Friedrich-Alexander-Universität
Erlangen-Nürnberg

Master Thesis

Optimization of transient coding in mixed lossless/lossy coding schemes

Handed in by:	Nima Majidi
Study Program:	Communications and Multimedia Engineering
Matriculation number:	22816885
First supervisor:	M.Sc. Franz Reutelhuber
Second supervisor:	Dipl.-Ing. Markus Schnell
Advisor:	Prof. Dr.-Ing. Bernd Edler
Editing time:	01.04.2024– 30.09.2024

International Audio Laboratories | Am Wolfsmantel 33 | 91058 Erlangen |
`https://www.audiolabs-erlangen.de`

International Audio Laboratories Erlangen
A joint institution of the
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and
the Fraunhofer-Institut für Integrierte Schaltungen IIS.

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, Sep. 2024, Nima Majidi

Abstract

This thesis examines how effective Temporal Noise Shaping (TNS) is in improving audio coding efficiency within both Variable Bit Rate (VBR) and Constant Bit Rate (CBR) frameworks (lossless and near-lossless coding). The study begins with a solid theoretical background on linear prediction for time signals and spectral information. It also provides an overview of the typical structure of perceptual audio coders and explores the principles of TNS, including a brief literature review.

The methodology section focuses on the LC3Plus codec, discussing the LD-MDCT and Spectral Noise Shaping prior to TNS implementation and also Arithmetic Coder. The details of the TNS module's implementation are provided, along with various configurations and software tools used to analyze its performance. Results show that TNS in VBR mode leads to significant compression gains for transient audio frames while maintaining quality in standard frames. Additionally, the adaptive TNS approach in CBR mode reduces quality loss, allowing for near-lossless coding by optimally removing the least significant bits based on predictive gains and energy ratios.

In conclusion, this research offers important insights into how adaptive TNS can enhance audio signal quality and encoding efficiency.

Keywords: Temporal Noise Shaping, audio coding, Variable Bit Rate, Constant Bit Rate, linear prediction, transient coding

Abbreviations

ABP Active Bit Planes

CBR Constant Bit Rate

DFT Discrete Fourier Transform

DPCM Differential Pulse Code Modulation

FIR Finite Impulse Response

GG Global Gain

IntMDCT Integer Modified Discrete Cosine Transform

LC3 Low Complexity Communication Codec

LD-MDCT Low Delay Modified Discrete Cosine Transform

LPC Linear Predictive Coding

LSB Least Significant Bit

MDCT Modified Discrete Cosine Transform

MSE Mean Squared Error

MSB Most Significant Bit

PSD Power Spectral Density

SFM Spectral Flatness Measure

SNR Signal to Noise Ratio

SNS Spectral Noise Shaping

TNS Temporal Noise Shaping

VBR Variable Bit Rate

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The quality of an audio signal, especially music, can be evaluated using different measures. One important aspect of coding is the number of bits used to encode the audio signal. The bitrate of an encoder, which refers to the total number of bits available per frame for encoding, can vary depending on the hardware and environment.

The human auditory system is uniquely designed, with a defined range of frequencies that are audible to humans. Achieving lossless coding is not always possible, especially when dealing with transient sounds. Transient signals are particularly challenging to encode due to their high energy across all regions of the frequency spectrum. The key question is how to effectively code these transient signals in a specific frame (e.g., 10 ms) while managing their high bit consumption. This study focuses on finding a way to encode these signals using fewer bits while maintaining the perceptual quality of the decoded audio. The primary motivation for this research is to save bits during the encoding of transient signals.

1.2 Aim of Study

The general goal of this thesis is to investigate the potential benefits of transient coding methods when applied to lossless or near-lossless scenarios. Transients, in this context, refer to moments in an audio signal that consist of a wide range of frequency components, such as the sound of clapping hands or a gunshot.

The challenge with transient signals, compared to other types of signals, is their high bit demand for encoding and transmission. Because transient signals usually have large energy across the frequency range and significantly more bits are required to encode them losslessly or with minimal loss compared to other signals. In this thesis, transient coding methods aim to encode/decode the transients of an audio signal in a more optimized way in sense of bit consumption. One approach to handling these transients is Temporal Noise Shaping (TNS), which is the focus of this master's thesis. Although TNS was primarily designed for perceptual coding, it can also be used as a pure prediction tool for lossless coding [2].

In this study, we address both lossless, which the input and output of the audio encoder is bit exact, and near-lossless coding, which the bit budget is within $5-10\%$ of what is required for lossless coding for all (or the majority of) frames. For the lossless implementation, the original TNS is applied, while for the lossy or near-lossless implementation, an adaptive version of TNS is used 5.2.3.3, which involves the removal of a certain number of Least

Significant Bits from the spectrum coefficients before TNS coding. We explored potential methods for achieving a seamless switch between pure prediction mode and noise shaping mode, which will be explained in detail in the Chapter 5. In this study, the Temporal Noise Shaping (TNS) module was already implemented. The goal is to integrate TNS into the existing LC3Plus coder, as described in section 3.1, and investigate:

1. Whether there is any benefit in employing TNS for VBR/CBR mode
2. How much the performance of TNS depends on the utilized bitrate.
3. If the performance of TNS in these scenarios can be further improved with additional modifications.

In Chapter 2, the basics of linear prediction in both the time and frequency domains are discussed. A perceptual audio coder is described, and its different components are briefly explained. The fundamentals of temporal noise shaping are introduced, followed by a short literature review on transient coding. Chapter 3 provides a general overview of the project and its various modules. In Chapter 4, the configuration of our investigations and the frameworks used throughout the study are outlined. Finally, Chapter 5 discusses the development and improvements for both lossless coding (Variable Bit Rate) and near-lossless coding (Constant Bit Rate) and Chapter 6 will conclude the master's thesis.

Chapter 2

Theory

In this chapter, several well-known linear predictors in both the time domain and frequency domain will be reviewed. Next, the general structure of a Perceptual Audio Coder will be explained. Finally, the fundamentals of Temporal Noise Shaping will be explored in detail [1] [2].

2.1 Linear Prediction: Time Signals

2.1.1 Correlation in Time Domain and Power Spectral Density

The main concept of linear prediction for a continuous time signal $x(t)$ is to find a second signal, call it $y(t)$, which is the closest possible estimation of $x(t)$ in terms of energy error, also known as Mean Squared Error (MSE). In this context, the estimated signal $y(t)$ is generated as a linear combination of past values of the input signal $x(t)$. We assume that there is a linear correlation between the past and present parts of the signal. This relationship can be captured using the autocorrelation function, Equation 2.1:

$$R_{xx}(t) = \int x(\tau) \cdot x^*(\tau - t)\, d\tau \tag{2.1}$$

In Equation 2.2, the relation between autocorrelation function in time domain and the Power Spectral Density (PSD) in frequency domain is shown:

$$S_{xx}(f) = \mathcal{F}\{R_{xx}(t)\} = \mathcal{F}\left\{\int x(\tau) \cdot x^*(\tau - t)\, d\tau\right\} \tag{2.2}$$

By applying the Fourier transform to the autocorrelation function, the Power Spectral Density (PSD) can be obtained.

2.1.2 Signal Redundancy / Bounds of Prediction Efficiency

Jayant et al. [4] demonstrated that the amount of redundancy recoverable from a signal using a linear predictor, under optimal conditions, can be derived from its Power Spectral Density (PSD) via the inverse of the Spectral Flatness Measure (SFM).

$$\gamma^2 = \frac{\exp\left[\frac{1}{2\pi}\int \log_e S_{xx}(f)\,df\right]}{\frac{1}{2\pi}\int S_{xx}(f)\,df}, \quad 0 \leq \gamma^2 \leq 1 \tag{2.3}$$

The range for the SFM is between 0 and 1. The higher the SFM value is, the more similar the signal is to white noise. For a lower SFM value, the signal becomes more tonal, and therefore predictable.

2.1.3 Prediction Filter / Predictive Coding

The estimated signal, $y(n)$, for discrete-time signals is usually derived from the input signal, $x(n)$, using linear combinations of previous samples. In Equation 2.4, this can be considered as a convolution of the input with an array of prediction coefficients, $a(n)$, similar to Finite Impulse Response (FIR) filtering. To calculate the optimal prediction coefficients from the autocorrelation function of a signal, several well-known methods can be found in [5]. The most popular method for finding these linear predictive coding (LPC) coefficients is the Levinson-Durbin recursion:

$$y(n) = \sum_{i=1}^{N} a(i) \cdot x(n-i) = x(n) * a(n) \tag{2.4}$$

where N shows the number of samples in time domain.

2.2 Linear Prediction: Spectral Information

If $X(f)$ represents the Fourier transform of $x(t)$, we can explore various methods to predict the value of $X(f)$ at a specific frequency based on its neighboring frequency components.

2.2.1 Correlation in Frequency Domain and Squared Hilbert Envelope

To calculate the square of Hilbert envelope $e(t)$ for an real signal $x(t)$, we can use Equation 2.5 [1]:

$$e(t) = \mathcal{F}^{-1}\left\{\int \tilde{X}(\zeta) \cdot \tilde{X}^*(\zeta - f)\,d\zeta\right\} \tag{2.5}$$

For positive frequencies, $\tilde{X}$ shows the single-sided spectrum for $x(t)$. Therefore, there is a relation between the Hilbert envelope and autocorrelation function of spectrum (single-sided)

2.2.2 Signal Redundancy / Bounds of Prediction Efficiency

Based on Equations 2.2 and 2.3, the redundancy within a signal is linked to applying the Flatness Measure function $FM()$ to the Fourier transform of the signal's autocorrelation function:

$$\gamma^2 = FM(S_{xx}(f)) = FM\left(F\{R_{xx}(t)\}\right) \quad 0 \leq \gamma^2 \leq 1 \tag{2.6}$$

Let's now consider $\tilde{X(f)}$ as the input signal of a predictor and apply the previous formula. Then for the predictability, we obtain Equation 2.7 across frequency:

$$\gamma^2 = FM\left(F\left\{\int \tilde{X}(\zeta)\cdot\tilde{X}^*(\zeta - f)\,d\zeta\right\}\right) \tag{2.7}$$

This can be further rewritten using Equation 2.5. By knowing the fact that $F\{F\{x(t)\}\} = x(-t)$ and Flatness Measure function is an even function, we find a dual relation between Equations 2.3 and 2.6.

$$\gamma^2 = FM\left(F\left\{F\{e(t)\}\right\}\right) = FM\left(e(-t)\right) = FM\left(e(t)\right) \tag{2.8}$$

2.2.3 Predictive Filter / Predictive Coding

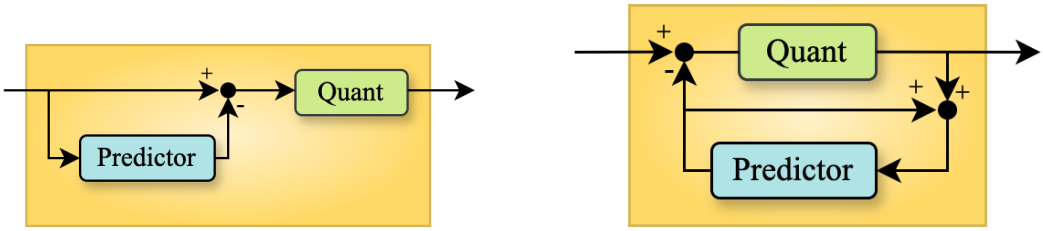
Assume that we have a discrete-time signal, $x(n)$. Predictive coding based on its Discrete Fourier Transform (DFT) is straight forward. Instead of $x(n)$, we use its spectral domain to calculate both optimum prediction coefficients and prediction filter. We can use either open-loop prediction (forward prediction), Figure 2.1a, or closed-loop prediction (backward prediction), Figure 2.1b, for this purpose. The details of these schemes are thoroughly examined in Section 2.3.

Based on the Equations 2.2 and 2.5, the squared Hilbert envelope of a signal $e(t)$ and its power spectral density $S_{xx}(f)$ represent complementary aspects in the time and frequency domains. If we assume that our signal is discrete, the following conclusion can be derived from these equations: If the Hilbert envelope $e(t)$ does not vary for each partial bandpass signal over an interval of frequencies, the autocorrelation between adjacent spectral values will also be unchanged. This implies that the sequence of spectral coefficients is stationary over frequencies, allowing predictive coding techniques to efficiently represent the signal using a single set of prediction coefficients.

2.3 Residual Energy Reduction

As discussed, the PSD and the squared Hilbert envelope are correlated. Due to this relationship, a prediction gain can be obtained, which refers to the potential reduction in the residual signal energy. The prediction gain depends on the "squared-envelope flatness measure" of the signal, which reflects the transient characteristics of the signal. In fact, the greater the "non-flatness" of the squared time envelope, the larger the coding gain for Differential Pulse Code Modulation (DPCM).

To reduce residual energy in transient signals (coding gain), conventional DPCM encoders can be employed [4], as shown in Figure 2.1. Both open-loop and closed-loop encoders have their own advantages. Although both approaches result in the same amount of error energy in the decoded signal, the key difference is that, unlike the closed-loop method, the quantization noise in the open-loop approach is shaped in the time domain and masked by the main signal. Consequently, the open-loop method addresses the problem of temporal masking, Figure 2.3. This approach is known as the "Temporal Noise Shaping" (TNS) algorithm. [2]



(a) Open-loop DPCM Encoder (b) Closed-loop DPCM Encoder

Figure 2.1: Forward prediction and backward prediction

2.4 A Typical Perceptual Audio Coder Structure

Now, we would like to outline the basic structure of a perceptual audio encoder and decoder for monophonic signals [1]:

- In the encoder side, Figure 2.2:
  - The input samples are transformed into a subsampled spectral representation using different filterbanks and transforms, for example: Modified Discrete Cosine Transform (MDCT) [6], polyphase filterbanks [7], or hybrid structures [8, 9].
  - A perceptual model is then applied to estimate the time-dependent masking threshold for the signal. This threshold indicates the maximum coding error that we accept while still preserving the perceived quality of the audio.
  - The spectral values are then quantised and encoded based on the precision dictated by the estimated masking threshold, ensuring that the quantisation error is masked by the transmitted signal and remains inaudible after decoding.
  - Finally, all necessary information, including the quantised/encoded spectral values and additional side information, is assembled into a bitstream and sent to the decoder.
- In the decoder side, we reverse this process, Figure 2.2:
  - The bitstream is decoded and separated into coded spectral data and side information.
  - Then we apply the inverse quantisation to the spectral values that are previously quantised.
  - Lastly, we use a synthesis filterbank to convert the spectral values back to a time-domain representation.

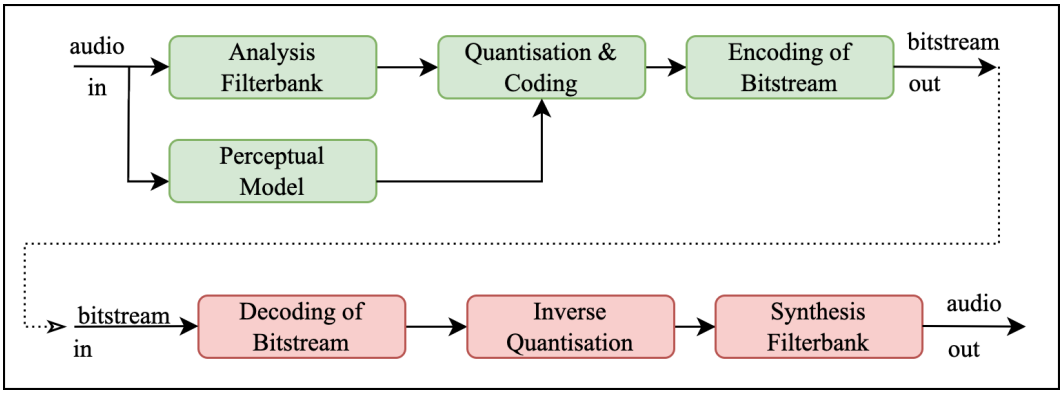


Figure 2.2: Typical Perceptual Audio Coder Structure

By using this general coding framework, it becomes possible to efficiently take benefit of the irrelevancies present in each signal because of the limitations of human hearing. In particular, the masking effect [10] [11] can be leveraged by shaping the quantisation error spectrum to align with the signal's masking threshold [12]. This method effectively conceals the noise beneath the encoded signal, maintaining a perceptually transparent quality. One practical example of such a perceptual coder is LC3Plus [3] which is the base for our coding system.

2.5 Fundamentals of Temporal Noise Shaping

Before discussing TNS, it is important to understand the problem with transients. Assume we have a signal in the time domain and apply the Modified Discrete Cosine Transform (MDCT) [13] to convert it into the frequency domain. After the quantization step, there will be a mismatch between the original signal and the quantized signal, referred to as quantization error. Once the synthesis filter is applied at the decoding side, this quantization error spreads across the time domain.

The main problem arises with transient signals. Transient signals consist of a wide range of frequencies, exhibiting high energy across the spectrum but only for a short period of time. The quantization noise is scaled according to this high energy and spreads across the entire window. As a result, when the signal is reconstructed on the decoding side, artifacts may be generated before the onset of the transient signal. This effect is known as the "pre-echo phenomenon". A similar "post-echo" effect after transients can also be observed but is generally less critical.

Due to the properties of the human auditory system, post-echoes can be masked by the ear after a sharp transient and will be less perceptible. These pre-echoes can also be masked, but only if the majority of the coding noise does not persist for more than approximately 2 ms before the onset of the transient signal. Otherwise, the pre-echo artifact becomes audible. You can find these masking effects in Figure, 2.3:

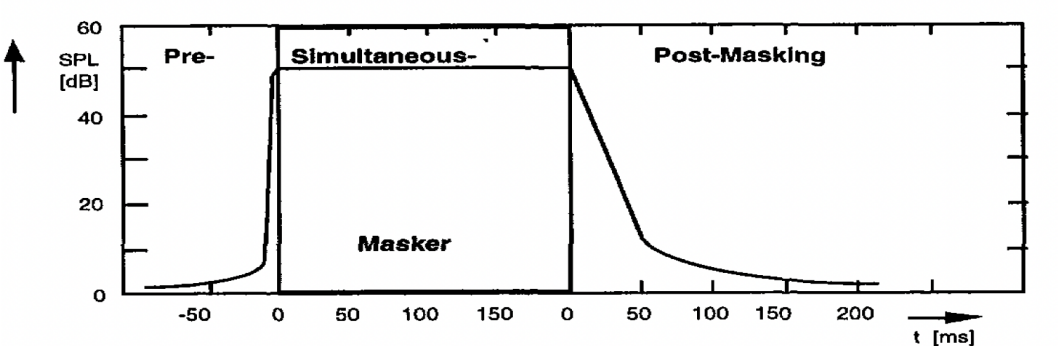
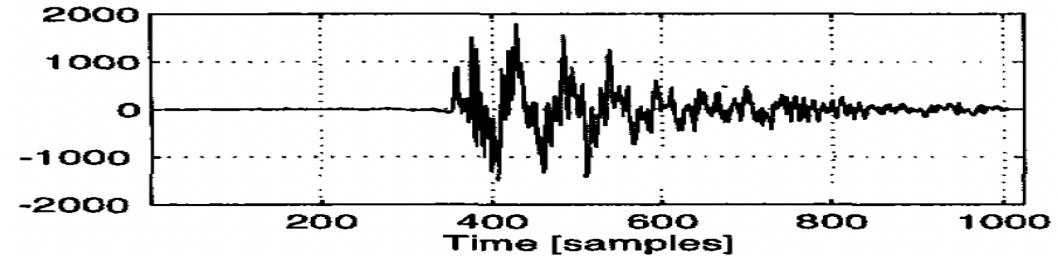


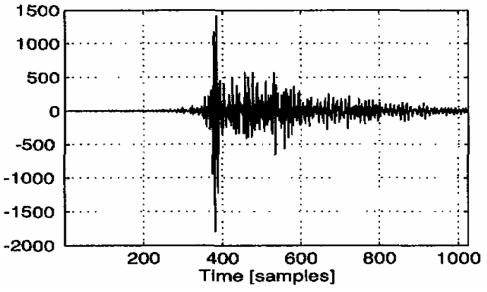
Figure 2.3: Principle of pre-masking and post-masking [1]

Two effective strategies for controlling pre-echo artifacts are MDCT window switching [14] and Temporal Noise Shaping [2]. Temporal Noise Shaping (TNS) is based on the duality of the LPC paradigm, meaning that instead of predicting in the time domain, prediction is done in the frequency domain, and instead of shaping noise in the frequency domain, it is shaped in the time domain. Measurements show that this approach provides significant improvements in the audio quality of the ISO/MPEG-AAC coding algorithm [1]. With TNS, the quantization noise is shaped to resemble the original signal.

