The H.264/MPEG4 advanced video coding

Article *in* Proceedings of SPIE - The International Society for Optical Engineering · June 2009

DOI: 10.1117/12.838234

Artur Gromek
Warsaw University of Technology
37 PUBLICATIONS **109** CITATIONS

The H.264/MPEG4 – Advanced Video Coding

Artur Gromek[a]

[a]Warsaw University of Technology, Institute of Electronic Systems, Nowowiejska 15/19, 00-650 Warsaw, Poland phone: (+48 22) 6607478, fax: (+48 22) 8252300 e-mail: agromek@elka.pw.edu.pl

ABSTRACT

H.264/MPEG4-AVC is the newest video coding standard recommended by International Telecommunication Union – Telecommunication Standardization Section (ITU-T) and the ISO/IEC Moving Picture Expert Group (MPEG). The H.264/MPEG4-AVC has recently become leading standard for generic audiovisual services, since deployment for digital television. Nowadays is commonly used in wide range of video application ranging like mobile services, videoconferencing, IPTV, HDTV, video storage and many more. In this article, author briefly describes the technology applied in the H.264/MPEG4-AVC video coding standard, the way of real-time implementation and the way of future development.

1. INTRODUCTION

The H.264/MPEG4 Advanced Video Coding (AVC) [1, 2] as the newest international digital video coding standard has demonstrated significant improvement of coding efficiency, substantially enhanced error robustness, and increased flexibility and scope of applicability relative to its predecessors [3]. The H.264/MPEG4-AVC specifies a set of "profiles and levels" that provides interoperability between encoder and decoder within various applications of the standard. A recently announced amendment to the H.264/MPEG4-AVC presents the fidelity range extension (FRExt), further broaden new standard towards to professional contribution, distribution, or studio/post production. Scalable video coding (SVC) allowing reconstruction of video signals with lower spatio-temporal resolution/ quality from coded video representation parts (i.e. from partial bitstreams). Another set of extension for multi-view video coding (MVC) is currently being designed. This article provides high level overview of the so-called video coding layer (VLC) of the H.264/MPEG4-AVC, complemented by network abstraction layer (NAL), providing header information in a manner appropriate for conveyance by a variety of transport layers and/or storage media. Moreover, this paper is intended to serve as a starting point for more efficient codec implementations to meet performance requirements of real time D1 resolution { NTSC: 720x480@30fps or PAL: 720x576@25fps } processing.

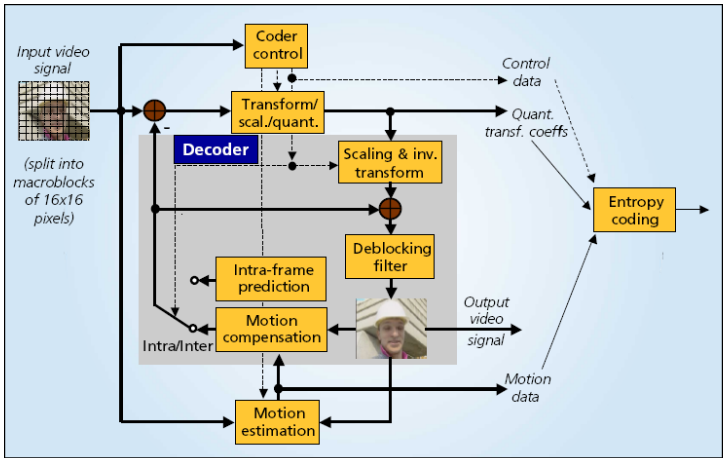


Figure 1. Typical H.264/MPEG4-AVC video encoder structure

1.1. Profiles and Levels

Profiles and levels specify the conformance points, designed to facilitate interoperability in various applications of the H.264/AVC standard that have similar functional requirements. A profiles defines a set of syntax features for use in generating compliant bitstreams, whereas a level places constraints of certain key parameters of the bitstream, i.e. maximum bit rate, maximum picture size etc. All decoders conforming to a specific profile have to support all features in that profile. Encoders are not required to make use of any particular set of features supported in a profile but have to provide conforming bitstreams. In the H.264/AVC, six profiles are defined, the major ones are - Baseline, Main, eXtended and High [1, 2, 4].

