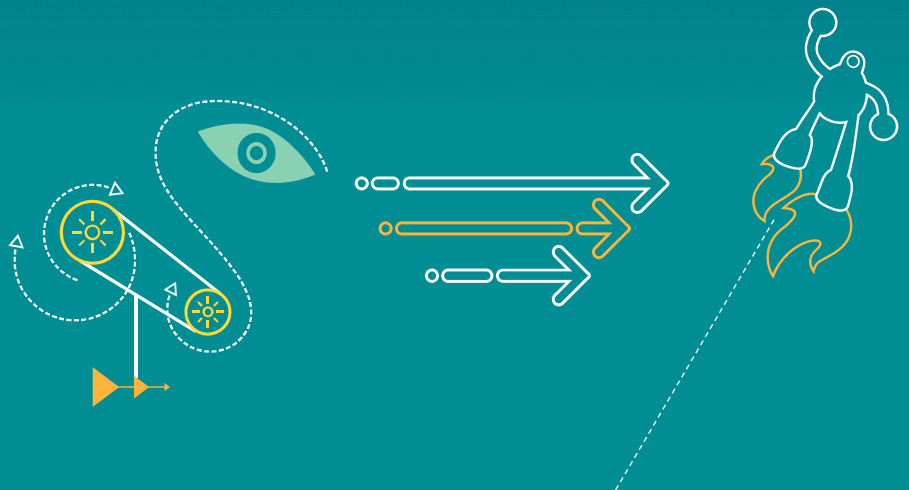
SDM670 启动和 CoreBSP 架构概述

Qualcomm Technologies, Inc.



80-PD126-2SC 版本 A

机密和专有信息 – Qualcomm Technologies, Inc.

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

修订记录

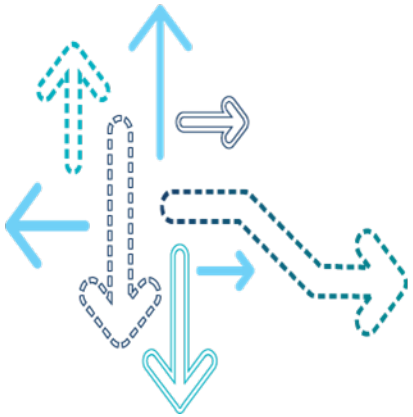
版本	日期	说明
A	2017 年 8 月	初始版本

目录

目标

在本演示文稿结束时，您将了解 SDM670 芯片组的启动架构。

启动架构



处理器启动地址

- 下表列出了 SDM670 芯片组中各处理器的类型和启动内存：

子系统	处理器	启动内存
应用程序	Qualcomm® Kryo™ CPU 360 （两个大核 + 六个小核）	APSS 启动 ROM
实时响应系统	实时响应处理器 (AOP) ARM Cortex-M3	AOP 代码 RAM
系统硬件资源管理器 (SHRM)	定制处理器	SHRM 代码 RAM
Modem	Qualcomm® Hexagon™ DSP	Modem 启动 ROM
视频核心	ARM9	LPDDR4x
LPASS	Hexagon DSP	LPDDR4x
Compute	Hexagon DSP	LPDDR4x
摄像头 ICP	ARM Cortex-A5	LPDDR4x

启动特性 – SDM660 和 SDM630 对比 SDM670

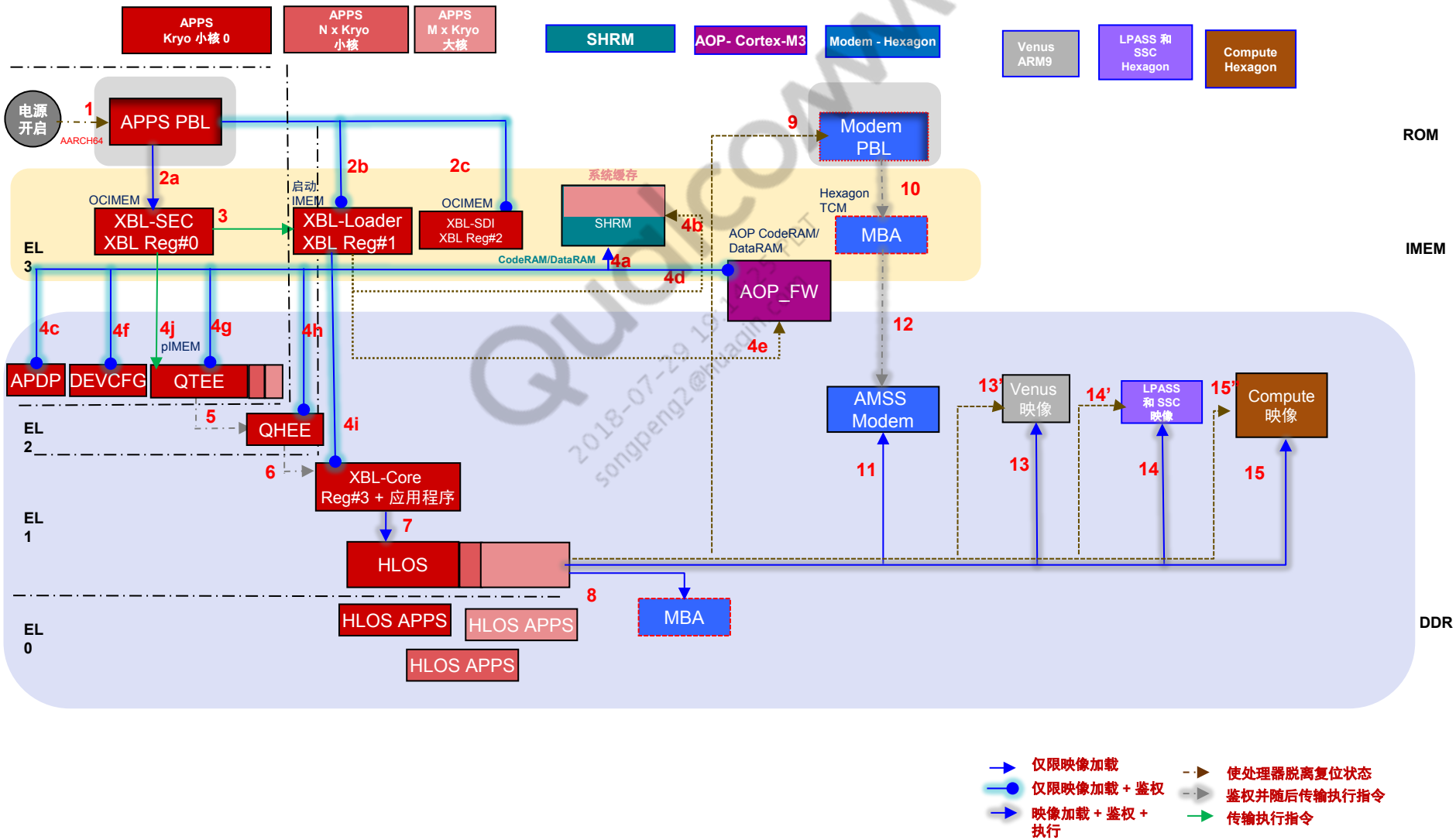
特性	SDM660 和 SDM630	SDM670
启动处理器	Kryo 小核 0	Kryo 小核 0
CPU	Kryo 200 （4 个大核+ 4 个小核）	Kryo 360 （2 个大核+ 6 个小核）
内存	2×16 位 LPDDR4/4x 1866 MHz（双通道，非 POP） eMMC 5.1、SD 3.0、UFS 2.0（单通道）；最大 DDR 大小为 8 GB **注：** 测试 DDR 大小为 6 GB	2×16 位 LPDDR4x 1866 MHz（双通道，非 POP） eMMC 5.1、SD 3.0、UFS 2.1 gear 3（单通道）；最大 DDR 大小为 8 GB **注：** 测试 DDR 大小将在本演示文稿的后续版本中更新
默认可启动设备	eMMC	eMMC GPIO_133 的 UFS 被拉高。有关详细信息，参见“启动选项”幻灯片
UFS 和 eMMC 启动	▪ HS G1。仅限嵌入式 UFS ▪ eMMC 5.1	▪ UFS 2.1 gear 3（单通道） ▪ eMMC 5.1
通过 QSPI_NOR 启动	通过主启动加载程序 (PBL) 启动	不支持
通过 SD 卡启动	▪ 仅在 PBL 中受支持 – 不支持 XBL 或统一可扩展固件接口 (UEFI) ▪ MCI（传统）支持	▪ 仅在 PBL 中受支持 – 不支持 XBL 或 UEFI ▪ SDHCi 支持
可扩展启动加载程序安全 (XBL-SEC)	▪ XBL-SEC Reg#0 使用 OCIMEM ▪ 支持二进制发布和安全保护设置	XBL-SEC Reg#0 使用 OCIMEM。更多详细信息，参见 *SDM670 冷启动流程图*幻灯片
XBL-Loader	▪ XBL-Loader Reg#1 – L2 TCM、VMEM、OCIMEM 和 DDR ▪ 加载并鉴权 Qualcomm® Trusted Execution Environment (QTEE)、Qualcomm Hypervisor Execution Environment (QHEE)、RPM_FW 和 XBL 核映像	▪ XBL-Loader Reg#1 – 启动 IMEM*、OCIMEM 和 DDR ▪ 加载并鉴权 QTEE、QHEE、AOP_FW、XBL-Core 和 SHRM 映像
XBL 系统调试映像 (XBL-SDI)	XBL-SDI Reg#2 – OCIMEM	▪ XBL-SDI Reg#2 – OCIMEM ▪ SDI 可启用第一轮操作定时器，以免出现 SDI 卡断等问题

