[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 13/42

[12] 发明专利说明书

[21] ZL 专利号 99816463.1

[45] 授权公告日 2003 年 9 月 3 日

[11] 授权公告号 CN 1120430C

[22] 申请日 1999.12.9 [21] 申请号 99816463.1

[30] 优先权

[32] 1999. 3.10 [33] US [31] 09/266,045

[86] 国际申请 PCT/US99/29321 1999.12.9

[87] 国际公布 WO00/54165 英 2000.9.14

[85] 进入国家阶段日期 2001.9.10

[71] 专利权人 爱特梅尔股份有限公司

地址 美国加利福尼亚州

[72] 发明人 M·奥勒斯

审查员 齐 霁

[74] 专利代理机构 上海专利商标事务所

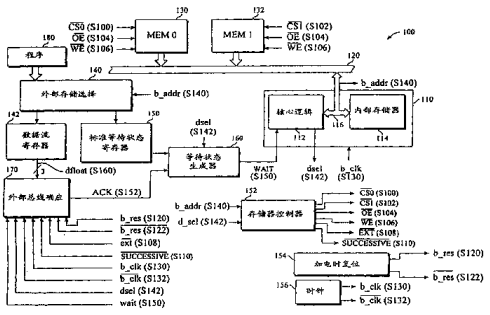
代理人 沈昭坤

权利要求书 4 页 说明书 11 页 附图 7 页

[54] 发明名称 具有可编程等待状态的微处理设备

[57] 摘要

本发明是涉及用微控制器(110)来控制数据总线(120)，考虑到了存储器输出驱动器在输出操作后需要有限的时间来电释放总线的事实。 每一存储器(130、132)有与之相应的等待状态数来有选择地置微处理器(110)在紧接读操作之后并在下一个 I/O 操作时之前进入可变长度的等待状态。



ISSN1008-4274

1、在具有第一存储器（130）和第二存储器（132）、等待状态电路（160）、逻辑装置（110、152）和数据装置（140、142）与公共数据总线（120）连接的计算设备中，一种访问第一和第二存储器的方法，其特征在于，包括以下几步：

通过在数据装置（140）接收到对应于第一存储器的地址时，选择与第一存储器相关的所述状态值，从而将第一等待状态值分配给第一存储器（130）；

通过在数据装置（140）接收到对应于第二存储器的地址时，选择与第二存储器相关的所述状态值，从而将第二等待状态值分配给第二存储器（132）；

依据所接收到的地址，从所述存储器中选择一个存储器；

在所选的存储器上执行第一 I/O 操作；

做出是否置逻辑装置（110）进入等待状态的决定，所述决定是基于在存储器上所执行的 I/O 操作序列；

当所做的决定是肯定时，置逻辑装置（110）为等待状态，其等待时间周期与所选存储器相关的等待状态值相对应，足够对所述第一和第二存储器进行访问；

当所做的决定是否定时，避免置逻辑装置（110）为等待状态；并且

对第一和第二存储器的另一个执行下一个 I/O 操作。

2、如权利要求 1 所述的方法，其特征在于，在所选存储器上的写入步骤先于在所选的存储器上执行第一 I/O 操作步骤执行，而后紧接着没有附加等待状态的读取步骤。

3、如权利要求 1 所述的方法，其特征在于，在所选的存储器上执行第一 I/O 操作步骤包括连续从那里读取多个数据，所述的连续读数据之间没有附加的等待状态。

4、如权利要求 1 所述的方法，其特征在于，置所述逻辑装置（110）为等待状态的步骤包括选取与所选存储器相应的等待状态值，并且对一个数进行计数，

直到等于该值。

5、如权利要求 1 所述的方法，其特征在于，所述第一和第二存储器是外部存储器。

6、如权利要求 1 所述的方法，其特征在于，所述第一和第二存储器是内部存储器。

7、如权利要求 1 所述的方法，其特征在于，所述第一和第二等待状态值是基于所述第一和第二存储器各自的输出电路的特性。

8、如权利要求 7 所述的方法，其特征在于，所述第一和第二等待状态值是进一步基于所述计算设备使用的时钟频率。

9、如权利要求 7 所述的方法，其特征在于，所述特性是指输出电路在读操作后，达到高阻抗状态所需的时间。

10、如权利要求 1 所述的方法，其特征在于，所述的第一和第二存储器是外部存储器，并且所述计算设备进一步具有第三内部存储器（114），所述方法包括访问所述第三存储器的步骤，所述步骤包括：

当在所述第一和第二存储器中所选的一存储器上执行的第一 I/O 操作步骤完成后，立刻对所述第三存储器继续进行下一个没有附加等待状态的 I/O 操作。

11、如权利要求 10 所述的方法，其特征在于，如果一个对所述内部 I/O 操作之后的外部读操作是针对所述第一和第二存储器中所选的一存储器，那么在所述内部 I/O 操作之后立即对所选存储器进行没有附加等待状态的所述外部读操作。

12、如权利要求 10 所述的方法，其特征在于，如果一个对所述内部 I/O 操作之后的外部读操作是针对所述存储器中所选的其他存储器，那么当完成所述的内部 I/O 操作，置逻辑装置（110）为等待状态，其等待时间周期与所选存储器相应

的等待状态值相对应。

13、一种微处理设备，其特征在于，包括：

系统总线（120）；

连接在系统总线上的逻辑电路（110、152），该逻辑电路至少具有第一内部存储器（114a）和第一信号（s108）的第一输出，第一信号在对外部存储器进行读操作期间具有第一逻辑状态，否则具有第二逻辑状态；至少一个第一外部存储器（130）与系统总线相连接；

等待状态电路（160）具有和所述逻辑电路（110、152）相连接的第二信号（s150）的第二输出，由此所述逻辑电路响应于第二信号被加电，而进入等待状态；

外部总线控制电路（170）含有计数电路（202、204、206），比较电路（220），和与等待状态电路（160）相连的第三信号（s152）的第三输出，其特征在于，等待状态电路响应于第三信号的被加电和不加电，对第二信号加电和不加电；并且

数据装置（140、142）用来存储多个等待状态值，其中的一个所述等待状态值对应于所述第一外部存储器（130）；外部总线控制电路（170）作为第一信号（s108）状态的一个函数对第三信号（s152）加电；

计数电路（202、204、206）响应于第一信号从第一逻辑状态转换到第二逻辑状态的改变开始一计数序列；

比较电路（220）具有与数据装置（140、142）相连的第一输入，来接收所选的等待状态数值之一，并且连接第二输入用来接收从计数器输出的计数；当比较电路的输入相等时，比较电路就不加电第三信号；逻辑电路（110，152）包括了第四信号（s110）的第四输出，表明了对同一存储器的连续读操作，第四输出和外部总线控制电路（170）相连，外部总线控制电路进一步作为第四信号的状态的一个函数对第三信号加电。

14、如权利要求 13 所述的微处理设备，其特征在于，所述逻辑电路（110，152）生成存储地址（s140），并且数据装置(140、142)包括了用于产生响应于所生成的存储器地址而产生等待状态值之一的选择器（140）。

15、如权利要求 13 所述的微处理设备，其特征在于，所述微处理设备进一步包括用于将等待状态数有选择地载入到数据装置（140、142）中的编程装置（180）。

具有可编程等待状态的微处理设备

(1)技术领域

本发明涉及到微处理设备，特别涉及与外部设备进行 I/O 的可编程等待状态方案。

(2)背景技术

通常，每当数据在同系统的两台设备（例如：微处理单元和存储器件）之间交换时，系统必须确认两台设备之间的交互必须是同步的。例如，最近，在微处理器和微控制器的制造中使用了允许高速运作的技术。时钟速度达到 300MHz 或者更高成为很普通的事。然而外部设备，特别是存储器件象闪存（FLASH）、静态存取存储器（SRAM）以及电可擦除只读存储器(EEPROM)，通常不具备这么高的时钟速度。

