Overview of Research Efforts on Media ISA Extensions and Their Usage in Video Coding

Ville Lappalainen, Timo D. Hämäläinen, and Petri Liuha

***Abstract*—This paper summarizes the results of over 25 research groups or individual researchers that have presented video coding implementations on general-purpose processors with the new single instruction multiple data media instruction set architecture extensions. The extensions are briefly introduced and the fundamentals for extensions, as well as some inherent problems, are explained. The reported attempts to utilize the extensions are divided into kernel- and application-level, as well as platform dependent and independent optimizations. Optimized applications include, in addition to some proprietary methods, all of the major video coding standards such as H.261, H.263, MPEG-4, MPEG-1, and MPEG-2. These optimized implementations include a complete video codec, several decoders, and several encoders. Additionally, a performance comparison is given for four representative encoder implementations based on the reported results. Also included is an overview of future trends for new instructions and architectural speed-up techniques.**

***Index Terms*—H.263, MPEG-4, multimedia instructions, single instruction multiple data (SIMD) media instruction set architecture (ISA) extensions, video coding.**

I. Introduction

DESPITE the huge increase in performance of general- purpose processors for PCs and workstations, the new applications, such as real-time video encoding or decoding, have been requiring special architectural enhancements. Traditionally, processors have been optimized for executing complex program code structures without much prior knowledge of the application demands from the execution point of view. However, video and other media processing applications have basically more predictable execution behavior and require regular, massively repetitive operations, both of which could be utilized in architectural optimizations.

Single instruction multiple data (SIMD) multimedia or *media instruction set architecture (ISA) extensions* consist of new instructions and resources added to processors to improve the performance of media processing. The extensions usually aim at exploiting the existing structure and resources of the processors as much as possible. Two thorough surveys on this topic are presented by Lee [52] and Ferretti [25].

HP's PA-RISC was the first ISA to introduce its SIMD media ISA extensions, called Multimedia Acceleration eXtensions (MAX-1) [48]. Other SIMD media ISA extensions include Sun's Visual Instruction Set (VIS) [40], [77], HP's MAX-2 [50], [51], Intel's Matrix Math eXtensions (MMX) [67], Mips Digital Media eXtensions (MDMX) for MIPS processors [60], Alpha's Motion Video Instructions (MVI) [11], Motorola's Altivec [20], Intel's Streaming SIMD Extensions (SSE) [75], AMD's 3DNow! [65], MIPS-3D Application Specific Extensions (ASE) [76], and Intel's SSE2 [28].

The main emphasis of this paper is to review video encoding and decoding implementations that make use of these extensions. In the following, we first summarize how the extensions speed-up media processing. Next, the review of implementations is presented. It is organized as follows.

The first group of implementations makes only use of the new instructions without any major algorithmic, i.e., platform-independent optimizations. For this group, we first review performance evaluations of the SIMD media ISA extensions with *distinct kernels* and after those we review implementations with *complete applications*.

The second group of implementations includes both platform-independent and the use of SIMD media ISA extensions. Again, both kernel-level and application-level research results are given.

We also present performance comparisons for four most comparable implementations and give an overview of the future trends for architectural speed-up methods and further developed new instructions.

II. SIMD Media ISA Extensions

Video coding algorithms most often process byte-wide data, for which a wide arithmetic unit such as a 64-bit one inside the processor is an overkill. This is the main motivation behind *subword parallelism*, in which a standard unit of computation or storage, a word, is partitioned into smaller units called subwords. The same operation can be performed on each of the subwords in parallel. Subwords can be of different sizes, the most commonly used sizes being 8 or 16 bits for media processing.

Theoretically, subwords can be either overlapping or nonoverlapping, either partially or completely fill the word, either implemented by software or hardware and deal with either integer or floating-point data [52]. In practice, nonoverlapping subwords that completely fill the word are used. More importantly, they are implemented by hardware. This subset of subword parallelism is also referred to as packed parallelism. In MMX, MAX, VIS, and MDMX, for example, the subwords are integer subwords, while in SSE, 3DNow!, MIPS-3D ASE, and Altivec, the subwords are floating-point subwords.

Subword parallelism provides a very low-cost form of small-scale SIMD parallelism in a word-oriented processor.

Manuscript received May 28, 2001, revised February 18, 2002.

V. Lappalainen and P. Liuha are with Nokia Research Center, FIN-33721 Tampere, Finland (e-mail: ville.lappalainen@nokia.com; petri.liuha@nokia.com).

T. D. Hämäläinen is with Tampere University of Technology, Institute of Digital and Computer Systems, FIN-33101 Tampere, Finland (e-mail: timo.d.hamalainen@tut.fi).

Publisher Item Identifier 10.1109/TCSVT.2002.800865.

A word-wide integer functional unit can be partitioned into parallel subword units, with small hardware overhead.

For example, a 64-bit integer two's-complement adder may be partitioned into four 16-bit subword integer adders. Such a partitionable adder allows four 16-bit additions, or a single 64-bit addition, to be performed in a single cycle. The overhead cost is very small, since the same datapaths are used in either case: two 64-bit register reads and one register write. A superscalar processor with two 64-bit partitionable Arithmetic Logic Unit (ALUs) could support eight parallel 16-bit operations with just a 6-ported register file, while a SIMD processor with eight independent 16-bit functional units needs a 24-ported register file. A 6-ported register file is enough, because two 64-bit writes and four 64-bit reads are needed [52].

There are two main problems in subword parallel computation. First, there will be performance overhead resulting from the need to provide precise subword data alignment. In general, the programmer is responsible for data alignment; thus, programming complexity increases [41]. Secondly, conventional implementations of subword parallel computation require deep software loop unrolling, which results in code size overhead. This code size expansion may not be a serious problem with general-purpose processors, but may have significant impact on the cost of an embedded system.

A. Approaches to Data Alignment

The approaches to data alignment fall roughly into three categories, as summarized in [26]. One simple solution to achieve alignment between two data sets (e.g., filter coefficients and input data) relies on maintaining several replicas of one data set (e.g., filter coefficients). Each replica has a different alignment offset relative to the other data set.

For example, in a processor with two-way subword parallel operations, one would need two copies of the filter coefficients: one with the even data elements in the even positions of the word and the other with the odd elements in the even positions of the word. This way, depending on which output is computed, one filter copy is selected and applied to the input data. This approach can be used with MMX technology [34], for example.

The second approach relies on memory system support for misaligned accesses. Some systems provide proper alignment in hardware, for example Intel's x86 memory systems [68]. The disadvantage of these memory systems is that a single misaligned access is significantly slower than an aligned access, hence reducing performance as computation becomes memory bound. Also, a typical memory system that supports misaligned accesses is able to supply data with high throughput as long as the data is aligned to the native word size, but its performance degrades when data is misaligned. As a result, this solution is both performance suboptimal and expensive, requiring a large hardware investment.

A third approach relies on a shifter or other specialized reordering unit for constructing the misaligned components of the algorithm and issue only aligned memory accesses. This reordering unit is explicitly controlled by the programmer. This technique is used with Altivec [20] and VIS [77] technologies, for example. Some of the shortcomings of this technique are that since the construction of a misaligned data elements requires the intervention of a unit other than the one where useful computation takes place, it requires larger code for the instructions of the ordering unit, extra fetch bandwidth and extra execution time. In addition, this technique requires that the additional reordering instructions are explicitly issued in an aggressively unrolled loop. This further increases the code size.

III. Video Coding With SIMD Media ISA Extensions

Performance evaluations of a complete set of architectural features in IBM PowerPC620, Pentium Pro, and Alpha 21 164 are presented in [6], [21] and [18], respectively. However, the processors analyzed in these studies do not contain any SIMD media ISA extensions.

The papers (mentioned in Section I) that introduce specific SIMD media ISA extensions usually focus on detailed descriptions of the additional instructions and examples of their use. The performance characterization in these papers is usually limited to a few sample code segments and possibly a brief mention of the benefits anticipated on larger applications.

As already mentioned, subword parallelism computations are also possible without any hardware support. Eckart mentions a software-only technique to utilize subword parallelism in the process of describing an MPEG-1 decoder optimized for Intel Pentium [22]. A more general approach is studied thoroughly by Zucker [86].

