WP426 (v1.0) April 5, 2013

Secure Boot in the Zynq-7000 All Programmable SoC

By: Lester Sanders

The availability of increasingly powerful system-on-chip devices (SoCs) has led to unprecedented growth of embedded computing over the past few years. Embedded devices are popular largely because they enable mobility. They are more vulnerable than enterprise devices due to their portability and accessibility, making them more susceptible to loss or theft. With the increased capability of these devices and the value of the information in them, the need to protect them is more important than ever.

When evaluating these devices, two key modes — device boot and device operation — are security-related concerns. This white paper describes how secure operation of the Zynq®-7000 All Programmable SoC (Zynq-7000 AP SoC) is achieved, from the instant power is applied (secure boot) until the user applications are running (secure operation). The white paper also explains how this "chain of trust" is implemented to maintain security at all stages of system operations.

Zynq-7000 All Programmable SoC

Xilinx has manufactured FPGAs with embedded microprocessors since the introduction of the Virtex®-II Pro family. The Zynq-7000 AP SoC, however, is a new class of embedded device. Previous generations used the embedded components (processors and peripherals) as slave devices to the programmable logic. Zynq device families use the processing system as the master component, with the programmable logic as the slave device. The Zynq-7000 family is a processing system (PS) consisting of an ARM® Cortex™-A9 MPCore™ with IP peripherals and a programmable logic (PL) device based on the Xilinx 7 series FPGA families of products.

Building on the security of the 7 series FPGAs (with features such as bitstream authentication and decryption, environmental monitoring, and others), the Zynq-7000 AP SoC adds the asymmetric RSA-2048 algorithm as an additional highly flexible layer of authentication. For example, when the user desires only authentication to prevent code tampering, RSA can be used. The advantages of RSA authentication is that the device stores a (hash) of the public key, not a private key, on the device, and the key management is more flexible and secure. When both authentication and confidentiality are desired to prevent unauthorized access and loss of IP, the AES256 [Ref 1] /HMAC [Ref 2] engine can be used. When authentication before decryption is required, a combination of these methods can be implemented.

Threats to Embedded Devices

Common threats to embedded devices include IP theft, cloning, and malicious modification. The threats compromise the designer's system, IP, and data. While these threats affect different stakeholders in different ways, a secure system requires that they are addressed.

IP Theft

Theft of systems represents a significant threat, especially given the portable nature of many embedded systems. Once the adversary has access to the system, the system code can be reverse- engineered; then, portions of the IP can either be copied for direct resale, or the knowledge gained can be used to enhance the adversary’s own design.

Cloning

In some cases, rather than go to the expense of reverse-engineering a design, an adversary might simply copy the compiled design (located inside a NVM, DDR, or similar device) and mass-reproduce the system under a new name. It is a simple but effective attack. Although clearly illegal, the industry has struggled to prevent cloning.

Malicious Modification

Similar to the IP theft case, but with more nefarious and persistent intent, malicious modification of an existing design allows the adversary to steal not just IP but any information in the system at any time throughout the operating life of the system. This attack can happen at many different points of the system — from boot time to run time — on systems that are unprotected. Once inside the system, the adversary has control and can execute whatever actions are desired, like skimming ATM PINs, credit card information, classified data, etc.

For additional information on threats and specific types of attacks, read XAPP1084, *Developing Tamper Resistant Designs with Xilinx Virtex-6 and 7 Series FPGAs* and WP365, *Solving Today's Design Security Concerns*.

Zynq-7000 AP SoC Security Architecture

The principle security objective of the Zynq-7000 AP SoC is to lay a foundation of trust using integrity, confidentiality, and authentication from the moment power is applied to the time the user asserts control, and to provide a method for the user to maintain this trust. The Zynq-7000 AP SoC achieves this by using the PL's built-in Advanced Encryption Standard (AES-256)/Hashed Message Authentication Code (HMAC) engines, and the PS's RSA authentication capability. Figure 1 shows some of the subsystems used when securely booting a Zynq-7000 AP SoC. All security-related registers use redundant implementation to ensure fault tolerance.

