

该附录介绍硬核处理器系统 (HPS) 的引导和 Altera 片上系统 (SoC)FPGA 器件的 FPGA 部分的配置。

当处理器从复位（例如，上电时）释放并且执行复位异常处理地址的内部引导 ROM 中的代码时，HPS 引导开始进行。当引导 ROM 中的代码跳到引导软件的下一个阶段时，引导程序结束。引导软件的下一个阶段被称为预加载器。预加载器可以被定制并且通常存储在 HPS 外部的基于闪存的非易失存储器。

处理器可从以下资源引导：

- 可通过 NAND 闪存控制器访问的 NAND 闪存
- 可通过 SD/MMC 闪存控制器访问的 Secure Digital/MultiMediaCard(SD/MMC) 闪存
- 可通过 quad SPI 闪存控制器访问的串行外设接口 (SPI) 和 quad SPI 闪存
- FPGA 内核逻辑

根据引导器件，HPS 引导支持间接或直接执行预加载器。间接执行时，引导 ROM 代码预加载器从引导器件复制到片上 RAM 并且跳转到它。间接执行用于闪存引导资源。直接执行时，引导 ROM 代码直接从引导器件或从 FPGA 内核逻辑引导资源执行预加载器。

当 FPGA 部分从复位状态（例如，上电时）释放时，器件的 FPGA 部分的配置开始。器件的 FPGA 部分中的控制模块 (CB) 负责获得 FPGA 配置镜像并且对 FPGA 进行配置。当配置镜像完全被加载并且 FPGA 进入用户模式时，FPGA 配置结束。FPGA 配置镜像由用户提供并且通常存储在非易失的基于闪存的存储器中。FPGA CB 可通过 FPGA 管理器从 HPS 获得配置镜像或从 Cyclone® V FPGA 系列支持的任何资源获得配置镜像。



要了解关于引导和配置过程期间使用的存储器和外设模块的更多信息，请参考它们各自的章节：

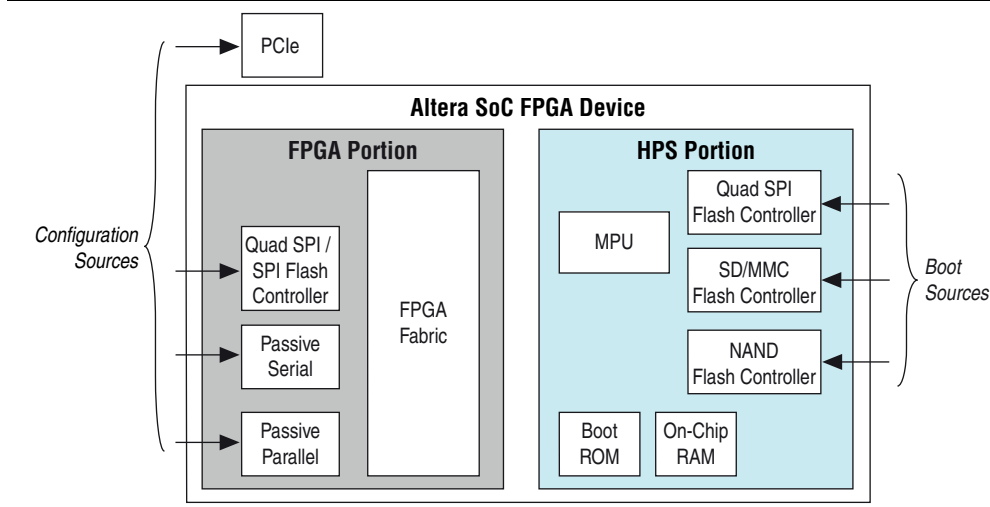
- [片上处理器](#)章节 (*Cyclone V 器件手册第 3 卷*)。
- [FPGA 管理器](#)章节 (*Cyclone V 器件手册第 3 卷*)。

以下 3 个图显示了 3 种 HPS 引导和 FPGA 配置方案。图中的箭头代表数据流方向。

■ 单独的

图 A-1 显示了 FPGA 配置和 HPS 引导单独地出现。FPGA 配置从非 HPS 资源获得配置镜像，而 HPS 引导从非 FPGA 内核逻辑资源获得预加载器。

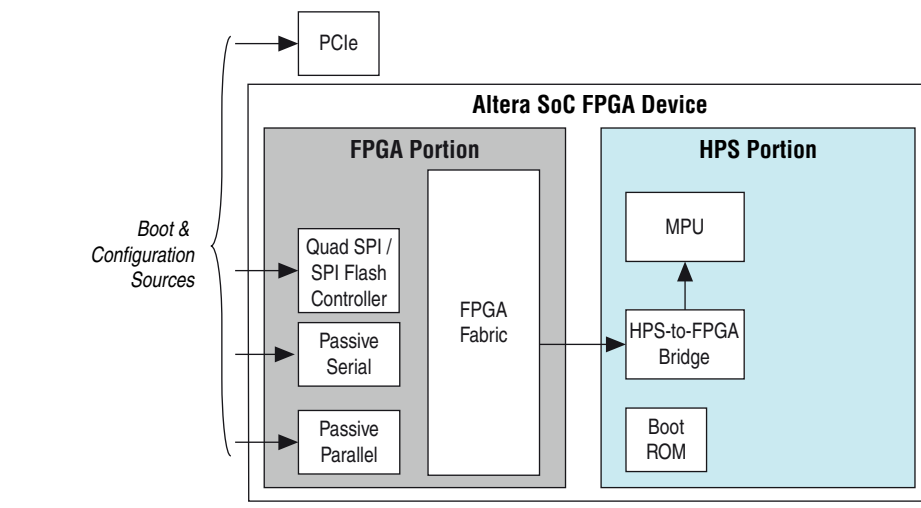
图 A-1. 单独地 FPGA 配置和 HPS 引导



■ 先配置 FPGA

图 A-2 显示了 FPGA 首先通过非 HPS 配置资源的其中一个进行配置，然后 HPS 从 FPGA 内核逻辑进行引导。HPS 引导等待 FPGA 内核逻辑上电并处于用户模式后才执行。HPS 引导 ROM 代码通过 HPS-to-FPGA 桥接从 FPGA 内核逻辑执行预加载器。根据设计和实现，预加载器可从 FPGA RAM 获得或可通过访问外部接口获得。

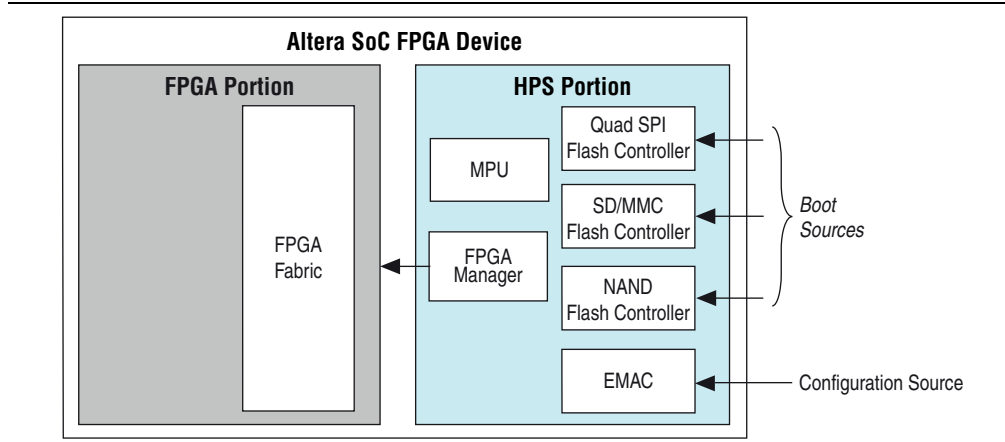
图 A-2. 先配置 FPGA



■ 先引导 HPS

图 A-3 显示了 HPS 首先通过其非 FPGA 内核逻辑引导资源的其中一个进行引导，然后运行在 HPS 上的软件通过 HPS FPGA 管理器配置 FPGA 内核逻辑。HPS 的软件从任何一个闪存器件或通信接口（例如，以太网介质访问控制，EMAC）获得 FPGA 配置镜像。软件由用户提供并且引导 ROM 不参与配置 FPGA 内核逻辑。