注： * 对于 SDM670 芯片组，通过 Modem DSP TCM（而非 CPU）来支持启动 IMEM，并且不支持将 L2 缓存作为 TCM。

启动特性 – SDM660 和 SDM630 对比 SDM670（续）

特性	SDM660 和 SDM630	SDM670
配置	▪ PMIC.elf – PMIC 配置设置 ▪ eCDT.bin – 平台 ID 和 DDR 配置	▪ xbl_config.elf 是 OEM 可配置设置的新映像 ▪ PMIC.elf 和 DDR 设置迁移到 XBL 配置 ▪ DDR 配置块 (DCB) 代替 CDT 用于配置 DDR 参数 ▪ eCDT.bin：CDT 仅用于平台配置
双重签名	适用于 XBL 和 TrustZone (TZ)	支持
启动版本	BOOT.XF.1.4	BOOT.XF.2.0
编译脚本	buildit.py	buildex.py

SDM670 冷启动流程图



SDM670 冷启动流程组件

组件	基于处理器	加载源	ZI/RW 分配位置	执行位置	功能
应用程序主启动加载程序 (APPS PBL)	Kryo 小核 0	–	启动 IMEM	ROM	▪ 建立 APPS 安全 RoT、APPS 安全启动和启动设备 ▪ 检查接口和紧急下载 (EDL) 模式支持 ▪ 将 XBL 段（XBL-Loader 和 XBL-SEC）加载到启动 IMEM 和 OCIMEM（系统 IMEM） ▪ 解密并鉴权 XBL 段
XBL-SEC	Kryo 小核 0	UFS、eMMC	OCIMEM	OCIMEM	▪ 在安全 EL3 中运行并设置访问保护单元 (xPU) ▪ 向在非安全 EL1 中运行的 XBL-Loader 提供安全关键型功能，比如鉴权和 pIMEM 初始化 ▪ 提供调试策略支持 ▪ 在运行时将执行指令传输到 EL3 中运行的 QTEE 映像

XBL	Kryo 小核 0	UFS、eMMC	启动 IMEM、OCIMEM、LPDDR4x	启动 IMEM	▪ 初始化内存子系统（总线、LPDDR4x、时钟和 CDT）
					▪ 通过 XBL-SEC 加载并鉴权 QTEE、QHEE、AOP_FW、XBL-Core 和 SHRM 映像
					▪ 支持映像防回滚保护
					▪ 支持系统驱动程序 – PMIC、USB、充电、发热检查和 DDR 定型
					▪ 提供复位调试支持（暖复位时）–
					  ▪ XBL-Loader 将 XBL-RAMDUMP 加载到 LPDDR4x
					  ▪ XBL-RAMDUMP 支持显示（支持字体）、通过 USB/Sahara 协议进行崩溃转储以及到存储器 (SD/UFS) 的崩溃转储

SDM670 冷启动流程组件（续）

组件	基于处理器	加载源	ZI/RW 分配位置	执行位置	功能
SHRM	定制处理器	UFS、eMMC	▪ SHRM DataRAM ▪ 系统缓存	▪ SHRM CodeRAM ▪ 系统缓存	▪ 为初始化和 DDR 定型提供启动时 DDR 频率切换支持 ▪ 提供运行时 DDR 电源管理（XO 关闭、深度睡眠和 DDR 频率切换） ▪ 提供系统缓存管理 ▪ 支持复位调试（DDR 退出自刷新状态）
实时响应处理器	ARM Cortex-M3	UFS、eMMC	▪ AOP DataRAM ▪ LPDDR4x	AOP CodeRAM	▪ 用作 AOP 固件 ▪ 处理硬件加速器中未涵盖的相关性、复杂聚合和极端情况
QTEE (TZ)	Kryo 小核或大核	UFS、eMMC	▪ OCIMEM ▪ pIMEM ▪ LPDDR4x	▪ OCIMEM ▪ LPDDR4x ▪ pIMEM	▪ 在安全 EL3 或 EL1 中运行 ▪ 等同于 TZ BSP ▪ 建立受信任运行时执行环境 ▪ 配置 xPU ▪ 支持熔丝驱动程序
QHEE (HYP)	Kryo 小核或大核	UFS、eMMC	LPDDR4x	LPDDR4x	▪ 在非安全 EL2 中运行 ▪ 配置 CPU 或 SMMU 第 2 阶段内存映射 ▪ 提供安全外设映像加载程序 (PIL) 支持
XBL-Core	Kryo 小核 0	UFS、eMMC	LPDDR4x	LPDDR4x	▪ 提供 HLOS 特有的多功能启动加载程序 ▪ 支持 UEFI 功能

