蛋蛋雄文：Flash Programming那些事

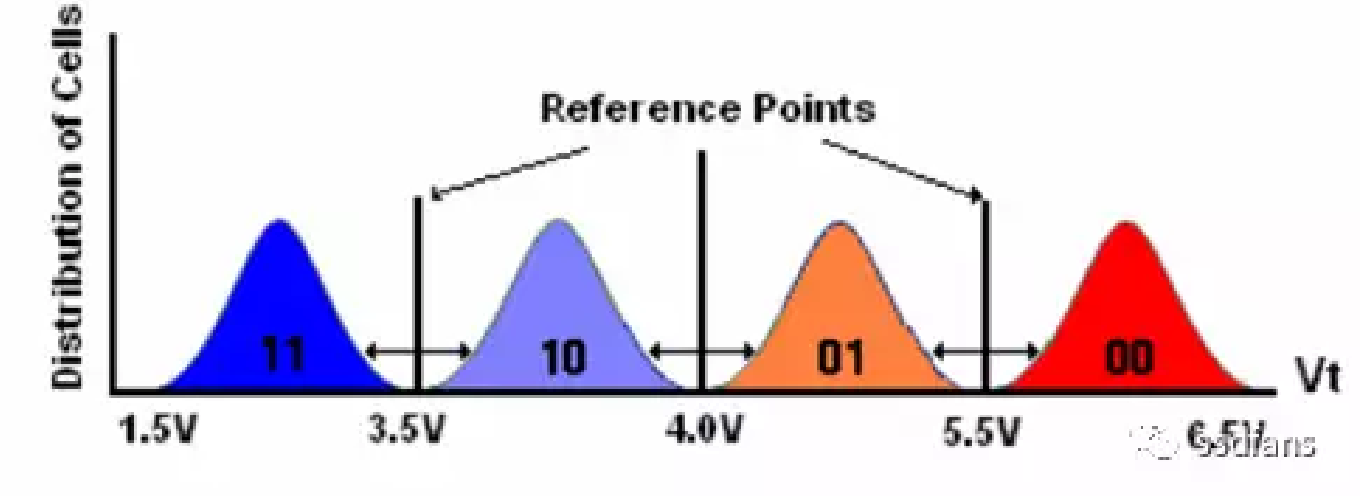
2017-07-11 蛋蛋 ssdfans

作者 蛋蛋

给SSD Fans原创投稿，拿>=100元稿费。微信搜nanoarch加阿呆哥好友，拉你进微信讨论群和蛋蛋切磋。

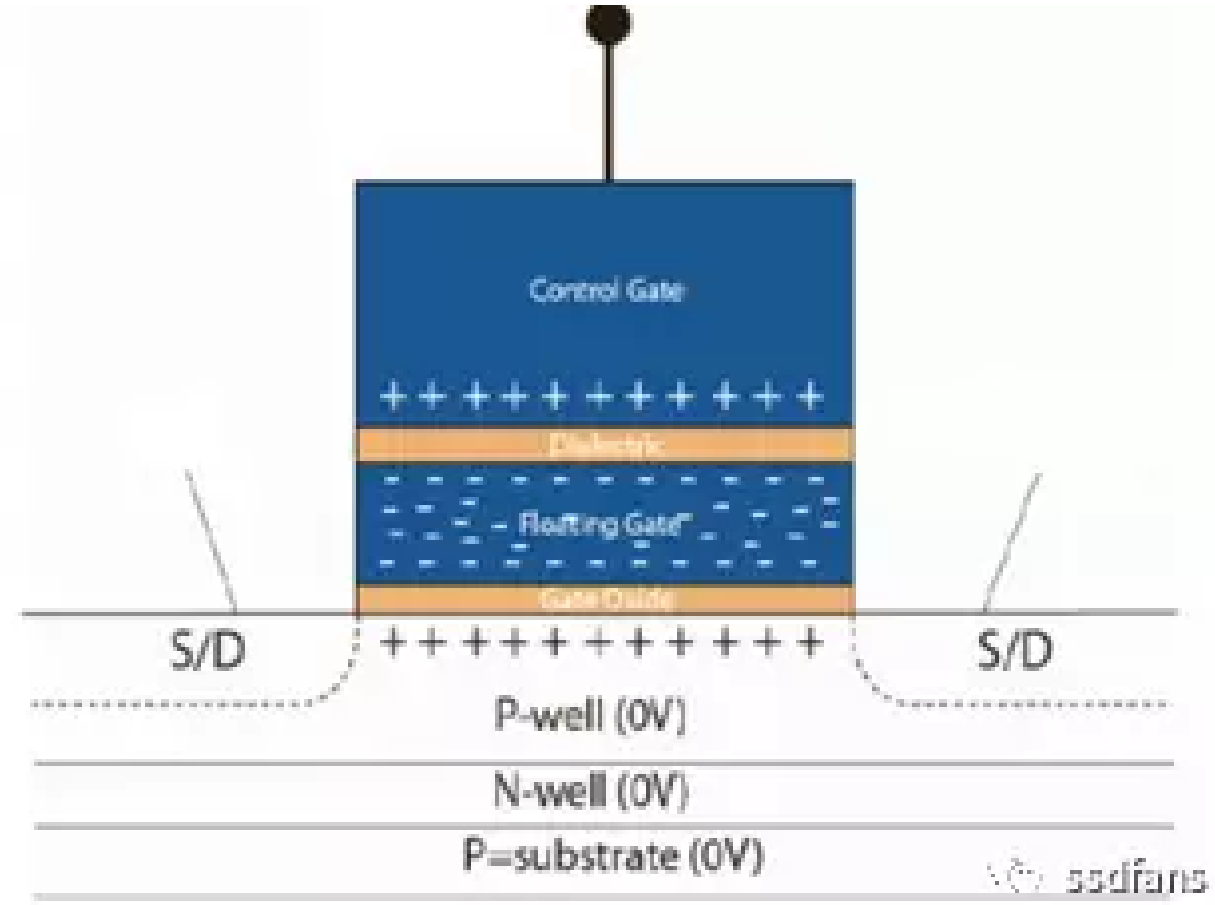
我们以MLC为例，来说说Flash Programming那些事情。（注：没有flash基础的读者，可以先看看《闪存基础》。）

所谓MLC，就是一个存储单元能存储两个比特信息的flash，有四种存储状态：11，10，01和00。



很多人对上图似懂非懂，这里先解释一下这个图。

Flash基本的存储单元（cell，注：后面存储单元和cell会混着说，记得两个是指同一个东西）是改造过的晶体管：



从上到下，依次为控制极、绝缘层、浮栅极（导体）、绝缘层和衬底，其中数据存储在浮栅极。往浮栅极注入电子的数目不同，其阈值电压（Vt, 前面那图的横坐标）也不

同。什么是阈值电压？就是往控制极上加电压，如果控制电压大于阈值电压，那么该晶体管就导通，否则就截止。

横坐标是Vt，那么纵坐标是什么呢？一个Page及其对应的Page（它们共享Cell）写完后，每个Cell可能处于上面四种状态的任一状态。假设一个Page大小为16KB，不考虑Spare空间，那么一个Page至少有16KB/2b = 16Kb*8/2b = 65,536个cell。以前面那个图为例，往该Page所在的Wordline上（控制极）上施加0-6.5V电压，比如0，0.1，0.2，0.3，0.4，…, 6.5，当施加0.1电压时，有100个cell导通，说明有100个cell的阈值电压低于0.1，当施加0.2时，有150个cell导通，说明有50个cell的阈值电压介于0.1和0.2之间。通过这样的

操作，对每个阈值电压（离散值），就能得到有多少个cell具有这个阈值电压，从而最后得到上图的阈值分布图（Vt distribution，阈值电压与其对应的cell的个数）。
需要说明的是，不要纠结上图中的横坐标那些具体的阈值电压值，不同flash阈值电压范围也是不同的，不要认为所有flash的阈值电压都是那样。

