[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/30 (2006.01)
G06F 9/38 (2006.01)

[12] 发明专利说明书

专利号 ZL 200410069611.8

[45] 授权公告日 2007 年 3 月 28 日

[11] 授权公告号 CN 1307536C

[22] 申请日 1998.9.4
[21] 申请号 200410069611.8
分案原申请号 98118581.9
[30] 优先权
[32] 1997. 9. 5 [33] US [31] 08/924,518
[73] 专利权人 摩托罗拉公司
地址 美国伊利诺斯
[72] 发明人 威廉姆·C.·莫耶
约翰·阿兰德斯
杰夫里·W.·斯考特
[56] 参考文献
EP0280821A2 1988.9.7
EP0092429A 1983.10.26
EP0261685A2 1988.3.30
审查员 曲 颖

[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 康建峰

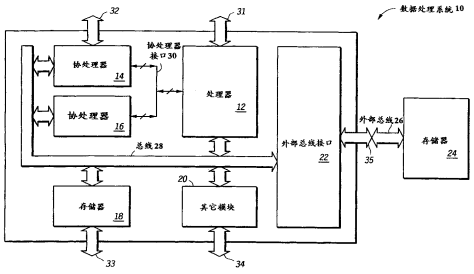
权利要求书 4 页 说明书 20 页 附图 25 页

[54] 发明名称

将一个处理器与一个协处理器相接口的方法和装置

[57] 摘要

本发明涉及一种将一个处理器与一个协处理器相接口的方法和装置。 其中，一个处理器(12)向协处理器(14)的接口，它支持多个协处理器(14，16)，用于使用编译器生成软件类型函数调用和返回，指令执行，以及可变加载和存储接口指令。 在一个双向共享总线(28)上，或是通过寄存器窥探和广播显式地，或者通过函数调用和返回以及可变加载和存储接口指令隐式地，在处理器(12)和协处理器(14)之间移动数据。 在断言一个执行信号之前，通过否定一个译码信号，指示已译码的指令删除来提供流水线操作。



1. 一种由具有包括多个寄存器的寄存器文件的处理器经由协处理器通信总线广播所述寄存器文件中的所有写入事务的方法，所述方法包括步骤：

接收将要写入所述寄存器文件的操作数；

选择所述寄存器文件中所述多个寄存器中的一个；以及

把以下各项经由所述协处理器通信总线提供给所述寄存器文件：所述将要写入所述寄存器文件的操作数、表明所述选择的所述寄存器文件中所述多个寄存器中的一个的第一控制信号、以及请求将所述操作数写入所述选择的所述多个寄存器中的一个的第二控制信号。

2. 一种由第一处理器经由协处理器通信总线监控到第二处理器中的的包括多个寄存器的寄存器文件中的所有写入事务的方法，所述方法包括步骤：

经由所述协处理器通信总线从所述第二处理器接收以下各项：将要写入所述寄存器文件的操作数、表明选择的所述寄存器文件中所述多个寄存器中的一个的第一控制信号、以及请求将所述操作数写入所述选择的所述多个寄存器中的一个的第二控制信号；以及

响应接收到所述操作数、所述第一控制信号、以及所述第二控制信号，执行预定的操作。

3. 一种由具有包括多个寄存器的寄存器文件的第一处理器经由协处理器通信总线广播到所述寄存器文件中的所有写入事务以便由第二处理器监控的方法，所述方法包括步骤：

在所述第一处理器中：

接收将要写入所述寄存器文件的操作数；

选择所述寄存器文件中所述多个寄存器中的一个；以及

把以下各项经由所述协处理器通信总线提供给所述寄存器文件：所述将要写入所述寄存器文件的操作数、表明所述选择的所述

寄存器文件中所述多个寄存器中的一个的第一控制信号、以及请求将所述操作数写入所述选择的所述多个寄存器中的一个的第二控制信号；

在所述第二处理器中：

经由所述协处理器通信总线从所述第一处理器接收以下各项：将要写入所述寄存器文件的操作数、表明所述选择的所述寄存器文件中所述多个寄存器中的一个的第一控制信号、以及请求将所述操作数写入所述选择的所述多个寄存器中的一个的第二控制信号；以及

响应接收到所述操作数、所述第一控制信号、以及所述第二控制信号，执行预定的操作。

4. 一种处理器，包括：

多个寄存器；

用于执行到多个寄存器中的一个的写入操作的电路；

用于为到所述多个寄存器中的一个的写入操作提供操作数的导线；以及

用于与协处理器通信总线进行通信的端口，所述端口包括：

至少一个第一协处理器通信总线信号端，用于提供由所述处理器生成的至少一个第一协处理器通信总线信号，所述至少一个第一协处理器通信总线信号表示在所述写入操作期间写入多个寄存器中的哪一个；

至少一个第二协处理器通信总线信号端，用于提供由所述处理器生成的至少一个第二协处理器通信总线信号，所述至少一个第二协处理器通信总线信号表示到所述多个寄存器中的一个的所述写入操作的发生；以及

至少一个第三协处理器通信总线信号端，用于提供由所述处理器生成的至少一个第三协处理器通信总线信号，所述至少一个第三协处理器通信总线信号在所述写入操作期间提供将要写入所述多个寄存器中的一个的所述操作数。

5. 一种数据处理系统，包括通过协处理器通信总线耦合的处理器和协处理器，

其中，所述处理器包括：

多个寄存器；

用于执行到多个寄存器中的一个的写入操作的电路；

用于为到所述多个寄存器中的一个的写入操作提供操作数的导线；以及

用于通过所述协处理器通信总线与所述协处理器进行通信的处理器端口，所述处理器端口包括：

至少一个第一协处理器通信总线信号端，用于提供由所述处理器生成的至少一个第一协处理器通信总线信号，所述至少一个第一协处理器通信总线信号表示在所述写入操作期间写入多个寄存器中的哪一个；

至少一个第二协处理器通信总线信号端，用于提供由所述处理器生成的至少一个第二协处理器通信总线信号，所述至少一个第二协处理器通信总线信号表示到所述多个寄存器中的一个的所述写入操作的发生；以及

至少一个第三协处理器通信总线信号端，用于提供由所述处理器生成的至少一个第三协处理器通信总线信号，所述至少一个第三协处理器通信总线信号在所述写入操作期间提供将要写入所述多个寄存器中的一个的所述操作数；

并且，其中所述协处理器包括：

用于与所述协处理器通信总线进行通信的协处理器端口，所述协处理器端口包括：

至少一个第四协处理器通信总线信号端，用于接收由所述处理器生成的所述至少一个第一协处理器通信总线信号，所述至少一个第一协处理器通信总线信号表示在所述写入操作期间写入多个寄存器中的哪一个；

至少一个第五协处理器通信总线信号端，用于接收由所述处理

器生成的至少一个第二协处理器通信总线信号，所述至少一个第二协处理器通信总线信号表示到所述多个寄存器中的一个的所述写入操作的发生；以及

至少一个第六协处理器通信总线信号端，用于接收由所述处理器生成的至少一个第三协处理器通信总线信号，所述至少一个第三协处理器通信总线信号在所述写入操作期间提供将要写入所述多个寄存器中的一个的所述操作数。

将一个处理器与一个协处理器相接口的方法和装置

本发明申请为 1998 年 9 月 4 日提出的、发明名称为“将一个处理器与一个协处理器相接口的方法和装置”的第 98118581.9 号的分案申请。

技术领域

本发明涉及具有一个处理器和至少一个协处理器的一个数据处理系统，尤其涉及将一个处理器与一个协处理器相接口的一种方法和装置。

背景技术

通过专用的和专门的硬件功能元件，扩展一个基准体系结构处理器功能的能力是可测量的和可扩展的体系结构的一个重要方面。

用于扩展一个基准体系结构处理器功能的优选方式中的一种是通过使用协处理器。这些协处理器通常是在处理器的指引下运行的专用的单用途处理器。协处理器一个常用的用途是用作数学协处理器，有选择的为没有直接提供浮点能力的体系结构提供这个功能。这种数学协处理器的例子是 Intel 8087 和 80287。协处理器的其它可能的用途或类型包括：乘-累加器，调制器/解调器（调制解调器），数字信号处理器（DSP），维特比计算器，加密处理器，图像处理器和向量处理器。

对于协处理器已经有两种不同的方法。一方面，对于数字设备公司（DEC）的 PDP-11 系列计算机，将浮点单元与它的主处理器紧密耦合。出现的一个问题是这种紧密耦合要求主处理器要知道协处理器计算的实际数量。这使电路设计复杂起来，这样，将一个新协处理器

加到一个集成系统上的扩展是一个主要的工程问题。

可选的实现方法是将协处理器与主处理器松散耦合。这样做的益处是从主处理器中提取并分离出协处理器的操作，并且能够真正减少将一个新的协处理器与一个已存在的处理器相集成时所要做的工作。但是这也必然要花费代价。性能的损失是这种方法的一个问题。由这种松耦合所引起的这种类型的性能命中的一个问题是调用这样一个协处理器时无亏损点也相应地增加。这样，对于协处理器，许多其它的吸引人的应用的效能价格比并不合算。此外，这样一种方法经常要求使用一条总线，还有所有的相应的附加电路和芯片区域。

这样具有一个协处理器接口就是非常重要的，将接口紧密耦合，使得接口的使用足够快，甚至在调用相对简单的函数时也占有优势，同时将接口提取成这样一个扩展，从任何给定的协处理器的细节中将处理器的体系结构尽可能多的分离出来。后者中的一部分包括使接口程序设计器友好，目的是可以方便使用软件而不是硬件设计新的协处理器应用。

发明内容

本发明提供一种用于具有包括多个寄存器的寄存器文件的处理器经由协处理器通信总线广播所述寄存器文件中的所有事务的方法，所述方法包括步骤：接收将要写入所述寄存器文件的操作数；选择所述寄存器文件中所述多个寄存器中的一个；以及提供给所述寄存器文件并经由所述协处理器通信总线：所述将要写入所述寄存器文件的操作数、表明所述选择的所述寄存器文件中所述多个寄存器中的一个的第一控制信号、以及请求将所述操作数写入所述选择的所述多个寄存器中的一个的第二控制信号。

本发明提供一种用于第一处理器经由协处理器通信总线监控所有写入事务到第二处理器中的包括多个寄存器的寄存器文件中的方法，所述方法包括步骤：经由所述协处理器通信总线从所述第二处理器接收：将要写入所述寄存器文件的操作数、表明选择的所述寄存器

文件中所述多个寄存器中的一个的第一控制信号、以及请求将所述操作数写入所述选择的所述多个寄存器中的一个的第二控制信号；以及响应收到的所述操作数、所述第一控制信号、以及所述第二控制信号，执行预定的操作。

本发明提供一种用于具有包括多个寄存器的寄存器文件的第一处理器经由协处理器通信总线广播由第二处理器监控的所述寄存器文件中的所有写入事务的方法，所述方法包括步骤：在所述第一处理器中：接收将要写入所述寄存器文件的操作数；选择所述寄存器文件中所述多个寄存器中的一个；以及提供给所述寄存器文件并经由所述协处理器通信总线：所述将要写入所述寄存器文件的操作数、表明所述选择的所述寄存器文件中所述多个寄存器中的一个的第一控制信号、以及请求将所述操作数写入所述选择的所述多个寄存器中的一个的第二控制信号；在所述第二处理器中：经由所述协处理器通信总线从所述第一处理器接收：将要写入所述寄存器文件的操作数、表明所述选择的所述寄存器文件中所述多个寄存器中的一个的第一控制信号、以及请求将所述操作数写入所述选择的所述多个寄存器中的一个的第二控制信号；以及响应收到的所述操作数、所述第一控制信号、以及所述第二控制信号，执行预定的操作。

本发明提供一种处理器，包括：多个寄存器；用于执行到多个寄存器中的一个的写入操作的电路；用于为到所述多个寄存器中的一个的写入操作提供操作数的导线；以及用于与协处理器通信总线进行通信的端口，所述端口包括：至少一个第一协处理器通信总线信号端，用于提供由所述处理器生成的至少一个第一协处理器通信总线信号，所述至少一个第一协处理器通信总线信号表示在所述写入操作期间写入多个寄存器中的哪一个；至少一个第二协处理器通信总线信号端，用于提供由所述处理器生成的至少一个第二协处理器通信总线信号，所述至少一个第二协处理器通信总线信号表示到所述多个寄存器中的一个的所述写入操作的发生；以及至少一个第三协处理器通信总线信号端，用于提供由所述处理器生成的至少一个第三协处理器通信

总线信号，所述至少一个第三协处理器通信总线信号在所述写入操作期间提供将要写入所述多个寄存器中的一个的所述操作数。

本发明提供一种数据处理系统，包括通过协处理器通信总线耦合的处理器和协处理器，其中，所述处理器包括：多个寄存器；用于执行到多个寄存器中的一个的写入操作的电路；用于为到所述多个寄存器中的一个的写入操作提供操作数的导线；以及用于通过所述协处理器通信总线与所述协处理器进行通信的处理器端口，所述处理器端口包括：至少一个第一协处理器通信总线信号端，用于提供由所述处理器生成的至少一个第一协处理器通信总线信号，所述至少一个第一协处理器通信总线信号表示在所述写入操作期间写入多个寄存器中的哪一个；至少一个第二协处理器通信总线信号端，用于提供由所述处理器生成的至少一个第二协处理器通信总线信号，所述至少一个第二协处理器通信总线信号表示到所述多个寄存器中的一个的所述写入操作的发生；以及至少一个第三协处理器通信总线信号端，用于提供由所述处理器生成的至少一个第三协处理器通信总线信号，所述至少一个第三协处理器通信总线信号在所述写入操作期间提供将要写入所述多个寄存器中的一个的所述操作数；并且，其中所述协处理器包括：用于与所述协处理器通信总线进行通信的协处理器端口，所述协处理器端口包括：至少一个第四协处理器通信总线信号端，用于接收由所述处理器生成的所述至少一个第一协处理器通信总线信号，所述至少一个第一协处理器通信总线信号表示在所述写入操作期间写入多个寄存器中的哪一个；至少一个第五协处理器通信总线信号端，用于接收由所述处理器生成的至少一个第二协处理器通信总线信号，所述至少一个第二协处理器通信总线信号表示到所述多个寄存器中的一个的所述写入操作的发生；以及至少一个第六协处理器通信总线信号端，用于接收由所述处理器生成的至少一个第三协处理器通信总线信号，所述至少一个第三协处理器通信总线信号在所述写入操作期间提供将要写入所述多个寄存器中的一个的所述操作数。

附图说明

通过下面的详细描述和附图能够更清晰的理解本发明的特点和益处，其中相同的数字用来表示相同和相对应的部分，其中：

图 1 是依据本发明的一个数据处理系统的一种实施方式的框图；

图 2 是图 1 中处理器的一部分的框图；

图 3 是图 1 中协处理器的一部分的一种实施方式的框图；

图 4 是依据本发明的一个寄存器窥探操作的时序图；

图 5 是用于指令信号交换的基本指令接口操作的时序图；

图 6 是在使用 H-BUSY*信号控制协处理器接口指令执行时，指令接口操作的时序图；

图 7 是指令删除的时序图；

图 8 是指令流水线停顿的一个例子的时序图；

图 9 是一个没有停顿的背对背操作的一个例子的时序图；

图 10 是具有内部流水线停顿的背对背操作的时序图；

图 11 是具有 H-BUSY*停顿的背对背协处理器接口 30 的时序图；

图 12 是图解响应协处理器接口操作码的译码和测试操作由一个协处理器断言的 H-EXCP*信号的一个例子的时序图。

图 13 是图解响应协处理器接口操作码的译码和测试操作当删除协处理器接口指令时，由一个协处理器断言的 H-EXCP*信号的一个例子的时序图。

图 14 是图解一个已经断言了 H-BUSY*以延迟一个协处理器接口操作码运行的例子的时序图。

图 15 是图解与 H-CALL 原语相关的寄存器传送的一个例子的时序图。

图 16 是图解与 H-LD 原语相关的寄存器传送的一个例子的时序图。

图 17 是图解协处理器接口的一个 H-LD 的传送顺序的时序图。

图 18 是图解在一个存储器存取造成一个存取异常的时候协议的时序图。

图 19 是图解与 H-ST 原语相关联的传送的一个例子的时序图；

图 20 是图解延迟的存储数据的传送的一个例子的时序图；

图 21 是图解在存储造成一个存取错误的时候协议信号的时序图；

图 22 图解依据本发明的 H-CALL 原语的一个指令格式；

图 23 图解依据本发明的 H-RET 原语的一个指令格式；

图 24 图解依据本发明的 H-EXEC 原语的一个指令格式；

图 25 图解依据本发明的 H-LD 原语的一个指令格式；

图 26 图解依据本发明的 H-ST 原语的一个指令格式。

具体实施方式

在下面的描述中，采用了大量的具体的细节，例如特定的字或字节长度，等等，以提供对本发明的一个透彻的理解。然而对本技术领域中的熟练技术人员来说，可能无须这些特定细节就能够实现本发明。在另外的例子中，以框图的形式来表示电路，目的是不在不必要的细节中混淆本发明。对于大部分，已经省略了有关时序的考虑和相类似的细节，因为对于完整理解本发明这些细节并不是必需的，而且它们也在相关领域的技术人员的技术范围之内。

