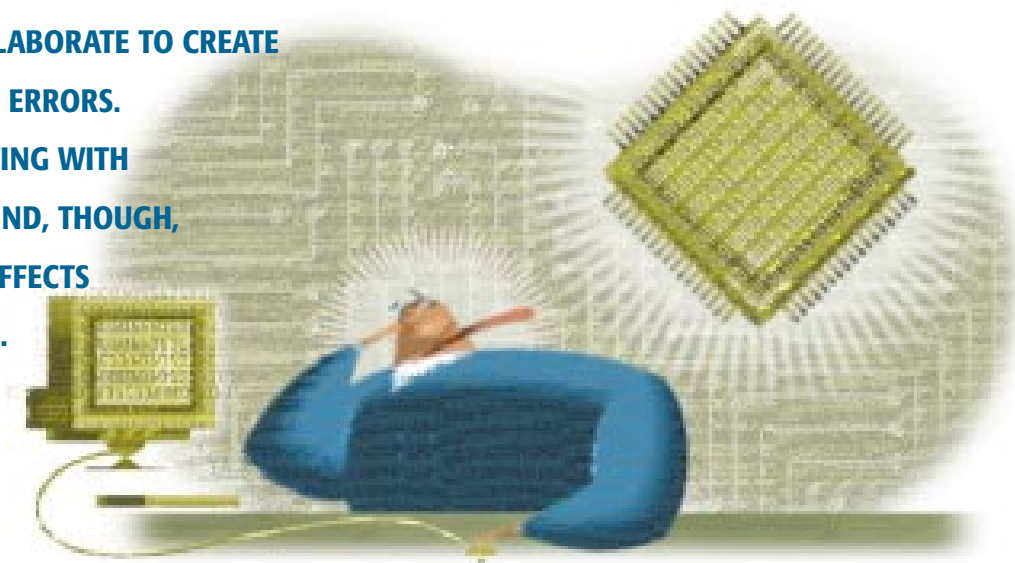


**NUMEROUS FACTORS—SOME OF WHICH YOU CAN INFLUENCE AND OTHERS BEYOND YOUR CONTROL—COLLABORATE TO CREATE MEMORY-SUBSYSTEM ERRORS. DESIGNING AND TESTING WITH THESE FACTORS IN MIND, THOUGH, CAN CURTAIL THEIR EFFECTS AT THE SYSTEM LEVEL.**



# Banish bad memories

**M**EMORY PRICES ARE AT AN ALL-TIME LOW, and enormous half-gigabit DRAMs, multigigabit flash memories and multimegabit SRAMs are rolling off suppliers' assembly lines. Those same memories are reading and writing faster and burning less power per bit than ever. It's a great time to be a software engineer writing code to

use up those bits or a hardware engineer designing memory into your next project. Or is it?

All those I/O buffers and internal device nodes toggling ever faster are creating increasing amounts of intrachip and interchip noise. Finer pitch packaging and the resulting greater density trace routing on pc boards boost the probability of manufacturing flaws. And smaller chip-fabrication lithographies, translating to lower cell capacitance, make it more likely that each memory will experience one or multiple defect-created hard errors and radiation-induced soft errors through its lifetime.

Forward-thinking chip designers

might minimize errors, but they can't eliminate them. All's not gloom and doom, though. Additional error-correction and -prevention hardware- and software-design work at the system level, along with robust testing in the system-manufacturing flow, can lower the probability of a memory-created system failure to the point at which its MTTF (mean time to failure) far exceeds the most optimistic projection of usable system lifetime. But there's the rub.

You don't want your customers to experience an unacceptable level of failure. However, you also don't want to place an excessive cost burden on or strap the performance of the system in attempting

Illustration by Daniel Guidera

*At a glance* .....62  
*Boom, boom, boom... out go the lights*..... 64  
*Web education...is making me great*.....66  
*For more information* ....70

to prevent failures such that you fail to meet other important design objectives. As with any other aspect of engineering, you have a balancing act on your hands. And you have to perform the act during system development, on the manufacturing line, and in the field. Similarly, the semiconductor vendors must decide to what extent they encumber their devices in cost, performance, and power in striving to maximize reliability, versus assuming that customers with stringent reliability needs will supplement chip capabilities with system-level error detection and correction.

