Theme Feature

Challenges to Combining General-Purpose and Multimedia Processors

Multimedia-processor media extensions to general-purpose processors present new challenges to the compiler writer, language designer, and microarchitect.

Thomas M. Conte
North Carolina State University

Pradeep K. Dubey
IBM T.J. Watson Research Center

Matthew D. Jennings
North Carolina State University

Ruby B. Lee
Hewlett-Packard Laboratories

Alex Peleg
Intel Corp.

Salliah Rathnam
Philips TriMedia

Mike Schlansker
Hewlett-Packard Laboratories

Peter Song
MicroDesign Resources

Andrew Wolfe
S3 Inc.

Multimedia workloads have always held an important role in embedded applications, such as video cards or set-top boxes, but these workloads are becoming increasingly common in general-purpose computing as well. Over the past three years the major vendors of general-purpose processors (GPPs) have announced extensions to their instruction set architectures that supposedly enhance the performance of multimedia workloads. These include MAX-2 extensions to Hewlett-Packard PA-RISC,[1] MMX for Intel's *x*86,[2,3] UltraSparc's VIS,[4] and MDMX extensions to MIPS V.[5]

Processors targeted to embedded multimedia applications—the so-called multimedia processors (MMPs)—have employed similar semantics. These processors include Philip's TriMedia (TM-1),[6] Samsung's Multi-Media Signal Processor (MSP),[7] and Chromatic's Mpact.[8] In addition, some MMPs incorporate multimedia-specific semantics, including vector instructions (MSP and Mpact); a very-long-instruction-word architecture (TM-1 and Mpact); and hardwired special-purpose hardware such as video and audio ports and bitstream codecs.[9]

Most of the new instruction semantics for both GPPs and MMPs are based on a subword execution model. This model uses the entire width of a processor data path (32 or 64 bits), even when processing the small native data types found in signal processing (8- or 12-bit pixel, 8- or 16-bit audio). For example, if the word size of a machine is 64 bits, the adder can be used to implement eight 8-bit additions in parallel (by, for example, disconnecting the carry chain in the adder at every eighth position). The same trick can be used to operate in parallel on four 16-bit data types or on two 32-bit data types (for more detail see the sidebar, "Instruction Set Architecture Semantics for Media Processing.")

Merging these new multimedia instructions with existing GPPs poses several challenges. Also, some doubt remains as to whether multimedia extensions are a real development or just a competition-induced fad in the GPP industry. If it is indeed a development, how must current processor microarchitectures change in reaction? And if they change, can GPPs and MMPs apply application-specific integrated circuit (ASIC) solutions to the same problems?

CHALLENGE 1: DEALING WITH LAGGING COMPILERS

Multimedia extensions pack multiple operations into one word's worth of work. This is often referred to as *subword execution* (as explained in the sidebar). Limited compiler support for targeting subword instructions complicates the use of multimedia benchmarks, even when suitable benchmarks exist. Today, much digital signal processing code is handwritten in assembly to ensure the highest possible efficiency, because such code executes frequently.

Typically, compiler support for targeting ISA-extended GPP subword instructions uses function inlining or macro calls in code segments that will benefit from subword execution. This requires programmer awareness of subword grouping opportunities and programming at a lower level than is often desirable. The compiler should be able to determine subword groupings on the basis of data parallelism and data dependence analysis, but modern dependence analysis techniques are not suited to having multiple quantities in a single register. Ideally, if new techniques were used, subword semantics could be made invisible to the programmer. Today, the reality and the ideal are far apart.

CHALLENGE 2: DEALING WITH LAGGING LANGUAGES

Putting multimedia extensions into the same package as a powerful GPP is not a solution to handling multimedia workloads. It is not reasonable to users to either program in assembly to access the instructions or to use a high-level language as though it were assembly.

Although the new instructions provide considerable potential to achieve high performance, they are not well suited to a compiler. Subword operations typically sup-

0018-9162/97/$10.00 © 1997 IEEE

Instruction Set Architecture Semantics for Media Processing

Today's ISA extensions for media processing provide parallel execution of multimedia applications. To accomplish this, they use *packed data types* and *subword execution*. Figures A1 through A3 show subword execution of common multimedia operations. There are subword executions for arithmetic operations, data conversion, and rearrangement operations.

Subword versions of arithmetic operations—such as multiply-add, multiply-accumulate, and shift-add—include saturation options that clip results to the largest and smallest representable numbers. Subword versions of bitwise logical operations are also included when integer counterparts cannot be leveraged for subword execution, such as when integer registers and integer functional units are not used.

