[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 21/00 (2006.01)

[12] 发明专利申请公布说明书

[21] 申请号 200780008396.2

[43] 公开日 2009 年 4 月 1 日

[11] 公开号 CN 101401103A

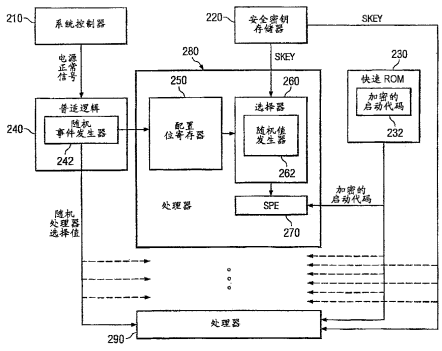
[22] 申请日 2007.5.11
[21] 申请号 200780008396.2
[30] 优先权
[32] 2006. 6. 9 [33] US [31] 11/423,342
[86] 国际申请 PCT/EP2007/054575 2007.5.11
[87] 国际公布 WO2007/141112 英 2007.12.13
[85] 进入国家阶段日期 2008.9.9
[71] 申请人 国际商业机器公司
地址 美国纽约阿芒克
[72] 发明人 C·M·奥尼尔 J·J·德门特
J·N·戴尔 C·J·斯潘迪科
[74] 专利代理机构 北京市金杜律师事务所
代理人 王茂华 李 辉
权利要求书 3 页 说明书 31 页 附图 9 页
按照条约第 19 条的修改 3 页

[54] 发明名称

用于跨越多个处理器的安全启动的系统和方法

[57] 摘要

提供一种用于跨越多个处理器的安全启动的系统和方法。 通过该系统和方法，将启动代码划分为多个启动代码分区。 将多处理器系统的处理器选为启动处理器，并且为每个处理器提供启动代码分区，以便按照预定的启动代码序列执行。 每个处理器根据启动代码序列来执行其启动代码分区，并且将其启动代码分区的成功、未折衷执行通过信号告知给下一处理器。 如果任何处理器都没有通过信号告知其启动代码分区的成功完成和/或未折衷执行，则启动操作失败。 关于启动操作，例如可以按照菊花链、环形或者主/从布置来布置处理器。



1. 一种与具有多个处理器的数据处理系统结合使用的用于启动所述数据处理系统的方法，包括:

将启动代码划分为多个启动代码分区;

在所述数据处理系统的所述多个处理器中的多个启动处理器的每一个中从所述多个启动代码分区中载入启动代码分区；以及

将所述多个启动代码分区作为多个会话在其各自相关联的启动处理器上执行，由此启动所述数据处理系统。

2. 根据权利要求 1 所述的方法，其中，当每个启动代码分区在其相关联的会话中的执行完成时，执行所述启动代码分区的所述多个处理器中的相关联启动处理器可进行操作以通过信号将所述启动代码分区的成功完成告知与启动代码序列中的下一启动代码分区相关联的另一启动处理器。

3. 根据权利要求 1 或权利要求 2 所述的方法，其中，在其各自相关联的启动处理器上执行所述多个启动代码分区包括:

在启动处理器之间的通信中使用安全机制，以确保所述多个启动代码分区在其各自相关联的启动处理器上的非折衷执行。

4. 根据权利要求 3 所述的方法，其中，所述安全机制至少包括以下各项中的一个: 在启动处理器之间传递安全令牌，在启动处理器之间传递数字签名，使用密码，传递启动代码分区的校验和，或者使用信号的公开密钥/私有密钥加密。

5. 根据前述权利要求中任一项所述的方法，其中，启动代码分区的数目等于启动处理器的数目。

6. 根据前述权利要求中任一项所述的方法，进一步包括:

从所述多个处理器中随机选择所述启动处理器，其中所述启动处理器是所述多个处理器的子集。

7. 根据权利要求 6 所述的方法，进一步包括:

在未被随机选作启动处理器的所述多个处理器的处理器上执行掩码。

8. 根据前述权利要求中任一项所述的方法，进一步包括:

随机选择哪个启动代码分区与每个启动处理器相关联，其中每个启动代码分区可以不同于其他启动代码分区。

9. 根据前述权利要求中任一项所述的方法，其中，用于所述启动处理器上的启动代码分区执行的会话布置为菊花链布置、环形布置或者主/从布置之一。

10. 一种与包含多个处理器的数据处理系统结合使用的用于启动所述数据处理系统的设备，包括:

用于将启动代码划分为多个启动代码分区的装置;

用于在所述数据处理系统的所述多个处理器中的多个启动处理器的每一个中从所述多个启动代码分区中载入启动代码分区的装置;以及

用于将所述多个启动代码分区作为多个会话在其各自相关联的启动处理器上执行以由此启动所述数据处理系统的装置。

11. 根据权利要求10所述的设备，进一步包括:用于当每个启动代码分区在其各自相关联的会话中的执行完成时，通过执行所述启动代码分区的所述多个处理器中的相关联启动处理器利用信号将所述启动代码分区的成功完成告知与启动代码序列中的下一启动代码分区相关联的另一启动处理器。

12. 根据权利要求10或者权利要求11所述的设备，进一步包括:用于在启动处理器之间的通信中使用安全机制以确保所述多个启动代码分区的非折衷执行的装置。

13. 根据权利要求12所述的设备，其中，所述安全机制至少包括以下各项中的一个:在启动处理器之间传递安全令牌，在启动处理器之间传递数字签名，使用密码，传递启动代码分区的校验和，或者使用信号的公开密钥/私有密钥加密。

14. 根据权利要求 10 到 13 任一项所述的设备，其中，启动代码分区的数目等于启动处理器的数目。

15. 根据权利要求10到14任一项所述的设备，进一步包括：用于从所述多个处理器中随机选择所述启动处理器的装置，并且其中所述启动处理器是所述多个处理器的子集。

16. 根据权利要求15所述的设备，进一步包括：用于在未被随机选作启动处理器的所述多个处理器的处理器上执行掩码的装置。

17. 根据权利要求10到16任一项所述的设备，进一步包括：用于随机选择哪个启动代码分区与每个启动处理器相关联的装置，并且其中每个启动代码分区可以不同于其他启动代码分区。

18. 根据权利要求10到17任一项所述的设备，其中，用于所述启动处理器上的启动代码分区执行的会话被布置为菊花链布置、环形布置或者主/从布置之一。

19. 一种包括程序代码装置的计算机程序，当所述程序在计算机上运行时，所述程序代码装置适于执行根据权利要求1到9任一项所述的步骤。

用于跨越多个处理器的安全启动的系统和方法

技术领域

本申请总体上涉及一种改进的数据处理系统和方法。更具体地，本申请涉及用于跨越多个处理器的安全启动的系统和方法。

背景技术

由于我们的社会变得越来越依赖电子通信和信息存储，所以对数字信息安全的关注也与日俱增，例如个人信息和数字权利管理（DRM）。而且，近年来计算机黑客和其他未授权入侵者进入计算机系统的技巧也越发高明。结果，在用于计算装置的安全系统的发展中投入了很多的努力，以便可以保护这种敏感的数字信息而不受未授权访问的威胁。

入侵者可以获得对计算系统的访问的一种方法是通过电子接口和其他可观察的电磁或热活动来观察计算系统的启动活动。通过按照这种方式来观察启动活动，入侵者可以推断出启动处理器正输入和输出什么数据信号，处理器上正运行什么加密算法等等。根据该信息，入侵者可以检测到启动序列中可以进行未授权侵入的点。此外，利用其中要求安全性密钥用以启动系统的安全启动序列，入侵者可以逆推出启动处理器所使用的加密算法，以获得对安全性密钥的访问，并且从而获得对计算系统的完全访问。由于计算系统的总体安全性经常取决于启动处理的安全性，所以当入侵者获得对启动序列的访问时，整个系统的安全性就处于危险当中。

因此，提供一种增加监控处理器的启动序列的难度以便使系统更安全而不受未授权入侵的威胁的装置和方法是有益的。

发明内容

示范性实施方式提供一种用于选择随机处理器来启动多处理器系统以及用于提供跨越多个处理器的安全启动的系统和方法。通过使得哪个处理器将用来启动多处理器系统随机化，来使未授权人员以击败系统的安全性为目的而监控电子接口、热活动和其他电磁活动来获得有关启动序列的信息的能力变得更加困难。例如，在多处理器系统中，未来入侵者需要以多个不同的次数运行启动序列，同时监控单个处理器以期望它可能被随机地选为启动处理器，或者未来入侵者需要在启动时监控所有的处理器，以便确定哪个处理器是实际启动处理器。这两种可选择方案都需要未来入侵者方付出相当大努力，这可以用作在实际中试图监控该系统来获得启动序列信息的不利因素，或者至少显著地延迟未来侵入者在危害系统时花费的时间。

利用示范性实施例的机制，在多处理器系统（例如片上系统）上提供普适逻辑，该普适逻辑控制多处理器系统的启动操作。普适逻辑包括随机事件发生器，该随机事件发生器随机选择在多处理器系统中的哪一个处理器将要成为启动处理器，该启动处理器运行启动代码，以由此使该系统达到操作状态。根据对启动处理器的随机选择，设置与启动处理器相关联的配置位，该配置位表示该处理器成为启动处理器。此后，所选择的启动处理器被提供有为使多处理器系统安全启动到操作状态所必需的安全性密钥。

在某些示范性实施例中，当随机选择的处理器执行安全启动操作时，多处理器系统中的其他处理器执行操作来对真正的安全启动操作进行掩码（mask）。该掩码包括执行其他代码序列而不是启动代码序列，该其他代码序列使处理器生成电磁和/或热输出，其中如果该电磁和/或热输出受到入侵者监控，则会使入侵者难以辨识哪一个处理器正在执行实际的安全启动操作。

一种可以生成不同的代码序列的方法是将随机延迟元素插入启动代码内，该启动代码运行重复一个随机量的循环。按照这种方式，每个处理器可以运行启动代码，但是具有不同的延迟量，从而使得生成不同的电磁和热特征（signature）。从入侵者的角度来看，由于这种

掩码而使得很难从多处理器系统中的其他处理器中辨识实际启动处理器。

在另一个示范性实施例中，由其他处理器执行的代码序列是与随机选择的处理器所执行的相同的启动代码序列，但由其他处理器执行的代码序列具有伪（dummy)安全性密钥。因此，对于入侵者来说，这些其他的处理器操作为以及看起来好像它们正在执行安全启动操作一样。然而，如果监控这些处理器，就会识别出假电磁和热输出，这使入侵者难以确定所监控的处理器是否为正在执行安全启动操作的实际随机选择的处理器。

在另一个示范性实施例中，可以通过提供伪处理器来对随机选择的启动处理器进行掩码。通过在伪处理器上运行不同于启动代码序列的处理从而使系统上的攻击改向该伪处理器，使得从电磁、热等的监控装置的角度来看，伪处理器看起来好像是唯一的。按照这种方式，当入侵者试图通过避开安全机制来访问该系统时，该入侵者仅访问了伪处理器而没有对多处理器系统的剩余部分进行实际的访问。

在其他示范性实施例中，可以跨越多处理器系统中的多个处理器来分布启动代码序列。通过跨越多处理器系统中的多个处理器来分布启动代码序列，增加了必须被危害以便获得关于启动序列的完整信息从而规避安全性措施的处理器的数目。因此，示范性实施例的分布式启动操作比利用单个安全核的多处理器数据处理系统更安全。此外，通过分布启动操作，如果启动操作的任何部分被危害，则启动操作都会失败，从而阻止未授权个体规避系统的安全性。

利用该示范性实施例，将启动代码序列划分成多个分区，以便可以将每个分区提供给多处理器系统的不同处理器。当执行启动代码序列的每个分区时，在启动代码序列可以在另一个处理器上进行之前，该分区必须在其各自的处理器上正确地完成。利用安全通信机制来传达对启动代码序列的先前分区的满意完成。该安全通信机制可以包括安全令牌，例如加密的密码或其他安全标识符，例如公钥/私钥对，其表示没有危害先前的会话。按照这种方式，创建必须令人满意地完成

相关的“会话”链。

启动代码的分布式执行中所涉及的处理器可以是多处理器系统中的所有处理器或者是多处理器系统中的处理器子集。例如，可以按分布方式利用正如上述用于选择单个启动处理器的随机选择机制来随机地选择将用于启动系统的多个启动处理器。而且，可以随机地选择由处理器所执行的启动代码的特定分区，使得利用每个加电复位（POR）操作，相同的处理器可以执行或不执行与之前 POR 操作中相同的启动代码分区。因此，可以对于分布式启动操作中涉及哪些处理器以及每个处理器将执行哪些启动代码分区进行随机化。

多处理器系统的其他处理器（也即，非启动处理器）可以在分布式启动操作期间不执行任何操作，或者可以执行先前描述的各种掩码示范性实施方式中一个或多个的掩码序列，以便对随机选择的处理器子集上的启动代码执行进行掩码。换言之，本示范性实施方式的分布式启动代码序列操作可以与先前描述的一个或多个示范性实施方式相结合。

在一个示范性实施方式中，提供一种在具有多个处理器的数据处理系统中用于启动该数据处理系统的方法。该方法可以包括：将启动代码划分为多个启动代码分区，以及在该数据处理系统的多个处理器内的多个启动处理器的每一个中从所述多个启动代码分区载入启动代码分区。多个启动代码分区可以在其各自相关联的启动处理器上作为多个会话来执行，由此启动数据处理系统。启动代码分区的数目可以等于启动处理器的数目。

如果任何会话导致启动代码分区的不成功执行或折衷执行，则数据处理系统的启动可能失败。当每个启动代码分区在其相关联会话中的执行完成时，执行所述启动代码分区的多个处理器中的相关联启动处理器可以通过信号将启动代码分区的成功完成告知与启动代码序列中的下一启动代码分区相关联的另一启动处理器。

在其各自相关联的启动处理器上执行多个启动代码分区可以包括在启动处理器之间的通信中使用安全机制，以确保多个启动代码分区

在其各自相关联的启动处理器上的非折衷执行。该安全机制可以至少包括以下各项中的一个：在启动处理器之间传递安全令牌，在启动处理器之间传递数字签名，使用密码，传递启动代码分区的校验和，或者使用信号的公开密钥/私有密钥加密。

该方法可以进一步包括从多个处理器中随机选择启动处理器，其中启动处理器是多个处理器的子集。可以在未被随机选作启动处理器的多个处理器的处理器上执行掩码。

而且，该方法可以包括随机选择哪个启动代码分区与每个启动处理器相关联。每个启动代码分区可以不同于其他启动代码分区。

此外，可以将用于启动处理器上的启动代码分区执行的会话布置为菊花链布置、环形布置或者主/从布置之一。数据处理系统可以是异质的多处理器片上系统，其具有根据第一指令集进行操作的第一处理器以及根据不同于该第一指令集的第二指令集进行操作的一个或多个第二处理器。

在另一示范性实施方式中，提供一种数据处理系统，其包含多个处理器，耦合至所述多个处理器的启动代码存储装置，以及耦合至所述多个处理器的普适逻辑。启动代码存储装置可以存储被划分为多个启动代码分区的启动代码。普适逻辑可以执行上文关于先前描述的示范性方法实施方式而概括的各种操作及其组合。