There are lots of studies, which evaluate the performance of specific SIMD media ISA extensions, when executing either complete video coding applications or bare kernels. These studies usually describe, in a varying level of detail, how the implementation of the specific algorithm utilizes the SIMD media ISA extensions of interest. In this paper, the emphasis is on the complete applications, for which more detailed results are presented.

A. Platform-Dependent Optimization of Kernels

In this section, the studies are grouped according to the SIMD media ISAs used. Kim and Choe report the performance of SIMD floating-point extensions (3DNow!) and several DSP kernels, including an finite-impulse response (FIR) filter. They report individual speed-ups ranging from 1.3 to 1.5 for an FIR filter with tap sizes of 1–10 on a 300-MHz AMD-K6-2 processor [39].

Nguyen and John present an evaluation of Altivec and seven multimedia kernels, including 8×8 IDCT and FIR. They perform trace-driven simulation using a cycle-accurate simulator. They report individual (i.e., not average) speed-ups of 11.7 and 2.4 for IDCT and FIR on the Apple Altivec Emulator, respectively [64].

Lee and McMahan present a very detailed implementation and optimization of four typical multimedia kernels (of which two are related to video coding) using MAX-2 extensions. They report individual speed-ups of 2.7 and 4.1 for 16×16 sum of absolute differences (SAD) computation and 8×8 IDCT on an HP PA-8000 processor, respectively [53].

Talla *et al.* study three MMX-optimized applications (no video applications included) and kernels, including FIR. They also evaluate the tradeoffs in superscalar performance with a combination of measurements on Pentium II and simulation

experiments. They report an individual speed-up of 1.8 for an MMX-optimized FIR filter on Pentium II [74].

Bhargava *et al.* study four MMX-enhanced applications (no video applications included) and four kernels, including FIR. They report an individual speed-up of 1.6 for FIR on Pentium II [7].

Rice presents a detailed performance evaluation of VIS and eight image processing kernels on an actual UltraSPARC-based system. However, no results for video coding kernels are presented [71].

Another evaluation of Sun's VIS and three kernels, including an FIR filter, is presented by Chen *et al.* They use a simulated approach. They report individual speed-ups of 6.3 and 3.4 for a VIS-optimized FIR filter over a floating-point and fixed-point implementation in the Ptolemy simulation environment, respectively [14].

B. Platform-Dependent Optimization of Complete Applications

In the following, the term *frame rate* is used to denote either the *frame decoding speed* of a decoder or the *frame encoding speed* of an encoder. The implementations are grouped according to the video coding method or standard used.

Mou *et al.* report individual frame rates of 60–243 fps (frames per second) for a VIS-optimized H.261 decoder using CIF (Common Intermediate Format, 352×288) sequences on a 167-MHz Sun Ultra 1 processor. It should be noted that color conversion and display operations are included in these performance figures [62].

Lee *et al.* report individual frame rates of 67 and 33 fps for MAX-optimized H.261 and MPEG-1 video decoders on a 80-MHz HP PA-RISC workstation [8], [49]. The MPEG-1 decoder used a resolution of 320×240 [48].

Hsu and Liu report average frame rates of 23 and 65 fps for an MMX-enhanced H.263 video encoder using QCIF (Quarter Common Intermediate Format, 176×144) sequences on a 200-MHz Pentium MMX and a 400-MHz Pentium II, respectively. They have applied speed optimizations at a significant expense of compression performance. With similar quality, the output bit rate increases 1.2–1.5 times over the reference [31].

Lappalainen reports an average frame rate of 15 fps for an MMX-enhanced H.263 video encoder using QCIF sequences on a 133-MHz Pentium MMX. Bit rates ranging from 8 to 48 kbps were used [42].

Ikekawa *et al.* report individual frame rates of 115 and 25 fps for MMX-optimized MPEG-1 and MPEG-2 (Main Profile, Main Level) decoders on a 200-MHz Pentium MMX processor [33].

Defee and Huttunen report individual frame rates of 15.6–25 fps for a VIS-enhanced MPEG-2 Transport Stream decoder on a 360-MHz Sun UltraSPARC. Bit rates ranging from 4 to 15 Mbps were used [19], [32].

Lappalainen reports an average frame rate of 15 fps (QCIF) for an MMX- and SSE-enhanced proprietary video encoder called MVC on a 733-MHz Pentium III processor. Bit rates ranging from 8 to 24 kbps were used [43].

All the studies mentioned so far have focused more or less on a single, specific architecture. Ranganathan *et al.* study the performance of several video coding and image processing applications on a *variety* of experimental architectural configurations commonly used in general-purpose processors. They use a detailed *simulated* approach based on VIS extensions. They report speed-ups for an MPEG-2 (Main Profile, Main Level) decoder and encoder on an experimental processor that has an issue width of 4 and supports out-of-order execution. A bit rate of 5 Mbps and an image size of 352×240 were used [69].

TABLE I
PERFORMANCE OF VIDEO CODING APPLICATIONS ON SEVERAL PLATFORMS WITH SIMD MEDIA ISA EXTENSIONS

Application	ISA	Reference	Bit rate (kbps)	Image size	Processor	Clock freq. (MHz)	Frame rate (fps)
H.261 dec.	MAX	[48]	n/a	n/a	HP712	80	67
H.261 dec.	VIS	[62]	n/a	CIF	Sun Ultra 1	167	60-243
H.263 enc.	MMX	[31]	n/a	QCIF	Pentium MMX	200	23
H.263 enc.	MMX	[31]	n/a	QCIF	Pentium II	400	65
H.263 enc.	MMX	[42]	8-48	QCIF	Pentium MMX	133	15
MPEG-1 dec.	MAX	[48]	n/a	320x 240	HP712	80	33
MPEG-1 dec.	MMX	[33]	n/a	n/a	Pentium MMX	200	115
MPEG-2 dec.	MMX	[33]	n/a	MP,ML	Pentium MMX	200	25
MPEG-2 dec.	VIS	[19]	4000-15000	n/a	Sun Ultra	360	16-25
MVC enc.	SSE, MMX	[43]	8-24	QCIF	Pentium III	733	15

Table I summarizes the above-mentioned results. As can be seen, the most commonly used platform is Intel Pentium. Furthermore, all implementations exceed the real-time performance limit of 10 fps and most decoders also exceed the frame rates used in digital TV. However, only QCIF-sized, low-bit-rate video encoders exceed the real-time limit. It should be noted that some of the implementations here also represent the first attempts to use SIMD media ISA extensions and the clock frequencies of the processors were not very high during that time.

We can try to omit the clock frequencies and other architectural differences by considering the speed-ups when comparing the results. Table II lists the speed-ups reported with the implementations. Included is also information about the type of a reference implementation, to which the results are compared. It should be noted that the speed-up figures depend heavily on the reference version and its optimization level. Thus, in general it is difficult to compare the speed-up figures reported by different researchers. Section IV clarifies and exemplifies this issue in more detail.

Talla and John present a performance characterization of Pentium II and several multimedia applications, some of which utilize SIMD media ISA extensions, for example a commercial (RealVideo) streaming video decoder. However, no frame rate or speed-up figures are reported [73].

Zhou *et al.* describe a complexity analysis for MPEG-1 and MPEG-2 video decoding with VIS. They do not provide any ex-

TABLE II
SPEED-UPS ON VIDEO CODING APPLICATIONS OBTAINED WITH PLATFORM-DEPENDENT OPTIMIZATIONS

Application	ISA	Reference	Reference version	Reference optimized	Speed-up
H.261 decoder	MAX	[48]	n/a	n/a	2.8
H.263 encoder	MMX	[31]	C	NO	9.6
H.263 encoder	MMX	[42]	ASM	YES	1.7
MPEG-1 decoder	MMX	[29]	commercial[1]	YES	1.6
MPEG-1 decoder	MAX	[48]	n/a	n/a	2.2
MPEG-2 decoder	VIS[2]	[69]	C	NO	1.3
MPEG-2 encoder	VIS[2]	[69]	C	NO	3.5
MVC encoder	MMX	[43]	C	YES	3.4
MVC encoder	SSE, MMX	[43]	MMX	YES	1.1

[1] The Mediamatics MPEG-1 video decoder was used.
[2] Ranganathan et al. used a detailed, simulated approach based on VIS

perimental results, but derive a quantitative performance bound for software MPEG decoders [85].