术语“总线”用来指多个信号或导线，可以使用它们来传送一个或多个不同类型的信息，例如数据，地址，控制或状态。术语“断言(assert)”和“否定(negate)”是用来将一个信号，状态为或相类似设备的描述相应的引用为它的逻辑真或逻辑假状态。如果逻辑真状态是一个逻辑电平 1，那么逻辑假状态将是一个逻辑电平 0。并且如果逻辑真状态是一个逻辑电平 0，那么逻辑假状态将是一个逻辑电平 1。

图 1 是一个图解数据处理系统 10 的一种实施方式的框图，该数据处理系统包括一个处理器 12，一个协处理器 14，一个协处理器 16，一个存储器 18，其它模块 20 和外部总线接口 22，它们都是通过总线 28 双向耦合的。本发明的可选实施方式可能仅仅包括一个协处理器 14，两个协处理器 14 和 16，或更多的协处理器（没有表示）。外部

总线接口 22 通过集成电路终端 35 与外部总线 26 双向耦合。存储器 24 与外部总线 26 双向耦合。可以选择通过集成电路终端 31 将处理器 12 与数据处理系统 10 外部耦合。可以选择通过集成电路终端 32 将协处理器 14 与数据处理系统 10 外部耦合。可以选择通过集成电路终端 32 将存储器 18 与数据处理系统 10 外部耦合。可以选择通过集成电路终端 34 将其它模块 20 与数据处理系统 10 外部耦合。通过协处理器接口 30 将处理器 12 与协处理器 14 和协处理器 16 双向耦合。

图 2 是一个图解图 1 中处理器 12 的一部分的框图。在一种实施方式中，处理器 12 包括控制电路 40，指令译码电路 42，指令管道 44，寄存器 46，算术逻辑单元（ALU）48，锁存多路转换器（MUX）50，锁存多路转换器（MUX）52，和多路转换器（MUX）54。在本发明的一种实施方式中，协处理器接口 30 包括信号 60-71。时钟信号是由控制电路 40 生成的。协处理器运行信号 61 是由控制电路 40 生成的，并将该信号提供给协处理器 14 和 16。

管理状态信号 62 是由控制电路 40 生成的，并将该信号提供给协处理器 14 和 16。译码信号 63 是由控制电路 40 生成的，并将该信号提供给协处理器 14 和 16。协处理器忙信号 64 是由控制电路 40 从协处理器 14 或 16 中接收到的。执行信号 65 是由控制电路 40 生成的，并将该信号提供给协处理器 14 和 16。异常信号 66 是由控制电路 40 从协处理器 14 或协处理器 16 中接收到的。寄存器写（REGWR*）信号 67 是由控制电路 40 生成的，并将该信号提供给协处理器 14 和 16。寄存器信号（REG{4: 0}）68 是由控制电路 40 生成的，并将该信号提供给协处理器 14 和 16。出错信号（H-ERR*）69 是由控制电路 40 生成的，并将该信号提供给协处理器 14 和 16。数据选通信号（H-DS*）70 是由控制电路 40 生成的，并将该信号提供给协处理器 14 和 16。数据确认信号（H-DA*）是由控制电路 40 从协处理器 14 或协处理器 16 中接收到的。硬件数据端口信号（HDP{31: 0}）72 也被认为是协处理器接口 30 的一部分，在协处理器 14 和 16 与处理器 12 的内部电路之间它们是双向的。

在本发明的一种实施方式中，从/或向总线28提供多个信号，目的是在存储器18和/或存储器24中加载或存储数据。在一种实施方式中，这些信号包括一个传输请求信号（TREQ*）73，它是由控制电路40生成的并将该信号提供给总线28。通过总线28将传输出错确认信号（TEA*）74提供给控制电路40。通过总线28将传输确认信号（TA*）75提供给控制电路40。从总线28通过导线76将指令提供给指令管道44。通过导线76将数据提供给MUX54。驱动数据信号79能够使三态缓冲器95通过导线88和76提供来自锁存MUX52的数据。地址选择信号78使锁存MUX50能够通过导线77向总线28提供地址。MUX50的另外一个输入是由HDP信号（HDP{31: 0}）72提供的。MUX54的另外一个输入是通过ALU结果导线86提供的。将MUX54的输出，结果信号83提供给寄存器46和三态缓冲器96的输入。驱动HDP信号82能够使三态缓冲器96在HDP信号72之上驱动结果信号83。三态缓冲器96的输出也与锁存MUX52的输入相耦合。在本发明的可选实施方式中，寄存器46中可能包括任何数量的寄存器。提供结果信号83，作为锁存MUX50的一个输入。通过MUX54将结果信号83提供给寄存器46。结果选择信号（RESULT-SELECT）81选择在结果导线83之上驱动MUX54的哪一个输入。将源选择信号（SOURCE-SELECT）80提供给锁存MUX52，以选择在导线88上将哪一个信号驱动到三态缓冲器95。控制电路40提供控制信息，并通过导线91从寄存器46接收状态信息。控制电路40提供控制信号，并通过导线92从算术逻辑单元48接收状态信号。控制电路40提供控制信号并通过导线93从指令管道44和指令译码电路42接收状态信号。耦合指令管道44以通过导线89向指令译码电路42提供指令。指令译码电路42通过导线90向控制电路40提供译码后的指令信息。寄存器46通过导线84向算术逻辑单元48提供源操作数。寄存器46通过导线84，锁存MUX52，三态缓冲器95和导线76提供数据，将该数据存储在存储器18或存储器24中。寄存器46通过导线84，锁存MUX50，和地址导线77向存储器18或存储器24提供地址信息。寄存器46通

过导线 85 向算术逻辑单元 48 提供一个第二源操作数。

图 3 是一个图解协处理器 14 的一部分的一种实施方式的框图。在一种实施方式中，协处理器 14 包括控制电路 100，计算电路 102，和可选的存储电路 104。控制电路 100 通过协处理器接口 30 与处理器 12 双向耦合，其中协处理器接口 30 包括信号 60-72。在本发明的一种实施方式中，控制电路 100 包括译码电路 106，该译码电路 106 从处理器 12 接收运行信号 61 和译码信号 63。控制电路 100 提供控制信息，并通过导线 108 从可选的存储电路 104 中接收状态信息。控制电路 100 提供控制信息，并通过导线 109 从计算电路 102 中接收状态信息。计算电路 102 和可选的存储电路 104 通过导线 110 双向耦合。向或从总线 28 或集成的电路终端 32 提供一个或多个信号 110。控制电路 100 通过导线 112 从或向总线 28 或集成的电路终端 32 接收或提供信息。信号 72 可以与计算电路 102 和可选的存储电路 104 双向耦合。此外，信号 72 可以与总线 28 或集成的电路终端 32 双向耦合。在本发明的一种可选实施方式中，可能并没有实现的可选的存储电路 104。在本发明的一种实现了可选的存储电路 104 的实施方式中，这可以是使用寄存器，任何类型的存储器，包括锁存或可编程的逻辑阵列等等的任何类型的存储电路来实现。在本发明的可选实施方式中，计算电路 102 可以执行任何类型的逻辑或计算功能。

该系统通过一个外协处理器 14（或硬件加速器）提供对任务加速的支持，对于与特定应用相关的操作将外协处理器 14（或硬件加速器）最优化。对于执行一个人口数计算，或更复杂的功能例如一个 DSP 加速协处理器 14 或能够进行高速乘/累加操作的协处理器 14，这些外协处理器 14，16 可能就象一个协处理器 14 那么简单。

对于一个特定实现，由若干个机制中的一个或几个，在处理器 12 和协处理器 14 之间传送数据。这可被分为向协处理器 14 的传送，和从协处理器 14 的传送。

向协处理器 14 传送数据的的机制中的一个是寄存器窥探机制，它不涉及指令原语，但却是正常的处理器 12 操作的副产品。这包括

在接口中反映对于处理器的12个通用寄存器（“GPR”）46的更新，这样一个协处理器 14 能够监视对一个或多个处理器 12 寄存器的更新。对于一个内部寄存器或功能，如果一个协处理器14“覆盖”了一个 GPR46，那么这可能是适合的。在这种情况下，不要求从处理器12向一个协处理器14显式的传递参数。

也在基处理器12中提供指令原语以在外协处理器14，16和处理器12之间显式传送操作数和指令。此外还提供了一个信号交换机制，以允许控制指令和数据传送的速率。

注意到协处理器 14 的功能被设计为是特定实现的单元，这样在不同的实现中，可以自由改变一个给定单元的精确功能，即使可能存在相同的指令映射。

图 4 是一个图解一个寄存器窥探操作的时序图。为了避免向协处理器 14 或外监视器额外的传递参数，提供了一个寄存器窥探机制。这允许一个协处理器14实现处理器的12个通用寄存器46中的一个或多个的阴影拷贝（shadow copy）。这个能力是通过传送被写入到处理器 GPR46 中的值，和一个指示对于每一个 GPR 寄存器 46 也正被更新的指示值来实现的。对于每一个 GPR 更新，断言一个选通信号 REGWR*67。在 32 位的双向数据路径 HDP[31: 0]72 中传送该值，并且一个 5 位的寄存器数字总线提供了一个指向正在被更新的（REG[4: 0]）68 的实际的处理器寄存器 46 的指针。在一个正常的文件或一个可选的文件中寄存器编码可能是一个寄存器 46。在优选的实施方式中，通过 REG[4]==1 来指示可选的文件寄存器，由 REG[4]==0 来指示正常的文件寄存器。但是，要注意本发明不以任何方式依赖于寄存器组实际的分区。

一个协处理器14可能在内部将该值与一个目的寄存器46序号的指示一起锁存，以避免后面的显式移动。也可能由一个调试协处理器14来使用这个功能，以跟踪寄存器46或它的一个子集的状态。

一个专用的 12 位指令总线（H-OP[11: 0]）61 提供被发布给外协处理器 14 的协处理器接口 30 操作码。这条总线反映了该处理器的

操作码的低位的 12 位。没有反映高位的 4 位，因为它们总是 0b0100。也提供了一个管理状态指示器（H-SUP）62 来指示 PSR（S）位的当前状态，指示处理器是运行在管理状态还是用户状态。这对于将某一协处理器功能限定到管理状态是有用的。使用处理器 12 和外协处理器 14，16 之间信号交换的一个集合来协调协处理器接口 30 指令的执行。

由处理器 12 生成的控制信号是处理器 12 内部流水线结构的一个反映。处理器流水线 44 包括取指令，指令译码 42，执行，和结果写回阶段。它包括一个或多个指令寄存器（IR）。处理器 12 也包括一个指令预取缓冲器以允许在译码阶段 42 之前缓冲一条指令。通过进入指令译码寄存器 IR，指令从这个缓冲器继续到指令译码阶段 42。

指令译码器 42 从 IR 接收输入，并且在 IR 中所保存的值的基础上生成输出。这些译码 42 输出不总是有效的，有可能要依据指令流中的异常条件或变化删除某些输出。即使在有效时，也可能在 IR 中保存指令，直到它们能够继续进行到指令流水线中的执行阶段。由于这个只有在已经完成了前面的指令时才能发生（这可采取多时钟），译码器将继续译码 IR 中所包括的值，直到 IR 被更新。

图 5 是一个图解用于指令信号交换的基本指令接口操作的时序图。提供一个指令译码选通（H-DEC*）信号 63，以由处理器 12 来指示协处理器接口 30 操作码的译码。当一个协处理器接口 30 操作码驻留在 IR 中时，将断言这个信号，即使可能并不执行该指令而将其删除。对于多时钟，同一指令，可能保持对 H-DEC*63 信号的断言，直到实际发布或删除该指令。

由处理器 12 来监测一个忙信号（H-BUSY*）64，以确定一个外协处理器 14 是否能够接收协处理器接口 30 指令，并且部分控制指令的发布何时发生。如果 H-BUSY*64 信号被否定，而断言了 H-DEC*63，接口将不能停顿指令的执行，并且一旦继续执行指令就能够断言 H-EXEC*65。在处理器 12 译码一个协处理器接口 30 操作码的时候（由 H-DEC*63 的断言来指示），如果断言了信号 H-BUSY*64，那么将强

迫执行延迟的协处理器接口 30 操作码。一旦 H-BUSY*64 信号被否定，处理器 12 能够通过断言 H-EXEC*65 来发布指令。如果协处理器 14 能够缓冲指令，H-BUSY*64 信号就能够被用来填充缓冲器。

图 6 是一个图解当使用 H-BUSY*64 控制协处理器接口 30 指令执行的时候指令接口操作的时序图。一旦已经分解了任何内部停顿条件，并已经否定了 H-BUSY*64 信号，处理器能够断言 H-EXEC*65 以指示协处理器接口 30 指令已经进入了流水线的执行阶段。一个外协处理器 14 应该监视 H-EXEC*65 信号以控制指令的实际执行，因为对于处理器有可能在某种条件下执行之前就删除了该指令。如果一个较早的指令执行导致了一个异常，那么将不会断言 H-EXEC*65 信号，并且将否定 H-DEC*63 输出。如果在 IR 中将该指令删除，作为程序流中一个改变的结果，那么有可能进行一个相似的处理。

图 7 是一个图解指令删除的时序图。如果删除了一条指令，那么在将另一个协处理器接口 30 操作码放在 H-OP[11: 0]61 总线上之前，否定 H-DEC*63 信号。

图 8 是一个图解指令流水线停顿的一个例子的时序图。会存在这样的情况，即便断言了 H-DEC*63 并否定了 H-BUSY*64，处理器 12 也可能延迟 H-EXEC*65 的断言。在等待完成一条较早的指令的时候，可能会发生这种情况。

图 9 是一个图解没有停顿的背对背操作的一个例子的时序图。对于背对背协处理器接口 30 指令，能够保持对 H-DEC*63 信号的断言，而不是将其否定，即使在新指令进入 IR 时更新了 H-OP[11: 0]61 总线。总之，H-EXEC*65 的断言对应于在前一个时钟被译码的指令的执行。

图 10 是一个图解具有内部流水线停顿的背对背操作的时序图。在这种情况下，否定了 H-BUSY*64，但是，直到内部停顿条件消失，处理器才为第二个协处理器接口 30 指令断言 H-EXEC*65。

图 11 是一个图解具有 H-BUSY*64 停顿的背对背协处理器接口 30 指令的时序图。在这个例子中，外协处理器 14 为忙，不能立即接

收第二条指令。断言 H-BUSY*64 以阻止处理器 12 发布第二条指令。一旦协处理器 14 空闲，则否定 H-BUSY*64，并且将下一个第二个协处理器接口 30 指令提前到执行阶段。

与译码协处理器接口 30 操作码相关的异常可能是由外协处理器 14 使用 H-EXEC*66 信号进行通知的。在断言 H-DEC*63 并否定 H-BUSY*64 的时钟周期中，采样处理器 12 的这个输入，并且如果没有象以前描述的那样删除协处理器接口 30 操作码，将会引起一个硬件协处理器 14 异常的异常处理。下面将会描述这个异常处理的细节。

图 12 是一个图解 H-EXCP*66 信号的一个例子的时序图，该信号是由协处理器 14 断言的以响应协处理器接口 30 操作码的译码和测试操作。在断言 H-DEC*63 并否定 H-BUSY*64 的时钟中，处理器 12 采样 H-EXCP*66 信号。断言 H-EXEC*65 信号，无论是否由该接口以信号形式通知一个异常；这个断言将异常时所采取的情况与指令删除情况区别开。

应注意到该异常对应于在前一个时钟周期被译码的指令，并且不应该采取实际的执行。一个协处理器 14 必须在它所能识别的处理器流水线的执行阶段之前接收一条不好的指令并且发出一个异常信号。对于所有否定 H-DEC*63 或断言 H-BUSY*64 的时钟周期，都忽略 H-EXCP*66 信号。

图 13 是一个图解 H-EXCP*66 信号的一个例子的时序图，该信号是由协处理器 14 所断言的以响应协处理器接口 30 操作码的解码和测试操作。对比这个与图 14 中的时序图，在这个例子中，删除了协处理器接口 30 指令，这样就没有断言 H-EXEC*65 信号，并且否定了 H-DEC*63。

图 14 是一个图解一个例子的时序图，在该例子中已经断言了 H-BUSY*64 以延迟一个将会引起一个异常的协处理器接口 30 操作码的执行

所有的协处理器 14，16 都能共享 H-BUSY*64 和 H-EXCP*66 信号，这样就必须以一种协调的方式来驱动它们。应该由对应于

H-OP[11: 10]61 的协处理器 14，16 在确定 H-DEC*63 的时钟周期来驱动这些信号（或高或低，无论哪一个都适合）。通过只在时钟的低部分驱动输出，多个协处理器 14，16 可以无争用的共享这些信号。在这个输入上提供处理器 12 内部的一个保持锁存，对于时钟的高阶段将它保持在一个有效的状态，同时没有单元在驱动它。

协处理器接口 30 指令原语中的一些也隐含着在处理器 12 和外协处理器 14 之间传送数据项。可以在协处理器接口 30 中传送操作码，作为被执行的特定原语的一个函数。在一个 32 位双向数据通路之中，提供向或是从协处理器 14 传送处理器 12 个 GPR 中的一个或多个。此外，也提供从/向具有数据汇/源，即协处理器接口 30 的存储器 18 加载或存储一个信号数据项。处理器 12 将通过 HDP[31: 0]72 总线在 CLK60 的高部分将参数传送到外协处理器 14，并且由处理器 12 在时钟的低阶段从协处理器接收操作码，并将其锁定。在驱动发生之前提供一个延迟做为时钟转移，以允许总线跨区转接的一个小的阶段。一个协处理器 14 接口必须在时钟的下降沿提供相同的小的延迟。由数据选通（H-DS*70）输出，数据确认（H-DA*71）输入和数据出错（H-ERR*69）输出信号来支持数据项的信号交换。