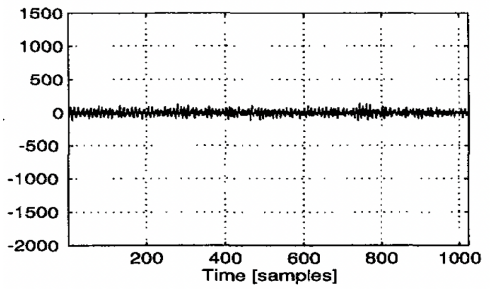
In Figure 2.4, the castanets sound is used as the input to the encoder. TNS is applied to this signal using 1024 Discrete Cosine Transform coefficients. The error signal, shown on the y-axis, is illustrated in the next subplots (b and c) for both TNS and No-TNS scenarios. As expected, the temporal shape of the coding error in the TNS scenario closely follows the envelope of the original signal, whereas in the No-TNS scenario, the quantization error is spread almost uniformly across the time axis [2]. In the figure, the x-axis represents time samples. The y-axis represents amplitude for subplot (a), and signal error for subplots (b) and (c).



(a) Castanets - uncoded signal



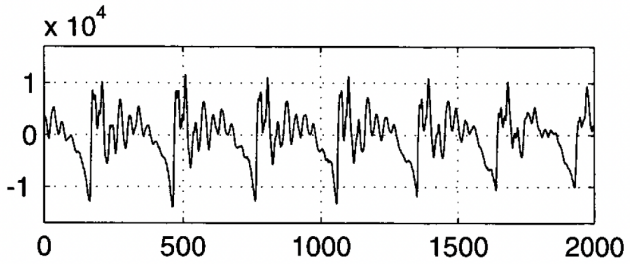
(b) With TNS



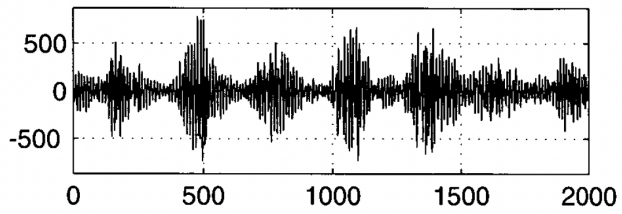
(c) No-TNS

Figure 2.4: An example of quantization error shaping with TNS [2]

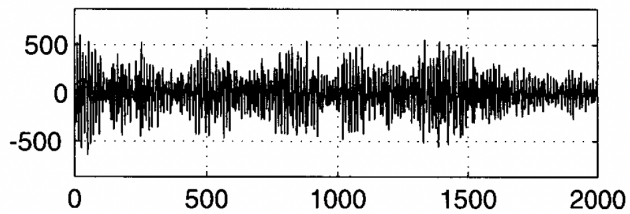
With this noise shaping, we can ensure that the noise is masked by the decoded signal. Another example is "German Male Speech", which has a distinct pitched structure [2], as shown in Figure 2.5 (a). For this example, an MDCT filter bank with 1024 coefficients is used. With TNS applied, the coding error is more concentrated around the glottal pulse (b). In contrast, when No-TNS is applied, a strong degree of noise shaping is not observed, making it more likely for artifacts to occur (c) [1]. For this figure, the x-axis and y-axis are defined the same way as in the previous figure.



(a) Uncoded speech



(b) With TNS



(c) Without TNS

Figure 2.5: An example of error shaping with TNS [1]

Transient parts of audio contain a large number of high-frequency components, which are less audible to the human auditory system. By using TNS, fewer bits are allocated to these parts, minimizing the perceptual impact of quantization noise. In other words, transient segments receive fewer bits during quantization, allowing more bits to be allocated to the more audible portions of the audio signal.

2.6 Literature Review

Based on what Geiger et al. [15] discussed, TNS can be used as a pure predictor. This work introduces an enhancement to MPEG-4 AAC that supports both scalable lossy and lossless audio coding in the frequency domain. The proposed system uses the Integer Modified Discrete Cosine Transform (IntMDCT) [16] [17] to achieve perfect reconstruction and extends AAC with a lossless mode while maintaining compatibility with the lossy AAC core. By combining lossy and lossless coding streams efficiently, the system reduces the overall bit rate compared to simultaneous transmission of an AAC bitstream and a lossless-only

coded bitstream. TNS modifies the MDCT spectrum before quantization by applying linear prediction filters, which can help improve the accuracy of transient audio signals. In this system, TNS is implemented in a lossless fashion by ensuring that all prediction steps are rounded to integer values. This ensures that even when TNS is applied, the IntMDCT spectrum can be bit exact and perfectly reconstructed during the decoding process. By using a closed-loop prediction, TNS provides efficient redundancy reduction for transient signals, as described in [2], crucial for maintaining audio fidelity during rapid signal changes. TNS thus plays a significant role in enhancing the system's handling of transient audio, reducing noise artifacts, and improving overall sound quality in both lossy and lossless modes.

Another paper introduces a novel approach called companding in audio coding, which involves QMF domain pre- and post-processing to shape temporally the coding noise [18]. In fact, they reduced the dynamic range of the signal locally in a QMF time windows and at the post decoding step, this dynamic range reduction will be restored again. One benefit of this approach is to send low side information, 1 or 2 bits. In compare to TNS, this method also used to improve the quality of very transient audio signals like applause and at the same time, achieving similar performance on speech signals. Due to the lack of time, investigation of this approach was beyond the scope of the thesis.

Chapter 3

Methodology

This chapter is primarily based on the [3] documentation. For further details, please refer to that document.

3.1 LC3Plus Description

The project the author used as the basis is called LC3plus, the Low Complexity Communication Codec [3]. It is an audio codec compliant with ETSI (European Telecommunications Standards Institute) standards. The general workflow for this codec is shown in Figure 3.1. As illustrated, a Low Delay Modified Discrete Cosine Transform (LD-MDCT) is applied to each frame. The frequency components are then processed by the Spectral Noise Shaping (SNS) module to reduce perceived spectral quantization errors, followed by the Temporal Noise Shaping (TNS) module, which reduces perceived temporal quantization errors. The spectral quantizer module quantizes the components shaped by SNS and TNS. Finally, the spectral coefficients and all side information are encoded using entropy coding and multiplexed into the bitstream.

LC3plus includes a high-resolution coding mode that utilizes Full-Band (FB) and Ultra-Band (UB) audio encoding at sampling rates of 48 kHz and 96 kHz. This high-resolution mode supports all frame durations and a wide range of bitrates. It is designed for higher bitrates, improved Signal-to-Noise Ratios (SNR), and the ability to encode the full audio spectrum up to the Nyquist frequency. The supported configurations are detailed in Table 5.2 of [3].

In this study, for lossless coding, the MDCT is replaced by an integer MDCT as described in [17]. To implement the MDCT, Givens rotations [19] were used. By applying the Lifting scheme [20], Givens rotations can be estimated and decomposed. The IntMDCT is an integer approximation of the MDCT that enables lossless reconstruction and lossless coding. Additionally, the spectral quantizer was disabled, and only simple rounding was applied for this study. The other modules like Noise Level, LTPF, BW Detect were also turned off for lossless operation.

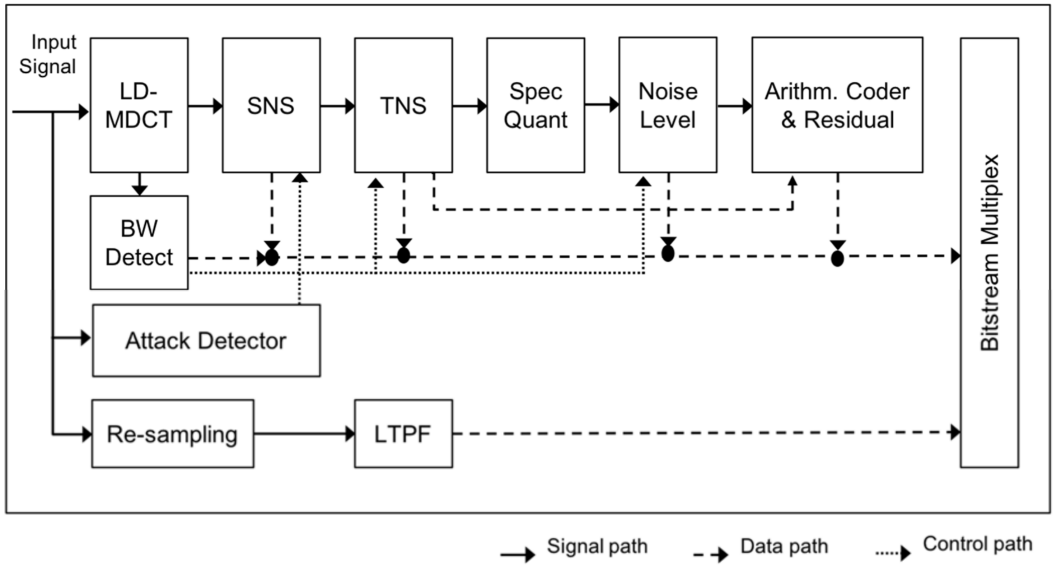


Figure 3.1: LC3Plus high level overview [3]

3.2 LD-MDCT and band-wise energy estimation

The Low Delay MDCT (LD-MDCT) transforms the audio input from time-domain samples into spectral coefficients and corresponding energy values. The spectrum is organized into 64 frequency bands and the energies are estimated on those bands. Figure 3.2 provides an overview of the process.

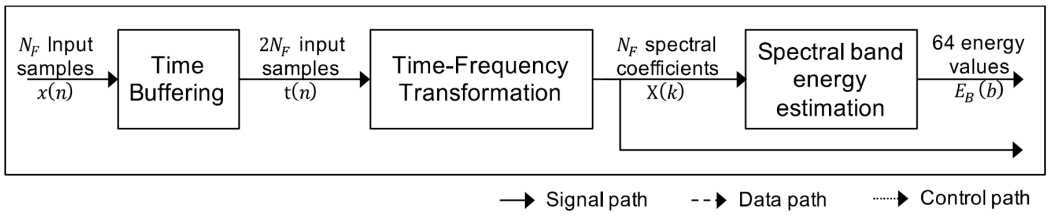


Figure 3.2: Low Delay MDCT overview [3]

In this block, N_F shows the number of input samples per frame and 64 band energies can be obtained by the Equation 3.1:

$$E_B(b) = \sum_{k=I_{f_S}(b)}^{I_{f_S}(b+1)-1} \frac{X(k)^2}{I_{f_S}(b+1) - I_{f_S}(b)} \quad \text{for} \quad b = 0 \dots N_b - 1 \tag{3.1}$$

Where $X(k)$ are representing MDCT coefficients and N_b is the number of bands which is 64 here and $I_{f_S}(b)$ are the band indices that are provided in clause 5.9.1: Band tables index in [3].

3.3 SNS Description

Spectral Noise Shaping (SNS) adjusts the MDCT spectrum by applying a set of scale factors to control the distribution of quantization noise in the frequency domain. These scale factors

are designed to shape the noise so that it is less noticeable to the human ear, thereby improving the perceived quality of the decoded audio.

The SNS encoding process involves four main steps. First, 16 scale factors are estimated based on the 64 band energies. These scale factors are then quantized and encoded. Afterward, they are interpolated to generate a set of 64 scale weights. This scheme of downsampling and interpolation of the SNS scale factors is applied in order to limit the transmitted side information required for SNS. Finally, the MDCT spectrum is shaped using these interpolated weights. Figure 3.3 illustrates these processing steps.

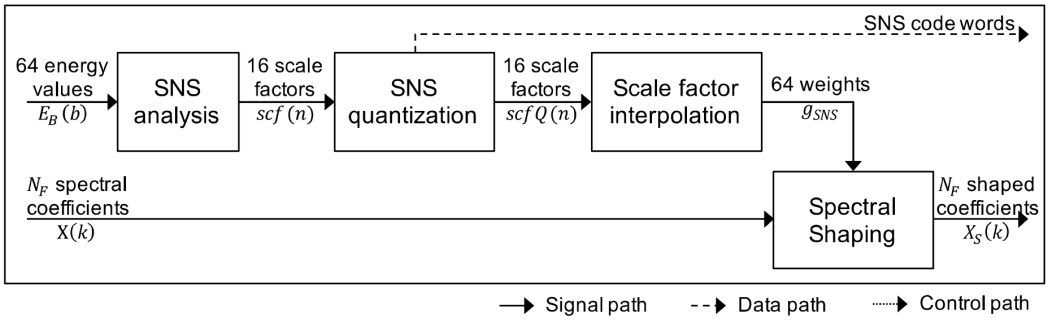


Figure 3.3: SNS overview [3]

It must be noted that the SNS must be disabled for a lossless version, as the spectral shaping is not invertible.

3.4 TNS Description

3.4.1 Structure

To control the temporal quantization noise shape inside a window of modified discrete cosine transform we can use the Temporal Noise Shaping (TNS). In our case, frame size was 10ms and input sampling rate was 48 kHz and according to the standard compliant configuration, we used up to two filters for each frame.

We can see the processing steps for TNS in Figure 3.4 :

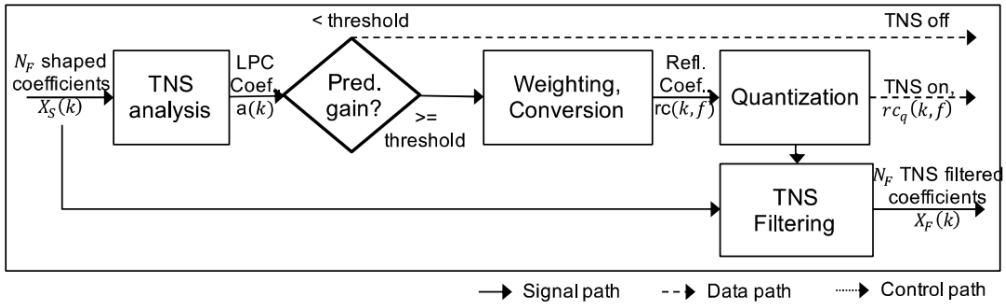


Figure 3.4: Temporal Noise Shaping [3]

3.4.2 TNS analysis

TNS coder will be applied to 480 MDCT coefficient bins. Two filters are considered: a low band filter for coefficients 13 till 200 i.e. 600 Hz - 10 KHz and a high band filter for coefficients 201 till 400 i.e. 10 KHz - 20 KHz. Therefore, each filter in TNS is exposed to some specific coefficients. For each TNS filter, this procedure is repeated: The normalized autocorrelation function will be calculated, for $k = 0 \dots 8$:

$$r(k) = \begin{cases} r_0(k) & , \text{if} \prod_{s=0}^{2} e(s) = 0 \\ \sum_{s=0}^{2} \frac{\sum_{n=\text{sub\_start}(f,s)}^{\text{sub\_stop}(f,s)-1-k} X_s(n) X_s(n+k)}{e(s)} & , \text{otherwise} \end{cases} \tag{3.2}$$

where k in $X_s(n+k)$ shows the distance between two shaped MDCT coefficients, $r(k)$ is the normalized autocorrelation function and

$$r_0(k) = \begin{cases} 3 & , \text{if k} = 0 \\ 0 & , \text{otherwise} \end{cases} \tag{3.3}$$

and

$$e(s) = \sum_{n=\text{sub\_start}(f,s)}^{\text{sub\_stop}(f,s)-1} X_s(n)^2 \tag{3.4}$$

For $s = 0 \dots 2$ where in our configurations the sub_start(f, s) is $\{\{12, 74, 137\}, \{200, 266, 333\}\}$ and the sub_stop(f, s) is $\{\{74, 137, 200\}, \{266, 333, 400\}\}$, 3 subsections per filter that basically just sub-divide the filter regions into equal-sized parts. X_s is the spectrum that is shaped by the SNS. (When the SNS is disabled, the X_s is the same as X).

The normalized autocorrelation function is lag-windowed using:

$$r_w(k) = r(k) \exp\left[-\frac{1}{2}(0.02\pi k)^2\right] \quad \text{for} \quad k = 0 \dots 8 \tag{3.5}$$

The LPC coefficients $a(k), k = 0 \dots 8$ and a prediction error e can be calculated by means of Levinson-Durbin recursion:

$$\begin{aligned} e &= r_w(0) \\ a^0(0) &= 1 \\ &\text{for } k = 1 \text{ to } 8 \text{ do} \\ rc &= -\frac{\sum_{n=0}^{k-1} a^{k-1}(n) \cdot r_w(k-n)}{e} \\ a^k(0) &= 1 \\ &\text{for } n = 1 \text{ to } k-1 \text{ do} \\ a^k(n) &= a^{k-1}(n) + rc \cdot a^{k-1}(k-n) \\ a^k(k) &= rc \\ e &= (1 - rc^2) \cdot e \end{aligned}$$

where $a(k) = a^8(k), k = 0 \dots 8$ are the estimated LPC coefficients and e is the prediction error. The TNS filter will be activated and turned on if predGain > predGain threshold. The prediction gain is computed by:

$$\text{predGain} = \frac{r_w(0)}{e} \tag{3.6}$$

The following steps will be followed only if the TNS filter f is turned on.

