(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 101539875 B
(45)授权公告日 2012.04.11

(21)申请号 200910132513.7

(22)申请日 2000.10.04

(30)优先权数据
09/469963 1999.12.21 US

(62)分案原申请数据
00819142.5 2000.10.04

(73)专利权人 英特尔公司
地址 美国加利福尼亚州

(72)发明人 N·T·夸克

(74)专利代理机构 中国专利代理(香港)有限公司 72001
代理人 张雪梅 刘春元

(51)Int.Cl.
G06F 11/16(2006.01)

(56)对比文件
US 4453215 A,1984.06.05, 说明书第31栏到第32栏,图6.
EP 0411805 A3,1991.02.06, 全文.

审查员 张行素

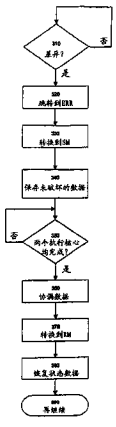
权利要求书 1 页 说明书 11 页 附图 8 页

(54)发明名称

纠正软错误的固件机制

(57)摘要

本发明涉及纠正软错误的固件机。本发明提供一种计算机系统,包括有双执行核心的处理器和存储错误恢复程序的非易失性存储器。当处理器处于冗余执行模式时处理器的执行核心以锁定步骤操作,当处理器处于分离执行模式时它们独立地操作。当处理器在以冗余执行模式执行的同时检测到软错误时错误恢复程序被调用。错误恢复程序切换处理器到分离执行模式。在分离模式下,每个执行核心保存未损坏的处理器状态数据到指定的存储单元并用来自另一个执行核心的相应的处理器状态数据更新任何已损坏的数据。错误恢复程序返回处理器到冗余模式,用恢复的处理器状态数据初始化每个执行核心,并返回处理器的控制到检测到软错误时正在执行的程序线程。



CN 101539875 B

1. 一种处理器，包括：

第一执行核心，可在分离模式或冗余模式中操作，所述第一执行核心包括第一指令高速缓存；以及第二执行核心，第二执行核心在分离模式中独立于第一执行核心来处理指令，而在冗余模式中与第一执行核心以锁定步骤处理指令，第二执行核心包括第二指令高速缓存和选择器，所述选择器在冗余模式中从第一指令高速缓存中选择指令而在分离模式中从第二指令高速缓存中选择指令。

2. 权利要求 1 的处理器，还包括：校验单元，用于在第一和第二执行核心处于冗余模式时比较来自第一和第二执行核心的结果以便检测错误。

3. 权利要求 2 的处理器，还包括：与第一执行核心相关联的第一组一个或多个存储结构，用于存储第一执行核心的处理器状态数据；以及与第二执行核心相关联的第二组一个或多个存储结构，用于存储第二执行核心的处理器状态数据。

4. 权利要求 3 的处理器，其中，第一和第二组一个或多个存储结构是受奇偶保护的。

5. 权利要求 2 的处理器，其中，当所述校验单元检测到错误时，第一和第二执行核心切换到分离模式并将未被破坏的处理器状态数据保存到第一和第二组一个或多个存储结构的每一个的不可高速缓存的部分。

纠正软错误的固件机制

[0001] 本发明申请是本发明申请人于2000年10月4日提交的、申请号为00819142.5、发明名称为“纠正软错误的固件机制”的发明申请的分案申请。

[0002] 发明背景

技术领域

[0003] 本发明涉及微处理器，尤其涉及能够以高可靠性模式操作的微处理器。

背景技术

[0004] 当 α 粒子或宇宙射线透过集成电路并更改存储在电路的电压波节上的电荷之时会出现软错误。如果电荷改变足够大，表示一种逻辑状态的电压就可能会被改变为表示不同逻辑状态的电压。例如表示逻辑真状态的电压可能会被改变为表示逻辑假状态的电压，而且包含该逻辑状态的任何数据都将遭到损坏。

[0005] 伴随着半导体加工技术向着更小的尺寸和更低的操作电压发展，集成电路，例如微处理器（“处理器”）的软错误率（SER）也增加了。更小的工艺尺寸允许在处理器电路小片上实现更高的器件密度。这增加了 α 粒子或宇宙射线透过处理器电压波节中之一的可能性。更低的操作电压意味着更小的电荷分裂就足以改变由节点电压表示的逻辑状态。这两种趋势将来都指向了更高的SER。如果在任何已遭破坏的数据被用来更新处理器的结构状态之前检测到软错误，它们就可以在处理器中得到纠正。

[0006] 处理器常常采用基于奇偶性的机制来检测由软错误产生的数据损坏。当奇偶位被存储时它与各个数据块相关。这个奇偶位是根据数据块中有奇数或偶数个1而被设置为1或0的。当数据块被从它的存储单元读出时，将该块中1的个数与奇偶位进行比较。这两个值不同表示数据块已经遭到了损坏。这两个值一致则表示或者未出现损坏或者已有2（或4...）位被改变。由于后者发生的概率很低，因而奇偶性给数据是否已遭损坏提供了可靠的指示。纠错码（ECC）是跟踪每个数据块附加信息的基于奇偶性的机制。附加信息允许识别并纠正损坏位。

[0007] 奇偶/ECC机制已经被广泛应用到了高速缓冲存储器、存储器和类似的数据存储阵列。这些结构有相对较高的数据存储节点密度，而且即使在当前的器件密度上对软错误也很敏感。它们的固定阵列结构使得实现奇偶/ECC机制相对容易一些。处理器上其余的电路包括数据通路、控制逻辑电路、执行逻辑电路和寄存器（”执行核心”）。这些电路的不同结构和它们在处理器电路小片（processor die）上的分布使得应用奇偶/ECC机制更为困难。

[0008] 检测执行核心中的软错误的一种方法是在双倍的执行核心上处理指令并比较以逐指令地比较每个核心确定的结果（“冗余执行”）。例如，一个计算机系统包括两个单独的处理器，可以引导它们或以对称多处理（“SMP”）模式或以功能冗余校验（“FRC”）模式运行。以SMP模式，指令执行被分配到各个处理器之间以提供比单处理器系统更高的整体性能。以FRC模式，一个处理器正常地执行代码，第二个处理器在与提供给第一个处理器相

同的数据上执行相同的指令。如果第二个处理器检测到在它的操作和第一个处理器的操作之间存在差异，它就用信号通知产生了错误。只能通过重启计算机系统在 SMP 和 FRC 之间切换操作模式。

[0009] 双处理器方法成本较高（就硅而论）。另外，通过其进行结果比较的处理器间信号传递太慢，以至于无法在损坏的数据更新处理器的结构状态之前就检测到它。因此，这种方法不适于纠正检测到的软错误。

[0010] 另一种计算机系统在单个处理器芯片上使用双执行核心提供执行冗余。两个执行核心以 FRC 模式操作，而且受 ECC 保护的校验点寄存器存储处理器中间状态上的信息。当在一个代码段中检测到错误时，处理器执行微代码程序以利用校验点寄存器来恢复处理器到最后未遭破坏的处理器状态。随后控制返回到该代码段，并从遇到错误的指令开始。错误恢复微代码程序存储在处理器芯片上，使它难以更新或修改。另外，足够灵活以纠正大范围错误的程序势必相对复杂。这些程序的微代码实现方式占据了处理器电路小片上的大量面积。

[0011] 本发明解决了现有的高可靠性计算机系统的这些和其它不足。

[0012] 发明概述

[0013] 本发明提供了用于在双执行核心处理器中恢复软错误的固件机制，该双执行核心处理器能够以冗余模式和分离模式操作执行核心。

[0014] 依据本发明的一种方法在处理器以冗余模式操作执行核心时检测软错误。在每个执行核心上执行错误恢复程序以保存来自与该执行核心相关的存储器结构的未损坏的数据。根据保存的数据恢复处理器状态数据，并用处理器状态数据初始化第一和第二执行核心。

[0015] 依据本发明的计算机系统包括一个双执行核心处理器和存储错误恢复程序的非易失性存储器。当处理器在以冗余模式操作的同时检测软错误时错误恢复程序被调用。该程序把处理器切换到分离模式，在该模式下每个执行核心将其相关的存储器结构中的未损坏数据保存到一个指定的存储单元。该程序依据所保存的数据恢复处理器状态数据。

[0016] 附图概述

[0017] 可以参考下列图来理解本发明，在这些图中同样的元件由同样的数字来表示。提供这些图是为了举例说明本发明的所选实施方案，并不意味着要限制本发明的范围。

[0018] 图 1 是适于实现本发明的计算机系统的一种实施方案的框图。

[0019] 图 2A 是图 1 中双执行核心处理器的一种实施方案的框图。

[0020] 图 2B 是图 2A 中处理器的 FET 级的一种实施方案的框图。

[0021] 图 2C 是图 2A 中处理器的校验单元的一种实施方案的框图。

[0022] 图 3 是说明依照本发明用于从软错误恢复的方法的一种实施方案的流程图。

[0023] 图 4 是说明由执行核心执行的错误恢复程序的一种实施方案的流程图。

[0024] 图 5A 和 5B 是说明为软错误恢复协调所保存的数据的不同机制的实施方案的框图。

[0025] 图 6 是说明图 5A 的数据协调机制的一种实施方案的流程图。

[0026] 图 7 是当双核心处理器以分离模式操作时提供集群级冗余的执行核心的一种实施方案的框图。

[0027] 发明详述

[0028] 下面的论述阐明了大量特定的细节以提供对本发明的彻底理解。然而，从本公开内容受益的本领域的普通技术人员将意识到可以在没有这些特定细节的情况下实践本发明。另外，为了把注意力集中在本发明的特性上，对多种众所周知的方法、程序、部件和电路没有进行详细描述。

