、次の python コードを使用します。

```
int("f6d20",16)
1010976
1010976 * 4
4043904
hex(4043904)
0x3db480
```

リファレンス デザインでは、パーティション ローダーが AES 復号化機能を介して hello_world ソフトウェアを Quad SPI フラッシュ メモリから DDR へコピーします。

次の手順に従って、空のアプリケーションとしてパーティション ローダー ソフトウェア プロジェクトを作成します。

1. [Project Explore] ペインで [**partition_loader**] を右クリックして、[**src**] → [**Import**] をクリックします。
2. [Import] ダイアログ ボックスで [**General**] → [**File System**] を選択して [Next] をクリックします。
3. [Browse] で、リファレンス デザインの `partition_loader_src` ディレクトリを選択して、[OK] をクリックします。
4. [Select All] をクリックして、[Finish] をクリックします。
5. [Project Explore] ペインで [partition_loader] をダブルクリックして、[src] → [lscript.ld] をクリックします。
6. [ps_ddr_0_S_AXI_BASEADDR] のベース アドレスとして「0x00200000」と入力します。

SDK Flash Writer のロードの後、`bbram_key_loader` パーティションが暗号化されて Quad SPI フラッシュ メモリ内に格納されます。bbram_key_loader パーティションを DDR へ復号化/コピーする標準的な FSBL 動作は、暗号化キーが `xilskey_input.h` 内に含まれているため使用できません。パーティション ローダーは、復号化機能を介さずに `bbram_key_loader` パーティションを DDR へコピーします。BRAM キー ローダーを使用する際に、パーティションは復号化されて OCM へコピーされます。パーティション ローダーは、OCM から実行されます。

パーティション ローダーは、パーティションのコピーや復号化のほかにも、暗号化された暗号化キーを Zynq-7000 デバイスの OCM 内に格納して、使用直前にキーを復号化するタスクもサポートしています。

Bootgen

Bootgen は、パーティションを統合して BIN または MCS フォーマットでイメージを生成する SDK ツールです。Bootgen を使用するには、Bootgen GUI またはテキスト エディターを使用して Bootgen イメージ フォーマット (BIF) ファイルを作成します。BIF ファイルは、イメージ内に含まれるパーティションのリストです。リスト内の各パーティションに対して、BIF ファイルはオプションでロード アドレス、暗号化、認証を指定するパーティション属性を含めることができます。partition_owner 属性は、FSBL がパーティションをロードするか否かを制御します。partition_owner 属性は、U-Boot などのセカンド ローダー (リファレンス デザインではパーティション ローダー) がパーティションをロードする場合に使用されます。

リファレンス デザインを用いて、ロード/実行の順序要件が満たされたイメージを生成する方法を示します。

Bootgen イメージ フォーマット ファイル

現在、Bootgen GUI では複数の暗号化キーを指定することができません。リファレンス デザイン用の BIF を生成するには、Bootgen GUI を使用してテンプレート BIF を生成した後、テキスト エディターを使用して BIF 内のキーを定義します。

1. SDK で [Tools] → [Create Zynq Boot Image] をクリックします。
2. [**Use encryption**] をオンにします。
3. [Key File] で [`zed_bbram1.nky`] を参照 (リファレンス デザインのキー ディレクトリにある) して、[Add] をクリックします。
4. [File path] で `fsbl.elf` を参照します。
5. [Partition type] に [bootloader] を選択します。[Encryption] に [aes] を選択して、[OK] をクリックします。
6. `fsbl.elf` に対して [Add] の手順を繰り返して `bbbram_key_loader.elf`、`partition_loader.elf`、および `hello_world.elf` を追加します。
7. [Partition type] を [datafile] に変更します。
8. [Browse] を使用して、[Output BIF file path:] に [`zed_key_change.bif`] を指定します。
9. [Browse] を使用して、[Output path:] に [`zed_key_change.mcs`] を指定します。
10. [Create Image] をクリックします。

