Abstract: The high definition video adoption has been growing rapidly for the last two years. The Blu-ray high-definition format has mandated MPEG-2, H.264 and VC-1 as video compression formats. The co-existence of these different video coding standards creates a need for transcoding. In this paper, an efficient transcoding algorithm from H.264 video to VC-1 video is discussed. The proposed architecture, which considers I and P frames, is a low complexity transcoder which re-uses mode decisions, macroblock partition sizes and motion vectors, obtained from the incoming H.264 coded video bitstream, to make intelligent and fast decisions in the transcoding process. The simulation results show comparable video quality as that of the cascaded architecture with 80% lesser encoding time. The proposed architecture thus offers a real-time solution for mobile applications and devices with less power and memory.
Vidhya Vijayakumar hails from Chennai (Madras), capital city of the southern state of Tamil Nadu in India. She received her bachelor’s degree in Electronics and Communication from Sathyabama Deemed University, Chennai in May 2005. After her bachelors, she worked in the Digital Signal Processing and Multimedia group at Wipro technologies. To pursue her interest in academia, she later joined the graduate program at University of Texas at Arlington (UT Arlington). During her stay at UT Arlington, she was a member of the Multimedia Processing Lab, headed by Dr. K.R. Rao. She did her internship with Adobe Inc working in their Advanced Technology Lab. She worked as a Graduate Researcher for the Multimedia laboratory, headed by Dr. Ishfaq Ahmad. She was also awarded the Dean’s Masters Scholarship at UT Arlington for the year 2008-10 for academic excellence. Her research interests are in the field of video compression, transcoding and wireless communications.
LOW COMPLEXITY H.264 TO VC-1 TRANSCODER

Vidhya Vijayakumar and K.R. Rao, EE Dept, University of Texas at Arlington
Box 19016, Arlington, TX 76019, USA

ABSTRACT

The high definition video adoption has been growing rapidly for the last two years. The blue ray high definition format has mandated MPEG-2, H.264 and VC-1 as video compression formats. The co-existence of these different video coding standards creates a need for transcoding. In this paper, an efficient transcoding algorithm from H.264 video to VC-1 video is discussed. The proposed architecture, which considers I and P frames, is a low complexity transcoder which re-uses mode decisions, macroblock partition sizes and motion vectors, obtained from the incoming H.264 coded video bitstream, to make intelligent and fast decisions in the transcoding process. The simulation results show comparable video quality as that of the cascaded architecture with 80% lesser encoding time. The proposed architecture thus offers a real time solution for mobile applications and devices with less power and memory.

1. INTRODUCTION

The high definition video adoption has been growing rapidly for the last two years. The blue ray high definition format has mandated MPEG-2, H.264 and VC-1 as video compression formats [1] [2]. H.264 is an emerging standard that is replacing MPEG-2 for digital video applications and is, presently, the state of the art codec. The VC-1 standard is derived from Microsoft’s proprietary WMV-9 [1] which is widely used on the Internet. VC-1 is a pure video compression technology developed by Microsoft, and is deployed as a key engine in satellite TV, IP set-tops and high-definition DVD recorders. The coexistence of these different video coding standards creates a need for transcoding. Another strong motivation for transcoding H.264 video to VC-1 video comes from the fact that VC-1 produces quality comparable video streams at the same bit rates as H.264 but is significantly less complex than H.264 [1]. In this paper, an efficient transcoding algorithm from H.264 video to VC-1 video is discussed. For H.264, mode decision for Intra MB and Inter MB is computationally intensive since each of these modes has to be checked to select the best coding mode. The key idea of the paper is to propose an efficient re-use of information in VC-1 encoding based on incoming H.264 encoded bitstreams. If mode decision and motion estimation take advantage of already encoded H.264 bitstreams, this will save a lot of computation in such MBs that would, otherwise, undergo different mode checks in the VC-1 encoding stage.

The field of mobile video and wireless video communication demands a better video coding technology at low bit rates and lowest computational complexities without any loss of visual quality. Although H.264 offers good efficiency in terms of quality, its computational complexity, makes its implementation difficult for mobile and low power applications. Thus there is a need to use less complex codecs like VC-1 and thereby creating a need for transcoding H.264 to VC-1. The emergence of H.264 has resulted in increasing research in the area of transcoding to and from H.264 format. Transcoding tools and algorithms have been proposed to transcode video from H.263 [3], MPEG-4 [4], MPEG-2 [5] and VC-1 [6] to the H.264 format. Using the proposed H.264 to VC-1 transcoder, we can achieve indirect transcoding from standards like H.263, MPEG-4 and MPEG-2 to VC-1 in combination with existing algorithms described in [3], [4] and [5].

