Designing Low-Energy Embedded Systems from Silicon to Software

Part 1 – Silicon Choices

Introduction

Low-energy system design requires attention to non-traditional factors ranging from the silicon process technology to the software that runs on microcontroller-based embedded platforms. Closer examination at the system level reveals three key parameters that determine the energy efficiency of a microcontroller (MCU): active power consumption, standby power consumption and the duty cycle, which determines the ratio of time spent in either state and is itself determined by the behavior of the software.

A low-energy standby state can make an MCU seem extremely energy efficient, but its true performance is evident only after taking into account all of the factors governing active power consumption. For this and other reasons, the tradeoffs of process technology, IC architecture and software construction are some of the many decisions with subtle and sometimes unexpected outcomes. The manner in which functional blocks on a microcontroller are combined has a dramatic impact on overall energy efficiency. Even seemingly small and subtle changes to hardware implementation can result in large swings in overall energy consumption over the lifetime of a system.

Low-Energy Applications

Metering and alarm systems, for example, are often powered for 10 years by a single battery. A small increase in current consumption for a sensor reading (of which hundreds of millions may occur over the lifetime of the product) can result in years being lost from the product's actual in-field lifetime. A simple smoke alarm that detects the presence of smoke particles in the air once a second will take 315 million readings during its lifespan.

The activity ratio or duty cycle of a simple smoke alarm is relatively low. Each sensor reading may take no more than a few hundred microseconds to complete, and much of that time is spent in calibration and settling as the microcontroller wakes up the analog-to-digital converters (ADCs) and other sensitive analog elements and allows them to reach a stable point of operation. In this case, the duty cycle is likely to lead to a design that is inactive approximately 99.98 percent of the time.

A traditional smoke alarm is comparatively simple. Consider a more complex RF design in which results are relayed over a sensor mesh to a host application. The sensor needs to listen for activity from a master node so that it can either signal that it is still present within the mesh network or provide recently-captured information to the router. However, this increased activity may not affect the overall duty cycle. Instead, more functions may be performed during each activation period using a higher-performance device. Because of its increased processing speed (made possible by a more advanced architecture and semiconductor technology), the faster device can provide greater energy efficiency than a slower device running for more cycles. The key lies in understanding the interaction between process technology, MCU architecture and software implementation.

Part One: Silicon Choices

CMOS Energy Profile

Practically all MCUs are implemented using a CMOS technology. The power consumption of any active logic circuitry is given by the formula CV^2f where: C is the total capacitance of the switching circuit paths within the device, V is the supply voltage, and f is the operating frequency. (See Figure 1.) The voltage and capacitance are factors of the underlying process technology. Over the past three decades, the on-chip operating voltage of CMOS logic has fallen from 12 V to less than 2 V as transistors have scaled down in size. Because voltage is a quadratic function in the active-power equation, the use of lower voltages has a significant impact.

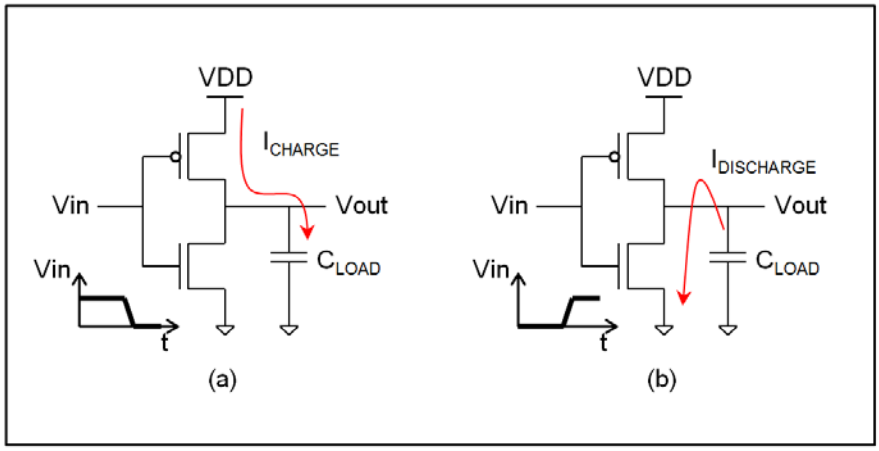

Figure 1. CMOS Logic Structure and Energy Consumption during Switching

Although the capacitance term is linear, the factors that lead to reductions in its overall level are also assisted greatly by Moore's Law scaling. A more recent process will, for a given logical function, offer lower capacitance than its predecessors and, with it, lower power consumption. In addition, advanced design techniques make it possible to reduce the overall switching frequency by only operating circuits with actual work to perform, a technique known as clock gating.

Compared to other technologies, CMOS dramatically reduces wasted energy; however, leakage current remains. In contrast to active power consumption, the leakage increases with Moore's Law scaling and needs to be taken into account in any low-energy application because of the proportion of time that a low duty-cycle system is inactive. However, as with active power consumption, circuit design has a dramatic impact on real-world leakage. Analogous to clock gating, power gating can greatly ameliorate the effects of leakage and make more advanced process nodes better choices for low duty-cycle systems, even though an older process technology may offer a lower theoretical leakage figure.

Appropriate Process Technology

There is an appropriate process technology for every feature set. The answer is not to simply rely on one process technology that has the lowest theoretical leakage just because the device will spend a long time in sleep mode. During sleep mode, it is possible to disable power to large segments of the MCU, taking the leakage component out of the equation. Leakage is a bigger issue when circuits are active, but can easily be outweighed by the advantages of more advanced transistors that switch far more efficiently.