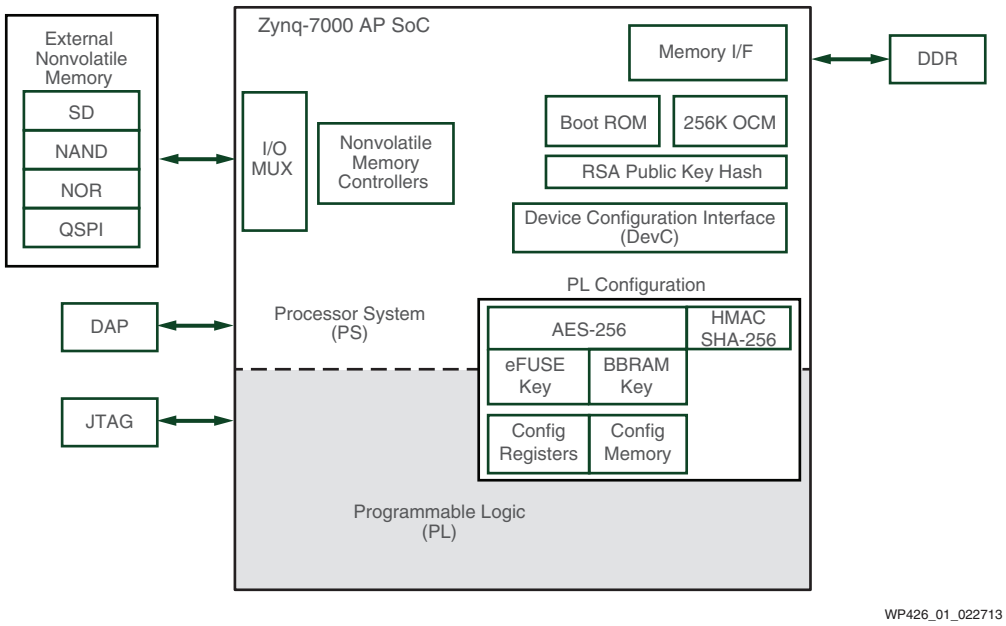

Figure 1: **Zynq-7000 AP SoC Security Architecture**

Some of the key features and their uses are:

- Nonvolatile Memory (NVM): Stores the object code for both the PS and PL, which is either Secure Digital, NAND, NOR, or QSPI flash memory
- DevC block: Efficiently moves the user's IP and handles the interface to the Decryption and Authentication blocks, when confidentiality and authentication are required
- 128K mask-programmed BootROM: Stores the initial boot code (BootROM code) and the RSA algorithm for authentication of the First Stage Boot Loader (FSBL)
- The 256K OCM: Provides secure storage of the FSBL and user code
- Either the One-Time Programmable (OTP) eFUSE array *or* the battery-backed RAM (BBRAM): Stores the 256-bit AES decryption key
- One-Time Programmable (OTP) eFUSE array: Stores the 256-bit hash of the RSA-2048 public key

The high level of integration of the security features and the large number of silicon peripherals, including Ethernet and PCIe® controllers, provide additional security against attacks. Interfaces that were previously exposed to the adversary are now fully contained within the Zynq-7000 AP SoC and under control of authenticated users.

Secure Boot

A typical boot flow is shown in Figure 2. The execution of code stored in the Zynq-7000 AP SoC BootROM is non-optional.

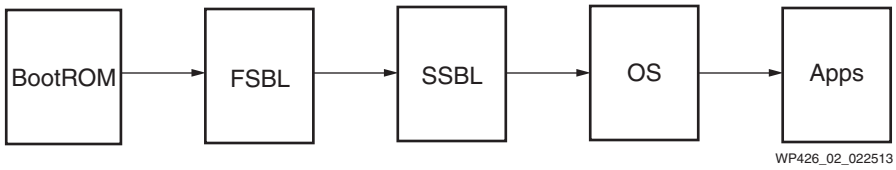

Figure 2: **Secure Boot Chain of Trust**

Zynq-7000 AP SoCs operate securely from when power is initially applied until such time the user chooses to relax security. As shown in Figure 2, booting typically requires multiple phases. For a secure and trusted boot, each stage of the process must hand off security responsibility to the next successive stage without compromising security or trust. With RSA authentication of each partition, a chain of trust is provided to the end of boot handoff.