[0029] 本发明提供了一种在双核心处理器中纠正软错误的灵活方法，该双核心处理器可以在冗余模式和分离模式之间动态切换。在冗余模式下，处理器执行相同的代码段上以锁定步骤 (in lock step) 操作执行核心并比较结果以识别错误。在分离模式下，执行核心可以独立处理指令，例如，执行核心在给定的时钟周期内可以操作在不同的指令上。

[0030] 对本发明的一种实施方案来说，冗余模式可以是高可靠性 (HR) 处理器执行模式，这种模式减少了执行临界代码段时发生软错误的风险。分离模式可以是高性能 (HP) 处理器执行模式，该可以通过增加所选代码可用的执行资源来更快地处理所选代码。分离模式也可以只用于所选择的目的，例如错误恢复或加速 (boostrapping) 处理器。分离模式的重要特征是它允许执行核心“独立”操作，即每个执行核心在给定的时钟周期上可执行不同的指令。

[0031] 依照本发明，错误恢复程序存储在非易失性存储器中。当以冗余模式实现一个程序线程的处理器检测到软错误时访问恢复程序。可以通过两个执行核心中结果之间的差异指示错误。该程序把处理器切换到分离模式，该模式下每个执行核心在它的相关存储单元检查损坏的数据。把未损坏的数据复制到指定的存储单元并从未损坏的数据恢复出足够的处理器状态数据以继续被中断的程序线程。恢复程序用已恢复的处理器状态数据初始化执行核心。

[0032] 对本发明的一种实施方案来说，与每个执行核心相关的不同资源是受奇偶保护的，并且恢复程序执行奇偶校验以识别执行核心中任何遭到破坏的数据。举例来说受奇偶保护的资源可以包括通用寄存器、浮点寄存器、控制和状态寄存器、低级高速缓冲存储器等。通常，用来存储处理器状态数据的任何结构都可以受奇偶保护。保护范围取决于系统的可靠性需求。

[0033] 对本发明的另一种实施方案来说，错误恢复程序识别并存储来自一个或两个执行核心的未遭破坏的数据到指定的存储单元。用从保存的数据恢复出的处理器状态数据初始化执行核心。例如，可以用从没有经历软错误的执行核心保存的处理器状态数据初始化两个执行核心。或者，可以把来自各个执行核心的未遭破坏的数据保存到不同的存储单元，来自一个存储单元的数据可用来代替其它存储单元中的已遭破坏的数据。这种替代方法用来自每个处理器的指定存储单元的处理器状态数据副本初始化各个执行核心。通过维护处理器状态数据的单独的集合，这种替代方法减少了初始化过程中产生的软错误在检测中被遗漏的风险。在识别并协调分离模式下来自执行核心的已保存的处理器状态数据之后，错误恢复程序为返回到冗余模式而同步执行核心。

[0034] 对本发明的另一种实施方案来说，处理器可以被配置用来在分离模式下的各个执行核心中保持一定级别的冗余。例如，可以为分离模式操作而把每个执行核心的执行资源在逻辑上组织成冗余执行集群。当处理器处于分离模式时执行集群在相同的指令上以锁定步骤执行，可以比较来自冗余集群的结果以检测单个执行核心中的软错误。

[0035] 图 1 是依照本发明的计算机系统 100 的一种实施方案的块级图示。计算机系统 100 包括处理器 102、非易失性存储器 104、动态存储器 106 和系统逻辑 108。系统逻辑 108 在处理器 102、非易失性存储器 104 和动态存储器 106 之间传递通信。错误恢复程序 190 存储在非易失性存储器 104 中，虽然程序 190 的一些部分也可以被复制（映像）到动态存储器 106 中。

[0036] 处理器 102 公开的实施方案包括第一执行核心 110(a)、第二执行核心 110(b)、对应的核心状态寄存器 (CSR)120(a) 和 120(b) 以及校验单元 130。每个 CSR120(a)、120(b) 分别包括一个核心状态位 (CSB)124(a)、124(b)。还显示出了向处理器 102 传入和传出数据 / 指令的总线接口 140。每个执行核心 110(a)、110(b) 包括取指令、译码、执行并退出指令的资源。在下面的论述中，除非对特定的执行核心 110 予以说明，否则对执行核心 110(a)、110(b) 的引用都不加索引。对 CSR120、CSB124 的引用以及在执行核心 110 中重复的任意其它资源也同样处理。

[0037] 在冗余模式下，执行核心 110 以锁定步骤执行来自一个代码段的相同指令，并由校验单元 130 比较结果以检测任一执行核心 110 中的错误。在分离模式下，执行核心 110“独立”操作。也就是说，每个执行核心可以执行来自一个或多个代码段的不同指令。如上所述，分离模式可以为所选择的程序提供可编程高性能模式，因为处理器中可用的执行资源在分离模式下被有效地加倍。

[0038] 分离模式下执行核心 110 之间的独立级别随处理器 102 的不同实施方案有所变化。对一种实施方案来说，在分离模式时处理器 102 可以作为实现在一个单独的处理器芯片上的 SMP 系统操作。这种情况下，每个执行核心 110 作为一个独立的处理器操作，只共享处理器电路小片和存储器系统的特定部件。处理器 102 的这种实施方案对需要高性能处理器的代码尤其有用。处理器 102 的另一种实施方案可以通过提供适当通道以共享处理器状态信息或执行核心资源而支持在分离模式下的执行核心 110 之间一定程度的耦合。

[0039] 可以在硬件、软件或固件控制下面实现在冗余和分离模式之间的切换。对软件和固件控制的切换来说，可以通过正在执行的不同程序线程、操作系统 (OS) 调度程序、中断处理器、固件程序或类似的源来提供模式切换指令。对硬件控制的切换来说，可以响应检测到的状态，例如，特定指令类型的执行、不同执行核心中结果之间差异的检测或者该差异的分辨率，来触发切换。

[0040] 对本发明的一种实施方案来说，CSB124(a) 和 124(b) 分别指示执行核心 110(a) 和 110(b) 的状态，且 CSB124(a) 和 124(b) 一起指示处理器 102 的操作模式。处理器的执行模式作为一个整体可以通过一个单独的处理器状态位 (PSB)128 来进行跟踪，由图 1 中的虚线框表示。对于一种实施方案来说，当处理器 102 将以冗余模式操作时，CSB 124 被设置成第一个值，例如 1，当处理器 102 将以分离模式操作时，CSB 124 被设置成第二个值，例如 0。当模式切换指令触发冗余模式和分离模式之间的切换时 CSB 124 会被调整。处理器 102 中的不同资源根据它的执行模式更改它们的操作。对处理器 102 的公开实施方案来说，指令获取、指令退出和结果校验是以冗余和分离模式做不同处理的，而且对应的资源根据 CSB 124（或 PSB 128）的状态调整它们的操作。

[0041] 在题为“Microprocessor With High Reliability OperatingMode”，且与本案同一日期提交的美国专利中请序列号 09/470,098 中公开了有双执行模式的处理器 102 的一

种实施方案。

[0042] 图 2A 更详细地说明了处理器 102 的一种实施方案。对该公开的实施方案来说，每个执行核心 110 被描述成指令执行流水线中的一系列级。每一级对应于由执行核心 110 实现用来执行它们的指令的一个或多个操作。或者，可以把流水级理解为代表执行所示操作的逻辑电路。指令和数据从存储器系统 270 提供给执行核心 110。存储器系统 270 代表动态存储器 106 和任意介于中间的高速缓冲存储器。例如，高速缓冲存储器 280 代表存储器系统 270 的一部分，来自执行过的指令的结果被写入其中。高速缓冲存储器 280 可以位于和处理器 102 相同的芯片上或者位于一个单独的芯片上。

[0043] 对处理器 102 的公开的实施方案来说，每个执行核心 110 被划分为取出 (FET) 级 210、译码 (DEC) 级 220、寄存器 (REG) 级 230、执行 (EXE) 级 240、检测 (DET) 级 250 和退出 (RET) 级 260。在 FET 级 210 中从存储器系统 270 取出一个或多个指令。在 DEC 级 220 中把取出的指令译码成微操作，并在 REG 级 230 中取出由微操作规定的源操作数。在 EXE 级 240 中在取出的操作数上执行微操作，并在 DET 级 250 中用信号通知由微操作产生的任何异常。如果没有检测到异常就在 RET 级 260 中退出微操作。对该公开的实施方案来说，来自已退出的微操作的结果通过退出通道 264 被写入高速缓冲存储器 280 中。