SDM670 冷启动流程组件（续）

组件	基于处理器	加载源	ZI/RW 分配位置	执行位置	功能
HLOS	Kryo 小核或大核	UFS、eMMC	LPDDR4x	LPDDR4x	▪ 支持 HLOS 内核和应用程序 ▪ 在非安全 EL1 和 EL0 中运行
Modem 主启动加载程序（Modem PBL）	Modem - Hexagon	–	Hexagon TCM	ROM	▪ 建立 Modem 安全 RoT ▪ 将 MBA 从 DDR 加载到 Hexagon TCM ▪ 解密并鉴权Modem 启动鉴权程序 (MBA) 映像
MBA	Modem - Hexagon	UFS、eMMC	Hexagon TCM	Hexagon TCM	启用对 Modem 固件的安全鉴权和解密
AMSS Modem	Modem - Hexagon	UFS、eMMC	LPDDR4x	LPDDR4x	用作 Modem 固件映像
Venus、LPASS、SSC 和 Compute 映像	ARM9、Hexagon、ARM Cortex-M3 + Hexagon	UFS、eMMC	LPDDR4x	LPDDR4x	子系统映像 – Venus 固件、音频和传感器固件以及 Compute 固件 **注：** 摄像头进程开始时，摄像头驱动程序会加载摄像头（ARM Cortex-A5 或 ICP）固件

SDM670 冷启动流程

1. MSM™ 复位后，Kryo Silver 核心 0 会退出复位状态并执行 PBL
2. APPS PBL 会执行以下操作：
   - 初始化硬件（时钟等）
   - 初始化 CPU 缓存和 MMU
   - 根据启动选项配置检测启动设备
     - 默认启动选项为 UFS → SD → USB
     - 默认启动选项可由 EDL cookie 或 Force USB GPIO 覆盖

   2a. 将 XBL-SEC (Reg#0) 从启动设备加载到 OCIMEM 并进行鉴权

   2b. 将 XBL-Loader (Reg#1) 从启动设备加载到启动 IMEM 并进行鉴权

   2c. 将 XBL-SDI (Reg#2) 从启动设备加载到 OCIMEM 并进行鉴权

   - 跳转到 XBL-SEC
3. XBL SEC 在 EL3 模式下运行安全配置，然后在 EL1 模式下执行 XBL-Loader

SDM670 冷启动流程（续）

4. XBL-Loader 对硬件和固件映像执行以下操作：
   - 初始化 CPU 缓存和 MMU
   - 初始化启动设备
   - 初始化 XBL 配置
   - 初始化 PMIC 驱动程序
   - 初始化 DDR 并执行 DDR 定型（若适用）
   - 对 XBL-SEC 执行 SCM 调用，以初始化 pIMEM
   - 初始化时钟并根据时钟规划配置时钟频率
   - 初始化共享内存 (SMEM)；更新平台 ID 和 RAM 分区表

   4a. 从启动设备加载 SHRM 映像并进行鉴权

   4b. 使 SHRM 退出复位状态

   4c. 从启动设备加载 APPS 调试策略 (APDP) 映像并进行鉴权。如果已设置 DLOAD cookie，则加载并鉴权 XBL RAMDUMP，然后跳转到 XBL RAMDUMP 收集崩溃转储

   4d. 从启动设备加载 AOP 映像并进行鉴权

   4e. 使 AOP 处理器退出复位状态

   4f. 从启动设备加载 TZ 设备配置 (DEVCFG) 映像并进行鉴权

   4g. 从启动设备加载 QTEE 映像并进行鉴权；若已存在映像，则从启动存储器加载 SEC.dat（熔丝熔断数据）

   4h. 从启动设备加载 QHEE 映像并进行鉴权

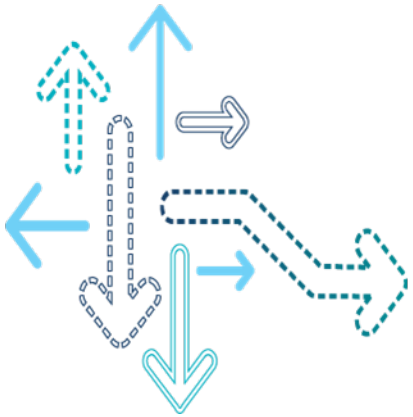
   4i. 从启动设备加载 XBL-Core 映像并进行鉴权

   4j. 对 XBL-SEC 执行 SCM 调用，以跳转到 QTEE 冷启动

SDM670 冷启动流程（续）

5. QTEE 建立安全环境并执行 QHEE 映像
6. QHEE 执行 XBL-Core（或 XBL region #3），然后 XBL-Core 安装并运行 UEFI APPS（ABL 固件区块(FV)）
7. Linux 加载器应用程序（ABL FV 的组成部分）通过已验证的启动方法来加载 HLOS 内核并进行鉴权
8. HLOS 中的 PIL 驱动程序将 Modem 固件映像和 MBA 映像加载到 DDR，并根据需要配置时钟和电源轨
9. HLOS 中的 PIL 驱动程序使 Hexagon DSP Modem 退出复位状态
10. Modem PBL 将 MBA 从 DDR 复制到 Modem TCM、对 MBA 进行鉴权，然后跳转到 MBA 映像
11. HLOS 通过 PIL 将 AMSS Modem 映像加载到 DDR
12. MBA 对 Modem 映像进行鉴权，然后跳转到 Modem
13. HLOS 中的 PIL 驱动程序加载 Venus 并配置时钟和电源轨。HLOS 中的 PIL 驱动程序执行 SCM 调用来请求安全 PIL 驱动程序、鉴权 Venus 并使其退出复位状态
14. HLOS 中的 PIL 驱动程序加载 LPASS 和 SSC 固件，然后执行 SCM 调用来请求安全 PIL 驱动程序、鉴权固件并使其退出复位状态
15. HLOS 中的 PIL 驱动程序加载 Compute 映像、执行 SCM 调用来请求安全 PIL 驱动程序、鉴权固件并使其退出复位状态

