An Adaptive Digital Dynamic Range Controller

by

A. Todd Schneider

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical Engineering

Waterloo, Ontario, Canada, 1991

©A. Todd Schneider 1991
I hereby declare that I am the sole author of this thesis.

I authorize the University of Waterloo to lend this thesis to other institutions or individuals for the purpose of scholarly research.

[Signature]

I further authorize the University of Waterloo to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

[Signature]
The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.
Abstract

High fidelity digital audio sources are capable of reproducing a much wider dynamic range than most conventional consumer media (e.g. AM/FM radio and audio cassettes). The research presented here addresses the problem of matching this wide dynamic range to that of a device (or channel) with lower dynamic range capabilities using a Dynamic Range Controller (DRC). Currently available digital signal processing hardware allows the implementation of entirely Digital DRC's (DDRC's) that interface directly to digital sources and eliminate unnecessary data (analog ↔ digital) conversions.

The DDRC design presented in this thesis uses an adaptive level measurement scheme and an adaptive recovery time to improve performance. The perceived distortion introduced by rapid gain reductions (attack) is lessened by allowing attacks only at the zero crossing preceding a transient. A single-channel version of the Adaptive DDRC has been implemented for real-time operation on a DSP56000 evaluation board.

Tests showed that the Adaptive DDRC has insignificant total harmonic distortion. Intermodulation distortion measurements compare favourably with a previous DDRC design [11] that was reported as having good subjective performance. The results of our listening tests show great promise for the Adaptive DDRC. Listeners rated the average sound quality of an Adaptive DDRC configuration higher than a conventional design (with peak level gain control). However, since other Adaptive DDRC configurations (i.e. different parameter sets) did not perform as well, further testing is required to optimize the Adaptive DDRC parameter set.
Acknowledgements

I would like to thank my supervisor, Professor John Hanson for his advice, encouragement and faith in my abilities. Dr. Robert Brennan deserves special mention for his insight, patience and enlightening answers to my many questions. Finally I would like to thank Dan Murray for coffee, advice and SNL re-runs.

I am grateful to Motorola Inc.'s University Support Program for the donation of the DSP56000 evaluation boards, software and DSP56ADC16 evaluation modules that allowed the real-time implementation of the Adaptive DDRC. The financial support of the Information Technology Research Center (ITRC) is gratefully acknowledged.
Design of an Improved DDRC

3.1 General .................................................................................................................. 23
  3.1.1 Statistics Selection .......................................................................................... 24
  3.1.2 Design Method ............................................................................................... 26
3.2 Recovery Time Adaptation ..................................................................................... 28
  3.2.1 Adaptation Equation ....................................................................................... 30
3.3 Level Adaptation .................................................................................................... 31
  3.3.1 Adaptation Equation ....................................................................................... 32
3.4 Attack Design .......................................................................................................... 33
  3.4.1 THD and Spectral Tests .................................................................................. 34
  3.4.2 Listening Tests ............................................................................................... 37
  3.4.3 Attack Implementation .................................................................................... 38
3.5 Summary .................................................................................................................. 39

4 Implementation ......................................................................................................... 40
  4.1 General .................................................................................................................. 40
  4.2 Input High-pass Filter ......................................................................................... 41
  4.3 Zero Crossing Detector and Buffering ................................................................. 44
    4.3.1 Zero Crossing Detector ............................................................................... 44
    4.3.2 Buffers ....................................................................................................... 44
  4.4 Level Measurement ............................................................................................... 46
    4.4.1 Peak Detector ............................................................................................. 46
    4.4.2 Average Level ............................................................................................. 47
A Code Listings 104
B Memory Map 129
C Level Adaptation Scaling Factor 131
D Polynomial Approximation Coefficients 133
E Mixer Schematic 135
F Listening Test Results 136
List of Tables

2.1 Typical Static Characteristics ........................................... 14

3.1 THD for Peak and Zero Crossing Gain Reductions .................. 36

4.1 Peak Error for BFP Look-up Tables ................................. 61

4.2 Error for Non-Uniform Look-up Tables for $\log_2(x)$ and $2^x$ .......................................................... 63