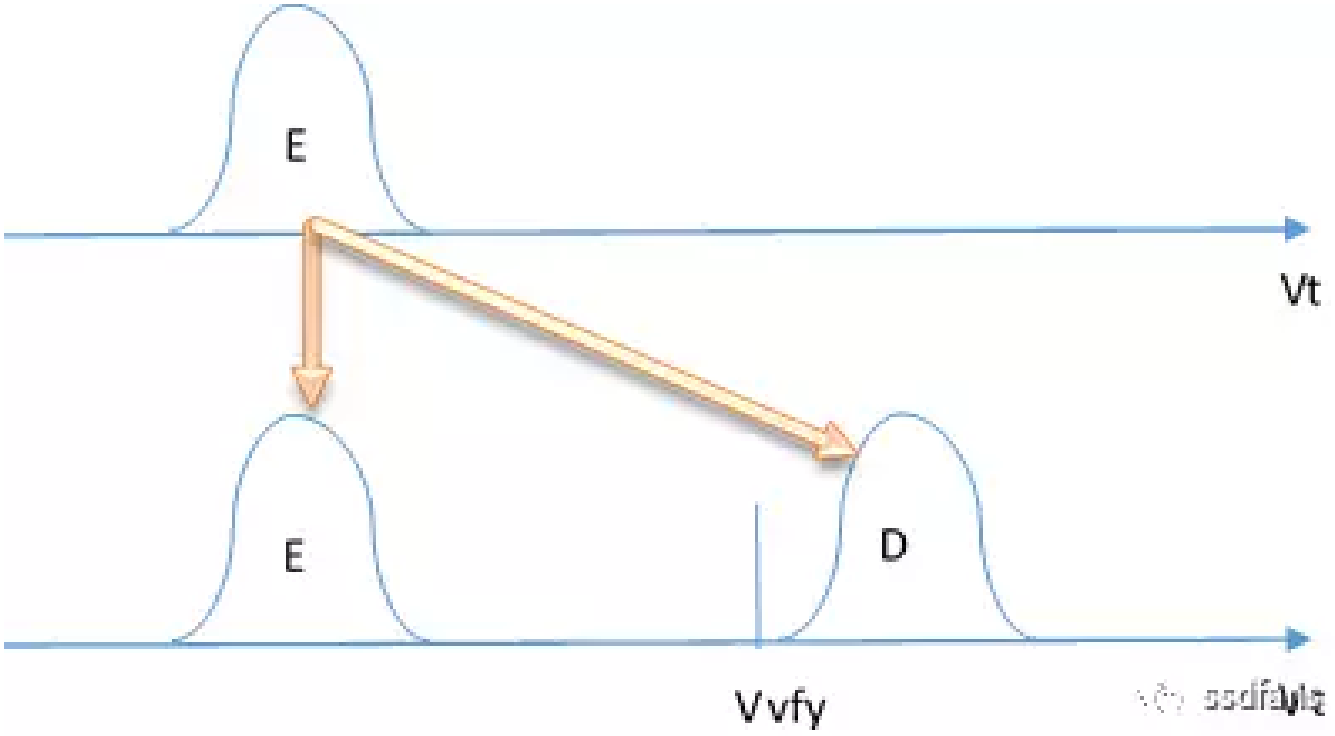
对FGF（Floating Gate Flash）技术的，MLC programming一般分两步走：先program Lower page，然后program Upper Page。为什么不一步到位？后面再说。

Program Page之前，整个block得擦除，因此，开始Page处于Erase（E）状态。

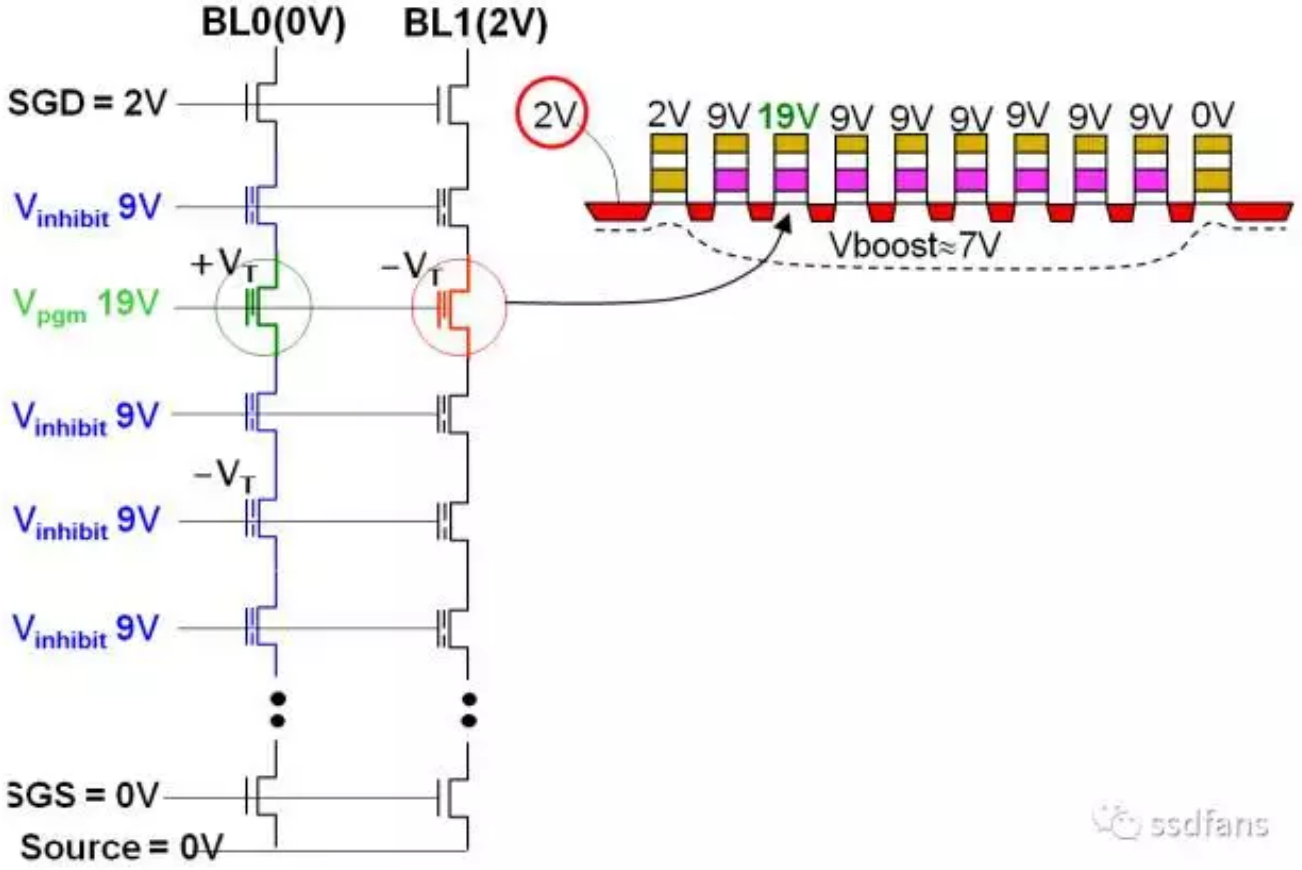
有人要问，为什么block擦除后（电子从浮栅极拽出来），所有Cell的阈值电压不是相同值（如下图），而是一个小区间（如上图所示）。

这是因为，并不是每个cell的体质都一样的，当擦除时，有些cell存储的电子全部清空，还有cell浮栅极内的电子或多或少还存在一些，因此每个cell的阈值电压落在一定范围内。只要这个范围在可接受范围内，那是允许的。

Program lower page:



