Summary

This application note provides a run time integrity checker (RTIC) for the Zynq®-7000 All Programmable SoC. The RTIC software runs on Zynq-7000 AP SoC devices and detects if there are any changes to external memory. When run periodically, the RTIC provides an effective means to ensure that nonvolatile or Double Data Rate (DDR) memory, two relatively vulnerable system components, have not been modified.

Included Systems

The `xapp1225-rtic` reference design contains the following software projects.

- `fsbl`
- `hello_world`
- `hello`
- `rtic`

The ELF/BINs of these software projects, cryptographic keys, and Bootgen Image Format (BIF) file are included in the `xapp1225-rtic/completed` directory. The source files are provided in the `hello_src` and `rtic_src` directories.

Introduction

A run time integrity checker is critical in the overall security of an embedded system. Modifications to embedded system memory can occur at boot and run time. When the security techniques described in Secure Boot of Zynq-7000 All Programmable SoC (XAPP1175) [Ref 1] are used, Zynq-7000 AP SoC devices are booted into a trusted state. Zynq-7000 AP SoC devices provide a high level of integration within their security perimeters. The external memory connected to a Zynq-7000 AP SoC device is a target when left unprotected. The RTIC ensures the security of external memory during operation, providing a necessary complement to the secure boot functionality.

This application note contains the following sections:
Hardware and Software Requirements

- **RSA Authentication in Zynq-7000 AP SoC devices** reviews Rivest, Shamir, Adleman (RSA) authentication used in Zynq-7000 AP SoC devices. The *Secure Boot of Zynq-7000 All Programmable SoC* application note (XAPP1175) [Ref 1] provides a detailed description of using RSA in Zynq-7000 AP SoC devices.

- **RTIC Application** discusses when the RTIC should be used and the type of information which should be checked for integrity.

- **Developing the RTIC Reference Design** shows how to create the RTIC. This section assumes Xilinx Software Development Kit (SDK) experience.

- **RTIC System Test** section shows how to run and review the results of the RTIC system on the ZC702 Evaluation Platform.

- **Security Considerations in Using the RTIC** discusses methods to run the RTIC securely.

---

**Hardware and Software Requirements**

The hardware requirements for updating and/or authentication of Zynq-7000 AP SoC devices are:

- ZC702, ZC706, Zed, or MicroZed evaluation boards
- AC power adaptor (12 VDC)
- USB Type-A to Micro-B cable
- Xilinx Platform Cable USB II
- Xilinx Vivado® Design Suite 2014.3
- Xilinx Software Development Kit 2014.3

---

**RSA Authentication in Zynq-7000 AP SoC devices**

RSA is public key cryptography used in Zynq-7000 AP SoC devices to ensure that an adversary has not modified software, data, or hardware (bitstream). RSA uses a private/public key pair for encryption and authentication. In RSA cryptography, the private key is not used in the embedded system. The Xilinx RSA library is used in secure boot and in the run time integrity check.

In RSA authentication, the SDK image writer (Bootgen) signs the partition(s) using the private key at the factory. The first stage boot loader (FSBL) and/or U-Boot use the RSA Library and the public RSA key(s) and signature(s) to authenticate the partition(s). For secure boot, this procedure is described in the *Secure Boot of Zynq-7000 All Programmable SoC* (XAPP1175) [Ref 1], which also includes details on generating and using RSA private and public keys, and the use of the Authentication Certificate(s) in the integrity check. The authentication methods used in secure boot are re-used in the RTIC.
RTIC Application

To detect tamper events on external memory during operation, the RTIC is run after boot. It can be started by the FSBL or a scheduler and run periodically uninterrupted until a system reset or power down. The frequency of the RTIC operation is user defined. Optimally, the integrity test is bound to the code which is to be executed, as this eliminates a “time of use - time of check” issue. If a compromise is detected quickly, corrective action might minimize the damage.

The RTIC checks a single partition, which is software, data, or hardware (bitstream). The partition can be plain text or encrypted. The RTIC tests data in external memory, NVM or DDR, which is read only, static data. The most common integrity check is on software (code). While the code in NVM/DDR is read/writable, an integrity check can only be done if the content does not change and is not relocated.

The bitstream in configuration memory cannot be authenticated. It can be authenticated when resident in NVM or DDR. The bitstream can then be copied to configuration memory. Since configuration memory resides within the security perimeter of a Zynq-7000 AP SoC device, it is much less accessible to an adversary than external memory. An SEU check can be done on configuration memory.

Developing the RTIC Reference Design

In the reference design, the rtic project checks the integrity of the hello partition. The hello partition is located at 0x20000000 in DDR.