図 5 に、4 つのパーティションが追加された Bootgen GUI インターフェイスを示します。

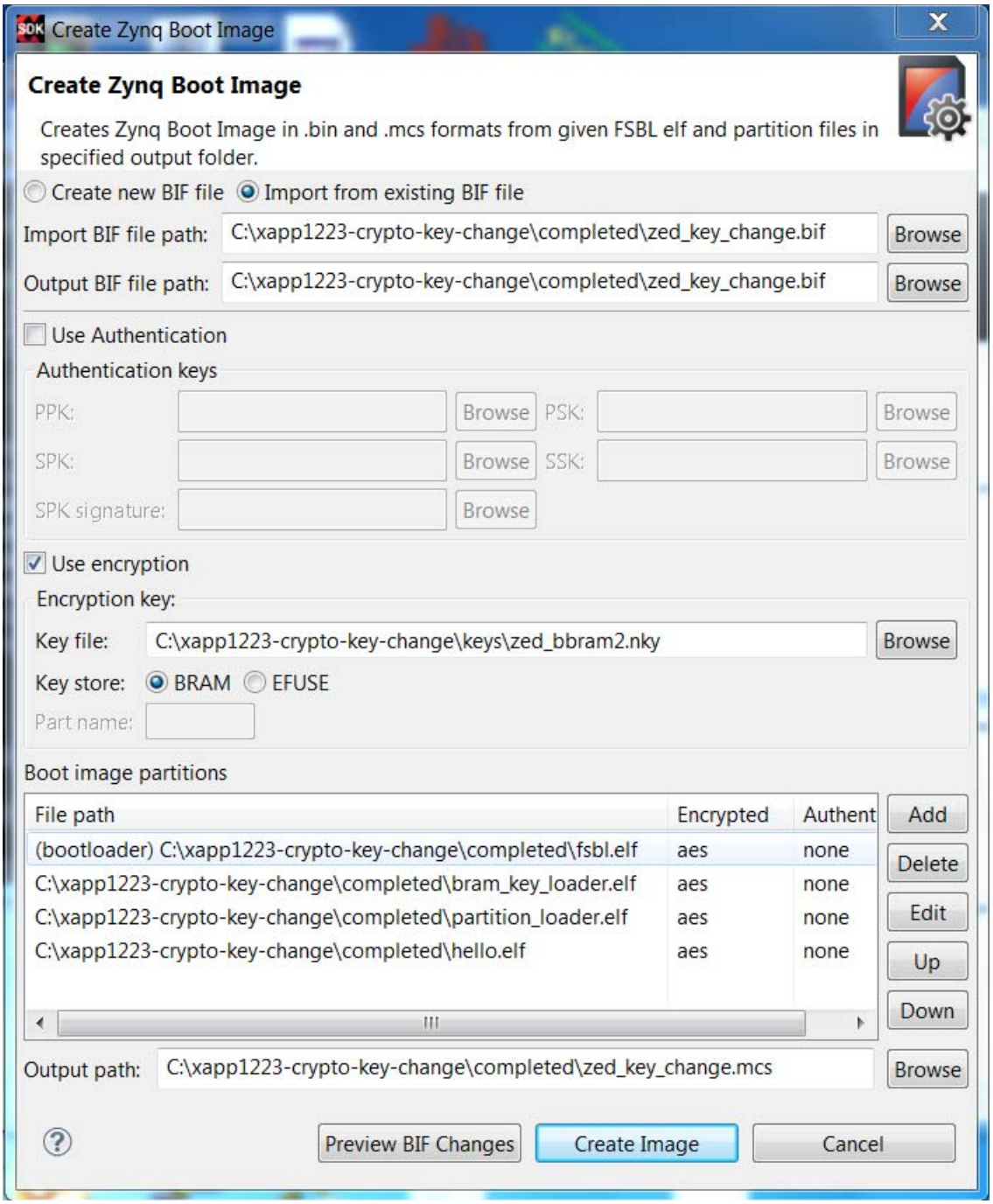

図 5 : Bootgen GUI を使用した BIF 生成

11. テキスト エディターを使用して、2 つ目の暗号化キー (`zed_bbram2.nky`) を Bootgen GUI で生成した BIF に追加します。partition_owner 属性を hello_world パーティションに追加します。最終的な BIF は、completed ディレクトリの zed_key_change.bif に配置されます。
12. コマンド プロンプトで、次のコマンドを実行して MCS ファイルを生成します。

    bootgen -image zed_change_key.bif -o zed_change_key.mcs -encrypt bbram -w on

    または、SDK を使用して、手作業で編集した `zed_change_key.bif` を Bootgen GUI に読み込んで、GUI を使用して MCS ファイルを生成することも可能です。