A complete decode followed by a low complex encode is proposed in this paper. The techniques proposed attempt to reuse the information gathered during the decoding stage to improve the transcoder performance. The transform domain approaches have limited applicability due to the fact that H.264 and VC-1 have different transform types and sizes. The use of variable block size motion estimation and deblocking filter also make transform domain transcoding impractical for full transcoding applications. The other approach is pixel domain transcoding. For example, in MPEG-2 to H.264 transcoding, MPEG-2 video is fully decoded followed by an accelerated H.264 encoding stage that uses information gathered during the
decoding stage. This approach has shown promising results and is reported in several papers [7], [8] and [9]. Hence this approach is chosen to transcode H.264 to VC-1.

While there has been recent work on MPEG-2 to H.264 [10], VC-1 to H.264 [6] transcoding, the published work on H.264 to VC-1 transcoding is non-existent. There is very limited amount of published work on VC-1 and no work has been published on H.264 to VC-1 transcoding. This paper gives a brief overview of H.264 and VC-1 and discusses the opportunities for low-complexity tools for H.264 to VC-1 transcoding. The rest of the paper is organized as follows: Section 2 gives an overview of the H.264 coding tools; Section 3 gives an overview of the VC-1 coding tools; Section 4 details the proposed algorithm and Section 5 discusses the experiments and results.

2. OVERVIEW OF H.264

This section presents an overview of H.264/AVC. H.264/AVC is the newest video coding standard [11] of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). The main goals of the H.264/AVC standardization efforts are enhanced compression performance and provision of a network friendly video representation addressing conversational (video telephony) and non conversational (storage, broadcast, or streaming) applications. H.264/AVC has achieved a significant improvement in rate-distortion efficiency relative to existing standards. Like previous video coding standards (MPEG-4, H263 etc), Advanced video coding (AVC) [12] is also based on hybrid block-based motion compensation and transform-coding model, but it supports a lot of additional and enhanced tools. Figure 1 and Figure 2 illustrate the encoder and decoder of H.264/AVC respectively.

H.264/AVC contains a number of new features that allow it to compress video much more efficiently than older standards and to provide more flexibility for application to a wide variety of network environments. In particular, some such key features include:
Multi-picture inter-picture prediction including the following features:

- Using previously-encoded pictures as references, allowing up to 16 reference frames (or 32 reference fields, in the case of interlaced encoding)
- Variable block-size motion compensation (VBSMC) with block sizes as large as 16×16 and as small as 4×4, enabling precise segmentation of moving regions. The supported luma prediction block sizes include 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4, many of which can be used together in a single macroblock. Chroma prediction block sizes are correspondingly smaller according to the chroma sub sampling in use.
- The ability to use multiple motion vectors per macroblock (one or two per partition) with a maximum of 32 in the case of a B macroblock constructed of 16 4×4 partitions. The motion vectors for each 8×8 or larger partition region can point to different reference pictures.
- The ability to use any macroblock type in B-frames, including I-macroblocks, resulting in much more efficient encoding when using B-frames.
- Six-tap filtering for derivation of half-pel luma sample predictions, for sharper sub pixel motion-compensation. Quarter-pixel motion is derived by linear interpolation of the half pel values, to save processing power.
- Quarter-pixel precision for motion compensation, enabling precise description of the displacements of moving areas. For chroma the resolution is typically halved both vertically and horizontally (4:2:0) therefore the motion compensation of chroma uses one-eighth chroma pixel grid units.
- Weighted prediction, allowing an encoder to specify the use of a scaling and offset when performing motion compensation, and providing a significant benefit in performance in special cases—such as fade-to-black, fade-in, and cross-fade transitions. This includes implicit weighted prediction for B-frames, and explicit weighted prediction for P-frames.

Spatial prediction from the edges of neighboring blocks for intra coding. This includes luma prediction block sizes of 16×16, 8×8, and 4×4 (of which only one type can be used within each macroblock).

Flexible interlaced-scan video coding features

New transform design features, including:

- An exact-match integer 4×4 spatial block transform.
- An exact-match integer 8×8 spatial block transform.
- Adaptive encoder selection between the 4×4 and 8×8 transform block sizes for the integer transform operation.
- A secondary Hadamard transform performed on DC coefficients of the primary spatial transform applied to chroma DC coefficients (and also luma in intra 16x16 case) to obtain even more compression in smooth regions.