A weighting factor is obtained by:

$$\gamma = \begin{cases} 1 - (1 - \gamma_{\min}) \dfrac{thresh2 - predGain}{thresh2 - thresh} & \text{, if } tns\_lpc\_weighting = 1 \text{ and } predGain < thresh2 \\ 1 & \text{, otherwise} \end{cases} \tag{3.7}$$

Assuming that thresh2 = 2, $\gamma_{\min} = 0.85$, and

$$tns\_lpc\_weighting = \begin{cases} 1 & \text{, if } nbits < 480 \\ 0 & \text{, otherwise} \end{cases} \tag{3.8}$$

In our configuration nbits, the number of MDCT coefficients were 480 and therefore $tns\_lpc\_weighting$, is not active. Factor γ is used to weight LPC coefficients:

$$a_w(k) = \gamma^k a(k) \quad \text{for } k = 0 \dots 8 \tag{3.9}$$

and then TNS coder approximates for each TNS filter a set of reflection coefficients by converting the weighted LPC coefficients using this algorithm:

$$\begin{aligned} &a^k(k) = a_w(k), \quad k = 0, \dots, 8 \\ &\text{for } k = 8 \text{ to } 1 \text{ do} \\ &\quad rc(k-1) = a^k(k) \\ &\quad e = (1 - rc(k-1)^2) \\ &\text{for } n = 1 \text{ to } k-1 \text{ do} \\ &\quad a^{k-1}(n) = \frac{a^k(n) - rc(k-1)a^k(k-n)}{e} \end{aligned}$$

where $rc(k, f) = rc(k), k = 0 \dots 7$ are the final estimated reflection coefficients for TNS filter f. If the TNS filter f is off, we set all the reflection coefficients to zero. Finally, an quantization step is applied to these reflection coefficients and the MDCT-spectrum will be filtered by the quantized reflection coefficients using the following algorithm:

for $k = 0$ to $(N_E - 1)$ do
  $X_f(k) = X_s(k)$
$st^0 = st^1 = \ldots = st^7 = 0$
for $f = 0$ to $(num\_tns\_filters - 1)$ do
  if $(rc_{order}(f) > 0)$
    for $n = start\_freq(f)$ to $stop\_freq(f) - 1$ do
      $t = X_s(n)$
      $st^{save} = t$
      for $k = 0$ to $(rc_{order}(f) - 1)$ do
        $st^{tmp} = rc_q(k, f) \cdot t + st^k$
        $t = t + rc_q(k, f) \cdot st^k$
        $st^k = st^{save}$
        $st^{save} = st^{tmp}$
      $X_f(n) = t^8(n)$

where $X_f(n)$ represents the MDCT spectrum after applying the TNS filter. The initial condition for $s^k(n-1)$ for the first TNS filter ($f = 0$) is set to 0, while for the second TNS filter ($f = 1$), it is inherited from the first TNS filter ($f = 0$).

3.5 Arithmetic Coder Description

The spectral coefficients X will be encoded noislessly. The encoding begins with the lowest-frequency coefficient and moves to the highest-frequency coefficient. Coefficients are grouped into pairs, referred to as 2-tuples $\{a, b\}$. Each 2-tuple $\{a, b\}$ is divided into three components: the most significant bit (MSB), the least significant bit (LSB), and the sign. The sign is encoded separately from the magnitude using a uniform probability distribution. It's important to note that a and b may have different signs, and signs are only encoded for non-zero values of a and b.

The magnitude is further split into two sections. The two MSBs of the 2-tuple $\{a, b\}$ are combined and encoded using an arithmetic encoder (you can find an illustrative example in Appendix 2. for arithmetic encoder.). The remaining LSBs, if the bit budget allows, are encoded individually with a uniform probability distribution. For 2-tuples where the magnitude of either coefficient exceeds 3, one or more escape symbols are sent initially to signal any additional bit planes.

Figure 3.5 illustrates the relationship between a 2-tuple, the individual spectral values a and b, the MSBs m, and the remaining LSBs r. In this example, three escape symbols are transmitted before the actual value m, indicating that three LSBs have been sent.

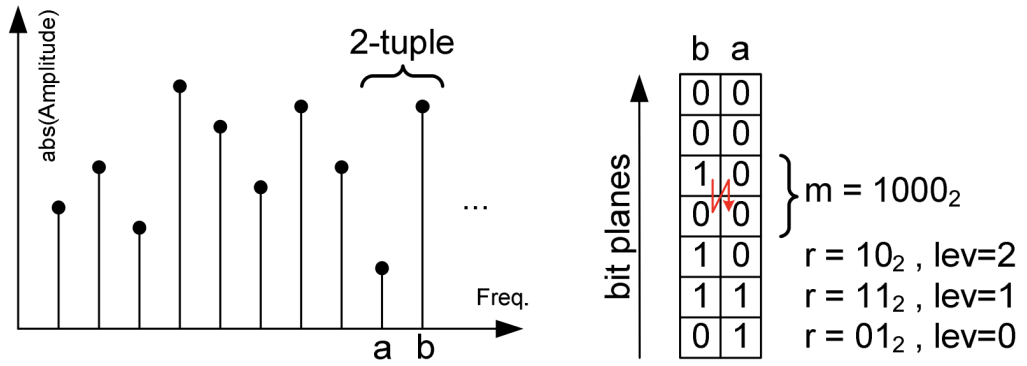


Figure 3.5: Encoded spectral values as tuple {a,b} and their representation as m and r [3]

3.6 Bitstream Encoding

The bitstream of an encoded audio frame is composed of four sections:

1. Initial side information
2. A dynamically coded block using arithmetic coding
3. A dynamic block containing the signs and least significant bits of the encoded spectrum
4. Residual data

Chapter 4

Material

4.1 Configurations

The sampling frequency in our study was 48 kHz, and all audio items had a 16-bit resolution. The frame size was 10 ms, resulting in 480 frequency coefficients after the Modified Discrete Cosine Transform. The prediction gain threshold was 1.5 in TNS module and two groups of audio items were used in this study: transient items and normal items. Transient items:

- glock___mono.wav (Glockenspiel accompanied by timpani/drums)
- Fatboy___mono.wav (Synthesized speech)
- Sample16_orig___mono.wav (Flamenco music)
- si02_44s___mono.wav (Castanet)
- Hotel_Trust_short___mono.wav (Applause)

Normal items:

- broadway_48_16___mono.wav (Saxophone instrument)
- flute_48_16___mono.wav (Flute instrument)
- avemaria_48_16___mono.wav (Opera music)
- cymbal_48_16___mono.wav (Cymbal instrument)
- clarinet_48_16___mono.wav (Clarinet instrument)
- etude_48_16___mono.wav (Classical music)
- blackandtan_48_16___mono.wav (Jazz music)
- waltz_48_16___mono.wav (Classical music)

Item thetest48.wav used as a test item with a concatenation of different signal types including transients.

4.2 Software

The TNS encoder and decoder were implemented using a fixed-point algorithm from the LC3Plus reference software in the C programming language to enable a true lossless coding mode. The fixed-point implementation was necessary to avoid rounding errors that could result in lossy coding. We used the Mex Interface to compile the existing C source code files and convert them into MEX (MATLAB Executable) files. These MEX files are directly executable within MATLAB and offer significant speed improvements for computationally intensive tasks. To generate the Signal-to-Noise Ratio (SNR) figures in Chapter 5.2.3.4, Python and the following libraries were used: matplotlib, pandas, numpy, and scipy.io. The author also used draw.io [21] to create some block diagrams.

Chapter 5

Implementation and Results

In this Section, two distinct scenarios are analyzed: one focusing on lossless coding and the other addressing lossy or near-lossless coding. Before examining the details, it is helpful to first illustrate the characteristics of the data in both the time domain and the frequency domain. In the time domain, the audio file *Fatboy___mono.wav* is regarded as a transient item, while *avemaria_48_16___mono.wav* is treated as a normal item, as shown in Figure 5.1.

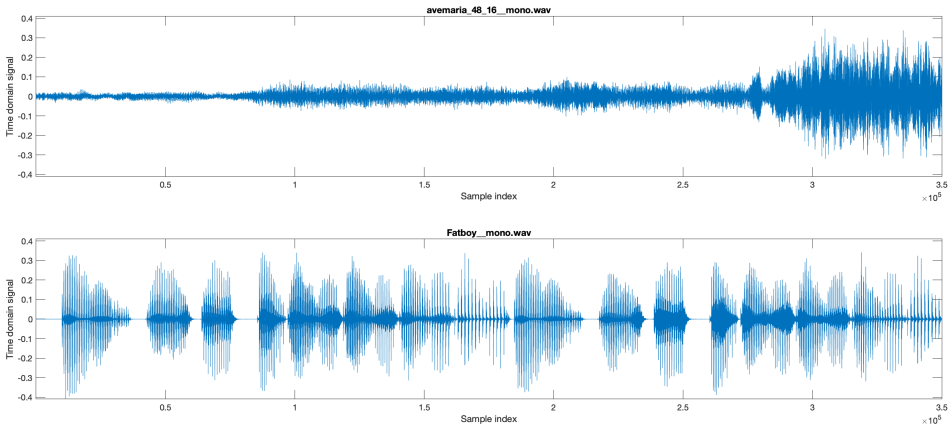


Figure 5.1: Time domain comparison

The transient behavior is clearly visible in the second plot. In the frequency domain, two figures are provided, each representing 1000 frames. Figure 5.2 shows the transient frames from the transient items where TNS is applied. Conversely, Figure 5.3 depicts non-transient frames from normal items, where TNS remains inactive.

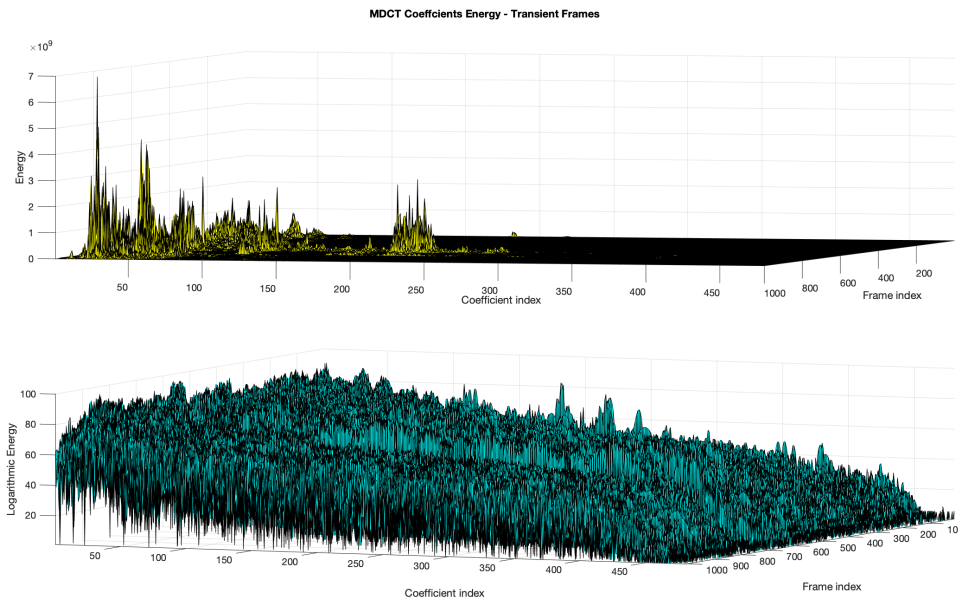


Figure 5.2: Energy spread for transient frames

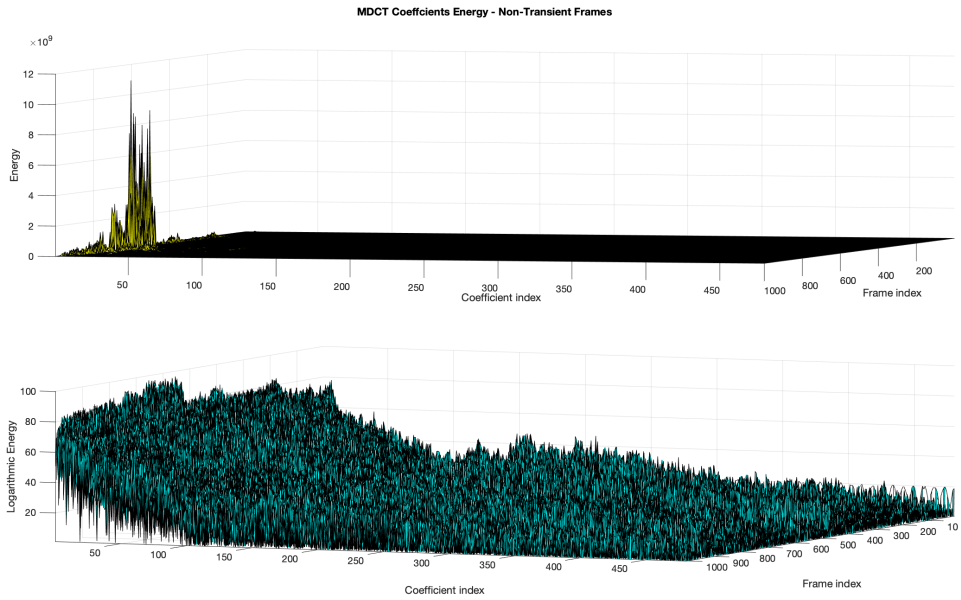


Figure 5.3: Energy spread for non-transient frames

As clearly observed, there is significantly more energy distributed across the frequency range for the transient items. In this chapter, various implementations and developments of TNS will be examined in detail.

5.1 TNS in Variable Bit Rate (Lossless Coding)

5.1.1 First steps

To initiate the investigation of the TNS effect, the compression gain is defined. The size of the encoded frame, represented by the total number of written bits, is taken as a reference, denoted as Numbits0 when TNS is inactive. This is then compared to the scenario where TNS is applied, denoted as Numbits1.

$$\text{Compression Gain} = \frac{\text{Numbits0}}{\text{Numbits1}} \tag{5.1}$$

Figure 5.4 presents a spectrogram of the input signal used to measure bit consumption in Figure 5.5. As expected, a significant amount of energy is distributed across the entire frequency range for the transient portion of the signal, resulting in a spike in bit consumption.

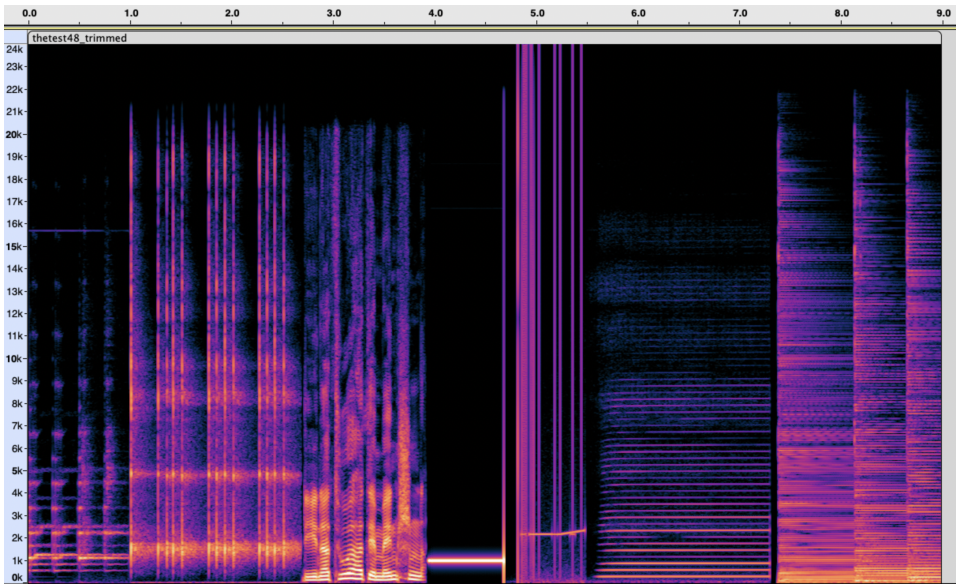


Figure 5.4: Spectrogram - item: thetest48.wav

In Variable Bit Rate (VBR) mode, a TNS block with a fixed-point implementation is incorporated. The coding remains lossless, and as shown in Figure 5.5, there is a substantial improvement in compression gain for transient frames when TNS is applied, compared to the original implementation where TNS is inactive. The figure illustrates that, after TNS activation, bit usage is reduced for these transient segments.

A plot of frame-wise bit consumption is presented. The second plot focuses on frames 480 to 500, which exhibit more transient behavior. As anticipated, there is a substantial reduction in bit usage (approximately 22%) for the transient portion of the audio signal (frames 480 to 500). However, the overall reduction in bit usage is around 2%. These measurements were further extended to include different items, with detailed results provided in Section 5.1.2.

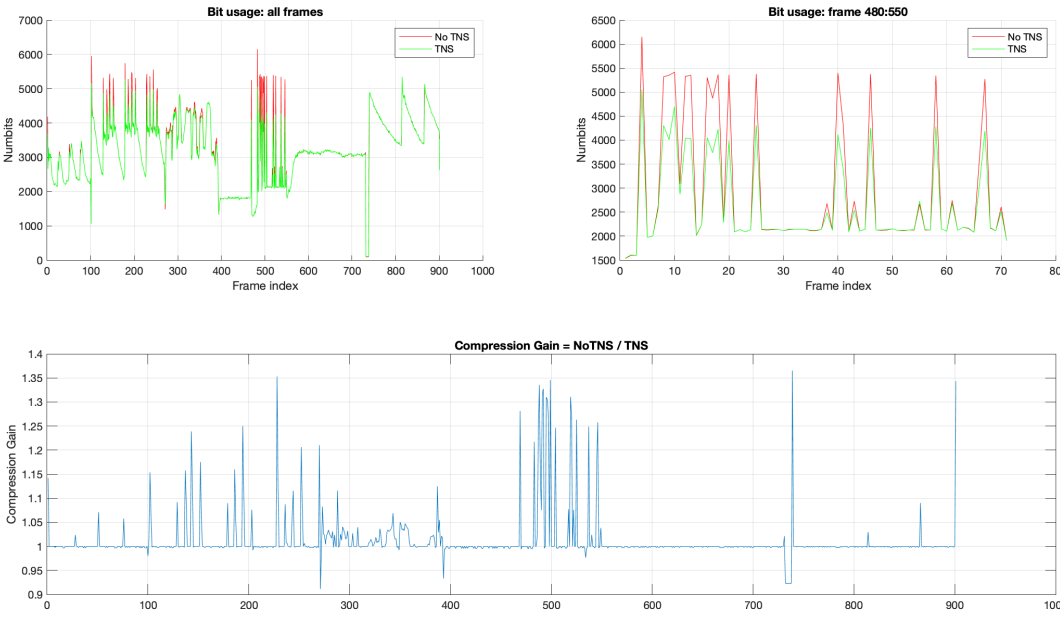


Figure 5.5: Bit usage comparison - item: thetest48.wav

Version 0 of the implementation (V0) applies the original TNS, meaning TNS without any modifications, to both normal and transient items. A minimum prediction gain threshold is defined in the TNS coder. The TNS filter is activated if the prediction gain exceeds this threshold, assigned in 4.1, and all such frames will be referred to as active-TNS frames throughout the text. The impact of this minimum threshold on the average compression gain is also analyzed, with the results presented in various figures in Section 5.1.3.