Data reorganization operations rearrange data that is not properly organized in memory in preparation for subword execution. Figures A4 through A8 show data rearrangement operations, a new feature of subword execution. Full-word rearrangement operations do not exist. Data rearrangement instructions are also used for conversion between packed data types, such as when increased precision is necessary for intermediate operations.

VIS extensions

The VIS extensions to UltraSparc are distinguished from the others by including several special-purpose

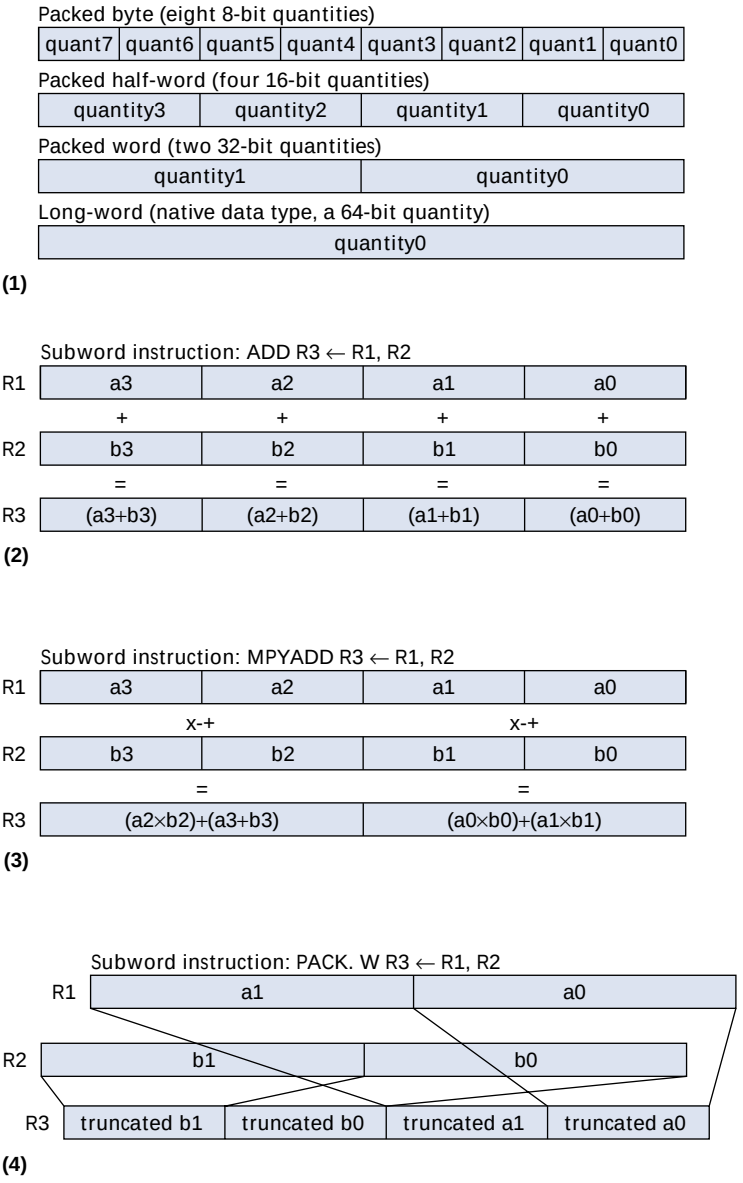


Figure A. Examples of subword execution. (1) Packed data types used to support subword execution; (2) subword arithmetic example using add instruction; (3) subword arithmetic example using multiply-add instruction; (4) data rearrangement example using pack instruction on word data; (5) data conversion example using unpack instruction on half-word data, low version; (6) data rearrangement example using mix instruction, high version; (7) data rearrangement example using permute instruction; (8) data rearrangement example using permute instruction on word.

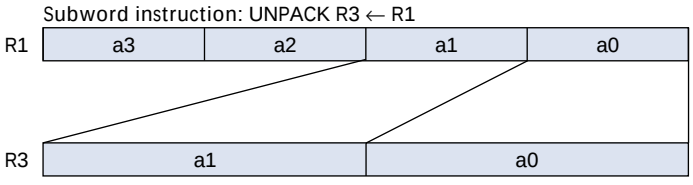
port saturating additions for applications such as video and audio processing (that is, underflow results in zero, overflow results in the maximum value).

