



[12] 发明专利申请公开说明书

[21] 申请号 00818342.2

[43] 公开日 2004 年 12 月 22 日

[11] 公开号 CN 1556950A

[22] 申请日 2000.12.15 [21] 申请号 00818342.2

[30] 优先权

[32] 2000.1.14 [33] US [31] 09/483,318

[86] 国际申请 PCT/US2000/034374 2000.12.15

[87] 国际公布 WO2001/052062 英 2001.7.19

[85] 进入国家阶段日期 2002.7.12

[71] 申请人 先进微装置公司

地址 美国加利福尼亚州

[72] 发明人 瑞夫·E·吉普森

罗南·J·雪林斯基

马克·A·麦克林

[74] 专利代理机构 北京纪凯知识产权代理有限公司

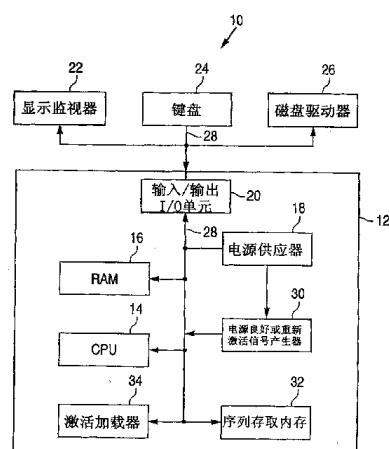
代理人 戈 泊 程 伟

权利要求书 4 页 说明书 16 页 附图 7 页

[54] 发明名称 通过储存在序列存取内存中的激活码的计算机系统起始

[57] 摘要

本发明提供一种计算机系统，包含处理器，以及具有激活程序储存于其中的序列存取内存。激活加载器包含状态机，可响应计算机系统的起始，并控制序列存取内存，以读取激活程序，接着控制处理器，使跳至序列内存的激活程序。激活程序的第一内存页更进一步使激活码转移至随机存取内存(RAM)。处理器接着跳至 RAM 里的程序代码，使得剩余的激活码由序列存取内存转移至 RAM 并执行。



1. 一种计算机系统，包含：
一个处理器；
5 一个用于储存激活程序的序列存取内存；以及
一个激活加载器，用以控制序列存取内存，以读取激活程序，并
控制该处理器，以便执行该序列存取内存里的激活程序，响应计算机
系统的起始。
2. 如权利要求 1 所述的计算机系统，其中，该激活加载器配置为可
10 输入读取指令及地址至序列存取内存，接着控制该序列存取内存执行
读取指令。
3. 如权利要求 2 所述的计算机系统，其中：
序列存取内存配置为储存激活程序于至少一个内存页；以及
读取指令使读取内存第一页。
15 4. 如权利要求 3 所述的计算机系统，其中，该读取指令为无间隔
读取指令。
5. 如权利要求 1 所述的计算机系统，其中：
该激活加载器配置为可抑制处理器，直至序列存取内存使部份激
活程序可读取，并致能处理器；以及
20 致能该处理器后，由该序列存取内存读取及执行激活程序。
6. 如权利要求 1 所述的计算机系统，其中，该激活加载器配置为
于计算机系统起始后会被激活。
7. 如权利要求 1 所述的计算机系统，其中，该激活加载器包含可
程序逻辑装置。
- 25 8. 如权利要求 1 所述的计算机系统，其中，该激活加载器包含状态机。
9. 如权利要求 8 所述的计算机系统，其中，该激活加载器包含可
程序逻辑装置，且状态机置于其中。
10. 如权利要求 8 所述的计算机系统，其中，该状态机配置为可输
30 入读取指令及地址至序列存取内存，接着控制该序列存取内存执行读
取指令。

11.如权利要求 1 所述的计算机系统，其中，该激活程序配置为可控制处理器将激活码由序列存取内存转移至挥发性随机存取内存(RAM)，接着跳至随机存取内存(RAM)里的激活码。

12.如权利要求 11 所述的计算机系统，其中，该随机存取内存(RAM)
5 包含在处理器里的快速缓冲贮存内存。

13.如权利要求 11 所述的计算机系统，其中，该随机存取内存(RAM)
包含与该处理器分离的原处执行内存(Execute-inPlace,XIP)。

14.如权利要求 1 所述的计算机系统，其中，该序列存取内存及激活加载器置于单一集成电路中。

10 15.如权利要求 1 所述的计算机系统，其中，该激活程序配置为可预测处理器的程序地址逻辑行为，以使处理器地址保持在选择序列存取内存的地址范围内。

16.如权利要求 1 所述的计算机系统，其中，该激活程序配置为可预测任何程序代码预先读取逻辑的行为，并确保必要的指令绝不会为
15 预先读取逻辑所预先读取及摒弃。

17.如权利要求 1 所述的计算机系统，其中，该激活程序配置为包含不操作(No-operation,NOP)指令，以作为程序代码区块间的填料，此处，程序代码流会因预先读取缓冲器逻辑造成中断。

18.如权利要求 1 所述的计算机系统，其中：
20 该激活程序配置为控制处理器由序列存取内存将激活码转移至挥发性随机存取内存(volatileRandomMemory)，接着跳至随机存取内存里的激活码；以及

激活程序更进一步配置为，随机存取内存(RAM)里的激活码内的分支为相对的。

25 19.如权利要求 1 所述的计算机系统，其中，该激活程序配置为控制处理器，使用立即移动指令，由序列存取内存将激活码转移至挥发性随机存取内存，接着跳至随机存取内存里的激活码。

20.一种用于计算机系统起始程序执行的方法，其中，包含处理器及序列存取内存，所包括的步骤有：

30 (a)储存激活程序于序列存取内存；
(b)提供激活加载器，用以控制序列存取内存，使读取激活程序，

并控制处理器，以便响应计算机系统的起始，跳至序列存取内存里的激活程序；以及

(c)起始计算机系统。

21.如权利要求 20 所述的方法，其中，步骤(b)包含配置激活加载器，使输入读取指令以及地址至序列存取内存，接着控制该序列存取内存执行读取指令。
5

22.如权利要求 21 所述的方法，其中：

步骤(a)包含配置序列存取内存，以储存激活程序于至少一内存页；
以及

10 步骤(b)包含配置激活加载器，使得读取指令能使读取第一内存页。

23.如权利要求 22 所述的方法，其中，该读取指令为无间隔读取指令。
15

24.如权利要求 20 所述的方法，其中：

步骤(b)包含配置激活加载器，以抑制处理器，直到序列存取内存
15 至少读取部份的激活程序，接着致能处理器；以及
致能处理器后，执行序列存取内存里的激活程序。

25.如权利要求 20 所述的方法，其中，步骤(b)包含配置激活加载器于计算机系统起始后被激活。

26.如权利要求 20 所述的方法，其中，步骤(b)包含配置激活加载器，此激活加载器包含可程序逻辑装置。
20

27.如权利要求 20 所述的方法，其中，步骤(b)包含配置激活加载器，此激活加载器包含状态机。

28.如权利要求 27 所述的方法，其中，步骤(b)包含配置激活加载器，此激活加载器包含可程序逻辑装置，且状态机置于其中。
25

29.如权利要求 27 所述的方法，其中，该状态机配置为输入读取指令以及地址至序列存取内存，接着控制序列存取内存执行读取指令。

30.如权利要求 20 所述的方法，更进一步包含有步骤：

(d)设有挥发性随机存取内存，其中：

步骤(a)包含配置激活程序，以控制处理器以将激活码由序列存取
30 内存转移至随机存取内存(RAM)，接着跳至随机存取内存里的激活码。

31.如权利要求 30 所述的方法，其中，步骤(d)包含设有随机存取

内存，此随机存取内存包含处理器里的快速缓冲贮存内存。

32.如权利要求 30 所述的方法，其中，步骤(d)包含设有随机存取内存，此随机存取内存包含和处理器分离的原处执行内存(Execute-in-Place,XIP)。

5 33.如权利要求 20 所述的方法，其中，步骤(b)包含将激活加载器及序列存取内存置于单一集成电路。

34.如权利要求 20 所述的方法，其中，步骤(a)包含配置激活程序，使能预测处理器的程序地址逻辑行为，以便使处理器地址保持在选择序列存取内存的地址范围内。

10 35.如权利要求 20 所述的方法，其中，步骤(a)包含配置激活程序，使能预测任何程序代码预先读取逻辑的行为，并确保必要的指令绝对不会被预先读取逻辑所预先读取及摒弃。

