Optimizing Low Power Embedded Designs

By Sachin Gupta, Applications Engineer Sr, and Madhan Kumar, Applications Engineer, Cypress Semiconductor Corp.

Low-power embedded design is motivated by the desire to be able to run applications for as long as possible while consuming the minimum power. In a battery-powered system, this need is magnified. Furthermore, low power implies lower cost of operation and smaller battery size to make applications more mobile. When energy comes at a premium as it does with today's green initiatives, ensuring that an embedded design consumes as little energy as possible is even important for wall-powered applications.

Designing power-efficient applications also ensures less overhead to manage thermal dissipation, and heat generation is controlled at the source by optimizing the power consumed. Given these advantages, embedded application designs today can no longer ignore the problem of optimizing power. This article will focus on the major factors contributing to power consumption in an embedded system by analyzing the various power modes which most microcontrollers offer. Then, we will analyze a real-life example of an embedded application in terms of power consumption and how its efficiency can be maximized.

MCU Power Consumption

To begin, there are several points to be aware of when selecting an MCU or external components. The overall power consumption of a MCU is defined by its power consumption in different modes, typically active and standby (includes sleep, hibernate, etc.), and taking into account the power consumed to transition from one mode to another. Let us go through these power modes in more detail.

Active power consumption by an MCU is the power consumed when the MCU is running. As almost all controllers are based upon CMOS logic, power is consumed primarily during the switching of transistors. As a starting point, let us analyze the power consumption of a CMOS inverter (see Figure 1) which is the basic building block of any CMOS design.

![CMOS inverter diagram](https://via.placeholder.com/150)

**Figure 1: CMOS inverter**
CMOS circuits dissipate power by charging the various load capacitances whenever they are switched. When considering internal architectures, this is mainly the gate capacitance but there are drain and source capacitances too. Power is dissipated across the PMOS transistor while the load capacitor is being charged and across the NMOS when the load capacitor is being discharged. Instantaneous power dissipation across the NMOS transistor of a CMOS inverter (figure 1) is given by equation 1.

\[ P_{PMOS} = i_L(V_{dd} - V_o) \]  

After substituting the value of \(i_L\) in equation 1:

\[ P_{PMOS} = C_L(V_{dd} - V_o) \frac{dV_o}{dt} \]  

Total power dissipation across the PMOS to switch the output from low to high can be found by integrating power dissipation across the PMOS to the charge load capacitor from 0 V to Vdd:

PMOS power consumption, \(P_{PMOS} = \frac{1}{2} C_L V_{dd}^2 \)  

Similarly, to switch the output from high to low, total power dissipation across the NMOS is:

NMOS power consumption, \(P_{NMOS} = \frac{1}{2} C_L V_{dd}^2 \)  

For one switching cycle, then, power dissipation is:

\[ P_{Total} = P_{NMOS} + P_{PMOS} = C_L V_{dd}^2 \]  

If we define the average power in terms of the switching frequency \((f)\), we get:

\[ P = fC_L V_{dd}^2 \]  

From equation 6 it can be seen that power consumption depends upon the switching frequency, load capacitance, and supply voltage. Load capacitance is determined by the technology parameters and the design layout, and is therefore beyond the control of the embedded system designer. However, the other two factors – switching frequency and supply voltage – are factors a system designer can modify with an intention to impact power efficiency for a given microcontroller. Of course, the value of these parameters is also heavily dependent on the application of the design.

However, modern controllers run at an internal regulated voltage irrespective of the input voltage on the supply pins. There are controllers available in market which can be operated from 0.5 V to 5.5 V, but the internal core runs at a fixed regulated voltage such as 1.8V irrespective no matter what the supply voltage is. Therefore, this parameter is not as important in the case of modern controllers as it was in the past. However, it is good to keep the supply voltage to the minimum requirement for regulators or near the voltage where the regulator is bypassed.

This leaves system designers have just one parameter available for affecting power control: switching frequency. Hence, in the active mode the minimum required operating speed for the MCU should be calculated and higher clock speeds should be avoided.

**Standby Power**

The other major factor which determines battery life is the standby power consumption of an embedded system. Most applications can spend significantly periods of time in standby mode. In these systems, the major contributor towards total system power consumption is the standby current rather than the active current. Standby current is the sum of leakage current, current consumed by power management circuits, clocking systems, power regulators, RTC, IOs, interrupt controllers, and so on. It varies from controller to controller, based upon the particular features and peripherals supported in standby mode.

Finally, the power consumed while transitioning from low power mode to active mode should not be overlooked. Devices may end up wasting a significant amount of power while transitioning between these two modes.

Based on these power modes, an MCU’s average power consumption is:

\[ MCU \text{ average power consumption} = (\text{Active Power} + \text{Sleep Power} + \text{Transition Power}) / \text{Total Time} \]
Where, Active Power = Time for which MCU is active * Active Current

Sleep Power = Time for which MCU is in sleep * Sleep Current.

