SCALABLE CODING OF H.264 VIDEO

by

KEMAL UGUR

B.Sc., Middle East Technical University, 2001

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

ELECTRICAL AND COMPUTER ENGINEERING

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

June 2004

Library Authorization

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

KEMAL UGUR — 16/06/2004

Name of Author *(please print)* — Date (dd/mm/yyyy)

Title of Thesis: Scalable Coding of H.264 Video

Degree: Master of Applied Science Year: 2004

Department of Electrical and Computer Engineering

The University of British Columbia
Vancouver, BC Canada

Abstract

Real-time transmission of digital video over media, such as the Internet and wireless networks has recently been receiving much attention. A big challenge of video transmission over such networks is the variation of available bandwidth over time. Traditional video coding standards whose main objective is to optimize the quality of transmitted video at a given bitrate, do not offer effective solutions to the bandwidth variation problem. To deal with this problem, different scalable video coding techniques have been developed.

The latest video coding standard, H.264, provides superior compression efficiency over all previous standards. This standard, however, does not include tools for coding the video in a scalable fashion. In this thesis, we introduce methods that allow encoding and transmitting of H.264 video in a scalable fashion. The method we propose is an adaptation of the existing MPEG-4 Fine Granular Scalability structure (FGS) to the H.264 standard. Our proposed algorithm minimizes the added number of the bits needed in adapting the advanced features of H.264 to the FGS system. Our proposed system has the advantages of being highly error resilient and having low computational complexity.

Due to its structure, the FGS standard has low coding efficiency when compared to single layer coding. To overcome this problem, we also introduce a hybrid method that combines our proposed H.264 based FGS approach with the stream-switching approach employed in the H.264 standard. By combining different techniques, our proposed system offers a complete solution for all kinds of applications. The proposed system outperforms existing systems by offering optimum bandwidth utilization and improved video quality for the end user.

Contents

Table of Figures

Acknowledgements

I would like to thank my research supervisors Dr. Panos Nasiopoulos and Dr. Rabab Ward for the valuable guidance and support they provided throughout my research. I am grateful to them for the fruitful discussions we had about research and life in general that will guide me through the rest of my life. I would also like to thank to my dear friends. Without them, I would not have enjoyed my days in Canada. I especially want to thank my friends Emre, Doruk, Caglar, Ari, Juan and my friends at Koerner's for being the reasons of my relaxed, happy and stress-free times in Vancouver. I want to thank Sebnem, for her dear support during the last stages of this thesis. I want to thank my colleagues at Image Processing Lab. All of you have brilliant skills, and I hope the future will be bright for you. I want to thank Dr. Mehran Azimi, for his support and kindness as the administrator of our lab during my studies. Finally, I would like to thank my beloved family, my mother Ayfer, my father Avni, and my sister Burcu for their continuous support of my graduate studies. I spent the last couple of years, the years my sister needed me the most, away from home pursuing graduate degree. I wish I was able to see her growing up, but I hope these times spent abroad will be worth it in the future. I dedicate this thesis to my beloved sister, Burcu.

CHAPTER I

1 Introduction

1.1 Thesis Objective

In networks used for video transmission environment, such as wireless networks and Internet, the available bandwidth for video transmission is not constant but varies over time. This variation in the available bandwidth possesses a problem for a video transmission system. Traditional video coding standards, whose objective is to optimize the quality of the video at a given bitrate, cannot cope with this bandwidth variation problem effectively. Scalable Video Coding techniques have been developed to more efficiently address this bandwidth variation problem.

Scalable Video Coding (SVC) is a video coding framework that enables a system to adapt the quality of the video sequence to the underlying channel's available bandwidth. Unlike traditional video coding standards, the objective of scalable video coding is to optimize the video quality *over a bitrate range* instead of *at a given bitrate* as the bandwidth available for each user can change over time according to the characteristics of each channel.

All popular video coding standards, such as MPEG-2 and MPEG-4, include some scalability tools. The latest video coding standard, H.264, provides superior compression efficiency over all previous standards, but it does not include tools for coding the video in a scalable fashion.

In this work, we introduce scalability to H.264 so that it can be more efficiently used in network environments with time varying bandwidth. We use the latest scalable video coding standard, Fine Granular Scalability (FGS) that is originally developed for MPEG-

4 and adapt it to H.264. We chose FGS to introduce scalability for H.264, as FGS has low implementation complexity and it is highly flexible. The research is based on already established industry standards that are proven to be superior to other methods. We modify the techniques present in FGS and include novel techniques, so that the proposed scalable H.264 solution has low complexity, high coding efficiency and high error-resiliency.

We also introduce a hybrid method that combines the FGS approach with the stream-switching approach employed in the H.264 standard. By combining different techniques our proposed system offers a complete solution for all kinds of applications. The proposed system outperforms existing systems by offering optimum bandwidth utilization and improved video quality for the end user.

1.2 Thesis Outline

In the remaining part of this chapter we first present the necessary background information on fundamentals of video coding, scalable video coding and different types of scalable video coding techniques (Section 1.3). Following that we present an overview of MPEG-4 FGS scalable video coding standard in Section 1.4. An overview on the H.264 video coding standard is presented in Section 1.5.

In Chapter 2, we present the proposed scalable H.264 based FGS structure. Following that, our proposed novel H.264 based FGS structure is presented. In Chapter 3, we present our proposed approach that further incorporates the highly efficient features present in H.264 with flexible structure of FGS. This approach combines the stream-switching structure of H.264 with our H.264 based FGS structure to provide an overall highly efficient and flexible system.

Chapter 4 presents the conclusions of the research together with suggestions for future work.

1.3 Introduction to Video Coding

Digital video applications have been growing tremendously in the past few years. Such applications include DVD-video (digital versatile disk), digital cable and direct broadcast systems (DBS), videophone and videoconferencing. In addition to these applications, recent advances have made Multimedia Messaging Service (MMS) over wireless networks, and high quality video streaming [1] over the Internet possible. The main driving reason behind all these applications is the advances in efficient representation of the digital video data using advanced video coding methods. Video coding is being used wherever digital video communications, processing, acquisition and reproduction occur.

The need for video coding is clear when one considers the amount of storage space or transmission bandwidth required for raw (uncompressed) video data. Consider a video program having a resolution of 720x480 pixels (a common resolution used in DVD), which is to be played at 25 frames-per-second (standard in PAL/SECAM). The bitrate of this video with three color components at 8 bits per pixel will be over 200 Mbits/s! To store this video in current DVD discs, compression by a factor of at least 200 is required. A similar requirement also holds when a digital video transmission scenario is considered. In summary, it is clear that efficient video coding is needed for feasible video transmission and storage. This need was realized by international standards organizations resulting in several standards for digital video coding, such as ISO/IEC MPEG-2[2] and ITU-T H.263[4]. In the next subsection, common techniques used in video coding standards are presented.

1.3.1 Fundamentals of Video Coding

Video coding can be viewed as coding of a sequence of images; in other words, image coding with a temporal component. Therefore, similar techniques used for image coding can be applied for video coding as well. Image coding techniques essentially exploit the statistical redundancy in the spatial domain to achieve high compression ratios. Spatial redundancy exists in images because of the high correlation between the brightness and color of a given pixel, and the brightness and color of the nearby pixels within the same picture. Techniques used to exploit the spatial redundancies are often referred to as *intra*-coding methods.

The most popular image coding standards are transform-based [6]. In transform coding, the raw image is divided into blocks and a transform is applied to each image block to compact the signal energy into a smaller number of coefficients. The coefficients of the transformed blocks are quantized and the quantized values are entropy coded to form the image bitstream.

In addition to spatial redundancy in each picture for a typical video sequence, there also exists a temporal redundancy between consecutive pictures. This is due to the fact that pictures are sampled in very short time intervals (such as 40 ms. for a 25 frame-per-second sequence) and the picture content usually changes slightly in this small amount of time. Exploiting temporal redundancy is referred to as *inter-frame-coding* in video coding terminology.

To remove the temporal redundancies in a video sequence, all popular video coding standards [2, 4] use Motion Compensated Prediction (MCP). MCP-based coders allow information about motion between frames to be transmitted as side information in the output video bitstream. Generally, MCP consists of two stages. The first stage estimates

the motion between the current encoded frame and a reference frame where reference is one of the previously reconstructed frames. This first stage is generally referred to as *motion estimation* (ME). The second stage creates a prediction for the current frame using the estimated motion parameters and the previous reconstructed frames. This stage is referred as *motion compensation* (MC).

Video coders that use both intra and inter-frame coding to achieve high compression ratios are called *hybrid video coders* and form the basis of all popular video coding standards. Block diagram of a basic hybrid-video coder is illustrated in Figure 1. This hybrid video encoder uses MCP to remove the temporal redundancies and transform coding to remove the spatial redundancies. After the redundancies are removed, the resulting signal is quantized, and then entropy coded to obtain the output video bitstream. The details of the coding process are explained below.

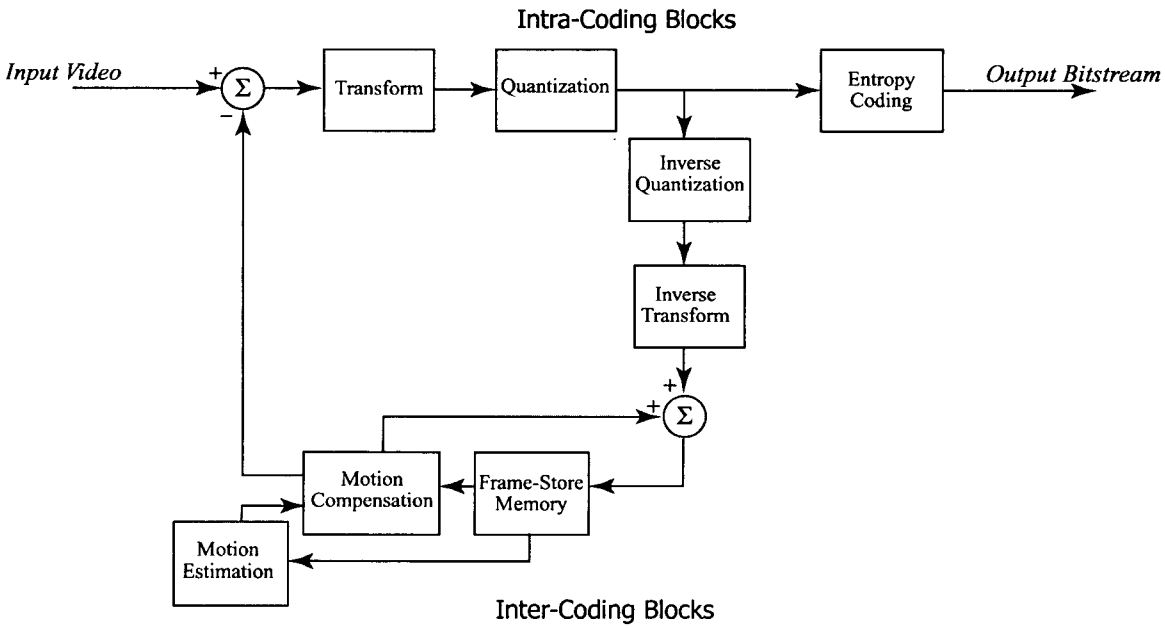


Figure 1 Hybrid video coder block diagram

1.3.1.1 Overview of Motion Compensated Prediction (MCP)

MCP is the essential technique used in video coders to remove the temporal redundancy present between frames of a video sequence. In the output video bitstream, motion information between frames are transmitted as side information. MCP can be analyzed in two stages, the motion estimation stage, and the motion compensation stage. The motion estimation's role is to find the best prediction for the current frame from a reference frame, using a specified *motion model*. For the motion estimation process, several motion models have been presented in literature such as pixel-recursive [9] and variable size block matching [11], but the translational block-matching motion estimation is the most widely adopted technique due to its simplicity and good performance. In this model, the current frame to be coded is divided into blocks, and for each block a best block match is searched in the reference frame. The spatial position of the best matching block is used to calculate the motion vector for the current block. This motion estimation process is illustrated in Figure 2.

The motion compensation stage forms the prediction for the current frame using the reference frame and the obtained motion vector information. The difference between the obtained prediction and the current frame is called the *prediction error*. This error is due to two assumptions implied in the motion compensation model. Firstly, it is assumed that all the pixels within the block undergo the same motion. Secondly the block's motion is assumed to be translational. This prediction error is then coded using transform based spatial coding methods. The coded prediction error and the motion information together form the output video bitstream.

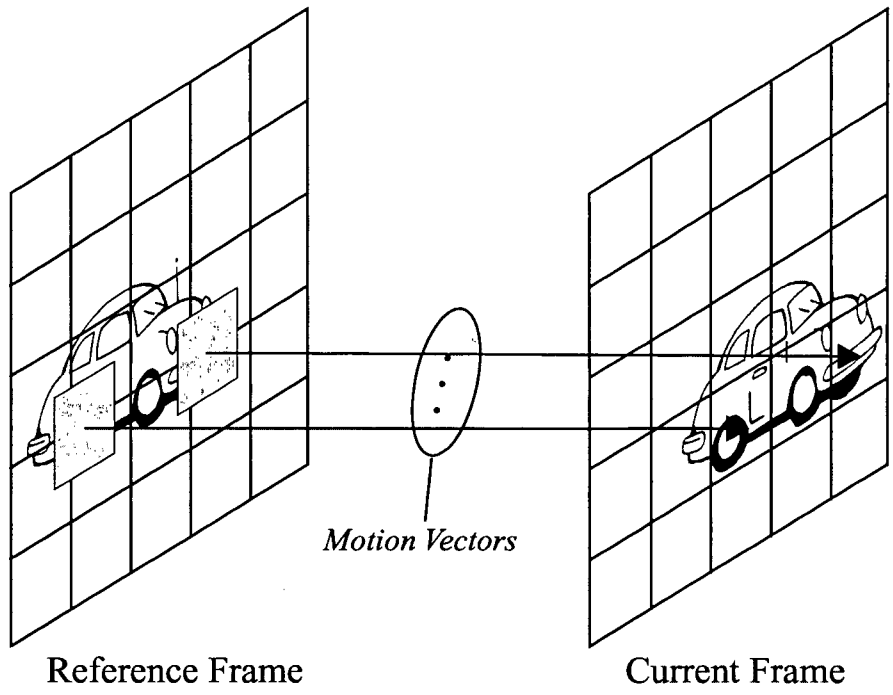


Figure 2 Illustration of Block Matching Motion Estimation

There are three types of frames classified according to which reference frames they use in the motion estimation stage. These are the *intra coded* (**I**) frames, the *predictive coded* (**P**) frames and the *bidirectionally predictive coded* (**B**) frames. For the **I** frames, MCP is not performed and the whole frame is intra-coded. The first frame of the video sequence has to be coded as an **I** frame, as there is no reference frames available at the start of coding hence MCP can not be performed. If the **I** frames are placed periodically in the bitstream, the decoder has the capability of random access to the video sequence. Thus, fast forward of the video sequence can be achieved by only decoding and displaying **I** frames. Also random access is very important in digital TV broadcast as viewers may change from one video program they are watching, to another one at anytime [7]. Because **I** frames do not exploit temporal redundancy their coding efficiency is low.

MCP is used to code the **P** and **B** frames. **P** frames are coded using prediction from the last **I** or **P** frame, whichever happens to be closer. This kind of prediction is called *forward prediction,* as the reference frame occurs temporally before the current frame. The coding efficiency of **P** frames is significantly higher than that of **I** frames, due to the MCP process involved. Besides forward prediction, **B** frames also use backward prediction where the reference frame occurs temporally after the current frame. Higher coding efficiency is achieved by using both the past and future frames as reference. Note that a **B** frame is not used for predicting any other frame. This makes it more tolerant to errors, as any error in its encoding will not propagate to other frames by the prediction process. Furthermore, B frames can be coded using a lower quality than that of the reference pictures, resulting in further bit savings [8]. Because a B frame uses a reference that may be temporally subsequent, that reference frame should be coded and made available prior to coding the B frame. Therefore, the display order and the coding order of frames are different. Figure 3 illustrates this difference in a typical coded video bitstream.

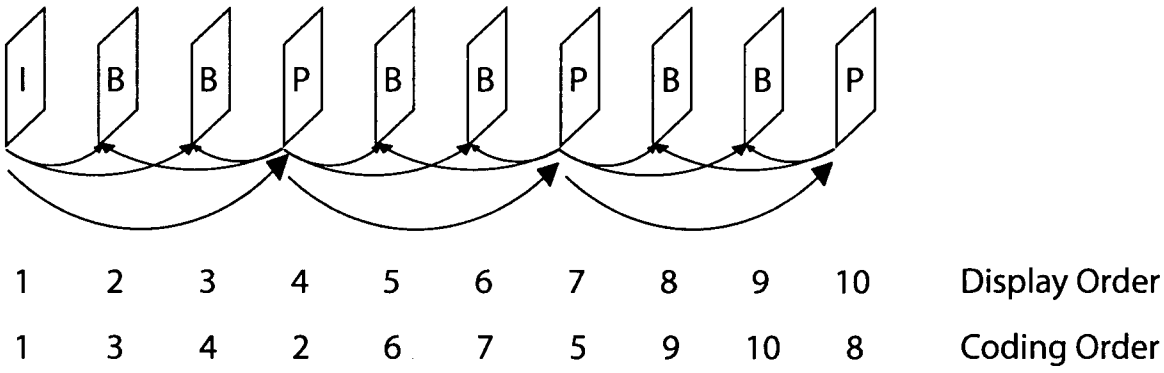


Figure 3 Coding and display order of Frames in a typical video bitstream

1.3.1.2 Transform Coding

After the motion compensated prediction (MCP) process is completed and a prediction is formed for the current frame, this prediction is subtracted from the current original frame to form the residual signal. The temporal redundancy is reduced at the MCP stage, but there is still spatial redundancy present in the residual signal. The most widely used method to exploit the spatial redundancy is the transform coding, in which a transform is applied to the residual signal to decorrelate the signal and compact its energy into smaller number of coefficients. After the signal is decorrelated, the resulting coefficients are entropy coded.

The best transform that gives the best energy compaction results is the Karhunel-Loeve transform (KLT) [10]. The rows of the KLT consist of the eigenvectors of the autocorrelation matrix of the input signal. The autocorrelation matrix for a random process X is a matrix whose *(i,j)*th element $[R]_{i,j}$ is given by

$$[R]_{i,j}=E[X_n X_{n+|i-j|}]$$

It can be shown that this transform minimizes the geometric mean of the variance of the transform coefficients [7]. However, this transform is data dependent and it must be recomputed for every input signal, if the input signal is non-stationary. This makes KLT unpractical for video coding. The Discrete Cosine Transform (DCT) is the most widely adopted transform in image and video coding standards. DCT is a suitable approximation to KLT and is data independent.

DCT gets its name from the fact that rows of the NxN transform matrix C are obtained as a function of cosines. In video coding, DCT is applied to an 8x8 block data and the transform is given as:

$$[C]_{i,j} = \begin{cases} \sqrt{\frac{1}{N}} \cos\frac{(2j+1)i\pi}{2N} & i = 0, j = 0,\ldots, N-1 \\ \sqrt{\frac{2}{N}} \cos\frac{(2j+1)i\pi}{2N} & i = 1,2,\ldots, N-1, j = 0,1,\ldots N-1 \end{cases}$$

For Markov sources with high correlation coefficient, the compaction ability of DCT is very close to that of KLT [7]. As video and image can be modeled as a highly correlated Markov sources, DCT is chosen to be part of the many video and image coding standards.

Because DCT is defined in terms of floating-point values, its implementation on digital processors is not efficient. Also, the floating-point nature of DCT introduces a mismatch between the decoded data in the encoder and the decoder. This error causes degradation in the quality of the decoded video. Because of these drawbacks, H.264 standard replaced the popular DCT with a low complexity 4x4 transform specified with integer arithmetic. The transform matrix H is designed as:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

It should be noted that, the rows of this transform is orthogonal, but do not have the same norm. This difference in norm is compensated in the quantization stage.

The implementation of this transform on digital processors is very efficient as computing its direct and inverse transform could be carried with only additions and shifts, no multiplications [13]. Also, it is observed that the smaller block size used decreases some artifacts known as ringing and that occur at low bitrates [15].

1.3.1.3 Quantization and Entropy Coding

The quantization stage of the video coder creates a lossy representation of the input. The quantization process divides the transform coefficients by a quantization parameter and then rounds them to the nearest integer. The quantization parameter determines the quality loss and the amount of bit savings. High values for the quantization parameter result in more loss of information and a decrease of video quality but achieves a higher compression ratio. Smaller values result in decrease of information loss, which in turn increase the output video quality but at the expense of smaller compression ratios.

The resultant quantized transform coefficients are zigzag ordered and then assembled into a one dimensional array using a zigzag pattern, as illustrated in Figure 4. The first coefficient placed in the one dimensional array, is the DC coefficient of the block. The DC coefficient is followed by AC coefficients ordered roughly from low frequency to high frequency. The assembled one dimensional array is coded using "run-level" coding. The number of consecutive zeros before a nonzero DCT coefficient is called a "run" and the absolute value of the nonzero DCT coefficient is called a "level".

Entropy coding is the last stage of the video coding process. In this stage the "run-level" symbols are coded in a lossless fashion along with the motion vectors and side information. During entropy coding, the input symbols are mapped to binary variable length *codewords*. The symbols that occur more frequently are represented with less number of bits whereas more bits are used for symbols that occur not very often.

There are different types of entropy coding methods with different methods to generate the codewords. The most common techniques used in video compression are Huffman coding and arithmetic coding.

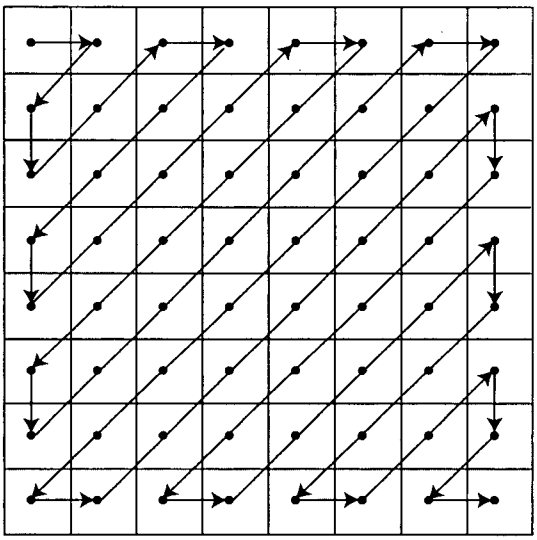


Figure 4 Zigzag Scan Order for DCT Coefficients

1.3.2 Overview of Scalable Video Coding

With the emergence of broadband wireless networks, wireless video transmission has been receiving great attention. At the same time, streaming of audiovisual content over the Internet is emerging as an important application. The primary challenge of transmitting video over wireless media and the Internet is the random fluctuations in the bandwidth available for each user [1]. In order to deliver the best visual quality to each user, video coding technologies need to deal with the problems created by bandwidth variations.

Scalable Video Coding (SVC) is a video coding framework that aims to cope with the bandwidth variation problem. It enables the streaming system to adapt the quality of the video sequence to the underlying channel's available bandwidth.

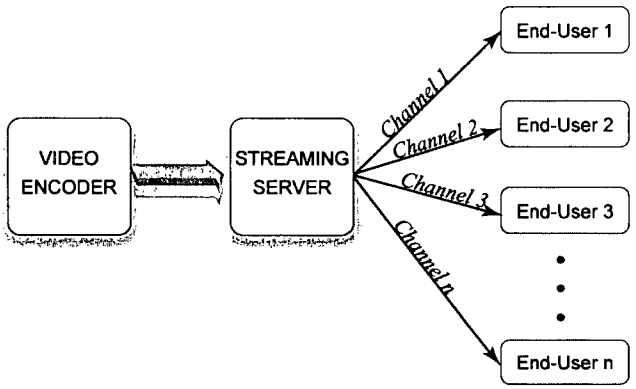


Figure 5 Architecture of a Streaming Video System

A typical system configuration for the next generation networked video applications is illustrated in Figure 5. In this configuration, video encoding takes place before the data are transmitted to the streaming server. For this reason, at encoding time, the bandwidth available for the video sequence to be streamed is not known. Also, the bandwidth available for each user can change dynamically according to the characteristics of each channel. As a result, the video encoder can not know the bitrate the video quality should be optimized at. Because of this uncertainty in the streaming bitrate, the objective of video coding for networked video is to optimize the video quality *over a bitrate range* instead of *at a given bitrate* [2].

Previous video coding standards (such as MPEG-2) include several layered scalable techniques. In layered scalable coding techniques, a video sequence is coded into a base layer and an enhancement layer. If the decoder receives only the base layer, the video sequence is reconstructed with a minimal quality. If the enhancement layer is also received by the decoder, the reconstructed video quality is increased. For layered scalable coding techniques, the enhancement layer stream must be completely received by the decoder, otherwise the video quality is not enhanced. The three different techniques for

layered scalable video coding are: signal-to-noise ratio (SNR) scalability, temporal scalability and spatial scalability. In the next three subsections, a brief overview of each of these different techniques is presented.

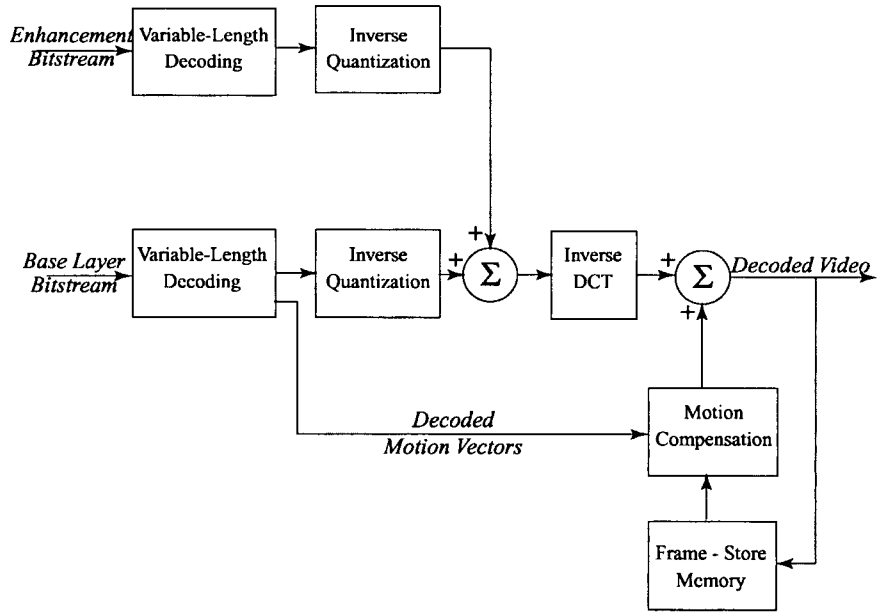


Figure 6 Block Diagram of MPEG-2 SNR Scalable Decoder

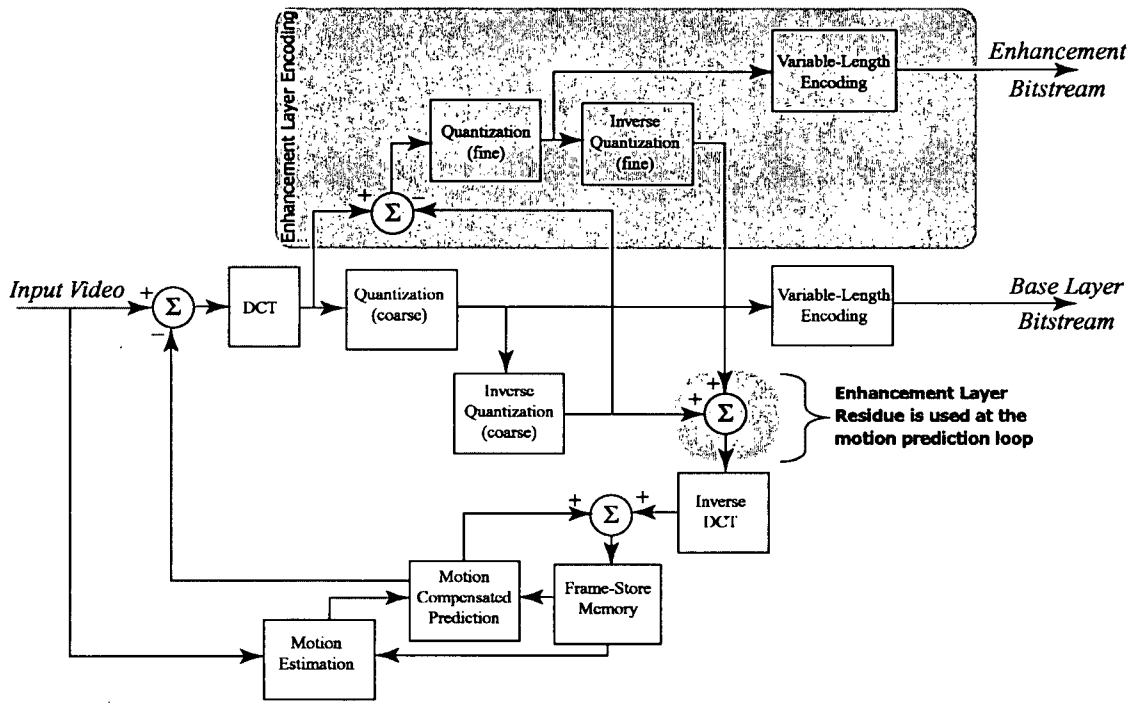
1.3.2.1 Layered SNR Scalability

SNR Scalability refers to the technique that codes the video sequence into two layers at the same frame rate and the same spatial resolution but with different quantization levels. Figure 6 shows the two-layer SNR scalable decoder, in the MPEG-2 video standard. The Variable Length Decoding block decodes the base layer bitstream. The decoded information includes the motion vectors and the quantized Discrete Cosine Transform (DCT) coefficients. The quantized DCT coefficients are reconstructed by inverse quantization. Similarly, the enhancement bitstream is decoded in the Variable Length

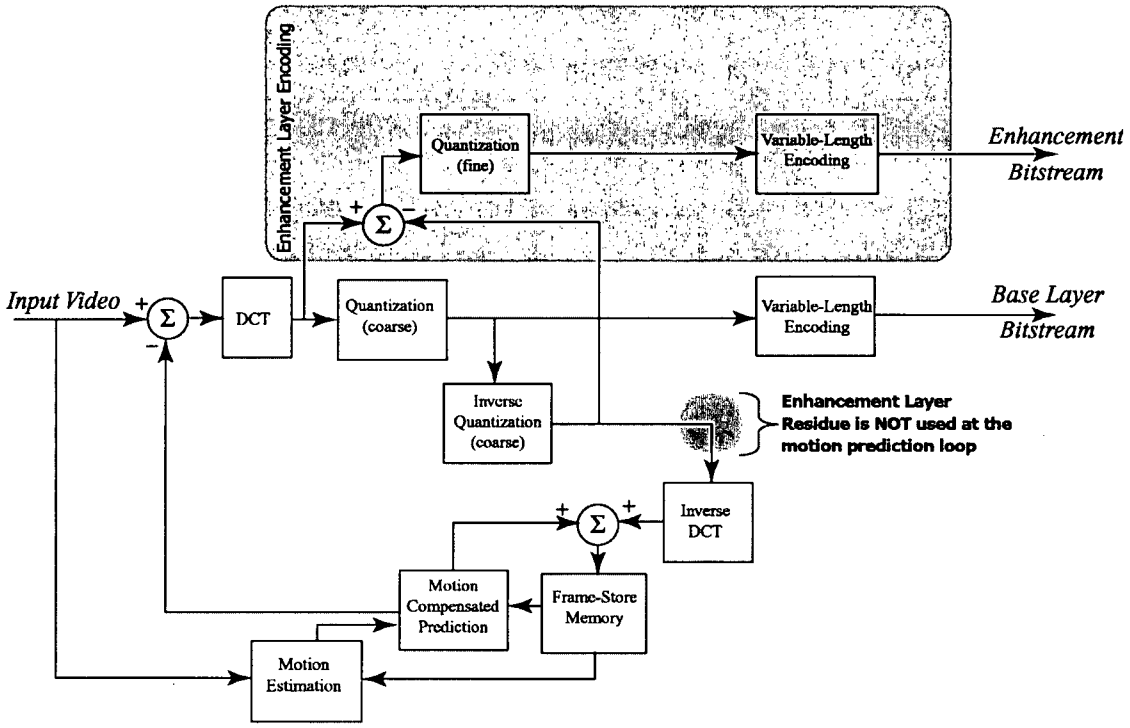
Decoding block and the residual DCT coefficients are then reconstructed by inverse quantization. The reconstructed residual DCT coefficients are added to the base layer reconstructed DCT coefficients to obtain the higher accuracy DCT coefficients. The inverse DCT is then applied on the higher accuracy DCT coefficients to obtain the image-domain difference frame. The motion compensated frame is added to the image-domain difference frames to form the decoded sequence.