5.1.2 Compression Gain

An initial investigation was conducted on both transient and normal items. The primary benefit of TNS is observed in the transient items, where the maximum bitrate is significantly reduced. The following Figures, 5.6 and 5.7, measure various criteria:

1. Average Compression Gain over all frames
2. Average Compression Gain over all Active-TNS frames
3. Max Compression Gain among all frames
4. Max bitrate among all frames
5. Compression gain for frame with Max bitrate
6. Percentage of Active-TNS frames

Average Compression Gain is defined in the following equation:

$$\text{Avg. Compression Gain} = \frac{\text{avg. number of bits used for each frame without TNS}}{\text{avg. number of bits used for each frame with TNS}} \tag{5.2}$$

The Max Compression Gain refers to the maximum compression gain among all frames within an item, while Maximum Bitrate denotes the highest number of bits used to encode

a frame in an item.

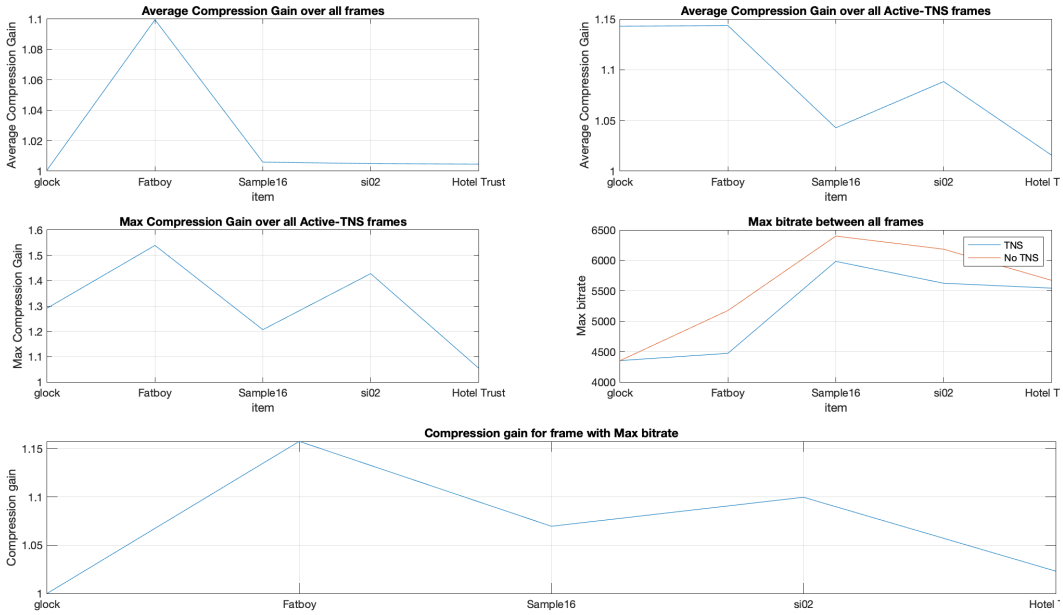


Figure 5.6: Comparison of compression gains for transient items

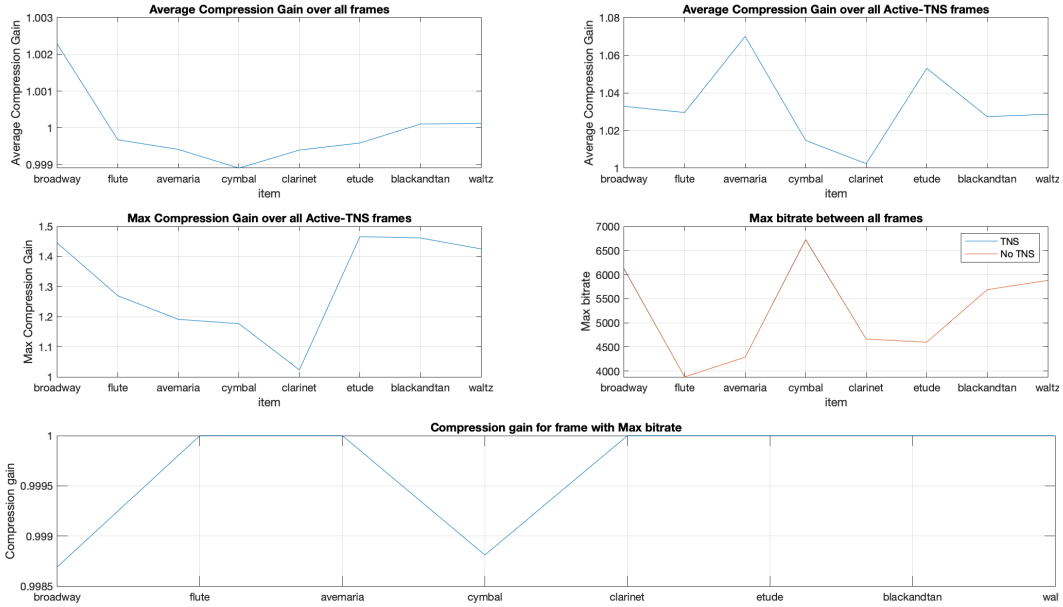


Figure 5.7: Comparison of compression gains for normal items

Based on Figures 5.6 and 5.7, there is a substantial improvement in the average compression gain, as indicated by Equation 5.2, for transient items. For example, there is a 10% increase for the Fatboy item. According to the "Max bitrate across all frames" metric, TNS clearly improves the worst-case scenarios by compressing the most expensive frames more effectively, Figure5.6. As illustrated in Figure 5.8, TNS is activated more frequently for transient items compared to normal items.

On the other hand, the compression gain for normal items does not deteriorate significantly. There is only a minor reduction in compression gain, primarily due to the necessity of sending TNS activation flags for all frames, whereas in the No-TNS scenario, no such side information is required. Figure 5.8 further illustrates why normal items exhibit minimal differences in compression gain: the percentage of TNS-activated frames is very low.

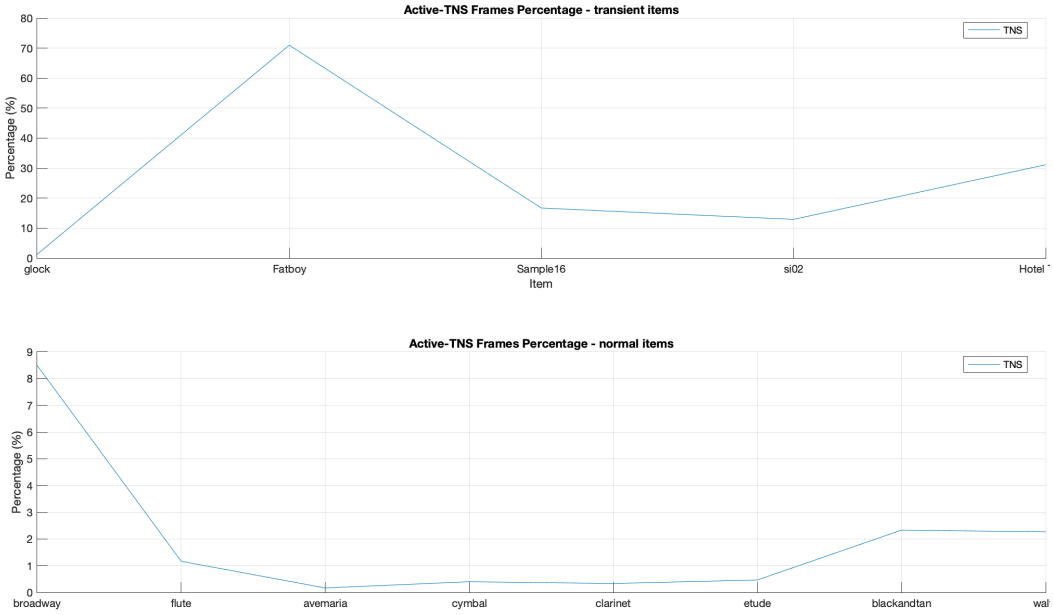


Figure 5.8: Percentage of Active-TNS frames

It is evident that we do not necessarily expect the frame with the maximum bitrate to also exhibit the maximum compression gain. While maximum compression gain serves as a useful statistic for understanding potential savings (in percentage terms), our primary focus is on reducing the maximum bitrate across a significant number of items. For nearly all transient items, the maximum bitrate is reduced, sometimes substantially. This illustrates that TNS is an effective tool for minimizing the maximum bit consumption of transient items.

5.1.3 Minimum Prediction Gain Threshold

The TNS module has a predefined value for the minimum prediction gain threshold. For each frame, the TNS algorithm calculates the achievable prediction gain 3.6 with TNS. This value is then compared to the minimum prediction gain threshold. If the predicted gain exceeds the threshold, the algorithm activates TNS for that specific frame. In this section, different minimum prediction gain thresholds are applied to the TNS module, with the default value set at 1.5. The following Figures, 5.9 and 5.10, illustrate the impact of various minimum prediction gain thresholds on the average compression gain.

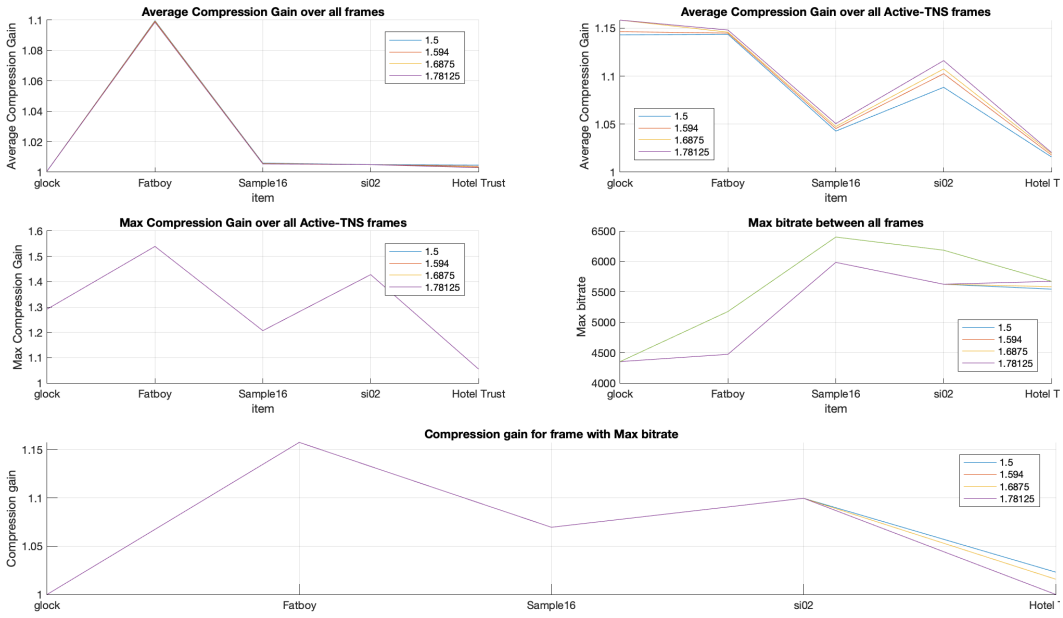


Figure 5.9: Comparison Gain for transient items with different thresholds

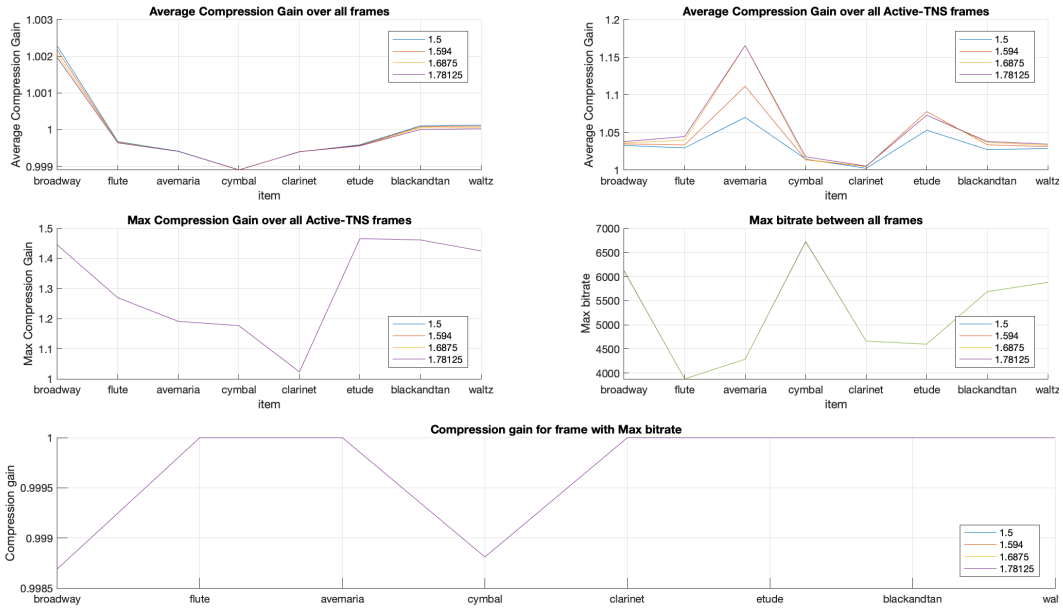


Figure 5.10: Comparison Gain for normal items with different thresholds

Based on the provided figures, it appears that a significant improvement in terms of maximum bitrate across all frames may not be achievable by adjusting this threshold.

5.2 TNS in Constant Bit Rate (Near-Lossless Coding)

When the bitrate is not high enough, the coding becomes lossy or near-lossless. The aim of this section is to handle lossy coding using TNS. Two estimations will be proposed. The first estimation relates to the final bit consumption of the encoder based on the MDCT

coefficient values, and the second estimation determines how many bits will be preserved by activating TNS for a specific frame, based on the two filters in the TNS coder module.

The general workflow of the encoder follows two paths: a lossless path and a lossy path. The encoder performs all necessary computations up to the arithmetic encoder. It then assesses whether the coding can be lossless. If so, the process continues with the arithmetic encoder; if not, it reverts several steps and repeats almost the entire encoding process. In fact, switching between the lossless and lossy paths based on the actual bit consumption would only be possible after the arithmetic coder, and for near-lossless coding, there would be a re-computation for some steps. One significant advantage of these mappings, as shown in equations 5.4, 5.7, and 5.8, is their ability to determine early on whether the coding can be lossless based on the estimated bit consumption for mid bitrates. This early-stage estimation aids in avoiding unnecessary computations in the subsequent modules of the audio encoder.

5.2.1 Spectral values and bit consumption

In this section, two experiments are conducted. In the first scenario, the TNS module is activated for all frames and items, while in the second scenario, the TNS module is deactivated, meaning no TNS block is involved in the encoding process. The goal is to establish a relationship between Active Bit Planes (ABP) and Bit Usage. Active Bit Planes refer to the number of bits occupied by the spectrum. For example, the number 5, represented as 101 in binary, has 3 ABP, and the number 13, represented as 1101, has 4 ABP. The following formula is used to calculate the ABP:

$$ABP = sum(ceil(log2(abs(X) + 1))) - (480 - LastNoneZero) \tag{5.3}$$

Here, LastNonZero represents the last non-zero value in the MDCT spectrum. For example, if LastNonZero = 470, all coefficients from 471 to 480 are zeros and are skipped during the encoding process. A preprocessing step is also performed before applying this formula, where all zero coefficients are converted to 1. This ensures that each zero is counted as 1 ABP. Bit Usage refers to the number of bits consumed by the encoder, i.e., the number of bits required to encode the spectrum for lossless coding in VBR mode.

The idea is to determine whether a mapping between Active Bit Planes and Bit Usage can be established. The challenge is that we only know the actual bit consumption after the arithmetic coder, but retracing the steps to the lossy path is computationally expensive. Therefore, estimating bit consumption as early as possible would be highly beneficial, and this mapping provides a way to do that immediately after the MDCT. Figure 5.11 shows the relationship between Active Bit Planes and Bit Usage as a scatter plot in two scenarios: 1. NoTNS and 2. TNS. As mentioned, both normal and transient items were used to generate these plots.

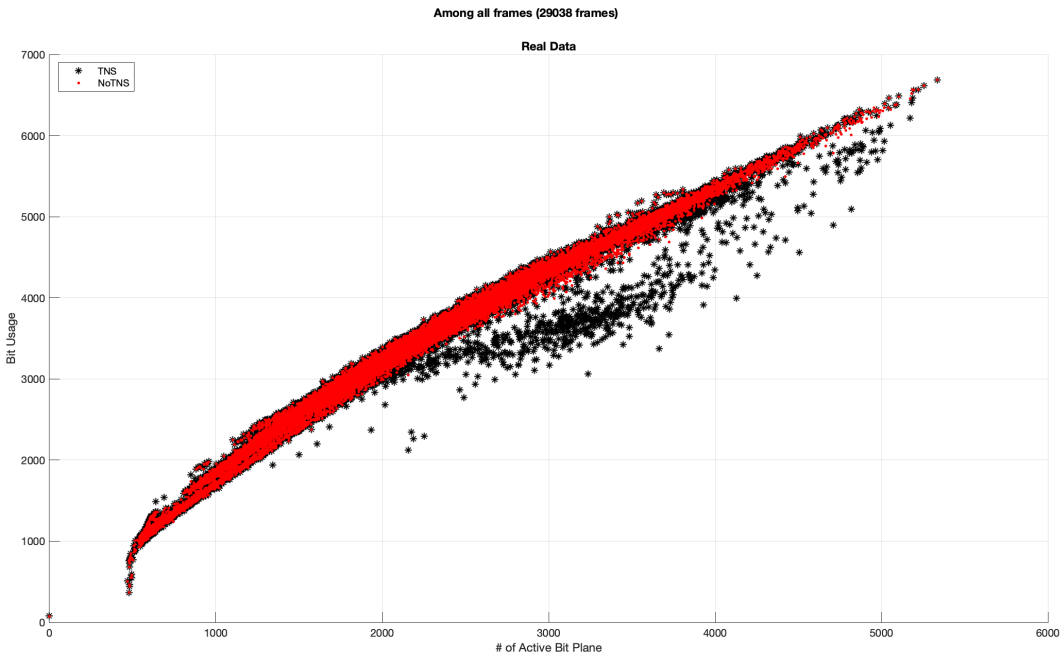


Figure 5.11: Relation between Active Bit Planes and Bit Usage NoTNS/TNS

As expected, TNS does not provide any benefit for certain frames, resulting in minimal bit overhead due to TNS signaling, which cannot be avoided for these frames. The black border at the top of the red curve highlights examples of these specific frames. On the other hand, there are some transient frames where TNS has a significant impact.

As indicated in Equation 5.4, we identified a function that relates the Active Bit Planes to the final Bit Usage of the encoder:

$$\text{Bit Usage estimate} = 3 \times 10^{-8} \cdot \text{ABP}^3 - 3.165 \times 10^{-4} \cdot \text{ABP}^2 + 2.155 \cdot \text{ABP} - 58.306 \quad (5.4)$$

The relationship between Bit Usage and the number of Active Bit Planes (ABP) is modeled by a cubic polynomial function. As illustrated in Figure 5.12, the TNS module is not active, and both normal and transient items are considered to establish this mapping. Using this function allows us to avoid recalculating certain steps during the encoding process, making the process more efficient. However, as seen in the scatter plot in Figure 5.11, TNS complicates finding an accurate estimate. Therefore, the following estimate is based solely on the no-TNS scenario, and further investigations are needed to properly account for TNS, as discussed in the subsequent sections.