在又一示范性实施方式中，提供一种计算机程序产品，包括具有计算机可读程序的计算机可用介质。当在数据处理系统上执行时，所述计算机可读程序可以使该数据处理系统执行上文关于先前描述的示范性方法实施方式而概括的各种操作及其组合。

将在以下本发明的示例性实施例的详细说明中描述本发明这些和其他特征与优势，对于本领域普通技术人员来说，考虑这些详细描述将使得本发明这些和其他特征与优势变得明显。

附图说明

在所附的权利要求中阐述了被认为是本发明特性的新颖性特征。

然而，在连同附图一起进行阅读时，参照以下示范性实施例的详细描述，将更好地理解发明本身及其优选使用方式、进一步目的和优势，其中：

图 1 是其中可以实现示例性实施例的多处理器系统的示意性方框图；

图 2 是描述根据一个示范性实施例的随机启动处理器选择机制的主要操作部件的示意图；

图 3A 是描述了根据一个示范性实施例的随机选择机制的示意图；

图 3B 是根据一个示范性实施例的抖动的图形表示，其中该抖动被引入到随机事件发生器的 LFSR 计数器的输入中；

图 3C 是描述了一个示范性实施例的示意图，其中利用平行信号线向处理器提供秘密密钥和多个随机生成的密钥值；

图 4A-4D 是描述根据示范性实施例的用于对随机选择的启动处理器的安全启动操作进行掩码的掩码操作的示意图；

图 5 是概述了用于在多处理器系统中随机地选择一个处理器作为启动处理器的示范性操作的流程图；

图 6 是概述了根据一个示范性实施例的用于对启动代码序列进行掩码的示范性操作的流程图；

图 7A 是描述了根据一个示范性实施例配置成菊链或环形布置的分布式启动操作的示意图；

图 7B 是描述了根据一个示范性实施例配置成主/从布置的分布式启动操作的示意图；以及

图 8 是概述了根据一个示范性实施例用于分布式启动多处理器系统的示范性操作的流程图。

具体实施方式

示范性实施例提供了一种用于选择随机处理器来启动多处理器系统的装置和方法。该示范性实施例可以供任何多处理器系统使用，可以在其中选择一个处理器来启动多处理器系统。因此，示范性实施例

的机制适用于对称多处理器（SMP）系统、异构多处理器系统、非相干不平衡多处理器系统等等。

一个可以在其中实现示范性实施例的多处理器系统是可从纽约阿芒克的国际商业机器公司获得的单元宽带引擎（CBE）。将参照 CBE 体系结构来说明该示范性实施例，然而，应该理解，对示范性实施例的说明仅仅是示范性的，而且并非旨在声明或暗示任何关于可以实现示范性实施例的机制的多处理器系统的类型或结构的限定。在不脱离本发明的精神和范围的情况下，可以对所述的 CBE 体系结构进行许多修改。

图 1 是在其中可以实现本发明的方面的数据处理系统的示意图。图 1 所示的示范性数据处理系统是单元宽带引擎（CBE）数据处理系统的一个实例。虽然 CBE 将用于本发明优选实施例的说明中，但是正如本领域技术人员在阅读以下描述后就会很容易理解的那样，本发明并不局限于此。

如图 1 所示，CBE 100 包括电源处理器元件（PPE）110 和多个协同处理器元件（SPE）120-134，其中电源处理器元件（PPE）110 具有电源处理器单元（PPU）116 及其 L1 和 L2 高速缓冲存储器 112 和 114，每个协同处理器元件（SPE）120-134 都具有它自己的协同处理器单元（SPU）140-154、存储器流量控制 155-162、本地存储器或存储器（LS）163-170 和总线接口单元（BIU 单元）180-194，例如，该总线接口单元可以是组合直接存储器访问（DMA）、存储器管理单元（MMU）和总线接口单元。还提供了高宽带内部元件互连总线（EIB）196、总线接口控制器（BIC）197 和存储器接口控制器（MIC）198。

CBE 100 可以是片上系统，以便可以在单个多处理器芯片上提供图 1 所述的每个元件。此外，CBE 100 是一种异构处理环境，其中每个 SPU 都可以从系统内的每个其他 SPU 接收不同指令。而且，为 SPU 而设置的指令不同于为 PPU 设置的指令，例如，PPU 可以执行基于简化指令系统计算机（RISC）的指令，而 SPU 执行单指令多数据（SIMD）指令。

SPE 120-134 通过 EIB 196 相互耦合并与 L2 高速缓冲存储器 114 耦合。另外，SPE 120-134 通过 EIB 196 与 MIC 198 和 BIC 197 耦合。MIC 198 提供到共享存储器 199 的通信接口。举例来说，BIC 197 在 CBE 100 与其他外部总线和设备之间提供通信接口，例如 SouthBridge™通信处理器。

PPE 110 是双线程 PPE 110。这种双线程 PPE 110 与 8 个 SPE 120-134 的组合使 CBE 100 能够处理 10 个同时的线程和超过 128 个未决存储器请求。PPE 110 用作对于处理大部分计算工作负荷的其他 8 个 SPE 120-134 的控制器。举例来说，PPE 110 可用来运行传统操作系统，而 SPE 120-134 进行矢量化浮点代码执行。

SPE 120-134 包括协同处理单元（SPU）140-154、存储器流量控制单元 155-162、本地存储器或存储器 163-170 以及总线接口单元 180-194。在示范性实施例中，本地存储器或存储器 163-170 包括 256KB 的指令和数据存储器，该指令和数据存储器对于 PPE 110 是可见的并且可由软件直接进行寻址。

PPE 110 可以向 SPE 120-134 加载小程序或线程，使 SPE 链接在一起以处理复合操作中的每个步骤。例如，结合了 CBE 100 的机顶盒可以加载用于读取 DVD、视频和音频解码以及显示的程序，而且数据将会挨个经过 SPE，直至它最后在输出显示器上而结束。在 4GHz 处，每个 SPE 120-134 利用具有类似性能水平的 PPE 110 给出理论上为 32 GELOPS 的性能。

存储器流量控制单元（MFC）155-162 用作 SPU 与系统其他部分和其他部件的接口。MFC 155-162 提供主要机制，用以在主存储器与本地存储器 163-170 之间进行数据传输、保护和同步。逻辑上存在针对处理器内的每个 SPU 的 MFC。某些实现可以多个 SPU 之间共享单个 MFC 的资源。在这种情况下，对于每个 SPU 来说，所有为 MFC 定义的设备和命令都必须与软件独立地出现。将共享 MFC 的效果限定为实现相关的设备和命令。

示范性实施例提供了一种用于选择一个随机处理器（诸如 SPE

120-134 中的一个）来启动多处理器系统（例如 CBE 100）的设备和方法。通过使将哪个 SPE 120-134 将用来启动 CBE 100 随机化，使未授权人员为了击败 CBE 100 的安全性而监控电子接口、热活动和其他电磁活动来获得关于启动序列的信息的能力变得更加困难。

利用示范性实施例的机制，在 CBE 100 上提供控制 CBE 100 的启动操作的普适逻辑 193。普适逻辑 193 包括随机事件发生器，它随机地选择哪一个 SPE 120-134 将会成为启动处理器，其中该启动处理器运行启动代码，以由此使系统启动到操作状态。根据启动 SPE 120-134 的随机选择，设置与所选择的 SPE（例如 SPE 120）相关联的配置位，其表示 SPE 120 为实际启动处理器。此后，所选择的 SPE 120 被提供有为使 CBE 100 安全启动到操作状态所必需的安全性密钥。当所选择的 SPE 成功地完成安全启动过程时，该 SPE 将会从安全状态转变成解锁状态，其中在安全状态期间，关闭 MIC 198、共享存储器 199 和 BIC 197 的一部分（除了到图 2 中快速 ROM 230 的通信链路），并阻止它们进行操作。一旦安全 SPE 进入解锁状态，它将通过执行快速 ROM 230 所提供的加密的代码来初始化充分启用 MIC 198、BIC 197（称为“训练”的处理）和所有其他处理器（SPE 和 PPE）的处理。对于与用于单元宽带引擎中的安全启动处理有关的更多信息，请参考公开号为 No.20050021944 的共同未决和普通转让的美国专利申请，在此引用该专利申请作为参考。

在某些示范性实施例中，当随机选择的 SPE 120 执行安全启动操作时，其他 SPE 122-134 执行操作，来对实际的安全启动操作进行掩码。该掩码包括执行其他代码序列而不是启动代码序列，该其他代码序列使 SPE 122-134 能够生成电、电磁和/或热输出，其中当这些电、电磁和/或热输出受到入侵者监控时会使入侵者难以辨识哪一个 SPE 120-134 正在执行实际的安全启动操作。

一种可以生成不同的代码序列的方法是在启动代码中插入随机延迟元素，该启动代码运行重复了一个随机量的循环。添加这些随机延迟元素，以便在启动处理器时，安全启动算法将以随机的方式进行改

变，以导致不同的电磁和热特征，从而使得难以随着时间变化来比较两个不同的启动操作。按照这种方式，每个 SPE 120-134 都可以运行启动代码，但是具有不同的延迟量，从而导致生成不同的电磁和热特征。而且，相同的 SPE 120-134 将会在每次它运行安全的启动代码时生成不同的电磁和热特征。从入侵者的角度来看，由于这种掩码，使得很难从 CBE 100 的其他处理器中辨识实际的启动 SPE 120。

在另一个实施例中，由其他 SPE 122-134 执行的代码序列是与随机选择的 SPE 120 所执行的相同的启动代码序列，但由其他 SPE 122-134 执行的代码序列具有伪安全性密钥。因此，对于入侵者来说，这些其他的 SPE 122-134 操作为并且看起来好像它们正在执行安全启动操作一样。然而，如果监控 SPE 122-134，则会识别到伪电、电磁和热输出，这使入侵者难以确定所监控的 SPE 是否为正在执行安全启动操作的实际随机选择的 SPE 120。

在另一个示范性实施例中，可以通过提供一个伪 SPE（未示出）来对随机选择的启动 SPE 120 进行掩码。通过在伪 SPE 上运行不同于启动代码序列的处理，从而使 CBE 100 上的攻击改向到该伪 SPE，使得从电磁、热等监控装置的角度来看，伪 SPE 看起来好像是唯一的。按照这种方式，当入侵者试图通过避开安全机制来访问该系统时，该入侵者只访问了伪 SPE 而没有对 CBE100 的剩余部分进行实际的访问。而且，如果入侵者危害伪 SPE 而试图执行代码，则然后伪 SPE 可以关闭 CBE 100 的剩余部分以防止进一步的侵入意图。

现在将更详细地描述上述每个示范性实施例。应当理解，虽然在这里分开地描述了每个示范性实施例，但是可以按各种方式来组合示范性实施例，以便获得多处理器系统（例如，CBE 100）更大的安全性。因此，被认为适合特定情况和多处理器环境的示范性实施例的任何组合都在本发明的精神和范围内。

图 2 是描述了根据一个示范性实施例的随机启动处理器选择机制的主要操作部件的示意图。应该理解，为了简化示范性实施例的解释，图 2 仅仅详细地显示了多处理器系统中的一个处理器。然而，应该理

解，多处理器系统中的每个处理器都具有类似的元件配置，并且按照与图 2 清楚示出的处理器相同的方式进行操作。在不脱离本发明精神和范围的情况下，多处理器系统内可以包括任意数量的处理器。然而，为了解释示范性实施例，假设处理器的数目是如图 1 所示的 CBE 体系结构中的 8 个。

如图 2 所示，随机启动处理器选择机制的主要操作部件包括系统控制器 210、安全密钥存储器 220、快速 ROM 230 和普适逻辑 240。在一个示范性实施例中，以图 1 的 CBE 体系结构作为示范，元件 210-240 可以是在其中实现 CBE 体系结构的芯片上所提供的元件。也就是说，可以将这些元件 210-240 整合到多处理器片上系统（SoC）的逻辑中，因而可以在片上执行由这些元件 210-240 所执行的操作。可替换地，也可以在片外提供一个或多个元件，例如，可以在片外提供快速 ROM 230。

系统控制器 210 负责执行加电复位（POR）的初始操作，以使系统电源达到一个可接受且稳定的水平。也就是说，正如现有技术中通常已知的那样，系统控制器 210 负责提升电压，开启系统时钟，以及为了使多处理器系统达到可以在其中开始启动操作的状态所需要的其他初始操作。作为该 POR 操作的一部分，使处理器 280-290 达到一种安全操作模式。在该安全操作模式中，在处理器以外无法访问该处理器的本地存储器。一旦完成这些初始操作且系统处于可接受电源状态，系统控制器 210 就利用信号将“电源正常”状态通知给普适逻辑 240。

响应于来自系统控制器 210 的“电源正常”信号，普适逻辑 240 开始启动操作，用于使多处理器系统启动到操作状态，以便软件程序可以开始执行。作为启动操作的一部分，普适逻辑 240 的随机事件发生器 242 随机地选择其中一个处理器（例如处理器 280）作为多处理器系统的启动处理器。随机事件发生器 242 生成一个信号，其发送到多处理器系统中的每个处理器。该信号仅仅对于选为启动处理器的处理器在逻辑上为高。该信号有效地将随机选择的处理器 280 的配置位寄

存器 250 内的值设置成表示该处理器 280 为启动处理器的值，例如“1”。其他处理器将使它们各自的配置位寄存器内的配置位值保持为初始值，从而表示这些处理器不是针对多处理器系统的随机选择的启动处理器。

在快速 ROM 230 中以加密的形式存储用于启动多处理器系统的启动代码。可以将加密的启动代码 232 提供给每个处理器 280-290。也就是说，作为启动序列的一部分，每个处理器 280-290 都可以试图从快速 ROM 230 中读取加密的启动代码 232。然而，由于仅仅随机选择了其中一个处理器作为启动处理器，所以其中将只有一个处理器能够对加密的启动代码 232 进行解密并适当地执行启动代码，以便使多处理器系统达到操作状态。这可以通过使用每个处理器中提供的选择器 260 来实现，该选择器在秘密密钥和随机生成的密钥值之间进行选择，其中秘密密钥是用来对加密的启动代码 232 进行解密的密钥值，而随机生成的密钥值将不能对加密的启动代码 232 进行解密。

利用配置位寄存器 250 存储器储的值来生成一个提供给选择器 260 的选择器信号。例如，选择器 260 可以是多路转换器，它接收来自于安全密钥存储器 220 的安全密钥（Skey）作为一个输入，它接收来自于随机值发生器 262 的随机生成的密钥值作为第二输入，并接收来自于配置位寄存器 250 的选择信号，该选择信号表示选择这两个输入中的哪一个。如果配置位寄存器 250 存储一个表示该处理器是随机选择的启动处理器的值，则选择 Skey 输入。如果配置位寄存器 250 存储一个表示该处理器不是随机选择的启动处理器的值，则可由选择器 260 选择该随机生成的密钥值输入。然后将所选择的密钥值输出到 SPE 270。

SPE 270 接收所选择的密钥值和加密的启动代码 232。然后 SPE 270 试图对加密的启动代码 232 进行解密。如果所选择的密钥值是来自于安全密钥存储器 220 的 Skey，则 SPE 270 将能够适当地对加密的启动代码 232 进行解密，并执行在其中的启动代码指令，以使系统达到操作状态。如果所选择的密钥值不是来自于安全密钥存储器 220 的

