目录

[**1.** **引言** 1](#_Toc141173540)

[**2.** **NOR FLASH单元操作和阵列架构** 2](#_Toc141173541)

[**3.** **NOR FLASH存储器接口** 9](#_Toc141173542)

[**4.** **NOR FLASH操作简述** 15](#_Toc141173543)

1. **引言**

现今主流的非挥发存储器主要包括EEPROM、NOR FLASH和NAND FLASH。在实际应用中，主要关心这几类产品的以下特性：

* 产品容量，包括扇区大小
* 工作电压范围
* 操作速度，包括读出速度和擦写速度
* 擦写次数
* 数据保持时间
* 功耗
* 成本和价格竞争力

EEPROM的存储单元CELL大多采用浮栅结构，每个存储单元包含2个晶体管：1个选择管和一个浮栅存储管串联形成一个EEPROM的存储单元。EERPOM容量一般以K数量级来计算，而且具备优越的擦写和数据保持可靠性，其擦写次数可以达到百万次量级，数据保持时间可以达到等效100年量级。EEPROM另一个特点是改写数据灵活，可以逐字节进行改写而不影响其他存储阵列的内容。EEPROM器件一般采用并行、串行IIC和串行SPI接口。

NOR FLASH存储器的存储单元CELL来源于EEPROM，以浮栅单元为最多见，但是每个存储单元包含1个晶体管，使得其集成密度比EEPROM来的高，NOR FLASH容量一般以M数量级来计算。NOR FLASH的浮栅晶体管的源和漏极形成并联结构，类似数字电路或非门的NMOS晶体管部分的结构，因此得名NOR FLASH。NOR FLASH一般擦写次数达到10万次量级，数据保持时间达到等效10年量级。NOR FLASH是一种在可靠性和容量之间平衡较好的器件，因此在业界取得广泛的应用。由于NOR FLASH的存储阵列是同一个扇区所有单元的源和衬底都连接再一起，因此采取了扇区擦除、字节改写的方式来改写数据。NOR FLASH接口一般采用并行接口和串行SPI接口。NOR FLASH的并联存储阵列结构使得其读出放大速度较快，适合于随机地址的访问，因此NOR FLASH更适合于存放程序甚至直接在NOR FLASH中运行程序，包括跳跃地址的JUMP指令等。NOR FLASH的擦写采用高压操作，因此对器件的可靠性和寿命提出了挑战。当器件容量大于1M位后，考虑到生产良率，需要引入冗余修复，也会使器件面积有进一步增加。另外考虑到可靠性，有些NOR FLASH还会引入内嵌ECC纠检错机制，当然这也进一步提升了产品面积。

NAND FLASH的存储单元CELL的结构和NOR FLASH类似，采用浮栅晶体管的结构，但是，对NAND FLASH阵列的访问方式和NOR FLASH不同。NAND FLASH的存储单元是串联在一起的，一个NAND FLASH浮栅晶体管的源同时也是另一个NAND FLASH浮栅晶体管的漏，类似数字电路与非门的NMOS晶体管部分的结构，因此得名NAND FLASH。NAND FLASH对阵列数据的读出放大采用串行方式，工作电流小速度慢，因此不适合于随机地址的读出而更适合于地址连续的读出，因此NAND FLASH更多被应用于数据存储场合。NAND FLASH编程时栅端要加20V以上高压，擦除时衬底要加20V以上高压，这么高的电压也会对器件的可靠性和寿命提出挑战。NAND FLASH接口一般采用并行ONFI接口和DDR接口或串行SPI接口。

1. **NOR FLASH单元操作和阵列架构**

以下对NOR FLASH存储单元的操作机理进行介绍，包括读出、编程和擦除。

NOR FLASH阵列（MATRIX）是将NOR FLASH的单元以行和列的形式排列在一起，行一般称为字线WORDLINE，列一般称为位线BITLINE。多个存储单元的栅极连接在同一字线上，多个存储单元的漏极连接再位线上。一个扇区中所有存储单元MOS管的源端和衬底都连接在一起，将这种结构重复8或16次就形成了8位（字节）或者16位（字）输出的存储结构。

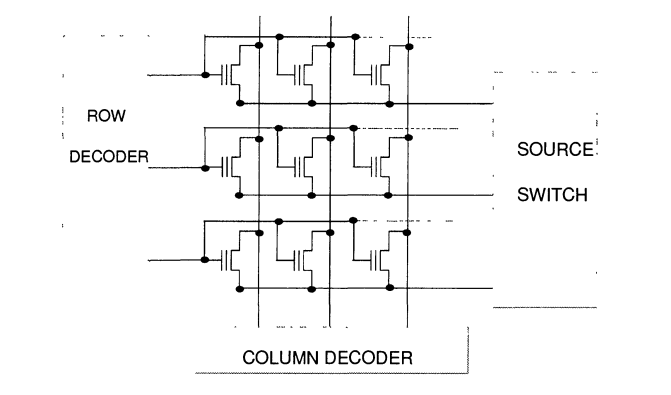


图 1 NOR FLASH结构图

读出操作是将选中的存储单元MOS管的阈值信息输出成0或1的数字值。一般存储单元采用浮栅MOS管结构，浮栅中注入电子则阈值高，读出为0；浮栅中没有电子或有空穴则阈值低，读出为1。读操作时，选中的字线被偏置在5V左右的读电压上，选中的位线被偏置在1V左右的读电压上。读出操作至少是以8位（字节）或者16位（字）同时进行的，近期产品也存在以页为单位的更多位宽的读出操作方式。存储单元读出方式有很多种，最常用的方法是通过比较器电路来比较存储单元的电流来实现，这种方式就要将被访问的存储单元去和一个参考单元进行比较，参考单元一般也是同样结构的浮栅MOS管，通过比较器电路后将最终结果转换成数字信号并输出。

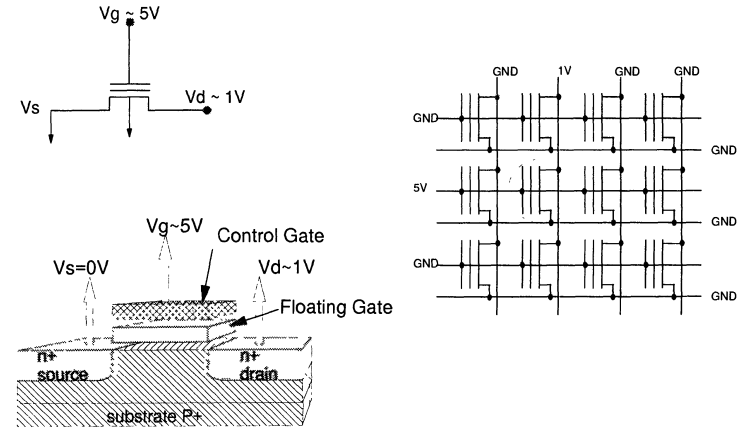


图 2 NOR FLASH读操作原理图

编程操作将电子注入浮栅并提升浮栅MOS管的阈值。编程采用沟道热电子效应（HOT-CARRIER）实现。选中的浮栅MOS管的栅端加一个10V左右的高电压，同时漏端施加一定的高压来得到热电子。漏端高压一般比VCC高，但必须谨慎设置以防止电压过高造成结击穿、沟道击穿、增益退化等危险效应。由于栅和漏高压的存在，NOR FLASH阵列在编程时产生的栅干扰和漏干扰都是要考虑的因素。NOR FLASH擦写次数是有限的，影响擦写次数的因素包括：

* 单元沟道长度、宽度和耦合比；
* 温度；
* 栅和漏的编程高压；
* 隧道氧化层质量；

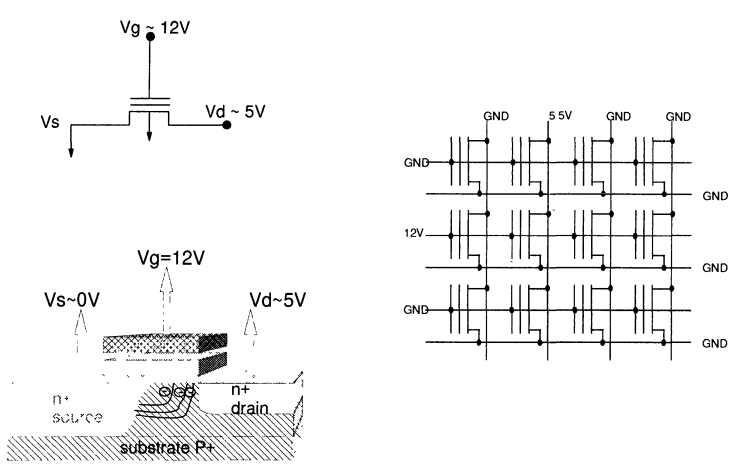


图 3 NOR FLASH编程操作原理图

就像读操作一样，编程操作也至少是以字、字节的宽度进行的，近期产品也存在以页为单位的更多位宽的编程操作方式。

擦除操作是会将电子扫出浮栅，甚至将空穴注入浮栅，从而降低浮栅MOS管的阈值。由于NOR FLASH阵列的组织架构，擦除操作是一个扇区内所有的浮栅MOS管同时进行的。擦除操作是通过FN隧穿效应（FN-TUNNELING）来实现的。FN隧穿效应发生的区域可以是源端，也可是沟道隧穿。如果采用源极隧穿，源极的PN结必须足够深来减少擦除时的衬底电流。这两种擦除方式也可称为源极擦除和栅极负压擦除。采用栅极负压擦除可以使得擦除时源极和衬底上不需要施加过高的正高压，防止PN结击穿并提高工艺的可靠性，但是为了提供负压必须设计专门的泵电路和负压通路，并要考虑负压隔离等，在一定程度上让工艺和设计复杂化。

擦除操作可以采用以下方式进行：

* 在源极施加一个正高压；
* 采用恒流源电路在源极施加一个恒定电流，使得隧道氧化层上得到一个固定的场强；
* 在源极和衬底施加一个正高压，在栅极施加一个负高压（此方法最为常用）；

扇区的擦除时间主要受到施加电压的幅度、隧道氧化层厚度等因素影响。NOR FLASH擦除中不可避免出现过擦除问题（单元阈值擦到负阈值从而影响相同位线上其他单元的读出电流），为了纠正过擦除需要对NOR FLASH擦除引入纠正算法，这也使得NOR FLASH擦除过程复杂化并降低了NOR FLASH擦写寿命。

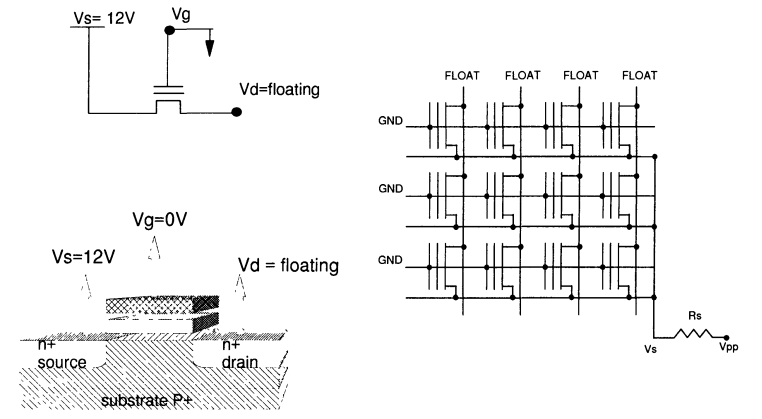


图 4 NOR FLASH源极擦除操作原理图

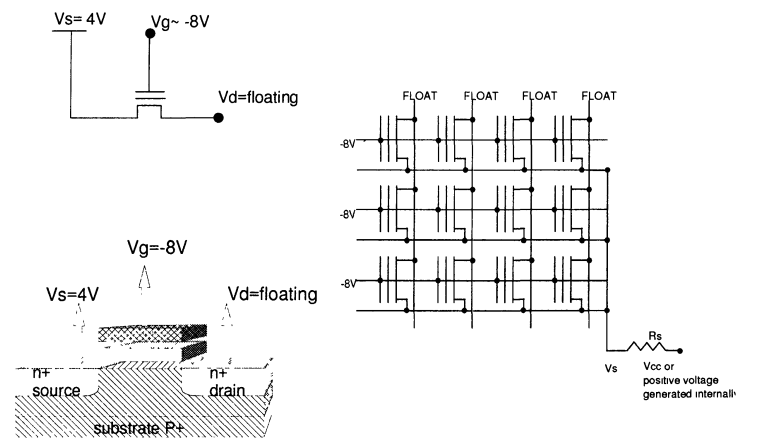


图 5 NOR FLASH栅极负压操作原理图

多电源供电器件对系统电源提出更高要求，增加了系统电源复杂度，因此只有早期的NOR FLASH器件采用多电源的方式，并且很多多电源擦写通过编程器实现。现在主流的NOR FLASH都采用单一电源，擦除编程所需要的高压由芯片内部电压泵产生，从而简化了对系统电源的要求，并使得现在的NOR FLASH可以摆脱对编程器的依赖，方便的在系统板上实现数据的改写。

对NOR FLASH操作有了初步认识后，可以进一步深入分析存储阵列结构。

通过行列译码电路，特定的存储单元会被译码选中；通过源极驱动电路，所要被擦除的单元会被加上特定的擦除电压。现在主流的器件的位宽是8位字节模式或16位字模式，甚至也有32位宽模式的器件。因此，位宽中的每一个位，存储阵列中都会有对应的列区域通过读出电路（用于转换存储单元中所存储的数据内容）和对应的输出相连，如下图所示。基于张图的架构，所有存储单元的源极都是连接在一起的，所以全片所有存储单元都是同时被擦除的。

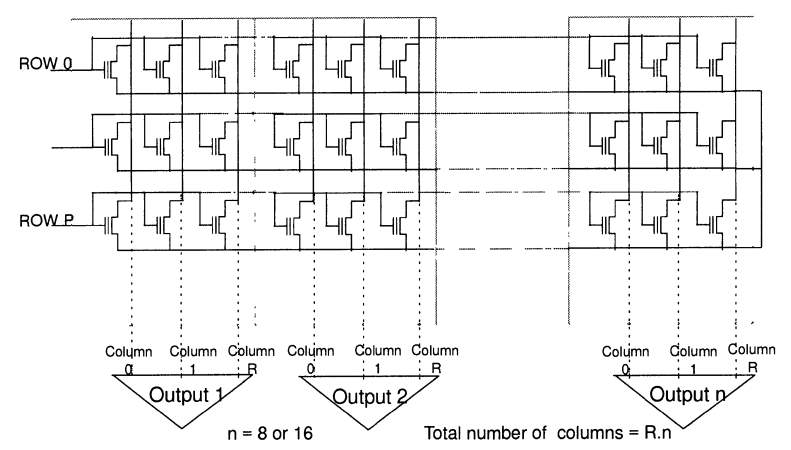


图 6 全片同时擦除的NOR FLASH架构图

在有的场合，同一个器件可以同时支持8位或16位模式（用户可以自由选择字节模式或字模式），模式的切换通常是通过BYTE#管脚来进行设置。为了实现相关功能，器件内部默认会工作在16位模式，而在输出控制时决定将高8位还是低8位进行输出，或者是将16位同时输出。

最早期的NOR FLASH只能进行全芯片同时擦除而无法对其中的某些区块进行单独的擦除，这是由于那时NOR FLASH存储阵列的架构中是所有的浮栅MOS管的源和衬底都是物理连接在一起的。然后扇区擦除概念被设计出来。整个NOR FLASH存储阵列被区分成几个独立的扇区，某一个扇区内的浮栅MOS管的源极连在一起但是和其他扇区的独立开，不同扇区的源极由源驱动电路来控制切换，从而实现对任意扇区的擦除而不擦掉其他扇区的内容。每个扇区的大小可以根据实际需求进行定义，扇区大小可以是相同或不同的。基于独立扇区的架构，可以实现BOOT扇区的功能，根据系统BOOT需求定义专门的BOOT扇区，设定专门的大小并给与特殊的写保护机制（比如有些器件WP#管脚只保护某些BOOT扇区），从而防止重要的BOOT扇区被误改写。

在有外灌VPP的场合，通常会用源极擦除的方式，因为擦除时浮栅单元源极可以获得一个高压，这样擦除时栅极就可以不用负压而是直接接到GND电位。通常NOR FLASH存储阵列的不同扇区的单元会共用字线但其漏极和源极却不会连接再一起，因此扇区的划分会通过列的方向来实现，基于这种架构结合以源极擦除的方式，也可以实现只擦除一个扇区内的部分单元。以下两张图分别从扇区维度和输出维度给出了独立扇区分布的阵列架构。

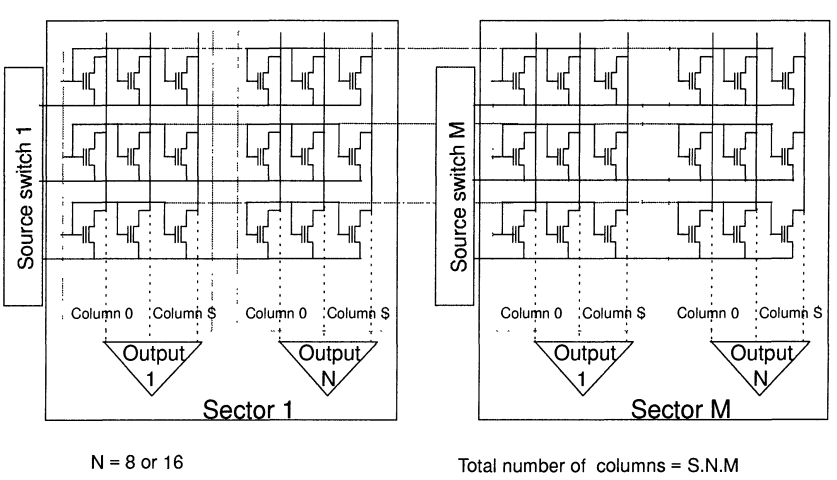


图 7 从扇区维度区分扇区架构示意图

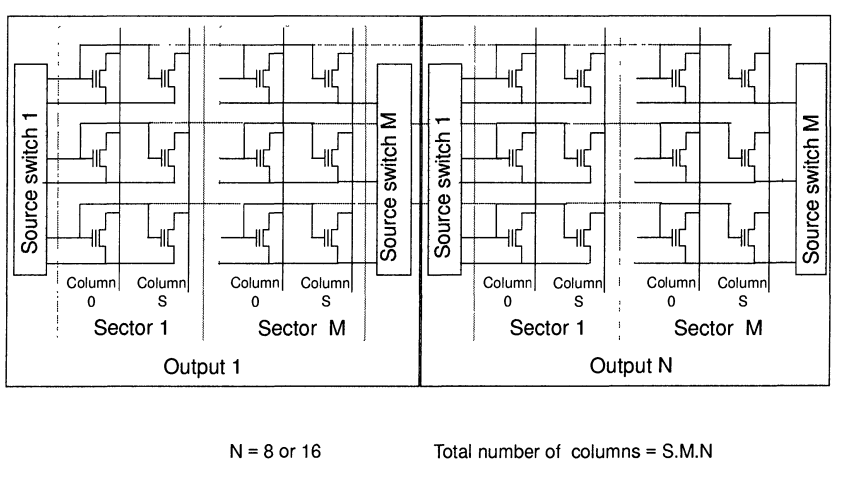


图 8 从输出维度区分扇区架构示意图

容易理解以上的架构都是双刃剑，各有利弊。扇区划分越细，则操作越灵活，但随着而来的是更多更复杂的切换电路和读出放大电路、更多的独立源极驱动线网等。

如果采用栅极负压擦除的方法，也可以将扇区在行的方向上进行划分。这种情况下如果进行擦除，要擦除的扇区的栅形成的行要被加上负压而不需要擦除的行需要连接到GND上。

从理论和技术上来说，扇区大小并没有限制。但在实际中，扇区大小会综合考虑面积、成本、性能和应用简便性，进行折中设计。这些综合因素细化来说包括：

* 将源极连线进行拆分，并增加源驱动切换电路；
* 字模式和字节模式造成的复杂连线；
* 电路复杂化以及更昂贵的冗余成本代价；
* 对擦写可靠性的影响。

单电源电压的FLASH存储器（器件上只提供一个VCC供电端）一般在行的方向上拆分扇区，如下图所示。图中阐述了位线的连接原理，不同扇区的字线完全独立，但位线有连接，不同扇区有自己独立的扇区内位线，各扇区内位线会连接到全局位线上。一般M1会用于扇区内位线而M2会用于全局位线。此时如果没有M3的帮助，字线只能通过POLY连接而无法用M3进行阻抗降低，会使得字线建立时间变长而读出速度变慢。

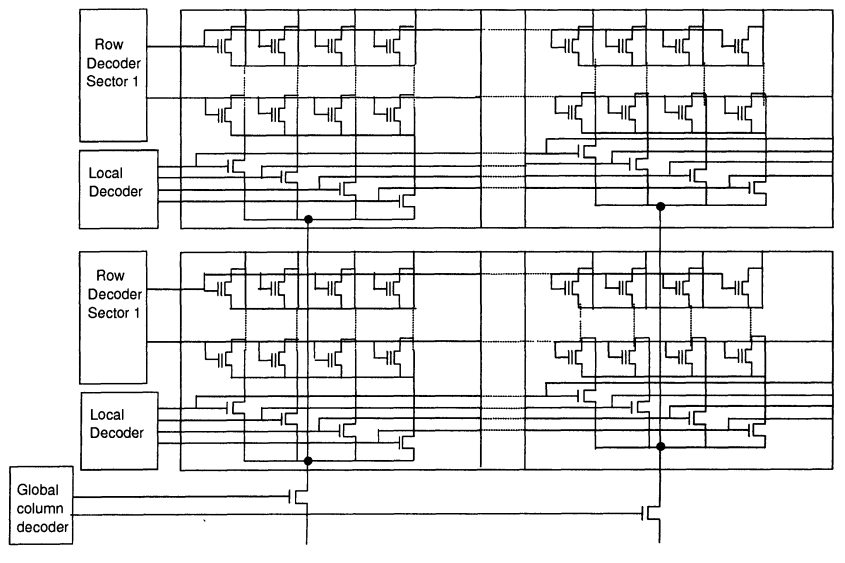


图 9 基于单电源的负压擦除FLASH阵列架构示意图

对于单电源FLASH器件来说，编程和擦除的高压都需要基于VCC进行内部产生。对于VCC低而擦写电压较高的场合，从低电压得到一个提升的高压有一定的难度。一般解决这个问题的方法是采用电压泵电路，该电路中一个重要器件是电容，后续章节会有描述。

1. **NOR FLASH存储器接口**

NOR FLASH存储器最常用的接口包括并行接口和串行SPI接口。

* 并行接口NOR FLASH

NOR FLASH并行接口一般采用三总线结构：控制总线、数据总线和地址总线，该总线接口属于通用并行接口，可以和主控器件（比如CPU或DSP）对应的总线实现简单连接。下图是一个典型的并行NOR FLASH接口示意图。

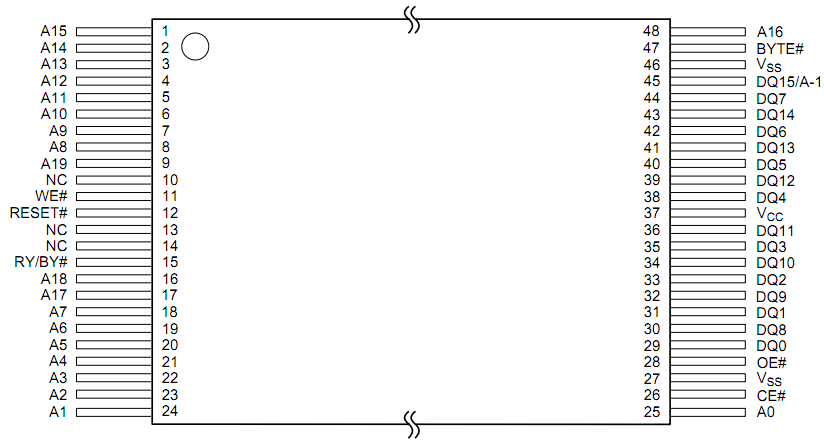


图 10 并行NOR FLASH接口示意图

并行NOR FLASH控制总线主要包括以下控制管脚：

* 片选（CE#/E#）；
* 输出使能（OE#/G#）；
* 写信号（WE#/W#）；
* 硬件复位（RESET#/RP#）；
* 字/字节控制（BYTE#）；
* 写保护（WP#）；

根据器件的不同，并行接口NOR FLASH还会有一些其他定制化的控制管脚。同时并行NOR FLASH根据容量不同拥有对应宽度的地址总线，并根据器件位宽不同拥有对应宽度的数据总线。

片选管脚用于启动存储器内的逻辑控制电路、输入缓冲电路、译码电路和读出放大电路等。#表示0使能，因此片选为高则器件不被选中同时器件的功耗急剧降低，进入待机模式。片选同时还控制了写通路，包括对写命令寄存器的写入和对存储阵列的写入。当写信号为低则启动相关写操作。命令字信息会出现在地址和数据总线上，通过片选信号的上升或下降沿被锁入器件的命令控制电路中。

输出使能管脚在读操作中控制了器件内部输出缓冲器到输出管脚的数据传输。

写信号管脚控制了将命令字写入器件命令寄存器的通路。当片选是低，写信号才能有效的控制命令字的写入。当然，片选的上升下降沿也可以控制命令字的写入。

READY/BUSY#（R/B#）管脚是一个开漏输出管脚，如果该管脚是低，表示器件内的擦写算法正在执行中（BUSY状态），此时器件无暇接受额外的命令或执行读出阵列的操作。由于是开漏结构，多个器件的R/B#管脚是可以短路在一起并共用一个外部上拉电阻。

硬件复位管脚提供了硬件复位器件的功能。当复位管脚给出下降沿，器件会启动内部复位操作。复位脉冲必须足够宽并且要避免非预期的毛刺以防止器件误复位。在擦写操作以外的状态，器件会马上响应硬件复位。在擦写操作中，不同器件设计会不一样，有的器件会设计成在擦写操作中不响应硬件复位，但有些器件会设计成即使擦写中也会马上响应硬件复位并立即中断擦写操作。硬件复位管脚的另一个作用是防止在系统复位期间产生的非预想写命令操作被误写入器件的命令寄存器中。

写保护管脚对特定的扇区的写操作提供了保护功能。一般写保护信号为高，所有扇区都能擦写；写保护为低，特定的扇区无法接受擦写命令。

字/字节控制管脚对输出管脚的位宽实施控制，将器件设定在8位的字节模式或16位的字模式。如果该管脚为低，器件进入8位的字节模式，数据进出都通过DQ0~DQ7管脚来执行，此时，DQ8~DQ15管脚会设置在高阻状态，而数据中的高8位或低8位需要一个额外的选择管脚来选择，很多器件用DQ15管脚来作为这个额外的选择管脚，也称之为A-1。如果字/字节控制管脚为高，器件进入16位的字模式，数据进出都通过DQ0~DQ15管脚来执行。

早期的并行NOR FLASH通过在管脚上施加特定电压组合，可以使器件进入特定的工作状态，包括：

* 读器件ID或电子签名；
* 临时解保护BOOT扇区；
* 扇区保护或解保护；
* 输出禁止；
* 器件待机；
* 器件复位；
* 命令写入。

如果要对NOR FLASH器件进行更为复杂的操作（比如擦写操作），需要对器件进行一系列复杂的算法控制，早期的系统通过外部微处理器来实现这一操作。而现在的NOR FLASH则通过写入一串复杂的命令码来启动器件内嵌的算法控制电路，相关操作由内嵌算法控制电路自动实现。可以实现的操作包括：

* 读器件ID或电子签名；
* 编程设置；
* 全片擦除设置和启动确认；
* 扇区擦除设置和启动确认，此时需要给出需要擦除的扇区地址；
* 擦除挂起和恢复；
* 读存储阵列数据；