15 36.如权利要求 20 所述的方法，其中，步骤(a)包含配置激活程序，使包含不操作(No-operation,NOP)指令，以作为程序代码区块间的填料，此处，程序代码流会因预先读取缓冲器逻辑造成中断。

37.如权利要求 20 所述的方法，其中：

步骤(a)包含配置激活程序，以控制处理器由序列存取内存将激活码转移至挥发性随机存取内存，接着跳至随机存取内存里的激活码；以及

20 步骤(a)更进一步包含配置激活程序，使随机存取内存里的激活码内的分支为相对的。

38.如权利要求 20 所述的方法，其中，步骤(a)包含配置激活程序，以控制处理器使用立即移动指令，由序列存取内存将激活码转移至挥发性随机存取内存，接着跳至随机存取内存里的激活码。

通过储存在序列存取内存中的激活码的计算机系统起始

5 技术领域

本发明涉及电子数字计算机技术，尤其涉及为一组激活码储存于序列存取内存的计算机系统。

背景技术

10 常见的数字计算机系统包括含有激活码指令的非挥发性只读存储器(non-volatile Read Only Memory, ROM)。此激活程序代码乃是于操作系统或应用程控系统前，用以设定系统低阶硬件功能。

15 系统的中央处理单元(Central Processing Unit, CPU)，通常为微控制器或微处理器，设定为跳至 ROM 内的一预先决定的内存地址，并于系统起始后(包括开启电源或重新激活)开始执行激活码。激活码使操作系统或起始应用程序加载及执行。

20 激活码 ROM 一般而言为随机存取内存，其任何地址的指令或数据可直接及独立地存取。此支持大多数程序的分支特性，使得分支后的指令可从内存内的任意地址读取。于供给电源后，这些内存亦很快地便可读取。

相对于随机存取内存，发展的序列存取内存亦具优点及缺点。在序列存取内存，个别地址无法直接使用。内存以页的方式组成，例如以 512 字节为一页，且为了获得储存于页里任一特定地址的程序代码，必须将整页或半页读出。

25 可有效地利用于实行本发明的序列存取内存为 Am30LV0064D UltraNAND™，可由 Advanced Micro Devices, Inc. (AMD) of Sunnyvale, CA 取得。此内存乃依据 NAND 结构的快闪只读存储器装置。

与序列存取非挥发性内存相比，随机存取 ROMs 需较多的外部接脚及连接，主要做为地址线，故对相同位密度而言，成本明显增加，
30 且无法如同序列存取内存具如此高密度。

另一方面，序列存取内存通常需将指令序列写入装置，以选择及

将数据设定为可读取，故无法于供给电源后立即可读取。直至写入新的指令序列后，其才能由内存内的序列位置读取数据，故只能支持直线程序执行。

当计算机系统需使用较高密度及较低成本的非挥发性序列内存时，同时亦需提供非挥发性随机存取内存，至少足以支持起始程序的执行，或称为激活系统。起始程序需于非挥发性 ROM 执行，直至序列存取内存设定完成而可读取，且序列存取内存内的程序可转移至挥发性随机存取内存(volatile Random Access Memory, RAM)来执行。

如上述，序列存取内存比随机存取内存具较低的成本及较高的储存密度。对于一个采用倾向序列存取内存系统而言，为了降低系统成本及尺寸，最好能消除需使用额外的非挥发性随机存取内存来支持起始程序执行的情形。然而，过去尚无技术能达到此功能。

发明概述

如上述，本发明提供一个计算机系统，此计算机系统的起始程序的执行只使用序列存取内存来完成，因此无须额外的非挥发性随机存取内存。

此可由本发明的计算机系统来达成，包括处理器，一个具有激活程序储存其中的序列存取内存，以及一个激活加载器(boot loader)。激活加载器包含状态机，可响应计算机系统的起始，并控制序列存取内存去读取含有激活程序第一部份的序列内存的第一页。于设计激活码第一部份时，需了解序列内存只能由内存传送连续的记忆字符至处理器。激活码第一部份指示处理器将激活码第二部分复制至挥发性 RAM。一旦激活码第二部分复制至 RAM，激活码第一部份执行一分支(跳跃, jump)指令，将控制权转移至 RAM 内的激活码第二部分。接着，激活码第二部分便可利用 RAM 的随机存取特性，得以执行正常程序代码，包括 RAM 内激活码的跳跃。激活码第二部分便可产生适当指令送至序列内存，而将其它需要的程序代码传送至 RAM。

对于此技艺的技术者而言，经由下列详细说明，并参照所附的图式，其中，相同的参考数字对应相同的部分，便可明白本发明的特征及优点。

附图说明

- 图 1 为本发明简化的区块图，用以说明计算机系统；
图 2 为电路图标，用以说明图 1 的计算机系统的激活加载器及序列存取内存；
5 图 3 为激活加载器的接脚图标；
图 4 为用以说明本发明方法的流程图；
图 5 为激活加载器的状态机的状态图标；
图 6 为本系统的时间安排图标；
图 7 为激活加载器及序列存取内存较详细的电路图标；以及
10 图 8 为简化的区块图标，用以说明将激活加载器及序列存取内存置于单一集成电路。

具体实施方式

图 1 所示为本发明的计算机系统 10，其中包含主机板 12。处理器或中央处理单元(Central Processing Unit, CPU)14，或称为微控制器或微处理器，挥发性随机存取内存(volatile Random Access Memory)16，以及电源供应器 18，依常见操作方式配置于主机板 12。输入/输出(Input/Output, I/O)单元 20 提供主机板 12 至典型 I/O 装置，如显示监视器 22，键盘 24，以及一个或多个磁盘驱动器 26 间的连接。这些组件经由信号总线、电源线、连接插头等连接，共同地显示于 28。
15
20

依照本发明，系统 10 还包含电源良好或重新激活信号产生器 30，于电源供应器 18 激活且操作电压达到适当程度后，其产生一 PWRGOOD(或 RESET)信号。系统 10 更包含序列存取内存 32 及激活加载器 34。

25 内存 32 最好采用 UltraNAND™ 内存，可由 Advanced Micro Devices (AMD) of Sunnyvale, CA. 取得。此范例所显示的内存 32 为根据 NAND 结构的 Am30LV

0064D UltraNAND™ 快闪电子可拭除式可程序只读存储器(Flash Electrically Erasable Programmable Read Only Memory, EEPROM)装置，
30 并可储存 64 兆位数据。数据储存于 16,384 内存页里，每页含有 512 字节的正常数据及 16 字节的备用数据。依照本发明，内存 32 由其最低

的内存页开始储存激活码指令，并于计算机系统 10 起始后(开启电源或重新激活)开始执行。且无使用非挥发性 RAM 作为此用途。

图 2 所示为激活加载器 34 及内存 32，且激活加载器 34 的接脚图标显示于图 3。激活加载器 34 最好采用 AmPALLV16V8-10SC 可程序逻辑装置(Programmable Logic Device, PLD)，可由 Vantis Corporation of Sunnyvale, CA 取得。内存 32 及激活加载器 34 的详细结构及操作将于下列说明的。

图 4 为流程图，用以说明本发明以系统 10 执行激活带(bootstrap)加载方法。于起始后，激活加载器 34 抑制处理机 14，从而防止其尝试执行指令。激活加载器 34 侦测到 PWRGOOD 信号后，且 PWRGOOD 信号显示电源供应器 18 产生适当操作电压时，将变为激活。

激活加载器 34 具内部状态机，且置入程序化逻辑于其中。于能够读取数据前，内存 32 需设定指令写入其中。激活加载器 34 的状态机产生这些设定指令并写入内存 32。首先，单元 34 写入读取指令至内存 32 的指令闩。接着激活加载器 34 写入地址至内存 32 的地址闩。

此范例乃使用无间隔-读取(Gapless-read)指令。无间隔-读取指令为 UltraNAND 内存 32 的超集指令，可使连续地输出多路的内存页。亦可使用正常读取指令，其一次可读取一页，且当每页加载至 528 位输出数据输入器(output data register)时，具有 7 微秒等待时间。无间隔-读取指令消除了页与页之间的等待时间，而只有当第一页加载时有一个 7 微秒等待时间。如此，不需再送任何读取指令至内存，激活码即可延伸至多路的内存页。激活加载器 34 加载至内存 32 地址闩的地址，为内存 32 的第一页的地址。然而，于本发明范畴内，正常读取指令可取代使用。

