Next Generation VP8 Video codecs for Mobile Multimedia Communications:

Basava Raju S, Dr B Siva Kumar (Prof and HOD of TE Dept., Dr AIT Bangalore)

Email: rajhunsur@yahoo.co.in, sivabs2000@yahoo.co.uk

***Abstract*— In this article, we give an overview of several core Technologies involved in the next generation mobile media communications system from both the media codecs and media transport perspectives. Here we introduce H264 and VP8 video codecs that are suitable for mobile communications, including a low complexity and low bit rate codec for video conferencing and generic scalable video coding. Video codec has been widely used on PC's with relatively strong capability. However mobile devices, such as Pocket PCs and Handheld PCs still suffer from weak computational power, short battery lifetime and limited display capability and good quality of Video. Regarding this there is very much need of practically low complexity real time video codec for mobile devices. So that here implementing Google VP8 and H264 baseline profile and comparing to get better results for mobile applications. And here several methods that can significantly reduce the computational cost are adopted in these codecs. H264 and VP8 video Codecs are the most widely-accepted video standard in recent years, here analysing performance of the both codec's and try to improving the quality of image for mobile applications. Both of these video decoders are very much necessary for today's entertainment and competitive mobile applications world. The explosion of mobile device market has caused an increase in the need for fast and low-power applications like video encoding, decoding, and image manipulation.**

***Keywords*— PCs, Codec's, H 264, VP8, Mobile applications, Bit rate, next generation multimedia communications.**

I. INTRODUCTION

Video compression is used to exploit limited storage and transmission capacity as efficiently as possible, which is important for the Internet, mobile applications and high definition media so that Both H 264 and VP8 codecs analysis as given here

VP8 decoder: Google has recently released the video compression format VP8 to the open source community. This new compression format competes against the existing H.264 video standard developed by the ITU-T Video Coding Experts Group (VCEG) in collaboration with the ISO/IEC Moving Picture Experts Group (MPEG). This paper compares these two video coding standards in terms of video bit rate-distortion (quality) performance and the video network traffic variability with different long video sequences. The main motivation was to avoid license fees for H.264 based products that will begin in 2011 for Google's products, such as the Chrome browser or youTube. In order to make this video codec widespread and increase adoption possibilities of VP8 as the default HTML5 video standard, Google open- sourced the formerly closed-source developed video codec.

And also Inheriting many great innovations from its predecessors (VP7 and VP6) such as golden frames, processor-adaptive real-time encoding and a low-complexity loop filter, VP8 adds more than fifty new techniques to achieve its goal of outstanding quality at low bitrates, with very low complexity and VP8 specifies exact values for reconstructed pixels. This greatly facilitates the verification of the correctness of a decoder implementation as well as avoiding difficult-to-predict visual incongruities between such implementations.

VP8 offers both VBR (variable bit rate) and CBR (constant bit rate) encoding options. CBR attempts to keep the bit rate more constant, i.e. the codec tries to remain within given buffering constraints. If the user sets CBR mode but gives very loose buffer restrictions, then the result will start to resemble VBR.

Mainly VP8 is an open source video compression format supported by consortium of technology companies. This paper provides a technical overview of the format, with an emphasis on its unique features. This paper also discusses how these features benefit VP8 in achieving high compression efficiency and low decoding complexity at the same time. This will helpful for mobile communication applications.

II PROCEDURE:

A. *Review Stage:*

From the very beginning of VP8's development, the developers were focused on Internet/web-based video applications. This focus has led to a number of basic assumptions in VP8's overall design:

Low bandwidth requirement: One of the basic design assumptions is that for the foreseeable future, available network bandwidth will be limited. With this assumption, VP8 was specifically designed to operate mainly in quality range from "watchable video" (~30dB in the PSNR metric) to "visually lossless" (~45dB).

Heterogeneous client hardware: There is a broad spectrum of client hardware connected to the web, ranging from low power mobile and embedded devices to the most advanced desktop computers with many processor cores. It must, therefore, be range from "watchable video" (~30dB in the PSNR.