The main benefit of this function approximation is in Constant Bit Rate (CBR) mode. At the beginning of encoding in CBR mode, we can easily estimate Bit Usage, whether TNS is active or not. If the estimated Bit Usage fits within the total available bits in the encoding buffer, we proceed with lossless coding. Otherwise, an alternative approach is selected. This approach, which will be explained in Section 5.2.3.3, highlights another advantage of using such an estimation in TNS coding.

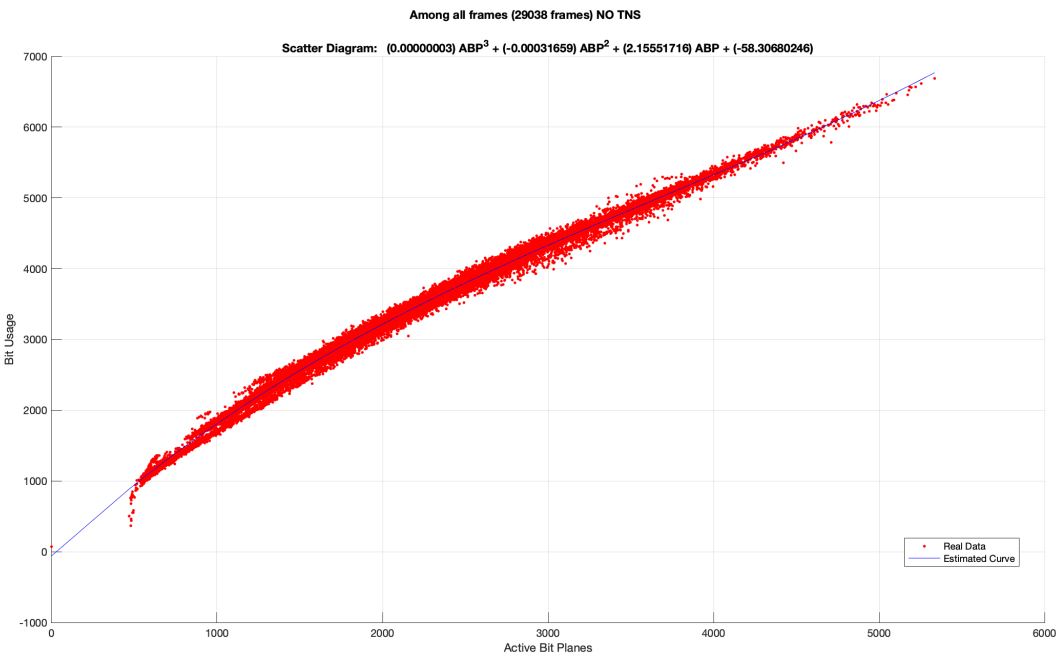


Figure 5.12: Relation between Active Bit Planes and Bit Usage

5.2.2 TNS - Fixed LSB Removal

5.2.2.1 MSE for LSB removal

In Constant Bit Rate mode, when the bitrate is not sufficiently high, the encoding will be lossy. Therefore, the effect of lossy coding in the TNS module must be investigated. We need to analyze the implications of the input to the TNS coder on the encoding side not matching the input to the TNS decoder on the decoding side. This situation simulates a scenario where the signal cannot be reconstructed perfectly, and the effects of removing 1, 2, and 3 LSBs from the spectrum coefficients before the TNS decoder are examined. Removing 1 or 2 LSBs refers to shifting the entire spectrum down by 1 or 2 LSBs.

In other words, a loss due to bitrate constraints is simulated by removing some bits on the decoder side. It appears that even a small input deviation from the original signal prior to TNS results in a significantly larger output deviation after TNS. In Figure 5.13, d-TNS-dec represents the input to the TNS decoder on the decoding side, while d-dec is the output of the TNS decoder. These three frames are selected from the Castanets item, ensuring that TNS is active for them. The Mean Squared Error (MSE) is calculated for these two plots. For brevity, we will refer to the original MDCT coefficients without LSB removal as the MDCT coefficients.

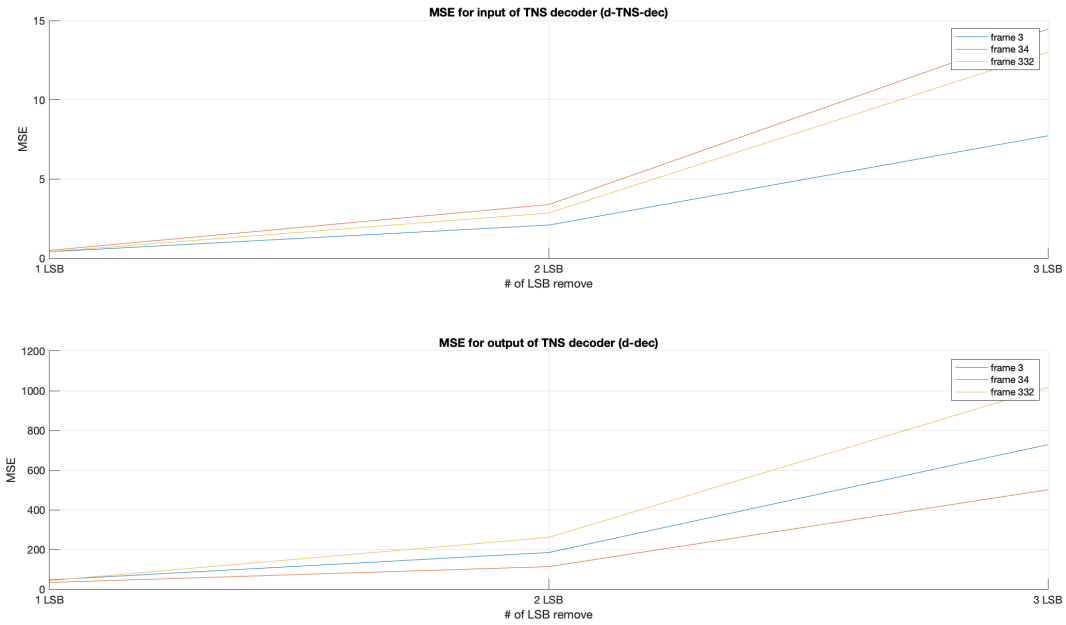


Figure 5.13: LSB removal effect on TNS

In the first plot of Figure 5.13, the Mean Squared Error (MSE) represents the difference between the MDCT coefficients and the input to the TNS decoder, which is the truncated spectrum resulting from LSB removal. In the second plot, the MSE represents the difference between the MDCT coefficients and the output of the TNS decoder, which is again the truncated spectrum due to LSB removal. Therefore, it is evident that as the number of LSBs removed increases, the MSE for the input to the TNS decoder also increases exponentially, leading to a greater error in the output of the TNS decoder. This figure clearly demonstrates that lossy input to the TNS presents a significant problem for near-lossless systems where maintaining most MSBs is the primary goal.

Assuming the encoding is in Constant Bit Rate (CBR) mode and there are insufficient bits to encode the entire spectrum in a lossless manner, one potential improvement could involve performing LSB removal from the spectrum before passing it to the TNS coder. This approach aims to maintain lossless or nearly lossless encoding up to the TNS decoding stage. While we may not have enough bits to perform residual coding for the removed LSBs, we can ensure that the data remains almost lossless between the TNS coder and the TNS decoder, allowing sufficient bits to encode the entire truncated spectrum.

By implementing this strategy, we can mitigate the deviation of the input signal at the TNS decoder from the input signal of the TNS coder, which would otherwise lead to significantly larger errors on the decoding side. For clarity, it should be noted that the LSB removal (truncating the input of the TNS encoder) discussed here is entirely different from the loss simulation (removing LSBs before the TNS decoder) illustrated in Figure 5.13.

5.2.2.2 Prediction Gain in TNS

Before considering spectrum truncation, it is essential to ensure that even after truncation, the prediction gains remain nearly similar to those of the original spectrum. Let's assume that the input to the TNS coder is the MDCT spectrum, from which a certain number of LSBs are removed and fed back into the TNS coder. We will then observe its effect on the prediction gain. It is anticipated that the prediction gains will remain nearly the same since

the information in the spectrum is not significantly altered.

In Figure 5.14, frame 7, which is an active TNS frame, is selected from the Castanets item. As illustrated, for the original TNS and for 1, 2, and 3 LSB removals, the prediction gains are 19.1282, 19.1244, 19.1872, and 19.0182, respectively. Although this observation is based solely on an active frame, it supports the expected outcome.

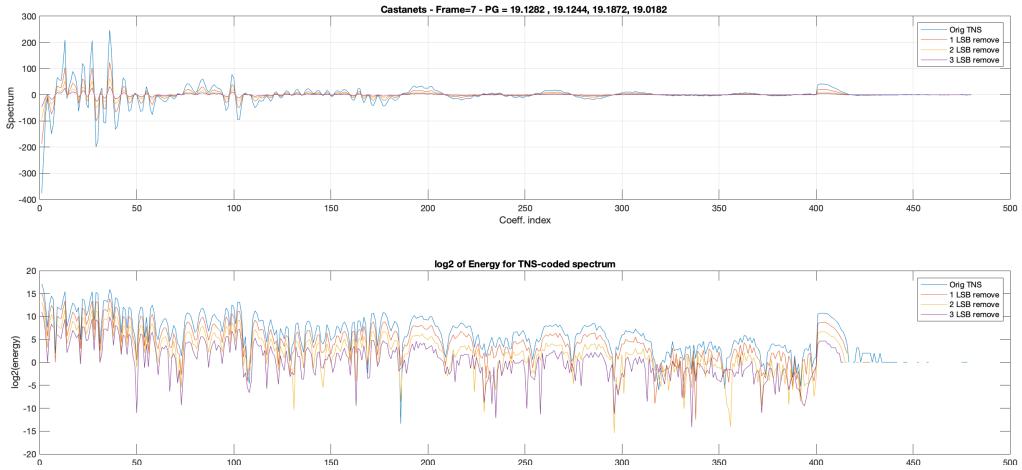


Figure 5.14: Energy Comparison for LSB removal

As observed, the removal of LSBs results in a decrease in the energy of the spectrum, and the MDCT spectrum is also shifted downward.

5.2.2.3 Version 1 of TNS development

In this section, we develop a more advanced TNS implementation step by step. The implementation of TNS version 1 (V1) involves removing LSBs prior to the TNS coder (referred to as the LSB remover) and subsequently adding those LSBs back after the TNS coder (referred to as the LSB adder), following the same procedure for the TNS decoder.

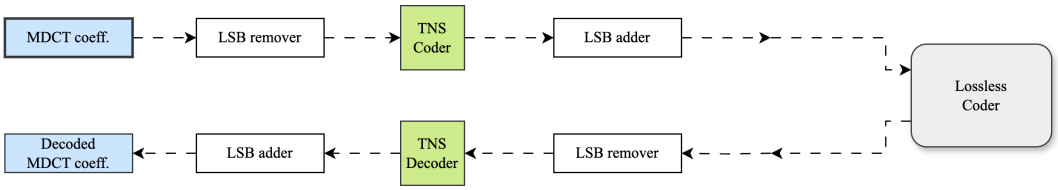


Figure 5.15: Architecture for V1 - Signs not considered

In the current development of TNS Version 1, as illustrated in Figure 5.15, lossless coding cannot be achieved. This limitation arises because, following spectrum truncation, some coefficients may become zero. To reintroduce the removed LSBs after TNS coding, it is essential to know the original sign of the coefficients. Without this information, it remains ambiguous whether a positive or negative value should be added to each coefficient. Consider the following example:

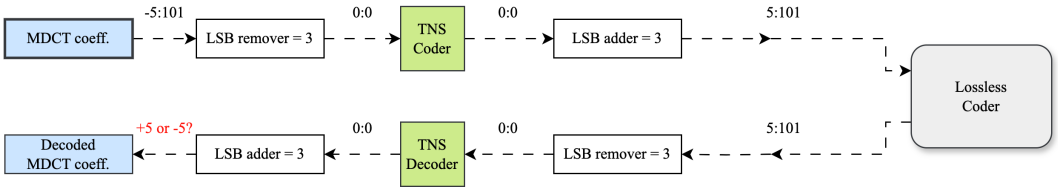


Figure 5.16: Version 1 - when signs not transmitted

As illustrated in Figure 5.16, it is unclear whether the constant amount to be added should be a positive or negative value in the last step. In fact, addressing the separation between positive and negative signs is crucial for achieving lossless coding.

One might question why it is not feasible, as shown in Figure 5.16, to determine the sign of the MDCT coefficient based on the value prior to LSB removal on the decoder side. Thus, we must consider the sign-flipping behavior of the TNS module. Consequently, we will investigate whether the signs of all coefficients are preserved before and after passing through the TNS coder.

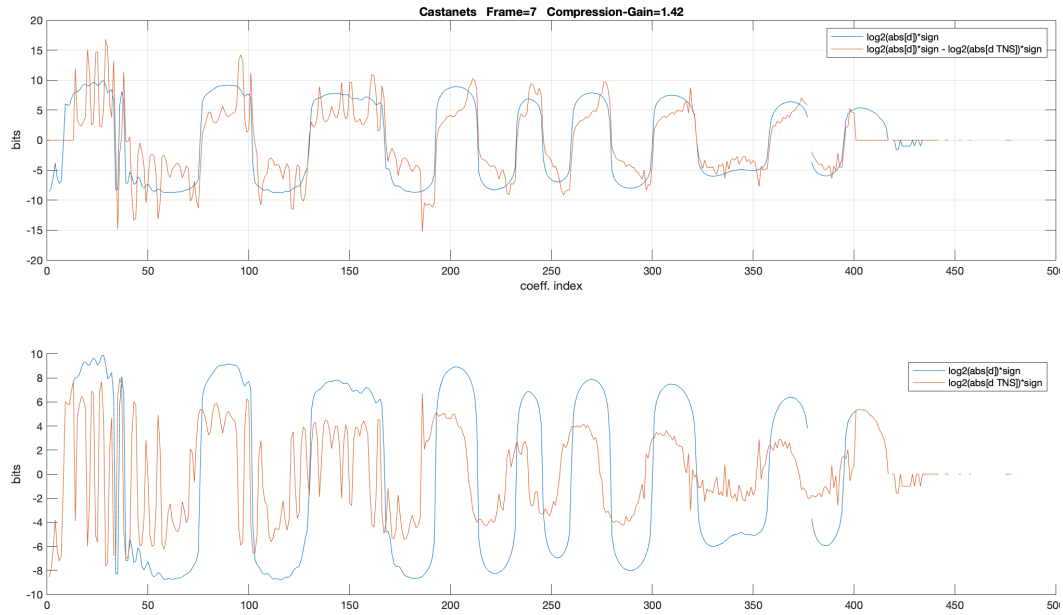


Figure 5.17: Sign Flipping Effect of TNS

Based on Figure 5.17, we can clearly observe some sign-flipping that occurs after encoding by the TNS. In this Figure, d represents the input to the TNS coder, while dTNS denotes the output of the TNS coder. For instance, for coefficient 50 in the second plot, the sign of the input to the TNS coder is negative, whereas the sign of the output is positive.

Assuming that LSB removal was applied to the MDCT spectrum before the TNS coder, and subsequent TNS decoding was performed, it becomes impossible to determine the original sign of the MDCT spectrum based on the input to the TNS decoder or even the earlier steps in the decoding process due to these sign-flipping occurrences. Therefore, to achieve lossless coding, it is essential to transmit the signs of the original MDCT coefficients. These signs are conveyed in a straightforward manner without any coding, as illustrated in Figure 5.18:

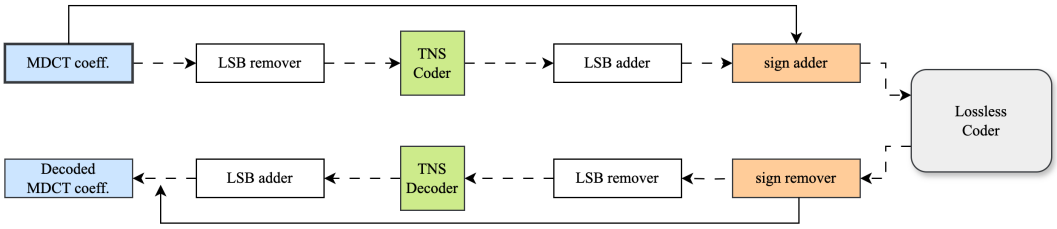


Figure 5.18: Architecture for V1 - Signs considered

This scheme serves as an initial solution; however, it introduces additional overhead to code associated with this approach. The question arises whether all the signs for the coefficients must be transmitted or only the signs for those coefficients that become zero after truncation. Alternatively, should we only transmit the zero coefficients? This prompts us to consider the implementation of residual coding in conjunction with LSB removal to manage the signs appropriately.

5.2.2.4 Version 2 of TNS development

In comparison to V1, we aim to establish a structure that introduces no coding overhead. By examining the previous architecture more closely, we can achieve further improvement by eliminating the LSB adder after the TNS coder and the LSB remover before the TNS decoder, as they would be redundant in this scheme. We can also transmit the removed LSBs using a residual coder. This residual coder writes bits sequentially, beginning with the highest removed bit plane and recording signs whenever a coefficient becomes zero after the removal. Consequently, signs for these coefficients will also be transmitted whenever required by the residual coder. This approach leads us to TNS version 2 (V2) of the implementation, as illustrated in Figure 5.19. A significant difference from the previous scheme is that the lossless coder will now operate on the truncated spectrum rather than the full spectrum.

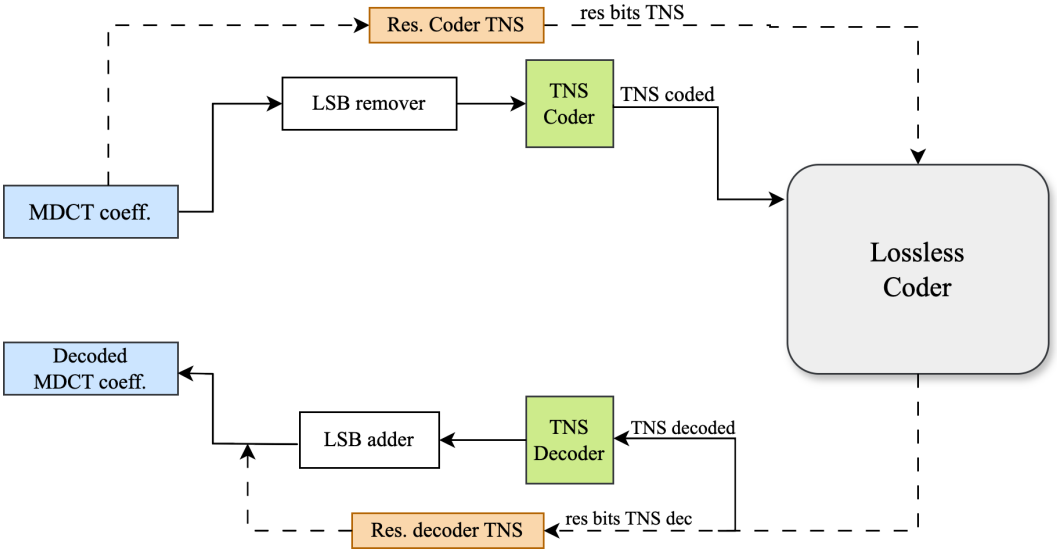


Figure 5.19: Architecture for V2

Regardless of whether TNS is active for a given frame, this scheme (V2) is applied to all frames.

