FLASH VIDEO TECHNOLOGY

Ing. Fabio Sonnati – **Progetto Sinergia**
(Revision 2 – September 2006)

Introduction to white paper

This whitepaper analyzes the various video technologies currently implemented in Flash Player 8/9. This is a technical document written to enhance the knowledge of Flash Video capabilities.

Flash implements a set of video codecs similar to, or in some cases, derived from H.263 and H.264 standards. To better understand the potentiality of Flash Video, it will be usefull to start examining these standards and then the main differences in Flash implementation. The first Chapter of this paper is dedicated to an indeep examination of the basic video compression standards (H.261, H.263, H.263+) and the advanced ones (H.263v2, H.264).

In the second Chapter, we'll analyze the specific Flash Video codec implementations (Spark and VP6) and the codec used for real-time video compression (Spark).

Differently from the first release of this whitepaper, the topic of improving the real-time encoding capability of Flash Player, will be discussed in a separate document (Flash Video Optimizations).

If you are expert of video compression techniques you may jump to chapter 2.

This whitepaper is dedicated to all the Flash Video and Flash Media Server developers out there.

1 - Introduction to video compression standards

Digital video is being adopted in an increasing range of applications covering video telephony, videoconferencing, DVD, digital TV, Internet streaming. The adoption of digital video in so many applications has been accelerated by the development of many video coding standards. These standards provide interoperability between different systems and applications facilitating the growth of the digital video market.

The ITU-T (International Telecommunications Union) and the ISO/IEC (International Standardization Organization/International Electrotechnical Commission) are the only two formal organizations that develope video coding standards. The ITU-T video coding standards, called recommendations and denoted with H.26x (e.g., H.261, H.262, H.263 and H.264) are usually optimized for real-time video comunication such as videoconference and video telephony while the ISO/IEC standards, denoted with MPEG-x (e.g., MPEG-1, MPEG-2 and MPEG-4), are mainly designed for storage (DVD) and broadcast (satellite and digital TV).

For the most part, the two standardization committees have worked independently on the different standards. The only exception has been the H.262/MPEG-2 standard, which was developed jointly by the two committees. Recently, the ITU-T and the ISO/IEC JTC1 have agreed to join their efforts in the development of the emerging H.264 standard, which was initiated by the ITU-T committee.

Figure summarizes the evolution of the ITU-T recommendations and the ISO/IEC MPEG standards.

<Sn>
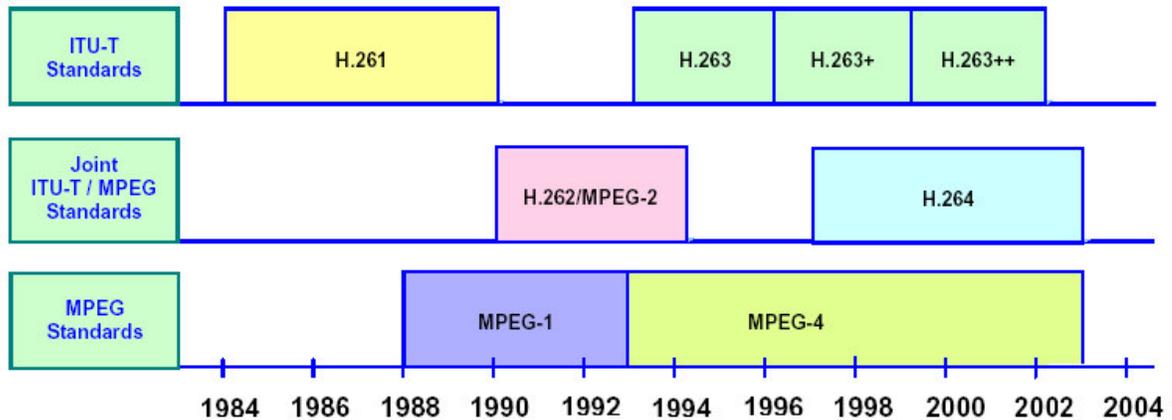
The main objective of the H.26x standards is to provide a mean to achieve substantially high video bandwidth compression for real-time communication applications.

The underlying approach of all H.26x is similar and consists of the following four main stages:

1. *Dividing each video frame into blocks of pixels so that processing of the video frame can be conducted at the block level.*

2. *Exploiting the spatial redundancies that exist within the video frame by coding some of the original blocks through spatial prediction, transform, quantization and entropy coding (or variable-length coding).*

3. *Exploiting the temporal dependencies that exist between blocks in successive frames, so that only changes between successive frames need to be encoded. This is accomplished by using* **motion estimation and compensation**. *For any given block, a search is performed in the previously coded frame to determine the motion vectors that are then used by the encoder and the decoder to predict the subject block.*

4. *Exploiting any remaining spatial redundancies that exist within the video frame by coding the residual blocks, i.e., the difference between the original blocks and the corresponding predicted blocks, again through transform, quantization and entropy coding.*

1.1 - H.261 Compression Standard

H. 261 Compression has been specifically designed for video telecommunication applications. Developed by CCITT (ITU-T) in 1988-1990, H.261 has been widely used in videotelephone applications and videoconference systems over ISDN lines. The Bandwidth target is an integral multiple of baseline ISDN bandwidth (p x 64Kbit/s).

The basic approach to H. 261 Compression is here summarized:

- Frame types are CCIR 601 CIF (352x288) and QCIF (176x144) images, in color space YCbCr with 4:2:0 chrominance subsampling.

