(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 117043775 A
(43)申请公布日 2023.11.10

(21)申请号 202280020820.X

(22)申请日 2022.04.21

(30)优先权数据
63/179,043 2021.04.23 US

(85)PCT国际申请进入国家阶段日
2023.09.12

(86)PCT国际申请的申请数据
PCT/US2022/071842 2022.04.21

(87)PCT国际申请的公布数据
WO2022/226520 EN 2022.10.27

(71)申请人 谷歌有限责任公司
地址 美国加利福尼亚州

(72)发明人 金恩赞 蒂莫西·杰伊·陈

(74)专利代理机构 中原信达知识产权代理有限责任公司 11219
专利代理师 邓聪惠 周亚荣

(51)Int.Cl.
G06F 21/85(2006.01)

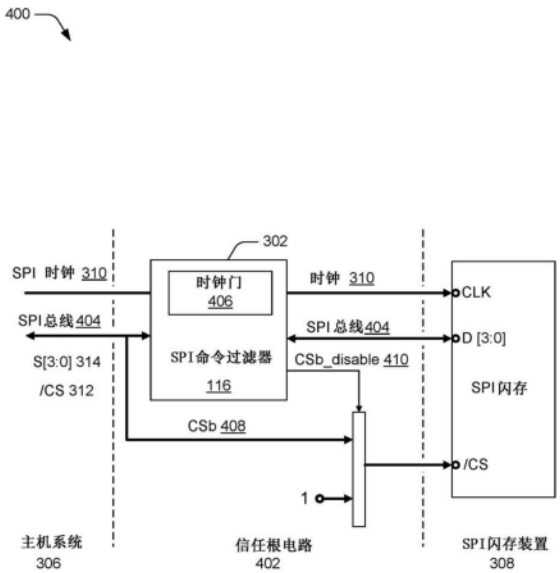
权利要求书2页 说明书41页 附图9页

(54)发明名称

安全串行外围接口通信

(57)摘要

本文档公开安全串行外围接口(SPI)通信的各方面。在一些方面中,安全SPI通信模块监测由主机传输到外围块的通信,所述外围块经由SPI互连件耦合到所述主机。所述模块将由所述主机发送的所述通信的相应命令与指示所述外围块未被授权执行的命令的信息进行比较。基于所述比较,所述模块确定相应命令中的一个是所述外围块未被授权执行的所述命令中的一个。所述模块然后防止所述外围块接收所述通信的相应命令的至少一部分。通过这样做,所述模块可以防止所述外围块执行未经授权的命令,所述未经授权的命令可能损害所述外围块的安全性。



CN 117043775 A

1.一种由安全电路系统实现的方法,所述安全电路系统与用于安全串行外围接口通信的系统的主机相关联,所述方法包括:

监测由所述主机传输到所述系统的外围块的通信,所述外围块经由串行外围接口SPI互连件耦合到所述主机;

将由所述主机发送的所述通信的相应命令与指示所述外围块未被授权执行的哪些命令的信息相比较;

基于所述比较,确定相应命令中的一个是所述外围块未被授权执行的命令;以及

防止所述外围块接收所述外围块未被授权执行的相应命令的至少一部分。

2.根据权利要求1所述的方法,其中,所述防止包括更改所述SPI互连件的芯片选择线或时钟线的状态,以防止所述外围块接收由所述主机发送的所述通信的相应命令的所述至少一部分。

3.根据权利要求1所述的方法,其中,所述防止包括更改所述SPI互连件的芯片选择线和时钟线的状态,以防止所述外围块接收由所述主机发送的所述通信的相应命令的所述至少一部分。

4.根据权利要求2或权利要求3所述的方法,其中:

所述通信的相应命令包括八个位;以及

所述更改包括在经由所述SPI互连件传送相应命令的第八位的第八时钟循环之前或在所述第八时钟循环上对所述芯片选择线取消断言,以防止相应命令的至少所述第八位由所述外围块处理。

5.根据权利要求2或权利要求3所述的方法,其中:

所述通信的相应命令包括八个位;以及

所述更改包括在经由所述SPI互连件传送相应命令的第八位的第八时钟循环之前或在所述第八时钟循环上对所述时钟线进行门控,以防止相应命令的至少所述第八位由所述外围块处理。

6.根据权利要求1至5中任一项所述的方法,其中,相应命令是第一通信的第一命令,并且所述方法进一步包括:

确定由所述主机发送到所述外围块的第二通信的第二命令包括写入状态寄存器命令;以及

更改所述写入状态寄存器命令以防止所述第二通信更改所述外围块的状态寄存器。

7.根据权利要求6所述的方法,其中,所述写入状态寄存器命令的所述更改包括:

将所述写入状态寄存器命令的位位置设置为预定义位值以提供更改后的写入状态寄存器命令;以及

将所述更改后的写入状态寄存器命令传递到所述外围块,所述更改后的写入状态寄存器命令包括所述预定义位值和所述写入状态寄存器命令的其它位。

8.根据权利要求1至7中任一项所述的方法,其中,指示所述外围块未被授权执行的所述命令的所述信息包括以下项的表:

所述外围块未被授权执行的所述命令;以及

所述外围块被授权执行的命令。

9.根据权利要求8所述的方法,进一步包括在所述通信的所述监测之前,配置所述外围

块未被授权执行的所述命令和所述外围块被授权执行的所述命令的所述表。

10.根据权利要求8或权利要求9所述的方法,其中,由所述主机发送的所述通信的相应命令与所述表的所述信息的所述比较包括:

基于相应命令索引化到所述表中,以确定相应命令是否被授权用于由所述外围块执行。

11.根据权利要求1至10中任一项所述的方法,其中:

所述系统的所述外围块包括经由所述SPI互连件耦合到所述主机的存储器模块,所述存储器模块包括非易失性存储器或闪存存储器中的一个;或

所述系统的所述外围块不包括写入保护输入节点,或所述外围块的写入保护输入节点被禁用。

12.根据权利要求11所述的方法,进一步包括在尝试经由所述SPI互连件将存储在所述系统的所述存储器模块上的二进制图像从所述存储器模块加载到所述主机之前,校验所述二进制图像。

13.根据权利要求12所述的方法,进一步包括:

响应于所述二进制图像的成功校验,允许所述存储器模块经由所述SPI接口将所述二进制图像发送到所述主机;或

响应于所述二进制图像的不成功校验,防止所述存储器模块经由所述SPI接口将所述二进制图像发送到所述主机。

14.根据权利要求13所述的方法,其中,所述二进制图像是存储在所述存储器模块的第一分区中的第一二进制图像,并且所述方法进一步包括:

操纵用于所述第一二进制图像的读取命令的地址,以使所述存储器模块从所述存储器模块的第二分区向所述主机发送第二二进制图像。

15.一种集成电路,所述集成电路包括用于安全串行外围接口通信的电路系统,所述电路系统包括:

具有功能核心的主机;

至少一个外围块;

串行外围接口SPI互连件,所述SPI互连件耦合所述主机和所述至少一个外围块;以及

安全SPI通信模块,所述安全SPI通信模块与所述SPI互连件能够操作地耦合并且被配置成执行根据权利要求1至14中任一项所述的操作。

安全串行外围接口通信

背景技术

[0001] 由于社会的计算机化程度不断提高并且越来越依赖个人计算装置存储用户敏感信息并为其用户执行各种操作，包括操作车辆、执行用户认证以及完成数字货币交易等，世界越来越容易受到对计算装置的敏感信息的各种代价高昂的攻击。

[0002] 最近基于故障的密码分析方法已经识别出涉及故障注入攻击的潜在安全威胁方法。故障注入攻击涉及攻击者将故障物理注入计算系统，而不是软件注入，由此有意地更改电子组件的行为。因此，故障注入攻击能够绕过许多低级别的系统安全功能、更改计算系统行为以实现恶意意图、和/或提取敏感信息。故障注入攻击可能涉及电压毛刺、时钟毛刺、激光注入、电磁注入等。在一些情况下，这些攻击能够在各种位置引入故障注入以破坏或削弱电子系统安全性。因此，故障注入攻击可以更改在计算系统内传递的命令或数据，并且能够潜在地更改系统的执行流程，从而导致下游问题，诸如密钥泄漏、权限提升或代码的无意执行。

发明内容

[0003] 本文档描述用于安全串行外围接口(SPI)通信的设备和技术。在一些方面中，安全SPI通信模块监测由主机或其它总线控制器传输到外围块的通信，所述外围块经由SPI互连件耦合到主机。安全SPI通信模块将由主机发送的通信的相应命令与指示外围块未被授权执行的命令的信息进行比较。基于该比较，安全SPI通信模块确定相应命令中的一个是外围块未被授权执行的命令中的一个。安全SPI通信模块然后防止外围块接收通信的相应命令的至少一部分。通过这样做，模块能够防止外围块执行未经授权的命令，这可能通过改变状态寄存器、禁用保护方案、访问敏感信息等来损害外围块的安全性。

[0004] 提供本发明内容以介绍下文在具体实施方式中进一步描述并且在附图中说明的用于实现安全SPI通信的简化概念。本发明内容并非旨在标识所要求保护的主题的基本特征，也不旨在用于确定所要求保护的主题的范围。

附图说明

[0005] 安全SPI通信的一个或多个方面的细节在整个公开中参考附图进行描述。在说明书和附图的不同情况下使用相同附图标记表示相同或相似元件：

[0006] 图1图示了示例操作环境，该示例操作环境包括可以实现安全SPI通信的各方面的设备；

[0007] 图2图示了示例系统，该示例系统包括能够实现安全SPI通信的各方面的处理器和多个电路组件；

[0008] 图3图示了根据一个或多个方面的实现SPI命令过滤的系统组件的示例配置；

[0009] 图4图示了包括用于实现安全SPI通信的各方面的时钟门控和命令过滤功能的示例系统；

[0010] 图5图示了其中可以实现安全SPI通信的各方面的SPI通信信令的示例时序图；

[0011] 图6图示了根据一个或多个方面的用于安全SPI通信的示例方法；

[0012] 图7图示了根据一个或多个方面的用于过滤SPI通信的命令的示例方法；

[0013] 图8图示了根据一个或多个方面的用于操纵命令地址以启用替代二进制图像访问的示例方法；并且

[0014] 图9图示了可以实现安全SPI通信的方面的示例片上系统。

具体实施方式

[0015] 计算系统通常包括具有安全电路系统和软件的集成电路，以提供针对缺陷、攻击和其它潜在危害事件的保护措施。在当今的计算环境中，不良行为者能够使用大量攻击向量在极大数量的级别上攻击计算装置。例如，故障注入攻击降低许多这些安全范例提供的保护。故障注入攻击能够绕过系统安全功能、更改系统行为以实现恶意意图、和/或泄露机密信息。使用故障注入攻击，攻击者能够使用毛刺(例如，系统中突然的、临时的、注入的故障)间接地或直接地更改电子组件(例如，中央处理单元)的编程操作。这种攻击有时能够“破坏”计算装置，但是在其它情况下，精确和有针对性的攻击能够带来危及安全的威胁。例如，故障注入攻击能够允许攻击者破坏程序的控制流，这可能导致调用不正确的函数，诸如在“return to libc”类型的攻击中。在一些情况下，这些攻击可能导致计算装置暴露敏感数据或执行未校验代码。因此，故障注入攻击可以更改在计算系统内传递的命令或数据，并且能够潜在地更改系统的执行流，从而导致下游问题，诸如密钥泄漏、权限提升或代码的无意执行。

[0016] 然而，旨在解决此类攻击的先前技术在防止攻击者更改命令和包括系统安全性方面通常薄弱且无效。例如，在一些先前方案中，装置可能改变操作码上的最后一位以尝试取消命令。然而，如果命令仍然由外围设备消耗，则这种类型的方案可能无效，或如果两个命令相邻(例如，断开一位)，则这种类型的方案不适用。其它先前方案维护一个命令表，该命令表将多个命令映射到单个命令操作码中。表代理将在接收命令的每一位时遍历命令表，并且向下游外围设备发送不同的操作码。此类映射表存在与将某些命令映射到表中的效率低下有关的问题，这些命令无法取决于系统设计者的偏好进行筛选或过滤。因此，先前技术容易受到各种命令的攻击，这些命令可以在没有限制或过滤的情况下通过系统传播，导致与未经授权或未经许可的命令的执行相关的系统不安全性。

[0017] 与先前的安全技术相比，本公开描述安全SPI通信的方面。在各方面中，安全SPI通信模块监测由主机或其它总线控制器传输到外围块的通信，该外围块经由SPI互连件耦合到主机。安全SPI通信模块将由主机发送的通信的相应命令与指示外围块未被授权或未经许可执行的哪些命令(例如，排除命令列表)的信息进行比较。基于比较，安全SPI通信模块确定相应命令中的一个是外围块未被授权执行的命令中的一个。响应于该确定，安全SPI通信模块然后更改SPI互连件的芯片选择线和/或时钟线的状态，以防止外围块接收通信的相应命令的至少一部分。通过这样做，模块能够防止外围块执行未经授权的命令，这可以通过改变状态寄存器、禁用保护方案、访问敏感信息等来损害外围块的安全性。

[0018] 因此，与先前技术相比，本文中所描述的安全SPI通信的各方面能够通过利用SPI外围组件(诸如闪存存储器装置或芯片)的本地操作、预期行为或保护机制来保护SPI外围组件。通过检查，闪存存储器装置通常可以通过使用写入保护引脚或输入(例如，Write_

Protect引脚)来保护闪存存储器装置的状态寄存器。然而,这些闪存存储器装置中的大多数在启用四读命令或与四读命令一起使用时,都会禁用该专用写入保护引脚。因此,如本文中所描述的安全SPI通信的各方面通过实现选择性命令过滤和/或选择性数据过滤来复制写入保护,提供闪存存储器装置保护。

[0019] 通常,SPI闪存装置或其它外围设备的状态寄存器包括特定的保护位,这些保护位可配置为保护或锁定存储器的各个区段,诸如用于块保护、扇区保护、顶部/底部范围保护等。换句话说,根据闪存存储器装置的各种保护方案,设置状态寄存器的这些保护位中的一个或多个转换为保护或防止对存储器的预定义区域的操纵。因此,闪存存储器装置的保护方案或保护定义仅能够通过使用启用对状态寄存器的访问的特定命令来改变。代替过滤所有命令,根据先前的技术,安全SPI通信的各方面可以保护对访问状态寄存器和改变闪存存储器装置的保护状态(例如,写入状态寄存器以及类似者)有用的选择性命令集。因此,如果安全SPI通信的所描述方面防止主机或另一总线主控器发出这些受保护命令来改变保护状态,则命令发布者将不能改变保护状态。通过这样做,主机或其它总线主控器无法编程或擦除闪存存储器装置的内容。在各方面中,读取命令仍将被正常转发,并且如果需要,可以划分出特定区域以维护内部邮箱。

[0020] 关于写入状态寄存器,安全SPI通信的各方面可以将写入状态寄存器命令的数据过滤到下游SPI闪存装置,而不是将写入状态寄存器命令过滤成没有副作用的值(例如,诸如读取命令)。因此,安全SPI通信模块可以允许所有命令字节从主机传递到SPI闪存装置而无需进行操纵。如果安全SPI通信模块将命令字节确定或识别为写入状态寄存器命令(SPI闪存装置或供应商能够提供有用于识别写入状态访问的可编程列表),则该命令的编程值能够由安全SPI通信模块控制。

[0021] 在一些方面中,SPI装置程序值可以通过选择寄存器和值寄存器维持。通常,选择寄存器控制允许特定数据位从主机通过还是强制特定数据位从安全SPI通信模块或SPI装置控制器通过。假设设置选择位,则值寄存器可以控制强制值是什么。例如,假设将选择寄存器和值寄存器被编程为设置成“0011_0000(0x30)”的选择寄存器和设置成“xx01_xxxx”的值寄存器。这意味着当检测到写入状态命令时,位4和5(第四和第五位位置)将被强制为“1”和“0”,而将允许命令的其余位通过。因为写入状态命令可能因芯片供应商而变化,所以表示写入状态的内容能够是可编程的或可配置的,尽管可能仅需要两到四个条目。

[0022] 除了写入状态命令之外,安全SPI通信可以更改或保护由主机或总线主控器发出的可能影响外围装置的保护位状态的其它命令,这些命令可以包括复位命令或其它供应商特定命令。为此,安全SPI通信的各方面可以实现选择性过滤,该择性过滤在系统的目的地外围设备能够接收、捕获或处理命令之前使命令静音或改变命令。如贯穿本公开所描述的,在完全捕获命令的最后一位之前,安全SPI通信模块或SPI命令过滤器可以对外围装置(诸如SPI闪存装置)的芯片选择线或信号(CSb)取消断言(deassert)。在各方面中,安全SPI通信模块可在第八时钟循环(例如,在捕获之前)检测到应过滤命令。然后,安全SPI通信模块提前对芯片选择线取消断言,以确保外围装置没有捕获最后一位(例如,第八位)命令位。替代地或另外,安全SPI通信模块能够在第七节拍或时钟循环之后门控或保持到外围设备的SPI时钟线,以防止外围装置捕获要过滤的命令的第八位。在一些情况下,这些操作是定时敏感的,并且安全SPI通信模块可以在SPI时钟周期的一半内起作用,诸如生成控制信号(例

如,其门控和过滤)。例如,对于25MHz到100MHz的SPI时钟,安全SPI通信模块可以在10到50纳秒内生成控制信号、切换芯片选择线,或门控SPI时钟以完成所描述的命令过滤。

[0023] 在安全SPI通信的各方面中,命令过滤器可以是可配置的或可编程的以通过一种或多种方式实现选择性命令过滤,从而提供灵活性。在一些情况下,256个可能命令的命令空间中的每个命令将具有1位启用,诸如通过用于命令空间的八个寄存器。安全SPI通信模块可以使用接收到的命令或接收到的命令的一部分来索引化到表格中,以确定允许命令通过还是禁止(例如,过滤)命令。例如,启用位控制指示或控制允许命令向下游传递到外围装置还是过滤命令。在一些实施方式中,启用位值一"1"指示允许命令通过,并且启用位值零"0"指示由安全SPI通信模块或命令过滤器过滤命令。在过滤操作之前,主机的软件或固件可以配置或编程过滤表。