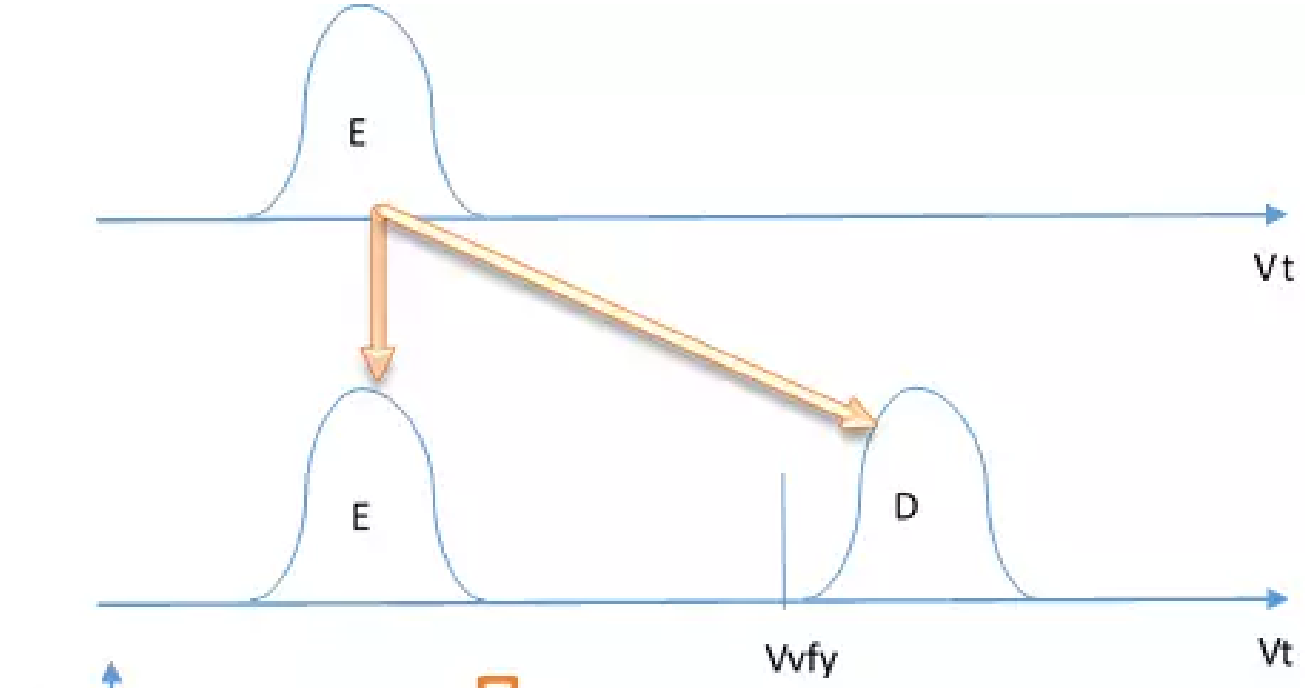
对某个Cell来说，当写入1时，无需Programming，阈值电压保持不变，还是处于E状态；当写入0时，必须进行Programming，使它的阈值电压落到D范围 ，即E→D。

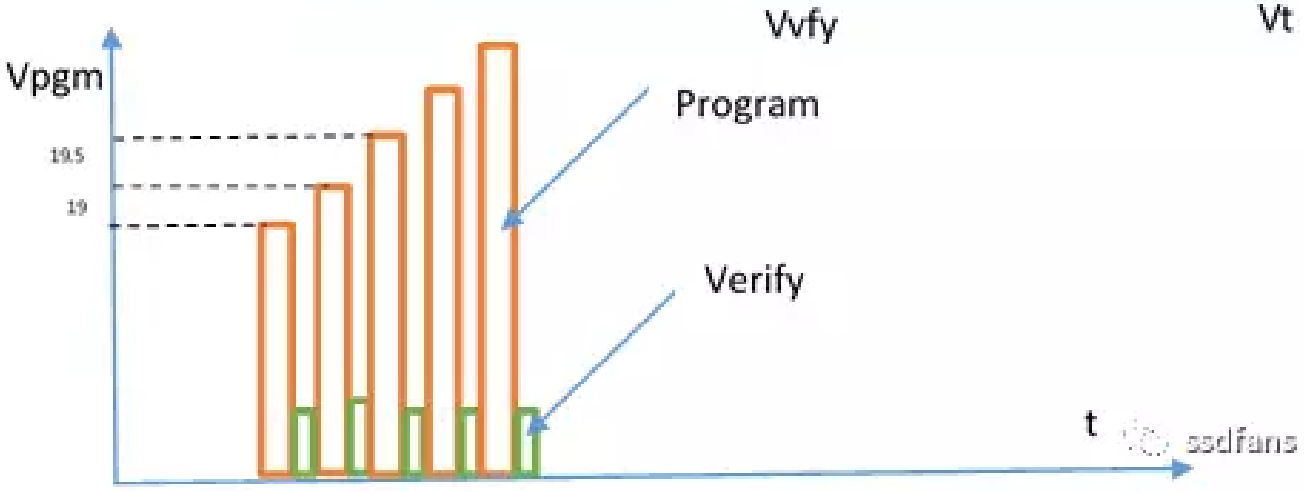


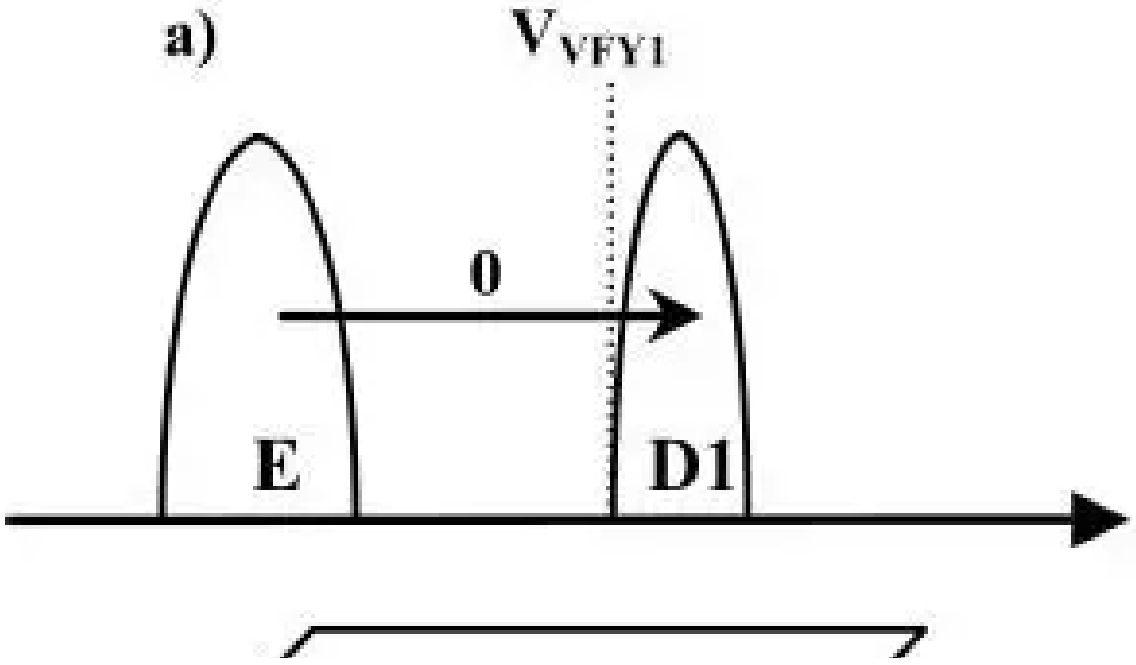
Programming某个page时，往其所在的WL控制极上施加高电平，衬底接地，对需要Programming的存储单元（如上图绿色圈起来的cell）：1→0，其bitline接地；对不需要Programming的存储单元（如上图红色圈起来的cell）：1→1，bitline拉高，就能抑制其Programming。

