(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 101809542 B
(45)授权公告日 2013.06.19

(21)申请号 200880108954.7
(22)申请日 2008.08.04
(30)优先权数据
11/864,292 2007.09.28 US
(85)PCT申请进入国家阶段日
2010.03.26
(86)PCT申请的申请数据
PCT/US2008/072109 2008.08.04
(87)PCT申请的公布数据
WO2009/045628 EN 2009.04.09
(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 W·C·莫耶
(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 11038
代理人 秦晨
(51)Int.Cl.
G06F 11/28(2006.01)
G06F 11/30(2006.01)
(56)对比文件
CN 1555526 A,2004.12.15,
US 6591378 B1,2003.07.08,
US 6321331 B1,2001.11.20,
审查员 郭全萍

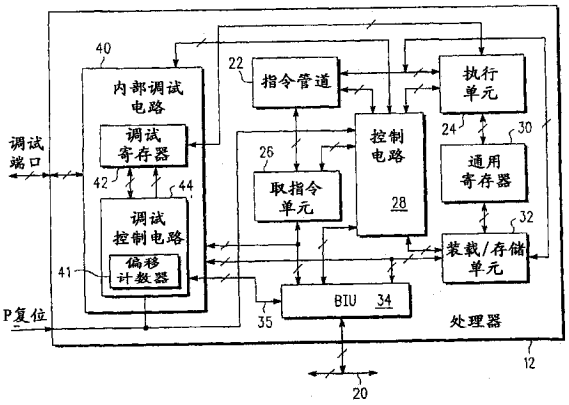
权利要求书2页 说明书20页 附图14页

(54)发明名称

用于监视调试事件的系统和方法

(57)摘要

一种系统具有用于通过顺序地取出、解码、执行及写入与每条指令的执行关联的结果来执行多条指令的流水线处理器(12)。调试电路(40)与流水线处理器耦连以用于监视指令的执行来确定调试事件发生的时间。该调试电路产生调试异常以中断指令处理流。该调试电路具有用于指示表示在引起调试事件的指令与获得调试异常时的在指令执行中的点之间完成指令执行的指令(如果有的话)的数量的值的控制电路(44)。



CN 101809542 B

1. 一种用于监视调试事件的系统，包括：

流水线处理器，用于通过顺序地取出、解码、执行及写入与每条指令的执行关联的结果来执行多条指令；以及

与所述流水线处理器耦连的调试电路，用于监视指令的执行以确定调试事件何时发生并产生调试异常以中断指令处理流，所述调试电路还包括控制电路，用于指示在引起调试事件的指令与响应于所述调试事件获得调试异常时的指令执行中的点之间通过进入所述流水线处理器的回写阶段完成指令执行的指令的数量的计数，

其中所述调试电路响应于对所述多条指令中的一条进行解码来确定调试事件已经发生，将由所述流水线处理器所形成的地址与一个或多个预定的调试地址进行比较以确定数据地址匹配是否已经发生，以及将由所述流水线处理器所存取的数据值与一个或多个预定的数据值进行比较以确定数据值匹配是否已经发生，所述调试事件需要发生两种比较操作的匹配。

2. 根据权利要求 1 所述的系统，其中所述调试电路的控制电路还包括：

用于计数并提供计数值的计数器，其中该计数值作为在引起调试事件的指令与响应于所述调试事件获得调试异常时的指令执行中的点之间通过进入所述流水线处理器的回写阶段完成指令执行的指令的数量的计数。

3. 根据权利要求 2 所述的系统，其中所述计数值大于 0，这指示：调试事件的处理是不准确的。

4. 根据权利要求 1 所述的系统，其中所述调试电路指示：引起调试事件的指令之后完成指令执行的一条或更多条附加的指令也引起 了调试事件发生。

5. 根据权利要求 1 所述的系统，其中所述流水线处理器执行其中每条都引起调试事件的至少两条指令，并且所述控制电路还包括多个计数器，每个计数器提供由引起调试事件的所述至少两条指令的相应一条之后完成指令执行的指令的数量所确定的相应的计数值。

6. 根据权利要求 1 所述的系统，其中所述调试电路还包括：

调试状态寄存器，具有用于保持在引起调试的指令与响应于所述调试事件获得调试异常时的指令执行中的所述点之间通过进入所述流水线处理器的回写阶段完成指令执行的指令的数量的计数的字段。

7. 一种用于监视调试事件的系统，包括：

流水线处理电路，用于通过顺序地取出、解码、执行及写入与每条指令的执行关联的结果来执行多条指令；以及

与流水线处理电路耦连的调试电路，用于通过将一个或多个调试地址与由指令执行所形成的地址进行比较并且将至少一个调试数据值与由指令执行所形成的数据进行比较来监视指令的执行，所述调试电路确定何时两种比较都导致匹配，以及，作为响应，指示调试事件，所述调试电路指示在引起调试事件的指令之后直到获得数据值断点时的指令执行中的预定点为止通过进入所述流水线处理器的回写阶段完成指令执行的指令的数量。

8. 根据权利要求 7 所述的系统，其中所述数据值断点是不准确的，并且所述调试电路还包括：具有指示不准确的数据值断点是否已经发生的位字段的寄存器。

9. 根据权利要求 7 所述的系统，其中所述调试电路还包括用于指示关于在调试事件发生的时间与获得数据值断点的时间之间发生的 事件的状态信息的状态寄存器。

10. 根据权利要求 9 所述的系统，其中在调试事件发生的时间与获得数据值断点的时间之间发生的事件之一还包括第二调试事件。

11. 根据权利要求 7 所述的系统，其中所述调试电路还包括：

用于计数并提供计数值的计数器，其中该计数值作为对在引起调试事件的指令之后直到获得数据值断点时的指令执行中的预定点为止通过进入所述流水线处理器的回写阶段完成指令执行的指令的数量的计数。

12. 根据权利要求 7 所述的系统，其中所述调试电路还包括：

状态调试寄存器，其具有用于保持指示引起调试事件的指令之后直到获得数据值断点时的指令执行中的预定点为止完成指令执行的附加的指令的数量的数值的字段。

13. 一种用于监视调试事件的方法，其特征在于包括：

通过顺序地取出、解码、执行及写入与每条指令的执行关联的结果来执行多条指令；

监视指令的执行以确定何时调试事件发生；

产生调试异常以中断指令处理流；

指示在引起调试事件的指令与响应于所述调试事件获得调试异常时的指令执行中的点之间通过进入流水线处理器的回写阶段完成指令执行的指令的数量；

响应于对多条指令中的一条进行的解码来确定调试事件已经发生；

将由流水线处理器所形成的地址与一个或多个预定的调试地址进行比较以确定数据地址匹配是否已经发生；以及

将由流水线处理器所形成的数据值与一个或多个预定的数据值进 行比较以确定数据值匹配是否已经发生，所述调试事件需要发生两种比较操作的匹配。

14. 根据权利要求 13 所述的方法，还包括：

对引起调试事件的指令之后完成指令执行的指令的数量的计数值进行计数并提供该计数值。

15. 根据权利要求 14 所述的方法，还包括：

形成大于 0 的计数值，这指示：调试事件的处理是不准确的。

16. 根据权利要求 13 所述的方法，还包括：

确定引起调试事件的指令之后完成指令执行的一条或更多条附加的指令也引起调试事件发生。

17. 根据权利要求 13 所述的方法，还包括：

确定其中每条都引起调试事件的至少两条指令；以及

在不同的计数器中存储：所述至少两条指令的每一条之后完成指令执行的附加的指令的数量的相应的计数值。

18. 根据权利要求 13 所述的方法，还包括：

在调试电路中提供调试状态寄存器，其中所述调试状态寄存器具有字段，所述字段用于保持指示引起调试事件的指令之后直到获得调试异常时的指令执行中的预定点为止完成指令执行的指令的数量的值。

用于监视调试事件的系统和方法

技术领域

[0001] 本公开内容一般地涉及数据处理系统，并且更特别地涉及用于监视调试事件的系统和方法。

背景技术

[0002] 数据值断点涉及比较数据存取地址，以及比较与数据存取地址关联的数据，以便确定是否应当产生断点异常。但是，当在典型的流水线处理器中时，数据可能直到存取地址可用之后的多个周期后才可用。因此，断点确切在指令流中的何处发生是不确定的，因为随数据存取之后的一条或更多条指令可以在存取数据还不可用的间隔内执行。在现今可用的一种数据处理系统中，附加的停车 (stall) 被添加到固有的执行中以便防止任何指令在存取数据仍不可用的间隔内执行。但是，这需要额外的开销并且干扰了正常的执行时序，这可能是调试过程中所不希望的。

附图说明

[0003] 本发明通过举例进行说明但并不限定于附图，在附图中相似的标记指示相似的元素。出于简单清晰的目的对附图中的元素进行了说明而并不一定按比例描绘。

[0004] 图 1 是根据本发明的一种实施方案的数据处理系统的框图。

[0005] 图 2 是根据本发明的一种实施方案与图 1 的数据处理系统关联的处理器的框图。

[0006] 图 3 是说明与图 1 的数据处理系统关联的示例性调试寄存器的图表。

[0007] 图 4 是根据本发明的一种实施方案与图 3 的调试寄存器关联的调试控制寄存器的图表。

[0008] 图 5 和图 6 以表格形式示出了图 4 中根据本发明的一种实施方案的调试控制寄存器的一部分的功能。

[0009] 图 7 是根据本发明的一种实施方案与图 3 的调试寄存器关联的调试状态寄存器的图表。

[0010] 图 8 以表格形式示出了图 7 中根据本发明的一种实施方案的调试状态寄存器的一部分的功能。

[0011] 图 9-11 示出了对图 2 的处理器的操作的不同实例进行说明的时序图。

[0012] 图 12-16 以表格形式示出了多种示例性数据地址比较 (DAC) 事件及其相应的结果。

[0013] 图 17 是根据本发明的一种实施方案与图 3 调试寄存器关联的调试状态寄存器的图表。

[0014] 图 18 示出了对图 2 的处理器的操作实例进行说明的时序图。

具体实施方式

[0015] 在调试 (debug) 代码时，通常难以了解实际上是哪条指令导致了所获得的调试异

常 (debug exception)。例如,在数据值断点的情形中,执行与所存取数据之间的数据存取地址比较及数据值比较,并且比较结果可能触发中断正常的指令流执行的调试异常。可以在存取初始化之后多个周期返回数据进行存取,这通过提供数据存取请求及存取地址来触发。在关于所请求存取的数据地址比较与关于所接收数据的数据值比较之间的滞后可能使得要确定是指令流中的哪条指令实际上导致了指令流中断尤为困难。在一种实施方案中,偏移值可以被用来指出在引起数据值断点的指令与指令流中实际发生断点异常的点之间执行了多少条指令。通过这种方式,可以获得改进的调试功能。

[0016] 如在此所使用的,术语“总线”被用来指示可以用来传输一种或多种不同类型的信息(例如数据、地址、控制、或状态)的多种信号或导体。在此所讨论的导体可以作为单导体、多导体、单向导体、或双向导体来说明或描述。但是,不同的实施方案可以改变导体的实现。例如,可以使用分离的单向导体而不是双向导体,反之亦然。此外,多导体可以用串行地或以时分复用的方式传输多种信号的单导体来代替。类似地,输送多种信号的单导体可以被分开成输送这些信号的子集的不同导体。因此,存在用于传输信号的多种选择。

[0017] 在此术语“确证 (assert)”和“置位 (set)”或“否定 (negate)”(或“取消确证 (deassert)”或“清除 (clear)”)在涉及使信号、状态位、或相似装置分别呈现出其逻辑真或逻辑假的状态时被使用。如果逻辑真的状态是逻辑电平 1,则逻辑假的状态是逻辑电平 0。而如果逻辑真的状态是逻辑电平 0,则逻辑假的状态是逻辑电平 1。

[0018] 在此所描述的每种信号可以被指定为正逻辑或负逻辑,其中负逻辑能够由信号名称上方的横线或名称后的星号(*)来指示。在负逻辑信号的情形中,信号是低有效的,其中逻辑真的状态对应于逻辑电平 0。在正逻辑信号的情形中,信号是高有效的,其中逻辑真的状态对应于逻辑电平 1。应当注意,在此所描述的任何信号都能够被指定为负逻辑信号或正逻辑信号。因此,在可替换的实施方案中,可以将那些描述为正逻辑信号的信号实现为负逻辑信号,并且可以将那些描述为负逻辑信号的信号实现为正逻辑信号。

[0019] 在此使用方括号以指示总线的导体或值的数位位置。例如,“总线 60[7:0]”或“总线 60 的导体 [7:0]”指示总线 60 的 8 个低阶导体,以及“地址位 [7:0]”或“地址 [7:0](ADDRESS[7:0])”指示地址值的 8 个低阶位。在数字之前的符号“$”指示该数字以其十六进制或基数 16 的形式来表示。在数字之前的符号“%”或“0b”指示该数字以其二进制或基数 2 的形式来表示。

[0020] 图 1 说明了根据本发明的一种实施方案的数据处理系统 10。数据处理系统可以是片上系统。数据处理系统 10 可以在单个集成电路或多个集成电路上实现。数据处理系统 10 包括可以通过总线 20 进行耦连的处理器 12、外部调试电路 14、I/O 模块 16、及存储器 18。在可替换的实施方案中,存储器 18 可以是任何类型的存储器以及可以位于与处理器 12 相同的集成电路上,或者位于与处理器 12 不同的集成电路上。存储器 18 可以是任何类型的存储器,例如,只读存储器 (ROM)、随机存取存储器 (RAM)、非易失性存储器(如闪存)等。此外,存储器 18 可以是位于另一外围设备或从动设备之内的或者位于不同的集成电路上的存储器或其它数据存储。

[0021] 图 2 是与图 1 的数据处理系统 10 关联的处理器 12 的框图。处理器 12 可以包括指令管道 22、执行单元 24、取指令单元 26、控制电路 28、通用寄存器 30、装载 / 存储单元 32、总线接口单元 (BIU)34 以及内部调试电路 40。处理器 12 可以通过与 BIU 34 耦连的总线

20 与数据处理系统 10 的其它组件通信。内部调试电路 40 可以通过图 2 所示的调试端口耦连至外部调试单元，例如 IEEE ISTO-5001 兼容性 Nexus™ 调试单元。Nexus™ 是位于德克萨斯州的奥斯汀市的 Freescale 半导体公司的商标。调试端口可以是串行接口，例如 JTAG，或者可以实现为并行端口，串行与并行端口的组合，或者以太网端口。内部调试电路 40 可以包括调试寄存器 42 和调试控制电路 44。调试控制电路 44 可以包括一个或更多个偏移计数器 41，其中偏移计数器 41 可以用来确定在引起调试事件的指令与指令执行中进行调试事件处理时的点之间完成指令执行的指令的数量（如果有的话）。调试寄存器 42 可以包括按字段分组用于控制各种调试关联的事件的位，包括指令断点、数据断点、监视点、以及与调试关联的其它消息接发。这些调试资源可以在处理器 12 与外部调试电路 14 之间共享。此外，调试控制电路 44 可以通过导体 35 与 BIU 34 进行地址及数据的通信。