[0024] 在安全SPI通信模块实现命令过滤的情况下,可能不需要对SPI闪存装置进行状态保护,因为安全SPI通信模件能够简单地过滤或阻止将启用操纵SPI闪存装置的状态寄存器的命令。因此,具有选择性或降低复杂性的命令过滤和/或状态寄存器命令数据过滤的益处是过滤了更少的命令或指令。换句话说,减少必须过滤的指令集(例如与写入状态寄存器相关的命令),并且在其余时间内,安全SPI通信模块允许系统的主机直接与SPI闪存装置通信。这导致固件或安全SPI通信模块较少介入,因为固件不需要捕获所有命令有效载荷并且确定这些有效载荷是否应向下游传递到SPI闪存装置。替代地或另外,主机或系统软件可以配置安全SPI通信模块,以实现用于命令或数据过滤的所描述的方面的任何组合。例如,安全SPI通信模块可以被配置成仅实现命令过滤、默认允许和/或不允许命令过滤、不允许具有状态保护的命令过滤、允许具有状态保护的命令过滤等等。本文中所描述的这些和其它方面可以确保在主机或系统软件设置或配置安全SPI通信模块以实现命令过滤和/或A/B分区访问之后,无法任意地改变SPI闪存装置(例如,SPI闪存芯片)的保护设置。因此,只要下游SPI闪存装置根据其内部保护机制起作用,这些方面就能够以比先前技术更低的复杂性和更高的效率实现安全SPI通信。

[0025] 在各个方面中,安全SPI通信模块耦合到系统的SPI互连件,该系统包括主机和SPI闪存装置,以及任何数目的其它系统外围块或组件。安全SPI通信模块可以实现命令控制方案,该命令控制方案包括窥探或监测从主机平台到SPI闪存装置的SPI事务,并且用于在检测到未授权业务时进行干预。命令控制方案旨在阻止任何有害的读取/写入请求或自行完成的命令(例如,芯片擦除)。这些自行完成的命令可以包括后面没有其它数据(诸如,地址或有效载荷)的命令。在各方面中,方案可以使用如本文所描述的SPI命令过滤器和SPI时钟控制器(例如,时钟门控单元)。通常,命令过滤器可以基于命令块列表或表而阻止传入命令,或过滤器可以仅允许授权命令基于经许可的命令列表通过。命令块列表或经许可的命令列表可以被实现为软件或在硬件电路系统中实现,这取决于使用哪一个。在各方面中,为了阻止命令,命令过滤器在命令传播期间的第八SPI时钟沿之前或在第八SPI时钟沿处更改芯片选择线或信号线。

[0026] 同时,命令过滤器可以门控或保持到SPI闪存装置的SPI时钟线,直到主机系统释放其芯片选择信号。注意,SPI芯片选择线和SPI时钟可以同时、非常接近(例如,在相同的时钟节拍或循环内)或以部分重叠的方式切换。在各方面中,安全SPI通信模块包括时钟门控单元以控制SPI时钟的传播。因此,时钟门控单元可以不将源自主机的SPI时钟信号传播到

SPI闪存装置，以有效地防止芯片选择线或信号上的毛刺。在一些情况下，当不对SPI时钟信号进行门控时，芯片选择线则可能需要一个以上时钟脉冲来去除毛刺。换句话说，在没有时钟信号门控的情况下，命令过滤器可能错过提高芯片选择信号的定时，以便取消未授权的命令。因此，利用SPI时钟门控，安全SPI通信模块能够将在未经授权的命令完全传输到SPI闪存装置之前对芯片选择线或信号取消断言而不加毛刺，并且SPI装置可以在不执行不完整命令的情况下取消该该不完整命令。

[0027] 本文中所描述的安全SPI通信的这些和其它方面可以确保存储器不能在安全模块或命令过滤器的下游受到攻击，并且对存储器的任何读取或写入都被绑定或限制到攻击者无法更改的特定地址。以下讨论描述操作环境、示例系统和组件、安全SPI通信的示例实施方式、示例方法以及其中可以体现操作环境的组件的片上系统(SoC)。在本公开的场境中，仅以示例的方式参考操作环境。

[0028] 示例环境

[0029] 图1图示了示例环境100，该示例环境包括能够实现安全SPI通信和相关联的通信完整性方案的方面的设备102。设备102可以被实现为任何合适的装置，一些装置被图示为智能手机102-1、平板计算机102-2、膝上型计算机102-3、游戏控制台102-4、台式计算机102-5、服务器计算机102-6、可穿戴计算装置102-7(例如，智能手表)和宽带路由器102-8(例如，移动热点)。尽管未示出，但是设备102也可以被实现为移动台(例如，固定或移动STA)、移动通信装置、客户端装置、用户设备、移动手机、娱乐装置、移动游戏控制台、个人媒体装置、媒体播放装置、健康监测装置、无人机、相机、能够进行无线互联网接入和浏览的互联网家用电器、IoT装置和/或其它类型的电子装置。设备102可以提供其它功能或包括为了清楚或视觉简洁而从图1省略的组件或接口。

[0030] 设备102包括集成电路104，该集成电路利用一个或多个处理器106和计算机可读介质(CRM 108)，该计算机可读介质可以包括存储器介质或存储介质。处理器106可以被实现为(例如，多核中央处理单元(CPU)或应用处理器(AP)的)通用处理器、专用集成电路(ASIC))、图像处理单元(GPU)或片上系统(SoC)，其中设备102的其它组件集成在其中。在安全SPI通信的各方面中，处理器106中的一个或多个还可以包括如在本公开中描述的完整性功能。

[0031] CRM 108能够包括任何合适类型的存储器介质或存储介质，诸如只读存储器(ROM)、可编程ROM(PROM)、随机存取存储器(RAM)、动态RAM(DRAM)、静态RAM(SRAM)或闪存存储器(例如，SPI闪存装置或芯片)。在该讨论的场境中，设备102的计算机可读介质108被实现为不包括瞬态信号或载波的至少一个基于硬件或物理存储装置。设备102的应用、固件和/或操作系统(未示出)能够作为处理器可执行指令体现在计算机可读介质108上，这些指令可以由处理器106执行以提供本文中所描述的各种功能。计算机可读介质108还可以存储装置数据110，诸如可通过设备102的应用、固件或操作系统访问的用户数据或用户介质。

[0032] 在该示例中，集成电路104包含安全电路系统112。设备102、集成电路104，或安全电路系统112可以实现安全密码处理器。安全电路系统112可以使用例如电路组件114-1到电路组件114-n的一个或多个电路组件114实现。电路组件114可以被组织成执行任何数目的操作以实现设备102的功能。电路组件的示例包括如图2中所描述的处理器和多个功能组件、外围设备和/或IP块。安全电路系统112能够被实现为例如受保护的飞地、可信芯片平

台、基于硬件的信任根(RoT)芯片(例如,硅RoT)等等。无论安全电路系统112如何或在何处结合到电子装置中,安全电路系统112都可以对抗或制止许多不同类型的攻击,诸如关于安全SPI通信描述的通过SPI总线或互连件发出的攻击或恶意命令。

[0033] 在各方面中,安全电路系统112包括电路组件114-1到114-n,该电路组件提供或实现安全电路系统112、RoT电路系统、集成电路104和/或设备102的相应功能。为了实现安全SPI通信的各方面,电路系统组件114包括SPI命令过滤器116和SPI块控制器118,它们可以被实现为安全SPI通信模块的一部分。在各方面中,SPI命令过滤器116监测由设备102的主机和设备102的闪存存储器模块(例如,CRM 108)交换的通信。通常,SPI命令过滤器116将由主机发送的通信的相应命令与指示闪存存储器模块未被授权执行的哪些命令的信息进行比较。基于该比较,SPI命令过滤器116确定相应命令中的一个是闪存存储器模块未被授权执行的命令中的一个(例如,写入状态寄存器命令)。响应于该确定,SPI命令过滤器116能够对闪存存储器模块的芯片选择线取消断言,或使用SPI块控制器118来门控或挂起闪存存储器模块的SPI时钟线。通过这样做,SPI命令过滤器116能够防止闪存存储器模块接收或处理未经授权的命令的至少一部分,这可以防止未经授权的命令包括闪存存储器模块的安全性。替代地或另外,SPI命令过滤器116或安全SPI通信模块可以校验存储在闪存存储器模块的第一分区上的二进制图像的完整性。如果完整性校验失败,则SPI命令过滤器116可以更改SPI互连件事务的地址,以使另一二进制图像从闪存存储器模块的第二分区加载,以防止主机从第一分区加载未校验且可能受损的二进制图像。这些只是对实现安全SPI通信有用的实体的少数示例,这些示例的实施方式和使用变化并且在整个公开中描述。

[0034] 如图所示,安全电路系统112耦合到互连件120,该互连件可以将安全电路系统的组件、外围设备和/或目的地与主机或主机接口耦合。互连件120能够使用例如总线、交换结构、链路、通信信道或使各种电路组件能够通信的总线网络来实现。在一些方面中,互连件包括根据串行外围接口通信标准实现的SPI互连件或总线。在一些实施方式中,SPI互连件包括芯片选择线、时钟线和四个I/O线(例如,S[3:0]或D[3:0])。电路元件中的每一个可以直接地或间接地耦合到互连件120。互连件120可以实现与设备102的数据端口或接口通信,以使得电路组件能够与其它装置或数据网络进行通信。

[0035] 设备102还可以包括显示器122、收发器124、输入/输出端口(I/O端口126)和/或传感器128。显示器122可以与处理器106中的一个(例如,图形处理单元(GPU))可操作地耦合,并且被配置成以图形方式呈现设备102的操作系统或应用的相应接口。收发器124可以被配置成实现根据任何合适的通信协议通过有线或无线网络进行数据(例如,装置数据110)的有线或无线通信。设备102的I/O端口126可以包括通用串行总线(USB)端口、同轴电缆端口和其它串行或并行连接器(包括内部连接器),这些连接器用于将电子装置耦合到各种组件、外围设备或附件,诸如键盘、麦克风或相机。

[0036] 设备102还包括传感器128,该传感器使设备102能够感测设备102操作的环境的各种特性、变化、刺激或特征。例如,传感器128可以包括各种运动传感器、环境光传感器、声学传感器、电容传感器、红外传感器、温度传感器、雷达传感器,或磁传感器。替代地或另外,传感器128可以诸如通过触摸感测、手势感测或接近感测实现与设备102的用户交互,或从设备102的用户接收输入。

[0037] 示例电路组件

[0038] 图2在200处图示了示例安全电路系统112，该安全电路系统包括能够被实现为支持安全SPI通信的各方面的处理器和多个电路组件。如图所示，安全电路系统112包括处理器106，该处理器耦合到互连件120。处理器106、多个存储器和多个其它电路组件114中的每一个可以直接地或间接地耦合到互连件120。在各方面中，图2的组件可以被体现为实现信任根和/或其它安全密码特征的安全计算平台或安全片上系统。替代地或另外，图2的组件可以被实现为通过互连件120耦合的系统的一个或多个IC、外围块，或IP块，该互连件可以被实现为可操作地耦合系统的组件或IP块的结构。

[0039] 在各方面中，互连件120可以包括SPI命令过滤器116和SPI时钟控制器118或与SPI命令过滤器116和SPI时钟控制器118耦合，它们可以被实现为安全SPI通信模块的一部分。在各方面中，SPI命令过滤器116监测由系统的主机和系统的SPI存储器装置交换的通信。通常，SPI命令过滤器116将由主机发送的通信的相应命令与指示闪存存储器模块未被授权执行的哪些命令的信息进行比较。基于该比较，SPI命令过滤器116确定相应命令中的一个是闪存存储器模块未被授权执行的命令中的一个(例如，写入状态寄存器命令)。响应于所述确定，SPI命令过滤器116能够对闪存存储器模块的芯片选择线取消断言，或使用SPI块控制器118来门控或挂起闪存存储器模块的SPI时钟线。这些只是SPI命令过滤器116和SPI时钟控制器118的少数示例，这些示例的实施方式和使用变化并且参考图3到图5在整个本公开中描述。

[0040] 处理器106可以通过互连件120与电路组件114耦合和/或直接与其它组件或接口耦合。如图2中所示，系统可以包括耦合到互连件120的多个电路组件114，从而实现与处理器106的交互，该处理器可以充当系统的主机。在该示例中，电路组件114包括寄存器堆202和各种存储器204至210。电路组件114可以包括任何合适配置的一个或多个存储器(例如，CRM 108)，并且包括ROM 204、SRAM 206和SPI闪存存储器208。在该示例中，SPI命令过滤器116和/或SPI时钟控制器118可以在互连件120与SPI闪存存储器208之间实现。

[0041] 在各方面中，SPI闪存存储器208包括控制状态寄存器(CSR)和闪存介质，该闪存介质被配置成存储系统或主机的信息。SPI闪存装置或其它存储器外围设备的状态寄存器可以包括特定的保护位，这些保护位可配置为保护或锁定存储器的各个区段，诸如用于块保护、扇区保护、顶部/底部范围保护等。在各方面中，SPI命令过滤器116和SPI时钟控制器118可以防止未经授权的命令由SPI闪存存储器208消耗并且更改SPI闪存存储器208的CRS的设置。在一些情况下，SPI闪存存储器208不包括写入保护输入节点，或诸如当启用四模式访问时，禁用外围块的写入保护输入节点。尽管未示出，但是电路组件114可以包括其它存储器(例如，一次性可编程存储器或DRAM存储器)和/或经由其它组件耦合的存储器，诸如另外的串行外围接口(SPI)或USB耦合存储器。

[0042] 如图2中所示，电路组件114还可以包括警报处置器210、高级加密标准(AES)单元(AES单元212)、基于散列的消息认证码(HMAC)引擎(HMAC引擎214)和串行外围接口(SPI)装置(SPI装置216)。这里，注意，SPI命令过滤器116和/或SPI时钟控制器118可以在互连件120与SPI装置216之间实现，这可以防止SPI装置216消耗如本文中所描述的未经授权的命令。电路组件114还能够包括通用异步接收器/发射器(UART)单元(UART单元218)、通用输入/输出(GPIO)接口(GPIO接口220)、引脚复用器(引脚复用器222)和焊盘控制器224。多个电路组件114还能够包括随机数生成器(RNG 226)和定时器228(例如，看门狗定时器)，其它组件可

以从该随机数生成器获得高熵值以用作认证令牌。尽管图2中描绘或本文中描述存储器和其它组件114的某些示例，但是安全电路系统112的给定实施方式可以包括处理器、控制器、存储器、模块或外围装置的更多、更少和/或不同示例，包括其复制。

[0043] 所图示的电路组件能够基于一个或多个时钟信号同步操作。尽管在图2中未示出，但是安全电路系统112可以包括至少一个时钟发生器以生成时钟信号，或可以包括复位电路系统以将一个或多个单独的组件彼此独立地、多个组件联合地或整个IC芯片复位。替代地，安全电路系统112可以从安全电路系统112外部的源接收至少一个时钟信号或复位信号，该源可以在分开的芯片上，也可以不在分开的芯片上。一个或多个分开的组件114可以在相应的单独时钟域中操作。例如，电路组件可以与相应组件本地的时钟同步。不同时钟域中的组件可以彼此异步地操作或通信。

[0044] 下文描述所图示的组件的示例实施方式。处理器106可以被实现为用于安全电路系统112的“主”、“中央”或“核心”处理器，主机或总线控制器的功能通过该安全电路系统实现。仅作为示例，处理器106可以用具有多级流水线的32位有序精简指令集计算(RISC)核心来实现。利用例如RISC-V功能，处理器可以实现M(机器)和U(用户)模式。激活复位引脚(未示出)(例如，通过激活低复位引脚的取消断言)使处理器106退出复位并且开始在其复位向量处执行代码。复位向量可以在ROM 204中开始，ROM在跳到系统的嵌入式闪存或任何闪存存储器之前验证系统的嵌入式闪存或任何闪存存储器中的代码。换句话说，在释放复位之前，代码预期已经实例化到嵌入式闪存中。在一些情况下，根据适合性规范，可以将整个安全电路系统112的复位为异步低激活，以支持各种电路组件之间的互操作性。复位可以由警报处置器210作为安全对策生成；由看门狗定时器生成等等。还可以将复位信号发送到其它电路组件，诸如存储器中的一个或其它组件114中的一个。

[0045] 调试模块230(DM)和中断控制器232(ItC)耦合到处理器106，它们中的任一个也可以是合适的。调试模块230提供对处理器106的调试访问。通过与IC的某些引脚接口，调试模块230中的逻辑允许处理器106进入调试模式，并且提供将代码注入装置(例如，通过模拟指令)或存储器的能力。中断控制器232可以设置在处理器106附近。中断控制器232能够接受来自安全电路系统112内的中断源的向量。中断控制器232还能够在将中断转发到处理器106以进行处理之前向中断分配均衡和优先级。

[0046] 处理器106能够提供任何所期望的性能水平或包括任何内部电路组件。例如，处理器106能够包括至少一个算术逻辑单元(ALU)(例如，包括用于计算分支目标以去除所采用的条件分支上的时延循环的另外的ALU)、寄存器堆、控制单元和输入/输出(I/O)单元，以及多个流水线级。对于多流水线级，流水线能够执行寄存器写回，以减少负载和存储的时延循环，并且防止流水线在请求之后的循环内对负载或存储的响应可用时停顿。处理器106能够实现单循环乘法器，或在对存储错误响应时产生不精确的异常，这允许处理器在不等待响应的情况下继续执行通过存储。尽管未描绘，但是处理器106，具体地来说安全电路系统112通常能够包括指令高速缓存，以提供指令的单循环访问时间。

