

Trends in Embedded-Microprocessor Design

Makers of embedded 32-bit processors have narrowed the gap between embedded and desktop systems, as new applications have fostered new classes of processors. How will this trend influence future embedded-processor design?

Copyright character 1998 IEEE. Reprinted, with permission, from IEEE Computer, Volum 31, NR 8, page 44-49, August 1998.

When discussing microprocessors, we tend to think of the Intel x86 architecture and its competitors—the IBM/Motorola PowerPC, Digital Alpha, Sun UltraSparc, Hewlett-Packard PA-RISC, or MIPS Technologies MIPS architectures. Designed primarily for the desktop market, these processors have dominated the scene—with the x86 being the clear winner—and architects are striving to deliver even more computational power to the desktop. [1] In concentrating on the desktop, however, we may be missing the next big thing in microprocessor design: embedded CPUs. As David Patterson has argued, Intel specializes in designing microprocessors for the desktop PC, which in five years may no longer be the most important type of computer. Its successor may be a personal mobile computer that integrates the portable computer with a cellular phone, digital camera, and video game player... Such devices require low-cost, energy-efficient microprocessors, and Intel is far from a leader in that area. [2]

So what are these embedded CPUs and how do they differ from general-purpose processors?

THE EMBEDDED MARKET PLACE

We tend to distinguish between microcontrollers and

microprocessors, even though there's no hard and fast definition. Viewed very simply, you often find microcontrollers associated with the embedded domain and microprocessors with the desktop arena. A more refined approach is to define microcontrollers as having RAM and ROM instead of caches, accompanied by a lot of peripherals. In contrast, we think of microprocessors as having a memory management unit and lots of cache. Sometimes, it is simply performance that determines whether we classify a device as a microcontroller or not; thus, 8- and 16-bit devices are normally called microcontrollers.

More recently, the market for 32-bit embedded processors has been growing. This trend may seem strange to developers of brake control systems or washing machines—and, indeed, forecasters predict that eight times more 8-bit than 32-bit embedded processors will ship in 1999. [3] But as demand for security and central control stations rises—or as refrigerators begin to include artificial intelligence—this trend becomes more understandable.

32-bit embedded processors

In the 32-bit embedded-processor market, we find more than 100 vendors and two dozen instruction set architectures. [3] In 1996, the most successful embed-

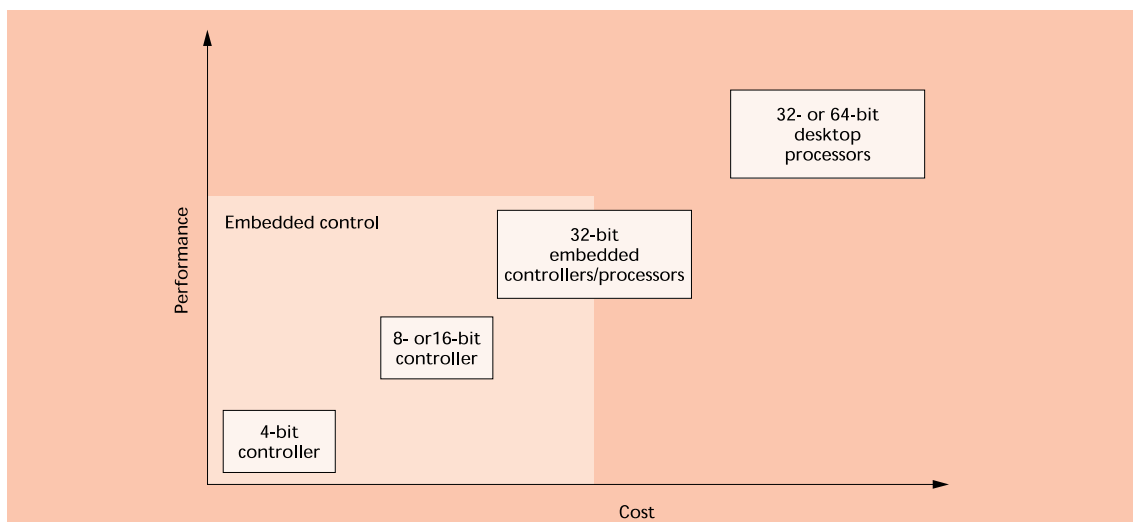


Figure 1. In the past, it was sufficient to partition the market into four basic, clearly separated, target markets. Emerging handheld, mobile, and multimedia applications require new classes of embedded processors.