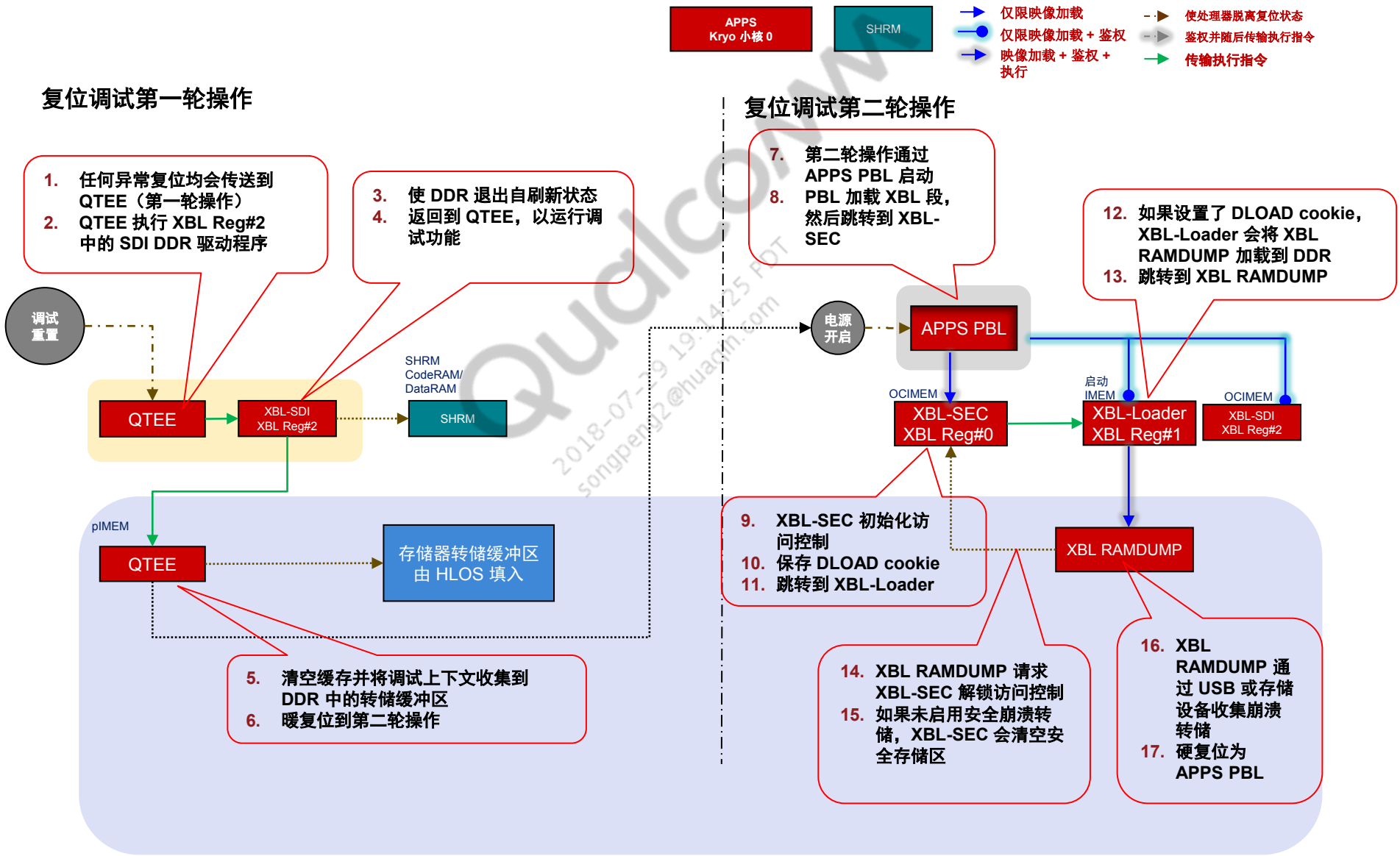
复位调试



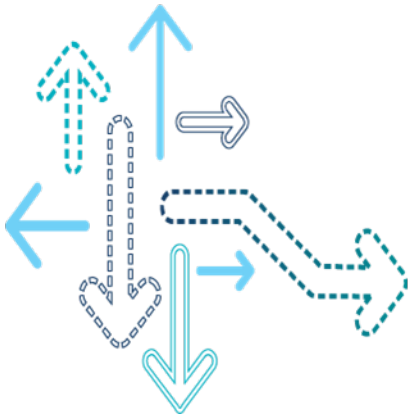
SDI

- 不存在独立的 SDI 映像 – DDR 驱动程序的一小部分 SDI 逻辑位于 XBL Reg#2 中，其余逻辑则在 QTEE 映像中
- 冷启动过程中，QTEE 的 SDI 逻辑可对非量产设备进行复位调试
- 任何异常复位（异常复位、看门狗复位或热复位）都会完成复位调试序列，SDI 会收集上下文并将其刷入 DDR 缓冲区
- 如果 SDI 因卡断等问题无法在指定超时时间内完成第一轮操作，可启动第一轮操作定时器，将复位控制器配置为复位到第二轮操作

SDM670 复位调试流程



PBL 日志



SDM670 APPS PBL 错误日志格式

APPS PBL 的最后一个错误详情位于启动 IMEM 的 0 × 14806088 与 0 × 14806177 之间

```
/* Error log structure to store data describing error */
typedef struct boot_pbl_err_type
{
  uint32                pbl_err_code_start;
  uint64                pbl_err_details;
  uint32                timestamp;
  uint32                pbl_id;
  uint32                patch_id;
  const char*           filename;
  uint32                line_num;
  uint32                pbl_err_code_end;
} boot_pbl_err_type;
```

提供附加的错误相关信息

0×EFAABBCC

EF – 错误签名

AA – PBL 功能块 ID

参见“SDM670 APPS PBL 错误代码定义”幻灯片 (pbl_log_block_type)

BB – 功能块中的子错误

CC – 错误计数，从 0 × 1 开始

指示子错误功能块，分为两类：

- 可恢复：硬件稳定，错误代码从 0 × 1 到 0 × 7F
- 不可恢复：硬件不稳定，错误代码从 0 × 80 到 0 × FF

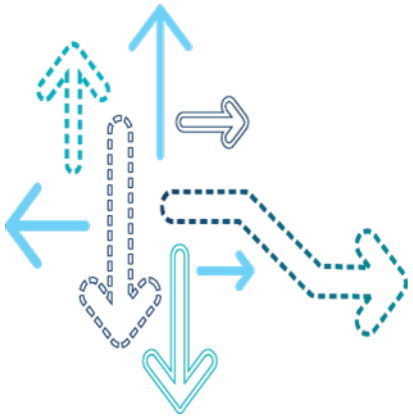
SDM670 APPS PBL 错误代码定义

```
typedef enum
{
  PBL_LOG_GENR           = 0x010000,
  PBL_LOG_PROC           = 0x020000,
  PBL_LOG_LOADER         = 0x030000,
  PBL_LOG_FUSE           = 0x040000,
  PBL_LOG_AUTH           = 0x050000,
  PBL_LOG_TIMER          = 0x060000,
  PBL_LOG_CLOCK          = 0x070000,
  PBL_LOG_SEC_HW         = 0x080000,
  PBL_LOG_SECBOOT        = 0x090000,
  PBL_LOG_SEC_IMG_AUTH   = 0x0A0000,
  PBL_LOG_SDCC           = 0x0B0000,
  PBL_LOG_SAHARA         = 0x0C0000,
  PBL_LOG_NAND           = 0x0D0000,
  PBL_LOG_PCIe           = 0x0E0000,
  PBL_LOG_UFS            = 0x0F0000,
  PBL_LOG_USB            = 0x100000,
  PBL_LOG_EXCEPTION      = 0x110000,
  PBL_LOG_ELF            = 0x120000,
  PBL_LOG_SPI            = 0x130000,
  PBL_LOG_EMM            = 0x140000,
  PBL_LOG_PRNG           = 0x150000,
  PBL_LOG_FORCE32BITS    = 0x7FFFFFFF /* To ensure it's 32 bits wide */
}pbl_log_block_type;
```

PBL 中使用的熔丝

熔丝位 (eFuse)	域	GPIO	说明
FAST_BOOT	OEM	4	▪ 指定设备必须启动为启动代码 ▪ 加速启动流程
VID	OEM	–	USB 供应商 ID
PID	OEM	–	USB 产品 ID
E_DLOAD_DISABLE	OEM	–	禁用紧急下载模式
PBL_USB_TYPE_C_DISABLE	OEM	–	禁用 Type-C 连接器 SS 双通道支持
USB_SS_DISABLE	OEM	–	禁用 USB SS 枚举
ENUM_TIMEOUT	OEM	–	提供 USB 枚举超时
FORCE_USB_BOOT	GPIO	1	强制通过 USB 启动
FORCE_DLOAD_DISABLE	OEM	–	禁用强制下载
PBL_LOG_DISABLE	OEM	–	禁用启动 ROM 日志
ANTI_ROLLBACK_1_LSB/MSB	OEM	–	提供 XBL 或紧急加载模式防回滚

