Summary

The Zynq®-7000 All Programmable (AP) SoC provides several mechanisms for securing sensitive data: cryptographic operations, anti-tamper (AT), and TrustZone. The cryptographic operations use the Advanced Encryption Standard (AES) and the Rivest Shamir Adleman (RSA) algorithms. Because of security, resource, and performance issues, embedded systems should be architected with specific security requirements for each software, hardware, and data partition. This application note discusses the use of a decrypt on demand methodology to efficiently protect the confidentiality of sensitive information during boot and run time.

Download the Reference Design files for this application note from the Xilinx website. For detailed information about the design files, see Reference Design.

Included Systems

The xapp1226-protecting-info.zip reference design shows how to use encryption to ensure the confidentiality of sensitive information. The reference design contains the following software projects:

- First Stage Boot Loader (FSBL)
- partition_loader
- U-Boot
- pl_pwrctrl

The ELF files for these software projects, cryptographic keys, and Bootgen Image Format (BIF) file are included in the xapp1226-protecting-info/completed directory. The source files for the software projects are provided in the partition_loader_src and pl_pwrctrl_src directories.

Introduction

Sensitive information needs to be protected at boot and run time. Sensitive information can be protected by storing it encrypted in system memory, either nonvolatile memory (NVM) or dual data rate (DDR) random access memory (RAM). In the Zynq-7000 AP SoC device, two cryptographic methods for protecting sensitive information during run time are inline decryption and the decrypt on demand method discussed in this application note. The inline
decryptor decrypts instructions and data. In the decrypt on demand method, encrypted partitions are copied to DDR at boot time. A partition can contain software, hardware (bitstream), or a data file. Because it is encrypted, the partition is not exposed as plaintext in DDR. When used, the partition is decrypted using the Zynq-7000 AP SoC device’s AES decryptor and placed into the Zynq-7000 device’s secure storage. The sensitive code is executed from secure storage. If the decrypted partition is data, the data is accessed from secure storage.

Both the inline decryptor and decrypt on demand method have a performance impact. In most applications, the performance impact of the decrypt on demand application is small. The inline decryptor uses programmable logic (PL) resources. The decrypt on demand method does not use PL resources. An implementation for the decrypt on demand method is given in this application note.

The System Background section discusses embedded system architecture as it relates to the security concepts presented in this application note. Some factors in identifying sensitive information in an embedded system are discussed. Because boot and run time performance decrease when a partition is encrypted, only partitions that require confidentiality should be encrypted. An overview of the reference design is given in Reference Design Background, page 5. The software projects and the image for NVM are built. In Test the Reference Design, page 11, the reference design is run and the results are analyzed. Methods to estimate and measure performance are discussed.

Hardware and Software Requirements

The hardware and software requirements for this application note for the Zynq-7000 AP SoC device are:

- ZC706 evaluation board
- AC power adaptor (12 VDC)
- USB cable, standard-A plug to micro-B plug
- Xilinx Platform Cable USB II
- Xilinx Software Development Kit 2014.3
- Vivado® Design Suite 2014.3

System Background

This section provides a background of boot and run time security in the Zynq-7000 AP SoC device. A block diagram of the Zynq-7000 based embedded system architecture outlines the resources available to the security architect (Figure 1). All partitions do not need to be encrypted. To aid a security architect in specifying which software, data, and hardware partitions should be encrypted, sensitive information is discussed. An overview of the secure boot and the decrypt on demand functionality is given.
Embedded System Architecture

The resources of embedded systems are limited when compared to enterprise systems. Figure 1 shows a Zynq-7000 based embedded system that consists of the Zynq-7000 AP SoC device, DDR, and NVM. The direct memory access controller (DMAC) efficiently copies data from NVM to secure storage. The on-chip memory (OCM) and block RAM is secure storage in the Zynq-7000 AP SoC device security perimeter. When secure storage is used, unlike NVM and DDR, an adversary does not have access to address, control, and data pins. Pin access allows the memory content to be read.

Configuration memory and cryptographic key storage are also secure storage. The hardware (bitstream) is encrypted in NVM and copied/decrypted to configuration memory. Because hardware is not typically stored in DDR, the reference design focuses on software and data partitions.

In the decrypt on demand method, sensitive software and data are stored in encrypted format in NVM and/or DDR. When the sensitive partition is used, the partition loader copies it into OCM or block RAM, routed through the AES decryptor. The CPU executes the code or reads the data from OCM or block RAM. The decrypt on demand functionality can be re-used as many times as needed during operation.