[0022] 现在参考图 3，调试寄存器 42 内的寄存器也可以被提供用于存储一个或更多个地址比较值、地址范围、以及用于实现指令和 / 或数据存取断点及监视点事件的数据匹配值，以及其它调试控制准则。这些地址及数据值，与各种控制准则一起，被用来确定处理器 12 何时为了产生断点或监视点事件而存取一个或更多个预定的指令地址或数据地址，这能够在内部调试模式有效时引起处理器 12 开始进行用于调试异常的异常处理，或者在外部调试模式有效时引起处理器 12 进入调试暂停模式，在该调试暂停模式中处理器 12 响应于由外部调试电路 14 通过内部调试单元 40 的调试端口提供的命令。举例来说，调试寄存器 42 可以包括各种调试控制寄存器，包括调试控制寄存器 50(DBCR0) 以及其它调试控制寄存器 43(DBCR1、DBCR2、DBCR3、及 DBCR4)。调试寄存器 42 还可以包括指令地址比较寄存器 45(IAC1 和 IAC2)。指令地址比较寄存器 45 可以为了地址比较的目的而存储指令地址。调试寄存器 42 还可以包括数据地址比较寄存器 47(DAC1 和 DAC2)。数据地址比较寄存器 47 可以为了地址比较的目的而存储数据存取地址。调试寄存器 42 还可以包括调试状态寄存器 49、调试计数器 51(DBCNT1 和 DBCNT2)、及数据值比较寄存器 53(DVC1 和 DVC2)。调试寄存器 42 可以是用户软件编程模型的一部分。当一个或更多个计数启用事件发生时，调试计数器 51 可以被配置成倒计数 (count-down)。当计数值达到零时，调试计数事件的可以被信号发出，并且可以产生调试中断（若已启用）。数据值比较寄存器 53 可以为了数据比较的目的而存储数据值。

[0023] 在内部调试模式中，这些寄存器资源由软件来管理，并且不需要使用外部调试电路。软件可以通过使用移至专用寄存器或由专用寄存器移来的指令（其是编程器模型软件指令）进行的数据移动来配置寄存器以初始化各个调试寄存器用于执行基于软件的调试活动，其中启用的调试事件引起软件调试中断发生。然后，软件中断处理器可以执行由数据处理系统 10 的软件编程器确定的各种所希望的活动。在内部调试模式中，外部调试电路 14 被分派这些共享的调试事件配置寄存器的所有权，并且当所配置的调试事件发生时，处理器 12 可以进入暂停状态并等待将由外部调试电路 14 提供的命令。当外部调试模式被启用 (enable) 时，软件不再拥有对共享调试资源的控制。外部调试电路 14 可以存取共享的调试资源，包括直接通过调试端口（如图 2 所示）存取的调试寄存器 42，其中该调试端口例如可以被实现为 JTAG TAP 端口。在一种实施方案中，调试寄存器 42 可以通过在用于各种 JTAG 指令的一个或更多个字段内所包含的寄存器选择编码映射为 JTAG 数据寄存器，这提供了由调试器通过 JTAG IR 及 DR 操作对寄存器执行的读存取及写存取。

[0024] 单组寄存器的共享需要更少的处理器 12 资源来实现，并且这为数据处理系统 10 的用户简化了编程模型。内部调试单元 40 监视处理器 12 内的活动并且响应基于所存储的调试配置信息的一个或更多个预定条件的探测，可以产生一个或更多个数据断点事件、指令断点事件、指令执行事件（例如分支或陷阱获得事件）、指令完成事件等。在这种操作方式下，处理器 12 起着本领域技术人员能够意识到的作用。

[0025] 图 4 是与图 1 的数据处理系统关联的调试控制寄存器 50 的图表。可以将调试控制寄存器 50 作为调试寄存器 42 的一部分包含于其中，该调试寄存器 42 可以进一步作为内部调试单元 40 的一部分包含于其中。调试控制寄存器 50 可以用来存储调试配置信息。尽管图 4 说明了本发明的一种使用具体位字段的具体实施方案，但是本发明的可替换实施方案可以使用每个字段中位的数量不同的不同位字段。在图 4 中所描述的具体位字段仅出于说明性的目的来示出。举例来说，调试控制寄存器 50 可以包括 32 个位。调试控制寄存器 50 可以包括标记为 :EDM 52、IDM 54、ICMP 58、BRT 60、IAC1 62、IAC264、DAC1 66、DAC2 68、DVC1 70、及 DVC2 72 的位字段。这些位字段只是示例性的并且调试控制寄存器 50 可以包括更少的或附加的位字段。另外，这些数位字段可以进行不同的布置。调试控制寄存器 50 也可以包括可在将来使用的保留位字段 56、61、及 74。各种位字段的功能在下文针对图 5 和图 6 进行说明。举例来说，调试控制寄存器 50 可以是可写的寄存器，其中该可写寄存器也可以是可读的并且可以是用户软件编程模型的一部分。在本发明的可替换实施方案中，调试控制寄存器 50 可以不是用户软件编程模型中的控制寄存器，而是可以被实现于用户软件编程模型之外。任何类型的存储电路都可以用来实现调试控制寄存器 50。

[0026] 图 5 以表格形式示出了图 4 中的调试控制寄存器 50 的一部分的功能。EDM 位 52 可以指示外部调试模式是启用 (enable) 的还是禁用 (disable) 的。例如，当 EDM 位 52 被置为 1 时，控制寄存器（例如调试控制寄存器 56）被置于外部调试电路 14 的独占控制之下并且数据处理系统 10 软件不能够将信息写入这些控制寄存器中。作为选择，当 EDM 位 52 被置为 1 时，软件不能够写调试控制寄存器的具体部分。因而，EDM 位 52 被用来选择性地阻止某些重置事件清除调试控制寄存器 50 及其它调试资源中所存储的信息，其中能够包含调试控制及设置信息。IDM 位 54 可以指示内部调试模式是启用的还是禁用的，从而指示调试异常是启用的还是禁用的。ICMP 位 58 可以用来指示指令完成调试事件是启用的还是禁用的。BRT 位 60 可以用来指示分支获得调试事件是启用的还是禁用的。位 6:7 61 可以被保留以备将来使用。现在参考图 6，图 6 以表格形式示出了图 4 中的调试控制寄存器 50 的一部分的功能。IAC1 位 62 可以用来指示指令地址比较 1 调试事件是启用的还是禁用的。IAC2 位 62 可以用来指示指令地址比较 2 调试事件是启用的还是禁用的。DAC1 位 66 可以用来指示数据地址比较 1 调试事件是启用的还是禁用的。如果已启用，则 DAC1 位 66 还指示数据地址比较 1 调试事件对于何种类型的存储存取（例如，用于存储类型数据存储存取，用于装载类型数据存储存取，或者用于装载类型或存储类型数据存储存取）是被启用的。DAC2 位 68 可以用来指示数据地址比较 2 调试事件是启用的还是禁用的。如果已启用，则 DAC2 位 68 还指示数据地址比较 1 调试事件对于何种类型的存储存取（例如，用于存储类型的存储存取，用于装载类型数据存储存取，或者用于装载类型或存储类型数据存储存取）是被启用的。DVC1 位 70 可以用来指示数据值比较 1 限定器 (qualifier) 是否被启用。DVC2 位 72 可以用来指示数据值比较 2 限定器是否被启用。位 16:31 可以被保留以备将来使用。尽管图 5

和图 6 描述了具体数量的位字段以提供与调试事件关联的不同配置信息，但是也可以使用数量与这些图中所示出的数量不同的位字段。

[0027] 图 7 是与图 1 中的数据处理系统关联的调试状态寄存器 49 的图表。调试状态寄存器 49 可以被引入作为调试寄存器 42 的一部分，该调试寄存器 42 可以进一步被引入作为内部调试单元 40 的一部分。调试状态寄存器 49 可以被用来存储关于调试事件的状态信息。尽管图 7 说明了本发明的一种使用具体位字段的具体实施方案，但是本发明的可替换实施方案可以使用在每个字段中位的数量不同的不同位字段。图 7 中所描述的具体位字段只是出于说明性的目的来示出。举例来说，调试状态寄存器 49 可以包括 32 个位。调试状态寄存器 49 可以包括标记为 :IDE 76、ICMP 78、BRT 80、IAC1 82、IAC2 84、IAC3 86、IAC4 88、DAC1R 90、DAC1W 92、DAC2R 94、DAC2W96、及 DAC_OFST 98 的位字段。这些位字段只是示例性的并且调试状态寄存器 49 可以包括更少的或附加的位字段。另外，这些位字段可以进行不同的布置。调试状态寄存器 49 还可以包括可在将来使用的保留的位字段 100。各种位字段的功能在下文针对图 8 进行说明。此外，对于调试状态寄存器 49，位的置位 (setting) 涉及存储逻辑电平 1 并且位的清除涉及存储逻辑电平 0。举例来说，调试状态寄存器 49 可以是其位通过硬件来置位以及通过软件来读取及清除的寄存器，并且该寄存器可以是用户软件编程模型的一部分。在本发明的可替换实施方案中，调试状态寄存器 49 可以不处于用户软件编程模型内，而是可以被实现于用户软件编程模型之外。在一种实施方案中，只有在内部调试模式被启用或者外部调试模式被启用的时候，调试状态寄存器 49 的调试状态位才由调试事件来置位。此外，在一种实施方案中，当调试中断被启用时，调试状态寄存器 49 中的置位位可以引起调试中断产生，其中调试中断处理器负责在返回正常执行之前清除调试状态寄存器 49 的位。此外，任何类型的存储电路都可以用来实现调试状态寄存器 49。

[0028] 图 8 以表格形式示出了图 7 中的调试状态寄存器 49 的功能。IDE 位 76 被用来指示不准确的调试事件的发生并且从而可以在调试异常被禁用并且调试事件引起其各自的调试状态寄存器位被置为 1 时被置为 1。也就是，尽管调试事件可以发生，但是调试异常可以保持为禁用，因为由于流水线的当前状态中断还不能产生。如果指令完成调试事件发生，则 ICMP 位 78 可以被置为 1。如果分支获得调试事件发生，则 BRT 位 80 可以被置为 1。如果 IAC1 调试事件发生，则 IAC1 位 82 可以被置为 1。如果 IAC2 调试事件发生，则 IAC2 位 84 可以被置为 1。如果 IAC3 调试事件发生，则 IAC3 位 86 可以被置为 1。如果 IAC4 调试事件发生，则 IAC4 位 88 可以被置为 1。当 DAC1 位 66 等于 % 10 或 % 11（指示 DAC1 调试事件被启用以对装载类型的数据存储进行存取，如图 6 所示）时，如果读取类型的 DAC1 调试事件发生，则 DAC1R 位 90 可以被置为 1。当 DAC1 位 66 等于 % 01 或 % 11（指示 DAC1 调试事件被启用以对存储类型的数据存储进行存取，如图 6 所示）时，如果写入类型的 DAC1 调试事件发生，则 DAC1W 位 92 可以被置为 1。当 DAC2 位 68 等于 % 10 或 % 11（指示 DAC2 调试事件被启用以对装载类型的数据存储进行存取，如图 6 所示）时，如果读取类型的 DAC2 调试事件发生，则 DAC2R 位 94 可以被置为 1。当 DAC2 位 68 等于 % 01 或 % 11（指示 DAC2 调试事件被启用以对存储类型的数据存储进行存取，如图 6 所示）时，如果写入类型的 DAC2 调试事件发生，则 DAC2W 位 96 可以被置为 1。DAC_OFST 位 98 可以被用来指示数据地址比较的偏移。在一种实施方案中，位 13 ～ 31 被保留以备将来可能的使用。

[0029] 如果数据值比较限定器为调试控制寄存器 50 中的 DAC1 或 DAC2 而被指示，那么数

据存取地址匹配以及数据值匹配（对于与数据存取地址关联的数据值）必须为了将要指示的 DVC1 DAC1 或 DVC2 DAC2 调试事件而发生。也就是，直到数据值匹配也发生时才置位 DAC1R 位 90、DAC1W 位 92、DAC2R 位 94、及 DAC2W 位 96 中的每一个以指示 DVC1 DAC1 或 DVC2 DAC2 调试事件。应当注意，将要匹配的值能够被存储于调试寄存器 42 中，例如在数据地址比较寄存器 47 及数据值比较寄存器 53 中。例如，如果调试控制寄存器 50 的 DVC1 位 70 被置位了，那么 DAC1 调试事件由数据值比较所限定，意味着对于将要被指示的 DVC1 DAC1 调试事件，地址必须与 DAC1 数据地址比较寄存器匹配并且该地址的关联数据值必须与 DVC1 数据值比较寄存器匹配。但是，由于处理器 12 的流水线性质，数据可能直到存取地址可用之后的多个周期后才可用于比较。而且，即使在数据一旦可用以及 DBC DAC 被指示时，指令流在那一刻也可以是不可中断的（即，可以禁用调试异常），这意味着在 DVC

[0030] DAC 被指示时可以不获得实际数据值断点。在这种情形中，IDE 位 76 能够被置为 1 以指示不准确的调试事件，在该不准确的调试事件中调试异常在调试事件被指示时不能够被获得。也就是，直到调试异常启用后才能够获得实际数据值断点，这可以在随后的指令流中的不同的（以及不可预料的）点发生，如同将参考图 9-11 讨论的。一旦调试异常被获得，则中断处理就开始并且调试事件被处理。

[0031] 因此，DAC_OFST 位 98 可以被用来指示在引起 DVC DAC 被指示的指令与数据值断点产生以及调试异常被获得的点之间执行了的指令的数量。DAC_OFST 位 98 可以被用来存储所保存的 DSRR0 值相对装载或存储指令的地址的“偏移 -1”，其中该装载或存储指令获得了数据地址比较调试异常。应当注意，DSRR0 对应于调试存储恢复寄存器 0，而所保存的 DSRR0 值对应于断点中断的返回指针。因此，如果数据值断点发生，则所保存的 DSRR0 地址值对应随中断处理之前所执行的最后指令之后的指令的地址值。通过这种方式，能够确定有多少指令已经在引起DVC1 DAC1的地址与指令流被中断以获得数据值断点的点之间的间隔内执行了。这允许用户来确定实际上是哪条指令引起了数据值断点。在一种实施方案中，DAC_OFST 位 98 正常被置为％ 00，而 DVC DAC 则将该字段置为％ 01、％ 10、或％ 11，这表示“偏移 -1”。实例将在下文参考图 9-11 来提供。但是，应当注意，如果同时的转换旁视缓冲器 (translation look-asidebuffer) 丢失（也称作 DTLB 错误）或者数据存储中断（例如 DSI）错误由于存取许可错误或其它存取关联的错误而发生，则 DAC_OFST 位 98 可以被置为％ 00 并且 IDE 位 76 可以被置为 1。在这些情形中，应当注意，与地址关联的数据可以由于这些错误中的某一错误的发生而始终不可用。

[0032] 不同的状况及条件能够影响在执行数据地址比较的时间与实际获得数据值断点的时间之间执行的指令的数量。图 9-11 示出了说明一系列的四装载指令（由 10、11、12、及 13 表示，其中 10 引起 DVCDAC 调试事件）是如何导致不同的 DAC 偏移值的时序图。这些能够由 DAC 偏移计数器 41 保持追踪的 DAC 偏移值代表或编码在 10 与获得在 10 的 DVC DAC 被指示之后的数据值断点的调试异常的时间点之间执行的指令的数量。对于图 9-11，假定 10 产生 DVC1 DAC1 调试事件。因此，DAC1 位 66 是％ 10 或％ 11，使得 DAC1 调试事件被启用以用于装载类型数据存储存取，并且 DVC1 位 70 被置位以指示 DAC1 调试事件由数据值比较所限定。10 的 DVC1 DAC1 调试事件的对应状态位是 DAC1R 位 90，该 DAC1R 位 90 直到相同存取的每个地址值比较及数据值比较产生匹配后才会被置位。在该实例中，对于数据地址比较，应当注意的是数据地址比较寄存器 47 中的 DAC1 值可以用于与由 10 计算的数据地址

