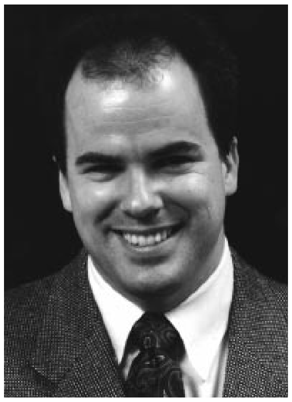

William H. Mangione-Smith
Mobile Computing Editor
UCLA
Billms@ucla.edu

Subword extensions for video processing on mobile systems

Matthew D. Jennings and Thomas M. Conte
North Carolina State University

PROVIDING VIDEO-OVER-WIRELESS capability to mobile-computing platforms results in several interesting challenges. Wireless networks provide less transmission bandwidth than hard-wired networks. Because today's wireless local-area-network technology can provide only around 2-Mbps transmission rates, video compression is essential for transmitting to mobile devices. Due to increased user sensitivity to cost and power consumption, mobile-computing platforms prefer a host-processor-only solution, opposed to a host processor in conjunction with a digital-signal processor. Most general-purpose microprocessor architectures have recently extended their instruction-set architectures to include parallel instructions for improved performance on multimedia applications, including MPEG (Motion Pictures Expert Group) video.

This essay will highlight the features of several of these extended ISAs for processing MPEG video. Each uses a modified single-instruction, multiple-data execution model as a technique to enable concurrent execution.[1] In the modified *micro-SIMD* execution model, a single instruction initiates parallel execution on data organized in parallel. Figure 1 illustrates the micro-SIMD execution of an `add` instruction.

Micro-SIMD execution using packed data types (with byte, half-word, or word quantities) makes more efficient use of the processor data path for 64- or 128-bit architectures. We'll refer to this particular form of micro-SIMD execution as *subword execution*.

MPEG video

The necessity to compress and encode video for reasonable transmission bandwidth requirements, and the corresponding computation required to decode a compressed video data stream, make video processing a compute-intensive mobile application. The MPEG standard for video compression receives the most attention as a video standard for multimedia systems.[2,3] While we do not intend to discuss the MPEG standard in detail here, we do want to touch on MPEG-1 and MPEG-2 to illustrate computational requirements.

The MPEG-1 video-coding scheme is a lossy compression one that has been optimized for coding video for digital-storage media, at rates of 1 to 1.5 Mbps. This range of rates is appropriate for transmitting data over a high-speed wireless LAN. Picture quality using MPEG-1 can be better than with analog VHS. MPEG consists of both intraframe- and interframe-coding techniques. The intraframe (I-frame) coding uses block-based discrete-cosine transform, quantization, and Huffman encoding for lossy compression. I-frame coding capitalizes on spatial redundancy in an individual frame to compress that frame. I-frame compression's lossy nature results from the quantization step; DCT and Huffman encoding are lossless (excluding precision losses for (I-)DCT).

Because the compression obtained from I-frame encoding cannot support video in the 1- to 1.5-Mbps range, the MPEG scheme also uses interframe encoding. Interframe encoding exploits temporal redundancy between frames, using a motion-estimation and motion-compensation technique. There are two interframe types, *predicted interframes* or *bidirectionally predicted interframes*. MPEG encodes P frames using motion-compensated predictions from past I or P frames. It encodes B frames using interpolated predictions from both past and future I or P frames. The compression for P frames is better than with I frames, while B frames provide the highest degree of compression. MPEG allows any sequence of I, P, and B frames.

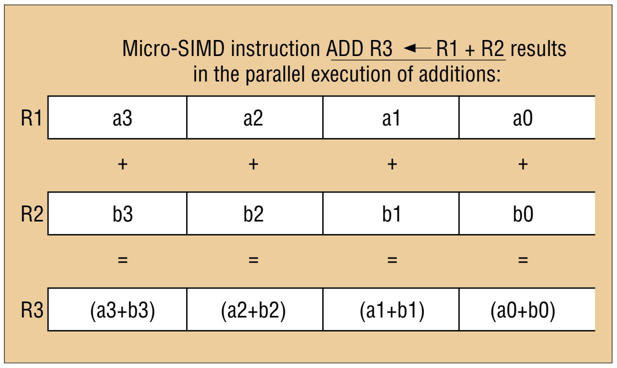
Micro-SIMD instruction ADD R3 ← R1 + R2 results in the parallel execution of additions:
R1
a3
a2
a1
a0
+
R2
b3
b2
b1
b0
=
R3
(a3+b3)
(a2+b2)
(a1+b1)
(a0+b0)

Figure 1. Micro-SIMD execution for an add instruction.