C. Platform-Independent Optimization of Kernels

In all the studies in the two previous sections, the main emphasis has been on the platform-dependent optimizations, especially on the utilization SIMD media ISA extensions. There are also studies which take a more video-coding-oriented approach. They propose some important algorithmic, i.e., platform-independent optimizations. However, further improvements are realized, when implementing the optimized algorithms with SIMD media ISA extensions. It should be noted that only the studies that utilize both platform-independent optimizations and SIMD media ISA extensions are covered in the following.

Murata *et al.* propose an adaptive control method and MMX implementation for IDCT and report real-time MPEG-2 video and Dolby AC-3 audio decoding at 4 Mbps on a 266-MHz Pentium II [63]. They also report speed-ups ranging from 1.11 to 1.32 at bit rates ranging from 4 to 10 Mbps over the LLM IDCT [58] algorithm.

Winger further improves Murata's method and describes an MMX-optimized IDCT algorithm. Speed-ups ranging from 1.22 to 1.54 for IDCT at 4, 5, and 15 Mbps over Murata's method are reported [81].

Ishii *et al.* propose a parallel method and MMX implementation for variable length decoding (VLD) and inverse quantization (IQ) for MPEG-2 decoders. They also report speed-ups ranging from 1.37 to 1.48 for VLD and IQ compared to a conventional method on a 200-MHz Pentium MMX. Bit rates ranging from 4 to 10 Mbps were used [35].

Table III summarizes the above-mentioned kernel results.

D. Platform-Independent Optimization of Complete Applications

Akramullah *et al.* describe several algorithmic optimizations, which are designed especially for MMX and VIS technologies. They also describe carefully how compiler and code optimizations are applied to achieve a real-time implementation for an H.263 video encoder. They report average frame rates of 16.1 and 11.3 fps (QCIF) for MMX- and VIS-optimized versions on a 233-MHz Pentium II and a 167-MHz Sun UltraSPARC-1, respectively. Moreover, individual frame rates ranging from 35.3 to 45.7 fps on a 600-MHz Pentium III are reported. Bit rates ranging from 25 to 128 kbps were used [1], [2].

TABLE III
SPEED-UPS ON VIDEO CODING KERNELS OBTAINED WITH PLATFORM-INDEPENDENT OPTIMIZATIONS

Kernel	ISA	Reference	Reference version	Reference optimized	Speed-up
MPEG-2 decoder, IDCT only	MMX	[63]	C	n/a	1.1-1.3
MPEG-2 decoder, IDCT only	MMX	[81]	[63]	YES	1.2-1.5
MPEG-2 decoder, VLQ and IQ only	MMX	[35]	C	NO	1.4-1.5

Erol *et al.* propose several platform-independent and dependent optimizations for an H.263 decoder and encoder. They report individual frame rates of 14 and 17 fps (QCIF) for an MMX-enhanced H.263 encoder on a 200-MHz Pentium MMX [24].

Lappalainen *et al.* propose several platform-independent and dependent optimizations for an H.26L video decoder and encoder. For the highly-optimized decoder, they report an average frame rate of 117 fps (QCIF) on a 400-MHz Pentium III processor [45]. For the highly optimized encoder, they achieve an average frame rate of 17 fps (QCIF) on a 733-MHz Pentium III processor [44]. Bit rates ranging from 8 to 25 kbps were used.

Tung *et al.* report average frame rates of 72 and 20 fps for MMX-optimized MPEG-1 and MPEG-2 video decoders on a 200-MHz Pentium MMX processor. For MPEG-2, a bit rate of 8 Mbps was used [78].

Casalino *et al.* report individual frame rates of 50–62 (CIF) and 190–405 fps (QCIF) for an MMX-enhanced MPEG-4 video decoder on a 266-MHz Pentium II dual processor [12].

Moriyoshi *et al.* describe an MMX-enhanced MPEG-4 visual codec (Simple Profile, Levels 1–3), which uses a fast adaptive motion vector search during the motion estimation. They report an average frame rate of 30 fps (CIF) for the MPEG-4 codec (i.e., both encoding and decoding are included) on a 450-MHz Pentium II processor [61].

In a recent study by McVeigh *et al.*, software-based real-time MPEG-2 (Main Profile, Main Level) video encoding is demonstrated on a 500-MHz Pentium III processor with less than 70% CPU usage. The main contribution of this work is a set of algorithmic simplifications (some of which are nonconformable with the MPEG-2 standard) that reduce complexity at the expense of degraded compression performance. The fastest version of their encoder achieves approximately the same visual quality operating at 6 Mbps as the reference implementation does at 4.5 Mbps. The version producing the best quality consumes 97% of the CPU usage on a 500-MHz processor [59].

Table IV summarizes the above-mentioned frame rate results and Table V summarizes the speed-up figures. Again, MMX is the most often used ISA extension and both encoders and decoders exceed the real time performance limit. What comes to the speed-ups, the gain of using both algorithmic optimizations and SIMD media extension instructions could be even tenfold. However, when comparing to optimized reference implementations, the speed-up is typically less than two.

TABLE IV
PERFORMANCE OF VIDEO CODING APPLICATIONS ON SEVERAL PLATFORMS WITH SIMD MEDIA ISA EXTENSIONS

Application	ISA	Reference	Bit rate (kbps)	Image size	Processor	Clock freq. (MHz)	Frame rate (fps)
H.263 enc.	MMX	[1]	26-128	QCIF	Pentium II	233	16
H.263 enc.	MMX	[24]	n/a	QCIF	Pentium MMX	200	14-17
H.263 enc.	VIS	[1]	25-116	QCIF	Sun Ultra-1	167	11
H.26L dec.	MMX	[45]	8-25	QCIF	Pentium III	400	117
H.26L enc.	MMX	[44]	8-25	QCIF	Pentium III	733	17
MPEG-1 dec.	MMX	[78]	n/a	n/a	Pentium MMX	200	72
MPEG-2 dec.	MMX	[78]	8000	n/a	Pentium MMX	200	20
MPEG-2 enc.	MMX[1]	[59]	6000	720x 480	Pentium III	500	30
MPEG-4 dec.	MMX	[12]	n/a	CIF	Pentium II dual	266	50-62
MPEG-4 dec.	MMX	[12]	n/a	QCIF	Pentium II dual	266	190-405
MPEG-4 codec	MMX	[61]	n/a	CIF	Pentium II	450	30

[1] McVeigh et al. do not explicitly describe MMX-optimizations, they just mention that the subsampling structure they use in motion estimation is amenable to SIMD processing [59].

IV. COMPARISONS OF OPTIMIZED IMPLEMENTATIONS

It is quite demanding to compare the performance of reported implementations, because of both different experimental arrangements and different performance metrics. Moreover, both of these factors are usually documented with different levels of details.

When evaluating the performance of video encoding or decoding applications, the following pieces of information about the experiments should be documented so as to yield reliable results. First, the properties of the source video sequences should be described. Second, both the encoding parameters that were used to generate the encoded bit streams and all implementation alternatives that can be selected freely such as block-matching algorithm should be explained. Finally, the experimentation environment should be documented. The items of information in these three categories are divided into two classes called "required info" and "useful info," as shown in Tables VI–VIII.

The following performance metrics are commonly used. First, a measure of complexity such as average decoding time of a frame or encoding or decoding speed in frames per second is required. Second, a measure of video quality such as PSNR would be very useful. The PSNR should be reported at least for the luminance components of the frames. Third, is would be very useful to report the compression performance by using PSNR versus bit rate curves. Finally, it would be useful to compare also subjective video quality in some cases, for example when there are two competing proposals targeting a video coding standard. More details on performance evaluation and comparison of video coding systems are given in [46] and [9]; the former focuses on complexity, the latter on video quality and compression performance.