As an example, the leakage current of a 90 nm process versus that of a dedicated low-power 180 nm process is approximately five times higher. The active mode power consumption is a factor of four lower, but this is based on a far larger figure. Take a 180 nm microcontroller with an active current consumption of 40 mA and a deep-sleep mode consumption of 60 nA and compare these power levels to those of a 90

nm implementation that is able to drive the active current draw down to 10 mA but suffers from a higher sleep mode current of 300 nA.

In the previous example, the MCU must be active for 0.0008 percent of the time for the 90 nm implementation to be more energy efficient overall. In other words, if the system is active one second per day, the 90 nm version is approximately 1.5 times as energy efficient as its 180 nm counterpart. The conclusion is that it is important to understand the application duty cycle when selecting a process geometry. (See Figure 2.)

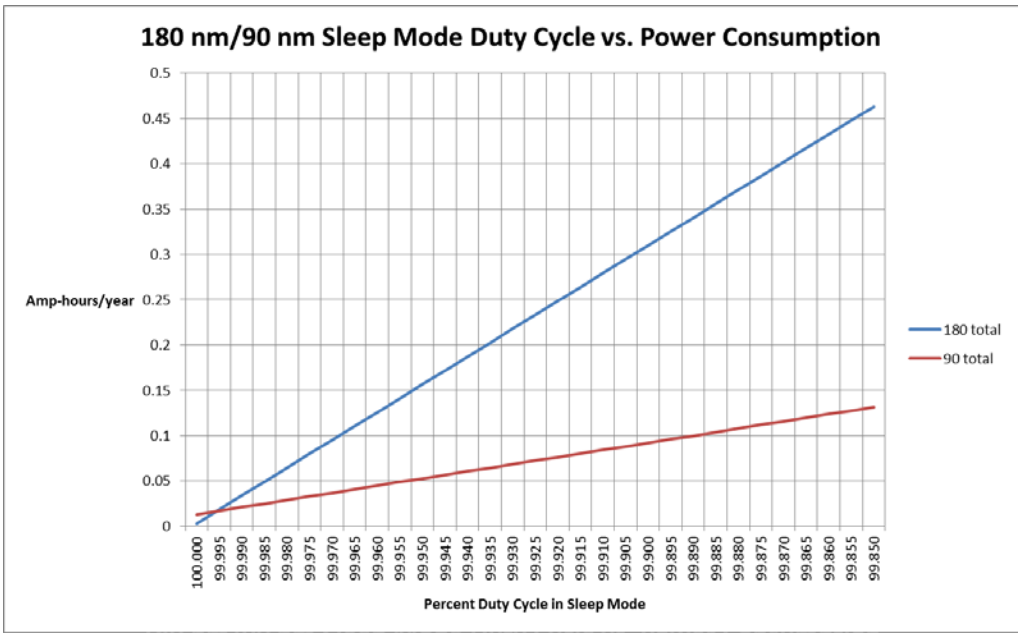

Figure 2. Process Technology and Duty-Cycle Tradeoffs

Once the appropriate process technology has been selected, the IC designer has the option to further optimize energy performance. When first introduced, the concept of clock gating was applied at a relatively coarse level. Clock gating increases the complexity of a design because the circuit designer needs to be aware of which logic paths will require a clock signal at any given time.

Clock Distribution

Most microcontroller implementations use a hierarchical structure to distribute clock signals and the appropriate voltage levels to each part of the IC. The functional units, such as instruction processing blocks and peripherals, are organized into groups. Each of these groups will be fed by a separate clock tree and power network. The clock signal for each group is derived from a common clock source by a frequency divider or multiplier. Similarly, the voltage delivered to each group of peripherals will be controlled by a set of power transistors and voltage regulators if the groups require different voltages (an approach that is becoming increasingly common).

To minimize design complexity, MCUs have used a relatively simple clock-gating scheme in which entire clock trees are disabled as long as no functional units inside a group are active; however, this allows logic that is performing no useful work to be clocked in groups that are active. For example, the adder unit in a CPU core might receive a clock even if the current instruction is a branch. The switching triggered by the clock signal within that adder increases power consumption by a factor of CV^2f, as described earlier.

Improvements in design tools and techniques have made it possible to increase the granularity of clock gating to the point where no peripheral or functional unit receives a clock signal if it has no work to perform during that cycle.

Voltage scaling provides further potential energy savings by making it possible to deliver a lower voltage to a particular group of functional units when required. The key to delivering the appropriate voltage to a group of functional units or peripherals lies in the implementation of on-chip voltage regulators or dc-dc converters and the use of monitoring circuits to ensure that the IC operates at the required voltage.

Power-Supply Considerations

On-chip voltage regulators provide the system designer with greater flexibility, making it possible to extract more charge from a battery. For example, an on-chip switching buck converter (like the ones found in Silicon Labs' SiM3L1xx series products) can be used to take the 3.6 V of an industrial battery and convert it to 1.2 V at more than 80 percent efficiency. Many MCUs do not have this feature and use linear components to drop the voltage to the right level with a greater degree of waste. In advanced implementations, the buck regulator can be switched off when the battery has discharged to such a level that it no longer makes sense to perform the conversion. As a result, the power supply can be optimized for energy efficiency over the lifetime of the device, all under software control.

Conclusion

Low-energy system design is a holistic process that is enabled by choosing a combination of the right silicon, software and development tools. By mastering the relationship between each of these variables, systems engineers can develop higher performance and more energy-efficient embedded systems that stretch the limits of battery-powered applications.

Part 2 of "Designing Low-Energy Embedded Systems from Silicon to Software" explores software decisions that the developer must make to optimize the embedded system for the lowest possible power consumption.

# # #