[0044] 在这个论述中，指令、指令束和宏指令被交替用作微操作和指令字节。后者指由处理器的执行单元识别出的指令。前者指提供给处理器的形式的指令。对一些实施方案来说，在这些实体之间可以有很小的差别或没有。

[0045] 处理器 102 的实施方案可以包括缓冲区以从后端级 (DEC、REG、EXE、DET 和 RET，或 REG、EXE、DET 和 RET) 去耦合前端级 (FET 或 FET 和 DEC)。该缓冲区临时存储取出 (或者取出并经译码) 的指令。这使得前端操作即使在后端操作被停止或延迟时也能继续。如果前端操作被延迟它还允许后端操作继续进行。处理器 102 的一种实施方案采用了去耦缓冲区来纠正在冗余模式下检测到的错误。

[0046] 本发明不需要把处理器 102 分成一组特定的流水级。例如，可以把一个已公开的级再分成两个或更多级以解决定时问题或促进更高的处理器时钟速度。或者，可以把两个或更多级组合成一个单一级。其它实施方案可以采用或不采用去耦缓冲区。其它实施方案仍然可包括用于处理指令无序的硬件。公开的流水线只提供了一个如何在处理器中划分操作的实例以配合本发明的应用。

[0047] 还为每个执行核心 110 显示了状态 / 控制 (S/C) 寄存器 234、数据寄存器 238 和数据高速缓冲存储器 244。S/C 寄存器 234 存储管理执行核心 110 的操作的信息。例如，S/C 寄存器 234 通常包括 CSR120(和 CSB124)。数据寄存器 238 存储由执行核心 110 中不同资源所用的操作数，数据高速缓冲存储器 244 在存储器系统 270 和执行核心中的其它资源之间缓存操作数。根据定时约束，数据高速缓冲存储器 244 可以向数据寄存器 238、EXE 级 240 中的执行资源、或者向这两者提供操作数。对本发明的一种实施方案来说，每个执行核心 110 为 S/C 寄存器 234、数据寄存器 238 和高速缓冲存储器 244 提供一定形式的奇偶保护。

[0048] 当处理器 102 处在冗余模式下时执行核心 110(a) 和 110(b) 被同步以锁定步骤在相同指令上操作。在分离模式下，执行核心 110(a) 和 110(b) 可以在不同的指令上独立操作。如上所述，处理器 102 的不同实施方案在分离模式可以支持执行核心 110(a) 和 110(b)

之间不同级别的协调合作，如图 2A 中的虚线箭头所示。例如，如果处理器 102 作为一个单独芯片 SMP 系统以分离模式操作，执行核心 110(a) 和 110(b) 之间的合作需要主要在模式切换期间出现。对处理器 102 的其它实施方案来说，执行核心 110(a) 和 110(b) 可以处理紧密耦合的进程。这些实施方案支持对 S/C 寄存器 234(a) 和 234(b)、数据寄存器 238(a) 和 238(b)、以及数据高速缓冲存储器 244(a) 和 244(b) 之间的数据的一些共享，以及不同流水级之间的操作的一些合作。

[0049] 图 2B 显示了适于在冗余模式和分离模式下分别向执行核心 110(a)、110(b) 提供指令的 FET 级 210(a) 和 210(b) 的一种实施方案。每个 FET 级 210 包括指令指针 (IP) 选择复用器 212 和耦合到 DEC 级 220 的指令高速缓冲存储器 214。S/C 寄存器 234 包括 IP 寄存器 236，它可以由软件进行初始化以指示将要执行的下一条指令。另外，FET 级 210(b) 包括在高速缓冲存储器 214(b) 的输出上的复用器 216。复用器 216 由 CSB 124 通过与门 218 控制。对本发明的一种实施方案来说，指令高速缓冲存储器 214 可以由像 ECC 这样基于奇偶性的机制来保护。

[0050] 复用器 212 在它的数据输入上从包括 IP 寄存器 236 在内的多个来源接收 IP。响应它的控制输入上的信号，复用器 212 选择一个 IP 来指示将要从高速缓冲存储器 214 取出的下一条指令。在分离模式下，CSB124 被设置为 0 而且复用器 216 传输由指令高速缓冲存储器 214(b) 提供的指令。在这种模式下，IP 寄存器 236(a) 和 236(b) 被独立地进行初始化和更新，高速缓冲存储器 214(a) 和 214(b) 分别提供对应的指令给 DEC 级 220(a) 和 220(b)。在冗余模式下，CSB 124 被设置为 1，复用器 216 从高速缓冲存储器 214(a) 提供指令给 DEC 级 220(b)。

[0051] FET 级 210 的一种替代实施方案不使用复用器 216。而是为冗余模式而把 IP 寄存器 236(a)、236(b) 和高速缓冲存储器 214(a)、214(b) 初始化成相同状态，并且 FET 级 210，包括高速缓冲存储器 214，均以锁定步骤操作。处理器设计领域的以及从本公开内容获益的技术人员将认识到可以根据处理器 102 操作的执行模式使用 FET 级 210 的其它变体来为执行核心 110 实现独立的和锁定步骤的指令取出。

[0052] 图 2C 是说明依照本发明的校验单元 130 的一种实施方案框图。校验单元 130 的公开的实施方案包括“n”个比较器 290(1)-290(n)、或门 294 和与门 298。为执行核心 110 中的每个执行单元提供一个比较器 290。例如，处理器 102 的一种实施方案可以在每个执行核心 110 的 EXE 级中包括整数执行单元 (IEU)、浮点执行单元 (FPU)、存储器执行单元 (MEU) 和分支执行单元 BRU。对这个实施方案来说，校验单元 130 包括 4 个比较器 290。比较器 290(1)、290(2)、290(3) 和 290(4) 分别监控来自执行核心 110(a) 和 110(b) 的 IEU、FPU、MEU 和 BRU 的输出。

[0053] 对校验单元 130 公开的实施方案来说，当施加到它的输入上的执行结果匹配时每个比较器 290 产生一个逻辑值 0，当执行结果不匹配时产生逻辑值 1。对校验单元 130 的一种实施方案来说，比较器 290 是自校验比较器。当任意一个比较器 290 指示出的与它对应的执行结果不匹配时或门 294 产生逻辑值 1。当与门 298 被使能时或门 294 的输出充当 ERROR 信号。对该公开的实施方案来说，当 CSB 124 都被设置为 1，即处理器 102 在冗余模式下时这才会发生。

[0054] 处理器设计领域的以及从本公开内容获益的技术人员将认识到当处理器在冗余

模式下时可以激活校验单元 130 的其它变体来监控执行核心 110 中的结果。

[0055] 当在冗余模式下检测出偏差时，本发明提供了一个可以独立实现或与基于硬件和软件的恢复机制结合在一起的基于固件的恢复机制，以把处理器恢复到无错状态。作为本发明目标的软错误通常不会在两个执行核心 110 中同时发生。由校验单元 130 检测到的执行结果之间的差异通常是由执行核心 110 之一的电路中的软错误而产生的。像寄存器文件、高速缓冲存储器、锁存器以及类似的这些存储器结构尤其易于发生这些错误。这些结构存储代表指令流水线的不同点上的操作数值、指令或者微操作的电压电平。当在由执行核心产生的结果之间检测到差异时本发明提供了一种恢复完整性到以冗余模式操作的处理器的执行核心的机制。

[0056] 根据本发明的一种实施方案，当在冗余模式下的处理器的两个执行核心产生的结果之间检测到差异时会访问基于固件的恢复程序。错误恢复程序切换处理器到分离模式，在该模式下每个执行核心检查来自它的执行资源的数据中是否有错误并把未遭损坏的数据保存到特定的存储单元。错误恢复程序用未遭破坏的数据初始化执行核心并返回控制到被中断的程序线程。对本发明的一种实施方案来说，所检查的执行资源是受奇偶保护的存储器结构。从未遭破坏的数据恢复出足够的处理器状态数据来初始化两个执行核心以使中断的程序线程得以继续处理。

[0057] 使用基于固件的恢复程序使处理器能够实现更复杂的恢复机制。这些机制比通过微代码或基于硬件的恢复机制能够解决更多种类的错误，后两种恢复机制均受限于它们对电路小片面积的影响。可以更方便地更新基于固件的恢复机制以处理不同的错误或实现错误恢复算法中的改进。另外，在必要时基于固件的恢复机制能够通知诸如系统级固件或操作系统的更高级的系统采取适当动作。

[0058] 对本发明的一种实施方案来说，可以通过机器校验 (MC) 访问错误恢复程序。当检测到差异时 MC 被校验单元 130 触发。或者，处理器可以先试图通过硬件恢复机制解决偏差并只在硬件机制失败时触发错误恢复程序。例如，硬件恢复机制包括重新控制 (resteering) 执行核心或执行核心的一些部分从检测到差异时的指令开始重新执行指令流。

