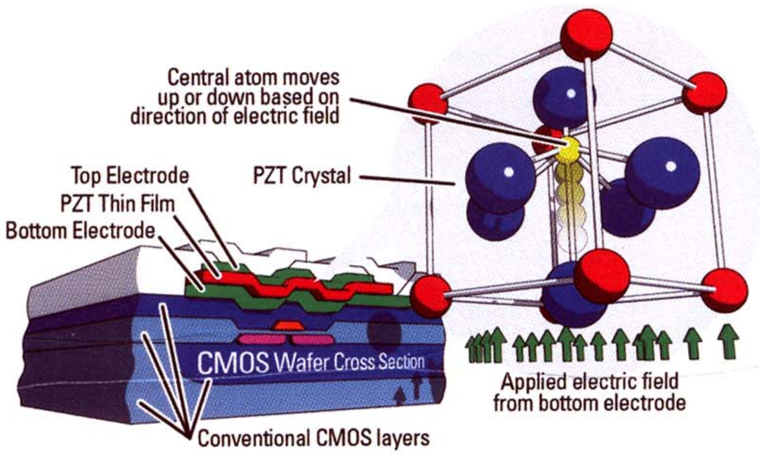
Quick on the draw

Fast write times, unlimited write endurance, low power: Mike Alwais takes a look at the case for embedded Ferroelectric RAM

FRAM crystal and CMOS architecture



Three distinct benefits account for the rise in popularity of FRAM. Firstly, it performs write operations at the same speed as read operations. Referred to as writing at bus speed, no delays are needed for the write data to become nonvolatile. This is unlike nonvolatile memories based on floating gate technology that have longer write delays (for example, in a typical EEPROM write operation, it takes 10 milliseconds for the write to be effective after the data is written to the input buffer). In addition, with FRAM there is no erase operation, since there is no preferred or default state. Data is written like with SRAM and other RAM technologies, that is, without regard to the previous state.

Secondly, 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. This is a hard failure mechanism; a fatigued memory cell can no longer store the programmed state. FRAM does not exhibit this type of wear out.

Thirdly, FRAM operates without a charge pump, enabling low power consumption. Floating gate technologies use high voltages to program a new state, giving write operations much higher power consumption than read operations. FRAM writes at the process core voltage, be it 5V, 3V or lower for more advanced processes.

These benefits collectively help system designers of write intensive applications where the system must write data either frequently or quickly. FRAM offers no special benefit for read intensive applications.

As a nonvolatile memory that can be written in-system, FRAM is capable of serving in place of any other memory choice such as Flash or EEPROM. However, the technology is less mature and more expensive than these technologies. Therefore it is typically used in cases where the three benefits mentioned above are needed in the application. These are write intensive applications featuring data collection or potentially frequent configuration changes. The benefits also apply in situations where the cost of service may be high and it is better to use a solution without a built-in failure mechanism. Since the benefits of FRAM are available from standalone memory devices, what is the motivation to embed it?

Why embed the memory?

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 nanoseconds. As an example, a 55ns cycle time FRAM with a x16 organisation delivers a memory bandwidth of roughly 290 million bits per second. The same core technology implemented on an embedded MCU might achieve a 40ns cycle time by eliminating the chip-to-chip interfaces. Organising that memory array with a x32 bus width would then deliver 800 million bits per second, a 2.75x speed improvement with the same underlying technology.

Embedded FRAM also simplifies the system design. It allows the designer to remove fixed partitions that naturally occur between Flash, RAM and EEPROM in an MCU due to the differing capabilities of those technologies. For example, it is usually not possible to allocate SRAM to serve if the application program size exceeds the amount of Flash on chip. Equally Flash cannot be used as working memory if the SRAM runs low. A single FRAM array can functionally perform the tasks of all of these memory types, eliminating the partitioning.

In a representative 8-bit MCU there might be 16KB Flash, 1KB of SRAM and 256B EEPROM. In this situation, if any individual requirement for code, working data or configuration increases beyond the particular memory size then the MCU must be changed. A 16KB 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. Additionally the external interface to the FRAM is removed, simplifying the hardware and reducing board space.

Multi-chip packaging

One possibility for bridging the gap between embedded and standalone memory is the use of multi-chip packages, especially if the underlying devices are designed for an MCP approach. This allows a tighter integration between processor and memory than in a standalone situation.

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 in the same way as 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. However, if the main driver for using FRAM is simplicity, power consumption or its high write endurance, then an MCP solution is the best of both worlds.

Ramtron | www.ramtron.com

* *Mike Alwais is Vice President of Marketing at Ramtron*