处理器 12 提供向协处理器接口 30 传送一个调用或返回参数表的能力，这与调用或返回软件例程的方式相同。在 H-CALL 或 H-RET 原语中指示变元的数量，以控制所通过的参数的数量。以处理器 12 寄存器 R4 的内容开始的寄存器值被传送到（出）外协处理器 14，作为 H-CALL（H-RET）原语的操作的一部分，总共可能通过七个寄存器参数。这个约定与软件例程调用的约定相类似。

操作数传输的信号交换由数据选通（H-DS*70）输出和数据确认（H-DA*71）输入信号控制。在传输期间处理器 12 断言数据选通，并且以一种覆盖的方式进行传输，这与处理器 12 接口操作相同。数据确认（H-DA*）71 用来指示一个协处理器 14 已经接受或驱动了一个数据单元。

图 15 是一个图解与 H-CALL 原语相关的寄存器 46 传输的一个例

子的时序图。提供指令原语以传输多处理器寄存器，并且在理想的情况下，在每个时钟都能进行传输。对于向一个外协处理器 14 的传输，处理器自动在确认当前项之前（或同时）开始驱动下一个操作数（如果需要的话）。外部逻辑必须具有一级缓冲的能力以确保数据不会丢失。这张图表示了向协处理器接口 30 的 H-CALL 传输的顺序，其中传送了两个寄存器。根据一个否定的数据确认（H-DA*）71，重复第二次传输。

对于从一个外协处理器 14 向处理器寄存器 46 的传输，处理器 12 能够在已经断言了 H-DS*70 之后，在每个时钟周期中都从一个外协处理器 14 中接受值，并且按接收时的情况将这些值写入寄存器文件 46，因此不需要缓冲。

图 16 是一个图解与 H-RET 原语相关的寄存器 46 传输的一个例子的时序图。在这个例子中，传输两个寄存器 46 值。协处理器 14 可以在断言 H-EXEC*65 信号之后的时钟的开始处驱动数据，因为这是第一次断言 H-DS*70 时的时钟。H-DS*70 输出随 CLK60 的上升沿转换，而在时钟 CLK60 的低阶段采样 H-DA*71 输入。

处理器 12 提供了使用 H-LD 或 H-ST 指令原语向或从协处理器接口 30 传输一个单存储器操作数的功能。

H-LD 原语被用来从存储器 18 向一个协处理器 14 传输数据。向协处理器 14 的操作数传输的信号交换是由数据选通信号（H-DS*）70 信号控制的。由处理器 12 来断言数据选通，以指示已经将一个有效操作数放在了 HDP[31: 0]72 总线上。对于这一传输，忽略了数据确认（H-DA*）71 输入。

图 17 是一个图解向协处理器接口 30 的 H-LD 传输的顺序的时序图。在这种情况中，存在一个立即状态的存储器 18 存取。对于 n 个等待状态的存储器 18 存取，将在 n 个时钟之后驱动操作数和 H-DS*70。如果选择用加载的有效地址更新基寄存器 46 的选项，则在计算之后的第一个时钟中（H-EXEC*65 确认之后的时钟），在 HDP[31: 0]72 上驱动更新值。

图 18 是一个图解当一个存储器 18 存取导致一个存取异常时协议的时序图。在这种情况中，断言 H-ERR*69 信号返回给外协处理器 14。

H-ST 原语能够被用来从一个协处理器 14 向存储器 18 传输数据。如果选择了用存储的有效地址更新基寄存器 46 的选项，则在被计算之后的第一个时钟（紧接在 H-EXEC*65 之后的时钟），在 HDP[31:0]72 上驱动更新值。

图 19 是一个图解与 H-ST 原语相关的传输的一个例子的时序图。与 H-ST 原语相关的信号交换由两部分组成，一个是来自协处理器 14 的初始信号交换，它必须提供要存储的数据，另一个是一旦完成存储器 18 的存储，来自处理器 12 的一个完成信号交换。

初始信号交换使用向处理器 12 的 H-DA*71 输入来表明协处理器 14 已经向处理器 12 驱动了存储数据。在与协处理器 14 将数据驱动到 HDP[31: 0]72 总线上的相同的时钟中，断言 H-DA*71 信号。在总线的下半部分取出存储数据，用于进行半个字长的存储，而不将高区的 16 位写入到存储器 18 中。在断言 H-EXEC*65 信号的时钟的开始处采样 H-DA*71 信号。在识别出 H-DA*71 的时钟中，请求存储周期，并且在下一个时钟将存储数据驱动到存储器 18。一旦完成了该存储，处理器 12 将断言 H-DS*70 信号。

图 20 是一个图解具有延迟存储数据的传输的一个例子的时序图。

图 21 是一个图解当存储导致一个存取错误时的协议信号的时序图。应注意断言了 H-ERR*69 信号。如果硬件单元是通过在断言 H-EXCP*65 的时钟中断言 H-EXCP*66，来异常终止指令，则不应该断言 H-DA*71 信号。

图 22 到 26 是图解做为指令集的一部分所提供的指令，以接口一个硬件加速器（或协处理器）14。处理器 12 解释原语中字段的一部分，其它字段则由协处理器 14 单独解释。

图 22 图解用于 H-CALL 原语的一个指令格式。这一指令用来调用由协处理器 14 执行的一个函数。该范例与一个标准的软件调用约

定相类似，但是是以一种硬件的前后关系出现的。H-CALL 原语是由处理器 12 和协处理器 14 共同解释的，以传输一个来自处理器 12 的“调用参数”或变元表，并且在协处理器 14 中初始化一个特定函数。

指令字的 UU 和 CODE 字段不是由处理器 12 解释的，它们被用来指定一个协处理器 14 特定功能。UU 字段可以指定一个特定的协处理器 14，16，CODE 字段可以指定一个特殊操作。CNT 字段是由处理器 12 和协处理器 14 一起解释，并指定传送到协处理器 14 的寄存器变元的数目。

从通用寄存器 46 传送变元，从 R4 开始并且继续通过 R（4+CNT-1）。在一个单独的 H-CALL 调用中可以通过多达七个参数或寄存器 46。

H-CALL 指令能够被用来实现模块功能块的调用。人们早就知道这种类型接口的使用会使软件系统具有更高的可靠性和较少的错误。通常最好由值来传递函数参数。这样会显著地降低副作用。在许多情况下，分程序结构语言的现代编译器，例如 C 和 C++，在寄存器 46 中向调用函数或子例程传送参数或变元的短序列。这种技术可以由 H-CALL 指令来实现。可以将一个编译器配置成向以 R4 开始的累次寄存器 46 加载多达七个参数或变元，接着生成 H-CALL 指令，该指令替代了生成子例程连接指令的标准编译器。

图 23 图解用于 H-RET 原语的一个指令格式。这一指令被用来从一个由协处理器 14 所实现的函数“返回”。该范例与处理器 12 使用的软件调用约定相类似，但是以硬件的前后关系出现的。该 H-RET 原语由处理器 12 和协处理器 14 共同解释，以从一个协处理器 14 向处理器 12 传输一个“返回参数”或值的列表。

指令字的 UU 和 CODE 字段不是由处理器 12 解释的，它们被用来指定一个协处理器 14 特定功能。UU 字段可以指定一个硬件单元，CODE 字段可以指定协处理器 14 中寄存器 46 的一个特殊操作或集合以返回。CNT 字段是由处理器 12 和协处理器 14 一起解释的，并且指定从协处理器 14 向处理器 12 传送的寄存器 46 变元的数目。

变元被传递到处理器 12 通用寄存器 46，以 R4 开始，并继续通过 R（4+CNT-1）。可以返回多达七个参数（或寄存器内容）。

至于 H-CALL 指令，H-RET 指令也可以被用来实现模块化编程。结构化编程要求最好通过值将函数返回值返回到一个调用例程。对于一个子例程或函数返 回，这经常可以由编译器在寄存器中放置一个或多个返回值来高效地完成。应该注意到传统的结构化编程希望在调用子例程或函数之后能立即返回。在使用协处理器 14 的情况下，执行通常与调用引用处理器 12 的执行相同步。H-RET 指令可以被用来使处理器 12 和协处理器 14 重新同步。这样，处理器 12 可以加载一个或多个寄存器 46，用一个或多个 H-CALL 指令激活协处理器 14，执行不相关指令，并且接下来通过发布 H-RET 指令，在从一个协处理器 14 接收一个结果值或多个值时，与协处理器 14 重新同步。

图 24 图解用于 H-EXEC 原语的一个指令格式。这一指令被用来初始化一个函数或进入一个由加速器实现的运行方式。H-EXEC 指令可以被用来控制在一个特定协处理器 14，16 中由一个 UU 字段指定的一个函数。代码字段不是由处理器 12 解释的，但是为指定的协处理器 14，16 所保留。指令字的 UU 和 CODE 字段不是由处理器 12 解释的，它们被用来指定一个协处理器 14 的特定函数。UU 字段可以指定一个特定协处理器 14，16，CODE 字段可以指定一个特殊操作。

图 25 图解用于 H-LD 指令的一个指令格式。使用这一指令来从存储器 18 向一个协处理器 14 传送一个值，传递过程中没有在一个通用寄存器（GPR）46 中临时存储存储器操作数。使用一个基指针和一个偏移来寻址该存储器操作数。

H-LD 指令执行在存储器 18 中加载一个值的操作，并且向协处理器 14 传送存储器操作数，而不是将其存储在一个寄存器 46 中。H-LD 操作有三个选项，w—字，H—半字以及 u—更新。通过加载的大小以及零延伸来换算 IMM2 字段，以来获取 Disp。将这个值加到寄存器 RX 的值上，并且从这个地址执行一个指定大小的加载，将加载的结果传送到硬件接口 28。对于半字加载，所取的数据是从零延伸到 32

位。如果指定 u 选项，则在计算之后，将加载的有效地址放入寄存器 RX46 中。

指令字的 UU 字段不是由处理器 12 解释的，这个字段可以指定一个特定协处理器 14，16。Sz 字段指定操作数的大小（只限于半字或字）。Disp 字段指定一个要加到由 Rbase 字段所指定的寄存器中内容的无符号偏移值，以形成加载的有效地址。Disp 字段的值是通过要传输的操作数的大小来换算的。Up 字段指定在计算之后是否要用加载的有效地址来更新 Rbase 寄存器 46。这个选项允许一个“自动更新”寻址模式。

图 26 图解用于 H-ST 指令的一个指令格式。这个指令被用来从一个协处理器 14 向存储器 18 传送一个值，而不在处理器 12 寄存器 46 中临时存储该存储器操作数。使用一个基指针和一个偏移来寻址该存储器操作数。

指令字的 UU 字段不是由处理器 12 解释的。但这个字段可以指定一个特定的协处理器 14，16。Sz 字段指定操作数的大小（只限于半字或字）。Disp 字段指定一个要加到由 Rbase 字段所指定的寄存器 46 中内容的无符号偏移值，以形成要存储的有效地址。通过要传输的操作数的大小来换算 Disp 字段的值。Up 字段指定在计算之后，是否要用有效地址来更新 Rbase 寄存器 46。这个选项允许一个“自动更新”寻址模式。

H-ST 指令执行从一个协处理器 14 向存储器 18 中存储一个操作数，而不在寄存器 46 中存储该操作数。H-ST 操作有三个选项，w—字，H—半字以及 u—更新。通过存储的大小以及零延伸来换算 IMM2 字段以获取 Disp。这个值被加到寄存器 RX 的值上，并且用从硬件接口获得的用于存储的数据对这个地址进行一个指定大小的存储。如果指定 u 选项，则在计算之后将所加载的有效地址，放入寄存器 RX 中。

H-LD 指令和 H-ST 指令提供了一种从存储器 18 向一个协处理器 14 以及从一个协处理器 14 向存储器 18 移动操作数，而不是通过寄存器 46 进行数据移动的有效机制。偏移和索引设备提供了一种通过阵

列进行步进的有效机制。这样，这些指令在循环中非常有用。应该注意对于操作数的每次加载或存储，两条指令都使处理器 12 与协处理器 14 相同步。如果这并不是必须的或者是优选的，则可以选择通过重复向一个指定寄存器或多个寄存器 46 加载来自存储器 18 的数据，来将数据注入到协处理器 14 中，并且由于协处理器接口总线 30 也被用于寄存器窥探，所以也可以让协处理器 14 检测这些加载。

熟悉本技术的人员会发现可以在不脱离本发明精神的情况下进行修改和变化。因此，本发明包含了所附权利要求书范围内的所有变化和修改。

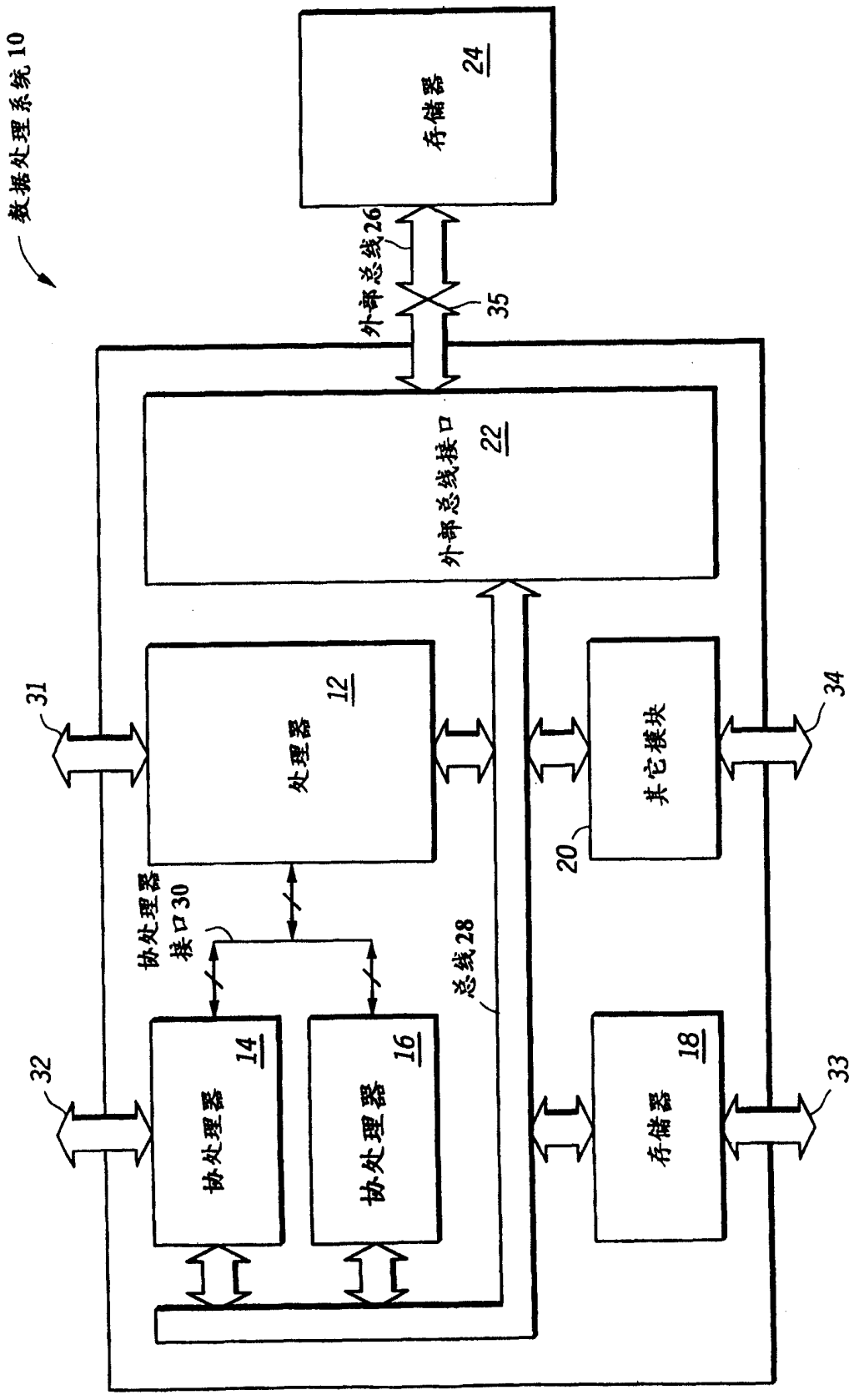
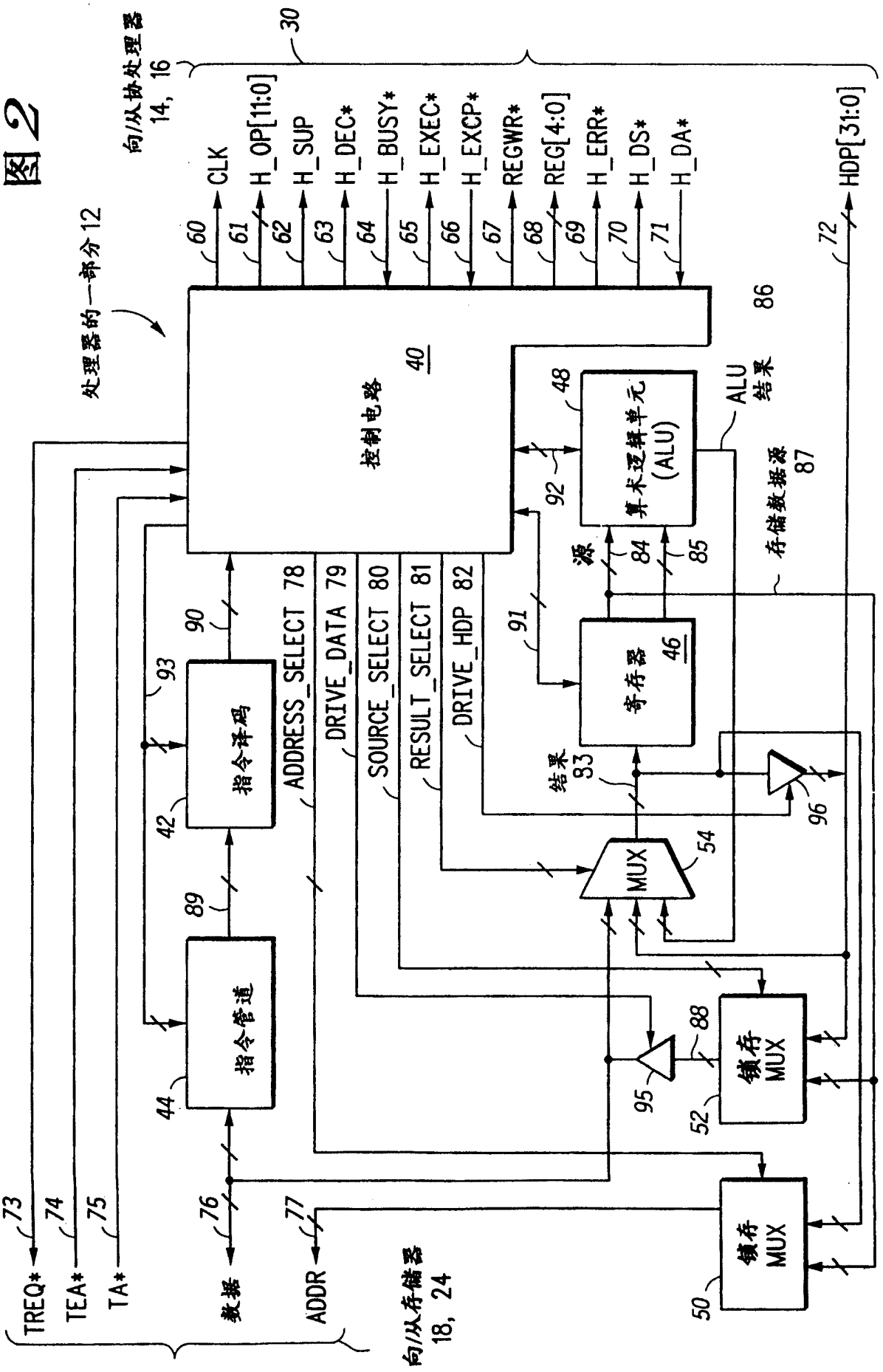