Figure 1 provides the steps to build the RTIC reference design. The first two steps, generating and programming the RSA keys, are discussed in Secure Boot of Zynq-7000 All Programmable SoC (XAPP1175) [Ref 1]. The fsbl, hello_world, hello, and rtic software projects are created. SDK provides application project templates for all projects. The templates for hello, hello_world, and the rtic projects are modified to enable the integrity check.

Figure 1: Runtime Integrity Check Flow
In the reference design, the FSBL loads the hello_world, hello, and rtic software projects. The rtic periodically checks the contents of hello.

To check the integrity of hello, rtic needs address and size information of the hello partition. Bootgen is used to create a standalone hello.bin partition with an authentication certificate. The address and size of the partition and the authentication certificate are determined from hello.bin. These parameters are input into the rtic project to enable the integrity check.

SDK is used to create the fsbl, hello_world, hello, and rtic software project templates.

1. Create the xapp1225-rtic directory.
2. Run the following commands to create a FSBL software project.
   a. File > New > Application Project
   b. Type fsbl in the Project Name text box.
   c. Select ZC702_hw_platform(predefined) as the hardware platform under Target Hardware
   d. Click Next
   e. Select Zynq FSBL
   f. Click Finish
3. Right-click fsbl, and select C/C++ Build Settings.
4. Click Symbols. Click ‘+’ in the Defined Symbols pane.
5. Enter RSA Support in the text box. Click OK, Apply, OK.
6. Repeat the steps used to create fsbl to create the hello and hello_world software projects, selecting Hello World in the Templates window for both projects.
7. Edit the hello_world project by copying helloworld.c and handoff.S from xapp1225-rtic/hello_world_src to the src area. With this edit, the CPU runs the rtic software after hello_world is finished.
8. Repeat the steps used to create the fsbl project to create the rtic software project, selecting RSA Authenticate App in the Templates window.
9. After the rtic project builds, right-click the rtic_bsp in the Project Explorer window, select Board Support Package Settings, check xilrsa, and click OK.
10. Click Finish.

In the Project Explorer pane, the rsa_auth_app.c and rsa_auth_app.h files are placed in the src directory under rtic.
The Project Explorer pane in Figure 2 shows the fsbl, hello_world, hello, and rtic software projects.

![SDK with FSBL, Hello, Hello_World and RTIC Projects](image)

Figure 2: SDK with FSBL, Hello, Hello_World and RTIC Projects

Edit the hello and rtic projects as outlined below.

The SDK tool Bootgen is used twice in this project. When an image is created for the boot stage, the Bootgen Image Format (BIF) input into Bootgen includes the FSBL, bitstream, and all software partitions. Bootgen generates a single file image, in either BIN or MCS format.

In the case of RSA authentication of the hello partition, the second use of Bootgen is to create an image which contains only the hello partition. As noted, this is to extract location and size parameters from hello.bin.

1. In SDK, select Xilinx Tools > Create Zynq Boot Image.
2. In the Create Zynq Boot Image window, browse to xapp1225-rtic and type hello.bif for the BIF File path.
3. Click Use Authentication.
4. In the **Authentication Keys** section, browse to `psk.pem` for the **PSK** entry. Sample Primary Secret Key (PSK), Secondary Secret Key (SSK) keys and the hash of the Primary Public Key (PPK) are provided in the `xapp1225-rtic/completed` directory. Browse to `ssk.pem` for the **SSK** entry.

5. Click **Add**.

6. In the Add Partition window, browse to `xapp1225-rtic/hello/Debug/hello.elf` to enter the File path.

7. Select **datafile** as the Partition type.

8. Select **rsa** as Authentication.

9. For the Output Path entry, browse to the `xapp1225-rtic/hello` and enter `hello.bin`.

10. Click **Create Image**.

11. As shown in **Figure 3**, open `hello.bin` in a text editor such as Textpad.

12. Search for three values starting at address **C80**:

   - **Partition Start Address** - C0 05 00 00 (second word on C90 line)
   - **Partition Size** - 03 20 00 00 (second word on C80 line)
   - **Authentication Certificate (AC) Start Address** - D0 25 00 00 (third word on CA0 line)
Developing the RTIC Reference Design

13. Calculate the size value using hex multiplication. Byte swapping is required for this operation.
   a. After byte swapping, the value 0xC0050000 translates to 0x5C0. Since there are four bytes per word, multiply by 4 to get 0x1700.
   b. As an example, to calculate the start address using python, invoke python, and enter the following commands.

   ```
   int("5C0",16)
   1472
   1472 * 4
   5888
   hex(5888)
   0x1700
   ```
The values calculated are given below:

- PARTITION START ADDRESS - 0x1700
- PARTITION SIZE - 0x8700
- AUTHENTICATION CERTIFICATE START ADDRESS - 0x9740

14. As shown in Figure 4, open rsa_auth_app.h, and provide the values just calculated to the `#define(s). When the authentication test is run, the image is loaded at an offset of 0x2000000. Add the 0x20000000 base address to the APPLICATION START ADDRESS and CERTIFICATE_START_ADDRESS.