The BootROM code sets the root of trust by securing all access points, verifying the integrity and operational state of the Zynq-7000 AP SoC, ensuring all operations in the process are valid and expected operations, and then loading the FSBL. The BootROM code then hands off to the FSBL in a secure state. The subsequent boot loaders are required to maintain the chain of trust both in operation and in subsequent handoffs. Since both the FSBL and Second Stage Boot Loader (SSBL) are configured by the designer, the designer controls all boot phases after the BootROM code handoff.

Xilinx provides a template for the FSBL to aid the developer in the generation of secure boot loaders. A common SSBL is U-Boot; a common OS is Linux, Android, or one of the systems produced by Xilinx Ecosystem Partners such as VxWorks®, Integrity®, or PikeOS®.

BootROM Code

The BootROM code executes code stored in masked programmed, non-readable, or modifiable ROM. The only external inputs affecting BootROM code execution are the boot mode pins and the user selected encryption mode specified in the image boot header. The primary purpose of the BootROM code is to bring up the Zynq-7000 device into a known good and secure state before the user takes over control. Upon power-up, immediately following hardware implemented Built-In-Self-Test (BIST) device checks, the BootROM code reads and validates the boot image of the image stored in the NVM. The boot header (BH) contains information about that image: identification, encryption, encryption key source (if encrypted), length, location, and checksum. After the FSBL is stored in OCM, a number of security checks are performed by the BootROM code. They include, but are not limited to:

- Built-in redundancy for security-critical decisions to increase the fault tolerance of the device
- CRC-based integrity check on BootROM
- Watchdog timers to ensure that reset, initialization, and PLL operations start up and run as expected
- Verification that there are no discrepancies between parameters specified in the eFUSE array and the BH
- Verification that the boot mode is legal for the set of parameters identified
- Unsecure boots/configurations are prohibited following a secure boot without full power cycle or POR

If any of these checks fail, the BootROM code shuts down the PL, clears the content of both the PS and PL memory and registers, and enters a secure lock-down state. The only way to exit the secure lock-down state is to cycle power to the device or activate the power-on reset (POR) pin.

If the tests pass, the CPU enables access to the boot subsystem used in the next boot task, executes the boot task, and subsequently disables that access. Failure to find the FSBL places the device in unsecure mode, allowing for programming by JTAG.

If the user selects the RSA option (in eFUSE) to authenticate the FSBL, the BootROM code:

1. Moves the FSBL image into OCM
2. Verifies the CRC of the key space storing the hash of the RSA key
3. Reads the public key from external memory, runs it through SHA-256, and compares it against the value stored in the PS eFUSEs
4. Runs RSA authentication of the FSBL image
5. Sends the FSBL image to the PL's AES/HMAC engine, where it is decrypted and authenticated using the private keys
6. Securely stores the calculated signature in OCM, then compares against the signature that is securely embedded in the authentication certificate

Again, if any of these checks fail, the BootROM code shuts down the PL, clears the contents of both the PS and PL memory and registers, and enters a secure lock-down state.

Setting the Boot Flow

As stated previously, the Zynq-7000 devices boot securely until the user decides to change from a secure to a non-secure mode. The starting decision point is the initial load of the user FSBL. If the user encrypts the FSBL, the BootROM code boots securely and hands off to the FSBL securely. If the FSBL is *not* encrypted, then the AES 256/HMAC engine is disabled and the JTAG ports can be accessed. To enable the RSA authentication, the RSA Enable eFUSE must be programmed.

The Xilinx® design tools allow the designer to specify whether the software and bitstream partitions are to be authenticated using the public-key RSA algorithm and whether subsequent images are encrypted/authenticated using the AES/HMAC engine. The designer can also specify that a partition is unencrypted. This allows the designer to increase security by using both public- and private-key algorithms on a partition basis.

Alternatively, when configuration time is critical, designers can trade off security against boot time, because configuration speed is faster for unencrypted partitions than for encrypted partitions. A relatively large, open-source U-Boot or Linux image, for example, loads faster if the image is unencrypted. If any of the secure features are selected, then the initial FSBL, at a minimum, *must* be encrypted with AES-256 and authenticated with the HMAC algorithm.

