[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/46 (2006.01)
G06F 9/30 (2006.01)

[12] 发明专利说明书

专利号 ZL 03812624.9

[45] 授权公告日 2009 年 3 月 25 日

[11] 授权公告号 CN 100472453C

[22] 申请日 2003.5.7 [21] 申请号 03812624.9

[30] 优先权

[32] 2002. 5.31 [33] US [31] 10/159,386

[86] 国际申请 PCT/US2003/014215 2003.5.7

[87] 国际公布 WO2003/102723 英 2003.12.11

[85] 进入国家阶段日期 2004.11.30

[73] 专利权人 飞思卡尔半导体公司

地址 美国得克萨斯

[72] 发明人 威廉·C·莫耶 约翰·H·阿伦茨

[56] 参考文献

US5386563A 1995.1.31

US5680588A 1997.10.21

US6145049A 2000.11.7

US5426766A 1995.6.20

US6170997B1 2001.1.9

A user – level process package for PVM. KONURU R. SCALABLE HIGH. PERFORMANCE COMPUTING CONFERENCE, 1994. , PROCEEDINGS OF THE KNOXVILLE, TN, USA. 1994

PIM architectures to support petaflops level computation inthe HTMT machine. KOGGE P M. INNOVATIVE ARCHITECTURE FOR FUTURE GENERATION HIGH. PERFORMANCE PROCESSORS AND SYSTEMS. 1999

审查员 于春晖

[74] 专利代理机构 中原信达知识产权代理有限责任公司

代理人 樊卫民 钟 强

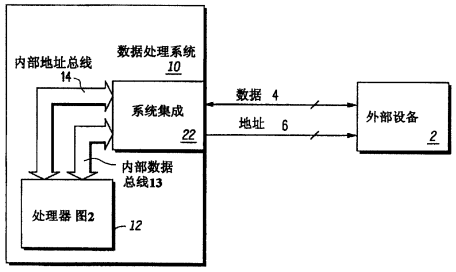
权利要求书 3 页 说明书 14 页 附图 6 页

[54] 发明名称

具有多寄存器上下文的数据处理系统及其方法

[57] 摘要

本发明公开一种数据处理系统(10)，具有多寄存器上下文(34、35、36)。 本发明的一个实施例使用用于多寄存器上下文的每个上下文的用户可编程上下文控制寄存器(140)来允许将部分替换寄存器上下文映射成当前的寄存器上下文。 上下文控制寄存器也可用于提供公共堆栈指针在多寄存器上下文中的共享。 因此，当操作在当前的寄存器上下文中时，上下文控制寄存器可用于访问部分替换寄存器上下文，而不是访问当前寄存器上下文的相应部分。



1. 一种数据处理系统（10），包括：

多个寄存器上下文（34、35、36）；

存储电路（38），用于储存对应于所述多个寄存器上下文的至少一部分的上下文控制信息，其中所述存储电路包括多个上下文控制寄存器，其中所述多个寄存器上下文的每一个包括对应的上下文控制寄存器；和

处理电路（12），用于有选择地访问所述多个寄存器上下文内的一部分替换寄存器上下文，而不是所述多个寄存器上下文内的当前寄存器上下文的对应部分，其中所述替换寄存器上下文和所述一部分替换寄存器上下文由对应于所述当前寄存器上下文的所述上下文控制寄存器指示。

2. 权利要求 1 的数据处理系统，其中，对应于所述当前寄存器上下文的所述上下文控制寄存器包括标识所述一部分替换寄存器上下文的映射字段以及标识所述替换寄存器上下文的替换上下文字段。

3. 权利要求 1 的数据处理系统，其中，所述一部分替换寄存器上下文包括堆栈指针。

4. 权利要求 1 的数据处理系统，其中，所述一部分替换寄存器上下文包括一组寄存器。

5. 一种在具有多个寄存器上下文的数据处理系统中执行的方法，包括：

在所述多个寄存器上下文内的第一寄存器上下文中操作，所述多个寄存器上下文的每一个具有相应的上下文控制信息，其中，在所述第一寄存器上下文中的操作包括：

访问所述第一寄存器上下文的第一部分；

基于所述第一寄存器上下文的相应的上下文控制信息，有选择地访问对应于所述第一寄存器上下文的第二部分的第二寄存器上下文的一部分，其中所述第一寄存器上下文的相应的上下文控制信息标识所述第二寄存器上下文和所述第二寄存器上下文的所述部分。

6. 权利要求 5 的方法，其中，所述上下文控制信息包括使能字段，其中，当声明所述使能字段时，所述方法包括访问对应于所述第一寄存器上下文的所述第二部分的所述第二寄存器上下文的所述部分。

7. 权利要求 5 的方法，其中所述第一寄存器上下文的相应的上下文控制信息包括标识所述第二寄存器上下文的所述部分的映射字段以及标识所述第二寄存器上下文的替换上下文字段。

8. 权利要求 5 的方法，还包括：

接收上下文选择器；

基于所述上下文选择器，在所述第二寄存器上下文中操作；和

基于所述第二寄存器上下文的相应的上下文控制信息，有选择地访问对应于所述第二寄存器上下文的第二部分的第三寄存器上下文的一部分。

9. 权利要求 5 的方法，其中所述多个寄存器上下文的每一个的相应上下文信息包括标识所述多个寄存器上下文的替换上下文的替换上下文字段以及标识由所述替换上下文字段标识的一部分替换上下文的映射字段。

10. 一种数据处理系统（10），包括：

多个寄存器上下文（34、35、36），其中每个寄存器上下文包括具有替换寄存器上下文字段和映射字段的相应的上下文控制寄存器；和

处理装置（12），用于有选择地访问所述多个寄存器上下文内的一部分替换寄存器上下文，而不是访问所述多个寄存器上下文内的当前

寄存器上下文的对应部分，其中，所述一部分替换寄存器上下文是由对应于当前寄存器上下文的上下文控制寄存器中的映射字段定义的，所述替换寄存器上下文是由对应于当前寄存器上下文的上下文控制寄存器中的替换寄存器上下文字段定义的。

具有多寄存器上下文的数据处理系统及其方法

技术领域

本发明一般涉及数据处理系统，更具体涉及具有多寄存器上下文的数据处理系统。

背景技术

在数据处理系统中，诸如微处理器中，使用处理器来控制操作的执行和处理。处理器包括储存寄存器上下文的寄存器，处理器在正常操作和异常处理期间将使用寄存器上下文。当发生中断或处理切换时，寄存器上下文信息可能被破坏，因为中断处理程序或新的处理将使用相同的寄存器并可能改变其中的某些值。

上述问题的一种解决方案是，在开始处理中断或新的处理之前将寄存器当前的值保存在存储器中，当中断处理完成或当返回到当前的处理时再从存储器中把保存的寄存器上下文值读出到寄存器中。但是，保存寄存器上下文、加载新上下文的开销在实时或高性能上下文中是所不希望的。因此，需要一种在数据处理系统中灵活并降低开销的寄存器上下文选择机制。

