XAPP1224 (v1.0) 2014 年 10 月 22 日

Zynq-7000 AP SoC システムのセキュア アップデート

著者 : Lester Sanders

概要

Zynq®-7000 All Programmable (AP) SoC ベースのエンベデッド システムは、インターネットに接続すると、動作中にソフトウェア、ハードウェア、およびデータ ファイルをアップデートできます。インターネット経由でアップデートされるパーティションは、安全性を確保するために保護される必要があります。このアプリケーション ノートでは、Zynq-7000 AP SoC が動作している状態でエンベデッド システムを安全にアップデートする方法について説明します。

含まれるシステム

リファレンス システム (system_update) には、次のソフトウェア プロジェクトが含まれています。

- fsbl
- hello
- partition_loader
- u-boot
- hello_update

これらのソフトウェア プロジェクトの ELF ファイル、暗号キー、および Bootgen イメージ フォーマット (BIF) ファイルは、`system_update/completed` ディレクトリにあります。ソース ファイルは、`hello_src` および `partition_loader_src` ディレクトリにあります。

はじめに

SoC または FPGA ベースのエンベデッド システムを再プログラムする場合の一般的な方法は次のとおりです。

1. エンベデッド システムの電源を切断する。
2. ザイリンクス ソフトウェアがロードされる PC へシステムを物理的に接続する。
3. すべてのパーティションを含むイメージを不揮発性メモリ (NVM) へダウンロードする。
4. 電源を投入する。

この間、エンベデッド システムは利用できません。インターネットに接続すると、Zynq-7000 デバイスでは動作中にパーティション ローダーまたは U-Boot を使用して、ソフトウェア、ハードウェア、またはデータをロードできます。リファレンス システムを用いて、パーティション ローダーと U-Boot を使用してシステムをアップデートする方法を示します。

このアプリケーション ノートは、次のセクションで構成されます。

- 「システム アップデートの利点」では、エンベデッド システムをリモートからアップデートする理由を挙げています。
- 「システム アップデートの高位記述」では、システムをアップデートするために必要なタスクを紹介しています。これらのタスクは、ファクトリで実行されるタスクと、フィールドでエンベデッド システムに対して実行されるタスクに分かれています。
- 「システム アップデートの実行」では、ザイリンクスのソフトウェア開発キット (SDK) を使用してソフトウェア プロジェクトを構築します。その後、SDK でイメージを作成およびプログラムします。SDK を使用したソフトウェア プ

ロジェクトの作成に不慣れな場合は、システム作成の手順を説明している『Zynq-7000 All Programmable SoC のセキュア ブート』(XAPP1175) [参照 1] を参照してください。このセクションでは、パーティション ローダーと U-Boot を使用してシステムをアップデートする方法についても説明しています。

- 「システム アップデートのテスト」では、リファレンス システムの実行方法および結果の確認方法について説明しています。システムを安全にアップデートするには、AES 暗号および RSA 認証を使用します。
- 「システム アップデートのセキュリティ」では、機密性および完全性を確保したアップデート方法について説明しています。

必要なハードウェアおよびソフトウェア

Zynq AP SoC のソフトウェア アップデートにおけるハードウェアおよびソフトウェアの要件は次のとおりです。

- ZC702、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
- ザイリンクス ソフトウェア開発キット 2014.3
- ザイリンクス Vivado® Design Suite 2014.3

システム アップデートの利点

システム アップデートがもたらす主な利点は次のとおりです。

- 開発サイクルを短縮 ：新機能を簡単に追加できるため、最初のリリースにすべての機能を含める必要がない
- エラー管理 ：パッチでエラーを修正できる
- ソフトウェアまたはハードウェアをアップデートすることで、シリコンのライフ サイクルが延長される
- 患者カルテなどの機密データ記録を確実かつ安全に管理できる
- ハードウェアまたはソフトウェア動作を時分割多重化することによって、シリコンのシステム機能を向上させる
- 消費電力をリモート制御する
- 安全への侵害が発生した場合、または従業員が退職した場合にリモートから情報を削除できる
- 最新のソフトウェア/ハードウェアを備えることによって、エンベデッド システムのライフ サイクルが延長される