**COME ON, HEAL THE NOISE**

Considering the large number of phenomena that can cause semiconductor devices to fail, you might conclude that it's a wonder memories work at all. "Failure" can mean either a "hard" or a "soft" error. A hard error involves the permanent breakdown of all or a portion of a chip's circuitry, lead frame, or packaging. Conversely, in a soft error, the breakdown is temporary and repairable. Soft errors may involve corruption of stored data, such as noise on a data line that results in incorrect information being written to a memory-storage location or bit-flipping of previously written data by an alpha particle or cosmic ray. Alternatively, the soft error may not corrupt the stored data at all—for example, if ground bounce temporarily causes a memory's sense amplifier or I/O buffer to misinterpret a bit's state during a memory read.

You should also know a few semiconductor-reliability terms. FITs (failures in time) are a measure of the number of failures in a billion ( $1 \times 10^9$ ) operating hours. Dust off your old probability textbook, crunch the math, and you'll discover that 1000 failures in time corresponds to an MTTF of approximately 114 years. Because the FIT rate is a function of the population of bits that could

**AT A GLANCE**

- ▶ Reliability is the Achilles' heel of today's otherwise-wonderful memory cost, density, speed, and power-consumption trends.
- ▶ Error prevention begins with attentive design.
- ▶ Robust testing screens out chips and systems that might otherwise fail in your customers' hands.
- ▶ Parity, correction codes, and redundancy are all means of overcoming field failures caused by radiation and other phenomena.
- ▶ Pay particular attention to nonvolatile memory's potential errors, which you can't "fix" by a reboot or a power cycle.

fail, a normalized FIT-per-megabit figure can be helpful when comparing the reliability specifications of different devices, densities, technologies, and suppliers.

After calculating the failure rate for your memory subsystem, determine the failure-rate tolerance of your application. If, for example, you're designing a pre-compression capture buffer or postdecompression playback buffer for an audio-, video-, or still-imaging system, an occasional bad bit may be unnoticeable. Bad bits in the compressed data and cor-

ruption of a system's firmware are more egregious failures, especially in mission-critical applications. Also, how long will you store the data? Your customers may be upgrading the PC BIOS every six months or replacing this year's cell phone with next year's even smaller, sleeker model, or you may be designing a graphics-card frame buffer that gets rewritten 100 times a second. In these scenarios, you can relax your reliability expectations versus those for a system you expect to continually run maintenance-free for several decades.

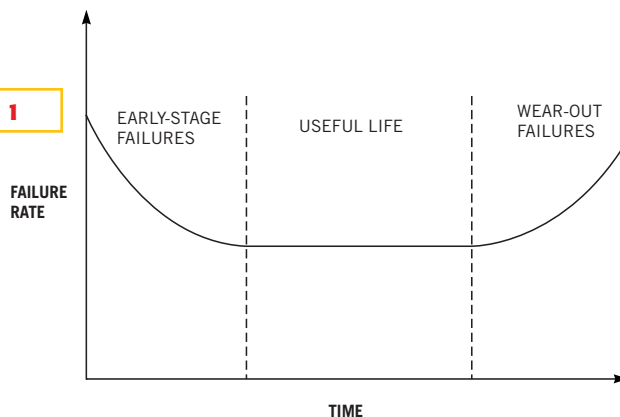
Your challenge, both during system development and in preproduction qualification, is to expose your prototype units to as many production-system usage scenarios as possible. Electrical interference is one key focus area in your debugging procedures. Look for ground bounce; supply-voltage transients; noisy data, address, and control lines; and fast-switching signals within the memory itself or within other components on the board that can create these problems.

Keep an eye on chips entering or exiting high-power operating modes, motors and displays turning on and off, and other factors. These situations can cause out-of-spec supply-voltage droops and overshoots. You should extensively and randomly power-cycle your systems during production qualification, either manually or with the aid of automated equipment such as MicroTools' Poc-It. Nonvolatile memories are especially vulnerable to power cycling

(see sidebar "Boom, boom, boom....out go the lights").

