A Performance Characterization of High Definition Digital Video Decoding using H.264/AVC

Mauricio Alvarez, Esther Salamí, Alex Ramirez, Mateo Valero
HiPEAC. European Network of Excellence on High-Performance Embedded Architecture and Compilation
Universitat Politecnica de Catalunya
Barcelona, Spain
{alvarez, esalami, aramirez, mateo}@ac.upc.edu

***Abstract*— H.264/AVC is a new international video coding standard that provides higher coding efficiency with respect to previous standards at the expense of a higher computational complexity. The complexity is even higher when H.264/AVC is used in applications with high bandwidth and high quality like high definition (HD) video decoding. In this paper, we analyze the computational requirements of H.264 decoder with a special emphasis in HD video and we compare it with previous standards and lower resolutions. The analysis was done with a SIMD optimized decoder using hardware performance monitoring. The main objective is to identify the application bottlenecks and to suggest the necessary support in the architecture for processing HD video efficiently. We have found that H.264/AVC decoding of HD video need to perform many more operations per frame that MPEG-4 and MPEG-2, has new kernels with more demanding memory access patterns and has a lot of coding options that can change inside a macroblock. In order to improve the H.264/AVC decoding process at HD it is necessary to explore a better support in media instructions, specialized prefetching techniques and possibly, the use of some kind of multiprocessor architecture.**

I. Introduction

In recent years, multimedia applications have become an important workload of computing systems. One of the most computational demanding tasks in the multimedia domain is video processing, specially due to its high information processing bandwidth and its real time requirements. Emerging applications like video over wireless channels and the increasing popularity of High Definition (HD) are requiring higher video quality and, at the same time, higher compression efficiency than the existing standards like MPEG-2 and MPEG-4 are capable to provide. For solving these issues a new international video standard called H.264/AVC has been defined which provides higher coding efficiency by using advanced compression techniques, that in turn, require more computational power. In applications with high bandwidth and high quality, like HD video, the computational requirements are even bigger [1].

The decoding of HD video using H.264/AVC is a big challenge for current processor architectures, specially in embedded systems where the processor not only needs to provide the required performance for real time operation but also to maintain a low power consumption and to allow low cost implementations. As a measure of the information processing requirements, Table I shows the bitrate for common applications of H.264/AVC. The high bandwidth for compressed bitstreams translates directly into high real time processing requirements.

Application	Resolution	Frame rate	Uncompressed bit rate	Compressed bit rate
HD-DVD	1920x1080	25	607 Mbps	8-20 Mbps
HDTV	1280x720	25	270 Mbps	2-8 Mbps
DVD	720x576	25	121 Mbps	1-2 Mbps
Video conferencing	352x288	25	30 Mbps	128 - 1000 kbps
Mobile video	176x144	15	9 Mbps	50 - 1000 Kbps

TABLE I
BITRATE REQUIREMENTS FOR VIDEO DECODING

In this paper, we analyze the computational requirements of H.264 decoder with a special emphasis in the processing of HD video and compare it with MPEG-2 and MPEG-4. The analysis was done over both the reference and an optimized decoder. The objective of this performance characterization is to identify the application bottlenecks and to suggest architecture support for processing H.264/AVC HD video efficiently. With such a performance characterization analysis the processing requirements for real time video decoding can be estimated and potential for optimization can be identified.

II. Overview of H.264/AVC Standard

H.264/AVC is based on the same block-based motion compensation and transform-based coding framework of prior MPEG video coding standards. It provides higher coding efficiency through added features and functionality that in turn entail additional complexity.

H.264/AVC includes a lot of new coding tools with different application scenarios. Here we present a summary of the most relevant ones for the performance of the decoder application. First, H.264/AVC introduces a variable block-size motion compensation with small block sizes that range from 16×16 to 4×4 pixels. Second, it is possible to use multiple reference frames for prediction with a weighted combination of the prediction signals. Third, fractional (1/2, 1/4) pixel precision is used in motion compensation. Fourth, the main transform is an integer DCT-like transform applied to 4×4 blocks, and in high definition profiles it is allowed to select between 4×4 and 8×8 transforms. Fifth, an adaptive deblocking filter was added in order to reduce the artifacts produced by the

block-based structure of the coding process. Sixth, an adaptive arithmetic coding technique (called CABAC) was developed for the entropy coding process. Finally in H.264/AVC different profiles has been defined that restrict the coding tools that can be used. The most relevant for HD are the main and high profiles.

In Table II the main features H.264/AVC are summarized and compared with MPEG-2 and MPEG-4 [2], [3]. One of the main difference between H.264 and the other video codecs is that it allows a variable block size which adapts better to motion changes in input content and MPEG-2 only support 16x16 pixel blocks and MPEG-4 16x16 down to 8x8 pixel blocks. Additionally H.264 uses a quarter sample resolution for motion estimation, a feature that is optional in MPEG-4 and it is not available in MPEG-2. Another important difference is that H.264 supports multiple reference frames for motion compensation compared to a single one in the other two codecs; and in the same way H.264 includes an in-loop deblocking filter that is not available in the previous standards. In general H.264 has more coding tools than the previous standards, and this is the main reason for the increase in the complexity of the codec. Next a description of the H.264/AVC coding/decoding process is presented.

In H.264/AVC all frames are either spatially or temporally predicted. Intra-macroblocks are spatially predicted and inter-macroblocks are temporally predicted using motion compensation. The prediction residual (difference between prediction and current data block) is encoded using transform coding. The transform is applied to 4×4 blocks using an integer like-DCT transform. The transform coefficients are then quantized and encoded using entropy coding methods. In the decoder these operations are performed in reverse order. First, video block is entropy decoded and the resultant coefficients are inverse quantized. An inverse transform is applied to those coefficients in order to obtain the error signal. If the macroblock was inter-predicted, motion compensation is applied using decoded reference frames that are stored in an image buffer. The resulting prediction block is added to the previously derived error signal to form a decoded macroblock. Finally the deblocking-blocking filter is applied to the macroblock and the resulting filtered macroblock is stored in the reference buffer and send it to the display memory.

III. Related work

Due to the fact that microprocessors are increasingly spending their cycles in multimedia applications, there has been a big effort on characterizing these applications. One of these studies was the design and characterization of the Mediabench benchmark which includes a MPEG-2 encoder and decoder with low resolution input sequences [4]. At that time the MPEG-4 and H.264/AVC video codecs were not available, for what most of the works based on Mediabench do not include results for these codecs. The Berkeley Multimedia Workload benchmark is based on Mediabench and for the video applications they use the same MPEG-2 codec but with input sequences at DVD, HD-720 (1280x720) and HD-1080 (1920x1080) resolutions [5]. This benchmark does not include the most recent video codecs either. This issue was partially solved by an extension to Mediabench in which the MPEG-4 and H.263 codecs were included [6]. But they do not address the issue of the increase in frame resolution either. Although these studies perform an analysis of the potential for optimization they do not include an analysis of the performance effect of using SIMD instructions.