Hybrid transform with adaptive quantization: VP8 uses 4x4 block-based discrete cosine transform (DCT) for all luma and chroma residual signal. Depending on the prediction mode, the DC coefficients from a 16x16 macro block may then undergo a 4x4 Walsh-Hadamard transform.
However, 3 differences between VP8′s scheme and H.264′s.

The first is that the 8×8 transform is omitted entirely (fitting with the omission of the i8x8 intra mode). The second is the specifics of the transform itself. This will not good for mobile communication applications H.264 uses an extremely simplified "DCT" which is so un-DCT-like that it often referred to as the HCT (H.264 Cosine Transform) instead. This simplified transform results in roughly 1% worse compression, but greatly simplifies the transform itself, which can be implemented entirely with adds, subtracts, and right shifts by 1. VP8 uses an extremely, needlessly accurate version that uses very large multiplies (20091 and 35468).The third difference is that the Hadamard hierarchical transform is applied for some *inter* blocks, not merely i16x16. In particular, it also runs for p16x16 blocks. While this is definitely a good idea, especially given the small transform size (and the need to decor relate the DC value between the small transforms).This will effect especially at high resolutions.. The one good new idea here is applying the hierarchical DC transform to inter blocks.

For quantisation, the core process is basically the same among all MPEG-like video formats, and VP8 is no exception**.** The primary ways that video formats tend to differentiate themselves here is by varying quantisation scaling factors. There are two ways in which this is primarily done: frame-based offsets that apply to all coefficients or just some portion of them, and macro block-level offsets. VP8 primarily uses the former; in a scheme much less flexible than H.264′s custom quantisation matrices, it allows for adjusting the quantiser of luma DC, luma AC, chroma DC, and so forth, separately. The latter (macro block-level quantiser choice) can, in theory, be done using its "segmentation map" features, albeit very hackly and not very efficiently.

Flexible reference frames: VP8 uses three types of reference frames for inter prediction: the "last frame", a "golden frame" (one frame worth of decompressed data from the arbitrarily distant past) and an "alternate reference frame." Overall, this design has a much smaller memory footprint on both encoder and decoder than designs with many more reference frames.

Golden Reference Frame: We have found experimentally that it is very rare for more than three reference frames to provide significant quality benefit, but the undesirable increase in memory footprint from the extra reference frames is substantial. And very often, the most beneficial reference frames are not the last three frames encoded. Depending on content, a frame from the distant past can be very beneficial in terms of inter prediction when objects re-appear after disappearing for a number of frames. Based on such observations, VP8 was designed to use one reference frame buffer to store a video frame from an arbitrary point in the past. This buffer is known as the "Golden Reference Frame." The format also defines a number of flags in the bit stream to notify a decoder when and how to update this buffer.

VP8 encoders can use the Golden Reference Frame in many ways to improve coding efficiency. It can be used to maintain a copy of the background when there are objects moving in the foreground, so that occluded regions can be easily and cheaply reconstructed when a foreground object moves away. Together with the last reference frame, the Golden Reference Frame may also be used to create a background sprite. Such an arrangement is helpful to compression efficiency in many video scenes. Another use of the golden frame is the coding of back and forth cut of two scenes, where the golden frame buffer can be used to maintain a copy of the second scene. Finally, the golden frame can also be used for error recovery in a real-time video conference, or even in a multi-party video conference for scalability

Alternate (Constructed) Reference Frame: Unlike other types of reference frames used in video compression, which are always displayed to the user by the decoder, the VP8 alternate reference frame is decoded normally but may or may not be shown in the decoder. It can be used solely as a reference to improve inters prediction for other coded frames. Because alternate reference frames have the option of not being displayed, VP8 encoders can use them to transmit any data that is helpful to compression. For example, a VP8 encoder can construct one alternate reference frame from multiple source frames, or it can create an alternate reference frame using different macro blocks from many different video frames. The flexibility in VP8 specification allows many types of usage of the alternate reference frame for improving coding efficiency. Here are two illustrative examples:

Noise-Reduced Prediction: The alternate reference frame is transmitted and decoded similarly to other frames; hence its usage does not increase computational complexity in the decoder. However, in off-line applications the VP8 encoder is free to use more sophisticated processing to create them. One application of the alternate reference frame is for noise-reduced prediction. In this application, the VP8 encoder uses multiple input source frames to construct one reference frame through temporal or spatial noise filtering. This "noise-free" alternate reference frame is then used to improve prediction for encoding subsequent frames.

