Low Latency Audio Processing

Yonghao Wang

School of Electronic Engineering and Computer Science
Queen Mary University of London
2017

Submitted to University of London in partial fulfilment of the requirements for the degree of
Doctor of Philosophy
Queen Mary University of London
2017

Statement of Originality

I, Yonghao Wang, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

Signature:

Date: 28/November/2017

Abstract

Latency in the live audio processing chain has become a concern for audio engineers and system designers because significant delays can be perceived and may affect synchronisation of signals, limit interactivity, degrade sound quality and cause acoustic feedback. In recent years, latency problems have become more severe since audio processing has become digitised, high-resolution ADCs and DACs are used, complex processing is performed, and data communication networks are used for audio signal transmission in conjunction with other traffic types. In many live audio applications, latency thresholds are bounded by human perceptions. The applications such as music ensembles and live monitoring require low delay and predictable latency. Current digital audio systems either have difficulties to achieve or have to trade-off latency with other important audio processing functionalities.

This thesis investigated the fundamental causes of the latency in a modern digital audio processing system: group delay, buffering delay, and physical propagation delay and their associated system components. By studying the time-critical path of a general audio system, we focus on three main functional blocks that have the significant impact on overall latency; the high-resolution digital filters in sigma-delta based ADC/DAC, the operating system to process low latency audio streams, and the audio networking to transmit audio with flexibility and convergence.

In this work, we formed new theory and methods to reduce latency and accurately predict latency for group delay. We proposed new scheduling algorithms for the operating system that is suitable for low latency audio processing. We designed a new system architecture and new protocols to produce deterministic networking components that can contribute the overall timing assurance and predictability of live audio processing. The results are validated by simulations and experimental tests. Also, this bottom-up approach is aligned with the methodology that could solve the timing problem of general cyber-physical systems that require the integration of communication, software and human interactions.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Professor Josh Reiss for his continuous support of my PhD work, for his patience, precision, motivation and wide knowledge. My sincere thanks also go to Professor Cham Athwal, who supports me to prioritise research and other commitments in the workplace. I'd like to thank all the colleagues in the Centre for Digital Music at the Queen Mary University of London and the DMT Lab at the Birmingham City University for their inspirations and collaborations. Many thanks to my PhD examiners Professor Rob Toulson and Dr Andrew McPherson, for their valuable advice and inspiration discussions. Finally, I would like to thank my family, parents, wife, and two lovely children for supporting me throughout eight years of work in this research area.

Contents

List of Figures

List of Tables

Acronyms

ADC Analogue to Digital Conversion.

ADAFX Adaptive Digital Audio Effects.

AF Asynchronous Flow.

AVB Audio Video Bridging.

CM Control Message.

CPS Cyber-physical system.

DAC Digital to Analogue Conversion.

DAW Digital Audio Workstation.

DSP Digital Signal Processing.

E2E end-to-end.

EDF Earliest Deadline First.

FFT Fast Fourier Transform.

FIR Finite Impulse Response.

GPOS General Purpose Operating System.

HAAF High-resolution Anti-aliasing Anti-image Filter.

IIR Infinite Impulse Response.

LCM Least Common Multiple.

LP Linear Phase.

MC Mixed-Criticality.

MP Minimum Phase.

NP-RMS Non-Preemptive Rate-Monotonic Scheduling.

OS Operating System.

OSI Open System Interconnection.

PTP Precision Time Protocol.

RJF Reduced Jumbo Frame.

RMS Rate-Monotonic Scheduling.

RTP Real-time Transport Protocol.

RTCP Real-Time Control Protocol.

SF Synchronous Flow.

SNR Signal to Noise Ratio.

STFT Short-Time Fourier transform.

SRC Sample-rate conversion.

TDCS Time Deterministic Cyclic Scheduling.

TDM Time Division Multiplexing.

TSN Time-Sensitive Networking.

WCET Worst-Case Execution Time.

Chapter 1

Introduction

1.1 Motivation

On January 25, 1915, to celebrate AT&T's first transcontinental telephone service, Alexander Graham Bell made a phone call to his former assistant Dr Watson in San Francisco from New York, repeating his famous statement: "Mr Watson, come here. I want you," Dr Watson replied, "It will take me five days to get there now!" [1, 2]. This is because this telephone call was made from about 4134km away.

When the room temperature is at 20ºC, the sound speed[1] is around 343.3 m/s [3], and the delay from a sound source to a listener is about 2.91 milliseconds(ms) per meter.

Sound transducers, which convert audio signals into electric waves, drastically increase the speed of dissemination of the signal to near the speed of light. The propagation speed of electromagnetic signals within copper is about 0.66 to 0.88 of the speed of light.

Imagining a sound is loud enough to be heard, it takes approximately 4.5 hours to transmit from San Francisco to New York, whereas with the telephone line, it takes at best 12ms on each way.

The audio and music industry is notoriously conservative when embracing new digital technology that potentially disrupts the 'vintage' of nostalgic analogue sound. Despite

[1] $C_s = 331.4(1 + 3.66 \times 10^{-3} \times \theta)^{1/2} \approx 331.4 + 0.6 \times \theta\ (m/s)$, where θ is temperature in Celsius.

this, computer music and digital audio effects have become mainstream since Moorer made the prediction in 2000 for future digital audio technology [4], as well as the audio industry has shifted from the mainframe to do-it-yourself (DIY) culture with miniaturised and democratised live applications [5].

Due to the rapid development of computer science and the Moore's law in the semiconductor industry, using a commodity computer with a Digital Audio Workstation (DAW) has become common practice. High-Resolution, low power audio ADC/DAC devices have become a necessity. Also, audio transmission is moving towards converged data communication networks along with other types of traffic. These advancements and conveniences sometimes bring more and unpredictable delay. The latency problem has often been discussed in various audio/music communities, second only to the complaint of 'cold' digital sound.

Human ears are sensitive to delays in many situations. For example, a standard telecommunication delay must be less than 250ms both ways to avoid deteriorating the conversation significantly [6], whereas high-quality music ensemble over network shall have end-to-end delay less than 25ms [7]. The human auditory system is also incredibly adaptive to the environment. Perhaps it is the unpredictable delay that makes people most uncomfortable with. In music ensembles, excessive delay between two audio sources makes musicians feel uneasy and hard to synchronise, and variations of delay makes bands difficult to coordinate.

From the system engineering point of view, the typical audio processing chain includes capturing, processing, transmission, routing and playback. The digital advancement enhances all parts of audio processing chain in aspects of conversion quality, processing capability, and cost-effectiveness of hardware and software. Meanwhile, the system latency seems not to be improved at a similar pace. On the contrary, lots of 'guesswork' and 'rule of thumb' of delay estimations are used in a system design.

1.2 Aim and Objectives

It is important to have accurate time delay parameters within a system design formula that help to predict the known delay effects to the users and audiences. In this research, we aim to answer the following research questions:

1. What are the sources of delay in the digital audio processing chain?
2. Can we accurately estimate the delay when designing a digital audio system?
3. Can we adopt deterministic approach in the system design to minimise the delay in the digital audio system?

The challenge of this research is to establish the deterministic and quantitative framework for evaluating the delay as a time parameter in system design from the bottom up. Related research in this field is often statistical and empirical, treating time delay as an uncontrollable parameters [8, 9, 10]. However, we aim to understand the root of this problem and propose reasonable solutions by fulfilling the following objectives:

- Investigate and identify the sources of latency in the digital audio system chain.
  - Investigate and test the delay of high-resolution ADC/DAC for audio conversion.
  - Test the delay of audio processing in Digital Audio Workstation (DAW) and operating systems.
  - Investigate the latency issues of audio networking.
- Develop accurate delay estimation methods for digital filters in ADC/DAC.
- Form a reduced group delay High-resolution Anti-aliasing Anti-image Filter (HAAF) design.
- Propose a new OS scheduling framework that is capable of doing low latency processing.
- Develop a deterministic audio networking architecture with supporting of convergence.

- Evaluate the performance of proposed methods and frameworks by using simulation and experimental tests.

1.3 The Method of Research

1.3.1 The Systems Engineering Point of View

In the digital age, there is an increasing human expectation of the 'smooth' and 'quick' response from digital systems, especially when dealing with interactive audio and video applications. In other words, being 'real-time' is not good enough. Users require a satisfactory responsiveness of the system - a predictable low latency.

In engineering science, latency is a measure of time delay experienced in a system and can be defined as the time delay between the input and output signals of a real-time signal processing system. Different applications have different latency requirements for real-time signal processing. Live audio processing can be considered as real-time audio processing with latency constraints. In a timing-driven system design and validation methodology for multi-input and multi-output scenario [11], the quantifiable constraints can be

1. On the same input to or output from a task: *a rate constraint*.
2. Between two different inputs or outputs: *an input correlation constraint* and *an output correlation constraint*.
3. From an input of a task to its output: *a latency constraint* or *an arbitrary separation constraint*, which is useful to define *response time constraints*.

The latency requirement in live audio processing is equivalent to the "*response time constraint*" from the system design point of view.

1.3.2 The Causes of Latency and Latency Taxonomy

The overall latency is caused by different components in the digital audio processing chain, and there are different classifications of the causes of latency in literatures range from 2002 to 2016 [12, 13, 14, 15, 16, 17, 18, 19]. However, in principle, they can be categorised into the following three factors:

- (a) Physical propagation latency (speed of sound and speed of light): G_p.
- (b) Buffering digital samples caused latency: G_b.
- (c) The phase delay of digital filters (group delay): G_d.

The delay components described in the literature in relation to these three factors can be depicted in the following Figure 1.1.

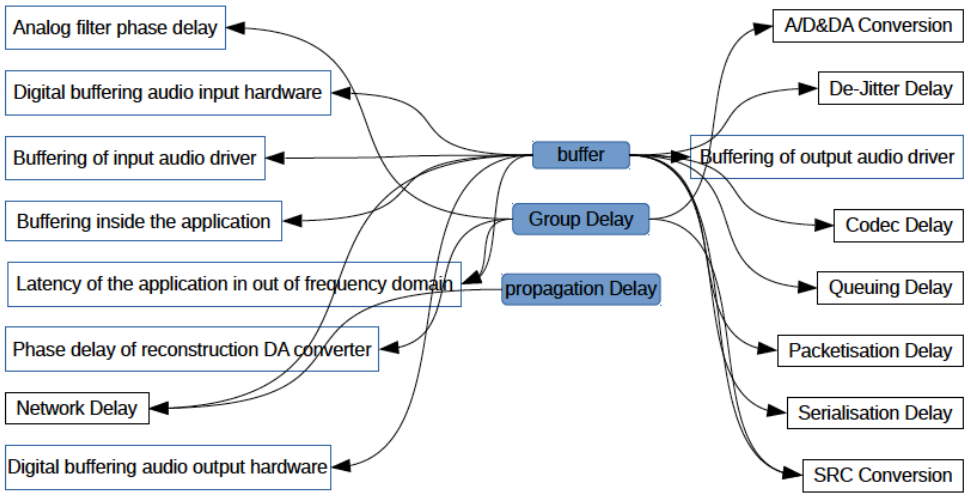


Figure 1.1: Different descriptions of the causes of latency in audio systems

Among these three factors, often (b) 'buffering' is the means to mitigate the problem of non-deterministic characteristics of some parts of the system. For example, buffering is used in computer sound subsystem as **digital buffering audio input/output hardware**, in the network interface to carry out **packetisation**, **serialisation**, **de-jitter**, and **queuing tasks**, in the bus system, and for the operating system scheduling algorithms. In other cases, buffering is used to accumulate a block of samples for further operation

such as frequency based processing (**Latency of the application in out of frequency domain**). The delay of a digital filter (c) can be observed from $\Delta\Sigma$ ADC/DAC devices because of their internal decimator and interpolator filters. Many DSP algorithms and SRC processes use these types of digital filters.

Some latency components are the combination of these three factors, for example, DSP algorithms that involve block based processing (b) and digital filters (c). Asynchronous SRC (ASRC) uses FIFO (b) and multiple FIR filters (c). Network delay also includes (a).

1.3.3 Reduce the Latency and Design Trade-Off

The overall latency can be reduced by minimising the latency of each component in the audio processing chain. However, in practice, there is often considerations and trading-off of design requirements over the whole system. For example, one can use ADC/DAC architectures other than $\Delta\Sigma$ based. However, the high resolution cannot be achieved in a cost effective way, and the analogue circuit and hardware cost can be very expensive in doing so. Therefore the effort can be made to use reduced group delay decimation and interpolation filter design for $\Delta\Sigma$ based ADC/DAC.

To reduce the buffer size, the system can be carefully designed to ensure that the real-time audio processing path is dealt with a time-deterministic manner. That requires not only fast enough computation, but also meeting the specific time deadline. This implies avoiding the use of non-real-time operating systems in the time critical path.

In addition, the DSP algorithms need to be designed in mind with minimum buffer requirements such as:

1. Use time domain sample based process, such as filter banks, rather than the frame based approach, such as FFT based processes [20, 21, 22].
2. Use reduced filter delay design such as minimum phase design, or linear phase filter that is optimised towards group delay.

In the large scale audio venue employing multiple audio channels, audio networking is a essential component. Current audio network technologies, such as AES50, and Ether-

sound, can achieve very low latency using time division multiplexing (TDM) methods, but these solutions require the sample rate of multiple channels to be rectified into the one sampling frequency or integer multiplication of that sampling frequency. Thus the SRCs are needed, which introduces additional latency that may be larger than the network latency itself.

1.3.4 Report Accurate Latency

There are misconceptions about zero delay design in the audio system. Bufferless processing often is regarded as 'zero delay' without considering the group delay caused by phase response. The delay of audio plugins and various DSP algorithms is either only reported incorrectly (using only buffer delay) or regarded as 'inherent delay' that is not accurately estimated.

The modern audio system is becoming complex with a large number of components and the accumulated delay can be significant. 'Live' audio systems that require processing audio input and output within certain time constraints because of human perception and interaction, are indeed examples of a Cyber-Physical Systems (CPS). Here, the current latency problem of the digital audio system is similar to that which exists in CPS, which is lack of timing accuracy reporting from different levels of abstract layers [23]. For example, the system can be described as a layered structure as in Figure 1.2.

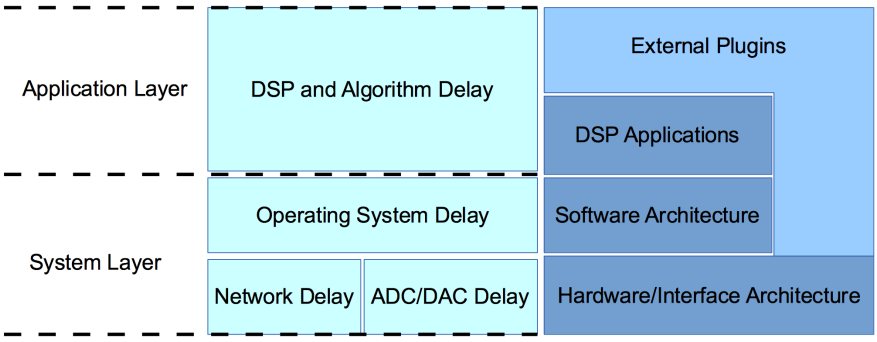


Figure 1.2: Different layers of delay in digital audio system

Often the lower layers of the system such as network delay and operating system delay

cannot be deterministic. They are often statistically measured as QoS values, examples like "98% tasks can be scheduled within 200μs" or "20ms when the network is not congested". These ambiguities affect the upper layer system design in terms of timing accuracy. The example of the latency of web audio is hard to manage due to it builds on top of various intermediate software layers [24].

1.3.5 Group Delay and Buffering Delay

Group delay and buffering are two distinguishable delay factors. Group delay is significant if it is caused by high order digital filtering. It is intrinsic to digital filters' time domain performance. Buffering is used for many different reasons. However, it is fundamentally to mitigate the non-deterministic time events. It is often set up for the worst case scenario. With propagation delay, they all contribute to the overall system latency.

To the best knowledge of the author, audio latency has not been clearly defined and evaluated at the system components level, and there is lack of low level design methods to optimise the system latency with consideration of various trade-offs of design options for the systems. This research investigates these areas using the methodology described in the following section.

1.3.6 Digital Audio System Signal Flow

We can model a typical digital audio system using the block diagram as in Figure 1.3. Figure 1.3 describes the general case of a digital audio system that can support live audio processing.

Figure 1.3 shows the typical components in a digital audio system with their logic and physical connections. The multichannel audio sources can be recorded in live and input from the block '*P1*' or the off-line sound files in block '*P2*'. The '*Routing Matrix'* acts as the audio bus to route audio channels according to the user configurations. Some of the audio sources are routed to the feature extraction and control block '*B3*'. The output audio signals are transmitted to the different destinations.

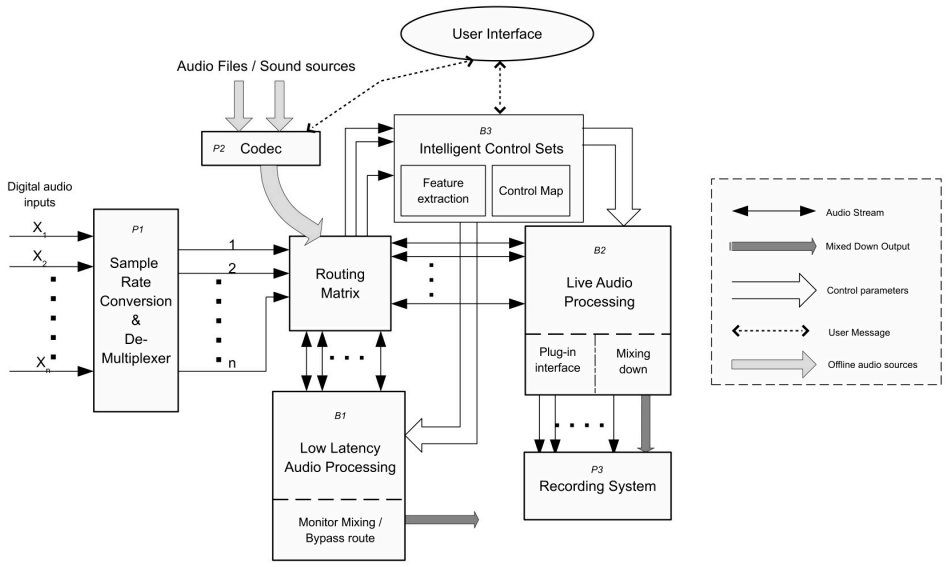


Figure 1.3: Digital audio system

For analysing latency along, the block diagram can be further refined using the signal flow graph that has been developed by Mason in 1956 [25]. This can be modelled as a 'signal flow graph' shown in Figure 1.4. The solid lines represent the audio signal flows and the dashed lines represent the control message flows. To use the system for low latency processing, a sub-route can be identified as the time critical path.

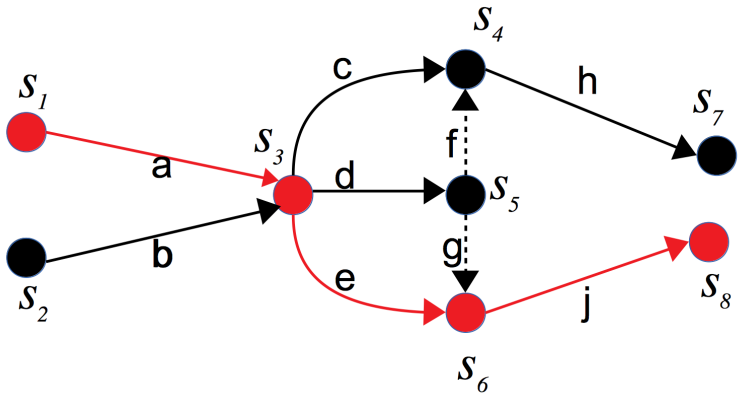


Figure 1.4: Signal flow graph model of audio processing system

The system can be represented as a graph $G_{sys} = (V, E)$, V are vertices, and E are edges. In this example, we have $V = \{S_1, S_2, S_3, ...S_8\}$, $E = \{a, b, c, ...j\}$. The time of system input and output that the user is interested in can be regarded as the critical path of the system as $C_{sys} \subseteq G_{sys} = (V_c, E_c)$. where V_c and E_c represent the vertices and edges of critical path subset. In this example $V_c = \{S_1, S_3, S_6, S_8\}$, $E = \{a, e, j\}$

Let G_{bi}, G_{di} be the buffering and the group delay of node S_i. G_{pi} is the propagation delay between two connected points S_i. We can define the interested delay of the digital audio system as Equation (1.1) :

$$D_{sys} = \sum_{i \in V_c} (G_{bi} + G_{di}) + \sum_{i \in E_c} G_{pi} \tag{1.1}$$

For audio signal transmitting within a limited geographic area such as within computers and peripherals, the G_p is in the range of picoseconds that can be ignored.

1.3.7 System Engineering Approach

The model in Figure 1.4 and Eq. (1.1) shows the system level latency and how it is composited. In this work, we investigate three main subsystems in the digital audio processing chain including ADC/DAC, operating system and audio networking. Both white box and black box approaches are used to carry out the detailed tests of some components latencies for part of the system with the properly designed methodologies. The results and findings then lead to the analytical approaches to the sources of the problem. Using the white box approach, we analyse and propose new mathematical models, optimisation methods, and a new system architecture, that overcome the problem of ambiguous reports of system latency values between system layers and provide a deterministic time domain performance.

Therefore, in this research we investigate the roots of audio processing latency using a bottom-up and deterministic approach, to identify and simplify the common factors that affect the overall latency, and to evaluate and estimate the latency quantitatively. In short, the method of the research is to make Eq. (1.1) determinable and try to minimise each term of it.

1.4 The Scope of the Research

The methods adopted in this research do not try to address the system design question, which is how to produce bespoke digital systems that satisfy the latency requirements. However, this research investigates the latency problems in three of the most significant parts of digital audio processing chain:

1. Latency caused by the $\Delta\Sigma$ based audio ADC DAC.
2. Latency caused by the General Purpose Operating System (GPOS) based audio processing platform.
3. Latency in the converged network infrastructure for the professional audio transmission.

Capturing (ADC) and playing back audio (DAC) are two most front-end interfaces between the analogue world and the digital world. We shall ask these questions: Is there any delay when we capture and playback the audio signal in the high-resolution digital era? What causes the delay? Can it be reduced?

The advantage of GPOS based system is to processing digital audio with tremendous flexibility and convenience. There are versatile DSP algorithms and almost infinite combinations of them to process audio. We focus on the fundamental questions that cause the processing delay in GPOS. Why do we need to wait for a number of audio samples to be accumulated (buffered) before processing them? Does processing algorithms cause the delay? Can all complexing processing tasks be scheduled properly so that the outcome delays are predictable?

In the audio networking area, is it the right direction to use a common networking infrastructure to transmit audio? What are the problems with it? How can we improve it so that the delay of audio transmission can be predictable and minimised? Can audio transmission co-exist with other types of traffic? How to transmit the multichannel audio streams each with different bandwidths and sampling rates?

These questions are reviewed and researched with consideration of human perception of delay and interactions, which give us the guidance of the magnitude of each problem

and the time constraints that we can work on. The outcomes of research are not limited to these three aspects. They address the more fundamental causes of latency, and the concepts and theory can be applied in broader perspectives of the signal processing chain.

1.5 The Organisation of the Thesis

Chapter 2 reviews the state of the art of current research concerning system latency in audio processing including the perception of latency and the tolerable latency of human interaction; the typical latency values in the digital audio processing chain; OS processing delay; the group delay of high order filters; and networking delay. Other relevant delay components such as the algorithm delay, the architecture delay are also discussed.

Chapter 3 presents the test of $\Delta\Sigma$ ADC/DAC delay and an evaluation of time domain properties of different multistage multirate filters that cause the delay in $\Delta\Sigma$ ADC/DAC. A theory of accurate prediction of delay and formalised delay as an objective function of multistage design parameters. A simplified method to find the overall balanced design that takes delay into account. An objective measurement of delay in minimum phase system is also presented with a quantitative analysis.

Chapter 4 presents the updated latency measurements of the desktop OSes with DAWs. Particularly, the latency issues under heavy CPU load caused by audio processing are discussed when using commodity General Purpose Operating System (GPOS). A new time deterministic OS scheduling framework specifically designed for low latency audio processing is proposed. The simulation results in comparison with classic rate monotonic scheduling are presented and discussed.

Chapter 5 proposed a new low latency audio network architecture and protocols that can support different audio sampling rates in conjunction with other non-real-time data. The proposed architecture provides the deterministic audio data delivery that can achieve the stability that below audible jitter. The software simulation and hardware prototype testing of it are also presented.

1.6 Contributions

Chapter 1: A new latency taxonomy based on meta research and system analysis.

Chapter 3: A latency measurement of modern audio $\Delta\Sigma$ ADC/DAC and a comprehensive evaluation of time domain performance of different types of multistage and multirate filters for $\Delta\Sigma$ ADC/DAC.

Chapter 3: A simplified optimal design sets search method for multistage and multirate filter design for $\Delta\Sigma$ ADC/DAC.

Chapter 3: Formalisation of group delay as an objective function of multistage filter design parameters and proposed a new multi-objective optimisation method to design filters for $\Delta\Sigma$ ADC/DAC including overall delay.

Chapter 3: A set of new objective measurements formulas for Minimum Phase (MP) filters and the evaluation over High-resolution Anti-aliasing Anti-image Filter (HAAF).

Chapter 4: A comprehensive latency test using modern DAWs and OSes with consideration of internal audio load and cross-adaptive effects.

Chapter 4: Time Deterministic Cyclic Scheduling (TDCS) - a new scheduling framework for low latency audio processing with simulation results.

Chapter 5: Design new low latency deterministic network protocol - Flexilink.

Chapter 5: A comprehensive performance evaluation of Flexilink against other priority-based network architectures with simulation and hardware test.

1.7 Associated Publications

Chapter 3, Section 3.1 was published as

- Y. Wang, “Latency Measurements of Audio Sigma Delta Analogue to Digital and Digital to Analogue Converts,” in 131st AES Convention, New York, NY, USA, 2011. [26].

The author of the thesis wrote and did the main research.

Chapter 3, Section 3.2 was published as

- Y. Wang and J. Reiss, "Time Domain Performance of Decimation Filter Architectures for High Resolution Sigma Delta Analogue to Digital Conversion," in Audio Engineering Society Convention 132, 2012. [27].

The author of the thesis wrote and did the main research. The other author had an editing and supervising role.

First part of Chapter 3, Section 3.3 was published as

- X. Zhu, Y. Wang, W. Hu, and J. D. Reiss, "Practical considerations on optimising multistage decimation and interpolation processes," in Digital Signal Processing (DSP), 2016 IEEE International Conference on, 2016, pp. 370–374. [28].

The author of the thesis wrote the main paper, discovered the regularity of optimal solution sets, and proposed the simplified search and balanced design algorithm. Xiangyu Zhu implemented factorisation algorithm as the core part of the simplified search method, as well as other implementations and obtaining the testing results. Other authors had an editing and supervising role.

Chapter 4, Section 4.1 was published as

- Y. Wang, R. Stables, and J. Reiss, "Audio Latency Measurement for Desktop Operating Systems with Onboard Soundcards," in Audio Engineering Society Convention 128, 2010. [12].

The author of the thesis wrote and did the main research. Other authors had an editing and supervising role.

Chapter 5, Section 5.1 was published as

- Y. Wang, J. Grant, and J. Foss, "Flexilink: A unified low latency network architecture for multichannel live audio," in 133th Audio Engineering Society Convention, 2012. [29].

The author of the thesis wrote the main paper. The design of new network protocol and architecture were the results of the discussion with John Grant, who shared the experi-

ences and ideas from his audio over ATM work and audio networking standardisation work. Other authors had an editing and supervising role.

Chapter 5, Section 5.2 was published as

- Y. Song, Y. Wang, P. Bull, and J. D. Reiss, "Performance Evaluation of a New Flexible Time Division Multiplexing Protocol on Mixed Traffic Types," in Advanced Information Networking and Applications (AINA), 2017 IEEE 31st International Conference on, 2017, pp. 23–30. [30].

The author of this thesis contributes the main idea of simulation, the design of simulation model, and the main testing strategy. Yangyang Song did the implementation and the test. Yangyang Song also written the first draft with author's help in structure and final modification. Other authors had an editing and supervising role.

1.8 Unpublished Results

Second part of Chapter 3, Section 3.3.

- The quantitative analysis of delay of high-resolution anti-aliasing and anti-image filter.

Chapter 4, Section 4.2.

- Time Deterministic Cyclic Scheduling (TDCS) - A new scheduling framework for real-time Multimedia OS.

Chapter 5, from architecture design to new hardware testing results.

- Towards true convergence, the architecture and performance evaluation of dynamic TDM system: Flexilink.

We aim to publish these results in the near future.

1.9 Other Relevant Publications

There are other research work the author contributed. They helped to reinforce the findings of this research, to understand the general problem in current audio processing software and hardware architecture, and to inspire the proposed solutions:

- Y. Wang, X. Zhu, and Q. Fu, "A Low Latency Multichannel Audio Processing Evaluation Platform," presented at the Audio Engineering Society Convention 132, 2012. [31].
- N. Jillings and Y. Wang, "CUDA Accelerated Audio Digital Signal Processing for Real-Time Algorithms," in Audio Engineering Society Convention 137, 2014. [32].
- N. Jillings, Y. Wang, J. D. Reiss, and R. Stables, "JSAP: A Plugin Standard for the Web Audio API with Intelligent Functionality," in Audio Engineering Society Convention 141, 2016. [33].
- N. Jillings, Y. Wang, R. Stables, and J. D. Reiss, "Intelligent audio plugin framework for the Web Audio API," 2017. [34].

Chapter 2

Literature Survey

In this chapter, we discuss the perception of latency and the tolerable latency of human interaction. The typical latency values in the digital audio processing chain are reviewed. We review the state of the art for different aspects of the system latency of audio processing within the research scope, including system latency test, delay in $\Delta\Sigma$ ADC/DAC, the group delay of digital filters, OS scheduling delay, and delay in audio networking.

2.1 Perception of Latency

The human ear is very sensitive to time properties of audio signals. The auditory perception of latency effects many live audio applications. In many cases, if the delay between two sequential expected events is beyond a certain threshold, it causes negative effects. In most cases, people prefer the lowest latency possible. Table 2.1 summarised the typical latency thresholds for different applications [6, 35, 36, 37, 7, 38].

For musical ensembles, the performance can be naturally synchronised when the latency is between 8ms to 25ms [37, 7], where the latency is equivalent to the time sound travels in common distances of players. If the latency is greater than the upper threshold, the performance can be seriously deteriorated due to the difficulty to synchronise. If the latency is shorter than the lower limit, the performance could result in 'racing' effect. In

Table 2.1: Perceptual Latency Thresholds

Application	**Threshold (up limit)**	**Threshold (lower limit)**
Telecommunication	250ms	Unknown
Interactive VR	50ms	Zero
Action to Sound	10-30ms	Zero
Music Ensemble	25ms	8ms
Live monitoring	10-30ms	Zero
In-ear monitoring	3-10ms	Zero

this case, each individual performer constantly feels falling behind and tries to accelerate their tempo to catch up. In addition, the visual cue of seeing conductor and companions plays an important role in synchronisation.

For audio monitoring, the tolerable latency varies according to different types of playing. In general, the 2007 research by Lester et al. found the vocalist and brass player prefer lowest latency (zero delay) in the in-ear monitoring situation, whereas for keyboard player, the performer is more tolerant for higher latency values [36]. For computer instruments that the tolerable latency normally shall be less than 10ms [39, 38]. It may vary depending on the different instruments [35].

For multichannel audio processing, the latency of each channel needs to be quantified so the correct compensation can be made to maintain the stereo image: a few sample differences between channels will cause colourisation of the mix [40].

2.2 Trends in Digital Audio Systems

A digital audio processing system used in professional audio can be depicted as the diagram in Figure 2.1. It normally consists most or all of the following components. Each of these components could contribute to the overall latency.

- analogue to digital conversion (ADC)

- digital to analogue conversion(DAC)
- audio routing and mixer (Mixer)
- digital signal processing (DSP)
- audio networking
- interfaces for external software and hardware plugins (Plugin)
- digital sample rate conversion (SRC)

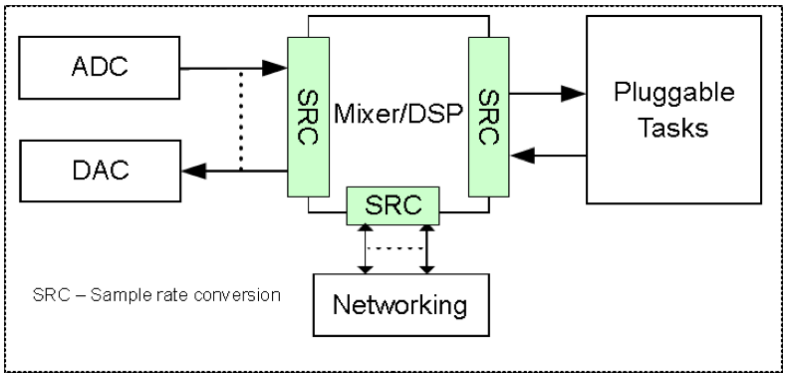


Figure 2.1: Digital audio system

In recent years, various digital technologies have been developed to enhance functionalities and features of each component of the system including high-resolution conversion, multi-inputs and multi-outputs, downsampling and upsampling, plugin architecture, and flexible routing/mixing capacity for a large number of channels. These trends can be summarised below:

- It becomes ubiquitous, compact and power efficient as well as dealing with an increased number of bits resolution. This enables portable and handhold devices to process audio signals with high resolution.
- The processing becomes more intelligent than before. It is common to process a large number of channels with intelligent functionalities such as automixing.
- It requires convenient and flexible distribution, transmission, and routing, often in conjunction with other multimedia data and control signals, hence audio networking is used rather than the traditional snake cables.

The oversampling $\Delta\Sigma$ ADC/DAC architecture is used to meet high-resolution ($>$20bits) requirement with low hardware cost. Currently, it is the only type of audio ADC/DAC that can be used in portable devices. The latency caused by it is due to the internal high performance digital filters.

The Mixer and DSP components are either in a hardware console or computer-based digital audio workstation (DAW) form. For a computer based system, it is difficult to schedule time-deterministic tasks at the audio sample rate. Therefore various buffers are used to mitigate this non-deterministic timing problem, which causes additional latency. For live Pro-audio applications, a dedicated hardware console is commonly used so that the low latency can be achieved and managed.

The audio specific network technology adopts protocols and architectures designed for transmitting audio signals which can achieve low latency up to a few samples such as AES50 [41]. However, the trend is to transmit audio over common network architecture such as Ethernet [42, 43] or Internet Protocol (IP) based infrastructure [44]. Audio processing can be part of a large multimedia system including audio, video and lighting. Audio signals are transmitted with various resolution requirements at different stages including mixing, broadcasting, and on-line streaming. Using a shared network infrastructure provides a better managed system and economical solution.

Multiple audio streams sources can be at different sampling rates. The sample rate conversion (SRC) is needed to convert them into the same sampling frequency for further processing, for example, to multiplex them at the same rate for transmitting over audio networking. In addition, downsampling and oversampling are very common functions for audio applications such as performing non-linear dynamic range compression at a higher sampling frequency to avoid unwanted aliasing noise. The SRC can cause latency when it needs to be done in a live situation.

Overall, these features sometimes cause negative effects on system latency. In some situations, the latency requirements are neglected. In others, the delay is used as one of the trade-off parameters of the system design, or is treated as an unpredictable value with certain statistical properties [9, 10].

2.3 Latency In Digital Audio System

Table 2.2 ([12, 26, 15, 45, 18, 19]) lists the typical latency values of each component in the audio processing chain. The latencies caused by various components are almost on a similar level of scale, except for using the Internet in uncontrolled conditions.

Table 2.2: Components latency in digital audio processing chain

Components		**Typical latency**	**Remark**
ADC (ΔΣ based)		0.25-1ms	Most ADC/DAC use cascaded linear phase filters causing the delay from fewer than 10 samples to about 50 samples. Higher sampling frequency and less stringent filter performance have the lower delay.
DAC (ΔΣ based)		0.25-1ms	
Mixer/DSP	DAW	5-10ms	DAW uses the block-based processing requires a buffer of 64 - 2048 samples typically. Hardware-based console works with sample-based DSP algorithms to reduce the latency down to a few samples.
	Console	0-2ms	
Networking	Pro-Audio	0.04-5ms	Complex factors affect the latency of networked audio. Controlled network tends to have a lower delay than open Internet. Commonly, the TDM based system has a smaller delay than the best-effort based. Uncompressed audio has a lower delay than the compressed codec.
	Internet	Up to a few seconds	
SRC		0.5-10ms	The sampling rate conversion scenario that requires a large number of up-sampling and down-sampling filters causes more latency.
Audio Plugins	Sample based	Group delay + a few sample delay (e.g. 0.08ms)	For software audio plugin, latency depends on the buffer of the software that conceals the algorithms buffer requirement. Hardware inserts introduce additional ADC/DAC delay.
	Frame based	Frame length * sampling frequency (e.g. 1024 samples at 48k Hz = 20ms)	
Processing Buffer		~ to 200ms	The system buffer is used to cope with kernel level scheduling variability. A typical value is between 10 to 200ms. Audio processing using pre-emptive real-time OS kernel features can have lower processing buffer delay.

The term ‘audio plugin’ has different meanings according to the context. People commonly refer it to the reusable software digital audio processing algorithms with agreed application programming interfaces (API). However, it may also refer to the standalone analogue or digital hardware devices which can be plugged in to the audio processing chain via hardware interfaces. Therefore the latency of these devices or software components can vary depending on the application. An analogue ‘audio plugin’ sometimes refers to a hardware ‘insert’. The latency of this type of plugin is similar to the latency caused by ADC/DACs.

For software plugins, the algorithm architecture affects the latency. For time domain sample based processing, the latency is affected by the minimum size of the buffer that it can achieve, for example 4-8 samples buffer in some designs, plus the group delay of the filters. If the algorithm uses frame based processing, the latency is normally larger. For example, the frame length of 512 to 2048 samples is not uncommon for plugins that do frame based processing, which is equivalent to 10ms to 40ms latency at 48 kHz sampling frequency.

2.4 System Latency

2.4.1 Latency Constraints

The maximum allowed latency in audio processing system varies between different applications.

- **One way streaming**. In this situation, the delay from sound source to end user can be hundreds of milliseconds to a few seconds such as most live broadcasting. In audio streaming over a packet switched network, the one-way delay can be at the magnitude of seconds, and still be regarded as ‘live’ [46]. The sound source does not need the feedback from the recipients. However, the recipients expect the uninterrupted audio signal once the event starts.
- **Interaction and music ensemble**. In conference calls, latency needs to be less than 250ms [6] to ensure the quality of conversation. For music ensemble over

networking or different studios that connected with audio system, the latency between performers should be equivalent to the time travel between them in a nature concert environments such as 8ms to 25ms (3 to 9 meters away) [47] [37].

- **Audio monitoring**. In the situation of audio monitoring and in-ear monitoring, musicians hear the audio from different paths. The direct path from sound source to human ears, and the processed audio from monitor speaker or in-ear headphones. The time difference between these two sources could make musicians uncomfortable depending on the type of performance. A comprehensive testing results in 2007 can be found at [36].
- **Binaural audio**. The interaural time difference (ITD) plays an essential role in the localisation of sound. In binaural audio processing, a few samples time difference between two channels will cause differences in the stereo image.
- **Multichannel mixing**. When mixing multichannel audio signals that have time differences, one sample difference will cause the comb filter effects. When recording a sound source with multiple microphones, the signals need to be aligned without delay before the mixing process. [40, 48, 49]

2.4.2 The System Latency Components

The typical values of components' latency in the modern digital audio processing system have been presented in Section 2.3 and detailed in Table 2.2. As discussed in Section 1.4, we focus on the three main components in the audio processing chain: (1) $\Delta\Sigma$ ADC/DAC; (2) Operating System (OS); (3) audio networking. The latency of $\Delta\Sigma$ ADC/DAC is caused by the group delay of high order filters that is represented as G_{di} in Eq. (1.1). The group delay of digital filter has wider implications than just in ADC/DAC. We will review it in Section 2.5. The OS delay is closely linked to the scheduling performance within a mixed criticality environments. It will be reviewed in Section 2.6. The current challenges and solutions of audio networking are discussed in Section 2.7. The delay caused by OS and networking delay appear as G_{bi} in Eq. (1.1).

2.4.3 GPOS Audio Processing Delay Test

Over the past 30 years, the increasing power of the personal computer drives the recent tendency towards using commodity computer based digital audio workstations (DAW) in live performance or recording environments. There are obvious advantages of using computers, which can be flexibly configurable with abundant software packages and plug-ins to replace or emulate some cumbersome hardware devices. With "High Definition Audio" being standardised as the onboard soundcard's hardware architecture for personal computers, and with advances in audio APIs, it is interesting to find out the latency of DAW based system.

Professional digital consoles normally have overall system latency no more than 2 ms. There are concerns that surround the use of computer based DAWs for low latency work (less than 10ms) due to unexpected jitters in sound when the CPU is heavy loaded [50]. The scale of the delay can beyond the perception thresholds (see Section 2.1 and Section 2.3). Therefore, professional audio interface cards provide hardware based monitor submixing or bypass routing for the purpose of offloading the CPU. It was indeed the audio community's initiative in 2000 that triggered the development of real-time Linux patch for Linux Operating System Kernel [51].

In 1998, researchers presented the results and discussed the causes of audio process latency of common operating systems [52]. It was suggested that the ideal latency time could be 3ms, and revealed the difficulties involved in achieving this. In 2001, research [53] indicated that the proper architecture of audio API stacks should keep the latency in the audio processing path constant without being affected by heavy CPU load tasks. The most promising low latency audio layers at that time were Linux ALSA (Advanced Linux Sound Architecture) and Mac OS X CoreAudio. At that time the tasks used in order to cause CPU load were not from audio dependent applications.

Audio driver architectures have evolved over the years, along with live audio applications and hardware platforms. The adaptive audio effects [54] and intelligent audio production [55] which use feature extraction to create control signals for the processing of sound have often been proven to have high computational cost, leading to heavy CPU loads. Most latency test of DAWs did not consider the latency of system under the

heavy CPU load from the audio processing itself [52, 53, 13]. With the appropriate side-chain design, multi-threading support from audio host platform and the concurrency of the software architecture, the hypothesis can be made that the intelligent subsystem and multiple audio processing paths should not affect the real time audio processing path even when the CPU load is coming from the audio application itself. The research that tested this was carried out by the author in [12]. The detailed testing plan and results are presented in Section 4. It revealed the problem of buffer underrun and unexpected error might link to the scheduling mechanism of GPOS that is used to carry out low latency audio processing. The related literature of OS scheduling is reviewed in Section 2.6.

2.4.4 Algorithm Latency

As described by J.A.Moorer in 2000 [4], we are in the 'supernatural' recording era. Audio and music can be extensively crafted in real-time or at the post processing stage. Digital audio effects [56] are the powerful tools being widely used in live audio production. Many digital audio effects applications have inherent latency due to the architecture of the algorithms. Audio engineers have concerns when using these audio effects with unknown latency. For plugin developers, it is important to estimate the accurate latency in order to report to the host applications. In general, there are three main sources of latency introduced by digital audio effects:

1. **Block based Processing**
   The time segment block or Fast Fourier Transform (FFT) block based processing can be problematic for low latency implementations. When the time frequency processing or the computational efficiency is needed, time segmentation and Short-Time Fourier transform (STFT) are the common approaches employed. Researchers in this area use sub-band (filter bank) approaches or mixed time/frequency domain methods to reduce latency.

   For example, high quality synchronised overlap-add method (SOLA) based pitch shifting uses typical 2046-8192 samples block, at 44.1kHz, corresponding to about 46 to 186ms. [20] has proposed a pitch shifting algorithm based on using a large number of time-domain filter banks to reduce the latency to under 10ms with ac-

ceptable quality.

Digital convolution reverberation based on impulse responses is a widely used effect. The impulse response can be as long as a few seconds, so the implementation using FIR convolution is not computationally practical for real-time applications. However, the FFT based algorithm introduces noticeable latency. In 1995, Gardner [22] proposed an efficient convolution method without input/output delay. Since then, this method has been continually improved by researchers and the algorithms based on this have been wildly used in audio production as well as in virtual game audio effects.

2. **Phase Delay in DSP Algorithm**
   The phase delay can be obtained from the phase response of linear time invariant (LTI) system. It is defined as $P(\omega) \triangleq -\frac{\Theta(\omega)}{\omega}$. The latency of the signal is closely linked to group delay which can be defined as the derivative of phase response $D(\omega) \triangleq -\frac{d\Theta(\omega)}{d\omega}$ [57]. When the phase response $\Theta(\omega)$ is a linear function of frequency, the group delay becomes a constant. Therefore for a linear phase Finite Impulse Response (FIR) filter, it is agreed that the latency d introduced by the FIR filter can be expressed as $d = (N-1)/2$, where N is the filter length of the coefficients. The common minimum phase FIR or Infinite Impulse Response (IIR) filter used in audio processing normally has smaller group delay but with non-linear phases. Most researchers have concentrated on reducing group delay of FIR filter design where the latency problem is more prominent [58, 17, 59, 60]. Group delay of digital filters is also the major cause of delay in high performance $\Delta\Sigma$ ADC/DAC, which will be reviewed in detail in Section 2.5.

3. **Architecture Delay**
   Architecture delay is the latency introduced by the implementation structure of algorithms. A typical example is the side-chain based dynamic range compressor with a look ahead buffer. In real-time processing the audio stream needs to be delayed by the same amount as the look ahead buffer [56]. In more advanced Adaptive Digital Audio Effects (ADAFX) [54], if using a side-chain mode, then synchronisation between the feature extraction and the audio processing stream can cause delay. Multichannel based cross-adaptive audio effects can be used in

automatic mixing [61, 55]. For most real-time automatic mixing tasks the control decision is made on side chain after convergence. The current implementations [62, 63, 64, 65] do not need to synchronise the feature extractions with the audio processing at the fine granularity level.

One obvious approach to deal with the synchronisation problem in ADAFX is to develop real-time feature extraction with low latency. Some audio feature extractions can be implemented in real-time [66, 67]. Most of them use STFT based techniques to estimate frequency information, in which cases, the latency problem is similar to the block based processing. A possible solution is to use a probabilistic model to predict future feature events based on historical empirical data [68].

In summary, DSP algorithm delay can be determined and reported if the accurate buffering and group delay information can be obtained. To design the low latency version of the algorithms that perform the equivalent audio processing effects and functions is beyond the scope of this research.

2.5 Delay of Digital Filters for High Resolution Audio Conversions

2.5.1 Delay of $\Delta\Sigma$ ADC/DAC

For digital filters used in high-resolution audio conversions such as the decimation or interpolation filters in professional $\Delta\Sigma$ ADC/DAC and SRCs, the performance of the filters needs to satisfy specific resolution and signal to noise ratio (SNR) requirements, for example greater than 24 bits bit resolution and higher than 120dB SNR. These requirements usually result in high order filters that have significant group delay.

Currently, the most popular audio range ADC/DAC architecture is the multi-bit $\Delta\Sigma$ ADC/DAC based converter [69, 70]. Most commercial audio converters use multi-bits rather than the 1bit type as the 1bit type is hard to dither properly [71, 72]. The oversampling structure of $\Delta\Sigma$ modulator based converter needs to use interpolation and decimation filters to convert from or into PCM stream [73]. These filters are commonly

implemented using cascaded linear phase FIR filters. Although using other filters is also possible [74]. In [14], Feerick gave a typical latency of 660μs at 48kHz sample rate.

$\Delta\Sigma$ ADC/DAC and SRCs for live sound also require low computational complexity as they are commonly implemented in compact hardware form. Therefore the associated filters are usually designed to be multistage. Each stage within the multistage system utilises a half-band or N-band linear phase filter when possible, in order to reduce the computational and implementation cost further. However, these structures often worsen the overall latency. The quantitative result of how these structures affect the overall latency is not available. To analytically quantify this effect is one of our objectives in this work.

Although digital SRC works entirely in the digital domain, it has similar digital filter structures as ADC/DAC [75]. It commonly involves oversampling and downsampling processes in multiple stages. For example, to convert sampling rate from 44.1k HZ to 48k HZ, the processing can be up-sampling to 7.056 MHz then downsampling to 48k Hz, or convert 44.1k Hz sampling rate at the ratio of 10:7, 8:7, and 2:3 to get 48k Hz. The alternative method is using farrow structure based fractional delay filters [76]. Asynchronous Sample Rate Converter (ASRC) using a FIFO buffer, multiple FIR filters, and interpolation to convert arbitrary sampling frequencies that derived from a different clock source [77, 78, 79]. [14] gives a typical 1-3ms latency of each conversion.

To understand the delay effects of the different filter structures, especially the cascaded FIR filters used in various audio signal conversion, it is worth to review the fundamental theory of digital filters including FIR, IIR, linear phase, minimum phase, multistage and multirate filters.

2.5.2 Group Delay Formula

The transfer function of a digital filter can be described as a function of z^{-1}:

$$H(z) = Gz^{-n_0}\frac{\prod_{i=1}^{N}(1 - z_i z^{-1})}{\prod_{i=1}^{M}(1 - p_i z^{-1})} \tag{2.1}$$

In Eq. (2.1), G is the overall gain. The denominator and numerator are the polynomial

of z^{-1} in forms of factorisation. z_i and p_i are the zeros and poles of the filter. The order of the filter is $max(M, N)$. n_0 is the pure delay.