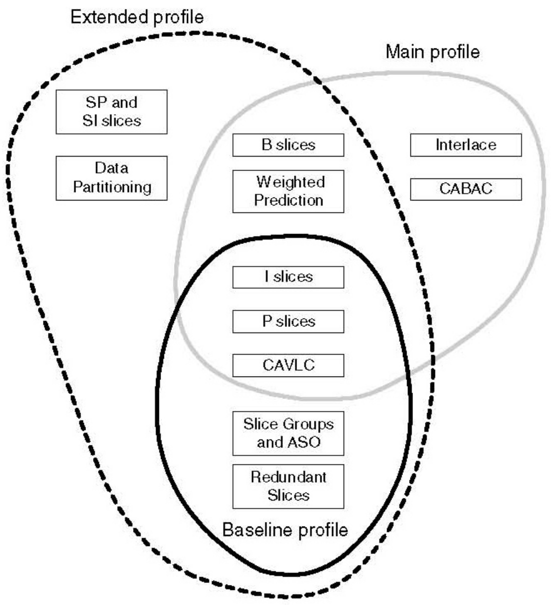


Figure 2. The H.264/MPEG4-AVC Profiles

1.2. Frame and Prediction Types

The H.264/AVC is based on the concepts of previous standards i.e. MPEG2, MPEG4 Visual, however introduce plenty of new features hence offers potentially better compression efficiency. Standard distinguish three types of video frames, considered as two interleaved fields – Intra (I in some cases also Key Frame), Inter (P) and Bi-frame (B) – Fig.3.

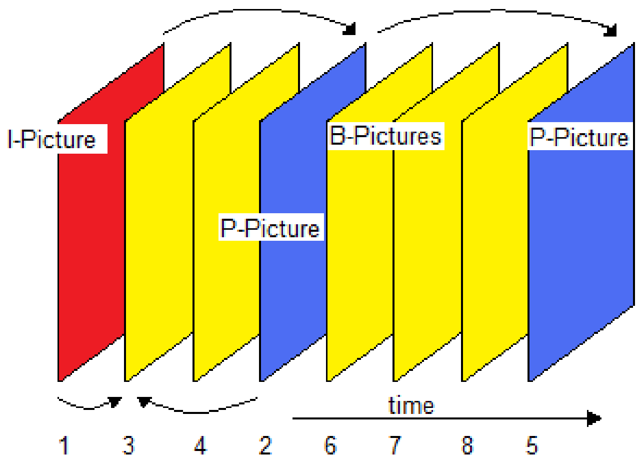
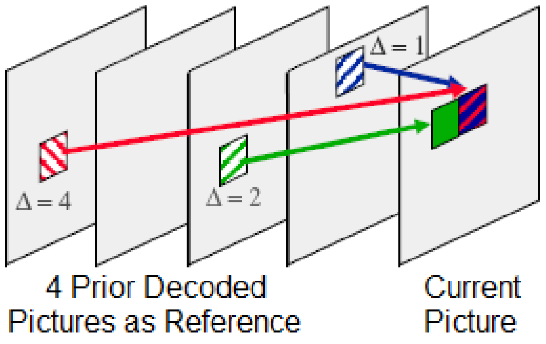


Figure 3. Video sequence frame types and their encoding order

Each video frame is processed in fixed size units of a macroblock (MB: 16x16 pixels). Encoder forms a macroblock prediction from previously coded data (inter prediction) – Fig.4a, and also from the current frame (intra prediction) – Fig.4b. The encoder subtracts the prediction from the current macroblock to form residual[1] for both the luma and chroma components. The macroblocks are organized in slices, which represent regions of given picture that can be processed independently of each other. The macroblock prediction methods/modes supported by the H.264/AVC are more flexible than those in the previous standards. Intra prediction uses 16x16, 8x8 and 4x4 block sizes to predict the macroblock. Inter prediction uses a range of block sizes from 16x16 to 4x4. The residual macroblock of the prediction (either Intra or Inter) and original input samples is than transformed.