启动功能配置



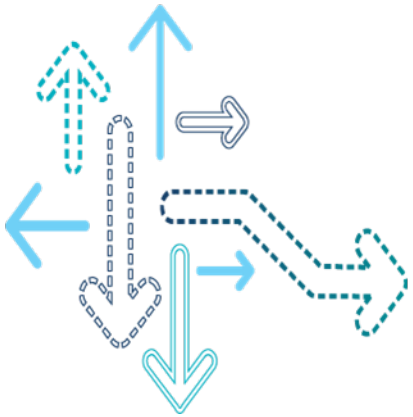
启动选项

启动功能	BOOT_CONFIG[4..1] (FAST_BOOT[3:0])	启动序列	使用熔丝选择启动选项				使用 GPIO 选择启动选项			
			OEM_CONFIG FUSE: FAST_BOOT_SELECT[3:0]				BOOT_CONFIG[4..1]			
			位 3	位 2	位 1	位 0	GPIO [39]	GPIO [133]	GPIO [100]	GPIO [99]
启动选项	0b000000	UFS → SD → USB 3.1 上的 USB	0	0	0	0	0	0	0	0
	0b000001	SD → UFS → USB 3.1 上的 USB	0	0	0	1	0	0	0	1
	0b000010	SD → USB 3.1 上的 USB	0	0	1	0	0	0	1	0
	0b000011	USB 3.1 上的 USB	0	0	1	1	0	0	1	1
	0b000100	eMMC → SD	0	1	0	0	0	1	0	0
	0b000101	SD → eMMC	0	1	0	1	0	1	0	1

注：

- SDM670 芯片组的默认可启动设备为 UFS → SD → USB。
- 对于 eMMC 可启动设备，会为 FASTBOOT_SEL_2 引脚拉高 GPIO 133。

安全启动



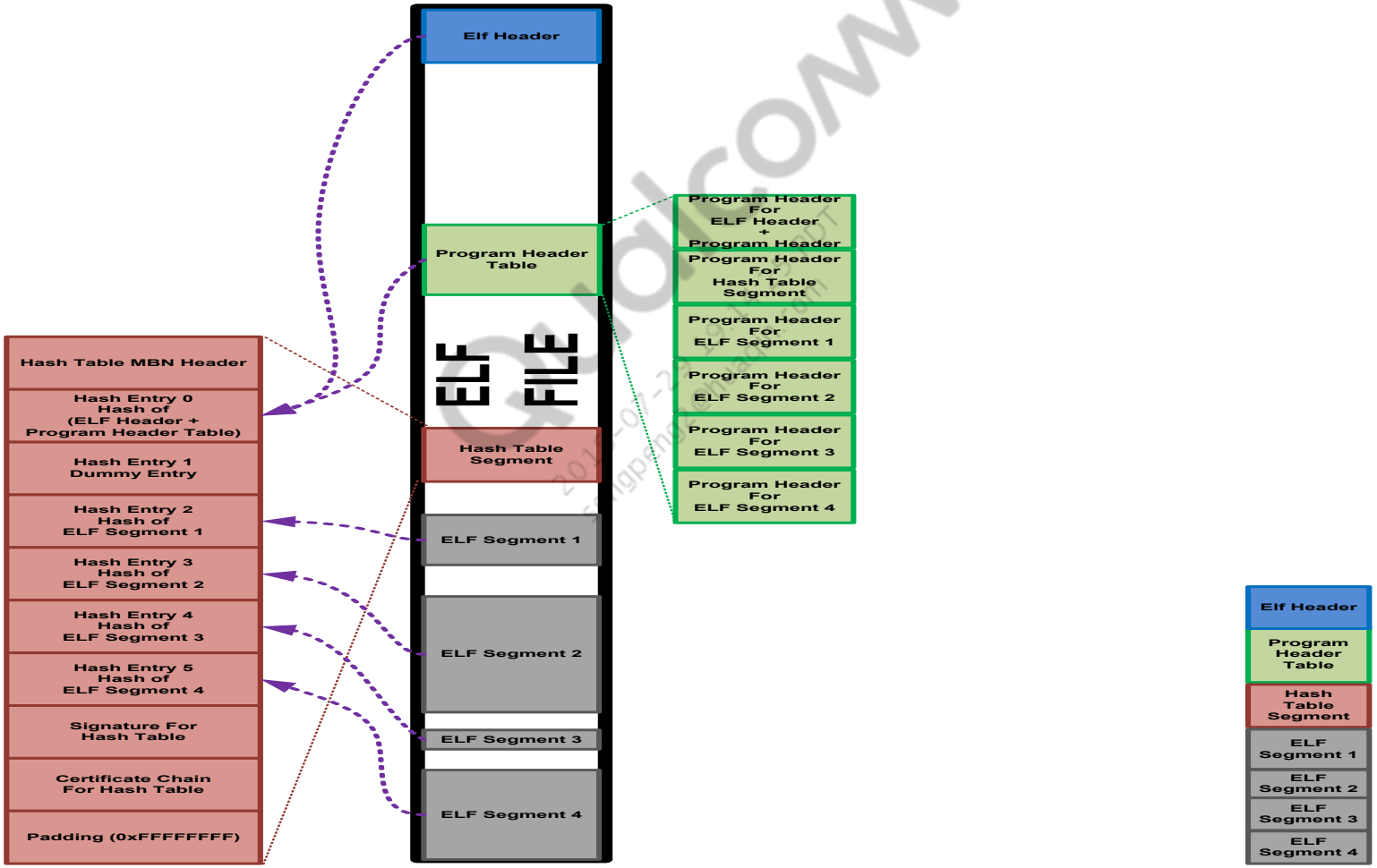
安全启动流程变更

- 之前，XBL-Loader 或 SBL1 在安全 EL3 下运行。
- 提供给 OEM 的 XBL 源代码可进行修改，也就是说，启动链中的一小部分软件在最高权限模式下运行。
- SDM670 芯片组具备下列功能：
  - XBL-SEC 映像作为中间层在 XBL 与 PBL 之间运行
  - XBL-SEC 由 QTI 信任根签名
  - XBL-SEC 仅以 QTI 签名的二进制形式提供给 OEM
  - XBL-SEC 在执行任何非安全软件或 OEM 可修改软件之前设置安全关键型资源的访问控制
  - XBL-SEC 跳转到 XBL-Loader 之后，XBL-Loader 在非安全模式下运行