The size of secure storage is much smaller than the size of DDR. The size of the OCM is 256 KB. ARM® Cortex™-A9 MPCore™ CPU operations execute very fast from OCM. The sensitive software or data partition must fit into the 256 KB OCM. OCM is large enough for most sensitive partitions. If OCM is not large enough, block RAM can be used. Another option is to divide the partition into multiple blocks that fit in OCM, with the blocks executed sequentially. The size of block RAM depends on the Zynq-7000 AP SoC device used, with the largest about 3 MB. Xilinx wikis provide information on running code from block RAM and locked L2 cache [Ref 1], [Ref 2].
Sensitive Information

The cryptographic operations supported by a Zynq-7000 AP SoC device are encryption, integrity check, and authentication. All partitions should be checked for integrity at boot. All static partitions should be checked for integrity periodically during run time. Not all partitions are confidential. Partitions that are not confidential do not need to be encrypted. Open source code like U-Boot or Linux generally does not need to be encrypted. The security architect should determine which partitions are encrypted. Some partitions are marginally sensitive. If code uses a proprietary algorithm that provides a competitive product advantage, it should be encrypted. If the code is trivial to reverse engineer or there are readily available alternative implementations, it might not warrant encryption. The decision might trade off performance and the sensitivity of the partition.

The most important sensitive information is private cryptographic keys. In a Zynq-7000 AP SoC device, the eFUSE and battery-backed random access memory (BBRAM) private keys reside in secure storage. The BBRAM key can be changed, and the new key(s) should be stored in secure storage. In some cases, cryptographic keys should be stored in secure storage in encrypted format until used.

Data partitions might be confidential and should be encrypted. Examples are patient health records and customer financial records.
With system-critical applications such as power control, the protection should principally be that an adversary cannot gain access or tamper with the partition. Access is controlled with TrustZone and access privileges. Tampering is detected using an RSA integrity check. Encryption prevents the adversary from reading the information in the partition.

**Boot Time Security**

When power is applied to a Zynq-7000 AP SoC device, each of other software, hardware (bitstream), and data partitions are copied from NVM to DDR or Zynq-7000 secure storage. Using the Bootgen Image Format (BIF) file input to Bootgen, the security architect specifies for each partition if it is AES encrypted and/or RSA signed. The factors affecting the decision on which partitions are encrypted were discussed in the previous section. Secure Boot of Zynq-7000 All Programmable SoC Application Note (XAPP1175) [Ref 3] shows how to specify on a partition basis whether a partition is encrypted/signed.

Security can be compromised if the Zynq-7000 AP SoC decrypts software/data partitions and copies them to DDR. If the software and data partitions in NVM are encrypted, the default behavior of the 2014.3 FSBL is to copy the partition to DDR, routed through the AES decryptor. The cleartext software or data partition in DDR is as vulnerable as it is if it is unencrypted in NVM.

To address the problem of a sensitive partition being stored in plaintext in DDR, there are three methods of copying a partition encrypted in NVM to DDR without decryption. The FSBL can be edited so that the call to PcapDataTransfer does not route the partition through the decryptor. A second method is a U-Boot **sf read** command. Alternatively, the partition loader can copy the partition from NVM to DDR without decryption.

---

**Reference Design Background**

*Figure 2* shows the flow for running the xapp1226-protecting-info reference design. In the reference design, boot time is modeled by the load of the bitstream software. Run time begins with the execution of software. In a real system, run time is defined as the time the Zynq-7000 AP SoC device is operational after boot. In the reference design, run time starts when CPU executes the **partition_loader**. During run time, the **pl_pwrctrl** turns off and later turns back on the power supply to the PL. The application is encrypted in NVM, and when scheduled to run, copied to OCM through the decryptor.

The FSBL copies the unencrypted **partition_loader** software project from NVM to DDR memory. The partition loader loads the encrypted **pl_pwrdown** partition into OCM, routed through the decryptor. When **pl_pwrctrl** is executed, the power down of the PL message is displayed in the communication terminal. After a delay, **pl_pwrctrl** re-applies power to the PL.