- There are two frame types: Intra-frames (*I-frames*) and Inter-frames (*P-frames*) forming the following frame sequence: I,P,P,…,P,P,I,P,P,…,P,P,I,…
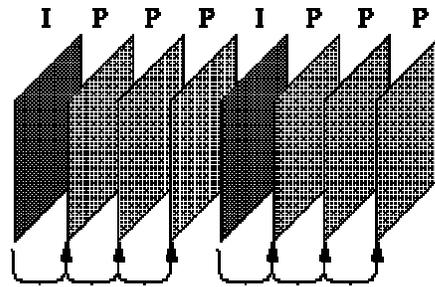
- I-frames use basically JPEG compression (DCT trasform, quantization, entropy coding)

- P-frames use **pseudo-differences** from previous frame (P or I), so frames depend on each other and prediction goes forward in time.

- I-frames provide a mean of sincronization between the stream source and the stream destinations. I-frames are also called Key-Frames.

Intra Frame Coding

In **intra frame coding** the lossy compression techniques are performed relative to information that is contained only within the current frame, and not relative to any other frame in the video sequence.

In other words, no temporal processing is performed outside of the current picture or frame. This mode will be described first because non-intra coding techniques are extensions to these basics.

Figure shows a block diagram of a basic video encoder for intra frames only. It is very similar to that of a JPEG still image video encoder:
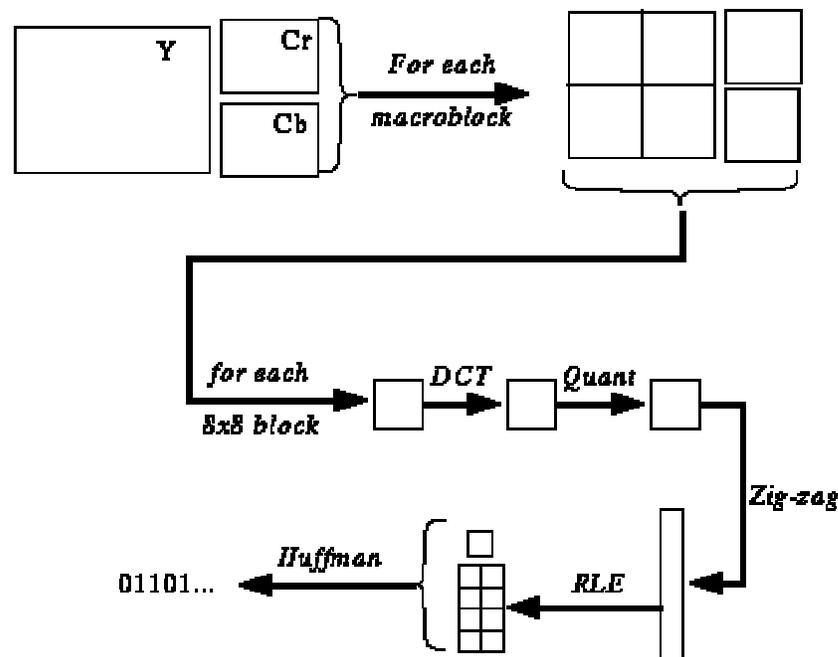
The basic processing blocks shown are the *video filter*, *discrete cosine transform*, *DCT coefficients quantizer*, and *run-length amplitude/variable length coder*.

The video filter processes and prepares the original frame format for the other blocks. The frame is converted in YCbCr (or YUV) color space and then logically subdivided in *Block* (8x8 pixels) and *Macroblocks* (16x16 pixels).

Research into the Human Visual System (HVS) has shown that the eye is most sensitive to changes in luminance, and less sensitive to variations in chrominance. To gain compression advantage from such eye's characteristic, H/261 (and MPEG) uses the YCbCr color space to represent the data values instead of RGB, where Y is the luminance signal and CbCr the chrominance signal (Cb is the blue color difference signal, and Cr is the red color difference signal).

A macroblock can be represented in several different manners when referring to the YCbCr color space. There are 3 mainly known formats: 4:4:4, 4:2:2, and 4:2:0 video. 4:4:4 is full bandwidth YCbCr video, and each macroblock consists of 4 Y blocks, 4 Cb blocks, and 4 Cr blocks. Being full bandwidth, this format contains as much information as the data would if it were in the RGB color space. 4:2:2 contains half as much chrominance information as 4:4:4 (downsampling of 2 along X axis), and 4:2:0 contains one quarter of the chrominance information (downsampling of 2 along X and Y axis). H.261 and the majority of video-codecs use this third mode that allows an immediate 1:2 bandwidth compression without appreciable loss of quality.

After Video filtering, each 8x8 block of the frame is trasformed from space domain to spatial frequency domain using a DCT (Discrete Cosine Transform). The DCT coefficients represent the spatial details of the Picture Block. Quantizing coarsely the higher frequency coefficients retaining more bits of information for the lower frequency coefficients brings to block bit-stream compression with moderate loss of quality.

The Zig-Zag reading of the DCT quantized coefficients is a typical trick to obtain a better organization of values for a more efficient compression in the successive Entropy Coding Block (bit-stream redundancies reduction through the use of Variable Length Coding).

What is the purpose of the Zig-zag Scan ?

• to group low frequency coefficients in top of vector.

