

MULTIPLEXING AND DEMULTIPLEXING HEVC VIDEO AND AAC AUDIO AND  
ACHIEVING LIP SYNCHRONIZATION DURING PLAYBACK

by

MRUDULA BALMOHAN WARRIER

Presented to the Faculty of the Graduate School of  
The University of Texas at Arlington in Partial Fulfillment  
of the Requirements  
for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2014

Copyright © by Mrudula Balmohan Warriar 2014

All Rights Reserved



## Acknowledgements

I'm very grateful to my thesis advisor, Dr.K.R. Rao for introducing me to specialize in the field of video coding and for his immense support, guidance and faith in every step of the thesis. He has been a great source of inspiration to learn, not only for the thesis, but also for many other things.

I would like to extend my appreciation to my Multimedia processing lab mates who provided me with their valuable inputs throughout.

I would like to extend my gratitude to UTA for providing me with opportunity and resources to excel in my graduate studies.

Finally, I would like to thank my parents, sister and friends for their continuous support, without which this would have not been possible.

November 10, 2014

## Abstract

# MULTIPLEXING AND DEMULTIPLEXING HEVC VIDEO AND AAC AUDIO AND ACHIEVING LIP SYNCHRONIZATION DURING PLAYBACK

Mrudula Balmohan Warriar, M.S

The University of Texas at Arlington, 2014

Supervising Professor: K.R. Rao

High efficiency video coding (HEVC) /H.265 [5], is the latest digital video coding standard which has proven to be superior to earlier standards in terms of compression ratio, quality and error resilience. In order for the end user to understand the video meaningfully, there should be an associated audio with it. Any video is incomplete without a proper audio. Advanced audio coding (AAC) [8] is the digital audio codec standard defined in MPEG-2 and later in MPEG-4 with few changes. The audio quality of an AAC stream is widely used as the audio coding standard in various applications. It would be a great advantage to the user to adopt HEVC as video codec and AAC as the audio coding, for transmission of digital multimedia through air (ATSC, DVB) or through the internet (video streaming, IPTV). However, multiplexing is required for these applications in order to combine and create a single bit stream from separate audio and video bit streams for transmission purposes. The objective of the thesis is to propose a method for effectively multiplexing the audio and video coded streams for transmission followed by demultiplexing the streams at the receiver and achieving lip synchronization between the audio and video during the playback. The proposed method uses the fact that frames are constant throughout the length of audio and video. The first step of the process is the

packetization of elementary audio and video bit streams. The frame number information is stored in the header of the packets which is used as the vital information to synchronize the video and audio during playback. Then second layer of packetization is carried out from the first layer in order to meet the requirements of MPEG-2 transport stream. Proposed method uses playback time as the criteria for allocating data packets during multiplexing in order to prevent buffer overflow or underflow at the demultiplexer. The information required during the demultiplexer process to ensure error free is put in the header. Flow and results of the thesis are discussed in detail in the chapters.

## Table of Contents

Acknowledgements .....	iii
Abstract .....	iv
List of Illustrations .....	ix
List of Tables .....	xi
Chapter 1 INTRODUCTION.....	1
1.1 Introduction .....	1
1.2 Thesis outline.....	2
Chapter 2 OVERVIEW OF HIGH EFFICIENCY VIDEO CODING (HEVC) .....	4
2.1 Introduction .....	4
2.2 HEVC Encoder .....	5
2.2.1 Working of HEVC Encoder.....	5
2.2.2 Coding tree units .....	6
2.2.3 Intra Prediction .....	9
2.3 Bitstream syntax of HEVC .....	10
2.4 Video coding layer topics.....	12
2.4.1 Motion vector signaling.....	12
2.4.2 Motion compensation .....	13
2.4.3 Quantization control .....	13
2.4.4 Entropy Coding.....	14
2.4.5 Deblocking Filter.....	14
2.4.6 Sample Adaptive Offset (SAO).....	14
2.5 Parallel decoding syntax and modified slice structuring.....	15
2.5.1 Tiles .....	15
2.5.2 Wavefront parallel processing .....	15

2.5.3 Dependent slice segments .....	15
2.5.4 Slices .....	16
2.6 HEVC Profile, level .....	16
2.7 Summary .....	16
Chapter 3 OVERVIEW OF ADVANCED AUDIO CODING (AAC) .....	17
3.1 Introduction .....	17
3.2 AAC Profiles .....	17
3.3 AAC Encoder .....	18
3.4 Summary .....	24
Chapter 4 MULTIPLEXING .....	25
4.1 Introduction .....	25
4.2 MPEG Bitstream Structure .....	25
4.2.1 Elementary Stream .....	26
4.2.2 Packetized Elementary Stream (PES) .....	26
4.3 MPEG-2 Multiplexing .....	28
4.4 MPEG-2 Transport Stream .....	29
4.5 Format of Transport Stream packet .....	30
4.6 Frame number as Time Stamp .....	34
4.7 Proposed Multiplexing method .....	35
4.8 Summary .....	37
Chapter 5 DEMULTIPLEXING AND SYNCHRONIZATION .....	38
5.1 Demultiplexing .....	38
5.2 Synchronization and playback .....	41
5.3 Summary .....	42
Chapter 6 RESULTS AND CONCLUSIONS .....	43

6.1 Test conditions.....	43
6.2 Results.....	45
6.3 Conclusion .....	49
6.3 Future research .....	49
APPENDIX A Platform .....	51
APPENDIX B List Of Acronyms .....	53
References.....	57
Biographical Information .....	63



## List of Illustrations

Figure 1-1	Multiplexing of audio and video streams.....	13
Figure 1-2	Demultiplexing of audio and video streams.....	13
Figure 2-1	Block diagram of HEVC encoder.....	16
Figure 2-2	Decoder diagram of HEVC.....	17
Figure 2-3	Format for YUV components.....	18
Figure 2-4	Quadtree structure.....	19
Figure 2-5	Modes for splitting a CB into PBs.....	20
Figure 2-6	Intra prediction modes of HEVC encoder.....	22
Figure 2-7	Comparison of HEVC and H.264 NAL units.....	23
Figure 2-8	Integer and fractional sample positions for luma interpolation.....	25
Figure 2-9	Three coefficient scanning methods in HEVC.....	27
Figure 3-1	AAC encoder block diagram.....	30
Figure 4-1	Streams supported by MPEG-2 .....	37
Figure 4-2	PES encapsulation from elementary stream .....	38
Figure 4-3	PES packet header .....	39
Figure 4-4	Single Program Transport Stream .....	40
Figure 4-5	transport stream (TS) header .....	41
Figure 4-6	Multiplexing of audio data and video data into MPEG-2 transport stream.....	42
Figure 4-7	Combining ES from encoders into TS (red) or a PS (yellow) .....	44
Figure 4-8	Flowchart of multiplexing process.....	49
Figure 5-1	overview of multiplexing and demultiplexing.....	50
Figure 5-2	Flowchart of the demultiplexer.....	51
Figure 6-1	Test Condition for Video file.....	56

Figure 6-2	HEVC encoder with IBBB setting.....	59
Figure 6-3	Morning Test sequence, WVGA 832x480, 25 FPS, 219 Frames.....	59
Figure 6-4	Extracting .wave and .yuv from .avi.....	60
Figure 6-5	Encoding .wav to .aac low complexity using FAAC .....	60

## List of Tables

Table 2-1	The NAL unit types and their associated meanings, classes in the HEVC standard.....	23
Table 3-1	ADTS header format.....	33
Table 3-2	ADTS profile bits in header .....	34
Table 4-1	PES packet header description .....	38
Table 4-2	Transport stream header glossary .....	47
Table 6-1	Test clip conditions.....	55
Table 6-2	Lip synchronization results for clip 1.....	57
Table 6-3	Lip synchronization results for clip 2.....	58

## Chapter 1

### INTRODUCTION

#### 1.1 Introduction

Digital television transmission has already replaced analog television transmission with better quality and less bandwidth. With the advent of HDTV, transmission schemes are aiming at transmitting superior quality video with provision to view both standard format and wide screen (16:9) format along with one or more audio streams per channel [17]. Choosing the right video codec and audio codec plays a very important role in achieving the bandwidth and quality requirements. H.265 or High Efficiency Video Coding (HEVC) [1], is the latest video coding standard by the ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC Moving Picture Experts Group (MPEG) as the product of a collective partnership effort known as the joint video team- video coding (JVT-VC) [10]. The new standard achieved about 50% bit rate savings as compared to H.264 [2]. In other words, this codec provides high quality video at the same bandwidth or same quality video in less bandwidth. HEVC provides the tools necessary to deal with packet losses in packet networks and bit errors in error-prone wireless networks. These features make this coding standard the right candidate for transmission. Advanced audio coding (AAC) [4] is a standardized lossy compression scheme for audio. The compression scheme was specified both as Part 7 of the MPEG-2 standard [6], and Part 3 of the MPEG-4 standard [5]. The video and audio streams based on these standards need to be multiplexed in order to construct a single stream, which is a requirement for transmission. Figure 1.1 [5] shows the multiplexing process which mainly focuses on splitting the individual streams into small packets, embedding information to easily realign the packets and achieving lip sync between the individual streams, providing provision to detect and correct bit errors and packet losses. In this

thesis, the process of encoding the raw streams, multiplexing the compressed streams followed by demultiplexing and synchronizing the individual streams during playback shown in figure 1.2 [17], is implemented in detail.

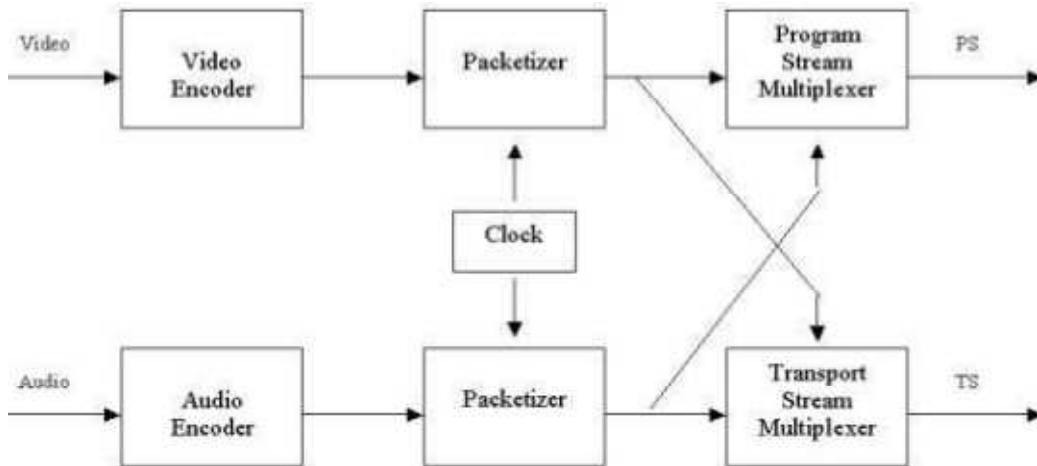


Figure 1-1 Multiplexing of audio and video streams. [17]

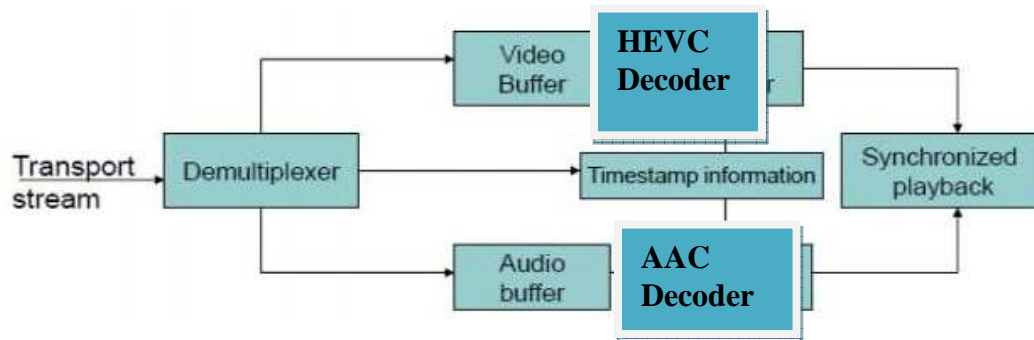


Figure 1-2 Demultiplexing of audio and video streams [17].

## 1.2 Thesis outline

Chapters 2 and chapter 3 give an overview of the H.265 [13] video standard and AAC audio standard [20] respectively. The bit stream formats along with the reason for choosing the standard are discussed in detail.

Chapter 4 explains the entire process of multiplexing the elementary streams and preparing the data packets for transmission. The additional information to be sent in the packet headers to assist the demultiplexing process, is also presented in this chapter.

In Chapter 5 demultiplexing of data packets and synchronization of reconstructed elementary streams are described. The adopted method of synchronization is compared with other methods of synchronization to analyze the advantages and disadvantages.

Chapter 6 outlines the test conditions, results and conclusions obtained using the proposed method of implementation.