The temporal behaviour of a signal passing through digital filters can be described analytically by the phase response of the system. The "group delay" is commonly used to quantify the "average" time lag between input and output signals. Officially the group delay is defined as "the derivative of radian phase with respect to radian frequency." [57]

$$\tau_g = -\frac{d\Theta(\omega)}{d\omega} \tag{2.2}$$

Let $z = e^{jw}$, and we express the poles and zeros in polar form: $z_i = r_i e^{j\theta_i}$, $p_i = \rho_i e^{j\varphi_i}$. We can have the group delay expressed as the function of polar form of zeroes and poles:

$$\tau_g(\omega) = n_0 + \sum_{i=1}^{N} \frac{r_i^2 - r_i \cos(\omega - \theta_i)}{1 - 2\mathrm{r}_i \cos(\omega - \theta_i) + r_i^2} - \sum_{i=1}^{M} \frac{\rho_i^2 - \rho_i \cos(\omega - \varphi_i)}{1 - 2\rho_i \cos(\omega - \varphi_i) + \rho_i^2} \tag{2.3}$$

2.5.3 Delay of Non-Recursive Linear Phase (LP) Digital Filters

For a non-recursive (FIR) filter with order N, Eq. (2.1) can be written as Eq. (2.4):

$$H(z) = \prod_{i=1}^{N} (z - z_i) = z^N + h_1 z^{N-1} + h_2 z^{N-2} + ... + h_N \tag{2.4}$$

or its difference equation form as:

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + ... + b_N x[n-N] \tag{2.5}$$

Where $y[n]$ represents output samples and $x[n]$ represents input samples. The coefficients b_n can be also treated as the "impulse response" of the FIR filter. The intuitive way to implement FIR filter is to convolve the input signal with the "impulse response".

Figure 2.2 shows a typical Linear Phase (LP) FIR filter coefficients with 42 order (so the number of coefficients is 43). The coefficients of the LP FIR filter normally follow a symmetric or antisymmetric pattern.

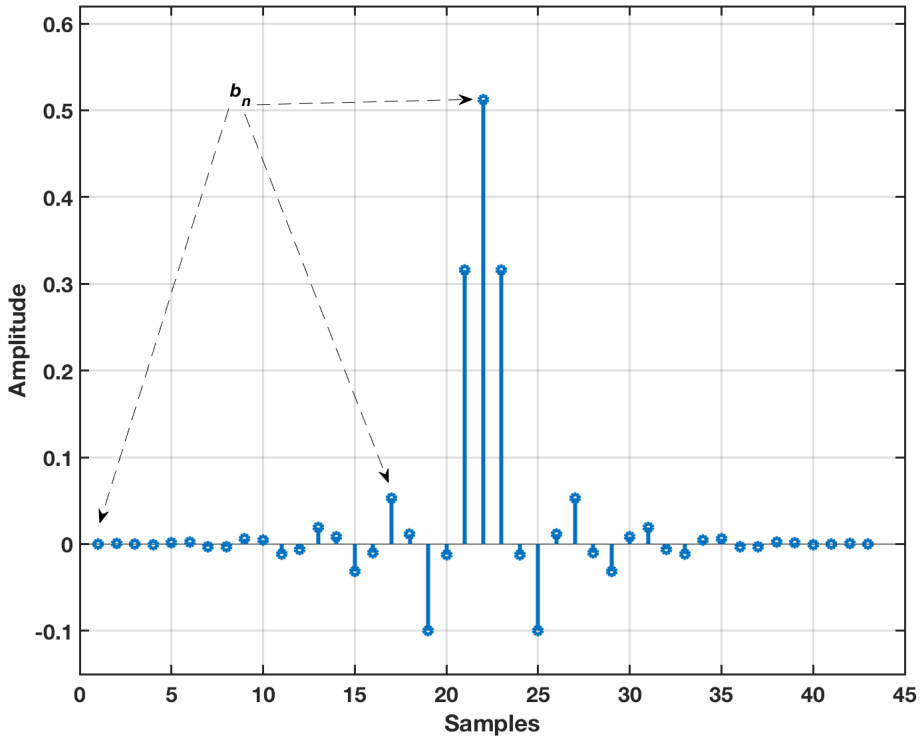


Figure 2.2: Typical linear phase FIR coefficients

[80, 81] prove that a necessary condition for having a linear phase filter is to have non-recursive, transversal structure (mirror image polynomial or negative mirror image polynomial) i.e. satisfying $h(n) = \pm h(N - n)$, the group delay is constant as $N/2$. Most Windowed-Sinc based FIR filter design methods should yield linear phase due to the symmetric property of the polynomial coefficients.

Eq. (2.4) can also be written in a factorised form as in Eq. (2.6), where z_i is the roots of the polynomial Eq. (2.4). In many cases, the closed form of roots of coefficients *h(n)* do not exist for the order of the polynomial is greater than 5 (Abel's impossibility theorem).

$$H(z) = \prod_{i=1}^{N} (z - z_i) \tag{2.6}$$

The roots of $H(z)$ in Figure 2.2 can be plot in complex domain as follows:

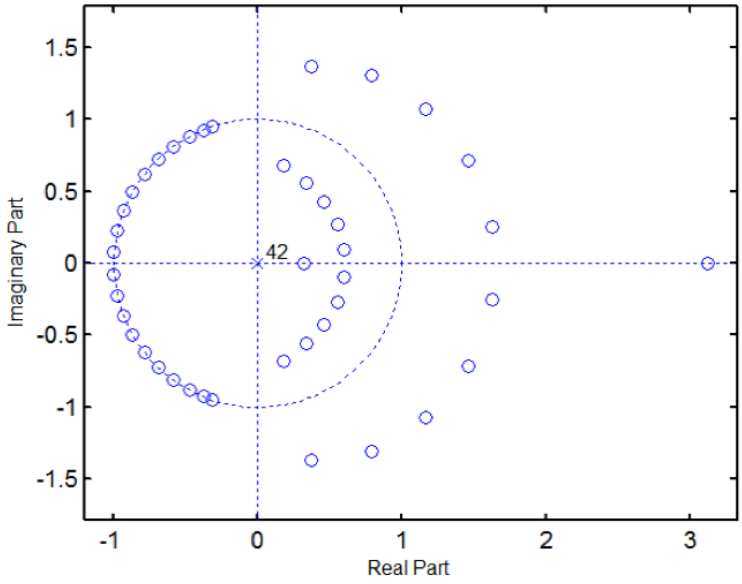


Figure 2.3: Roots of H(z) of LP FIR filter

As Eq. (2.2) shows the group delay τ_g is a function of phase response $\Theta(\omega)$, which is defined as the phase or angle of frequency response:

$$\Theta(\omega) \triangleq \angle\left(H\left(e^{j\omega}\right)\right) \tag{2.7}$$

Also the magnitude response of this filter is expressed as the absolute value of the transfer function and the function of ω:

$$G(\omega) = |H(e^{j\omega})| = \left|\prod_{i=1}^{N} (z - z_i)\right| \tag{2.8}$$

The following diagram shows the magnitude response and group delay in the same figure. The group delay is constant, which is 21 samples in this case.

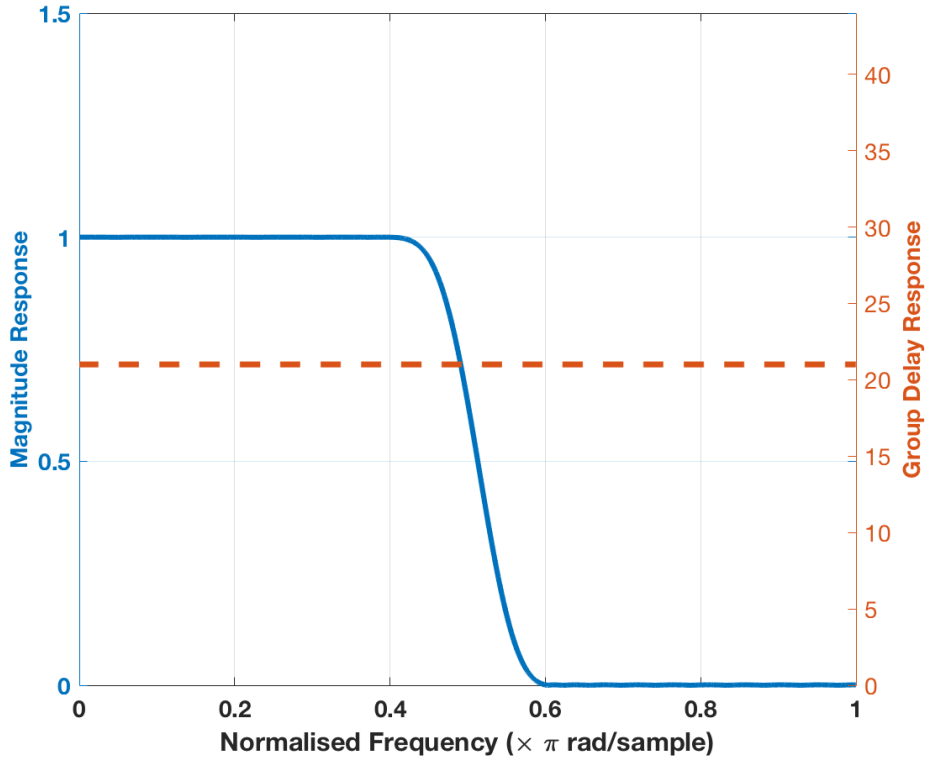


Figure 2.4: Magnitude and Group Delay response of a 42nd order Linear Phase Filter

It is easy to show that a polynomial with symmetric coefficients will have group delay independent of ω.

In summary, the delay caused by a LP filter is proportional to the order of the filter. The filter used in high resolution audio conversion typically has a very high attenuation ($>$120dB) and narrow transition band, which results in very high order (over thousands long), thus not only high latency but also practically very expensive to implement. Therefore the multi-stage filter design is normally employed to reduce the number of order by a factor of ten. However, although the order can been reduced, the overall delay of the multistage filter is greater than equivalent single stage filter due to the stages are distributed towards the lower end of the sampling frequency [27]. This effect is discussed in details in Chapter 3.

2.5.4 Filter Order Estimation Methods

Not only the filter order affect the delay behaviour of both LP and Minimum Phase (MP) types of filters, but also is the key parameter of multistage multirate filter design technique. It is worth to review the various filter order estimation methods.

According to J. F. Kaiser (1974) [82], the order of the filter design can be estimated by the following:

$$N \cong \frac{-20\log_{10}(\sqrt{\delta_p \delta_s}) - 13}{14.6(f_s - f_p)/2} + 1 \tag{2.9}$$

Where:

- δ_p is the tolerance in the magnitude response in the passband.
- δ_s is the tolerance in the magnitude response in the stopband.
- f_s is the normalized passband edge frequency.
- f_p is the normalized stopband edge frequency.

According to Crochiere (1973) and Rabiner (1975) [83, 84], the order of the filter can be estimated by the following equation:

$$N \cong \frac{D_\infty(\delta_p, \delta_s)}{\Delta F} - f(\delta_p, \delta_s)\Delta f + 1 \tag{2.10}$$

Where ΔF is the normalised transition band, and D_∞ can be calculated by the following empirical function according to:

$$\begin{aligned} D_\infty(\delta_p, \delta_s) = [5.309 \times 10^{-3}(\log_{10}\delta_p)^2 + 7.114 \times 10^{-2}(\log_{10}\delta_p) - 0.4761]\log_{10}\delta_s \\ -[2.66 \times 10^{-3}(\log_{10}\delta_p)^2 + 0.5941(\log_{10}\delta_p) + 0.4278] \end{aligned} \tag{2.11}$$

And

$$f(\delta_p, \delta_s) = 0.51244\log_{10}(\delta_p/\delta_s) + 11.01217 \tag{2.12}$$

For typical high resolution oversampled audio conversion filter design according to [85], the reciprocal of the transition band is over 300. Therefore the single stage filter will result in an order more than 1500. The order estimation methods can produce the value that is close to the result of optimal design methods. The classic optimal multistage design is based on the order estimation functions on Eq. (2.10) to Eq. (2.12). The optimal order and estimated order almost linearly increase as the transition band reduces. However, it is worth noting that the order estimation is an approximation and there will be differences when the filters are designed and realised.

2.5.5 The Multirate Multistage Filter System

In some audio applications, the filters require very high stopband attenuation, very small passband ripple and a narrow transition bandwidth. For example, in case of the high performance anti-aliasing or anti-image FIR filters, the transition band can be less than 0.01 (normalised frequency). These types of digital filters can be found in $\Delta\Sigma$ ADC/DAC that are used for anti-aliasing and anti-image purpose as well as extracting the PCM data from lower resolution (e.g 1 bit DSD) and oversampled bit streams. For high-resolution audio applications, the typical filter specifications are below [86, 87, 74]:

1. Small passband ripple $<$0.0001
2. High attenuation at stopband $>$100 dB
3. Small transition band less than 1/300 due to the oversampling

To achieve this design specification with a single stage FIR filter, for example, a typical 64x oversampling rate design for 48 kHz the f_s is 3.072 MHz, and the transition bandwidth is 4.8k Hz, the order N can be over 2300, which is not only high in group delay but also unrealistic for hardware implementations.

The multistage digital filter design technique can be used to reduce the filter order. Optimal design techniques have been developed since the 1970s up to today [84, 88, 89, 90, 91]. In a multistage design, the overall design specification can be satisfied by cascading a number of multirate filters.

Although the classic multistage filter design method reduces the overall order of the

filter, it often worsens the overall group delay. For a single stage linear phase filter, the group delay is equivalent to the half of the filter order. The order reduction of a multistage filter design does not reduce the group delay. This is because the multistage filter needs to work at different sampling frequencies for each stage. Hence the reduced order at lower sampling frequency has higher delay weight which overwrites the effects of the delay saved by the reduction of the order.

The filter order estimation method plays the central role in designing an optimal multistage filter. The classical 'optimal' methods refer to the computational and area costs of the filter realisation. The optimum design based on order estimation is firstly proposed by Crochiere [84, 88]. Coffey [89, 90] also shows the optimisation design can be reduced to a one-dimensional roots finding problem for both computational and area cost.

For example, a "k" stage decimation filter can be depicted as in Figure 2.5:

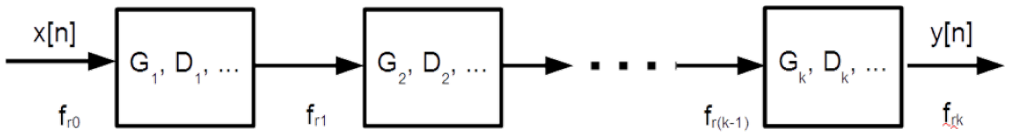


Figure 2.5: Multistage filter structure

Where $D_i, (i = 1, 2, 3...k)$ is the decimation factor of stage 'i', G_i represents the group delay of each stage. Therefore

$$f_{ri} = \frac{f_{r(i-1)}}{D_i}; \text{where } (i = 1, 2, ...k) \tag{2.13}$$

$$D = \prod_{i=1}^{k} D_i \tag{2.14}$$

Figure 2.6 is a typical example of 3-stage (k=3), 64 times (64x) decimation filter which alters the input sampling frequency from 3.072MHz to 48k Hz.

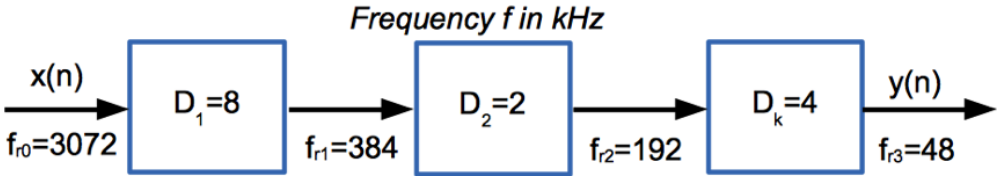


Figure 2.6: Example of 64x multistage decimation filter

Crochiere's method is to find the optimal design of such structure. The optimisation is in terms of computational cost as total number of Multiples and Adds (MADS). In his work, the R_t is defined as the total cost, and R_i is defined as the cost of the stage i. Therefore:

$$R_t = \sum_{i=1}^{k} R_i \tag{2.15}$$

$$R_i = \frac{N_i f_{r(i-1)}}{D_i} \tag{2.16}$$

Where N_i is the order of stage i. This order is evaluated using order estimation functions in relation to design specifications as in Eq. (2.10) to Eq. (2.12). Therefore the final objective function can be found in Eq. (2.17) and Eq. (2.18), where $\Delta f = (f_s - f_p)/f_s$.

$$R_t \cong D_\infty \left(\frac{\delta_p}{k}, \delta_s \right) f_{r0} S \tag{2.17}$$

$$S = \frac{2}{(\Delta f \prod_{j=1}^{k-1} D_j)} + \sum_{i=1}^{k-1} \frac{D_i}{(\prod_{j=1}^{i} D_j)(1 - \left(\frac{2-\Delta f}{2D}\right) \prod_{j=1}^{i} D_j)} \tag{2.18}$$

This final objective function Eq. (2.17) can be solved by a computer-aided optimisation routine when k is greater than 2. Coffey also proved Eq. (2.17) and Eq. (2.18) can be simplified as a root finding problem. Both Crochiere and Coffey show the optimal design can be achieved by the number of stage as 3 or 4. The total order of multistage filter is reduced by the factor of ten in comparing with the single stage design for the same design specifications.

Other important computationally efficient techniques include "the cascaded integrator-comb (CIC) filter structure" [92], and "half-band or N-band filters" [93, 94, 95], which can be used in some special cases of multistage design. The polyphase filter structures [96] have been widely adopted as effective implementations in multirate signal processing, including decimation and interpolation. However it is not so clear that how these cost efficient methods affect the overall latency.

2.5.6 Delay of Linear Phase Multistage System

Figure 2.5 shows the structure of the 'k' stage multistage filter with input signal $x[n]$, output signal $y[n]$. The sampling frequency of input signal is f_{r0} and the sampling frequency of output signal is f_{rk}. 'D' is the sample rate alteration factor such as decimation or interpolation factor. 'G' is the delay of each stage. The overall latency can be simply regarded as the summation of latency of each stage as shown in Eq. (2.19).

$$G(\omega) = \sum_{i=1}^{k} G_i(\omega) \tag{2.19}$$

Due to the design duality of decimation and interpolation processes [84], we can treat this sample rate alteration factor as either integer decimation or interpolation to analyse the overall latency.

The multistage filter structure has negative effects on the overall latency in comparison with single stage filter satisfying same magnitude response performance. The comprehensive evaluation is done by author in 2012 [27].

However, this effect is not shown quantitatively in the literature. Notably the number of stages and the alteration factor of each stage are adjustable. The hypothesis can be made that in the multistage linear filter system, the overall delay can be expressed as the function of design parameters and it can be optimised towards. One of the important objectives of this work is to prove this hypothesis using analytical approach. Furthermore, to understand the trade-off between delay and other filter performance measures.

2.5.7 The Minimum Phase System

Minimum Phase (MP) digital filters are the typical cases of minimum phase system. In control theory and digital signal processing theory, a system is MP if all poles and zeroes of the system are inside the unit circle. MP system has minimum phase lag and minimum energy delay properties. In addition, given a specific magnitude response with MP, the original system is uniquely defined. The MP filters can be used for low latency applications [97, 98] including audio converter application [99]. However, there is still lack of qualitative or quantitative analysis of the delay of linear and minimum phase behaviour of multistage filters with various design variables including the filter specifications and multi-stage design parameters, such as number of stage and associated decimation factors.

2.5.7.1 Minimum Phase Filter Group Delay Distortion and Hearing Threshold

Another key issue of MP filter is the non-linearity of group delay, for example the filter responses in Figure 2.10. This non-linearity contributes to group delay distortion. It would be nice to know the threshold of audibility of group delay with respect to frequency, but this remains an area where not a great deal seems to have been done. No extensive data is available and so far, the best table is from Blauert and Laws [100] and Bloom and Preis [101], that are summarised in Table 2.3.

Table 2.3: Group Delay Audibility Thresholds

Frequency	**Threshold**
500Hz	3.2 ms
1kHz	2 ms
2kHz	1 ms
4kHz	1.5 ms
8kHz	2 ms

Figure 2.10 shows the group delay plots of a MP filter and a LP filter both with 42 order.

The group delay distortion caused by a MP filter within the audio band is mainly affected by the selection of corner frequency and filter order. The corner frequencies are part of the design specifications, whereas the order of FIR filter is closely linked to the user design specifications. The filter order plays an important role in affecting both linear phase FIR filter delay and the delay distortion of minimum phase FIR filter.

2.5.7.2 The Minimum Phase FIR System

Both IIR and FIR filters can be minimum phase. In this work we concentrate on the minimum phase FIR system due to the following reasons.

1. The IIR filter and its analogue counterparts have been widely used in audio applications already, for example, shaping the frequency response in an EQ. Most of these applications do not need high order filters so latency is not a significant problem.
2. High order FIR filters with significant latency are commonly used for data conversion and sample rate conversion applications which require separation of different frequency components, because they can achieve a very sharp roll-off and close to brick wall magnitude response.
3. The FIR system can be effectively used in hardware implementation without sophisticated overflow control, round off error estimations, and zero input limited cycle problems. The pure feed forward structure of FIR filter can be utilised for simplifying the 1-bit $\Delta\Sigma$ modulated signal filtering task very efficiently.
4. The minimum phase FIR filter properties and design techniques are challenge in theory. The work in this area can be found from 1970s to present [102, 58, 103, 104, 105, 60]. Although there is abundant literature regarding general and special digital filter design techniques, there is a lack of a system framework to enable qualitative and quantitative time domain driven approach for MP multistage FIR system for high-resolution audio signal processing. To achieve the framework either analytically or numerically, there are many unsolved research questions which makes the domain itself attractive for researchers.

2.5.7.3 Delay of Minimum Phase Digital Filters

A minimum phase system is the system that all its zeroes are within the unit circle. Among all causal filters $h_i(n)$ having identical magnitude spectra, the minimum-phase filter $h_{mp}(n)$ has the fastest decay in the sense that the signal energy is maximally concentrate toward time zero [57], the proof can be found in [86] :

$$\sum_{n=0}^{k} |h_{mp}(n)|^2 \geq \sum_{n=0}^{k} |h_i(n)|^2 , \; k = 0, 1, 2, ... \tag{2.20}$$

The linear phase (LP) filter can be converted into minimum-phase (MP) filter while maintaining the same magnitude response by flipping the zeroes from outside unit circle into inside unit circle. To have the same magnitude response, the zeroes of filter polynomial are exchangeable with their reciprocal pair. For example, if the original polynomial has zero at $r_k e^{j\theta_k}$ outside the unit circle then, if we replace it with $\frac{1}{r_k} e^{j\theta_k}$, the magnitude response should be the same as the original one. The reciprocal pair is shown in Figure 2.7.

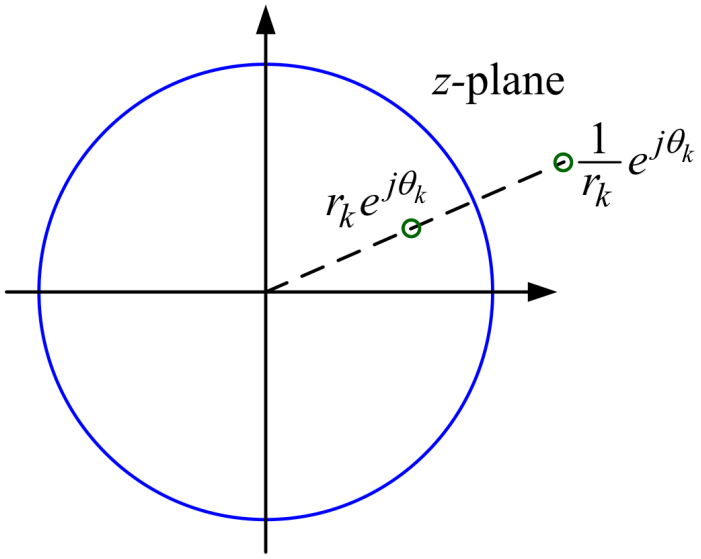


Figure 2.7: Zero and reciprocal zero

A new set of filter coefficients $\widehat{h_i}$ can be found to match a given magnitude response, but with a minimum phase design. This can be achieved using the following approach:

Let

$$H(z) = \left[\prod_j (z - r_{jin})\right]\left[\prod_j (z - r_{jout})\right]\left[\prod_j (z - r_{jo})\right] \\ = H_{min}(z)H_{max}(z)H_o(z) \tag{2.21}$$

Where $H_{min}(z)$ are the zeroes within unit circle, $H_{max}(z)$ are the zeroes outside unit circle, and $H_o(z)$ are the zeroes on the unit circle. Determine $H_{max}(z)$ and reflects its zeroes into the unit circle. Because of the conjecture and reciprocal property of linear phase polynomial roots, the new transfer function $\hat{H}(z) = H_{min}(z)^2 H_o(z)$ can be determined.

If we reflect the roots outside unit circle into inside unit circle as described above for the 42 order filter we used in previous example as in Figure 2.2, we will have the root map as in Figure 2.8.

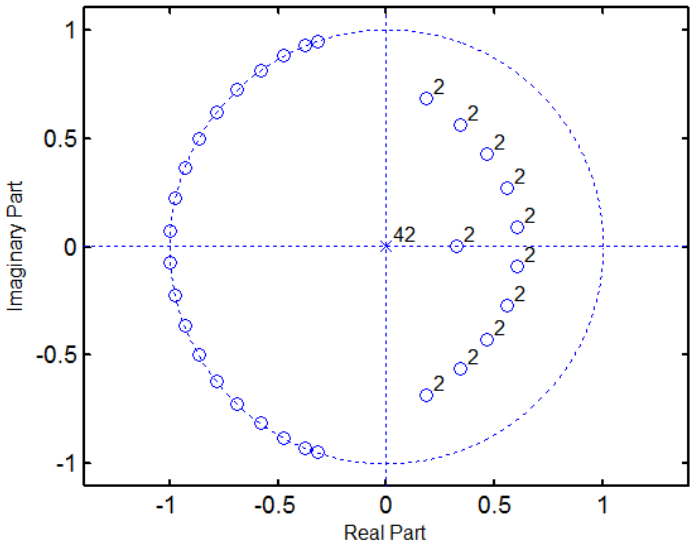


Figure 2.8: MP Filter designed by reflecting zeroes outside unit circle

By doing that, we create a filter with exact same magnitude response as the original.

However, in this case, it is a minimum-phase filter. The time domain behaviour of this filter should be more compact compared with LP FIR filter.

The kernel of the new filter $\hat{H}(z) = H_{min}(z)^2 H_o(z)$, is plotted in Figure 2.9. It can also be regarded as the impulse response of the minimum phase filter. Figure 2.9 demonstrates this impulse response has high energy pulses close to time zero.

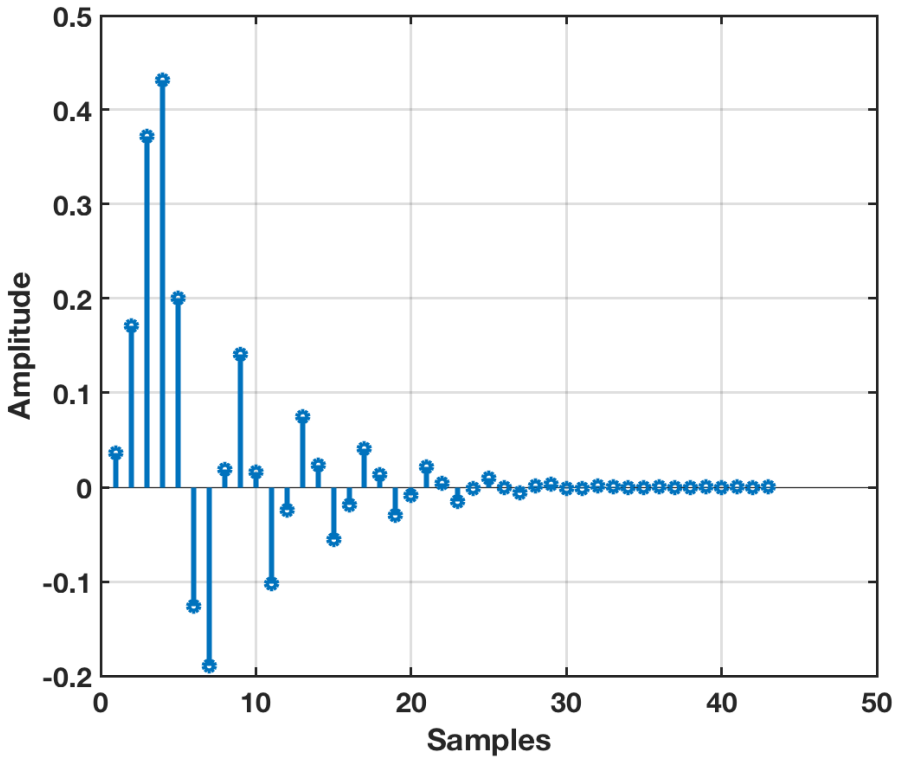


Figure 2.9: Transformed Minimum Phase FIR coefficients

The generalised group delay of the MP filter as a function of frequency is expressed as following:

$$\tau_g(\omega) = \sum_{i=1}^{N} \frac{r_i^2 - r_i \cos(\omega - \theta_i)}{1 - 2\mathrm{r}_i \cos(\omega - \theta_i) + r_i^2} \tag{2.22}$$

From Eq. (2.22), we can see that the group delay can be expressed as the polynomial of cosine function of angular frequencies θ_i and the vector of roots r_i.

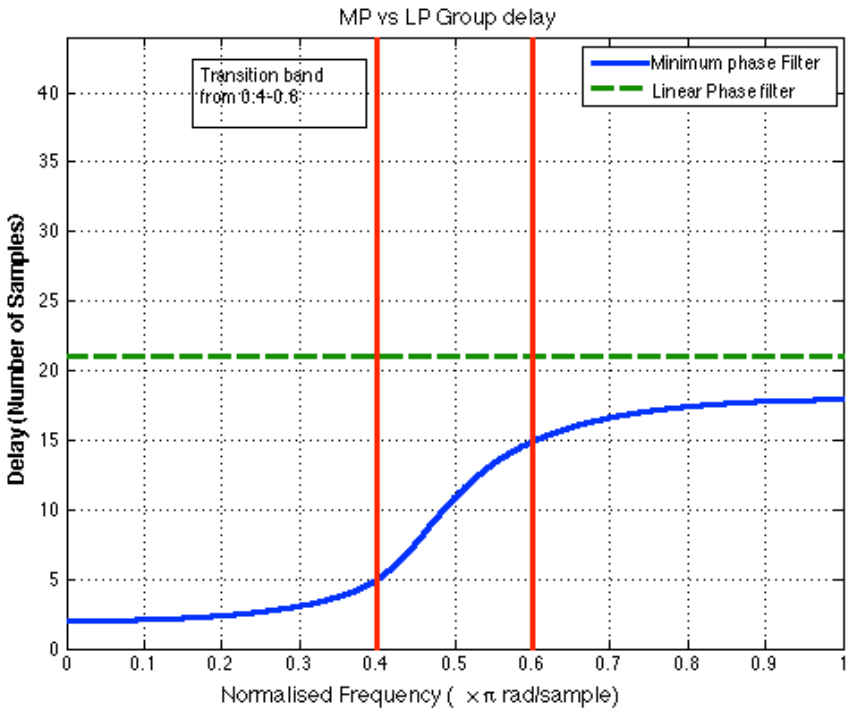


Figure 2.10: Group delay of LP and MP filters

Figure 2.10 shows the group delay vs. frequency curve of linear phase and minimum phase lowpass filters. It clearly shows that the minimum phase design has significantly lower delay with passband for this lowpass filter.

If a minimum phase lowpass FIR filter is designed with stipulated filter performance, Can we describe the group delay behaviour function of this filter in mathematical "closed form" as a function of design specifications such as:

$$G_i(\omega) = f_2(\omega_{si}, \omega_{pi}, \delta_{pi}, \delta_{si}, \omega) \tag{2.23}$$

Also it would be convenient for the expression of Eq. (2.23) to be concise and intuitive for helping further multistage analysis.

There are challenges of achieving that. The group delay cannot be expressed concisely when the transfer function *H(z)* is in the direct form as in Eq.(2.4). It could be expressed

if the filter is in polar form as in Eq. (2.22). However, most popular FIR filter design methods are based on optimisation which involves some sort of iterative process, so the filter transfer functions cannot be expressed directly by the design specifications. There are direct FIR filter design methods such as 'Kaiser window', 'Frequency sampling', and 'Maximally Flat FIR design' methods. However the results cannot help express group delay as a closed form of function of design parameters.

The closed forms, if they exist, are fairly complicated to express. The high order coefficients cannot translate to locations of zeroes in the Z-plane without the numerical approach. The classical root finding and spectrum lifting procedures are based on LP prototypes so the direct form of LP design methods can also further develop to the MP filter design methods. However they share same factorisation problem. There are other methods such as Cepstrum and root moment based approaches [106, 104]. To the best knowledge of the authors, none of them provides the direct form between design parameters and zeroes locations of filters. For the filter with higher order (greater than 5) which normally is the case for audio sample rate conversion applications, the numeric method of finding roots is needed unless the coefficients in some special form. This makes closed form group delay expression difficult to find.

Therefore, in this research, we try to quantify the MP filter properties based on design specifications and the experiments results. The detailed work is in section 3.4.1.

2.6 Operating System Scheduling Latency

In this section, we review the delay caused by the GPOS for audio processing. For real-time audio processing, the incoming samples are commonly buffered and processed in a block based manner. It is to reduce the processing and the scheduling overhead in order to prevent the buffer underrun or overrun, which can cause glitches and distortion of audio [50]. Using a high buffer setting is a safe option for processing audio to avoid unexpected glitch and noise. In the most recent audio latency test work by the author in [12], which is detailed in Section 4.1, it shows that one has to use a trial and error method to identify the minimum buffer that each system can endure.

The reason for using buffer is to accommodate the non-deterministic processing and to reduce the computation requirements of GPOS based DAWs. The scheduling algorithm of a modern operating system can be modified to support better soft real-time application tasks via priority based pre-emptions. In our test in 2010 [12], the Linux OS with a real-time feature enabled in the kernel configuration performs better than other counterparts.

In [107], William Clark from RedHat performed comprehensive testing on the system scheduling latency of different scheduling policies for the Linux kernel. The results are summarised in Table 2.4 to Table 2.6. The three tables represent three different Linux kernel configurations. Table 2.4 shows the results of vanilla kernel without the pre-emption patch. Table 2.5 and Table 2.6 show the results of the Linux kernel with two different versions of low latency pre-emption patches. Over millions repetitive testing tasks are driven by Real Time Clock at 2kHz frequency, the left column of each table shows the percentage of tasks that finish within the time of the values in the right column. The unit of the time is millisecond.

Table 2.4: Case 1 vanilla 2.4.17 kernel

Percentage	Delay (ms)
92.84442	0.1
97.08432	0.2
99.73050	0.5
99.84382	0.7
99.94038	1
99.97922	5
99.98096	10
99.98590	50
99.98828	100
100	232.7

Table 2.5: Case 3 low-latency 2.4.17 kernel

Percentage	Delay (ms)
96.66250	0.1
99.21158	0.2
99.99592	0.5
99.99984	0.7
99.99992	1
100	1.4

Table 2.6: Case 2 preempt 2.4.17 kernel

Percentage	Delay (ms)
97.95326	0.1
99.55722	0.2
99.97026	0.5
99.98960	0.7
99.99650	1
99.99954	5
99.99982	10
100	45.3

[107] also tested the scheduler delay of combined low latency patch with pre-emption patch which showed similar testing results with slight improvement. Overall, this test showed real-time patched Linux system can achieve 100% scheduler latency lower than 1.3ms statistically at the time.

However, the scheduling delay happens in a non-deterministic manner. It revealed the difficulty of setting up a deterministic buffer for GPOS, even with soft real-time features enable. In addition, the variation of delay is larger than the common audio sampling period. It shows the difficulties to use GPOS to carry out sample based processing.

2.6.1 GPOS vs RTOS for Audio Processing

There is a trend to use DAW to perform live music on stage, or as the mixer for a front of house (FOH) system, or as the mixer for in-ear monitoring, because of the versatile functionalities, flexible configurations, and expandability of DAWs. On the other hand, the audio processing becomes increasingly multifunctional and complex. It incorporates with the advancement of computation, DSP and networking technologies such as

high resolution multichannel audio processing, feature extraction, machine learning and intelligent audio production [55].

Using a DAW in a live or interactive environment is constrained by the responsiveness of the input and output audio signals. Systems are restrained by interactions of the computational components with the physical processes and are commonly referred to as Cyber-physical system (CPS) [23]. An example of CPS is autonomous vehicle where the system needs to response to multiple sensor signal input at low latency. The real-time and low latency issue of an audio system shares similar fundamentals as CPS.

DAWs are the software systems running on a General Purpose Operating System (GPOS). Though it is widely accepted that comparing with real-time system, GPOS does not have time constraint or time criticality. There is a trend of using GPOS within a mixed criticality system. GPOS has already been widely used in various pseudo-real-time, and soft real-time applications, such as multimedia live streaming. An OS has to deal with a different level of time criticality is called Mixed-Criticality (MC) system [108, 109, 110]. It is worth noting the concept of MC implies some trade-off between isolation and integration on resource sharing, whereas systems that solely focus on isolation of tasks are regarded as multiple-criticality systems [110]. It is commonly acknowledged that for a MC CPS, one of the key challenges is to report accurate timing parameters from the bottom up [23].

Real-time systems are the systems that not only perform computation with logic correctness but also timing correctness [111] [112]. There are classic real-time schedulers such as Cyclic Scheduling, Rate-Monotonic Scheduling (RMS) and Earliest Deadline First (EDF). However, there is a trend to have real-time extensions to GPOS such as real-time Linux, due to the wide adoption of using open source GPOS in the industrial area. Most modern telecommunication platform and professional live console have embedded Linux with real-time extension in them to ensure the real-time performance [113]. Also, most commodity GPOS such as Linux, Windows or MacOS all have some hierarchical scheduling scheme that enables real-time tasks being executed with minimum jitter and latency. The latest Windows OS has six different priority classes. Mac OS has four different priority bands. Linux by default uses Completely Fair Scheduler (CFS), but it can be configured to use real-time scheduling policy such as First in First

Out (FIFO), Round Robin (RR) and Earliest Deadline First (EDF) with the number of different priority levels of 99.

However, these modifications may not work for the hard real-time system due to the non-deterministic attributes of the file system and device drivers such Virtual File System (VFS) framework. That's why traditional Real Time Operating System (RTOS) avoid using a file system. To be able to support hard RT in GPOS, one has to rewrite all the file system interfaces and device drivers, such as the proposed work from CMU's RT Mach and RT file system [114].

There are fundamental differences between RTOS and GPOS from the design point of view: RTOS is optimised on the worst case, whereas GPOS is optimised on an average case. RTOS targets predictable scheduling whereas GPOS targets efficient scheduling. RTOS has simple executive whereas GPOS provides wide range of services. RTOS tries to minimise the latency whereas GPOS tries to maximise the throughput. The earlier work of hard RT Linux proposed by Yodaiken et al. was done by replacing all 'cli', 'sti' and 'iret' [1] to the soft interrupt macros and using hardware triggered interrupts to execute hard RT tasks [115]. However, these hard real-time tasks have no access to Linux kernel services at the time. A similar but more sophisticated approach is to run dual kernels such as Xenomai system with RTOS kernel along with a general Linux kernel. The carefully mapped audio tasks can run on such a system with the low latency [116, 117].

Other effort [118] tried to emulate RTOS within GPOS to provide soft RT performance. [118] predicts the GPOS will be more popular for the soft real-time tasks such as telecommunication and finance transactions. This research direction is interesting since it is reappearing in the contemporary cloud and virtualisation studies.

2.6.2 Multimedia Support on GPOS with Real-Time Features

There were attempts to implement RTOS scheduling algorithm such as RMS or its modifications statistical RMS [119] in GPOS to support multimedia applications. RMS is extended into SRMS (statistical RMS) to accommodate for various execution time with average QoS. SRMS has been implemented in KURT Linux [120]. The RT Linux work

[1] 'cli', 'sti' and 'iret' are the instruction opcode or interrupt related register in Intel x86 architecture

was initially driven by requirements from the live audio community. In 2005, the primary target of RT Linux was to support high-performance multimedia and telecommunication applications [51]. [51] also revealed an anecdote regarding an open letter from the audio community to the Linux kernel community that is called “a joint letter on low latency and Linux” in 2000. This open letter influenced the Linux RT pre-emptive patches developed since 2001.

Since 2008, Linux based Android OS has gained popularity not only in the smartphone area but also in the embedded world such as set set-top boxes and media players. There are development and commercial initiatives to make Android OS support RT applications including on-demand or interactive media applications [121].

2.6.2.1 Live Audio Processing using OS

Modern audio processing include some computational intensive tasks such as configurable feature extractions, machine learning and advance multichannel digital signal processing (DSP). GPOS based DAW can provide these functionalities on multiple channels with different source sampling rates and flexible routing as shown in Figure 1.3. This sometimes results in occasional bursting CPU utilisation that is close to full, which causes the unpredictable drop outs or delay of real-time task processing [107] and audio signals [12].

Using GPOS with software RT extensions reduces the audio processing latency. However, we suspect that using pre-emptive priority scheduling based OS to carry real-time tasks with optimisation can achieve the performance at most of the time, although rarely but it still can fail occasionally, see results from Table 2.4 to Table 2.6.

It shows for low buffer settings such as a few audio samples (under 1ms), the system might only successfully schedule the tasks within time under certain percentage statistically. In addition, the traditional heuristic and ad hoc design approaches have the same predictability problem as described in [112] in 2011. The trail and error approach needs to be adopted to find out the minimum latency. This poses the challenge of using traditional scheduling especially when the time and jitter constraints have to be met, whereas the use case requires mixed criticality with sporadic burstiness of tasks. For

audio processing, this often results in loss of audio samples occasionally. The human ear is especially sensitive to the time correctness, even one sample missing will cause audible effects.

2.6.3 Typical Real-time Scheduling Algorithms

Cyclic executive scheduling is one of the earliest schemes that is realised in the real-time operating system between 1970s and 1980s [122] especially for hard RT tasks. When the system is complex, it is regarded as inflexible and difficult to maintain. The rate monotonic scheduling (RMS) and the earliest deadline first (EDF) methods were proposed to overcome these difficulties to assign priority dynamically based on the characteristics of the tasks. In the case of RMS, the task with shortest period is assigned with highest priority [123, 124, 125, 126].

RMS enabled the significant engineering advancement in many areas such as in space exploration. On the other hand, the modern forms of cyclic executive based approaches can be found in safety-critical systems as temporal segmentation mechanism such as "ARINC Specification 653" that sets standards for avionics RTOS [127] and "WorldFIP" that defines the Factory Instrumentation Protocol later becomes part of IEC 61158 standard [128]. The cyclic executive approach is often part of hierarchical scheduling model to ensure the temporal segmentation [129, 109].

2.6.3.1 Characteristics of Modern Real-Time Audio Systems

The modern live and interactive audio system is an example of the cyber-physical system (CPS) that integrates different inputs and outputs functions with physical interactions. It has a variety of tasks that require different criticality. To find the best scheduling schemes for this situation, firstly we look into the characteristics of the live audio systems.

Traditionally, there are two categories of real-time systems: hard real-time system and soft real-time system. In hard real-time system, the missing deadline of tasks are regarded as failure, whereas the soft real-time system can have some level of tolerances of missing deadlines. Some textbook regards multimedia system as hard real-time system,

due to the strict jitter requirements or human perception of glitch caused by missing task deadline.

We think the live audio system is neither hard real-time nor soft real-time. It lays somewhat in between. In a professional audio environment, we would not want to miss any audio/video frames that may cause the negative perceptual effects. However, the jitter of processing single audio or video frame can be tolerated within acceptable user perceptions, using a de-jitter buffer to compensate it at the cost of delay. In summary, the characteristics of such system are below:

- It mainly deals with multiple periodic tasks at different update rate with pseudo-isochronous tasks pattern.
- It is acceptable to miss the deadline for some tasks, but will affect the quality of experiences. Ideally those tasks that missed the deadline shall not be discarded but still be scheduled at a later point.
- It can provide the trade-off between delay and jitter, using buffering to mitigate the jitter of the samples.

The traditional cyclic scheduling based approach is difficult to grow and maintain when the number of tasks increases and the periods of tasks are not harmonically related. However, the properties of the multimedia system indicate there are compromises that can be made that is to adjust the tasks deadlines within the perceptual tolerance to simplify the system realisation and increase the efficiency of the scheduler.

In this work, we propose a new OS scheduling framework that is called Time Deterministic Cyclic Scheduling (TDCS), which is specifically tailored for the real-time audio system with trading off mechanism of latency and predictable QoE requirements that aims to achieve the predictability of specific type of tasks using systematic design approach. The performance evaluation based on simulation between TDCS and RMS is also given to show the pros and cons.

2.7 Latency in Audio Networking

From the earliest "audio" (long haul telephone call) network in 19th century invented by Bell Telephone Company to the current Audio over IP network in the digital era, the delay of the digital network seems to be worse than the analogue era. There are concerns of losing responsiveness of audio transmission quality when EU decides to retire ISDN technology and move all the broadcasting to IP network [130, 131]. The convergence of network seem to be inevitable. The first VoIP started in 1995 and gained popularity from 2004. Similarly, the first Audio over Network was experimented in 1995. Later CobraNet was invented in 1996. In 2013, the new standard AES67 was published for low latency high-quality audio over IP networks [132, 133, 44] aiming for professional audio industry.

Still, professional audio with low latency requirements tends to use an isolated network to avoid the uncertainty and jitter that could happen in a statistically multiplexed packet switched based network such as IP and Ethernet based network. The major problem of using the current packet-switched network to support audio and video is the non-deterministic timing performance. The current network can deliver audio data with low latency in good condition, but it might suffer the loss of packets and jitter when the network is congested.

The International Organisation of Standardisation of Open System Interconnection (ISO-OSI) model has divided the network system into seven abstract layers, from lowest physical layer to highest application layer. At the moment, the latency of transmitting audio over a packet-switched network is caused by various factors such as coder delay, transmission delay, packetisation delay, queuing delay, and de-jitter buffering etc. Most of them are associated with the way how statistical multiplexing works at the different OSI layers.

2.7.1 Network Support Real-Time Applications

It is essential to work from the bottom layer up to ensure the time accuracy for the top application layer. The low latency and real-time features were not the modern networks

are designed for. There are various initiatives and modifications at different network layers that try to address the timing issues. In general, they can be categorised into three main approaches [134, 135, 136].

- (1) Direct modification of layer 2.
- (2) New or modification based on existing layer2.
- (3) Solutions that work on top of Layer 3 - IP layer.

These approaches are summarise in the below figure:

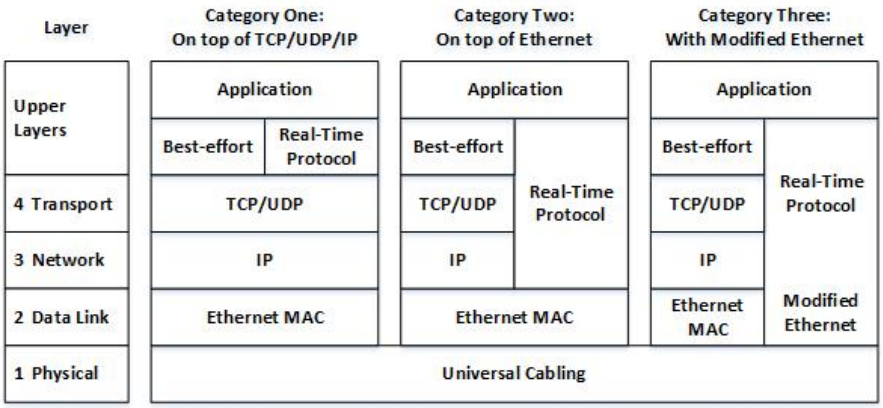


Figure 2.11: Different approaches of real-time network protocols

(1) Direct modification of layer 2

This process focuses on modifying the Ethernet MAC layer to realise the isochronous RT requirements. In theory, this approach could get a much better RT performance without the limitations of best effort based Ethernet IP, in spite that they may increase the complexity when customising the lower layers or even using dedicated firmware. For instance, the SERCOS, EtherCAT, TTEtherent, PROFINET-IRT and recent Time-Sensitive Networking (TSN) Protocol suite are based on this mechanism[134, 137].

The TSN protocol suite originates from the work of the Audio Video Bridging (AVB) task group. In 2011, several IEEE standard groups including AVB formed TSN task group in order to achieve the QoS requirements for low latency streaming in Ethernet

networks [138]. TSN Standards included the IEEE 802.1AS time synchronization protocol, the IEEE 802.1Qav forwarding and queuing protocol, and the IEEE 802.1 Qat signalling protocol.

These systems are capable of isochronous transmission which would satisfy most current time-critical applications. Nonetheless, the first two protocols both are based on the master-slave principle with its limitations [139, 136]. TTEtherent adopts a TDM based time cycle which does not utilise the full bandwidth, while TSN implements a priority based control scheme with multiple types of services, which does not prevent the queueing problem among different flows with the same priority.

(2) New or modification based on existing layer2

This approach is based on software configuration. Realisations like Time-Critical Control network (TCnet), and PROFINET CBA (Component-Based Automation) fall into this category [134, 140, 141]. These solutions can obtain hard RT requirements with a cycle time of 1-10 ms. However, some professional multimedia applications would require a low jitter within the range of nanoseconds [142].

(3) Solutions on top of TCP/UDP/IP

Examples of RT multimedia systems are Real-time Transport Protocol (RTP) [143], Real-Time Control Protocol (RTCP), and Integrated Services (IntServ) and Differentiated Services (DiffServ) based solutions. The RTP suite can achieve soft RT behaviour with a delay of millisecond level [144], which can be used in scenarios such as simple multicast audio conference, audio and video conference, mixers and translators, and layered encodings [143]. Many current network music types of research focus on improved latency management and prediction at millisecond scale, which may affect the music ensembles over the network [8, 9, 10].

Typically, IntServ and DiffServ are deployed as fine-grained and coarse-grained systems, respectively. The former uses the per-flow reservation with the Resource Reservation Protocol (RSVP), while the latter is based on traffic classification and marking. Per-flow

reservation requires procedures like call set up, maintain and termination, and thus many status information need to be stored in the routers through a path, which makes pure IntServ complex and not scalable [145]. In contrast, DiffServ needs no setup time and offers scalability. However, no end-to-end (E2E) guarantees are provided in DiffServ due to the lack of bandwidth reservation.

In 2016, the IETF working group Deterministic Networking (DetNet) was formed [146]. The main focus of the group is to introduce a model for forwarding real-time traffic (traffic with timing constraints in general also referred to as deterministic traffic) beyond LAN boundaries. The proposed model will enable fully scheduled operation controlled by a central controller to guarantee the timing constraints for deterministic traffic without compromising the ability of the network to carry traditional best-effort traffic. However, the high level network deterministic relies on the timing support from lower layers.

