

硬件处理器系统 (HPS) 提供一个四 (quad) 串行外围接口 (SPI) flash 控制器，对串行 NOR flash 器件进行访问。quad SPI flash 控制器支持标准 SPI flash 器件，以及高性能的 dual 和 quad SPI flash 器件。quad SPI flash 控制器基于 Cadence® Quad SPI Flash Controller (QSPI_FLASH_CTRL)。

Quad SPI Flash 控制器的特性

quad SPI flash 控制器支持下面特性：

- 单 (single)，双 (double) 和四 (quad) I/O 命令
- 高达 108 MHz 的器件频率
- 直接访问和间接访问模式
- 外部 DMA 控制器支持间接访问
- 极性，相位和延迟可配置
- 可编程的写保护区域
- 带有 ECC 功能的本地 buffer 用于间接访问
- 高达四个器件
- eXecute-In-Place (XIP) flash 器件

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

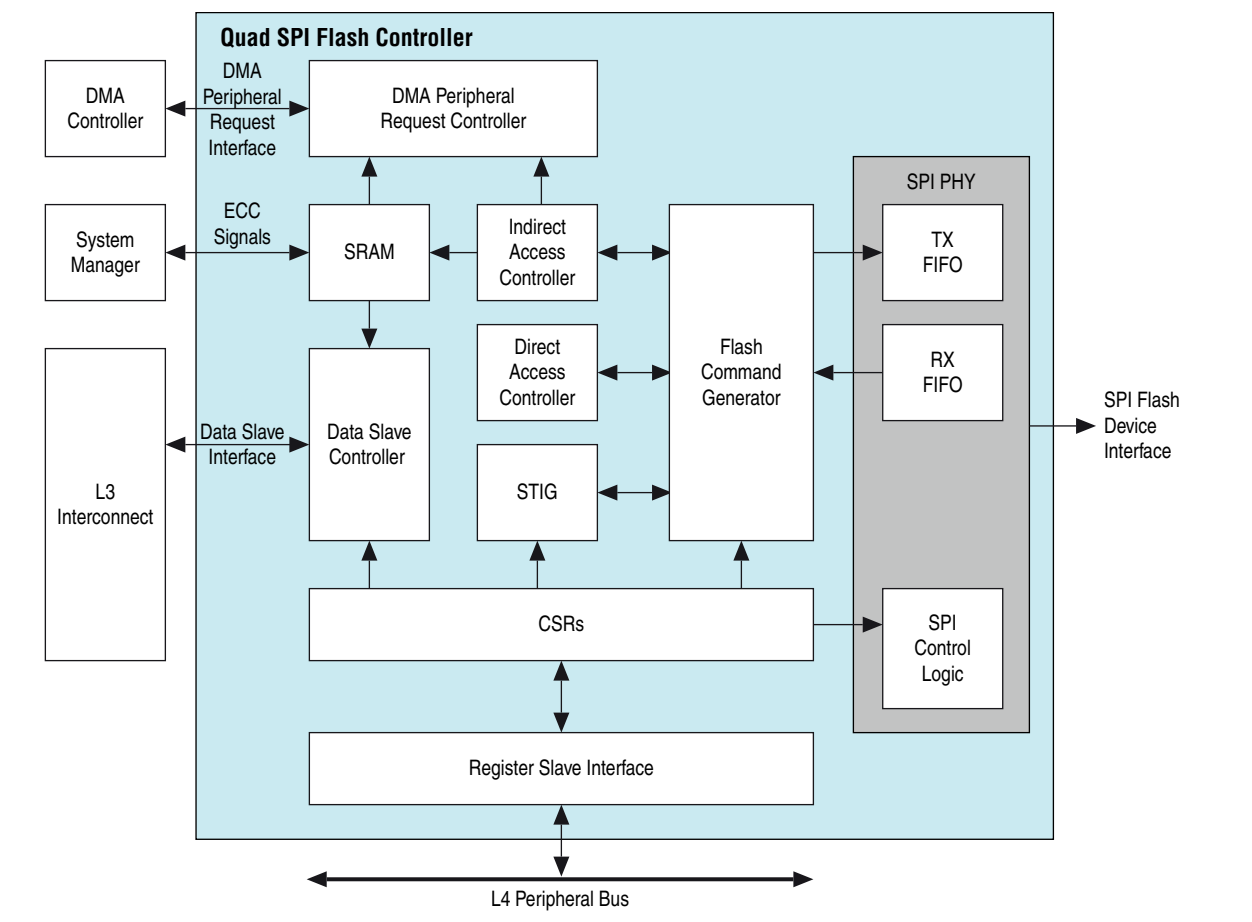
Portions © 2011 Cadence Design Systems, Inc. Used with permission. All rights reserved worldwide. Cadence and the Cadence logo are registered trademarks of Cadence Design Systems, Inc. All others are the property of their respective holders.



Quad SPI Flash 控制器结构图与系统集成

图 12-1 显示了 quad SPI flash 控制器结构图。

图 12-1. Quad SPI Flash 控制器结构图



quad SPI 控制器包含下面模块和接口:

- 寄存器从接口 — 提供对控制和状态寄存器 (CSR) 访问的接口
- 数据从控制器 — 提供下面功能的从接口和控制器:
 - 执行到 level 3 (L3) 互联的数据传递
 - 验证输入访问
 - 执行字节或半字重排序
 - 执行写保护
 - 转送传递请求到直接和间接控制器
- 直接访问控制器 — 提供对闪存的存储器映射从直接访问
- 间接访问控制器 — 通过本地缓冲和软件传递请求来提供对闪存的更高性能访问
- 软件触发指令生成器 (STIG) — 通过 flash 命令寄存器 (flashcmd) 生成 flash 命令并提供对闪存的低级访问