FLASH器件上电会自动进入读存储阵列的模式，从而防止非预期的命令对存储阵列内的数据进行误改写。

命令由一系列二进制编码组成，分别给到地址总线和数据总线上并结合片选和写信号管脚用特定的时序写入命令寄存器，此时片选和写信号管脚也可以理解为写命令用的时钟信号。

下图是一个并行NOR FLASH器件读出时序图。

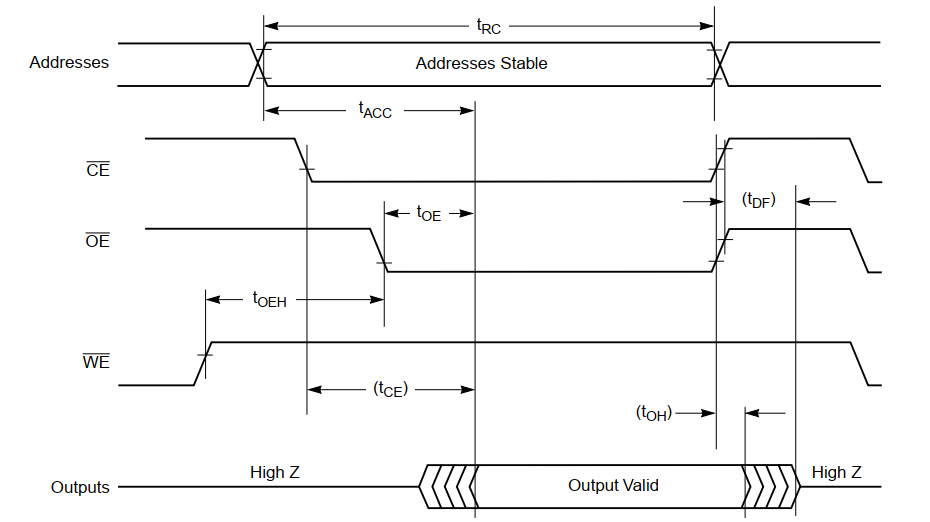


图 11 并行NOR FLASH读出时序图

下图是一个并行NOR FLASH器件编程时序图。

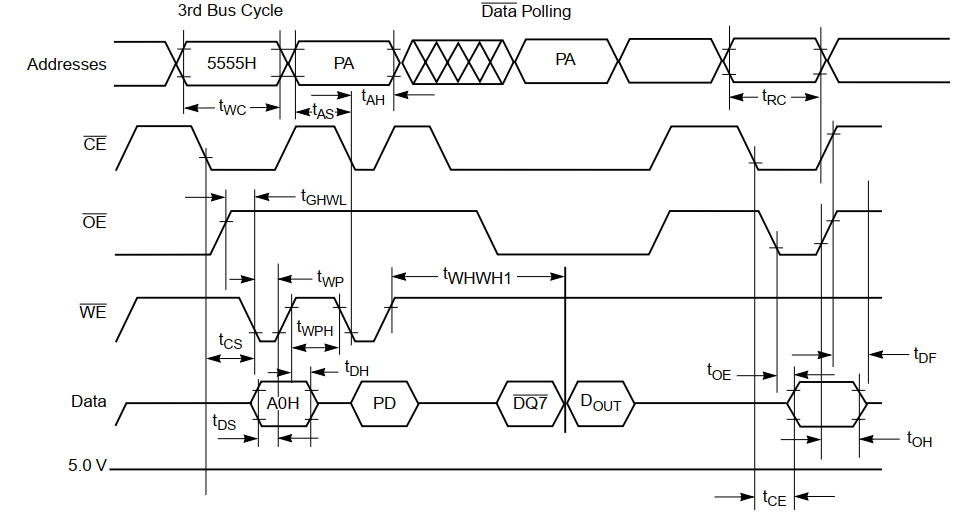


图 12 并行NOR FLASH编程时序图

下图是一个并行NOR FLASH器件擦除时序图。

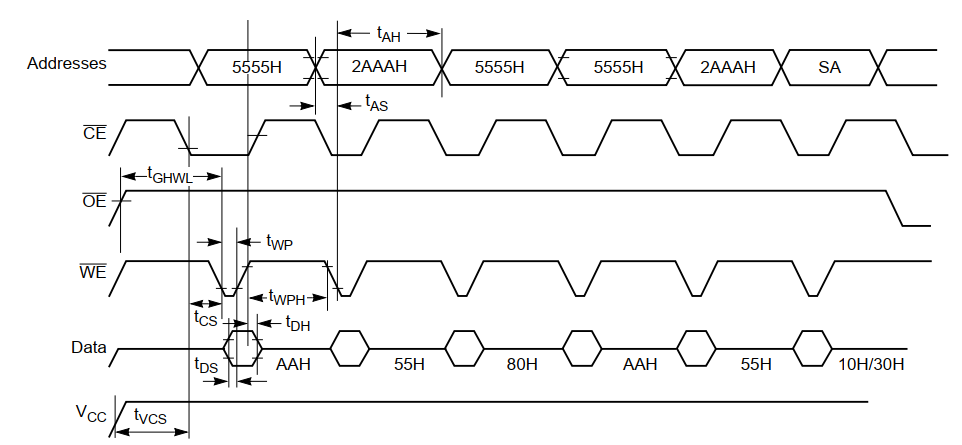


图 13 并行NOR FLASH擦除时序图

* 串行接口NOR FLASH

先进主流的NOR FLASH的另一个主要接口是SPI接口。并行接口NOR FLASH属于异步器件，管脚众多，而且随着容量增加，地址管脚也要增加。但SPI接口的NOR FLASH属于同步器件，管脚精简，所有地址和数据都从几个固定的串行管脚进出，并有CLK管脚作为同步，容量从1M位~2G位管脚管脚完全一样，具备小型化、低成本、可扩展性强的特点，成为现在NOR FLASH器件市场上最主流的接口。

SPI接口包括以下主要管脚：

* 片选（CS#）：用于器件的选择；
* 时钟（CLK）：时钟同步信号；
* 输入管脚（DI）：命令码、地址码、数据码都从这个管脚输入；
* 输出管脚（DO）：数据信息从这个管脚输出；
* 写保护（WP#）：一般是对特定的寄存器的写保护而不是对扇区的写保护；
* 维持（HOLD#）：器件歇息，歇息期间不接受SPI命令。

下图是一个SPI NOR FLASH的接口示意图。

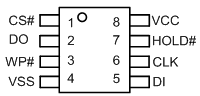


图 14 串行SPI NOR FLASH接口示意图

下图是一个SPI NOR FLASH器件读出时序图。

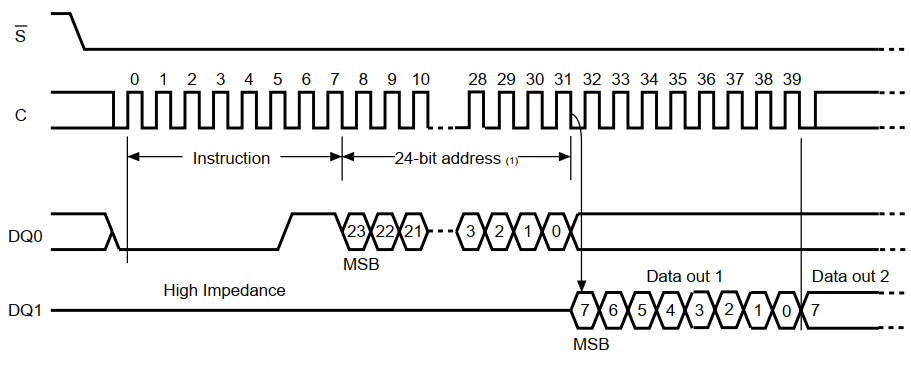


图 15 SPI NOR FLASH读出时序图

下图是一个SPI NOR FLASH器件编程时序图。

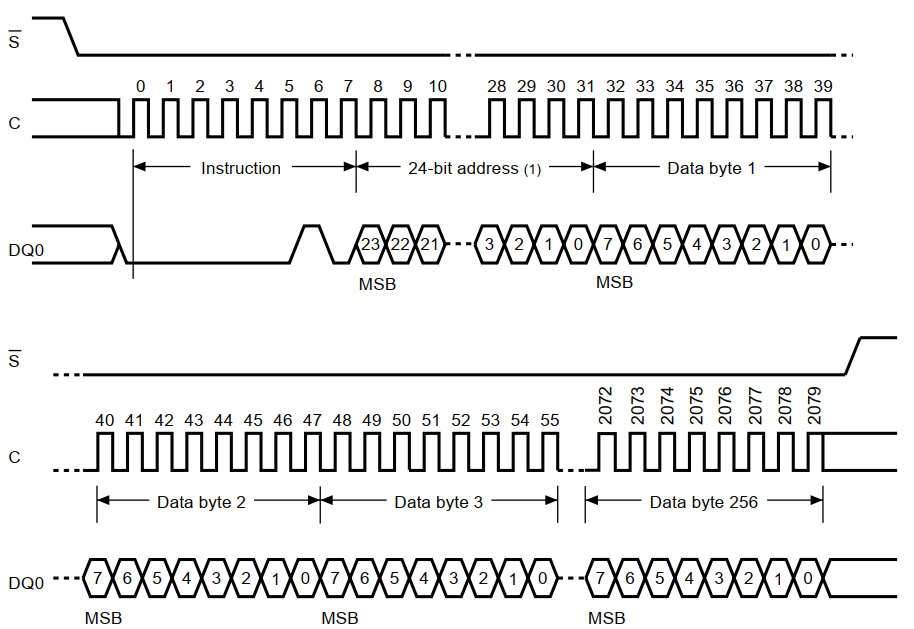


图 16 SPI NOR FLASH编程时序图

下图是一个SPI NOR FLASH器件擦除时序图。

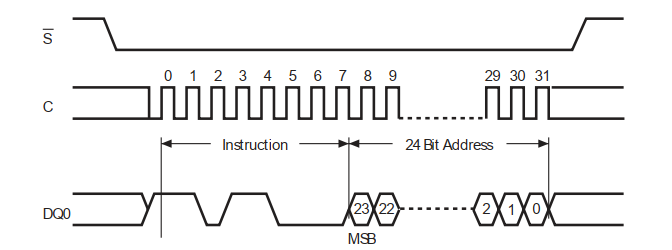


图 17 SPI NOR FLASH擦除时序图

1. **NOR FLASH操作简述**

NOR FLASH产品的内部模块架构都大同小异，但是根据工艺的不同，具体模块的线路实现会有一定的差异。另外不同的产品在DC参数、AC参数和可靠性参数上有所差异。

如前文所述，在双电源的NOR FLASH器件中擦写操作必须依赖于VPP管脚的高压输入，而单电源的NOR FLASH器件擦写由器件内部自己产生高压。这两种不同形式的器件内部线路的设计方式自然也不同。

下图是一个并行NOR FLASH器件内部的架构图。

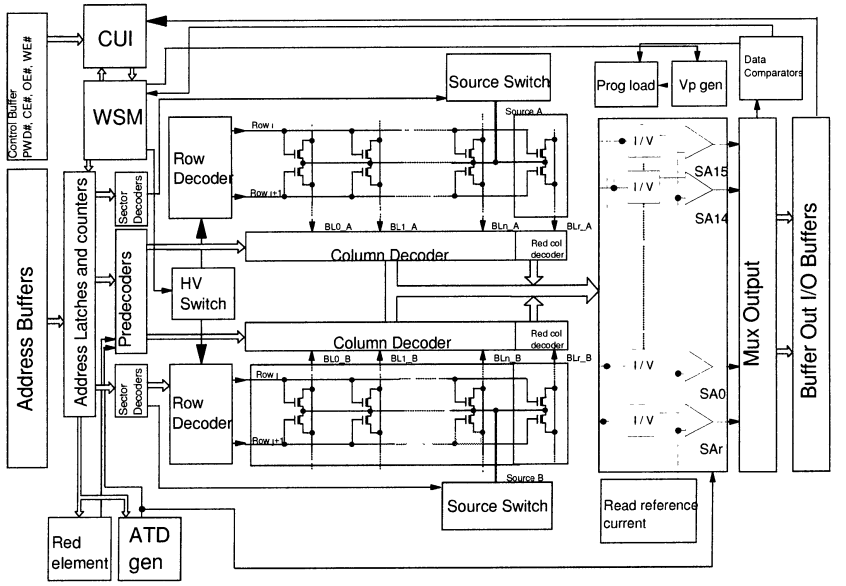


图 18 并行NOR FLASH内部架构图

以下对主要模块的线路原理其工作原理进行简述。

* 读出通路简介

根据每个FLASH浮栅MOS管所存数据的位数，可以区分成两个种类：标准存储STANDARD和多电平存储MULTILEVEL。前者每个浮栅MOS管存储1位数据，后者每个浮栅MOS管存储2n位数据。标准存储即每MOS管存1位数据一般称为SLC，多电平存储MULTILEVEL中每MOS管存2位数据的一般称为MLC，每MOS管存3位数据一般称为TLC，每MOS管存4位数据一般称为QLC。多电平存储技术中制造工艺能力起了决定性作用，这是由于多电平存储需要将编程后的阈值压缩到一个很窄的范围，这对于工艺中存储单元本身的性能和操作电路复杂度提出了很高的挑战。

下图显示了SLC和MLC的浮栅MOS管阈值分布图，其中也包含了区分数据的电平判据。

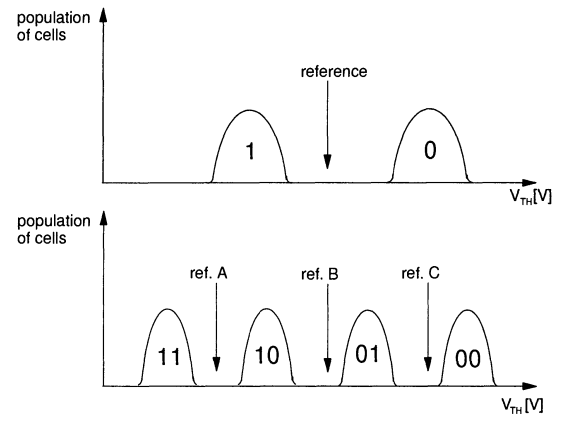


图 19 不同种类FLASH阈值分布图

无论每个MOS管存储多少位数据，都需要对存储内容的阈值分布进行区分并解析出最终数字数据，实现这个功能的电路称为读出电路。

对于读出来说，首先要使器件获取需要输出的数据的地址信息，地址通过输入缓冲器输入器件，一般要求输入缓冲器有一定的抗干扰能力，标准的施密特电路等都可以实现相关功能。输入缓冲器一般有TTL电平兼容和CMOS电平兼容特性，有的特殊PAD还有3电平特性，也就是除了标准的高电平区分和低电平区分外，还有一个高电压电平区分能力，即判断管脚是否输入了一个比VCC高2倍甚至3倍以上电压，这个功能一般用来使得器件从标准的用户模式进入特殊的测试模式。

输入缓冲器后级将驱动译码电路。译码电路一般会分成多级，包括扇区级、行级、列级，译码电路的目的是将所要读出的FLASH单元偏置到对应的读电压上。一般8个、16个或一个页的单元会同时被选中并通过多级列译码电路进行输出，当然漏极电压也要加到对应的存储单元上。

为了得到每个存储单元阈值大小，一般会通过比较器电路将存储阵列内的单元和一个已知参考单元的阈值进行差分放大。差分放大比较器通常会通过电流比较来进行，采用的机理是浮栅MOS管阈值不同则通过单元的电流也会不同。差分放大比较器会将存储单元和参考单元的电流切换成电压，然后将两个电压通过差分电路进行比较并输出数字信号，如下图所示。要注意的是存储单元和参考单元的设计要匹配，以保证工艺加工不均匀情况下比较器两边都产生共模变化，从而保证最终放大比较的效果。

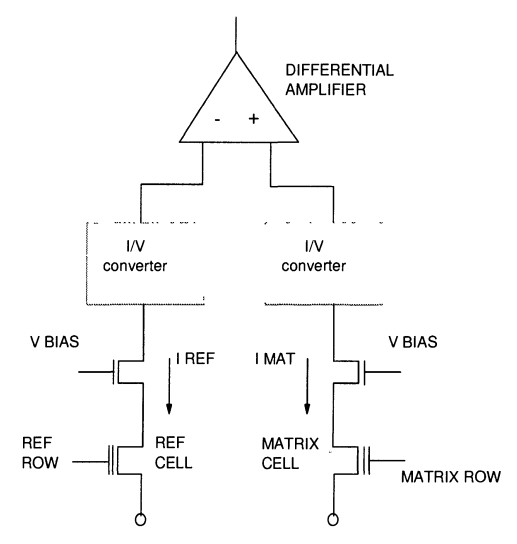


图 20 差分放大比较器原理图

在栅极相同偏压的情况下，被编程的阵列存储单元MOS管的电流会比参考单元MOS管的小，因此比较器中从阵列单元端获得的电压会比参考单元端的高。如果阵列单元处于本征态或擦除态，单元MOS管中的电流将和参考单元的基本一致。线路设计中可以进行失衡设计从而使得存储单元处于本征态时其等效电流来的比参考单元端的大从而使得存储单元更加表现为本征。

读出放大的结果以电压形式通过多路开关输出给输出缓冲器，从而驱动器件的PAD端给出相应的外部数字信号。考虑到多个数据线同时驱动外部电容所需要的瞬态电流，一些常用的抗干扰线路可以帮助降低VCC和GND端的噪声。

整个读操作主要要解决的问题是要降低从地址输入到数据吐出这整个过程中的延迟时间，从而使器件获得最好的读速度特性。

* 编程通路简介

NOR FLASH可以进行字节编程或字编程，现在的一些产品还支持页编程操作。编程操作是一个复杂而且缓慢的操作，通常器件内的擦写状态控制器对整个操作进行控制。用户只需要给出正确的指令来启动编程算法即可。编程指令一般包括一系列的不同命令组合，其中还要包含要编程的地址和数据信息。器件接受完命令序列后自动启动后续的编程算法对存储单元进行编程操作，此时器件进入忙状态而且不可读出阵列数据，用户可采用外部主控器件通过FLASH器件的RYBY脚或者DQ0~DQ7数据位来检查器件的状态（检查器件是否处于忙状态的形式包括通过查询轮询位或翻转位或状态寄存器）。有些器件提供了擦写挂起功能，在器件忙时能临时暂停擦写算法来读出非擦写阵列的数据。

编程算法启动时，要被编程的地址的阵列原始数据会和要编入的数据进行比较，如果编程数据合法（对于FLASH来说，只有编程操作能将数据1改写成0，只有擦除操作能把数据从0改写成1，如果编程操作中存在要将数据0改写成1，被视为非法），内部高压会通过高压切换电路施加到要编程单元所在的字线上，同时要编程单元的位线也就是漏极会被加上一个编程高压脉冲。编程脉冲结束后，会启动编程校验，将编程后阵列单元内的数据和所要编程的数据（要编程的数据一开始就会被锁存）进行比较，如果比较结果不一致，会在没有被编程掉的单元的上再次施加一次编程脉冲。下图给出了一个编程操作中主要信号的工作流程图。随着存储单元阈值的提升，编程的效率也会下降，这使得编程后阈值电压形成了一个自收敛效应，不同单元即使起始阈值电压不同，最后编程完毕后的最终阈值也相差无几。

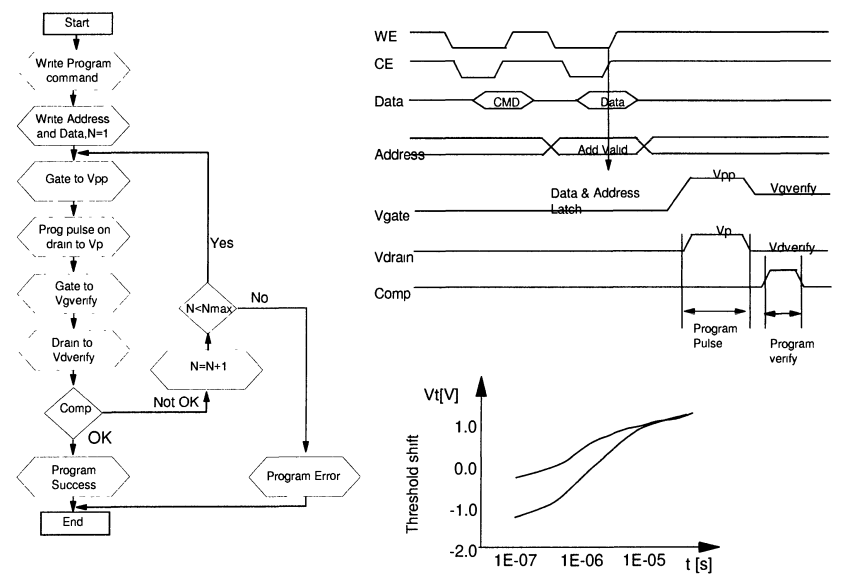


图 21 编程操作流程图

* 擦除通路简介

FLASH阵列中同一个扇区内所有单元是被同时擦除的，这是由FLASH阵列架构中同一个扇区内单元共用源极和衬底的物理架构所决定的。擦除操作可以分为以下几个主要步骤：

* 预编程全0；
* 擦除；
* 过擦除纠正；

从以上步骤可见，NOR FLASH擦除操作复杂度会比编程高得多。首先扇区所有的存储位都要被预编程到数据0以保证后续擦除所有位都有相对一致的阈值起点，为了达到全部扇区编程的目的，器件内部会设计一个地址遍历计数器来遍历扇区内的地址。第2步就需要通过源极切换电路，将要擦除扇区所有单元的源极（有时包含衬底）加上高压，这个高压来自于外部的VPP管脚或由器件内部产生。此时要擦除的所有单元的栅极保持GND电位或施加一个负电压。擦除过程中，所擦除单元的漏极一般保持浮空。擦除脉冲结束后，要对每个单元进行擦除后校验以确认所有单元都被擦除到足够低的阈值，如果校验失败，需要继续施加擦除脉冲直到最后所有单元都擦除到位。第3步是进行一个过擦除校验来判断是否存在过擦除的位，过擦除是指将存储单元阈值擦除到负阈值以至于位线上一直存在一路关不掉的电流影响位线上其他单元的读出。如果发现存在过擦除现象，需要启动特殊的回编算法将这些单元的阈值纠正到0V以上。下图给出了一个擦除操作中主要信号的工作流程图。

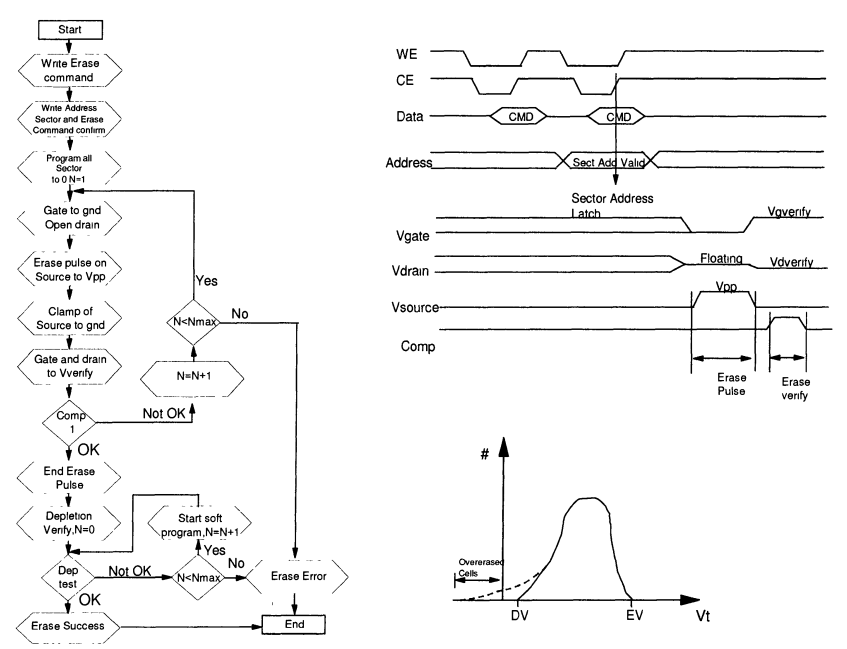


图 22 擦除操作流程图

5.1读取路径：解码

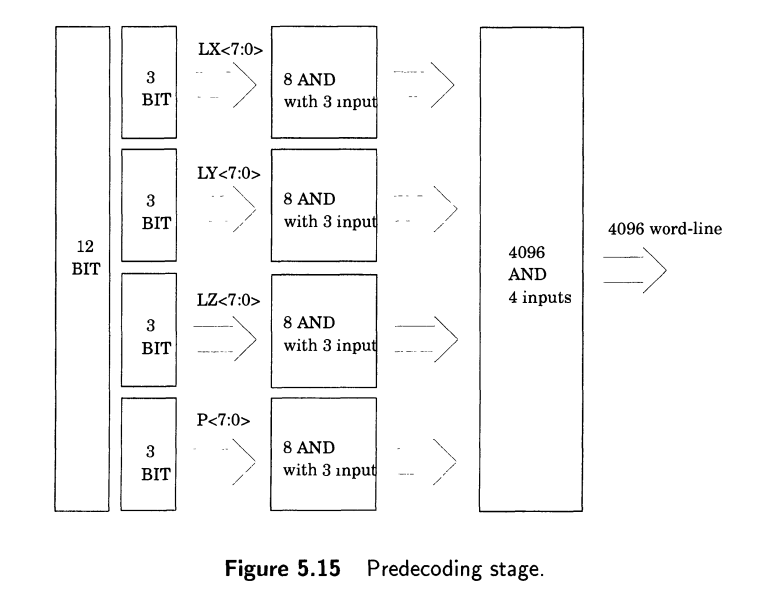
就信息存储或检索而言，有必要设计一种方法来处理存储器阵列内的数据。如第5.1.2节所述，考虑了NOR矩阵组织：闪存单元通过单独的地址解码电路选择的行和列中排列.每两行相邻行内的细胞都是镜像的，因此每两个单元共用源线。单元控制栅在多晶硅中实现，排列成条形，称为字线;单元漏极称为位线，由条形金属接触实现。

在这些假设下，解码意味着选择单行和单列以寻址矩阵内的所需数据：外部提供的地址，即n个二进制数字，必须转换为相应的2^n个二进制数字数据，其中一次只有一个位是打开的。此外，在非易失性存储器的情况下，解码电路还应能够处理擦除、编程和读取存储单元所需的所有不同偏置电压。

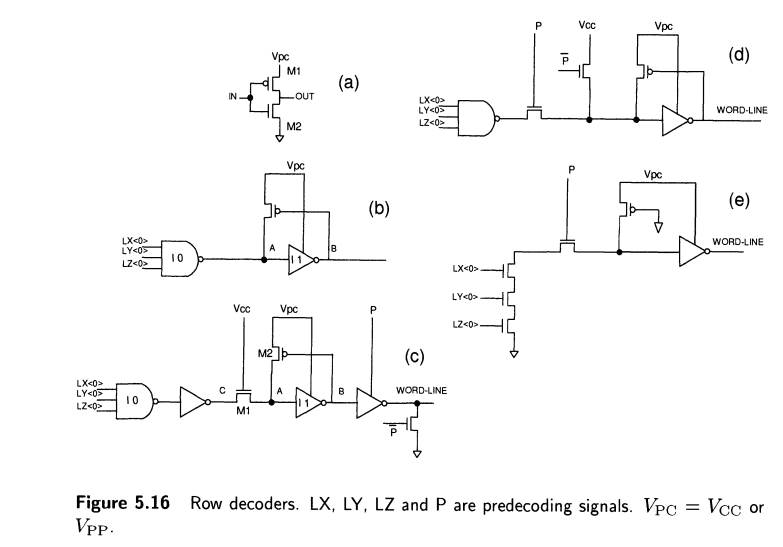
在下面的段落中，为了简单起见，整个主题分为两个独立的部分：“预解码”，即根据地址选择行和列的电路，以及“解码”，即其任务是偏置单元栅极和漏极的电路。

5.2.1 预解码

原则上，考虑由n行组成的闪存，可以使用2^n个AND门来实现预解码，其输入是n个地址的可能组合;当然，这样的解决方案是不可接受的，因为当达到Mbits的密度时，所需的面积和布局相当复杂。



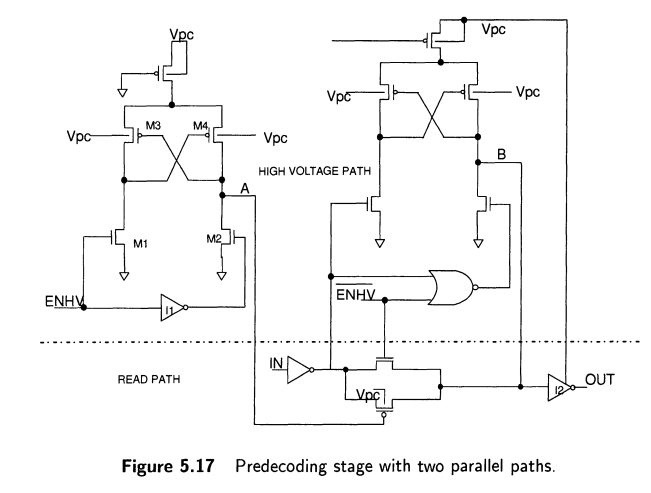
因此，设计了一种分层结构，其中地址位被分组为几个子集并单独处理。 在图5.15中显示了一个示例，其中地址宽度为12;使用32个3输入AND加上2^12个4输入AND而不是2^12个12输入AND来实现预解码。

5.2.2 行解码器

行解码器根据应用的编码地址对内存数组中的各个行进行寻址。行解码器的基本方案可以由组合电路控制的多个反向器（每行一个）表示，该电路接收输入地址并驱动反向器，以便一次只有一个反向器呈现高输出（见图5.16a）。上述简化的布置在读取模式下正确工作，其中组合电路和反向器都将读取电压VCC作为高逻辑电平，但在编程模式下，其中组合电路在未选择的行反向器的输入IN处提供读取电压VCC作为高逻辑电平，而电源线VPC在编程时，电压VPP>VCC。因此，反向器上拉晶体管的栅极和源端之间存在压降，当它达到阈值电压时，Ml被打开：输出OUT以这种方式没有达到所需的零电压，导致对连接的单元施加了应力并没有确保输出端的正确逻辑电平。

一种可能的解决方案是使用正反馈反向器，其PMOS反馈晶体管连接在VPC和输入A之间，栅极连接到输出B（图5.16b）。因此，当输出B处的电压下降时，反馈晶体管导通并将节点A连接到编程电压VPC，从而确保反向器I1的上拉晶体管完全关断和零输出电压。但是，上述解决方案存在一些缺点。首先，由于反向器的输出被带回来，出现了布局问题;通过用单独的信号驱动反馈晶体管来解决问题，反过来又会在同步方面产生问题。其次，NAND栅极IO的PMOS晶体管的漏极-衬底结的直接偏置出现了问题，这将使源区和衬底偏置在VCC处，漏极区域（连接到输出）偏置在VPP处。

一种可能的解决方案是提供一个NMOS调整管，将低压（预解码）部分与高压（实际解码）部分分开。这样的解决方案如图5.16c所示，其中3输入AND，以VCC的读取电压供电并构成组合电路的一部分，用于选择行，通过NMOS传输晶体管Ml驱动反向器I1，栅极端偏置在VCC。当节点C处于VCC时，传递晶体管Ml通过在相同电压下呈现两个端口（栅极端口和连接到C的端口）来作为二极管工作，因此在节点C和节点A之间引起等于其阈值电压的压降（加上体效应）。

除了使电路进一步复杂化外，解决方案如图5.16c所示。在低电源电压下也不令人满意：传输晶体管Ml两端的压降阻止节点A达到确保I1的上拉晶体管完全关闭所需的高电压。5.16d和5.16e包含其他常用的解决方案;基本原理是一样的。此外，除了将不需要的偏置问题转移到电路的其他部分之外，CMOS通过开关体积太大，无法适应解码阶段，解码阶段是在阵列行之间的间距内形成的。因此可以实现另一种类型的解码器;即使在低电源电压下，它也可以在读取和编程中正常工作，并且在布局或同步方面没有问题。

在行解码器中，高（编程）电压不仅提供给最终反相级，而且还提供给预解码级：为此，预解码级提供两条并行路径，一条以低电压供电并在读取模式下使用，另一条以高电压供电并以编程模式使用（图5.17）。