TABLE V
SPEED-UPS ON VIDEO CODING APPLICATIONS OBTAINED USING BOTH PLATFORM-INDEPENDENT OPTIMIZATIONS AND SIMD MEDIA ISA EXTENSIONS

Application	ISA	Reference	Reference version	Reference optimized	Speed-up
H.263 encoder	MMX	[1]	C	NO	9.2
H.263 encoder	MMX	[24]	C	NO	1.8-2.3
H.263 encoder	VIS	[1]	C	NO	22.3
H.26L decoder	MMX	[45]	C	NO	8.6
H.26L decoder	MMX	[44]	C	YES	1.3
H.26L encoder	MMX	[44]	C	YES	1.8
MPEG-1 decoder	MMX	[78]	commer-cial[1]	YES	1.2
MPEG-2 decoder	MMX	[78]	commer-cial[2]	YES	1.5
MPEG-4 decoder	MMX	[12]	C	YES	1.2-1.3
MPEG-4 decoder	MMX	[61]	C	NO	1.9
MPEG-4 encoder	MMX	[61]	C	YES	4.4

[1] The Cyberlink MPEG-1 decoder (VCDPlayer) was used.
[2] The Cyberlink MPEG-2 decoder (DVDPlayer) was used.

TABLE VI
SOURCE VIDEO SEQUENCES

Name	required info
Spatial frame resolution	required info
Input (reference) frame rate	required info
Duration	required info
Classification of motion and spatial details	useful info

TABLE VII
ENCODING PARAMETERS AND IMPLEMENTATION ALTERNATIVES

Target frame rate (or # of frames coded)	required info
Fixed or variable target frame rate	required info
Quantization parameter (QP)	required info
Fixed or variable QP	required info
Bit rate	required info
Rate control method	required info
Block-matching algorithm	required info
Search window size	required info
DCT algorithm	required info
Optional coding modes	required info
Platform-independent (P-I) optimizations	required info
Platform-dependent (P-D) optimizations	required info

TABLE VIII
EXPERIMENTATION ENVIRONMENT

Processor and clock frequency	required info
Level 1 (L1) cache size	required info
L2 cache size	useful info
System bus bandwidth	useful info
Memory size	useful info
Operating system	required info
Compiler	required info
Compiler optimizations	required info

In the following, comparisons are given for four most similar implementations of an H.263 encoder reported by Lappalainen [42], [44] (two different implementations), Erol *et al.* [23], [24] and Akramullah *et al.* [1], [2].

TABLE IX
SOURCE VIDEO SEQUENCES

Name	Akiyo	Foreman
Spatial frame resolution	176x144 (QCIF)	
Input frame rate	30 fps	
Duration	300 frames	

TABLE X
ENCODING PARAMETERS AND IMPLEMENTATION ALTERNATIVES

Parameter	Author	
	Erol et al.	Lappalainen
Target frame rate	10 fps	9 or 10 fps
Fixed or variable target frame rate	Variable	Fixed
Quantization parameter (QP)	18	14 or 19
Bit rate	<8 or >28 kbps	8 or 28 kbps
Fixed or variable QP	Fixed	Variable
Rate control method	Fixed QP	TMN-5 [36]
Block-matching algorithm	computation-constrained layered search [27]	
Search window size	ca. 10-15 positions required [27]	
DCT algorithm	Arai et al. [3]	Chen et al. [13]
Optional coding modes	none	Annexes D & F
P-I optimizations	4 operations	3 operations
P-D optimizations (MMX)	8 operations	8 operations

TABLE XI
EXPERIMENTATION ENVIRONMENT

Author	Erol et al.	Lappalainen
Processor, clock frequency	Pentium MMX, 200 MHz	Pentium MMX, 133 MHz
Level 1 (L1) cache size	16 KB (instr.) + 16 KB (data)	
Operating system	n/a	Windows 95
Compiler	n/a	MS Visual C++ 5.0
Compiler optimizations	n/a	Maximize speed

A. Performance of H.263 Encoder Implementations on Pentium Processor With MMX Technology

Tables IX–XI summarize the information about the experiments used in the two studies; the first is conducted by Erol *et al.*, the second by Lappalainen. [1]

Erol *et al.* use algorithmic optimizations for DCT, IDCT, quantization, and half-pixel motion estimation. Additionally, they use platform-dependent MMX optimizations on DCT, IDCT, integer, and half-pixel motion estimation, image interpolation, motion compensation, SAD calculation, data interleaving, and memory copying routines.

Lappalainen uses algorithmic optimizations for half-pixel motion estimation and quantization. Additionally, MMX optimizations on DCT, IDCT, quantization and inverse quantization, integer and half-pixel motion estimation, image interpolation, motion compensation, and SAD calculation are used.

Due to the use of variable bit rate, Erol *et al.* always achieve the target frame rate of 10 fps, while in Lappalainen's study, the target bit rate is always achieved but the target frame rate may not always be reached. Lappalainen uses the Unrestricted Motion Vector (Annex D of H.263) and Advanced Prediction (Annex F of H.263) Modes, which increases complexity.

[1]It should be noted that most of the commonly used source sequences can be found at http://kbs.cs.tu-berlin.de stewe/vceg/sequences.htm or http://standard.pictel.com/ftp/video-site/sequences/.

TABLE XII
PERFORMANCE OF TWO H.263 ENCODER IMPLEMENTATIONS ON 200 MHZ PENTIUM PROCESSOR WITH MMX TECHNOLOGY

Sequence	Bit rate (kbps)	Target frame rate (fps)	Author	Frame encoding speed (fps)
Akiyo	<8	10	Erol et al.	17
Akiyo	8	10	Lappalainen	26[1]
Foreman	>28	10	Erol et al.	14
Foreman	28	9	Lappalainen	21[1]

[1] Scaled up.

The following experiments performed by Lappalainen correspond quite accurately to the ones performed by Erol *et al.*: Akiyo at 8 kbps with the average QP of 14 and frame rate of 10 fps and Foreman at 28 kbps with the average QP of 19 and frame rate of 9 fps. Erol *et al.* achieve frame rates of 14 fps and 17 fps for the Foreman and Akiyo sequences, respectively.

If Lappalainen's results, obtained on a 133-MHz Pentium MMX, are scaled up to correspond to the results of Erol *et al.* obtained on a 200-MHz processor, the resulting frame rates are about 1.5 times higher: 21 and 26 fps for Foreman and Akiyo, respectively. The results of this comparison are summarized in Table XII.

This scaling is reasonable and accurate because there are no architectural differences between these two processors; the only difference is the clock frequency. More details on scaling the results can be found in [2].

In fact, this comparison should be reliable, because of several equivalent implementation options such as the block-matching algorithm and equivalent encoding parameters such as video sequences and reference frame rates. Although the bit rates and QPs are not equivalent, they are sufficiently close to each other to make the comparison accurate. However, for a complete comparison, it is very important to take into account objective video quality. Furthermore, the experimentation environment should be identical.

B. Performance of H.263 Encoder Implementations on Pentium III Processor

Akramullah *et al.* describe the performance and several optimizations of an H.263 encoder on both Pentium II and Pentium III processors. Although as accurate a comparison as the one presented above is not possible, because of several different encoding parameters and implementation options, it is still reasonable to perform comparisons with the results obtained on Pentium III by Lappalainen *et al.*Tables XIII–XV summarize the information about the experiments used in the two studies.

Akramullah *et al.* utilize extensive algorithmic, code and compiler optimizations including MMX optimizations of DCT, IDCT, and motion estimation. They use a fast, zone-based, block-matching algorithm [30] with some modifications [1]. The DCT algorithm used is described in [70, Appendix A.2].

The H.263 encoder used by Lappalainen *et al.* in [44] is similar to the one used in [42]; the only difference is the two-level hierarchical motion estimation with the full-search block-matching algorithm.