## Chapter 2

### OVERVIEW OF HIGH EFFICIENCY VIDEO CODING (HEVC)

#### 2.1 Introduction

The High Efficiency Video Coding (HEVC) is the latest video project by ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) standardization organizations, working together in a partnership known as the Joint Collaborative Team on Video Coding (JCT-VC) [1]. An increasing diversity of services like high definition (HD) TV signals over satellite, cable, and terrestrial transmission systems, video content acquisition and editing systems, camcorders, security applications, Internet and mobile network video, Blu-ray Discs, and real-time conversational applications such as video chat, video conferencing, and telepresence systems and the growing popularity of HD video, and the emergence of beyond- HD formats (e.g., 4kx2k or 8kx4k resolution) are creating even stronger needs for coding efficiency superior to H.264/MPEG-4 AVC's capabilities.[13] HEVC has 50 percent more bit rate savings than H.264/MPEG-4 AVC [6] at similar visual quality. HEVC has two major improvements compared to H.264/MPEG-4 AVC that is increased use of parallel processing and increased video resolution. There are many new features in HEVC like wave front processing, tiles; dependent slices etc. which are introduced into the new standard. This advantage of HEVC comes at the price of high encoder complexity.

## 2.2 HEVC Encoder

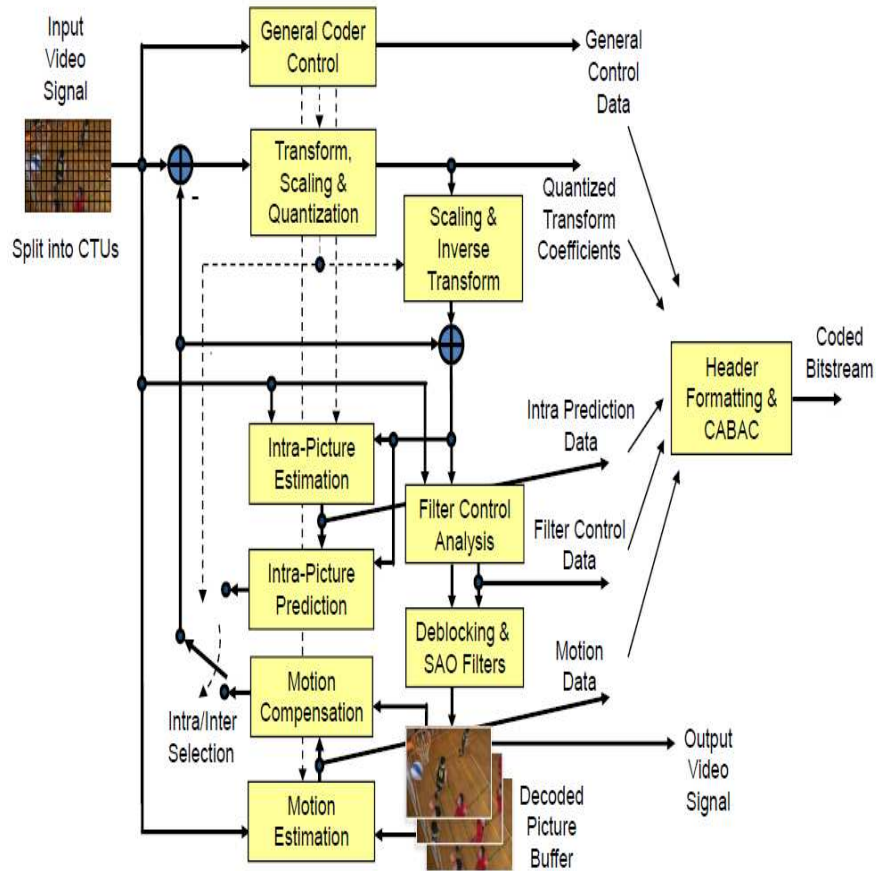


Figure 2-1 Block diagram of HEVC encoder [1]

### 2.2.1 Working of HEVC Encoder

Figure 2.1 [1] depicts the block diagram of a hybrid video encoder, which creates a bit stream according to the HEVC standard. The first frame of the video is entered in the encoder and it splits into corresponding coding tree units. The first frame is always intra-predicted. The difference of original image pixel and its predicted pixel is called the residual. This residual is transformed using integer approximations of discrete cosine



transform (DCT) for all blocks except luma 4x4 intra –predicted where a transform related to Discrete Sine transform (DST) is used. After transformation, quantization and scaling are performed to approximate the coefficient values. The decoder loop consists of inverse transform and scaling followed by filtering done by deblocking and SAO filters in the encoder block diagram. This reduces the error or drift between what the encoder predicts and what the decoder actually has. From this loop, the encoder can get information regarding the motion compensation and intra prediction. These are sent to entropy encoder which uses Context Adaptive Binary Arithmetic coding (CABAC) to form the bit stream. Figure 2.2 shows the block diagram of HEVC decoder [3].

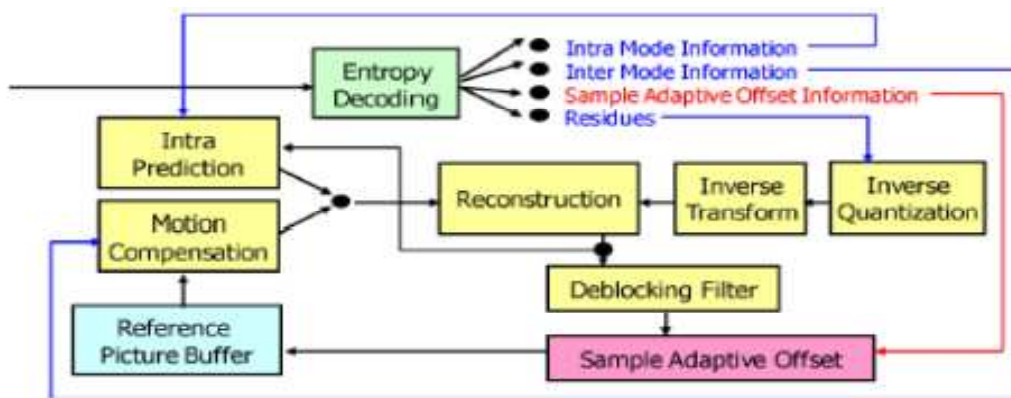


Figure 2-2 Decoder block diagram of HEVC [3]

### 2.2.2 Coding tree units

The first frame of the video is entered in the encoder and it splits into corresponding coding tree units. This is a new concept in HEVC which replaces the traditional macroblocks in H.264. A picture is partitioned into coding tree units (CTUs), which each contain luma Coding Tree Blocks (CTBs) and chroma CTBs. A luma CTB covers a rectangular picture area of  $L \times L$  samples of the luma component and the corresponding chroma CTBs cover each  $L/2 \times L/2$  samples of each of the two chroma components for the 4:2:0 and 4:4:4 formats as shown in figure 2.3 [22] . The value of  $L$

may be equal to 16, 32, or 64 as determined by an encoded syntax element specified in the Sequence Parameter Set (SPS). The luma CTB and the two chroma CTBs together with the associated syntax form a CTU. The CTU is the basic processing unit used in the standard to specify the decoding process.

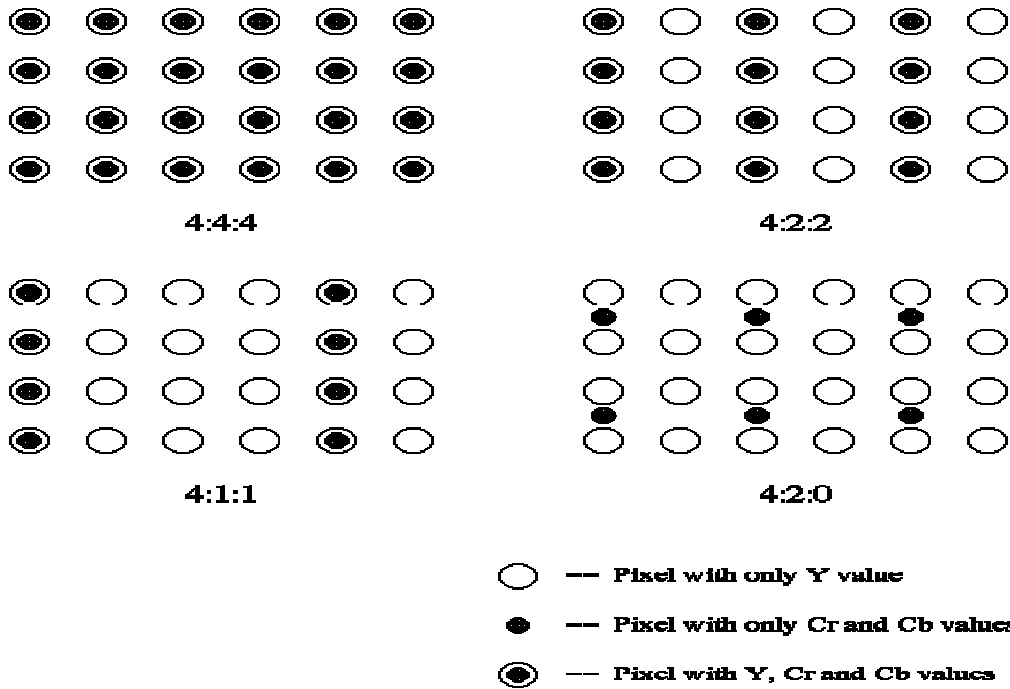


Figure 2.3 Format for YUV components [22]

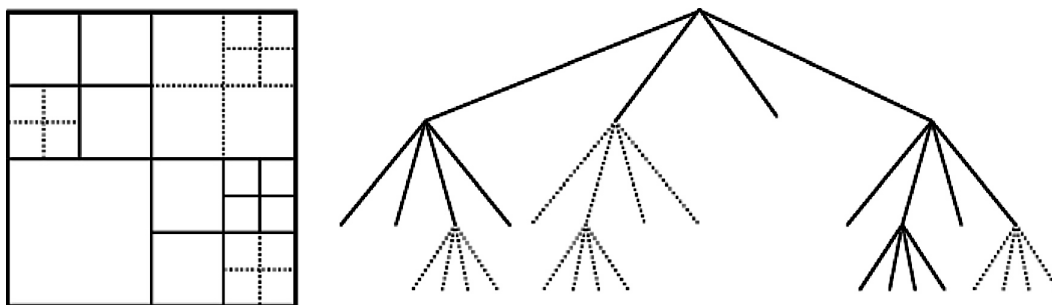


Figure 2-4 Left: CTB to CBs and TBs. Solid lines indicate CB boundaries and dotted lines indicate TB boundaries. Right: Corresponding quadtree.[1]

The blocks specified as luma and chroma CTBs can be directly used as CBs or can be further partitioned into multiple Coding Blocks (CBs). Partitioning is achieved

using tree structures as shown above in figure 2.4 [1]. The tree partitioning in HEVC is generally applied simultaneously to both luma and chroma, although exceptions apply when certain minimum sizes are reached for chroma. The CTU contains a quadtree syntax that allows for splitting the CBs to a selected appropriate size based on the signal characteristics of the region that is covered by the CTB. The quadtree splitting process can be iterated until the size for a luma CB reaches a minimum allowed luma CB size that is selected by the encoder using syntax in the Sequence Parameter Set and is always 8x8 or larger (in units of luma samples). [1]

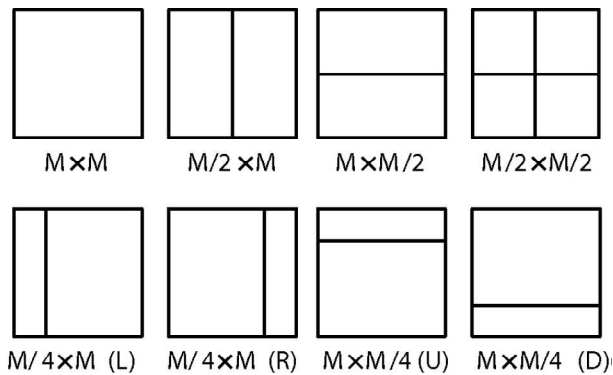


Figure 2-5 Modes for splitting a CB into PBs, subject to certain size constraints. For

intrapicture-predicted CBs, only  $M \times M$  and  $M/2 \times M/2$  are supported. [1]. D: Down, L: Left,

R: Right, U: Up.

