230409.FF-A

Created: 2023-04-15 11:17:26

Questions

- Ch 2 - Introduction
  - The below text are chapter 2 of “Arm Firmware Framework for Arm A-profile”
  - What is “Arm A-profile”? (2023/4/11)
  - Does “Arm Firmware Framework for Arm A-profile” document is only about Cortex-A? (2023/4/11)
- Ch 3 - Software Architecture
  - The below text are chapter 3 of “Arm Firmware Framework for Arm A-profile”
- Ch 3.1 - Isolation boundaries
  - The below text are chapter 3.1 of “Arm Firmware Framework for Arm A-profile”. Please summarize it.
- Ch 3.2 - Partitions
  - The below text are chapter 3.2 of “Arm Firmware Framework for Arm A-profile”. Please summarize it.
- Ch 3.3 - Partition manager
  - The below text are chapter 3.3 of “Arm Firmware Framework for Arm A-profile”. Please summarize it.
  - So SPM and Hypervisor are partition managers. Right?
  - What is VMSA?
  - Is VMSA a software or a hardware?
- Ch 3.4 Example configurations
  - The below text are chapter 3.4 of “Arm Firmware Framework for Arm A-profile”. Please summarize it.
  - Can SPM see SP0 and SP1?
  - Can SP0 or SP1 see SPM?
  - Is TrustZone isolation boundary physical isolation or logical isolation?
- Ch 4.1 - SPM architecture
  - The below text are chapter 4.1 of “Arm Firmware Framework for Arm A-profile”. Please summarize it.
  - What is SPMC in the document?
  - Is SPMC of TrustZone?
- FF-A, TF-A, and TrustZone
  - What is TF-A? (2023/4/10)
  - Please draw the relationship among TrustZone, TF-A, and FF-A. (2023/4/10)
  - What is relationship between FF-A and TF-A? (2023/4/10)
  - Could I say that TF-A refers FF-A? (2023/4/10)
  - When to enable secondary cores? (2023/4/11)
  - Is TF-A persistent in memory at run-time? (2023/4/12)
  - What is SMCCC? (2023/4/13)
  - What is SPL? (2023/4/13)
  - What is fip.bin? (2023/4/13)

Q & A

Ch 2 - Introduction

Question: The below text are chapter 2 of “Arm Firmware Framework for Arm A-profile”

Answer: (Hide)

Question: What is “Arm A-profile”? (2023/4/11)

Answer:

The Arm A-profile is a family of Arm CPU architectures that are designed for high-performance and power-efficient applications, including mobile devices, consumer electronics, servers, and networking infrastructure. The A-profile CPUs are based on the Armv7-A and Armv8-A architectures, which support both 32-bit (AArch32) and 64-bit (AArch64) execution states. The A-profile processors provide a wide range of performance and power options to suit various use cases and include popular cores like Cortex-A series (e.g., Cortex-A72, Cortex-A57, Cortex-A53, etc.).

Question: Does “Arm Firmware Framework for Arm A-profile” document is only about Cortex-A? (2023/4/11)

Answer:

The “Arm Firmware Framework for Arm A-profile” document is focused on the Arm A-profile architecture, which includes Cortex-A series processors. However, it is not limited to Cortex-A processors only. The document provides guidelines and specifications for developing firmware for Arm A-profile processors in general, including Cortex-A series processors and potentially other A-profile processors developed by Arm or its partners. It aims to ensure a consistent approach to firmware development, security, and functionality across different Arm A-profile implementations.

Ch 3 - Software Architecture

Question: The below text are chapter 3 of “Arm Firmware Framework for Arm A-profile”

Answer: (Hide)

Ch 3.1 - Isolation boundaries

Question: The below text are chapter 3.1 of “Arm Firmware Framework for Arm A-profile”. Please summarize it.

Answer: (Hide)

Ch 3.2 - Partitions

Question: The below text are chapter 3.2 of “Arm Firmware Framework for Arm A-profile”. Please summarize it.