Skey，则解密将会失败，而且 SPE 270 将无法执行启动代码指令。

可以利用多处理器系统所执行的每个加电复位（POR）操作来执行上述随机选择一个启动处理器并且利用随机选择的启动处理器来启动该多处理器系统的处理。因此，每当启动多处理器系统时，就会从多个处理器中随机地选择一个不同的处理器作为启动处理器。结果，潜在的系统入侵者将无法在之前就确定哪一个处理器是启动处理器以及将多处理器系统的电磁和热状况的测量导向该特定处理器。

相反，潜在的入侵者必须通过多处理器系统的多个启动操作来监控单个处理器，以期望最后将选择该单个处理器作为用作启动处理器的随机处理器，或者潜在的入侵者必须监控所有的处理器，从而识别哪一个处理器是启动处理器，并试图通过测量其各自的电磁及热状况来获得必要的信息。举例来说，在八处理器系统中，由于必须监控所有 8 个处理器，所以使监控启动序列的难度增加了八倍。而且，需要更多的探针和硬件来进行这种监控，从而增加了试图进行这种监控的难度。

图 3A 是描述了根据一个示范性实施例的随机选择机制的示意图。如上所述，支持该示范性实施例的原理概念是从多个处理器中随机选择一个处理器成为多处理器系统的启动处理器。为了进行该随机选择，提供随机事件发生器和选择器机制。在示范性实施例中，在多处理器系统的普适逻辑中提供随机事件发生器，并与每个处理器相关联地提供选择器。图 3A 提供了对根据一个示范性实施例的随机事件发生器和选择器的一个实现的描述。

如图 3A 所示，举例来说，随机事件发生器 310 包括线性反馈移位寄存器（LFSR）计数器 320、环形振荡器 330 和选择器信号寄存器/解码器 340，其中随机事件发生器 310 可以对应于图 2 中的随机事件发生器 242。环形振荡器 330 是一种由奇数个非门组成的器件，这些非门的输出在两个电压电平之间振荡。该非门或反相器以链状耦合，将最后一个反相器的输出反馈到第一反相器中。一连串奇数个反相器的最后输出是第一输入的逻辑非。在断言第一输入之后的一段有限的

时间后断言该最后输出。该最后输出到输入的反馈导致不稳定的振荡，它将根据随机因素而随时间改变，这些因素例如温度和电源上的电磁噪声。

连同时钟信号 clk 一起将环形振荡器 330 的输出提供作为 LFSR 计数器 320 的输入。LFSR 计数器 320 是移位寄存器，其输入位是它的以前状态的线性函数。单个位的唯一线性函数是异或和异或非（inverse-XOR），因此，LFSR 是一种通过对整个移位寄存器值的某些位进行异或操作（XOR）来驱动其输入位的移位寄存器。

LFSR 计数器 320 的初始值称为种子，而且由于寄存器的操作是确定性的，所以完全由 LFSR 计数器 320 目前（或以前）的状态来确定 LFSR 计数器 320 所生成的序列值。带有精选的反馈功能的 LFSR 计数器 320 可以生成出现随机和具有很长周期的序列位。在示范性实施例中，使该随机性更加明显，因为到 LFSR 计数器 320 的输入是由环形振荡器 330 所生成的振荡结果与环形振荡器 330 的频率和输入时钟 clk 的频率之间的差异的乘积，其中环形振荡器 330 的频率和输入时钟 clk 的频率彼此独立的变化。

LFSR 计数器 320 接收来自于环形振荡器 330 的输出以及时钟信号 clk 作为输入，并生成一个输出位流，其存储在选择器信号寄存器/解码器 340 内。环形振荡器 330 的反相器在到 LFSR 计数器 320 的输出信号中引入延迟，因此，在环形振荡器 330 的频率与输入时钟 clk 的频率之间就存在差异。如图 3B 所示，频率之间的差异引起到 LFSR 计数器的输入中的抖动。该抖动提供一种随机性的度量，它使 LFSR 计数器 320 所生成的输出随机化。

在选择器信号寄存器/解码器 340 中存储 LFSR 计数器 320 的输出。在所述实例中，LFSR 计数器 320 是一个 3 位的计数器，它生成一个 3 位的输出，其中解释该输出来对值 1-8 进行编码。选择器信号寄存器/解码器 340 的解码器功能根据随机的 3 位的输入值选择 8 个唯一输出中的一个。根据选择器信号寄存器 340 中所存储的位的状态，向各种处理器（例如图 1 中的 SPE0-SPE7 120-134）的配置位寄存器输出

高或低状态信号，从而设置配置位寄存器内所存储的值，因此选择其中一个处理器成为多处理器系统的启动处理器。

一旦设置了配置位寄存器值，就使用这些值来向相应的选择器350-370提供选择器信号。如图3A所示，连同Skey输入和随机密钥值输入一起向多路转换器352、362、372提供选择器信号。根据选择器信号的状态，由每个多路转换器352、362、372选择Skey输入或随机密钥值输入。可由一个或多个与上述用于选择启动处理器的随机事件发生器配置相同或不同类型的随机值发生器来生成该随机密钥值输入。也就是说，如上所述的类似的随机事件发生器配置可以用来随机生成一个长度与Skey相同的密钥值。然后将这些随机密钥值输入多路转换器352、362和372。

设计系统，使得例如借助于上述解码器功能，输入到多路转换器352、362、372中的选择器信号中只有一个选择器信号将选择Skey输入，同时其他所有的选择器信号将选择一个随机密钥值输入。将来自于多路转换器352、362、372的输出提供给相应的SPE，以便SPE可以在随机选择的启动处理器的情况下利用这些输出对启动代码进行解密并执行该启动代码，或者正如在多处理器系统中其他所有的处理器的情况下，尝试对启动代码进行解密并且未能启动该多处理器系统。

应当理解，上述用于提供随机事件发生器和选择器的机制仅仅是示范性的，而且并非旨在声明或暗示任何镇对可以用于示范性实施例的随机事件发生器和选择器的类型的限制。例如，可以使用其他随机事件发生器，而不使用图3A所示的环形振荡器和LFSR计数器布置。例如，可以利用热传感器来测量热噪声，该热噪声然后可用来生成用于选择其中一个处理器作为启动处理器的随机事件。类似地，可以利用量子点（q-dot）或半导体钠米晶体来测量量子源效应，其可用作用于选择一个处理器作为启动处理器的随机源。在该示范性实施例中可以使用任何强的随机源来提供处理器随机选择，其中该处理器用作多处理器系统的启动处理器。

而且，应当理解，尽管图 3A 显示了具有 5 个反相器的环形振荡器 330，但是该示范性实施例并不局限于此。反而，在不脱离本发明的精神和范围的情况下，可以使用任何数目的反相器，只要存在奇数个反相器。实际上，为了在到 LFSR 计数器 320 的输入中提供附加的抖动，对环形振荡器 330 中一连串的反相器添加附加的反相器是合乎要求的，以便在输入时钟信号 clk 的频率和来自于环形振荡器 330 的输入之间引入甚至更多差异。可以根据其中实现示范性实施例的特定多处理器系统所要求的操作特性来选择差异的数量。

而且，尽管图 2 和 3A 描述了由单独的随机密钥值发生器为每个处理器生成随机密钥值，但是该示范性实施例并不局限于此。相反，可以为所有处理器提供单个随机密钥值发生器，该随机密钥值发生器生成一个或多个输入到处理器的随机密钥值。因此，举例来说，随机密钥值发生器可以生成提供给所有处理器的单个随机密钥值、对于每个单个处理器的单独的随机密钥值（例如，在此情况下，可以生成 7 个不同的随机密钥值）或者生成可以有选择地提供给多处理器系统的各个处理器的任何数量的随机密钥值。

在示范性实施例中，如图 3C 所示，可以提供多个随机密钥值发生器 390，其中每个随机密钥值发生器都输出一个不同的随机密钥值。可替换地，如上所述，可以用单个随机密钥值发生器来代替这些单独的随机密钥值发生器。可以连同来自于 Skey 存储器 395 的安全密钥（Skey）一起提供这些随机密钥值作为多处理器系统中处理器（例如，SPE 393 和 SPE 394）的选择器（例如，多路转换器 391 和 392）的输入，其中该安全密钥（Skey）例如 eFuse，它实际上用来对用于启动多处理器系统的启动代码进行解密。如图所示，可以对随机生成的密钥值和 Skey 值进行多路复用，并在 8 个相同的信号线上将它们提供给每个多路转换器 391 和 392，以便使入侵者更难以从密钥值存储器 395 中分离出其中作为信号线的一条线路。

可以向多路转换器 391 和 392 提供总共 8 个密钥值输入，而且可以用来自于普适逻辑 397 中的随机事件发生器 396 的选择信号来选择

8 个输入中的一个。在此情况下，多路转换器 391 和 392 可以在 Skey 输入和 7 个随机密钥值之间进行选择而不是简单地在 Skey 输入和一个随机密钥值之间进行选择。因此，第一处理器可以在随机选择第一处理器作为启动处理器的基础上选择 Skey 输入，第二处理器可以选择第三随机密钥值，第三处理器可以选择第四随机密钥值，第五处理器可以选择第一随机密钥值，以此类推。因此，每个处理器可以接收不同的密钥值，或者是 Skey 或者是随机生成的密钥值。结果，对于入侵者来说就变得难以在监控多处理器系统的总线业务量时辨识哪一个密钥值是正确密钥值。

还应当理解，如果是在单芯片上设计的话，优选地在其中提供多处理器系统的陶瓷封装的较低层金属层或互连的最底层中提供图 3A 和 3C 所示的机制。由于探测多处理器的电特征和热特征的能力目前被限定在多处理器陶瓷封装的上层，所以通过将这些元件放置在较低层金属层中，使得探测这些元件的操作的能力变得更加困难。因此，如果可能的话，对于未来的入侵者来说，要监控随机事件发生器和选择器的热特征和电特征以便确定由这些元件所提供的密钥值是非常困难的。

利用上述机制，可以在多处理器系统的多个处理器内随机地选择一个处理器来启动该多处理器系统。按照这种方式，使监控处理器的电特征和热特征以便获得用来启动多处理器系统的秘密信息（例如秘密密钥）的能力变得更加困难，而且潜在地变成对那些希望访问多处理器系统而没有得到授权的人员的不利因素。

尽管上述用于随机选择一个处理器来启动多处理器系统的机制针对监控启动序列提供了良好的保护，但是如果未授权个体足够有耐心的话，他仍然有可能“攻击（hack）”该系统。为了使得基本上不可能进行这样的监控，示范性实施例提供了附加的机制用以在随机选择的处理器上对启动序列进行掩码，以便未授权个体无法辨识哪一个处理器正确地执行用于启动多处理器系统的实际的启动序列。

在一个示范性实施例中，掩码操作包括由每个未选为启动处理器

的处理器运行一个不同的指令集，从而生成掩码的电和热特征，它们使得难以对启动处理器与该系统中的其他处理器进行辨识。由不同的处理器所运行的代码序列可以是相同的默认代码序列，默认的代码序列提供在与处理器相关联的存储器中，或者可以在处理器无法对启动代码序列进行解密时可由该处理器访问。例如，可以在与每个处理器相关联的本地存储器的安全部分中提供默认代码序列。可替换地，可以在片上或片外所提供的快速 ROM 或其他存储设备中提供默认代码序列。

当处理器不能对从快速 ROM 中接收的实际的加密的启动代码进行解密时，处理器默认返回本地存储器的安全部分，该安全部分使处理器执行指令来对另一个处理器上正在执行的启动代码序列进行掩码。该指令序列也许不生成任何可用的信息，而且可能仅仅用作掩码功能。可替换地，举例来说，可以利用该指令序列来执行用于在启动操作期间对系统进行监控的操作或其他有用的操作。

在示范性实施例中，在每个非选择的处理器（例如非启动处理器）上执行的代码是相同的。在其中由每个非选择的处理器所执行的代码为相同的示范性实施例中，在这些非选择的处理器的每一个上运行的代码优选为生成电和热特征文件（profile）的代码，这些特征文件相似于实际启动代码但不提供任何安全信息，其中入侵者为了规避多处理器系统的安全性就需要这些安全信息。这样的代码可以执行类似于实际启动代码的操作，但不访问多处理器系统的敏感部分。实际上，在一个示范性实施例中，用来启动多处理器系统的相同启动代码可由非选择的处理器使用，但是非选择的处理器可访问安全密钥（Skey），而不可访问其它特权信息。

结果，这些非选择的处理器的热特征文件和总线业务量将近似于实际启动序列。这样，从利用监控探针来监控热特征文件、总线业务量等等的入侵者的角度来看，由于通过监控探针来看所有核都是相同的，所以该入侵者将无法译码出哪一个核正在执行实际启动操作。这种不明确性阻止了篡改，并使得更加难以分离出真正的启动代码序

列、秘密密钥信息等等。

在其他示范性实施例中，每个非选择的处理器都可以执行不同的指令集。通过在每个非选择的处理器上执行不同的指令集，在利用电或热探针来监控这些处理器时，没有一个处理器看上去是唯一的。结果，无法通过探针来识别诸如热特征文件或总线业务量那样的明显特征以便识别哪一个处理器是启动处理器。

可以为多处理器系统中每个处理器随机地选择这些不同的指令集。因此，举例来说，可以随机地选择针对在片上存储设备（例如快速 ROM 等）上存储的代码序列的不同开始地址，并将该不同的开始地址提供给多处理器系统的处理器。然后处理器开始执行在随机选择的开始地址处的指令，从而生成不同的热特征文件和总线业务量，以对实际启动代码序列进行掩码。

一种为不同的处理器提供不同的代码序列的方法是提供启动代码，该启动代码具有插入到其内的随机延迟元素。举例来说，这些延迟元素可以是重复随机次数的循环。可以在随机选择的启动处理器所运行的实际启动代码序列中以及在非选择的处理器所运行的启动代码中提供这样的延迟元素。从监控处理器的热和总线业务量特性的入侵者的角度来看，该随机延迟导致该启动代码“看上去”在每个处理器上都不同。结果，入侵者不可能辨识出哪一个处理器正在运行启动多处理器系统的实际启动代码。

在另一个示范性实施例中，提供伪处理器，其中在入侵者监控伪处理器时该伪处理器看起来像是唯一的。该示范性实施例是先前的实施例的组合，其中随机地选择一个处理器作为启动处理器，从非选择的处理器中选择一个处理器作为运行代码的伪处理器，该代码提供来自于启动代码序列的唯一热和总线业务量特征文件，而其他处理器运行尽可能对实际启动代码序列的热特征文件和总线业务量进行复制的代码序列。按照这种方式，入侵者将从其他处理器中检测到唯一的伪处理器，并推断该处理器正在运行实际启动代码序列。因此，入侵者将它的攻击导向该伪处理器而不是实际启动处理器，其中从热特征