システム アップデートの高位記述

これまで、フィールド プログラマブル ゲート アレイ (FPGA) は設計者がプログラムするもので、エンド ユーザーが機能を変更することはありませんでした。Zynq-7000 AP SoC デバイスは、ユーザー主導のシステム アップデートをサポートしているため、製品の機能性が向上します。プロセッシング システム (PS) とプログラマブル ロジック (PL) の両セクションがプログラムできます。従来のデバイス ドライバーを使用して PS と PL の両方をプログラムできるため、ユーザー主導のアップデートが可能です。

ユーザー主導型アップデートのほかにも、サード パーティ ベンダーから、多くのエンベデッド システム アップデートに対応するデバイス マネージメント ソフトウェアが提供されています。これらのベンダーは、OMA DM (Open Mobile Alliance - Device Management) 規格への高位インターフェイスを提供し、フィールドに展開された多数のエンベデッド システムの効率的で安全なアップデートをサポートしています。

このアプリケーション ノートで使用するシステム アップデート方法は、ユーザー主導とデバイス マネージャー主導のシステム アップデートをサポートします。

システム アップデートには、有線または無線いずれかのインターネット接続を使用しますが、無線接続の方が効率的です。1 回のシステム アップデートで、1 つのパーティションがアップデートされます。パーティションとは、ソフトウェア (ELF/BIN)、ハードウェア (ビットストリーム)、またはデータ ファイルのいずれかです。アップデートは、セキュア モードまたは非セキュア モードで実行可能です。図 1 に、システム アップデートの概要を示します。ここでいうファクトリとは、ソフトウェアを開発してフィールド展開されたエンベデッド システムに分配する主要 サイトのことです。ファクトリを使用すると、デバイス マネージャーは開発の構造から独立して動作できます。ファクトリはパーティションを作成します。このパーティションは、ファクトリまたはデバイス マネージャーによってエンベデッド システムへ送信されます。通常、パーティションの最初のデスティネーションは、エンベデッド システムの DDR メモリです。

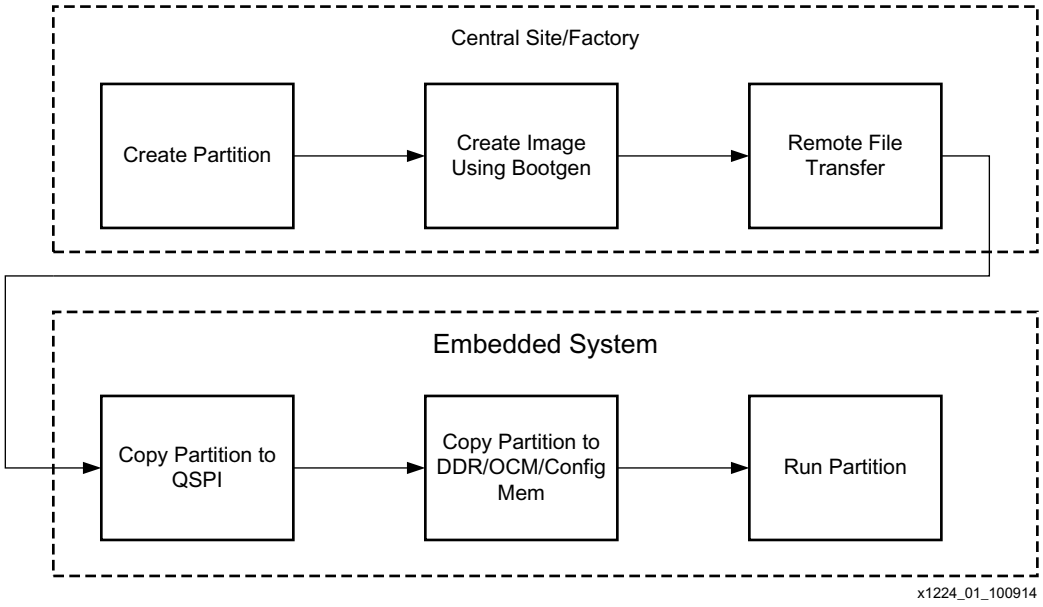


図 1 : システム アップデートのフロー図

ここでは、ファクトリからエンベデッド システムへのパーティションの伝送については詳しく説明しません。通常、この伝送には有線または無線イーサネットを使用します。

パーティションがエンベデッド システムへ伝送された後、パーティション ローダーまたは U-Boot のいずれかが DDR のパーティションを DDR の別の場所、オンチップ メモリ (OCM)、またはコンフィギュレーション メモリへコピーします。U-Boot を使用すると、DDR のパーティションを Quad-SPI に書き込むことができます。リブート時にエンベデッド システムが最新のパーティションをロードする場合には Quad-SPI への書き込みが必要です。パーティション ローダーは Quad-SPI への書き込みをサポートしていません。