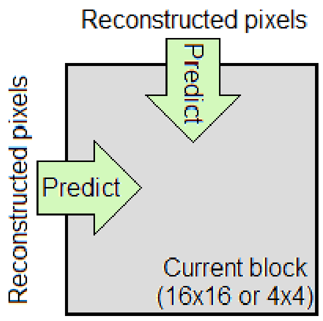


Figure 4. a) Inter prediction, b) Intra prediction

1.3. Transform and Quantization

As already be denoted, macroblock prediction residual are transformed to produce set of coefficients. The H.264/AVC standard defines two integer transforms, 4x4/8x8 discreet cosine transform (DCT) and 2x2/4x4 discreet Haar/Haddamard transform (DHT). An additional DHT transforms is only used for all resulting DC coefficients in the case of using luma 16x16 intra coding type as well as for both chroma components in all cases. The output block of transform coefficients is quantized, i.e. each coefficient is divided by an integer value. In that way, quantization procedure reduces the precision of the transform coefficients according to quantization parameter (QP). As a result we obtain block of data with only a few non-zero coefficients typically. For high QP values, more coefficients are cut down (set to zero) hence higher compression rate is present in expense to poor video quality and vice versa. The quantized transform coefficients of a block are scanned in a “zig-zag” manner and further processed by entropy coder.

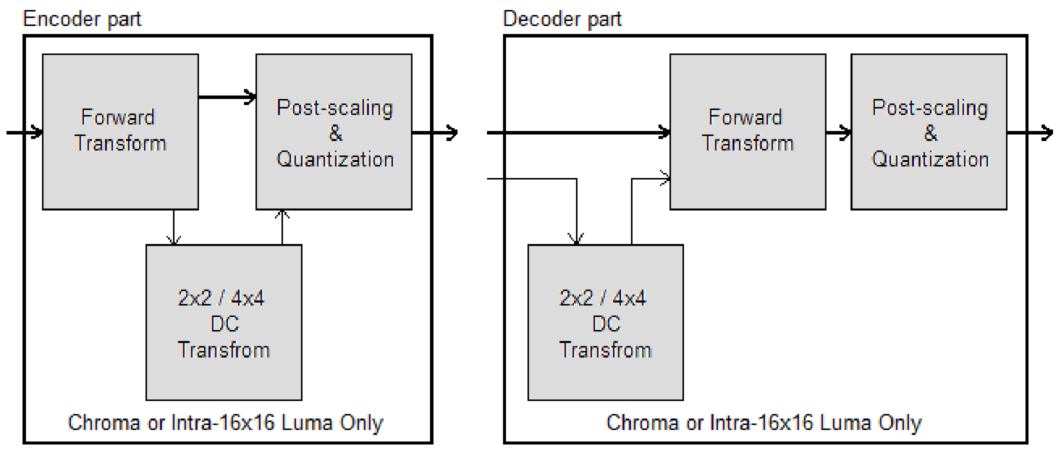

Encoder part
Forward Transform
Post-scaling & Quantization
2x2 / 4x4 DC Transfrom
Chroma or Intra-16x16 Luma Only
Decoder part
Forward Transform
Post-scaling & Quantization
2x2 / 4x4 DC Transfrom
Chroma or Intra-16x16 Luma Only

Figure 5. Transform and quantization block diagram (for Intra_16x16 mode)

1.4. Deblocking filter

The block-based video coding usually results in visually noticeable distortions along the block boundaries for low bit rates. Furthermore these artifacts will also diffuse into the interior of blocks by means of the motion compensation. The H.264/AVC standard defines in-loop deblocking filter which operates within the predictive coding loop, and thus constitutes a required component of the decoding process. Highly adaptive deblocking filter works on three levels, starting from slice level along the edge level down to individual samples level. Consequently, the objective quality is improved by blockiness reduction without significant sharpness waste. Moreover bit rate reduction is observed at the same time, comparing to the non-filtered video sequence.

1.5. Bitstream encoding

The video coding process produces many syntax elements that must to be encoded in bitstream. The bitstream needs to contain:

- complete video sequence information (SEI, SPS, PPS, …)
- information about compression tools/methods used during encoding
- quantized transform coefficients, motion vectors etc.