In an actual system, a scheduler runs programs. The reference design is a heuristic reference design that emulates boot and run time operation. In actual systems, sensitive partitions are run or accessed dynamically when needed.
Using the Partition Loader and U-Boot in the Decrypt on Demand Method

In the decrypt on demand method, when an encrypted partition is run, either the partition loader or U-Boot copies the partition from DDR to OCM or block RAM. The partition is routed through the AES decryptor.

The partition loader is smaller and easier to use than U-Boot. U-Boot provides functionality not available in the partition loader. Both the partition loader and U-Boot copy software, hardware, or data partitions, and both support AES decryption and RSA authentication. If custom modifications are needed, the modifications to the partition loader can remain proprietary. U-Boot is open source. Some embedded systems use both the partition loader and U-Boot.
The following steps use SDK to implement the xapp1226-protecting-info reference design:

1. Create fsbl, partition_loader, and/or U-Boot, and pl_pwrctrl software projects.
2. Create the system.mcs image using Bootgen.
3. Write Quad serial peripheral interface (SPI) flash memory using Flash Writer with the MCS file.

Table 1 shows the address map used in the reference design.

<table>
<thead>
<tr>
<th>Partition</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSBL</td>
<td></td>
</tr>
<tr>
<td>system.bit</td>
<td></td>
</tr>
<tr>
<td>partition_loader</td>
<td>0x100000</td>
</tr>
<tr>
<td>pl_pwrctrl</td>
<td>0x200000</td>
</tr>
</tbody>
</table>

SDK is used to create the fsbl project. Create a directory named xapp1226-protecting-info and invoke SDK.

1. In the Workspace Launcher text box, browse to the xapp1226-protecting-info directory. Click OK.
2. Click File > New > Application Project.
3. Enter fsbl as the Project Name, and change Target Hardware > Hardware Platform to ZC706_hw_platform. Click Next.
4. In the New Project Templates dialog box, click Zynq FSBL. Click Finish.
5. After FSBL compilation is complete, right-click on fsbl in the Project Explorer pane.
6. Click C/C++ Build Settings.
7. Click Symbols.
8. Click +.
9. Enter FSBL_DEBUG_INFO in the text box. Click OK.

Partition Loader

The partition loader copies software, data, or hardware partitions from the specified source address to the specified destination address. The partition loader is a wrapper for the devcfg device driver. The devcfg driver source code is located in the SDK install area, under the XilinxProcessorLib/driver/devcfg_v* directories.

The functions used by the partition loader, PcapDataTransfer and PcapLoadPartition, are located in pcap.c. The PcapDataTransfer function copies a software or data partition. The PcapDataTransfer function can copy the bitstream from NVM to DDR without decryption. The PcapLoadPartition function copies the bitstream to configuration memory. The arguments to
the partition loader functions are source and destination address, source and destination size, and whether to route the partition through the decryptor.

The partition loader in the reference design provides basic decrypt on demand capability. The partition address and size parameters are hard-coded. To provide the user with a more useful interface, modify the partition loader to allow the address and size parameters to be provided as arguments or memory addresses.

There are two methods to get a partition’s address and size. One method is to run bootgen using the -debug option:

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

Bootgen also provides address and size information when the BIF includes all partitions.

The second method to obtain address and size information is to use the log output of the FSBL. The log is provided when the FSBL is compiled with the FSBL_DEBUG_INFO option.

The partition loader does not currently use RSA authentication. *Run Time Integrity and Authentication Check of Zynq-7000 AP SoC System Memory Application Note* (XAPP1225) [Ref 4] shows how to use the Xilinx RSA library to authenticate a partition.

Create the partition loader software project as follows:

1. Click File > New > Application Project.
2. Enter partition_loader in the Project Name text box. Click Next.
3. In the New Project Templates, select Empty Application. Click Finish.
4. In the Project Explorer pane, right-click partition_loader > src, and select Import > General > File System. Browse to xapp1226-protecting-info/partition_loader_src, and select all files. Click OK.
5. In the Project Explorer pane, select partition_loader_src/lscript.ld, and modify the linker script to locate partition_loader at 0x200000.

The partition loader is included in the BIF used for initial boot, so that it is resident in memory at run time.

**U-Boot**

U-Boot is a widely used universal boot loader commonly used to load the Linux kernel, device tree, root file system, and Linux applications. The *U-Boot—The Universal Boot Loader website* [Ref 5] provides documentation, including the user manual, for U-Boot. The *U-Boot for Zynq AP SoC website* [Ref 6] provides Xilinx-specific documentation on configuring and building U-Boot. Most texts on embedded Linux include a section on U-Boot.