Memory read and write speeds are increasing, and you more frequently access the memories through system lifetime. These phenomena increase the probability of soft errors. Finer pitch leads on advanced packaging results in not only more stringent impedance characteristics, but also an improved likelihood of EMI coupling between pins and board traces. And ever lower internal operating voltages reduce the zero-versus-one

**Figure 1**



**Qualification before production with stresses beyond those normally seen, along with similar testing on the production line, enable manufacturers to screen out infant-mortality failures in both chips and the systems containing them.**

signal margin, making the chips less noise-tolerant.

Vendors qualify semiconductor devices to operate not just across specific ranges of voltage and current, but also at maximum junction and ambient temperatures, instantaneous shock and long-term vibration thresholds, and atmospheric-moisture levels. Most memories, except for high-speed devices, such as Rambus DRAMs, don't themselves generate a lot of heat. Thermal issues are more likely to come from nearby chips on the board. Heat flows from the warmer chips to the memory by

**Figure 2**

D <sub>7</sub> .....D <sub>0</sub>	A <sub>7</sub> .....A <sub>0</sub>
00000001	00000000
00000010	00000001
00000100	00000010
00001000	00000100
00010000	00001000
00100000	00010000
01000000	00100000
10000000	01000000
(a)	(b) 10000000

**A walking-ones pattern is useful for testing a memory's data bus (a), whereas the similar, but not identical, power-of-two sequence exercises the address bus (b).**

either conduction or convection. Fans, which designers often do not favor because of their own reliability issues and noise; heat sinks; and additional ventilation holes are all possible remedies to thermal problems.

Many semiconductor-failure mechanisms are physicochemical, because they are patterned after the Arrhenius equation, which describes the temperature-dependent rate of a chemical reaction. As a result, when vendors qualify their processes and devices for production, they test them beyond the levels of voltage, current, temperature, shock, vibra-

## BOOM, BOOM, BOOM....OUT GO THE LIGHTS

The contents of volatile memories are, as their name implies, unable to survive the removal of power. Volatile-memory corruption might cause lockup or other faulty behavior, but toggling the reset or power switches should restore normal system function. If it doesn't, your design has made unrealistic assumptions.

Nonvolatile memories, on the other hand, should survive power loss without data loss. In the days of ROMs, you could easily achieve this goal. The only thing you had to worry about was that an uncovered EPROM window might allow enough ultraviolet light onto the die to erase data within a few years.

Now, however, you use nonvolatile, in-system-writable memories: a potentially troublesome combination. Regardless of whether you use battery-backed, otherwise-volatile RAM technology or an inherently nonvolatile approach, such as flash memory, EEPROM, or ferroelectric RAM, the potential exists for spurious alteration of stored code or data. The system must have the smarts to prevent, detect, and recover from this situation.

Some people believe that the multicommand sequences necessary to program or erase a nonvolatile memory are in and of themselves sufficient write

protection. This assumption seems reasonable because it is difficult to imagine the random coincidental combinations of address and data that are required to occur to cause data to be lost. However, it can take only one write to put the memory into a mode in which it outputs status-register data instead of the expected memory-array information, for example, to subsequent read operations. Such a deviation from expected memory function is enough to bring many systems to their knees.

RC circuits do not provide sufficient write protection. Such a circuit, with its slow charge characteristics, may indeed keep a CPU in a safe standby mode during system power-up, but it does nothing to prevent wildly gyrating outputs in intermediary logic. However, the RC circuit's equally slow discharge characteristics don't protect the nonvolatile memory during supply-voltage undershoots. Instead, employ an analog power-monitoring circuit that performs functions such as blocking chip selects, holding flash-memory reset inputs active, and instantaneously switching RAM power to battery backup in response to voltage transitions outside a valid range.

