(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 107656880 B
(45)授权公告日 2020.12.15

(21)申请号 201710873051.9
(22)申请日 2017.09.25
(65)同一申请的已公布的文献号
申请公布号 CN 107656880 A
(43)申请公布日 2018.02.02
(30)优先权数据
15/337,169 2016.10.28 US
15/337,140 2016.10.28 US
15/590,883 2017.05.09 US
(73)专利权人 上海兆芯集成电路有限公司
地址 201203 上海市浦东新区张江高科技园区金科路2537号301室
(72)发明人 G·葛兰·亨利 罗德尼·E·虎克 泰瑞·派克斯 道格拉斯·R·瑞德
(74)专利代理机构 北京林达刘知识产权代理事务所(普通合伙) 11277
代理人 刘新宇
(51)Int.Cl.
G06F 12/0855(2016.01)
G06F 12/0862(2016.01)
G06F 9/38(2006.01)
(56)对比文件
CN 103176752 A,2013.06.26
审查员 彭玉芝

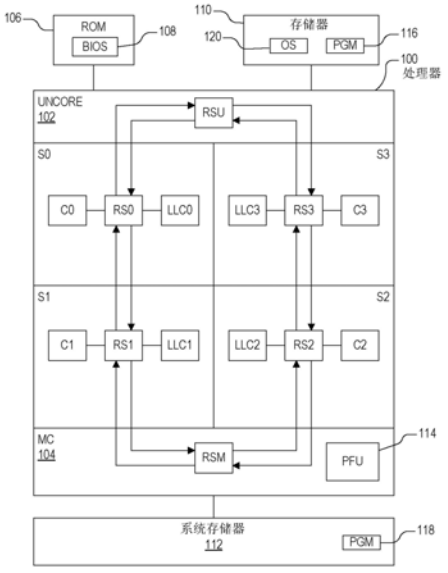
权利要求书3页 说明书13页 附图6页

(54)发明名称

具有包括动态可编程的功能单元的存储器控制器的处理器

(57)摘要

一种具有包括动态可编程的功能单元的存储器控制器的处理器，该处理器包括存储器控制器，其中该存储器控制器用于使外部存储器和可编程功能单元即PFU接合。利用PFU程序来对PFU进行编程以修改存储器控制器的操作，其中该PFU包括可编程逻辑元件和可编程互连器。例如，利用PFU程序对PFU进行编程，以在处理器的操作期间添加功能或以其它方式修改存储器控制器的现有功能，从而增强该存储器控制器的功能。这样，一旦制造了处理器，存储器控制器的功能和/或操作不是固定的，而是作为代替，可以在制造之后修改存储器控制器，以诸如在执行相应进程时提高处理器的效率和/或增强处理器的性能。



CN 107656880 B

1.一种处理器,包括:

存储器控制器,用于接合外部存储器;

可编程功能单元即PFU,其由PFU程序进行编程以修改所述存储器控制器的操作,其中所述PFU包括多个可编程逻辑元件和多个可编程互连器;以及

可编程存储器,用于接收所述PFU程序以对所选择的所述可编程逻辑元件进行编程;

其中,所述可编程逻辑元件包括:

查找表,由所述可编程存储器中的相应查找表值位进行编程,并由指令的操作数提供多个输入值以选择相应的查找表值位作为输出;

寄存器,包括输出以及耦接于所述查找表的输出的输入;

第一复用器,包括输出、由所述可编程存储器中所存储的相应存储器位所控制的选择输入、耦接于所述查找表的输出的第一输入以及耦接于所述寄存器的输出的第二输入;

第二复用器,包括输出、由所述可编程存储器中所存储的相应存储器位所控制的选择输入、耦接于所述可编程存储器中所存储的可编程位的第一输入以及耦接于所述可编程互连器的第二输入;

加法器,包括输出、耦接于所述第一复用器的输出的第一输入以及耦接于所述第二复用器的输出的第二输入;以及

第三复用器,包括由所述可编程存储器中所存储的相应存储器位所控制的选择输入、耦接于所述第一复用器的输出的第一输入、耦接于所述加法器的输出的第二输入以及耦接于所述可编程互连器的输出。

2.根据权利要求1所述的处理器,其中,还包括PFU编程器,所述PFU编程器用于使用本地存储器中所存储的PFU程序来对所述PFU进行编程。

3.根据权利要求2所述的处理器,其中,所述处理器对程序命令作出响应,其中所述程序命令用于使所述PFU编程器利用所述本地存储器中所存储的多个PFU程序中的指定的PFU程序来对所述PFU进行编程。

4.根据权利要求1所述的处理器,其中,还包括配置映射,所述配置映射用于将多个不同处理模式中的各处理模式与本地存储器中所存储的多个PFU程序中的相应PFU程序进行映射。

5.根据权利要求1所述的处理器,其中,所述多个可编程逻辑元件和所述多个可编程互连器被细分为大致相同的多个可编程区段,其中所述处理器还包括PFU编程器,所述PFU编程器用于分配多个所述可编程区段,并利用所述PFU程序来对所分配的多个所述可编程区段进行编程,以对所述PFU进行编程。

6.根据权利要求1所述的处理器,其中,所述PFU包括所述可编程存储器,以及所述PFU程序包括被扫描到所述PFU的所述可编程存储器中的位流。

7.根据权利要求1所述的处理器,其中,利用多个PFU程序来对所述PFU进行编程,其中所述处理器还包括PFU编程器,所述PFU编程器用于在所述处理器的操作期间,一次启用所述多个PFU程序至少之一。

8.根据权利要求1所述的处理器,其中,所述PFU程序对所述PFU进行编程,以进行用于对所述外部存储器中所存储的数据进行加密的加密功能。

9.根据权利要求8所述的处理器,其中,所述加密功能包括加密处理和反向加密处理,

所述反向加密处理采用与地址相组合的预定密钥以开发进一步与数据值组合的填充值。

10.一种用于提供处理器的可编程存储器控制器的方法,所述可编程存储器控制器使所述处理器与外部存储器接合,所述方法包括以下步骤:

包含可编程功能单元即PFU,所述PFU包括多个可编程逻辑元件和多个可编程互连器;

利用PFU程序来对所述PFU进行编程,以修改所述可编程存储器控制器的操作;以及

设置可编程存储器,以接收所述PFU程序以对所选择的所述可编程逻辑元件进行编程,包括:

设置查找表、寄存器、第一复用器、第二复用器、加法器和第三复用器;

所述查找表由所述可编程存储器中的相应查找表值位进行编程,并由指令的操作数提供多个输入值以选择相应的查找表值位作为输出;

所述寄存器接收所述查找表的输出;

所述第一复用器接收由所述可编程存储器中所存储的相应存储器位所控制的选择输入,并且接收所述查找表的输出作为第一输入,接收所述寄存器的输出作为第二输入;

所述第二复用器接收由所述可编程存储器中所存储的相应存储器位所控制的选择输入,并且接收所述可编程存储器中所存储的可编程位作为第一输入,接收所述可编程互连器的输出作为第二输入;

所述加法器接收所述第一复用器的输出作为第一输入,接收所述第二复用器的输出作为第二输入;以及

所述第三复用器接收由所述可编程存储器中所存储的相应存储器位所控制的选择输入,并且接收所述第一复用器的输出作为第一输入,接收所述加法器的输出作为第二输入,以及将所述第三复用器的输出提供给所述可编程互连器。

11.根据权利要求10所述的方法,其中,还包括以下步骤:在所述PFU内设置PFU编程器和PFU引擎,其中,在所述PFU中,所述PFU编程器利用本地存储器中所存储的所述PFU程序来对所述PFU引擎进行编程。

12.根据权利要求11所述的方法,其中,还包括以下步骤:利用所述处理器执行程序命令,其中所述程序命令用于命令所述PFU编程器利用所述本地存储器中所存储的PFU程序来对所述PFU引擎进行编程。

13.根据权利要求10所述的方法,其中,还包括以下步骤:在所述PFU中设置配置映射,其中所述配置映射用于将多个不同处理模式中的各处理模式与本地存储器中所存储的多个PFU程序中的相应PFU程序进行映射。

14.根据权利要求10所述的方法,其中,还包括以下步骤:

将所述多个可编程逻辑元件和所述多个可编程互连器细分为大致相同的多个可编程区段;

分配多个所述可编程区段,以根据所述PFU程序来配置所述PFU;以及

利用至少一个PFU程序来对所分配的多个所述可编程区段进行编程。

15.根据权利要求11所述的方法,其中,还包括以下步骤:

将所述PFU设置为所述可编程存储器;以及

对所述PFU进行编程包括:将至少一个所述PFU程序作为位流扫描到所述PFU引擎的所述可编程存储器中。

16.根据权利要求10所述的方法,其中,还包括以下步骤:利用多个PFU程序来对所述PFU进行编程;以及在所述处理器的操作期间,一次启用所述多个PFU程序至少之一。

具有包括动态可编程的功能单元的存储器控制器的处理器

技术领域

[0001] 本发明通常涉及处理器的可编程资源，并且更特别地涉及在存储器控制器级别具有动态可编程的功能单元的处理器。

背景技术

[0002] 处理器持续变得更强大，其中这些处理器在更高的效率等级具有更高的性能。如这里所使用的术语“处理器”是指包括微处理器、中央处理单元(CPU)、一个或多个处理核、微控制器等的任意类型的处理单元。如这里所使用的术语“处理器”还包括诸如集成在芯片或集成电路(IC)上的处理单元等的任意类型的处理器配置，其中这些芯片或集成电路包括片上系统(SOC)内所包含的芯片或集成电路等。半导体制造技术正持续改善，从而使速度提高、功耗降低并且使处理芯片上所集成的电路的尺寸减小。集成尺寸的减小允许在处理单元内并入附加功能。然而，一旦制造了传统的处理器，其内部功能和操作中的许多内部功能和操作基本上是固定的。

[0003] 存储器控制器提供处理器和通常被配置为动态随机存取存储器(DRAM)的外部系统存储器之间的接口。尽管存储器控制器可以是单独设置的，但在许多现代的常规处理配置中，存储器控制器可以集成到与具有向外部系统存储器的输入/输出(I/O)接口的处理器相同的芯片或IC上。在传统配置中，一旦制造了处理器，存储器控制器的功能基本上是固定的。