[0059] 图 3 是说明依照本发明用于纠正软错误的方法 300 的流程图。当处理器以冗余模式操作并检测到错误 310 时方法 300 被启动。错误可以由第一和第二个执行核心产生的结果之间的差异来指示。当检测到错误时 310，处理器跳转到错误恢复程序 (ERR)320。例如，这可以通过触发 MC 并通过相关向量表控制处理器到 ERR 来完成。

[0060] ERR 把处理器从冗余模式 (RM) 切换到分离模式 (SM)330。对方法 300 的一种实施方案来说，每个执行核心以分离模式操作以从受奇偶保护的存储器结构把未遭破坏的数据保存到指定的存储单元 340。可以通过扫描受奇偶保护的资源中的数据检查奇偶错误或者通过把数据从这些资源复制到一个存储单元并扫描复制来的数据以检查奇偶错误来识别未受损坏的数据。当两个执行核心都完成了它们的数据保存时 350，对所保存的数据进行协调 360 以提供未遭破坏的一组处理器状态数据。对方法 300 的另一种实施方案来说，只有在它的存储结构中识别出没有遭破坏的数据的执行核心才把它的数据保存到指定的存储单元 340。处理器被返回到冗余模式 370，用恢复出的处理器状态数据初始化执行核心 380，并继续中断的进程线程的执行 380。

[0061] 对本发明的一种实施方案来说，处理器 102 在软件控制下切换于冗余和分离模式之间，例如通过模式切换指令。对本发明的另一种实施方案来说，处理器 102 可通过硬件机制进行切换。

[0062] 在检测到差异前，处理器工作于冗余模式而且执行核心以锁定步骤进行操作。当检测到差异时，错误恢复程序可以通过给每个执行核心提供“切换到分离模式”指令 (SW_SM) 来实现模式切换。在冗余模式下，这些指令被以锁定步骤沿着两个执行核心的流水线级分级，当它们退出时切换处理器到分离模式。因而不需要在切换到分离模式时同步执行核心。一旦处理器处于分离模式下，每个执行核心可以独立地执行 ERR 的数据恢复操作。

[0063] 当执行核心完成了它的数据恢复操作时，处理器从分离模式转回到冗余模式以继续中断的程序线程。对本发明的一种实施方案来说，这种切换可以通过在每个执行核心中执行“切换到冗余模式”指令 (SW_RM) 分级地实现。把切换分级能够适应执行核心完成它们的数据保存操作时在时间上的差别。

[0064] 对该公开的实施方案来说，SW_RM 在第一级把一个执行核心切换到“就绪状态”，例如 CSB(b) 124 ＝ 1。在第二级，SW_RM 指令把另一个执行核心切换到“就绪状态”，例如 CSB(a) ＝ 1。当两个执行核心都到达就绪状态，例如 PSB ＝ CSB(a)&CSB(b) 时，处理器返回到冗余模式。第一和第二级可以完全重合（同时切换）、部分重合或完全不重合（顺序切换），取决于用来调度模式切换指令的算法。

[0065] 就绪状态允许在处理器从分离模式切换到冗余模式之前同步执行核心。对本发明的其它实施方案来说，可以通过硬件机制实现同步。例如，当检测到程序结束条件时第一个执行核心可以转换到就绪状态，且当第一个执行核心完成它的转换时第二个执行核心可以转换到就绪状态。和用软件控制的模式切换一样，就绪状态允许执行核心在处理器被切换回冗余模式之前会合。

[0066] 图 4 说明了由执行核心实现用来保存未损坏数据的方法 400 的一种实施方案。例如，方法 400 可以代表在切换到分离模式之后由执行核心实现的 ERR 的一部分。如上所述，可以通过机器校验来到达 ERR，机器校验通过相关的向量表把处理器的控制传递到 ERR。ERR 从冗余模式切换处理器到分离模式，这允许每个执行核心独立地访问并实现数据校验和由图 4 所说明的恢复程序。

[0067] 根据该公开的方法，非临界存储器结构被刷新 410。非临界存储器结构包括诸如高速缓冲存储器、寄存器和不存储结构状态数据的存储器结构。这些存储器结构通常是不受奇偶保护的。根据处理器实施方案，这些设备可以包括 L0 指令和数据高速缓冲存储器、分支目标缓冲区、高级装载跟踪表等等。

[0068] 检查数据寄存器文件的内容以查找奇偶错误 420。如果没有发现奇偶错误 430，就把数据寄存器文件的内容复制到一个指定的存储单元 440。如上所述，这可以是与另一个执行核心共享的存储单元或者为特定的执行核心保留的存储单元。如果发现了奇偶错误 430，就不从数据寄存器文件复制任何数据。换句话说，只有未损坏的数据才从数据寄存器文件中得到复制。

[0069] 还校验 450C/S 寄存器文件以查找奇偶错误。这些包括例如上述核心状态寄存器、存储有关当前线程信息的不同寄存器以及任意翻译后援缓冲区 (TLB) 中的翻译寄存器。如果在 C/S 寄存器文件中没有发现任何奇偶错误 460，就把 C/S 寄存器文件的内容复制到指定

的存储单元 470。如果发现了奇偶错误，就不从 C/S 寄存器文件复制任何数据。换句话说，只有未损坏的数据才从 C/S 寄存器文件中得到复制。

[0070] 处理器的实施方案可以用固件为所选的高速缓冲存储器实现错误纠正代码 (ECC)。这消除了对处理器中 ECC 纠错硬件的需要，虽然高速缓冲存储器仍然存储 ECC 位。对这些实施方案来说，可以通过 ERR 纠正 1 和 2 位的错误 480。如果用硬件实现 ECC 就绕过块 480。一旦纠正了所有检测到的错误，执行核心就跳转到等待状态 490。

[0071] 一旦各个执行核心都已经执行了方法 400，指定的存储单元就只包含未损坏的数据。软错误的统计特性的一个结论是相同的数据块同时在两个执行核心中被破坏的可能性很小。因此，数据保存过程提供了足够的未遭损坏的处理器状态数据来初始化执行核心并重新开始中断的程序线程。

[0072] 本发明不需要以任何特定的顺序刷新或校验存储器结构。例如，可以在 C/S 寄存器文件之后检查数据寄存器文件并在该过程中任何方便的时间刷新非临界文件。上面所指出的顺序只用于说明目的。

[0073] 除了保存未遭破坏的数据到指定的存储单元之外，ERR 还提供了一种机制来协调所保存的数据并将足够的未损坏的处理器状态数据提供给每个执行核心以再继续中断的程序线程。

[0074] 图 5A 是说明用于协调由双执行核心处理器中的每个执行核心保存的数据的一种机制的框图。执行核心 110(b) 实现 ERR 以复制 (1) 它的未损坏的数据到存储器系统 500 中一个指定的单元 510(b)。执行核心 110(a) 实现 ERR 以复制 (2) 它的未损坏的数据到一个指定的存储单元 510(a)。对所示范的机制来说，执行核心 110(b) 的存储器结构中的软错误给单元 510(b) 留下了一个不完全的处理器状态数据集。没有遇到奇偶错误的执行核心 110(a) 复制 (2) 了一个完整的处理器状态数据集到单元 510(a)。这里，“完整”指足以使处理器 102 重新开始执行中断的程序线程的处理器状态数据的子集。可以在执行核心 110(b) 复制 (1) 它的未遭破坏的数据到存储单元 510(b) 的同时跟踪执行核心 110(b) 的存储器结构中遭到破坏的数据，以维持需要被更新的数据记录。

[0075] 在数据保存操作 (1)、(2) 之后，执行核心 110(b) 实现 ERR 以从单元 510(a) 复制 (3) 未损坏的数据到单元 510(b)。这为重新初始化处理器 102 的执行核心提供了第二个处理器状态数据的完整集合。例如，当执行核心 110(b) 确定执行核心 110(a) 已经完成了它的复制操作时，它可以只把对于它的相关存储器结构中遭到破坏的数据相对应的数据块把数据从单元 510(a) 复制到单元 510(b)。这个数据协调机制在存储器 500 中提供了两组处理器状态数据，随后可分别用它们来初始化 (4) 和 (5) 执行核心 110(a) 和 110(b)。初始化 (4) 和 (5) 可以由各个执行核心 110 在分离模式下分别独立完成或者在冗余模式由执行核心并发完成。

[0076] 除了 ERR 中从存储单元 510(a) 复制 (3) 未损坏的数据到存储单元 510(b) 的那部分之外，该公开的数据协调机制通过冗余操作继续进行。数据由各个执行核心校验、保存到存储器并写回到存储器结构。这减少了在恢复期间一组处理器状态数据中产生的软错误将破坏两个执行核心的可能性。

[0077] 图 6 是说明图 5A 的数据协调机制的流程图。例如，这对应于图 4 的模块 490。实现 ERR 的数据协调部分的处理器确定 610 另一个执行核心是否已经完成了它的数据保存操