[0047] ALU可以被配置成对接收到的数据执行算术和逻辑运算。参考图3A和3B进一步描述的寄存器堆(例如，寄存器堆202)可以是处理器寄存器(例如，控制寄存器)的阵列，用作配置用于在编程或功能处理期间的快速数据访问的高速半瞬态存储器。寄存器堆可以紧密地耦合到处理器106的ALU。为了进一步便于访问数据，寄存器堆可以包括多个读取端口或

多个写入端口，以使ALU和/或执行单元能够在单个循环中同时检索多个操作数。寄存器堆可以由触发器形成以加速数据的读取和写入位。控制单元可以被配置成控制整个系统中的数据流。I/O单元可以包括与装置或安全电路系统112的其它组件可操作地接口的端口。参考图3到图9在整个本公开中描述处理器106、电路组件114、SPI命令过滤器116、SPI时钟控制器118，或安全SPI通信模块的其它方面。

[0048] 图3在300处图示了根据一个或多个方面的实现SPI命令过滤的系统组件的示例配置。图3的示例插入器和/或其它组件可以与在整个本公开中描述的任何其它组件、架构、实体或系统相关联地实现。通常，安全SPI通信模块302(例如，RoT电路的SPI过滤器)在插入器304上实现，或可操作地与主机系统306与诸如SPI闪存装置308的下游外围设备之间的基于SPI的互连件相关联。在该示例中，安全SPI通信模块302包括SPI命令过滤器116(或SPI命令检测器)和SPI时钟控制器118的示例，它们可以被实现为时钟门控单元(未示出)或包括时钟门控单元。

[0049] 在各方面中，主机系统306与SPI闪存装置308之间的SPI接口包括SPI时钟线310(SCK310)、SPI芯片选择线312(/CS 312)和串行数据线314(S[3:0]314)，但是可以根据安全SPI通信的方面实现其它互连件或结构配置。在一些情况下，插入器304可以被实现为硅RoT电路或其它安全电路系统的一部分，该其它安全电路系统被配置成监测在主机系统306与SPI闪存装置308之间交换的通信的命令。如图3中所示，安全SPI通信模块302或插入器304可以包括SPI命令过滤器116和SPI时钟控制器118。在各方面中，SPI命令过滤器有权访问串行数据线314，以监测遍历SPI接口到SPI闪存装置308的命令和操作码。SPI命令过滤器116可以包括向逻辑门318或耦合到芯片选择线312的逻辑电路系统提供命令过滤信号316(过滤316)的输出。在一些情况下，逻辑门318包括异或门、或门、与门、多路复用器、逻辑门的组合，或任何其它合适的逻辑。通常，通过更改命令过滤信号线的状态，SPI命令过滤器能够更改芯片选择线312的状态(例如，取消断言)，以防止SPI闪存装置308接收、消耗或处理通过SPI接口的数据线312传送的命令或操作码。替代地或另外，SPI命令过滤器116可以包括向SPI时钟控制器118提供时钟启用信号320的输出。当检测到未经授权的命令时，SPI命令过滤器116能够更改时钟启用线320的状态，以使SPI时钟控制器118对传播到SPI闪存装置308的SPI时钟310进行门控。

[0050] 在图3的场境中，SPI命令过滤器116可以监测或窥探在主机系统306与SPI闪存装置308之间交换的通信。SPI通信过滤器116能够将由主机系统发送的通信的相应命令或操作码与指示授权或未授权SPI闪存装置执行的哪些命令的信息(例如，命令的包含或排除表)进行比较。如果授权通信的命令，则SPI命令过滤器116允许命令或操作码向下游传递到SPI闪存装置306。替代地，SPI命令过滤器116可以确定未授权命令或操作码以供SPI闪存装置308执行，诸如在操作码或命令的第八位。响应于该确定，SPI命令过滤器116使用逻辑门318对芯片选择线312取消断言，并且使用SPI块控制器118对到SPI闪存装置的SPI时钟信号310进行门控。通常，如果在命令或操作码的第八位或节拍之前对到下游SPI闪存装置的时钟选择线312取消断言，则SPI闪存装置308将丢弃命令或其一部分。在一些情况下，如果SPI命令过滤器116仅切换芯片选择线312，则诸如由于来自操作码的第八位的组合逻辑，芯片选择线可能容易出现毛刺。通常，芯片选择线312的控制应该是无毛刺的注册输出，因为当SPI闪存装置308已经接收到第八操作码时可能为时已晚。为了解决该毛刺问题，SPI命令过

滤器116可以对SPI时钟310进行门控以控制或切换芯片选择线312而不引入毛刺，这可以确保防止SPI闪存装置308接收或消耗操作码或命令的第八位。

[0051] 图4在400处图示了包括用于实现安全SPI通信的各方面的时钟门控和命令过滤功能的示例系统。图3的示例信任根电路和/或其它组件可以与在整个本公开中描述的任何其它组件、架构、实体或系统相关联地实现。示例系统400包括信任根电路402，该信任根电路耦合在主机系统306与SPI闪存装置308之间并且与主机系统306和SPI闪存装置308耦合。通常，SPI命令过滤器116和/或安全SPI通信模块302可以实现如本文中所描述的安全SPI接口的方面。SPI命令过滤器116可以被实现为安全串行外围接口(SPI)装置直通模块(例如，安全SPI通信模块302)。借助于示例，该安全SPI装置直通模块可以包括用于窥探从主机系统到SPI闪存装置的SPI事务以及用于干预未经授权业务的特征。可以实现该模块以用于阻止任何有害的读取/写入请求和/或用于保证将真实的二进制图像返回到主机系统。

[0052] 在该示例系统中，信任根电路402可操作地与SPI总线404耦合，该SPI总线包括SPI时钟310、SPI芯片选择线312和串行数据线314，它们可以类似于或不同于参考图3所描述的这些实现。时钟门单元406(时钟门406)从系统主机或时钟电路接收SPI时钟310，并且将SPI时钟310提供到SPI闪存装置308。在各方面中，SPI命令过滤器116监测SPI总线404，该SPI总线包括用于SPI闪存装置308的芯片选择位408。如图4中所示，SPI命令过滤器116可以选择性地向门控逻辑生成芯片选择位禁用信号410(CSb_disable 410)，该门控逻辑对芯片选择位408向SPI闪存装置308的应用或传播进行门控。

[0053] 在安全SPI通信的方面中，可以基于SPI时钟310将SPI命令或操作码的至少一些位(例如，八个位中的七个)计时到SPI闪存装置308。在一些情况下，SPI命令过滤器116可能无法确定授权还是未授权命令，直到SPI命令过滤器116接收到命令的全部八个位。SPI命令过滤器116可以在对应于命令的第八位的SPI时钟310时钟脉冲的早期部分中确定该第八位，然后确定是否授权该命令。例如，SPI命令过滤器可以将完整的命令或操作码与指示授权以供SPI闪存装置308执行的哪些命令的信息表进行比较。响应于确定未授权命令，SPI命令过滤器116对时钟门406的时钟启用信号取消断言，以暂停或延迟到SPI闪存装置308的SPI时钟信号310，以防止命令的第八位由SPI闪存装置308注册或消耗。替代地或另外，响应于确定未授权命令，SPI命令过滤器116能够断言门控逻辑的芯片选择位禁用信号410，由此使门控逻辑对到SPI闪存装置308的SPI芯片选择信号312取消断言，以防止装置消耗未授权的命令的一个或多个位。通过这样做，SPI命令过滤器116能够防止未授权的命令到达SPI闪存装置或在SPI闪存装置内注册，从而防止执行未授权的SPI操作码。

[0054] 通常，SPI命令过滤器116或安全SPI通信模块302能够在下游命令不被许可的情况下阻止任何下游命令，和/或将地址字段交换为读取命令以支持A/B分区或二进制图像方案。如本文中所描述，SPI命令过滤器116可以被配置成在传入命令在已经配置了系统的软件或硬件的块列表(例如，用于CSR命令)中的情况下阻止传入命令。当SPI命令过滤器116阻止命令时，过滤器可以在SPI时钟310的第八节拍或循环的边缘处更改或切换芯片选择线或芯片选择位，并且门控或保持SPI时钟310为低，直到主机系统释放其芯片选择线(例如，在SPI命令过滤器116的上游)。在各方面中，SPI命令过滤器116使用时钟门单元406不将SPI时钟310传播到所附接的SPI闪存装置308，这可以减少或防止芯片选择线上的毛刺并确保SPI闪存装置的有序操作。当SPI命令过滤器116不对SPI时钟线310进行门控时，芯片选择线312

可能需要一个或多个时钟循环来解决任何毛刺。在一些情况下，此定时延迟可能导致SPI命令过滤器116错过升高或更改芯片选择线的正确定时，以便取消自完成命令，诸如芯片擦除命令。因此，SPI命令过滤器116可以对SPI芯片选择线取消断言并且对SPI时钟线进行门控，以防止SPI芯片选择线的毛刺并且确保芯片选择线切换的定时有效，从而防止自完成命令由SPI闪存装置消耗。此外，为了使SPI芯片选择线的取消断言能够工作，可以在SPI命令过滤器控制芯片选择线的场境下做出关于SPI通信的一些假设。这些假设可以包括在传输操作的中间对SPI芯片选择线取消断言不会降低闪存装置的I/O质量，并且如果在第八SPI时钟循环或节拍之前对芯片选择线取消断言，则SPI闪存装置308取消命令或过程，而无需对命令的预期行为做出任何假设。例如，SPI闪存装置308可以不在第七SPI时钟循环或节拍中对充电泵进行充电，以准备用于擦除的SPI闪存装置芯片。尽管在主机到SPI装置访问的场境中进行描述，但是本文中所描述的各方面也可以应用于SPI装置到主机访问，以防止由SPI闪存发布的未经授权的有效载荷或数据到达主机装置。

[0055] 安全SPI通信模块302还可以实现地址操纵，其中软件能够配置该模块以在SPI命令之后将SPI数据线0交换到下游装置以用于前32个数据节拍。例如，除了双IO和四IO命令之外，这还被支持用于作为读取命令的命令。通常，SPI装置能够提供两个可编程CSR以支持该地址操纵特征。在各方面中，这些寄存器包括掩码寄存器和数据寄存器，该数据寄存器保持或存储用于发送到下游装置的数据。掩码寄存器可以由主机或安全SPI通信模块302设置，以指示要交换的一个或多个地址位。如果对应于地址位位置的掩码位是1，则传递到下游装置的值能够是来自数据寄存器的值，而不是来自主机系统的值。在各方面中，一个或两个寄存器被实现为32位寄存器。如果没有启用SPI闪存装置的4B地址模式，则来自寄存器的较低24位可以用于地址操纵。这些地址操作特征的使用可能变化，其中一个目的是提供对A/B二进制图像的主机访问。在各方面中，系统的信任根电路402或其它RoT电路可以校验由SPI闪存装置308存储的二进制图像。如果该二进制图像已经由恶意攻击或可靠性问题操纵，则RoT电路系统中的逻辑能够设置寄存器以将主机系统的请求重定向到SPI闪存装置的其它分区(例如，闪存的另一半或分割分区)。

[0056] 图5在500处图示了其中可以实现安全SPI通信的各方面的SPI通信信令的示例时序图。时序图500描绘根据安全SPI通信的一个或多个方面的信号的相应定时和转换。如本文中所描述的，可以由SPI命令过滤器116和/或安全SPI通信模块接收各种SPI互连件信号，SPI命令过滤器116和/或安全SPI通信模块然后可以将这些信号传递、门控或传播到下游SPI外围设备，诸如SPI闪存装置。在该示例中，SPI芯片选择信号502(CSb_In502)可以在SPI芯片选择线上从主机系统306行进到安全SPI通信模块302，SPI时钟信号504(SCK_In504)可以在SPI时钟线上从主机系统306行进，并且SPI I/O线506(IO[0]_I 506)可以在SPI互连件的串行数据线上从主机系统306行进到安全SPI通信模块302。在各方面中，过滤器信号508(过滤器508)可以表示指示SPI命令过滤器116何时检测到或确定未授权的命令的信号或中断。响应于检测到或确定未授权的命令，SPI命令过滤器116可以根据本文中所描述的各方面生成、更改或切换一个或多个其它信号。在该示例中，SPI命令过滤器产生命令过滤信号510(过滤510)，该命令过滤信号可以传播到SPI芯片选择线逻辑和/或与SPI时钟耦合的时钟门控单元。这里，SPI时钟输出信号512(SCK_out 512)可以表示时钟门控单元到SPI闪存装置的时钟输入的输出，并且芯片选择输出信号514(CSb_out 514)可以表示芯片选择线逻

辑到SPI闪存装置的芯片选择输入的输出。

[0057] 例如,当将命令作为IO[0]_i 506信号从主机系统306传输到SPI装置308时,CSb_in信号502由主机系统306的SPI接口断言并且在传输的持续时间内保持断言。CSb_in信号502由过滤信号510锁存断言并且传递到SPI闪存装置308上。以这种方式锁存芯片选择信号(例如,CSb_in信号502和CSb_out信号514)可以确保干净的芯片选择断言和随后的操纵(如果需要的话)而没有毛刺。在没有该特征的情况下并且如果在芯片选择线上引入毛刺,则SPI闪存装置可能展现未知行为,这可以包括复位损坏或破坏SPI闪存装置308的内容。在各方面中,基于SCK_out 512信号是SCK_In 504信号,将命令或操作码的位(例如,八个位)中的每一个计时到SPI装置308。在该示例,命令过滤器116可以不确定授权还是未授权命令,直到命令过滤器116接收到命令的全部8个位。当命令过滤器116在516处在对应于命令的第八位的SCK_in 502时钟脉冲的早期部分中确定该第八位并且确定命令是未经授权的命令(例如,将命令与命令块列表进行比较)时,命令过滤器116实现命令过滤。在该示例中,SPI命令过滤器116在518处断言过滤命令信号510,该过滤命令信号进而断言或取消断言(取决于SPI闪存引脚的极性配置)CSb_out 514信号。SPI命令过滤器116还对时钟门控单元的时钟启用取消断言,以暂停或延迟到SPI闪存装置的SCK_out 512信号。通过这样做,SPI命令过滤器116可以防止命令或操作码的第八位由SPI闪存装置308注册或消耗。此外,通过对时钟进行门控并且取消断言芯片选择信号,SPI命令过滤器116可以避免对芯片选择线加毛刺,这可以防止SPI闪存装置308接收或解释任何不正确或未经授权的命令。虽然该示例实现由主机系统306向SPI装置308发送命令的情况,但是可以使用相同逻辑实现相反应用,即,如果SPI装置308向主机系统306发送未经授权的有效载荷或数据,则实施方式将防止主机消耗如该示例中的未经授权的通信。

[0058] 示例方法

[0059] 方法600到800图示为描绘可以执行的动作或操作的相应块集合,但不一定限于所示的用于由相应块执行操作的顺序或组合。此外,可以重复、组合、重组或链接一个或多个操作中的任一个,以提供各种各样的另外和/或替代方法。所描述的技术不限于由在一个系统或装置上操作的一个实体或多个实体执行。在各方面中,方法600到800的操作或动作由处理器、安全电路系统组件、存储器、安全SPI通信模块、SPI命令过滤器、SPI时钟控制,或配置为实现安全SPI通信的其它实体执行或管理。为了清楚起见,参考图1的元件描述方法,和/或参考图2到5和图9描述实体、组件或配置。

[0060] 图6图示了根据一个或多个方面的安全SPI通信的(多个)示例方法600,该安全SPI通信可以由SPI命令过滤器116或与主机或SPI互连件可操作地相关联的安全SPI通信模块实现。在各个方面中,SPI命令过滤器116或安全SPI通信模块可以实现方法600的操作,以在未授权或不允许下游命令的情况下阻止这些命令到达一个或多个外围块(例如,闪存存储器装置)。

[0061] 在602处,SPI命令过滤器116监测通过SPI互连件在系统的主机与系统的外围块之间交换的通信。例如,SPI命令过滤器能够窥探或监测串行外围接口互连件或总线的数据线,以获得通过SPI互连件传送的SPI通信的命令或操作码的位。在一些情况下,将命令从系统主机发送到SPI闪存装置。

[0062] 在604处,SPI命令过滤器将通信的相应命令与指示外围块未被授权执行的哪些命

令的信息相比较。在一些情况下，SPI命令过滤器使用接收到的命令或接收到的命令的一部分来索引化到表格中，以确定允许命令通过还是禁止(例如，过滤)命令。例如，启用位控制指示或控制允许命令向下游传递到外围装置还是过滤命令。在一些实施方式中，启用位值一“1”指示允许命令通过，并且启用位值零“0”指示由安全SPI通信模块或命令过滤器过滤命令。

[0063] 在606处，SPI命令过滤器基于比较确定相应命令中的一个是外围块未被授权执行的命令中的一个。换句话说，SPI命令过滤器检测在SPI互连件上传播到外围块的未允许或未经授权的命令。SPI命令过滤器可以基于部分或完整命令，诸如基于命令或对应操作码的第八位来确定命令是未经授权的。

[0064] 在608处，SPI命令过滤器防止外围块接收外围块未被授权执行的相应命令的至少一部分。在各方面中，SPI命令过滤器更改到外围块的SPI互连件的芯片选择线或时钟线的状态。在一些情况下，SPI命令过滤器对SPI外围设备的时钟进行门控，然后切换芯片选择线，这可以在SPI命令过滤器用于防止消耗未经授权的命令时防止对芯片选择线加毛刺。这可以防止外围块接收、消耗或处理通信的未经授权的命令的至少一部分。从操作608，方法600可以返回到操作602，以继续监测或窥探SPI互连件来寻找跨SPI互连件发布到系统的外围块或其它外围块的未经授权的命令。

[0065] 图7图示了根据一个或多个方面的用于过滤SPI通信的命令的示例方法700。在各个方面中，SPI命令过滤器116或安全SPI通信模块可以实现方法700的操作，以在未授权或未许可下游命令的情况下阻止这些命令到达一个或多个外围块(例如，闪存存储器装置)。

