Memories in Computers
Part 3: Flash Memories

by

Dr. William R. Huber, P.E.

A. Nonvolatile Memory History

The technological origin of nonvolatile memory was a little-noticed 1967 paper by Dawon Kahng and Simon Sze of Bell Labs. The authors proposed a floating gate structure capable of long-term storage of electrons and of controlling the conducting channel of an MOS transistor. Both a theoretical analysis and an actual fabricated device were reported.[1]

By adding an erase gate to the floating gate transistor of Kahng and Sze, Fujio Masuoka of Toshiba invented flash memory.[2,3,4] In an ironic twist to the stereotype of, "The US invents and Japan produces," Toshiba ignored Masuoka's work and Intel quickly became the world leader in flash memory production.[5] Descendants of Masuoka's pioneer device appear in virtually every computer, MP3 player, camera, camcorder, video game console, thumb drive and other advanced electronic product made today.

What distinguishes flash memory from other forms of nonvolatile memory? The term 'flash' historically had been used to describe a mode of programming or erasing an entire memory array at one time. Although designers soon made flash memories that were erasable by sections, the name stuck and has come to designate those electrically erasable programmable read only memories (EEPROMs) that are made with one-transistor cells rather than the two or more transistor cells of the low-density conventional EEPROM.[6]

Why is flash memory important? Here are the major reasons:

- Low cost—presently about 40% of the cost per bit of DRAM;
- High density—presently 8x as many bits per chip as DRAM;
- Nonvolatility—Unlike DRAM, flash memory retains its stored information when the power is turned off.

---

[1] D. Kahng and S.M. Sze, "A floating-gate and its application to memory devices," The Bell System Technical Journal, vol. 46, no. 6, July-August 1967, pp. 1288-1295

[2] Masuoka, F. and H. Iizuka, "Semiconductor Memory Device and Method for Manufacturing the Same," U. S. Patent 4,531,203; Application date Nov. 13, 1981; FAP date Dec. 20, 1980; Issue date July 23, 1985; NOR-structure flash memory

[3] Masuoka, F. et al., "A New Flash EEPROM Cell Using Triple Polysilicon Technology," IEEE IEDM Technical Digest, 1984, p. 464

[4] Momodomi, M., Masuoka, F., et al., "Electrically Erasable Programmable Read-Only Memory with NAND Cell Structure," U. S. Patent 5,050,125; Application date Nov. 17, 1988; FAP date Nov. 18, 1987; Issue date Sep. 17, 1991

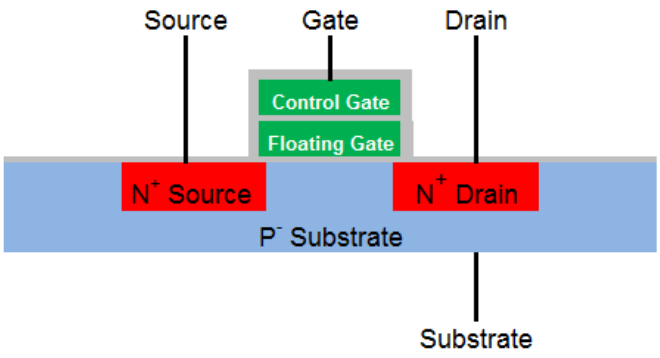
[5] Fulford, B., "Unsung Hero," Forbes.com, June 24, 2002 (www.forbes.com/global/2002/0624/030_print.html)

[6] Prince, B., ***Semiconductor Memories***, Second Edition, ISBN 0 471 92465 2, 1983, John Wiley & Sons, West Sussex, England, page 586

B. Storage Mechanism

As in a DRAM, information in a flash memory is stored as charge (or lack of charge) on a capacitor. In a DRAM, charge leaks off of the capacitor through the access transistor, and thus must be refreshed on the order of every 100 milliseconds. In a flash memory, the charge is stored on a floating conductive layer, usually polycrystalline silicon (polysilicon) that is surrounded by a high quality silicon dioxide insulator. Leakage from a flash floating gate is so small that retention times are on the order of 10 years! Starting with the analogy for DRAM storage of a balloon filled with air that leaks out over time, flash memory storage would be akin to a pressurized steel tank.

The basic floating gate structure is shown below. The green gate regions are typically doped polysilicon; the gray regions between and below the gates are thin silicon dioxide; and the N^+ source and drain are doped by ion implantation into the P^- substrate. Without the floating gate, the structure is a standard NMOS transistor. Thus the floating gate transistor is a simple addition to standard MOS processing. The trick is getting electrons onto and off of the floating gate, which would seem to be electrically isolated.

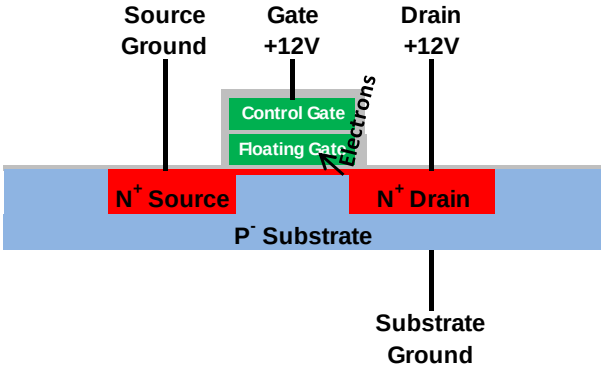


1. Moving Electrons Through an Insulator

The process of moving electrons to the floating gate is called “Programming” the memory cell, and removing electrons from the floating gate is called “Erasing” the memory cell. Programming acts to increase the threshold voltage of the MOS transistor; erasing reduces the threshold voltage to its lowest value.

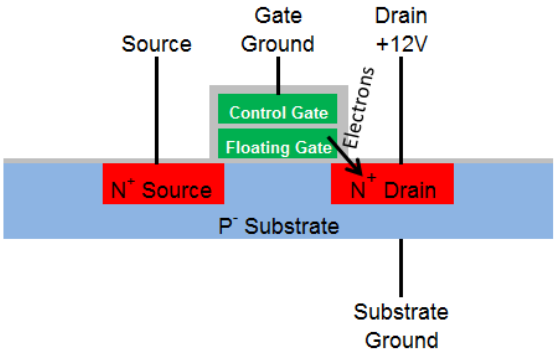
There are two primary methods for getting electrons to move through the gate dielectric: hot electron injection and Fowler-Nordheim tunneling.

a. Hot Electron Injection (HEI)[7,8,9]



In hot electron injection, shown at right, the transistor is turned-on by applying a high positive voltage (e.g., +12V) to the control gate. Then a high source-to-drain current is forced by biasing the drain to a high voltage (e.g., +12V) and grounding the source. This source-drain current is sufficient to cause carrier pair creation due to impact ionization at the drain. Some of the resulting high energy electrons are able to jump over the 3.2 eV energy barrier at the interface between the silicon substrate and the silicon dioxide. The positive control gate voltage attracts the electrons toward the floating gate, where they lodge in trap sites. As the floating gate becomes more negatively charged, additional electrons from the channel are repelled, so HEI is self-limiting. Because of the high source-drain current, the +12V supply is quite difficult to generate on-chip, so an off-chip high voltage supply is usually required.

b. Fowler-Nordheim (F-N) Tunneling[7,10]



With the simple structure at left, F-N tunneling is useful only for erasing, i.e., removing electrons from the floating gate. To erase the cell, the control gate is grounded and the drain is biased to a high positive voltage (the source is usually floating). The high electric field across the gate-dielectric sandwich repels electrons from the floating gate and attracts them to the drain. F-N tunneling is not self-limiting, so over-erasure (so that the transistor becomes depletion mode) is a common problem. Depending on the transistor and array structure (more on array structures

---

[7] Brown, W.D. and J.E. Brewer, Editors; ***Nonvolatile Semiconductor Memory Technology***, ISBN 0-7803-1173-6, 1998, IEEE Press, New York, NY

[8] Cottrell, P.E. et al., "Hot Electron Emission in N-Channel IGFETs," *IEEE Journal of Solid-State Circuits*, Volume SC-14, 1979, page 442

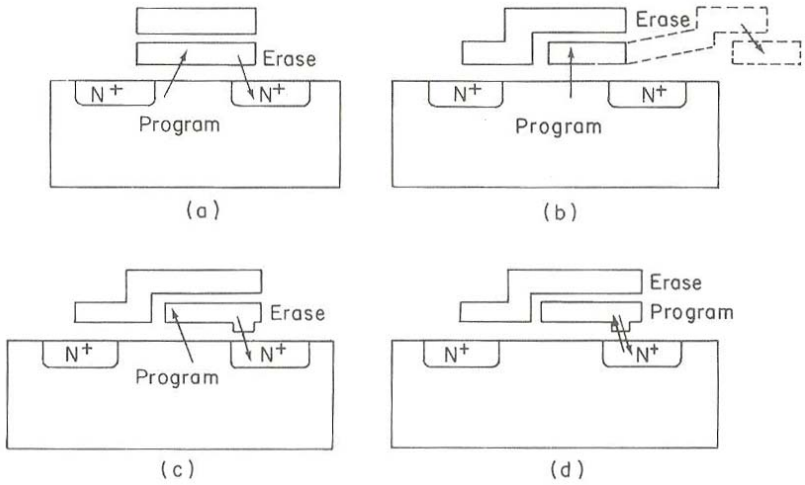
[9] Eitan, B. and D. Frohman-Bentchkowsky, "Hot Electron Injection into the Oxide in N-Channel MOS Devices," *IEEE Transactions on Electron Devices*, Volume ED-23, 1981, page 328

[10] Lenzlinger, M. and E.H. Snow, "Fowler-Nordheim Tunneling in Thermally Grown SiO_2," *Journal of Applied Physics*, Vol. 40, 1969, p. 278

below), F-N tunneling can occur from the floating gate to the source, drain, source and drain, the substrate or even to a dedicated erase electrode.

2. More Complex Structures

The diagrams at the right show the simple structure we have been discussing [(a)], along with three more complex structures.[11] The three more complex structures [(b), (c) and (d)] all include a "split gate" transistor, in which a series transistor directly controlled by the control gate prevents conduction of the floating gate transistor in case it is over-erased. The following table summarizes the important properties of each of these cell structures.



Drawing	(a)	(b)	(c)	(d)
Programming Method	HEI	HEI	HEI	F-N Tunneling from Drain
Erasing Method	F-N Tunneling to Drain	F-N Tunneling to separate poly erase gate	F-N Tunneling to Drain	F-N Tunneling to Drain
Advantages	Simple	Less stress on thin gate oxide		Low current allows on-chip high voltage generation
Disadvantages	HEI could require off-chip high voltage; No protection against over-erasure	HEI could require off-chip high voltage; High voltage required for poly-to-poly erase	HEI could require off-chip high voltage	Programming by F-N Tunneling is slower that HEI

[11] Prince, B., ***Semiconductor Memories***, Second Edition, ISBN 0 471 92465 2, 1983, John Wiley & Sons, West Sussex, England, page 186

3. Endurance and Retention

Nonvolatile memory cells are different than DRAM and Static RAM cells in that nonvolatile cells have a wearout mechanism. While there are no moving parts as in a mechanical system such as a magnetic hard drive, the programming and erase operations, whether HEI or F-N tunneling, involve moving electrons through an insulator. Eventually damage, in the form of carriers (electrons and/or holes) trapped in the thin oxide or in the floating gate, will prevent effective programming or erasing. The number of program and erase cycles a memory cell can tolerate before becoming unusable is called its endurance.

While the physics underlying the wearout of nonvolatile memory cells is beyond the scope of this course,[12] the resulting plot of cell threshold voltage vs. program-erase cycles shows the effect.[13] As the number of program-erase cycles increases, the damage accumulates and the threshold voltage difference between the erased state and the programmed state decreases. When the threshold voltage difference shrinks such that the sensing circuitry can no longer distinguish two distinct states, the cell becomes unusable.

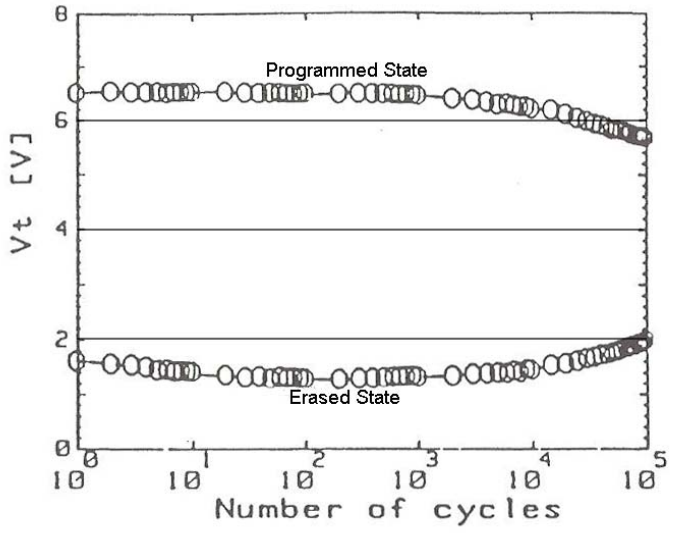


Figure 4.74 Program/erase endurance curve of a typical Flash memory cell [4.104].

This inherent wearout mechanism has impact in many areas. Among these are:

- Because the number of erase and program operations that any given cell has experienced is unpredictable, the success of the next such operation is also unpredictable. So it is standard practice to read data from the flash memory cell immediately after erasing or programming it to verify that the operation has been successful. The data from that cell reading is

[12] Brown, W.D. and J.E. Brewer, Editors; op. cit.; pp. 255-281
[13] Ibid., p. 256; Originally presented by Chang, C.; "Flash Memory Reliability;" Tutorial at International Reliability Physics Symposium; 1993

compared internally to the desired data, and is not conveyed to the memory output terminals as it would be in an actual memory read operation.

- Program and erase algorithms that minimize oxide stress have been developed. For example, erasing and programming might be done in smaller incremental steps with verification after each step. Once the verification is satisfied, erasing or programming is stopped; thus reducing oxide stress.
- Spare regions of memory are provided to replace areas that have received excessive program-erase cycles, as evidenced by failed verification.
- Devices and systems utilize error detection and correction schemes.

These measures and others have pushed endurance to beyond 10^5 cycles, sufficient for all but the most critical applications.

If a memory claims to be nonvolatile, it had better retain data for a long time. Data retention in nonvolatile memory cells has improved from the "longer than one hour" of Kahng and Sze's first floating gate memory device[14] to 10-20 years on current devices. Retention time is influenced by many factors, but the only one under user control is ambient temperature. In general, the lower the temperature (to the minimum allowable temperature), the longer the data retention.