パーティション ローダーは U-Boot より小さくて使用が簡単です。U-Boot はパーティション ローダーより機能性に優れています。パーティション ローダーと U-Boot は共にソフトウェア、ハードウェア、またはデータ パーティションをコピーし、AES 暗号および RSA 認証をサポートします。カスタム変更が必要な場合は、パーティション ローダーへの変更を非公開として保持できます。U-Boot はオープン ソースです。一部のエンベデッド システムは、パーティション ローダーと U-Boot の両方を使用します。

Zynq デバイスで実行する場合、Linux はビットストリームなどのパーティションをロードできます。Linux は、パーティションを Quad-SPI へ書き込むこともできます。Vivado Design Suite 2014.3 リリースの時点で、Linux のビットストリーム コンフィギュレーション動作は、セキュア モードでテストされていません。

注記 : ブート時のシステム アップデートを保持するには、U-Boot が NVM にアップデート情報を書き込む必要があります。Quad-SPI を NVM として使用する場合は、U-Boot **`sf write`** コマンドを使用します。

システム アップデートの実行

図 2 に、システム アップデート フローの機能を示します。FSBL (ファースト ステージ ブートローダー) が不揮発性メモリ (NVM) の hello ソフトウェア プロジェクトを DDR メモリへコピーします。hello ソフトウェアは通信端末に「Hello World」と表示します。この hello プログラムの後に、パーティション ローダーが実行されます。システム アップデートを実行する場合、パーティション ローダーは hello_update をロードします。実行中、通信端末には「Hello World Update」出力が表示されます。

実際のシステムでは、スケジューラーがプログラムの実行を管理します。このアプリケーション ノートのソフトウェア プロジェクトは、ブートやランタイム動作をエミュレートする経験則に基づいたリファレンス システムを提供しています。実際のシステムにおけるシステム アップデートは非同期的に行われます。FSBL と hello のロードはブート時に発生します。リファレンス システムでは、CPU が hello から partition_loader へ遷移するときにランタイムがモデル化されます。