图 A-3. 先引导 HPS



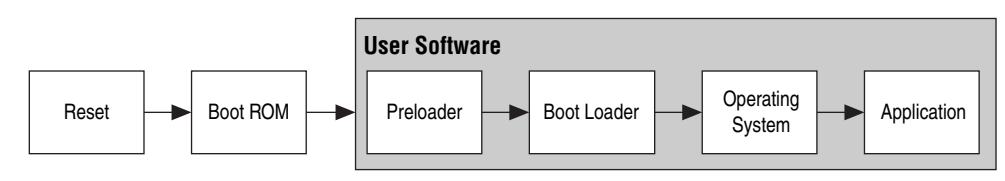
HPS 引导

引导 HPS 的软件是一个多阶段进程。每个阶段均负责加载下一个阶段。第一个软件阶段是引导 ROM。引导 ROM 代码查找并且执行称为预加载器的第 2 个阶段软件。预加载器如果找到下一个阶段软件，那么对其执行。预加载器和接下来的引导阶段（如果存在）统称为用户软件。

只有引导 ROM 代码位于 HPS。用户软件位于 HPS 的外部并且由用户提供。引导 ROM 代码仅知道预加载器，但不知道接下来任何可能存在的引导阶段。

图 A-4 显示了典型的引导流程。用户软件中的引导阶段可能比显示的多或少，并且引导阶段的作用可能会不同。

图 A-4. 典型引导流程



引导过程概述

复位

当 MPU 中的 CPU 从复位状态退出时，引导过程开始。当 CPU 退出复位时，它开始运行复位异常处理地址的代码。正常操作中，引导 ROM 被映射到复位异常处理地址，所以代码开始在引导 ROM 中运行。

映射复位异常处理地址的片上 RAM 或 SDRAM 并且运行除引导 ROM 代码以外的代码是可能的。然而，该章节假设引导 ROM 映射到复位异常处理地址。

引导 ROM

引导 ROM 包含 HPS 复位退出后执行的软件代码。引导 ROM 代码决定它运行在 CPU0 还是 CPU1。如果运行在 CPU1，那么引导 ROM 代码会跳转到系统管理器中引导 ROM 代码寄存器组 (romcodegrp) 的 CPU1 起始地址寄存器 (cpulstartaddr) 的值。如果运行在 CPU0，那么引导 ROM 代码读取引导选择 (BSEL) 和时钟选择 (CSEL) 值 (在系统管理器中引导信息寄存器 (bootinfo) 的 bsel 和 csel 域中) 以便决定引导源并且设置时钟管理器。bsel 和 csel 域值来自 BSEL 和 CSEL 管脚。当退出复位时，系统管理器采样这些管脚的值。

表 A - 1 列出了每个闪存器件选择的 bsel 域值。

表 A - 1. bsel 域值和闪存器件选择

bsel 域值	闪存器件
0x0	保留
0x1	FPGA (HPS-to-FPGA 桥接)
0x2	1.8 V NAND 闪存
0x3	3.0 V NAND 闪存
0x4	外部收发器的 1.8 V SD/MMC 闪存
0x5	内部收发器的 3.0 V SD/MMC 闪存
0x6	1.8 V SPI 或 quad SPI 闪存
0x7	3.0 V SPI 或 quad SPI 闪存

对于闪存引导资源的间接执行，引导 ROM 代码将预加载器镜像从闪存器件加载到片上 RAM 并且将软件控制传递到片上 RAM 的预加载器。对于 FPGA 内核逻辑引导资源的直接执行，引导 ROM 代码等待器件的 FPGA 部分处于用户模式，然后将软件控制传递到 FPGA RAM 中的预加载器。

预加载器

预加载器的功能是用户定义的。然而，主要功能包括初始化 SDRAM 接口和配置 HPS I/O 管脚。初始化 SDRAM 支持预加载器加载下一个阶段的引导软件 (可能不适合片上 RAM 中的 60 kilobytes (KB))。下一个典型的软件阶段是开源引导加载器，U-boot。

预加载器被允许从 HPS 可用的任何器件加载下一个阶段引导软件。典型源头包括含有预加载器的同一闪存器件，一个不同的闪存器件或一个如 EMAC 的通信接口。

引导加载器

引导加载器加载操作系统并且将软件控制传递到操作系统。

引导 ROM

引导 ROM 代码的功能是决定引导资源、HPS 复位后对其初始化并且跳转到预加载器。间接执行的情况中，引导 ROM 代码将预加载器镜像从闪存加载到片上 RAM。引导 ROM 执行以下操作来初始化 HPS:

使能指令缓存、分支预测器、浮点单元和 NEON 矢量单元

- 设置 level 4 (14) 看门狗 0 计时器
- 根据 CSEL 值配置主 PLL 和外设 PLL
- 根据 BSEL 值配置 I/O 单元和管脚复用

■ 以默认设置初始化闪存控制器

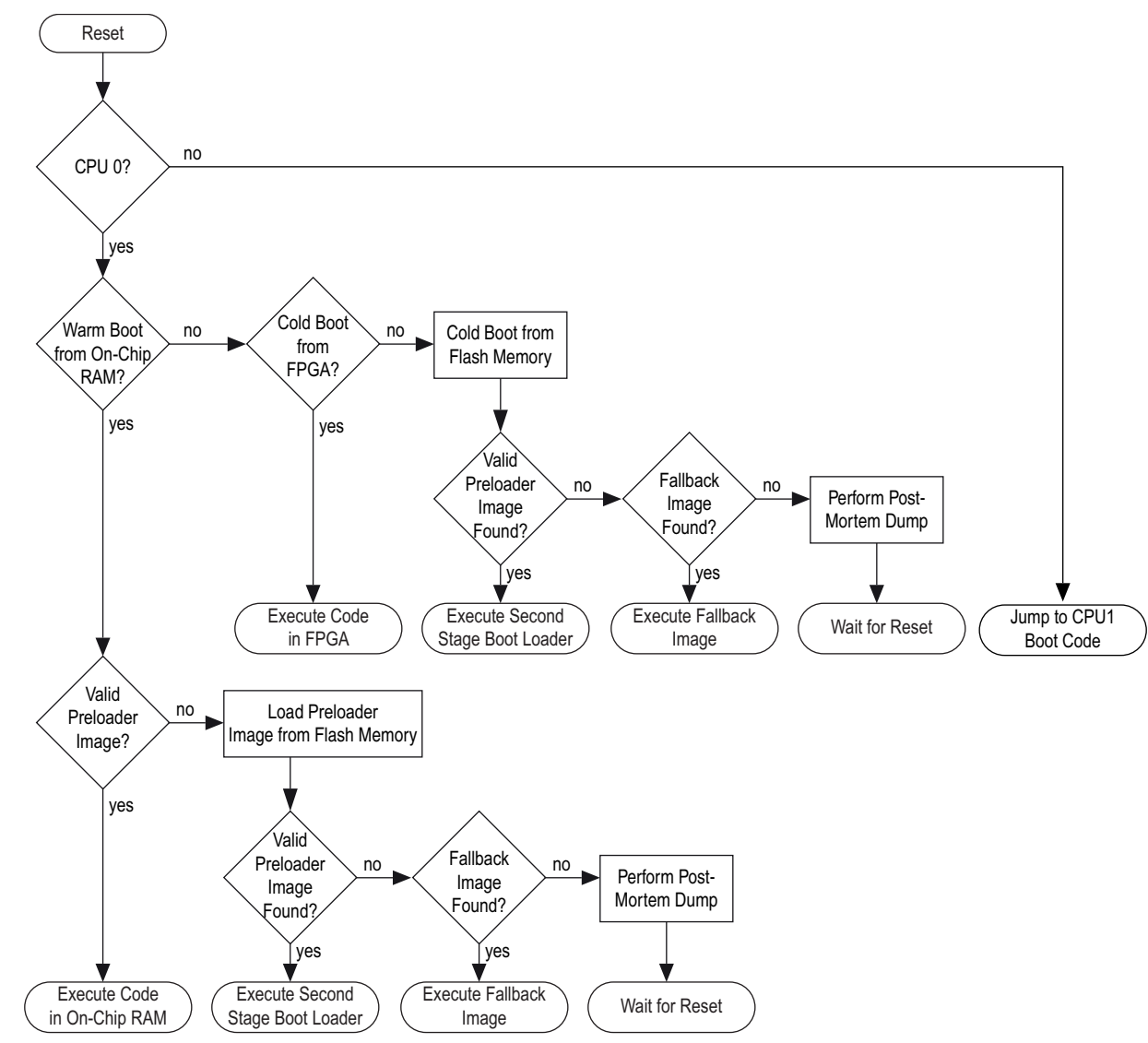
当执行时，引导 ROM 代码使用片上 RAM 的顶部 4 KB 作为数据工作区。该区域为复位后的引导 ROM 代码保留直到引导 ROM 代码将软件控制传递到预加载器。这将间接执行中的预加载器的最大容量限制为 60 KB。