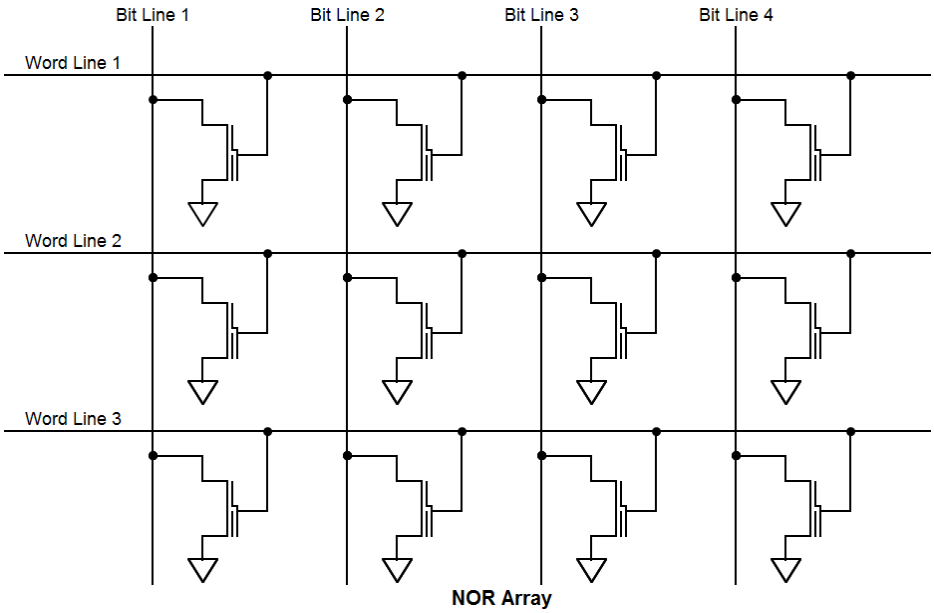
[14] D. Kahng and S.M. Sze, op. cit., page 1295

C. Array Structure—NOR Array

To make a usable device, many memory cells must be placed into one or more arrays; and support circuitry such as address buffers, decoders, sense amplifiers, data input/output circuits and charge pumps (for high-voltage generation) must be added. The most common array structures for flash memory are designated as NOR and NAND. We will examine the NOR array first.

1. NOR Array Structure

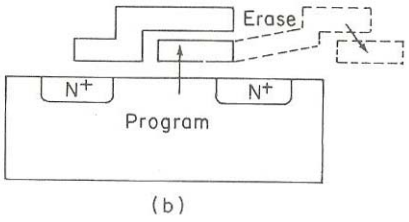
The structure of the NOR array looks very much like that of a DRAM. Every cell is individually addressable, thus allowing quick random access. All memory bits are guaranteed good for the life of the device.



Some NOR array memories use the structure of drawing (b) (on page 5), with a separate polysilicon erase gate. We will assume that structure in the following discussion about erasing, programming and reading this device. The voltage levels mentioned below are consistent with older devices operating at V_{DD} = 5V; lower voltage levels would apply to more modern devices with V_{DD} = 2.5V.

2. Erasing in a NOR Array

With the structure as show at right, erasing (setting all bits to "1") occurs by applying a high positive voltage pulse (12V to 22V) to the erase gate, which is common to a large group of memory cells called a sector. With all other terminals (control gate, source and drain) grounded, electrons are pulled off of the floating gate and tunnel through the inter-polysilicon oxide to the erase gate. An "erase verify" operation insures that all cells are erased properly. If the erase verify fails, additional



voltage pulses of the same or different amplitude and duration (depending on the specific device design) are applied to the erase gate.

3. Programming in a NOR Array

First we have to recognize that erased memory cells have a low threshold voltage, thus are set to a "1". Thus, only those cells intended to store a "0" have to be programmed to a high threshold voltage by moving electrons to their floating gate. To achieve the channel hot electron programming, the selected word line (and its associated control gates) is biased to a boosted voltage of 12V; and the bit lines of those cells to be programmed are pulsed to 7V. With source terminals grounded, a high source-to-drain current will flow, and some of the resulting hot electrons will be injected into the gate oxide and onto the floating gate. Usually a "program verify" operation is performed; and if the desired threshold voltage level (above V_{DD}) has not been achieved, additional pulses of the same or different amplitude and duration will be applied until the desired threshold results.

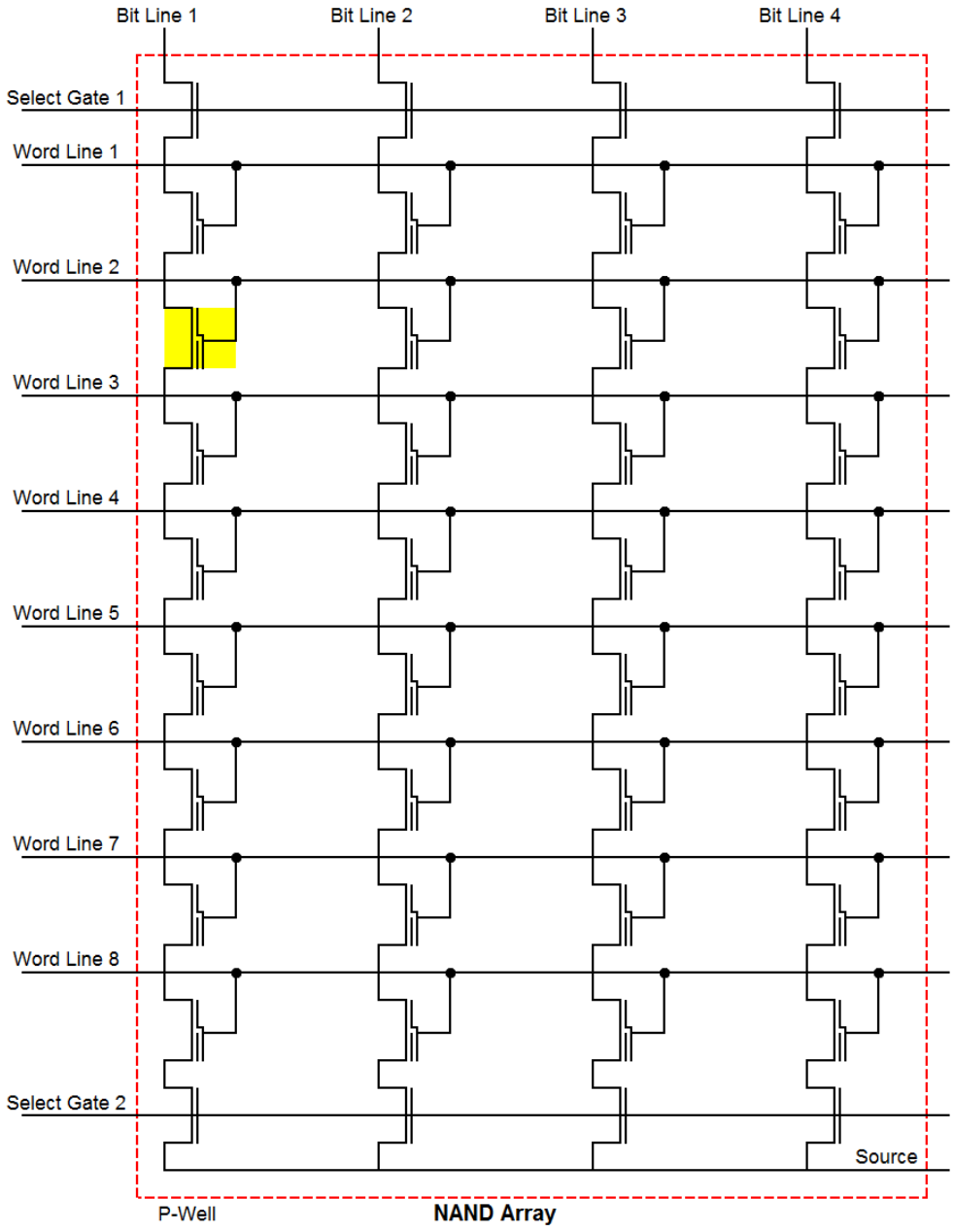
4. Reading in a NOR Array

There are different possibilities for reading the cells in a NOR array. As the readout process is non-destructive, there is no need to restore the information in the cells. Therefore we do not necessarily need a sense amplifier dedicated to each bit line. One approach is to precharge the bit lines to a voltage above the sense point of the sense amplifiers, typically $V_{DD}/2$. Then the selected word line is driven to V_{DD}. Any memory cell storing a "1", with a low threshold voltage, will turn on and discharge its bit line. The associated sense amplifier will detect that low voltage and invert it to output a "1". Any memory cell that was programmed to a high threshold voltage, a "0", will not turn on and will not discharge its bit line; the sense amplifier will detect that high voltage, invert it and output a "0".

D. Array Structure—NAND Array

Let's now examine a NAND array.

1. NAND Array Structure



As shown above, a typical NAND array has 8, 16 or even 32 floating gate memory cells arranged in series between normal NMOS transistors that are driven by Select Gate signals. The control gates along each row of memory cells are connected together to a word line. Because there are no contacts required within the array, the area required for a NAND array is just 40% of that required for a NOR array with the same number of memory cells. At least 98% of bits are good when shipped, but additional bits can fail during the life of the device,[15] so error detection and correction are necessary.

2. Erasing, Programming and Reading in a NAND Array

Because the memory cells are connected in series, the separate sources and drains are not accessible. Therefore, hot electron injection (HEI) cannot be used for programming. Thus, for this NAND array, F-N Tunneling is used for both programming and erasing. Voltage levels mentioned below reflect 5V operation, and would be scaled for more modern 2.5V devices. The following table indicates the voltages required for each operation, assuming that the cell selected for programming (writing) and reading is the one colored yellow, on Bit Line 1 and Word Line 2.

Terminal	Erase	Program	Read
Bit Line 1	Open	0V	2.5V (Precharge) and Sense Amp Input
Other Bit Lines	Open	7V	Open
Select Gate 1	20V	20V	5V
Word Line 2	0V	20V	0V
Other Word Lines	0V	7V	5V
Select Gate 2	20V	0V	5V
Source	Open	0V	0V
P-Well	20V	0V	0V
Substrate	20V	0V	0V

[15] Spansion Application Note , “Flash Memory: An Overview,” November 10, 2005

a. Erasing

With the indicated voltages, all cells will be erased to "1". In fact, all cells on the entire length of the word lines and between the Select Gate devices will be erased. The 20V bias on the P-well will attract electrons from the floating gates via F-N tunneling. The resulting threshold voltage of those erased cells must be negative, so that they will conduct with zero applied gate voltage. As with the NOR array, an "erase verify" operation insures that all cells are erased properly.

b. Programming

With the voltages indicated, only the cell on Bit Line 1 and Word Line 2 will be programmed to a high threshold voltage (storing a "0"). Of course, any or all other cells along the same word line can be programmed at the same time by biasing their bit lines at 7V. Cells in other locations can be programmed with the appropriate voltages. Electrons will travel from the P-well to the floating gate by F-N tunneling. The objective of the programming is to increase the threshold voltage to about $V_{DD}/2$, or 2.5V, so that the programmed cells will be non-conductive with zero control gate voltage and will conduct with V_{DD} (5V) on the word line/control gate. Again, as with the NOR array, a "program verify" operation is performed; and if the desired threshold voltage level ($V_{DD}/2$) has not been achieved, additional pulses of the same or different amplitude and duration will be applied until the desired threshold results.

c. Reading

During reading, all word lines/control gates except for the one containing the cell (or cells) to be read are biased at V_{DD}. Because the threshold of even the programmed cells is $V_{DD}/2$, all of these transistors will be conducting. The selected word line is biased at zero volts. Thus, if the cell to be read is erased (storing a "1"), it will also conduct and the bit line, which had been precharged to 2.5V, will discharge to ground. If the cell to be read is programmed (storing a "0"), it will not conduct and the bit line will remain at its precharged voltage of 2.5V. The sense amplifier, connected to the top of the bit line, will detect either case and output the appropriate result. To read multiple bits along the selected word line, it is only necessary to connect a sense amplifier to each of the desired bit lines.

Note that the bit line discharge path is through 18 transistors (or more, depending on the memory architecture) in series. Thus the time to discharge the bit line (read the selected cell) will be much longer than that for the NOR array where a single transistor discharges the bit line.

E. Comparison of NOR and NAND Arrays—Advantages, Disadvantages, Applications

The NOR architecture is best suited for random access applications such as code execution (e.g., BIOS in computers). NAND is best for sequential access applications such as data storage. The following table compares some of the characteristics of the two architectures.

Characteristic	Symbol	NOR (PC28F128J3)[16]	Symbol	NAND (MT29F2G08A)[17]
Access Method		Random		Sequential
Random READ Access Time	t_{AVQV}	75 ns	t_R	25,000 ns (25 µs)
Sequential READ Access Time	t_{APA}	25 ns	t_{RC}	30 ns
Erase Mechanism		F-N Tunneling		F-N Tunneling
Erase Block Size		128 KB		132 KB
Erase Time per Block	$t_{ERS/AB}$	4 seconds (Max)	t_{BERS}	3 ms (Max)
Programming Mechanism		HEI or F-N Tunneling		F-N Tunneling
Program Page Size		512 Bytes		2112 Bytes
Page Program Time	t_{PROG}	3600 µs (Max)	t_{PROG}	700 µs (Max)
Endurance		10^5 Program/Erase Cycles		10^5 Program/Erase Cycles
Data Retention		Not Specified		10 Years

As the table shows, NOR flash excels in random access performance; while NAND flash has much faster block erase times and faster page program times. The most important factor for most applications doesn't appear in the table: cost per bit. Because the silicon area per bit for NAND is only about 40% of that for NOR, NAND has a very significant density and cost advantage.

[16] Numonyx Embedded Flash Memory (J3 65 nm) Single Bit Cell (SBC) Data Sheet, March 2010
[17] Micron MT29F2G08A NAND Flash Data Sheet, Rev. 1, 1/06 EN

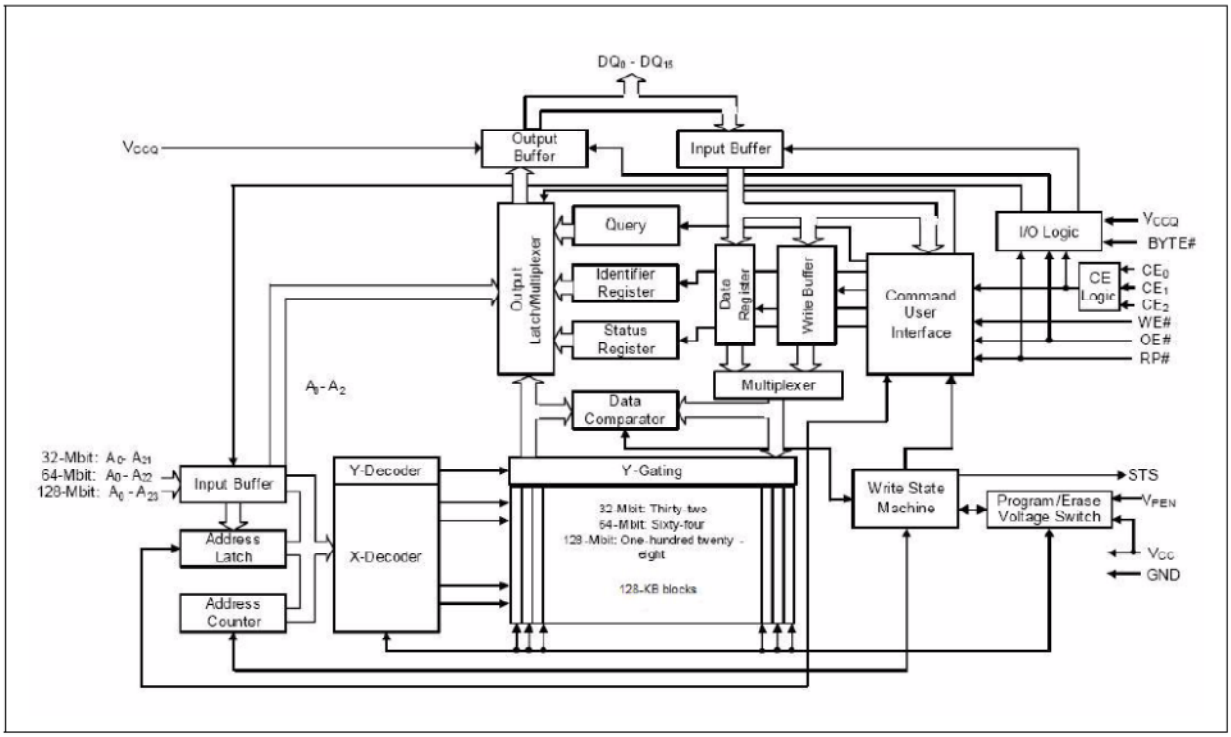
F. Representative Product Operation—NOR Array

We will now examine an actual NOR-array based flash memory to see how it is organized and how it operates. The device is the Numonyx PC28F128J3,[18] 128 Mb flash memory, the basic characteristics of which were highlighted in Section E above. The data sheet provides 66 pages of information about the device; only the most important portions are covered here.