1. To download U-Boot from the Xilinx Git server, enter

   ```
git clone git://github.com/Xilinx/u-boot-xlnx.git
   cd u-boot-xlnx
   ```
2. Add the following define to include/configs/zynq-common.h.

#define CONFIG_CMD_ZYNQ_AES

3. Configure and build U-Boot by entering

make arch=ARM zynq_zc70x (zynq_zed)

The following set of commands illustrates how to use U-Boot to copy the partition from NVM to DDR.

sf probe 0 0 0;

sf read <src_address> <dest_address> <size>

When copying the partition from DDR to secure storage, to route the partition through the decryptor, use the zynqaes command:

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

The include/configs/zynq-common.h section for the Quad SPI flash memory boot loads the Linux kernel, device tree, root file system, and Linux applications. The zynq_common.h commands can be changed to copy pl_pwrdown and pl_pwrup partitions using zynqaes.

U-Boot is principally a boot time application. It is used here as a run time application that copies a single partition.

Bootgen
1. To invoke the SDK Bootgen GUI, select Xilinx Tools > Create Zynq Boot Image.
2. Specify the path and system.bif file name in the BIF File Path dialog box.
3. Click + and add fsbl.elf.
4. Click Add, and add system.bit, partition_loader.elf, and pl_pwrctrl.elf partitions. Specify system.mcs as the output file. Click Create Image.
Figure 3 shows the Bootgen GUI with the partitions for the xapp1226-protecting-info system.

Figure 3: Bootgen BIF in GUI

If U-Boot is used rather than the partition loader for the run time copy/decrypt function, replace the partition_loader partition with the U-Boot partition in the BIF, and use the partition_owner = uboot attribute on the file(s) to be loaded by U-Boot. The xapp1226-protecting-info/completed directory contains system.bif.
Programming Quad SPI Flash Memory

1. Invoke SDK Program Flash by selecting Xilinx Tools > Program Flash.
2. Browse to system.mcs to complete the Image File input.
3. Enter 0x0 as in the Offset dialog box.
4. Click Program.

Test the Reference Design

1. Invoke a communication terminal such as Tera Term using a 115200 baud rate.
2. On the evaluation board, change the boot mode select switch MIO5 to 1 to boot using QSPI mode.

As shown in Figure 4, the message in the communication terminal shows the activity of the reference design. The partition_loader loads the encrypted partition that powers down and powers up the PL.

![Communication Terminal Output Showing the Protection of the pl_pw/ctrl Software](x1226_03_103114)

Figure 4: Communication Terminal Output Showing the Protection of the pl_pw/ctrl Software
Performance Notes

The performance impact occurs when the sensitive code or data is copied from DDR, decrypted, and placed in secure storage. The load/decrypt time is a function of the size of the partition. After the code or data is in secure storage, code should run marginally faster in OCM than DDR.

The Cortex-A9 processor includes a performance monitoring unit (PMU) that provides counters to provide statistics on the processor and memory system. The performance data can be used to determine if the performance degradation due to encryption is acceptable. The Zynq-7000 All Programmable SoC Technical Reference Manual, Chapter 3, provides information on the PMU [Ref 7].

AXI performance can be monitored using the axipmon core in SDK/../XilinxProcessorIPLib/drivers/axi_pmonv6_1.

ARM DS-5 Development Studio Streamline performance analyzer provides system performance metrics for users with access to DSTREAM debugger. Refer to Zynq-7000 Platform Software Development Using the ARM DS-5 Toolchain Application Note (XAPP1185) [Ref 8].

Reference Design

The reference design uses Quad SPI flash memory as the NVM. Quad SPI flash memory is provided on all Zynq-7000 evaluation boards. Quad SPI flash memory is commonly used on production boards. The methods used with Quad SPI for xapp1226-protecting-info can be used with NAND, NOR, or SD NVM.

Download the Reference Design files for this application note from the Xilinx website.