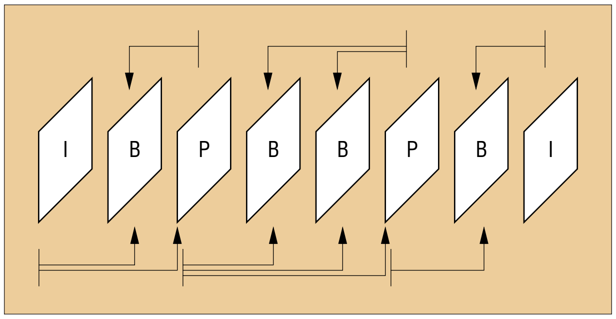

Figure 2. Example of interdependence among I, P, and B frames in a video sequence.

Figure 2 illustrates a sequence of frames and the interdependence among I, P, and B frames.

MPEG uses 8 × 8 blocks of pixels for its block-based (I-)DCT. Block sizes for motion-compensation is larger, at 16 × 16, and we call these blocks *macro blocks*. The larger macro block size represents a tradeoff between the compression provided by motion compensation and the cost associated with transmitting motion vectors.

MPEG-1 standardizes both video encoding and decoding. Figure 3a shows the encoder's block diagram. The important computational kernels for providing encoding on a host processor that supports subword ISA extensions are DCT and I-DCT, quantization and inverse quantization, motion estimation, motion compensation, and Huffman encoding. Figure 3b shows a decoder block diagram. The important computational kernels for providing decoding are Huffman decoding, inverse quantization, I-DCT, and motion compensation.

It is easier to interpolate P and B frames from the motion vectors during decoding than it is to calculate the motion vectors using motion estimation and motion compensation during encoding. This unbalanced encoding/decoding process limits MPEG-1's usefulness for video conferencing because real-time encoding is more difficult than real-time decoding. Decode will be a more important feature of a mobile multimedia system than encode, as typical use will be for video playback.

Similar to MPEG-1, MPEG-2 is a lossy video-compression scheme based on DCT coding, block-based motion-compensation, predictive and interpolative interframe coding, and Huffman coding. MPEG-2 is backward compatible with MPEG-1. Compared to MPEG-1, MPEG-2

- supports interlaced pictures,
- supports more color subsampling formats,
- supports more prediction modes for motion compensation,
- allows for more quantization matrices (and therefore more efficient encodings), and
- offers performance improvements because of scalability.

The enhancements for MPEG-2 do not change the nature of computations when compared with MPEG-1, except that MPEG-2 targets data rates in the range of 4 Mbps (for broadcast-quality TV) to 9 Mbps (for near-studio-quality video). These data rates are high for transmission over today's wireless LAN technology. The scalability improvement allows for robustness when portions of the bitstream are lost because it allows for progressive video-stream encoding. This feature can improve picture quality in the presence of higher bit-error rates inherent to wireless LAN networks. It is possible to use the scalability feature of MPEG-2 while maintaining data rates closer to the target MPEG-1 rates by having a higher mix of P and B frames than intended for MPEG-2. Correspondingly, MPEG-2 obtains higher picture quality by having a richer mix of I frames, which can operate in the presence of faster wireless networks.

Subword ISA extensions are appropriate for MPEG processing because pixel quantities are the main elements of computation. Important architectural features include subword versions of `multiply-accumulate`, `multiply-add`, or `shift-add` for (I-)DCT. `Shift-add`, or multiplication by constant values, can substitute for `multiply-accumulate` or `multiply-add` for certain (I-)DCT algorithms. Sum of accumulated differences is an essential calculation for motion-estimation. The UltraSparc with VIS has added a special-purpose instruction to perform SAD computations. Subword shift operations will be useful for performing Huffman decoding.