作。当这些操作完成时，该执行核心确定 620 它是否需要更新它的保存数据。

[0078] 对方法 600 的一种实施方案来说，在它的存储器结构中检测到损坏数据的执行核心用来自另一个执行核心的存储单元的对应数据更新 630 它的保存数据。例如，该执行核心可以存储在它的保存操作期间似乎遭到破坏的数据块的指示。这个执行核心然后可以用这些指示从另一个执行核心的指定存储单元中的适当存储器地址中取出该数据的未损坏的版本。没有检测到任何损坏数据的执行核心等待另一个执行核心完成 640 它的更新 630。当两个执行核心都到达等待状态 640 时，处理器返回 650 到冗余模式并用协调过的数据初始化 660 执行核心。方法 600 的一种替代实施方案可以分离模式初始化执行核心然后返回到冗余模式。

[0079] 图 5B 说明了数据协调的一种替代机制。对这种机制来说，两个执行核心 110(a)、110(b) 分别把它们未损坏的数据写入 (1) 和 (2) 到相同存储单元 520 中的对应存储器地址中。对该机制的一种实施方案来说，只有数据没遭到破坏的的执行核心才能把所有它的处理器状态数据写入到存储单元 520。在自己的存储器结构中检测到损坏数据的执行核心不把损坏的数据复制到存储单元 520。

[0080] 图 5B 中的机制的另一种实施方案可以从在其相关存储器结构中没有检测到损坏数据的执行核心写数据并跳过另一个执行核心，即包含有损坏数据的执行核心的保存步骤。无论哪种情况，复制到单元 520 的处理器状态数据都是完整的，并且都能够被分别写回 (3) 和 (4) 到执行核心 110(a) 和 110(b) 以重新初始化它们从而恢复中断的程序线程。因为在图 5B 的数据协调机制中只维护了一个处理器状态数据集，在数据恢复期间这个数据集中的软错误不能通过简单的结果比较就检测出来。

[0081] 对本发明的上述实施方案来说，错误恢复是由处理器以分离模式实现以允许各个执行核心独立地识别出它的相关存储器结构中的任何错误。当冗余执行不能用来检测软错误时，在这些错误恢复操作期间也有可能发生软错误。对高可靠性系统来说，即使软错误的这个有限的弱点也有可能过大。

[0082] 对本发明的一种实施方案来说，可以配置每个执行核心的执行资源以在分离模式操作期间作为两个或更多逻辑集群并行操作。在各个执行核心独立实现 ERR 的同时，每个执行核心中逻辑定义的执行集群冗余地实现来自 ERR 的指令。对这个实施方案来说，来自 ERR 的指令在每个执行核心内部重复执行并被导向不同的执行集群。可以比较由不同的执行集群产生的结果以确定在处理 ERR 指令的同时是否有软错误出现。

[0083] 图 7 是包括该内部冗余特性的执行核心 710 的一种实施方案的框图。该执行核心 710 的公开实施方案包括一个指令高速缓冲存储器 (I-cache)714、译码或分散单元 720、寄存器文件 730、分别包括第一和第二执行集群 740(a) 和 740(b)、校验单元 760 和退出单元 770。在下面的论述中，除非为了避免多义性的必要之外，对执行集群 740(a) 和 740(b) 的引用均不加索引。对执行核心 710 中有副本的其它资源，例如构成执行集群 740 的执行单元，也做相同处理。

[0084] 对该公开的实施方案来说，每个执行集群 740 包括分支执行单元 (BRU)752，整数执行单元 (IEU)754，存储管理（装载 / 存储）单元 (MMU)756 和浮点单元 (FPU)758（一般也称“执行单元”）。执行核心 710 的不同实施方案在不偏离本发明的范围的情况下可以包括不同类型和数量的执行单元。例如，Intel 公司的 Merced 处理器采用的分支执行单元包

括三个独立的分支执行单元。执行集群 740 被独立显示以说明在分离模式下它们的逻辑组织，并非为了反映不同执行资源之间的实际分离。

[0085] 执行核心 710 的其它实施方案可以不重复所有的执行单元 750 或者不重复指令到特定的执行单元。例如，分支执行单元需要大量的处理器资源支持，而且不重复 BRU 752 可以节省可观的电路小片面积。这由图 7 中 BRU 752(b) 周围的虚线框表示。这种情况下，分支指令或者不被冗余执行，或者可以采用替代机制来检查它们的执行。例如，分支指令可重复并串行执行。同样，重复装载和存储指令可以使 HR 模式下到存储器系统的带宽饱和。这种情况下，可以实现多个 MMU 756 来满足一个指令束中多个装载和 / 或存储操作的存在，但在 HR 模式下单个的装载 / 存储操作并不被重复这些指令。在不偏离本发明的精神的情况下也可以对其它执行单元 750 和指令类型可采用类似的装置。

[0086] I-cache714 提供指令给译码单元 720，译码单元 720 通过寄存器文件 730 把指令导向适当的执行单元。寄存器文件 730 包括数据寄存器文件 732、控制 / 状态 (C/S) 寄存器 734 和寄存器重命名单元 738。数据寄存器文件提供对例如由执行单元 750 操纵的整数和浮点操作数的临时存储。重命名单元 738 把微操作中指定的虚拟寄存器标识符映射到寄存器文件 730 中的物理寄存器。

[0087] C/S 寄存器 734 存储控制执行核心 710 的操作模式和不同执行资源状态的信息。对执行核心 710 的一种实施方案来说，当处理器处于分离模式时分发单元 720 提供相同的指令给执行集群 740(a) 和 740(b)。例如，分发单元 720 提供来自一条指令的微操作（或来自相同指令的微操作）给执行集群 740(a) 和 740(b) 中适当的执行单元。由校验单元 760 比较执行集群 740(a) 和 740(b) 产生的结果，如果执行结果不同就指示发生了错误。在冗余模式下，可以独立控制执行集群 740(a) 和 740(b) 以处理不同的指令，并通过在另一个执行核心上执行相同的指令来提供冗余。当处理器以分离模式操作时执行核心 710 提供执行集群 - 级冗余以保护处理器不发生软错误。在题为“Microprocessor With High Reliability Operating Mode”，并且与本案同一日期提交的美国专利申请序列号 09/470,098，中详细论述了包括执行核心 710 的处理器的一种实施方案。

[0088] 因而已经公开了用于在双核心处理器中纠正软错误的基于固件的机制，该处理器可以在冗余执行模式和分离执行模式之间进行切换。当在以冗余模式执行的程序线程中检测到错误时，处理器切换到分离模式。在分离模式下，执行核心在它的存储器结构中识别出未遭损坏的处理器状态数据并把该未遭损坏数据复制到一个指定的存储单元。协调未遭损坏的数据以提供足够的处理器状态数据来恢复中断的程序线程的执行。处理器返回到冗余执行模式，用处理器状态数据初始化执行核心，并恢复中断的程序线程。可以通过模式切换指令实现冗余模式和分离模式之间的切换。

[0089] 已经提供了该公开的实施方案来说明本发明的不同特性。从本公开内容获益的处理器设计领域的技术人员将认识到公开的实施方案的变体和改进依然属于所附权利要求的精神和范围之中。