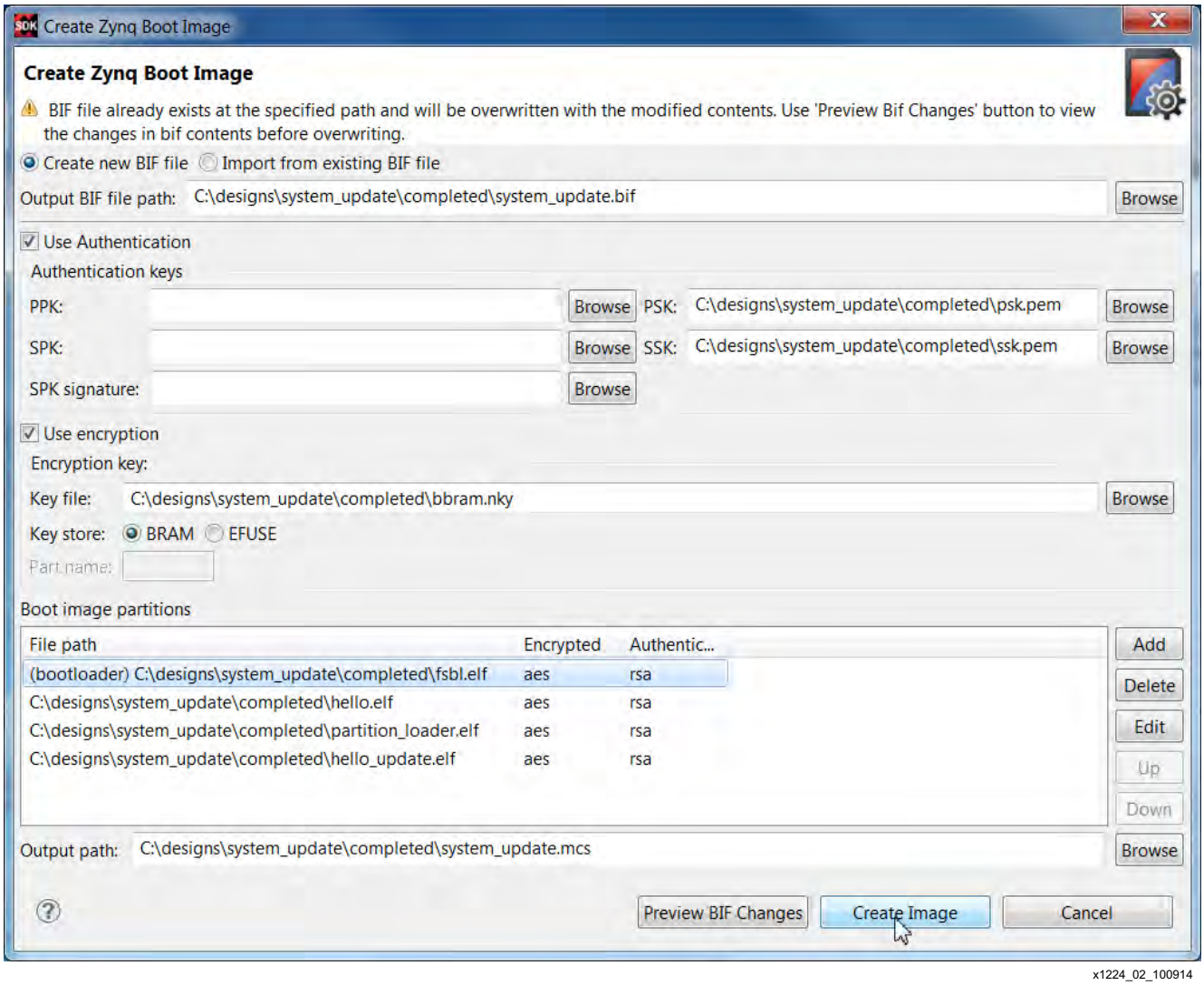


図 2 : ブート タイム ー ランタイムの抽象化

SDK を使用してシステム アップデートする手順は次のとおりです。

1. fsbl、hello、partition_loader/U-Boot、および hello_update ソフトウェア プロジェクトを作成します。
2. Bootgen を使用して system_update.mcs イメージを作成します。

3. Flash Writer を使用して Quad-SPI に書き込みます。
4. SDK で FSBL および Hello プロジェクト テンプレートを作成します。`system_update` という名前のディレクトリを作成して SDK を起動します。
5. [Workspace Launcher] のテキスト ボックスで `<path>\system_update` を参照して、[OK] をクリックします。
6. [File] → [New] → [Application Project] をクリックします。
7. [Project Name] に「**`fsbl`**」と入力します。
8. [Target Hardware] → [Hardware Platform] で [ZC702_hw_platform] に変更して、[Next] をクリックします。
9. [New Project Templates] ダイアログ ボックスで [Zynq FSBL] を選択して、[Finish] をクリックします。
10. fsbl のコンパイル完了後、[Project Explorer] ペインで [fsbl] を右クリックします。
11. [C/C++ Build Settings] をクリックします。
12. [Symbols] をクリックして、+ をクリックします。
13. 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。
14. FSBL プロジェクトの作成時に行った手順を繰り返して、`hello` プロジェクトと `hello_update` プロジェクトを作成します。[New Project Templates] ダイアログ ボックスで [Hello World] を選択します。
15. [Project Explorer] ペインで `hello` プロジェクトを変更します。変更したソース ファイルは、`hello_src` ディレクトリに格納されます。
    a. [File Import] → [General] → [File System] をクリックして、`hello/src` に `system update/hello_src/handoff.S` を追加します。
    b. `hello/src/helloworld.c` を `system_update/hello_src/helloworld.c` に置き換えます。

       新しい `helloworld.c` は、コードに 3 つの行が追加されており、`hello` プロジェクトが「Hello World」と表示した後にパーティション ローダーが実行されます。

       int main(): の直前に、次の 2 行が挿入されています。

       ```
       u32 PartitionLoaderStartAddress = 0x00400000;
       void HandoffExit(u32 PartitionLoaderStartAddress);
       ```

       `cleanup_platform();`: の後に次の行が挿入されています。

       ```
       HandoffExit(0x00400000);
       ```

    c. [Project Explorer] ペインで `hello_update` プロジェクトを変更します。`helloworld.c` の print ステートメントを次のように変更します。

       ```
       print(“Hello World from updated hello”);
       ```

    d. [Project Explorer] ペインで、`hello_update/src/lscript.ld` リンカー スクリプトを変更して、`hello_update` を `0x400000` に配置します。