A quantization design including:

- Logarithmic step size control for easier bit rate management by encoders and simplified inverse-quantization scaling
- Frequency-customized quantization scaling matrices selected by the encoder for perceptual-based quantization optimization

An in-loop deblocking filter that helps prevent the blocking artifacts common to other DCT-based image compression techniques, resulting in better visual appearance and compression efficiency

An entropy coding design including:

- Context-adaptive binary arithmetic coding (CABAC)
- Context-adaptive variable-length coding (CAVLC)
- A common simple and highly structured variable length coding (VLC) technique for many of the syntax elements not coded by CABAC or CAVLC, referred to as Exponential-Golomb coding (or Exp-Golomb).

Support of monochrome, 4:2:0, 4:2:2, and 4:4:4 chroma sampling (depending on the selected profile).

Support of sample bit depth precision ranging from 8 to 14 bits per sample (depending on the selected profile).
These techniques, along with several others, help H.264 to perform significantly better than any prior standard 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 the bit rates or less, especially in high bit rate and high resolution situations. The enhanced performance of H.264 comes at the price of being very complex.

3. OVERVIEW OF VC-1

This section provides a brief overview of VC-1 [1] with emphasis on the features that impact transcoding. Like all MPEG standards, VC-1 is based on motion compensated transform coding. There is no fixed GOP structure in VC-1. I, P, B and Skipped P are defined as pictures/frames. Unlike MPEG standards, I (Intra) frame does not have to occur periodically. Therefore, if there is no big scene change for a lengthy period of time, there could be only P frames in the sequence after the first I frame. Unlike H.264, B frames cannot be used as reference frames. Skipped P frame is signaled when the frame is exactly the same as the previous reference. In I frames, no intra-prediction is used. For Intra-coded MBs (such as in I frames), only 8x8 transform size is used. For Inter-coded MBs (such as in P/ B frames), 4 transform sizes – 8x8, 4x8, 8x4, 4x4 – are potentially used on the residual data. Transform block size can change adaptively in P/ B frames with 4 different size options, while block size for motion compensation is either 16x16 or 8x8 in VC-1. Note that this is quite the opposite to that of H.264. H.264 normally uses fixed size 4x4 transform with variable block size prediction for motion compensation. The transforms are 16 bit transforms where both the sums and the products of two 16 bit values produce results within 16 bits – the inverse transform can be implemented in 16 bit fixed point arithmetic. Note that the transform approximates a DCT, and norms of basis functions between transforms are identical to enable the same quantization scheme through various transform types. Figure 3 and Figure 4 illustrate the encoder and decoder of VC-1 respectively.

![Figure 3 VC-1 encoder block diagram [1]](image1)

![Figure 4 VC-1 decoder block diagram [1]](image2)
VC-1 supports a few options for motion compensation:

1) Half-pel or quarter-pel resolution motion compensation can be used.
2) Bi-cubic or bi-linear filter can be used for the interpolation.
3) 16x16 or 8x8 block size can be used.

Only some combinations of such options can be defined to signal at the frame level. Quantization is generally defined with two parameters in video standards – Quantization parameter (QP) and Dead zone. The QP varies from 1 to 31, while there are two choices for Dead-zone in VC-1 – 3Qp and 5Qp. In I frames, primary quantization parameter is applied to all the MBs. However, differential quantization parameter is used to adaptively describe QP in each MB in P/B frames. Another QP usage option is to use only two QPs for an entire frame depending on the MB positions – either boundary MB or non-boundary MB. There are two techniques used in VC-1 to reduce blocky effects around transform boundary – Overlapped Transform (OLT) smoothing and In Loop deblocking Filtering (ILF). OLT and ILF are performed on reference frames I and P. Thus, the result of filtering affects only the quality of following pictures that use overlap transformed and/or in loop filtered frames as references.

4. H.264 TO VC-1 TRANSCODING

For H.264 to VC-1 standards transcoding, it is required to implement several changes in order to accommodate the mismatches between the two standards. For instance, for motion estimation and compensation, H.264 supports 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4 macroblock partitions, but VC-1 supports 16x16 and 8x8 only. The transform size and type - 8x8 and 4x4 in H.264 and 8x8, 4x8, 8x4 and 4x4 in VC-1 are different and make transform domain transcoding prohibitively complex. Hence, the use of frequency domain transcoders is poor for heterogeneous transcoding and pixel domain transcoding is preferred.