On the other hand, in a study about the available parallelism in video applications the authors analyze the performance of the MPEG-1, MPEG-2, H.263 and MPEG-4 video codecs by simulating an idealistic machine with infinite resources [7]. They show that with such a machine it is possible to obtain speed-ups from 30X to 1000X compared to a reference implementation. They do not address the problem of the increase in image resolution, and do not use SIMD instructions. In a similar study the authors analyze several multimedia applications including MPEG-2 and H.263 video codecs and conclude that the variability observed in the execution of these applications comes from application properties, like I-P-B type of frames, not from the unpredictability introduced by cache memories and other architecture features of superscalar processors [8]. In a comparison of different multimedia instruction sets the authors use the MPEG-2 codec and analyze in detail the motion compensation kernels with different image resolutions, but the focus of this analysis is the comparison of media ISAs [9]. In another study the authors present an evaluation of some multimedia applications including the MPEG-2 codec but using low resolution sequences [10]. They analyze the impact in performance by using the VIS extension to the SPARC architecture.

There are some studies that deal with the cache performance of multimedia applications. One of them explored the cache behavior of various applications including the MPEG-2 codec [11]. The authors perform an experiment for evaluating the effect of cache capacity on miss ratio for different frame resolutions that shows that for determined size of the data cache the miss rate is not influenced by the frame resolution. Similarly a more recent study performs an analysis of cache performance for some Mediabench applications and compares their performance with SPECint and SPECfp benchmarks [12]. They conclude that multimedia applications have better cache behavior than SPEC benchmarks and argue that this is due to the block based structure of media applications.

There are some recent works that deal with the performance of the H.264 codec. Some of them perform a complexity analysis of H.264 with special attention on the video quality for low bitrate applications [13], [14]. They study the complexity of the H.264 decoder and conclude that H.264 is approximately 2.5 times more complex than H.263. In an another study the authors have developed a SIMD optimized version of the H.264 decoder using Intel SSE instructions and analyze the performance of the decoder for CIF and DVD resolutions [15]. In a different microarchitectural study of the H.264 reference decoder, the authors use low and standard video resolutions

Features	MPEG-2	MPEG-4 ASP	H.264/AVC Main	H.264/AVC High
Macroblock size	$16x16$	$16x16$	$16x16$	$16x16$
Block size	$8x8$	$16x16$,$16x8$,$8x8$	$16x16$,$16x8$,$8x16$, $8x8$,$4x8$,$8x4$,$4x4$	$16x16$,$16x8$,$8x16$, $8x8$,$4x8$,$8x4$,$4x4$
Transform	$8x8$	$8x8$ DCT	$4x4$ integer DCT	$8x8$,$4x4$ integer DCT
Pel-Accuracy	$1, 1/2$ pel	$1, 1/2, 1/4$ pel	$1, 1/2, 1/4$ pel	$1, 1/2, 1/4$ pel
Reference frames	One frame	One frame	Multiple frames (up to 5 frames)	Multiple frames (up to 5 frames)
Bidirectional prediction	forward/backward	forward/backward	forward/backward forward/forward backward/backward	forward/backward forward/forward forward/backward
deblocking filter	No	No	Yes	Yes
Weighted prediction	No	No	Yes	Yes
Custom Quantizer Matrix	No	No	No	Yes
Entropy Coding	VLC	VLC	CAVLC, CABAC	CAVLC, CABAC

TABLE II

COMPARISON OF VIDEO CODING STANDARDS

in order to analyze the availability of ILP by simulating a machine with infinite resources [16]. They suggest that the main bottleneck of the application is the unpredictable branch behavior. In an another performance characterization of MPEG-2 and H.264 decoders the authors develop a performance analysis of these codecs on the Pentium architecture in which they compare the differences between the kernels of H.264 and MPEG-2 for video at DVD resolution[17]. They conclude also that branch misprediction is a limiting factor for H.264 decoding due to the data dependent branches of some kernels.

The main difference of our study with respect to all the previously mentioned works is the performance characterization of H.264 at very high resolutions. Most of the published results are focused on low bitrate applications like mobile video or video conferencing; some of then explore applications with higher bandwidth like DVD playback but the performance requirements of HD has been not analyzed previously. Additionally our study performs a comparison of the performance of H.264 with other video codecs like MPEG-2 and MPEG-4.

IV. METHODOLOGY

The performance of a video decoding application depends on the algorithm, the input sequences and the architecture in which it is implemented [13]. For making a complete analysis we have selected various input sequences and different video codecs. The measurements have been done on a real machine using hardware performance monitoring.

A. Performance Monitoring

The experiments have been done on a PowerPC970 machine using the performance monitoring counters and the Apple CHUD tools. The experimentation platform is described in Table III. Hardware monitor counters have been used to collect profiling information, completed instructions, CPU cycles, cache accesses and misses and branch prediction. Additionally a time interval sampling analysis was conducted for analyzing the phase behavior of the program execution [18].

Processor	IBM PowerPC 970 1.6 GHz
Level 1 I-cache	64 KB
Level 1 D-cache	32 KB
Level 2 cache	512 KB
Main Memory	512 MB
System Bus	800 MHz
Operating System	Mac OS-X
Compiler	gcc 3.3.3
Compiler Optimizations	-O3, mtune=G5

TABLE III

EXPERIMENTATION PLATFORM

B. Codec Configuration

H.264/AVC has a lot of coding configuration parameters that can be varied depending on the target application and in turn they can have a great impact in the decoder performance. The selection of configuration parameters was done trying to reproduce the scenario of HD applications. The main profile of the JM-9.5 reference codec [19] was selected, and used with a constant quantization parameter and a I-P-B-B-P-B-B sequence of pictures. The baseline decoder is the JM-9.5 reference decoder with some platform independent optimizations for better memory performance and SIMD optimizations in some relevant kernels. The reference decoder is compared with the FFMPEG highly optimized H.264 decoder [20] and with the XviD MPEG-4 decoder [21] and the libmpeg2 MPEG-2 decoder [22]. The configuration parameters of all codecs were equally balanced in order to maintain a similar quality in the resulting videos.