5.2.2.5 Version 3 of TNS development

A more advanced structure for TNS—Fixed LSB Removal—is associated with TNS version 3 (V3), as illustrated in Figure 5.20. In this version, we employ the residual coder whenever there is an active TNS frame. If the prediction gain does not exceed the minimum prediction gain threshold and TNS is not activated, we utilize this feedback to transmit the original MDCT spectrum.

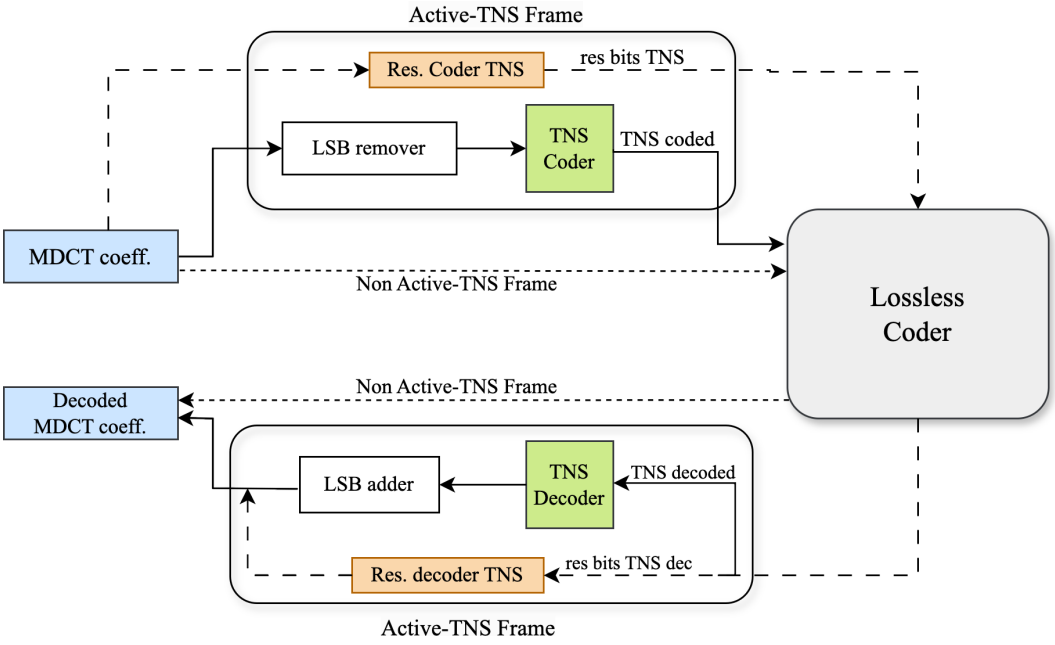


Figure 5.20: Architecture for V3

5.2.2.6 Relation of bit plane distortion and LSBs removal

An investigation has also been conducted on all transient items across various bitrates. To calculate the distortion, the following formula is used:

$$\text{Distortion} = mean[\ ceil(log2[|\text{Input of TNS Encoder} - \text{Output of TNS Decoder}| + 1])] \tag{5.5}$$

In Equation 5.5, the input to the TNS encoder simply refers to the original MDCT coefficients, while the output of the TNS decoder represents the decoded coefficients. The distortion defined here indicates the estimated number of bit plane differences between these input and output values per frame and per coefficient. This metric illustrates the extent to which the input to the TNS coder is distorted by LSB removal for that specific bitrate. A mean operator is also applied across all frames to obtain the distortion per coefficient. Figure 5.21 considers the Fatboy item, which contains 533 active TNS frames. As shown, there are some benefits to the LSB removal method for different coefficients, although these benefits only extend to a certain level. Additional figures can be found in Appendix 1.

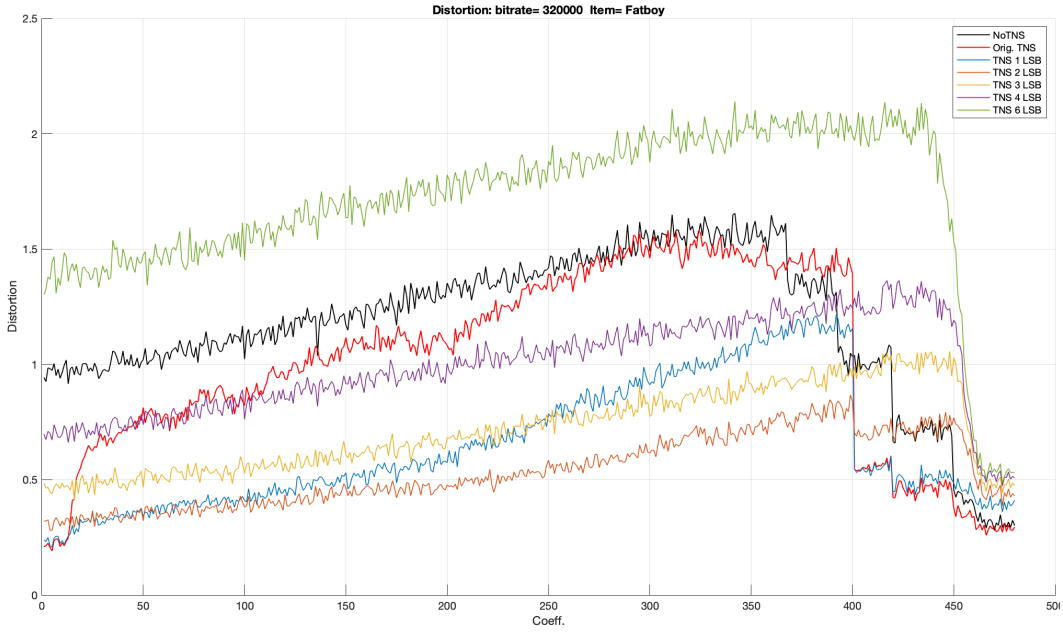


Figure 5.21: Average distortion per coefficient for Bitrate 320000

So far, for all frames where TNS is active, a fixed value will be established for the number of LSBs removed. For example, if the LSB removal is set to 2, the spectrum will be truncated by 2 LSBs for all frames. This scheme has been implemented to develop an adaptive version of TNS, which will be discussed in detail in the following Section, 5.2.3.

5.2.3 TNS - Adaptive LSB Removal

5.2.3.1 Minimum LSB Removal to be Lossless

So far, we have established a fixed value for LSB removal, which is set to a constant number. Now, we are looking for a way to be more adaptive and assign an LSB removal value on a frame-by-frame basis. We begin with Figure 5.22, conducted in CBR mode, which considers only active TNS frames for transient items. The five black graphs refer to different bitrates and illustrate the minimum number of LSBs that need to be removed to achieve lossless coding up to the TNS decoder. There are instances where the LSB level is -2, indicating that even removing 6 LSBs is insufficient for achieving lossless coding. As expected, as the bitrate increases, there is less need for LSB removal, making it easier to attain lossless coding.

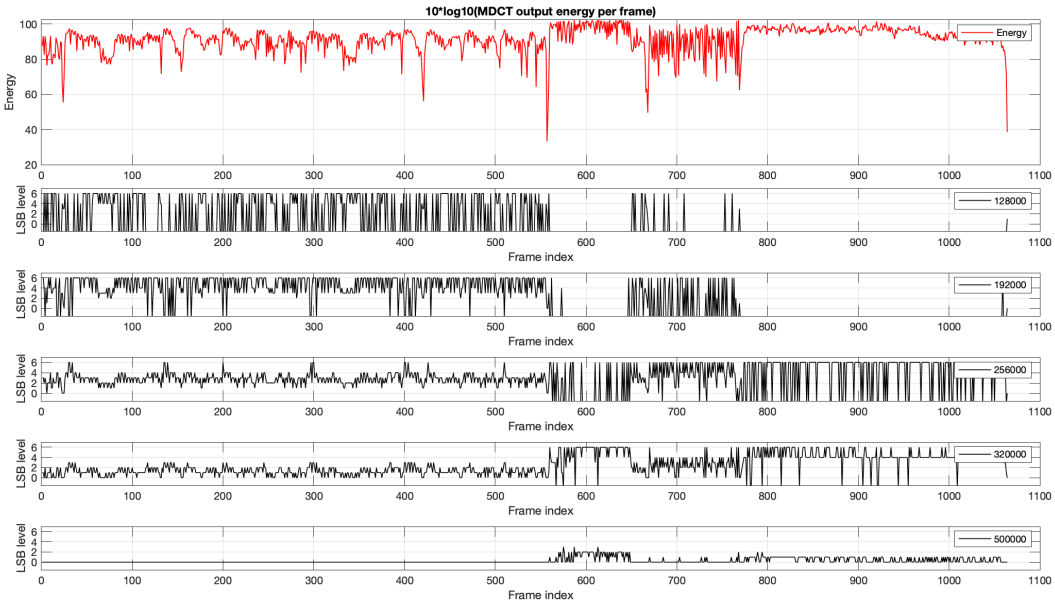


Figure 5.22: Minimum LSB removal to be lossless

The objective was to establish a relationship between the energy of the MDCT spectrum and the minimum number of least significant bits (LSBs) that need to be removed, which is not clearly depicted in this plot. Consequently, an alternative representation is employed to ascertain a better estimator, as discussed in Section 5.2.3.2.

5.2.3.2 Relation Prediction Gain Vs. Energy Ratio

The aim of this section is to establish a relationship between Prediction Gain and Energy Ratio. The Energy Ratio is defined as the ratio of the input energy to the output energy of the TNS coder. This energy ratio is articulated in Equation 5.6, where X represents the MDCT spectrum.

$$\text{Energy Ratio filter 1 } [\frac{\text{in}}{\text{out}}] = 10 * log10(\frac{sum(X_{in}^2)}{sum(X_{out}^2)}) \tag{5.6}$$

There are two filters in the TNS module within our configurations. Depending on the content of each frame, either one filter, two filters, or neither may be activated. The activation of these filters is determined by the prediction gain threshold. At the end of the TNS module, the prediction gain is recomputed based on the quantized TNS coefficients. For the sake of simplicity, these final prediction gains are referred to as "quantized gains" in the remainder of this text. These quantized gains will be utilized for coefficient processing. As illustrated in the following Figure, 5.23, there is a significant difference between the values of prediction gains and quantized gains:

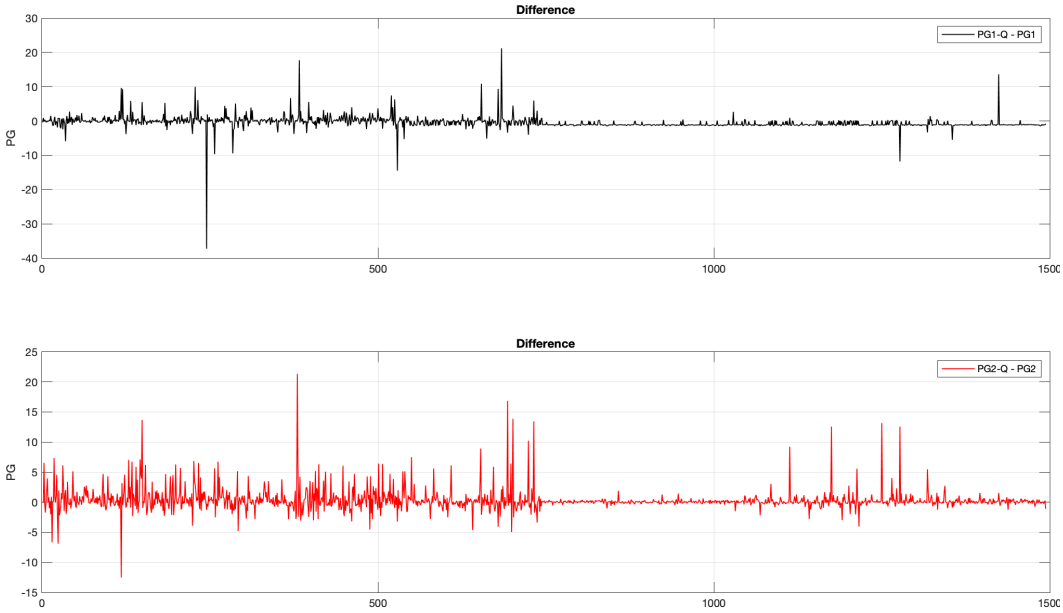


Figure 5.23: Prediction gains and quantised prediction gains

The quantized Prediction Gains from both filters in the TNS module are employed, as they have been found to provide a more accurate estimation. This is illustrated in Figure 5.24. If a mapping exists between these two variables, we can approximately ascertain the amount of energy reduced by TNS coding, thereby indicating the number of bits that are nearly preserved through this process.

This experiment could prove beneficial in Constant Bit Rate mode, given the coding is lossy or near-lossless coding. Based on the discussion in 5.2.2.1, to mitigate the substantial error that can arise from the lossy coding of TNS, we can remove some least significant bits (LSBs) and truncate the spectrum in such a manner that it can be encoded by TNS with minimal loss. In fact, we can compare the encoder's bit budget with a more realistic estimation of bit usage. If the bit budget cannot accommodate the estimated bit usage, we opt to truncate the spectrum and provide it to the TNS coder, thereby expecting it to be significantly closer to the lossless coding of TNS. By "more realistic estimation of bit usage," we refer to the consideration of the number of bits that can be saved through the TNS coder.

For instance, if a specific value of prediction gain corresponds to an energy ratio of 6 dB, it implies that the TNS reduces the energy to one-fourth of the original energy of the frame. This roughly equates to halving the spectral coefficients, which can be interpreted as the removal of one bit plane. In other words, after TNS coding, the spectrum is effectively shifted down by one bit plane. Following equations show this fact:

$$6dB = 10 * log10(\frac{sum(X_{in}^2)}{sum(X_{out}^2)}) = 10 * log10(\frac{eng_{in}}{eng_{out}})$$
$$0.6dB = log10(\frac{eng_{in}}{eng_{out}})$$
$$\frac{eng_{in}}{eng_{out}} = 3.98$$
$$eng_{in} \approx 4 \cdot eng_{out}$$
$$sum(X_{in}^2) \approx 4 \cdot sum(X_{out}^2)$$
$$sum(X_{in}) \approx 2 \cdot sum(X_{out})$$

Based on Figure 5.24, the relationship between prediction gain and energy ratio is modeled by a cubic polynomial function, as expressed in equations 5.7 and 5.8.

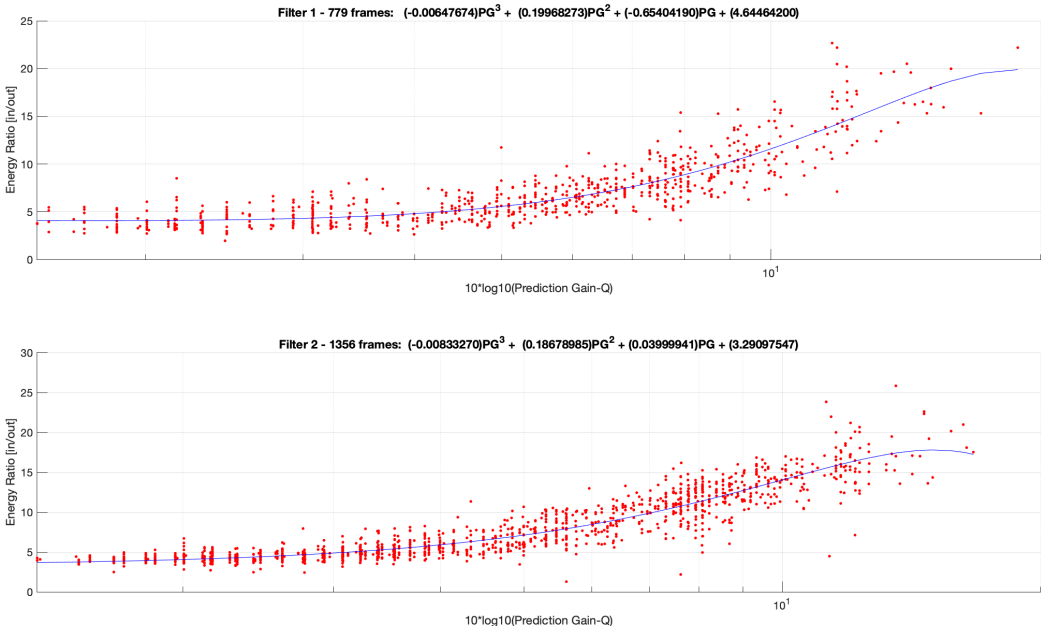


Figure 5.24: Relation between Prediction Gain and Energy Ratio

$$\text{Energy Ratio filter 1 } [\frac{\text{in}}{\text{out}}] = -0.006 \cdot \text{PG}^3 + 0.199 \cdot \text{PG}^2 - 0.654 \cdot \text{PG} + 4.644 \qquad (5.7)$$

$$\text{Energy Ratio filter 2 } [\frac{\text{in}}{\text{out}}] = -0.008 \cdot \text{PG}^3 + 0.186 \cdot \text{PG}^2 + 0.039 \cdot \text{PG} + 3.290 \qquad (5.8)$$

Figures 5.25 and 5.26 illustrate the transmitted bits and the actual number of active bits across the frame index under both conditions: when TNS is off and when it is on. In the second figure, the original active bit plane in the second plot indicates the number of active bit planes (ABP) for the MDCT coefficients. The reduced active bit plane is defined by Equation 5.9:

$$\text{Reduced Active Bit Plane} = \text{Original ABP} - \text{Estimated Preserved Bit Planes} \tag{5.9}$$

Thus, from this plot, we can observe the estimated number of preserved active bit planes as outlined in the following section.

