NAND Flash Memory

&

LDPC Coding

Name:温嘉颖

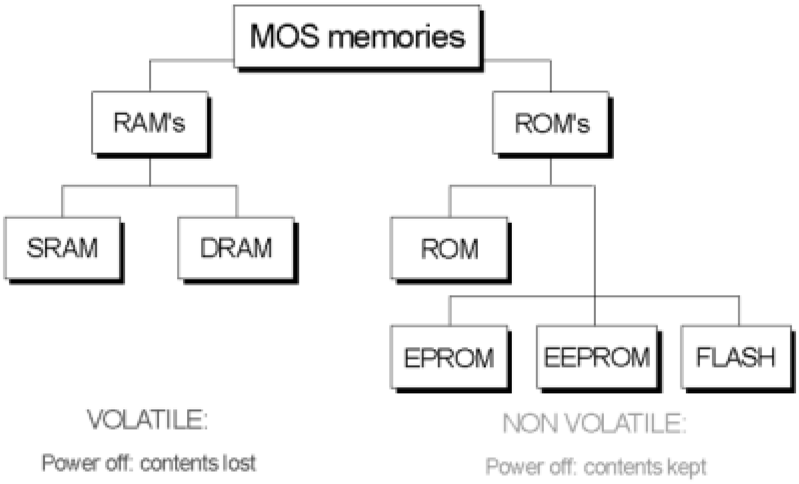
Tutor:韩国军

目录

第一章 非易失性存储技术

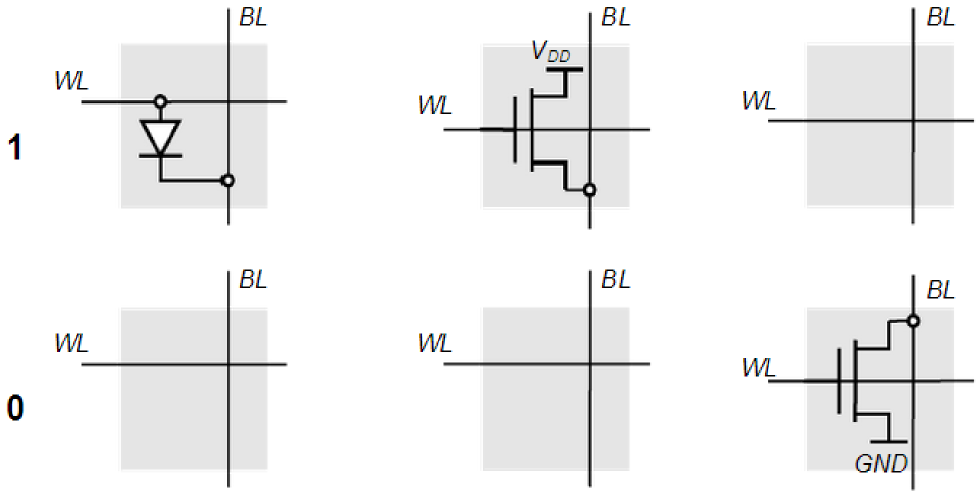
1.1 非易失性存储器

非易失性存储器(non-volatile memory , NVRAM)是指当电流关掉后，所存储的数据不会消失者的计算机存储器。非易失性存储器中，依存储器内的数据是否能在使用计算机时随时改写为标准，可分为二大类产品，即 ROM 和 Flash memory。



1.2 基本的 ROM 单元

只读存储器阵列可以看做是一种简单的组合布尔逻辑，即它对每个输入组合（地址）都会产生一个指定的输出值。因此，在一个特定地址存储二进制信息，可以通过被选行（字线）与被选列（位线）间有无数据路径（相当于特定位置上有无元件或元件是否在标准电压下导通）来实现。而实现数据路径的基本结构有两种，即 NOR 和 NAND 阵列。



(a)Diode ROM　　(b)MOS ROM 1　　(c)MOS ROM 2

首先，考虑最简单的单元，如图(a)所示，这是一个基本的 ROM 单元。假设位线 BL 通过一个电阻接地，没有任何其他的激励或输入。这就是 0 单元中的情况。由于字线 WL 和位线 BL 之间不存在任何实际的连接，所以 BL 的值为低电平而 WL 得值无关。反之，当把

一个高电压 WLV 加在 1 单元的字线上时二极管导通，字线被上拉至 $V_{WL}-V_{D(on)}$，结果在位线上形成了一个 1。总之，在 WL 和 BL 之间是否存在一个二极管区分了 ROM 单元中存放的是 1 还是 0。

1.2.1 NOR ROM 结构

然而，由于二极管单元的位线与字线是不隔离的，所有需要用来充电位线电容的电流，必须通过字线和它的驱动器来提供，而这些电流这大容量存储器中是非常大的，因此，这一方法只适用于小存储器。一个改善隔离的方法是在单元中使用一个有源器件，如图(b)所示，其工作原理与二极管单元相同，但是它的所有输出驱动电流都是由单元中的 MOS 管提供的，字线驱动器只负责充电和放电字线电容。但是，这一改进的直接代价是单元比较复杂和面积较大（额外的电源接触孔所致）。下面是使用这一个单元的 MOS ROM 阵列。

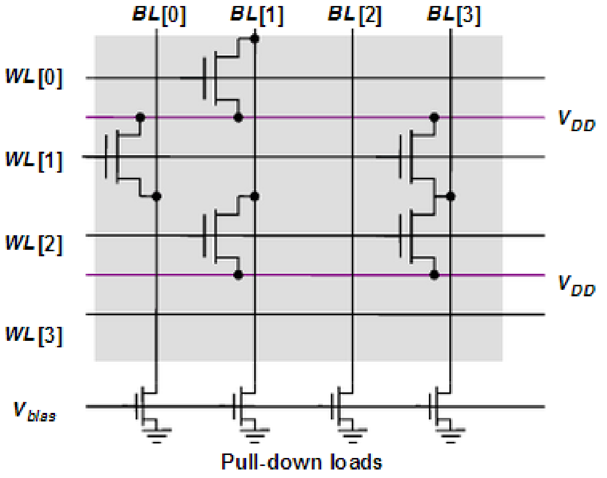


图2.2　一个4×4的OR ROM 单元阵列，使用图 2.1(b)单元；

或者使用(c)类型的 MOS RON 阵列。

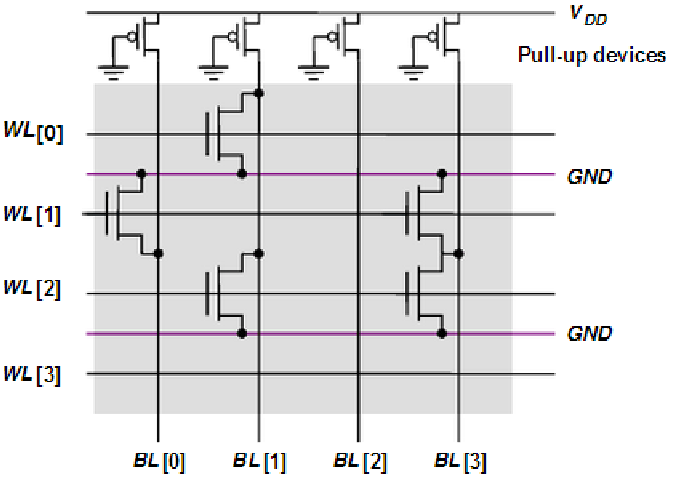


图 2.3　一个 4×4 的 NOR ROM 单元阵列，使用图 2.1(c)单元；

图 2.3 中 4×4 NOR ROM 阵列的两种可能的版图如下：

1.2.2 NAND ROM 结构

NOR ROM 的两种结构类型中，晶体管只占据了整个单元尺寸的很小比例，单元的大部分面积用于位线接触和接地连接。避免这一开销的一种方式是采用不同的存储结构，即 NAND ROM 结构

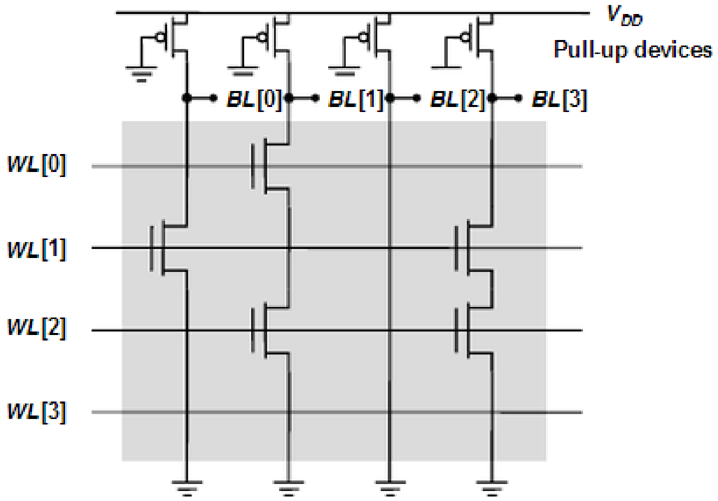


图 2.5 4×4 MOS NAND ROM

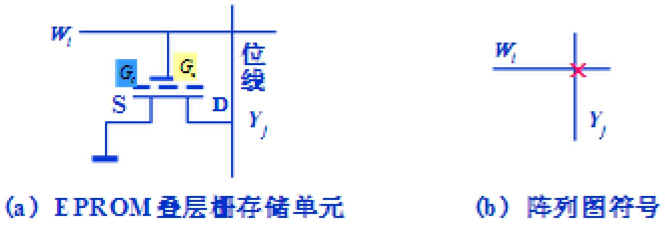
正常操作中，被选中的字线被下来为逻辑低电平，未被选中的所有字线保持为高电平。如果一个晶体管位于被选中的行与列的交点上，则此晶体管截止，且列电压被负载元件拉到高电平。另一方面，在多输入的 NAND 结构中，如果在此特定交点上无晶体管（短路），那么列电压会被其他的 nMOS 晶体管拉到低电平。因此，在交点上，无晶体管则表示存储‘0’，交点处有晶体管则存储‘1’。

1.3 可编程只读存储器(PROM)

PROM (Programmable Read-Only Memory)可编程只读存储器,也叫 One-Time Programmable (OTP)ROM“一次可编程只读存储器”，是一种可以用程序操作的只读内存。最主要特征是只允许数据写入一次，如果数据烧入错误只能报废。

1.4 可擦除可编程只读存储器(EPROM)

EPROM(*Erasable programmable read only memory*)是通过封装在一个透明窗口把紫外线(UV)照射到单元上来进行擦除的。PROM 的存储单元采用叠栅注入 MOS 管（SIMOS 管)。单元结构简单，密度高，成本低。但是，采用 UV 擦除的两个主要缺点是擦除过程很慢和可靠性问题。

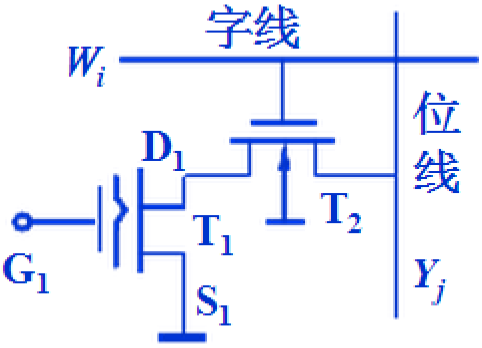


（a）EPROM 叠层栅存储单元　　（b）阵列图符号

1.5 电擦除可编程只读存储器(EEPROM)

EEPROM(*Electrically erasable programmable read only memory*)采用了一种称为 FLOTOX (floating-gate tunneling oxide)晶体管浮栅器件作为可支持电擦除过程的可编

程器件，如图 3.5 所示。它与 FAMOS 器件类似，但隔离浮栅与沟道和漏端的那一小部分绝缘介质的厚度减少到大约 10nm 或更少。当把一个约 10V 的电压加到这一很薄的绝缘层时，电子通过隧穿机理穿入或穿出浮栅。下图为隧穿节的 I-V 特性曲线。 这一编程的方法的主要优点在于它的可逆性，即只要在写过程中所加的电压反过来即可实现擦除。向浮栅注入电子将使阈值V_T升高，而相反的操作则降低V_T。但是这一双向工作带来了阈值控制的问题：V_T可能低于 0V，从而有效地产生一个即使栅上施加 0V 电压也不会关闭的器件。使用选择器件的原因是阈值电压很难精确地控制，因为这取决于生产过程中的一些变化和浮栅上初始存储的电荷。如果由于初始储存的电荷使得不可能可靠地达到希望的内部电压，那么存储器单元将不能正确工作。为了避免这个问题，与 FLOTOX 晶体管串联一个选择管，并连接到字线和位线。选择管使用正常的字线电平，而 FLOTOX 晶体管有一个位于两个可能阈值之间的合适的栅电压，如下图所示。

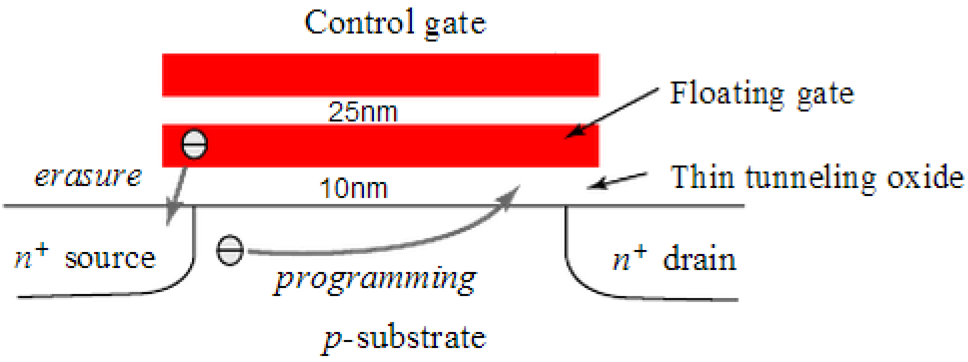


1.6 Flash Memory

Flash EEPROM 的概念在 1984 年由 Masuoka 等提出，并很快发展成为应用最普遍的非易性存储器结构。Flash EEPROM 是 EPROM 和 EEPROM 方法的结合，大多数 Flash EEPROM 器件采用雪崩热电子注入的方法来编程器件；擦除则和 EEPROM 单元一样，采用 FN 隧穿(Fowler-Nordheim)来完成的。它集中了两者的优点，既具有像 EPROM 一样的单管结构，又沿用了传统 EPROM 热电子隧道效应的编写机制，并具有 EEPROM 在线、冷电子隧道效应的擦除机制。基本存储单元尺寸比 EEPROM 小 10 倍左右；但是，Flash EEPROM 的擦除是对整个芯片或存储器的子部分成批进行的。它是目前唯一具有大存储容量、非易失性、低价格、可在线改写和较高速度等特性的存储器。

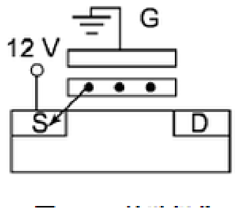
1.6.1 Flash 存储器的基本存储单元

下图是 Intel 推出的 ETOX(EPROM Tunnel Oxide) Flash 单元，这只是现在各种 Flash 单元中的一个。它与 FAMOS 门相似，但是采用了一个非常薄的隧道栅氧化层（10nm）。用栅氧的不同区域来进行编程和擦除。



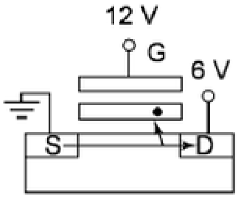
1.6.2 工作原理

A) 擦除操作



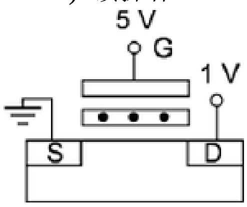
在栅上加 0V 电压，源端加上一个高电压(12V)。浮栅和源极间发生 FN 隧穿效应。浮栅上的电子被拉到源区，开启阈值电压变低(2V 以下)。

B) 写操作



把一个高电压加在所选器件的栅上，如果这时漏端加上‘1’，就会产生热电子并将其注入到浮栅上，使阈值电压上升；如果不加‘1’，则浮栅上仍保持在原来没有电子的状态。

C) 读操作



选择一个单元，将其字线上升至 5V，使位线有条件地放电。我们假设在本文的所有存储器中，都使用了预充电技术，即在读操作之前，位线已进行了预充电至 V_{DD} 。

在擦除操作的过程中，单元初始阈值电压的不同以及氧化层厚度的不同都会引起擦除操作结束时阈值电压的不同。这一点可以从两方面来弥补：(1)在应用擦除脉冲之前，将阵列中的所有单元都编程，以使所有的阈值都从大致相同的值开始；(2)在此之后，加上一个可控制宽度的擦除脉冲。接着读整个阵列以检查这些单元是否已被擦除。如果尚未全部擦除，则再应用另一个擦除脉冲，接着又是一个读周期；如此循环，直到所有单元的阈值电压都低于所要求的电平。

另外，源极加高压擦除是利用浮栅与源区构成的小电容、分压大、场强高的原理实现的。小电容是又浮栅和源区侧向扩散区面积构成的。因侧向扩散区相对于浮栅的面积很小，所以电容也很小。之所以利用源极擦除是因为存储矩阵或部分存储矩阵单元的源极都是连接在一起的，这样可以实现整个芯片或分块快速擦除（如 NOR Flash 结构）。但是，如果存储矩阵或部分存储矩阵单元的源极不是连接在一起的，则高压不是加到源极，而是加到部分存

储单元或全部存储单元的公共衬底上（如 NAND Flash 结构）.