文件和总线业务量的观点来看，启动处理器看起来与其他处理器类似。此外，如果入侵者试图运行启动或积极地干扰伪处理器，则伪处理器就可以利用信号通知系统关闭。

图 4A-4D 是描述根据示范性实施例的用于对随机选择的启动处理器的安全启动操作进行掩码的掩码操作的示意图。图 4A 描述了第一掩码操作，其中从监控探针角度来看与启动代码序列相同的代码运行在每个非选择的处理器上。如图 4A 所示，例如借助于之前描述的机制，随机地选择 SPE0 410 作为多处理器系统 400 的启动处理器。因此，SPE0 410 接收秘密密钥，对来自于快速 ROM 的启动代码序列进行解密，并执行为使多处理器系统 400 达到操作状态所要求的实际启动代码操作。其他 SPE（即 SPE1-SPE7 412-424）执行从检测探针的角度看起来好像启动代码序列的代码。

如上所述，其他 SPE 412-424 所运行的代码序列可以是在本地存储器的安全部分中提供的默认代码序列，该安全部分使 SPE 412-424 执行指令来对 SPE 0 410 上正在执行的启动代码序列进行掩码。优选地，在这些非选择的 SPE 412-424 中的每一个上运行的代码是生成电和热特征文件的代码，其中所述电和热特征文件类似于实际启动代码但并不提供任何秘密信息，其中入侵者为了规避多处理器系统的安全性就需要秘密安全信息。这样的代码可以执行与实际启动代码的操作相类似但并不访问多处理器系统 400 的敏感部分的操作。

图 4B 描述了另一个示范性实施例，其中在每个非选择的处理器上运行不同的随机选择的算法。如图 4B 所示，再次选择 SPE0 作为启动处理器，并且因此，SPE0 运行用于将多处理器系统 400 启动到操作状态的启动代码。其他 SPE 412-424 中的每一个都运行单独的随机选择的算法，该算法在 EIB 上生成不同的热特征文件和不同的总线业务量。因此，在与其他 SPE 412-424 中的每一个进行比较时，每个 SPE0-7 看起来是唯一的。因此，不可能辨识出哪一个 SPE0-7 412-424 是用于启动多处理器系统 400 的实际启动处理器。

如上所述，可以为多处理器系统 中的每个 SPE 412-424 随机地选

择这些不同的算法。因此，举例来说，可以随机地选择在片上存储设备（例如快速 ROM 等等）中存储的代码序列的不同开始地址，并将该不同的开始地址提供给 SPE 412-424。然后，SPE 412-424 可以开始执行在随机选择的开始地址处的指令，从而生成对实际启动代码序列进行掩码的不同的热特征文件和总线业务量。

可替换地，可以将启动代码提供给 SPE 412-424 中的每一个，其中在启动代码内插入随机延迟元素。举例来说，这些延迟元素可以是迭代随机次数的循环。从监控处理器的热和总线业务量特性的入侵者的角度，该随机延迟导致启动代码"看上去"在每个处理器上都不同。结果，入侵者不可能辨识出哪一个处理器正在运行用于启动多处理器系统的实际启动代码。

图 4C 描述了另一个示范性实施例，其中提供一个伪处理器，可以使入侵者的攻击改向到该伪处理器。如图 4C 所示，SPE0 是随机选择的启动处理器，它执行启动序列。正如上述参照图 4A 的实施例中的那样，SPE1-SPE4 412-418 和 SPE6-SPE7 422-424 运行从热和总线业务量检测角度看起来像启动代码序列的代码。另一方面，SPE5 420 运行随机选择的算法，其中可以按照上述参照图 4B 所描述的类似方式来随机地选择该算法。

因此，从监控处理器 410-424 的特性的入侵者角度来看，所有 SPE0-SPE4 410-418 和 SPE6-SPE7 422-424 看起来都在执行相同的代码。然而，与其它 SPE 相比，SPE5 420 看起来是唯一的。因此，希望攻击多处理器系统的启动代码序列的入侵者会改向对 SPE 5 420 进行攻击而不是实际启动处理器 SPE0 410，这是因为对于入侵者来说 SPE5 420 看起来是实际启动处理器。

正如利用每个加电复位（POR）操作来随机地选择实际启动处理器，同样可以从非选择的处理器中随机地选择伪处理器。因此，利用每个 POR 操作，可以选择不同的启动处理器和伪处理器，从而使入侵者难以推断哪一个处理器正在执行实际启动序列，该实际启动序列可能会被危害以便获得对多处理器系统的访问。

为完整起见，图 4D 描述了之前上述的示范性实施例，其中由每个处理器来执行用来启动该系统的启动代码。在该示范性实施例中，只允许随机选择的启动处理器访问秘密密钥（Skey），而其他处理器接收随机选择的密钥（ Rkey1-Rkey7）。每个处理器都试图利用提供给它们的密钥（例如 Skey 或 Rkey）来解码和执行该启动代码。只有随机选择的启动处理器能正确地对启动代码进行解密并执行该启动代码来使数据处理系统达到操作状态。然而，对于外部的监控器来说，看起来好像所有处理器都在启动该系统，从而对实际启动处理器进行掩码，这是因为每个处理器都执行类似的任务来试图解密和启动该系统。也就是说，每个处理器都将生成类似的热和/或电特征，这使得未来的入侵者利用检测探针等难以辨识哪一个处理器是实际启动处理器。

通过使用如示范性实施例中所提供的对启动处理器的随机选择和对启动序列的掩码，对于到多处理器系统的未来入侵者来说，辨识哪一个处理器正在执行启动代码序列就变得非常困难。因此，对于未来入侵者来说，要监控处理器热特征文件和总线业务量并识别用于访问加密的启动代码的秘密密钥信息就变得非常困难。此外，对于未来入侵者来说，变得难以识别启动代码序列中可以侵入系统的位置。因此，使多处理器系统更加安全而没有未授权访问启动序列的危险。

图 5-6 是概述用于在多处理器系统内随机地选择一个处理器作为启动处理器并对启动代码序列进行掩码的示范性操作的流程图。应当理解，可以通过计算机程序指令来实现流程图示例中的每个块和流程图示例中的块的组合。可以将这些计算机程序指令提供给处理器或其他可编程数据处理装置来生成机器，以便在处理器或其他可编程数据处理装置上执行的指令创建用于实现在流程图块中所指定功能的装置。还可以在计算机可读存储器或存储介质中存储这些计算机程序指令，其中计算机可读存储器或存储介质可以引导处理器或其他可编程数据处理装置按照特定方式操作，以便存储在计算机可读存储器或存储介质内的指令生成一个制造产品，包括指令装置，该指令装置实现

流程块内所指定的功能。

因此，流程图示例的块支持用于执行指定功能的装置的组合、用于执行指定功能的步骤的组合和用于执行指定功能的程序指令装置。还应该理解，可由执行指定功能和步骤的基于专用硬件的计算机系统或者通过专用硬件和计算机指令的组合来实现流程图示例中的每个块和流程图示例中的块的组合。

图 5 概述了一种随机选择用于启动多处理器系统的启动处理器的示范性操作。如图 5 所示，该操作从系统控制器执行加电复位（POR）操作开始（步骤 510）。在执行初始 POR 操作之后，系统控制器向多处理器系统的普适逻辑提供“电源正常”信号（步骤 520），以及普适逻辑初始化一个随机启动操作（步骤 530）。

普适逻辑从多个处理器中随机地选择一个处理器作为启动处理器（步骤 540）。然后普适逻辑根据该随机选择来设置处理器的配置位（步骤 550）并利用信号通知处理器开始启动操作（步骤 560）。快速 ROM 向处理器提供加密的启动代码，而且从秘密密钥存储器和随机密钥发生器向处理器提供密钥值（步骤 570）。然后处理器根据它们的配置位的设置来选择将由处理器使用的密钥（步骤 580）。处理器试图根据所选择的密钥来对启动代码进行解密（步骤 590）。该选择的处理器利用秘密密钥对启动代码进行解密并启动该系统（步骤 595）。应当注意，由所有其他非选择的处理器解密启动代码的试图将会失败，而且只有该选择的处理器能够启动该系统，然后操作结束。

图 6 是概述了根据一个示范性实施例对启动代码序列进行掩码的示范性操作的流程图。举例来说，可以在多处理器系统的每个处理器中执行图 6 所述的操作。

如图 6 所示，处理器接收信号以开始启动操作（步骤 610）。举例来说，该步骤可以与图 5 中的步骤 530 相对应。处理器试图对启动代码进行解密（步骤 620），而且判断该解密意图是否失败（步骤 630）。如果解密成功，即该处理器是随机选择的启动处理器，则执行启动代码，以由此将多处理器系统启动到操作状态（步骤 640）。

如果解密失败，则选择执行一个代码序列来对启动序列进行掩码（步骤650）。如上所述，取决于特定实施例，可以根据本地存储器的安全部分中的默认代码序列、随机选择的开始地址、对具有随机延迟元素的启动代码的使用等等来进行对掩码代码序列的选择。运行掩码代码序列（步骤660），而且判断该系统是否处于操作状态，即是否完成了启动序列（步骤670）。如果否，该操作就返回到步骤660并继续运行掩码代码序列。如果该系统处于操作状态，则结束掩码代码序列的执行（步骤680），并且操作终止。

因此，上述示范性实施例提供了一种机制，利用该机制可以从多个处理器中随机地选择一个处理器作为启动处理器，用以将多处理器系统启动到操作状态。该示范性实施例还提了一种机制，用于对随机选择的处理器正执行的启动代码序列进行掩码，以便使入侵者难以辨识已经随机选择了哪个处理器来执行实际启动代码序列。利用这些机制，通过使入侵者极难通过监控启动代码序列来实现对系统的访问而使多处理器系统更加安全。

相对于多处理器系统中单个处理器正执行的启动代码序列描述了上述示范性实施例。然而，示范性实施例并不局限于此。如以下所描述的，在其他示范性实施例中，可以在多处理器系统中的多个处理器上来分配启动代码序列。通过在多处理器系统中的多个处理器上分配启动代码序列，增加了必须被危害以便获得关于启动序列的完整信息从而避开安全性测量的处理器的数目。

因此，以下所描述的示范性实施例的分布式启动操作比利用单个安全核的多处理器数据处理系统更为安全。此外，通过分布启动操作，如果危害了启动操作的任何部分，则启动操作失败，从而防止未授权个体规避系统的安全性。换言之，尽管未来的入侵者可以危害启动操作的部分，但是该入侵者不能危害全部启动操作，因此不能获得对多处理器数据处理系统的访问。

利用该示范性实施例，将启动代码序列划分成多个分区，以便可以将每个分区提供给多处理器系统的不同的处理器。当执行启动代码

序列的每个分区时，在启动代码序列可以在另一个处理器上进行之前，该分区必须在其各自的处理器上正确地完成。利用安全通信机制来传达对启动代码序列的先前分区的满意完成。该安全通信机制可以包括一个安全令牌，例如加密的密码或其他安全标识符，例如公钥/私钥对，其表示先前的会话没有被危害。按照这种方式，创建必须令人满意地完成的相关“会话”链。

启动代码的分布式执行中所涉及的处理器可以是多处理器系统中的所有处理器或者是多处理器系统中的处理器子集。例如，可以按分布方式利用随机选择机制（例如上述用于选择单个启动处理器的随机选择机制）来随机地选择将用于启动系统的多个启动处理器。而且，可以随机地选择由处理器所执行的启动代码的特定分区，使得利用每个加电复位(POR)操作，相同的处理器可以执行或不执行与之前 POR 操作中相同的启动代码分区。因此，可以针对分布式启动操作中涉及哪些处理器以及每个处理器将执行哪些启动代码分区来进行随机化。

多处理器系统的其他处理器（即，非启动处理器）可以在分布式启动操作期间不执行任何操作，或者可以执行上述各种掩码代码示范性实施例中的一个或多个掩码代码序列，以此对随机选择的处理器子集上的启动代码执行进行掩码。换言之，在不脱离本发明精神和范围的情况下，该示范性实施例的分布式启动代码序列操作可以与之前所述的一个或多个示范性实施例相结合。

图 7A 是描述根据一个示范性实施例配置为菊链或环形布置的分布式启动操作的示意图。如图 7A 所示，提供多个处理器 720-750，用于启动多处理器数据处理系统。在所述实例中，在分布式启动操作中利用了所有协处理器，即 SPE，而控制处理器（例如 PPE）不执行该分布式启动代码。当然，在其他示范性实施例中，PPE 也可以包括在分布式启动操作中。而且，在其他示范性实施例中，正如之前所述的，只有多处理器数据处理系统中的处理器子集可以用来执行分布式启动操作。

举例来说，可以将加密的启动代码 710 划分成可单独执行的分区，

即启动代码分区 1 至 n，其中该加密的启动代码 710 可以存储在与多处理器数据处理系统相关联的存储设备中，例如图 2 中的快速 ROM 230。例如，可以提供这些分区作为加密的启动代码中的模块或子程序，其中利用相同的加密算法和相同的秘密密钥（Skey）对它们分别进行加密。优选地，启动代码分区的数目等于分布式启动操作中将涉及的处理器的数目，即启动处理器的数目。然而，在某些示范性实施例中，例如在环形布置的启动处理器中，启动代码分区的数目并不局限于启动处理器的数目，而可以是少于或多于启动处理器数目的任意多个分区。

举例来说，在普适逻辑 790 的控制下执行分布式启动操作，该普适逻辑 790 可以是图 1 中相同的普适逻辑 193。举例来说，通过随机事件发生器的用户，普适逻辑 790 可以随机地选择将用作启动处理器的处理器 720-750，以及可以随机地选择每个随机选择的处理器 720-750 将执行哪个分区。在这一实施例中，普适逻辑 790 可以跟踪启动代码分区要执行的次序，以便利用安全通信机制来确保启动代码序列的安全性，其中安全通信机制指示分布式启动操作的前一个会话是否已经被危害。然而，为了该描述的简单起见，将假设：在所述实例中，在分布式启动操作中利用了多处理器系统的所有处理器或至少协处理器，而且按相继次序向处理器 720-750 提供启动代码分区。

普适逻辑 790 向处理器 720-750 提供选择器信号，用以选择每个处理器 720-750 将要执行哪个启动代码分区。另外，普适逻辑 790 提供密钥值选择器信号，用以使处理器 720-750 从 Skey 存储器中选择 Skey 作为将用来对处理器 720-750 的相应启动代码分区进行解密的密钥。举例来说，处理器 720-750 利用所提供的 Skey 对它们的启动代码分区进行解密，以及然后借助于按照菊链体系结构的处理器 720-750 的布置或者在普适逻辑 790 的控制下，以合适的序列执行启动代码分区。

在所述实例中，SPE0 720 通过以下步骤开始分布式启动操作：对其启动代码分区 1 进行解密，执行该启动代码分区，然后向 SPE1 730

安全地传达启动代码分区 1 的成功完成。而且，可以在这些 SPE 之间利用安全机制，用以表示没有危害先前的会话，即，包括先前的启动代码分区的执行的会话。举例来说，该安全机制可以传送安全令牌、数字签名、密码、先前启动代码分区的校验和，利用成功完成消息的公钥/私钥加密，等等。任何可以用来传达是否危害了分布式启动操作的先前会话的安全机制都旨在本发明的精神和范围内。