一种与慢速设备同步的通常方法是通过使用等待状态改变微处理器件对每一设备访问的总线周期持续时间，有时称为“总线伸展”。系统地址译码器对地址译码，并且决定哪块存储器被选择。系统控制逻辑接着决定该存储器是否是需要一个等待状态，并且如果需要就对一根等待信号线加电。微处理核心在一个时钟周期中对信号线采样，并且如果该线被加电，就进入到一个时钟周期的等待状态（本质上就是不做任何事的状态）。微处理核心在下一个时钟周期再对信号线采样，并且当等待信号线被取消（不加电）时，继续运行。这样就提供给慢速的存储器器件足够的内部时间来对地址译码，访问存储器，以及使得在其输出缓冲里的数据能保持稳定。

一种使用等待状态的已有技术包括在自身的存储器件中存入等待状态数值。计算器件包括的逻辑电路访问这个数值来决定总线周期需要扩充的时间。这个方法需要含有存储等待状态数值的逻辑电路和将数值提供给计算器件的特殊器件。

另一个使用等待状态的方法被应用在中断驱动的 I/O 设备中。在程序解码器的逻辑部分侦测某些 I/O 指令的发生，并且依据指令声明等待信号将微处理器操作挂起一周期时间。这种方法对存取低速 I/O 设备很有用，微处理器在中断指令返回期间暂停，给了低速 I/O 设备执行操作的时间。

一个上面没有提及的问题是存取多存储器设备，一个读访问第一设备的操作在开始访问另一设备操作之前没有给予足够的时间来完成，导致总线冲突。这个问题由来是因为当访问另一设备操作发生时，第一设备的输出驱动还在驱动这两个设备公共的数据总线。除非在第二设备执行访问之前有足够的时间经过，否则两个设备都会试图驱动总线。其结果是数据混淆以及增加了由于驱动电路同时访问带来的能量消耗。然而，这样的问题自身并不是经常出现。例如，从同一设备中连续发出的读操作不会引起所述驱动电路的问题。一个写操作和跟随的读操作，同样地，也不会引起所述驱动电路的问题。

欧洲专利申请，EP-A-0 386 935 的 Canon Kabushiki Kaisha 揭示了能够根据访问时间改变访问存储器的等待状态数的信息处理设备。可相互交换的外部存储器存储访问等待状态的数据或值，并且该设备读取数据，从而控制等待状态值。

NEC 公司的欧洲专利申请，EP-A-0 713 170 揭示了一种数据处理设备，该设备有一 CPU 部分和一 BCU 部分，其中时钟控制部分提供的时钟分配给 CPU 部分和 BCU 部分的内部。在来自 CPU 部分对存储设备的读访问请求期间，提供给 CPU 部分的 CPU 时钟的周期延长了。在当 CPU 部分通过内部数据总线输入输入数据时的 CPU 时钟切换点之前，状态马上被延长，直到在内部数据总线上的读数据完成。

我们所需要的是在微计算结构中加入控制电路，来侦测对多存储的访问以及根据 I/O 所执行操作的类型插入等待状态。我们期望能有这样的电路，可以侦测哪个存储器被访问以及适时插入等待状态。我们更期待有这样的电路能依照执行的 I/O 操作序列来插入等待状态。

(3)发明内容

在具有第一和第二外部存储器的计算设备中，一种访问存储器的方法包括将第一等待状态值与第一存储器关联，并且将第二等待状态值与第二存储器关联，接着选择与被选择的存储器之一相关的等待状态值，并且读该存储器。并且在继续读步骤和下一步 I/O 操作之前，CPU 被置成与所选的等待状态值成比例的周期时间的等待状态；例如等待状态 x 时钟周期。

然而，如果下一步 I/O 操作是针对 CPU 内部存储器，那么 CPU 在 I/O 操作之前不进入等待状态，而是立刻处理该 I/O 操作。如果接着内部 I/O 操作之后的读

操作是针对于最初选定的外部存储器，那么不进入等待状态而访问立刻进行。如果接着内部 I/O 操作之后的读操作是针对于不同的外部存储器，那么将进入的等待状态时间周期是与最初选定的外部存储器关联的等待状态成比例，从完成读操作的计时开始。

根据本发明，一种微处理设备包括连接外部存储器、核心逻辑和存储控制器的系统总线。等待状态电路生成一使核心逻辑进入等待状态的等待信号。连接等待状态电路的是外部总线电路，用来侦测在系统总线上发生的存储访问序列以及激活等待状态电路，并因而使核心逻辑进入等待状态。连接外部总线电路的是用来存储每一存储器件数据流等待状态值的数据装置。

数据装置对选中的存储器件响应而输出数据流等待状态值给外部总线电路。外部总线电路侦测出外部读操作的发生，该操作后面跟着访问另一个外部存储器的操作，并且初始化数据流等待状态周期，该周期与被访问的存储器关联的等待状态值成比例，例如等待状态为 x 时钟周期。然而，如果外部总线电路侦测到一写操作，后面跟着一读操作，那么没有数据流等待状态发生，虽然一个标准的等待状态可能仍然发生。类似的，如果一外部读操作，后面跟着一内部访问，那么没有数据流等待状态发生。

(4)附图说明

图 1 是本发明数据处理系统的方框图。

图 2 是如图 1 所示的外部总线确认逻辑方框图。

图 3-6 是说明根据本发明存储访问方案采用的多种数据处理系统操作配置时序图。

图 7 是数据处理系统的替代实施例方框图。

(5)具体实施方式

参照图 1，本发明的数据处理系统 100 包括多种安排在系统总线 120 周围的各种子系统。微处理单元 110 例如微控制器或类似的，包括核心逻辑 112 和内部存储器 114。内部时钟 156 提供了时钟信号 s130 给驱动器微处理单元 110。在核心逻辑 112 和存储器 114 之间的数据传输发生在内部总线 116 上。

内部总线和系统总线 120 相连，来提供数据向微处理单元 110 外部的传输。离开总线 116 的是输送给存储控制器 152 的存储器地址线 s140。核心逻辑 112 提供了数据选择信号 s142，表明了核心需要数据传输（读或写）。特别地，信号 s142 HI 表明了一数据传输周期，而信号 s142 LO 表明了一地址周期。数据选择信号输送给了存储控制器 152 以及等待状态生成器 160。

连接系统总线 120 的是外部存储器 130 和 132。典型的外部存储器包括 EEPROM、FLASH 和 SRAM。存储器 130 和 132 被典型的控制信号控制，例如片选信号 s100，存储器 132 的输出允许信号 s104 和写允许信号 s106。这些信号依据 I/O 操作和出现在地址线 s140 上的地址由存储控制器 152 产生。存储控制器也提供外部读信号 s108，表明了从存储器到微处理单元 110 的读周期。存储控制器也提供连续的读信号 s110，表明了发生在同一先前访问的存储器上的读操作。

等待状态生成器 160 提供了输送给核心逻辑 112 的等待信号 s150（激活 HI）。对信号 s150 加电引起核心逻辑进入等待状态，而不对信号 s150 加电则允许核心逻辑继续工作。等待状态生成器 160 从标准的等待状态寄存器接收等待状态数值。等待状态生成器 160 也接收数据选择信号 s142，表明了数据传输将开始。根据传统的操作，当外部存储器被访问时，等待信号 s150 就被生成，以便让微处理单元 110 进入等待状态，从而使外部存储器位置有足够的时间周期被访问。这种时间周期是基于标准等待状态寄存器 150 生成值。

根据本发明，有用于存储不同存储器类型的等待状态数值的外部存储选择逻辑 140。通常，每一存储器类型都和一地址范围关联。给定数据的地址表明了存储数据的存储器类型。因此，地址线 s140 输送给外部选择逻辑 140，外部选择逻辑 140 再输出对应于线上地址的存储器件的等待状态数值。可见，每一存储器都有两个相关联的等待状态数值，标准等待状态数值和数据流等待状态数值。标准等待状态数值是用于让 CPU 进入一定时间周期等待状态的已有技术等待状态数值，使得地址数据有足够的时间被取出，并在系统总线 120 上驱动。对应于给定地址的标准等待状态数值被存储在寄存器 150 中，并且从那里输送到等待状态生成器 160。