More specifically .

Figure 1(a) shows flash memory cell structure having control gate, floating gate, source,and drain, where the floating gate is insulated from the control gate and substrate. The cell is programmed by applying a high voltage to the control gate, which moves electrons from the substrate into the floating gate, while the cell is erased by applying a high voltage to the substrate, which removes electrons from the floating gate. Figures 1(b) and (c) illustrate the program/erase operations.

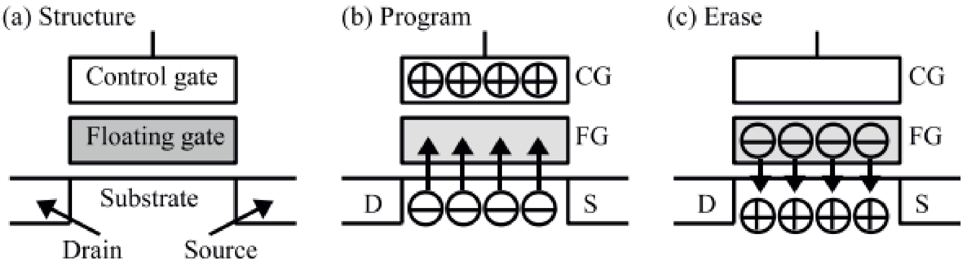


Figure 1. Flash memory cell.

In order to **read out data** from the cell, the charge level of the floating gate is identified by applying specific voltages to the control gate. That is, if the floating gate has few electrons, the drain-source current flows with a low control gate voltage, while if the floating gate has many electrons, the drain-source current flows only when the control gate voltage is sufficiently high.

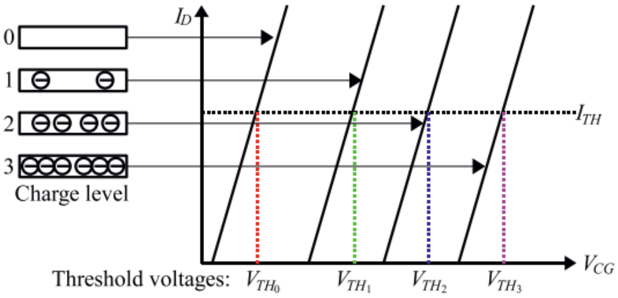
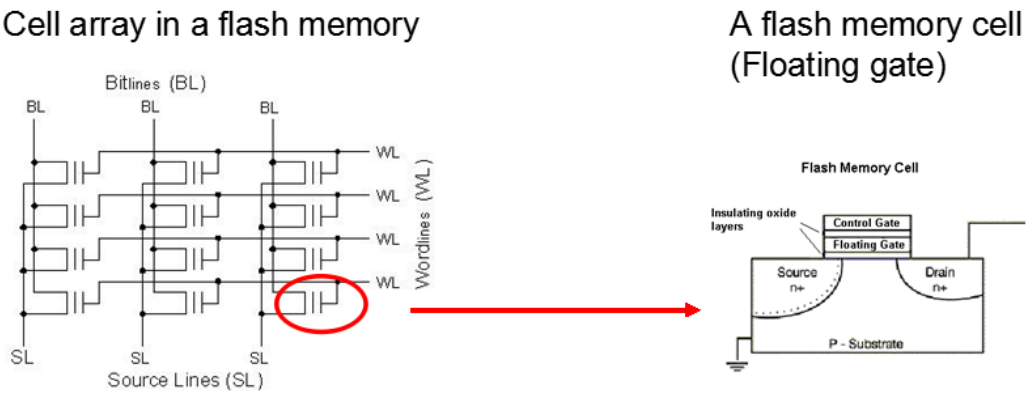


Figure 2. Schematic relation between control gate voltage V_{CG} and drain-source current I_D in 4-level cell memory.

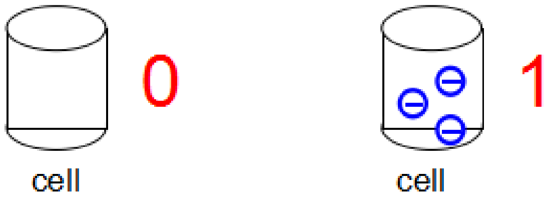
1.6.3 Flash 存储器存储矩阵结构

Flash 存储器存储矩阵结构有“或”阵列和“与”阵列两大类。前者存储单元并联，呈“或”关系，包括 OR 和 NOR 两种。后者存储单元是串联，呈“与”关系，包括 AND 和 NAND 两种。其中 NOR 和 NAND 两种结构最为常见。

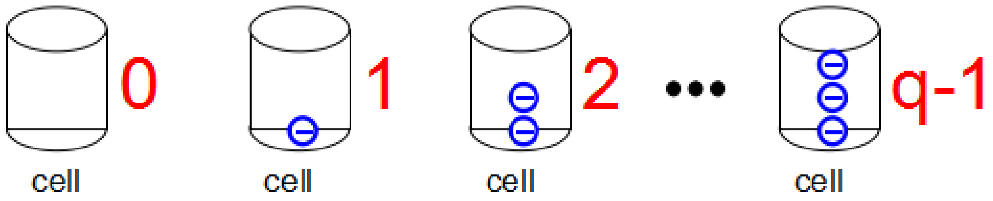
Flash Memory 的物理结构：A flash memory cell can store charge. And the charge level represents data.

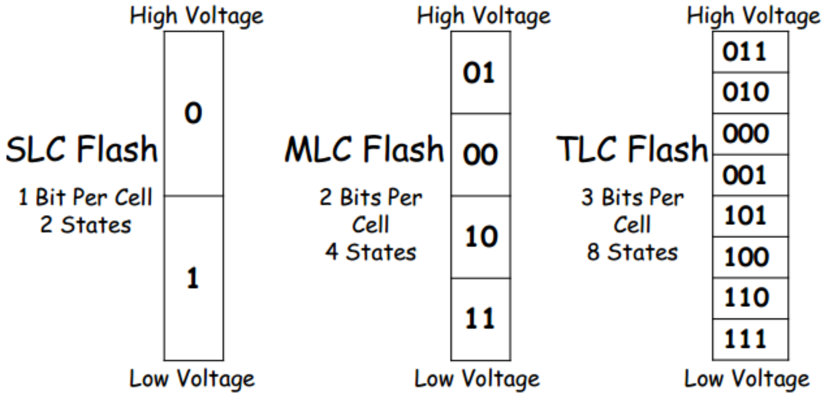


□ **Single-level cell:** Two levels → One bit



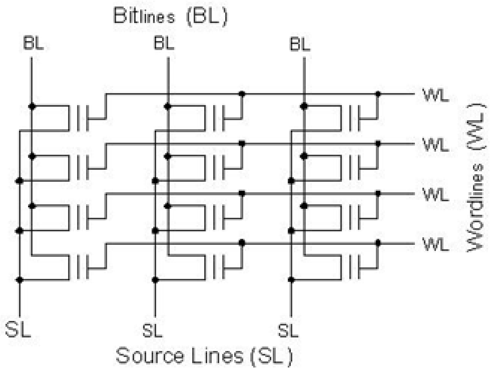
□ **Multi-level cell:** q levels → $\log_2 q$ bits

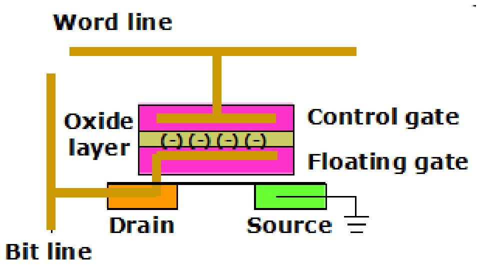




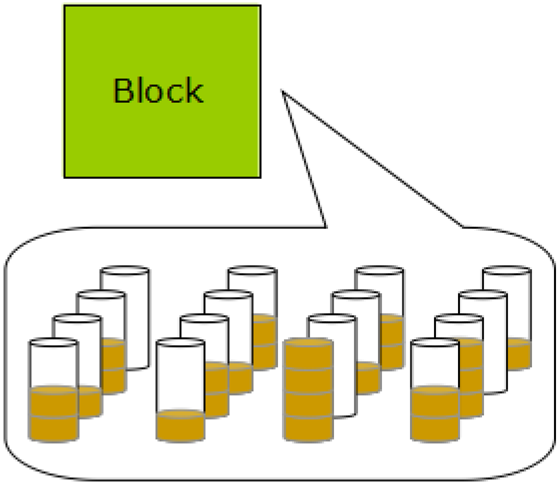
A) NOR Flash 存储单元（Older, still used）

Cells form blocks. A block, has about 100,000 cells. NOR is a random-access device. Every cell is directly addressable by the processor. That is, a cell can be individually read and programmed





In NOR, the level of a cell can be increased individually and multiple times. But to lower any cell level, the whole block must be erased at the same time. (Block erasure)

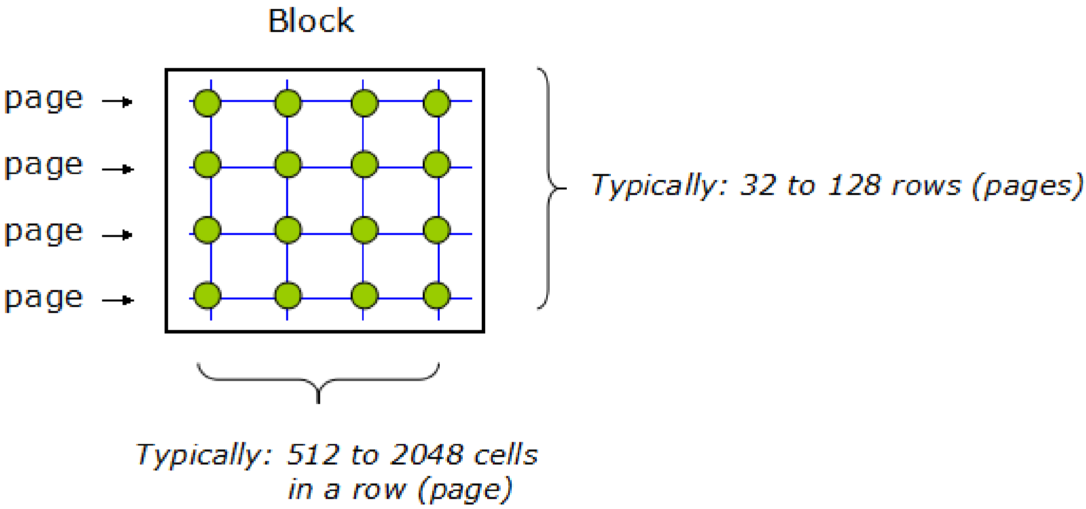


block of cells

B) NAND Flash 存储单元

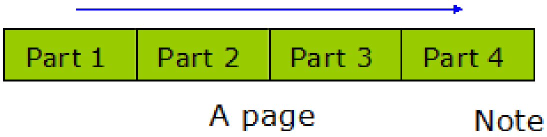
Cells form blocks. Every block is an array. Every row is a page.

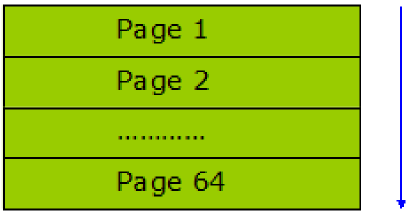
Read and write: A page as a unit. It also Block erasure.


Block
page
page
page
page
Typically: 32 to 128 rows (pages)
Typically: 512 to 2048 cells in a row (page)

- A page can be written only once before the block is erased.
  It is even recommended that the pages are written sequentially.

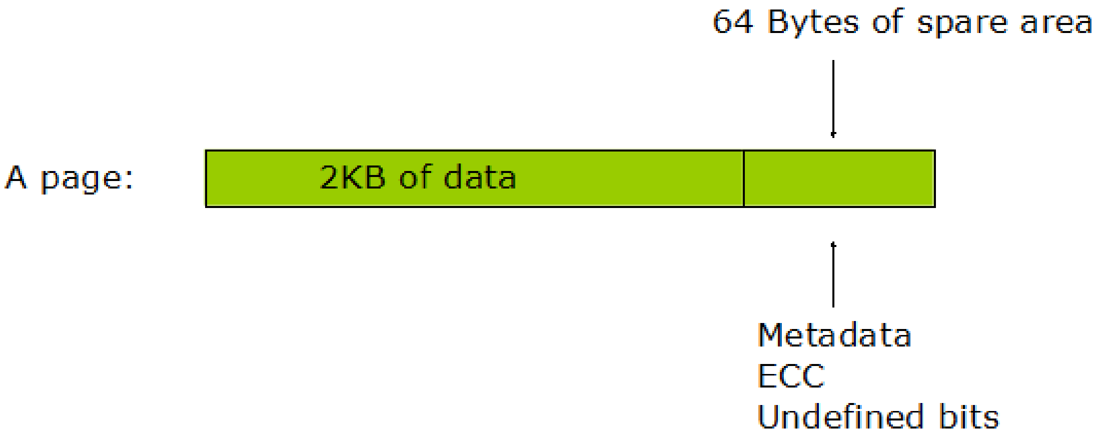
- Partial writing: A page is partitioned into 4 parts, and we can write a part at a time.





Note: This is logic partition.

- A typical NAND page with spare bytes ：

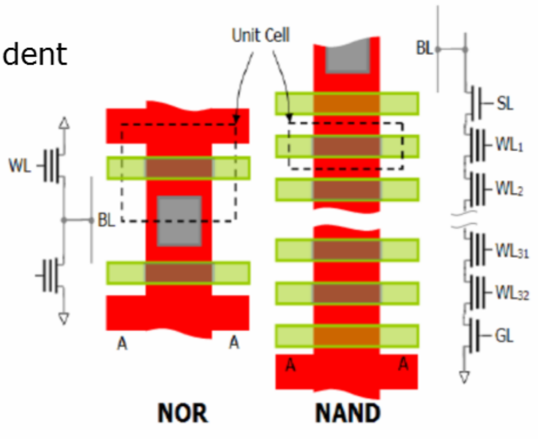


- Comparison of NOR & NAND flash：

Basic difference: Different ways to connect cells in a block.