引导 ROM 流程

该部分对软件复位流程一直到引导 ROM 代码将软件控制传递到预加载器进行介绍。

图 A - 5 显示了引导 ROM 流程。引导 ROM 代码可以执行片上 RAM 的热引导、器件的 FPGA 部分的冷引导或闪存的冷引导。

图 A - 5. 引导 ROM 流程



器件的 FPGA 部分的冷引导期间，引导 ROM 代码等待直到 FPGA 准备就绪，然后试图通过 HPS-to-FPGA 桥接在地址 0x0 直接执行。例如，引导软件可能会由器件的 FPGA 部分中的地址 0x0 的初始化的片上 RAM 提供。闪存的冷引导期间，引导 ROM 代码试图将第一个预加载器镜像从闪存加载到片上 RAM 并且将控制传递到预加载器。如果该镜像无效，那么引导 ROM 代码试图从闪存加载接下来的 3 个镜像。如果在接下来的加载中仍然没有有效镜像，那么引导 ROM 代码检查器件的 FPGA 部分是否存在一个备用镜像。

片上 RAM 的热引导期间，引导 ROM 代码读取系统管理器中 romcodegrp 组的预加载器状态寄存器 (initstate)，以便决定片上 RAM 中是否具有有效预加载器镜像。如果片上 RAM 中具有有效预加载器镜像，那么引导 ROM 代码跳过从闪存加载预加载器镜像，而是将控制传递到片上 RAM 中的预加载器。

如果片上 RAM 中没有有效预加载器镜像，那么引导 ROM 代码试图加载从闪存加载的最后一个有效预加载器镜像（由系统管理器的 romcodegrp 组的初始软件最后镜像加载寄存器 (initlastld) 的 index 域识别）。如果镜像无效，那么引导 ROM 代码试图从闪存加载接下来的 3 个镜像。如果片上 RAM 或闪存中不存在有效预加载器镜像，那么引导 ROM 代码检查器件的 FPGA 部分是否有备用镜像。

加载预加载器

对于间接执行，引导 ROM 代码将预加载器镜像从闪存加载到片上 RAM 并且将控制传递到预加载器。引导 ROM 代码通过验证预加载器镜像中的头和循环冗余校验 (CRC) 来检查有效镜像。图 A-6 显示了预加载器头。

引导 ROM 代码检查头的以下信息：

- 验证字 — 验证预加载器镜像。验证字具有一个固定值为 0x31305341。
- 版本 — 表明预加载器版本。
- 标记 — 未使用
- 编程长度 — 从偏移 0x0 到代码区域结束的整个镜像长度（以字节表示），包括异常向量和 CRC。
- 校验和 — 头中的所有字节的校验和，从偏移 0x40 到 0x49。

预加载器镜像的最大容量为 60 KB。该容量受限于 64 KB 大小的片上 RAM，其中 4 KB 被保留为引导 ROM 数据和堆栈的工作区。引导 ROM 代码传递控制到预加载器后，预加载器可以使用该 4 KB 区域（例如用于堆栈和数据）。接下来复位时，该 4 KB 区域由引导 ROM 代码而重写。

图 A - 6 显示了 RAM 从引导 ROM 加载后的片上 RAM 中的预加载器镜像布局。

图 A - 6. 预加载器镜像布局 (1)

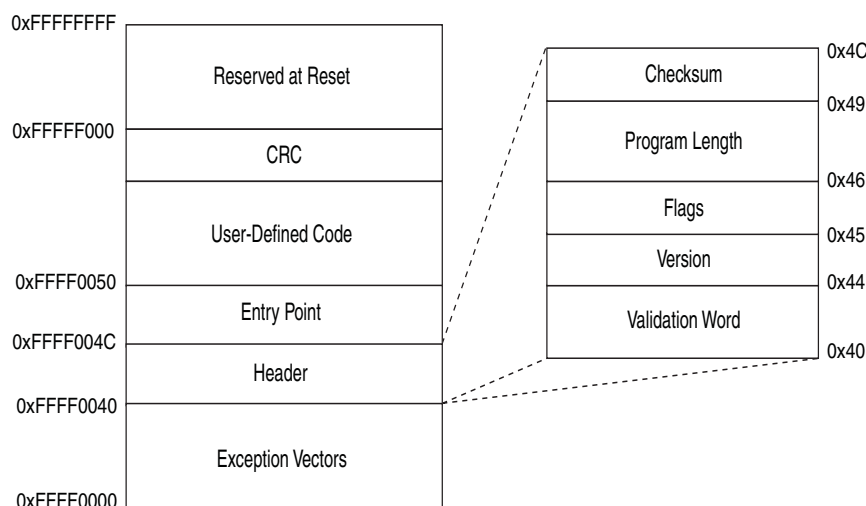


图 A - 6 注释:

(1) 地址是不按比例。

Exception vectors— 异常向量位于片上 RAM 的开端。通常，预加载器将存储器映射的最低区域重新映射到片上 RAM (来自引导 ROM) 以便更易于访问异常向量。

Header— 包含例如验证字、版本、标记、编程长度和校验的信息，以便引导 ROM 代码将控制传递到预加载器之前验证预加载器镜像。

Entry point— 包含预加载器镜像地址。引导 ROM 代码验证头之后，引导 ROM 代码跳转到该地址。

User-defined code— 通常包含预加载器的编程代码。

CRC— 包含从地址 0xFFFF0000 到 0xFFFF0000 + (*Program Length* * 4) - 0x0004 的数据的 CRC。用于验证预加载器镜像的多项式是 $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ 。没有比特反射。余数的初始值是 0xFFFFFFFF，最后值是 0xFFFFFFFF 的 XORed。

Reserved at reset— 复位后顶部 4 KB 为引导 ROM 代码而保留。引导 ROM 代码将该区域用作内部结构、工作区和事后分析转储 (post-mortem dump)。该区域包括共享内存，其中引导 ROM 代码将信息传递到预加载器。