在接收到启动代码分区 1 执行的成功和未危害的确认之后，SPE1 730 可以对它的启动代码分区 2 进行解密，执行启动代码分区，然后向 SPE2 740 传达其成功完成启动代码分区 2。该处理可以继续进行，直至利用信号通知了所有处理器它们已经完成了它们的分布式启动操作部分而没有被危害。在这种启动代码分区相关链中的任何中断（例如利用信号通知未成功的执行或者被危害的执行）都会导致启动失败，其中可以利用信号通知系统控制器启动失败。一旦成功地完成了所有启动代码分区，该多处理器数据处理系统就处于操作状态，其中可以在各个处理器上执行软件应用。

上述示范性实施例利用了针对启动代码分区的处理器的菊链布置，其中在处理器上执行该启动代码分区。在不脱离本发明的精神和范围的情况下可以使用确保连续执行启动代码分区的其它布置。例如，对上述菊链布置的扩展是提供一种针对分布式启动操作的处理器的环形布置，以便最后一个的处理器（例如，SPE7 750）反过来向第一个处理器（例如，SPE0 720）传达其启动代码分区的执行的成功的和未受危害的完成，其中选择第一个处理器（例如，SPE0 720）作为“主要”启动处理器。按照这种方式，可以在主要启动处理器处使用通过环形布置从一个会话传送到下一个会话的安全机制来对整个分布式启动操作的未被危害的执行进行检验，验证机制例如为安全令牌、增加的计数值等等。

而且，处理器的环形布置允许利用比启动处理器的数目多的启动代码分区。因此，如果只选择多处理器数据处理系统中的处理器子集作为启动处理器，则该处理器子集可以在针对分布式启动操作设置成

的环形布置时执行任意数目个启动代码分区。这不仅提供了普适逻辑790在多处理器数据处理系统中随机地选择哪些处理器将成为启动处理器的能力，还提供了普适逻辑790随机地选择多少个处理器将在分布式启动操作中作为启动处理器的能力。因此，在第一POR操作中，可以选择四个处理器作为启动处理器，而在随后的POR操作中，可以选择三个启动处理器。普适逻辑790可以包含用于随机地选择多个处理器来将其选择为启动处理器的逻辑，然后如上所述，利用该逻辑来控制对处理器的随机选择。

另一种针对分布式启动操作的可能的启动处理器布置是提供了一种主/从布置。图7B是描述根据一个示范性实施例配置为主/从布置的分布式启动操作的示意图。如图7B所示，指定处理器760为主机处理器。该处理器可以是其中的一个协处理器（例如，SPE）或控制处理器（例如，PPE）。每个从处理器（例如，SPE0-SPE7 720-750）都负责以类似于图7A中所述的方式来完成它们的启动代码分区并向主核安全地传达它们已经完成了执行和没有被危害。一旦主处理器760接收了来自每个从处理器720-750的信号，并确认它本身没有被危害，则允许多处理器数据处理系统进入其中可以执行软件应用的操作状态。

应当理解，尽管在这里描述了针对分布式启动操作的处理器菊链布置、环形布置和主/从布置，但是本发明并不局限于这些所述的布置。相反，在不脱离本发明的精神和范围的情况下，任何针对分布式启动操作的处理器布置都可以与示范性实施例的机制一起使用。

图8是概述了根据一个示范性实施例的用于多处理器系统的分布式启动的示范性操作的流程图。如图8中所示，该操作从普适逻辑接收来自于系统控制器的“电源正常”信号开始（步骤810）。普适逻辑从多处理器数据处理系统中的多个处理器中选择处理器作为启动处理器（步骤820）。如上所述，这种选择会导致选择所有处理器或选择多处理器数据处理系统中的处理器子集作为启动处理器。举例来说，可以利用普适逻辑中的随机事件发生器来执行这种选择。

普适逻辑选择将要分配给所选择的启动处理器的启动代码分区（步骤 830）。由相关联的启动处理器执行下一个启动代码分区（步骤 840）。启动处理器判定启动代码分区的执行是否成功和没有被危害（步骤 850）。如果否，则利用信号通知系统控制器启动失败（步骤 860），并且操作结束。

如果启动代码分区成功地执行并且没有被危害，则启动处理器判定是否已经成功地执行了所有启动代码分区（步骤 870）。如果否，则操作返回步骤 840，并由其相关联的启动处理器来执行下一个启动代码分区。如果成功地执行了所有的启动代码分区，则启动处理器利用信号通知系统控制器数据处理系统的成功启动（步骤 880），并且操作结束。

因此，如上所述，除了随机地选择单个启动处理器并对多处理器数据处理系统中的其他处理器上的操作进行掩码之外，示范性实施例还提供了用以在多个处理器上分布启动操作的机制。示范性实施例提供了用于以下内容的机制：随机地选择启动处理器；随机地选择要在所选择的启动处理器上执行的启动代码分区；以确保各个启动处理器执行启动代码分区的安全性。所有的这些机制旨在提高多处理器数据处理系统的安全性，保护该系统不受启动操作的未授权监控的威胁。

示范性实施例可以采用完全硬件的实施例、完全软件的实施例或包含硬件和软件元素的实施例的形式。在优选的实施例中，可以通过软件来实现本发明，它包括但不局限于固件、常驻软件、微代码等。

此外，示范性实施例可以采用可通过计算机可用或计算机可读介质访问的计算机程序产品的形式，该介质提供为计算机或任何指令执行系统所使用或与其结合使用的程序代码。为了本描述目的，计算机可用或计算机可读介质可以是任何包含、存储、传送、传播或传输程序的装置，该程序供指令执行系统、装置或设备使用或与其结合使用。

该介质可以是电、磁、光学、电磁、红外线或半导体系统（或装置或设备）或者传播介质。计算机可读介质的实例包括半导体或固态存储器、磁带、可移动计算机盘、随机存取存储器（RAM）、只读存

储器（ROM）、硬磁盘和光盘。光盘的当前实例包括致密盘-只读存储器（CD-ROM）、致密盘-读/写（CD-R/W）和 DVD。

上述电路可以是集成电路芯片设计的一部分。可以利用图形计算机编程语言来创建芯片设计，并在计算机存储介质（例如盘、磁带、物理硬盘驱动器或诸如存储访问网络中的虚拟硬盘驱动器）中存储该芯片设计。如果设计者不制造芯片或用来制造芯片的光刻掩模，则设计者就可以直接或间接地通过物理装置（例如，通过提供存储该设计的存储介质的副本）或电子地（例如，通过因特网）向这类实体发送生成的设计。然后可以将所存储的设计转换成合适的形式（例如，GDSII），用于光刻掩模的制造，光刻掩模通常包括将在晶片上形成的所讨论的芯片设计的多个副本。可以利用光刻掩模来限定将要蚀刻或进行处理的晶片区域（和/或其上的层）。

可由制造者按原始晶片的形式来分布生成的集成电路芯片（也就是，作为具有多个非封装芯片的单个晶片）作为裸模，或者按照封装的形式来分布生成的集成电路芯片。在后一种情况中，可以将芯片安装在单芯片封装（例如，塑性载体，利用附着到主板或其他高层载体的引线）或多芯片封装（例如，陶瓷载体，它具有表面互连结构和埋藏的互连结构）内。然后，在任何情况下芯片都可以与其他芯片、分立电路元件和/或其他作为（a）中间产品（例如主板）或（b）最终产品的一部分的信号处理装置进行集成。最终产品可以是任何包括集成电路芯片的产品，即从玩具和其他低端应用到具有显示器、键盘或其他输入设备以及中央处理器的高级计算机产品。而且，在其中可以配备集成电路芯片的最终产品可以包括游戏机、游戏控制台、手持计算设备、个人数字助理、诸如无线电话等的通信装置、膝上型计算设备、桌面型计算设备、服务器计算设备或任何其他计算设备。

为了描述和说明目的已经提出了本发明的说明，而本发明的说明并非旨在穷尽性的或着将本发明限制为所公开的形式。许多修改和变化对本领域普通技术人员来说是明显的。选择和说明了实施例，以便最佳地解释本发明地原理、特定应用，而且使本领域其他技术人员

能够理解针对具有各种修改的各种实施例的本发明，正如所述各种修改适合于所考虑的特定用途。

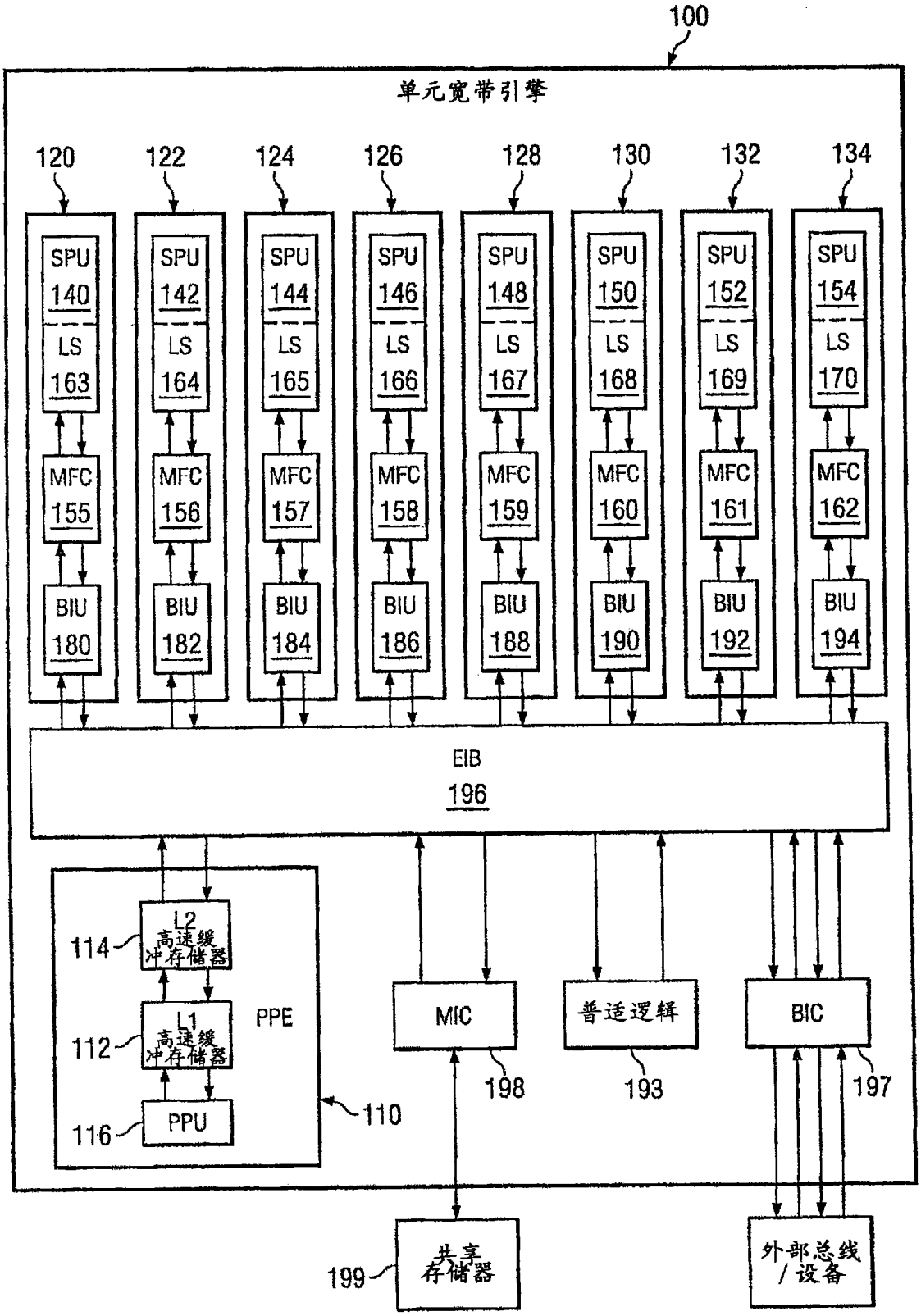

100
单元宽带引擎
120
122
124
126
128
130
132
134
SPU 140
SPU 142
SPU 144
SPU 146
SPU 148
SPU 150
SPU 152
SPU 154
LS 163
LS 164
LS 165
LS 166
LS 167
LS 168
LS 169
LS 170
MFC 155
MFC 156
MFC 157
MFC 158
MFC 159
MFC 160
MFC 161
MFC 162
BIU 180
BIU 182
BIU 184
BIU 186
BIU 188
BIU 190
BIU 192
BIU 194
EIB 196
L2 高速缓冲存储器
L1 高速缓冲存储器
PPU
PPE
114
112
116
110
MIC
198
普适逻辑
193
BIC
197
199
共享存储器
外部总线/设备

图 1

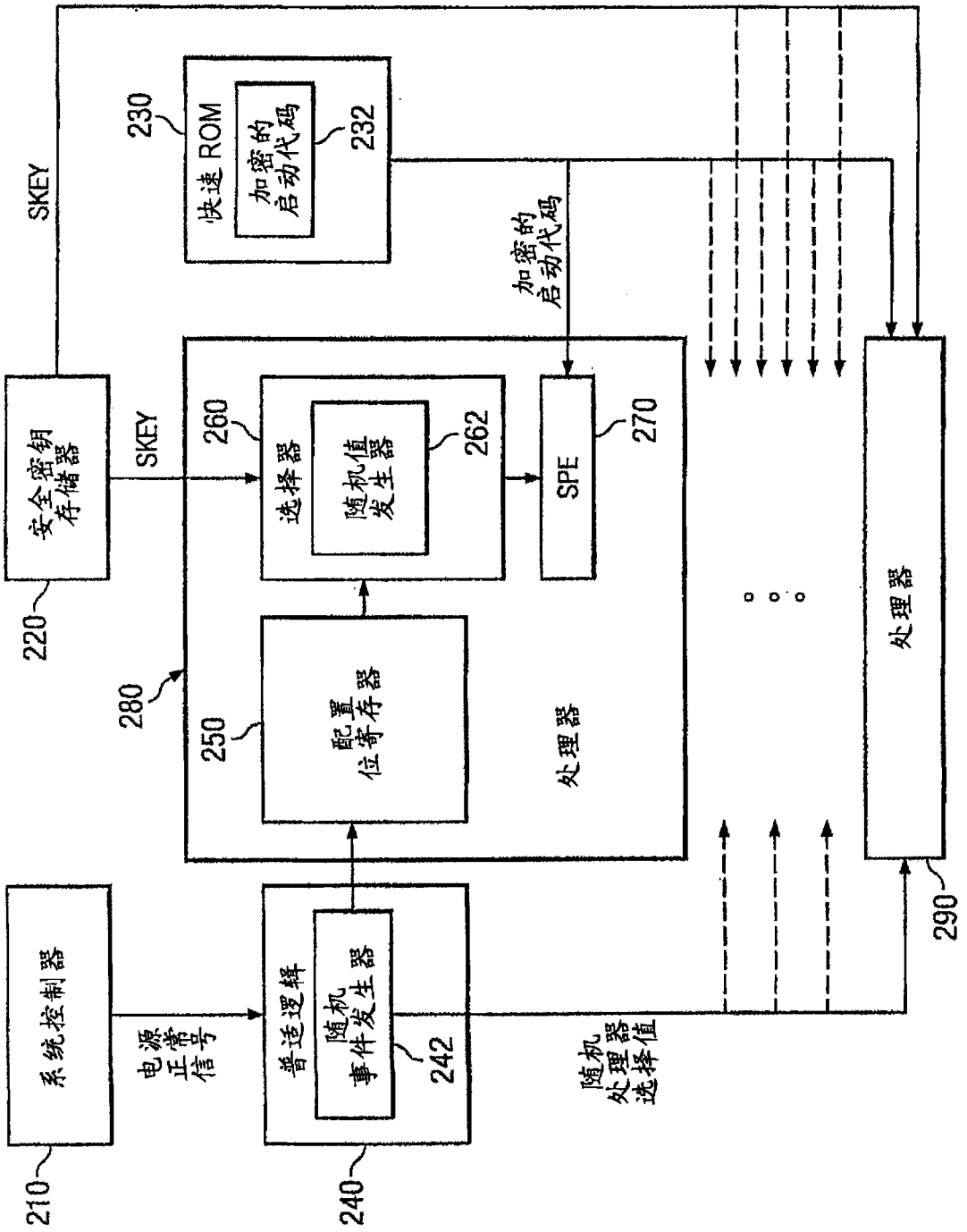

SKEY
230
快速 ROM
加密的启动代码
232
加密的启动代码
安全密钥存储器
SKEY
260
选择器
随机值发生器
262
SPE
270
220
280
250
配置寄存器
处理器
处理器
系统控制器
电源正常信号
普适逻辑
随机事件发生器
242
随机处理器选择值
210
240
290