于下一步，UltraNAND 内存 32 由其内部结构读取激活码的第一页至其输出数据输入器，并藉由设定其 准备好/忙碌 (RY/BY#)信号至准备好(Ready)状态，来指示此页读取操作的完成。接着激活加载器 34 的状态机藉由将 INIT#改为无效，以激活处理器 14，使其继续执行指令。处理器 14 含有常见的状态机或硬微码(hard microcode)，使其跳至预先决定，处理器可读取到系统的激活码的地址。系统会译解此地址以便选择内存 32。接着处理器会开始由内存 32 的输出数据输入器内的

内存页读取并执行激活码指令。

注意处理器会接受来自内存激活码指令的任何数据。处理器所显示的激活码地址将为内存所忽略。此意谓相同的第一内存页可用于任何处理器，无论处理器使用低或高数值地址来选择激活码。然而，激活码必须知道处理器用以选择激活码的地址。在处理器使用高内存地址做为激活码第一字符的情形下，激活码必须包含跳跃指令，使得处理器的地址改变至较低地址。此地址仍须位于系统所译解，用以选择内存 32 的地址范围内。若无跳跃指令，处理器会增加其地址，至某一点后，会回复至不再选择激活内存 32 的低内存地址。所用的实际地址并不要紧，因 UltraNAND 装置仅会传送和地址总线无关的连续指令流。然而，跳跃终点地址必须远低于内存顶端，以使地址维持于内存 32 地址空间，直至激活码第一部份跳至复制至 RAM 的激活码第二部分。

激活码亦需知道处理器如何读取指令。许多处理器具有指令读取状态机，于处理器的指令执行状态机实际使用指令前，会读取指令至缓冲器。此通常称为指令的预先读取(prefetching)。当执行跳跃指令时，将忽略预先读取指令。当跳跃指令由下一个将执行的指令改变地址时，任何预先读取，且置于预先读取缓冲器的指令，将不执行。此点很重要，因序列存取内存会忽略处理器地址。每当内存读取完成时，序列地址内存只由内存传送下一指令。因指令预先读取状态机于跳跃指令执行且清除预先读取缓冲器前，可由内存取得数个指令，激活码必须知道内存中有多少字符会为跳跃指令所摒弃。下一个需于跳跃之后执行的指令，不可在跳跃指令之后，会在由预先读取缓冲器被清除的范围内。若无法精确预测必须跳过多少字符，则在跳跃指令后，激活码需含有足够的不操作(No-operation, NOP)指令，以确保不会摒弃跳跃后的指令。直到下一个有效指令送达激活码前，皆可安全地执行跳跃后的任何 NOP 指令。所读取以及稍后由预先读取缓冲器所清除的 NOP 指令数目，并不会影响或干扰剩余激活码的执行。

处理完上述有关指令地址读取的调整后，下一步乃是将激活码第二部分复制至 RAM。激活码的下一指令最好为“字符串复制”(string copy)操作，使得内存 32 中一连串紧接的字符复制至原处执行(Execute in Place, XIP)挥发性随机存取内存。这些指令为和激活码第一部份相连

的激活码第二部分。XIP 内存可为 RAM 16，快取缓冲储存区，或是处理器 14 里的其它内存。

再者，因有些处理器会执行指令预先读取，故也许需在字符串读取指令及激活码第二部分起点间插入 NOP 指令。此在于确保，字符串复制开始由内存读取字符前，激活码的第二部分不会读取至预先读取缓冲器。也许无法预测已经在预先读取缓冲器内的字符，和最先由字符串复制指令所复制的字符，两者的起始分界。因此，欲复制的字符串末端必须接着额外的 NOP 指令，且复制的长度需确保会复制所有激活码的第二部分。故所复制的字符串也许会包含一些 NOP 指令的起点或末端。NOP 指令不会影响激活码第一或第二部分所执行的操作。然而，他们使已执行的指令及已复制的字符串间的分界更有弹性。

因无法得知字符串复制里程序代码的确切校正，故须使用已加载至 XIP 内存里，部分激活带程序内的相对分支。

为集合 XIP 内存里的激活码第二部分，亦可使用一连串的立即移动(move immediate)指令。如此可对程序代码校正有较多的控制。然而，此举将使程序代码变大且更难设计，因程序代码第二部分需由指令流转换至立即移动指令流，而其欲移动的数据为激活码第二部分的映像。

激活码第二部分储存至 XIP 内存后，由内存 32 执行的激活码第一部份的最后一个指令，使处理器 14 跳至 XIP 内存里的一个地址。接着，
20 处理器开始执行由内存 32 复制而来的激活码第二部分。此为激活带过程，可分为几个阶段。第一阶段为由序列存取内存 32 执行的激活码。下一阶段以及其后的任意阶段为起始码的额外区块，这些区块由内存 32 复制至 XIP 内存，接着跳至此执行。虽然并未明确阐明，激活带过程亦会继续由硬盘加载及激活操作系统，或是一个或多个应用程序。

状态机及激活加载器 34 相关功能最好能够和序列存取内存置于单一集成电路 40，如图 8 所示的 32' 及 34'。此部分，例如，可藉由增加足够的逻辑电路至序列存取内存来执行所需的激活加载器功能而达成。此实施例，和序列内存需结合一小部份随机存取内存的方式相比，较为简单且需少装置的修改。此乃因状态机不需任何额外处理步骤或
30 额外地址接脚。UltraNAND 或其它序列存取内存仍可和其它相似种类的内存的接脚兼容。

下列为本发明的详细功能说明

概要

UltraNAND 生产线乃 Advanced Micor Devices, Inc. (AMD)为满足高密度非挥发性内存需求所发展。应用对象包括于固定或可移动媒体系统的程序代码及数据储存。以下为激活加载器 PLD 和 UltraNAND 内存共同使用的详细说明。

使用微控制器系统通常具可程序 输入/输出 接脚(Programmable Input/Output, PIO)，可用于直接提供 UltraNAND 控制信号。若无这些 PIO 接脚，则必须具额外的逻辑接口，以做为处理器，激活加载器，以及序列内存间的适当连接。激活加载器 PLD 可设计置于单一 AmPALLV16V8-10SC 芯片，其目的在于支持于激活电源后，需直接由 UltraNAND 装置激活的激活码储存应用。

ULTRANAND 程序代码储存的相关优点

UltraNAND 设计为和在市面已可取得的 NAND 结构闪存，不论在硬件或软件均可完全兼容。然而，UltraNAND 为一改进产品，可提供 100,000 次 写入程序/清除，而不需错误修正电路(Error Correction Circuits, ECC)。UltraNAND 亦具有 100 % 良好区块，而无须损坏区块检测。

每位成本较低(低于 NOR Flash)，不需 ECC 即可增加 写入程序/清除 次数，且具有 100% 的良好装置，使得 UltraNAND 为理想的程序代码储存应用。尤其是系统码储存于 UltraNAND 闪存，并转移或投影至高速内存资源，例如同步动态随机存取内存(synchronous DRAM)，以作为快速随机存取或执行。

激活加载器 PLD

对于需要直接由 UltraNAND 激活的程序代码储存应用，UltraNAND 起始需有某种的激活加载器。此乃因激活电源后，UltraNAND 数据输入器并无有效数据。本激活加载器 PLD 34 隔绝处理器和总线间连接，并提供激活所需的控制信号序列，最高可激活两个 UltraNAND 快闪装置。

起始过程释出 AMD 超集无间隔读取指令(02h) (AMD superset Gapless Read command(02h))至每个 UltraNAND 装置，以预先加载含有

激活码的第一闪存页至内部 UltraNAND 数据输入器。此可使系统的微控制器或处理器 14 于激活电源后，由内存 32 读取并执行序列系统激活码。对程序储存应用而言，系统的微控制器可由 UltraNAND 装置 32 激活，接着将储存于 UltraNAND 内的程序代码剩余部分转移至 5 RAM，以执行程序代码。因程序代码可由 RAM 执行，故称的为原处执行(Execute-in-Place, XIP)内存。

激活存取的 ULTRANAND 接口需求

UltraNAND 内存 32 采用多路传输 地址/数据总线。所有指令、地址、以及数据经由 I/O[0..7](八位 I/O 端口)传入或传出装置。装置所 10 提供的控制信号有 CE#(芯片激活, Chip Enable)、CLE(指令闩激活, Command Latch Enable)、ALE(地址闩激活, Address Latch Enable)、WE#(写入激活, Write Enable)、SE#(备用区域激活, Spare Area Enable)、以及 WP#(写入保护, Write Protect)。此外亦有开启汲极 RY/BY#(准备好/忙碌, Ready/Busy)输出接脚，用于指示何时装置正在忙于内部操作。