Transition Power = Power consumed while making transition from sleep to active mode

The amount of time the system remains in active and standby mode is application-dependent. Some applications may need to have MCU running all the time while some may need to have it running only occasionally. There are MCUs available on the market which come with additional power down modes other than sleep, such as hibernate mode, deep sleep mode, or shut down mode in which power consumption can be on the order of 10s of nA. System designers need to look at the power consumption of the mode in which the system has to operate for the majority of time to ensure that the overall design is as power efficient as it can be.

If we look deeper, there are some vital tradeoffs that must be considered. For some applications, it could prove beneficial for the system to run at a higher speed so it can finish its job faster and return to a low power mode. Alternatively, other systems may do better running at a slower speed to keep active power consumption low. Here, the system designer has to analyze the best case for the application considering the current at different operating speeds, the time it takes to come out of low power mode, the current consumption in low power mode, and the frequency with which the system needs to switch between active and sleep modes.

**Peripheral Power**

MCU power consumption is only one factor when considering system power consumption. Sometimes, engineers tend to concentrate too much on the MCU and ignore the power consumed by external peripherals. If the objective is to optimize the power consumption of the entire embedded system solution, one cannot afford to do this.

Consider a simple temperature measurement system for home use (see Figure 2).

![Figure 2: Temperature monitoring system](http://www.low-powerdesign.com/article_cypress_093010.html)
Total power consumption = MCU power consumption + ADC power consumption + DAC power consumption + LCD power consumption

For this system, the sample rate need not be very high since temperature does not change rapidly. Power consumption can be kept to a minimum by switching on the ADC and DAC only when required and optimizing the time ratio for which the MCU stays in active and standby mode. If this system is made of discrete components, it can get quite challenging to coordinate. Power consumption for discrete component-based architectures will look similar to the system shown in Figure 3.

Figure 3: Power consumption system using discrete-based designs

In Figure 3, standby current is mainly contributed to by MCU standby current and the active current of the ADC, DAC and LCD. The ADC may have an option by which the MCU can stop ADC conversion before it goes to sleep to save ADC power consumption as well. However, there will still be some standby current for ADC and similarly for the DAC. Alternatively, the system could be implemented using a system-on-chip (SoC) architecture where all of the peripherals are integrated onto a single chip along with the ability to control the power of each individual peripheral. Power consumption of this system will look similar to shown in Figure 4 and can lead to a dramatic reduction in power consumption compared to a discrete component-based implementation.
While designing any system, we should use what is needed rather than simple using what is available. When requirements are overestimated, such as choosing a faster or more sophisticated component than is needed, this results in higher cost and lower power efficiency. For example, a 20-bit ADC running at 1 Msps is clearly more than is needed for a temperature measurement application. In addition, the ADC needs a high-frequency operating clock to sample at this rate.

Advancements in SoC technology allow developers to access a wide range of on-chip peripherals such as filters, ADCs, DACs, Op-amps, and programmable analog and digital blocks. For example, PSoC devices from Cypress Semiconductors have a wide operating frequency range with programmable clock sources for different blocks, including the MCU, and support numerous power management modes. These modes range from active mode where all the features on the device are available to hibernate modes where current can be as low as 100 nA while retaining the contents of configuration registers and RAM.

As complex as SoC architectures are, they represent almost the complete system, making it more straightforward to compute power consumption. For example, if the system is doing nothing, then the standby current of the whole system can be as low as 100 nA. Since peripherals and the MCU can be switched on or off individually, only the appropriate blocks need to resume operation after the next wake up event. This is one of the key features which an SoC can have from a system point of view.

In some systems, it is possible that for a certain period only some hardware functions are needed and not the MCU, such as when generating a waveform using a DAC. This task can be completed by the DMA (Direct Memory Access) and DAC without the MCU, and so the MCU can be switched off. SoCs enable developers to design ultra low power embedded systems that are also cost and space efficient, with the added advantage of fast time to market.

A system’s average power calculation in a SoC-based system becomes more complex since along with the average MCU current, we need to consider each the operating state of each individual peripheral on the chip. Average system current is:
Battery Life

Battery life is a critical specification for any battery-powered application. Battery ratings are given in the units of mA - Hr, meaning the battery can supply 'X' mA of current for one hour. If we know the average current, we can calculate battery life:

\[
\text{Battery Life} = \frac{\text{Battery rating}}{I_{\text{avg}}} \quad (9)
\]

Equation 9 gives the battery life in hours if \( I_{\text{avg}} \) is given in mA.

Power consumption is one major factor which almost every consumer cares about in both wall-powered and battery-powered devices. In today's competitive market, designing a product which consumes higher power or which costs more than competitive products can result in reduced market success. When optimizing power consumption is a major criterion, designers should look at critical parameters like choosing the appropriate components and making sure that they are not overrated for the desired end application, as well as making sure the system does not operate at higher speeds than are required. In addition, developers will want to seriously consider how long the system spends in active and standby modes and the relative power consumption in each.