发明内容

[0004] 根据一个实施例的一种处理器，其包括存储器控制器，其中该存储器控制器用于使外部存储器和可编程功能单元(PFU)接合。利用PFU程序来对PFU进行编程以修改存储器控制器的操作，其中该PFU包括可编程逻辑元件和可编程互连器。例如，利用PFU程序对PFU进行编程，以在处理器的操作期间添加功能或以其它方式修改存储器控制器的现有功能，从而增强该存储器控制器的功能。这样，一旦制造了处理器，存储器控制器的功能和/或操作不是固定的，而是作为代替，可以在制造之后修改存储器控制器，以诸如在执行相应进程时提高处理器的效率和/或增强处理器的性能。

[0005] 该处理器包括用于存储PFU程序的本地存储器。该本地存储器可以是用于存储从外部存储器检索到的PFU程序的随机存取存储器(RAM)。该处理器可以对写入命令作出响应，其中该写入命令用于命令处理器将PFU程序从外部存储器写入随机存取存储器。该处理器还可以包括PFU编程器，其中该PFU编程器用于使用PFU存储器中所存储的PFU程序来对PFU进行编程。该PFU存储器可以是或可以包括只读存储器(ROM)，其中该只读存储器用于存储用于对PFU进行编程以根据预先确定的PFU定义进行工作的至少一个预先确定的PFU程序。例如，PFU程序可以是默认PFU程序，其中在处理器的启动时，PFU编程器使用该默认PFU来对PFU进行编程。作为代替或另外，处理器可以对程序命令作出响应，其中该程序命令用于使PFU编程器利用PFU存储器中所存储的多个PFU程序中的所指定的PFU程序来对PFU进行

编程。可以包括配置映射，其中该配置映射用于将多个不同处理模式中的各处理模式与PFU存储器中所存储的多个PFU程序中的相应PFU程序进行映射。

[0006] 可编程逻辑元件和可编程互连器可被细分为大致相同的多个可编程区段。可以包括PFU编程器，其中该PFU编程器用于分配多个可编程区段，并利用PFU程序来对所分配的多个可编程区段进行编程，以对PFU进行编程。

[0007] 可编程逻辑元件可以包括可编程查找表。另外或作为替代，可编程逻辑元件可以包括加法器、复用器和寄存器。PFU可以包括可编程存储器，其中在该可编程存储器中，PFU程序可以是被扫描到PFU的可编程存储器中的位流。可以利用多个PFU程序来对PFU进行编程，并且可以包括PFU编程器，其中该PFU编程器用于在处理器的操作期间，一次启用这些PFU程序至少之一。

[0008] 作为更具体的非限制性示例，PFU程序可以对PFU进行编程，以进行用于对外部存储器中所存储的数据进行加密的加密功能。加密功能可以包括加密功能和反向加密功能，其中该反向加密功能采用与地址相组合的预定密钥，以开发进一步与数据值组合的填充值。

[0009] 一种用于提供处理器的可编程存储器控制器的方法，所述可编程存储器控制器使所述处理器与外部存储器接合，所述方法包括以下步骤：将包括可编程逻辑元件和可编程互连器的可编程功能单元(PFU)并入所述存储器控制器；以及利用PFU程序来对所述PFU进行编程，以修改所述存储器控制器的操作。

[0010] 所述方法可以包括将所述PFU程序存储在所述处理器的本地存储器中。所述方法还可以包括利用所述处理器执行写入命令，其中所述写入命令用于命令所述处理器将所述PFU程序从所述外部存储器写入所述本地存储器的随机存取存储器。所述方法可以包括在所述PFU内设置PFU编程器和PFU引擎，其中，所述PFU编程器利用所述本地存储器中所存储的所述PFU程序来对所述PFU引擎进行编程。所述方法可以包括利用所述处理器执行程序命令，其中所述程序命令用于命令PFU编程器利用PFU存储器中所存储的PFU程序来对PFU引擎进行编程。所述方法可以包括在所述PFU中设置配置映射，其中所述配置映射用于将多个不同处理模式中的各处理模式与PFU存储器中所存储的多个PFU程序中的相应PFU程序进行映射。

[0011] 所述方法可以包括：将所述可编程逻辑元件和所述可编程互连器细分为大致相同的多个可编程区段；分配多个所述可编程区段，以根据所述PFU程序来配置所述PFU；以及利用至少一个PFU程序来对所分配的多个所述可编程区段进行编程。所述方法可以包括：将所述PFU设置为可编程存储器；以及将所述至少一个PFU程序作为位流扫描到PFU引擎的可编程存储器中。所述方法可以包括：利用多个PFU程序来对所述PFU进行编程；以及在所述处理器的操作期间，一次启用所述多个PFU程序至少之一。

附图说明

[0012] 将针对以下的说明和附图来更好地理解本发明的益处、特征和优点，其中：

[0013] 图1是根据本发明的一个实施例所实现的包括可编程功能单元(PFU)的处理器耦接至外部存储器和内存装置的简化框图；

[0014] 图2是根据本发明的一个实施例所实现的图1的PFU的更详细框图；

[0015] 图3是根据本发明的一个实施例的使用可编程逻辑所实现的、图2中的PFU编程器和控制器与PFU引擎接合的简化框图；

[0016] 图4是示出根据本发明的一个实施例的用于对图1的PFU进行初始编程的方法的框图；

[0017] 图5是描述根据本发明的一个实施例的可用于对图1的PFU进行编程或以其它方式进行重新编程的可执行二进制应用的简化框图；

[0018] 图6是根据本发明的一个实施例所实现的图3的可编程逻辑的更详细框图；

[0019] 图7是根据本发明的一个实施例所实现的图6的可编程逻辑元件的示意框图；

[0020] 图8是根据本发明的一个实施例所实现的图7的LUT的示意图；

[0021] 图9是根据本发明的一个实施例的用于对图2的PFU引擎进行编程的PFU程序的格式的简化框图；

[0022] 图10是示出根据本发明的一个实施例的用于生成对图2的PFU引擎进行编程所用的图1的PFU程序的示例性方法的简化框图；

[0023] 图11是示出在向图1的系统存储器存储数据时可被编程到PFU中并且由MC进行的示例性加密处理的简化框图；以及

[0024] 图12是示出在从图1的系统存储器加载数据时可被编程到PFU中并且由MC进行的反向加密处理的简化框图。

具体实施方式

[0025] 本发明人已意识到与存在于传统处理器中的预定存储器控制器相关联的可能限制。因此，本发明人研发了具有包含可编程功能单元(PFU)的存储器控制器的处理器，其中该可编程功能单元(PFU)是可配置的或以其它方式可编程的，以修改或以其它方式增强存储器控制器的操作。基本输入/输出系统(BIOS)或操作系统(OS)可以包括用于对PFU进行编程的配置信息。BIOS在上电、复位或重启等(这里称为POR)时、或者OS(在BIOS之后在启动期间被加载的情况下)可以将该配置信息复制到存储器中并且向PFU发送命令以访问该配置信息。另外或作为替代，特定软件程序、进程或应用的编程人员或开发人员可以将PFU程序并入用于对PFU进行编程的应用中，以修改或增强该特定应用所使用的存储器控制器的操作。作为示例，PFU可被配置为在相对于处理器所使用的外部系统存储器进行写入或读取时，进行编程后的加密功能。

[0026] 图1是根据本发明的一个实施例所实现的包括可编程功能单元(PFU)114的处理器100耦接至外部存储器和内存装置的简化框图。处理器100的标准指令集架构(ISA)可以是x86架构，其中在x86架构中，可以正确地执行被设计为在x86处理器上执行的大多数应用程序。如果获得了预期的结果，则应用程序被正确执行。特别地，处理器100执行x86指令集的指令并且包括x86用户可见寄存器集。然而，本发明不限于x86架构，使得可以根据如本领域普通技术人员已知的任何可选ISA来实现处理器100。

[0027] 处理器100包括单独标记为S0、S1、S2和S3(S0～S3)的4个片区(slice)，其中应当理解，片区的数量是任意的，并且包括仅一个(1)和多达任意正整数个。各个片区S0～S3包括四个核C0、C1、C2和C3(C0～C3)中的相应核、四个高速缓冲存储器或“末级高速缓存器”LLC0、LLC1、LLC2和LLC3(LLC0～LLC3)中的相应高速缓冲存储器、以及四个环形站R0、R1、R2

和R3(R0～R3)中的相应环形站。各个核C0～C3包括耦接至环形站R0～R3中的相应环形站的一个或多个内部高速缓冲存储器(例如,未示出的一个或多个L1高速缓存器和L2高速缓存器等),其中该相应环形站进一步耦接至末级高速缓存器LLC0～LLC3中的相应高速缓存器。应当理解,处理器100可被配置为单核处理器、中央处理单元(CPU)或微处理器,而不是具有多个核的多个片区。

[0028] 处理器100还包括具有相应环形站RSU的“uncore(非核)”102和具有相应环形站RSM的存储器控制器(MC)104。环形站R0～R3、RSU和RSM以环形配置耦接在一起,以使得能够在分区S0～S3、uncore102和存储器控制器104之间进行通信。如图所示,例如,RS0与RS1进行双向通信,RS1与RSM进行双向通信,RSM与RS2进行双向通信,RS2与RS3进行双向通信,RS3与RSU进行双向通信,RSU与RS0进行双向通信。考虑到环形和双向通信,环形配置中的环形站的特定排序是任意的,其中所示配置仅是许多可能的替代配置其中之一。

[0029] uncore102包含或以其它方式接合处理器100的如下功能,其中这些功能不是位于分区S0～S3中的任意分区或相应核C0～C3中,而是应当紧密地连接至这些核以实现期望的性能水平。在所示配置中,例如,提供uncore102以接合通常包含基本输入/输出系统(BIOS)108的外部只读存储器(ROM)106。BIOS 108是在处理器100的POR时所执行的固件,其中处理器100用于在POR期间进行硬件初始化,以向操作系统(OS)120以及程序或应用提供运行时服务。uncore102还被设置为接合外部存储器110,其中该外部存储器110可以包括诸如一个或多个硬盘驱动器、光盘驱动器、闪速驱动器等的任意数量的数据存储装置,并且通常存储OS 120。

[0030] MC 104使处理器100接合至外部系统存储器112。分区S0～S3共享系统存储器112的资源,并且还可以经由环形站RS0～RS3、RSU、RSM彼此共享信息。可以使用诸如一个或多个动态随机存取存储器(DRAM)芯片等的合适内存装置或芯片来实现系统存储器112。