进行比较，以及，对于数据值比较，应当注意的是数据值比较寄存器 53 中的 DVC1 值可以用于与关联 10 的所引起的装载数据进行比较。在可替换的实施方案中，10 可以与 DAC2 DVC2 对应，使用 DAC2 位 68、DVC2 位 72、DAC2R 位 90 等。

[0033] 应当注意，图 9-11 中的每个图都参照 6 级流水线来描述，包括下列阶段：取出、解码、有效地址 (EA)/E0、存储器 1(mem1)/E1、存储器 2(mem2)/E2、及回写。应当注意，这被提供作为处理器 12 的指令管道 22 的一种流水线实例；但是，可替换的实施方案可以包括不同的流水线，例如具有不同的级数。指令流水线（例如图 9-11 中所参照的 6 级流水线）的操作是本领域已知的并且因此在这里将不作详细描述。此外，图 9-11 中的每个图都包括可以与处理器 12 的处理器时钟对应的时钟信号。处理器时钟能够是系统时钟或者是仅给处理器 12 的一部分提供的时钟，并且能够如同本领域已知的那样来产生。

[0034] 参照图 9，10 在时钟周期 1 内进入取出阶段，在时钟周期 2 内进入解码阶段，在时钟周期 3 内进入 EA/E0 阶段，其中 10 的有效地址在时钟周期 3 内被计算，并且然后在时钟周期 4 内进入 mem1/E1 阶段。因此，有效地址在时钟周期 3 结束时准备就绪并且被提供于到调试控制电路 44 的导体 35 的地址部分上使得地址比较能够在 10 的有效地址与数据地址比较寄存器 47 中所存储的 DAC1 值之间执行。（应当注意，在这一时间点地址还可以由 BIU 34 布置于总线 20 上。）该 DAC1 地址比较在时钟周期 4 内执行，如 DAC1 地址比较信号中的正脉冲所指示的，其中该 DAC1 地址比较信号能够是在调试控制电路 44 内确证以指示指令 10 的 DAC1 地址比较匹配正在发生的控制信号。但是，由于 DAC1 需要 DVC1 限定器（因为 DVC1 位 70 的置位），因而 DAC1R 调试事件还没有被指示，即使地址比较产生匹配。指令处理继续进行，因为与 10(10 将响应于 10 装载指令而被检出）关联的数据仍然不可用于数据值比较。10 然后进行到 mem2/E2 阶段，在该阶段中数据正被从存储器（例如，图 1 中的存储器 18）内装载。该装载数据在这个阶段结束时可以是可用的（并且例如由 BIU 34 通过总线 20 来接收）以及然后被提供于到调试控制电路 44 的导体 35 的数据部分上，使得数据值比较能够在关联 10 的装载数据值与存储于数据值比较寄存器 53 内的 DVC1 值之间执行。因此，数据值比较在下一时钟周期（时钟周期 6）内执行，如同 DVC1 数据比较信号中的正脉冲所指示的，其中该 DVC1 数据比较信号能够是在调试控制电路 44 内确证以指示指令 10 的 DVC1 数据比较匹配正在发生的控制信号。在与数据值比较寄存器 53 中所存储的 DVC1 值匹配时，则 DVC1 DAC1 调试事件就被指示，从而硬件对 DAC1R 位 90 进行置位。

[0035] 应当注意，当在周期 6 内探测到 10 上的 DVC1 DAC1 时，DAC 偏移计数器被清零并被启用以开始在每个后续指令完成时计数。但是，流水线在时钟周期 6 内还不能被中断，因为 11 和 12 处于存储器阶段 (mem1 和 mem2) 并且从而必须在对未决的调试异常进行进一步的处理之前完成，因为存储器存取一旦被初始化它们就不能被中断，而是必须完成。但是后续的存储器存取没有被初始化，因为 DVC1 DAC1 异常当前是未决的。11 和 12 的每次存储器存取在进入回写阶段后就被看作是完成的。应当注意，12（指令 11 和 12 的后者）在时钟周期 8 内处于回写阶段，在时钟周期 8 之后能够获得未决的调试异常并且能够开始中断处理。也就是，DVC1 DAC1 调试事件的中断处理直到时钟周期 8 之后（例如，在时钟周期 9 内）才开始。应当注意，在 DVC1 DAC1 被探测到时，13 只是处于流水线的 EA/E0 阶段并且因而能够被杀死。也就是，指令流中断于 13（即调试异常在 12 执行之后，13 执行之前被获得）。因此，正常的执行可以在中断处理完成之后从 13 再次开始。因此，DSRR0 寄存器可以

存储 13 的地址，因为该地址对应于数据断点调试中断的返回指针。在调试中断处理器的软件例程完成之后，通过执行从中断指令处的返回使正常的执行重新开始，这使用作为要继续正常的（非中断的）指令执行而返回到的指令（在此情形中为 13）的指针存储于 DSRR0 中的值。

[0036] 应当注意，在当探测到 DVC1 DAC1 调试事件时就清除 DAC 偏移计数器之后，随着每个后续指令完成时，DAC 偏移计数器就加 1 直到调试异常被获得（中断处理在该点发生）。因此，DAC 偏移计数器在时钟周期 8 内递增至 2，由于后续的指令 11 和 12 分别于时钟周期 7 和 8 完成，因此，调试状态寄存器 49 的 DAC_OFST 位 98 被置为％ 10。也就是，在引起 DVC1 DAC1 的指令（在本实例中为 10）之后且在调试事件中断处理之前执行的指令的数量是 2，其中该数量被存储为 DAC_OFST。在这种情形中，用户能够了解，数据值断点一旦发生，则该断点不是实际引起数据值断点的前一条指令（在本实例中为 12），而是在 12 之前 2 条多加指令（其中该值“2”对应于 DAC_OFST）。在 12 之前 2 条多加指令的那条指令是 10，其中该指令 10 在当前实例中确实引起了数据值断点。如上文针对 DAC_OFST 位 98 所讨论的，如果 DVC DAC 发生，则这指示所保存的 DSRR0 值相对获得 DAC 调试异常的指令（该指令是 10）的地址的“偏移 -1”。所保存的 DSRR0 值相对 10 的偏移（在本实例中对应于 13 的地址）是 3，并且 DAC_OFST 位 98 从而指示 2，其中该值 2 为“3-1”。应当注意，在可替换的实施方案中，能够使用不同的边界来计算 DAC_OFST 或者可以使用不同的计数方法以允许 DAC_OFST 来指示该实例中的 10。

[0037] 参照图 10，10 在时钟周期 1 内进入取出阶段，在时钟周期 2 内进入解码阶段，在时钟周期 3 内进入 EA/E0 阶段，其中 10 的有效地址在时钟周期 3 内被计算，以及在时钟周期 4 内进入 mem1/E1 阶段。有效地址被提供于到调试控制电路 44 的导体 35 的地址部分上使得地址比较能够在 10 的有效地址与存储于数据地址比较寄存器 47 中的 DAC1 值之间执行。10 的这种 DAC1 地址比较在时钟周期 4 内执行，并且地址比较匹配发生，如 DAC1 地址比较信号中的正脉冲所指示的。但是，由于 DAC1 需要 DVC1 限定器（因为 DVC1 位 70 的置位），因而 DAC1R 调试事件仍没有被指示，即使地址比较匹配发生。指令处理继续进行因为与 10（该指令 10 将响应于 10 装载指令而被检出）关联的数据还不可用于数据值比较。10 然后进行到 mem2/E2 阶段，在该 mem2/E2 阶段内数据正被从存储器（例如，图 1 中的存储器 18）中装载。但是，在时钟周期 5 内，10 的装载数据由存储器等待状态所停车 (stall)。（应当注意，10 的停车导致流水线停车，在该流水线停车中 11-13 的每个阶段也都在时钟周期 5 内停车。）因而，代替那个时钟周期 5 结束时提供 10 的装载数据（如图 9 那样），这里直到时钟周期 6 结束后才提供于到调试控制电路 44 的导体 35 的数据部分上。然后，数据值比较在下一时钟周期（时钟周期 7）执行，并且数据匹配发生，如 DVC1 数据比较信号中的正脉冲所指示的。在与数据值比较寄存器 53 中所存储的 DVC1 值匹配时，DVC1 DAC1 调试事件被指示，从而硬件对 DAC1R 位 90 进行置位。

[0038] 应当注意，当在周期 7 内探测到 10 的 DVC1 DAC1 时，DAC 偏移计数器被清零并且被启用以在每个后续指令完成时开始计数。但是，流水线在时钟周期 7 内还不能被中断，因为 11 处于存储器阶段 (mem2) 并且因而必须在对未决的调试异常进行进一步处理之前完成，因为存储器存取一旦被初始化，它们就不能够被中断，而是必须完成。但是后续的存储器存取没有被初始化，因为 DVC1 DAC1 异常当前是未决的。但是，在图 10 的实例中，指令 12

返回指示违反存取允许的DSI错误并且存储器从而将不给12返回任何装载数据。因此，即使12也处于存储器阶段(mem1)，它也能够由于DSI错误而被杀死(kill)。处于EA/E0阶段的13也能够被杀死，因为它还没有开始存储器存取，其中该存储器存取正常情况下在流水线阶段mem1内发生。指令11在时钟周期8内处于回写阶段，在该时钟周期8之后未决的调试异常能够被进一步处理使得调试异常能够被获得并且中断处理能够开始执行。也就是，DVC1 DAC1调试事件的中断处理直到时钟周期8后（例如，在时钟周期9中）才开始。因此，指令流在12中断（即调试异常在11执行之后及12执行之前被获得）。然后，正常的执行可以在调试中断的中断处理完成之后从12再次开始。因此，DSRR0寄存器可以存储12的地址，因为该地址对应于数据断点调试中断的返回指针。在调试中断处理器的软件例程完成之后，通过执行从中断指令处的返回使正常的执行重新开始，这使用作为要继续正常的（非中断的）指令执行而返回到的指令（在此情形中为12）的指针保存于DSRR0中的值。

[0039] 应当注意，在当探测到DVC1 DAC1时就清除DAC偏移计数器之后，随着每个后续指令完成时，DAC偏移计数器就加1直到调试异常被获得（中断处理在该点发生）。直到调试异常被获得（在其点中断处理发生）。因此，DAC偏移计数器在当前实例中只递增至1，因此，调试状态寄存器49的DAC_OFST位98被置为％01。也就是，在引起DVC1 DAC1的指令（在本实例中为10）之后且在调试事件中断处理之前执行的指令的数量是1，其中该数量被存储为DAC_OFST。在这种情形中，用户能够了解，数据值断点一旦发生，则该断点不是实际引起数据值断点的前一条指令（在本实例中为11），而是在12之前1条多加指令。在11之前1条多加指令的那条指令是10，其中该指令10在当前实例中确实引起了数据值断点。如上文针对DAC_OFST位98所讨论的，如果DVC DAC发生，则这指示所保存的DSRR0值相对获得DAC调试异常的指令（该指令是10）的地址的“偏移-1”。所保存的DSRR0值相对10的偏移（在本实例中对应于12的地址）是2，并且DAC_OFST位98从而指示2，其中该值1为“2-1”。应当注意，在可替换的实施方案中，能够使用不同的边界来计算DAC_OFST或者可以使用不同的计数方法以允许DAC_OFST来指示该实例中的10。

[0040] 参照图11，10在时钟周期1内进入取出阶段，在时钟周期2内进入解码阶段，在时钟周期3内进入EA/E0阶段，其中10的有效地址在时钟周期3内被计算，以及在时钟周期4内进入mem1/E1阶段。有效地址被提供于到调试控制电路44的导体35的地址部分上使得地址比较能够在10的有效地址与存储于数据地址比较寄存器47中的DAC1值之间执行。这种DAC1地址比较在时钟周期4内为10执行并且地址匹配发生，如DAC1地址比较信号中的正脉冲所指示的。但是，由于DAC1需要DVC1限定器（因为DVC1位70的置位），因而DAC1R调试事件仍没有被指示，即使地址比较匹配发生。指令处理继续进行因为与10（该指令10将响应于10装载指令而被检出）关联的数据还不可用于数据值比较。10然后进行到mem2/E2阶段，在该mem2/E2阶段内数据正被从存储器（例如，图1中的存储器18）中装载。10的装载数据在那个时钟周期5结束时准备就绪，并且被提供于到调试控制电路44的导体35的数据部分上。然后，数据值比较在下一时钟周期（时钟周期6）执行，并且数据匹配发生，如DVC1数据比较信号中的正脉冲所指示的。在与数据值比较寄存器53中所存储的DVC1值匹配时，DVC1 DAC1调试事件被指示，从而硬件对DAC1R位90进行置位。

[0041] 当在周期6内探测到10上的DVC1 DAC1时，DAC偏移计数器被清零并被启用以

在每个后续指令完成时开始计数。在图 11 的实例中，应当注意的是存储器阶段 mem1 内的 11 由于转换旁视缓冲器 (TLB) 中的地址转换丢失引起了错误，并且因而 11 没有继续执行。（应当注意，TLB 没有在附图中示出，但是能够与存储器管理单元 (MMU) 一起被布置于处理器 12 中，如本领域所已知的，其中 TLB 和 MMU 两者都可以如本领域所已知的那样操作）。因此，在时钟周期 6 中，指令 11-13 全部都能够被杀死。因此，调试异常能够在时钟周期 7 内立即启用，使得调试异常能够被获得并且中断处理能够开始执行。也就是，指令 10 的 DVC1 DAC1 调试事件的中断处理能够在时钟周期 6 之后（例如，在时钟周期 7 中）开始。因此，指令流在 11 中断（即调试异常在 10 执行之后及 11 执行之前被获得）。然后，正常的执行可以在中断处理完成之后从 11 再次开始。因此，DSRR0 寄存器可以存储 11 的地址，因为该地址对应于数据断点调试中断的返回指针。在当前的实例中，DAC 偏移计数器保持为 0 ；因此，调试状态寄存器 49 的 DAC_OFST 位 98 保持为％ 00。也就是，引起 DVC1 DAC1 的指令（在本实例中为 10）是在被中断的指令 (11) 之前的指令。然后，正常的执行可以在调试中断的中断处理完成之后从 11 再次开始。因此，DSRR0 寄存器可以存储 11 的地址，因为该地址对应于数据断点调试中断的返回指针。在调试中断处理器的软件例程完成之后，通过执行从中断指令处的返回使正常的执行重新开始，这使用作为要继续正常的（非中断处理）指令执行而返回到的指令（在此情形中为 11）的指针保存于 DSRR0 中的值。

[0042] 因此，应当注意，关于 DVC DAC 的数据值断点的时序能够根据各种因素来改变，其中许多因素是不可预料的以及事先不知道的。但是，通过使用 DAC_OFST，调试异常一旦被实际获得，用户就能够确定是哪条指令实际上引起了数据值断点（即实际上引起了要被获得的调试异常）。

[0043] 在某些情况下，例如当用户已经通过用中断屏蔽控制对调试中断进行明确的屏蔽来禁用它们时，IDE 位 76 则通知用户调试事件（例如 DVC DAC 调试事件）已经发生，但它是不准确的因为调试异常已经被用户全局禁用了。也就是，即使调试事件发生了，但调试异常当前正由用户屏蔽着。因此，一旦调试异常在对调试中断解除了屏蔽之后被实际获得，也不能使用 DSC_OFST 值来准确判定是哪条指令引起调试异常发生的，因为许多指令可以在用户对调试中断再次解除屏蔽之前被执行。I 在这种情形中，DE 位 76 可以由软件使用以限定 DAC_OFST 的有效性，因为它不反映所执行指令的实际数量。