Additional difference: Ways to inject charge, used voltages.

NOR: cells are independent



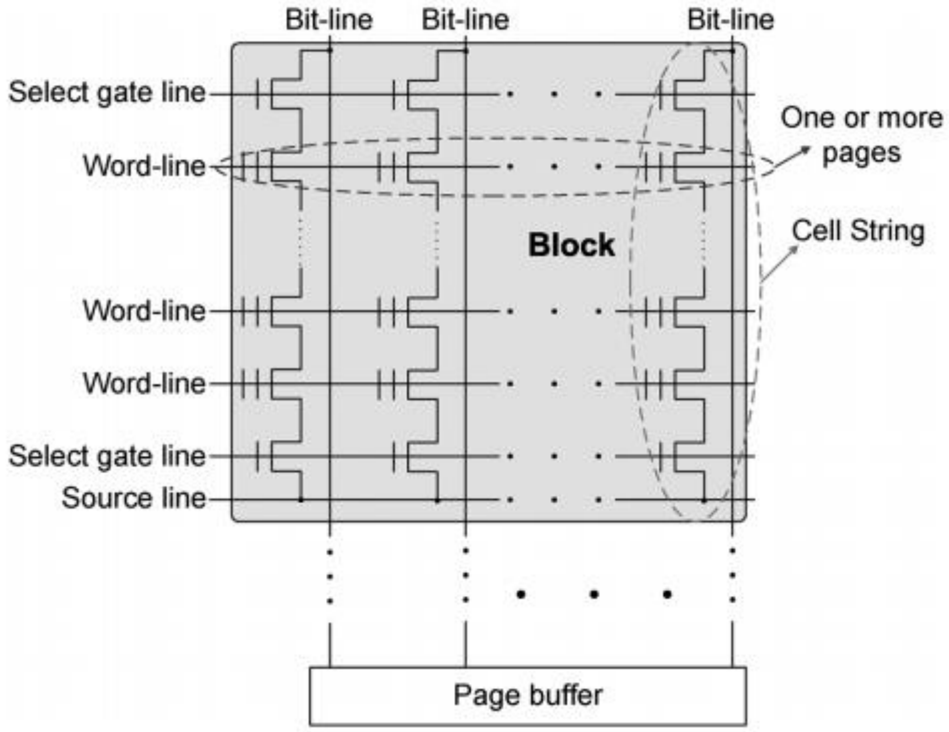
NAND: Cells in the same column are connected (and disturb each other).

Specific Comparison：

	NOR	NAND
擦除方法	Fowler-Nordheim (FN)隧穿	Fowler-Nordheim (FN)隧穿
编程方法	热电子注入	Fowler-Nordheim (FN)隧穿
擦除速度	慢	快
编程速度	快	慢
读速度	快	慢
单元尺寸	大	小
可测量性	困难	容易
最大擦写次数	十万次	百万次
芯片价格	高	低
主要供应商	INTEL ,MICRO	SAMSUNG 、Toshiba
应用	嵌入式系统	大容量存储(U 盘、各种存储卡、MP3 播放器)

1.7 NAND Flash Memory Basics

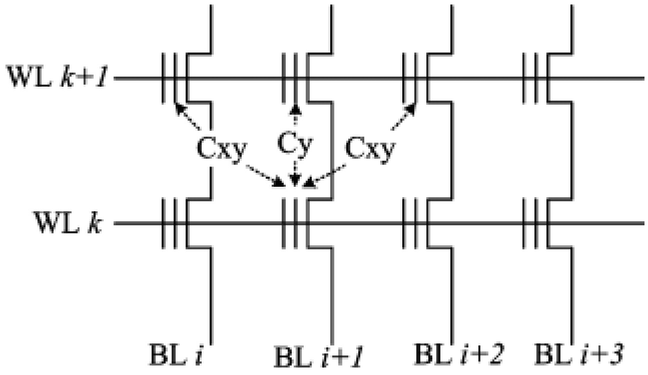
NAND Flash memory cells are organized in an array →block →page hierarchy, as shown in Fig, where one NAND Flash memory array is partitioned into many blocks and each block contains a certain number of pages. Within one block, each memory-cell string (as shown in Fig. 2) typically contains 16－64 memory cells. All the memory cells within the **same block must be erased at the same time.**



All the memory-cell blocks share the bit lines and an on-chip page buffer that holds the data being programmed or fetched. Modern NAND Flash memories use either even/odd bit-line structure , or all-bit-line structure .**For MLC NAND flash memory, all the bits stored in one cell belong to different pages**, which can be either simultaneously programmed at the same time, referred to as full-sequence programming, or sequentially programmed at different time, referred to as multi-page programming.

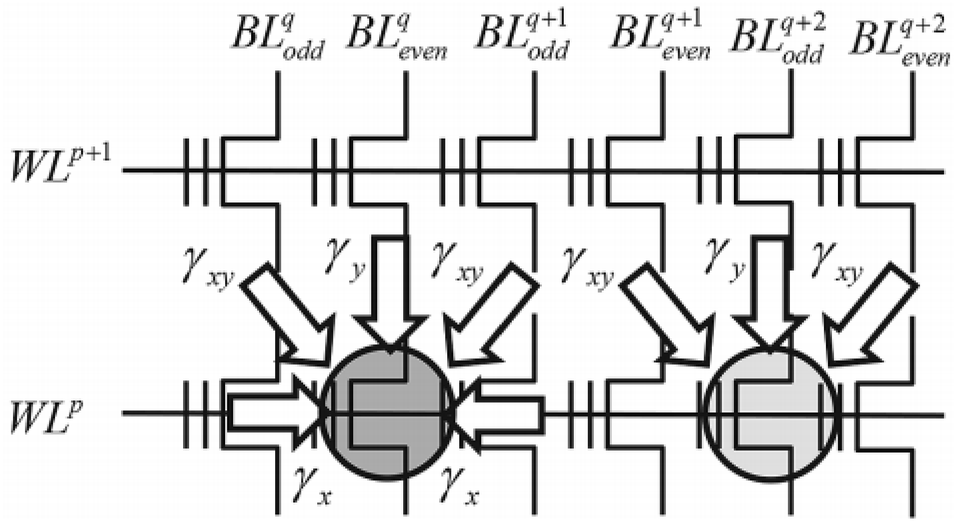
A) all-bit-line structure

In the all-bit-line structure, all the bit lines are accessed at the same time, which **aims to trade peripheral-circuit silicon cost for better immunity to cell-to-cell interference.**



B) even/odd bit-line structure

In the even/odd bit-line structure, even and odd bit lines are interleaved along each word line and are alternatively accessed. Hence, **each pair of even and odd bit lines can share peripheral circuits such as sense amplifiers and buffers**, leading to less silicon cost of peripheral circuits.



Moreover, **a relatively simple voltage-sensing scheme can be used in even/odd bit-line structure**, while a current-sensing scheme must be used in the all-bit-line structure. For MLC NAND Flash memory, all the bits stored in one cell belong to different pages, which can be either **simultaneously programmed at the same time, referred to as full-sequence programming, or sequentially programmed at different times, referred to as multipage programming.** Since full-sequence programming can achieve a higher programming throughput but incur more severe cell-to-cell interference, we mainly consider the full-sequence programming strategy , i.e., under the even/odd bit-line structure, cells in even bit line, referred to as even cells, are programmed first with the full-sequence programming, and then cells in odd bit line, referred to as odd cells, are programmed with the full-sequence programming

1.8 Effects of program/erase cycling

As a cell is cycled the tunneling oxide forms traps, it will broken atomic bonds in oxide matrix due to tunneling. Electrons can more easily leak from the FG to the channel by Trap Assisted Tunneling (TAT) .When filled with electrons, traps can increase the potential barrier, reducing the tunneling current and increase V_{th} .

Flash memory PE cycling causes damage to the tunnel oxide of floating gate transistors in the form of charge trapping in the oxide and interface states, which directly results in threshold voltage shift and fluctuation and hence gradually degrades memory device noise margin. Major distortion sources include

1. Electrons capture and emission events at charge trap sites near the interface developed over PE cycling directly result in memory cell threshold voltage fluctuation,

which is referred to as random telegraph noise (RTN);

2. Interface trap recovery and electron detrapping gradually reduce memory cell threshold voltage, leading to the data retention limitation.

1.8.1 RTN

RTN causes random fluctuation of memory cell threshold voltage, where the fluctuation magnitude is subject to exponential decay. Hence, we can model the probability density function $p_r(x)$ of RTN-induced threshold voltage fluctuation as a symmetric exponential function:

$$p_r(x) = \frac{1}{2\lambda_r} e^{-\frac{|x|}{\lambda_r}}$$

Let N denote the PE cycling number, λ_r scales with N in an approximate power-law fashion, i.e., λ_r is approximately proportional to N_a .

1.8.2 Data retention

Threshold voltage reduction due to interface trap recovery and electron detrapping **can be approximately modeled as a Gaussian distribution** $N(\mu_d, \delta_d^2)$. Both μ_d and δ_d^2 scale with N in an approximate power-law fashion, and scale with the retention time t in a logarithmic fashion. Moreover, the significance of threshold voltage reduction induced by interface trap recovery and electron detrapping is also proportional to the initial threshold voltage magnitude, i.e., the higher the initial threshold voltage is, the faster the interface trap recovery and electron detrapping occur and hence the larger threshold voltage reduction will be.

For example. For the function $N(\mu_d, \sigma_d^2)$ to capture trap recovery and electron detrapping during retention, we set that μ_d scales with $N^{0.5}$ and σ_d^2 scales with $N^{0.6}$, and both scale with $\ln(1+t/t_0)$, where t denotes the memory retention time and t_0 is an initial time and can be set as 1 hour. In addition, as pointed out earlier, both μ_d and σ_d^2 also depend on the initial threshold voltage. Hence we set that both approximately scale $K_s(x - x_0)$, where x is the initial threshold voltage, and x_0 and K_s are constants. Therefore, we have

$$\begin{cases} \mu_d = K_s(x-x_0)K_d N^{0.5}\ln(1+t/t_0) \\ \sigma_d^2 = K_s(x-x_0)K_m N^{0.6}\ln(1+t/t_0) \end{cases}$$

1.9 Cell-to-Cell Interference

In NAND Flash memory, the **threshold-voltage shift of one floating-gate transistor can change the threshold voltages of its neighboring floating-gate transistors through parasitic capacitance-coupling effect . This is referred to as cell-to-cell interference**, which clearly tends to result in data-storage reliability degradation.As the technology continues to scale down and, hence, adjacent cells become closer, the parasitic coupling capacitance(寄生耦合电容) between adjacent cells continues to increase and results in increasingly severe cell-to-cell interference. Recent studies have clearly identified cell-to-cell interference as the major challenge for future NAND Flash memory scaling.

the threshold-voltage change of a victim cell due to cell-to-cell interference can be estimated as

$$F = \sum_k (\Delta V_t^{(k)} \cdot \gamma^{(k)})$$

where $\Delta V_t^{(k)}$ represents the threshold-voltage shift of the kth interfering cell which is **programmed after the victim** cell and the coupling ratio $\gamma^{(k)}$ is defined as

$$\gamma^{(k)} = \frac{C^{(k)}}{C_{total}}$$

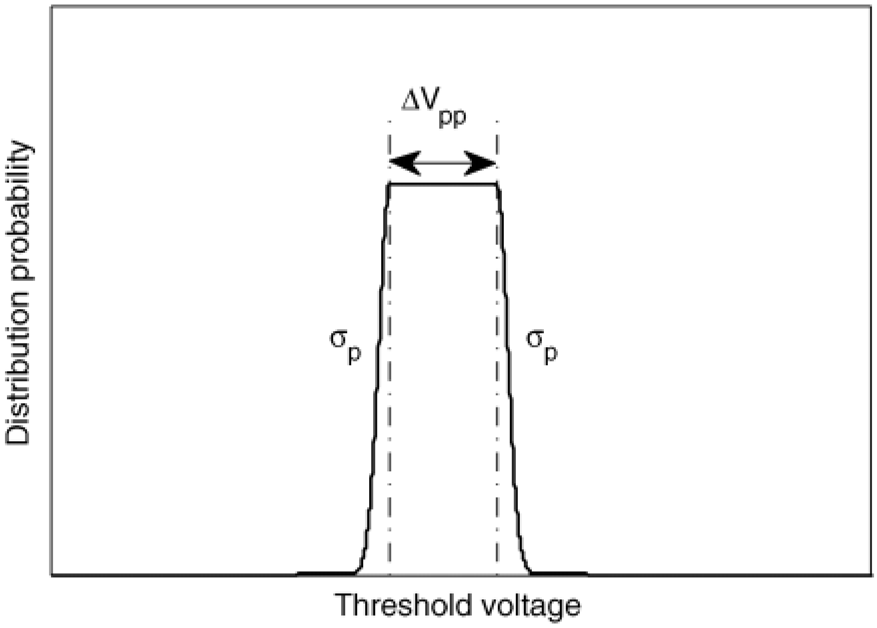
where $C^{(k)}$ is the parasitic capacitance between the interfering cell and the victim cell and C_{total} is the total capacitance of the victim cell. Clearly, **one cell can only be interfered by its neighbors which are programmed after this cell has been programmed**. In practice, only the cell-to-cell interference from immediately adjacent cells are considered because parasitic coupling capacitance quickly diminishes as the cell-to-cell distance increases.