Answer: (Hide)

Ch 3.3 - Partition manager

Question: The below text are chapter 3.3 of “Arm Firmware Framework for Arm A-profile”. Please summarize it.

Answer: (Hide)

Question: So SPM and Hypervisor are partition managers. Right?

Answer:

Yes, that’s correct. In the context of the Arm Firmware Framework for Arm A-profile, the Secure Partition Manager (SPM) operates in the Secure world, while the Hypervisor operates in the Normal world. Both SPM and Hypervisor are partition managers responsible for creating and managing the physical isolation boundaries of partitions, assigning resources, and enabling inter-partition communication.

Question: What is VMSA?

Answer:

VMSA, or Virtual Memory System Architecture, is a part of the Arm architecture that defines the memory management and address translation mechanisms. It is responsible for managing the address spaces and memory protection features in Arm systems. VMSA allows for the creation of separate virtual address spaces for each process, providing isolation between processes and the operating system. In the context of the Arm A-profile, the Hypervisor uses the virtualization extension in VMSA to create physical isolation boundaries between different partitions or virtual machines.

Question: Is VMSA a software or a hardware?

Answer:

VMSA (Virtual Memory System Architecture) is not a specific piece of software or hardware, but rather a part of the Arm architecture specification that defines how memory management and address translation should be implemented in Arm-based systems. VMSA provides guidelines for designing hardware and software components that handle memory management, address translation, and access control, allowing for the creation of separate virtual address spaces and ensuring proper isolation between processes and the operating system.

Ch 3.4 Example configurations

Question: The below text are chapter 3.4 of “Arm Firmware Framework for Arm A-profile”. Please summarize it.

Answer: (Hide)

Question: Can SPM see SP0 and SP1?

Answer:

Yes, the Secure Partition Manager (SPM) can “see” and manage both SP0 and SP1. The SPM is responsible for creating and managing the physical isolation boundaries of Secure Partitions (SPs) like SP0 and SP1. It uses partition manifests to assign resources, initialize the partitions, and enable discovery of their services. Additionally, the SPM implements FF-A ABIs (Application Binary Interfaces) to enable inter-partition communication for access to partition services. So, while the SPM isolates SP0 and SP1 from each other, it still has visibility and control over both partitions.

Question: Can SP0 or SP1 see SPM?

Answer:

In the ARM Firmware Framework, SP0 and SP1 can communicate with the Secure Partition Manager (SPM) through well-defined interfaces, such as FF-A ABIs (Application Binary Interfaces). While the SPM is physically isolated from SP0 and SP1 in scenarios where they are physical partitions, they can still interact with the SPM using these interfaces for communication, service discovery, and memory management.

However, it is important to note that SP0 and SP1 cannot access the internal workings or the memory of the SPM directly. They can only communicate with the SPM through the defined interfaces, ensuring that the isolation between them is maintained.

Question: Is TrustZone isolation boundary physical isolation or logical isolation?

Answer:

TrustZone isolation boundary is a form of physical isolation. TrustZone is a security extension in ARM architecture that separates the processor’s resources, such as memory and peripherals, into two distinct security domains: the Secure world and the Non-secure world. By utilizing hardware-based isolation, TrustZone ensures that sensitive data and functions in the Secure world are protected from unauthorized access by software running in the Non-secure world.

Ch 4.1 - SPM architecture

Question: The below text are chapter 4.1 of “Arm Firmware Framework for Arm A-profile”. Please summarize it.

Answer: (Hide)

Question: What is SPMC in the document?

Answer:

In the document “Arm Firmware Framework for Arm A-profile,” the SPMC (Secure Partition Manager Core) is a component of the Secure Partition Manager (SPM) responsible for:

1. Initializing and isolating Secure Partitions (SPs) at boot time.
2. Ensuring inter-partition isolation during runtime.
3. Managing inter-partition communication between Secure Endpoints (S-Endpoints) and Non-Secure Endpoints (NS-Endpoints).