Most nonvolatile memories

take microseconds, milliseconds, or even seconds to complete their write and erase operations. Ensure that if system power fails, enough reserve charge, perhaps in large capacitors, is available to allow these operations to complete. Similarly, if you're in the middle of a multi-byte write, such as transferring a 32-bit data value to an 8-bit memory, ensure that an impending power-loss-triggered, nonmaskable interrupt doesn't distract the system CPU from the task. Also, make sure the system has sufficient juice to complete the write.

First-generation flash memories were bulk-erasable, meaning that if a system lost power during an update, it would reboot and find a corrupted copy of the firmware. Newer block-erasable architectures often contain hardware pin-lockable or configuration-bit-lockable boot blocks. Software stored in the boot block can calculate a checksum for each other blocks' contents, compare it with a checksum within the block, and alert the user if they don't match. Similarly, data table and file storage can use block ECC, which is more space-efficient than the byte ECC that directly executing firmware uses.

Your file system algorithms should adjust for the lengthened

program and erase times that many nonvolatile memories require at extended cycle counts, as well as the fact that, beyond a certain cycling threshold, further programming and erasing is impossible. Wear leveling to avoid cycling "hot spots" within the array, along with linked lists, circular queuing, and other cycle-minimization techniques, can extend memory life. Finally, if you're working with removable storage media, keep in mind that a user could eject the media while it's in the middle of a foreground or, more likely, a background program or erase procedure.

### REFERENCES

- A. Dipert, Brian, and Markus Levy, *Designing with Flash Memory*, ISBN 0-929392-17-5, Annabooks Press, San Diego, CA, 1994.
- B. Ganssle, Jack, "Nonvolatile RAM," *Embedded Systems Programming*, April 1999, pg 105.
- C. Hinerman, David, "Making nonvolatile data reliable," *Embedded Systems Programming*, August 1997, pg 60.
- D. Hu, Chenming (editor), *Nonvolatile Semiconductor Memories: Technologies, Design and Applications*, ISBN 0-87942-269-6, IEEE Press, New York, NY, 1991.

tion, pressure, and other environmental variables that they'll see in normal use to accelerate the emergence of latent-failure mechanisms (**Figure 1**). Both the extremes of these variables and the rapidity with which the testing transitions them from one extreme to another—that is, hot to cold and quickly back again—are important considerations when manufacturers attempt to weed out bad chips. Similarly, *you* should test your system at extended operating conditions, such as the well-known 85/85 combination (85°C, 85% humidity).

### SLAY IT ON THE LINE

Once you have your system into production, your work is, alas, not over. Naturally occurring variations in the components you place on a pc board, in the pc board itself, and in the process used to attach the components to the pc board necessitate extensive assembly-line testing. Keep in mind that anything out of spec that you do to a chip after you receive it is a potential reliability issue above and beyond what the vendor's testing has attempted to encompass. Insufficient antistatic protection, bent leads caused by rough handling, chemical contamination and heat damage caused by soldering and other manufacturing processes can cause many system failures.

After you ensure that your assembly line properly handles the chips, most of the flaws your testing uncovers will probably have their root in open- and short-circuited traces and pins and in stuck-at-one and stuck-at-zero faults. You might even find a few chips improperly inserted in their sockets, incompletely soldered to the board, or completely missing! As a result, you need to test the memories' address, data, and control buses and you might also execute a full-chip evaluation; the order in which you conduct these operations is important. The address-bus test assumes a functional data bus, and full-chip test results are undecipherable unless you've previously determined that the address and data buses are properly functioning (see sidebar "Web education...is making me great").

To test the data bus, write and then read back each value of a walking-ones pattern (**Figure 2a**). Because you're testing only data-bus functions, you can

### WEB EDUCATION... IS MAKING ME GREAT

Well-known industry consultant Bob Zeidman offers the \$40 online class "Testing Memory Devices Quickly and Efficiently," via his company's Web site [www.chalknet.com](http://www.chalknet.com). In it, he provides a consolidated alternative algorithm to the separate walking-ones data-bus test and power-of-two address-bus test.

write the entire pattern to a single location within each memory. (Remember, though, if you have multiple memories covering different system-address ranges, you need to sequentially test each of them.) If you decide to skip the full-chip test, you should, in the data-bus test, insert a delay between writing a value and reading it back. Otherwise, you might unknowingly be reading the capacitance of the data-bus traces and conclude you have a functional chip when in fact the entire memory is missing!