パーティション ローダー

パーティション ローダーは、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をコピーします。パーティション ローダーは、devcfg デバイス ドライバーのラッパーです。devcfg ドライバーのソース コードは、`XilinxProcessorLib/driver/devcfg_v*` ディレクトリにある SDK インストール エリアに含まれています。

`PcapDataTransfer` および `PcapLoadPartition` 関数は `pcap.c` にあります。`PcapDataTransfer` 関数は、ソフトウェアやデータのパーティションをコピーします。ビットストリームがコンフィギュレーション メモリにコピーされる場合、`PcapLoadPartition` 関数が使用されます。暗号化せずに NVM から DDR へビットストリームをコピーする場合は、`PcapDataTransfer` 関数を使用できます。

パーティション ローダー関数の引数は、ソース/デスティネーションのアドレス、ソース/デスティネーションのサイズ、およびパーティションの送信を復号化するか否かを示します。パーティションのアドレスおよびサイズ パラメーターは、ハード コードされています。

リファレンス システムのパーティション ローダーは、基本的なシステム アップデート機能を示します。システム アップデート用のインターフェイスをユーザー (デバイス マネージャー) に提供するため、パーティション ローダーの関数を変更して、変数あるいはメモリ アドレスとしてアドレスおよびサイズ パラメーターが与えられるようにします。

パーティションのアドレスおよびサイズを取得するには、次の 2 つの方法があります。

- `-debug` オプションを使用して Bootgen を実行します。

  ```
  bootgen -image hello_update.bif -o hello_update.bin -w on -debug
  ```

  BIF がすべてのパーティションを含む場合、Bootgen はアドレスおよびサイズ情報も提供します。

- FSBL のログ出力を使用して、アドレスおよびサイズ情報を取得します。FSBL_DEBUG_INFO オプションを用いて FSBL をコンパイルすると、ログが提供されます。

現在、パーティション ローダーは RSA 認証を使用しません。ザイリンクスの RSA ライブラリを用いてパーティションを認証する方法は、『Zynq-7000 All Programmable SoC システム メモリのランタイム整合性および認証の確認』(XAPP1225) [参照 2] を参照してください。

次の手順に従って、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. [File] → [New] → [Application Project] をクリックします。
2. [Project Name] に「**`partition_loader`**」と入力して、[Next] をクリックします。
3. [New Project Templates] で [Empty Application] を選択して、[Finish] をクリックします。
4. [Project Explore] ペインで [partition_loader] を右クリックして、[src] を選択します。
5. [Import] → [General] → [File System] をクリックします。
6. `system_update/partition_loader_src` を参照し、すべてのファイルを選択して [OK] をクリックします。
7. [Project Explore] ペインで `partition_loader/src/lscript.ld` を選択します。
8. リンカー スクリプトを変更して `partition_loader` を `0x600000` に配置します。

パーティション ローダーは、最初のブートで使用される BIF 内に含まれるため、ランタイム時にはメモリ内に常駐します。

U-Boot

U-Boot は、さまざまな用途に使用されるユニバーサル ブート ローダーであり、一般に Linux カーネル、デバイス ツリー、ルート ファイル システム、Linux アプリケーションをロードする場合に利用されます。[参照 3] では、U-Boot 用のユーザー マニュアルなどの資料を提供しています。[参照 4] では、U-Boot の設定および構築に関するザイリンクスの資料を提供しています。組み込み Linux のほとんどのテキストには、U-Boot セクションが含まれます。

1. ザイリンクスの Git サーバーから U-Boot をダウンロードする場合、次のコマンドを入力します。

   ```
   git clone git://github.com/Xilinx/u-boot-xlnx.git
   cd u-boot-xlnx
   ```

2. `include/configs/zynq-common.h` に、次の `define` ステートメントを追加します。

   ```
   #define CONFIG_CMD_ZYNQ_AES
   ```

3. 次のコマンドを入力して U-Boot を設定および構築します。

   ```
   make arch=ARM zynq_zc70x (zynq_zed)
   ```

次に示すコマンド セットは、U-Boot を使用して PL を構成する方法を示しています。

```
promgen -w -b -p bin -o system.bin -u 0 system.bit -datawidth 32
Copy system.bin to a SD card.
```

4. XMD を起動して次の XMD コマンドを実行します。

   ```
   connect arm hw
   dow fsbl.elf
   con
   stop dow u-boot.elf
   con
   ```