发明内容

根据本发明的一个方面，提供一种数据处理系统，包括：

多个寄存器上下文；

存储电路，用于储存对应于所述多个寄存器上下文的至少一部分的上下文控制信息，其中所述存储电路包括多个上下文控制寄存器，其中所述多个寄存器上下文的每一个包括对应的上下文控制寄存器；和

处理电路，用于有选择地访问所述多个寄存器上下文内的一部分

替换寄存器上下文，而不是所述多个寄存器上下文内的当前寄存器上下文的对应部分，其中所述替换寄存器上下文和所述一部分替换寄存器上下文由对应于所述当前寄存器上下文的所述上下文控制寄存器指示。

根据本发明的另一方面，提供一种在具有多个寄存器上下文的数据处理系统中执行的方法，包括：

在所述多个寄存器上下文内的第一寄存器上下文中操作，所述多个寄存器上下文的每一个具有相应的上下文控制信息，其中，在所述第一寄存器上下文中的操作包括：

访问所述第一寄存器上下文的第一部分；

基于所述第一寄存器上下文的相应的上下文控制信息，有选择地访问对应于所述第一寄存器上下文的第二部分的第二寄存器上下文的一部分，其中所述第一寄存器上下文的相应的上下文控制信息标识所述第二寄存器上下文和所述第二寄存器上下文的所述部分。

根据本发明的另一方面，提供一种数据处理系统，包括：

多个寄存器上下文，其中每个寄存器上下文包括具有替换寄存器上下文字段和映射字段的相应的上下文控制寄存器；和

处理装置，用于有选择地访问所述多个寄存器上下文内的一部分替换寄存器上下文，而不是访问所述多个寄存器上下文内的当前寄存器上下文的对应部分，其中，所述一部分替换寄存器上下文是由对应于当前寄存器上下文的上下文控制寄存器中的映射字段定义的，所述替换寄存器上下文是由对应于当前寄存器上下文的上下文控制寄存器中的替换寄存器上下文字段定义的。

附图说明

通过举例并不限于附图来说明本发明，其中相似的标号表示类似的元素，并且，在附图中：

图 1 以框图形式说明了根据本发明一个实施例的数据处理系统；

图 2 以框图形式说明了根据本发明一个实施例的处理器；

图 3 说明了根据本发明一个实施例的寄存器上下文；和

图 4 和 5 说明了根据本发明不同实施例的寄存器上下文内的示例映射。

本领域技术人员可认识到，为了清楚简洁地说明图中元素，没有必要按比例绘制。例如，图中某些元素的尺寸相比其他元素可能较为夸大，这是为了帮助增进对本发明实施例的理解。

具体实施方式

如这里所使用的，术语“总线（bus）”用来指代多个信号或导线，其用于传输一种或多种不同类型的信息，诸如数据、地址、控制或状态。术语“声明（assert）”和“否定（negate，或 deassert）”用于指代使信号、状态位等分别呈现其逻辑真或逻辑假状态。如果逻辑真状态是逻辑电平 1，则逻辑假状态为逻辑电平 0。如果逻辑真状态是逻辑电平 0，则逻辑假状态是逻辑电平 1。数字前的符号“$”表示该数字是十六进制或基十六形式（base sixteen form）的表示。

图 1 说明了通过数据总线 4 和地址总线 6 连接到外部设备 2 的数据处理系统 10。数据处理系统 10 包括处理器 12。在一个实施例中，数据处理系统 10 和外部设备 2 都实现为分别的集成电路。在替换的实施例中，数据处理系统 10 和外部设备 2 可以实现在一块集成电路上。在数据处理系统 10 内，处理器 12 通过内部数据总线 13 和内部地址总线 14 连接到系统集成电路 22。

注意到，在本发明的某些实施例中，数据处理系统 10 形成在一块集成电路上。此外，在某些实施例中，数据处理系统 10 可以是单芯片的微控制器、微处理器、数字信号处理器或任何其他类型的数据处理系统。而且，数据处理系统 10 可以使用任意类型的电路来实现。外部设备 2 可以是任意类型的电路，包括存储器或任意类型的外围设备。

替换实施例可包括更多、更少或不同的外部集成电路。此外，总线 4 和 6 可使用任意比特数来实现。

在操作中，系统集成 22 用于允许处理器 12 和外部设备 2 之间的通信。也就是说，处理器 12 将数据和地址信息通过内部总线 14 和 13 传递到系统集成 22，然后系统集成 22 用适于外部设备 2 的方法和形式来通过总线 4 和 6 传递该数据和地址信息。下面将参考图 2 详细讨论处理器 12。

图 2 说明了根据本发明一个实施例的部分处理器 12。处理器 12 包括算术逻辑单元（ALU）24、地址发生器 26、指令流水线 28、指令解码电路 30、寄存器文件集 32 和向量偏移发生器 39。寄存器文件包括多寄存器上下文（context），诸如上下文 0 34、上下文 1 35 和上下文 N 36。因此，寄存器文件 32 包括 N+1 个寄存器上下文，并且虽然在图 2 中只示出了 3 个，处理器 12 可包括任意数量的寄存器上下文，其取决于硬件支持多少寄存器上下文。寄存器文件 32 还包括控制寄存器文件 38。内部地址总线 14 连接到地址发生器 26，地址发生器 26 通过寄存器源总线 40 连接到寄存器文件 32，还连接到内部数据总线 13。向量偏移发生器 39 通过向量偏移总线 27 连接到地址发生器 26。内部数据总线 13 连接到指令流水线 28、ALU 24 和寄存器文件 32。指令解码电路 30 通过指令总线 29 双向连接到指令流水线 28。寄存器文件 32 通过归档信息总线（filed information bus）42 连接到 ALU 24。

图 3 说明了根据本发明一个实施例的寄存器上下文 51。图 3 的寄存器上下文 51 可表示图 2 的上下文 0 到 N 中的任意一个。在图 3 的实施例中，寄存器上下文 51 包括 32 个通用寄存器（GPR）50、一个链接寄存器、一个计数寄存器 56、一个条件寄存器 58、一个整形异常寄存器（integer exception register）60、一个机器状态寄存器 62 和一个上下文控制寄存器 64。链接寄存器 54 用于在从子程序调用和返回时保持子程序链接信息。计数寄存器 56 用于保持处理指令计数循环的计数信息。

条件寄存器 58 用于保持条件代码计算的结果。整形异常寄存器 60 用于提供不同的异常状态。机器状态寄存器 62 用于控制和提供处理器 12 内的不同功能的状态。上下文控制寄存器 64，如下面更详细讨论的，用于提供根据本发明实施例的上下文切换。同时注意到，GPR 50 之一是堆栈指针寄存器 52，保留用于储存当前的堆栈指针。