• to map an 8 x 8 matrix to a 1 x 64 vector
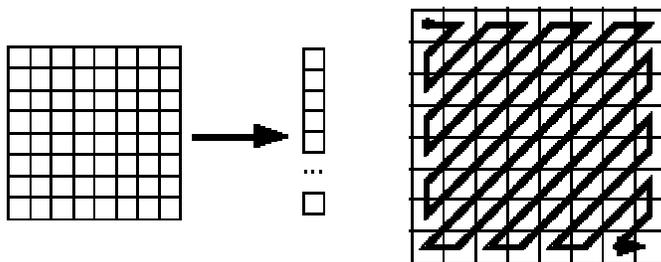
Inter-frame (P-frame) Coding

The previously discussed intra frame coding techniques were limited to processing the video signal on a spatial basis, relative only to information within the current video frame. Considerably more compression efficiency can be obtained however, if the inherent temporal, or time-based redundancies, are exploited as well.

Temporal processings, to exploit this redundancy, use a technique known as **block-based motion compensated prediction, using motion estimation**. A block diagram of the basic encoder with extensions for non-intra frame coding techniques is given in Figure. Of course, this encoder can also support intra frame coding as a subset.
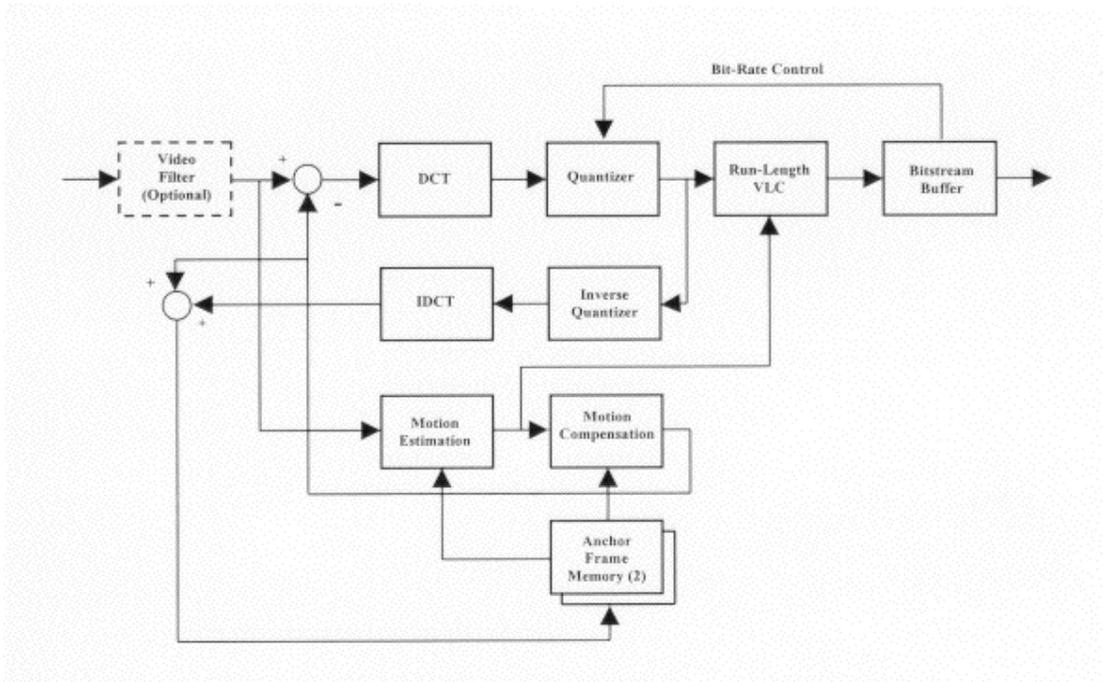
P-Frame Coding

Starting with an intra, or I frame, the encoder can forward predict a future frame. This is commonly referred to as a P frame, and it may also be predicted from other P frames, although only in a forward time manner. As an example, consider a group of pictures that lasts for N frames. In this case, the frame ordering is given as I,P,P,P,…,P,I,P,P,P,P,…

Each P frame in this sequence is predicted, on a Macroblock basis, from the frame immediately preceding it, whether it is an I frame or a P frame. As a reminder, I frames are coded spatially with no reference to any other frame in the sequence.

P-coding can be summarised as follows:
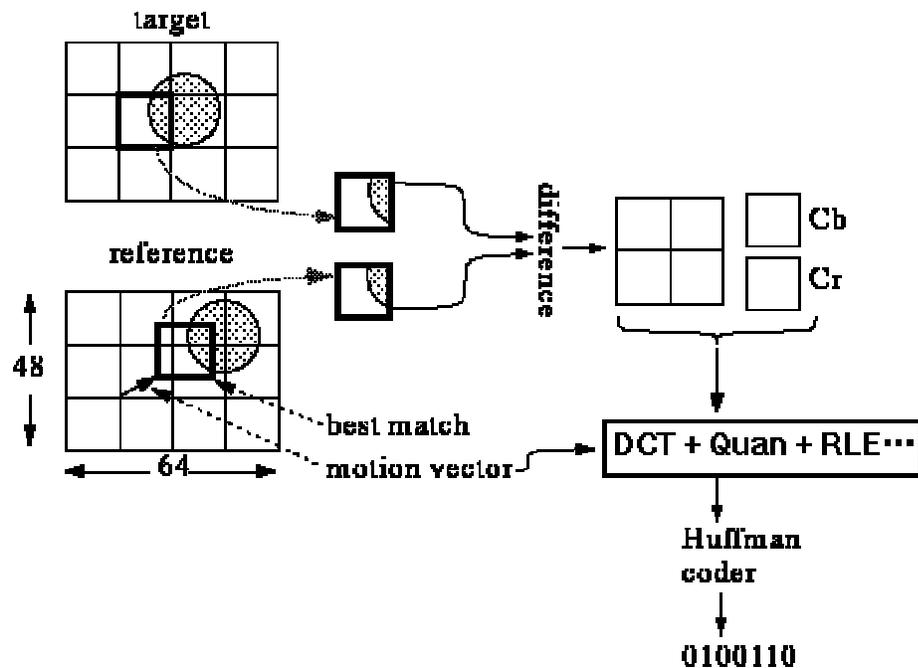
The key principle of Inter-frame compression is to recover, from the previous frame, as much information as possibile. This is based on the fact that information between adiacent frame is redundant, expecially for slow-variating or still frames. Recovering information using Inter-frame prediction, reduces this temporal-redundancy.