C. Test Sequences

Due to the fact that our goal is to characterize the performance of the decoder at HD resolutions we selected input sequences for 720x576, 1280x720 and 1920x1088 pixel resolutions [23]. The set of input sequences has different motion characteristics in order to cover a broad range of video content. All of them have 100 frames with progressive scanning at 25 frames per second and use a 4:2:0 chroma subsampling format. The selected input sequences are described in Table IV with the average bit rate and quality (PSNR) for the three codecs under study. The efficiency of H.264 over the others codecs

Sequence	Resolution	H.264		MPEG-4		MPEG-2	
		bit rate Kb/s	PSNR dB	bit rate Kb/s	PSNR dB	bit rate kb/s	PSNR dB
576_blue_sky	720x576	2033	42.1	2236	41.3	6318	43.0
576_pedestrian	720x576	2372	43.5	2827	43.6	4562	44.1
576_riverbed	720x576	10863	40.6	15368	42.7	17295	42.2
576_rush_hour	720x576	1914	44.3	2136	44.0	3567	45.0
720_blue_sky	1280x720	3471	42.9	4050	42.4	10010	43.9
720_pedestrian	1280x720	4155	43.8	5199	44.1	8278	44.4
720_riverbed	1280x720	19057	41.2	27571	43.3	31848	42.7
720_rush_hour	1280x720	3363	44.6	3972	44.4	6248	45.2
1088_blue_sky	1920x1088	6724	43.3	8064	43.0	17723	44.2
1088_pedestrian	1920x1088	8132	43.4	11723	43.8	18424	44.2
1088_riverbed	1920x1088	34710	41.5	51508	43.4	60216	42.8
1088_rush_hour	1920x1088	6469	44.0	8900	43.9	14076	44.4

TABLE IV

INPUT TEST SEQUENCES AND CODEC COMPARISON

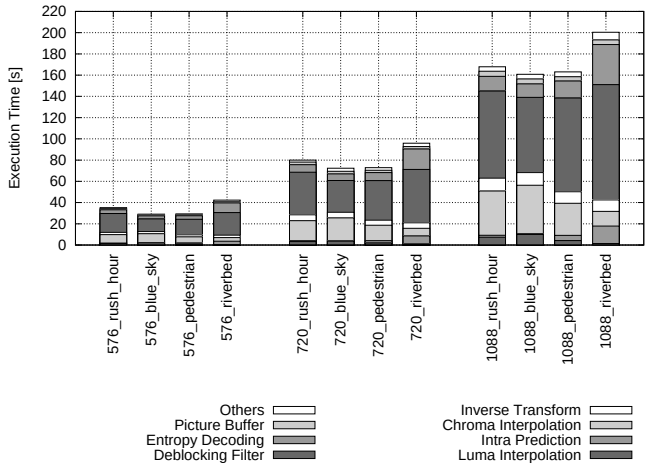

Fig. 1. Profiling of H.264 reference decoder with altivec optimizations.

is represented by its ability to provide a similar quality at a smaller bit rate.

V. ANALYSIS

By running the workloads on a real machine we have collected different kind of measures. First, we have performed a profiling, next we calculated average results per frame, and finally we have performed a sampling analysis that allows us to detect the presence of phases in the program execution.

A. Profiling of the H.264

In order to improve the performance of the reference decoder we have implemented some kernels using Altivec media instructions [24]. The selected kernels were the luma interpolation (part of the motion compensation process) and the inverse discrete transform. In Figure 1 the profiling of the H.264/AVC decoder with Altivec optimizations is shown. With the Altivec optimizations the total execution time has been reduced in 1.3X in average, and the luma interpolation kernel itself has a speed-up of 3.27X. After Altivec optimization the execution time is dominated by the deblocking filter (49.01%), the chroma interpolation (19.98%) and the entropy decoding (12.08%). It is important to note that there is more room for SIMD optimization, specially in the interpolation of chroma signals [15] but the execution time now is dominated by the deblocking filter and entropy decoding (61%).

1) Kernels of the H.264: The interpolation process is a computationally intensive kernel used for generating sub-sample positions. Half-sample positions are generated with a 6-tap FIR filter and, once these are available, quarter-sample positions are generated using linear interpolation. Chroma interpolation is done at eighth-sample resolution by using a linear interpolation from integer positions. Interpolation is a computationally intensive kernel but also it is memory demanding because half-sample interpolation requires six neighbor pixels in vertical, horizontal or diagonal directions inside the frame.

The deblocking filter is a very significant kernel because it is applied to each decoded macroblock. Filtering is applied to vertical or horizontal edges of 4×4 blocks, but the decision of filtering depends on the boundary strength and on the gradient of image samples across the boundary [25]. Both of these conditions are evaluated based on the values of the pixels at the edges and on some coding parameters like quantization parameter (QP). The filter itself is a FIR filter with different taps (3, 4 or 5) that are selected depending on the filtering conditions. As a result of that there are a lot of data dependent branches and a small data level parallelism in this kernel.

The inverse integer transform (IT) takes an average of 5.74% of the execution time, which is smaller than the Discrete Cosine Transform (DCT) in MPEG-2 that takes 29% [9]. The inverse transform (IT) kernel has a very low speed-up of 1.05X when implemented with Altivec SIMD instructions. That is due to the small DLP exhibited by the implementation of the algorithm in the reference code and the small quantity of data to be processed [26]. As a way to overcome these problems some implementations have reported bigger speed-ups using the original matrix multiplication algorithm [15].

H.264 uses a entropy coding technique called context adaptive binary arithmetic (CABAC) that achieves higher compression ratios than previous techniques by selecting different probability models for each syntax element, adapting the

Sequence	H.264-REF	H.264-HO	MPEG-4	MPEG-2
	fps	fps	fps	fps
576_rush_hour	2.84	27.04	66.80	338.98
576_blue_sky	3.43	30.68	58.21	299.40
576_pedestrian	3.41	31.04	78.55	158.98
576_riverbed	2.36	18.75	53.88	237.53
AVG 576	3.01	26.88	64.36	258.72
720_rush_hour	1.25	14.15	31.72	179.53
720_blue_sky	1.38	14.82	31.28	161.29
720_pedestrian	1.37	14.53	37.98	167.50
720_riverbed	1.04	9.30	27.09	80.84
AVG 720	1.26	13.20	32.02	147.29
1088_rush_hour	0.60	6.36	13.91	71.28
1088_blue_sky	0.62	6.73	13.98	72.10
1088_pedestrian	0.61	6.56	15.52	67.34
1088_riverbed	0.50	4.50	12.73	37.01
AVG 1088	0.58	6.04	14.04	61.93

TABLE V

DECODING PERFORMANCE IN FRAMES PER SECOND

Instruction Type	% of Class	% of Total
All Integer	100	50.5
Integer Add/Sub	49.7	25.1
Integer Multiply	4.4	2.2
Integer Divide	2.6	1.3
Integer Logical	17.3	8.7
Integer Shift	10.7	5.4
Integer Compare	14.2	7.2
All AltiVec	100	0.8
AltiVec Simple	30.0	0.3
AltiVec Complex	15.4	0.1
AltiVec Permute	54.6	0.4
All Load	100	24.7
All Store	100	7.5
All Branch	100	14.6
Miscellaneous	100	2.0

TABLE VI

DYNAMIC DISTRIBUTION OF INSTRUCTIONS H.264 REFERENCE DECODER

probability of each model based on local statistics and using arithmetic coding [25]. Due to the fact that each type of syntax element has different models the arithmetic decoder has to execute a lot of branches that are data dependent; and because of the serial nature of the arithmetic decoder it has to perform a lot of bit serial operations that restrict SIMD parallelization.