4.3 Peak Error for BFP Chebyshev Polynomial Approximations .... 65

4.4 Computation Time for Selected Approximation Methods ........ 68

4.5 Execution Times for Search Algorithms on DSP56000 ........... 69

5.1 Parameters for DDRC Adaptation Control ........................... 73

5.2 Parameters for Static Characteristics ............................... 73

5.3 Execution Times for DDRC Algorithm Components ................ 76

5.4 Measured Static Characteristics ................................... 78

5.5 Five-point Opinion Scale ............................................. 93

5.6 Mean Opinion Scores ................................................ 96

D.1 BFP Chebyshev Polynomial Coefficients ............................. 133
D.2 BFP Chebyshev Polynomial Coefficients ................. 134

D.3 BFP Chebyshev Polynomial Coefficients (two polynomials) ...... 134

F.1 Listening Test Results ........................................... 137
# List of Figures

1.1 Dynamic Range of Various Media ........................................ 2

2.1 Negative Feedback DRC ...................................................... 7

2.2 Feedforward DRC ............................................................ 8

2.3 Feedback DRC with Duplicate Gain Stage ............................... 8

2.4 Possible Static Characteristics .......................................... 10

2.5 Four Region Static Characteristics ...................................... 11

2.6 Gain for Four Region Static Characteristics ............................ 13

2.7 Attack and Recovery Times ................................................. 15

2.8 Block Diagram for McNally's Autorecovery Method ..................... 19

2.9 Block Diagram of Peak Detector ......................................... 20

2.10 Block Diagram of Digital Peak Detector ............................... 20

2.11 Block Diagram of RMS Level Detector ................................. 22

3.1 Block Diagram of the Adaptive DDRC .................................. 25

3.2 Calculation of Peak Variation ............................................. 27

3.3 Adaptive Level Weight Adjustment ....................................... 32
3.4 Spectra of Low-frequency Test Signal .......................... 35
3.5 Spectra of High-frequency Test Signal .......................... 36
3.6 Envelope of Test Signal Segment ................................. 37
3.7 Listening Test Set-up ............................................. 38

4.1 Overview Flow Chart of Adaptive DDRC Algorithm ............... 42
4.2 First-order Input High-pass Filter ................................ 43
4.3 Frequency Response for First-order HPF .......................... 43
4.4 Input and Peak Buffer Operation .................................. 48
4.5 Averaging Filter .................................................. 49
4.6 Block Diagram of Adaptive Level Scheme ........................ 50
4.7 Flow Chart of Adaptive Level Implementation ..................... 52
4.8 First Order Recovery Filter ...................................... 53
4.9 Filter Parameter (a) versus T_r ................................... 53
4.10 Attack/Recovery Section of DDRC ................................. 57
4.11 Computation of Static Characteristics ............................ 58
4.12 Error for Chebyshev Approximation over BFP Range for \( \log_2(x) \) .... 65
4.13 Error for Two Chebyshev Approximations to \( \log_2(x) \) for BFP range .... 66
4.14 Flow Chart of Static Characteristics Implementation ......... 70

5.1 Real-time Test Setup .............................................. 75
5.2 Measured Static Characteristics ................................... 79
5.3 Static Characteristics Measured Using Simulator .................. 79
5.4 Error for Static Characteristics ................................... 80
Chapter 1

Introduction

1.1 General

With the advent of digital audio equipment (like Compact Disc and Digital Audio Tape), the quality of audio fidelity has increased dramatically. In particular, these digital sources allow the reproduction of wider dynamic range (the ratio of the softest to the loudest sound level) than the majority of previously available analog media. This wide dynamic range enhances the realism of audio programs, but it can cause over-load or over-modulation in some situations.

Typically, the dynamic range of a high fidelity digital source is greater than 90 dB. While this is still less than the dynamic range of a live performance (>100 dB), it still exceeds the dynamic range capabilities of many conventional recording and reproduction methods (Figure 1.1). Since these media are still in use, the problem of matching the dynamic range of the source to that of the media must be addressed. This problem is particularly relevant for radio broadcasting where the robust nature of digital sources is desirable but where their dynamic range can cause over-modulation and a visit from the Department of Communications!