15. Highlight the rtic project, right-click, and click Build Project.

16. To use Bootgen to create the image for the xapp1225-rtic project, enter Xilinx Tools > Create Zynq Boot Image.

17. To add the four partitions, follow the steps used for the hello.bin partition.

   Alternately, click Import from existing BIF file and browse to xapp1225-rtic/completed/rtic.bif.
18. With the Bootgen content as shown in Figure 5, click **Create Image** to write `rtic.mcs`.

![Figure 5: Use Bootgen to Create RTIC Image](image-url)
RTIC System Test

This section shows two methods to run a system test of the xapp1225-rtic reference design. The simplest test uses XMD commands. The second test runs the reference design from Quad SPI flash memory.

1. To view the system test results, set up a communication terminal such as Tera Term using a 115,200 baud rate.

2. To run the simple test, invoke XMD and run the following commands
   
   ```
   connect arm hw
   dow fsbl.elf
   run
   stop
   dow -data hello.bin 0x20000000
   dow rtic.elf
   con
   ```

3. To run the reference design from Quad SPI flash memory, set the ZC702 Mode Select pins to JTAG mode.

4. From SDK, click Xilinx Tools > Program Flash

5. To enter the Image File text box, browse to xapp1225-rtic/completed/rtic.mcs.

6. Type 0x0 in the Offset text box.

7. Click Program

8. On the ZC702 board, change the Mode Select pins to select the Quad SPI flash memory mode.

9. Power cycle the board.

Figure 6 shows the tail of the expected output from the system test. A RTIC is run periodically. In the system test, it is run three times, in five second intervals.

The complete rtic.log file is provided in the xapp1225-rtic/completed directory. The FSBL is compiled with the FSBL_DEBUG_INFO Build setting, so detailed information is provided on the loading of the partitions. The log file contains address location, size, and authentication information on each partition. Following this, the system test is run three times.
The reference design shows the basics of how to run the RTIC periodically. In an actual system, the period should be unending, and additional security measures should be used. When not used, the RTIC should be located in system memory, either in NVM or DDR. When the RTIC is scheduled to run, the RTIC should be authenticated.

The RTIC should run in the most privileged mode from OCM. If Trustzone is used, the RTIC should reside in the secure world.

**Security Considerations in Using the RTIC**

The reference design shows the basics of how to run the RTIC periodically. In an actual system, the period should be unending, and additional security measures should be used. When not used, the RTIC should be located in system memory, either in NVM or DDR. When the RTIC is scheduled to run, the RTIC should be authenticated.

The RTIC should run in the most privileged mode from OCM. If Trustzone is used, the RTIC should reside in the secure world.

**Conclusion**

This application note provides a method to check that there have not been any modifications to sections of system memory which contain code and/or static data. The run time integrity...
checker provides an effective method of detecting an attack on system memory which might occur during operation.

---

Reference Design

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

Table 1 shows the reference design matrix.

**Table 1: Reference Design Matrix**

<table>
<thead>
<tr>
<th>Parameter Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer name</td>
<td>Lester Sanders</td>
</tr>
<tr>
<td>Target devices</td>
<td>Zynq-7000 All Programmable 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>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>N/A</td>
</tr>
<tr>
<td>Test bench format</td>
<td>Verilog</td>
</tr>
<tr>
<td>Simulator software/version used</td>
<td>Vivado simulator</td>
</tr>
<tr>
<td>SPICE/IBIS simulations</td>
<td>N/A</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>Yes</td>
</tr>
<tr>
<td>Hardware verified</td>
<td>Yes</td>
</tr>
<tr>
<td>Hardware platform used for verification</td>
<td>ZC702, ZC706, or Zed evaluation board</td>
</tr>
</tbody>
</table>

---

**References**

1. Secure Boot of Zynq-7000 All Programmable SoC ([XAPP1175](#))
2. ZC702 Evaluation Board for Zynq-7000 XC7020 AP SoC ([UG850](#))
3. Zynq-7000 All Programmable SoC Software Developers Guide ([UG821](#))
4. Secure System Update of Zynq-7000 All Programmable SoC (XAPP1224)

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>10/23/2014</td>
<td>1.0</td>
<td>Initial Xilinx release.</td>
</tr>
<tr>
<td>10/24/2014</td>
<td>1.0.1</td>
<td>Corrected board number in [Ref 2] to ZC702.</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; 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.

© 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.

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 RELATIVE 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.