Flash Memory Software Design Concept

Flash is a non-volatile storage that can be programmed and erased by pushing and polling the electrons from floating gates. The principle comes from the early generation memory: Electrically-Erasable Programmable Read-Only Memory (EEPROM). Flash memory is distinguished from EEPROM by larger erase unit and better erase performance. That’s the reason why we call it “flash”.

Flash memory is extensively used in electronic devices. It is the most popular storage except hard disk. There is no magnet disk, spindle motor and disc read-write device (arm actuator + head) in Flash, so it can be smaller, shake proof, lower power consumption and higher random access performance. However, higher cost, lower data reliability and more complex software design-in are the disadvantages that people are working for.

Macronix International Co. LTD. designs and produces flash memories since 1989. Now it is the leading company of “Read-Only Memory” (ROM) and serial flash manufacture. Macronix starts to produce NAND flash from 2011, and continually focus on new technology of flash memories, such as advanced manufacture technique “Bandgap Engineered SONOS” (BE-SONOS), 3 Dimensions Vertical Gate techniques (3DVG), “Phase Change Memory” (PCM) and “Resistive Random Access Memory” (ReRAM).

We will introduce the design issues of flash memory in this document. You could find more detailed information on our web. Welcome to contact with us if you have any problem or suggestion: flash_model@mxic.com.tw.

Flash Characteristics

Generally, there are three main characteristics related to flash software design:

- **Small program unit but large erase range**
  The cell bit of flash memory will be changed from 1 to 0 after pushing electrons. This behavior is called “Program”. Program that can only change bits from 1 to 0, so we need to clear data from 0 to 1 by “Erase” command for next programming to the same address again. However, flash memory has smaller program unit than erase unit. For example, it may have 2
Kbytes program unit, and 128 Kbytes erase unit in a NAND flash. So how to coordinate the behavior of program to reduce the side effect of erase is the first issue that flash software designers must concern.

- **Limited lifetime of flash result from limited erase counts of blocks**
  Erase is a necessary behavior of flash memory, but it is destructive. The erase unit of flash is a “Sector” or a “Block”. Usually a block can only sustain 10,000~100,000 erase times before it is wore out. A broken block can not be programmed or erased anymore, so we call it “Bad block”. A flash memory cannot be used if there are too much bad blocks in it. So we need special mechanisms of decreasing erase times and handling bad blocks to extend lifetime of flash memories.

- **Data reliability**
  Some kinds of flash memories may lose electrons easily after storing data and thus causes the data error. So we need to use special detecting or correcting mechanisms to resolve it. This problem occurs with different variation according to different architectures or component characteristics of flash memories. Till now, there is no concern on some memories, such as “Macronix’s NOR flash” for this problem.

The above flash characteristics make designers focus on data type accessing and data arrangement. Most of solutions are achieved by software algorithms which have been discussed for many years. Macronix had researched in flash software and firmware design since 2009 and we'll have more achievements in recent years. You can get some fundamental information about software design in chapter “Software Design Issue”.

**NOR or NAND**

Flash memory can be classified to “NOR” or “NAND” type by their manufacture method. The advantages of NOR flash are faster reading speed, smaller unit of program and higher data reliability. NAND flash has better performance on programming and erasing. So most of NOR flash are applied to “code storage” such as game card or “Basic Input/Output System” (BIOS). NAND flash is used in “data storage”, such as: memory card, USB drive…etc. The demand required for NAND grows as the demand for mobile devices. NAND flash is widely used, even includes “Embedded Multi Media Card” (eMMC) and “Solid State Drive”
(SSD), which can replace the HDD of Server or Notebook, or for the application of smart phone and tablet PC.

【Table 1】 represents the comparison of Serial NOR and NAND flash (Note 1). Macronix produces two kinds of NOR flash: Serial and Parallel. Serial NOR flash is more popular in mobile devices, because of its smaller packaged size with low pin count. The table describes the performance of flash in fast read mode (We don’t use QPI mode here because it needs hardware support). You can find the datasheet on Macronix web at: http://www.macronix.com.tw/.

<table>
<thead>
<tr>
<th></th>
<th>NOR</th>
<th>NAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>MX25L12835E</td>
<td>MX30LF1G08AA</td>
</tr>
<tr>
<td>Programming unit</td>
<td>1 Byte</td>
<td>2 KByte</td>
</tr>
<tr>
<td>Read performance</td>
<td>About 200 MB / s</td>
<td>About 25 MB / s</td>
</tr>
<tr>
<td>Write performance</td>
<td>About 0.2 MB / s</td>
<td>About 8 MB / s</td>
</tr>
<tr>
<td>Erase delay</td>
<td>600 ms</td>
<td>2 ms</td>
</tr>
<tr>
<td>Error correcting</td>
<td>No need</td>
<td>Need</td>
</tr>
</tbody>
</table>

(Note1) The performance of flash is estimated from datasheet, not standing for real condition

For NAND flash, it is more difficult to deal with fragments of storage if they occur, because basic programming unit of NAND flash is "Page" and it is not the same as "Byte" which is used for NOR flash. Frequent "Page" programming certainly causes fragments of storage, which will shorten the lifetime of NAND flash. Hence, general software design for NAND is more important than for NOR, such as Wear Leveling technique for flash.