Dynamic Range Controllers (DRC's) have many applications. Typically they are used
CHAPTER 1. INTRODUCTION

for dynamic range control of signals (music and speech) for recording and broadcasting. They also find application in hearing aids where they are used to compress the dynamic range of a listening environment into the residual dynamic range of a hearing impaired person. Some noise reduction systems [2],[4],[19] use compression and a complementary expansion to increase dynamic range and reduce the media noise for magnetic tape and similar media. By making the amount of compression a function of the background noise, a DRC can also be used to match listening levels to a listener's environment in noisy areas (e.g. car audio).

Dynamic range control is accomplished via a Dynamic Range Controller (DRC). The gain of a DRC varies with time, as a function of the input signal level. A DRC can produce a reduction in dynamic range (know as compression) or an increase in dynamic range (know as expansion). Many DRC's also perform limiting, which is just severe compression. A DRC intentionally amplitude distorts an input signal to reduce the dynamic range while introducing minimum perceived distortion. Thus, DRC evaluation is subjective, although objective measurements are used in the initial stages of design.

Extensive work has been done into the design and implementation of DRC's in the past. Recently, some Digital Dynamic Range Controllers (DDRC's), DRC's that utilize
digital signal processing (DSP) methods, have been developed. Using DSP provides flexibility that is not attainable via analog methods. If a DDRC provides a digital input port (e.g. an AES/EBU interface), unnecessary analog-to-digital (A/D) and digital-to-analog (D/A) conversions can also be eliminated. By using at least 24 bit values for intermediate processing very high fidelity can be maintained.

A survey of previous work indicated that older DDRC designs had not exploited the full power of DSP, such as adaptation. All of the designs used (expensive) custom hardware. With the powerful single-chip DSP processors available, it is now possible to implement an adaptive DDRC using off-the-shelf signal processing hardware.

1.2 Overview

This thesis covers the design and implementation of a single-channel adaptive DDRC. The design uses adaptive recovery time and level measurement to obtain improved performance. To reduce distortion, sudden gain reductions (known as attack) occur only at zero crossings of the output signal. The design provides wide frequency response (40Hz to 20kHz) and uses all of the capabilities of the Motorola DSP56000 to achieve the best sound fidelity possible.

The goal of this research is to design and implement (in real-time) a flexible single-channel DDRC on a single DSP processor. The possibilities for real-time adaptation of the recovery time and signal level measurement are also explored. Because of the large number of user adjustable parameters, only simple tests are conducted to confirm operation and compare the performance of non-adaptive and adaptive DDRC’s. No attempt is made to arrive at the “best” set of parameters since this is dependent on user requirements.

The thesis is divided into six chapters. Chapter two introduces the terminology necessary to understand the design of the adaptive DDRC. It also covers basic topologies and presents a brief survey of previous research into DDRC’s. The high-level design of
CHAPTER 1. INTRODUCTION

the Adaptive DDRC (ADDRC) is presented in Chapter three. Heuristic adaptation rules and the rationale for them are presented here. Chapter four covers the real-time implementation of the ADDRC on the Motorola DSP56000 Evaluation Module. The trade-offs and compromises made for the real-time implementation are examined. The results of subjective and objective tests are presented in Chapter five. Chapter six summarizes the test results, presents conclusions and suggests further work to improve the ADDRC design.
Chapter 2

Background

2.1 General

The simplest form of dynamic range control is a volume (or level) control that is adjusted (over time) by a human operator to maintain a sound level that is not "buried" in channel (or media) noise and does not introduce objectionable distortion as a result of clipping. However, a human operator is very slow acting and would be likely to disregard short, isolated transients. A better approach would be to design a device (a DRC) that automatically measures the signal and adjusts its gain to maintain a signal that is above the noise floor and below clipping. To achieve good performance, the DRC should perceptually fool the listener into thinking that the processed signal is almost the same as the original.

In operation, a DRC measures the input and/or output signal to obtain a set of parameters used to determine the gain applied to the input signal. It is desirable that the parameters have some psychoacoustic significance. However, knowledge of psychoacoustics is so limited that any parameter set will be a rough approximation to the ideal set of perceptually relevant parameters.
CHAPTER 2. BACKGROUND

The channel (or media) characteristics are well understood. The channel has a maximum level above which clipping will occur and a minimum level below which the signal will be inaudible over noise. The dynamic range of the channel is the ratio of the clipping level to the minimum audible level (often called the noise floor). The difference between the dynamic range of the channel and the dynamic range of the signal (computed in a manner similar the the channel dynamic range) is the difference in amplification for high and low level signals. This is equivalent to the DRC output for periodic input waveforms (i.e. steady state). These characteristics are specified by the static parameters.