数据处理系统 10
存储器 24
外部总线26
35
外部总线接口 22
处理器 12
31
其它模块
34
20
协处理器接口 30
总线28
协处理器 14
协处理器 16
32
存储器 18
33

图 1

图 2


向/从协处理器
14, 16
30
处理器的一部分 12
CLK
H_OP[11:0]
H_SUP
H_DEC*
H_BUSY*
H_EXEC*
H_EXCP*
REGWR*
REG[4:0]
H_ERR*
H_DS*
H_DA*
HDP[31:0]
60
61
62
63
64
65
66
67
68
69
70
71
72
控制电路
40
86
ALU 结果
87
存储数据源
算术逻辑单元 (ALU)
48
92
源
84
85
寄存器
46
91
结果
83
96
MUX
54
指令译码
42
90
93
指令管道
44
89
ADDRESS_SELECT 78
DRIVE_DATA 79
SOURCE_SELECT 80
RESULT_SELECT 81
DRIVE_HDP 82
锁存 MUX
52
88
95
锁存 MUX
50
TREQ*
TEA*
TA*
数据
ADDR
73
74
75
76
77
向/从存储器
18, 24

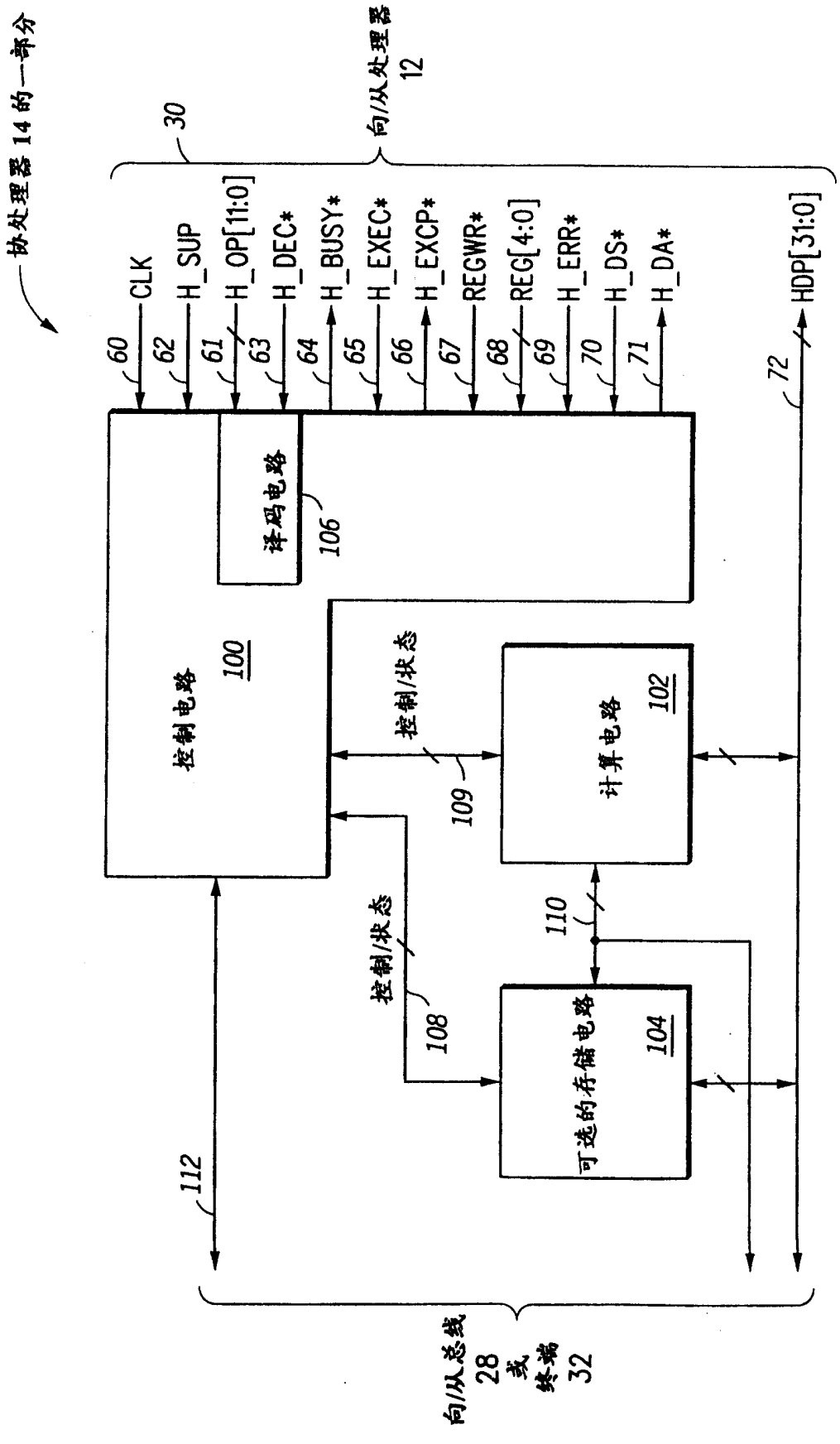

协处理器14的一部分
向/从处理器
12
30
CLK
H_SUP
H_OP[11:0]
H_DEC*
H_BUSY*
H_EXEC*
H_EXCP*
REGWR*
REG[4:0]
H_ERR*
H_DS*
H_DA*
HDP[31:0]
60
62
61
63
64
65
66
67
68
69
70
71
72
译码电路
106
控制电路
100
控制/状态
109
计算电路
102
控制/状态
108
110
可选的存储电路
104
112
向/从总线
28
或
终端
32

图3

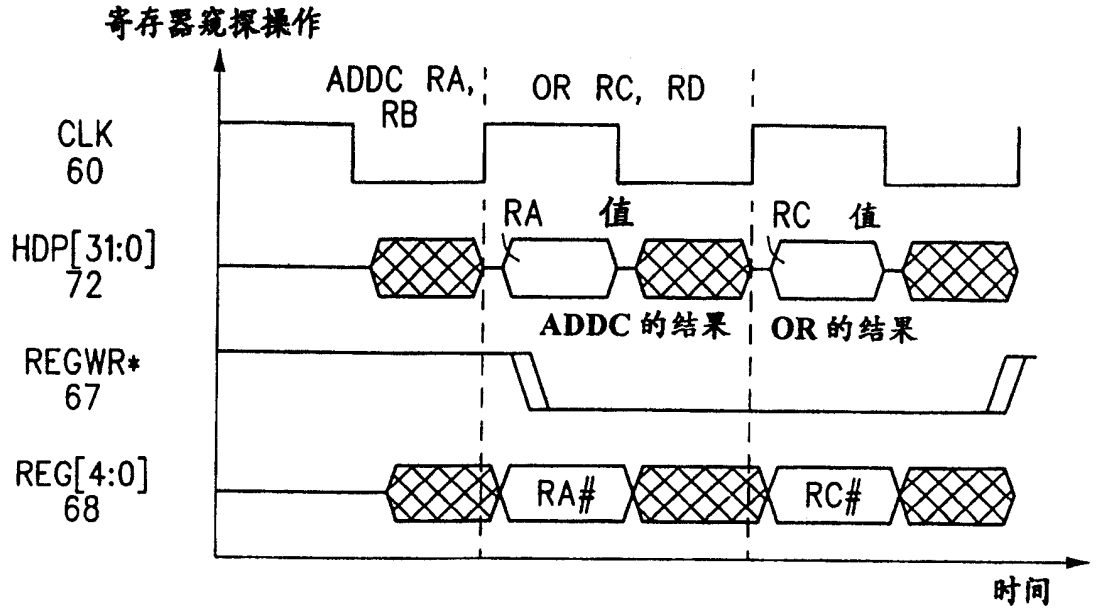

寄存器窥探操作
ADDC RA, RB
OR RC, RD
CLK
60
RA 值
RC 值
HDP[31:0]
72
ADDC 的结果
OR 的结果
REGWR*
67
REG[4:0]
68
RA#
RC#
时间

图4

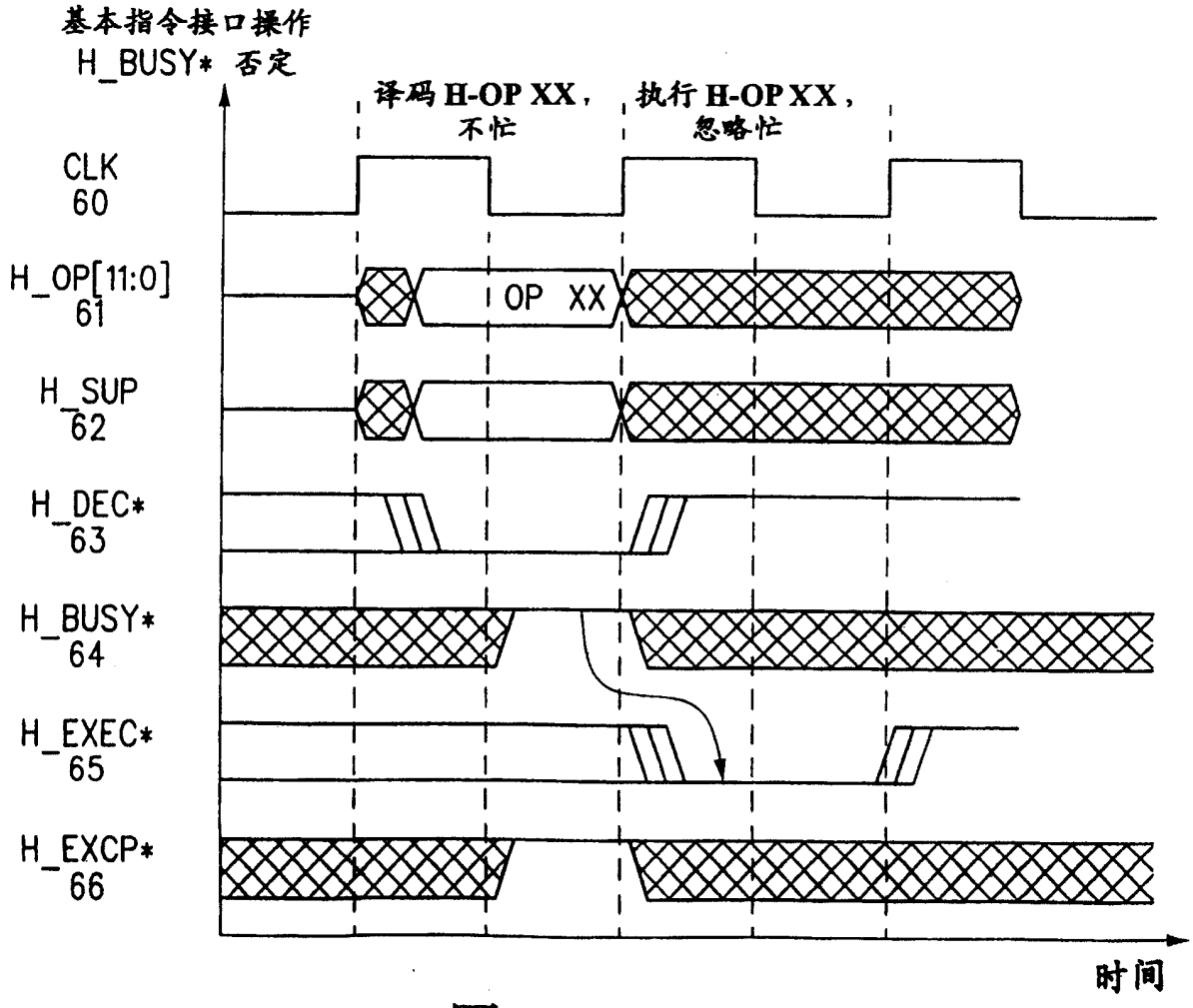

基本指令接口操作
H_BUSY* 否定
译码 H-OP XX，
不忙
执行 H-OP XX，
忽略忙
CLK
60
H_OP[11:0]
61
OP XX
H_SUP
62
H_DEC*
63
H_BUSY*
64
H_EXEC*
65
H_EXCP*
66
时间

图5

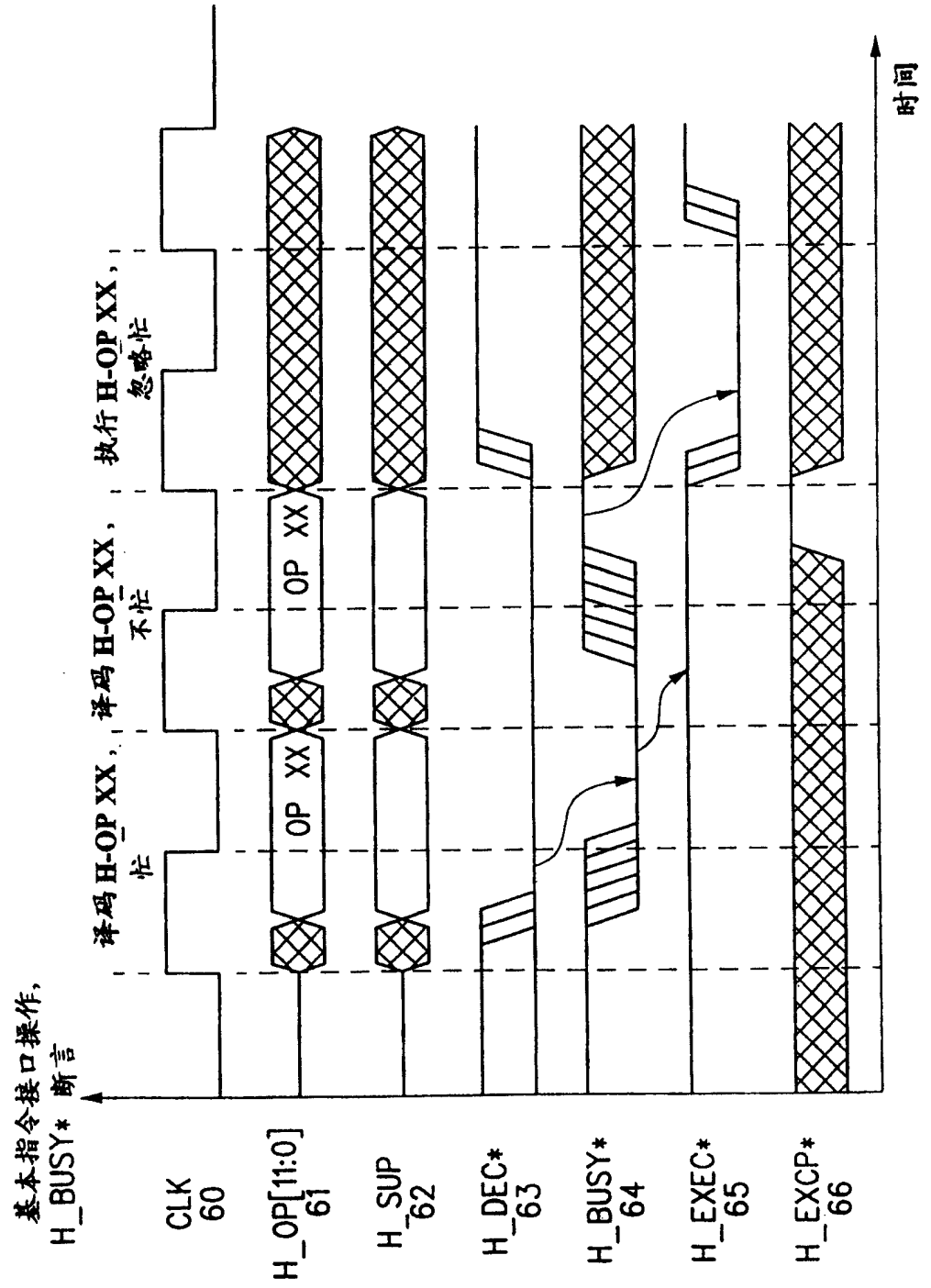

基本指令接口操作,
H_BUSY* 断言
译码 H-OP XX,
忙
译码 H-OP XX,
不忙
执行 H-OP XX,
忽略忙
CLK
60
H_OP[11:0]
61
OP XX
OP XX
H_SUP
62
H_DEC*
63
H_BUSY*
64
H_EXEC*
65
H_EXCP*
66
时间