1. Functional Block Diagram, Addressing and Pin Functions

Here is a functional block diagram of the device:

Figure 1: Memory Block Diagram for 32-, 64-, 128-Mbit

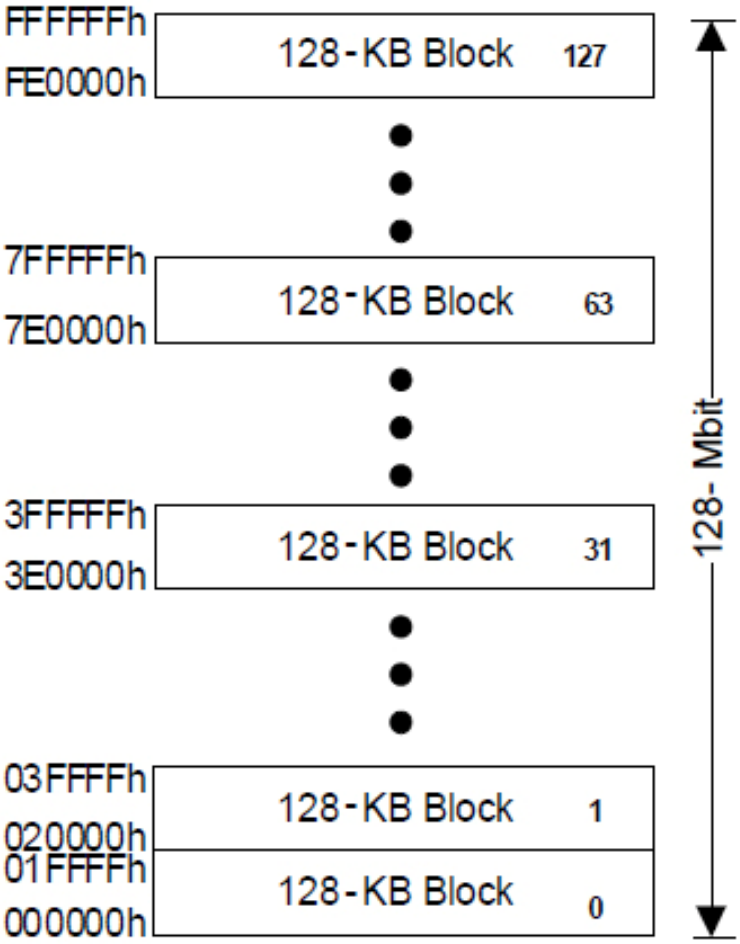


The main memory array consists of 128 blocks, and a block is the smallest erasable unit. Each block contains 128K Bytes, and a Byte is the smallest addressable/programmable unit.

[18] Numonyx Embedded Flash Memory (J3 65nm) Single Bit per Cell (SBC) Flash Data Sheet, March 2010

How do the address inputs map to the memory array? The following memory map provides the details:

Figure 2: J3 65 nm SBC Memory Map



The memory array is byte-addressable. The most significant 7 bits of the address, A[23:17], select the block; and the remaining 17 bits, A[16:0], select the byte within that block. The addresses shown to the left of each block are expressed in hexadecimal notation. Each symbol (0 thru 9, A thru F) represents 4 binary bits. For example, 6h corresponds to 0110 binary (or 6 decimal); and Eh corresponds to 1110 binary or 14 decimal. Appendix A provides conversions among hexadecimal, decimal and binary notations.

The functions of some of the other pins shown on the Functional Block Diagram are as follows.

CE[2:0] are the Chip Enable inputs. When appropriate logic levels are applied to the CE inputs, the device is activated.

DQ[7:0] is the low byte data bus. Data input and output, as well as command inputs are carried on these pins.

DQ[15:8] is the high byte data bus. In the x8 mode we are examining, these pins float and are not used.

RP# is the RESET pin. When low it resets internal circuits and places the device in the power-down mode. When high, it enables normal operation.

OE# is Output Enable. When low, it activates the device outputs during a read operation. When high, the outputs are disabled.

WE# is Write Enable. When low, it enables writing to the memory array, the Command User Interface and the write buffer. When high, no writes can occur.

STS is an open-drain status pin. In its default mode, it acts as a Ready/Busy# (R/B#) pin. The pin must be tied to V_{DD} through a pull-up resistor.

V_{PEN} is Program Enable. When low, it prevents erasing or programming.

The signals (CE, RP#, OE#, WE#, V_{PEN}) control bus operations, as tabulated below:

Table 16: Bus Operations

Mode	RP#	CE_x (1)	OE# (2)	WE# (2)	V_{PEN}	$DQ_{15:0}$ (3)	STS (Default Mode)	Notes
Async., Status, Query and Identifier Reads	V_{IH}	Enabled	V_{IL}	V_{IH}	X	D_{OUT}	High Z	4,6
Output Disable	V_{IH}	Enabled	V_{IH}	V_{IH}	X	High Z	High Z	—
Standby	V_{IH}	Disabled	X	X	X	High Z	High Z	—
Reset/Power-down	V_{IL}	X	X	X	X	High Z	High Z	—
Command Writes	V_{IH}	Enabled	V_{IH}	V_{IL}	X	D_{IN}	High Z	6,7
Array Writes	V_{IH}	Enabled	V_{IH}	V_{IL}	V_{PENH}	X	V_{IL}	5,8

Notes:

1. See Table 17 for valid CE_x configurations.
2. OE# and WE# should never be asserted simultaneously. If done so, OE# overrides WE#.
3. DQ refers to DQ[7:0] when BYTE# is low and DQ[15:0] if BYTE# is high.
4. Refer to DC characteristics. When $V_{PEN} \leq V_{PENLK}$, memory contents can be read but not altered.
5. X should be V_{IL} or V_{IH} for the control pins and V_{PENLK} or V_{PENH} for V_{PEN}. For outputs, X should be V_{OL} or V_{OH}.
6. In default mode, STS is V_{OL} when the WSM is executing internal block erase, program, or a lock-bit configuration algorithm. It is V_{OH} (pulled up by an external pull up resistance ≈ 10k) when the WSM is not busy, in block erase suspend mode (with programming inactive), program suspend mode, or reset power-down mode.
7. See Section 11.0, "Device Command Codes" on page 48 for valid DIN (user commands) during a Write operation.
8. Array writes are either program or erase operations.

2. Device Commands

Many different commands are available for this NOR flash memory, as tabulated below. We will examine the most important ones.

Table 34: Command Bus Cycles and Command Codes

Command		Setup Write Cycle		Confirm Write Cycle	
		Address Bus	Data Bus	Address Bus	Data Bus
Registers	Program Enhanced Configuration Register	Register Data	0060h	Register Data	0004h
	Program OTP Register	Device Address	00C0h	Register Offset	Register Data
	Clear Status Register	Device Address	0050h	—	—
	Program STS Configuration Register	Device Address	00B8h	Device Address	Register Data
Read Modes	Read Array	Device Address	00FFh	—	—
	Read Status Register	Device Address	0070h	—	—
	Read Identifier Codes (Read Device Information)	Device Address	0090h	—	—
	CFI Query	Device Address	0098h	—	—
Program and Erase	Word/Byte Program	Device Address	0040h/ 0010h	Device Address	Array Data
	Buffered Program	Block Address	00E8h	Block Address	00D0h
	Block Erase	Block Address	0020h	Block Address	00D0h
	Program/Erase Suspend	Device Address	00B0h	—	—
	Program/Erase Resume	Device Address	00D0h	—	—
Security	Set Block Lock Bit	Block Address	0060h	Block Address	0001h
	Clear Block Lock Bits	Device Address	0060h	Device Address	00D0h
Blank Check	Blank Check	Block Address	00BCh	Block Address	00D0h

Note that most commands require a Setup Cycle and a Confirm Cycle. This is done to prevent accidental erasure or programming. Because the NOR device has dedicated address pins, addresses and command codes can be loaded into the device simultaneously. All commands load zeros into the highest 8 bits, so command codes are often identified without the two leading zeros.

3. Read Array Command

To perform a read of a random byte in the array, we must provide the address (Block and Byte) on the address pins A[23:0], and the Read Array (FFh) command on the Data Bus I/O[7:0]. This address and command information is input to the NOR device on a Command Write bus operation, so RP# and OE# must be high, WE# must be low and CE must be enabled.

Once the address and command information have been input to the NOR device, the logic states of OE# and WE# are reversed (OE# goes low and WE# goes high); thus the I/O[7:0] pins switch from inputs to outputs. After a brief time (75 ns minimum), valid data from the addressed byte appears on I/O[7:0].

4. Byte Program Command

Suppose we now want to write (program) a random byte in the array.[19] We must provide the address (Block and Byte) on the address pins A[23:0], and the Byte Program (10h) command on the Data Bus I/O[7:0]. This address and command information is input to the NOR device on a Command Write bus operation, so RP# and OE# must be high, WE# must be low and CE must be enabled.

Because we want to write data into the array, a Confirm Cycle is required. For this Confirm Cycle, the same address information is repeated on A[23:0], and the input data is placed on I/O[7:0]. In addition, as this is an Array Write bus operation, V_{PEN} must be high.

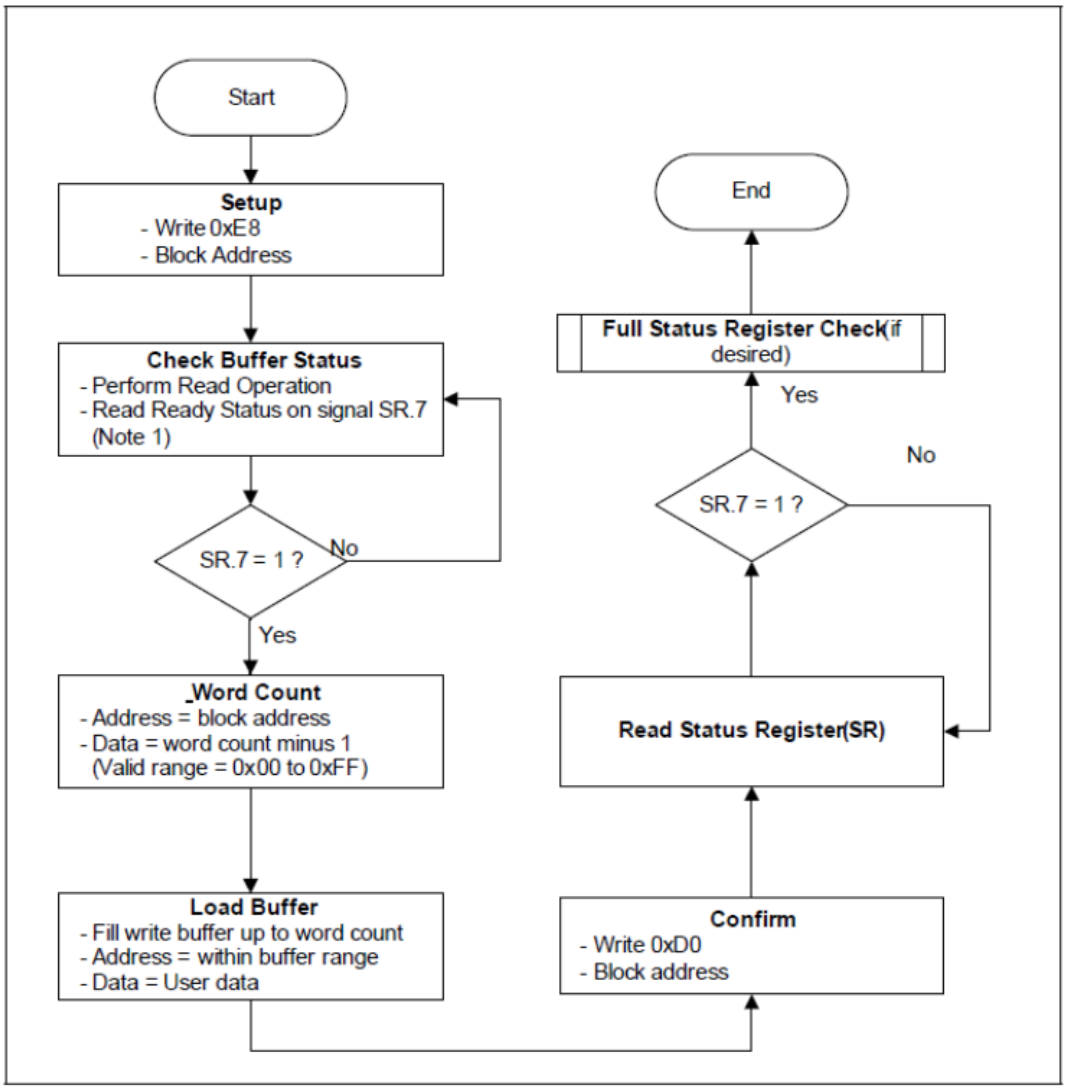
To check on whether the programming succeeded, we then read the Status Register. To execute the command, we must provide the address (Block and Byte) on the address pins A[23:0], and the Read Status Register (70h) command on the Data Bus I/O[7:0]. This address and command information is input to the NOR device on a Command Write bus operation, so RP# and OE# must be high, WE# must be low and CE must be enabled. Once the address and command information have been input to the NOR device, the logic states of OE# and WE# are reversed (OE# goes low and WE# goes high); thus the I/O[7:0] pins switch from inputs to outputs. After a brief time data from the Status Register appears on I/O[7:0]. If SR bit 7 = 1, the byte programming is complete. Other Status Register bits provide more detail about the operation.

[19] Programming converts a stored “1” to a “0.” Only erasing can convert a “0” to a “1,” so erasing usually precedes programming.

5. Buffered Program Command

A more efficient method of programming the array is by use of the Buffered Program command. Through this command, multiple bytes are programmed simultaneously. Data to be programmed is first written into the internal Write Buffer, then programmed into the memory array in buffer-sized increments. The details of how to perform this operation are rather involved and specific to this device, but here is a flow chart to illustrate the procedure:

Figure 16: Write to Buffer Flowchart



Notes:

1. The device defaults to output SR data after the Buffered Programming Setup command (E8h) is issued. CE# or OE# must be toggled to update Status Register. Don't issue the Read SR command (70h), which would be interpreted by the internal state machine as Buffer Word Count.

6. Block Erase Command

For a Block Erase, we must provide the address (Block and any Byte) on the address pins A[23:0], and the Block Erase (20h) command on the Data Bus I/O[7:0]. This address and command information is input to the NOR device on a Command Write bus operation, so RP# and OE# must be high, WE# must be low and CE must be enabled.