The prediction mode for the CU is signaled as being intra or inter, according to whether it uses intrapicture (spatial) prediction or interpicture (temporal) prediction. When the prediction mode is signaled as intra, the Prediction Block (PB) size, which is the block size at which the intrapicture prediction mode is established, is the same as the Coding Block (CB) size for all block sizes except for the smallest CB size that is allowed in the bitstream. When the prediction mode is signaled as inter, it is specified whether the luma and chroma CBs are split into one, two, or four PBs as shown in figure 2.5 [1]. The luma

and chroma PBs, together with the associated prediction syntax, form the Prediction Unit (PU). For residual coding, a CB can be recursively partitioned into transform blocks (TBs). The partitioning is signaled by a residual quadtree. In contrast to previous standards, the HEVC design allows a TB to span across multiple PBs for interpicture-predicted CUs to maximize the potential coding efficiency benefits of the quadtree-structured TB partitioning. [5]

### 2.2.3 Intra Prediction

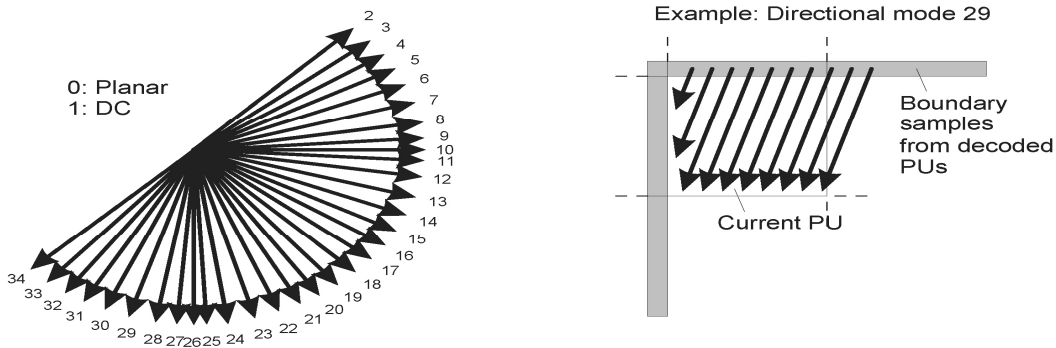


Figure 2-6 Intra prediction modes of HEVC encoder. [1]

Intrapicture prediction operates according to the TB size, and previously decoded boundary samples from spatially neighboring TBs are used to form the prediction signal. Directional prediction with 33 different directional orientations is defined for (square) TB sizes from 4x4 up to 32x32 as shown in figure 2.6 [1]. HEVC supports various intrapicture predictive coding methods referred to as Intra-Angular, Intra-Planar, and Intra-DC. HEVC supports a total of 33 prediction directions, denoted as Intra-Angular[k], where k is a mode number from 2 to 34. The angles are intentionally designed to provide denser coverage for near-horizontal and near-vertical angles and coarser coverage for near-diagonal angles to reflect the observed statistical prevalence of the angles and the effectiveness of the signal prediction processing. Intra-DC prediction uses an average

value of reference samples for the prediction. For Intra-Planar, average values of two linear predictions using four corner reference samples are used. [7]

### 2.3 Bitstream syntax of HEVC

The high-level syntax of HEVC mainly contains from the Network Adaptation Layer (NAL) [1] of H.264/MPEG4 AVC. The NAL provides the ability to map the Video Coding Layer (VCL) data that represent the content of the pictures onto various transport layers, including RTP/IP [11] , ISO MP4 [10], and H.222.0/MPEG2 [9] Systems, and provide a framework for packet loss resilience .The comparison between NAL units of H.264 and HEVC is shown in figure 2.7. [6][1]

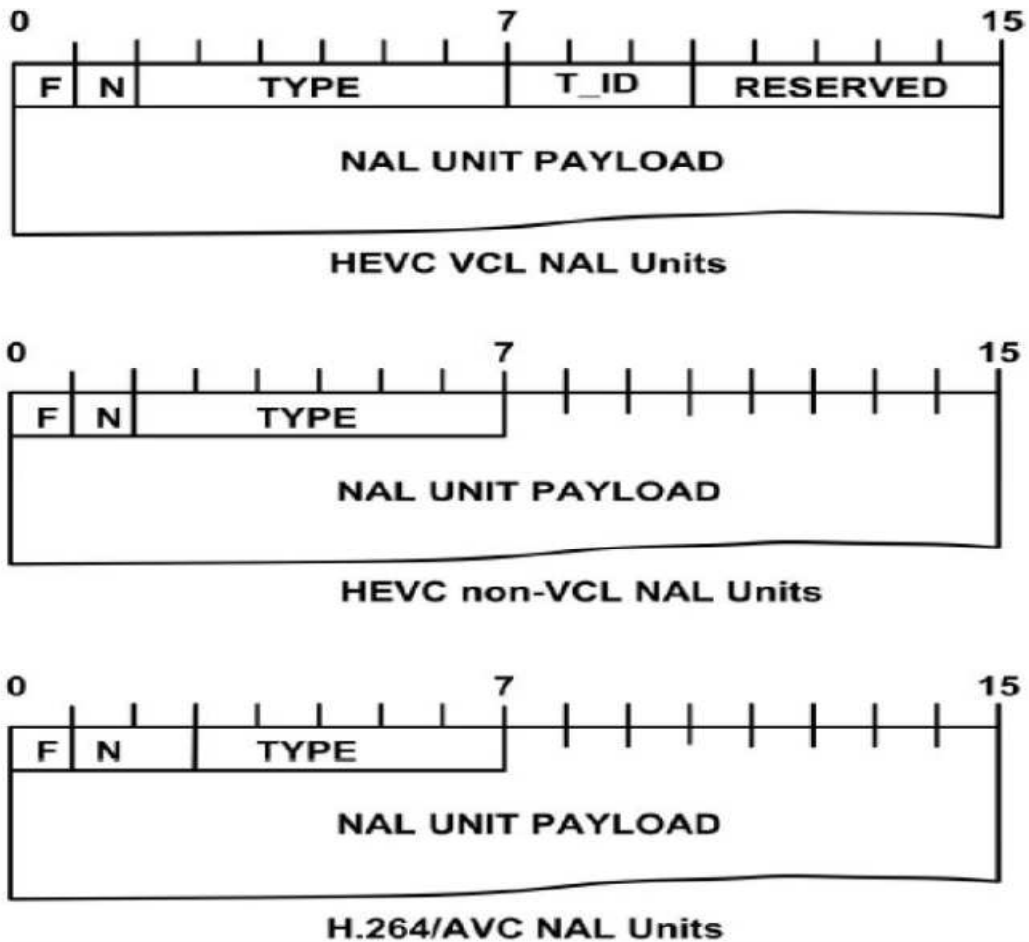


Figure 2-7 Comparison of HEVC and H.264 NAL units [1]

In HEVC each slice is encoded in a single NAL unit. HEVC uses a two byte NAL unit header. The size of a slice (and the subsequent NAL unit) may be matched to that of the Maximum Transmission Unit (MTU) of the network, over which the video will be streamed. NAL units are classified into VCL and non VCL NAL units according to whether they contain coded pictures or other associated data, respectively shown in Table 1 [1] [2].

Table 1-1 The NAL unit types and their associated meanings, classes in the HEVC standard. [1]

**TABLE I**  
**NAL UNIT TYPES, MEANINGS, AND TYPE CLASSES**

Type	Meaning	Class
0, 1	Slice segment of ordinary trailing picture	VCL
2, 3	Slice segment of TSA picture	VCL
4, 5	Slice segment of STSA picture	VCL
6, 7	Slice segment of RADL picture	VCL
8, 9	Slice segment of RASL picture	VCL
10–15	Reserved for future use	VCL
16–18	Slice segment of BLA picture	VCL
19, 20	Slice segment of IDR picture	VCL
21	Slice segment of CRA picture	VCL
22–31	Reserved for future use	VCL
32	Video parameter set (VPS)	non-VCL
33	Sequence parameter set (SPS)	non-VCL
34	Picture parameter set (PPS)	non-VCL
35	Access unit delimiter	non-VCL
36	End of sequence	non-VCL
37	End of bitstream	non-VCL
38	Filler data	non-VCL
39, 40	SEI messages	non-VCL
41–47	Reserved for future use	non-VCL
48–63	Unspecified (available for system use)	non-VCL

## 2.4 Video coding layer topics

### 2.4.1 Motion vector signaling

Advanced motion vector prediction (AMVP) is used. This includes derivation of several most probable candidates based on data from adjacent PBs and the reference

picture. A merge mode for motion vector coding is also used, allowing the inheritance of motion vectors from temporally or spatially neighboring PBs. [2] [4]

#### 2.4.2 Motion compensation

Quarter-sample precision is used for the motion vectors. 7-tap (weights: -1, 4, -10, 58, 17, -5, 1) or 8-tap (weights: -1, 4, -11, 40, 40, -11, 4, 1) filters are used for interpolation of fractional-sample positions as shown in Fig 2.8. For each PB, either one or two motion vectors can be transmitted, resulting either in unipredictive or bipredictive coding, respectively. [6][1]

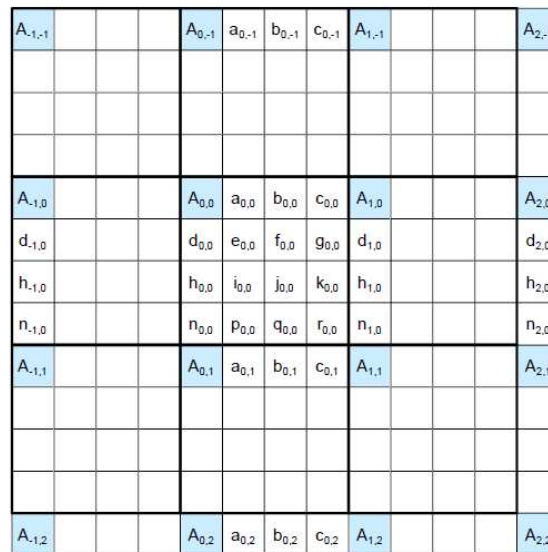


Figure 2-8 Integer and fractional sample positions for luma interpolation.  $A_{i,j}$ , represent the available luma samples at integer sample locations and the other positions labeled with lower-case letters represent samples at noninteger sample locations, which need to be generated by interpolation. [1] [2]

#### 2.4.3 Quantization control

As in H.264/MPEG-4 AVC [3], uniform reconstruction quantization (URQ) is used in HEVC, with quantization scaling matrices supported for the various transform block sizes. [2]



#### 2.4.4 Entropy Coding

HEVC specifies only one entropy coding method, CABAC [13] rather than two as in H.264/MPEG-4 AVC. Three coefficient scanning methods, diagonal up-right, horizontal, and vertical scans as shown in Fig. 2.9, are selected implicitly for coding the transform coefficients of 4x4 and 8x8 TB sizes in intrapicture-predicted regions.

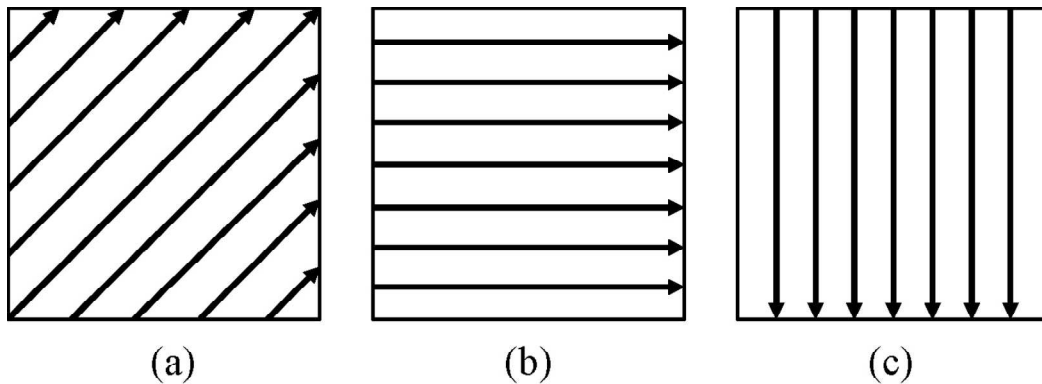


Figure 2-9 Three coefficient scanning methods in HEVC. (a) Diagonal up-right scan. (b) Horizontal scan. (c) Vertical scan [1]

#### 2.4.5 Deblocking Filter

The deblocking filter is applied to all samples adjacent to a PU or TU boundary except the case when the boundary is also a picture boundary, or when deblocking is disabled across slice or tile boundaries (which is an option that can be signaled by the encoder). It should be noted that both PU and TU boundaries should be considered since PU boundaries are not always aligned with TU boundaries in some cases of interpicture-predicted CBs. Syntax elements in the SPS and slice headers control whether the deblocking filter is applied across the slice and tile boundaries.[1][3]

#### 2.4.6 Sample Adaptive Offset (SAO)

A non-linear amplitude mapping is introduced in the inter-picture prediction loop after the deblocking filter. The goal is to better reconstruct the original signal amplitudes

by using a look-up table that is described by a few additional parameters that can be determined by histogram analysis at the encoder side. [3]

## 2.5 Parallel decoding syntax and modified slice structuring

Four new features are introduced in the HEVC standard to enhance the parallel processing capability or modify the structuring of slice data for packetization purposes.

### 2.5.1 Tiles

The option to partition a picture into rectangular regions called tiles has been specified. The main purpose of tiles is to increase the capability for parallel processing rather than provide error resilience. Tiles are independently decodable regions of a picture that are encoded with some shared header information. Tiles provide parallelism at a more coarse level of granularity (picture/subpicture), and no sophisticated synchronization of threads is necessary for their use. [1][4]

### 2.5.2 Wavefront parallel processing

When wavefront parallel processing (WPP) is enabled, a slice is divided into rows of CTUs. The first row is processed in an ordinary way, the second row can begin to be processed after only two CTUs have been processed in the first row, and the third row can begin to be processed after only two CTUs have been processed in the second row, and so on. WPP provides a form of processing parallelism at a rather fine level of granularity, i.e., within a slice.[1][2]

### 2.5.3 Dependent slice segments

A structure called a dependent slice segment allows data associated with a particular wavefront entry point or tile to be carried in a separate NAL unit, and thus potentially makes that data available to a system for fragmented packetization with lower latency than if it were all coded together in one slice.[1][5]

#### 2.5.4 Slices

A slice is a data structure that can be decoded independently from other slices of the same picture, in terms of entropy coding, signal prediction, and residual signal reconstruction. A slice can either be an entire picture or a region of a picture. One of the main purposes of slices is resynchronization in the event of data losses. [1][2]

#### 2.6 HEVC Profile, level.

Profiles, tiers, and levels specify conformance points for implementing the standard in an interoperable way across various applications that have similar functional requirements. A profile defines a set of coding tools or algorithms that can be used in generating a conforming bitstream, whereas a level places constraints on certain key parameters of the bitstream. Only three profiles targeting different application requirements, called the Main, Main 10, and Main Still Picture profiles, are finalized [1]. In the Main and Main Still Picture profiles, only a video precision of 8 bit per sample is supported, while the Main 10 profile supports up to 10 bit per sample. In the Main Still Picture profile, the entire bit stream must contain only one coded picture (and thus interpicture prediction is not supported) [1] [2] [4]. Extensions to these such as scalable video coding, 3D-TV etc are being finalized. [15]

#### 2.7 Summary

HEVC with main profile and intra prediction is used in this thesis. The main concepts of HEVC have been explained in chapter 2. Chapter 3 explains the AAC standard with detailed information.