Ad RT Encyclopaedia

ded microprocessor was Motorola's 68000 architecture; and although its further evolution came to a certain end, analysts believe that demand for 68000 derivatives will continue to grow over the next couple of years. [3]

Other architectures for the embedded market include Intel's i960, Motorola's Coldfire, Sun's Sparc, and embedded x86 platforms. Exciting new architectures are bursting onto the scene—Hitachi's SuperH, Advanced RISC Machines' ARM, and the aforementioned MIPS architecture. (ARM and MIPS have sold their architectures to licensees as cores, which are manufactured in different versions by various semiconductor companies.)

The growing success of these architectures stems mainly from the demand for video games, handheld computers, digital still cameras, and cellular phones. If you look inside today's video game consoles, you will find incredible processing power equivalent to a Silicon Graphics workstation of not too long ago.

Desktop versus embedded

A couple of years ago it was sufficient to partition the microprocessor world into the desktop and embedded markets. Figure 1 illustrates the traditional view in which it was sufficient to classify processors and controllers into a few straightforward classes.

Is this still sufficient? What is the difference between a desktop and an embedded processor? Some embedded platforms arose from architectures designed primarily for the desktop market (for example, MIPS, Sparc, and the x86). Thus, the difference cannot be the register organization, basic instruction set, or the pipelining concept.

Instead, such issues as power consumption, cost, and integrated peripherals differentiate a desktop CPU from an embedded processor. Other important features include the interrupt response time, the amount of on-chip RAM or ROM, and the number of parallel ports. Whereas the desktop world values processing power, an embedded microprocessor must do the job for a particular application at the lowest possible cost.

Economically meeting the needs of new applications is the driving force behind new classes of embedded processors, devices that narrow the gap between the desktop and the embedded world. Applications like handheld, palmtop, or network PCs, video game consoles, and car information systems require a display, a powerful processor, storage media, and interfaces to communicate with the outside world. For these reasons, embedded processors have in some ways begun to incorporate capabilities traditionally associated with conventional CPUs—but with a twist: They are subject to challenging cost, power consumption, and application-imposed constraints.

NEW APPLICATIONS DRIVE REQUIREMENTS

New applications and market segments create the economic demand that permits the development of new or specialized devices. Cost and time-to-market considerations, however, make it mandatory to reuse

these devices in other applications, avoiding the steep learning curves that accompany product development. For these reasons, most company's design embedded processors for a very specific market and later extend those designs to the general embedded market.

The current applications driving product development include

- Video game consoles, which require excellent graphics performance;
- Handheld, palmtop, automobile, and network PCs, which require virtual memory management and standard peripherals;
- Cellular phones and mobile personal communicators, which require ultralow power consumption while offering high performance and digital signal processing (DSP) capabilities;
- Modems, fax machines, and printers, which require low-cost components;
- Set-top boxes and DVD (digital versatile disk) equipment, which require a high level of integration; and
- Digital cameras, which require both general-purpose and image-processing capabilities.

Manufacturers sell most current 32-bit embedded processors to the consumer application market, followed by the communication and office automation markets. The requirements of these markets force embedded microprocessor designers to reduce manufacturing costs while simultaneously increasing the level of integration and performance.

Hence, the following criteria are essential when comparing embedded processors:

- Power consumption. For mobile applications, the benchmark is the MIPS/watt ratio.
- Code density. The goal here is to avoid the complexity of CISC and density of 32-bit, fixed-length RISC architectures.
- Peripheral integration and chipsets. Serial communication interfaces are mandatory for embedded processors, for example, and for reducing overall cost for dedicated applications.
- Multimedia acceleration and acceleration of special application software. This is done via enhanced instruction set functionality.
- Price/performance ratio. After all, cost—measured in MIPS/dollar—is the main issue.