因此，寄存器上下文指的是上述寄存器（上下文 51 的寄存器）的内容。另外的实施例可定义寄存器上下文为具有寄存器上下文 51 中的全部或某些相同寄存器，或可包括与寄存器上下文 51 不同的寄存器集合。因此，如这里所使用的，寄存器上下文可被定义为具有任意数量和任意类型的寄存器。典型地，寄存器上下文包含组成所有或部分编程者用于处理器的寄存器模型的寄存器资源。在正常操作期间或在上电或复位时，数据处理系统 10 可缺省使用上下文 0 34。（注意到，在另外的实施例中，正常操作可缺省使用不同的上下文。）但是，当发生中断或处理切换时，为了不破坏上下文 0 34 中的值，数据处理系统 10 选择新的上下文（从上下文 1 到 N 中）用于处理中断或执行新处理或线程。因此，中断管理和处理切换（例如多线程）可导致需要在数据处理系统 10 内的寄存器上下文切换。同时，在某些实施例中，可能希望在寄存器上下文中与另一寄存器上下文共享部分寄存器。因此，如下面将要描述的，部分寄存器上下文可映射到另一寄存器上下文，以帮助在上下文切换过程中减少开销并增加速度。

在数据处理系统 10 中，在指令流水线 28 的解码阶段或执行阶段识别异常和中断。因此，当指令提供到指令解码电路 30 并被解码时，可识别和处理中断，而不是正常指令处理。在此描述的一个实施例中，有多个中断级，其确定给定中断是否具有比任何其他中断更高的优先级。这样，具有高优先级的中断将比具有较低优先级的中断更快得到处理，较低优先级的中断必须等待处理。每个中断或中断类型或具有相同优先级的中断因此可共享相同的寄存器上下文（如果希望）。

当接收到中断时，数据处理系统 10 开始执行异常处理序列。在该序列过程中，向量偏移发生器 39 通过向量偏移总线 27 向地址发生器 26 提供向量偏移值。地址发生器 26 使用向量偏移值来形成开始执行处理中断的指令地址。在一个实施例中，除了向量偏移值，向量偏移发生器还提供表示要用于中断处理的寄存器上下文的上下文选择器（context selector）。在一个实施例中，上下文选择器是向量偏移值的一部分，或者可以是由向量偏移发生器 39 提供的单独的值。同时，上下文选择器可以直接提供给寄存器文件 32。在另外的实施例中，上下文选择器可以是从存储器（未示出）中读取的值或者可以通过指令接收。在数据处理系统 10 在寄存器文件 32 中具有 8 个寄存器上下文的例子中，上下文选择器可以是用于识别寄存器上下文之一的 3 比特值。

同时，数据处理系统 10 可能能够进行处理切换，其中处理器 12 能够从一种处理切换到另一种处理，每个处理都可在不同寄存器上下文中操作。例如，在多线程应用中，处理器 12 可以在不同处理线程中连续地切换，其中不同的处理线程（或处理线程组）使用不同的寄存器上下文。在处理切换的例子中，中断可用于向数据处理系统 10 表示处理切换（其中，中断管理包括切换处理）。另外，可使用其他方法向地址发生器 26 表示需要进行处理切换，以使地址发生器 26 能够产生新处理的起始地址。同时，在处理切换时，还提供上下文选择器以表明新处理需要哪个寄存器上下文。如上所述，可以多种不同方式（即，来自向量偏移发生器 39、来自存储器、来自用户指令，等等）且可以直接或间接（例如通过地址发生器 26）提供上下文选择器给寄存器文件 32，从而选择正确的寄存器上下文。

一旦建立了寄存器上下文，处理器 12 执行的指令将引用适当的通用寄存器（GPR 50）或专用寄存器（例如，LR 54、CTR 56、CR 58、XER 60、MSR 62 或 CTXCR 64），对应于当前建立的上下文。其他上下文内的寄存器将不受影响（除非如下面所述建立映射），这样，不需要在执行用于刚建立的上下文的指令之前保存或储存替换的上下文到

存储器中。这提供了开销上的节省。

图 4 和 5 说明了可用于数据处理系统 10 内的寄存器上下文内的示例映射。图 4 说明了三种寄存器上下文：上下文 0 70、上下文 1 72 和上下文 2 74。这些寄存器上下文可表示图 2 的上下文 0 到 N 内的三个上下文。假定在图 4 的例子中，上下文 0 70 对应于数据处理系统 10 的正常操作，上下文 1 72 对应于关键（critical）中断（最高优先级），上下文 2 对应于外部中断（较低优先级）。如上所述，在某些情况下，希望多寄存器上下文“共享”一部分寄存器。因此，在图 4 的例子中，如箭头 82 所示，寄存器上下文 2 74 的堆栈指针寄存器 80 映射到寄存器上下文 1 72 的堆栈指针 78，表明寄存器上下文 1 72 和寄存器上下文 2 74 能够共享相同的堆栈指针，从而在两个寄存器上下文中使用相同的堆栈指针值。这样的映射减少了开销，并帮助维持堆栈指针的一致。因此，当处理外部中断时，上下文 2 74 由数据处理系统 10 来选择。但是，由于堆栈指针寄存器 80 映射到堆栈指针寄存器 78，在寄存器上下文 2 74 的操作过程中访问寄存器上下文 1 72 中的堆栈指针寄存器 78 以访问堆栈指针。换句话说，尽管操作当前上下文值选择寄存器上下文 2 74，试图访问堆栈指针寄存器 80 的指令和其他操作被重定向来访问寄存器上下文 1 72 内的堆栈指针寄存器 78。这允许在上下文 1 和上下文 2 之间共享单独一致的堆栈和堆栈指针值，而免去了同步分别的堆栈指针寄存器 80 和 78 的开销。

注意到，（上下文 0 70 的）堆栈指针寄存器 76 和（上下文 1 72 的）堆栈指针寄存器 78 没有映射；因此，当在这些寄存器上下文中操作时，在访问堆栈指针时不需要访问其他寄存器上下文。每个寄存器上下文 70、72、74 内的上下文控制寄存器 77、79 和 75 分别表示是否映射了相应寄存器上下文的堆栈指针，如果映射了，其映射到哪个其他的寄存器上下文。上下文控制寄存器的细节将在下面参考图 6 进行详细讨论。

图 5 说明了根据另一个例子的三种寄存器上下文：寄存器上下文 1 90、寄存器上下文 2 92 和寄存器上下文 3 94。如同图 4，图 5 的寄存器上下文可表示图 2 的寄存器上下文 0 到 N 中的三个寄存器上下文。在图 5 的例子中，寄存器上下文 1 90 对应于处理 A，寄存器上下文 2 92 对应于处理 B，寄存器上下文 3 94 对应于处理 C。因此，当数据处理系统 10 执行处理 A 时，数据处理系统 10 工作在寄存器上下文 1 90。当处理切换（诸如从处理 A 到处理 B）时，上下文选择器选择寄存器上下文 2 92 以在执行处理 B 时使用。如上参考图 4 所述，每个堆栈指针寄存器能够映射到不同的寄存器上下文。例如，在图 5 中，寄存器上下文 1 90 的堆栈指针寄存器 96 映射到寄存器上下文 2 92 的堆栈指针寄存器 98，如箭头 124 所示。因此，当执行处理 A（使用寄存器上下文 1 90）时，到堆栈指针的访问实际上导致了在不同寄存器上下文（即寄存器上下文 2 92）内到堆栈指针寄存器 98 的访问。注意到，寄存器上下文 3 94 的堆栈指针寄存器 100 没有映射。同时，在一个实施例中，有可能具有分层的映射。例如，就在堆栈指针寄存器 96 映射到堆栈指针寄存器 98 的时候，堆栈指针寄存器 98 还可映射到例如堆栈指针寄存器 100。同时，特殊的堆栈指针寄存器可以具有映射到它的多堆栈指针寄存器。例如，堆栈指针寄存器 100 和 96 都可以映射到堆栈指针寄存器 98。其他的映射也是有可能的。