15 使用 UltraNAND 的系统应用必须产生适当控制信号给装置，而这些装置，在很多情形下并未被其它系统资源所使用。此处所述的激活加载器 PLD 34，提供装置起始时，UltraNAND 内存 32 所需的全部独特信号。

20 为了将 UltraNAND 装置内所含的一页数据转移至内部数据输入器，需执行指令及地址序列。于指令及地址输入之后，UltraNAND 装置会于 7 微秒的读取等待时间周期内(最坏情形)，将数据由适当的快闪页转移至数据输入器，且在此时装置会显示忙碌。一旦非挥发性页的数据成功地转移至数据输入器，UltraNAND 装置会预准备好，且系统可连续地以最快每字节 50 奈秒的速度，由快闪装置 32 读取有效数据。

系统接口说明

25 于此范例，激活加载器 PLD 单元 34 支持单一 Ultra NAND 装置 32。于激活加载过程中，激活加载器 PLD 暂时变为由 UltraNAND 快闪所需的正常系统控制信号来源。激活加载器 PLD 接着便可产生 UltraNAND 起始所需的信号序列。

30 电源良好信号产生器 30 提供 PWRGOOD 输入信号至激活加载器 PLD 34，其仍维持在低点，直至 Vcc 有效。当 PWRGOOD 由无效状态

(低点, low)过渡至有效状态(高点, high), 激活加载器 PLD 34 侦测到此过渡并起始 UltraNAND 装置 32。

于某些无 PWRGOOD 信号的应用, 可使用系统 RESET#信号。激活加载器 PLD 所产生的 INIT#(起始)信号, 乃用于指示激活加载器 PLD 34 正忙于起始 UltraNAND 装置 32。系统 10 必须监控 INIT#信号, 并隔绝系统微控制器 14 和总线间的连接, 直至 INIT#变为无效(高点)。

一旦 INIT#指示 UltraNAND 激活过程已完成, 系统 10 便可允许微控制器 14 由 UltraNAND 装置读取数据。

许多情形下, INIT#信号可作为系统 10 其它部分的重新激活信号。
正常系统 RESET#信号只到达激活加载器 PLD 34, 且激活加载器使用 INIT#信号使系统 10 的其余部分维持在重新激活状态, 直到 UltraNAND 装置 32 可开始读取为止。

于激活加载器过程的地址阶段, 直到装置变为忙碌状态前, 地址字节会写入 UltraNAND。如此可使激活加载器 34 支持需三个或以上地址脉冲的装置。于激活加载器起始过程后, 激活加载器会三态 (tri-state)WE#、 CLE、 ALE、 及 I/O1, 并将 CE0# 及 CE1#由系统总线 28 传至 Ultra
NAND 装置。

激活加载器信号说明

UltraNAND 激活加载器 34 利用无间隔读取指令序列产生起始一或二个 UltraNAND 装置所需的所有信号。PLD 34 内的简易状态机控制 INIT#、 WE#、 CLE、 ALE、 及 I/O1, 藉此控制 UltraNAND 的起始。为了启始激活加载器 PLD, 系统需提供 SYSCLK (系统时间, System Clock)及 PWRGOOD (电源良好指示)。所有适用信号的定义用于产生信号的来源均列于下表。激活加载器起始顺序的时间图标表示于图 6。

信号	来源	定义
ALE	激活加载器/系统	UltraNAND 地址周期的三态地址门激活
CLE	激活加载器/系统	UltraNAND 指令周期的三态指令门激活
WE#	激活加载器/系统	于激活时用于起始 UltraNAND 的三态写入信号
INIT#	激活加载器	指示激活加载器 PLD 正在起始 UltraNAND
I/O [0..7]	激活加载器/系统	用于转移指令及地址至 UltraNAND 的 I/O 线
ST3	加载激活器	PLD 未使用的最明显状态机输出
OE#	激活加载器	必须和 INIT#一起使用, 以激活 PLD 登录输出
OUTCE [0..1] #	激活加载器	芯片激活由 PLD 至 UltraNAND 装置的输出
CE [0..1]#	系统	芯片激活系统至 UltraNAND 的译解信号
PWRGOOD	系统	电源良好信号维持在低点, 直至 VCC 有效
STSCLK	系统	激活加载器状态机所需的系统时间
READY	UltraNAND	连接至 RY/BY#信号, 以指示 UltraNAND 状态

激活加载器 PLD 操作原理

此节将说明激活加载器 PLD 34 的设计及操作。激活加载器 PLD 34 使用 flip-flops 作为状态机及 I_01。组合逻辑用于产生控制最高为两个 UltraNAND 32 装置的其它所有信号。本节所指定的装置参考指示显示于图 7, 其为“支持两个 UltraNAND 装置的典型激活加载器应用”的图例。

输出信号产生

INIT#为一组合输出信号, 用于通知系统激活加载器 PLD 34 正处于起始 UltraNAND 32 数组状态。当 INIT#激活, 系统 10 必须维持三态系统信号(如地址线或 PIO 接脚), 这些信号使用于正常系统操作, 用以驱动 UltraNAND CLE、ALE、WE#、及 I/O[0..7]信号。许多情形下, INIT#信号可作为至其它系统组件的重新激活信号, 因重新激活状态通常会使处理器输出维持在三态状态。

于 INIT#时期, PLD 34 将指令及地址数据送至 UltraNAND 32 数据

总线，并控制 UltraNAND 起始所需的 CLE、ALE、及 WE#。于指令阶段，CLE 激活，且 I_01 用于将 02h 值送至 UltraNAND 32 数据总线，以将无间隔读取指令写入 UltraNAND 32。于地址阶段，CLE 关闭，ALE 激活，将 00h 值送至 UltraNAND 32 数据总线。

藉此激活加载器 PLD 34 可将区块 0, 页 0, 字节 0 的地址加载至 UltraNAND 32 装置，使系统于起始后由第一字节开始读取。于 INIT#，不管芯片激活输入的状态为何，PLD 32 会激活输出芯片激活。此可使激活加载器 PLD 34 于激活加载器操作时，同时起始最高可达两个的 UltraNAND 32 装置。

I_01 为三态登录输出，于激活加载器 PLD 34 起始阶段时，用以将无间隔读取指令写入 UltraNAND 32 装置。当 INIT#激活时，系统需将 UltraNAND 32 数据总线，I/O[0..7]维持在三态。当下拉电阻 R4 使其它所有数据位维持在低点，I_01 便可于指令阶段变为高点，并将 02h op-code 写入至 UltraNAND 32。于地址阶段，I_01 处于低点，使得所有地址周期将 00h 值写入 UltraNAND 32 地址登录器。OUTCE[0..1]# 为激活 UltraNAND 32 装置所需的两个芯片激活输出。

于激活加载器运作的 INIT#时期，芯片激活必须强迫激活。一旦 INIT#变为关闭(高点)，芯片激活输出乃由芯片激活输入所决定。ST[0..3] 为四个三态登录状态位，用于定义状态机状态，最高可至 16 个状态，其中 9 个状态为实际使用的。于 INIT#激活时期，亦激活这些位，当 INIT#关闭时，则处于三态。

其中三个状态位用于定义 WE# (ST0)、CLE (ST1)、及 ALE(ST2)。激活加载器 PLD 34 内的状态机为典型米力(Mealy)机，其中 PLD 输出乃依据电流状态及输入条件。AmPALLV16V8-10SC 装置 34 激活时，所有登录输出位于高点状态。因此，状态机定义 IDLE 状态为所有均为输出条件。

状态 01 至状态 09 为灰阶编码变化(gray coded variations)，以消除组合输出的噪声脉冲，并避免不同步输入所导致的竞争情形。状态机的流程图显示于图 5。

ST0，如同 WE#，乃是于起始时，用于将数据写入 UltraNAND 32 快闪。于起始时期，系统必须使系统 WE#输入至 UltraNAND 32 维持

三态。一旦起始完成，激活加载器 PLD 34 会三态其 WE#输出，接着系统 10 需要时便可驱动 WE#。

ST1，如同 CLE，乃于起始时，用于激活写入至 UltraNAND 快闪 Flash 32 的指令。于起始时期，系统必须使系统 CLE 5 输入至 UltraNAND 32 维持三态。一旦起始完成，激活加载器 PLD 34 会三态其 CLE 输出，接着系统 10 需要时便可驱动 CLE。