Improving Prediction without B-Frames: The lack of B frames has led to discussion in the research community about VP8's ability to achieve high compression efficiency. The VP8 format, however, supports intelligent use of the golden reference and the alternate reference frames together to compensate for this. The VP8 encoder can choose to transmit an alternate reference frame assembled with content from many "future" frames using sophisticated filtering. Encoding of subsequent frames can then make use of information from the past (last frame and golden frame) and from the future

(alternate reference frame). Effectively, this helps the encoder to achieve compression efficiency without requiring frame reordering in the decoder.

Efficient intra prediction and inter prediction: VP8 makes extensive uses of intra and inter prediction. VP8's intra prediction features a new "TM_PRED" mode as one of the many simple and effective intra prediction methods. For inter prediction, VP8 features a flexible "SPLITMV" mode capable of coding arbitrary block patterns within a macro block.

VP8 uses two classes of prediction modes: Intra prediction uses data within a single video frame, and Inter prediction uses data from previously encoded frames.

VP8 Intra Prediction Modes: VP8 intra prediction modes are used with three types of blocks:

- 4x4 luma
- 16x16 luma
- 8x8 chroma

Four common intra prediction modes are shared by these blocks:

- H_PRED (horizontal prediction): Fills each column of the block with a copy of the left column
- V_PRED (vertical prediction): Fills each row of the block with a copy of the above row.
- DC_PRED (DC prediction): Fills the block with a single value using the average of the pixels in the row above and the column to the left
- TM_PRED (True Motion prediction): In addition to the row and column , TM_PRED uses the pixel above and to the left of the block. Horizontal differences between pixels and vertical differences between pixels are propagated to form the prediction block. For 4x4 luma blocks, there are six additional intra modes corresponding to predicting pixels in different directions. As mentioned above, the TM_PRED mode is unique to VP8.

VP8 Inter Prediction Modes: In VP8, inter prediction modes are used on inter frames (non-key frames). For any VP8 inter frame, there are typically three Previously coded reference frames that can be used for prediction. A typical inter prediction block is constructed using a motion vector to copy a block from one of the three frames. The motion vector points to the location of a pixel block to be copied. In video compression schemes, a good portion of the bits is spent on encoding motion vectors; the portion can be especially large for video encoded at lower data rates. VP8 provides efficient motion vector coding by reusing vectors from neighbouring macro blocks. For example, the prediction modes "NEAREST" and "NEAR" make use of last and second-to-last, non-zero motion vectors from neighbouring macro blocks. These inter prediction modes can be used in combination with any of the three different reference frames.

In addition, VP8 has a sophisticated, flexible inter prediction mode called SPLITMV. This mode was designed to enable flexible partitioning of a macro block into sub-blocks to achieve better inter prediction. SPLITMV is useful when objects within a macro block have different motion characteristics. Within a macro block coded using the SPLITMV mode, each sub-block can have its own motion vector. Similar to the strategy of reusing without transmitting motion vectors at the macro block level, a sub-block can also use motion vectors from neighbouring sub-blocks above or left of the current block without transmitting the motion vectors.