The temporal characteristics of the gain must also be considered. The gain must change over time—otherwise the DRC will be nothing more than a volume control. The onset and duration of gain changes are specified by the dynamic characteristics. The dynamic characteristics are related to the modulation distortion, effective compression ratio and noise masking capabilities of the DRC.

When specifying the characteristics of a DRC, the application must also be considered. DRC's are used for a number of basic tasks:

1. maintain output peaks below a specified level (peak limiting)
2. reduce output dynamic range (compression)
3. increase output dynamic range (expansion)
4. eliminate all signal below a specified level (noise gate)
5. any combination of the above

These applications place restrictions on the performance requirements that must be considered when selecting a topology and specifying static and dynamic characteristics. For example, if a DRC is to be used for peak limiting, the peak level must always be kept below the maximum output level. Clearly, it is desirable to design a DRC that can perform all of the above functions.
2.2 Topologies

The first DRC's were constructed using negative feedback control loops or side chains (Figure 2.1). Because negative feedback linearizes and stabilizes the feedback loop, the exact shape of the side chain characteristics are unimportant. However, the side chain propagation delay allows a transient to reach the output before the gain is reduced. Thus, it is inevitable that overshoots will reach the output. This is a serious problem for peak limiter designs.

A feedforward system with a delay in the forward signal path can eliminate this overshoot problem (Figure 2.2). The delay in the signal path accounts for the side chain propagation delay. Thus, the gain can act to suppress a transient before it reaches the output. This also improves the subjective performance [13]. This design has the disadvantage that the linearity and stability of the side chain must be precisely controlled. This presents a challenge for analog implementations, but not for DSP methods.

Feedforward and feedback topologies may be combined to yield a structure that does not require variable gain elements with accurately specified characteristics (Figure 2.3). With this design, the gain control elements must be matched. This is still a difficult
CHAPTER 2. BACKGROUND

Figure 2.2: Feedforward DRC

Figure 2.3: Feedback DRC with Duplicate Gain Stage
task but it is easier to accomplish than accurately specifying a single characteristic. This technique is used successfully in modern high-performance compressors and limiters.

Both the feedforward and combined methods require a delay in the signal path. In analog designs, it is costly and difficult to generate a high-fidelity delay (this is typically done using CCD delay lines). Using DSP, accurate control over the side chain characteristics is easy to achieve if high precision fixed point arithmetic is used (24 bits). Also, it is easy to generate a delay.

Olivera [14] showed that feedforward designs are superior to feedback designs because they can achieve infinite compression (i.e. limiting) with finite side chain gain while a feedback design requires infinite gain. For large compression ratios, a feedback design operates at almost open-loop, a situation that can lead to instability. As well, as the amount of compression is changed in a feedback system, the dynamic characteristics are altered.

2.3 Static Characteristics

The static characteristics specify the steady-state performance of the DRC. They describe the "instantaneous" input level versus output level (in dB) relationship and do not consider the temporal variations of the DRC gain. To match the input signal dynamic range to that of the channel (or media), the difference in the dynamic range between the channel and the program must equal the difference in amplification for high and low-level signals.

This difference specifies the range of the static characteristics, but it provides no information about the distribution of the gain. The gain distribution is an important consideration because it will affect the output program quality. For example, if we were to compress a signal (reduce its dynamic range) we might (1) apply a gain of one to peak signals and amplify the low-level signals or (2) attenuate peaks and amplify low-level
signals somewhat. We may concentrate the gain variation or spread it across the entire input range. Because the gain distribution is user and input signal dependent, there is no optimum solution to this problem. We can specify any number of characteristics that provide the correct total dynamic range compression (Figure 2.4 [3]).

Static characteristics are typically specified using ratios and thresholds:

1. Ratio (R): The ratio between changes in level (measured in dB) at the input and output of the DRC. R is the slope of the static characteristics (output versus input).