The transcoding algorithms discussed in this paper assume full H.264 decoding down to the pixel level, followed by a reduced complexity VC-1 encoding. The data gathered during the H.264 decoding stage is used to accelerate the VC-1 encoding stage. It is assumed that the H.264 encoded bitstreams were generated with an R-D optimized encoder. Table 1 shows a comparison of the VC-1 and H.264 features from a transcoding point of view. The picture coding types used are similar. The transform size and type are different resulting in transform domain transcoding prohibitively complex. The semantics of intra MBs are similar except for the intra prediction allowed in H.264 and the mixed inter MBs in VC-1. The inter prediction has significant differences including the block size of MC, block size of transform, and reference frames used. These similarities between the codecs can be exploited in reducing the transcoding complexity. Figure 5 illustrates the proposed transcoder architecture.

![Figure 5 Proposed transcoder architecture](image-url)
<table>
<thead>
<tr>
<th>Feature</th>
<th>H.264 Baseline</th>
<th>VC-1 Simple</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picture type</td>
<td>I, P</td>
<td>I, P</td>
</tr>
<tr>
<td>Transform Size</td>
<td>4x4</td>
<td>4x4, 4x8, 8x4, 8x8</td>
</tr>
<tr>
<td>Transform</td>
<td>Integer DCT</td>
<td>Integer DCT</td>
</tr>
<tr>
<td>Intra Prediction</td>
<td>4x4, 16x16 spatial, IPCM</td>
<td>Frequency domain DC and AC Prediction</td>
</tr>
<tr>
<td>Motion Compensation Block Size</td>
<td>16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4</td>
<td>16x16, 8x8</td>
</tr>
<tr>
<td>Total MB Modes</td>
<td>7 inter + (9 + 4) intra</td>
<td>3</td>
</tr>
<tr>
<td>Motion Vector resolution</td>
<td>¼ pixel</td>
<td>¼ pixel</td>
</tr>
<tr>
<td>In loop filter</td>
<td>Deblocking</td>
<td>Deblocking, Overlap transform</td>
</tr>
<tr>
<td>Reference Frames</td>
<td>Single, Multiple</td>
<td>Single</td>
</tr>
<tr>
<td>Entropy coding</td>
<td>CAVLC</td>
<td>Adaptive VLC</td>
</tr>
</tbody>
</table>

Table 1: Comparison of H.264 baseline and VC-1 simple profile tools

4.1. Intra MB Mode Mapping

An intra MB in the incoming H.264 bitstream is coded as a VC-1 intra MB. An H.264 intra MB can be coded as Intra 4x4 (9 different directional modes) or Intra 16x16 (4 different modes). But a VC-1 intra MB has four 8x8 blocks and has no intra prediction modes. Figure 6 illustrates the nine intra 4x4 directional modes available in H.264/AVC. Figure 7 illustrate the intra 4x4 and intra 16x16 directional modes specified in H.264/AVC.

Figure 6 4x4 Luma prediction (intra-prediction) modes in H.264 [2]
Since intra MB in VC-1 uses an 8x8 transform, irrespective of the block size (16x16 or 4x4) in H.264, we need not carry over the information of the intra prediction type in H.264. Table 2 shows the proposed intra MB mapping.

<table>
<thead>
<tr>
<th>H.264 Intra MB</th>
<th>VC-1 Intra MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intra 16x16 (Any mode)</td>
<td>Intra MB 8x8</td>
</tr>
<tr>
<td>Intra 4x4 (Any mode)</td>
<td>Intra MB 8x8</td>
</tr>
</tbody>
</table>

Table 2 H.264 and VC-1 Intra MB mapping

4.2. Inter MB Mode Mapping

An inter coded MB in the incoming H.264 bitstream is coded as inter MB in VC-1. The inter MB in H.264 has 7 different motion compensation sizes – 16x16, 16x8, 8x16, 8x8, 4x8, 8x4, 4x4. The inter MB in VC-1 has 2 different motion compensation sizes 16x16 and 8x8. Another significant difference is that H.264 uses 4x4 (and 8x8 in fidelity range extensions) transform sizes where as VC-1 uses 4 different transform sizes – 8x8, 4x8, 8x4 and 4x4. The 16x16 motion compensation sizes are usually selected in H.264 for areas that are relatively uniform and will be mapped to inter 16x16 MB in VC-1 with a transform size of 8x8. Motion compensation sizes 8x16, 16x8 have small non-uniform motion and hence they are mapped to inter 8x8 MB in VC-1 since 16x16 MB size will yield worse quality due to the non-uniform motion. Using the selected H.264 block size as a measure of homogeneity in the block, the transform size is determined and applied in VC-1. In other words, the H.264 block size determines the transform size used for that particular block. This method eliminates the need to compute the half sum and half difference values of each 8x8 block to determine the transform size. Figure 8 and Figure 9 illustrate the block sizes used in motion estimation in H.264 and VC-1 respectively.

