

A quick look at FRAM in MCUs

Fast write time, high endurance, and low-voltage operation contribute to popularity

BY MIKE ALWAIS
Ramtron International
Colorado Springs, CO
<http://www.ramtron.com>

Ferroelectric random access memory (FRAM) is gaining popularity among design engineers as a nonvolatile memory choice. Since memory technologies commonly migrate from standalone to embedded form as they mature, there is increasing interest in FRAM solutions embedded with a processor.

Three distinct benefits account for the rise in popularity of FRAM. First, it performs write operations at the same speed as read operations. FRAM imposes no delay for the written data to be nonvolatile, unlike nonvolatile memories based on floating-gate technology that have long write delays. For example, a typical EEPROM write operation takes 10 ms for the write to be effective. In addition, FRAM has no erase operation since there is no preferred or default state. As with other RAM technologies such as SRAM, data are written with-out regard to the previous state.

Second, FRAM offers effectively unlimited write endurance—it doesn't wear out like other nonvolatile memory choices. Floating-gate devices stop retaining data when they have been erased too many times—a hard-failure mechanism—FRAM does not exhibit this type of wearout.

Third, FRAM operates without a charge pump. Floating-gate technologies use high-voltage to program a new state, so that write operations use much more power than read op-

erations. FRAM writes at the process core voltage, be it 5 V, 3.3 V, or lower. Furthermore, the energy consumed by FRAM is orders of magnitude lower when the lower power consumption is multiplied by the much faster write speed.

These benefits collectively help system designers working on write-intensive applications where the system must write data either frequently or quickly. FRAM offers no special bene-

Why embed the memory?

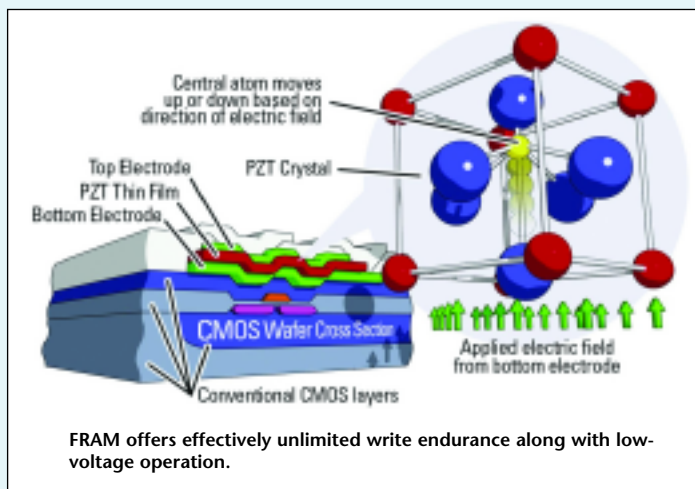
Since the benefits of FRAM are available from a standalone memory device, what is the motivation to embed it?

Embedded memories offer higher performance than standalone memories by taking advantage of wider internal memory buses and by eliminating off-chip I/Os. Bus widths of 32 bits or more are easy to achieve on-chip but create packaging and board-

level interconnect challenges for a discrete memory. Eliminating chip-to-chip interfaces can also reduce cycle times by 10 to 15 ns. As an example, a 55-ns cycle time FRAM memory with a x16 organization delivers a memory bandwidth of roughly 290 Mbits/s. The same core technology implemented on an embedded MCU might achieve a 40 ns cycle time by eliminating the chip-to-chip interfaces. Doubling the

memory array bus width to a x32 organization would deliver 800 Mbits/s, a 2.75X speed improvement with the same underlying technology.

Embedded FRAM simplifies system design. It allows the designer to remove fixed partitions that naturally occur between flash, RAM and EEPROM in a microcontroller (MCU) due to the differing capabilities of those technologies. For example, it is usually impossible to allocate SRAM for program memory if the application program size exceeds the amount of flash on chip, and flash cannot be used as working memory if the SRAM runs low. A single FRAM array can functionally perform the tasks of all



fits for read-intensive applications.

As an in-system programmable nonvolatile memory, FRAM can serve in the place of any other memory choice, including flash or EEPROM. However, FRAM is less mature and more expensive than these other memory technologies. Therefore, it is typically used when the three aforementioned benefits are especially useful in the application, such as write-intensive applications featuring data collection or potentially frequent configuration changes. The benefits also apply where the cost of service may be high and it is better to use a solution without a built-in failure mechanism.



of these memory types, eliminating the partitioning.