[0044] 图 12-16 以表格形式示出了说明 DAC 和 DVC DAC 调试事件的发生的多种实例以及在一种实施方案中说明调试状态寄存器 49 的结果更新，例如 DAC_OFST 位 98。在图 12-16 的表格中，一系列的三指令，10、11 及 12，被使用于每个实例。第一指令，10，是装载 / 存储类指令，第二指令，11，除非另有说明否则是装载 / 存储类指令，以及第三指令，12，除非另有说明否则是装载 / 存储指令。应当注意，图 12 包括行 201-208，图 13 包括行 209-212，图 14 包括行 213-215，图 15 包括行 216-218，以及图 16 包括行 219-220。

[0045] 参考代表一个实例的表格的行 201，假定是 10 导致了 DTLB 错误并且没有 DAC 调试事件。在这种情形中，DTLB 异常被获得并且对调试状态寄存器没有更新被执行。行 202 代表一个实例，在该实例中 10 导致了数据存储中断 (DSI) 错误并且没有 DAC 调试事件。在这种情形中，DSI 异常被获得并且对调试状态寄存器没有更新被执行。行 203 代表一个实例，在该实例中 10 导致了 DTLB 错误 ；但是，DAC 调试事件被指示（例如 DAC1 或 DAC2 调试事件）。在这种情形中，作为 DACx 调试事件的结果而引起的调试异常被获得，并且对应的 DACx 位字

段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W）并且 IDE 位 76 被置位，并且 DAC_OFST 位 98 被置为％ 00。DSRR0 寄存器指向 10（即存储 10 的地址）。行 204 代表一个实例，在该实例中 10 导致了 DSI 错误；但是，DACx 调试事件被指示（例如 DAC1 或 DAC2 调试事件）。在这种情形中，作为 DACx 调试事件的结果而引起的调试异常被获得，并且对应的 DACx 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W）并且 IDE 位 76 被置位，并且 DAC_OFST 位 98 被置为％ 00。DSRR0 寄存器指向 10（即存储 10 的地址）。行 205 代表一个实例，在该实例中 10 导致了 DACx 调试事件（例如 DAC1 或 DAC2 调试事件）。在这种情形中，作为 DACx 调试事件的结果而引起的调试异常被获得，并且对应的 DACx 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W），并且 DAC_OFST 位 98 被置为％ 00。DSRR0 寄存器指向 11（即存储 11 的地址）。应当注意，对于在行 203 和 204 中所指示的条件，IDE 位 76 被置位，但是对于行 205，IDE 位 76 则没有被置位。这被用来指出行 203 和 204 的 DAC_OFST 置为％ 00 指示着指令 10 引起了中断发生，以及 DSRR0 中所保存的程序计数器值当在 10 上的调试中断由于 DAC1 或 DAC2 事件而被获得时正指向着 10 而不是指向正常的位置（11）。这发生了是因为在 10 上还存在同时的 DTLB 或 DSI 异常，并且从而 10 应当在调试中断后被重新执行，因为它还没有完成执行。在重新执行时，DAC1、DAC2 事件可以被用户禁用，以及正常的 DTLB 或 DSI 异常然后被获得并被适当地处理。但对于行 205，指令 10 已经完成执行，因而在 DAC_OFST 值为％ 00 时所保存的 DSRR0 值指向 11，并且 IDE 被清除，指示 DSRR0 中对于 DAC_OFST 值为％ 00 时的正常边界条件。

[0046] 行 206-220 代表了多个实例，在这些实例中 10 引起了 DVCxDACx 调试事件（例如 DVC1DAC1 或 DVC2DAC2 调试事件）。行 206 代表一个实例，在该实例中 11 没有引起异常并且能够是任何指令并且 13 没有引起异常并且不是装载 / 存储类型的指令。在这种情形中，作为 DVCx DACx 调试事件的结果而引起的调试异常被获得，并且对应的 DACx 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W），并且 DAC_OFST 位 98 被置为％ 01。DSRR0 寄存器指向 12 之后的指令。通过检查 DAC_OFST 位 98，能够判定 10 引起了 DVCx DACx 事件。

[0047] 行 207 代表与图 9 的实例相似的一个实例，在该实例中 11 没有引起异常并且 13 没有引起异常，但 13 是装载 / 存储类指令。在这种情形中，作为 DVCx DACx 调试事件的结果而引起的调试异常被获得，并且对应的 DACx 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W），并且 DAC_OFST 位 98 被置为％ 10。DSRR0 寄存器指向 12 之后的指令。在该实例中，指令 12 应当允许完成，因为它已经开始了存储器存取。行 208 代表与图 11 的实例相似的一个实例，在该实例中 11 引起 DTLB 错误且没有 DAC。在这种情形中，作为 DVCx DACx 调试事件的结果而引起的调试异常被获得，并且对应的 DACx 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W），并且 DAC_OFST 位 98 被置为％ 00。DSRR0 寄存器指向 11，因为它招致了 DTLB 错误，且没有完成执行。通过检查 DAC_OFST 位 98，能够判定 10 引起了 DVCx DACx 事件。

[0048] 行 209 代表一个实例，在该实例中 11 引起了 DSI 错误且没有 DAC。在这种情形中，作为 DVCx DACx 调试事件的结果而引起的调试异常被获得，并且对应的 DACx 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W），并且 DAC_OFST 位 98 被置为％ 00。DSRR0 寄存器指向 11，因为它招致了 DSI 异常并且没有完成。行 210 代表一个实例，在该实例中 11 引起了 DTLB 错误及 DACy（例如 DAC1 或 DAC2 调试事件）。在这种情形中，作为 DVCx DACx 调试事件的结果而引起的调试异常被获得，并且对应的 DACx 位字段被置位（例如 DAC1R、DAC1W、DAC2R、

DAC2W)，并且 DAC_OFST 位 98 被置为％ 00。DSRR0 寄存器指向 11。在这种情形中，11 的 DACy 事件没有被报告，因为它由于 11 上的 DTLB 错误无论如何都应当被重新执行。行 211 代表一个实例，在该实例中 11 引起了 DSI 错误及 DACy（例如 DAC1 或 DAC2 调试事件）。在这种情形中，作为 DVCx DACx 调试事件的结果而引起的调试异常被获得，并且对应的 DACx 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W)，并且 DAC_OFST 位 98 被置为％ 00。DSRR0 寄存器指向 11。在这种情形中，11 的 DACy 事件没有被报告，因为它由于 11 上的 DTLB 错误无论如何都应当被重新执行。应当注意，对于行 208-211 的实例，11 异常被屏蔽；但是，这是随实现而定的并且在其它处理器上可以是不同的。

[0049] 行 212 代表一个实例，在该实例中 11 引起了 DACy（例如 DAC1 或 DAC2 调试事件）。在这种情形中，调试异常被获得，对应的 DACx 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W)，对应的 DACy 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W)，并且 DAC_OFST 位 98 被置为％ 01。DSRR0 寄存器指向 12。行 213 代表一个实例，在该实例中 11 引起了 DVCy DACy（例如 DVC1 DAC1 或 DVC2 DAC2 调试事件）并且是正常的装载 / 存储指令而在该实例中 12 不是装载 / 存储指令。在这种情形中，调试异常被获得，并且对应的 DACx 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W)，对应的 DACy 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W)，并且 DAC_OFST 位 98 被置为％ 01。DSRR0 寄存器指向 12。还应当注意，在这种情形中，如果 x 等于 y，那么调试状态寄存器与 DSRR0 的结果状态 (resultant state) 可能无法区别于“无 DACy”的情形（即其中 12 没有引起 DACy 的情形）。

[0050] 行 214 代表一个实例，在该实例中 11 引起了 DVCy DACy（例如 DVC1 DAC1 或 DVC2 DAC2 调试事件）并且是正常的装载 / 存储指令并且在该实例中 12 是没有引起异常的装载 / 存储指令。在这种情形中，在 12 完成之后获得调试异常，因为它已经开始存储器存取，并且对应的 DACx 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W)，对应的 DACy 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W)，并且 DAC_OFST 位 98 被置为％ 10。DSRR0 寄存器指向 12 之后的指令。还应当注意，在这种情形中，如果 x 等于 y，那么调试状态寄存器与 DSRR0 的结果状态可能无法区别于“无 DACy”的情形（即其中 12 没有引起 DACy 的情形）。

[0051] 行 215 代表一个实例，在该实例中 11 引起了 DVCy DACy（例如 DVC1 DAC1 或 DVC2 DAC2 调试事件）并且是正常的装载 / 存储指令并且在该实例中 12 引起了 DSI 错误。行 216 代表一个实例，在该实例中 11 引起了 DVCy DACy（例如 DVC1 DAC1 或 DVC2 DAC2 调试事件）并且是正常的装载 / 存储指令并且在该实例中 12 引起了 DTLB 错误。在这些情形的任何一种情形中，调试异常被获得，并且对应的 DACx 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W)，对应的 DACy 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W)，并且 DAC_OFST 位 98 被置为％ 01。DSRR0 寄存器指向 12。还应当注意，在这种情形中，如果 x 等于 y，那么调试状态寄存器与 DSRR0 的结果状态可能无法区别于“无 DACy”的情形（即其中 12 没有引起 DACy 的情形）。此外，应当注意，在这些情形中，指令 12 被屏蔽；但是，这种行为是随实现而定的并且在其它处理器上可以是不同的。

[0052] 行 217 代表一个实例，在该实例中 11 引起了 DVCy DACy（例如 DVC1 DAC1 或 DVC2 DAC2 调试事件）并且是正常的装载 / 存储指令并且在该实例中 12 是正常的装载 / 存储指令或者是引起 DACy 或 DVCy DACy 的多字装载 / 存储指令。在这种情形中，调试异常被获得，并且对应的 DACx 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W)，对应的 DACy 位字

段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W），并且 DAC_OFST 位 98 被置为％ 10。DSRR0 寄存器指向 12 之后的指令。还应当注意，在这种情形中，如果 x 等于 y，那么调试状态寄存器与 DSRR0 的结果状态可能无法区别于“无 DACy”情形（即其中 11 或 12 没有引起 DACy 的情形）。

[0053] 行 218 代表一个实例，在该实例中 11 引起了 DVCy DACy（例如 DVC1 DAC1 或 DVC2 DAC2 调试事件）并且是多字装载 / 存储指令并且在该实例中 12 可以是任何指令。在这种情形中，调试异常被获得，并且对应的 DACx 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W），对应的 DACy 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W），并且 DAC_OFST 位 98 被置为％ 01。DSRR0 寄存器指向 12。在这种情形中，因为 11 装载或存储多寄存器，所以有足够的时间来防止 12 存取存储器，一 10 的 DVCxDACx 条件发生，则它就可以被杀死。还应当注意，在这种情形中，如果 x 等于 y，那么调试状态寄存器与 DSRR0 的结果状态可能无法区别于“无 DACy”情形（即其中 12 没有引起 DACy 的情形）。

[0054] 行 219 代表一个实例，在该实例中 11 可以是任何指令并且没有引起异常并且在该实例中 12 是正常的装载 / 存储或者是引起 DSI 错误以及可能或可能不引起 DAC 的多字装载 / 存储指令。在这种情形中，调试异常被获得，并且对应的 DACx 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W），对应的 DACy 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W），并且 DAC_OFST 位 98 被置为％ 01。DSRR0 寄存器指向 12。还应当注意，在这种情形中，指令 12 被屏蔽；但这种行为是随实现而定的并且在其它处理器上可以是不同的。

[0055] 行 220 代表一个实例，在该实例中 11 可以是任何指令并且没有引起异常并且在该实例中 12 是正常的装载 / 存储或者是引起 DACy 或 DVCy DACy 的多字装载 / 存储指令。在这种情形中，调试异常被获得，并且对应的 DACx 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W），对应的 DACy 位字段被置位（例如 DAC1R、DAC1W、DAC2R、DAC2W），并且 DAC_OFST 位 98 被置为％ 10。DSRR0 寄存器指向 12 之后的指令。还应当注意，在这种情形中，如果 x 等于 y，那么调试状态寄存器与 DSRR0 的结果状态可能无法区别于“无 DACy”情形（即其中 12 没有引起 DACy 的情形）。

[0056] 图 17 是根据本发明的另一种实施方案与图 1 的数据处理系统关联的调试状态寄存器 49 的图表。在图 17 的实例中，调试状态寄存器 49 包括标记为：IDE 76、ICMP 78、BRT 80、IAC1 82、IAC2 84、IAC3 86、IAC4 88、DAC1R 90、DAC1W 92、DAC2R 94、DAC2W96、DAC_OFSTA 102、OCCA 104、DAC_OFSTB 106、及 OCCB108 的位字段。图 17 的实例还可以包括保留的位字段 110，该位字段 110 可以被保留以备将来使用。应当注意，以上（例如，在对图 7 和图 8 的讨论中）针对调试状态寄存器 49 以及位字段 IDE 76、ICMP78、BRT 80、IAC1 82、IAC2 84、IAC3 86、IAC4 88、DAC1R90、DAC1W 92、DAC2R 94、及 DAC2W 96 所提供的说明同样应用于图 17，并且因此将不在这里重复。但是，与图 7 的实例不一样，图 17 的实例包括的是多 DAC 偏移字段（例如 DAC_OFSTA 102 和 DAC_OFSTB 106）而不是 DAC_OFST 98。多 DAC 偏移字段的使用使其具有保持追踪多偏移的能力，其中每个 DAC 偏移字段都可以如上文参考 DAC_OFST 98 所描述的那样来操作。但是，由于多 DAC 偏移字段的存在，每个 DAC 偏移字段（例如 DAC_OFSTA 102 和 DAC_OFSTB 106）都具有对应的发生字段（例如分别为 OCCA 104 和 OCCB 108）。

[0057] 因此，DAC_OFSTA 位 102 可以用来指示已经在引起要被指示的第一 DVC DAC(DVC1

DAC1 或 DVC2 DAC2) 的指令与数据值断点发生及 DVC DAC 的调试异常被获得的点之间执行的指令的数量。DAC_OFSTA 位 102 可以用来存储所保存的 DSRR0 值相对装载或存储指令的地址的“偏移 -1”，其中所述装载或存储指令以与 DAC_OFST 98 相同的方式获得了第一数据地址比较调试异常。如上文所讨论的，DSRR0 对应于调试保存恢复寄存器 0，并且所保存的 DSRR0 值对应于调试中断的返回指针。通过这种方式，能够确定有多少指令已经在引起第一 DVC DAC 的地址与指令流于其上中断以获得数据值断点（其中该数据值断点可以被获得作为第一 DVC DAC 或另一 DVC DAC 的结果）的点之间的间隔内执行了，这允许用户判定是哪条指令引起了该第一 DVC DAC。在一种实施方案中，DAC_OFSTA 位 102 在正常情况下置为％ 00，并且 DVC DAC 将把该字段置为％ 00、％ 01、％ 10、或％ 11，这表示“偏移 -1”。因为 DVC_OFSTA 位 102 在默认下正常情况置为％ 00，OCCA 104 可以在第一 DVC DAC 发生时置位，以指出 DAC_OFSTA 保持着指示 DVCDAC 偏移的值。因此，DAC_OFSTB 位 106 可以用来指示已经在引起要被指示的第二 DVC DAC 的指令与数据值断点发生以及 DVCDAC 的调试异常被获得的点之间执行的指令的数量。DAC_OFSTB 位 106 可以用来存储所保存的 DSRR0 值相对获得第二数据地址比较调试异常的装载或存储指令的地址的“偏移 -1”。通过这种方式，还能够确定有多少指令已经在引起第二 DVC DAC 的地址与指令流于其上中断以获得数据值断点（这里再次，数据值断点可以被获得作为第一 DVC DAC 或第二 DVC DAC 的结果）的点之间的间隔内执行了。在一种实施方案中，DAC_OFSTB 位 106 在正常情况下置为％ 00，并且 DVC DAC 将把该字段置为％ 00、％ 01、％ 10、或％ 11，这代表“偏移 -1”。因为 DAC_OFSTB 位 106 在默认下正常情况置为％ 00，OCCB 108 可以在第二 DVC DAC 发生时置位。（此外，如上所述，应当注意，如果 DVC DAC 的同时的 DTLB 错误或 DSI 错误发生，则对应的 DAC 偏移位可以被置为％ 00 并且 IDE 位 76 可以被置为 1，如上文参考 DAC_OFST 位 98 所讨论的。）