![Block sizes and macroblock partitioning in H.264 for inter prediction](image)

(a) (L-R) 16x16, 8x16, 16x8, 8x8 blocks

(b) (L-R) 8x8, 4x8, 8x4, 4x4 blocks
The 8x8, 8x4, 4x8 and 4x4 modes are usually selected in H.264 for areas that have non-uniform motion. The 16x16 mode in VC-1 is eliminated for such non-uniform MBs. The MB is then mapped to an 8x8 block size in VC-1 with the H.264 block size determining the transform size to be used in VC-1.

Table 3 describes the decision making for mapping the inter MBs and the type of transform to be used in VC-1.

<table>
<thead>
<tr>
<th>H.264 Inter MB</th>
<th>VC-1 Inter MB</th>
<th>Transform size in VC-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter 16x16</td>
<td>Inter 16x16</td>
<td>8x8</td>
</tr>
<tr>
<td>Inter 16x8</td>
<td>Inter 8x8</td>
<td>8x4</td>
</tr>
<tr>
<td>Inter 8x16</td>
<td>Inter 8x8</td>
<td>4x8</td>
</tr>
<tr>
<td>Inter 8x8</td>
<td>Inter 8x8</td>
<td>8x8</td>
</tr>
<tr>
<td>Inter 4x8</td>
<td>Inter 8x8</td>
<td>4x8</td>
</tr>
<tr>
<td>Inter 8x4</td>
<td>Inter 8x8</td>
<td>8x4</td>
</tr>
<tr>
<td>Inter 4x4</td>
<td>Inter 8x8</td>
<td>4x4</td>
</tr>
</tbody>
</table>

Table 3 H.264 and VC-1 Inter MB mapping and VC-1 transform type

4.3. Motion vector mapping

Re-use of motion vectors selected in H.264 can significantly reduce the complexity of VC-1 encoding. Since the proposed transcoder does format conversion from H.264 to VC-1, median motion vectors [13] [14] are selected when there are more than 2 motion vectors in the incoming H.264 macroblock and average of the motion vectors when there are 2 motion vectors. Table 4 describes the selection of motion vectors.

<table>
<thead>
<tr>
<th>H.264 Inter MB</th>
<th>VC-1 Inter MB</th>
<th>Motion Vector Re-use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter 16x16</td>
<td>Inter 16x16</td>
<td>Same motion vectors for 16x16 block</td>
</tr>
<tr>
<td>Inter 16x8</td>
<td>Inter 8x8</td>
<td>Average of motion vectors for each 8x8 block</td>
</tr>
<tr>
<td>Inter 8x16</td>
<td>Inter 8x8</td>
<td>Average of motion vectors for each 8x8 block</td>
</tr>
<tr>
<td>Inter 8x8</td>
<td>Inter 8x8</td>
<td>Same motion vectors for each 8x8 block</td>
</tr>
<tr>
<td>Inter 4x8</td>
<td>Inter 8x8</td>
<td>Median of motion vectors for each 8x8 block</td>
</tr>
<tr>
<td>Inter 8x4</td>
<td>Inter 8x8</td>
<td>Median of motion vectors for each 8x8 block</td>
</tr>
<tr>
<td>Inter 4x4</td>
<td>Inter 8x8</td>
<td>Median of motion vectors for each 8x8 block</td>
</tr>
</tbody>
</table>

Table 4 H.264 and VC-1 Inter MB motion vector mapping

4.4. Reference Pictures:

The H.264/AVC standard defines the use of up to sixteen reference pictures for motion estimation, while VC-1 uses only one or two, according to the slice type P or B respectively. The reuse of motion vectors implies using the same reference pictures to maintain their meaning. Since the profiles considered are baseline profile for H.264 (single
reference picture) and simple profile for VC-1 (single reference picture), the same reference picture as the incoming bitstream is used and no re-scaling of motion vectors is required.