图6

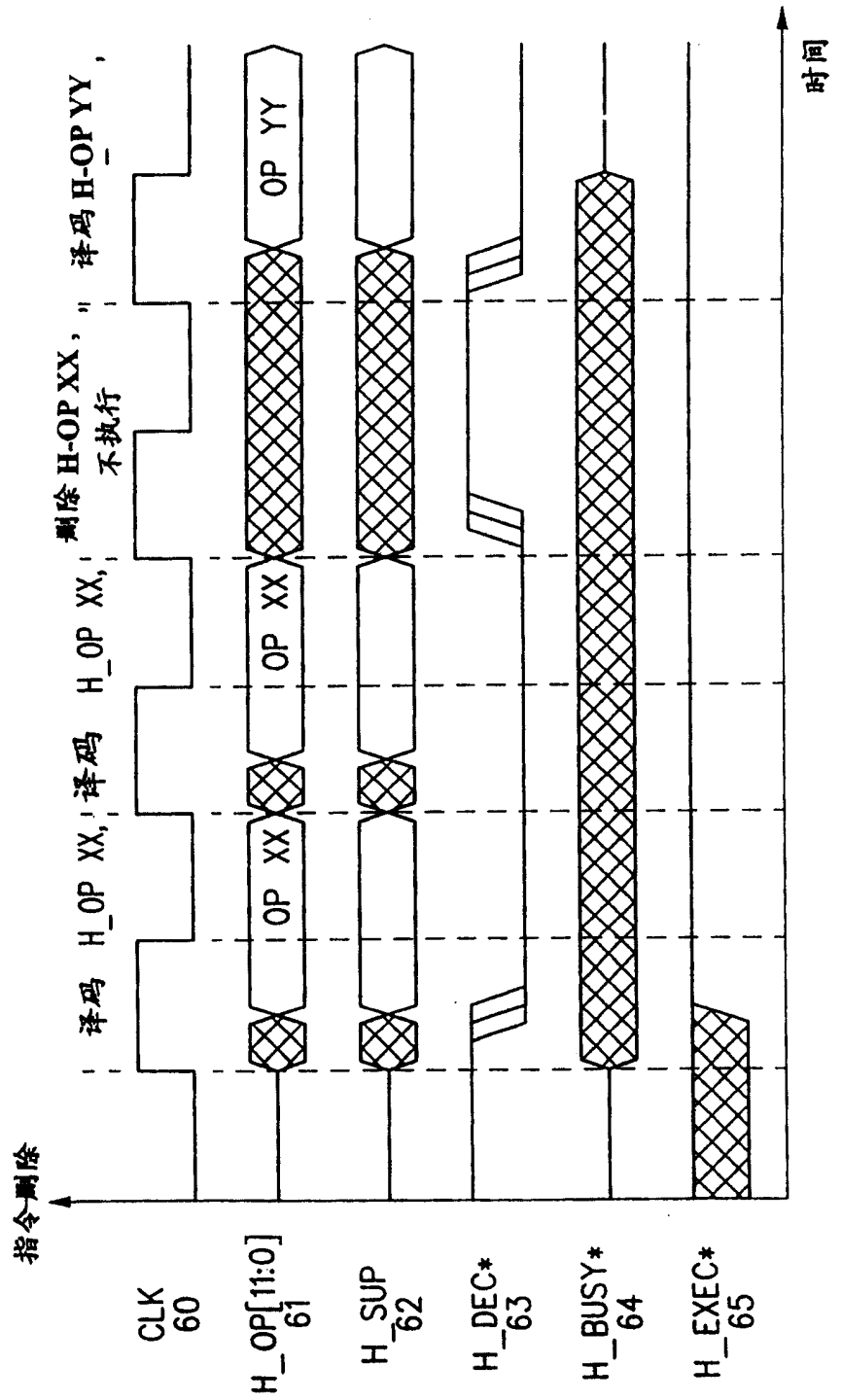

指令删除
译码 H_OP XX,
译码 H_OP XX,
删除 H-OP XX，不执行
译码 H-OP YY，
时间
CLK 60
H_OP[11:0] 61
H_SUP 62
H_DEC* 63
H_BUSY* 64
H_EXEC* 65
OP XX
OP XX
OP YY

图 7

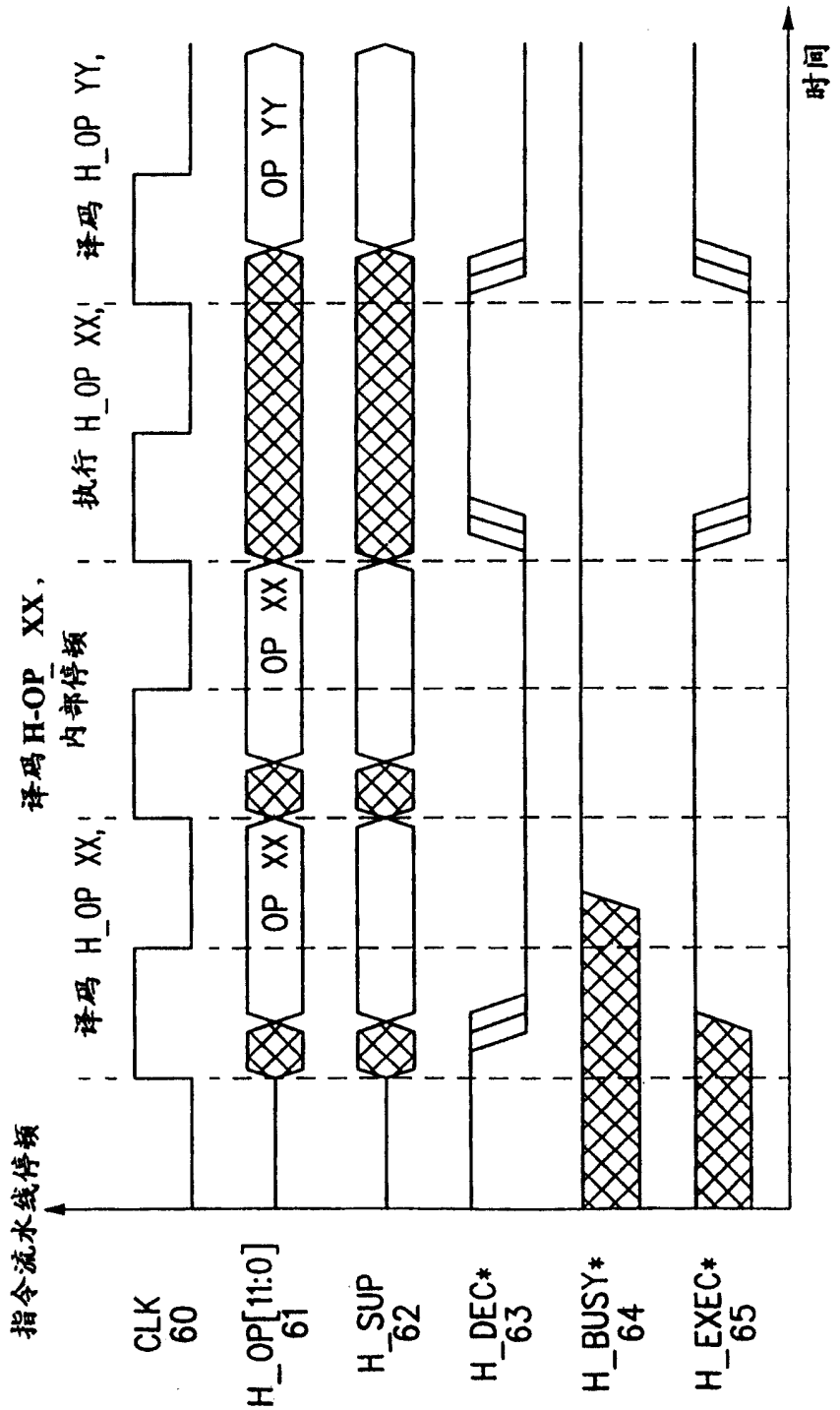

指令流水线停顿
译码 H_OP XX,
译码H-OP XX,
内部停顿
执行 H_OP XX,
译码 H_OP YY,
OP XX
OP YY
时间
CLK 60
H_OP[11:0] 61
H_SUP 62
H_DEC* 63
H_BUSY* 64
H_EXEC* 65

图 8

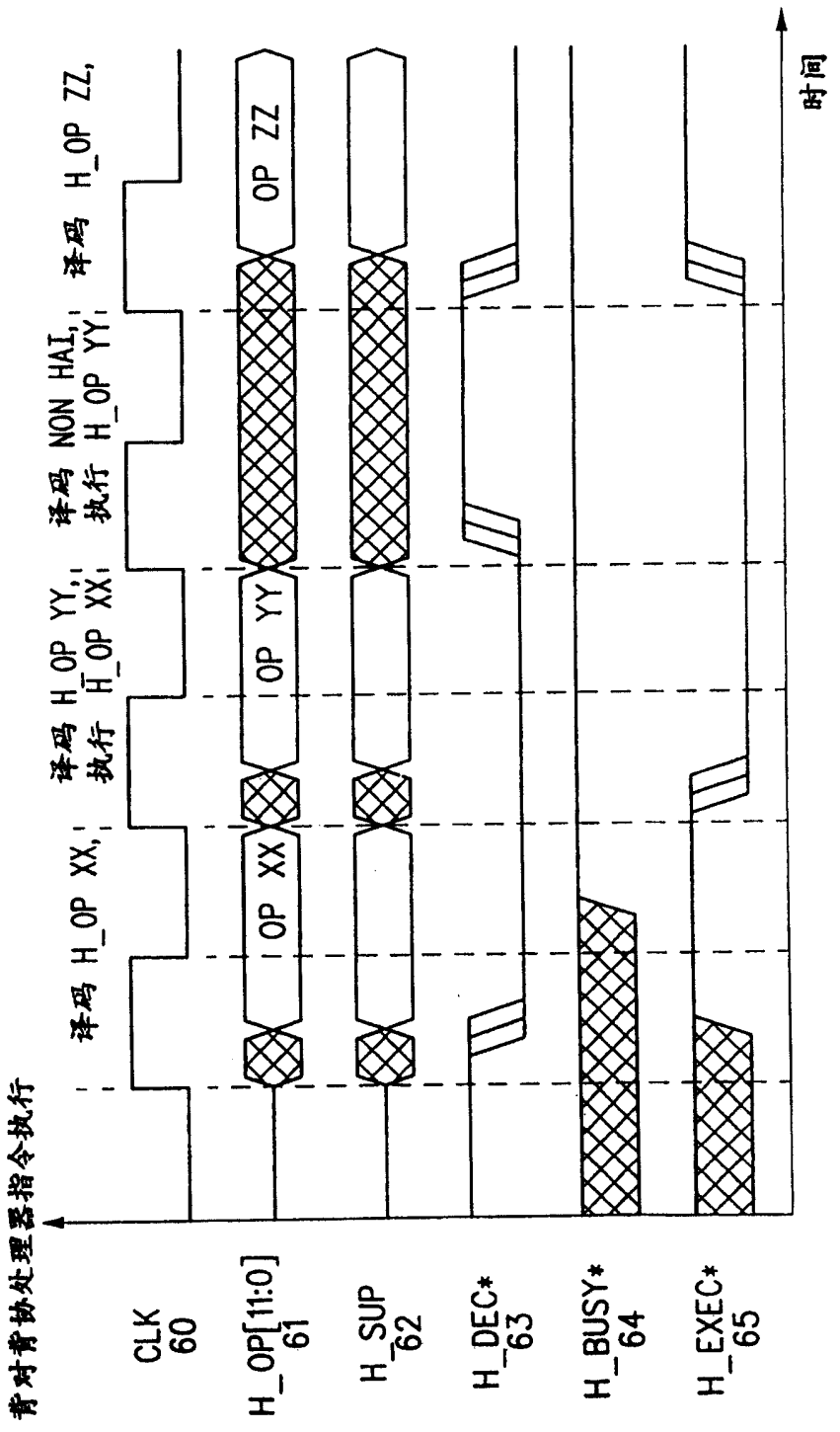

背对背协处理器指令执行
译码 H_OP XX,
执行
译码 H_OP YY,
执行 H_OP XX,
译码 NON HAI,
执行 H_OP YY,
译码 H_OP ZZ,
OP XX
OP YY
OP ZZ
CLK 60
H_OP[11:0] 61
H_SUP 62
H_DEC* 63
H_BUSY* 64
H_EXEC* 65
时间

图 9

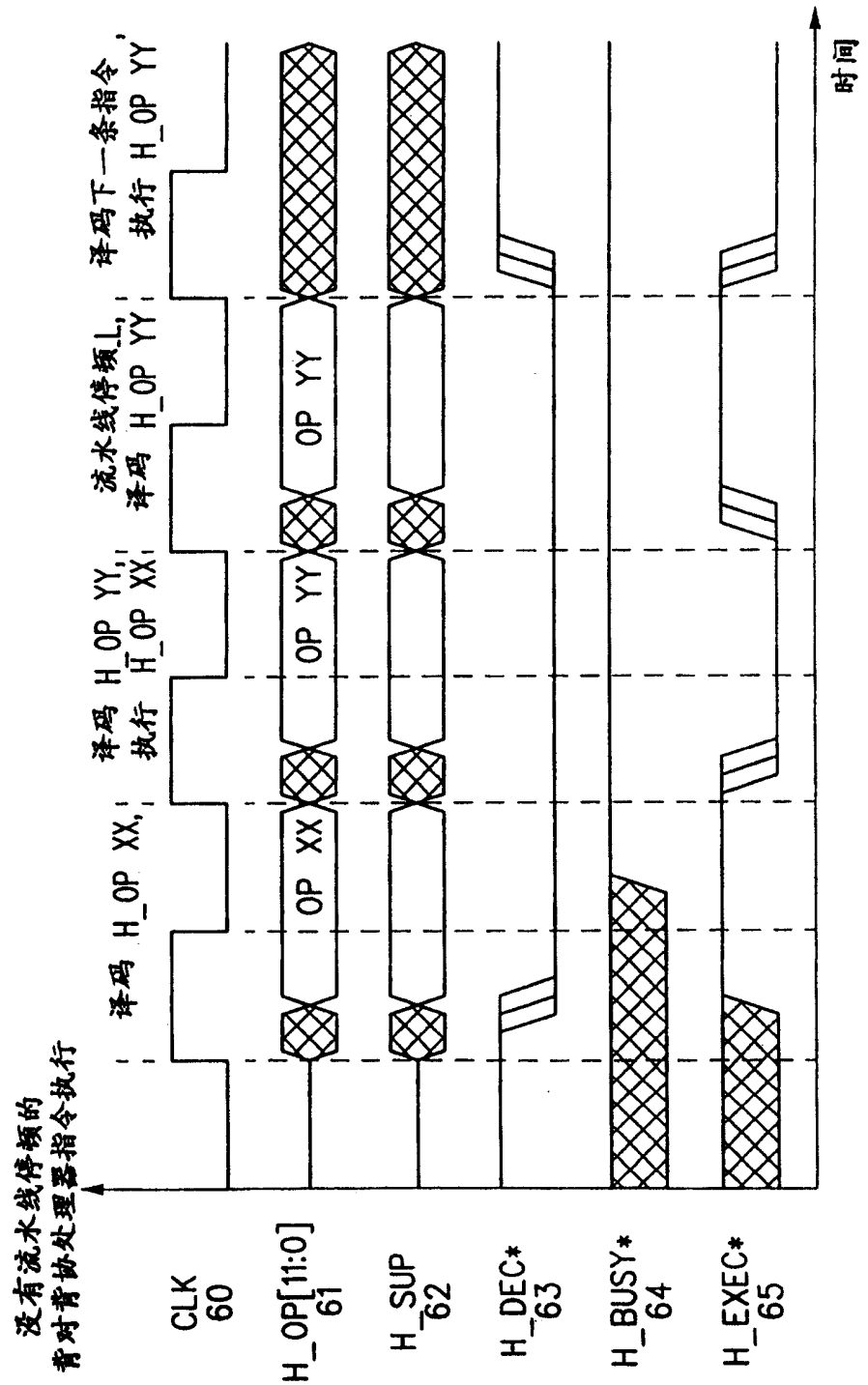

没有流水线停顿的
有对有协处理器指令执行
译码 H_OP XX,
执行 H_OP YY,
译码 H_OP YY,
流水线停顿L,
译码下一条指令,
执行 H_OP YY
CLK
60
H_OP[11:0]
61
H_SUP
62
H_DEC*
63
H_BUSY*
64
H_EXEC*
65
OP XX
OP YY
OP YY
时间