Table 2 shows the reference design matrix.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General</strong></td>
<td></td>
</tr>
<tr>
<td>Developer name</td>
<td>Lester Sanders</td>
</tr>
<tr>
<td>Target devices</td>
<td>Zynq-7000 AP SoC</td>
</tr>
<tr>
<td>Source code provided</td>
<td>Yes</td>
</tr>
<tr>
<td>Source code format</td>
<td>VHDL/Verilog</td>
</tr>
<tr>
<td>Design uses code and IP from existing Xilinx application note and reference designs or third party</td>
<td>No</td>
</tr>
<tr>
<td><strong>Simulation</strong></td>
<td></td>
</tr>
<tr>
<td>Functional simulation performed</td>
<td>No</td>
</tr>
<tr>
<td>Timing simulation performed</td>
<td>N/A</td>
</tr>
<tr>
<td>Test bench used for functional and timing simulations</td>
<td>No</td>
</tr>
<tr>
<td>Test bench format</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Conclusion

The hardware and software programmability of the Zynq-7000 AP SoC device provide distinct advantages during the life cycle of the embedded system. The ability to decrypt a partition on demand provides a reasonable security versus performance compromise. Two methods, a partition loader and U-Boot, are provided that decrypt an encrypted partition on demand and load the partition into the Zynq-7000 device’s secure storage. The method is straightforward, flexible, and does not use PL resources.

References

1. Zynq-7000 AP SoC - Using BBRAM for Additional On-Chip Memory Tech Tip [wiki]
2. Zynq-7000 AP SoC Boot - Locking and Executing out of L2 Cache Tech Tip [wiki]
3. Secure Boot of Zynq-7000 All Programmable SoC Application Note ([XAPP1175](https://www.xilinx.com))
4. Run Time Integrity and Authentication Check of Zynq-7000 AP SoC System Memory Application Note ([XAPP1225](https://www.xilinx.com))
5. U-Boot—The Universal Boot Loader website
6. U-Boot for Zynq AP SoC
8. Zynq-7000 Platform Software Development Using the ARM DS-5 Toolchain Application Note ([XAPP1185](https://www.xilinx.com))
9. ZC702 Evaluation Board for Zynq-7000 XC7020 All Programmable SoC User Guide ([UG850](https://www.xilinx.com))

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulator software/version used</td>
<td>N/A</td>
</tr>
<tr>
<td>SPICE/IBIS simulations</td>
<td>N/A</td>
</tr>
<tr>
<td>Implementation</td>
<td></td>
</tr>
<tr>
<td>Synthesis software tools/versions used</td>
<td>Vivado synthesis</td>
</tr>
<tr>
<td>Implementation software tools/versions used</td>
<td>Vivado implementation</td>
</tr>
<tr>
<td>Static timing analysis performed</td>
<td>No</td>
</tr>
<tr>
<td>Hardware Verification</td>
<td></td>
</tr>
<tr>
<td>Hardware verified</td>
<td>Yes</td>
</tr>
<tr>
<td>Hardware platform used for verification</td>
<td>ZC702</td>
</tr>
</tbody>
</table>

Table 2: Reference Design Matrix (Cont’d)
Revision History

The following table shows the revision history for this document.

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/17/2014</td>
<td>1.0</td>
<td>Initial Xilinx release.</td>
</tr>
<tr>
<td>11/19/2014</td>
<td>1.0.1</td>
<td>Updated the Reference Design files link.</td>
</tr>
</tbody>
</table>

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at [http://www.xilinx.com/legal.htm#tos](http://www.xilinx.com/legal.htm#tos); IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at [http://www.xilinx.com/legal.htm#tos](http://www.xilinx.com/legal.htm#tos).

Automotive Applications Disclaimer

XILINX PRODUCTS ARE NOT DESIGNED OR INTENDED TO BE FAIL-SAFE, OR FOR USE IN ANY APPLICATION REQUIRING FAIL-SAFE PERFORMANCE, SUCH AS APPLICATIONS RELATED TO: (I) THE DEPLOYMENT OF AIRBAGS, (II) CONTROL OF A VEHICLE, UNLESS THERE IS A FAIL-SAFE OR REDUNDANCY FEATURE (WHICH DOES NOT INCLUDE USE OF SOFTWARE IN THE XILINX DEVICE TO IMPLEMENT THE REDUNDANCY) AND A WARNING SIGNAL UPON FAILURE TO THE OPERATOR, OR (III) USES THAT COULD LEAD TO DEATH OR PERSONAL INJURY. CUSTOMER ASSUMES THE SOLE RISK AND LIABILITY OF ANY USE OF XILINX PRODUCTS IN SUCH APPLICATIONS.

© Copyright 2014 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.