Now for the address bus. Your goal is to isolate and confirm that you can toggle each address bit without disturbing data stored at other locations. It's unnecessary to test every single address within the device; the row-and-column arrangement of memories means that a much simpler power-of-two address pattern works equally well (**Figure 2b**). First, initialize each power-of-two address location to a data value of your choice, such as 55h. Now, one address at a time,

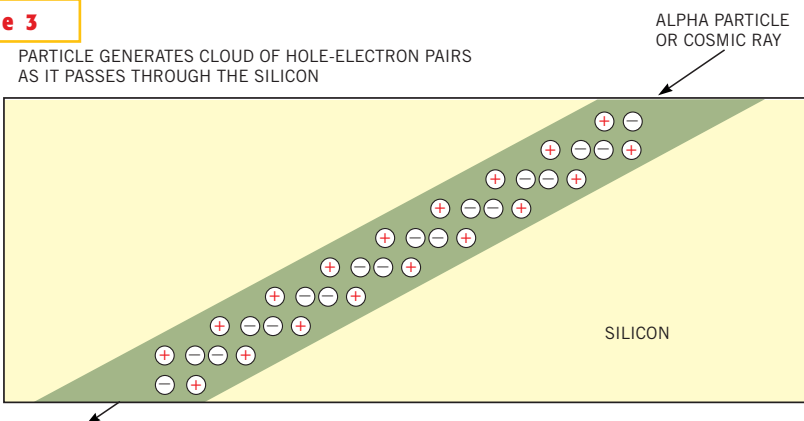
change the data by inverting the previous data pattern. (For example, AAh, is a good bet.) Then confirm that no other power-of-two locations' data got corrupted.

Finally, you can do a full-chip test, which gives you the greatest flexibility. You might choose to skip it, relying on the assumptions that the vendor has tested the chip and your address- and data-bus tests have caught any catastrophic failure that might remain. Because testing takes time, and time is money on a high-volume system-manufacturing line, such a decision is reasonable. On the other end of the spectrum, you might decide to twice write and read back every location within the memory, using inverted data in the second pass. If you choose this ultimate test-coverage option, use a data pattern that doesn't repeat on binary-number address boundaries. You can, for example, exercise memories with a 52-byte pattern comprising the ASCII values for A through Z in both uppercase and lowercase. Compromise tests between the two extremes of "test nothing" and "test everything" exercise a subset of all possible addresses.

### ERROR-FREE FIELDS FOREVER

By now, you've pretested a number of possible system-operating scenarios in the lab, and you're prescreening for bad chips and pc boards in the manufacturing line. Unfortunately, you still have more error-detection, -correction, and -prevention work to do. Invariably, at

**Figure 3**



Alpha particles and cosmic rays leave a trail of electron/hole pairs in their wake as they traverse semiconductor material (courtesy MoSys).

least a few in-the-field usage situations will arise that you haven't predicted. Power to the entire system or to a subsystem might go on or off in the middle of a memory update, for example, or some obscure sequence and timing of user-interface inputs might lead to runaway software and consequent data or code corruption.

Chip vendors extensively stress-test their chips, particularly at the early stages of device and process production, to screen out "infant-mortality" failures. But there's no guarantee that some obscure defect won't slip through their—and your—testing. The device-failure rate significantly drops for most of a device's specified operating life, but it doesn't go all the way to zero. And, beyond a certain time threshold, the failure rate again climbs, a reflection of oxide defects, metal-