外部选择逻辑 140 也提供了存储在数据流寄存器 142 中的数据流等待状

态数值。数据流寄存器上的三位数据流 s160 被输送到外部总线逻辑 170 中。另外，外部总线逻辑接收外部读信号 s108，连续读信号 s110，数据选择信号 s142，和从等待状态生成器 160 中出来的等待信号 s150。基于这些输入信号，外部总线逻辑 170 将产生 ACK 信号 s152（激活 HI）来输送给等待状态生成器，从而当 ACK 信号 s152 为 LO 时，使等待信号 s150 加电。

地址范围和等待状态数值来自于用户提供程序码 180。在实施例中，外部存储选择逻辑 140 包括多种可以通过存储器映射方案访问的寄存器，其特征在于，寄存器具有在程序的控制下可访问的相关地址。用户可以通过设定地址范围和对应等待状态数值来对寄存器编程，可以是标准的或数据流的。

继续描述图 1，在重置电路 154 上有电源，来提供重置信号 s120 和它的补码信号 s122。还有提供时钟信号 s130 和其补码 s132。

参照图 2，展示了包括外部总线逻辑 170 的电路。计数逻辑由被时钟信号 s130 计时的触发器 202、204 和 206 组成。计数逻辑输出由单独触发器的反相输出组成。这些输出输送到比较器电路 220，它包括三个和与非门连接的异或门（XOR）。每个异或门的输出被连接到触发器 202-206 的反相输出上，而例外的是触发器 204 的输出是由缓冲 254 驱动，随后输送给比较器电路 220。比较器的另一输入来源于数据流位 s160。比较器 220 的输出输送给了与非门 260 的一个输入。

数据流位 s160 在逻辑电路 222 中一起进行或操作。或操作的结果输送给了与非门 260 的另一个输入。

等待信号 s150 和外部读信号 s108 分别输送给了反相器 267 和 268，其输出通过与非门 266 连接。反相器和与非门一起作为一个或门工作。与非门 266 的输出输送给与门 264 的一个输入。数据选择信号 s142 和与门 264 的另一个输入连接。与门 264 的一输出和与非门 262 的一输入相连接。与非门 262 的另一输入与触发器 208 的输出相连接。

数据选择信号 s142 也通过反相器 265 反相，并且通过或门 263 与连续读信号 s110 相或。或门 263 输出和与非门 262 的输出通过与门 261 相连接，与门 261 的输出输送给与非门 260 的第三输入。与非门 260 依次又输送到产生 ACK 信号 s152 的触发器 208。

回到计数逻辑，触发器 202 通过反相器 251 被与非门 250 驱动。与非门接收触发器 202 的反相输出和触发器 208 的反相输出。继续向前，或非门 253 接收到触发器 202 的反相输出和触发器 208 的非反相输出。或非门 253 的输出输送给了复用器 210 的输入 1。复用器 210 的其他输入连接到时钟信号 s130。复用器的输出输送给了触发器 204。

触发器 204 的反相输出输送(通过缓冲 254)给了与非门 255 和反相器 256，复用器 210 的选择器控制。与非门 255 也接收触发器 208 的反相输出。与非门 250 和 255 的输出连接到了与非门 258 上。反相器 256 连接在与门 259，与门 259 也接收通过反相器 257 从触发器 202 反相输出的输入信号和从触发器 208 反相输出的输入信号。门 258 和 259 的输出分别输送给了复用器 212 的输入 0 和 1。复用器 212 的输出给了触发器 206 的输入，触发器 206 的反相输出作为了复用器 212 的选择输出。

外部总线逻辑 170 的运作将在下面解释。通常，为了外部存储器访问，对应于被访问的存储器的数据流数值从外部选择逻辑 140 中载入，并送给寄存器 142。经过一个读周期，计数逻辑（触发器 202、204 和 206）将开始增加至数据流数值（信号 s160）。在计数期间，ACK 信号 s152 是不加电（LO）的，表明了系统总线 120 仍然被读存储器的输出缓冲驱动。作为回应，等待状态生成器 160 加电等待信号 s150，使得微控制器 110 进入等待状态。一旦计数达到数据流值 s160，计数将复位，并且 ACK 信号 s152 将为 HI，表明系统总线 120 空闲。

外部总线逻辑 170 的不同操作配置可以结合图 3-6 描述的时序图来进行最佳的解释。时序图是如图 2 所示的外部总线逻辑 170 的模拟产物。模拟是通过使用 Cadence 制造的 VeriLog XL 模拟器来实现的。

参照图 1、2 和 3，假设存储器 130（MEMO）是一 8 位存储器，它的地址最高位等于“0x010”。外部选择逻辑 140 被编程为地址从 0x010 开始时，对应数据流等待状态数值为 5。图 3 所示的时序图说明了一个写操作后面跟着两个连续的对存储器 MEMO 的读操作信号。另外，写和读操作涉及到 16 位字，但每个 I/O 操作是由两个 8 位存储访问（低位和高位）组成。

图 3 中时序图的顶部展示了存储地址信号 s140 和时钟信号 s130。片选 s100

（CS0,激活 LO）表明了 MEM0 被选中。第一访问地址是 0x01000500。这地址在地址总线 s140 上被选中，外部选择逻辑 140 输出 5 给数据流寄存器 142。接着，数据流位 s160 被置为 5。

在时间周期 t1，写允许信号 s106 在当高位字节和低位字节被写进存储位置 0x01000500 时,被两次加电（激活 LO）.因为是写操作，数据传输信号 s142 被加电，并且外部读信号 s108 为不加电（激活 LO）。虽然根据本发明实施例的逻辑电路 170、ACK 信号 s152 依然被加电，并且计数输出（触发器 202、204 和 206 的输出）在写周期结束依然是低电平，标志为如图 3 所示的事件 A_3，表明了系统总线 120 空闲。观察在周期 t1 中，等待信号 s150 仍然由等待状态生成器 160 加电。这是标准等待状态的结果，对应于写操作产生。

注意，连续读信号 s110 在事件 A_3处由存储控制器 152 加电，因为一个访问同一存储器（MEM0）的操作被执行，即一个读操作。在时间周期 t2 中，读操作发生于输出允许信号 s104 在事件 B_3处转到 LO 来输出访问字的第一字节。同样在事件 B_3处，外部读信号 s108 变为 LO，表明了一外部读访问发生。第二字节在事件 C_3处读取。再次，在周期 t2 中标准等待状态出现，来提供足够的时间完成读操作。

在事件 D_3,新地址被加电，即 0x01000500。再次，因为访问同一存储器（MEM0），连续读信号 s110 被加电。因此，根据本发明，计数器不开始计数，并且，触发器 208 还是继续输出 HI，因而对 ACK 信号 s152 加电，表明系统总线 120 空闲。

结束讨论图 3 前，在时间周期 t3 期间，两个读访问在事件 E_3和 F_3处发生，来读取存储在存储位置 0x01000500 的构成字的两字节。该操作伴随着在同周期中的标准等待状态，由等待信号 s150 被加电所指示。

从前面讨论的图 3 中，可以了解外部总线逻辑 170 在完成写周期时没有始发一等待状态，因为总线冲突不是个问题。另外，对从同一存储器中来的连续读信号，总线逻辑也不始发等待状态。

参照图 1、2 和 4，假设存储器 130（MEM0）是一 8 位存储器，它的地址最高位等于“0x010”。进一步假设外部选择逻辑 140 被编程为地址从 0x010 开始时，对应数据流等待状态数值为 5。图 4 所示的时序图说明了一个从 MEM0

来的读操作时限，其后面跟着对存储器 MEM0 写操作和从 MEM0 来的读操作。如图 3，写和读操作涉及到 16 位字，因而每个 I/O 操作是由两个 8 位存储访问（低位和高位）组成。