图 5 的寄存器上下文还包括寄存器的分组。例如，通用寄存器分组成四个寄存器的组。在寄存器上下文 1 90 中，GPR 4-7 一起分组到寄存器组 102，GPR 8-11 一起分组到寄存器组 104，GPR 28-31 一起分组到寄存器组 106。因此，在图 5 的例子中，寄存器上下文 1 90 包括三个组（组 102、104 和 106），每个组具有四个寄存器，其中每个组可映射（作为一个组）到不同的寄存器上下文。在替换的实施例中，可分组任意数量和类型的寄存器。可替换地，每个单个的寄存器可以被认为是一个单独的组，取决于所需的颗粒度（granularity）。类似地，寄存器上下文 2 92 包括三个分组，每组四个寄存器：组 114 具有 GPR 4-7，组 116 具有 GPR 8-11，组 118 具有 GPR 28-31。同时，寄存器上

下文 3 94 包括三个分组，每组四个寄存器：组 108 具有 GPR 4-7，组 110 具有 GPR 8-11，组 112 具有 GPR 28-31。这些分组允许寄存器组在不同的寄存器上下文中映射。

例如，如箭头 120 所示，寄存器上下文 2 92 的分组 118 映射到寄存器上下文 1 90 的分组 106。如箭头 126 所示，寄存器上下文 3 94 的分组 112 也映射到寄存器上下文 1 90 的分组 106。也就是说，寄存器组 106 由所有的三个寄存器上下文（寄存器上下文 1 90、寄存器上下文 2 92 和寄存器上下文 3 94）所共享。因此，当执行处理 B 或处理 C 时，到当前寄存器上下文（分别是寄存器上下文 2 92 或寄存器上下文 3 94）的 GPR 28-31 的访问实际上导致到寄存器上下文 1 90 的 GPR 28-31 的访问。在图 5 中，如箭头 122 所示，寄存器上下文 1 90 的分组 104 映射到寄存器上下文 2 92 的分组 116。也就是说，组 116 的寄存器由寄存器上下文 1 90 和寄存器上下文 2 92 所共享。因此，当执行处理 A 时，到当前寄存器上下文的 GPR 8-11 的访问实际上导致到寄存器上下文 2 92 的 GPR 8-11 的访问。因此，可存在任意数量的映射，无论是单寄存器（诸如堆栈指针寄存器 96、98 或 100）还是寄存器组。同时，每个寄存器上下文可以具有映射到一个寄存器上下文的一些寄存器和映射到另一寄存器上下文的其他寄存器。同时，寄存器或寄存器组可具有映射到它的多寄存器上下文的寄存器。

每个寄存器上下文的映射在每个寄存器上下文的上下文控制寄存器中定义（例如，图 5 的上下文控制寄存器 128、130 和 132）。因此，图 2 的每个寄存器上下文 0-N 具有相应的上下文控制寄存器，其可能包括在每个寄存器上下文中（如在图 4 和 5）或者可能分别地储存（诸如在图 2 的控制寄存器文件 38 中）。图 6 说明了根据本发明一个实施例的上下文控制寄存器 140 的内容。上下文控制寄存器 140 可以是图 4 的上下文控制寄存器 77、79、75，或者图 5 的上下文控制寄存器 128、130 和 132。在一个实施例中，上下文控制寄存器是专用 32 比特寄存器，其具有多个不同的字段，用来控制寄存器的映射，并保持当前的、

替换的和保存的上下文信息。

上下文控制寄存器 140 的比特 0 对应于上下文使能字段 142，其使多寄存器上下文能够使用。例如，如果上下文使能字段 142 为否定，则仅使能一个上下文，忽略上下文控制 140 中的所有其他控制字段，并将当前的上下文设置为缺省寄存器上下文（在图 2 所示的实施例中是寄存器上下文 0 34）。如果声明上下文使能字段 142，则使能多个上下文。比特 3-5 对应于上下文数量字段 144，其是只读字段，表示硬件所支持的最高上下文数量。在图 6 的例子中，值 000 表示支持一个上下文，值 111 表示硬件支持八个上下文。如果数据处理系统 10 可支持 8 个以上的寄存器上下文，则可使用附加比特用于上下文数量字段 144。但是，在图 6 的实施例中，将假定支持最大的 8 个寄存器上下文。

比特 6-8 对应于当前上下文字段 146，其定义了当前使能的寄存器上下文。在一个实施例中，当复位以表示缺省寄存器上下文是寄存器上下文 0 时该字段被清零。当前上下文字段 146 对应于上面讨论的、可以各种不同方式提供的上下文选择器，诸如通过图 2 的向量偏移发生器 39 提供。因此，当上下文切换（由中断或处理切换引起）时，现有上下文字段 146 被设置为如上下文选择器所表示的新的寄存器上下文。例如，参看图 5，如果数据处理系统 10 当前执行处理 A，则在上下文切换到处理 B 时，上下文选择器表示寄存器上下文 2，值 2 写入到寄存器上下文 2 92 的上下文控制寄存器的当前上下文字段。

注意到，尽管每个寄存器上下文具有其自己的上下文控制寄存器，某些字段可以在不同上下文控制寄存器中间共享。例如，可以实现单独上下文使能比特、单独上下文数量字段和单独当前上下文字段，由所有上下文控制寄存器使用，因为值在不同上下文控制寄存器中间总是相同的。替换的实施例可使用上下文使能或上下文数量字段或当前上下文字段用于每个上下文控制寄存器，但是通过使用用于每个的单独共享字段可以减少硬件需求。

比特 9-11 对应于保存的上下文字段 148，其定义了先前使能的上下文。注意到，该字段也可以在复位时清零。因此，在从处理 A 切换到处理 B 的上述例子中（参看图 5），当前上下文字段设置为 2（表示寄存器上下文 2 92），保存的上下文字段设置为 1（表示先前上下文，寄存器上下文 1 90）。