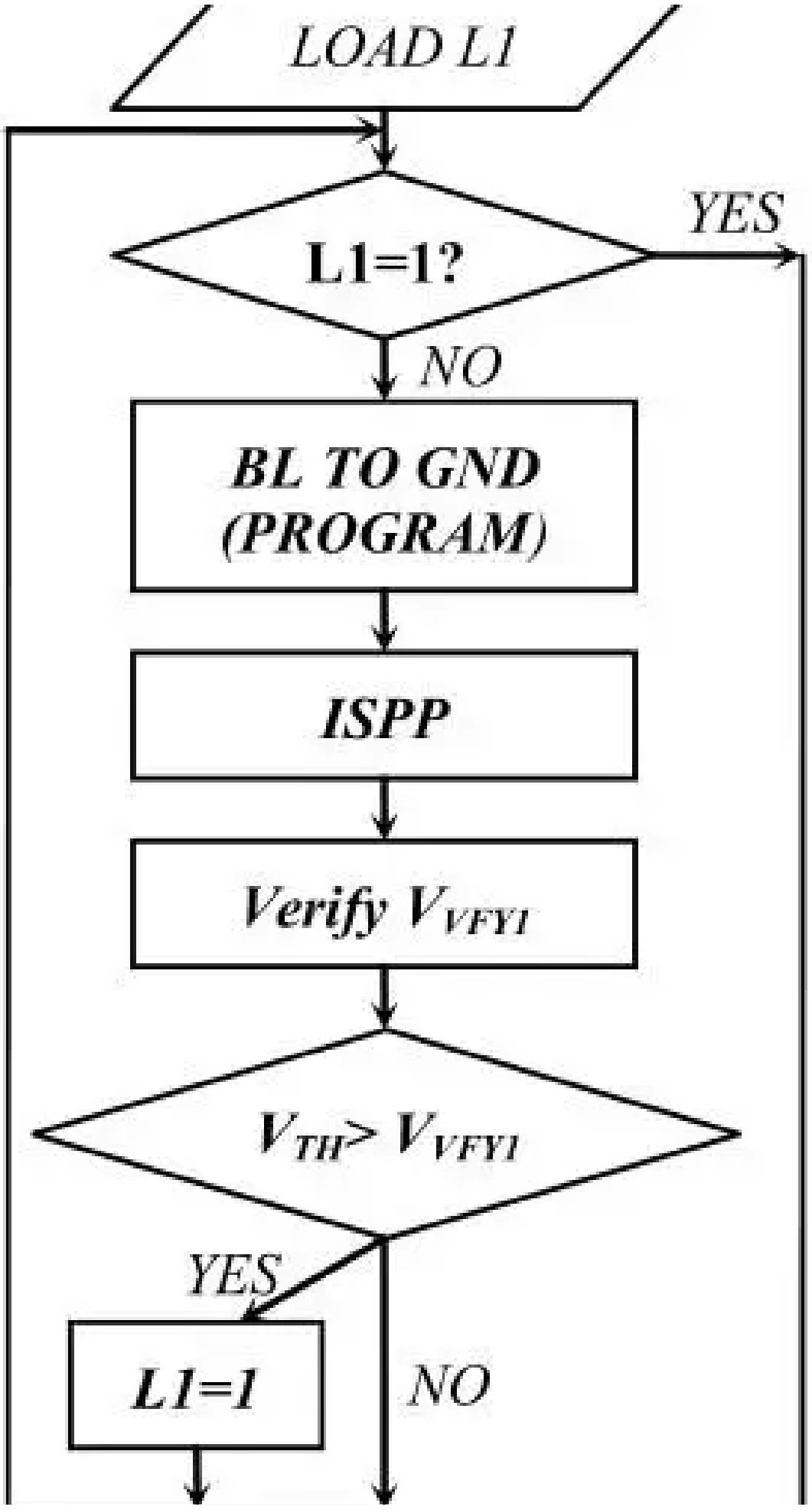
由于每个存储单元的体质不一样，有些cell很快就能让其阈值电压跳到D范围，而有些存储单元需要若干次Programming，才能使其到达D。flash采用Program + Verify的方式进行Programming，即开始在WL控制极上施加一个电压，如上图所示，假设Vpgm =19V，让这个电压持续一段时间，然后撤掉，对每个Programming的存储单元进行验证，看其阈值电压是否超过验证电

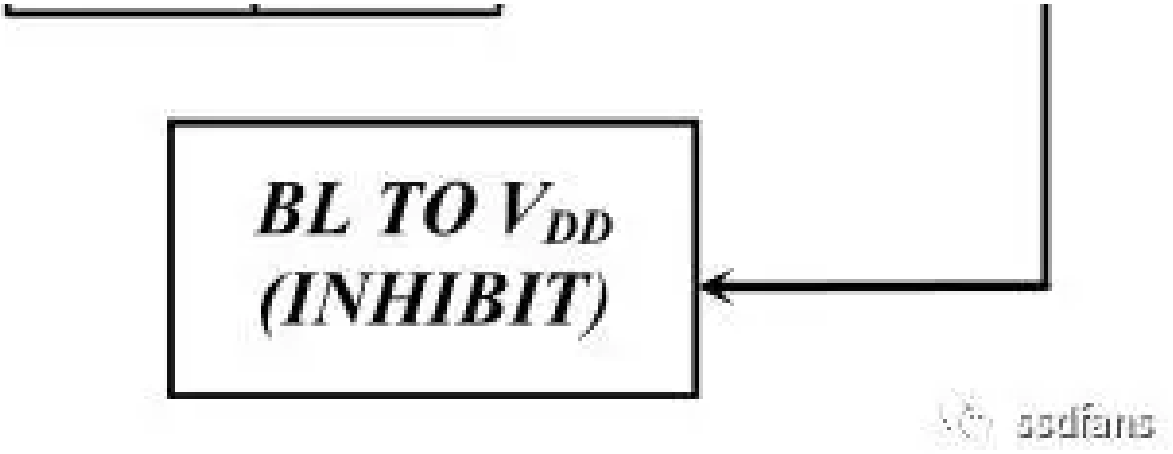
压Vvfy，如果超过，说明达到D状态，这些存储单元的bitline从接地到拉高，让其不再继续Programming，而对那些没有达到D状态的存储单元，还需要继续Programming。Vpgm会加大，比如增加0.5V，Vpgm = 19.5V，继续Programming，verify，直到达到D状态。


E
Vt
E
D
Vvfy
Vt


Vpgm
19.5
19
Program
Verify
t
ssdfans


a)
V_{VFY1}
0
E
D1


LOAD L1
L1=1?
YES
NO
BL TO GND
(PROGRAM)
ISPP
Verify V_{VFY1}
V_{TH}> V_{VFY1}
YES
L1=1
NO



该流程图（来自PAGE285，INSIDE NAND FLASH MEMORIES ）描述了Lower Page Programming的过程。我前面已经描述了这个过程，不再赘述。ISPP: Incremental Step Pulse Programming, 就是递进增加电压进行Programming。初始电压Vpgm和步进值都是闪存预先设置好的。

好比一个班级有很多同学（每个同学对应一个存储单元），老师讲一个知识点，有些学生接受能力强，老师讲一遍就明白，有些学生能力差点，可能需要讲好几遍才能明白。

老师怎么判断学生到底掌握了没有，他通过给学生出测试题，测试通过的学生可以休息不听，没有通过的话，老师更加卖力的讲解：又是画图，又是举例，讲完后，再给这些学生做测试题，有些学生在老师卖力的讲解下，通过了测试，还有一些学生还是没有理解知识点，没有通过测试。哎，没有办法，老师为了让大家都掌握好，再变着花样讲，直到全班学生都把问题弄明白。