システム テスト

暗号化キーの変更をテストするには、SDK Flash Writer を使用して Quad SPI フラッシュ メモリに zed_key_change.mcs を書き込みます。QSPI へ書き込みできない場合は、Zed/MicroZed ボードのスイッチを JTAG プログラミング モードに設定してこの動作を行います。すべてのパーティションは、Quad SPI フラッシュ メモリにロードされます。

1. SDK で [Xilinx Tools] → [Program Flash] をクリックします。
2. [Program Flash Memory] ダイアログ ボックスで、[Browse] を使用してイメージ ファイルとして zed_key_change.mcs を選択します。
3. [Offset] で、「0x0」を指定します。
4. [Program] をクリックします。
5. Zed ボードを使用し、バッテリが追加されない場合は、J16 BATT/GND 端子に電源を接続します。iMPACT を使用して、最初の zed_bbram1.nky キーを BBRAM にプログラムします。

   impact -batch xapp1223-crypto-key-change/keys/bbram1_key_load.cmd

6. 図 1 および表 2 に定義されているとおりに、Platform Cable USB II を JE1 コネクタに置き換えます。
7. Quad SPI フラッシュ メモリへのロードが完了した後、ボードの電源を切断し、Mode Select ピンを QSPI に切り換えてボードに電源を投入します。TeraTerm などの通信端末を起動します。ボード上の PS_RST (RST) ボタンを押します。

   図 6 に、予想されるデバッグ ログ出力の最後の部分を示します。

```
CTRL: 0x4E00EEFF
MCTRL: 0x30800100
PCAP:StatusReg = 0x40000A30
PCAP:device ready
PCAP:Clear done
PCAP register dump:
PCAP CTRL 0xF8007000: 0x4E00EEFF
PCAP LOCK 0xF8007004: 0x00000012
PCAP CONFIG 0xF8007008: 0x00000508
PCAP ISR 0xF800700C: 0x00033000
PCAP IMR 0xF8007010: 0xFFFFFFFF
PCAP STATUS 0xF8007014: 0x50000A30
PCAP DMA SRC ADDR 0xF8007018: 0xFC042401
PCAP DMA DEST ADDR 0xF800701C: 0x00300001
PCAP DMA SRC LEN 0xF8007020: 0x000020CB
PCAP DMA DEST LEN 0xF8007024: 0x00002003
PCAP MCTRL 0xF8007080: 0x30800100


Handoff to hello
Hello World
```

図 6 : BBRAM キー変更のログ出力

8. リファレンス デザインのログを使用して、BBRAM キー ローダーが BBRAM に書き込みを行い、パーティション ローダーへ実行をハンドオフすることを検証します。パーティション ローダーが AES 復号化機能を介して hello_world パーティションを DDR へコピーした後 hello_world へ実行をハンドオフし、通信端末に「Hello World」が書き込まれることを検証します。

インプリメンテーションの注意事項

このアプリケーション ノートでは、暗号化キーを変更する方法について説明しています。実際のシステムでは、このセクションで言及すること以外にも考慮すべき事項があります。

Zynq-7000 AP SoC デバイスにおける暗号化キーの変更は、エンベデッド システムへ不正にアクセスして MIO/EMIO から JTAG ポート間の接続で信号を傍受する攻撃者に対して、わずかな防御策に過ぎません。キーを保護するには、外部の JTAG 接続を保護することが不可欠です。改ざん対策については、『Virtex-6 および 7 シリーズ FPGA での不正操作防止デザインの開発』(XAPP1084) [参照 3] を参照してください。

Zynq-7000 AP SoC デバイスの場合、BBRAM の書き込みトランザクションはプログラマブル ロジック (FPGA) セクションを消去してしまうため、ビットストリームの再度書き込みが必要です。既に述べたとおり、パーティションの書き込みにはパーティション ローダーを使用します。

リファレンス デザインでは、2 つの暗号化キーを使用します。この方法では、任意のキーの数に拡張可能です。任意数の BBRAM キー ローダー /パーティション ローダーを BIF に含めることができます。

