Solid-State-Drives and their underlying memory architecture

Luis Michaelis Philip Laskowicz Malte Lohmann

May 2, 2019

Contents

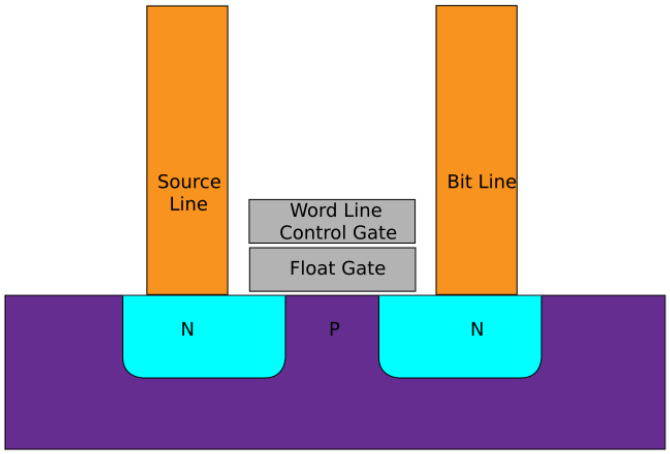
1 Prerequisites

Most consumer Solid-State-Drives (*SSD*s) are using flash memory to store data. This is the same kind of memory used in SD-Cards and USB-Sticks. They work using a special kind of transistor which traps electrons inside a conductor to store a bit. Flash cells are chained together to form memory pages usually in the range of 4KiB to 16KiB. These are then put together to create a memory block usually 1024KiB to 4096KiB in size. The inner workings of these chained cells will be explained in section 3.

2 Flash Memory Cell

A flash memory cell as shown below resembles a standard *metal-oxide-semiconductor field-effect transistor* (MOSFET) with an additional gate. The normal MOSFET-gate is then called *Control Gate* and the additional gate is called the *Floating Gate*. It is insulated from all sides to prevent charges from easily entering or exiting the conductor.

Figure 1: A flash memory cell.[1]



If there are no electrons on the floating gate the transistor behaves normally, enabling a current from source to drain if the control gate has a positive charge and therefore creates a positive electrical field (this happens when applying a voltage U_0). If however electrons are on the floating gate they will cause a

[1] `https://commons.wikimedia.org/wiki/File:Flash_cell_structure.svg`

negative electrical field to build up. This negative field will screen the positive electrical field created by the control gate. Therefore the transistor will not switch "on" when applying the same positive electrical charge to the control gate as before. It will need a higher positive charge to enable current-flow (let's say you need a voltage U_1). So to store data on the cell we need to somehow get electrons onto the floating gate. This is done using either of two methods, *Hot Electron Injection* or *Fowler-Nordheim Tunneling*. For the first method, the so-called *Hot Electron Injection*, a positive voltage is applied to the control gate and a high negative voltage is applied between the source and drain of the transistor. Because the electrons are now able to travel through the transistor to drain and they have a high kinetic energy due to the high voltage some of them break through the insulation layer at the bottom of the floating-gate and end up trapped inside it. For the second method, so-called *Fowler-Nordheim tunneling*, a high negative voltage is applied to the control gate, causing some electrons on it to quantum-tunnel onto the floating gate. To read the cell a negative voltage U_R is applied to the source-line and U_0 is applied to the control gate. If the transistor switches (according to the rules above) we can assume no electrons on the floating gate so no writing has taken place therefore the cell contains a logical *0*. If it does not switch we have to have written data to it so we assume a logical *1*. The reasoning behind that naming scheme will be discussed later.

To reuse a cell we will also need to erase the data from it. This can only be done using quantum tunneling: A high positive voltage is applied to the control gate which causes electrons stored on the floating gate to quantum tunnel out of it onto the control gate.

3 Page arrangement

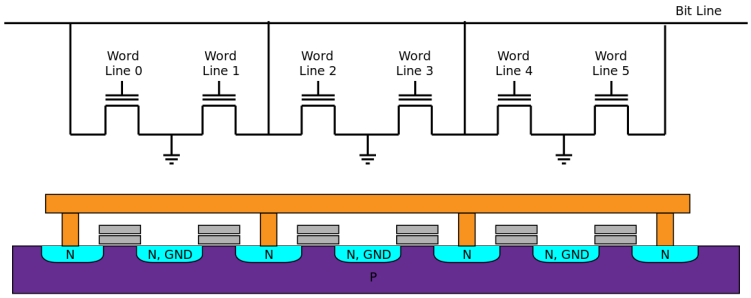
There are multiple ways of creating memory pages. Approximately 4.000 to 16.000 flash cells are chained together using either of two methods: *NOR flash* or *NAND flash*. Due to power management limitations erasing (resetting the transistor to it's default state 0) *NAND* or *NOR* flash is only possible in blocks (1.024.000 to 4.096.000 cells or 64 - 1024 pages depending on their size).

