Two recent product announcements nicely mark a crossroads in SoC architectural thinking. The iPhone, standing at the end of a long tradition in hardware-block-oriented SoC design, points aloofly into the past, and an H.264 high-profile CoDec chip from Mobilygen, coming from a very different tradition points into a possibly very different future. In a way, the two fingerposts not only indicate the heritages of their respective architectures, but the evolution of their companies, as well.

Let’s begin with the vastly over-exposed iPhone. There are few certain details about the internals of the device, because Apple tries to maintain a level of secrecy—in what is otherwise known as a profession that would warm the heart of Dick Cheney. But a few points are relatively well-established.

The heart of the iPhone is reportedly a Samsung-designed SoC. The chip includes a significant application processor, reputedly a big ARM-11, surrounded by a number of hardware blocks to handle the handset’s major functions: audio and video playback, a highly dynamic touch-screen display with the beginnings of a gesture-based interface, a similarly dynamic graphics user interface, a rather clunky 2 Mpixel still camera and a reputedly quite competent WiFi port.

Most of these functional blocks are autonomous, each comprising its own local memory and its own ARM processor. For example—though not for certain, as even off-the-record sources are very careful around the long arm of Apple’s retaliatory tendencies—the computing loads involved suggest that the baseband processor would need to be a modest-sized ARM-11 with DSP extensions. The WiFi block would require an ARM-9-class core along
with supporting hardware accelerators, and functions with lower computing loads, such as the audio engine, Bluetooth block and system control block, would each have an ARM-7.

The only one of these processors that would be in any way multifunction would be the applications processor, which would be called upon to run user-interface code, work with an accelerator to do graphics, do the image processing for the camera module, and in its spare time run applications software. If the design were running out of budget, the architects might even call on the applications processor to do the pixel-level signal processing normally done in a dedicated image signal processing core. But this would be a low-cost trick that would harm camera response time and frame rate, and would only work with limited imager resolutions.

There are very substantial advantages to this approach of assembling the SoC from autonomous blocks. For one, as ARM mobile segment manager James Bruce points out, it allows the system integrator to license completed—and therefore, presumably, verified—modules of the design from outside, rather than having to gain the expertise to do the implementation in-house. In a converged device such as the iPhone, and in a relatively technology-light OEM such as Apple, this has distinct advantages. It is also an obvious advantage in a program with a compressed schedule, such as the timeline the iPhone reportedly had after Samsung was selected to replace PortalPlayer as the SoC vendor.

A huge part of this advantage is that autonomy of the functional blocks vastly simplifies system-level modeling. It is not necessary to understand what worst-case use scenarios are or to model them, except for shared resources such as DRAM. Hardly any real-time tasks have to concurrently share a CPU core with other high-priority tasks, so much of the uncertainty in interrupt response, real task-switching latency, and priority assignment evaporates, leaving clarity.

In addition, by defining use scenarios beforehand, it is possible to not just idle but power-down large sections of the SoC most of the time. If the user is just listening to tunes, or just squinting at a video clip, much of the rest of the system can be powered off, leaving only enough interfaces alive to stream the media data into the handset and keep a watchful eye over the cellular link.

“Apple has a huge advantage in power management because the iPhone is essentially a closed system,” Bruce observed. “They can literally try all the allowable use scenarios. That will always give a better power profile than they could achieve with a general-purpose, programmable device.”

In comparison, consider a recently announced Mobilygen EnViE, an H.264 high-profile, high-definition video encoder-decoder (CoDec) SoC. In some ways the block diagrams of the iPhone system chip and the Molilygen CoDec look remarkably similar—a core of secret sauce surrounded by every interface you could need for a variety of use scenarios, and a carefully crafted pipe to hard-pressed external DRAM. Functionally, as well, each chip faces a combination of interface servicing, system management, data streaming and hard real-time tasks. But of course in detail the two chips differ entirely.

Mobilygen started life thinking about video CoDecs in terms of abstract algorithms and software implementation, not in terms of SoC design. That watermark remains in the company’s architectural approach. The heart of the Mobilygen chip is not a cluster of autonomous functional blocks, but a pair of proprietary real-time multi-threading processor cores, according to Mobilygen CTO Sorin Cismas. The multi-thread architecture allows a core to handle a mix of tasks from different functions on the same CPU, meeting hard real-time deadlines on each task. It also allows the CPUs to be very tolerant of memory latencies, because the cores can simply switch threads—in one cycle, by the way—on a cache miss.
Interestingly enough, this means that the hardware block diagram for the CoDec core—a pair of CPUs and some accelerators clustered around a data switch—is entirely different from the functional block diagram, which would look more like a three-stage pipeline.

Without loading too much freight onto what is, after all, a rather specialized architecture to do H.264 encoding and decoding at low power levels, it is worth observing that this represents a very different approach to system design from the autonomous hardware block school. Rather than dealing with system dynamics by isolating code streams from each other, it blends and centralized code execution, and ensures real-time performance by analysis, not overdesign. There are not several CPUs powered down on the chip during normal operation.

Perhaps such an architecture does come from thinking of the system not in terms of hardware blocks, but in terms of software tasks. The job of the SoC—with two CPU cores, an ARM-926 for user code, and a variety of hardware accelerators—is to provide a flexible fabric onto which the tasks can be dynamically mapped, not to provide a fixed site with unshared resources for each task. In theory such an architecture can be more efficient in its use of hardware. And, again in theory, it can have significant advantages for power management, as well.

On paper, either architecture can fully exploit the most advanced ideas in power management: voltage islands, dynamic power-down, and dynamic voltage-frequency scaling. But in practice, ARM’s Bruce observed, real SoC designers can go only so far with these techniques. Dynamic voltage-frequency scaling using ARM’s Intelligent Energy Management architecture is a formidable tool, but it is also formidable difficult.

“There’s always a silicon penalty for voltage islands. You have to weigh how far to go,” Bruce said. It would be impractical for most design teams to attempt it on more than the central ARM-11 processor in our hypothetical iPhone SoC. That leaves less effective and more response-impacting techniques, such as clock gating and power gating, for the other functional blocks in the chip. And it creates real issues for memory structures outside the CPU core, since at the geometries we are assuming here, RAM arrays leak a lot of current and to need very close attention for power management.

The multi-use core approach, on the other hand, puts the vast majority of the activity in two CPU cores that can be power-managed to whatever extent the implementers deem necessary. The system can dynamically control CPU core voltages and frequencies based on instantaneous task loads, or even based on pending real-time deadlines. And it centralizes the critical caches and scratchpads where they can be managed by one energy-management and error-correction block. Mobilygen doesn’t appear to be going quite this far, but their power figures -- such as 500 mW for a 1080i high-definition encode operation -- suggest they are quite a ways along this path.

Is it fair to say that we are seeing a crossroads in SoC design, with functional autonomous blocks gradually being abandoned in favor of a task-based approach, in which dynamic tasks are allocated across a more general-purpose computing fabric? It’s probably too early to tell. Dynamic fabrics and even multicore SoCs may turn out to be a dead end, undone by the difficulty of dynamic system performance modeling. Or it may turn out that the functional-block approach has overwhelming advantages in some applications. But looking at these two chips, one can’t resist the urge to speculate.