The SNR scalable decoder is standardized in MPEG-2 and uses the enhancement layer residue information in the motion compensation loop. However, MPEG-2 does not standardize how the scalable encoder generates the base and enhancement layer streams. Depending on whether or not the encoder uses the enhancement layer information in the motion prediction, the coding efficiency of the base and enhancement layer may change.

Two standard compliant encoders are illustrated in Figure 7a Figure 7b. In these encoders, motion compensated prediction is formed using the reconstructed picture held in the frame store memory. This prediction is then subtracted from the original video and the prediction difference is formed. The latter is DCT transformed and then quantized using a high quantization parameter (coarse quantization – low quality). The base layer bitstream is formed by variable length coding of the quantized DCT coefficients. In the feedback path of the encoder, the quantized coefficients are reconstructed using inverse quantization with the same high quantization parameter. The enhancement layer residue is formed by taking the difference between the original prediction error DCT coefficients and the base layer reconstructed DCT coefficients. The enhancement layer residue is quantized using a smaller quantization parameter (fine quantization – high quality) and variable length coded to produce the enhancement layer bitstream.


Enhancement Layer Encoding
Variable-Length Encoding
Enhancement Bitstream
Quantization (fine)
Inverse Quantization (fine)
Input Video
DCT
Quantization (coarse)
Variable-Length Encoding
Base Layer Bitstream
Inverse Quantization (coarse)
Enhancement Layer Residue is used at the motion prediction loop
Inverse DCT
Motion Compensated Prediction
Frame-Store Memory
Motion Estimation

(a)


Enhancement Layer Encoding
Quantization (fine)
Variable-Length Encoding
Enhancement Bitstream
Input Video
DCT
Quantization (coarse)
Variable-Length Encoding
Base Layer Bitstream
Inverse Quantization (coarse)
Enhancement Layer Residue is NOT used at the motion prediction loop
Inverse DCT
Motion Compensated Prediction
Frame-Store Memory
Motion Estimation

(b)

Figure 7 Block Diagrams of two types of SNR Scalable Encoders (a) Enhancement Layer Residue is used at the motion prediction loop at the base layer (b) Enhancement Layer Residue is not used

The inverse quantized values produced in the encoding of the enhancement layer are then added to inverse quantized values at the base layer in the feedback loop. The reconstructed frames are formed by applying the inverse DCT and are stored in the frame store memory. The reconstructed frames stored at the memory of the encoder are identical to the frames stored in the SNR Scalable decoder's memory. However, for the case where a decoder does not receive the enhancement layer bitstream, the reconstructed frames at the decoder side and the encoder side will not be the same. This is because the decoder will only use the base layer information to form the reconstruction, whereas the encoder had used both the base and the enhancement layers information. The mismatch in the reconstructed encoder and decoder frames causes errors to accumulate in the decoded base-layer video sequence. This error is called *drift*. The drift problem decreases the coding efficiency of the base layer video. On the other hand, the high quality reference frames used at the motion compensated prediction increases the coding efficiency when the decoder receives and decodes the enhancement layer as well. Hence, the SNR scalable encoder results in *low coding efficiency* for the base layer, but *high coding efficiency* for the enhancement layer.

The encoder illustrated in Figure 7b only uses the base layer information to form the prediction. In this case, the drift problem at the base layer is removed. However, if the decoder receives and decodes the enhancement layer, a drift will also occur due to a similar mismatch between the reconstructed encoded and decoded frames. Therefore, for

this SNR scalable encoder, the base layer coding efficiency is high, but the enhancement layer coding efficiency is low due to the drift problem.

To summarize, for the layered Scalable SNR Decoder standardized in MPEG-2, there are two possibilities, namely, either the base layer has a poor performance to ensure a good performance for the enhancement layer, or the enhancement layer has a poor performance to ensure a good performance for the base layer.

1.3.2.2 Temporal Scalability

In the layered temporal scalability, video is coded into two layers at the same spatial resolution but at different frame rates. If the decoder receives and decodes only the base layer, the video sequence is displayed at a low frame rate. The enhancement layer fills the missing frames and upon decoding, the video can be displayed at a higher frame rate. Several techniques are used for temporal scalable coding [3]. Figure 8 shows a possible frame structure for temporal scalability. In this structure, the prediction at the base layer is only from the base layer. This ensures that the decoder will be able to correctly decode the sequence even if only the base layer is received. The enhancement layer provides the additional frames needed to decode the sequence at a higher frame rate. The prediction at the enhancement layer can be formed using either the base or the enhancement layer itself.

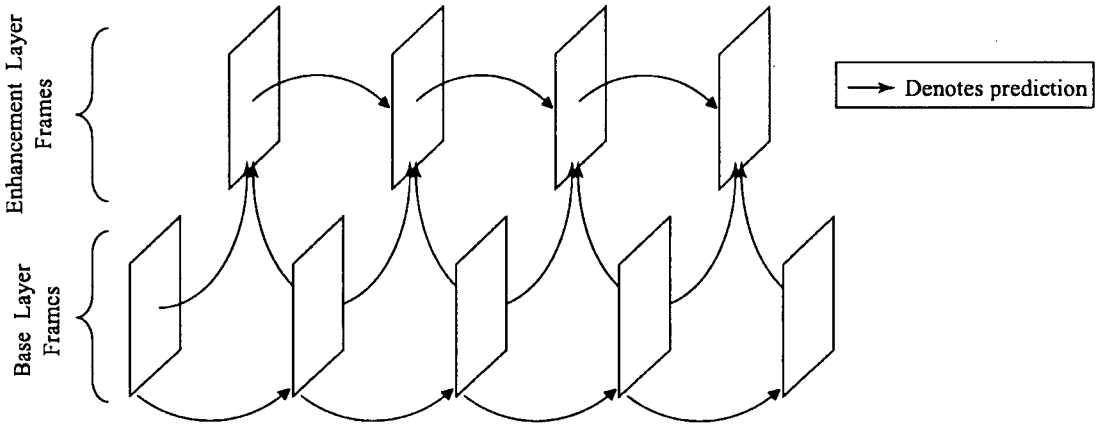


Figure 8 Frame Structure used in Temporal Scalable Systems

1.3.2.3 Spatial Scalability

Spatial Scalability refers to the technique where the video is coded into two layers at the same frame rate but with different spatial resolutions. The base layer is coded at a low resolution, whereas the enhancement layer is coded at a higher resolution. At the time of encoding, the up-sampled base layer picture can be used as prediction for the enhancement layer. The MPEG-4 spatial scalable decoder allows a "bi-directional" prediction at the enhancement layer. Both the up-sampled picture from the base layer and the previously reconstructed frame from the enhancement layer can be used as prediction for the frames at the enhancement layer. Figure 9 shows the picture structure for this kind of scalability. The frames are either coded as P or B type at the enhancement layer. The frame at the enhancement layer which is temporally coincident with an I-frame at the base layer is encoded as a P-frame. The frame at the enhancement layer which is temporally coincident with a P-frame at the base layer is encoded as a B-frame. For the P-frames at the enhancement layer, the prediction is the up-sampled reconstructed frame from the temporally coincident I-frame at the base layer. The B-frames at the enhancement layer allows "bi-directional" prediction using the up-sampled reconstructed

frame from the base layer as the backward reference and the previously reconstructed frame in the enhancement layer as the "forward reference". For the cases where the prediction from the base layer is selected, the motion vectors are not encoded to reduce the amount of side information transmitted.

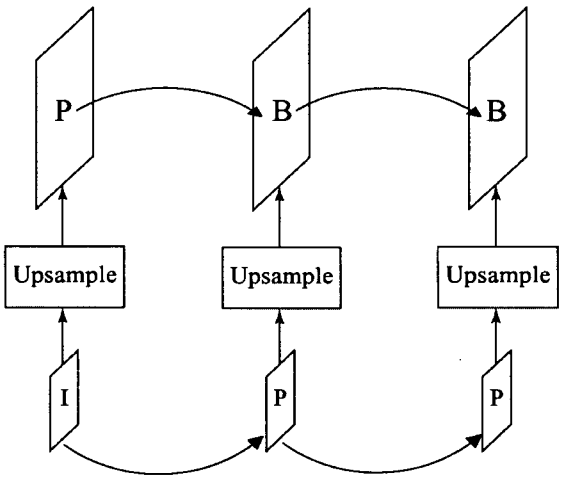


Figure 9 Frame Structure used in Spatial Scalability Systems

Figure 10 illustrates the diagram for the discussed spatial scalable encoder. The original video signal is downsampled and the low resolution video signal is generated. The low resolution video signal is encoded separately and the resulting bitstream represents the base-layer information. The reconstructed low resolution frames are upsampled and are made available as an additional prediction for the enhancement layer frames. The resulting prediction error of this combined prediction is encoded and forms the enhancement layer bitstream.

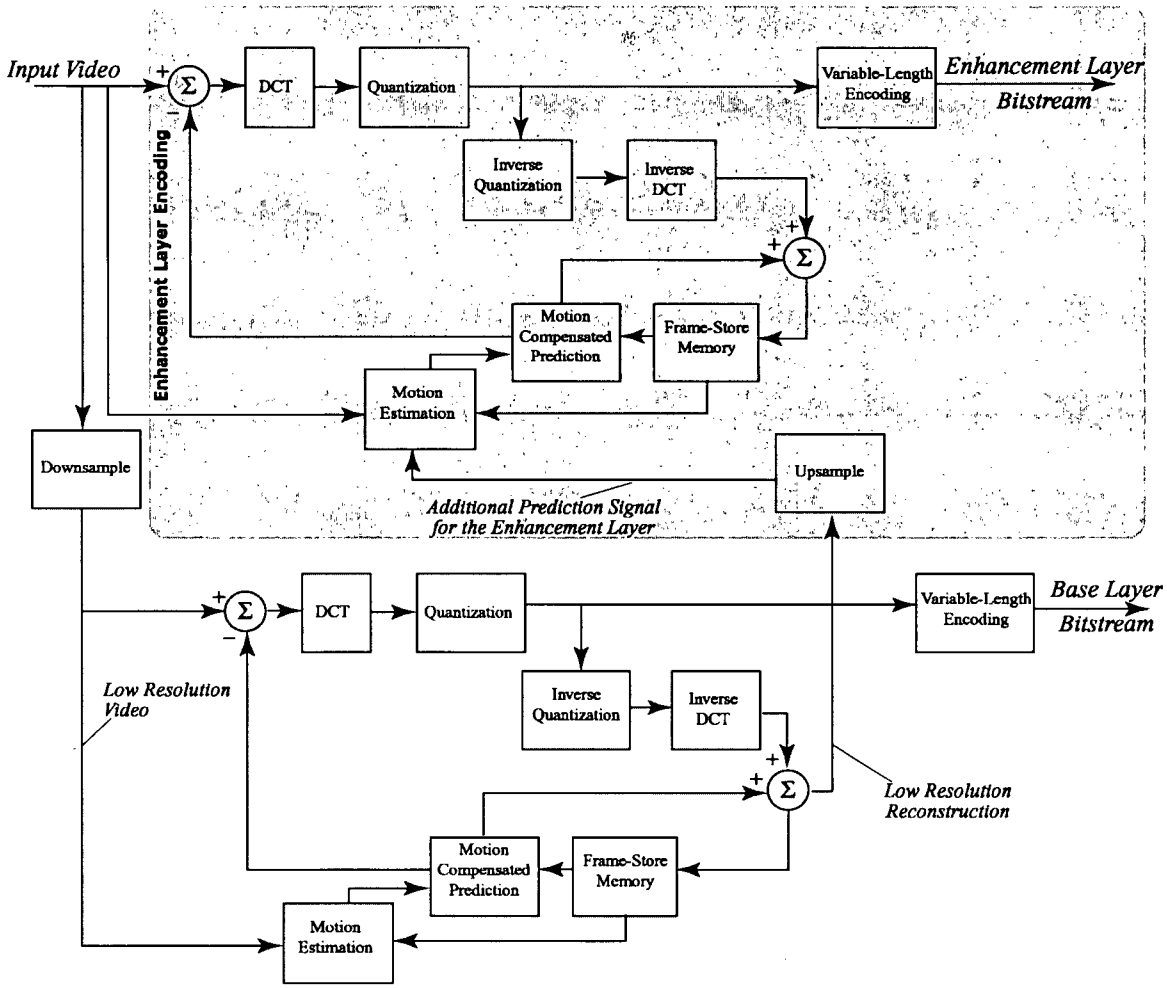


Figure 10 Block diagram of Spatial Scalable Encoder

1.4 Overview of MPEG-4 FGS Video Coding Standard

The system for delivering MPEG-4 FGS video is illustrated in Figure 11. This system consists of three components, FGS encoder, Streaming Server and FGS decoder. FGS encoder encodes the original video into two layers, base and enhancement layer. Because of the variation in the transmission bandwidth over time, the FGS encoder does not know what bitrate the video is going to be transmitted. For this reason, the base layer is encoded at the minimum bitrate that is guaranteed by the transmitting channel, $R_{min.}$ The enhancement layer is encoded at the maximum bitrate that the transmission channel can

deliver, R_{max}. During transmission, the streaming server truncates the enhancement layer bitstream according the available bandwidth. The number of bits sent to the decoder depends on the available bandwidth at the time of transmission. Thus, an FGS decoder receives the base layer and the truncated enhancement layer bitstreams. The quality of the decoded video is proportionally related to the number of bits received by the decoder for the corresponding frame. To summarize, FGS uses three components to deliver the video to the end user:

1. Scalable Video Encoder: encodes in a scalable manner at the highest possible quality.
2. Streaming Server: delivers scalable video to a given client. Maximum bandwidth utilization is achieved by truncating the video bitstream according to the available bandwidth
3. Decoder: decodes a truncated video bitstream. The reconstructed video quality decreases according to the amount of truncation performed at the streaming server.

The FGS Encoder and Decoder are further described in the next subsections.

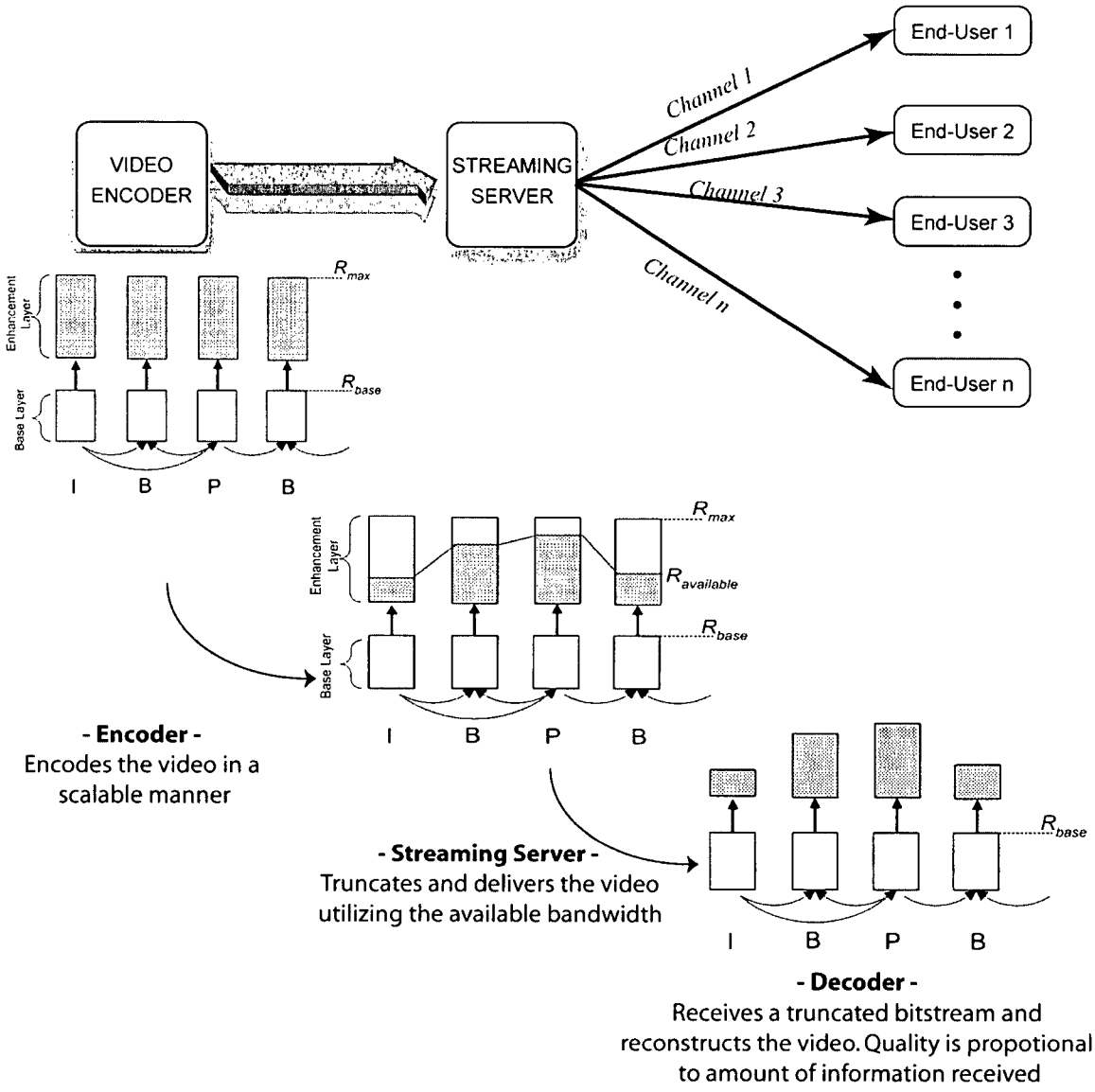


Figure 11. Structure of the end-to-end FGS video delivery system. The system comprises of *Encoder, Streaming Server* and *Decoder*

1.4.1 MPEG-4 FGS Encoder Structure

We illustrate the MPEG-4 FGS Encoder standard in Figure 12. The FGS results in an MPEG-4 non-scalable base layer encoded at an R_{base} bit-rate and an enhancement layer encoded using bitplane coding with a maximum bit-rate of R_{max}. To encode the enhancement layer, first the residual frame is formed by taking the difference of the original (high quality) and reconstructed base layer (low quality) frames. The residual

frame is then DCT transformed to remove the spatial redundancy. The obtained DCT coefficients are bitplane and entropy coded to form the enhancement layer bitstream. The main steps of FGS enhancement layer coding can be summarized as:

1. Constructing the Residual Frame
2. DCT Transforming the residual frame to decrease the spatial correlation
3. Bitplane encoding of DCT coefficients
4. Entropy encoding of bitplane encoded symbols

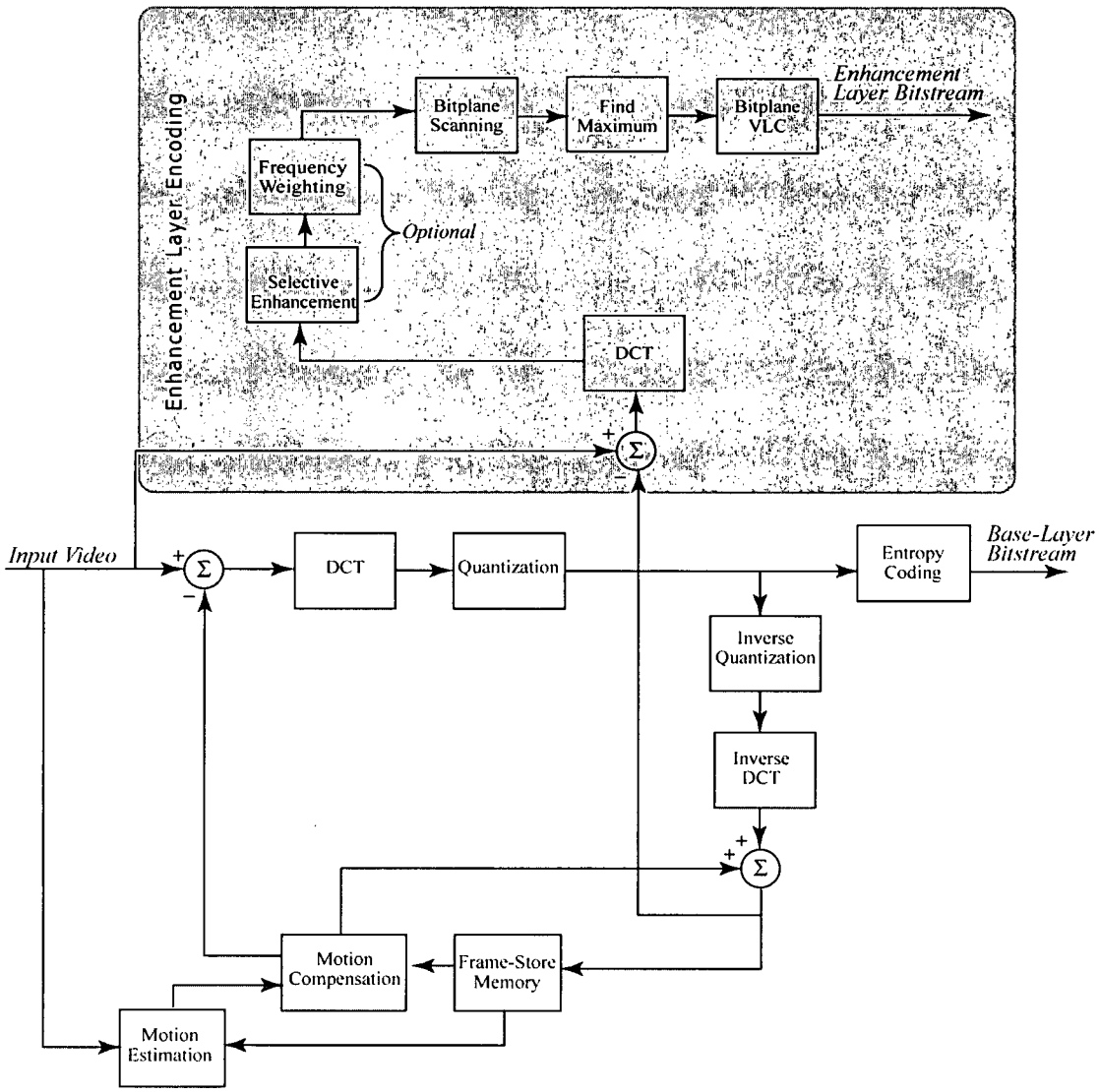


Figure 12. Block diagram of MPEG-4 FGS Encoder

1.4.1.1 Bitplane Coding of DCT Coefficients

The residual frames that are found by subtracting the base layer frames from the enhancement layer frames are coded by bitplane coding instead of conventional DCT coding. In the conventional DCT coding, the quantized DCT coefficients are zigzag scanned, then a symbol for every non-zero coefficient within the block (containing its value and information regarding the number of consecutive zeros before it) is found. The resulting symbols are mapped to binary codewords using a VLC table.

In bitplane coding, every DCT coefficient is treated as a binary number of several bits instead of a decimal integer of a certain value [21, 22]. For each block in the residual frame, the *absolute* values of its coefficients are scanned in the zigzag order as shown in Figure 4 and then assembled in a one dimensional array as shown on the left hand side of Figure 14. In Figure 14, we assume the absolute value of any coefficient is between 0 and 31 meaning 5 bitplanes are needed to represent all the coefficients correctly. The maximum number of bitplanes needed for each frame is found before bitplane coding, at the "Find Maximum" stage. It should be noted that, the number of bitplanes needed to code the luminance and chrominance components of the frame may be different as illustrated in Figure 13. Therefore, there are three syntax values **maximum_level_y**, **maximum_level_u, maximum_level_v** and they are coded in the frame header to indicate the maximum numbers of bit-planes for the Y-U-V components of the frame respectively (For the case illustrated in Figure 13, **maximum_level_y** is 6, **maximum_level_u** is 4 and **maximum_level_v** is 4).

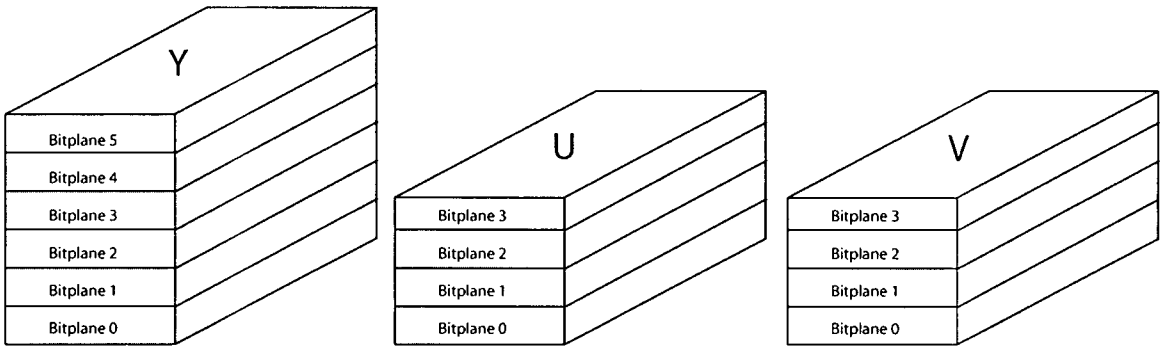


Figure 13 Maximum Number of Bitplanes Needed for Bitplane Coding maximum_level_y=6, maximum_level_y=4, maximum_level_v=4

When each entry in this array is written in binary form, a binary matrix results (see right hand side of Figure 14). A bit-plane of a block is defined as the one dimensional array of bits corresponding to a column of this binary matrix. The first bit-plane corresponds to the binary bits formed by the Most Significant Bit's (MSB)'s of the coefficients, whereas the MSB-1 form the second bit-plane and so on.

Transform Coefficients (Decimal)	Tranform Coefficients (Binary)				
16	1	0	0	0	0
15	0	1	1	1	1
14	0	1	1	1	0
19	1	0	0	1	1
⋮			⋮		
0	0	0	0	0	0
	MSB	MSB-1	MSB-2	MSB-3	MSB-4

Figure 14. The Bitplane Generation Process

After all the bitplanes are formed for each 8x8 transform block of the frame, symbols are generated for each bitplane. For each `1` in the bitplane, a symbol is formed. Each symbol

has two components, RUN and EOP. RUN specifies the number of consecutive zeros before the 1 and EOP specifies whether there are any more 1 left on this bitplane. If a bitplane contains all zeros, a special symbol ALL_ZERO is formed to represent it. For example, consider a bitplane that consists of following elements:

10000011000000001000...0

There are four 1's in the array, hence four symbols are generated. The first symbol refers to the first 1 in the array, which happens to be the first element of the array. There are no 0's preceding the first 1, so the RUN is 0 for the first symbol. Because there are more 1's after the first 1, EOP is 0. So the first symbol is generated as (0,0). There are five 0's between the first and the next 1, so the RUN for the second symbol is 5. EOP is still zero as there are more 1's in the bitplane. So the second symbol is generated as (5,0). Similarly, the next symbol is found to be (0,0). The last symbol refers to the last 1 in the bitplane array. There are eight 0's preceding the last 1, so the RUN is found as 8 for this symbol. As this is the last 1 in the bitplane array, EOP is 1. So the last symbol is found as (8,1). To summarize (0,0), (5,0), (0,0) and (8,1) are the symbols generated for this bitplane.