リファレンス デザインでは、ランタイム中にハンドオフが実行されるため、ブート プロセスとランタイム プロセスの両方を使用します。

パーティション ローダーを使用する場合、リモートでリキーイングできます。このアプリケーション ノートでは、キーの交換については言及していません。リモートのエンティティ (ピア、ホスト) へアクセスするには、暗号化キーを交換する必要があり、それを暗号化、また符号化する必要があります。Bootgen はシングル パーティションの暗号化/符号化をサポートしています。256 ビットの暗号化キーはデータ パーティションです。格納されるキーは、Zynq-7000 デバイスのセキュリティ領域内にある NVM、DDR、またはオンチップ メモリ内で暗号化され、キーが必要なときにパーティション ローダーがコピー /復号化します。

一般にセキュア キー ドライバーが `xilskey_input.h` ファイルにハードコードされている暗号化キーを BBRAM へ書き込みます。一部のアプリケーションでは、OCM または AXI ブロック RAM のメモリ アドレス内にキーが暗号化されて格納されます。`xilskey_input.h` 内のハードコードされたキーではなく、キー用のメモリ アドレスを使用する場合には、セキュア キー ドライバーにわずかなコード変更が必要です。

BBRAM キーは、Zynq-7000 デバイスのセキュリティ領域内に格納されるため安全です。キーが不要な場合は、`xilskey_input.h` でキーの値に 0 を使用して、セキュリティ オプションをゼロ化します。ゼロ化は、BBRAM キー ローダー /パーティション ローダーのペアを使用して実行できます。暗号化キーがロードされた後、CTRL(23) の書き込みを実行して `xilskey_bram_example.c` で JTAG ポートを無効にします。

暗号化キー要件の指定では、デバイスのライフサイクルを考慮してキー要件を定義する必要があります。ブート時にロードされるパーティションで使用される暗号化キーは、ブート時に BBRAM 内に存在している必要があります。

注記 : BBRAM キー ローダー /パーティション ローダー操作を実行してランタイム中に暗号化キーを変更した後は、電源投入時に使用される暗号化キーをリロードする必要があります。

まとめ

このアプリケーション ノートでは、Zynq-7000 All Programmable SoC で BBRAM キーを変更する方法について説明しました。リファレンス デザインで使用されるテクニックを利用して、必要に応じて複数回キーを変更できます。キーは、ソフトウェア、データ、またはハードウェアのパーティションに対して変更できます。BBRAM キー ローダーとパーティション ローダーを使用して、それぞれのキー要件に対してキーを変更できます。

リファレンス デザイン

このアプリケーション ノートのリファレンス デザインは、ザイリンクスのウェブサイトからダウンロードできます。

表 3 に、リファレンス デザインの詳細を示します。

表 3 : リファレンス デザインの詳細

パラメーター	説明
全般	
開発者	Lester Sanders

表 3 : リファレンス デザインの詳細 (続き)

パラメーター	説明
ターゲット デバイス	Zynq-7000 AP SoC
ソース コードの提供	あり
ソース コードの形式	VHDL、Verilog
既存のザイリンクス アプリケーション ノート/リファレンス デザイン、サードパーティからデザインへのコード/IP の使用	なし
シミュレーション	
論理シミュレーションの実施	なし
タイミング シミュレーションの実施	N/A
論理シミュレーションおよびタイミング シミュレーションでのテストベンチの利用	N/A
テストベンチの形式	Verilog
使用したシミュレータ/バージョン	Vivado シミュレータ
SPICE/IBIS シミュレーションの実施	N/A
インプリメンテーション	
使用した合成ツール/バージョン	Vivado 合成
使用したインプリメンテーション ツール/バージョン	Vivado インプリメンテーション
スタティック タイミング解析の実施	あり
ハードウェア検証	
ハードウェア検証の実施	あり
使用したハードウェア プラットフォーム	Zed ボード

参考資料

1. 『ZedBoard スタートアップ ガイド』
2. 『Zynq-7000 All Programmable SoC のセキュア ブート』(XAPP1175)
3. 『Virtex-6 および 7 シリーズ FPGA での不正操作防止デザインの開発』(XAPP1084)
4. 『ZedBoard Configuration and Booting Guide』zedboard.org/support/design/1521/11
5. 『ZedBoard Hardware User's Guide』zedboard.org/sites/default/files/documentations/ZedBoard_HW_UG_v2_2.pdf
6. 『MicroZed and System on Module Hardware User Guide』 zedboard.org/sites/default/files/documentations/MicroZed_HW_UG_v1_5.pdf

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
2014 年 11 月 6 日	1.0	初版

法的通知