## Chapter 3

### OVERVIEW OF ADVANCED AUDIO CODING (AAC)

#### 3.1 Introduction

Advanced audio coding (AAC) is a combination of state-of-the-art technologies in perceptual audio coding technology standardized by Moving Picture Experts Group (MPEG). AAC is an audio compression scheme first standardized within MPEG in 1997.[1] AAC has been standardized under the joint direction of the International Organization for Standardization (ISO) and the International Electro-Technical Commission (IEC), as part 7 of the MPEG-2 specification. Now, it is getting more popular for commercial purposes. Some of the important features added to AAC as compared to other standards are temporal noise shaping, backward adaptive linear prediction and enhanced joint stereo coding techniques which are used for applications like music delivery over cellular phone networks, “transparent” quality (indistinguishable from the original source material) for the most discriminating listeners. AAC allows using wide range of sampling rates (8–96 kHz), bit rates (16–576 kbps) and from one to 48 audio channels [3]. AAC provides audio of higher quality at the same bit rate as previous standards or same quality audio at lower bit rates. AAC is the first codec to fulfill the ITU-R/EBU requirements for indistinguishable quality at 128 kbps/stereo. In contrast to MP3's hybrid filter bank, AAC uses the modified discrete cosine transform (MDCT) together with the increased window lengths of 1024 or 960 samples. AAC can be used on HDTV, DVB, iTunes and iPod, iPhone, iPad, Apple TV, mobile phone, PDA and so on.

#### 3.2 AAC Profiles

Three default profiles have been defined, using different combinations of the available tools:

Main Profile: Uses all the encoding and decoding tools except the gain control module. This is the most complex of the three profiles and provides the highest quality for applications where the amount of random accessory memory (RAM) and processing power are not constraints.

Low-complexity Profile: Deletes the prediction tool and reduces the temporal noise shaping tool in complexity. This profile is favorable if memory and power constraints are to be met.

Scaleable sampling rate (SSR) Profile: Adds the gain control tool to the low-complexity profile. Allows the least complex decoder. This profile is most appropriate in applications with reduced bandwidth.

### 3.3 AAC Encoder

The block diagram of the AAC encoder is shown in Fig. 3.1 [9]. It comprises a perceptual model, a filter bank, a temporal noise shaping (TNS) module, a joint-stereo coding module (intensity stereo and mid/side stereo), a quantizer and a noiseless coding module. All modules are controlled by the perceptual model and the rate/distortion control process. [8]

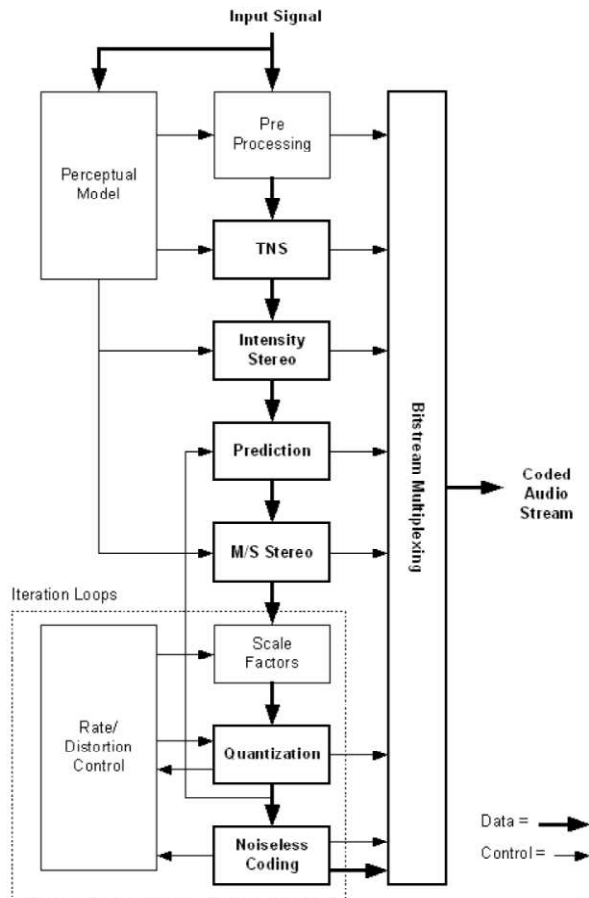


Figure 3-1 AAC encoder block diagram [9]

Filter Bank: The first task of an audio coder is to break an audio sample into segments, called blocks. A time domain filter, called a window, provides smooth transitions from block to block by modifying the data in these blocks. AAC uses modified discrete cosine transform (MDCT) in the filter bank module. Generally, transform coding controls the quantization noise in the MDCT component based on the frequency masking property. AAC uses two types of transform sizes according to the stationarity of the input signal. The transform size is fixed at 1,024 sample (long block mode) for a stationary segment and 128 points (short block mode) for a transient segment. The use of the short block

mode efficiently reduces the degradation called “pre-echo”. AAC also switches between two different types of long blocks: sine-function and Kaiser-Bessel derived (KBD) according to the complexity of the signal [4] [8].

Temporal Noise Shaping (TNS): The TNS technique provides enhanced control of the location, in time, of quantization noise within a filter bank window. This allows for signals that are somewhere between steady state and transient in nature. The TNS module temporally controls the quantization noise in the encoded signal obtained by inverse MDCT according to the temporal masking characteristics of human auditory perception. In this module, the magnitude of the quantization noise is controlled in proportion to the signal strength through frequency-domain linear prediction (FDLP). Note that TNS can be applied to either the entire frequency spectrum, or to only a part of the spectrum, such that the time-domain quantization can be controlled in a frequency-dependant fashion.

Intensity Stereo: The intensity stereo module encodes input stereo signals using a monaural signal and the spatial localization information. This module is very effective in the bit reduction when the stereo signal is a source with specific sound localization.

It is based on an analysis of high-frequency audio perception based on the energy-time envelope of the region of the audio spectrum. Intensity stereo coding allows a stereo channel pair to share a single set of spectral values for the high-frequency components with little or no loss in sound quality. This is achieved by maintaining the unique envelope for each channel by means of a scaling operation so that each channel produces the original level after decoding.[8]

Prediction: The prediction module is used to represent stationary or semi-stationary parts of an audio signal. Instead of repeating such information for sequential windows, a simple repeat instruction can be passed, resulting in a reduction of redundant information. The prediction process is based on a second-order backward adaptive model in which the

spectral component values of the two preceding blocks are used in conjunction with each predictor. The prediction parameter is adapted on a block-by-block basis. [6]

Mid/Side (M/S) Stereo Coding: The mid/side (M/S) stereo module encodes the input stereo signal in terms of Hadamard transformed signals, that is, the mid-signal obtained by adding the left and right signals, and the side signal obtained by subtracting the right signal from the left signal. This module is effective when the left and right signals are highly correlated. [8]

Quantization and Coding: While the previously described modules attain certain levels of compression, it is in the quantization phase that the majority of data reduction occurs. This is the AAC module in which spectral data is quantized under the control of the psychoacoustic model. The number of bits used must be below a limit determined by the desired bit rate. Huffman coding is also applied in the form of twelve codebooks. In order to increase the coding gain, scale factors with spectral coefficients of value zero are not transmitted. [5]

Noiseless Coding: This method is nested inside of the previous module, Quantization and Coding. Noiseless dynamic range compression can be applied prior to Huffman coding. A value of +/- 1 is placed in the quantized coefficient array to carry the sign, while magnitude and an offset from base, to mark frequency location, are transmitted as side information. This process is only used when a net savings of bits results from its use. Up to four coefficients can be coded in this manner.

Bit stream Multiplexing: AAC has very flexible bit stream syntax. A single transport is not ideally suited to all applications, and AAC can accommodate two basic bit stream formats: audio data interchange format (ADIF) and audio data transport stream (ADTS).

ADIF (audio data interchange format) actually is just one header at the beginning of the AAC file. The rest of the data are consecutive raw data blocks. This file format is



meant for simple local storing purposes, where breaking of the audio data is not necessary. [1] [2]

ADTS (audio data transport stream) has one header for each frame followed by raw block of data. ADTS headers are present before each AAC raw data block or block of 2 to 4 raw data blocks in a frame to ensure better error robustness in streaming environments. Hence in this thesis, ADTS bit stream format is adopted. The details of the ADTS header are given in Tables 3.1 And 3.2.

Table 3-1 ADTS header format [3]

Field name	Field size in bits	Comment
		ADTS Fixed header: these do not change from frame to frame
Syncword	12	always: '111111111111'
ID	1	0: MPEG-4, 1: MPEG-2
Layer	2	always: '00'
protection_absent	1	
Profile	2	
Sampling_frequency_index	4	
private_bit	1	
channel_configuration	3	
original/copy	1	
Home	1	

Table 3.1 – Continued

		ADTS Variable header: This can change from frame to frame
Copyright_identification_bit	1	
Copyright_identification_start	1	
aac_frame_length	13	length of the frame including header (in bytes)
ADTS_buffer_fullness	11	0x7FF indicates VBR
No_raw_data_blocks_in_frame	2	
		ADTS Error check
crc_check	16	Only if protection_absent == 0
Raw block of data	Variable	

Table 3-2 ADTS profile bits in header [3]

Profile bits	ID 1 (MPEG-2 profile)
00 (0)	Main profile
01 (1)	Low complexity profile (LC)
10 (2)	Scalable sample rate profile (SSR)
11 (3)	(reserved)

### 3.4 Summary

In this chapter, the AAC audio coding standard is discussed with a detailed description of the encoding process. Low complexity profile with ADTS bit stream formatting is used as per the advantages explained in the chapter. Chapter 4 focuses on the MPEG-2 transport stream and the multiplexing algorithm implemented.

## Chapter 4

### MULTIPLEXING

#### 4.1 Introduction

A multimedia file consists of video, audio and other metadata like subtitles. Each of the video, audio or data has to be transported in different media. In order to reduce the cost, and to optimize the use of expensive resources, multiplexing has been introduced. Multiplexing is the process of combining different bit streams into one single signal over a shared medium. Thus, elementary streams of video, audio and data are multiplexed into one single stream that would carry all the data and transmitted over the network. Video and audio streams contain several number of frames, thus are of large bandwidth. A high quality video/audio will require large bandwidth for transmission. Therefore, compression and coding standards like H.264/AVC [7], HEVC/H.265 [8] etc are employed for video and AAC [9], HE-AAC [10] for audio. After compression, the quality remains almost the same but the bandwidth requirement is reduced. For example, in the DVB (satellite TV) world, a satellite needs to deliver, via radio, one stream to subscribers. That one stream needs to carry many TV channels. To do this, the many channels are multiplexed into one stream. There are new standards stated for multiplexing like MPEG-2, RTP etc. In this thesis, MPEG-2 transport stream [3] is used.