共享内存

共享内存包含引导 ROM 代码传递到预加载器的信息。引导 ROM 代码将共享内存的位置传递到预加载器的寄存器 r0，如 A - 8 的“进入预加载器时的 HPS 状态”中所介绍。

共享内存包含以下信息:

- 共用信息 — 引导 ROM 代码使用的非闪存指定的设置。
- 保存的硬件寄存器信息 — 包含硬件寄存器值 (引导 ROM 代码在寄存器由引导 ROM 代码修改之前保存)。

- 闪存器件指定的信息 — 包含引导 ROM 代码使用的闪存器件指定的设置，预加载器会使用该设置以便在没有重新初始化的情况下继续使用闪存器件。
- 表 A-2 列出了共享内存模块的内容。

表 A-2. 共享内存模块

信息类型	内容	说明
共用	闪存镜像	表明引导 ROM 代码从闪存器件加载了哪个预加载器镜像 (0-3)。一个非零值表明存在一个错误加载镜像 0 并且引导 ROM 代码加载了另一个镜像。
	使用的 CSEL 值	表明引导 ROM 代码使用的 CSEL 值。通常，该值从系统管理器的 romcodegrp 组的 bootinfo 寄存器中的 csel 域读取。然而，如果 PLL 没有成功锁定，那么引导 ROM 代码忽略 csel 域并且使用零来表示 PLL 旁路模式。
	使用的 BSEL 值	表明引导 ROM 代码使用的 BSEL 值。通常，该值从系统管理器的 romcodegrp 组的 bootinfo 寄存器的 bsel 域读取。
	最后一页	表明从闪存器件读取的最后一页的地址。
	页面大小	表明闪存器件使用的页面大小。 <ul style="list-style-type: none">■ 对于 NAND 闪存，引导 ROM 代码从 NAND 闪存控制器读取该值。■ 对于 SPI 和 quad SPI 闪存，引导 ROM 代码通过 quad SPI 闪存控制器配置页面大小。■ 对于 SD/MMC 闪存，引导 ROM 代码通过 SD/MMC 闪存控制器配置页面大小。
	闪存器件类型	表明引导 ROM 代码使用的闪存器件。
	步骤完成	跟踪引导过程中多达 64 个单独主要步骤的完成状态。64 位值的每个位被设置到引导过程中完成的每个主要步骤。
	CPU0 和 CPU1 损坏数据	表明 CPU0 和 CPU1 损坏数据。这些值为指针，指向引导 ROM 代码在损坏事件中保存损坏数据的位置。
保存的硬件寄存器	复位管理器的状态寄存器 (stat)	包含复位源和事件超时信息。
	系统管理器的 romcodegrp 组中的控制 (ctrl) 寄存器	包含用于控制引导 ROM 代码的信息。
	系统管理器的 romcodegrp 组的 initswstate 寄存器	包含预加载器达到有效状态时的魔值 0x49535756。
闪存器件指定的 (SD/MMC)	is_sd_card	表明卡类型。1 表示 SD 卡； 0 表示 MMC。
	is_sector_mode	表明卡的寻址模式。1 表示扇区寻址模式； 0 表示字节寻址模式。
	rca	包含相对卡地址，用于卡识别期间的主机卡通信。
	partition_start_sector	表明单元扇区中的分区起始偏移。0 表示原始模式。
	partition_size	单元扇区中的分区的大小。0 表示原始模式。

L4 看门狗 0 计时器

L4 看门狗 0 计时器被保留用于引导 ROM。如果看门狗复位在软件控制传递到预加载器之前发生，那么引导 ROM 代码试图加载最后有效预加载器镜像 (由系统管理器中 romcodegrp 组中的 initswlastld 寄存器识别)。

进入预加载器时的 HPS 状态

当引导 ROM 代码准备将控制传递到预加载器时，处理器 (CPU0) 处于以下状态：

- 指令缓存被使能
- 分支预测器被使能
- 数据缓存被禁用
- MMU 被禁用
- 浮点单元被使能
- NEON 矢量单元被使能
- 处理器处于 ARM 安全监督模式

引导 ROM 代码将 ARM® Cortex™-A9 MPCore™ 寄存器设置为以下值:

r0— 包含共享内存模块的指针, 可用于将信息从引导 ROM 代码传递到预加载器。共

- 享内存模块位于片上 RAM 的顶部 4 KB。
- r1— 包含共享内存的长度。
- r3— 未使用的并且设置为 0x0。
- r4— 未使用的并且设置为 0x0。

所有其它 MPCore 寄存器均未被定义。



当使用 FPGA 引导对 CPU0 进行引导, 或使用任何引导源对 CPU1 进行引导时, 所有 MPCore 寄存器、缓存、MMU、浮点单元和 NEON 矢量单元均未被定义。HPS 子系统和 PLL 均未被定义。

当引导 ROM 代码将控制传递到预加载器时, 以下情况也会出现:

- 引导 ROM 仍被映射到地址 0x0。
- L4 看门狗 0 计时器有效。

预加载器

预加载器通常执行以下操作:

- 初始化 SDRAM 接口
- 配置 L3 (NIC-301) GPV 寄存器 (l3regs) 的 remap 寄存器, 从而将片上 RAM 映射到地址 0x0, 以便预加载器处理异常情况。



片上 RAM 在地址 0xFFFF0000 也是可访问的。地址 0x0 是一个别名。

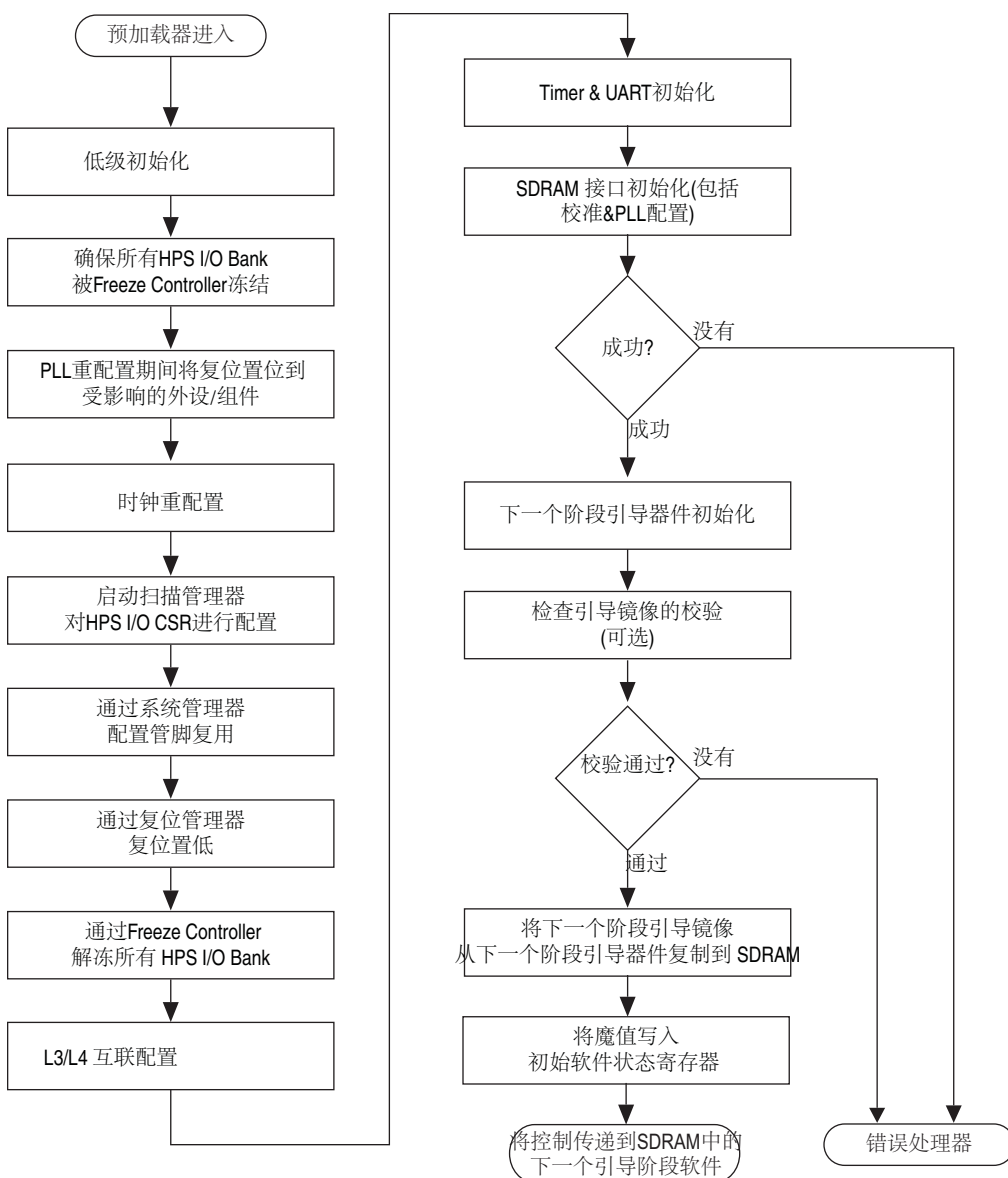
- 通过扫描管理器配置 HPS I/O。
- 通过系统管理器配置管脚复用。
- 通过时钟管理器配置 HPS 时钟。
- 初始化包含下一个阶段引导软件的闪存控制器 (NAND、SD/MMC 或 quad SPI)。
- 将下一个阶段引导软件加载到 SDRAM 并且将控制传递到下一个阶段引导软件。