2.7.2 Audio Networking

The networking of live audio for professional applications typically uses layer 2 based solutions such as AES50 [41] and MADI that utilise fixed time slots similar to Time Division Multiplexing (TDM). However, these solutions are not effective for best effort traffic where data traffic utilises available bandwidth and is consequently subject to variations in QoS. There are audio networking methods such as AES47 which is based on asynchronous transfer mode (ATM), but ATM equipment is rarely available. Audio can also be sent over Internet Protocol (IP) in wide area network (WAN), but the size of the packet headers and the difficulty of keeping latency within acceptable limits make it unsuitable for some low latency applications.

For low latency live audio, TDM based protocols such as AES50 can provide excellent performance. The proprietary AES50 router can achieve the latency as low as a few samples with a fixed number of channels reserved for other traffic. Converged networks based on IP, scalable from LAN to WAN are required to support the vast (and growing) interactive audio/video media traffic on the Internet. In high-resolution and low latency audio applications, many audio specific networking technologies modify the existing layer-2 or layer-3 protocols to utilise the current network infrastructure such as Ethernet

and IP. Since 2003, there are a number various network protocol suites and techniques proposed from different organisations such as Livewire, Wheatnet-IP, Dante, N/ACIP, Q-LAN, RAVENNA and AVB. Some of them are proprietary solutions. Most of them have two important elements: transport protocol and clock synchronisation mechanism. The most recent effort is to standardise these similar technologies under the umbrella of AES67 to make them talk to each other [44].

However connectionless packet architectures are inevitably problematic for deterministic data, especially when low latency is required. Furthermore, it is difficult to find a networking architecture, which supports both time-critical audio data and best effort data (such as file transfer and emails). Current QoS, traffic engineering and over-provisioning solutions cannot solve the entire problem: they complicate the system and increase power requirements and cost. To effectively use AES67 based audio over IP for professional settings, there are a number of caveats. Both UDP and RTP are supported as transport layer protocols in AES67, but for sharing the platform with other types of traffic, some QoS mechanisms are needed such as using DiffServ. The AES67 synchronisation is based on IEEE1588/PTP, but the accuracy of the clock depends on the underlying network timing quality [147, 148]. To have a TSN based timing management scheme to support AES67 is also proposed. In practice, though using existing converged IP over Ethernet technology, often the audio network is still configured in an isolated physical domain.

The main question is how to have a network technology that supports both time deterministic data and best effort data efficiently form bottom up. In this work, we propose a new low latency network architecture that supports both time-deterministic and best effort traffic towards full bandwidth utilisation with high-performance routing/switching. For live audio, this network architecture allows low latency as well as the flexibility to support multiplexing multiple channels with different sampling rates and word lengths.

2.8 Summary

In this chapter, we reviewed the previous work in relation to system latency in real-time audio processing, especially focusing on the three aspects in the audio processing chain

that cause significant problems. Each of these aspects have some unanswered questions that need to be addressed.

For **delay in $\Delta\Sigma$ ADC/DAC delay**, the significant delay is caused by the group delay of multistage multirate High-resolution Anti-aliasing Anti-image Filter (HAAF) implemented in most $\Delta\Sigma$ ADC/DAC devices. There is a lack of accurate delay reporting mechanisms and standards. There were no evaluations of how different filter architectures in $\Delta\Sigma$ ADC/DAC perform in the time domain. For the most commonly used cascaded Linear Phase FIR multistage filters, we suspect that the overall delay can be expressed as a function of design parameters and can be optimised along with computational cost. Finally, for minimum phase filter, there is a lack of proper objective measurements.

For **delay caused by using the operating system** to process audio, the delay is caused by using buffering to accommodate unpredictable jitter and to reduce the CPU load. High CPU utilisation often worsens the de-jitter buffer underrun situation. There is a lack of updated latency test using desktop operating systems. We reviewed how RTOS deal with the time-critical tasks and the solutions of extending GPOS with RTOS features. We suspect the existing scheduling algorithms may not be able to produce deterministic tasks outputs under high CPU utilisation for low latency audio processing.

For **delay caused by using the data network to transmit audio signal**, apart from the propagation delay in network, to ensure the low latency and stable timing performance of the system, the current approach is to configure network in a physical isolation manner. Although, the audio network is moving towards using existing data network architectures and protocols. The key problem seems to avoid unexpected queuing and jitter caused by other competing traffic on the same network. There is a lack of an elegant way to support both multiple channels of time deterministic low latency audio signal as well as other types of traffic.

In the next few chapters, we will show how to address these identified gaps, verify the hypothesis and provide possible solutions.

Chapter 3

Delay of Multistage Filter System in Audio Conversion

In this chapter, we present the work for understanding and reducing the delay in $\Delta\Sigma$ ADC/DAC. It is organised as the following.

- Section 3.1, we present the latency measurements of various typical audio ADC/DAC that can be found in the digital audio system [26]. This work was conducted and published in 2011.
- Section 3.2, we evaluated different filter architectures that can be used in $\Delta\Sigma$ ADC/DAC on their time domain performance with reference to other important measures such as computational cost and Signal to Noise Ratio (SNR) [27]. This work was conducted and published in 2012.
- Section 3.3, we focus on the delay and cost of multistage linear phase FIR filter that is the most popular filter architecture used in $\Delta\Sigma$ ADC/DAC. A new balanced design is proposed to yield the final design that takes overall delay into account. Part of this section was based on the research work conducted and published in a 2016 paper [28].
- Section 3.4, we develop a set of objective time domain measurements for minimum phase filters that have non-linear delay behaviour.

Though the latency measurements of ΔΣ ADC/DAC work was conducted in 2011, the updated latency survey based on 15 popular ΔΣ ADC/DAC chips datasheets was also presented in the more recent publication of [28] in 2016 that showed similar latency scale. The time domain performance of multistage filter design and evaluation for ΔΣ ADC/DAC work in section 3.2 was conducted in a 2012 paper [27], based on this work, the new optimal design research was published in a 2018 paper [149], that showed the earlier work in 2012 is still relevant today and open the new research territory.

3.1 Delay of $\Delta\Sigma$ Audio ADC/DAC Converters

3.1.1 Motivations of the Work

Previous research has reported measurements of the latency of the whole audio processing chain based on a "blackbox" approach as reviewed in Chapter 2, Section 2.4, especially on computer based DAW systems where the latency is more severe. However, this approach does not provide great help for latency management and compensation. It would be better to have accurate latency measurement of each stage of the whole audio processing chain.

In computer based Digital Audio Workstation (DAW) systems the major latency (typically around 5 to 10ms) is caused by the buffer size [12], so the ADC/DAC converter latency is sometimes neglected or a rule of thumb of 1ms to 2ms is assumed. Latency of professional digital consoles is normally under 2ms. Recording engineers are still concerned with latency caused by routing audio channel from digital console through various external analogue devices. In this case, the latency is mainly caused by additional ADC/DACs.

To the best knowledge of authors, no research has shown accurate latency measurement of hardware $\Delta\Sigma$ ADC/DAC. This section presents the results of latency measurement of typical compact analogue to digital and digital to analogue converters (ADC/DACs) in isolation from computer system processing overheads by using a high-speed data acquisition device. The report discusses the testing methods and pitfalls. It confirms that the latency is almost exclusively accounted for by the expected group delay of the digital decimation filters and interpolation filters used in the Sigma-Delta converter.

3.1.2 Latency Test for Hardware Audio Codecs

Modern audio converters are commonly available in compact form by using one chipset to integrate both multichannel ADC and DAC, which is known as a hardware codec. That can be found in computer motherboards and USB soundcards. They normally operate at relatively high frequency in order to multiplex multiple audio channels.

The most commonly used compact architecture for audio range converter is based on multi-bit sigma-delta modulators (SDMs). Accurate latency values for these converters are not normally available and cannot be easily measured, although it is well known that this latency is mainly due to the group delay of the SDM's internal digital decimation and interpolation filters. The group delay of the digital filter is expressed in a variety of formats in manufacturer's datasheets, quite often as a function of the number of samples divided by the sampling frequency.

3.1.2.1 Testing Platforms and Devices

The tested audio converters are listed in Table 3.1. They are all single chipset that integrate multiple ADC and DAC channels with available group delay data from datasheet. The protocols supported by these devices are typically used in most embedded audio systems and computer based system.

Table 3.1: List of Testing Systems

Code	**Model**	**System**	**Protocol**
(a)	TLV320-AIC23B	TMS320VC5510 DSK	McBSP
(b)	AD1836A	ADSP-21161N EZ-KIT Lite	TDM/I²S
(c)	AD1981B	PC	AC'97
(d)	AD1882	PC	HD Audio

The Multichannel Buffered Serial Ports (McBSP) supported by codec (a) is software configurable protocol that supports various I²S frame formats. Overall, all tested devices use standard serial transfer digital audio interfaces with multichannel support by Time Division Multiplexing (TDM) technique.

The purpose of the test is to evaluate the latency caused by the ADC and DAC of computer based DAWs. The most popular on-board audio subsystem are based on Audio Codec'97 (AC'97) standard or the Intel HD Audio standard, which are evaluated by devices (c) and (d). Devices (a) and (b) have similar architectures and supporting protocols that can be found in external audio interface cards or other compact professional audio systems.

3.1.2.2 The Testing Method

The previous latency tests rely on comparing the offset between input and output signals of the system. However this method cannot be used for codec latency testing. Normally, codecs provide an internal analogue loopback circuit to support a "listen to input" feature for low latency applications such as audio monitoring. To the best knowledge of the authors, codecs seldom implement digital loopback. Therefore, we cannot treat the codec as blackbox and obtain the latency value of ADC and DAC by directly measuring the time domain signal offset from analogue inputs and outputs. The measurement has to be carried out by comparing analogue stimuli and their digital binary patterns.

The test is performed by directly probing the two test points of the analogue input pin and digital data output pin for the ADC module, or the analogue output pin and digital data input pin for the DAC module, using a high speed data acquisition tool. The testing method is depicted in Figure 3.1.

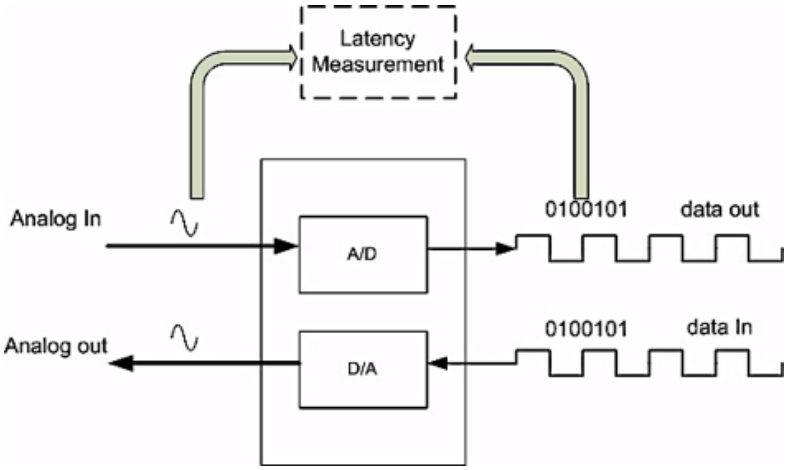


Figure 3.1: Codec latency measurement method

The multichannel features are supported by multiplexing samples in a digital serial data link. So the clock rate of the bit stream is normally much higher than the sampling frequency in order to support this feature. The range of codecs we tested have bit stream clock rate varying from 12MHz to 48MHz. Because we need to observe the change of binary pattern, a high time resolution is needed to interpret the binary data. Therefore, the

National Instrument high speed data acquisition module NI PXI-5114 is used to support up to 250M samples pre second.

The digital data is represented internally with two's complement. It is relatively easy to test DAC latency, because the digital data can be clearly defined by software such as discrete impulse signal (a series "0"s which has only one "1" inserted). Hence the clear binary pattern change can be captured, with observable "sinc function" like impulse output in the corresponding analogue pin.

It is more of a problem to measure ADC latency, the test signal needs to be considered so that the binary pattern change can be clearly identified. The noise interferences around the reference point zero can easily make binary pattern flips between all zeros and all ones. Therefore, we select the step input, which steps from the most negative to most positive voltage according to the codecs' electrical characteristics.

3.1.2.3 The Limitation of the Test

The selection of the measuring point affects the accuracy of the measured values. The analogue measure point is selected as 90% of the final step value to indicate the "change" of the signal.

Most codecs input and output the digital stream in frame mode, and multiplex the multichannel samples inside a single frame. Therefore the selection of a binary pattern to reflect the analogue signal change might affect the accuracy around ± the frame time. For example if the frame rate is 48k Hz, then reading error might be $\pm 20.8\mu$s.

Some protocols such as Intel HD audio uses fixed frame speed at 48K data bus to delivery different sampling frequencies. It will cause the uneven timing point of samples when measuring the digital end [150].

3.1.3 Testing Results and Discussions

The latency of a codec is mainly caused by the group delay of internal digital decimation and interpolation filters. However, sometimes manufactures do not provide details or

completed information. Table 3.2 shows the available group delay information of tested codecs. They do not have the same format and sometimes they are hard to follow due to lack of details.

Table 3.2: Group delay of codec obtained from datasheet

Codec	**Group Delay**		**Notes**
	ADC	**DAC**	
(a)	(12, 20, 3, 6) / Fs	(11, 18, 5, 5) / Fs	Filter type (0, 1, 2, 3)
(b)	990.20μs	446.35μs	At sampling frequency 44.1kHz
(c)	16 / Fs	16 / Fs	
(d)	20 / Fs	20 / Fs	Typical value

The codec (a) provides the group delay information in number of samples according to different internal filter types. The filter type can be determined by codec register setting for different operation modes and sampling frequencies. In this case, codec (a) provides the most completed information among the codecs we tested.

The codec (b) provides the typical group delay in microseconds at sampling frequency of 44.1k Hz for both digital decimation filter in ADC and interpolation filter in DAC. However we don't know whether the filter varies when different sampling frequencies is used.

The codec (c) provides overall delay as a function of sampling frequency with fixed 16 samples. From the measured results shown later, we assume that the 16 samples is for both decimation filter and interpolation filter. The codec (d) provides similar information as codec (c), with the typical delay of 20 samples. Apart from the same problem as codec (c), it also shows that the maximum delay can be -100 samples, which is hard to interpret.

Therefore the theoretical latency caused by group delay at typical sampling frequency 48k Hz, which can be summarised in Table 3.3.

Table 3.4 shows the measured latency values in microseconds. The measured latency values are correlated with reported group delay values. However the measured latency in some cases is longer than the group delay. The measured latency can be around 7 samples more than datasheet group delay.

Table 3.3: Manufacture group delay data and equivalent latency at 48k Hz sampling frequency

Codec	**Group Delay (samples)**		**Equivalent Delay (μs)**	
	ADC	**DAC**	**ADC**	**DAC**
(a)	12	11	250	229
(b)	44	20	917	417
(c)	16	16	333	333
(d)	20	20	417	417

We keep the decimal point of estimated samples due to the group delay caused by multi-stage decimation or interpolation filter. It can be normalised at the final sampling frequency with a fractional number of samples.

Table 3.4: Measured latency in microseconds at 48k Hz

Codec	**Latency (μs)**		**Equivalent Delay (samples)**	
	ADC	ADC	DAC	DAC
(a)	308	275	14.8	13.2
(b)	1001	550	48	26.4
(c)	335	364	16.1	17.5
(d)	516	553	24.8	26.5

Table 3.5 shows the measured latency at different sampling frequency for the Intel HD codec (d).

It is worth to note that the 44.1kHz data stream of the Intel HD audio protocol is delivered at 48kHz frame rate by only delivering 147 samples per 160 sample slots. Therefore the latency measurement of the digital data end has inherent deviations.

The test results show that the current hardware audio codec in both analogue to digital and digital to analogue direction will cause the latency range from around half millisecond to 1.5 milliseconds. The current datasheet information of group delay is either lacking of details or hard to use for estimation of sample accurate latency values. It is suggested that the latency information of an ADC/DAC needs to be available in standard

Table 3.5: Latency in microseconds at different sampling frequency and equivalent samples for HD Audio Codec

Fs (kHz)	**Measured Latency (ms)**		**Equivalent Delay (samples)**	
	ADC	DAC	ADC	DAC
32	748	853	23.9	27.3
44.1	562	620	24.8	27.3
48	516	553	24.8	26.5
96	283	288	27.2	27.7

and accurate format.

Some protocols designed for multiplexing channels make data bus delivery timing different from sampling frequency. This architecture might cause the problem for sample accurate live audio systems with minimum sample buffers.

Most decimation interpolation filters are implemented using multistage linear FIR structures. This delay could be reduced by adopting a different digital filter structure, such as a minimum phase filter with trade off of non-linearity [151].

It would be beneficial to have the facility to report the latency of a hardware audio codec to the up-layer software stack. For example, the simplest facility can be the digital loopback between ADC and DAC, which bypasses the up-layer audio driver and operating system. It would be interesting to see whether the advanced Built-In Self Test (BIST) architecture can be designed to automatically report latency of converters [152].

However, the hardware testing and manufacture datasheets cannot reveal the internal structure of the decimation and interpolation filters of $\Delta\Sigma$ ADC/DAC. These high-resolution anti-aliasing and anti-image filters (HAAF) can be realised in different forms and that have different latency effects. Next section we will look into the time domain performance of these filter types from the theoretical and white box approach.

3.2 Time Domain Performance of Decimation Filter Architectures for High Resolution $\triangle\Sigma$ ADC/DAC

In the last section, we presented the accurate latency measurements of typical compact analogue to digital and digital to analogue converters with $\Delta\Sigma$ architecture. It showed that the latency is almost exclusively accounted for by the expected group delay of the digital decimation filters and interpolation filters used in the Sigma-Delta converter. Various filter design methods can be used to realise these filters. However, to the best knowledge of the authors, most research in this area focuses on computational cost and energy efficiency. No research has shown the time domain performance of these filters in a comprehensive way.

In this section, we present the results of a comparison of different decimation architectures for high-resolution sigma delta analogue to digital conversion in terms of pass-band, transition band performance, simulated signal to noise ratio, and computational cost. In particular, we focus on the comparison of time domain group delay response of different filter architectures including classic multistage FIR, cascaded integrator-comb (CIC) with FIR compensation filters, particularly multistage polyphase IIR filter, cascaded halfband minimum phase FIR filter, and multistage minimum phase FIR filter designs.

3.2.1 Introduction of the Work

In a sigma delta analogue to digital conversion ($\Delta\Sigma$ ADC) based high-resolution audio system, decimation filters are used for obtaining PCM data from density modulated 1-bit or multi-bit signals [153]. Most modern digital audio systems include some sort of oversampling and downsampling processes in either software format or integrated circuits.

Common practice in the audio industry is to use cascaded half-band linear phase FIR filters for interpolation or decimation processes. Recently, there has been increasing interest in adopting different filter architectures [74] to eliminate pre-ringing (mainly

for DAC) and high group delay (for both ADC/DAC) caused by the linear phase design.

The advances in digital hardware fabrication now allow fairly sophisticated structures to be used. The 64 times oversampling ratio audio band decimator can be easily implemented in silicon with a three stage FIR filter [87]. There are also commercial audio codec products that enable users to directly process modulator outputs or to customise the internal digital filters.

In ΔΣ ADC/DAC for high-resolution audio systems, the significantly large number of taps and the multistage architecture introduce high group delay that may not be desirable for some live or low latency applications.

In Section3.1, we showed the latencies of ΔΣ ADC/DAC, which are mainly contributed by the group delay of internal digital filters and can be as high as 1.5 milliseconds. Many live audio applications or electronic musical instruments and software synthesizers require overall latency less than a few milliseconds. In these situations, the phase response can be diffused by the live environments, and hence becomes less important. And the high group delay caused by linear FIR filters can be undesirable. Therefore, it would be interesting to see how different types of filters perform in comparison with classic linear FIR filter within multiple constraints such as cost, signal to noise ratio (SNR), and filter characteristics.

Next, we evaluate time domain performances of different decimation filter architectures with typical HAAF design specifications for the ΔΣ ADC as in [87]. The tradeoffs of filter characteristics are discussed as well.

3.2.2 Basic Concept of Decimation Filter

The principle of the decimation process is similar to sample rate conversion, for which one must comply with the Nyquist theorem in order to avoid aliasing. Decimation can be treated as two cascaded function blocks: the downsampling process and the anti-aliasing filtering process. To downsample an input signal $x(n)$ with positive integer factor M, the output signal can be represented as $y(m) = x(Mn)$. If there is any frequency component greater than $f_s/(2M)$ in the original signal, where the original sampling frequency is fs,

the downsampling process will result in aliasing. In order to avoid the aliasing problem, a low-pass filtering process $H(z)$ is needed in decimation [75].

The purpose of the decimation filter in ΔΣ ADC is threefold:

1. To avoid the aliasing in the decimation process.
2. Help relax the analogue anti-aliasing filter design requirements.
3. To remove quantization noise caused by the ΔΣ modulator and to obtain the effective number of bits (ENOB) in PCM format.

Almost any type of lowpass filter design techniques can be used for to decimation filter design [99] [84] [88]. However because the ΔΣ ADC has its own characteristics and specific application requirements, the filter design work has always been the tradeoff of various design and implementation constraints.

The straightforward design can be linear phase single stage FIR lowpass filter. The order of FIR filter N can be estimated by the Equation (3.1) as summarised in [153] [84]:

$$\begin{aligned} N \approx ((\log_{10}\delta_s)[a_1(\log_{10}\delta_p)^2 + a_2(\log_{10}\delta_p) + a_3] \\ +a_4(\log_{10}\delta_p)^2 + a_5(\log_{10}\delta_p) + a_6) f_s/\Delta f \end{aligned} \tag{3.1}$$

Where δ_p is passband ripple, δ_s is stopband ripple in linear, $a_1 = 0.005309$, $a_2 = 0.07114$, $a_3 = -0.4761$, $a_4 = -0.00266$, $a_5 = -0.5941$, and $a_6 = -0.4278$. The Δf is the transition bandwidth and f_s is the sampling frequency at oversampled rate.

When the oversampling ratio is large and the desired transition bandwidth of decimation filter is narrow, the order N can be very large, i.e., up to several thousand [153] [88]. So although a single stage FIR filter can be realised, it is sometimes impractical due to this extremely high order. A more effective approach is to use cascaded multistage design [84], which provides an efficient general solution for decimation, interpolation and narrow band filter design. [84] also indicates the duality of the decimation and interpolation processes, so the same filter structure can apply to both.

For decimation filters in ΔΣ ADC, significant effort has been made to use simplified filter structures and implementation methods [92, 154, 73, 155] of multistage de-

sign. Among these methods, two important approaches are the cascaded integrator-comb (CIC) filter structure [92], and using halfband or N-band filters [93, 95, 96]. The polyphase filter structures [93] have also been widely adopted as effective implementation in multirate signal processing, including decimation and interpolation.

3.2.3 Group Delay of the Decimation Filter in $\Delta\Sigma$ ADC/DAC

Recall the literature review section, the group delay of a digital filter is defined as the first derivative of phase response as in Equation (3.2),

$$D_m(\omega) = -d\phi(\omega)/d\omega \tag{3.2}$$

where ϕ is the total phase shift in radians, and ω is the angular frequency in radians per unit time. When the phase is linear then the group delay is constant. For non-linear filters, the group delay is a function of frequency. The decimation filter is essentially a digital anti-aliasing filter. Therefore the filter is typically designed and normalized at input sampling rate. The group delay at the output sampling rate can be calculated as in Equation (3.3), where M is the decimation factor.

$$D(\omega) = \frac{D_M(\omega)}{M} \tag{3.3}$$

For linear phase FIR filters, the group delay is around half the filter order *N*. Hence higher order results in higher group delay. The multistage design significantly reduces the filter order in total. However due to the fact that the stages operate at decreasing sampling frequency, the overall group delay normally is worse than the single stage filter by the same filter design method.

$\Delta\Sigma$ ADC is commonly used in high resolution audio because it can achieve more than 20bit ENOB (Effective Number of Bits) [70]. The higher oversampling ratio of the $\Delta\Sigma$ modulator also helps improve the SNR as well as signal-to-noise-and-distortion ratio (SINAD). Therefore, a more restricted decimation filter specification is needed in this case in terms of good stopband attenuation, small passband ripples and narrow transition

band in order to obtain the PCM signal with equivalent ENOB. A linear phase digital filter to meet such requirements normally has high order and a multistage design. But in both cases, it worsens the group delay response.

The group delay of the digital decimation filter is the largest contributor to the latency of $\Delta\Sigma$ ADC [26] [14]. Minimum phase FIR and IIR filters can be used in delay critical applications when phase linearity is not required. To the best knowledge of the authors, there is little literature available to provide detailed qualitative or quantitative reviews of how different decimation filter architectures impact time domain performance in high resolution audio $\Delta\Sigma$ ADC.

3.2.4 Evaluation Methodology

When the linear phase in the passband is not restricted, the decimation filter can be any type of lowpass filter. The design space is so wide that there is no systematic approach to optimal design choice [153]. Therefore we have to properly consider the different filter architectures for evaluation with justified rationale.

3.2.4.1 Selection of Testing Filters

We evaluated the time domain performance of the following main filter architectures with the typical filter design specifications in Table 3.6, based on a commercial ADC product, as specified in [87]. The traditional and modern linear phase filter as well as the nonlinear phase, low group delay filters were evaluated. All the filters evaluated should satisfy the 90 dB stopband attenuation specification. The group delays of filters are calculated and compared. The group delay response figures are also provided to assess the effects of group delay distortion of nonlinear filters.

3.2.4.2 Linear Phase Single Stage FIR and Multistage FIR Filters

The linear phase single stage FIR filter and the multistage FIR filter are well-understood decimation approaches [88, 84]. They can be designed by Windowed-Sinc or optimal

Table 3.6: Filter Design Specifications

Parameters	**Desired values**
Decimation Factor	64x
Output Sampling Rate	48 kHz
Input Sampling Rate	3.072 MHz
Stopband Attenuation	$>$ 90 dB
Passband Ripple	$<$ 0.006 dB
Passband edge	21.6 kHz
Stopband edge	26.4 kHz

design methods, and they provide a good reference design in comparison with other architectures. The optimal design should give the minimum order of the filter. Hence it could help reduce group delay. In multistage filter design, the number of stages can be optimized as well.

In this case, the following filters are investigated:

1. A single stage FIR filter designed by windowed-sinc method with Kaiser Window (Kaiserwin).This design normally provides very good performance among different window functions.
2. A popular optimal equiripple FIR filter in single stage for decimation.
3. A 3-stage FIR filter designed by the optimisation method.

3.2.4.3 Cascaded CIC Filter With Linear Phase FIR Compensation

The CIC filter [92] is a very cost effective filter structure without multipliers. It is widely used in decimation. CIC filters are inherently linear phase, hence with constant group delay. Due to its simple and regular representation, the design of the CIC filter has less control of fine tuned parameters such as passband ripple and transition bandwidth. Therefore compensation filters are always adopted to improve the passband and other performances.

3.2.4.4 Six-Stage Half-Band FIR Filters With Linear Phase

The halfband filter is another effective architecture [93, 95] used in decimation. 64 times decimation can be realized by 6 cascaded halfband filters with each performing decimation by a factor of 2. This design [73, 93, 94] should have the theoretical minimum taps within the FIR decimators catalogue. However the additional stages may complicate the control structure and have negative impact on group delay. Therefore, this filter is designed for evaluating the group delay.

3.2.4.5 Multi-Stage Polyphase IIR Filters

Compared with FIR filters, the same magnitude response can generally be achieved by IIR filters with less coefficients. The IIR filter also typically has less group delay but with phase distortion.

The FIR filter is commonly used in multirate signal processing due to the effective filter structure realizations, such as the polyphase network [96], and the linearity requirements in most applications. But when nonlinear phase is allowed, the research [156, 157, 158] shows that recursive filters can also be designed and realised in a very cost effective way, especially halfband design with allpass polyphase decomposition.

In addition, the phase of a recursive filter can be equalized to approximate a linear phase filter. Thus, it would be interesting to find out how linear phase IIR filter performance in the time domain compares with linear phase FIR filters as well.

In this case we designed two types of IIR filters:

- the 6 cascaded halfband IIR filter with elliptic response.
- the 6 cascaded halfband IIR filter with quasi-linear phase response.

3.2.4.6 Multistage Minimum Phase FIR Filters

Minimum phase FIR filters with all zeros within the unit circle should have theoretical minimum group delay, and hence the fastest signal response. In this case, we designed

two minimum phase multistage FIR filters based on two typical effective linear phase designs:

- 3 stage minimum phase FIR filter.
- 6 stage minimum phase halfband filter.

A summary of evaluated filters is given in Table 3.7 [1].

Table 3.7: List of evaluated filters

Filter Type	**Filter code**
Kaiserwin FIR	Kaiser
Equiripple FIR	Eqrip
3-Stage Equiripple FIR	3-stage
3-Stage FIR minimum Phase	3-min
CIC without compensator	CIC
CIC with compensator	CICom
Six-stage halfband FIR	6hb
Six-stage halfband minimum phase FIR	6hbmin
Six stage elliptic IIR filter	6IIR
Six stage Quasi linear IIR filter	6IIRlin

3.2.5 The Filter Performances Matrix

Although the filters are designed to meet the specifications in Table 3.6, the actual designed filters may result in slightly different performances in terms of magnitude responses. Therefore some comparisons of magnitude response are also presented to see the correlation between the frequency domain and time domain performances.

The theoretical implementation cost is given in terms of the number of multipliers, the number of adders. The number of multiplications and additions per input sample for

[1] The source code of filter design listed in the Table 3.7 can be downloaded from `https://github.com/wyonghao/MultiStageDesign/tree/master/FilterDesign`

these filters will also be compared. The theoretical SNRs will be evaluated by using a Matlab Simulink model of the $\Delta\Sigma$ ADC with full amplitude sinusoid signals as inputs.

3.2.6 Filter Design and Group Delay Impact Results and Discussions

In this section the evaluation results of different types of filters are presented. Firstly, the designs of different types of filters are discussed. The correlation between various aspects filter design and its group delay properties are explored. Then the summary of group delay of all evaluated filters is presented and discussed.

3.2.6.1 Linear Phase Single Stage and Multistage FIR Filters

Design Considerations

Two FIR filter design methods are used for design of single stage FIR filters. According to Equation (3.1) and the specification (Table 3.6), the filter order is estimated up to 2314. The single stage filter can be designed by the Kaiser Window (Kaiserwin) method with very good passband and stopband performance. The Kaiser Window design meets the design specifications but with overestimated filter order N. However, the optimal equiripple algorithms sometimes underestimates the order, which is close to but does not meet the specifications.

The multistage linear phase FIR filter design uses three stages by the equiripple method. The decimation factors are /8, /2, and /4 respectively. The three stage design correlates with the decimation architecture in AD1877, as described in [87]. The second stage has decimation factor of 2, which is also a halfband filter. The stage 2 filter coefficients have zeros in every second order except that of the central point. The number of stages is also regarded as optimum by the automatic design algorithm from Matlab.

Results and Discussions

Table 3.8 shows a comparison of the single stage Kaiserwin design, the equiripple design, and the three stage equiripple design in terms of filter order and magnitude responses. Figure 3.2 shows the passband ripple at -0.01 dB to 0.01 dB of three different designs. All of these three filter designs have constant and relatively high group delays, which are in the range 500μs to 600μs delay at 48kHz sampling frequency (for detailed group delay values see Table 3.9). The Kaiserwin filter has better passband performance than the other two but with highest filter order *N*. Hence it also has highest group delay. The 3 stage equiripple filter has fewer orders in total but it has higher group delay than the same equiripple filter with single stage. This shows that the multistage structure normally worsens the group delay response.

Table 3.8: Compare single stage FIR filters with multi stage FIR filters

Filter	**Order**	**Passband Ripple**	**Stopband attenuation**
Kaiser	3658	0.0005 dB	91.34 dB
Eqrip	3023	0.0045 dB	89.95 dB
3-stage	39-14-193	0.0043 dB	90.34 dB

The 3-stage linear phase FIR is a typical implementation with the specified design criteria. Hence it is used as reference design to be compared with other filter architectures.

3.2.6.2 Cascaded CIC Filter With Linear Phase FIR Compensation

Design Considerations

There are various compensation methods to improve CIC frequency responses since the initial CIC concept from Hogenauer in 1981. We are interested in time domain performance on a typical CIC filter with an FIR compensation. Therefore a CIC filter and a linear phase FIR compensator are designed.

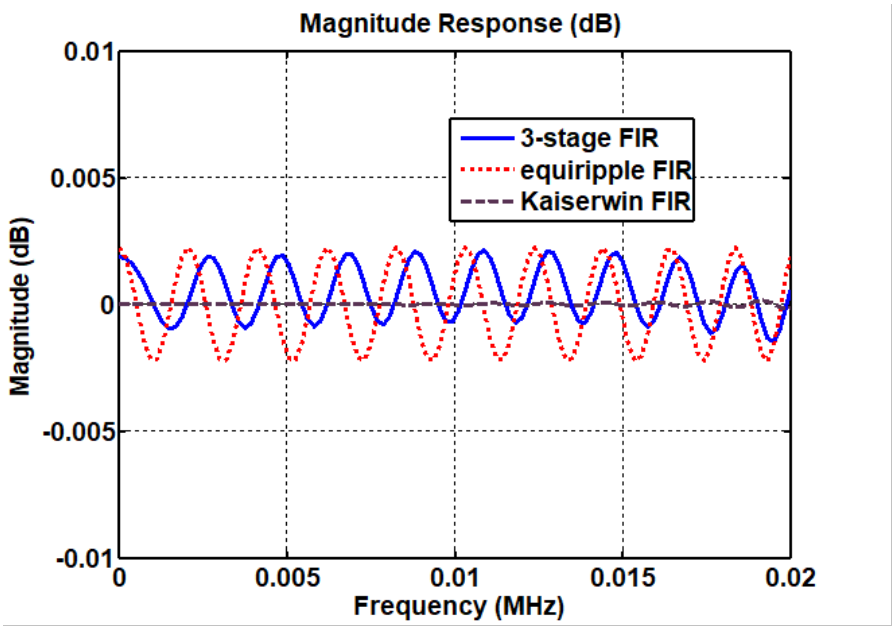


Figure 3.2: Magnitude passband of 3-stage FIR, equiripple FIR and Kaiserwin FIR filters

Figure 3.3 shows the magnitude response of the CIC without compensator. It clearly shows the passband performance does not meet the specification in comparison with reference design.

We designed the FIR compensator to flatten the passband ripple within the design specification, as shown in Figure 3.4. But in this case the transition band is still not compensated well in this case.

Results and Discussions

CIC filters are inherently linear phase, hence with constant group delay. This CIC filter without compensation has 19 sections with constant group delay of 598.5 samples (Table 3.9) but with unsatisfactory passband performance. To flatten the passband within specification, a fairly expensive linear phase FIR filter design method is required. Therefore overall it illustrates high group delay. Our compensator design results in group delay of 4022.5 samples (Table 3.9). Reducing the sections will decrease group delay but with

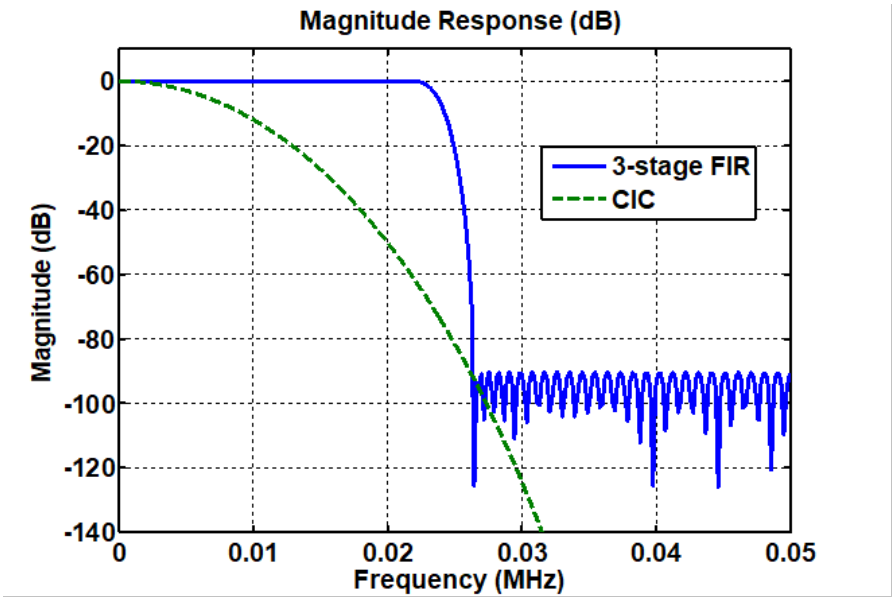


Figure 3.3: CIC without compensator in comparison with reference filter

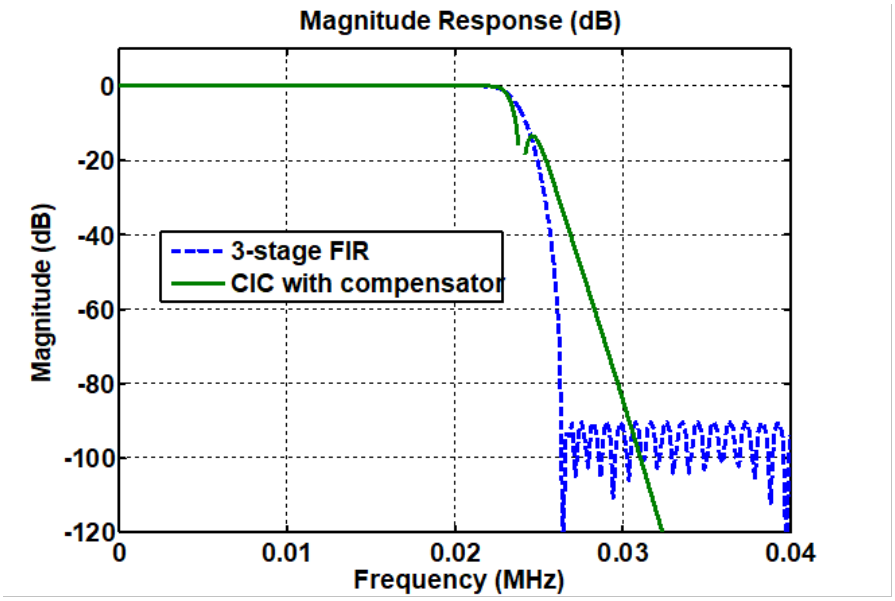


Figure 3.4: CIC Filter with compensator in comparison with reference design

worse passband and transition band performance.

There are advanced CIC filter design and compensation methods. In general they are low pass filter design techniques and thus they may not be very helpful in terms of reduction

of group delays.

3.2.6.3 Six Stage Linear Phase Halfband FIR Filter

Design Considerations

Halfband is a class of N-band filters where the number of bands N is 2. It is an effective filter structure and design method for decimation. Halfband is most effective in N-band filter class in terms of filter coefficients. So the 64 times decimation filter can be designed by cascading six halfband filters.

In [73], the author designed a 64 times decimation filter with both cascaded CIC and FIR filters and 6 stage halfband FIR filters. We designed a similar 6 stage halfband filters to meet the specification defined in Table 3.6 to compare its time domain performance.

Results and Discussions

The 6 stage halfband FIR filter performs well in terms of magnitude response (see Figure 3.5). It has lower implementation cost (Table 3.10) than other FIR filter design methods, but it worsens the group delay as compared with the reference design and other linear phase FIR filter design methods (Table 3.9).

3.2.6.4 Multi-stage Polyphase IIR Filters

Design Considerations

It is commonly believed that the IIR has less group delay but with non-linear phase response. The IIR filter can have an effective realisation structure, which is suitable in decimation and multirate signal processing as well [159], especially by halfband design with allpass polyphase decomposition.

Since the technique is available to design linear phase (quasi-linear) IIR to take advantage of the efficiency of IIR filter while maintaining the linear phase. Therefore it would

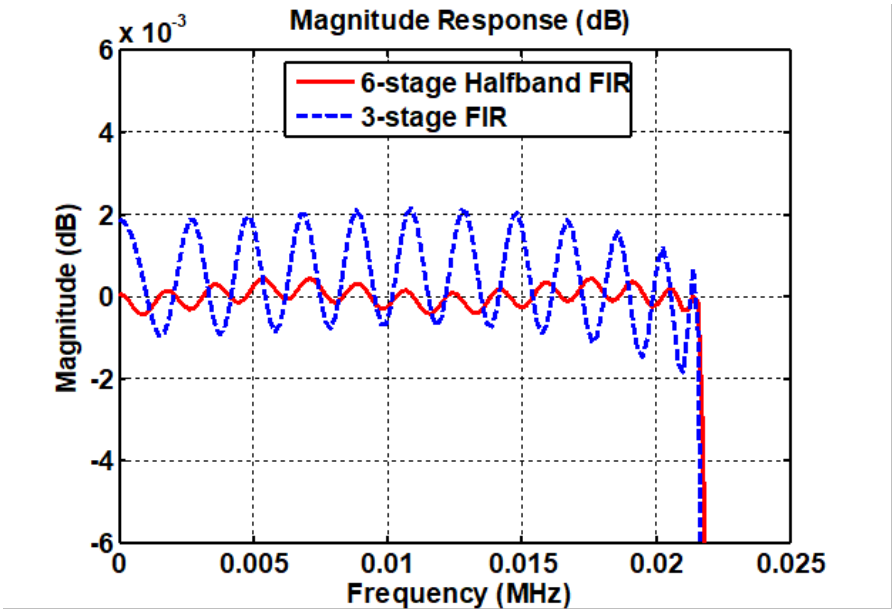


Figure 3.5: Passband performance of 6 stage halfband FIR filter in comparison with reference design

be interesting to see if the quasi-linear phase could help reduce group delay as well. In this case we designed two types of six stage IIR filters.

1. The 6 stage quasi-linear IIR filter with quasi-linear phase on each stage.
2. The 6 stage elliptic IIR filter with elliptic frequency response on each stage.

Results and Discussions

The IIR filters perform very well in passband and satisfy the stopband and transition band requirements. Also the design results in a very efficient theoretical implementation cost as shown in Table 3.10. Figure 3.6 illustrates the passband performance of IIR filters in comparison with the reference design.

Figure 3.7 shows that the quasi-linear IIR filter has almost constant group delay around 1514 samples at passband. It has slightly lower group delay than the reference design but still in similar scale.

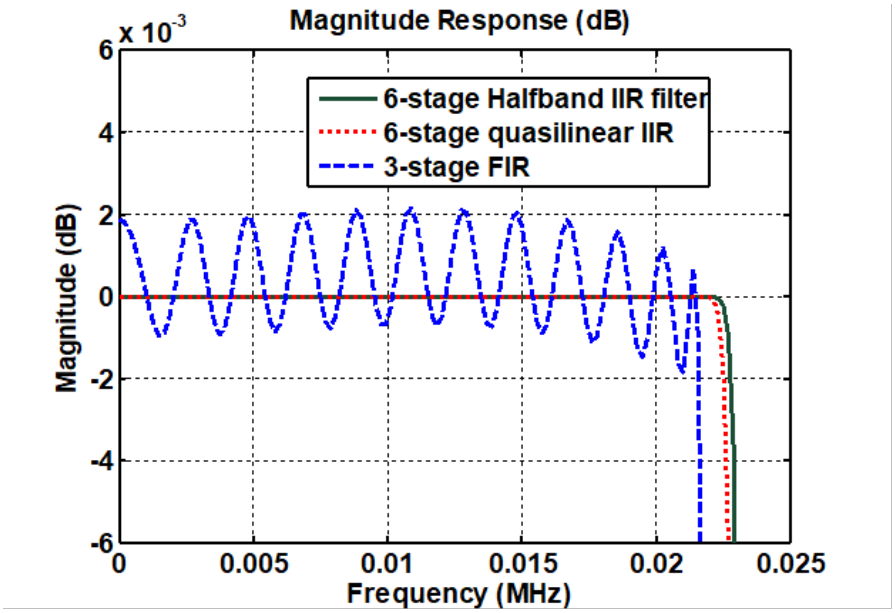


Figure 3.6: Passband performance of 6-stage elliptic IIR and 6-stage quasi-linear IIR decimator in comparison with reference design

The elliptic halfband IIR filter has very low group delay which is far better than the linear phase filters. However it has group delay distortion due to the nonlinearity of filter phase response. As shown in Figure 3.7, the group delay is 176.6 samples at frequency zero and 410.5 at frequency 20 kHz.

3.2.6.5 Multistage Minimum Phase FIR Filters

Design Considerations

Theoretically, the minimum phase FIR filter has the fastest signal response as compared to equivalent nonminimum phase approaches. It would be interesting to see how minimum phase FIR filter performs in the decimation filter design. Based on two effective linear phase filter architectures: "6 stage halfband FIR filter" and "3 Stage FIR", we designed the minimum phase version of these two architectures. We replaced each stage with a minimum phase FIR filter with the same magnitude responses by using a poly-

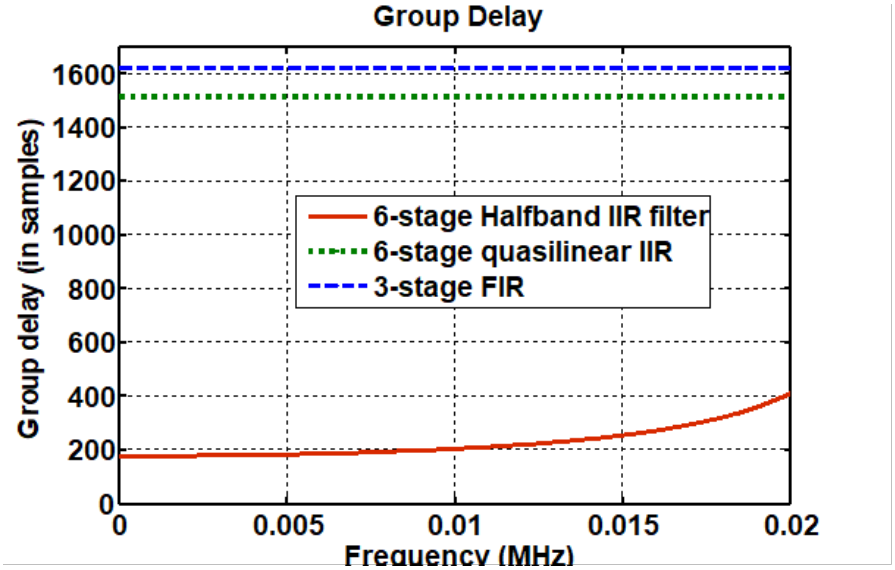


Figure 3.7: Group delay of 6 stage quasilinear IIR and 6 stage elliptic IIR in comparison with reference design

nomial roots finding design algorithm. The minimum order of each stage might not be optimal (actually the optimal minimum phase FIR filter design algorithm has a convergence problem when the filter order is large). However, the algorithm we used has good numerical robustness and produces almost identical magnitude response as the linear phase version even for high order filters.

Figure 3.8 shows the comparison of impulse response (IR) of one stage of linear phase FIR filer and the impulse response of a minimum phase FIR filter which can produce exact magnitude response. The linear IR will be replaced by minimum phase IR in our design.

For the minimum phase 6 stage halfband design, each stage is a minimum phase halfband filter, which decimates the sampling frequency by 2. However in this case the "halfband" is in terms of the frequency response properties. The minimum phase halfband filters do not have the efficient coefficients property as linear phase halfband filters.

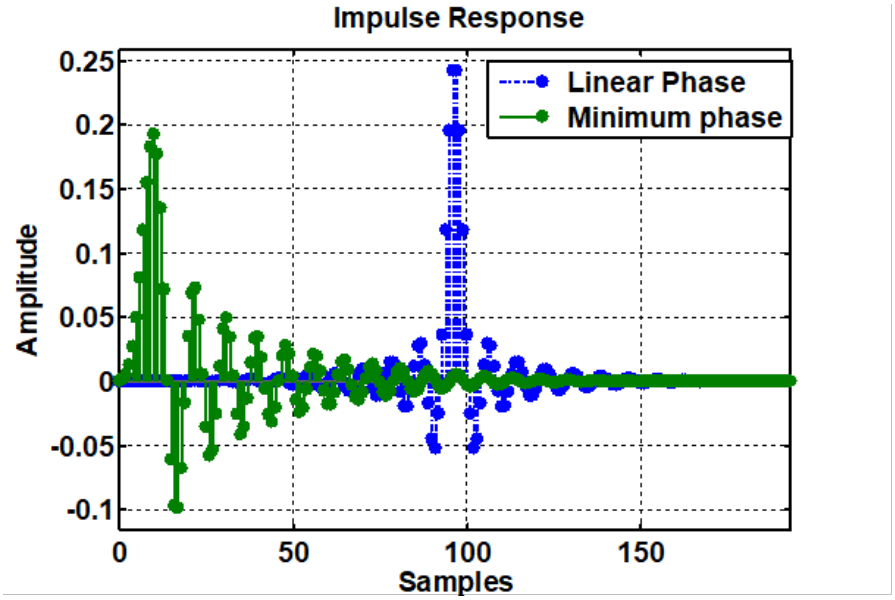


Figure 3.8: Impulse responses of minimum phase FIR and linear phase FIR filters

Results and Discussions

Both 3 stage minimum phase FIR filter and 6 stage minimum phase halfband FIR filter perform very well in time domain in comparison with the reference design, as shown in Figure 3.9. The group delay of the minimum phase 6 stage halfband FIR filter has similar shape as 3-stage minimum phase FIR filter with slightly higher delay (see detail in Table 3.9).

For 6 stage minimum phase halfband FIR filter, the group delay is 164.4 samples at frequency zero and 387 samples at frequency 20 kHz. For 3 stage minimum phase FIR filter, the group delay is 155 samples delay at frequency zero and 380 samples at frequency 20 kHz. Although group delay distortion happens in the 3 stage minimum phase FIR filter, within the audio band, this is equivalent to only 3.5 samples difference at the output sampling rate.

3.2.7 Summary of Group Delay of All Evaluated Filters

Table 3.9 shows the group delays of all the filters we evaluated with the equivalent delay time at output sampling rate. The 3 stage minimum phase FIR decimator, 6-stage minimum phase halfband FIR decimator, and 6 stage multistage IIR decimator perform very well in terms of low group delay. There are some group delay distortions within the passband, as shown in Figure 3.10. There is a trend to high group delay near the Nyquist frequency. However it is only 3 to 4 samples difference in relation to output sampling rate.

