DSPs: Back to the Future

To understand where DSPs are headed, we must look at where they've come from.

From the dawn of the DSP (digital signal processor), an old quote still echoes: "Oh, no! We'll have to use state-of-the-art $5\mu\text{m}$ NMOS!" The speaker's name is lost in the fog of history, as are many things from the ancient days of $5\mu\text{m}$ chip design. This quote refers to the first Bell Labs DSP whose mask set in fact underwent a 10 percent linear lithographic shrink to $4.5\mu\text{m}$ NMOS (N-channel metal oxide semiconductor) channel length and taped out in late 1979 with an aggressive full-custom circuit design. The designer I quoted had realized that the best technology of the time would be required to meet the performance demands of the then cutting-edge digital Touch-Tone receiver.

W.PATRICK HAYS, ULTRADATA CORPORATION

DSPs: Back to the Future

A parallel project at Intel would result in the Intel 2920 announced 25 years ago at ISSCC79 (International Solid-State Circuits Conference 1979).¹ The Intel 2920 included on-chip (D/A) digital/analog and A/D (analog/digital) converters but lacked a hardware multiplier and soon faded from the market. An NEC project resulted in the NEC μ PD7720—one of the most successful DSPs of all time. The Bell Labs DSP-1 and NEC μ PD7720 were announced at ISSCC80.² DSP-1 achieved 5-MHz clock speed, executing 1.25-MB multiply-accumulates per second at four clock cycles each—enough to allow Touch-Tone receiver filters to execute in realtime.

The once formidable performance demands of the Touch-Tone receiver are now ludicrously easy, but new applications in turn arose throughout the last 20 years to put new demands on DSP technology (see figure 1). According to Will Strauss, president and principal analyst at Forward Concepts, “DSP shipments were up a healthy 24 percent in 2003, and we are forecasting a bit higher growth for 2004, at 25 percent. Longer term, we forecast a 22.6 percent compound growth rate through 2007.”³ So the game has been: Boost DSP performance, run the algorithm at an acceptable cost, and open up a new commercial market. It is perhaps too glib to project this trend indefinitely into the future. In fact, savvy analysts have periodically predicted the demise of the DSP.⁴

Will performance requirements outstrip the ability of programmable DSP architectures to keep up, thus demanding a new approach? Or if DSPs are

to maintain their historical growth curve, what kinds of tools and architectures are needed? Ultimately, these questions will be answered by creative architects, market competition, and application demands. The goal of this article is to illuminate current and future trends by reviewing how technology and application pressures have shaped DSP architecture in the past.

WHAT IS A DSP?

At the outset, it is important to distinguish between *digital signal processing* and *digital signal processors*. The techniques and applications of digital signal processing, as compared with *analog signal processing*, are well established and are more important commercially than ever. Throughout this article, DSP refers to the VLSI (very large-scale integration) *processor* component. Therefore, what special demands in digital signal processing make a DSP different from another programmable processor? In other words, what makes a DSP a DSP?

DSP Applications vs. Data Rate and Complexity

Source: *Buyer's Guide to DSP Processors*, 2001 Edition, BDTI. Fig. 3.0-1, p. 13.

The Realtime Requirement. The essential application characteristic driving DSP architecture is the requirement to process realtime signals. *Realtime* means that the signal represents physical or “real” events. DSPs are designed to process realtime signals and must therefore be able to process the samples at the rate they are generated and arrive. Adding significant delay, or *latency*, to the output can be objectionable.

While high realtime rates often demand that DSPs be “fast,” fast and realtime are different concepts. For example, simulations of VLSI designs must be fast—the faster the better—but the application doesn’t fail if the simulator completes a little slower. Conversely, a realtime application need not be fast—for example, a hospital room heart monitor doesn’t need to be fast (30-Hz sample rate) but does need to be realtime; it would be disastrous if the processing of a sample took so long that after a few hours, the monitor was displaying five-minute-old data.