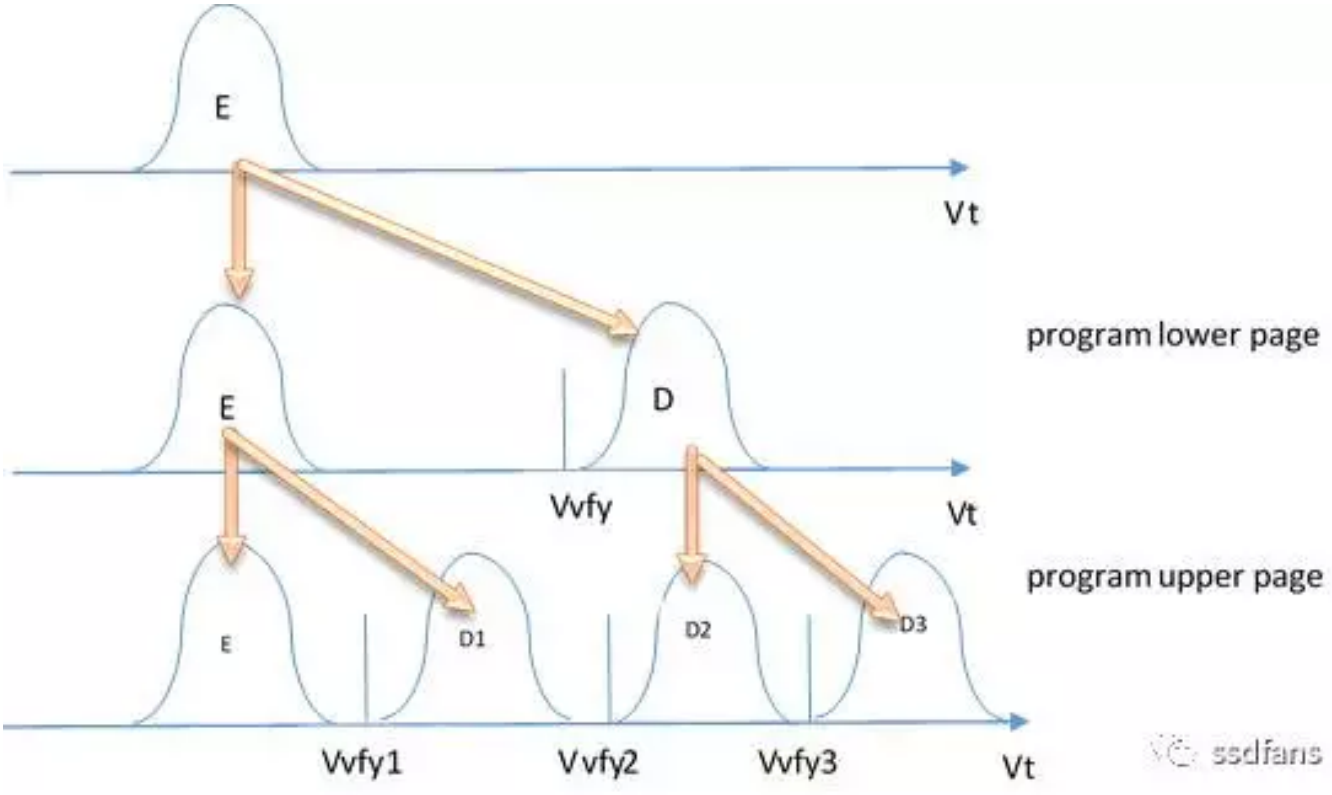
如果在一节课时间内，班上还是有同学没有通过测试，如果这个比例很小，比如只有两三个学生，这节课还是算成功的；如果失败的学生超过一定的数量，那么这节课可能就没有达到目的了，算失败了。

类似的，如果一个Page在经过若干次Programming后，还是有很多cell无法达到指定的状态（一些cell体质太差），如从E→D，那么这就是我们常说的Program failure。SSD的一般处理，检测到Program failure，把该block标为坏块，不再使用。

Programming upper page :

事情变得稍微复杂一些，Programming lower page之前，存储单元只处于擦除状态，即E状态；经历过 lower page Programming后，存储单元可能处于E状态，也可能处于D状态。

如果继续Programming upper page，cell要达到的状态，不仅取决于要写入的数据，而且要基于当前cell的状态。

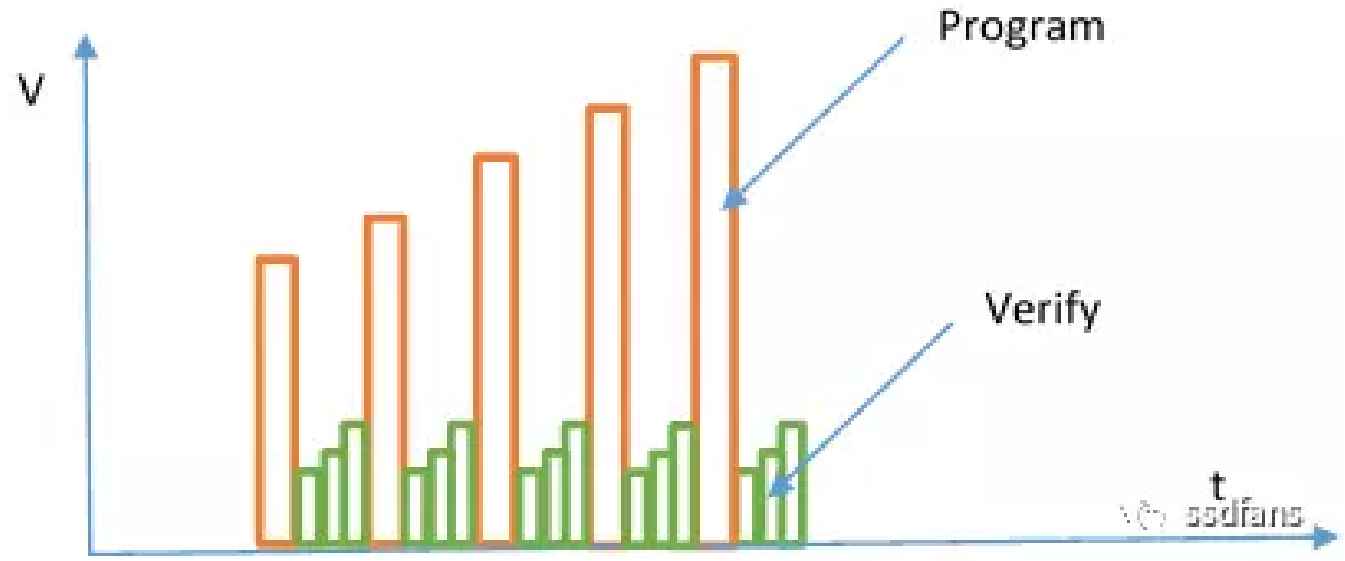


在Programming upper page之前，flash需要读取该Page，获取page中所有cell的当前状态，然后根据输入，Programming cell到相应的目标状态。

举个例子：

如果当前cell状态处于E，如果写入1，无需Programming，E状态保持不变;如果写入0，则需要Programming该cell到状态D1；

如果当前cell状态处于D，如果写入1，program该cell到D2;如果写入0，则需要Programming该cell到状态D3；



同样的，Programming Upper Page的过程还是program+verify，不过，每次verify的电平由1个（Vvfy）变成三个（Vvfy1，Vvfy2，Vvfy3）。具体说来，Vpgm电压加在WL控制极一段时间后：

首先用参考电平Vvfy1去读，如果之前cell状态为E且输入为0，导通则说明该cell还没有program到D1状态，还需继续program；如果截止，说明其阈值电压大于Vvfy1，已经从E状态跳到D1状态，则这