Among these three low group delay decimators, the 3-stage minimum phase FIR filter has lowest group delay. The 6-stage halfband IIR filter has lowest theoretical implementation cost (Table 3.10) and the best passband performance (Figure 3.6). However there are complications for practical implementation since more stages normally requires a larger stage control structure.

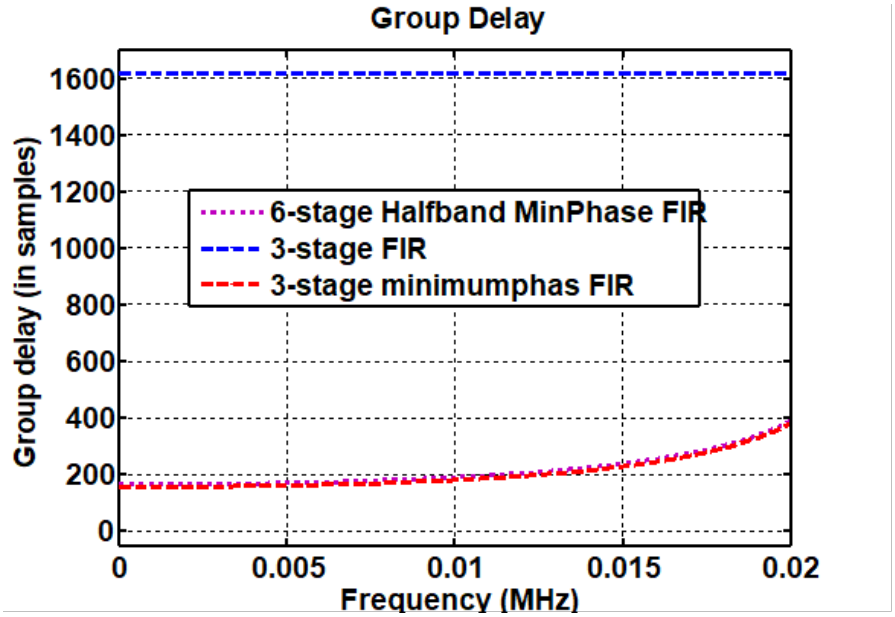


Figure 3.9: Group delay of 6 stage halfband FIR with minimum phase and 3 stage minimum phase FIR

Table 3.9: Group delay of different evaluated filters

Group delay for linear phase filters		
Filter	**Group delay (samples)**	**Delay at 48 kHz(μs)**
Kaiser	1829	595
Eqrip	1511.5	492
3-stage	1619.5	527
CIC	598.5	195
CICom	4022.5	1309
6hb	1961	638
6IIRlin	1514	493
Group Delay for nonlinear phase filters		
Filter	**Group delay (samples)**	**Delay at 48 kHz (μs)**
3-min	155 - 380	50 - 123
6hbmin	164.4 - 387	53 - 126
6IIR	176.6 - 410.5	57 - 134

3.2.8 Compare Cost and Signal to Noise Ratio

Table 3.10 shows the theoretical implementation cost of ten filters evaluated. In the table the "**NM**" indicates "Number of Multipliers", "**NA**" indicates "Number of Adders", "**M/I**" indicates "Multiplications per Input Sample", and "**A/I**" indicates "Additions per Input Sample".

In order to verify whether the different decimation filter architectures affect the overall SNR of the ADC system, we converted the designed decimation filters into Matlab Simulink model blocks. The decimation block processes the simulated 1-bit first order $\Delta\Sigma$ modulator output, and outputs PCM data. The input signals are full amplitude sinusoid waveform with different frequencies, and the output data is calculated by FFT-based SNR estimation [152]. Table 3.11 shows the SNRs at three frequencies at typical

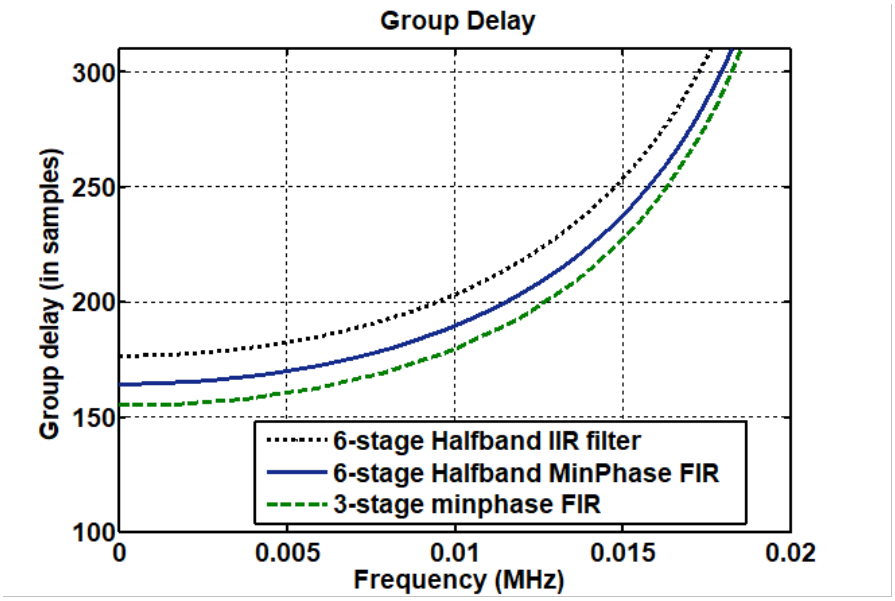


Figure 3.10: Group delay of 6-stage halfband FIR, 3 stage minimum phase FIR, and 6 stage halfband IIR filters

Table 3.10: Implementation cost of different filters

Filter	**NM**	**NA**	**M/I**	**A/I**
Kaiser	3659	3658	57.1719	57.1562
Eqrip	3024	3023	47.25	47.2344
3-stage	243	240	8.5938	8.3906
3-min	249	246	8.9688	8.7656
CIC	1	38	1	19.2969
CICom	109	145	2.6875	20.9688
6hb	96	90	6.9531	5.9688
6hbmin	174	168	10.9531	9.9688
6IIR	19	38	1.6719	3.3438
6IIRlin	33	66	1.9062	3.8125

low, mid and high audio band. It shows that there are no significant differences between different types of decimation filters.

Figure 3.11 shows the Matlab Simulink model of a decimation subsystem which consists

of 6 cascaded minimum phase halfband filters. Figure 3.12 shows the step response of three minimum phase decimators (the second to fourth display) in comparison with linear FIR decimator (at the top display). The one grid of X axis is simulated time of 0.5 millisecond. It shows linear decimator has around 0.5ms latency, whereas the minimum phase ones are shorter.

Table 3.11: Simulated SNR values

Simulated SNR for selected filters			
Filter	**Frequency of Input signals**		
	500 Hz	**3000 Hz**	**12000 Hz**
Eqrip	-120.5733	-107.411	-107.177
3-min	-120.6725	-107.3478	-107.1542
6hbmin	-120.8044	-107.3686	-107.1479
6IIR	-120.8078	-107.3693	-107.1469

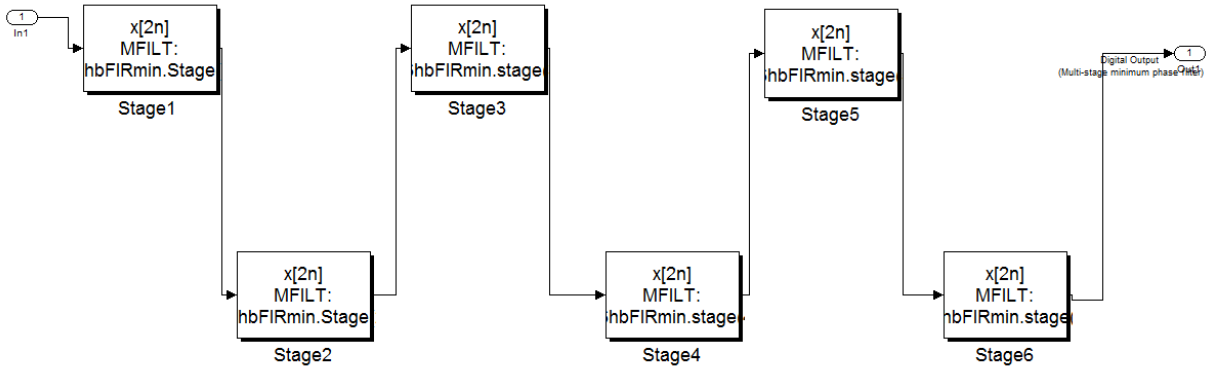


Figure 3.11: Simulink model for a subsystem of cascaded 6 stage halfband minimum phase filters

3.2.9 Conclusion of the Section

In this section, we evaluated time domain performance of different decimation filter architectures that can be used in high resolution $\Delta\Sigma$ ADC. Ten filters were designed based

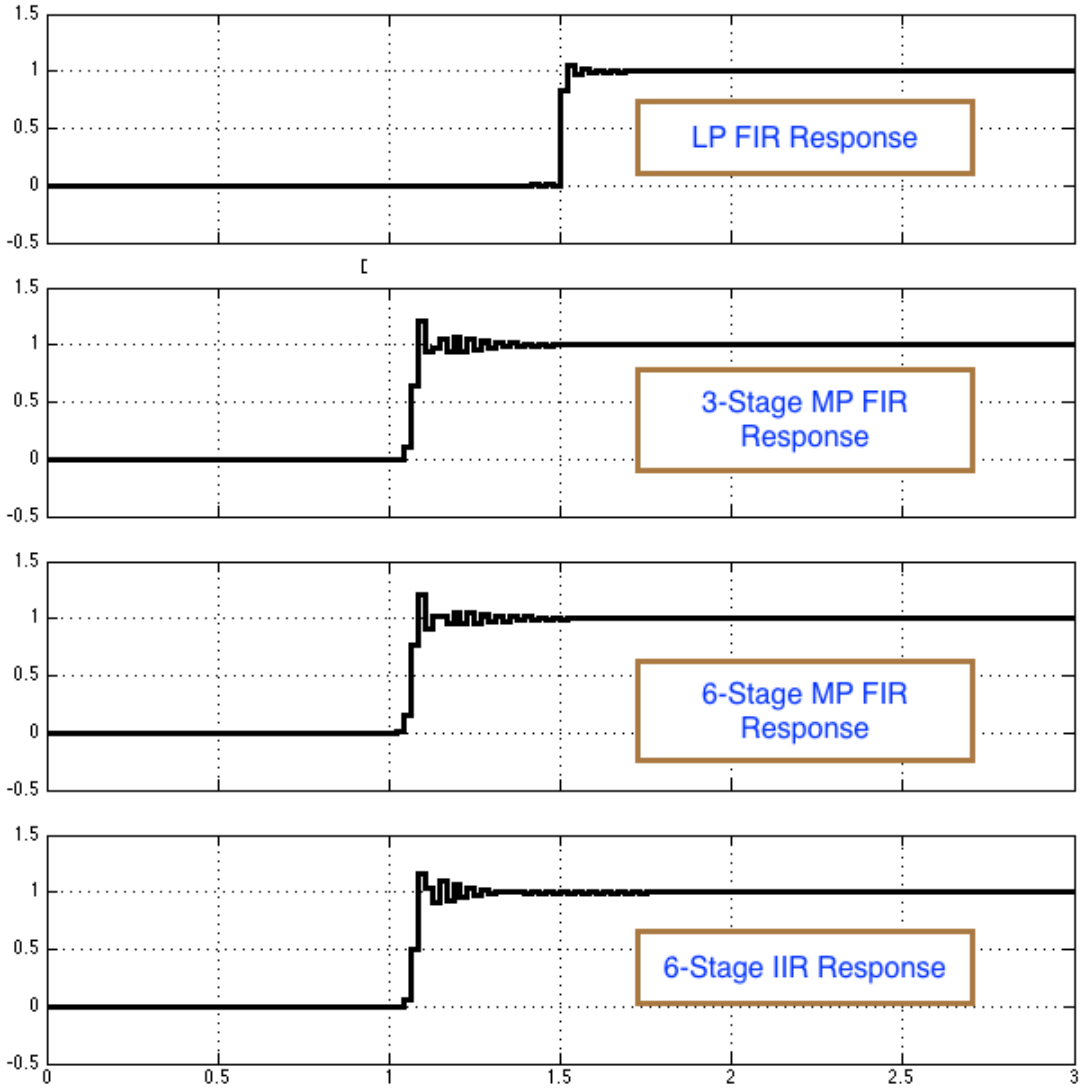


Figure 3.12: Simulated step responses of Linear phase FIR, 3-stage minimum phase, 6-stage halfband FIR, and 6-stage halfband IIR filters

on the typical anti-aliasing 64 times decimation filter design specifications. The group delay properties of both linear phase and non-linear phase multistage filters were investigated in consideration with other frequency performances such as passband, stopband and transition band.

The analysis showed that the multistage minimum phase FIR filter and multistage polyphase IIR filter are promising for low group delay audio applications. The group delay increases near the Nyquist frequency, but this might not be a problem for some live audio applications. The theoretical implementation costs were listed. However, these results were just for typical reference designs. There are vast amount of methods and techniques being developed in optimisation of filter design and realisation, such as the optimal minimum phase FIR filter design method [160, 103]. For halfband FIR filter design, minimum phase filter design without altering the linear impulse response [59, 161] could be interesting to consider. It would be interesting for authors to further research some of these specific areas.

Simulated SNR for typical architectures were evaluated as well. But in real hardware implementations, the effects of quantization of coefficients needs to be further investigated. There are also other practical factors such as hardware and software architectures, which might influence the tradeoff and selection of decimation filters.

This section also revealed that the different multistage design parameters do affect the overall delay such as the number of stages N, the different sample rate alteration factor D_i as described from Eq. (2.13) to (2.18). Traditionally most of these parameters are selected in order to optimise the computational and area cost. Next section we will use analytical approach and developed simplified models to show the relationships between multistage filter design parameters, cost and overall delay.

3.3 Simplified Model of Analysis Design Parameters of Optimal Multistage Multirate Filters

In this section, we analyse the classical approaches of design optimal multistage and multirate filter for $\Delta\Sigma$ ADC/DAC. Based on their performance properties, we propose simplified models of searching optimal distribution of D_i in the design space that yields optimal design. There are two parts of the work:

1. Simplified integer-based design tables for sampling rate alteration up to more than 5000 with balanced design selection algorithm in terms of computational and area cost.
2. Mathematical model of latency as a function of filter design parameters and their optimal design properties for group delay.

3.3.1 Description of the Background Theory

Multistage filter design is a complex multidimensional optimisation problem. The formulae for optimal design generally yield non-integer real numbers for the sample-rate-changing factors of multiple stages. Approaches yielding useful integer results have high computational cost and do not consider important multistage filter design properties. We develop a simplified algorithm for directly searching the optimal integer results. Considering the most useful practical design parameters, optimal results can be approximated with a limited number of sets for any designs satisfying certain constraints, with negligible costs. This vastly simplifies the complexity of the problem.

[84, 88] presented the theory and quantification of cost optimisation of multistage structures. [89, 90] found that optimal solutions can be derived analytically by taking the partial differential equation (PDE) of the cost function, hence reducing it to a one dimensional problem without needing complex numerical search algorithms. However optimal solutions are often groups of non-integer real numbers that cannot be implemented in practical systems. Manual adjustment of results is needed, one still needs to retreat to numerical methods to solve the equations, and for each design, the roots of the

equation must be put back into a cost function to find the optimal solution set. Alternatively, one can yield the integer solution directly. [162] represents this problem in the integer domain using set theory, and then performs integer factorization. [163] showed that the problem can be solved using exhaustive search or a genetic algorithm.

We show that properties of solutions allow simplification of the search algorithm. Based on distribution of the solution sets, we propose a new search algorithm and use it to generate optimal solution lookup tables for practical designs. A balanced trade-off strategy is developed to find a best solution set for both computational and memory area cost. Conclusions are given in the toward the end of the section.

Recall in Chapter 2, Section 2.5, we showed the computational cost can be calculated by R_t as in Eq. (3.4)

$$R_t \cong D_\infty \left(\frac{\delta_p}{k},\ \delta_s \right) f_{r0} S \tag{3.4}$$

where S is expressed as in Eq. (3.5)

$$S = \frac{2}{(\Delta f \prod_{j=1}^{k-1} D_j)} + \sum_{i=1}^{k-1} \frac{D_i}{(\prod_{j=1}^{i} D_j)(1 - \left(\frac{2-\Delta f}{2D}\right) \prod_{j=1}^{i} D_j)} \tag{3.5}$$

We can further have the total memory storage cost N_T of such a filter

$$N_T = \ D_\infty \left(\frac{\delta_p}{K}, \delta_s \right) GT \tag{3.6}$$

Where G is a proportionality constant that relates to the implementation of filter coefficients and T is given by

$$T = \ \frac{2}{\Delta f} \ \frac{D}{\int_{j=1}^{K-1} D_j} + \ \int_{i=1}^{K-1} \frac{D_i}{1 - \alpha \int_{j=1}^{i} D_j} \tag{3.7}$$

where $\alpha = \frac{2-\Delta f}{2D}$.

To minimise R_T is to minimise S in Eq. (3.5) and to minimise N_T is to minimise T in Eq. (3.7). S and T are only dependent on the D_i and Δf.

[89] and [90] took a PDE approach to cost functions Eq. (3.5) and Eq. (3.7), treating D_i as continues real value. For example, finding D_1 in a 3-stage design can be formulated as a roots finding problem:

$$\frac{\partial T}{\partial D_1} = -\frac{2D}{\Delta f D_1^2 D_2} + \frac{1}{(1-\alpha D_1)^2} + \alpha\sqrt{\frac{2D}{\Delta f}\frac{1}{D_1}}D_2 = 0 \tag{3.8}$$

Often solving Eq. (3.8) requires numeric methods and results in complex and irrational roots.

[162] and [163] directly search integer sets of $\{D_i\}$ that are factors of D, using exhaustive search or a genetic algorithm to produce integer valued optimal results, but did not taking the properties of the real valued solution into account.

We observe the distributions of both real valued and integer valued optimal solution sets. We find both of these follow certain regular patterns. This enables us to vastly simplify the optimisation problem and the size of the problem. The findings are discussed in the following section.

3.3.2 Knowledge-Based Search and Lookup Tables

Observing from the experiments' results of both real-valued and integer valued optimal solutions, there are three important properties of the distributions of optimal solutions for both optimal computational cost and memory storage cost:

(a) $\{D_i\}$ is always in descending order for multistage decimation and in ascending order for multistage interpolation.

The larger value of D_i means the larger decimation or interpolation factors stage i. In order to minimise the narrow transition band effects on high sampling frequency (over-sampled), it is understandable to have larger D_i close to the higher sampling frequency end.

The experiments show that the average real valued solution of smallest value of D_i for 3-stage design is around 2.65 (D_3) and for 4-stage design is around 1.52 (D_4) for

decimation process. These number is close to integer number 2 that is the minimum sampling rate changing factor, and is close to the lowest sampling frequency stage.

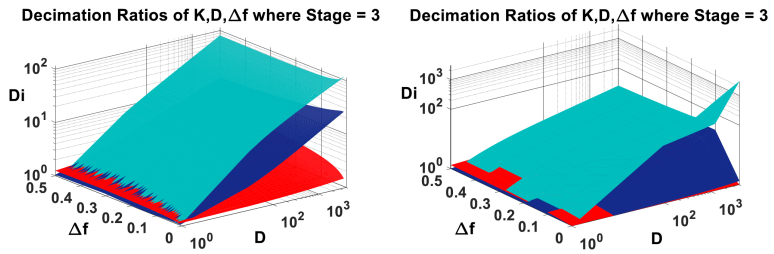
(b) Δf is related to the width of transition band. The variation of Δf changes the order of the filter but not the sampling rate changing factor of each stage.

This is because for same overall value D, the distribution of D_i still follows the trend of property (a) regardless of filter order. According to [89, 90], Δf has only a small effect on the results. Also because the results are not integer, rounding is needed and often, the difference due to using different Δf is smaller than the difference caused by the rounding process.

(c) Because of (a), we need to test the optimal set by cost functions only for highly composite number (non-prime) D that can be factorised more than number of stages K.

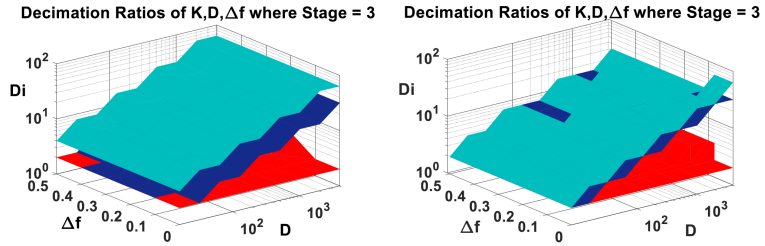
Thus, search for integer valued solutions can be informed by (a), and because of (b) and (c), the problem size is considerably small than it appeared to be.

Figure.3.13 shows 3D plots of the 3 stage real-valued optimal solution distribution for optimising R_T (Figure.3.13.a) and N_T (Figure.3.13.b). The real valued optimal result sets $\{D_1, D_2, D_3\}$ are formed from three independent disjoint surfaces with different $D < 5000$ and $0 < \Delta f < 0.5$. The values of lowest surface is close to the minimum value of interpolation and decimation factor 2. The optimal integer valued solutions follow a similar trend. Figure 3.14 demonstrates this for $D = 2^n$.



(a) (b)

Figure 3.13: Real-valued solution sets distribution



(a) (b)

Figure 3.14: Integer solution sets distribution for $D = 2^n$

The simplified search algorithm is described in Table 3.12 :

Table 3.12: Pseudocode of optimal integer valued solution search algorithm

For $D \in \mathbb{N}_1$ and represented as a prime factorization:
$D = p_1^{d_1} * p_2^{d_2} * p_3^{d_3} ... * p_r^{d_r}$ where p_i is a prime number and d_i the corresponding exponent,
1) Check whether D can be factorized into M unique sets of k (stage) factors $M = \left\{ \begin{array}{c} D_{1.1},\ D_{2.1},\ ...\ D_{k.1} \\ D_{1.2},\ D_{2.2},\ ...\ D_{k.2} \\ \vdots \\ D_{1.m},\ D_{2.m},\ ...\ D_{k.m} \end{array} \right\}$
2) Sort M so that $\{D_{i.1} > D_{i.2} > ... > D_{i.k}\}$
3) Substitute sets into Eq. (3.5) or Eq. (3.7) and find the set with the minimum solution.

The algorithm simplifies the search since we only care about unique sets (e.g., {8, 4, 2} is equivalent to {4, 8, 2}), and we sort candidate sets as descending ordered sequences. Only when the number of D's factors is larger than the required number of stages, the step 2) sorting required. For example, for $D < 5000$, 3 stage decomposition, only 1692 (33.8%) numbers can be factorised in different unique sets of 3 factors that need to be put back into the cost function. For typical design value $\Delta f = 0.18$ and $D < 5000$ and

2, 3, 4 stage design, our method provides 85.4% average reduction when compared with exhaustive search in terms of the number of cost function tests, and 65% computing time reduction. Computing time was averaged over 100 iterations using a standard Intel Core i7 based PC.

In addition, the variation of Δf does not cause much change in the optimal integer values. For the same example of $D < 5000$ and 3 stage case, within the 1692 cases that have possible multiple solutions, in 994 (59%) cases for computational cost optimisation and 1167 (69%) cases for memory cost optimisation, the optimal solution sets change only once or twice over the (0 to 0.5) Δf region.

Table 3.13 summarises 15 popular ADC/DAC chipsets used in audio devices from on-board sound cards to professional mixers. The supported sampling frequencies F_s range from 44.1 kHz, 48 kHz, 96 kHz with different supported filter type configurations such as low latency, sharp or slow roll off, etc. Δf is within 0.08-0.44 with most common designs between 0.15 and 0.4.

We use a bisection method to find the Δf points where optimal solutions change. For common 2, 3, 4 stage filter design, two groups of lookup tables for computational and area costs can be generated from the algorithm. They are small since optimal solutions are smooth over usable design specification ranges.

Figure 3.15 shows changing of optimal solution sets $\{D_i\}$ against *D* and *f*. The *Z* axis value is calculated from Eq. (3.9), where ω is a weighting factor, and $\sigma\left(D_i, K\right)$ is the standard deviation of optimal set $\{D_i\}$. It provides information regarding both $\int D_i$ and the distribution value of D_i within a solution set. Since the elements of set $\{D_i\}$ are descending or ascending, $\sigma\left(D_i, K\right)$ indicates the slope of the value changing across stages. The same value of Z indicates same set of D_i being chosen.

$$Z = \omega\ D + \sigma\left(D_i, K\right) \tag{3.9}$$

Figure 3.15 shows that optimal integer solution sets can be the same values for a large range of design specification Δf. Similar figures can be produced for both optimal area cost and computational cost sets. Thus, we can create lookup tables to store these optimal solutions with the critical values of Δf that cause changes in optimal solution

Table 3.13: Summary of values of common oversampling-based audio ADC/DAC design specifications

Type	**Max f_p**	**Min f_s**	**Min Δf**
AD1871	0.45 F_s	0.55 F_s	0.17
AD7768	0.43 F_s	0.50 F_s	0.14
AD1974	0.44 F_s	0.56 F_s	0.21
ADAU1966A	0.36-0.45 F_s	0.55-0.64 F_s	0.18-0.44
PCM1807	0.45 F_s	0.58 F_s	0.22
PCM4220	0.42-0.45 F_s	0.55-0.58 F_s	0.18-0.28
PCM1794A	0.46-0.49 F_s	0.55-0.73 F_s	0.11-0.37
PCM5242	0.40-0.47 F_s	0.55 F_s	0.15-0.18
CS5364/66/68	0.45-0.47 F_s	0.58-0.68 F_s	0.19-0.34
CS5381	0.45-0.47 F_s	0.58-0.68 F_s	0.19-0.34
CS4398	0.499 F_s	0.55-0.58 F_s	0.09-0.14
WM8740	0.27-0.45 F_s	0.46-0.49 F_s	0.08-0.41
WM8741	0.4 F_s	0.5 F_s	0.2
ALC885	0.45 F_s	0.60 F_s	0.25
CS4207	0.45-0.499 F_s	0.55-0.60 F_s	0.09-0.25

sets. The structure of a database of such tables is depicted in Figure 3.16. Using our algorithm to generate K= 2, 3, 4 stage design tables for both memory and computational cost, with $1<D<5000$ and $0 < \Delta f < 0.5$, there are 15,783 total optimal computational cost sets and 17,622 total optimal memory usage sets in the three tables.

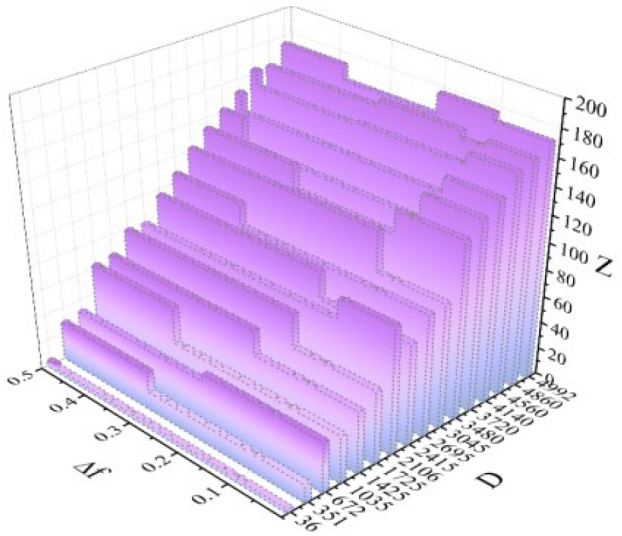


Figure 3.15: Changing of optimal value against Δf for some highly composite number D

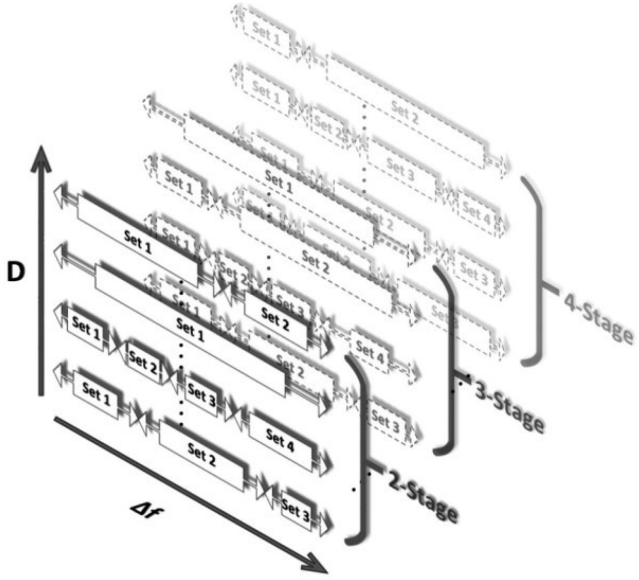


Figure 3.16: Depiction of optimal solution sets lookup tables of computational cost or memory storage cost

3.3.3 Tradeoff Strategy for Minimisation of Both Area and Computational Cost

In practise, only one solution can be used in a system. A trade-off strategy based on error effects was developed to find a best solution set for both computational and memory area cost within the lookup tables.

Table 3.14: Cost balance search algorithm

when $D_{i\,m} \neq D_{i\,c}$
if $\frac{C_m - C_c}{C_c} > \frac{M_c - M_m}{M_m}$ choose $D_{i\,c}$
else choose $D_{i\,m}$

Where:

- $D_{i\,c}$ is the integer valued solution set for optimal computational cost.
- $D_{i\,m}$is the integer valued solution set for optimal memory usage cost.
- C_c is computational cost with optimal computational cost set.
- C_m is computational cost with optimal memory usage set.
- M_c is memory usage with optimal computational cost set.
- M_m is memory usage with optimal memory usage set.

To evaluate the error effects of this trade-off strategy, we evaluated the average error of computational cost: S_{diff}=($\int_{i=1}^{N} \frac{C_i - C_{ci}}{C_{ci}}$)/$N$, and the average error of memory cost: T_{diff}= ($\int_{i=1}^{N} \frac{M_i - M_{mi}}{M_{mi}}$)/$N$, where N is total number of design cases tested; C_i and M_i are the actual computational cost and memory usage when the trade-off sets being used; C_{ci} and M_{mi} are the computational cost and memory usage when the corresponding optimal sets being used.

For 3 stage design with $0<\Delta f<0.5$ and $0<D<5000$, the average error is 0.26% for S_{diff} and 8.88% for T_{diff}.

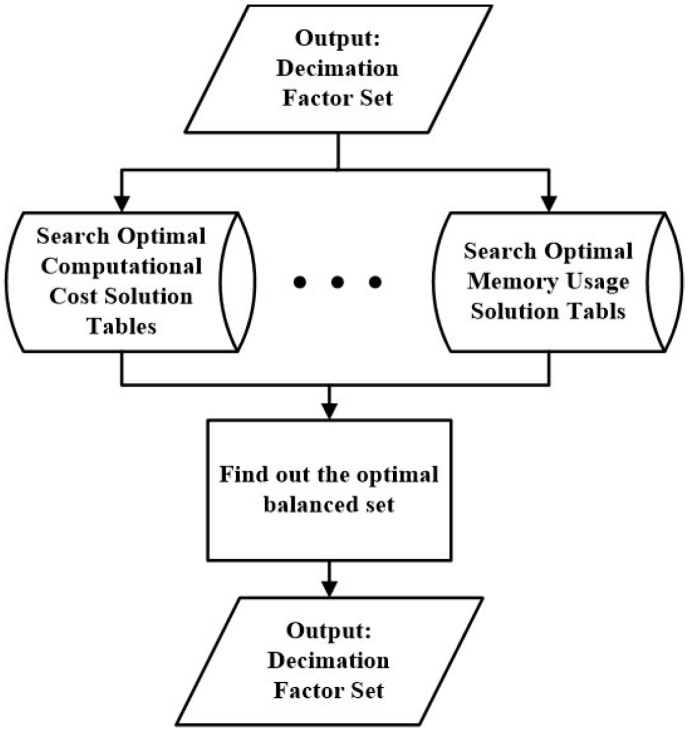


Figure 3.17: Flow chart of overall database query algorithm

Figure 3.17 shows the flowchart for finding semi-optimal solutions for both constraints. It is worth mentioning that this method can be altered with extra weighting factor to consider actual implementation effects. The design parameter Δf, D and different platform realisation techniques might have influence of actual selection decisions. Further work needs to be done to decide the form of weighting factor. A numeric approach has been adopted to simplify the design space for optimal cost and area design. Next we use analytical approach to find the optimal design in terms of overall delay.

3.3.4 Model of Delay of Multi-Stage Linear Phase Filter

First we show the analytical approach of how multistage decimation filter design affects the overall latency. In previous sections we uses computational cost as objective function, since the computational cost of filter is directly linked to filter order hence the design specifications. For linear phase FIR filter, the group delay is also directly linked

to filter order, therefore it should be possible to have the analytical expression of overall delay as a function of design specifications for linear phase filter.

3.3.5 Formulae of Total Delay of Linear Phase FIR Based Multistage Design

The delay of single stage linear phase FIR filter with symmetric (either odd or even) coefficients structure is equivalent to the number of samples of the half of filter order: "$N/2$". The order N can be estimated by the filter design specifications such as passband frequency f_p, stopband f_s, passband ripple δ_p, stopband ripple δ_s . For multistage design, each stage can be a single linear phase FIR lowpass filter with overall filter performances specified as below:

1. Overall passband frequency f_p, and stopband frequency f_s.
2. Overall passband ripple δ_p and stopband ripple δ_s.

The delay T_i of stage "i" is defined as:

$$T_i = \frac{N_i}{2f_{r(i-1)}} \tag{3.10}$$

where N_i is the number of order at stage "i".

The total delay can be defined as:

$$T_D = \sum_{i=1}^{k} T_i \tag{3.11}$$

The N_i can be approximately estimated as the following formula, if D is greater than 10 and relatively narrow band filter according to [84]

$$N_i = \frac{D_\infty(\delta_p/k, \delta_s)}{\Delta F} = \frac{D_\infty(\delta_p/k, \delta_s)L_i f_{r(i-1)}}{f_{ri} - f_s - f_p} \tag{3.12}$$

Therefore the T_i can be approximately defined as following formula:

$$T_i = \frac{D_\infty(\delta_p/k, \delta_s)}{2(f_{ri} - f_s - f_p)} \tag{3.13}$$

Substitute Eq.(3.13) into Eq.(3.11), we have the formula of total delay of multi-stage decimator T_d

$$T_d = \frac{1}{2} D_\infty(\delta_p/k, \delta_s) \sum_{i=1}^{k} (\frac{f_{ro}}{\prod_{j=1}^{i} D_i} - f_s - f_p)^{-1} \tag{3.14}$$

The D_k is only independent up to D_{k-1} the Eq.(3.14) can be further refine to equation Eq. (3.15) below.

$$\begin{aligned} T_d &= \frac{1}{2} D_\infty(\delta_p/k, \delta_s) P \\ P &= (f_s - f_p)^{-1} + \sum_{i=1}^{k-1} (2D \frac{f_s}{\prod_{j=1}^{i} D_i} - f_s - f_p)^{-1} \end{aligned} \tag{3.15}$$

Now we have the final objective function of total delay in Eq.(3.15) in a closed form. Next we will show some important properties of this formula analytically.

3.3.6 Properties of Delay Formula for Multistage Linear Phase Filter Design

In this section, we show the two important properties of delay formula Eq.(3.15)

(a) The overall delay increases as the number of stages increases.

Although the total number of order of filter reduces around 10x folds when using optimal multistage design, the overall delay increases. It is due the later stages have lower input sampling frequencies. Therefore, the delay caused by longer sampling period at later stage overwrites the gains from reduced filter length.

This can be proved below. The formula Eq.(3.15) can be rewritten into the summation of two parts A and B. Surely the number of stage k is positive integer number.

$$T_d = A + B \tag{3.16}$$

Where A is defined as:

$$A = \frac{1}{2} D_\infty(\delta_p/k, \delta_s)(f_s - f_p)^{-1} \tag{3.17}$$

And B is defined as:

$$B = \frac{1}{2} D_\infty(\delta_p/k, \delta_s) \sum_{i=1}^{k-1} (2D \frac{f_s}{\prod_{j=1}^{i} D_i} - f_s - f_p)^{-1} \tag{3.18}$$

When *k=1*, we only have part A as the single stage delay:

$$T_d = A = \frac{1}{2} D_\infty(\delta_p, \delta_s)(f_s - f_p)^{-1} = \frac{1}{2} D_\infty(\delta_p, \delta_s) \frac{1}{\Delta F f_{ro}} \tag{3.19}$$

Which is equivalent to the half of number of order when $N >> 1$, and it does not depend on overall decimation factor D.

When $k > 1$; refer to the "TABLE I" in reference [84] (Crochiere 1975), for given δ_p and δ_s, the $D_\infty(\delta_p/k, \delta_s)$ increases when *k* increasing. Therefore A is increasing. Part B is the running sum of $(2D \frac{f_s}{\prod_{j=1}^{i} D_i} - f_s - f_p)^{-1}$, the maximum value of $\prod_{j=1}^{i} D_i$ is *D*, therefore we can proof this running sum is always greater than 0, so does the value of part B. The overall T_d is increasing. Therefore, when the $k > 1$, the delay increases when the k increases.

The second property of delay formula Eq.(3.15) is:

(b) Given fixed stage "k", the overall delay tends to be smaller when distributing larger decimation factor D towards the later stage.

We regard the decimation factor *D* is a composite number which can be factorised into the number *k* of D_i. The single stage delay in formula Eq.(3.13) can be further written as:

$$T_i = \frac{A}{f_{ri} - B} \tag{3.20}$$

Where both A, and $B > 0$. The delay is inversely proportional to the input sampling frequency. The sketch is below:

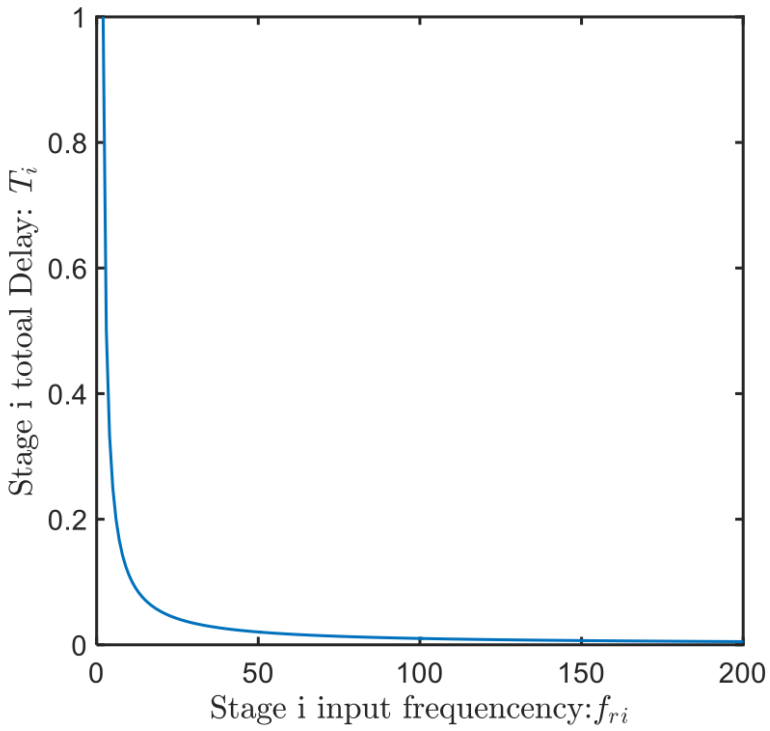


Figure 3.18: Delay vs Input Frequency

Therefore, given the number of stage k is fixed, for each stage, it would be have maximum $f_{ri}$to have minimum T_i. The maximum value of f_{ri} is f_{r0}. The $D_i$should be "1" for that stage. The minimum delay should be the form of $[1, 1, 1..., f_{r0}/f_{rk}]$, which is equivalent to single stage filter.

For the multistage structure where $k >$1 and $D > 1$, both k and D will be whole number. Let's take the partial derivative of equation Eq.(3.20) with respect to f_{ri}:

$$\frac{\partial T_i}{\partial f_{ri}} = \frac{-A}{(f_{ri} - B)^2} \tag{3.21}$$

The sketch is below:

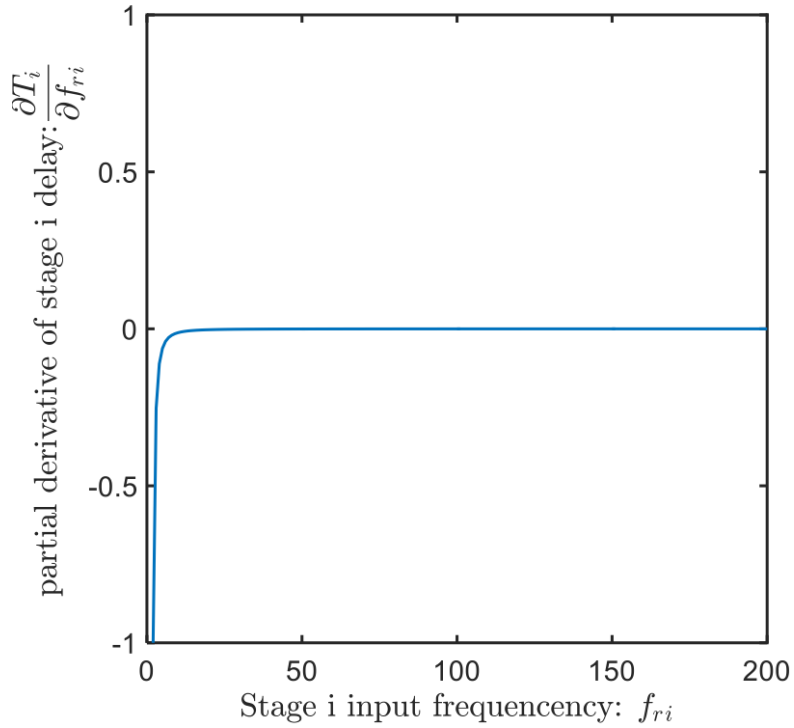


Figure 3.19: Partial derivative of single stage delay vs. input frequency

For $(f_{ri} > f_s + f_p)$, the partial derivative is regarded as gradient: ∇T_i . If we regard the multistage decimation process as adding number of frequencies together to reach the highest input frequency f_{r0} from f_{rk}. This basically shows when a small stage of frequency increasing df is needed, to have lower increment of T_i, the df should be put toward lower end of f_{r0}. For example, in theory when $k > 1$ and $D >$1, the form of decimation to have smaller total delay will be $[2, 2, 2...remains]$.

Next we show the simulation and practical results match these two properties.

3.3.7 Simulation Results

3.3.7.1 Delay Effects of Number of Stages

Table 3.15 shows a typical 64 x 48 kHz decimation audio linear phase filter with different number of stages design to achieve same magnitude response performance. We can see the delay increases when the stage increases. However with multi-stage design the number of adds reduces significantly from single stage to multistage.

Table 3.15: Delay effects of number of stage - linear phase

Filter type	Group Delay at f_{r0}	Number of Adders
Optimal design 1 stage	1511.5	3023
Optimal design 3 stage	1619.5	240
Half-band design 6 stage	1961	90

Table 3.16 shows a typical 64x decimation audio minimum-phase filter with different number of stages design to achieve same magnitude response performance.

Table 3.16: Delay effects of number of stage - minimum-phase

Filter type	Group Delay at f_{r0} from 20 – 20kHz	Number of Adders
Minimum phase FIR 3 stage	155 – 380	246
Minimum phase FIR 6 stage	164.4 – 387	168
Minimum phase IIR 6 stage	176.6 – 410.5	38

Observations

1. The results generally match the first property of the formula. It shows the increases of number of stages has negative effects of latency.
2. The computational cost reduction from single stage to multistage is significant which may surpass the benefits of reduced delay gained by single stage.

3. For multistage designs, it generally shows the fewer stages has advantages of lower delay.

4. There is significant delay reduction by using multistage minimum phase design which provide both superior computational cost and overall delay. However the delay is uneven for at the different frequencies.

3.3.7.2 Delay Effects of Distribution of Decimation Factors Within the Stages

The simulation is run at the design specifications with passband ripple $\delta_p = 0.01$; stop-band attenuation $\delta_s = -80dB$, and decimation factor $D >> 10$, and $N$is estimated number of order, so the approximation in the formulae in previous sections can be valid.

Table 3.17: Simulation result D=32, k =1 and k=2

No. of Stage k=1		Number of stages k=2; ($D_1 \times D_2 = 32$)			
D=32	N/A	(2, 16)	(4, 8)	(8, 4)	(16, 2)
N (Order)	1565	837	433	254	234
Delay (No. of sample at f_{ro})	782	834.5	840.5	855	897

Table 3.18: Simulation result K= 3, D=32

Number of stages k=3; ($D_1 \times D_2 \times D_3 = 32$)					
(2, 2, 8)	(2, 4, 4)	(2, 8, 2)	(4, 2, 4)	(4,4,2)	(8,2,2)
866.5	885.5	931.5	889	937	947

Table 3.19: Simulation result D=64 K=2

Number of stages k=2; (D_1 x D_2=64)				
(2, 32)	(4, 16)	(8,8)	(16,4)	(32,2)
1666.5	1672	1683	1709.5	1793.5

Table 3.20: Simulation result D=64 K=3

Number of stages k=3; (D_1 x D_2 x D_3=64)								
(2, 2,16)	(2,4, 8)	(2,8,4)	(2,16,2)	(4,2,8)	(4,8,2)	(8,2,4)	(8,4,2)	(16,2,2)
1723.5	1736.5	1768.5	1860.5	1736.5	1866.5	1779.5	1875.5	1894.5

If we look at Table 3.20, the second row indicates how the decimation factor D_i is distributed. We will see the delay of $D_1 = 2$, $D_2 = 4$, $D_3 = 8$ is 1768.5 which is lower than the delay of $D_1 = 8$, $D_2 = 4$, and $D_3 = 2$ which is 1875.5. However it is not significant difference. Therefore if the distribution of decimation factor D_i is selected by the optimal design choice, there is no significant effects on overall delay as minimum phase does.

Observations

1. This result generally agrees with the second property of delay formulae, which shows the distribution of larger decimation factor at later stage will result in shorter delay.
2. Comparing with optimal design by Crochiere [84, 88] and our simplified method in the early part of this section, the distribution of D_i for optimal computational cost normally have larger D_i at earlier stage, which is the contradictory requirements for set D_i as to optimal delay.

Next, we present the new global balanced design that takes total group delay into account with user-controllable weighting factor by utilising the theory developed here.

3.3.8 Global Balance Design D_i for Both Cost and Delay for 3-stage Design

The set D_i that produces a multistage filter with optimal computational or area cost usually has sampling frequency alteration rate '2' at the last stage of the filter, whereas the set D_i that produces a multistage filter with optimal delay shall have factor '2' at the beginning of the stage. In addition, where there is a factor of '2' decimation or interpo-

lation, a more economic half-band filter can be employed to reduce the cost further. We could develop an algorithm to shift the position of the number within the set D_i to find out the global balanced solution with comparing the delta of the cost and the delay. In addition, a user defined weighting factor can be used to gear the design to produce the results that more computational cost optimal or delay optimal.

This idea has been implemented in 3-stage cases. According to the previous research, 3-stage design normally yields optimal solutions among the different number of stages. To develop this algorithm, an average saving factor of using halfband filter in multistage filter design for High-resolution Anti-aliasing Anti-image Filter (HAAF) is obtained by experiment results. We use number $k_1 = 0.9830$ as the cost-saving factor and number $k_2 = 1.0033$ as the delay saving factor. Let D_i be the output set from the algorithm mentioned in Section 3.3.3 that produces the balanced cost of optimal design. Let β represents the user weighting factor of minimising group delay design. We have the following algorithm that outputs the set D_{gi} as the global balanced solution set for both cost and group delay, based on a condition function in Eq. (3.22).

$$(2 - k_1(\frac{T_2}{T_1} + \frac{S_2}{S_1})) \times 0.5 - \frac{(P_1 - P_2 \times k_2) \times \beta}{P_1} > 0 \tag{3.22}$$

In Eq. (3.22), T represents the value of filter storage or area cost as in Eq. (3.7). S represents the value of filter computational cost as in Eq. (3.5). P represents the value of filter group delay as in Eq. (3.15).

Table 3.21: Global balanced solution set search algorithm

1. Obtain β from user input as well as the filter design specifications.
2. Obtain the balanced solution set D_i using previous algorithm as in Table 3.14.
3. Where there is '2' in the set, shift the position of this factor in the solution set and make new set.
4. Compare the two solution sets using condition equation in Eq. (3.22).
5. If the condition is met, select set 1, otherwise select set 2.

Testing Results

We run this algorithm for D up to 2000 for possible 3-stage filter design. The results are shown in Table 3.22 list the first 10 values in the database. Each column represents the optimal solution set for different cases. From the left to the right, there are optimal solution sets for 'Minimum computational cost', 'Minimum area cost', 'Balanced Cost' for both computational and area cost, 'Minimum Group Delay', and 'Global' solution that considering all the factors. The rightmost column are the solution sets generated by the algorithm in Table 3.21.

Table 3.22: Global balanced results

D	Computation	Area	Balance	Groupdelay	Global
64	(8, 4, 2)	(8, 4, 2)	(8, 4, 2)	(2, 2, 16)	(8, 2, 4)
66	(11, 3, 2)	(11, 3, 2)	(11, 3, 2)	(2, 3, 11)	(11, 2, 3)
70	(7, 5, 2)	(7, 5, 2)	(7, 5, 2)	(2, 5, 7)	(7, 2, 5)
72	(9, 4, 2)	(9, 4, 2)	(9, 4, 2)	(2, 2, 18)	(9, 2, 4)
78	(13, 3, 2)	(13, 3, 2)	(13, 3, 2)	(2, 3, 13)	(13, 2, 3)
80	(10, 4, 2)	(8, 5, 2)	(10, 4, 2)	(2, 2, 20)	(10, 2, 4)
88	(11, 4, 2)	(11, 4, 2)	(11, 4, 2)	(2, 2, 22)	(11, 2, 4)
90	(15, 3, 2)	(9, 5, 2)	(9, 5, 2)	(2, 3, 15)	(9, 2, 5)
96	(12, 4, 2)	(8, 6, 2)	(12, 4, 2)	(2, 2, 24)	(12, 2, 4)
104	(13, 4, 2)	(13, 4, 2)	(13, 4, 2)	(2, 2, 26)	(13, 2, 4)

In this case, we set the user weighting factor that treats the percentage of change of computational cost and delay as the indicator. For the first 10 decompose D_i in Table 3.22, the results of 'Global' solution search algorithm outputs have average 7% increase in computational cost and 6.5% reduction in the overall delay. This validates the algorithm. However, the choice of the weighting factor is really an engineering decision based on the application context.

