Realtime MPEG Video via Software Decompression on a PA-RISC Processor

Ruby B. Lee
Hewlett-Packard Company
rblee@nsa.hp.com

Abstract

This paper describes the first software implementation of an MPEG video decoder that runs at realtime rates of 30 frames per second, on a general purpose microprocessor. Previously, realtime MPEG decoding could only be achieved by using special-purpose MPEG or video chips, with "programmability" either nonexistent or limited to low-level firmware. In this paper, we describe how synergistic software and hardware optimizations allow us to implement realtime, high-fidelity MPEG video and audio decompression in a software video player written in C, running on a PA-RISC microprocessor in the entry-level HP712 workstations. ThePA7100LC microprocessor described here is a fully-compliant PA-RISC 1.1 general purpose processor with a few generic instructions added to speedup parallel arithmetic on subword (16-bit) data.

1. Introduction

Traditionally, computers improved productivity by helping people compute faster and more accurately. Today, computers can further improve productivity by helping people to communicate better and more naturally. Towards this end, we at Hewlett-Packard looked for more natural ways to integrate communications power into our desktop machines. This would allow a user to access distributed information more easily, and allow him to communicate with other users more readily.

We felt that adding audio, images and video information would enrich the information media of text and graphics normally available on desktop computers such as workstations and personal computers. However, for such enriched multimedia communications to be useful, it must be fully integrated into the user's normal working environment. Hence, we decided to integrate increasing levels of multimedia support, as the technology became ripe, into both the user interface and the basic hardware platform.

In terms of user interface, we integrated a panel of multimedia icons into the standard graphical user interface, HP VUE, that comes with all HP workstations. This is part of the MPower product [13], that enables a desktop user to receive and send faxes, share printers, access and manipulate images, hear and send voice and CD-quality stereo audio, send and receive multimedia email, share an X-window or an electronic whiteboard with other distributed users, and capture and playback video sequences. MPower stands for Multimedia Power, "empower"-ing the user through more natural, multimedia communications capabilities.

In terms of hardware platforms, we integrated successive levels of multimedia support into the baseline PA-RISC[5-7] workstations. First, we integrated support through software for shared printing, shared fax, shared X, shared whiteboard, and images. All the popular image formats are supported, including JPEG[4] compressed images. Then, we added hardware and software support for audio, starting with 8 kHz voice-quality audio, followed by support for numerous audio formats including alaw, ulaw, and 16-bit linear mode, up to 48kHz, mono and stereo. This allowed high fidelity 44.1kHz stereo 16-bit CD-quality audio to be played back or recorded on HP workstations. At the same time, we supported uncompressed video capture and playback.

In January 1994, HP introduced MPower 2.0 and the entry-level enterprise desktops, the HP712, based on the multimedia-enhanced PA-RISC processor known as the PA7100LC or Hummingbird [8-10]. The video player integrated in the MPower 2.0 product is the first product that achieves *realtime* MPEG-1 video decompression [1] via software running on a general-purpose processor [3]. All existing products can only achieve realtime MPEG decompression via special-purpose chips or boards. The fact that this was achieved via a *low-end* workstation, the HP712 at 80 MHz, rather than an expensive high-end workstation, is also significant.

In this paper, we discuss the support of MPEG-compressed video as a new (video) data type. Section 2 describes MPEG compression and decompression. Section 3 describes the methodology we used for performance tuning. Section 4 describes the algorithm, software,

processor and system optimizations made to achieve real-time MPEG video decoding. Section 5 shows the performance of software MPEG decoders for different machines. Section 6 summarizes the paper.

2. The MPEG Standard

MPEG-1 (Moving Pictures Experts Group) is a video compression and decompression standard [1,2] that achieves roughly the video quality of today's analog TVs and VCRs, at a bit rate of around 1.5 Megabits per second. Studies show that this level of video fidelity is needed for sustained user acceptance. Hence, we chose MPEG-1 over other computationally simpler, but lower fidelity, software-decoded video decompression methods such as Indeo or Quicktime.

MPEG-1 defines a compressed bitstream of SIF sized frames (352x240 pixels). MPEG-1 decompression takes such an MPEG-compressed video bitstream, decompresses it, and then renders each decompressed frame. The rest of this section describes MPEG compression and decompression and may be skipped if desired.