Not all digital signal processing applications require realtime processing. Many applications are performed offline. For instance, encoding high-fidelity audio for mastering CD-ROMs uses sophisticated digital signal processing algorithms, but the work isn’t done in realtime. Consequently, a DSP isn’t required—any old processor fast enough for the engineer to get home for dinner will do. To summarize, the most important distinguishing characteristic of DSPs is that they process realtime signals—the signals can be fast or slow, but they must be realtime.

Programmability. Do DSPs need to be programmable? No: it’s quite feasible to process digital signals without a programmable architecture. In this article, however, *DSP* refers to *programmable DSP*—more specifically, to *user-programmable DSPs*, because my bias is that that’s where the most interesting architectural issues lie. Often, the most demanding applications have required nonprogrammable architectures. For instance, first-generation programmable DSPs could execute a single channel of the 32-Kbps ADPCM/DLQ (adaptive differential pulse code modulation/dynamic locking quantizer) codec, whereas a special custom-integrated circuit that was not programmable but deeply pipelined could run eight channels in the same technology.

The reason for this is that programmability comes at a cost: Every single operation in a programmable chip—no matter how simple—requires *fetch-decode-execute*. That’s a lot of silicon area and power devoted to, say, shifting left by two bits. Nonprogrammable architectures succeed when the shift-left-by-two-bits function is a small building block, allowing other building blocks to operate

simultaneously. It’s easy to imagine many building blocks working simultaneously to achieve a 10x performance advantage in nonprogrammable logic. The problem with specialized DSP hardware is that you have to develop a new chip for each application. As development costs increase, the break-even point is constantly shifting in favor of using a programmable architecture.

More Power. Higher clock speed permits more instructions to be executed during a fixed time interval. In 1980, the Bell Labs team struggled to run DSP-1 at 5 MHz; today in 130-nm technology, clock speeds greater than 500 MHz can be attained. The advantage of more instructions in a fixed time period can be used to achieve one or more of the following:

1. At a fixed data rate, more complicated algorithms can be programmed.
2. At a fixed data rate, more channels of the same algorithm can be programmed.
3. At a higher data rate, algorithms of similar complexity can be programmed.

An example of the first case is G.729A, a CELP (coded-excited linear predictor) speech codec. It allows good quality at low data rates. The algorithm requires about 30 times more computations per sample than G.711 PCM.

Examples of number 2 are VoIP (voice over IP) applications where four channels are supported for SoHo (small office/home office) products, and up to 256 or more channels for CO (central office) products. Channel density is the key metric for VoIP processing.

An example of the third case is the MPEG-2 video compression algorithm applied to decode DVDs at different picture resolutions. The computational power is directly proportional to the video resolution. Stretching MPEG-2 from NTSC (National Television System Committee) resolution to high definition requires not only a six-fold increase in processing power but new blue laser DVD technology for faster readout of the data from the disk.

In addition, advancing VLSI permits the programmable architecture to reduce power and/or cost for a fixed algorithm at a fixed data rate. Advancing technology conspires in many ways to move the boundary in favor of programmable DSPs. Applications that require highly specialized design today become programs for inexpensive DSPs tomorrow; costly power-hungry DSPs today become the jelly beans of tomorrow. The past 25 years has seen the ascendancy of the user-programmable DSP as the dominant architectural approach to implementing digital signal processing applications.

DSP architecture is driven by a number of specialized application characteristics. Let’s look at a few of these

DSPs: Back to the Future

before returning to the architectural influence of the all-important realtime constraint.

