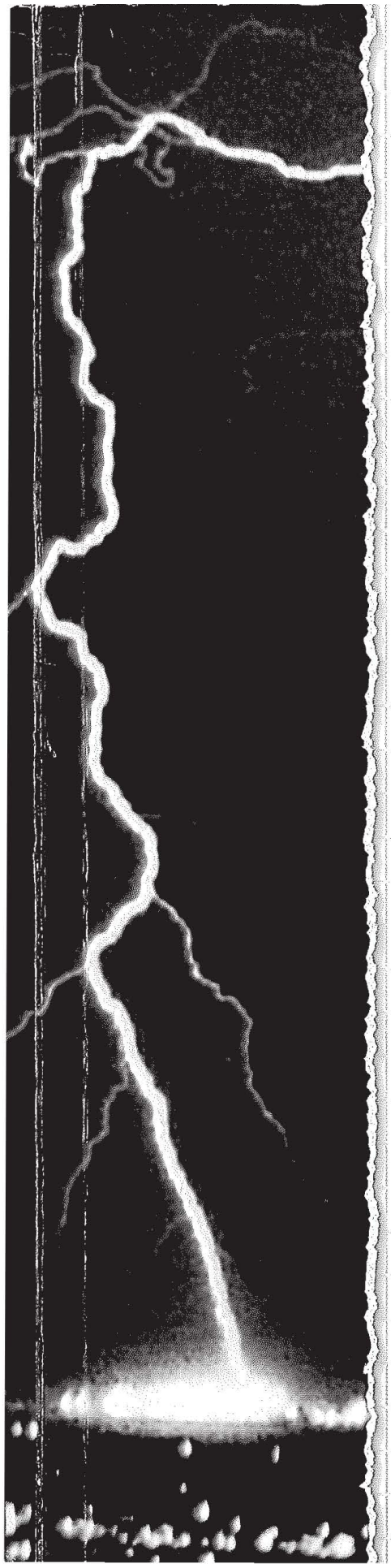


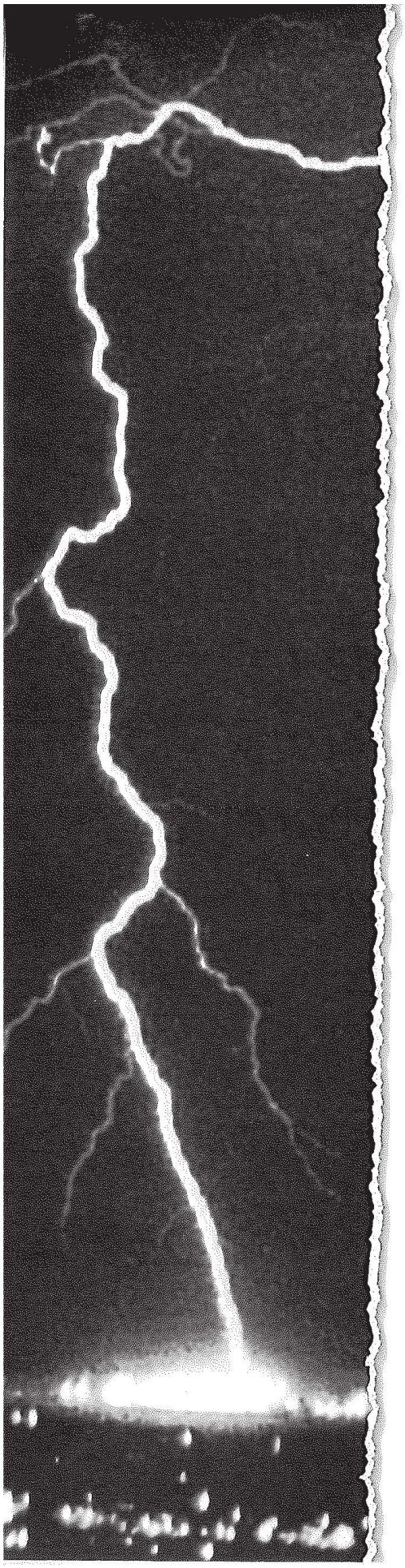
Designing with FLASH MEMORY

Brian Dipert and Markus Levy

The definitive guide to designing flash memory hardware and software for components and PCMCIA cards.

Designing with Flash Memory

The definitive guide to designing flash memory hardware and software for components and PCMCIA cards



Designing with Flash Memory

Brian Dipert & Markus Levy

The definitive guide to designing flash memory hardware and software for components and PCMCIA cards

Annabooks
San Diego

Designing with Flash Memory

BY

BRIAN DIPERT & MARKUS LEVY

PUBLISHED BY

Annabooks
11848 Bernardo Plaza Ct., Suite 110
San Diego, CA 92128
USA

619-673-0870

Printed in the United States of America

ISBN 0-929392-17-5
Second Printing April 1994

Contents

Figures

Tables

Chapter Three: Flash Memory Technologies

From a very high-level perspective, Chapter 1 answered the question, "What is Flash Memory?" As a review, flash memory has the following primary characteristics:

- Nonvolatility (retains data stored to it when powered off), and
- In-System Updateability (stored data can be erased and replaced under system processor control)

As you can see, this is a pretty broad definition! Various semiconductor vendors have chosen unique and quite dissimilar silicon technology approaches to answer the above application requirements. Some flash memory approaches are *evolutionary*, based on existing memory types that are already nonvolatile and updateable. Other technologies choose a more *revolutionary* path.

This chapter will discuss in detail three flash memory technologies: NOR, EEPROM, and NAND. All three approaches meet the basic criteria for flash memory (nonvolatility and updateability). Where they differ, however, is in their secondary characteristics, some of which are listed below[3]:

- Read Performance
- Program/Erase Performance

[3]Chapter 2 discussed specific flash memory applications and indicated the highest priority features in each case.