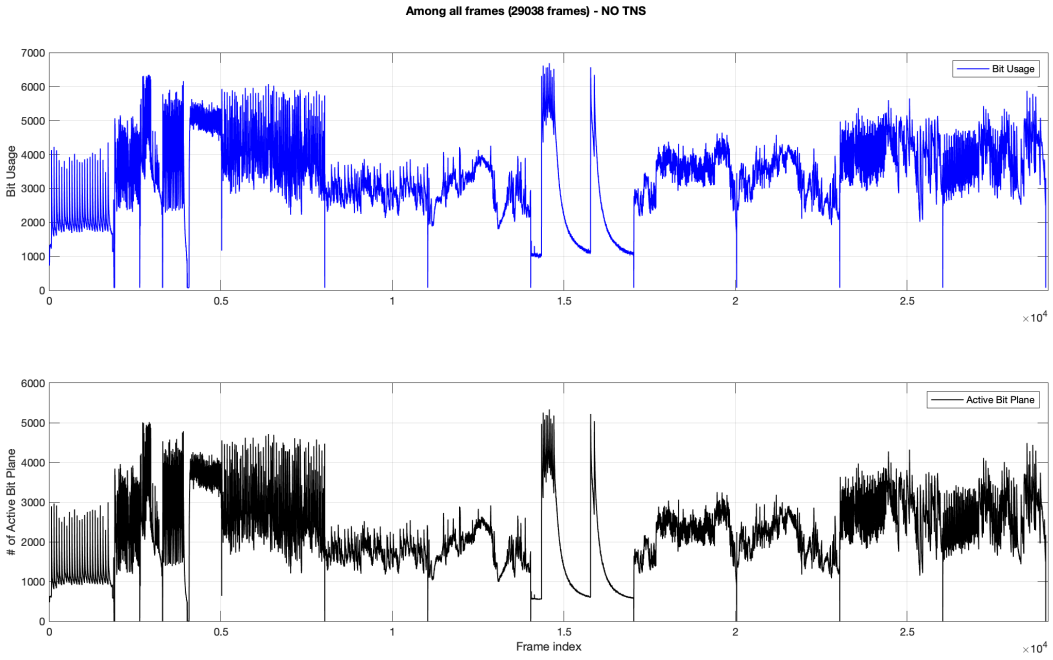


Figure 5.25: Number of Active Bit Planes and Bit Usage - NoTNS

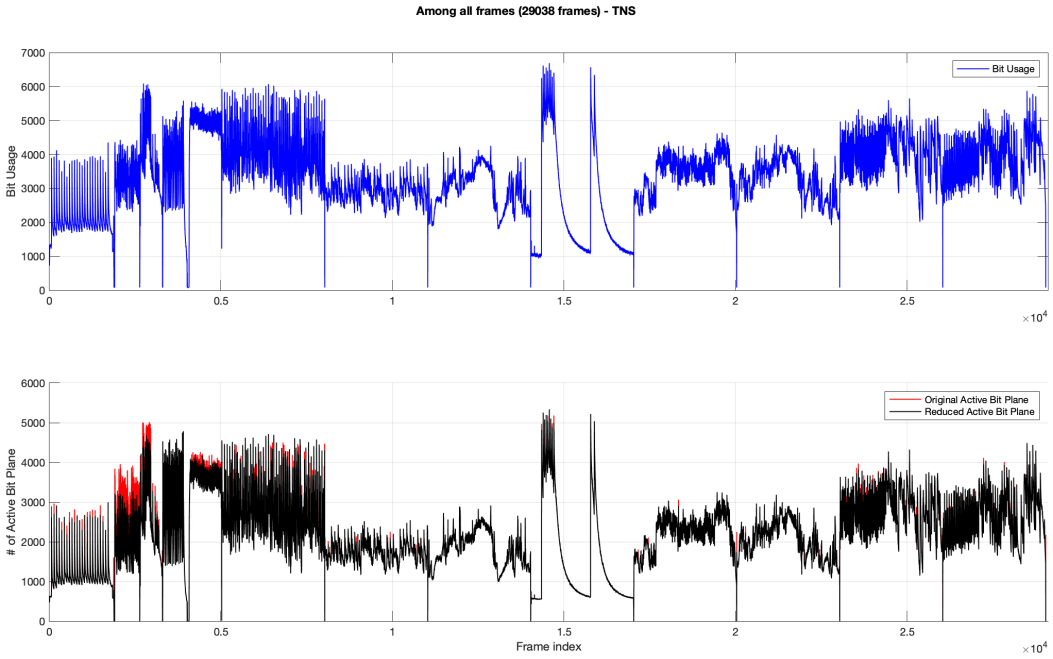


Figure 5.26: Number of Active Bit Planes and Bit Usage - TNS

5.2.3.3 Final Scheme

The final scheme relates to the adaptive TNS, which is the last framework for this study. In this scheme, we considered some function estimates. One involves mapping from active bit planes to the final bit usage, as shown in Figure 5.4, and the other concerns the two filters used in TNS. For the second estimate, we consider the mapping from prediction gain to the energy ratio of the input and output of the TNS coder, as shown in Figures 5.7 and 5.8.

Figure 5.27 describes the final scheme for each frame. First, the TNS coder is applied to check if the TNS filter is active. If the prediction gain is not large enough to activate the TNS coder, the decision will be made not to perform any TNS coding, and the original MDCT output will be sent. If TNS can be activated, further investigations must be carried out.

It must be checked whether only one filter is active for this frame or if both are. For each active filter, the mapping found for prediction gain and energy ratio, as shown in 5.7 and 5.8, will be used to calculate how many bit planes can be approximately saved (ABP reduction module in the figure, which shows the estimated number of preserved bits). The overall number of preserved bits is subtracted from the current active bit planes (ABP), as shown in Equation 5.9, to give us the estimation of active bit planes. This estimation of active bit planes is then used in the Bit Usage Estimation module, as shown in Equation 5.4, to estimate the bit usage. Next, this bit usage estimation must be compared with the total bits available in the encoding buffer. If the total bits are sufficient, TNS coding will be performed (original TNS coding module). If not, the difference between the total bits available and the bit estimation is calculated to determine how many ABP must be reduced to achieve almost lossless coding. This step is important, as discussed in Section 5.2.2.1. After that, the spectrum will be truncated by removing the estimated number of LSBs (TNS LSB removal module), and the adaptive TNS coder will be applied to the truncated spectrum (adaptive TNS coding module). In this scenario, the TNS LSB removal value must also be sent as side information.

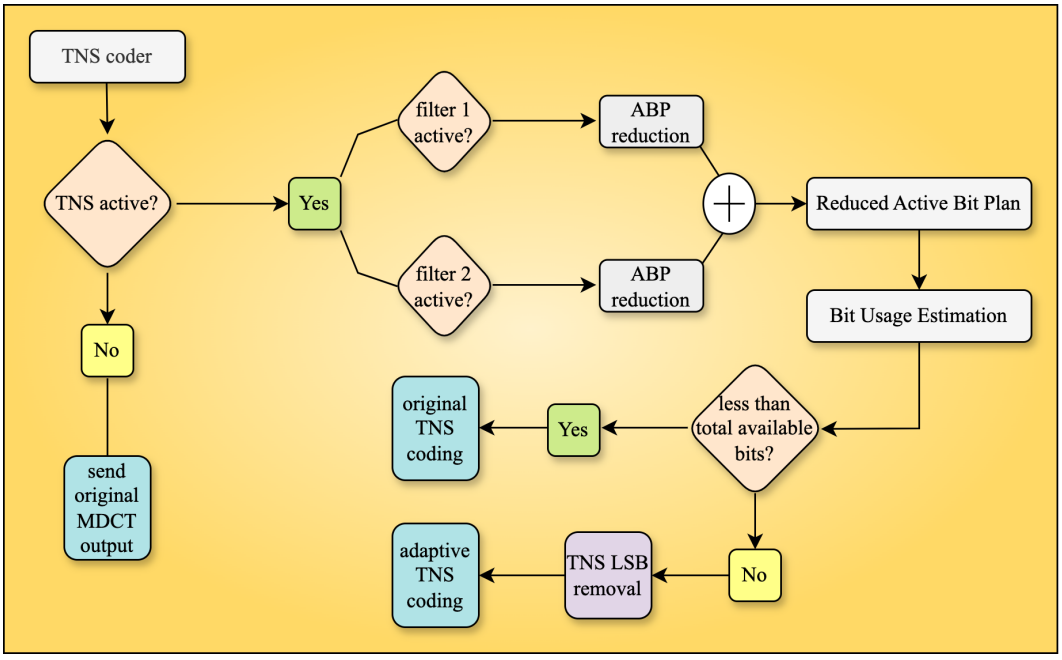


Figure 5.27: Adaptive TNS Scheme

5.2.3.4 Signal to Noise Ratio Comparison

In this section, we compare two differences in Signal to Noise Ratio (SNR): one is the difference between the SNR with TNS and the SNR without TNS, and the other is the difference between the SNR with adaptive TNS and the SNR without TNS. The computation is performed in the time domain, and the segmental SNR for each set of 480 samples is calculated. The experiment is conducted only for transient items across all frames, which includes both active and non-active TNS frames. We expect to see a small difference in SNR for non-active TNS frames due to the additional side information for the TNS flag. Since the analysis is in the time domain, the segments do not correspond one-to-one with the frequency domain because of windowing and overlapping. Equation 5.10 shows how to calculate the SNR.

$$\mathrm{SNR} = 10 \cdot \log_{10} \left(\frac{x[t]^2}{(x[t] - y[t])^2} \right) \tag{5.10}$$

Let x be the input signal and y be the output signal. The Signal to Noise Ratio (SNR) is defined as the ratio of input signal power to error signal power.

The last column in the following figures (labeled "Get Lossless") shows cases where there is zero difference between the input and output, resulting in infinite SNR. The blue color indicates that, without TNS, the coding was lossy, but after applying the original TNS, it becomes lossless coding. The orange color shows that, without TNS, the coding was also lossy, but after applying the adaptive TNS, it is lossless coding.

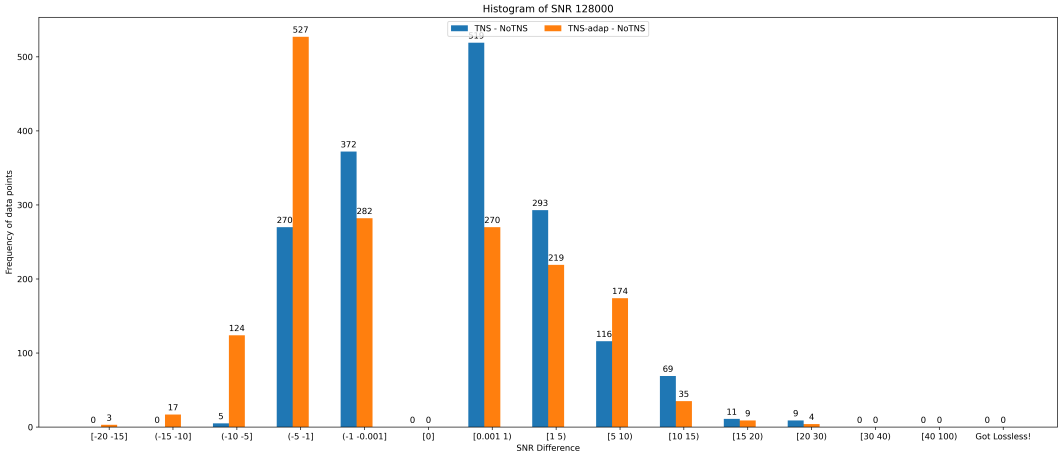


Figure 5.28: SNR Difference for bitrate 128000

In Figure 5.28, we can observe that with a bitrate of 128,000, there are no losslessly encoded frames. At this low bitrate, the benefits of adaptive TNS are not apparent.

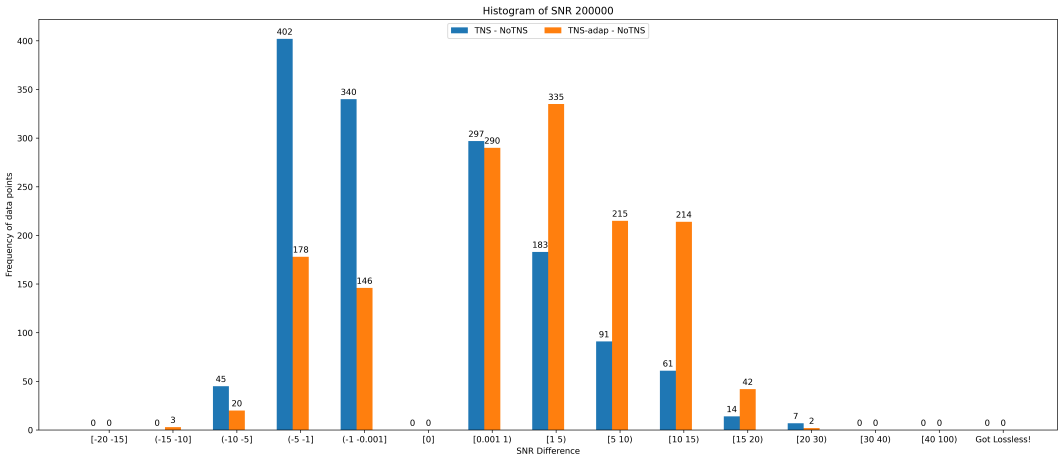


Figure 5.29: SNR Difference for bitrate 200000

In Figure 5.29, it is clear that at a bitrate of 200,000, there are benefits to using adaptive TNS. This is indicated by the ranges that show a positive SNR difference, which display larger values for adaptive TNS. However, there are still no losslessly encoded frames.

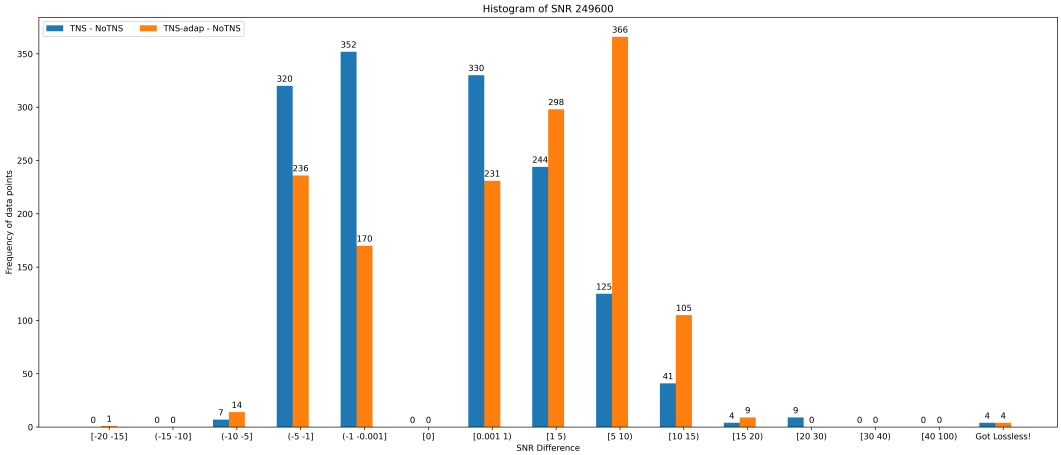


Figure 5.30: SNR Difference for bitrate 249600

In Figure 5.30, at a bitrate of 249,600, four frames are encoded losslessly due to the original TNS and adaptive TNS.

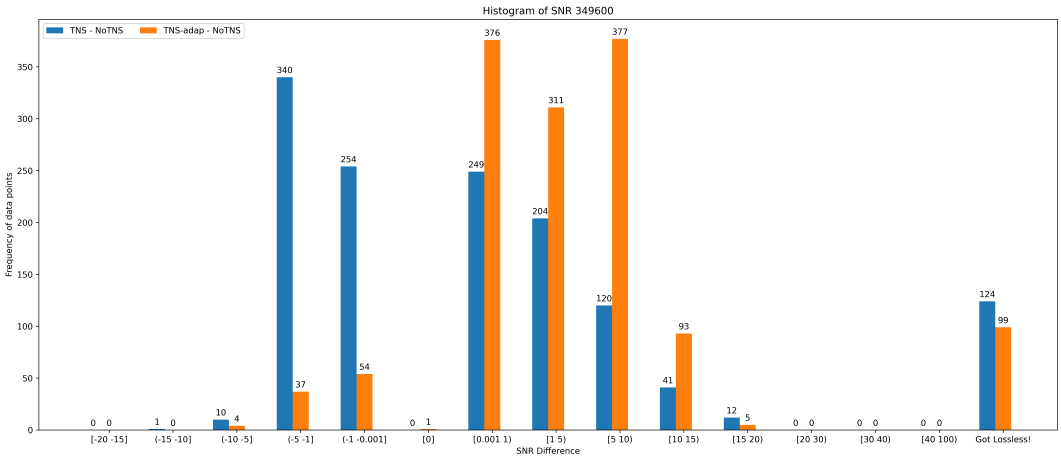


Figure 5.31: SNR Difference for bitrate 349600

In Figure 5.31, at a bitrate of 349,600, although 25 more frames are encoded losslessly with the original TNS compared to adaptive TNS, the benefits of adaptive TNS are evident. This is because the values for the negative ranges of SNR differences have reduced drastically, while the values for the positive ranges of SNR differences have increased.

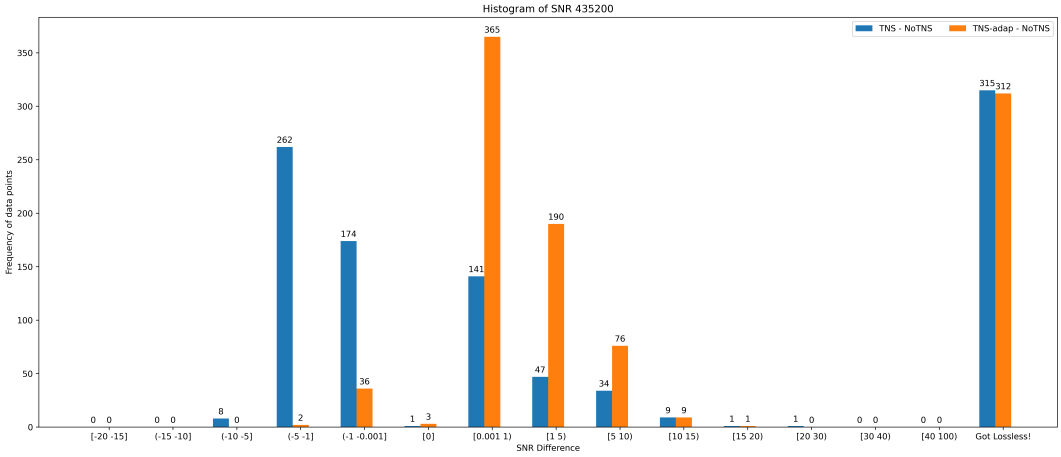


Figure 5.32: SNR Difference for bitrate 435200

In Figure 5.32, at a bitrate of 435,200, only three more frames are encoded losslessly with the original TNS compared to adaptive TNS. However, it is clear that adaptive TNS performs much better than the original TNS, as the values for the negative ranges of SNR differences have decreased drastically, approaching nearly zero. Additionally, the values for the positive ranges of SNR differences have increased significantly. It seems that the higher the bitrate, the better the performance of adaptive TNS.

Chapter 6

Conclusions

In this thesis, the effectiveness of Temporal Noise Shaping (TNS) as a pure predictor in both Variable Bit Rate (VBR) and Constant Bit Rate (CBR) encoding schemes is examined. The theoretical foundation, which includes linear prediction in both time and spectral domains, provides a comprehensive understanding of signal redundancy and predictive coding efficiency, leading to an introduction to TNS. Insights are also provided into how modern codecs handle audio compression. This study addresses both lossless coding and near-lossless coding.