CMOS开关将两条路径分开，它由预解码阶段已经可用的高电压驱动，并且在预解码形成时，不涉及最终解码阶段带来的整合问题，因为预解码器位于矩阵之外。

如果逻辑信号ENHV为低电平，则节点A也为低电平，所以passCMOS是开的，从而允许信号IN进入OUT。在此情况下，由于内存处于读取模式，节点VPC保持在 VCC。为了让电压VPC（不等于VCC）到达行解码器，ENHV在VCC处偏置;然后节点A通过由Ml，M2，M3，M4和I1（电平移位器）组成的反馈结构，被带到VPC并关闭passCMOS。

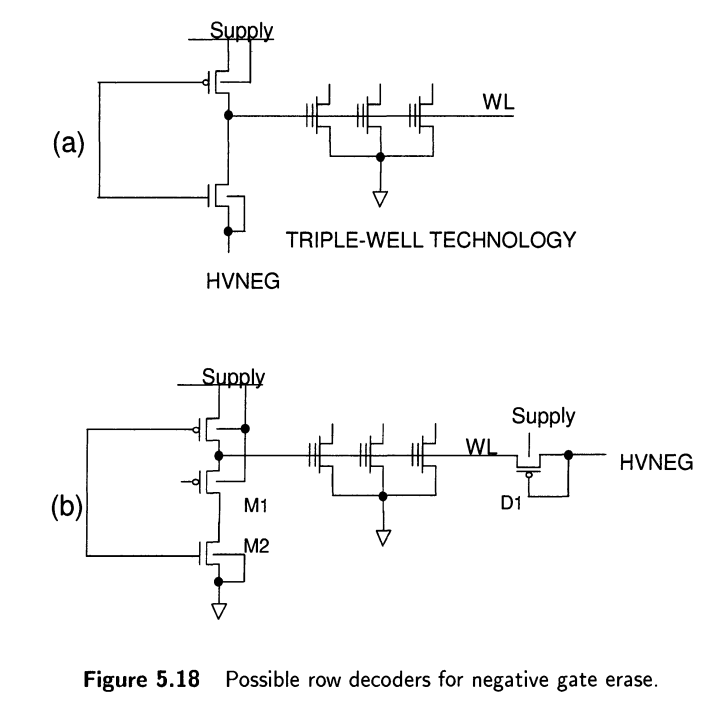
通过另一个电平转换器，信号IN能够在CMOS逻辑中驱动反向器I2。所描述的电路的优点如下：即使在低电压下也能正常工作，因为它没有NMOS传输晶体管或其他涉及端口压降的元件。它通过电源分支来确保行驱动信号的传输路径上的组件关闭或完全打开，从而确保正确的输出数据，并防止对连接到未选定行的单元施加压力。在预解码电路中形成高压路径使得能够使用CMOS pass开关，尽管没有压降，但需要高压驱动电路才能正常工作，并且体积过大，无法容纳在最终的解码电路中。尽管在预解码阶段使用了许多高压元件，但由于解码器不需要任何反馈分支，因此简化了其布局。

最后，通过在预解码电路中形成两条独立的路径，低压路径可以非常简单（仅仅是一条连接线）来大大减少访问时间，并且通过使用电压移位器可以优化高压路径，尽管它们在信号传播中引入了轻微的延迟（它是一种反馈结构），在读取性能方面没有影响。

在分析了连接VCC偏置和VPP偏置器件的电路后，可以更深入地了解行解码器最终反向器的结构。该电路的设计与擦除类型和可用技术密切相关（这些概念将在第5.62节中再次讨论。为了电擦除闪存单元，有必要在单元的源极和栅极之间施加适当的电压偏置（通常为12V）。

当栅极保持在GND并且整个擦除电压施加在扇区的源上时，就会发生“源擦除”。在这种情况下，最终的反向器只是一个CMOS反向器。如果需要“负栅极擦除”（即栅极为负，通常为-8V，源为正，通常为5V），简单的CMOS反向器不再适用。

如果可以使用三阱技术，即如果可以实现其B端与芯片基板分开偏置的nMOS晶体管，则可以将负电压直接通过最终反相器传输（图5.18a）。否则，有必要将最终反向器中的nMOS与字线隔离，以避免当对字线本身施加负电压时漏极/p阱结的直接偏置。在这种情况下，负电压HVNEG通过绝缘二极管DI提供给单元的控制栅极，M2通过Ml进行保护（图5.18b）。



5.2.3列解码器

一旦选择好了行，为了选中所需的存储单元，就必须对列进行解码。如果阵列的列宽为m，输出由b位组成，因为一次会访问b个位，所以必须具有m/b个互斥解码信号。预解码和解码之间进行同样的划分：预解码是一个层次化的组合逻辑电路，组成了图5.19中YM和YN解码的一级信号。解码器由nMOS晶体管多路复用开关组成，每个输出都是重复结构。列解码器的任务是传递适当的偏置电压到存储单元的漏极上。YM和YN解码器由VPCY供电：在读模式下，这个电压等于*V*CC，而在编程模式下，此电压升高至*V*PP，因为这个电路为要编程的单元提供合适的电压（通常 ≈ 6V）。连接*V*CC和VPCY的设计方案与行解码器相同。

5.2.4分级解码器

如第5.1.2节所述，非易失性存储器中的扇区可以按行或列划分。在后一种情况下，字线对于每个扇区都是通用的，扇区维度是根据其中包含的列数来选择的（图5.20）。扇区定义为共享同一源节点的一组单元，即同时被擦除的单元。使用这种结构，位线的寄生电容受到限制，并且这种解决方案有利于灵敏放大器。

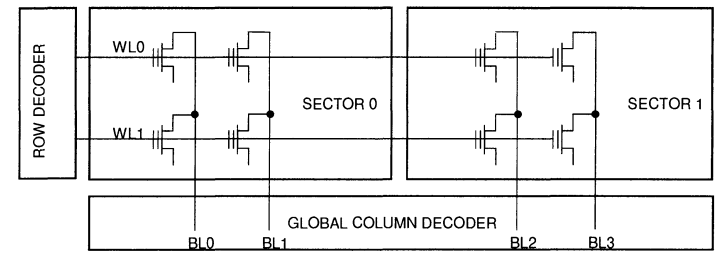


图 5.20扇区按列划分

不幸的是，这个方案有几个缺点：首先，连接同一行上所有单元控制栅的字线采用多晶硅实现。从电学角度来看，它可以被认为是一个分布的RC网络：寄生电阻取决于多晶硅，而寄生电容取决于单元的栅接触。即使考虑少量的单元（例如1000个），与此网络相关的时间常数也是几纳秒。这个延迟直接影响访问时间，因为它会导致所选单元施加正确的偏置电压有延迟。考虑双金属工艺流程，位线采用一层金属，而字线采用另一层金属短路，从而降低寄生电阻和相关的时间常数。

共享字线方法还有另一个问题，每次寻址一个单元，同一字线上的所有其他单元都受到所谓的栅极应力的影响，这可能会在设备寿命期间影响单元的正确功能。此外，在栅极加负压擦除的情况下，共享字线方法不是最佳选择，因为正压差也应用于扇区内不被擦除的单元。解决这些问题的一种方法是选择行划分扇区，尤其是与栅极压力相关。在这种情况下，所有扇区共享的是位线，影响扇区尺寸的是行数。实际上，为了避免漏极应力，不可能只共享一条位线；每个扇区都有一组位线，称为局部位线：每个位线通过一个传输晶体管连接到另一个称为主位线的金属层，如图5.21所示。

每个扇区都有一组局部的传输晶体管，这些晶体管仅用于寻址的扇区，从而避免对其他扇区的单元产生漏极应力效应。由于存储单元高度集成，局部和主位线通常使用两种金属层进行设计；如果没有第三种金属层，字线不可能采用金属层短路，这个问题对访问时间影响很大：因此必须使用具有低电阻多晶硅的技术。按行组织方法对列解码器的结构有相当大的影响，因为它要求每个扇区都有一个局部解码器，其对面积的影响是不可忽略的。另一方面，消除存储单元上的任何应力都会大大提高可靠性。



图 5.21扇区按行划分

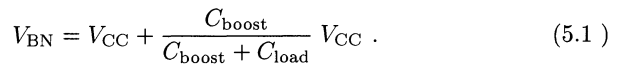
5.2.5低*V*cc问题

如前所述，读非易失性存储器的单元是通过在其栅极加读电压偏置并检测流过单元的电流。如果单元被编程，其阈值电压必须高于读取电压，以便电池不会产生电流；如果单元被擦除，其阈值电压必须能够让电流流过单元；检测电流可以区分已编程和已擦除的单元。为了确保存储阵列的正确读取和可靠的擦写循环（多周期操作），单元阈值电压的分布必须遵守某些限制。更具体地说，目前使用的技术要求最佳擦除单元的阈值高于零，最差擦除单元的阈值电压约为2.5V。下限主要是由于要防止耗尽单元（阈值电压低于零的单元）导致读取错误；而上限是由单元阈值由固有分布决定，这个分布由所使用的制造技术决定。由于读取电压通常与电源电压一致，因此超过3V的电源电压不会带来任何问题。在低*V*CC下工作的存储器会出现问题。事实上，当电源电压*V*CC为2.5 V时，阈值电压*V*TH接近该值的所有单元导通的电流很少或没有，该单元会被判定为编程态，从而导致读取错误。解决这个问题的方法是提高读电压，即向被读取的单元的栅极提供高于*V*CC的电压，这个电压由适当的升压级数产生。

5.2.6增压概念：连续升压和“一拍”升压

图5.22显示了升压电路的可能方案，以及所有相关信号的所需时序；BN是要升压的节点。

开始时，辅助升压电容（*C*boost）和寄生电容（*C*load）通过pMOS晶体管预充电至电源电压。当需要增压时，信号B切换到零，从而将*C*boost的下“板”电压提升到*V*CC。同时，关闭M1，节点BN被隔离。由于隔离节点上的电荷守恒，这种配置操作会产生电荷共享现象。根据基尔霍夫电压定律和能量守恒，BN上的最终电压为



基于这一原理，通常会采用两种方案实现：连续升压和“一拍”升压。在连续升压解决方案中，需要根据时钟信号提供时序电路，将升压电容逐渐充电至高于*V*CC的电压；然后，升压电容将存储器的公共（升压）线维持在所需的高电压。这种解决方案的优点在于升压电容的尺寸较小，因为通过一系列小增量可以达到高电压。然而，就是由于这个，初始充电以及打开或从待机状态重新进入工作时，内存的访问非常缓慢。为了消除后一种延迟，可以使用第二个较小的升压电路来保持升压电容在待机模式下充电，而代价仅是增加消耗。另一方面，“一拍”升压解决方案采用了一个巨大的升压电容，该电容仅在预定时间由单个脉冲充电（在读取模式下切换地址或主动切换E#信号时）。这个方案虽然解决了从待机状态（或存储器开启时）重新进入时访问速度慢以及待机模式下功耗增加的问题，“一拍”升压解决方案本身也存在其他缺点，因为其电容和必要的驱动电路需要较大的面积。

5.2.7一种新的升压方法

还有另一种称为“微升压”的解决方案，可以提供比*V*CC更高的读电压。该解决方案保持了已知脉动式和“一次性”升压解决方案的优点，同时将缺点降至最低。一种可行方案如图5.23所示。

BN是行解码器的电源节点；仅描述了一条带驱动反相器的字线。在没有箝位二极管D1的情况下，电荷共享效应将驱动节点BN上的电压达到节点BN和GND之间的（5.1）给定值，中间还存在一个寄生电容，它表示pMOS晶体管的阱电容、结电容和连接到节点BN的其他寄生电容。M1和M2代表行解码器的最终反相器。当通过M1传输到要读取的行时，上述电压将驱动寻址线至高的读电压，确保能正确读取单元，但另一方面，这将导致非寻址行上的电压偏置问题：其最终反相器的pMOS晶体管的栅极偏置于电源电压*V*CC，而源极偏置于更高的电压*V*BN。因此，非寻址行的pMOS晶体管肯定不能被关闭，从而导致节点BN的异常放电，影响电压提升功能。另一方面，箝位二极管的目的恰恰是只允许节点BN升压到电压（*V*CC+*V*d），其中*V*d是二极管上的压降。该二极管由类似于晶体管M1的pMOS晶体管实现，采用二极管连接方式，因此压降*V*d等于M1的阈值电压*V*THp。当节点BN的电压趋于超过（*V*CC +*V*THp）时，会被导电二极管钳制，从而以受控方式使升压电容向GND（地）放电。因此，驱动非寻址行的晶体管M1的源极和栅极之间的压降等于二极管D1的阈值电压，二极管D1即使接近开启，晶体管M1也保持断开状态。

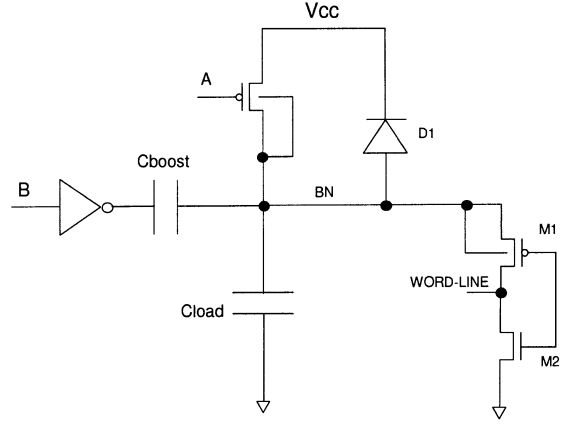


图 5.23 局部升压或“微升压”

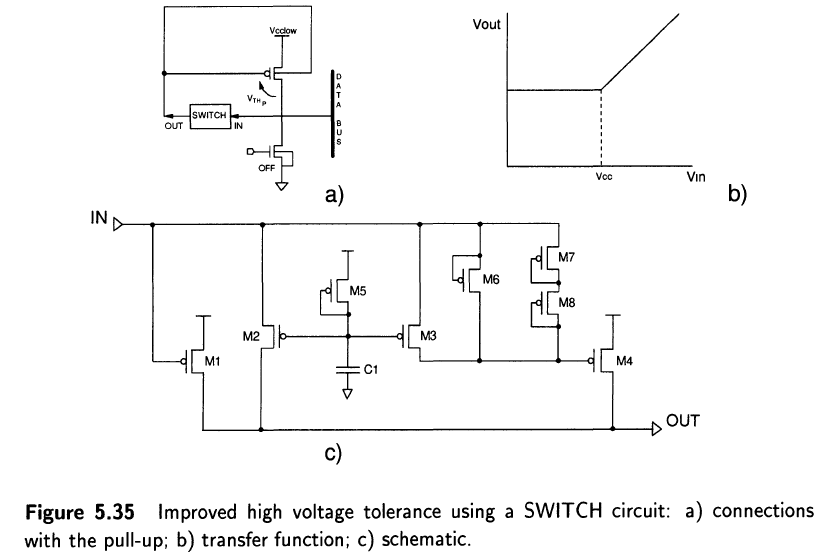
因此，该解决方案允许将寻址单元的读取电压提升到一个值，该值即使仅略高于电源电压，通常也足以确保正确读取单元，并且如果需要继续提高电压，可以通过提供本征晶体管（阈值接近0）来实现。无论如何实现升压，都必须始终要能被箝位住，以便（i）限制选定字线上所有单元上的应力，以及（ii）不降低*V*CCmax（见第5.4节）。

在读操作结束时，必须通过适当的信号恢复初始状态（首先是将BN箝位到*V*CC），以便可以在后续的每次读操作时重复上述步骤。上述的升压电路通过借助单个脉冲提高读电压来解决访问慢的问题，这通常与连续升压电路有关。此外，占用的面积较小，因为可以使用多个局部小面积的升压电容（对于每个扇区或半扇区），这些小的升压电容可以集成在芯片内未使用的区域中，或者至少可以集成在存储中优化布局。

5.4读路径:感测技术

本节介绍非易失性存储器中用于读取存储单元信息的主要结构，即EPROM和Flash。涵盖了从早期到近期使用的低电源电压设计方案。

读出存储单元信息的模块称为感测放大器，通常分成不同的部分，第一部分是电流-电压转换器，第二部分是比较器。比较器是一种具有提高换相速度的经典成熟电路，所以本文将重点放在研究转换器上。



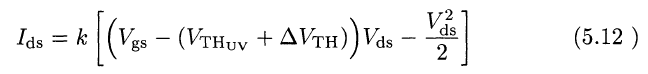
这类转换器的主要特点是换相速度: 范围在10ns到30ns之间，取决于单元下拉电流，它对任何温度变化、电源电压和工艺漂移导致的阈值电压变化都具有很好的鲁棒性。

5.4.1感应技术概述

感应出以电荷形式存储在单元浮栅内的信息，大概是非易失性存储器的最为关键的操作了。在此过程中，存储单元的源端连接到地GND，栅极由行解码器驱动到某个电压，通常为VCC(在升压情况下会更高，请参见5.2.5节)，而漏极通过负载连接到电源电压(见图5.36a);

Cbitline是金属线的寄生电容，在这条线上，同一列上所有单元的漏端都连接在一起。图5.36b中还显示了列解码器晶体管M2和M3，以及级联码配置的晶体管M1;在分析中，通常忽略M2和M3，因为它们的宽长比很大，因此通道上的压降不明显。

在读操作中，单元的漏极电压不能太高，以避免应力现象，但不能很低，要有适当的电流流动，并保证快速读出。由于M1的贡献，该值约为1V，使单元工作在线性区。

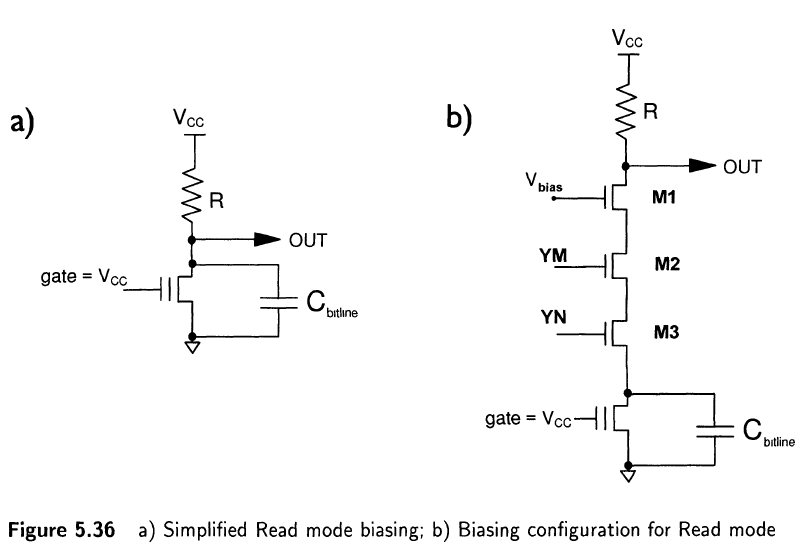


其中VTHuv是UV擦除的非挥发单元的阈值，△VTH是和VTHuv相比的单元阈值偏移。

如果Vds = 1 V 且 Vgs = VCC



由Eq.(5.13)可以清楚地看出，不同单元的特性曲线是平行的，与UV擦除单元的特性之间的差异仅由阈值位移VTH决定。



此时可以分析图5.36b的偏置配置的方法: 设Vbias = 2V，设M2和M3是开的，则存在一个对寄生位线电容Cbitline充电的峰值电流: 一旦达到1V，则M1关闭(为负反馈)。单元的栅极偏置在VCC:如果电池被擦除，例如阈值为2V，那么它开始下拉电流，Cbitline放电和降低节点OUT的电位;如果单元被编程(阈值高于VCC)， Cbitline不放电，节点OUT保持在VCC。

用理想电流源代替电阻R，读操作可以看作是电流源Iref的电流与单元下拉电流之间的比较(图5.37)。

同样，应该考虑以下两个相反条件:

* 单元是初始态的: 如果所示，如果VCC < VCCm，单元不能吸收负载提供的所有电流，节点OUT被拉到VCC: 初始单元被读出为编程态;如果VCC > VCCm，单元电流总是大于负载电流，并且节点OUT被拉到地GND: 初始单元能正确读出。
* 单元是编程态: 如果VCC < VCCM，单元电流低于负载电流，节点OUT保持高(正确读出);如果VCC > VCCM，然后也是一个编程单元下拉一个大于Iref的电流，节点OUT被拉低(错误读出)。

因此，能正确感应单元的VCC范围为VCCm<VCC<VCCM，由于平行特性，等于阈值电压偏移了△VTH。



负载电流的合理选择要满足动态约束; 既要足够大，来给Cbitline 快速充电，但不能过高，以防止初始单元的输出被拉到节点OUT。

Vbias的产生方式对电路的动态行为也有影响: 如果M1的栅端为一个固定值，当M2和M3打开时，M1的Vgs等于Vbias，因为节点A几乎在GND(图5.38a);因此Cbitline通过有限Vgs的M1进行充电。为了解决这个问题，Vbias网络被反向器取代，如图5.38b所示;当节点A’在GND时， Vbias在VCC; 因此，M1的Vgs是可用的最大Vgs, Cbitline的充电速度很快。通过设计反向器nMOS长宽比大于Pmos, （W/L）nMOS>>（W/L）pMOS, 当节点A' 接近VTHn,大约1V时, M1被关闭。 这个方案的主要缺点是功耗: 反馈需要电流才能正常工作。

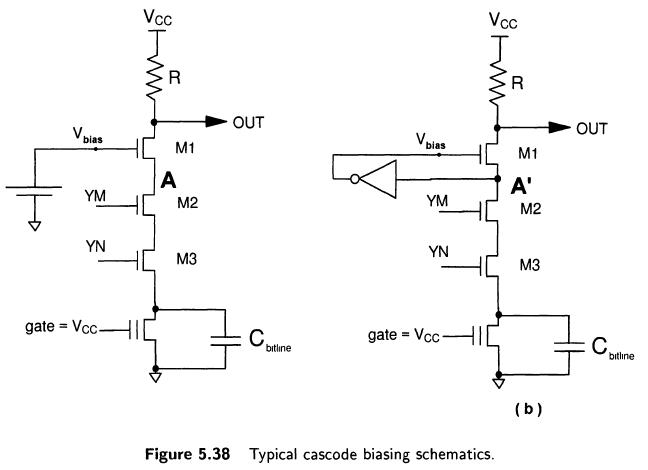
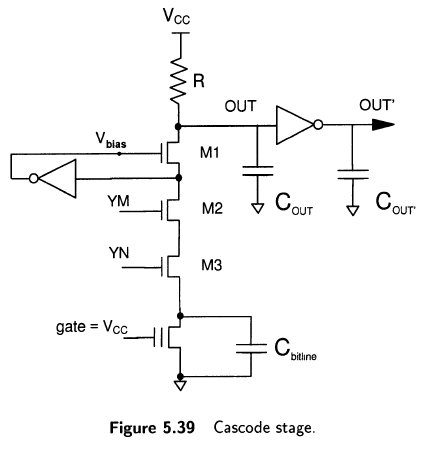
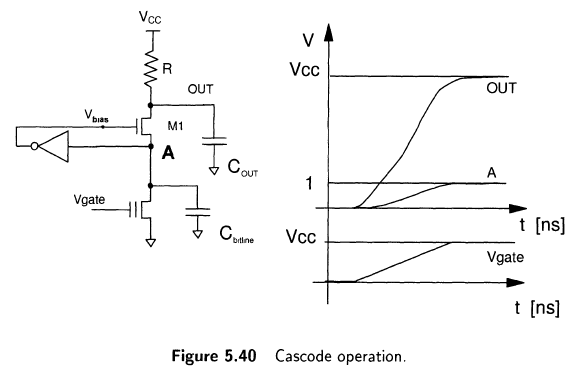


图5.39显示了节点OUT到极简缓冲器(即反向器)的连接，用于在OUT'处产生CMOS电平; 缓冲器用来加强驱动信号，使信号可以长距离传播。在这一点上，值得注意的是Cbitline非常大(几pf)，而COUT要小得多(几十ff); 因此，节点A上电压变化很小，意味着Cbitline内的电荷变化很小，但OUT电压变化很大(如图5.40所示)。

使用“反向器”方法的主要缺点是，编程时单元阈值电压偏移必须很大，而工艺趋势是减少编程时间和电压，以减少对单元氧化层的应力。为了保证可靠的读出操作，需采用差分传感结构。



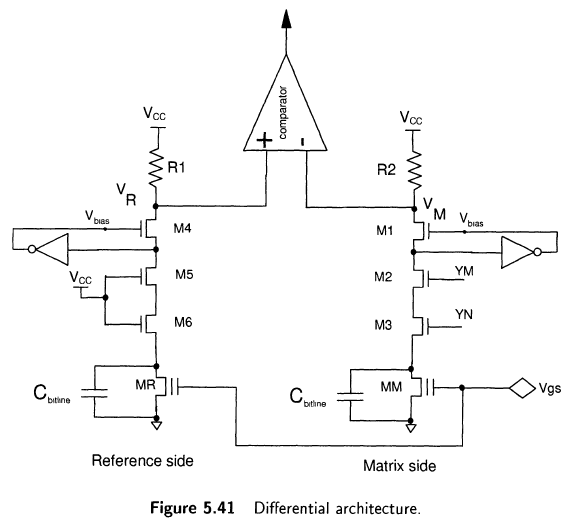


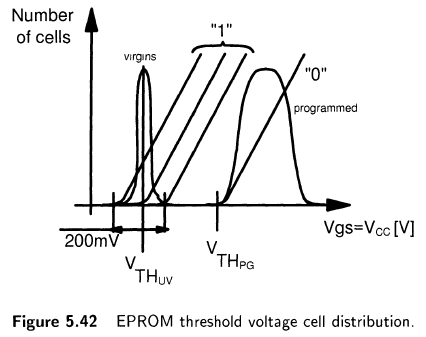
5.4.2差分传感技术

图5.41显示了差分传感架构的基本方案:其中MM是要读取的阵列单元，MR是阈值电压已知的参考单元。

如果是EPROM器件，MR是一个原始单元(即UV擦除态)，其阈值VTHuv等于2V，而MM是一个阵列单元，阈值与浮栅内电荷有关; 这两个单元的信号和寄生元件是相同的。

EPROM器件的典型单元阈值分布如图5.42所示;对于被擦除的单元来说，它非常窄，而对于编程的单元来说，它要大得多;值得注意的是，VTHUV是分布的中心，而VTHPG被定义为最差编程单元的阈值。





典型的VTHPG值为VTHuv + 3V，以保证分布之间可以很好分离，使读操作更可靠，因为存在一个Vgs范围，初始单元会下拉电流，而编程单元不会，图5.42显示了在不存在分布增益的假设下，不同单元的特性。

再次考虑图5.41:电流-电压变换器的两个支路的电气性能和布局配置相同;M5和M6用来平衡列解码器的晶体管M2和M3。R1和R2暂时作为负载，以后将由主动负载代替。

如果MM被编程，阵列支路不会下拉电流，VM在VCC，而MR(原始单元)开启, VR被拉下:因此存在电压差，比较器切换输出。当MM也是初始状态时，图5.41的结构出现了问题: VR和VM相同, 而比较器无法判断出正在读取的是什么类型的信息。为了在每一种情况下获得正确的行为，可以改变负载R1和R2的值; 无论如何选择这些值，VR节点应该总是在VMv(由原始单元产生的电位)和VMp(由编程单元产生的电位)之间，应该总是具有相同的值，根据动态考虑进行选择，独立于阵列单元的支路。因此，关系如下:

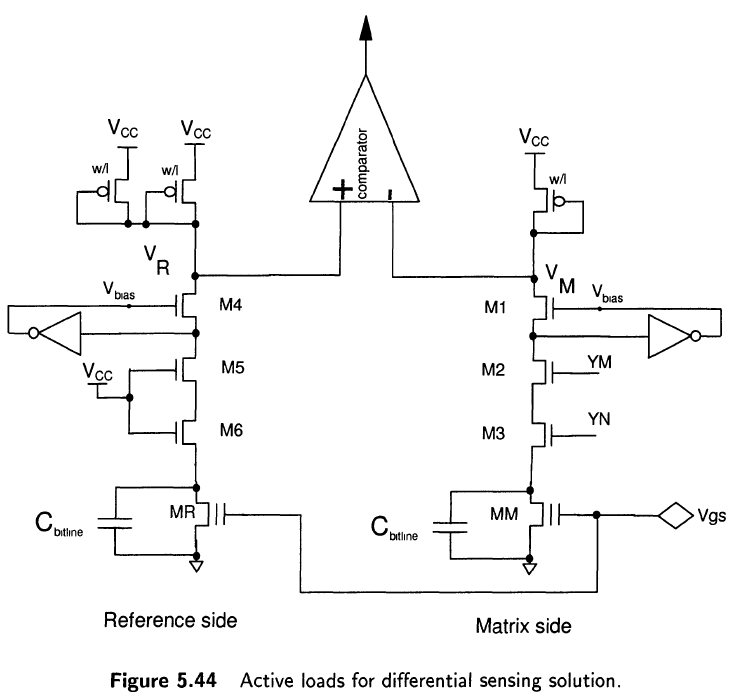


在满足上限值的情况下，有两种方法可以得到下限值(如图5.43所示):(i)减小R1，上拉节点VR; (ii)增大R2，下拉节点VMv。在这两种情况下，如果两个支路汇聚相同的电流，则节点VR高于VMv，因为R1的压降低于R2 (R1<R2)。



最优的解决方案是前者，因为较小的负载值意味着更高的电流，可以快速充电Cbitline。

图5.44所示用二极管接法的PMOS替代负载电阻的第一原理图: 参考支路有两个晶体管，阵列支路有一个晶体管(R2=2R1)。



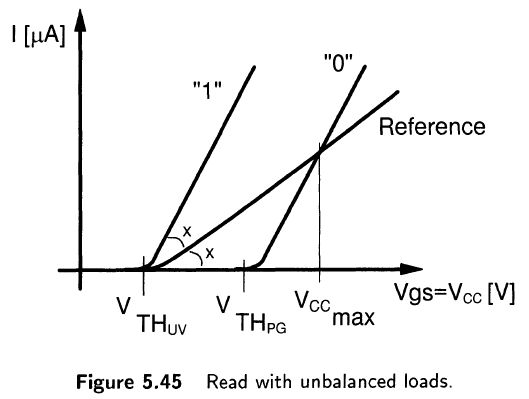
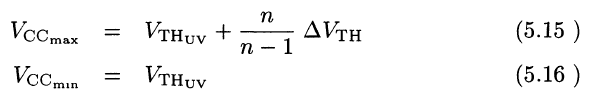