These are the evaluation parameters for embedded processors that define the milestones against which microprocessor development must be measured.

Power consumption

In the embedded domain, applications cannot use a heat sink or a fan: A cellular phone with a fan would probably not be a top seller. Thus, most embedded microprocessors have three different modes: fully operational; standby or power down; and clock-off. (Some

TRENDS-TO SEE FORWARD, LOOK BACK

Every period in the history of microprocessors—embedded and desktop—has a certain focus. To estimate future trends, it is always useful to look to previous periods. As most processors in the embedded domain are related to a desktop architecture, the evolution of embedded processors cannot be viewed independently.

The 1980s.

The basics of today's most popular architectures were developed in the 1980s when RISC was introduced. Architectures have been extended, but the basics are the same. At that time, RISC really meant reduced and CISC meant complex.

Early 1990s.

During this period, we saw many papers on superscalar, cache strategies, branch prediction, and the RISC versus CISC debate had its hottest moments. But something else happened: ASIC design and the licensing of core technology became popular. Power consumption, code density, and integration strategies became important.

Mid 1990s.

More 32-bit embedded processors derived from basic cores came to market, but they focused on special markets. Multimedia and DSP became major points of discussion, supplanting the CISC/RISC debate. We witnessed the first multimedia extensions and a merging of RISC and DSP architectures, fueled by a niche market for high-end embedded multimedia processors.

Late 1990s.

Today we see a revival of the SIMD (single-instruction, multiple-data) idea for multimedia acceleration. Java and Windows CE are driving new markets. Previous ASIC discussions and advances in process technology have led to discussions of superintegration: the system-on-a-chip. The rise of portable consumer applications has made power consumption a critical issue. Vendors introduced chipsets for dedicated consumer applications.

Today, we see several hot topics in the embedded world, the most notable being superintegration, software, multimedia capabilities, and power consumption.

vendors use different names, but they mean basically the same thing.) Fully operational means that the clock signal is propagated to the entire processor and all functional units are available to execute instructions. In standby mode, the processor is not actually executing an instruction, but all its stored information is still available—for example, DRAM is still refreshed, register contents are valid, and so forth. In the event of an external interrupt, the processor returns (in a couple of cycles) to fully operational mode without losing information. In clock-off mode, the system has to be restarted, which takes nearly as long as the initial start-up.

Reducing power consumption.

Most new processors focus on reducing power consumption in fully operational and standby modes. They do so by stopping transistor activity when a particular block is not in use. For that reason, such designs connect every register, flip-flop, or latch to the processor's clock tree. The implementation of the clock therefore becomes crucial, and it often must be completely redesigned. (In traditional microprocessor design, the clock signal is propagated from a single point throughout the chip.)

The simplest way to reduce power consumption is to reduce the voltage level. A CPU core voltage of 1.8V or even less is the state-of-the-art process technology. Core voltage, however, no longer tells the entire story as it did in the past. The increasing integration of power-consuming peripherals alongside embedded cores forces us to measure the power consumption of the entire system.

Overall power consumption differs a lot, depending on your system design. Core voltage means less as a performance metric, because the activities inside a core

are normally related to the activities of the peripherals. Increasingly, the processor core is only a small part of the entire system. Power consumption will likely be the most important challenge in future system design.

Performance.

Power consumption is never a stand-alone benchmark: It has to be related to performance. The announced target of all vendors in the 32-bit embedded arena is currently 1,000 MIPS/watt. But the MIPS/watt ratio alone is an insufficient tool for comparing embedded processors. Consider that even NOP instructions—during which the processor does no useful work—can theoretically be the basis for a MIPS/watt ratio.

To alleviate the inadequacy of MIPS/watt ratio alone, benchmark programs like Dhrystone have been developed that include more than just the number of instructions executed.

Another limiting factor in the embedded world is data throughput or bandwidth. As more embedded systems include massive data processing, such as image coding, there will be increased emphasis on cache design and data bus architecture—as is the case in the desktop arena. The difference will be the additional emphasis on low cost and real-time capabilities. As with power consumption, increasing overall system performance will be a major challenge for future embedded-processor designs.