KEY REQUIREMENTS OF DSP APPLICATIONS

The basic operation in most DSP algorithms is multiply-accumulate:

$$\text{accumulator} \leftarrow \text{accumulator} + X * Y$$

Consider the FIR (finite impulse response) filter in figure 2. The FIR is one of the most important algorithms for processing digital samples either to remove or to enhance parts of the signal. The filter outputs $y(k)$ are represented as the accumulation of samples $x(i)$ multiplied by the filter coefficients $c(j)$. The more coefficients (or taps) there are, then the more accuracy there is in the filter—hence the desire to have as many samples and coefficients as feasible. Many important algorithms can be implemented using multiply-accumulate—for example, matrix multiplication, required by transforms used in video compression. Most filters have more coefficients than can fit into a general-purpose register file. As a result, x and y operands

and the coefficients are memory based.⁵

The FIR inner loop computes a single new $y(k)$. When the inner loop is programmed in MIPS-I, a typical RISC assembly language, nine instructions are required. Figure 3 shows the filter as programmed in TI TMS320C54xx assembly. The DSP resorts to memory-based operands with post-modified register indirect addressing (*AR2+, *AR3+), zero-overhead loop counters (RPT) that test and increment simultaneously, and implicit hardware support for circular buffers—all CISC (complex instruction set computer) techniques. As a result, the TMS320C54xx inner loop (typical of other 16-bit DSPs) can be squeezed into a single 16-bit instruction. At a comparable clock speed, the performance of the RISC code is almost 10 times slower. For early DSPs this penalty certainly meant the difference between whether most applications could be executed or not.

The micro-architecture to support a single-cycle execution rate for the DSP instruction requires an instruction bus, an X data bus, and a Y data bus. This architecture was often called a “modified Harvard architecture.”⁶ As

a result of the trade-offs required to achieve sufficient performance in a low-cost IC, DSPs became a “poor relations” branch in processor taxonomy. This lineage continued for generations.

ARCHITECTURE AND REALTIME CONSTRAINT
The essential requirement of realtime processing indeed constrains the DSP architecture in basic ways. The DSP program must sustain processing at the realtime rate under all circumstances, and, in fact, the programmer must

somehow *know* that this has been accomplished so that the application doesn't fail in the field. In other words, the DSP program must *deterministically allocate realtime*. Sources of indeterminacy, common in desktop CPUs, can be catastrophic for the DSP programmer. For example, page faults and cache misses can cause hundreds of cycles to be missed by the CPU as it is idled while the operation is satisfied. If you must sample a value every microsecond, then the page fault or cache miss could cause the window to be missed. As a result, DSPs need either fixed memories or caches that can be locked after the program is booted. Other less critical examples of indeterminacy include branch prediction and data-dependent termination of functions such as "divide." Although nice for the average case, the DSP program must also allow for the worst case.

Deterministic allocation of realtime not only must be achieved, but traditionally DSPs have made it *straightforward* to achieve. In newer DSPs, realtime allocation is indeed knowable at compile time, but very careful profiling and iterative programming are often required to achieve the desired outcome.

ILLUSTRATION:
THE TI TMS320C54XX
To bring the discussion down to earth, let's illustrate with a real DSP. Targeted at the cellphone, TI's TMS320C54xx was introduced in 1994; in a sense, it is the fruition of TI's 16-bit DSP product line, which started with the introduction of the TMS32010 in 1983 and moved through the 'C1x, 'C2x, 'C2xx, and 'C5x generations to the 'C54xx. Although strict compatibility wasn't maintained, the follow-on architectures were close enough for TI to migrate its growing customer base with each

new product generation much as Intel has done with the x86 family.

Earlier TI DSPs sacrificed much performance to improve ease of use. Numerous other shortcomings such as the lack of accumulator guard bits were also rectified over the years. Table 1 shows how the TMS320C54xx has addressed each of the DSP features discussed here. For later comparison the TMS320C62xx is also listed.

THE 16-BIT DSP RUNS OUT OF GAS

Another form of "accumulation" other than the multiply-accumulate arithmetic operation was taking place in the TI DSP product line: by the mid-1990s, the TI architecture had grown to more than 130 instructions. New specialized instructions are one way of improv-

FIR Filter Programmed in TI TMS320C54xx Assembly

```
RPT    #127                      /* Causes 128 iterations of MAC instruction */
MAC    *AR2+, *AR3+, A           /* A ← A + MEM[AR2] * MEM[AR3];
                                AR2 ← AR2 + 1;
                                AR3 ← AR3 + 1 */
```

FIG 3

Source: *Buyer's Guide to DSP Processors*, 2001 Edition. BDTI. p. 439.