2.1 MPEG Compression

MPEG uses a combination of compression techniques. An *intra-coded* frame, also called an I-frame, is compressed relative to itself, using only spatial redundancy reduction techniques. JPEG (Joint Photographic Experts Group) type compression for still images [4] is used.

A *non-intracoded* frame uses temporal redundancy (redundancy between frames) as well as spatial redundancy (redundancy within a frame) to reduce the number of bits required to encode the frame. Temporal redundancy tracks the motion of the same objects across frames that are close by in time. Non-intracoded frames are further divided into P-frames and B-frames. P-frames are *Predicted frames* based on comparisons with an earlier reference frame, while B-frames are *Bidirectionally-predicted frames,* using one backward reference frame and one forward reference frame. A reference frame can be an I-frame or a P-frame, but not a B-frame. By detecting motion of blocks from both a frame that occured earlier and a frame that will be played back later in the video sequence, B-frames can be encoded in even fewer bits.

Each frame is divided into macroblocks of 16x16 pixels for the purposes of motion estimation in MPEG compression, and motion compensation in MPEG decompression. A frame with only I-blocks is an I-frame, whereas a P-frame has P-blocks or I-blocks, and a B-frame has B-blocks, P-blocks or I-blocks. For each P-block in the current frame, the block in the reference frame that matches it best is identified by a motion vector. Then the differences between the pixel values in the matching block in the reference frame and the current block in the current frame is encoded by a Discrete Cosine Transform (DCT).

The color space used is the YCbCr color representation, rather than the RGB color space, where Y represents the luminance (or brightness) component, and Cb and Cr represent the chrominance (or color) components. Because human perception is more sensitive to luminance than to chrominance, the Cb and Cr components can be subsampled in both the X and Y dimensions. This means that there is one Cb value, and one Cr value, for every four Y values. Hence, a 16x16 macroblock contains four 8x8 blocks of Y, and only one 8x8 block of Cb and one 8x8 block of Cr values. This is a reduction from the twelve 8x8 blocks (four for each of the three color components), if the Cb and Cr were not subsampled. The six 8x8 blocks in each 16x16 macroblock then undergo transform coding.

Transform coding concentrates energy in the lower frequencies. The transformed data values are then quantized, by dividing by the corresponding quantization coefficient. This results in discarding some of the high frequency values, or lower frequency but low energy values, since these become zeros. Both transform coding and quantization enable further compression by run-length encoding of zero values.

Finally, the non-zero coefficients of an 8x8 block used in the DCT can be encoded via variable-length entropy encoding, e.g., Huffman coding. Entropy encoding removes coding redundancy by assigning the code words with the fewest number of bits to those coefficients that occur most frequently.

2.2 MPEG Decompression

MPEG decompression reverses the functional steps taken for MPEG compression.There are 6 basic steps involved in MPEG decompression.

First, the MPEG header is decoded. This gives a lot of information about the following block, macroblock, frame or sequence of frames.

Second, the compressed video data stream is Huffman decoded from variable-length codes into fixed-length numbers.

Next, inverse quantization is performed on the numbers to restore them to their original range.

Fourth, an Inverse Discrete Cosine Transform (IDCT) is performed on the 8x8 blocks in each frame. This converts from the frequency domain back to the original spatial domain. This gives the actual pixel values for

I-blocks, but only the differences for each pixel for P-blocks and B-blocks.

Fifth, motion compensation is performed for P-blocks and B-blocks. The differences calculated in the IDCT step are added to the pixels in the reference block as determined by the motion vector, for P-blocks, and to the average of the forward and backward reference blocks, for B-blocks. In MPEG-1, motion vectors have half-pixel resolution, which again requires taking the average of pixels.

Finally, the display step includes color conversion from YCbCr coordinates to RGB color coordinates, and writing to the frame buffer for displaying each frame in the decoded video sequence.

3. Methodology

Our philosophy is to improve the algorithms and tune the software first, resorting to hardware support only if necessary. We initially set a goal of 10 to 15 frames per second (fps) for software MPEG video decompression, since that is the rate at which motion begins to appear smooth.

We first measured the performance of the MPEG decode software we had purchased. This initially took 2 seconds to decode one frame (0.5 fps) on a 50 MHz HP720 workstation, for video only, without audio. Profiling indicated that IDCT took the largest chunk of the execution time, followed by Display, followed by Motion Compensation. The decoding of the MPEG headers was insignificant.