安全启动流程变更（续）

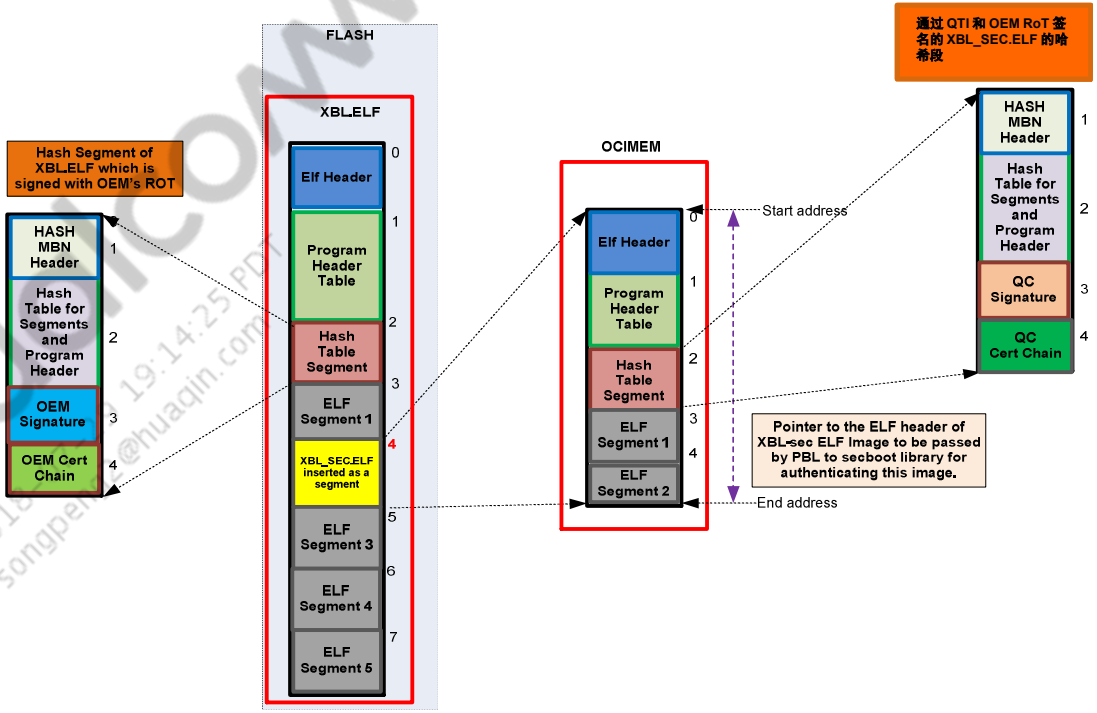
- 为了确保以最高安全级别执行鉴权例程，引入了以下更改：
  - PBL 鉴权 XBL.elf，其中包含 XBL-SEC
  - XBL-Loader 加载映像，比如 QTEE、AOP 和 ABL。鉴权由 XBL-SEC 通过 SMC 调用完成，XBL-SEC 使用 PBL 的安全 API 在 EL3 模式下进行鉴权
    - PBL 强制设备编程器执行相同操作，且必须具有 XBL-SEC 段
- 之前，QTEE 在安全模式下运行，并且仅由 OEM 签名。为了增强平台安全性，QTEE 映像签名包含以下变更：
  - QTEE 映像分别由 QTI 和 OEM RoT 签名
  - QTEE 映像的哈希段包含 OEM 和 QTI 的签名和证书链
  - 要执行 QTEE 映像，两个签名必须都通过鉴权，而鉴权通过 XBL-SEC 来完成

标准 ELF 格式

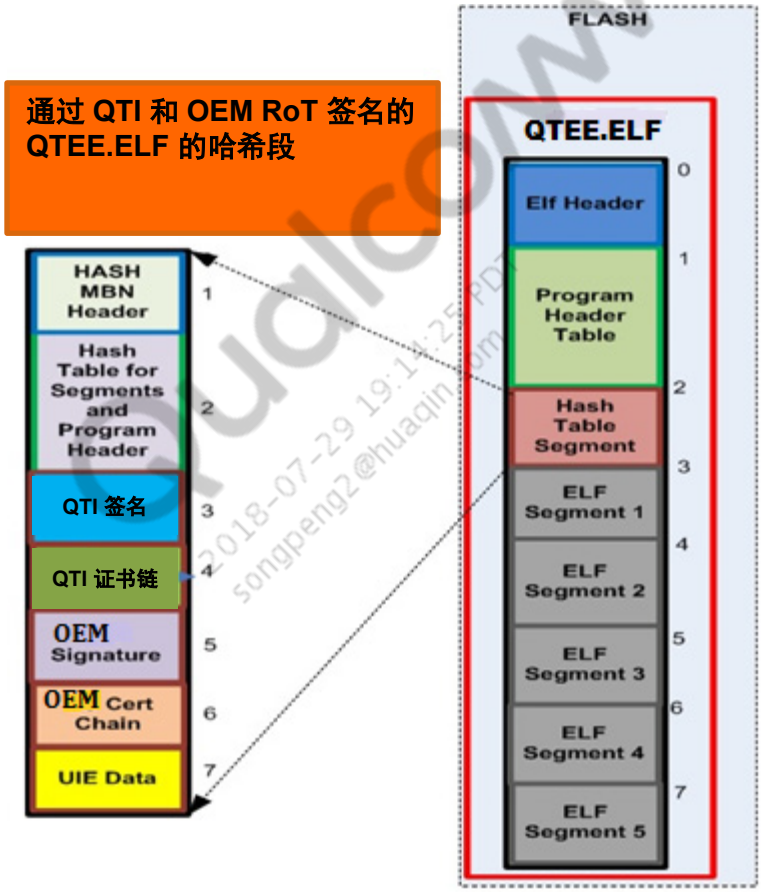


XBL.ELF/Deviceprogrammer.ELF 双重签名格式

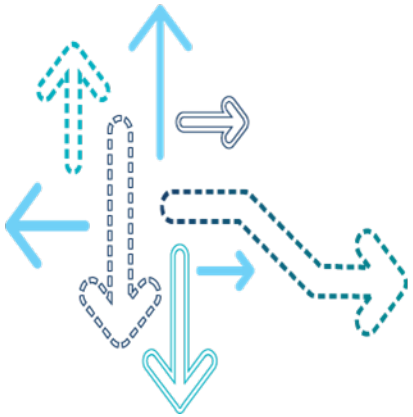
- XBL-SEC 按以下顺序进行两次签名和鉴权：
  1. XBL_SEC.ELF 的哈希段通过 QTI ROT 签名
  2. 签名后的 XBL_SEC.ELF 作为段的组成部分插入到 XBL.ELF
  3. XBL.ELF 的哈希段更新为使用已插入 XBL_SEC 段的哈希值
  4. XBL.ELF 的哈希段通过 OEM ROT 签名