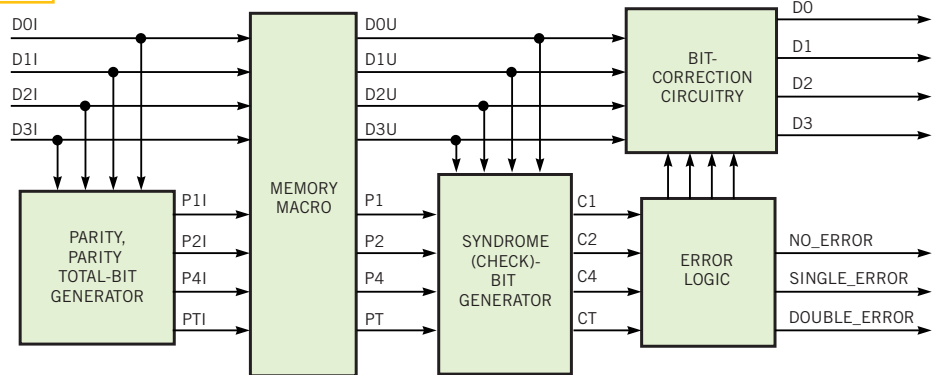
interconnect electromigration, and other phenomena.

Occasional product and process changes have the potential to cause you headaches, too. A process change on an existing memory design may cause I/O buffers and internal nodes to switch more rapidly, or it may cause vastly different power-consumption profiles in various operating modes. Changes you

make also have an effect: A board redesign to reduce cost might move the memory array closer to a heat-generating component on the board or alter traces' impedance, for example.

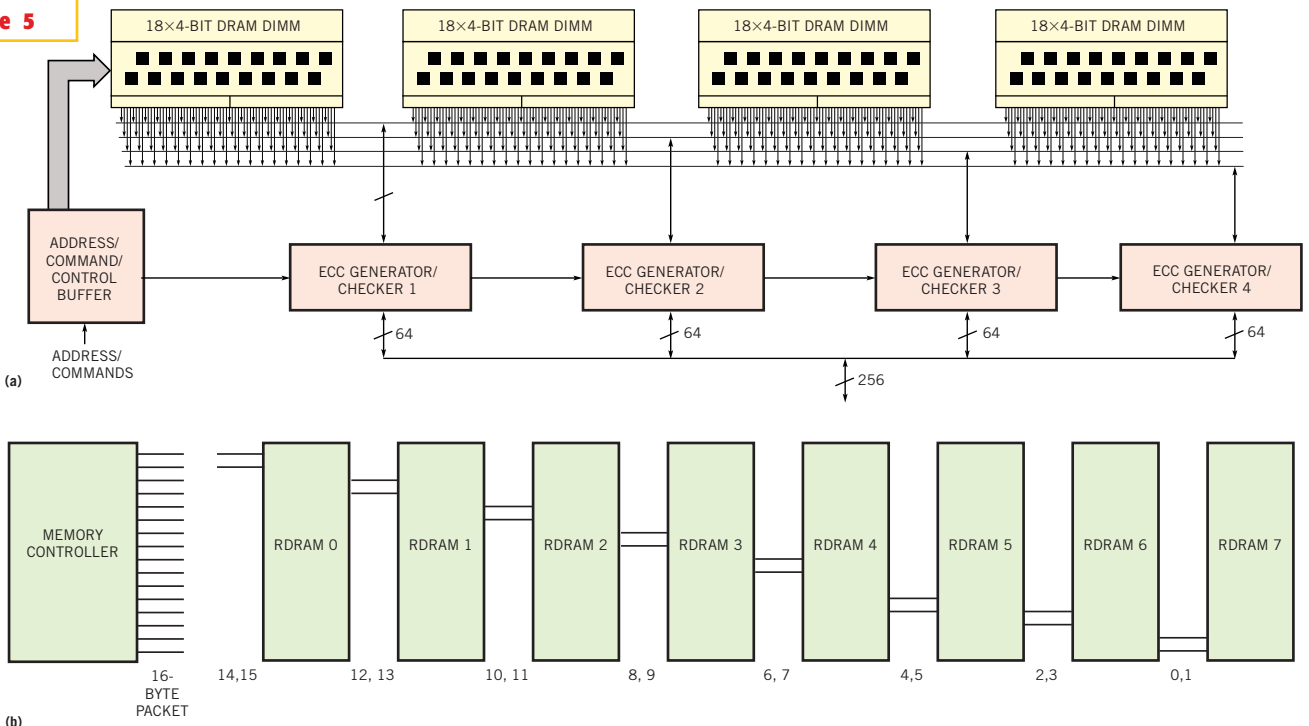
Alpha-particle and cosmic-ray effects can create soft errors at any point in a memory's life, and they're unfortunately difficult both to predict and to prevent. The failure mechanism is similar

**Figure 4**



Additional check bits beyond those needed for parity enable both memory-error detection and correction (©2001 IEEE Spectrum).