图5.45，在假设MR和MM具有相同尺寸的情况下，绘制参考特征曲线和矩阵特征曲线。参考特征曲线从原始单元的同一点开始（重要的是，这不是真实情况，因为不可能有一半阵列单元的参考单元，但这是获得清晰绘图的一种简单方法）。根据该图，如果一个单元的特征曲线高于一个参考单元曲线，则该单元为原始单元；但是，如果通过简单的检查，该单元低于一个参考单元，则该单元为编程单元，很明显，由于参考和编程单元特性曲线之间的交叉，这种说法并不完全正确：如果对设备电源电压的最大值施加约束，结论仍然有效。假设特征曲线是直线，则以下关系成立



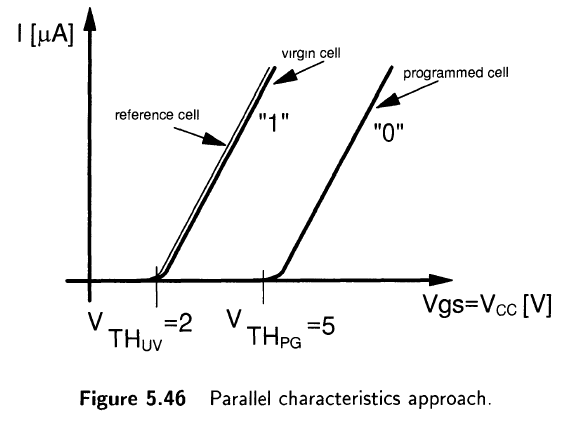
n等于参考支路和阵列支路上负载尺寸之间的比率。

通过选择不同的n值，可以根据需要修改基准侧和阵列侧之间的电流差，从而根据需要为原始单元或编程单元提供更多裕度。

5.4.3偏置电流差动传感技术

具有不平衡负载的电流-电压转换器的主要问题是VCCMAX为器件操作确定了最大电源电压； 事实上，如果VCC变得高于VCCMAX，则读出电路会将已编程的单元误解为原始单元。对于典型设备，VCCMAX等于7V，VCC范围为4.5/5.5V（因此VCCMAX似乎足够高）；然而，必须有一个裕度来满足所有参数变化，如温度、Vcc、频率、工艺、等等，并避免因寿命期间的电荷损失而导致单元工作窗口变小的问题。

避免非易失性存储器中VCCMAX限制的第一个解决方案是引入“平行参考”，即参考单元特性曲线与阵列单元的特性曲线平行。如图5.46所示， 显示编程单元和原始单元的特征曲线，后者与参考单元的特征曲线一致。

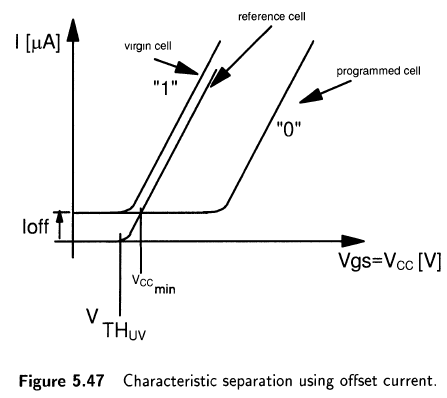


通过严格地向上移动这两个特性，可以得到图5.47的曲线图：原始单元的特性曲线完全位于参考单元的左侧，而编程单元的特性仅在低VCC下共用了一个公共区域，但其余的都在右侧。

因此，对于VCC>VCCMIN，参考特性曲线始终与阵列特性曲线分离；要注意的是，这个方案解决了VCCMAX问题，但会产生VCCMIN。

VCCMIN值（见图5.47）与Ioff有关，Ioff通常为20/30µA；

后一种选择是VCCMIN与参考和原始单元曲线分离之间的良好折衷。图5.47修改为图5.48，因为Ioff电流是由同一VCC供电的电路产生的，因此，如果VCC=0，则不可能实现。



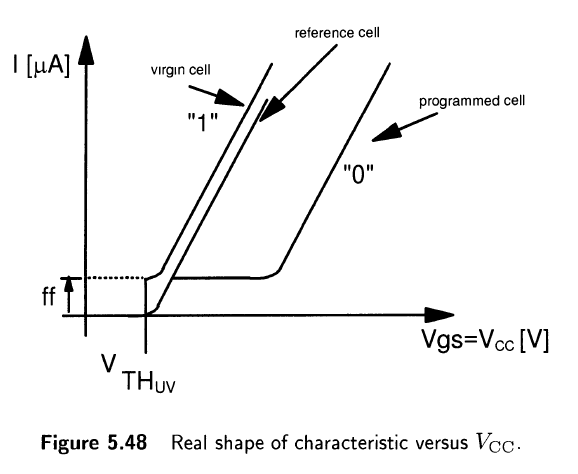
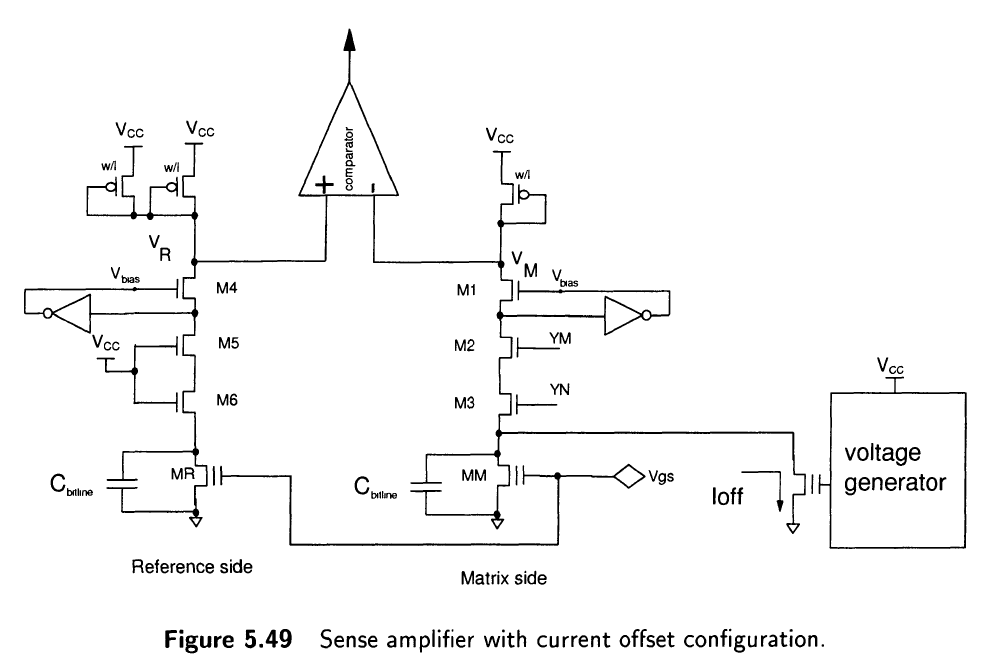
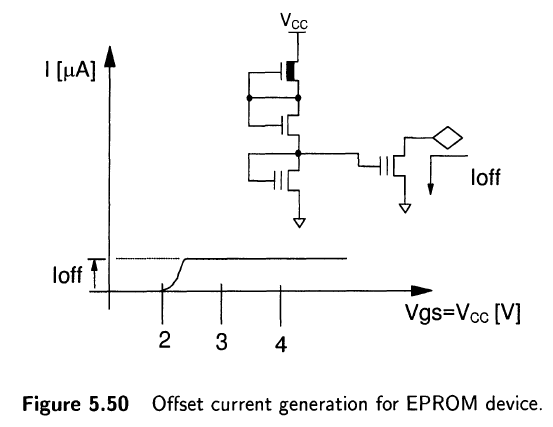
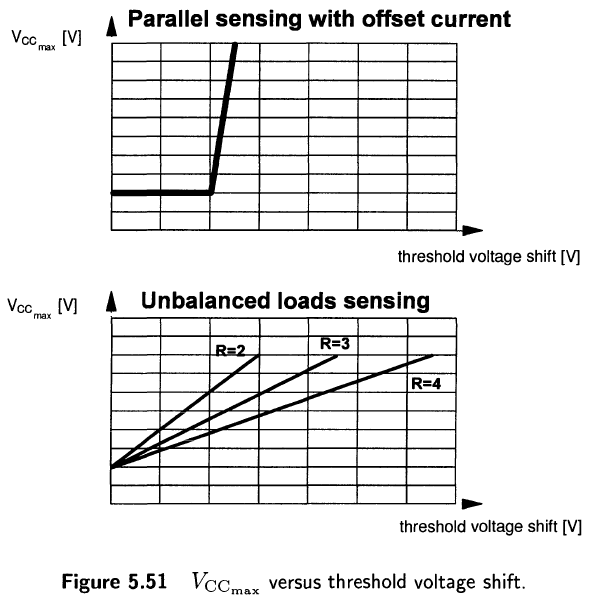


图5.49显示了用于获得图5.48所示特性曲线的电路，图5.50显示了使用nMOS器件和EPROM单元产生偏移电流的电路。



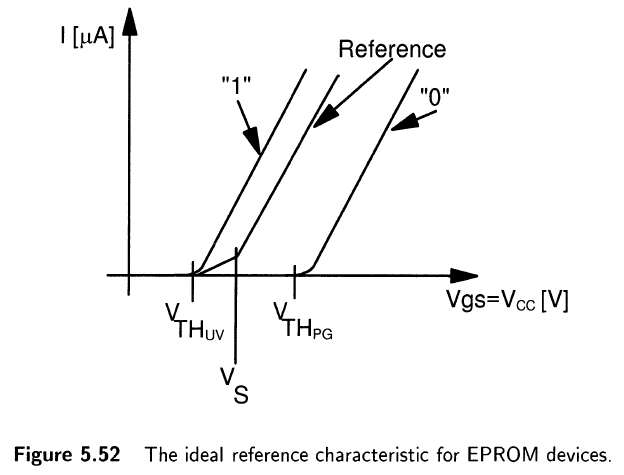


对于这种类型的电流-电压转换器，VCCMAX的理想值为无穷大。对于分析的两种不同类型的转换器，对于VTH的每个正值，VCCMAX与阈值电压偏移的关系如图5.51所示。

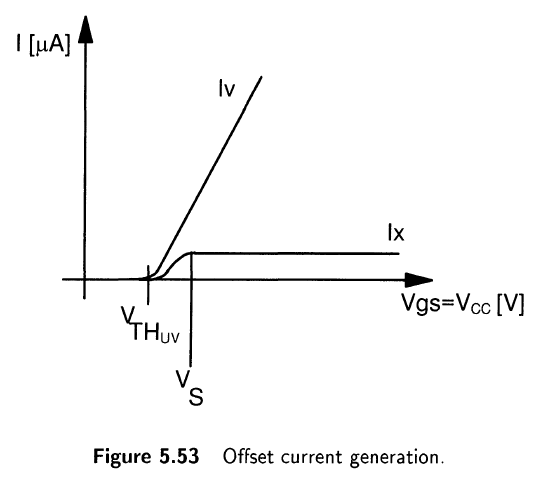


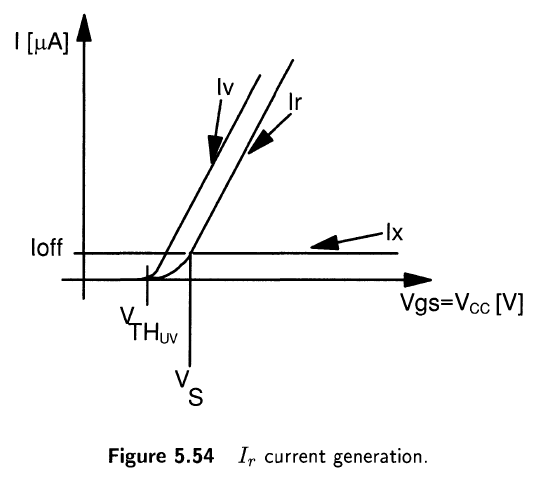
5.4.4差分半平行传感技术

前面描述的转换器被许多nMOS器件采用，并取得了良好的效果，要特别注意收缩对偏移电流值的影响。在CMOS非易失性存储器上设计了一种新型转换器，来产生EPROM器件的“理想参考特性”。目的是将VCCMIN降低到VTHuv；图5.52是设计目标。

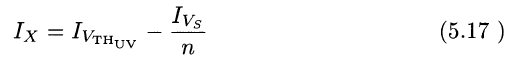


参考特性是之前检查过的特性的一个组成部分：在VTHuv和Vs之间的VCC范围内使用非平衡负载法（这是一个可以确保参考曲线和原始曲线之间的较好分离的“安全”值），在其他VCC范围内使用平行法。





首先，需要获得如图5.53所示的Ix特性；然后，可以通过从Iv曲线中减去Ix曲线来生成新的特征曲线Ir（图5.54）。Ix的获取方式为



其中，系数n必须再现不平衡负载的情况。

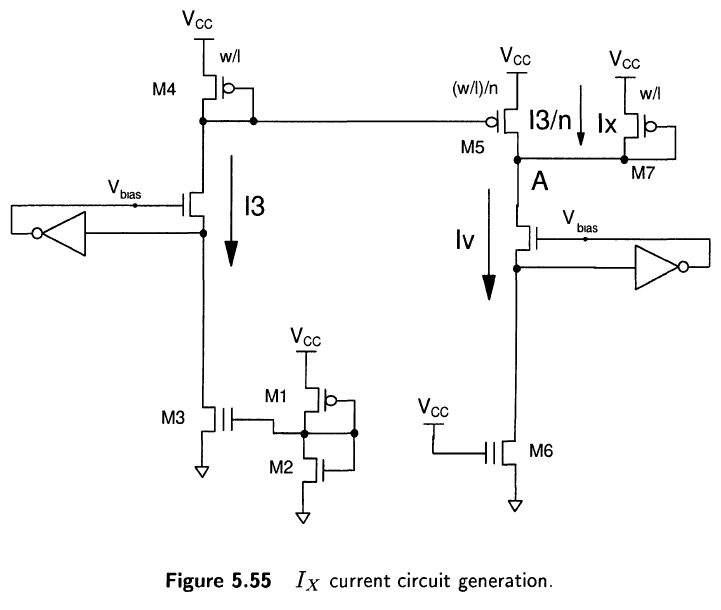
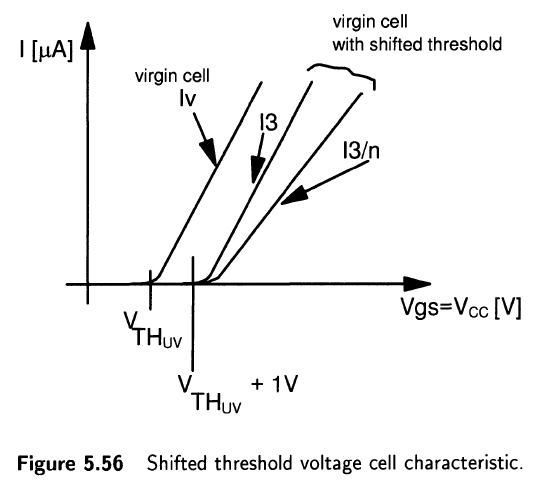
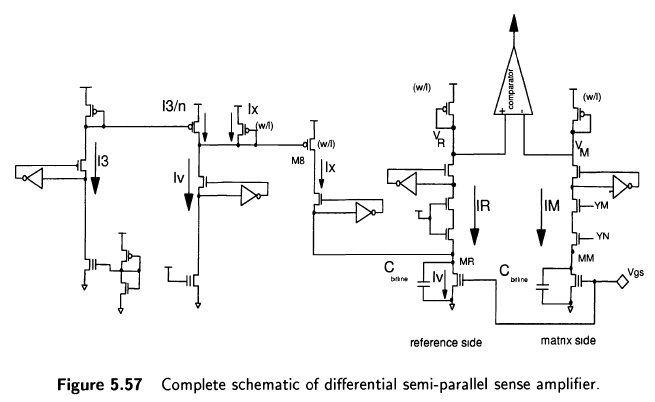


图5.55显示了Ix电流发生器：M6是一个原始单元，而M1和M2（尺寸适当）用≈1V电压驱动M3原始单元，等于VCC-VTHPG：这是一种获得M3“虚拟阈值”的方法，M3的“虚拟阈值”比M6高≈1V。这种技巧对于无法写参考单元的EPROM器件很有用；对于闪存器件，使用可EWS编程的参考单元可以获得相同的特性。晶体管M4和M5形成镜像配置；将M5的宽长比相对于M4的宽长比减小一个因数n，镜像电流也会减小一个因数（见图5.56）。

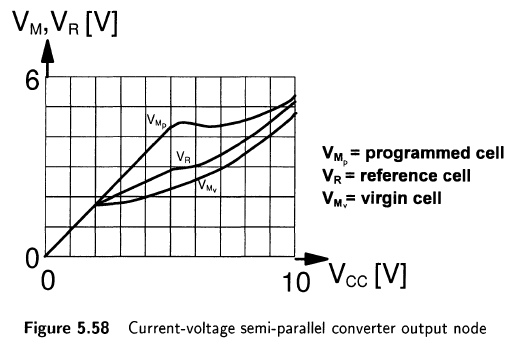


节点A上电流由基尔霍夫定律电流给出：





差分半并行感测放大器的完整方案如图5.57所示；Ir是从UV擦除单元（即MR）电流中减去Ix得到的。VM和VR与VCC的对比如图5.58所示：它们总是分开的且VCC的最小值为VTHuv，不存在VCC最大值。对于EPROM器件来说，这种类型的感测电路是理想的。



5.4.5读出加速技术

在分析读出非易失性单元的电路后，有必要通过检索一些技术来提高读出速度和可靠性。

第一种是“节点均衡”，即减少某些节点的电压变化来加速换相。图5.59显示了具有两个阵列单元的不平衡负载转换器：MM1，由YN1编程和解码，MM2，由YN2编程和解码。

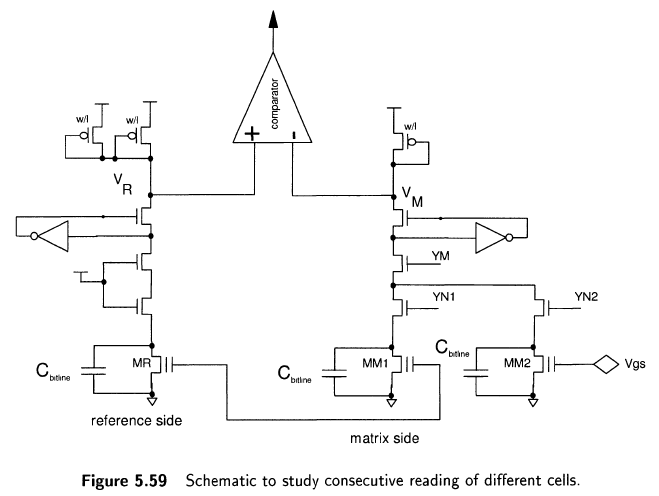
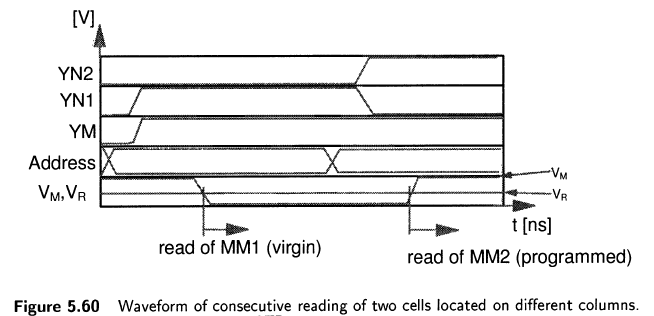
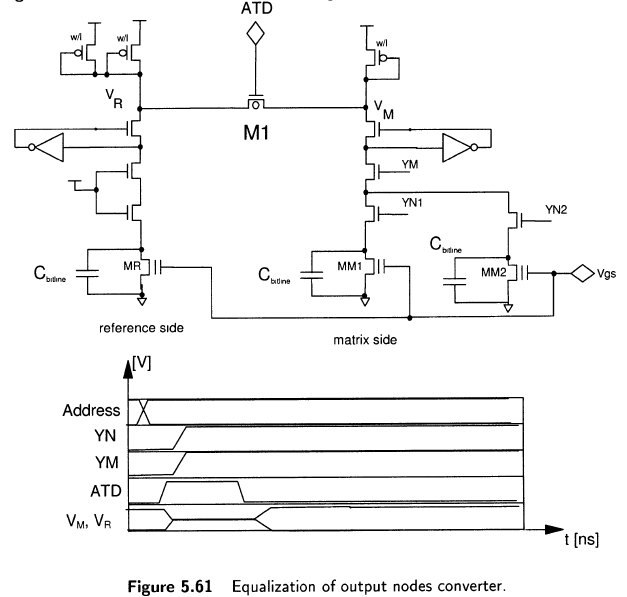
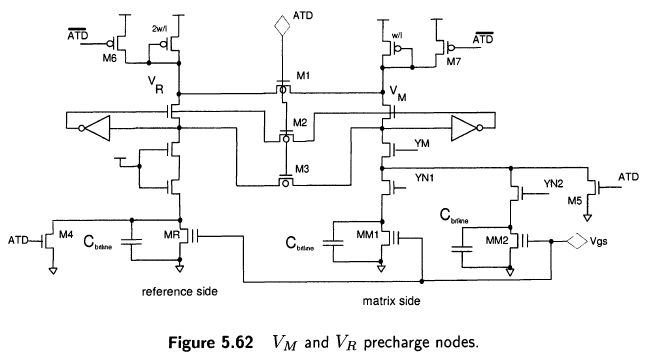


图5.60显示了两个连续读取序列的时间图。假设第一次读取在MM1上，第二次读取在MM2上，并且为了简单起见，VR为一常值，那么很明显在VM上有相当大的电压摆幅，这意味着很长的瞬态时间。相反，为了增强电路的时间响应性，所有节点应围绕其偏置值进行小幅度移动：因此，将图5.59修改为图5.61，其中添加了自然晶体管（即阈值电压0.1V，低于正常的注入增强nMOS晶体管）M1；其任务是在ATD高时使节点VR和VM短路（ATD是“地址转换检测器”的首字母缩写，即当E#或一个或多个地址引脚的状态改变时，能够产生用作时钟的脉冲的电路）。在此阶段，VM和VR均被钳制在适当的值，例如VCC/2：当ATD脉冲停止时，节点VR和VM开始摆动，以根据阵列单元内容达到终值，电压偏移低于之前的方案。均衡技术通常还涉及其他节点，从而提高了电路的速度（不幸的是也增加了复杂性）。

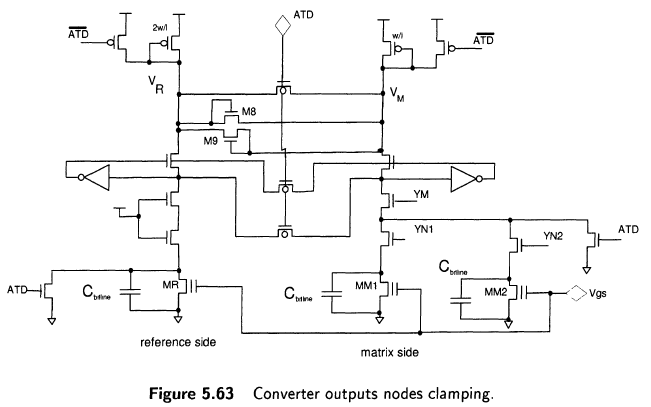




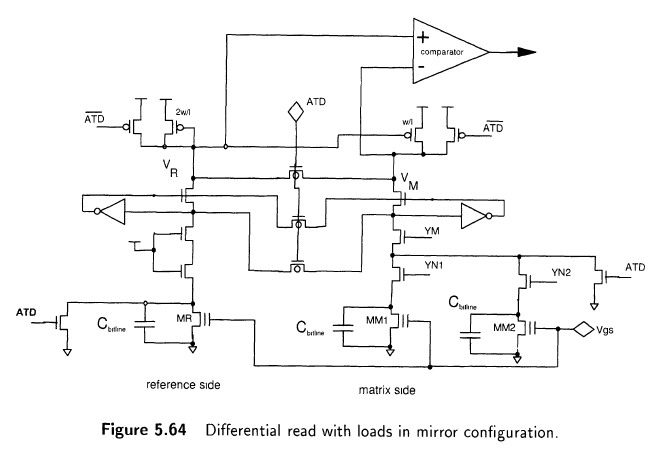
“预充电”技术可以和均衡技术一起用于加速读操作；对节点进行预充电可以在不同相位提前对寄生电容充电：图5.62显示了不平衡负载电流-电压转换器的预充电晶体管：在ATD激活阶段，M6和M7被打开，以共享杂散电容充电：通过M1/M5，均衡与预充电同步，仅通过负载执行读操作。



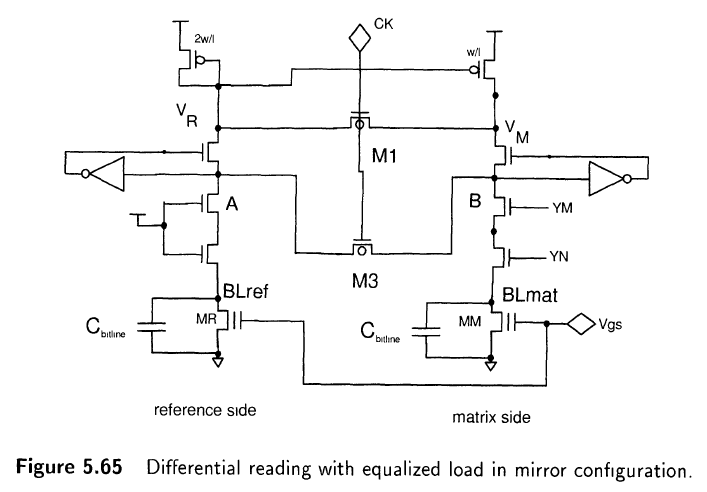
可以设计许多其他“技巧”来加速转换器：如图5.63所示，可以在二极管配置中添加两个nMOS，M8和M9，一个与另一个相反；其效果是节点VM和VR不能被两个以上的VTHn分开，因此减少了电压摆动并节省了时间。这种解决方案的缺点是噪声裕度恶化。



另一种提高读数的方法是考虑pMOS负载晶体管的不同布置。与之前方案中的二极管配置不同，可以实现镜像配置，如图5.64所示。优点是VM的动态特性得到了改善：下限（原始单元的读数）从IV降低到GND，而上限（已编程单元的读数）从VCC-VTHp提高到VCC。



重要的是要正确选择M1和M3（均衡晶体管，如图5.65所示）的尺寸：如果它们太大，则VM和VR及其栅极（由均衡信号驱动）之间的杂散电容变得显著：图5.66显示了仿真结果，其中M1和M3的源极栅极和漏极栅极之间有明显的耦合。



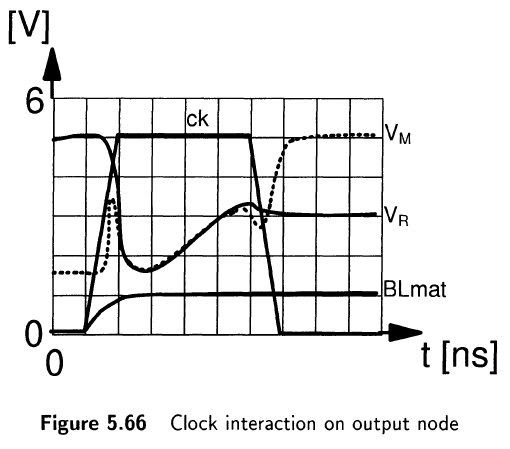
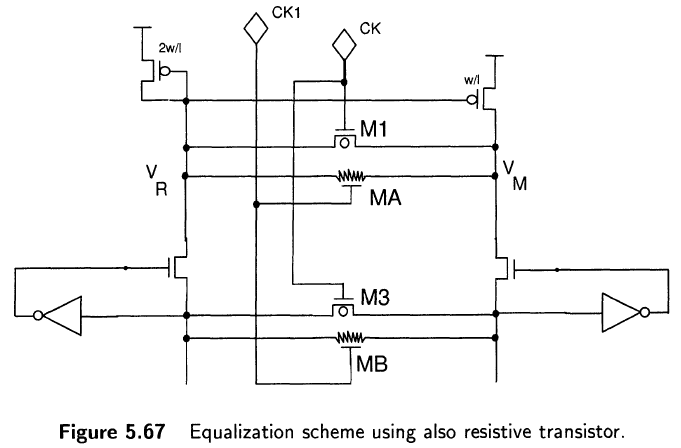
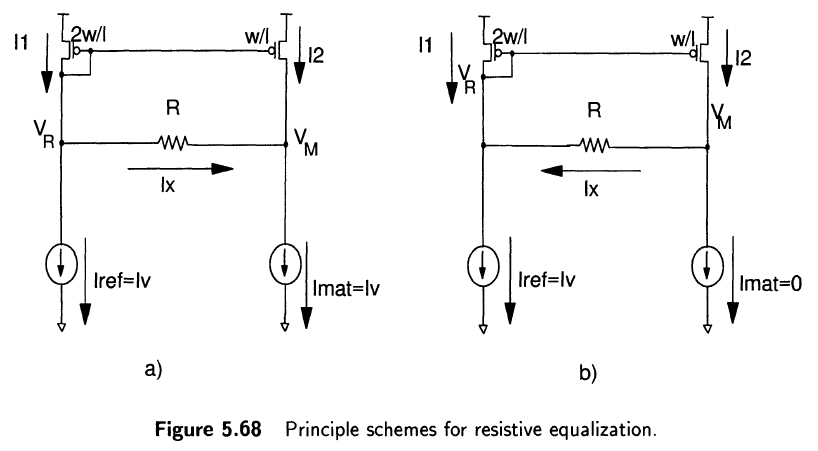