We then optimized the algorithms and software in each step of the MPEG decompression software, and measured the MPEG decode performance again. While we had achieved an order of magnitude improvement in frames decoded per second, the rate of 4 to 5 fps was not sufficient to meet our goal.

Hence, we entertained possible multimedia enhancements to the basic PA-RISC processor, and other system-level enhancements, that would not only speed up MPEG decoding, but also be generally useful for improving performance in other computations. Since the PA-RISC processor we were targeting, the PA7100LC [8-10], was already deep into its implementation phase, any chip enhancements we added could not adversely impact its design schedule, complexity, cycle time and chip size.

We approached this problem by studying the distribution of operations executed by the software MPEG decoder. Then, we found ways to reduce the execution time of the most frequent operation sequences. The application of algorithms enhancements, software tuning, and projected hardware enhancements was iterated until the goal was attained. The optimizations were so successful that we increased the goal from 10-15 fps to realtime rates of 24-30 fps. (Movies are frequently encoded at 24 fps, while TV shows are frequently encoded at 30 fps in the U.S. and 25 fps in Europe.)

4. Optimizations

We applied optimizations in the algorithms, software, processor and system. While the algorithm and software optimizations were MPEG specific, the processor and system enhancements are useful over a broad range of computations.

In terms of MPEG video algorithms, we improved on the Huffman decoder, the Motion Compensation and the Inverse Discrete Cosine Transform. A faster Huffman decoder, based on a hybrid of table lookup and tree-based decoding is used. The look-up table sizes were chosen to reduce cache misses. For Motion Compensation, we sped up the pixel averaging operations.

For the IDCT, we used a faster fourier transform, which significantly reduced the number of multiplies for each 2-dimensional 8x8 IDCT. In addition, we used the fact that the 8x8 IDCT matrices are frequently sparse, to further reduce the multiplies and other operations required for an 8x8 block.

The MPEG audio decompression was also done by software. This algorithm was improved by using a 32-point Discrete Cosine Transform to speed up the subband filtering [11].

In terms of software tuning, we "flattened" the code to reduce the number of procedure calls and returns, and the frequent building up and tearing down of contexts present in the original MPEG code. We also did "strength

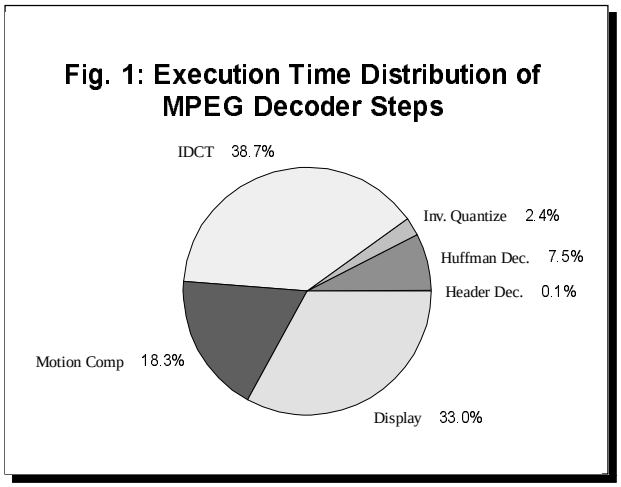


Fig. 1: Execution Time Distribution of MPEG Decoder Steps

reductions", e.g., reducing multiplications to simpler operations like shift and add operations or table lookup.

Figure 1 (and the last column of Table 1) shows the percent of execution time spent in each of the six MPEG decompression steps, after the above algorithm and software improvemements were made. Columns 2 and 3 of Table 1 show the number of instructions executed and the percent of the pathlength (total instructions executed) that this represents.

The input video sequence was an MPEG-compressed clip of a 49er football game (nfl.mpg). The total time taken is 7.57 seconds on a 99MHz PA-RISC workstation, the HP735, with 256 Kbytes of Instruction cache and 256 Kbytes of Data cache.

The largest slice of execution time (38.7%) and the largest chunk of instructions executed (38.3%) was still the IDCT. We studied the frequencies of generic operations in this group and attempted to execute them faster.