1.10 Cell-Threshold-Voltage Distribution Model

Here, we present a NAND Flash memory-cell threshold-voltage distribution model that is used in all the quantitative studies throughout almost all model. As pointed out earlier, before one Flash memory cell is programmed, it must be erased first, leading to the lowest threshold voltage. The threshold voltage of an erased state tends to have a wide Gaussian-like distribution . Hence, we model the threshold-voltage distribution of the **erased state as**

$$p_e(x) = \frac{1}{\sigma_e\sqrt{2\pi}} \exp^{-\frac{(x-V_e)^2}{2\sigma_e^2}}$$

where V_e and σ_e are the mean and standard deviation of the erased-state threshold-voltage distribution. For the other programmed states, as pointed out earlier, each memory cell is programmed using an iterative program-and-verify approach with a step voltage of ΔV_{pp}. **Due to the process variability, given the word-line programming step voltage during each program-and verify cycle, different cells may experience different threshold voltage boosts with up to** ΔV_{pp}. Since the verify voltage ΔV_p is used to determine whether the programming should continue or stop, the threshold voltage of each programmed state tends to have a uniform distribution over $[V_p, V_p + \Delta V_{pp}]$ with the width of ΔV_{pp}. However, because of the inevitable noise at device and circuit levels, charge leakage, and cell-to-cell interference,such an ideal uniform threshold-voltage distribution is subject to a certain distortion. **We assume that the aggregate effect of all the other distortions, except cell-to-cell interference, tend to add two symmetric Gaussian tails on both sides of the uniform distribution part**, as shown in Fig **:**

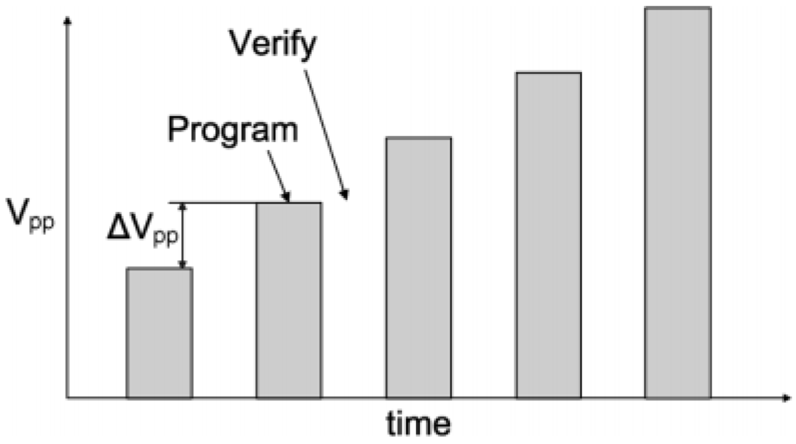


Let P_0 and P_1 denote the probabilities of the uniform distribution and Gaussian tails, respectively. We have the overall **Probability Distribution Function** （PDF） $p_p(x)$ as：

$$p_p(x)=\begin{cases}\dfrac{c_p}{\sigma_p\sqrt{2\pi}}e^{-\frac{(x-V_p)^2}{2\sigma_p^2}},if \quad x<V_p \\ \dfrac{c_p}{\sigma_p\sqrt{2\pi}}e^{-\frac{(x-V_p-\Delta V_{pp})^2}{2\sigma_p^2}}, \qquad if \quad x\geq V_p+\Delta V_{pp} \\ \dfrac{c_p}{\sigma_p\sqrt{2\pi}},else\end{cases}$$

program-and-verify programming strategy：

Under such a program-and-verify programming strategy, each programming state (except the erased state) associates with a verify voltage that is used in the verify operations. Denote the verify voltage of the target programming state as V_p. During each program-and-verify cycle,the floating-gate threshold voltage is V_t first boosted by up to ΔV_{pp} and then compared with V_p .If V_t is still lower than V_p ,the program-and-verify iteration will continue; otherwise, the corresponding bit line will be configured so that further programming of this cell is disabled.



Due to the process variability, we assume that all the coupling ratios γ_x, γ_y, and γ_{xy} follow bounded Gaussian distributions as

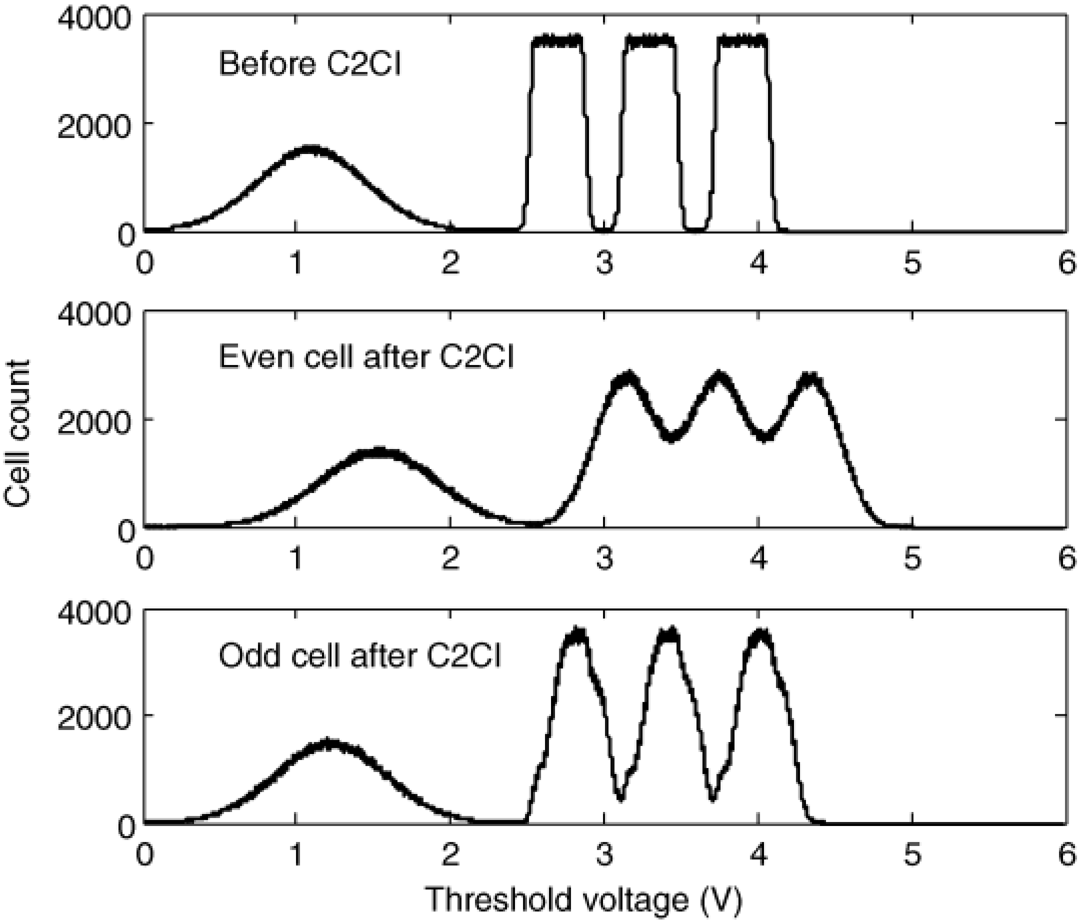
$$p_r(x) = \begin{cases} \frac{c_r}{\sigma_r\sqrt{2\pi}} \cdot e^{-\frac{(x-\mu_r)^2}{2\sigma_r^2}}, & \text{if } |x-\mu_r| \le w_r \\ 0, & \text{else} \end{cases}$$

Where μ_r and σ_r are the mean and standard deviation and c_r is chosen to ensure that the integration of this bounded Gaussian distribution is equal to one. We set $w_r = 0.2\mu_r$ and $w_r = 0.3\mu_r$. Because the wordline pitch variation introduces variations on the μ_r of γ_y and γ_{xy}, we set the μ_r of γ_x as constant while assuming that the μ_r of γ_x and γ_{xy} also follows bounded Gaussian distributions as:

$$p_\mu(t) = \begin{cases} \frac{c_t}{\sigma_t\sqrt{2\pi}} \cdot e^{-\frac{(t-\mu_t)^2}{2\sigma_t^2}}, & \text{if } |t-\mu_t| \le w_t \\ 0, & \text{else} \end{cases} .$$

c_t is chosen to ensure that the integration of this bounded Gaussian distribution is equal to one.

Example: Let us consider a 2-bit/cell NAND Flash memory. For the erased state, its mean and standard deviation are set to 1.1 and 0.35 V, respectively. For the other three programmed states, we set their verify voltages V_p's as 2.55,3.15, and 3.75 V, respectively, and set the program step voltage ΔV_{pp} .Regarding the programming threshold-voltage distribution shown in last Fig, we set the standard deviation σ_p of double-side Gaussian tails as 0.03 and have the constant factor $c_p = 0.2$. Fig shows the simulation results, where cell-to-cell interference is abbreviated as C2CI for the purpose of being concise and where the means of γ_x, γ_y, and γ_{xy} are set to 0.08, 0.064, and 0.0048, respectively. It shows how cell-to-cell interference may distort the cell-threshold-voltage distribution and how the impacts on even and odd cells can differ significantly. The results also suggest that, during the memory read operations, **sensing reference voltages for even and odd cells should be configured differently** in order to minimize sensing BERs in the presence of significant cell-to-cell interference.



参考文献：

1. A 3.3 V 32 Mb NAND flash memory with Incremental Step Pulse Programming Scheme.

2. Anxiao-2012-Coding Methods for Emerging Storage Systems.

3. Cell-to-Cell Interference Compensation Schemes Using Reduced Symbol Pattern of Interfering Cells for MLCNAND Flash Memory.

4. Erro1r Correction Codes and Signal Processing in Flash Memory

5. On the Use of Soft-Decision Error-Correction Codes in NAND Flash Memory.

6. Using Data Post-compensation and Pre-distortion to Tolerate Cell-to-Cell Interference in MLC NAND flash memory.

7. Introduction to Flash Memory

8.Error Control Coding for Multilevel Cell Flash Memories Using Nonbinary Low-Density Parity-Check Codes

第二章 LDPC Codes

2.1 Low-Density Parity-Check Codes

低密度校验码(Low density parity-check code, LDPC)首先是由 Gallager 在 20 个世纪 60 年代提出的，但由于当时计算机硬件的限制，使 LDPC 码没有得到很大的重视，20 世纪 90 年代末，随着计算机运算能力的大大提升，LDPC 码由 Neal 与 Mackey 等人重新发现。近几年来，LDPC 码因其优异的性能和 相对较低的解码运算复杂度而成为理论界和工业界研究的热点之一。LDPC 码的译码算法通常被称为和积译码算法(sum product algorithm，SPA)或置信传播算法(BP 算法)，在置信传播算法中，与译码有关的消息沿着边所在节点之间进行传播。而且进一步研究表明：基于非规则双向图的 LDPC 长码的性能可以优于 Turbo 码，具有更低的线性译码复杂度。

2.2 LDPC 码的描述和图模型表达

LDPC 码是一类线性分组码，由它的校验矩阵来定义，设码长为 N，信息位为 K，校验位为 M =N−K，码率为 R =K/N，则码校验矩阵 H 是一个 M×N 的矩阵。

定义：二元 LDPC 码的校验矩阵 H 矩阵要满足以下四个条件：

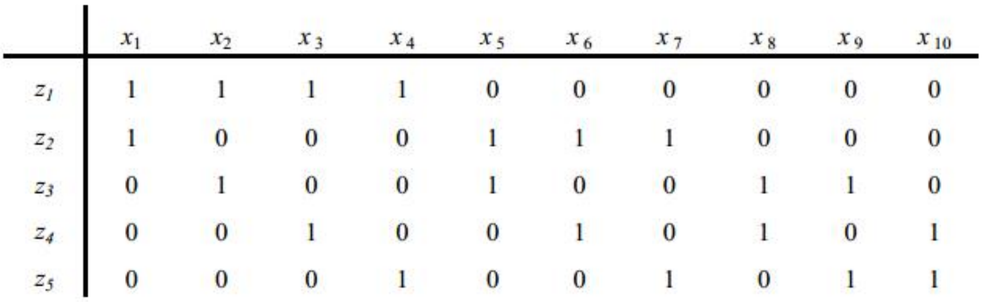
(1) H 矩阵的每行有 ρ 个“1” ；

(2) H 矩阵的每列有 γ 个“1” ；

(3) H 矩阵的任意两行（或两列）间共同为“1”的个数不超过 1；

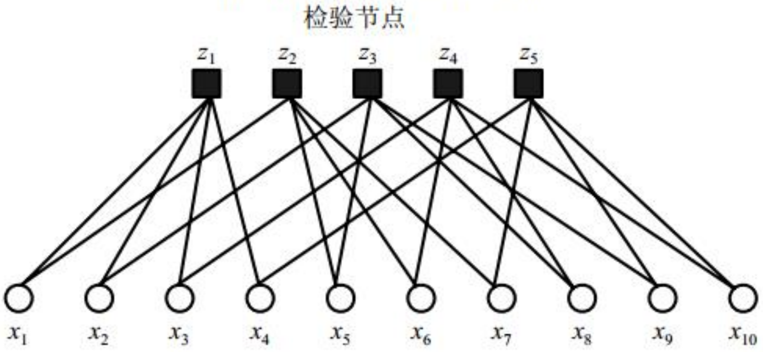
(4) 与码长和 H 矩阵中的行数相比较，ρ 和 γ 很小，也就是说矩阵中很少一部分元素非零，其他大部分元素都是零，LDPC 码的校验矩阵是稀疏矩阵。

满足以上四个条件的 H 校验矩阵对应的 **LDPC 码一般表述为(N, γ，ρ)**，码率则为 **R ≥ 1-γ/ρ**。LDPC 码的校验矩阵对应可用一个 **Tanner（双向）图**表示，图的下边有 N 个节点，**每个节点表示码字的信息位，称为信息节点**$\{x_j, j = 1,2,3,......, N\}$ **,是码字的比特位，对应于校验矩阵各列，信息节点也称为变量节点**；**上边有 M 个节点，每个节点表示码字的一个校验集，称为校验节点**$\{z_i, i = 1,2,3,......, M\}$**，代表校验方程，对应于校验矩阵各行**；与校验矩阵中“1”元素相对应的左右两节点之间存在连接边。我们将这条边两端的节点称为相邻节点，每个节点相连的边数称为该节点的度数（Degree)，每个信息节点与 γ 个校验节点相连，称该变量节点的度数为 γ；每个校验节点与 ρ 个信息节点相连，称该校验节点的度数为 ρ。例如：（10,2,4）LDPC 码的校验矩阵和双向图如下图所示，信息节点的度数为 2，校验节点，的度数为 4。



	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
z_1	1	1	1	1	0	0	0	0	0	0
z_2	1	0	0	0	1	1	1	0	0	0
z_3	0	1	0	0	1	0	0	1	1	0
z_4	0	0	1	0	0	1	0	1	0	1
z_5	0	0	0	1	0	0	1	0	1	1

(a) (10,2,4)LDPC 码校验矩阵



(b) (10,2,4)LDPC 码的双向图

一般情况下校验矩阵是随机构造的，因而是非系统化的。**在编码时，对校验矩阵 H 进行高斯消去**，可得：

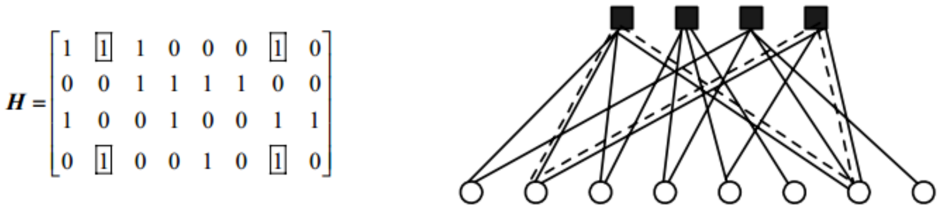
$$H = [I \quad P]$$

得生成矩阵为：

$$G = [-P^{-1} \quad I]$$

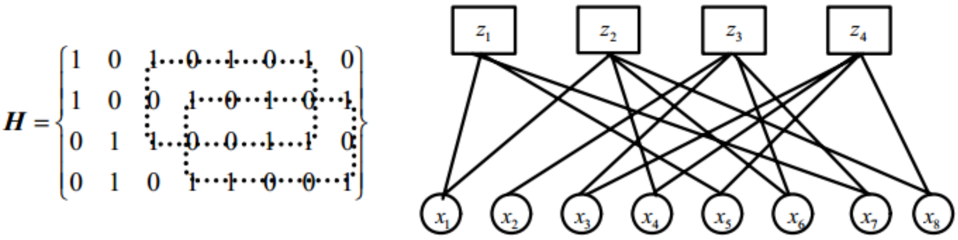
2.3 LDPC 码的环

一个 LDPC 码的矩阵的结构对码的性能有决定性的影响。由于 LDPC 码译码采用迭代译码，其算法的推导是基于在节点间传递的信息统计独立，当 LDPC 码校验矩阵对应的双向图中有环存在时，某一节点发出的信息经过一个环长的传递会被传回本身，从而造成自身信息的叠加，所以**环越小，其传递的信息相干性就越强，置信度就越低，破坏了独立的假设，**影响译码的准确性。所谓环也称为循环（cycle），在 LDPC 码的双向图中我们总是希望大环多，小环少，**所以我们总希望与一个信息节点连接的检验节点越多越好，得到的信息量就大，而与某一检验节点连接的信息节点越少越好，信息的置信度就越高。**



$$H=\begin{bmatrix}1 & \boxed{1} & 1 & 0 & 0 & 0 & \boxed{1} & 0\\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0\\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1\\ 0 & \boxed{1} & 0 & 0 & 1 & 0 & \boxed{1} & 0\end{bmatrix}$$

图 3-6　环在校验矩阵和双向图中的表示



$$H=\begin{bmatrix}1 & 0 & 1 & 0 & 1 & 0 & 1 & 0\\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1\\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0\\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1\end{bmatrix}$$

图 3-7　校验矩阵和双向图中的环

2.4 Decoding based on belief propagation

Methods for decoding LDPC codes can be classified into three general categories: **1) soft-decision decoding** in which the decoder processes the values of the received symbols as real numbers and thus retains all the information provided by the channel; **2) hard-decision decoding** in which the decoder quantizes the received symbols as elements in the same set of transmitted symbols, typically zeros and ones, before processing and thus loses some of the channel information for the sake of simplifying the decoder; and **3)reliability-based decoding** in which the decoder processes the quantized received symbols as in hard-decision decoding but also uses reliability measures extracted from the unquantized symbols to improve performance with some increase in decoding complexity.