图5.67显示了一种不同的解决方案，该解决方案使用了两个额外的晶体管MA和MB，其导通电阻约为2kΩ；CK1的上升沿与CK信号一致，但在较长时间内保持较高；当MA和MB仍然打开，而M1和M3已经关闭时，通过MA和MB的电流在VM和VR之间产生电位降。图5.68a和图5.68b分别显示了原始单元和编程单元情况下的电路行为。





如果电池为原始电池（图5.68a），则



然后：



使用不平衡负载的感应放大器：



因此：

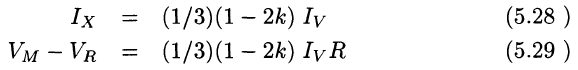




如果单元是编程态，则：



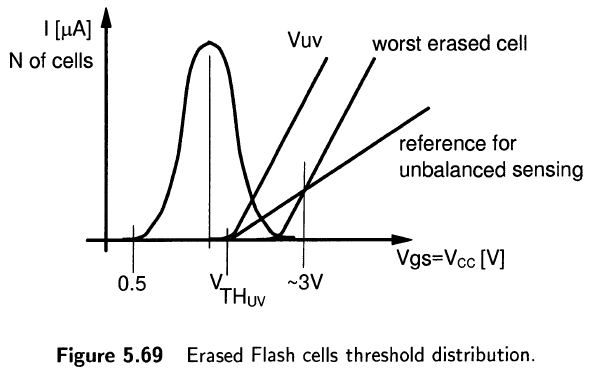
定义k=Is/Iv，即原始单元和编程单元电流之间的比率，并假设0<k<1/2，则



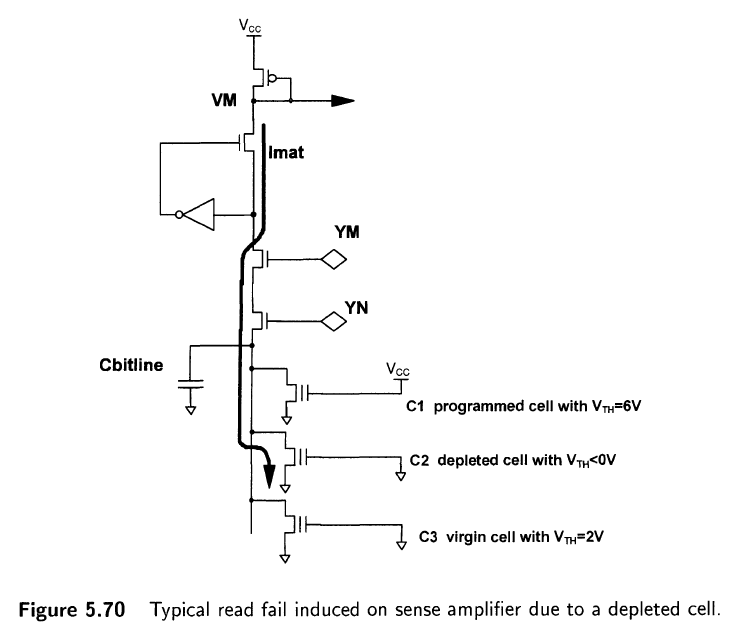
根据单元状态，电流Ix以相反的方向流动；如果单元为原始单元，参考支路“帮助”阵列支路对电容充电，而如果单元编程，则会出现相反的情况。从等式（5.29）可以清楚地看出，即使在ATD阶段，只要已经译码选中单元，也可以从已知数量分开VM和VR节点。

5.4.6从EPROM到闪存

前面段落中介绍的所有概念也适用于为闪存设计感测放大器，可以使用不同的方法（不平衡、并联或半并联负载）。但是，也要考虑与电擦除相关的新问题。图5.69显示了擦除单元阈值和UV擦除单元阈值的典型分布（VTHuv==2V）。前者不应超过0.5V的下限，因此应置于0.5V和2.5V之间。在这些条件下，很明显，如果以VCC==2.7V（典型低压设备的最小值）读取最差擦除单元（VTH==2.5V），则其过驱动仅为200mV，即仅为1/5µA电流：这意味着感应放大器的分辨率必须非常好。

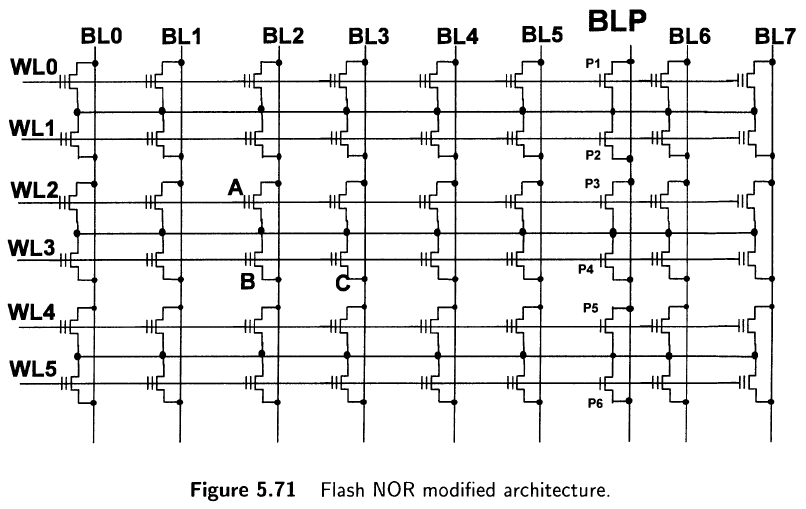


另一方面，当存在VTH<0的已擦除单元（即“耗尽”单元）时，会出现另一个问题：图5.70中描述了这种不希望出现的情况。假设C1（已编程，VTH==6V）是寻址单元；其他单元共享相同的位线：C3（原始，VTH==2V）和C2（耗尽，VTH<O）；显然，“传统”的感应放大器无法理解电流贡献是来自C1还是C2。通常，耗尽位的问题是通过所谓的“软编程”来解决的：在擦除算法结束时，检测到耗尽的单元，并稍微重新编程，直到其阈值超过零。下一段将介绍一种不同的方法。



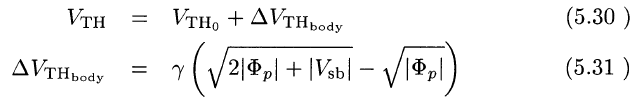
5.4.7 7读取耗尽位的闪存

如果矩阵结构如图5.71所示发生变化，则无需恢复耗尽位，即使它们存在于选定的位线上，也可以正确读取：另一条“位线”（BLP，由nMOS 通过P1…Pn组成）放置在扇区内：这些晶体管以与闪存单元相同的方式连接到位线、字线和源。如果A是寻址单元，则WL2为高电平，P3为on，从而将BLP线与字线为WL2或WL3的所有单元的源连接。



如果BLP偏置于GND，则上述源也位于GND，而其他单元的源是浮动的，因为它们的Pi晶体管关闭。不巧的是，如果选择了NOR阵列体系结构（即所有单元共享同一源），耗尽单元（例如单元B或C）即使其pass（本例中为P4）关闭，也会产生不需要的电流贡献，因为源节点（位于GND）对于所有由WL2或WL3驱动栅极的单元来说是公用的。

为了克服这个问题（即关闭非寻址单元，无论是否耗尽），可以利用体效应：阈值的值修改如下：



式中，Φp是费米势，Vsb是源体电压，VTHo是Vsb=0时的阈值电压，γ是体效应系数。

偏差如下：

•BLP在VCC而非GND下驱动；

•没选中的位线连接在VCC；

•选中的位线连接在VCC-1.5V；

•没选中的字线连接在GND上；

•选中的字线被加强驱动。（见第5.2.5节）。

因此，对于选定位线上的所有没选中的单元（即Vg=0V）（即Vs=VCC-1.5V）



如果其中一个单元是耗尽态，该方程也适用：由于特定的偏压方案，其Vsb高于零，因此其阈值（不理想的负值）可能会升高，从而使其更难接通。因此，要打开阈值为VT0的耗尽单元，其VGS必须为:

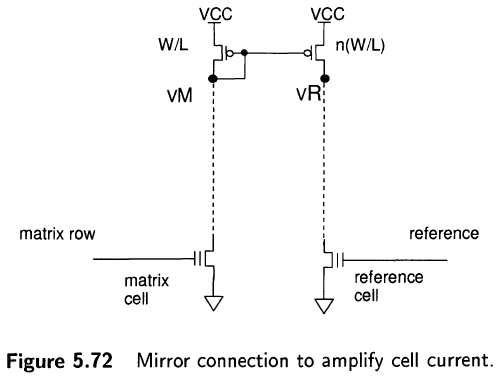


上述解决方案的另一个优点是，可以反转单元源极和漏极，从而设计不同的方法来执行编程和擦除。

5.4.8低压FLASH读出

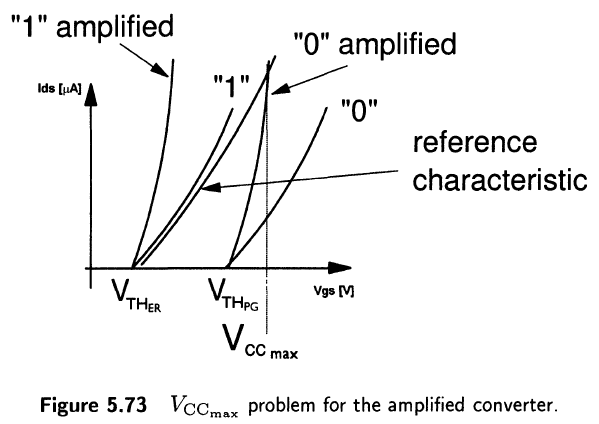
在低电压下读取时的主要问题是单元的有效电流量。假设最差擦除单元的阈值为2.5V，则该单元吸收的电流在2.7V时的范围为1µa至5µa，这是低压设备的最小VCC值。另一方面，快速读出要求参考特性与阵列单元特性分离：然而，在这种情况下，参考电流和矩阵电流之间的差异可能很小，意味着传感时的开关时间很长。

为了在所有节点上快速工作，必须使已擦除单元和参考单元都拉大电流，以便分别对位线电容放电和读取已编程单元。

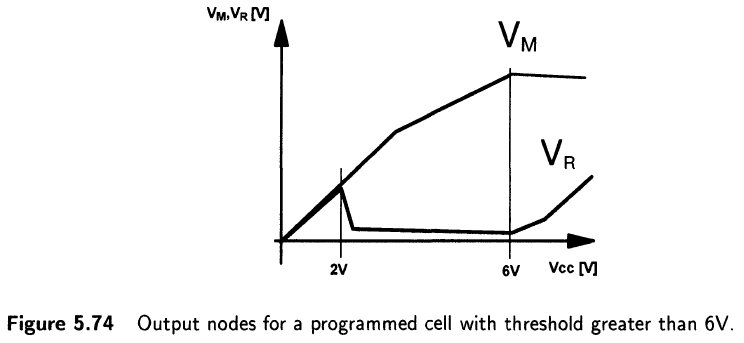


有必要设计一种新型转换器，能够放大电池电流；这种电路如图5.72所示：矩阵单元电流被镜像到参考支路，并通过镜像因子n进行放大，从而克服了在执行低压操作时弱电流的问题。

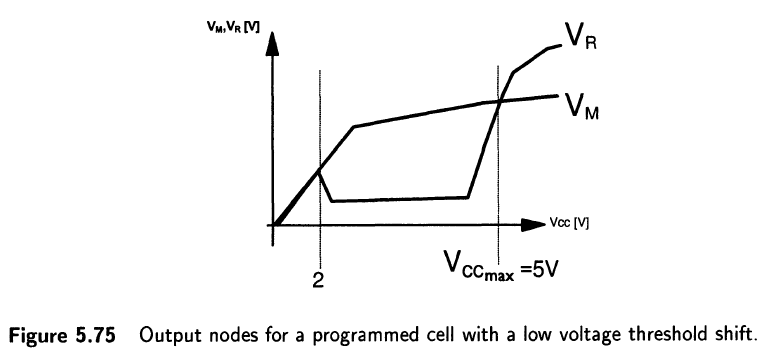
图5.73显示了编程阵列单元和参考单元的特性，而图5.74显示了VTHPG>6V时的节点VM和VR。



在图5.74中，应研究两种不同的条件：VCC<VTHPG和VCC>VTHPG；在第一种情况下，节点VM和VR在VCC-VTHPG处连接在一起，直到VCC足够高，可以打开参考单元。随后VR被参考电压强制为0V，而VM电压持续上升，因为阵列单元不产生任何电流。当VCC>VTHpg时，阵列单元开始传导，因此节点VM被迫向下，VR由于镜像放大而向上驱动。

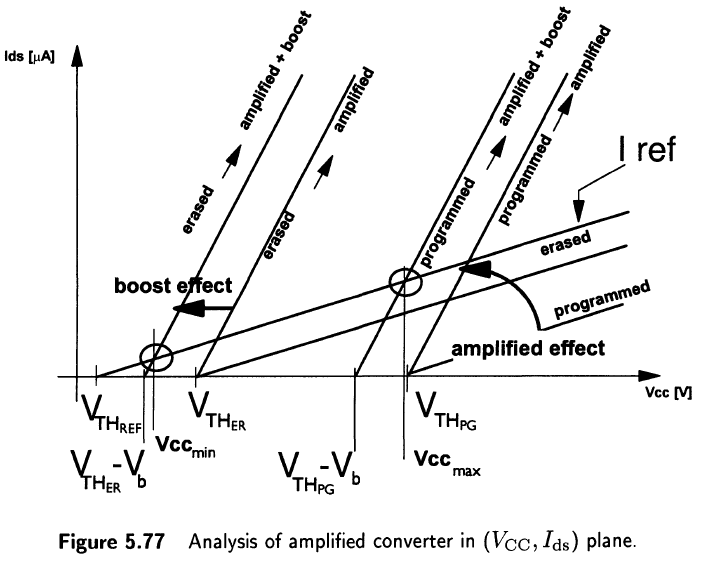


然而，如果编程单元的阈值偏移很小，则可能存在一个电源电压值，其中两条曲线VR和VM具有一个截面，从而确定了VCC最大值（图5.75）。另一方面，对于已擦除的单元，两个节点总是分开，如图5.76所示。

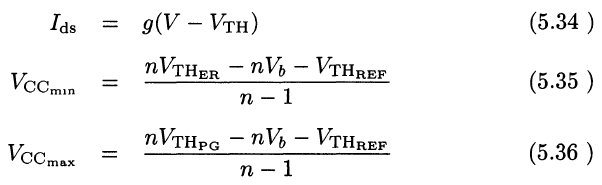




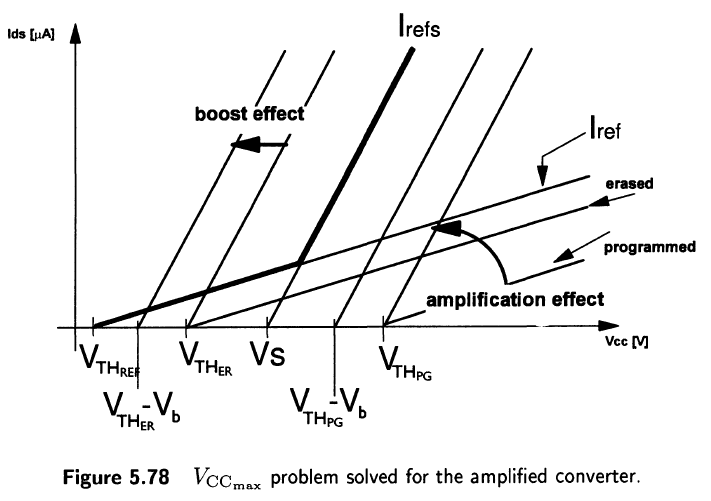
重要的是要设计一个参考特性，确保擦除和编程特性曲线在整个电源电压范围内不能交叉；图5.77显示了在漏极电压等于1V的假设下线性化的单元和参考特性曲线：VTHREF、VTHER和VTHPG分别是参考、最差擦除单元和最差编程单元的阈值。

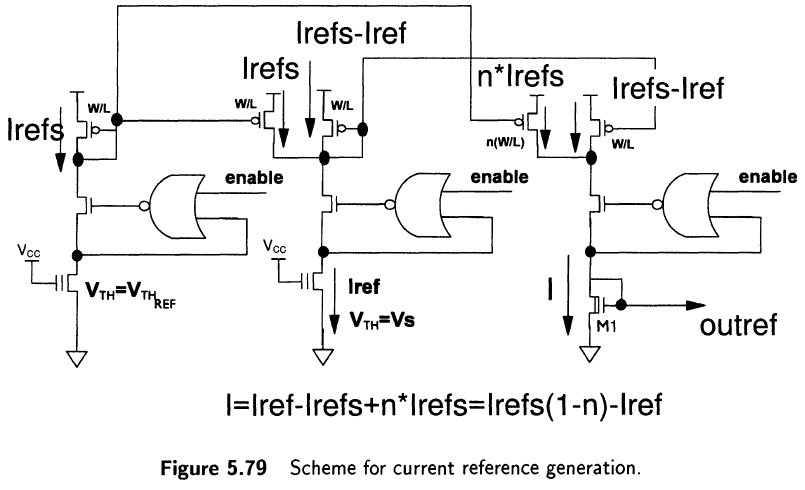


假设对Vb施加升压驱动，特性向左移动；如果g是单元的跨导，n是阵列电流的放大因子，那么：



如图5.78所示，如果参考单元的斜率在适当的值Vs处发生变化，则不再发生VCCMAX问题；图5.79显示了通过适当的镜像方案产生参考电流的可能方案。设计这种转换器时必须小心，因为位线上的漏电可能会导致读出失败，而位线漏电流可能会被镜像电路放大：编程的单元可能会误读为擦除态。





5.4.9参考问题

如图5.79所示，参考电流由参考单元产生，参考单元通常位于数据阵列外的小阵列中；该电流通过二极管配置的nMOS晶体管在芯片周围传输。

为了理解并更好地理解此解决方案，应进一步研究参考电路的全局结构。

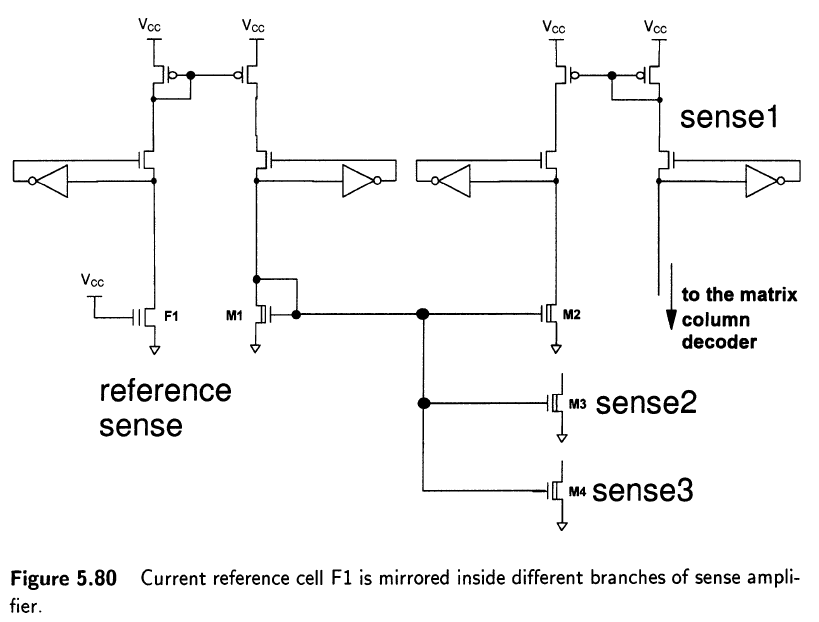
在EPROM设备中，每个输出都有一个专用列，其中所有单元都是参考单元；选中的一行会自动打开输出相关的参考单元。该解决方案保证了阵列和参考单元之间良好的时序相关性，因为行是相同的，并且由于架构选择，两个转换器分支上的负载也是相同的。使用这种方法的缺点是参考单元的数量太大：例如，对于具有8个输出的1兆位闪存，按1024行乘以1024列进行组织，参考单元的数量为8192，每个输出1列，每行1个单元。

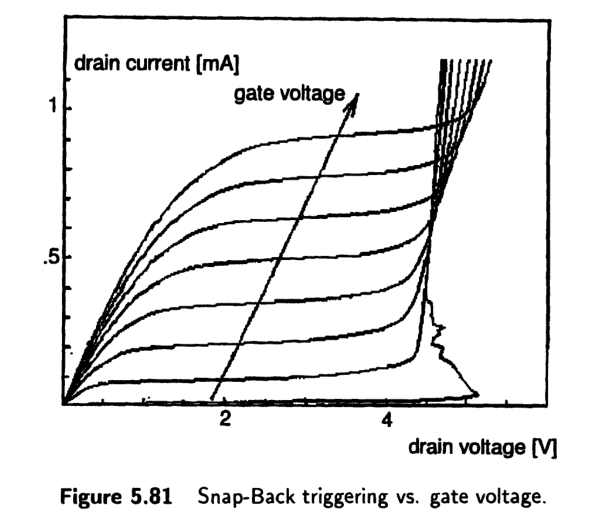
在第一代闪存中，当尚未引入选择参考单元阈值的可能性时，使用了UV擦除单元，其架构与EPROM中的一样，每个输出都有一个参考列。这些FLASH的特点是源擦除，即栅极为零伏，源极为12V，参考单元的源与阵列的源是分离的。

引入具有负电压擦除功能的新一代FLASH，从阵列中移除了参考单元，以避免应力修改参考单元自身阈值。

在这种情况下，会出现在芯片周围传送参考电路电流的问题，并且很容易理解（见图5.79）使用自然管nMOS M1更容易镜像其他所有参考支路参考电流（图5.80）（实际上，图5.80所示的电路更为复杂，因为只有当两个下拉晶体管F1和M1具有相同的电气特性时，用于镜像M1中F1电流的配置才能像完美的镜像一样工作：这是不对的，电路在VCC≈4V时停止正常工作；使用具有电流补偿功能的电路可以消除此限制）。

讨论了一种通过主电流-电压转换器读取FLASH非易失性存储器电流单元的方法。省略了许多细节，只是为了让读者了解此类设计中涉及的基本问题。





5.5编程操作电路

在5.1章节，指出修改闪存单元的状态，需要一些高于VCC的电压。更具体地说，是使用通道热电子效应对闪存单元进行编程，必须将两种不同的正电压施加到栅极和漏极，而源极则连接到GND。此外，这些电压大于传统的电源电平（5或3V），并且还需要调节，以避免在执行多个编程/擦除周期后可能损坏存储单元或降低其可靠性的现象。此外，执行编程验证操作还需要其他电压电平，即检查单元状态，以确保其阈值在编程脉冲后已正确移动。 本节分析专用于产生和调节这些编程电压的电路。

5.5.1单元编程电压：最佳选择

选择最佳编程电压通常是一项艰巨的挑战，并且有几个考虑因素决定了电平和时序的定义。首先，在编程阶段，存储单元的工作点必须保持在其“工作窗口”，定义了漏极电压的允许范围与有效单元长度的关系，需考虑到编程效率、回弹和漏极应力效应。栅极电压还会影响一些重要的参数，如编程效率、漏极电流和回弹触发点。回弹触发和栅极电压的相关性如图5.81所示。从该图中很容易看出存在一个越小的栅极电压，其回弹触发发生在越低的漏极电压值。

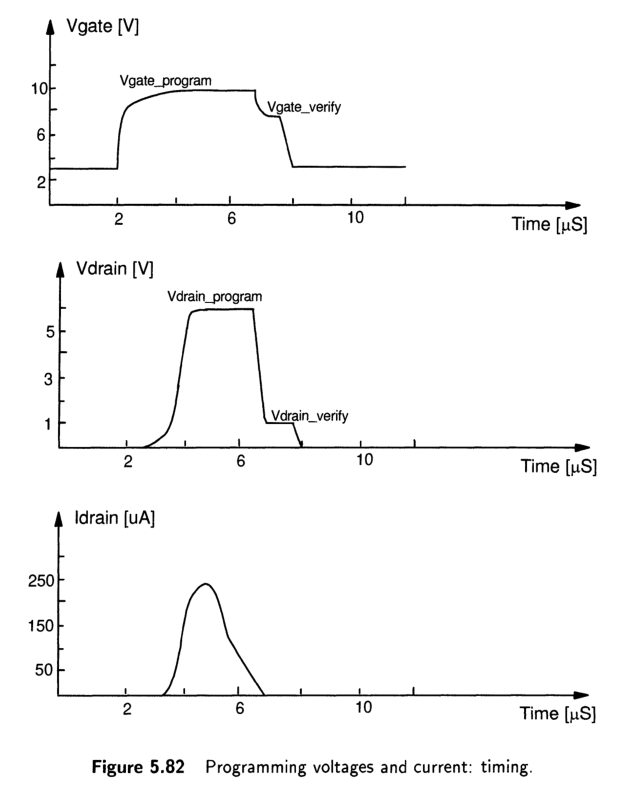
因此，在编程负载激活之前（提供漏极电压），必须提高存储单元的栅极。这意味着，在栅极电压斜坡上升期间，漏极编程电压不会施加到单元上，从而避免了任何回弹的风险。

图5.82显示了编程期间栅极/漏极电压和漏极电流的典型时序图。编程验证阶段也在此图中显示。

就漏极应力而言，重要的是要考虑施加到漏极的电压值及其施加的时间。由于前面描述的寄生效应（回弹和漏极应力）很大程度上取决于编程电压，因此栅极和漏极电压通常由专用电路调节。总之，驱动最佳编程电压选择的主要限制因素如下：

漏极电压：

• 编程效率（即编程时间）;

•对同一BL线的其他单元的漏极应力效应;

•回弹效应;

•漏极电流;

栅极电压：

•编程效率;

•回弹效应;

•漏极电流

5.5.2典型编程路径

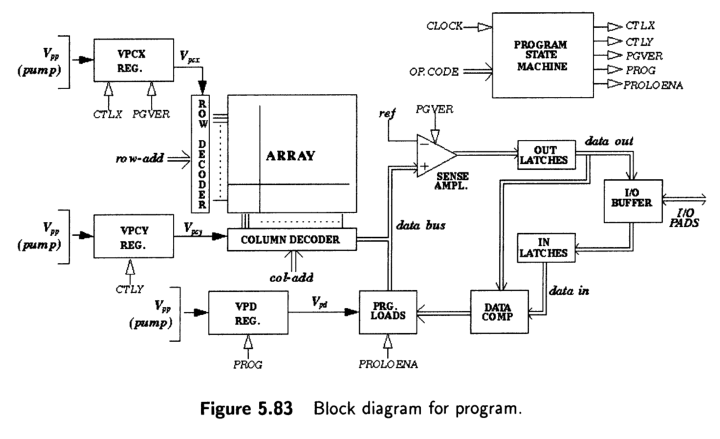


图5.83显示了闪存设备中模拟编程信号的框图。

假设适当的指令已应用于设备，以激活编程/擦除控制器，因此而进入其算法；此外，假设单元地址和要编程的数据已存储在输入锁存器中。编程算法从检查（编程验证）实际单元内容开始，如果后者不等于输入锁存器的内容，高压信号在给定的时间段内被强制到所选存储器元件的栅极和漏极，然后形成另一个编程验证以检查结果。同样，如果锁存数据和内存数据之间的比较成功，算法将终止;否则，上述过程将重复，直到编程尝试次数超过允许的最大限制，在这种情况下。算法以失败终止。

在本段中，分析了为闪存单元提供稳压高压信号的电路;在下文中，VPCX是通过行解码器施加到单元栅极的的电压，VPCY是用于列解码器的电压，VPD是cell通过编程加载阶段加在漏极的电压。

5.5.3漏极电压调节: 原理和基本电路

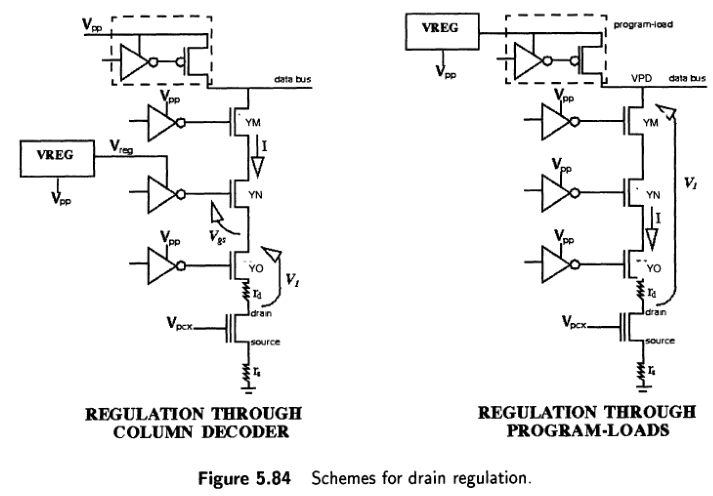
值得记住的是，上述电压来自“双电源”器件中名为VPP的外部引脚，而在“单电源”器件中，它们通过电荷泵电路从电源 Vee 获得（在第 5.6.2 节中进行了说明）。目前，考虑前一种类型的设备。从图5.84中可以清楚地看出，有两种方法可以将合适的电压强制到单元的漏极：

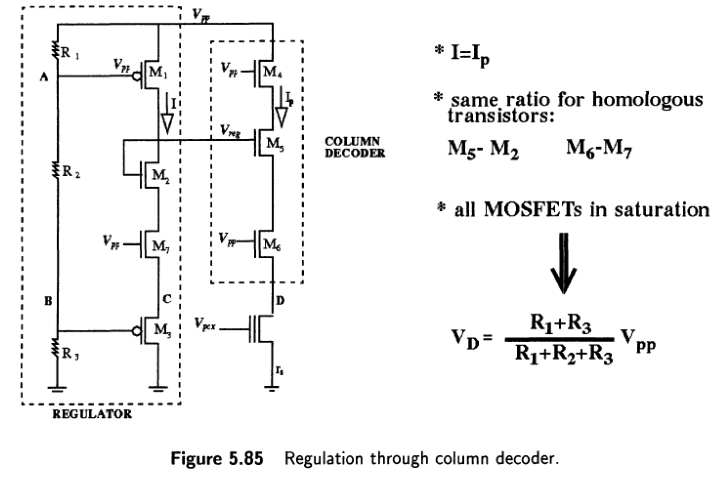
1.“通过列解码器调节”（图中的左侧电路。 5.84),

即严格控制其中一列解码器通过传输晶体管的栅极。在这种情况下，全VPP电压在没有控制的情况下施加到编程负载，而YN列解码器传递晶体管的调节电路将单元漏极上的电压限制为以下值：