图10

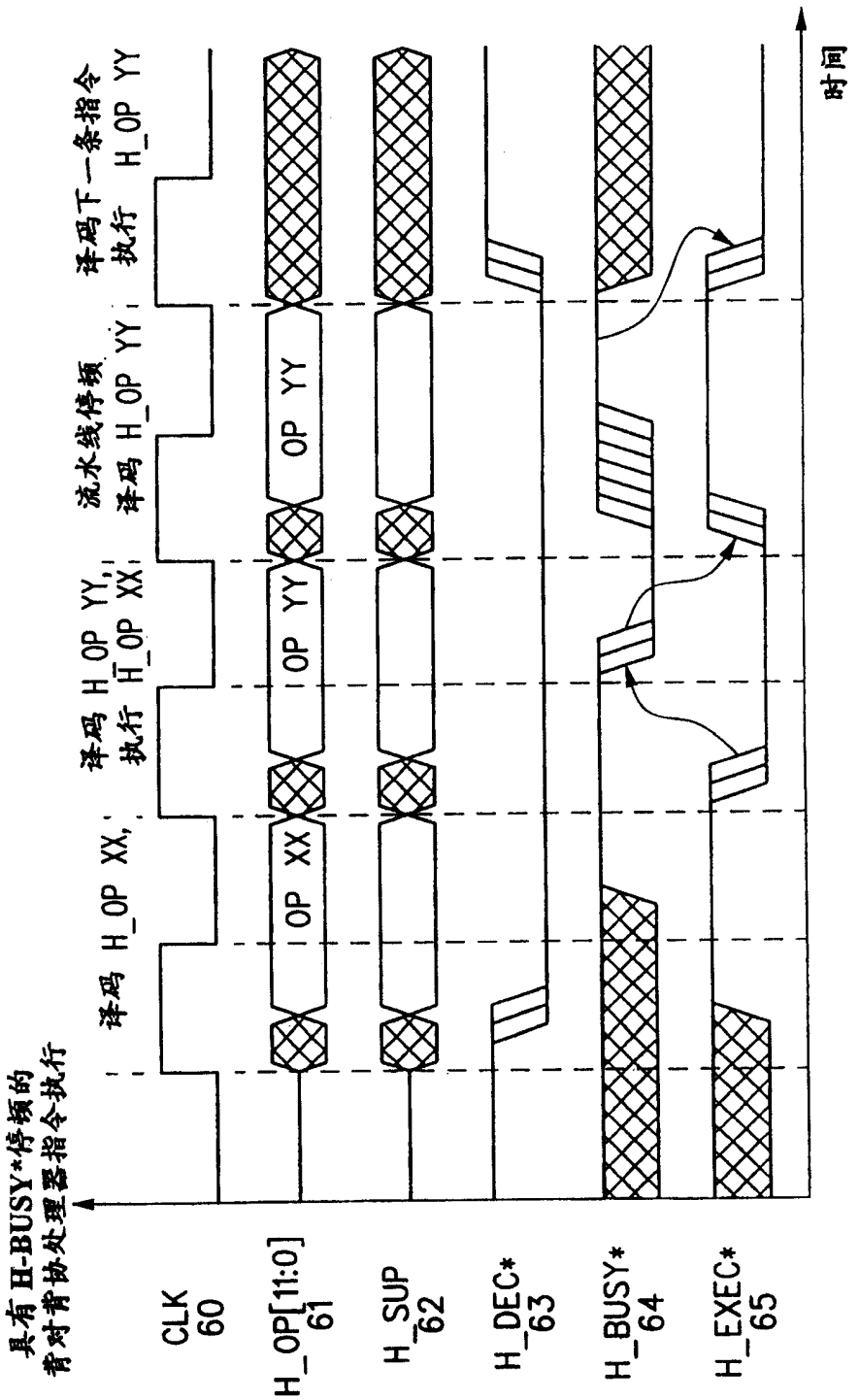

具有H-BUSY*停顿的
背对背协处理器指令执行
译码 H_OP XX,
译码 H_OP YY,
执行 H_OP XX,
流水线停顿
译码 H_OP YY,
译码下一条指令
执行 H_OP YY
CLK
60
H_OP[11:0]
61
H_SUP
62
H_DEC*
63
H_BUSY*
64
H_EXEC*
65
OP XX
OP YY
OP YY
时间

图 11

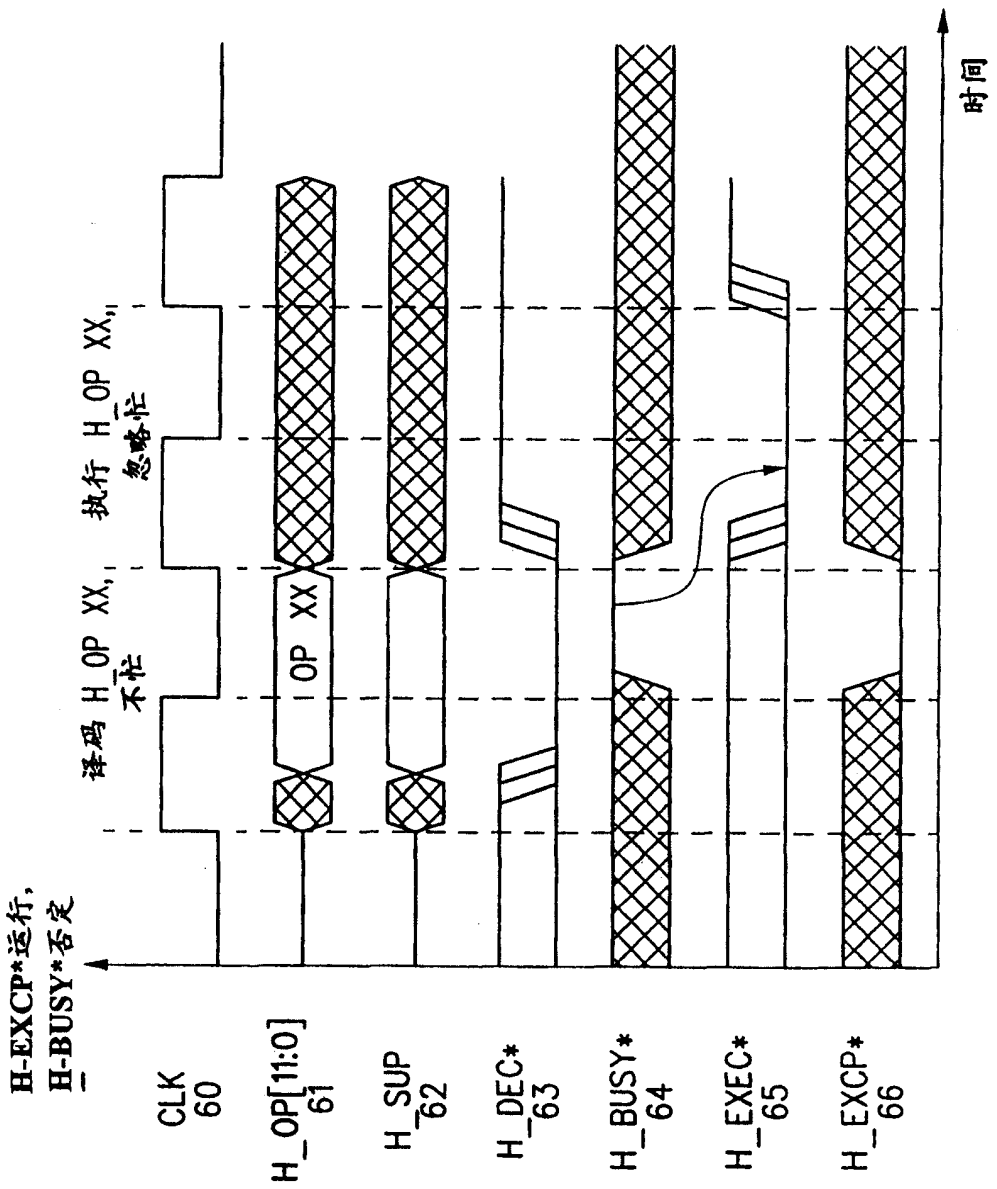

H-EXCP*运行,
H-BUSY*否定
译码 H_OP XX,1 执行 H_OP XX,1
不忙 忽略忙
OP XX
CLK 60
H_OP[11:0] 61
H_SUP 62
H_DEC* 63
H_BUSY* 64
H_EXEC* 65
H_EXCP* 66
时间

图 12

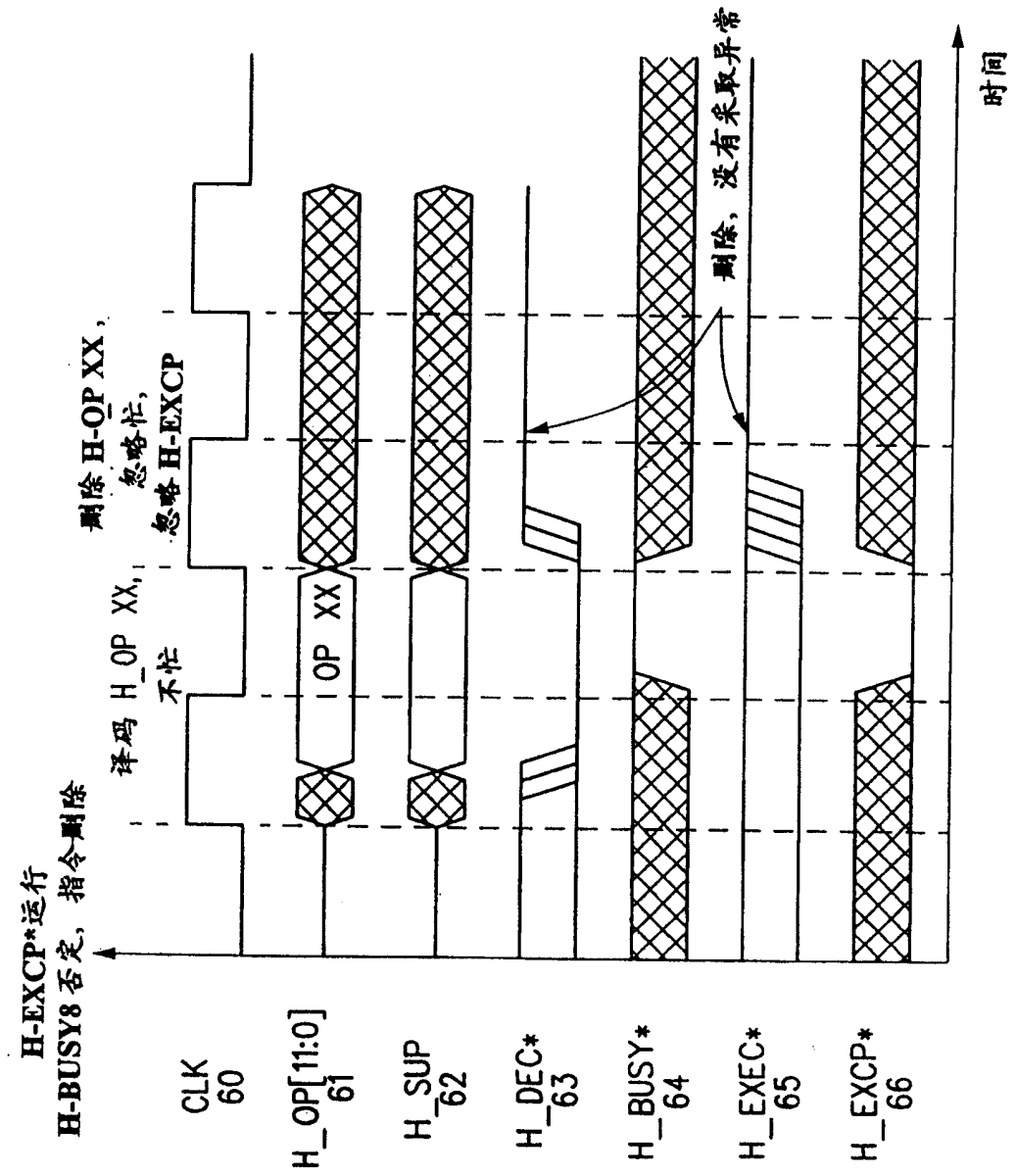

H-EXCP*运行
H-BUSY8否定，指令删除
译码 H_OP XX，
不忙
删除 H-OP XX，
忽略忙，
忽略 H-EXCP
删除，没有采取异常
时间
CLK
60
H_OP[11:0]
61
OP XX
H_SUP
62
H_DEC*
63
H_BUSY*
64
H_EXEC*
65
H_EXCP*
66

图13

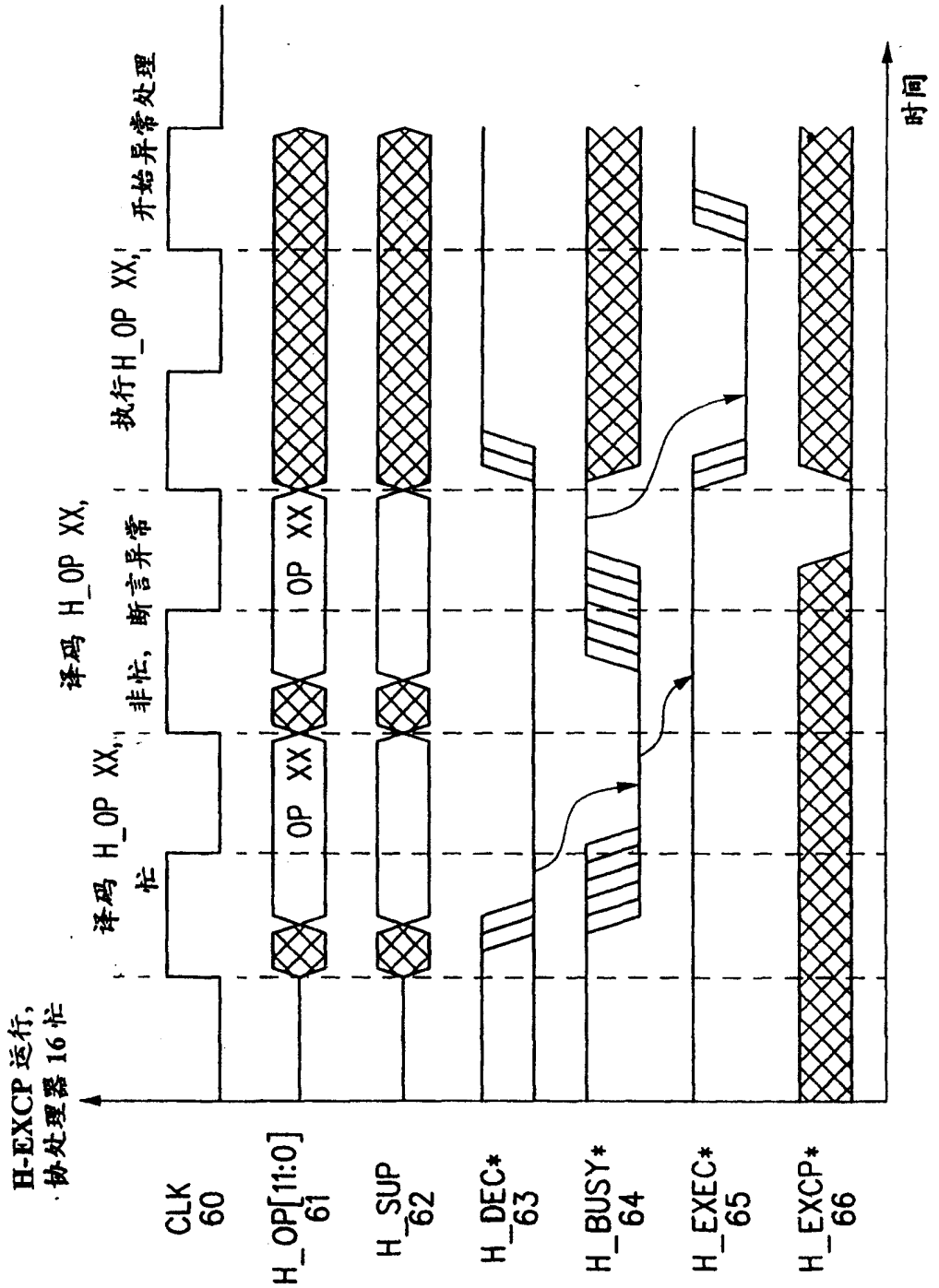

H-EXCP 运行，
协处理器 16 忙
CLK
60
H_OP[11:0]
61
H_SUP
62
H_DEC*
63
H_BUSY*
64
H_EXEC*
65
H_EXCP*
66
译码 H_OP XX,
忙
译码 H_OP XX,
非忙，断言异常
执行H_OP XX,
开始异常处理
OP XX
OP XX
时间