- Flash命令生成器—根据直接或间接访问控制器或者STIG的指令生成flash命令和地址指令
- DMA 外围请求控制器 — 发出请求到 DMA 外围请求接口，与外部 DMA 控制器进行通信
- SPI PHY— 串行传递数据和命令到外部 SPI flash 器件

Quad SPI Flash 控制器的功能描述

本小节介绍了 quad SPI flash 控制器的功能。

概述



本小节中使用的术语在接下来的章节中被详细定义。

quad SPI flash 控制器使用寄存器从接口选择操作模式和配置用于数据传递的数据从接口。quad SPI flash 控制器使用数据从接口进行直接和间接访问，使用寄存器从接口用于 STIG 操作和 SPI 传统模式访问。

对数据从接口的访问被转送到直接或间接访问控制器。如果访问地址在配置的间接地址范围内，那么该访问被发送到间接访问控制器。

数据从接口

quad SPI flash 控制器对直接，间接和 SPI 旧式模式访问使用数据从接口。关于这些模式的详细信息，请参考下面章节。

数据从接口为 32 bit 宽，允许字节，半字和字访问。

对于写访问，仅支持增量式突发和 size 1, 4, 8 和 16 传递。对于读访问，支持所有的突发类型和尺寸。

寄存器从接口

quad SPI flash 控制器使用寄存器从接口通过 quad SPI 配置寄存器来配置 quad SPI 控制器，通过 STIG 中的 flashcmd 寄存器在软件控制下访问闪存。

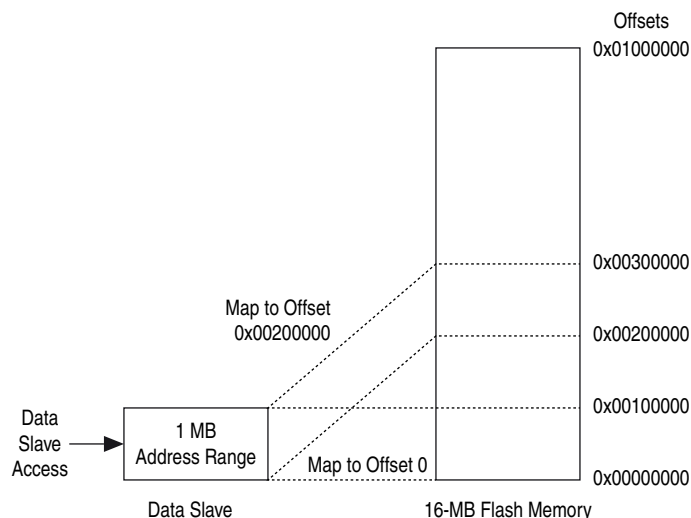
直接访问模式

在直接访问模式中，对数据从接口的访问会触发对闪存的读或写命令。要使用直接访问模式，需要通过使能 quad SPI 配置寄存器 (cfg) 的直接访问控制器比特 (endiracc) 来使能直接访问控制器。

外部主器件（例如处理器）通过对数据从接口的读或写操作触发直接访问控制器。数据从接口释放 1 MB 窗口到闪存中。您可以重映射该窗口到闪存中的任意 1 MB 位置。

图 12-2 显示了一个重映射实例。

图 12-2. 数据从接口重映射实例



要重映射数据从接口来访问闪存中的其它 1 MB 区域，需要在 `cfg` 寄存器的使能 AHB 地址重映射域 (`enahbremap`)。所有数据从接口的输入访问都被重映射到重映射寄存器 (`remapaddr`) 中指定的偏移。

输入地址的 20 LSB 用于访问 1 MB 区域，较高比特被忽略。



quad SPI 控制器对连接的闪存空间外的访问不发出任何错误状态。

间接访问模式

在间接访问模式中，flash 数据暂时缓存在 quad SPI 控制器的 SRAM 中。软件通过寄存器从接口控制和触发间接访问。控制器通过数据从接口完成数据的传递。

间接读操作

间接读操作读取闪存中的数据时先将这些数据存储在 SRAM 中，然后通过数据从接口将这些数据传递到外部主接口。间接读操作读由下面的寄存器控制：

- 间接读传递寄存器 (`indrd`)
- 间接读传递流水线寄存器 (`indrwater`)
- 间接读传递起始地址寄存器 (`indrdstaddr`)
- 间接读传递字节数寄存器 (`indrdcnt`)
- 间接地址触发寄存器 (`indaddrtrig`)

在发出间接读操作前需要配置这些寄存器。需要在 `indrdstaddr` 寄存器中定义起始地址，要获取的字节总数在 `indrdcnt` 寄存器中指定。将 1 写入到 `indrd` 寄存器的起始间接读比特 (`start`) 将触发对闪存的间接读操作，以使用返回的数据填充 SRAM。

要从闪存中读取数据到 SRAM 中，外部主接口需要发出 32-bit 读传输到数据从接口。读访问的地址必须在间接地址范围内。您可以通过 `indaddrtrig` 寄存器配置间接地址。外部主接口能够发出 32-bit 读操作直到间接传递的最后一个字。在最后的读操作中，外部主接口可以发出一个 32-bit，16-bit 或者 8-bit 读以结束传递。如果在最后的传递上读取的数据少于四个字节，那么外部主接口仍然能够发出 32-bit 读，quad SPI 控制器将使用 0 填充响应数据的高位比特。

假设在 quad SPI 控制器接收到数据从读的同时所请求的数据出现在 SRAM 中，数据从 SRAM 获取并且用最短延迟完成突发读响应。如果请求的数据没有立即出现在 SRAM 中，数据从接口会进入等待状态直到数据从闪存读取到 SRAM 中。一旦外部主接口从 SRAM 中读取了数据，quad SPI 控制器就会释放 SRAM 中的相关资源。如果 SRAM 是满的，那么 SPI 接口上的读操作被反压 (backpressured)，直到 SRAM 中有可用空间。quad SPI 控制器完成当前所有的读突发，等待 SRAM 释放，并在前一个突发终止的地址发出一个新的读突发。