在图 4 中我们可以看出，在时间周期 t1，产生的读操作伴随着相应的标准等待状态。当在事件 A_4 处完成了读操作，输出允许信号 s104 不加电，并且外部读信号 s108 同样不加电，ACK 信号 s152 不加电，引起计数器（触发器 202、204、206）开始计数和使等待状态生成器 160 加电等待状态信号 s150 来表明系统总线 120 正忙着。结果，CPU110 进入等待状态。

与已有技术相比，下一步 I/O 操作将在事件 B_4 处开始，会因为 CPU 试图在 MEM0 输出缓冲仍然占用系统总线 120 进行它的读操作的情况下，同时在系统总线上驱动自身输出数据，而导致总线争夺。而根据本发明，由于在事件 B_4 处进入等待状态替代了原来操作，MEM0 的输出缓冲有足够的时间（5 个时钟周期）来释放系统总线，因而当写操作开始执行时，总线已经空闲，并且也没有在微控制器和 MEM0 之间由于争夺而产生混淆数据的危险。

在事件 C_4 处，当计数器计数到 5，并且比较器 220 侦测到等值的数据流 s160，触发器 208 输出 HI 给再次加电 ACK 信号 s152。作为回应，等待状态生成器 160 应该如虚线所示不加电等待信号 s150。而在随后的写操作期间，标准等待状态被插入到 t2 时间周期。在写周期结束（事件 D_4）时，计数器并没启动，因为 ACK 信号 s152 仍然是 HI，因为外部选择信号被加电，并且因而在下一步 I/O 操作前没有等待状态发生。回顾外部总线逻辑 170 在写操作结束时没有启动计数器。因而，可以看见在从存储位置为 0x010001f8 读取时，标准等待状态在 t3 时间周期仍然产生。

概括一下，图 4 的时序图展示了本发明依据刚完成的操作有选择地加电等待状态。在外部读操作情况下且如果数据流寄存器含有非零值，则产生等待状态以允许输出驱动释放总线。在写操作的情况下，不发出等待状态。

参照图 1、2 和 5，假设存储器 130（MEM0）是一 8 位存储器，它的地址最高位等于“0x010”。进一步假设内部存储器 114 具有最高位为“0x000”的地址。外部选择逻辑 140 被编程为地址从 0x010 开始时，对应数据流等待状态数值为 5。图 5 所示的时序图说明了一个从 MEM0 来的读操作信号，其后

面跟着从内部存储器 114 来的连续三个写操作和从 MEM0 来的另一个读操作。对 MEM0 读操作涉及到 16 位字，但每个 I/O 操作是由两个 8 位存储访问（低位和高位）组成。

图 5 中时序图的顶部展示了存储地址信号 s140 和时钟信号 s130。在 t1 时间周期中，第一读访问是对位置为 0x010001e8 的外部存储器 MEM0 的。注意，每一标准等待状态操作的等待信号 s150 在相同的时间周期被加电。

事件 A_5 处读操作的结果是，ACK 信号 s152 变成 LO，而引发计数器开始计数。回想一下，ACK LO 表明系统总线 120 不空闲。然而可以看见等待信号 s150 仍然未被加电；原因是后来的读操作是对内部存储器 114，该存储器不依靠由 MEM0 输出驱动器释放的系统总线 120。

回到图 1，当一内部存储地址被侦测到，外部存储选择 140 载入一个零数值给标准等待状态寄存器 150。这输送给等待状态生成器 160，生成器维持等待信号 s150 不加电状态以作为对接收的零标准等待状态值的响应，而不管 ACK 信号 s152 的逻辑状态。因为核心逻辑 112 可以在单个时钟周期里访问内部存储器，所以并没有必要产生等待状态。

而在从 0x00000ec8,0x00000ecc 和 0x00000ed0 三个地址开始的内部读操作持续时间中，计数器继续计数。维持这样的活动是因为 MEM0 的输出驱动还需要时间释放系统总线，并且如果微处理单元 110 想访问外部存储器，它必须进入等待状态，直到计数器完成；即达到的计数值和数据流值 s160 相同。否则，总线争夺就会发生。

另一方面，如果同一外部存储器被访问，那么就没有必要进入等待状态。这在图 5 中有精确的情况描述。注意在事件 B_5 处，连续读信号 s110 变为 LO，表明了 MEM0 被再次访问。这引起 ACK 信号 s152 变为 HI，立刻重置计数器（触发器 202、204、206）为 0。在随时间周期 t2 和 t3 后跟着从 MEM0 中地址为 0x010001ea 和 0x01000258 开始的两条读操作以及伴随着对应的标准等待状态。

概括图 5 带来的要点，可以理解当内部存储器访问跟随着从外部存储器的读操作，计数器依然被激活，但 CPU 没有进入等待状态。这将使后来的外部存储器访问成为可能，这时 CPU 被迫等待计数的完成以便系统总线可以被

释放。然而，如果外部访问是对于同一外部存储器，那么计数立刻停止，提供没有等待状态的外部存储器访问。

参照图 1、2 和 6，假设存储器 130（MEM0）是一 8 位存储器，它的地址最高位等于“0x010”，并且存储器 132(MEM1)是另一最高位地址为“0x011”的 8 位存储器。进一步假设外部选择逻辑 140 被编程为针对 MEM0 对应数据流等待状态数值为 1，并且针对 MEM1 数据流等待状态数值为 5。图 6 所示的时序图说明了一个对 MEM1 写操作的定时，其后面跟着从 MEM0 的读操作和从 MEM1 的读操作以及从 MEM0 的 4 个读操作。对 MEM0 读操作涉及到 8 位数据，因此每一 I/O 操作需要单个访问。对 MEM1 的写和读操作涉及 32 位字。但每个 I/O 操作是由 4 个 8 位存储访问组成。

在时间周期 t1，MEM1 被选中，由片选信号 s102 被加电所标识。四个写访问对位置 0x01100040 进行操作。可以理解，被选中的数据流 s160 是 5。还应注意的是在周期 t1’中的等待状态归因于标准等待状态的生成。

当在事件 A_6的写周期完成，MEM0 中的地址 0x01000270 被选中，伴随着对应数据流 s160 改变为 1。当 MEM1 重新被选中地址 0x01100040 时，读操作开始于 B_6,并且在 C_6处完成。计数器继续计数，使得触发器 208 变成 LO，从而依次引起等待状态发生。因为数据流是 1，计数器计数 1，并且触发器 208 在 D_6处变为 HI。在事件 C_6和 D_6之间的这一等待状态允许 MEM0 输出驱动释放系统总线 120，以便后来的在事件 D_6处对 MEM1 的读出可继续进行而没有总线冲突的危险。因而在时间周期 t2 期间的等待状态归因于从读 MEM0 中来的数据流和在读 MEM1 期间到标准等待状态生成的数据流。

当 MEM1 在 H_6处被选中时，可以理解数据流 s160 又一次被设为数据流值为 5。当在 H_6处完成读操作，计数器开始计数，这次计数到 5。ACK 信号 s152 对应于时间周期变成 LO，表明 MEM1 释放了系统总线。一达到计数值 5，ACK 信号 s152 变为 HI，并且对 MEM0 的读操作继续在位置 0x0100274 上进行（事件 F_6）。再次，时间周期 t3 中的等待状态由数据流等待状态和标准等待状态组成。在 G_6处发生对同一 MEM0 的后续读操作，也不需要在每一读操作后有数据流等待状态。而标准等待状态仍然生成。

从前面所讨论的图 6 中，可以理解数据流值 s160 的变化依赖于被访问的

存储器。数据流值能在每一存储器设备基础上被用户选择。这允许在慢速存储器上使用长延迟，并且在快速存储器上使用短延迟，而不是必须使用对应最慢速存储器的单一、最坏情况下延迟。这种方法避免了在不同存储器件之间的总线冲突，而同时将涉及不同存储器的连续 I/O 操作延迟最小化。