3.3.9 Summary of Contributions

In this section, firstly we analysed the integer solution sets of classic multistage linear filter design problem. Based on their properties, we proposed a search algorithm within a simplified search space that produces a balanced solution set for both minimising computational and area cost.

Secondly, we have analytically shown the delay effects of multistage filter design of linear phase filters by defining the overall object function of delay as a function of design specifications. This work is inspired by the classic work of Crochiere [84, 88] and Coffey [89, 90] which defined the object function as computational cost. The analysis of the

optimal delay solution sets shows they follow the different distribution pattern and the solution sets for the optimal cost.

Finally, based on the delay objective function, we proposed an algorithm that takes the delay into account for the overall design incorporate with halfband techniques to produce 'Global balanced design'. The test results show that the outputs solutions achieve an overall cost-effective design with considerably reduced group delay. In the next section, we will look into the objective evaluation of minimum phase FIR filters that can be used in the multistage design.

3.4 Performance Evaluation of Minimum Phase Multistage Filter

3.4.1 Defining Delay Measurements of MP Filters

Minimum phase filters do not have constant "delay". It would be nice to have quantitative measures of group delay of the non-linear filters. Next, we define a various delay measurement methods. These methods can be considered to gauge the delay of MP filters.

1. **Group delay** at certain frequency points

This is the most obvious measure. We are interested in the group delay of specific points of interests so that the overall group delay distortion and the flatness of group delay can be gauged. For example, the lowpass MP filter has group delay behaving monotonically over the passband. It would be interesting to see the group delay at the edges of the audio band such as at 20Hz and 20kHz.

2. **Central Time** (Time domain)

"Central Time" or "Centre of Gravity" of a time event of a non-linear phase system is defined in various literatures in both continuous form and discrete form [164, 165, 166]. A normalised first-order temporal moment about time "$t = 0$" is referred to as the Central Time: T, and is defined as Eq. (3.23). This value can be thought of as "the

centroid of the area under the amplitude-squared time history and is directly analogous to the first statistical moment or mean value."

$$T = \frac{\left(\int_{-\infty}^{+\infty} t \times g(t)^2 dt\right)}{\mathrm{E}} \tag{3.23}$$

The discrete form of this equation:

$$T_g = \frac{\sum_{\mathrm{n}=0}^{\mathrm{N}} n\, A_n^2}{\sum_{\mathrm{n}=0}^{\mathrm{N}} A_n^2} \tag{3.24}$$

"Central time" is defined at time domain for the entire impulse response stream. Although it represents the time property of the whole system, it is difficult to evaluate this measurement in the limited frequency band, which in many cases is desired for audio applications.

3. **Delay Centroid** (Frequency Domain)

For audio applications, we are only interested in the audio band (band of interest) delay behaviour. Therefore, like frequency centroid, we can define the delay centroid:

$$\tau_c = \frac{\sum_{i=1}^{N} \tau_i}{N} \tag{3.25}$$

Where N is the length of frequency bin of interested band and τ_i is the group delay at that frequency bin.

3.4.2 The Effects of Corner Frequencies

Observing the group delay behaviour of different lowpass filters. The delay curves almost monotonically increase towards the corner frequencies or cutoff frequencies. For the filters with the same order, the group delay value is almost the same at the designed corner frequencies. For filters designed at same corner frequencies but with different orders, the group delay various very little in the frequency band much lower than the cutoff frequencies. These effects are described in below two experiments.

To get smoother (semi-linear) group delay within the passband, the simplest way is just to push the passband corner frequency higher. Figure 3.20 shows when the cut-off frequency varies from 21k Hz to 25k Hz, the group delay at 20k Hz drops from 10.36 to 3.9. The orders are all 133.

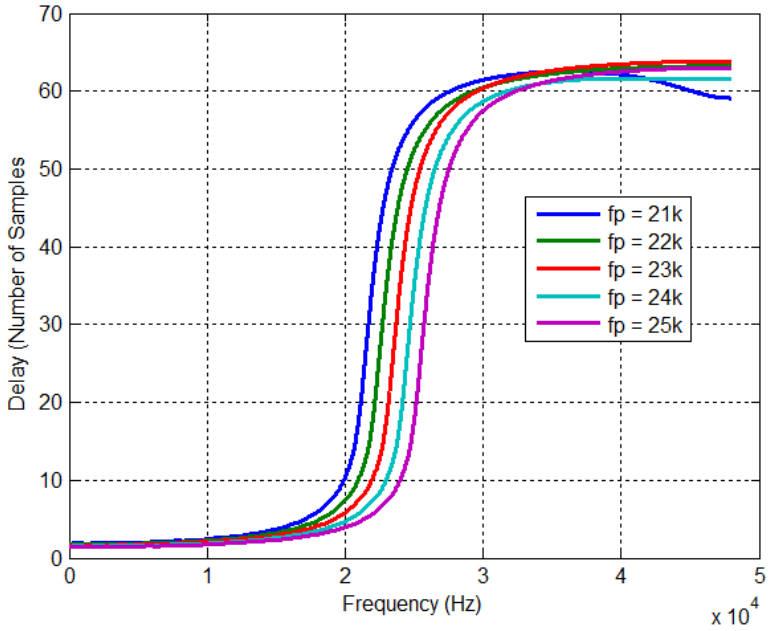


Figure 3.20: Different passband corner frequencies with same transition bandwidth

The values are summarised in the following Table 3.23, we can see the group delay is roughly same at the cut-off point. Providing the shape of the curve, the cut-off point raises significantly comparing the overall section.

Table 3.23: Group delay at Same order with different cutoff

Cut off Frequency (kHz)	Group delay (number of samples)	Group delay at 20 kHz
21	17.93	10.36
22	17.69	7.34
23	17.87	5.71
24	17.66	4.63
25	17.52	3.94

In summary, the group delay is considerably low for MP filter within the passband. The delay curve vs frequency behaves monotonically in general towards the cut-off frequency.

3.4.3 Group Delay Behaviour of MP Filters With Different Order

The filter order is a very important measure of filter performance. The group delay of a linear phase FIR filter is proportional to the order of the filter. The optimal single stage or multistage filter design aims to minimise the order of the filter. Therefore it will be interesting to investigate how group delay varies against the changes of filter order for the minimum phase case.

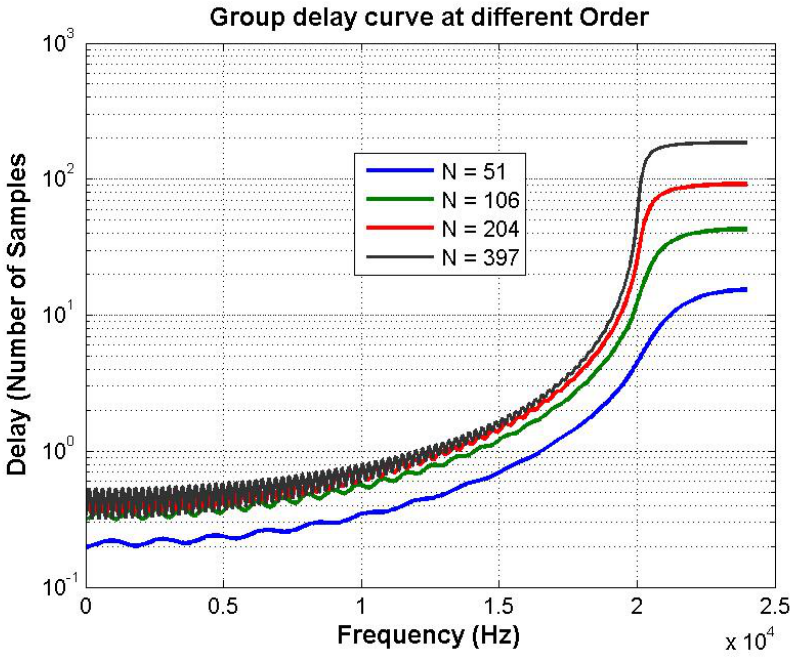


Figure 3.21: Group delay of MP filter at different Order

Group delay behaves monotonically in general over the passband of MP filter, as in Figure 3.21 although for higher order MP filter, there is noticeable ripple at passband.

It is worth to note that the group delay values do not vary much at lower band far away from the cut off frequency. Figure 3.21 shows the group delay values vary only below ONE sample almost at the entire lower quarter of Nyquist frequency band, even when the filter orders are very different.

3.4.4 Quantitative Group Delay Measurements of MP Filters With Different Order

In Figure 3.22 and Figure 3.23, we compared the different quantitative delay measurements of MP filter (curve 2 3 4 5) with a group delay of linear phase filter (curve 1) in a logarithmic scale and linear scale respectively. Figure 3.23 allows us to look closely into the different delay estimation results with the linear phase value being omitted.

We can see that the delay of the minimum-phase filter in all different quantitative measurements are much less than the linear phase design. For the whole audio band 0-18k Hz and with different delay estimation methods, the various measurements of MP filter are very close.

In this experiment, the lowpass filter is designed and converted to a MP filter by the zero reflection method. Then we compare the magnitude response to see if the root mean square error is greater than 0.00001(1e-05) over 4096 points. We got the order up to around 725.

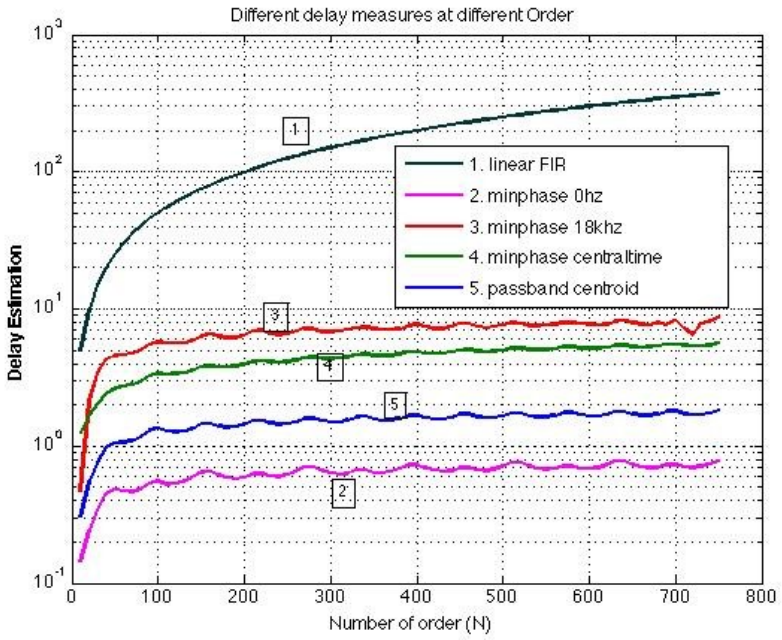


Figure 3.22: Different Delay measurements methods vs filter order N

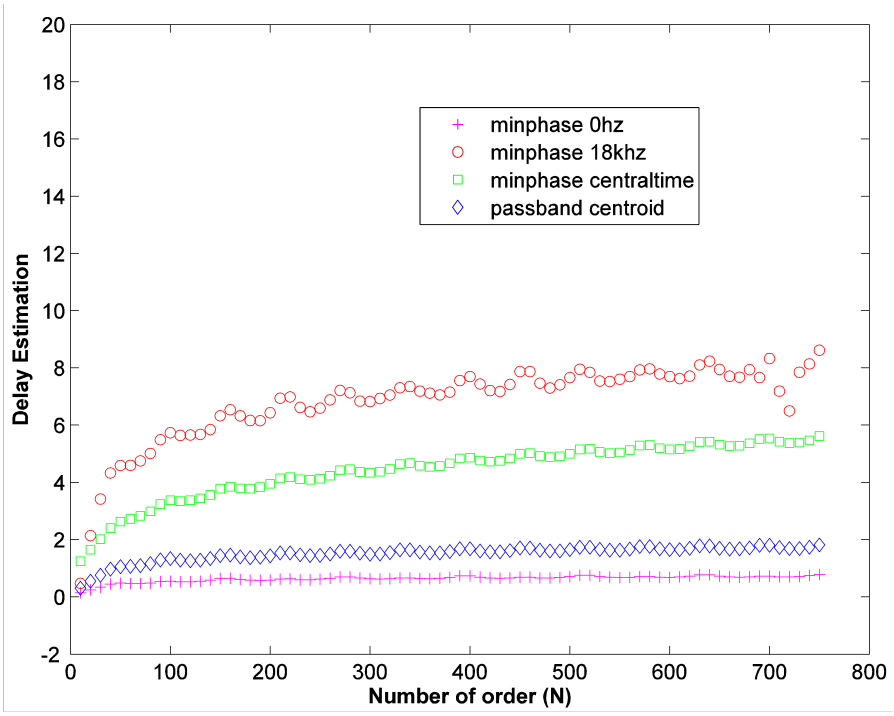


Figure 3.23: MP filter delay estimation

Figure 3.23 shows the closed look of linear scale of comparison of different quantitative delay measurements, “passband centroid” is measured at 0-18k Hz with cut off frequency 20k Hz. The slightly sudden drop of the 18k Hz curve is a numeric error caused by the minimum-phase filter design algorithm.

In Figure 3.23, the most interesting curve is the “passband centroid” curve, which is drawn by diamond symbol. We can see even the order is very high (up to 700 order), this delay measurement is almost kept same value as low order filter. We can most conclude that with proper selection of corner frequency, the MP filter delay is almost independent of orders.

3.4.5 Summary of MP Filter Evaluations

3.4.5.1 Can We Hear the MP Filter Group Delay Distortion?

Comparing with Table 2.3, if we design a MP filter around 100 order with cut off frequency greater than 21 kHz, the group delay distortion within the audio band only has 10 samples (200 us at 48k HZ sampling frequency) difference between 0Hz to 20k Hz, which should not cause the audible effects.

Also if we take delay centroid within the audio band as measurements in this particular design, the delay centroid value varies from 0.3 samples to 1.8 samples with the filter order range from 10 to 750. This can be almost regarded as constant, which we could use for estimation in multistage filter design.

3.4.5.2 Two Major Design Factors Affecting MP Group Delay Behaviour

In this section, we investigated the delay behaviour of minimum phase non-recursive filter in detail, especially for high-performance lowpass filters which are used in audio rate conversion as anti-aliasing and anti-image filters.

Basically, we evaluated the general properties of Equation: $G_i(\omega) = f_2(\omega_{si}, \omega_{pi}, \delta_{pi}, \delta_{si}, \omega)$ intuitively. We have the following important findings which would help develop the further approach to express the it in analytical form.

1. The four filter design parameters could be simplified into two main design factors which affect the group delay of MP FIR filter mainly. These are **a) the corner frequency**; **b) the order of the filter**. The passband ripple, stopband attenuation, and transition bandwidth can be translated into the filter order by order estimation technique.

2. The MP lowpass FIR filter's group delay increases monotonically with frequency. However, it varies little within the frequency band that is far away from the transition band. Therefore, the group delay distortion has a major effect at the high frequency components. Due to 1), this effect is mainly affected by filter order and value of the corner frequency.

In addition, various delay measurements methods were considered. The “central time” in the time domain and the “delay centroid” in the frequency domain are proposed for quantifying the group delay within the band of interest.

3.4.5.3 Formulation for Multistage Evaluation

These work provide the useful fundamental toolset for quantifying the group delay in multistage design when minimum phase technique is employed. We know the group delay of multistage filter is the summation of single stage: $G(\omega) = \sum_{i=1}^{k} G_i(\omega)$. The overall non-linearity of delay behaviour of a MP multistage system should also be affected by the above two design factors (corner frequency and filter order) of each stage.

In the multistage situation, the corner frequency of each stage can be the same value as the overall corner frequency specification. The filter order of each stage is affected by the transition band at that stage, which is the main design factor of the multistage filter in relation to the number of stages and the distribution of frequency alteration factors. Therefore, if we can qualify and quantify the group delay curve in relation to these two design factor, then we can have an analytical approach to the overall group delay curve of the multistage system.

3.5 Conclusion

In this chapter, we had an detailed investigation of the delay in $\Delta\Sigma$ ADC/DAC, which is mainly caused by the linear phase multistage filter that used for high-performance anti-aliasing or anti-image filtering purpose. We started the hardware ADC/DAC latency test. Then we reported comprehensive time-domain performance evaluations for different multistage filter systems in this application domain. A new objective formula of overall delay of the linear phase multistage filter is developed. Based on this formula, a new global balanced design algorithm is proposed to take account of both hardware cost and group delay. Finally, the objective time measurement methods and formulas of non-linear phase filter especially the minimum phase FIR filter are presented and evaluated.

Multistage filter design is a complex design problem. There are various design techniques such as halfband, and mixed types of filter can be used in a multistage system. The filter performance of each stage will have chain effects on the design requirements of other stages hence to the overall system. We believe the design can be further optimised for all different measurements using mathematical methods or computer algorithms. Our initial work of using annealing algorithm to further optimising the overall design already showed promising results.

In practice, the rule of thumb of manufacturing $\Delta\Sigma$ ADC/DAC for professional audio grade usually adopt two to three stages cascaded linear phase FIR design using halfband filter where is possible. The different sampling frequencies involved in these systems are the common ones for the digital audio domain, such as 44.1 kHz, 48 kHz, 96 kHz, and 192 kHz and their integer multiplications. Therefore there are limited design spaces D_i to select. The rule of thumb approach produces enough savings on the cost and with adequate performance, though it might not be the optimal balanced design.

Nevertheless, the formulas produced in this research Eq. (3.14) (3.15) (3.22) provided a mathematical accuracy to describe these proprieties on any design spaces and that how much the cost can be reduced and how much precisely the delay is worse off.

Our work filled the theoretical gap that the delay and cost trade-off can be determined and calculated before the implementation. In some particular situations where the subsample delay accuracy is needed, our formula could help to design the system with delay factor included. At least, when future applications are emerging, that might involve arbitrary sampling frequencies and oversampling ratios, our theory provides a ready knowledge base for such applications.

There is limited literature on perception effects of group delay distortion and what differences of minimum phase and linear phase on the perception of audio signal, especially when these filters work just above the audible band as anti-aliasing and anti-image purpose. It would be interesting to derive an appropriate listening test methodology to find out these effects for the different types of music.

Chapter 4

Delay of DAW on GPOS and New Time Deterministic OS Scheduling Framework

In this chapter, we present the work for understanding and reducing the delay in audio processing using DAWs and General Purpose Operating System (GPOS). It is organised as the follows.

- In Section 4.1, we present the updated latency measurements using the desktop operating systems to process audio signals, under heavy CPU audio processing loads, a large number of channels and possible cross-adaptive configurations [12].
- In Section 4.2, we present the new OS scheduling framework called "Time Deterministic Cyclic Scheduling" (TDCS) with simulation results. TDCS is specifically designed for low latency real-time multimedia processing that can be integrated into modern GPOS potentially.

Though the latency test work in Section 4.1 was conducted and published in 2010. A research group conducted similar measurements on the latest hardware, operating systems, and DAW software and the results were published in 2018 [167] with help from the author. The methodology in research [167] is same as author's work. The results of [167] indicate the relevance of the latency problem presented in this work is still up to

date. The new OS scheduling framework presented in the Section 4.2 is submitted for publication in 2017 and currently under the review.

4.1 Latency Test of Audio Processing Using Modern DAWs and GPOS

Using commodity computers in conjunction with live music digital audio workstations has become increasingly more popular in recent years. The latency of these DAW audio processing chains for some application such as live audio monitoring has always been perceived as a problem when DSP audio effects are needed. With "High Definition Audio" being standardised as the onboard soundcard hardware architecture for personal computers, and with advances in audio APIs, the low latency and multi-channel capability has made its way into home studios. In the following, we discuss the results of latency measurements of current popular operating systems and hosts applications with different audio APIs and audio processing loads.

4.1.1 Background

The latency of the DAWs has always been perceived as a problem for some real-time audio applications. The constraint of maximum allowed latency in audio processing varies between different applications. In audio streaming over a packet switched network, the one-way delay can be at the magnitude of seconds, and still be regarded as real-time [46]. In live performance and record monitoring environments, the maximum tolerable delay is around 10ms to 30ms depending on the different environments of performers and instruments [47][37]. For some performers, such as saxophone players, the threshold is even lower. Recent comprehensive testing results can be found at [36]. In the digital audio chain for live music, the DSP and software monitoring platform seems to be the main cause of latency [15].

Professional digital consoles normally have overall system latency no more than 2 ms. There are concerns that surround the use of computer-based DAWs for low latency work

(less than 10ms) due to unexpected buffer underrun noise[1] in sound when the CPU is heavy loaded [50]. Therefore, professional audio interface cards provide hardware-based monitor sub-mixing or bypass routing for the purpose of offloading the CPU.

In 1998, researchers presented the results and discussed the causes of audio process latency of common operating systems [52]. It was suggested that the ideal latency time could be 3ms, and revealed the difficulties involved in achieving this. In 2001, research [53] indicated that the proper architecture of audio API stacks should keep the latency in the audio processing path constant without being affected by heavy CPU load tasks. The most promising low latency audio layers at that time were Linux ALSA (Advanced Linux Sound Architecture) and Mac OS X CoreAudio. Some recent research in 2014 and 2016 [18, 19] found some fundamental sound architecture and communication protocols used in computers still have latency problem in the similar scale.

Audio driver architecture has evolved over the years, along with live audio applications and hardware platforms. The adaptive audio effects [54] which use feature extraction to create control signals for the processing of sound have often been proven to have high computational cost, leading to heavy CPU loads. However, with the appropriate side-chain design, multi-threading support from audio host platform and the concurrency of the software architecture, the hypothesis can be made that the intelligent subsystem and multiple audio processing paths should not affect the real time audio processing path even when the CPU load is coming from the audio application itself.

4.1.2 Testing Method

The sound source can be constructed mathematically in the form of either a single pulse or pulse train. When playing back the sound source, it is split into two channels. One channel is sent directly to recording devices, bypassing the operating system and the second is routed through the test system and recorded as a second channel using the same recording device. The recording device can be a digital recorder or a computer with professional sound interface. By analysing the final recording, the latency of audio

[1] It is worth noting that some online articles wrongly refer to the buffer underrun noise as jitter noise that usually appears in the frequency domain due to the uneven paces between audio samples.

processing path can be determined as shown in Figure 4.1.

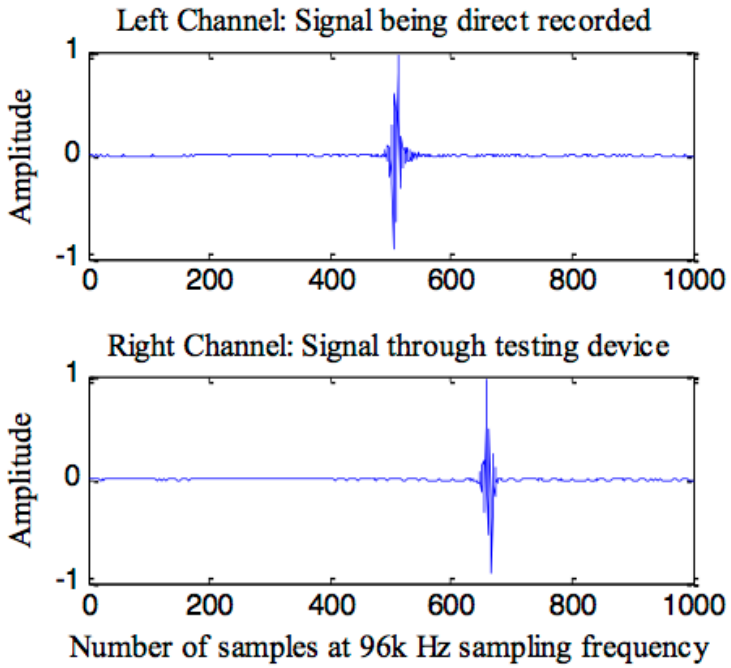


Figure 4.1: Latency measurement from recorded signals

The single pulse was used to access the minimum latency we could possibly achieve, whereas the pulse train was used for testing the glitches, variable latency and loss of information.

The capability of adjusting software buffers needs to be considered in order to make them comparable for different test cases.

4.1.2.1 Test Plan

Overall, there are many combinations of host DAWs, operating systems, driver APIs, soundcards and hardware. Therefore a carefully designed test plan is needed that contains a set of test cases in order to verify specific aspects of system latency by fixing and altering variables in the test domain.

In addition to this, the work presented in the study contains some special conditions for

the purpose of cross-reference. These included latencies of commonly used professional digital consoles and DSP development hardware.

The focus of our testing is the commodity personal computer. Therefore the results are taken mainly from the onboard soundcard of the machine. For the purpose of comparison however, external soundcards were also included in our testing procedures.

In general, four different testing groups were set:

- Test case 1 - Vanilla test, the purpose of this test is to obtain general latency results for different operating system and host combinations with exhaustive available software hosts. We tested common software such as Audacity, Logic Pro, Ableton Live, and Ardour.
- Test case 2 - Stress test, based on the results from the "Vanilla test", we selectively tested the latency of our chosen hosts with a heavily loaded CPU.
- Test case 3 - Adaptive effect test, to test if the audio processing latency is affected when the CPU load comes from the audio application itself, especially when the host handles multichannel audio and the adaptive audio effects are actively being used.
- Test case 4 - Cross-reference test. The purpose of this test is to get latency measurements from various systems other than common operating systems with onboard soundcards in order to avoid bias when evaluating the results of the above three tests. The tests include digital consoles, external soundcard, and the audio development hardware.

4.1.2.2 Variables

The variables of all the test cases were comprised with hardware, operating systems, and host applications. Ideally, the same hardware platform installed with multiple operating systems were used wherever possible. Different hardware platforms were tested as cross-reference for separating the hardware performance influences from that of the operating system.

4.1.2.3 Hardware Platforms

The Intel-based Apple computers were used as the main test platforms as they are able to support all three popular operating systems with additional configurations. A common PC laptop with similar hardware specification was also tested in order to verify the validity of test results taken from the Apple computers when the operating systems other than Mac OS X installed.

The main component of an on-board sound system is the hardware audio codec. Both ALC885 and CS4206A codec chips are in compliance with Intel HD audio, which support multiple inputs and outputs channels with sampling frequency up to 192k Hz [150] [168]. Table 4.1 lists the details of computer platforms, in which the CPUs are Intel Core 2 Duo with different CPU clock speed.

Table 4.1: Hardware platform of the test systems

Made	**CPU Speed (GHz)**	**Memory (GB)**	**Sound card Codec**
iMac	2.66	2	ALC885
Mac Book Pro	2.4	2	ALC885
Mac Book Pro	2.8	4	CS4206A

For cross-reference testing, the tested devices and hardware listed in the Table 4.2.

Table 4.2: Cross-reference testing devices

Type	**Made**
Digital Console	Yamaha 01v
Digital Console	Yamaha O2R 96
Digital Console	Yamaha DM2000
SHARC Board	ADSP-21161N EZ-KIT
USB soundcard	M-Box 2 Mini

Table 4.3: List of test operating systems

Operation System	Short Name
Apple Mac OS X 10.5.8	OSX (Leopard)
Apple Mac OS X 10.6.2	OSX (Snow Leopard)
Microsoft Windows XP	WinXP
Microsoft Windows 7	Win7
Ubuntu Linux 9.10	Linux

4.1.2.4 Operating Systems

Table 4.3 lists the operating systems tested. The Windows and Linux operating systems tested are all 32-bit versions. All operating systems updated with latest patches.

The Linux Operating system Ubuntu 9.10 has "Ubuntu Studio Audio Package" installed which contains the real-time preemption kernel patch for the 2.6.31 kernel.

One of the most important components within operating systems is the audio Application Programming Interfaces (APIs). They play the important roles in relation to audio processing latency to provide the middle layers between the low level sound system and the high level software applications. The default APIs of our tested operating systems are listed in the Table 4.4.

Table 4.4: List of audio APIs

API	Platform	Short name
Microsoft DirectSound & DirectSound Capture	Windows XP, Windows 7	DirectSound
Microsoft Multimedia Extensions	Windows XP, Windows 7	MME
Apple CoreAudio	Mac OS X	CoreAudio
Advanced Linux Sound Architecture	Linux	ALSA

There are additional APIs which are used by some audio applications but not included

by default by operating system, such as Steinberg Audio Stream Input Output (ASIO), PortAudio [169], and JACK API [170]. Each of them serves different application purpose.

4.1.2.5 Audio Application Hosts

In order to test the audio processing latency of operating systems, software is needed to capture the audio signal and playback it.

Rather than using a simple "play through" code, in most test cases, the completed DAW hosts were tested, because the goal of test case 3 is to test whether audio latency is affected by multichannel audio processing and intelligent audio effects. The host software provides the facilities to be able to carry out this test. Table 4.5 lists the hosts we used in the testing.

Table 4.5: List of test Audio Hosts

Hosts code	**Host name**	**Notes**
1	Apple Logic Pro 8.0	
2.a	Ableton Live 8.1.1	with Max/Msp
2.b	Ableton Live 8.0.1	
3.a	Audacity 1.2.5	Stable version
3.b	Audacity 1.3.11	Beta version
4	Ardour 2.8.7	Version 2.8.2
5	CAPlayThrough	Play through code

4.1.2.6 The Limitations of Test Plan

The latency measurements were tested based on the popular operating systems installed in common Apple computers in combination with onboard soundcards. The range of different computer hardware is limited, however, given that the commodity computer architecture is fairly standard, the computers we tested are common platform for DAWs. The results should be interesting in some aspects.

The second limitation is the limited number of external soundcards we tested. The results however should still be valid for showing the performance of operating systems performs with onboard soundcards and default APIs.

The third limitation is that there are very few audio application which supports all different operating systems. Perhaps the portability of audio applications and the optimisation of using native API and operating system features are the two conflicting efforts for software development. Therefore the cross-platform application such as Audacity uses the middle layer API “portaudio” to unify the audio programming interfaces for different operating system platforms.

The matrix of host applications and supported operating systems are listed in the Table 4.6.

Table 4.6: Matrix of Hosts and Operating systems

Host	**Windows**	**Linux**	**Mac OS X**
1	No	No	Yes
2.a	Yes	No	Yes
2.b	Yes	No	Yes
3.a	Yes	Yes	Yes
3.b	Yes	Yes	Yes
4	No	Yes	Yes

4.1.3 Test Results

4.1.3.1 Vanilla Test

In this test, we try to obtain the general picture of latency over our various audio hosts and operating systems with the built-in onboard sound systems and default settings. Table 4.7 shows the latencies measured using Audacity in different platform:

Table 4.7: Latency of Audacity host with sampling frequency 44100 Hz

Host	OS	APIs	Latency (ms)
3.a	MacOSX	CoreAudio	19
3.a	WindowsXP	MME	257
3.a	Windows7	MME	244
3.b	MacOSX	CoreAudio	30
3.b	WindowsXP	MME	398
		DirectSound	152
3.b	Windows7	MME	399
		DirectSound	201

We tested two versions of Audacity, the stable version 1.2.5, which uses "portaudio v18" and the beta version 1.3.11, which uses "portaudio v19". The "portaudio" library provides cross-platform audio API interfaces with encapsulation of platform dependent APIs such as CoreAudio, ALSA, DirectSound, or ASIO.

In Audacity 1.2.5, no buffer settings are available for end user, whereas in Audacity 1.3.11, the recording audio buffer is set to zero. No special drivers were installed for Windows platforms.

In Ubuntu Linux 9.10, the current software playback function of Audacity has some problems with newly adopted PulseAudio sound server system. It is considered that further testing using other Linux distribution is needed.

Vanilla test identified some low latency hosts for further test groups. The Table 4.8 shows the latency measurements of these hosts. The buffer settings are either the lowest that hosts applications support or the lowest at which monitored incoming sound can be recorded.

Table 4.8: Low latency hosts with sampling frequency 44100 Hz

Host	OS	API	Buffer[2]	Latency(ms)
1	OSX	CoreAudio	32*2	5
2.a	OSX	CoreAudio	14*2	4.2
2.b	WinXP	DirectX	512	73
2.b	Win7	DirectX	512	81
4	Linux	ALSA	64*2	3.3
4	OSX	CoreAudio	32*2	6.2

Table 4.9 shows the lowest possible latency in different operating system we can possibly get by using highest sampling frequency at 96k Hz supported by onboard soundcards.

Table 4.9: Lowest latencies from the Villain test

Host	OS	API	Buffer	Latency(ms)
2.b	WinXP	DirectX	512	73
4	Linux	ALSA	64*2	1.68
4	OSX	CoreAudio	32*2	3.54

In addition, the Vanilla Test found the latency measurements taken from Mac OS X 10.5.8 Leopard are almost identical to Mac OS X 10.6.2 Snow Leopard. And there are similar results for Windows system whether installed on an Apple computer or on a PC laptop with similar hardware specification.

4.1.3.2 Stress Test

The audio processing latency caused by CPU stress is tested rather than by the I/O stress. With advanced DSP techniques being widely used in real-time audio processing, the computational cost is more likely to be CPU stressed tasks.

The sound source, consisting of a series of pulses at constant intervals is used as test signal. It is observed that even without CPU stress, when the latency is less than 5 ms, the audio signal suffers from distortion and glitches and loss of information.

The following table shows the latency with and without CPU load. System monitoring software is used to ensure the CPU load outside audio application is 100%. In addition, the host software normally has a built-in CPU meter to indicate the CPU load of audio processing only [171]. This means that even when the system monitor indicates 100% CPU load, the audio application CPU load may still be very low. It is observed, however, that if the audio processing CPU load increases, it is reflected on the outside system monitor meter.

Table 4.10: Audio latency with different CPU loads

Host	**OS (API)**	**Without Load**	**With Outside load**	**With Audio load**
2.b	WinXP (DirectX)	73ms (buffer 512)	81ms (buffer 512)	104ms (buffer 512)
2.a	OS X (CoreAudio)	4ms (buffer 14*2)	4 ms (buffer 14*2)	5.80ms (buffer 14*2)
4	Linux (ALSA)	3.31ms	3.31ms	error
		22ms (buffer 512*2)	22ms (buffer 512*2)	22ms (buffer 512*2)

The test results indicate that the CPU load generated outside of the audio applications have very little effect on the latency of the audio processing chain. For Mac OS X and Linux systems, this effect cannot be observed, for Windows system, this is very small.

To some extent, when the CPU load comes from the inside of the audio application, latency is increased by 1-2ms for Mac OS X.

In the Linux system, it causes a system error when Ardour tries to connect to the JACK audio server. With an increased audio buffer setting to 512 samples in the Linux system, the inside CPU load doesn't seems affect the latency.

However, with high audio processing load, test signal being the pulse train, it was observed that the signal suffers distortions or loss of pulses as shown in the Figure 4.2.

The Figure 4.2 show in Linux, the signal channel processed by operating system lost some information when the buffer setting and sampling frequency were set in order to

obtain very low latency. The similar effects were observed in Windows and Mac OS X systems.

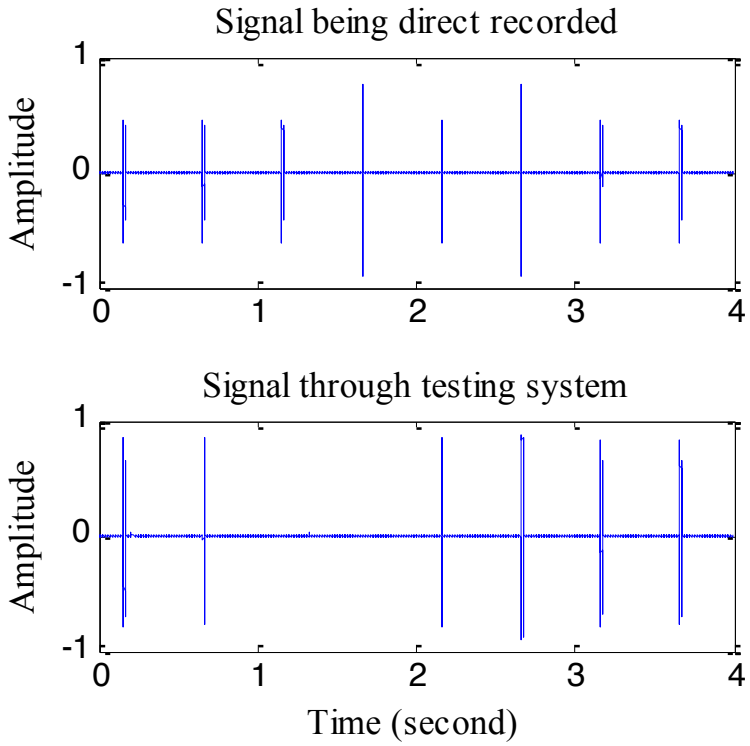


Figure 4.2: Loss of pulses in audio processing path at low latency setting

4.1.3.3 Multichannel Latency

The Table 4.11 shows the latency variation caused by the large number of channels e.g. over 50 channels. This is the same effects to that of increased the internal CPU load.

Table 4.11: Multichannel latency effects

Host	OS	Latency Single Channel	Latency Multi Channel
2.b	WinXP	73 ms	104 ms
2.a	OS X	4 ms	5.80 ms
4	Linux	3.31 ms	error
		22 ms	22 ms

4.1.3.4 Adaptive Effect Latency

Adaptive audio effects combine audio feature extraction and audio processing in order to give musicians and audio engineers another creative dimension. This helps in generating new musical concepts and contributes to making existing tasks and processes more intelligent. In the real-time multichannel mode, it may require the audio analysis subsystem to synchronise with the audio processing chain in order to make an audio effect decision, which can be computational cost if the large number of channels are involved and the required analysis rate is high.

The measurement in Table 4.11, however, did not include the adaptive audio effects. The current audio application hosts have not widely supported this type of audio effects yet. The “Max for Live” functionality of Ableton combines Max/Msp with Ableton Live plug-in structure, providing an interesting starting point. Adaptive audio effects are created fairly easily using “Max for Live”. It is of interest to test the latency in this configuration.

In order to obtain an undistorted audio signal, the buffer is set to 256 samples. The effect plug-in is based on feature extraction created to obtain “loudness”, “brightness”, “noisiness”, and “onsets” of audio signal. Based on these features, the amplitude of the signal is modulated with some random parameters. This patch is then applied to multiple channels in order to increase the internal CPU load of the host. The test shows the latency performance has a vast difference when Max/Msp edit window is opened and closed.

Table 4.12: Latency measurement of “Max for Live”

Host	OS	Max Window	Single channel (ms)	MultiChannel (ms)
2.a	OS X	Opened	97-99	100-103
2.a	OS X	Closed	32 -51	39-67

4.1.3.5 Cross-Reference Test

Table 4.13 shows the latency measurement of dedicated hardware audio devices.

Table 4.13: Latency of dedicated digital audio hardware

Type	Latency
Yamaha 01v	2.42 ms
Yamaha O2R 96	2.04 ms
Yamaha DM2000	1.99 ms
ADSP-21161N EZ-KIT (SHARC)	1.60 ms

Table 4.14 shows the latency measurement using an external soundcard and dedicated ASIO soundcard driver for Windows. Under this circumstance, the Windows platform performs at a comparable level to Mac OS X with the same buffer setting. Though, the Mac OS X supports lower buffer settings up to 6.8ms.

Table 4.14: Latency measurement of external soundcard M-box 2 mini

Host	Platform	API	Buffer setting	Latency (ms)
2.a	MacOSX	CoreAudio	128*2	11.9
2.b	WinXP	ASIO	128*2	12

The Mac OS X CoreAudio driver has also been patched by the manufacturer to support this particular soundcard.

4.1.4 Discussion

4.1.4.1 Overall Latency Pictures

Beginning with the cross-platform host Audacity, the Vanilla Test obtained the general latency picture of operating systems with onboard soundcards.

It shows that the latency of record enabled monitoring of the beta version of Audacity is actually worse than the old stable version. This might link to the regression report of using newer "portaudio v19" library (according to Audacity development website). In addition, the Audacity software used in this testing are pre-built binaries. Giving its open source nature, to test again with compiling "portaudio" and Audacity from the source code to take advantage of the native audio API could be further investigated

With the onboard Intel HD audio sound system, the Linux and Mac OS X operating system have low latency performance, and windows DirectSound API performs better than its legacy MME sound API.

With native supported sound driver APIs, the audio hosts dedicated for live application could have low latency within 8-10 ms monitoring requirements.

In 2001, [53], the lowest measured latency was 2.72 ms. It was measured from Linux system with the ALSA audio API that replaced the default OSS at that time.

ALSA has already become the default Linux audio driver. Our test results show that the lowest latency is provided by open source DAW project Ardour in Linux with ALSA sound driver and JACK audio connection. When using a sampling frequency at 96k Hz, the measurable latency can be as low as 1.68 ms (see Table 4.9). This is comparable with the Yamaha digital consoles tested (see Table 4.13).

Another observation is that the reported latencies of most software hosts do not match the measured values. The only exception is Ardour with Linux systems.

Figure 4.3 shows the plotting of latency measurements and the latency reported by the hosts according to buffer settings. The measured value and software reported values are consistent for Ardour in Linux. It needs further research to confirm if the real time Linux kernel helped the audio host to maintains accurate timing information and scheduling.

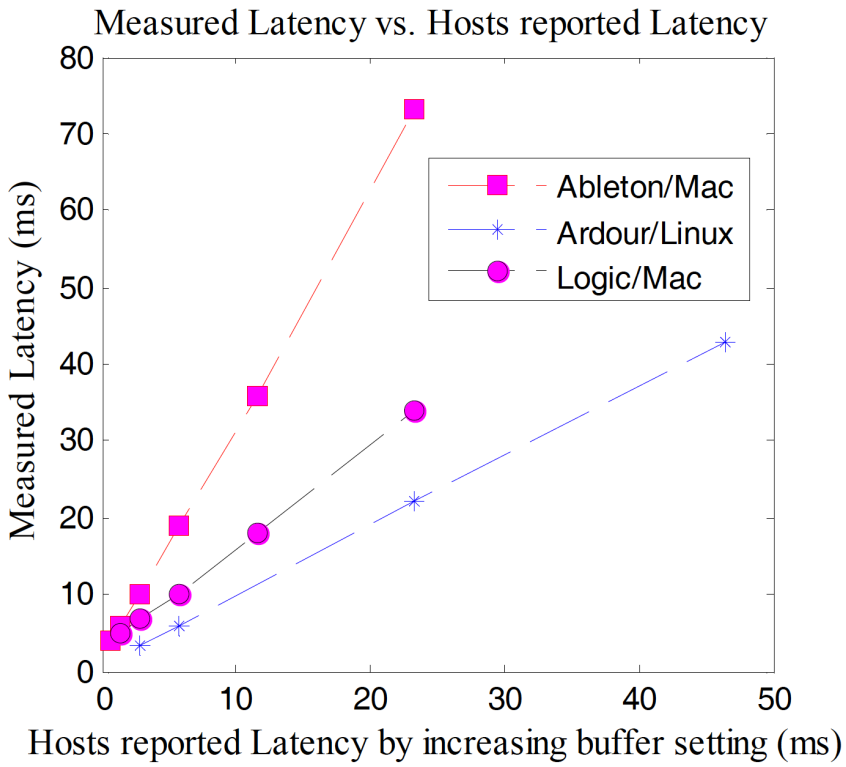


Figure 4.3: Measured Latency vs. Hosts reported Latency in millisecond with different buffer setting

It is noted that according to the results of cross-reference test, the measured latencies did match the reported latencies for Mac OS X system if an external soundcard is used.

With the external soundcard and driver being used, the Windows system could have comparable low latency as Mac OS X when using the same buffer setting.

4.1.4.2 Latency under load

In [53], it was shown that the CPU load outside the audio application had little effect on the latency of audio processing.

In our research, the effects on audio processing latency by CPU load caused by audio application itself are evaluated. Consistent with [53], the CPU load outside audio ap-

plication has an unnoticeable effect on the latency for Mac OS X and Linux System, whereas the CPU load inside audio application has caused some small increases of latency generally for all operating systems.

However, the research in [53] did not mention the quality of audio signal when the low latency is required the CPU is stressed. Our research indicates that in low latency mode, especially with CPU is stressed by internal audio processing load, the signal suffers losses and distortion.

It is worth noting that the hardware architecture of the SHARC board is fairly similar to that Intel HD-Audio architecture [172]. However, it has very low latency (see Table 4.13) with good signal quality. The measurement is taken by running an embedded "talk through" example code. This embedded software is driven by the hardware level interrupt with enabled DMA features. The software architecture of this is quite different with computer-based sound system.

4.1.4.3 Multichannel and Adaptive Audio Effects

Increasing the number of channels alone does not lead to increased audio processing latency. Only when the channel number is increased considerably to around 50 audio channels, it does affect the latency in the same way as increasing the internal CPU load from the audio application host (see Table 4.11).

The adaptive audio effects provide new creative dimension and intelligent workflow. Use advanced feature extraction based audio processing in real-time is proven interesting and challenging. However the current audio application hosts have not been able to support it widely and flexibly, with the exception of side chain based plug-ins etc. Therefore the test is limited by the available host and the way the host operates.

The "Max for Live" product supports this flexibility by incorporating a Max patch as plug-in. However the results show that the variations of latency do not strongly correlate with audio processing load. The variations of latency might be caused by the configuration and software structure themselves.

4.1.4.4 Summary

We demonstrated and discussed the test results of the real-time audio processing latency of current popular operating systems with onboard soundcards. Though the results presented in this chapter were published in 2010, there are researches carried out the similar test in 2018 using similar methodologies on the latest DAW based systems including web audio [167, 173]. The similar problems persist.

In addition to testing the effects on audio processing latency by CPU load outside audio applications, this research method measured whether latency is affected by the load coming from the audio processing application itself, especially with a large number of concurrent audio processing channels.

The general latency pictures of common operating systems were obtained. Though the lowest latency of an operating system with onboard soundcard can be close to the professional digital audio hardware, it may suffer losses of audio signals when the CPU is fully loaded with audio processing tasks due to the buffer underrun. The latency of adaptive audio effects processing has also been evaluated. Due to the constraint of the software structure, the value of the testing results is limited.

There are many factors affecting the latency of using operating systems to process audio in real-time. For example, how audio software is programmed to avoid blocking in the audio callback and to avoid complex algorithms with unbounded execution time.

There are suggestions that GPOS is lack of real-time scheduling support, so they underperform RTOS. However, most modern GPOS support pre-emptive scheduling schemes. We suspect the priority based pre-emptive scheduling can provide good performance but might still fail to meet deadline occasionally when CPU is under full utilisation. In the next section, we propose the new scheduling algorithm that aims to provide deterministic behaviour.

4.2 TDCS: A New Scheduling Framework for Real-Time Multimedia

4.2.1 How TDCS Relates GPOS and RTOS

General Purpose Operating System (GPOS) can satisfy versatile processing requirements for intelligent audio production[55], that needs configurable feature extraction, machine learning and digital signal processing (DSP) tasks. It can process multiple channels with different source sampling rates with flexible routing capability as shown in Figure 1.3. However, it often results in occasional bursting CPU utilisation that is close to full, which may cause the unpredictable buffer underrun of real-time audio processing [107] for audio signals [12].

The conventional approach to overcome the buffer underrun problem is to increase the buffer size, but it results in increased latency. Another approach is to design the system so that the processing time of audio tasks does not exceed the planned deadline hence reduce the uncertainty of buffer underrun. The timing performance of scheduling the audio processing tasks is the key to reduce excessive buffer size.

The scheduling of traditional GPOS is optimised toward tasks throughput. It often adopts fair scheduling that each task is sharing equal amount processing time. An RTOS is used for time-critical systems. RTOS is supposed to give a predictable response. The task scheduling features of an RTOS usually include preemption and priority based scheduling algorithm such and RMS or EDF.

Modern desktop operating systems though belong to the category of GPOS. They do also have some important RTOS features. Windows OS has six different priority classes. Mac OS has four different priority bands. Linux has preemptive kernels and can be configured to use priority based RTOS scheduling algorithms such as EDF.

Therefore properly configured modern desktop OSes with RTOS features and optimisation can reduce the audio processing latency. However, we suspect this approach can achieve the performance most of the time, although rarely but it still can fail occasionally [107].

This is because that when the high priority tasks occupy the full CPU cycles (up to 100% CPU utilisation). Even the classic RTOS scheduling algorithm will fail to achieve the deadline at all time. For example, the theoretical up-limit of the CPU utilisation of the RMS scheduling algorithm is about 69.3% (Eq. (4.2)). That might be the case when we tested the system with full CPU load form audio processing itself and the system suffered underrun problem.

In this work, we proposed a new set of OS scheduling framework that is called Time Deterministic Cyclic Scheduling (TDCS) that is specifically tailored for the real-time multimedia system with trading off mechanism of latency and predictable QoE requirements that aims to achieve the predictability of specific tasks using systematic design approach. The design philosophy of TDCS is to have a two-tier system. The foreground tasks are scheduled in a cyclic and cooperative way so that they will have time deterministic behaviour. Whereas the background tasks can be scheduled in classic priority-based approach but always only can fill the processing gaps of the foreground tasks. The performance evaluation based on the simulation results between TDCS and RMS is given in the following sections to show the pros and cons of it.

4.2.2 The Characteristics of Modern Real-Time Multimedia System

Traditionally, there are two categories of real-time systems: hard real-time system and soft real-time system. In hard real-time system, the missing deadline of tasks are regarded as a failure, whereas the soft real-time system can have some level of tolerances of missing deadlines. Some textbooks regard multimedia systems as hard real-time systems, due to the strict timing requirements or human perception of glitch caused by missing task deadline.

We think the live multimedia system is neither hard real-time nor soft real-time. It lays in between. In a professional audio environment, we would not want to miss any audio/video frames that cause the negative perceptual effects. The jitter of processing single audio or video frame can be tolerated within acceptable user perceptions, using de-jitter buffer to compensate it at the cost of delay. However, the behaviour of jitter should be predictable. In summary, the characteristics of such system are below:

- It mainly deals with multiple periodic tasks at different update rates with pseudo-isochronous tasks pattern.
- It is acceptable to miss the deadline for some tasks, but will affect the quality of experiences. Ideally those tasks that missed deadlines shall not be discarded but still to be scheduled at later points.
- It shall provide the trade-off between delay and jitter, using buffer to mitigate the jitter of the samples. There shall not be any unexpected jitter buffer underrun when the CPU utilisation is close to full.