处理器也可以使用 SRAM 填充寄存器 (`sramfill`) 中的 SRAM 填充水平来控制何时应该从 SRAM 中获取数据。

另外一个方法是使用在 `indrwater` 寄存器中配置的 SRAM 填充水平水线 (`fill level watermark`)。当 SRAM 填充水平超过水线水平时生成间接传递水线中断。通过在 `indrwater` 寄存器中写入 0 可以禁用水线功能。

对于 quad SPI 控制器读取的和存储在 SRAM 中的数据最后字节，如果水线水平大于零，那么即使实际的 SRAM 填充水平没有超过水线时会生成间接传递水线中断。

如果读访问的地址在间接触发地址的范围外，那么会出现下面其中的一个情况：

- 当直接访问模式使能时，读操作使用直接访问模式。
- 当直接访问模式禁用时，从接口返回一个错误到请求的主接口。

通过将 `indr` 寄存器中的取消间接读比特 (`cancel`) 设为 1，您能够取消间接操作。关于更多信息，请参考第 12 - 13 页 “间接读操作”。

间接写操作

间接写操作编程从 SRAM 到闪存的数据。间接写操作由下面寄存器控制：

- 间接写传递寄存器 (`indwr`)
- 间接写传递水线寄存器 (`indwrwater`)
- 间接写传递起始地址寄存器 (`indwrstaddr`)
- 间接写传递字节数寄存器 (`indwrcnt`)
- `indaddrtrig` 寄存器

在发出间接写操作之前需要配置这些寄存器。起始地址需要在 `indwrstaddr` 寄存器中定义，要写入的字节总数在 `indwrcnt` 寄存器中指定。`indwr` 寄存器的起始间接写比特 (`start`) 触发从 SRAM 到闪存的间接写操作。

要将 SRAM 中数据写入到闪存中，外部主接口发出 32-bit 写传输到数据从接口。写访问的地址必须在间接地址范围内。您可以通过 `indaddrtrig` 寄存器来配置间接地址。外部主接口能够发出 32-bit 写直到间接传递的最后一个字。在最后的写操作中，外部主接口可以发出一个 32-bit，16-bit 或者 8-bit 写以完成传递。如果在最后的传递上写入的数据少于四个字节，那么外部主接口仍然能够发出 32-bit 写，quad SPI 控制器将丢弃额外字节。

SRAM 容量能够限制 quad SPI 控制器从外部主接口接收到的数据量。如果 SRAM 在写访问时不是满的, 那么数据写入 SRAM 的延迟是最小的。如果外部主接口试图存储超过 SRAM 所能接受的数据, 那么 quad SPI 控制器会发出反压信号使外部主接口处于等待状态。通过将 SRAM 中的数据存储在闪存中以释放 SRAM 资源时, SRAM 可以接收来自外部主接口的更多的数据。当 SRAM 保持一个大于或等于 flash 页面大小的字节数时, 或者当 SRAM 保持当前间接传递的所有剩余字节时, quad SPI 控制器将启动对闪存的写操作。

处理器也可以使用 SRAM 填充水平 (在 `sramfill` 寄存器中) 来控制何时对 SRAM 写入更多数据。

或者, 您可以配置 `indwrwater` 寄存器中 SRAM 的填充水平水线。当 SRAM 填充水平低于水线水平时, 将生成一个间接传递水线中断以告知软件写入下一页数据到 SRAM。由于 quad SPI 控制器启动对闪存的非数据结束 (non-end-of-data) 写操作, 所以您必须将水线水平设成一个大于一个 flash 页面的值, 以避免系统停止。通过在 `indrwater` 寄存器中写入 0 可以禁用水印功能。

如果写访问的地址在间接触发地址的范围以外, 那么会出现下面其中一个情况:

- 当直接访问模式使能时, 写操作使用直接访问模式。
- 当直接访问模式禁用时, 从接口返回一个错误到请求的主接口。

通过将 `indwr` 寄存器中的取消间接写比特 (`cancel`) 设为 1, 您能够取消间接操作。

关于更多信息, 请参考第 12 - 14 页 “间接写操作”。

连续读和写

通过快速连续地触发 `indr` 或者 `indwr` 寄存器中的起始比特两次, 可以同时触发两个间接操作。在第一个操作进行期间可以触发第二个操作。例如, 在一个间接写操作进行期间, 软件可以触发一个间接读或写操作。在软件触发每个传输操作之前, 必须正确地配置相应的起始和计数寄存器。

此方法允许在一个间接操作的结束与第二个操作的开始之间有短周转时间。尝试按排两个以上操作会导致生成一个间接读拒绝中断。

本地存储缓存器

SRAM 本地存储缓存器 $128 \times 32\text{-bit}$ (总共 512 个字节) 存储器, 包括对 ECC 的支持。当检测到可纠正单比特错误或者不可纠正双比特错误时, ECC 逻辑提供输出通知系统管理程序。ECC 逻辑也支持单比特和双比特错误的注入, 用于测试目的。



关于更多信息, 请参考 *Cyclone V 器件手册* 卷 3 中的 *System Manager* 章节。

SRAM 有两个分区, 低分区保留用于间接读操作, 高分区保留用于间接写操作。分区大小在 SRAM 分区寄存器 (`srampart`) 中基于 32-bit 字大小指定。例如, 要指定四个字节的存储容量, 需要写入值 1。写入到间接读分区大小域 (`addr`) 中的值定义了对间接读操作保留的入口数量。例如, 写入值 32 (0x20) 将 128-entry SRAM 分成 32 个入口 (25%) 用于读和 96 个入口 (75%) 用于写。