典型预加载器引导流程

该部分介绍从预加载器进入到软件将控制传递到下一个阶段引导软件的典型软件流程。

图 A - 7 显示了典型预加载器引导流程。

图 A - 7. 典型预加载器引导流程



低级初始化阶段包括重配置或禁用 L4 看门狗 0 计时器、使指令缓存和分支预测器无效、将片上 RAM 重新映射到最底部存储区域并且设置数据区域。

一进入预加载器，L4 看门狗 0 计时器就会有效。预加载器可以禁用、重配置或保持看门狗计时器不变。一旦在复位后被使能，看门狗计时器不可以被禁用，只可以被暂停。之前被引导 ROM 代码使能的指令缓存和分支预测器需要被更改为无效。

预加载器需要重新映射异常向量表，因为当预加载器开始执行时，异常向量仍然指向引导 ROM 中的异常处理函数。通过将 L3 互联重新映射位 0 设置为高电平，片上 RAM 映射到存储器映射的最底部区域。重新映射后，异常向量将会使用预加载器镜像中的异常处理函数。

图 A - 8 显示了重新映射之前和之后的存储器映射。

图 A - 8. 重新映射片上 RAM (1)

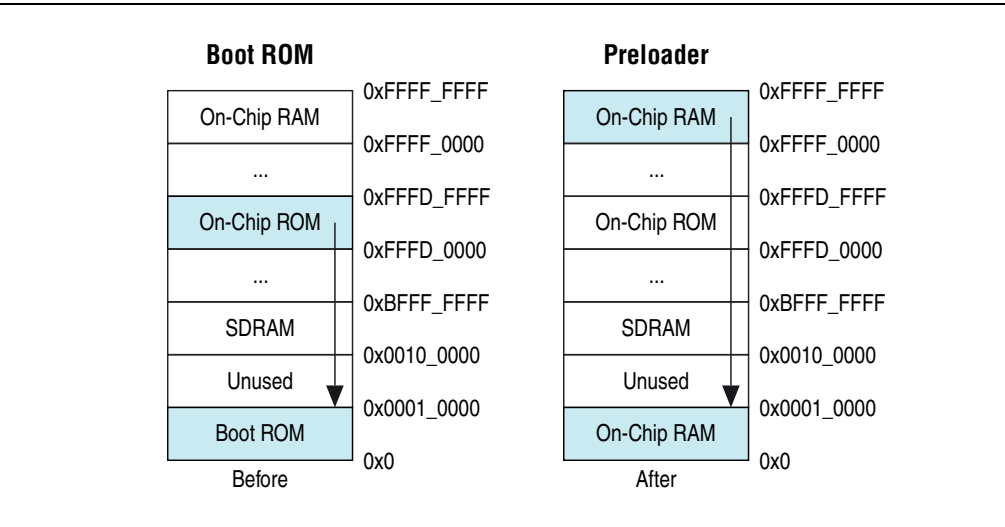


图 A - 8 注释:

(1) 地址是不按比例。

预加载器可以重配置所有 HPS 时钟。时钟重配置期间，预加载器将复位到受时钟更改影响的 HPS 中的外设。

预加载器通过扫描管理器配置 HPS I/O 管脚并且通过系统管理器配置管脚复用。预加载器启动扫描管理器中的冻结控制器来冻结所有 I/O 管脚并且在 I/O 配置和管脚复用期间使它们处于安全状态。

在冷启动过程中 SDRAM 会进行完整初始化而在热启动过程中进行部分初始化。对于完全初始化，预加载器在将 SDRAM 接口从复位释放之前配置 SDRAM PLL。SDRAM 校准调整 I/O 延迟和 FIFO 设置以补偿任何板上、器件的 FPGA 部分或存储器件的电路抖动或者不匹配。对于部分初始化，不需要进行 SDRAM PLL 配置和 SDRAM 校准。

通过检查引导镜像验证数据和镜像中的校验和，预加载器查找下一个阶段引导器件中的有效引导镜像。一旦被验证，预加载器就会将下一个阶段引导镜像从下一个阶段引导器件复制到 SDRAM。


软件将控制传递到下一个阶段引导软件之前，预加载器会写入一个有效值（例如 0x49535756）到系统管理器中 romcodegrp 组的预加载器 initstate 寄存器。该值表示在片上 RAM 中有一个有效引导镜像。当热复位发生时，引导 ROM 代码检查 initstate 寄存器的魔值以决定它是否需要将预加载器镜像重新加载到片上 RAM。

闪存器件

HPS 引导可用的闪存器件是 NAND、SD/MMC、SPI 和 quad SPI。闪存器件会存储以下类型的文件来进行引导：

- 预加载器二进制文件（多达 4 个副本）
- 引导加载器二进制文件
- 操作系统二进制文件
- 应用文件

■ FPGA 编程文件

 预加载器文件必须存储在没有文件系统的分区。

NAND 闪存器件

图 A - 9 显示了 NAND 闪存镜像布局。预加载器镜像位于模块大小倍数的偏移。如果镜像大小少于 64 KB，那么仅使用一个模块大小。因为一个模块是用于擦除操作的最小区域，那么所有特定镜像的更新都不会影响其它镜像。