其中 Vi 是解码器的最终 YO 通过和本地BL线的压降。出于实际原因，调节设置在YN上而不是YO传输晶体管上：实际上，YO通过栅极物理分布在整个矩阵中，而YN和YM栅极位于底端。该解决方案的成就是稳压器不提供电流，电压降V1仅由于局部BL线压降，而其缺点是等式（5.37）中的Vgs（非线性）依赖于电池的编程电流，并且要增强列解码器宽度以减少Vgs和Vi的压降。图5.85给出了这种调节的一个例子。电路侧的电流相等，所有晶体管在饱和区域工作，如果同源晶体管（M5和M2，M6和M7）工作给定相同的宽长比，可以证明单元的漏极电压仅仅是Vpp和电阻的函数（计算见图5.85）。 值得注意的是，图5.85的解决方案没有利用反馈，因此不能对温度、单元老化或工艺离散进行控制。





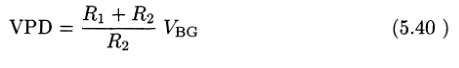
2“通过编程负载进行调节”（图5.84中的右侧电路）：在这种情况下，通过编程负载将稳压VPD电压施加到数据总线，而强制到列解码器传递晶体管栅极的电压足够高，以使它们保持在三极管区域，从而限制了它们之间的压降。然后，漏极电压等于：



该解决方案的显着优点是可以采用更紧凑的列解码器，并且将证明可以实现对Vi实际值的主动计算，而其缺点是稳压器本身必须为单元提供编程电流，并且稳压器电路比上述方案更复杂。图5.86显示了这种类型的一个例子，它利用了运算放大器，dummy解码器（即没有连接单元的列解码器）和将编程电流lc镜像到虚拟解码器中的电路。在图5.86中，没有指出编程负载级，因为它们两端的压降可以忽略不计。同样，通过适当选择镜像因子k，电压在传输管的压降和dummy解码器△Vd和△Vc可以相互补偿，因此漏极电压与温度、电流和老化无关：



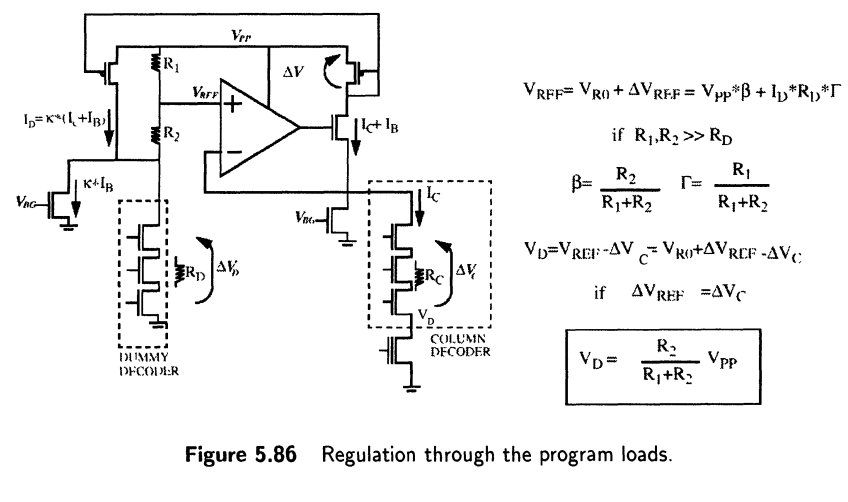
5.39清晰地表明Vd只和稳压器电阻相关。如果考虑采用单电源器件，则任何高电压都必须来自电荷泵电路。因此，很明显，虽然图5.85的调节方案可以很容易地适应这种情况，但图5.86的调节方案不能，因为电流镜两端的压降使解码器的可用VPD至少降低了1伏，因为电流镜两端的压降为△V。因此，通过编程负载进行漏极调节的典型配置是图5.87，它使用带有分压器的运算放大器，带隙参考电压VBG关系式为：



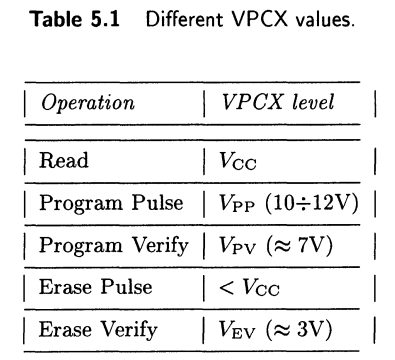
因此，VPD电压在工作阶段保持稳定，上面那个电路则不可以。

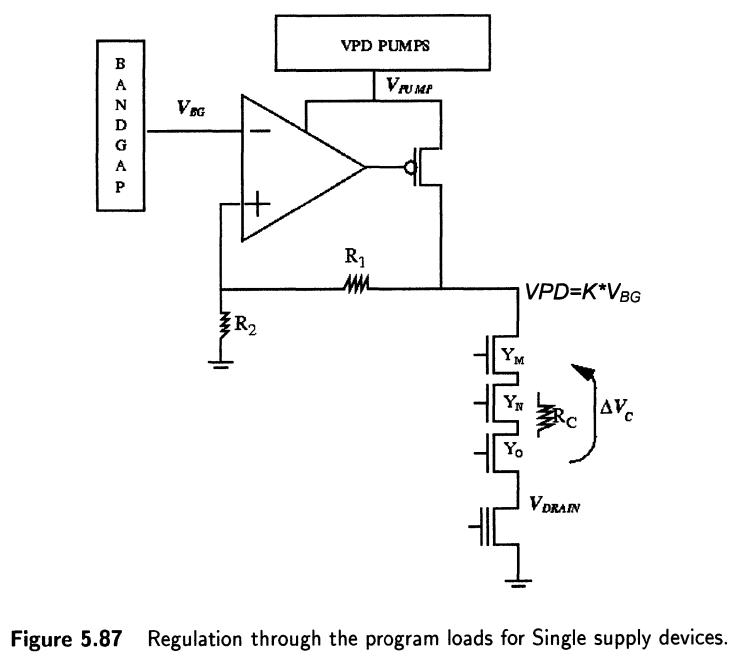
5.5.4栅极电压调节基本原理

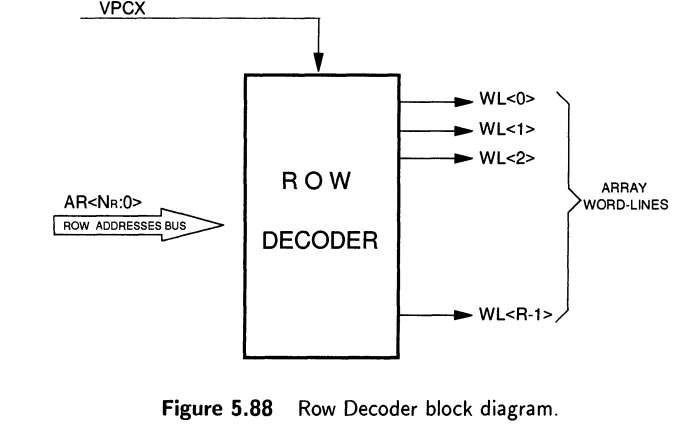
存储器阵列内的每个闪存单元通过字线从行解码器接收栅电压;在图 5.88 中，显示了行解码器与行地址总线、阵列字线和专用电源 VPCX 的连接。该电路允许将VPCX电源传输到由行地址选择的存储单元。

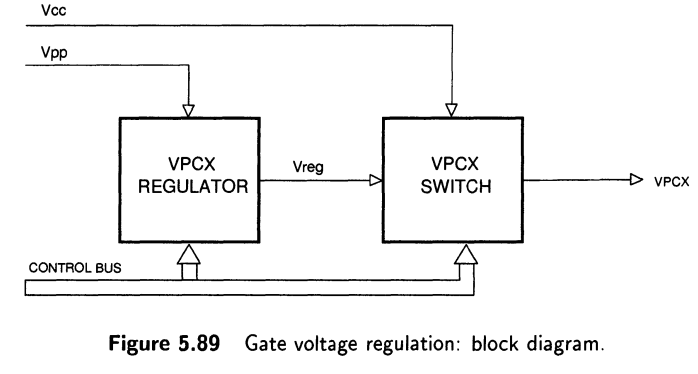


每个内存操作（读取、编程、验证等）都需要不同的 VPCX 电压，如 Tab. 5.1 所示。



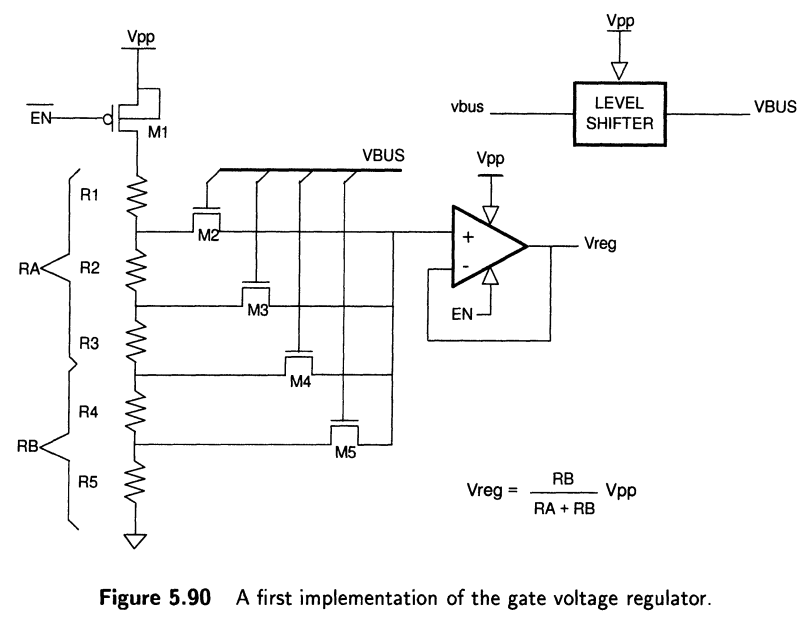




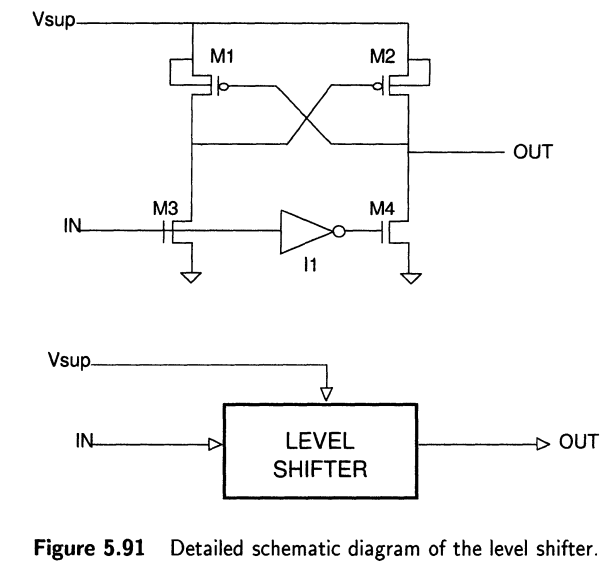


为了产生所有这些电压电平，使用根据图5.89的框图的电路，该电路由稳压器和高压（HV）开关组成。稳压器提供与 VCC 不同的所有 VPCX 电平，而高压开关将行解码器电源连接到 VCC 或 vreg（栅极稳压器的输出）。 两个电路的工作模式由控制总线定义，由编程/擦除控制器驱动。

一款简单的稳压器架构，主要用于双电源闪存，如图5.90所示。在这种情况下，明确定义的电平电压是用户在VPP引脚上提供的编程电压VPP。 器件规格通常要求Vpp电平为12 V，容差仅为5%，因此该电源可用于通过简单的分压器获得基准电压。



然后通过一组抽头（图5.90中的M2-M5）获得各种基准电压，这些抽头根据控制总线的某些位使用模拟开关进行选择。 控制总线的数字信号通过一些电平转换器进行转换，以允许模拟开关完全导通。这些电平转换器的典型实现如图5.91所示。



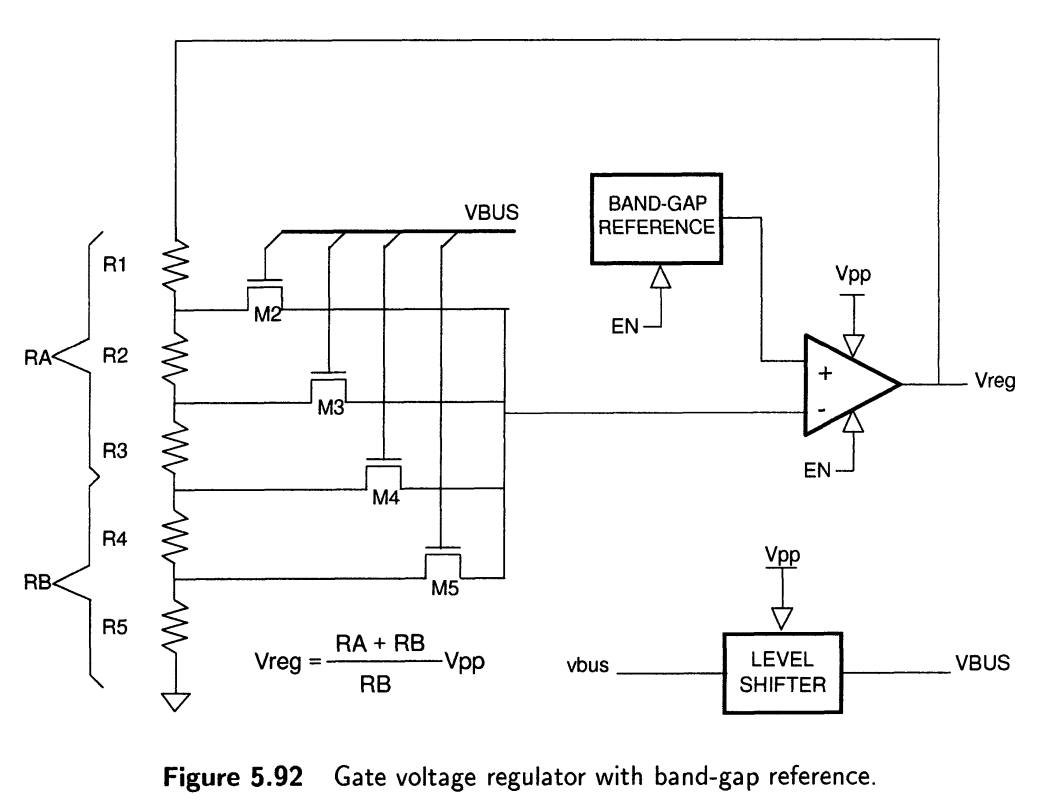
在该原理图中，Ml，M2，M3和M4形成一个双稳态电路，该电路由输入数字信号IN驱动，并由反相器I1产生的反向信号驱动，如果IN处于逻辑电平高电平，M3和M2导通，输出OUT连接到Vsup，Ml，M4关闭。由于Vsup和GND之间没有任何直流路径，因此电平转换器在没有任何电流消耗的情况下保持此状态。

当输入变为低电平时，M3 和 M2 关闭，Ml，M4 打开，将输出连接到 GND。除了完成两种稳态之间转换所需的时间间隔外，功耗再次为零。

电压跟随器配置的运算放大器用于将基准电压发生器与高压开关和行解码器分离。值得注意的是，稳压器不提供高电流裕度值，因为它仅通过行解码器连接到存储单元门。此外，内部算法的编程方式使得所有地址转换（可能产生crow-bar电流峰值）在VPCX = VCC时发生。

因此，稳压器的设计方式必须使其输出电流能够保证VPCX电压的所需建立时间。与任何反馈电路一样，稳压器的稳定性由适当的补偿电路保证。

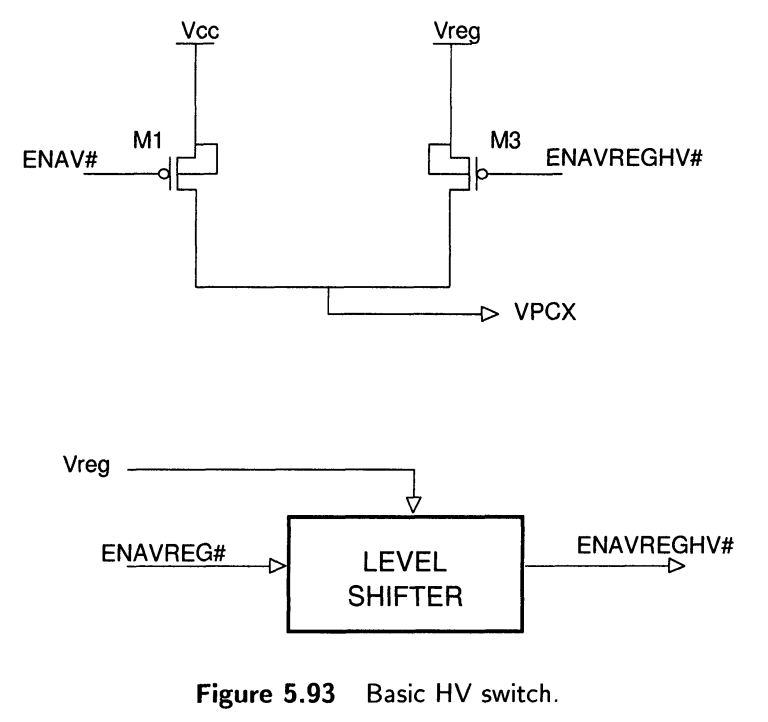
相反，在单电压闪存中，只有VCC电源可用，但通常其值不如VPP精确。 因此，在典型的单电压器件内增加了一个内部基准电压发生器（例如带隙基准电压源）。新型栅极调节器架构如图5.92所示。



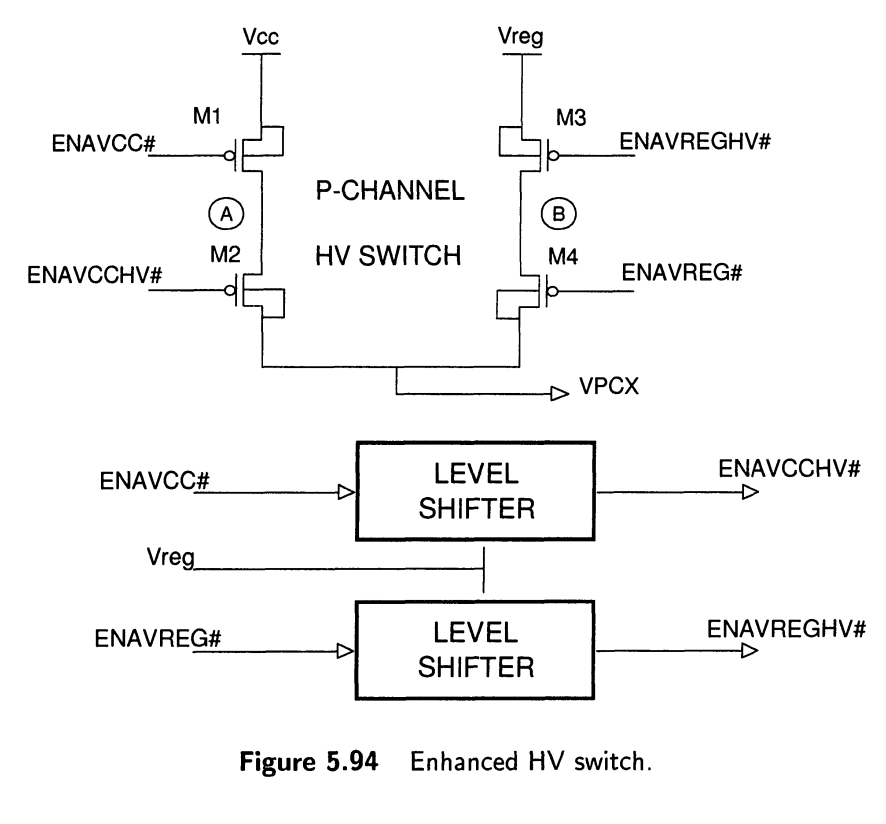
由于带隙输出电压低于稳压输出电压，因此运算电子放大器现在配置为具有可编程增益的非反相放大器。 5.92图中名为Vpp的电源，大于VCC，由电荷泵电路提供，因此稳压器的设计应特别注意限制其电流消耗。 为了降低这种贡献，可以使用MOS电容器实现分压器。

此时，可以考虑高压开关，即为闪存内的行解码器供电的电路（由栅极稳压器直接驱动）;它是一种能够选择两个或多个电源电平的电路，即使在电压大于传统电源电平的情况下（即VCC）。

本电路没有任何功耗;这意味着输入电压电平必须完全传输到输出端。在CMOS工艺中，最简单的解决方案是基于PMOS的HV开关：开关元件是PMOS，其栅极由两个电平转换器驱动，这两个电平转换器由两个输入电压提供（见图5.93）。



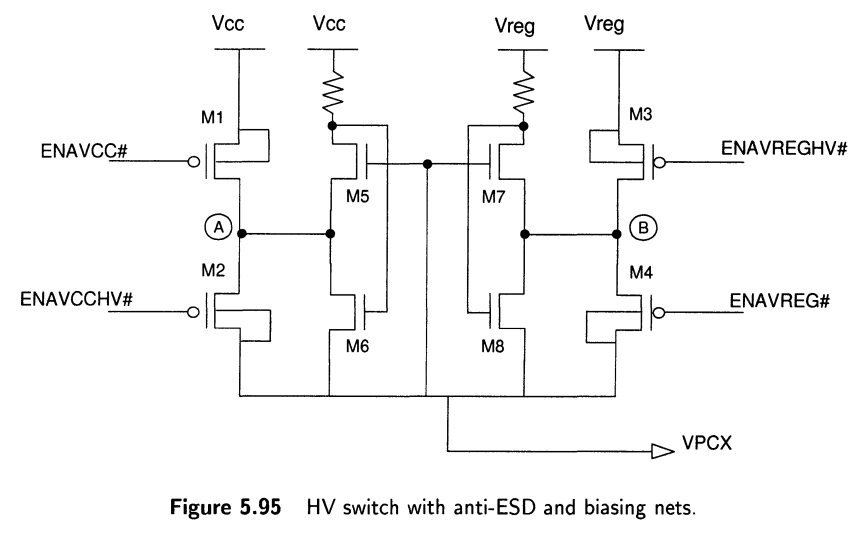
两个PMOS晶体管被正确驱动，但不幸的是，当较高电源传输到输出时，连接到较低电源的PMOS的漏极结直接被偏置。显然，这在CMOS电路中是不被允许的，并且肯定必须采用适当的对策。 图中的原理图 5.94 为此问题提供了一个很好的解决方案。 每个分支由两个具有相反n阱的PMOS组成。如果通过具有正确电压摆幅的信号正确选择栅极，则该开关可以管理所有可能的电源电平组合（VCC >VREG，VCC<VREG，VCC = VREG），没有任何缺点。



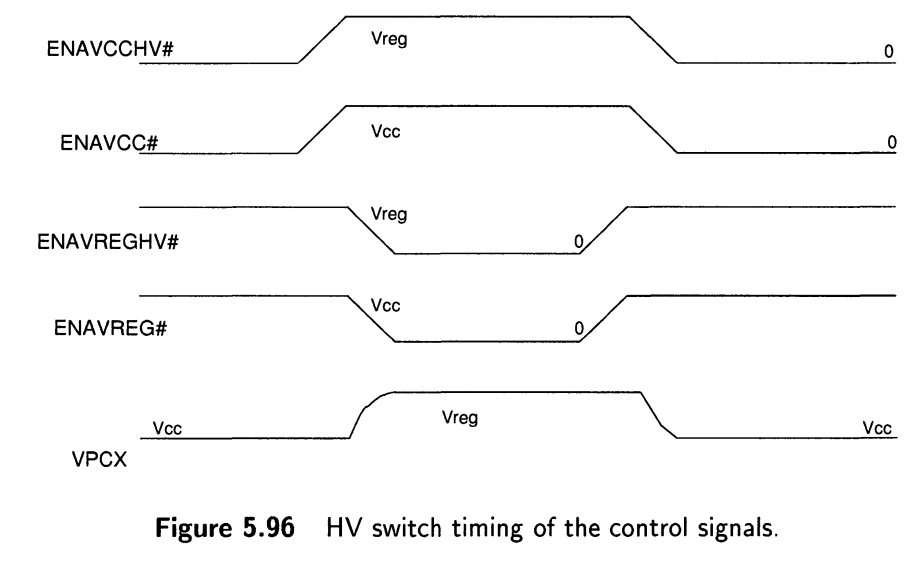
例如，如果VCC低于VREG并且VPCX必须连接到VCC，则通过将两个使能信号ENAVCCHV#和ENAVCC#采用GND来打开左侧分支;然后，节点 A 位于 VCC，并且所有结点都正确偏置. 右分支关闭ENAVREGHV#=Vreg，ENAVREG# = VCC。M3和M4都关闭，因为它们的Vgs = 0，这意味着节点B是浮动的，并且M3和M4的源极和漏极结被正确偏置。

如果必须将 VPCX 设置为 VREG，则开关的右分支将打开，设置 ENAVREGHV# = GND 和 ENAVREG# = GND，而左侧分支将被中断，由 ENAVCC# = VCC 和 ENAVCCHV# = VREG。同样在这种情况下，所有晶体管的源极和漏极结均存在良好偏置，节点A为浮动。

该高压开关的复杂版本如图5.95所示。在此实现中，使用双对NMOS晶体管来保持节点A和B始终偏置。 当VCC传递到VPCX时，节点B的电压等于VCC - VTHN（如果VREG低于VCC并且M7打开）或等于VCC（如果VREG大于VCC并且M8打开）。当 VREG传递到 VPCX 时，如果 VREG> VCC，则节点 A 位于 VCC;如果 VREG< VCC，则节点 A 位于 VCC - VTHN。引入两个保护电阻是为了避免在电源引脚上发生静电放电（ESD）期间M5和M7的n+/psubstrate漏极结击穿。



由于高压开关提供两个电源，因此必须防止两个分支同时打开;当高压开关改变其状态时，这种临界情况可以通过适当的信号定时来避免，如图5.96所示：一个分支关闭，然后通过非重叠的控制信号打开另一个分支。



5.6擦除操作电路

电擦除是闪存的全局操作，应用于内存阵列的整个扇区。这种操作的机理是EEPROM和Flash之间存在的本质区别：由于选择晶体管的存在，前者可以逐个单元擦除，而在后者中（其目标是高密度），公共源是同一阵列部分的所有单元共享。将内存矩阵分区为特定大小的扇区，只允许擦除某些部分，而其他部分保持不变。因此，每个扇区都有自己的内部SL线和自己的电路来切换这条线。 为了有效地执行电擦除，必须在源和电池的浮栅之间提供高电场，通过FN随道从浮栅中提取负电荷;该任务是在源栅和控制栅之间施加大于10V电压差来完成的。

根据所考虑的器件类型，双电源或单电源（见第5.1节），必须设计不同的电路; 在接下来的章节中，将分析这种选择背后的标准，以及对电路结构的影响。

5.6.1双电源方法

如果外部提供高电压可用，则可以立即连接要在Vpp处擦除的扇区的源线和GND处相应单元的栅极。这种设计选择降低了高压内部线路的复杂性，并允许使用最高的压降，而无需任何其他内部操作（分区或调节）。

擦除内部高电压通过SL线通过长度约为10ms的脉冲提供。在每个脉冲结束时，验证单元的阈值电压，如果所有单元都低于所需值，则停止操作。有关擦除算法的更多详细信息，请参见第 5.7.4 节。

由于其高反向偏置，在单元源-sub连接处发生了两种不希望的现象：带间隧穿效应和雪崩击穿。前者是由于n+源区外围的导带和价带有很大的折叠;一些电子穿过两个波段之间的间隙，并以最高电位（即源接触）被端口捕获。后者（雪崩击穿）可能发生在对源-sub结施加高反向压降时。这两种不良效应具有共同特性即从SL线（来自vpp外部源）到GND的电流，该电流在整个擦除脉冲中都是稳定的。



5.6.1.1 源擦除电路。

对于所有擦除脉冲长度，必须将擦除下扇区的SL线从GND级别切换到高电压。在读取和编程操作期间，必须将此线钳接到GND。完成此任务的电路通常称为“源开关”。

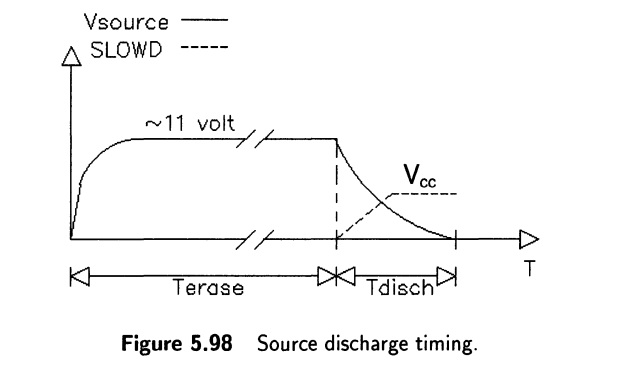
原理方案如图5.97所示。在读取操作期间ERASEl\_N，SLOWD和ERASE2\_N信号处于VCC，因此晶体管MNl，MN2和MN3导通;擦除信号为低电平，以便将高压开关的输出保持在VPP（该电路行为的完整描述可在第5.5.4节中找到），从而关闭MPI。

SL线显示了对GND的巨大寄生电容，如图5.97中标记为Csect。该电容来自不同的贡献，即：每个闪存单元源底结的周长和面积电容，以及SL线在零电位下朝向每个节点的耦合电容（例如，连接单元门的字线）。

对于闪存阵列的最新几何结构，Csect 的估计值约为 2nF/Mbit。在擦除脉冲期间，可以通过引入反向偏置齐纳二极管（图5.97中的DZ）来阻止不希望存在的击穿电流。为了让电路自限电流量，在Sl线和vpp之间放置了一个电阻元件。一种简单的方法是设计具有电阻长宽比的MP1。MPl的Vds压降不允许源线达到完整的Vpp，它将整个扇区的源电流限制在10÷20mA（对于大扇区尺寸和高工作温度，该值可能会增长）。

5.6.1.2 关键节点的缓慢放电。

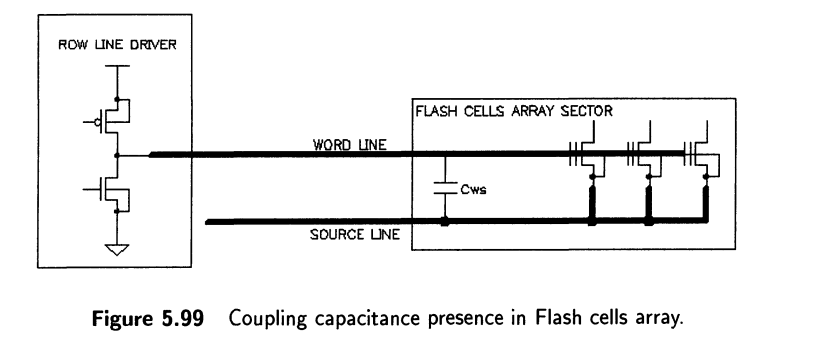
必须通过引入一些预防措施来执行源线的泄放。首先，Csect寄生电容器放电引起的接地噪声很大，它会在内部逻辑网络中引入寄生。为了限制这种现象，有必要减少源线中的放电电流。在实践中，缓慢的放电是根本。在图5.97中，慢放电晶体管为MN2。栅极信号SLOWD（见图5.98）具有缓慢上升的前端，以延迟放电开始时MN2的导通。同时，MNl和MN3通过降低ERASELLN和ERAS2N来关闭。



源线在擦除脉冲期间达到的电压水平大约比VPP低1伏（因为MP1上的电压降）。为了将放电电流保持在一些MA的范围内，放电时间（图5.98中的Tdisch）必须在10到50ms之间。当然，这个时间是被擦除扇区大小的函数。

慢速上升的另一个原因是为了避免MN2的回弹触发效应。这种不希望出现的情况，如图5.81，当施加高漏极和栅极电压时发生；避免NM2反向触发窗口的一种方法是降低其漏极电压（源极线），同时降低其栅极电压（SLOWD）。此外，必须遵守一些设计规则，以减少此晶体管的回弹触发效应，确保非最小沟道长度

字线和源线之间的电容耦合是考虑放电时应考虑的另一个效应。在擦除脉冲期间，耦合电容Cws（见图5.99）在源线擦除电压下充电；当该线路放电时，其他端口，如字线在负电压下驱动。这种效应导致字线下拉驱动器的漏极-B结再次发生正向偏压，慢放电适用于减小该结中的正向电流.