Code density

CISC architectures traditionally have better code density as a result of their more complex instructions. The RISC design philosophy introduced—as a basic requirement—fixed-length instructions, to simplify and acceler-

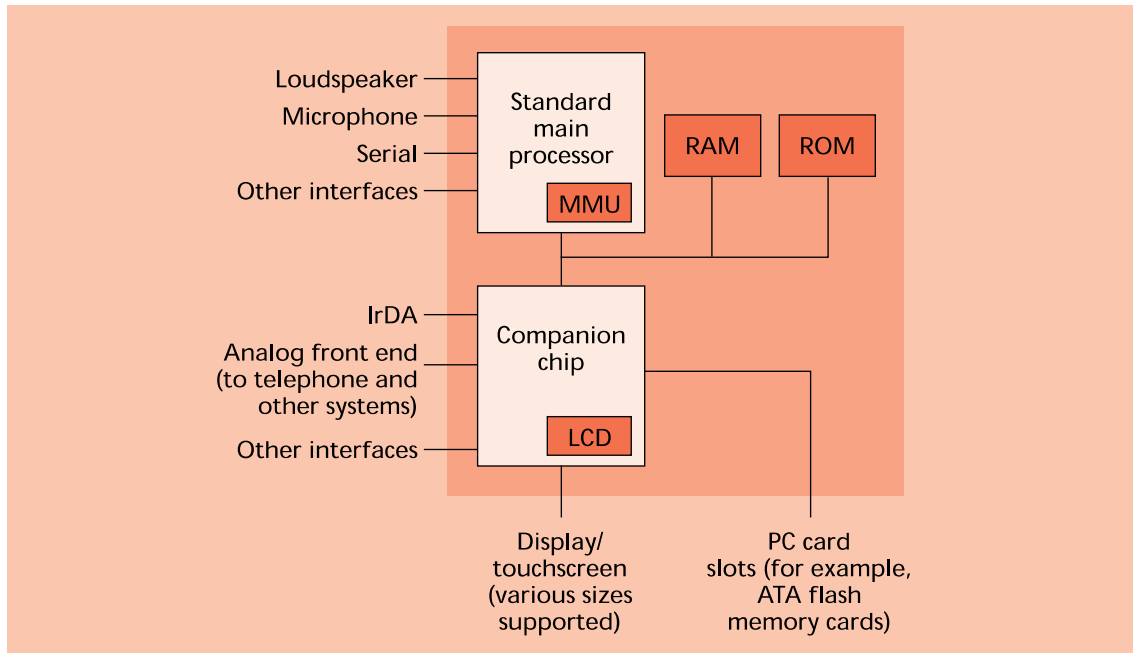


Figure 2. Two-chip approach for devices like handheld PCs. We can offer increased flexibility by providing a family of companion chips and by changing the standard main processor. Such companion chips—for example, Hitachi's HD64461 LCD controller with additional interfaces for special markets—offer a road to further standardization in the embedded market.

ate the instruction decoding. Although decoding faster, RISC architectures required more instructions to do the same work. The RISC design philosophy is closely related to the introduction of pipelining. Pipelined architectures forced processors to have a more unified decoding process for all instructions, which resulted in the fixed 32-bit instruction of, for example, the MIPS, Sparc, or PowerPC RISC architectures. The fixed 32-bit instruction length negatively affected code density of programs for 32-bit RISC processors.

To overcome the problem, vendors have introduced new strategies. Hitachi, for example, is using a fixed 16-bit instruction for their SuperH architecture, which reserves only 16 bits (rather than 32) for each instruction. This approach simply localizes the program code except for handling the address and immediate values. This also increases memory bandwidth for instruction load.

ARM used another strategy when it introduced the Thumb extension, a subset of 32-bit ARM instructions recoded into 16-bit opcodes. On-chip logic decompresses these opcodes into their 32-bit equivalents in real time.[4] Another example is the MIPS 16 approach.

A different approach is to use a variable instruction length of, say, 16, 32, and 48 bits. [5] Immediate values are not implemented by referring to tables; they are encoded directly into a 48-bit instruction. This approach improves the decoding process over that of CISC architectures. It is, however, still more complicated than fixed-length approaches because the state machine has to decide very early what instructions to decode and what data to transfer to the next stage.