ST2，如同 ALE，乃于起始时，用于激活写入至 UltraNAND 快闪 Flash 32 的地址。于起始阶段，系统必须使系统 ALE 输入至 UltraNAND 32 维持三态。一旦起始完成，激活加载器 PLD 34 会三态其 ALE 输出，10 接着系统 10 需要时便可驱动 ALE。

其它信号

CE[0..1]#乃由系统 10 所产生的两个芯片激活输入信号，用以选择 UltraNAND 32 装置。于起始过程，CE[0..1]#输入被激活加载器 PLD 34 所忽略，且 OUTCE[0..1]#激活，以选择所用的 UltraNAND 32 装置。15 一旦起始完成，激活加载器 PLD 34 只将系统 10 的 CE[0..1]#信号经由 OUTCE[0..1]#输出，传至 UltraNAND 32 装置。

PWRGOOD 为系统所产生的信号，用以指示激活加载器 PLD 34 何时该起始 UltraNAND 32 快闪装置。激活电源时，PWRGOOD 维持在无效(低点)，直至 Vcc 有效。激活加载器 PLD 34 供给电源后，状态机变为 IDLE 状态，直到 PWRGOOD 变为激活(高点)才解除。一旦激活加载器 PLD 34 侦测到 PWRGOOD 由低点到高点的变化，激活加载器 PLD 34 便会继续起始 UltraNAND 闪存 32。

若系统无 PWRGOOD 信号，系统的 RESET#信号亦可执行相同功能。当 RESET#变为激活(低点)以重新激活系统时，激活加载器的状态 25 机会进入 IDLE 状态。当 RESET#关闭时(高点)，激活加载器便会起始 UltraNAND 32 装置。

RY/BY#乃由 UltraNAND 32 装置所产生，用以指示何时装置正忙于内部操作。系统 10 亦可使用 RY/BY#硬件信号，或读取状态登录机内的 RY/BY#状态位，以决定何时操作正在进行，或是已经完成。激活 30 加载器 PLD 34 经由 READY 输入，监控 RT/BY#信号，以决定何时 UltraNAND 快闪 32 已完成将数据由内部快闪数组转移至内部数据输

入器。于此时，起始内存 32，且控制权可转移至系统 10。

SYSCLK 为激活加载器 PLD 34 所需，用以驱动装置状态机。因 UltraNAND 32 具有最小写入脉冲宽度规格 25 奈秒，SYSCLK 的最大频率为 40 兆赫。

5 激活加载器状态机

激活加载器 PLD 34 中具有简易的状态机，用以执行 UltraNAND 起始。激活加载器状态机驱动激活加载器 PLD 34 输出，以提供用于起始一个或两个 UltraNAND 32 装置时所需的适当信号。激活加载器 PLD 状态机的流程图示于图 5。

10 实行细节

如此处所述的激活加载器 PLD，于激活电源后，PLD 最高可起始两个 UltraNAND 装置。当使用激活加载器 PLD 时，一些基本系统接口考量亦需考虑。

15 CLE、ALE、WE#、及 I/O[0..7]的三态控制

于激活电源后，当激活加载器 PLD 正在起始 UltraNAND 快闪时，激活加载器过程必须必须控制 I/O[0..7]、CLE、ALE、以及系统 WE# 信号。因此，当 INIT#有效时(低点)系统必须使这些信号处于三态状态。一旦 INIT#变为高点，表示激活加载器起始已经完成，系统接着便可适当地驱动这些信号。

20 序列激活码范例

因系统位置总线并非直接决定 UltraNAND 快闪读取地址，故顶端或底端激活微控制器不需不同的 PLD。系统的地址译码器逻辑仅需保证当激活电源及快闪起始后，UltraNAND 可开始让微控制器读取程序代码。以下的范例为在 x86 等级微控制器中，系统激活码如何储存于 UltraNAND 快闪激活区域。

一旦起始 UltraNAND，系统微控制器便可连续地直接由 UltraNAND 数据输入器开始读取程序代码。

如果系统里的处理器使用低内存地址作为激活码的起点，储存于快闪装置的激活码不会使微控制器来回分支或跳跃。此举可确保，由 30 任何处理器的指令预先读取逻辑自 UltraNAND 所读取的指令，以及实际指令执行顺序，彼此之间没有中断。

如果处理器使用高内存地址作为激活码的第一字符，程序代码必须包含跳跃指令，使得处理器的地址变为较低的地址，但仍须位于系统所译解，用以选择内存 32 的地址范围内。若无跳跃指令，处理器会增加其地址，至某一点后，会回复至不再选择激活内存 32 的低内存地址。

当预先读取缓冲器被跳跃所清除时，跳跃会使一些由预先读取的指令流所得的字节被摒弃。因无法得知预先读取缓冲器里有多少数目的字节会被跳跃所摒弃，故于跳跃填满整个预先读取缓冲器后，需有足够的 NOP 指令。

最先直接由 UltraNAND 装置所执行的激活码仅由 UltraNAND 装置加载较为广泛的激活带程序至 XIP 内存资源，如静态随机存储器 (SRAM) 或动态随机存取内存 (DRAM)。加载激活带程序代码可藉由字符串移动或连续立即移动指令来完成。

若使用字符串移动，字符串指令后的指令流于起点及末端必须有足够的 NOP 指令，以填满处理器所使用的指令预先读取缓冲器。此举可确保由字符串移动读取操作所取得的程序代码流，由预先读取缓冲器填满后的位置开始。因无法预测预先读取缓冲器停止由 UltraNAND 读取字节的确切位置，以及字符串移动操作开始读取字节的位置，故加载至 XIP 内存资源的程序代码或许会开始于一些余留的 NOP 指令。

因无法控制程序码的确切校正，故须使用已加载至 XIP 内存里，部分激活带程序内的相对分支。对需要严格控制程序码字节校正的处理器而言，立即移动指令流或许为较好的方式来加载激活带程序代码至 XIP 内存。

一旦激活码于 XIP 内存组集合之后，最后一个指令会使微控制器跳至激活码执行。下列的范例包含以组合程序代码形式的 x86 激活码序列，其将会于起始后，由 Ultra NAND 连续地执行。

定义

SOURCE	内存对应基地址至 UltraNAND 装置。读取时，此地址必须选择 UltraNAND 装置。
DESTINATOIN	XIP 内存资源用以保留储存于 UltraNAND 的系统码的终点地址。一旦 UltraNAND 内存的内容转移至投影内存，激活码范例的最后一列跳至 XIP 区域，执行系统程序代码。
XIP-	XIP 内存的程序段地址
ULTRA -	序列内存的程序段地址
COUNT -	由 UltraNAND 装置转移至 XIP 内存的字节数目。此数目必须包含可预先读取的字节的最大值。于此例中，486 处理器的预先读取缓冲器的大小为 32 字节。
START -	激活码的起点地址。X86 处理器由 F...FFoh 开始执行，即位于内存顶端的 16 字节里。因此，需先跳至较低的程序代码地址，但此地址仍需于选择 UltraNAND 装置的范围内。如此可防止地址回复至不再选择 UltraNAND 的地址。实际地址并非紧要的，因 UltraNAND 装置仅会传送和地址总线无关的连续指令流。然而，跳跃终点地址必须远低于内存顶端，以使地址维持于 UltraNAND 地址内，直至此起始激活带程序代码跳至已复制至 XIP 内存的激活带程序代码剩余部分。
NOP -	使用 NOP 指令填满剩余字节，直至正常执行系统里的 F...FFFh 地址。如此可防止因指令预先读取单元读取分支指令外的位置，所造成的意外的程序代码执行或是分界问题。部分或所有 NOP 指令可藉由预先读取单元由 UltraNAND 读取，部分则是当分支清除预先读取缓冲器时会被摒弃。然而，由 UltraNAND 所读取的字节流内的指令，会摒弃的确切数目则无法得知。藉由使用 NOP 指令，处理器会持续读取直至遭遇到指令流内的剩余指令。