比特 12-14 对应于替换的上下文字段 150，其定义了替换的使能上下文，其用于为寄存器分组定义上下文映射。比特 15-18 对应于映射字段 151。比特 15 对应于寄存器组 A（定义为 GPR 4-7），比特 16 对应于寄存器组 B（定义为 GPR 8-11），比特 17 对应于寄存器组 C（定义为 GPR 16-23），比特 18 对应于寄存器组 D（定义为 GPR 27-31）。每个寄存器组 A-D 可独立地通过声明对应比特而使能。例如，如果声明比特 15，组 A 使能，使组 A 映射到由替换上下文字段定义的寄存器上下文。如果比特 15 为否定，组 A 不映射。类似地，如果声明比特 16、17 或 18，则对应的寄存器组（分别为 B、C 或 D）映射到由替换上下文字段定义的寄存器上下文。如果比特 16、17 或 18 为否定，则对应的寄存器组（分别为 B、C 或 D）不映射。因此，参看图 5，寄存器上下文 1 90 的上下文控制寄存器在其替换上下文字段中包括一个 2，表示所选寄存器组映射到寄存器上下文 2 92。同时，声明比特 16（对应于具有 GPR 8-11 的组 B），使得寄存器上下文 1 90 的分组 104 映射到寄存器上下文 2 92 的分组 116。

在示例上下文控制寄存器，图 6 的上下文控制寄存器 140 中，单独的替换上下文字段可用，每个组（A-D）可使能以映射到相同的替换寄存器上下文。也就是说，如果分组 A 映射到特殊寄存器上下文，则分组 C-D 只能映射到相同的上下文。但是，在替换实施例中，每个分别的寄存器分组（诸如分组 A-D）可以具有相应的替换上下文字段，使它们能够映射到不同的替换寄存器上下文。另外，可将分别的替换上下文字段用于分组的组（例如，一个替换上下文字段用于分组 A 和

B，另一个用于 C 和 D）。同时，分组可以以任意方式定义。例如，每个分组可以具有多于或少于 4 个的寄存器，每个分组可以是一个寄存器，或者每个分组可以具有不同数量的寄存器。同时，在替换实施例中，可使用更多或更少的分组用于更多或更少的替换上下文字段。因此，图 6 的上下文控制寄存器 140 只是一个例子。同时，每个字段可使用更多或更少的比特，按需要，以表示字段的值。上下文控制 140 包括未使用的比特 1、2、19-23 和 28-31；但是，替换实施例可以不包括未使用的比特或可以需要多个寄存器来储存上下文控制信息。

上下文控制寄存器 140 的比特 24 对应于堆栈指针上下文使能字段 152，其使能堆栈指针的映射，如参看图 4 和图 5 讨论的。比特 25-27 对应于堆栈指针上下文选择字段 154，其为堆栈指针选择替换寄存器上下文。因此，如果声明比特 24，堆栈指针映射到堆栈指针上下文选择字段 154 所指示的寄存器上下文；但是，如果否定，则堆栈指针不映射（即，其保持由当前上下文字段 146 所定义的当前上下文）。堆栈指针上下文选择字段 154 是 3 比特值，可表示在数据处理系统 10 中的 8 个寄存器上下文中的哪个用作堆栈指针的替换上下文。例如，值 000 可对应于寄存器上下文 0，值 111 对应于寄存器上下文 7。因此，如果堆栈指针上下文选择字段 154 设置为 001，（且声明堆栈指针上下文使能字段 152）则当前上下文的堆栈指针映射到寄存器上下文 1。例如，参看图 4，声明寄存器上下文 2 74 的上下文控制寄存器的堆栈指针上下文使能字段，堆栈指针上下文选择字段设置为 001，表示堆栈指针寄存器 80 映射到寄存器上下文 1 72 的堆栈指针寄存器 78。同时注意到，替换实施例可包括除允许共享堆栈指针之外的允许映射其他个体寄存器的字段。

上下文控制寄存器可以通过各种不同方式来编程。例如，在一个实施例中，每个上下文控制寄存器可以是用户直接编程的。可替换地，上下文控制寄存器可以使用当前上下文控制寄存器的替换上下文字段来间接映射。例如，在一个实施例中，当上电或复位时，数据处理系

统缺省为寄存器上下文 0。替换上下文字段然后可被设置为表示哪个寄存器上下文的上下文控制寄存器要被编程的值。例如，在寄存器上下文 0 中，将值 2 写入寄存器上下文 0 的上下文控制寄存器的替换上下文字段，允许通过专用寄存器访问寄存器上下文 2 的上下文控制寄存器的编程。在所有上下文控制字段都被编程之后，它们都可以同时使能（在具有单个共享上下文使能字段的情况中是通过声明该比特来实现的）。同时，在一个实施例中，可以在上下文控制寄存器的编程过程中切断中断处理。

注意到，参考特殊字段和比特位置描述了上下文控制寄存器 140。注意到，替换实施例可以包括更多或更少的字段，按需要，每个字段可以包括更多或更少的比特。同时，在替换实施例中，上下文控制寄存器可以位于数据处理系统 10 中的任意位置，或者可以位于数据处理系统 10 的外部。

因此，可以认识到上下文控制寄存器可怎样通过减少的开销而用于提供灵活上下文选择。当数据处理系统 10 内上下文切换时，新寄存器上下文的上下文控制寄存器更新。例如，新寄存器上下文写到当前上下文字段，先前的上下文写到保存的上下文字段，当操作在新的寄存器上下文内时，使用映射字段内提供的寄存器映射。寄存器映射允许不同的寄存器上下文共享寄存器值。同时，寄存器映射允许访问当前寄存器上下文之外的其他寄存器上下文，如当前寄存器上下文的上下文控制寄存器所定义的。同时，用户可编程上下文控制寄存器允许在怎样定义映射方面的灵活性。因此，在此所述的本发明的一个方面提供一种灵活的机制，将部分替换寄存器上下文映射到当前寄存器上下文（反之亦然），并且提供公用堆栈指针在多上下文中间的灵活共享，从而带来增强的实时性能。通过将部分当前上下文映射到替换上下文，在操作上下文之间传输信息值的开销可以被去除或者消除，从而带来增强的性能或灵活性。

在前面的说明书中，通过特定实施例描述了本发明。但是，本领域普通技术人员认识到，在不背离本发明如权利要求书所阐述的范围的前提下，可作出各种不同修改和改变。例如，框图可以具有与所说明的不同的框，可以有更多或更少的框，或者不同的布置。因此，说明书和附图只是说明性的而非限制性的，所有这样的修改都应该包括在本发明的范围中。

上面参考特定实施例描述了利益、其他优点和问题的解决方案。但是，利益、其他优点、问题的解决方案和任何能够带来任何利益、优点或使解决方案更加显著的元素不应被视为任意或所有权利要求的关键的、所需的或必需的特征和元素。如在此使用的，术语“包括、包含（comprises、comprising）”或其任意其他的变化都意旨涵盖非排他性的包涵，因此，包括一系列元素的过程、方法、物品或装置不仅包括所列出的元素，还包括其他未明确列出或这样的过程、方法、物品或装置所固有的元素。