Because we want to erase a block in the array, a Confirm Cycle is required. For this Confirm Cycle, the same address information is repeated on A[23:0], and the Confirm Code (D0h) is placed on I/O[7:0]. In addition, as this is an Array Write bus operation, V_{PEN} must be high.

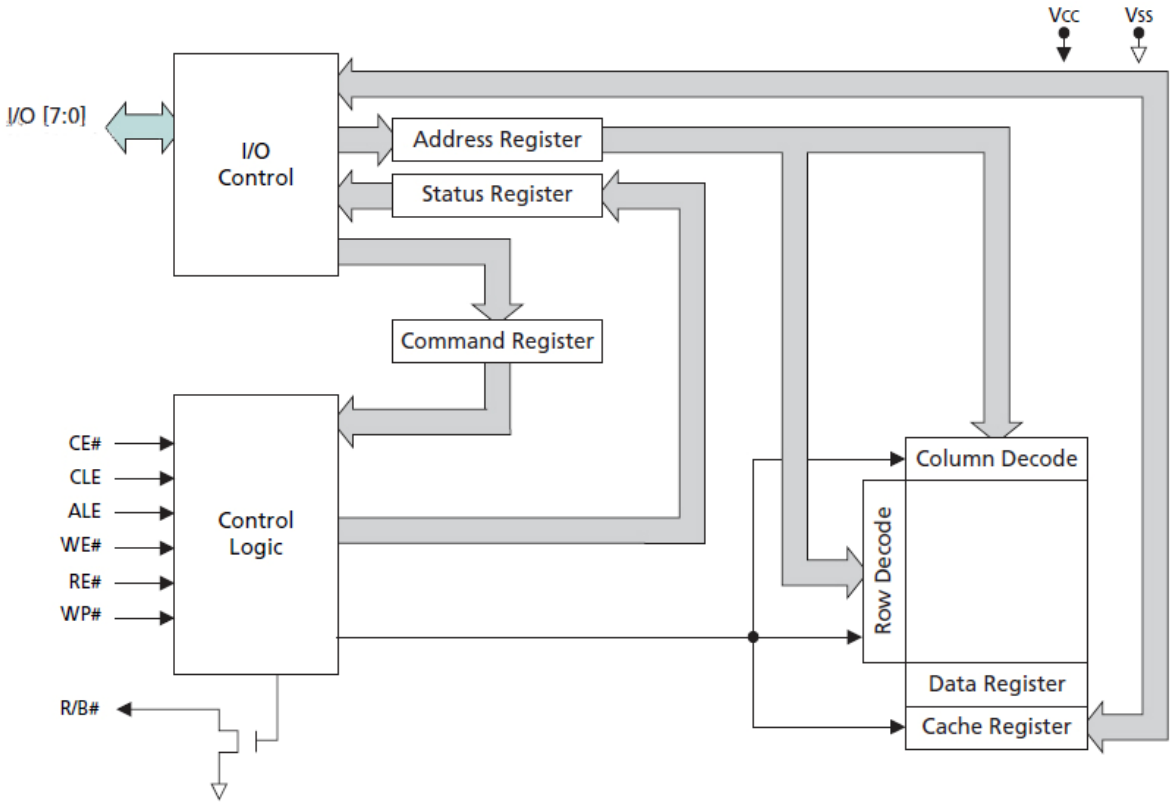
To check on whether the erase succeeded, we then read the Status Register. To execute the command, we must provide the address (Block and any Byte) on the address pins A[23:0], and the Read Status Register (70h) command on the Data Bus I/O[7:0]. This address and command information is input to the NOR device on a Command Write bus operation, so RP# and OE# must be high, WE# must be low and CE must be enabled. Once the address and command information have been input to the NOR device, the logic states of OE# and WE# are reversed (OE# goes low and WE# goes high); thus the I/O[7:0] pins switch from inputs to outputs. After a brief time data from the Status Register appears on I/O[7:0]. If SR bit 7 = 1, the erasing is complete. Other Status Register bits provide more detail about the operation.

G. Representative Product Operation—NAND Array

We will now examine an actual NAND-array based flash memory to see how it is organized and how it operates. The device is the Micron MT29F2G08A,[20] the basic characteristics of which were highlighted in Section E above. The data sheet provides 57 pages of information about the device; only the most important portions are covered here.

1. Functional Block Diagram, Addressing and Pin Functions

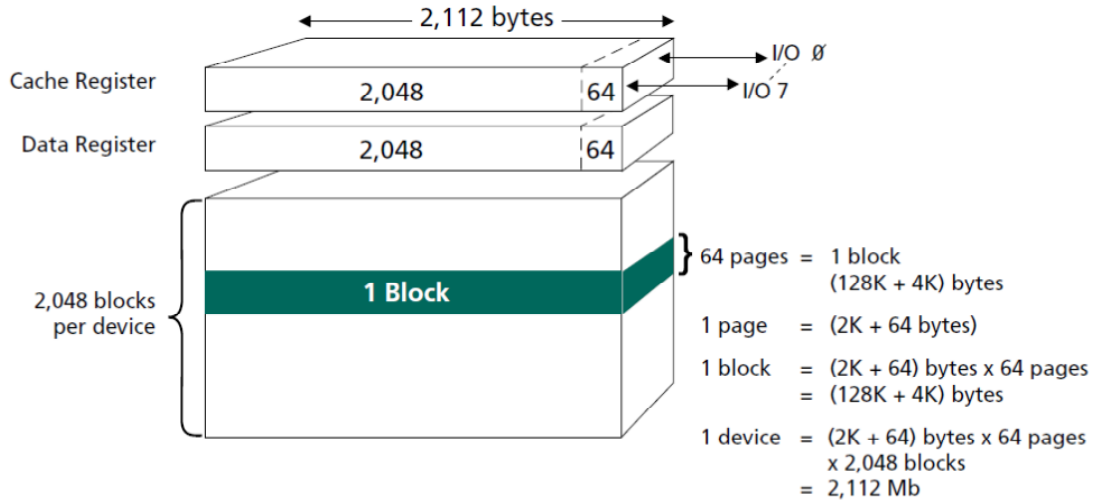
Here is a functional block diagram of the device:



First notice that, unlike DRAMs and the NOR-based device, there are no dedicated address pins on this NAND device. Addresses, Data In and Data Out (as well as commands) are all multiplexed on the pins labeled I/O [7:0]. Addressing is best understood by looking at the internal organization of the memory array, as shown here:

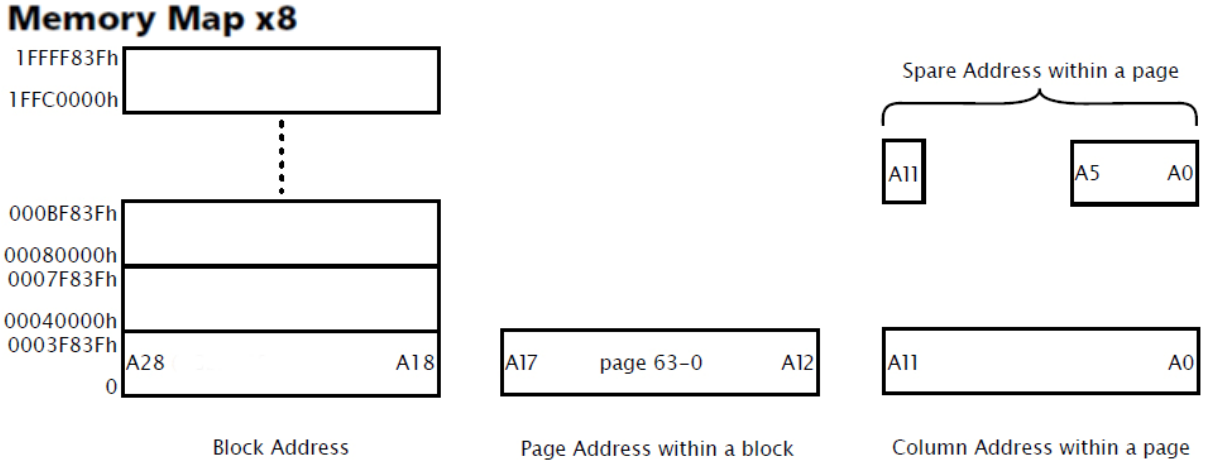
[20] Micron MT29F2G08A NAND Flash Data Sheet, Rev. 1, 1/06 EN

Figure 7: Array Organization for MT29F2G08AxB (x8)



The Cache Register and Data Register are specialized memory arrays[21] used in the reading and programming of the main memory. The main memory array consists of 2,048 blocks, and a block is the smallest erasable unit. Each block contains 64 pages; and each page contains 2K bytes plus 64 "extra" bytes that can be used for error management functions such as bad byte address storage. A page is the smallest unit that can be programmed, although some devices offer special options to program smaller storage units.

How do the address inputs map to the memory array? The following memory map provides the details:



[21] Although the data sheet does not say so, the Cache Register and Data Register are probably static RAMs.

As shown on the Memory Map above, the Block is selected by addresses A18-A28 (11 addresses; 2^{11} = 2048). The page is selected by addresses A12-A17 (6 addresses; 2^6 = 64); and the column within a page is selected by addresses A0-A11 (12 addresses; 2^{12} = 4096).

Table 2: Array Addressing: MT29F2G08AxB

Cycle	I/O7	I/O6	I/O5	I/O4	I/O3	I/O2	I/O1	I/O0
First	CA7	CA6	CA5	CA4	CA3	CA2	CA1	CA0
Second	LOW	LOW	LOW	LOW	CA11	CA10	CA9	CA8
Third	RA19	RA18	RA17	RA16	RA15	RA14	RA13	RA12
Fourth	RA27	RA26	RA25	RA24	RA23	RA22	RA21	RA20
Fifth	LOW	LOW	LOW	LOW	LOW	LOW	LOW	RA28

Note: CAx = column address; RAx = row address.

As shown in the Array Addressing table above, the addresses are input via I/O [7:0] on 5 successive clock cycles; CA0 –CA7 on cycle 1; CA8-CA11 on cycle 2; RA12-RA19 on cycle 3; RA20-RA27 on cycle 4; and RA28 on cycle 5.

The functions of the other pins shown on the Functional Block Diagram are as follows.

CE# is the Chip Enable input (active low). When CE# is low, transfers can occur between the system and the NAND device.

CLE is the Command Latch Enable. When CLE is high, information is transferred from the I/O [7:0] pins to the Command Register. Otherwise it must be driven low.

ALE is the Address Latch Enable. When ALE is high, information is transferred from the I/O [7:0] to the Address Register. Otherwise it must be driven low.

WE# is Write Enable, and when low, gates transfers from the system to the NAND device.

RE# is Read Enable, and when low, gates transfers from the NAND device to the system.

WP# is Write Protect. All PROGRAM and ERASE operations are disabled when WP# is low.

The R/B# (Ready/Busy#) output of the device indicates the status of internal operations. When that signal is high (the pin must be tied to V_{DD} through a pull-up resistor), the device is ready to accept an external command. R/B# is normally tied to an interrupt pin on the system controller.

The six signals entering the Control Logic box in the functional block diagram on page 21 (CE#, CLE, ALE, WE#, RE#, WP#) control the device mode, as tabulated below:

Table 6: Mode Selection

CLE	ALE	CE#	WE#	RE#	WP#[1]	PRE[2]	Mode	
H	L	L	⌐↑	H	X	X	Read mode	Command input
L	H	L	⌐↑	H	X	X		Address input
H	L	L	⌐↑	H	H	X	Write mode	Command input
L	H	L	⌐↑	H	H	X		Address input
L	L	L	⌐↑	H	H	X	Data input	
L	L	L	H	↓⌐	X	X	Sequential read and data output	
L	L	L	H	H	X	X	During read (busy)	
X	X	X	X	X	H	X	During program (busy)	
X	X	X	X	X	H	X	During erase (busy)	
X	X	X	X	X	L	X	Write protect	
X	X	H	X	X	0V/VCC	0V/VCC	Standby	

Notes: 1. WP# should be biased to CMOS HIGH or LOW for standby.
2. PRE should be tied to VCC or ground. Do not transition PRE during device operations. The PRE function is not supported on extended-temperature devices.
3. Mode selection settings for this table: H = Logic level HIGH; L = Logic level LOW; X = VIH or VIL.

2. Command Sequence

Operations of the memory are controlled by a Command Sequence, consisting of:

- Command cycle 1;
- Address (5 or 3 cycles);
- Command cycle 2 (if required);
- Data cycles (either READ or WRITE).

The available commands are tabulated here:

Table 7: Command Set

Operation	Cycle 1	Cycle 2	Valid During Busy
PAGE READ	00h	30h	No
PAGE READ CACHE MODE START[1]	31h	–	No
PAGE READ CACHE MODE START LAST[1]	3Fh	–	No
READ for INTERNAL DATA MOVE[2]	00h	35h	No
RANDOM DATA READ[3]	05h	E0h	No
READ ID	90h	–	No
READ STATUS	70h	–	Yes
PROGRAM PAGE	80h	10h	No
PROGRAM PAGE CACHE[1]	80h	15h	No
PROGRAM for INTERNAL DATA MOVE[2]	85h	10h	No
RANDOM DATA INPUT for PROGRAM[4]	85h	–	No
BLOCK ERASE	60h	D0h	No
RESET	FFh	–	Yes

Notes: 1. Do not cross die address boundaries when using cache operations. See Tables 4 and 5 for definition of die address boundaries.
2. Do not cross die address boundaries when using READ for INTERNAL DATA MOVE and PROGRAM FOR INTERNAL DATA MOVE. See Tables 4 and 5 for definition of die address boundaries.
3. RANDOM DATA READ command limited to use within a single page.
4. RANDOM DATA INPUT for PROGRAM command limited to use within a single page.

Commands, which appear on the I/O [7:0] pins, are written to the Command Register when:

- CE# and ALE are LOW, and
- CLE is HIGH, and
- The device is not busy, and
- WE# transitions LOW to HIGH.

3. READ Operations

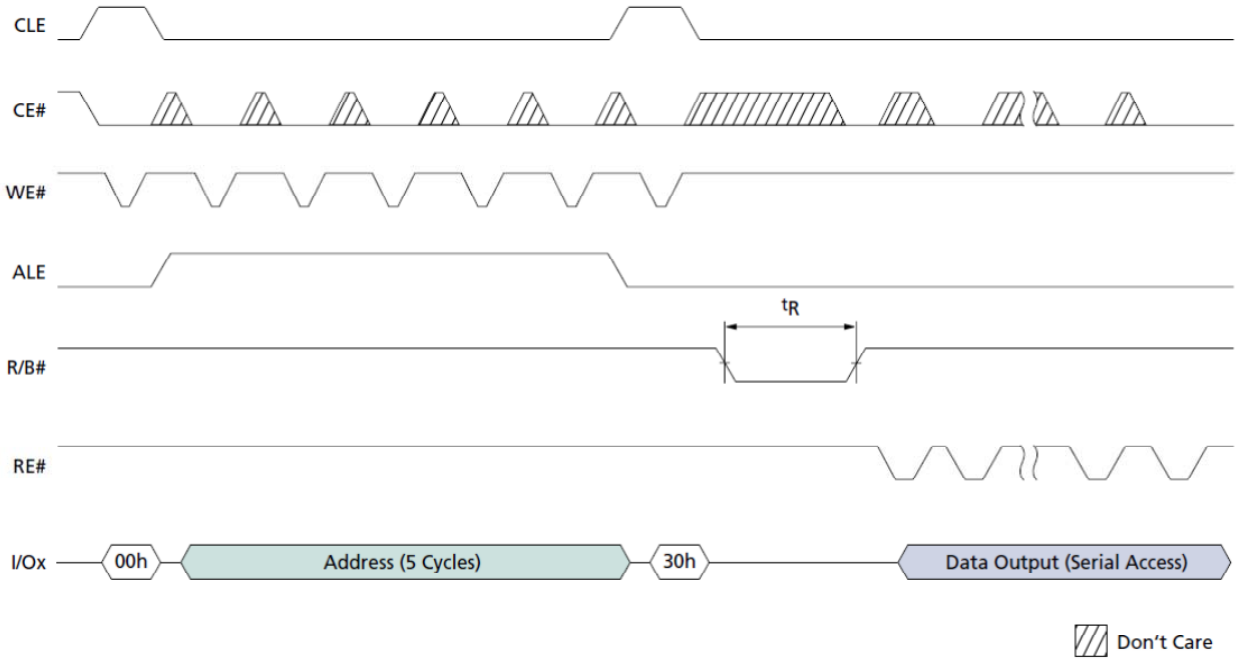
For a PAGE READ operation, the command input 00h[22] (I/O [7:0] pins all LOW) would be written to the Command Register. Then the address inputs would follow on the next 5 cycles. Then the command input 30h (I/O [7:0] pins set to 0-0-1-1-0-0-0-0) would be written to the command register. Finally RE# would cycle to read out sequential bytes of data starting at the initial column address and ending at the end of the page.