图 A - 9. NAND 闪存镜像布局

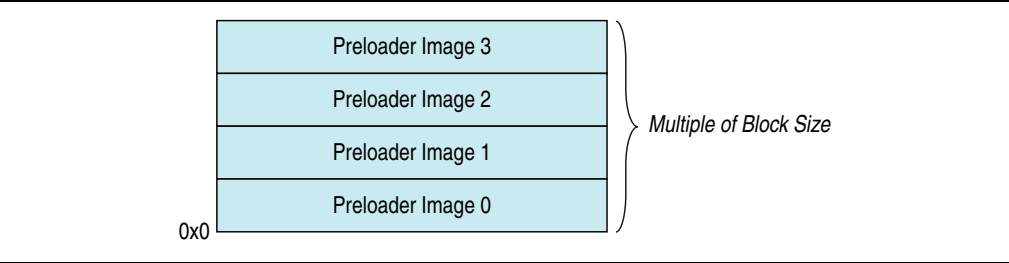


表 A - 3 列出了引导 ROM 代码中所支持的 NAND 闪存驱动器功能。

表 A - 3. NAND 闪存支持功能

功能	驱动器支持
器件	开放 NAND 闪存接口 (ONFI) 1.0 原始 NAND 或电子签名器件、使用集成错误纠正代码 (ECC) 的单层单元 (SLC) 和多层单元 (MLC) 器件。
片选	仅使用 CS0。只有 CS0 可用于 HPS，其它 3 个片选被布线到器件的 FPGA 部分。
总线宽度	仅为 x8
页面大小	512 字节、2 KB、4 KB 或 8 KB。
每模块页面	8、16、32 或 128
ECC	具有 8-bit 纠正的 512- 字节

表 A - 4 列出了 quad SPI 控制器的 CSEL 管脚设置。

表 A - 4. NAND 控制器 CSEL 管脚设置

设置	CSEL 管脚			
	0	1	2	3
oscl_clk (EOSC1 管脚) 范围	10 - 50 MHz	10 - 12.5 MHz	12.5 - 25 MHz	25 - 50 MHz
nand_x_clk /25 器件频率	oscl_clk/25, 2 MHz max	oscl_clk*20/25, 9.6 MHz max	oscl_clk*10/25, 9.6 MHz max	oscl_clk*5/25, 9.6 MHz max
nand_x_clk 控制器时钟	oscl_clk, 50 MHz max	oscl_clk*20, 240 MHz max	oscl_clk*10, 240 MHz max	oscl_clk*5, 240 MHz max
mpu_clk	oscl_clk, 50 MHz max	oscl_clk*32, 400 MHz max	oscl_clk*16, 400 MHz max	oscl_clk*8, 400 MHz max
PLL 模式	旁路	锁定	锁定	锁定

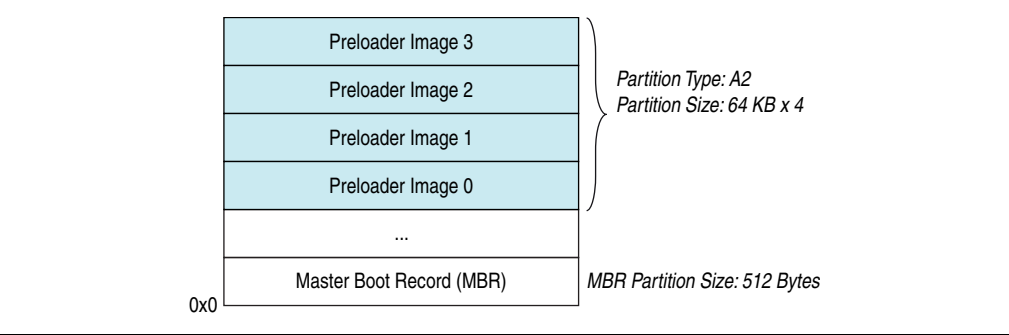
要了解关于 NAND 闪存的更多信息，请参考 *Cyclone V 器件手册* 第 3 卷的 [NAND 闪存控制器](#) 章节。

SD/MMC 闪存器件

图 A - 10 显示了 SD/MMC 闪存镜像布局。主引导记录 (MBR) 位于存储器的第一个扇区的 512 个字节。MBR 包含分区的信息 (地址和分区大小)。预加载器镜像通常存储在分区 A2 中。分区 A2 是一个没有文件系统的定制原始分区。

每个镜像的起始地址均根据以下公式：
起始地址 = 起始地址 + (<n> * 64 K)，其中 <n> 是镜像数

图 A - 10. SD/MMC 闪存镜像布局



SD/MMC 控制器支持两个引导模式：

- MBR（分区）模式
 - 引导镜像从定制分区 (0xA2) 读取
 - 第一个镜像位于分区的起始位置的偏移 0x0
 - 起始地址 = 分区起始地址
- 原始模式
 - 如果不存在 MBR 签名，那么 SD/MMC 驱动器假设它处于原始模式。引导镜像数据从用户区域的扇区直接读取并且位于 SD/MMC 的第一个扇区。
 - 第一个镜像位于存储卡的起始位置的偏移 0。
 - 起始地址 = 0

MBR

MBR 包含分区表，它总是位于存储器大小为 512 字节的第一个扇区 (LBA0)。MBR 包含可执行代码、4 个分区入口和 MBR 签名。一个 MBR 会由指定的工具 (如 FDISK 编程) 而创建。

表 A - 5 列出了 MBR 结构。

表 A - 5. MBR 结构 (1/2)

偏移	大小 (字节)	说明
0x000	446	代码区
0x1BE	16	分区 1 的分区入口
0x1CE	16	分区 2 的分区入口

表 A-5. MBR 结构 (2/2)

偏移	大小 (字节)	说明
0x1DE	16	分区 3 的分区入口
0x1EE	16	分区 4 的分区入口
0x1FE	2	MBR 签名: 0xAA55

标准 MBR 结构包含一个 4 个 16- 字节入口的分区。因而, 使用该标准表的存储卡不可以超过 4 个主分区或具有 3 个主分区和一个扩展分区。

每个分区类型均由分区入口定义。引导镜像存储在定制分区类型为 (0xA2) 的主分区。SD/MMC 闪存驱动器不支持文件系统, 所以引导镜像位于分区 A2 的固定位置。表 A-6 列出了分区入口。

表 A-6. 分区入口

偏移	大小 (字节)	说明
0x0	1	引导指示器。0x80 表示可引导的
0x1	3	起始 CHS 值
0x4	1	分区类型
0x5	3	结束 CHS 值
0x8	4	分区中的第一个部分的 LBA
0xB	4	分区中的扇区数

为了使用所支持的 SD/MMC 闪存, 引导 ROM 代码将 SD/MMC 控制器配置为默认设置。表 A-7 列出了 SD/MMC 控制器的默认设置。

表 A-7. SD/MMC 控制器默认设置

参数		默认	寄存器值
卡类型		1 位	SD/MMC 控制器寄存器 (sdmmc) = 0x0 中的卡类型寄存器 (ctype)
超时		最高	超时寄存器 (tmout) = 0xFFFFFFFF
FIFO 阈值 RX 水标级数		1	FIFO 阈值水标寄存器 (fifoth) 的 RX 水标级数 (rx_wmark)=0x1
时钟源		0	时钟源寄存器 (clksrc) = 0x0
模块大小		512	模块大小寄存器 (blksiz) = 0x200
时钟分频器	识别模式	32	时钟分频器寄存器 (clkdiv)= 0x10 (2*16=32)
	数据传输模式	旁路	时钟分频器寄存器 (clkdiv)= 0x00

表 A-8 列出了 SD/MMC 控制器的 CSEL 管脚设置。

表 A-8. SD/MMC 控制器 CSEL 管脚设置 (1/2)

设置	CSEL 管脚			
	0	1	2	3
oscl_clk (EOSC1 管脚) 范围	10 - 50 MHz	10 - 12.5 MHz	12.5 - 25 MHz	25 - 50 MHz

表 A - 8. SD/MMC 控制器 CSEL 管脚设置 (2/2)