[0066] 在702处，SPI命令过滤器监测通过SPI互连件在主机与外围装置之间交换的通信的相应命令。例如，SPI命令过滤器能够窥探或监测串行外围接口互连件或总线的数据线，以获得通过SPI互连件传送的SPI通信的命令或操作码的位。在一些情况下，命令从系统主机发送到SPI闪存装置。

[0067] 在704处，SPI命令过滤器将相应命令与外围装置未被授权执行或接收的命令表相比较。在一些情况下，SPI命令过滤器使用接收到的命令或接收到的命令的一部分来索引化到表格中，以确定允许命令通过还是禁止(例如，过滤)命令。例如，启用位控制指示或控制允许命令向下游传递到外围装置还是过滤命令。在一些实施方式中，启用位值一“1”指示允许命令通过，并且启用位值零“0”指示由安全SPI通信模块或命令过滤器过滤命令。

[0068] 在706处，SPI命令过滤器确定相应命令是否被授权用于由外围装置执行。从706，响应于确认授权执行相应命令，方法前进到708，在708处，SPI命令过滤器经由SPI互连件将包括相应命令的通信传递到外围装置。

[0069] 替代地，响应于确定未授权执行相应命令，方法前进到710，在710处，SPI命令过滤器可以制定对策操作以防止外围装置接收或处理未授权的命令的至少一部分。为此，在710处，SPI命令过滤器对到外围装置的SPI时钟线进行门控。这样可以防止时钟节拍、转换或信号传播到外围芯片，这可以防止外围装置消耗或处理命令的至少一部分。在一些情况下，SPI命令过滤器在第八节拍或时钟循环上或附近对SPI时钟线进行门控，通过该第八节拍或时钟循环将第八位命令传送到外围装置，这可能导致外围装置丢弃未经授权的命令。

[0070] 在712处，SPI命令过滤器对到外围装置的SPI芯片选择线取消断言。在一些情况下，SPI命令过滤器在第八节拍或时钟循环上或附近对SPI芯片选择线取消断言，通过该第

八节拍或时钟循环将第八位命令传送到外围装置。在各方面中，在门控或暂停SPI时钟时对SPI芯片选择线取消断言(或断言，这取决于逻辑极性)使SPI命令过滤器能够切换SPI芯片选择线，而不对SPI芯片选择线或SPI外围设备加毛刺。通过这样做，SPI命令过滤器可以防止外围装置消耗或处理命令的第八位，这可能导致或强制外围装置丢弃未经授权的命令或其接收到的部分。在一些实施方式中，SPI命令过滤器可以在同一时间或短时间跨度内，诸如小于SCK周期的一半(例如，具有25MHz到100Mhz SCK的10到50纳秒)内对芯片选择线取消断言并且对时钟信号进行门控。这可以有效防止芯片选择线的取消断言对外围设备加毛刺。在714处，当主机释放上游芯片选择线时，SPI命令过滤器恢复SPI时钟线，这可以实现SPI互连件的连续操作。

[0071] 图8图示了根据一个或多个方面的用于操纵命令地址以实现替代二进制图像访问的示例方法800。在各个方面中，SPI命令过滤器116或安全SPI通信模块可以实现方法800的操作，以实现闪存存储器装置的A/B模式二进制图像或分区访问。

[0072] 在802处，系统的安全SPI通信模块或RoT装置经由SPI互连件访问SPI闪存存储器装置，该SPI互连件将系统的主机耦合到闪存存储器装置。闪存存储器装置可以包括多个分区，主机系统的不同二进制图像或其它信息存储在该多个分区上。在804处，安全SPI通信模块或RoT装置校验存储在SPI闪存存储器装置的第一分区上的二进制图像。

[0073] 可选地，在806处，响应于校验二进制图像是真实的，安全SPI通信模块或RoT装置允许系统的主机从SPI闪存存储器装置加载二进制图像。可选地，在808处，安全SPI通信模块操纵遍历SPI互连件的命令的一个或多个地址位，以访问SPI闪存存储器装置的第二分区。在810处，安全SPI通信模块使系统的主机能够从SPI闪存存储器装置的第二分区加载另一二进制图像。

[0074] 示例片上系统

[0075] 图9图示了根据一个或多个方面的能够实现安全SPI通信的示例片上系统900(SoC 900)的各种组件。SoC 900可以被实现为固定、移动、独立或嵌入式装置中的任何单个或多个；以任何形式的消费者、计算机、便携式、用户、服务器、通信、电话、导航、游戏、音频、相机、消息传递、媒体播放和/或其他类型的支持SoC的装置，诸如在图1中描绘或参考图1描述的那些设备102实现。所图示的组件中的一个或多个可以被实现为离散组件、模块、IP块，或实现为SoC 900的至少一个集成电路上的集成组件。通常，SoC 900的各种组件经由互连件120和/或根据安全SPI通信的一个或多个方面支持组件之间的通信的一个或多个结构耦合。

[0076] SoC 900能够包括一个或多个通信收发器124，该通信收发器实现装置数据110的有线和/或无线通信，该装置数据诸如是接收到的数据、传输的数据或上面识别的其它信息。通信收发器124的示例包括近场通信(NFC)收发器、符合各种IEEE 802.15(Bluetooth™)标准的无线个域网(PAN)(WPAN)无线电、符合各种IEEE 802.11(WiFi™)标准中的任一个的无线局域网(LAN)(WLAN)无线电，用于蜂窝连接的无线广域网(WAN)(WWAN)无线电(例如，符合第三代合作伙伴计划(符合3GPP)的WWAN无线电)、符合各种IEEE 802.16(WiMAXTM)标准的无线城域网(MAN)(WMAN)无线电、符合红外数据协会(IrDA)协议的红外(IR)收发器，以及有线局域网LAN以太网收发器。SoC 900还可以包括一个或多个数据输入/输出端口126(I/O端口126)，能够经由该I/O端口传送任何类型的数据、媒体内容和/或其它输入，诸如用户可

选择输入、消息、应用、音乐、电视内容、录制的视频内容以及从任何内容和/或数据源接收到的任何其它类型的音频、视频和/或图像数据，包括如麦克风或相机的传感器。数据I/O端口126可以包括USB端口、同轴电缆端口、用于光纤互连或布线的光纤端口以及用于可操作地耦合闪存存储器、光学介质写入器/读取器(例如，DVD、CD)等的其它串行或并行连接器(包括内部连接器)。这些数据I/O端口126可以用于将SoC耦合到组件、外围设备或附件，诸如键盘、麦克风、相机或其它传感器。

[0077] 该示例的SoC 900包括至少一个处理器106(例如，应用处理器、微处理器、数字信号处理器(DSP)、控制器等中的任一个或多个)，该处理器能够包括组合的处理器和存储器系统(例如，实现为SoC的一部分)，其处理(例如执行)计算机可执行指令以控制装置的操作。在各方面中，处理器106可以执行计算机可读指令(例如，操作系统或固件)以实现与SoC的其它组件和/或外围块交互的SoC 900的主机功能。处理器106可以被实现为应用处理器、嵌入式控制器、微控制器、安全处理器、人工智能(AI)加速器等。通常，处理器或处理系统可以至少部分地在硬件中实现，硬件能够包括集成电路或片上系统的组件、数字信号处理器(DSP)、专用集成电路(ASIC)、现场可编程门阵列(FPGA)、复杂的可编程逻辑装置(CPLD)以及硅和/或其它材料的其它实施方式。

[0078] 替代地或另外，SoC 900能够利用电子电路系统中的任何一个或组合来实现，该电子电路系统可以包括结合处理和控制电路实现的软件、硬件、固件或固定逻辑电路系统，该处理和控制电路通常在902处指示(作为电子电路系统902)。该电子电路系统902能够诸如通过存储在计算机可读介质上的处理/计算机可执行指令、通过逻辑电路系统和/或硬件(例如，诸如FPGA)等实现可执行的或基于硬件的模块(图9中未示出)。

[0079] 在各方面中，SoC 900包括互连件120，该互连件可以包括系统总线、链路、信道、互连件、交叉开关、数据传送系统或其它交换结构中的任一个或多个，该交换结构耦合装置内的各个组件，以实现与安全SPI信令和/或通信的各个方面。系统总线或互连件能够包括不同总线结构中的任一个或组合，诸如存储器总线或存储器控制器、SPI总线、SPI互连件、外围总线、奇偶校验块、CRC块、ECC块、TL-UL结构、通用串行总线，和/或利用各种总线架构中的任一个的处理器或本地总线。

[0080] SoC 900还包括实现数据存储的一个或多个存储器装置904，该存储器装置的示例包括随机存取存储器(RAM)、非易失性存储器(例如，只读存储器(ROM)、闪存存储器、可擦除可编程只读存储器(EPROM)和电可擦除可编程只读存储器(EEPROM))以及磁盘存储装置。存储器装置904中的一个或多个可以通过SPI互连件进行通信，该SPI互连件耦合到SPI命令过滤器116和/或SPI时钟控制器118，该SPI命令过滤器和/或时钟控制器实现如本文中所描述的安全SPI通信的各方面。(多个)存储器装置904可以被分布在系统的不同逻辑存储级别以及不同物理组件处。(多个)存储器装置904提供数据存储机制来存储装置数据110、其它类型的代码和/或数据，以及各种装置应用906(例如，软件应用或程序)。例如，操作系统908能够作为软件指令维护在存储器装置904内并且由处理器106执行。

[0081] 在一些实施方式中，SoC 900还包括音频和/或视频处理系统910，该音频和/或视频处理系统处理音频数据和/或将音频和视频数据传递到音频系统912和/或显示系统914(例如，视频缓冲器或智能手机或相机的屏幕)。音频系统912和/或显示系统914可以包括处理、显示和/或以其它方式呈现音频、视频、显示器和/或图像数据的任何装置。显示数据和

音频信号能够经由RF(射频)链路、S视频链路、HDMI(高清晰度多媒体接口)、复合视频链路、分量视频链路、DVI(数字视频接口)、模拟音频连接、视频总线，或诸如媒体数据端口916的其它类似通信链路传送到音频组件和/或显示组件。在一些实施方式中，音频系统912和/或显示系统914是SoC 900的外部或分离组件。替代地，例如，显示系统914能够是示例SoC900的集成组件，诸如集成触摸接口的一部分。

[0082] 图9的SoC 900可以是图1的设备102的示例实施方式、能够执行参考图1到图8所描述的安全SPI通信的方面的装置或系统的示例实施方式。因此，SoC 900能够包括安全电路系统112(例如，RoT装置)、SPI命令过滤器116，和/或SPI时钟控制器118，其能够是分开的电路系统或IP块，或包括为如处理器106、电子电路系统902或存储器装置904的另一IC芯片或装置的一部分。因此，所图示的组件中的一个或多个可以集成在相同的半导体衬底、半导体封装、IC芯片、SoC或单个印刷电路板(PCB)上。

[0083] 如图所示，SoC 900的安全电路系统112利用SPI命令过滤器116和SPI时钟控制器118的实例实现，该SPI命令过滤器和SPI时钟控制器可以被配置或包括如图1到5中所描述的组件(例如，作为安全SPI通信模块)。因此，安全电路系统112、SPI命令过滤器116和SPI时钟控制器118可以使SoC 900能够实现如本文中所描述的安全SPI通信的各方面。例如，SPI命令过滤器116能够监测由SoC 900的主机和SoC 900的闪存存储器模块(例如，存储器装置904)交换的通信。在各方面中，SPI命令过滤器116将由主机发送的通信的相应命令与指示闪存存储器模块未被授权执行的命令的哪些信息进行比较。基于该比较，SPI命令过滤器116确定相应命令中的一个是闪存存储器模块未被授权执行的命令中的一个(例如，写入状态寄存器命令)。响应于该确定，SPI命令过滤器116能够对闪存存储器模块的芯片选择线取消断言，或使用SPI块控制器118来门控或挂起闪存存储器模块的SPI时钟线。通过这样做，SPI命令过滤器116能够防止闪存存储器模块接收或处理未经授权的命令的至少一部分，这可以防止未经授权的命令包括闪存存储器模块的安全性。替代地或另外，SPI命令过滤器116或安全SPI通信模块可以校验存储在闪存存储器模块的第一分区上的二进制图像的完整性。如果完整性校验失败，则SPI命令过滤器116可以更改SPI互连件事务的地址，以使另一二进制图像从闪存存储器模块的第二分区加载，以防止主机从第一分区加载未经校验且可能受损的二进制图像。因此，如本文中所描述的安全SPI通信的概念能够由图9的SoC 900或结合图9的SoC 900实现。

[0084] 除非上下文另有规定，否则本文中使用词语“或”可以被视为使用“包含性或”，或许可包含或应用由单词“或”连接的一个或多个项目的术语(例如，短语“A或B”可以被解释为只许可“A”，只许可“B”，或同时许可“A”和“B”)。而且，如本文中所使用的，指代项目列表中的“至少一个”的短语是指那些项目的任何组合，包括单个成员。例如，“a、b或c中的至少一个”能够覆盖a、b、c、a-b、a-c、b-c和a-b-c，以及与多个相同元素的任何组合(例如，a-a、a-a-a、a-a-b、a-a-c、a-b-b、a-c-c、b-b、b-b-b、b-b-c、c-c和c-c-c，或a、b和c的任何其它排序)。此外，在本文中所讨论的附图和术语中表示的项目可以指示一个或多个项目或术语，因此可以互换地参考本书面描述中的项目和术语的单个或多个形式。尽管已经用特定于某些特征和/或方法的语言描述安全SPI通信的各方面，但是所附权利要求的主题不一定限于所描述的特定特征或方法。相反，将特定特征和方法公开为安全SPI通信的示例实施方式。

[0085] 在下文中，定义和描述安全SPI直通模块的示例实施方式：

```
module spi_passthrough
  import spi_device_pkg::*;
(
  input clk_i,     // SPI input clk
  input rst_ni,    // SPI reset
  input clk_out_i, // SPI output clk
```

[0086]

[0087] 配置命令过滤器信息能够作为256位寄存器给出。如果命令配置被存储在DPSRAM中,则可能被改变。如果被支持,则命令配置在第6个命令循环有效并且仅给定8个位。

[0088] input[255:0]cfg_cmd_filter_i,

[0089] //address manipulation(地址操纵)

[0090] input[31:0]cfg_addr_mask_i,

[0091] input[31:0]cfg_addr_value_i,

[0092] 地址模式:

[0093] input cfg_addr_4b_en_i,

[0094] input spi_mode_e spi_mode_i,

[0095] 对于SPI,直通模块能够重新使用现有spi_s2p和cmdparse。但是直通实现其自己的s2p和cmdparse来支持A/B二进制方案。

```
input                    host_sck_i,
input                    host_csb_i,
```

[0096]

```
input          [3:0] host_s_i,
output logic [3:0] host_s_o,      // clk_out_i domain
output logic [3:0] host_s_en_o, // clk_out_i domain
```

[0097] 在一些实施方式中,SPI到SPI_HOST以及端子到下游装置输出passthrough_req_t passthrough_o、输入passthrough_rsp_t passthrough_i,

[0098] 邮箱指示器,在各方面中,如果读取命令落入邮箱地址中并且启用邮箱特征,则“读取命令”处理模块向直通发送信号以控制SPI线路。如果该信号在地址阶段期间断言,则直通将CSb丢弃到SPI闪存装置并且等待主机的CSb取消断言。

```
  input mailbox_hit_i,
  // event
  // `cmd_filtered`: indicator of the incoming command filtered out（滤除
```

[0099]

```
的传入指令的指示符）
  output event_cmd_filtered_o
);
```

[0100] 与安全SPI直通相关联,一些定义如下定义:

[0101] 状态

[0102] typedef enum logic[2:0]{

[0103] 在空闲状态下,它将传入的SPI转发到SPI_HOST,并且最终转发到SPI闪存装置。

[0104] StIdle,

[0105] 当节拍命中第八节拍(准确地7.5)时,状态机能够检查传入数据,并且根据给定的config,cmd_filter确定阻止命令还是继续转发到下游装置。

[0106] StFilter,

[0107] 如果命令过滤,则该SCK到SPI_HOST Ip可能关闭,并且CSb在StWait中被取消断言。SCK应在StFilter中关闭。在该等待状态下,状态机等待来自主机系统的CSb取消断言。

[0108] StWait,

[0109] 输出命令处理-这大部分是SPI闪存模式中的复制代码,但存在于这里是为了控制输出启用。在直通模式中,如果命令被允许,则数据来自下游装置,但它不给出输出状态的任何提示(高Z或驱动)。因此,直通模块应遵循SPI协议来控制上游焊盘的‘oe’。它遵循读取命令模块、状态模块、SFDP/JEDEC模块中描述的相同协议。当St从StIdle或StAddress移动到StHighZ时,应设置等待计时器。等待计时器到期,StHighZ移动到StDriving,这是一个死胡同,直到CSb被取消断言。

[0110] StDriving,

[0111] StHighZ,

[0112] 在各方面中,可以如本文中所描述的实现地址操纵。例如,SPI直通的一个特殊特征是支持A/B二进制图像。逻辑能够将数据的特定节拍交换为预先配置的值。在该状态下,逻辑看到addr_mask和数据并在必要时交换线路。在此之后,ST移动到StDriving或StHighZ。如果地址命中邮箱区域和启用SW的邮箱,则ST取消当前事务并移动到StWait状态。

[0113] StAddress

[0114] }passthrough_st_e;

[0115] passthrough_st_e st,st_d;

[0116] 示例命令类型和各种命令

[0117] 直通模块可能不严格遵循SPI线上的每个位,而是松散地跟踪命令的相位。SPI闪存中的命令能够被分类为以下几类:

// - {Address, PayloadOut}: 示例是 ReadData

// - {Address, Dummy, PayloadOut}: FastRead/Doul/Quad 命令具有伪命令

// - {Dummy, PayloadOut}: 释放电源关闭/制造商 ID

// - {PayloadOut}: 在操作码之后，装置立即向主机发送数据

[0118] // - {Address, PayloadIn}: 主机背靠背发送地址和有效载荷

// - {PayloadIn}: 主机在没有任何地址提示的情况下发送有效载荷

// - None: 命令在没有任何地址有效载荷（输入/输出）的情况下完成

[0119] 如果接收到的命令具有多于一个状态,则计数器值将//设置为帮助状态机在准确的定时移动到下一个状态。`cmd_type_t`结构具有命令的信息。命令的实际值是编译时间参数。当逻辑接收到8位操作码时,它将参数锁存到该结构中并通过事务引用该结构。

[0120] 操作码计数器之后的示例地址或任何主机驱动

[0121] localparamint unsigned MaxAddrBit＝32;

[0122] localparamint unsigned AddrCntW＝$clog2(MaxAddrBit);

[0123] 示例伪命令

```
localparam int unsigned MaxDummyBit = 8;
localparam int unsigned DummyCntW = $clog2(MaxDummyBit);
typedef enum logic {
```

[0124]

```
    PayloadIn  = 1'b 0,
    PayloadOut = 1'b 1
} payload_dir_e;
typedef struct packed {
```

[0125] 地址存在

[0126] logic addr_en;

[0127] 如果swap_en是1,则逻辑将incomind addr替换为某些位的预配置值。

[0128] logic addr_swap_en;

[0129] 如果为1,则addr_size受‘cfg_addr_4b_en_i’逻辑addr_4b_affected的影响

[0130] 示例,当伪命令存在时

[0131] logic dummy_en;

[0132] 有效载荷方向:如果设置payload_en,则命令在任一方向上具有有效载荷。‘payload_dir’确定输入(0)或输出(1)。

[0133] logic[3:0]payload_en;

[0134] payload_dir_e payload_dir;

[0135] 在各方面中,基于‘addr_4b_affected’确定addr_size。如果1和‘cfg_addr_4b_en_i’ir 1,则‘addr_size’是31。否则其被设置成23。

logic [AddrCntW-1:0] addr_size;

[0136] logic [DummyCntW-1:0] dummy_size;

} cmd_type_t;

[0137]

```
localparam cmd_type_t CmdInfoNone = '{
    addr_en:                1'b 0,
    addr_swap_en:           1'b 0,
    addr_4b_affected: 1'b 0,
    dummy_en:                 1'b 0,
    payload_en:             4'h 0,
    payload_dir:    PayloadIn,
    addr_size:                 '0,
    dummy_size:                 '0
};
localparam cmd_type_t CmdInfoPayloadIn = '{
    addr_en:                1'b 0,
    addr_swap_en:           1'b 0,
    addr_4b_affected: 1'b 0,
    dummy_en:                 1'b 0,
    payload_en:             4'h 1,
    payload_dir:    PayloadIn,
    addr_size:                 '0,
    dummy_size:                 '0
};
localparam cmd_type_t CmdInfoPayloadOut = '{
    addr_en:                1'b 0,
    addr_swap_en:           1'b 0,
    addr_4b_affected: 1'b 0,
    dummy_en:                 1'b 0,
    payload_en:             4'h 2, // S[1]
    payload_dir: PayloadOut,
    addr_size:                 '0,
    dummy_size:                 '0
};
```

```
localparam cmd_type_t CmdInfoAddrPayloadIn = '{
  addr_en:          1'b 1,
  addr_swap_en:     1'b 0,
  addr_4b_affected: 1'b 1,
  dummy_en:         1'b 0,
  payload_en:       4'h 1, // S[0] only（仅 S[0]）
  payload_dir:  PayloadIn, // Host sends Data
  addr_size:          '0, // Logic decide（逻辑决定）
  dummy_size:         '0
};
localparam cmd_type_t CmdInfoAddrPayloadInQuad = '{
  addr_en:          1'b 1,
  addr_swap_en:     1'b 0,
  addr_4b_affected: 1'b 1,
  dummy_en:         1'b 0,
  payload_en:       4'h F, // S[3:0]
  payload_dir:  PayloadIn, // Host sends Data（主机发送数据）
  addr_size:          '0, // Logic decide（逻辑决定）
  dummy_size:         '0
};
localparam cmd_type_t CmdInfoAddrPayloadOut = '{
  addr_en:          1'b 1,
  addr_swap_en:     1'b 1,
  addr_4b_affected: 1'b 1,
  dummy_en:         1'b 0,
  payload_en:       4'h 2, // S[1] only（仅 S[1]）
  payload_dir: PayloadOut, // Flash device sends Data（闪存装置发送
数据）
  addr_size:          '0, // Logic decide（逻辑决定）
  dummy_size:         '0
```

[0138]

```
};
// Address + Dummy + Payload but Address is 3B always（地址+伪命令+有效载荷但地址总是 3B） localparam cmd_type_t CmdInfoAddr3BDummyPayloadOut = '{
    addr_en:          1'b 1,
    addr_swap_en:     1'b 0,
    addr_4b_affected: 1'b 0,
    dummy_en:         1'b 1,
    payload_en:       4'h 2, // S[1] only（仅 S[1]）
    payload_dir: PayloadOut, // Flash device sends Data（闪存装置发送数据）
    addr_size:        '0, // Logic decide（逻辑决定）
    dummy_size:       'h 7
};
localparam cmd_type_t CmdInfoAddrDummyPayloadOut = '{
    addr_en:          1'b 1,
    addr_swap_en:     1'b 1,
    addr_4b_affected: 1'b 1,
    dummy_en:         1'b 1,
    payload_en:       4'h 2, // S[1] only（仅 S[1]）
    payload_dir: PayloadOut, // Flash device sends Data（闪存装置发送数据）
    addr_size:        '0, // Logic decide（逻辑决定）
    dummy_size:       'h 7
};
localparam cmd_type_t CmdInfoAddrDummyPayloadOutDual = '{
    addr_en:          1'b 1,
    addr_swap_en:     1'b 1,
    addr_4b_affected: 1'b 1,
    dummy_en:         1'b 1,
```

[0139]

```
    payload_en:          4'h 3, // S[1:0] only（仅 S[1:0]）
    payload_dir: PayloadOut, // Flash device sends Data（闪存装置发送
数据）
    addr_size:            '0, // Logic decide（逻辑决定）
    dummy_size:          'h 7
  };
  localparam cmd_type_t CmdInfoAddrDummyPayloadOutQuad = '{
    addr_en:             1'b 1,
    addr_swap_en:        1'b 1,
    addr_4b_affected: 1'b 1,
    dummy_en:            1'b 1,
    payload_en:          4'h F, // S[3:0]
    payload_dir: PayloadOut, // Flash device sends Data（闪存装置发送
数据）
    addr_size:            '0, // Logic decide（逻辑决定）
    dummy_size:          'h 7
  };
  localparam cmd_type_t CmdInfoAddr = '{
    addr_en:             1'b 1,
    addr_swap_en:        1'b 0,
    addr_4b_affected: 1'b 0, // TODO: ??
    dummy_en:            1'b 0,
    payload_en:          4'h 0,
    payload_dir: PayloadOut, // Flash device sends Data（闪存装置发送
数据）
    addr_size:            '0, // Logic decide（逻辑决定）
    dummy_size:          'h 0
  };
  localparam cmd_type_t PassThroughCmdInfo [256] = '{
    CmdInfoNone,                          // 8'h 00
```

[0140]

[0141]

```
CmdInfoPayloadIn,                    // 8'h 01 Write Status 1（写入状态 1）
CmdInfoAddrPayloadIn,                // 8'h 02 Page Program（页程序）
CmdInfoAddrPayloadOut,               // 8'h 03 Read Data（读取数据）
CmdInfoNone,                         // 8'h 04 Write Disable（写入禁用）
CmdInfoPayloadOut,                   // 8'h 05 Read Status 1（读取状态 1）
CmdInfoNone,                         // 8'h 06 Write Enable（写入启用）
CmdInfoNone,                         // 8'h 07
CmdInfoNone,                         // 8'h 08
CmdInfoNone,                         // 8'h 09
CmdInfoNone,                         // 8'h 0A
CmdInfoAddrDummyPayloadOut,          // 8'h 0B Fast Read（快速读取）
CmdInfoNone,                         // 8'h 0C
CmdInfoNone,                         // 8'h 0D
CmdInfoNone,                         // 8'h 0E
CmdInfoNone,                         // 8'h 0F
CmdInfoNone,                         // 8'h 10
CmdInfoPayloadIn,                    // 8'h 11 Write Status 3（写入状态 3）
CmdInfoNone,                         // 8'h 12
CmdInfoNone,                         // 8'h 13
CmdInfoNone,                         // 8'h 14
CmdInfoPayloadOut,                   // 8'h 15 Read Status 3（读取状态 3）
```

[0142]

```
CmdInfoNone,                          // 8'h 16
...
CmdInfoNone,                          // 8'h 1F
CmdInfoAddr,                          // 8'h 20 Sector Erase (4kB)
（扇区擦除）
CmdInfoNone,                          // 8'h 21
...
CmdInfoNone,                          // 8'h 30
CmdInfoPayloadIn,                     // 8'h 31 Write Status 2（写入
状态 2）
CmdInfoAddrPayloadInQuad,             // 8'h 32 Quad Input Page
Program（四输入页程序）
CmdInfoNone,                          // 8'h 33
CmdInfoNone,                          // 8'h 34
CmdInfoPayloadOut,                    // 8'h 35 Read Status 2（读取
状态 2）
CmdInfoAddr,                          // 8'h 36 Individual Block
Lock（单个块锁定）
CmdInfoNone,                          // 8'h 37
CmdInfoNone,                          // 8'h 38 Enter QPI (filtered)
（输入 OPI（过滤的））
CmdInfoAddr,                          // 8'h 39 Individual Blck
Unlock（单个块解锁）
CmdInfoNone,                          // 8'h 3A
CmdInfoAddrDummyPayloadOutDual,       // 8'h 3B Fast Read Dual Out
（快速读取双输出）
CmdInfoNone,                          // 8'h 3C
CmdInfoAddrPayloadOut,                // 8'h 3D Read Block Lock（读
取块锁定）
CmdInfoNone,                          // 8'h 3E
```

```
CmdInfoNone,                        // 8'h 3F
CmdInfoNone,                        // 8'h 40
CmdInfoNone,                        // 8'h 41
CmdInfoNone,                        // 8'h 42 TODO
CmdInfoNone,                        // 8'h 43
CmdInfoNone,                        // 8'h 44 TODO
CmdInfoNone,                        // 8'h 45
CmdInfoNone,                        // 8'h 46
CmdInfoNone,                        // 8'h 47
CmdInfoNone,                        // 8'h 48 TODO
CmdInfoNone,                        // 8'h 49
CmdInfoNone,                        // 8'h 4A
CmdInfoNone,                        // 8'h 4B Read Unique ID
(TODO)（读取唯一 ID）
```

[0143]

```
CmdInfoNone,                        // 8'h 4C
CmdInfoNone,                        // 8'h 4D
CmdInfoNone,                        // 8'h 4E
CmdInfoNone,                        // 8'h 4F
CmdInfoNone,                        // 8'h 50
CmdInfoNone,                        // 8'h 51
CmdInfoAddr,                        // 8'h 52 Block Erase (32kB)
（块擦除）
CmdInfoNone,                        // 8'h 53
CmdInfoNone,                        // 8'h 54
CmdInfoNone,                        // 8'h 55
CmdInfoNone,                        // 8'h 56
CmdInfoNone,                        // 8'h 57
CmdInfoNone,                        // 8'h 58
CmdInfoNone,                        // 8'h 59
CmdInfoAddr3BDummyPayloadOut,       // 8'h 5A Read SFDP（读取
```

SFDP）

```
    CmdInfoNone,                          // 8'h 5B
    ...
    CmdInfoNone,                          // 8'h 6A
    CmdInfoAddrDummyPayloadOutQuad, // 8'h 6B Fast Read Quad Out
```

（快速读取四输出）

```
    CmdInfoNone,                          // 8'h 6C
    ...
    CmdInfoNone,                          // 8'h 9E
    CmdInfoPayloadOut,                   // 8'h 9F JEDEC ID
    CmdInfoNone,                          // 8'h A0
    ...
    CmdInfoNone,                          // 8'h D7
    CmdInfoAddr,                          // 8'h D8 Block Erase (64kB)
```

（块擦除）

[0144]

```
    CmdInfoNone,                          // 8'h D9
    ...
    CmdInfoNone                           // 8'h FF
  };
```

DC 中不可合成的示例 localparam cmd_type_t PassThroughCmdInfoOld [256] = '{

```
    // 8'h 00
    'h 00: CmdInfoNone,
    // 8'h 01 Write Status 1（写入状态 1）
    'h 01: CmdInfoPayloadIn,
    // 8'h 15 Write Statur 2（写入状态 2）
    'h 31: CmdInfoPayloadIn,
    // 8'h 11 Write Status 3（写入状态 3）
    'h 11: CmdInfoPayloadIn,
    // 8'h 02 Page Program（页程序）
```

'h 02: CmdInfoAddrPayloadIn,

// 8'h 32 Quad Input Page Program : Expect to be filtered（四输入页程序：预期被过滤）

'h 32: CmdInfoAddrPayloadInQuad,

// 8'h 03 Read Data（读取数据）

'h 03: CmdInfoAddrPayloadOut,

// 8'h 04 Write Disable（写入禁用）

'h 04: CmdInfoNone,

// 8'h 05 Read Status 1（读取状态 1）

'h 05: CmdInfoPayloadOut,

// 8'h 35 Read Status 2（读取状态 2）

'h 35: CmdInfoPayloadOut,

// 8'h 15 Read Status 3（读取状态 3）

'h 15: CmdInfoPayloadOut,

[0145]

// 8'h 06 Write Enable（写入启用）

'h 06: CmdInfoNone,

// 8'h 0B Fast Read（快速读取）

'h 0B: CmdInfoAddrDummyPayloadOut,

// 8'h 3B Fast Read Dual Output（快速读取双输出）

'h 3B: CmdInfoAddrDummyPayloadOutDual,

// 8'h 6B Fast Read Quad Output（快速读取四输出）

'h 6B: CmdInfoAddrDummyPayloadOutQuad,

// 8'h 20 Sector Erase (4kB)（扇区擦除）

'h 20: CmdInfoAddr,

// 8'h 52 Block Erase (32kB)（块擦除）

'h 52: CmdInfoAddr,

// 8'h D8 Block Erase (64kB)（块擦除）

'h D8: CmdInfoAddr,

// 8'h 36 Individual Block Lock（单个块锁定）

'h 36: CmdInfoAddr,

```
// 8'h 39 Individual Block Unlock（单个块解锁）
'h 39: CmdInfoAddr,
// 8'h 3D Read Block Lock（读取块锁定）
'h 3D: CmdInfoAddrPayloadOut,
// 8'h 38 Enter QPI : Expect to be filtered（输入 QPI：预期被过滤）
'h 38: CmdInfoNone,
// 8'h 42 Program Security Register（程序安全寄存器）
// 8'h 44 Erase Security Register（擦除安全寄存器）
// 8'h 48 Read Security Register（读取安全寄存器）
// 8'h 4B Read Unique ID（读取唯一 ID）
// 8'h 5A Read SFDP（读取 SFDP）
'h 5A: CmdInfoAddr3BDummyPayloadOut,
// 8'h 90 Manufacture/Device ID（制造/装置 ID）
// 8'h 9F JEDEC ID
'h 9F: CmdInfoPayloadOut,
default: CmdInfoNone
};
*/
```

[0146]

[0147] 示例信号

[0148] 内部时钟

[0149] logic[3:0]host_s_en_inclk;

[0150] logic[3:0]device_s_en_inclk;

[0151] 指示直通模式是否被启用

[0152] logic is_active;

[0153] assignis_active＝(spi_mode_i＝＝PassThrough);

[0154] logic[7:0]opcode,opcode_d;

[0155] 如果过滤器在第8节拍变为1,则它将SCK启用信号降低到CG单元,然后,在SCK的第8上升沿,csb_deassert变为1。

[0156] 逻辑过滤器;

[0157] 如果为1,则SCK传播到下游SPI闪存装置。

[0158] logic sck_gate_en;

[0159] CSb到下游装置控制,如果为1,则取消断言CSb。该信号//对毛刺敏感。该值在SCK上升沿处改变。这不会//直接驱动CSb输出。CSb到下游与该值以及来自主机系统的CSb进行OR。

[0160] logic csb_deassert;

[0161] 开始计数器-bitcnt计数到伪命令

[0162] localparam int unsigned MaxBeat＝8+32+8;//Cmd+Addr+Dummy

[0163] localparam int unsigned BitCntW＝$clog2(MaxBeat);

[0164] logic[BitCntW-1:0]bitcnt;

[0165] 操作码计数器之后的地址或任何主机驱动

[0166] logic[AddrCntW-1:0]addrcnt,addrcnt_outclk;

[0167] 伪命令计数器

[0168] logic[DummyCntW-1:0]dummycnt,dummycnt_d;

[0169] 结束:计数器

[0170] 示例事件

[0171] assign event_cmd_filtered_o＝filter;

[0172] 示例邮箱击中

[0173]
```
logic mailbox_hit;
always_ff @(posedge clk_i or negedge rst_ni) begin
  if (!rst_ni) mailbox_hit <= 1'b 0; // reset by CSb（由 Scb 复位）
  else if (mailbox_hit_i) mailbox_hit <= 1'b 1; // set by event（由事件设置）
end
```

[0174] 示例数据路径

[0175] 操作码锁存

[0176]
```
assign opcode_d = {opcode[6:0], host_s_i[0]};
always_ff @(posedge clk_i or negedge rst_ni) begin
  if (!rst_ni) begin
    opcode <= 8'h 00;
  end else if (bitcnt < BitCntW'(8)) begin
    opcode <= opcode_d;
```