本发明的数据流等待状态方法适用于具有多内部存储器类型的结构，每个存储器潜在地竞争对内部总线 116 的访问权，并且每个完成读周期后都具有不同的释放数据总线的时间要求。例如，图 7 展示了图 1 的一种变化，微处理器包括了内部存储器 114A 和 114B 两部分。可以理解图 7 所示的结构中的内部总线 116 和外部总线 120 所引起的总线冲突的潜在性相等。在前面所讨论图 3-6 中的原则很容易被本领域的普通技术人员应用于多内部存储结构中去，例如如图 7 中所示的结构。

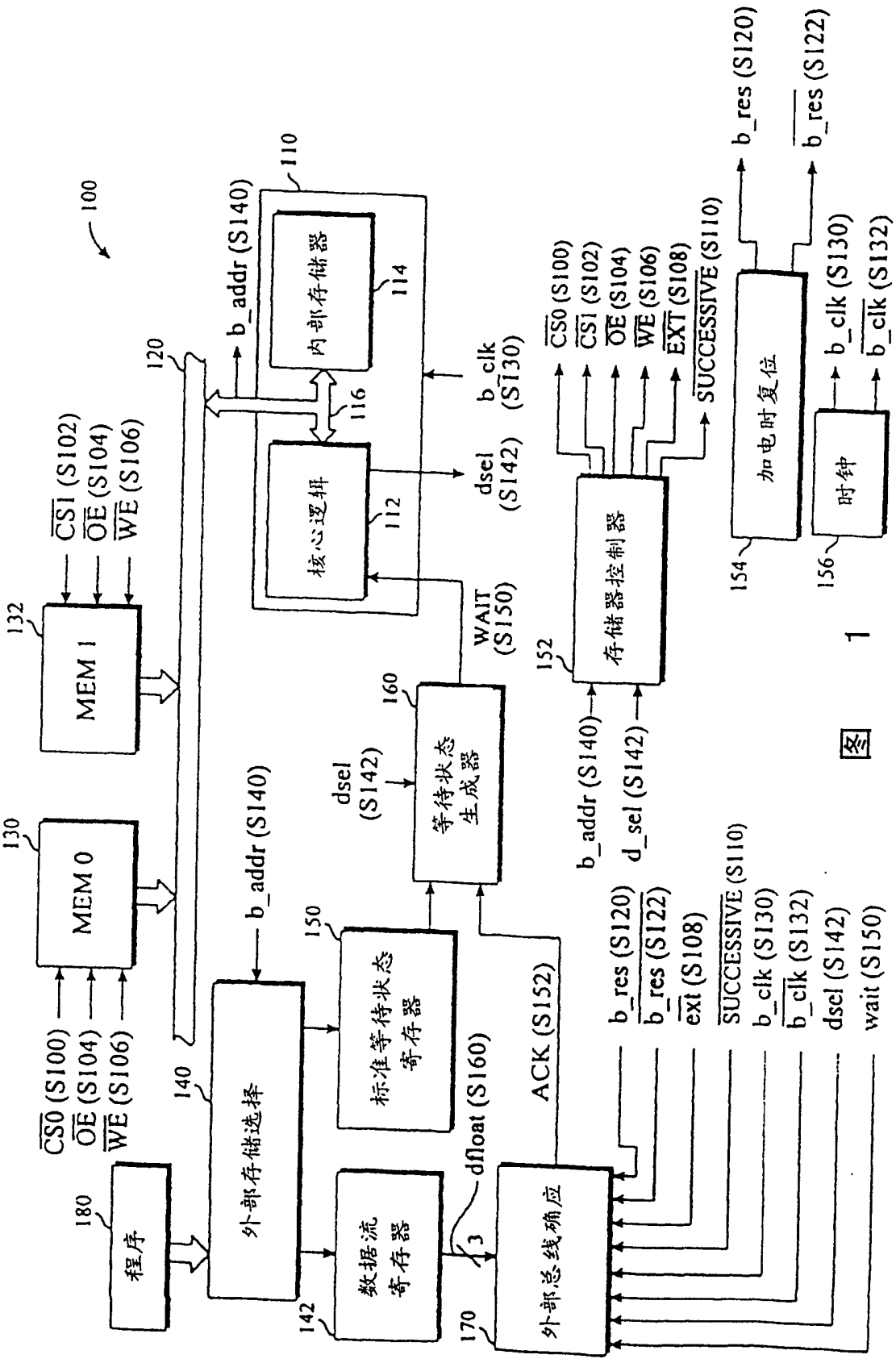

100
110
内部存储器
114
核心逻辑
112
116
b_addr (S140)
120
CS1 (S102)
OE (S104)
WE (S106)
132
MEM 1
130
MEM 0
CS0 (S100)
OE (S104)
WE (S106)
140
外部存储选择
b_addr (S140)
150
标准等待状态寄存器
dsel (S142)
160
等待状态生成器
WAIT (S150)
dsel (S142)
b_clk (S130)
152
存储器控制器
CS0 (S100)
CS1 (S102)
OE (S104)
WE (S106)
EXT (S108)
SUCCESSIVE (S110)
154
加电时复位
b_res (S120)
b_res (S122)
156
时钟
b_clk (S130)
b_clk (S132)
180
程序
142
数据流寄存器
dfloat (S160)
3
170
外部总线确应
ACK (S152)
b_addr (S140)
d_sel (S142)
b_res (S120)
b_res (S122)
ext (S108)
SUCCESSIVE (S110)
b_clk (S130)
b_clk (S132)
dsel (S142)
wait (S150)