[0031] MC 104还包括PFU 114,其中该PFU 114可被编程为修改或以其它方式增强MC 104的功能。可以以依赖于配置的详情的多个方式中的任一方式来对PFU 114进行编程。在一种情况下,BIOS 108在对存储器110和系统存储器112进行初始化之后,访问存储器110中所存储的PFU程序(PGM)116,并且将该PFU程序116复制到处理器100上的存储器或者复制到系统存储器112。例如,在复制之后,PFU程序116的副本被示出为系统存储器112中所存储的PFU程序118。在一个实施例中,PFU程序116可以是以加密和/或压缩格式所存储的,其中在将该PFU程序116存储于处理器100上的存储器中或者存储于系统存储器112中时,可以首先对该PFU程序116进行解密和/或解压缩。然而,如这里进一步所述,PFU程序116可以具有包括无需进行解密或压缩的一系列逻辑一(1)和零(0)的位流的形式。然后,BIOS 108向PFU 114发送命令或指令等,以利用复制后的PFU程序118来定位PFU 114自身并对PFU 114自身进行编程。一旦进行了编程,PFU 114能够在处理器100的操作期间修改或增强MC104的操作。

[0032] 在另一情况下,在执行BIOS 108之后,将OS 120加载到处理器100中并且安装在处理器100上,并且在OS安装期间,OS 120通过复制PFU程序116、然后指示PFU 114利用诸如PFU程序118等的PFU程序定位自身并对自身进行编程,来进行实质相同的过程。在又一情况下,程序或应用等进行相似的处理,其中在该处理中,应用包含PFU程序116,并且应用指示PFU 114使用诸如PFU程序118等的复制后的PGM信息来定位自身并对自身进行编程。在另一实施例中,PFU 114包含用于存储PFU程序118的本地存储器(例如,图2的本地存储器206)。

在这种情况下,除PFU程序118存储在PFU 114的本地存储器206中、并且PFU 114从其本地存储器访问PFU程序118以进行编程外,BIOS108、OS 120或应用进行相似的编程处理。

[0033] 图2是根据本发明的一个实施例所实现的PFU 114的更详细框图。设置PFU引擎202,其中利用PFU程序118对PFU引擎202进行编程,以修改和/或者增强MC 104的操作。在PFU 114中可以包括PFU编程器和控制器204,其中该PFU编程器和控制器204用于管理和/或控制PFU引擎202的操作,包括对PFU引擎202进行编程。PFU编程器和控制器204访问用于对PFU引擎202进行编程的所识别的一个或多个PFU程序,并且使得能够将该一个或多个PFU程序中的至少一个程序编程到PFU引擎202中。PFU编程器和控制器204被示出为单独单元,而且可以包含在PFU引擎202自身内。在一个实施例中,PFU 114不包括本地存储器206,其中在这种情况下,可以使用系统存储器112来存储PFU程序118。在不具有本地存储器206的情况下,BIOS 108、OS 120或应用发送识别系统存储器112中的PFU程序118的位置的编程命令,并且PFU编程器和控制器204从系统存储器112访问PFU程序118并对PFU引擎202进行编程。

[0034] 在一个实施例中,PFU引擎202可以配置有利用多个PFU程序要进行编程的充足资源,其中PFU编程器和控制器204将每一个PFU程序编程到PFU引擎202中,并且仅仅激活或者启用与执行中的特定进程或处理器100的特定操作模式相关联的适当PFU程序。作为示例,PFU引擎202最初可以在POR时被编程并且针对处理器100的大多数操作被启用。进程(例如,程序或应用等)可以利用另一PFU程序对PFU引擎202进行编程,以供在该进程处于活动状态并且执行中的情况下使用。PFU编程器和控制器204通过一次激活被编程到PFU引擎202中的PFU程序中的仅一个PFU程序来管理PFU引擎202的操作。在不具有本地存储器的配置中,可以利用有限数量的PFU程序来对PFU引擎202进行编程。

[0035] 应当理解,PFU引擎202可以是可以在任何给定时间加载有限数量的PFU程序的有限资源。PFU引擎202可能不具有利用在处理器100的操作期间在任何给定时间可以激活的总数个PFU程序要进行编程的充足容量。在这种配置中,特别是在系统存储器112内的PFU程序中的一个或多个PFU程序的位置信息可能不再有效或可能不可用的情况下,可能难以对随时间的经过而针对不同的模式具有不同的PFU程序的PFU引擎202的编程进行切换。此外,PFU引擎202可以包括利用仅一个大型PFU程序或两个较小型PFU程序根据其实现而要编程的充足资源。

[0036] 在另一实施例中,PFU 114包含本地存储器206,其中该本地存储器206用于存储对PFU引擎202进行编程所用的至少一个PFU程序。本地存储器206可以包括随机存取存储器(RAM)208,其中在这种情况下,PFU程序116可被复制到RAM 208并且由PFU编程器和控制器204访问,以对PFU引擎202进行编程。在一个实施例中,RAM 208可以具有足以存储被示出为PGMA、PGMB、PGMC等的多个PFU程序的大小。响应于程序命令,PFU编程器和控制器204访问PFU程序中的所识别的PFU程序,以对PFU引擎202进行编程。这样,如果PFU引擎202不具有足以保持可以随时激活的所有PFU程序的资源,则PFU编程器和控制器204响应于命令或响应于模式变化,可以即时从本地存储器206对PFU引擎202进行重新编程。

[0037] 本地存储器206还可以包括只读存储器(ROM)210,其中该ROM 210用于存储被示出为PGM1、PGM2、PGM3等的一个或多个标准或预先确定的PFU程序。在一个实施例中,将这些预先确定的PFU程序其中之一指定为默认PFU程序(例如,PGM1)。在处理器100的初始启动期间,代替从存储器110复制PFU程序116(或者除从存储器110复制PFU程序116外),BIOS 108

或OS 120指示PFU编程器和控制器204利用默认PFU程序(在包括的情况下)来对PFU引擎202进行编程,然后激活PFU引擎202的默认PFU程序。作为替代或另外,BIOS108、OS 120或者任何应用或进程可以识别ROM 210内所存储的预先确定的PFU程序中的任意PFU程序以对PFU引擎202进行编程。

[0038] 为了方便多个PFU程序,可以设置PFU配置映射212,其中该PFU配置映射212将处理器100的特定操作模式与针对该模式所设置的相应PFU程序进行映射。该操作模式可以包括在特定进程采用相应PFU程序的情况下的进程标识信息。如图所示,例如,将多个模式标识为分别与相应的PFU程序PGMA、PGM1、PGM2、PGMB等相关联的M1、M2、M3、M4等。PFU编程器和控制器204在每次将PFU程序编程到PFU引擎202中时,更新PFU配置映射212。根据PFU配置映射212中所设置的映射,PFU编程器和控制器204在任何给定时间识别活动模式(或进程),并且激活被编程到PFU引擎202内的相应PFU程序,或者以其它方式对PFU引擎202进行编程。一旦加载和/或激活了正确的PFU程序,则相应地利用PFU引擎202来修改或增强MC 104的操作。

[0039] 这样,PFU编程器和控制器204可以将各模式(或进程)与相应的PFU程序进行映射,除非被另一模式取代或者直到被另一模式取代为止。响应于各后续编程命令或模式变化,PFU编程器和控制器204从ROM 210或RAM 208利用所识别的预先确定的PFU程序来激活PFU引擎202或以其它方式对PFU引擎202进行编程,然后相应地更新PFU配置映射212。特别地,PFU编程器和控制器204咨询PFU配置映射212,并且判断与相应模式相关联的PFU程序是否已被加载到PFU引擎202内。如果与相应模式相关联的PFU程序已被加载到PFU引擎202内,则PFU编程器和控制器204停用当前的PFU程序(在存在的情况下),并且针对激活中的模式激活PFU引擎202内的下一PFU程序。如果PFU引擎202没有加载适合新模式的PFU程序,则PFU编程器和控制器204访问存储所识别的PFU程序的RAM 208或ROM 210,并且相应地对PFU引擎202进行编程。

[0040] 在一个实施例中,PFU编程器和控制器204识别PFU引擎202在无需覆盖PFU引擎202内当前所加载的任何PFU程序的情况下、是否具有足以对下一PFU程序进行编程的可用空间。如果PFU引擎202具有该可用空间,则将下一PFU程序加载到该可用空间中。然而,如果PFU引擎202不具有足以加载下一PFU程序的可用空间,则PFU编程器和控制器204使用替换策略以覆盖当前驻留在PFU引擎202内的一个或多个PFU程序。该替换策略可以是最近最少使用(LRU)算法等,但还可以考虑到加载中的PFU程序所需的可编程空间的量。例如,如果较小的最近最少使用的PFU程序不会为要加载的下一PFU程序提供充足的空间,则尽管最近使用较大的PFU程序的频率更高,也可以选择并覆盖该较大的PFU程序。在一个实施例中,如果在PFU引擎202内正覆盖的任何PFU程序的副本没有存储在ROM 210或RAM 208内,并且如果RAM 208具有充足的可用存储空间,则在PFU引擎202中覆盖PFU程序之前,PFU编程器和控制器204可以将该PFU程序从PFU引擎202卸载或复制到RAM 208中。

[0041] 尽管RAM 208可以存储数量相当可观的PFU程序,但在RAM 208不够大而无法存储在任何给定时间尝试下载的所有PFU程序的情况下,PFU编程器和控制器204可以采取适当动作。例如,如果进程尝试对未被发现的或不可用的PFU程序进行配置,则PFU编程器和控制器204可以仅仅针对该进程禁用PFU引擎202的操作。可选地,PFU编程器和控制器204可以加载或以其它方式激活诸如默认PFU程序PGM1等的标准PFU程序,只要任何其它PFU程序未被永久覆盖即可。

[0042] 图3是根据本发明的一个实施例的使用可编程逻辑301所实现的、PFU编程器和控制器204与PFU引擎202接合的简化框图。在所示实施例中，可编程逻辑301被细分为一组“P”个大致相同的可编程区段303，分别被示为可编程区段P1、P2、…、PP，其中“P”是正整数。PFU编程器和控制器204将一个或多个PFU程序编程到可编程逻辑301中。特别地，PFU编程器和控制器204分配可编程区段303中的足以对PFU程序进行编程的一个或多个可编程区段303，然后将该PFU程序加载到已分配区段303中以在PFU引擎202内实现相应的PFU功能。PFU编程器和控制器204保持用以识别并定位加载到PFU引擎202中的各PFU程序的指针等，并且基于操作模式或活动进程来激活或停用所加载的PFU程序。