2) Decoding performance: Using the Altivec optimized version of the reference decoder (H.264-REF) we have calculated the decoding performance in frames per second which is shown in Table V. The table also presents a comparison with the FFMPEG highly optimized H.264 decoder (H.264-HO) with MPEG-2 and MPEG-4 decoders. For all the resolutions the MPEG-2 decoder can process the sequences in real time (25 frames per second), but for MPEG-4 this is not possible for the 1920x1088 resolution. For the H.264-REF decoder the performance is far from real time in all the resolutions, but it is necessary to take into account that the reference decoder defines the functionality of the H.264/AVC standard and it is not designed for high performance. Results from the H.264-HO decoder show that it is possible to decode sequences at DVD resolution in real time but for 1280x720 and 1920x1088 resolutions the obtained performance is half and a quarter of the necessary for real time respectively.

B. Instruction Breakdown

Using H.264-REF decoder we have performed an analysis of dynamic instruction distribution, which is shown in Table VI. H.264/AVC is dominated by integer operations, most of them are adds, subs and shifts. In the Altivec portion of the code there are more overhead instructions (permute 54%) than effective computation ones. That is very important for the luma interpolation kernel where there is a lot of instructions dedicated to data re-organization.

C. Instructions and Cycles

Table VII shows the average CPU cycles and instructions per frame for the different input sequences and different codecs under study. The H.264-REF decoder executes 10.5X and 66X more instructions in average than MPEG-4 and MPEG-2 respectively, and the H.264-HO decoder executes 1.36X and 8.61X more instructions respectively.

Also can be noted that the IPC changes with different input sequences but not with the increase in frame resolution. Taken that $IPC = (InstCount)/(Freq \times ExecTime)$ and with the information of instructions per frame and assuming a 3.0 GHz clock frequency it is possible to estimate the necessary IPC for decoding H.264/AVC in real time. The results are 2.99, 6.43 and 14.33 for the reference decoder at 720x576, 1280x720 and 1920x1088 resolutions respectively. For the H.264-HO decoder it is possible to reach the real time performance at 720x576, 1280x720 resolutions at 3.0 GHz, but for the 1920x1088 resolution it is necessary an IPC of 1.83. From these results we can note that even with a high performance next generation processor and highly optimized decoder it is not possible to process HD video in real time. Based on that we can conclude that it is necessary to explore multiple ways of parallelization apart from SIMD extensions in order to achieve the required performance for real time operation.

Similar to other video standards H.264/AVC has three different kind of frames: intra-coded (I-frames), inter-coded with unidirectional references (P-frames) and inter-coded with bi-directional references (B-frames). The impact of the different types of frames in performance is shown in Figure 2 with the distribution of cycles and instructions in I, P and B frames. The increase in cycles and instructions is proportional to the increase of the image area. The 1280x720 and 1920x1088 frame resolutions have an area that is 2.2 and 5 times bigger that the 720x576 resolution respectively. Also it can be noted that intraprediction takes more cycles and instructions than interprediction (P and B). Consequently for the sequences with a lot of I-macroblocks, like riverbed, the decoding time is bigger. In intra-prediction the samples of a macroblock are derived from macroblocks of the same image using different spatial modes which require quite complicated memory accesses.

D. IPC variability

A previous study [8] has demonstrated that most of the variability in the performance of frame based multimedia applications comes from the different kind of frames that exist

Sequence	H.264-REF			H.264-HO			MPEG-4			MPEG-2		
	Cycles $\times 10^6$	Inst. $\times 10^6$	IPC	Cycles $\times 10^6$	Inst. $\times 10^6$	IPC	Cycles $\times 10^6$	Inst. $\times 10^6$	IPC	Cycles $\times 10^6$	Inst. $\times 10^6$	IPC
576_rush_hour	438	337	0.77	42	43	1.03	32	34	1.05	5.6	4.0	0.71
576_blue_sky	426	336	0.79	40	42	1.05	34	37	1.07	6.6	4.9	0.74
576_pedestrian	429	330	0.77	40	40	1.02	28	27	0.97	5.8	4.2	0.73
576_riverbed	580	432	0.74	72	68	0.94	44	41	0.93	11.0	9.9	0.90
Average 576	468	359	0.77	48	48	1.01	34	34	1.00	7.3	5.8	0.77
720_rush_hour	945	729	0.77	88	90	1.03	70	74	1.05	11.0	8.0	0.73
720_blue_sky	913	726	0.80	82	86	1.05	71	77	1.07	11.9	9.0	0.76
720_pedestrian	925	714	0.77	83	86	1.04	60	59	0.98	11.5	8.6	0.75
720_riverbed	1239	917	0.74	144	139	0.96	91	85	0.93	21.6	19.8	0.92
Average 720	1005	771	0.77	99	100	1.02	73	73	1.01	14.0	11.4	0.79
1088_rush_hour	2128	1632	0.77	192	198	1.03	160	165	1.03	26.5	19.2	0.72
1088_blue_sky	2060	1631	0.79	181	191	1.05	160	168	1.05	25.9	19.1	0.74
1088_pedestrian	2094	1613	0.77	185	196	1.06	146	144	0.99	27.5	20.7	0.75
1088_riverbed	2717	2004	0.74	295	295	1.00	194	182	0.94	47.4	41.1	0.87
Average 1088	2250	1720	0.77	213	220	1.04	165	165	1.00	31.8	25.0	0.77