[0058] 因此，应当注意，指令的数量可以导致 DVC DAC，以及在指令流被中断以获得数据值断点的点上，假定对应的发生位被置位了，则偏移字段 DAC_OFSTA 102 和 DAC_OFSTB 104 中的每个都能够指示是指令流中的哪条（或哪些）指令引起了其中可能已经引起数据值断点的 DVC DAC。如果 OCCA 位 104 和 OCCB 位 108 两者都被置位了，指示着两个 DVC DAC 已发生，用户可以使用对应的偏移值以指示是哪条指令实际上引起了每个 DVC DAC，但是用户可能有必要实际确定是这两条指令中的哪一条（即该两个 DVC DAC 中的哪一个）实际上导致了中断指令流的数据值断点。

[0059] 图 18 示出了说明多偏移字段的使用的时序图，如同参考图 17 所讨论的。对于图 18 的实例，假定 10-14 中的每条都是装载指令，并且 10 产生第一 DAC DVC 调试事件 (DVCx DACx，其中 x 能够是 1 或 2) 并且 12 产生第二 DAC DVC 调试事件 (DVCy DACy，其中 y 能够是 1 或 2)。图 18 同样参照 6 级流水线来描述；但是，可替换的实施方案可以包括不同的流水线，例如具有不同的级数。指令流水线（例如图 18 中所参照的 6 级流水线）的操作，是本领域已知的并且因此在这里将不作详细描述。此外，图 18 包括可以与处理器 12 的处理器时钟对应的时钟信号。处理器时钟能够是系统时钟或者是仅给处理器 12 的一部分提供的时钟，并且能够如同本领域已知的那样来产生。

[0060] 参照图 18，10 在时钟周期 1 内进入取出阶段，在时钟周期 2 内进入解码阶段，在时钟周期 3 内进入 EA/E0 阶段，其中 10 的有效地址在时钟周期 3 内被计算，并且然后在时钟周期 4 内进入 mem1/E1 阶段。因此，有效地址在时钟周期 3 结束时准备就绪并且被提供

于到调试控制电路 44 的导体 35 的地址部分上使得地址比较能够在 10 的有效地址与存储于数据地址比较寄存器 47 中的 DAC 值 (DAC1 和 DAC2) 之间执行。(应当注意，在这一时间点地址还可以由 BIU 34 布置于总线 20 上。) 指令 10 的这种 DACx 地址比较在时钟周期 4 内执行，并且匹配发生，如 DAC 地址比较信号中的正脉冲所指示的，其中该 DAC 地址比较信号能够是在调试控制电路 44 内确证以指示 DACx 地址比较正在执行并且匹配已经发生的控制信号。但是，由于 DACx 需要 DVCx 限定器 (因为 DVCx 位的置位，例如 DVC1 位 70 或 DVC2 位 72)，所以调试事件还没有被指示，即使 DACx 地址比较发生匹配。指令处理继续进行，因为与 10(该指令 10 将响应于 10 装载指令被检出) 关联的数据还不可用于数据值比较。10 然后进行到 mem2/E2 阶段，在该 mem2/E2 阶段内数据正被从存储器 (例如，图 1 中的存储器 18) 中装载。该装载数据在该阶段结束时可以是可用的 (以及例如由 BIU 34 通过总线 20 来接收) 并且然后被提供于到调试控制电路 44 的导体 35 的数据部分上使得数据值比较能够在关联 10 的装载数据值与存储于数据值比较寄存器 53 中的 DVCx 值之间执行。因此，数据值比较在下一时钟周期 (时钟周期 6) 内执行，并且匹配发生，如 DACx 数据比较信号中的正脉冲所指示的，其中该 DACx 数据比较信号能够是在调试控制电路 44 内确证以指示 DAC 数据比较正在执行的控制信号。当与数据值比较寄存器 53 中所存储的 DVCx 值匹配时，DVCx DACx 调试事件被指示，从而硬件对相应的状态位 (例如 DAC1R 位 90 或 DAC2R 位 94，取决于 10 是引起了 DAC1 DVC1 还是 DAC2 DVC2) 进行置位。

[0061] 应当注意，当在周期 6 内探测到 10 的 DVCx DACx 时，DAC 偏移 A 计数器被清零并被启用以开始在每个后续指令完成时计数。(DAC 偏移 A 计数器可以包含于计数器 41 中。) 但是，流水线在时钟周期 6 内还不能被中断，因为 11 和 12 处于存储器阶段 (mem1 和 mem2) 并且从而必须在对未决的调试异常进行进一步的处理之前完成，因为存储器存取一旦被初始化，它们就不能被中断，而是必须完成。但是后续的存储器存取没有被初始化，因为 DVC1 DAC1 异常当前是未决的。11 和 12 的每次存储器存取在进入回写阶段后就被看作是完成的。应当注意，12(指令 11 和 12 的后者) 在时钟周期 8 内处于回写阶段，在时钟周期 8 之后能够处理未决的调试异常使得调试异常能够被获得并且能够开始中断处理。也就是，DVC1 DAC1 调试事件的中断处理直到时钟周期 8 之后 (例如，时钟周期 9) 才能开始。应当注意，在 DVC1 DAC1 被探测到时，13 只是处于流水线的 EA/E0 阶段并且因而能够被杀死。也就是，指令流中断于 13。因此，执行可以在中断处理完成之后从 13 再次开始。因此，DSRR0 寄存器可以存储 13 的地址，因为该地址对应于数据断点调试中断的返回指针。在调试中断处理器的软件例程完成之后，通过执行从中断指令处的返回使正常的执行重新开始，这使用作为要继续正常的 (非中断的) 指令执行而返回到的指令 (在此情形中为 13) 的指针存储于 DSRR0 中的值。

[0062] 应当注意，在当探测到 DVCx DACx 调试事件时就清除 DAC 偏移 A 计数器之后，随着每个后续指令完成时，DAC 偏移 A 计数器就加 1 直到调试异常被获得 (中断处理在该点发生)。因此，DAC 偏移 A 计数器在时钟周期 8 内递增至 2，因此，调试状态寄存器 49 的 DAC_OFSTA 位 102 被置为％ 10 并且 OCCA 位 104 也被置位。也就是，在引起 DVCx DACx 的指令 (在本实例中为 10) 之后且在调试事件中断处理之前执行的指令的数量是 2，其中该数量被存储为 DAC_OFSTA。

[0063] 应当注意，12 在时钟周期 5 内进入 EA/E0 阶段。因此，有效地址在时钟周期 5 结

束时准备就绪并且被提供于到调试控制电路44的导体35的地址部分上使得地址比较能够在12的有效地址与数据地址比较寄存器47中所存储的DAC值(DAC1和DAC2)之间执行。(应当注意,在这一时间点地址还可以由BIU 34布置于总线20上。)该DACy地址比较在时钟周期6内执行,并且匹配发生,如DACy地址比较信号中的正脉冲所指示的,其中该DACy地址比较信号能够是在调试控制电路44内确证以指示DAC地址比较正在执行并且匹配正在发生的控制信号。但是,由于DAC1需要DVCy限定器(因为DVCy位的置位,例如DVC1位70或DVC2位72),因而调试事件还没有被指示,即使DACy地址比较发生匹配。指令处理继续进行,因为与12(该指令12将响应于12装载指令而被检出)关联的数据仍然不可用于数据值比较。12然后进行到mem2/E2阶段,在该阶段中数据正被从存储器(例如,图1中的存储器18)内装载。该装载数据在这个阶段结束时可以是可用的(以及例如由BIU34通过总线20来接收)并且然后被提供于到调试控制电路44的导体35的数据部分上,使得数据值比较能够在关联12的装载数据值与存储于数据值比较寄存器53内的DVCy值之间执行。因此,数据值比较在下一时钟周期(时钟周期8)内执行,并且匹配发生,如DVCy数据比较信号中的正脉冲所指示的,其中该DVCy数据比较信号能够是在调试控制电路44内确证以指示DVC数据比较正在执行以及匹配正在发生的控制信号。在与数据值比较寄存器53中所存储的DVCy值匹配时,则DVCy DACy调试事件就被指示,从而硬件对相应的状态位(例如DAC1R位90或DAC2R位94,取决于12是引起了DAC1 DVC1还是DAC2 DVC2)进行置位。

[0064] 应当注意,当在周期8内探测到12的DVCy DACy时,DAC偏移B计数器被清零并被启用以开始在每个后续指令完成时计数。(DAC偏移B计数器可以包含于计数器41中。)流水线在时钟周期8内还不能被中断,因为11和12两者都必须在数据值断点响应于10所引起的DVCx DACx被获得之前完成。因此,与12对应的数据也被返回,使DVCy DACy能够在获得调试异常之前发生。如上所述,指令流在周期8之后中断于13,意味着在获得调试异常时DAC偏移B计数器没有递增并保持为0。因此,调试状态寄存器49的DAC_OFSTB位106被置为%00并且OCCB位108也被置位(这将使用户能够了解到在还没有第二DVC DAC发生时%00指示实际的偏移值而不仅仅返回默认值)。因此,在引起DVCy DACy的指令(在本实例中为12)之后且在中断处理之前执行的指令的数量是0。通过这种方式,应当注意,DAC_OFSTB位106和OCCB位108能够提供关于在第一调试事件(例如DVCx DACx)与该第一调试事件的调试异常被获得的点之间发生的调试事件(例如DVCy DACy)的信息。

[0065] 因此,应当注意,在图18的实例中,两个DVC DAC在周期8之后获得调试异常之前发生,其中每个DVC DAC由不同的指令引起。与对应的OCCA位104和OCCB位108一起使用DAC_OFSTA位102和DAC_OFSTB位104,应当了解的是10(在指令12之前2条多加指令,对应于指令流中断的点)和12(在指令12之前0条多加指令,对应于指令流中断的点)每条指令都引起了DVC DAC。然后,用户能够译解是哪条指令实际上导致了数据值断点(该指令在本实例中为10,它引起了第一DVC DAC,DVCx DACx)。还应当注意,x和y可以是不同的使得10引起DVC1 DAC1或DVC2 DAC2而12引起DVC1 DAC1及DVC2 DAC2中的另一个。作为选择,x可以等于y,使得10和12两者都引起了DVC1 DAC1或DVC2DAC2。此外,应当注意,在一种实施方案中,DAC A偏移计数器和DAC B偏移计数器可以位于偏移计数器41内。

[0066] 因此,能够认识到一个或更多个偏移值可以被如何使用以帮助确定在调试事件发生与调试异常获得之间(即在调试事件发生与调试事件的中断处理起始之间)执行的指令

的数量。这可以允许进行改进的调试，在该改进的调试中用户能够使用该一个或更多个偏移值以更好地了解是哪条指令实际上引起了将被获得的调试异常。尽管以上说明已经针对导致数据值断点的 DVC DAC 调试事件进行了提供，但是这些偏移值（例如 DAC_OFST、DAC_OFSTA、及 DAC_OFSTB）也可以使用于其它类型的调试事件。也就是，偏移字段（例如 DAC_OFST、DAC_OFSTA、及 DAC_OFSTB）可以用来指示在任何类型的调试事件发生与对应的调试异常获得之间（即在任何类型的调试事件发生与具体调试事件的中断处理起始之间）执行的指令的数量。

[0067] 此外，应当注意，了解自调试事件发生以来已经执行的指令的数量，而非已经引起该事件的指令的位置可能是有帮助的，因为能够理解相同的指令（具有相同的位置）可以在达到流水线边界之前多次执行，其中指令流可以为了未决的调试事件在该流水线边界中断，因而位置信息单独可能不足以确定事件的实际次序。

[0068] 在一种实施方案中，一种系统包括用于通过顺序地取出、解码、执行及写入与每条指令的执行关联的结果来执行多条指令的流水线处理器，以及与流水线处理器耦连的用于监视指令的执行以确定调试事件发生的时间并产生调试异常以中断指令处理流的调试电路，该调试电路还包括用于指示表示在引起调试事件的指令与获得调试异常时的在指令执行中的点之间完成指令执行的指令（如果有的话）的数量的值的控制电路。

[0069] 在另一种实施方案中，调试电路的控制电路还包括用于计数并提供计数值的计数器，其中该计数值作为表示在引起调试事件的指令之后完成指令执行的指令的数量的值。在另一种实施方案中，计数值大于 0，这指示出调试事件的处理是不准确的。

[0070] 在另一种实施方案中，调试电路指示随引起调试事件的指令之后完成指令执行的一条或更多条附加的指令也引起了调试事件发生。

[0071] 在另一种实施方案中，流水线处理器执行其中每条都引起调试事件的至少两条指令以及控制电路还包括多个计数器，其中每个计数器提供由随引起调试事件的该至少两条指令的各自一条之后完成指令执行的附加的指令的数量所确定的各自的计数值。

[0072] 在另一种实施方案中，调试电路还包括调试状态寄存器，其中该调试状态寄存器具有用于保持表示随引起调试事件的指令之后直到获得调试异常时的在指令执行中的预定点之前完成指令执行的附加的指令（如果有的话）的数量的值的字段。

[0073] 在另一种实施方案中，调试电路响应于对多条指令中的一条进行的解码来确定调试事件已经发生，将由流水线处理器所形成的地址与一个或更多个预定的调试地址进行比较以确定数据地址匹配是否已经发生，以及将由流水线处理器所存取的数据值与一个或更多个预定的数据值进行比较以确定数据值匹配是否已经发生，其中该调试事件需要两种比较操作中的一种匹配来发生。

[0074] 在再一种实施方案中，一种系统包括用于通过顺序地取出、解码、执行及写入与每条指令的执行关联的结果来执行多条指令的流水线处理电路，以及与流水线处理电路耦连的用于通过选择性地将一个或更多个调试地址与由指令执行所形成的地址进行比较以及选择性地将至少一个调试数据值与由指令执行所形成的数据进行比较来监视指令的执行的调试电路，其中该调试电路确定两种比较发生匹配的时间以及，作为响应，指示调试事件，该调试电路指示在引起调试事件的指令之后直到获得数据值断点时的在指令执行中的预定点之前所执行的指令的数量。

[0075] 在该再一种实施方案的另一种实施方案中，数据值断点是不准确的以及测试电路还包括具有指示不准确的数据值断点是否已经发生的位字段的寄存器。

[0076] 在该再一种实施方案的另一种实施方案中，测试电路还包括用于指示关于在调试事件发生的时间与获得数据值断点的时间之间发生的事件的状态信息的状态寄存器。在另一种实施方案中，在调试事件发生的时间与获得数据值断点的时间之间发生的事件之一还包括第二调试事件。

[0077] 在该再一种实施方案的另一种实施方案中，调试电路还包括用于计数并提供随引起调试事件的指令之后完成指令执行的指令的数量的计数值的计数器。

[0078] 在该再一种实施方案的另一种实施方案中，调试电路还包括状态调试寄存器，具有用于保持指示随引起调试事件的指令之后直到获得数据值断点时的在指令执行中的预定点之前完成指令执行的附加的指令（如果有的话）的数量的数值的字段。