Subword ISA extensions to general-purpose processors

Six modern microprocessors have implemented multimedia extensions to their architectures.[4–10] Although all of these extensions are based on subword execution, key distinctions between them are in the breadth and complexity of operations added, the use of integer or floating-point registers for new packed-data types, and compatibility issues.

Multimedia Acceleration eXtensions (MAX)

The multimedia extensions found in the PA-RISC 2.0 architecture[4] (MAX-2) are second-generation multimedia extensions for HP, following MAX-1 found in the PA-7100LC microprocessor. With both MAX-1 and MAX-2, HP takes a minimalist approach to multimedia extensions. MAX-1 included only subword versions of `add`, `subtract`, `shift left & add`, `shift right & add`, and `average` for 16-bit data. `Shift left & add` and `shift right & add` are used for multiplying by integer and fractional constants, respectively. This allows for the most common form of multiplication for video processing used in (I-)DCT.

MAX-2, implemented in the PA-8000 processor, also introduces parallel shifts for data alignment and the data-conversion instructions `mix` and `permute`. The `mix` instruction mixes every other quantity of a packed data-type source register with the corresponding quantity from the second source register. The `permute` instruction has one packed data-type source and allows for any permutation of the source quantities in the packed data-type destination.

The MAX-2 extensions continue to support only subword execution on packed half-word (16-bit) data types. Packed byte (8-bit) versions were rejected for insufficient precision (even for pixel calculations) and packed word (32-bit) versions were rejected for insufficient parallelism. Using `shift & add` required a minor modification to an existing preshifter in the integer arithmetic lookup unit, rather than a new integer mul-

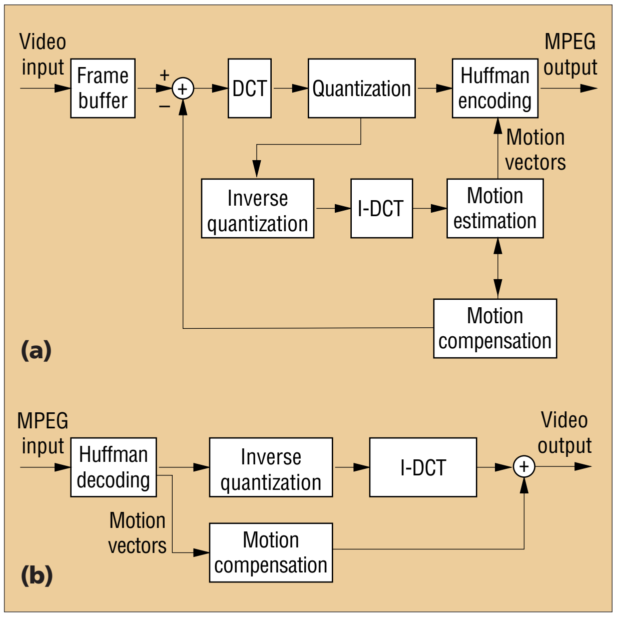

Figure 3. MPEG block diagrams: (a) encoder and (b) decoder.

tiplier functional unit, while maintaining a one-cycle latency for all integer ALU operations. MAX-2 utilizes integer registers, thereby permitting the use of existing operations for 64-bit logical operations and 64-bit shift operations. Ruby Lee's article in *IEEE Micro*[4] provides many short coding examples using MAX-2 for more complex functionality, justifying the absence of more complex operations. MAX-2 required less than 0.1% of the PA-8000 silicon area.

Matrix Math eXtensions (MMX)

For the MMX extensions to the Intel Architecture (x86),[5,6] the key challenges were retaining full compatibility and not introducing new architectural states (such as additional control register or condition codes). MMX instructions alias the floating-point registers, both for compatibility and because of the small number of integer registers. MMX achieves full compatibility with existing operating systems and applications by using the existing x86 floating-point state as temporary storage for MMX data.

MMX has some additional functionality over MAX-2, including parallel versions of `multiply`, `multiply-add`, `comparisons`, and the data-conversion instructions *`pack`* and *`unpack`*. The *`pack`* and *`unpack`* instructions allow for conversion between packed data types. Because MMX instructions are aliased to the floating-point registers, MMX includes bit-wise logical and move instructions, because integer counterparts could not be leveraged in the floating-point unit. The `move` instruction allows for the loading of data from memory or the transfer of data from the integer registers. A special instruction must maintain compatibility when switching between MMX execution and floating-point execution modes.