In a representative 8-bit MCU, there might be 16 Kbytes of flash, 1 Kbyte of SRAM, and 256 bytes of EEPROM. In this situation, if any individual requirement for code, working data, or configuration increases beyond the particular memory size, the MCU must be changed. A 16-Kbyte FRAM

would allow the designer to use memory for any purpose as long as the total byte count for all types of memory did not exceed the device capacity.

Advantages of standalone FRAM

Applying a nonvolatile memory as an embedded solution has a number of benefits. Why then would anyone fa-

vor a standalone memory? A major reason is cost. When comparing the cost of a standalone memory to an embedded memory, the wafer cost adder for the memory process over a plain logic process is a major factor. Since all transistors on a wafer cost the same, the cost adder for putting memory on chip directly increases the cost of the logic. For example, if the cost adder for a nonvolatile memory process is 30%, an MCU with nonvolatile memory costs 30% more than one without it, regardless of the size of the memory.

The logic itself costs more on an MCU with embedded memory. Today FRAM, as a less mature technology, has a higher wafer cost adder than either flash or EEPROM. Therefore, it penalizes the cost of the MCU more than other memories do.

***Nonvolatile
FRAM performs
write and read
operation at the
same time.***

In considering actual cost, there are other factors such as memory cell size, array efficiency and yield. These differences largely apply to standalone memory in the same way as embedded memory. The penalty of the cost adder on the logic is the clearest difference between embedded and standalone memory.

FRAM is currently running on older processes such as 0.35- μm CMOS. More advanced choices are expected in the near future, but today embedded FRAM is restricted mainly to 8-bit MCUs and to memory sizes of 1 Mbit or less. More foundries and more advanced process will eliminate this disadvantage.

FRAM simplifies design

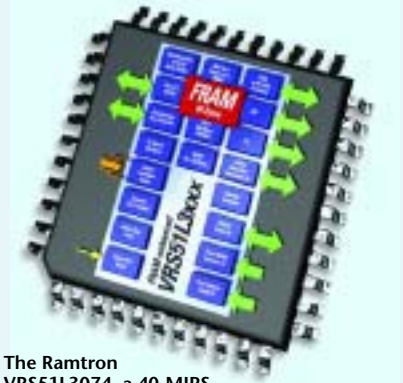
The simplicity of system design with FRAM means that every designer would prefer it over the basket of embedded memory choices for typical MCUs. However, considering the cost, FRAM is primarily used in applications that take maximum advantage of its unique benefits. These are data-intensive applications, such as in-car

<http://epinfo.us/5758-43>

DVD players and industrial motion sensors.

For instance, the resume-play feature enables a DVD player to remember the disk location where playback stopped on power-down. Home players remain powered so this data are retained in SRAM. In a portable or automotive environment, the standby power used

makes it preferable to power-down completely. Therefore, nonvolatile storage is needed. Since the DVD tracking information is stored about every 3 s it would wear out alternative memory choices. For industrial motion sensing, the position information occurs too rapidly to tolerate the write delays of either flash or EEPROM.



The Ramtron VR551L3074, a 40-MIPS 8051-based microcontroller, has 8 Kbytes of FRAM along with 64 Kbytes of flash.

A multichip package approach

One possibility for bridging the gap between embedded and standalone memory is the use of multichip packages (MCP). This allows a tighter integration between processor and mem-

Embedded FRAM is restricted mainly to 8-bit MCUs and to memory sizes of 1 Mbit or less.

ory than in a standalone situation if the underlying devices are designed for an MCP approach. It does not capture all of the embedded benefits, but is an improvement over a standalone memory.

If the processor-to-memory interaction is designed for an MCP, the memory can be mapped as a processor resource rather than an external device. From a firmware perspective the in-package memory would function the same way as an embedded memory.

The processor and memory become inseparable, despite there being two die. Depending on the die-to-die interface, speed will be higher than a discrete implementation, but not as fast as a truly embedded memory.

Board space and design simplicity are well served by MCP. The tradeoff between true embedded FRAM and MCP is mainly one of performance. Only the end application can determine the best approach. If the main driver for using FRAM is simplicity, power consumption, or high write endurance, then an MCP solution represents the best of both worlds. ■

<http://epinfo.us/5758-45>