Traditional cyclic scheduling based approach is difficult to grow and maintain when the number of tasks increases and the periods of tasks are not harmonically related. However, the properties of multimedia system indicates there are compromises can be made that is to adjust the tasks' deadlines within the perceptual tolerance to simplify the system realisation and increase the efficiency of the scheduler.

4.2.3 Proposed Scheduling Scheme

Therefore, Time Deterministic Cyclic Scheduling (TDCS) provides trade-off between overall input output delay with other system measures, that is based on a traditional cyclic execution based scheduling. In addition, the TDCS is the main part of a hierarchical scheduling scheme for mixed criticality system.

4.2.4 Model of Rate Monotonic Tasks

In real-time system, a task consists with a sequence of jobs. For rate monotonic tasks, we have $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$, where τ is a task set that contains n different tasks. Each task can be defined as $\tau_i = \{C_i, T_i, D_i, P_i\}$, where

- C_i is worst-case execution time;
- T_i is the period of τ_i;
- D_i is the deadline of τ_i;

- P_i is the priority of τ_i;

For each task the CPU utilisation is $U_i = C_i/T_i$, so the total utilisation is

$$U = \sum_{n=1}^{\infty} U_i \tag{4.1}$$

Normally we have $D_i = T_i$. The rate monotonic scheduling (RMS) algorithm assigns the task priority according to the task period. Task with shorter period has higher priority. Liu and Layland [124] proved that the sufficient condition of scheduling of RMS is

$$U \leq n(2^{(1/n)} - 1) \rightarrow 0.6931 \tag{4.2}$$

where n is the number of tasks. This condition is sufficient but not necessary based on the tasks are preemptible. Lehoczky 1989 [174] shows the sufficient and necessary condition of schedulability of RMS with less up bond CPU utilisation but more complex schedulability test formula. According to [174], for task set τ, we can define:

$$W_i(t) = \sum_{j=1}^{i} C_j \lceil t/T_i \rceil \tag{4.3}$$

Eq. (4.3) represents the accumulated CPU utilisation between time interval $[0, t]$. We can also defines

$$L_i(t) = W_i(t)/t \tag{4.4}$$

$$L_i = min_{\{0<t\leq T_i\}} L_i(t) \tag{4.5}$$

$$L = max_{\{0\leq i\leq n\}} L_i \tag{4.6}$$

- When the i^{th} task where $(1 \leq i \leq n)$, τ_i is schedulable, we need $L_i \leq 1$;
- For the whole task set τ is schedulable, we need to have $L \leq 1$.

4.2.5 TDCS Algorithm

4.2.5.1 Temporal Segmentation

The foundation of timing correctness this scheduling is based on temporal segmentation which is similar to ARINC 653 standard. The OS is partitioned as independent hyper-periodic segments or Major Cycle in time domain driven by accurate low level system timer. We define this cycle as T_c. The chosen of T_c is decided by applications' context.

The selection of length of segmentation is similar to the problem of selection of hyper-period or size of "Super frame". However, we have a flexible architecture of using the segmentation size that practically match the requirements of connected system such as USB or networking interfaces.

In theory the period of temporal segmentation can be the Least Common Multiple (LCM) of different task periods. However, this can be very large and impractical to implement. In our design, The period tasks will go through two buffer system an input "mapping buffer" and an output "de-jitter buffer". The former buffer is used to convert to an arbitrary hyper-period that you want. The second buffer is to render the tasks as their own source rate.

4.2.5.2 Hyper-Period Conversion Algorithms

The selection of Hyper-period in TDCS can be flexible. It is not necessary of the LCM of periods of all tasks. As mentioned above, it could depend on the application context that driven by master clock or the design criteria that needs to protect the criticality within certain time period.

We define LCM of the periods of all tasks as 'Major Cycle': T_L, and 'Minor Cycle' be T_c. We can have different way to decide the value of T_c.

For example T_c can be the longest period of tasks to be scheduled when the CPU utilisation is under the upper bond of RMS schedulability conditions. In case of the very high CPU utilisation that exceeds RMS schedulability but under 100%, we proposed an "expanded hyper-period conversion algorithm" to calculate T_c

Case 1 General hyper-period conversion algorithm We propose a hyper-period conversion algorithm to convert different T_i to new T_i'. For example the following algorithm convert longest T_i as minor cycle T_c that is normally much shorter than LCM based hyper-period:

We define the following formula:

$$T_c = max\{T_i\} \tag{4.7}$$

$$T_L = LCM\{T_i\} \tag{4.8}$$

$$K_i = \lceil \frac{T_c}{T_i} \rceil \tag{4.9}$$

we have $f_c = min\{f_i\}$. Increase f_i by Δf_i to be f_i' so that $f_i' = K_i f_c$; where K_i is integer. We have the new hyper-period T_c and for each task τ_i, we can schedule K_i of them in one hyper-period.

For instance, that we have 3 tasks with period {10,20,35}, the T_L is 140. $f_i = \{14, 7, 4\}$, we then can find $f_i' = \{16, 8, 4\}$; and $K_i = \{4, 2, 1\}$. so in this case, we can use $Task3$ period as hyper-period. With task queue, we schedule $Task1$ four times, $Task2$ twice and $Task3$ once within hyper-period. We introduce a scheduling jitter Δf_i that can be mitigated by de-jitter task queue. This algorithm creates empty scheduling slots f_i^{empty}, which can be calculated below:

$$f_i^{empty} = f_i' - f_i = K_i \times \frac{T_L}{T_c} - \frac{T_L}{T_i} \tag{4.10}$$

Let $M = \frac{T_L}{T_c}$, we have M number of Minor cycles in one Major Cycle.

Case 2 Expanded hyper-period conversion algorithm For the CPU utilisation exceed the upper bond of RMS schedulability, we can alter the algorithm and provide new hyper-period and Task mapping *'As tight as possible'*, for utilisation that is close to 100%.

Let total empty slots in one T_c be TS, we have

$$TS = \sum_{i=1}^{N} \Delta f_i \times C_i = \sum_{i=1}^{N} f_i^{empty} \times C_i \tag{4.11}$$

$$T_{ce} = T_c + \frac{TS}{M} \tag{4.12}$$

In this case, the new Minor Cycle T_{ce} shall guarantee the all the tasks to be scheduled even the total CPU capacity is 100%. However, the Major Cycle as the production of $T_{ce}$and Mexceeds the value of LCM. We can further have an adjustment algorithm to make an uneven minor cycle to fit all task within one major cycle LCM:

Let $j$be the index of minor cycle T_{ce} within Major Cycle, so we have

$$T_L = \sum_{j=1}^{M} T_c \tag{4.13}$$

Therefore we can have the uneven T'_{ce} can be calculated as below. Let

$$j_{expend} = min\{\lfloor \frac{f_i}{K_i} \rfloor\} \tag{4.14}$$

$$T'_{ce}(j) = \begin{cases} T_{ce}, & \text{when} \quad j \leq j_{expand} \\ T_{ce} - \sum_{i=1}^{|\tau|} L_j(i), & \text{when} \quad j > j_{expand} \end{cases} \tag{4.15}$$

where

$$L_j(i) = \begin{cases} (j \times K_i - f_i) \times C_i, & \text{when} \quad j \times K_i > f_i > (j-1) \times K_i \\ K_i \times C_i, & \text{when} \quad (j \times K_i - f_i) > K_i \end{cases} \tag{4.16}$$

where $1 < j < M$; and $1 < i < |\tau|$.

4.2.5.3 Periodic Tasks Mapping Schemes

For most period task τ_i that happens every T_i, the tasks are scheduled off-line by allocating the τ_i in cycle T_c. This is done by the task allocation mapping algorithms below:

1. Calculate the number of tasks N of τ_i in every minor cycle by $N \geq T_c/T_i$. For example, if T_c is 20ms, T_i is 5ms, then N=4;
2. Allocate K timing positions within every T_c for τ_i, ensure the same positions for every T_L for any τ_i. There are different methods to allocate the positions such as 'even spread', or 'as tight as possible'. Each of them has different performance effects.
3. Prepare the multiple low level timers for dispatching the mapped task τ_i. and prepare the de-jitter input/output task queue for τ_i.

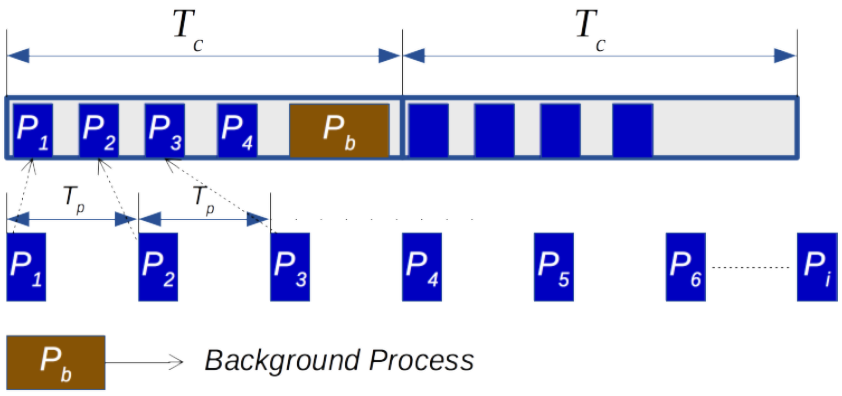


Figure 4.4: Periodic Tasks Mapping

One of the advantages of TDCS is the predictability and CPU utilisation. We implement the TDCS scheduling algorithm with "As tight as possible' tasks mapping scheme and carried out the test. The various results are compared with classic RMS and Non-Preemptive Rate-Monotonic Scheduling (NP-RMS). Next, we show the performance of TDCS in comparison with RMS and NP-RMS.

4.2.6 Compare TDCS with Non-Preemptive RMS and RMS

It is worth to compare TDCS with classic fixed priority scheduling algorithms. For multimedia applications, the absolute deadline is not essential. The Non-Preemptive RMS (NP-RMS) [175, 176] is the RMS scheduling algorithm without preemption, which

reduces the system complexity. NP-RMS is suitable for the multimedia applications where hard RT is not essential. We compare TDCS with both RMS and NP-RMS based scheduling policies. We implemented TDCS scheduling in TORSCHE [177].

4.2.6.1 Schedulability Simulation

We define three task sets that represent the load of CPU from sparse to dense. The tasks sets used for simulation are described in Table 4.15. The Set 1 is designed so that all three schedulers can successfully schedule all the tasks. The Set 2 is designed to make NP-RMS fail to scheudle whereas RMS and TDCS can. The Set 3 is designed to simulate the CPU is heavily loaded so that both RMS and NP-RMS will miss some deadlines.

Table 4.15: Simulation Tasks Sets and CPU utilisation

Task Set	**Details**	**Utilisation**
Set 1	task1 = {2, 10, 10} task2 = {4, 20, 20} task3 = {10, 35, 35}	68.57%
Set 2	task1 = {2, 10, 10} task2 = {4, 20, 20} task3 = {15, 35, 35}	82.86%
Set 3	task1 = {2, 10, 10} task2 = {4, 10, 10} task3 = {20, 10, 10}	97.14%

4.2.6.2 Simulation Results

Figure 4.5 shows the simulation results of Set 1 which has average CPU utilisation of 68.57%. The top sub figure shows the three original tasks defined in Set 1. Task1 has highest frequency hence will be assigned to the highest priority in RMS scheduling policy. The P_b process is the background Non-RT tasks that can be fit into the gap of RT tasks. The second sub-figure of Figure 4.5 is the task map of RMS, which shows some of task2 and task3 are delayed or interrupted but can finish within the deadline D_i. The

third sub figure of Figure 4.5 shows the task map of NP-RMS scheduling. For NP-RMS, none of the task can be interrupted, so even highest priority tasks in task1 have shown some delays. The bottom sub figure in Figure 4.5 shows the task map of TDCS. We use 'as tight as possible' tasks mapping scheme to maximise the capacity of the system. It shows the clear pattern within minor cycle and major cycle.

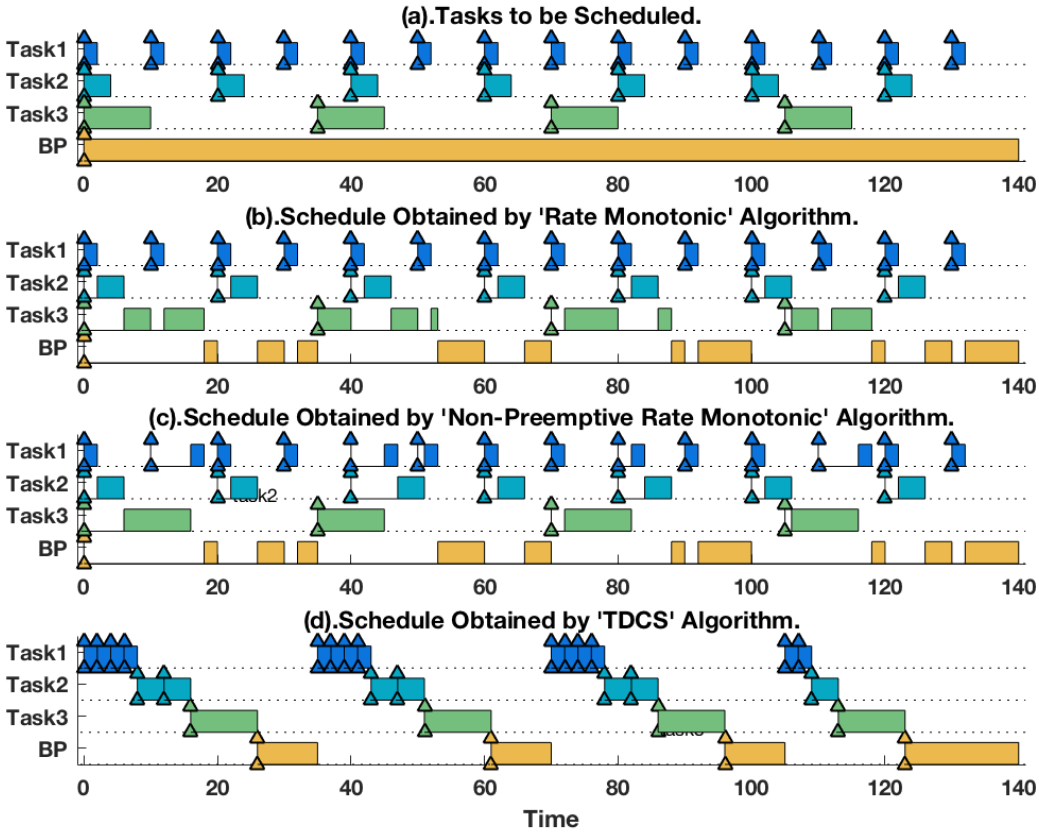


Figure 4.5: Simulation Results for Set 1

Figure 4.6 shows the results of Set 2 where the average CPU utilisation is 0.8286: In this case, the non-preemptive scheduling algorithm cannot schedule all the tasks. Some tasks in Task1 get lost such as the 2nd 5th, 9th and 13th tasks in Task1. Figure 4.7 shows the results of Set 3 which has average CPU utilisation of 97.14%. In this case, both non-preemptive RMS and RMS scheduling algorithms cannot schedule all the tasks. But with expanded hyper-period conversion algorithm and 'as tight as possible'. The TDCS

can successfully schedule all the tasks.

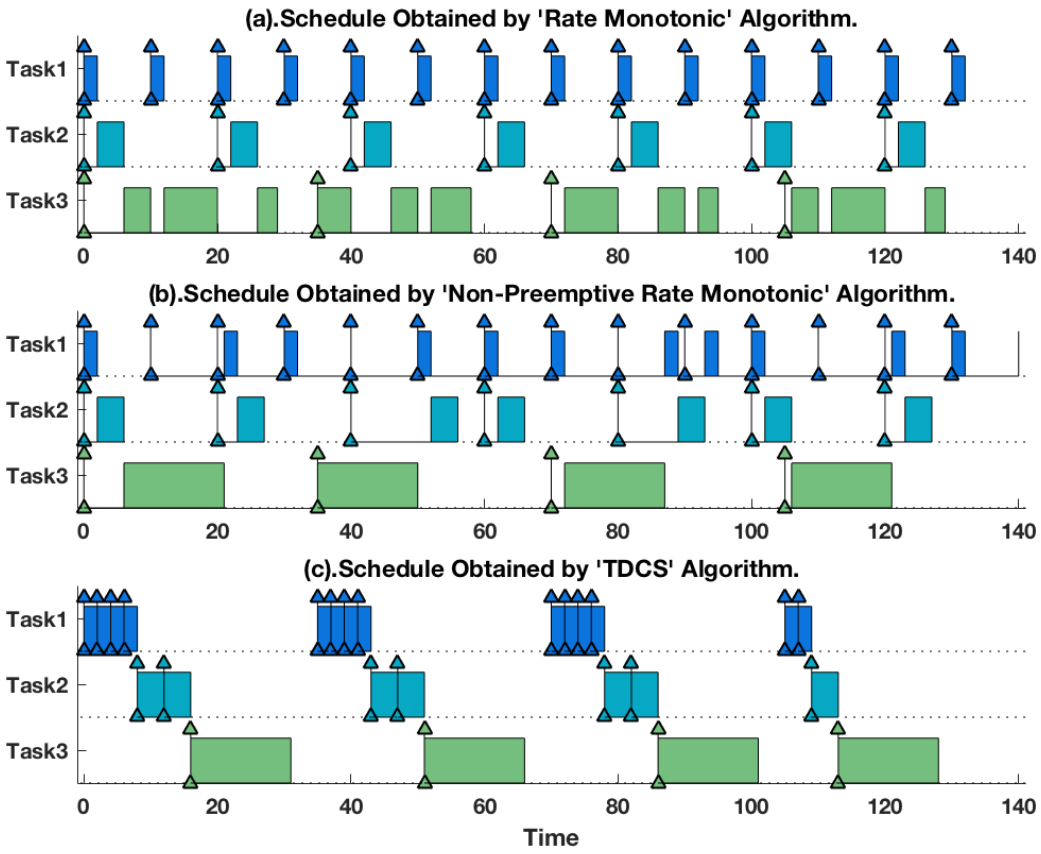


Figure 4.6: Simulation Results Set 2

4.2.6.3 Jitter of Individual Delay Simulation

Figure 4.8 shows the individual task starting time (a) and the delay between the time of the tasks actually starting execution and the time when the tasks are released (b). Figure 4.8 is based on Set 1 and compares between TDCS with NP-RMS. Figure 4.9 also based on Set 1 but shows the results between TDCS and RMS. The negative value of TDCS delay in the figures is because we buffered TDCS tasks then condense the 'future' tasks together as often does in frame based audio processing. It shall be added with offset of buffering delay.

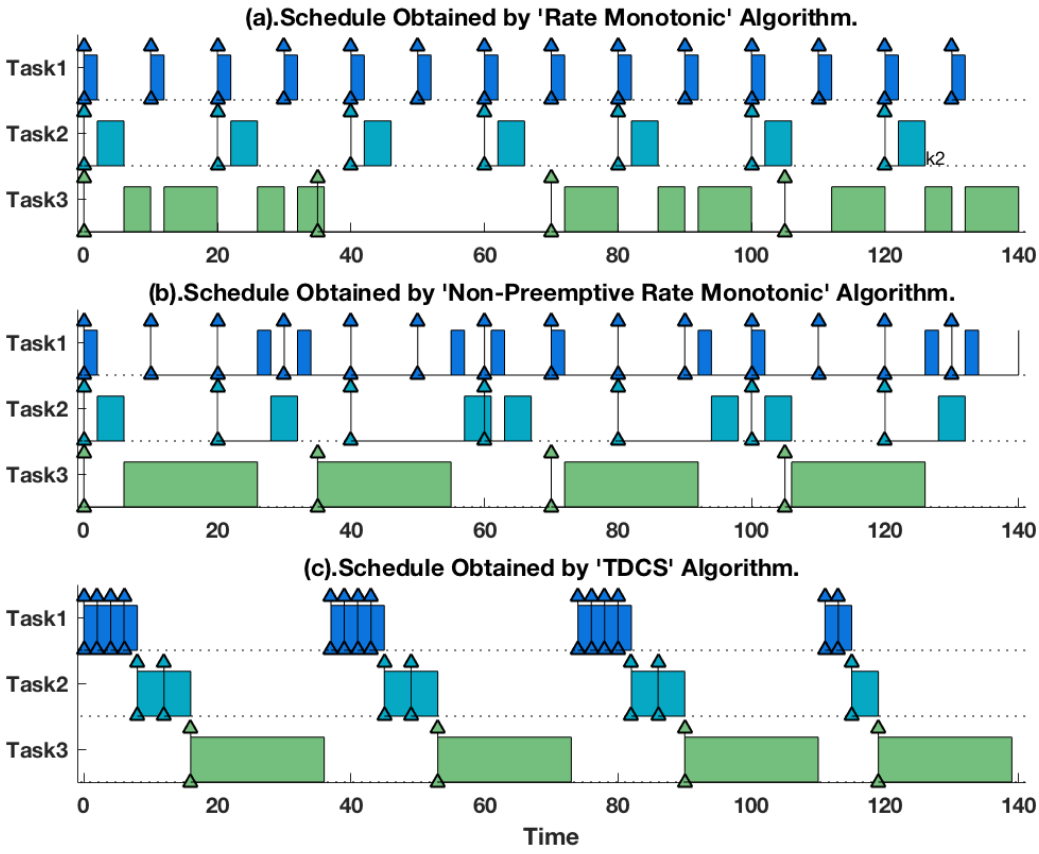


Figure 4.7: Simulation Results Set 3

RMS and NPRMS performs very well if the tasks loading is not very high. The simulation shows the "As tight as possible" tasks mapping of TDCS that is geared towards maximise CPU utilisation. Therefore TDCS shows some fluctuation of tasks delay in comparison with tasks release time. With de-jitter buffer, this effect will be alleviated. The future work will look into how to trade off between the jitter and delay performance and overall tasks utilisation.

4.2.6.4 Overall Task Throughput Simulation

In this simulation,we created random tasks sets for all three cases TDCS, RMS and NP-RMS. The random tasks sets have three RT tasks make up the CPU utilisation range from

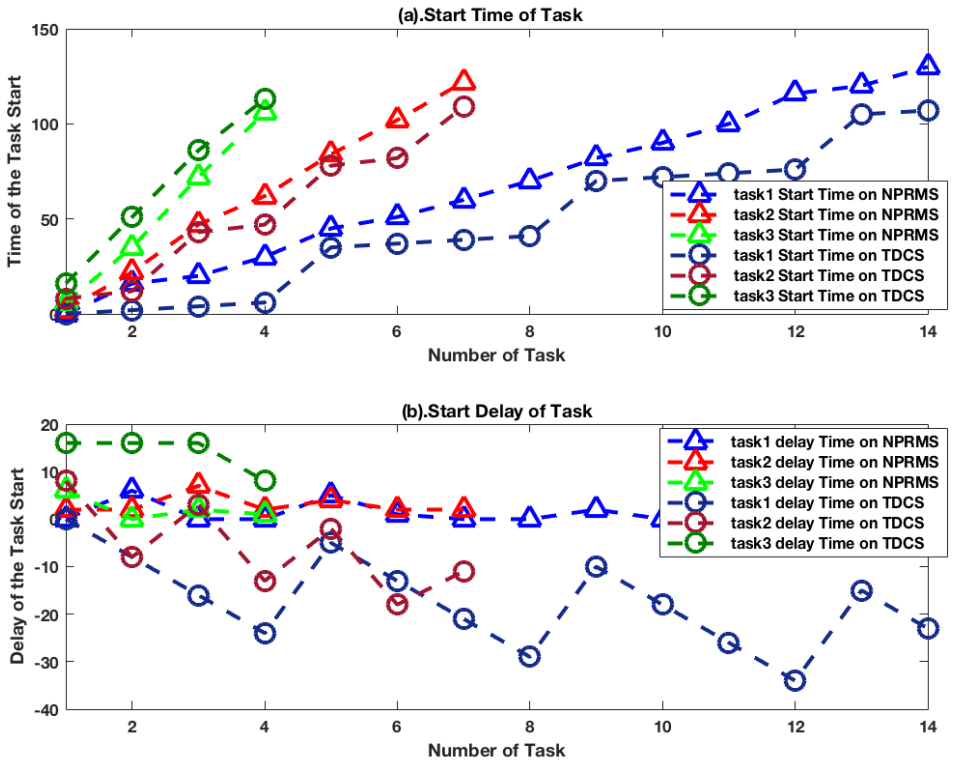


Figure 4.8: Task delay and starting time TDCS vs NP-RMS

60%, 65%, ..., until 100%. For each case, we created 100 random sets. The percentages of RT tasks that can be executed successfully are plotted against CPU utilisation for all three different scheduling algorithms. The result is shown in Figure 4.10. It clearly shows that our proposed TDCS scheduling and "Expanded hyper-period conversion algorithm" can handle the task load up to 100%.

4.2.7 Using TDCS in Mixed-Criticality System

In this section, we briefly discuss how to use TDCS in mixed criticality system or how to incorporate TDCS with GPOS as RT extension.

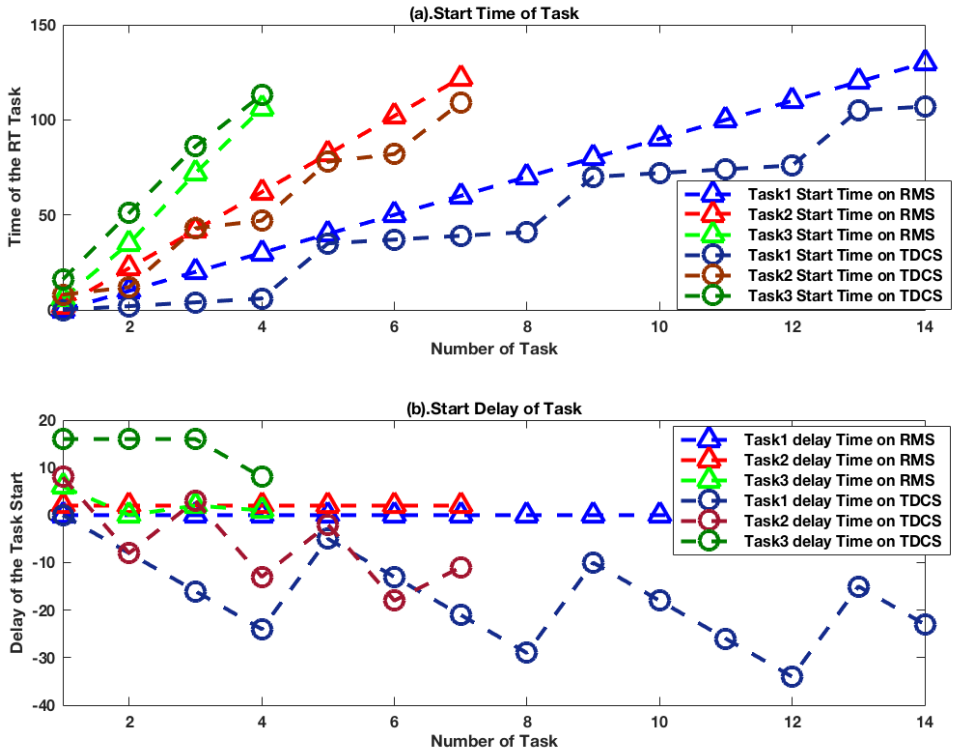


Figure 4.9: Task delay and starting time TDCS vs RMS

4.2.7.1 Slack Stealing

The new scheduling framework shall support the slack stealing concept. Even the multiple periodic tasks would not occupy 100% of CPU time. The background tasks P_b with lower priority than multimedia periodic tasks τ_i can be scheduled in the slack time of τ_i. However, they can be interrupted by τ_i that has higher priority and driven by pre-set low level timer. This concept is demonstrated in Figure 4.5.

4.2.7.2 Hierarchy Priority Scheme

The system can be designed with three tiers of priority ring. The inner ring has higher priority. In the inner most is the Tier 0 tasks that assign to the most emergency tasks such

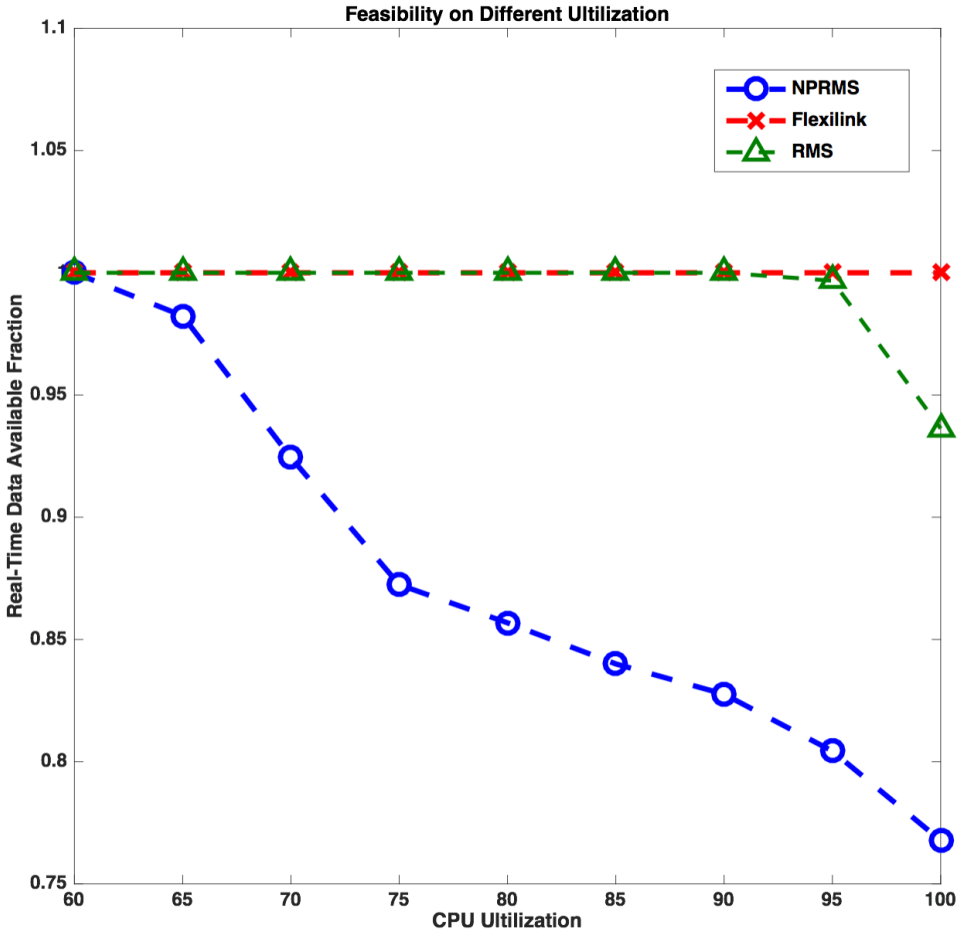


Figure 4.10: Task throughput vs CPU utilisation

as manually terminate the programme etc. The most multimedia periodic tasks will be given tier 1 priority. These tasks will be statistically scheduled off-line by TDCS before the task flow initiated. The background tasks P_b are assigned as tier 2. these can be based on classic pre-emptive scheduling and are scheduled within slack time of tier 0 and tier 1.

- Tier 0 emergency tasks - the adjustable system timer that drive Major Cycle and dispatch the periodic tasks belongs to this tier.
- Tier 1 periodic task - multimedia live tasks. audio samples or frames, video frames, period check and control message.

- Tier 2 pre-emptive background processes.

4.2.8 Advantages and Disadvantages of TDCS

There are a few advantages of integrating TDCS into the current system, especially in mixed criticality system or as RT extension of GPOS.

1. Time deterministic for multimedia periodic tasks. The accurate delay can be estimated in advance.
2. Flexible hyper-period conversion schemes that can create different hyper-periods for different application context.
3. Support high CPU utilisation up to 100% with selectable task mapping scheme.
4. Simplified schedulability test, which enables pseudo on-line scheduling mechanism.
5. Efficiency: The multimedia tasks can be driven by low level system timer not the software interrupts. It has less context switches.

One of the disadvantages of TDCS is similar to classic RMS and EDF that is to rely on worst-case execution time (WCET) of proper off-line planing. If the WCET value is not accurate, that will affect the overall schedulability. TDCS also performs worse than RMS in terms of suitability for hard-real time tasks, because TDCS provides flexibility of re-allocation of tasks in the time line. It introduces the execution jitter, however, the jitter is predicable and can be managed.

4.2.8.1 Contribute to Low Latency

When using TDCS to schedule low latency audio tasks. It shall not allow the unexpected jitter buffer underrun and loss of audio frames happen in this case by providing full schedulability under 100% CPU load. The system shall provide certainty of whether it can accept more tasks or not. It shall prevent the uncertainty of scheduling latency performance when large high priority tasks are presented in the system.

4.3 Conclusions

4.3.1 Summary

In this chapter, we firstly demonstrated and discussed the test results of the real-time audio processing latency of current popular operating systems with onboard soundcard. It provides updated knowledge and issues of using GPOS to process low latency audio signals, especially losses of audio signals when the latency setting is as low as dedicated hardware consoles (<2ms) and with heavy CPU load from audio processing itself via large number of channels and adaptive audio effects. It demonstrates the problem of using GPOS to schedule low latency high frequency tasks.

Secondly, we present a new TDCS scheduling framework for real-time multimedia applications especially for low latency audio processing. The design of TDCS is based on classic cyclic executive concept, but with more flexible allocation of tasks and hyper-period design. With ever increased CPU processing power, the TDCS has been designed with possibilities of integration of Mixed-Criticality (MC) system and GPOS in mind.

The simulation shows TDCS has comparable performance as classic RMS scheduling, especially TDCS is flexible to use different hyper period that trade-off with re-allocation of tasks. In addition, TDCS can achieve high CPU utilisation of RT tasks without loss of executions of tasks. That is important for heavy loaded multimedia processing.

4.3.2 Further Work

There are many areas of this work that can be further explored. One interesting work in theoretical aspects might be to have a generic mathematical model for generating arbitrary length of hyper-periods that is optimised towards different measures such as minimise delay and/or tasks buffer size. One of the main direction of this work is to actually put this in the use of GPOS such as Linux system, so next step, we will try to integrate TDCS into current Linux kernel and provide a feasible interface to applications. It might result in some compromise of original design and practical alteration of the mechanism.

For developing TDCS framework further in couple with current computer science trends. The following areas are worth to consider to work with:

- Different TDCS tasks mapping schemes (As Even As Possible, As fit As Possible).
- Trade off between jitter buffer delay and QoS.
- Possible off-line vs on-line scheduling.
- Multi threading TDCS.
- Multi-core support for TDCS.
- TDCS in virtualisation and cloud computing

As [23] mentioned, perhaps the biggest challenge for real-time in CPS system is the absence of time abstract from different lower layers. That is reflected as difficulty of estimation of WCET in many cases. For multimedia processing, lots of tasks are DSP based. It might be worth to see how modern DSP acceleration mechanism can be accurately timed and reported for upper layer scheduling algorithm such as TDCS.

Desktop computing has moved into the multi-core era whether adopting heterogeneous or homogenous architectures. With an integrated effect processor in the sound subsystem, it could be interesting to evaluate the TDCS scheme to maintain the priority of the low latency audio processing path. It would be challenging to satisfy both flexibility of emerging audio processing tasks and the stability of constant low latency in the audio signal path, especially when the flexibility of processing, routing, synchronising, and feature extraction over multiple channels is needed.

Chapter 5

Delay of Audio Networking and New Low Latency Audio Networking Architecture

In this chapter, we present the work of a new audio networking architecture that supports low latency media delivery over existing networking infrastructure with network traffic convergence. It is organised as the following.

- In Section 5.1, we present a new network architecture and protocol design: Flexilink with its key concepts and features [29].
- In Section 5.2, we present the simulation results of jitter and latency performance of Flexilink comparing with traditional priority-based networks [30].
- In Section 5.3, we present the hardware performance test of Flexilink comparing with the popular professional Audio over IP devices.

The Flexilink concept and design presented in Section 5.1 was conducted in 2011 and published in 2012 [29]. The simulation model of Flexilink protocol was developed and evaluated in around 2015 and the results were published in 2017 as a paper [30]. The work presented in Section 5.3 was submitted for publication in 2018. Also a recent work on scheduling method for Flexilink guided by the author was published in a 2018 paper

[178] that indicates the relevance of the research up to date.

5.1 Design New Low Latency Deterministic Network Protocol: Flexilink

5.1.1 Motivations

The industry revenue of streaming music over the Internet had already overtaken digital downloading and CD sale since 2015. For audio production and live broadcasting, using TCP/IP based network infrastructure to transmit audio has become very popular in recent years. The advantages of using the digital network for live audio production are obvious, including saving analogue cables, providing flexible routing, and overcoming the geographic limitations.

For low latency applications such as live broadcasting or live music performance, using TDM based systems is a standard approach. For example, using AES50 in live console performance or using ISDN for long distance live broadcasting.

However, these TDM technology-based solutions are planned to be made obsolete [130, 131]. The industry is moving toward IP based solutions such as AES67 [179] . The idea is to use IP technology to support productions, broadcasting and distribution network. The vision is that all different types of traffic can be transmitted over a single IP network without sacrificing the timing performance and audio quality.

However, this vision is not without its problems. We had various feedbacks from the industry who adopted the approaches of using the IP network to transmit audio signal for live and low latency applications. They suffered various issues especially when using public Internet in some section of their network. The quality and stability cannot be compared with the traditional TDM based solutions. We believe this is because that the statistical multiplexing nature of the IP based best-effort network cannot provide time deterministic packet delivery even with sophisticated QoS.

Therefore, we designed an network architecture combining the features of both TDM

and best-effort technologies to provide low latency and deterministic network performance that close to the traditional TDM system. It can support flexible configurations of audio streams with different sampling rates and packet sizes. Also, the new protocol is designed with convergence in mind so that is compatible with best effort network traffic such as the IP traffic.

5.1.2 Introduction

In an audio processing system such as in Figure 5.1, the E2E latency arises from different sources [15]: conversion from analogue to digital and back to analogue (ADC/DAC) [26]; networking and routing; the digital console; and the computer system with software plugins (DAW) [12] etc. Buffering is the major cause of latency in IP networks [45], especially in the Internet, but also including intranets. For professional live audio applications, where low latency is required, closed networks with the audio specific layer 1 and layer 2 technology are commonly adopted.

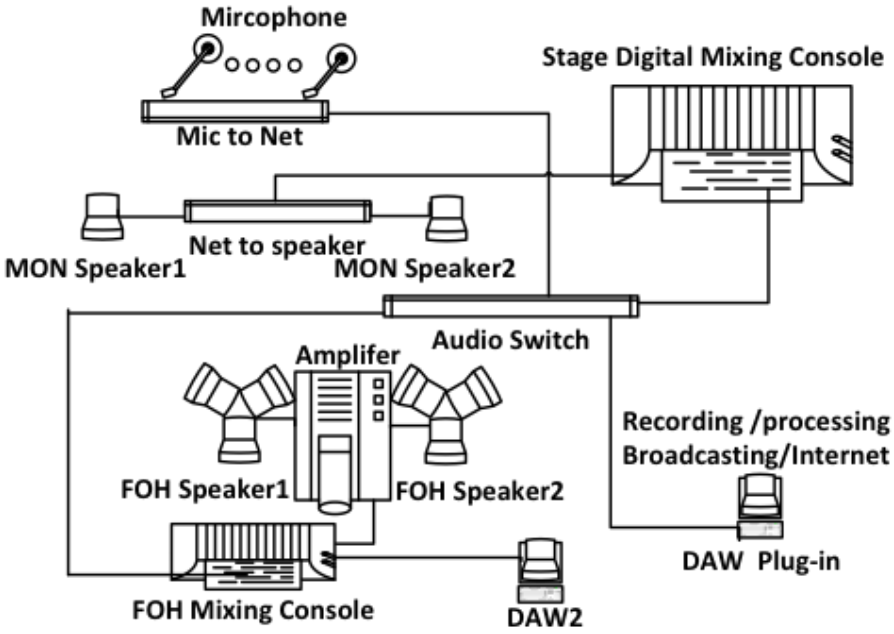


Figure 5.1: Audio Processing System

In high resolution and low latency audio applications, many audio specific networking

technologies modify the existing layer 2 or layer 3 protocols to utilise the current lower layer network infrastructure such as Ethernet. However, it is difficult to achieve a networking architecture, which supports both time critical audio data and also best effort data (such as file transfer and emails). Converged networks based on IP, scalable from LAN to WAN are required to support the vast (and growing) interactive audio/video media traffic on the Internet. However connectionless packet architectures are inevitably a problem for deterministic data, especially when low latency is required. Current QoS, traffic engineering and over-provisioning solutions cannot solve the entire problem: they complicate the system, and increase power requirements and cost. The professional audio networking industry is reluctant to use current Internet solution for time critical applications. Instead, they normally adopt solutions based on specific protocols designed and modified from physical layer up to layer 3 such as AES50, EtherSound, CobraNet etc. For low latency live audio, TDM based protocols such as AES50 can provide very good performance. The proprietary AES50 router can provide latency as low as a few samples with a fixed number of channels reserved for packet data.

It appears that there is no unified network solution to provide flexible and bandwidth efficient support for both low latency deterministic traffic and best effort traffic. There is also an issue with multi-channel digital audio streams with a range of sampling frequencies and variable bit lengths, and the need for flexible routing and channel assignments. Current multiplexing methods are insufficient to support them without sampling rate conversion and data format rectification.

The proposed architecture is aiming to effectively support both best effort data and time deterministic data (audio sample packets). It should also interwork with existing network infrastructure and protocols at maximum compatibility. Since the current physical network layer (such as full duplex Ethernet) can be viewed as time deterministic bit pipe there is no reason why a time deterministic logical control layer cannot be implemented. This allows a guaranteed Quality of Service (QoS) and expected Quality of Experience (QoE) for the higher layer protocols. The exact time delay for the transmission of time-critical data can therefore be estimated.

Based on earlier work [180], we proposed a novel unified network architecture that combines the advantages of TDM and best effort networks. The proposed layer 2 protocols,

"Flexilink", have been developed along with a prototyped network processor architecture and interface cards. Compatibility with existing Ethernet infrastructure is maintained. Flexilink can operate at full-duplex mode, where non-deterministic CSMA-CD can be avoided.

5.1.3 The Architecture Design of Flexilink

5.1.3.1 The Rationale of the Design

User generated data can be categorised as (i) data to be transmitted as time-deterministic with constant intervals and predicable delay, i.e. real-time data; (ii) data to be transmitted at earliest opportunity but without the constraints of real time, i.e. best effort data. The network also conveys network management data.

This gives us the three data categories as follows:

1. Synchronous Flow (SF) for audio/video and other time deterministic data.
2. Asynchronous Flow (AF) for best effort data.
3. Control Message (CM) for session control and link management.

The theoretical requirements of a single SF can be determined; for example - transmitting a 44.1kHz sampled CD with 16 bit samples, without any headers and error checking mechanism will require 1.4112Mb/s. Compressed formats will also have a nominal bit rate allocated (with the associated compromised quality). So, for a link with sufficient bandwidth, we are able preallocate spaces or slots for the SF data packets. AF data can be transmitted in the gaps between SF data packets. A simple theoretical link model is shown in Figure 5.2, where SF data packets are transmitted at constant time t_0. Since SF packets are of variable length, gaps to be filled are also variable.

Figure 5.2: Ideal Link for the Traffic

To ensure SF data are transmitted in a time deterministic manner, resources (bandwidth requirements) need to be reserved when the link is established. Individual SF data packets are identified by the position of the SF packets within the stream (similar to time slots in a TDM frame).

Control messages (CM) with associated protocols are used for establishing links and negotiating resource reservations.

Flexilink supports variable length SF packets where the varying length gaps are filled by any AF traffic awaiting transmission. To facilitate this a small header is added to an SF packet. The header is simplified to contain only the length of the SF packet plus basic error checking bits. Therefore, there is no need for AF data to be encapsulated with a new header when it is fragmented by the SF flow. This also simplifies the hardware logic required to forward both SF and AF traffic effectively.

This operation could be considered as a continuous AF stream frequently interrupted by the frequent real-time SF data, since the main parameters are all known: the speed of network link, the data rate of SF, and length of the SF data packets. There is no additional reassembly required to reconstruct the segmented AF traffic. In addition, this design can achieve the maximum utilisation of link bandwidth with all the gaps (unused capacity) filled by the available best effort traffic. [181] and [182] proposed a similar system, but [181] has fixed TDM channels so that the capacity allocated but not carrying data is not utilised. [182] has two types of traffic, but the low priority traffic is fragmented with an additional header carrying type and destination information being required for each fragment.

5.1.3.2 Architecture Design

The network node supporting the proposed Flexilink would have a common network architecture below in Figure 5.3, with two major functional blocks: the “control unit” for setting up and tearing down call flows, allocating the resources, and route finding algorithms; and the data “forwarding logic” for fast forwarding and switching the data for both SF and AF data.

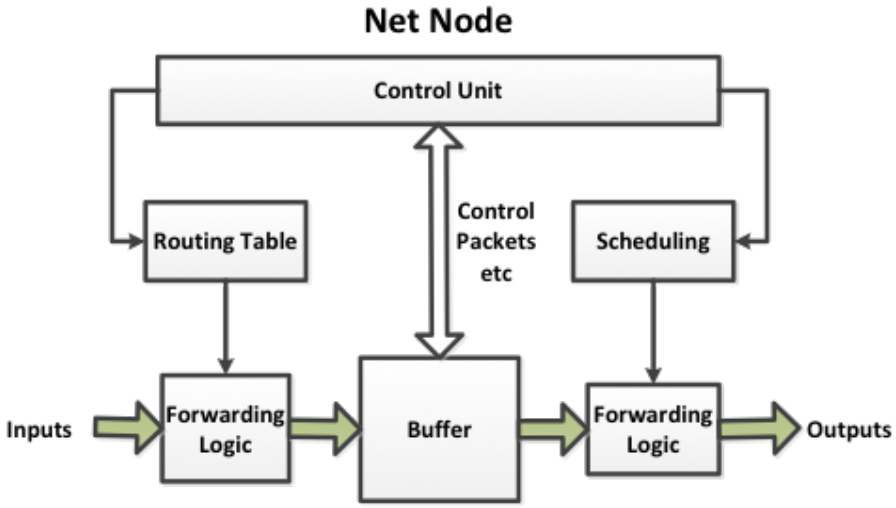


Figure 5.3: Architecture of Network Node

First of all, there is a process for setting up and tearing down the link between two end points with an intelligent time slot map allocation algorithm, which is based on the available network link resources and the requirements of deterministic traffic (SF). This setup can be managed by control messages (CM).

The CM can be implemented as standard IEC 62379-5-2 (Common Control Interface for networked audio and video products) messages [183]. Essentially, they are considered as normal AF traffic for the purposes of the link. However, whereas normal AF traffic is routed to the output to be transmitted over the link, the CMs are directed to the controller. CMs have priority over AFs on each link. For audio traffic, the packages in a SF can be as small as audio samples, for example 48000 packets per second with a payload of 4 bytes.

Design Header of SF

The simple header added to SFs contain only the length information. To minimise the header cost, the length of the header is also variable as shown in Figure 5.4(a) and Table 5.1:

Table 5.1: The length of a SF packet's header

Header Length	Data Length
1 byte	0 ~ 15 bytes
2 bytes	16 ~ 255 bytes
3 bytes	256 ~ 4096 bytes

A 1-byte header consists of 4 bits to encode the length information; 3 bits for CRC; and 1 bit flag to indicate if there are further header bytes as shown in Figure 5.4(b).

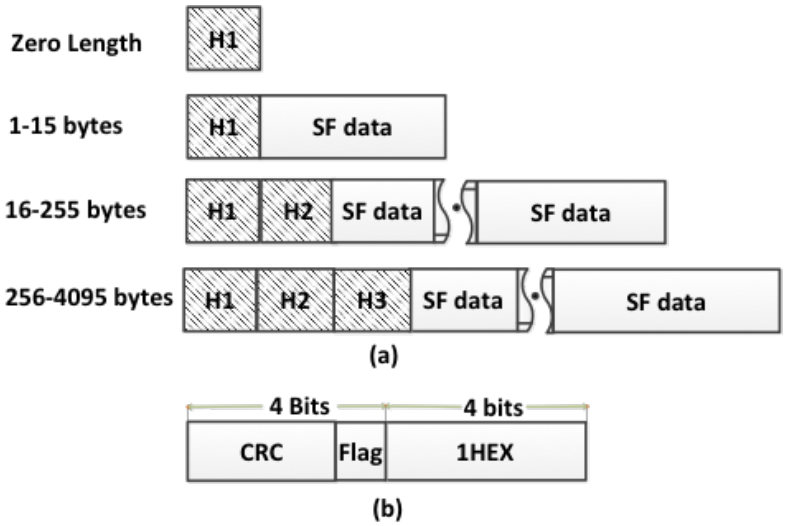


Figure 5.4: Header of SF packet

Interface Architecture to Support Flexilink

To maximise compatibility, Flexilink should be able to use the existing physical network interface. However to support the proposed Flexilink protocol, a new media access control (MAC) layer architecture needs to be considered, which allows AF and SF to be treated differently. Figure 5.5 shows the simplified Flexilink MAC layer in which AF and SF have separate buffers and copy logic allocated for them.