图 2

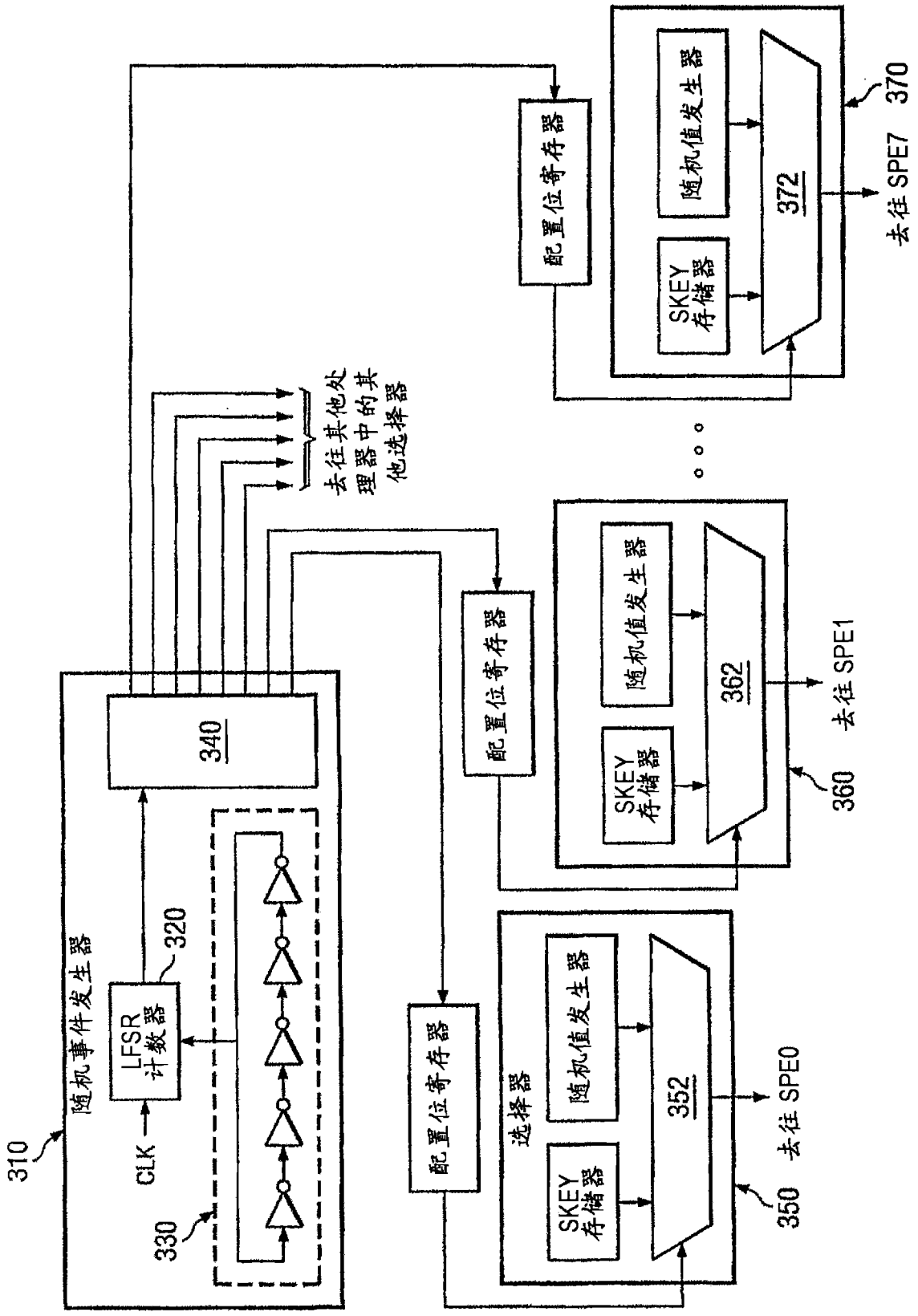

310
随机事件发生器
LFSR
计数器
320
CLK
330
340
去往其他处
理器中的其
他选择器
配置位寄存器
SKEY
存储器
随机值发生器
372
370
去往 SPE7
362
360
去往 SPE1
选择器
352
350
去往 SPE0

图 3A

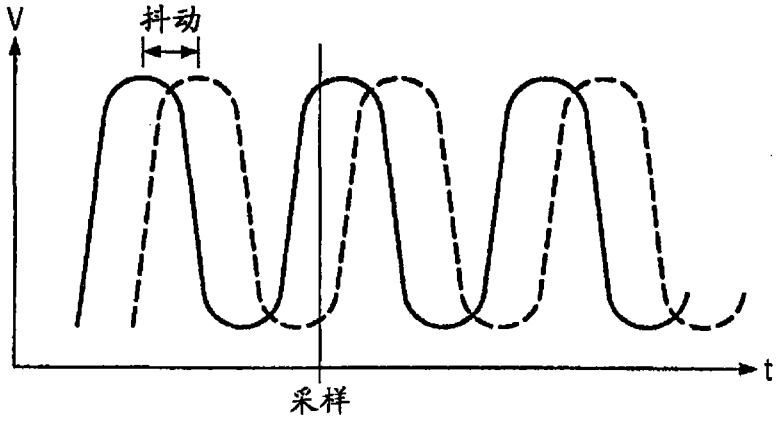

V
抖动
t
采样

图 3B

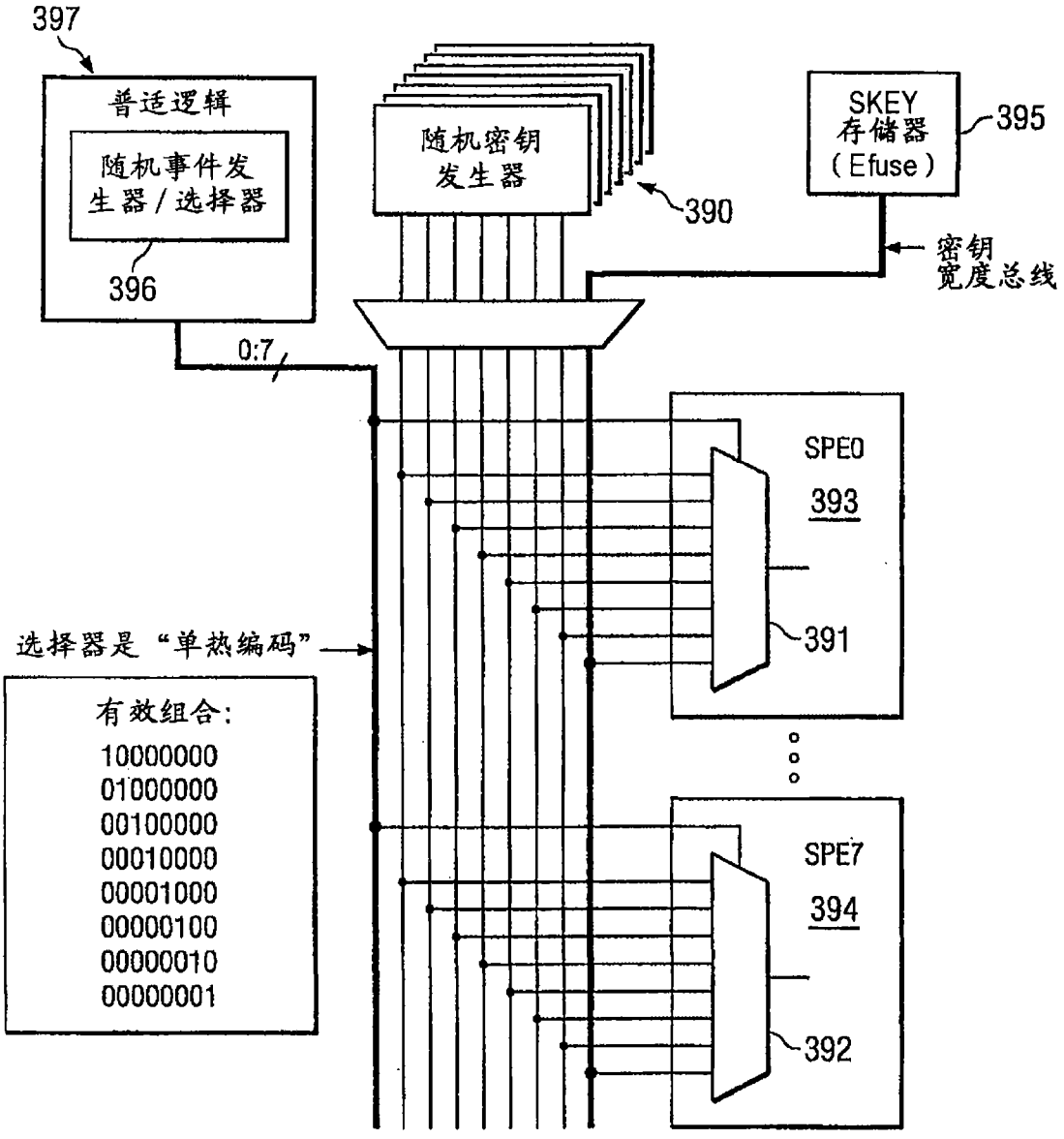

397
普适逻辑
随机事件发生器/选择器
396
随机密钥发生器
390
SKEY存储器(Efuse)
395
密钥宽度总线
0:7
SPE0
393
391
选择器是“单热编码”
有效组合:
10000000
01000000
00100000
00010000
00001000
00000100
00000010
00000001
SPE7
394
392

图 3C

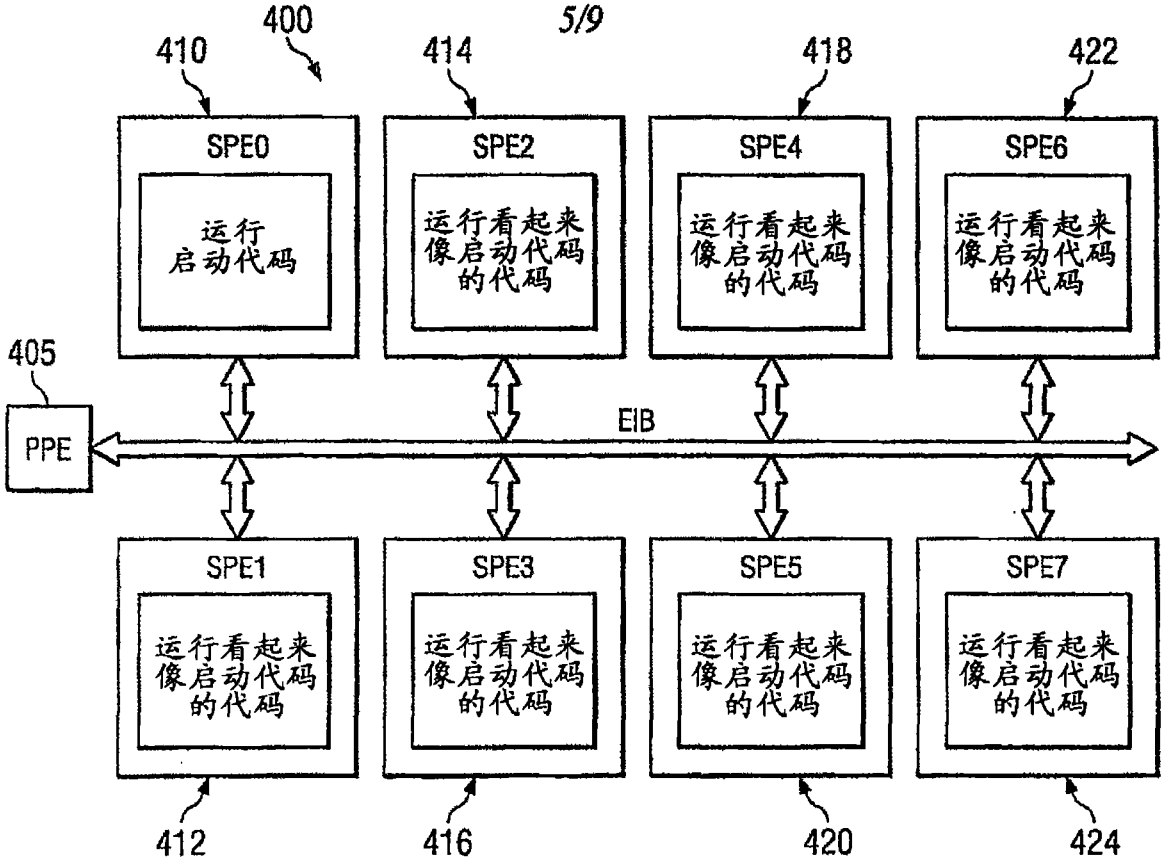

400
410
414
418
422
SPE0
运行
启动代码
SPE2
运行看起来像启动代码的代码
SPE4
运行看起来像启动代码的代码
SPE6
运行看起来像启动代码的代码
405
PPE
EIB
SPE1
运行看起来像启动代码的代码
SPE3
运行看起来像启动代码的代码
SPE5
运行看起来像启动代码的代码
SPE7
运行看起来像启动代码的代码
412
416
420
424