[0177]
```
  end
end
```

[0178] 命令过滤器:CSb控制

[0179]
```
always_ff @(posedge clk_i or negedge rst_ni) begin
  if (!rst_ni)        csb_deassert <= 1'b 0;
  else if (filter) csb_deassert <= 1'b 1;
end
```

[0180] 查看上面的波形以了解sck_gate_en为什么是过滤器ORcsb_deassert的反转

[0181] assign sck_gate_en＝～(filter|csb_deassert)；

[0182] 示例Bitcnt计数器，Bitcnt增加，直到达到最大值，然后等待复位。

```
always_ff @(posedge clk_i or negedge rst_ni) begin
  if (!rst_ni) begin
    bitcnt <= '0;
```

[0183]

```
  end else if (bitcnt != '1) begin
    bitcnt <= bitcnt + BitCntW'(1);
  end
end
```

[0184] 示例在第7个cmd操作码处仅锁存两个位

```
logic cmd_7th; // 7th beat of transaction（业务的第 7 节拍）
logic cmd_8th; // in 8th beat of transaction（在业务的第 8 节拍中）
logic [1:0] cmd_filter;
assign cmd_7th = (bitcnt == BitCntW'(6));
```

[0185]

```
assign cmd_8th = (bitcnt == BitCntW'(7));
always_ff @(posedge clk_i or negedge rst_ni) begin
  if (!rst_ni) begin
    cmd_filter <= 2'b 00;
  end else if (cmd_7th) begin
```

[0186] 在该示例第7节拍中，cmd_filter从cfg_cmd_filter_i锁存2个位。

[0187] 这将最后一个过滤器数据路径从多路复用256减少到仅多路复用器2如果以下语法不起作用，则能够替换为

```
    for (int unsigned i = 0 ; i < 128 ; i++) begin
    it (i == opcode[6:0]) cmd_filter <= cfg_cmd_filter_i[2*i+:2];
    end
```

[0188]

```
    cmd_filter <= cfg_cmd_filter_i[{opcode_d[6:0],1'b0}+:2];
  end
end
```

[0189] 命令信息锁存器的示例

```
cmd_type_t cmd_info, cmd_info_d;
cmd_type_t [1:0] cmd_info_7th;
logic cmd_info_latch;
always_ff @(posedge clk_i or negedge rst_ni) begin
  if (!rst_ni) begin
    cmd_info_7th <= '0;
  end else if (cmd_7th) begin
```

[0190]

```
    cmd_info_7th <= {PassThroughCmdInfo[{opcode_d[6:0],1'b1}],
                     PassThroughCmdInfo[{opcode_d[6:0],1'b0}]};
  end
end
always_ff @(posedge clk_i or negedge rst_ni) begin
  if (!rst_ni) begin
    cmd_info <= '0;
  end else if (cmd_info_latch) begin
```

[0191] 当第7位到达时,仅锁存两个cmd_info的示例。然后在cmd_info_d的第8节拍处的两者中进行选择以减少定时。

```
    cmd_info <= cmd_info_d;
  end
```

[0192]

```
end
always_comb begin
cmd_info_d = '0;
```

[0193]

```
if (cmd_8th) begin
```

[0194] 当第7位到达时,仅锁存两个cmd_info的示例。然后在cmd_info_d的第8节拍处的两者中进行选择以减少定时。

```
        cmd_info_d = cmd_info_7th[host_s_i[0]];
        TODO 的示例：地址大小
        if (cmd_info_7th[host_s_i[0]].addr_4b_affected) begin
          cmd_info_d.addr_size = (cfg_addr_4b_en_i)
```

[0195]

```
                                  ? AddrCntW'(31) : AddrCntW'(23);
        end
      dummy_size 何时在状态机内设置的示例
      end
    end
```

[0196] 示例地址交换

```
      logic addr_set;
      logic addr_phase, addr_phase_outclk;
      assign addr_phase = (st == StAddress);
      always_ff @(posedge clk_i or negedge rst_ni) begin
        if (!rst_ni) begin
          addrcnt <= '0;
        end else if (addr_set) begin
            // When addr_set is 1, cmd_info is not yet latched.（当 addr_set
```

[0197]

```
是 1，cmd_info 尚未被锁存。）
            addrcnt <= cmd_info_d.addr_size;
        end else if (addrcnt != '0) begin
          addrcnt <= addrcnt - AddrCntW'(1);
        end
      end
      always_ff @(posedge clk_out_i or negedge rst_ni) begin
        if (!rst_ni) addrcnt_outclk <= '0;
      else          addrcnt_outclk <= addrcnt;
```

[0198]

```
    end
```

[0199] 基于AddrCnt,逻辑交换。

[0200] TODO:处理DualIO、QuadIO情况

[0201]

```
logic addr_swap;
```

[0202]

```
assign addr_swap＝cfg_addr_mask_i[addrcnt_outclk]
```

[0203] ?cfg_addr_value_i[addrcnt_outclk]

[0204] :host_s_i[0];

[0205] 在各方面中,地址交换发生在outlk域中。状态机在inclk域中操作。状态生成多路复用器选择信号。锁存在outlk域中的信号然后激活多路复用器。

[0206]
```
always_ff @(posedge clk_out_i or negedge rst_ni) begin
  if (!rst_ni) addr_phase_outclk <= 1'b 0;
  else         addr_phase_outclk <= addr_phase;
end
```

[0207] 示例伪命令计数器

[0208]
```
logic dummy_set;
always_ff @(posedge clk_i or negedge rst_ni) begin
  if (!rst_ni) dummycnt <= '0;
  else if (dummy_set) begin
    dummycnt <= dummycnt_d;
  end
end
```

[0209] 直通多路复用器的示例,由于addr_phase_outclk在outclk域中,可以直接使用addr_swap。addr_swap值也由addrcnt_outclk确定。

```
assign passthrough_o.s = (addr_phase_outclk)
                          ? {host_s_i[3:1], addr_swap} : host_s_i;
```

[0210]
```
logic [3:0] passthrough_s_en;
always_ff @(posedge clk_out_i or negedge rst_ni) begin
  if (!rst_ni) passthrough_s_en <= 4'h 1; // S[0] active by default
```

```
    else passthrough_s_en <= device_s_en_inclk;
  end
  assign passthrough_o.s_en = passthrough_s_en;
  assign host_s_o = passthrough_i.s;
  always_ff @(posedge clk_out_i or negedge rst_ni) begin
[0211]  if (!rst_ni) host_s_en_o <= '0; // input mode
    else          host_s_en_o <= host_s_en_inclk;
  end
  assign passthrough_o.sck_gate_en = sck_gate_en;
  assign passthrough_o.sck         = host_sck_i;
  assign passthrough_o.sck_en      = 1'b 1;
```

[0212] CSb传播的示例:CSb_deassert信号应该是FF或锁存器的输出,以使CSb无毛刺。

```
    assign passthrough_o.csb_en = 1'b 1;
    assign passthrough_o.csb    = host_csb_i | csb_deassert ;
```

[0213]

```
  passthrough_en
    assign passthrough_o.passthrough_en = is_active ;
```

[0214] END:Passthrough Mux结束:直通多路复用器

[0215] 示例状态机

```
    always_ff @(posedge clk_i or negedge rst_ni) begin
      if (!rst_ni) begin
        st <= StIdle;
      end else begin
        st <= st_d;
[0216]  end
    end
    always_comb begin
      st_d = st;
  an example filter
      filter = 1'b 0;
```

```
        an example Command Cfg Latch
          cmd_info_latch = 1'b 0;
        an example addr_set
          addr_set = 1'b 0;
      an example Dummy
          dummy_set = 1'b 0;
          dummycnt_d = '0;
        an example Output enable
[0217]    host_s_en_inclk      = 4'h 0;
          device_s_en_inclk = 4'h 1; // S[0] MOSI active
          unique case (st)
            StIdle: begin
              if (!is_active) begin
                st_d = StIdle;
              end else if (cmd_8th && cmd_filter[host_s_i[0]]) begin
                st_d = StFilter;
                filter = 1'b 1;
```

[0218] 示例将通知事件发送到SW

[0219] end else if(cmd_8th)begin

[0220] cmd_info_latch=1'b 1;

[0221] 转移到多个状态的示例。以下状态主要用于控制输出启用信号。然而,StAddress状态控制SPI线,以便在读取命令的情况下进行地址交换。

```
          // Order: addr_en , dummy_en, |payload_en 顺序： addr_en ,
      dummy_en, |payload_en
          if (cmd_info_d.addr_en) begin
[0222]      st_d = StAddress;
            addr_set = 1'b 1;
          end else if (cmd_info_d.dummy_en) begin
            st_d = StHighZ;
            dummy_set = 1'b 1;
```

dummycnt_d = cmd_info_d.dummy_size;

[0223]

end else if (cmd_info_d.payload_en != 0) begin

[0224] 任何输入/输出有效负载的示例

if (cmd_info_d.payload_dir == PayloadOut) begin

st_d = StWait;

end else begin

st_d = StDriving;

[0225]

end

end

end

end

StFilter: begin

[0226] 命令被过滤。等待复位(CSb)到来。

st_d = StFilter;

host_s_en_inclk = 4'h 0; // explicit

[0227] device_s_en_inclk = 4'h 0;

end

StWait: begin

[0228] 设备向主机返回数据

[0229] st_d=StWait;

[0230] 主机输出启用

host_s_en_inclk = cmd_info.payload_en;

device_s_en_inclk = 4'h 0;

[0231]

end

StDriving: begin

[0232] 主机向设备发送数据st_d=StDriving;

[0233] 为设备输出启用

host_s_en_inclk = 4'h 0; // explicit

[0234] device_s_en_inclk = cmd_info.payload_en;

end

```
StHighZ: 开始
  host_s_en_inclk     = 4'h 0; // explicit
  device_s_en_inclk = 4'h 0; // float
  if (dummycnt == '0) begin
    // Assume payload_en not 0（假设 payload_en 不是 0）
    st_d = (cmd_info.payload_dir == PayloadOut) ? StWait :
StDriving;
  end
end
StAddress: begin
  // based on state, addr_phase is set. So just check if counter
reaches 0（基于状态，设置 addr_phase。所以只检查计数器是否达到 0）
  if (addrcnt == '0) begin
    if (mailbox_hit_i) begin
      // In Address phase, mailbox region hits. Then Passthrough
filters（在地址阶段，邮箱区域命中。然后直通过滤器）
      // the command and deligates the control to ReadCmd
submodule. （命令并将控制权委托给 Read Cmd 子模块）
      st_d = StFilter;
      filter = 1'b 1;
    end else if (cmd_info.dummy_en) begin
      st_d = StHighZ;
      dummy_set = 1'b 1;
      dummycnt_d = cmd_info.dummy_size;
    end else if (cmd_info.payload_en != 0) begin
      st_d = (cmd_info.payload_dir == PayloadOut) ? StWait :
StDriving;
    end else begin
      // Addr completed command. goto wait state （Addr 完成
命令。进入等待状态）
```

[0235]

```
                st_d = StWait;
              end
            end
          end
[0236]    default: begin
            st_d = StIdle;
          end
        endcase
      end
```

[0237] 诸如当邮箱命中发生在地址阶段的中间,而不是结束时的示例断言。

[0238] `ASSERT(MailboxHitConflictAddrCnt_A,mailbox_hit_i|->(addrcnt!=0))

[0239] endmodule:spi_passthrough

[0240] 另外的示例

[0241] 下文提供安全SPI通信的示例:

[0242] 示例1:一种由安全电路系统实现的方法,所述安全电路系统与用于安全串行外围接口通信的系统的主机相关联,所述方法包括:监测由所述主机传输到所述系统的外围块的通信,所述外围块经由串行外围接口(SPI)互连件耦合到所述主机;将由所述主机发送的所述通信的相应命令与指示所述外围块未被授权执行的命令的哪些信息相比较;基于所述比较,确定相应命令中的一个是所述外围块未被授权执行的命令;以及

[0243] 防止所述外围块接收所述外围块未被授权执行的相应命令的至少一部分。

[0244] 示例2.1:根据示例中任一项所述的方法,其中,所述防止包括更改所述SPI互连件的芯片选择线或时钟线的状态,以防止所述外围块接收由所述主机发送的所述通信的相应命令的所述至少一部分。

[0245] 示例2.2:根据示例中任一项所述的方法,其中,所述防止包括更改所述SPI互连件的芯片选择线或时钟线的状态,以防止所述外围块接收由所述主机发送的所述通信的相应命令的所述至少一部分。

[0246] 示例3:根据示例中任一项所述的方法,其中所述通信的相应命令包括八个位;并且所述更改包括在经由所述SPI互连件传送相应命令的第八位第八时钟循环之前或在第八时钟循环上对所述芯片选择线取消断言,以防止相应命令的至少所述第八位由所述外围块处理。

[0247] 示例4:根据示例中任一项所述的方法,其中:所述通信的相应命令包括八个位;并且所述更改包括在经由所述SPI互连件传送相应命令的第八位的第八时钟循环之前或在第八时钟循环上对所述时钟线进行门控,以防止相应命令的至少所述第八位由所述外围块处理。

[0248] 示例5:根据示例中任一项所述的方法,其中所述系统的所述外围块不包括写入保护输入节点,或禁用所述外围块的写入保护输入节点。

[0249] 示例6:根据示例中任一项所述的方法,其中相应命令是第一通信的第一命令,并

且所述方法进一步包括:确定由所述主机发送到所述外围块的第二通信的第二命令包括写入状态寄存器命令;以及更改所述写入状态寄存器命令以防止所述第二通信更改所述外围块的状态寄存器。

[0250] 示例7:根据示例中任一项所述的方法,其中所述写入状态寄存器命令的所述更改包括:将所述写入状态寄存器命令的位位置设置为预定义位值以提供更改后的写入状态寄存器命令;以及将所述更改后的写入状态寄存器命令传递到所述外围块,所述更改后的写入状态寄存器命令包括所述预定义位值和所述写入状态寄存器命令的其它位。

[0251] 示例8:根据示例中任一项所述的方法,其中指示所述外围块未被授权执行的所述命令的所述信息包括以下项的表:所述外围块未被授权执行的所述命令;以及所述外围块被授权执行的命令。

[0252] 示例9:根据示例中任一项所述的方法,进一步包括在所述通信的所述监测之前,配置所述外围块未被授权执行的所述命令和所述外围块被授权执行的所述命令的所述表。

[0253] 示例10:根据示例中任一项所述的方法,其中由所述主机发送的所述通信的相应命令与所述表的所述信息的所述比较包括:基于相应命令索引化到所述表中,以确定所述外围块是否被授权执行相应命令。

[0254] 示例11:根据示例中任一项所述的方法,其中所述系统的所述外围块包括经由所述SPI互连件耦合到所述主机的存储器模块,所述存储器模块包括非易失性存储器或闪存存储器中的一个。

[0255] 示例12:根据示例中任一项所述的方法,进一步包括在尝试经由所述SPI互连件将存储在所述系统的所述存储器模块上的二进制图像从所述存储器模块加载到所述主机之前,校验所述二进制图像。

[0256] 示例13:根据示例中任一项所述的方法,包括:响应于所述二进制图像的成功校验,允许所述存储器模块经由所述SPI接口将所述二进制图像发送到所述主机;或响应于所述二进制图像的不成功校验,防止所述存储器模块经由所述SPI接口将所述二进制图像发送到所述主机。

[0257] 示例14:根据示例中任一项所述的方法,其中所述二进制图像是存储在所述存储器模块的第一分区中的第一二进制图像,并且所述方法进一步包括:操纵用于所述第一二进制图像的读取命令的地址,以使所述存储器模块从所述存储器模块的第二分区向所述主机发送第二二进制图像。

[0258] 示例15:一种集成电路,所述集成电路包括用于安全串行外围接口通信的电路系统,所述电路系统包括:具有功能核心的主机;至少一个外围块;串行外围接口(SPI)互连件,其耦合所述主机和所述至少一个外围块;以及安全SPI通信模块,其与所述SPI互连件可操作地耦合并且被配置成执行根据前述示例中任一项所述的操作。

[0259] 结论

[0260] 尽管已经用特定于特征和/或方法的语言描述用于实现安全SPI通信的所描述系统和方法的方面,但是所附权利要求的主题如根据前述示例中任一项所述不一定限于所描述的特定特征或方法。相反,将特定特征和方法公开为安全SPI通信的示例实施方式,并且其它等效特征和方法预期在所附权利要求书的范围内。此外,描述安全SPI通信的各种不同方面,并且应理解,可以独立地或结合一个或多个其它所描述的方面实现每个所描述的方面。