TABLE XIII
SOURCE VIDEO SEQUENCES

Name	Claire, Grandma, Miss America, Salesman	Akiyo
Spatial frame resolution	176x144 (QCIF)	
Input frame rate	30 fps	
Duration	300 frames	
Classification of motion	Slow motion only	

TABLE XIV
ENCODING PARAMETERS AND IMPLEMENTATION ALTERNATIVES

Parameter	Author	
	Akramullah et al.	Lappalainen
Target frame rate	30 fps	10 fps
Fixed or variable target frame rate	Fixed	
Quantization parameter (QP)	10	4 or 6
Bit rate	25.8, 27.0, 31.2, or 41.1 kbps	26.8 or 44.3 kbps
Fixed or variable QP	Fixed	
Rate control method	Fixed QP	
Block-matching algorithm	zone-based search [30], [1]	2-level hiearchical search
Search window size	31x31	
DCT algorithm	Appendix A.2 of [70]	Chen et al. [13]
Optional coding modes	none	Annexes D & F
P-I optimizations	3 operations	3 operations
P-D optimizations (MMX)	4 operations	8 operations

TABLE XV
EXPERIMENTATION ENVIRONMENT

Author	Akramullah et al.	Lappalainen
Processor, clock frequency	Pentium III, 600 MHz	Pentium III, 733 MHz
Level 1 (L1) cache size	16 KB (instr.) + 16 KB (data)	
Operating system	n/a	Windows NT 4.0
Compiler	MS Visual C++	MS Visual C++ 6.0
Compiler optimizations	Maximize speed	

The Akiyo at 26.8 and 44.3 kbps experiments performed by Lappalainen *et al.* correspond approximately to Akramullah's results on Claire at 27.0 kbps, Grandma at 25.8 kbps, Miss America at 31.2, and Salesman at 41.1 kbps. Akramullah achieves frame encoding speeds of 46, 45, 45, and 44 fps for Claire, Grandma, Miss America, and Salesman, respectively.

If the results of Lappalainen *et al.*, obtained on a 733-MHz Pentium III are scaled down to correspond to Akramullah's results obtained on a 600-MHz processor, the resulting frame encoding speeds are about 1.2 times higher: 54 fps and 53 fps for Akiyo at 26.8 and 44.3 kbps, respectively. However, Akramullah *et al.* have encoded every frame of the sequence, while Lappalainen has encoded only every third frame of the sequence, as the target frame rates of 30 and 10 fps indicate. In general, the complexity of whole encoding task does not grow linearly with the target frame rate, because motion estimation is less complex with higher target frame rates. In Akramullah's implementation, motion estimation consumes about 30% of the execution time. Thus, it can be concluded that Akramullah's implementation is clearly faster. However, for a complete comparison, it is very important to take into account objective video quality, as already mentioned. The results of this comparison are summarized in Table XVI.

TABLE XVI
PERFORMANCE OF TWO H.263 ENCODER IMPLEMENTATIONS ON 600-MHZ PENTIUM III PROCESSOR

Sequence	Bit rate (kbps)	Target frame rate (fps)	Author	Frame encoding speed (fps)
Akiyo	26.8	10	Lappalainen	54[1]
Akiyo	44.3	10	Lappalainen	53[1]
Claire	27.0	30	Akramullah et al.	46
Grandma	25.8	30	Akramullah et al.	45
Miss America	31.2	30	Akramullah et al.	45
Salesman	41.1	30	Akramullah et al.	44

[1] Scaled down.

V. FUTURE TRENDS

Recently, there have been studies which propose some hardware or compiler techniques to improve the performance obtained with today's ISA extensions. The efforts are trying to increase the performance in five distinct main areas.

First, the number of operations executed per one instruction is being extended so that more complex functionality can be performed. Additionally, reconfigurable techniques can be used.

Second, the level of parallelism can be increased. Multithreaded architectures combined with both SIMD media ISA extensions and long vector architectures have been proposed. Utilization of thread-level parallelism is reasonable, because according to our experiences on video coding, the data-level parallelism cannot be easily exposed beyond the currently available level of parallelism in SIMD media ISAs.

Third, flexibility in data-dependent control constructs is increased by conditional vector processing and automatic data alignment. This flexibility is necessary in multimedia applications if the data-level parallelism is to be increased because multimedia applications differ from traditional applications that are executed on vector processors (e.g., scientific computations) in that they have more data-dependent computations.

Fourth, research on optimization of subword sizes has yielded at least two improvement proposals; arbitrary boundary-packed arithmetic, as well as hardware mechanisms for dynamically recognizing and capitalizing narrow-bit-width operations, have been proposed.

Finally, research on compilation techniques has reduced the complexity of the programming task.

A. Increasing Functionality

Ueng *et al.* describe the design of ISA extensions, called a Multimedia Function Unit (MFU), utilized in NSC-98, which is an experimental MMX-compatible CPU. They propose three additional instructions: 1) permutation; 2) parallel distance computation; and 3) parallel average computation. They also study various design alternatives related to MFU subunits. The proposed instructions improve further the original MMX

extensions, especially when video coding algorithms are implemented and the same instructions have been implemented in some commercial SIMD media ISA extensions such as VIS and SSE. For example, the parallel distance computation is essentially a video encoding instruction and it constitutes the majority of computational speed-up of the given examples [79].

Villalba *et al.* propose an MMX-like architecture extension to support the vector rotation operation, which is derived from the well-known CORDIC architecture. It supports many types of signal processing algorithms such as DCT with quite small modifications. They approach could be extended to support other butterfly type of operations on vectors, although this can also be achieved with certain permutation operations as described below [80].

Berekovic *et al.* propose extensive ISA extensions specialized for MPEG-4 video coding and decoding. These extensions add function-specific or almost application-specific blocks to the datapath of a CPU. In addition to block-based coding (MPEG-4 simple profile), they target their extensions at object-based coding (MPEG-4 core or main profiles), which are not as generic as the traditional media ISA extensions. They target the new functions that are being incorporated in new multimedia standards and not so much present in earlier standards such as MPEG-1 and MPEG-2 [5].

Lee proposes a new permutation instruction, which achieves maximum subword permutation performance with half the hardware cost of the previous solution. She defines subword rearrangements for 2-D blocks [55].

Lee also defines an alphabet of fundamental permutation primitives along with new permutation instructions covering these primitives. Together, all these permutations extend the generic usefulness of SIMD media ISA extensions by enabling nonstraight-forward data rearrangements [56].

Yang *et al.* extend the work of Lee by proposing a new instruction that reduces the subword permutation problem to a single butterfly type permutation instruction. The use of this instruction is limited to a given subword width. It simplifies the permutation problematic, although it does not provide optimum implementation for every case compared to Lee's proposals [84].

Wong *et al.* propose a multimedia enhanced general-purpose processor architecture called M + GPP. Additionally, they discuss reconfigurable hardware units to support the execution of new media instructions [82]. In a recent study, they extend their previous study by proposing new instructions for the DCT and Huffman operations. Simulation results are also provided to show the usefulness of the extensions [83]. The use of reconfigurable blocks for ISA extensions would open new possibilities to speed-up a variety of functions without being too function-specific. Wong have described the possible control mechanisms of such units. However, further studies would be needed concerning the optimum set of basic building blocks in these reconfigurable units for multimedia applications, for example.

B. Increasing Parallelism

Many proposals to increase parallelism attempt to include traditional vector processing instead of combining a complex superscalar core with short vector processing.

Lee and Stoodley started with MIPS R10000 architecture and modified it to represent a vector processor with long vectors and simple control logic [54]. They show that a two-issue, 64-long vector processor outperforms a four-issue superscalar with short vector extensions with equal area cost. Their proposal is limited to uni-threaded architecture, which is suitable for speeding up kernel computation.

Simultaneous Multi Thread (SMT) execution is rapidly becoming applied in implementations. Oehring *et al.* model PowerPC 604 and extend it to support SMT processing with MMX-like ISA extensions [66]. They suggest that a four-issue, at least two-threaded architecture would be optimal over the reference superscalar architecture.

Corbal *et al.* use R10000 as a reference and have presented a matrix-oriented architecture that combines traditional ISA SIMD to vector processing [16], [17]. They show that SMT provides 2.1X speed-up over reference MMX model. SMT combined to their matrix architecture provides 3.3X increase in performance over uni-threaded MMX model. As a conclusion, they propose a future media processor should combine SMT and scalar oriented SIMD extensions to exploit explicit thread-level parallelism and decrease issue width and instruction fetch bandwidth.