[0079] 在又一种实施方案中，一种方法包括通过顺序地取出、解码、执行及写入与每条指令的执行关联的结果来执行多条指令，监视指令的执行以确定调试事件发生的时间，产生调试异常以中断指令处理流，以及指示在引起调试事件的指令与获得调试异常时的在指令执行中的点之间完成指令执行的指令（如果有的话）的数量。

[0080] 在该又一种实施方案的另一种实施方案中，方法还包括计数以及提供随引起调试事件的指令之后完成指令执行的指令的数量的计数值。在另一种实施方案中，方法还包括形成大于 0 的计数值，这指示出调试事件的处理是不准确的。

[0081] 在该又一种实施方案的另一种实施方案中，方法还包括确定随引起调试事件的指令之后完成指令执行的一条或更多条附加的指令也引起了调试事件发生。

[0082] 在该又一种实施方案的另一种实施方案中，方法还包括确定其中每条都引起调试事件的至少两条指令，以及在不同的计数器中存储随该至少两条指令的每一条之后完成指令执行的附加的指令的数量的各自的计数值。

[0083] 在该又一种实施方案的另一实施方案中，方法还包括在调试电路中提供调试状态寄存器，其中该调试状态寄存器具有用于保持指示随引起调试事件的指令之后直到获得调试异常时的在指令执行中的预定点之前完成指令执行的附加的指令（如果有的话）的数量的值的字段。

[0084] 在该又一种实施方案的另一种实施方案中，方法还包括响应于对多条指令中的一条进行的解码来确定调试事件已经发生，将由流水线处理器所形成的地址与一个或更多个预定的调试地址进行比较以确定数据地址匹配是否已经发生，以及将由流水线处理器所形成的数据值与一个或更多个预定的数据值进行比较以确定数据值匹配是否已经发生，其中该调试事件需要两种比较操作中的一种匹配来发生。

[0085] 因为实现本发明的装置大部分包括本领域技术人员所已知的电子组件及电路，为了本发明的基本概念的了解及理解以及为了不使本发明的内容模糊或分散，除了上文所说明的认为必要的电路细节以外将不对其它更多的电路细节进行说明。

[0086] 在此所使用的术语“程序”被定义为用于在计算机系统上执行而设计的的指令序列。程序或计算机程序可以包括子程序、函数、过程、对象方法、对象实现、可执行的应用程序、小程序 (applet)、服务小程序 (servlet)、源代码、目标代码，共享库 / 动态装载库和 / 或用于在计算机系统上执行而设计的其它指令序列。

[0087] 以上实施方案中的一些（若适用）可以使用多种不同的信息处理系统来实行。例如，尽管图 1 及其讨论描述了示例性的信息处理体系结构，但是该示例性的体系结构只是为了在本发明的各个方面的讨论中提供有用的参考而示出。当然，为了讨论的目的已经简化了对体系结构的描述，并且它只是可以根据本发明来使用的许多不同类型的适合的体系结构中的一种。本领域技术人员将认识到在逻辑块之间的边界仅仅是说明性的并且可替换的实施方案可以合并逻辑块或电路元件或者对各种逻辑块或电路元件实行可替换的功能分解。

[0088] 从而，应当了解在此所描述的体系结构只是示例性的，并且实际上能够实现获得相同功能的许多其它体系结构。在抽象的而仍然明确的意义上，要实现相同功能的组件的任何布局是有效“关联的”使得所希望的功能得以实现。因此，在不考虑体系结构或中间的组件的情况下，在此结合以实现特别功能的任何两个组件都能够被看作是彼此“关联的”使得所希望的功能得以实现。类似地，这样关联的任何两个组件还能够被看作是彼此“可操控地连接的”或者“可操控地耦连的”以实现所希望的功能。

[0089] 还例如，在一种实施方案中，所说明的系统 10 的元件是位于单个集成电路上或者位于相同的器件之内的电路。作为选择，系统 10 可以包括彼此互连的任何数量的分离的集成电路或分离的器件。还例如，系统 10 或其部分可以是物理电路的或可转换成物理电路的逻辑表示的软表示或代码表示。这样，系统 10 可以用任何适合类型的硬件描述语言来实现。

[0090] 而且，本领域技术人员会认识到在上述操作的功能之间的边界只是说明性的。多个操作的功能可以结合到单个操作中，和 / 或单个操作的功能可以分布于附加的操作中。而且，可替换的实施方案可以包括具体操作的多种实例，并且在不同的其它实施方案中可以改变操作的顺序。

[0091] 在此所描述的所有或某些软件可以是数据处理系统 10 从例如计算机可读媒体（例如存储器 18 或其它计算机系统上的其它媒体）所接收的元素。该计算机可读媒体可以是与信息处理系统（例如数据处理系统 10）固定地、可移动地或远程地耦连的。计算机可读媒体可以包括，例如而不限于，任何数量的下列媒体 ：磁存储媒体，包括磁盘和磁带存储媒体 ；光存储媒体，例如压缩磁盘媒体（如 CD-ROM、CD-R 等）和数字视频磁盘存储媒体 ；非易失性存储器存储媒体，包括基于半导体的存储器单元，例如 FLASH 存储器、EEPROM、EPROM、ROM ；铁磁数字存储器 ；MRAM ；易失性存储媒体，包括寄存器、缓冲器或高速缓存器、主存储器、RAM 等 ；以及数据传输媒体，包括计算机网络、点对点远程通信设备、及载波传输媒体，仅列举几个。

[0092] 尽管本发明在此参考具体的实施方案来描述，但是在没有脱离权利要求书所阐明的本发明的范围的情况下能够进行各种修改和改变。因此，说明书和附图应当被看作是说明性的而非限制性的，并且所有这样的修改都意欲要包括于本发明的范围之内。在此针对具体的实施方案所描述的任何利益、优点、或问题的解决方案并不是要理解为任何或所有的权利要求的关键的、必需的、或本质的特征或元素。

[0093] 在此所使用的术语“耦连的”并不是要限定于直接的耦连或机械上的耦连。

[0094] 而且，在此所使用的词语“a”或“an”被定义为一个或多于一个。此外，引入性短语（例如“至少一个”和“一个或更多个”）在权利要求中的使用不应当被解释为暗示着由

不定冠词“a”或“an”引入另一权利要求元素将包含该引入的权利要求元素的任何特别的权利要求限定于只包含一个该元素的发明，即使在相同的权利要求包括引入性短语“一个或更多个”或“至少一个”及不定冠词（例如“a”或“an”）的时候。这对定冠词的使用也有效。

[0095] 除非另有说明，否则词语例如“第一”和“第二”被用来任意地区分这样的词语所描述的元素。因而，这些词语并非必须要指示这样的元素的时间先后或其它优先序。

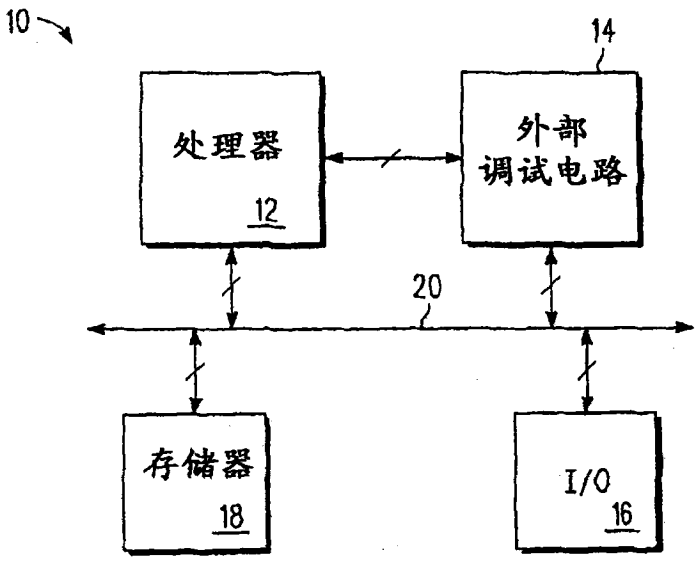

10
14
处理器
12
外部
调试电路
20
存储器
18
I/O
16

图 1

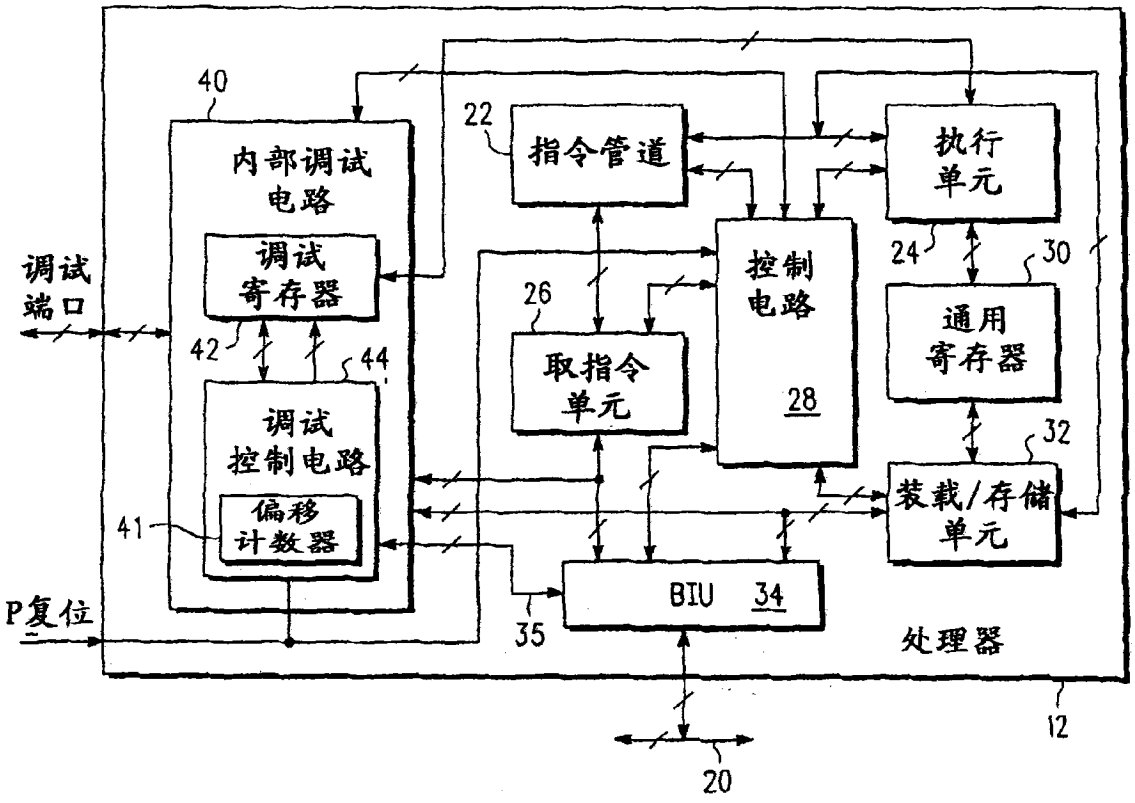

40
内部调试
电路
调试
寄存器
42
44
调试
控制电路
偏移
计数器
41
调试
端口
P复位
22
指令管道
26
取指令
单元
控制
电路
28
执行
单元
24
30
通用
寄存器
32
装载/存储
单元
BIU 34
35
处理器
12
20

图 2

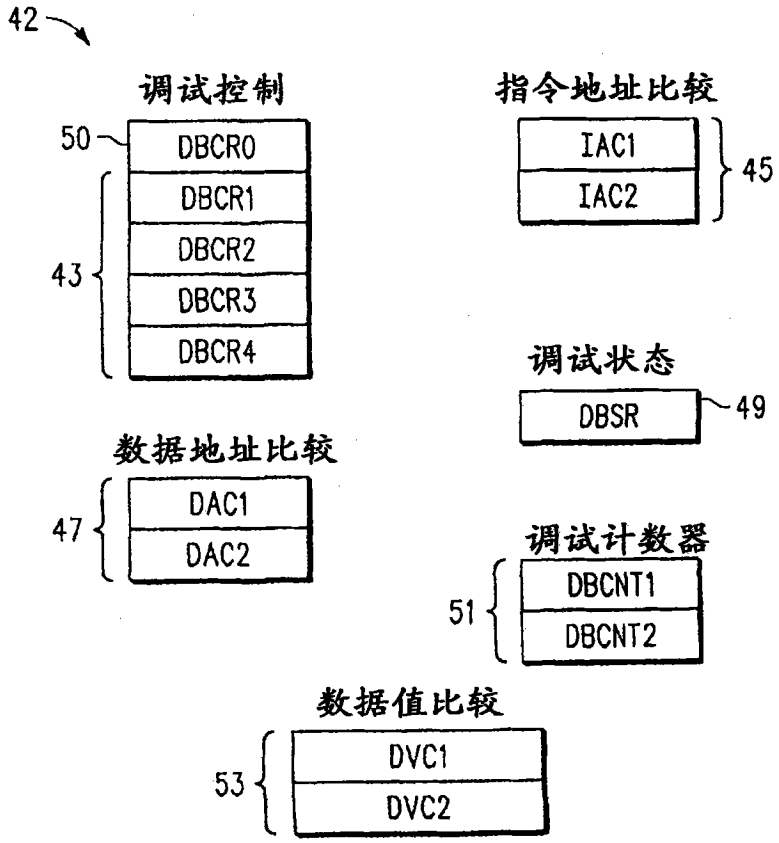

42
调试控制
50
DBCR0
DBCR1
DBCR2
DBCR3
DBCR4
43
指令地址比较
IAC1
IAC2
45
调试状态
DBSR
49
数据地址比较
47
DAC1
DAC2
调试计数器
51
DBCNT1
DBCNT2
数据值比较
53
DVC1
DVC2

图 3

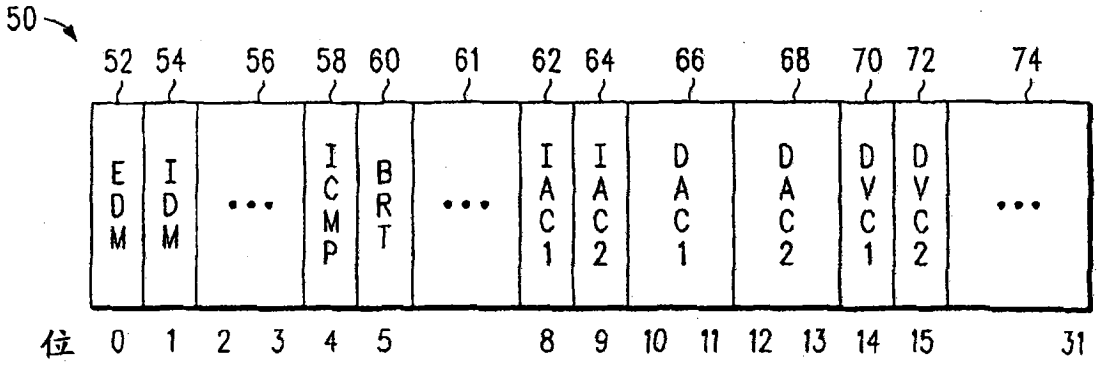

50
52 54 56 58 60 61 62 64 66 68 70 72 74
EDM
IDM
ICMP
BRT
IAC1
IAC2
DAC1
DAC2
DVC1
DVC2
位 0 1 2 3 4 5 8 9 10 11 12 13 14 15 31