DMA 外设请求控制器

DMA 外设请求控制器仅用于操作的间接模式, 其中的数据暂时存储在 SRAM 中。quad SPI flash 控制器使用 DMA 外设请求接口来触发外部 DMA 执行存储器与 quad SPI 控制器之间的数据传递。

有两种 DMA 外设请求接口，一种用于间接读操作，另一种用于间接写操作。DMA 外设请求控制器能够发出两种类型的 DMA 请求（单一或突发）到外部 DMA。单一或突发请求的字节数在 DMA 外围寄存器 (dmaper) 的单字节数 (numsglreqbytes) 和突发字节数 (numburstreqbytes) 域中指定。DMA 外设请求控制器通过将总字节数除以突发请求中指定字节数，然后将剩余的字节数除以单一请求中的字节数，将要传递的总数据分成一组 DMA 突发和单一请求。



当对 DMA 控制器编程时，突发请求大小必须匹配 quad SPI 控制器中设置的突发请求大小，以避免迅速达到上溢或下溢的情况。

对于间接读操作，接收到来自闪存的数据并写入到 SRAM 中之后，DMA 外设请求控制器才发出 DMA 请求。DMA 请求发出的速率取决于水线水平。indrdwater 寄存器定义了 DMA 外设请求控制器能够发出 DMA 请求的最小填充水平（以字节表示）。该数值越高，外部 DMA 移动数据前必须在 SRAM 中缓存的数据就越多。当 SRAM 填充水平超过水线水平时会生成传递水线达到的中断。

例如，考虑下面的情况：

- 使用间接模式要读取的数据总量是 256 字节
- SRAM 水线水平设为 128 字节
- 软件将突发类型传递大小设为 64 字节

在这些情况下，当 SRAM 填充水平超过 128 字节（水线水平），DMA 外设请求控制器会发出第一个 DMA 突发请求。只要 SRAM 中有足够的数据来执行突发类型请求，DMA 外设请求控制器就触发连续的 DMA 突发请求。在此实例中，DMA 外设请求控制器能够发出至少两个连续的 DMA 突发请求来传递总共 128 个字节。如果 SRAM 中有足够的数据，那么 DMA 外设请求控制器将立即请求第三个 DMA 突发。否则 DMA 外设请求控制器等待 SRAM 填充水平再次超过水线水平以触发下一个突发请求。水线水平被触发时，SRAM 中有足够的数据来执行第三个和第四个突发请求以完成完整的传输。

对于间接写操作，DMA 外设请求控制器在触发传递后立即发出 DMA 请求，继续此操作直到传递了整个间接写传递。发出 DMA 请求的速率取决于水线水平。indwrwater 寄存器定义最大填充水平（以字节表示），在此填充水平上控制器能够发出第一个 DMA 突发或者单一请求。当 SRAM 填充水平低于水线水平时会生成传递水线达到的中断。当 SRAM 中有一个 flash 页面的数据时，quad SPI 控制器会启动从 SRAM 到闪存的写操作。

软件能够通过 cfg 寄存器的 enddma 域禁用 DMA 外设请求接口。如果一个主器件（非 DMA）执行间接操作的数据传递，那么必须禁用 DMA 外设请求接口。默认情况下，间接水线寄存器设为零，这意味着 DMA 外设请求控制器能够尽快发出 DMA 请求。关于 HPS DMA 控制器的详细信息，请参考 *Cyclone V 器件手册* 卷 3 中的 *DMA Controller* 章节。

STIG 操作

STIG 提供一种软件方法来直接访问 flash 器件寄存器。flashcmd 寄存器使用下面的参数来定义对 flash 器件发出的命令：

- 指令操作码 (Instruction opcode)
- 地址字节数 (Number of address bytes)
- 空字节数 (Number of dummy bytes)
- 写数据字节数 (Number of write data bytes)
- 写数据 (Write data)

■ 读数据字节数 (Number of read data bytes)

通过 flash 命令地址寄存器 (flashcmdaddr) 指定地址。一旦指定了这些设置, 软件就能够通过 flashcmd 寄存器的执行命令域 (execcmd) 来触发命令, 并且通过轮询 flashcmd 寄存器的命令执行状态比特 (cmdexecstat) 来等待它的完成。从 flash 命令读数据较低 (flashcmdrddatalo) 和 flash 命令读数据较高 (flashcmdrddataup) 寄存器每个命令最多可读取 8 个字节的数据, 同样, 对 flash 命令写数据较低 (flashcmdwrdatalo) 和 flash 命令写数据较高 (flashcmdwrdataup) 寄存器每个命令最多可写入 8 个字节的数据。

通过 STIG 发出的命令具有比所有其它读访问更高的优先权, 从而中断任何由直接或间接控制器正在请求的读命令。然而, STIG 不中断通过直接或间接访问控制器发出的写序列。在这些情况下, flashcmd 寄存器的 cmdexecstat 比特可能需要较长的时间来指示操作完成。



Altera 建议使用 STIG 来 (而不是 SPI 旧式模式) 访问 flash 器件寄存器和执行擦除操作。

SPI 旧式模式 (SPI Legacy Mode)

SPI 旧式模式支持软件直接访问内部 TX FIFO 和 RX FIFO 缓存, 因此旁路直接, 间接和 STIG 控制器。当旧式模式使能时, 软件通过数据从接口对任意地址写入任意值来访问 TX FIFO 和 RX FIFO 缓存。您可以通过 cfg 寄存器的旧式 IP 模式使能比特 (enlegacyip) 使能旧式模式。

旧时模式支持用户对 flash 器件发出任何的 flash 指令, 但为有效地管理 FIFO 缓存的填充水平, 旧时模式承受很大的软件开销。旧式 SPI 模式本质上是双向的, 芯片选择 (chip select) 功能使能时, 在两个方向上连续传递数据。如果驱动器只需要从 flash 器件读取数据, 必须写入空数据 (dummy data) 来确保芯片选择功能有效, 对于写操作也是一样的。