[22] The suffix "h" indicates that the preceding numbers (and/or letters) are in hexadecimal notation. The first "0" describes the contents of I/O pins 7 to 4 and the second "0" describes the contents of I/O pins 3 to 0. Appendix A provides conversions among hexadecimal, decimal and binary notations.

The internal memory is accessed on a page basis. When a PAGE READ operation is executed, a full page of data is copied from the memory array to the Data Register (see functional block diagram m on page 21). This transfer to the Data Register accounts for the very long random access time (25 µs) to the first byte of data. Once that copying is complete, the data is output sequentially, byte-by-byte, from the Data Register at a rate of about 30 ns per byte.

Here are the waveforms corresponding to the PAGE READ operation:

Figure 18: PAGE READ Operation



In order to increase device speed, a Cache Register is included (see functional block diagram on page 21). For reads of sequential pages in a block, the Cache Register is used as follows:

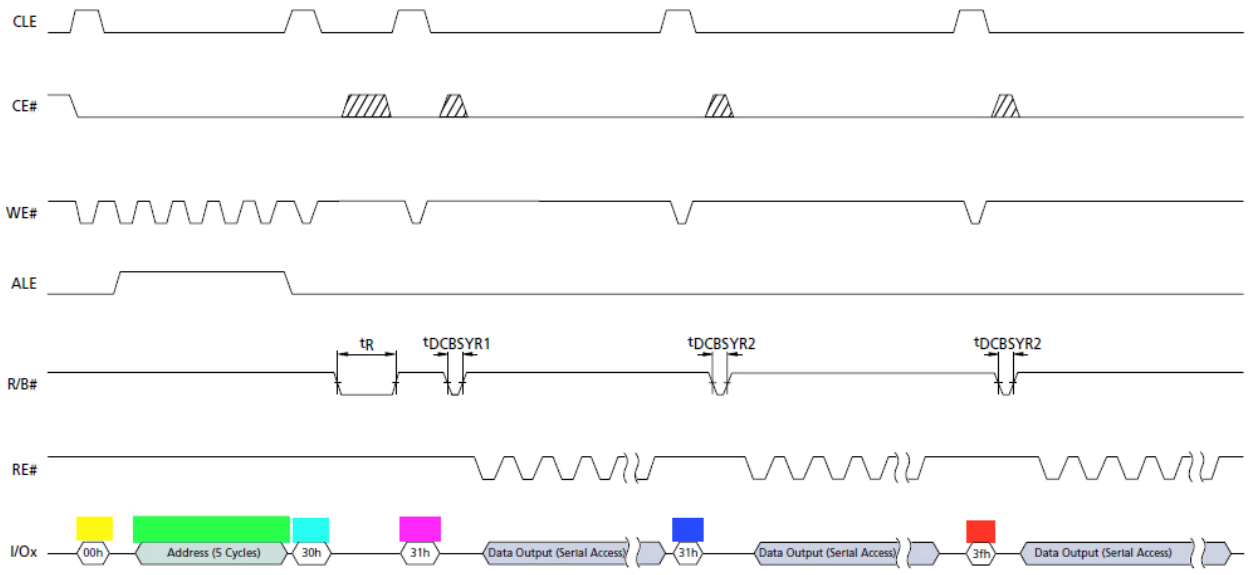
- A PAGE READ (00h-30h)[23] command (as described above), including address information, is issued. The R/B# pin goes low, indicating that the memory is busy.
- After the data is transferred to the Data Register, R/B# goes high and a PAGE READ CACHE MODE START (31h) command can be issued. This

[23] Color-coding matches the waveforms shown on the next page

causes the data from the Data Register (which came from the memory page to be read) to be transferred again, this time to the Cache Register.

- While data is being output from the Cache Register by pulsing RE#, the next sequential page of data is being loaded into the Data Register.
- The second page (and subsequent pages if desired) of data is transferred from the Data Register to the Cache Register by issuing additional PAGE READ CACHE MODE START (31h) commands.
- To read out the last page of data, a PAGE MODE CACHE READ START LAST (3Fh) command is issued. This command causes transfer of the last desired page of data from the Data Register to the Cache Register so it can be read out by pulsing RE#.

The waveforms for the PAGE READ CACHE MODE are shown below.



4. PROGRAM Operations

Similar to the PAGE READ operation, the memory array is also programmed on a page basis by the PROGRAM PAGE command. After the block, page and starting column address are loaded into the internal Address Register, data is sequentially written to the Data Register, up to the end of the page. Then the data is programmed into the selected destination page.

Within a block, pages must be programmed from the least significant bit (LSB) page of the block to the most significant bit (MSB) page of the block.

Here are the steps for the PROGRAM PAGE (80h-10h) operation:

- Issue the SERIAL DATA INPUT (80h) command into the command register (CLE, RE# and WP# high, ALE and CE# low, and WE# low-to-high transition);
- Load the address (block, page and starting column address) into the Address Register using 5 cycles (ALE, RE# and WP# high, CLE and CE# low, and WE# cycling 5 times);
- Input the data to be programmed, starting at the selected address, byte-by byte (RE# and WP# high, CLE, ALE, and CE# low and WE# cycling);
- After all data is loaded, issue the PROGRAM (10h) command into the command register (CLE, RE# and WP# high, ALE and CE# low, and WE# low-to-high transition). The internal program circuits automatically execute the proper algorithm and control all the necessary timing to program and verify the operation. Write verification detects any "ones" that are not successfully written to "zeros."
- At the end of programming, R/B# goes high. Then issue the READ STATUS REGISTER (70h) command to determine if the programming was successful. Bit 0 of the Status Register (I/O pin 0) reports the results of programming. If I/O 0 = 0, the programming was successful. If I/O 0 = 1, the programming had an error.

Other programming operations, including Cache Register-based operations, are available.

5. ERASE Operation

Erasing occurs at the block level, one block at a time. Here are the steps for the BLOCK ERASE (60h-D0h) operation:

- Issue the ERASE SETUP (60h) command;
- Load the address (block and page) into the Address Register using 3 cycles. Although the page addresses A[17:12] are loaded, they are ignored.
- Issue the ERASE CONFIRM (D0h) command. The internal erase circuits automatically control the timing and erase-verify operations.
- At the end of erasing, R/B# goes high. Then issue the READ STATUS REGISTER (70h) command to determine if the erasing was successful. If bit 0 of the Status Register (I/O pin 0) = 0, the erasing was successful. If I/O 0 = 1, the erasing had an error.

H. On-Chip High Voltage Generation

Both of the flash memories discussed in the previous two sections operate from a single power supply with voltage between 2.7V and 3.6V. Yet they both require much higher internal voltages for erasing and programming. Where do these higher voltages come from? All single-power supply flash memories, and virtually all DRAMs, use on-chip circuits called charge pumps to provide boosted DC voltages.

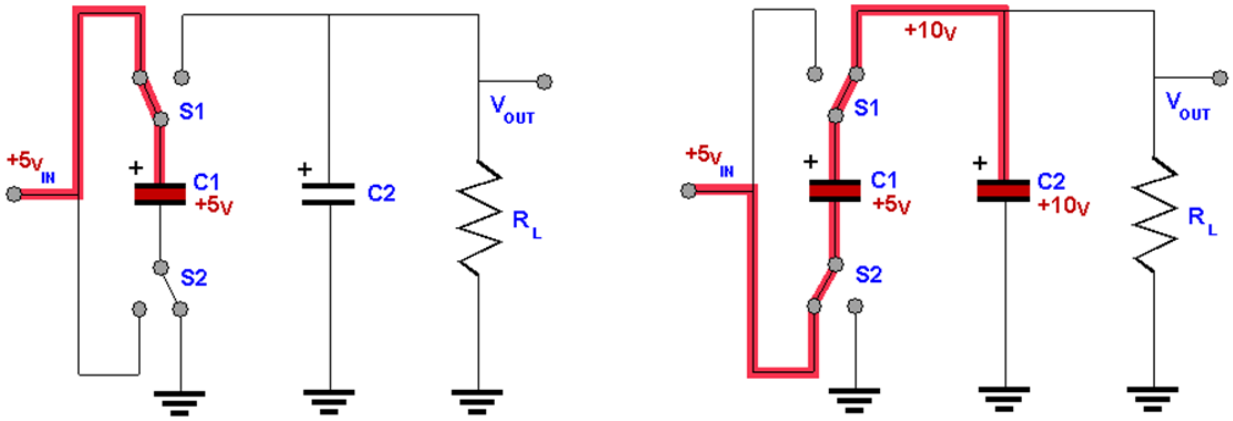
Obviously any DC voltage between the power supply voltage and ground can easily be derived from a resistive divider with or without a regulator circuit. But obtaining voltages above the power supply voltage, or of an opposite polarity, requires a more creative approach. Charge pumps provide that approach, and they have been around since 1919 when Heinrich Greinacher, a Swiss physicist, first proposed this kind of circuit.[24] In 1932, Cockcroft and Walton used the Greinacher circuit design to power their particle accelerator, performing the first artificial nuclear disintegration in history.[25] The voltage produced by Cockcroft and Walton's circuit was 800,000V; a bit higher than we are interested in.

In 1976, with the advent of integrated circuits and in particular digital watches powered by single cell batteries, Dickson proposed an integrated voltage multiplier circuit.[26] We will look first at how a charge pump works, and then come back to Dickson's circuit.

[24] http://en.wikipedia.org/wiki/Greinacher_multiplier
[25] Cockcroft, J.D. and E.T. Walton, "Production of high velocity positive ions," Proceedings of the Royal Society, A, Vol. 136, 1932, pp. 619-630
[26] Dickson, J. "On-chip High-Voltage Generation in NMOS Integrated Circuits Using an Improved Voltage Multiplier Technique" IEEE Journal of Solid-State Circuits, Vol. 11, No. 6, June 1976, pp. 374–378

Charge pumps involve capacitors which are connected in different ways during charging and discharging. The diagram below left shows the charging of capacitor C1.[27] Then switches S1 and S2 (which are actually implemented by MOS transistors) change connections, as shown at below right, with the following results:

- The voltage at the bottom of capacitor C1 immediately rises from ground to 5V;
- Because the voltage across a capacitor cannot change instantaneously (that would take infinite current; i = C dv/dt), the voltage at the top of capacitor C1 rises to 10V;
- When capacitors C1 and C2 are connected via switch S1, they share the charge stored on C1. The resulting voltage at the load, represented by resistor R_L, will rise.
- After several cycles from the left configuration to the right configuration, and assuming R_L is very large, the output voltage will approach 10V. More stages can be added to achieve higher voltages. Practical charge pump circuits can achieve conversion efficiencies up to 90% or so.



[27] From http://www.answers.com/topic/charge-pump

Dickson's circuit, with MOS transistors implementing the switches, is shown here:[28]

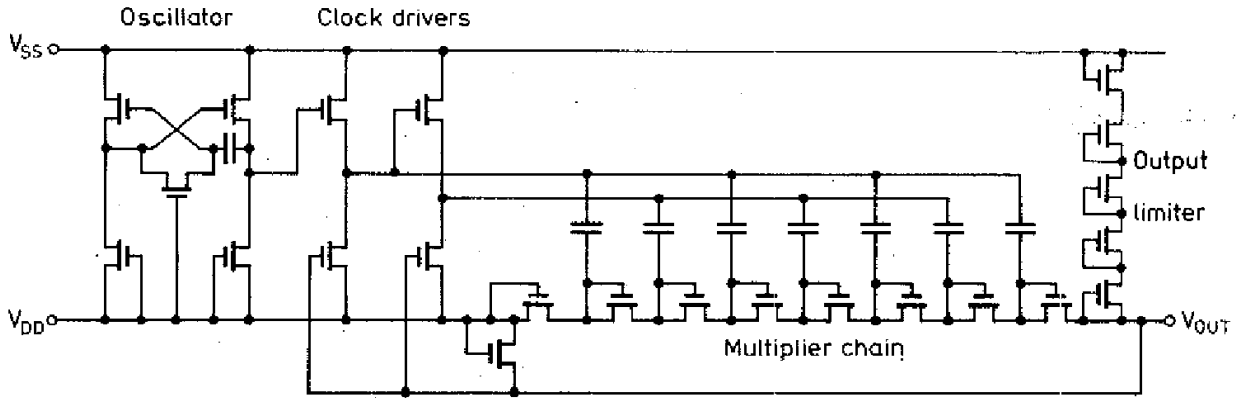


Fig. 5. Practical implementation of multiplier using MOS technology.

A common factor in all charge pump circuits is that the DC supply voltage has to be converted to AC, which is then used to drive either the switches (as in the circuit on p. 29) or the capacitors (as in Dickson's circuit above). Dickson uses a single stage oscillator to accomplish the DC-AC conversion; most modern devices use a ring oscillator (an odd-stage inverter loop).

Most integrated charge pump circuits have very low driving capability. That is the characteristic that limits their capability to support hot electron injection in flash memories. The following table from Dickson's paper illustrates this limitation:[29]

MULTIPLIER OUTPUT AND RIPPLE VOLTAGES AS A FUNCTION OF LOAD CURRENT

Load Current I_{OUT} (μA)	Output Voltage V_{OUT} (volts)		P – P Ripple Voltage V_R (volts)	
	Predicted	Measured	Predicted	Measured
3.9	44.3	39.1	0.29	0.28
5.4	39.5	37.5	0.31	0.30
6.6	35.7	33.3	0.32	0.32
8.2	30.6	27.2	0.34	0.34

Dickson's charge pump boosts the 14V power supply voltage to about 35V, but it can supply less than 10 µa of current to the load.

[28] Dickson, op. cit., p. 377
[29] Ibid., p. 378

I. Multi-Level Storage

From the first days of integrated circuit development in 1958, one primary objective has been to increase the density (number of transistors/bits/functions per chip) of the devices. This increase in density was observed by Gordon Moore, the co-founder of Intel, in 1965. Moore observed[30] that the number of transistors per square inch of silicon had doubled every year since the integrated circuit was invented, and predicted that such doubling would continue for the next 10 years. His observation was later dubbed "Moore's Law." The density increase has slowed a bit, to a doubling every 18 months, but continues to this day. In fact, future roadmaps of semiconductor innovation are based on the continued validity of Moore's Law.