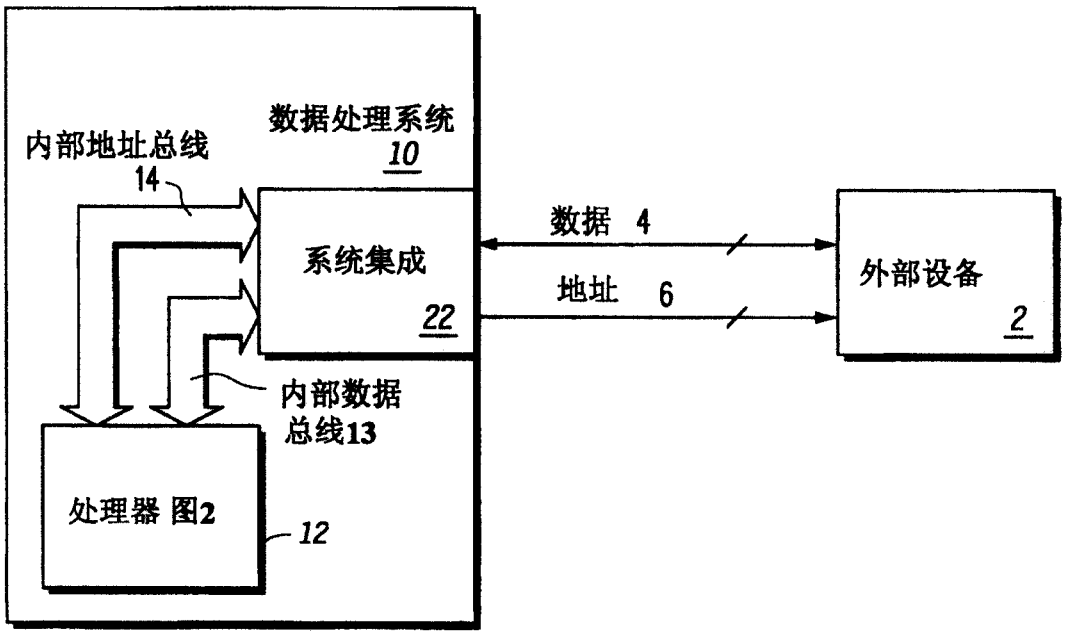


图1

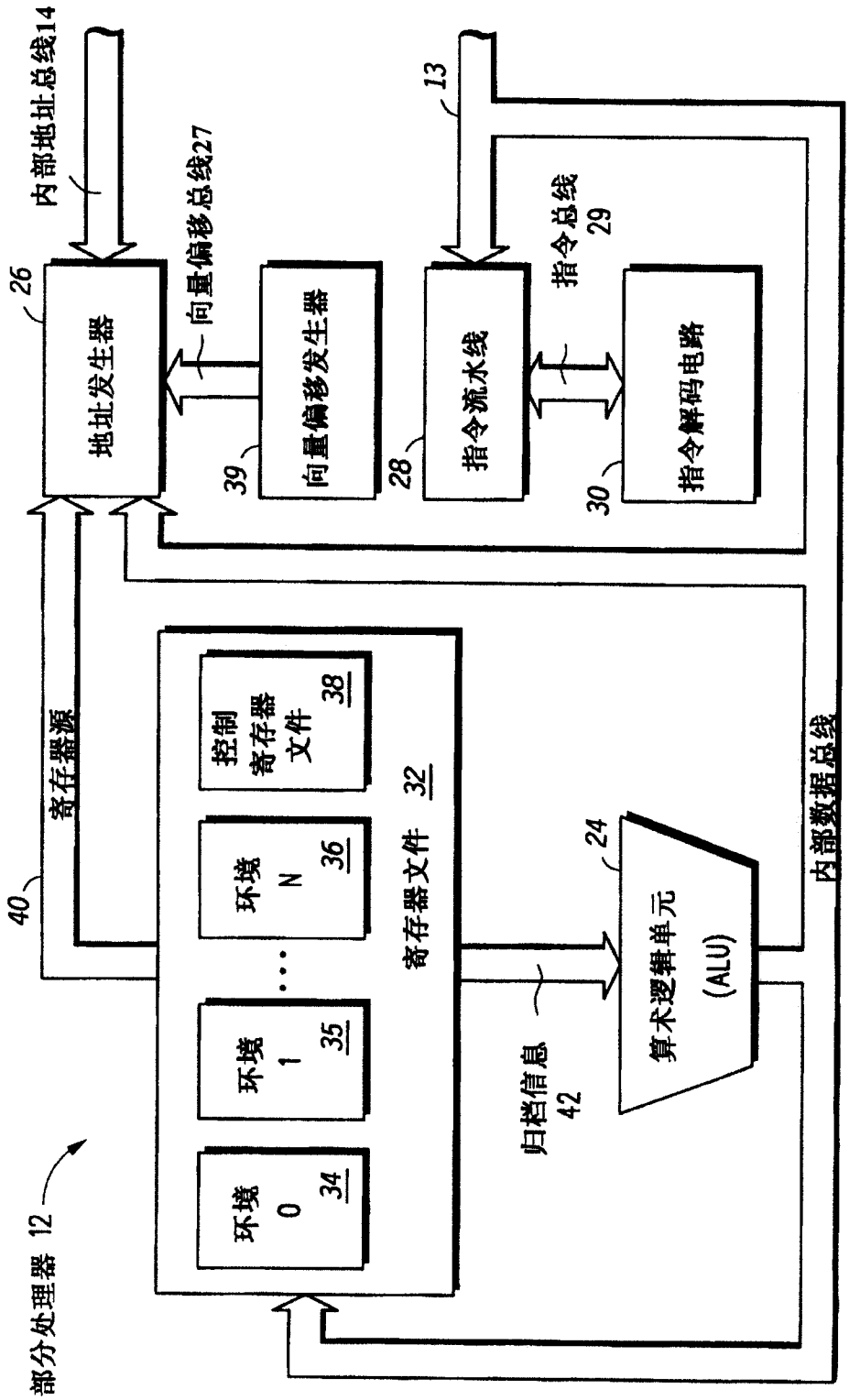

内部地址总线14
26
地址发生器
向量偏移总线27
39
向量偏移发生器
13
指令总线
29
28
指令流水线
30
指令解码电路
寄存器源
40
控制
寄存器
文件
38
寄存器文件 32
环境
N
36
环境
1
35
环境
0
34
24
算术逻辑单元
(ALU)
归档信息
42
内部数据总线
部分处理器 12