例如, 对一个具有三个地址字节的 flash 器件执行基本的读操作, 软件必须写入一共 8 个字节到 TX FIFO 缓存。第一个字节可能会是指令操作代码, 接下来的三个字节是地址, 最后四个字节可能是空数据以确保读数据返回时芯片选择功能有效。类似地, 由于写入了 8 个字节到 TX FIFO 缓存, 因此软件应该期望返回 RX FIFO 缓存中的 8 个字节。其中的头 4 个字节可能被丢弃, 剩下最后 4 个字节保持器件的数据读操作。

由于 TX FIFO 和 RX FIFO 缓存是四字节深度, 因此软件必须维持 FIFO 缓存水平以确保 TX FIFO 缓存不会下溢和 RX FIFO 缓存不会上溢。提供的中断用于指示填充水平何时超过水准水平, 通过 TX 阈值寄存器 (txtresh) 和 RX 阈值寄存器 (rxtresh) 可以配置这些水准水平。

配置 flash 器件

对于读和写访问, 软件必须初始化器件读指令寄存器 (devrd) 和器件写指令寄存器 (devwr)。这些寄存器包括用于初始化指令操作代码的域, 该域也用于指令类型, 以及对于地址和数据传递, 指令使用单管脚, 双管脚还是四管脚。要确保四 SPI 控制器能够从复位状态运行, 操作代码寄存器要复位成与单 I/O flash 器件兼容的操作代码。

quad SPI flash 控制器使用 devrd 寄存器的指令传递宽度域 (instwidth) 来对写操作以及读操作设置指令传递宽度。在 devwr 寄存器中没有 instwidth 域。如果指令类型被设成双或四模式, 那么这两个寄存器的地址传递宽度 (addrwidth) 和数据传递宽度 (datawidth) 域是冗余的, 因为地址和数据类型是基于指令类型的。因此, 软件能够支持不常使用的 flash 指令, 其中操作代码, 地址和数据在两个通道或者四个通道上发送。对于大多数指令, 甚至包括双和四指令, 操作代码串行发送到 flash 器件,

Micron N25Q128 flash 器件支持指令通过两个或四个通道发送操作代码。表 12-1 显示了对于 Micron N25Q128 器件支持的指定读写指令，软件应该如何配置 quad SPI 控制器。

表 12-1. Micron N25Q128 器件的 Quad SPI 配置

指令	操作代码使用的通道	用于发送地址的通道	用于发送数据的通道	instwidth 值	addrwidth 值	datawidth 值
读指令						
Read	1	1	1	0	0	0
Fast read	1	1	1	0	0	0
Dual output fast read (DOFR)	1	1	2	0	0	1
Dual I/O fast read (DIOFR)	1	2	2	0	1	1
Quad output fast read (QOFR)	1	1	4	0	0	2
Quad I/O fast read (QIOFR)	1	4	4	0	2	2
Dual command fast read (DCFR)	2	2	2	1	Don' t care	Don' t care
Quad command fast read (QCFR)	4	4	4	2	Don' t care	Don' t care
写指令						
Page program	1	1	1	0	0	0
Dual input fast program (DIFP)	1	1	2	0	0	1
Dual input extended fast program (DIEFP)	1	2	2	0	1	1
Quad input fast program (QIFP)	1	1	4	0	0	2
Quad input extended fast program (QIEFP)	1	4	4	0	2	2
Dual command fast program (DCFP)	2	2	2	1	Don' t care	Don' t care
Quad command fast program (QCFP)	4	4	4	2	Don' t care	Don' t care

XIP 模式

如果 flash 器件支持 XIP 模式，那么 quad SPI 控制器就支持 XIP 模式。取决于 flash 器件，XIP 模式将 flash 器件置于只读模式，从而减少命令开销。

quad SPI 控制器必须通过发送模式比特来指导 flash 器件进入 XIP 模式。当 cfg 寄存器的下个读操作进入 XIP 模式比特 (enterxipnextrd) 设为 1 时，quad SPI 控制器和 flash 器件可在下一个读指令进入 XIP 模式。当 cfg 寄存器的立即进入 XIP 模式比特 (enterxipimm) 设为 1 时，quad SPI 控制器和 flash 器件立即进入 XIP 模式。

当 cfg 寄存器的 enterxipnextrd 或者 enterxipimm 比特设为 0 时，quad SPI 控制器和 flash 器件在下一个读指令退出 XIP 模式。

关于详细信息，请参考第 12-14 页 “XIP 模式操作”。

写保护

通过编程控制器可以对 flash 器件的指定区域进行写保护。受保护的区域被定义成一组数据块，由起始和结束数据块指定。对受保护区域进行写操作会生成一个错误，并触发一个中断。

您可以通过器件大小寄存器 (devsz) 的每个数据块字节数比特 (bytespersubsector) 指定每个数据块的字节数量来定义数据块大小。降低写保护寄存器 (lowwrprot) 指定受保护区域中的第一个 flash 数据块。较高写保护寄存器 (uppwrt) 指定受保护区域中的最后一个 flash 数据块。

写保护寄存器 (wrprot) 的写保护使能比特 (en) 使能和禁用写保护。wrprot 寄存器的写保护反转比特 (inv) 反转保护定义，使 lowwrprt 和 uppwrt 指定的区域不受保护，并使该区域外的所有闪存被保护。

数据从接口顺序访问检测

quad SPI flash 控制器通过比较当前访问与之前访问来检测对数据从接口的顺序访问。当一个访问满足下面条件时，它是一个顺序访问：

- 当前访问的地址按顺序跟着之前访问的地址。
- 当前访问的方向（读或写）与之前访问的方向相同。
- 当前访问的大小（字节，半字或者字）与之前访问的大小相同。

当访问被检测为非顺序的，对 flash 器件的顺序访问会被终止，开始一个新的顺序访问。Altera 建议顺序访问数据从接口。顺序访问有更少的命令开销，从而增加数据吞吐量。