For the first and second bitplanes, it is very probable that most of the (8x8) blocks in a (16x16) macroblock will have ALL_ZERO symbols. That is every entry in the bitplane is zero. Instead of ALL_ZERO bitplanes separately, it is more efficient to group the ALL_ZERO bitplanes in the macroblock and code them together. This is only done for the first and second bitplanes. FGS standard uses *Coded Block Pattern* (CBP) for this purpose. CBP is a variable length coded binary string, placed at the macroblock header of

the bitstream, and specifies which blocks are ALL_ZERO within the macroblock. For details of CBP coding specified in FGS, please refer to [1].

1.4.1.2 Entropy Coding

At the final stage, the (RUN, EOP) symbols of the enhancement layer are variable length coded (VLC). In VLC coding, the generated symbols are mapped into binary codewords according to the symbols' statistics. These binary codewords are stored at the encoder and they constitute the VLC table for the enhancement layer. The exact VLC table is also stored at the decoder allowing identical reconstruction of the coded symbols.

1.4.2 MPEG-4 FGS Decoder Structure

Figure 15 illustrates the MPEG-4 FGS Decoder standard. The structure of FGS Decoder is similar to that of FGS Encoder. The FGS Decoder consists of two layers, base and enhancement layer. The FGS Decoder base layer is a standard MPEG-4 decoder that outputs the base layer video with the minimum quality. The FGS enhancement layer decoder is built on top of the base layer decoder to generate the enhancement video. The enhancement layer decoder operates on a truncated version of the enhancement layer bitstream. After the enhancement layer decoding process is done, the output is added to the output of the base layer decoding process to produce the high quality, enhancement video. Decoding steps of the enhancement layer for FGS decoding are presented below.

The enhancement layer bitstream is first decoded with an entropy decoder. The output of the entropy decoder are (RUN,EOP) symbols and some syntax elements that will be used in bitplane decoding. The next step is the bitplane decoding step where the DCT coefficients are reconstructed. Note that, unless all the enhancement layer information is transmitted to the decoder, the reconstructed DCT coefficients at the decoder side are not

identical to the DCT coefficients encoded at the FGS encoder, before transmission. The more information the decoder receives, the more accurate the reconstructed DCT coefficients are. The reconstructed coefficients are then inverse transformed and the result is added to the base layer video to obtain the enhancement video.

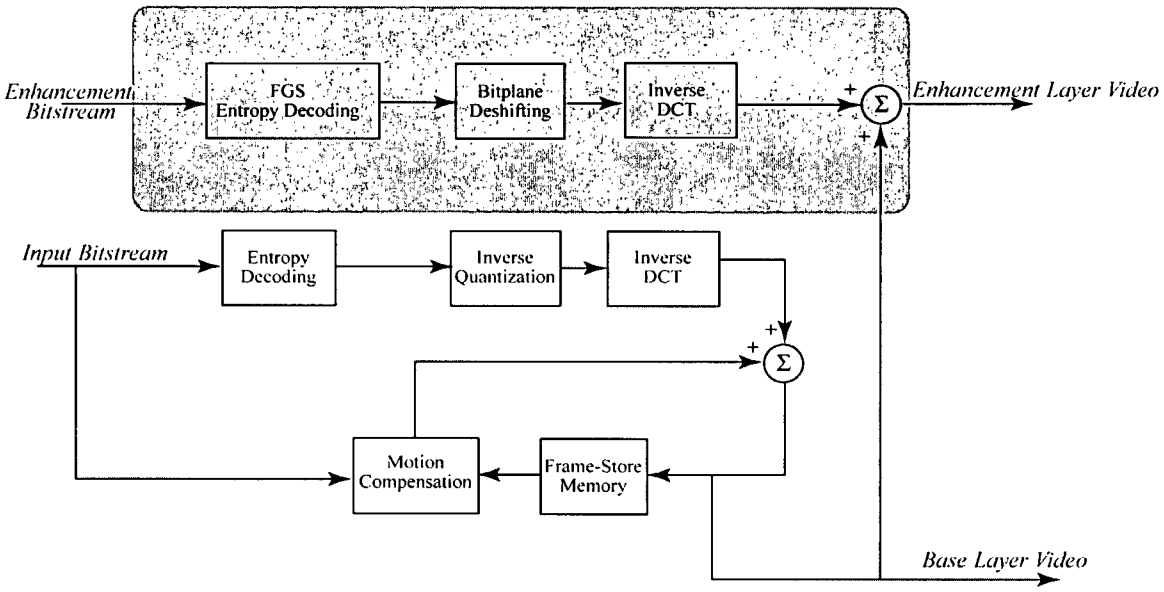


Figure 15 Block Diagram of the MPEG-4 FGS Decoder

1.5 Overview of H.264 Video Coding Standard

H.264 is the newest video coding standard and it was developed by ITU-T Video Coding Experts Group and ISO/IEC Moving Picture Experts Group. H.264 includes a number of advances in video coding technology, making it highly efficient in terms of coding and network friendliness. The design of the standard is based on a conventional block based motion compensation video coding concept described in the previous sections. However, the design also includes several new features that result in a 50% bit rate savings when compared with previous standards [18]. In this section, the main advancements offered by the H.264 video coding standard is presented, for further details please refer to [15].

1.5.1 Advances in Motion Compensated Prediction

H.264 is much more flexible in terms of motion compensation block sizes and it can support a luminance motion compensation block size as small as 4x4. The use of a smaller block size in the motion compensation stage allows the encoder to describe complex motions more accurately, thus decreasing the prediction error. In addition, H.264 supports quarter-pel motion compensation that further improves the coding efficiency of the video coding system. H.264 supports multiple reference pictures for motion compensation by the addition of new inter-prediction types. These features enable motion to be represented a lot more accurately than previous standards. They also increase the coding efficiency of the system considerably.

H.264 standard also includes an in-loop deblocking filter to decrease the blocking artifacts and increase the coding efficiency of the video. The blocking artifacts originate from both the motion compensated prediction and the residual coding stages of the process and are especially visible at low bitrates. Although, the application of a deblocking filter, i.e., after the video is decoded, has been used in previous video coding standards, H.264 places such a filter in the motion compensation loop. This improves the coder's ability to do inter-prediction that in turn, results in a better compression ratio.

1.5.2 Advances in Transform Coding

One of the most important features of H.264 is its use of a different transform. Unlike the major video coding standards such as MPEG-2, MPEG-4 that use 8x8 DCT, H.264 uses a 4x4 Integer Transform. It was observed that the smaller block size decreases some of the

artifacts associated with transform coding. Apart from the size, the low complexity nature of the 4x4 Integer Transform makes it very efficient to implement on hardware platforms such as ASIC's or digital signal processors. Unlike DCT, the Integer Transform was designed to allow exact-matching inverse transform. This eliminates the "drift" problem due to a slight mismatch between the encoder and decoder representation of video.

1.5.3 Advances in Entropy Coding

H.264 includes two methods for entropy coding, the first one is Context Adaptive Variable Length Coding (CAVLC) and the second one is Context Adaptive Binary Arithmetic Coding (CABAC). Both coding methods use context based adaptivity to improve the performance of the encoder for all types of sequences. CAVLC has relatively less computational complexity and also includes Reversible Exp-Golomb codes to code some syntax elements. Reversible Exp-Golomb codes can be used to improve the error resilience of the system and they are further described in Section 2.3.2.1.

CABAC is a more powerful than CAVLC and significantly improves the coding performance of the system but with an additional complexity to encode/decode.

CHAPTER 2

2 H.264 Based Fine Granular Scalability (FGS)

2.1 Introduction

The latest video coding standard, H.264, provides superior compression efficiency to all previous standards, but it does not include tools for coding the video in scalable fashion. We introduce scalability for the H.264 standard, so that it can be used more efficiently in network environments where bandwidth varies over time. This chapter presents the details of our developed scalable H.264 structure. This structure is based on the latest scalable video coding standard, Fine Granular Scalability (FGS) that is originally developed for MPEG-4. The proposed structure is not a straightforward extension of FGS, where the FGS structure is implemented on H.264 without any modifications. The techniques present in FGS are modified and novel techniques are developed in order to achieve the best adaptation of FGS to H.264. By modifying the FGS structure, we achieve low complexity, high coding efficiency and high error-resiliency for the overall system.

FGS is the latest scalable video coding standard that was developed within the MPEG committee and is included in MPEG-4 Streaming Video Profile. FGS encodes the video with two different layers, the base layer and the enhancement layer. The enhancement layer is encoded using bitplane coding and its fine granular scalable nature makes the FGS standard a very flexible coding tool for adapting to the dynamic bandwidth change of the underlying network. The base layer of the FGS standard is encoded using

traditional video coding technologies. The current MPEG-4 FGS standard uses MPEG-4 to encode the base layer.

H.264, the latest video coding standard, developed by the Joint Video Team (JVT) of ITU-T and ISO provides superior compression efficiency to MPEG-4. Because at a given bitrate H.264 is able to provide better video quality than previous video coding standards, it is predicted to be widely adopted. One possible application area for H.264 is video communications over best-effort networks, where the available bandwidth for video transmission varies with time.

Although H.264 offers better efficiency than MPEG-4 in terms of compression ratio, it lacks tools that make it scalable for use at different bitrates. One possible way of introducing scalability to H.264 is to directly apply the FGS process as is done in MPEG-4. Thus the FGS base layer is encoded using H.264 instead of MPEG-4, while the same process as in MPEG-4 FGS enhancement layer is applied as is [12]. Such a straightforward extension of FGS is possible due to FGS' design that allows the use of any video coding standard for encoding the base layer video. This, however, presents serious drawbacks because of the fundamentally different video coding tools used in H.264 & MPEG-4. Firstly, encoding the enhancement layer using FGS (as in MPEG-4 coding) introduces Discrete Cosine Transform (DCT) computations to the H.264 system that uses Integer Transform. This significantly increases the complexity of both the encoder and decoder (particularly the latter). Secondly, the resulting system would fail to encode the enhancement layer using the advanced techniques introduced by H.264, which has proved to significantly improve the picture quality, increase the error resilience and decrease the complexity of the overall system.

In this thesis, we overcome the aforementioned drawbacks, by modifying the FGS video coding standard and by introducing new techniques. The developed tools increase the error resilience and decrease the encoding and decoding complexity of the scalable video coding system.

In Section 2.2, we first present the trivial extension of FGS to H.264 and discuss its drawbacks. Following that discussion, our proposed H.264 based FGS structure is presented in Section 2.3. In Section 2.4, we present the experimental results. We summarize and conclude the chapter in Section 2.5.

2.2 Trivial Extension of FGS into H.264

Figure 16 illustrates the encoder that is a straightforward implementation of FGS into H.264. The base layer is encoded using H.264 instead of MPEG-4.

There are two different approaches to calculate the residual signal, resulting in two different ways of encoding and decoding. This separation is due to the in-loop deblocking filter present in the H.264 standard. The residue signal for the enhancement layer can be formed by taking the difference between the original signal and the reconstructed base layer signal right after the deblocking filter is applied to the base layer signal. In an alternative way, the residue signal can be formed using the base layer signal prior to filtering operation. In this case, an additional deblocking operation would be needed at the decoder side for the enhancement layer to reduce the blockiness of the decoded video which in turn increases the complexity of the decoder. Figure 17 shows the decoders for both cases.

As mentioned before, this direct implementation is not the most efficient solution for an H.264 based FGS encoder. The reason for this is explained in the following subsection.

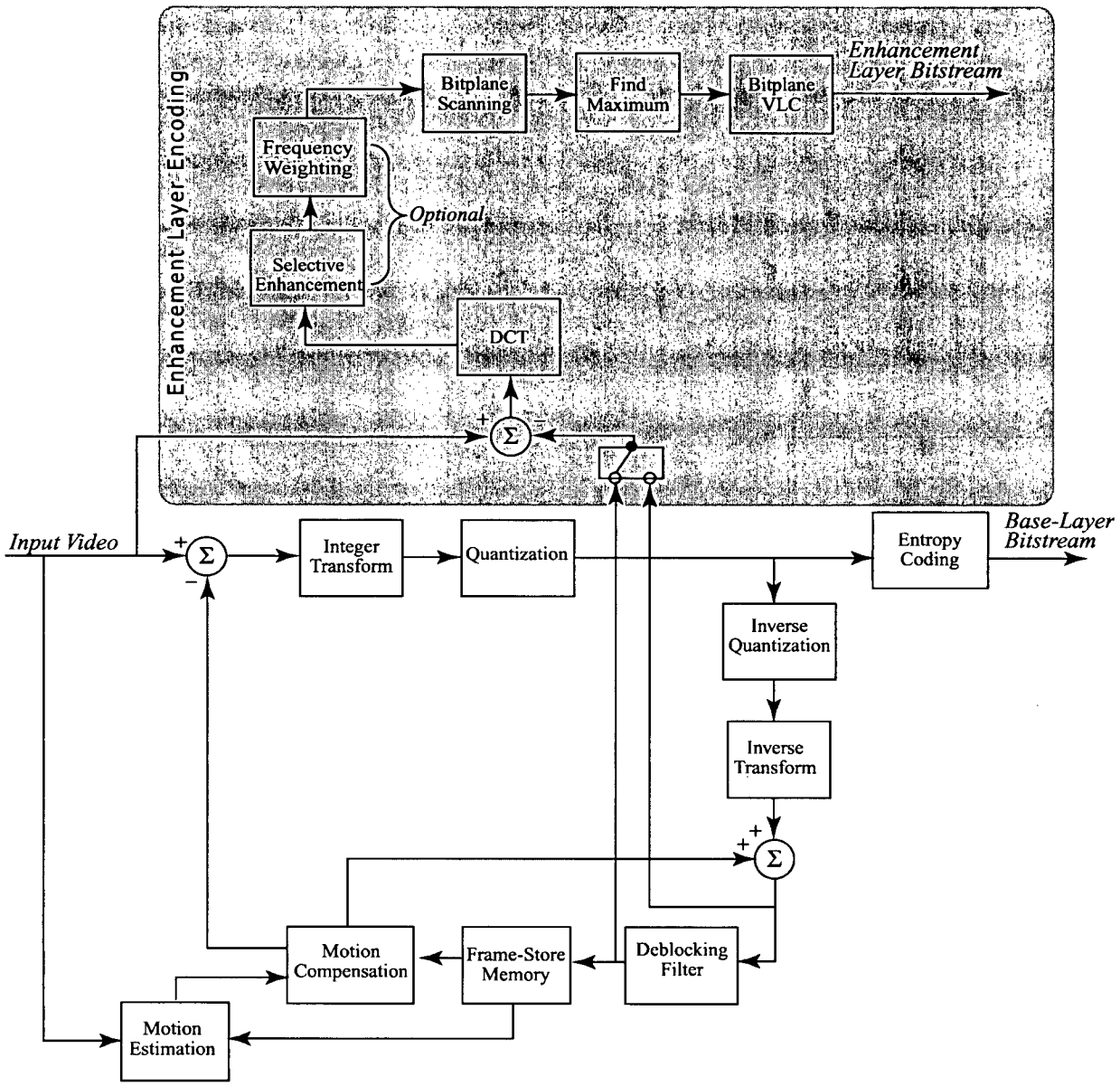
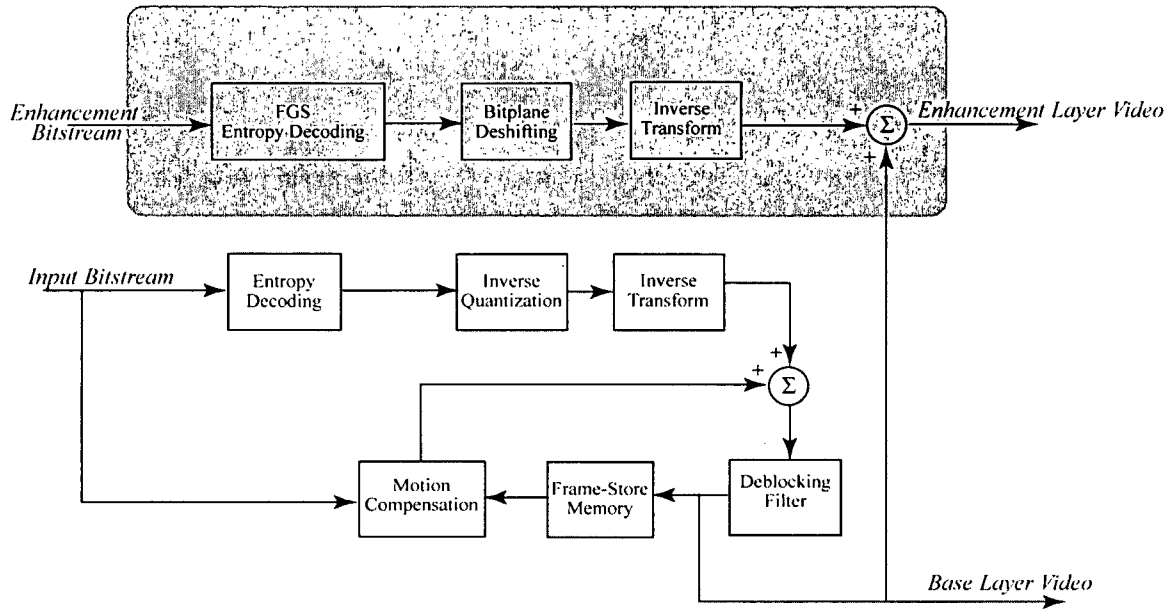
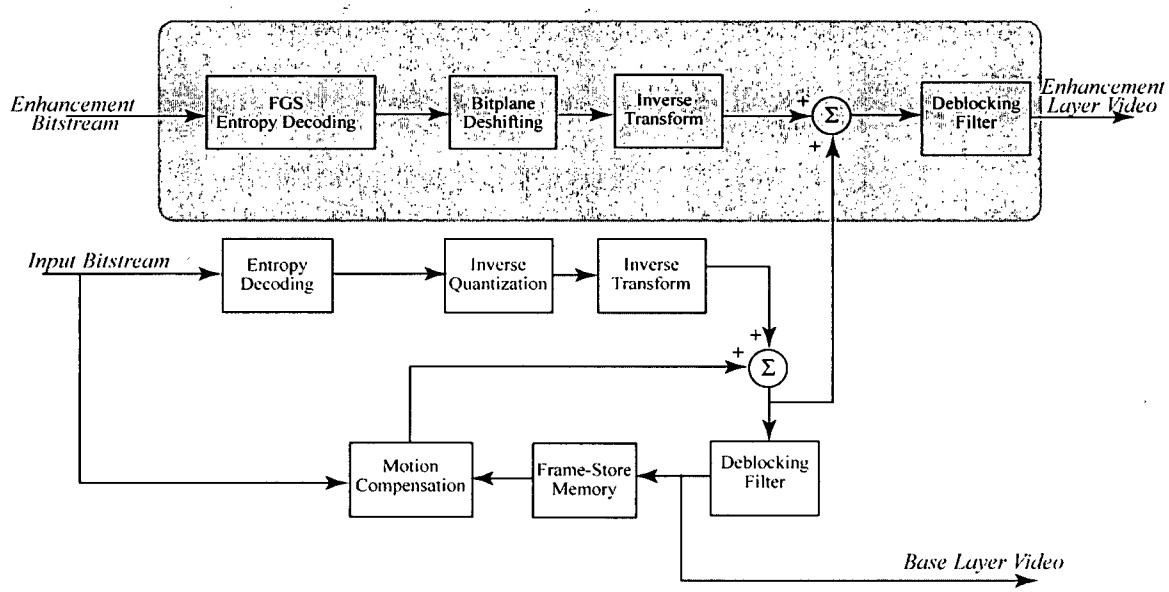


Figure 16.Block Diagram of the direct implementation of FGS Encoder on H.264 Encoder



(a)



(b)

Figure 17.Block diagram of corresponding decoders for two cases of direct implementation of FGS on H.264
(a) Residual Signal is calculated after the deblocking filter at the base layer
(b) Residual signal is calculated before the deblocking filter at the base layer

2.2.1 Drawbacks of Trivial Extension of FGS to H.264

MPEG-4 FGS employs DCT for transform coding both at the base and the enhancement layers. On the other hand, the H.264 video coding standard replaces DCT with a low complexity 4x4 Integer Transform. The encoder used for trivial extension of FGS, depicted in Figure 16, uses 4x4 Integer Transform at the base layer and DCT at the enhancement layer. Using two different transforms introduces additional complexity to the entire system (both to the encoder and decoder). Also, by using DCT at the enhancement layer, the system cannot make use of the superior features of the 4x4 Integer Transform, such as its low implementation complexity and increased subjective quality [13].

The FGS video coding standard uses four different VLC tables at the entropy coding stage. In contrast, H.264 employs Reversible Exp-Golomb codewords where a VLC table needs not to be stored. Also, Reversible Exp-Golomb codewords increase the error resilience of the system and also can be very efficiently implemented on digital processors [23]. In the trivial extension of FGS over H.264, the enhancement layer cannot take advantage of these features of Exp-Golomb coding. Also additional complexity is introduced to the system by its storage need of four more VLC tables.

In summary, the trivial extension of FGS introduces new computation blocks to the system complexity of both the encoder and decoder (particularly the latter). Secondly, the system fails to encode the enhancement layer using the advanced techniques introduced by H.264.

2.3 Proposed H.264 based FGS System

In this section, the proposed H.264 based FGS system is presented. Our proposed system mainly modifies Transform Coding and Entropy Coding structures of the FGS standard. The technical details of these modifications are presented in the following subsections.

2.3.1 Proposed Transform Coding Structure

In our proposed encoder, the DCT transform at the enhancement layer is replaced by the H.264 4x4 low complexity Integer Transform. Consequently, the original FGS macroblock structure has to be changed since the size of the transform has changed. Figure 18 compares the macroblock structures for the 8x8 DCT and 4x4 Integer Transforms.

For the case of the 8x8 transform, one macroblock contains 4 blocks of luminance and 2 blocks of chrominance that is a total of 6 transform blocks. On the other hand, the smaller 4x4 transform results in 16 blocks of luminance and 8 blocks of chrominance, for a total of 24 transform blocks for each macroblock. This increased number of transform blocks increases the number of bits needed to code the *Coded Block Pattern* (CBP) for each macroblock at the macroblock header, and as a result decreases the coding efficiency. CBP is a variable length coded binary string, placed at the macroblock header of the bitstream. For details of CBP coding specified in FGS, please refer to [1].

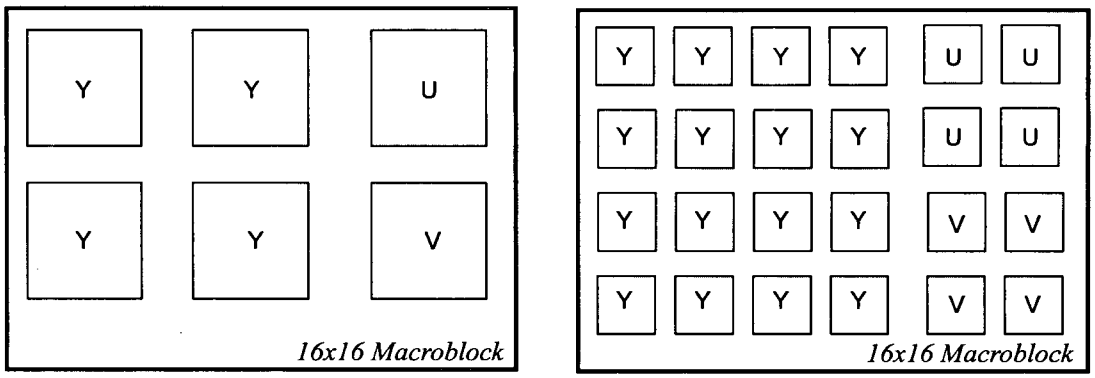


Figure 18. FGS Macroblock Structures for two different transform sizes
Left: The 8x8 DCT Transform used by MPEG-4
Right: 4x4 Integer Transform used by H.264

2.3.1.1 Proposed CBP Coding Scheme

The reason behind the increased overhead in CBP is that for the 4x4 Integer Transform, one macroblock contains 24 blocks instead of 6, and thus, each CBP code needs to provide information for more blocks. The increased overhead can be analyzed by considering Figure 19. In Figure 19, the 24 (4x4) blocks within a macroblock are grouped into 4. Each (8x8) group contains four blocks of luminance and two blocks of chrominance. The structure of the resulting 8x8 groups is the same as that of the MPEG-4 FGS macroblock structure, as shown in Figure 19. Hence, the same coding algorithm as MPEG-4 FGS CBP coding can be used to code the CBP for each of the new groups. This approach results in using 4 CBP codewords for each macroblock. So the amount of bits spent for CBP is approximately quadrupled.

In order to reduce this overhead, we propose a hierarchical scheme to code the CBP. The main idea behind this scheme is grouping the transform blocks into larger size groups and coding the CBP code in steps. The proposed Hierarchical CBP Coding Scheme is presented in the next subsection. The experimental results of the proposed scheme are presented in Section 2.4.

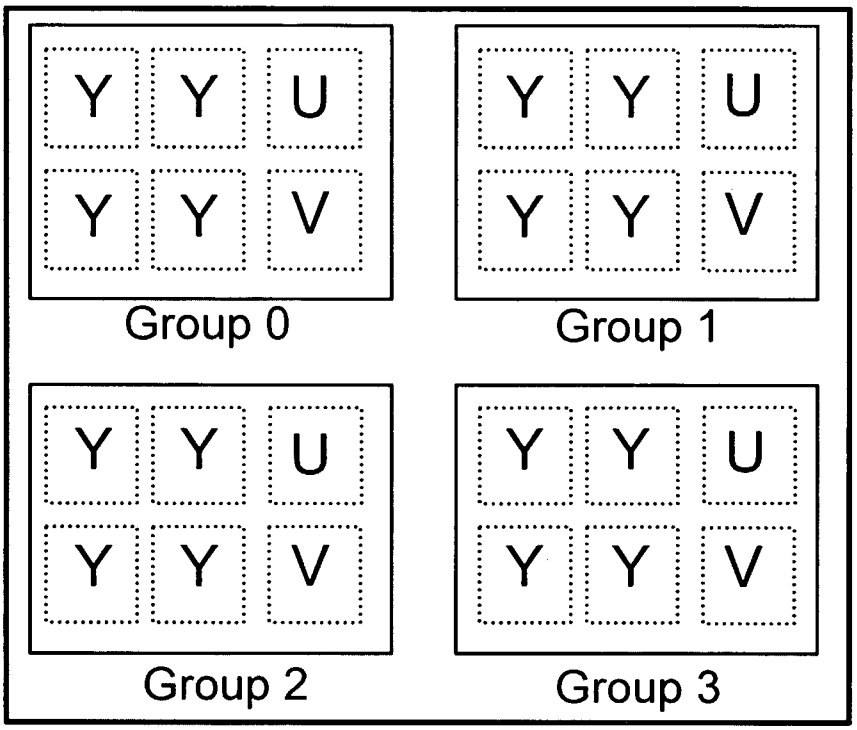


Figure 19. Grouping Scheme for the Simple CBP Coding

2.3.1.1.1 Hierarchical CBP Coding Scheme

In the proposed CBP coding scheme, the 4x4 blocks are grouped into groups of four as illustrated in Figure 20. This scheme groups the blocks into 6, each group containing four transform blocks of either luminance or chrominance.

The proposed CBP coding scheme refers to blocks in a hierarchical fashion. (see Figure 21). There are two steps in the proposed CBP coding scheme for the first bitplane. At the first step, each group within the macroblock is checked whether all the 4x4 blocks belonging to the group are ALL_ZERO or not. If all the blocks within the group are ALL_ZERO, then the group is classified as an ALL_ZERO group, otherwise that group is classified as non-zero.