TABLE 1 DSP Implementation in the TI TMS320C54xx and TMS320C62xx Architectures

	TMS320C54xx	TMS320C62xx
Year of introduction	1994	1997
Architecture	16-bit DSP	VelociTI. VLIW RISC-DSP. (8) 32-bit instructions.
Pipeline stages	6	11
Instructions	130	80
Number of special registers	40	10
Clock speed in 0.15µm	160 MHz	300 MHz
MMACS in 0.15µm	160	600
Multiply-accumulate inst	Yes	No
Post-modified pointers	Yes	Yes
Zero-overhead loops	Yes	No
Circular buffers	Yes	Yes

DSPs: Back to the Future

ing performance—the way early DSPs used to meet cost goals. It became difficult to pack new instructions into the TMS320C54xx's burdened instruction set. Clock speed can be increased over time but does not take full advantage of advancing technology if the CISC instruction growth continues. Somehow the DSP architecture needed to find a way to use the extra transistors of later-generation IC technology to increase performance. Deeper pipelining gives little benefit because the deeper pipeline must benefit all critical paths and CISC instructions have many complex critical paths. An alternative strategy, VLIW (very long instruction word) parallelism, boosts performance by executing multiple instructions in parallel. VLIW is relatively ineffective on CISC instruction sets because it's difficult to identify instructions that are commonly executed in parallel.

It is also important to note that compilers have had little success with complex 16-bit DSP instruction sets.⁷ Yet as higher clock speeds and larger local memories permit larger programs, the demand for good DSP compilers becomes paramount. Consequently, the 16-bit DSP is out of gas: It's too complicated to scale performance with Moore's law and too complicated to support good compilation. While the 'C54xx was running 160 MHz using full custom 0.15µm circuit design, the StrongARM RISC broke 600 MHz in 0.18µm.

Faced with this crisis, in 1997 TI introduced the all-new 32-bit VelociTI instruction set with its TMS320C62xx architecture. The TMS320C62xx has had enormous publicity as an eight-issue VLIW architecture (thus, the real instruction length is 8 x 32 or 256 bits, and it is possible to execute eight 32-bit instructions in parallel on the chip). Less remarked, but equally important, is that each instruction is a relatively simple 32-bit RISC-like instruction. In fact, it's ironic that RISCs have included multiply-accumulate instructions since the mid-1990s, but TI—the company that has shipped more multiply-accumulates than any vendor—chose to “out-RISC the RISCs” by requiring a multiply followed by an add instruction to implement the common DSP kernel.⁸ I call the new RISC-like DSP instruction sets, “RISC-DSP.”

RISC-DSP FILTER EXAMPLE

For an illustration of RISC-DSP, let's return to the FIR filter program. We saw that the instruction count of the inner loop is nine times better in the DSP case than for the conventional RISC. Keep in mind, though, that clock speeds today are 100 times that of the 1980s when DSPs started down the CISC path. As a result, the RISC can execute the FIR almost 10 times faster than a 1980s DSP but one-tenth the speed of an optimized DSP architecture with the same clock speed—assuming of course that the

RISC can provide lockable caches and other means to avoid indeterminate realtime behavior.