3.1 NOR

NOR flash consists of memory cells with the drain line tied to ground. The bit line which will give the final bit is tied to high by default. To read a bit, the positive voltage U_0 is applied to the word line of the corresponding cell. If the cell has been written (contains a logical 0) the transistor will conduct and tie the bit line to ground. In this case the controller would read a 0 from the bit line. A one will be read if the transistor doesn't switch. This is called *NOR* flash because if just one of the transistors switches, the whole bit line will be tied to ground and thus a logical 0 will be read. *NOR* may be written using any of the techniques described above. It has the advantage of faster read / write access

[2] `https://en.wikipedia.org/wiki/Flash_memory#/media/File:NOR_flash_layout.svg`

Figure 2: NOR flash.[2]



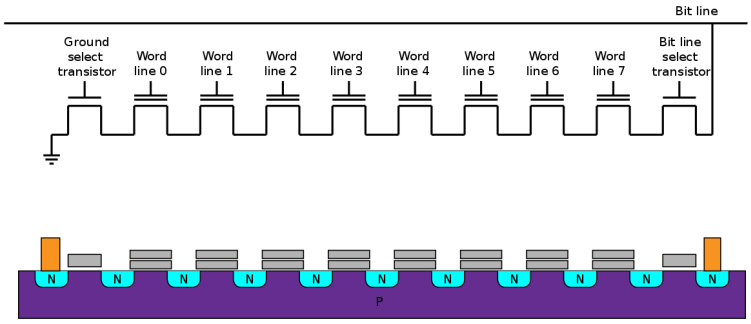
when compared to *NAND* but requires more space due to the many ground connections between the transistors.

3.2 NAND

A *NAND* flash page consists of flash memory cells where the first cells source line is connected to the bit line and the last cells drain line is connected to ground. The source line of each intermediate cell is connected to the drain line of the previous cell. This chain can be engaged or disengaged using a bit line select transistor before the first memory cell. This transistor is a regular MOSFET in most cases. The bit line which will contain the final bit is tied to high by default. To read a bit, the positive voltage U_1 is applied to all cells except the one to be read which will cause them to conduct regardless of what value is stored. The word line of the cell to be read is applied the intermediate voltage U_0. If the cell has been written (contains a logical 0) the cell will conduct and tie the bit line to ground which will indicate a logical 0 to the controller. If it doesn't conduct the controller will see a positive voltage on the bit line and read a logical 1. This is called *NAND* flash because only of all of the cells conduct, a logical 0 will be read. *NAND* may only be written using *Fowler-Nordheim tunneling* because of the nature of *NAND*. *NAND* flash has the advantage of

[3] `https://en.wikipedia.org/wiki/Flash_memory#/media/File:Nand_flash_structure.svg`

Figure 3: NAND flash.[3]



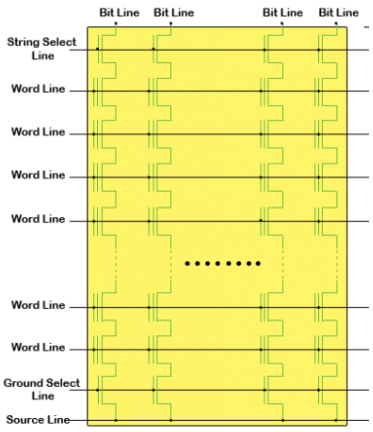
being more compact than *NOR* flash and thus have a lower cost per bit. The read speeds of *NAND* are slower than those of *NOR* flash.

3.3 Block arrangement

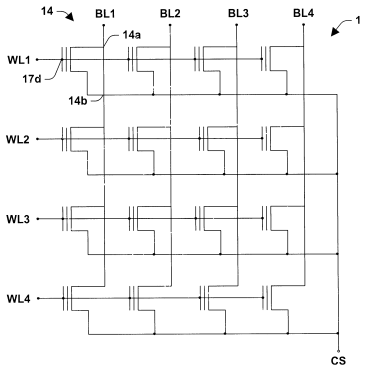
NAND and NOR flash pages are arranged into memory blocks as shown below.

Figure 4: NAND and NOR block arrangements

(a) NAND block[4]



(b) NOR block[5]



The word lines of the corresponding cell in each page are connected so that the same transistor can be accessed on each page. The bit lines may be handled in several ways but the figures show a setup in which two neighboring bit lines are fed into an amplifier to boost the retrieved bit line output. This arrangement means that only one of the two bit lines may be read at any given time because the other one is needed to provide the amplifier with power. The amplifier is especially needed when using *MLC* technology.