5.6.2单电源电压结构

在单电源器件中，高于VCC的电压和低于GND的电压（闪存单元的编程和擦除所必需的）都必须在内部生成。这是通过所谓的“电荷泵”实现的。这种方法的缺点之一是电荷泵几乎没有驱动电流能力。这意味着必须尽可能地限制从它们那里沉没的电流。 因此，擦除操作在栅极上使用负电压（-8V）执行，从而允许在源电压上降低电压。

请注意，在控制门和单元源之间施加的电场与双电压方法相同，其中这些端口偏置的绝对值不同，但它们在两种情况下的差异是相同的。这意味着源电流要低得多，因为由于源结击穿引起的元件不再存在（只有频带间隧穿电流仍然存在）。

5.6.2.1 电荷泵。

电荷泵的原理是基于在所需节点处存储具有正确符号的电荷。存在两种泵：

正泵，通过在所需节点中存储正电荷来产生高于电源的电压;

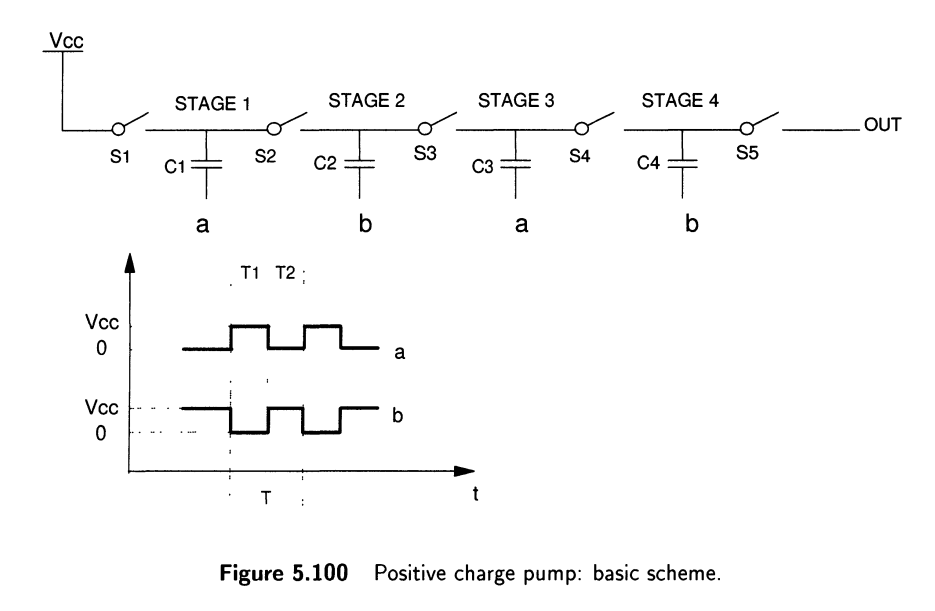
负泵，通过从所需节点中去除正电荷来产生低于GND的电压。

在正泵和负泵中，用于将电荷从一个节点传输到另一个节点的元件是电容器，因为只要考虑小的时间间隔，该元件就具有保持其板上的压降不变的特性。正电荷泵的基本方案如图5.100所示：它包括几个串行连接的级，每个级由一个电容器和一个开关组成。

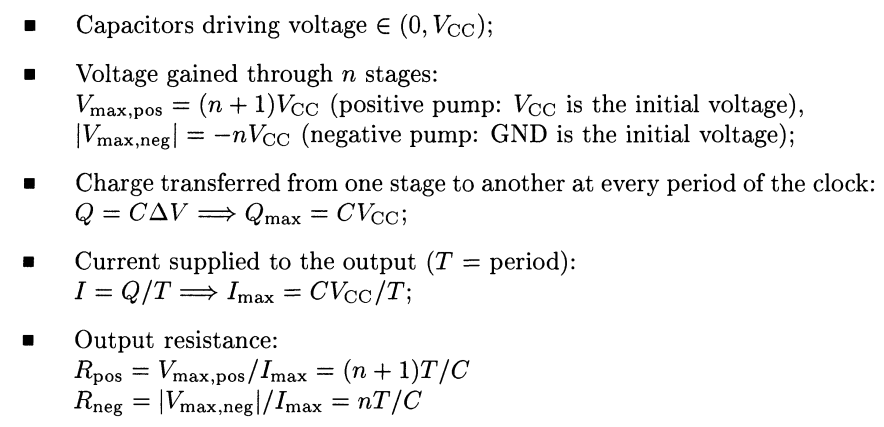
需要两相在GND和VCC之间的相位对立的CLK，图中的“a”和“b”，如图所示驱动电容器。 当相位“a”为高电平而“b”为“低”时（图5.101），开关82和84保持关闭，而其他开关则保持打开状态。 正电荷从阶段1转移到阶段2，从阶段3转移到阶段4。

当相位a为低电平且“b”为高电平（图5.102）时，正电荷从VCC转移到级1，从级2传递到级3，从级4传递到输出。正电荷在输出节点处累积，使其电压增加。在电荷传输期间，阶段i和阶段i + 1的电压均衡。如果电容器具有相同的值，则均衡后的最终电压将是两个初始电压的平均值。

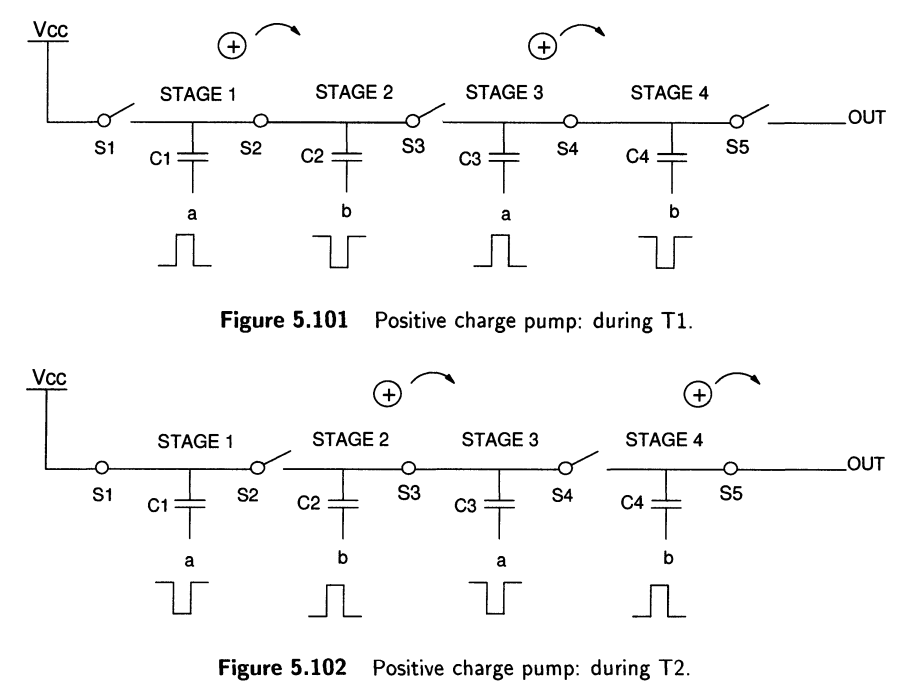
由于输出端没有电流，在初始过渡阶段之后，每一级获得的电压将等于VCC。显然，相同的概念也适用于负泵。

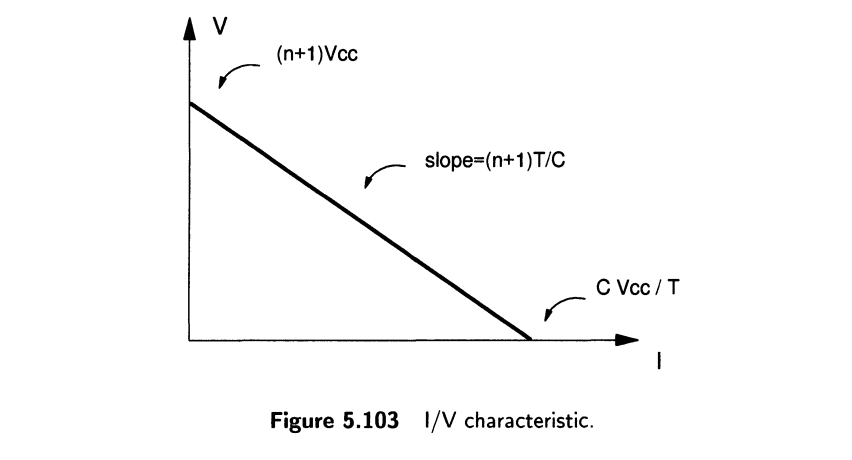


关于电荷泵的基本分析关系如下：



任何电荷泵的主要设计问题是开关的实现方式。当然，开关应该是完全打开或完全关闭的，这根本不是微不足道的。主要问题是MOSFET受到身体效应的影响，并且已经找到了不同的解决方案，具体取决于可用的技术。





关于电荷泵的另一个问题是它们所需的面积。如图5.103所示，输出端所需的电流越高，可达到的输出电压越低; 电压和电流都取决于面积，尽管方式不同：

一旦给出VCC，输出电压仅取决于（不定式地）电容器的数量（即级）n（更多的级意味着更大的面积），而电流与所用电容器的尺寸有关。这导致使用许多具有大面积的电容器，这反过来又对整体布局区域产生负面影响。最后，但并非最重要的一点是，必须面对功耗问题。驱动升压电容器的反向器消耗大量功率，这对总功耗和GND和VCC节点内部产生的噪声都是负面的。

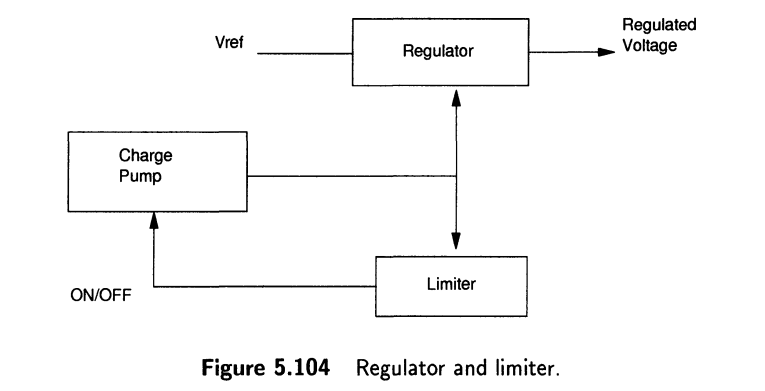
由于泵提供电流的能力有限，因此连接到它们的所有电路必须绝对满足两个要求：

1.它们应该尽可能少地吸收电流（例如，不允许使用下拉电流）;

2它们应尽可能少地引入寄生电容，以避免节点从其初始电压驱动到所需电压的瞬态时间变慢。

5.6.2.2 稳压器

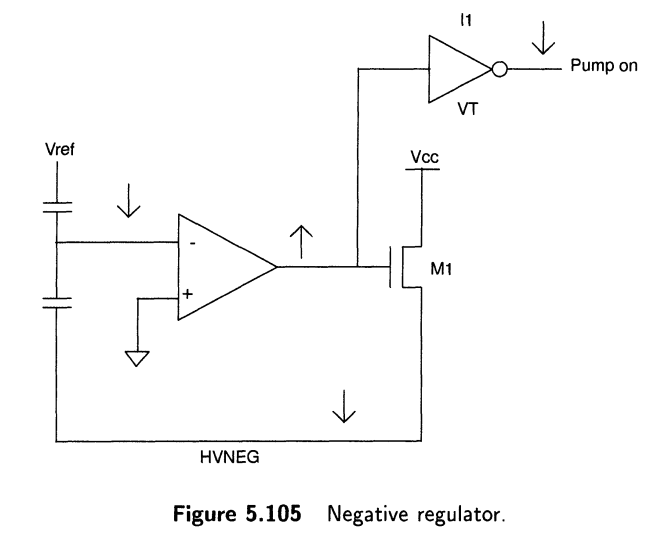
稳压器的目的是限制和调节电荷泵提供的电压。通常，这两个目标通过两个不同的电路来实现：限幅器和精密稳压器（见图5.104），两者都利用了反馈概念。



限幅器对电荷泵的输出进行采样，如果电荷泵的输出变得太高（在模块中），则通过关闭其相位（时钟）来禁用泵。这预防了因电压过高而造成的损坏。稳压器（有时由电荷泵的输出提供）比较基准电压的值必须与要调节的电压尽可能精确。

通常，稳压器的输出级由上拉和下拉组成。 如果稳压高于所需电压，则下拉打开，上拉关闭; 如果稳压电压较低，则相反。典型的负电压调节器如图5.105.所示。

HVNEG是要调节的电压;它通过电容电压采样，并与GND进行比较。使用容性分压器，而不是更常见的电阻分压器，可以避免泵节点的电流消耗，如前所述。当然，电容器的值与VREF值一起决定了稳压。



如果负电压过低（绝对值过高），运算放大器将打开NMOS，NMOS向稳压节点HVNEG提供电流，从而增加其值。 如果这还不够（即运算放大器的输出很高，超过一定限值VT），则限幅器将关闭泵。或者，可以省略Ml，放弃精确的规定。

5.6.2.3 源开关

对于擦除操作，必须将负电压施加到字线（单元的门）上。基本上，使用两种方法：

1.字线二极管（无三阱技术），

2.普通行解码器（三阱技术）。

第5.2.2节已经处理了这个问题：如果可以选择后一种方法，那么很明显，该下拉的源必须切换为GND和负电压，这需要一个特定的（小）电路，在前一种实现中暂不存在。

除了施加到源的不同电压外，单电压和双电压方法之间没有特别的差异。主要区别在于，对于前者，即通过泵提供的源电压和电流（因此具有有限的电流能力），不需要任何元件来限制单元下沉的电流。当然，第5.6.1节中分析的关于关键节点的慢放电的注意事项仍然有效。然而，在单电压器件的情况下，擦除操作是在栅极上的负电压下执行的，这一事实“有助于”线路放电。

由被擦除扇区的单元的栅极引入的寄生电容与相同单元源的寄生电容数量级相同（约lnF）。这些必须在擦除操作后都放电到GND，并且两者都是巨大的。然而，放电源 - 带正电 - 意味着朝向GND的正电流，而放电栅极 - 带负电 - 需要朝向GND的负电流。

这两个电流相互补偿，因此放电操作可能更快，因为GND反弹受到的限制要大得多。尽管如此，必须对这种放电进行一定的控制：两个电流必须同时流动，并且或多或少地以相同的量流动，因此需要恒定的电流放电，并且它们都不应该太大，以避免电迁移和焦耳效应。

众所周知，通过向源施加恒定电流（而不是通常的恒定电压）来执行的擦除操作允许恒定的擦除时间，这与单元的几何差异甚至单元所经历的循环次数无关。

在这种情况下，与图5.97中描述的结构相比，源开关的结构变得更加复杂，因为必须引入电流发生器（图5.106）。这是通过晶体管Ml到M4实现的，它们实现了两个电流镜，镜像了IREF的适当比例。扇区源的电压由单元的源施加。

5.7 控制逻辑和嵌入式算法

与其他方法一样，闪存的不同操作模式可以根据其持续时间进行分类。持续时间这个参数差异很大，读取约为100ns，编程约为10us，擦除约为几秒钟。然而，在大多数应用中，微控制器并不是专用于控制闪存，还必须与其他多个设备进行交互，它不可能在单一或复杂的操作中一直保持忙碌状态（例如，处理一个完整的擦除周期），因为这意味着会导致系统总体性能下降到不能接受的程度，并且使控制软件变得相当复杂。

为了解决这个问题，在设计时，应当把整个处理慢速操作（即编程和擦除）所需的所有逻辑电路都嵌入闪存当中，这样的话微控制器就不必去处理这种耗时的任务。这种方法的另一个优点是接口简单，正确执行闪存的操作所需的所有时序（设置不同的电压、编程或擦除脉冲的脉冲计数、校验等）对用户是透明的，用户只需提供操作代码（编程的数据或擦除的扇区），并在微控制器不执行其他任务时检查闪存的当前操作状态。

闪存的设计一开始并没有遵循这种方法。比较不同设备中嵌入式逻辑的百分比（即排除常规逻辑，如行和列解码器、输出多路复用开关等），假设将第一代典型的闪存中的这个比例作为参考值（设为0%)，可以看到这个比例在持续增加：从第二代闪存的15%到现有最先进闪存的40%。这与新的闪存内部设计方案密切相关，例如，需要外部提供高压的闪存（双电压器件）比内部产生这些电压的闪存（单电压器件）需要更少的控制逻辑，因为后一种设计方案需要在内部设计复杂的高压产生电路。对于后一种闪存，嵌入式算法的优势也更加明显。

5.7.1逻辑架构

如第5.1.3节中所述的闪存可以通过其引脚进行控制，分析设备引脚，可以将其分为三类（电源引脚和GND引脚除外）：地址、数据和控制。通常，控制信号用于激活设备并获得一些简单的操作，如“读取电子签名”、“禁用输出”等。当控制信号与一些数据引脚（通常低8位有效）组合使用时，允许微处理器或微控制器（由于闪速存储器可以与微处理器或微控制器一起工作，在下文中，这两种设备将被无差别地引用）将命令写入用户接口，以便闪存可以执行复杂的操作。写入命令的协议有几种，如图5.107给出了一个示例，图中CODE为编程的指令码，PD为编程的数据，PA为编程的地址。在该图中，数据和地址都在控制信号

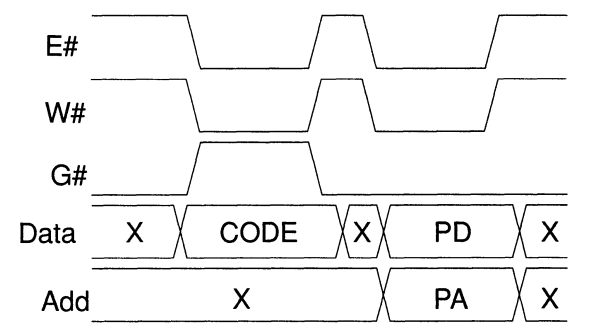


图5.107指令写入示例

W#（写启用）的上升沿上被锁存。另外，锁存阶段可以通过信号E#（芯片启用）、两个信号的任意组合或其他方式来控制。当发生锁存时，数据和/或地址表示特定命令；把用户接口译码的一组命令码定义为一个操作（指令）。

闪存的两个主要操作（指令）是“编程一个字节（或一个字）”和 “擦除一个扇区”。 也存在其他指令，例如“读取电子签名读取”、“清除状态寄存器”等，但最复杂的是与编程或擦除相关的操作。因此，在读闪存的规格说明书时，可以找到如下指令：字节（字）编程，让器件在内存的任何地址执行8（16）位数据编程操作；芯片擦除，让器件擦除所有扇区（顺序的或同时的）：扇区擦除，让器件擦除一个或多个扇区；扇区擦除挂起/恢复，由于擦除可以在扇区子集上执行，擦除时间很长（与微处理器执行时间相比），因此可以暂停此操作，以读取或编程那些不进行擦除的扇区。如前所述，闪存的所有复杂操作都是内部执行的，不需要外部控制器辅助；此外，存储器能够把每个操作的状态和正确性通知微控制器。状态寄存器，或类似的寄存器，在嵌入式算法的执行过程中有记录信息的功能，以便微控制器可以读取它们。

很清晰的，闪存中的控制逻辑都有明确定义要执行的任务，因此逻辑电路必须有很好的时序控制，模拟和数字部分必须协调工作。

图5.108描述了关于闪存架构的简单方框图，简洁到没有画出所有的连接，只显示了主要模块，但这种简图足以展示闪存内的基本电路，也展示了驱动设计的概念；从这个点出发的任何发展都有可能，但基础总是相似的。例如，曾经或将要提到的编程操作并没有提到其实际的实现。对图5.108进行简单的阐述，圆角方框是数字部分，其他的部分并不完全是模拟的；事实上，一些电路主要特性必须是模拟的，但它们也可以实现逻辑功能，或者它们的输出可以驱动其他逻辑部分。

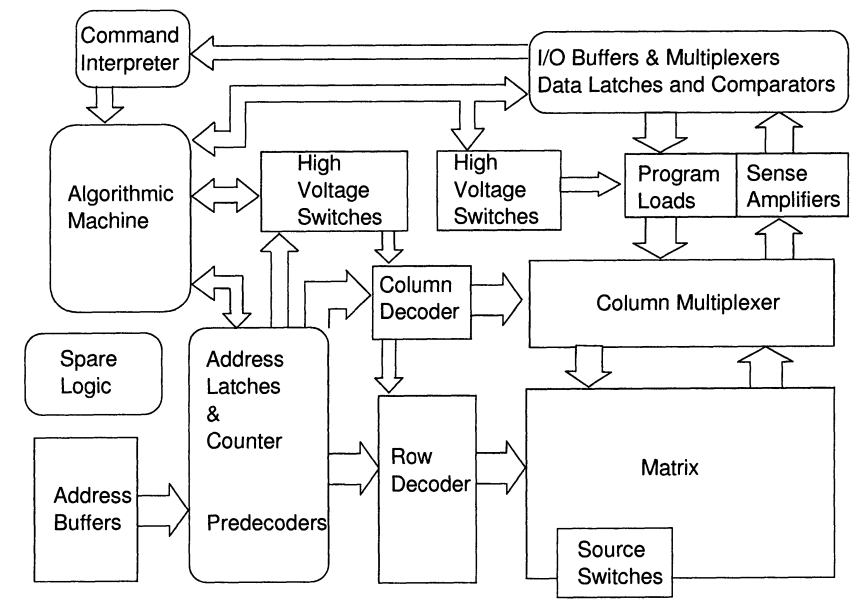


图5.108 模块框图

图5.108描述了前面提到的命令解释器：它的任务是锁存代码，并解析提供的命令序列是否正确；如果正确，这一组信号将生效，以便器件准备去执行指令解码。这个模块用于写入命令并应答微控制器，根据选择的协议，它可以非常简单，也可以非常复杂；通常，这种选择是在代码安全性和节省时间开销之间进行权衡。此外，命令解释器本身可以管理某些指令所需的简单操作，也可以仅作为其他工作电路的代码转换器；当然，在编程或擦除时，控制权须交给第5.1节中提到的编程/擦除控制器。

在分析其结构和行为之前，最好先考虑控制逻辑中涉及的其他电路。首先是地址输入缓冲区和数据输入/输出缓冲区。前者连接到命令解释器（如果指令协议需要使用地址来构建命令），并连接到行和列的预解码器，以选择存储阵列中的位置。后者用于输入命令代码（通常仅使用8位最低有效位），并在编程操作时写入要编程的数据；也用于输出电子签名、状态寄存器和存储器内指定地址的内容。为了执行它们的任务，这些缓冲区连接到命令解释器、程序/擦除控制器、灵敏放大器和编程负载。

在下文中，将考虑闪存内执行的主要操作（即“读”、“编程”和“擦除”），并在其“路径”中漫游。

读路径包括地址输入缓冲区、行和列的预解码器、行解码器、列多路复用开关、灵敏放大器和数据输出缓冲区。输入缓冲区的任务是在外部和内部信号之间设置过滤器，并把TTL输入信号恢复成CMOS电平。数据输出缓冲区指示一个存储器的位置，该位置由一行和8（或16）列之间的交叉指定。行和列的解码器通常采用两个级别结构，用于选择行和列。在确定存储器位置后，灵敏放大器读取存储单元的电流，并将比较的结果提供给数据输出缓冲器，该输出缓冲器作为内部电路和外部数据总线之间的接口。“读”是闪存（以及任何存储器件）的默认操作，不需要控制逻辑；这意味着，一旦给定了存储器地址，整个操作都是自动定时的（实际上还缺少一个模块，就是地址转换检测器，至少有一根地址线发生转换时，就会触发读操作），唯一的控制信号是G#（输出启用），它启用数据输出缓冲区。设计所述的电路也要符合其他操作的要求；事实上，它们并不完全只用于读操作，也允许下文描述的编程和擦除操作。

这两个写操作（编程和擦除都被视为写操作，因为它们会改变存储器的内容）以相同的方式发出正确的代码序列开始构建所需的指令。如前所述，这一阶段由指令解释器管理；控制信号缓冲区和数据输入缓冲区提供输入信号，以构建每个指令；其他输入则来自内部电路的反馈。输出信号仅在写入操作时要求锁存地址输入缓冲区中的地址和数据输入缓冲区中的数据。指令解码后，整个设备的控制传递给编程/擦除控制器（算法状态机），直到写操作（编程或擦除）结束。在这一阶段，命令解释器的行为有多种规范，但基本概念是在编程/擦除控制器给出确认结束之前，写操作不能停止。

实际上，将编程/擦除控制器简化为算法状态机过于简单，因为它还包括：振荡器、计时器、周期计数器、结束计数的逻辑和状态寄存器。振荡器向编程/擦除控制器提供时钟；在一些设计实现中，还有一个相位生成器，它提供非重叠相位。计数器和结束计数的逻辑分别保证算法所需的时间步长（例如等待状态和脉冲宽度）以及编程、擦除中允许的尝试次数。状态寄存器通知微处理器在写操作期间器件的状态以及最终结果。

如果向器件发送了编程指令，除了解码器和阵列之外，涉及的模块还有：高压开关、高压发生器及相关编程负载（为选定存储单元的栅极和漏极提供电压）；灵敏放大器和数据比较器，用于校验编程数据。如果器件获取到擦除指令，除解码器和阵列外，还涉及的模块有：为寻址的扇区（们）提供适当电压的电源开关；地址计数器、灵敏放大器和数据比较器，以校验擦除的扇区（们）。在擦除脉冲后，地址计数器逐字节（或逐字）遍历整个寻址的扇区，对扇区内容进行校验。所有其他提及的电路通常是模拟电路（见第5.5节），但无论它们是如何构建的，都需要逻辑控制。事实上，它们的输出必须有合适的时序或选择，以便阵列在开始的时间接收到正确的电压，或将其输出反馈给编程/擦除控制器。例如，在编程校验后，灵敏放大器的输出和锁存的数据进行比较，编程/擦除控制器读取这个比较结果，以决定是否需要启动另一个编程脉冲。在这种情况下，逻辑控制无论在读或比较时，都必须启用灵敏放大器。最后一个还未提及的逻辑部分是输入/输出多路复用开关。闪存可以和8位或16位（通信的位宽）的设备结合使用，因此需要字节或字模式。使用控制信号（通常称为BYTE#）可以在两种模式之间进行动态切换。当设备使用字模式时，多路复用开关是透明的，所有16个数据输入/输出缓冲器都工作，直接连接到16条内部数据线。当选择字节模式时，仅使用8个低位数据输入/输出缓冲器，多路复用开关根据另一个地址引脚将这些数据连接到内部低字节或高字节。

5.7.2嵌入式算法

至于一种算法的硬件实现，为了获得一个控制单元，它能够执行算法本身所需的正确序列，可以遵循不同的设计方法。主要可能的方法如下：

* 序列发生器（伪微控制器）；
* 有限状态机。

5.7.2.1 序列发生器（伪微控制器）

由于外部的微控制器不能完全适用于闪存，最简单的解决方案是在闪存内有一个类似模块。嵌入式逻辑由两个独立的部分组成：

1. 一个ROM（Read-Only Memory，只读存储器），其中控制程序以一系列指令存储（其指令集通常比常规微控制器的指令集更小、更简单），包括解码和采样。
2. 一个指令解码器，其任务是在其他指令后获取指令并解码，然后驱动相应的各种控制信号（例如设置特定电压、设置脉冲宽度等）。