QTEE.ELF 双重签名格式



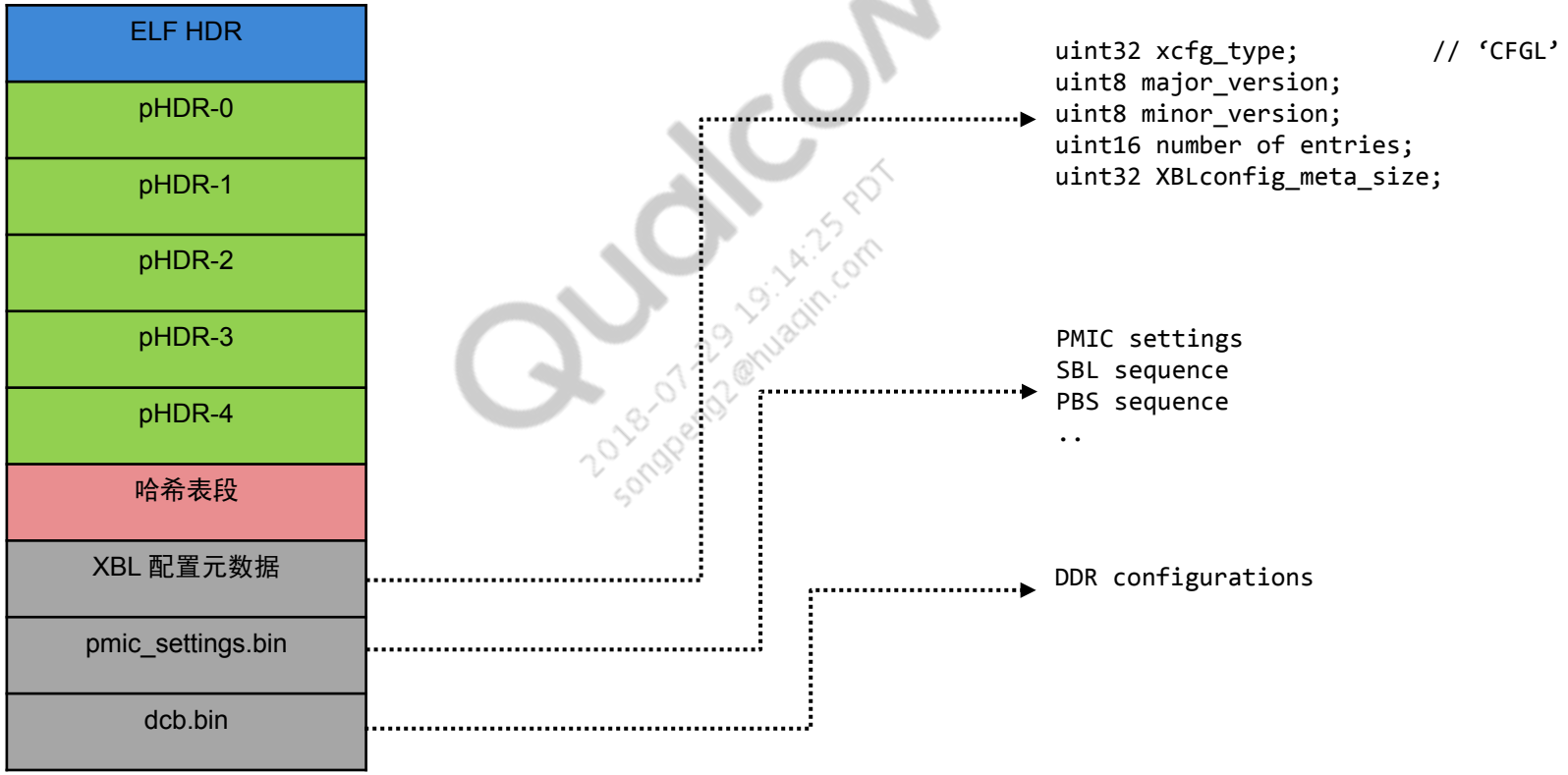
XBL 配置



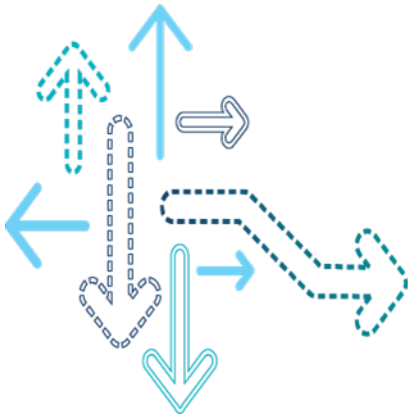
概述

- 对于 SDM670 芯片组，PMIC 设置和 DDR 配置会打包到 XBL 配置中
- XBL 配置将配置从 XBL 代码序列中分离
- 单个配置文件即可实现：
  - 内存优化 – PMIC、DDR 和其他驱动程序可将设置分为多个配置二进制项，然后只加载或使用需要的设置
  - 按照硬件模块版本、芯片组等灵活地对配置项进行更好的管理
  - 通过独立工具将已更改的配置二进制项转换为 ELF，无需调用编译操作
- OEM 可将配置添加到 XBL 配置