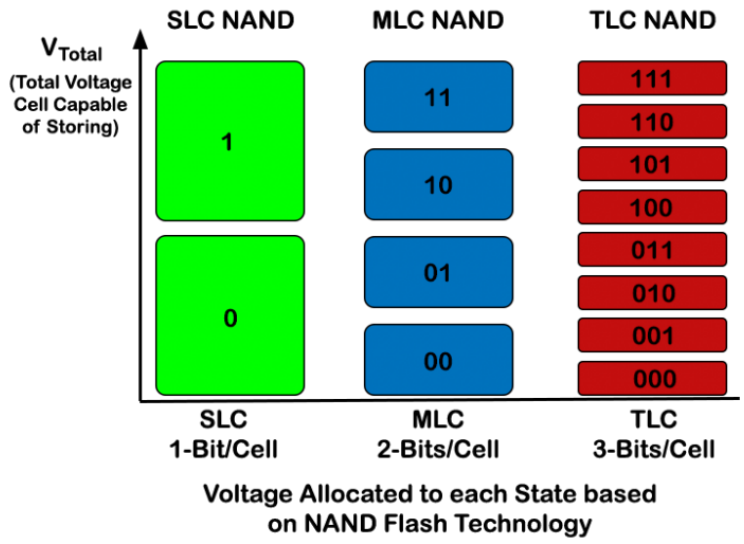
4 Data Storage

In modern flash-chips it is possible to store more than one bit in one cell, because modern chips can have precise voltage control when reading or writing to a flash cell. Because of this, chips are categorized by how many bits they can store per cell.

[5] `https://www.cactus-tech.com/resources/blog/details/solid-state-drive-primer-5-nand-architecture-planes-and-die`

[5] `https://patents.google.com/patent/US6449188`

Figure 5: Voltage Levels for different flash architectures.[6]



4.1 SLC

In an Single Layer Chip only one bit per cell is stored. This has the advantage that the signal is already in a usable form and does not need to be transformed, which makes this type very fast. *SLC*s follow the technique described in section 2 to read and write data. You can see in Figure 5 that there are only two voltage levels that the cell can store.

4.2 MLC

Multi Layer Chips store two bits per cell. To make this possible the voltage is separated into four layers as can be seen in Figure 5. Every layer (or voltage level) is than given an 2 bit code: *00*, *01*, *10*, or *11*. To read from multi-layer flash chips, different voltages have to be applied, one for each state that the cell can be in (see why in section 2) to get the correct bit. Alternatively a voltage that will make the cell conduct regardless may be applied, in which case the controller would then have to figure out which bit is one the cell by reading the outputted voltage level. This makes *MLC*s slower than *SLC*s.

[6] `https://www.cactus-tech.com/resources/blog/details/solid-state-drive-primer-2-slc-mlc-and-tlc-nand-flash`

4.3 TLC

Triple Layer Cells work the same way as *MLC*s. The only difference is that *TLC* use eight layers which makes it possible to store three bits in one cell. This makes it even slower than *MLC*.

4.4 Implications

MLC and *TLC* have reduced write cycles because of tighter voltage levels which have to stay very precise. Such a system is very difficult to maintain because electrons may somehow end up on the floating gate unintendedly. This can destroy the stored data. Electrons have to be written precisely too such a system is very vulnerable to outside influences. On the other hand *MLC* and *TLC SSD*s cost less money per byte because of the use of half or even a third of flash cells than a *SLC* would use.

5 Controller

The Controller of an *SSD* has many different functions. It converts the stored data into data usable by the computer by providing a digital interface for the CPU to access. With Bad Block Mapping it protects data store on the *SSD* from being corrupted by no longer storing data on broken blocks. This needs to happen because the flash cells are only able to sustain a limited number of write/erase cycles. For *SLC NAND* flash it would be around 100.000 erases per block, *MLC NAND* can endure around 5.000 to 10.000 erases per block and *TLC NAND* can be erased around 1.000 times. For *NOR* flash the numbers are different: up to 1.000.000 erases for *SLC* and around 100.000 for *MLC*[7]. Read and Write Caching accelerates the *SSD* by storing frequently used (so called "hot-data") in a faster cache (often SDRAM). Data that has to be written is being temporarily stored in the cache too. By storing the data with 22 bits of associated data, the controller can detect up to two failed bits and can repair one failed bit using Error-Correcting-Code (ECC). To prevent broken data the controller uses ECC in a process called *Read Scrubbing*. It repeatedly checks the data on the *SSD* in this process and corrects failed bits. Unused data is deleted using a process known as *Garbage Collection*. Using this process the controller will free blocks by copying used pages from these blocks to another block so that it can then erase the whole, now unused block and make it available to be rewritten. Because of the limited write/erase cycles the controller needs to move hot-data into blocks with more cycles left to prevent single blocks to wear off too quickly. This is called *Wear-Leveling*.

[7] `https://en.wikipedia.org/wiki/Flash_memory#Write_endurance`