#### 4.2 MPEG Bitstream Structure

To understand how the component parts of the bit stream are multiplexed, there is a need to first look at each component part. The most basic component is known as an *Elementary Stream* in MPEG. A program (perhaps most easily thought of as a television program, or a Digital Versatile Disk (DVD) track) contains a combination of elementary streams (typically one for video, one or more for audio, control data, subtitles, etc). Figure 4.1 shows the streams and formats supported by MPEG -2 [5]

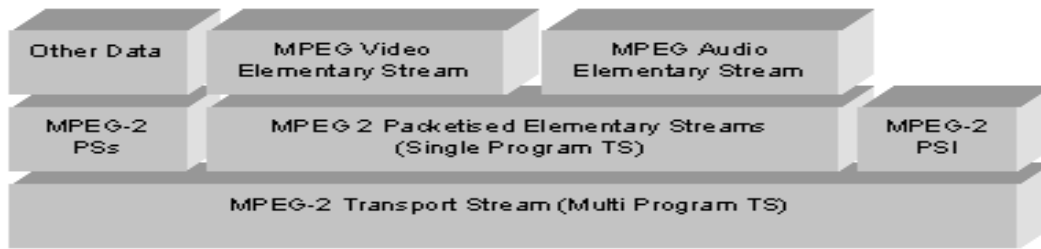


Figure 4-1 Streams supported by MPEG-2 [5]

#### 4.2.1 Elementary Stream

Each *Elementary Stream (ES)* output by an MPEG audio, video and (some) data encoders contain a single type of (usually compressed) signal. There are various forms of ES, including:

- Digital Control Data
- Digital Audio (sampled and compressed)
- Digital Video (sampled and compressed)
- Digital Data (synchronous, or asynchronous)

For video and audio, the data is organized into access units, each representing a fundamental unit of encoding. For example, in video, an access unit will usually be a complete encoded video frame. [5] [1]

#### 4.2.2 Packetized Elementary Stream (PES)

Each ES is input to an MPEG-2 processor (e.g. a video compressor or data formatter) which accumulates the data into a stream of Packetized Elementary Stream (PES) packets. A PES packet may be a fixed (or variable) sized block, with up to 65536 bytes per block and includes a 6 byte protocol header. A PES is usually organized to contain an integral number of ES access units as shown in figure 4.2. [2]

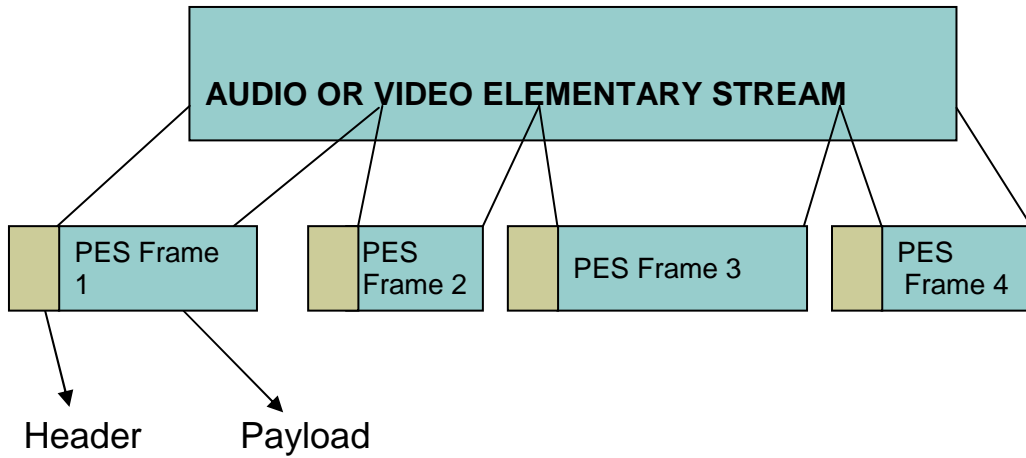


Figure 4-2 PES encapsulation from elementary stream [2]

The PES header starts with a 3 byte start code, followed by a one byte stream ID and a 2 byte length field. Figure 4.3 and Table 4.1 show the PES packet glossary. [5]

The following well-known stream IDs are defined in the MPEG standard:

- 110x xxxx - MPEG-2 audio stream number x xxxx.
- 1110 yyyy - MPEG-2 video stream number yyyy. [5]

Table 4-1 PES packet header description [7]

Name	Size	Description
Packet start code prefix	3 bytes	0x000001
Stream id	1 byte	Unique id for each audio and video stream
PES Packet length	2 bytes	Can be zero if more than 65536.
Timestamp	2 bytes	Frame number
Data		

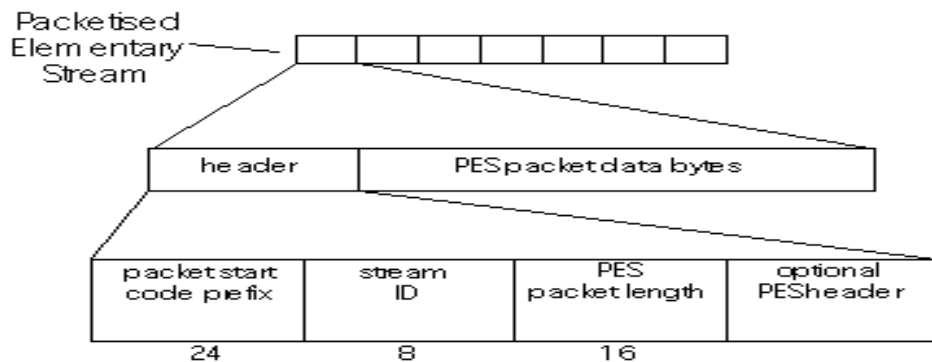


Figure 4-3 PES packet header [4]

The next field contains the PES Indicators. These provide additional information about the stream to assist the decoder at the receiver. The following indicators are defined:

PES\_Scrambling\_Control - Defines whether scrambling is used, and the chosen scrambling method.

PES\_Priority - Indicates priority of the current PES packet.

data\_alignment\_indicator - Indicates if the payload starts with a video or audio start code.

copyright information - Indicates if the payload is copyright protected.

original\_or\_copy - Indicates if this is the original ES.

A one byte flags field completes the PES header. This defines the following optional fields, which if present, are inserted before the start of the PES payload.

The PES packet payload includes the ES data. The information in the PES header is, in general, independent of the transmission method used.

### 4.3 MPEG-2 Multiplexing

The MPEG-2 standard allows two forms of multiplexing:

- MPEG Program Stream: A group of tightly coupled PES packets referenced to the same time base. Such streams are suitable for transmission in a relatively error-free

environment and enable easy software processing of the received data. This form of multiplexing is used for video playback and for some network applications. [2]

- MPEG Transport Stream: Each PES packet is broken into fixed-sized transport packets forming a general purpose way of combining one or more streams, possibly with independent time bases. This is suitable for transmission in which there may be potential packet loss or corruption by noise, and / or where there is a need to send more than one program at a time. [2]

Digital Video Broadcast (DVB) uses the MPEG-2 transport stream over a wide variety of underlying networks. Since both the program stream and transport stream multiplex a set of PES inputs, interoperability between the two formats may be achieved at the PES level.

#### 4.4 MPEG-2 Transport Stream

A MPEG-2 transport stream, also referred to as MPEG or MPEG-2 TS or simply TS, is a special format for transmitting MPEG (MPEG-1, MPEG-2, or MPEG-4) video multiplexed with other streams. It is commonly used for digital television and streaming across networks, including the internet.[6]

Unlike programs streams, which are optimized for efficient storage and assume the decoder has access to the entire stream for synchronization purposes, transport streams are designed for delivering data in real time over unreliable transport media, to a device which is assumed to start reading data from some point after the beginning of transmission.

In order to accommodate this, extra timestamps must be added to the stream at regular intervals, with synchronization of various packets (chunks of elementary streams) set relative to the most recent timestamp instead of a single point at the beginning of the file like a program stream.



A transport stream consists of a sequence of fixed sized transport packets of 188 bytes. Each packet comprises 184 bytes of payload and a 4 bytes header. One of the items in this 4 bytes header is the 13 bit packet identifier (PID) which plays a key role in the operation of the transport stream.



Figure 4-4 Single Program Transport Stream (Audio and Video PES).

Figure 4.4 shows two elementary streams sent in the same MPEG-2 transport multiplex. Each packet is associated with a PES through the setting of the PID value in the packet header (the values of 64 and 51 in the figure 4.4). The audio packets have been assigned PID 64, and the video packets PID 51 (these are arbitrary, but can be different values). As is usual, there are more video than audio packets, but the two types of packets are not evenly spaced in time. The MPEG-TS is not a time division multiplex, packets with any PID may be inserted into the TS at any time by the TS multiplexor. If no packets are available at the multiplexor, it inserts null packets (denoted by a PID value of 0x1FFF) to retain the specified TS bit rate. The multiplexor also does not synchronize the two PESs; indeed the encoding and decoding delay for each PES may be and usually is different. A separate process is therefore required to synchronize the two streams. [5]

#### 4.5 Format of Transport Stream packet

Each MPEG-2 TS packet carries 184 bytes of payload data prefixed by a 4 bytes (32 bit) header as shown in Figure 4.5 and Table 4.2 [4].

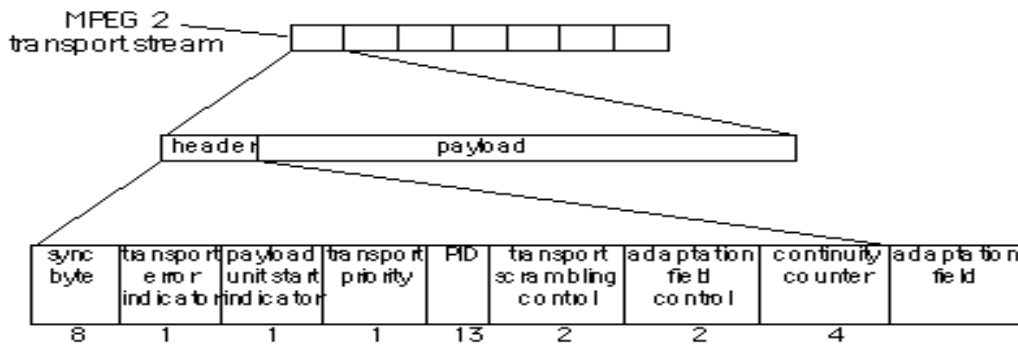


Figure 4-5 Transport stream (TS) header [4]

Table 4-1 PES packet header description [7]

Abbr	Function
SB	Synchronization Byte
TEI	Transport Error Indicator
PUSI	Payload Unit Start Indicator
TSC	Transport Scrambling Control
TP	Transport Priority
PID	Packet Identifier
AFC	Adaptation Field Control
CC	Continuity Counter
AF	Adaptation Field (Optional)

The header starts with a well-known synchronization byte (8 bits). This has the bit pattern 0x47 (0100 0111).

A set of three flag bits are used to indicate how the payload should be processed. The first flag indicates a transport error. The second flag indicates the start of a payload (`payload_unit_start_indicator`). The third flag indicates transport priority bit.

The flags are followed by a 13 bit packet identifier (PID). This is used to uniquely identify the stream to which the packet belongs (e.g. PES packets corresponding to an ES) generated by the multiplexer. The PID allows the receiver to differentiate the stream to which each received packet belongs. Some PID values are predefined and are used to indicate various streams of control information. A packet with an unknown PID, or one with a PID which is not required by the receiver, is silently discarded. The particular PID value of 0x1FFF is reserved to indicate that the packet is a null packet (and is to be ignored by the receiver).

The two scrambling control bits are used by conditional access procedures to encrypt the payload of some TS packets.

Two adaptation field control bits which may take four values:

01 – no adaptation field, payload only

10 – adaptation field only, no payload

11 – adaptation field followed by payload

00 - RESERVED for future use.

Finally there is a half byte *continuity counter* (4 bits).

Figure 4.6 shows how the audio and video PES packets are placed in MPEG-2

TS. [2]

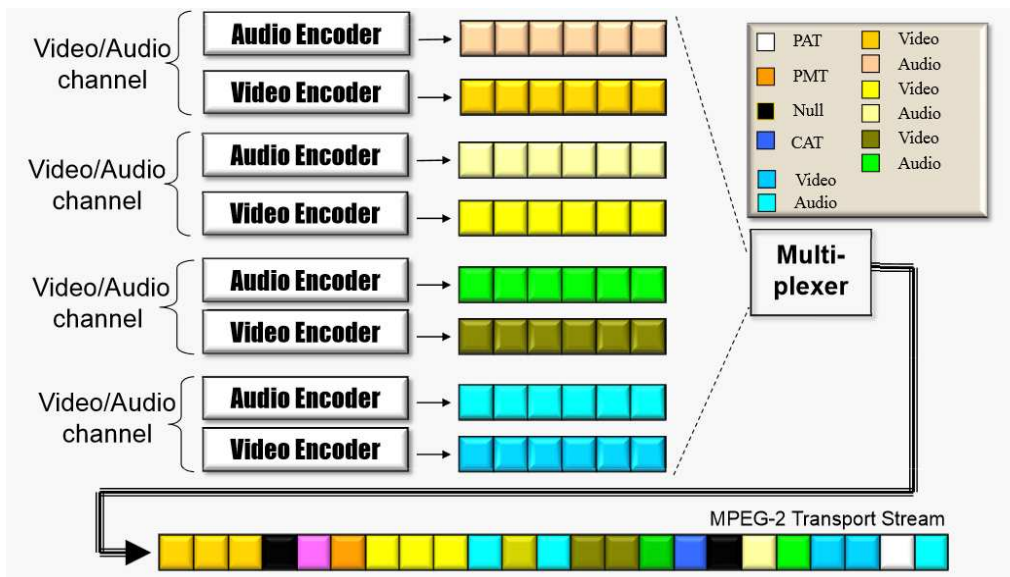


Figure 4-6 Multiplexing of audio data and video data into MPEG-2 transport stream. [2]

Two options are possible for inserting PES data into the TS packet payload:

The simplest option, from both the encoder and receiver viewpoints, is to send only one PES (or a part of single PES) in a TS packet. This allows the TS packet header to indicate the start of the PES, but since a PES packet may have an arbitrary length, also requires the remainder of the TS packet to be padded, ensuring correct alignment of the next PES to the start of a TS packet. In MPEG-2 the padding value is the hexadecimal byte 0xFF.

In general a given PES packet spans several TS packets so that the majority of TS packets contain continuation data in their payloads. When a PES packet is starting, however, the payload\_unit\_start\_indicator bit is set to '1' which means the first byte of the TS payload contains the first byte of the PES packet header. Only one PES packet can start in any single TS packet. The TS header also contains the PID so that the receiver can accept or reject PES packets at a high level without burdening the receiver with too

much processing. Figure 4.7 shows the combining of elementary streams into transport stream [4].