The SPMC can co-reside with the SPM Dispatcher (SPMD) in EL3 or reside in adjacent exception levels, such as S-EL1 or S-EL2. It is a part of the Trusted Computing Base and has access to the entire physical address space.

Question: Is SPMC of TrustZone?

Answer:

The SPMC (Secure Partition Manager Core) is part of the Arm Firmware Framework for Arm A-profile, which is built on top of the Arm TrustZone technology. TrustZone provides the underlying hardware support for isolation and security, while the SPMC is responsible for managing partitions and their communication within the Secure world enabled by TrustZone. So, while the SPMC is not TrustZone itself, it is a component that utilizes and operates within the TrustZone technology.

FF-A, TF-A, and TrustZone

Question: What is TF-A? (2023/4/10)

Answer:

TF-A, or Trusted Firmware-A, is a reference implementation of secure world software for Arm A-profile processors. It provides a platform for implementing security features and building trusted execution environments in devices that use Armv8-A processors, such as smartphones, tablets, and other embedded systems.

TF-A includes implementations of various Arm security technologies, such as Secure Boot, Secure Monitor (for switching between secure and non-secure worlds), and runtime services. The project is open-source, allowing developers to use and modify the code to develop secure firmware solutions that meet the requirements of their specific applications.

Question: Please draw the relationship among TrustZone, TF-A, and FF-A. (2023/4/10)

Answer:

TrustZone, TF-A, and FF-A are technologies and frameworks related to the security aspects of Arm-based systems. Here’s a simple representation of their relationships:

```
+-----------------+
|   TrustZone     |
|  (Technology)   |
| +-------------+ |
| |    TF-A     | |
| | (Firmware)  | |
| +-------------+ |
| +-------------+ |
| |    FF-A     | |
| |  (Framework)| |
| +-------------+ |
+-----------------+
```

1. TrustZone is a hardware-based security technology built into Arm processors. It provides a secure execution environment by creating two separate worlds: a Secure World and a Non-secure World. TrustZone ensures that the Secure World's data and code are protected from unauthorized access.
2. Trusted Firmware-A (TF-A) is an open-source reference implementation of secure world software for Arm A-profile processors. It leverages TrustZone technology and provides secure boot, runtime services, and a secure monitor for switching between the secure and non-secure worlds.
3. Firmware Framework for Armv8-A (FF-A) is a software architecture specification that aims to standardize the way partitions, services, and communication are managed in Arm A-profile systems. It defines the interactions between the Secure Partition Manager (SPM), partitions, and the Non-secure world. FF-A can be used in conjunction with TrustZone and TF-A to build a complete secure software solution.

Question: What is relationship between FF-A and TF-A? (2023/4/10)

Answer:

The relationship between FF-A (Firmware Framework for Armv8-A) and TF-A (Trusted Firmware-A) is that FF-A provides a standardized software architecture specification for managing partitions, services, and communication on Arm A-profile systems, while TF-A is an open-source reference implementation of secure world software that leverages TrustZone technology.

TF-A can be used as a base for implementing the secure world software stack on Arm A-profile systems, and it can be extended or adapted to conform to the FF-A architecture. By aligning the TF-A implementation with the FF-A specification, developers can achieve a more modular, maintainable, and interoperable software stack that can better address the needs of secure systems.

In summary, the relationship between FF-A and TF-A is complementary, with FF-A providing the architectural guidelines, and TF-A serving as a reference implementation that can be tailored to adhere to those guidelines.

Question: Could I say that TF-A refers FF-A? (2023/4/10)

Answer:

Yes, you can say that TF-A refers to FF-A, as TF-A can be extended or adapted to conform to the FF-A architecture. The FF-A provides the architectural guidelines for secure systems, while TF-A serves as a reference implementation that can be tailored to adhere to those guidelines. By aligning the TF-A implementation with the FF-A specification, developers can achieve a more modular, maintainable, and interoperable software stack for secure systems.