- Number of Program/Erase Cycles Through Device Lifetime
- Power Supply Voltage Requirements
- Current Draw in Device Operating Modes
- Erase Block Size

When evaluating flash memory alternatives, do not overlook the manufacturing process complexity, and the size of the flash memory cell and periphery logic. Both factors translate into component cost, and ultimately to the price you pay for the component or flash-based subsystem from the manufacturer or distributor. Keep this in mind as you read about the "latest and greatest" flash memory technology unveilings. Creating something in the laboratory is one thing; consistently recreating it in high volume and with low cost in a manufacturing facility is entirely another matter!

As a framework for the following discussion, Figure 3.1 shows the 1992 relative market share for several flash memory semiconductor vendors. The anticipated demand for flash memory in the very near future is evident, and many semiconductor companies are gearing up to supply this market.

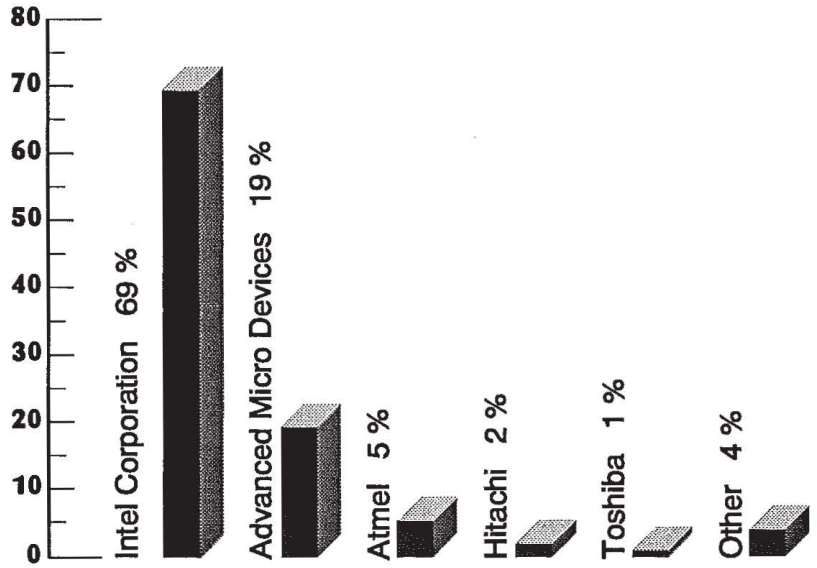


Figure 3.1: Dataquest 1992 Flash Memory Market Share (by company)

NOR FLASH MEMORY

(Examples: Intel Corporation, Advanced Micro Devices, Hitachi, Mitsubishi, NEC, SGS-Thompson, Fujitsu, Toshiba Corporation)
NOR flash memory was introduced by Intel Corporation in 1988, using the company's ETOX™ (EPROM Thin Oxide) process technology. Since that time, products based on similar technologies have been announced by several other semiconductor vendors. Figure 3.2 compares the ETOX flash memory cell with an EPROM (Erasable Programmable Read-Only Memory) cell. The similarity in this revolutionary approach is clear; NOR flash memory derives from an EPROM base. The key difference is in the silicon oxide thickness between the floating gate and substrate. This thinner oxide is the key to NOR flash memory operation; we'll see why in a moment.

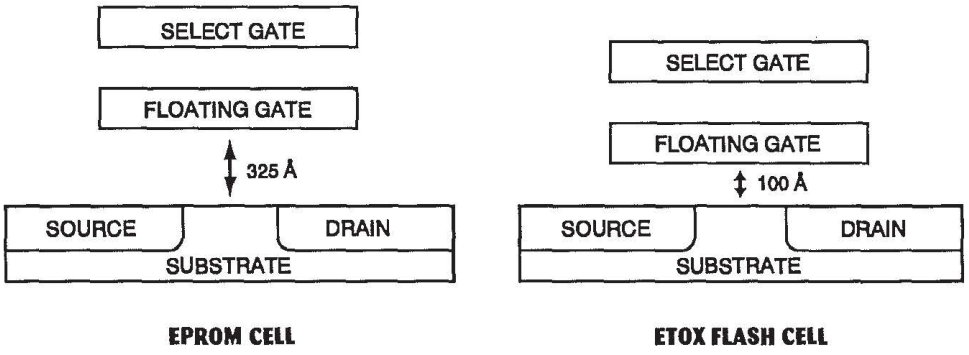


Figure 3.2: ETOX™ Flash Memory Cell Similarities Leverage EPROM Learning Curve

When shipped from the vendor, the default state of all cells in a NOR flash memory is one, corresponding to an *erased* condition. Figure 3.3 shows the voltages present on the cell when read. When erased, the floating gate of the flash memory cell does not block the cell from being turned on by the applied voltages on the select gate and drain. The resulting current is sensed at the transistor source, and translated to a one at the memory output pin.

Figure 3.4 shows a portion of a flash memory array and the interconnection of the various transistors. Device addresses enable specific wordlines and bitlines; in combination they select one transistor

within the array per device output. This organization also explains the NOR name for this architecture; any "on" transistor (i.e., a selected, erased cell) in the chain results in the earlier-described current draw, sensed at the end of the chain and converted to an output one.