图2

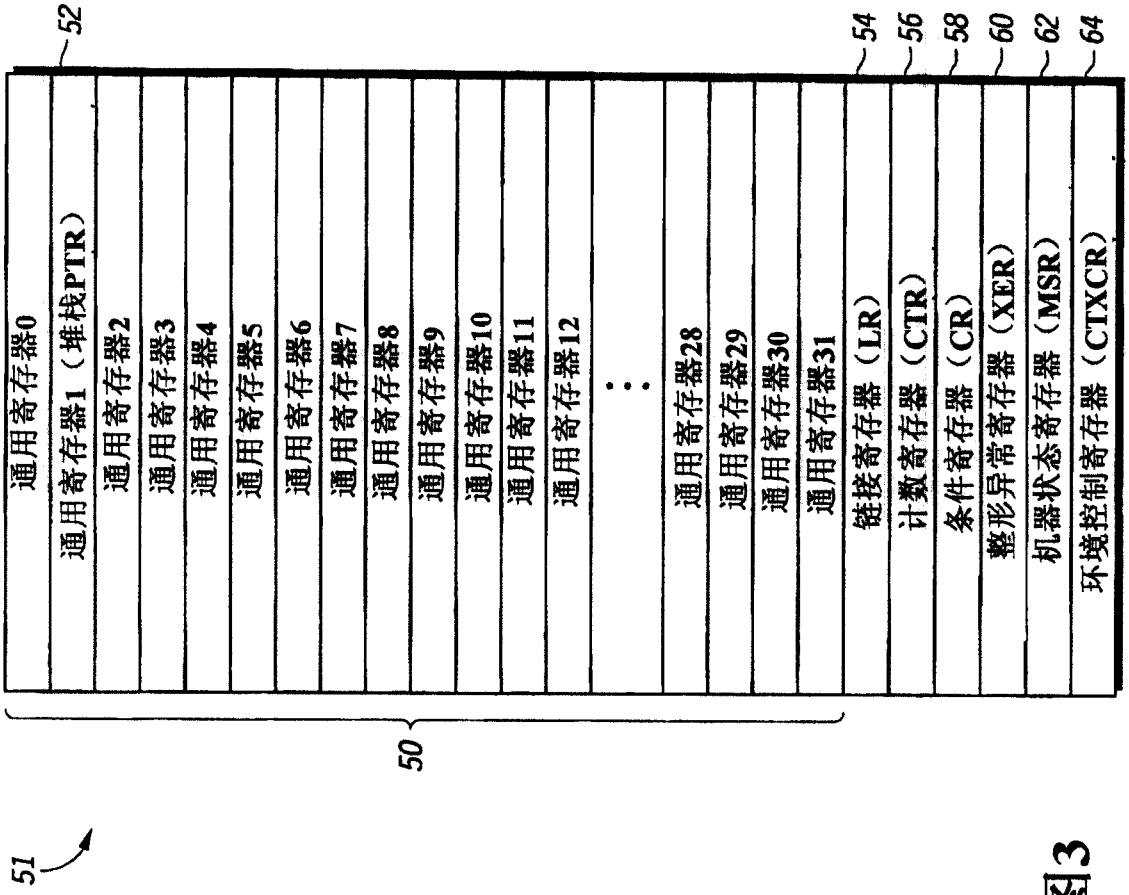

通用寄存器0
通用寄存器1（堆栈PTR）
通用寄存器2
通用寄存器3
通用寄存器4
通用寄存器5
通用寄存器6
通用寄存器7
通用寄存器8
通用寄存器9
通用寄存器10
通用寄存器11
通用寄存器12
通用寄存器28
通用寄存器29
通用寄存器30
通用寄存器31
链接寄存器（LR）
计数寄存器（CTR）
条件寄存器（CR）
整形异常寄存器（XER）
机器状态寄存器（MSR）
环境控制寄存器（CTXCR）
52
54
56
58
60
62
64
50
51

图3

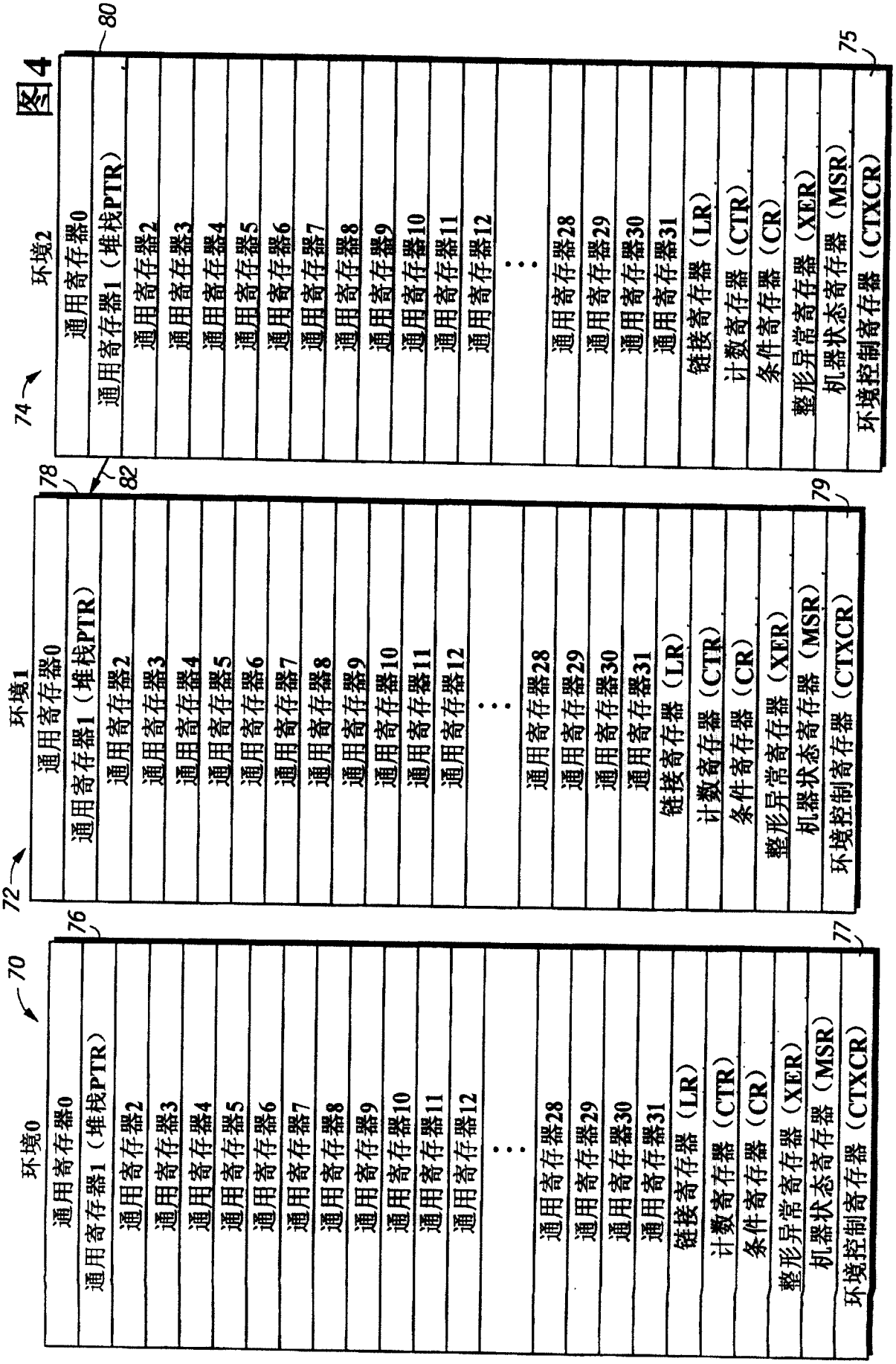


图4
环境0
70
通用寄存器0
通用寄存器1（堆栈PTR）
通用寄存器2
通用寄存器3
通用寄存器4
通用寄存器5
通用寄存器6
通用寄存器7
通用寄存器8
通用寄存器9
通用寄存器10
通用寄存器11
通用寄存器12
通用寄存器28
通用寄存器29
通用寄存器30
通用寄存器31
链接寄存器（LR）
计数寄存器（CTR）
条件寄存器（CR）
整形异常寄存器（XER）
机器状态寄存器（MSR）
环境控制寄存器（CTXCR）
76
77
环境1
72
78
79
环境2
74
80
75
82