时钟

quad SPI 控制器有两个时钟输入 (l4_mp_clk 和 qspi_clk) 和一个时钟输出 (sclk_out)。quad SPI flash 控制器使用 l4_mp_clk 时钟对数据从接口传递和存储器从接口访问提供时钟。qspi_clk 时钟是 quad SPI 控制器的参考时钟，并用于串化数据和驱动外部 SPI 接口。sclk_out 时钟是输出到 flash 器件的时钟源。

qspi_clk 时钟频率必须大于两倍的 l4_mp_clk。通过 cfg 寄存器的波特率除数域 (bauddiv) 来分频 qspi_clk 时钟得到 sclk_out 时钟。



关于更多信息，请参考 *Cyclone V Device Handbook* 卷 3 中的 *Clock Manager* 章节。

复位

单一复位信号 (qspi_flash_rst_n) 被提供作为 quad SPI 控制器的输入。复位管理器在冷或热复位上驱动此信号。



关于更多信息，请参考 *Cyclone V Device Handbook* 卷 3 中的 *Reset Manager* 章节。

中断

所有中断源组合成一个水平敏感，高有效的中断 (qspi_intr_in)。软件通过读取中断状态寄存器 (irqstat) 能够决定中断源。默认情况下，当软件写入中断状态寄存器时，中断源被清零。中断通过中断屏蔽寄存器 (irqmask) 是独立可屏蔽的。表 12-2 列出了 irqstat 寄存器中的中断源。

表 12-2. irqstat 寄存器中的中断源

中断源	说明
检测到的下溢	为 0 时，没有检测到下溢。为 1 时，非常慢地提供数据从写数据。当提供数据从写数据太慢而不能跟进请求的写操作时会出现此情况。该比特只被系统复位复位，并只有在写入寄存器时清零。
间接操作完成	控制器已经完成一个触发的间接操作。
间接读拒绝	请求了一个间接操作，但不能被接受，因为队列中已经有两个间接操作了。
受保护区写尝试	试图写入被保护的区域，但被拒绝了。
检测到的非法数据从访问	已经检测到非法数据从访问。数据从包装突发和分开并重试访问的使用能够导致这一中断。这通常表明 FPGA 架构中的软核主器件正试图以一种不支持的方式访问 HPS。
达到传递水印	已经达到间接传递水印水平。
接收上溢	此情况仅出现在旧式 SPI 模式。为 0 时，没有检测到上溢。为 1 时，出现了 RX FIFO 缓存的上溢。该比特只被系统复位复位，并只有在写入寄存器时清零。如果在读取寄存器的同时出现对 RX FIFO 缓存的新的写操作，那么此标志比特 (flag) 设为 1。
TX FIFO 未满	此情况仅出现在旧式 SPI 模式。为 0 时，TX FIFO 缓存是满的。为 1 时，TX FIFO 缓存未满。
TX FIFO 已满	此情况仅出现在旧式 SPI 模式。为 0 时，TX FIFO 缓存未满。为 1 时，TX FIFO 缓存是满的。
RX FIFO 未空	此情况仅出现在旧式 SPI 模式。为 0 时，RX FIFO 缓存时空的。为 1 时，RX FIFO 缓存未空。
RX FIFO 已满	此情况仅出现在旧式 SPI 模式。为 0 时，RX FIFO 缓存未满。为 1 时，RX FIFO 缓存是满的。
间接读分区上溢	SRAM 的间接读分区是满的，不能立即完成间接操作。

接口信号

quad SPI 控制器提供四个片选输出来支持高达四个外部 quad SPI flash 器件的控制。根据器件用于的操作模式（单 (single)，双 (double) 或四 (quad) 操作模式），这些输出可用于不同的目的。表 12-3 列出了对于每种操作模式的 quad SPI 控制器接口信号的 I/O 管脚使用。

表 12-3. 接口信号

信号	模式	方向	功能
data[0]	Single	输出	数据输出 1
	Dual or quad	双向	数据 I/O 0

表 12 - 3. 接口信号

信号	模式	方向	功能
data[1]	Single	输入	数据输入 0
	Dual or quad	双向	数据 I/O 1
data[2]	Single or dual	输出	低有效写保护
	Quad	双向	数据 I/O 2
data[3]	Single, dual, or quad	双向	数据 I/O 3
ss_n[0]	Single, dual, or quad	输出	低有效从选择 0
ss_n[1]			低有效从选择 1
ss_n[2]			低有效从选择 2
ss_n[3]			低有效从选择 3
sclk			串行时钟

Quad SPI Flash 控制器编程模型

本小节介绍了 quad SPI 控制器的编程模型。

设置 Quad SPI Flash 控制器

下面的步骤描述了如何设置 quad SPI 控制器：

1. 等待直到所有暂挂的操作完成。
2. 通过 `cfg` 寄存器的 quad SPI 使能域 (`en`) 来禁用 quad SPI 控制器。
3. 使用您希望的指令类型来更新 `devrd` 寄存器的 `instwidth` 域，用于间接和直接读写操作。
4. 如果 `devrd` 寄存器的模式比特使能比特 (`enmodebits`) 使能，那么更新模式比特寄存器 (`modebit`)。
5. 根据需要更新 `devsz` 寄存器。该寄存器的部分或全部在初始化后可能被更新了。地址字节的数量是读写操作所要求的一个关键配置设置。执行任何的写操作需要每个页面的字节数量。每个器件模块的字节数仅在使用写保护功能时需要。
6. 更新器件延迟寄存器 (`delay`)。此寄存器支持用户字每次 flash 复位后调整如何驱动芯片选择。每个器件可能有不同的时序要求。如果串行时钟频率增加，那么这些时序要求会变得更加严格。该寄存器中指定的数量基于 `qspi_clk` 时钟的周期。