Using Lock Registers to Isolate Boot Subsystems

Isolation and access control are fundamental tenets of computer security. Although it is unlikely that an adversary can access PS/PL subsystems, secure protocols mandate locking boot subsystems that are not in use to provide additional protection.

Several methods can be used to lock boot subsystems. The lock registers are accessible by the BootROM code, FSBL, SSBL, and software applications. Until the PS/PL clearing and error checks are done, critical boot subsystems, including the DevC, JTAG, and PS-PL interface, are locked, prohibiting access by either a legitimate user or an adversary. At power-up, the Zynq-7000 device isolates the PS and PL regions. Since the PS and PL regions are isolated, an attack on the PL cannot provide an attack vector on the PS. Additionally, the PL is reset and verified reset as one of the initial checks performed by the BootROM code.

The AES/HMAC engine, key source, DAP, security, and single event upset (SEU) subsystems can also be locked. Depending on the boot mode, some boot subsystems are automatically locked, as in the case of a non-secure boot where the AES/HMAC engines are disabled and locked. The highest level lock mechanism is the eFUSE array. A lock in the eFUSE array cannot be changed — even with a power cycle.

AES/HMAC and RSA Protect Intellectual Property

To protect intellectual property (IP), Zynq-7000 AP SoCss provide confidentiality, integrity, and authentication. AES-256 (for confidentiality) and HMAC SHA-256 (for authentication) are used in combination where confidentiality, integrity, and authentication are desired. Should the user require public key authentication, RSA-2048 can be used to sign the user's code. A combination of the two methods can also be employed. Use of these methods prevents any adversary from reading or modifying any user code or IP. In all cases, the keys for this protection are determined and programmed by the user.

Zynq-7000 AP SoCs have two locations to store the AES decryption keys:

- Electrically programmable metal fuses (eFUSEs)
- Battery Backed Random Access Memory (BBRAM)

The keys can be accessed only via the JTAG port. Both key sources can be locked down such that JTAG no longer has access to the keys.

The BBRAM key is reprogrammable and can be erased. Erasing the key is a common countermeasure after detecting a tamper event. The OTP eFUSE key cannot be erased.

A 256-bit hash of the 2,048-bit RSA key is stored in the eFUSEs. Since this is a hash of the public key, if an adversary gains access, it is not as problematic as is access to a private key. With RSA authentication, the key used can be changed on a partition basis, and during the life cycle of the device. This reduces not only the vulnerability of the key but the information the key is protecting. If RSA is enabled, the BootROM code does an integrity check of this hash before loading and using it.

Zynq-7000 AP SoC On-Chip Secure Storage

Secure storage is required for secure boot and is useful for security in all phases of operation. In this context, the Zynq-7000 AP SoC has two types of secure storage:

- Internally isolated memory
- Physically secure memory

Internally Isolated Memory

When located in on-chip memory, data and code can be stored in unencrypted format. The Zynq-7000 device contains four forms of on-chip storage, inaccessible to an attacker, to enable the boot loaders to boot securely. These are:

- BootROM
- OCM
- AXI block RAM
- PL configuration memory

Additionally, using the cache lockdown feature, the 512K L2 cache can be directly used as on-chip memory rather than pure cache.

The OCM is used for secure storage of the FSBL. Since the OCM is RAM, it is reusable after the FSBL hands off control to the SSBL or application code. The AXI block RAM in the PL is also internally isolated memory. The address/data lines of the OCM and AXI block RAM do not connect to external pins. Exterior access to these internal memories when the system is active is available only if the user chooses to create connections when the system is built. If access to these storage blocks needs to be restricted internally — from direct memory access controllers (DMACs), for example — the ARM TrustZone® technology can provide isolation from unauthorized users.

Sensitive data stored off-chip in NVM should be encrypted. The FSBL and PL bitstreams are stored in encrypted format in flash or SD memory. The DMAC in the DevC subsystem moves these partitions through the AES/HMAC engine to the OCM and PL configuration memory, respectively.

Application code is generally moved to external DDR. While the application code is object code, an adversary can monitor the PCB address/data lines. To increase the security of the application code, it should be stored encrypted. It can then be decrypted inside the Zynq-7000 device and moved to internal storage in either OCM or AXI_BRAM. In addition to providing secure storage, the OCM is the fastest

memory available to the CPUs. The block RAM size is selectable depending on device size, while the OCM is 256K.