设置		CSEL 管脚			
		0	1	2	3
ID 模式	sdmmc_cclk_out 器件时钟	oscl_clk/128, 391 KHz max	oscl_clk/32, 391 KHz max	oscl_clk/64, 391 KHz max	oscl_clk/128, 391 KHz max
	控制器波特率除数	32	32	32	32
数据传输模式	sdmmc_cclk_out 器件时钟	oscl_clk/4, 12.5 MHz max	oscl_clk*1, 12.5 MHz max	oscl_clk/2, 12.5 MHz max	oscl_clk/4, 12.5 MHz max
	控制器波特率除数 (仅为偶数)	1 (旁路)	1 (旁路)	1 (旁路)	1 (旁路)
sdmmc_clk 控制器时钟:		oscl_clk, 50 MHz max	oscl_clk, 50 MHz max	oscl_clk, 50 MHz max	oscl_clk*2, 50 MHz max
mpu_clk		oscl_clk, 50 MHz max	oscl_clk*32, 400 MHz max	oscl_clk*16, 400 MHz max	oscl_clk*8, 400 MHz max
PLL 模式		旁路	锁定	锁定	锁定

要了解关于 SD/MMC 的更多信息，请参考 *Cyclone V 器件手册 Handbook* 第 3 卷的 *SD/MMC 控制器* 章节。

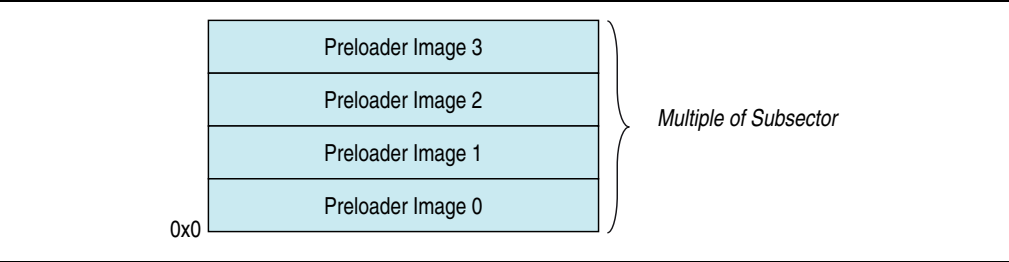
SPI 和 Quad SPI 闪存器件

图 A - 11 显示了 SPI 和 quad SPI 闪存镜像布局。预加载器镜像总是位于大小为子扇区倍数的偏移。如果镜像少于 64 KB，那么只有一个子扇区被使用。如果子扇区擦除不被支持，那么扇区擦除被支持。因为子扇区是用于擦出操作的最小区域，所以所有特定镜像的更新均不影响其它镜像。

第一个镜像位于偏移地址 0，接下来的镜像紧跟其后。每个镜像的起始地址基于以下公式：

起始地址 = (N * 64K)，其中 N 是镜像数。

图 A - 11. SPI 和 Quad SPI 闪存镜像布局



为了使用所支持的 SPI 或 quad SPI 闪存，引导 ROM 代码将 quad SPI 控制器配置为默认设置。表 A - 9 列出了 quad SPI 控制器的默认设置。

表 A - 9. Quad SPI 控制器默认设置 (1/2)

参数	默认设置	寄存器值
SPI 比特率	除以 4	quad SPI 控制器寄存器 (qspiregs) 中 quad SPI 配置寄存器 (cfg) 的主器件模式波特率除数域 (bauddiv) = 1。
读操作码	正常读取	器件读指令寄存器 (devrd) 中非 XIP 模式域 (rdopcode) 中读操作码 = 0x3。

表 A - 9. Quad SPI 控制器默认设置 (2/2)

参数	默认设置	寄存器值
指令类型	单一 I/O (1 位宽)	devrd 寄存器的地址传输宽度域 (addrwidth) 和数据传输宽度域 (datawidth) = 0。
当时钟相位为零时, 主器件模式片选输出在字之间被置低的主器件参考时钟延迟长度。	200 ns	quad SPI 器件延迟寄存器 (delay) 中片选置无效域 (nss) 的时钟延迟 请参考第 A - 17 页的表 A - 11 中的 delay[31:24]。
一个片选无效和另一个片选激活之间的主器件参考时钟延迟。该延迟确保两个不同的从器件的选择之间的平静周期并且要求发送 FIFO 为空。	0 ns	delay 寄存器中片选无效域 (btwn) 的时钟延迟 = 0x0。
当前传输的最后位和下一个传输的第一个位之间的主器件参考时钟中的延迟。如果时钟相位为零, 那么下一个传输的第一个位特指片选被取消选择的周期。	20 ns	delay 寄存器中最后传输位域 (after) 的时钟延迟。 请参考第 A - 17 页的表 A - 11 中的 delay[15:8]。
设置 qspi_n_ss_out 为低电平和第一个位传输之间的主器件参考时钟中添加的延迟。	20 ns	delay 寄存器中 qspi_n_ss_out 域 (init) 的时钟延迟。 请参考第 A - 17 页的表 A - 11 的 delay[7:0]。
地址字节数。	3 字节	器件大小寄存器 (devsz) 的地址字节域 (numaddrbytes) 数 = 2。

表 A - 10 列出了 quad SPI 控制器的 CSEL 管脚设置。

表 A - 10. Quad SPI 控制器 CSEL 管脚设置

设置	CSEL 管脚			
	0	1	2	3
oscl_clk (EOSC1 管脚) 范围	10 - 50 MHz	20 - 50 MHz	25 - 50 MHz	10 - 25 MHz
sclk_out 器件时钟	oscl_clk/4, 12.5 MHz max	oscl_clk/2, 25 MHz max	oscl_clk*1, 50 MHz max	oscl_clk*2, 50 MHz max
qspi_clk 控制器时钟	oscl_clk, 50 MHz max	oscl_clk*2, 100 MHz max	oscl_clk*4, 200 MHz max	oscl_clk*8, 200 MHz max
控制器波特率除数 (仅为偶数)	4	4	4	4
闪存读指令 (1 个空字节的 READ_FAST)	READ	READ	READ_FAST	READ_FAST
mpu_clk	oscl_clk, 50 MHz max	oscl_clk*8, 400 MHz max	oscl_clk*8, 400 MHz max	oscl_clk*16, 400 MHz max
PLL 模式	旁路	锁定	锁定	锁定

quad SPI 控制器寄存器 (qspi_regs) 中的 delay 寄存器对于生成主器件输出信号配置相对的延迟。

SPI 或 quad SPI 闪存需要满足以下时序要求:

- T_{SLCH} (delay[7:0]): 20 ns
- T_{CHSH} (delay[15:8]): 20 ns
- T_{SHSL} (delay[31:24]): 200 ns

表 A - 11 列出了 CSEL 管脚设置的 delay 寄存器配置。

表 A - 11. SPI 和 Quad SPI 闪存延迟配置

CSEL 管脚	$T_{\text{ref_clk}}$ (ns)	$T_{\text{sclk_out}}$ (ns)	器件延迟寄存器		
			delay[7:0]	delay[15:8]	delay[31:24]
0	20	80	0	0	6
1	10	40	0	0	16
2 - 3	5	20	0	0	36

计算延迟的公式是 $(T_{\text{delay}} - T_{\text{sclk_out}}) / T_{\text{ref_clk}}$ 。

要了解关于 quad SPI 闪存的更多信息，请参考 *Cyclone V 器件手册* 第 3 卷的 *Quad SPI Flash 控制器* 章节。

FPGA 配置

可通过利用非 HPS 资源或 HPS 配置 SoC 器件的 FPGA 部分。通过写入配置镜像到 HPS 中的 FPGA 管理器，执行在 HPS 上的软件配置 FPGA。通过访问 FPGA 管理器中的控制和状态寄存器 (CSR) 接口，软件可以控制配置过程并监控 FPGA 状态。