Question: When to enable secondary cores? (2023/4/11)

Answer:

Secondary cores are typically enabled after the primary core has successfully initialized the system and started the operating system. The exact process for enabling secondary cores can vary depending on the hardware and operating system being used. Here is a general outline of when secondary cores are enabled:

1. Power on the system, and the primary core initializes hardware components and executes the bootloader (e.g., Trusted Firmware-A for Arm-based systems).
2. The bootloader loads and starts the operating system on the primary core.
3. The operating system initializes its kernel, essential components, and device drivers on the primary core.
4. After the primary core has completed its initialization, the operating system enables the secondary cores. This process is often referred to as “core bring-up” or “CPU hotplug.”
5. The operating system assigns tasks or processes to the secondary cores, allowing them to participate in executing applications and system processes.

It is important to note that the specific process for enabling secondary cores depends on the hardware platform, firmware, bootloader, and operating system in use. Some systems may require additional configuration or software support to enable secondary cores.

Question: Is TF-A persistent in memory at run-time? (2023/4/12)

Answer:

Yes, the Trusted Firmware-A (TF-A) is typically persistent in memory at run-time. It resides in the secure memory region and provides various services, such as Secure Monitor Call (SMC) handling, Power State Coordination Interface (PSCI) implementation, and other platform-specific services. Since it acts as a secure monitor, it must remain present in memory during the system’s operation to handle any secure world requests or transitions between secure and non-secure worlds.

Question: What is SMCCC? (2023/4/13)

Answer:

The SMCCC (Secure Monitor Call Calling Convention) is a standardized calling convention for ARM-based systems, which defines how Secure Monitor Calls (SMCs) are used to request services from a secure monitor, such as Trusted Firmware-A (TF-A). SMCCC provides a consistent interface for communication between the non-secure world (e.g., an operating system) and the secure world (e.g., a trusted execution environment or firmware). This standardized interface allows for better interoperability between different software components and platforms in ARM-based systems.

Question: What is SPL? (2023/4/13)

Answer:

SPL (Secondary Program Loader) is a term used in the context of embedded systems and bootloaders. It is a small, lightweight bootloader that runs before the primary bootloader and is responsible for initializing essential hardware components, such as clocks, memory, and peripherals. After performing the required initialization, the SPL loads the primary bootloader (e.g., U-Boot) into memory and transfers control to it, allowing the primary bootloader to continue the boot process and eventually load the operating system. The use of an SPL can help improve boot time, reduce the complexity of the primary bootloader, and provide a uniform interface for different hardware platforms.

Question: What is fip.bin? (2023/4/13)

Answer:

`fip.bin` (Firmware Image Package) is a binary file used in Arm Trusted Firmware-A (TF-A) for packing multiple firmware images into a single file. It is used as a part of the boot process in Arm-based systems, particularly in the context of Armv8-A architecture.

The `fip.bin` file typically contains a collection of firmware images, such as:

1. BL1 (First Level Bootloader): Responsible for initial platform setup and loading BL2.
2. BL2 (Second Level Bootloader): Responsible for loading BL3x images.

3. BL3-1 (EL3 Runtime Firmware): Provides runtime services, such as PSCI (Power State Coordination Interface), to the lower ELs (Exception Levels).
4. BL3-2 (Trusted OS Firmware): The Secure-EL1 Payload, typically the TrustZone-based OS or a secure monitor.
5. BL3-3 (Non-Trusted Firmware): The Non-secure-EL1 Payload, usually the main OS bootloader like U-Boot or GRUB.

The `fip.bin` file is created using the `fiptool` utility provided by Arm Trusted Firmware. The tool combines the various firmware images into a single binary, which can then be loaded and executed by the platform's bootloader during the boot process. This approach simplifies firmware management and distribution while also enabling the bootloader to locate and authenticate the required images more easily.