图 4A

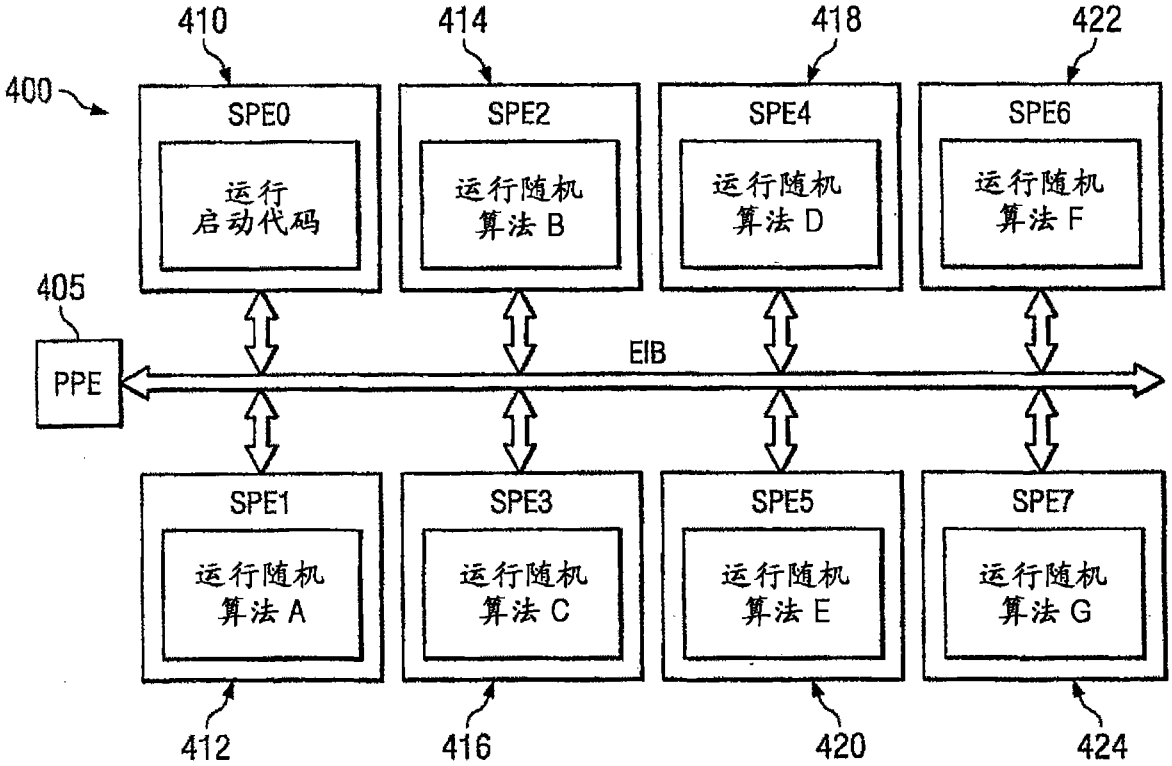

410
414
418
422
400
SPE0
运行
启动代码
SPE2
运行随机
算法 B
SPE4
运行随机
算法 D
SPE6
运行随机
算法 F
405
PPE
EIB
SPE1
运行随机
算法 A
SPE3
运行随机
算法 C
SPE5
运行随机
算法 E
SPE7
运行随机
算法 G
412
416
420
424

图 4B

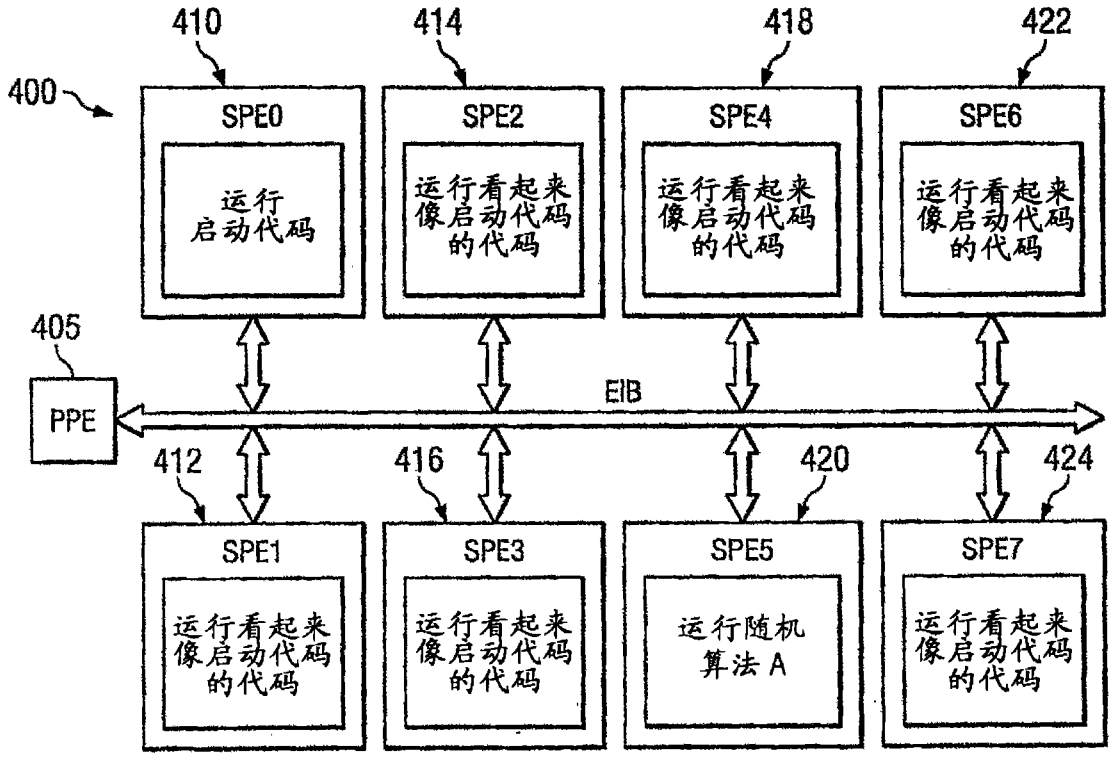

410
414
418
422
400
SPE0
运行启动代码
SPE2
运行看起来像启动代码的代码
SPE4
运行看起来像启动代码的代码
SPE6
运行看起来像启动代码的代码
405
PPE
EIB
412
416
420
424
SPE1
运行看起来像启动代码的代码
SPE3
运行看起来像启动代码的代码
SPE5
运行随机算法 A
SPE7
运行看起来像启动代码的代码

图 4C

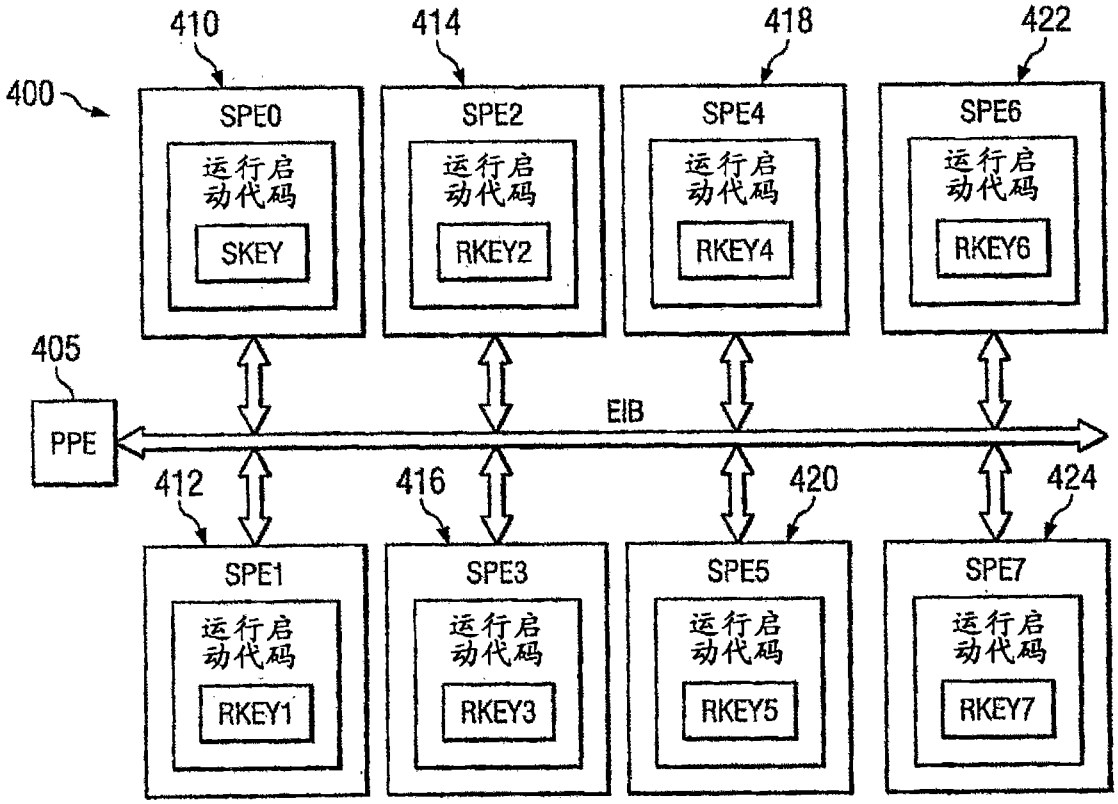

410
414
418
422
400
SPE0
运行启动代码
SKEY
SPE2
运行启动代码
RKEY2
SPE4
运行启动代码
RKEY4
SPE6
运行启动代码
RKEY6
405
PPE
EIB
412
416
420
424
SPE1
运行启动代码
RKEY1
SPE3
运行启动代码
RKEY3
SPE5
运行启动代码
RKEY5
SPE7
运行启动代码
RKEY7

图 4D

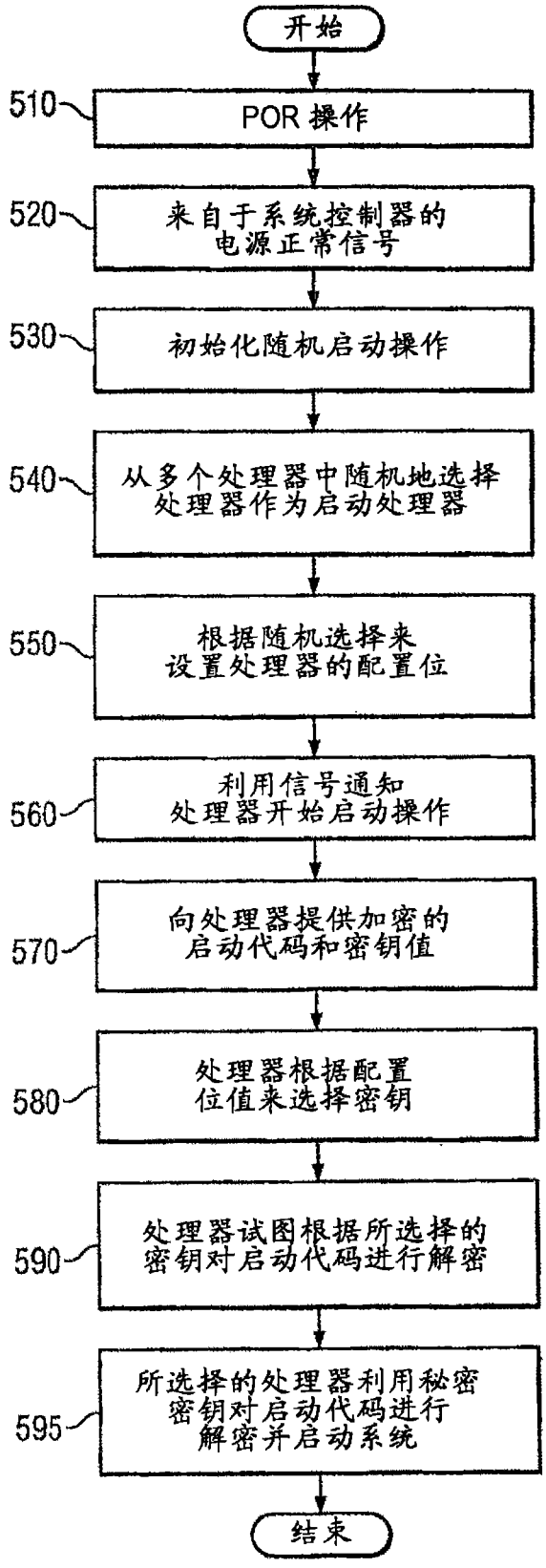

开始
510
POR 操作
520
来自于系统控制器的
电源正常信号
530
初始化随机启动操作
540
从多个处理器中随机地选择
处理器作为启动处理器
550
根据随机选择来
设置处理器的配置位
560
利用信号通知
处理器开始启动操作
570
向处理器提供加密的
启动代码和密钥值
580
处理器根据配置
位值来选择密钥
590
处理器试图根据所选择的
密钥对启动代码进行解密
595
所选择的处理器利用秘密
密钥对启动代码进行
解密并启动系统
结束

图 5

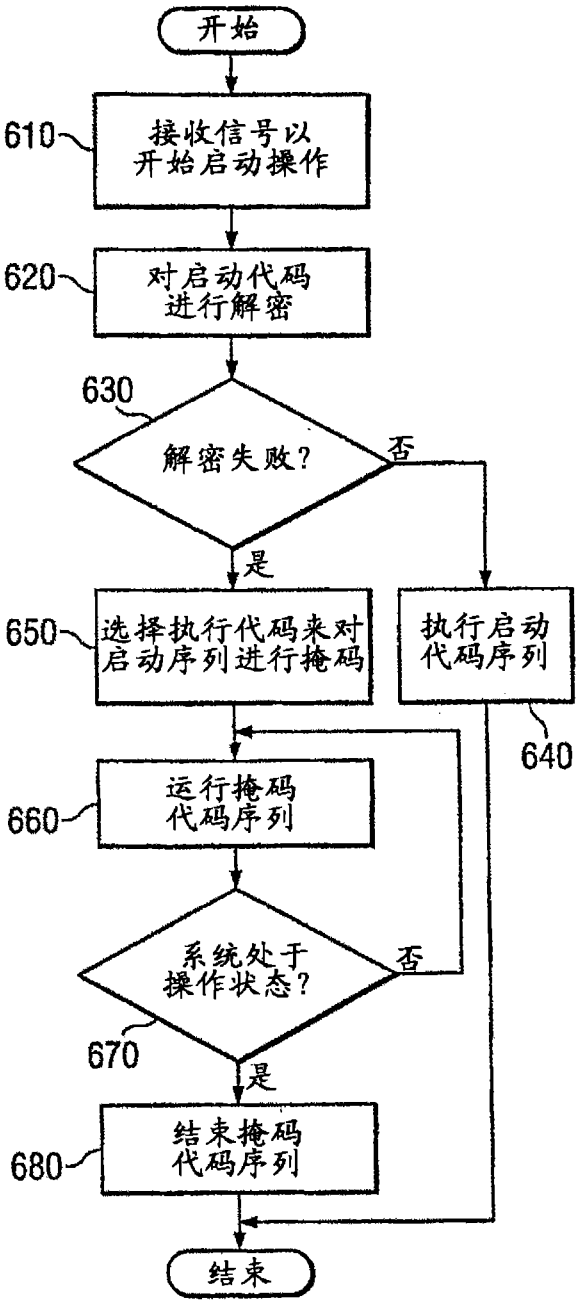

开始
610
接收信号以
开始启动操作
620
对启动代码
进行解密
630
解密失败?
否
是
650
选择执行代码来对
启动序列进行掩码
执行启动
代码序列
640
660
运行掩码
代码序列
系统处于
操作状态?
否
670
是
680
结束掩码
代码序列
结束