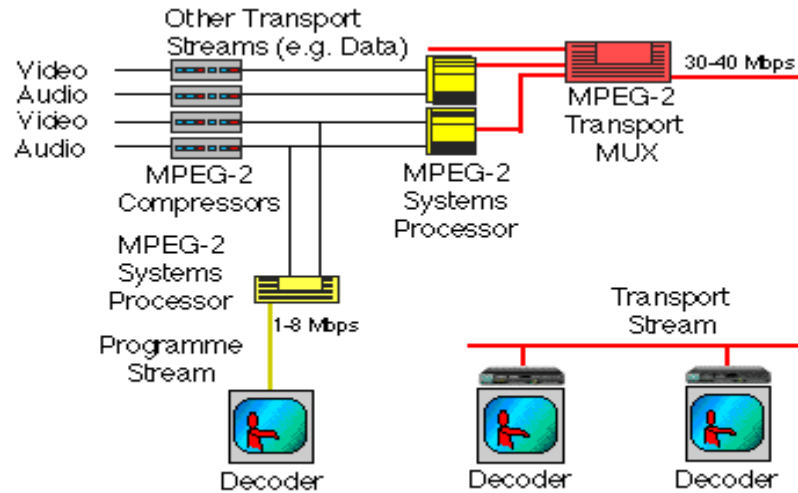


Figure 4-7 Combining ES from encoders into TS (red) or a PS (yellow) [4]

#### 4.6 Frame number as Time Stamp

This thesis proposes a method to use the frame number as timestamps. This section explains how frame numbers can be used to synchronize audio and video streams. Both H.265 [2] and AAC [3] bit streams are composed of data blocks sorted into frames. A particular video bit stream has a constant frame rate during playback specified by frames per second (fps). Hence, given the frame number, one can calculate the time of occurrence of this frame in the video sequence during playback as follows:

$$\text{Time of playback} = \text{Frame number} / \text{fps} \quad (4-1)$$

The AAC compression standard defines each audio frame to contain 1024 samples. The audio data in the AAC bit stream can have any discrete sampling frequency between 8 KHz and 96 kHz. The frame duration increases from 96 kHz to 8 kHz. However, the sampling frequency and hence the frame duration remain constant

throughout a particular audio stream. So, the time of occurrence of the frame during playback is as follows:

$$\text{Time of playback} = 1024 * \text{frame number} / (\text{sampling freq}). \quad (4-2)$$

Thus from (4-1) and (4-2) it is found the time of playback by encoding the frame numbers as the time stamps. In other words, given the frame number of one stream, the frame number of the other streams that will be played at the same time as the frame of the first stream. This will help to synchronize the streams during the playback. This idea can be extended to synchronize more than one audio stream with the single video stream like in the case of stereo or programs with single video and multiple audio channels.

The timestamp is assigned in the last 2 bytes of the PES packet header. This implies that timestamp can carry frame numbers up to 65536. Once the frame number exceeds this, in the case of long video and audio streams, the frame number is rolled over. The rollover takes simultaneously on both audio and video frame numbers as soon as either one of the stream crosses the maximum allowed frame number. This will not create a conflict at the demultiplexer during synchronization because the audio and video buffer sizes are much smaller than the maximum allowed frame number. Hence, at no point of time there will be two frames in the buffer with the same timestamp. [7]

#### 4.7 Proposed Multiplexing method

The final transmission stream is formed by multiplexing the TS packets of the various elementary streams. The number of packets allocated for a particular elementary stream during transmission, plays an important part in avoiding buffer overflow or underflow at the demultiplexer. If more video TS packets are sent as compared to audio TS packets, then at the receiver there may be a situation when video buffer is full and is overflowing whereas audio buffer does not have enough data. This will

prevent the demultiplexer from starting a playback and will lead to loss of data from the overflowing buffer.

In order to prevent such a scenario, timing counters are employed at the multiplexer. Each elementary stream has a timing counter, which gets incremented when a TS packet from that elementary stream is transmitted. The increment value depends on the playback time of the TS packet. The playback time of each PES can be calculated since the frame duration is constant in both audio and video elementary streams. By finding out how many TS packets are obtained from a single PES packet, the playback time of each TS packet can be calculated. The elementary stream whose counter has the least timing value is always given preference in packet allocation. This method will make sure that at any point of time, the difference in the fullness of the buffers, in terms of playback time is less than the playback time of one TS packet. This is never more than the duration of a single frame and is typically in milliseconds. The flowchart of multiplexing process as shown in figure 4.8.

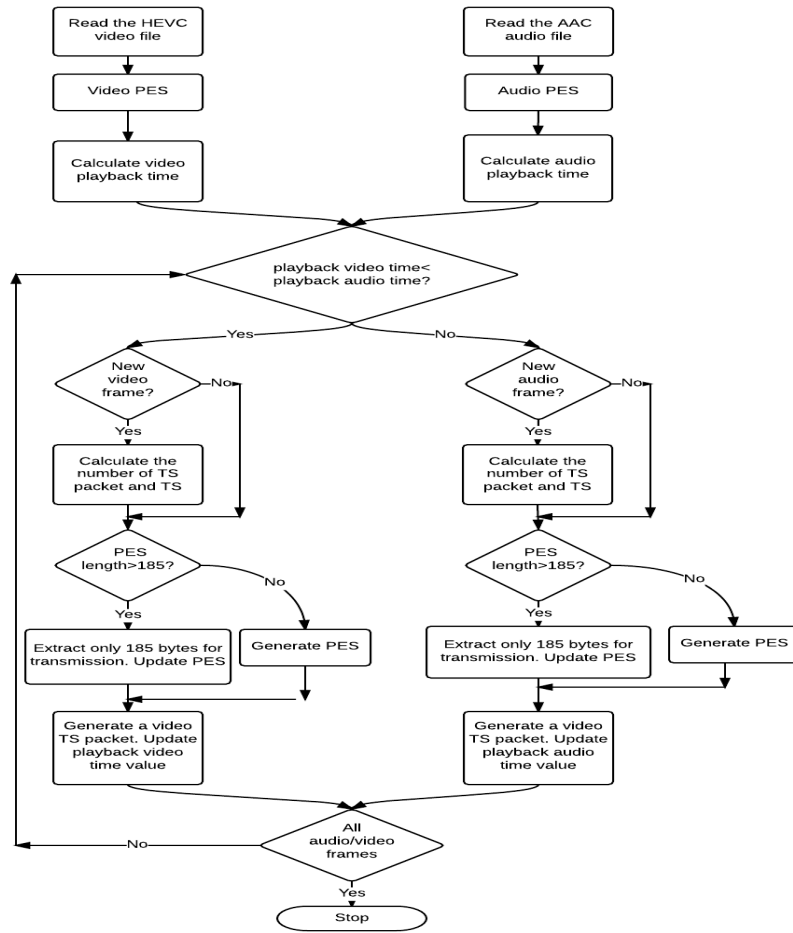


Figure 4-8 Flowchart of multiplexing process.

#### 4.8 Summary

This chapter provides with the information of the standard MPEG -2 transport stream and how it is used for multiplexing video and audio bitstream. Eventually, a method for multiplexing the TS packets is proposed that can prevent buffer overflow or underflow at the demultiplexer. Next chapter gives the procedure for demultiplexing the audio and video streams and lip synchronization.



## Chapter 5

### DEMULTIPLEXING AND SYNCHRONIZATION

#### 5.1 Demultiplexing

Demultiplexing performs the reverse function of multiplexing that is split a combined stream arriving from a shared medium into the original information streams as shown in Figure 5.1 [15]. The process of extracting the elementary streams from the multiplexed transport stream is called demultiplexing. This is the first step carried out at the receiver, in the process of delivering a complete multimedia program to the end user.

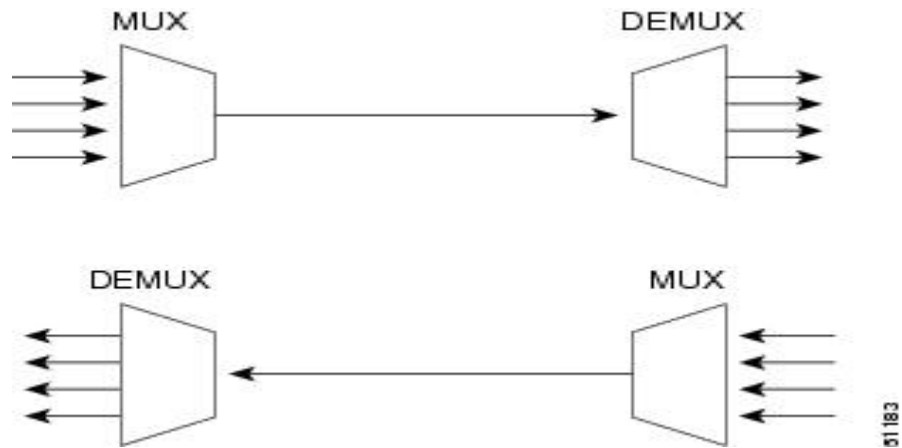


Figure 5-1 Overview of multiplexing and demultiplexing.[23]

The flowchart of the demultiplexer algorithm is shown in Fig 5.2.

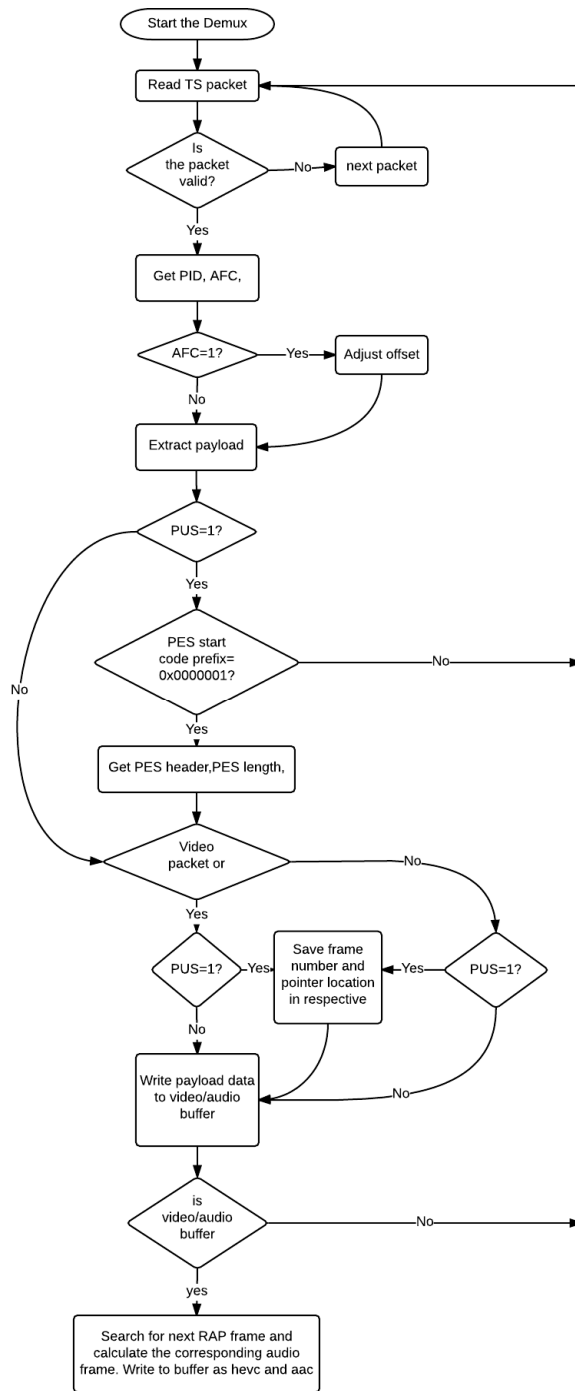


Figure 5-2 Flowchart of the demultiplexer.

The procedure is almost reverse as in the multiplexing process and it is explained below.

When the packets from the transport stream is received, the corresponding packet identifier (PID) values are obtained. If the packet has value 14, then it is recognized as an audio packet or if the packet value is 15, it is recognized as a video packet. If the packet has any PID value that is not relevant to the multimedia program that is being recovered, the packet is dropped and the next packet is analyzed.

At this stage, all the TS packets from other programs or null packets are discarded as there should not be unwanted data to be added in the decoded output. When the packet is found out to be the correct one, it is checked for more details. The 'adaptation field control' (AFC) bit is checked to see if any data other than the elementary stream data is present in the packet. If the AFC bit is set, then the data is recovered from the payload starting from the byte obtained by reading the 'byte offset value' from the header. The remaining data is rejected if it is filled with stuffing bytes.

The data is identified to be audio data or video data through the PID value and is redirected to the appropriate buffer. The packet is also analyzed to check if the 'payload unit start' (PUS) bit is set. If it is set, then the PES header is present in the packet. The header information is read to recover the frame length and timestamp, which is the frame number. The frame number and location of the frame in the data buffer are stored in a separate buffer. This process is continued until one of the elementary stream buffers is full.

In order to detect packet losses, 4 bit continuity counter value is continuously monitored for each PID separately, to check if the counter value increments in sequence. If not a packet loss is declared and the particular frame in the buffer, which is involved in the loss, is marked to be erroneous. In some transmission schemes, retransmission of

the packet is requested to correct the error. Otherwise, the frame is skipped during playback to prevent any stall in the decoder.

The most important thing is to continuously monitor the fullness of the elementary stream buffers. The buffer should not be allowed to overflow or underflow. This will lead to loss of data. This is taken into account during the multiplexing process as explained in section 4.4.