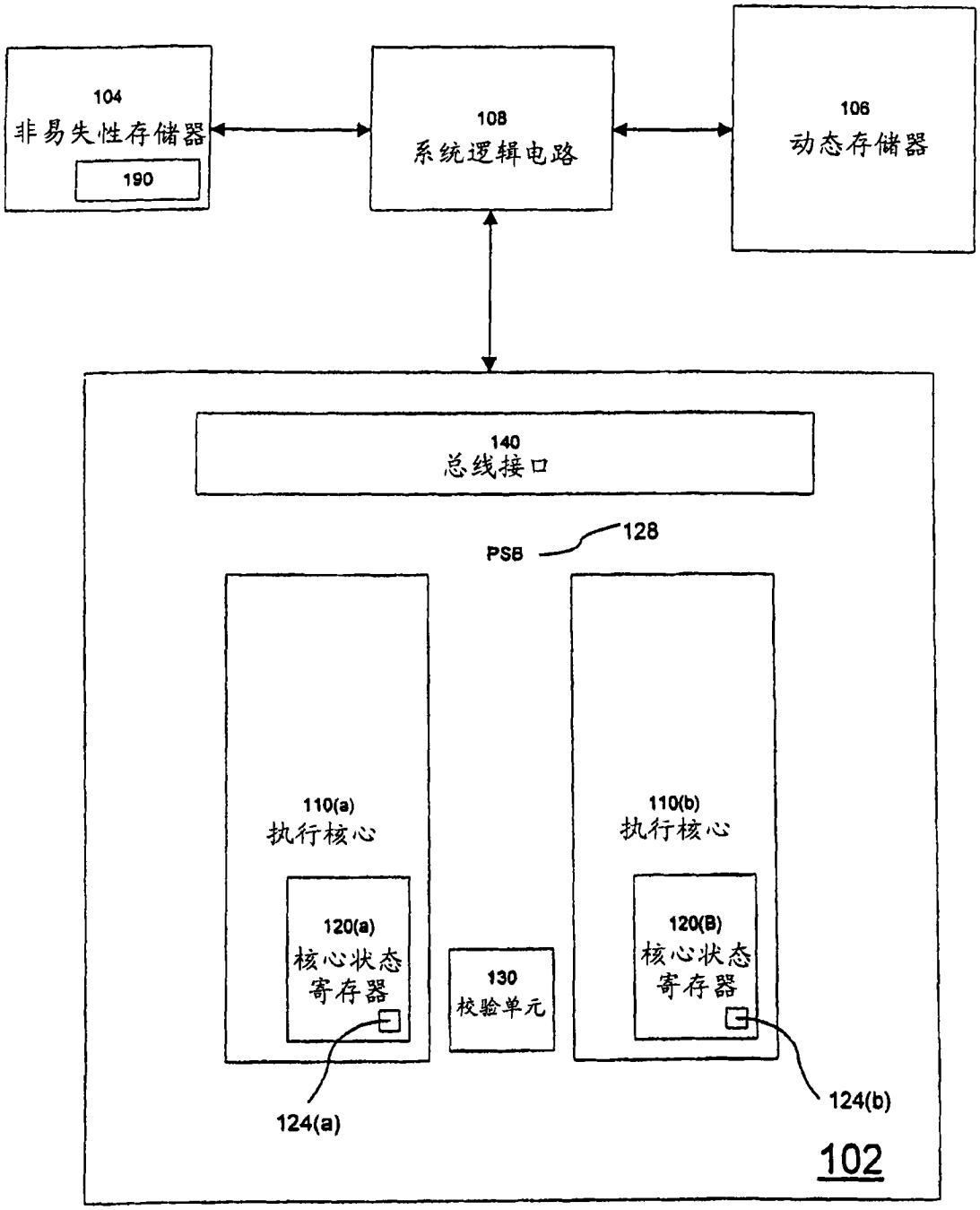

104
非易失性存储器
190
108
系统逻辑电路
106
动态存储器
140
总线接口
PSB
128
110(a)
执行核心
120(a)
核心状态
寄存器
130
校验单元
110(b)
执行核心
120(B)
核心状态
寄存器
124(a)
124(b)
102

图 1

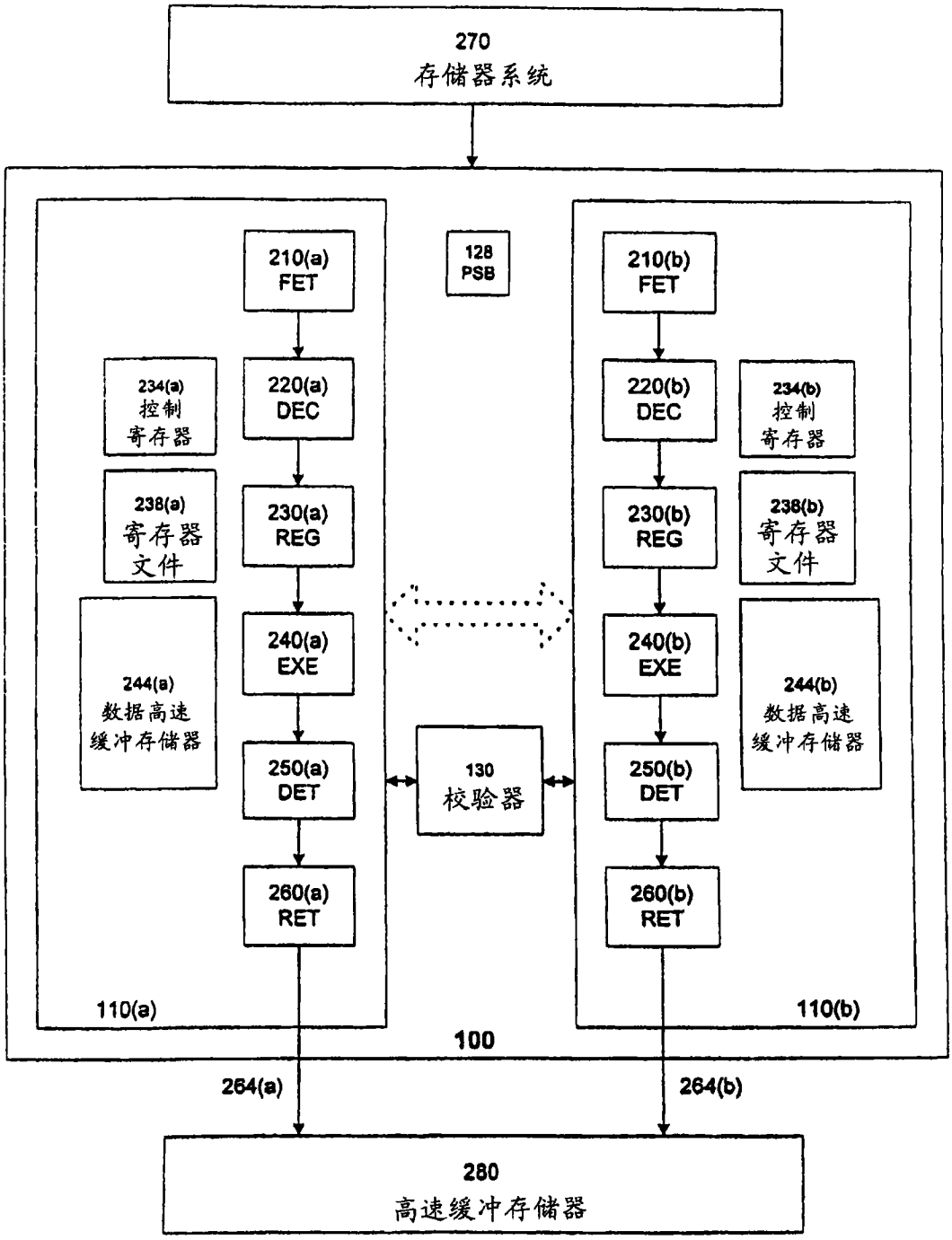

270
存储器系统
210(a)
FET
128
PSB
210(b)
FET
234(a)
控制
寄存器
220(a)
DEC
220(b)
DEC
234(b)
控制
寄存器
238(a)
寄存器
文件
230(a)
REG
230(b)
REG
238(b)
寄存器
文件
240(a)
EXE
240(b)
EXE
244(a)
数据高速
缓冲存储器
244(b)
数据高速
缓冲存储器
130
校验器
250(a)
DET
250(b)
DET
260(a)
RET
260(b)
RET
110(a)
110(b)
100
264(a)
264(b)
280
高速缓冲存储器

图 2A

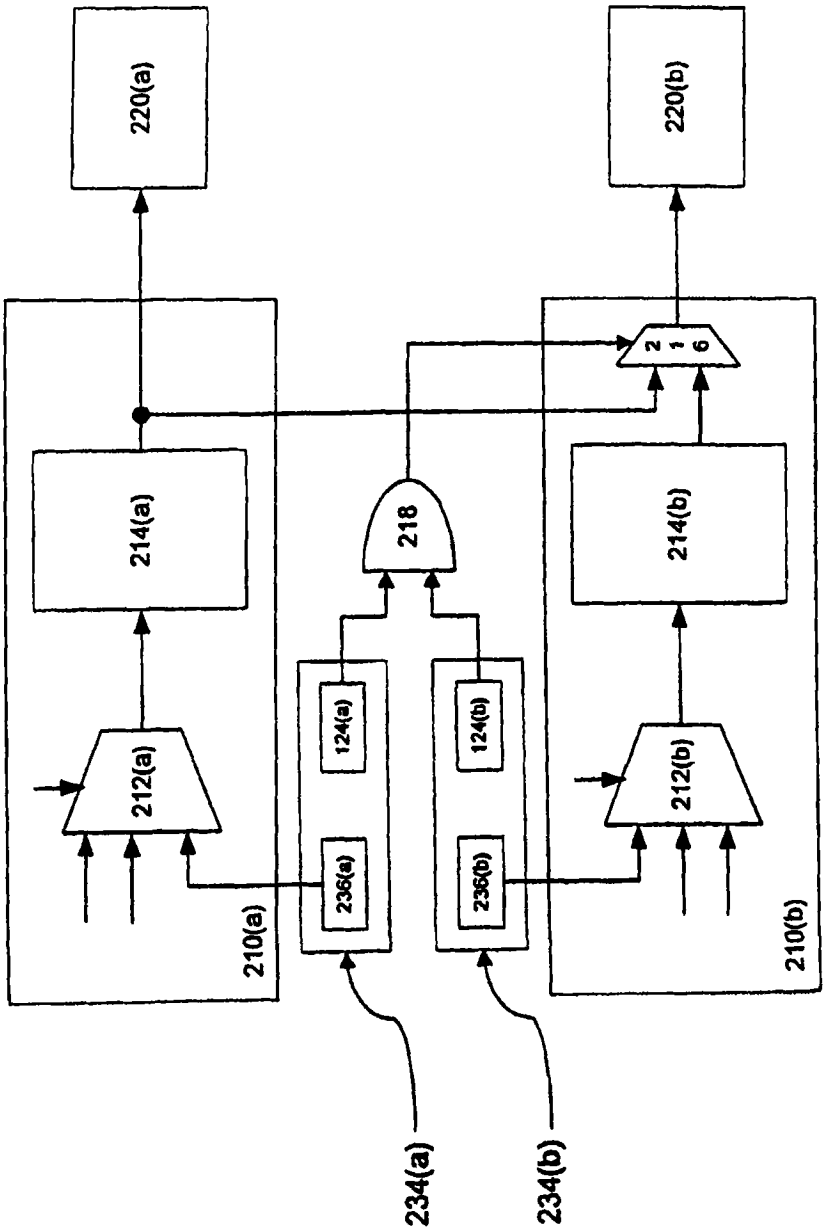

220(a)
220(b)
214(a)
214(b)
218
212(a)
212(b)
124(a)
124(b)
236(a)
236(b)
210(a)
210(b)
234(a)
234(b)