程序代码范例

JMP START	; 由重置地址跳至激活码地址范围，使得指令地址计数器不会包覆并跳至序列内存地址范围外。
NOP	; 使用 NOP 填满接下的 32 字节，以确保预先读取缓冲器于跳跃执行并清除预先读取缓冲器前，不会读取任何随后的指令。
START: STD MOV DS, ULTRA MOV ES, XIP MOV ESI, SOURCE MOV EDI, DESTINATION MOV ECX, COUNT REP MOVS B NOP... {XIP code stream} NOP... JMP DESTINATION	<p>; 于字符串移动时设定方向标记以增加终点地址</p> <p>; 加载程序段地址至 UltraNAND 内存</p> <p>; 用程序段地址加载额外程序段登录器作为 XIP 记忆</p> <p>; 加载 UltraNAND 起始地址</p> <p>; 加载终点地址以作为字符串移动的准备</p> <p>; 加载字符串移动时所转移的字节数目</p> <p>; 将字节由 UltraNAND 移至投影内存</p> <p>; 32 字节的 NOP，以填满处理器预先读取缓冲器</p> <p>; 欲移入 XIP 内存的程序代码流</p> <p>; 需 64 或更多的 NOP 指令，因字符串移动的确切停止位置必须藉由程序代码前的许多 NOP 指令以找到程序代码的终点，以确保所有的程序代码皆被移除</p> <p>; 一旦转移完成，跳至 XIP 内存内的激活带程序代码</p>

总结上述，本发明提供一组只使用序列存取内存，便完成起始程序执行的计算机系统，因此，不需独立的非挥发性随机存取内存。

对于此技艺的技术者而言，于了解本揭示之后，只要不背离本发明的范畴，便可做不同的修改。

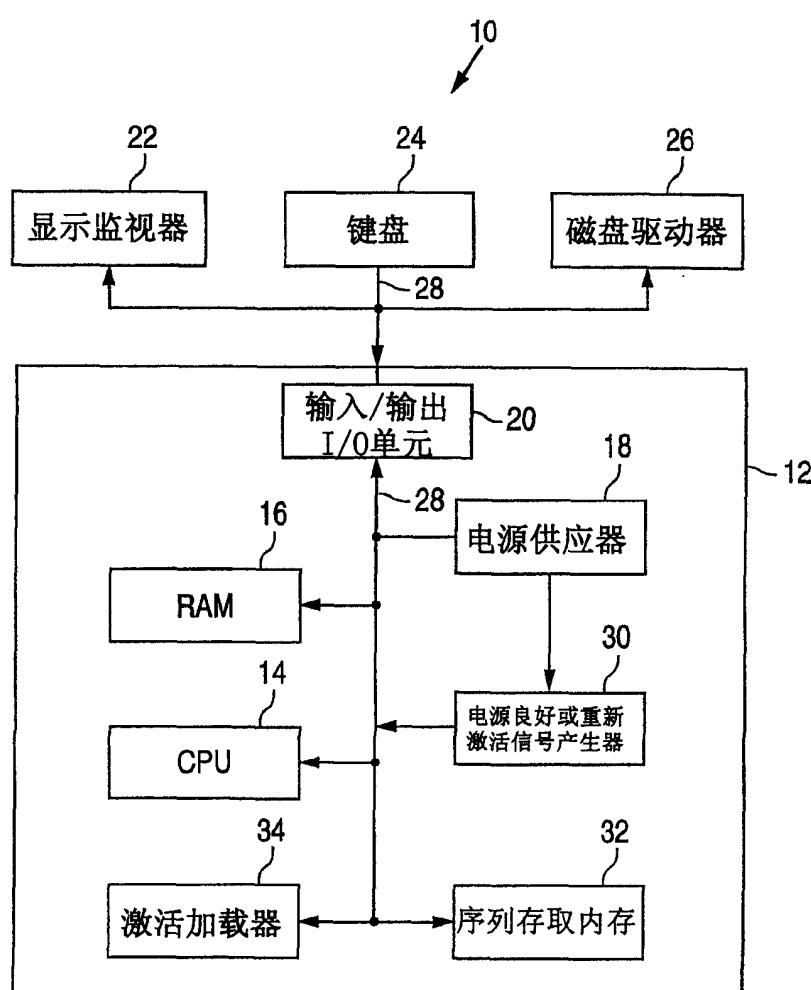


图1

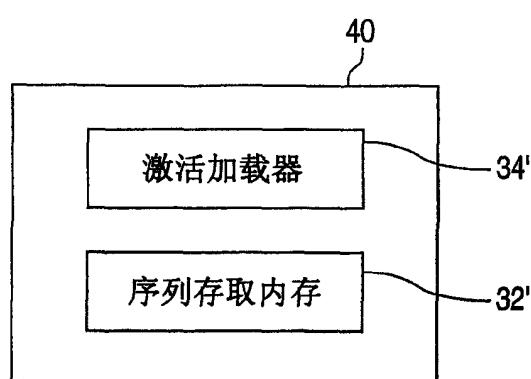


图8

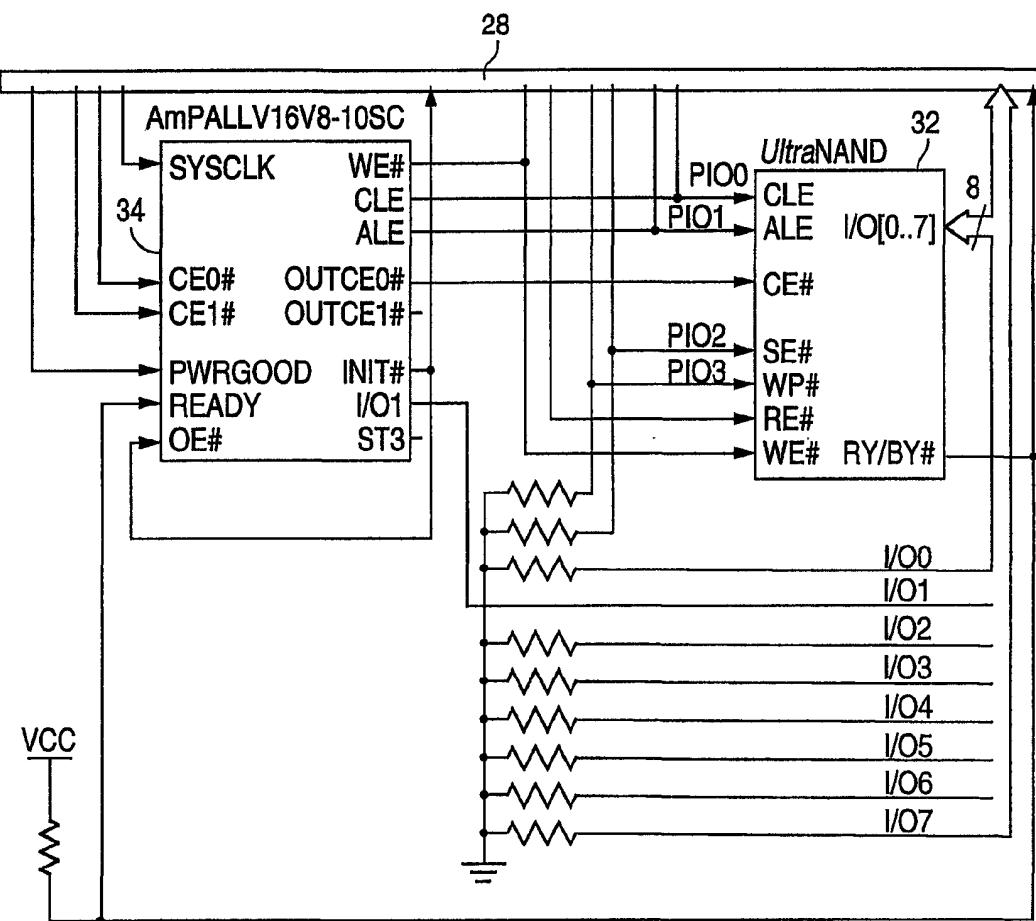


图2

	34	
SYSCLK	1	VCC
PWRGOOD	2	ST0 (WE#)
READY	3	ST1 (CLE)
CE0#	4	ST2 (ALE)
CE1#	5	ST3
NC	6	INIT#
NC	7	OUTCE0#
NC	8	OUTCE1#
NC	9	I/O1
VSS	10	OE# (to INIT#)