High level syntax elements are coded using infinite-extend variable length coding (VLC), called a zero-order exponential-Golomb code. A few syntax elements are also coded using simple fixed-length code representations. For the remaining syntax elements (i.e. QCoeffs, MVs), two types of entropy coding can be used, context-adaptive variable length coding (CAVLC) and/or context-adaptive binary arithmetic coding (CABAC) – Fig.6. Each of these encoding methods produces an efficient, compact binary representation of the information. However, CABAC encoding typically provide reductions in bit rate of 10-20 % of the same objective video quality over CAVLC encoding but computationally more intensive. At the end of the process we get encoded bitstream that can be than stored and/or transmitted.

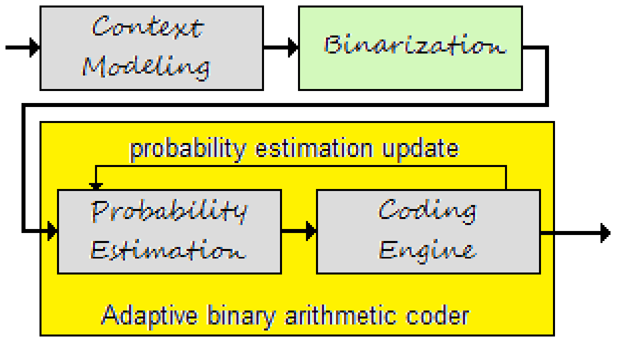


Figure 6. CABAC block diagram

2. REAL TIME H.264 CODEC IMPLEMENTATION

Enhanced compression capabilities i.e. compression rates, objective quality, given by the H.264/AVC standard, are occupied by a high degree of complexity. For real time two way visual communication applications like videophone, extended effort must be performed to ensure that codec parts (encoder, decoder) works smoothly together on a single chip/SoC. Small resolution video systems e.g. CIF@30fps might be successfully implemented on SIMD DSPs like ADSP “Blackfin” family. For higher resolution video systems e.g. Full-HD@60fps – videoconferencing, MIMD architecture DSPs (e.g. TI DaVinci) and/or GPUs (e.g. NVidia) are required. Regardless of processor type, basic problems remains the same, the H.264/AVC standard recommendation specifies decoding part only. Performance and complexity of the encoder adaptive algorithms such as rate-distortion (RD) control, motion estimation (ME), macroblock (MB) mode decision (Intra/Inter/Skip) and decoder error resilience and concealment – Fig.7, left up to the manufacturer. Apart from research, develop and implementation of optimal algorithms, fistful optimization steps need to be performed to reduce computation complexity.

A) Conceptual / Architecture optimizations

Conceptual / Architecture level optimizations should be performed at system design stage. Complete codec pipeline should exploit the hardware limits i.e. extensive usage of video processing co-procs, efficient as possible internal memory usage as memory access is Achilles heel, maximum data re-usage to avoid data redundancy hence memory bandwidth save. Furthermore, computational complexity can be goes down by bypassing MBs that are predicted to be skipped – early MB skip detection.

B) Memory optimizations

The memory requirements of codec is significant, CPU spends a lot of cycles waiting for data from external memory. To minimize CPU cycle waste, we need to apply Direct Memory Access (DMA) to fetch required data into some temporal internal buffers beforehand processing. All DMA transfers should work in concurrent mode (a.k.a. ping-pong) with core processing, to avoid any memory latency. Code and data layout optimization is also very important issue in case of cache misses minimize. Embedded systems characterize smaller cache memory sizes (L1, L2), cache misses are significant thus affecting overall performance. To avoid it, CPU and DMA shouldn't touch and modify the same external memory areas (D-cache coherence), code section should be breakdown into subsections, organized in order of function call frequency downwards (I-cache coherence).

C) Arithmetic optimizations

To fully exploit embedded processor architecture capabilities, some low level optimizations should be carrying out. Core functions (containing pixel operations loops) should be re-write in assembler using, hardware loops, vector operations (SIMD instructions), conditional branches, conditional execution etc. Furthermore some function merging and fast function calls might be applied. Only control code could remain in high-level language e.g. "C", as it's not a bottleneck.