图 2B

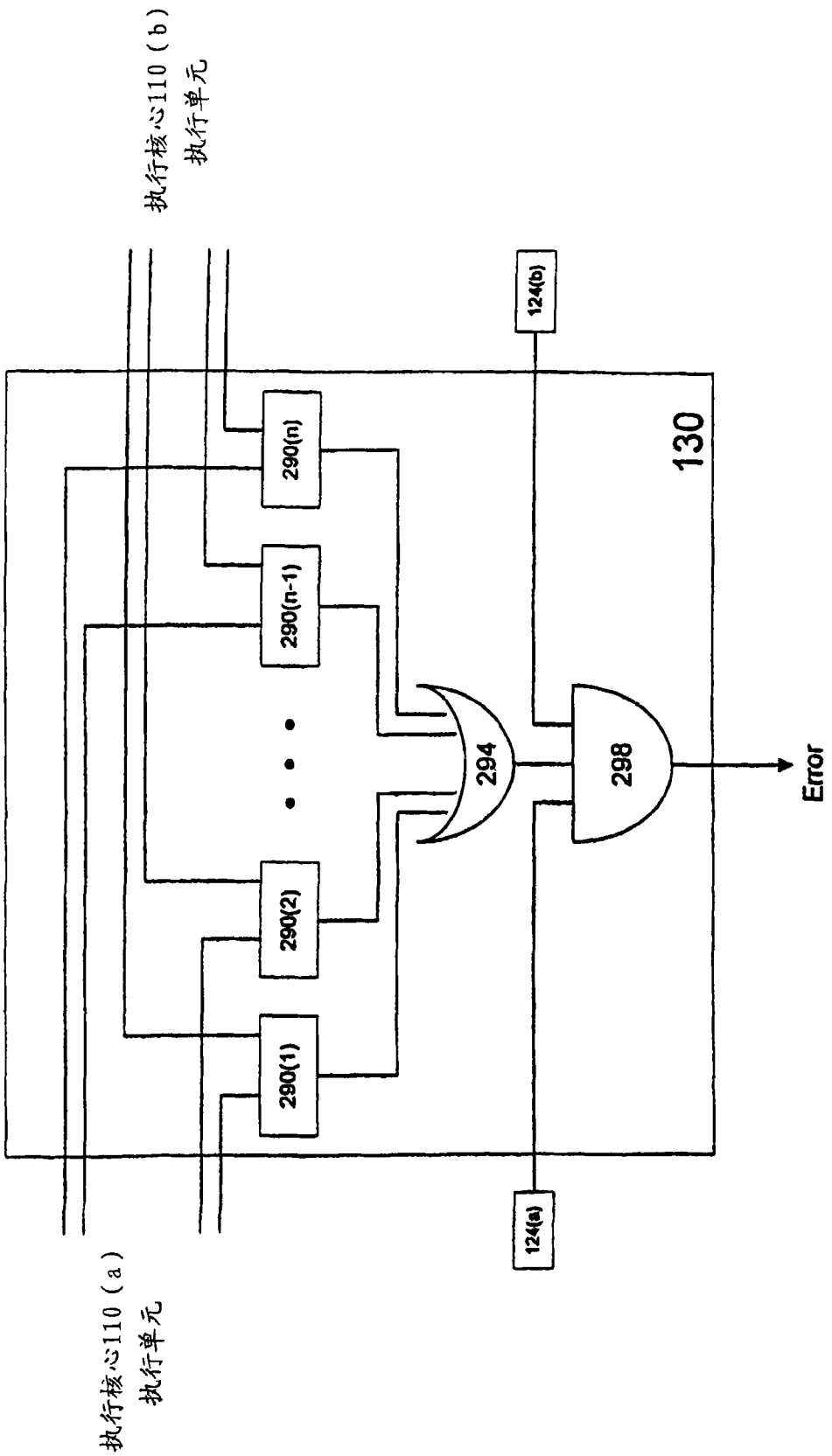

执行核心110（b）
执行单元
执行核心110（a）
执行单元
124(b)
124(a)
130
290(n)
290(n-1)
290(2)
290(1)
294
298
Error

图 2C

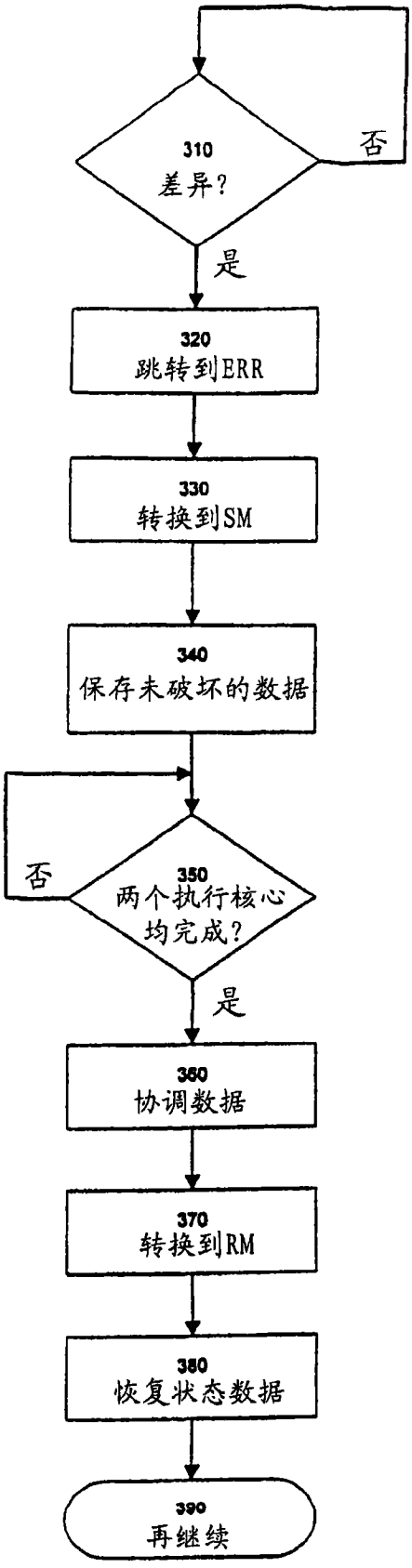

310
差异?
否
是
320
跳转到ERR
330
转换到SM
340
保存未破坏的数据
350
两个执行核心均完成?
否
是
360
协调数据
370
转换到RM
380
恢复状态数据
390
再继续

图 3

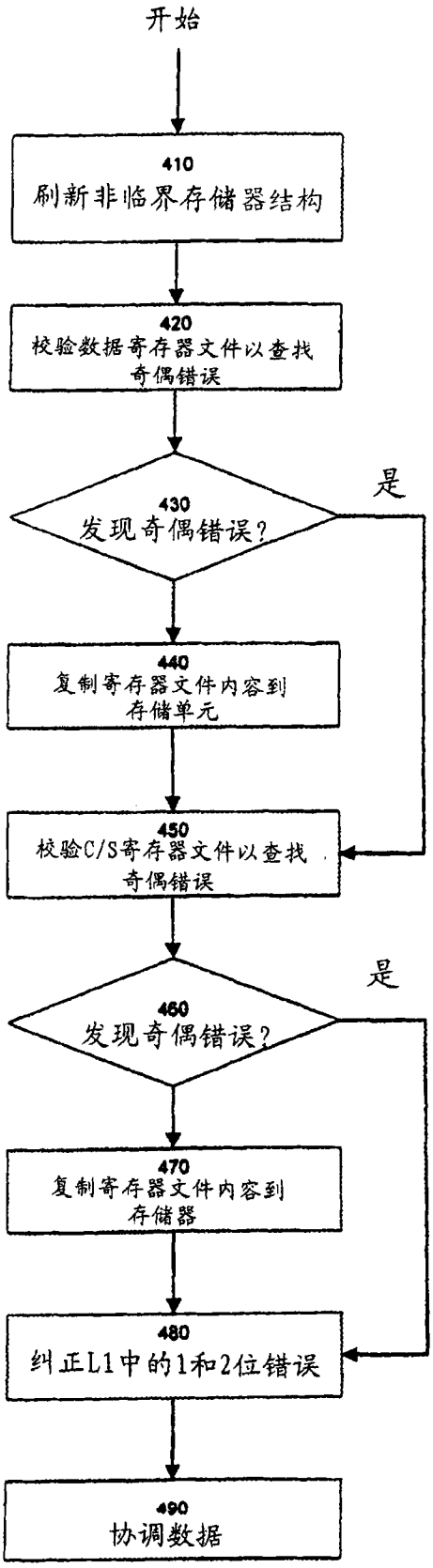

开始
410
刷新非临界存储器结构
420
校验数据寄存器文件以查找奇偶错误
430
发现奇偶错误?
是
440
复制寄存器文件内容到存储单元
450
校验C/S寄存器文件以查找奇偶错误
460
发现奇偶错误?
是
470
复制寄存器文件内容到存储器
480
纠正L1中的1和2位错误
490
协调数据