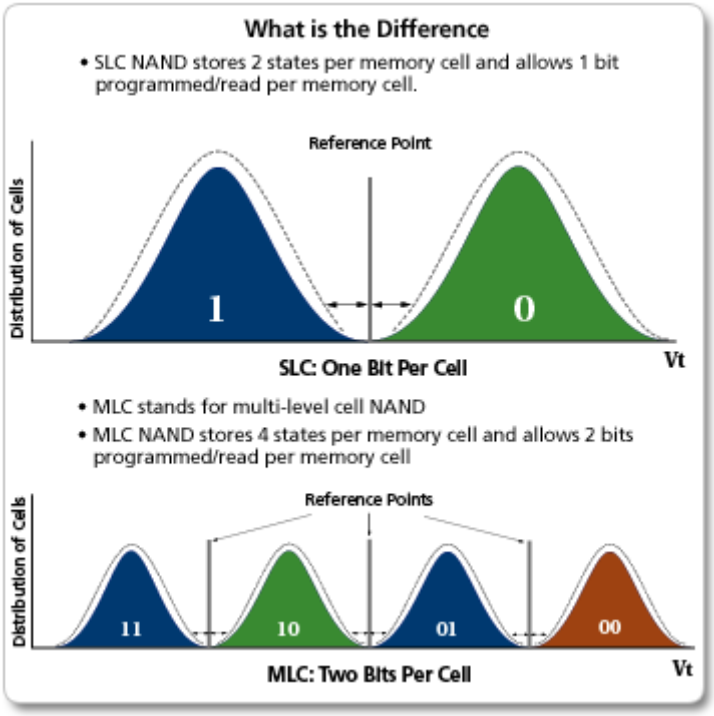
Flash memories follow, and indeed have extended Moore's Law. The devices we have discussed thus far store one bit per memory cell and achieve densities up to about 4 GB (32×10^9 bits) per chip. Devices using single bit per cell are abbreviated as either SBC or SLC (for Single Level Cell).

Because the charge on the floating gate of a flash memory doesn't leak off, circuit designers have been able to design circuits that enable storage of more than one bit per memory cell. Devices using this Multi-Level Cell (MLC) (4 levels yielding 2 bits-per-cell) technology now are available with capacities up to 8 GB (64×10^9 bits) per chip.

Designs with even more bits per cell are possible. At the February 2011 International Solid State Circuits Conference, Samsung discussed an 8 level, 3 bit-per-cell device with a capacity of 8GB (64×10^9 bits). The downside of resolving so many levels with a single cell is reduced endurance. Instead of 100,000 erase/program cycles common on an SLC device, 2 bit-per-cell and 3 bit-per-cell devices might achieve as few as 3,000 cycles.

[30] Moore, G., *Electronics Magazine,* "Cramming more components onto integrated circuits," April 19, 1965

How does the MLC technology work? Let's compare SLC and MLC cell storage:[31]



In a single-level cell, there are two possible states per cell: either "1" (low threshold voltage; erased) or "0" (high threshold voltage; programmed). The reference point, used by the sense amplifier to distinguish between the two states, is set at a voltage between the two states.

In a multi-level cell, there are four possible states per cell:

- "11" with very low threshold voltage, generally the erased state;
- "10" with moderately low threshold voltage;
- "01" with moderately high threshold voltage; and
- "00" with the highest threshold voltage.

The "10," "01" and "00" states all result from precision programming operations. Three different reference points, between pairs of the four threshold voltage distributions, are used by three different sense amplifiers to distinguish among the four states.

[31] http://www.micron.com/products/nand_flash/nandcom.html#mlcnand

The table below compares some characteristics of SLC and MLC NAND flash devices:[32]

Table 11: MLC vs. SLC

Symbol		MLC NAND 3.3V (x8)			SLC NAND 3.3V (x16/x8)			Units
		Min	Typ	Max	Min	Typ	Max	
^{t}PROG	Time to transfer contents of data register to the NAND Flash array	–	900	2200	–	220	600	µs
NOP	Number of partial-page programs supported per page before an ERASE is required	–	–	1	–	–	4	Cycles
^{t}R	Time to transfer contents of one page in the NAND Flash array to the data register	–	–	50	–	–	25	µs
Endurance with ECC and invalid block marking		5K	–	–	100K	–	–	PROGRAM/ ERASE cycles
MIN ECC required		12	–	–	1	–	–	Correctable bits per 512 bytes

As expected, the precise programming to achieve four distinct threshold voltage levels takes longer, 900 µs for MLC vs. 220 µs for SLC. A more serious limitation is the endurance, or number of program/erase cycles—5,000 for MLC vs. 100,000 for SLC. Therefore, MLC is typically used in low-performance consumer applications such as media players, cell phones and flash memory cards (USB, SD/MMC and CF—more about flash memory cards in a later section). Finally, the error rate of MLC flash is much higher than SLC; so much more robust error detection and correction is required for MLC applications.

J. Error Correction Coding (ECC)

The details of error correction coding (ECC) are beyond the scope of this course, but some simple examples will illustrate the concept. All error detection and correction methods rely on adding bits to the data transmission. The sole purpose of these extra bits is to monitor undesired changes in the bits carrying actual data. Of course the extra bits are also subject to error, and such extra bit errors are also detected (and potentially corrected) by ECC.

1. Parity Bit

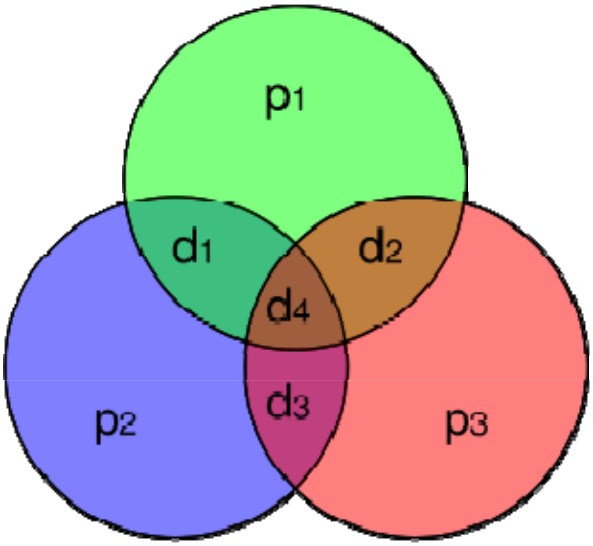
The simplest form of single bit error detection (not correction) is the parity bit. By adding one bit to a data transmission, it is possible to detect one defective bit. For example, suppose we are transmitting four bits and using “even” parity. If the four bits are 0-0-1-0, the parity bit (to bring the total number of “ones” transmitted to an even

[32] Micron Technical Note TN-29-19, “NAND Flash 101,” Rev. B, 4/10 EN, page 24

number) will be a "1." So the transmission becomes 0-0-1-0-1, with the final "1" being the parity bit. Now suppose a single bit error occurs in bit 2 during the transmission, and the received data stream is 0-1-1-0-1. That received data stream has odd parity (three "1's"), thus indicating that an error occurred. But we don't know which bit is in error, because all single bit errors give the same result—"bad parity."

2. Hamming Code

If more "extra" bits are included with a message, and if those bits can be arranged such that different bad bits produce different error results, then bad bits could be identified and corrected. This is the basis of the Hamming Code, invented in 1950 by Richard W. Hamming of Bell Labs. Hamming Code adds three check bits to every four data bits of the data being transmitted. This structure is denoted as (7,4), and can correct any single-bit error, or detect all single-bit and two-bit errors. The goal of Hamming codes in general is to create a set of parity bits that overlap such that a single-bit error in a data bit or a parity bit can be detected and corrected. Here is a Venn diagram pictorial representation of the (7,4) Hamming Code:[33]



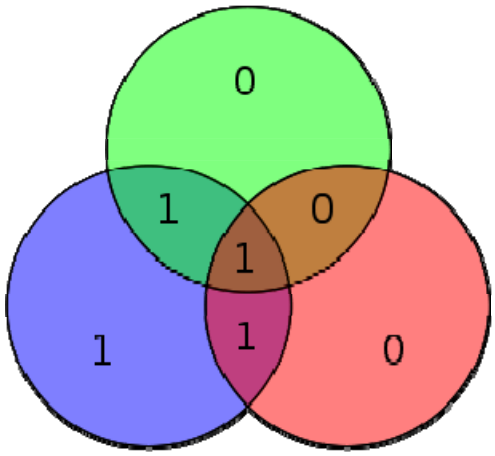
Data bits are denoted by d_1, d_2, d_3 and d_4. Parity bits are denoted by p_1, p_2 and p_3. The following table shows which data bits are covered by which parity bits:

	d_1	d_2	d_3	d_4
p_1	Yes	Yes	No	Yes
p_2	Yes	No	Yes	Yes
p_3	No	Yes	Yes	Yes

The order of bits in the transmission is p_1 p_2 d_1 p_3 d_2 d_3 d_4.

Matrix arithmetic is involved in the actual implementation of Hamming codes, but we will work through an example using the Venn diagram approach. For the example, assume that d_1 d_2 d_3 d_4 = 1 0 1 1. Placing these data bits into the Venn diagram above, and choosing parity bits to provide even parity yields the following diagram:

[33] From Wikipedia, http://en.wikipedia.org/wiki/Hamming(7,4)

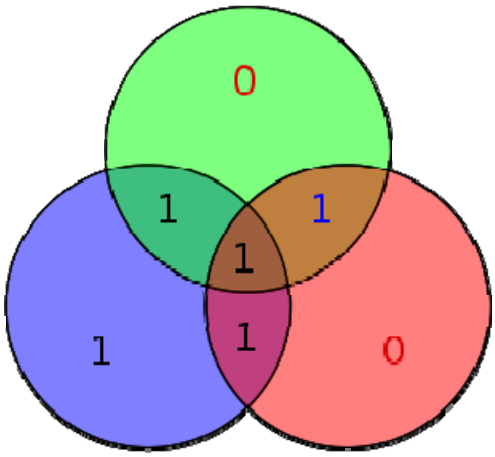


Observe that we have achieved even parity:

- The red circle has two 1's
- The green circle has two 1's
- The blue circle has four 1's

The sent information would be 0 1 1 0 **0** 1 1.

Now assume that a transmission error has caused the fifth bit (bold and underlined in the sent information above) to flip to a 1. Thus, the received information is 0 1 1 0 1 1 1. The corresponding Venn diagram is:



Bit 5, the bad bit, is shown in blue type. This bad bit results in parity errors (shown in red type) in the red and green circles; i.e., both of these circles have three 1's, or odd parity instead of the desired even parity. If bad parity is detected, then the data bit that overlaps *only* the bad parity circles is the bit with the error. It is now a simple matter to correct the bad bit by inversion, and to extract the original data bits.

3. ECC in Flash Memories[34]

As discussed, NAND flash requires ECC to ensure data integrity. ECC has been used for many years in RAM modules as well as in many other types of storage. The NAND flash memory we examined in Section G includes a 64-byte spare area for extra

[34] Micron Technical Note TN-29-19, "NAND Flash 101," Rev. B, 4/10 EN, pp. 24-25

storage on each page (64 bytes per 2K-byte page). This spare area can be used to store the ECC code as well as other software information, such as wear-leveling or logical-to-physical block-mapping information. ECC can be performed in hardware or software; however, hardware implementation provides a performance advantage.

During a programming operation, the ECC circuit calculates the ECC code based on the data stored on the page. The ECC code for the data area is then written to the corresponding spare area. When the data is read out, the ECC code is also read out, and the reverse operation is applied to check that the data is correct. It is possible for the ECC algorithm to correct data errors. The number of data errors that can be corrected depends on the correction strength of the algorithm used. The inclusion of ECC in hardware or software provides a robust solution at the system level.

Simple Hamming codes provide the easiest hardware implementation for error correction; however, they can correct only single-bit errors. Reed-Solomon codes provide more robust error correction capability and are used in many controllers on the market today. BCH codes are also becoming popular due to their improved efficiency over Reed-Solomon codes.

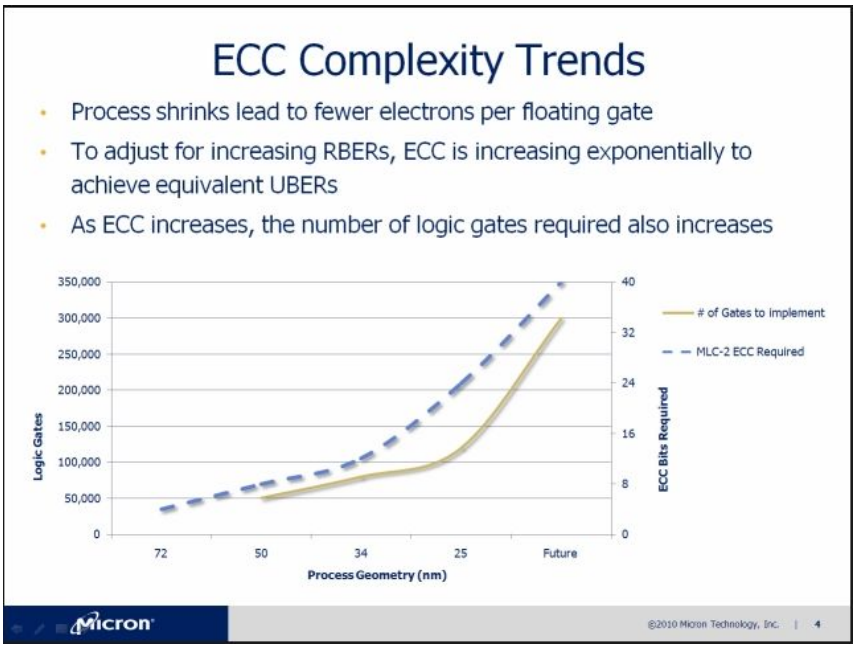
The next table shows the number of bits required for various ECC correction levels.

Table 12: Number of Bits Required for Various ECC Correction Strengths

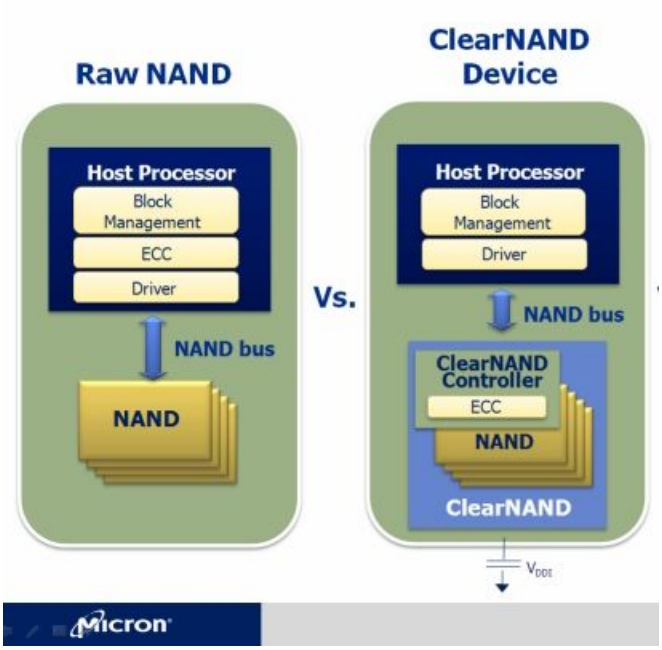
Error Correction Level	Bits Required in the NAND Flash Spare Area		
	Hamming	Reed-Solomon	BCH
1	13	18	13
2	N/A	36	26
3	N/A	54	39
4	N/A	72	52
5	N/A	90	65
6	N/A	108	78
7	N/A	126	91
8	N/A	144	104
9	N/A	162	117
10	N/A	180	130

Note: Codes in the shaded table cells can fit in the spare area of the NAND flash device discussed in Section G.