[0043] 可编程逻辑301可以是相对较大的资源，诸如由现场可编程门阵列(FPGA)等实现，以针对多个应用进程中的各应用进程一次对多个PFU程序进行编程。然而，可编程逻辑301是有限的资源，因为其余的未分配区段303可能不足以对要编程的新的PFU程序进行编程。在这种情况下，PFU编程器和控制器204在RAM 208中已不存在副本、并且在RAM 208中存在可用的充足空间的情况下，将现有的PFU程序从可编程逻辑301复制到RAM 208中，然后可以利用新的PFU程序来对已分配区段303进行编程。在进程已完成了操作、使得该进程终止的情况下，或者在模式切换的情况下，在PFU引擎202和/或RAM 208内，针对该进程已被编程的任何PFU程序可被无效并且最终被覆盖。

[0044] 各可编程区段303可以包括足以执行简单的PFU程序的可编程逻辑。如图所示，例如，将第一PFU程序PGMA(相对简单)加载到第一可编程区段P1中以实现第一程序PFUA，并且将第二PFU程序PGMB(较复杂)加载到两个可编程区段P2和P3中以实现第二程序PFUB。另外，可以将甚至更多个复杂的PFU程序加载到多于两个的区段303中。根据PFU程序的相对大小和复杂度以及可编程区段303的总数，可以将任何数量的PFU程序编程到可编程逻辑301中。

[0045] 在一个实施例中，PFU编程器和控制器204进行动态分配，其中PFU编程器和控制器204识别可用于分配的下一区段303，并且在扫描新的PFU程序时，开始编程。如果PFU程序在第一分配区段303已被完全编程之后继续进行使得需要附加区段303来完成编程，则对附加区段进行即时动态分配，直到PFU程序被完全编程到PFU引擎202中为止。在一个替代实施例中，PFU编程器和控制器204首先评价新的PFU程序的大小，并且在编程之前相应地分配适当数量的可编程区段303。在另一替代实施例中，PFU程序可被配置为包括用于表示该PFU程序所需的区段303的数量(或者至少可编程元件的数量和类型)的资源声明(RSRC)903等(图9)。在这种情况下，PFU编程器和控制器204检索资源声明903，预先分配所指示的数量的区段303，然后使用PFU程序来对已分配区段进行编程。

[0046] 一旦针对给定进程将PFU程序编程到可编程逻辑301中、并且相应地更新PFU配置映射212，PFU编程器和控制器204监测或以其它方式被提供模式信息，并且使得相应的PFU程序能够在该模式期间进行工作。

[0047] 图4是示出根据本发明的一个实施例的用于对PFU 114进行初始编程的方法的框图。在POR时，在块302中，BIOS 108进行用于进行硬件初始化以向OS 120以及程序或应用提供运行时服务的初始化处理和例程。初始化例如包括供处理器100使用的存储器110和系统存储器112的初始化。

[0048] 下一组块304、306和308可以由BIOS 108或OS 120根据实现来进行。在下一块304中，判断在设置有PFU 114的ROM 210的情况下、PFU程序116是否位于ROM 210上。例如，该

PFU程序可以作为PGM1(例如,默认PFU程序等)存储在ROM 210(在设置的情况下)上。如果PFU程序116不是位于ROM210上、或者没有设置ROM 210,则操作进入块306,其中在该块306中,在存储器110上访问PFU程序116,并且将该PFU程序116复制到本地存储器206的RAM 208(在设置的情况下)、或者复制到系统存储器112。

[0049] 在块304或306之后,操作进入块308,其中在该块308中,将编程命令PGM<ADDR>发送至MC 104的PFU 114以对PFU引擎202进行编程。该PGM命令可以由PFU编程器和控制器204接收到,其中该PFU编程器和控制器204使用所包括的地址ADDR来定位PFU程序118。在将PFU程序118预先存储在处理器100内的ROM 210上的实施例中,ADDR标识ROM 210内的位置,例如PGM1(或ROM 210内的任何其它预先存储的PFU程序)的位置等。在没有预先存储PFU程序118、而且在处理器100上设置有本地存储器206的RAM 208的实施例中,可以将PFU程序116复制到RAM 208内的ADDR对所复制的PFU程序的位置进行标识的位置。例如,ADDR可以标识RAM 208上的作为PGMA等所存储的所复制的PFU程序118的位置。在没有设置本地存储器206的情况下,将PFU程序116复制作为系统存储器112中所存储的PFU程序118,并且ADDR标识系统存储器112中的PFU程序118的位置。

[0050] 然后,操作进入块310,其中在该块310中,PFU编程器和控制器204使用所设置的ADDR来访问PFU程序(例如,PFU程序118和/或PGM1和/或PGMA),并且相应地对PFU引擎202进行编程并启用PFU引擎202。然后,初始编程的方法完成。一旦这样对PFU引擎202进行了编程并且启用该已编程的PFU引擎202,该已编程的PFU引擎202根据PFU程序来修改和/或增强MC104的操作。

[0051] 图5是描述根据本发明的一个实施例的、可用于对PFU 114进行编程或以其它方式进行重新编程的可执行二进制应用(APP)502的简化框图。二进制APP 502包括头部504和主体506。二进制APP 502是以通用形式示出的,并且可被实现为可以由处理器100的处理核C0～C3中的任一个或多个处理核成功执行的二进制可执行文件(.EXE)文件、字节码文件(.NET、Java等)或任何其它类型的可执行代码。在所示配置中,头部504包括至少一个PFU写入指令,其中提供各写入指令以指定或定位可用于对PFU 114进行编码的相应PFU程序。如图所示,例如,头部504包括用于标识头部504内所包含的相应PFU程序PGMA_PFU的包含操作数(或参数)PGMA的PFU写入指令WRITE_PFU。可选地,PFU程序PGMA_PFU可以设置在二进制APP 502的不同区段内。在任何情况下,操作数PGMA可以是用于定位二进制APP 502和/或系统存储器112内的PFU程序PGMA_PFU的地址或偏移量。尽管二进制APP 502包括用于标识相应PFU程序的仅一个PFU写入指令,但可执行二进制应用可以包括用于加载可以在任何给定时间加载到处理器100中的任何数量的PFU程序的任何数量的PFU写入指令。

[0052] 在操作期间,处理核(例如,C0)进行从存储器110向系统存储器112访问和/或加载二进制APP 502,并且执行WRITE_PFU指令。假定本地存储器206的RAM 208存在,则使用WRITE_PFU指令的操作数PGMA来定位二进制APP502内的PFU程序PGMA_PFU,并且将PFU程序PGMA_PFU写入RAM 208中。可选地,可以将PFU程序PGMA_PFU写入处理器100的PFU 114可访问的任何其它存储器中。头部121还包括具有位置(或地址)操作数LOC的PFU编程指令PGM_PFU,其中该PFU编程指令PGM_PFU被转发至PFU 114的PFU编程器和控制器204。LOC标识PFU程序PGMA_PFU的RAM 208内的从二进制APP 502所复制的位置。然后,PFU编程器和控制器204利用来自RAM 208的PFU程序PGMA_PFU来对PFU引擎202进行编程。

[0053] 在处理器100内没有设置本地存储器206(或任何其它适当存储器)的配置中，WRITE_PFU指令可以简单地标识二进制APP 502内的PFU程序PGMA_PFU的位置，而无需实际将PFU程序PGMA_PFU复制到处理器100的任何本地存储器中。在这种情况下，利用PFU程序PGMA_PFU在系统存储器112内的地址来更新LOC。将PFU编程指令PGM_PFU转发至PFU 114的PFU编程器和控制器204，其中该PFU编程器和控制器204使用操作数LOC来定位系统存储器112中的PFU程序PGMA_PFU以对PFU引擎202进行编程。

[0054] 在替代配置中，在二进制APP 502中可以使用单个指令或命令，其中该单个指令或命令在执行的情况下，被转发至PFU编程器和控制器204。PFU编程器和控制器204使用所包括的采用地址或偏移量等的形式的操作数来定位PFU程序PGMA_PFU，其中使用该PFU程序PGMA_PFU来对PFU引擎202进行直接编程。在任意的编程配置中，PFU编程器和控制器204启用新编程到PFU引擎202中的PFU程序PGMA_PFU。

[0055] 系统存储器112(和/或其它外部存储器)可以包括被加载以供处理器100随时间经过而执行的多个应用程序。多个应用或进程可以被加载到处理核C1～C3中的任一个或多个处理核中，但在所示实施例中各处理核通常一次仅执行一个进程。各处理核一次执行多个进程的实施例也被考虑。可以将多个应用程序分配给其中一个处理核来执行。OS 120包括用于调度处理器100的应用程序的执行的调度器等，处理器100的应用程序的执行包括针对给定处理核一次一个地换入换出多个进程中的各进程以供执行。多个应用可以由给定处理核来执行，其中各应用可以包括用于对PFU 114进行编程的一个或多个PFU程序。可以使用PFU编程器和控制器204和本地存储器206以及PFU配置映射212来管理与处理器100的不同处理模式相对应的不同进程，以随时间的经过控制PFU引擎202的编程。

[0056] 图6是根据本发明的一个实施例所实现的图3的可编程逻辑301的更详细框图。所示的可编程逻辑301包括可编程元件的阵列，该阵列包括被示出为配置在逻辑元件601的XY矩阵中的可编程逻辑元件(LE)601，这些可编程逻辑元件各自被示出为LExy，其中x和y分别表示阵列的行标和列标。各行还包括杂项逻辑块603的阵列中的至少一个，其中杂项逻辑块603各自包括用以补充逻辑元件601的矩阵的支持逻辑。各杂项逻辑块603可以例如包括一个或多个存储元件、一个或多个寄存器、一个或多个锁存器、一个或多个复用器、一个或多个加法器(用以相加或相减数字值)、一组布尔逻辑元件或门(例如，诸如或(OR)门、与(AND)门、反相器、异或(XOR)门等的逻辑门)等。各杂项逻辑块603可以包括可以被配置为移位寄存器或数据拌和器(swizzler)等以用于灵活的数据操作的一个或多个寄存器。逻辑元件601和杂项逻辑块603与路由网格耦接到一起，其中该路由网格包括可编程交叉开关或互连器605的矩阵。各可编程互连器605包括多个开关以选择性地将可编程装置连接在一起。路由网格包括足以将逻辑元件601和杂项逻辑块603中的多个器件连接在一起以进行简单处理操作和更复杂处理操作的连接性。

[0057] 如本文进一步描述的，各可编程区段303包括一个或多个可编程元件(逻辑元件601、逻辑块603)以及用于选择性地将装置和元件连接在一起以实现PFU 114的用于修改MC 104的操作的相应功能的相应路由网格(互连器605)。路由网格是包括多个开关等以在逻辑元件601和杂项逻辑块603之间进行输入和输出的重定向的切换矩阵。