**Figure 5**



Elaborate multichip arrangements enable a wide-bus memory subsystem to remain functional after an entire device fails (a), and RDRAMs will support similar capabilities beginning at the 256-Mbit product generation (b) (courtesy Rambus).

for both types of charged particles. When ionizing radiation passes through a semiconductor, it creates electron/hole pairs in the area surrounding its path. If the radiation strikes a bit line leading from a storage cell to the device output, corruption of the stored data may not occur, although a read from the memory at that moment will return invalid information. Signal corruption during a memory-write operation, on the other hand, results in the storage of invalid data. And, if the critical charge the radiation-induced electron/hole generation causes is stronger than the capacitance of the cell storage node, bit-flipping can take place.

Alpha particles fortunately have an influence range of only a few dozen nanometers, so shielding the die with a thin layer of plastic or other coating suppresses them. Maintaining chemical purity during silicon processing and preventing the use of radioactive materials in packaging and lead frames are other steps that vendors take to reduce alpha-particle populations. Cosmic rays are, unfortunately, more difficult to circumvent: They effortlessly travel even through many feet of concrete. Both the earth's atmosphere and soil gradually block these rays, however, so underground equipment receives less exposure to them than gear at sea level. Hence, the electronics in airplanes, satellites, and other in-the-air applications are the most vulnerable.

#### EXCUSE ME WHILE I FIX THIS BYTE

A number of options mitigate the system-level impacts of soft- and hard-memory errors. Because the probability

of stored-data corruption due to radiation particles inversely correlates to the capacitance of the storage cell, you might want to investigate using higher-capacitance DRAM instead of SRAM. MoSys' 1T-SRAM product naming is deceptive: The underlying technology isn't SRAM at all, but DRAM. Most vendors, with performance in mind, construct their DRAM cells from n-channel structures, but MoSys uses p-channel structures surrounded by an n-type well for greater soft-error resistance. MoSys obtains speed by dividing the memory array into a large number of small banks, which also results in short bit lines and large signal margins that further improve soft-error immunity.

Regardless of your preferred memory technology, a mission-critical application might employ multiple redundant-memory arrays. For example, memory-write operations might in parallel change the contents of three arrays, and logic between the arrays and the system microprocessor would compare the arrays' outputs during read operations. If this operation didn't yield a three-way match, the logic would select and pass on to the microprocessor the output of the two arrays that did correspond and then write that same data back to the third—presumed corrupted—array.

If your bill-of-materials budget has no room for multiple memory arrays, consider adding parity bits. Commonly, an extra bit, representing either even or odd parity, pairs up to each eight data bits; that is, to each byte. Parity support alerts your system to the presence of an odd number of errors, but it doesn't fix the errors, and it doesn't detect an even num-

## FOR MORE INFORMATION...

For more information on products such as those discussed in this article, go to [www.ednmag.com](http://www.ednmag.com) and click on the Reader Service link under the Tools & Services section. When you contact any of the following manufacturers directly, please let them know you read about their products in *EDN*.

#### MicroTools

1-860-651-6170  
[www.microtoolsinc.com](http://www.microtoolsinc.com)  
Enter No. 318

#### MoSys

1-408-731-1800  
[www.mosys.com](http://www.mosys.com)  
Enter No. 319

#### Rambus

1-650-947-5000  
[www.rambus.com](http://www.rambus.com)  
Enter No. 320

#### SUPER INFO NUMBER

For more information on the products available from all of the vendors listed in this box, go to [www.ednmag.com](http://www.ednmag.com), click on the Reader Service link, and enter no. 321

ber of errors. When you receive a parity error, you can attempt reread in the hope that ground bounce; bit-line corruption; or other temporary, self-healing phenomena caused it and that the storage cell is still intact. You can also communicate error status to the equipment user as a precursor to a “graceful” abort. But normal system operation beyond the point of the error is generally impossible.