5. キーを押して自動ブートを回避します。
6. U-Boot のプロンプト画面で、通信端末に次のコマンドを入力します、

   ```
   mmcinfo
   ```

```
fatload mmc 0 0x100000 system.bin
fpga load 0 0x100000 0x3DBAFC
```

7. 復号化してビットストリームを送信するには、fpga コマンドを zynqaes に置き換えます。

```
zynqaes <srcaddr> <srclen> <destaddr> <destlen>
```

qspiboot の include/configs/zynq-common.h セクションは、Linux カーネル、デバイス ツリー、およびルート ファイル システムをロードします。コマンドは、zynqaes を使用して hello_update をロードするように変更できます。system.bin パーティションは Bootgen で暗号化されます。

Bootgen

次の手順で SDK Bootgen GUI を起動します。

1. [Xilinx Tools] → [Create Zynq Boot Image]
2. [BIF File Path] ダイアログ ボックスで、パスと system_update.bif ファイル名を指定します。
3. + をクリックして fsbl.elf を追加します。
4. [Add] をクリックして、hello.elf、partition_loader.elf、および hello_update.elf ファイルを追加します。
5. 出力ファイルとして system_update.mcs を指定します。
6. [Create Image] をクリックします。

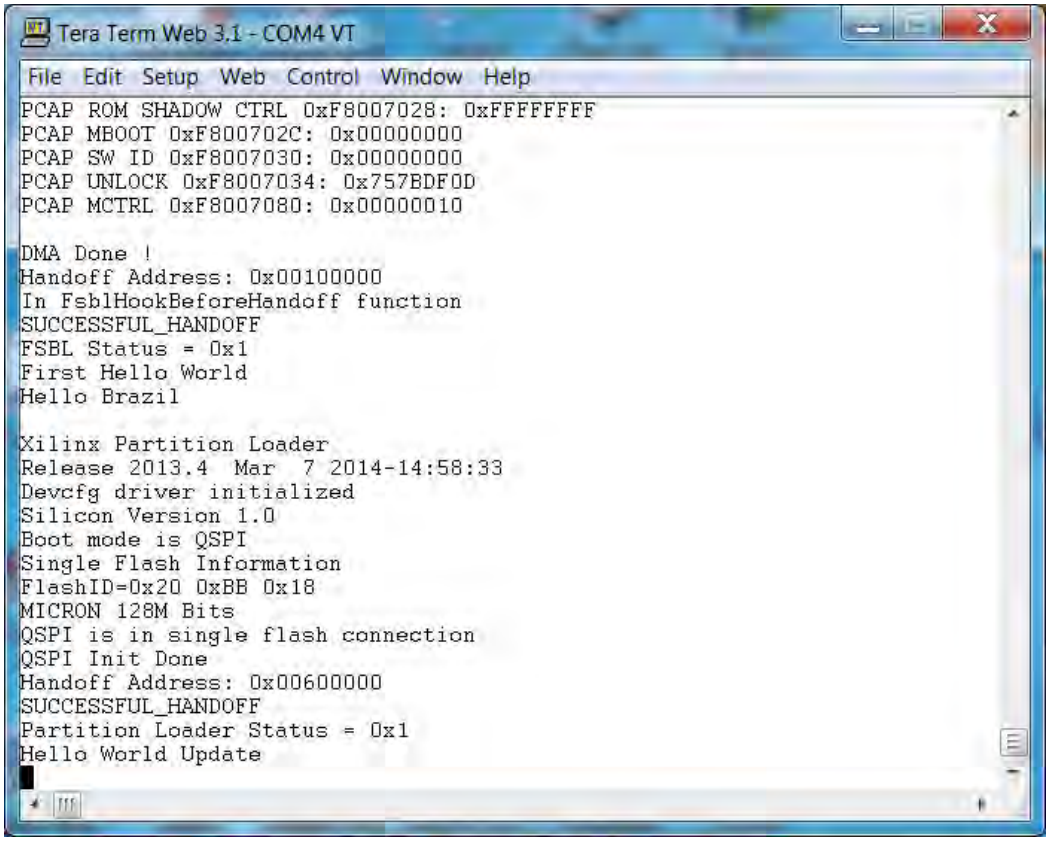
クワッド シリアル ペリフェラル インタフェース (Quad-SPI) のプログラム