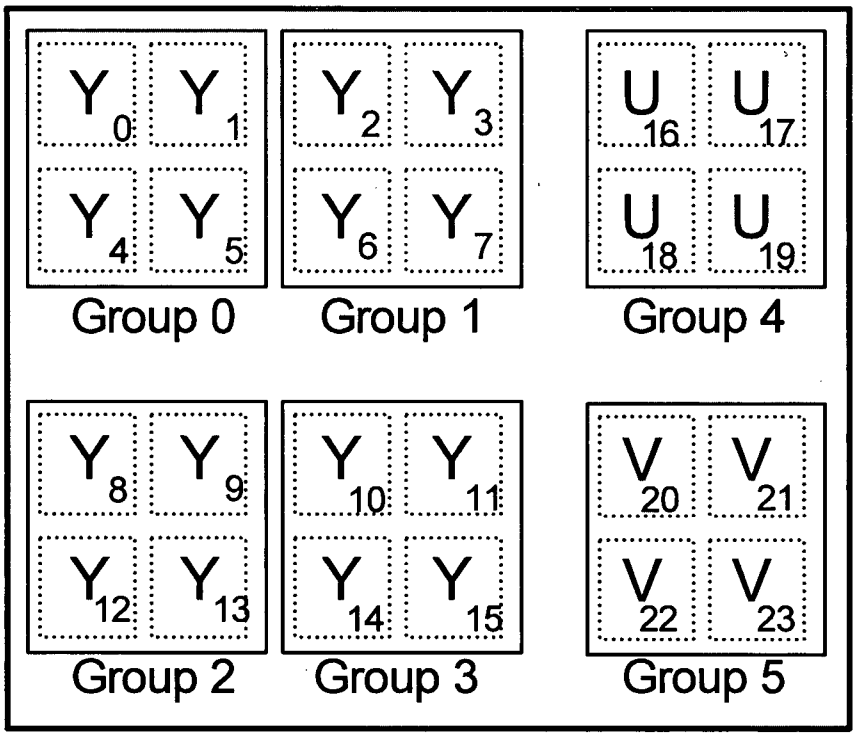


Figure 20. Grouping Scheme for Hierarchical CBP Coding

At the second step of the proposed CBP coding scheme, non-zero groups are considered only. The reason for this is, ALL_ZERO and non-zero blocks can co-exist in a non-zero group, whereas only ALL_ZERO groups exist in an ALL_ZERO group. Thus, if a group is classified as ALL_ZERO at Step-1, no further information is required for the blocks within that group. For example, in Figure 21, the groups numbered 0,2,3 and 4 are found to be ALL_ZERO (shown shaded in Step-1). The blocks belonging to those groups are not coded at Step-2 of CBP coding. At Step-2, only blocks belonging to non-zero groups are coded (groups 1 and 5).

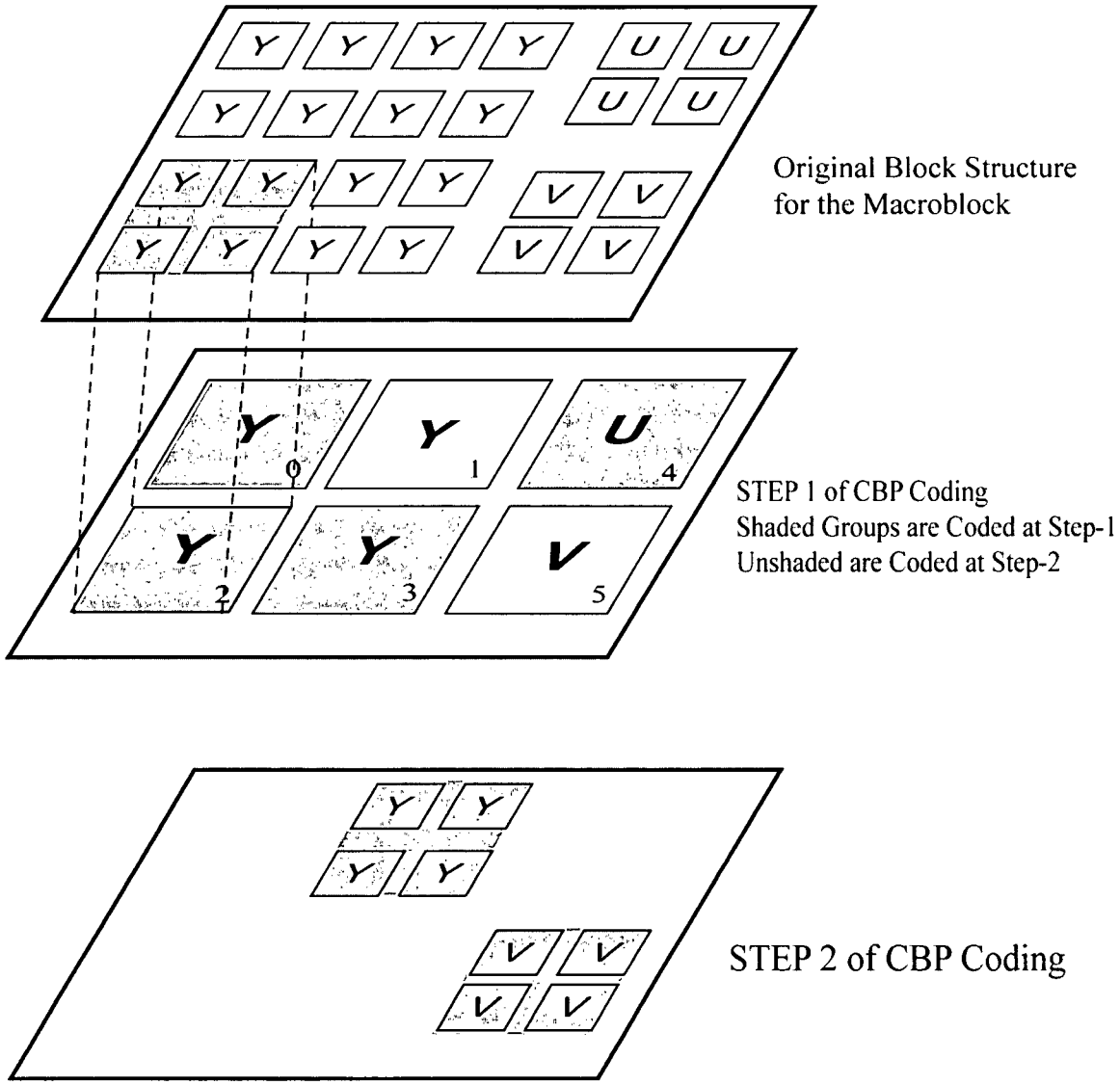


Figure 21 Illustration of Hierarchical CBP Coding

So, the two steps of our proposed CBP coding algorithm can be summarized as follows:

1. First step of CBP. At this step, the CBP specifies information about each group within the macroblock. (Step 1 at Figure 21). From now on, the procedure for specifying this information will be referred to as *group_cbp*. This procedure indicates if each **group** is ALL_ZERO or not.

2. Second step of CBP. At this step, the CBP specifies information about each block within a nonzero group. (Step 2 at Figure 21). This procedure is referred to as *block_cbp*. This procedure indicates if each **block** is ALL_ZERO or not.

Figure 22 illustrates the proposed main algorithm used to create the CBP code for a macroblock.

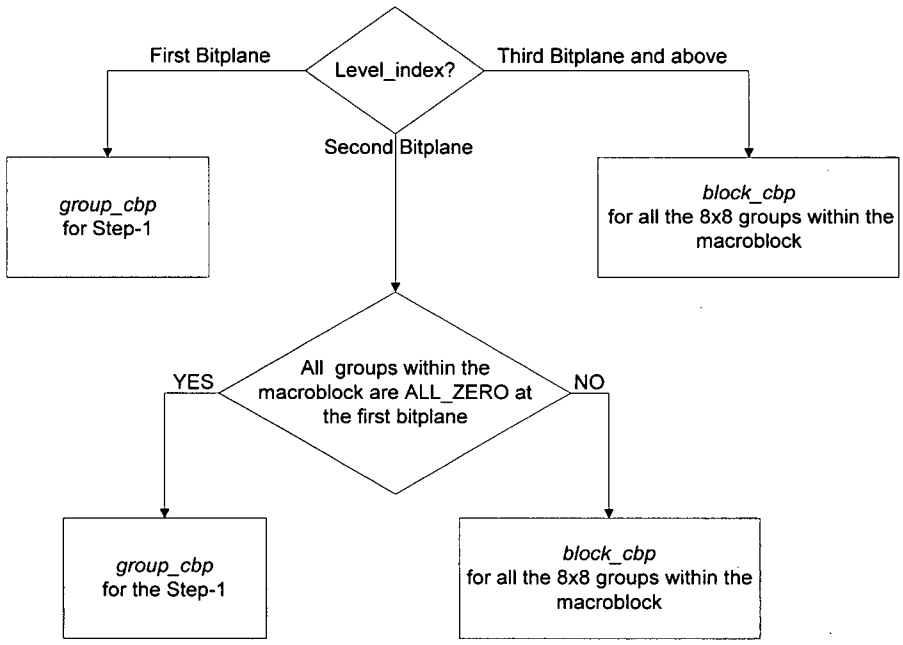


Figure 22 Block Diagram of the Main Hierarchical CBP Coding Algorithm

As mentioned before and can also be seen from Figure 22, *group_cbp* and *block_cbp* are the two procedures that are used to code the CBP. Based on the characteristics of the macroblock, either *group_cbp* or *block_cbp* procedure is used. Also, CBP coding for the first bitplane, second bitplane and bitplanes above second one change slightly. The details of these procedures for different cases are explained in detail in the following sections.

Step-1 of CBP Coding – group_cbp Procedure

The aim of this procedure is to specify, which groups within the macroblock are ALL_ZERO. Figure 23 illustrates the algorithm for the *group_cbp* procedure. It should

be noted that this procedure is not invoked for all the bitplanes of the macroblock. As can be seen from the main algorithm depicted in Figure 22, the cases where this procedure is invoked can be summarized as:

- For all the macroblock's first bitplanes
- For all the macroblock's second bitplanes, if the macroblock has an ALL_ZERO first bitplane.

This procedure generates a binary string called *CBP_CODE* for the entire macroblock. *CBP_CODE* specifies which groups within the macroblock are ALL_ZERO. In the *CBP_CODE*, a binary `1` means that the corresponding group is ALL_ZERO, while a `0` represents a non-zero group. It should be noted that, if any block within the group is not ALL_ZERO then the corresponding group is not an ALL_ZERO group. After the *CBP_CODE* is generated, it is variable length coded using the VLC tables presented in Appendix B. The VLC tables for *CBP_CODE* are based on Exp-Golomb codewords that is different from the codewords present in FGS standard. The details of Exp-Golomb coding are explained in detail in Section 2.3.2.1. The VLC tables are constructed based on the statistics of the *CBP_CODE.*

If a group is ALL_ZERO, this means that all the blocks within the group are ALL_ZERO and no further information is needed in the CBP for those blocks. However, an ambiguity exists for non-zero groups, since the *CBP_CODE* does not specify which of the blocks belonging to a non-zero group are ALL_ZERO. In order to address these blocks, the *block_cbp* procedure is invoked.

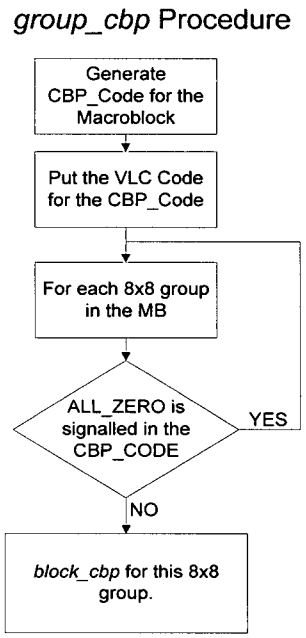


Figure 23 Block Diagram of *group_cbp* Procedure

Step-2 of CBP Coding, block_cbp Procedure

The aim of this procedure is to determine which blocks within a group are ALL_ZERO. The algorithm for this procedure is illustrated in Figure 24. It should be noted that, not all the groups within a macroblock are coded at this step. As can be seen from the main algorithm and the *group_cbp* procedure depicted in Figure 22 and Figure 23 respectively, this procedure is invoked for the following cases:

- For each non-zero group at the first and second bitplane.
- For all the groups of a macroblock at the second bitplane, if the entire macroblock has non-zero first bitplane.
- For all the groups at the third bitplane and above

This procedure first checks if all the blocks within the group are ALL_ZERO at the lower bitplanes (i.e., if we are coding a group at the third bitplane, we first check if this group has ALL_ZERO first and second bitplanes). If all the blocks within the group are

ALL_ZERO at previous bitplanes, then the variable length binary string called *SUB_CBP_CODE* is generated. *SUB_CBP_CODE* specifies which blocks within those groups are ALL_ZERO. In the *SUB_CBP_CODE*, a binary `1` means that the corresponding block is ALL_ZERO, while a `0` represents a non-zero block. For example, if only the first block in the group is ALL_ZERO, then the *SUB_CBP_CODE* would be `1000`. After the *SUB_CBP_CODE* is generated, it is variable length coded using the VLC tables presented Table 22 in Appendix C. The VLC tables are constructed based on the statistics of the *SUB_CBP_CODE*.

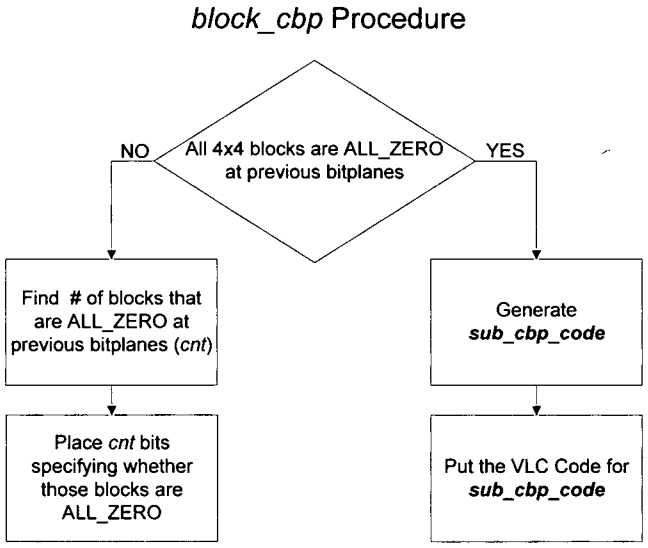


Figure 24 Block Diagram of block_cbp Procedure

If not all the blocks within the group are ALL_ZERO at lower bitplanes (i.e., the group contains a block that was non-zero at a lower bitplane), a different approach is taken. Blocks that are non-zero at previous bitplanes has very low probability of being ALL_ZERO at the current bitplane, thus, they are excluded in the process. For other blocks (have ALL_ZERO previous bitplanes), one bit is used to specify if they are ALL_ZERO at the current bitplane. Let's say, in a given group, only two blocks were

non-zero at lower bitplanes, and the first block of those two is ALL_ZERO at the current bitplane. Then, the code that will be placed to the bitstream is `10`.

2.3.1.1.2 Example of Hierarchical CBP Coding

The following example illustrates how the CBP is coded using the proposed Hierarchical CBP Coding method. This example considers CBP coding of a single macroblock for the first and second bitplanes. For simplicity, we only consider the coding of luminance component (i.e., the macroblock under consideration does not contain any color components).

The structure of the macroblock under consideration for the first bitplane is shown in Figure 25. For this macroblock, the first bitplanes of blocks 2, 7 and 10 contain non-zero coefficients, whereas all the rest have ALL_ZERO first bitplanes.

For the first bitplane, the *group_cbp* procedure is invoked for all the groups of the macroblock. Groups 0 and 2 are classified as ALL_ZERO groups because all the blocks' bitplanes within the group (i.e., blocks 0,1,4 and 5 for Group-0 and blocks 8,9,12 and 13 for Group-2) are ALL_ZERO. Groups 1 and 3 are classified as non-zero due to non-zero bitplanes these groups contain (the bitplanes of blocks 2 and 7 are non-zero for Group-1 and the bitplane of block 10 is non-zero for Group-3). So the *CBP_CODE* for the macroblock is found to be `1010`. First and third bits of *CBP_CODE* are 1, which indicates Group-0 and Group-2 as ALL_ZERO. Second and fourth bits of *CBP_CODE* are `0`, which indicates Group-1 and Group-3 as non-zero. After this, the VLC code corresponding to the *CBP_CODE* is found using the tables in Appendix B and placed in the bitstream. For this case, the VLC code is found as `0001101`.

This page is left intentionally blank

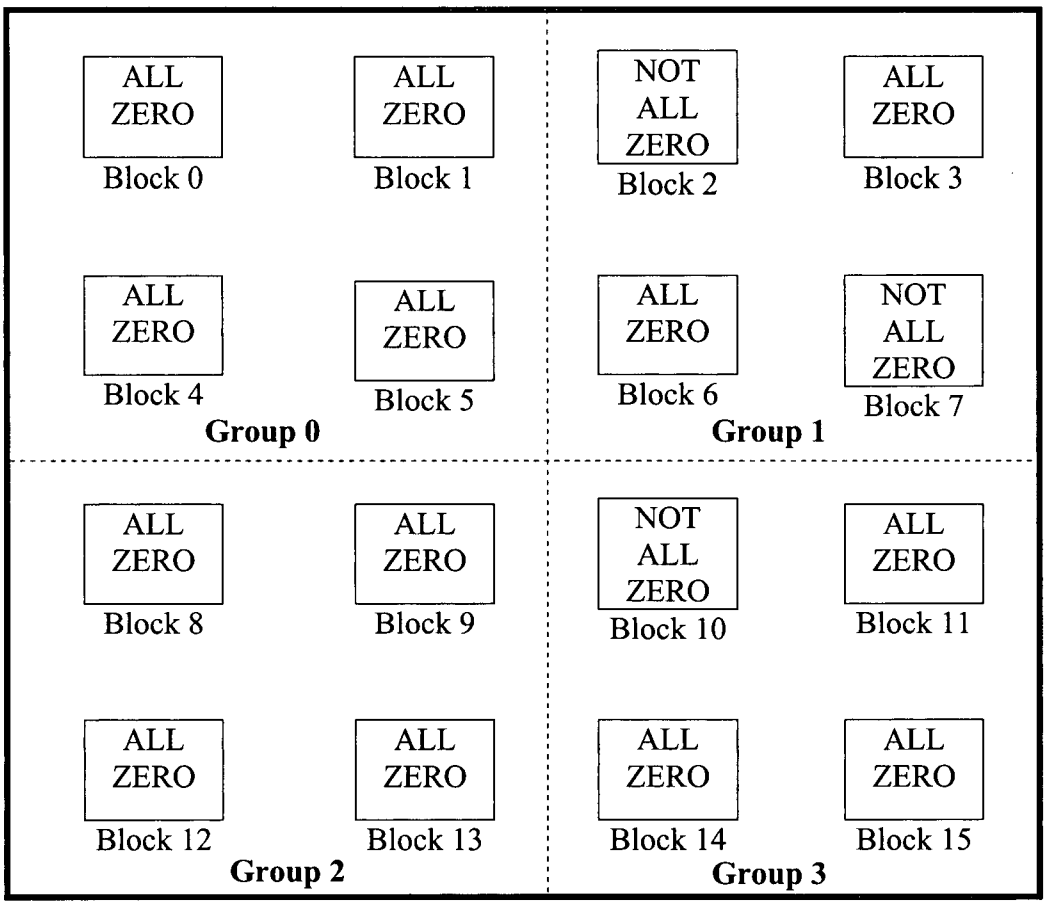


Figure 25 Example on Hierarchical CBP Coding, Group Structure of the First Bitplane

At the next step, *block_cbp* procedure is invoked for non-zero groups. So, Group-1 and Group-3 (having non-zero first bitplanes) are further coded using *block_cbp* procedure. The *block_cbp* procedure first checks if all the blocks within the group are ALL_ZERO at the lower bitplanes. As the current bitplane being coded is the first one, all the blocks within Group-1 and Group-3 are defined as ALL_ZERO at lower bitplanes. For these groups, a binary string that is called *SUB_CBP_CODE* is generated and placed in the bitstream. *SUB_CBP_CODE* is similar to *CBP_CODE*, but it specifies which blocks are ALL_ZERO instead of specifying which groups are ALL_ZERO. In Group-1, Block-2

and Block-7 have ALL_ZERO bitplanes, this means *SUB_CBP_CODE* for Group-1 is `0110` (second and third blocks within Group-1 has ALL_ZERO bitplanes). In Group-3, Block-11, Block-14 and Block-15 have ALL_ZERO bitplanes, this means *SUB_CBP_CODE* for Group-3 is `0111` (second, third and fourth blocks within Group-3 has ALL_ZERO bitplanes). After the *SUB_CBP_CODE* is constructed for all the non-zero groups, their VLC Codes are found using tables presented in Appendix C.

	SUB_CBP_CODE	**VLC Code**
Group 1	`0110`	`00110`
Group 3	`0111`	`00111`

This step concludes the CBP coding for the first bitplane of the macroblock. The code placed for CBP for this macroblock at the first bitplane is: `001100 00110 00111` that results in a total number of bits of 16. It should be noted that, in general number of bits needed to code the CBP of the macroblock is lower than this specific example. For detailed analysis, please refer to experimental results at the end of this section.

Figure 26 illustrates the structure of the macroblock under consideration for the second bitplane.

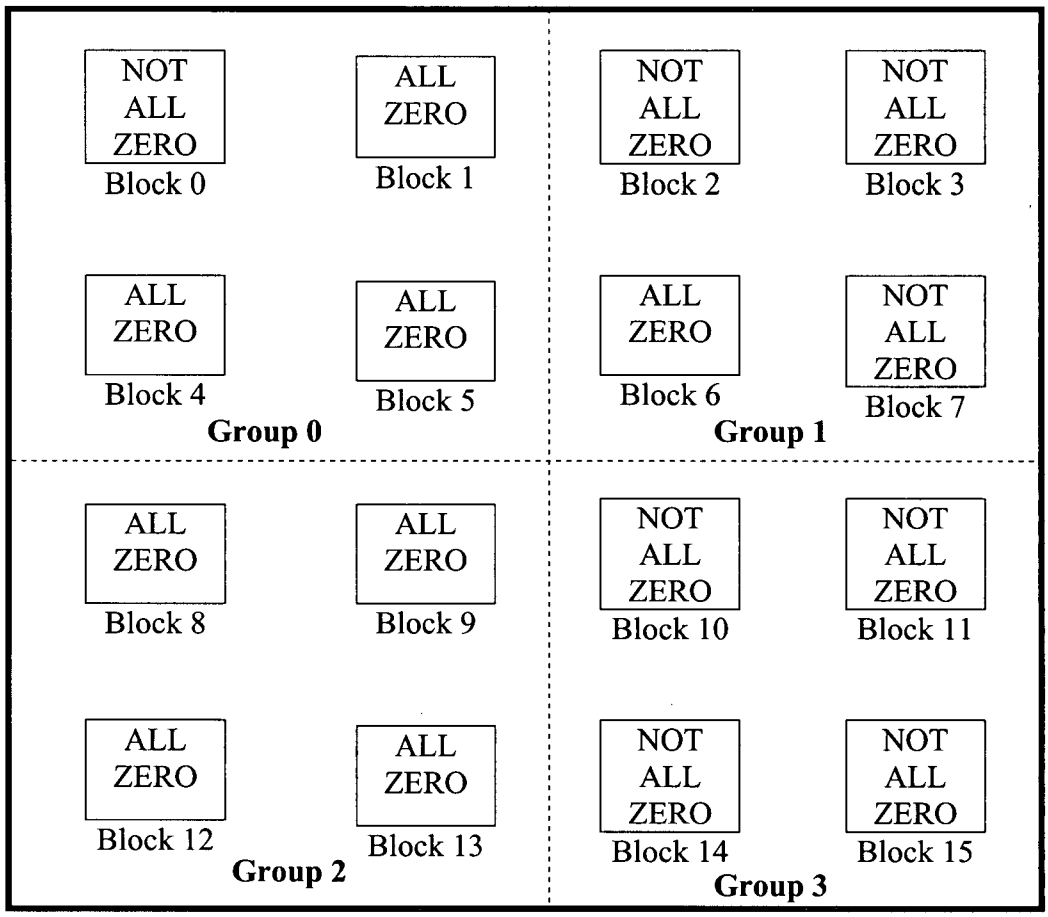


Figure 26 Example on Hierarchical CBP Coding, Group Structure of the Second Bitplane

It is first checked if all the groups within the macroblock are ALL_ZERO or not at the first bitplane. For this example, Group-1 and Group-3 are non-zero at the first bitplane. Thus, *block_cbp* procedure is invoked for each group within the macroblock.

In the *block_cbp* procedure, it is first checked if all the blocks within the group are ALL_ZERO at the first bitplane. For our example, all the blocks belonging to Group-0 and Group-2 are ALL_ZERO at the first bitplane. For these groups, *SUB_CBP_CODE* binary string is generated. In Group-0, only Block-0 has non-zero bitplane, rest of the blocks has ALL_ZERO bitplanes. Thus *SUB_CBP_CODE* is found as `0111` (only the

first block within the group is non-zero). In Group-2, all the blocks have ALL_ZERO bitplanes, thus *SUB_CBP_CODE* is found as `1111`. After *SUB_CBP_CODE* is constructed for all the non-zero groups, their VLC Codes are found using tables presented in Appendix C.

	SUB_CBP_CODE	**VLC Code**
Group 0	`0111`	`00111`
Group 2	`1111`	1

Not all the blocks within Group-1 and Group-3 are ALL_ZERO. That's why, *SUB_CBP_CODE* is not used for Group-1 and Group-3, but a different approach is taken. First, each block within those groups are checked whether they have ALL_ZERO first bitplanes. For the blocks having ALL_ZERO bitplanes, one bit is used to specify whether they have ALL_ZERO second bitplanes. Thus, in this approach, the number of bits placed in the bitstream is equal to the number of blocks being ALL_ZERO at previous bitplanes. In Group-1, Block-3 and Block-6 have ALL_ZERO first bitplanes and Block-3 has non-zero and Block-6 has ALL_ZERO second bitplane. So for this group, binary string `01` is generated and placed in the bitstream (first bit specifies Block-3 is non-zero at the second bitplane and second bit specifies Block-6 is ALL_ZERO at the second bitplane). In Group-3, only Block-10 has non-zero first bitplane and Block-11, Block-14 and Block-15 have ALL_ZERO first bitplane(three bits are used for Group-3). As seen from Figure 26, all these blocks are ALL_ZERO at the second bitplane. So for this group, binary string `000` is generated and placed in the bitstream The total code placed for CBP for this macroblock at the second bitplane is: `00111 1 01 000` that results in a total number of 11 bits.

2.3.1.2 Summary of Proposed Transform Coding Structure

In this section, we presented our novel structure that replaces the 8x8 DCT at the enhancement layer by the 4x4 Integer Transform. By replacing DCT by the H.264 4x4 Integer Transform, the complexity of the entire system (both to the encoder and decoder) is decreased. Also, by using Integer Transform at the enhancement layer, the system can make use of the superior features of the 4x4 Integer Transform, such as its low implementation complexity and increased subjective quality.

The consequence of using Integer Transform, instead of DCT is the change of the original FGS macroblock structure. This is due to the fact that the size of the Integer Transform is different than that of DCT. For the case of the 8x8 transform, one macroblock contains 4 blocks of luminance and 2 blocks of chrominance that is a total of 6 transform blocks. On the other hand, the smaller 4x4 transform results in 16 blocks of luminance and 8 blocks of chrominance, for a total of 24 transform blocks for each macroblock. This increased number of transform blocks increases the number of bits needed to code the binary string called *Coded Block Pattern* (CBP) for each macroblock at the macroblock header, and as a result decreases the coding efficiency.

In order to reduce this overhead, we presented our novel scheme that codes CBP more efficiently. The main idea behind the proposed scheme is grouping the transform blocks into larger size groups and coding the CBP in steps.

2.3.2 Proposed Entropy Coding Structure

As mentioned earlier, the entropy coding technique used in FGS is different than that of H.264. FGS uses four different VLC tables to code its symbols resulting from bitplane coding. H.264 uses Reversible Exp-Golomb Codewords to code some of its syntax elements (Context Adaptive VLC and Context Adaptive Binary Coding are other techniques supported by H.264, but are not considered in this thesis). Using different entropy coding techniques in enhancement and base layers of the H.264 FGS system increases the complexity of the system, thus, it is desirable to have the same structure used in base and enhancement layers. Also Reversible Exp-Golomb coding has the following advantages over FGS entropy coding technique:

1. Exp-Golomb codewords standardized in H.264 can be implemented very efficiently on digital processors [23].
2. Exp-Golomb codewords increase the error resilience of the system due to their reversible nature [14].

These advantages are particularly important in wireless video communication environments that are usually characterized as highly error-prone. Also, the size and cost limitations of low-end processors embedded in mobile units severely limit the complexity of the algorithms that can be used. Thus, for these applications low complexity features of the Exp-Golomb codes offer an additional advantage.

Based on all these reasons, we replace the VLC technique present in FGS, with its H.264 counterpart (based on Reversible Exp-Golomb Codewords). In this section, we first present the details of Exp-Golomb Coding process adapted in H.264. Following that

overview, we present the details of the proposed entropy coding scheme for our H.264 based FGS system.

2.3.2.1 Overview of Reversible Exp-Golomb Coding

The Exp-Golomb codeword table used in H.264 entropy coding is written as:

$$1$$

$$0\ 1\ x_0$$

$$0\ 0\ 1\ x_1\ x_0$$

$$0\ 0\ 0\ 1\ x_2\ x_1\ x_0$$

$$0\ 0\ 0\ 0\ 1\ x_3\ x_2\ x_1\ x_0$$

$$\ldots\ldots\ldots\ldots\ldots\ldots$$

where x_n take the values of 1 and 0. Each codeword is referred to by its length in bits, $L = 2n + 1$, and INFO = x_{n-1}, ... x_1, x_0. The codewords are numbered from 0 and upwards. When the number of bits, L, and INFO are known, the regular structure of the table makes it possible to create a codeword. This eliminates the need for storing a VLC table for the codewords. The first 10 codewords and their corresponding code numbers are presented in Table 1. Example of Exp-Golomb Codes for a larger sample is given in Appendix A, Table 13.

Code Number	Codeword
0	1
1	0 1 0
2	0 1 1
3	0 0 1 0 0
4	0 0 1 0 1
5	0 0 1 1 0
6	0 0 1 1 1
7	0 0 0 1 0 0 0
8	0 0 0 1 0 0 1
9	0 0 0 1 0 1 0

Table 1 First ten Exp-Golomb Codewords

A decoder decodes the codeword by reading the n+1 bit prefix followed by n bits for the INFO. The n+1 bit prefix is a string of zeros followed by a 1. (i.e. `00001` for n=4). The following n bits after the prefix give the INFO.