Several proposals have also been made for system-on-chip or chip-multiprocessor implementations, where several simple processors contribute the media computation. In one extreme, the processor consists of dedicated streaming units. However, these cannot be considered as extensions to general-purpose processors and are not considered in this paper.

C. Increasing Flexibility of Control Constructs

Smith *et al.* provide a good summary of support of conditional operations on vector processors to date. Moreover, they conclude that solutions whose performance depends on the fraction of valid values in a vector are preferred to those whose performance depends simply on the length of the vector. They also state that the current SIMD media ISA extensions, which use short vectors with SIMD implementations, are to be replaced by long vector, pipelined implementations in the future. According to them, long vectors have a number of advantages and are a logical next step in media ISA development [72].

Kapasi *et al.* discuss efficient conditional operations for data-parallel architectures, especially for SIMD architectures. In general, data-dependent control constructs reduce efficiency of data-parallel architectures, as the constructs do not map well to these architectures. Conditional streams are a mechanism to convert these control constructs into data-dependent data movement operations (routing) and can result in significant speed-ups on media processing applications such as a speed-up of 1.8 for polygon rendering, as shown by the results obtained from simulations [36].

Fridman presents an approach to data alignment for subword parallel computation using an alignment resource called data alignment buffer (DAB). The benefits of this approach over those in other processors are that it achieves high computational efficiency primarily because the DAB can be tightly scheduled in software, which results in smaller code (requiring less software loop unrolling) [26].

D. Optimization of Subword Sizes

Brooks and Martonosi propose hardware mechanisms that dynamically recognize and capitalize on narrow-bitwidth (16 bits or less) instances in programs without any compiler intervention. They describe two optimizations: power- and performance-oriented ones. The latter one is more interesting in the context of this study. It improves performance by merging together narrow-integer operations and by allowing them to share a single functional unit. Simulated results show speed-ups of 7% and 15% for MPEG-2 decoding and encoding, respectively [9].

Balakrishnan and Nandy propose a subword parallel scheme with arbitrary sized subwords in addition to currently used sizes of 8, 16, 32, and 64 bits. The motivation behind their work is based on the fact that several operations, such as DCT and motion compensation, need also different sizes, e.g., 12- and 9-bit sized subwords. They also discuss the implementation of arbitrary boundary-packed addition [4].

Karthikeyan and Ranganathan extend the work on arbitrary boundary packed arithmetic by presenting a packed multiplication scheme based on the Wallace tree algorithm. They also describe how to implement saturation arithmetic for arbitrary boundary-packed addition [38].

E. Compiler Issues

In general, there has been no compiler support for producing optimized MMX or any other code based on SIMD media ISA extensions. Thus, high-performance MMX code requires writing hand-optimized assembly code either by using inline assembly code available in, e.g., Microsoft's compiler or using C wrappers such as intrinsics available in Intel's compiler. Vectorizing compilers that can turn scalar C code into parallel MMX assembly code without user intervention have been introduced by Intel, for example. Still, the performance of the code generated by the compiler has not reached that of hand-optimized code in general. Perhaps the easiest, but the most limited way of utilizing SIMD media ISA extensions is to use existing library code, such as Intel's Image Processing Library. Recently, few studies have addressed this problem.

Conte *et al.* discuss the complexity of the programming model of MMX and SSE. Moreover, they introduce a programming methodology and the Aphelium compiler. Of video coding operations, IDCT is selected as an example case. It is demonstrated that Aphelium is able to output high-quality MMX code that matches the speed of the code hand-optimized by Intel, without requiring deep assembly-coding expertise from the user. The performance of Aphelium is based on optimizing the code for MMX and Intel P6 processor core, which is used in all Pentium processors since Pentium Pro. Thus, there is no support for other ISA extensions or processor cores [15].

Larsen and Amarasinghe propose a robust compiler algorithm for synthesizing SIMD instructions from the statements in single basic blocks instead of in loop nests only. Although their approach can be applied to many SIMD media ISA extensions, they believe Altivec can best utilize the algorithm. They report speed-ups for SPEC95 fp and multimedia kernels such as FIR filtering and RGB to YUV conversion that range from 1.2 to 6.7 on a 450-MHz Motorola MPC7400 processor with the Altivec extensions. Most of the benchmarks use single-precision floating point numbers; RGB to YUV conversion uses 16-bit integers [47].

Leupers presents a novel code selection technique capable of exploiting SIMD instructions also when compiling plain C source code. It permits to take advantage of SIMD instructions while still using portable source code. His approach builds on the classical tree-based code selection paradigm, but it generates alternative covers. The detailed code selection is performed only later, when enough information for the generation of SIMD instructions for an entire data flow graph is available. The results for 2- and 4-parallel SIMD processing, presented for TI C62xx and Philips Trimedia TM1000 show that it is possible to reduce the number of compiler-generated machine instruction by using portable C code. A 50% reduction in the instruction count was obtained for the vector add and image decomposition kernels [57].

VI. Conclusion

As numerous research efforts on SIMD media ISA extensions indicate, the new, proposed hardware techniques contain potential for improved performance.

On the other hand, the continuous research efforts on the optimized implementations of video coding algorithms produce increasingly higher performance on existing processors. Here, the challenge of compilation and code generation will probably remain and might even increase with the newly proposed hardware techniques. Currently, obtaining high-performance code requires manually writing hand-optimized assembly code.

In an open application development environment, there is a clear demand for supporting different functions for compilation. This could be achieved via special fixed programming models, data types, or Application Programming Interface (APIs) that would implement larger functional entities than single instructions. As is well known, such an approach (e.g., OpenGL) has taken place in the 3-D computer graphics domain. As video applications and their basic algorithms will stabilize more, this kind of approach could also be possible.

In the context of 3-D graphics applications, first there was a standardized API, and only after that, hardware implementations of the functions in this API were realized. Contrary to 3-D graphics applications, video applications first utilized hardware implementations of several algorithms such as DCT, and only in the future could a standardized, more general API potentially be expected.

Recently, DSPs such as TI's TMS320C64x and embedded general-purpose processors such as ARM have announced their SIMD media ISA extensions. Although the majority of current research on SIMD media ISA extensions is focused on MMX, SSE, MAX, and VIS extensions available in PC and workstation processors, these extensions share a number of fundamental similarities with the ones available in modern embedded processors. Thus, to a certain extent, the current research results are applicable to mobile devices that utilize modern embedded processors.

Acknowledgment

The authors would like to thank Dr. R. Castagno and Prof. J. Takala for useful discussions and comments. Also, the authors would like to thank the anonymous reviewers who have helped to improve the quality of this paper.

REFERENCES

[1] S. M. Akramullah, "Software-based video encoding using high-performance computing," Ph.D. dissertation, Dept. Elect. Electron. Eng., The Hong Kong Univ. of Sci. and Technol., Hong Kong, 1999.

[2] ——, "Optimization of H.263 video encoding using a single processor computer: Performance tradeoffs and benchmarking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 901–915, Aug. 2001.

[3] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," *Trans. IEICE*, pp. 1095–1097, 1988.

[4] S. Balakrishnan and S. K. Nandy, "Arbitrary precision arithmetic—SIMD style," in *Proc. 11th Int. Conf. VLSI Design*, 1998, pp. 128–132.

[5] M. Berekovic *et al.*, "Instruction set extensions for MPEG-4 video," *J. VLSI Signal Processor 23*, pp. 27–49, 1999.

[6] D. Bhandarkar and J. Ding, "Performance characterization of the Pentium Pro processor," in *Proc. 3rd Int. Symp. High-Performance Computer Architecture*, 1997, pp. 288–297.

[7] R. Bhargava, L. K. John, B. L. Evans, and R. Radhakrishnan, "Evaluating MMX technology using DSP and multimedia applications," in *Proc. 31st Annu. ACM/IEEE Int. Symp. Microarchitecture MICRO-31*, 1998, pp. 37–46.

[8] V. Bhaskaran *et al.*, "Algorithmic and architectural enhancements for real-time MPEG-1 decoding on a general purpose RISC workstation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 380–386, Oct. 1995.