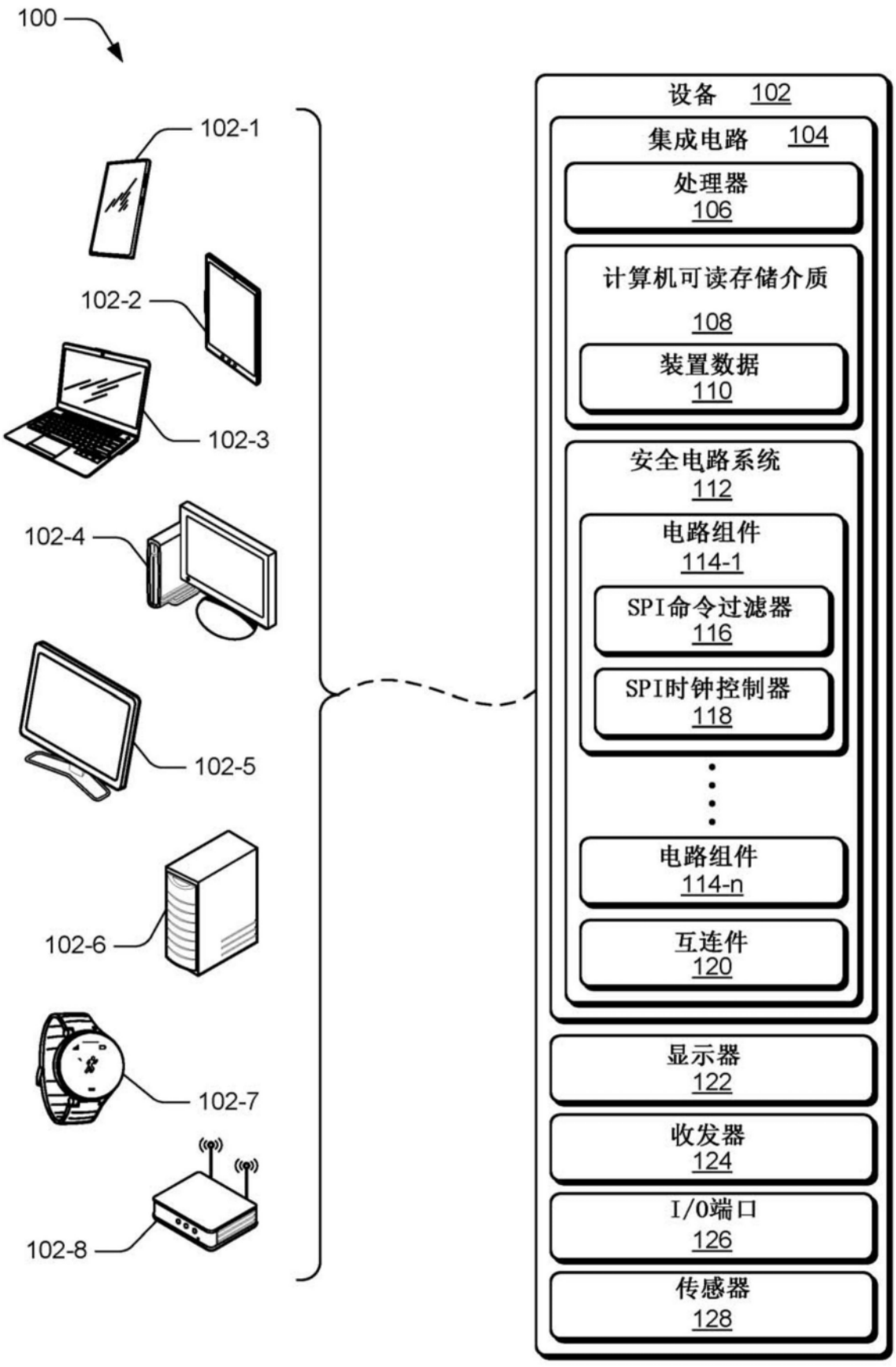

100
102-1
102-2
102-3
102-4
102-5
102-6
102-7
102-8
设备 102
集成电路 104
处理器 106
计算机可读存储介质 108
装置数据 110
安全电路系统 112
电路组件 114-1
SPI命令过滤器 116
SPI时钟控制器 118
电路组件 114-n
互连件 120
显示器 122
收发器 124
I/O端口 126
传感器 128

图1

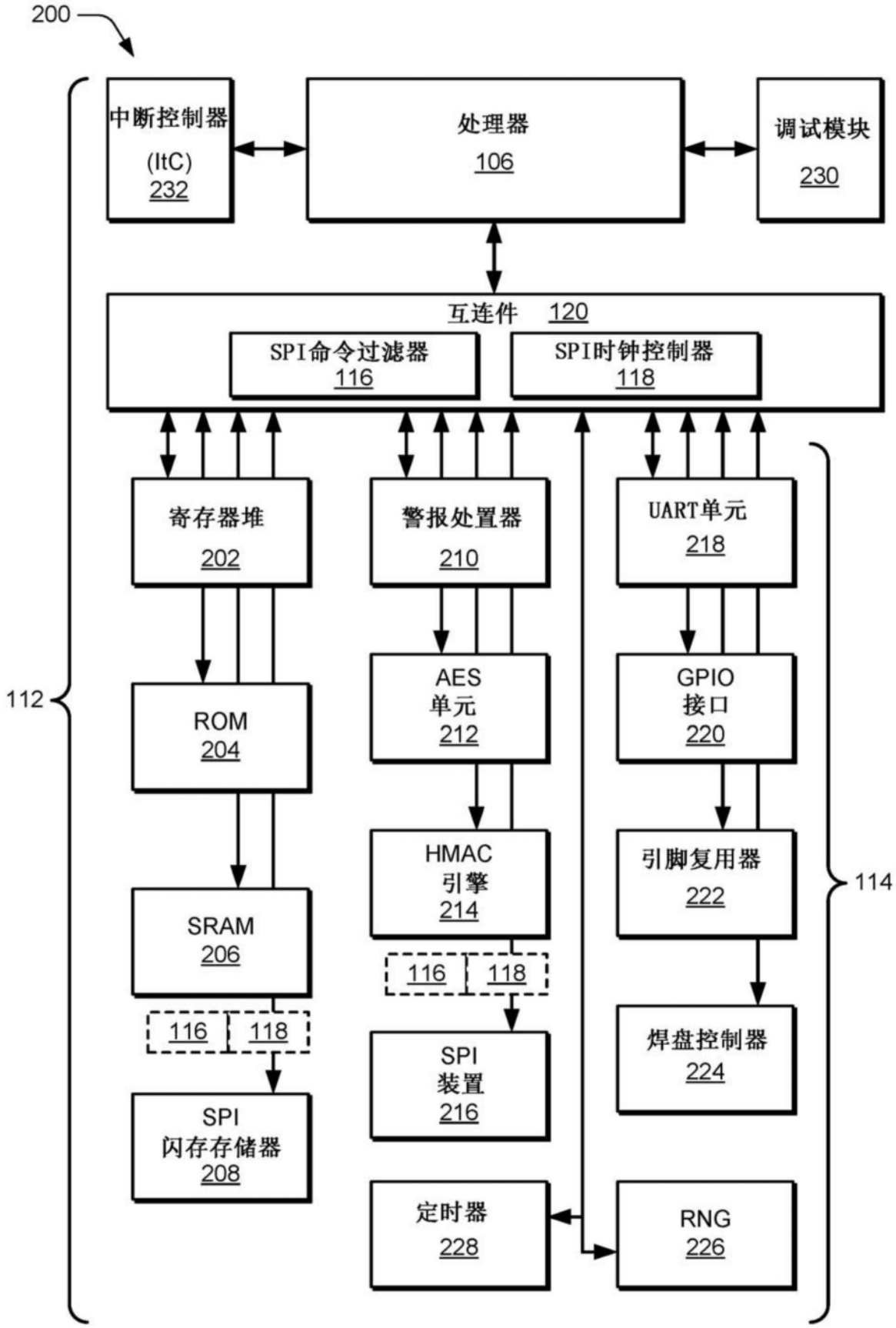

200
中断控制器
(ItC)
232
处理器
106
调试模块
230
互连件 120
SPI命令过滤器
116
SPI时钟控制器
118
寄存器堆
202
警报处置器
210
UART单元
218
ROM
204
AES
单元
212
GPIO
接口
220
SRAM
206
HMAC
引擎
214
引脚复用器
222
116
118
116
118
SPI
装置
216
焊盘控制器
224
SPI
闪存存储器
208
定时器
228
RNG
226
112
114

图2

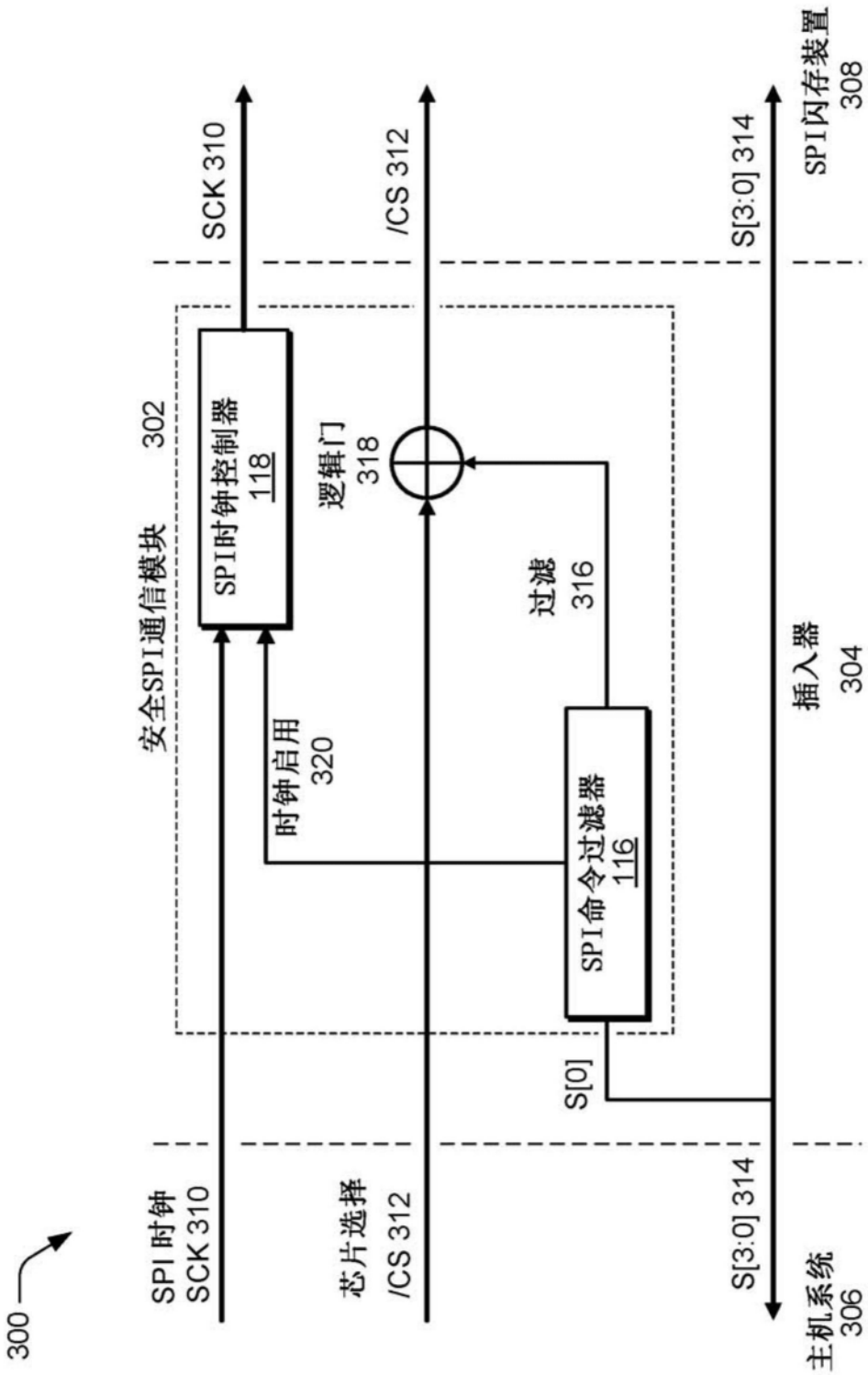

300
SPI 时钟
SCK 310
芯片选择
/CS 312
S[3:0] 314
主机系统
306
S[0]
安全SPI通信模块
302
SPI时钟控制器
118
时钟启用
320
SPI命令过滤器
116
逻辑门
318
过滤
316
SCK 310
/CS 312
S[3:0] 314
插入器
304
SPI闪存装置
308

图3

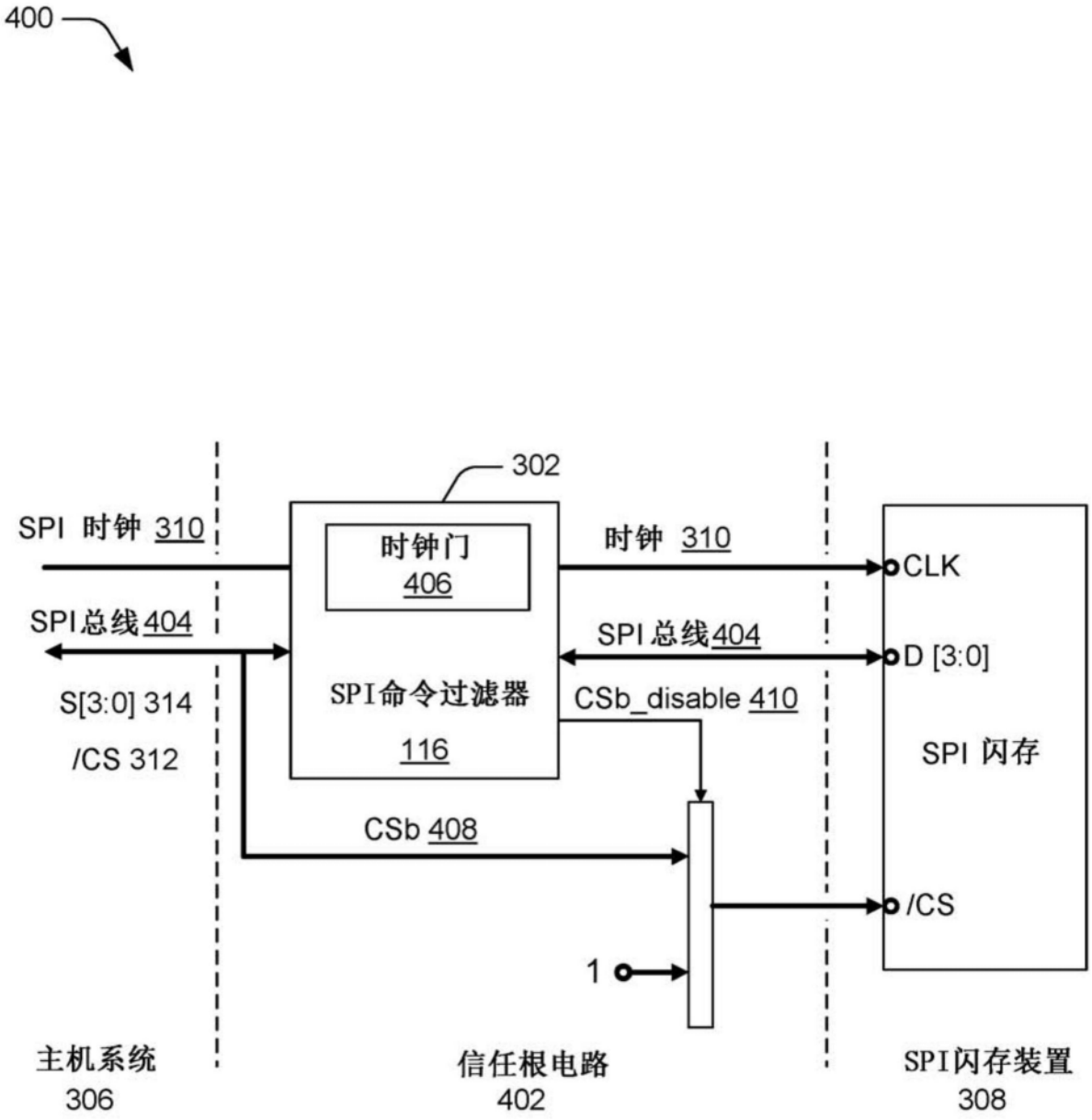

400
302
SPI 时钟 310
时钟门
406
时钟 310
CLK
SPI总线404
SPI 总线404
D [3:0]
S[3:0] 314
/CS 312
SPI命令过滤器
116
CSb_disable 410
SPI 闪存
CSb 408
/CS
1
主机系统
306
信任根电路
402
SPI闪存装置
308

图4

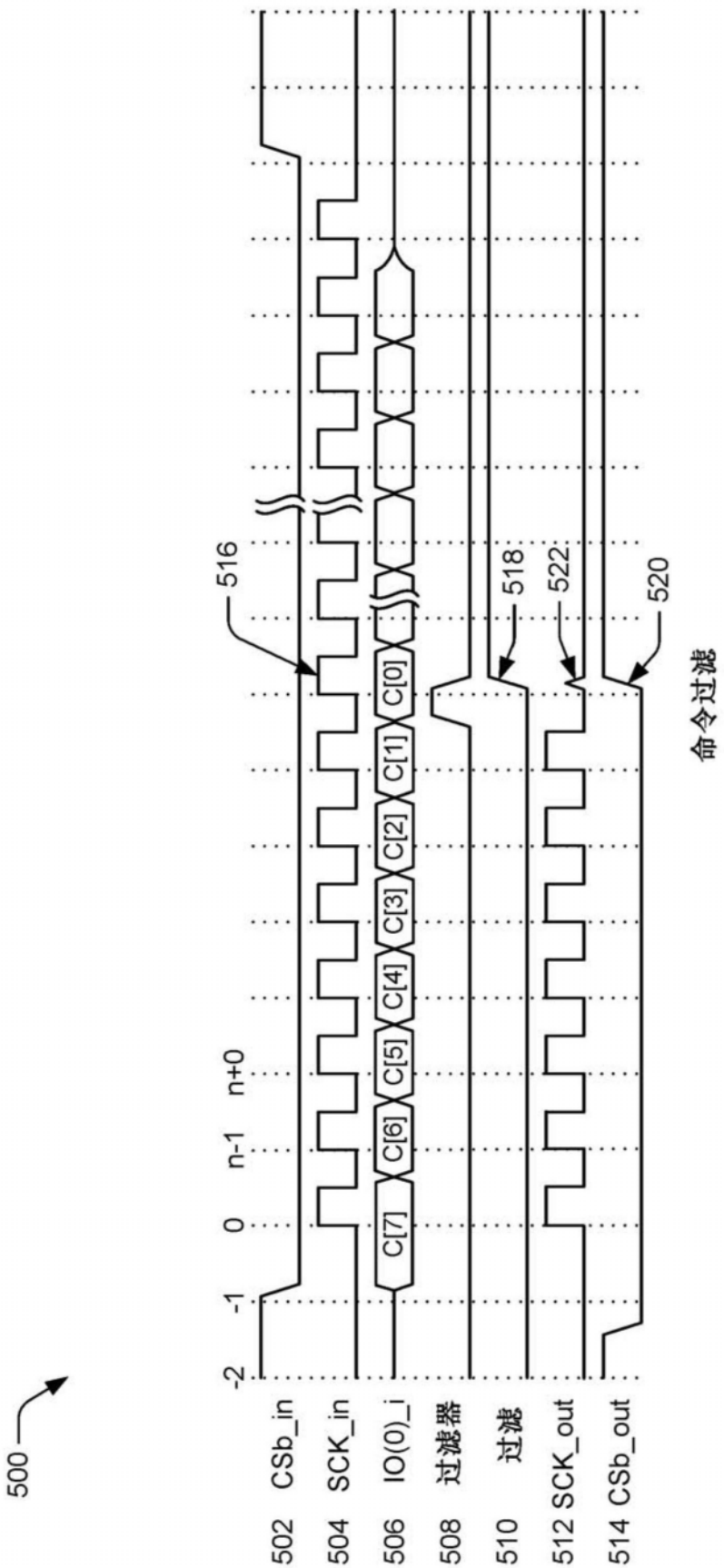

500
502 CSb_in
504 SCK_in
506 IO(0)_i
508 过滤器
510 过滤
512 SCK_out
514 CSb_out
-2
-1
0
n-1
n+0
C[7]
C[6]
C[5]
C[4]
C[3]
C[2]
C[1]
C[0]
516
518
520
522
命令过滤