图 6

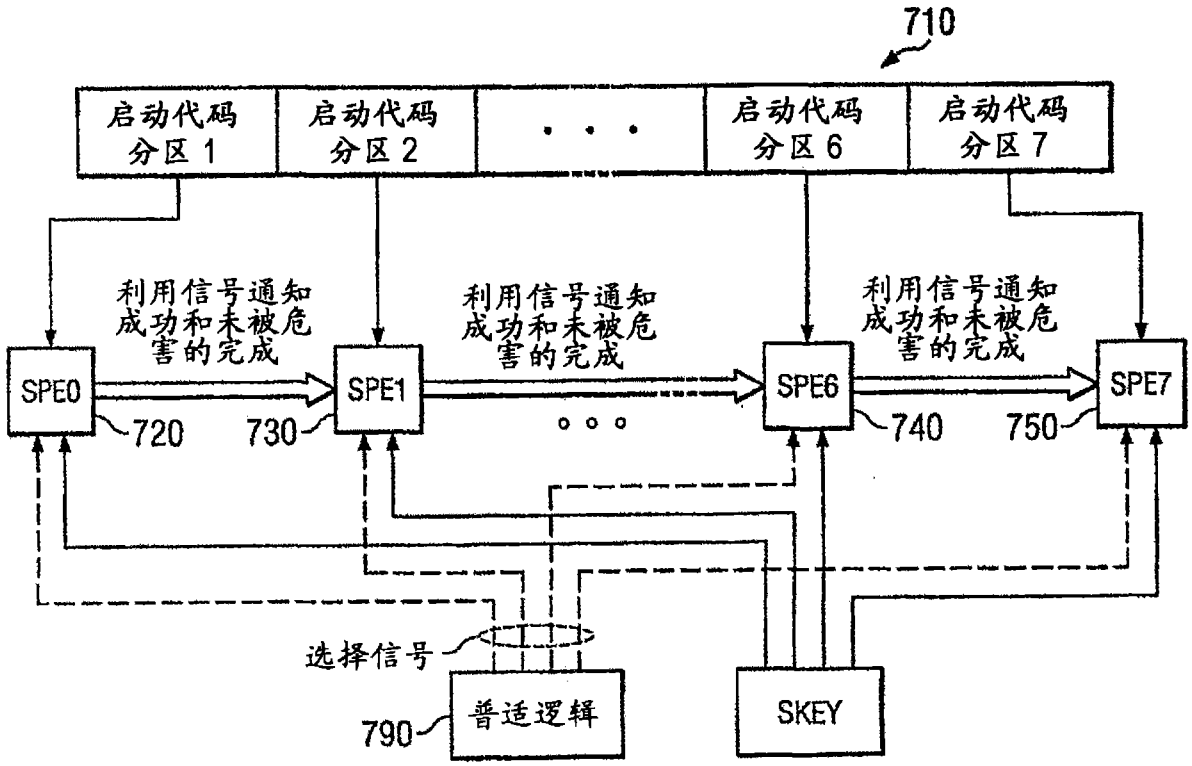

710
启动代码分区 1
启动代码分区 2
. . .
启动代码分区 6
启动代码分区 7
利用信号通知成功和未被危害的完成
利用信号通知成功和未被危害的完成
利用信号通知成功和未被危害的完成
SPE0
SPE1
SPE6
SPE7
720
730
740
750
选择信号
790
普适逻辑
SKEY

图 7A

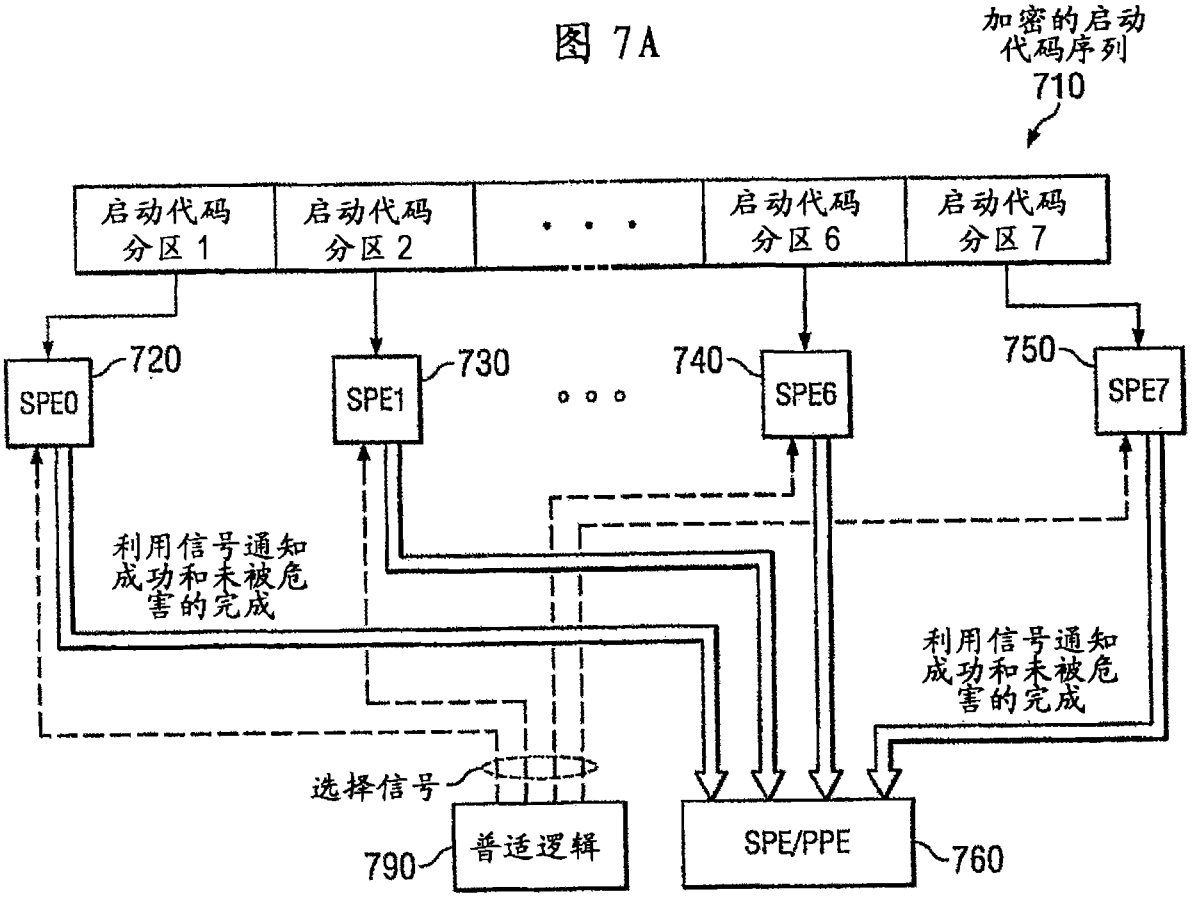

加密的启动代码序列
710
启动代码分区 1
启动代码分区 2
. . .
启动代码分区 6
启动代码分区 7
720
730
740
750
SPE0
SPE1
SPE6
SPE7
利用信号通知成功和未被危害的完成
利用信号通知成功和未被危害的完成
选择信号
790
普适逻辑
SPE/PPE
760

图 7B

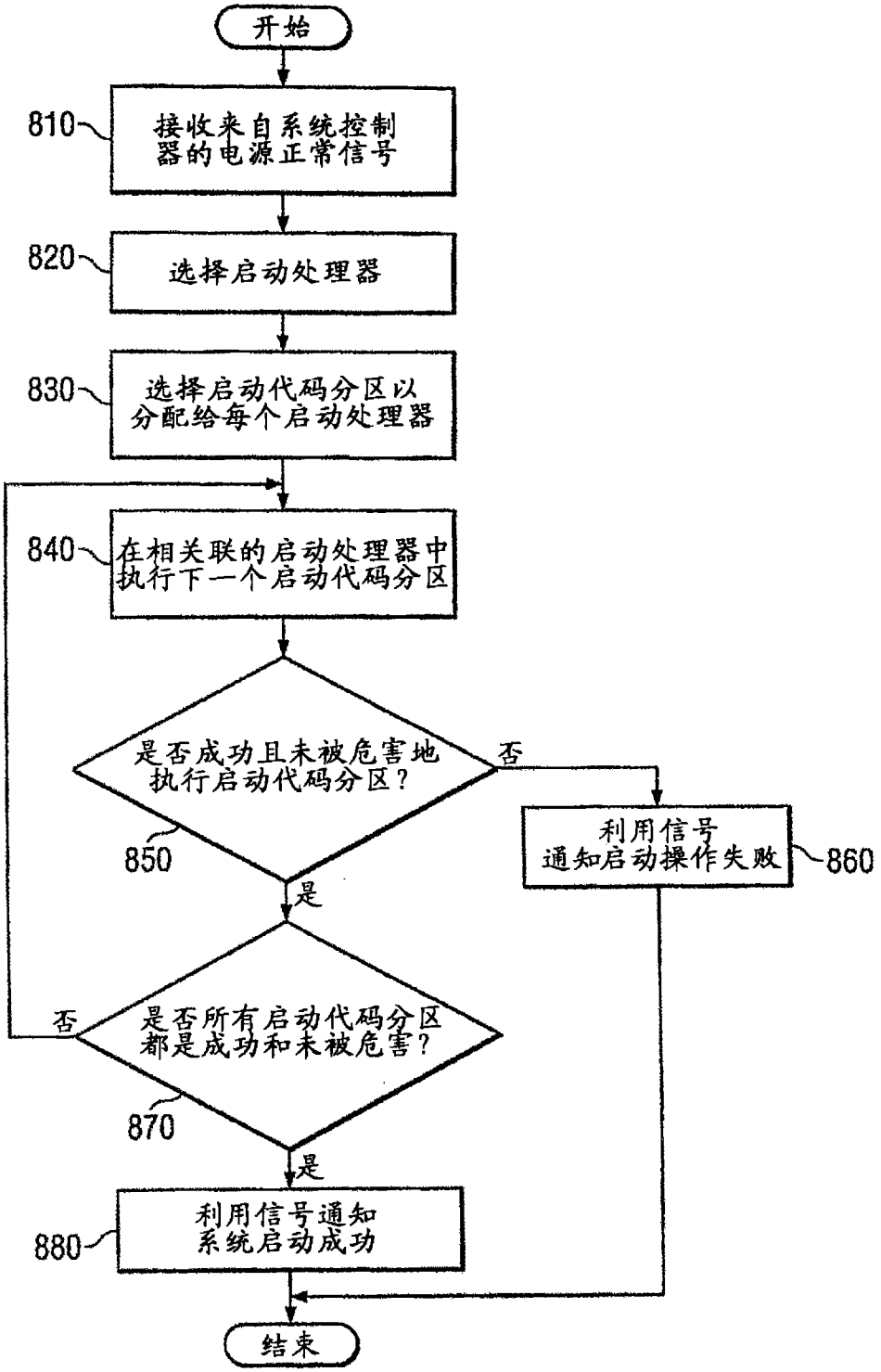

开始
810
接收来自系统控制器的电源正常信号
820
选择启动处理器
830
选择启动代码分区以分配给每个启动处理器
840
在相关联的启动处理器中执行下一个启动代码分区
是否成功且未被危害地执行启动代码分区？
850
否
利用信号通知启动操作失败
860
是
是否所有启动代码分区都是成功和未被危害？
870
否
是
利用信号通知系统启动成功
880
结束

图 8

1. 一种与具有多个处理器的数据处理系统结合使用的用于启动所述数据处理系统的方法，包括:

将启动代码划分为多个启动代码分区;

在所述数据处理系统的所述多个处理器中的多个启动处理器的每一个中从所述多个启动代码分区中载入启动代码分区;

将所述多个启动代码分区作为多个会话在其各自相关联的启动处理器上执行，由此启动所述数据处理系统;以及

从所述多个处理器中随机选择所述启动处理器，其中所述启动处理器是所述多个处理器的子集。

2. 根据权利要求 1 所述的方法，其中，当每个启动代码分区在其相关联的会话中的执行完成时，执行所述启动代码分区的所述多个处理器中的相关联启动处理器可进行操作以通过信号将所述启动代码分区的成功完成告知与启动代码序列中的下一启动代码分区相关联的另一启动处理器。

3. 根据权利要求 1 或权利要求 2 所述的方法，其中，在其各自相关联的启动处理器上执行所述多个启动代码分区包括:

在启动处理器之间的通信中使用安全机制，以确保所述多个启动代码分区在其各自相关联的启动处理器上的非折衷执行。

4. 根据权利要求 3 所述的方法，其中，所述安全机制至少包括以下各项中的一个:在启动处理器之间传递安全令牌，在启动处理器之间传递数字签名，使用密码，传递启动代码分区的校验和，或者使用信号的公开密钥/私有密钥加密。

5. 根据前述权利要求中任一项所述的方法，其中，启动代码分区的数目等于启动处理器的数目。

6. 根据前述权利要求中任一项所述的方法，进一步包括:

在未被随机选作启动处理器的所述多个处理器的处理器上执行掩码。

7. 根据前述权利要求中任一项所述的方法，进一步包括:

随机选择哪个启动代码分区与每个启动处理器相关联，其中每个启动代码分区可以不同于其他启动代码分区。

8. 根据前述权利要求中任一项所述的方法，其中，用于所述启动处理器上的启动代码分区执行的会话布置为菊花链布置、环形布置或者主/从布置之一。

9. 一种与包含多个处理器的数据处理系统结合使用的用于启动所述数据处理系统的设备，包括:

用于将启动代码划分为多个启动代码分区的装置;

用于在所述数据处理系统的所述多个处理器中的多个启动处理器的每一个中从所述多个启动代码分区中载入启动代码分区的装置;

用于将所述多个启动代码分区作为多个会话在其各自相关联的启动处理器上执行以由此启动所述数据处理系统的装置; 以及

用于从所述多个处理器中随机选择所述启动处理器的装置，并且其中所述启动处理器是所述多个处理器的子集。

10. 根据权利要求9所述的设备，进一步包括: 用于当每个启动代码分区在其各自相关联的会话中的执行完成时，通过执行所述启动代码分区的所述多个处理器中的相关联启动处理器利用信号将所述启动代码分区的成功完成告知与启动代码序列中的下一启动代码分区相关联的另一启动处理器。

11. 根据权利要求9或者权利要求10所述的设备，进一步包括: 用于在启动处理器之间的通信中使用安全机制以确保所述多个启动代码分区的非折衷执行的装置。

12. 根据权利要求11所述的设备，其中，所述安全机制至少包括以下各项中的一个: 在启动处理器之间传递安全令牌，在启动处理器之间传递数字签名，使用密码，传递启动代码分区的校验和，或者使用信号的公开密钥/私有密钥加密。

13. 根据权利要求 9 到 12 任一项所述的设备，其中，启动代码分区的数目等于启动处理器的数目。

14. 根据权利要求9到13任一项所述的设备，进一步包括：用于在未被随机选作启动处理器的所述多个处理器的处理器上执行掩码的装置。

15. 根据权利要求9到14任一项所述的设备，进一步包括：用于随机选择哪个启动代码分区与每个启动处理器相关联的装置，并且其中每个启动代码分区可以不同于其他启动代码分区。

16. 根据权利要求9到15任一项所述的设备，其中，用于所述启动处理器上的启动代码分区执行的会话被布置为菊花链布置、环形布置或者主/从布置之一。

17. 一种包括程序代码装置的计算机程序，当所述程序在计算机上运行时，所述程序代码装置适于执行根据权利要求1到8任一项所述的步骤。