These codewords are characterized as *reversible*, which means decoding of these codewords from the reverse direction is possible. For non-reversible codewords, recovery of data occurring after the erroneous bits in the bitstream is not possible. Thus data till the

next resynchronization marker is lost, although there may not be any errors in it. However, reversible codewords make the decoding of the data having no errors occurring after the erroneous bits possible (see Figure 27). This way, the error resiliency of the system is increased. This feature becomes very important in erroneous transmission environments, such as wireless networks and the Internet. Generally speaking, using reversible codewords is less efficient in terms of compression ration, but shows better performance in the presence of losses.

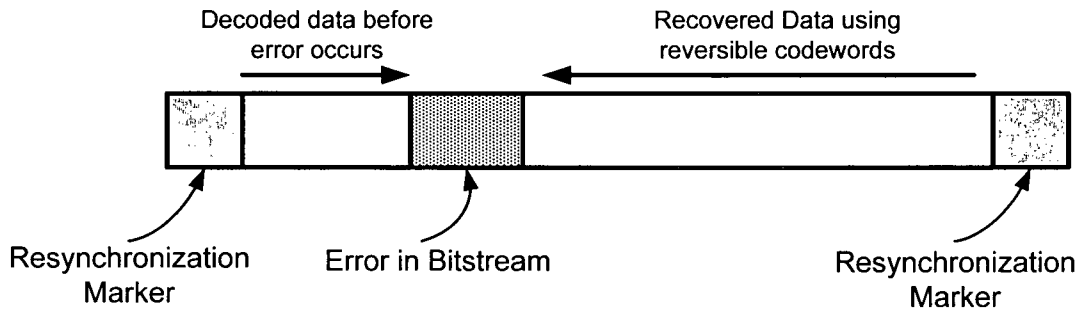


Figure 27 Data recovery using reversible codewords

2.3.2.2 Proposed Entropy Coding Structure

FGS standard uses four different VLC tables to code the (RUN,EOP) symbols resulting from bitplane coding (refer to Section 1.4.1.1 for details of bitplane coding). This entropy coding structured is replaced by the H.264 counterpart that uses Reversible Exp-Golomb codewords.

In order to construct a VLC table to entropy code an information source, one need to know the statistics of the symbols that the source generates. In our case, we want to code (RUN,EOP) symbols, resulting from the bitplane coding process. The FGS standard gives the statistics of these symbols based on DCT coding. As the DCT coding is replaced by Integer Transform in our proposed system, the statistics of the (RUN,EOP)

symbols have changed. Therefore, we have to collect the statistics of these symbols resulting after bitplane coding based on Integer Transform.

For this purpose, different sequences at different bitrates are encoded and the statistics for the (RUN,EOP) symbols resulting after bitplane coding of 4x4 Integer Transform coefficients are collected. The different sequences and the corresponding are presented Table 2.

Sequence	**Characteristic**	**Base Layer Bitrates**[1]	**Size**	**Number of Frames**
Tempete	Camera zooming out a flower, no motion, medium texture detail	500 Kbps, 1.5 Mbps	CIF (352x288)	300
Bus	Camera panning from left to right following a bus, medium-high motion and medium texture detail	500 Kbps, 1.5 Mbps	CIF (352x288)	300
Mobile	Camera following a toy train, low motion, very high texture detail	500 Kbps, 1.5 Mbps	CIF (352x288)	300

Table 2 Sequences used to gather statistics for (RUN,EOP) symbol

We present the statistics obtained from BUS sequence coded at 500 Kbps in Figure 28. The statistics for other cases can be found in Appendix D.

[1] At the time of this work, the H.264 codec did not have a mechanism for bitrate control. The target bitrate is achieved by changing the quantization parameter for the sequence, hence it does not represent the exact bitrate rather an approximate one.

Each figure is comprised of 4 graphs, denoting the statistics of the symbols at different bitplane levels. In the graphs, first half is for **EOP=0** and the second half is for **EOP=1**. By definition, the first bitplane does not contain an ALL_ZERO symbol. For the other bitplanes, ALL_ZERO symbol is shown after at the middle, just following symbols belonging to EOP=0.

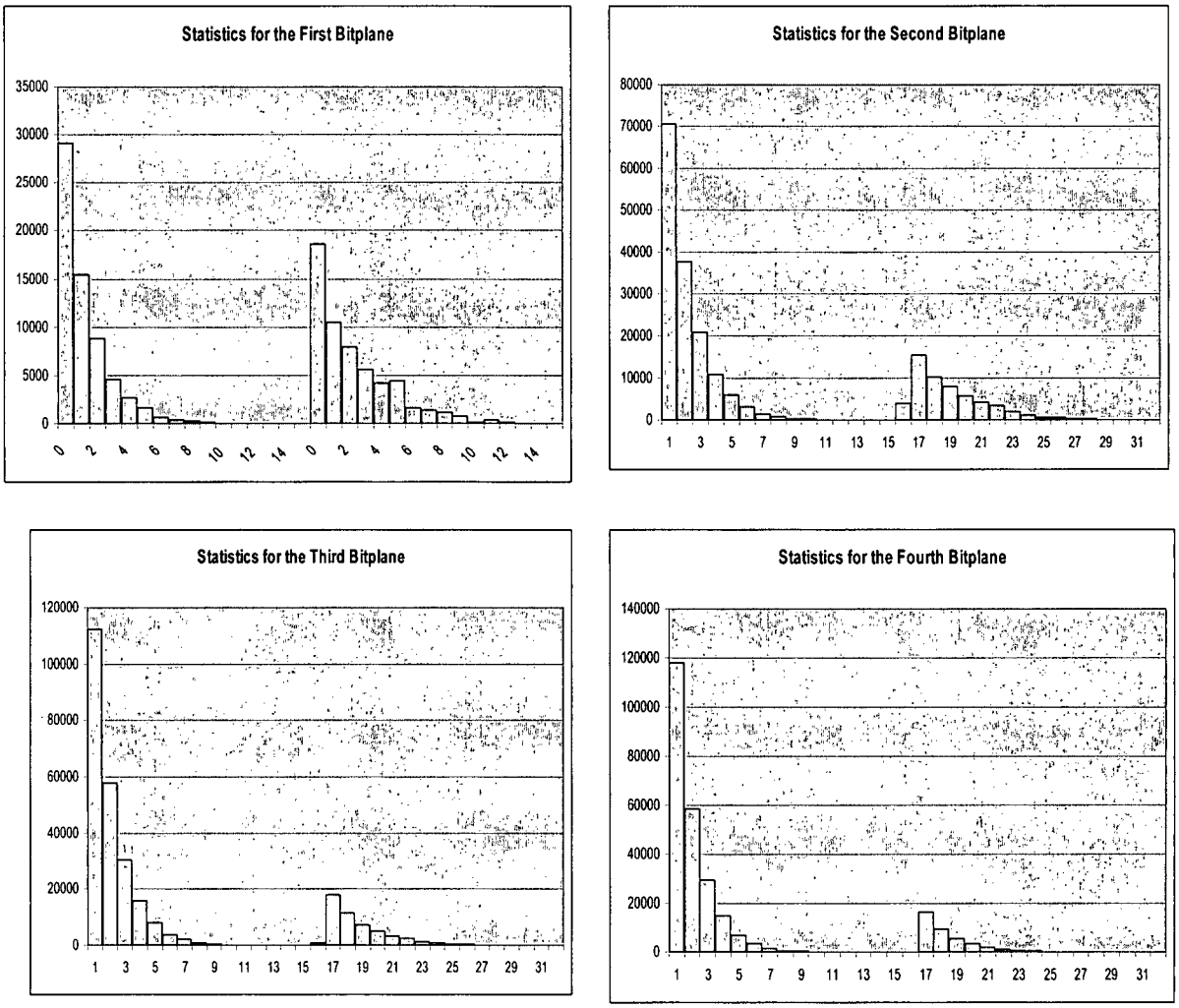


Figure 28 (RUN,EOP) Statistics for the BUS Sequence at 500 Kbps

MPEG-4 FGS uses four different VLC tables to code the (RUN,EOP) symbols. One table is used for the first bitplane, one for the second, one for the third and one for the

rest of the bitplanes (fourth and above). The reason for this is that the statistics of the symbols vary significantly among different bitplanes, and constructing new VLC table increases the coding efficiency of the system. For the proposed structure, it was observed that three VLC tables (one for the first bitplane, one for the second bitplane and one for the rest of the bitplanes) are enough to characterize the distribution of the symbols. There are two reasons for decreasing the number of VLC tables. The first reason is that the decrease in the number of possible symbols (RUN can be at most 15 in our case instead of 63, due to the smaller transform size), limits the amount of variation that symbols can possess. The second reason is, a slight variation in the statistics can not be captured by the Exp-Golomb codewords.

It is important to note that these tables, unlike in the case of MPEG-4 FGS, do not contain an **ESCAPE** code. MPEG-4 FGS uses **ESCAPE** to signal a symbol with large RUN value because the probabilities of a large RUN value are very small. Following the **ESCAPE** code, 6 bits are used to code the RUN and 1 bit for the EOP. In the proposed structure, the maximum value for RUN is 15 and the number of symbols is significantly lower when compared to MPEG-4 FGS. For this reason, we omit the use of **ESCAPE** in VLC coding.

After the statistics are obtained, the VLC tables are constructed. The codewords in the VLC tables are based on Reversible Exp-Golomb codewords and are the same as the H.264 uses. The following table presents the proposed VLC table for the first bitplane. The tables for the other bitplanes can be found in Appendix E.

Index	(RUN,EOP)	Code
0	(0,0)	1
1	(0,1)	010
2	(1,0)	011
3	(1,1)	00100
4	(2,1)	00101
5	(2,0)	00110
6	(3,1)	00111
7	(3 ,0)	0001000
8	(4,1)	0001001
9	(5,1)	0001010
10	(4,0)	0001011
11	(6,1)	0001100
12	(5,0)	0001101
13	(7,1)	0001110
14	(8,1)	0001111
15	(6,0)	000010000
16	(9,1)	000010001
17	(7,0)	000010010
18	(10,1)	000010011
19	(8,0)	000010100
20	(11,1)	000010101
21	(9,0)	000010110
22	(12,1)	000010111
23	(14,1)	000011000
24	(10,0)	000011001
25	(13,1)	000011010
26	(11,0)	000011011
27	(15,1)	000011100
28	(13,0)	000011101
29	(12,0)	000011110
30	(14,0)	000011111

Table 3 Proposed VLC Table for the First Bitplane

2.4 Experimental Results

In this chapter, we presented our proposed H.264 based FGS system. Our proposed system modifies the original FGS structure to achieve the best adaptation of FGS on H.264. We first replaced the DCT at the enhancement layer, with 4x4 Integer Transform to decrease the complexity of the system. This modification brings an overhead in CBP

coding at the macroblock header, due to the smaller transform size used. To code the CBP more efficiently, we developed novel Hierarchical CBP Coding Scheme that is presented in Section 2.3.1.1.1. Secondly, the entropy coding scheme of FGS is modified to achieve higher compression ratios, lower complexity and increased error resilience of the overall system.

In this section, we present experimental results of our proposed system. In order to provide a better comparison for each developed technologies, we present the experimental results in two subsections. Section 2.4.1 presents the performance of our proposed Hierarchical CBP Coding Scheme. In Section 2.4.2, we present the experimental results for the proposed entropy coding structure.

2.4.1 Experimental Results for the Proposed CBP Coding Scheme

The aim of the proposed Hierarchical CBP Coding scheme is to code the CBP more efficiently for our H.264 based FGS system, than the present scheme of the FGS standard. This section presents the detailed analysis about the amount of bit savings on CBP coding that can be achieved using our proposed scheme. For this analysis, we compare the number of bits used to code the CBP in our H.264 based FGS system using two different methods. First method is our proposed Hierarchical CBP Coding scheme that is presented in Section 2.3.1.1. Second method is the same CBP coding structure as in MPEG-4 FGS. However, for the second method the blocks within the macroblock are grouped as in Figure 19, so that each group's structure is identical to FGS macroblock structure. As also shown in Figure 19, the 24 (4x4) blocks within a macroblock are grouped into 4. Structure of each (8x8) group is the same as theMPEG-4 FGS macroblock structure. Thus, the same coding algorithm as MPEG-4 FGS CBP coding is

used to code the CBP for each of the groups. This approach results in using 4 CBP codewords for each macroblock. So the amount of bits spent for CBP is approximately quadrupled.

For this experiment, we use the test sequences presented in Table 2. Because the techniques for coding the CBP for the first and second bitplane are not the same, separate results are presented for each of the bitplanes.

The following tables present the average number of bits used for coding the CBP using both methods. It is seen that, more bits are allocated for coding the CBP at the second bitplane, no matter which method is used. This is due to the fact that, at the second bitplane, there are less ALL_ZERO but more non-zero blocks. This means, there are less blocks that can be grouped and coded as ALL_ZERO, decreasing the efficiency of the CBP coding. For the first bitplane, almost all the blocks are ALL_ZERO, and CBP can efficiently group and code them together, using less bits overall.

It is clearly seen that, the proposed Hierarchical CBP Coding Scheme outperforms the FGS CBP coding scheme significantly for the H.264 based FGS system on all the sequences. On average the proposed scheme uses 70% less bits than FGS scheme for coding the CBP at the first and second bitplanes. For some cases the proposed scheme uses up to 75% less bits than the FGS scheme.

As mentioned before, CBP coding scheme in the FGS standard is not suitable for our H.264 based FGS system. This is due to the increased number of transform blocks within the macroblock. The proposed Hierarchical CBP coding scheme, decreases this overhead significantly. This way, the 4x4 Integer Transform can be used more efficiently in the enhancement layer.

	Proposed CBP Coding Scheme		FGS CBP Coding Scheme	
	First Bitplane	Second Bitplane	First Bitplane	Second Bitplane
Bits Used for CBP Coding	450	2051	1653	6257

Table 4Average Number of Bits Used for CBP Coding for Bus Sequence

	Proposed CBP Coding Scheme		FGS CBP Coding Scheme	
	First Bitplane	Second Bitplane	First Bitplane	Second Bitplane
Bits Used for CBP Coding	513	2147	1953	7244

Table 5 Average Number of Bits Used for CBP Coding for Tempete Sequence

	Proposed CBP Coding Scheme		FGS CBP Coding Scheme	
	First Bitplane	Second Bitplane	First Bitplane	Second Bitplane
Bits Used for CBP Coding	339	1937	2004	7453

Table 6 Average Number of Bits Used for CBP Coding for Mobile Sequence

2.4.2 Experimental Results for the Proposed Entropy Coding Scheme

In this section, we compare the coding efficiency of the proposed entropy coding scheme that was presented in Section 2.3.2, with the one standardized in FGS. For this comparison we encode different sequences at different base layer bitrates. Two different methods are used for entropy coding: i) Proposed Entropy Coding Method and ii) Original FGS Entropy Coding Method. The sequences used for this experiment are presented in Table 2.

After encoding the different sequences with both methods, the number of bits used to code the (RUN,EOP) symbols at different bitplane levels are found for each method.

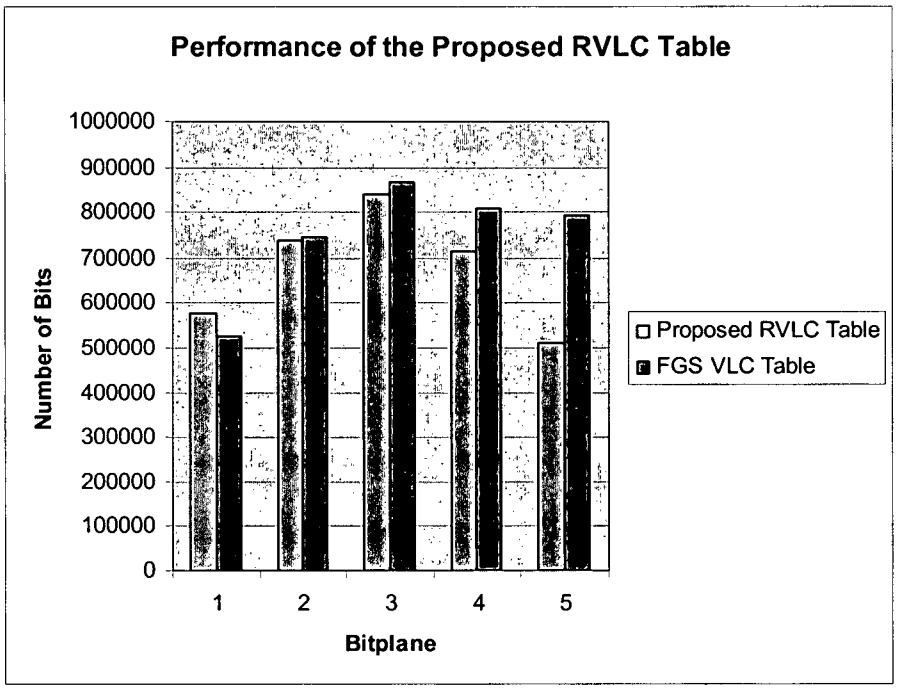
Table 4 presents the results for all the different sequences, for different bitplane levels. Each row of Table 4 presents results for a sequence encoded at a specific base layer bitrate. The columns of the table are grouped into four, and each group presents the results for one bitplane level (first, second, third and the fourth bitplanes). The number of bits used to code symbols using two different methods of entropy coding methods, is presented side by side for each case. We also illustrate the results for each sequence separately at Table 8,Table 9 and Table 10. In these tables, there are two figures for each sequence, representing the different bitrates that the base layers are encoded at.

When compared with FGS entropy coding method, the proposed method uses 7% less number of bits on average to code the (RUN,EOP) symbols. For all the sequences, the proposed entropy coding method outperforms FGS entropy coding at all bitplane levels, except for the first one. At bitplane levels higher than the first one, the performance of the proposed table goes up to 22% better than the standard FGS. Also, as mentioned previously in this section, the proposed method has high resilience to errors and less computational complexity.

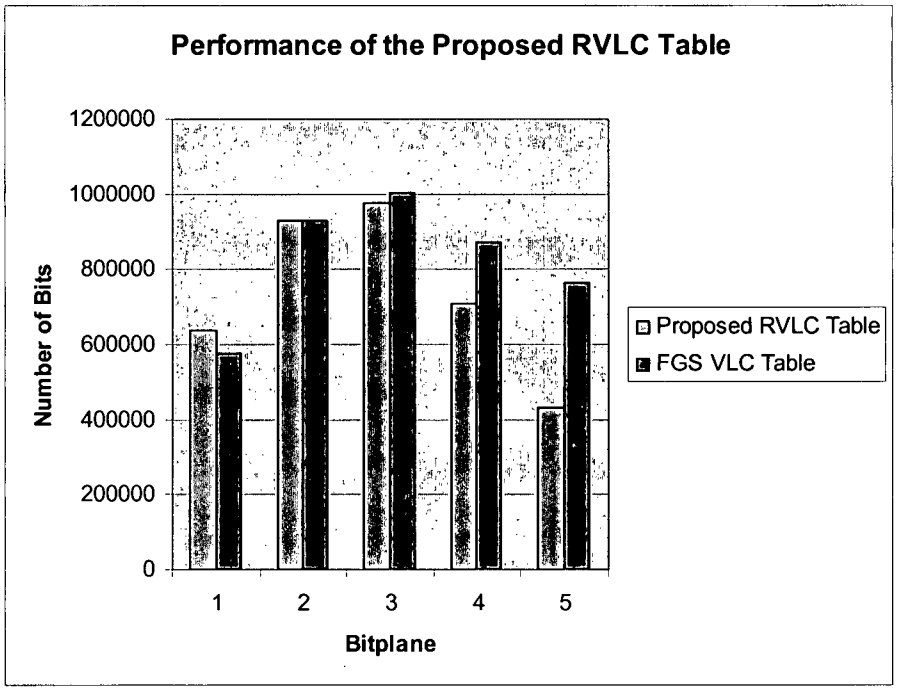
In conclusion, our proposed entropy coding scheme that is based on 4x4 Integer Transform, achieves 7% coding efficiency gain on average with increased error resiliency and less computational complexity over the FGS entropy coding method, in our H.264 based FGS system.

	Bitplane 1			Bitplane 2			Bitplane 3			Bitplane 4		
	Bit Usage Proposed	Bit Usage FGS	Percent Improvement	Bit Usage Proposed	Bit Usage FGS	Percent Improvement	Bit Usage Proposed	Bit Usage FGS	Percent Improvement	Bit Usage Proposed	Bit Usage FGS	Percent Improvement
Tempete, 500 Kbps	511615	481014	-6.4%	735792	751718	1.8%	808554	834000	3.0%	804711	899462	10.8%
Tempete, 1.5 MBps	584541	537204	-8.8%	876300	882159	0.7%	958044	987387	3.0%	658771	792700	16.9%
Bus, 500 Kbps	487332	460814	-5.7%	734221	757194	3.0%	909247	946014	3.9%	799157	914871	22.1%
Bus, 1.5 Mbp	644091	597791	-7.7%	885134	899774	1.6%	964557	1001262	3.7%	791827	1016858	22.1%
Mobile, 500 Kbps	577955	527785	-9.5%	737767	745780	1.0%	840804	867021	3.0%	714023	809052	11.7%
Mobile, 1.5 Mbps	637521	576352	-10%	929701	929701	0%	974428	1004211	3.05	709261	871830	18.6%

Table 7 Coding Efficiency of Reversible Exp-Golomb Codewords

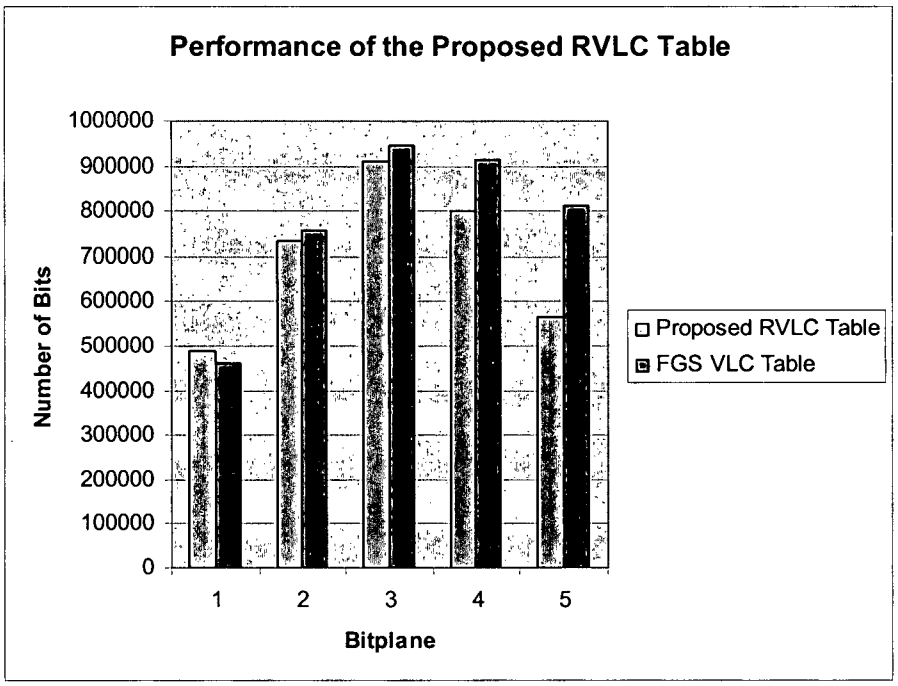


(a)

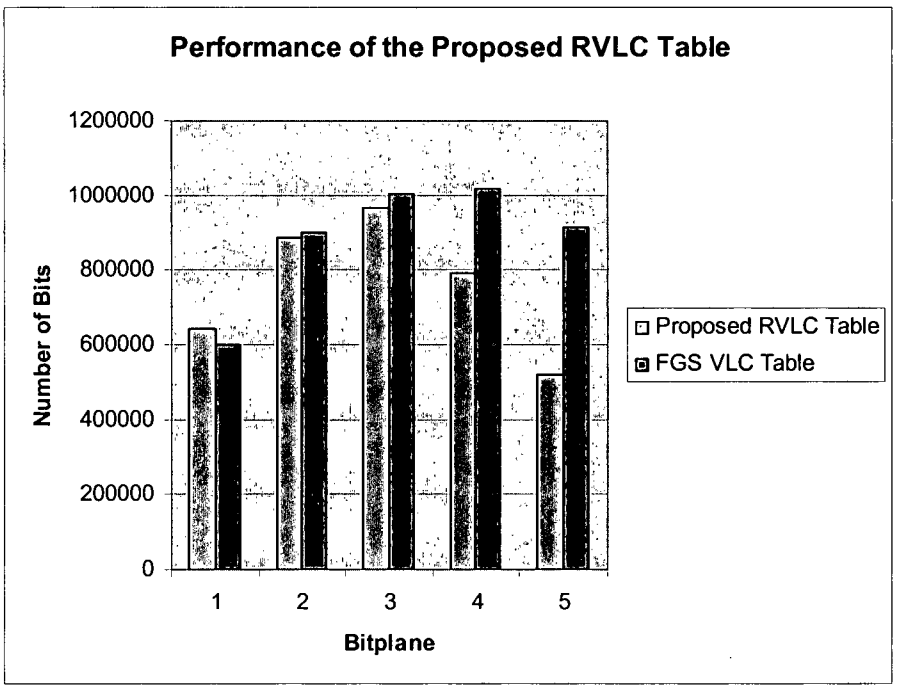


(b)

Table 8 Coding Efficiency Test Results for the Proposed RVLC Table. Sequence is Mobile coded at (a) 500 Kbps (b) 1.5 Mbps

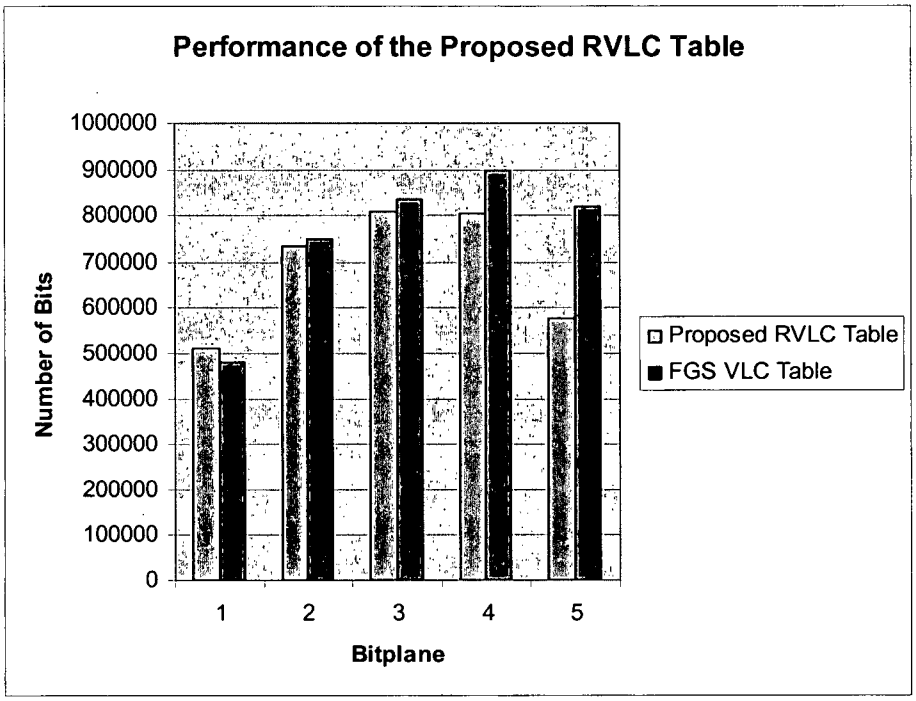


(a)

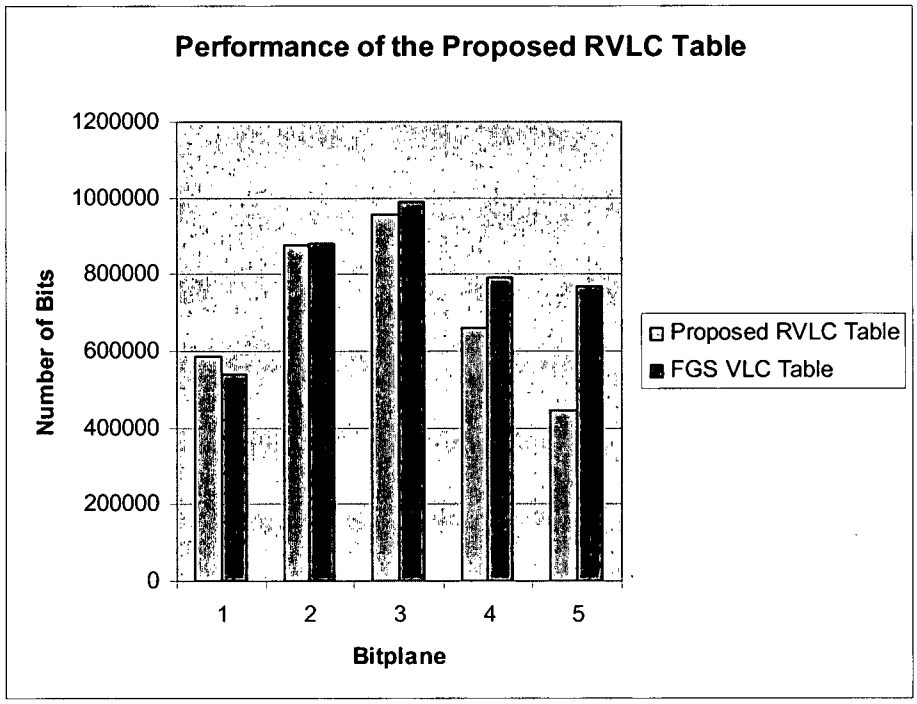


(b)

Table 9 Coding Efficiency Test Results for the Proposed RVLC Table. Sequence is Bus coded at (a) 500 Kbps (b) 1.5 Mbps



(a)



(b)

Table 10 Coding Efficiency Test Results for the Proposed RVLC Table. Sequence is Tempete coded at (a) 500 Kbps (b) 1.5 Mbps

2.5 Conclusion

In this chapter, we presented our proposed H.264 based FGS structure. Instead of simply extending the FGS to use H.264 at the base layer, the proposed structure modifies the FGS coding blocks to take full advantage of H.264's superior features present in the base layer. The proposed modifications can be grouped under two groups:

1. DCT is replaced by 4x4 Integer Transform at the enhancement layer
2. Entropy Coding structure is modified to use the Reversible Exp-Golomb coding technique

We replaced the DCT at enhancement layer by 4x4 Integer Transform, introduced the novel Hierarchical CBP Coding structure, that significantly decreases the overhead caused by using 4x4 Integer Transform. The VLC tables at the enhancement layer are changed and the new tables are built with Reversible Exp-Golomb codewords using the symbol statistics resulting from 4x4 Integer Transform.