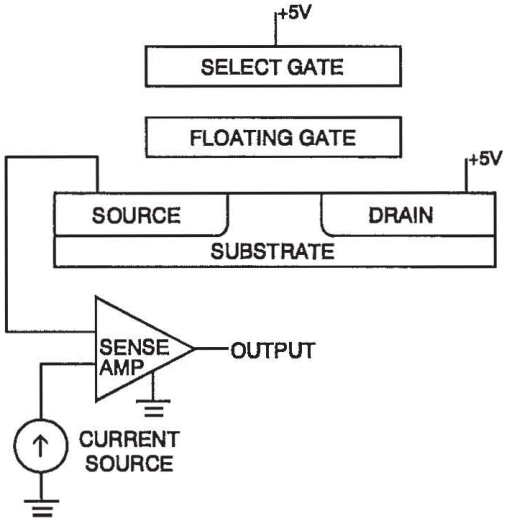


Figure 3.3: ETOX™ Flash Memory Cell Being Read

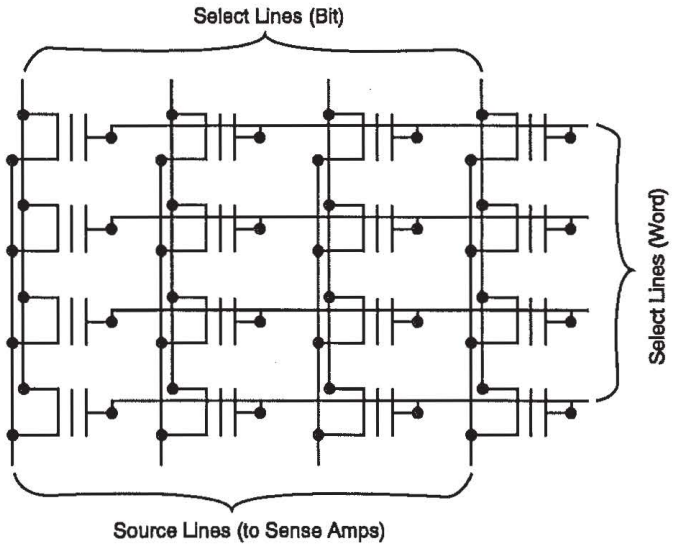


Figure 3.4: NOR Flash Memory Array Interconnect

Program

Changing a flash memory cell (or bit) to a zero is called *programming*. NOR flash memory employs the same programming mechanism as EPROM, namely hot electron injection. Figure 3.5 shows an ETOX flash memory cell being programmed. As electrons travel from the source to the drain through the substrate, the electric field generated by high voltage on the select gate causes some of the highest energy electrons to jump the gap and collect on the floating gate. What's the result? Referring back to Figure 3.3, we see that the electrons now present on the floating gate counteract the voltage on the select gate and prevent the flash memory cell from turning on. No current flows from drain to source, resulting in a zero on the memory output pin.

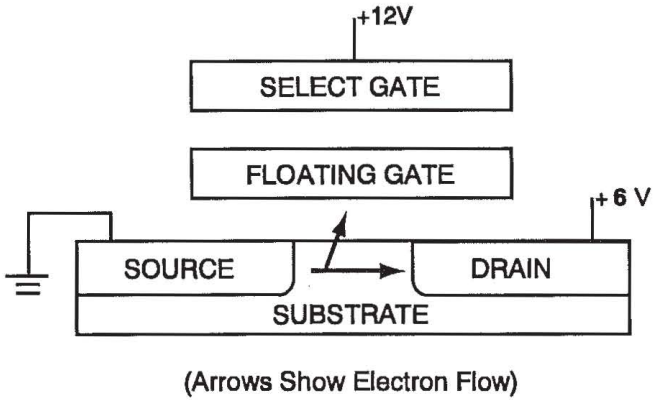


Figure 3.5: ETOX™ Flash Memory Cell Being Programmed

NOR flash memory cells can be selectively programmed to zero. In other words, programming is a *bit-level* operation. On a byte-wide flash memory device, for example, one bit of a selected byte can be programmed to zero, leaving the other seven bits at one. Later programming of the same byte can change other bits to zero in the same way. However, one key point to note about NOR flash memory (and about other flash memory approaches, too) is that *programming only changes ones to zeros*. Here lies a fundamental difference between flash memory and other rewriteable memory technologies like RAM. To change programmed zeros back into ones, we must use a different mechanism, called erase.

Erase

EPROMs are erased by ultraviolet light. As shown in Figure 3.6, the extra energy generated by UV light enables electrons on the floating gate (put there by programming) to overcome the inherent semiconductor energy potential and return to the substrate. After erasure, an EPROM cell once again reads as a one. To allow UV light to shine on all EPROM cells on a device array, the package must include a built-in glass window. As manufacturing lithographies become smaller and smaller, it becomes harder and harder to ensure that UV light can reach all array cells. The window requirement also puts limits on how small the device package can become.

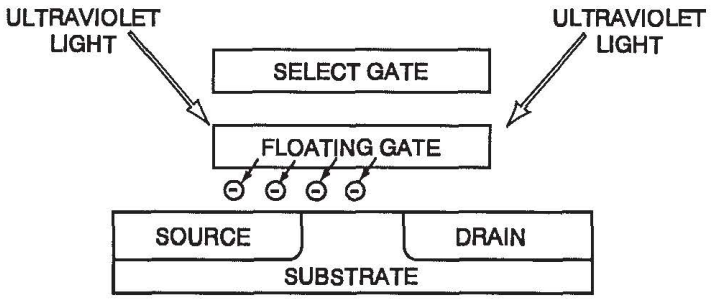


Figure 3.6: EPROM Cell Being UV Erased

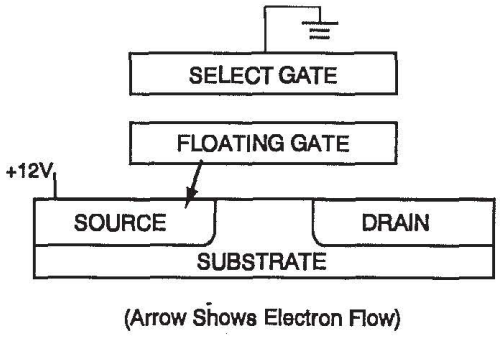


Figure 3.7: ETOX™ Flash Memory Cell Being Erased

Rather than using UV light, NOR flash memory cell erasure is accomplished electrically using a process called Fowler-Nordheim tunneling. Figure 3.7 shows the voltages on the flash memory cell during

erase. The generated electric field pulls electrons from the floating gate. First generation *bulk-erasure* NOR flash memories erase all cells in the array at the same time. Second generation NOR devices erase in smaller blocks. Following the same train of thought, this is called *block erase*. Erase block size varies from flash memory vendor to vendor, and from device to device, based on the targeted applications.