图 4

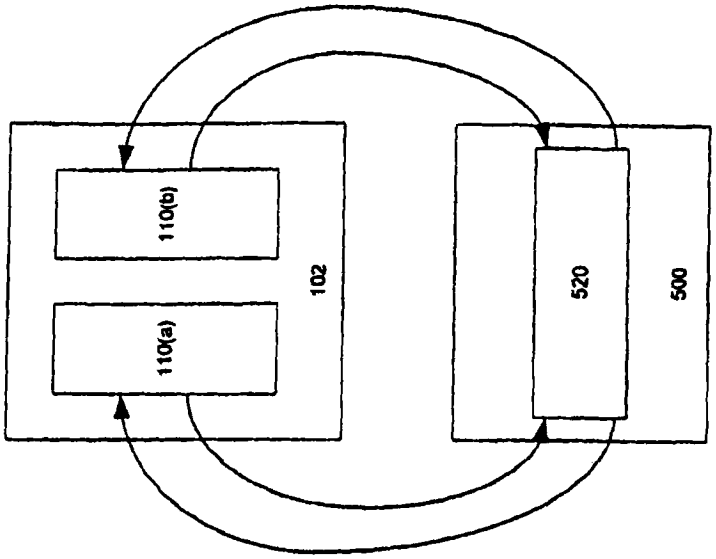


图 5B

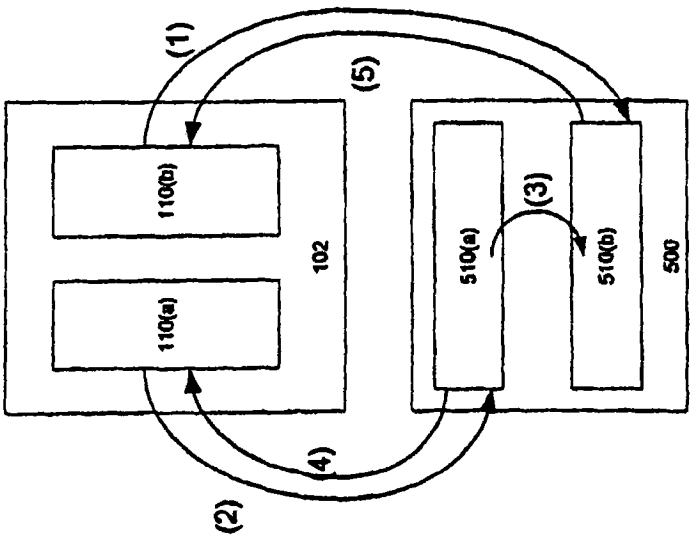


图 5A

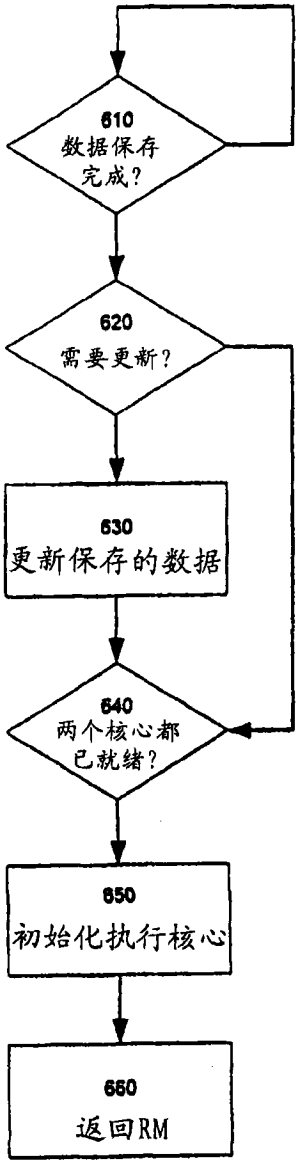

610
数据保存
完成?
620
需要更新?
630
更新保存的数据
640
两个核心都
已就绪?
650
初始化执行核心
660
返回RM

图 6

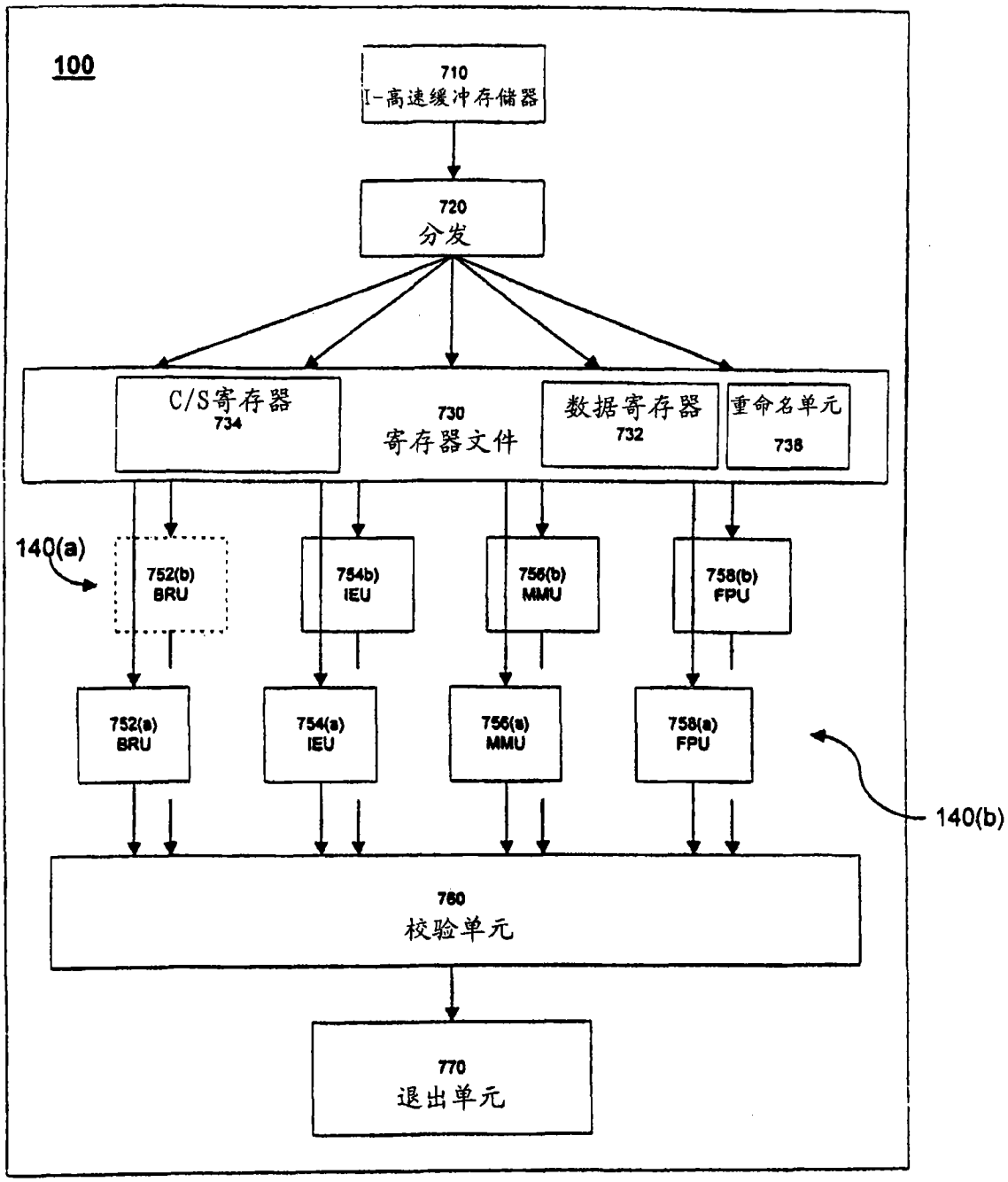

100
710
I-高速缓冲存储器
720
分发
C/S寄存器
734
730
寄存器文件
数据寄存器
732
重命名单元
738
140(a)
752(b)
BRU
754b)
IEU
756(b)
MMU
758(b)
FPU
752(a)
BRU
754(a)
IEU
756(a)
MMU
758(a)
FPU
140(b)
760
校验单元
770
退出单元

图 7