TABLE VII
CYCLES, INSTRUCTIONS AND IPC PER FRAME

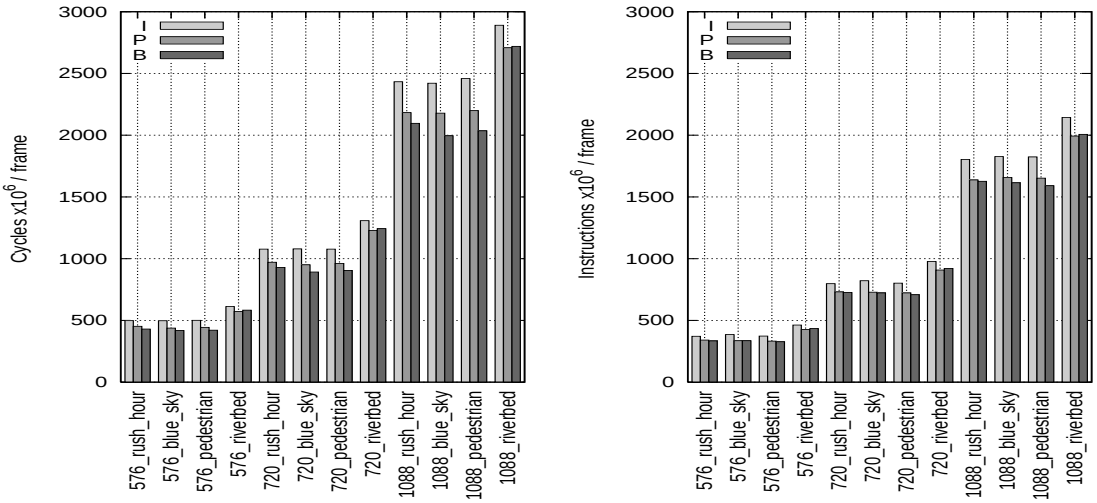

Fig. 2. Cycles and instructions per frame H.264-REF decoder

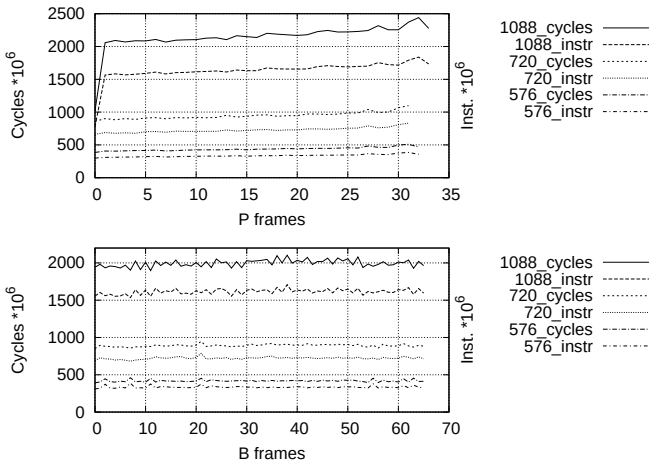

Fig. 3. Cycles and instructions in P and B frames for blue_sky

in the application (ie I,P,B). Figure 3 shows the instructions and cycles classified by P and B frames for the blue_sky sequence. Instructions and cycles exhibit a small variation between frames of the same type, and because of that the IPC remains almost constant. Based on that we can conclude that the amount of computational work necessary to decode each frame depends on the frame type (P and B) and on the density of I-macroblocks in each frame.

E. IPC sampling

In addition, we have made an analysis of the decoding of the whole sequence and we found that the application exhibits a phase behavior at the granularity of a P-B-B sequence. Figure 4 shows the time behavior for the decoding of a P-B-B group of frames for the 1088_blue_sky sequence. The three frames are separated by big peaks and fluctuations of IPC which corresponds to the copy of the decoded frame out of the frame buffer. In turn, the processing of each frame has three clearly differentiated stages: the first, which includes the

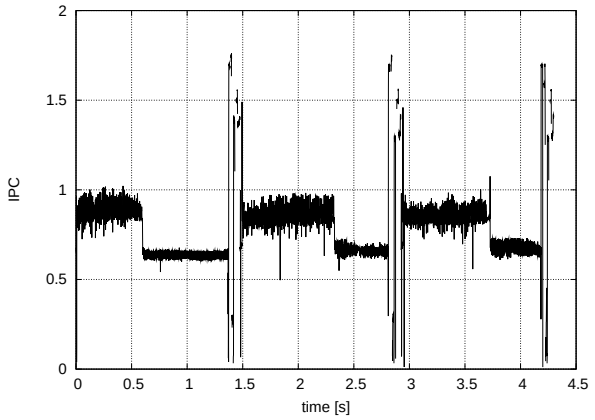

Fig. 4. Sampling of IPC for 1088_blue_sky

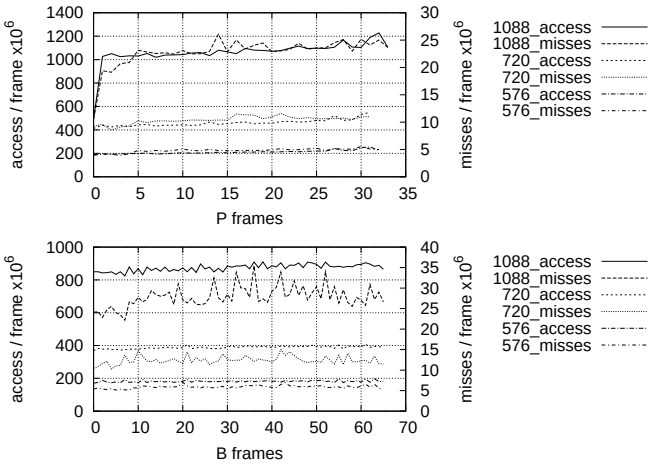

Fig. 6. dL1 accesses and misses in P frames for blue_sky

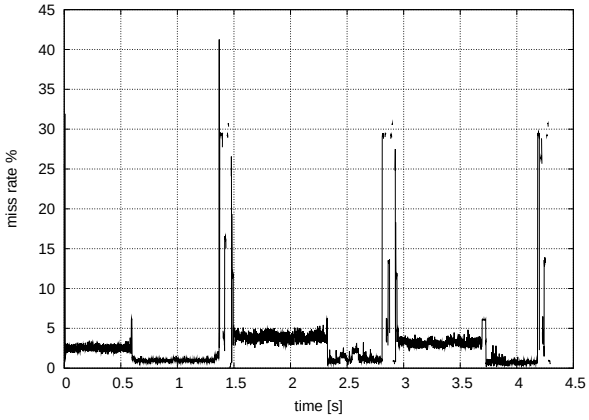

Fig. 7. Sampling of d-L1 miss rate for 1088_blue_sky

entropy decoding, inverse transform and motion compensation, has an IPC close to one; the second, which consist of the deblocking filter, has a lower IPC near 0.6; and the third phase the IPC exhibits big fluctuations due to the ”memcopy” kind of operations. The behavior is similar for both P-and-B frames (I frames are not analyzed) and in B frames the motion compensation stage is longer. For space reasons we only show the results for 1920x1088 resolution but for the other resolutions the figure is very similar with the only difference of time scale.

F. Cache Analysis

Table VIII shows the average number of accesses and misses per frame for the L1 data cache. H.264/AVC decoder has many more L1 data cache accesses and misses than the other two codecs; for example in the 1920x1088 resolution the H.264-REF decoder performs 15.2X and 79.06X more memory accesses than MPEG-4 and MPEG-2 respectively and the H.264-HO decoder performs 1.29X and 6.6X more memory accesses.