[0058] 可编程逻辑301包含可编程存储器607，其中使用该可编程存储器607来接收PFU程序(例如，PFU程序116、PFU程序118、PGMA、PGMB、PGMC、…、PGM1、PGM2、PGM3等中的一个或多

个)，以对逻辑元件601、相应杂项逻辑块603和可编程互连器605中的所选择器件进行编程，从而创建用于在被激活或以其它方式启用时修改MC 104的操作的相应PFU功能。可编程存储器607还可以包括存储位置或寄存器等以接收输入操作数或值并且存储PFU程序的输出结果。可编程存储器607分散在可编程逻辑301的可编程区段303之间，并且可以由进行特定PFU操作的所选已分配区段303中的各可编程区段303单独或共同地使用。可编程存储器607可以被配置为可编程逻辑301内或者甚至MC 104内的专用存储器空间，并且无法进行外部访问。存储器607可以以诸如静态随机存取存储器(SRAM)等的任意合适方式来实现。

[0059] 图7是根据本发明的一个实施例所实现的可编程逻辑元件601的示意框图。逻辑元件601包括查找表(LUT)701、三个2输入复用器(MUX)705、706和707、2输入加法器709以及时钟寄存器(或锁存器)711。可编程存储器607的一部分被示出为用于对逻辑元件601、任意所包括的杂项逻辑块603和一个或多个互连器605的一部分进行编程。如以上所说明的，可编程存储器607可以用于提供输入值、存储输出结果、以及/或者存储针对处理操作的多次迭代中的各次迭代所更新的中间值。

[0060] 如图所示，使用被示出为PGM_PFU的PFU程序来对存储器607进行编程。LUT 701被示出为利用存储器607中的相应LUT值(LV)位进行编程的4X1LUT。MUX 705、706和707各自具有由存储器607所存储的相应存储器位(被分别示出为存储器位M1、M2和M3)所控制的选择输入。将LUT 701的被示出为LO的输出提供给MUX 705的一个输入和寄存器711的输入，其中将寄存器711的输出提供给MUX 705的另一输入。将MUX 705的输出提供给MUX 706的一个输入和加法器709的一个输入。将加法器709的输出提供给MUX 706的另一输入，其中将MUX 706的输出提供给可编程互连器605的输入。存储器607包括可编程位V，其中将该可编程位V提供给MUX 707的一个输入，将MUX 707的另一输入耦接至可编程互连器605的输出，并且将MUX 707的输出提供给加法器709的另一输入。将加法器709的输出提供给MUX 706的另一输入。存储器607还可以用于对互连器605和任意杂项逻辑块603的相应部分进行编程。

[0061] 所示的逻辑元件601仅是示例性的，并且替代版本可以根据特定配置被考虑。逻辑元件601可以被配置在位片粒度级以应对数据值的单个位。针对包括多个位的数据值，使用多个位片逻辑元件。例如，针对64位数据值，并行使用64个位片逻辑元件。

[0062] 在操作中，利用LUT 701的LUT数据值(LV)、MUX 705～707的选择输入M1～M3和提供给MUX 707的输入的可编程数据值V来对存储器607进行编程。从指令的操作数，从存储器607，或者从另一编程块来提供四个输入值S0～S3，以选择16个值中被编程到LUT 701中的值，其中在LUT 701的输出处提供所选择的值作为LO。对MUX 705进行编程，以直接提供LUT 701的LO输出或提供被寄存的版本。可以使用被寄存的版本以插入为了PFU操作的定时为目的的延迟。对MUX 706进行编程，以直接提供MUX 705的输出、或者将作为输出所要提供的或者要提供给另一编程块的加法器709的输出提供给互连器605。加法器709将所选择的值与MUX 705的输出相加，其中所选择的值是编程值V或者来自于互连器605的输出(从另一输入或者从另一编程块所提供)。

[0063] 图8是根据本发明的一个实施例所实现的LUT 701的示意图。提供被组织为二进制MUX树的一组2输入MUX，以基于选择输入S3:S0(其中S0是最低有效位)而在16个输入值LV0～LV15之间进行选择。如先前所述，将LV0～LV15编程到存储器607中。将16个输入值LV0～LV15的各相邻对(LV0和LV1、LV2和LV3、…、等等)提供给八个2输入MUX 801的相应输入对，

其中这些2输入MUX 801各自在其选择输入处接收S0。将MUX 801的8个输出的各相邻对提供给四个2输入MUX 803的相应输入对，其中这些2输入MUX 803各自在其选择输入处接收S1。将MUX 803的四个输出的各相邻对提供给两个2输入MUX805的相应输入对，其中这些2输入MUX 805各自在其选择输入处接收S2。将MUX 805的输出对提供给输出MUX 807的输入对，其中输出MUX 807在其选择输入处接收S3并且在其输出处提供LUT输出LO。应该理解，图8所示的配置仅是本领域普通技术人员能够理解的很多合适LUT实现其中之一。

[0064] 图9是根据本发明的一个实施例的用于对PFU引擎202进行编程的PFU程序901的格式的简化框图，其中PFU程序901可以表现PFU程序116、118、PGMA、PGMB、PGMC、…、PGM1、PGM2、PGM3等中的任意的形式。在这种情况下，PFU程序901可以包括资源声明(RSRC)903，其中该RSRC 903用于表示为了实现PFU程序而在可编程逻辑301内所需的资源量。作为示例，资源声明903可以表示为了完成编程所需的可编程区段的数量P。PFU编程器和控制器204可以在对PFU引擎202的编程期间读取资源声明903以分配相应数量的可编程区段303。尽管诸如通过追踪各逻辑元件601、杂项逻辑块603、可编程互连器605和/或可编程存储器607的量等可以使用较大的粒度，但这可能要求PFU编程器和控制器204随时间的经过而追踪可编程逻辑301的各个体元件。

[0065] PFU程序901还可以包括被称为位流的一系列的逻辑一(1)和零(0)。在一个实施例中，例如，响应于处理核所接收到的编程指令，PFU编程器和控制器204将可编程区段303的已分配区段的可编程存储器(包括可编程存储器607和互连器605的相应可编程存储器)排成大的序列化移位寄存器，然后在位流中移位、直到在各个已分配区段中进行了完全加载为止，然后解除可编程存储器的排列并且提供用以定位并标识编程后的PFU的指针。可以使用包括并行编程的可替代编程方法和格式。此外，可以将资源声明设置在PFU编程器和控制器204要进行读取的诸如开始或结束等的任意合适的位置处，以确保合适的编程。

[0066] 图10是示出根据本发明的一个实施例的、用于生成对PFU 114的PFU引擎202进行编程所用的PFU程序116的示例方法的简化框图。诸如编程器等的应用生成器以所选格式来编写用于描述或以其它方式定义用于修改或增强MC 104的存储器控制器操作的PFU功能描述1002。PFU功能描述1002在其它方面可被称为PFU定义。可以以诸如LegUp、(Catapult technology公司的)Catapult、Verilog、HDL(硬件描述语言)、寄存器控制逻辑(RCL)、寄存器传送逻辑(RTL)等的任意合适的硬件编程语言来编写该PFU功能描述1002。将PFU功能描述1002提供给相应的PFU编程工具1004，其中该PFU编程工具1004被配置为将PFU功能描述1002转换为适合对PFU引擎202进行编程以根据PFU功能描述1002进行工作的PFU程序116。作为示例，PFU编程工具1004可以将PFU功能描述1002转换成可用于对PFU引擎202的可编程逻辑301的可编程区段303中的一个或多个可编程区段进行编程的相应位流。

[0067] 一旦生成了PFU程序116，可以将该PFU程序116存储在存储器110上的供BIOS 108或OS 120访问的适当位置处，以根据前面所述的任何方法来对PFU114进行编程。可选地，可以将PFU程序116并入诸如二进制APP 502等的应用中，以在被执行时由该应用进行编程。

[0068] 图11是示出在向系统存储器112存储数据时可被编程到PFU 114中并且由MC 104执行的示例性加密处理的简化框图。移动(MOV)指令1102表示处理器100的任意核为了将寄存器(REG)1103中所存储的数据值DATA(数据)存储至系统存储器112中的指定地址ADDR所执行的任意类型的存储指令。利用KEY(密钥)1104和加密算法1106来对PFU 114的PFU引擎

202进行编程。KEY1104是可以预先确定的并且被存储在PFU程序116内的任意二进制或十六进制值。加密算法1106根据任何标准或定制加密算法，例如数据加密标准(DES)、RSA公钥系统、MD5算法、高级加密标准(AES)、各种散列算法等。

[0069] 在操作中，如由PFU 114进行修改后的MC 104从MOV指令1102中提取地址ADDR并且将该地址ADDR应用于加密算法1106的一个输入。将KEY 1104应用于另一输入，并且加密算法1106在其输出处提供相应的PAD(填充)值1108。换句话说，加密算法1106实质将KEY 1104和ADDR转换成PAD值1108。将来自REG 1103的DATA值应用于诸如异或(XOR)运算1110等的布尔逻辑函数的一个输入，将PAD值1108应用于另一输入，并且XOR运算1110进行所指示的布尔运算(例如，XOR)并在其输出处提供相应的加密数据值XDATA1112。MC 104将加密XDATA值1112而不是原始的DATA值存储于系统存储器112的地址ADDR处。

[0070] 图12是示出在从系统存储器112加载数据时可被编程到PFU 114中并且由MC 104执行的反向加密处理的简化框图。图12的反向加密处理与图11的加密处理互补，其中将这两个处理一起存储在PFU程序116中，以实现用于相对于系统存储器112来存储并加载信息的完整加密处理。另一MOV指令1202表示处理器100的任意核为了从系统存储器112的定址位置将数据值加载或读取到处理器100的诸如REG 1103等的指定寄存器中所执行的任意类型的加载指令。

[0071] 从加载指令1202提取地址ADDR并将该地址ADDR应用于反向加密算法1206(或解密算法)的一个输入，并且将KEY 1104应用于反向加密算法1206的另一输入，其中反向加密算法1206在其输出提供相应的PAD 1208。还将MOV指令1202应用于系统存储器112以检索加密XDATA值1112。将加密XDATA值1112和PAD 1208应用于XOR运算1110的各个输入，其中XOR运算1110输出相应的解密数据值DATA。MC 104将DATA值而不是所检索到的XDATA值1112存储到如利用MOV指令1202所指定的REG 1103中。

[0072] 假定加密算法1106和反向加密算法1206是互补的，则在执行MOV指令1202时所检索到的解密DATA值与在执行MOV指令1202之前在REG 1103中最初存储的原始DATA值相同。这样，PFU 114修改MC 104的操作，以对系统存储器112中所存储的数据进行加密并且对从系统存储器112检索到的数据进行解密。注意，对于诸如AES等的对称密钥加密，加密算法1106和反向加密算法1206相同(即，是相同算法)，使得仅需要一个加密/解密算法。