[9] G. Bjontegaard, "Recommended Simulation Conditions for H.26L," ITU-T SG16 Doc. VCEG-M75, 2001.

[10] D. Brooks and M. Martonosi, "Dynamically exploiting narrow width operands to improve processor power and performance," in *Proc. 5th Int. Symp. High-Performance Computer Architecture*, 1999, pp. 13–22.

[11] D. A. Carlson, R. W. Castelino, and R. O. Mueller, "Multimedia extensions for a 550-MHz RISC microprocessor," *IEEE J. Solid-State Circuits*, vol. 32, pp. 1618–1624, Nov. 1997.

[12] F. Casalino *et al.*, "MPEG-4 video decoder optimization," in *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, vol. 1, 1999, pp. 363–368.

[13] W.-H. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the Discrete Cosine Transform," *IEEE Trans. Commun.*, vol. COM-25, pp. 1004–1009, 1977.

[14] W. Chen *et al.*, "Native signal processing on the Ultrasparc in the Ptolemy environment," in *Conf. Rec. 30th Asilomar Conf. Signals, Systems and Computers*, vol. 2, 1997, pp. 1368–1372.

[15] G. Conte *et al.*, "The long and winding road to high-performance image processing with MMX/SSE," in *Proc. 5th Int. Workshop Computer Architectures for Machine Perception*, 2000, pp. 302–310.

[16] J. Corbal *et al.*, "Exploiting a new level of DLP in multimedia applications," in *Proc. MICRO -32*, Haifa, Israel, Nov. 1999.

[17] ——, "DLP + TLP processors for the next generation of media workloads," in *Proc. 7th Int. Symp. High-Performance Computer Architecture HPCA*, 2001, pp. 219–228.

[18] Z. Cvetanovic and D. Bhandarkar, "Performance characterization of the Alpha 21 164 microprocessor using TP and SPEC workloads," in *Proc. 2nd Int. Symp. High-Performance Computer Architecture*, 1996, pp. 270–280.

[19] I. Defee, "Software decoding of HDTV," *IEEE Trans. Consumer Electron.*, vol. 45, pp. 1277–1283, Nov. 1999.

[20] K. Diefendorff *et al.*, "AltiVec extension to PowerPC accelerates media processing," *IEEE Micro*, vol. 20, pp. 85–95, Mar.–Apr. 2000.

[21] T. A. Diep, C. Nelson, and J. P. Shen, "Performance evaluation of the PowerPC 620 microarchitectur," in *Proc. 22nd Annu. Int. Symp. Computer Architecture*, 1995, pp. 163–174.

[22] S. Eckart, "High performance software MPEG video player for PC's," in *Proc. SPIE*, vol. 2419, Feb. 1995, pp. 446–454.

[23] B. Erol *et al.*, "Implementation of a fast H.263 + encoder/decoder," in *Conf. Rec. 32nd Asilomar Conf. Signals, Systems, and Computers*, vol. 1, 1998, pp. 462–466.

[24] ——, "Efficient coding and mapping algorithms for software-only real-time video coding at low bit rates," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 843–856, Sept. 2000.

[25] M. Ferretti, "Multi-media extensions in super-pipelined microarchitectures. A new case for SIMD processing?," in *Proc. 5th IEEE Int. Workshop Computer Architectures for Machine Perception*, 2000, pp. 249–258.

[26] J. Fridman, "Sub-word parallelism in digital signal processing," *IEEE Signal Processing Mag.*, pp. 27–35, Mar. 2000.

[27] M. Gallant *et al.*, "An efficient computation-constrained block-based motion estimation algorithm for low bit rate video coding," *IEEE Trans. Image Processing*, vol. 8, pp. 1816–1823, Dec. 1999.

[28] P. N. Glaskowsky, "Pentium 4 (Partially) Previewed," Cahners Microprocessor Report—The Insider's Guide to Microprocessor Hardware, 2000.

[29] M. A. Greene, "Pentium processor with MMX technology performance," in *Proc. IEEE Compcon*, 1997, pp. 263–267.

[30] Z. L. He and M. L. Liou, "A high performance fast search algorithm for block matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 101–111, Nov. 1997.

[31] P. Hsu and K. J. R. Liu, "Software optimization of H.263 video encoder on Pentium processor with MMX technology," in *Proc. IEEE Multimedia and Expo*, New York City, NY, Aug. 2000.

[32] A. Huttunen and I. Defee, "Performance of desktop software MPEG-2 TS decoder," in *Proc. 1999 IEEE Int. Symp. Circuits and Systems*, vol. 4, 1999, pp. 352–355.

[33] M. Lkekawa *et al.*, "A real-time software MPEG-2 decoder for multimedia PC's," in *Dig. Tech. Papers Int. Conf. Consumer Electronics ICCE*, 1997, pp. 2–3.

[34] Intel. (1999) Using MMX technology instructions to compute a 16-bit FIR filter. [Online]. Available: http://developer.intel.com/drg/mmx/AppNotes/ap559.htm

[35] D. Ishii, M. Ikekawa, and I. Kuroda, "Parallel variable length decoding with inverse quantization for software MPEG-2 decoders," in *Proc. IEEE Workshop Signal Processing Systems—Design and Implementation*, 1997, pp. 500–509.

[36] *Video Codec Test Model Near-Term Version 7 (TMN7)*, ITU-T Study Group 15, 1997.

[37] U. J. Kapasi *et al.*, "Efficient conditional operations for data-parallel architectures," in *Proc. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2000, pp. 159–170.

[38] P. S. Karthikeyan and P. S. Ranganathan, "More on arbitrary boundary arithmetic," in *Proc. IEEE 5th Int. Conf. High Performance Computing*, 1998, pp. 19–24.

[39] D. Kim and G. Choe, "AMD's 3DNow! vectorization for signal processing applications," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, vol. 4, 1999, pp. 2127–2130.

[40] L. Kohn *et al.*, "The visual instruction set (VIS) in UltraSPARC," in *Dig. Papers Compcon '96—Technologies for the Information Superhighway*, 1995, pp. 462–469.

[41] C. E. Kozyrakis and D. A. Patterson, "A new direction for computer architecture research," *IEEE Computer*, pp. 24–32, Nov. 1998.

[42] V. Lappalainen, "Performance analysis of Intel MMX technology for an H.263 video encoder ," in *Proc. 6th ACM Int. Multimedia Conf.*, Bristol, U.K., Sept. 1998.

[43] ——, "Performance of an advanced video codec on a general-purpose processor with media ISA extensions," *IEEE Trans. Consumer Electron.*, vol. 45, Aug. 2000.

[44] V. Lappalainen, A. Hallapuro, and T. D. Hämäläinen, "Optimization of emerging H.26L video encoder," in *Proc. IEEE Workshop Signal Processing Systems Design and Implementation*, Antwerp, Belgium, Apr. 2001.

[45] V. Lappalainen *et al.*, "Optimized implementations of emerging H.26L video decoder on Pentium III," in *Advances in Signal Processing and Computer Technologies*, G. Antoniou, N. Mastorakis, and O. Panfilov, Eds. Crete, Greece: WSES Press, July 2001, pp. 233–238.

[46] ——, "Low bit rate video coding on general-purpose processor," Dr.Technol. dissertation, Tampere Univ. of Technol., Tampere, Finland, 2001.

[47] S. Larsen and S. Amarasinghe, "Exploiting superword level parallelism with multimedia instruction sets," in *Proc. ACM SIGPLAN '00 Conf. Programming Language Design and Implementation*, June 2000, pp. 145–156.

[48] R. Lee, "Accelerating multimedia with enhanced microprocessors," *IEEE Micro*, vol. 15, pp. 22–32, Apr. 1995.

[49] ——, "Real-time MPEG video via software decompression on a PA-RISC processor," in *Dig. Papers IEEE Compcon—Technologies for the Information Superhighway*, 1995, pp. 186–192.

[50] ——, "Subword parallelism with MAX-2," *IEEE Micro*, vol. 16, pp. 51–59, Aug. 1996.

[51] R. Lee and J. Huck, "64-bit and multimedia extensions in the PA-RISC 2.0 architecture," in *Dig. Papers Compcon '96—Technologies for the Information Superhighway*, 1996, pp. 152–160.