High performance sub-pixel interpolation: VP8's motion compensation uses quarter pixel accurate motion vectors for luma pixels. The sub-pixel interpolation of VP8 features a single-stage interpolation process and a set of high performance six-tap interpolation filters. The filter taps used for The six tap filters are:

[3, -16, 77, 77, -16, 3]/128 for ½ pixel positions

[2, -11, 108, 36, -8, 1]/128 for ¼ pixel positions

[1, -8, 36, 108, -11, 2]/128 for ¾ pixel positions

Chroma motion vectors in VP8 are calculated from their luma counterparts by averaging motion vectors within a macro block, and have up to one eighth pixel accuracy. VP8 uses four-tap bicubic filters for the 1/8, 3/8, 5/8 and 7/8 pixel positions. Overall, the VP8 interpolation filtering process achieves optimal frequency response with high computation efficiency.

Adaptive in-loop deblocking filtering: Loop filtering is a process of removing blocking artifacts introduced by quantization of the DCT coefficients from block transforms. VP8 brings several loop-filtering innovations that speed up decoding by not applying any loop filter at all in some situations. VP8 also supports a method of implicit segmentation

Where different loop filter strengths can be applied for different parts of the image, according to the prediction **modes** or reference frames used to encode each macro block. For example it would be possible to apply stronger filtering to intra-coded blocks and at the same time specify that inter coded blocks that use the Golden Frame as a reference and are coded using a (0,0) motion vector should use a weaker filter. The choice of loop filter strengths in a variety of situations is fully adjustable on a frame-by-frame basis, so the encoder can adapt the filtering strategy in order to get the best possible results. In addition, similar to the region-based adaptive quantization in section 3, VP8 supports the adjustment of loop filter strength for each segment. Fig. 4 shows an example where the encoder can adapt the filtering strength based on content.

Frame level adaptive entropy coding: VP8 uses binary arithmetic coding extensively for almost all data values except a few header bits. Entropy contexts are adaptive at the frame

level, striking a balance between compression efficiency and computational complexity.

Parallel processing friendly data partitioning: VP8 can pack entropy coded transform coefficients into multiple partitions, to facilitate parallel processing in decoders. This design improves decoder performance on multi-core processors, with close to zero impact to compression efficiency and no impact to decoding performance on single core processors.

So that this analysis will helpful for mobile communication applications in terms of bit rate wise.

III. PERFORMANCE COMPARISON

Fig. 1: H.264 encoder block diagram.

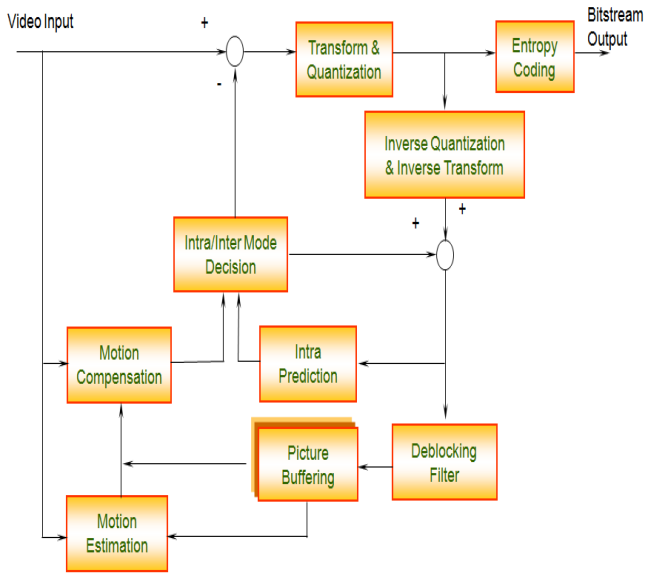


Fig. 2: H.264 decoder block diagram.