PA-RISC Multimedia Instructions

In terms of processor optimizations, we added a few PA-RISC "multimedia" instructions, which allowed simple arithmetic operations to be executed in parallel on subword data in the standard integer datapath. In particular, each 32-bit integer ALU (Arithmetic Logic Unit) in the PA7100LC was "partitioned" so that it could execute a pair of 16-bit arithmetic operations in a single cycle with a single instruction. The arithmetic operations thus accelerated were **add, subtract, average, shift_left_and_add,** and **shift_right_and_add.** The later two operations were effective in implementing multiplication by constants. They allow one operand to be shifted left or right by one, two or three bits, before being added to the other operand. The **average** instruction does parallel aritmetic averages of corresponding 16-bit elements in the two source registers.

In addition, saturation arithmetic can optionally be invoked for these parallel halfword arithmetic operations. A result is said to have a "positive overflow" if it is larger than the largest value in the defined range of the result. It is said to have a "negative overflow" if it is smaller than the smallest value in the defined range of the result. If the saturation option is used, then the result is clipped to the maximum value in its defined range if positive overflow occurs, and to the minimum value in its defined range if negative overflow occurs. This further speeds up the processing, since otherwise around ten instructions may have to be used to check for positive and negative overflows, and perform the desired clipping of the result, for a pair of operations in one instruction.

Saturation arithmetic is highly desirable in dealing with pixel values, which often represent hues or color intensities. It is undesirable to perform the normal "modular arithmetic" where overflows wrap around from the largest value to the smallest value and vice versa. For example, in 8-bit pixels, if 0 represents black and 255 represents white, a result of 256 should not change a white pixel into a black one, as would occur with modular arithmetic. In saturation arithmetic, a result of 256 would be clipped to 255.

These parallel subword arithmetic operations significantly speed up several critical parts of the MPEG decoder program, especially in the IDCT and motion compensation steps. More than half of the instructions executed for the IDCT step were these parallel subword arithmetic instructions. Their implementation did not impact the processor's cycle time, and added less than 0.2% of silicon area to the PA7100LC processor chip [8-10]. Actually, the area used was mostly previously empty space around the ALU, so that these multimedia enhancements can be said to have contributed to more efficient area utilization, rather than adding incremental chip area.

In addition, since the PA7100LC processor has two integer ALUs, we essentially have a parallelism of four halfword operations per cycle. This gives a speedup of four times, in places where the superscalar ALUs can be used in parallel. Because of the built-in saturation arithmetic option, speedup of certain pieces of code is even greater.

Table 1: Instructions and Time spent in each MPEG Decompression step on HP735

	Millions of Instructions	%Pathlength	%Time
Header decode	0.6	0.1%	0.1%
Huffman decode	55.3	10.2%	7.5%
Inv. Quantize	8.7	1.6%	2.4%
IDCT	206.5	38.3%	38.7%
Motion Comp.	79.9	14.8%	18.3%
Display	188.7	35.0%	33.0%
Total	539.7	100%	100%

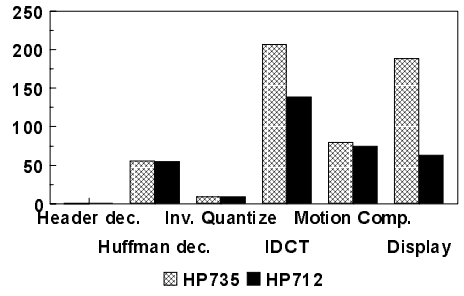
The second largest functional step (see table 1) in MPEG decompression is the display step. Here, we leveraged the graphics subsystem to implement the color conversion step together with the color recovery already being done in Artist, the graphics chip [9]. Color conversion converts between color representations in the YCbCr

color space to the RGB color space. Color recovery allows 24-bit RGB color that had been "color compressed" into 8 bits to be converted back to 24-bit color before being displayed. Color compression allows the use of 8-bit frame-buffers in low-cost workstations to achieve almost the color dynamics of 24-bit frame-buffers [12]. This leveraging of low-level pixel manipulations close to the frame buffer between the graphics and video streams also contributed significantly to the attainment of realtime MPEG decompression.

Other PA7100LC processor enhancements include the streamlining of the memory to I/O path. By having the memory controller and the I/O interface controller to the system bus integrated in the PA7100LC chip, overhead in the memory to frame-buffer bandwidth is reduced. Overhead in the processor to Artist graphics controller chip is also reduced, for both control and data.

The HP712 has a pathlength of only 340.8 million instructions compared to the 539.7 million required for the HP735, for the same MPEG decoder decompressing the same video clip, nfl.mpg. Figure 2 shows the significant decrease in instructions required for both the IDCT and Display steps, going from the HP735 to the multimedia-enhanced HP712. We significantly reduced the number of instructions required in the two largest steps of IDCT (38%) and Display (35%) in Table 1.