[0073] 已经给出了前述描述以使本领域普通技术人员能够在特定应用的上下文及其要求中所提供的那样实现和使用本发明。虽然已经参考本发明的某些优选版本相当详细地描述了本发明，但是其它版本和变形是可能的并被预期。对优选实施例的各种修改对于本领域技术人员将是显而易见的，并且本文设定的一般原理可以应用于其它实施例。例如，本文所描述的电路可以以包括逻辑装置或电路等的任何合适的方式来实现。本领域技术人员应当理解，可以容易地使用所公开的概念和具体实施例作为设计或修改用于在不脱离本发明的精神和范围的情况下实现本发明的相同目的的其它结构的基础。因此，本发明并不意图被限制于本文所示出以及所描述的特定实施例，而是符合与本文公开的原理和新颖特征一致的最宽范围。

[0074] 相关申请的交叉引用

[0075] 本申请是以下的美国专利申请的部分延续申请，在此通过引用包含其全部内容以用于所有的目的和用途。

[0076]

代理人案号	序列号	申请日	发明名称
VAS.2787	15/337,140	2016年10月28日	PROCESSOR WITH AN EXPANDABLE INSTRUCTION SET ARCHITECTURE FOR DYNAMICALLY CONFIGURING EXECUTION RESOURCES

[0077] 本申请与以下的美国专利申请有关，在此通过引用包含其全部内容以用于所有的目的和用途。

[0078]

代理人案号	序列号	申请日	发明名称
VAS.2794	15/337,169	2016年10月28日	PROCESSOR WITH PROGRAMMABLE PREFETCHER

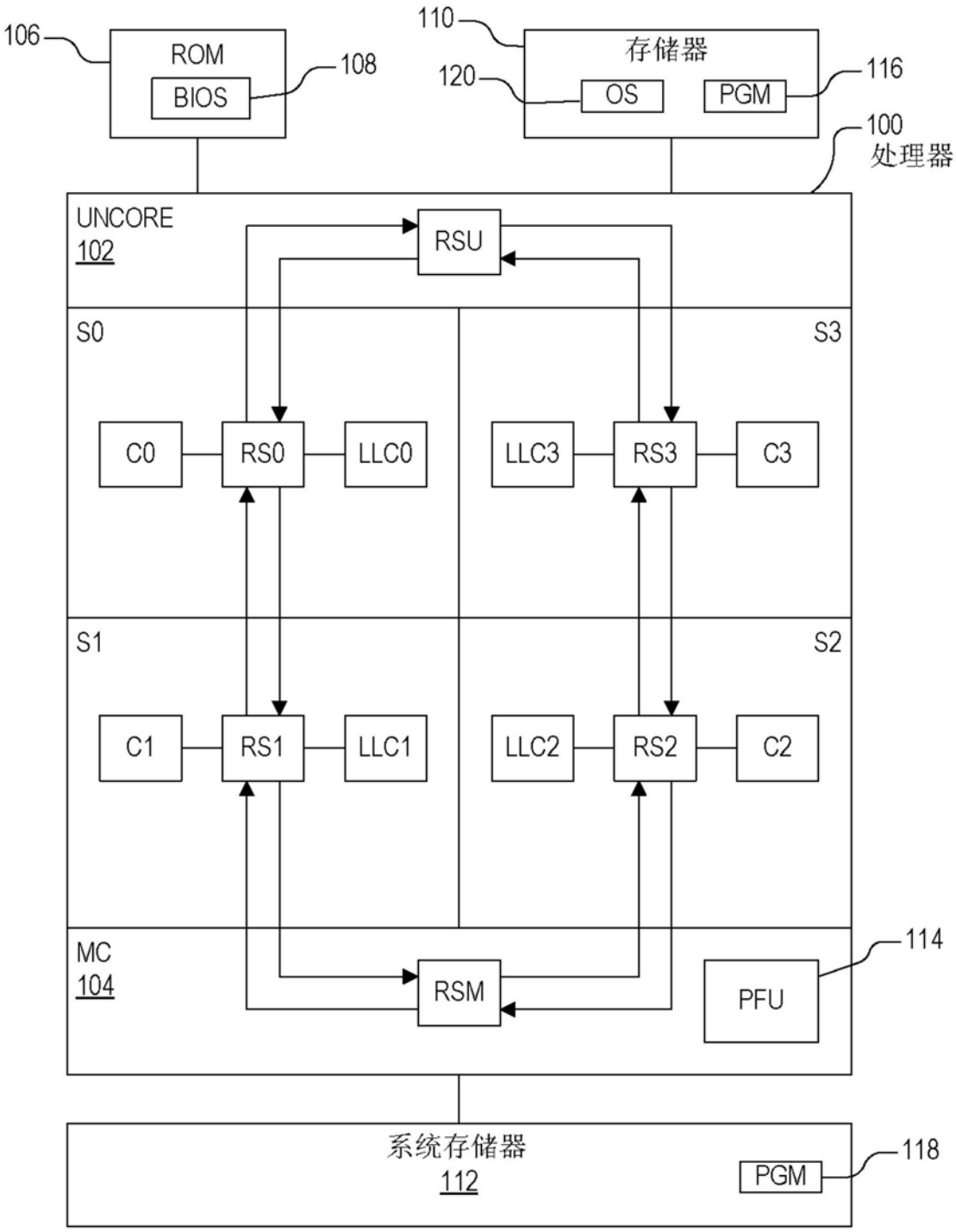

106
ROM
BIOS
108
110
存储器
120
OS
PGM
116
100
处理器
UNCORE
102
RSU
S0
S3
C0
RS0
LLC0
LLC3
RS3
C3
S1
S2
C1
RS1
LLC1
LLC2
RS2
C2
MC
104
RSM
PFU
114
系统存储器
112
PGM
118

图1

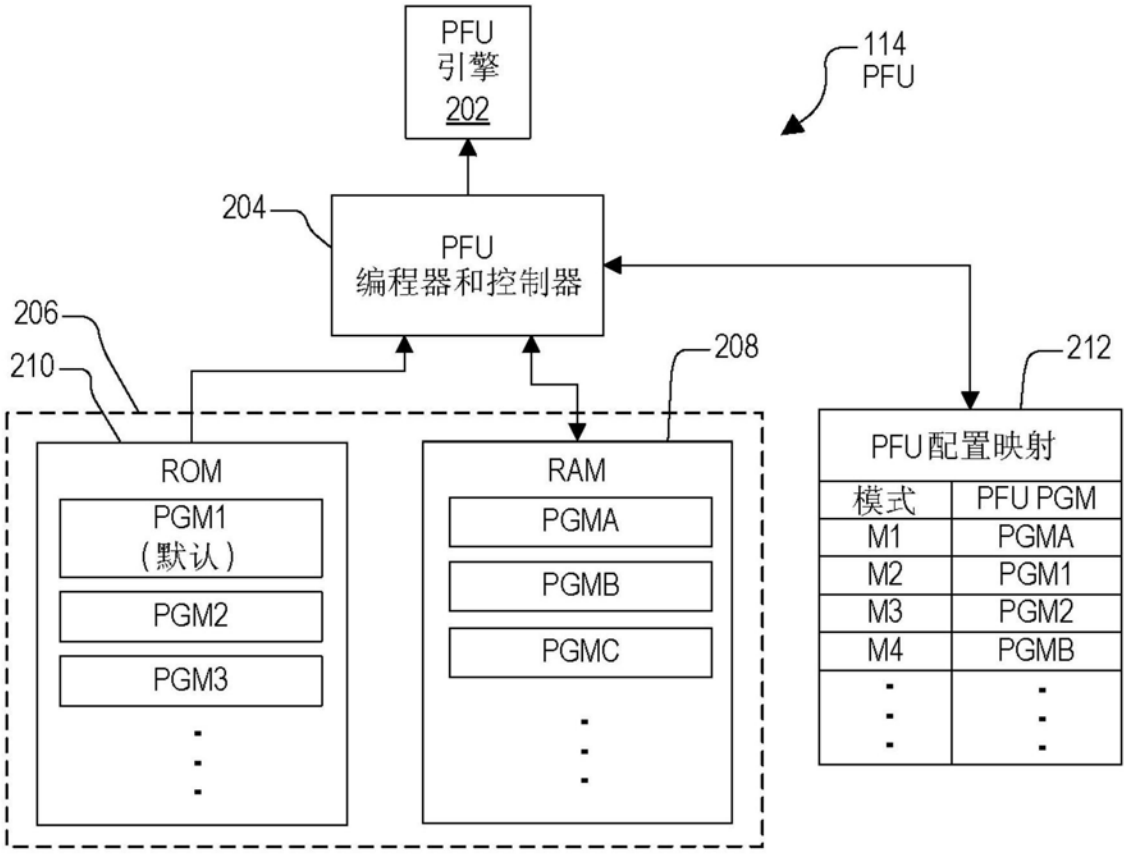

PFU
引擎
202
114
PFU
204
PFU
编程器和控制器
206
210
208
212
ROM
PGM1
（默认）
PGM2
PGM3
RAM
PGMA
PGMB
PGMC
PFU配置映射
模式
PFU PGM
M1
PGMA
M2
PGM1
M3
PGM2
M4
PGMB