图 4

位	名称	描述	
0	EDM	外部调试模式。该位是由软件只读的。 0-外部调试模式禁用。内部调试事件不被映射为外部调试事件。 1-外部调试模式启用。事件将不会引起 CPU 导向中断代码。不允许软件写调试寄存器{DBCRx、DBSR、DBCNT、IAC1-2、DAC1-2、DVC1-2}。	52
1	IDM	内部调试模式 0-调试异常被禁用。除非 EDM 被置位，否则调试事件不会影响 DBSR。 1-调试异常被启用。所启用的调试事件将更新 DBSR 并可以引起 CPU 导向中断代码。允许软件写调试寄存器。	54
2:3	-	保留	56
4	ICMP	指令完成调试事件启用 0-ICMP 调试事件被禁用 1-ICMP 调试事件被启用	58
5	BRT	分支获得调试事件启用 0-BRT 调试事件被禁用 1-BRT 调试事件被启用	60
6:7	-	保留	61

图 5

位	名称	描述	
8	IAC1	指令地址比较 1 调试事件启用 0-IAC1 调试事件被禁用 1-IAC1 调试事件被启用	62
9	IAC2	指令地址比较 2 调试事件启用 0-IAC2 调试事件被禁用 1-IAC2 调试事件被启用	64
10:11	DAC1	数据地址比较 1 调试事件启用 00-DAC1 调试事件被禁用 01-DAC1 调试事件只对存储类型的数据存储存取启用 10-DAC1 调试事件只对装载类型的数据存储存取启用 11-DAC1 调试事件对装载类型或存储类型的数据存储存取启用	66
12:13	DAC2	数据地址比较 2 调试事件启用 00-DAC2 调试事件被禁用 01-DAC2 调试事件只对存储类型的数据存储存取启用 10-DAC2 调试事件只对装载类型的数据存储存取启用 11-DAC2 调试事件对装载类型或存储类型的数据存储存取启用	68
14	DVC1	数据值比较 1 限定器 0-DAC1 调试事件不受数据值比较的影响 1-DAC1 调试事件受数据值比较的限定	70
15	DVC2	数据值比较 2 限定器 0-DAC2 调试事件不受数据值比较的影响 1-DAC2 调试事件受数据值比较的限定	72
16-31	–	保留	73

图 6

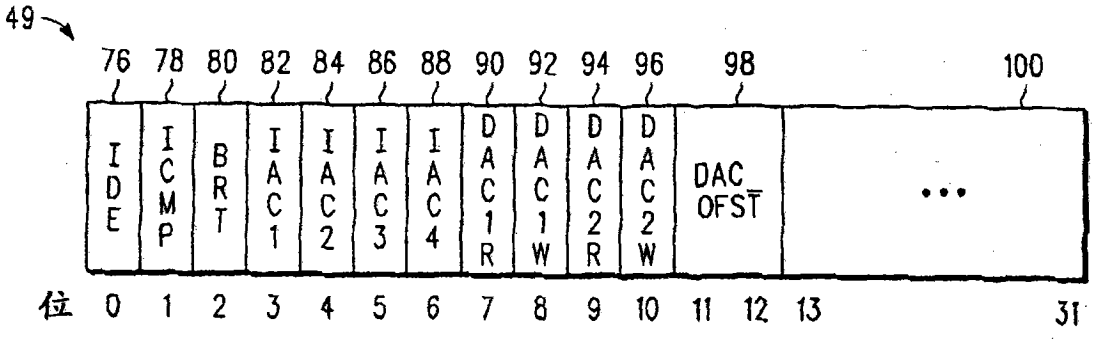


图 7

位	名称	描述	
0	IDE	不准确的调试事件 如果调试异常被禁用并且调试事件引起其各自的调试状态寄存器位被置为 1，则置为 1	76
1	ICMP	指令完成调试事件 如果指令完成调试事件发生，则置为 1	78
2	BRT	分支获得调试事件 如果分支获得调试事件发生，则置为 1	80
3	IAC1	指令地址比较 1 调试事件 如果 IAC1 调试事件发生，则置为 1	82
4	IAC2	指令地址比较 2 调试事件 如果 IAC2 调试事件发生，则置为 1	84
5	IAC3	指令地址比较 3 调试事件 如果 IAC3 调试事件发生，则置为 1	86
6	IAC4	指令地址比较 4 调试事件 如果 IAC4 调试事件发生，则置为 1	88
7	DAC1R	数据地址比较 1 读取调试事件 如果读取类型 DAC1 调试事件发生而 DBCR0[DAC1]=0b10 或 DBCR0[DAC1]=0b11，则置为 1	90
8	DAC1W	数据地址比较 1 写入调试事件 如果写入类型 DAC1 调试事件发生而 DBCR0[DAC1]=0b01 或 DBCR0[DAC1]=0b11，则置为 1	92
9	DAC2R	数据地址比较 2 读取调试事件 如果读取类型 DAC2 调试事件发生而 DBCR0[DAC2]=0b10 或 DBCR0[DAC2]=0b11，则置为 1	94
10	DAC2W	数据地址比较 2 写入调试事件 如果写入类型 DAC2 调试事件发生而 DBCR0[DAC2]=0b01 或 DBCR0[DAC2]=0b11，则置为 1	96
11:12	DAC OFST	数据地址比较偏移 指示所保存的来自获得 DAC 调试异常的装载或存储指令的地址的 DSRR0 值的偏移-1，除非同时的 DTLB 或 DSI 错误发生，在这种情形中该字段被置为 0b00 并且 DBSR[IDE]被置为 1。通常置为 0b00。DVC DAC 将设置该字段为 0b01、0b10 或 0b11。	98
13-31	-	保留	100

图 8

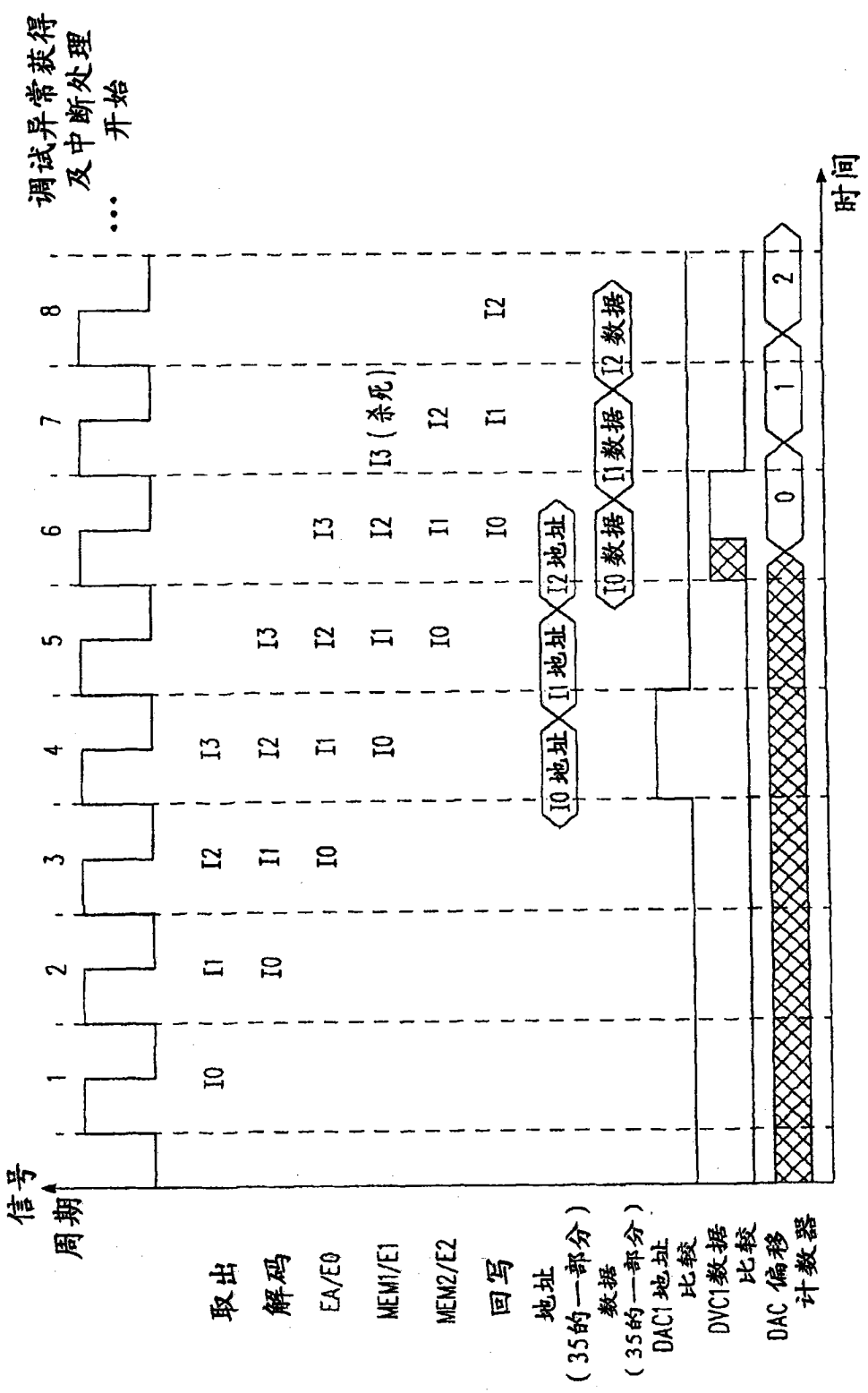

调试异常获得
及中断处理
…… 开始
信号
周期
1
2
3
4
5
6
7
8
时间
取出
解码
EA/E0
MEM1/E1
MEM2/E2
回写
地址
(35的一部分)
数据
(35的一部分)
DAC1地址
比较
DVC1数据
比较
DAC 偏移
计数器
I0
I1
I2
I3
I3 (杀死)
I0 地址
I1 地址
I2 地址
I0 数据
I1数据
I2 数据
0
1
2

图 9

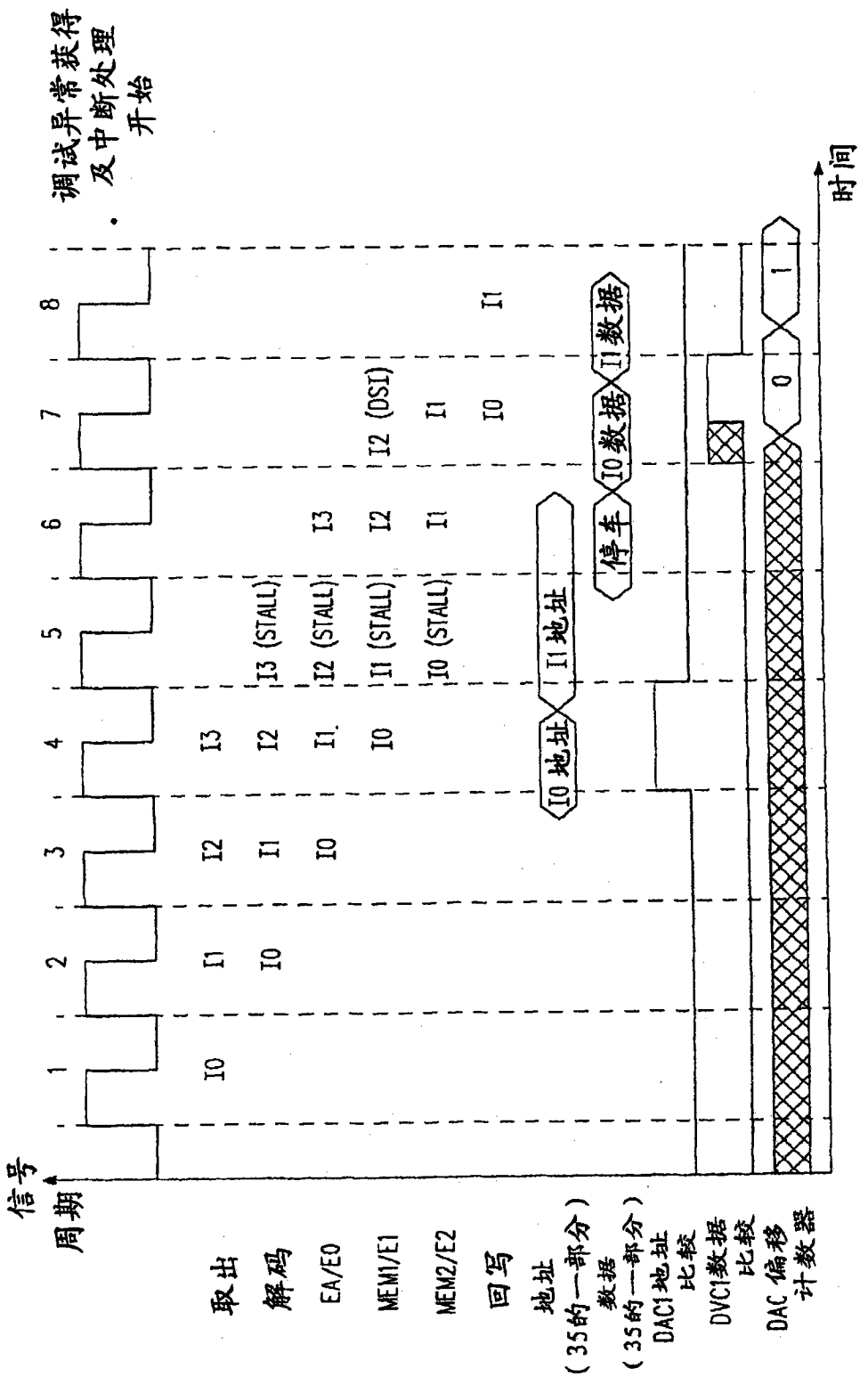

调试异常获得
及中断处理
开始
信号
周期
1
2
3
4
5
6
7
8
时间
取出
解码
EA/E0
MEM1/E1
MEM2/E2
回写
地址
（35的一部分）
数据
（35的一部分）
DAC1地址
比较
DVC1数据
比较
DAC 偏移
计数器
I0
I1
I2
I3
I3 (STALL)
I2 (STALL)
I1 (STALL)
I0 (STALL)
I2 (DSI)
I0 地址
I1 地址
停车
I0 数据
I1 数据
0
1