图 1

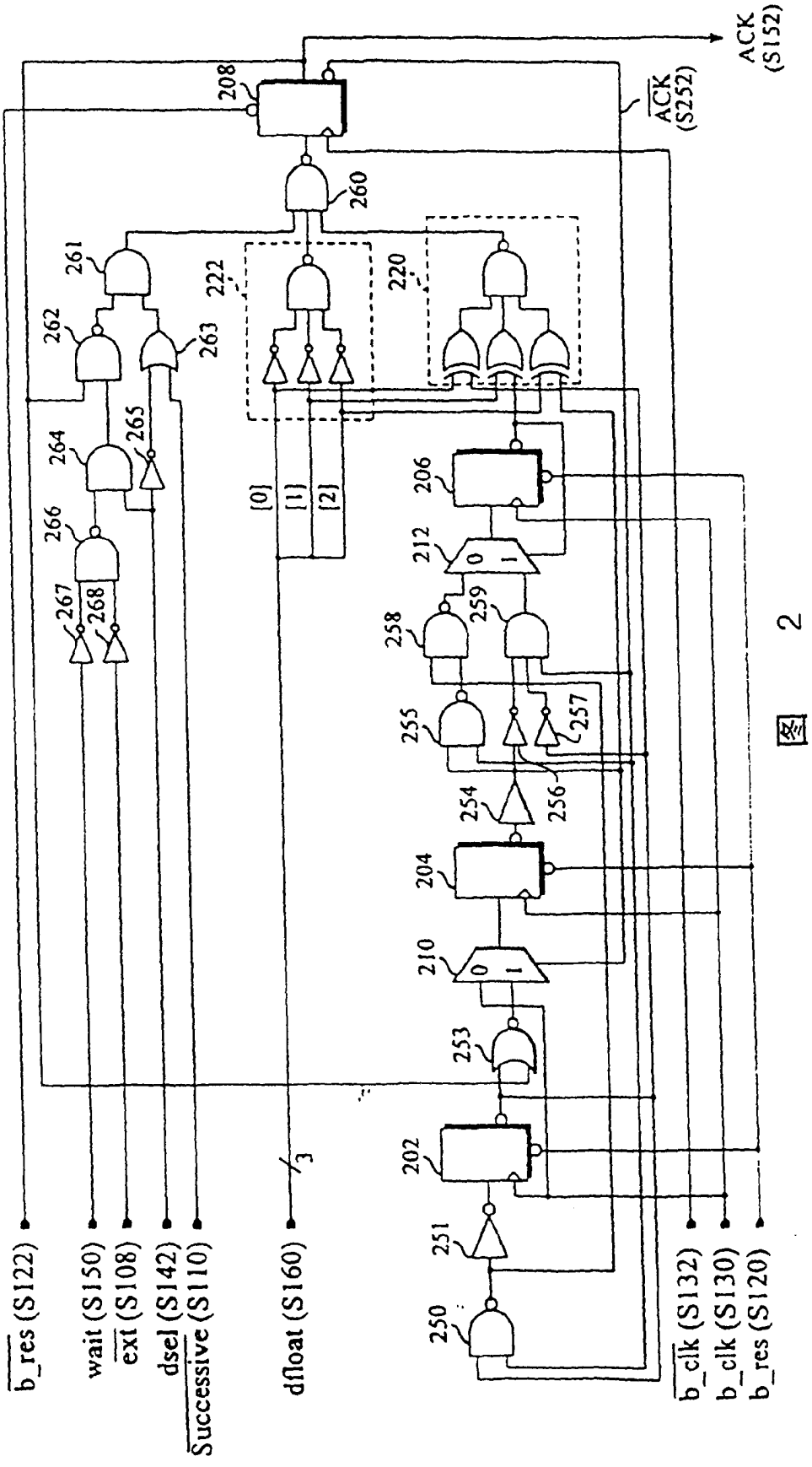

ACK (S152)
ACK (S252)
208
260
261
262
263
264
265
266
267
268
222
220
206
212
258
259
255
257
256
254
204
210
253
202
251
250
[0]
[1]
[2]
3
b_res (S122)
wait (S150)
ext (S108)
dsel (S142)
Successive (S110)
dfloat (S160)
b_clk (S132)
b_clk (S130)
b_res (S120)

图 2

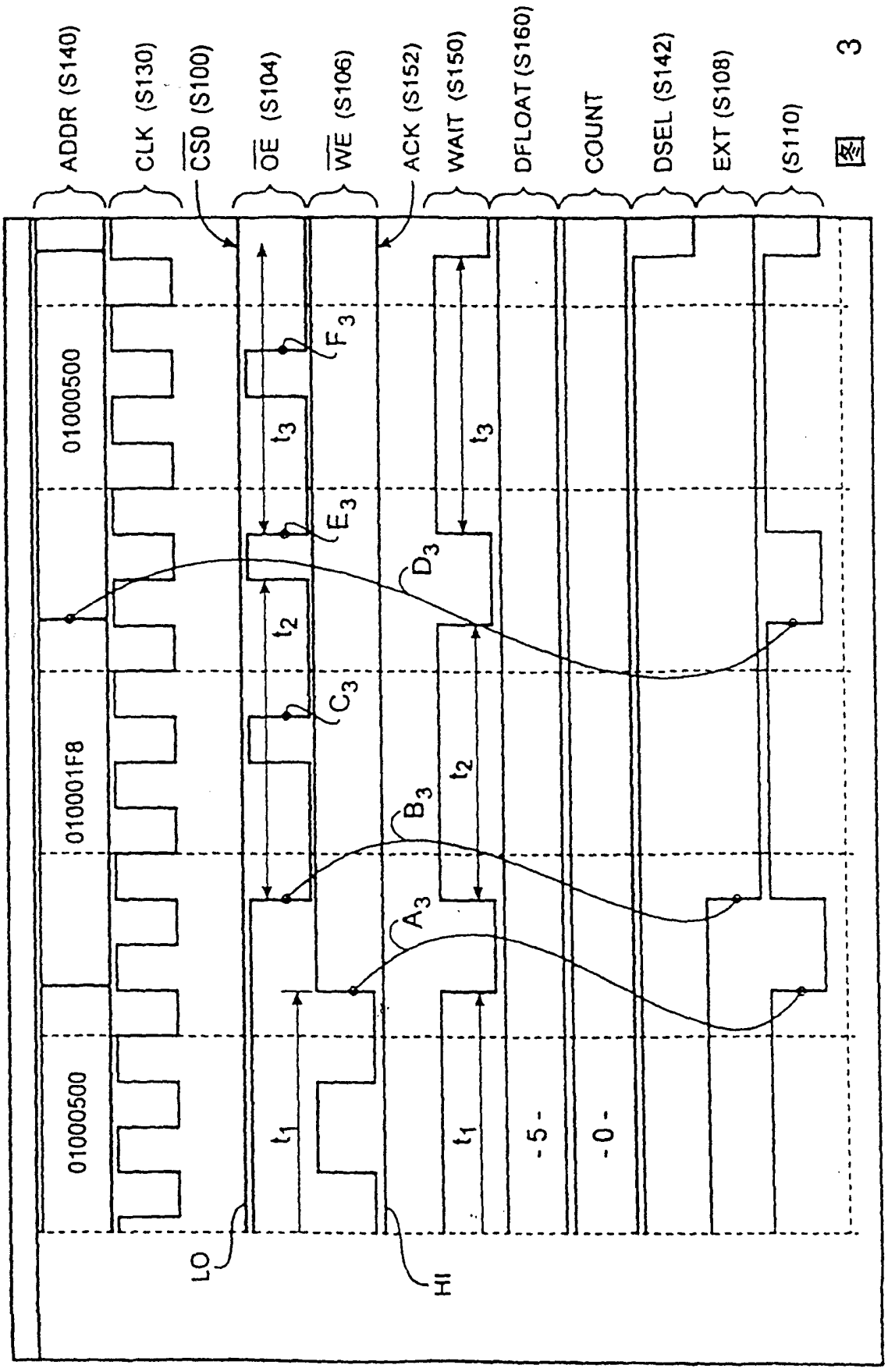

ADDR (S140)
CLK (S130)
CS0 (S100)
OE (S104)
WE (S106)
ACK (S152)
WAIT (S150)
DFLOAT (S160)
COUNT
DSEL (S142)
EXT (S108)
(S110)
01000500
010001F8
01000500
LO
HI
t1
t2
t3
A3
B3
C3
D3
E3
F3
-5-
-0-