Compared to EPROM, the array transistors in a flash memory need not be accessible to UV light exposure. This allows flash memory designers to run layers of interconnection over the cell versus around it, simplifying the design and minimizing the device die size. As an analogy, think of a multi-layer versus a single-layer printed circuit board. Also, flash memory does not require the window of an EPROM, allowing very small footprint (and less expensive) packaging[4].

Negative Gate Erase

Negative gate erase is similar but not identical to the conventional cell erase approach described earlier. Figure 3.8 shows the voltages on the flash memory cell during negative gate erase. Comparing this diagram with Figure 3.7, we see that although the voltages on the cells are different, the resultant voltage potential difference (and electric field) between gate and source is similar. Negative gate erase also uses Fowler-Nordheim tunneling to remove electrons from the floating gate.

Overerase

Removing too many electrons from the floating gate of a flash memory cell may theoretically result in an *overeased* condition (i.e., removing more electrons than were put there by a previous cell program). The effects of overerase are destructive to the flash memory device. Once overerased, a flash memory cell cannot be programmed again (within practical limits). Reads of this cell, as well as adjacent cells in the array, produce erratic and invalid results. Referring back to Figure 3.4, we see that an overerased cell, being "always on" even if not selected, overrides any valid data on the array transistor "chain". *Oops!*

[4]We'll see this again in Chapter 4.

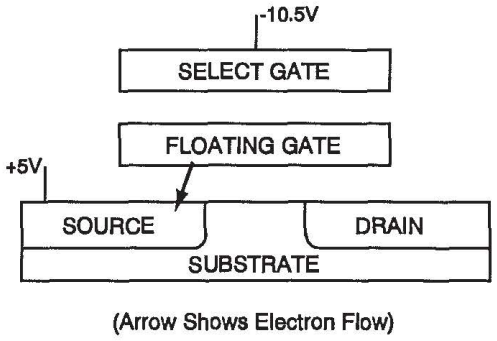


Figure 3.8: Negative Gate Erase

Fortunately, flash memory erase algorithms include built-in procedures to eliminate the potential for overerase[5]. First, cell erase (like cell programming) uses an iterative algorithm. Shown in simplified form in Figure 3.9, the built-in feedback loop ensures that the algorithm terminates and does not allow further removal of floating gate electrons once sufficient cell erase has been detected. Secondly, since all flash memory cells in a given device (or block within an device) are erased in parallel (and at approximately the same rate), preprogramming ensures that all cells are at a common initial programmed state. Without preprogramming, already-erased cells in the device or in a given erase block would be overerased while programmed cells were being erased.

Newer NOR flash memories control the erase algorithm internally, and automate both the erase preprogramming and iterative erase/verify steps. This dramatically simplifies system software algorithms and eliminates any potential for error. For more information, reference Chapter 7.

NOR Flash Memory Specifications

Table 3.1 provides a summary of NOR flash memory device characteristics, derived from data on Intel Corporation's latest-generation products. These specifications are indicative of the relative levels of performance possible today using NOR flash memory. However, exact

[5]If they are implemented *exactly* as published; a 'word to the wise' for system software programmers!

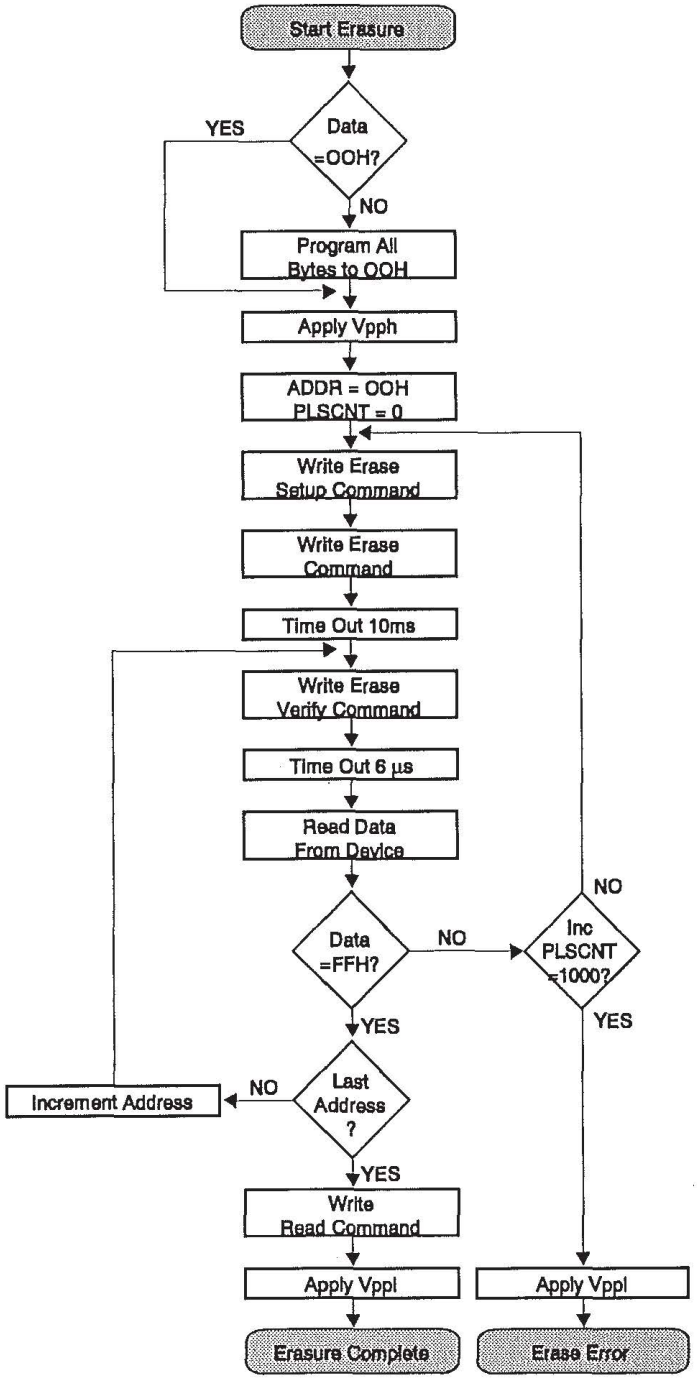


Figure 3.9: Iterative Basic Flash Memory Erase Algorithm

values will vary from device to device, from manufacturing process to manufacturing process, and from vendor to vendor[6].