[52] R. Lee, "Multimedia extensions for general-purpose processors," in *Proc. IEEE Workshop Signal Processing Systems—Design and Implementation*, 1997, pp. 9–23.

[53] R. Lee and L. McMahan, "Mapping of application software to the multimedia instructions of general-purpose microprocessors," in *Proc. SPIE*, vol. 3021, 1997, pp. 122–133.
[54] C. G. Lee and M. G. Stoodley, "Simple vector microprocessors for multimedia applications," in *Proc. IEEE/ACM Int. Symp. Microarchitecture*, Nov. 1998, pp. 25–36.
[55] R. B. Lee, "Efficiency of microSIMD architectures and index-mapped data for media processors," in *Proc. SPIE*, vol. 3655, 1999, pp. 34–46.
[56] ——, "Subword permutation instructions for two-dimensional multimedia processing in microSIMD architectures," in *Proc. IEEE Int. Conf. Application-Specific Systems, Architectures and Processors*, 2000, pp. 3–14.
[57] R. Leupers, "Code selection for media processors with SIMD instructions," in *Proc. ACM Conf. Design, Automation and Test in Europe*, Mar. 2000, pp. 4–8.
[58] C. Loeffler *et al.*, "Practical fast 1-D DCT algorithm with eleven multiplications," in *Proc. ICASSP '89*, 1989, pp. 988–991.
[59] J. McVeigh *et al.*, "A software-based real-time MPEG-2 video encoder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 1178–1184, Oct. 2000.
[60] MIPS Technologies. (1997) MIPS extension for digital media with 3D. [Online]. Available: http://www.mips.com
[61] T. Moriyoshi *et al.*, "Real-time software video codec with a fast adaptive motion vector search," in *Proc. IEEE Workshop Signal Processing Systems*, 1999, pp. 44–53.
[62] Z. J. A. Mou *et al.*, "VIS-based native video processing on UltraSPARC," in *Proc. IEEE Int. Conf. Image Processing*, vol. 2, 1996, pp. 153–156.
[63] E. Murata *et al.*, "Fast 2D IDCT implementation with multimedia instructions for a software MPEG2 decoder," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, vol. 5, 1998, pp. 3105–3108.
[64] H. Nguyen and L. K. John, "Exploiting SIMD parallelism in DSP and multimedia algorithms using the AltiVec technology," in *Proc. Int. Conf. Supercomputing*, 1999, pp. 11–20.
[65] S. Oberman *et al.*, "AMD 3DNow! technology: Architecture and implementations," *IEEE Micro*, vol. 19, pp. 37–48, Mar./Apr. 1999.
[66] H. Oehring *et al.*, "MPEG-2 video decompression on simultaneous multithreaded processors," in *Proc. IEEE/ACM Conf. Parallel Architectures and Compilation Techniques*, Newport Beach, CA, Oct. 1999.
[67] A. Peleg and U. Weiser, "MMX technology extension to the Intel architecture," *IEEE Micro*, vol. 16, pp. 42–50, Aug. 1996.
[68] A. Peleg, S. Wilkie, and U. Weiser, "Intel MMX for multimedia PC's," *Commun. ACM*, vol. 40, no. 1, pp. 25–38, Jan. 1997.
[69] P. Ranganathan *et al.*, "Performance of image and video processing with general-purpose processors and media ISA extensions," in *Proc. 26th Int. Symp. Computer Architecture*, 1999, pp. 124–135.
[70] K. P. Rao and P. Yip, *Discrete Cosine Transforms: Algorithms, Advantages, Applications*. New York: Academic, 1990.
[71] D. S. Rice, "High-performance image processing using special-purpose CPU instructions: The UltraSPARC visual instruction set," Master's thesis, Stanford Univ., Stanford, CA, 1996.
[72] J. E. Smith *et al.*, "Vector instruction set support for conditional operations," in *Proc. Int. Symp. Computer Architecture*, June 2000, pp. 260–269.
[73] D. Talla and L. K. John, "Execution characteristics of multimedia applications on a Pentium II Processor," in *Proc. 19th IEEE Int. Performance, Computing and Communications Conf.*, Feb. 2000, pp. 516–524.
[74] D. Talla *et al.*, "Evaluating signal processing and multimedia applications on SIMD, VLIW and superscalar architectures," in *Proc. IEEE Int. Conf. Computer Design*, Austin, TX, Sept. 2000.
[75] T. Thakkar and T. Huff, "The internet streaming SIMD extensions," *IEEE Computer*, pp. 26–34, Dec. 1999.
[76] R. Thekkath, "An architecture extension for efficient geometry processing," in *Proc. HOTCHIPS11*, Aug. 1999, pp. 263–274.
[77] M. Tremblay *et al.*, "VIS speeds new media processing," *IEEE Micro*, vol. 16, pp. 10–20, Aug. 1996.
[78] Y.-S. Tung *et al.*, "MMX-based DCT and MC algorithms for real-time pure software MPEG decoding," in *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, vol. 1, 1999, pp. 357–362.
[79] J.-F. Ueng *et al.*, "The design and performance analysis for the multimedia function unit of the NSC-98 CPU," in *Proc. IEEE Int. Conf. Information, Communications and Signal Processing ICICS*, vol. 3, 1997, pp. 1513–1517.
[80] J. Villalba *et al.*, "MMX-like architecture extension to support the rotation operation," in *Proc. IEEE Int. Conf. Multimedia and Expo ICME*, 2000, pp. 1383–1386.
[81] L. L. Winger, "Source adaptive software 2D IDCT with SIMD," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, vol. 6, 2000, pp. 3642–3645.
[82] S. Wong, S. Cotofana, and S. Vassiliadis, "Multimedia enhanced general-purpose processors," in *Proc. IEEE Int. Conf. Multimedia and Expo ICME*, vol. 3, 2000, pp. 1493–1496.
[83] ——, "Coarse reconfigurable multimedia unit extension," in *Proc. 9th Euromicro Workshop Parallel and Distributed Processing*, 2001, pp. 235–242.
[84] X. Yang *et al.*, "Fast subword permutation instructions based on butterfly networks," in *Proc. SPIE—Media Processors 2000*, vol. 3970, 2000, pp. 80–86.
[85] C.-G. Zhou *et al.*, "MPEG video decoding with the UltraSPARC visual instruction set," in *Dig. Papers Compcon '95—Technologies for the Information Superhighway*, 1995, pp. 470–477.
[86] D. F. Zucker, "Architecture and arithmetic for multimedia enhanced processors," Ph.D. dissertation, Dept. Elect. Eng., Stanford Univ., Stanford, CA, 1997.

Ville Lappalainen was born in Lempäälä, Finland, in 1974. He received the M.Sc. degree (with distinction) and the Dr.Technol. degree in computer science from Tampere University of Technology, Tampere, Finland, in 1997 and 2001, respectively.

In 1996, he joined Nokia Research Center, Tampere, Finland, where he is a Project Manager and Senior Research Engineer in the Media Processors Group. He has authored or coauthored about 15 scientific journal and conference articles. His current research interests are in the area of video coding, mainly in areas regarding architecture developments for efficient implementations of video coding algorithms.

Timo D. Hämäläinen received the M.Sc. degree (with distinction) in electrical engineering in 1993 and the Dr.Technol. degree in electrical engineering and computer science in 1997, both from Tampere University of Technology, Tampere, Finland.

He has been a Professor in the Institute of Digital and Computer Systems, Tampere University of Technology, since 2001, where he leads several academic and industrial research projects. His research interests are on parallel system-on-chip implementations for wireless multimedia systems, as well as automated mapping of algorithms on parallel platforms.

Petri Liuha received the M.Sc. degree in computer science from Tampere University of Technology, Tampere, Finland, in 1992.

From 1991 to 1992, he was an R&D Engineer with Nokia Consumer Electronics. In 1993, he joined Nokia Research Center, Tampere, Findland, where he has worked as a Research Engineer in different areas of implementation of video signal processing and multimedia. Currently, he is a Research Manager of the Media Processors Group. His current research interests are in architectural developments for implementations of multimedia applications.