Although H.264/AVC performs more memory accesses per frame it has a smaller miss rate. This good cache behavior is due to the fact that H.264/AVC perform more operations per frame than the other codes but it has a high data locality at the macroblock level in which most of the operations are done, and macroblocks fits well into the data cache, even where the whole frame will not. Additionally the miss rate remains almost equal with the resolution, only in MPEG-4 there is an increment in miss rate with resolution. These results are in consonance with some previous studies on memory behavior for multimedia applications [11], [12] that claim that the use of cache memories benefits in a significant way the performance of these applications.

Figure 5 shows the distribution of L1 data cache accesses and misses for the different type of frames in the H.264/AVC sequences. In I-frames the decoder performs more accesses to the L1 data cache than the other type of frames, mainly because intra-prediction uses several spatial prediction modes with different memory access patterns. On the contrary B frames have more data cache misses, and that is because these kind of frames have to access multiple reference frames, that are stored in a picture buffer, which does not fit in the L1 level cache.

Figure 6 shows the variation of accesses and misses in P and B frames respectively. In the P-frames L1 data cache accesses and misses remain almost constant, then the miss rate is not affected by the resolution and input content. But in B-frames the misses exhibit big variations for the 1920x1088 resolution (more than 40% of the total misses) that are not present in lower resolutions. This result shows that for HD resolution the bidirectional prediction puts a big pressure on the data cache. This behavior can be more severe if the encoder uses more than two frames for the predictions of the current frame, which is a technique available in H.264/AVC.

Figure 7 shows the L1 data cache time behavior for the decoding of a P-B-B sequence of the 1088_blue_sky sequence. As with the time behavior for IPC shown in Figure 4 there are three different phases of the execution. The first phase, in which motion compensation is performed, has a bigger miss rate than the second phase in which deblock filtering is applied. Motion compensation exhibits a bigger miss rate that

Sequence	H.264-REF			H.264-HO			MPEG-4			MPEG-2		
	Accesses $\times 10^6$	Misses $\times 10^6$	Miss rate	Accesses $\times 10^6$	Misses $\times 10^6$	Miss rate	Accesses $\times 10^6$	Misses $\times 10^6$	Miss rate	Accesses $\times 10^6$	Misses $\times 10^6$	Miss rate
576_rush_hour	204	5.0	2.4	18	0.52	2.9	15	0.54	3.6	2.2	0.15	6.8
576_blue_sky	192	5.1	2.7	18	0.51	2.9	16	0.46	2.9	2.3	0.16	6.8
576_pedestrian	205	4.8	2.4	17	0.44	2.6	13	0.54	4.2	2.3	0.16	6.9
576_riverbed	299	5.0	1.7	35	0.35	1.0	15	0.82	5.5	3.6	0.22	6.1
Average 576	225	5.0	2.3	22	0.45	2.3	15	0.59	4.1	2.6	0.17	6.6
720_rush_hour	442	10.5	2.4	37	1.04	2.9	34	1.46	4.3	5.2	0.30	5.8
720_blue_sky	415	10.7	2.6	36	1.17	3.3	35	1.39	4.0	5.4	0.33	6.0
720_pedestrian	446	10.2	2.3	35	0.91	2.6	29	1.47	5.1	5.5	0.33	6.0
720_riverbed	646	10.3	1.6	69	1.22	1.8	31	1.79	5.7	7.8	0.47	6.1
Average 720	487	10.4	2.2	44	1.09	2.6	32	1.53	4.8	6.0	0.36	6.0
1088_rush_hour	1002	24.1	2.4	79	2.24	2.8	77	4.95	6.4	12.6	0.79	6.2
1088_blue_sky	948	24.8	2.6	79	3.22	4.1	78	5.04	6.5	12.4	0.80	6.4
1088_pedestrian	1015	23.3	2.3	78	2.07	2.7	68	4.55	6.7	13.1	0.82	6.2
1088_riverbed	1431	22.6	1.6	139	2.42	1.7	67	4.09	6.1	17.6	1.11	6.3
Average 1088	1099	23.7	2.2	93	2.48	2.8	72	4.66	6.4	13.9	0.88	6.3

TABLE VIII

D-L1 ACCESSES, MISSES AND MISS RATE COMPARISON

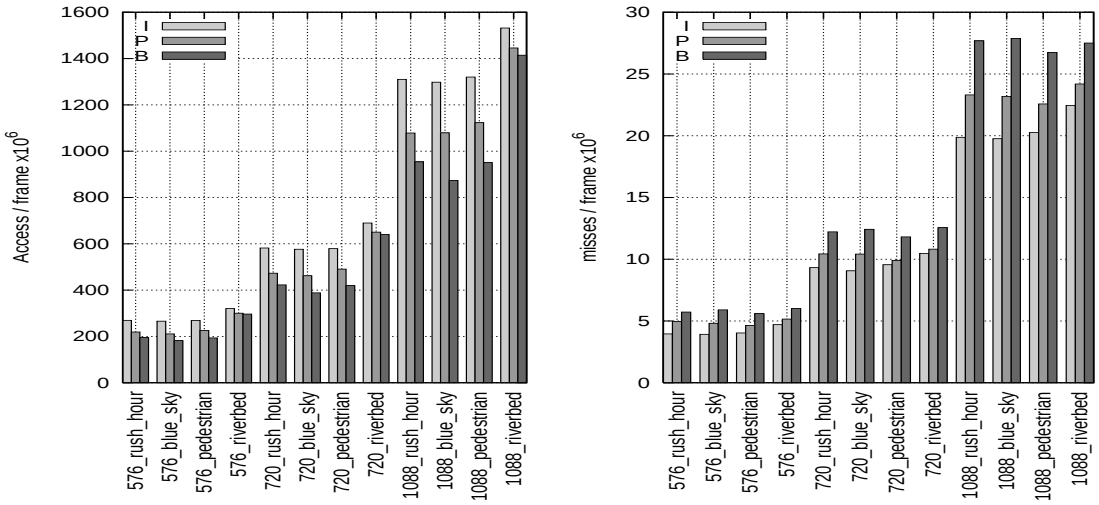

Fig. 5. L1 accesses and misses per frame

is related to the inter-prediction decoding process in which a reference frame is used to predict the current frame. The time of this phase is bigger in the B frames in which there are more than one reference frame. The peaks are related again with the third phase in which the decoded frame is sent out of the decoded picture buffer.

The access to multiple reference frames in interprediction seems to be the main factor for the cache miss rate. For solving that some prefetching mechanism could be used. In this case software prefetching could help to reduce significantly the cache miss rate because the algorithm knows what frames (and macroblocks in that frame) are going to be used as reference.