图3

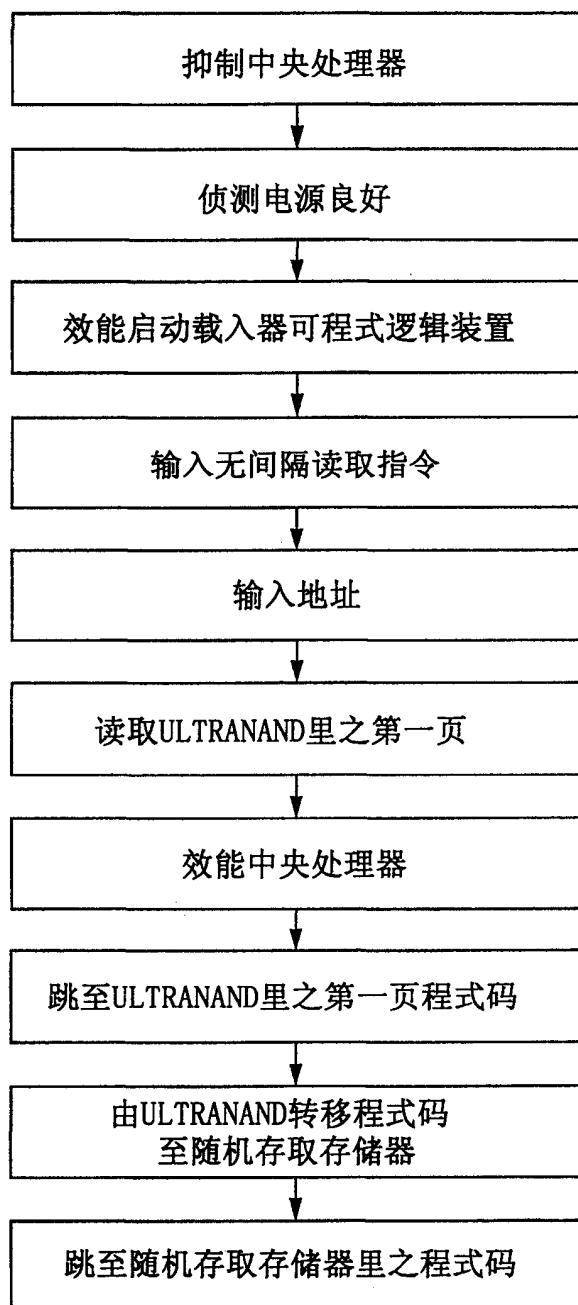


图4

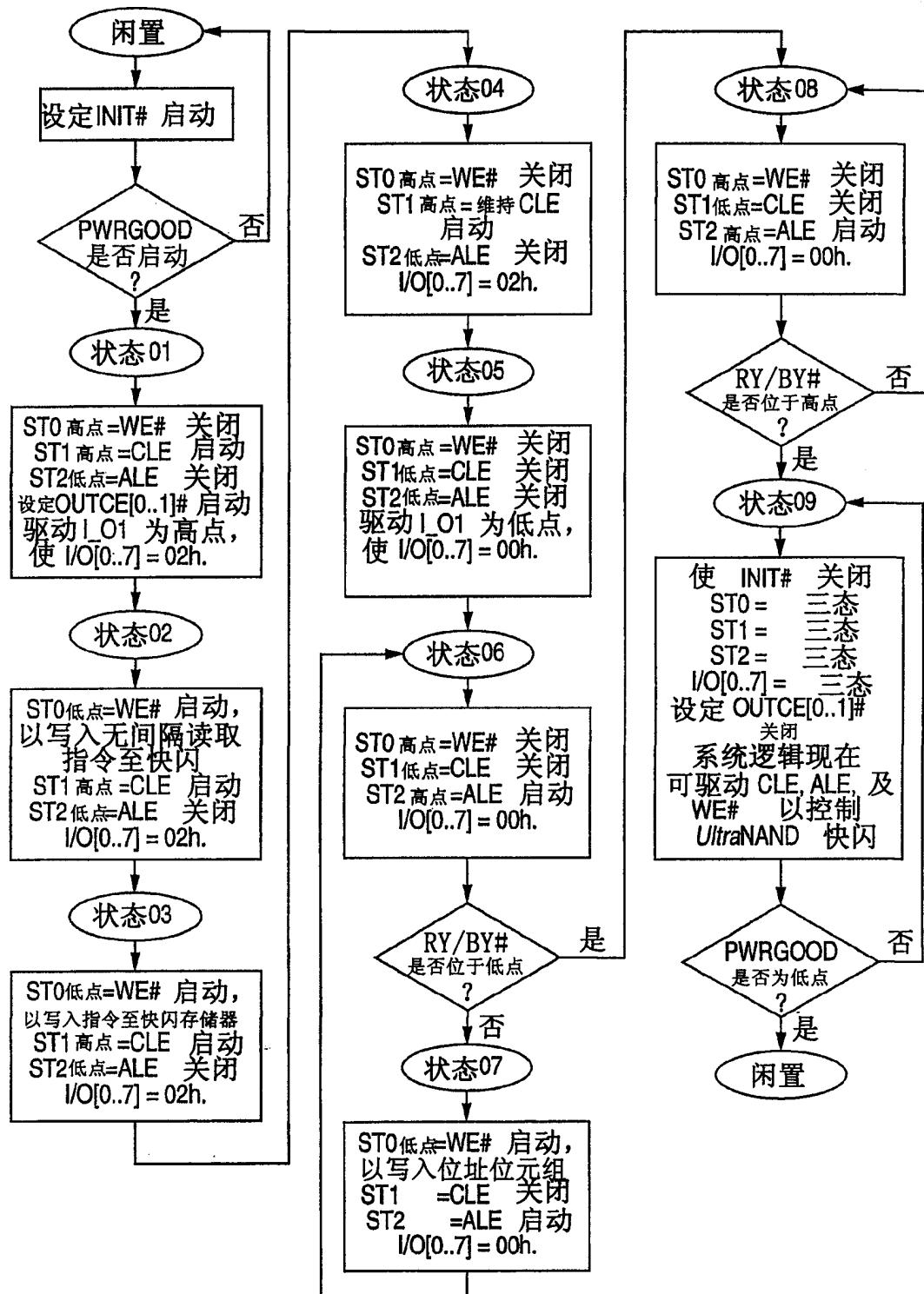


图5

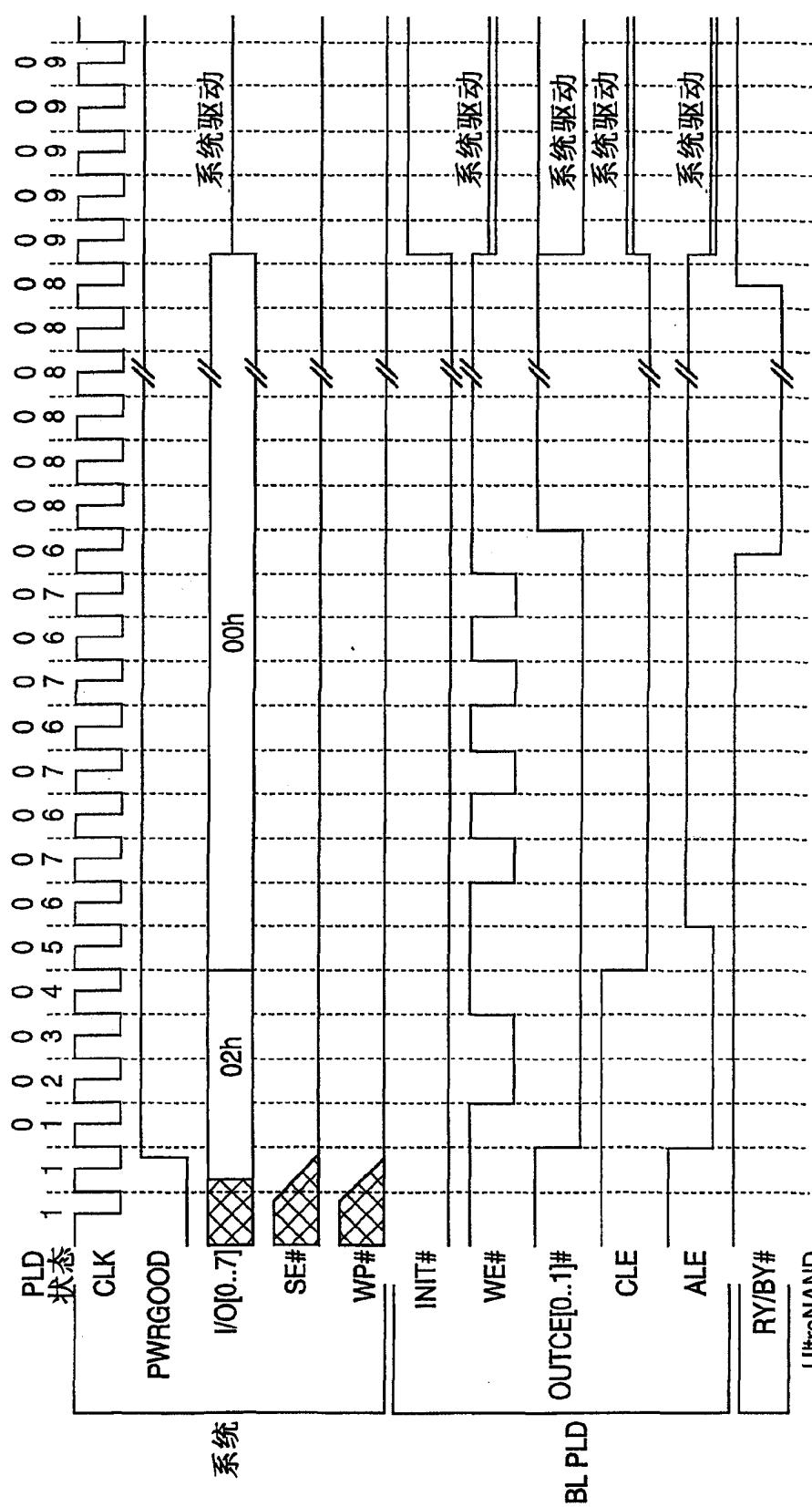