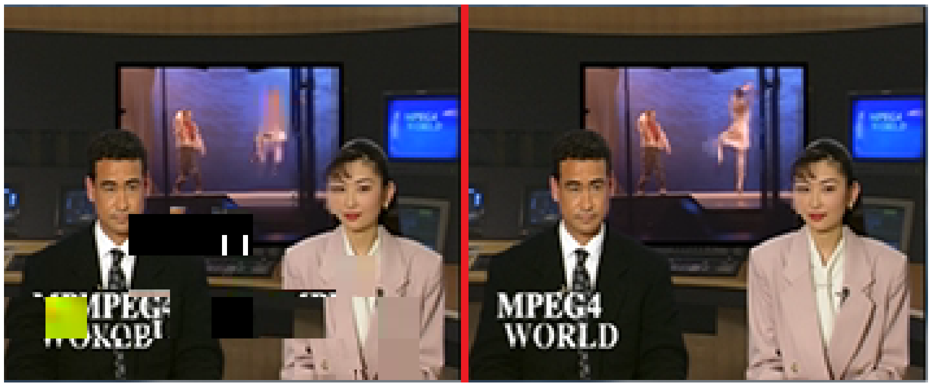


Figure 7. Decoder error concealment

2.1. Early skip detection

Videoconferencing scenario, typically characterize static background scene, to save the encoding effort we need to process only moving parts – "talking heads". All passed over MBs are marked as skipped, if they are full-MBs (16x16) with zero MV and QCoeffs. Skip detection is performed in two steps. First / prior check is added at MB level before encoding (called *early skip detection*). Second / posterior check is added at block level (4x4) after transform and quantization, examining if any non-zero coefficients left to code. Due to high temporal correlation among consecutive frames, lot of macro/blocks tends to have very minimal residual data after motion compensation. Hence, early skip detection can be added prior / into motion estimation, at the expense of minimally increased distortion. Basic idea of early MB skip detection is to compare Sum of Absolute Difference (SAD) – eq. 1, of currently processed MB with some adaptive threshold value – eq. 2, 3.

$$SAD_{0\,MB} = \sum_{x=0,y=0}^{15,15} |\, C_C[x,y] - P_C[x,y] \,| \quad (1)$$

$$\frac{SAD_0}{Q} < Thresh_{SAD0} \quad (2)$$

$$Thresh_{SAD0} \cong N * 2^{\frac{Q}{6}+1} \quad (3)$$

where: C_C[x,y] – MB samples in the current frame
P_C[x,y] – co-located reference frame samples
Q – quantization parameter <0,51>
N – <1,10>

In some cases, when MB occur on edge of moving object – partial motion, more complex analysis needs to be done – frequency domain analysis [5], otherwise more distortions in the decoded / reconstructed video sequence is present.

2.2. Motion estimation

Fast block matching motion estimation (BMME) algorithm needs to be incorporated into real time encoding as motion estimation consumes major percentage of overall power processing and memory/ bandwidth. Lots of fast algorithms was developed over a span of time i.e. UMHexagonS, 3 step search algorithm (3SS), 4 step search algorithm (4SS), diamond search (DS) [6, 7, 8, 9] etc. Most of them minimize SAD cost / distance function – due to its simplicity, eq. 4, 5. Bear in mind that the speed accelerated by the fast motion estimation (up to and more than 90%) results in PSNR loss and bit-rate increase.

$$J_{MB\_cost} = SAD_{MB} + MV\_COST(MVD, Q) \quad (4)$$

$$MVD = |MVP - MV| \quad (5)$$

where: MV_COST – encoding motion vector cost table
MVD – motion vector difference
Q – quantization parameter <0,51>

Search range (estimated motion vector range) puts major influence on computation complexity, hence for high motion video sequence few tricks might be used to enlarge MV scope (quality improve), i.e. get motion vector prediction (PMV) or global motion vector (GMV) as a search center. For full-pel search, smart reference data buffer management of successive MBs result in memory footprint and bandwidth save - Fig. 8.