如果控制流程需要修改，则只需更改ROM的内容就足够了（只修改工艺流程中的少数几层）；另一方面，指令的行为（和数量）是严格固定的，要修改的话，就要完全重新设计解码器。

5.7.2.2有限状态机

在控制程序和解码器没有分离时，状态机流程是逐步执行的，在每个不同的状态下，根据序列的实际状态驱动一组适当的控制信号。有限状态机的灵活程度取决于硬件实现：

* 寄存器+组合逻辑：即使最小的修改也意味着模块要全部重做布局。其优点是这种方案可以使用自动综合工具来实现。
* 寄存器+PLA（可编程逻辑阵列）：该方案需要更复杂的时钟处理（因为PLA必须以预充电和再计算的方式实现），但PLA内容（即流程）修改可以通过改变工艺流程中的少数几层来轻松实现。缺点是该块的布局是“单片集成”的，很难整合到自动布局的环境中。

此处描述的流程涉及通用单电源闪存；但除了与电荷泵必要性相关的问题外，相同的概念也适用于双电压器件的情况。

5.7.3编程流程

编程指令解码成功后，器件就存储好了编程数据和目标地址。编程算法工作如下（见图5.109）：

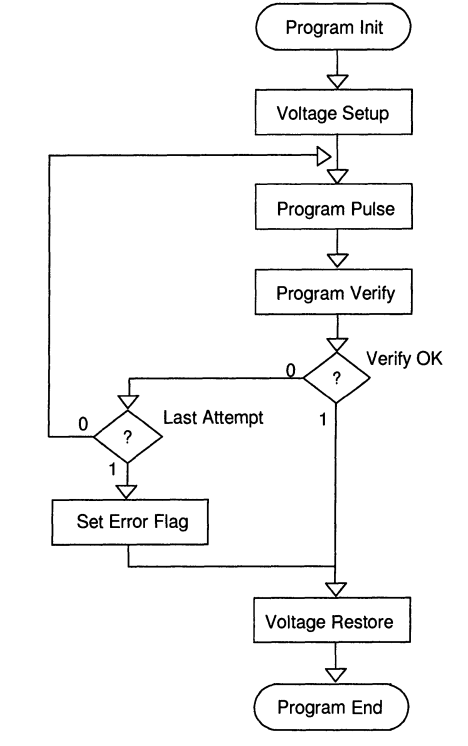


图 5.109 编程流程图

编程初始化：执行启动流程（例如计数器重置）。

电压设置：启用电荷泵将电压从*V*CC升高到编程电压。

编程脉冲：在编程所需的时间内（通常为几微秒），向选定的存储单元提供编程电压。在脉冲结束时，电荷泵准备好切换到校验阶段所需的电压。

编程校验：这个流程就是在选定地址执行一种读操作，此时偏置闪存单元的栅极电压远高于正常读操作中通常提供的电压。如果读取的数据与编程数据一致，则意味着（1）闪存单元的阈值电压已从原始（或擦除）值上升到编程值，（2）由于在较高电压下进行的“读”操作的结果是正确的，因此即使在常规的读操作中，也足以检测到阈值差。在这种情况下，算法已成功完成：电荷泵被关闭，控制权被传递回命令解释器。如果检验失败，则可以继续进行多次的编程尝试（其次数取决于特定计数器中存储的值），或者编程算法结束（再次关闭电荷泵），产生用户可以检测到的故障信号。

5.7.4擦除流程

为了简单起见，这里介绍单个扇区的擦除：如果执行多扇区擦除或芯片擦除，则采用相同的考虑和算法。如图5.110所示，流程由以下步骤组成：

擦除初始化：执行启动流程（例如计数器重置）。

全部编程为0：在此步骤中，对扇区中所有的低阈值单元（逻辑“1”）进行编程（即，将状态更改为“0”），以便在启动擦除脉冲之前，它们的特性尽可能相似。假如是它应用于扇区的每个单元，算法就和编程单个字的算法完全一样（见第5.7.3节）。

擦除脉冲：整个扇区在适当的时间（几毫秒）内施加擦除电压的偏置：在此期间，通过Fowler-Nordheim隧道效应来擦除单元。

擦除校验：这个流程就是在选定地址执行一种读操作，此时偏置闪存单元的栅极电压远低于正常读操作中通常提供的电压。如果读取的数据为“1”（单元是擦除态），则意味着（1）单元的阈值电压已设置为擦除值，（2）由于在较低电压下进行的“读”操作的结果是正确的，因此即使在常规的读操作中，也足以检测到阈值差。如果检验失败，则可以进行多次擦除尝试（其次数取决于特定计数器中存储的值），或者擦除算法结束（再次关闭电荷泵），产生用户可以检测到的故障信号。如果扇区的每个单元都校验正确，则可以执行以下步骤。

耗尽校验（过擦校验）和软编程：理论上讲，擦除脉冲后，扇区中的所有单元应具有相同的特性；但不幸的是，由于存储阵列是非理想的，单元的初始阈值也有差异，整个阈值分布会出现离散。前面提到的擦除检验是为了重新迭代启动擦除脉冲，直到所有存储单元都被正确擦除：问题是如果一个单元已经被正确擦除，下一个擦除脉冲可能会将其阈值降到零以下，这样即使没有施加偏置电压，也会产生电流（即耗尽型单元）。通过耗尽校验，可以检测到耗尽单元的存在并对其进行轻微的重新编程，将其阈值提高到零以上。如果这个操作在指定的尝试次数内完成，则擦除执行成功：否则，将发出失败信号通知用户。

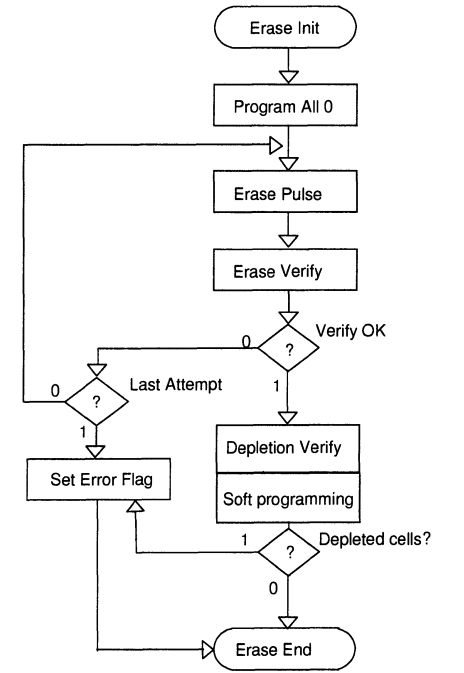


图 5.110 擦除流程图

5.7.5擦除挂起擦除恢复

如上所述，典型的擦除周期可能持续几秒钟。在某些情况下，必须同时访问存储器本身，从而中断算法。为了执行这种任务，微控制器应发出擦除挂起命令：在挂起期间，可以对器件进行读或编程。在实现这种功能时，考虑的主要问题是必须保存器件的当前状态，以便准确地知道擦除流程在哪个位置中断。因此，应设计足够的存储系统用于保存主要参数；此外，器件必须将功耗降低到标准的读操作，因此须关闭所有的内部泵，将内部电压降低到*V*CC。最后，该器件可进行读或编程操作。

解决“中断”后，擦除操作允许重新启动。首先，必须恢复先前的状态，以便算法准确地知道已经做了什么。之后，可以让电荷泵恢复暂停阶段特有的电压，并恢复和执行算法，直到擦除结束或执行另一个擦除暂停命令。

5.7.6可测试性问题

在工艺寿命早期，良率可能很低，但在工艺固化后，生产的良率应该很高；此时，器件的功能变得非常重要。对于集成器件，这几个字描述指出了测试流程的主要任务。首先，通过测试器件，必须发现并解决工艺问题，以加快开发并缩短上市时间。其次，必须测试产品规范中所述的所有特性和参数，以确保在规定的条件和使用年限内具有完整的功能。最后，通过对实际器件进行适当的测试，可以发现未来的开发方向。将问题聚焦在闪存上，将通用测试流程适用于此类器件并不容易，因为供应商和用户都要求闪存有多个复杂的功能。这里指出了主要需求：由于硅的成本，器件的面积必须可能小；为了允许使用低电压电池，功耗必须最低；为了连接高速微处理器，访问时间必须最快；为了减少生产时间，生产中的测试时间必须最短。如果收集了所有这些需求，并且在每个器件内部实现，那么将导致器件面积很大，测试时间很长，最终器件无法销售。通常，必须找到一个折衷方案，以满足供应商和客户的要求。这里还必须提到设计师，因为他们的工作介于供应商需求（即在小面积内进行所有可能的测试）和客户需求（即每个器件工作满足规范）之间。这些考虑促使我们在不影响面积和性能的前提下设计测试模式电路。很明显，每一个电路的设计都是测试特性与标准特性相结合，如果可能和必须的话，只严格地添加必要的晶体管：完全用于测试的电路必须尽可能少。这种选择使得器件面积很小，但有时不可能实现；因此，引入了专用控制信号。添加的信号通常会导致先前快速电路的速度变慢，因此最好在不需要快速的电路上添加测试功能，但这可能还不够。另一种可能方案是使用一些专用于测试的电路，即在用户模式下使用标准电路，但测试模式切换到专用的测试电路。最后，设计将符合以下规范以及之前的考虑。现在让我们试着给出一些细节。第一代的闪存与EPROM（Erasable Programmable Read-Only Memory，可擦写可编程只读存储器）类似，这意味着差不多整个内部电路都是模拟的。如果将内存视为存储数据的器件，则不需要其他电路。要测试的内容包括单元（基本存储元件）、整个阵列（由行和列组成的单元组）和输入/输出电路。首先，了解储存单元的特性（读、编程和擦除特性）非常重要；这些测量对于监控工艺和修复电路非常有用。通过收集这一阶段的结果，可以进行统计研究。标准、生产、测试旨在确保整个设备的功能。在这种情况下，涉及的电路包括解码器、用于编程和擦除的高压开关、灵敏放大器、输入和输出缓冲区。每一个可能发生的故障都被认为是警铃：使用特定的测试模式测试存储单元的其他特性并确定其原因（每个项目都有许多测试电路，仅在非常特殊的情况下使用）。我们已经看到，该器件变得复杂，也可以专用于某些应用。实际结果是，在最初“纯模拟”器件的相同硅体上，设计了纯逻辑电路。所有电路都围绕阵列工作，其主要任务是读、编程和擦除存储单元，但也有一些纯数字电路主要用于管理用户接口和嵌入式算法。它们自己的功能可以在阵列以外进行测试，因为它们可以独立工作；因此，要回答的第一个问题是，已知的测试概念是否可以直接应用。关于概念本身，答案是肯定的，但如果我们关注它们的实现，又是负面的，这种说法是因为很多简单观点的复杂混合；每一个理由对于这个答案都不是很重要，但所有这些理由加在一起就非常有力。关于面积的考虑再次出现：测试的经典概念（扫描、自建测试等）需要锁存网络中要分析的每个节点。但不幸的是，内部信号上只有很少的锁存器，因此引入缺少的锁存器会导致面积大大增加。另一个观点是关于测试的目的。这里不需要进行完整的功能测试：电路设计用于在特定条件下执行某些操作。生产测试必须确保规范所述的完美功能，即使设计可能可以在更大范围内工作。最后一点是关于数字和模拟部分之间的连接。虽然数字电路完全可以与模拟电路分开工作，但实际上这两部分是结合在一起工作，执行两个独立的测试流程并不现实。这意味着，首先激活测试模式的方式必须是唯一的；然后测试数字电路驱动模拟电路的能力；如果任一个电路出现故障，则应停止测试流程。一旦明确了测试的目的，就可以在闪存中使用一些关于激活测试模式的说明。显然不能让用户知道进入测试模式的方式，否则可能有关电路的许多设计会被发现。此外，激活测试模式的协议必须足够复杂，以防止用户意外进入，但也不能太难造成测试时间变慢（事实上，在测试流程中，会重复多次进入测试模式的序列）。

5.8冗余和纠错代码

以下分析的目的是建立闪存的生产良率与插入此类器件的冗余单元数量之间的关系。

5.8.1良率

存储器的生产必须解决工艺良率的问题。很明显，“良率”一词是指在给定的生产批次内，按照规范可以正常工作的器件数量。换句话说，它表示在零时刻器件可以正常工作的比例。验证闪存这类复杂器件的成品率是一个至关重要的问题，它会影响设计的目标。这意味着必须在设计流程的每个步骤中采取适当的措施提高良率。尤其重要的是，设计应考虑到工艺缺陷，并采用电路的解决方案减少其负面影响。为实现这个目标，有必要获得芯片上最常见缺陷的完整信息。例如，针对存储器，统计研究表明，良率的损失主要在存储阵列；很明显，因为在这种器件中，芯片面积中相当大的部分是存储阵列。这里发现的最常见缺陷类型可分为两类：

1、相邻行的金属短路。

2、功能异常的存储单元。

为了克服第一种失效，设计者在阵列中使用额外的行（“行冗余”），而在另一种情况下使用“列冗余”。无论实际实现方式是什么，这两种情况的想法都是用冗余资源替换有缺陷的单元。这种策略的缺点是，器件越大，管理冗余的实现方案就越复杂，因为冗余资源的选择就是比较外部地址与用于存储故障单元地址的寄存器的内容。

最常用于描述良率评估的统计模型是使用“泊松（Poissonian）”近似，其表达式如下：



其中*Y*tot是总良率，A是器件面积，D是每单位面积的缺陷数量。

“原始良率”*Y*0是指没有进行任何干预的良率；此次分析中，它作为自变量用于评估冗余系统的性能。第一步操作是将主要良率分解为外围电路良率（*Y*p）和存储阵列良率（*Y*m0）。

这一步是必要的，因为冗余仅对后者有效，因为引入冗余的控制电路不具有成本效益。

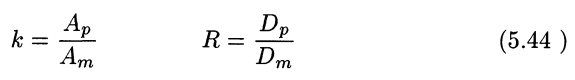
进行这种分解的正确方法是假设缺陷的数量是恒定的；假设*A*p和*A*m对应器件外围和阵列两部分的面积，*D*p和*D*m对应缺陷数。我们可以写为：



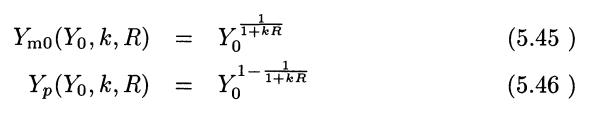
通过将这些关系代入公式（5.41），我们得到：



现在趁机引入以下二维参数：



因此，我们现在可以得到以下形式的*Y*m0和*Y*p良率：



我们对Ym0的良率特别感兴趣，因为可以关联单元失效率p。我们将假设不同单元上的故障事件是独立的，而忽略任何不属于单元的故障原因。

根据这些近似值，阵列的相对良率可表示为：



在下一段中，我们将利用这种关系来估计冗余系统对良率的影响。

5.8.2静态冗余

首先，我们将参考列冗余，而没有使用行冗余，后面将对此进行解释。

设p为一个块中单个单元上的错误概率，即单元上发生故障的概率。此外，我们假设：

*N*c 一个块中的列数；

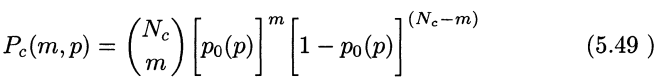
*N*r 一个块中的行数；

*N*red 一个块中的冗余列数。

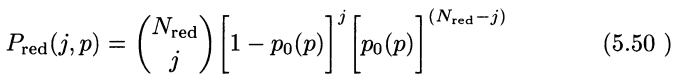
我们现在可以在一列中计算错误概率；由于后者由*N*r位组成，因此概率可以表示为：



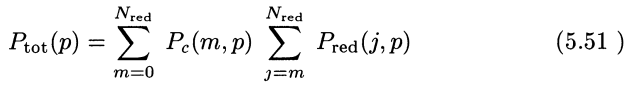
如果我们假设m列，0≤m≤*N*c，每个列至少有一个错误，我们得到：



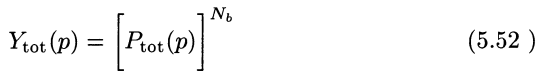
同样，我们可以获得无错误的冗余列数，其中0≤i≤*N*red：



我们现在必须确定可以被纠正的故障情况。我们指出，只要有效的冗余资源数量超过阵列中失效单元的数量，我们就可以修复具有故障列的扇区；因此，我们可以修复一个扇区的概率可以通过以下公式得出：



这是对于一个给定块的纠错表达式；现在，如果整个存储器阵列由*N*b个块组成，则总修复概率的计算就是单个事件的交集。同样，如果我们假设不同扇区上的缺陷相互独立，那么我们得到：



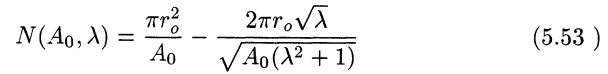
在前面的等式中使用符号*Y*tot是合理的，因为后者可以解释为可修复芯片数量与芯片总数的比率。这个关系可以明确地通过中间参数*Y*m0和*Y*p将*Y*0与*Y*tot联系起来，并能够定量研究通过使用冗余实现的良率提升情况。

5.8.3 晶圆的良率

将*Y*tot描述为初始良率*Y*0的函数的曲线表明，良率最终会饱和；换句话说，当主要良率足够高时，引入更多冗余资源只能略微提高最终良率。

除了这些考虑之外，值得注意的是，芯片的几何图形是矩形的，但晶圆的几何图形是圆形的。当器件的面积增加超过一定限度时，一个晶圆上获得的器件数量就会减少，因此在使用冗余后，必须比较良率增加与每个晶圆上芯片数量减少的关系，以确定冗余数量。

具体而言，以下关系给出了一个“可用半径”为*r*o的晶圆上获得面积为*A*0的芯片数量：



其中λ是芯片纵横比。

从N(A0，λ)可以得出以下两个量：

1. *N*sr：未采取良率提高策略时正常器件的数量



1. *N*cr：使用列冗余后正常器件的数量



我们指出冗余的使用不仅增加了阵列面积，而且还增加了外围电路面积，因为必须在芯片内部引入更多的控制电路。

5.8.4一个实例

为了应用我们之前的考虑，我们以8Mbit容量的闪存为例，该闪存由16个扇区组成，每个扇区包含512kbit。

每个扇区由2048列组成，按4列分组（我们称为“局部位线”），形成512条“主位线”。

这种结构的一个优点是，相邻主位线（金属层-2）间保持四个单元的宽度，从而放宽了这个层次的限制。

由于列使用了金属层-1和金属层-2，因此行将采用多晶硅层实现；因此，虽然这种选择将影响行切换时间，但它将允许设计者避免使用行冗余，因为对于集成技术的实际标准，短路发生在相邻的两行多晶硅之间的概率非常低。

每条局部位线通过11位地址寻址：其中7位地址用于识别属于同一字的列，而其他地址用于在所选字中选择所需的列。显然，一条主位线是通过9位地址寻址的。

行地址



列地址



在器件中，故障列地址存储在称为“UPROM”（即不可擦除可编程ROM）的非易失性寄存器中；其中一个（“引导”UPROM）将指定UPROM信息是否有效。因此，UPROM（*N*UPROM）的总数将为：

*N*UPROM = 12*N*red *N*b 局部位线 （5.58）

*N*UPROM = 10*N*red *N*b 主位线 （5.59）

其中，*N*b是基于冗余资源把阵列分解成块的数量。值得注意的是，后者与行业的数量并不一致。因为冗余不一定与每个物理扇区相关。

外围电路增加的面积公式如下：



其中，*S*f是器件的缩小系数，*A*UPROM是一个UPROM单元和相应控制逻辑占用的硅面积。考虑到冗余列与其他列相同，可以轻松计算阵列增加的面积。



因此，我们最终可以获得*N*sr和*N*cr对冗余列数的依赖性，以研究恢复的可能性。

对于8Mbit存储器，我们设计了一个冗余系统，它可以替换8个失效的局部位线和另外40个列，这些列被分为4组，每组10列；这个方案是为了在灵活性和占用的面积之间做合理的权衡。和普通单元相比，为了避免增加冗余的访问时间，这里使用了一个冗余专用灵敏放大器。缺点是我们无法纠正同一个字中出现的一个以上的错误。

图5.111和图5.112显示了将冗余列连接到灵敏放大器的电路。信号*Y*Oi对应于*A*1-*A*0地址的解码，而*X*Mri和*Y*Oi是根据访问单元的地址和UPROM内容的比较产生。在进行主位线的冗余替换时，*Y*MRj仅用于选择，而*Y*Ork用于局部位线替换。

图5.113和图5.114中的数组RDC[3:0]指定字中必须替换的列。

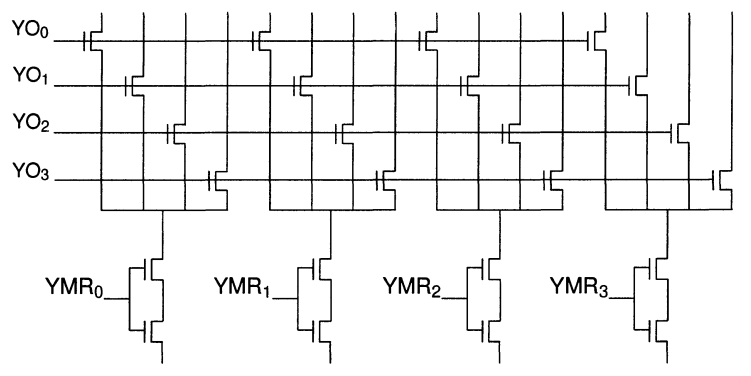


图5.111 冗余列 – 1

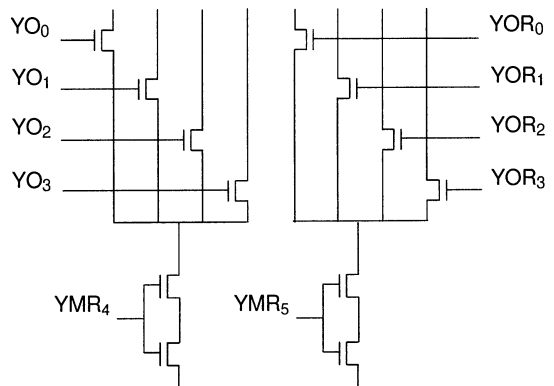


图5.112 冗余列 – 2

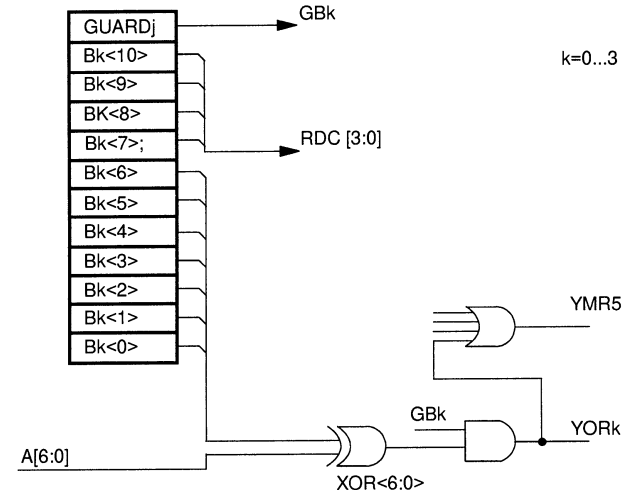


图5.113 局部位线的选择信号

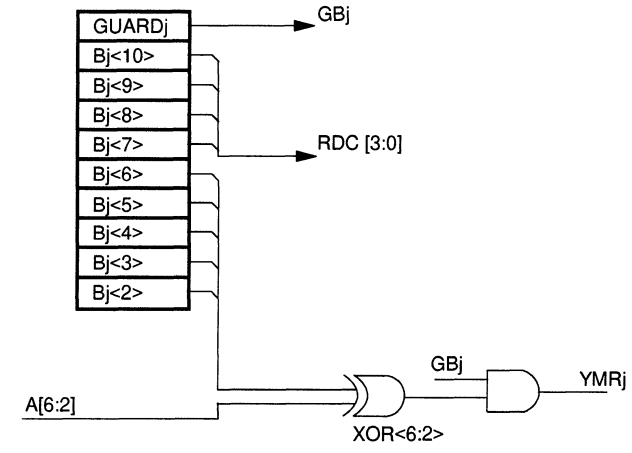


图5.113 主位线的选择信号

5.8.5纠错码

静态冗余的一种可能替代方法是使用纠错码。

目前，这个机会纯粹是理论上的，因为市场上的大多数器件都没有使用这一技术。此外，无论在哪里使用纠错码，都是出于可靠性考虑，而不是预期的良率收益。

然而，可以将静态冗余和纠错码这两种替换方案进行比较，以阐明它们的特性和局限性。

一般来说，我们可以将两个有限集合，即域（或“源字母表”）和共域（或“代码字母表”）之间的任何应用程序定义为“代码”。

任何代码背后的基本思想都是“信息冗余”的概念，即该代码实际能够传输比严格要求更多的信息；因此，信息编码的主要目的是引入一定数量的“冗余”。为了定量描述“代码”冗余的概念，有必要引入两个字之间的距离概念，即两个字间差异的位置数。

根据这个定义，很明显，两个字之间的差异越大，代码就越冗余。

例如，汉明码是用于单个错误纠错的最简单的码之一。其基本参数之间的关系如下：

n = 2m – 1 码字长度

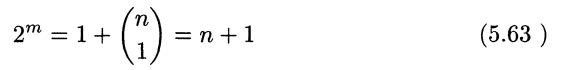
k = 2m - m – 1 信息位数

m 冗余位数

为了纠正*t*个错误，代码中的最小冗余位数必须符合所谓的“汉明下限”。



上述不等式可以解释如下：左侧表示奇偶校验位的可能不同配置的数量，并且必须大于右侧数量，右侧数量是最多改变*t*位错误的事件数。如果*t*=1，关系式可以简化为：



这个关系在之前已经指出。

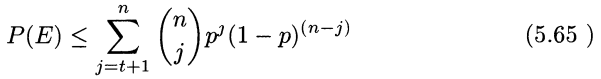
值得注意的是，这些代码严格得引入了必要（和充分）的冗余位，因此它们可以被称为“完美”代码。

对给定代码纠错能力的评估是为了评估剩余错误概率，即错误超过代码修复能力的概率。

例如，假设我们的信息通道损坏每个输入的概率（*p*）相同，并且它不会记忆以前的事件（我们称之为“对称二进制通道”）；我们可以描述*n*位-字上出现*j*个错误的概率，并给出公式如下：



错误数j超过代码恢复能力t的概率可以用以下不等式表示:



这里使用了不等式符号，因为除了完美的代码外，任何距离为2*t*+1的代码都可以纠正距离超过t的一些错误。

基于这一点，有趣的是在不可恢复概率的基础上比较可供选择的列冗余和代码冗余的方案；在图5.115中，我们绘制了

1. 为了实现静态冗余，1Mbit（210行210列）块有四个额外列；
2. 一种缩短的汉明纠错码，具有27个信息位和23个奇偶校验位。

我们现在指出，汉明码解决方案比静态方案需要更多的面积：事实上，在前一种方案，我们每136位需要有8个专用位，对应需要增加5.88%的面积；而在后一种方案，增加的面积仅仅是整个阵列的0.39%。在面积相等的情况下，可以进行另一个比较。在图5.116中，我们比较了前一种汉明码解决方案和60列静态冗余方案。

虽然到目前为止，我们还没有考虑这两个系统的实际实现问题，但图5.116显示，对于超过静态方案极限（*p*），汉明校正码是有效的。

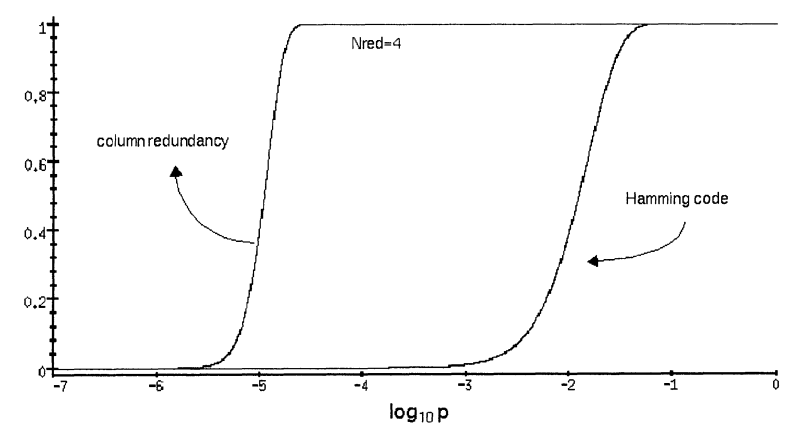


图5.116 不可修复情况的曲线

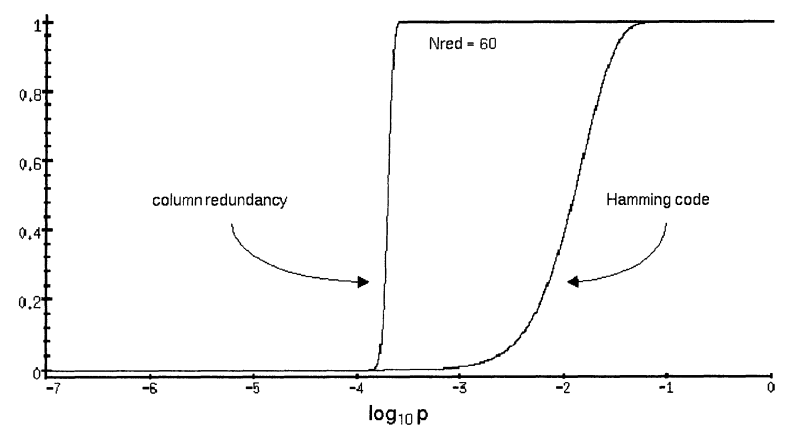


图5.116 不可修复情况的曲线