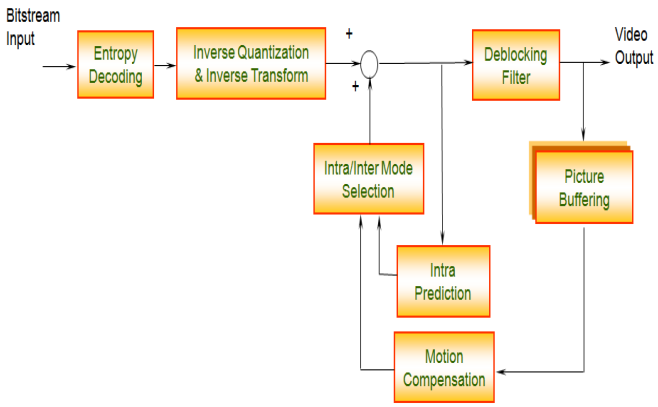
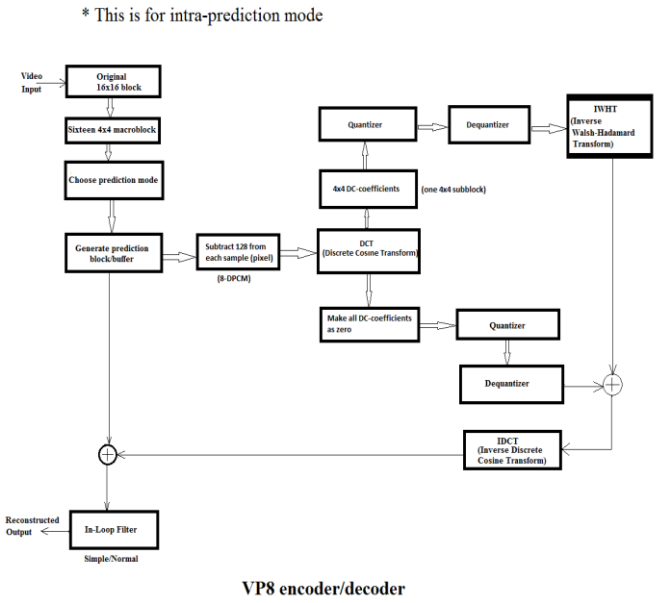


Fig 3: VP8 Encoder and Decoder diagram



Quantisation Parameter
tion Parameter
Parameter
er

Quantisation Parameter	Encoding time (sec)	Bit Rates (Kbits/secs)	Compression Ratio
2	0.234	2393.4	42.38
8	0.140	323.9	311.4
20	0.47	124.3	810

ABOVE Table –1: H.264 - calculation for hall_cif.yuv (90 frames):

And Below Fig 4 respective image:

Fig. 4: **H.264 - hall_cif.yuv**

Below Table –2: VP8 - calculation for hall_cif.yuv (90 frames):

And **Fig 5** respective image:

Quantisation Parameter	Encoding time (sec)	Bit Rates (Kbits/secs)	Compression Ratio
2	3.931	5179.5	19.19
8	2.761	2036.5	49.77
20	2.028	693.6	146.06

Fig 6:
Bit rate v/s compression ratio for hall_cif.yuv (90 frames)

Fig. 5 VP8 - hall_cif.yuv

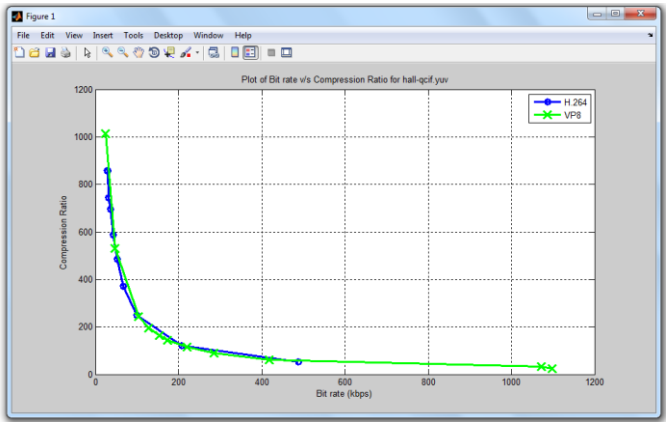


Table –3 H.264 - calculation for hall_qcif.yuv (90 frames):

And **Fig 7** respective image:

Quantisation Parameter	Encoding time (sec)	Bit Rates (Kbits/secs)	Compression Ratio
2	0.047	488.6	51.8
8	0.047	100.2	
20	2.028	693.6	146.06