2. Threshold: The input signal level where the static characteristics change slope (i.e. a new Ratio).

The definitions can be used to formalize the notions of compression, expansion and limiting:

- **Compression**: $0 < R < 1$
- **Expansion**: $R > 1$
- **Limiting**: Ideally, $R = 0$ but typical designs use $0 < R < 0.01$. 

\[
\text{Figure 2.4: Possible Static Characteristics}
\]
The properties of level independence and low-level expansion have been shown to be desirable attributes of static characteristics [12],[18],[3].

1. **Level Independence:** Static characteristics where $R$ is constant (i.e. straightline on a dB scale) between thresholds. That is, $R$ is not a function of the input level.

2. **Low-level Expansion:** Whenever a signal is compressed, the signal-to-noise ratio (SNR) is reduced by the amount that the signal is compressed. Low-level expansion is used to restore the SNR to its original value by suppressing (low-level) signals below an expansion threshold. This threshold is typically set just above the noise floor.

In principle, a large number of regions (each with a ratio and threshold) can be used as a piece-wise linear approximation to any desired characteristic. However, many designs incorporate these feature into static characteristics with four regions (Figure 2.5). These characteristics are simple to implement and they have been used successfully in many compressor/limiter designs in the past [3],[12] [18]. Four region characteristics have three thresholds:
CHAPTER 2. BACKGROUND

- **Expansion Threshold** ($E_{th}$): Below this threshold expansion is used to maintain the SNR.

- **Compression Threshold** ($C_{th}$): Above this threshold (and below the limit threshold) compression is used to reduce the output signal dynamic range.

- **Limit Threshold** ($L_{th}$): Above this threshold (and below the maximum input level) limiting keeps the output from exceeding the maximum input to the channel.

There is also a no-action region between the expansion and compression thresholds. In this region the signal is passed with a gain of one. The ratios within all other regions are within the ranges listed previously. Some designs also use characteristics with "soft-knees" [14]. That is, the ratios change asymptotically from one region to another with smooth transitions as opposed to sharp breaks.

Let $X$ be the input level in decibels (dB) and $Y$ be the output level in dB.

$$X = 20 \log_{10} \left( \frac{V_{in}}{V_{max}} \right)$$

$$Y = 20 \log_{10} \left( \frac{V_{out}}{V_{max}} \right)$$

The input and output are both scaled by the maximum input level, which is equal to $32767/32768 \approx 1.0$ for 16 bit fractional twos complement arithmetic. Within each region, the input-output relationship is

$$Y = RX$$

where $R$ is the ratio. Since $V_{max} \approx 1.0$, this can be rewritten as

$$V_{out} = V_{in}^R$$

We see that equal level differences on the input are mapped to equal level differences on the output that are $R$ times smaller.

The DRC output signal is computed as $V_{out} = V_{in} \times G$ where $G$ is the DRC gain. The gains within each region are as shown below, where $R_E$ is the expansion ratio; $R_C$ is the compression ratio and $R_L$ is the limit ratio:
• Expansion: $G_{dB} = (R_E - 1)(X - E_{th}) = ES(X - E_{th})$

• Compression: $G_{dB} = (R_C - 1)(X - C_{th}) = CS(X - C_{th})$

• No-action: $G_{dB} = 0$

• Limiting:

\[
G_{dB} = (R_L - 1)(X - L_{th}) + (R_C - 1)(L_{th} - C_{th})
\]
\[
= LS(X - L_{th}) + CS(L_{th} - C_{th})
\]

where $ES = R_E - 1$ is the expansion slope. The limit slope and compression slopes are defined in a similar manner with their respective thresholds. The gain expressions for the expansion and compression regions are similar in form. The gain expression for the limiting region contains an additional term that accounts for the gain applied by the compression before the limit threshold. Table 2.1 shows typical static characteristics for a DRC. Figure 2.5 shows the input-output relationships for these characteristics. Figure 2.6 shows the input-gain relationship for the same characteristics.
2.4 Dynamic Characteristics

As indicated previously, the dynamic characteristics describe the temporal variations of the DRC gain. When specifying the dynamic characteristics, there are three basic questions (originally suggested by Blesser [3]) that must be answered:

1. When should the gain change?
2. How quickly should the gain increase?
3. How quickly should the gain decrease?

Blesser also presents three rules to aid in the design of the dynamic characteristics:

1. The gain should not change during the duration of a single note to ensure that the output is a faithful reproduction of the input.
2. Transients should not dominate the program (i.e. the gain control)
3. The subjective balance between different musical notes should appear the same even if the actual balance has changed.