G. Branch Prediction

H.264/AVC has a lot of different coding options that can change from macroblock to macroblock. These options result in a high density of branches and branch mispredictions as can be seen in Table IX. The reference decoder executes 4X and 119X more branches compared to MPEG-4 and MPEG-2 respectively and the HO decoder executes 1.67X and 15X respectively. In all the codecs branch prediction exhibits a big variation between different sequences and also but does not exhibit considerable changes with frame resolution. The H.264-HO decoder has a bigger branch misprediction rate than the H.264-REF decoder (2X for the 1920x1088 resolution). This come from the fact that the reference decoder has support for more (exotic) coding options than the H.264-HO decoder, and these options need to be checked in each each frame (sometimes each macroblock) but are easier to predict.

In Figure 8 the branch misprediction sampling is shown for a P-B-B sequence of the 1088_blue_sky sequence. The three phases scheme is again evident, and in this case the entropy decoding and motion compensation stage exhibits a bigger misprediction rate than filtering. CABAC entropy decoding has a lot of data dependent branches that are difficult to predict and also motion compensation has a lot of options that can change inside each macroblock (block size, pixel interpolation)

Sequence	H.264-REF			H.264-HO			MPEG-4			MPEG-2		
	Branches $\times 10^6$	Mispred. $\times 10^6$	Mispred. rate	Branches $\times 10^6$	Mispred. $\times 10^6$	Mispred. rate	Branches $\times 10^6$	Mispred. $\times 10^6$	Mispred. rate	Branches $\times 10^6$	Mispred. $\times 10^6$	Mispred. rate
576_rush_hour	54	2.54	4.7	6.4	0.63	9.8	3.4	0.12	3.5	0.39	0.04	11.2
576_blue_sky	53	2.51	4.7	5.5	0.57	10.2	3.5	0.11	3.2	0.52	0.06	12.0
576_pedestrian	54	2.43	4.5	6.5	0.61	9.4	3.3	0.14	4.4	0.43	0.05	12.1
576_riverbed	79	3.64	4.6	11.3	1.48	13.1	7.2	0.54	7.6	1.14	0.14	12.4
Average 576	60	2.78	4.63	7.4	0.82	10.7	4.3	0.23	4.68	0.62	0.07	11.90
720_rush_hour	117	5.41	4.6	13.6	1.22	8.9	7.3	0.24	3.3	0.71	0.07	10.3
720_blue_sky	115	5.35	4.7	11.6	1.11	9.5	7.4	0.23	3.1	0.90	0.10	11.7
720_pedestrian	115	5.14	4.5	13.8	1.18	8.6	7.1	0.29	4.1	0.82	0.09	11.5
720_riverbed	165	7.51	4.5	23.3	2.86	12.3	14.8	1.10	7.4	2.15	0.26	12.2
Average 720	128	5.85	4.57	15.6	1.59	9.8	9.2	0.46	4.47	1.14	0.13	11.44
1088_rush_hour	261	11.89	4.6	30.5	2.52	8.3	16.9	0.58	3.4	1.64	0.17	10.3
1088_blue_sky	258	11.9	4.6	26.5	2.34	8.8	16.2	0.49	3.1	1.78	0.20	11.5
1088_pedestrian	260	11.6	4.5	31.1	2.51	8.1	17.4	0.72	4.1	1.87	0.21	11.4
1088_riverbed	357	15.96	4.5	50.0	5.71	11.4	32.0	2.28	7.1	4.25	0.51	12.0
Average 1088	284	12.84	4.52	34.5	3.27	9.1	20.6	1.02	4.44	2.38	0.27	11.29

TABLE IX
BRANCHES AND BRANCH MISPREDICTION

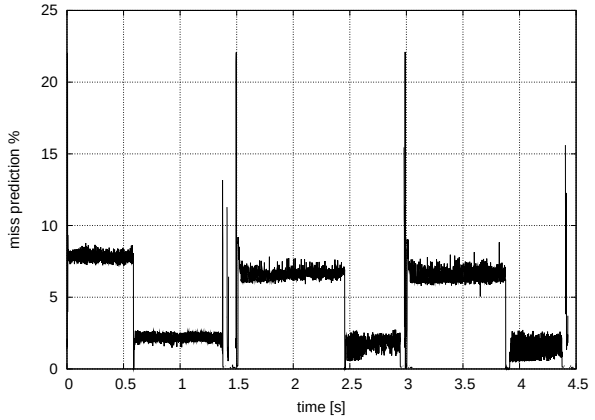

Fig. 8. Branch misprediction

and these values are selected by the encoder according to the content of the input sequence also making them difficult to predict. As the problems with the branch prediction come from the values of the data and not from the internal flow of the code, it is difficult to design a branch predictor that predicts these values. For this case it could be beneficial to use of techniques like total or partial predication.

VI. CONCLUSIONS AND FUTURE WORK

We have evaluated the performance of the H.264/AVC video decoder with an emphasis on HD resolution. Using profiling techniques we identified the most time demanding kernels and some of them were optimized using SIMD extensions. Two H264/AVC decoders was compared with previous video codecs, and the decoding of HD sequences was compared with lower resolutions. We measured and presented statistics such as IPC, branch prediction and cache statistics. Additionally we performed sampling analysis of key performance metrics in order to analyze the variability and presence of phases in the execution.

The profiling analysis has shown that the H.264/AVC has new kernels, like pixel interpolation and deblocking filter, that have different computational requirements and performance behavior than the kernels of previous video standards. Conversely some kernels that are more important in former MPEG codecs like the IDCT have a little impact on the performance of the H.264/AVC decoding. In addition the proportion of execution time of each kernel changes severely with the coding parameters. In this study we select the coding parameters for high bitrate applications and they are somewhat different to other published results that address low bitrate applications. Specifically in our study the deblocking filter is the most time consuming kernel, followed by pixel interpolation.

Although with the SIMD optimization of some kernels it is possible to speed-up the application, the dynamic instruction breakdown shows that most of media instructions are devoted to data transformation and reorganization, that suggest that there is a mismatch between the data structures of kernels and the register structure of the multimedia extension.

In addition we have quantified the complexity of the H.264/AVC decoding process by measuring the instructions and cycles that are necessary for decoding a frame. The comparison with MPEG-4 and MPEG-2 shows that H264/AVC requires many more operations than these codecs. This enormous amount of data that needs to be processed makes harder the real time operation at HD resolutions. Although it could be possible to provide this processing requirements with some new generation of high frequency high performance superscalar processors, this choice is not suitable for embedded systems that have strict power and cost constraints. For this application scenario there is an open research field for finding processor architectures that meet the high performance processing requirements and at the same time maintain a low power consumption. Our future work goes in that direction.