## 5.2 Synchronization and playback

Once the elementary stream buffer is full, the content is ready to be played back for the viewer. The audio bit stream format i.e., audio data transport stream (ADTS), enables us to begin decoding from any frame. However, the video bit stream does not have the same kind of sophistication. The decoding can start only from the anchor frames, which are the IDR frames. IDR frames are forced during the encoding process at regular intervals. Hence, the video buffer is first searched from the top to get the first occurring IDR frame. Once this is found, the timestamp or the frame number is obtained for that IDR frame. Then audio stream is aligned accordingly to achieve synchronization. This is done by calculating the audio frame number that would correspond to the IDR frame in terms of playback time. This is calculated as follows:

$$\text{Audio frame number} = \frac{\text{Video framenumbersamplingfrequency}}{1024 \times \text{fps}}$$

(5-1)

If the frame number calculated is a non-integer value, then the value is rounded off and the corresponding frame is taken. The round off error is discussed later in this

chapter. If the calculated frame number is not found in the buffer, next IDR frame is searched and a new audio frame number is calculated. Once video frame number and audio frame number are obtained, the location of these frames is looked up in the buffer and the block of data from this frame to the end of the buffer is taken and sent to the decoder for playback. The buffer is then emptied and the incoming data is filled in the buffer and the process is repeated. In order to have a continuous playback, block of data from first IDR frame to last IDR frame in the buffer is played back and during this playback the next set of data buffering takes place in the background. This process continues and the program is continuously played for the viewer.

Once the buffer is full and the synchronized frames are calculated, the audio and video content are played back. The buffering is continued after the playback and next set of data is put through the same process. This process continues and thus successful demultiplexing and synchronized playback are achieved.

This completes the process of multiplexing, demultiplexing and a synchronized playback of multimedia programs.

### 5.3 Summary

In this chapter, the process of extracting data from the multiplexed TS packets, to reconstruct the elementary stream is explained in detail. Synchronization and playback method using frame numbers as timestamps are proposed and the limitations due to possible error in synchronization are discussed. Next chapter discusses the test conditions, results and conclusion obtained after implementing the procedure.

## Chapter 6

### RESULTS AND CONCLUSIONS

#### 6.1 Test conditions

In order to evaluate the proposed multiplexing algorithm, a single multimedia stream consisting of elementary video and elementary audio streams are used. Raw elementary video stream are in YUV format and raw elementary audio stream are present in WAVE format. There are many standard raw formats of individual audio and video elementary streams available but the combination of both is not freely available. Hence, an AVI or MOV file is used that is freely available. The raw YUV format of the video and the raw WAVE format of the audio is extracted using a very powerful open source software called ffmpeg[39]. The YUV video file is of very large size and it is encoded using HM 14.0 software[29]. The raw YUV file is used as an input to the HM and output is a .hevc file which is compressed. The encoder setting used is main intra profile with GOP structure as IBBB. The raw audio stream is encoded using open source software called FAAC[40]. The audio stream is an encoded in low-complexity profile. The input .WAVE is used as an input to the FAAC and output is an .aac file which is compressed as shown in figure 6.1. Both elementary streams are then multiplexed to form transport stream packets. The details of the test clips are described in Table 6.1:

Table 6-1 Test clip conditions

Test clip details	Clip 1: Morning.avi	Clip 2: SpeedBag
.avi file size (kB)	901	12650
.yuv file size (kB)	128115	390794
.hevc file size (kB)	57.1	391
.wave file size (kB)	1277	897
.aac file size (kB)	126	69
Clip Duration (sec)	9 sec	4 sec
Frame rate (frames/sec)	25	50
Audio frequency (KHz)	44.100	48
Video Compression Ratio	2242.06	999.4
Audio compression ratio	10.1349	13.1
No. of TS packets	1377	2704
Total TS size (kB)	128	141
.mkv movie size (kB)	192	462

The results in Table 6.1 clearly show that the compression achieved, using HEVC video codec and AAC audio codec, in the proposed method is more than that of the MKV movie files (.mkv) or AVI files. The .avi file is played using the VLC media player by VideoLAN[38]. Then the TS packets are given as the input to the proposed demultiplexer algorithm to achieve the synchronized playback. It is verified to see if synchronization between audio and video is achieved, irrespective of which TS packet the user starts demultiplexing from. The demultiplexed audio and video are obtained and merged using MKVtoolNix [42]. Since HEVC video is the latest standard, there are very

few players which play it. Open-source DivX player [41] with HEVC plug-in is used to play the final mkv movie file. Hence, some random TS packets are chosen to begin the demultiplexing process and the results of the synchronization process are given in Table 6.2. The delay between audio and video is less than 13 milliseconds in the observed cases.

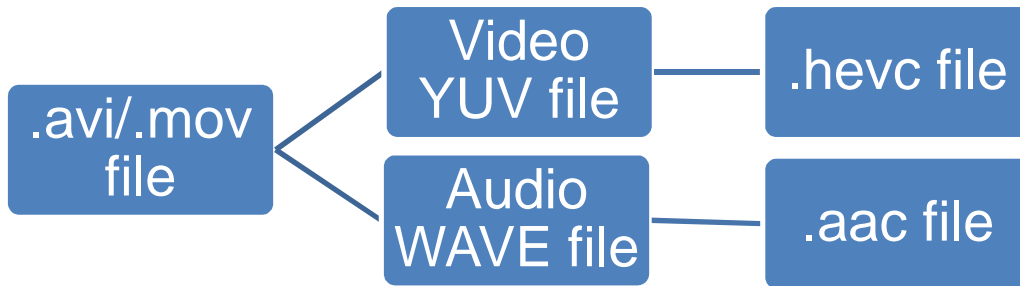


Figure 6-1 Test Condition for Video file

## 6.2 Results

Table 6-2 Lip synchronization results for clip 1

TS packet number	Synchroniz ed video frame	Synchroniz ed audio frame	Frame Playba ck	Audio playback	Delay ( msec)	Visual Delay (yes/no)
57	7	12	0.28	0.278	2	No
117	16	28	0.640	0.650	10	No
250	32	56	1.290	1.3	10	No
500	63	108	2.52	2.507	13	No
820	121	208	4.840	4.829	11	No
1001	153	264	6.120	6.130	10	No
1240	205	353	8.200	8.196	4	No



Table 6-3 Lip synchronization results for clip 2

TS packet number	Synchronized Video Frame	Synchronized Audio frame	Frame video playback	Frame audio playback	Delay	Visual Delay (yes/no)
61	6	5	120	106	14	No
251	18	17	360	362	2	No
524	27	25	540	533	7	No
1001	52	48	1040	1024	16	No
1500	88	82	1760	1749	11	No
2000	120	112	2400	2389	11	No
2500	132	124	2640	2645	5	No

```

Command Prompt
POC 209 TId: 0 < B-SLICE, nQP 35 QP 35 > 96 bits IY 89.6729 dB U 72.7836 dB U 72.6584 dB I ET 125 I IL0 208 204 200 196 I IL1 208 204 200 196 I
POC 210 TId: 0 < B-SLICE, nQP 34 QP 34 > 96 bits IY 89.6729 dB U 72.7836 dB U 72.6584 dB I ET 125 I IL0 209 208 204 200 1 IL1 209 208 204 200 1
POC 211 TId: 0 < B-SLICE, nQP 35 QP 35 > 96 bits IY 89.6729 dB U 72.7836 dB U 72.6584 dB I ET 125 I IL0 210 208 204 200 1 IL1 210 208 204 200 1
POC 212 TId: 0 < B-SLICE, nQP 33 QP 33 > 128 bits IY 89.6729 dB U 72.7836 dB U 72.6584 dB I ET 125 I IL0 211 208 204 200 1 IL1 211 208 204 200 1
POC 213 TId: 0 < B-SLICE, nQP 35 QP 35 > 96 bits IY 89.6729 dB U 72.7836 dB U 72.6584 dB I ET 125 I IL0 212 208 204 200 1 IL1 212 208 204 200 1
POC 214 TId: 0 < B-SLICE, nQP 34 QP 34 > 96 bits IY 89.6729 dB U 72.7836 dB U 72.6584 dB I ET 125 I IL0 213 212 208 204 1 IL1 213 212 208 204 1
POC 215 TId: 0 < B-SLICE, nQP 35 QP 35 > 96 bits IY 89.6729 dB U 72.7836 dB U 72.6584 dB I ET 126 I IL0 214 212 208 204 1 IL1 214 212 208 204 1
POC 216 TId: 0 < B-SLICE, nQP 33 QP 33 > 128 bits IY 89.6729 dB U 72.7836 dB U 72.6584 dB I ET 126 I IL0 215 212 208 204 1 IL1 215 212 208 204 1
POC 217 TId: 0 < B-SLICE, nQP 35 QP 35 > 96 bits IY 89.6729 dB U 72.7836 dB U 72.6584 dB I ET 126 I IL0 216 212 208 204 1 IL1 216 212 208 204 1
POC 218 TId: 0 < B-SLICE, nQP 34 QP 34 > 96 bits IY 89.6729 dB U 72.7836 dB U 72.6584 dB I ET 125 I IL0 217 216 212 208 1 IL1 217 216 212 208 1

SUMMARY
-----
Total Frames | Bitrate | V-PSNR | U-PSNR | U-PSNR | YUU-PSNR
219 | a | 52.6265 | 49.4518 | 49.7283 | 50.0563 | 43.5497

I Slices
-----
Total Frames | Bitrate | V-PSNR | U-PSNR | U-PSNR | YUU-PSNR
1 | i | 1082.0000 | 41.1939 | 44.6092 | 45.5935 | 42.1462

P Slices
-----
Total Frames | Bitrate | V-PSNR | U-PSNR | U-PSNR | YUU-PSNR
0 | p | -1.#IND | -1.#IND | -1.#IND | -1.#IND | -1.#IND

B Slices
-----
Total Frames | Bitrate | V-PSNR | U-PSNR | U-PSNR | YUU-PSNR
218 | b | 47.9046 | 49.4897 | 49.7518 | 50.0768 | 43.5573

RUM: 0.000
Bytes written to file: 58513 <53.437 kbps>
Total Time: 41960.778 sec.
C:\Users\MRUDULA\Documents\SlikSov\bin\trunk\bin\vc9\Win32\Debug>

```

Figure 6-2 HEVC encoder with IBBB setting.



Figure 6-3 Morning test sequence, WVGA 832x480, 25 fps, 219 Frames



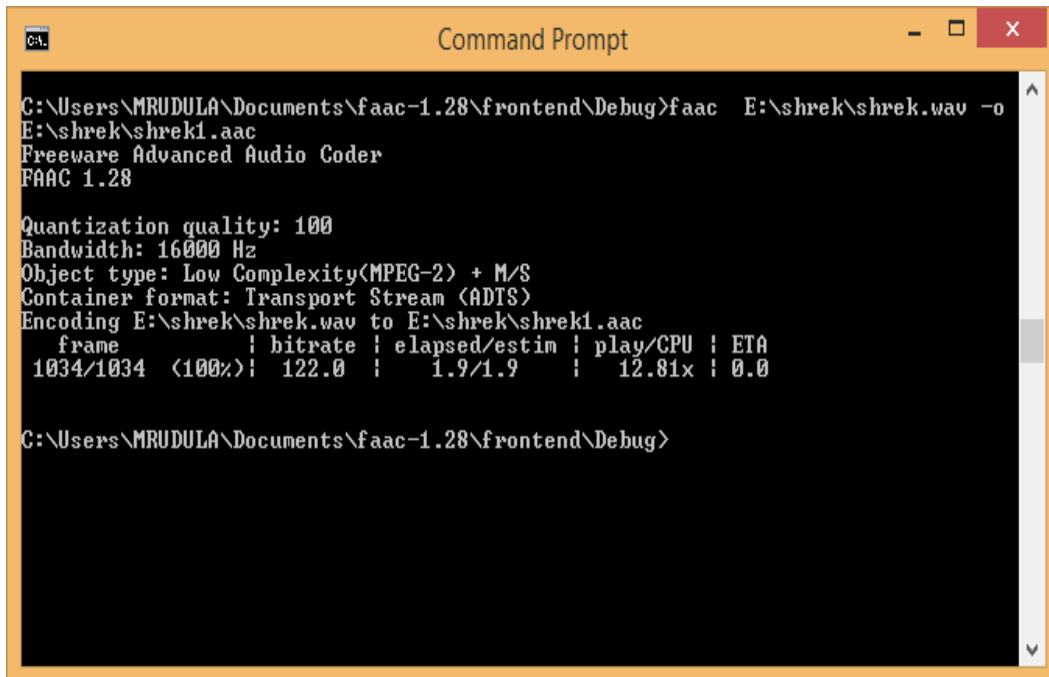
Figure 6-4 SpeedBag test sequence, SD 1280x720, 50 fps, 204 Frames

```

C:\ffmpeg\bin>ffmpeg -i E:\shrek\ShrekTheThird_study.avi -vn E:\shrek\shrek.wav
ffmpeg version N-66438-g4f4f08e Copyright (c) 2000-2014 the FFmpeg developers
  built on Sep 24 2014 22:35:54 with gcc 4.9.1 (GCC)
  configuration: --enable-gpl --enable-version3 --disable-w32threads --enable-av
  isynth --enable-bzlib --enable-fontconfig --enable-frei0r --enable-gnutls --enab
  le-iconv --enable-libass --enable-libbluray --enable-libbs2b --enable-libcaca --
  enable-libfreetype --enable-libgme --enable-libgsm --enable-libilbc --enable-lib
  modplug --enable-libmp3lame --enable-libopencore --enable-libopenmpt --enable-lib
  opus --enable-libopus --enable-librtmp --enable-libschroedinger --enable-libsoxtr
  --enable-libspeex --enable-libtheora --enable-libtwolame --enable-libvidstab --en
  able-libvoaacenc --enable-libvoamrwbenc --enable-libvorbis --enable-libvpx --enab
  le-libx265 --enable-libxavs --enable-libxvid --enable-zlib
  libavutil      54. 7.101 / 54. 7.101
  libavcodec     56. 1.101 / 56. 1.101
  libavformat    56. 5.100 / 56. 5.100
  libavdevice    56. 1.100 / 56. 1.100
  libavfilter     5. 1.102 / 5. 1.102
  libswscale     3. 1.100 / 3. 1.100
  libswresample  1. 1.100 / 1. 1.100
  libpostproc   53. 1.100 / 53. 1.100
Input #0, avi, from 'E:\shrek\ShrekTheThird_study.avi':
  Metadata:
    encoder         : Lavf54.29.104
  Duration: 00:00:23.99, start: 0.000000, bitrate: 655 kb/s
  Stream #0:0: Video: h264 (Main) (avc1 / 0x31637661), yuv420p, 854x480, 508 kb/s, 59.94 fps, 29.97 tbr, 59.94 tbn, 59.94 tbc
  Stream #0:1: Audio: aac (L2551101101101 / 0x00FF), 44100 Hz, stereo, fltp, 125 kb/s
Output #0, wav, to 'E:\shrek\shrek.wav':
  Metadata:
    ISFT            : Lavf56.5.100
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, stereo, s16, 1411 kb/s
  Metadata:
    encoder         : Lavc56.1.101 pcm_s16le
Stream mapping:
  Stream #0:1 -> #0:0 (aac (native) -> pcm_s16le (native))
Press [q] to stop, [?] for help
size=4132kB time=00:00:23.98 bitrate=1411.2kbits/s
video:0kB audio:4132kB subtitle:0kB other streams:0kB global headers:0kB muxing
overhead: 0.001843%
C:\ffmpeg\bin>

