Designing in Low Power

An Overview of the Power Saving Mechanisms used by Motorola's M•CORE Architecture

Overview

Portable consumer electronics, such as cellular phones, and GPS and messaging devices, represent a lucrative new market opportunity for embedded systems applications. Developers of portable systems have one over-riding concern — battery life. This need has driven demand for a new type of embedded processor: one which has minimal power consumption yet provides the processing performance required by sophisticated applications.

The traditional approach to developing low-power processors has been to spin variants of existing architectures. However, these architectures were originally designed to provide optimal performance rather than optimal power consumption. To meet the demands of portable applications, a processor must be designed from the ground up to optimize power consumption.

Motorola's M•CORE architecture is the first processor designed specifically for sophisticated, yet low power, applications. The M•CORE architecture provides very low voltage cores and combines a number of attributes to achieve outstanding power consumption. In addition to providing mechanisms to power down the processor and system logic, the M•CORE architecture focuses on minimizing dynamic power consumption when the system is active.

By combining the performance of 32-bit processor with the cost and power advantages previously available only with 16-bit architectures, the M•CORE architecture is an enabling technology for a new range of embedded applications aimed at the mass market.

The need for low power

Market requirements are driving the emergence of a class of highly portable, yet sophisticated devices. Digital cellular phones continue to get smaller and smarter. Messaging devices, such as pagers, add functionality such as e-mail. Finally, there are completely new applications, such as handheld GPS systems. An overriding concern for such applications is battery life, since it is highly inconvenient for users to have to constantly recharge a device or carry spare battery sets.

These devices, therefore, present the designer with a paradox. On the one hand, their small form factor prevents the use of heavy batteries. On the other hand, their sophistication demands more processing power. Since battery performance is not increasing as fast as the market requires, designers need improved electronics that provide high performance but also maximize battery life by reducing power consumption.

Low power consumption provides other benefits for portable consumer electronics. In some devices, it may be desirable to use a smaller battery to reduce weight, making a device easier to carry around for long periods. Another important driver for consumer electronics is, of course, cost. In addition to smaller batteries reducing cost, low power consumption helps reduce cost by enabling lower cost packaging to be used.

The advantages of low power consumption are not exclusive to portable devices. Although it might be counterintuitive to think that a car-borne device requires low power consumption, this is indeed the case for an airbag controller. Since airbags are most likely to be triggered in a serious collision, it is impossible to assume that the car's battery will continue to supply power to the vehicle's electronics. The airbag controller must therefore be able to function from a capacitor charge, necessitating minimal power consumption.

Low power is not something that is achieved simply by accident. It must be designed in — both at the architectural and implementation levels. In response to the market demand for a truly low power solution, Motorola has introduced the M•CORE architecture.

M•CORE: The low power choice

In the past, most "low-power" processors were variants of processors designed for desktop applications. Typically the die would be moved to a new process and shrunk to reduce power consumption and some power management functions added. However, as any notebook computer user can tell you, such variants do not lead to a sufficiently long battery life or a lightweight product. A major reason for this is that the original processor architecture was not designed with optimizing power consumption in mind.

Motorola's M•CORE architecture, designed specifically to provide high performance but minimal power consumption, sets a new benchmark for power consumption. The M•CORE architecture achieves its exceptionally low power consumption through a variety of mechanisms, each of which is discussed in more detail below.

System-level power management
To provide optimal static power management for the overall system, the M•CORE architecture provides three instructions (stop, wait, and doze) that enable external logic to disable power to parts of the system. Execution of any of these instructions causes the processor to assert the LPMD1-0 output signals in the manner described in Table 1.

LPMD1	LPMD0	Mode
0	0	STOP
0	1	WAIT
1	0	DOZE
1	1	normal

Table 1: Low power mode signal encoding

The external logic uses the LPMD1-0 inputs to determine exactly which parts of the overall system logic should be placed in a low-power state. The external logic can also place the processor in a low power mode by forcing the CLK input high.

Dynamic power consumption
Although reducing a system's static power usage achieves the greatest overall reduction in power consumption, a true low power solution must address the issue of dynamic power consumption. By dynamic power consumption, we are referring to the power required by the system when it is actually being used. The M•CORE architecture optimizes dynamic power consumption by both minimizing the power needed to execute an instruction and minimizing the number of bytes that need to be fetched to perform a given function.

"Power Aware" instruction pipeline
The low power instructions discussed earlier provide a mechanism to power down select parts of the system when not used. With processors themselves becoming more complex, a logical extension of this is to only power up the parts of a processor that are required to execute an instruction. The M•CORE architecture achieves this benefit through its advanced "power aware" pipeline. The instruction pipeline recognizes which processor functions are required to execute a particular instruction. This enables it to ensure that data is only transitioned through the processor blocks that are actually needed to implement the instruction. For example, an add instruction would cause data to transition through the adder but not through the barrel shifter. By eliminating unnecessary transitions, the M•CORE architecture prevents switching of gates, loads, and wires in unused blocks, all of which would otherwise consume additional power.

Code density
Code density is a measure of how many bytes of code are required to implement an application or function. Code density affects power consumption both statically and dynamically. The M•CORE architecture's high code density results in a smaller executable image. This reduces an application's memory requirements, which in turn reduces system cost and system power consumption.

However, there is a second benefit to code density. Every time the processor fetches an instruction from memory, it must use a bus cycle. Bus cycles, of course, consume power. Since the M•CORE architecture's

dense code allows it to perform equivalent functionality with fewer bytes of code, a program executing on an M•CORE processor will consume less power because it will fetch fewer bytes from memory.

Although M•CORE is a 32-bit architecture, it utilizes a 16-bit instruction set to achieve high code density. In addition, to providing improved code density, the 16-bit instruction set provides a performance advantage over conventional RISC architectures in many low-cost applications. It is common for such applications to minimize cost through use of a 16-bit bus. Since conventional RISC architectures use 32-bit wide instructions, they have to perform two bus cycles to fetch an instruction, negatively impacting overall instruction throughput. In contrast, the M•CORE architecture would only require a single bus cycle to perform an instruction fetch, enabling it to run at full speed even with a 16-bit bus.

Rich register set
To further minimize bus activity, the M•CORE architecture reduces the need to read and write data to and from memory. It achieves this by providing a rich set of registers that enables a program to keep data variables in memory while they are "live". The M•CORE architecture provides a total of 37 32-bit data registers that are available to system programmers, one set of 16 general purpose registers, an alternate register file with 16 registers, and 5 scratch registers.

Support for multiple data sizes
Some commonly used data types such as chars or shorts have 8- or 16-bit, rather than 32-bit, representations. This provides an additional opportunity for the M•CORE architecture to reduce power consumption when fetching data from memory. For example, the M•CORE architecture would only toggle the 8 bits required to read or write a char, minimizing power consumption by logic external to the processor core.

Low Voltage
Since dynamic power consumption is proportional to the square of the supply voltage required, lowering the voltage provides a disproportionately large boost to battery life. M•CORE processors are designed to require only 1.8 volts to operate, with future versions planned to use as little as 0.9 volts.

Conclusion

The arrival of portable mass-market consumer electronic devices has created a need for a processor that maximizes battery life through low power consumption. Motorola's M•CORE architecture represents the first processor specifically designed to meet the requirements of such devices. The M•CORE architecture combines high performance with exceptionally low power consumption through both static and dynamic power management techniques combined with high code density.

For more information, please refer to the M•CORE web site at http://www.motorola.com/mcore or call the Motorola Technical Resource Center at 800.521.6274.