Soft decision decoding algorithms for LDPC codes are mostly iterative in nature and devised based on belief propagation.The most well known soft-decision iterative decoding algorithms **based on belief propagation** are the sum-product algorithm (SPA) and its simplified versions. The SPA provides the best performance but requires the largest computational complexity among all the decoding algorithms devised for LDPC codes. The most well known simplified version of the SPA is the min-sum algorithm . It requires less computational complexity than the SPA.

2.4.1 和积译码算法

LDPC 码的迭代概率译码算法——和积译码算法(sum-product algorithm (SPA))，它是基于置信传播(belief propagation algorithm)的迭代概率译码算法，是逐符号软判决译码方法。

设一个 N 长 LDPC 码的校验矩阵 $H=(h_{ij})M\times N$ 。下图是其双向图，码字 $X=\{x_1,x_2,x_3......x_N\}$ 表示为一组信息节点，$\{x_j:j=1,2,......N\}$，$\{z_1,z_2,z_3,......,z_M\}$ 表示为一组校验节点 $\{z_i:i=1,2,......M\}$，仅当 $h_{ij}=1$ 时，节点 x_j 到 z_i 由一条有向边连接。

每一个信息节点称为相邻的校验节点的父节点，每一个校验节点称为相邻的信息节点的子节点。

收到的实数向量集合记为{r}，有信息节点x_j，我们将信息节点X满足包含x_j的所有校验方程这个事件记为S，比特 x_j=1（或x_j=0）关于{ r }和S的条件概率$P_r(x_i=1)|\{r\},S)$（或$P_r(x_i=0)|\{r\},S)$），若

$$\frac{P_r(x_j=0)|\{r\},S)}{P_r(x_j=1)|\{r\},S)}\geq 1$$

成立，则接受符号r_i的估计$\hat{x}_j$=0，若不成立，则$\hat{x}_j$=1，

在二进制输入高斯信道(Gaussian Channel)下，设一个长度为 N 的码字$c=(c_1,c_2,...,c_N)$,c_n=0或l，采用BPSK调制后的信号为$x=(x_1,x_2,...,x_N)$，$x_n=2c_n-1$，该信号经过均值为零、功率谱为$N_0/2$高斯信道后，接收端收到的信号为$y_n=x_n+n$，n是均值为零，方差为$N_0/2$的随机数。为方便描述，对BP迭代译码算法中涉及的符号做如下说明。$N(m)=\{n:H_{mn}=1\}$表示与校验节点m相连的所有变量节点的集合，$N(m)\backslash n$表示不包含变量节点n的集合；$M(n)=\{m:H_{mn}=1\}$表示与变量节点n相连的所有校验节点的集合，$M(m)\backslash n$表示不包含校验节点m的集合；P_n表示接收信号获得的变量x_n的LLR信息

$$P_n=\log\frac{P(x_n=1|y_n)}{P(x_n=0|y_n)}=4\frac{y_n}{N_0}$$

R_{mn}表示从校验节点m发送到变量节点玎的LLR信息；Q_{mn}表示从变量节点疗发送到校验节点n的LLR信息；Q_n表示每次迭代后计算获得的变量x_n的LLR信息。

则BP迭代译码中校验节点到变量节点消息更新规则为：

$$R_{mn}^{new}=\varphi^{-1}\left(\sum_{n^{'}\in N(m)\backslash n}\varphi(Q_{mn^{'}})\right)$$

式中：$\varphi(x)=\left(sign(x),-\log\tanh\frac{|x|}{2}\right)$，$\varphi^{-1}(x)=(-1)^{sign(x)}\times\left(-\log\tanh\frac{x}{2}\right)$

变量节点到校验节点的消息更新规则为：

$$Q_{mn}^{new}=P_n+\sum_{m'\in M(n)\backslash m}R_{m'n}$$

BP 算法的步骤可以简单的叙述为：

定义 $L(q_{ij})$ 和 $L(r_{ji})$ 分别表示变量节点 i 向校验节点 j 和校验节点 j 向变量节点 i 传输的对数似然比信息。

1)初始化

设 y_i 是信息经信道后的输出，即译码器的输入，使用 y_i 对变量节点译码器进行初始化，对　二进制加性白高斯信道，初始化的值为：

$$L(q_{ij}) = \frac{2y_i}{\sigma^2}$$

2)校验节点译码

每一个变量节点通过与之相连的边把信息输入相邻的校验节点，校验节点计算：

$$L(r_{ji}) = \prod_{i' \in V_{j \backslash i}} a_{i'j} \varphi(\sum_{i' \in V_{j \backslash i}} \varphi(\beta_{i'j}))$$

其中 $\phi^{-1}(x) = -\log[\tan\mathrm{h}(\frac{x}{2})] = \log\frac{e^x+1}{e^x-1}$， $a_{i'j} = sign[L(q_{i'j})]$， $\beta_{i'j} = | L(q_{i'j}) |$

3)变量节点译码

每个变量节点从相邻的校验节点获得信息，计算：

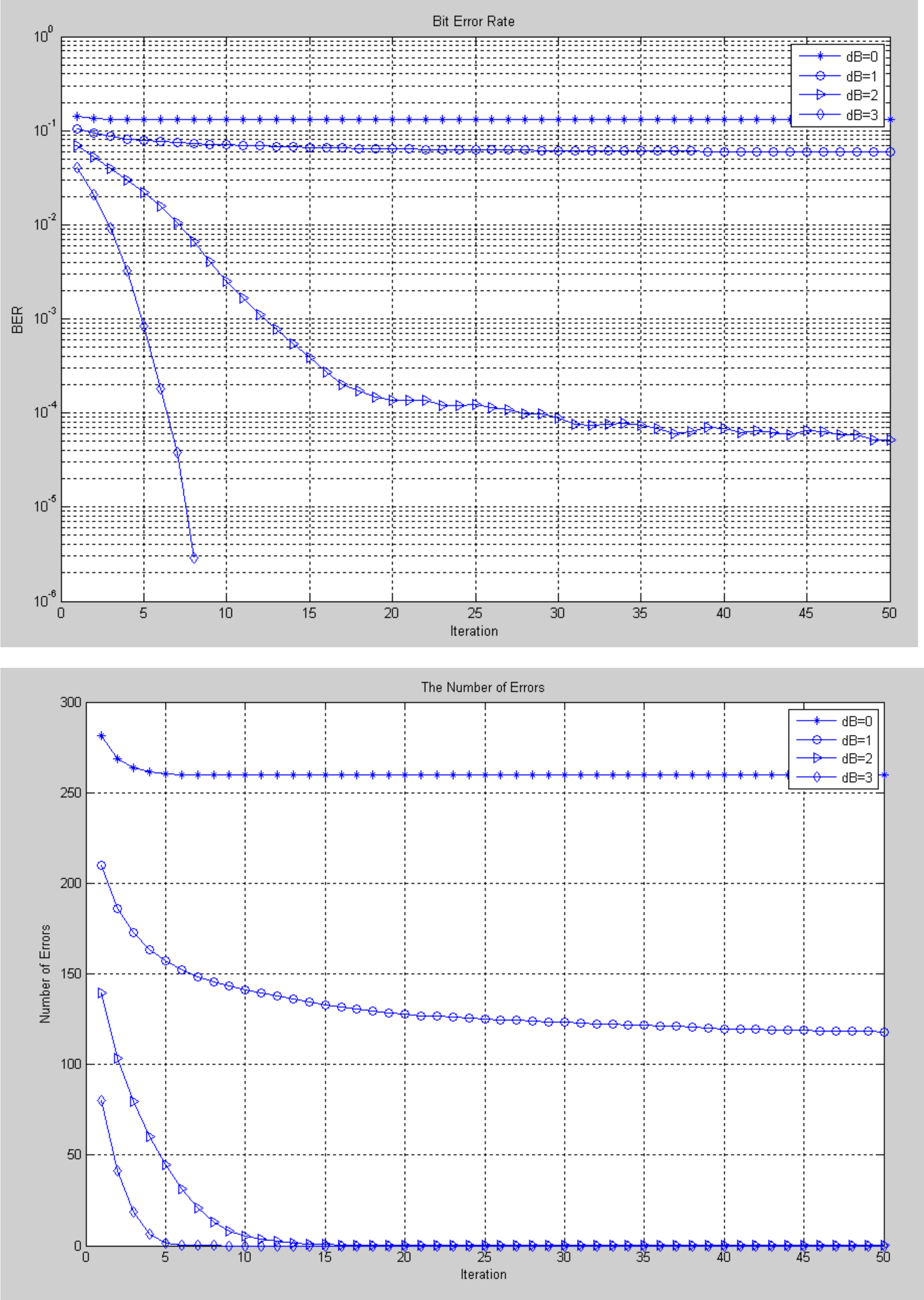
$$L(q_{ij}) = L(c_i) + \sum_{j' \in c_{i \backslash j}} L(r_{j'i})$$

4)硬判决

计算 $L(Q_i) = L(c_i) + \sum_{j' \in C} L(r_{j'i})$ 并判决：

$$\hat{c} = \begin{cases} 1 & L(Q_i) < 0 \\ 0 & others \end{cases}$$

如果 $\hat{c} H = 0$，说明以正确译码，译码终止，否则回到第二步继续循环，直到正确译码或达到预定的最大迭代次数。



BP algorithm with BPSK modulation H=1000*2000 (1500frame)

2.4.2 Min-Sum Algorithm

文献：Hagenauer, J. ; Tech. Univ. Munchen, Germany ; Offer, E. ; Papke, L.

《Iterative decoding of binary block and convolutional codes》

LLR 域 BP 算法中的校验节点计算式(2)，可以看作各个输入的 LLR 信息两两作 LLR 域的运算处理。

文献给出了当校验式的两个输入节点满足统计独立条件时，LLR 域的相加运算定义：

$$L(v_1)\oplus L(v_2)=L(v_1\oplus v_2)=\ln(\frac{1+e^{L(v_1)+L(v_2)}}{e^{L(v_1)}+e^{L(v_2)}})$$

同时，文献在此基础上对上式作了近似处理方案：

$$\begin{aligned}&L(v_1+v_2)\\&=\operatorname{sgn}(L(v_1))\cdot\operatorname{sgn}(L(v_2))\cdot\min(|L(v_1)|,|L(v_2)|)+\log(1+e^{-|L(v_1)+L(v_2)|})-\log(1+e^{-|L(v_1)+L(v_2)|})\\&\approx\operatorname{sgn}(L(v_1))\cdot\operatorname{sgn}(L(v_2))\cdot\min(|L(v_1)|,|L(v_2)|)\end{aligned}$$

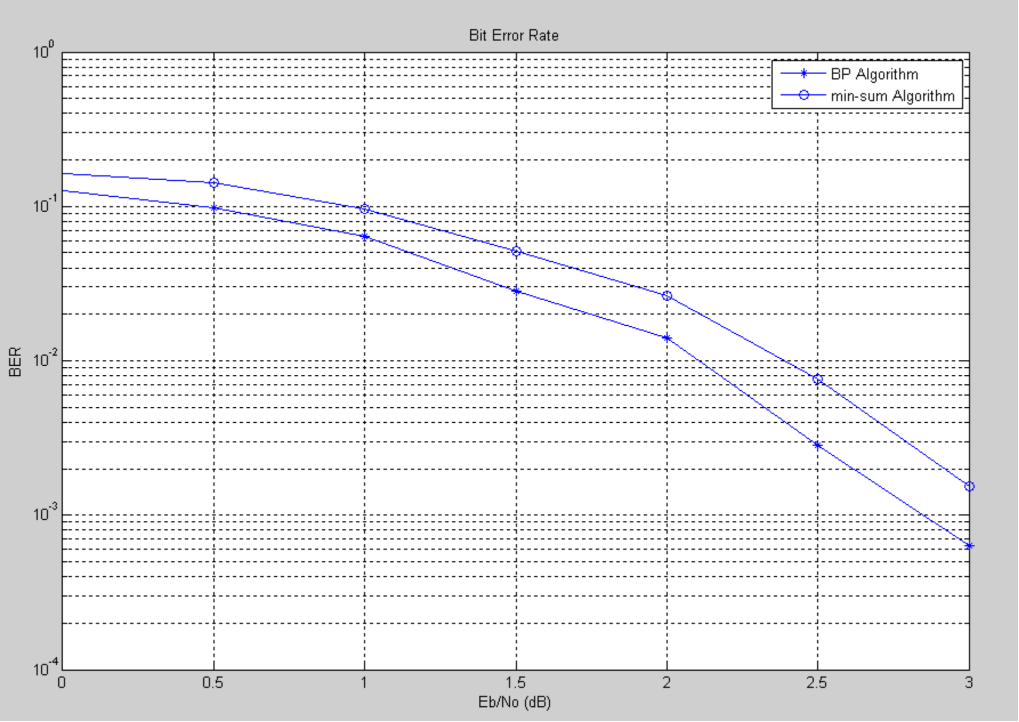
min-sum 算法运动了上式对 BP 算法进行了简化处理，即对检验点计算简化为：

$$L(r_{ji})=\prod_{i'\in V_{j\backslash i}}a_{i'j}\varphi(\sum_{i'\in V_{j\backslash i}}\varphi(\beta_{i'j}))$$