The addition of more than a single integrity bit per data bit grouping enables not only error detection, but also correction (Figure 4). ECC efficiency improves as the data-bit string gets longer. To correct one data-bit error in a 64-bit grouping, for example, requires only eight ECC bits, the same number of bits that simple eight-groupings-of-eight data-bit parity-error detection requires. Ideally, the memory vendor should have designed the array such that physically adjacent cells are in different logical words. In this way, if an alpha particle or cosmic ray disturbs both cells’ data, you end up with two correctable single-bit errors versus an uncorrectable double-bit error.

Aside from the additional memory cost to store the ECC bits, error detection and correction has other disadvantages. Read performance decreases due to the presence of slow XOR gates between the memory array and the rest of system. Error detection and correction also impacts write performance, especially if you don’t rewrite the entire memory data bus at one time. Because the ECC algorithm generates check bits across the entire data field, alteration of any part of that data field results in an invalid ECC code. A read-modify-write approach is the cleanest, albeit the slowest, means of resolving this problem; delayed write-back buffers can also find use if you eventually update the entire data field.

Taking ECC to the extreme, so-called chip-kill architectures, common in servers, can compensate for the failure of an entire memory component. As an analogy, think of how a RAID hard-drive array works. Chip kill extends the ECC concept by architecturally partitioning the memories such that any individual chip contributes only one bit to each data field. You can most easily accomplish chip kill with single-bit data-bus memories, which are rare, so the technique becomes increasingly cumbersome to implement

as per-chip data buses widen (Figure 5a). Rambus’ Interleaved Data Mode, to appear on RDRAMs beginning at the 256-Mbit generation, supports chip kill without incurring the additional latency and bandwidth degradation common in traditional SDRAM implementations (Figure 5b). □

---

#### REFERENCES

1. Barr, Michael, “Software-based memory testing,” *Embedded Systems Programming*, July 2000, pg 28.
2. Cataldo, Anthony, “SRAM soft errors cause hard network problems,” *EE Times*, Aug 20, 2001, pg 1.
3. Dell, Timothy, and IBM Microelectronics, “The benefits of chipkill-correct ECC for PC server main memory,” White Paper, 1997.
4. Ganssle, Jack, “Testing RAM,” *Embedded Systems Programming*, October 1997, pg 153.
5. Gray, Ken, “Adding error-correcting circuitry to ASIC memory,” *IEEE Spectrum*, April 2000, pg 55.
6. “Fault tolerance decision in SDRAM applications,” IBM Microelectronics, 1997.
7. “IBM chipkill memory,” IBM, 1999.
8. Jones, Mark-Eric, “When memories forget: soft errors in very deep sub-micron memories,” IP World Forum, 2001.
9. Lakshminarayanan, Venkataraman, “Predicting semiconductor failure modes,” *ISD*, July 2000, pg 62.
10. Leung, Wingyu, Fu-Chieh Hsu, and Mark-Eric Jones, “The ideal system-on-chip memory: 1T-SRAM,” IEEE SOC/ASIC Conference, 1999.
11. Locklear, David, and Dell Computer, “Chipkill correct memory architecture,” 2000.
12. Micron Technology, “TN-04-28 DRAM soft error rate calculations,” 1994.
13. Peterson, W Wesley, and EJ Weldon Jr, *Error-Correcting Codes*, ISBN 0-262-16-039-0, MIT Press, Cambridge, MA, 1972.

---

#### AUTHOR’S BIOGRAPHY



Technical editor Brian Dipert wishes his memory problems were so easily solved. You can reach him at 1-916-454-5242, fax 1-916-454-5101, [bdipert@pacbell.net](mailto:bdipert@pacbell.net), and [www.bdipert.com](http://www.bdipert.com).