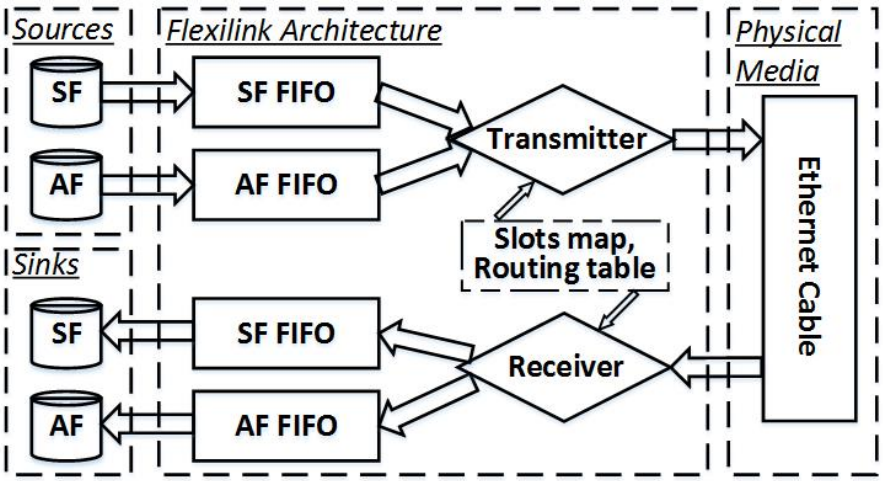


Figure 5.5: The MAC layer design of Flexilink

Layered Traffic Model

The theoretical link traffic slot allocation is shown in Figure 5.6. The SF and AF access may be implemented over an existing point-to-point link mechanism in order to utilise the current network infrastructure. Figure 5.6 shows a practical implementation of the layered traffic model.

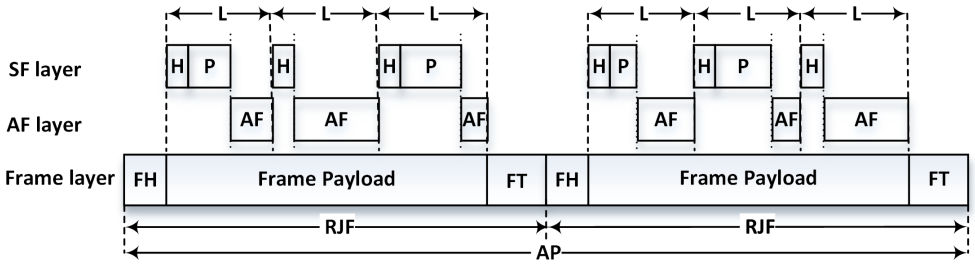


Figure 5.6: The layered structure of Flexilink

The Frame layer can be a standard fixed size Ethernet frame, so the position of an SF packet in relation to the start symbol of frame can be used as a reference for identification of the SF data packets.

Supporting Flexible Multichannel Audio Streams With Different Sampling Frequencies

A current problem in networking audio traffic is the lack of flexibility to support arbitrary numbers of audio channels with different sampling frequencies and of compressed or uncompressed data format. In the proposal given here, multiple channels of different sampling frequencies can be readily supported as long as the capacity of the link above the frame layer is greater than the total bandwidth requirements of number of the SFs.

For Ethernet physical media, the Ethernet Jumbo Frame format can be used to maximise the capacity of available bandwidth and minimise the cost of inter-frame gaps. The time slot map is allocated within the payloads of the frames. The time slot map allocation algorithm will ensure the SF streams are distributed evenly. A phase control algorithm corrects the delivery of samples at the end points to ensure the samples are rendered at certain jitter requirements. The Precision Time Protocol can be adopted for accurate timing and provision of clocking.

5.1.4 Ethernet Implementations of Flexilink

The Flexilink design can be implemented in different types of physical layer as long as the bit transfer rate can be fixed and the bit pipe is guaranteed.

This section will discuss how Flexilink is implemented using 1 Gigabit Ethernet infrastructure, since 90% internet traffic is originated from Ethernet or WiFi, and Gigabit Ethernet is commonly used by other proprietary audio network technology.

5.1.4.1 Ethernet Frame Structure for Flexilink

To maximise bandwidth efficiency, Flexilink adopts the Jumbo Frame format in which the payload size is greater than the standard 1500bytes common Ethernet frame. The maximum size of a Jumbo Frame cannot exceed 11455 bytes in order to allow CRC algorithm to effectively working [184]. Another limitation is that some Ethernet PHY chips do not support frames larger than 9000 bytes.

To be able to accurately access the allocation slots and the positions of SF packets, it is preferred to have a fixed allocation period at the frame layer. In this case the allocation period of 124.96μs is chosen to be slightly faster than 125μs since the 125μs is the frame cycle used in many other designs such as Synchronous Digital Hierarchy (SDH), Firewire and full speed USB.

The network devices can use standard AES51 [185] negotiation packets (which are standard Ethernet MAC packets) to setup the links. Once both ends are in Flexilink mode, a "Reduced Jumbo Frame" (RJF) format is utilised to maximise the payload. The RJF eliminates some unnecessary parts of the standard Ethernet frame (for example, the source and destination addresses, because all frames are sent from the node at one end of the link to the node at the other) to give more payload space to the allocation periods.

It is described as below:

1. 2 bytes preamble + Start Frame Delimiter (SFD).
2. 5 bytes AES51 packet type and timing information.
3. 7785 bytes payload data.
4. 4 bytes FCS.
5. 14 bytes inter frame gap (IFG).

In total the RJF frame size including IFG is 7810 bytes long. Two successive RJF combine together to make a 124.96's allocation period (AP) at full duplex 1 Gigabit Ethernet link. This design is to guarantee the 8000 allocation periods/second can be transmitted over 1 Gigabit Ethernet link. Each allocation period has 15570 bytes payload space to transmit SF and AF traffic. The theoretical bandwidth utilisation can up to 99.6%.

The FCS is only used to check that the link is working reliably. Routing of AFs and SFs does not wait until the FCS has been received, and for many media formats it is better to deliver data with a few bit errors than to discard whole frames.

5.1.4.2 Methods for Support Low Latency Multiple Sampling Rate Audio Streams

The following example demonstrates how Flexilink supports multichannel audio streams with different sampling frequencies.

Assuming that we have audio streams as (i) Flow 1: 48kHz mono 24bits; (ii) Flow 2: 44.1kHz stereo 16bits; and (iii) Flow 3: 96kHz mono 24 bits; we need to transmit (or multiplex) them using Flexilink.

The audio data of all three flows plus other possible metadata is less than 15 bytes; therefore 1 byte SF header is needed for each packet.

The control unit of the sender node will reserve the network resource and allocate transmitting slots for each flow as SF packets; i.e. the slot allocation map will be established and agreed by both ends of the link by CM messages. Within each packet, in addition to the audio data, there is 1 byte containing the synchronisation information as specified in 7.3.2 of IEC 62379-5-2 [183]. (An additional 1 byte per channel of overhead may also be added to carry channel status CRC, etc.)

1. For flow 1: The number of allocation slots within allocation period should be $\geq$ 48/8. Therefore 6 slots are needed from one allocation period with each slot being 3+1+1 = 5 bytes long.
2. For flow 2: The number of allocation slots $\geq$44.1/8 = 5.5. Hence 6 slots as well, with each slot being 4+1+1 = 6 bytes long.
3. For flow 3: The number of allocation slots $\geq$ 96/8. Hence also 12 slots, with each slot being 3+1+1 = 5 bytes long.

Note that for flow 2 although the number of slots we allocated is more than actually needed for delivery of audio samples at 44.1Khz, this is because the number of slots is rounded up to a whole number. Approximately every twelfth slot there will be an empty SF data packet for flow 2. However this is not a problem since the header will indicate an empty packet, therefore the space can be used for AF data.

Having AES51 packet type and timing information in the Flexilink over Ethernet implementation, the sender and receiver can be synchronised easily. When accurate time

information needs to be distributed, the Precision Time Protocol (PTP) can also be used for synchronisation between sender and receiver to avoid potential drift of clock and jitter over a period of transmission.

Typical latency for synchronous flows would be 3 to 6 microseconds per hop, plus the "speed of light" delay in the transmission line.

5.1.5 Interconnection With Existing Network

In a mixed network environment with multiple low latency audio flows of different sampling frequencies, bit depths and numbers of channels, can be well supported by Flexilink as described in Section 5.1.4.2

Normal data transfer, typically IP-based traffic (e.g. file transfer), is mapped into the AF traffic and so is safely transmitted over a Flexilink network.

The architecture design maintains separation between the AF and SF data and thus ensures that there is no interference between two types of data whilst fully utilising the bandwidth not used by SF data.

For IP based audio data, Flexilink can map audio IP packets to SF data packet according to the identified service priority. So Flexilink should not negate the original QoS.

In the case where a Flexilink interface is peering with a normal Ethernet interface which does not support the Flexilink mode, Flexilink can (i) switch to standard Ethernet mode or (ii) negotiate an Ethernet AVB mode to prioritize delivery of the audio traffic.

5.2 Simulation of the Flexilink Architecture

5.2.1 Motivation of the work

In the last section, we proposed the new low latency audio network architecture: Flexilink. It adopts a simplified model that combines the advantages of TDM network and the best-effort network such as IP based Internet. The design philosophy of Flexilink takes the future trend of the audio delivery into account: The system should deliver multiple audio channels each with the different source transmission rate. The metadata, compressed and uncompressed audio signals, and other types of traffic need to coexist in the same transmission channel with low E2E delay. Mapping the new architecture to the existing network infrastructure such as Ethernet is also proposed. How does this design compare to the current networking technology needs to be investigated. In this section, we use a simulation model to evaluate the performance of Flexilink in comparison with the existing Ethernet and priority based RT Ethernet and present the results of the simulation tests.

5.2.2 Simulation Model

The simulation structure model is illustrated in Figure 5.7 based on the MAC layer design of Flexilink demonstrated in Figure 5.5. The main Flexilink modules are simulated in node A, such as two traffic sources, the dual-buffer model, and the control logic. The transmitter and receiver work together to schedule the slots allocation and transmission. Flexilink can use the existing physical media. In this section, a simulation model is going to be built for Flexilink to verify the performance of this architecture. The model is realised on the SimEvents [186] platform within Simulink, which is developed for the discrete event simulation. The main parameters associated with each block are listed in Figure 5.8.

Assuming node A is the transmitter and node B the receiver, connected by the *cable* block. The propagation delay is calculated as dividing the cable's length by $2 \times 10^8 m/s$, 2/3 of the speed of light. It is the same for every packet as it depends on only the materials

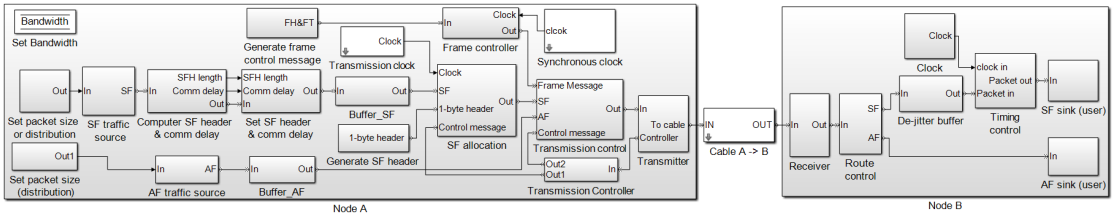


Figure 5.7: The Flexilink architecture simulation model

of the physical media. The SF and AF sources are controlled separately given a packet generation rate and packet sizes (a specified distribution). The *set SF packet header* block will first check the length of the SF packet's payload and compare it with the parameter settings expressed in Table 5.1, then a calculated header will be generated and attached to the SF packet. Following the two traffic sources, there are two buffers to store the packets before they can be forwarded to the transmitter.

The *SF allocation* block works with the transmission clock and the transmission controller. It determines the SF packets' allocation and transmission time. When there is an empty slot, it will accept a 1-byte header from the *generate SF header* block. The *transmission control* block schedules the transmission of both SF and AF as well as skipping the frame control messages. The frame control messages containing the FH and FT are used to encapsulate the SF and AF packets into a RJF. The size and position of a RJF are fixed on a link as presented in Figure 5.6. The fix-sized frame control messages are transmitted at a constant frequency, which is controlled by the *frame control* and *synchronous clock* blocks.

All packets will stay in the single server located in the transmitter for some time depending on the value of transmission delay calculated as dividing the packet length by the link bandwidth. The transmitter has a feedback scheme which gives the transmission information back to the *transmission controller* block to further schedule the transmission. The transmission controller decides when the packets are able to be transmitted, depends on the packet size and available bandwidth. The link slots are allocated based on SF transmission rate characteristics to minimise the SF delay. An AF packet may need more than one slot to be transmitted.

Given a 44.1K audio transmitted on a one Gigabit Ethernet link, 5.5 slots are needed in

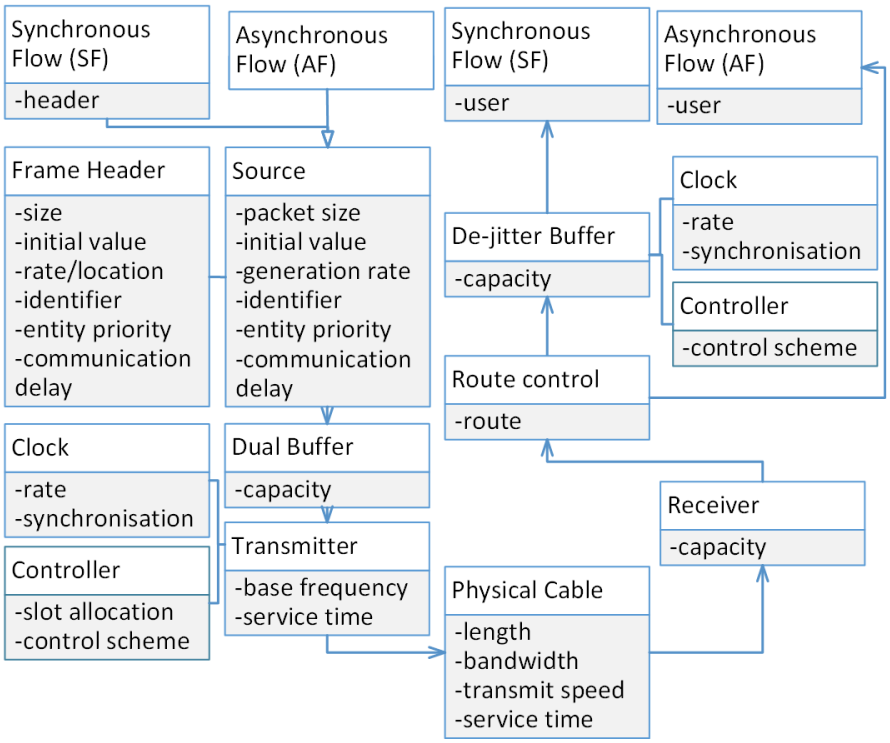


Figure 5.8: The parameter list diagram

one AP. In practice, 6 slots will be allocated. Therefore, there will be some empty slots, one empty slot in every two APs in this case. An empty slot needs a one-byte header to inform the controller so that it can transmit AF packets to make a better utilisation of the available bandwidth. The one-byte header is generated whenever there is no SF packet waiting to be transmitted in a slot.

In node B, packets are first received and stored by the *receiver* block. The *route control* block will route the packets to different sinks (users) identified by their locations on the link. A de-jitter buffer is applied to SF packets before they are sent to the sink, which is controlled by the synchronous clock given a fixed frequency the same as the SF traffic source. Some SF packets will have a little bit of jitter caused by the transmission (empty slots) and synchronisation.

5.2.3 Simulation Overview

The Flexilink protocol is designed to provide a guaranteed deterministic traffic transmission with an acceptable low latency. It also supports the best effort traffic without affecting the RT traffic. In the following, several scenarios are employed to evaluate the performance of Flexilink, including increasing the amount of AF data, adding a burst of traffic and having multiple sources. The amount of AF traffic is increased through all the scenarios to verify whether the AF traffic would have any influence on the transmission of SF traffic.

In this model, assume the SF traffic source is a flow comprised of 128 audio channels. Each channel has a 44.1 KHz stereo 16 bits flow. The SF packet will have a 1-byte meta data, therefore, the size of each SF packet is 5 bytes and the total rate of the SF source is 640 bytes per sample (packet) in average. The size of each SF packet is variable which obeys a uniform distribution between 390 and 890 bytes. According to Table 5.1, each SF packet will have a 3-byte header. The AF traffic source is simulated using a uniform distribution given a minimal and a maximal packet size, 64 and 1518 bytes, respectively. Flexilink will put AF packets into the gaps between two successive SF packets, therefore, it is the amount of AF data rather than the size of each AF packet that matters in this simulation.

Using a one-Gigabit Ethernet link, Flexilink guarantees 8 K APs per second. Thus 48 K allocation slots are needed when transmitting a 44.1 KHz audio. The basic parameters that will be used all through these several scenarios are listed in Table 5.2.

Table 5.2: The global parameters

Parameter	**Value**	**Parameter**	**Value**
SF packet frequency	44.1 KHz	FH size	7 bytes
SF allocation slot	48 KHz	FT size	18 bytes
Frame frequency	16 KHz	Cable length	100 meters

For comparison, two Ethernet network models are also built. The main differences are listed as follows.

1) Basic Ethernet Network This simulation model has only the *transmitter* and *receiver* blocks. There is no QoS guaranteed transmission in this model.

2) Priority based Ethernet It has a higher priority for the SF port than the AF port, as well as a priority based transmission buffer, which gives the SF packet a higher priority to be transmitted.

In both models, the two traffic sources are kept the same, but without the Flexilink reservation, allocation and control schemes. All the parameters and settings used for the sources, the cable, the sinks and so on will also be kept the same as the Flexilink model.

5.2.4 Simulation Scenarios

Given the global parameters described in the last section, we can calculate how much bandwidth is left for the AF traffic. This model simulates variable SF packet sizes. The average SF bandwidth utilisation is 22.688% of the total bandwidth including SF packet headers of the empty slots. This amount of SF data reflects the AV traffic in the real world scenario according to paper [187]. The frame's control messages take 0.32% of the total bandwidth. Therefore, there is about 77% of the total bandwidth available for the AF traffic. In the following, several simulation cases are used to compare their performances. Table 5.3 gives a brief introduction to each case.

Table 5.3: Cases introduction

Test Case	**Brief Description**
Linear stress test	Increase the AF load of the link.
Burst test	Give a burst of AF traffic to each scenario.
Multiple sources	Extend each scenario to three SFs and three AFs.
Multiple-port mixed sources	Given two source nodes with both SFs and AFs.

5.2.4.1 Case One: Linear Stress Test

In this simulation, the amount of AF traffic is increased to increase the overall network load in each scenario to see whether it will affect the transmission of the SF flow. The AF flow will take 20%, 40%, 60%, 80%, 100% and 120% of the total bandwidth, respectively.

The simulation results are presented in Figure 5.9, the average End-to-End (E2E) delay, and Figure 5.10, the jitter, for each model against the increasing AF load. The jitter is calculated as the variance of the E2E delay in each scenario.

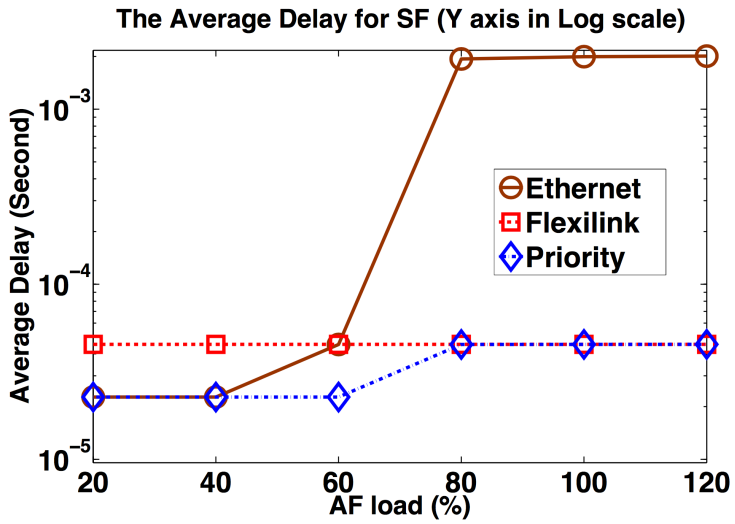


Figure 5.9: Average Delay in Case One

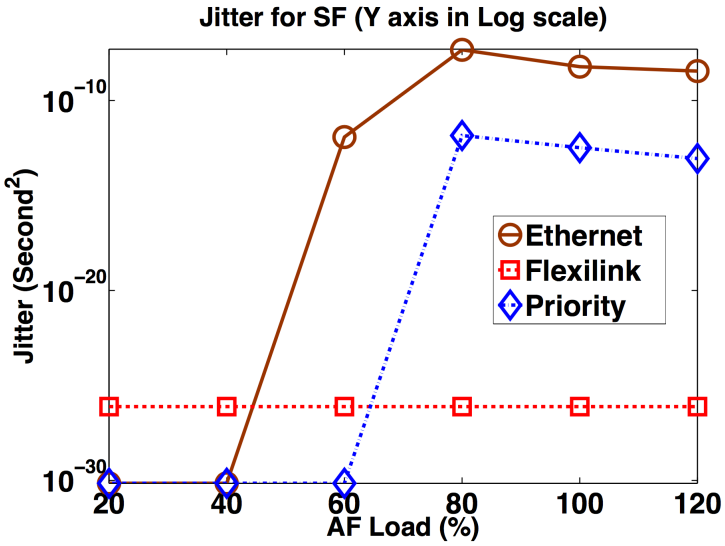


Figure 5.10: Jitter in Case One

The basic Ethernet network without QoS implementation has a poor behaviour unless there is only a little traffic on the link. Generally speaking, Flexilink maintains a very stable performance no matter how much AF traffic we have pushed to the link. However, Flexilink performs slightly worse than the Priority Ethernet (PE) when it has a low

network load. This is because Flexilink has an allocation mechanism to guarantee SF's transmission, which requires SF packets to wait in the buffer for some periods for the right slot to be transmitted. It gives one sample's delay in this situation. While in the PE model, the SF packet can be transmitted immediately unless there is a packet being transmitted, which will give a maximal delay of 12.144 μs. The Flexilink SF can be configured to be close to the source data rate in order to reduce the latency, although additional mechanisms may be needed for automatic configuration. However, the PE network will get worse when having a burst of traffic and/or multiple SFs, which will be discussed in the next several cases. PE also becomes precarious and gets more delay when given massive AF traffic.

The simulation models all have a de-jitter buffer implemented, which alleviates the time fluctuation problem. It is designed in the Flexilink architecture, while in reality, most network switches do not have this mechanism.

In addition, AF sources will not always obey a uniform distribution in practice. Sometimes people may occasionally need to transmit or download a big file, then the burst of traffic may appear during some periods. Therefore, we are going to add a burst of traffic source with massive data which will take up all the bandwidth for some time.

5.2.4.2 Case Two: Burst Test

In this case, a burst of traffic source is added to every model. It will generate plenty of 1518-byte packets during a small period with a frequency of 82345 packets per second, which will be able to take up all the available bandwidth on the link. The E2E delay and jitter for the SF are illustrated in Figure 5.11 and Figure 5.12, respectively. It can be seen that the basic Ethernet network model has a significant delay and dramatic fluctuation. The E2E delays are similar for the Flexilink and PE networks. However, the PE model has a much larger jitter, than that of Flexilink. If we represent the jitter as standard deviation, it is around 1.88μs for the PE, which exceeds the $20ns$ limitation to be audible [142]. For Flexilink, it is about $1.54e^{-6}ns$, is much lower than that.

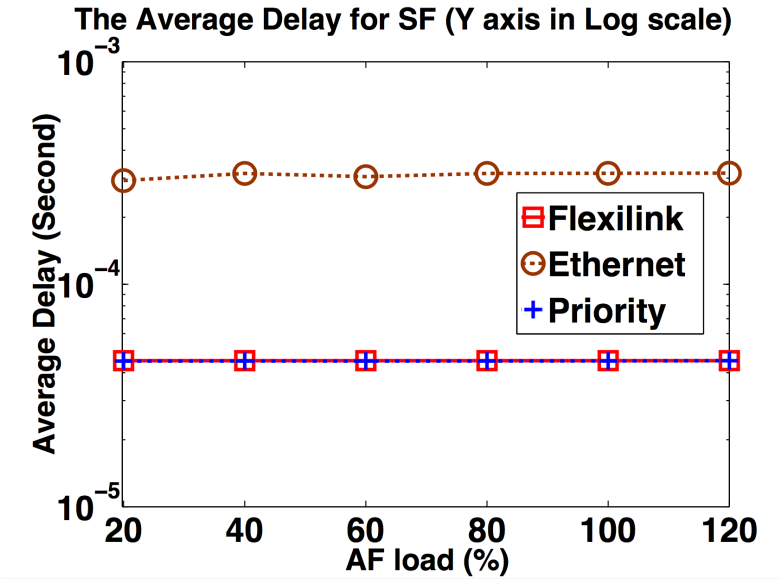


Figure 5.11: Average Delay for Case Two

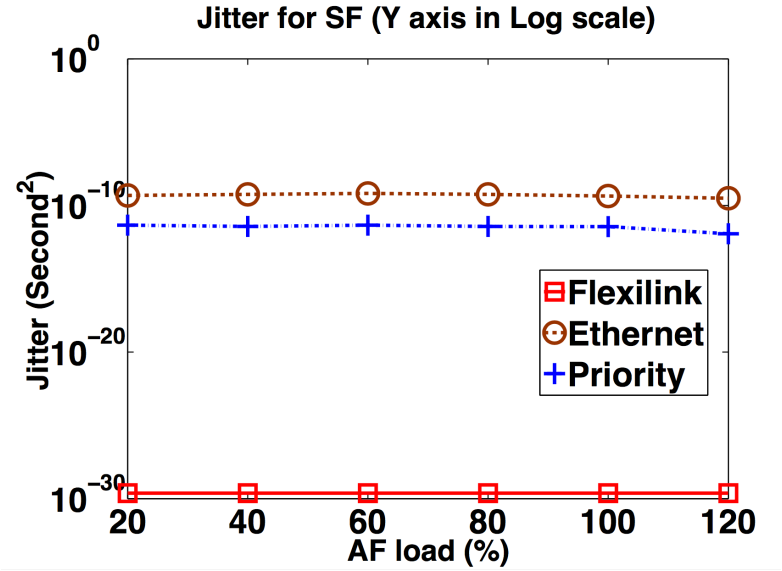


Figure 5.12: Jitter for Case Two

This change in performance can be explained by Figure 5.13, which shows the detailed E2E delay for every SF packet in scenario three which has an AF load of 60%. When there is a burst of traffic, it will take up all the available bandwidth, which leads to an

overflow in the buffer. The following packets will take more buffering time and may need to wait for another sample's period in the de-jitter buffer at the receiver. Thus, there is a jump in the PE's performance, which gives a higher delay as well as a higher jitter. We can also find that Flexilink is not affected by the burst of traffic.

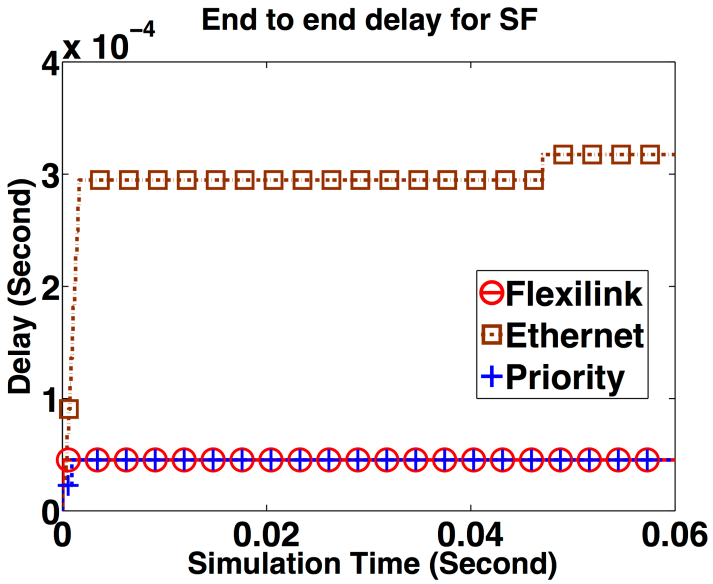


Figure 5.13: The E2E delay for each packet for Case Two

5.2.4.3 Case Three: Multiple Sources

The model is extended to several SF and AF sources, which is a more general situation in practice. In this simulation, we will use three SFs and three AFs. The amount of traffic in each flow will be decreased to keep the total traffic the same as the last two cases. In the PE model, SF packets have higher priorities than AF packets, but there are no priorities among the three SFs.

Here we choose the third scenario with an AF load of 60% as an example to analyse their performance. The detailed E2E delays for Flexilink compared with the Ethernet and PE models are illustrated in Figure 5.14 and Figure 5.15, separately.

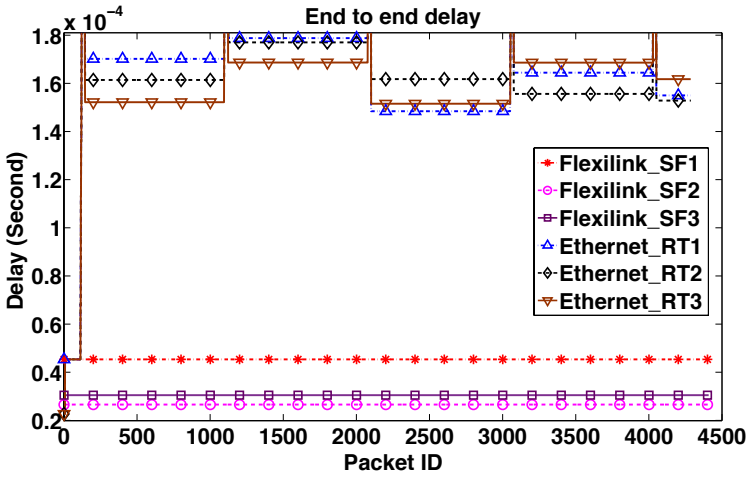


Figure 5.14: Detailed E2E delay for SF in Case Three

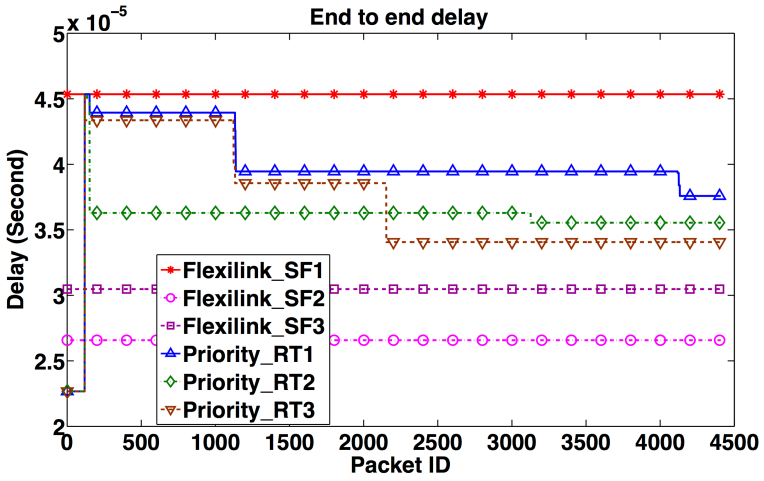


Figure 5.15: Detailed E2E delay for SF in Case Three

It can be seen from Figure 5.14 that the E2E delays for all SFs preserve at a relatively high level and experience fluctuations during some periods in the basic Ethernet model. It also started to drop SF packets with a rate of 2.8%, approximately. From Figure 5.15 we can see that SFs all have experienced a jump and several steps of decrease in the PE model. Different SFs do not have the same delay at the same time, and there are no

patterns during a period. This relatively random delays will also lead to a higher jitter. Each SF in the Flexilink model holds the same E2E delay, which leads to a very small jitter. Different SFs have different delays. This is because a world clock was used in the model which will cause a different but stable delay within a sample period, depending on the synchronisation between the transmission and receiving ends.

5.2.4.4 Case Four: Multiple-port Mixed Sources

Considering the following scenario as described in . Nodes A and B are both transmitting several SFs and AFs to node C, simultaneously. There is only one node can be set to a higher priority, for example, node A. Based on Case Three, we assume node A has one SF and two AFs, while node B has two SFs and one AF. Within each node, the SF has a higher priority than the AF. All other parameters are kept the same as Case Three. Figure 5.17 shows the detailed E2E delay results.

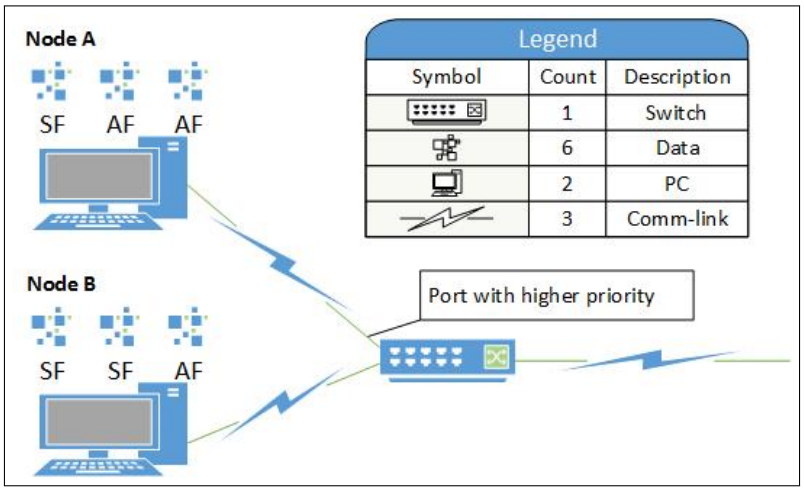


Figure 5.16: Case Four settings description

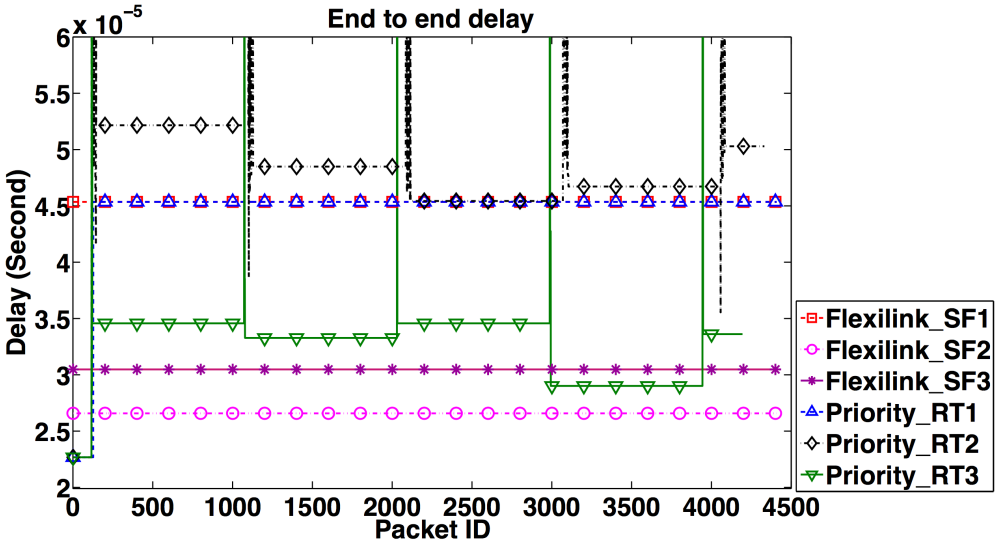


Figure 5.17: Detailed E2E delay for SF in Case Four

A similar conclusion can be given to the basic Ethernet and Flexilink network models as Case Three. However, the PE model experiences a big change where there are some spikes whenever there is a burst of traffic, which means some SF packets have significant large delays. The third SF also has a higher average E2E delay than any other SFs in the PE model or the Flexilink model. In addition, the second and third SFs began to drop SF packets with an average rate of 1.77% and 4.88%, respectively.

5.2.5 Simulation Results Analysis

In the Flexilink network model, the average E2E delay of each SF packet for the first SF in all the scenarios is approximately 45.35μs. The delay is mostly caused by the transmission buffer, the transmission delay (transmitter), the propagation delay (cable) and the de-jitter buffer [188]. The values of these parameters are listed in Table 5.4. The total delay, a summation of the delays mentioned above, is shown at the end of the table. The calculated total delay is close to the value obtained from the simulation, 45.35μs.

Table 5.4: Factors that cause delay

Factor	**Delay (Second)**
Buffer	1.038×10^{-5}
Transmission delay	5.120×10^{-6}
Propagation delay	5.000×10^{-7}
De-jitter buffer	2.933×10^{-5}
In total	4.500×10^{-5}

Note that the average delay caused by the buffer is slightly less than 10.417μs which is half a sample period of the transmission link which has a base frequency of 48 KHz in this case. Similarly, the average delay caused by the de-jitter buffer is about 1.3 audio samples. If we change the synchronous clock at the receiver end, the delay may be slightly different but within an audio sample's period.

$$D_{difference} = D_{Flexilink} - D_{PE} \tag{5.1}$$

$$J_{ratio} = J_{Flexilink}/J_{PE} \tag{5.2}$$

Table 5.5: Average E2E delay and jitter improvement for Flexilink compared to Priority based Ethernet

Case	**Delay Difference (μs)**	**Jitter Ratio**
Case 1 [a]	+ 11.35	$2.48e{-}14$
Case 2 [a]	+ 0.16	$6.71e{-}19$
Case 3 [b]	- 3.41	$1.934e{-}19$
Case 4 [b]	- 8.07	$8.06e{-}21$

[a] Delay and jitter are calculated as the mean of six scenarios (AF loads).

[b] Delay and jitter are calculated as the mean of three real-time flows.

A summary of the average E2E delay and jitter performance improvement for Flexilink compared to Priority based Ethernet is presented in Table 5.5, which are calculated using

Equation 5.1 and 5.2, respectively. In Equation 5.1, $D_{difference}$ denotes the different difference, while $D_{Flexilink}$ and D_{PE} denote the average delay of the Flexilink and PE networks, respectively. Similarly, J_{ratio} denote the jitter ratio, while $J_{Flexilink}$ and J_{PE} denote the mean jitter of the Flexilink and PE networks, respectively.

Through all scenarios, we can see that the E2E delay over the Flexilink network will not increase as the amount of AF data grows, and the jitter for all scenarios is close to zero. For comparison, both the E2E delay and jitter increase rapidly in the Ethernet model. The PE has a similar low latency and jitter when there are a small amount of AF data, but it gets worse when having massive AF data and/or multiple SF sources. The burst of traffic also has a significant influence on the performances of the basic Ethernet and PE models. Flexilink keeps performing well when more and more AF data are pushed to the Ethernet link, even with the burst of traffic and/or multiple sources.

5.3 Flexilink Hardware Test

5.3.1 Hardware Implementation

The Flexilink architecture has been prototyped as hardware switches with collaboration of external partners. The key building blocks of these devices are the data forwarding unit and the control unit. The data forwarding unit needs to fast forward uses cut-through forwarding for the SF data packets as well as storing and forwarding the AF data, which can be implemented in hardware logic by FPGA. The control unit needs to flexibly allocate the resource for multichannel SF data requests, such as positioning the SF data packet in the continuous bit stream, and allocating the SF data packets evenly to allow phase correction algorithms work efficiently. The control unit needs to be an effective general purpose CPU with accelerated networking processing capabilities.

The current version of prototype has 4 Gigabit Ethernet ports, four small form-factor pluggable (SFP) ports for optical fibre or digital video extensions, 2 AES10 and 1 AES59 interfaces. The picture of the prototype hardware is shown in Figure 5.18. It can accept audio native digital audio stream in AES format and convert them into Flexilink traffic

over Ethernet.

Figure 5.18: Picture of Flexilink switches

5.3.2 Testing Scenarios

We run two hardware tests to mainly test the latencies and jitters of Flexilink hardware under different circumstances. The first *Test Case 1* see Figure 5.19 is a proof of concept test to find out a general benchmark of performance. The second *Test Case 2* see Figure 5.20 is a stress test to find out the effects of network performance under heavy background traffic in comparison with other popular Audio over IP (AoIP) technology.

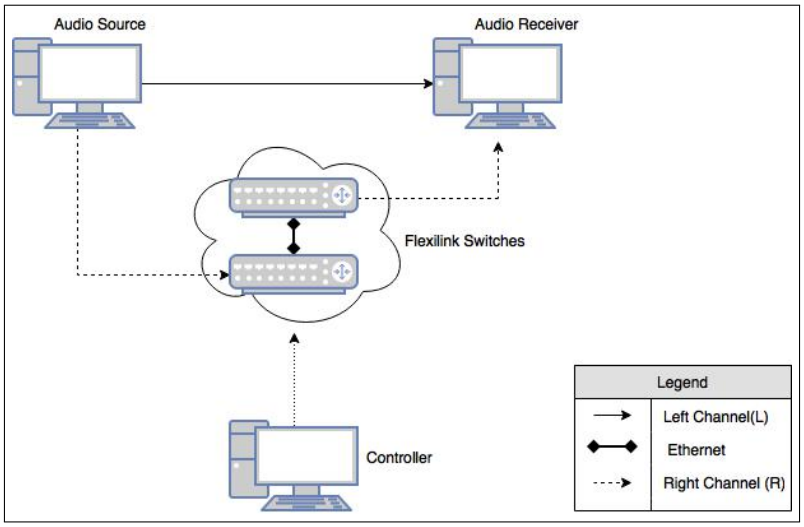


Figure 5.19: Test Case 1 Proof of Concept Test

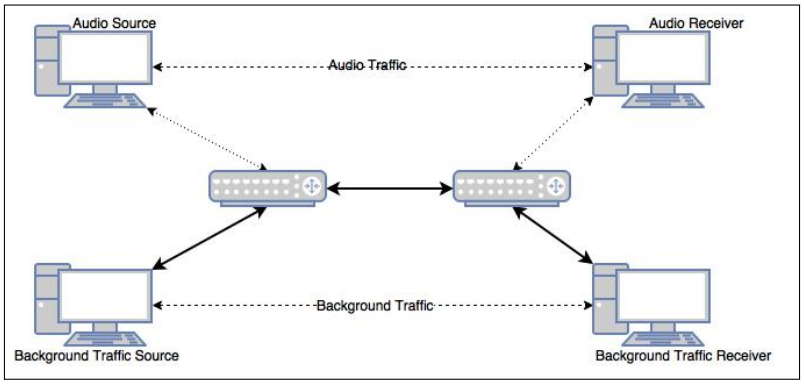


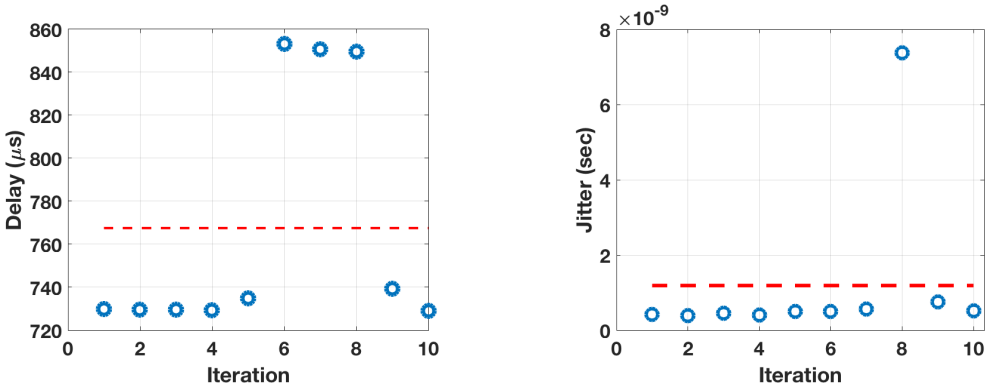
Figure 5.20: Test Case 2 Stress Test

The Test Case 1 topology is to route both channels of a stereo audio pulse trains separately from Audio Source A to Audio Receiver B. The left channel is directly routed from A to B, whereas the right channel is routed through 3 Flexilink switches. The Test Case 2 is to route audio signal through multiple Flexilink switches as well as having background traffics ranging from 10% to 90%. The same topology is applied to Dante networking devices such as RedNet-D16 switches that are essentially the same technology as defined in AES67. The background traffic (sent as best-effort) is sent as asynchronous flow (AF-flow) in Flexilink is generated using "Ostinato", a free software tool used for traffic generation. UDP flows are generated between sender and receiver using maximum size Ethernet packets of 1518 bytes. Each experiment is repeated 10 times for 2 minutes each.

5.3.3 Testing Results

Figure 5.21 shows the results of *Test Case 1*. The end-to-end delay of audio traffic over Flexilink network using Gigabit Ethernet is shown in Figure 5.21a. It shows average delay of 767.517µsec over 3-hop Flexilink network. Figure 5.21b shows the average jitter of 1.188ns over Ethernet that is below audio playback threshold 20ns [142]. Transmitting audio signal over optical fibre connection is also tested. The average delay is 817.479µsec and jitter of 1.567ns respectively.

Figure 5.22 shows the results of *Test Case 2 Stress Test*. This experiment is to evaluate



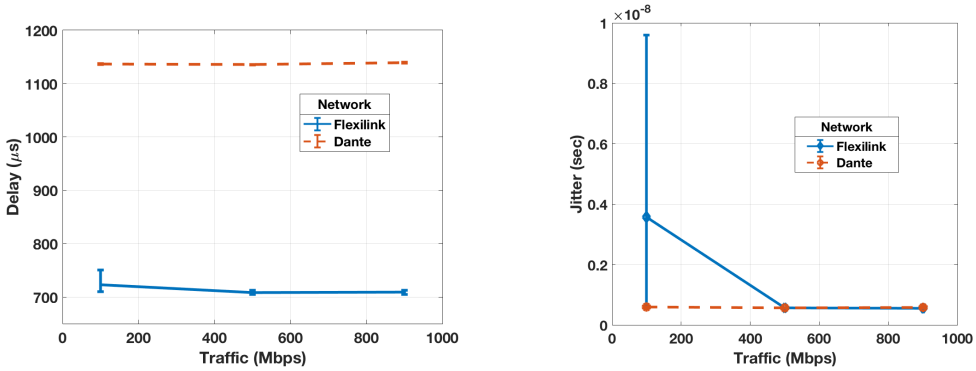
(a) End-to-End Latency

(b) Jitter

Figure 5.21: Test Case 1 Flexilink over Gigabit Ethernet

the performance of real-time audio traffic over Flexilink under stressed network conditions. In order to achieve this, the amount of background traffic (AF) is increased to increase the overall network load to see whether this will affect the performance of the real-time audio (SF) traffic. The AF traffic used 10%, 50% and 90% of the available bandwidth i.e. at 100 Mbps, 500 Mbps and 900 Mbps.

Figure 5.22a shows the delay experienced by the real-time audio traffic against the increasing background traffic load. It can be shown that Flexilink provides a stable performance to the real-time audio traffic regardless of the amount of background traffic being sent on the same link. Same tests had been carried out on AoIP network (Dante). Dante network also achieve stable performance under stress of best efforts data with higher latency values. It is worth noting that, at very high background traffic condition. Dante is discarding all packets other than audio packets to maintain the performance of audio networking, whereas Flexilink is able to transmit background traffic with high throughput as well as audio signals. This indicates that Flexilink has better mechanism of dealing with converged data. Figure 5.22b shows the statistically the jitter performance of Flexilink and Dante. There is no significant different between these two technologies. In general, Flexilink achieved low jitter performance no matter of network load.



(a) Flexilink vs Dante End-to-End Latency

(b) Flexilink vs Dante Jitter

Figure 5.22: Test Case 2 Stress Test with comparison

5.4 Conclusion

5.4.1 Summary

The prominent latency problem in audio networking is the unexpected network jitter and uneven packet delivery. Therefore the various buffers are used to mitigate the problem with the cost of increased latency. Also, the latency cannot be predicted accurately. Using carefully managed or isolated audio network can provide good delay performance. However, it lacks the mechanism to support future versatile audio delivery requirements, such as multiple real-time streams mixed with other types of traffic. Therefore the key solution is to provide time deterministic transmission mechanism that is able to support different real-time and non-real-time streams effectively.

In this chapter, we proposed a new time deterministic audio networking protocol and architecture: Flexilink, that combines the merits of both TDM and packet switched network. It is designed to support both real-time traffic (synchronous flow) and best effort traffic (asynchronous flow), with a capability of utilisation of full network bandwidth. Flexilink is a promising approach as an efficient model of the converged network technology in the professional media industry, without complex QoS management.

We presented the simulation model and results of Flexilink. It shows that although it

may introduce minor latency due to slot scheduling. There are advantages in some of the key multimedia quality measures such as the end-to-end delay, jitter and packet loss rate. In particular, Flexilink is the only architecture examined whose jitter is below the audible threshold in the case of streaming live digital audio signals. Flexilink has been prototyped as a hardware switch with multiple network and digital audio interfaces. The hardware test also shows it performs better than current professional audio networking devices in terms of latency and jitter.

Most relevant researches in networked music performance focus on prediction and management of network latency which treats underline network latency as some level of uncontrollable parameters with statistical profile [8, 9, 10]. Whereas our research provide a bottom up approach to provide deterministic network latency performances. If it is adopted in network infrastructure, it will be a great help for higher level applications and researches.

The Flexilink project is trying to re-introduce some nice time deterministic features from TDM and virtual circuit switching network to the new converged media network. The design goal is to ensure the timing and quality of audio signal delivery to service the up-layer applications. It is a hybrid approach. Many aspects of this proposed architecture are still under research. There are trade-off and pros and cons comparing IP based solution. The Table 5.6 gives brief summaries of these three main design approaches.

5.4.2 Future Work

The architecture of Flexilink is not only for audio applications, but potentially for all real time media content with isochronous pattern. Future work should look into how Flexilink could support broadcasting network content that has audio, video, control message and meta data, as well as to support synchronised audio/video delivery and low latency interactive media distribution networks.

Efficient scheduling and routing mechanism are key requirements to guarantee a professional performance in the wide area networks. We should also investigate how the slot reservation based mechanism can be developed and optimised over multiple hops and routes.

Table 5.6: Compare different networking infrastructure designs for audio

Technology	**TDM & Virtual circuit switching based**	**IP/Ethernet & Packet switching based**	**Flexilink - Hybrid mode (TDM + Packet)**
Delivery mode	Deterministic	Best effort	Deterministic for SF Best effort for AF
Quality of Service mechanism	Simple: Reserving virtual circuit signal path, no QoS required.	Complex: Priority and traffic engineering-based Layer 2&3 solutions: IntServ, DiffServ, IEEE 802.1p, IEEE 802.1Q.	Simple: Two tier system, SF using point to point path reservation; AF transmission on the background.
Latency	Very low: speed of light in the physical media.	Variable: unpredictable when network is congested.	Low: predictable based on de-jitter buffer size of SF scheduling mapping.
Supporting different traffic types	Low efficiency, low bandwidth utilisation, such as IP over TDM/ISDN.	Hight efficiency and high bandwidth utilisation: everything over IP.	High efficiency to support different types of traffics, high bandwidth utilisation.
Compatibility	Becoming obsolete for data network.	Becoming predominate network infrastructure.	Can be connected to IP network but require customised hardware and software.
Resilience	Low, failure signal path results in loss of the communication.	High, can automatically establish the routing path.	Redundancy and backup communication methods are not implemented yet.
Cost for traffic	High	Low	Can be made low if there are enough Flexilink networks.
Typical protocols for audio	ISDN AES50 ATM/AES47	AES67 Dante CobraNet	Flexilink

The Flexilink network architecture also provides an ideal solution for a QoS guaranteed end-to-end ‘Integrated Service’, although more development is required on the scalability and interoperability with current rapidly evolving networks.