This 10x advantage is more than the RISC should give up. The performance of the RISC code can be improved with a series of extensions. Table 2 illustrates that conventional RISC performance and conventional DSP performance are simply two points on the spec-

TABLE 2 Filter Performance as a Function of Feature Set

	Inner Loop Instruction Count
MIPS-I (simple RISC baseline)	9
Incremental DSP feature	
Multiply-accumulate instruction (targeted)	7
Post-modified pointers	5
Circular buffer pointer	4
Zero-overhead loop instruction	3
VLIW (parallel execute, sample load, coefficient load)	1

trum of performance for the FIR filter. In 1980 the dial needed to be turned all the way to “DSP” in order to meet the minimal performance goals; today the architect can choose different points on the spectrum with less performance. At today’s clock speeds, RISC-DSP performance will be sufficient for many applications and have other advantages as well. Sources and destinations from a general-purpose register file are easily encoded in a 32-bit RISC-DSP instruction, making compilers more successful. Decoupling data loads from execution permits higher clock speed because data can be preloaded into a general-purpose register file. For each special feature, careful study of the potential number of instructions saved, critical path impact, interrupt overhead, and, of course, compilation is required.

To summarize, 32-bit RISC-DSP instruction sets have moved DSPs onto the historic RISC technology learning curve.

APPLICATIONS OF RISC + DSP

We’ve seen that the need for good tools and continued performance scaling have forced DSP architects to break with the complex 16-bit instruction sets of the past. RISC-DSP, however, really comes to fruition in applications combining both “RISC tasks” and “DSP tasks.” Applications of this type are proliferating commensurate with DSP applications on packet networks.

An important example is the 3G wireless handset of the near future, with video communications and speech recognition. Table 3 lists the key tasks, classifying them as either conventional RISC tasks or conventional DSP tasks. We see that a single RISC-DSP at about 200 MHz has sufficient performance for all tasks.⁹ The important advantage achievable in this application is that separate RISC and DSP chips—or separate RISC and DSP cores—aren’t required. Significant architectural efficiency is gained because data doesn’t need to be communicated between two different subsystems. This efficiency translates into hardware and performance advantages and therefore reduced cost and power. Because this is a handheld consumer device, opportunities to save power and cost are critically important.

The key barrier to merging DSP applications and RISC applications on a common processor is the need for deterministic response time. The RISC processor often supports an operating system complicating the realtime problem. Packetized networks, however, have relaxed the realtime constraint somewhat: The samples arrive in packets; therefore, the realtime response rate is the (somewhat irregular) packet arrival rate. For example, 80,000 packets per second are transmitted over 1 GbE (Gigabit Ethernet). MontaVista’s sponsored Linux preemptive kernel guarantees a worst-case kernel preemption latency under 1 millisecond on several CPUs. VxWorks guaran-

tees an interrupt response of a few microseconds on a 500-MHz Pentium. So DSP applications can often now be run under major realtime operating systems.

The Siemens Tricore deserves recognition as one of the first RISC-DSPs. “Tri” signifies that microprocessor, DSP, and microcontrol functions are combined in a common processor. Intel and Analog Devices have recently collaborated to build the Intel MSA (Micro Signal Architecture). The first product, the ADSP-21535 (Blackfin), appears to be targeted at the 3G cellphone. StarCore and the Philips Trimedia are two additional high-perfor-

TABLE 3 3G Wireless Handset:
DSP and RISC Processing Tasks

Processing Tasks	RISC-DSP Workload
CDMA EVRC speech codec	38 Mhz
Automatic gain control and MIC array	7 Mhz
Acoustic echo canceller (32 ms window)	10 Mhz
MP3 decode	32 Mhz
Audio mixer	2 Mhz
MPEG-4 QCIF decode (15 FPS)	16 Mhz
MPEG-4 QCIF encode (15 FPS)	62 Mhz
Speech recognizer (limited vocabulary)	n/a
Communications protocols (384 KBPS)	10 Mhz
QVGA rendering (15 FPS)	16 Mhz
I/O	2 Mhz
RTOS and java virtual machine	5 Mhz
Total	200 Mhz

Source: Hays, W. P., Hanna, C., and Probell, J. LX5380: RISC-DSP for new Internet applications. *Microprocessor Forum* (October 2001).

DSPs: Back to the Future

mance RISC-DSP architectures, each with VLIW implementations.