Another important factor that affects code density is the quality of C compilers. ANSI C is the current de facto standard in the embedded world. As embedded-processor performance increases, object-oriented lan-

guages will play an increasing role in the future. Fortunately, compiler vendors have already developed sophisticated solutions for reducing code density, and a further reduction will mainly provide a performance gain.

Peripherals and higher integration

When designing a processor, you must also consider which peripherals to integrate for your target market. Integrating everything onto the same die is not always the cheapest solution. Peripherals increase chip complexity and tend to reduce yield; additionally, peripherals mean more pins and more testing. Yield and pin-count are essential measures for the final price and, thus, for potential market success. Thus, integrated peripherals must simplify system design and shorten the development cycle of complete systems.

Another demand arises from the fast-changing DRAM business. Users often demand integration of DRAM onto the die to avoid this volatile market. They can do so because of our increasing ability to integrate more transistors on a die, thus avoiding the electromagnetic interference problems that occur with off-chip memory. Finally, embedded DRAM, or eDRAM, offers further power savings because the extremely small load capacitance of the bus wiring between the CPU and DRAM dissipates less power. [6]

There are two strategies for integrating peripheral logic such as dedicated interface controllers.

- You can provide the basic core and integrate additional logic for a custom device to create an ASIC or ASSP (application-specific standard part).
- You can offer a standard microprocessor together with a companion chip that serves application-specific needs. This chipset approach parallels a well-

known practice in the PC board business.

The difference is not really a technical one. If you develop a macrocell for a companion chip, it is quite simple to integrate it into an ASIC. It's more a question of marketing, and in some sense represents an effort to standardize the embedded market in the same manner as the PC market. Figure 2 contains a simplified block diagram of a handheld PC that illustrates the chipset approach to further standardization.

Multimedia acceleration

In the embedded domain, conventional CPU and DSP capabilities are merging in a way similar to what we have seen on the desktop. The so-called Media MIPS (MMIPS) capabilities are also dominating discussions in the embedded world today. To implement MMIPS capabilities, architects are enhancing the conventional microcontroller instruction set with instructions that

and discrete cosine transform instructions for JPEG and MPEG image compression. Filtering in particular is a traditional DSP function. Functions that a separate DSP processor performed in the past—at a time when the microcontroller only ran the control protocol stack—are now more frequently executed by the embedded microprocessor itself.

For example, consider the implementation of a cellular phone. Such an application needs a lot of control functionality to handle the protocol stack and the interface to the external world, but it also does an increasing amount of equalization, voice encoding/decoding, and compression. Traditionally, we handled this with a low-performance microcontroller and a dedicated DSP processor, as shown in Figure 3a, perhaps integrated into a single ASIC or ASSP.

Handling it with a single microprocessor drastically simplifies the overall system design by having a single instruction stream as shown in Figure 3b. Examples of such an integration path include ARM's Piccolo or Hitachi's SH-DSP line. Examples of high-end embedded processors include NEC's V830R/AV, [7] which offers software decoding for MPEG-2 and Hitachi's SH4, [8] which offers graphics support.

Besides general-purpose architectures with multimedia extensions, new high-end multimedia processors can handle several MPEG streams while running modem code and so forth. Although the success of these new strategies to support multithreading is not guaranteed, they will certainly influence future processor design.

STANDARDIZATION VIA STANDARD OPERATING SYSTEMS: WINDOWS CE AND JAVA

Where the desktop is ruled by a few operating systems, the embedded arena is still a collection of several platforms supporting many different applications with varying requirements. But we seem to be witnessing a path to increased standardization and unification. The growing interest in handheld and palmtop PCs, personal communicators, Internet phones, and video game consoles has created a demand for a standard operating system that could unify the embedded processor market, just as it did the desktop market.