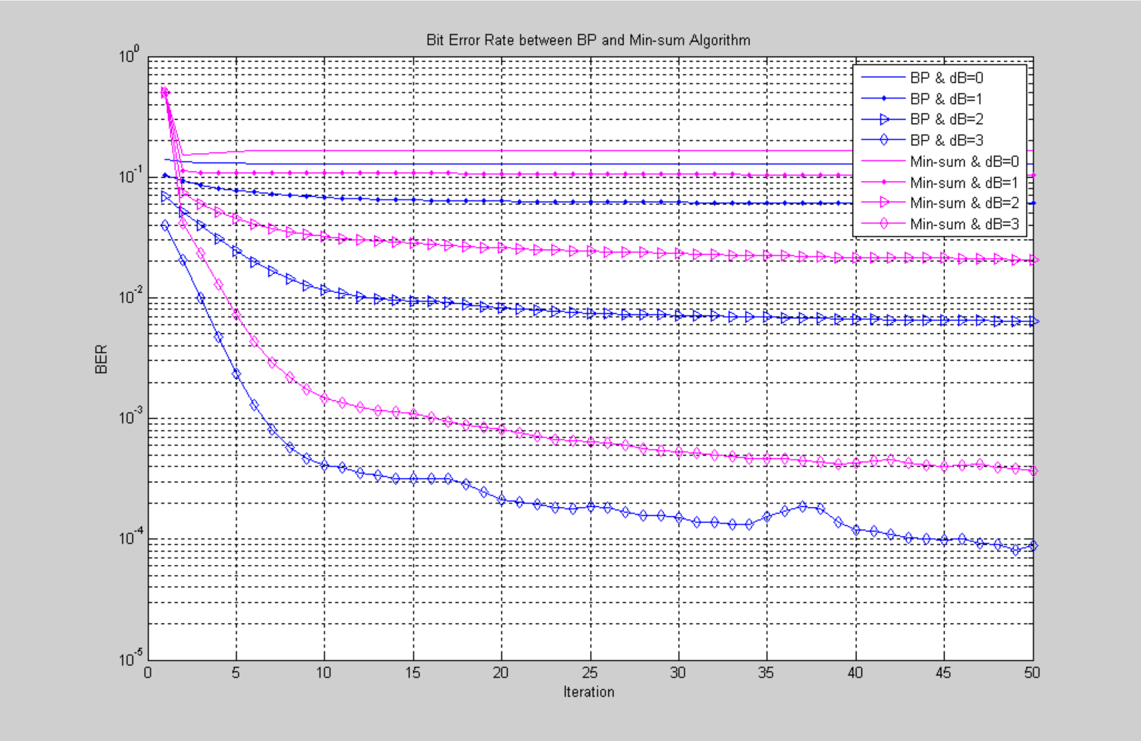
简化为

$$L(r_{ji})=\prod_{i'\in V_{j\backslash i}}a_{i'j}\min_{i'\in V_{j\backslash i}}\beta_{i'j}$$

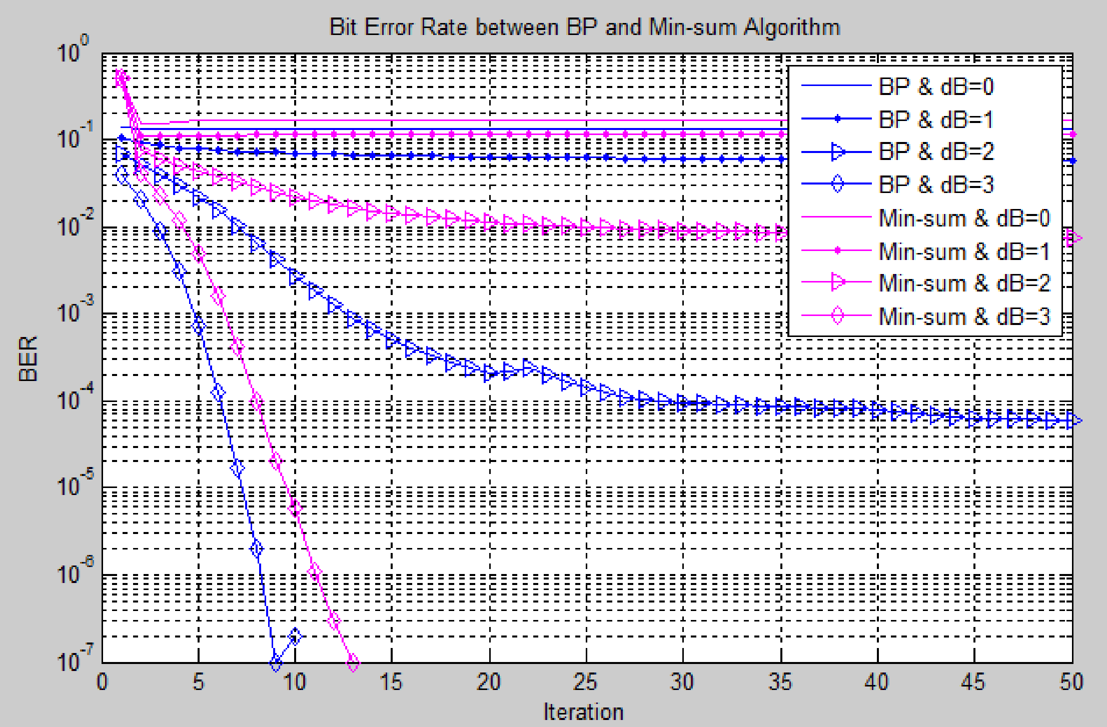
简化后大大降低了计算的复杂度，但同时也加大了误差程度，min-sum 算法的误码率性能和 BP 算法相比，有较大的差距。



BP & Min-sum algorithm of H=128*256 with max iteration=50



BP & Min-sum algorithm with H of 204*408 with max iteration=50



BP & Min-sum algorithm with H of 1000*2000 with max iteration=50 (5000frame)

2.4.3 自适应均衡器

均衡器针对于不同频率的信号进行电信号的放大或衰减使混频信号达到一定的有益效果。大概原理就是不同频率的信号分离，然后放大或衰减，然后混合的过程。

均衡器技术是通信系统中的一项重要技术，分为两种方式：**频域均衡和时域均衡**。频域

均衡是利用可调滤波器的频率特性来弥补实际信道的幅频特性和群延时特性，使包括均衡器在内的整个系统的总频率特性满足无码间干扰传输条件。时域均衡是直接从时间响应角度考虑，使包括均衡器在内的整个传输系统的冲激响应满足无码间干扰条件。频域均衡满足奈奎斯特整形定理的要求，仅在判决点满足无码间干扰的条件相对宽松一些。所以，在数字通信中一般采用时域均衡。**时域均衡器可以分两大类：线性均衡器和非线性均衡器。**如果接收机中判决的结果经过反馈用于均衡器的参数调整，则为非线性均衡器；反之，则为线性均衡器。在线性均衡器中，最常用的均衡器结构是线性横向均衡器，它由若干个抽头延迟线组成，延时时间间隔等于码元间隔 。非线性均衡器的种类较多，包括判决反馈均衡器(DFE)、最大似然(ML)符号检测器和最大似然序列估计等。均衡器的结构可分为横向和格型等。因为很多数字通信系统的信道(例如无线移动通信信道)特性是未知和时变的，要求接收端的均衡器必须具有自适应的能力。所以，均衡器可以采用自适应信号处理的相关算法，以实现高性能的信道均衡，这类均衡器称为自适应均衡器。

自适应均衡器的工作过程包含两个阶段，**一是训练过程，二是跟踪过程**。在训练过程中，发送端向接收机发射一组已知的固定长度训练序列，接收机根据训练序列设定滤波器的参数，使检测误码率最小。典型的训练序列是伪随机二进制信号或一个固定的波形信号序列，紧跟在训练序列后面的是用户消息码元序列。接收机的自适应均衡器采用递归算法估计信道特性，调整滤波器参数，补偿信道特性失真，训练序列的选择应满足接收机均衡器在最恶劣的信道条件下也能实现滤波器参数调整，所以，训练序列结束后，均衡器参数基本接近最佳值，以保证用户数据的接收，均衡器的训练过程成功了，称为均衡器的收敛。在接收用户消息数据时，均衡器还不断随信道特性的变化连续地改变均衡器参数。 均衡器的收敛时间受均衡算法、均衡器结构和信道特性的变化情况所决定。通常，**均衡器需要通过重复性地周期训练保证能够一直有效地抑制码间干扰**。所以，用户数据序列需要被分割成数据分组或时隙分段发送。 均衡器通常工作在接收机的基带或中频信号部分，基带信号的复包络含有信道带宽信号的全部信息，所以，均衡器通常在基带信号完成估计信道冲激响应和解调输出信号中实现自适应算法等

2.5 Reliability-Based Decoding

The most well known reliability-based decoding algorithms for LDPC codes are various weighted BF (WBF)algorithms . WBF-algorithms provide trade-offs between the high performance of the SPA and the simple decoding complexity of the hard-decision OSMLGD and BF-algorithms. The best WBF-algorithm, in performance, performs well within1 dB from the SPA.

2.5.1 Majority-Logic Decoding

Majority-logic decoding (MLGD) was first devised by Reed in 1954 and later extended by Massey in 1963 . Here, **we present the OSMLGD algorithm for LDPC codes.**

Let C be a(γ , ρ) LDPC code of length n given by an RC-constrained $m \times n$ matrix H over GF(2) with column and row weights γ , ρ respectively. Assume transmission using BPSK signaling with unit energy per signal. Let $y = (y_0, y_1, ..., y_{n-1})$ be the sequence of samples at the output of the channel receiver sampler.

For $0 \leq j < n$ suppose each sample y_i of y is quantized into two levels and decoded independently based on the following hard-decision rule:

$$\begin{cases} z_j = 0, \text{for } y_j \leq 0, \\ z_j = 1, \text{for } y_j > 0. \end{cases}$$

Let $h_0, h_1, h_2......h_{m-1}$ denote the rows of H,Where the ith row h_i is an n-tuple over GF(2) , $h_i = (h_{i,0}, h_{i,1},..., h_{i,n-1})$,for $0 \leq i < m$,to decode the hard-decision received vector z,the first step is to compute its m-tuple syndrome:

$$\mathbf{s} = (s_0, s_1, \ldots, s_{m-1}) = \mathbf{z}\mathbf{H}^T,$$

where, for $0 \leq i < m$,

$$s_i = \mathbf{z} \cdot \mathbf{h}_i = z_0 h_{i,0} + z_1 h_{i,1} + \cdots + z_{n-1} h_{i,n-1}.$$

For the OSMLGD, the γ orthogonal check-sums in S_j are used to decoded z_j If a clear majority of the check-sums in S_j assumes the value “1” , z_j is assumed to be erroneous and is decoded into its one’ s-complement, $1+z_j$ (the addition+ is modulo-2 addition); otherwise, z_j is assumed to be error free and remains unchanged. If the received vector z contains $[\gamma/2]$ or fewer errors, the above decoding corrects all the errors in z

For $0 \leq j < n$,define the following (integer) sum:

$$\Gamma_j = \sum_{s_i \in S_j} (2s_i - 1).$$

Since each check-sum s_i is either 0 or 1, the value of Γ_j is an integer ranging from $-\gamma$ to $+\gamma$ .It is easy to see that Γ_j >0 ,if and only if a majority of the check-sums in S_j assumes the value. Consequently, we can use Γ_j as the decoding function for z_j .The OSMLGD rule is given as follows: for $0 \leq j < n$,the jth received bit z_j is changed to its one’ s-complement $1+z_j$, if

Γ_j >0;otherwise z_j remains unchanged.

2.5.2 A Soft Reliability-Based Iterative MLGD-Algorithm

Over the BI-AWGN channel with BPSK signaling, Let y be the corresponding received sequence of unquantized samples. Suppose the samples of **y** are symmetrically clipped and uniformly quantized into 2^b intervals, symmetric with respect to the origin(关于原点对称),. Each interval has a length Δ and is represented by b bits. For $0 \le j < n$,let q_j **denote the quantized value of the sample** y_j **which is an integer representation of one of the** $2^b - 1$ **quantization intervals.** Therefore, the range of q_j is $[-(2^{b-1}-1),+(2^{b-1}-1)]$. If the magnitude $|y_j|$ exceeds the quantization range ($2^b - 1$)Δ, then set $q_j = 2^{b-1} - 1$ .With this quantization, the magnitude $|q_j|$ of the quantized sample q_j gives a soft measure of the reliability of the **hard decision received bit** z_j. In the following, we develop an RBIMLGD-algorithm using q_j as the initial reliability measure of a received bit z_j at the beginning of decoding process. This reliability measure will be improved through the decoding iterations. Since a soft reliability measure is used in decoding a received bit, the decoding algorithm to be developed is referred to as a soft RBI (SRBI)-MLGD-algorithm.

Since each check-sum s_i in S_j contains the received bit z_j **,**it can be expressed as the following modulo-2 sum:

$$s_i = z_j + \sigma_{i,j}$$

Where

$$\sigma_{i,j} = \sum_{l \in N_i \setminus j} z_l h_{i,l}$$

The sum $\sigma_{i,j}$ is simply the extrinsic-information contributed to the received bit z_j by the other received bits participating in the check-sum s_i in S_j orthogonal on z_j. If we remove z_j from each check-sum s_i in Γ_i given by above, we obtain the following sum:

$$E_j = \sum_{i \in M_j} (2\sigma_{i,j} - 1)$$

Since $\sigma_{i,j}$ is either 0 or 1, the value of E_j is an integer ranging from $-\gamma$ to $+\gamma$.

In RBIMLGD-algorithm,follows from above equals:

$$\sigma_{i,j}^{(k)} = s_i^{(k)} + z_j^{(k)} = \sum_{l \in N_i \setminus j} z_l^{(k)} h_{i,l}$$

And $$E_j^{(k)} = \sum_{i \in M_j} (2\sigma_{i,j}^{(k)} - 1)$$

The value of $E_j^{(k)}$ is an integer ranging from $-\gamma$ to $+\gamma$ .Then, for $0 \le k < k_{\max}$ and $0 \le j < n$ the reliability of the *j*th received bit $z_j^{(k+1)}$ at the *(k+1)*th iteration is given by

$$R_j^{(k+1)} = R_j^{(k)} + E_j^{(k)}$$

For k=0 and $0 \le j < n$ **,the initial reliability** $R_j^{(0)}$ **of the received bit** $z_j^{(0)} = z_j$ **is set to** q_j **.**

the SRBIMGLD-algorithm for an LDPC code can be formulated as follows:

Initialization:Set k=0; $z^{(0)} = z$ **and the maximum number of iterations to** $k_{\max}$,for $0 \le j < n$,set $R_j^{(0)} = q_j$

1. Compute the syndrome $s^{(k)} = z^{(k)} H^T$ **of** $z^{(k)}$ **,if** $s^{(k)}$ **=0,stop decoding and output** $z^{(k)}$ **as the decoded codeword.**