图2

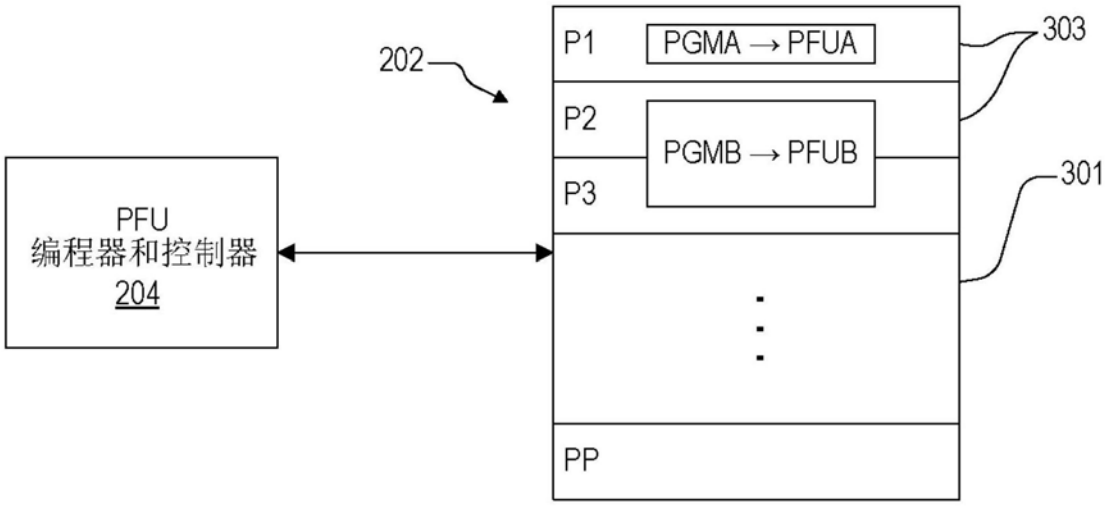

202
P1
PGMA → PFUA
303
P2
PGMB → PFUB
P3
301
PFU
编程器和控制器
204
PP

图3

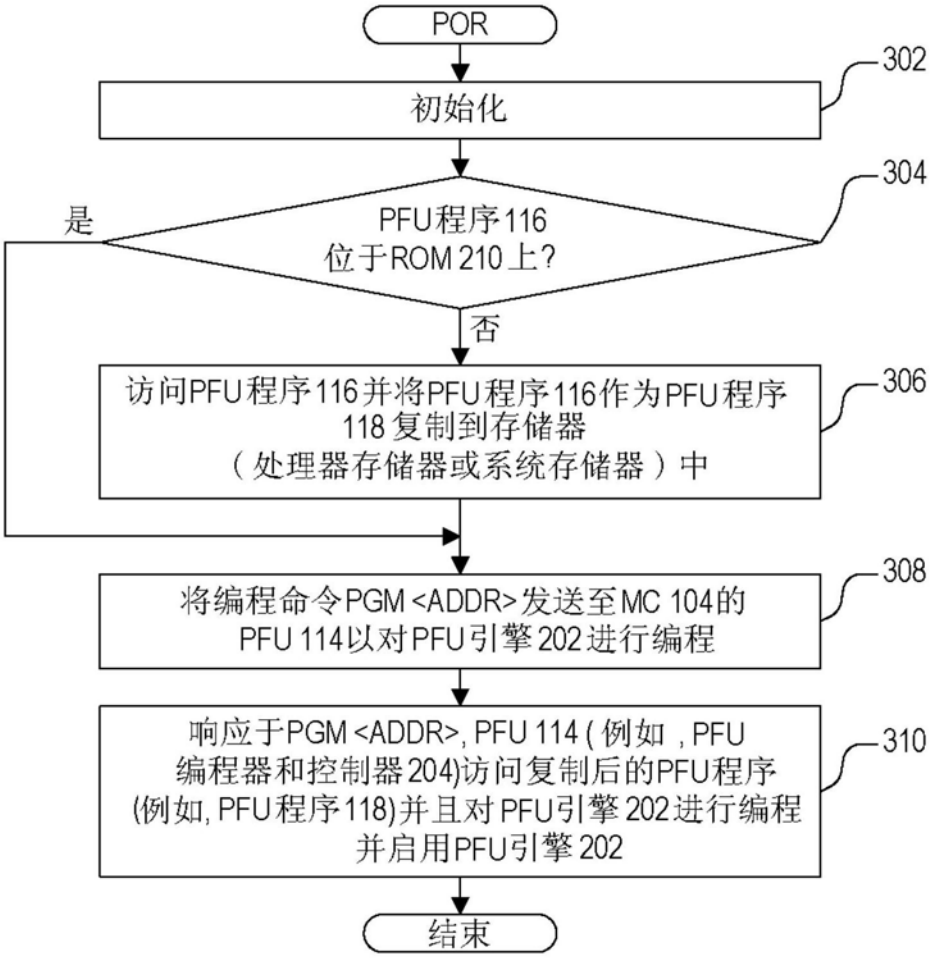

POR
302
初始化
304
是
PFU程序116
位于ROM 210上?
否
306
访问PFU程序116并将PFU程序116作为PFU程序
118复制到存储器
（处理器存储器或系统存储器）中
308
将编程命令PGM <ADDR>发送至MC 104的
PFU 114以对PFU引擎202进行编程
310
响应于PGM <ADDR>, PFU 114 (例如，PFU
编程器和控制器204)访问复制后的PFU程序
(例如, PFU程序118)并且对PFU引擎202进行编程
并启用PFU引擎202
结束

图4

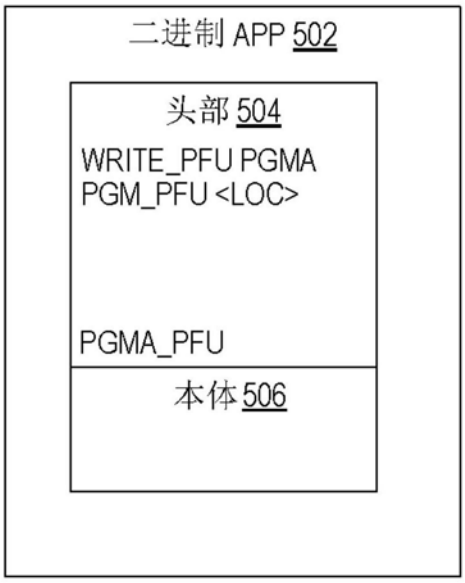

二进制 APP 502
头部 504
WRITE_PFU PGMA
PGM_PFU <LOC>
PGMA_PFU
本体 506

图5

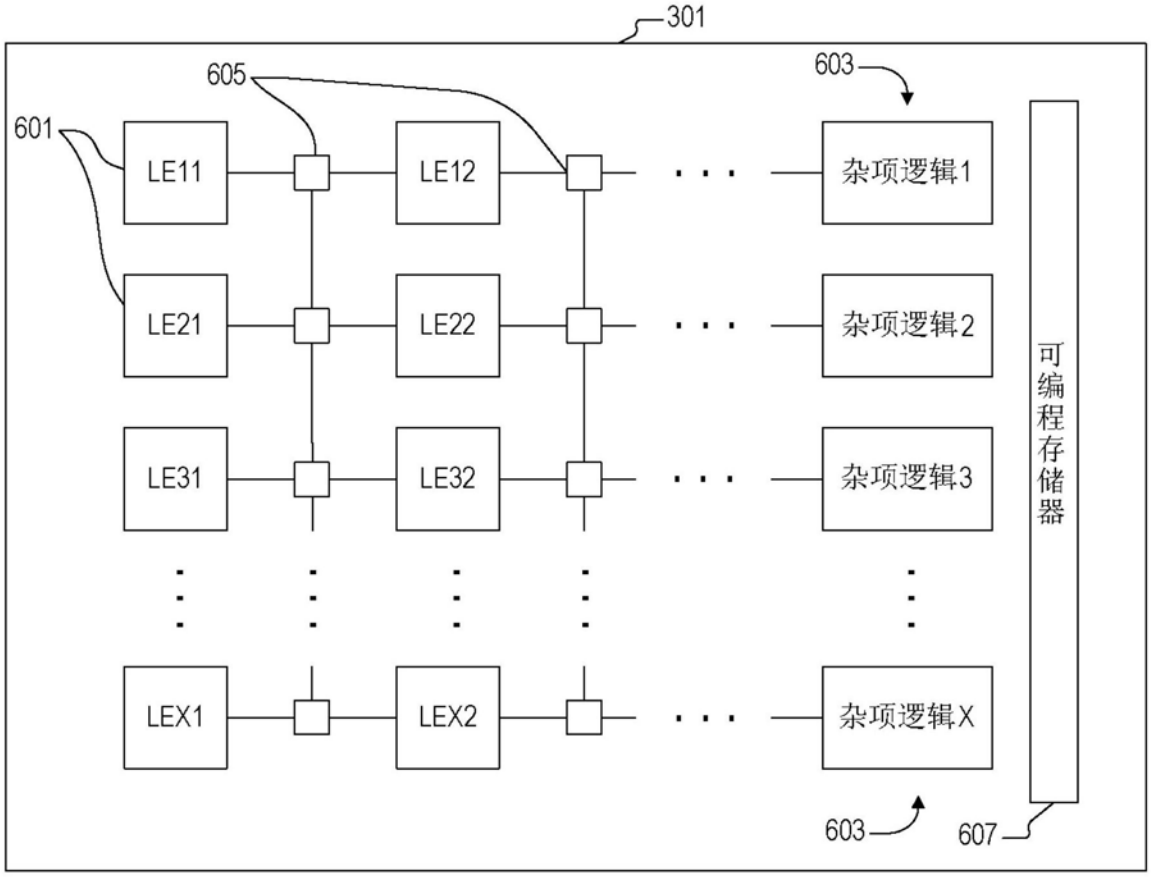

301
605
603
601
LE11
LE12
杂项逻辑1
LE21
LE22
杂项逻辑2
LE31
LE32
杂项逻辑3
LEX1
LEX2
杂项逻辑X
可编程存储器
603
607