The graph[35] at right shows the error correction level required and the number of logic gates to implement that correction, both plotted vs. the process geometry. As device geometries shrink, much more robust error correction is required, and that in turn requires many more logic gates. In the text on this figure, RBER stands for Raw Bit Error Rate; and UBER stands for User Bit Error Rate.



In December 2010, Micron announced their ClearNAND product,[36] which combines an ECC controller and 8 GB to 64 GB of NAND inside a single multi-chip package. This approach looks quite attractive, as the ECC will be tailored by Micron to meet the needs of the particular NAND device inside the package; whether it be SLC, MLC or even a 3 bit-per-cell (TLC) design. ClearNAND shifts the burden of implementing the appropriate ECC technique from the user to the NAND vendor, and could make NAND devices from various vendors interchangeable. The comparison diagram[37] at left illustrates how ClearNAND moves the ECC function from the host processor to the NAND device package.

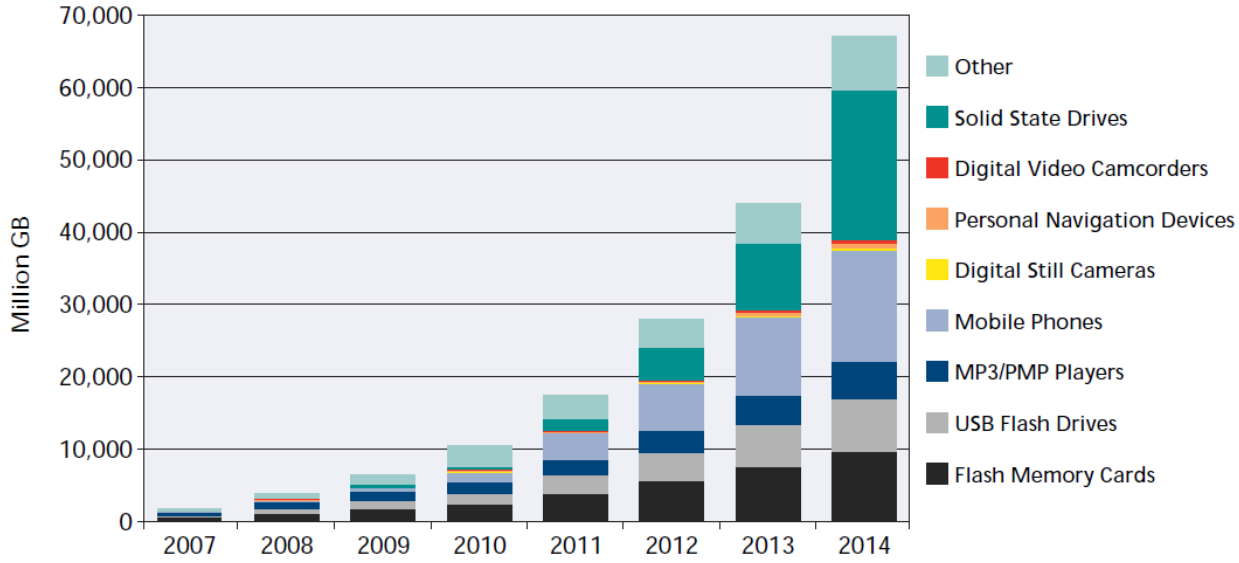


[35] http://extmedia.micron.com/webmedia/clearnand_intro/clearnand.html
[36] http://www.eetimes.com/General/DisplayPrintViewContent?contentItemId=4211180
[37] http://extmedia.micron.com/webmedia/clearnand_intro/clearnand.html

K. Flash Memory Market

Flash memory is expected to account for $26B in sales in 2010. Only analog, logic, DRAMs and MPUs (microprocessors) generate more in total sales.[38] NAND flash memory sales are expected to grow 18% in dollar volume (72% in GB) to $22B in 2011 after growing 38% in dollar volume to $18.7B in 2010.[39]

The following chart illustrates the growth of NAND flash memory, and the applications where it is used.[40]



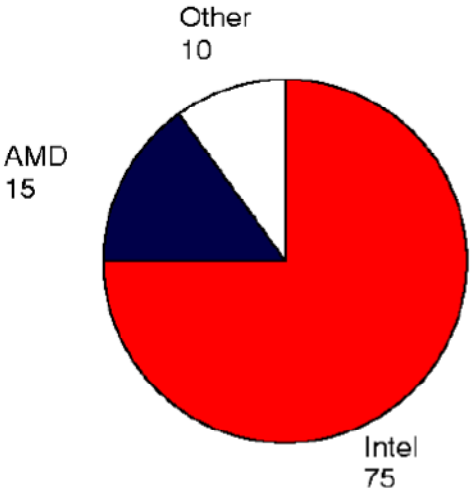
The number of bits of flash used are staggering. The vertical scale of the chart is "Million GB." So the first major division—10,000 Million GB—is 10^{19} Bytes of memory!

Where does all of the flash memory come from? The US International Trade Commission monitors world trade in various products, and has published a report that provides insight into the flash memory market. The following charts show the major flash memory vendors up to 2005.[41] It is interesting to note how Intel's market share has dropped, from 75% in 1992 to just 11% in 2005. The other interesting point is that only Intel has had a significant market share throughout the history of flash memory.

---

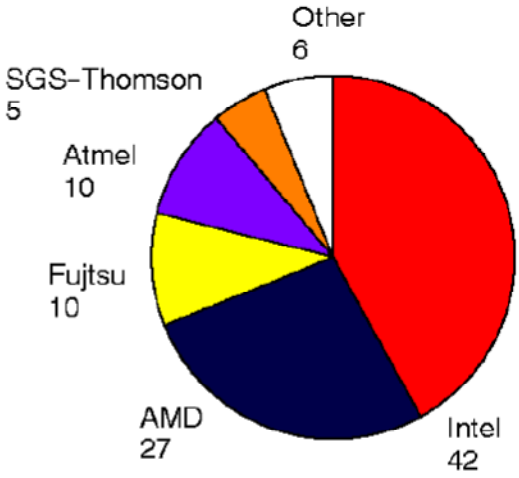
[38] http://www.databeans.net/reports/current_reports/SemiProd_Flash.php as of August 2010
[39] Clarke, Peter; "NAND flash market set to grow 18% in 2011;" http://www.eetimes.com/General/DisplayPrintViewContent?contentItemId=4212348; January 20, 2011
[40] Micron Technical Note TN-29-19, "NAND Flash 101," Rev. B, 4/10 EN, page 1
[41] Yinug, F., "The Rise of the Flash Memory Market: Its Impact on Firm Behavior and Global Semiconductor Trade Patterns," United States International Trade Commission, July 2007

Flash Memory Market Share, 1992
($310 million)



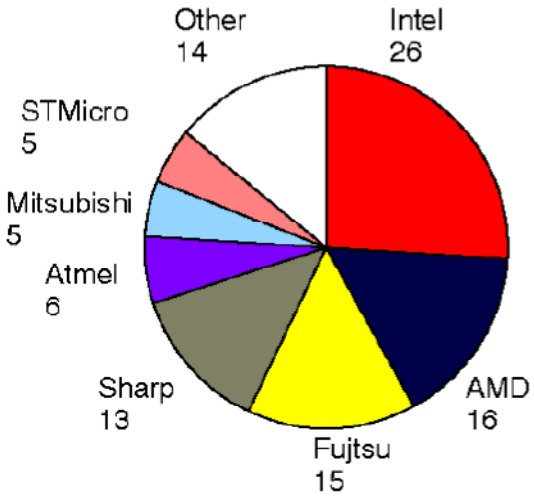
Source: ICE.

Flash Memory Market Share, 1995
($1.9 billion)



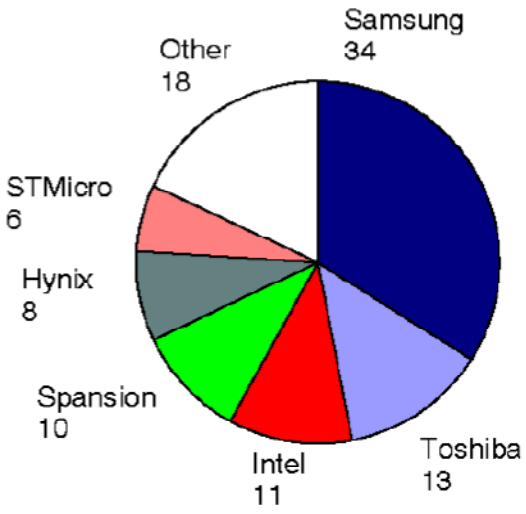
Source: ICE.

Flash Memory Market Share, 1999
($4.7 billion)



Source: ICE.

Flash Memory Market Share, 2005
($20.5 billion)



Source: Web-Feet Research.

Because of the enormous cost of building a new integrated circuit manufacturing facility,[42] many partnerships among IC vendors have recently occurred. Here are the partnerships that are focused on flash memory production:

Venture Name	Date	Partners	
FlashVision	4/2002	SanDisk (49.9%)	Toshiba (50.1%)
Spansion	2003	AMD	Fujitsu
front-end wafer fab in China	2004	Hynix (67%)	STMicroelectronics (33%)
Flash Partners	9/2004	SanDisk (49.9%)	Toshiba (50.1%)
IM Flash Technologies (IMFT)	1/6/2006	Intel (49%)	Micron (51%)
Numonyx[43]	3/31/2008	Intel	STMicroelectronics and Francisco Partners

L. Flash Memory Cards

From the chart on page 39, two of the largest markets for flash memories are Flash Memory Cards (including USB Flash Drives) and Solid State Drives. We will briefly examine these two applications to discover their important characteristics.

Removable flash memory cards are used primarily in digital cameras, camcorders, cell phones and PDAs. Proprietary/non-removable flash memory cards are used in music players (e.g., iPod, Zune) and home video game machines (e.g., Wii, X-Box 360, PlayStation). We will focus here on removable flash memory cards, as those are the ones we often buy as stand-alone products. NAND flash is used in virtually all removable flash cards.

There are an overwhelming variety of removable flash memory cards, most of which are not interchangeable. Here are photos of the largest (and oldest) format and the smallest:

CompactFlash (CF) Card

microSDHC Card

[42] Estimated at $2.5 Billion, Ibid., footnote on page 11
[43] Numonyx was acquired by Micron on February 9, 2010

The following table lists the most important characteristics of all of the card formats (as of 2009):[44]

Card family	Primary Vendors	Varieties	Entry date	Major features
CompactFlash	SanDisk	I	1994	Thinner (3.3 mm), flash based only, sizes available up to 128 GB
		II		Thicker (5.0 mm), older flash based, but usually Microdrives, sizes available up to 128 GB
SmartMedia	Toshiba	3.3/5 V	1995	Very thin (45.0 × 37.0 × 0.76 mm thick), no wear leveling controller, sizes available up to 128 MB
MultiMediaCard	Siemens AG, SanDisk	MMC	1997	Thin and small (24 mm × 32 mm × 1.4 mm), sizes available up to 16 GB
		RS-MMC/MMC Mobile	2003/ 2005	Compact size (24 mm × 18 mm × 1.4 mm), sizes available up to 16 GB
		MMCplus	2005	Compact size (24 mm × 32 mm × 1.4 mm), faster, optional DRM, sizes available up to 16 GB
		MMCmicro	2005	Sub compact size (14 mm × 12 mm × 1.1 mm), optional DRM, 16 MB – 4 GB
Secure Digital	Panasonic, SanDisk, Toshiba, Kodak	SD	1999	Small (32 mm × 24 mm × 2.1 mm), DRM, sizes available up to 4 GB
		miniSD	2003	Compact size (21.5 mm x 20 mm x 1.4 mm), DRM, available up to 16 GB
		microSD	2005	Sub compact size (11 mm x 15 mm x 1 mm), DRM, available up to 16 GB
		SDHC	2006	Physically the same as SD, but offers higher capacity and transfer speed, available up to 32 GB
		miniSDHC	2008	Physically the same as miniSD, but offers higher capacity and transfer speed, available up to 32 GB
		microSDHC	2007	Physically the same as microSD, but offers higher capacity and transfer speed, available up to 32 GB
		SDXC	2009	Physically the same as SD, but offers higher capacity and transfer speed, currently available up to 64 GB although the standard goes up to 2 TB

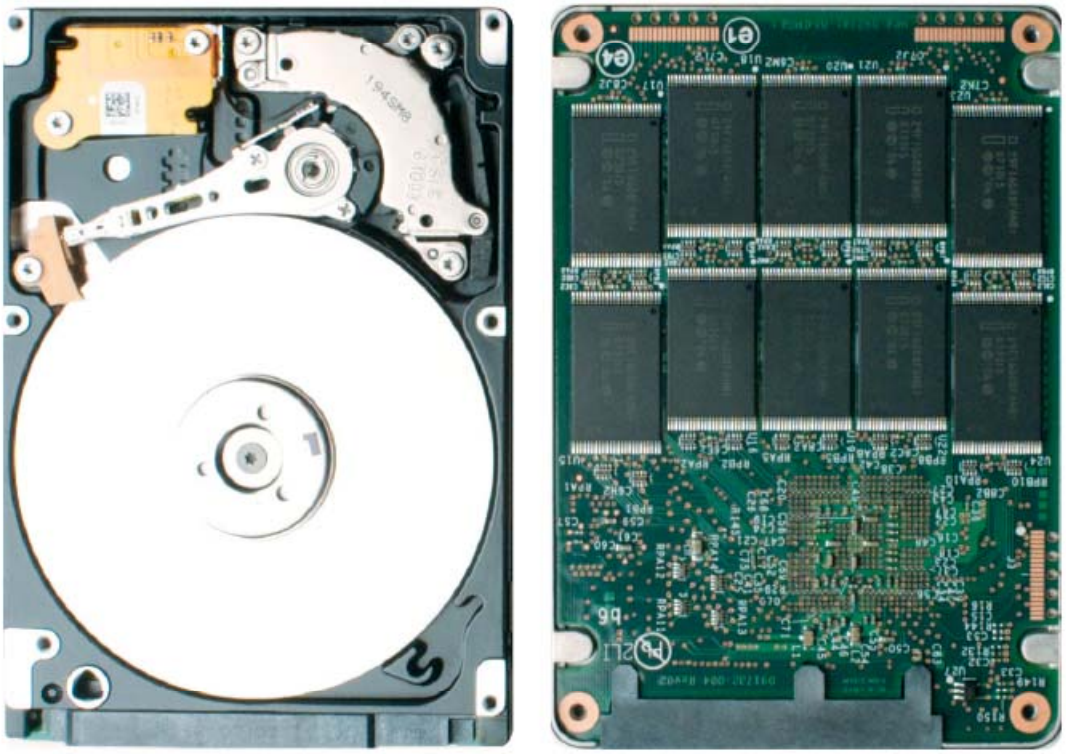
[44] http://en.wikipedia.org/wiki/Comparison_of_memory_cards This website provides extensive information about all types of removable flash memory cards.