图 14

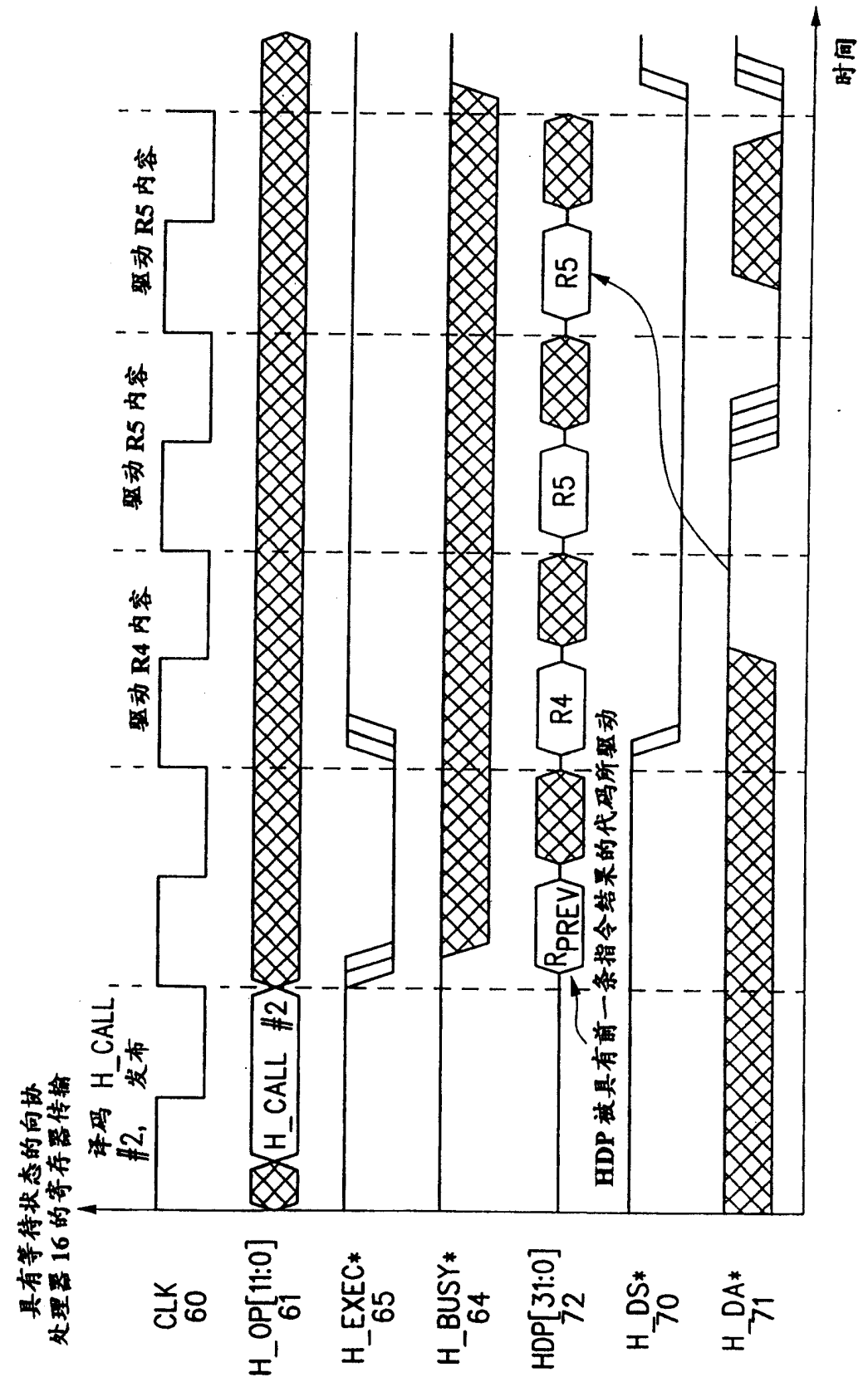

具有等待状态的向协处理器16的寄存器传输
译码 H_CALL #2, 发布
驱动 R4 内容
驱动 R5 内容
驱动 R5 内容
时间
CLK 60
H_OP[11:0] 61
H_CALL #2
H_EXEC* 65
H_BUSY* 64
HDP[31:0] 72
RPREV
R4
R5
R5
HDP 被具有前一条指令结果的代码所驱动
H_DS* 70
H_DA* 71

图15

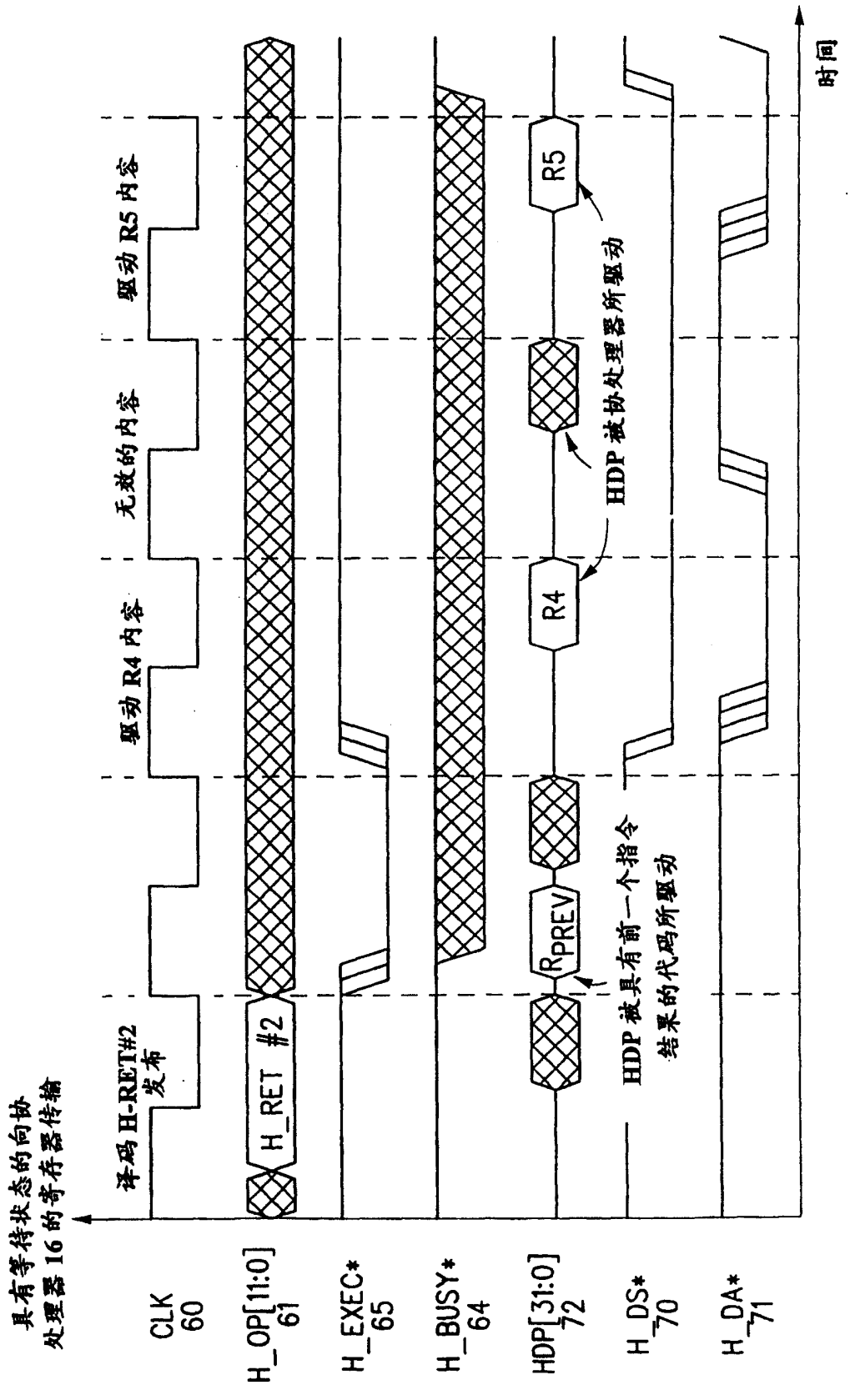

具有等待状态的向协
处理器16的寄存器传输
译码H-RET#2
发布
驱动R4内容
无效的内容
驱动R5内容
CLK
60
H_OP[11:0]
61
H_RET #2
H_EXEC*
65
H_BUSY*
64
HDP[31:0]
72
R4
R5
R PREV
HDP被协处理器所驱动
HDP被具有前一个指令
结果的代码所驱动
H_DS*
70
H_DA*
71
时间

图16

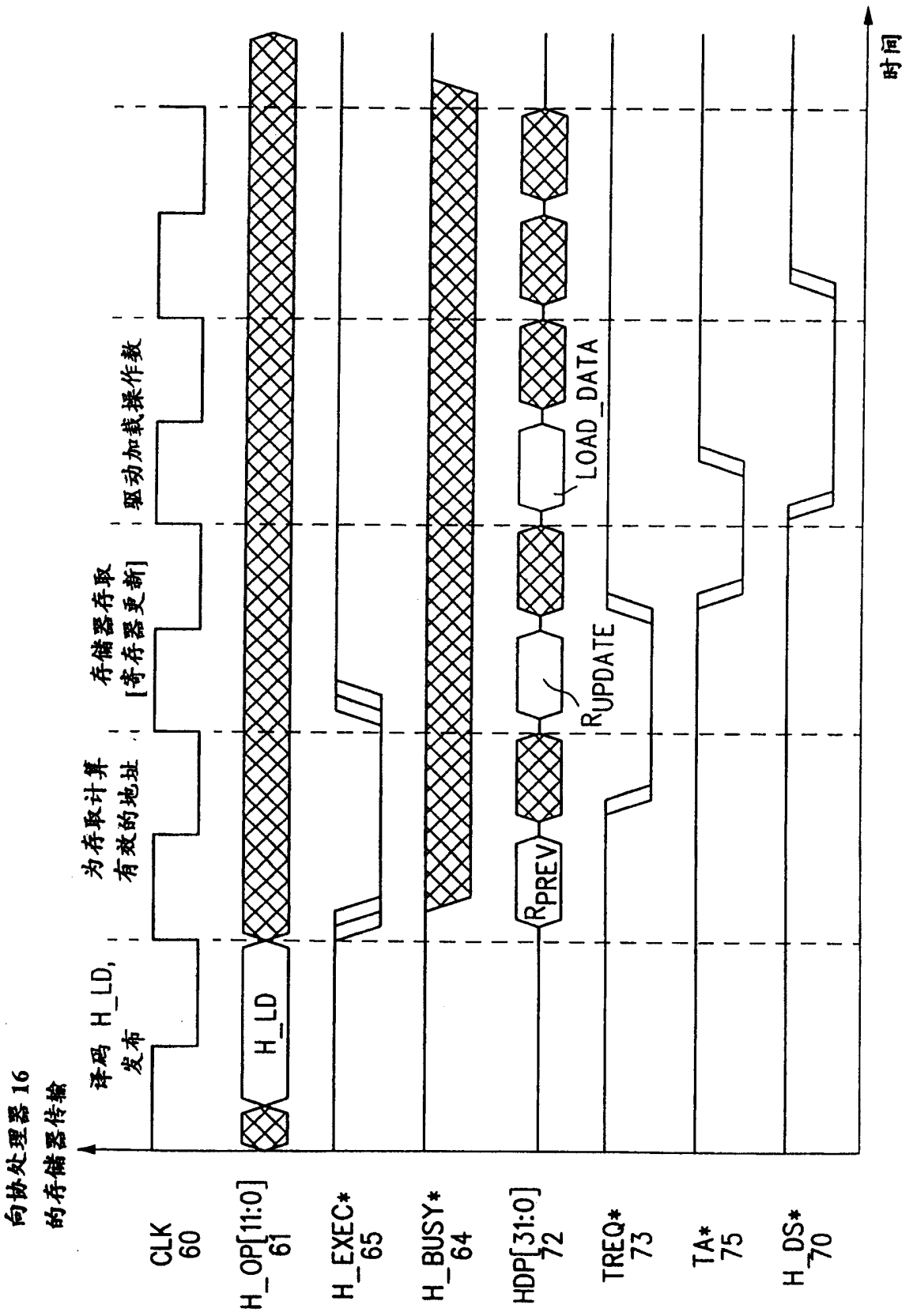

向协处理器 16
的存储器传输
译码 H_LD,
发布
为存取计算
有效的地址
存储器存取
[寄存器更新]
驱动加载操作数
时间
CLK
60
H_OP[11:0]
61
H_LD
H_EXEC*
65
H_BUSY*
64
HDP[31:0]
72
RPREV
RUPDATE
LOAD_DATA
TREQ*
73
TA*
75
H_DS*
70

图 17

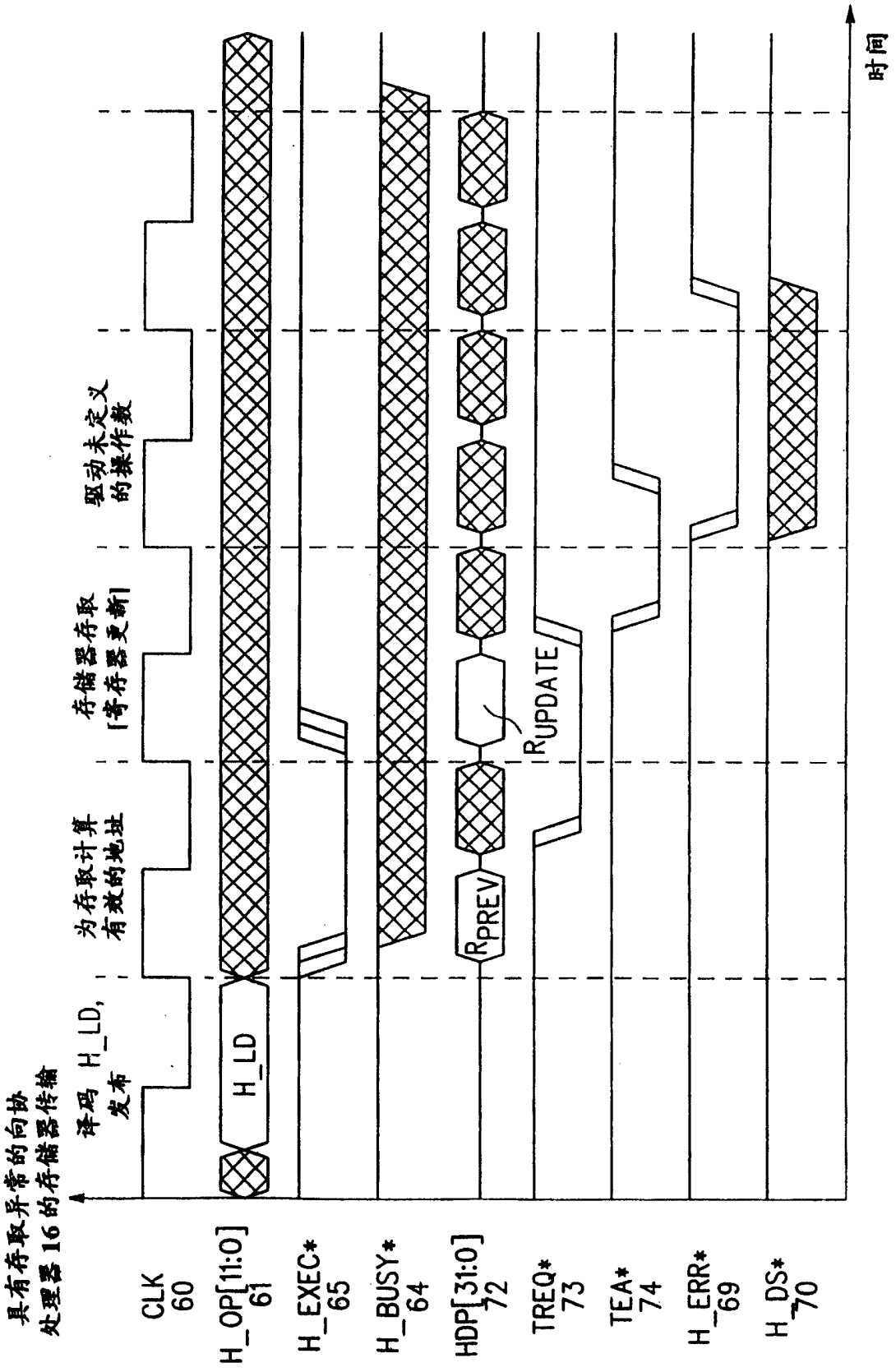

具有存取异常的向协
处理器16的存储器传输
译码 H_LD,
发布
为存取计算
有效的地址
存储器存取
[寄存器更新]
驱动未定义
的操作数
时间
CLK
60
H_OP[11:0]
61
H_LD
H_EXEC*
65
H_BUSY*
64
HDP[31:0]
72
RPREV
RUPDATE
TREQ*
73
TEA*
74
H_ERR*
69
H_DS*
70