Fig. 2: Reduce Instructions Executed in MPEG Decoder Steps

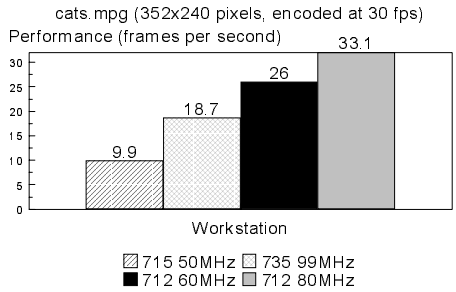


Pathlength reduction [6-7] may not translate into execution time reduction in exactly the same proportions since implementation parameters like clock rate, cache sizes and processor pipeline stalls also affect the execution time. But it clearly plays a significant role here, since the HP712 at 60MHz with 32 Kbytes each of Instruction and Data caches has a shorter execution time than the HP735 at 99 MHz with 256 Kbytes each of Instruction and Data cache.

5. Performance

The performance of the PA-RISC architectural enhancements and the leveraging of the graphics subsystem for video decompression can be seen in figure 3, for another video clip, cats.mpg, which was compressed at 30 fps. The older , high-end HP735 running at 99 MHz, achieves 18.7 fps, while the current entry-level HP712 enterprise desktops achieve 26 fps at 60 MHz, and 33.1 fps at 80 MHz. These frame decompression rates are quoted for MPEG video only (no audio) with no constraints on how fast the decoding may proceed. In other words, the decoding rate is not constrained by the rate at which the MPEG stream had been compressed. Hence, although the video clip used was MPEG compressed at 30 fps, the 80 MHz HP712 can decode it faster than 30 fps, in unconstrained mode.

Fig.3: Maximum MPEG decode rate



In the actual video player product in MPower 2.0, the decode rate is synchronized to run at the rate at which the compressed video stream was encoded. If the decoder is too fast, it waits or task switches. If it is too slow, frames are skipped, resulting in a lower effective frame rate, since skipped frames are not counted.

In order to put the performance of the entry-level HP712 workstations in perspective, it is helpful to consider the performance attained by other software MPEG decoders on different workstations or desktops, using other general-purpose processors. Figure 4 shows the reported frame rates that we were able to collect in the summer of 1994, about six to eight months after the introduction of HP's MPower 2.0 and HP712 products. We did not verify any of these numbers. The only other software MPEG decoder running in realtime was on a high-end 275 MHz Alpha workstation, in a laboratory experiment. None of the other MPEG players was being offered as a product.

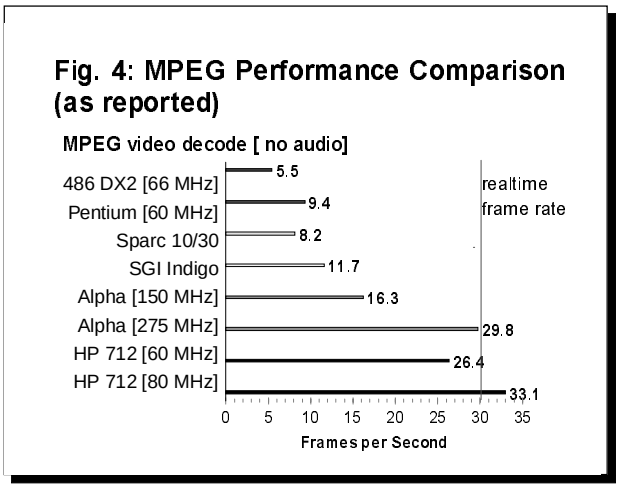


Figure 5 shows a comparison of the enhanced Berkeley software MPEG decoder [3] with the HP software MPEG decoder, running on the older HP720 (with no hardware multimedia enhancements) and the newer HP712 workstations (with hardware multimedia enhancements). It illustrates the performance obtainable with synergistic software and hardware enhancements (fourth column). The first three columns show, respectively, the performance possible with neither the software nor hardware enhancements describe in section 4 (first column), only software enhancements (second column), or only hardware enhancements (third column).