By modifying the standard FGS structure, the complexity of the encoder-decoder pair is decreased and the error resilience of the overall system is increased. Performance evaluations have shown that our method also improves the coding efficiency of the system by 7% on average.

CHAPTER 3

3 Hybrid Structure using Stream-Switching and FGS for Scalable H.264 Video Transmission

As mentioned before, the H.264 video coding standard lacks scalability, i.e., video adaptation to different bitrates. Instead, H.264 uses a different approach, called stream-switching, to cope with the fluctuations of the available bandwidth of the underlying network upon which media information is transmitted. In the stream-switching approach, the video is independently coded into several non-scalable bitstreams of different bit-rates. The system dynamically switches between these different bitrate coded video versions depending on the bandwidth availability. The advantage of this method is its high coding efficiency, which results from the independent coding of non-scalable bitstreams. However, this method provides a coarse capability in adapting to changing bandwidth conditions due to the limited number of bitstreams. There are two main reasons for having a limited number of bitstreams. The first is due to fact that the encoder needs to encode the original bitstream at different rates. This increases the complexity of the system and therefore there is a trade off between the number of bitstreams that can be offered and the cost of the system. Second, since all the generated bitstreams need to be stored at a streaming server, storage requirements may be a limiting factor. For the stream switching approach, the H.264 video coding standard has specified special key-frames, called Synchronization-Predictive (SP) frames that allow efficient switching between video bitstreams [17].

In this chapter, we present a unique method of combining the FGS scalable video coding with the stream-switching techniques to maximize the video quality for the end user. Unlike other proposed scalable switching systems proposed, our proposed system is based on established standards. This means that streams created by this proposed method can still be processed by an existing H.264 decoder that supports switching of streams and does not have FGS capability. We also have developed a novel algorithm to select the rates of the base layer streams adaptively. The proposed algorithm involves encoding the video at different rates with different enhancement layer streams and R-D performance analysis of these streams. Results of this analysis are used to determine the optimal rates at which the base layers should be encoded and where the switching between streams takes place.

In Section 3.1, we first give a brief overview of the stream-switching technique and the SP-frame concept used in H.264. Also, a brief comparison of stream-switching and scalable video coding is presented, along with the advantages and disadvantages of both methods. Section 3.2, presents our proposed hybrid approach and the novel adaptive rate selection algorithm. Section 3.3 presents the performance evaluation of the proposed approach and compares it with:

1. FGS enabled H.264 video compression system without switching capability as proposed in Chapter 2, and with
2. H.264 video compression system with stream-switching capabilities only as proposed by the H.264 standard (i.e., without FGS support).

3.1 Stream-Switching and SP-Frames

3.1.1 Overview of Stream-Switching

Stream-switching is a technique used in video communication systems to cope with bandwidth variations. In this technique, video is independently coded to several streams at different bitrates and quality levels. After encoding, the streaming server dynamically switches between the streams, according to the available bandwidth in order to accommodate the bandwidth variations.

One important restriction of stream-switching is that the streaming server can not switch the stream at arbitrary frames, but only at key frames. The reason for this is the temporal predictive coding techniques of present video coding standards require that the frame being coded to depend on previous frames. Let's consider an example where there are two bitstreams generated independently at different quality levels. Let $\{..., P_{1,n-1}, P_{1,n}, P_{1,n+1}, ...\}$ and $\{..., P_{2,n-1}, P_{2,n}, P_{2,n+1}, ...\}$ denote the sequence of the decoded frames from two bitstreams, bitstream 1 and bitstream 2 respectively. Let's also assume all these frames are P-frames and that switching takes place at time instant n, i.e., the server sends $\{P_{1,n-1}, P_{2,n}, P_{2,n+1}\}$. In this case, the decoder can not decode $P_{2,n}$ correctly, since the reference frame to encode the frame $P_{2,n}$, which is $P_{2,n-1}$ is not received. This mismatch leads to erroneous decoding which further propagates due to motion compensation.

For this reason, in existing video coding standards, switching of bitstreams is only made possible at frames that do not use information prior to their location, i.e., I frames. However, placing I frames periodically in the bitstream reduces the coding efficiency, as these frames do not exploit any temporal redundancy. The H.264 standard introduces a new frame type, called SP-frame. SP-frames make use of motion compensated predictive

coding so to exploit the temporal redundancy in the sequence, in a similar manner to that of P-frames. However, identical SP-frames can be reconstructed even when different reference frames are used for their prediction [17]. In the next subsection, we describe the SP-frame concept introduced in the H.264 video coding standard.

3.1.2 Overview of the SP Frame Switching Concept used in H.264

The stream-switching operation is realized by placing keyframes that do not use information prior to their corresponding temporal locations. This approach, however, decreases the coding efficiency of the system, as these keyframes do not exploit the temporal redundancies of the video sequence.

H.264 introduced a new frame type, called SP-frame, for this purpose. Similar to P-frames, SP-frames make use of motion compensated predictive coding to exploit temporal redundancy in the video sequence. However, unlike P-frames, SP-frames allow identical frames to be reconstructed even when they are predicted using different reference frames. This property of SP-frames allows them to be used instead of I-frames in stream-switching applications. In this section, the technical details of SP-frames are overviewed. It should be noted that, SP-frames can be used for other applications such as random access, error recovery and error resiliency, but only the stream-switching application is considered here.

In order to explain how SP-frames are used during stream-switching, consider an example illustrated in Figure 29. Let's assume that a bitstream is encoded twice at two different bitrates. Their corresponding frames are denoted by $\{P_{1,1}, P_{1,2}, SP_{1,3}, P_{1,4}, P_{1,5}\}$ and $\{P_{2,1}, P_{2,2}, SP_{2,3}, P_{2,4}, P_{2,5}\}$ for the first and second bitstreams, respectively (see Figure

29). In each bitstream, SP-frames are placed at the same temporal location that switching is desired to take place (in this case it is $SP_{1,3}$ and $SP_{2,3}$).

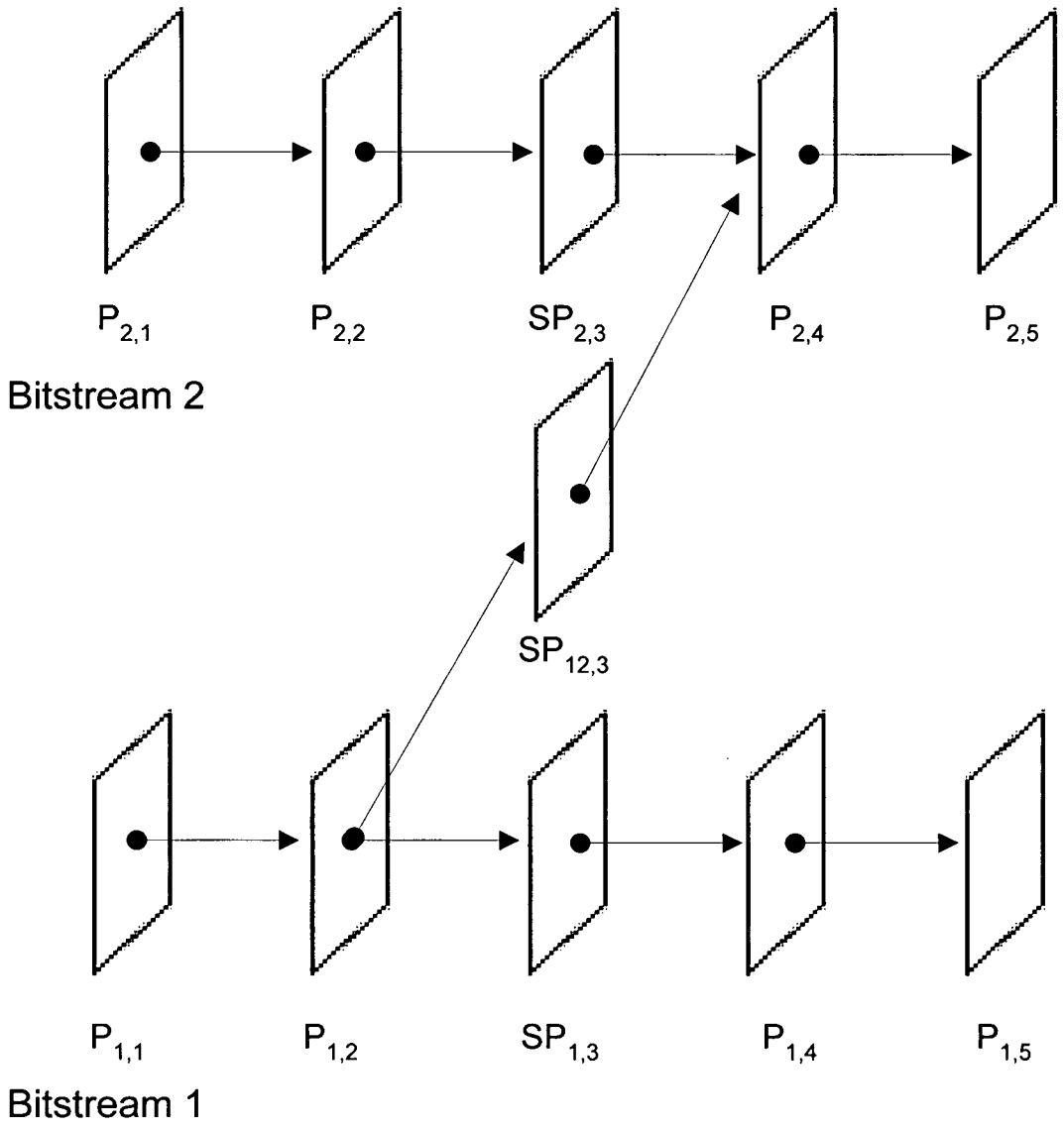


Figure 29 Switching between streams using SP-Frames

SP-frames placed within a bitstream are called *primary SP-frames*. For each primary SP-frame, another SP-frame, called *secondary SP-frame*, which allows switching from that bitstream to another bitstream, is generated. The secondary SP-frames are used only during switching. At the streaming server, two bitstreams (bitstream 1 and bitstream 2)

and all the secondary SP-frames needed for switching are stored. At the time of switching, the streaming server sends the secondary SP-frame corresponding to the stream that the server switches to. For example, if we switch from $P_{1,2}$ to $P_{2,4}$, then the *secondary SP frame* $SP_{12,3}$ is used in between (see Figure 29). Similarly if we switch is from $P_{2,2}$ to $P_{1,4}$ then a different SP frame $SP_{21,3}$ will be used. Secondary SP-frames result in the same future frames as a primary SP-frame even though they use a different reference frame. At the time of switching, the decoder receives the secondary SP-frame ($SP_{12,3}$), with its reconstruction identical to its respective primary frame ($SP_{2,3}$). The next frame the decoder receives just after switching is $P_{2,4}$ and it uses $SP_{2,3}$ as reference. The decoding process continues normally without any error, as the reconstruction for frames $SP_{2,3}$ and $SP_{12,3}$ is identical although they use different reference frames.

3.1.3 Comparison of Stream-Switching and Scalable Video Coding

In this section we compare the two approaches *i) Stream-Switching* and *ii) Scalable Video Coding*. In particular, the SP-frame approach introduced in H.264 and the FGS approach are considered.

The common objective of these two approaches is to cope with the bandwidth variations and offer optimum video quality to the end user. However, the way that FGS tries to achieve this objective is quite different from that of the stream-switching approach. The latter was proposed and described in Chapter 2.

The FGS encoder generates one bitstream that contains the enhancement and the base layers. The base layer is encoded at a bitrate, R_{base} and the enhancement layer is encoded using bitplane coding at a maximum bitrate, R_{max}. The bitstream is scaled at the FGS streaming server by truncating the enhancement layer portion of the video according to

the available bandwidth. The video quality at the end user is directly proportional to the amount of the enhancement layer information being sent. Main advantages of this method are its low complexity and its high flexibility. Low complexity is due to the fact that the same encoded bitstream is used for all the different bitrates, and thus encoding is performed only once. In addition, there is minimal overhead for the streaming server, as it should only do simple truncation to achieve scalability. FGS is highly flexible since the streaming server can truncate the enhancement layer to any desired bitrate, maximizing the bandwidth utilization using all the available bandwidth to send video information. As its name implies, FGS allows scalability in a fine-granular manner. Unlike other layered scalable technologies implemented in previous video coding standards, FGS video quality can be adjusted to any bitrate between R_{base} and R_{max}.

The main disadvantage of FGS video coding is its low coding efficiency when compared to single layer coding. In particular, for bitrates that are considerably higher than the base layer bitrate R_{base}, the penalty in coding efficiency becomes significant. This is because low quality base layer frames are used as references for motion estimation, and as a result, the temporal redundancies in the enhancement layer are not fully exploited.

The other approach, named stream-switching, simply encodes the same video with different quality levels and bitrates. The streaming server switches dynamically between the streams to accommodate the variations of the available bandwidth. For bandwidths that are considerably high, high-quality video is sent to the end user. If the available bandwidth drops, the server switches to the low quality version of the video. Based on the reasons discussed in the previous section, switching can take place only at key frames.

Thus, the system's response time to a bandwidth variation is low when compared to that of the scalable video coding approach.

When compared to scalable video coding, stream-switching approach can not use a single bitstream, but rather two or more bitstreams with different bitrates. That's why stream-switching involves more computational complexity than scalable video coding, as the encoding process should be repeated two or more times, depending on the number of bitstreams used. One other disadvantage of stream-switching is the insufficient bandwidth utilization achieved by the system. It should be noted that, the bandwidth utilization of a stream-switching system increases with the number of bitstreams used, but the larger the number of streams the more impractical the system becomes.

When compared with FGS, the stream-switching approach can not adapt to bandwidth variation in a fine-granular way, as it is not based on scalable coding of the video. Despite all the disadvantages, the stream-switching technique has very high coding efficiency (due to independent coding of non-scalable video), which makes it very attractive.

The following table summarizes the advantages and disadvantages of both approaches, providing a quick overview.

	Stream – Switching	**Scalable Video Coding (FGS)**
Coding Efficiency	High, due to independent coding of non-scalable bitstreams.	Low at increased enhanced layer bitrates, due to low quality reference frames used in motion estimation.
Bandwidth Utilization	Low – due to limit in number of streams.	High, close to 100%.
Response time to change in bandwidth	Low - Can only adapt to bandwidth change at key frames.	High - Depending on streaming server, it can adapt instantly.
Scalability Step	Coarse capability in adapting to changing bandwidth.	Fine Granular.
Computational Complexity	Encoding should be performed several times depending on the number of bitstreams.	Encoding is performed once, for base and enhancement layer.

Table 11 Comparison of Stream-Switching Approach with Scalable Video Coding Approach

In the next section, we propose a hybrid method that is a combination of the FGS method with Stream-Switching. The proposed hybrid method takes advantage of both methods to improve bandwidth utilization and video quality.

3.2 Combining Stream-Switching and FGS

The main disadvantage of FGS is its low coding efficiency at high enhancement layer bitrates, which is due to the low quality reference frames used in motion estimation. The main disadvantage of stream-switching is its coarse capability in adapting to bandwidth changes and its low bandwidth utilization. We aim to eliminate those two disadvantages with our combined FGS – stream-switching architecture. Thus, the combined system is not only scalable and can adapt to bandwidth changes in a fine granular way, but it also has high coding efficiency.

Figure 30 illustrates the architecture behind our proposed hybrid method for the case of two bitstreams. Our system encodes the video into two independent scalable bitstreams. Each bitstream is an H.264 based FGS stream consisting of a base layer and an enhancement layer.

Based on the available bandwidth, the streaming server sends one of the base layers along with its corresponding enhancement layer portion. If there is a low bandwidth variation that can be accommodated by the enhancement layer, the streaming server continues to send the same base layer along with its enhancement layer. However, the streaming server switches between scalable bitstreams if high bandwidth variations occur.

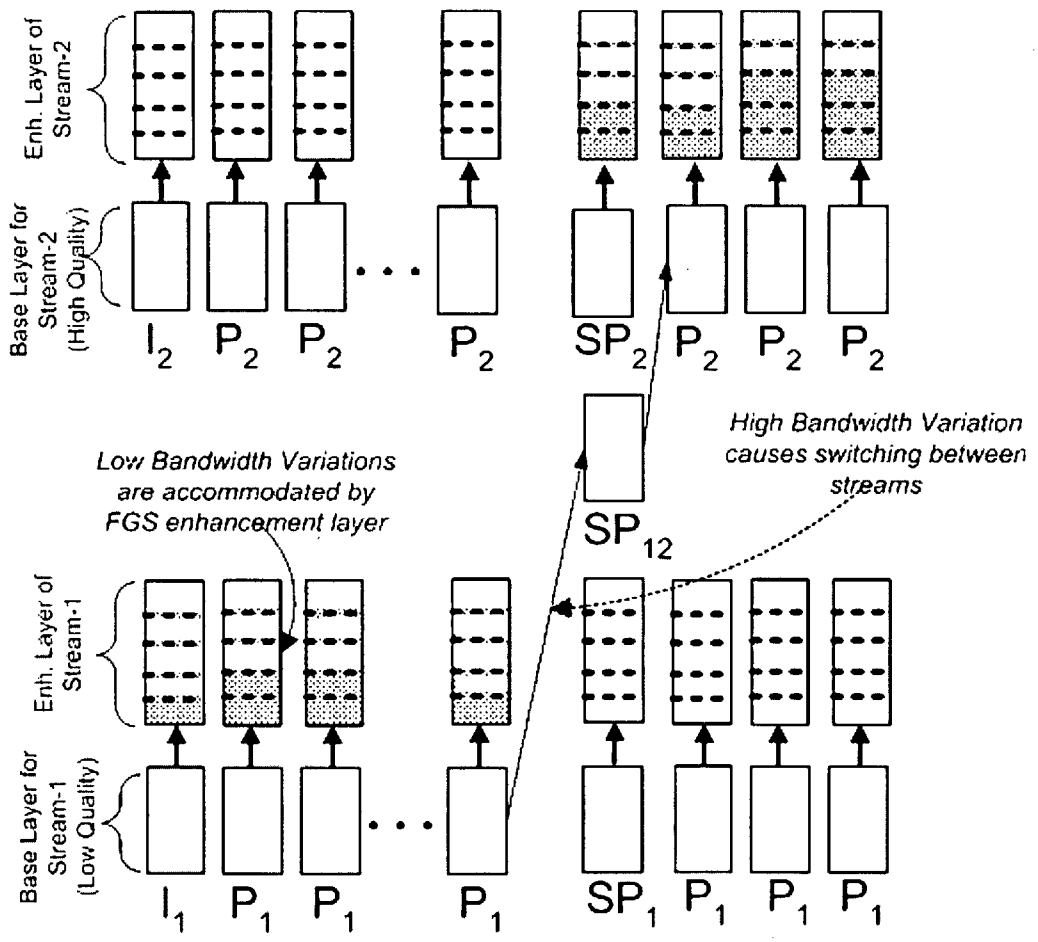


Figure 30 Structure of the proposed hybrid system

Therefore, the streaming-server performs both switching and scaling operations. As can be observed from Figure 30, the low bandwidth variation is accommodated by FGS, but if the variation exceeds a certain threshold, it causes the system to switch streams. This structure increases the overall efficiency since:

1) bandwidth utilization is always at 100%, and
2) the picture quality increases with FGS enhancement layer portion operating at its higher efficiency regions.

Assume that the network bandwidth changes dynamically in the range of $[R_{\min} - R_{\max}]$. The base layer bitrates are R_{base_1} and R_{base_2} for the base layers of low and high quality

scalable streams respectively, where $R_{\min} \le R_{base\_1} \le R_{base\_2} \le R_{\max}$. If at a given time instant, the available bandwidth $R_{available}$ is greater than the low quality base layer bitrate but lower than the high quality base layer bitrate, that is $R_{\min} \le R_{base\_1} \le R_{available} \le R_{base\_2}$, then the streaming server sends the low quality video and truncates its corresponding scalable stream to utilize the rest of the available bitrate. So the amount of enhancement layer transmitted is $R_{enhancement\_1} = R_{available} - R_{base\_1}$.

One of the challenges in the proposed system and in stream-switching systems in general, is how to choose the bitrates for the independent streams. In the next subsection, we present a novel adaptive rate selection method for stream-switching. For the sake of simplicity, only two independent not scalable bitstreams are considered. Later, the algorithm is generalized to include more than two bitstreams that also have their scalable enhancement layer information.

3.2.1 Adaptive Bitrate Selection for Stream-Switching

The bitrates of the streams that are used for switching are an important parameter that affects the performance of the system in a dynamic environment. Assume the available bandwidth fluctuates in the range $[R_{\min} - R_{\max}]$, and two streams, one having low quality and the other with higher quality, will be used to cover this bandwidth range, with bitrates R_1 and R_2 respectively. One condition for the bitrates is $R_{\min} \le R_1 \le R_2 \le R_{\max}$. Also, the bitrate of the lowest quality stream should not be higher than the minimum available bandwidth to be able to send a stream at any given available bandwidth (i.e., $R_1 = R_{\min}$). Thus, for two streams, only the bitrate of the higher quality stream is variable. The bandwidth range can simply be divided into equal portions and can be half the

fluctuating bandwidth range $R_2 = 0.5(R_{\max} + R_{\min})$. This technique can be used for n streams with straightforward extension. However, this bitrate selection does not use any distortion measure and may not guarantee the best R-D performance. We developed an adaptive rate selection by analyzing the encoded video quality at different rates. In the case of two streams, we are seeking the rate for the higher quality stream, R_2 which minimizes the total distortion at the fluctuating bandwidth range given by:

$$\sum_{R_{\text{aval}}=R_{\min}}^{R_{aval}=R_2} D_1 + \sum_{R_{aval}=R_2}^{R_{aval}=R_{\max}} D_2, \; R_{\min} < R_2 < R_{\max} \qquad (3.1)$$

where D_1 and D_2 are calculated distortions of the low and high quality decoded streams respectively. The distortion measure D is the *mean square error* and is given by the following equation:

$$\mathrm{D} = \frac{1}{\mathrm{M.N}} \sum_{i=1}^{M} \sum_{j=1}^{N} (I(i,j) - K(i,j))^2 \qquad (3.2)$$

where I and K are the two pictures and M and N are their height and width respectively.

3.2.2 Generalized Adaptive Rate Selection

Generalization of the above problem for n streams requires finding the rates $(R_1, R_2 \ldots R_n)$ where $R_1 = R_{\min}$, $R_1 \le R_2 \ldots \le R_{n-1} \le R_n$ and $R_{n+1} = R_{\max}$ to minimize the total distortion given by Equation (3.3):

$$\sum_{i=1}^{n} \Big(\sum_{R_{aval}=R_i}^{R_{aval}=R_{i+1}} D_i \Big) \qquad (3.3)$$

In order to accommodate the R-D characteristics of the FGS enhancement layer for adaptive rate selection, the problem is similar and entails finding the bitrates $(R_1, R_2 \ldots R_n)$ where $R_1 = R_{\min}$, $R_1 \le R_2 \ldots \le R_{n-1} \le R_n$ and $R_{n+1} = R_{\max}$, except that now the total distortion measure is a modified version of Equation (3.3):

$$\sum_{i=1}^{n} \left(\sum_{R_{aval}=R_i}^{R_{aval}=R_{i+1}} D_i^{bp\_j} \right) \tag{3.4}$$

where $D_i^{bp\_j}$ is the distortion of the i^{th} *base* stream and *bp_j* is the number of bitplanes at the enhancement layer. The number of bitplanes sent is the maximum number that can be sent for the given base layer stream given an available bandwidth, R_{aval}.

3.3 Experimental Results

We compare the proposed hybrid FGS, Stream-Switching approach with the FGS and the Stream-Switching approaches separately. In order to evaluate the performance of the proposed algorithm, we consider a transmission channel where the available bandwidth changes dynamically. For this specific experiment, the available bandwidth is simulated to increase from 30 Kbps up to 250 Kbps and then decrease back to 30 Kbps in the course of 90 frames, which corresponds to 3 seconds at a rate of 30 frames/second. These test conditions are specified by MPEG Scalable Video Coding group at the recent Call for Proposals on Scalable Video Coding Technology [25], as one of the experiment test conditions.

For our experiment, we first encode the *Foreman* sequence at different bitrates and we encode their corresponding enhancement layers with the proposed H.264 based FGS encoder presented in Chapter 2. Then, our adaptive rate selection algorithm is used to determine the best bitrate at which switching between bitstreams is performed.

Afterwards, depending on the available bandwidth at a given time instant, the network simulator chooses the base and enhancement layer streams to send to the decoder. Below, we evaluate the performance of the following three methods:

1. Proposed Combination of FGS with Stream-Switching approach using Adaptive Rate Selection
2. Scalable Video Coding using FGS
3. Stream-Switching using SP Frames

The parameters for the base layer encoding are presented at Table 12.

SP Picture Periodicity:	At every 15 frames (i.e. at every half a second for 30 fps. video)
Quantization Parameter for S_1 frames: (QPSP)	Same as Quantization Parameter for P Frames (QP)
Quantization Parameter of S_{12} frames: ($QPSP_2$)	QPSP-6
Frame Structure:	I P P P...P P P SP P P P ...

Table 12 Encoding Parameters for H.264 Base Layers

We first compare the performance of the three approaches in a channel, where the available bandwidth changes over time. The available bandwidth first increases from 30 Kbps to 250 Kbps through frames 1 to 45 and starts to decrease back to 30 Kbps at frame 45. The base layer bitrate for FGS and the low quality bitrate for the proposed and stream-switching approaches are the same and equal to the lowest bandwidth that the channel can deliver (30 Kbps). For the stream-switching approach, the bitrate for the high quality bitstream is found to be 100 Kbps using Equation (3.3), which describes the

adaptive rate selection for non-scalable bitstreams. For our proposed approach, equation (3.4) is used to find the high quality base layer bitrate which was found to be same as that of the stream-switching (i.e., 100 Kbps).

The performance of the three tested methods is shown in Figure 31. When compared with FGS, the enhancement layer performance of the proposed approach is relatively high due to switching at a key bit-rate. This results in a higher overall efficiency. The average PSNR gain is 2.9 dB, while for some frames the gain goes up to 3.5 dB. In addition, when our hybrid method is compared with the stream-switching approach, it is clearly seen that the video quality keeps increasing as the available bandwidth increases. This is because our method fully utilizes the available bandwidth. In this case, the average PSNR gain is 1.5 dB and can go up to 3 dB for some frames.

The R-D performance of the proposed approach is further analyzed and compared with the Stream-Switching and FGS approaches in Figure 32 and Figure 33, respectively. It is clear from Figure 32, that the advantages of the proposed approach over Stream Switching are mainly the increased bandwidth utilization and granular adaptation of the system to the varying bandwidth. When compared with FGS, the proposed approach does not suffer from the low coding efficiency at high enhancement layer bitrates as it is seen from Figure 33.

In our final experiment, we analyze the performance of the proposed adaptive rate selection algorithm. We use the proposed hybrid method and use two algorithms to find the bitrate of the high quality video (adaptive and non-adaptive rate selection algorithms). For the non-adaptive case, the bitrate of the high-quality base layer is found by dividing the bandwidth range into two (i.e., (30+250)/2 Kbps). Figure 34 illustrates

the results for this experiment. On average, the adaptive rate selection algorithm results in, 1 dB performance increase.