图18

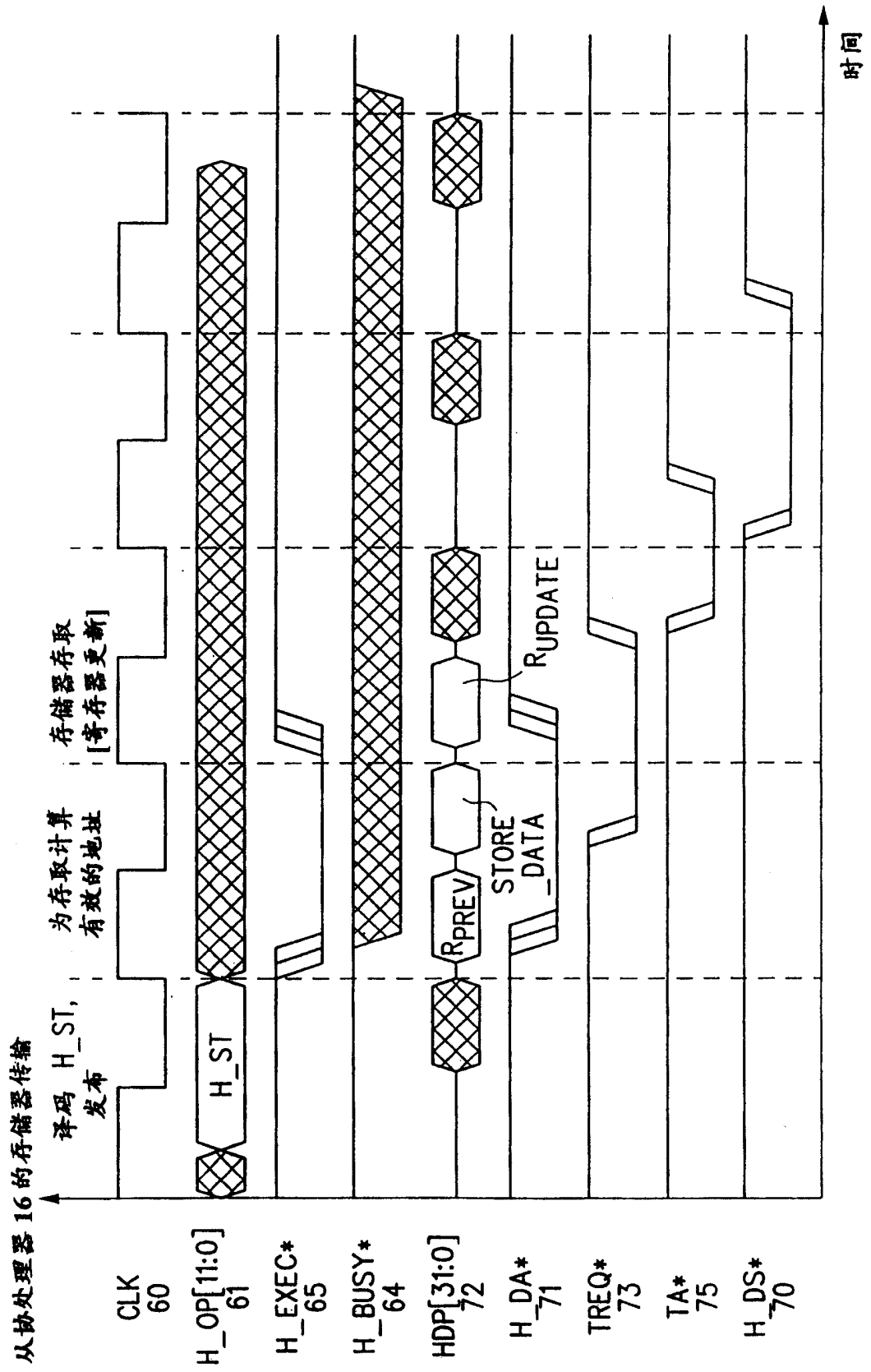

从协处理器 16 的存储器传输
译码 H_ST，发布
为存取计算有效的地址
存储器存取[寄存器更新]
CLK 60
H_OP[11:0] 61
H_ST
H_EXEC* 65
H_BUSY* 64
HDP[31:0] 72
RPREV
STORE_DATA
RUPDATE
H_DA* 71
TREQ* 73
TA* 75
H_DS* 70
时间

图 19

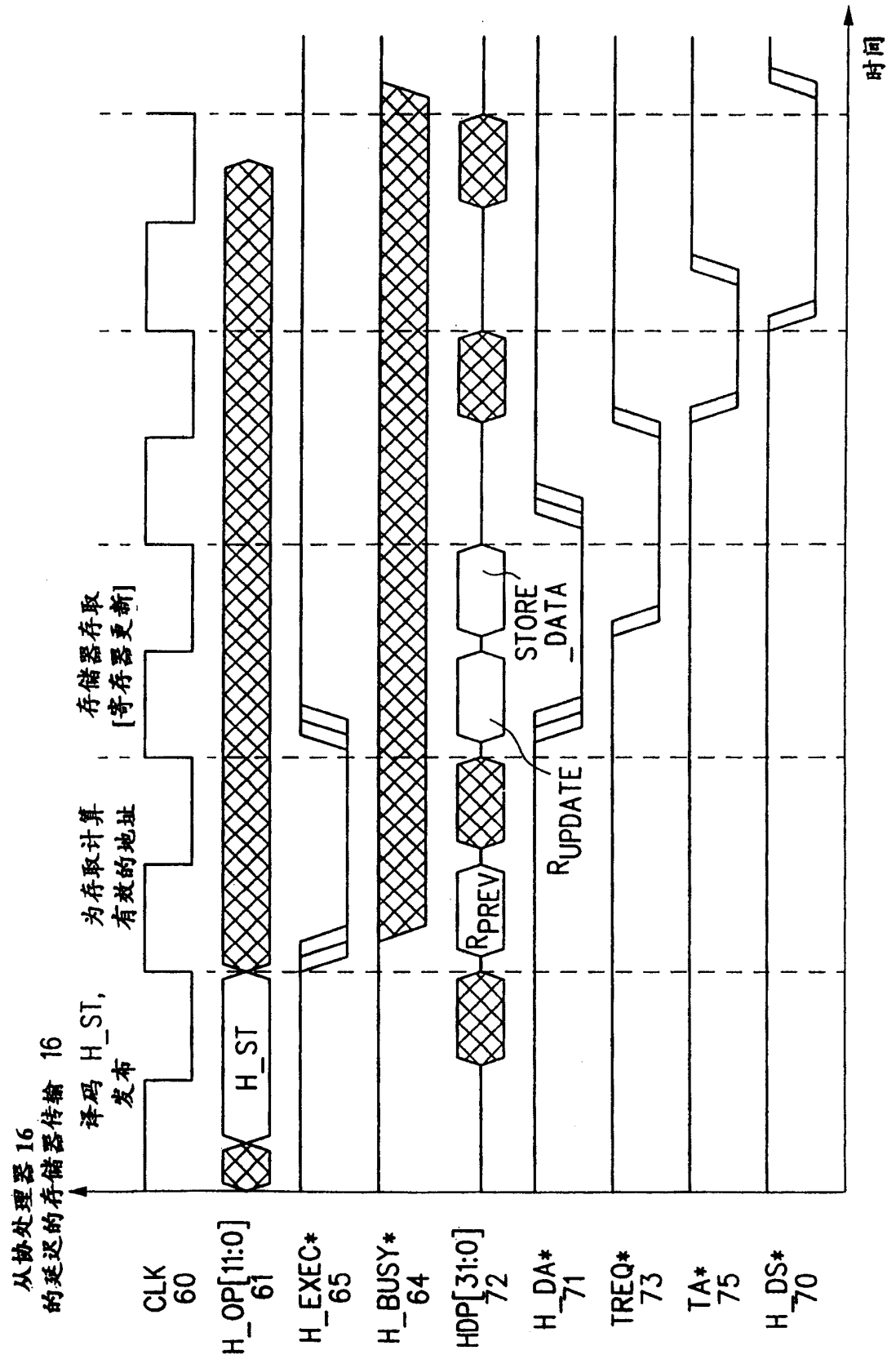
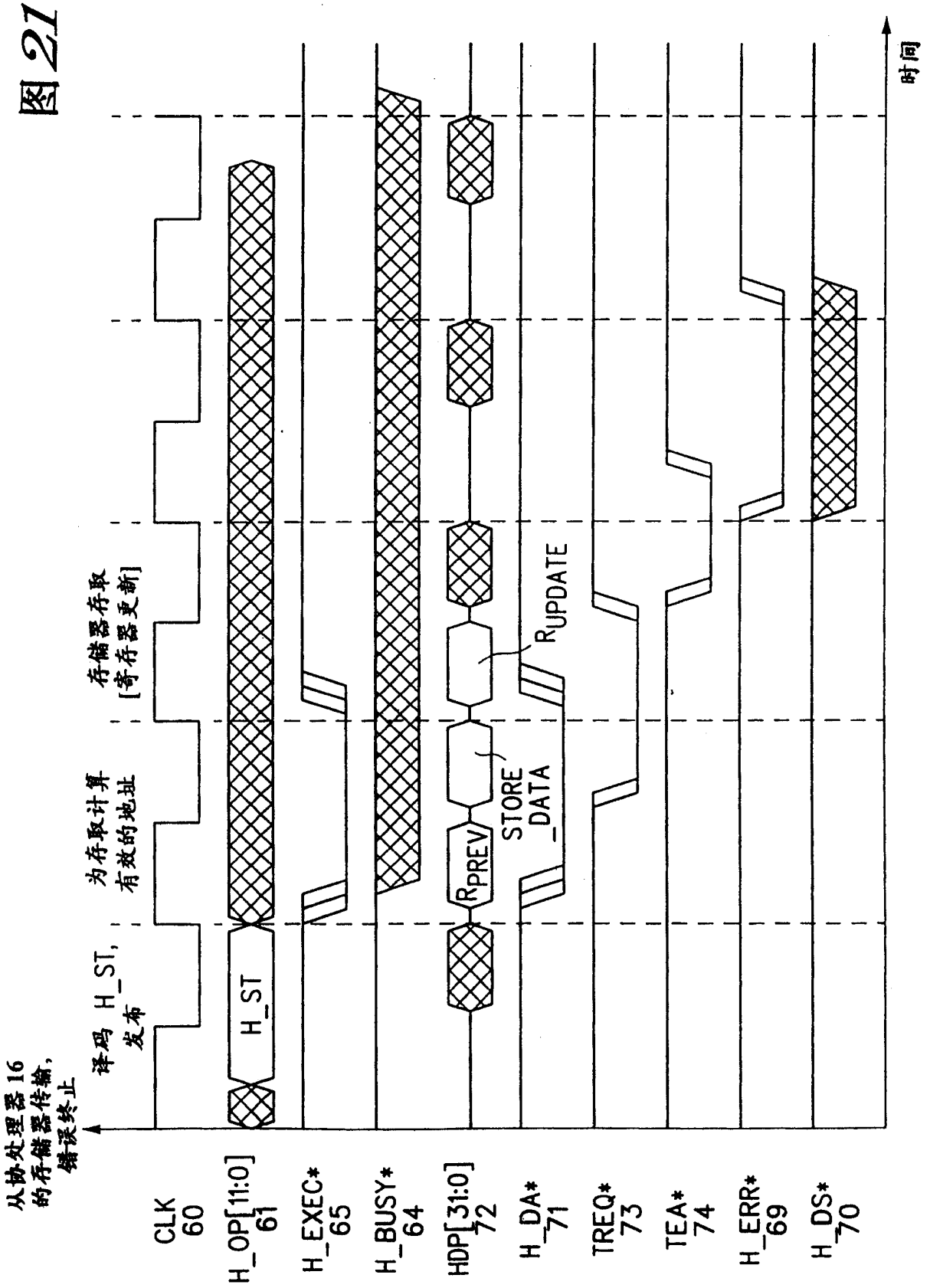

从协处理器16
的延迟的存储器传输 16
译码 H_ST,
发布
为存取计算
有效的地址
存储器存取
[寄存器更新]
CLK
60
H_OP[11:0]
61
H_ST
H_EXEC*
65
H_BUSY*
64
HDP[31:0]
72
RPREV
STORE
_DATA
RUPDATE
H_DA*
71
TREQ*
73
TA*
75
H_DS*
70
时间

图 20

图 21


从协处理器 16
的存储器传输，
错误终止
译码 H_ST，
发布
为存取计算
有效的地址
存储器存取
[寄存器更新]
时间
CLK
60
H_OP[11:0]
61
H_ST
H_EXEC*
65
H_BUSY*
64
HDP[31:0]
72
RPREV
STORE
_DATA
RUPDATE
H_DA*
71
TREQ*
73
TEA*
74
H_ERR*
69
H_DS*
70

H_CALL 硬件加速器（协处理器）调用原语

操作：向硬件加速器传递参数

汇编语法：H_CALL #UU, R4-RLAST, #CODE

描述：H-CALL 向硬件程序块（协处理器） UU 传递一个基于寄存器的参数集合和一个代码

条件代码：未受影响

指令格式

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	UU		0	1	1	CNT			CODE			

指令字段：

UU 字段-指定硬件程序块（协处理器）

00 – 程序块 0

01 – 程序块 1

10 – 程序块 2

11 – 程序块 3

CNT 字段-指定要传递的寄存器的编码，以 R4 开始

000-保留，未使用

001-传递 R4

⋮

111-传递 R4-R10

图 22

H_RET 硬件加速器（协处理器）返回原语

操作：从硬件加速器传递参数

汇编语法：H_RET #UU, R4-RLAST, #CODE

描述：H-RET 向协处理器#UU 传递一个代码并接收一个将要被加载到 CPU 寄存器中的返回参数的集合

条件代码：未受影响

指令格式

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	UU		0	1	0	CNT			CODE			

指令字段：

UU 字段-指定硬件程序块（协处理器）

00 - 程序块 0
01 - 程序块 1
10 - 程序块 2
11 - 程序块 3

CNT 字段-指定要传递的寄存器的编码，以 R4 开始 PASS, BEGINNING WITH R4

000 - 保留，未使用
001 - 传递 R4
010 - PASS R4-R5
⋮
111-传递 R4-R10

图 23

H_EXEC 硬件加速器（协处理器）执行原语

操作：向硬件加速器传递参数

汇编语法： H_EXEC #UU, #CODE

描述： H-EXEC 被用来控制协处理器#UU 中的一个函数，代码字段不是由 CPU 解释的

条件代码：未受影响

指令格式

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	UU		0	0	CODE							

7：指令字段：

UU 字段-指定硬件程序块（协处理器）

00-程序块 0

01-程序块 1

10-程序块 2

11-程序块 3

代码字段-为一个硬件程序块指定一个操作代码

图 24

H_LD 硬件加速器（协处理器）加载原语

操作：从存储器加载操作数并传递到硬件加速器

汇编语法：

H_LD.[HW][U] #UU, (RX, DISP)
H_LD.[U] #UU, (RX, DISP)

描述：H-LD 执行在存储器中加载一个值的操作，并且向协处理器传存储器操作数，而不将其存储在一个 GPR 中．H-LD 操作有三个选项，W—字，H—半字以及 U—更新．通过加载的大小以及零延伸来换算 IMM2 字段，以获取 DISP．将这个值加到寄存器 RX 的值上，并且从这个地址执行一个指定大小的加载，将加载的结果传送到硬件接口．对于半字加载，所取的数据是从零延伸到 32 位．如果指定 U 选项，则在计算之后，将加载的有效地址放入寄存器 RX 中．

条件代码：未受影响

指令格式

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	UU		1	SZ	0	UP	IMM2		RX			

指令字段：

UU 字段-指定硬件程序块（协处理器）

00-程序块 0
01-程序块 1
10-程序块 2
11-程序块 3

SIZE 字段-指定加载大小

0- 字
1- 半字

UP-指定是否应该更新基寄存器

0- 没有更新
1- 用有效地址更新基寄存器

IMM2 字段-指定一个 2 位的换算立即值

寄存器 X-指定将被加到换算的立即字段的基地址

图 25

H_ST 硬件加速器（协处理器）存储原语

操作：从硬件加速器向存储器存储操作数

汇编语法：

H_ST.[HW][U] #UU, (RX, DISP)

描述：H-ST 指令执行从一个协处理器向存储器存储一个操作数的操作，而不在 GPR 中存储该操作数．H-ST 操作有三个选项，W—字，H—半字以及 U—更新．通过存储的大小以及零延伸来换算 IMM2 字段以获取 DISP．这个值被加到寄存器 RX 的值上，并且用从硬件接口获得的用于存储的数据对这个地址执行一个指定大小的存储．如果指定 U 选项，则在计算之后将所加载的有效地址放入寄存器 RX 中．

条件代码：未受影响

指令格式

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	UU		1	SZ	1	UP	IMM2		RX			

指令字段：

UU 字段-指定硬件程序块（协处理器）

00-程序块 0

01-程序块 1

10-程序块 2

11-程序块 3

SIZE 字段-指定加载大小

0- 字

1- 半字

UP-指定是否应该更新基寄存器

0- 没有更新

1- 用有效地址更新基寄存器

IMM2 字段-指定一个 2 位的换算立即值

寄存器 X-指定将被加到换算的立即字段的基地址

图 26