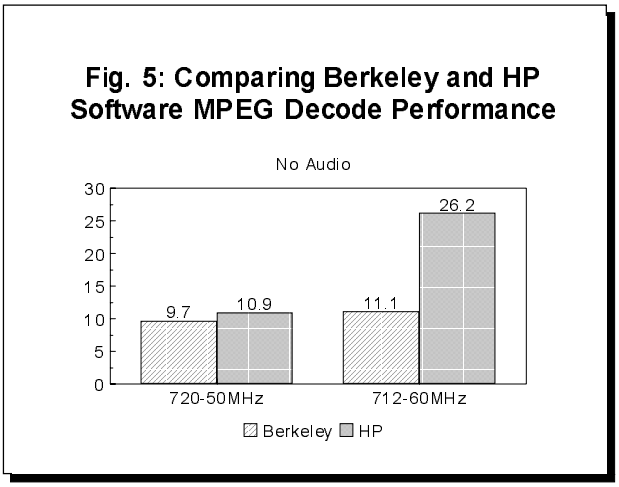
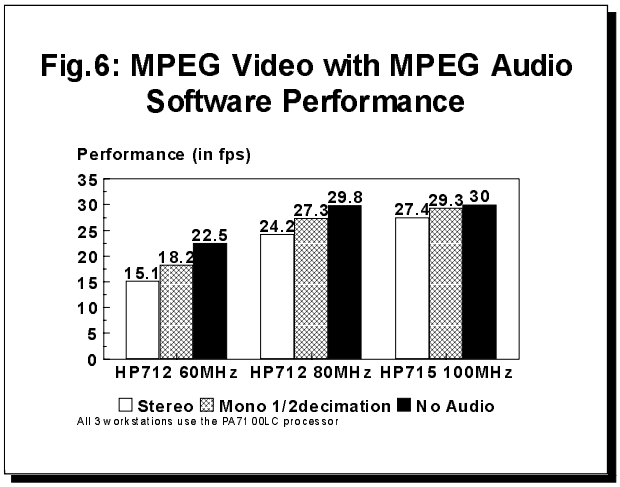


Figure 6 shows the performance when MPEG audio, of various fidelity levels, is also decompressed by software running on the general-purpose PA-RISC processor. The highest fidelity audio is stereo with no decimation, that is every audio sample comes as a pair of left and right channel values, and every sample is used. Half decimation means that one out of every two audio samples is used. (3/4 decimation means that only one out of every four audio samples is used.) Mono means that every audio sample is a single value (channel) rather than a pair of values.



Software decompression of MPEG-audio degrades the performance in terms of frames decoded per second, since audio is given higher priority than video decoding. However, even with the highest fidelity 44.1 kHz stereo 16-bit linear audio formats with no decimation, a 60 MHz HP712 workstation achieves frame-rates over 15 fps, the 80 MHz HP712 achieves a rate of 24.2 fps, and the 100MHz HP715 achieves 27.4 fps. All three workstations use the same PA-RISC processor, the PA7100LC, at three different Megahertz rates. With further enhancements of audio decoding and audio-video synchronization, we should be able to do even better.

6. Conclusions

We wanted a software approach to MPEG decoding, because we felt if video is to be useful, it has to be pervasive, and to be pervasive, it should exist at the lowest incremental cost on all platforms. With a software video decoder, there is essentially no additional cost. In addition, the evolving standards and improving algorithms pointed to a flexible solution, like software running on a general-purpose processor. Using special-purpose chips designed for MPEG decoding, or even for JPEG, MPEG and H.261 compression and decompression, would not allow one to take advantage of improved algorithms and adapt to evolving standards, without buying and installing new hardware.

Furthermore, since the performance of general-purpose microprocessors continues to improve with each new generation, we wanted to be able to leverage these improvements for multimedia computations such as video decompression. This approach also allows us to focus

hardware design efforts on improving the performance of the general-purpose processor and system, without having to replicate performance efforts in each special-purpose subsystem, such as the graphics subsystem, and the video subsystem.

The technological breakthrough is that we were the first to achieve *realtime* MPEG decoding of video streams, at 30 fps, with software running on a general-purpose processor in a low-end workstation. This was done with a combination of algorithm enhancements, software tuning, PA-RISC processor multimedia enhancements, and system tuning like combining video and graphics support for color conversions and color compression. The simple processor and system multimedia enhancements enabled us to meet and exceed our original goal of 10 to 15 frames per second for a software-based MPEG video decoder.