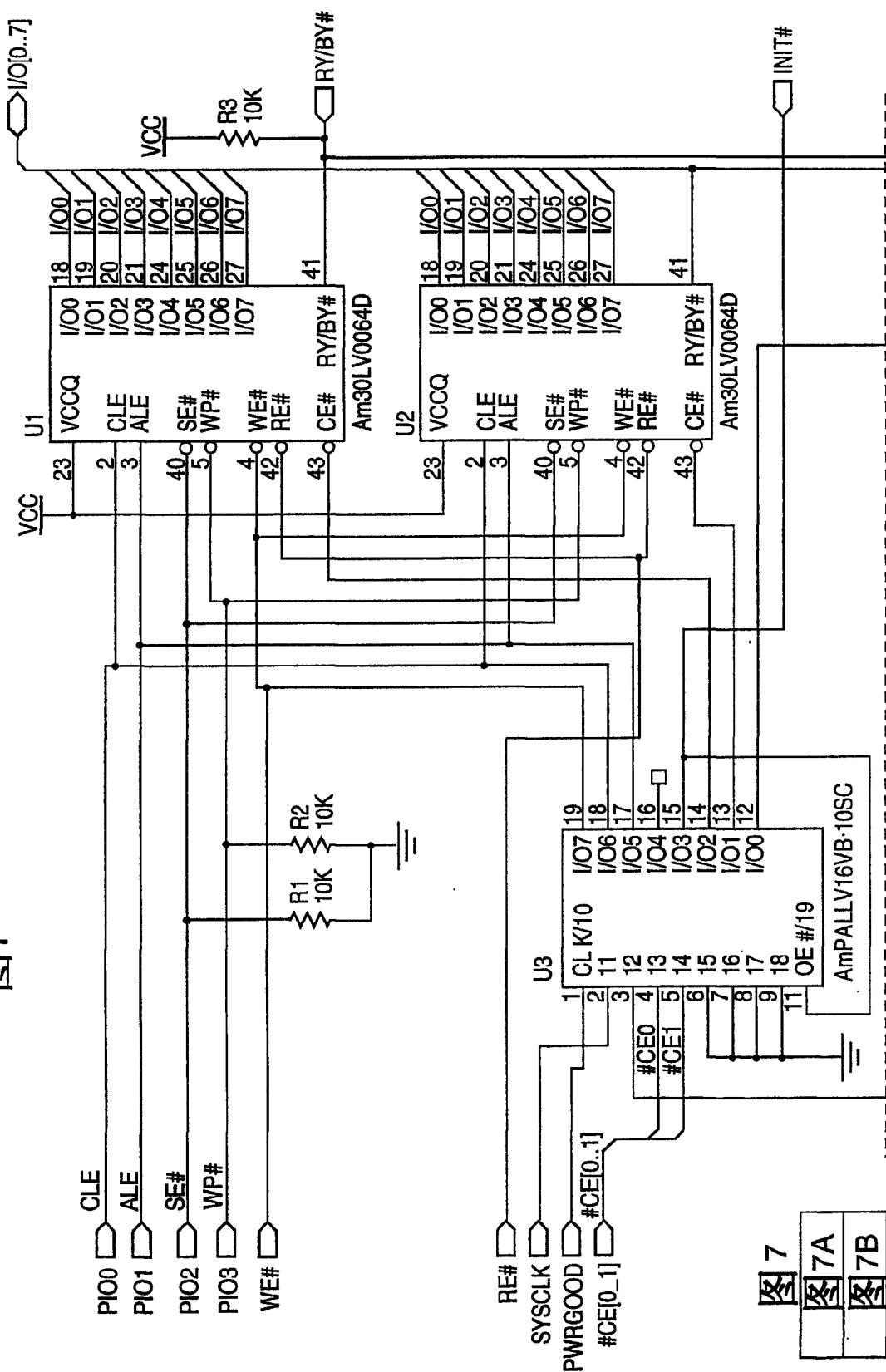


图6

图7

图7
图7A
图7B

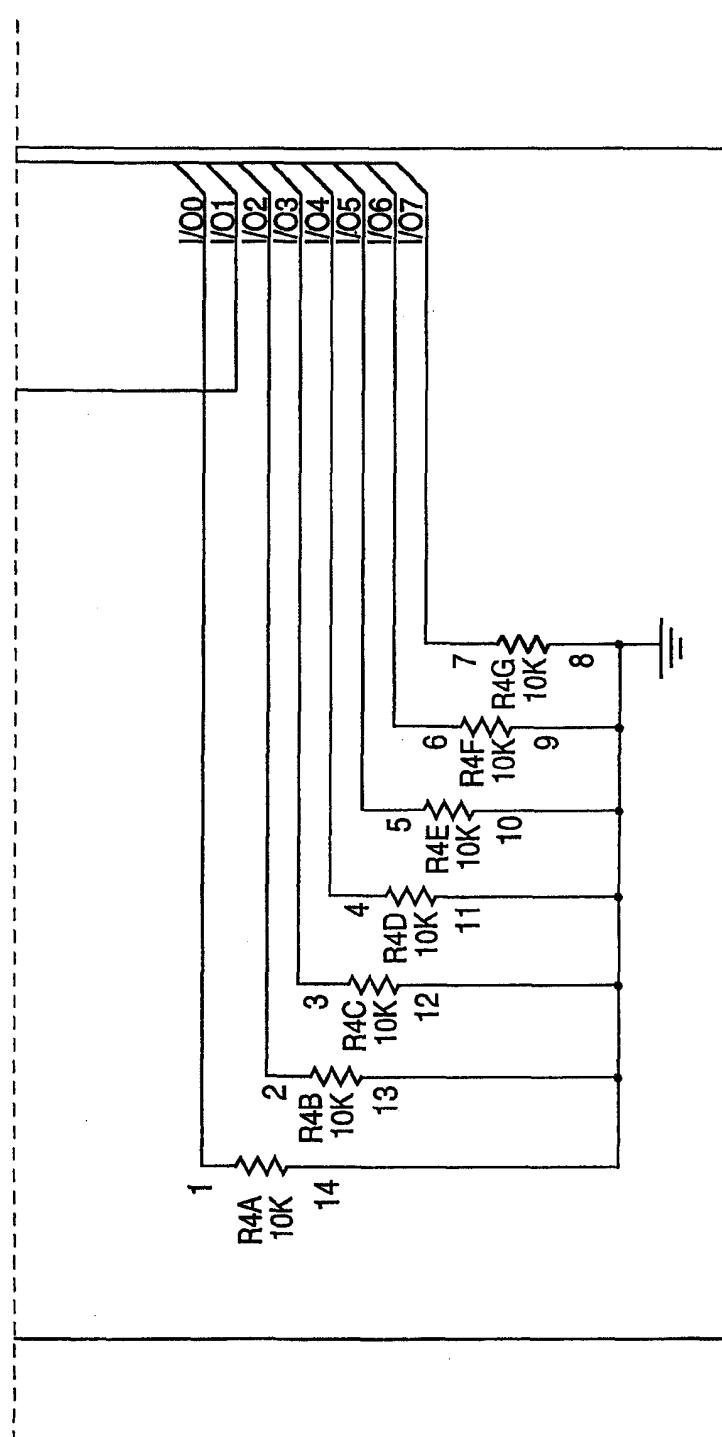


图7B