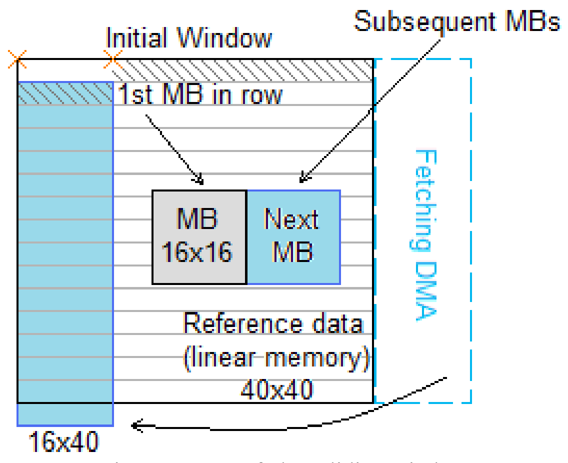


Figure 8. ME ref. data sliding window

Linear reference data organization combined with buffer growing technique come out with significant reduce of internal memory usage and amount of data transferred. For example from - Fig.8, at Full-HD - 1080p video resolution we can save ~60% on data transfers and ~95% on internal memory usage respectively (depending on search range area).

$$40\ x\ 40 * 120(MBs\ in\ row) = 192000B$$
$$40\ x\ 40 + 119 * 16\ x\ 40 = 76160B$$
$$\left(1 - \frac{76160B}{192000B}\right) * 100\% = \sim 60\%$$
$$40\ x\ 40\ +\ 119 * 16 = 3504B$$
$$\left(1 - \frac{3504}{76160}\right) * 100\% = \sim 95\%$$

To increase search range (e.g. +/-32x32 pel) for high resolution video sequences, we can use 2 stage ME strategy. First stage – "Rough ME", works on decimated (↓2) data, giving coarse MV. Second stage – "Fine ME", works on regular data, using coarse MV as search center and giving fine MV as a result. Exact sub-pel[2] motion estimation requires 6-tap filter (1, -5, 20, 20, -5, 1) for generating half-pels, since quarter-pel interpolation is even more complex, to simplify sub-pel MV search a 4-tap filter (-4, 20, 20, -4) approximation might be used [10, 11]. The 4-tap filter reduces computational complexity substantially with minimal quality impact.

2.3. Rate control

Due to the type of compression (context-adaptive) used in the H.264/AVC, encoder produces variable bit-rates from frame to frame, depending on the contents of the input video sequence. We can distinguish three types of encoder working modes in aspect of rate-distortion control:

- *Constant QP (CQP),* quantization parameter remains unchanged over whole video sequence. Output quality and bit-rate varies.

- *Constant bit-rate (CBR),* fixing output bit-rate by leveraging QP value to meet the transmission bandwidth. Output quality varies.

- *Variable bit-rate (VBR),* regulating output bit-rate by leveraging QP value to meet the quality. Output bit-rate varies.

Most of the modern transmission channels are typically constant bit-rate (e.g. PSTN). Under these circumstances, *CBR* seems to be one of the most frequently used mode. Many of the rate control (RC) algorithms are based on rate-distortion (RD) model [12,13] where bit-rate and distortion are functions of the quantization

step size (Qstep), controlled by QP. RD model requires full MB reconstruction for each MB-mode prior decision, which makes it too heavy in terms of complexity to be used in real time applications. In fact, we need fast and simple single pass RC model, giving low end to end delay, controlling QP to meet channel bit budget, where distortion control is done independently – eq. 6, 7.

$$SAD_{MB}(C, P, MODE) = \sum_{x=0,y=0}^{15,15} |C_C(x,y) - P_C(x,y,MODE)| \qquad (6)$$
$$MODE_{MB} = \arg\left[\min SAD_{MB}(C, \hat{C}, MODE|Q)\right] \qquad (7)$$

where: MODE – INTRA {all modes}, INTER {all modes}, IPCM

Indeed TMN5 algorithm [14,15] belongs to this category. Basically algorithm can be divided into three levels of bit allocation:

- *GOP level* control, distributing bit budget on Group of Pictures, first *GOP* frame is I-frame and (N-1) P-frames follow.
- *Frame level* control, distributing bit budget per frame.