For each Macroblock on the current frame the codec tries to find, in the surrounding of the relative macroblock of the previous frame, the *reference area* (again of 16x16 pixel) that best matches the current macroblock of the current frame. Indeed the codec tryes to minimize the difference signal (Mean Absolute Difference or Mean Squared Error).

Once found the *reference area*, we can process (compress) the difference signal again using DCT and quantization, and transmit this compressed bitstream plus the Motion Vector that point to the reference area. These information are sufficient for the decoder to rebuilt the block with lesser amount of information than that required compressing the stand-alone Macroblock using Intra-compression only.

H.261 defines the Motion Vector as an Integer in the range [-16,+15] (1 pixel or "pel" accuracy).

The processing of Macroblocks in I-frames and P-frames are summarized by the following Figure:
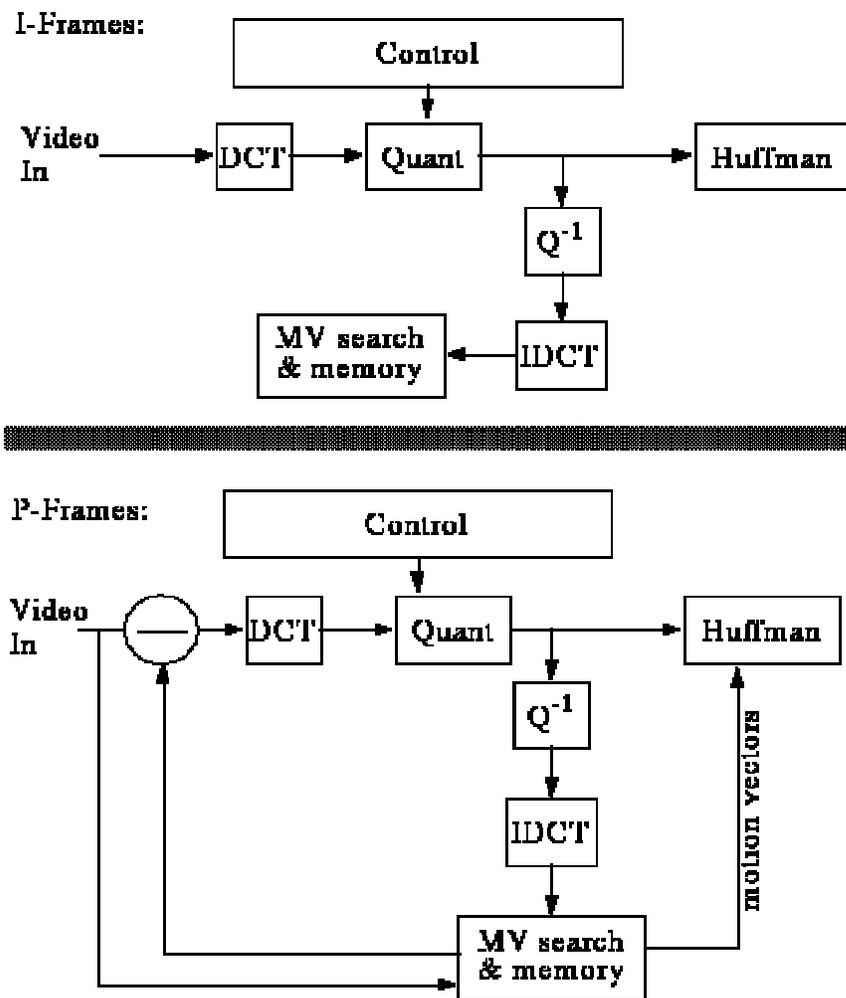
1.2 - H.263 Compression Standard

Introduction

ITU-T developed H.263 in the years 1993-1996. H.263 has the goal to achieve better compression than H.261 with much more flexibility, especially for low bit rate IP channels.
H.263 may be thought as an evolution of H.261 combined with MPEG-like features and others optimizations for very low bit rates. Compared to H.261, H.263 has the following base improvements:

– *More picture formats and different GOB structures.*
– *Half-pel motion compensation.*
– *Improved VLC table.*
– *Four negotiable options.*

The base improvements lead to an average PSNR (Power Signal Noise Ratio) that is 3-4dB better than H.261 at bit-rate lower than 64kbit/s.

More picture formats

The picture format is the same YCbCr 4:2:0 used in H.261
The allowed resolutions are: Sub-CIF (128x96), QCIF(176x144), 4CIF(704x576) and 16CIF(1408x1152) up to 30-60 Hz (Fps).

