Choosing NAND or NOR Flash Memory: Tradeoffs and Strategies

Consumer electronics and embedded software devices are using larger amounts of flash memory for nonvolatile storage than ever before. One important decision in designing such devices is what kind of flash memory to use: NAND or NOR?

NOR flash memory has traditionally been used to store relatively small amounts of executable code for embedded computing devices such as PDAs and cell phones. NOR is well suited to use for code storage because of its reliability, fast read operations, and random access capabilities. Because code can be directly executed in place, NOR is ideal for storing firmware, boot code, operating systems, and other data that changes infrequently.

NAND flash memory has become the preferred format for storing larger quantities of data on devices such as USB Flash drives, digital cameras and MP3 players. Higher density, lower cost, and faster write and erase times, and a longer re-write life expectancy make NAND especially well suited for consumer media applications in which large files of sequential data need to be loaded into memory quickly and replaced with new files repeatedly.

The choice between using NAND and NOR Flash may not be a simple one for the complex embedded devices being developed today. While ever-larger media files are driving increased demand for inexpensive NAND, powerful new operating systems and intricate applications running on fast processors call for the kind of fast-executing code NOR can support. An important example is a smart phone or PDA that combines a tremendous need for storage with a demanding set of application performance requirements. In some cases an optimal design might call for both types of flash memory in the same device.

Whichever type of flash is used in a device, there are certain negative performance characteristics that need to be mitigated. NOR is fast to read current data but markedly slower to erase it and write new data. NAND is fast to erase and write, but slow to read non-sequential data through its serial interface. NAND is also prone to single-bit errors, requiring rigorous algorithms for error detection and correction.

Well-designed software strategies can be very effective in increasing the performance and reliability of Flash hardware. The goals of flash memory management software include:
Avoid loss of data. Perhaps the most important goal in managing flash memory is to assure that no data is ever lost as a result of an interrupted operation or the failure of a memory block. There are several ways that flash management software can achieve this goal. Rewrite operations, for example, can be managed in such a way that new data is written and verified before the old data is deleted, so that no power loss or other interruption can result in the loss of both old and new data. Bad block management is another important safeguard to prevent data being written to memory blocks that have failed. Software can check for bad blocks shipped from the factory, as is typical with NAND, and avoid writing to those blocks from the beginning. When blocks go bad over time they can be identified and managed so that they are no longer used. Finally, as the end of media life nears, good memory management software can implement a graceful strategy such as placing the entire flash unit in a read-only state, thereby avoiding data loss when the number of block errors exceeds a predefined number.

Improve effective performance. Two ways media management software can improve performance are background compaction and multithreading. Compaction reclaims space by identifying blocks that have obsolete data that can be erased, copying any valid data to a new location, then erasing the blocks to make them available for reuse. Such compaction increases the amount of usable space on the media and improves write performance. Compaction may also help to defragment noncontiguous data for improved performance on read operations. The space recovery is particularly valuable for the more costly NOR memory and the defragmentation benefits the slower-reading NAND. Compaction is best performed in the background during idle time, however, or it can interfere with critical operations and degrade performance. This is where a multithreading system becomes important. By allowing high-priority read requests to interrupt low-priority maintenance operations, a multithreading system can reduce read latency by orders of magnitude compared to a single-thread solution.

Maximize media lifespan. When some blocks of memory contain fixed content, such as binary code, the remaining blocks will experience increased demand for erase and write operations, leading to earlier failure. Wear-leveling algorithms can prevent overuse of memory blocks and prevent a “stalemate” scenario in which a small region of memory becomes locked in a pattern of repeated writing and compaction. Wear leveling software can monitor block usage to identify high-use areas and low-use areas containing static data, then swap the static data into the high use areas. It can also balance write operations across all available blocks by choosing the optimal location for each write operation.

The decision between NAND and NOR memory will ultimately depend on both technical and pricing requirements of the device being built. Whatever type or combination of flash is used, it is prudent to include memory management software to prevent data loss while improving the performance and maximizing the lifespan of the memory.