- *Macroblock level* control, distributing bit budget per macroblocks unit e.g. macroblock lines.

Predictive RC scheme of rate allocation decision, results in random noise distribution throughout flat / smooth regions for low bit-rates. *Perceptual Quantization* is for efficient bit allocation by taking into account the activity of the MBs – eq. 8, 9, 10, 11, in the frame to be encoded. In that way, more bits are spend for relatively flat regions and less bits are spend for high texture regions as to maintain overall bit-rate at the same level. Hence, for low bit-rates objective (visual) quality is improved.

$$ACT_{16}H_{(n)} = \sum_{x=0,y=0}^{14,15} \left|C_{(n)}(x,y) - C_{(n)}(x+1,y)\right| \tag{8}$$

$$ACT_{16}V_{(n)} = \sum_{x=0,y=0}^{15,14} \left|C_{(n)}(x,y) - C_{(n)}(x,y+1)\right| \tag{9}$$

$$ACT16_{(n)} = ACT_{16}H_{(n)} + ACT_{16}V_{(n)} \tag{10}$$

$$Q_{(n)} = \left(\frac{4ACT16_{(n)} + ACT16_{avg}}{ACT16_{(n)} + 4ACT16_{avg}}\right) Q_{base} \tag{11}$$

where: $C_{(n)}$ – n-th MB

$ACT16_{(n)}$ – MB activity

$ACT16_{avg}$– previous frame activity

Q_{base} – Q-scale computed by the RC (TMN5)

$Q_{(n)}$ – current MB Q-scale prediction

3. CONCLUSIONS

The H.264/AVC standard represents huge step forward in the video coding development. More flexible prediction types and more sophisticated entropy coding methods used in the H.264/AVC, results in better compression rates and higher video quality (compared to the previous standards) or, conversely the same video quality at much lower bit-rates. The H.264/AVC reference software (so-called Joint Model or simply JM) provided by *Joint Video Team* (JVT) includes all the features specified in the standard. Although, this piece of software is highly configurable, it's very slow and not suitable for practical implementation – tab.1-2. Completely new design based on embedded platform architecture is needful. For embedded platform/EVM, processing time (encode, decode) @D1:NTSC resolution is about 33[ms/frame] which is on the edge of real time video. Assuming some subsystem activity - pre/post video processing, stand alone codec should meet at least 32fps - 30[ms/frame]. For lower bit-rates (<1.5Mbps), presented embedded codec hit the spot.

Encoder	Performance*	
	[ms/frame]	PSNR[dB]
JM15.1 (PC†)	2000	31.34
Embedded (PC†)	170	30.26
Embedded (EVM‡)	21	30.26

Table 1. H.264 encoder performance: BP L4.0, CBR 2Mbps, Content: Cheerleader D1(NTSC: 720x480@30fps)

Decoder Performance*	JM15.1 (PC†)	Embedded (PC†)	Embedded (EVM‡)

[ms/frame]	237	75	11

Table 2. H.264 decoder performance: BP

* average, depending on content and parameter set
† Intel Core2 Duo T5450 @ 1.66GHz, 2GB RAM
‡ dm6446(ARM@300MHz, DSP@600MHz), 256MB RAM

The mix of optimization techniques and algorithms presented in this paper were cross-checked and implemented by the author on Texas Instruments (TI) dm6446 platform and can be effectively used on many other programmable platforms with alike architecture (e.g. dm64xx, adsp-bf5xx, msc81xx).