图 3

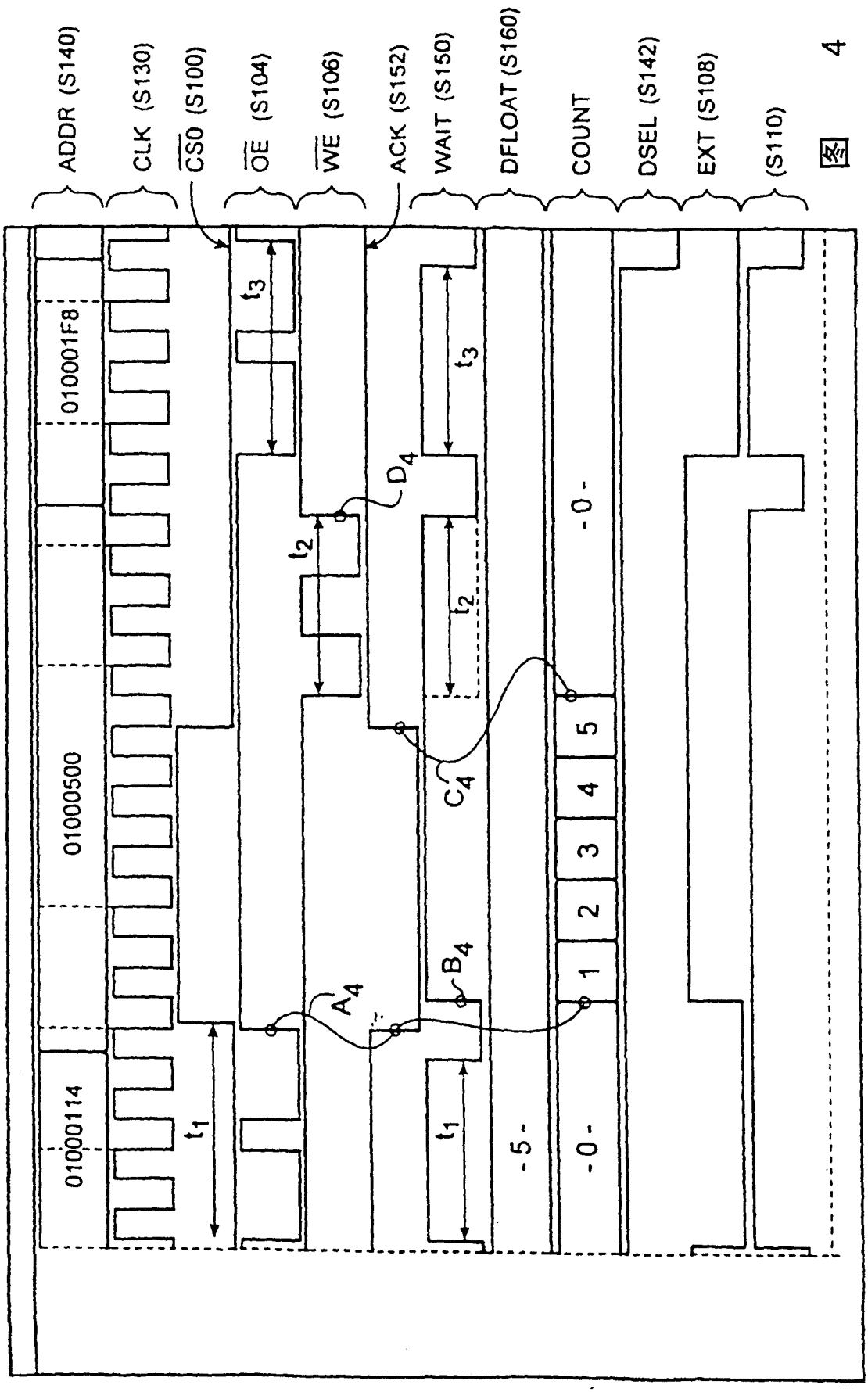

ADDR (S140)
CLK (S130)
CS0 (S100)
OE (S104)
WE (S106)
ACK (S152)
WAIT (S150)
DFLOAT (S160)
COUNT
DSEL (S142)
EXT (S108)
(S110)
010001F8
01000500
01000114
t1
t2
t3
A4
B4
C4
D4
-0-
-5-
1
2
3
4
5

图 4

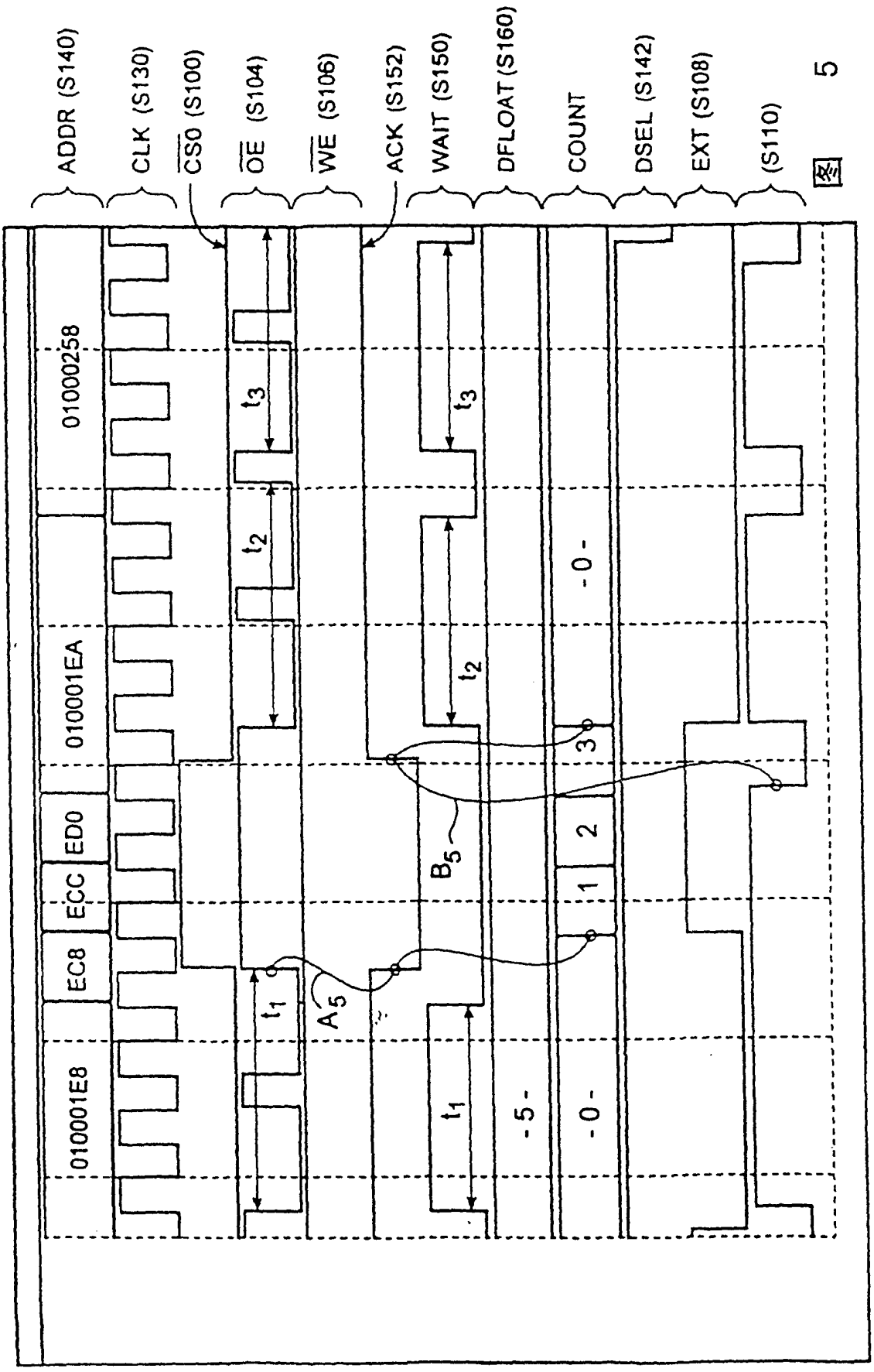

ADDR (S140)
CLK (S130)
CS0 (S100)
OE (S104)
WE (S106)
ACK (S152)
WAIT (S150)
DFLOAT (S160)
COUNT
DSEL (S142)
EXT (S108)
(S110)
01000258
010001EA
ED0
ECC
EC8
010001E8
t3
t2
t1
A5
B5
-0-
-5-
3
2
1