例如，从选择 (`slave select`) 被置低后，在它被重新置位前，某些器件需要 50 ns 最短时间。当器件运行在 100 MHz，时钟周期是 10 ns，因此需要额外的 40 ns。如果 `qspi_clk` 时钟运行在 400 MHz (2.5 ns 周期)，那么在 `delay` 寄存器的芯片选择置低域 (`nss`) 对时钟延迟要指定一个至少为 16 的值。
7. 根据需要更新 `remapaddr` 寄存器。该寄存器仅影响直接访问模式。
8. 当需要写保护时，设置并使能写保护寄存器 (`wrprot`，`lowwrprot` 和 `uppwprot`)。
9. 通过 `irqmask` 寄存器使能所需要的中断。
10. 设置 `cfg` 寄存器的 `bauddiv` 域来定义目标器件的所需时钟频率。

11. 根据需要更新读数据采集寄存器 (read data capture register (rddatacap))。当采集到读数据时, 该寄存器延迟, 当器件到 quad SPI 控制器的读数据路径很长时和器件时钟频率很高时, 该寄存器是有帮助的。
12. 通过 cfg 寄存器的 en 域使能 quad SPI 控制器。

间接读操作

下面步骤描述了对 DMA 禁用的间接读操作设置 quad SPI 控制器的一般软件流程:

1. 执行第 12 - 12 页 “设置 Quad SPI Flash 控制器” 中描述的步骤。
2. 在 indrdstaddr 寄存器中设置闪存起始地址。
3. 在 indrdcnt 寄存器中设置要传递的字节数。
4. 在 indaddrtrig 寄存器中设置间接传递触发地址。
5. 通过 irqmask 寄存器设置所需中断。
6. 如果使用水线水平, 那么通过 indrdwater 寄存器设置 SRAM 水线水平。
7. 通过将 indrd 寄存器的 start 域设置成 1 来开始间接读操作。
8. 在 sramfill 寄存器中使用水线水平中断或者轮询 SRAM 填充水平来决定何时 SRAM 中有足够的数据。
9. 发出一个读传输到间接地址以访问 SRAM。如果需要更多的读传输来完成间接读传递, 那么重复步骤 8。
10. 使用间接完成中断来确定间接读操作何时完成, 或者通过 indrd 寄存器的间接完成状态比特 (ind_ops_done_status) 轮询间接读操作的完成状态。

下面步骤描述了对 DMA 使能的间接读操作设置 quad SPI 控制器的一般软件流程:

1. 执行第 12 - 12 页 “设置 Quad SPI Flash 控制器” 中描述的步骤。
2. 在 indrdstaddr 寄存器中设置闪存起始地址。
3. 在 indrdcnt 寄存器中设置要传递的字节数。
4. 在 indaddrtrig 寄存器中设置间接传递触发地址。
5. 在 dmaپر 寄存器中设置单一型和突发类型 (single and burst type)DMA 传递的字节数。
6. 可选择在 indrdwater 寄存器中设置 SRAM 水线水平来控制发出 DMA 请求的速率。
7. 通过将 indrd 寄存器的 start 域设置成 1 来开始间接读访问。
8. 使用间接完成中断来确定间接读操作何时完成, 或者通过 indrd 寄存器的 ind_ops_done_status 域轮询间接读操作的完成状态。

间接写操作

下面步骤描述了对 DMA 禁用的间接写操作设置 quad SPI 控制器的一般软件流程:

1. 执行第 12 - 12 页 “设置 Quad SPI Flash 控制器” 中描述的步骤。
2. 在 indwrstaddr 寄存器中设置闪存起始地址。
3. 在 indwrcnt 寄存器中设置要传递的字节数。
4. 在 indaddrtrig 寄存器中设置间接传递触发地址。

5. 通过中断屏蔽寄存器 (irqmask) 设置需要的中断。
6. 可选择设置 indwrwater 寄存器中的 SRAM 水线水平以控制发出 DMA 请求的速率。设置的值必须大于一个 flash 页面。关于详细信息, 请参考第 12 - 5 页 “间接写操作”。
7. 通过将 indwr 寄存器的 start 域设为 1 来开始间接写操作。
8. 在 sramfill 寄存器中使用水线水平中断或者轮询 SRAM 填充水平来决定何时 SRAM 中有足够的数据。
9. 发出一个写传输到间接地址以对 SRAM 写入一个 flash 页面的数据。如果需要更多写传输来完成间接写传递, 那么重复步骤 8。最后的写操作可能少于一个页面的数据。

下面步骤描述了对 DMA 使能的间接写操作设置 quad SPI 控制器的一般软件流程:

1. 执行第 12 - 12 页 “设置 Quad SPI Flash 控制器” 中的步骤。
2. 在 indwrstaddr 寄存器中设置闪存起始地址。
3. 在 indwr 寄存器中的 indcnt 域中设置要传递的字节数。
4. 在 indaddrtrig 寄存器中设置间接传递触发地址。
5. 在 dmaper 寄存器中设置单一型和突发类型 (single and burst type) DMA 传递的字节数。
6. 可选择设置 indwrwater 寄存器中的 SRAM 水线水平以控制发出 DMA 请求的速率。设置的值必须大于一个 flash 页面。关于详细信息, 请参考第 12 - 5 页 “间接写操作”。
7. 通过将 indirwr 寄存器的 start 域设为 1 来开始间接写访问。
8. 使用间接完成中断确定间接读操作何时完成, 或者通过 indwr 寄存器的 ind_ops_done_status 域轮询间接读操作的完成状态。