Density	8 Mbit
Access Time	60 ns
Data Program Time	6 µs (min) 9 µs (typ)
Block Erase Time (64 kbyte block)	0.3 sec (min) 1.6 sec (typ)

Table 3.1: NOR Flash Memory Characteristics

Note the relatively slow erase time compared to read and program. Cell erase time is a primary function of two parameters; oxide thickness between floating gate and substrate, and internal erase voltage (it is also affected by device temperature, and by the number of times the cell has been erased previously, or *cycled*). The cell erase time of the ETOX processes is a direct result of the relatively low 12V and low current used to pull electrons from the floating gate. However, a low erase voltage also translates to excellent cell reliability and extended cycling performance. Later chapters will give examples of flash memory applications where cell erase time is (and is not) a concern, as well as discussing hardware and software techniques to hide the slow erase as a background system task.

FLASH EEPROM

(Examples: Atmel Corporation, Samsung, SunDisk, Catalyst Semiconductor)

The previous discussion showed how NOR flash memory was derived from an existing EPROM base. Similarly, flash EEPROM shares many similarities with standard EEPROMs. Figure 3.10 shows a diagram of a flash EEPROM memory cell.

[6]Consult vendor datasheets, application notes, and engineering reports for information on specific devices. Vendor contact information is in Appendix A.

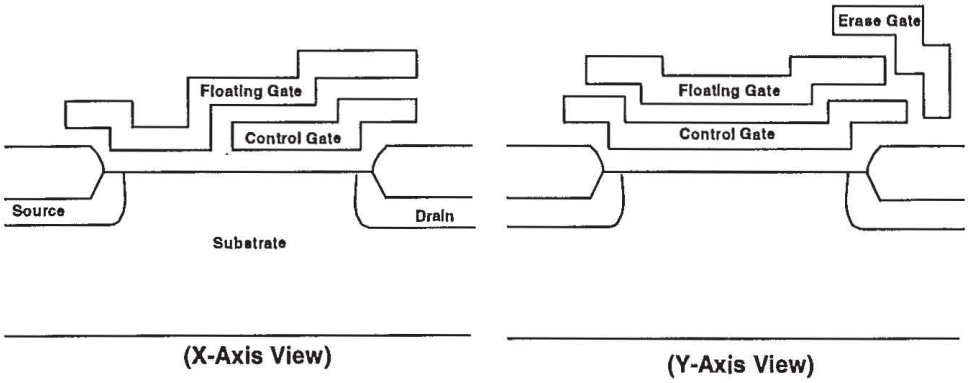


Figure 3.10: EEPROM-Based Flash Memory Cell

A standard EEPROM can be fully altered on a byte-by-byte basis. The byte erase operation is integrated in the write function, i.e., the byte is first erased and then reprogrammed with the desired data. A flash EEPROM, on the other hand, simplifies the silicon design by erasing on a block-level basis. When an EEPROM flash memory block is written, it is first erased and then programmed with data stored in an on-chip buffer.

Erase

Flash EEPROMs erase using Fowler-Nordheim tunneling, as do NOR flash memories. Most, however, use a separate erase gate per cell to collect electrons pulled off the floating gate. Regardless of the specific method, flash EEPROMs use much higher internally-generated voltages because of their greater oxide thickness compared to NOR flash, and to speed erase performance. Remember....erase is a built-in part of rewrite, not a separate operation as in the case of NOR flash. Figure 3.11 shows an EEPROM flash memory cell being erased.

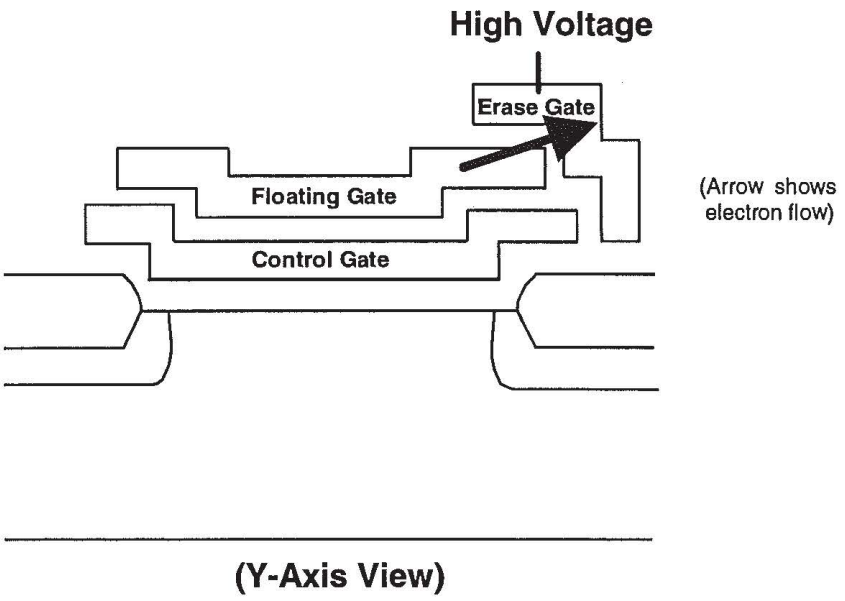


Figure 3.11: EEPROM-Based Flash Memory Cell Being Erased

Programming

Some flash EEPROMs program cells via hot electron injection. Most, however, use a reverse form of Fowler-Nordheim tunneling shown in Figure 3.12. The combination of voltages on the select gate and drain stores electrons on the floating gate, versus removing them, as seen with Fowler-Nordheim erasure. Again, high internal voltages are used for fastest programming performance.