```

Figure 6-5 Extracting .wave and .yuv from .avi



```
C:\Users\MRUDULA\Documents\faac-1.28\frontend\Debug>faac E:\shrek\shrek.wav -o
E:\shrek\shrek1.aac
Freeware Advanced Audio Coder
FAAC 1.28

Quantization quality: 100
Bandwidth: 16000 Hz
Object type: Low Complexity(MPEG-2) + M/S
Container format: Transport Stream (ADTS)
Encoding E:\shrek\shrek.wav to E:\shrek\shrek1.aac
  frame      | bitrate | elapsed/estim | play/CPU | ETA
1034/1034   | <100%> | 122.0        | 1.9/1.9  | 12.81x | 0.0

C:\Users\MRUDULA\Documents\faac-1.28\frontend\Debug>
```

Figure 6-6 Encoding .wav to .aac low complexity using FAAC

### 6.3 Conclusion

An effective transport stream which carries multiple audio and video streams and also easily decodable with synchronization is implemented in this thesis. Two layers of packetization are used and multiplexing of the packets is implemented. The proposed multiplexing procedure is effective in that the user could start demultiplexing from any TS packet and achieve synchronized playback. The results show that there is no visual synchronization delay. The buffer overflow/underflow problem is taken care of and there is a delay of approximately 13 ms. Video broadcasting applications should be error-free and packet losses should not occur. Hence, both layers of packetization are accompanied by a packet identifier header.

### 6.3 Future research

The multiplexing algorithm in this thesis uses two elementary streams. However, it could be modified to implement multiple streams and also, multiple programs at the

same time. Some robust error correction codes can be integrated to the transport packets, to make them more suitable for applications such as Video conferencing, broadcasting where the packets are prone to be lost.

## APPENDIX A

### Platform

The research was carried out using an Intel Core i5 3317 CPU @ 1.7 GHz machine with Microsoft Windows 8 64bit version running with 4 GB RAM at a speed of 1.7GHz.

APPENDIX B  
List Of Acronyms



AAC: Advanced audio coding  
ADIF: Audio data interchange format  
ADTS: Audio data transport stream  
AFC: Adaptation field control  
ATSC: Advanced television systems committee  
AVC: Advanced Video Coding  
CABAC: Context-based Adaptive Binary Arithmetic Coding  
CAVLC: Context-based Adaptive Variable Length Coding  
CB: Coding Block  
CPU: Central Processing Unit  
CTU: Coding Tree Unit  
CTB: Coding Tree Block  
CU: Coding Unit  
DCT: Discrete Cosine Transform  
DTS: Decoding Time Stamp  
DVB: Digital video broadcasting  
DVD: digital video disc or digital versatile disc  
EBU: European broadcasting union  
ES: Elementary stream  
FAAC: Free advanced audio coder  
FAAD: Free advanced audio decoder  
FPS: Frames per second  
GOP: Group of pictures  
HDTV: High definition television.  
HEVC: High Efficiency Video Coding

IDR: Instantaneous decoder refresh

ISO: International Organization for Standardization

ITU: International Telecommunication Union

ITU-T: International Telecommunication Union – Telecommunication Standardization sector

JCT-VC: Joint Collaborative Team on Video Coding

JVT: Joint Video Team

LC: Low Complexity

MC: Motion Compensation

MDCT: Modified discrete cosine transform

MPEG: Moving Picture Experts Group

NAL: Network Adaptation Layer

NALU: Network Abstraction Layer Unit

PB: Prediction Block

PCM: Pulse Code Modulation

PCR: Program Clock Reference

PES: Packetized elementary stream

PID: Packet identifier

PPS: Picture parameter set

PTS: Presentation Time Stamp

PU: Prediction Unit.

PUS: Payload unit start.

RTP/IP: Real Time Transport protocol/Internet protocol

SPS: Sequence parameter set

SSR: Scalable sampling rate

TB: Transform block

TNS: Temporal noise shaping

TS: Transport stream

TU: Transform unit

VCEG: Video Coding Experts Group

VCL: Video coding layer

VPS: Video parameter set

YUV: Luminance and chrominance color components

## References

- [1] G. Sullivan et al, "Overview of the high efficiency video coding (HEVC) standard", IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, n 12, pp. 1649-1668, Dec. 2012.
  
- [2] Multimedia Processing Lab website: <http://www.uta.edu/faculty/Krrao/dip>
  
- [3] MPEG-2 advanced audio coding, AAC. International Standard IS 13818-7, ISO/IEC JTC1/SC29 WG11, 1997.
  
- [4] MPEG: Information technology — generic coding of moving pictures and associated audio information, part 3: Audio .International Standard IS 13818-3, ISO/IEC JTC1/SC29 WG11, 1994.
  
- [5] MPEG: Information technology — generic coding of moving pictures and associated audio information, part 4: Conformance testing .International Standard IS 13818-4, ISO/IEC JTC1/SC29 WG11, 1998.
  
- [6] Information technology—Generic coding of moving pictures and associated audio—Part 1: Systems, ISO/IEC 13818-1:2005, International Telecommunications Union.
  
- [7] MPEG-4: ISO/IEC JTC1/SC29 14496-10: Information technology – Coding of audio-visual objects - Part 10: Advanced Video Coding, ISO/IEC, 2005.

- [8] M. Bosi and M. Goldberg “Introduction to digital audio coding and standards”, Boston: Kluwer Academic Publishers, 2003.
  
- [9] R.Linneman, “Advanced audio coding on FPGA”, BS honors thesis, October 2002, School of Information Technology, Brisbane, Australia.
  
- [10] Y. Kubo et al,” Improved high-quality MPEG-2/4 advanced audio coding encoder”, The Acoustical Society of Japan, 2008.
  
- [11] K. Brandenburg, “MP3 and AAC explained”, AES 17th International Conference, Florence, Italy, September 1999.
  
- [12] J. Nightingale, Q. Wang and C. Grecos, “HEVStream: A framework for streaming and evaluation of High Efficiency Video Coding (HEVC) content in lossprone networks”, IEEE Transactions on Consumer Electronics, vol. 59, pp.404-412, May 2012.
  
- [13] G. Sullivan, P. Topiwala and A. Luthra, “The H.264/AVC video coding standard: overview and introduction to the fidelity range extensions”, SPIE Conference on Applications of Digital Image Processing XXVII, vol. 5558, pp. 53-74, August 2004.
  
- [14] C. Fogg, “Suggested figures for the HEVC specification”, ITUT/ISO/IEC Joint Collaborative Team on Video Coding (JCTVC) document JCTVCJ0292r1, July 2012.

- [15] K.R.Rao, D. Kim and J.J. Hwang," Video coding standards: AVS China, H.264/MPEG-4 Part10, HEVC, VP6, DIRAC and VC-1", Springer, 2014.
  
- [16] I.E.Richardson, "The H.264 advanced video compression standard", 2nd Edition, Wiley, 2010.
  
- [17] ISO/MP4 information: [http://en.wikipedia.org/wiki/MPEG4\\_Part\\_14](http://en.wikipedia.org/wiki/MPEG4_Part_14).
  
- [18] T.Schierl *et al*, "RTP Payload Format for High Efficiency Video Coding", Nokia, February 27, 2012.
  
- [19] HEVC tutorial <http://www.vcodex.com/h265.html>.
  
- [20] G.Sullivan et al , " Standardized Extensions of High Efficiency Video Coding (HEVC) ", IEEE Journal of Selected Topics in Signal Processing, vol. 7, pp. 1001-1016, Dec. 2013.
  
- [21] T. Wiegand et al, "Overview of the H.264/AVC Video Coding Standard," IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, pp. 560-576, July 2003.
  
- [22] MPEG-4: ISO/IEC JTC1/SC29 14496-10: Information technology – Coding Of audio-visual objects - Part 10: Advanced Video Coding, ISO/IEC, 2005.

- [23] J. Herre and H. Purnhagen, "General audio coding," in The MPEG-4 Book (Prentice Hall IMSC Multimedia Series), F. Pereira and T.Ebrahimi, Eds. Englewood Cliffs, NJ: Prentice-Hall, 2002.
- [24] V. Sze, M. Budhagiavi, G.J. Sullivan,"High efficiency video coding : Algorithms and architecture", Springer 2014.
- [25] Website for AC-3:  
<http://www.digitalpreservation.gov/formats/fdd/fdd000209.shtml>
- [26] Basics of video: <http://lea.hamradio.si/~s51kq/V-BAS.HTM>
- [27] The HEVC website: <http://hevc.hhi.fraunhofer.de/>
- [28] HEVC open source software (encoder/decoder):  
[https://hevc.hhi.fraunhofer.de/svn/svn\\_HEVCSoftware/branches/HM14.0dev/](https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/branches/HM14.0dev/)
- [29] JCTVC documents are publicly available at <http://ftp3.itu.ch/avarch/jctvcsite> and <http://phenix.itsudparis.eu/jct/>.
- [30] HEVC software manual:  
[https://hevc.hhi.fraunhofer.de/svn/svn\\_HEVCSoftware/branches/HM-9.2-dev/doc/software-manual.pdf](https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/branches/HM-9.2-dev/doc/software-manual.pdf)

Special issues on HEVC.

- [31] Special issue on emerging research and standards in next generation video coding, IEEE Transactions on Circuits and Systems for Video Technology (CSVT), vol.22, pp. 1646-1909, Dec. 2012.
  
- [32] "Introduction to the issue on video coding: HEVC and beyond", IEEE journal of Selected Topics in Signal Processing, vol.7, pp. 931-1151, Dec. 2013.
  
- [33] D. K. Fibush, "Timing and Synchronization Using MPEG-2 Transport Streams," SMPTE Journal, pp. 395-400, July, 1996.
  
- [34] Z. Cai et.al "A RISC Implementation of MPEG-2 TS Packetization", in the proceedings of IEEE HPC conference, pp 688-691, May 2000.
  
- [35] P.A. Sarginson, "MPEG-2: Overview of systems layer", BBC RD 1996/2.
  
- [36] MPEG 2 TS: <http://www.erg.abdn.ac.uk/future-net/digital-video/mpeg2-trans.html>
  
- [37] VLC software and source code website [www.videolan.org](http://www.videolan.org)
  
- [38] Ffmpeg software and official website  
<http://ffmpeg.mplayerhq.hu/>
  
- [39] "FAAC and FAAD AAC software" [www.audiocoding.com](http://www.audiocoding.com)
  
- [40] DivX player : [www.divx.com](http://www.divx.com)



[41] MKVToolNix GUI preview

[42] T.Ogunfunmi, M. Narasimha, "Principles of speech coding", Boca rattan, FL,  
CRC press, 2010.

### Biographical Information

Mrudula Warriar received her Bachelor of Engineering degree in Instrumentation from Mumbai University, India, in June 2012. She pursued her masters studies at the University of Texas at Arlington from August 2012 and received her M.S degree in Electrical Engineering in December 2014. She was a member of the multimedia processing research group guided by Dr. K. R. Rao. She worked as an intern and is currently employed with Extron Electronics, Raleigh, NC. Her research interests are video and audio processing, digital signal processing.