XIP 模式操作

本小节介绍了进入和退出 XIP 模式。大多数 SPI flash 器件都支持 XIP 模式。然而, flash 器件制造商不采用统一的标准方法, 大多数使用签名比特, 签名比特在地址字节之后立即发送到器件。某些器件使用签名比特, 同时也需要 flash 器件配置寄存器写操作来使能 XIP 模式。

进入 XIP 模式

以下部分介绍了各种类型 flash 器件进入 XIP 模式的软件步骤。

支持 Basic-XIP 的 Micron Quad SPI Flash 器件

要在支持 Basic-XIP 的 Micron quad SPI flash 器件中进入 XIP 模式, 需要执行下面的步骤:

1. 如果想要在退出时还原模式比特的值, 则需要以模式比特保存这些值。
2. 禁止直接访问控制器和间接访问控制器以确保没有新的读或写访问发送到 flash 器件。
3. 将 modebit 寄存器中的 XIP 模式设为 0x80。

4. 通过将 `cfg` 寄存器的 `enterxipnext` 比特设成 1 来使能 quad SPI 控制器的 XIP 模式。
5. 重使能直接访问控制器，和间接访问控制器（如果需要）。

不支持 Basic-XIP 的 Micron Quad SPI Flash 器件

要在不支持 Basic-XIP 的 Micron quad SPI flash 器件中进入 XIP 模式，需要执行下面的步骤：

1. 如果想要在退出时还原模式比特的值，则需要以模式比特保存这些值。
2. 禁止直接访问控制器和间接访问控制器以确保没有新的读或写访问发送到 flash 器件。
3. 确保通过将易失配置寄存器 (VCR) bit 3 设成 1，在 flash 器件中使能 XIP 模式。使用 `flashcmd` 寄存器发出 VCR 写命令。
4. 将 `modebit` 寄存器中的 XIP 模式比特设为 0x00。
5. 通过将 `cfg` 寄存器的 `enterxipnext` 比特设成 1 来使能 quad SPI 控制器的 XIP 模式。
6. 重新使能直接访问控制器，和间接访问控制器（如果需要）。

Winbond Quad SPI Flash 器件

执行下面的步骤在 Winbond quad SPI flash 器件中进入 XIP 模式：

1. 如果想要在退出时还原模式比特的值，则需要以模式比特保存这些值。
2. 禁止直接访问控制器和间接访问控制器以确保没有新的读或写访问发送到 flash 器件。
3. 将 `modebit` 寄存器中的 XIP 模式比特设为 0x20。
4. 通过将 `cfg` 寄存器的 `enterxipnext` 比特设成 1 来使能 quad SPI 控制器的 XIP 模式。
5. 重新使能直接访问控制器，和间接访问控制器（如果需要）。

Spansion Quad SPI Flash 器件

执行下面的步骤在 Spansion quad SPI flash 器件中进入 XIP 模式：

1. 如果想要在退出时还原模式比特的值，则需要以模式比特保存这些值。
2. 禁止直接访问控制器和间接访问控制器以确保没有新的读或写访问发送到 flash 器件。
3. 将 `modebit` 寄存器中的 XIP 模式比特设为 0xA0。
4. 通过将 `cfg` 寄存器的 `enterxipnext` 比特设成 1 来使能 quad SPI 控制器的 XIP 模式。
5. 重新使能直接访问控制器，和间接访问控制器（如果需要）。

退出 XIP Mode

执行下面的步骤退出 XIP 模式：

1. 禁止直接访问控制器和间接访问控制器以确保没有新的读或写访问发送到 flash 器件。

- 2. 根据 flash 器件和制造商，将模式比特还原到进入 XIP 模式之前的值。
- 3. 将 `cfg` 寄存器的 `enterxipnextrd` 比特设为 0。


flash 器件在禁止它的内部 XIP 模式状态之前必须接收一个读指令。因此，XIP 模式内部地保持有效 (active) 直到进行下一个读指令。要确保在任何的读序列结束前 XIP 模式是禁止的。

上电复位上的 XIP 模式

某些作为非易失配置设置的 flash 器件可以是 XIP 使能的，支持 flash 器件在无需软件的干预下在上电复位 (POR) 时进入 XIP 模式。软件不能通过状态寄存器读操作来发现 POR 上的 XIP 状态，因为一个 XIP 使能的 flash 器件只能通过 XIP 读操作进行访问。如果您知道器件将在 POR 上进入 XIP 模式，那么您的初始引导软件要配置 `modebit` 寄存器，并将 `cfg` 寄存器的 `enterxipimm` 比特设为 1。

如果事先不知道器件是否将要在 POR 上进入 XIP 模式，那么您的初始引导软件要通过 `flashcmd` 寄存器发出一个 XIP 模式退出命令，然后执行第 12 - 15 页“进入 XIP 模式”中的步骤。软件必须知道器件的模式比特要求，因为 XIP 模式的进入和退出会根据器件的而不同而不同。


Quad SPI Flash 控制器地址映射和寄存器定义

 地址映射和寄存器定义位于本卷的 [hps.html](#) 文件中。点击链接打开文件。

要查看模块描述和基地址，需要点击下面模块实例的链接：

- [qspiargs](#)
- [qspidata](#)

然后点击寄存器名来查看寄存器和域描述。寄存器地址是相对于每个模块实例基地址的偏移。

 所有模块的基地址也列在 *Cyclone V 器件手册卷 3* 中的 *Introduction to the Hard Processor System* 章节中。

文档修订历史

表 12 - 4 显示了本文档的修订历史。

表 12 - 4. 文档修订历史

日期	版本	修订内容
2012 年 11 月	1.2	次要更新。
2012 年 5 月	1.1	添加了结构图和系统集成，功能描述，编程模型和地址映射和寄存器定义章节。
2012 年 1 月	1.0	首次发布。