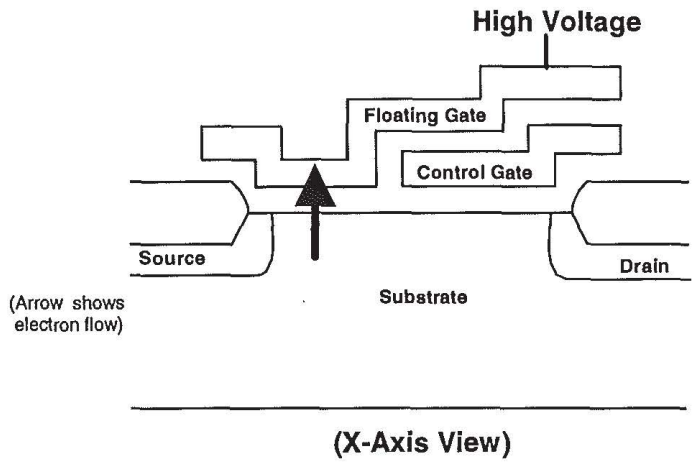


Figure 3.12: EEPROM-Based Flash Memory Cell Being Programmed

Flash EEPROM Memory Specifications

Table 3.2 summarizes flash EEPROM memory characteristics[7]. Since erase is a built-in part of the flash EEPROM program algorithm, flash EEPROMs speed up the erase process time compared to NOR flash memory, primarily via the higher internal voltages on the EEPROM cell. However, over time this may potentially have a negative impact on cell reliability. As the EEPROM cell undergoes repeated erasure, the high electrical field can break down the thin oxide region, causing failure. Some EEPROM vendors have implemented redundant cell and internal error-correction schemes to combat this "Achilles Heel".

Density	1 Mbit
Access Time	90 ns
Data Program Time	150 µs

Table 3.2: EEPROM Flash Memory Characteristics

[7]Taken from Atmel Corporation documentation.

NAND FLASH MEMORY

(Example: Toshiba Corporation)

NAND flash memory is a relatively new technology approach pioneered by Toshiba Corporation. As shown in Figure 3.13, the NAND flash memory cell looks very much like a NOR cell! However, the periphery logic designed into NAND is very different, and the internal program and erase approaches most closely resemble flash EEPROM methods.

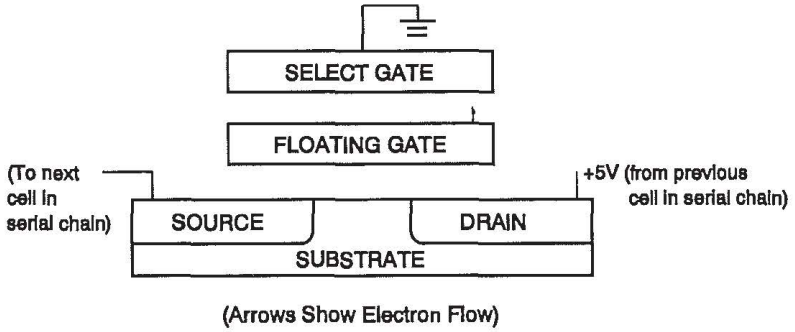


Figure 3.13: NAND Flash Memory Cell Being Read

Like Figure 3.4, Figure 3.14 shows the interconnection of transistors in a NAND array. Data sensing along the chain is serial in nature, and the architecture reflects its name.

Program and Erase

NAND flash memory cells program and erase via reverse and forward Fowler-Nordheim tunneling, respectively. Figures 3.15 and 3.16 show the internal voltages on the cell in each case. Note that unlike flash EEPROM memory tunneling, NAND flash memory applies voltages to the substrate itself, in addition to the select gate.

NAND Flash Memory Specifications

Table 3.3 shows initial specifications for Toshiba's first NAND flash memory-based device. NAND flash memory primarily targets solid state disk drive replacement applications, and the feature set reflects this, with fine-resolution blocking and fast cell erase. However note the slow initial read access time due to serial data read, which may limit broad application usage. Some NAND devices include error detection and correction (EDAC) cells and associated EDAC logic.