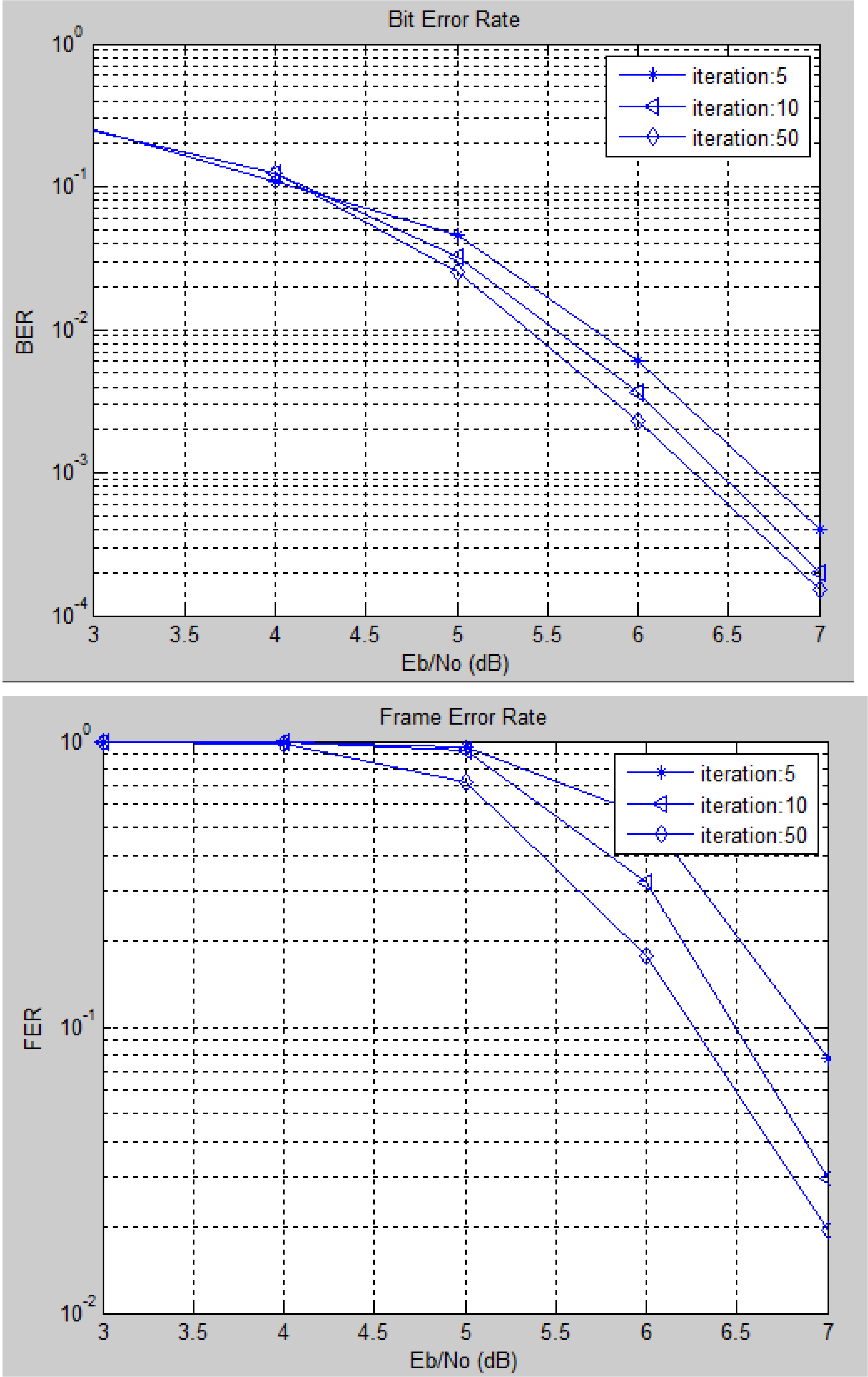
2. If k= $k_{\max}$ **,stop decoding and declare a decoding failure.**

3. Update the reliability measure of all the received bits of $z^{(k)}$ **,** $R_j^{(k+1)} = R_j^{(k)} + E_j^{(k)}$ **,for** $0 \le j < n$ **.store the update reliability measure of the received bits in** $z^{(k)}$ **.**

4. $k \leftarrow k+1$**.for** $0 \le j < n$ **,make the hard-decision :1) if** $R_j^{(k)} \le 0$ **,** $z_j^{(k)}$ **=0. 2) if** $R_j^{(k)} > 0$ **,** $z_j^{(k)}$ **=1. Form a new received vector** $z^{(k)}$ **and back to step 1.**

When updating the reliability of a received bit $z^{(k)}$ at step 2 in the *k*th iteration ,if the magnitude $| R_j^{(k)} + E_j^{(k)} |$ of the sum $R_j^{(k)} + E_j^{(k)}$ is greater than quantization range ($2^{b-1} - 1$) ,we set the magnitude $| R_j^{(k+1)} |$ of the reliability $R_j^{(k+1)}$ of the new received bit

$z^{(k+1)}$ to ($2^{b-1}-1$) (the maximum measure set for a received bit by quantization).



Ber & Fer for H with 1008 × 504 under AWGN with BPSK modulation
Catch 60 Error Frames

2.5.3 A Hard Reliability-Based Iterative MLGD-Algorithm

The SRBI-MGLD-algorithm presented in the last section can be modified by replacing the soft reliability measures of received bits by some type of hard reliability measures. Such modification results in a hard reliability-based iterative (HRBI)-MLGD-algorithm.

Using the decoding rule of the OSMLD-algorithm, decoding of a received bit z_j is most reliable if the value of Γ_j is either $-\gamma$ or $+\gamma$. Based on the above facts, a reasonable assignment of the initial reliability measure to a received bit z_j is: 1) $-\gamma$ to z_j =0; and 2) $+\gamma$ to z_j =1.

With the above initial hard reliability measures of received bits, the HRBI-MLGD algorithm can be formulated as follows:

Initialization: Set k=0; $z^{(0)} = z$ and the maximum number of iterations to $k_{\max}$,for $0 \le j < n$,set $R_j^{(0)}$=$-\gamma$ for $z_j^{(0)}$=0 and $R_j^{(0)}$=$+\gamma$ for $z_j^{(0)}$=1 .

1. Compute the syndrome $s^{(k)} = z^{(k)}H^T$ of $z^{(k)}$,if $s^{(k)}$ =0,stop decoding and output $z^{(k)}$ as the decoded codeword.

2. If k= $k_{\max}$,stop decoding and declare a decoding failure.

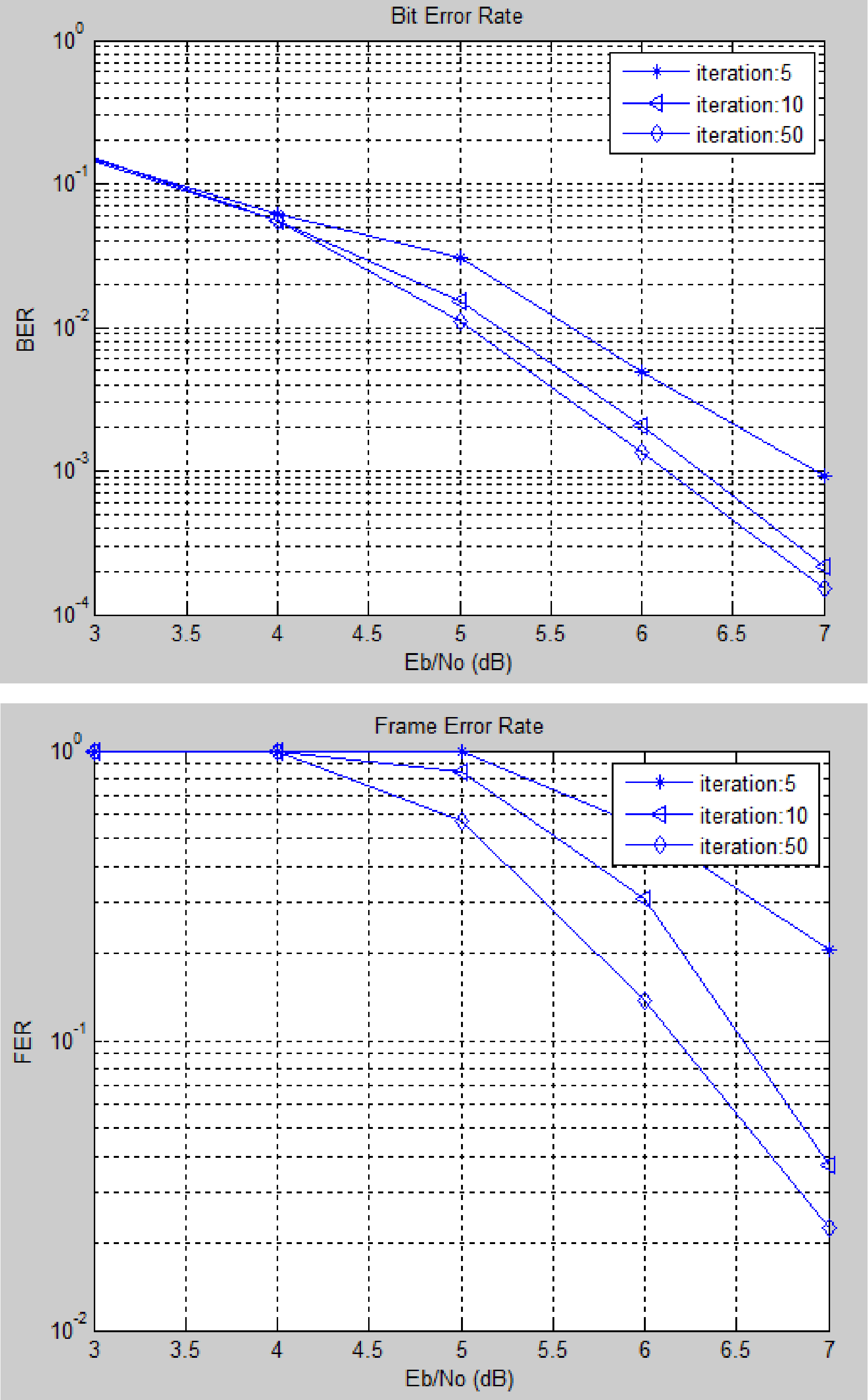
3. Update the reliability measure of all the received bits of $z^{(k)}$, $R_j^{(k+1)} = R_j^{(k)} + E_j^{(k)}$,for $0 \le j < n$ .store the update reliability measure of the received bits in $z^{(k)}$.

4. $k \leftarrow k+1$.for $0 \le j < n$,make the hard-decision : 1) if $R_j^{(k)} \le 0$, $z_j^{(k)}$ =0. 2) if $R_j^{(k)} > 0$, $z_j^{(k)}$ =1. Form a new received vector $z^{(k)}$ and back to step 1.

Comparing the HRBI-MLGD- and the SRBI-MLGD algorithms, the only difference is the initial assignment of reliability measures to the received bits. As a result, the computational complexity of the HRBI-MLGD-algorithm is the same as that of the SRBI-MLGD-algorithm except that fewer bits may be needed to represent integers.

Let b be the smallest positive integer such that 2γ < 2^b .Since the reliability of each received bit is an integer in the range of [$-\gamma$, $+\gamma$], it can be represented by ***b*** bits in radix-2 form. Therefore, ***b*** units of memory are needed to store the reliability measure of a received bit. This results in a memory of ***b*** units to store the reliability measures of the *n* bits of a received vector. Addition of two integers in the range of [$-\gamma$, $+\gamma$] can be accomplished with a ***b*** bit binary adder using logic gates. **Consequently, the HRBI-MLGD-algorithm for an LDPC code can be**

implemented with combinational logic circuits to compute the m check-sums and n reliability measures of the received bits in each decoding iteration.



Ber & Fer for H with 1008 × 504 under AWGN with BPSK modulation

Catch 60 Error Frames

2.6 Density Evolution

For message-passing decoding on general memoryless channels, the bit to check messages are the log likelihood ratios (LLRs) of the probabilities that a given bit is ‘1’ or ‘0’ . As these LLR values are continuous, the probability that a message is a particular LLR value is described by a probability density function (PDF).

Recall that the LLR of a random variable x is

$$L(x) = \log \frac{P(x = 0)}{P(x = 1)}$$

and so L(x) will be positive if p(x = 0) > p(x = 1) and negative otherwise. Consequently the probability that the corresponding codeword bit is a ‘1’ is the probability that the LLR is negative.

To analyze the evolution of these PDFs in the message-passing decoder we define $p(M_l)$ to be the probability density function for a bit to check message at iteration l and $p(E_l)$ to be the probability density function for a check to bit messages at iteration l and *p(r)* to be the probability density function for the LLR of the received signal.

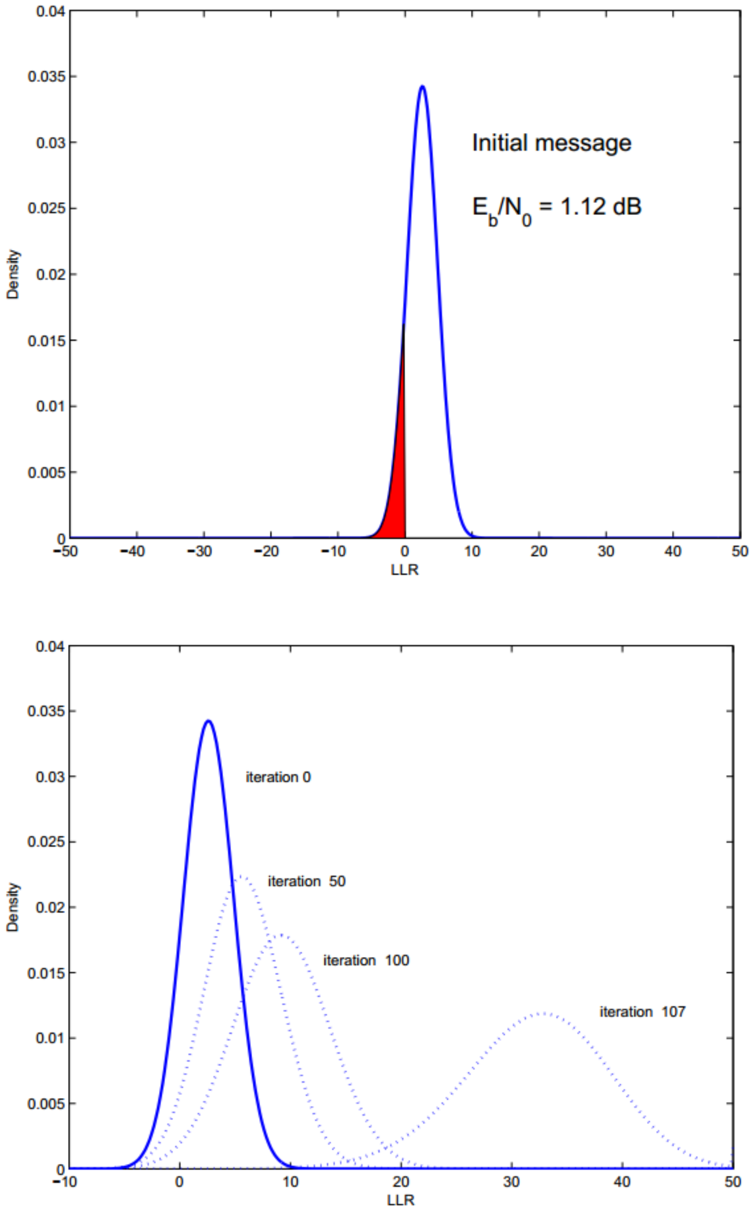
Again we make the assumption that all of the incoming messages are independent of one another. That is, we are assuming firstly that the channel is memoryless, so that none of the original bit probabilities were correlated, and secondly that there are no cycles in the Tanner graphs of length 2l or less, as a cycle will cause the messages to become correlated. The outgoing message at a bit node is the sum of the incoming LLRs on the other edges into that node:

$$M_{j,i} = \sum_{j' \in A_i, j' \neq j} E_{j',i} + r_i$$

Since the incoming messages are independent, the PDF of the random variable formed from this summation can be obtained by the convolution of the PDFs of the $w_c - 1$ incoming messages from the check nodes and the PDF of the incoming message from the channel

To apply density evolution on general channels it is assumed that the original codeword was all zeros. Consequently the probability that the bit is in error is the probability that the LLR is negative.

参考文献：Introducing Low-Density Parity-Check Codes.　Sarah J. Johnson



The evolution of probability density functions with iteration number in density evolution

第三章 LDPC 码在闪存信道的应用

3.1 NAND flash channel model

There are many noise sources existing in NAND flash, such as cell-to-cell interference, random-telegraph noise, background-pattern noise, read/program disturb, charge leakage and trapping generation, etc. It would be of great help to have a NAND flash channel model that emulates the process of operations on flash as well as influence of various program/erase (PE) cycling and retention period.