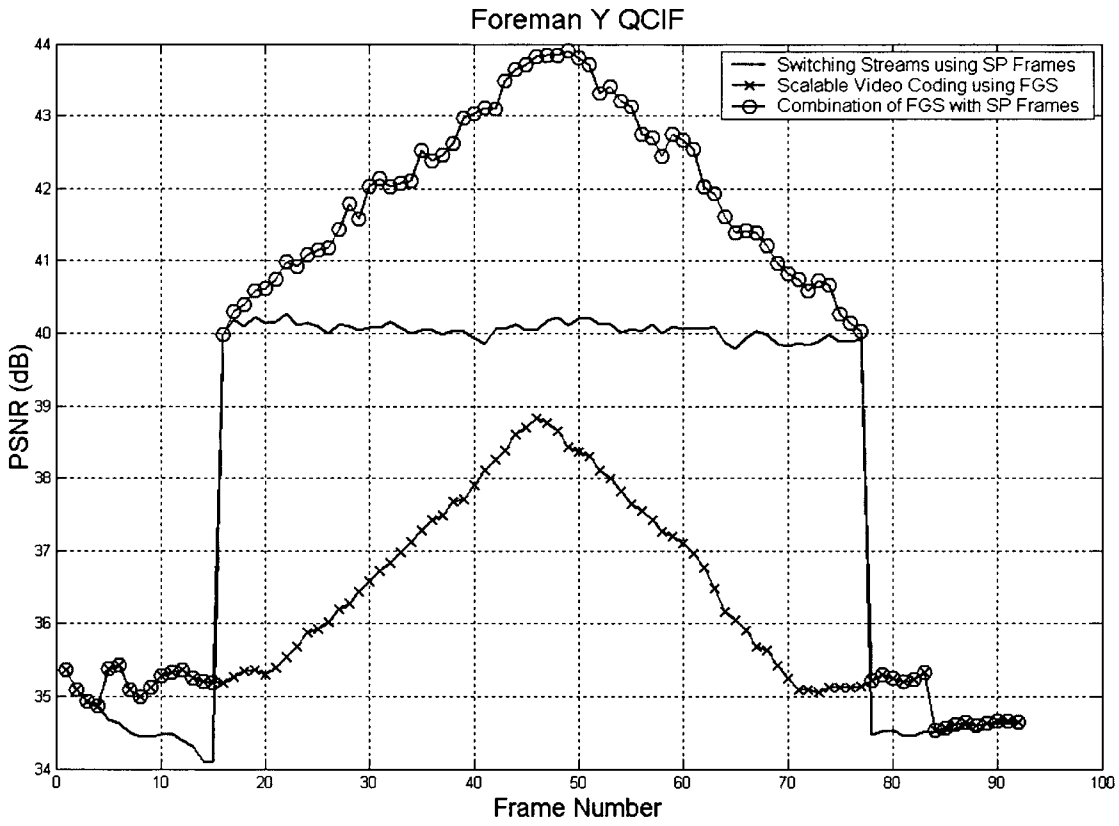


Figure 31 Performance of the Proposed Approach Compared with two other approaches i. Scalable Video Coding using FGS ii. Stream-Switching using SP Frames

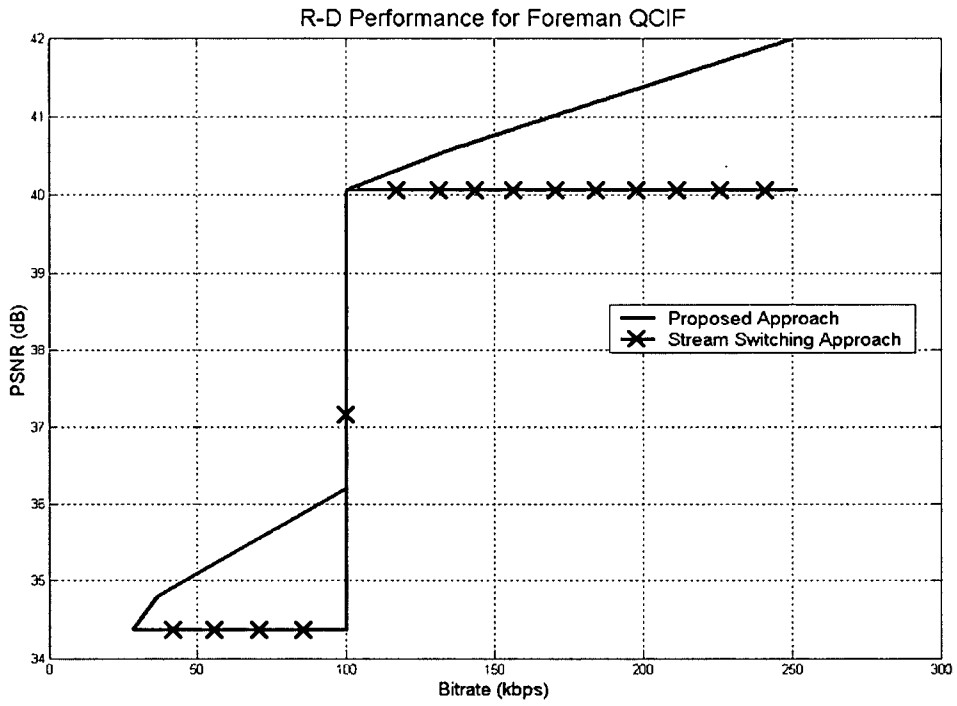


Figure 32 R-D Performance Comparison of the Proposed and Stream-Switching Approach

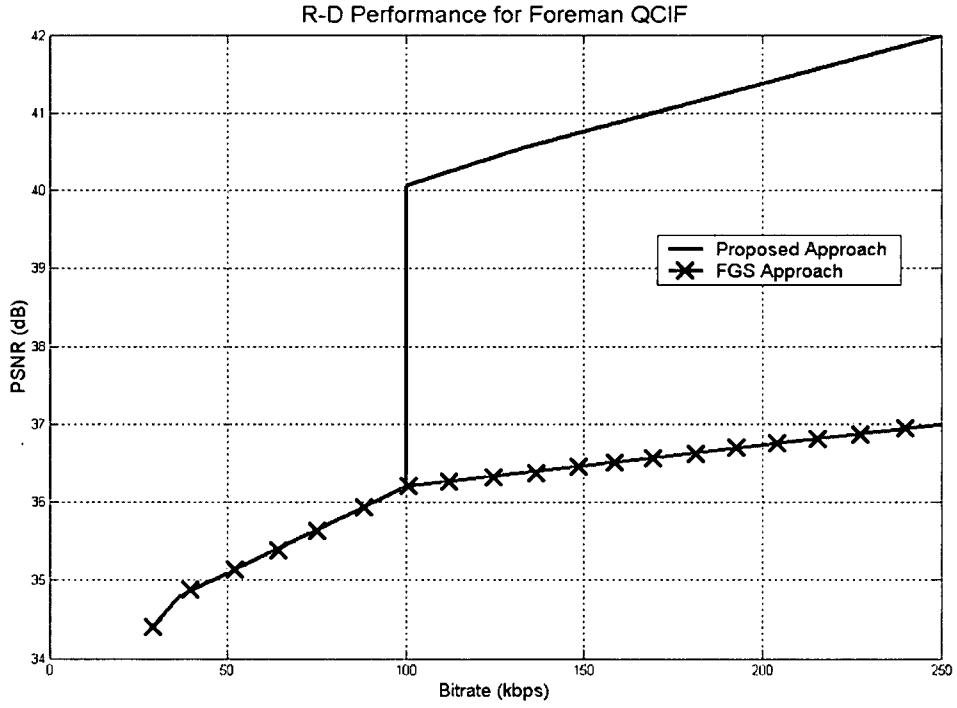


Figure 33 R-D Performance Comparison of the Proposed FGS Approach

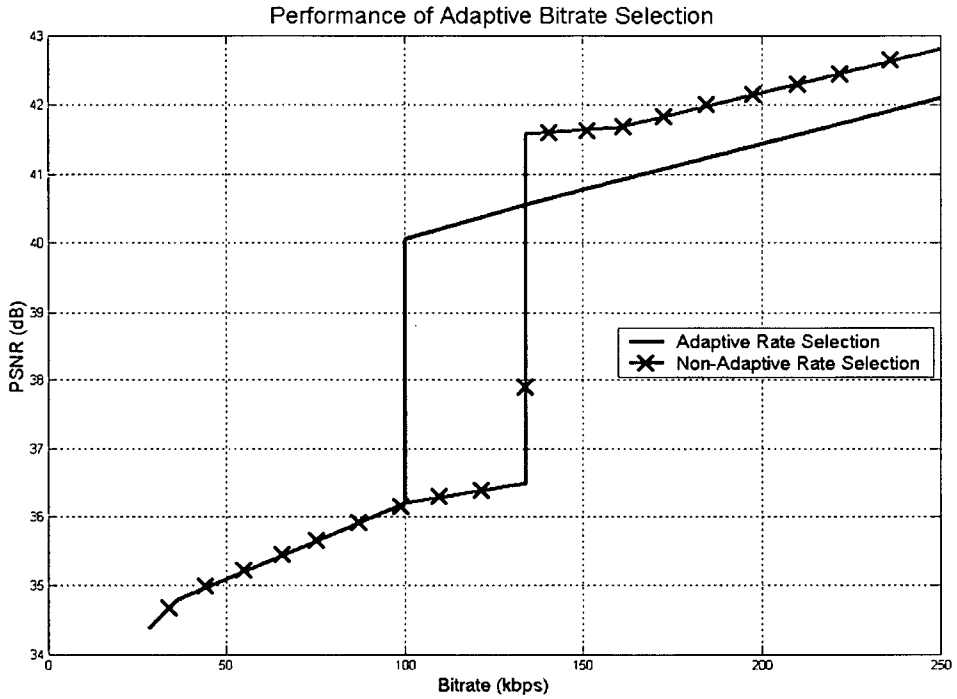


Figure 34 R-D Performance of the Adaptive Rate Selection Algorithm

3.4 Conclusion

In this chapter, we introduce a novel hybrid approach that combines FGS scalability with stream-switching based on the H.264's SP frame concept. We also introduce a novel R-D optimized adaptive rate selection algorithm for choosing the rates of the base layer streams. Combining FGS with SP frames is made possible by using our H.264 based FGS technology, presented in Chapter 2. For high bandwidth variations, our proposed system switches from a low-quality stream to a higher-quality stream, whereas low bandwidth variations are accommodated by using only the corresponding FGS enhancement layer. This way, the FGS enhancement layer mostly operates in the high efficiency regions of its R-D curve. In a network environment where bandwidth changes dynamically, our proposed hybrid method outperforms FGS by 2.9 dB and the stream-switching approach by 1.5 dB on average.

CHAPTER 4

4 Conclusions and Future Work

4.1 Conclusions

For networks used for video transmission environment, such as wireless networks and Internet, the available bandwidth for video transmission is not constant but varies over time. This variation in the available bandwidth possesses a problem for a video transmission system. Traditional video coding standards, whose objective is to optimize the quality of the video at a given bitrate, cannot cope with this bandwidth variation problem effectively. Scalable Video Coding techniques have been developed to more efficiently address this bandwidth variation problem.

Scalable Video Coding (SVC) is a video coding framework that enables a system to adapt the quality of the video sequence to the underlying channel's available bandwidth.

All popular video coding standards, such as MPEG-2 and MPEG-4 include some scalability tools. The latest video coding standard, H.264, provides superior compression efficiency over all previous standards, but it does not include tools for coding the video in a scalable fashion.

In this work, we developed a scalable video coding scheme based on the most advanced video coding standard, H.264. Up to now, H.264 standard offered limited scalability, but there were no solution that achieves highly flexible fine-granular-scalability using the H.264 standard. In order to achieve scalability with H.264, Fine Granular Scalability (FGS) that is originally developed for MPEG-4 is adapted to H.264. We modified the techniques present in FGS and developed novel techniques, so that the proposed scalable

coding system has low computational complexity, high error resiliency and has high coding efficiency.

At our proposed H.264 based FGS structure, the DCT is replaced by 4x4 Integer Transform at the enhancement layer. This brings an overhead for the *Coded Block Pattern* (CBP) coding at each macroblock header. The number of bits used to code the CBP is approximately quadrupled due to the change in macroblock structure. We proposed a Hierarchical CBP Coding scheme to decrease this overhead. On average the proposed scheme uses 70% less bits than FGS scheme for coding the CBP at the first and second bitplanes. For some cases the proposed scheme uses up to 75% less bits than the FGS scheme.

We adapt the entropy coding method of H.264 standard to the FGS structure. The method is based on Reversible Exp-Golomb coding and it is proved to be highly error resilient with low computational complexity. By replacing the entropy coding method, we achieve 7% gain in coding efficiency.

To overcome this problem, we also introduce a hybrid method that combines our proposed H.264 based FGS approach with the stream-switching approach employed in the H.264 standard. By combining different techniques, our proposed system offers a complete solution for all kinds of applications. The proposed system outperforms existing systems by offering optimum bandwidth utilization and improved video quality for the end user. We also introduce a novel R-D optimized adaptive rate selection algorithm for choosing the bitrates of the base layer streams. In a network environment where bandwidth changes dynamically, our proposed hybrid method outperforms FGS by 2.9 dB and the stream-switching approach by 1.5 dB on average. Combining FGS with SP

frames is made possible by using our H.264 based FGS technology, presented in Chapter 2.

4.2 Future Work

The proposed H.264 based FGS is designed to introduce minimal computation complexity to the overall system. However, in the scope of this work, no formal testing and analysis was performed to analyze the exact amount of complexity gain achieved. Also, by introducing Reversible Exp-Golomb codewords to the FGS enhancement layer, the error resiliency of the system is increased. This feature should be further tested and analyzed for real-world application scenarios, such as 3G wireless environments or the Internet.

H.264 video coding standard offers several methods for entropy coding. In this thesis, we used Reversible Exp-Golomb coding method due to its low computational complexity and high error resiliency. However, other entropy coding methods such as Context Adaptive Variable Length Coding (CAVLC), and Context Adaptive Binary Arithmetic Coding (CABAC) can also be incorporated in our system.

The combined approach introduced in Chapter 3, gives very good results and it can be further optimized for certain bitrates. The developed approach is good for pre-recorded video, where the entire video stream is available at the time of streaming. This approach can be further developed for real-time streaming applications.

The latest trends in scalable video coding are mostly based on wavelet coding tools and motion compensated temporal filtering (MCTF) based video codecs. Future work could include incorporation of these tools to existing video coding standards.

APPENDIX

A. Exp-Golomb Codes for Entropy Coding

The following table presents the Exp-Golomb codewords and their corresponding numbers used for entropy coding. For the rest of the tables in the appendixx, only the Exp-Golomb code number is indicated where a codeword is specified.

Exp-Golomb Code Number	Codeword	Number of bits
0	1	1
1	010	3
2	011	3
3	00100	5
4	00101	5
5	00110	5
6	00111	5
7	0001000	7
8	0001001	7
9	0001010	7
10	0001011	7
11	0001100	7
12	0001101	7
13	0001110	7
14	0001111	7
15	000010000	9
16	000010001	9
17	000010010	9
18	000010011	9
19	000010100	9
20	000010101	9
21	000010110	9
22	000010111	9
23	000011000	9
24	000011001	9
25	000011010	9
26	000011011	9
27	000011100	9
28	000011101	9
29	000011110	9
30	000011111	9
31	00000100000	11

32	00000100001	11
33	00000100010	11
34	00000100011	11
35	00000100100	11
36	00000100101	11
37	00000100110	11
38	00000100111	11
39	00000101000	11
40	00000101001	11
41	00000101010	11
42	00000101011	11
43	00000101100	11
44	00000101101	11
45	00000101110	11
46	00000101111	11
47	00000110000	11
48	00000110001	11
49	00000110010	11
50	00000110011	11
51	00000110100	11
52	00000110101	11
53	00000110110	11
54	00000110111	11
55	00000111000	11
56	00000111001	11
57	00000111010	11
58	00000111011	11
59	00000111100	11
60	00000111101	11
61	00000111110	11
62	00000111111	11

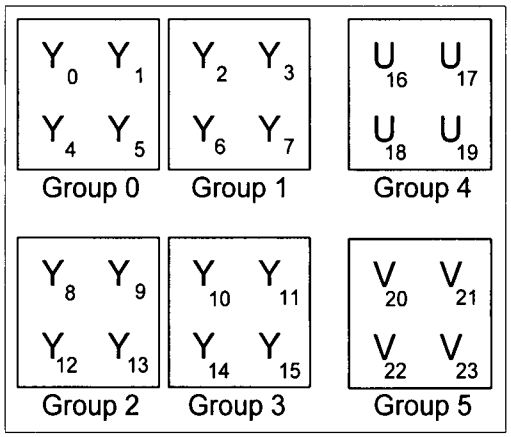
Table 13 Exp-Golomb Codes for Entropy Coding

B. VLC Codes for CBP_CODE

B.1. CBP Codes for the First Bitplane

Category 0

All the four luminance and two color components are present in the bitplane.



CBP (*uv,yyyy*)	Exp-Golomb Code Number	Number of Bits
11,1111	0	1
11,0000	1	3
11,0001	2	3
11,0010	3	5
11,0011	4	5
11,0100	5	5
11,0101	6	5
11,0110	7	7
11,0111	8	7
11,1000	9	7
11,1001	10	7
11,1010	11	7
11,1011	12	7
11,1100	13	7
11,1101	14	7
11,1110	15	9
00,0000	16	9
00,0001	17	9
00,0010	18	9
00,0011	19	9

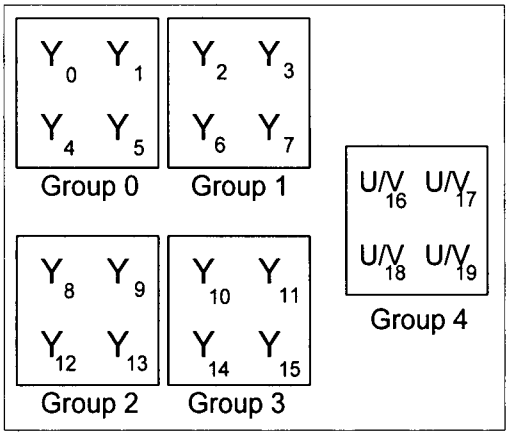
00,0101	21	9
00,0110	22	9
00,0111	23	9
01,0000	24	9
01,0001	25	9
01,0100	26	9
01,0101	27	9
10,0000	28	9
10,0001	29	9
10,0010	30	9
00,1000	31	11
00,1001	32	11
00,1010	33	11
00,1011	34	11
00,1100	35	11
00,1101	36	11
00,1110	37	11
00,1111	38	11
01,0010	39	11
01,0011	40	11
01,0110	41	11
01,0111	42	11
01,1000	43	11
01,1001	44	11
01,1010	45	11
01,1011	46	11
01,1100	47	11
01,1101	48	11
01,1110	49	11

10,0011	51	11
10,0100	52	11
10,0101	53	11
10,0110	54	11
10,0111	55	11
10,1000	56	11
10,1001	57	11
10,1010	58	11
10,1011	59	11
10,1100	60	11
10,1101	61	11
10,1110	62	11
10,1111	63	13

Table 14 CBP Codes for the First Bitplane, Category 0

Category 1-2

All the four luminance components and only one color component (U or V) are present in the bitplane



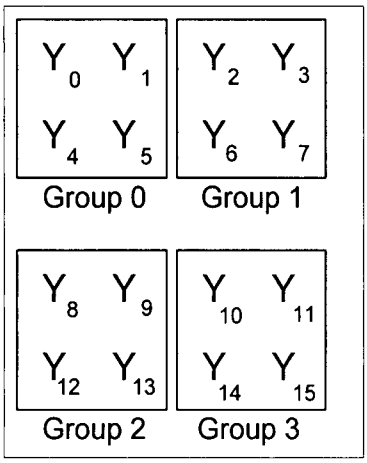
CBP (*u/v,yyyy*)	**Exp-Golomb Code Number**	**Number of Bits**
1,1111	0	1
1,0111	1	3
1,0011	2	3
1,1011	3	5
1,1101	4	5
1,1110	5	5
1,0001	6	5
1,0010	7	7
1,0100	8	7
1,0101	9	7
1,0110	10	7
1,1001	11	7
1,1010	12	7
1,1100	13	7
0,0001	14	7
0,0000	15	9
0,0010	16	9
0,0011	17	9
0,0100	18	9
0,0101	19	9
0,0110	20	9
0,0111	21	9
0,1001	22	9

0,1010	23	9
0,1011	24	9
0,1100	25	9
0,1101	26	9
0,1110	27	9
0,1111	28	9
1,0000	29	9
1,1000	30	9
0,1000	31	11

Table 15 CBP Codes for the First Bitplane, Category 1-2

Category 3

All the four luminance components are present without any color component in the bitplane



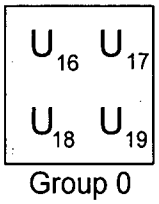
CBP (*yyyy*)	**Exp-Golomb Code Number**	**Number of Bits**
1111	0	1
0111	1	3
0011	2	3
1011	3	5
1101	4	5
1110	5	5
0001	6	5
0010	7	7
0100	8	7
0101	9	7
0110	10	7
1001	11	7

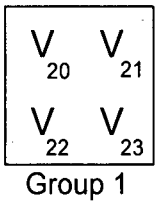
1010	12	7
1100	13	7
0000	14	7
1000	15	9

Table 16 CBP Codes for the First Bitplane, Category 3

Category 4

Only two color components are present in the bitplane without any luminance components.



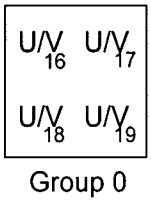


CBP (*uv*)	Fixed Length Code
00	00
01	01
10	10
11	11

Table 17 CBP Codes for the First Bitplane, Category 4

Category 5-6

Only one of the color components (either U or V) is present in the bitplane without any luminance components



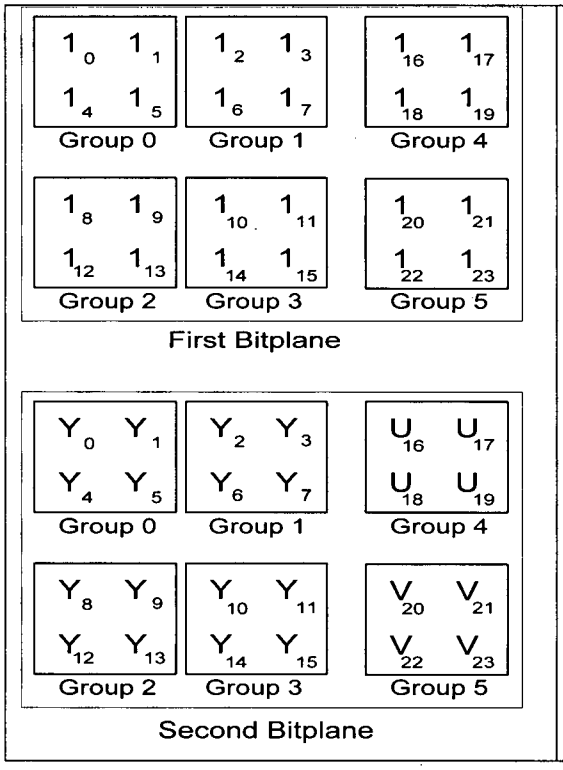
CBP (*u/v*)	Fixed Length Code
0	0
1	1

Table 18 CBP Codes for the First Bitplane, Category 5-6

B.2. CBP Codes for the Second Bitplane

Category 0

CBP_CODE is used at the second bitplane and second bitplane contains all the luminance and the chrominance components.

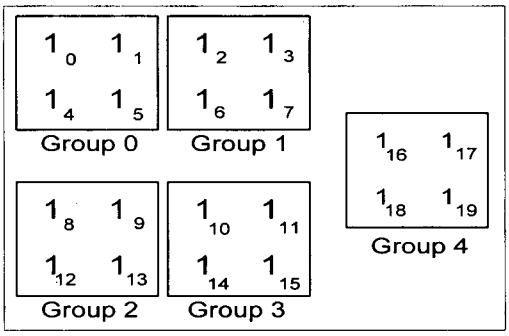


Same table as Table 14

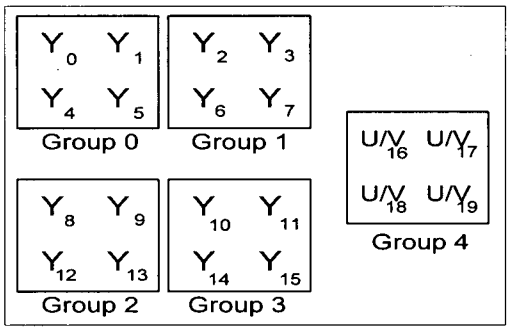
Table 19 CBP Codes for the Second Bitplane, Category 0

Category 1-2

CBP_CODE is used at the second bitplane and second bitplane contains all the four luminance components and only one color component (U or V).



First Bitplane



Second Bitplane

CBP (*u/v,yyyy*)	Exp-Golomb Code Number	Number of Bits
0,1111	0	1
0,0111	1	3
0,0011	2	3
0,1011	3	5
0,1101	4	5
0,1110	5	5
0,0001	6	5
0,0100	7	7
0,0101	8	7
0,0110	9	7
0,1001	10	7
0,1010	11	7
0,1100	12	7
1,0001	13	7
0,0010	14	7
1,0011	15	9
1,0100	16	9
1,0101	17	9
1,0110	18	9
1,0111	19	9
1,0010	20	9
1,1111	21	9
1,1001	22	9
1,1010	23	9
1,1011	24	9
1,1100	25	9
1,1101	26	9
1,1110	27	9

1,0000	28	9
0,0000	29	9
0,1000	30	9
1,1000	31	11

Table 20 CBP Codes for the Second Bitplane, Category 1-2

Category 3

CBP_CODE is used at the second bitplane and second bitplane contains all the four luminance components but no color component.

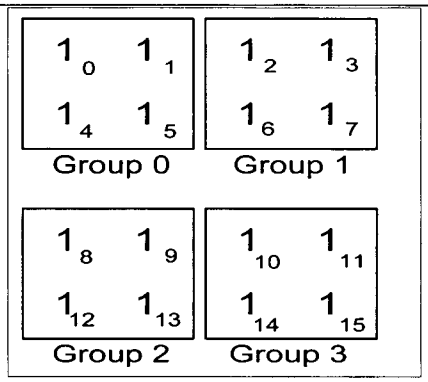
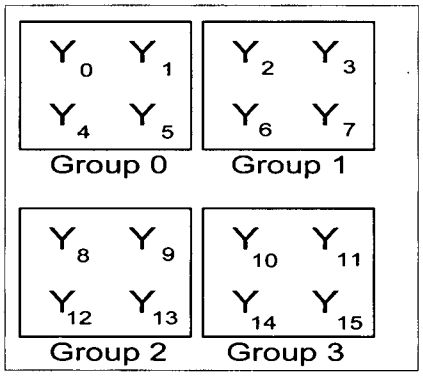
 	Same as Table 17

Table 21 CBP Codes for the Second Bitplane, Category 3

C. VLC Codes for SUB_CBP_CODE

Following table illustrates the SUB_CBP_CODE VLC codes used in *block_cbp* procedure of CBP coding.