Card family	Primary Vendors	Varieties	Entry date	Major features
Memory Stick	Sony/ SanDisk	Standard	1998	Thin and narrow (50 mm x 21.5 mm x 2.8 mm), optional DRM, available up to 128 MB
		PRO	2003	Thin and narrow (50 mm x 21.5 mm x 2.8 mm), faster, optional DRM, Memory up to 4 GB
		Duo	2003	Compact size (31 mm x 20 mm x 1.6 mm), optional DRM, Memory up to 128 MB
		PRO Duo	2002-06	Compact size (31 mm x 20 mm x 1.6 mm), optional DRM, available up to 32 GB
		PRO-HG Duo	2007-08	Compact size (31 mm x 20 mm x 1.6 mm), faster , optional DRM, available up to 32 GB
		Micro (M2)	2006-02	Sub compact size (15 mm x 12.5 mm x 1.2 mm), optional DRM, available up to 16 GB
xD	Olympus, Fujifilm	Standard	2002-07	Thin and small (20 mm × 25 mm × 1.78 mm), electrically identical to SmartMedia, no wear leveling controller, available up to 512 MB
		Type M	2005	Thin and small (20 mm × 25 mm × 1.78 mm) but slower read/write, no wear leveling controller, available up to 2 GB
		Type H	2005	Thin and small (20 mm × 25 mm × 1.78 mm) and faster than previous two versions, no wear leveling controller, available up to 2 GB
USB flash drive	Various	USB 1.1/2.0/3.0	2001	Universally compatible across all computer platforms, but larger size suits them better to file transfer/storage instead of use in portable devices, available up to 256 GB

M. Solid State Drives (SSD) for Hard Drive Replacement

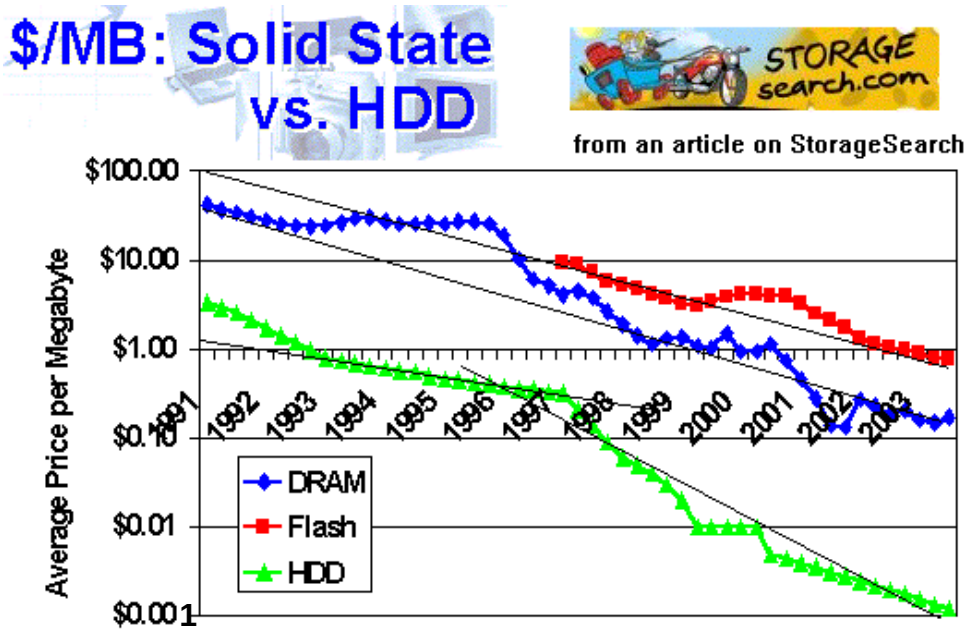
An obvious application area for flash memories would seem to be as replacements for magnetic hard disk drives (HDDs). SSDs even come in the same external package as HDDs, as shown below:[45]

HDD **SSD**



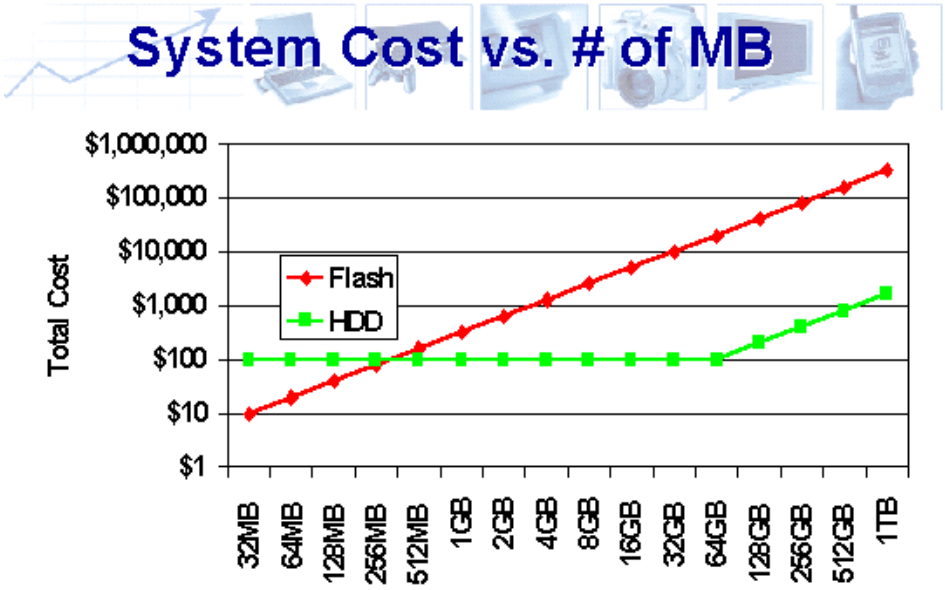
Flash memory has no moving mechanical parts, its read and write speeds are 100 to 1000 times faster, it makes no noise, has comparable reliability, and consumes much less power. But it costs much, much more per GB. And in consumer electronics, cost is an overriding factor. The following graph compares the cost history of DRAMs, Flash and HDD:[46]

[45] Ekker, N, T. Coughlin and J. Handy; "Solid State Storage 101—An Introduction to Solid State Storage," SNIA, January 2009
[46] http://www.storagesearch.com/semico-art1.html



From this graph, it looks like flash prices will never catch up with HDD prices. If this graph tells the whole story, why would anyone ever consider flash as an HDD replacement?

However, if we look at the bigger picture and consider System Cost vs. Storage Capacity, the picture is somewhat different.[47]



[47] loc. cit.

From this graph we see that HDDs have a floor price, regardless of capacity, of about $100. And we can buy a flash-based system with 512 MB of memory for that same $100. Of course, we could buy an HDD-based system with 64 GB of capacity for $100 also.

So will flash-based SSDs ever replace HDDs? The answer is not yet clear. But in some systems where the advantages of SSDs are essential, the answer is yes. For example, the Apple MacBook Air comes standard with a 64 GB SSD, and can be upgraded to 128 GB or 256 GB. The Apple iPad has a standard 16 GB SSD, and can be upgraded to a 32 GB or 64 GB SSD. The Amazon Kindle has a 2 GB NAND flash memory to store all those books.

In late 2010, Intel announced the 310 Series of SSDs, including a 40 GB version for $99; an 80 GB model for $179; a 120 GB model for $249; and a 160 GB model for $415. In January 2011, Micron announced sample availability of its RealSSD C400 SSDs in capacities from 64 to 512 GB. Memoright, a Taiwan-based company, recently announced a line of 2.5" form factor SSDs based on MLC NAND flash memory chips. Capacities are 60, 120, 240 and 400 GB.

Beyond these all-flash solutions is a hybrid SSD/HDD approach that takes advantage of the speed and low power of SSDs and the low cost of HDDs. Future consumer systems are likely to use this approach, at least until flash memory gets much less expensive.

Possibly the strongest evidence that SSDs will become a vital part of future computer systems is the involvement of JEDEC, the most important industry-wide standards-making body for semiconductor devices. In September 2010, JEDEC released two standards for determining the endurance of SSDs, JESD218 and JESD219. These standards were developed by JEDEC's JC-64.8 Subcommittee for Solid State Drives.[48] In January 2011, JEDEC announced that the JC-64.8 Subcommittee will develop standards for SSDs for applications beyond conventional disk drive form factors, such as tablets and ultra thin notebook PCs.[49]

[48] "Solid State Drives" at http://www.jedec.org/standards-documents/focus/flash/solid-state-drives
[49] http://www.eetimes.com/electronics-news/4212163/Jedec-to-create-standards-for-smaller-SSDs?cid=NL_EETimesDaily

Appendix A
Hexadecimal-Decimal-Binary Conversions

Hexadecimal	Decimal	Binary
0h	0	0000
1h	1	0001
2h	2	0010
3h	3	0011
4h	4	0100
5h	5	0101
6h	6	0110
7h	7	0111
8h	8	1000
9h	9	1001
Ah	10	1010
Bh	11	1011
Ch	12	1100
Dh	13	1101
Eh	14	1110
Fh	15	1111
10h	16	0001 0000
11h	17	0001 0001
12h	18	0001 0010
13h	19	0001 0011
14h	20	0001 0100
15h	21	0001 0101
16h	22	0001 0110
17h	23	0001 0111
18h	24	0001 1000
19h	25	0001 1001
1Ah	26	0001 1010
1Bh	27	0001 1011
1Ch	28	0001 1100
1Dh	29	0001 1101
1Eh	30	0001 1110
1Fh	31	0001 1111

Hexadecimal	Decimal	Binary
20h	32	0010 0000
21h	33	0010 0001
22h	34	0010 0010
23h	35	0010 0011
24h	36	0010 0100
25h	37	0010 0101
26h	38	0010 0110
27h	39	0010 0111
28h	40	0010 1000
29h	41	0010 1001
2Ah	42	0010 1010
2Bh	43	0010 1011
2Ch	44	0010 1100
2Dh	45	0010 1101
2Eh	46	0010 1110
2Fh	47	0010 1111
30h	48	0011 0000
31h	49	0011 0001
32h	50	0011 0010
33h	51	0011 0011
34h	52	0011 0100
35h	53	0011 0101
36h	54	0011 0110
37h	55	0011 0111
38h	56	0011 1000
39h	57	0011 1001
3Ah	58	0011 1010
3Bh	59	0011 1011
3Ch	60	0011 1100
3Dh	61	0011 1101
3Eh	62	0011 1110
3Fh	63	0011 1111

Cited References

1. D. Kahng and S.M. Sze, "A floating-gate and its application to memory devices," *The Bell System Technical Journal*, vol. 46, no. 6, July-August 1967, pp. 1288-1295
2. Masuoka, F. and H. Iizuka, "Semiconductor Memory Device and Method for Manufacturing the Same," U. S. Patent 4,531,203; Application date Nov. 13, 1981; FAP date Dec. 20, 1980; Issue date July 23, 1985
3. Masuoka, F. et al., "A New Flash EEPROM Cell Using Triple Polysilicon Technology," *IEEE IEDM Technical Digest*, 1984
4. Momodomi, M., Masuoka, F., et al., "Electrically Erasable Programmable Read-Only Memory with NAND Cell Structure," U. S. Patent 5,050,125; Application date Nov. 17, 1988; FAP date Nov. 18, 1987; Issue date Sep. 17, 1991
5. Fulford, B., "Unsung Hero," Forbes.com, June 24, 2002 (www.forbes.com/global/2002/0624/030_print.html)
6. Prince, B., ***Semiconductor Memories***, Second Edition, ISBN 0 471 92465 2, 1983, John Wiley & Sons, West Sussex, England
7. Brown, W.D. and J.E. Brewer, Editors; ***Nonvolatile Semiconductor Memory Technology***, ISBN 0-7803-1173-6, 1998, IEEE Press, New York, NY
8. Cottrell, P.E. et al., "Hot Electron Emission in N-Channel IGFETs," *IEEE Journal of Solid-State Circuits*, Volume SC-14, 1979
9. Eitan, B. and D. Frohman-Bentchkowsky, "Hot Electron Injection into the Oxide in N-Channel MOS Devices," *IEEE Transactions on Electron Devices*, Volume ED-23, 1981
10. Lenzlinger, M. and E.H. Snow, "Fowler-Nordheim Tunneling in Thermally Grown SiO_2," *Journal of Applied Physics*, Vol. 40, 1969
11. Spansion Application Note , "Flash Memory: An Overview," November 10, 2005
12. Numonyx Embedded Flash Memory (J3 65 nm) Single Bit Cell (SBC) Data Sheet, March 2010
13. Micron MT29F2G08A NAND Flash Data Sheet, Rev. 1, 1/06 EN
14. http://en.wikipedia.org/wiki/Greinacher_multiplier
15. Cockcroft, J.D. and E.T. Walton, "Production of high velocity positive ions," *Proceedings of the Royal Society*, A, Vol. 136, 1932, pp. 619-630
16. Dickson, J. "On-chip High-Voltage Generation in NMOS Integrated Circuits Using an Improved Voltage Multiplier Technique" *IEEE Journal of Solid-State Circuits*, Vol. 11, No. 6, June 1976, pp. 374–378

17. http://www.answers.com/topic/charge-pump
18. Moore, G., *Electronics Magazine,* "Cramming more components onto integrated circuits," April 19, 1965
19. http://www.micron.com/products/nand_flash/nandcom.html#mlcnand
20. Micron Technical Note TN-29-19, "NAND Flash 101," Rev. B, 4/10 EN
21. Wikipedia, http://en.wikipedia.org/wiki/Hamming(7,4)
22. http://extmedia.micron.com/webmedia/clearnand_intro/clearnand.html
23. http://www.eetimes.com/General/DisplayPrintViewContent?contentItemId=4211180
24. http://www.databeans.net/reports/current_reports/SemiProd_Flash.php
25. Clarke, Peter; "NAND flash market set to grow 18% in 2011;" http://www.eetimes.com/General/DisplayPrintViewContent?contentItemId=4212348; January 20, 2011
26. Yinug, F., "The Rise of the Flash Memory Market: Its Impact on Firm Behavior and Global Semiconductor Trade Patterns," United States International Trade Commission, July 2007
27. "Solid State Drives" at http://www.jedec.org/standards-documents/focus/flash/solid-state-drives
28. http://www.eetimes.com/electronics-news/4212163/Jedec-to-create-standards-for-smaller-SSDs?cid=NL_EETimesDaily

Other References

1. Hu, C., Editor, ***Nonvolatile Semiconductor Memories—Technologies, Design and Applications***, ISBN 0-87942-269-6, IEEE Press, New York, NY, 1991
2. Foss, R.C. et. al., "High Voltage Boosted Word Line Supply Charge Pump and Regulator for DRAM," U. S. Patent 5,406,523; Application date October 12, 1993; FAP date April 6, 1990; Issue date April 11, 1995
3. Sharma, A.K., ***Semiconductor Memories—Technology, Testing and Reliability***, 1996, IEEE Press, New York, NY
4. Prince, B., ***Emerging Memories—Technologies and Trends***, 2002, Kluwer Academic Publishers, Norwell, MA
5. Cooke, J., "Flash memory 101: An Introduction to NAND flash," http://eetimes.com/General/DisplayPrintViewContent?contentItemId=4009410, March 20, 2006

6. Cooke, J., “NAND 201: the continued evolution of NAND Flash,” http://eetimes.com/General/DisplayPrintViewContent?contentItemId=4213098, February 13, 2011