The PA-RISC multimedia instructions are also useful for graphics, image and audio computations, or any computations requiring arithmetic on many numbers with precision less than 16 bits. They allow parallel processing of subword data in the standard integer datapath, at insignificant addition to the silicon area. The total area used was less than 0.2% of the PA7100LC processor chip, with no impact on the cycle time or the control complexity. It is significant that such a small set of simple, generic enhancements was enough to enable realtime MPEG video decoding on the entry-level HP712. In 64-bit PA-RISC 2.0 processors [14] the parallelism is doubled, compared to the 32-bit PA7100LC processor. Complex instructions and circuitry announced recently by other microprocessor vendors may have marginal utility for improving software MPEG decoding performance.

We expect to see continuous improvement in the MPEG decoding rate as the performance of the general-purpose processors increases. With PA-RISC processors, there has been roughly a doubling of performance every 18-24 months. This would imply that larger windows, or multiple smaller windows, or MPEG-2 video streams may be decoded in the future by such multimedia-enhanced general-purpose processors.

7. Acknowledgements

The success of the MPEG software decoder performance tuning depended on close, inter-divisional teamwork distributed across four HP sites at Chelmsford, Cupertino, Palo Alto and Fort Collins. I would like to thank all members of the Multimedia Architecture Team, especially John Beck, Vasudev Bhaskaran, Peter Kaczowka, Joel Lamb, Behzad Razban, Pat McElhaton, Larry Thayer, Konstantine Konstantinides and Larry McMahan. I would also like to thank the MPower team, the PA-RISC Extensions team, especially Michael Mahon, the PA7100LC team, especially Mark Forsythe and Charlie Kohlhardt, the HP712 Gecko team, the Performance Lab, especially Tian Wang, and the Compiler Lab, for their support.

8. Bibliography

1. ISO/IEC JTC/SC29, "Coded Representation of Pictures, Audio and Multimedia/Hypermedia Information", Committee Draft of Standard ISO/IEC 11172, Dec 6, 1991.

2. LeGall Didier, "MPEG - A Video Compression Standard for Multimedia Applications", Communications of the ACM, April 1991, Vol 34 Num 4, pp 46-58.

3. Patel Ketan, Smith Brian and Rowe Lawrence, "Performance of a Software MPEG Video Decoder", Proceedings of First ACM International Conference on Multimedia, 1-6 August 1993, Anaheim California, pp. 75-82.

4. ISO/IEC JTC1/SC2/WG10, "Digital Compression and Coding of Continuous-Tone Still Images", ISO/IEC Draft International Standard 10918-1, Jan 10 1992.

5. Lee, Ruby, "Precision Architecture", IEEE COMPUTER, Vol 22 Num 1, Jan 1989, pp. 78-91.

6. Lee Ruby, Mahon Michael and Morris Dale, "Pathlength Reduction Features in the PA-RISC Architecture", Proceedings of IEEE Compcon, February 24-28, 1992, San Francisco, California, pp. 129-135.

7. McMahan Larry, and Lee Ruby, "Pathlengths of SPEC Benchmarks for PA-RISC, MIPS and SPARC", Proceedings of IEEE Compcon, February 22-26, 1992, San Francisco, California, pp. 481-490.

8. Knebel P., Arnold B., Bass M., Kever W., Lamb J.D., Lee R.B., Perez P.L., Undy S. and Walker W., "HP's PA7100LC: A Low-Cost Superscalar PA-RISC Processor", Proceedings of IEEE Compcon, February 22-26, 1993, San Francisco, California, pp. 441-447.

9. Undy Steve, et al, "A VLSI Chip-Set for Graphics and Multimedia Workstations", IEEE MICRO, Vol 14 Num 2, April 1994, pp. 10-22.

10. Gwennap Linley, "New PA-RISC Processor Decodes MPEG Video", Microprocessor Report Vol 8 Num 1, Jan 24, 1994, pp . 16-17.

11. Konstantinides K., "Fast Subband Filtering in MPEG Audio Coding", IEEE Signal Processing Letters, Vol. 1 No. 2, February 1994, pp. 26-28.

12. Barkans A., "Color Recovery: Millions of Colors from an 8-bit Graphics Device", Proceedings of IEEE Compcon, March 5-9, 1995, San Francisco, California.

13. Hewlett-Packard Journal, Vol. 45 no. 2, April 1994.

14. Hunt D., "Advanced Performance Features of the 64-bit PA8000", Proceedings of IEEE Compcon, March 5-9, 1995, San Francisco, California.