More accurate motion prediction

Resolution of Motion Vectors is now half-pel (half-pixel) in the range [-16,+15.5]
The generic Motion Vector is predicted by the values of the surrounding MV. Infact, in a frame, the motion is a local property and adiacent blocks have similar motion vectors. The predicted value is used to reduce the amount of information trasmitted. Only the difference signal (delta signal) between the real vector and its prediction is transmitted (see Figure).
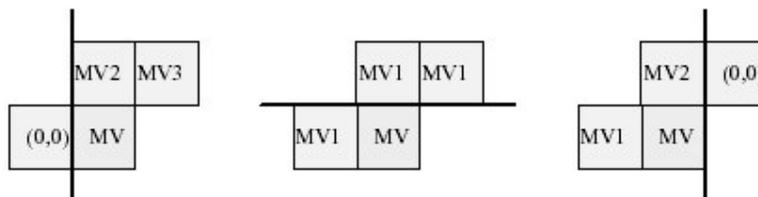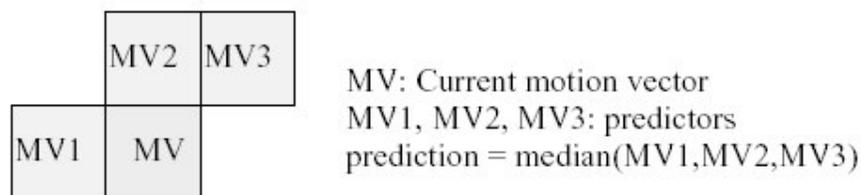
Improved VLC Table

Tables containing the Variable Length Code for the Entropy Coding of various parameters (DCT coefficients, quatization levels, motion vectors) have been optimized. This leads to an improvement in bit-stream compression of about 5-7%.

Negotiable Options in H.263

- Unrestricted Motion Vector Mode
- Advanced Prediction Mode
- PB-Frame Mode
- Syntax-based Arithmetic Coding Mode

Unrestricted Motion Vector (UMV) Mode

• Motion vectors may point outside the picture
Edge pels are repeated as prediction of the points outside the boundaries of picture. Significant gain for movements along the edge of the pictures, especially for smaller picture formats.
• Extension of the motion vector range
[–31.5, 31.5] instead of [–16, 15.5]
Especially useful for 4CIF, 16CIF where motion is usually of wider range.
• Good for camera movement and background motion

**Advanced Prediction Mode (AP)**

• Four motion vectors for each MB (one for each 8×8 block)
The sub-division is usefull in areas with compound motion. Use more bits, but give better prediction. Encoder decides which MBs to apply

Overlapped block motion compensation (OBMC)

• This option reduce blocking artifacts in scenes with strong motion level.
Motion vectors may point outside as in UMV

PB-Frame (PB) Mode

• A PB-frame consists of two pictures
P-picture: Predicted from the last decoded picture as usual
B-picture: Predicted from both the last valid picture (P or I) and following P-picture (see Figure)
Bidirection prediction improves compression ratio but introduces latency and the need of an higher processing power. B picture is never referenced by other picture thus it is possible to discard B-pictures in a stream to provide time-scalability and bandwidth scalability.
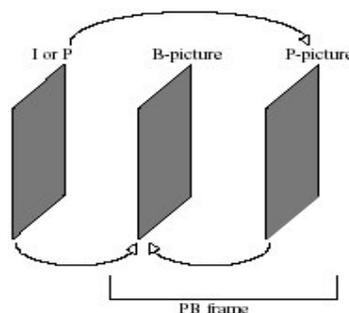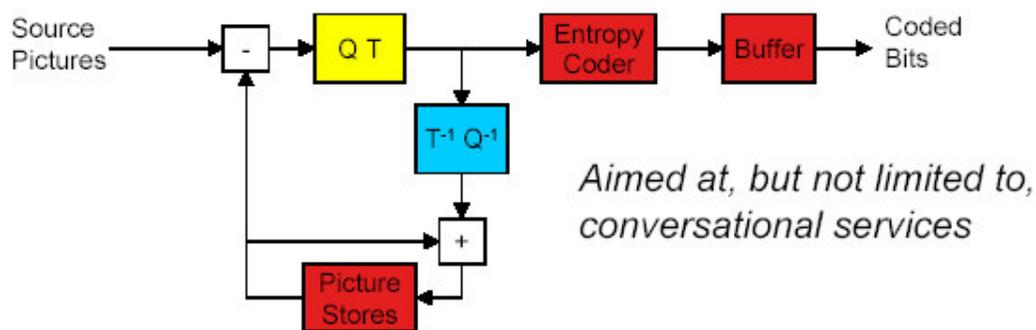
Syntax-Based Arithmetic Coding (SAC) Mode

• Arithmetic coding is used instead of VLC (Variable Length Coding)
Instead of using static VLC table, SAC uses Cumulative frequencies calculated both at coder and decoder level.
• SAC gives about 5% less bits at the same SNR
Inter frames gain: 3~4%
Intra blocks and frames gain: ~10%

Key features of H.263 and schematic block diagram are here summarized:

1.3 - H.263 Version 2 (H.263v2 or H.263+)

H.263+ is the natural evolution of the base standard. ITU-T developed H.263+ in the years 1996-1998. The basic concepts and techniques of this standard may be found in the later MPEG4 standard.
Enhancements of H.263+ over H.263 are:

– *Extended source formats*
– *16 negotiable optional mode*
– *Supplemental enhancement information*

New Negotiable Options Examples:

Advanced Intra Coding Mode

The standard introduces a spatial prediction of DCT coefficients in Intra compression. This is similar to Motion Vector prediction but applied to DCT coefficients. There are 3 prediction mode: DC only, vertical DC & AC, horizontal DC & AC.
10% improvement in Intra compression.

Alternate Inter VLC Mode

This mode uses separate VLC table for inter DCT and intra DCT. The use of different VLC tables for the various parameters allows better compression at the cost of higher coding complexity.

Deblocking Filter Mode

Depending on quantization step size the codec applies a debloking filter to Blocks in order to improve visual quality perception. This is very helpful in the case of very high compression ratio.