图 10

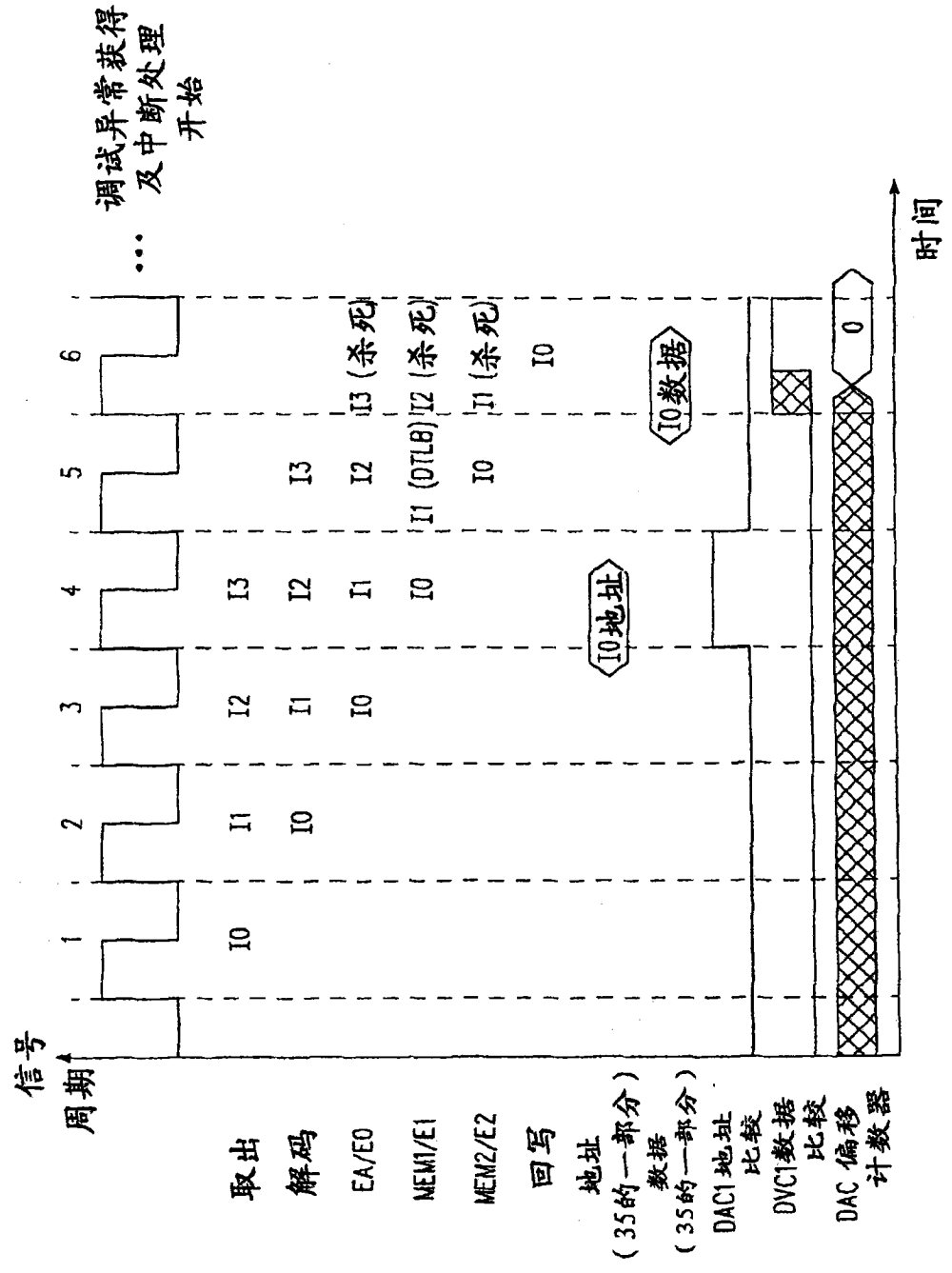

信号
周期
1
2
3
4
5
6
...
调试异常获得
及中断处理
开始
取出
解码
EA/E0
MEM1/E1
MEM2/E2
回写
地址
(35的一部分)
数据
(35的一部分)
DAC1地址
比较
DVC1数据
比较
DAC偏移
计数器
I0
I1
I2
I3
I0地址
I0数据
I1 (DTLB)
I2 (杀死)
I3 (杀死)
I1 (杀死)
I2 (杀死)
I3 (杀死)
0
时间

图 11

	第一装载/存储类指令（I0）	第二类指令（装载/存储类，除非另有说明）（I1）	第三类指令（装载/存储类，除非另有说明）（I2）	结果
201	DTLB 错误，无 DAC	–	–	获得 DTLB 异常，无 DBSR 更新
202	DSI，无 DAC	–	–	获得 DSI 异常，无 DBSR 更新
203	DTLB 错误，有 DACx	–	–	获得调试异常，DBSR 更新设置 DACx 及 IDE，DAC_OFST 置为 0b00。DSRR0 指向第一装载/存储类指令。
204	DSI，有 DACx	–	–	获得调试异常，DBSR 更新设置 DACx 及 IDE，DAC_OFST 置为 0b00。DSRR0 指向第一装载/存储类指令。
205	DACx	–	–	获得调试异常，DBSR 更新设置 DACx，DAC_OFST 置为 0b00。DSRR0 指向第二装载/存储类指令。
206	DVCx DACx	无异常，任何指令	无异常，非 LD/ST 指令	获得调试异常，DBSR 更新设置 DACx，DAC_OFST 置为 0b01。DSRR0 指向第三类指令。
207	DVCx DACx（图 9）	无异常	无异常，LD/ST 指令	获得调试异常，DBSR 更新设置 DACx，DAC_OFST 置为 0b10。DSRR0 指向第三类指令之后的指令。
208	DVCx DACx（图 11）	DTLB 错误，无 DAC	–	获得调试异常，DBSR 更新设置 DACx，DAC_OFST 置为 0b00。DSRR0 指向第二装载/存储类指令。 注意：在这种情形中第二 LD/ST 异常被屏蔽。这种行为随实现而定并且在其它处理器上可以是不同的。

图 12

	第一装载/存储类指令（I0）	第二类指令（装载/存储类，除非另有说明）（I1）	第三类指令（装载/存储类，除非另有说明）（I2）	结果
209	DVCx DACx	DSI，无 DAC	-	获得调试异常，DBSR 更新设置 DACx，DAC_OFST 置为 0b00。DSRR0 指向第二装载/存储类指令。 注意：在这种情形中第二 LD/ST 异常被屏蔽。这种行为随实现而定并且在其它处理器上可以是不同的。
210	DVCx DACx	DTLB 错误，有 DACy	-	获得调试异常，DBSR 更新设置 DACx，DAC_OFST 置为 0b00。DSRR0 指向第二装载/存储类指令。 注意：在这种情形中第二 LD/ST 异常被屏蔽。这种行为随实现而定并且在其它处理器上可以是不同的。
211	DVCx DACx	DSI，有 DACy	-	获得调试异常，DBSR 更新设置 DACx，DAC_OFST 置为 0b00。DSRR0 指向第二装载/存储类指令。 注意：在这种情形中第二 LD/ST 异常被屏蔽。这种行为随实现而定并且在其它处理器上可以是不同的。
212	DVCx DACx	DACy	-	获得调试异常，DBSR 更新设置 DACx，DACy。DAC_OFST 置为 0b01。DSRR0 指向第三类指令。 注意：在这种情形中，若 X==Y，则 DBSR 与 DSRR0 的结果状态可能无法区别于“无 DACy”情形。

图 13

	第一装载/存储类指令（I0）	第二类指令（装载/存储类，除非另有说明）（I1）	第三类指令（装载/存储类，除非另有说明）（I2）	结果
213	DVCx DACx	DVCy DACy，正常的 LD/ST	非-LD/ST 指令	获得调试异常，DBSR 更新设置 DACx、DACy。DAC_OFST 置为 0b01。DSRR0 指向第三类指令。 注意：在这种情形中，若 X==Y，则 DBSR 与 DSRR0 的结果状态可能无法区别于“无 DACy”情形。
214	DVCx DACx	DVCy DACy，正常的 LD/ST	LD/ST 指令无异常	获得调试异常，DBSR 更新设置 DACx、DACy。DAC_OFST 置为 0b10。DSRR0 指向第三装载/存储类指令。 注意：在这种情形中，若 X==Y，则 DBSR 与 DSRR0 的结果状态可能无法区别于“无 DACy”情形。
215	DVCx DACx	DVCy DACy，正常的 LD/ST	DSI，有或无 DAC	获得调试异常，DBSR 更新设置 DACx、DACy。DAC_OFST 置为 0b01。DSRR0 指向第三类指令。 注意：在这种情形中，若 X==Y，则 DBSR 与 DSRR0 的结果状态可能无法区别于“无 DACy”情形。 注意：在这种情形中第三 LD/ST 异常被屏蔽。这种行为随实现而定并且在其它处理器上可以是不同的。

图 14

	第一装载/存储类指令（I0）	第二类指令（装载/存储类，除非另有说明）（I1）	第三类指令（装载/存储类，除非另有说明）（I2）	结果
216	DVCx DACx	DVCy DACy，正常的 LD/ST	DTLB，有或无 DAC	获得调试异常，DBSR 更新设置 DACx、DACy。DAC_OFST 置为 0b01。DSRR0 指向第三类指令。 注意：在这种情形中，若 X==Y，则 DBSR 与 DSRR0 的结果状态可能无法区别于“无 DACy”情形。 注意：在这种情形中第三 LD/ST 异常被屏蔽。这种行为随实现而定并且在其它处理器上可以是不同的。
217	DVCx DACx	DVCy DACy，正常的 LD/ST	DACy，或 DVCy DACy，正常 LD/ST 或多字 LD/ST	获得调试异常，DBSR 更新设置 DACx、DACy。DAC_OFST 置为 0b10。DSRR0 指向第三装载/存储类指令之后的指令。 注意：在这种情形中，若 X==Y，则 DBSR 与 DSRR0 的结果状态可能无法区别于“无 DACy”情形。
218	DVCx DACx	DVCy DACy，LD/ST 多 (LMW，STMW)	包括 LD/ST 的任何指令	获得调试异常，DBSR 更新设置 DACx、DACy。DAC_OFST 置为 0b01。DSRR0 指向第三类指令。 注意：在这种情形中，若 X==Y，则 DBSR 与 DSRR0 的结果状态可能无法区别于“无 DACy”情形。

图 15

	第一装载/存储类指令（I0）	第二类指令（装载/存储类，除非另有说明）（I1）	第三类指令（装载/存储类，除非另有说明）（I2）	结果
219	DVCx DACx (FIG. 10)	任何指令（无异常）	DSI，有或无DAC，正常的LD/ST或多字LD/ST	获得调试异常，DBSR更新设置DACx、DACy。DAC_OFST置为0b01。DSRR0指向第三类指令。 注意：在这种情形中第三LD/ST异常被屏蔽。这种行为随实现而定并且在其它处理器上可以是不同的。
220	DVCx DACx	任何指令（无异常）	DACy，或DVCy DACy正常的LD/ST或多字LD/ST	获得调试异常，DBSR更新设置DACx、DACy。DAC_OFST置为0b10。DSRR0指向第三类指令之后的指令。 注意：在这种情形中，若X==Y，则DBSR与DSRR0的结果状态可能无法区别于“无DACy”情形。

图 16

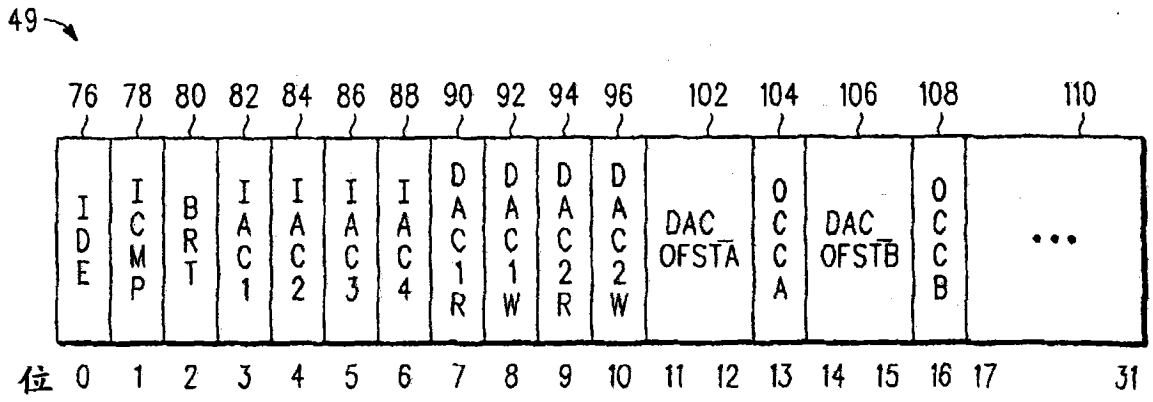


图 17

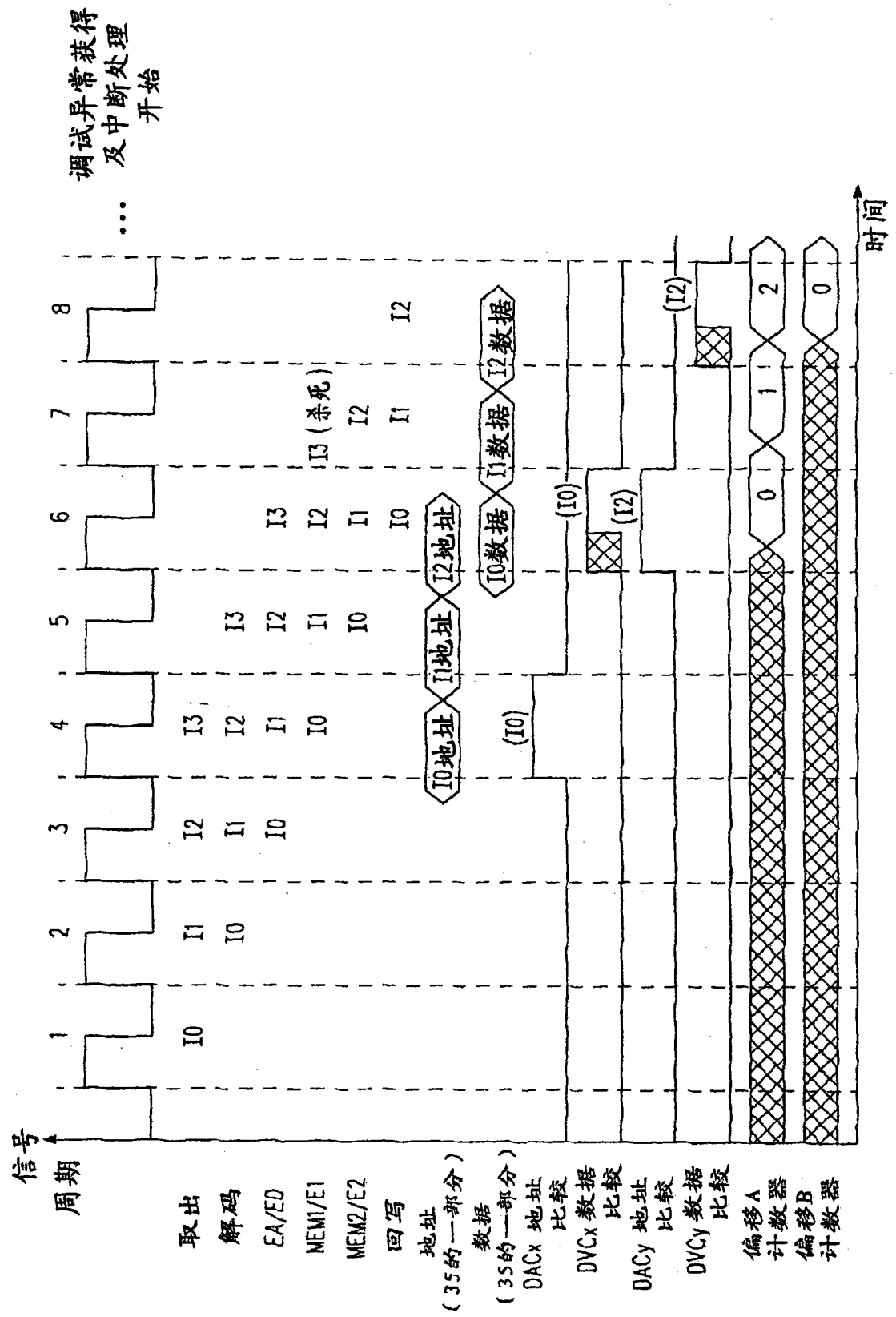

调试异常获得及中断处理开始
信号
周期
取出
解码
EA/ED
MEM1/E1
MEM2/E2
回写
地址（35的一部分）
数据（35的一部分）
DACx 地址比较
DVCx 数据比较
DACy 地址比较
DVCy 数据比较
偏移A计数器
偏移B计数器
I0地址
I1地址
I2地址
I0数据
I1数据
I2数据
I3（杀死）
时间

图 18