图5

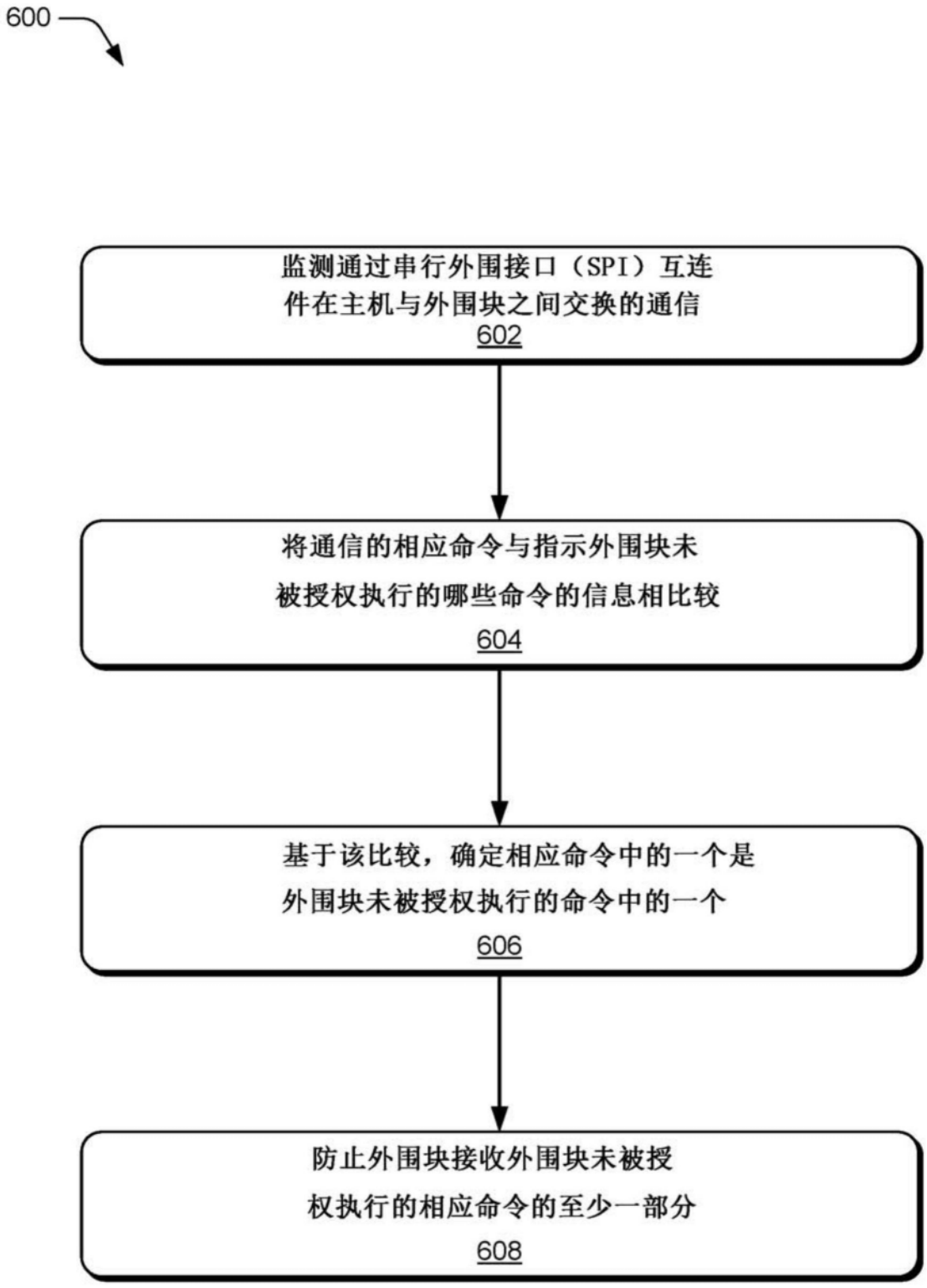

600
监测通过串行外围接口（SPI）互连件在主机与外围块之间交换的通信
602
将通信的相应命令与指示外围块未被授权执行的哪些命令的信息相比较
604
基于该比较，确定相应命令中的一个是外围块未被授权执行的命令中的一个
606
防止外围块接收外围块未被授权执行的相应命令的至少一部分
608

图6

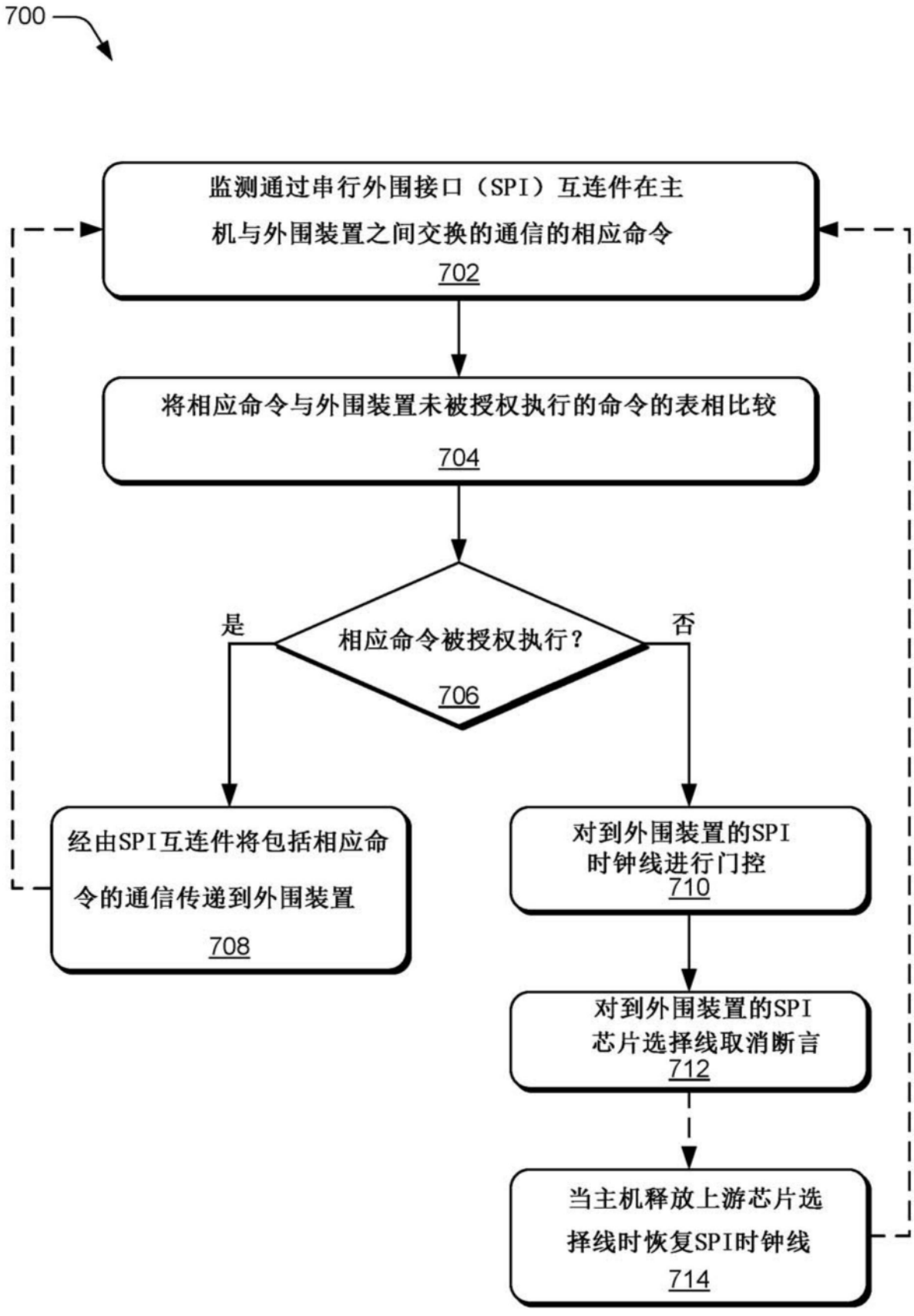

700
监测通过串行外围接口（SPI）互连件在主机与外围装置之间交换的通信的相应命令
702
将相应命令与外围装置未被授权执行的命令的表相比较
704
相应命令被授权执行？
706
是
否
经由SPI互连件将包括相应命令的通信传递到外围装置
708
对到外围装置的SPI时钟线进行门控
710
对到外围装置的SPI芯片选择线取消断言
712
当主机释放上游芯片选择线时恢复SPI时钟线
714

图7

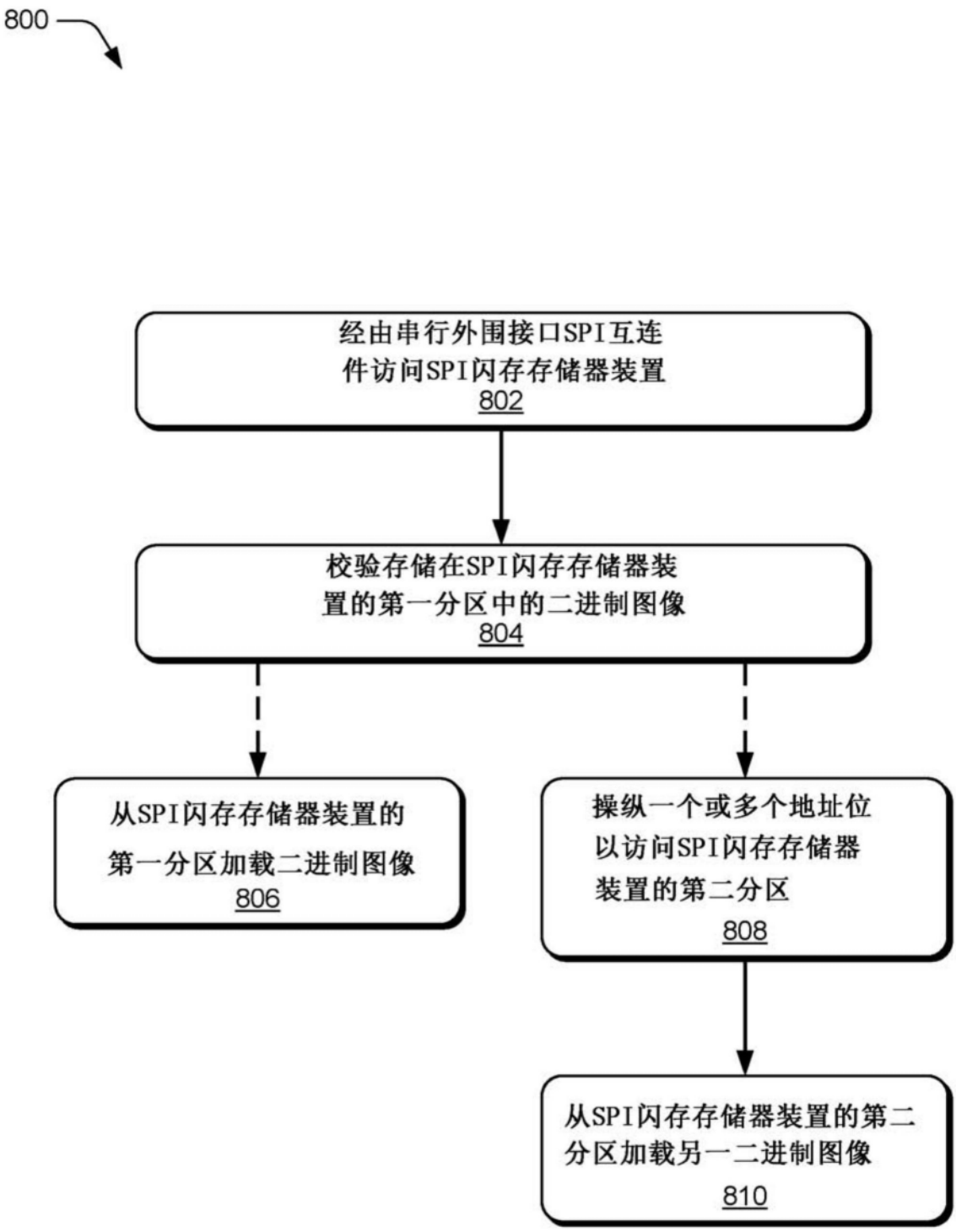

800
经由串行外围接口SPI互连件访问SPI闪存存储器装置
802
校验存储在SPI闪存存储器装置的第一分区中的二进制图像
804
从SPI闪存存储器装置的第一分区加载二进制图像
806
操纵一个或多个地址位以访问SPI闪存存储器装置的第二分区
808
从SPI闪存存储器装置的第二分区加载另一二进制图像
810

图8

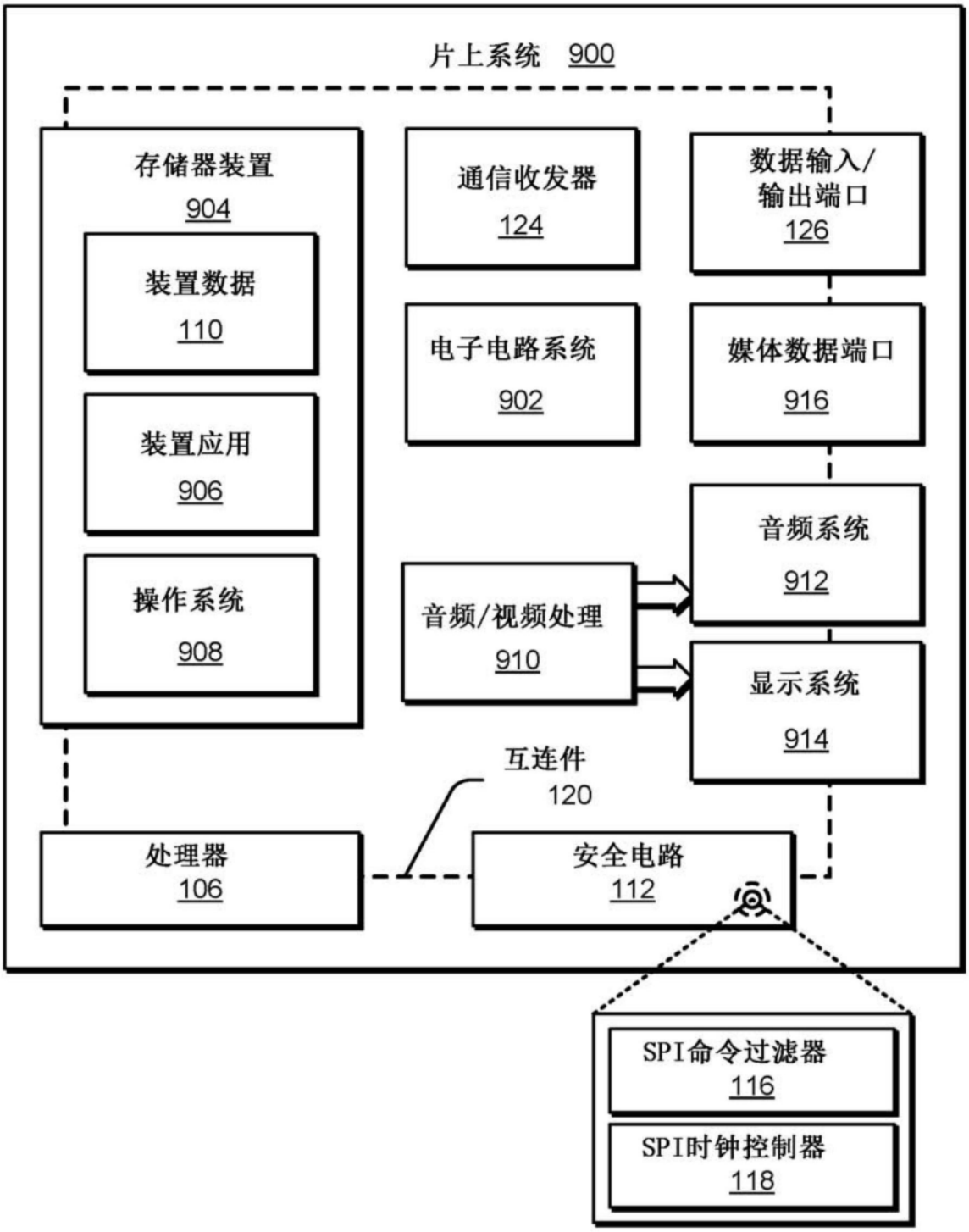

片上系统 900
存储器装置
904
装置数据
110
装置应用
906
操作系统
908
通信收发器
124
电子电路系统
902
数据输入/
输出端口
126
媒体数据端口
916
音频系统
912
音频/视频处理
910
显示系统
914
互连件
120
处理器
106
安全电路
112
SPI命令过滤器
116
SPI时钟控制器
118

图9