MMX provides subword instructions for 8-, 16-, and 32-bit quantities, but supports subword `multiply` and `multiply-add` only for 16-bit quantities. MMX includes a limited set of bit-wise logical operations.

3DNow!

The 3DNow! extensions to AMD's K6 architecture[9] are a superset of the MMX instructions that include subword versions of single-precision (32-bit) floating-point operations, allowing two single-precision quantities to execute in parallel for each 64-bit operation. While the floating-point additions are more applicable to 3D graphics and image rendering, floating-point capability can also provide more accuracy for (I-)DCT and, therefore, better-quality MPEG video. In addition, 3DNow! provides an integer subword average instruction to facilitate motion compensation and also includes data-prefetch instructions. As with MMX, the 3DNow! extensions also alias the floating-point registers for compatibility.

Visual Instruction Set (VIS)

The VIS extensions to the UltraSparc architecture are distinguished from MAX and MMX by including several instructions with higher complexity.[7] These include an instruction for pixel-distance calculation, useful for motion estimation, and block load/store instructions (others are not covered here because they are not applicable to MPEG video). Like MMX, the VIS extensions use existing floating-point registers.

The *`pdist`* (pixel-distance calculation) instruction computes the absolute difference between corresponding 8-bit components in a pair of packed byte registers and accumulates the error values. This instruction targets a predominant computation, the sum of accumulated differences, in motion estimation for MPEG video or video-conferencing protocols.

Block load-and-store instructions transfer 64-byte blocks between memory and a group of eight consecutive registers. These accesses do not cause allocations in the caches on a miss, which lets the processor move large amounts of data that are touched only once without displacing useful data from the caches. This is useful for MPEG because streaming video data does not benefit from caching and can destroy useful cache data.

AltiVec

With the AltiVec extensions, the PowerPC architecture[10] now includes a 128-bit vector execution unit, operating concurrently with the integer and floating-point units. AltiVec includes subword versions of integer (byte, half-word, word) and floating-point (single-precision) operations. Because a new execution unit is used, no existing register resources, execution resources, or opcodes could be leveraged, resulting in many additional operations. AltiVec introduces 162 new instructions covering arithmetic, compare, logical, shift/rotate, rounding/conversion, load/store, formatting, prefetch, and processor-control operations.

MIPS Digital Media eXtensions (MDMX)

Most significantly, the MDMX extensions[8] include a 192-bit accumulator and a `multiply-accumulate` instruction. Most DSP architectures include a `multiply-accumulate` instruction. The `multiply-accumulate` instruction works on packed byte or packed half-word source quantities with 3X precision for corresponding accumulator quantities. For example, a `mul-`

`tiply-accumulate` operation on 8-bit quantities has eight results of 24-bit precision for a total of 192 bits. Having a wide accumulator register saves the 2X, or even 4X, parallelism factor lost by having to unpack to a larger data type before performing a subword `multiply-accumulate`. This greater accuracy helps in the implementation of IEEE-compliant (I-)DCT. It negates the need of a special SAD instruction for motion estimation by enabling the higher-precision sum-of-difference-squared computation. MDMX also include a comprehensive set of other subword operations, including `add`, `sub`, `logical ops`, `shift`, `min`, `max`, `saturation`, `permuting`, and `comparing`. In addition, the MIPS V architecture added paired single-precision subword floating-point operations. MDMX uses floating-point registers for holding subword data types.

Table 1 summarizes key aspects for the subword extensions to host general-purpose processors.

Table 1. Subword extensions feature list.