For cost saving purpose, the evolution of manufacturing process continues all the time. However, the more process evolution proceeds, the more problems happen on new process, due to the physical phenomenon that the quantity of electrons stored in memory cells is decreased, which causes lower reliability of data storage. Besides, for the tolerance and performance of NAND flash, lower cost "Multi Level Cell (MLC, 2 bits in one cell)" NAND or even "Triple Level Cell (TLC, 3 bits in one cell)" NAND are getting worse than "Single Level Cell" (SLC, 1 bit in one cell) NAND. Therefore, applying complex design such as "Error Checking and Correcting (ECC)" is required for better tolerance and performance. All of these concerns above result in more challenges for software (firmware) design in data programming and storage.
Software Design Issue

Because of the flash characteristics introduced in the first chapter, we need some special access methods for flash, such as additional flash controller with firmware codes or a special file system. This is the reason why we need different flash management algorithms. The most important issues are “Address Translation”, “Wear Leveling”, “Error Checking and Correcting” and “Power Cycling Management”.

- Address Translation
  We need to erase old data before we program the new one to flash, but it costs lots of time on erasing before programming to the same place every time. So we usually program data to other place which is cleared. The addresses of that place needs to be recorded. That’s the problem called “Address Translation” or “Data Rearrangement”. The unit of recorded addresses could be a page, double page or a block. According to the unit of recorded addresses, we simply classify the translation algorithms as “Fine-Grained Mapping” or “Coarse-Grained Mapping”.

  Fine-grained mapping has better performance, because of its precise data arrangement. But it also costs more RAM space to setup “Mapping Table” to record data addresses. The more addresses data we write to mapping table, the less RAM space is left, and that means that we may need to write back the information. We write back the data to a table which is called “Swapping Table”. So how to decide the size of the mapping table and the proper time of memory swapping are what we need to consider.

  When the data is mapped to a new address, the old address of the data is marked as “Invalid” and avoided to be used. After repeating above actions continually, the flash will be filled with invalid data and no more usable space available. Then we need to erase blocks to free space. However, there may have some “Valid” data in blocks such that we need to move them to other free place. We call this behavior “Garbage Collection” which will cause another address translation.

  The above complicated mechanisms are considered with integrating performance, RAM consumption and space usage. Therefore many
schemes are designed for different purposes. “Log-based” or “Hybrid” address mapping is used in devices which concern with small RAM consumption such as memory cards or USB drives. Fine-grained mapping method is adopted by eMMC, high class SSD, or host software solution. The related algorithms have been discussed for years. You could refer to related themes such as KAST and LazyFTL.

- **Wear Leveling**
  Wear leveling is used to extend flash’s lifetime, and avoid bad block occurrence. Its main ideas are “reducing erase times” and “averaging each block’s erase times”. These two things are always related to garbage collection. For achieving wear leveling, we may need to record the erase times of each block, and classify the data according to different access behavior. So that we can put the data to the proper place. We’ll also need to handle bad blocks by “Bad Block Management”.

  It costs additional RAM and performance to do wear leveling, so how to cooperate wear leveling with other algorithms such as address translation or garbage collection is important. Macronix products such as NOR, SLC NAND flash all have high erasing endurance, so we are able to use low cost wear leveling algorithm to extend flash lifetime without too much performance or RAM cost.

- **Error Checking and Correcting**
  For solving data reliability problem, must of systems that support “Error Correction Code” (ECC). It uses mathematic methods to add some special codes in the end of data. When data error occurs, the system corrects the data by the codes. Generally speaking, “Hamming Code” is already built in the operating system or hardware controller. It can support single bit error correction. Some of file system use multi bits correction such as BCH. These two methods can plentifully support low density flash’s error occur rate. Macronix now researches on “Low Density Parity Check” (LDPC) which is an advanced ECC scheme that may be used in the future.

- **Power Cycling Management**
  The abrupt power lost will break off programming or erasing action of flash and result in data error. The high security system such as cloud server had established “Uninterruptible Power Supply” (UPS) to avoid power lost
situation. However, UPS is not suitable in portable flash, so the software method is needed. The designer need to consider all conditions of data access and make sure the data has been saved. The timing of system log or metadata programming is especially important. By this way, the lost data can be recovered as much as possible.

Generally, complicated flash algorithms costs more RAM. Besides, different combination of algorithms may cause different results. It is important to notice that there is no best design. There is just most suitable design which depends on your application. There are many other advanced flash design issues, such as “Buffer Cache” to reduce accessed response time, “Data Compression” to save flash space and “Multi Channel” in SSD to increase performance. You may be interested in one of them in the future.