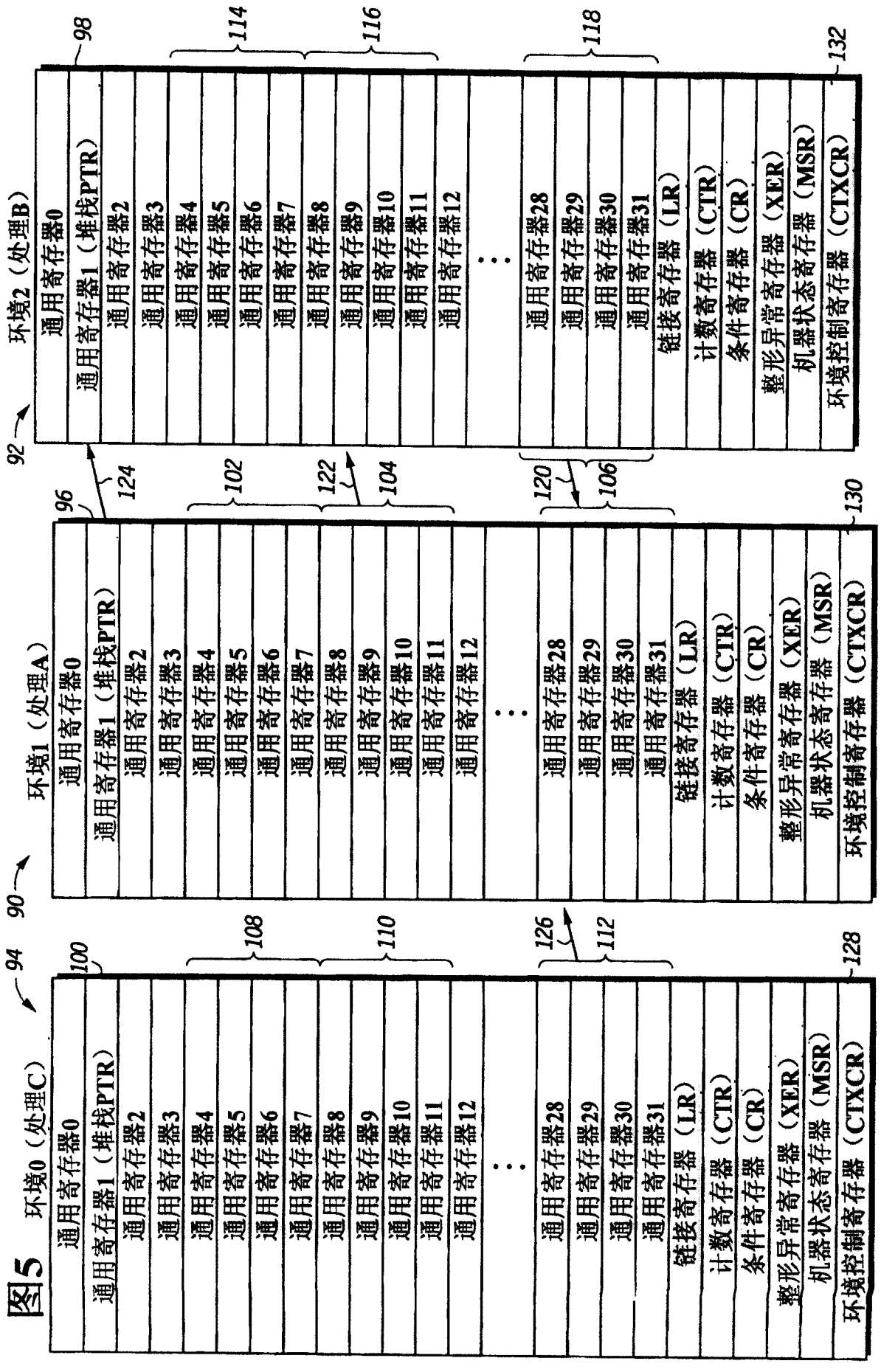

环境0（处理C）
环境1（处理A）
环境2（处理B）
通用寄存器0
通用寄存器1（堆栈PTR）
通用寄存器2
通用寄存器3
通用寄存器4
通用寄存器5
通用寄存器6
通用寄存器7
通用寄存器8
通用寄存器9
通用寄存器10
通用寄存器11
通用寄存器12
通用寄存器28
通用寄存器29
通用寄存器30
通用寄存器31
链接寄存器（LR）
计数寄存器（CTR）
条件寄存器（CR）
整形异常寄存器（XER）
机器状态寄存器（MSR）
环境控制寄存器（CTXCR）
90
92
94
96
98
100
102
104
106
108
110
112
114
116
118
120
122
124
126
128
130
132

图5

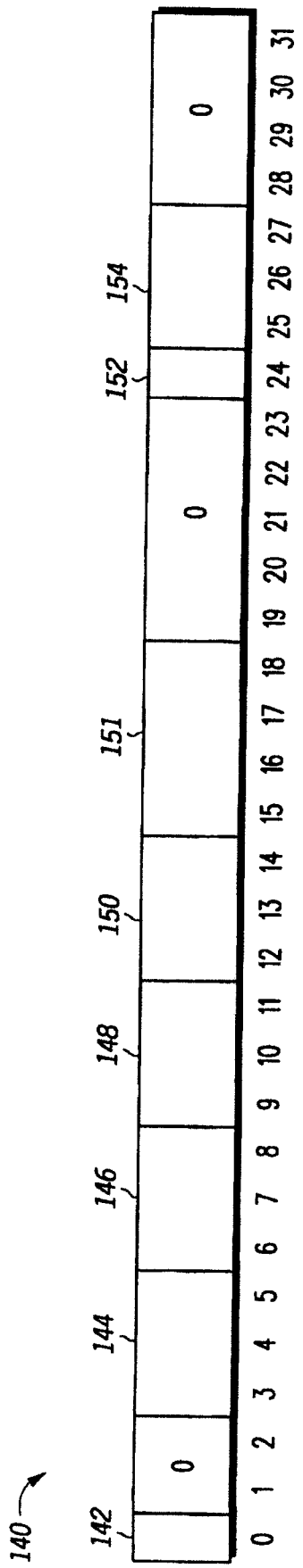

140
142
144
146
148
150
151
152
154
0
0
0
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

图6