些cell无需再programming，bitline拉高抑制其继续programming；

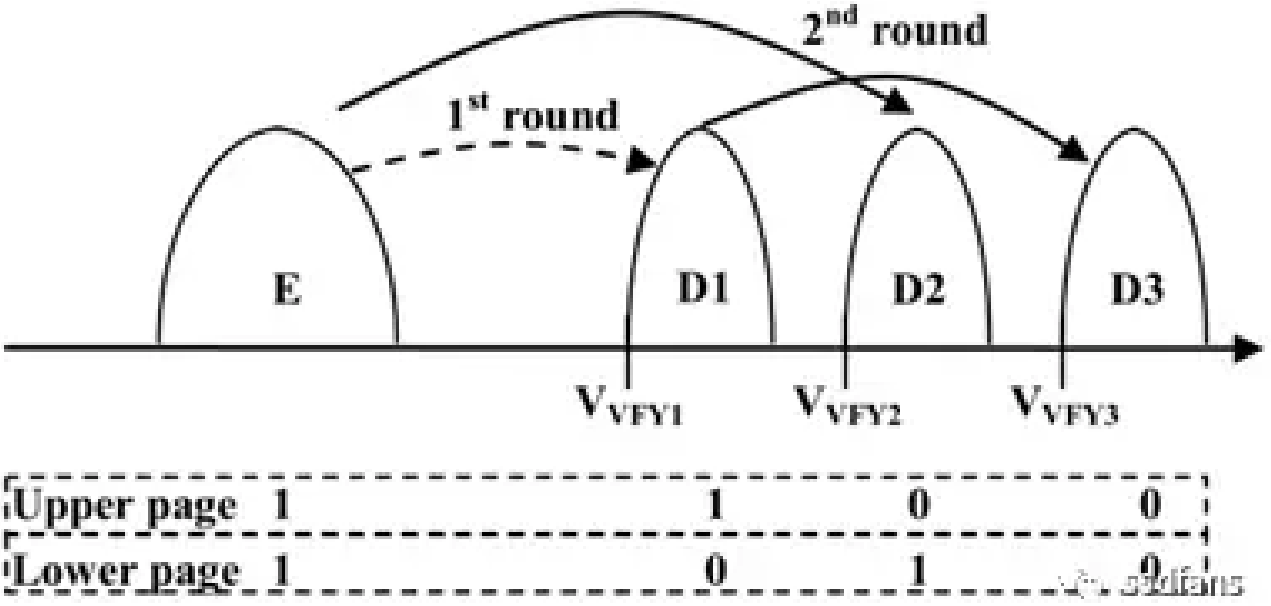
然后用参考电平Vvfy2去读，如果之前cell状态为D且输入为1，导通则说明该cell还没有program到D2状态，还需继续program；如果截止，说明其阈值电压大于Vvfy2，已经从D状态跳到D2状态，则这些cell无需再programming，bitline拉高抑制其继续programming；

最后用参考电平Vvfy3去读，如果之前cell状态为D且输入为0，导通则说明该cell还没有program到D3状态，还需继续program；如果截止，说明其阈值电压大于Vvfy3，已经从D状态跳到D3状态，则这

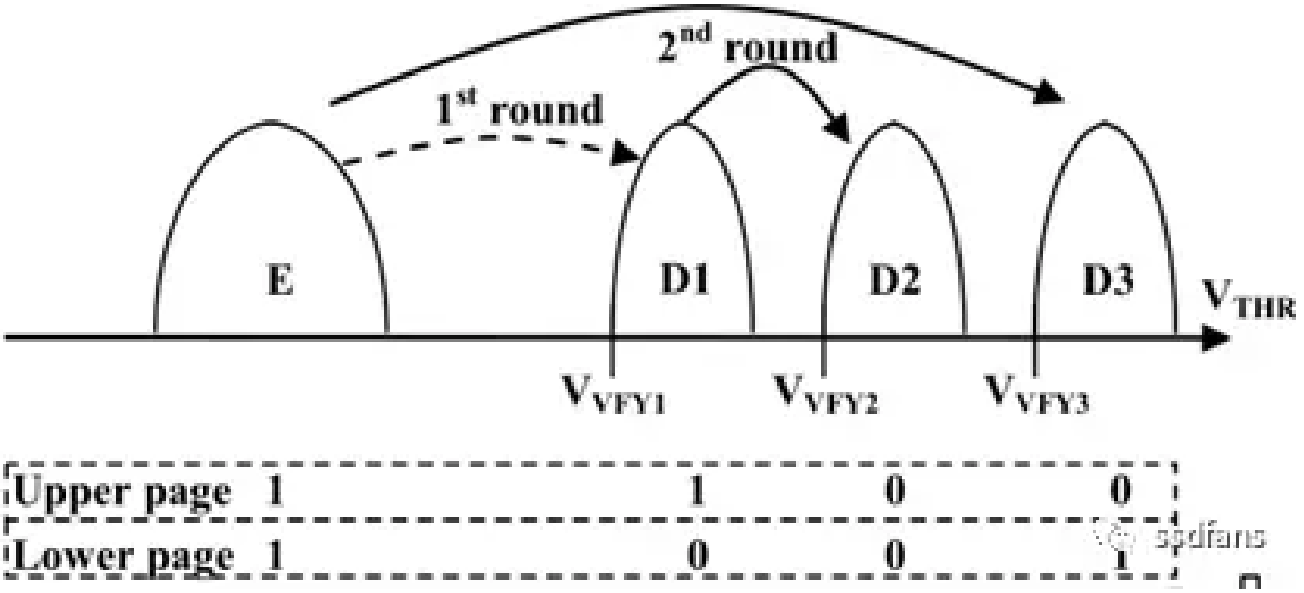
些cell无需再programming，bitline拉高抑制其继续programming。

如果不是所有的cell都program到目标状态，则增加Vpgm，然后重复前述Program+verify过程，直到program成功。

需要说明的是，不同厂家的flash的状态编码和跳转不一定都是这样，比如有这样的：



还有这样的：



但page programming的过程都是相同的，即Program+verify。本文旨在描述这个过程。

比较一下lower page programming和upper page programming，有以下几个不同：

- Program upper page时，需先知道当前cell的状态，在programming之前需要读所有

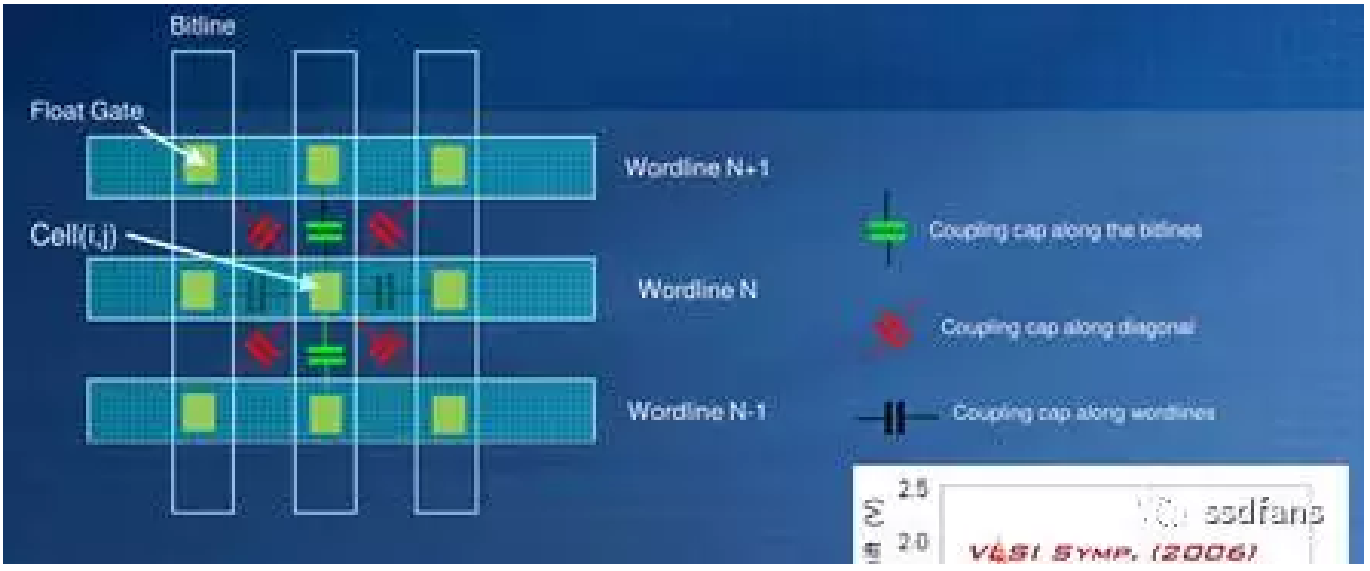
的cells； Program lower page时候无需此操作，因为所有cell都处于擦除状态；