Chapter 6

Conclusions

6.1 Summary of Main Contributions

Analogue electronic audio transmission drastically increases the speed of dissemination of the sound. However, in the digital audio era, the delay caused by AD/DA conversion, multichannel and complex processing, and network transmission draws the attention of audio engineers and system designers.

In this thesis, we investigated the latency problem in the real-time audio processing chain. We have attempted to answer the research questions that were raised at the beginning of this work.

1. What are the sources of delay in the digital audio processing chain?
2. Can we accurately estimate the delay when designing a digital audio system?
3. Can we adopt deterministic approach in the system design to minimise the delay in the digital audio system?

For the first question, we understand that the latency in audio processing is mainly caused by three factors: *Group Delay*, *Buffering*, and *Physical Propagation Delay*. This leads to the second question of delay estimation. For the delay caused by buffering, there are many reasons why a large buffering is needed. In this research, we focus on how

to remove the uncertainty of audio data processing and transmission. Then we can accurately estimate the buffer requirements hence the precise values of latency. For the group delay caused by digital filters, the most prominent problem is in the multistage multirate design for High-resolution Anti-aliasing Anti-image Filter (HAAF), a series mathematical formulas are derived to accurately calculate the overall group delay from the design parameters.

For the question 3, We focused on how to minimise Group Delay and Buffering Delay with a deterministic system design approach. We treat digital audio systems as multi-nodes signal flow graphs. The latency of the whole system is the critical path of the graph in terms of time. There are three main functional blocks in the critical path: ADC/DAC, Audio processing using DAW and General Purpose Operating System (GPOS), and Audio Networking. Group Delay is the main contribution of latency of $\Delta\Sigma$ ADC/DAC. Buffering delay is the main cause of GPOS delay and audio networking. The formulas of group delay estimation provide an optimisation approach to design the low delay filters for HAAF. We proposed and designed a new deterministic network system architecture and OS scheduling framework to enable minimised buffering design and to provide deterministic latency behaviour. The main contributions of this work are described in the following sections.

6.1.1 Group Delay Caused by Multistage Filters in Audio Conversion

In this area, we investigated the latency caused by digital decimation and interpolation filters in $\Delta\Sigma$ ADC/DAC. We name this type of filter High-resolution Anti-aliasing Anti-image Filter (HAAF) which has a large number of orders and multistage multirate structure. The group delay caused by this type of filter has not been fully studied before this research. We investigated the delay problem caused by HAAF with below research outputs:

1. Engineering brief of delay measurement of audio ADC/DAC [26].
2. Time Domain Performance of Decimation Filter Architectures for High Resolution Sigma Delta Analogue to Digital Conversion [27].

3. Practical considerations on optimising multistage decimation and interpolation processes [28].

In addition, there are two unpublished findings presented in this report.

4. The quantitative analysis of delay of High-resolution Anti-aliasing Anti-image Filter with overall trade-off design. (To be submitted to a journal publication)

5. Quantitative and Qualitative evaluation of minimum phase filter for High-resolution Anti-aliasing Anti-image Filter.

Paper 1 as in Chapter 3 Section 3.1, measured the latency of popular $\Delta\Sigma$ ADC/DAC used in most audio devices using a high speed multichannel data acquisition device. The test revealed latency problem of $\Delta\Sigma$ ADC/DAC caused by internal group delay of high-performance digital filters. There is lack of accurate reporting mechanism and standard way of describing delay of the hardware codecs. Paper 2 as in Chapter 3 Section 3.2 carried out comprehensive test and analysis on time domain performances of different types of multistage and multirate filters that can be used in high resolution ADC/DAC. Being the first time comprehensive review of time domain performance of different types of filters in this area, the work demonstrated some impacts and had been cited more than ten times by conference and journal papers till the time this thesis is written.

Paper 3 is the first part of Chapter 3 Section 3.3. This work simplified the traditional methods of searching optimal design of HAAF in terms of computational and area cost. This provides the good foundation of the second parts of the Chapter 3 Section 3.3 which is intended to be unpublished work item 4. This work considered an overall balanced design that taking group delay into account. It also formalised the new mathematical model of group delay formula that is used for analytically showing how different design parameters affect the overall delay. It provided a new HAAF design approach that considers both hardware cost and group delay.

Research work item 5 in Chapter 3 Section 3.4.1, created a set of new delay estimation formulas that can be used to quantitatively measure non-linear group delay, especially for minimum phase filters that have lowest average delay within the passband. The qualitative measure can be further developed by conducting listening tests in the future.

6.1.2 Delay of OS Processing and New Scheduling Framework

In this area, we measured the current latency values of using computer systems to process audio, especially with a large number of concurrent channels, with intelligent audio processing features, and with high CPU load. The relevant outputs are below:

1. Audio Latency Measurement for Desktop Operating Systems with Onboard Soundcards [12].
2. TDCS - A new scheduling framework for real-time Multimedia OS (ready to submit).

There are other research works the author contributed and co-authored [31, 32, 33, 34]. These work helped reinforce the findings, understand the general problem of audio processing software architecture, and inspire the solutions. Paper 1 [12] as in Chapter 4 Section 4.1 conducted the comprehensive audio processing latency test on GPOS and DAW. It revealed the problem of using GPOS to process low latency audio, especially when CPU is heavily loaded. And the current DAW lacks support and timing assurance for adaptive and cross-adaptive audio effects. This work has been widely cited more than 14 times until now. It shows the prevalence of the problem. Especially the intelligent audio processing paradigm is moving towards not only on GPOS but also within the Web platform [33] [34]. The underlying non-deterministic timing issues become the major obstacles of low latency audio processing tasks.

The research output item 2 as in Chapter 4 Section 4.2 proposed a new time deterministic scheduling framework. The simulation results showed good performance in comparing with classic scheduling algorithms that are commonly used in the real-time embedded system. Integrating this new scheduling framework into GPOS could be promising for solving the low latency audio processing problem. This proposed direction has already been awarded funding from a provisional key lab in China that will continue support this work further.

6.1.3 New Audio Networking Architecture

In this area, the tendency requires transmitting multiple channels of different source rates audio signals on a single communication channel. There is also increasing demand of transmitting different types of real-time multimedia data along with non-real time best effort data. We proposed a new network architecture and protocol to provide the means of carrying audio in low latency over existing networking infrastructure with network traffic convergence.

1. Flexilink: A unified low latency network architecture for multichannel live audio [29].
2. Performance Evaluation of a New Flexible Time Division Multiplexing Protocol on Mixed Traffic Types [30].
3. Towards true convergence, the architecture and performance evaluation of dynamic TDM system: Flexilink (paper drafted).

Paper 1 as in Chapter 5 Section 5.1 proposed the conceptual design of the Flexilink. Paper 2 as in Chapter 5 Section 5.2 presented the simulated performance of Flexilink in comparison with the traditional priority-based systems. It shows the deterministic characteristics of Flexilink with superior low jitter and latency performance. The planned paper 3 will provide the overarching overview of the Flexilink technology that developed so far including the performance test of prototyped hardware. The initial performance test of the hardware prototyping of Flexilink is presented in Chapter 5 Section 5.3.

6.2 Critical Assessment of Work

This research breaks down the latency issues of the whole audio processing chain into three main functional blocks and three main latency factors. The audio processing chain is continuously evolving to be the more complex system. There are many other aspects of the system can be investigated in-depth to improve the responsiveness of the system. This research looked into the overall picture of three main functional components. However, this study covers a little bit too wide areas, any one of these three aspects has lots

of open questions that can be further researched on.

For group delay of the digital filter for ADC/DAC, the significant saving of the delay could be using minimum phase filter in multistage structure. However, the minimum phase structure and its suitability of being used in $\Delta\Sigma$ ADC/DAC are not fully understood. The group delay curve of Minimum Phase (MP) filters tilt towards the 20k edges. Especially, it is an interesting question that whether this property would affect people's perception of music. The objective minimum phase measurements proposed in this study can further correlate with future perception tests to give qualitative analysis results.

For the new OS scheduling framework Time Deterministic Cyclic Scheduling (TDCS), the simulation showed that the new scheduling algorithm is able to provide stable performance when CPU load is 100% full. However, the work towards the combination of RT scheduling into GPOS needs system implementation to validate the results. The software engineering work of creating appropriate API interfaces and programming practices for the low latency applications to access this new scheduling scheme might not be a trivial work.

For the new audio networking architecture Flexilink, the current trend is using AoIP such as AES67 for professional live audio transmission. This approach has been proven to have a good level of latency performance. The critics of our proposed technology are that the level of improvement and scale of the jitter performance of Flexilink might not be necessary. However, for the future 5G network, the survey mentioned about the requirements of control plane latency under 10ms and user plane latency under 0.5ms [189]. The concepts and design of our proposal might be able to help in terms of the future carrier network architecture.

6.3 Future Work

The criticisms mentioned in the last section sheds light on the future work.

For *HAAF in $\Delta\Sigma$ ADC/DAC*, there are a few directions that can be further explored:

Design and conduct a listening test in a controlled environment for comparison of lin-

ear phase and minimum phase multistage filter in $\Delta\Sigma$ ADC/DAC. Because the filter curves of them have differences mainly around 20kHz corner frequency, the perception differences can be very subtle [100, 190, 101]. The recent research only shows the distinguishable differences all around 14KHz between minimum phase and linear phase filter. The proper design of testing method needs to be considered and any other filtering components that affect the results in the audio processing chain needs to be eliminated.

The efficient hardware implementation of HAAF. The HAAF eventually will be implemented in hardware form. There are theoretical and practical questions to be answered: To what extent the extra number of stages and associated clock controls will offset the cost saved by the multistage structure and what is a balanced design? How to effectively using a fixed-point parallel structure of multiplier and adders to implement preferred filters? Moreover, what are the pros and cons of using minimum phase or IIR filter in the hardware form and how to avoid pitfalls of using them?

For the *new OS scheduling scheme TDCS*, it will be interesting to develop a general theory of TDCS hyper-period conversion algorithm, and its objective function in terms allocated flow jitter and accumulated delay. Then it would be good to see if this can be converted into an optimisation problem concerning different timing properties measures. The general theory can also help quantifying the QoS in a practical case when the hyper-period is decided by system factor rather than optimal theoretical values. Furthermore, the task mapping scheme can be elaborated to have different types of mapping such as 'as even as possible' or based on other static scheduling policies. Essentially to have the real impact, to implement TDCS in an OS such as in Linux is desired.

For the *new network architecture Flexilink*, it is interesting to study how the reserve based SF flow can be expanded into the multi-hop network. There are concerns about the scalability of the reserve based protocols. However, today there are limited nodes in the backbone network. Flexilink might be more suitable for the carrier network, then to work out the inter-domain communication protocol that can preserve the dynamic TDM features of this architecture. The current SDN standards are not well defined for time constraints. Collective efforts from different groups need to be made to work together such as IEEE time sensitive networking group, IETF Deterministic Networking group, AES audio networking standardisation group, and SMPTE standardisation group. Some

core concepts of Flexilink can be applied within the standardisation framework. To have real-world impact, a software driver stack can be developed to enable common operating systems to operate their network interface in Flexilink mode.

In fact, this work tried to tackle similar problems to those that current Cyber-physical system (CPS) research does [191]. A typical example of CPS is autonomous vehicle that needs to response the multiple sensor signals in a very short time window. Live audio system can be regarded as a particular case of real-time CPS system that involves communication, computing, and physical interactions as the constraints (music interaction). It will be interesting to use formal CPS system design method to consider the design and implement the live audio system and extend the concept to the broader interactive digital media system.

Fundamentally, as professor Edward A. Lee mentioned in "Cyber-Physical Systems: Design Challenges" [23], maybe because "Turing completeness" does not dictate timing assurance, the "timing" is semantically lost in the different abstraction layers of computing. This causes loss of predictability and reliability at the application layer, even though the underlying 'digital hardware (layer) delivers astonishingly precise timing behaviour'. It needs to be tackled from a bottom-up approach. Our work has made some efforts and advancement in this way, and it could be the continuous direction in the future.

Bibliography

[1] L. Hoddeson. “The Emergence of Basic Research in the Bell Telephone System, 1875-1915”. In: *Technology and Culture* 22.3 (1981), pp. 512–544. JSTOR: 3104388.

[2] *First Transcontinental Telephone Call*. In: *Wikipedia*. Page Version ID: 742125793. Oct. 1, 2016.

[3] G. S. K. Wong. “Speed of Sound in Standard Air”. In: *The Journal of the Acoustical Society of America* 79.5 (1986), pp. 1359–1366.

[4] J. A. Moorer. “Audio in the New Millennium”. In: *J. Audio Eng. Soc* 48.5 (2000), pp. 490, 492, 494–498.

[5] A. Tanaka, L. Gaye, and R. Richardson. “Co-Production and Co-Creation: Creative Practice in Social Inclusion”. In: *Cultural Computing*. Ed. by R. Nakatsu et al. Vol. 333. IFIP Advances in Information and Communication Technology. 10.1007/978-3-642-15214-6_17. Springer Boston, 2010, pp. 169–178.

[6] N. Kitawaki and K. Itoh. “Pure Delay Effects on Speech Quality in Telecommunications”. In: *IEEE Journal on Selected Areas in Communications* 9.4 (May 1991), pp. 586–593.

[7] C. Chafe, J. Cáceres, and M. Gurevich. “Effect of Temporal Separation on Synchronization in Rhythmic Performance”. In: *Perception* 39.7 (2010), pp. 982–992.

[8] C. Chafe and M. Gurevich. “Network Time Delay and Ensemble Accuracy: Effects of Latency, Asymmetry”. In: *Audio Engineering Society Convention 117*. Oct. 2004.

[9] B. Vera and E. Chew. “Towards Seamless Network Music Performance: Predicting an Ensemble’s Expressive Decisions for Distributed Performance.” In: *ISMIR*. 2014, pp. 489–494.

[10] C. Rottondi et al. “Feature-Based Analysis of the Effects of Packet Delay on Networked Musical Interactions”. In: *Journal of the Audio Engineering Society* 63.11 (2015), pp. 864–875.

[11] A. Dasdan, D. Ramanathan, and R. Gupta. “A Timing-Driven Design and Validation Methodology for Embedded Real-Time Systems”. In: *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 3.4 (1998), pp. 533–553.

[12] Y. Wang, R. Stables, and J. Reiss. “Audio Latency Measurement for Desktop Operating Systems with Onboard Soundcards”. In: *Audio Engineering Society Convention 128*. May 2010.

[13] M. Wright, R. Cassidy, and M. Zbyszynski. “Audio and Gesture Latency Measurements on Linux and OSX”. In: *Proceedings of the ICMC*. 2004, pp. 423–429.

[14] M. Feerick. “How Much Delay Is Too Much Delay ?” In: *Audio Engineering Society Convention 112*. Apr. 2002.

[15] N. Bouillot et al. “AES White Paper: Best Practices in Network Audio”. In: *Journal of Audio Engineering Society* 57.9 (2009).

[16] M. Poimboeuf. “Audio Subsystem Design for Workstations and Personal Computers”. In: *Audio Engineering Society Conference: UK 17th Conference: Audio Delivery*. Apr. 2002.

[17] J. S. A. Eyebe Fouda et al. “Toward a Group Delay Reduction in Digital Filtering”. In: *Digital Signal Processing* 19.1 (Jan. 2009), pp. 22–32.

[18] J. W. Topliss, V. Zappi, and A. McPHERSON. “Latency Performance for Real-Time Audio on BeagleBone Black”. In: (2014).

[19] A. P. McPherson, R. H. Jack, and G. Moro. “Action-Sound Latency: Are Our Tools Fast Enough?” In: (2016).

[20] N. Juillerat, S. Schubiger-Banz, and S. Arisona. “Low Latency Audio Pitch Shifting in the Time Domain”. In: *Audio, Language and Image Processing, 2008. ICALIP 2008. International Conference On*. July 2008, pp. 29–35.

[21] N. Juillerat and B. Hirsbrunner. “Low Latency Audio Pitch Shifting in the Frequency Domain”. In: *Audio Language and Image Processing (ICALIP), 2010 International Conference On*. Nov. 2010, pp. 16–24.

[22] W. Gardner. “Efficient Convolution without Input-Output Delay”. In: *Journal of the Audio Engineering Society* 43.3 (1995), pp. 127–136.

[23] E. A. Lee. “Cyber Physical Systems: Design Challenges”. In: ISORC ’08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 363–369.

[24] J.-P. Lambert, S. Robaszkiewicz, and N. Schnell. “Synchronisation for Distributed Audio Rendering over Heterogeneous Devices”. In: *Html5. In WAC Web Audio Conference. Georgia Institute of Technology*. 2016.

[25] S. J. Mason. “Feedback Theory: Further Properties of Signal Flow Graphs”. In: (1956).

[26] Y. Wang. “Latency Measurements of Audio Sigma Delta Analogue to Digital and Digital to Analogue Converts”. In: *131st AES Convention*. 131st AES Convention. New York, NY, USA, 2011.

[27] Y. Wang and J. D. Reiss. “Time Domain Performance of Decimation Filter Architectures for High Resolution Sigma Delta Analogue to Digital Conversion”. In: *Audio Engineering Society Convention 132*. Apr. 2012.

[28] X. Zhu et al. “Practical Considerations on Optimising Multistage Decimation and Interpolation Processes”. In: *Digital Signal Processing (DSP), 2016 IEEE International Conference On*. IEEE, 2016, pp. 370–374.

[29] Y. Wang, J. Grant, and J. Foss. “Flexilink: A Unified Low Latency Network Architecture for Multichannel Live Audio”. In: *133th Audio Engineering Society Convention*. 2012.

[30] Y. Song et al. “Performance Evaluation of a New Flexible Time Division Multiplexing Protocol on Mixed Traffic Types”. In: *Advanced Information Networking and Applications (AINA), 2017 IEEE 31st International Conference On*. IEEE, 2017, pp. 23–30.

[31] Y. Wang, X. Zhu, and Q. Fu. “A Low Latency Multichannel Audio Processing Evaluation Platform”. In: Audio Engineering Society Convention 132. Audio Engineering Society, Apr. 26, 2012.

[32] N. Jillings and Y. Wang. “CUDA Accelerated Audio Digital Signal Processing for Real-Time Algorithms”. In: *Audio Engineering Society Convention 137*. Oct. 2014.

[33] N. Jillings et al. “JSAP: A Plugin Standard for the Web Audio API with Intelligent Functionality”. In: *Audio Engineering Society Convention 141*. Audio Engineering Society, 2016.

[34] N. Jillings et al. “Intelligent Audio Plugin Framework for the Web Audio API”. In: (2017).

[35] T. Mäki-Patola and P. Hämäläinen. “Latency Tolerance for Gesture Controlled Continuous Sound Instrument without Tactile Feedback.” In: *ICMC*. 2004.

[36] M. Lester and J. Boley. “The Effects of Latency on Live Sound Monitoring”. In: *Audio Engineering Society Convention 123*. Oct. 2007.

[37] S. Farner et al. “Ensemble Hand-Clapping Experiments under the Influence of Delay and Various Acoustic Enviroments”. In: *Audio Engineering Society Convention 121*. Oct. 2006.

[38] R. H. Jack, T. Stockman, and A. McPherson. “Effect of Latency on Performer Interaction and Subjective Quality Assessment of a Digital Musical Instrument”. In: *Proceedings of the Audio Mostly 2016*. AM ’16. New York, NY, USA: ACM, 2016, pp. 116–123.

[39] D. Wessel and M. Wright. “Problems and Prospects for Intimate Musical Control of Computers”. In: *Computer Music Journal* 26.3 (Sept. 1, 2002), pp. 11–22.

[40] E. Perez-Gonzalez and J. D. Reiss. “Determination and Correction of Individual Channel Time Offsets for Signals Involved in an Audio Mixture”. In: *Audio Engineering Society Convention 125*. Oct. 2008.

[41] Audio Engineering Society. “AES50-2011 AES Standard for Digital Audio Engineering - High-Resolution Multi-Channel Audio Interconnection (HRMAI)”. In: *AES STANDARDS* (2011).

[42] G. Garner et al. “Timing and Synchronization for Audio/Video Applications in a Converged Residential Ethernet Network”. In: *3rd IEEE Consumer Communications and Networking Conference, 2006. CCNC 2006*. 3rd IEEE Consumer Communications and Networking Conference, 2006. CCNC 2006. Vol. 2. Jan. 2006, pp. 883–887.

[43] S. Schmitt and D. J. Cronemeyer. *Audio over Ethernet:There Are Many Solutions –but Which One Is Best for You?* URL: `http://www.dspecialists.com/sites/default/files/publication/110126a_networking_paperembeddedworld2011_paperen_final_st_jc.pdf` (visited on 01/22/2013).

[44] Audio Engineering Society. "AES67-2015, AES Standard for Audio Applications of Networks High-Performance Streaming Audio-over-IP Interoperability". In: *AES STANDARDS* (2015).

[45] J. P. Cáceres and C. Chafe. "JackTrip: Under the Hood of an Engine for Network Audio". In: *Journal of New Music Research* 39.3 (2010), pp. 183–187.

[46] W. Bolton, Y. Xiao, and M. Guizani. "IEEE 802.20: Mobile Broadband Wireless Access". In: *Wireless Communications, IEEE* 14.1 (Feb. 2007), pp. 84–95.

[47] N. Lago and F. Kon. "The Quest for Low Latency". In: *Proceedings of the International Computer Music Conference*. 2004, pp. 33–36.

[48] N. Jillings, A. Clifford, and J. D. Reiss. "Performance Optimization of GCC-PHAT for Delay and Polarity Correction under Real World Conditions". In: *Audio Engineering Society Convention 134*. May 2013.

[49] A. Clifford and J. Reiss. "Calculating Time Delays of Multiple Active Sources in Live Sound". In: *Audio Engineering Society Convention 129*. Nov. 2010.

[50] K. Williston and J. Bier. "Trends and Directions in Signal-Processing Hardware for Audio Applications". In: *AES Conference: 23rd International Conference: Signal Processing in Audio Recording and Reproduction*. 2003.

[51] S.-T. Dietrich and D. Walker. "The Evolution of Real-Time Linux". In: *7th RTL Workshop*. 2005.

[52] E. Brandt and R. Dannenberg. "Low-Latency Music Software Using off-the-Shelf Operating Systems". In: *Proc. 1998 Intl. Computer Music Conf.(ICMC-98)*. 1998, pp. 137–141.

[53] K. MacMillan, M. Droettboom, and I. Fujinaga. "Audio Latency Measurements of Desktop Operating Systems". In: *Proceedings of the International Computer Music Conference*. 2001, pp. 259–262.

[54] V. Verfaille, U. Zolzer, and D. Arfib. "Adaptive Digital Audio Effects (a-DAFx): A New Class of Sound Transformations". In: *Audio, Speech, and Language Processing, IEEE Transactions on* 14.5 (Sept. 2006), pp. 1817–1831.

[55] J. D. Reiss. “Intelligent Systems for Mixing Multichannel Audio”. In: 2011 17th International Conference on Digital Signal Processing (DSP). July 2011, pp. 1–6.

[56] U. Zölzer and X. Amatriain. *DAFX: Digital Audio Effects*. John Wiley and Sons Inc, 2002.

[57] J. O. Smith. *Introduction to Digital Filters: With Audio Applications*. Vol. 2. Booksurge Llc, 2006.

[58] I. Selesnick and C. Burrus. “Maximally Flat Low-Pass FIR Filters with Reduced Delay”. In: *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on* 45.1 (Jan. 1998), pp. 53–68.

[59] G. Apaydin. “Realization of Reduced-Delay Finite Impulse Response Filters for Audio Applications”. In: *Digital Signal Processing* 20.3 (May 2010), pp. 620–629.

[60] C. Wu, D. Gao, and K. Lay Teo. “A Direct Optimization Method for Low Group Delay FIR Filter Design”. In: *Signal Processing* 93.7 (July 2013), pp. 1764–1772.

[61] E. Perez-Gonzalez and J. D. Reiss. “Automatic Mixing”. In: *Digital Audio Effects*. John Wiley and Sons, 2011.

[62] E. Perez-Gonzalez and J. D. Reiss. “Improved Control for Selective Minimization of Masking Using Interchannel Dependancy Effects”. In: *Proc. of the 11th Int. Conference on Digital Audio Effects (DAFx-08), Espoo, Finland*. 2008.

[63] E. Perez-Gonzalez and J. D. Reiss. “Automatic Equalization of Multichannel Audio Using Cross-Adaptive Methods”. In: *Audio Engineering Society Convention 127*. Oct. 2009.

[64] E. Perez-Gonzalez and J. D. Reiss. “Automatic Gain and Fader Control for Live Mixing”. In: *Applications of Signal Processing to Audio and Acoustics, 2009. WASPAA ’09. IEEE Workshop On*. Oct. 2009, pp. 1–4.

[65] E. Perez-Gonzalez and J. D. Reiss. “A Real-Time Semiautonomous Audio Panning System for Music Mixing”. In: *EURASIP J. Adv. Signal Process* 2010 (Feb. 2010), 5:1–5:10.

[66] P. Brossier. “Automatic Annotation of Musical Audio for Interactive Applications”. Citeseer, 2007.

[67] A. M. Stark, M. D. Plumbley, and M. E. P. Davies. “Real-Time Beat-Synchronous Audio Effects”. In: *Proceedings of the 7th International Conference on New Interfaces for Musical Expression*. NIME ’07. New York, NY, USA: ACM, 2007, pp. 344–345.

[68] G. Bocko et al. “Automatic Music Production System Employing Probabilistic Expert Systems”. In: *Audio Engineering Society Convention 129*. Nov. 2010.

[69] P. Aziz and H. Sorensen. “An Overview of Sigma-Delta Converters”. In: *Signal Processing Magazine, IEEE* 13.1 (1996), pp. 61–84.

[70] J. D. Reiss. “Understanding Sigma-Delta Modulation: The Solved and Unsolved Issues”. In: *Journal of the Audio Engineering Society* 56 (1-2 2008), pp. 49–64.

[71] J. Vanderkooy and S. Lipshitz. “Why 1-Bit Sigma-Delta Conversion Is Unsuitable for High-Quality Applications”. In: *AES 110th Convention*. Vol. 1. 2001.

[72] J. D. Reiss and M. B. Sandler. “Dither and Noise Modulation in Sigma Delta Modulators”. In: AES 115th Convention. 2003.

[73] S. Park and W. Chen. “Multi-Stage IIR Decimation Filter Design Technique for High Resolution Sigma-Delta A/D Converters”. In: *Instrumentation and Measurement Technology Conference, 1992. IMTC ’92., 9th IEEE*. May 1992, pp. 561–566.

[74] P. Lesso and A. J. Magrath. “An Ultra High Performance DAC with Controlled Time-Domain Response”. In: *Audio Engineering Society Convention 119*. Oct. 2005.

[75] L. Tan. *Digital Signal Processing: Fundamentals and Applications*. Academic Pr, 2008.

[76] K. Rajamani, Y.-S. Lai, and C. Furrow. “An Efficient Algorithm for Sample Rate Conversion from CD to DAT”. In: *Signal Processing Letters, IEEE* 7.10 (Oct. 2000), pp. 288–290.

[77] P. Beckmann and T. Stilson. “An Efficient Asynchronous Sampling-Rate Conversion Algorithm for Multi-Channel Audio Applications”. In: *Audio Engineering Society Convention 119*. Audio Engineering Society, 2005.

[78] T. Heeb. “High Performance Real-Time Software Asynchronous Sample Rate Converter Kernel”. In: *Audio Engineering Society Convention 120*. Audio Engineering Society, 2006.

[79] P. Midya, B. Roeckner, and T. Schooler. “Asynchronous Sample Rate Converter for Digital Audio Amplifiers”. In: *Audio Engineering Society Convention 121*. Audio Engineering Society, 2006.

[80] D. Nowak. “Linear Phase Digital Filter Design”. In: *Proceedings of the IEEE* 57.5 (1969), pp. 850–851.

[81] A. Deczky. “General Expression for the Group Delay of Digital Filters”. In: *Electronics Letters* 5.25 (1969), pp. 663–665.

[82] J. F. Kaiser. “Nonrecursive Digital Filter Design Using the I_0-Sinh Window Function”. In: *Proc. IEEE Int. Symp. Circuits & Syst., 1974*. 1974.

[83] L. Rabiner. “Approximate Design Relationships for Low-Pass FIR Digital Filters”. In: *IEEE Transactions on Audio and Electroacoustics* 21.5 (1973), pp. 456–460.

[84] R. Crochiere and L. Rabiner. “Optimum FIR Digital Filter Implementations for Decimation, Interpolation, and Narrow-Band Filtering”. In: *IEEE Transactions on Acoustics, Speech and Signal Processing* 23.5 (Oct. 1975), pp. 444–456.

[85] K. Ichige, M. Iwaki, and R. Ishii. “Accurate Estimation of Minimum Filter Length for Optimum FIR Digital Filters”. In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 47.10 (2000), pp. 1008–1016.

[86] A. V. Oppenheim and R. W. Schafer. *Digital Signal Processing*. Prentice-Hall, 1975. 616 pp.

[87] W. Kester. *Mixed-Signal and DSP Design Techniques*. Vol. 1. Newnes, 2003.

[88] R. Crochiere and L. Rabiner. “Interpolation and Decimation of Digital Signals - A Tutorial Review”. In: *Proceedings of the IEEE* 69.3 (1981), pp. 300–331.

[89] M. W. Coffey. “Optimizing Multistage Decimation and Interpolation Processing”. In: *IEEE Signal Processing Letters* 10.4 (2003), pp. 107–110.

[90] M. W. Coffey. “Optimizing Multistage Decimation and Interpolation Processing Mdash;Part II”. In: *IEEE Signal Processing Letters* 14.1 (2007), pp. 24–26.

[91] D. Shi and Y. J. Yu. “Design of Discrete-Valued Linear Phase FIR Filters in Cascade Form”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 58.7 (2011), pp. 1627–1636.

[92] E. Hogenauer. “An Economical Class of Digital Filters for Decimation and Interpolation”. In: *Acoustics, Speech and Signal Processing, IEEE Transactions on* 29.2 (Apr. 1981), pp. 155–162.

[93] M. Bellanger, J. Daguet, and G. Lepagnol. “Interpolation, Extrapolation, and Reduction of Computation Speed in Digital Filters”. In: *Acoustics, Speech and Signal Processing, IEEE Transactions on* 22.4 (Aug. 1974), pp. 231–235.

[94] M. Bellanger. “Computation Rate and Storage Estimation in Multirate Digital Filtering with Half-Band Filters”. In: *Acoustics, Speech and Signal Processing, IEEE Transactions on* 25.4 (Aug. 1977), pp. 344–346.

[95] F. Mintzer. “On Half-Band, Third-Band, and Nth-Band FIR Filters and Their Design”. In: *Acoustics, Speech and Signal Processing, IEEE Transactions on* 30.5 (Oct. 1982), pp. 734–738.

[96] M. Bellanger, G. Bonnerot, and M. Coudreuse. “Digital Filtering by Polyphase Network:Application to Sample-Rate Alteration and Filter Banks”. In: *Acoustics, Speech and Signal Processing, IEEE Transactions on* 24.2 (Apr. 1976), pp. 109–114.

[97] T. G. Foxall, A. A. Ibrahim, and G. J. Hupe. “Minimum-Phase CCD Transversal Filters”. In: *Solid-State Circuits, IEEE Journal of* 12.6 (1977), pp. 638–642.

[98] R. Boite and H. Leich. “A New Procedure for the Design of High Order Minimum Phase FIR Digital or CCD Filters”. In: *Signal Processing* 3.2 (Apr. 1981), pp. 101–108.

[99] S. R. Norsworthy and D. G. Shaw. *Data Converter with Minimum Phase Fir Filter and Method for Calculating Filter Coefficients*. U.S. Classification 341/126, 708/306; International Classification H03H21/00, H03H17/00, H04B3/23, H03H17/06; Cooperative Classification H03H17/06; European Classification H03H17/06. US Patent No. US5561424 A, Oct. 1, 1996.

[100] J. Blauert and P. Laws. “Group Delay Distortions in Electroacoustical Systems”. In: *Journal of the Acoustical Society of America* 63.5 (1978), pp. 1478–1483.

[101] P. Bloom and D. Preis. “Perceptual Identification and Discrimination of Phase Distortions”. In: *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP ’83*. Vol. 8. Apr. 1983, pp. 1396–1399.

[102] O. Herrmann and W. Schuessler. “Design of Nonrecursive Digital Filters with Minimum Phase”. In: *Electronics Letters* 6.11 (1970), pp. 329–330.

[103] N. Damera-Venkata, B. Evans, and S. McCaslin. “Design of Optimal Minimum-Phase Digital FIR Filters Using Discrete Hilbert Transforms”. In: *Signal Processing, IEEE Transactions on* 48.5 (May 2000), pp. 1491–1495.

[104] T. Stathaki and I. Fotinopoulos. “Equiripple Minimum Phase FIR Filter Design from Linear Phase Systems Using Root Moments”. In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 48.6 (June 2001), pp. 580–587.

[105] S.-C. Pei and H.-S. Lin. “Minimum-Phase FIR Filter Design Using Real Cepstrum”. In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 53.10 (2006), pp. 1113–1117.

[106] T. Stathaki. “Root Moments: A Digital Signal-Processing Perspective”. In: *Vision, Image and Signal Processing, IEE Proceedings* - 145.4 (Aug. 1998), pp. 293–302.

[107] C. Williams. “Linux Scheduler Latency”. In: *Red Hat Inc* 3 (2002).

[108] D. d Niz, K. Lakshmanan, and R. Rajkumar. “On the Scheduling of Mixed-Criticality Real-Time Task Sets”. In: 2009 30th IEEE Real-Time Systems Symposium. Dec. 2009, pp. 291–300.

[109] O. R. Kelly, H. Aydin, and B. Zhao. “On Partitioned Scheduling of Fixed-Priority Mixed-Criticality Task Sets”. In: 2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications. Nov. 2011, pp. 1051–1059.

[110] A. Burns and R. Davis. “Mixed Criticality Systems-a Review”. In: *Department of Computer Science, University of York, Tech. Rep* (2013).

[111] J. Stankovic. “Misconceptions about Real-Time Computing: A Serious Problem for next-Generation Systems”. In: *Computer* 21.10 (Oct. 1988), pp. 10–19.

[112] G. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Springer Science & Business Media, Sept. 15, 2011. 528 pp.

[113] P. Leroux. “RTOS vs. GPOS: What Is Best for Embedded Development”. In: *Embedded Computing Design* (2005).

[114] A. Molano, K. Juvva, and R. Rajkumar. “Real-Time Filesystems. Guaranteeing Timing Constraints for Disk Accesses in RT-Mach”. In: *Proceedings Real-Time Systems Symposium*. Proceedings Real-Time Systems Symposium. Dec. 1997, pp. 155–165.

[115] V. Yodaiken et al. “The Rtlinux Manifesto”. In: *Proc. of the 5th Linux Expo*. 1999.

[116] A. McPherson and V. Zappi. “An Environment for Submillisecond-Latency Audio and Sensor Processing on BeagleBone Black”. In: *Audio Engineering Society Convention 138*. Audio Engineering Society, 2015.

[117] G. Moro et al. “Making High-Performance Embedded Instruments with Bela and Pure Data”. In: http://www.liveinterfaces.org. 2016.

[118] B. Adelberg, H. Garcia-Molina, and B. Kao. “Emulating Soft Real-Time Scheduling Using Traditional Operating System Schedulers”. In: *1994 Proceedings Real-Time Systems Symposium*. 1994 Proceedings Real-Time Systems Symposium. Dec. 1994, pp. 292–298.

[119] A. Atlas and A. Bestavros. “Statistical Rate Monotonic Scheduling”. In: Proceedings 19th IEEE Real-Time Systems Symposium (Cat. No.98CB36279). Dec. 1998, pp. 123–132.

[120] A. Atlas and A. Bestavros. “Design and Implementation of Statistical Rate Monotonic Scheduling in KURT Linux”. In: Proceedings 20th IEEE Real-Time Systems Symposium (Cat. No.99CB37054). 1999, pp. 272–276.

[121] I. Kalkov et al. “A Real-Time Extension to the Android Platform”. In: *Proceedings of the 10th International Workshop on Java Technologies for Real-Time and Embedded Systems*. JTRES ’12. New York, NY, USA: ACM, 2012, pp. 105–114.

[122] T. P. Baker and A. Shaw. “The Cyclic Executive Model and Ada”. In: *Real-Time Systems* 1.1 (June 1, 1989), pp. 7–25.

[123] L. Sha et al. “Real Time Scheduling Theory: A Historical Perspective”. In: *Real-Time Syst.* 28 (2-3 Nov. 2004), pp. 101–155.

[124] C. L. Liu and J. W. Layland. “Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment”. In: *J. ACM* 20.1 (Jan. 1973), pp. 46–61.

[125] L. Sha, R. Rajkumar, and S. Sathaye. “Generalized Rate-Monotonic Scheduling Theory: A Framework for Developing Real-Time Systems”. In: *Proceedings of the IEEE* 82.1 (Jan. 1994), pp. 68–82.

[126] L. Sha and J. B. Goodenough. “Real-Time Scheduling Theory and Ada”. In: *Computer* 23.4 (Apr. 1990), pp. 53–62.

[127] S. Samolej. “ARINC Specification 653 Based Real-Time Software Engineering.” In: *e-Informatica* 5.1 (2011), pp. 39–49.

[128] L. Almeida et al. “Schedulability Analysis of Real-Time Traffic in WorldFIP Networks: An Integrated Approach”. In: *IEEE Transactions on Industrial Electronics* 49.5 (Oct. 2002), pp. 1165–1174.

[129] A. Hangan and G. Sebestyen. “Cyclic Executive-Based Method for Scheduling Hard Real-Time Transactions on Distributed Systems”. In: 2011 IEEE 7th International Conference on Intelligent Computer Communication and Processing. Aug. 2011, pp. 441–444.

[130] A. Scroxton. *ISDN’s Days Are Numbered: What Should You Do?* Nov. 2016. URL: `http://www.computerweekly.com/blog/The-Full-Spectrum/ISDNs-days-are-numbered-What-should-you-do` (visited on 11/28/2017).

[131] Orange Business Services. *Dealing with the Great ISDN Switch Off*. Oct. 28, 2015. URL: `https://www.orange-business.com/en/magazine/dealing-with-the-great-isdn-switch-off` (visited on 11/28/2017).

[132] A. Kovalick. “A Review of the Technology and Migration Patterns for IP/IT Media Infrastructures”. In: *SMPTE Motion Imaging Journal* 124.6 (2015), pp. 69–77.

[133] G. F. Shay. “How AES-67, the New Audio-over-IP Standard, Will Bring the Convergence of Telecommunications, Intercom, Radio and Television Broadcast Studio Audio”. In: *Persistence of Vision-Defining the Future, SMPTE15:* SMPTE, 2015, pp. 1–10.

[134] M. Felser. “Real-Time Ethernet - Industry Prospective”. In: *Proceedings of the IEEE* 93.6 (June 2005), pp. 1118–1129.

[135] J. Jasperneite et al. “A Proposal for a Generic Real-Time Ethernet System”. In: *IEEE Transactions on Industrial Informatics* 5.2 (May 2009), pp. 75–85.

[136] J.-D. Decotignie. “The Many Faces of Industrial Ethernet [Past and Present]”. In: *IEEE_M_IE* 3.1 (2009), pp. 8–19.

[137] G. Prytz. “A Performance Analysis of EtherCAT and PROFINET IRT”. In: *2008 IEEE International Conference on Emerging Technologies and Factory Automation*. 2008 IEEE International Conference on Emerging Technologies and Factory Automation. Sept. 2008, pp. 408–415.

[138] M. D. Johas Teener et al. “Heterogeneous Networks for Audio and Video: Using IEEE 802.1 Audio Video Bridging”. In: *Proceedings of the IEEE* 101.11 (Nov. 2013), pp. 2339–2354.

[139] J.-D. Decotignie. “Ethernet-Based Real-Time and Industrial Communications”. In: *Proceedings of the IEEE* 93.6 (June 2005), pp. 1102–1117.

[140] M. Popp and P. Wenzel. “PROFInet-Linking Worlds”. In: *ETFA 2001. 8th International Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No.01TH8597)*. ETFA 2001. 8th International Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No.01TH8597). Vol. 2. Oct. 2001, 519–522 vol.2.

[141] P. Ferrari et al. “Experimental Evaluation of PROFINET Performance”. In: *IEEE International Workshop on Factory Communication Systems, 2004. Proceedings*. IEEE International Workshop on Factory Communication Systems, 2004. Proceedings. Sept. 2004, pp. 331–334.

[142] E. Benjamin and B. Gannon. “Theoretical and Audible Effects of Jitter on Digital Audio Quality”. In: *105th Audio Engineering Society Convention*. 1998.

[143] H. Schulzrinne et al. “RTP: A Transport Protocol for Real-Time Applications”. In: *IETF RFC3550* (2003).

[144] P. Neumann. “Communication in Industrial Automation What Is Going On ?” In: *Control Engineering Practice* 15.11 (2007), pp. 1332–1347.

[145] M. A. El-Gendy, A. Bose, and K. G. Shin. “Evolution of the Internet QoS and Support for Soft Real-Time Applications”. In: *Proceedings of the IEEE* 91.7 (2003), pp. 1086–1104.

[146] D. N. Finn. “Internet-Draft P. Thubert Intended Status: Standards Track Cisco Expires: February 19, 2017 August 18, 2016”. In: (2016).

[147] K. Wolter, P. Reinecke, and A. Mittermaier. “Model-Based Evaluation and Improvement of PTP Syntonisation Accuracy in Packet-Switched Backhaul Networks for Mobile Applications”. In: *Proceedings of the 8th European Conference on Computer Performance Engineering*. EPEW’11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 219–234.

[148] O. Ronen and M. Lipinski. “Enhanced Synchronization Accuracy in IEEE1588”. In: *2015 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*. 2015 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS). Oct. 2015, pp. 76–81.

[149] R. Ma et al. “Optimum Design of Multistage Half-Band FIR Filter for Audio Conversion Using a Simulated Annealing Algorithm”. In: 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA). May 2018, pp. 74–78.

[150] Intel. *High Definition Audio Specification, Revision 1.0a*. June 2010.

[151] R. Schreier et al. *Understanding Delta-Sigma Data Converters*. IEEE press New Jersey, 2005.

[152] M. Toner and G. Roberts. “A BIST Scheme for a SNR, Gain Tracking, and Frequency Response Test of a Sigma-Delta ADC”. In: *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on* 42.1 (1995), pp. 1–15.

[153] S. R. Norsworthy, R. Schreier, and G. C. Temes. *Delta-Sigma Data Converters: Theory, Design, and Simulation*. Vol. 97. IEEE press New York, 1996.

[154] J. Candy. “Decimation for Sigma Delta Modulation”. In: *Communications, IEEE Transactions on* 34.1 (Jan. 1986), pp. 72–76.

[155] S. Park. “Multi-Stage Decimation Filter Design Technique for High-Resolution Sigma-Delta A/D Converters”. In: *IEEE Transactions on Instrumentation and Measurement* 41.6 (1992), pp. 868–873.

[156] R. Ansari and B. Liu. “Efficient Sampling Rate Alteration Using Recursive (IIR) Digital Filters”. In: *IEEE Transactions on Acoustics, Speech and Signal Processing* 31.6 (Dec. 1983), pp. 1366–1373.

[157] R. Ansari. “Elliptic Filter Design for a Class of Generalized Halfband Filters”. In: *Acoustics, Speech and Signal Processing, IEEE Transactions on* 33.5 (1985), pp. 1146–1150.

[158] P. Steffen et al. “Recursive Halfband-Filters”. In: *AEU-International Journal of Electronics and Communications* 55.6 (2001), pp. 377–388.

[159] F. Harris. *Multirate Signal Processing for Communication Systems*. Prentice Hall PTR, 2004.

[160] Y. Kamp and C. Wellekens. “Optimal Design of Minimum-Phase FIR Filters”. In: *Acoustics, Speech and Signal Processing, IEEE Transactions on* 31.4 (Aug. 1983), pp. 922–926.

[161] B. C. Garai, P. Das, and A. K. Mishra. “Group Delay Reduction in FIR Digital Filters”. In: *Signal Processing* 91.8 (2011), pp. 1812–1825.

[162] D.-F. Huang. “The Direct Integer Factorization Approach to the Crochiere and Rabiner Multistage FIR Designs for Multirate Systems”. In: *Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis, 2003. ISPA 2003*. Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis, 2003. ISPA 2003. Vol. 2. Sept. 2003, 1060–1065 Vol.2.

[163] D.-F. Huang and S.-R. Hung. “The Optimum Design of Multistage Multirate FIR Filter for Audio Signal Sampling Rate Conversion via a Genetic Algorithm Approach”. In: *2nd International Congress on Image and Signal Processing, 2009. CISP ’09*. 2nd International Congress on Image and Signal Processing, 2009. CISP ’09. Oct. 2009, pp. 1–5.

[164] H. Mertens. “Directional Hearing in Stereophony Theory and Experimental Verification”. In: *EBU Review* 92 (1965), pp. 146–158.

[165] A. Sontacchi et al. “An Objective Model of Localisation in Binaural Sound Reproduction Systems”. In: *Audio Engineering Society Conference: 21st International Conference: Architectural Acoustics and Sound Reinforcement*. 2002.

[166] C. R. Farrar and K. Worden. *Structural Health Monitoring: A Machine Learning Perspective*. Wiley-Blackwell, Nov. 30, 2012. 654 pp.

[167] D. Ye et al. “Measurement and Analysis on Audio Latency for Multiple Operating Systems”. In: 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA). May 2018, pp. 2496–2500.

[168] Realtek. *ALC885-GR, ALC885M-GR 7.1+2 Channel High-Performance HDA Codec with Content Protection Datasheet*. Oct. 18, 2006.

[169] *PortAudio - an Open-Source Cross-Platform Audio API*. URL: `http://www.portaudio.com/` (visited on 01/15/2010).

[170] JACK. *JACK Audio Connection Kit*. URL: `http://jackaudio.org/files/docs/html/index.html` (visited on 01/15/2010).

[171] B. Roggendorf et al. *Ableton Reference Manual Version 8, Live Version 8.02 for Windows and Mac OS*. May 2009.

[172] John Tomarakos. *Interfacing the ADSP-21161 SIMD SHARC DSP to the AD1836 (24-Bit/96kHz) Multichannel Codec*. 2001.

[173] R. Jia et al. “Analysis and Test of Sound Delay on Web Audio under Different Situations”. In: 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA). May 2018, pp. 1515–1519.

[174] J. Lehoczky, L. Sha, and Y. Ding. “The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior”. In: *[1989] Proceedings. Real-Time Systems Symposium*. [1989] Proceedings. Real-Time Systems Symposium. Dec. 1989, pp. 166–171.

[175] M. Park. “Non-Preemptive Fixed Priority Scheduling of Hard Real-Time Periodic Tasks”. In: *Computational Science–ICCS 2007* (2007), pp. 881–888.

[176] M. Nasri and B. B. Brandenburg. “Offline Equivalence: A Non-Preemptive Scheduling Technique for Resource-Constrained Embedded Real-Time Systems (Outstanding Paper)”. In: 2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS). Apr. 2017, pp. 75–86.

[177] P. Sucha et al. “TORSCHE Scheduling Toolbox for Matlab”. In: 2006 IEEE Conference on Computer Aided Control System Design. Oct. 2006, pp. 1181–1186.

[178] T. Ma et al. “Evaluation of Flexilink as Unified Real-Time Protocol for Industrial Networks”. In: 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA). May 2018, pp. 123–128.

[179] *BBC Academy - Technology - Virtualising Local Radio*. URL: http://www.bbc.co.uk/academy/technology/article/art20130802172702565 (visited on 05/07/2015).

[180] J. S. Grant. *Method and Apparatus for Transceiving Data*. Vol. PCT/GB2010/050029. WO Patent WO/2010/082,042. Patent, July 2010.

[181] D. G. Froggatt. *Data Transmission System*. Vol. EP Patent App. EP19,850,303,555. Google Patents, Jan. 1986.

[182] P. Strong, T. Wild, and G. Dean. "Latency Reduction by Adaptive Packet Fragmentation". In: *U.S. Patent Application No. 11/469,196.* (2006).

[183] IEC Project Team 62379. "IEC 62379 Common Control Interface for Networked Digital Audio and Video Products - Part 5-2: Transmission over Networks - Signalling". In: *IEC Standards* (2012).

[184] A. Networks. *Extended Frame Sizes for Next Generation Ethernets*. 1998.

[185] Audio Engineering Society. "AES51-2006 (R2011) AES Standard for Digital Audio Digital Input-Output Interfacing Transmission of ATM Cells over Ethernet Physical Layer". In: *AES STANDARDS* (2006).

[186] MathWorks. *SimEvents User's Guide*. The MathWorks, Inc., 2014.

[187] C. Labovitz et al. "Internet Inter-Domain Traffic". In: *Proceedings of the ACM SIGCOMM 2010 Conference*. SIGCOMM '10. New York, NY, USA: ACM, 2010, pp. 75–86.

[188] D. D. Clark, S. Shenker, and L. Zhang. *Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism*. Vol. 22. ACM, 1992.

[189] I. Parvez et al. "A Survey on Low Latency Towards 5G: RAN, Core Network and Caching Solutions". In: *arXiv preprint arXiv:1708.02562* (2017).

[190] H. Suzuki, S. Morita, and T. Shindo. "On the Perception of Phase Distortion". In: *Journal of the Audio Engineering Society* 28.9 (Sept. 1, 1980), pp. 570–574.

[191] S. K. Khaitan and J. D. McCalley. "Design Techniques and Applications of Cyberphysical Systems: A Survey". In: *IEEE Systems Journal* 9.2 (2015), pp. 350–365.