Several years ago, using NAND flash was pretty simple: the devices were reliable enough that when you wrote data, you had a pretty good chance of reading back the same data. Over time, however, NAND flash has increased storage density by moving to smaller geometries and storing more bits per cell, which has required higher levels of error correction in order to ensure the integrity of the data on the flash device. [More on this in a later post]

**Hamming Codes:**

On the original NAND devices, manufacturers recommended that users use a simple error correction code (ECC) called a block Hamming code that could correct a single bit error and detect when multiple bit errors occurred. These early devices had 512B pages, so the correction was typically applied to the entire flash page. Block Hamming codes are relatively easy to construct and are often implemented in software, since the overhead to generate the parity and corrections is trivial. Hamming codes are a simpler form of the SEC/DED (single error correct, double error detect) codes used in DRAM. The difference, however, is that for NAND flash, the correction is applied to a block of data (thus it is a block code) versus a single doubleword of DRAM.

**Reed-Solomon (R/S or RS) Codes:**

As error rates increased on the NAND devices, the likelihood of multiple bit errors within a single page increased. To accommodate this, Hamming codes were used over smaller blocks, or designers turned to stronger error correction to ensure the data integrity. Many engineers were familiar with Reed-Solomon codes from communications and other storage applications (CD-ROM), so it became the obvious choice for NAND flash error correction. By this time, devices had moved to 2KB or 4KB pages and MLC devices required 4-bit corrections. A typical Reed-Solomon implementation will split the 512B correction blocks into multi-bit symbols (typically 8 or 9 bits) and provide N-symbol corrections. If any number of bits within the symbol are corrupted, the RS code will correct the entire symbol.

**Bose-Chaudhuri-Hocquenghem (BCH) Codes:**
As error correction requirements for MLC flash increased above 8 bits per 512B correction block, the storage requirements for RS parity bytes began to outgrow the number of bytes available in the spare area of the flash devices. In order to minimize the number of parity bits required, NAND manufacturers began recommending that BCH codes be used for corrections. BCH is a more efficient ECC algorithm because NAND flash errors are non-correlated, meaning that they do not occur in groups or clusters. Instead, they are randomly distributed across the NAND page. Since RS codes correct symbols, its strength is in applications where errors tend to be clumped together -- such as a scratch on a CD-ROM or a garbled communication channel.

BCH is more efficient since it targets only single bit errors (the error mode in flash) and can correct more of these bit errors than an RS code when given the same number of parity bytes. As NAND error rates continued to grow, BCH ECC levels increased and the correction block size changed to 1KB in order to further improve correction efficiency.

Low-Density Parity Check (LDPC) Codes:

The advent of TLC (3-level cell) flash and even smaller geometries will further increase the error correction requirements in NAND flash. In the future, BCH correction will likely give way to LDPC codes as the ECC of choice until another technology is available to replace NAND flash as the cutting edge storage technology.