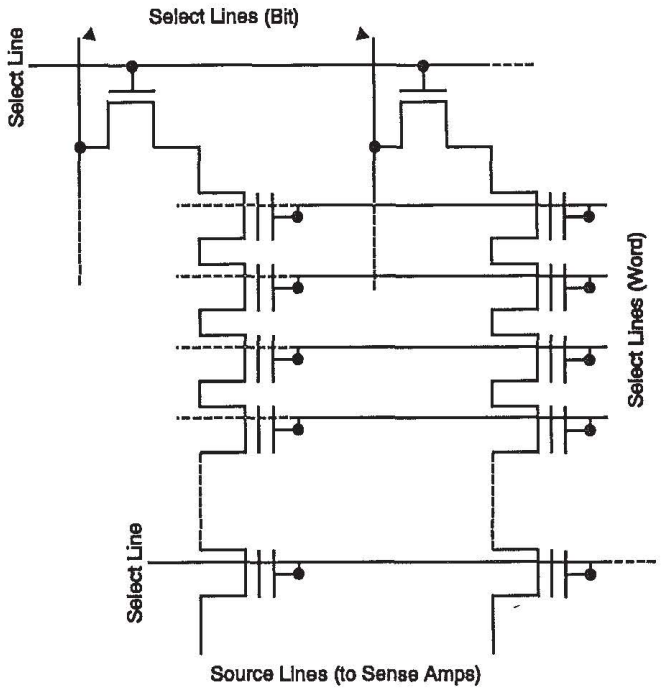


Figure 3.14: NAND Flash Memory Array Interconnect

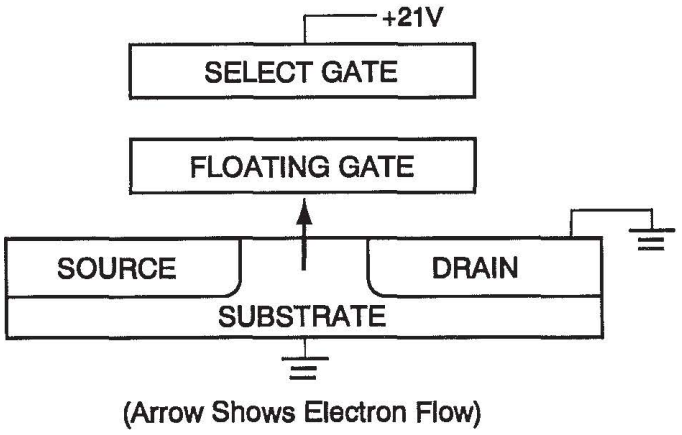


Figure 3.15: NAND Flash Memory Cell Being Programmed

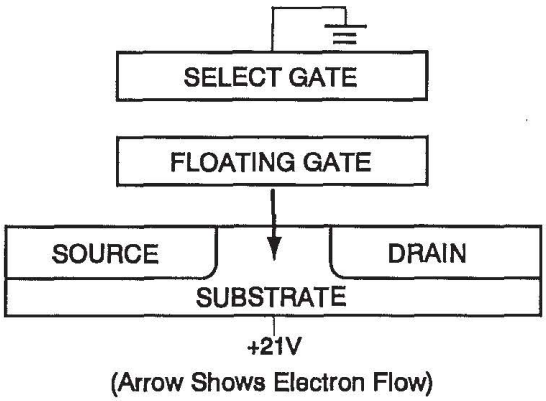


Figure 3.16: NAND Flash Memory Cell Being Erased

Density	4 Mbit
Access Time	15 µs (initial read) 80 ns (subsequent serial access)
Data Program Time	4 ms (min)
Block Erase Time (4 kbyte block)	6 ms

Table 3.3: NAND Flash Memory Characteristics

WHAT'S ALL THIS *CYCLING* STUFF, ANYWAY?

The subject of cycling is quite possibly the most abused (by companies supplying flash memory) and most misunderstood (by companies buying and using flash memory in their system designs) of any topic you'll find discussed in this book! All sorts of outlandish claims have been made, are being made, and will probably be made in the future, concerning the cycling capabilities of various flash and "flash-like" memory technologies. To confuse you even further, concepts such as MTBF (mean time before failure) are often used in conjunction with cycling specifications. Flash memory vendors often mean well (from a marketing perspective) when they include these numbers, but since an industry standard for the determination or calculation of MTBF doesn't

exist, it is often subject to liberal interpretation and modification. Therefore, MTBF numbers for different flash memory devices and technologies cannot be directly compared without knowing the recipes that were used and the assumptions that were made when the measurements were taken.

We are going to explain cycling in its most fundamental definitions for you, and provide guidelines by which you can calculate your own MTBF numbers for your specific flash memory design and implementation. Our goal here is to cut through all the meaningless marketing hype and provide you with valid, useful information.

What is cycling? A cycling number is:

a) The minimum number of times a flash memory device (or block within a device) can be erased and programmed in a reasonable amount of time without loss of device functionality, at

b) A specified failure rate percentage, or FIT (failure-in-time) level.

Flash memory vendors often ignore the latter part of the above definition when publishing their cycling specifications. What good is it to know how many times you can erase an array of flash memory cells if you have no idea of the probability that some of the cells will fail before reaching this cycle count? A parallel can be drawn here with stereo equipment, where inflated claims are sometimes made of an amplifier's output power capability without mentioning how distorted the output signal was when this power was measured. What good is it to hear loud music if you can't understand it? (Of course, with some forms of popular modern music this could be seen as a positive!) Similarly, what good is it to be able to erase flash memory to an extended number of cycles if the media is essentially unusable when it reaches this cycle count? Clearly both parts of the cycling definition are valuable and useful information.

Failure Analysis

Before each flash memory device is shipped to a customer, it undergoes extensive testing to screen out known and detectable failure mechanisms both in the circuitry itself and in the manufacturing process on which the device was made. Even after this testing, it is known and accepted that a certain very small (hopefully!) number of devices will eventually fail, even when operated at all recommended specifications. Some sources of this failure, common to all flash memories as well as other memory technologies, are listed below:

- Package Integrity Failures
- Random Circuitry Failures
- Data Reliability Failures (i.e., programmed zeros turning back into ones)
- Program Failures (inability to change a one to a zero), and
- Erase Failures (inability to change a zero back to a one)

Reputable flash memory vendors spend a great deal of time and effort calculating and predicting their failure rates. Published reliability reports contain these predicted failure percentages, and are available for your inspection. We'll restrict the following discussion to the last two failures listed above, program and erase (or cycling) failures.

How and why does a flash memory cell fail due to cycling? Two different mechanisms combine here; one a more "destructive" phenomenon (oxide breakdown) and the other "non-destructive" in nature (electron trapup).

Oxide Breakdown

Notice the thin oxide region between the substrate and floating gate regions in Figure 3.10. As a flash memory cell is repeatedly erased and reprogrammed, the electrons move back and forth through the oxide region under an electric field. This stresses the oxide, and in its most severe form can result in oxide breakdown and a short circuit between oxide and substrate, rendering the cell non-functional. High quality oxide with low probability of defects, as well as a lowered electric field

to minimize oxide stress, are ways that flash memory vendors can minimize the likelihood of oxide breakdown.

Electron Trapup

Recall that the earlier definition of cycling included the phrase "erased and programmed in a reasonable amount of time". This is key to the definition of electron trapup. As a flash memory cell accumulates higher and higher cycle counts, electrons become trapped in the oxide region, lowering electron mobility through the oxide and resulting in increased program and erase times. The program and erase algorithms must apply more pulses to program or erase the cell sufficiently to ensure data integrity and retention. Since the impact of electron trapup is simply a failure to program or erase within an allowed time and not a "hard" failure of the cell itself, we call it a "non-destructive" phenomenon.

Mean Time Before Failure

With cycling and failure rate data, and with a good understanding of how flash memory will be used in your system, you can calculate MTBF values for your specific design. As an example, we'll use the Intel 28F008SA 1 Mbyte FlashFile™ memory in a configuration of 20 chips (20 Mbytes total).

