MLC vs. SLC NAND Flash in Embedded Systems

Yingbo Hu and David Moore
Micro Digital Inc

There are two primary types of NAND flash technology: Single-level cell and multi-level cell. Multi-level cell was developed more recently, to achieve higher bit density, so that a much higher capacity flash chip could be created for a given die size. MLC might allow you to save cost for flash chips and save board space by reducing the number of chips you need. But before you rush to design MLC into your embedded system, there are a few things you should know, and other options you can consider.

Although MLC beats SLC on density and cost per bit, SLC has the advantage for reliability, performance, power consumption, and operating temperature range. Normally MLC flash is intended for consumer products but SLC should be used for industrial products. The following table summarizes these and other parameters.

Features	**SLC**	**MLC**
Bits per cell	1	2 or more
Voltage	3.3v, 1.8v	3.3v
Data bus width (bits)	x8, x16	x8

Architecture	**SLC**	**MLC**
Planes	1 or 2	2
Page size (bytes)	2112	2112-4314
Pages per block	64	128

Reliability	**SLC**	**MLC**
ECC (per 512 bytes)	1-bit	4-bit or more
Erase/Program cycles	100,000	10,000
Partial programming times	4	—

Performance	**SLC**	**MLC**
Read	25 us	50 us
Program	200-300 us	600-900 us
Erase	1.5-2 ms	3 ms

The reliability and performance characteristics are most significant to a flash file system. In particular, consider these MLC limitations vs. SLC:

1. The error rate for MLC flash is very high compared to SLC, so at least 4-bit ECC is required vs. 1-bit for SLC. This is too slow in software, so you need an MLC NAND controller with built-in 4-bit ECC controller. A small number of SoCs have such a controller built in. Otherwise, you need to add an external controller to your design or

purchase the IP to include in your ASIC or FPGA. The following are a few ARM processors that have built-in ECC sufficient to support MLC:

- NXP LPC3180/3250 processor has built-in MLC controller and Reed-Solomon ECC engine.
- TI Davinci DM355 has built-in ECC engine for 1-bit and 4-bit ECC
- TI OMAP 35xx processor has built-in ECC engine for 1-bit (Hamming) and 4-bit (BCH)

When ECC is calculated in hardware, performance is only minimally reduced. In a test we did, write performance was reduced by only about 1.3%. By contrast, using software to calculate ECC in our flash file system using software reduced performance to a crawl:

Without ECC	600 KB/s
1-bit ECC	300 KB/s
4-bit ECC	**5 KB/s** (in software)

Clearly 4-bit ECC in software is unacceptable. Even for a 512 MHz ARM11 we achieved only 30 KB/s. There are several algorithms for calculating 4-bit (or more) ECC. BCH (Bose, Ray-Chaudhuri, Hocquenghem) is popular because of its improved efficiency over Reed-Solomon. However, even BCH needs too many microprocessor cycles. A 256 KB flash block has 256*1024*8 = 2 Mbit. The ECC calculations (done for each 256 bytes) need 48 loops per bit, and for each bit it executes about 10 instructions. So totally it needs 2M*48*10 (about a billion) instructions to calculate ECC codes for one 256 KB flash block. Even on a 2 GHz Windows PC, it needs about 400-500 milliseconds.

2. MLC supports only 1/10 the number of erase/program cycles of SLC. The flash management software must have a carefully designed cache system that reduces the number of erase/program cycles. Also, the needs of the application must be considered. MLC is not suitable for applications that must do small frequent data write operations.

3. No partial page programming. The flash management software cannot change or append any new data to a page after data has been written to that page. Flash file system algorithms typically depend on this capability for efficiency and to reduce wear on the flash. This is especially important considering the lower maximum erase/program cycle count.

4. Programming must be sequential, from LSB to MSB. The flash management software must make sure its algorithm will not write random pages within a block.

A flash file system designed for SLC flash is not likely to work on MLC flash, without significant modification. Depending on the algorithms used, it may be impossible without rewriting it. We have read this is true of JFFS2. In the case of our flash driver

(smxNAND, used by smxFFS and smxFS), relatively few modifications were needed. They were related to block table management and ECC. Fortunately, no changes were needed in other areas, such as its handling of data blocks. However, the modified driver is a little less efficient in its use of flash pages, causing a little more wear on the flash. But if you choose a significantly larger MLC flash part than you need (which may be cheaper than an SLC chip of the capacity you need), the static and dynamic wear leveling done by the driver will spread the usage over the larger flash area, wearing the cells to some intermediate and possibly acceptable level.

Why Use MLC Then?

Generally, we recommend using SLC in typical embedded systems. However, in very high volume products (millions), it probably makes sense to consider MLC. For example, one customer of ours doesn't care much about flash capacity; they just buy the cheapest flash chip. In their case, the MLC chip is $1.50 cheaper than the SLC chip, in the quantity they buy, which would save them $millions.

Other Options

Instead of using raw NAND chips, you can use managed NAND devices, such as eMMC, eSD, iNAND, and microSD. These put the NAND flash software into the device itself to handle block management, ECC, wear leveling, etc. The system software does not need to know the details and can just access the flash disk as a normal block device. No flash driver software is needed, only a file system with a driver for the device. This might be a good option if you want to use a lower-cost low-performance processor or one that has no MLC + ECC controller, or if you are developing a typical, low-volume embedded product. Besides avoiding the need for an MLC flash file system, this also reduces your system's requirements for RAM, ROM, and CPU power. Some such devices are removable and some can be soldered to your board. Note that removable devices can be put fully within your enclosure to prevent user removal.

Helpful References

1. MLC NAND Flash Webinar
   www.micron.com/products/nand/**mlc**-webinar

2. SLC vs. MLC: An Analysis of Flash Memory
   www.supertalent.com/datasheets/SLC_vs_**MLC**%20whitepaper.pdf

3. Implement MLC NAND Flash for Cost-Effective, High Capacity Memory
   www.data-io.com/pdf/NAND/MSystems/Implementing_**MLC**_NAND_**Flash**.pdf

4. Are MLC SSDs Ever Safe in Enterprise Apps?

http://www.storagesearch.com/ssd-slc-mlc-notes.html

5. File System support on Multi Level Cell (MLC) flash in open source
http://www.celinux.org/elc08_presentations/ELC2008%20Filesystem%20support%20on%20Multi%20Level%20Cell%20flash%20in%20open%20source.ppt

Whether you decide to use MLC or SLC in your system, please consider using one of our file systems, smxFFS or smxFS, which use the smxNAND flash driver.

Contact us if you have questions about using NAND flash in your design.
Yingbo Hu, R&D Embedded Software Engineer, yingbohu@smxrtos.com
David Moore, Director of Development, davidm@smxrtos.com