Fig 7: H.264 - hall_qcif.yuv

Table –4 VP8 - calculation for hall_qcif.yuv (90 frames):

And **Fig 9** respective image:

Quantisation Parameter	Encoding time (sec)	Bit Rates (Kbits/secs)	Compression Ratio
2	0.733	1098.3	23.6
8	0.546	418.5	60.53
20	0.479	173.7	144.65

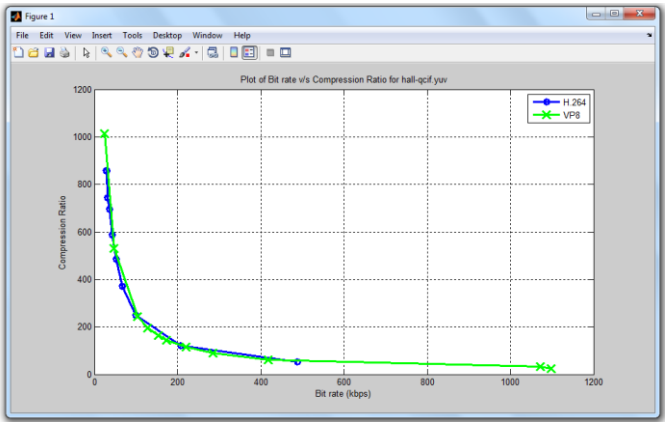


Fig 8 shows
Bit rate v/s compression ratio for hall_qcif.yuv (90 frames)

Fig 9: VP8 - hall_qcif.yuv

IV. CONCLUSIONS

As a result of the many advanced coding features, VP8 can make the best use of computation power in modern hardware for improving compression efficiency while maintaining fast decoding speed on majority devices. Figure shows the decoding speed test results on two different hardware platforms for video files encoded in VP8 and H.264 at similar bitrates.

Therefore the decoding speeds may reflect the intrinsic decoding complexity. As shown in Fig. 6, the decoding speeds of VP8 encoded files are consistently faster, average around 30%, than those of H.264 encoded files at a similar bitrate across the two different hardware platforms.

It is not difficult to conclude from the test results that, in the designed operating range of web video, VP8 can achieve Compression efficiency that is competitive to the best H.264/AVC encoder available. At the same time, however, the low complexity design of the VP8 format enables decoder implementations to achieve much faster decoding speeds than H.264/AVC on various platforms.

Comparing the compression ratios v/s bit rate it shows that H.264 and VP8 have similar performance except VP8 has a slight edge over H.264 at lower bit rates. The main reason being use of golden frames in real time low bit rate applications. And Comparing the encoding times it shows that H.264 encoder is almost 15-20 times faster then VP8 encoder. These are early stages of VP8 development and constant upgrading of VP8 encoder is in progress. The main reason for slow encoder is lack of B-frames (bipredictive) in VP8.

REFERENCES

[1] A. Puri, X. Chen and A. Luthra, "Video coding using the H.264/MPEG-4 AVC compression standard", Signal Processing: Image Communication, vol. 19, pp. 793-849, Oct. 2004

[2] S. K. Kwon, A. Tamhankar and K. R. Rao, "Overview of H.264/MPEG-4 Part 10" J. Visual Communication and Imag Representation, vol. 17, pp.186-216, Apr. 2006.

[3] VP8 Decoder -Google

[4] http://multimedia.cx/eggs/vp8-the-savior-codec/ - VP8: The Savior Codec.

[5] http://multimedia.cx/eggs/vp8-transform-and-quantization/ - VP8 encoder and decoder explanation.

[6] *http://iphome.hhi.de/suehring/tml/ - JM software source code*

[7] http://www.webmproject.org/code/- Explore the WebM Source Code for VP8.

[8] J. Bankoski, P. Wilkins, Y. Xu, "VP8 Data Format and Decoding Guide," http://www.ietf.org/internet-drafts/draftbankoski-vp8-bitstream-01.txt, Jan 2011.

[9] [9] J. Bankoski, "On2's Truemotion VP7 video codec and golden frames", EE Times, Jul 2008.