Half-Wits: Software Techniques for Low-Voltage Probabilistic Storage on Microcontrollers with NOR Flash Memory

Mastooreh Salajegheh*, Yue Wang†, Anxiao (Andrew) Jiang†, Erik Learned-Miller*, Kevin Fu*
*Department of Computer Science, University of Massachusetts Amherst
†Department of Computer Science and Engineering, Texas A&M University
{negin,kevinfu,elm}@cs.umass.edu, {yuewang,ajiang}@cse.tamu.edu

Abstract

This work analyzes the stochastic behavior of writing to embedded flash memory at voltages lower than recommended by a microcontroller's specifications to reduce energy consumption. Flash memory integrated *within* a microcontroller typically requires the entire chip to operate on common supply voltage almost double what the CPU portion requires. Our approach tolerates a lower supply voltage so that the CPU may operate in a more energy efficient manner. Energy efficient coding algorithms then cope with flash memory that behaves unpredictably.

Our software-only coding algorithms (*in-place writes*, *multiple-place writes*, *RS-Berger codes*) enable reliable storage at low voltages on unmodified hardware by exploiting the electrically cumulative nature of half-written data in write-once bits. For a sensor monitoring application using the MSP430, coding with in-place writes reduces the overall energy consumption by 34%. In-place writes are competitive when the time spent on computation is at least four times greater than the time spent on writes to flash memory. Our evaluation shows that tightly maintaining the digital abstraction for storage in embedded flash memory comes at a significant cost to energy consumption with minimal gain in reliability.

1 Introduction

Billions of microcontrollers appear in embedded systems ranging from thermostats and utility meters to tollway payment transponders and pacemakers. Many of these systems use on-chip flash memory for storage.

While the reliability, low cost, and high storage density of flash memory make it a natural choice for embedded systems [2], its relatively *high voltage requirement* (Table 1) introduces challenges for energy-efficient designs aiming to maximize the system's effective lifetime. Instrumenting the system to operate at a fixed low voltage v_l is one way to reduce power consumption; however, achieving *consistently correct* results for flash writes are guaranteed only if v_l is higher than a manufacturer-specified threshold. Moreover, in energy-limited devices that cannot provide a constant supply voltage, scenarios may arise in which the flash memory is the only part of the circuit whose operating requirements are not met. In such cases, applications can expect normal operation when they are not performing flash writes and unpredictable behavior when they are.

Microcontroller	CPU Min. voltage	Flash write Min. voltage
TI MSP430	1.8 V	2.2 or 2.7 V
PIC32M	2.3 V	3.0 V
ATmega128L	2.7 V	4.5 V

Table 1: Flash memory restricts choices for the CPU voltage supply on microcontrollers because the CPU shares the same power rail as the on-chip flash memory.

Because embedded flash memory typically shares a common voltage supply with the CPU (separate power rails are cost prohibitive), a single voltage must be chosen that satisfies different components with different minimum voltage requirements. Current embedded systems address the voltage limitations of flash memory in one of the following ways:

i) A system can choose a high supply voltage sufficient for both reliable writes to flash memory and reliable CPU operation. This is a common choice for embedded systems with on-chip flash memory, but causes the CPU to consume more energy than necessary.

ii) A system can choose a low supply voltage sufficient for CPU operation, but insufficient for reliable writes to flash memory. This choice allows the energy source to last longer and for the CPU to compute more efficiently.

This paper appears in the Proceedings of the 9th USENIX Conference on File and Storage Technologies (FAST '11), San Jose, CA, February 2011. `http://www.usenix.org/events/fast11/tech/full_papers/Salajegheh.pdf`.

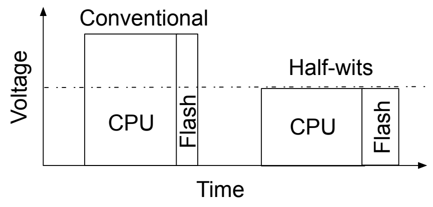

Figure 1: Operating at a lower voltage and tolerating errors instead of the conventional case of choosing the highest minimum voltage requirement may help decrease energy consumption. Considering that *Power* $\propto$ *voltage*^{2}, decreasing voltage decreases the power consumption quadratically.