For the lossless coding operating in Variable Bit Rate, the TNS module was added to the existing codec. A significant compression gain was observed for transient frames, as shown in figure 5.6, and the maximum bitrate often decreased, which is an important feature. At the same time, the compression gain did not worsen for the normal items, as illustrated in figure 5.7, even though additional side information was added due to TNS.

In near-lossless coding, the Constant Bit Rate mode was actived, and we developed a novel TNS module called adaptive TNS. This adaptive version is achieved through two function mappings: one mapping relates Bit Usage to Active Bit Planes, and the other relates Prediction Gain to Energy Ratio. The figures in Section 5.2.3.4 clearly show the benefits of this adaptive version. Specifically, as observed, there are some frames that degrade with the activation of the original TNS. This adaptive version significantly reduces these instances. We can reasonably conclude that the error from lossy TNS impacts the SNR considerably and that removing bit planes to ensure lossless TNS improves signal quality (at least according to objective measures). Future work could explore testing this adaptive TNS across different codecs and audio formats, which could broaden the applicability of these findings and enhance the robustness of modern audio coding systems.

Chapter 7

Acknowledgement

The author would like to express his sincere gratitude to his supervisors, Dipl.-Ing. Markus Schnell and M.Sc. Franz Reutelhuber, for their guidance during the Master's thesis. He would also like to thank his colleagues, Alexander Tschekalinskij and Maximilian Schlegel, for all the support he received during his time at Fraunhofer IIS. Additionally, the author extends his appreciation to Prof. Bernd Edler, his academic supervisor at FAU Erlangen-Nürnberg. Completing the Master's thesis at Fraunhofer IIS has been an honor, and the author will always be grateful for the opportunity.

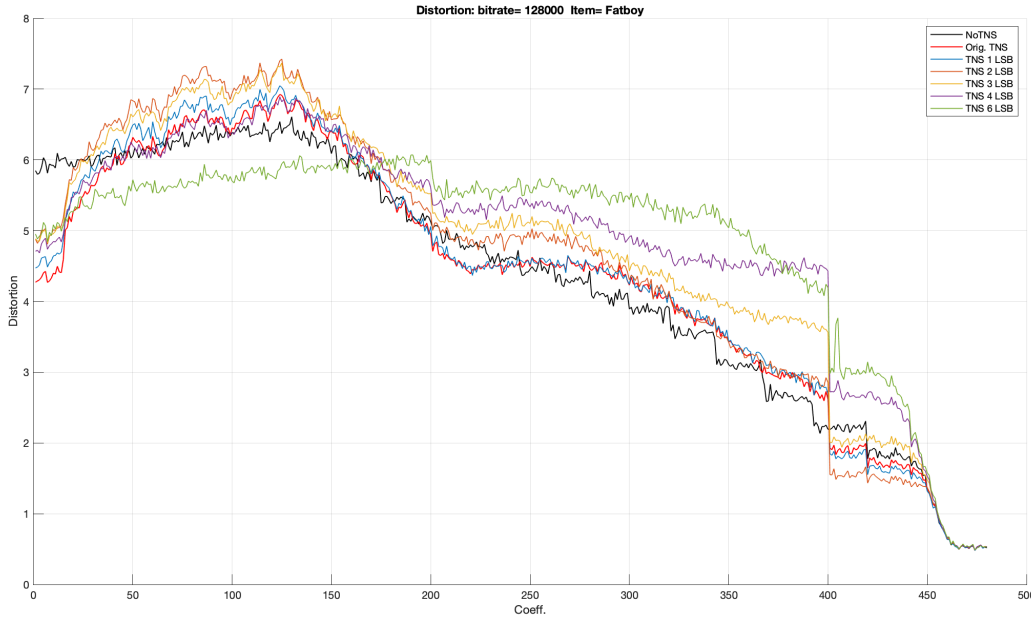
Chapter 8

Appendix

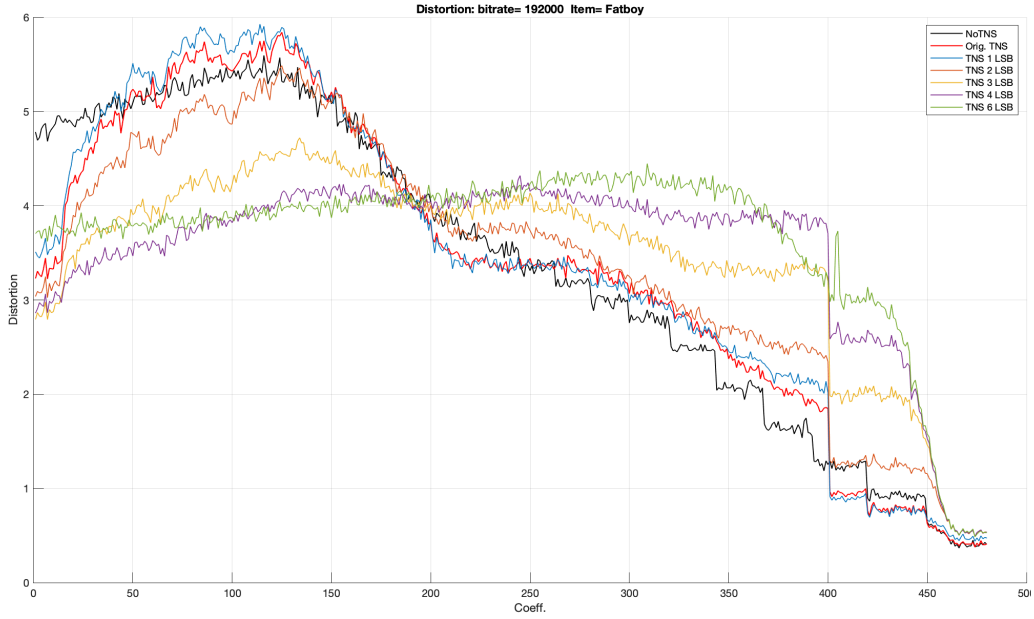
.1 About the Author

Nima Majidi an M.Sc. student at FAU Erlangen-Nürnberg, joined the Low Delay Audio Coding group at Fraunhofer IIS as a research assistant in October 2021. He contributed to both the DevOps environment and the signal processing aspects of the LC3Plus codec until October 2023. After completing a 5-month full-time research-internship at Stabilo International GmbH, he began his Master's thesis at Fraunhofer IIS in Tennenlohe. Currently, he is writing his thesis on the topic of transient coding and working as a student at Siemens Healthineers in Forchheim.

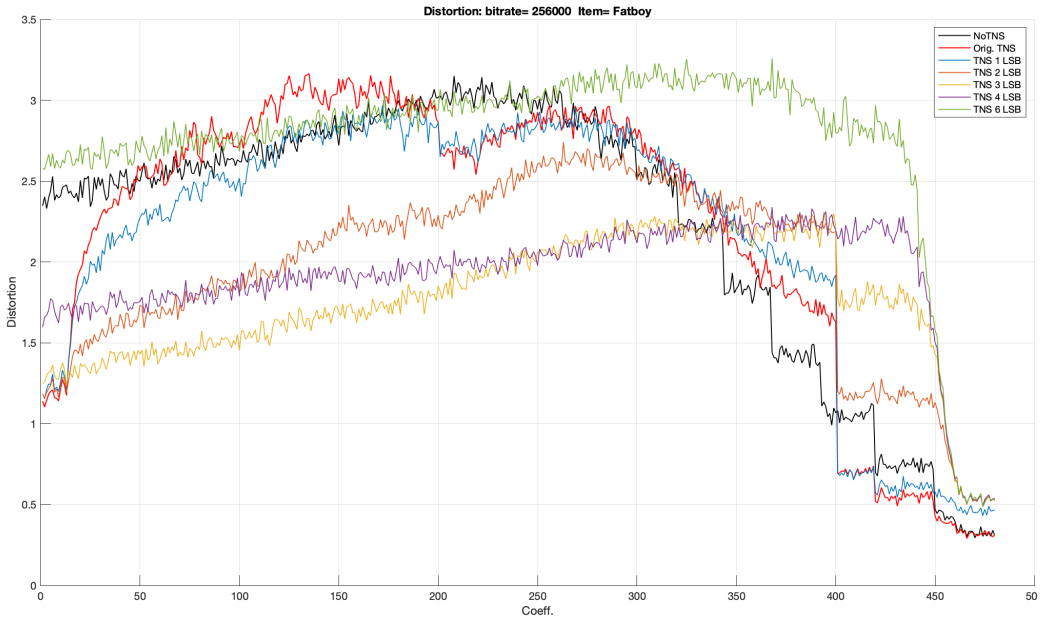
.2 Supplementary Figures



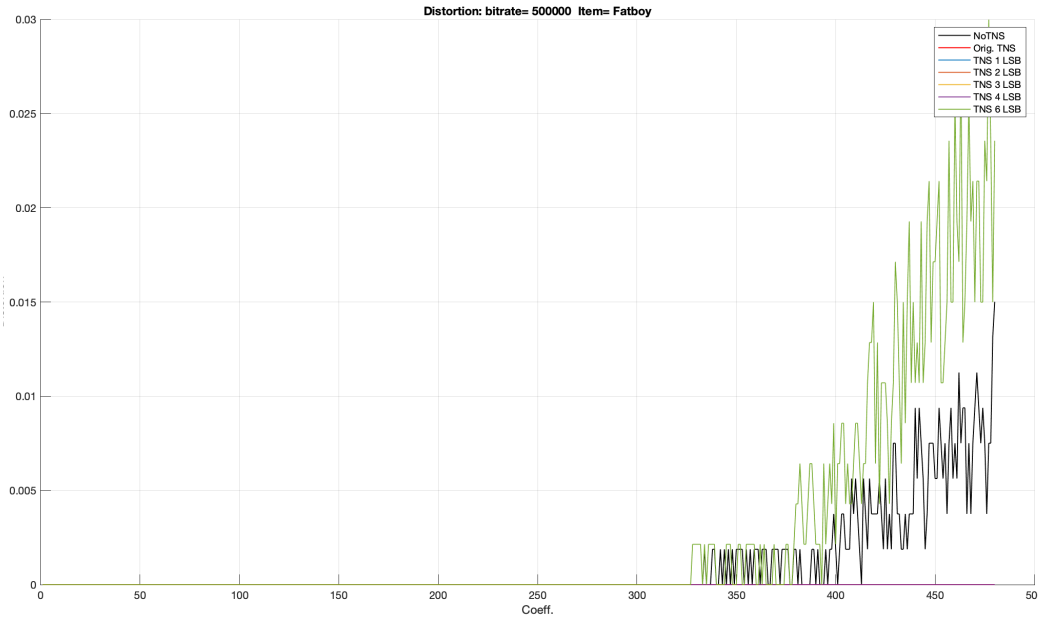
(a) Bitrate 128000



(b) Bitrate 192000



(c) Bitrate 256000



(d) Bitrate 500000

Figure 1: Average distortion per coefficient for different bitrates

Symbol	cumulative probabilty	symbol probabilty	length
d	0 = .000	0.125 = .001	3
b	0.125 = .001	0.25 = .010	2
a	0.375 = .011	0.5 = .100	1
c	0.875 = .111	0.125 = .001	3

New code point C = Current C + (A * cum_prob.)

New interval A = Current A * sym_prob.

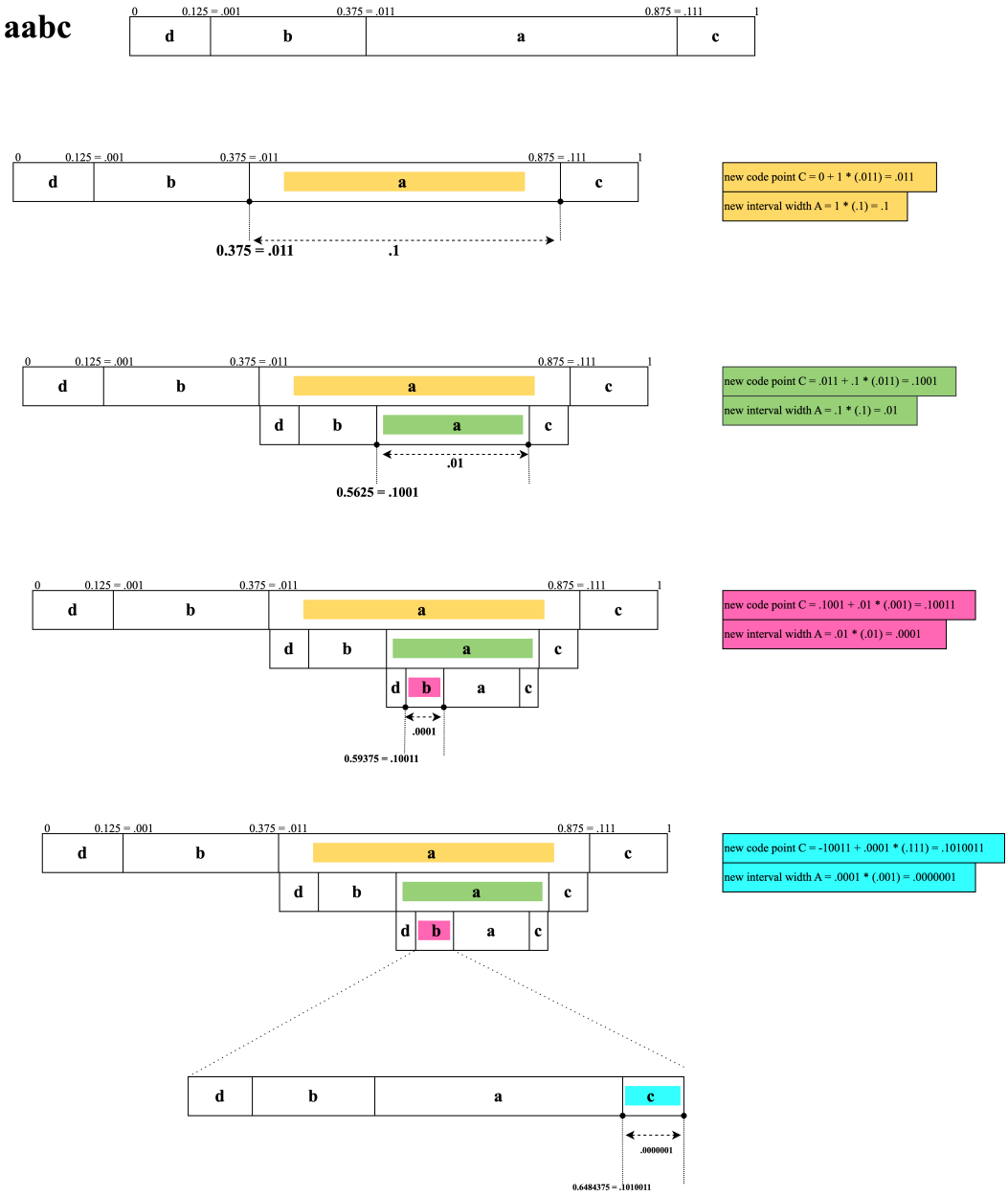


Figure 2: Arithmetic Encoder Algorithm

Bibliography

[1] Jürgen Herre and James D Johnston. Exploiting both time and frequency structure in a system that uses an analysis/synthesis filterbank with high frequency resolution. In *Audio Engineering Society Convention 103*. Audio Engineering Society, 1997.

[2] Jurgen Herre and James D Johnston. Enhancing the performance of perceptual audio coders by using temporal noise shaping (TNS). In *Audio Engineering Society Convention 101*. Audio Engineering Society, 1996.

[3] ETSI TS 103 634 V1.5.1. Digital enhanced cordless telecommunications (DECT); low complexity communication codec plus (lc3plus). `https://www.etsi.org/deliver/etsi_ts/103600_103699/103634/01.05.01_60/ts_103634v010501p.pdf`, 2024-06. Accessed: 2024-08-07.

[4] Nuggehally S Jayant and Peter Noll. *Digital coding of waveforms: principles and applications to speech and video*, volume 2. Prentice-Hall Englewood Cliffs, NJ, 1984.

[5] John R Deller Jr, John G Proakis, and John H Hansen. *Discrete time processing of speech signals*. Prentice Hall PTR, 1993.

[6] Johnson Princen, A Johnson, and A Bradley. Subband/transform coding using filter bank designs based on time domain aliasing cancellation. In *ICASSP'87. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 12, pages 2161–2164. IEEE, 1987.

[7] Joseph Rothweiler. Polyphase quadrature filters–a new subband coding technique. In *ICASSP'83. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 8, pages 1280–1283. IEEE, 1983.

[8] Karlheinz Brandenburg and James D Johnston. Second generation perceptual audio coding: the hybrid code. In *Audio Engineering Society Convention 88*. Audio Engineering Society, 1990.

[9] Karlheinz Brandenburg, Ernst Eberlein, Jurgen Herre, and B Edler. Comparison of filterbanks for high quality audio coding. In *[Proceedings] 1992 IEEE International Symposium on Circuits and Systems*, volume 3, pages 1336–1339. IEEE, 1992.

[10] William Yost. *Fundamentals of hearing: An introduction*. Brill, 2021.

[11] Brian CJ Moore. *An introduction to the psychology of hearing*. Brill, 2012.

[12] Manfred R Schroeder, Bishnu S Atal, and JL Hall. Optimizing digital speech coders by exploiting masking properties of the human ear. *The Journal of the Acoustical Society of America*, 66(6):1647–1652, 1979.

[13] Ye Wang and Mikka Vilermo. Modified discrete cosine transform: Its implications for audio coding and error concealment. *Journal of the Audio Engineering Society*, 51(1/2):52–61, 2003.

[14] Bernd Edler. Codierung von audiosignalen mit überlappender transformation und adaptiven fensterfunktionen. *Frequenz*, 43(9):252–256, 1989.

[15] Ralf Geiger, Gerald Schuller, Jürgen Herre, Ralph Sperschneider, and Thomas Sporer. Scalable perceptual and lossless audio coding based on mpeg-4 aac. In *Audio Engineering Society Convention 115*. Audio Engineering Society, 2003.

[16] Ralf Geiger, Thomas Sporer, Jurgen Koller, and Karlheinz Brandenburg. Audio coding based on integer transforms. In *Audio Engineering Society Convention 111*. Audio Engineering Society, 2001.

[17] Ralf Geiger, Jürgen Herre, Jürgen Koller, and Karlheinz Brandenburg. Intmdct-a link between perceptual and lossless audio coding. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages II–1813. IEEE, 2002.

[18] Arijit Biswas, Per Hedelin, Lars F Villemoes, and Vinay Melkote. Temporal noise shaping with companding. In *Interspeech*, pages 3548–3552, 2018.

[19] David Bindel, James Demmel, William Kahan, and Osni Marques. On computing givens rotations reliably and efficiently. *ACM Transactions on Mathematical Software (TOMS)*, 28(2):206–238, 2002.

[20] Ingrid Daubechies and Wim Sweldens. Factoring wavelet transforms into lifting steps. *Journal of Fourier analysis and applications*, 4:247–269, 1998.

[21] https://github.com/jgraph. https://www.drawio.com/. © 2005–2023 JGraph Ltd, Artisans' House, 7 Queensbridge, NN4 7BF, Northampton, England.