Microsoft has already reacted to that demand by developing its Windows Compact Edition (CE) operating system. Sun Microsystems used this demand to promote Java. With Java, developers can write code for a specific system independently of the underlying processor platform. Sun also introduced special processors to run Java more efficiently and thus further the unification process. Java's and Windows CE's current success in the embedded domain results largely from their excellent graphical user interfaces. Their further growth will depend on their ability to run conventional embedded-control programs with hard real-time requirements.

For a long time, managers have wanted embedded-system design to be more standardized and less platform-dependent. Yet the embedded marketplace has for years been a refuge for engineers looking for the

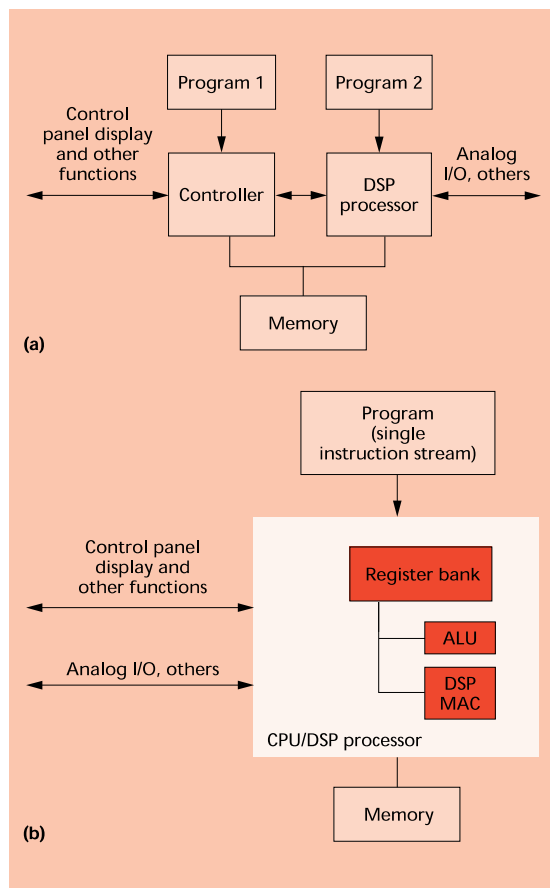


Figure 3. The (a) traditional approach to designing a system that requires control and DSP functionality requires two program instruction streams. A (b) simplified system approach uses next-generation CPU/DSP embedded processors.

accelerate the execution of multimedia code. In most cases, these instructions have a lot in common with the instruction set of a DSP processor.

For example, these instructions for multimedia support the multiply-accumulate operation for fast filtering enhanced addressing modes, graphics acceleration,