4. REFERENCES

[1] ITU-T Recommendation H.264 | ISO/IEC 14496-10, "Advanced video coding for generic audiovisual Services" | "Part 10: Advanced Video Coding", ITU-T | ISO/IEC, March 2009 | 2008

[2] I. Richardson, "H.264 and MPEG4 Video Compression", John Wiley & Sons, September 2003

[3] G. Raja[*] and M. J. Mirza[†], "Evaluation of Emerging JVT H.264/AVC with MPEG Video", *Department of Electrical Engineering, University of Engineering and Technology, Taxila, Pakistan , †Advance Engineering Research Organisation, Wah Cantt. Pakistan

[4] G. J. Sullivan[*], P. Topiwala[†] and A. Luthra[‡], "The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions", *Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, †FastVDO LLC, 7150 Riverwood Dr., Columbia, MD 21046, ‡Motorola Inc., BCS, 6420 Sequence Dr., San Diego, CA 92121

[5] Y. Zhao and I. E. G. Richardson, " Macroblock Skip-Mode Prediction For Complexity Control Of Video Encoders", The Robert Gordon University, Aberdeen, UK

[6] ITU-T SG16/Q6, "Comments on Motion Estimation Algorithms in Current JM Software", Nice, FR, 14-21 October, 2005

[7] K.Ramkishor and S.Krishna, "Spatio-temporal correlation based fast motion estimation algorithm for MPEG-2", IEEE Proceedings of the 35th Asimolar conference on Signals, Systems and Computers, pp. 220-224 vol. 1, November 2001

[8] P. De Pascalis[*], G. A. Mian[*], L. Pezzoni[†] and D. Bagni[†], "Fast Motion Estimation with Size-Based Predictors Selection Hexagon Search in H.264/AVC encoding", *Department of Information Engineering, University of Padova, Italy, †Advanced system Technology labs, STMicroelectronics, Agrate Brianza, Italy

[9] Chi-Wai Lam[*], Lai-Man Po[*] and Chun Ho Cheung[*†], "A New Cross-Diamond Search Algorithm for Fast Block Matching Motion Estimation", *Department of Electronic Engineering City University of Hong Kong, Hong Kong SAR, †Department of Information Technology, Hong Kong Institute of Technology, Hong Kong SAR

[10] PSSBK Gupta and K. Ramkishor, "Novel Algorithm to Reduce Complexity of Quarter Pixel Motion Estimation", Visual Communication and Image Processing (VCIP), California, January 2004

[11] *ITU-T SG16/Q15, "*Results of core experiment on Adaptive Motion Accuracy (AMA) with 1/2, 1/4 and 1/8-pel accuracy*",* Osaka, Japan, 16-18 May, 2000

[12] JVT-D030-1, "Rate Control on JVT Standard", Klagenfurt, Austria, 22-26 July, 2002

[13] JVT-E069, "Improved Rate Control Algorithm", Geneva, CH, 9-17 October, 2002

[14] P. Hsu and K. J. Ray Liu, "A Predictive H.263 Bit-Rate Control Scheme Based on Scene Information", Department of Electrical and Computer Engineering, University of Maryland at College Park, College Park, Maryland, USA, 2000

[15] Siwei Ma[*], Wen Gao[*], Feng Wu[†] and Yan Lu[‡], "Rate Control for JVT Video Coding Scheme with HRD Considerations", *Institute of Computing Technology, Chinese Academy of Science, Beijing, 100080, China, †Microsoft Research Asia, Beijing, 100080, China, ‡Department of Computer Science, Harbin Institute of Technology, Harbin, 150001, China, 2003

[16] V. Ramamoorthy, "A ρ-domain Rate Control Algorithm for the H.264 Encoder", 6704 Paseo San Leon Pleasanton, CA 94566, 2004

[17] JVT-H017, "Proposed Draft of Adaptive Rate Control", Geneva, May 20-26,2003

[18] JVT-Q080, "The Rate Distortion Function of H.264 Transform Coefficients", Nice, FR, 14-21 October, 2005

Links:
http://www.itu.int/
http://www.vcodex.com/
http://www.itu.int/ITU-T/studygroups/com16/jvt/
http://iphome.hhi.de/suehring/tml/index.htm
http://www.videolan.org/developers/x264.html
http://www.chiariglione.org/
http://www.mpeg.org/
http://www.mpegif.org/

http://wftp3.itu.int/av-arch/video-site/sequences/
http://www.its.bldrdoc.gov/vqeg/
http://trace.eas.asu.edu/yuv/index.html
ftp://ftp.tnt.uni-hannover.de/pub/