Modified Quantization Mode

More flexible changes of quantization step sizes and finer quantization for chrominance.
The gain in SNR for chrominance levels is considerable. Less chromatic artifacts.

Improved PB-Frame Mode

For each Macroblock is now possible to choose for forward, backward, or bi-directional prediction. The motion vectors are predicted from the mean values of the previous and following frame. If prediction is good enough, delta signal is not trasmitted at all with a consequent bandwidth gain.
Other modes are dedicated to scalability in time, SNR (quality), space, and error resilience.
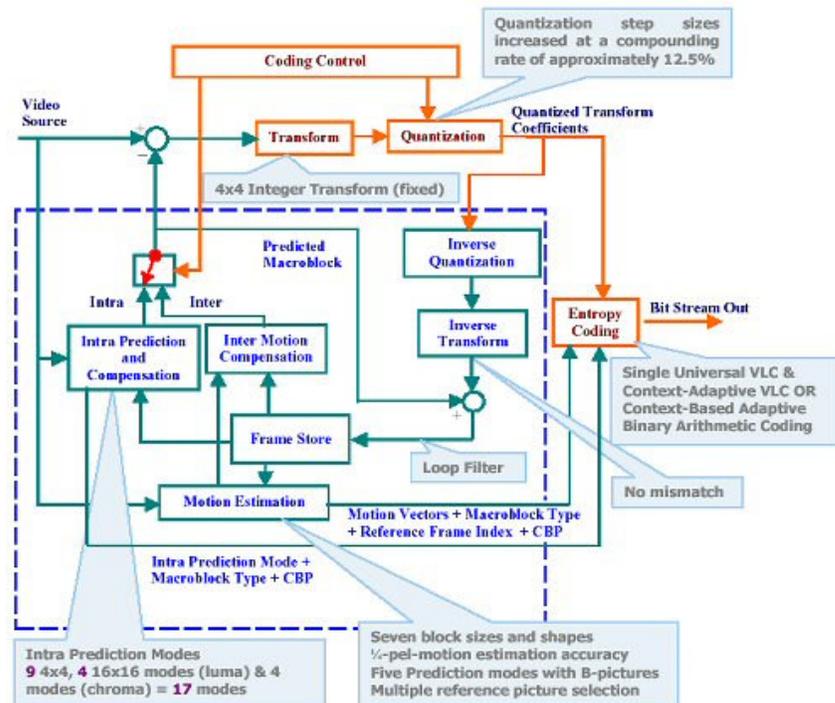Key features of H.263+ and schematic block diagram are here summarized:
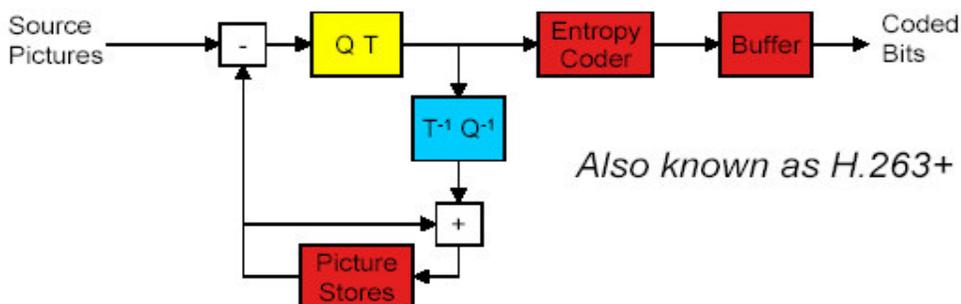
Further optional modes of operation added:
- Advanced Intra Coding - using spatial prediction
- Deblocking Filter
- Reference Picture Selection
- Scalability and B pictures
- Reference Picture Resampling

<Sn>

1.4 - H.264 (formerly H.26L)

H.264, **MPEG-4 Part 10** was written by the ITU-T together with the ISO/IEC Movie Picture Experts Group (MPEG) as the product of a collective partnership effort known as the Joint Video Team (JVT). The ITU-T **H.264** standard and the ISO/IEC **MPEG-4** **Part 10** standard (formally, ISO/IEC 14496-10) are technically identical. The final drafting work on the first version of the standard was completed in May of 2003.

H.264 contains a number of new features that allow it to compress video much more effectively than older H.26x standards:

New trasform design:

An exact-match integer 4×4 spatial block transform is used instead of the well known 8x8 DCT. It is conceptually similar to DCT but with less ringing artifacts. There is also a 8x8 spatial block transform for less detailed areas and chroma.

A secondary Hadamard Transform (2x2 on chroma and 4x4 on luma) can be usually performed on "DC" coefficients to obtain even more compression in smooth regions.

There are also an optimized quantization and two possible zig-zag pattern for Run Length Encoding of transformed coefficients.

Intra-frame compression

H.264 introduces complex spatial prediction for intra-frame compression.
Rather than the "DC"-only prediction found in MPEG2 and the transform coefficient prediction found in H.263+, H.264 defines 6 prediction directions (modes) to predict spatial information from neighbouring blocks when encoded using 4x4 transform.

There is also 4 prediction mode appliable to 16x16 blocks. Residual data are coded with 4x4 trasforms and a further 4x4 Hadamard trasform is used for DC coefficients.

Multiple Reference Frames

H.264 uses previously-encoded pictures as references in a much more flexible way than in past standards, allowing up to 5 reference pictures to be used (unlike in prior standards, where the limit was typically one or, in the case of conventional B frame, two). In certain types of scenes, for example scenes with rapid repetitive flashing or back-and-forth scene cuts or uncovered background areas, it allows a very significant reduction in bit rate.