On the other hand we have confirmed the results from previous works that the variability in performance is mainly due to the different type of frames. For H.264/AVC this behavior is stronger because I, P and B frames have very different coding tools. For instance I frames performs more accesses per frame than the other ones but B frames have

more misses per frame. This behavior suggests the possibility of including different architectural support for each type of frames. In B frames it seems to be more useful to include some type of macroblock prefetching to reduce the miss rate, but for I-frames it would be better a special type of load that facilitates the different memory patterns of the spatial intraprediction process.

With respect to memory behavior we have found that, although H.264/AVC performs many more memory accesses per frame than the other codecs, it has a higher cache hit rate. This result suggest that H.264/AVC performs more operations per frame but these additional operations are done over small units, like macroblocks or sub-macroblocks, then the increment in memory accesses does not means an increment in cache miss ratio.

We also found that the H.264/AVC decoding process has differentiate phases of execution which correspond to the P-B-B sequence of frames. Additionally we have found that inside each frame there are well defined execution phases that are directly related with key functional sections of the decoder. Our study reveals that the motion compensation process and the deblocking filter are the most dominant and differentiated stages. Taken that into account it is possible to suggest different architecture support for each one of these execution phases.

Our fundamental conclusion is that H.264/AVC decoding of HD video is a big challenge for current processor architectures, because it presents a tremendous amount of data that needs to be processed in real time but not as regular as it has been with other video codecs. H.264/AVC has new kernels with more demanding memory access patterns and has a lot of coding options that can change inside each macroblock, then the H.264/AVC decoding of HD video will require not only more performance for the multimedia instruction sets of one processor, but possibly, the use of some kind of multiprocessing approach.

Currently we are working on the development of extensions to the current multimedia ISAs for better support of the new features of the video processing with H.264 and also we are analyzing the memory and multiprocessor organizations that can provide the required performance under low power design constraints.

ACKNOWLEDGMENT

This work has been supported by the Ministry of Science and Technology of Spain, the European Union (FEDER funds) under contract TIC2004-07739-C02-01 and by IBM. We acknowledge the Barcelona Supercomputing Center (BSC) for supplying the computing resources for our research.

REFERENCES

[1] T. Sikora, "Trends and Perspectives in Image and Video Coding," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 6–17, Jan 2005.

[2] A. Tamhankar and K. R. Rao, "An Overview of H.264/MPEG-4 PART 10," in *4th EURASIP Conference focused on Video/Image Processing and Multimedia Communications*, July 2003, pp. 1–51.

[3] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video Coding with H.264/AVC: Tools, Performance, and Complexity," *IEEE Circuits and Systems Magazine*, vol. 4, no. 1, pp. 7–28, Jan 2004.

[4] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "Mediabench: A Tool for Evaluating and Synthesizing Multimedia and Communicatons Systems," in *30th International Symposium on Microarchitecture*, 1997, pp. 330–335. [Online]. Available: citeseer.nj.nec.com/lee97mediabench.html

[5] N. T. Slingerland and A. J. Smith, "Design and Characterization of the Berkeley Multimedia Workload," *Multimedia Systems*, vol. 8, no. 4, pp. 315–327, 2002.

[6] J. Fritts, W. Wolf, and B. Liu, "Understanding Multimedia Application Characteristics for Designing Programmable Media Processors," in *SPIE Photonics West, Media Processors '99*, 1999, pp. 2–13.

[7] H. Liao and A. Wolfe, "Available Parallelism in Video Applications," in *International Symposium on Microarchitecture*, 1997, pp. 321–329.

[8] C. J. Hughes, P. Kaul, S. V. Adve, R. Jain, C. Park, and J. Srinivasan, "Variability in the execution of multimedia applications and implications for architecture," in *Proceedings of the 28th Annual International Symposium on Computer Architecture*, 2001, pp. 254–265.

[9] N. Slingerland and A. J. Smith, "Measuring the Performance of Multimedia Instruction Sets," *IEEE Transactions on Computers*, vol. 51, no. 11, pp. 1317–1332, Nov 2002.

[10] P. Ranganathan, S. V. Adve, and N. P. Jouppi, "Performance of Image and Video Processing with General-Purpose Processors and Media ISA Extensions," in *International Symposium on Computer Architecture*, 1999, pp. 124–135.

[11] N. T. Slingerland and A. J. Smith, "Cache Performance for Multimedia Applications," in *ICS '01: Proceedings of the 15th International Conference On Supercomputing*, 2001, pp. 204–217.

[12] Z. Xu, S. Sohoni, R. Min, and Y. Hu, "An Analysis of Cache Performance of Multimedia Aplications," *IEEE Transactions on Computers*, vol. 53, no. 1, pp. 20–38, Jan 2004.

[13] M. Horowitz, A. Joch, and F. Kossentini, "H.264/AVC Baseline Profile Decoder Complexity Analyis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 704–716, July 2003.

[14] V. Lappalainen, A. Hallapuro, and T. D. Hamalainen, "Complexity of Optimized H.26L Video Decoder Implementation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 717–725, July 2003.

[15] X. Zhou, E. Q. Li, and Y.-K. Chen, "Implementation of H.264 Decoder on General-Purpose processors with Media Instructions," in *Proceedings of SPIE Conference on Image and Video Communications and Processing 2003*, 2003.

[16] T. T. Shih, C. L. Yang, and Y. S. Tung, "Workload Characterization of the H.264/AVC Decoder," in *Proceeding of the 5th Pacific Rim Conference on Multimedia*, December 2004.

[17] M. Holliman and Y.-K. Chen, "Mpeg Decoding Workload Characterization," in *Proceedings of Workshop on Computer Architecture Evaluation Using Commercial Workloads*, Feb 2003.

[18] F. E. Levine and C. P. Roth, "A Programmer's View of Performance Monitoring in the PowerPC Microprocessor," *IBM Journal of Research and Development*, vol. 41, no. 3, p. 345, 1997.

[19] "H.264/AVC Software Coordination," 2005, http://iphome.hhi.de/suehring/tml/.

[20] "Ffmpeg multimedia system. a solution to record, convert and stream audio and video," 2005, http://ffmpeg.sourceforge.net.

[21] "Xvid. An ISO MPEG-4 Compliant Video Codec," 2005, http://www.xvid.org.

[22] "Libmpeg2 - a Free MPEG-2 Video Stream Decoder," 2005, http://libmpeg2.sourceforge.net/.

[23] "Mpeg-test sequences." 2005, http://www.ldv.ei.tum.de/liquid.php?page=70.

[24] K. Diefendorff, P. Dubey, R. Hochsprung, and H. Scales, "Altivec Extension to PowerPC Accelerates Media Processing," *IEEE Micro*, vol. 20, no. 2, pp. 85–95, April 2000.

[25] I. Richardson, *H.264 and MPEG-4. Video Compression for Next-generation Multimedia*. Chichester, England: Wiley, 2004.

[26] H. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity Transform and Quantization in H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 598–603, July 2003.