- Program upper page时，需要verify好几个电平，而program lower page时，只需verify一个电平；
- Program lower page时，E状态和D状态可以隔得比较远，因此program时候可以加更大的初始program电压和更大的步进（step）电压；而Program Upper page时，E，D1，D2，D3之间的间隔比较小，因此program需要更精细的控制电压，不然一下program过头了。比如本来期望从E状态跳跃到D1状态，不小心电压加大了，直接就跳到D2，那就麻烦了。

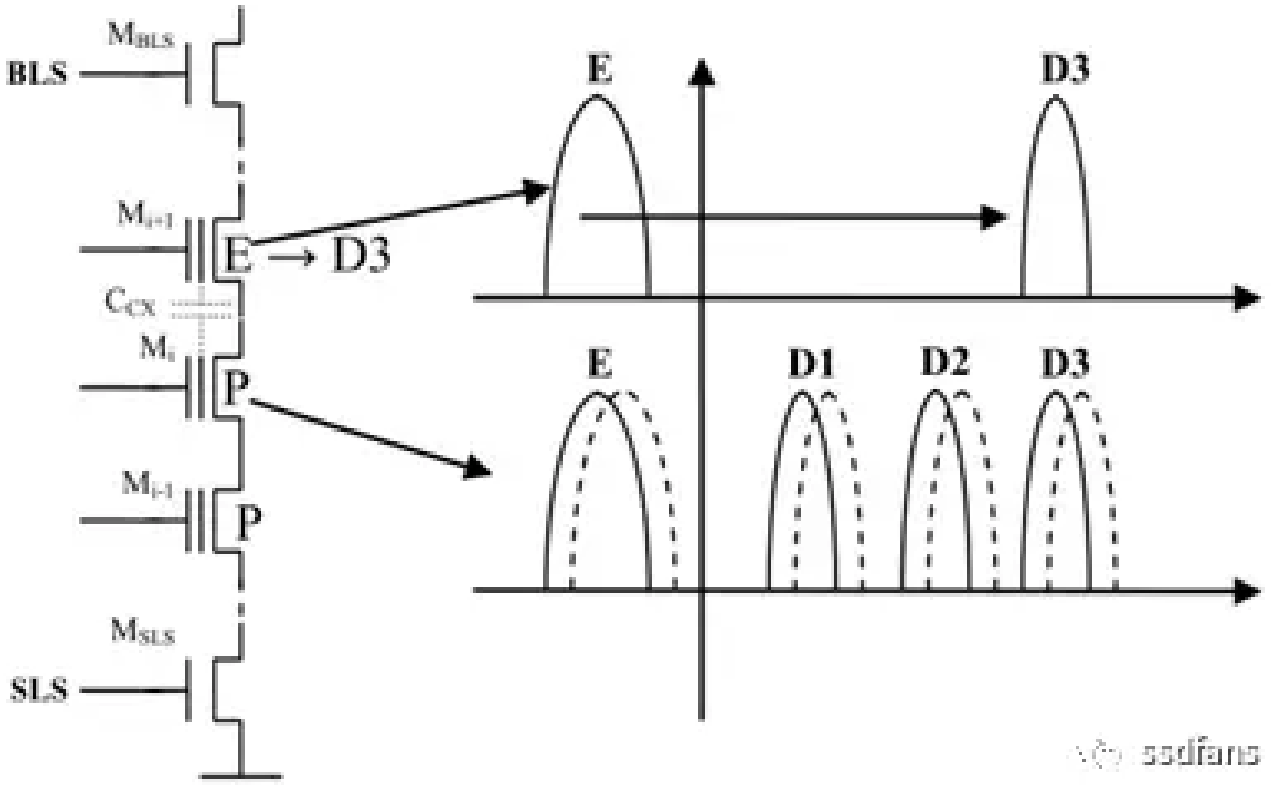
大家可能都知道Lower page programming比Upper page

programming更快，之前可能知其然不知其所以然，现在看了flash内部programming的过程，应该明白了吧。

回过头来说说为什么FGF flash需要分两步走来programming，而不是一步到位。FGF存储电荷的浮栅极是导体，每个cell和每个cell之间构成一个电容，一个cell电荷的变化会影响其它cell的状态。一个cell受影响和影响最大的是离它最近的9个cell。

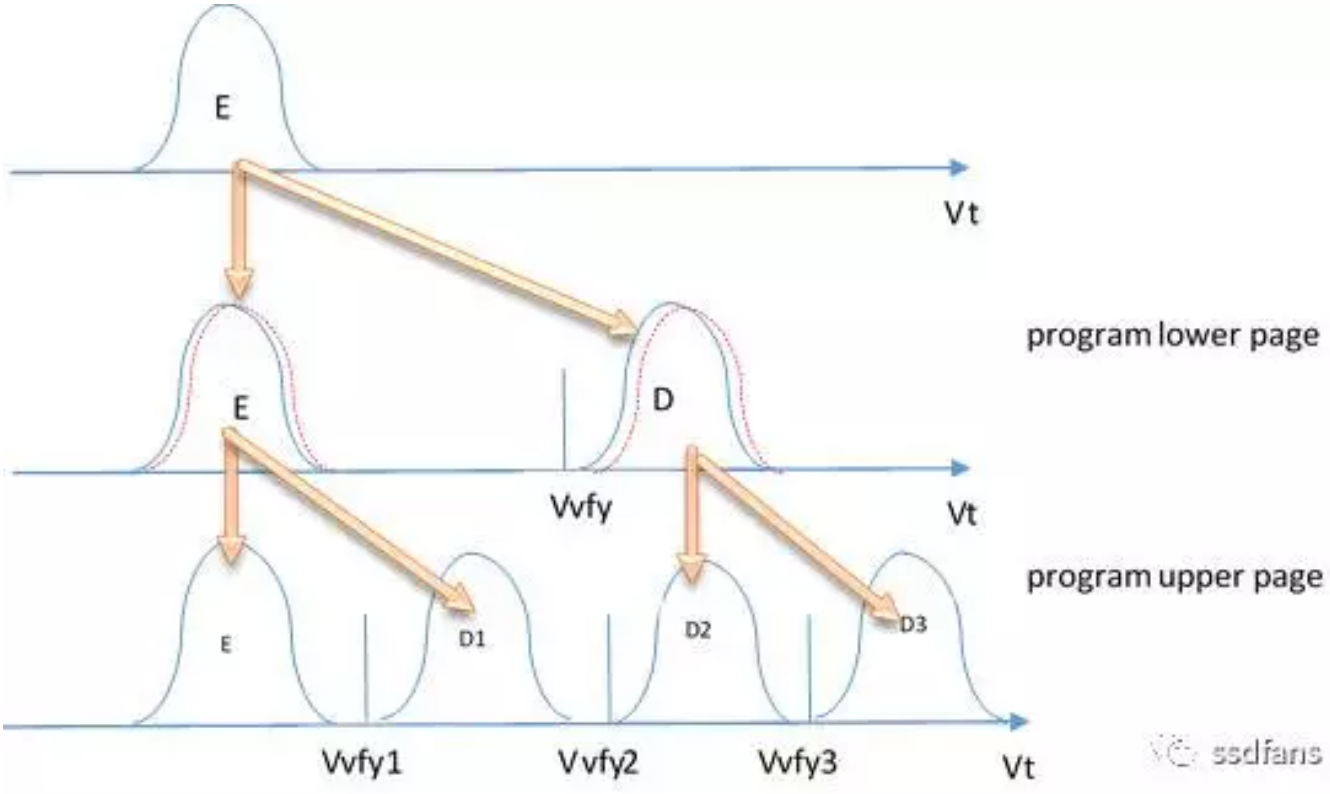


如果一次就把lower page和upper page都programming好了，那么当programming下一个WL时，会影响之前programming过的cell状态：