An example of such a system is the Intel WISP [5], a batteryless RFID tag that excludes access to flash memory to save power.

iii) A system can modify hardware to enable dynamic voltage scaling. This approach requires additional analog circuitry such as voltage regulators and GPIO-controlled switches. Because many embedded systems are extremely cost sensitive, this choice is unattractive for high-volume manufacturing with low per-unit profit margins. An additional 50 cent part on a thermostat control can be cost prohibitive. Moreover, small changes may necessitate a new PCB layout—upsetting the delicate supply chain and invalidating stocked inventories of already fabricated PCBs.

Approach. Our approach reduces the operating voltage of the microcontroller to a point at which the resulting power savings of the CPU portion of the workload exceeds the power cost of the algorithms for ensuring reliable writes (Figure 1). Our low-power storage scheme benefits from the accumulative property of flash memory by repeating writes to the same cell. Each write operation will increase the chance of success by forcing some number of state transitions. That is, a failed write is still progress.

Of wits and half-wits. In 1982, Rivest and Shamir introduced the notion of write-once bits (wits) in the context of coding theory to make write-once storage behave like read-write storage [4]. Bits in flash memory behave like wits because a programmed bit cannot be reprogrammed without calling an energy-intensive erase operation to a block of memory much larger than a single write. We coin the term *half-wits* to refer to wits used in a manner inconsistent with a manufacturer's specifications, resulting in stochastic behavior. Half-wits in this work are wits of flash memory used below the recommended supply voltage.

In examining error rates at low voltage and constructing a system that provides reliable storage despite errors, our work suggests that it is appropriate to relax previously assumed constraints and reexamine the costly digital abstractions layered above on-chip flash memory.

Contributions. Our primary contributions include an empirical evaluation that characterizes the behavior of on-chip flash memory at voltages below minimum levels specified by manufacturers, and algorithms that enable reliable writes to flash memory while coping with low voltage. Our evaluation identifies three key factors affecting error rates: voltage, Hamming weight of the data, and the wear-out history of the flash memory.

The first algorithm, *in-place writes*, makes attempts at *write time* to store a value correctly in the given memory address. The *in-place writes* method repeatedly writes data to the same memory address. The intuition behind this approach is that repeating a write attempt in a consistent location accumulates the charge in the same cell, increasing the chance of storing a bit of information correctly. In addition, since flash writes only change bits in a single direction, a correctly written bit cannot be reversed to produce an error on a second write attempt. The second algorithm, *multiple-place writes*, tries to decrease the probability of error by making attempts at both write time and read time. This method stores data in more than one location aiming that the data (even partially) will be stored correctly in at least one of these locations. The third algorithm is a hybrid error-correcting code combining Reed-Solomon (RS) [3] and Berger [1] codes. The Berger code detects, but does not correct, asymmetric errors caused by the low write voltage. Given the approximate locations of errors, which are determined by the Berger code, the RS code efficiently recovers the originally stored data.

The paper compares all three methods in terms of energy consumption, execution time, and error correction rate. We also show that our methods are most effective for CPU-bound workloads. With respect to cost and energy, our techniques may enable already deployed embedded flash memory to remain competitive with emerging technology for low-power, non-volatile memory.

References

[1] J. Berger. A note on error detection codes for asymmetric channels. *Information and Control*, 4(1):68–73, 1961.

[2] A. Jiang, R. Mateescu, M. Schwartz, and J. Bruck. Rank modulation for flash memories. In *IEEE International Symposium on Information Theory (ISIT)*, pages 1731–1735, 2008.

[3] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[4] R. L. Rivest and A. Shamir. How to reuse a write-once memory. *Information and Control*, 55:1–19, 1982.

[5] A. P. Sample, D. J. Yeager, P. S. Powledge, A. V. Mamishev, and J. R. Smith. Design of an RFID-based battery-free programmable sensing platform. *IEEE Transactions on Instrumentation and Measurement*, 57(11):2608–2615, Nov. 2008.