3.2 NAND flash memory erase and program operation model

A. Erased State.

Before a flash memory cell is programmed, it must be erased, i.e., remove all the charges from the floating gate to set its threshold voltage to the lowest voltage window. It is well known that the threshold voltage of erased memory cells tends to have a wide Gaussian-like distribution. Hence, we can approximately model the threshold voltage distribution of **erased state** as

$$p_e(x) = \frac{1}{\sigma_e\sqrt{2\pi}} e^{-\frac{(x-\mu_e)^2}{2\sigma_e^2}}$$

B. Programmed.

When memory cells are programmed, a tight threshold voltage control is typically realized by using incremental-step-pulse program, i.e., all the memory cells on the same word line are recursively programmed using a program-and-verify approach with a staircase program word-line voltage V_{pp} . Let ΔV_{pp} denote the incremental program step voltage. Under such a program-and-verify programming strategy, each programmed state (except the erased state) associates with a verify voltage that is used in the verify operations. Denote the verify voltage of the kth programmed state as V_p . During each program-and-verify cycle,the floating-gate threshold voltage V_t is first boosted by up to ΔV_{pp} and then compared with the corresponding verify voltage. If memory-cell threshold voltage is still lower than the verify voltage, the program-and-verify iteration continues; otherwise, the corresponding bit line is configured so that further programming of this cell is disabled. Therefore, the threshold voltage of the ***k***th programmed state tends to have a **uniform distribution(均匀分布)** over $[V_p, V_p + \Delta V_{pp}]$ with the width of ΔV_{pp}. **Denote** V_p **and** V_p **+** ΔV_{pp} **for the *k*th programmed state as** $V_l^{(k)}$ **and** $V_r^{(k)}$. We can model the ideal threshold-voltage distribution of the ***k*th** programmed state as

$$p_p^{(k)}(x)=\begin{cases}\dfrac{1}{\Delta V_{pp}}, & ifV_l^{(k)}\le x\le V_r^{(k)}\\ 0, & else.\end{cases}$$

We note that certain device and circuit noises, particularly random telegraph noise , may introduce exponentially decreasing tails at both sides of the uniform distribution. As well discussed in the open literature, cell-to-cell interference may most severely distort the threshold-voltage distribution of programmed cells.

C. Cell-to-Cell Interference

The threshold-voltage shift of a victim cell caused by cell-to-cell interference from neighbor interfering cells which are programmed after this victim cell can be estimated as:

$$F=\sum_n(\Delta V_t^{(n)}\cdot\gamma^{(n)})$$

where $\Delta V_t^{(n)}$ represents the threshold-voltage shift of one interfering cell which is programmed after the victim cell and the $\gamma^{(n)}$ is coupling ratio .

For the cell-to-cell interference modeling in this work, we assume that both the vertical coupling ratio γ_y and diagonal coupling ratio γ_{xy} are random variables with bounded Gaussian distributions

$$p_r(x)=\begin{cases}\dfrac{c_\gamma}{\sigma_\gamma\sqrt{2\pi}}\cdot e^{-\frac{(x-\mu_\gamma)^2}{2\sigma_\gamma^2}}, & if\ |x-\mu_\gamma|\le\omega_\gamma\\ 0, & else\end{cases}$$

where μ_γ and σ_γ are the mean and standard deviation, ω_γ represents the bounded distribution region, and c_γ is normalization coefficient（归一化系数）, According to relative paper ,we set the ratio between the means of γ_y and γ_{xy} as 0.08:0.006, i.e., **vertical cell-to-cell interference with γ_y tends to play a dominating role.** To study a wide range of cell-to-cell coupling strength, we introduce a parameter **s called cell-to-cell coupling strength factor** and have the mean of equal 0.08 s and the mean of equal 0.006 s.

3.3 Mathematical Formulation of LLR Calculation

Let V_{th} represent the sensed threshold voltage of one memory cell, and we simply assume that each bit in a cell has apriori probability of 0.5 being 0 or 1, i.e., all the storage states in one memory cell have equal apriori probability. Therefore, the LLR of the ith bit stored in one cell is

$$L(b_i) = \log \frac{p(b_i = 1 | V_{th})}{p(b_i = 0 | V_{th})} = \log \frac{p(V_{th} | b_i = 1)}{p(V_{th} | b_i = 0)}$$

Let N_b represent the number of bits stored in each NAND Flash memory cell; hence, there are $K = 2^{N_b}$ storage states. Let $p^{(k)}(x)$ denote the probability density function of the threshold voltage of the ***k***th storage state, where $0 \le k \le K-1$ and the state 0 corresponds to the erased state and state K-1 corresponds to the state with the highest threshold voltage. Let S_i denote the set of states whose ***i***th bit is 1. Hence, given the threshold voltage V_{th} of a cell, we can calculate the LLR of each bit as

$$L(b_i\ ) = \log \frac{\sum_{k \in S_i} p^{(k)}(V_{th})}{\sum_{k} p^{(k)}(V_{th}) - \sum_{k \in S_i} p^{(k)}(V_{th})}$$

Clearly, LLR calculation demands the knowledge of the probability density functions of all the K states. Based on the discussions in Section II, LLR calculation is trivial if cell-to-cell interference has not occurred yet. Hence, we only consider LLR calculations for cells in the presence of cell-to-cell interference. In this context, we should first know the probability density function of cell-to-cell interference, denoted as $p_c(x)$. Suppose a victim cell suffers cell-to-cell interference from N interfering cells. Let x_n and y_n denote the threshold voltage of the nth interfering cell before and after being programmed, and let γ_n denote the coupling ratio between the nth interfering cell and the victim cell. The overall cell-to-cell interference can be expressed as

$$F = \sum_{n=0}^{N-1} \gamma_n (y_n - x_n) = \sum_{n=0}^{N-1} \gamma_n y_n - \sum_{n=0}^{N-1} \gamma_n x_n$$

As discussed earlier, the threshold voltage of the erased state has a Gaussian distribution. Hence, all the x_n 's are random Gaussian variables, and $x_n' = \sum_{n=0}^{N-1} \gamma_n x_n$ is still a Gaussian variable $N(\mu', \sigma')$, where

$$\mu' = \sum_{n=0}^{N-1} \gamma_n \mu_e \ \ , \ \ \sigma' = \left(\sum_{n=0}^{N-1} \gamma_n^2 \sigma_e^2 \right)^{1/2}$$

Let $p_{x'}(x)$ denote the probability density function of the Gaussian variable x' . After a cell is programmed, its threshold voltage ideally follows a uniform distribution, as discussed earlier. Hence, each can be simplified as a scaled uniform distribution. Let $y' = \sum_{n=0}^{N-1} \gamma_n y_n$, and assuming that each γ_n is a constant, we can obtain its probability density function using the results presented in paper that derives a closed form for the probability density function $p_{y'}(y)$ of the sum of u independent non-identically distributed uniform random variables. Hence, the probability density function of the overall cell-to-cell interference strength $F = y' - x'$ can be estimated as

$$p_c(x) = \int_t p_{y'}(x+t) p_{x'}(t) dt$$

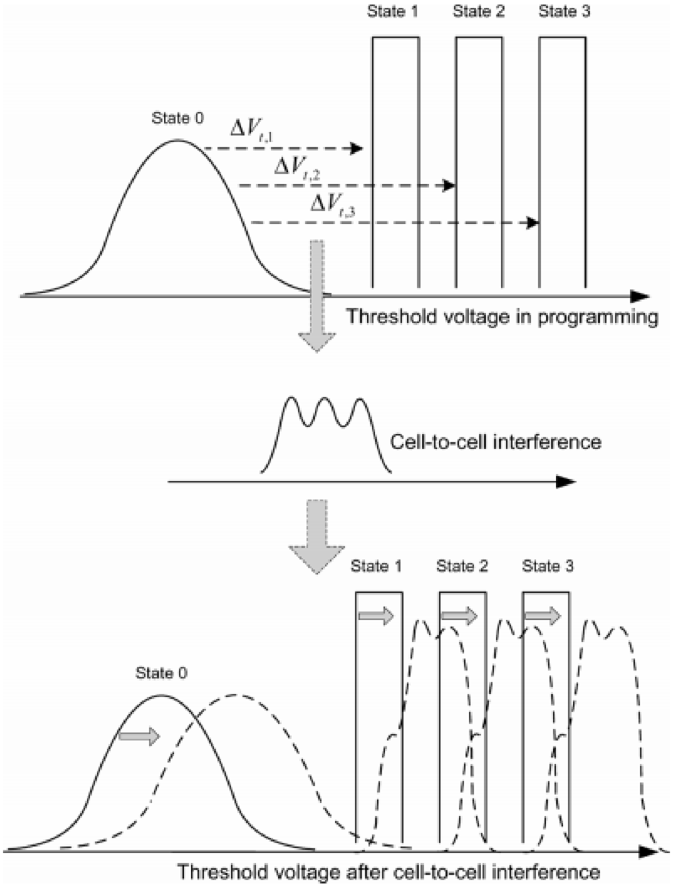
After obtaining the cell-to-cell interference strength, the last step is to estimate the distribution of victim-cell threshold-voltage shift induced by cell-to-cell interference. If the victim cell stays in the erased state, its threshold-voltage distribution in the presence of cell-to-cell interference can be expressed as

$$p_0(x) = \int_t p_e(t) p_c(x-t) dt$$

If the victim cell is programmed to the *k*th programmed state, its threshold-voltage distribution in the presence of cell-to-cell interference can be expressed as

$$p^{(k)}(x) = \int_t p_p^{(k)}(t) p_c(x-t) dt$$

In the following, we further elaborate on the scenario when the all-bit-line structure is being used. In this context, as discussed previously, one victim cell has three interfering cells, one along the vertical direction with γ_y and two along the two diagonal directions with γ_{xy} . According to other documents, γ_{xy} **is one order of magnitude less than** γ_y **. Hence, to simplify the following mathematical derivations, we ignore the cell-to-cell interference from the two diagonal directions and also simply fix** γ_y **as a constant.** We note that **we still take into account of the cell-to-cell interference from the two diagonal directions and model the coupling ratios as bounded Gaussian variables for all the computer simulations** presented throughout this file.


State 1
State 2
State 3
State 0
$\Delta V_{t,1}$
$\Delta V_{t,2}$
$\Delta V_{t,3}$
Threshold voltage in programming
Cell-to-cell interference
State 1
State 2
State 3
State 0
Threshold voltage after cell-to-cell interference

Let $\Delta V_{t,k}$ represent the threshold-voltage shift when one interfering cell is being programmed to the kth state, and let $p_{\Delta}^{(k)}(x)$ represent the probability density function of $\Delta V_{t,k}$ .As shown in Fig, in the following, we first derive $p_{\Delta}^{(k)}(x)$ and $p_c(x)$ then , based on which we derive $p^{(k)}(x)$ for calculating LLR .

When the interfering cell is being programmed to the state (where $0 \le k \le K-1$), the probability density function of its threshold-voltage shift can be obtained as

$$
\begin{aligned}
p_{\Delta}^{(k)}(x) &= \int_t p_e(t-x)\, p_p^{(k)}(t)\,dt \\
&= \int_{V_l^{(k)}}^{V_l^{(k)}} \frac{1}{\Delta V_{pp}} \frac{1}{\sigma_e\sqrt{2\pi}} e^{-\frac{(t-x-\mu_e)^2}{2\sigma_e^2}}\,dt \\
&= \frac{1}{2\Delta V_{pp}}\left(erf\left(\frac{V_r^{(k)}-x-\mu_e}{\sqrt{2}\sigma_e}\right) - erf\left(\frac{V_l^{(k)}-x-\mu_e}{\sqrt{2}\sigma_e}\right)\right)
\end{aligned}
$$

Where

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_{o}^{x} e^{-t^2} dt$$

is the error function. Given the probability density function $p_{\Delta}^{(k)}(x)$ of threshold voltage shift of one interfering cell and the corresponding coupling ratio γ_y , the cell-to-cell interference experienced by the victim cell can be obtained as

$$p_c^{(k)}(x) = \frac{p_{\Delta}^{(k)}(x/\gamma_y)}{\gamma_y} .$$

If a cell should stay in the erased state, it will not experience threshold voltage shift from programming operation, and thus will not induce cell-to-cell interference to its neighbors, therefore the corresponding interference can be modeled as

$$p_c^{(0)}(x) = \delta(x)$$

Where $\delta(x)$ represents the Dirac delta function.

Assuming that the interfering cells have the same probability to be programmed to each state (i.e., each state has a probability of 1/K), the overall distribution of cell-to-cell interference can be modeled as

$$p_c^{(x)} = \frac{1}{2K\gamma_y \Delta V_{pp}} \times \sum_{v=1}^{K-1} \left(erf\left(\frac{V_r^{(v)} - x/\gamma_y - \mu_e}{\sqrt{2}\sigma_e} \right) - erf\left(\frac{V_l^{(v)} - x/\gamma_y - \mu_e}{\sqrt{2}\sigma_e} \right) \right) + \frac{\delta(x)}{K}$$

We can then estimate the overall victim-cell threshold-voltage distribution after the occurrence of cell-to-cell interference. If the victim cell is in the erased state, its threshold-voltage distribution after the occurrence of cell-to-cell interference can be modeled as

$$\begin{aligned} p^{(0)}(x) &= \int_t p_e(t) p_c(x-t) dt \\ &= \frac{1}{C} \int_t e^{-\frac{(t-\mu_e)^2}{2\sigma_e^2}} \times \left(\sum_{v=1}^{K-1} \left(erf\left(\frac{V_r^{(v)} - \frac{x-t}{\gamma_y} - \mu_e}{\sqrt{2}\sigma_e} \right) - erf\left(\frac{V_l^{(v)} - \frac{x-t}{\gamma_y} - \mu_e}{\sqrt{2}\sigma_e} \right) \right) + 2\gamma_y \Delta V_{pp} \delta(x-t) \right) \end{aligned}$$

Where

$$C = 2\sqrt{2\pi}\sigma_e \gamma_y K \Delta V_{pp}$$

If the victim cell is in the *k*th programmed state, its threshold-voltage distribution after the occurrence of cell-to-cell interference can be modeled as

$$p^{(k)}(x) = \int_t p_p(t) p_c(x-t) dt$$

$$= \int_{V_r^{(k)}}^{V_r^{(k)}} \Bigg(\frac{1}{2K\Delta V_{pp}^2} \times \Bigg(\sum_{v=1}^{K-1} \Bigg(erf\Bigg(\frac{V_r^{(v)} - \frac{x-t}{\gamma_y} - \mu_e}{\sqrt{2}\sigma_e} \Bigg) - erf\Bigg(\frac{V_l^{(v)} - \frac{x-t}{\gamma_y} - \mu_e}{\sqrt{2}\sigma_e} \Bigg)\Bigg)\Bigg) + 2\gamma_y \Delta V_{pp} \delta(x-t) \Bigg)\Bigg)$$

Ideally, given the previously obtained distribution functions and the sensed threshold voltage V_{th} of memory cells in the presence of cell-to-cell interference, we can calculate the corresponding LLR. In practice, threshold-voltage sensing is realized by the comparison with a series of reference voltages. Assume that the threshold voltage V_{th} falls into the range $(R_l, R_r]$ (**where** R_l **and** R_r **are two adjacent reference voltages**)（区别于$V_l^{(k)}, V_r^{(k)}$ 和）,we can estimate the corresponding LLR of the *i*th bit as

$$L(b_i) = \log \frac{\int_{R_l}^{R_r} \sum_{k \in S_i} p^{(k)}(x) dx}{\int_{R_l}^{R_r} \sum_{k} p^{(k)}(x) dx - \int_{R_l}^{R_r} \sum_{k \in S_i} p^{(k)}(x) dx}$$

In the aforementioned mathematical formulation, we assume that the erased-state distribution parameters μ_e and σ_e are known and we treat the coupling ratio γ_y as a known constant. As pointed out earlier, γ_y is essentially a random variable; hence, we should use the average value of γ_y in practice.