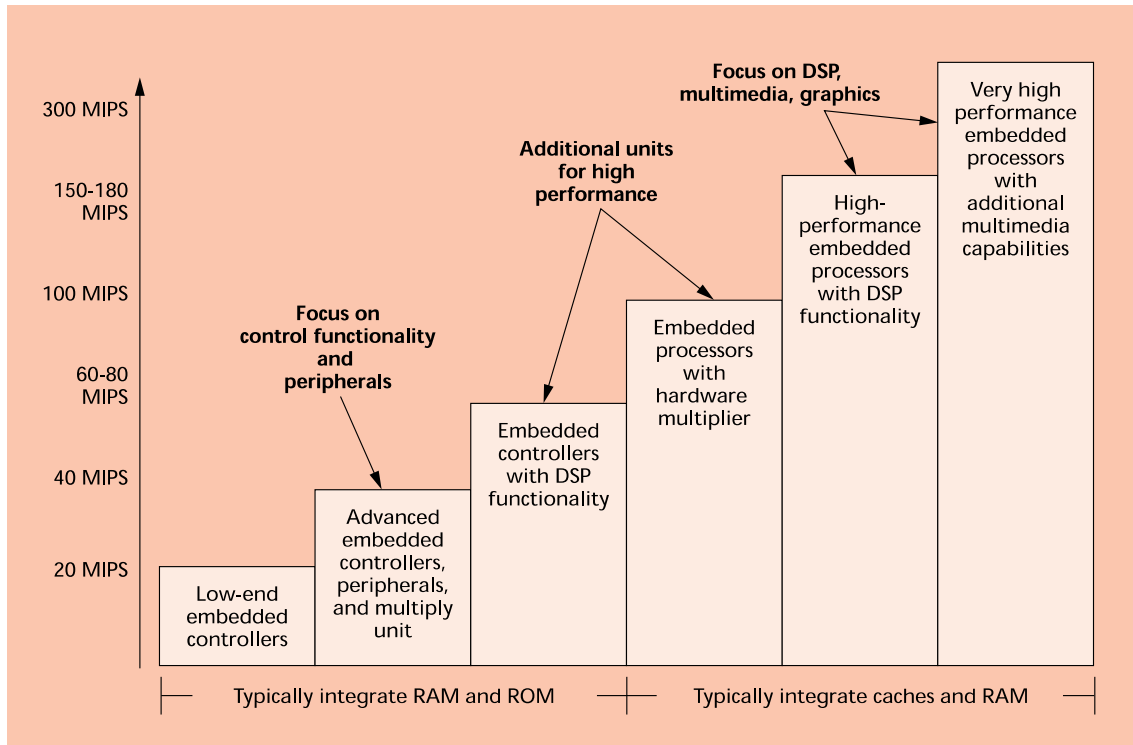


Figure 4. New view of the 32-bit market: From left to right, the traditional embedded area, traditional DSP, merged CPU/DSP field, extended architectures with multimedia extensions.

best technical solution. Standardization has yet to really affect design and platform choices—the best technical solution still drives design decisions. Up to now, embedded microprocessors have offered a price/performance ratio that easily outweighed the benefits of standardization.

But engineers should be aware that they cannot choose on their own anymore; management will increasingly influence platform decisions. By using standardized platforms, design houses can drastically reduce design cycle time and learning curves—and this is what matters to management.

The ideal embedded processor of the future will offer plenty of MIPS, run DSP programs like a dedicated DSP processor, integrate all its peripherals, cool the environment like a refrigerator, and cost but a few cents. Although this processor probably won't be available for many years, it describes the road ahead.

The booming embedded market has created several new classes of 32-bit embedded controllers and processors. A family of 32-bit processors typically starts with a low-end 10-MHz device and currently ends in the 300 MIPS area. In between, we see several new classes, each related to a certain market segment and application area. Figure 4 attempts to classify these new 32-bit embedded processors.

The big question is whether the embedded world will have a dominating architecture like the x86 in the desktop arena. Because the embedded world is driven by a variety of applications, this will probably not happen. But the trend toward using standard operating systems and platforms will arise from management's demand to reduce development costs, increase reusability, and improve design cycle time. A very real-

istic possibility is that each market segment will have its dominant architecture—and perhaps vendor. ■

REFERENCES

1. J. Wilson et al., "Challenges and Trends in Processor Design," *Computer*, Jan. 1998, pp. 39-50.
2. David Patterson, "Vulnerable Intel," *The New York Times*, June 9, 1998.
3. J. Turley, *Evaluating Embedded Processors*, MicroDesign Resources, Sebastopol, Calif., 1997.
4. S. Segars, K. Clarke, and L. Gouge, "Embedded Control Problems, Thumb, and the ARM7TDMI," *IEEE Micro*, Oct. 1995, pp. 22-30.
5. M. Dolle and M. Schlett, "A Cost-Effective RISC/DSP Microprocessor for Embedded Systems," *IEEE Micro*, Oct. 95, pp. 32-40.
6. Y. Nanomura et al., "M32R/D-Integrating DRAM and Microprocessor," *IEEE Micro*, Nov. 1997, pp. 40-48.
7. F. Arakawa et al., "SH4 RISC Multimedia Processor," *IEEE Micro*, Vol. 18, No. 2, pp. 26-34.
8. K. Suzuki et al., "V830R/AV," *IEEE Micro*, Vol. 18, No. 2, pp. 36-47.

Manfred Schlett is a marketing engineer at Hitachi Europe GmbH. His research interests include micro-processor architectures, advanced VLSI design, signal processing, and multimedia. Previously, he worked for Hyperstone Electronics as a design engineer. Schlett received his PhD in mathematics from the University of Karlsruhe, Germany. He is a member of the IEEE Computer Society.