Coming from the RISC side, all vendors are taking digital signal processing requirements into account: ARM with "E" extensions, Hitachi with SH-DSP now into its third generation, IBM's PowerPC with Book E. MIPS has recently announced CoreExtend. Jonah Probell has demonstrated that DSP extensions with CoreExtend can achieve 3x speed-up on audio applications.¹⁰

INTO THE FUTURE

VLIW architectures applied to RISC-DSP instruction sets offer an important path for increasing performance, but the silicon cost of these architectures is not negligible. Although the eight functional units in the TI TMS320C62xx data path have capability well beyond the 'C54xx, when applied to a typical case like the direct-form FIR, the eight-issue 'C62xx architecture uses 256 instruction bits to accomplish about what the 'C54xx can do in 16 bits. The extra silicon cost also extends to data-path elements and to the 15 ports on the register file required to sustain the eight functional units. As a result, TI's VelociTI products are positioned for high-performance applications that, at the same time are not price- and power-sensitive. Another turn of the technology crank will be needed before VLIW architectures crowd out older 16-bit DSPs altogether.

The H.264 codec is an example of an application requiring VLIW DSP. UB Video has developed H.264 decoder software for the 600-MHz TI TMS320DM642. This device uses the 'C64xx core along with specialized audio and video interfacing. It is capable of 4,800 MMACs (million multiply-accumulates per second) in eight-bit precision. The UB Video software supports decode at SDTV (standard-definition TV) resolutions. It's important for H.264 decoder ICs to support the vast number of MPEG-2-encoded DVDs, as well as other codecs and future evolution in the ITU-T/ISO standard itself. The view of TI's Eric Braddom, worldwide manager for DSP video imaging, that "programmability is essential at this stage," is understandable from a technical as well as busi-

ness perspective.¹¹

Meanwhile, there is still room for architectural innovation within the framework of VLIW using RISC-DSP instruction sets. The promise of H.264 in the DVD market is its potential for high-definition (1080i, 720p) decoding. Currently, HD resolution in H.264 is beyond the 600-MHz 'DM642. TI has announced it will apply its forthcoming 1-GHz DSP. Other high-end VLIW DSPs such as Philips Trimedia with five-instruction-issue aren't waiting and are expected to attack the HD problem soon. Other competitors will reduce programmability, resorting to specialized hardware for MPEG-2 and H.264 alone.

To see how far DSP architectures are from "maturity," it's eye-opening to look at Appendix C, "Survey of Architectures," in Hennessey and Patterson's text.¹² The authors compare five RISC architectures. After a decade of research on compiler performance using well-established benchmarks, we find that RISCs are more alike than different. DSPs are a long way from "Appendix C status," but now that DSP applications are mainstream and both RISC vendors and DSP vendors are converging on RISC-DSP, the increasing cost of software development—along with the emergence of good benchmarks such as BDTI-mark2000 from Berkeley Design Technology (BDTI) and EEMBC (Embedded Microprocessor Benchmark Consortium) with which to measure design progress—will drive DSP architectures to become more similar, not unlike the path RISC followed a generation ago.

Whenever a programmable DSP architecture can meet an application's cost and power goals, it will be the preferred solution. But what about the applications that just can't fit? In the desktop market, application and system software seem to lag VLSI capabilities; in digital signal processing, the demands of "faster/cheaper/lower-power" have always pushed DSP VLSI. Nick Tredennick believes that the "leading-edge wedge ... of zero-cost, zero-power, and zero-delay segments of the embedded systems market," will drive DSPs to dynamic logic design.¹³ In fact, it's not an either/or decision: programmable DSPs can be extended with specialized hardware, whether classic fixed coprocessors or reconfigurable logic. The most visible

effort today to develop reconfigurable DSPs is Altera's Code:DSP program. ARM or Nios processors can be used to add programmability to Altera's DSP IP. FPGA (field-programmable gate array) vendors, Altera and Xilinx, provide DSP solutions with tremendous data parallelism that are capable of performing an entire set of FIR filter multiplications in single cycle. Because of the relatively high cost of Altera and Xilinx parts, they are practical for only the most demanding cost-insensitive applications. As silicon costs decrease, FPGAs might consume a larger share of the market.