Physically Secure Memory

For private keys, the user can use either nonvolatile eFUSE or BBRAM storage. Once programmed and secured, there is no access to the keys even to a trusted user. For the BBRAM space, any attempt to read or re-write causes a clearing of the stored key and resets the PL contents.

Debug Features

Device debug capabilities are a necessity for user integration and test, yet they are also the source of the most common exploits to SoCs. Enabling debug functionality while not compromising security requires careful control using eFUSEs and the SLCR/DEVC registers. When active, the Zynq-7000 AP SoC debug ports provide an adversary access to I/Os, registers, and memory. The user must disable JTAG ports for fielded systems when not in use. On a secure boot, the BootROM code disables the PL JTAG port prior to handoff to the FSBL.

The ARM Debug Access Port (DAP) and FPGA JTAG Test Access Port (TAP) are used by designers to debug their system (C or HDL). The ARM DAP is used to debug the PS, while the FPGA TAP is used to debug the PL. Fundamentally, both are JTAG chains. The chains can be configured as independent chains or can be concatenated into a single chain. Figure 3 shows the cascade and independent modes of the JTAG chain(s).

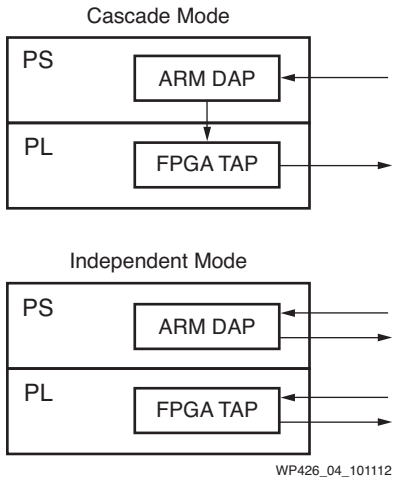

Figure 3: **Debug–JTAG Chain**

As stated previously, the Zynq-7000 device starts out in secure mode. Consequently, the ARM DAP is disabled until such time an authenticated user opens this port up or boots the device in unsecure mode. The DAP/FPGA TAP enable registers are not locked by default but can be locked by the FSBL or any following software. This disable deactivates the ARM DAP and subsequently breaks the chain between the ARM DAP and FPGA TAP. As with all security related features of the Zynq-7000 device, this disable feature is implemented redundantly.

The Zynq-7000 AP SoC Design for Test (DFT) mode is a Xilinx internal test feature and represents a potential threat to design security as it bypasses the BootROM code. When enabled, the internal memory and configuration are completely cleared, and the

device is placed in an unsecure mode, *disabling all security features*. This is a valuable feature for device test and for debugging problems with customer returns.

While Zynq-7000 AP SoCs have been designed to ensure these test features do not pose vulnerability, the user can permanently disable them. When debug capabilities are no longer needed and device security is paramount, the user can blow redundant eFUSEs to permanently disable the device's test capabilities.

Conclusion

The expanded use of embedded devices is inevitably followed by a growing threat to them and the information they contain. With Zynq-7000 AP SoC security features, embedded designers can now address the associated security concerns. While there are devices on the market that offer security features, few build a foundation of trust from the moment power is applied.

The Zynq-7000 AP SoC provides all the on-chip hardware resources necessary to boot securely and set the initial foundation of trust. From this foundation a system can be built to extend this chain of trust for the life cycle of the embedded device. With its fully contained secure storage, peripherals, and programmable logic, the Zynq-7000 AP SoC offers the developer a secure platform with the shortened development time of a programmable device. Adding a rich suite of IP offered by Xilinx and a strong ecosystem of secure solutions, the Zynq-7000 AP SoC is a clear advantage to the developer of secure systems.

References

1. FIPS PUB 197, *Advanced Encryption Standard (AES)*, Federal Information Processing Standards Publications, National Institute of Standards and Technology
2. FIPS PUB 198-1, *The Keyed-Hash Message Authentication Code (HMAC)*, Federal Information Processing Standards Publications, National Institute of Standards and Technology

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
04/05/13	1.0	Initial Xilinx release.

Notice of Disclaimer