Standard declarations of integers in high-level languages implicitly specify modulo addition. A minimum requirement of a programmer is strict specification of data types to guide a compiler in selecting packed data types of enough precision for variables of small native data sizes. For example, in C the common declarations `char`, `short`, `int`, and `long long` could be used to specify 8-, 16-, 32-, and 64-bit native data sizes, respectively. But such use is nonportable because one machine's interpretation of the size of `int` is different from another's. Since there are currently subtle differences between the GPP multimedia extensions, this may not matter. However, if a high-level language programmer has to make assumptions about the instruction semantics of the machine, then the compiler is not doing its job well.

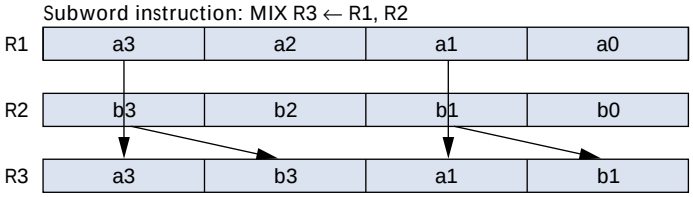
What is the long-term solution? Languages must allow programmers to specify data types and overflow semantics at variable declaration time so that multimedia applications can be coded correctly. In many ways, subword quantities are similar to short vector quantities in semantics, but the lack of new data types limits the application of automated vectorization methods already developed for scientific codes.

CHALLENGE 3: SORTING OUT WHAT'S REAL

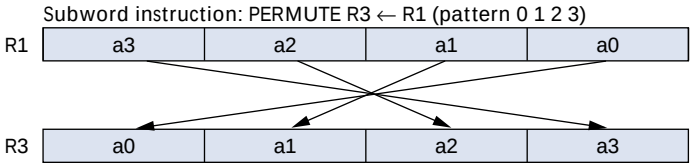
It is tempting to argue that multimedia workloads



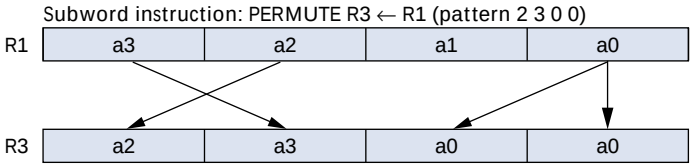
(5)



(6)



(7)



(8)

instructions with more complexity, including a pixel distance instruction, block load and store instructions, and a partial store instruction.

The pixel distance instruction computes the absolute difference between corresponding 8-bit components in a pair of packed byte registers and then accumulates these values. This instruction targets a predominant computation, the sum of accumulated differences, used in motion-estimation for MPEG video compression and in videoconferencing protocols.

Block load and store instructions transfer blocks of data between memory and registers without cache allocations. This allows the processor to move large amounts of data that are touched only once without displacing useful data from the caches. (Streaming multimedia algorithms, with data sets larger than the caches, read a block of data, perform a few operations, and write the results.)

The partial store instruction provides for masked store to memory with byte granularity. A byte mask is used with the partial store instruction; locations not selected by the mask are not modified by the instruction. In some instruction sets it is also possible to conditionally select which subword quantities are operated on.

MMP extensions

The MMP architectures (TM-1, MSP, and Mpact) include special-purpose features more typical of DSP architectures. These features include hardwired video and audio ports and bitstream codecs, multimedia operations (motion-estimation and (I-)DCT instructions, for example), and DSP instructions (saturation, minimum, and maximum).

In addition to subword execution, the MMP architectures use very long instruction words (as do TM-1 and Mpact) and vector instructions (as do MSP and Mpact) for concurrency.

are just a fad: Consider that applications for home systems now typically involve CD-ROM, video playback, audio, image processing, modem functions, and navigating the World Wide Web. Web content increasingly consists of video and audio clips, images, 3D graphics, and animation. High-performance video, such as digital videodisk playback (which consists of MPEG-2 and Dolby AC-3 audio), is now possible and is expected to command a significant portion of future workloads.

While office applications still consist of more traditional general-purpose applications such as word processors and spreadsheets, multimedia content is now found in presentations, and the Web is increasingly being used to retrieve information and advertise products. Mature desktop videoconferencing technology could revolutionize how business is conducted. Also, the creation of multimedia content is now a significant commercial endeavor in and of itself, whether for product marketing, CD-ROM titles, or Hollywood-scale productions. All this indicates that multimedia workloads are not a fad.

CHALLENGE 4: ADAPTING MICROARCHITECTURES

Although most GPP and MMP vendors have jumped on the bandwagon and included subword-style extensions in their processors, are they including the right semantics? If so, what is the implication for the microarchitecture?