要了解关于通过 HPS FPGA 管理器配置 FPGA 的更多信息，请参考 *Cyclone V 器件手册* 第 3 卷的 *FPGA 管理器* 章节。

要了解关于配置 FPGA 的更多信息，请参考 *Cyclone V 器件手册* 第 1 卷的 *Configuration, Design Security, and Remote System Upgrades in Cyclone V Devices* 章节。

完全配置

HPS 使用 FPGA 管理器对器件的 FPGA 部分进行配置。以下序列建议了软件执行完全配置的一种方式：

1. 设置 FPGA 管理器寄存器 (fpgamgrregs) 的 ctrl 寄存器的 cdratio 和 cfgwidth 位以便匹配配置镜像的特征。这些设置取决于 MSEL 管脚输入。
2. 设置 ctrl 寄存器的 nce 位为 0 来使能 HPS 配置。
3. 设置 ctrl 寄存器的 en 位为 1 以使 FPGA 管理器可对输入信号配置进行控制。
4. 设置 ctrl 寄存器的 nconfigpull 位为 1 来下拉 nCONFIG 管脚并且使器件的 FPGA 部分处于复位阶段。
5. 轮询 stat 寄存器的 mode 位并且等待直到 FPGA 进入复位阶段。
6. 设置 ctrl 寄存器的 nconfigpull 位为 0 以将 FPGA 从复位释放。
7. 读取 stat 寄存器的 mode 位并且等待直到 FPGA 进入配置阶段。
8. 清除配置监控寄存器 (fpgamgrregs.mon) 中的 nSTATUS 位 (ns) 的状态。
9. 设置 ctrl 寄存器的 axicfggen 位为 1 以使能够发送配置数据到 FPGA。
10. 写入配置镜像到 FPGA 管理器模块配置数据寄存器 (fpgamgrdata) 的配置数据寄存器 (data)。也可以选用 DMA 控制器将配置镜像从外设器件传输到 FPGA 管理器。

11. 使用 fpgamrregs.mon 寄存器来监控 CONF_DONE (cd) 和 nSTATUS (ns) 位。
 - a. CONF_DONE = 1 和 nSTATUS = 1 表示成功配置。
 - b. CONF_DONE = 0 或 nSTATUS = 0 表示不成功配置。完成步骤 12 和 13, 然后再返回重复步骤 3 到 10 以重新加载配置镜像。
 12. 设置 ctrl 寄存器的 axicfggen 位为 0 以禁用 AXI 从器件的配置数据。
 13. 发送 FPGA 所需的 DCLK 以进入初始化阶段。
 - a. 如果 DCLK 未被使用, 那么将值 4 写入 DCLK 计数寄存器 (dclkcnt)。
 - b. 如果 DCLK 被使用, 那么将值 20,480 (0x5000) 写入 dclkcnt 寄存器。
 14. 轮询 DCLK 状态寄存器 (dclkstat) 的 dcntdone 位直到它更改为 1, 从而表明所有 DCLK 已经被发送。
 15. 写入 1 到 DCLK 状态寄存器的 dcntdone 位以清除完成状态标记。
 16. 读取 stat 寄存器的 mode 位以等待 FPGA 进入用户模式。
 17. 设置 ctrl 寄存器的 en 位为 0 以支持外部管脚驱动配置输入信号。
- 如果在进入用户模式前, HPS 在正常配置数据传输中复位, 那么软件会假设配置不成功。HPS 复位后, 软件必须重复完全配置的步骤。

部分重配置

部分重配置使您能够在其它部分保持运行时重配置部分的器件。当器件的 FPGA 部分处于用户模式时, HPS 执行部分重配置。以下序列建议软件执行部分配置的一种方式:

1. 读取 fpgamrregs 中 stat 寄存器的 mode 位以确保 FPGA 处于用户模式。
2. 设置 ctrl 寄存器的 cdratio 位以匹配部分重配置镜像的特征, 并且对于 16-bit 配置数据宽度, 设置 ctrl 寄存器的 cfgwidth 位为 0。
3. 设置 ctrl 寄存器的 en 位为 1 以使 FPGA 管理器可以对输入信号配置进行控制。
4. 设置 ctrl 寄存器的 prreq 位为 1 以置位 PR_REQUEST。
5. 写入 1 到 dclkcnt 寄存器以生成一个时钟周期的 DCLK 脉冲。
6. 轮询 fpgamrregs.mon 寄存器以观测 PR_READY(prr) 位。
 - a. 如果 PR_READY=1, 那么继续步骤 7。
 - b. 如果 PR_READY 是 0, 那么返回并重复步骤 5。请注意至少需要 16 个 DCLK 脉冲。
7. 写入值 3 到 dclkcnt 寄存器以生成 3 个时钟周期的 DCLK 脉冲。
8. 设置 ctrl 寄存器的 axicfggen 位为 1 以使能够将配置数据发送到 FPGA。
9. 写入部分重配置镜像到 FPGA 管理器 fpgamgrdata 寄存器中的 data 寄存器。也可以选择使用一个 DMA 将配置镜像从外设器件传输到 FPGA 管理器。

10. 轮询 fpgamrregs.mon 寄存器以观测 PR_DONE (prd)、PR_READY (prp)、PR_ERROR (pre) 和 CRC_ERROR (cre) 位直到位匹配表 A-12 中所示的完成状态的其中一个。

表 A-12. 部分重配置状态

位				部分重配置状态
PR_DONE	PR_READY	PR_ERROR	CRC_ERROR	
1	0	0	0	完成
0	0	1	0	完成时出现错误
0	0	0	1	完成时出现 SEU 事件 CRC 错误

11. 设置 ctrl 寄存器的 axicfgn 位为 0 以禁止将配置数据发送到 FPGA。
12. 设置 ctrl 寄存器的 prreq 位为 0 以置低 PR_REQUEST。
13. 写入值 128 到 dclkcnt 寄存器以生成 128 个时钟周期的 DCLK 脉冲。
14. 轮询 dclkstat 寄存器的 dcntdone 位直到它更改为 1，从而表示所有 DCLK 已经被发送。
15. 写入 1 到 dclkstat 寄存器的 dcntdone 位以清除完成状态标记。
16. 轮询 fpgamrregs.mon 寄存器以观测 PR_DONE (prd)、PR_READY (prp)、PR_ERROR (pre) 和 CRC_ERROR (cre) 位。当所有位设置为 0 时，FPGA 可以开始下一个传输。
17. 设置 ctrl 寄存器的 en 位为 0 以支持外部管脚驱动配置输入信号。

如果 HPS 在部分重配置期间复位，那么软件假设配置不成功。HPS 热复位后，软件必须重复部分配置的步骤。HPS 冷复位后，软件必须重复 A-17 的“完全配置”的步骤。



要了解关于配置模式和管脚设置的更多信息，请参考 *Cyclone V 器件手册* 第 3 卷的 *FPGA 管理器* 章节。

文档修订历史

表 A-13 显示了该文档的修订历史。

表 A-13. 文档修订历史

日期	版本	修订内容
2012 年 11 月	1.3	<ul style="list-style-type: none">■ 扩展了共享内存模块表。■ 添加了 CSEL 表。■ 额外的少量文本编辑。
2012 年 6 月	1.2	更新了 HPS 引导和 FPGA 配置部分。
2012 年 5 月	1.1	<ul style="list-style-type: none">■ 更新了 HPS 引导部分。■ 添加了用于 HPS 引导的闪存器件的信息。■ 添加了关于 FPGA 配置模式的信息。
2012 年 1 月	1.0	初稿。