DISAPPEARING DSPs?

Applications of digital signal processing are more prevalent than ever. Processing of natural data types has become one of the major roles of computation. These applications were opened by the 16-bit DSP architectures, which were highly specialized to attain performance requirements at low cost. In the early 1990s the 16-bit instruction sets hit the wall, and a break to new 32-bit RISC-DSP was required to continue scaling performance. This break was both necessitated and enabled by technology advances: The new 32-bit DSPs were able to deliver better performance, with better software development tools at a moderate increase in silicon cost.

The current trend is toward a convergence of embedded RISC and DSP architectures leading to a more standardized programmable architecture for digital signal processing. This trend is driven by the maturing of DSP architecture research and the cost of third-party software. Techniques like VLIW, supplemented where necessary by specialized hardware, will continue to extend the envelope for programmable DSPs well into the future.

To hazard an answer to the question I raised at the outset, DSPs won't disappear, but—as a result of their own success—will disappear as a separate and distinct branch of computer architecture. Q

ACKNOWLEDGMENTS

It's my pleasure to thank Jeff Bier and Berkeley Design Technology (BDTI) for loaning me a copy of *Buyer's Guide to DSP Processors* to assist with my background research. I also thank Jonah Probell of Ultra Data for his help with the article.

REFERENCES

1. ISSCC Digest of Technical Papers XXII, February 1979.
2. ISSCC Digest of Technical Papers XXIII, February 1980.
3. Strauss, W. Forward Concepts. Quote supplied for this article.

4. Tredennick, N. The death of the DSP. June 6, 2000; see: <http://www.ttvanguard.com/dublin/dspdeath.pdf>.
5. Input samples are memory based rather than from I/O registers because they are reused cyclically.
6. Howard Aiken, a WWII computer pioneer, classified processors according to the number of buses used. According to this classification, DSPs aren't "modified" Harvard architectures. They are, in fact, "Class III" Aiken machines.
7. How do you pack over 130 instructions into 16 bits? With numerous special registers.
8. The next-generation 'C64xx restored multiply-accumulates.
9. Speech recognition isn't included in the tally of the worst-case load because it's an offline function.
10. Probell, J. Improving application performance with instruction set extensions to embedded processors. DesignCon 2004; see: <http://www.ultradatacorp.com/publications.html>.
11. Yoshida, J. TI and UB Video get a jump on H.264 decoding. EE Times (December 2, 2002); <http://www.eetimes.com/semi/news/OEG20021202S0048>.
12. Hennessy, J. L., and Patterson, D. *Computer Architecture: A Quantitative Approach*, Appendix C. Morgan Kaufman, San Francisco: CA, 1996.
13. See reference 4.

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums

W. PATRICK HAYS (pat@ultradatacorp.com) is cofounder and vice president of VLSI Engineering at Ultra Data Corporation, a Waltham, Massachusetts, developer of licensable processor IP for high-definition video processing. Previously, Hays was cofounder and CTO of Lexra, where he led the definition of several high-performance CPU micro-architectures and created DSP and packet-processing extensions to RISC architectures. Prior to Lexra, Hays held director-level positions at TranSwitch and Polycom (PictureTel), where he led the development of new programmable architectures for realtime telecom and video applications. At Bell Laboratories, Hays was the principal architect of the DSP32xx, the world's first processor with on-chip floating-point arithmetic. He also managed the architecture team that developed the first fixed-point DSP16. He is coinventor of 11 U.S. patents and patent applications. He received an A.B. degree in physics from Harvard and a Ph.D. from MIT, also in physics.

© 2004 ACM 1542-7730/04/0300 \$5.00