To answer this, it is important to step back and look at the motivating workloads. Perceived workloads and

It is tempting to argue that multimedia workloads are just a fad.

benchmark performance are important in determining which processor features to include in systems. Benchmark performance is especially important to justify proposed architectural features. Perceived workloads, as opposed to actual workloads, are more important in defining features for consumer multimedia systems, as is evident from system features (such as full-motion video and sound boards) that have appeared before widely recognized media benchmarks were developed. Intel proposed a media benchmark set in conjunction with the release of their MMX technology, and UCLA has also developed MediaBench, a more-general benchmark set decoupled from any vendor's instruction semantics.[10] Both of these benchmark sets are beginning to fill the multimedia workload vacuum.

Using perceived benchmarks, it is possible to speculate about multimedia architectures. Current microprocessor features, such as only 32- and 64-bit data types (prior to subword extensions), large on-chip caches, complex interlocking hardware, and complex branch prediction hardware, have resulted from emphasis on performance for integer benchmarks (such as the SPECint92 and SPECint95 benchmarks).

Contrast those features with the following list of attributes that generally characterize multimedia applications:

- Small native data types (8 or 16 bits)
- Large data set sizes
- Large amounts of inherent data parallelism
- Computationally intensive features, but with highly predictable branches
- Real-time processing requirements
- Multiple concurrent media streams (such as video and audio)
- Large I/O bandwidth requirements

The first five are very different from the workloads used to design today's microprocessors. The remaining items suggest the need for new features or features not yet common to GPPs. Here are suggestions on what these features might look like:

- Unaligned accesses to register files at 8-bit granularity to reduce the need for PACK and UNPACK operations (see the sidebar for an explanation of these operations).
- Streaming prefetch units and compiler-driven on-chip memories to aid fetching large data sets.
- Compiler-assisted interlocking or other techniques that reduce the burden on the interlock hardware for certifying that a set of instructions is indeed parallel.
- Support for enhanced real-time predictability of execution, without also trading off high performance.

This is only a partial list. As workloads such as MediaBench are understood, this list is sure to grow.

In addition to microarchitectural features, the ISA extensions are continuing to evolve (for example, HP has released two generations of extensions, MAX-1 and MAX-2). Semantics for the MMPs include many special-purpose features, while semantics for the ISA extensions to GPPs are more general in scope (with the notable exception of VIS).

The new MMP architectures have had the freedom to include complex functionality because of fewer code compatibility concerns. Extending an ISA for a GPP requires more careful consideration and must focus on minimizing cycle-time impacts and maximizing implementation simplicity, as such extensions are more permanent. Combining GPP and MMP into one package will require a convergence on ISA semantics.

Improved benchmarking and experience with media processing will determine the utility of the more special-purpose features. The cost of implementation, in terms of present and future cycle-time and chip-real-estate impacts, needs to be determined for each possible feature. Utility and cost of implementation should both be considered before permanently adding a feature to an ISA.

CHALLENGE 5: USING PROGRAMMABLE MEDIA PROCESSING

The current solution to media processing is to design an ASIC for the application. Programmable media processing (via ISA extensions or embedded MMPs) is expected to supplant this special-purpose hardware for several reasons: First, it is less expensive. Programmable processors for media processing can replace several ASICs. To do so, the performance of the programmable alternative will need to be comparable to or exceed the combined performance of the application-specific hardware it replaces. Second, programmable media processing offers solution generality. As multimedia algorithms adapt, features of a multimedia system that uses a programmable solution may be updated with software.

A single-chip solution for combining GPP and MMP technologies follows the historical precedent of merging floating-point execution hardware with GPPs. In the case of floating-point execution, the paradigm shifted from special-purpose hardware to programmable floating-point coprocessors, and finally to floating-point functional units on GPPs. With media processing, it remains to be seen if GPPs will include multimedia execution units or if MMPs will incorporate improved general-purpose capability. Much depends on the amount of multimedia content in future workloads.

Target markets will also determine where emphasis should be. Consumer products, such as video phones, WebTVs, and "edutainment" devices will emphasize media processing over general-purpose fea-

tures. MMP architectures that include GPP features (TM-1 and MSP today) are suitable as stand-alone devices for these products. Multimedia computer systems will likely evolve from GPPs, unless shifts in workload to include multimedia content overwhelm traditional general-purpose applications.