The Intel 28F008SA is rated for 100,000 cycles on each of its sixteen 64 kbyte blocks (independent of any other block). Data taken through 10,000 cycles shows no cycling failures, translating to a 0% cycling failure rate (pretty impressive!). Therefore, for this example we'll use the more stringent device failure rate of .01%, which encompasses *all* device failure mechanisms listed earlier in this chapter. The value 0.01% is the historic worst-case device failure rate seen with production-rated Intel flash memories, and the 28F008SA should perform at least this well (if not better).

A 0.01% failure rate (translating to 100 FITS or failures-in-time) means that fewer than 1 in 10,000 devices will fail after 10,000 cycles and 1,000 hours of operation. The scenario under which we'll calculate MTBF assumes that a 10 kbyte file is written to the 20 Mbyte array of flash memory every 10 minutes; a pretty rigorous set of assumptions if you think about it!

A flash-friendly file system could use a linked list structure to write multiple copies of a file and fill up clean flash memory, marking old versions of the file "dirty" but not erasing them immediately[8]. This significantly minimizes cycling of flash memory media. Therefore, given the file and flash memory array sizes, we can make the following calculations:

(20 Mbyte array) / (10 kbyte file) = 2,000 file writes can be done before an array erase is required
(2000 file writes/erase) x (10,000 cycles per 28F008SA block) = 20 x 10^6 file writes
(20 x 10^6 file writes) x (10 minutes/write) x (1 hr/60 minutes) = 3.33 x 10^6 hours MTBF

This means that our 20 Mbyte flash memory array has a Mean Time Between Failures of over 3 million hours, at a failure rate of 0.01%. Not bad, eh?

Extended Cycling-The Flash Memory Manufacturer's Options

Earlier when defining cycling, we inferred that the easiest way some flash memory vendors achieve extended cycling was by downplaying the negatives and accentuating the positives of their technology approaches. This, while true, is not the only means of reaching the extended cycling "Holy Grail"! Several other concrete tradeoffs have been made by various flash memory suppliers, both in technology and architecture, in pursuit of this goal.

Oxide breakdown can be eliminated by producing very high quality, uniform oxide for each flash memory cell. This is much more difficult than it might first appear, and in fact is probably the most complex problem that semiconductor vendors have struggled with as they attempt to ramp up their flash memory manufacturing capabilities. The oxide layer, at 100 Å thick, is made by laying down several layers of silicon *atoms*, no simple task. Remember, too, that for an 8 Mbit flash memory, not one cell but over *8 million* must be manufactured correctly to yield a functional device, and that potentially several hundred devices can be made from each 6" or 8" silicon wafer.

Another technology tradeoff can be made with respect to the internal electric field during program and erase, which is a function of the

[8]See Chapter 9 for more information.

magnitude of the internal voltages. A lower electric field lowers the stress on the oxide (a positive) but also slows program and erase times (a negative). Intel Corporation, with its ETOX flash memory approach, has made this choice, and has added device functionality to minimize the system performance impact of the resultant slow block erase time[9].

Where flash memories use higher internal voltages (flash EEPROM and NAND flash memories), added circuitry attempts to circumvent the impact of oxide breakdown and resultant cell damage. EEPROMs often use redundancy schemes which lower cycling failures at the expense of doubling cell size and adding complexity. Toshiba's NAND flash memory integrates error detection and correction (EDAC or ECC) directly on the silicon to mask the device impact of single cell failures. While potentially extending the cycling capability of the device, this approach adds complexity and die size to each device, and also impacts read performance.

Extended Cycling-What Can You Do?

What can you do to match the cycling requirements of your design to an appropriate flash memory architecture? First and foremost, fully analyze the cycling you truly require, and take all possible steps to minimize this cycling. A design that uses flash memory for embedded code storage may only be erased and reprogrammed ten times through its lifetime. On the other hand, a memory card used for file storage may have blocks of flash memory updated thousands or hundreds of thousands of times. Specifically with respect to file storage, Chapter 9 will explain how software companies have re-architected file storage beyond the hard drive paradigm to match the unique characteristics and capabilities of flash memory. These concepts, while possibly not directly applicable to your specific design, will provide examples of cycle minimization and management, linked list structures, and wear leveling.

In Chapter 7 we'll discuss the system software algorithms that initiate and control flash memory erase and program. In cases where erase failure has occurred due to non-destructive electron trapup, this chapter

[9]Upcoming chapters will discuss flash memory automation, the RY/$\overline{BY}$ output and erase suspend/resume capability.

will show you how to extend cycling by supplying the flash memory media with additional erase and program pulses.

Finally, it's your responsibility to understand the conditions under which various flash memory vendors have calculated their products' cycling capabilities, and to request additional information if needed. By correctly interpreting not only minimum cycling information but also the failure rates associated with this cycling, you can intelligently compare and choose among the many flash memory offerings in today's market, as they match the requirements of your design.

SUMMARY

The basic concept of the flash memory cell is relatively simple. Again referencing Figure 3.3 as an example, storing electrons on the floating gate changes the stored cell data from a one to a zero, and removing them changes it back to a one. The challenge for flash memory vendors has been to make flash memory:

- Simple, with the smallest possible cell and minimal periphery logic, translating to a small die size and lowest silicon cost,
- Manufacturable, with a technology development approach that can be easily and cheaply moved to the vendor's production line, and
- Feature-set-rich, with technologies and devices that answer the requirements of their target markets.

The flash memory market is still in its infancy. The system designer has a wide range of product offerings from multiple flash memory vendors to choose from, based on several unique technology approaches. In Chapter 2, we've already covered flash memory applications in detail, and discussed the features that are of highest importance in each case. In combination with the information from this chapter, you'll be able to choose the flash memory that makes the most sense for your design!

Index

You are welcome to send us comments or questions concerning this or other Annabooks products, or to request a catalog of our products and seminars.

Annabooks
11848 Bernardo Center Drive, Suite 110
San Diego, CA 92128

616-673-0870

1-800-462-1042

616-673-1432 FAX