1. [Xilinx Tools] → [Program Flash] をクリックし、SDK Program Flash を起動します。
2. system_update.mcs を参照して Image File 入力を完成させます。
3. [Offset] に「**0x0**」と入力します。
4. [Program] をクリックします。

システム アップデートのテスト

1. ボー レートを 115200 に設定した Teraterm 通信端末を起動します。
2. 評価ボード上で、ブート モード選択スイッチ MIO5 を 1 に変更して、Quad-SPI モードでブートします。

図 3 に示すように、通信端末のメッセージ画面に FSBL、hello、パーティション ローダー、および hello update プログラムのアクティビティが表示されます。最初の hello ソフトウェアでは、「Hello World」および「Hello Brazil」と表示されます。コードがパーティション ローダーへ移動すると、「Xilinx Partition Loader」および「Handoff Address:0x00600000」と表示されます。hello システムのアップデートは、「Hello World Update」というメッセージで確認できます。



x1224_04_100914

図 3 : システム アップデートを示す通信端末の出力

システム アップデートのセキュリティ

インターネット経由でシステムをリモートからアップデートする場合、中間者攻撃にさらされる可能性があります。ザイリンクスは、機密性を保持するための AES 暗号および機密情報の整合性を保護するための RSA 認証を提供しています。

ファクトリで Bootgen を使用して、AES 暗号化済みまたは RSA 署名済みのアップデート パーティションを作成します。Bootgen は、ブート用のパーティションをすべて含むイメージ、またはランタイム アップデート用のパーティションを 1 つ含むイメージを作成できます。Bootgen イメージ フォーマット (BIF) ファイルのセキュリティ属性の使用については、『Zynq-7000 All Programmable SoC のセキュア ブート』(XAPP1175) [参照 1] を参照してください。

パーティション ローダーは、`PcapLoadPartition` および `PcapDataTransfer` 関数の引数を使用してセキュリティをサポートします。U-Boot は、`zynqaes` コマンドおよび `zynqrsa` コマンドを利用してセキュリティをサポートします。リファレンス システムでは、パーティション ローダーと U-Boot を用いた信頼性の高いアップデート方法を示しています。

エンベデッド システムでは、パーティションの復号化や認証を行うために、Zynq SoC がハード AES 復号化とソフト RSA ライブラリを提供します。

非セキュア モードとセキュア モードの違いは、BIF ファイルです。図 4 に、セキュア モードのシステム アップデート用 BIF ファイルを示します。この図に示すように、AES と RSA のキーを入力します。