(Block_0, Block_1, Block_2, Block_3)	**SUB_CBP_CODE**
1111	1
0000	00000
0001	00001
0010	00010
0011	00011
0100	00100

0101	00101
0110	00110
0111	00111
1000	01000
1001	01001
1010	01010
1011	01011
1100	01100
1101	01101
1110	01110

Table 22 VLC Codes for SUB_CBP_CODE

D. RUN – EOP Statistics

D.1. BUS Sequence, Base Layer at 500 Kbps

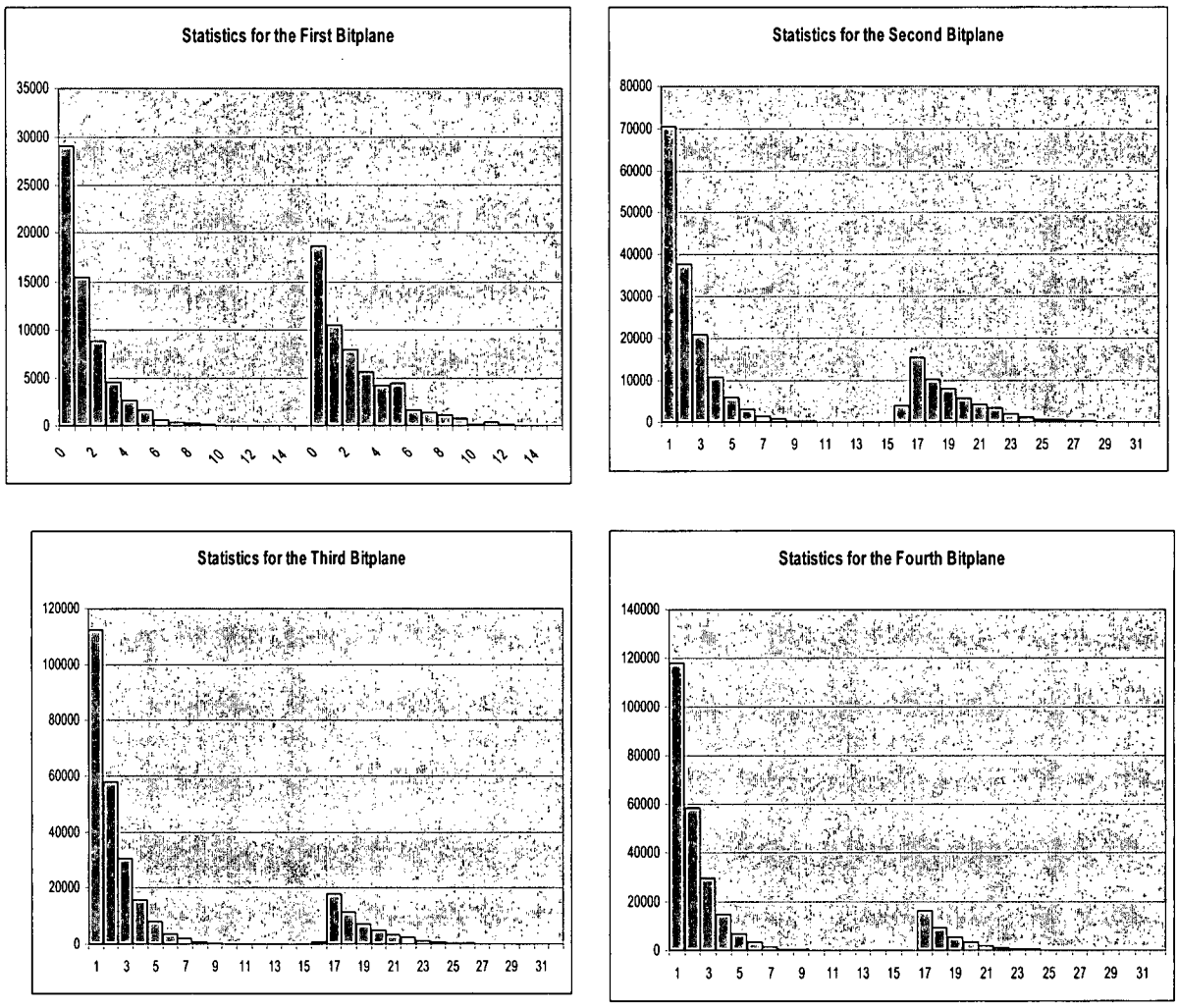
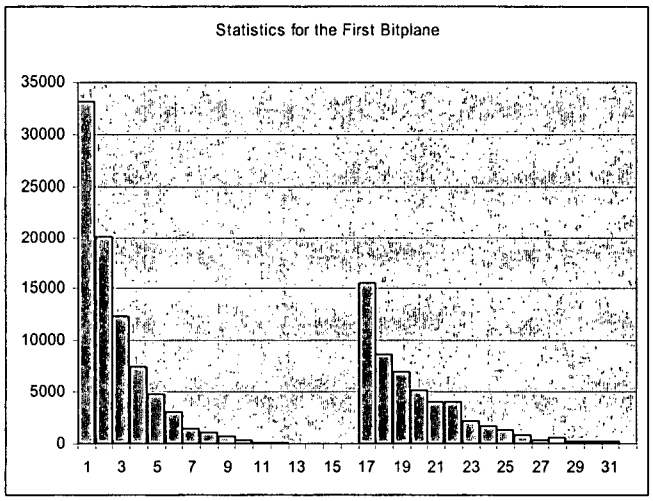
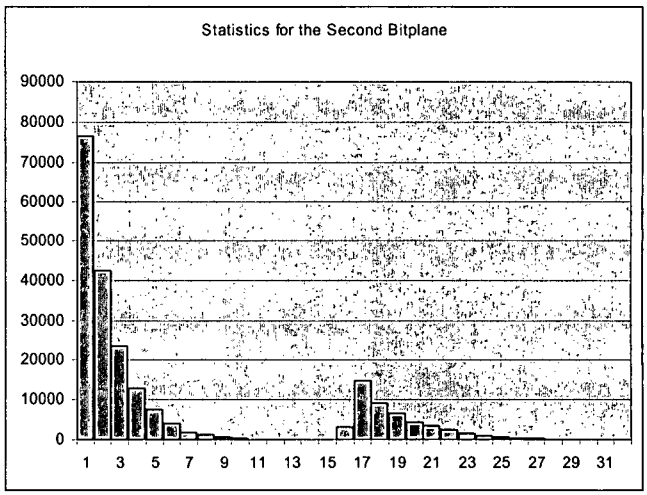
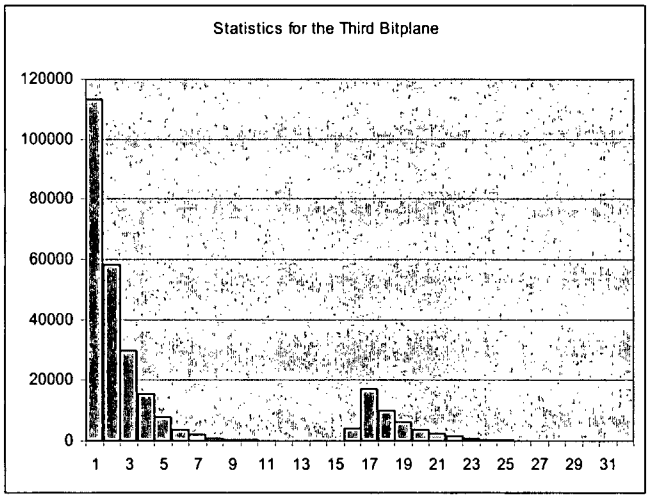


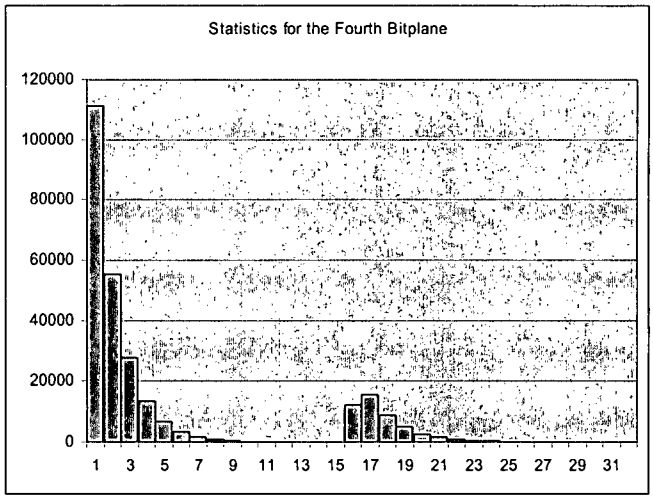
Figure 35 (RUN,EOP) Statistics for the BUS Sequence at 500 Kbps

D.2. BUS Sequence, Base Layer at 1.5 Mbps

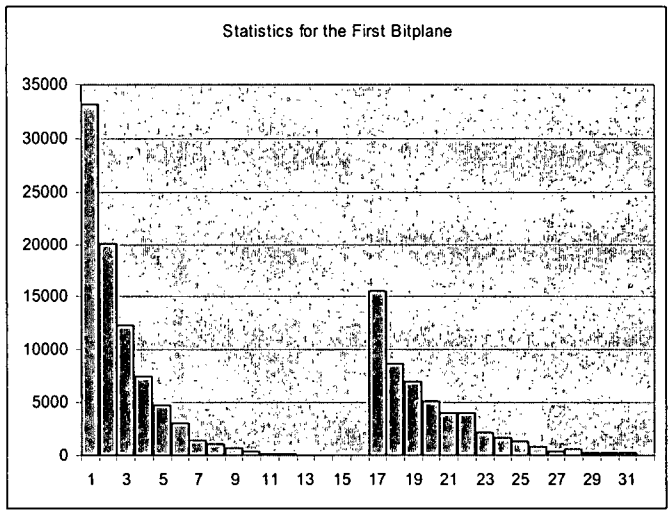


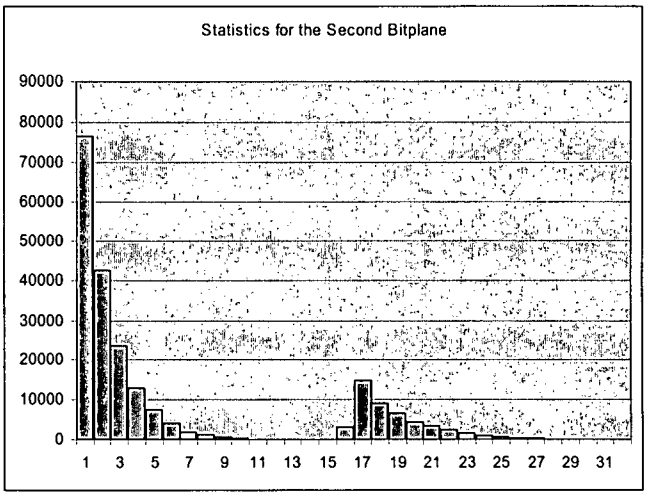


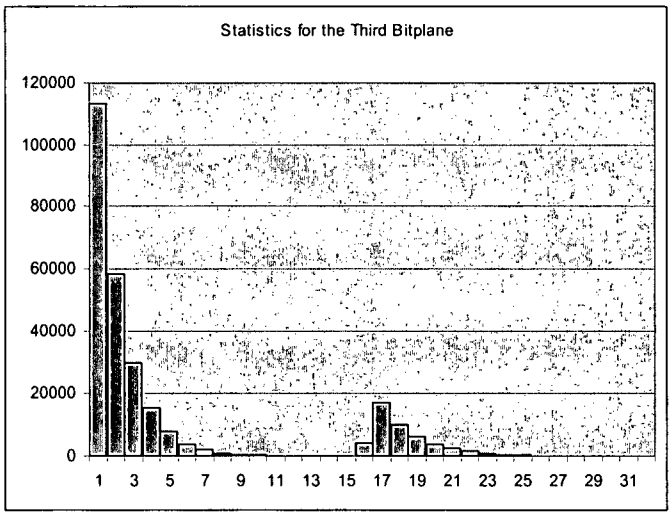


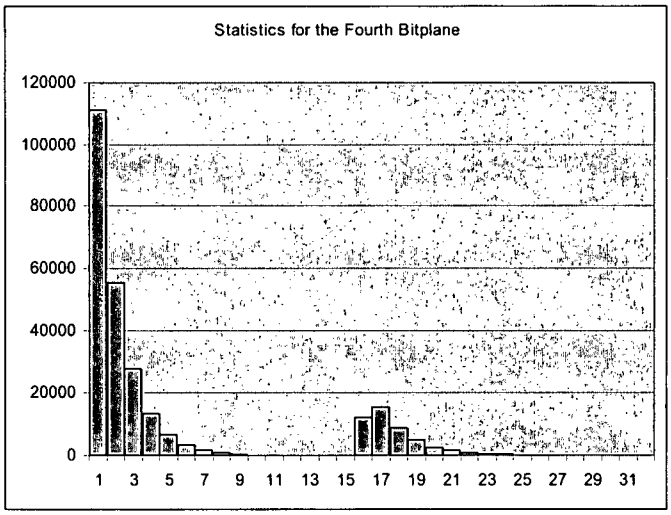


D.3. MOBILE Sequence, Base Layer at 500Kbps

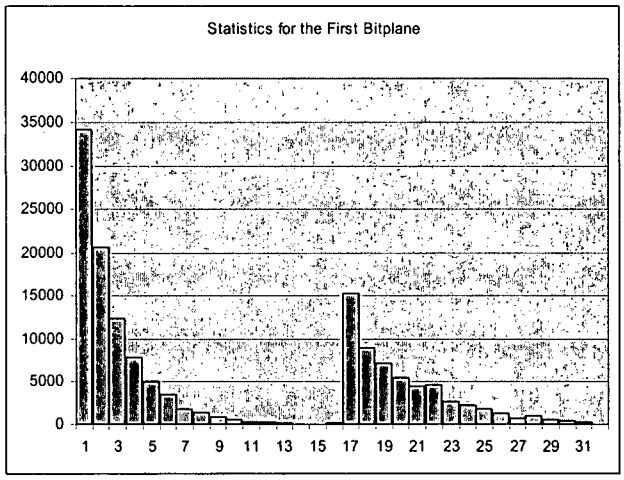


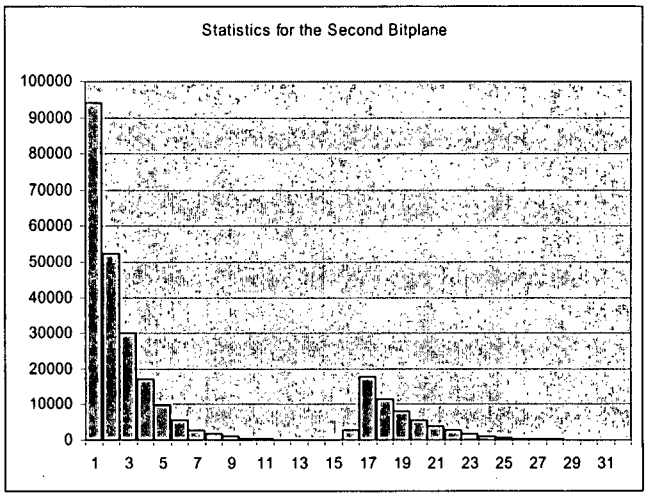


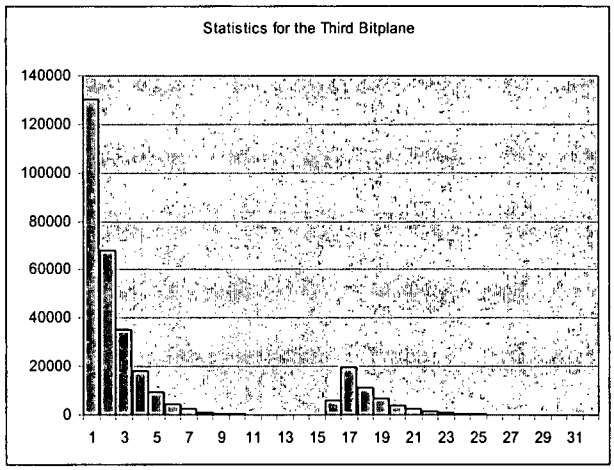


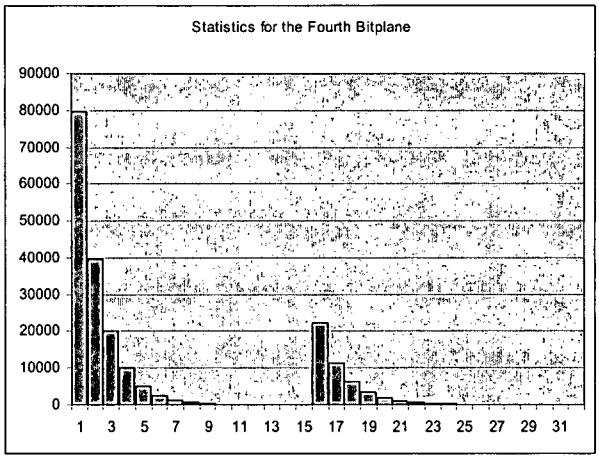


D.4. MOBILE Sequence, Base Layer at 1.5 Mbps

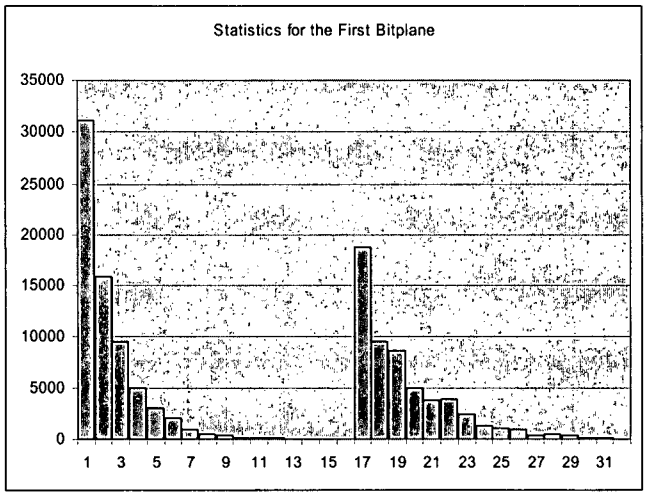


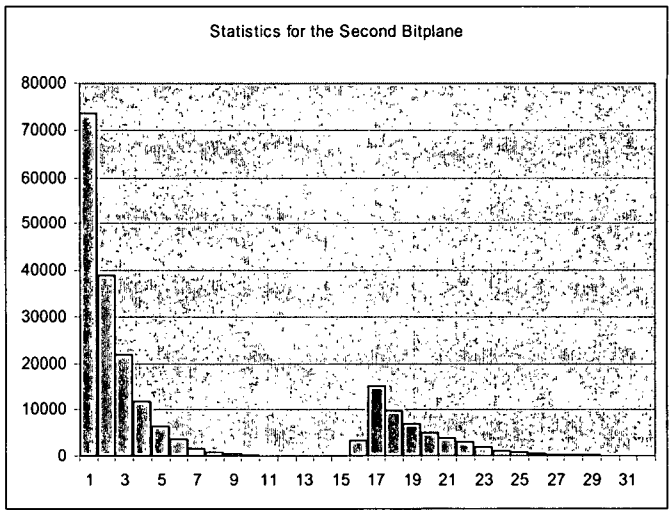


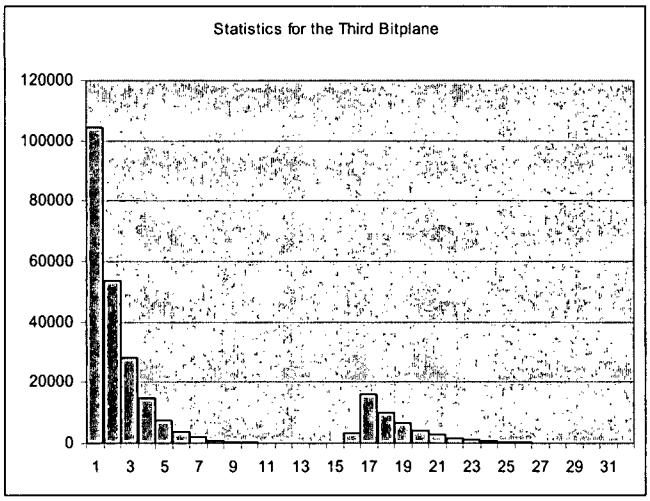


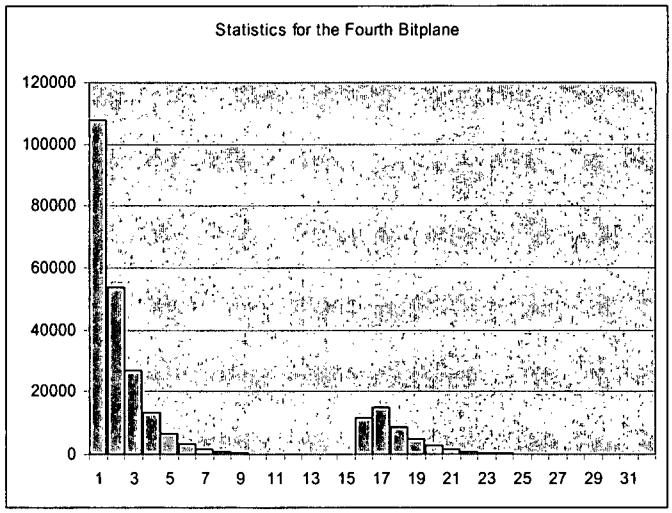


D.5. TEMPETE Sequence, Base Layer at 500 Kbps

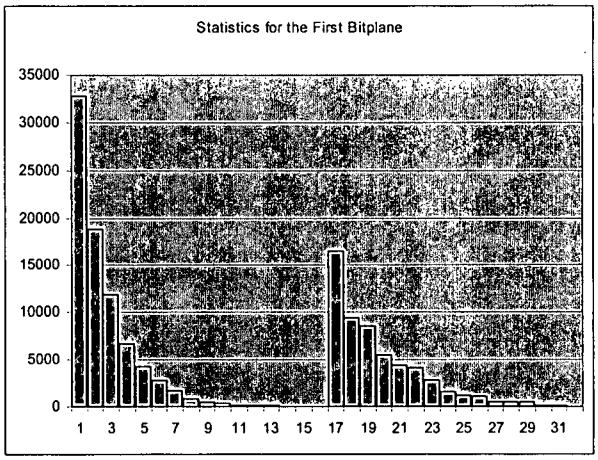


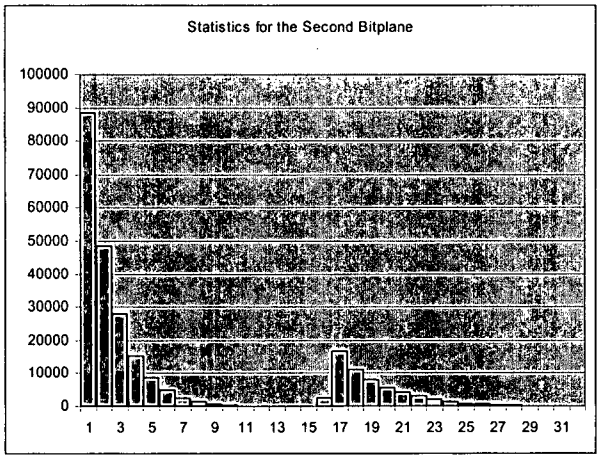


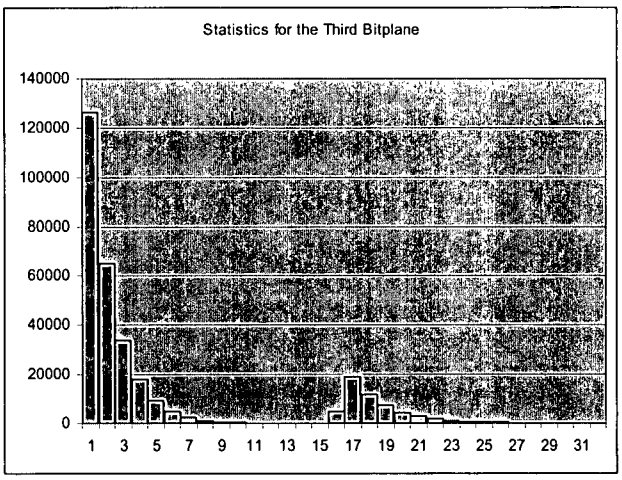


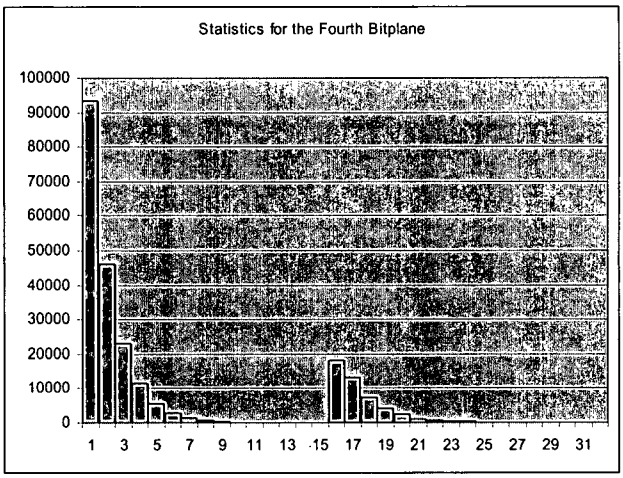


D.6. TEMPETE Sequence, Base Layer at 1.5 Mbps









E. Proposed VLC Tables

E.1. (RUN,EOP) Symbols for the First Bitplane

Index	(RUN,EOP)	Code
0	(0,0)	1
1	(0,1)	010
2	(1,0)	011
3	(1,1)	00100
4	(2,1)	00101
5	(2,0)	00110
6	(3,1)	00111
7	(3 ,0)	0001000
8	(4,1)	0001001
9	(5,1)	0001010
10	(4,0)	0001011
11	(6,1)	0001100
12	(5,0)	0001101
13	(7,1)	0001110
14	(8,1)	0001111
15	(6,0)	000010000
16	(9,1)	000010001
17	(7,0)	000010010
18	(10,1)	000010011
19	(8,0)	000010100
20	(11,1)	000010101
21	(9,0)	000010110
22	(12,1)	000010111
23	(14,1)	000011000
24	(10,0)	000011001
25	(13,1)	000011010
26	(11,0)	000011011
27	(15,1)	000011100
28	(13,0)	000011101
29	(12,0)	000011110
30	(14,0)	000011111

E.3. (RUN,EOP) Symbols for the Second Bitplane

Index	(RUN,EOP)	Code
0	(0,0)	1
1	(1,0)	010
2	(2,0)	011
3	(0,1)	00100
4	(3,0)	00101
5	(1,1)	00110
6	ALL-ZERO	00111
7	(2,1)	0001000
8	(4,0)	0001001
9	(3,1)	0001010
10	(4,1)	0001011
11	(5,1)	0001100
12	(5,0)	0001101
13	(6,1)	0001110
14	(6,0)	0001111
15	(7,1)	000010000
16	(7,0)	000010001
17	(8,1)	000010010
18	(9,1)	000010011
19	(8,0)	000010100
20	(10,1)	000010101
21	(11,1)	000010110
22	(10,0)	000010111
23	(9,0)	000011000
24	(12,1)	000011001
25	(11,0)	000011010
26	(14,1)	000011011
27	(13,1)	000011100
28	(15,1)	000011101
29	(12,0)	000011110
30	(13,0)	000011111
31	(14,0)	00000100000

E.4. (RUN,EOP) Symbols for the Other Bitplanes

Index	(RUN,EOP)	Code
0	(0,0)	1
1	(1,0)	010
2	(2,0)	011
3	(0,1)	00100
4	(3,0)	00101
5	(1,1)	00110
6	(4,0)	00111
7	(2,1)	0001000
8	(5,0)	0001001
9	(3,1)	0001010
10	(4,1)	0001011
11	(6,0)	0001100
12	(5,1)	0001101
13	(7,0)	0001110
14	(6,1)	0001111
15	(8,0)	000010000
16	(7,1)	000010001
17	(8,1)	000010010
18	ALL-ZERO	000010011
19	(9,0)	000010100
20	(9,1)	000010101
21	(10,0)	000010110
22	(10,1)	000010111
23	(11,1)	000011000
24	(11,0)	000011001
25	(12,1)	000011010
26	(12,0)	000011011
27	(13,1)	000011100
28	(14,1)	000011101
29	(13,0)	000011110
30	(14,0)	000011111
31	(15,1)	00000100000

Bibliography

1. H. Radha, M. van der Schaar, and Y. Chen, "The MPEG-4 Fine-Grained Scalable Video Coding Method for Multimedia Streaming Over IP," IEEE Transactions on Multimedia, vol. 3, no. 1, pp. 53– 68, Mar. 2001.

2. Information Technology: Generic Coding of Moving Video and Associated Audio Information, ISO/IEC CD 13818 MPEG 2 International Standard, pt. 1–3, 1992.

3. ISO/IEC JTC1, "Generic Coding of Audiovisual Objects – Part 2: Visual (MPEG-4 Visual)", ISO/IEC 14496-2, Version 1: Jan. 1999, Version 2: Jan. 2000; Version 3: Jan. 2001.

4. ITU-T Recommendation H.263, "Video Coding for Low Bit-Rate Communication", Version 1: Nov. 1995, Version 2: Jan.1998, Version 3: Nov. 2000.

5. ISO/IEC 15444-1: Information technology—JPEG 2000 image coding system—Part 1: Core coding System, 2000.

6. ISO/IEC IS 10918-1 | ITU-T Recommendation T.81 - JPEG Image Coding Standard-Part 1

7. K. Sayood, *Introduction to Data Compression.* Morgan Kaufmann Publishers, Inc., 1996.

8. B. Girod, "Why B-pictures work: a theory of multi-hypothesis motion-compensated prediction," Proc. IEEE International Conference on Image Processing (ICIP), vol. II, pp. 213-217, Chicago, October 1998.

9. H.G. Mussman, P. Pirsch and H.J. Grallert "Advances in picture coding", *Proc. IEEE,* vol. 73, no.4, pp. 523-548, April 1985

10. K.R. Rao, J.J. Hwang, *Techniques & Standards for Image & Audio Coding*, Upper Saddle River, NJ: Prentice-Hall, 1996.

11. Sullivan, G.J., Baker, R.L., "Rate-distortion optimization for tree-structured source coding with multi-way node decisions" Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., vol. 3, pp. 393 – 396, March 1992

12. Y. He, F. Wu, S. Li, Y. Zhong, and S. Yang, "H.26L-based Fine Granularity Scalable Video Coding," in Proc. of IEEE International Symposium on Circuits and Systems (ISCAS), Scottsdale, Arizona, USA, vol. 4, pp. 548-551, May 2002

13. Malvar, H.S., Hallapuro, A., Karczewicz, M., Kerofsky, L., "Low-complexity transform and quantization in H.264/AVC", IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp.598 - 603, July 2003.

14. R. Talluri, "Error-Resilient Video Coding in the MPEG-4 Standard," IEEE Commun. Mag., vol. 36, no. 6, pp. 112-119, June 1998.

15. Wiegand, T., Sullivan, G.J., Bjntegaard, G., Luthra, A., "Overview of the H.264/AVC video coding standard", IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560- 576, July 2003.

16. List, P., Joch A., Lainema, J., Bjntegaard, G., Karczewicz, M., "Adaptive deblocking filter", IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 614- 619, July 2003.

17. Karczewicz, M., Kurceren, R., "The SP- and SI-frames design for H.264/AVC", IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 637- 644, July 2003.

18. Wiegand, T., Schwarz, H., Joch, A., Kossentini, F., Sullivan, G.J., "Rate-constrained coder control and comparison of video coding standards", IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 688- 703, July 2003.

19. Xiaoyan Sun, Feng Wu, Shipeng Li, Wen Gao, Ya-Qin Zhang, "Seamless switching of scalable video bitstreams for efficient streaming", IEEE International Symposium on Circuits and Systems, vol. 3, pp. 385-388, May 2002

20. Yuwen He, Feng Wu, Shipeng Li, Yuzhuo Zhong, Shiqiang Yang, " H.26L-based fine granularity scalable video coding", IEEE International Symposium on Circuits and Systems, vol. 4, pp. 548-551, May 2002

21. W. Li, F. Ling, and H. Sun, "Bitplane coding of DCT coefficients,", ISO/IEC JTC1/SC29/WG11, MPEG97/M2691, Oct. 22, 1997.

22. Ling, W. Li, and H. Sun, "Bitplane coding of DCT coefficients for image and video compression," in Proc. SPIE Visual Communications and Image Processing (VCIP), San Jose, CA, Jan. 25–27, 1999.

23. L. Kerofsky, M. Zhou, "Reduced Complexity VLC" in Joint Video Team of ISO/IEC JTC1/SC29/WG11 & ITU-T SG16/Q.6 Doc. JVTB029, Geneva, Switzerland, Feb. 2002.

24. "Draft ITU-T Recommendation H.264 and Draft ISO/IEC 14 496-10 AVC," in Joint Video Team of ISO/IEC JTC1/SC29/WG11 & ITU-T SG16/Q.6 Doc. JVT-G050, T. Wieg, Ed., Pattaya, Thailand, Mar. 2003.

25. ISO/IEC JTC1/SC29/WG11, "Call for Proposals on Scalable Video Coding Technology", MPEG2003/N6193, Waikoloa, December 2003.