Subword Op Class	MAX-2	MMX	3DNow	VIS	AltiVec	MDMX
Integer arithmetic, logical, shift	Yes	Yes	Yes	Yes	Yes	Yes
Multiply-add	No	Yes	Yes	Yes	Yes	No
Shift-add	Yes	No	No	No	No	No
Multiply-accumulate 192-bit accumulator	No	No	No	No	No	Yes
Floating-point arithmetic	No	No	Yes	No	Yes	Yes
Special purpose ops: pixel distance, block load/store, array, partial store	No	No	No	Yes	No	No

EACH VERSION OF SUBWORD EXTENSIONS to host general-purpose processors we've discussed includes important features for processing MPEG video. Excluding a couple of examples of special-purpose instructions for motion estimation and motion compensation, they all provide similar capabilities for the integer subword computations important to processing MPEG video. Because of microarchitectural support for pixel-distance calculations and block loads and stores, the VIS extensions should have an advantage over the other implementations. The more recent entries include floating-point subword capabilities, and more are expected to follow soon, including MMX-2. It is not clear that subword floating-point support will affect MPEG processing, because floating-point (I-)DCT algorithms are more computationally expensive.

Despite the similarities, each approach to subword extensions is unique. It is interesting to differentiate the overall approaches for supporting these multimedia extensions. Key differences include the amount of additional hardware required, ranging from MAX-2, which leverages integer registers and execution units for virtually no additional hardware, to AltiVec, which requires an entirely new execution unit. The others alias floating-point hardware, which simplifies compatibility but doesn't allow simultaneous subword and floating-point execution. Special-purpose instructions, such as those included in VIS, may have cycle-time implications for future implementations. The verdict is still out on which implementation will provide the right balance of cost, usability, and performance. All of the approaches will cut costs for mobile-computing platforms because additional DSP hardware requirements can be removed.

References

1. M.J. Flynn, "Very High-Speed Computing Systems," *Proc. IEEE*, IEEE Press, Piscataway, N.J., Vol. 54, No. 12, Dec. 1966, pp. 1901–1909.
2. V. Bhaskaran and K. Konstantinides, "Image and Video Compression Standards Algorithms and Architectures," *Kluwer Academic Publishers*, Dordrecht, The Netherlands, 1995, pp. 161–194.
3. R. Aravind et al., "Image and Video Coding Standards," *AT&T Technical J.*, Jan./Feb. 1993, pp. 58–79
4. R.B. Lee, "Subword Parallelism with MAX-2," *IEEE Micro*, Vol. 16, No. 4, Aug. 1996, pp. 51–59.
5. A. Peleg and U. Weiser, "MMX Technology Extension to the Intel Architecture," *IEEE Micro*, Vol. 16, No. 4, Aug. 1996, pp. 42–50.
6. A. Peleg, S. Wilkie, and U. Weiser, "Intel MMX for Multimedia PCs," *Comm. ACM*, Vol. 40, No. 1, Jan. 1997, pp. 25–38
7. M. Tremblay et al., "VIS Speeds New Media Processing," *IEEE Micro*, Vol. 16, No. 4, Aug. 1996, pp. 10–20.
8. "MIPS Digital Media Extension," *Instruction Set Architecture Specification*, http://www.mips.com/MDMXspec.ps (current July 29, 1998).
9. "AMD-3DNow! Technology Manual," *Instruction Set Architecture Specification*, http://www.amd.com/K6/k6docs/pdf/21928c.pdf (current July 29, 1998)
10. "AltiVec Technology Programming Environments Manual," *Instruction Set Architecture Specification*, http://www.mot.com/SPS/PowerPC/teksupport/teklibrary/manuals/altivec_pem.pdf (current July 29, 1998).

Matthew D. Jennings is a PhD student in computer engineering at North Carolina State University in Raleigh (Research Triangle Park). His research interests are in microarchitectural and compiler support for multimedia applications and static scheduling for instruction-level parallelism. He received his BS in electrical engineering from the University of Minnesota and his MS in computer engineering from NC State University. He is a member of IEEE. Contact him at Dept. of ECE, Box 7911, North Carolina State Univ., Raleigh, NC 27695-7911; mdjennin@eos.ncsu.edu.

Thomas M. Conte is an associate professor of electrical and computer engineering at North Carolina State University in Raleigh (Research Triangle Park). He received his PhD from the University of Illinois in 1992. His research interests include microarchitecture and performance evaluation. Conte is chair of the IEEE Computer Society Technical Committee on Microprogramming and Microarchitecture. He is the recipient of an NSF Career award and the IBM T.J. Watson Partnership Award for Faculty Development. Contact him at the Engineering Graduate Research Center, Box 7914, North Carolina State Univ., Raleigh, NC 27695-7914; conte@ncsu.edu.