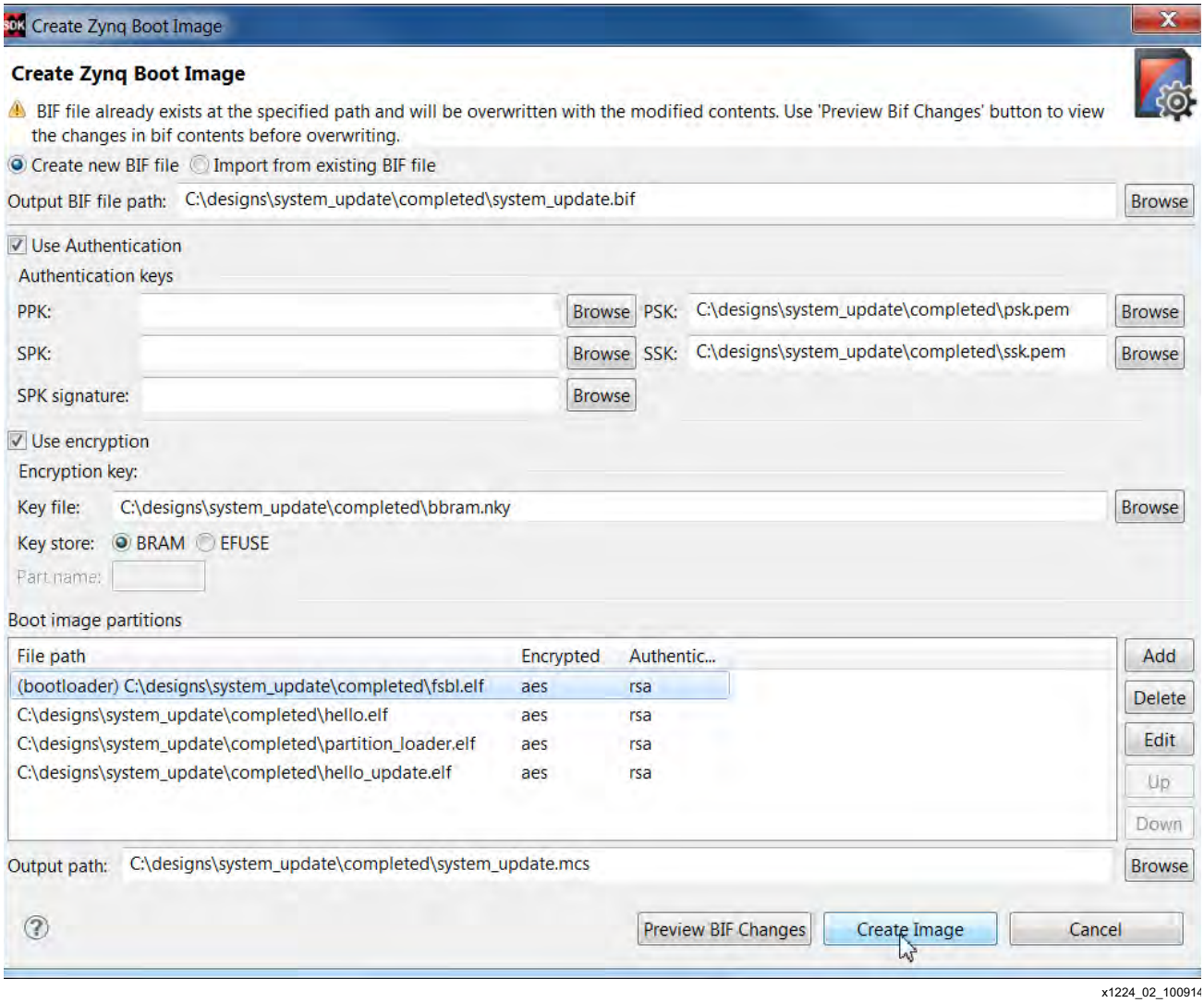


図 4 : セキュア システム アップデートの BIF

リファレンス システム

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 1 に、リファレンス デザインの詳細を示します。

表 1 : 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
ターゲット デバイス	Zynq All Programmable 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 ノートやリファレンス デザイン、またはサードパーティからデザインへのコード/IP の使用	なし
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	なし
テストベンチの形式	N/A
使用したシミュレータ/バージョン	N/A
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	なし
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	ZC702

リファレンス システムは、不揮発性メモリ (NVM) としてクワッド シリアル ペリフェラル インターフェイス (Quad-SPI) を使用します。Quad-SPI は、すべての Zynq-7000 評価ボードに搭載されており、プロダクション ボードでは一般的によく使用されます。Quad-SPI を用いたシステム アップデート方法は、NAND、NOR、または SD の NVM で可能です。

まとめ

Zynq All Programmable SoC のハードウェア/ソフトウェア プログラマビリティは、エンベデッド システムのライフサイクルにおいてさまざまなメリットをもたらします。動作中にユーザーまたはデバイス マネージャーがシステムをアップデートできるため、エンベデッド システムの機能性、有効性、信頼性が向上します。システムは、Zynq デバイスが動作している間に、パーティション ローダーまたは U-Boot を使用する 2 つの方法でアップデートできます。

参考資料

1. 『Zynq-7000 All Programmable SoC のセキュア ブート』(XAPP1175)
2. 『Run Time Integrity and Authentication Check of Zynq-7000 All Programmable SoC システム メモリのランタイム インテグリティおよび認証チェック』(XAPP1225)
3. www.denx.de/wiki/U-Boot
4. www.wiki.xilinx.com/Build+U-Boot#Zynq
5. 『Zynq-7000 XC7020 AP SoC 向け ZC702 評価ボード ユーザー ガイド』(UG850)
6. 『Zynq-7000 All Programmable ソフトウェア開発者向けガイド』(UG821)

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 10 月 22 日	1.0	初版

法的通知

この資料に関するフィードバックおよびリンクなどの問題につきましては、jpn_trans_feedback@xilinx.com まで、または各ページの右下にある [フィードバック送信] ボタンをクリックすると表示されるフォームからお知らせください。フィードバックは日本語で入力可能です。いただきましたご意見を参考に早急に対応させていただきます。なお、このメール アドレスへのお問い合わせは受け付けておりません。あらかじめご了承ください。