MMPs are specialized to target multimedia applications, but some also include general-purpose computing functionality. Today's ISA-extended GPP and MMP technologies are complementary for multimedia computer systems. Currently, MMPs implement system features not found in GPPs, such as hardwired video and audio ports and bitstream encoders and decoders. MMPs also enhance performance by providing more media processing and I/O bandwidth capability. MMPs are now being used as coprocessors, providing media processing capability not yet possible on ISA-extended GPPs. In this way, MMPs today are being used for product differentiation, such as for high-performance graphics or digital video disk decode.

Multimedia workloads and GPP and MMP extensions are here to stay. However, compiler designers, language designers and microarchitects face large, difficult problems. Some of these problems will be solved by the industry as the pull of the new workloads causes investment and risk taking. Others, such as automatic extraction of subword parallelism by the compiler, likely will be developed in universities and transferred to industry. Universities need to do more of this. The ultimate solution to the problems presented here will be a collaborative effort between universities and industry—a familiar success story to processor designers. ❖

References

1. R.B. Lee, "Subword Parallelism with MAX-2," *IEEE Micro*, July/Aug. 1996, pp. 51–59.
2. A. Peleg and U. Weiser, "MMX Technology Extension to the Intel Architecture," *IEEE Micro*, July/Aug. 1996, pp. 42–50.
3. A. Peleg, S. Wilkie, and U. Weiser, "Intel MMX for Multimedia PCs," *Comm. ACM*, Jan. 1997, pp. 25–38.
4. M. Tremblay et al., "VIS Speeds New Media Processing," *IEEE Micro*, July/Aug. 1996, pp. 10–20.
5. "MIPS Digital Media Extension," *Instruction Set Architecture Specification*, http://www.mips.com/MDMXspec.ps (current Oct. 21, 1997).
6. G.A. Slavenburg, S. Rathnam, and H. Dijkstra, "The Trimedia TM-1 PCI VLIW Media Processor," *Proc. Hot Chips VIII Symp.*, IEEE CS Press, Los Alamitos, Calif., 1996.
7. L.T. Nguyen et al., "MSP: Multi-Media Signal Processor," *Proc. Hot Chips VIII Symp.*, IEEE CS Press, Los Alamitos, Calif., 1996.
8. P. Kalapathy, "Hardware/Software Interactions on the Mpact," *IEEE Micro*, Mar./Apr. 1997, pp. 20–26.
9. C. Lee, M. Potkonjak, and W. Mangione-Smith, "MediaBench: A Tool for Evaluating Multimedia and Communications Systems," *Proc. IEEE/ACM Int'l Symp. Microarchitecture (MICRO-30)*, IEEE Press, New York, 1997.
10. R.B. Lee and M.D. Smith, "Media Processing: A New Design Target," *IEEE Micro*, July/Aug. 1996, pp. 6–9.

Thomas M. Conte is an assistant professor of electrical and computer engineering at North Carolina State University. He received an MS and a PhD in electrical engineering from the University of Illinois, Urbana-Champaign. He is a member of IEEE, ACM, Tau Beta Pi, and Eta Kappa Nu.

Pradeep K. Dubey is a research staff member at the IBM T.J. Watson Research Center. He received a BS in electronics and communication engineering from Birla Institute of Technology, India; an MS in electrical engineering from the University of Massachusetts, Amherst; and a PhD in electrical engineering from Purdue University. He is a senior member of IEEE.

Matthew D. Jennings is currently pursuing a PhD in computer engineering at North Carolina State University. He received a BS in electrical engineering from the University of Minnesota and an MS in computer engineering from North Carolina State University.

Ruby B. Lee is chief architect for multimedia architecture and senior architect for processors and systems at Hewlett-Packard Laboratories. She received a BA from Cornell University, and an MS in computer science and a PhD in electrical engineering from Stanford University. She is a member of IEEE, ACM, Phi Beta Kappa, and Alpha Lambda Delta.

Alex Peleg is a senior computer architect at the Intel Israel Design Center, where he leads the team of architects that defined MMX. Peleg received a BS in computer science and an MS in electrical engineering from Technion, the Israel Institute of Technology. He is a member of IEEE and ACM

Mike Schlansker's biography appears on p. 69.

Peter Song is a senior analyst at MicroDesign Resources and a senior editor of Microprocessor Report.

Andrew Wolfe is director of technology and an S3 fellow at S3 Inc., in Santa Clara, Calif. Wolfe received a PhD in computer engineering from Carnegie Mellon University in 1991.

Contact Conte at the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695-7911; conte@eos.ncsu.edu.