Motion Compensation

Uses P frame (predicted) and B frame (interpolated) with enhanced pel resolution (1/4).
Motion compensation is done using seven macroblock configurations with block size as large as 16x16 and as small as 4x4. Each macroblock can have a different reference picture.
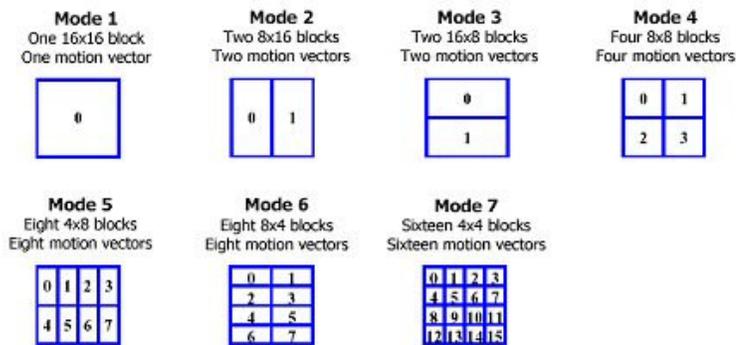
B frames are predicted from previous and/or future pictures with 5 prediction Modes (intra, forward, backward, interpolated and direct).

Weighted prediction allows an encoder to specify the use of a scaling and offset when performing motion compensation providing a significant benefit in performance in special cases—such as fade-to-black, fade-in, and cross-fade transitions.

Loop Filter

Loop filtering is mandatory in the encoder. A lot of efficiency is due to the loop filter. The strength of filter depends on intra/inter coding, differential vectors, quantization level. Up to 40% of total processing power may be required by this kind of filter.

Entropy Coding

For entropy coding, H.264 may use an enhanced VLC, a more complex context-adaptive variable-length coding (CAVLC) or an ever more complex Context-adaptive binary-arithmetic coding (CABAC) which are complex techniques to losslessly compress syntax elements in the video stream knowing the probabilities of syntax elements in a given context.

These techniques, along with several others, help H.264 to perform significantly better than any prior standard can, under a wide variety of circumstances in a wide variety of application environments. H.264 can often perform radically better than MPEG-2 video—typically obtaining the same quality at half of the bit rate or less. Today, H.264 belongs to the State of the Art in video encoding.

<Sn>

2 - Technical Analysis of Flash Video Codecs

Starting from Flash Player 6 (2002), Macromedia equipped its plug-in with audio and video codecs. The first video codec was Spark, provided by Sorenson. It has represented the first generation of Flash Video Technology. The Player has, even now, both decoding and encoding capabilities with this video codec.
However, Flash Player 6 had the limitation to use its video capabilities only in streaming mode, with the need of an expensive Flash Communication Server (now Flash Media Server) license also for playing simple video clips. This limit has been a reason for a very slow adoption rate at beginning.

Flash Player 7 (late 2003) extended the use of Spark video introducing a progressive downloading mode. With the progressive downloading of FLV video files, Flash has started to become a very interesting platform for video delivery over the web. Ubiquity of the player, possibility to integrate video with custom interfaces and the efficiency of progressive downloading are features that developers were waiting for.

The third chapter of this story is the adoption of a new and improved video technology starting from Flash Player 8 (sept. 2005). The second generation video technology is called VP6 and is provided by On2, a firm specialized in video codecs. This time, however, Macromedia has chosen to include in the Flash Player the decoder only, leaving the possibility to encode video in real-time only using the old spark codec. The VP6 implementation is not therefore a true "codec" but simply a "decoder".

Let's now analyze the features of these two video technologies. We'll compare the techniques used by Spark and VP6 with those decribed over and used by international video encoding standards.

2.1 – Sorenson's Spark

Sorenson's Spark is a video codec derived from H.263 standard. The Codec has a small footprint (approx. 100Kbytes), an important feature for an internet plug-in that must be as light as possible.

Spark supports a subset of H.263 basic features. In details, it doesn't support arithmetic coding (SAC mode), B frames (PB mode) and Advanced Prediction Mode. These features was probably too much expensive in terms of processing power and codec complexity to be included in a small codec, furthermore B frames are good for pre-recorded video but not for realtime streaming.

B frames introduce latency and double the required processing power. Spark codec replaces B frames with D frames (disposable frames). D Frames are essentially B frames with only forward prediction (a B frame uses both previous and next P or I frame as references, while D frame uses only the previous P or I frame). When the codec uses D frames, it achieves lesser compression but has a method to modulate in realtime the bandwidth required to stream the video because the D frames are "Discartable", like B frames, without the need to resync to the next keyframe.

When used as a video compressor to feed a Flash Media Server, Flash Player's spark codec always uses D frames and the video stream appears as a sequence of frames like this:

I-D-P-D-P-D-P-…-D-P-I-D-P….(I=keyframe, D=desposable frame, P=predictive frame)

When a video is encoded to spark off-line by a dedicated application, only I and P frames are used.

Spark supports by default the UMV(Unrestricted Motion Vector) mode and admits arbitrary frame dimensions. Furthermore it supports a Deblocking filter mode similar to that of H.263+

With these features, the Spark codec is technologically 20-30% less efficient than an optimal MPEG4 codec based on H.263+ in terms of quality / bandwidth ratio. A good performance in the years of the first implementation but every day less satisfactory in more recent time. Although that, spark encoded FLVs are even today widely used thanks to free, efficient and very fast encoders like FFMPEG.

Real-time Compression Capabilities

As we have read, the "theoretical" compression capabilities of Spark codec are not the state of the art but are good.