图6

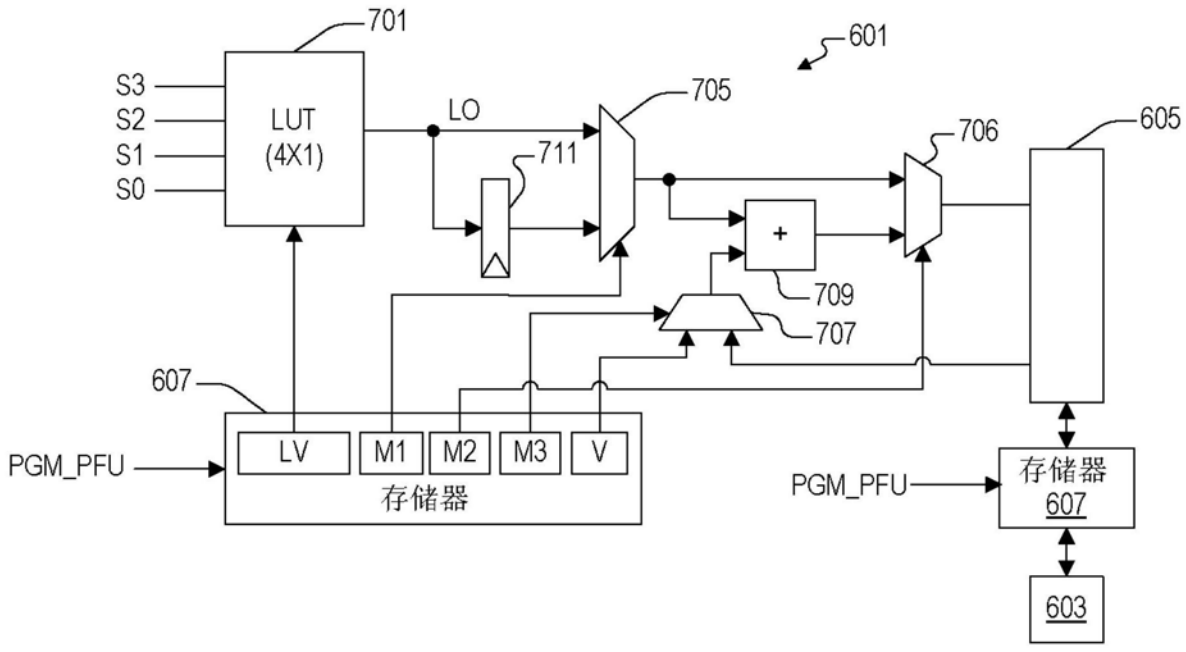

701
601
S3
S2
S1
S0
LUT
(4X1)
LO
705
711
706
605
+
709
707
607
PGM_PFU
LV
M1
M2
M3
V
存储器
PGM_PFU
存储器
607
603

图7

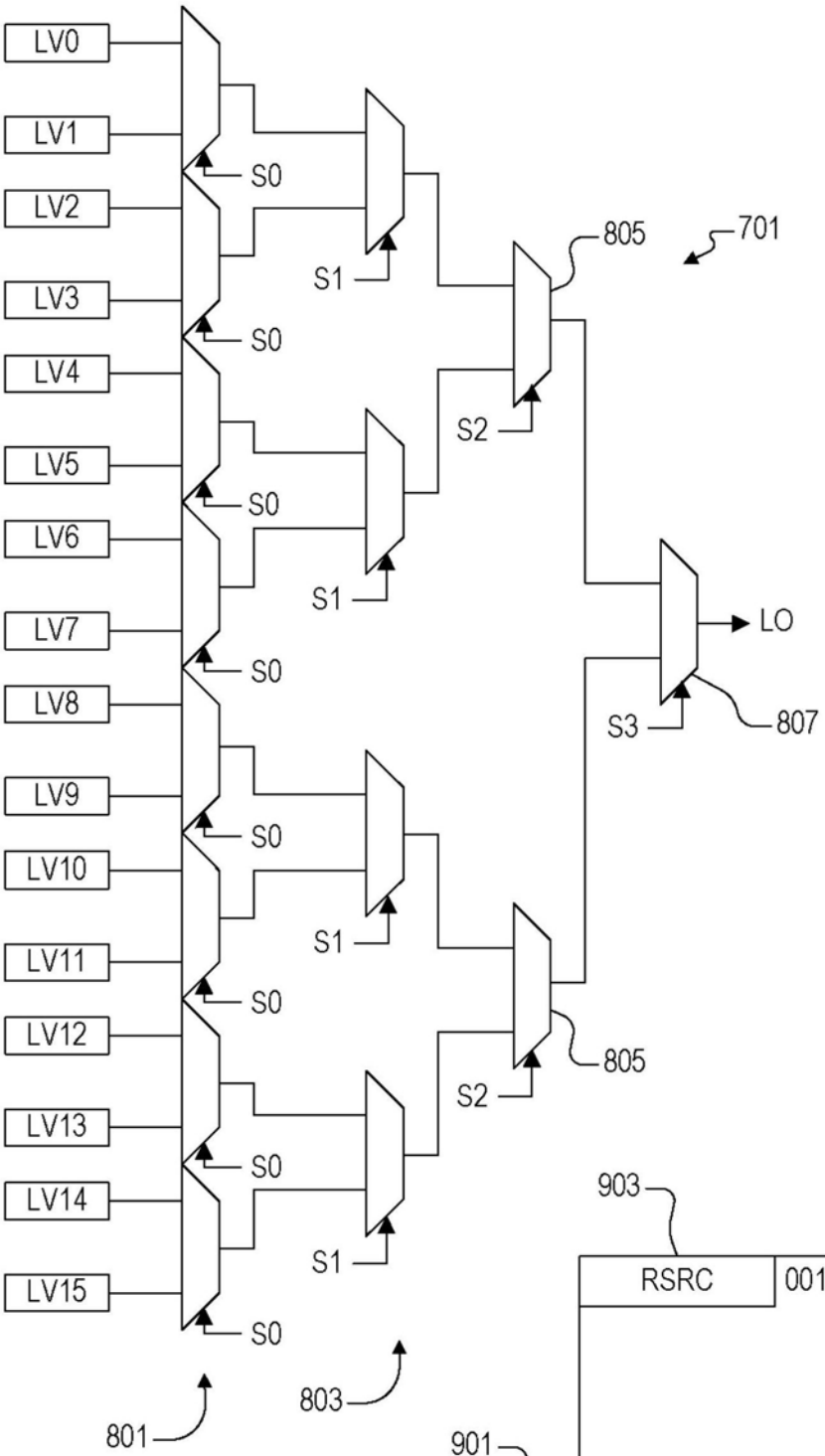

LV0
LV1
LV2
LV3
LV4
LV5
LV6
LV7
LV8
LV9
LV10
LV11
LV12
LV13
LV14
LV15
S0
S1
S2
S3
805
701
LO
807
801
803

图 8

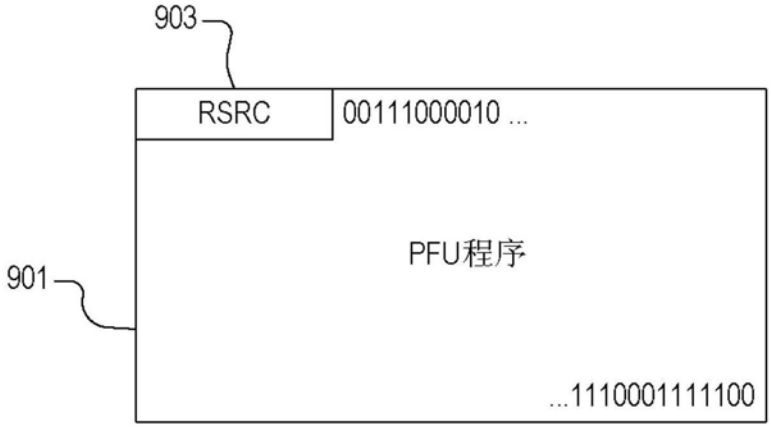

903
RSRC
00111000010 ...
PFU程序
...1110001111100
901

图 9


1002
PPF
功能描述
1004
PPF
编程工具
PFU程序
116

图10

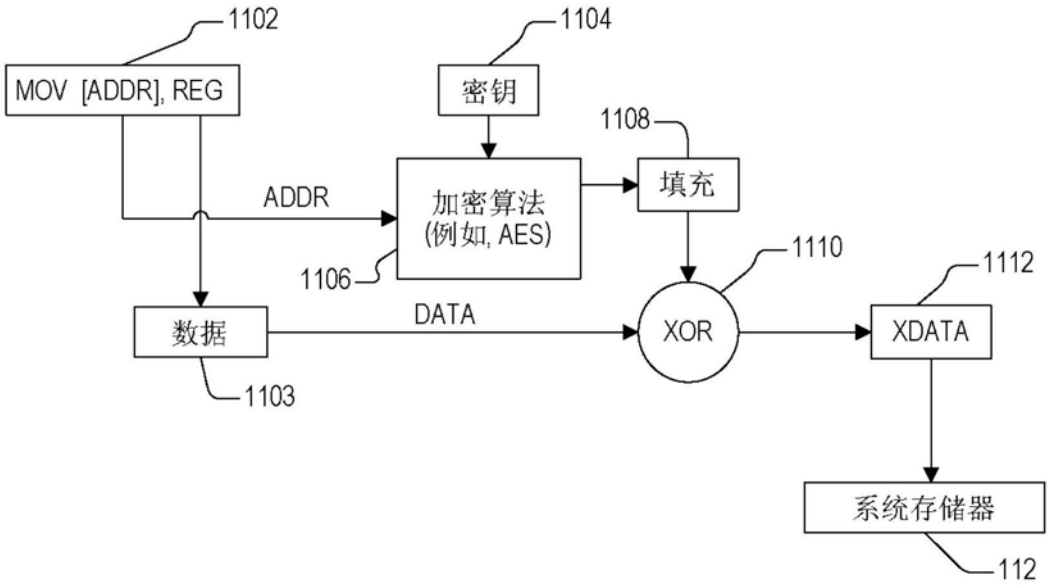

1102
MOV [ADDR], REG
1104
密钥
1108
填充
ADDR
加密算法
(例如, AES)
1106
1110
1112
数据
DATA
XOR
XDATA
1103
系统存储器
112

图11

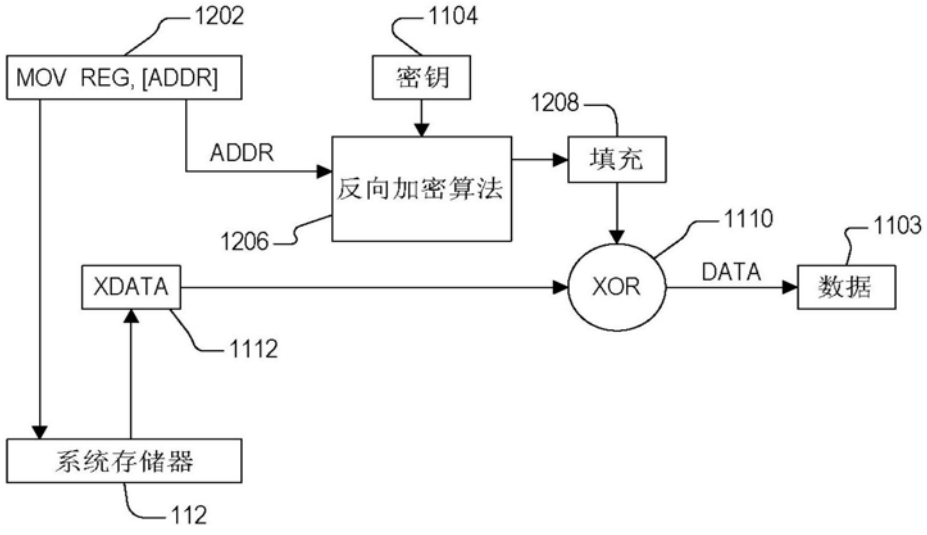

1202
MOV REG, [ADDR]
1104
密钥
1208
ADDR
反向加密算法
填充
1206
1110
1103
XDATA
XOR
DATA
数据
1112
系统存储器
112

图12