4.5. Skipped Macroblock

When a skipped macro block is signaled in the bit stream, no further data is sent for that macro block. The mode conversion of skipped macroblocks in H.264 to skipped macroblocks in VC-1 is a straight forward process. Since the skipped macro block definition of both standards is fully compatible, a direct conversion is possible.

4.6. Extraction of re-usable information

The H.264 bitstream has the macroblock type, sub-macroblock type, reference picture index and motion vectors (if applicable). These details are extracted out of the bitstream. This information is used while encoding in VC-1. H.264 bitstream contains information about

1. Macroblock type
   - P16x16 – P block type 16x16
   - P16x8 – P block type 16x8
   - P8x16 – P block type 8x16
   - P8x8 – P block type 8x8
   - I4MB – I block type 4x4
   - I16MB – I block type 16x16
2. Macroblock sub block type
   - SMB8x8 – sub macroblock type 8x8
   - SMB8x4 – sub macroblock type 8x4
   - SMB4x8 – sub macroblock type 4x8
   - SMB4x4 – sub macroblock type 4x4
3. Reference picture index
4. Motion vector x, y

5. IMPLEMENTATION & RESULTS

5.1. Implementation

The implementation of the transcoder consists of two steps; modify the H.264 decoder to output the required information and then modify the VC-1 encoder to re-use this information. The H.264 codec used was the joint module implementation of the H.264 standard [15] and the VC-1 software [16] was the SMPTE implementation. Both the codecs are implemented in the C programming language.

5.2. Results

Three test sequences are chosen for testing and evaluation purposes. They are Akiyo (low motion QCIF sequence), Foreman (medium-high motion, CIF sequence) and Football (high motion, CIF sequence).

5.3. Comparison of proposed transcoder with cascaded transcoder with respect to H.264 video

The performance of the proposed transcoder is compared against the performance of cascaded transcoder architecture as there is no previous work that can be compared with. The results are divided into the comparison of quality and comparison of complexity.

5.3.1. Rate Distortion Analysis

A comparison of the PSNR and SSIM [17] between the proposed architecture and the cascaded re-encoding transcoder is illustrated for various quantization parameters (QP). The reference video used for all the comparison purposes is the H.264 decoded video. This is the
available video to compare in real time applications when the original video may or may not be available.

Figure 10 Transcoder Results - Akiyo, QCIF, Comparison of Y peak signal to noise ratio, cascade transcoder Vs proposed transcoder

Figure 11 Transcoder Results - Akiyo, QCIF, Comparison of Y structural similarity index, cascade transcoder Vs proposed transcoder
Figure 12 Akiyo sequence (a) Original (b) H.264 decoded (c) Reference cascade decoded at QP 10 (d) Proposed transcoder decoded at QP 10

Figure 13 Transcoder Results – Foreman, CIF, Comparison of Y peak signal to noise ratio, cascade transcoder Vs proposed transcoder
Figure 14 Transcoder Results – Foreman, CIF, Comparison of Y structural similarity index, cascade transcoder Vs proposed transcoder

Figure 15 Foreman sequence (a) Original (b) H.264 decoded (c) Reference cascade decoded at QP 10 (d) Proposed transcoder decoded at QP 10
Figure 16 Transcoder Results – Football, CIF, Comparison of Y peak signal to noise ratio, cascade transcoder Vs proposed transcoder

Figure 17 Transcoder Results – Football, CIF, Comparison of Y structural similarity index, cascade transcoder Vs proposed transcoder
5.3.2. Complexity Analysis

A comparison of the encoding time between the proposed architecture and the cascaded re-encoding transcoder is illustrated for various quantization parameters (QP).

Figure 18 Football sequence (a) Original (b) H.264 decoded (c) Reference cascade decoded at QP 10 (d) Proposed transcoder decoded at QP 10
5.4. Conclusions

This paper addresses an important problem of transcoding H.264 to VC-1 coding format. Both VC-1 and H.264 are hybrid video coding algorithms that exploit motion compensation and transform coding. The H.264 coded video has enough similarities with VC-1 to enable
reduced complexity transcoding. The paper proposes reducing the MB coding mode and motion estimation complexity in VC-1 by re-using the motion vectors from the H.264 encoded bit stream. Also, the variable transform size used in VC-1 is determined by the H.264 block partition. The quality vs. complexity tradeoffs are addressed using a low cost design for the transcoder. The proposed transcoder performs comparable to the cascade architecture in terms of quality and is 70% less complex.

REFERENCES

[16] VC-1 SMPTE software http://store.smpte.org/category-s/30.htm