如上图，WLi program好了，然后program WLi+1，会导致整个WLi状态集体右移，当去读整个WL时，可能就数据错了。

分两步走就能克服（或者缓解）这个问题。首先WLi的lower page被program，cell处于D（中间状态，不需要高的精度）或E两种状态，两者间隔比较开，然后program下个WLi+1，WLi受到影响，D和E向右移。

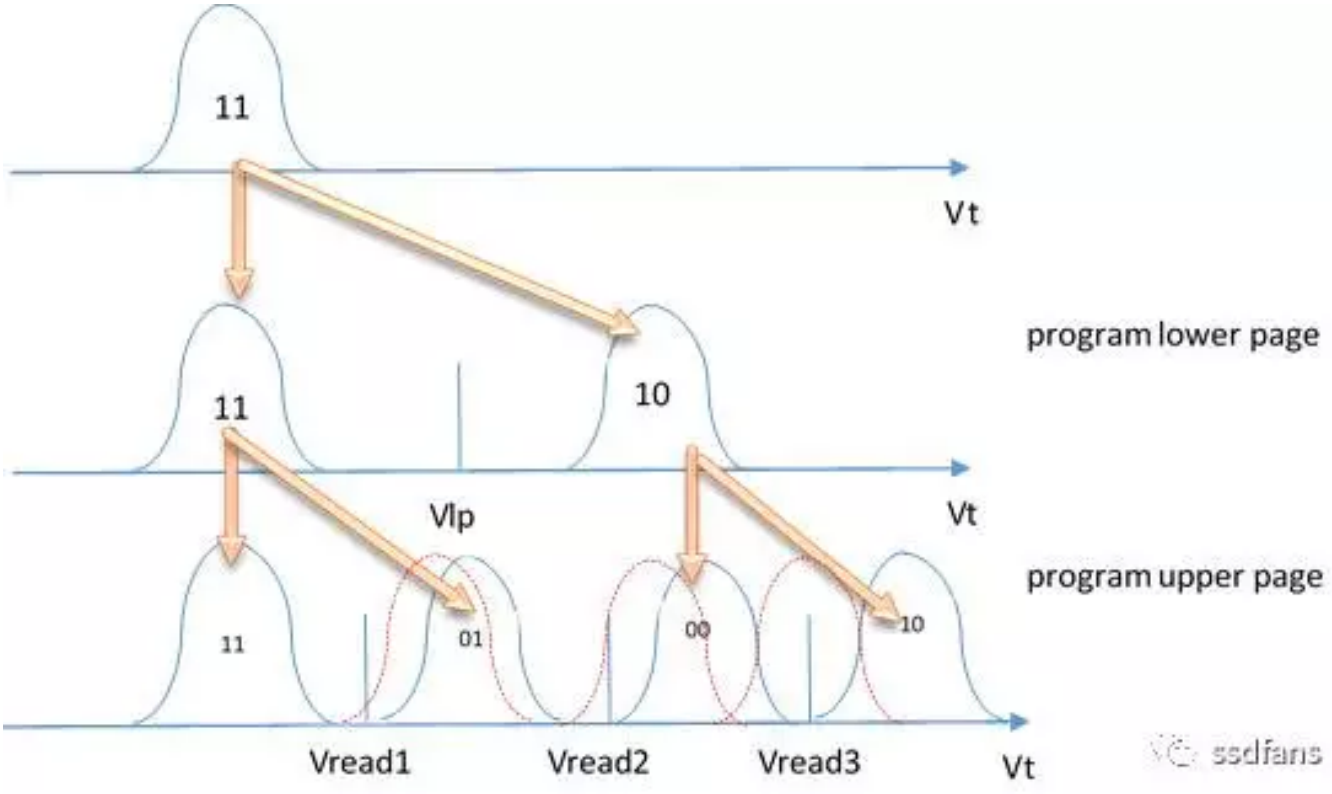


当Programming upper page时，即使E和D向右移动，只要目标状态明确，还是可以通过Program+verify把cell Program到目标状态。

当Programming完WLi的upper page时，然后再programming WLi+1的upper page，这个时候WLi+1对WLi的影响还是存在的，但是已经没有之前大了。

两步走可以减少cell和cell之间耦合造成的影响，因此FGF（Floating Gate Flash）一般lower page和upper page分开写。对CTF（Charge Trap Flash），由于存储电荷的单元是绝缘体，cell与cell之间耦合影响天然消除，因此没有必要分两次进行programming。

分两次写最大的问题是，如果在programming upper page时发生异常掉电，那么之前Lower page的数据也丢失。我们看看为什么会这样。



如上图，在programming upper page的时候，发生了异常掉电，flash中cell都没有program到目标状态，如红色虚线所示。

正常情况，如果upper page没有被program，那么读lower page数据时候，只需在该WL控制极加Vlp电压，导通的cell读出的数据为1，不导通的cell读出的数据为0；如果upper page和lower page都被program了，那么读lower page数据时，在该WL控制极加Vread2，就能读出lower page数据：导通的cell读出的数据为1，不导通的cell读出的数据为0。

异常掉电发生后，如果还是用Vread2去读Lower Page数据，就有一块数据（黑色粗箭头所指）会发生误判：本应为0的数据被判读成1，如果误判的数据超出ECC能纠的比特，那么lower page数据读就失败，意味着之前写入的数据丢失。

结束前，总结一下flash programming的那些事：

1. Vt distribution 是什么？
2. Page 如何program？ 什么是program + verify？
3. 什么是program failure？
4. 为什么lower page比upper page写的快？
5. 为什么要分两步写？
6. 为什么upper page programming失败会导致lower page数据丢失？

就到这吧，希望大家看完有收获。

喜欢就请分享转发！

怎么阅读ssdfans其他文章？点击文末**阅读原文**进入www.ssdfans.com，用搜索框搜索关键字即可。

不想错过后续精彩文章？长按或扫描下面二维码关注ssdfans就可以了！

ssdfans微信群介绍

技术讨论群	覆盖2000多位中国和世界华人圈SSD以及存储技术精英
固件、软件、	固件、软件和测试技术讨论

测试群	
异构计算群	讨论人工智能和GPU、FPGA、CPU异构计算
ASIC-FPGA群	芯片和FPGA硬件技术讨论群
闪存器件群	NAND、3D XPoint等固态存储介质技术讨论
企业级	企业级SSD、企业级存储
销售群	**全国SSD供应商都在这里，砍砍价，会比某东便宜20%！**
工作求职群	存储行业换工作，发招聘，要关注各大公司招聘信息，赶快来
高管群	**各大存储公司总监以上高管**

和主任研发工程师

想加入这些群，请加nanoarch为微信好友，介绍你的**昵称-行业-职务**，注明群名，拉你进群。

Read more