ELF 格式

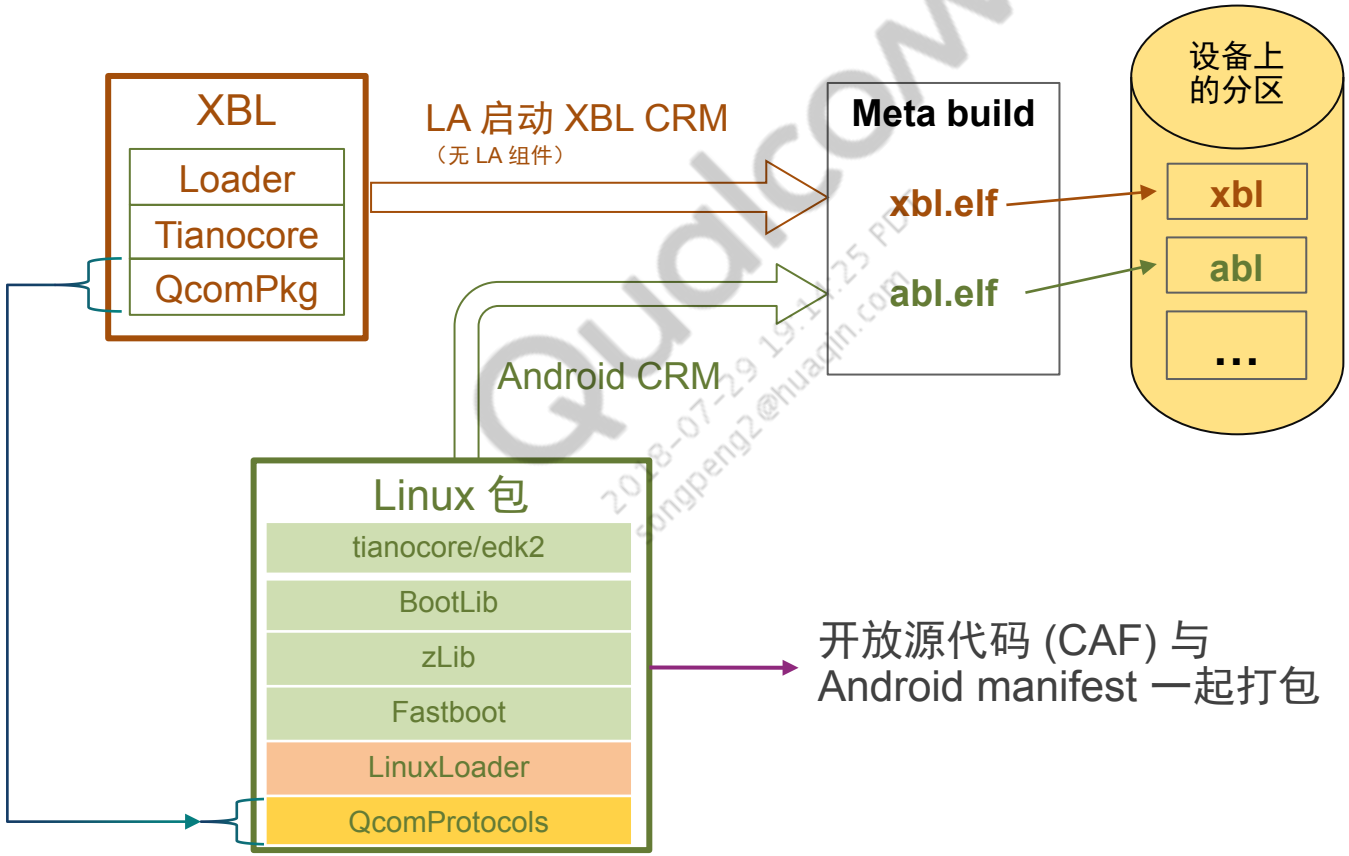


Linux UEFI 启动

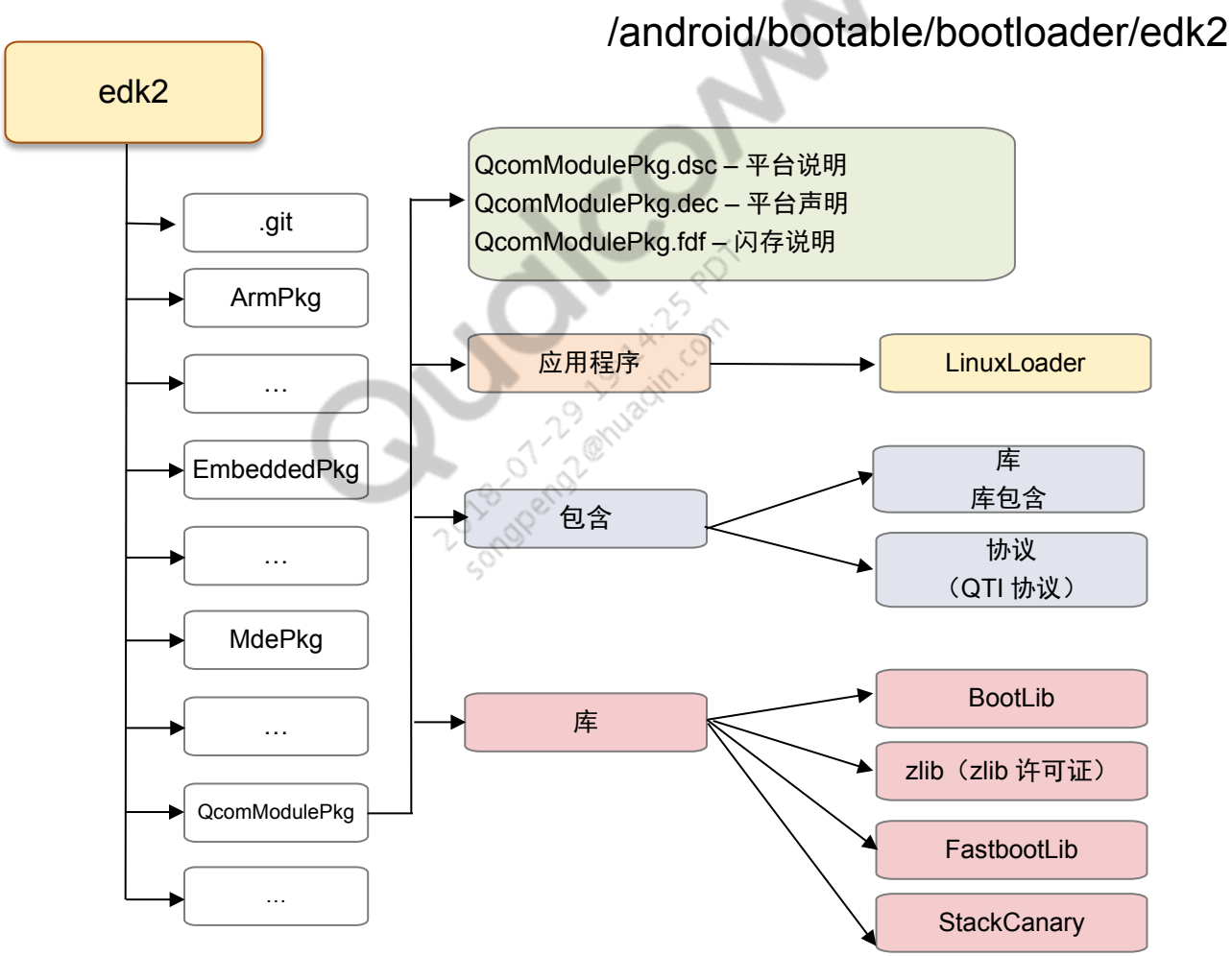


UEFI 启动架构

- UEFI Linux 启动交付在 xbl.elf 和 abl.elf 这两个映像中执行，如下图所示



Android 版本中的 UEFI 目录



UEFI 应用程序

- UEFI 应用程序是由 EFI 加载程序加载的 EFI 可加载映像
- UEFI 应用程序占用协议，且由用户驱动，也就是说，任务完成后，UEFI 应用程序会退出
- 默认情况下，SDM670 Android 版本和 Linux 加载器上有 UEFI 应用程序，OEM 可以创建自己的应用程序
- 多个 UEFI 应用程序可打包到固件区块中
- 更多信息，参见 *Linux Android UEFI Overview* (80-P2484-37)

UEFI 启动功能

- 以下是 Linux UEFI 启动的功能：
  - 从存储设备读取启动映像
  - 执行重启模式管理（正常模式、恢复模式、工厂模式和 OEM 模式）
  - 处理按键事件以进行启动模式选择
  - 音量增大（启动到恢复模式）
  - 音量减小（启动到快速启动模式）
  - 使用已验证的启动方法鉴权内核
  - 将 DDR 信息传递到内核
  - 将命令行参数传递到内核，以提供关于充电器模式、存储类型、UART 控制台等的信息

参考资料

标题	文档号
Qualcomm Technologies, Inc.	
DDR SDRAM CDT/ECDT User Guide	80-N1218-1
Software Configuration Data Table (CDT)	80-N3411-1
Linux Android UEFI Overview	80-P2484-37
SDM670 Digital Baseband Design Guidelines/Training Slides	80-PB873-5B
SDM670 Device Specification (Advance Information)	80-PB873-1

参考资料（续）

缩略词或术语	定义
APPS PBL	应用程序主启动加载程序 (Application Primary Boot Loader)
ABL FV	应用程序启动加载程序固件区块 (Application Boot Loader Firmware Volume)
AOP	实时响应处理器 (Always-On Processor)
APDP	APPS 调试策略 (APPS Debug Policy)
DCB	DDR 配置块 (DDR Configuration Block)
HLOS	高级操作系统 (High Level Operating System)
MBA	Modem 启动鉴权程序 (Modem Boot Authenticator)
OCIMEM	片上内存 (On-Chip Internal MEMory)
PIL	外设映像加载程序 (Peripheral Image Loader)
QHEE	Qualcomm Hypervisor 执行环境 (Qualcomm Hypervisor Execution Environment)
QTEE	Qualcomm 受信任执行环境 (Qualcomm Trusted Execution Environment)
SDI	系统调试映像 (System Debug Image)
SHRM	系统硬件资源管理器 (System Hardware Resource Manager)
SP PBL	安全处理器主启动加载程序 (Secure Processor Primary Boot Loader)
UEFI	统一可扩展固件接口 (Unified Extensible Firmware Interface)
XBL	可扩展启动加载程序 (eXtensible Boot Loader)

问题？

https://createpoint.qti.qualcomm.com