图 5

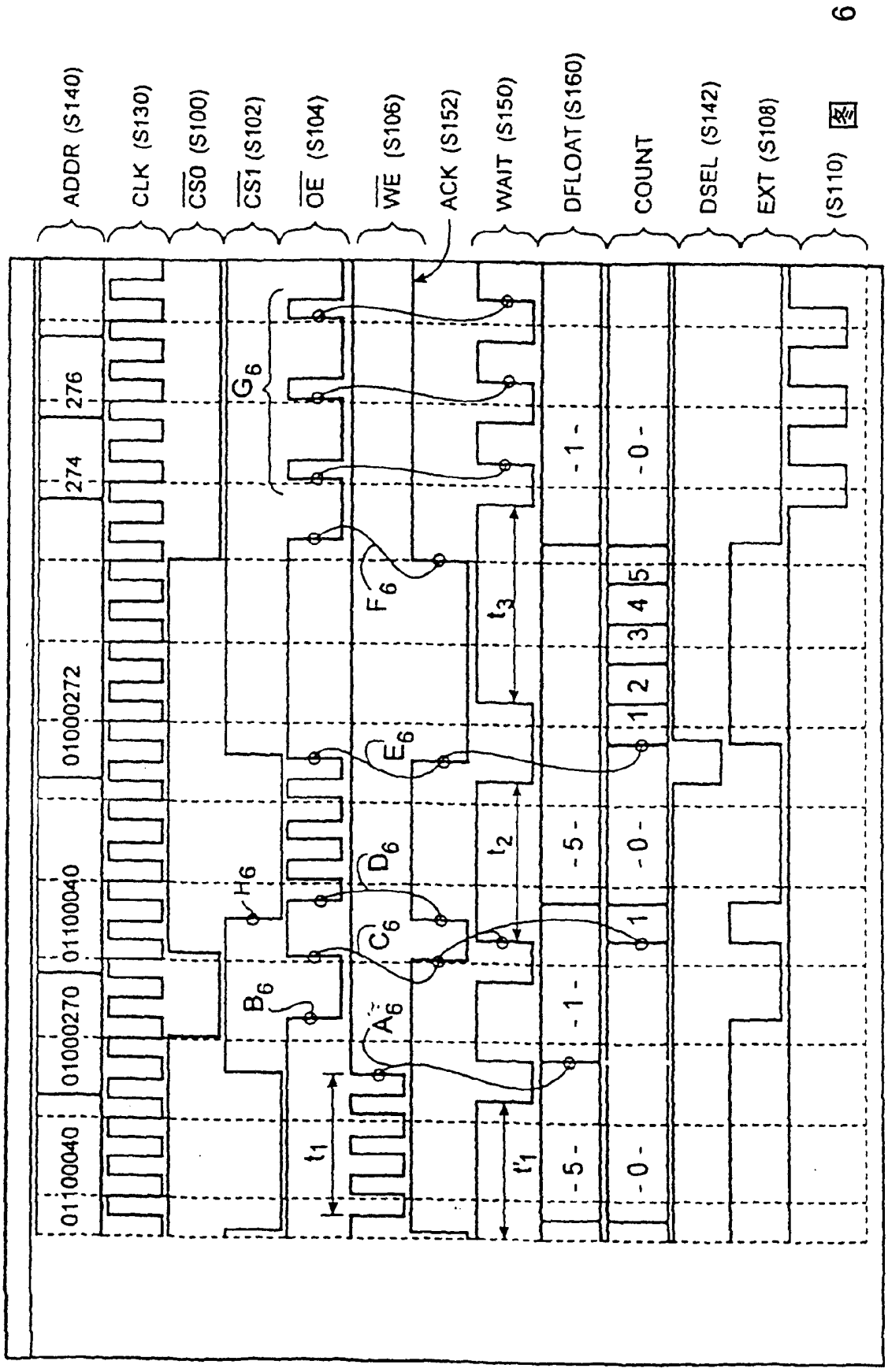

ADDR (S140)
CLK (S130)
CS0 (S100)
CS1 (S102)
OE (S104)
WE (S106)
ACK (S152)
WAIT (S150)
DFLOAT (S160)
COUNT
DSEL (S142)
EXT (S108)
(S110)
276
274
01000272
01100040
01000270
01100040
G6
F6
E6
D6
C6
B6
A6
H6
t1
t'1
t2
t3
- 1 -
- 0 -
- 5 -
1
2
3
4
5

图 6

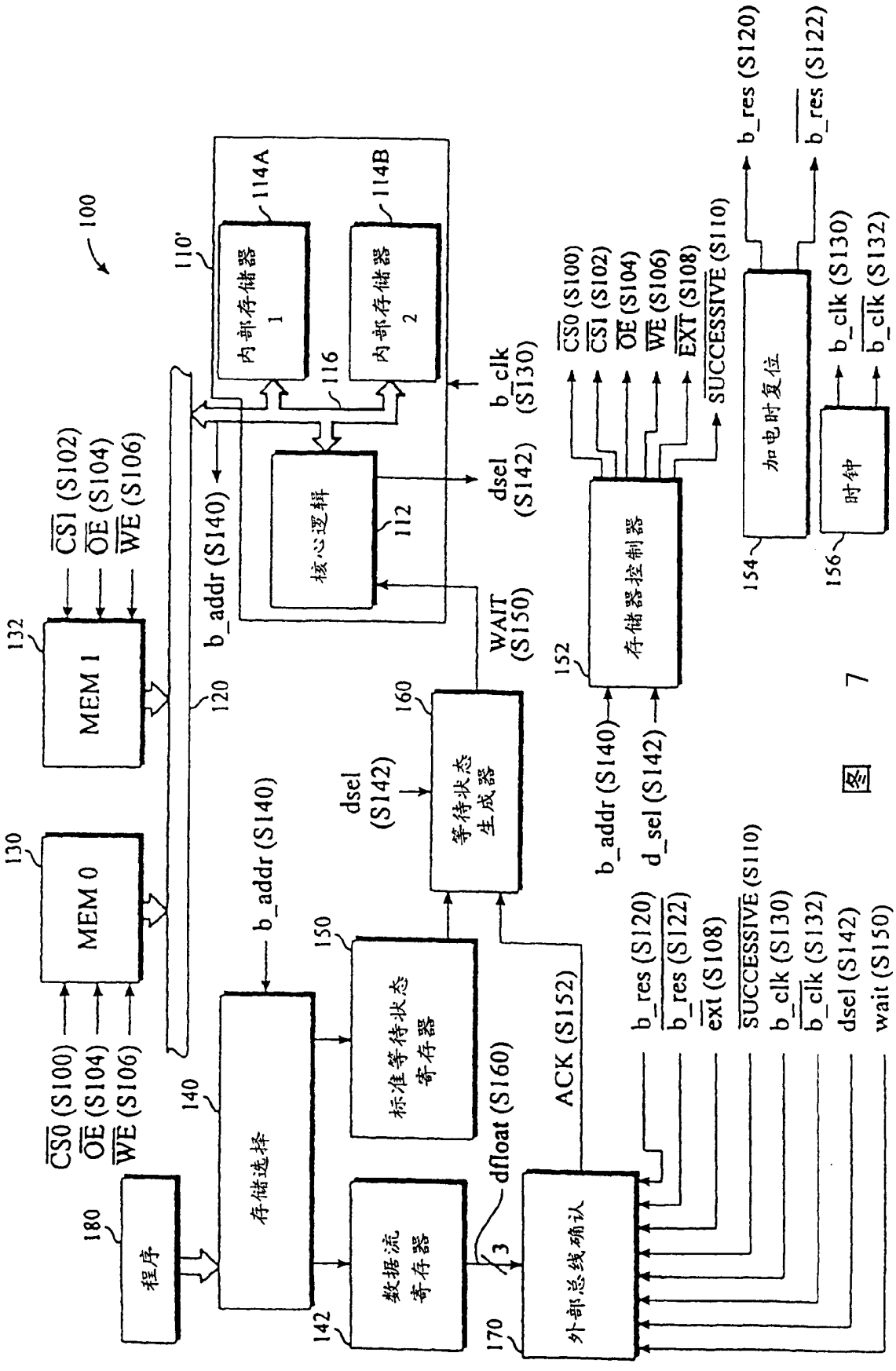

100
110'
114A
内部存储器 1
114B
内部存储器 2
116
核心逻辑
112
132
MEM 1
130
MEM 0
120
b_addr (S140)
dsel (S142)
b_clk (S130)
WAIT (S150)
160
等待状态生成器
152
存储器控制器
154
加电时复位
156
时钟
CS0 (S100)
CS1 (S102)
OE (S104)
WE (S106)
EXT (S108)
SUCCESSIVE (S110)
b_res (S120)
b_clk (S130)
b_clk (S132)
CS0 (S100)
OE (S104)
WE (S106)
140
存储选择
150
标准等待状态寄存器
dfloat (S160)
ACK (S152)
d_sel (S142)
180
程序
142
数据流寄存器
170
3
外部总线确认
b_res (S122)
ext (S108)
dsel (S142)
wait (S150)

图 7