If you use a dedicated application like Flash Video Importer, Sorenson Squeeze, FlixPro or even FFMPEG to produce a stand alone FLV from a good video source, you are able to obtain a good result.

A situation completely different is to use a Flash movie to acquire and send a Video stream to a Flash Media Server for recording or real-time delivery. In this case, the compression is done in real time by the Flash movie itself and the final result, in most cases, may be far from the optimal.

We must consider that video compression is an asymmetric task. Compression is quite complex and time consuming while de-compression is much faster, therefore compression part of the Flash video codec has been simplified to fit in the small plug-in.

For any given codec there may be a great difference in efficiency between different implementation in video compression. Compression strategies are various and differ each other in terms of required processing power, memory and code length. Macromedia programmers have chosen to implement only basic video compression strategies in the Flash Player code to keep the codec footprint and processing power requirement very low.

Analysing the final encoded stream, it's possible to recognise an appreciable loss of quality compared to the same source compressed off-line, and this happens with all the real-time encoding options. We will cover in a separate whitepaper a set of optimization tecniques to improve widely the real-time Flash Player's encoding performances.

2.2 – On2's VP6

For the Flash Player 8, Macromedia needed a state-of-the-art video codec to sustein the adoption of the Flash Player as the ideal video delivery platform. The choice fallen on On2's VP6. You may ask why did Macromedia chosen a proprietary technology instead of H.264. The reason is clear. H.264 must be licensed by paying expensive royalties to two patent pools, MPEG LA and VIA Licensing.

As a result, the costs of deploying H.264 are very high. Under the MPEG LA terms, in larger deployments encoder and decoder fees are $0.20 per unit up to five million units (beyond the price is $0.10). There are also content-based fees. The cap for these fees is $3.5 million (per firm) in 2005 and 2006, $4.25 million in 2007 and 2008 and $5 million in 2009 and 2010.

Under the terms of the VIA License, in larger deployments, encoder and decoder fees are $0.25 per unit, as well as content fees. As an annual cap on content fees, non-PC OEMs pay $2.5 million a year and PC OEMs pay $4 million.

These royalty are only for the intellectual property licenses and not for a working codec, thus to this cost we must add the cost of developing an efficient H.264 compliant codec.

The choice of VP6, which is royalty-free, has permitted Macromedia to avoid the payment of several million dollars in fees.

On2 is a firm specialized in high-quality video codec. In the last 15 years, On2 has released a set of very good codecs: VP3 (later released as open source and become the base for Theora project), VP4, VP5, VP6 and VP7. Macromedia has licensed both VP6 and VP7 but, by now, has included only a VP6 decoder in Flash Player 8 and an encoder in the Flash Player 8 Professional Video Exporter. After the deal Adobe-Macromedia, It's not clear what use Adobe will do of the VP7 license.

On2 declares VP6 to be a H.264-class codec and even to surpass it in Power Signal / Noise Ratio (PSNR) in many scenarios. Beeing a proprietary technology, we don't have detailed informations about compression techniques used. All informations below derive from unofficial web materials, VP3 format, On2's white papers and interviews.

Technical details

VP6 makes use of Intra compressed frames (I-frames) and unidirectionally predicted frames (P-frames). There are no B-type frames, but P frames can have multiple reference frames in the past.

VP6 use a somewhat traditional 8x8 iDCT-class transform for spatial to frequency domain transformation (VP7 uses a 4x4 trasform, similar to H.264) . Intra-compression makes use of spatial predictors, although not as much advanced as those found in VP7 or in H.264 (probably similar to what is used in H.263++).

Macroblocks are arrays of 16x16 pixels and motion prediction is done with one vector per macroblock or 4 vectors (one for each 8x8 block). There are a number of motion vector prediction modes and for each macroblock is possible to choose between two reference frames: the previous frame, and a previously bookmarked frame.

In older VPx codecs, this second reference frame was, necessarily, the previous I-frame (keyframe). The bookmarked reference frame approach is more accurate and is very usefull to reduce bitrate in fast-changing scenes. Using more than 2 reference frame, leads indeed to modest improvements in compression ratio.

VP6 uses an "adaptive sub-pixel motion estimation". The filters used for motion estimation are sensitive to content which allows the motion estimation to better preserve detail.
Quarter pel motion compensation is supported as well as unrestricted motion compensation. The range of motion compensation has been increased from previous codecs (extended long range).

VP6 takes advantages also of a better prediction of low-order frequency coefficients and an improved quantization strategy that preserves more details in the output.

For entropy coding, VP6 uses various techniques based on complexity and/or overall frame size, including VLC and context modeled binary coding.

To achieve any requested data rate, the codec chooses automatically to adjust quantization levels, adjust encoded frame dimensions, or drop frames altogether.

VP6 vs H.264 and Conclusions

For what we have said, it is clear that VP6 is much simplier than H.264. And this is its main advantage. VP6 requires less complexslity both in decoding and encoding stage. Therefore the

<Sn>

codec is also considerably smaller.

Being simpler than H.264, how can VP6 obtain encoding performance comparable to it ?

We must consider that VP6 uses fewer but efficient encoding techniques. A mix of adaptive sub-pixel motion estimation, better prediction of low-order frequency coefficients, improved quantization strategy, de-blocking and de-ringing filters or enhanced context based entropy coding, produces compressed movies with a very good "perceptual" quality.

Furthermore, we have already described the difference between a codec technology and a codec implementation. H.264 has a lot of modes, options and annexs, and to built an efficient encoder is quite difficult. VP6 is probably technically less advanced than H.264 but On2 has produced a quite efficient and compact implementation.