Because the gain is computed instantaneously by the static characteristics, the dynamic characteristics are always included in the measurement section. All designs use

<table>
<thead>
<tr>
<th>Region</th>
<th>Ratio ($R$)</th>
<th>Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expansion</td>
<td>2</td>
<td>$E_{th} = -50dB$</td>
</tr>
<tr>
<td>No-action</td>
<td>1</td>
<td>—</td>
</tr>
<tr>
<td>Compression</td>
<td>0.5</td>
<td>$C_{th} = -35dB$</td>
</tr>
<tr>
<td>Limiting</td>
<td>0.01</td>
<td>$L_{th} = -15dB$</td>
</tr>
</tbody>
</table>

Table 2.1: Typical Static Characteristics
either peak detection, average signal level or RMS signal level for gain control. Each of these methods is characterized by an attack time and a recovery time. A hold time is used in some designs.

### 2.4.1 Attack Time

Attack times have been subject to a number of definitions on the past [12]. For our work, the attack time \( T_a \) is defined as the time taken for 63.2% of the total change in gain when the input exceeds a DRC threshold (Figure 2.7 [12]). This definition is consistent with the conventional notion of a time constant.

Most previous designs have used a first-order exponential attack time that is typically less than 10ms for compression/limiting of musical signals. This value is a compromise. A short \( T_a \) makes the output look more like the input while a long \( T_a \) accentuates the initial part of a transient. Short attack times (less than 10ms) are generally not perceived; however, the gain tends to be determined by the peaks in the program. Since the ear responds to loudness and not peaks, signals with high-level peaks are not necessarily loud. Under some conditions, this can produce "a very flat sound with an apparently negative compression ratio" [3]. This problem is typically solved by using an attack time
CHAPTER 2. BACKGROUND

of approximately 5ms. Extensive subjective tests conducted by Wagenaars et al. [20] agree with this value.

A recent peak limiter design [11] has taken a different approach. In this design the gain attacks at zero crossings. "This makes it possible to obtain an instantaneous attack without distortion of the waveform peak." The reports on the subjective performance of this device are very promising.

2.4.2 Recovery Time

As with attack times, recovery times have been defined in a number of ways in the past [12]. To be consistent with our definition of attack time, the recovery time ($T_r$) is defined as the time taken for the DRC gain to achieve 63.2% of the total change in gain when the input falls below a threshold (Figure 2.7). This definition is consistent with the notion of a first-order time constant. Most recovery characteristics are first-order exponential.

Typically, DRC's use a long $T_r$ to reduce the distortion introduced by the gain amplitude modulating the signal. However, with a short $T_a$ and a long $T_r$, a single high-level transient can create a "hole" in the program by blanking out the musical program following it. Also, a very long $T_r$ will change the gain so slowly that the compression/limiting will be more like volume control. A short $T_r$ can also lead to an exaggeration of speech and breath noises and modulation of otherwise constant amplitude background noise. These phenomena are often called "gain pumping," "breathing" and "swishing" [11]. Thus, if a fixed $T_r$ is used, a compromise between modulation distortion, the creation of holes in the program, effective signal compression and gain pumping must be made.

Compromise $T_r$'s on the order of 200-250ms are typically used. A study by Wagenaars et al. [20] found that a $T_r$ of approximately 200ms$^1$ produced good subjective results.

$^1$This design[18] used a 3"-order recovery stage with $T_{r3} = 62.5$ms to "achieve a kind of hold effect." An equivalent 1"-order recovery time is $T_{r1} = 3.25T_{r3} \approx 203$ms
In a classic paper on the design of an adaptive compressor/limiter [3], Blesser argues that "a single recovery time does not take into account the psychophysical characteristics of hearing." He considered the dynamic range of a musical program in two components:

- **Short-term dynamic range** - The ratio of maximum to minimum level over about a second. It corresponds to the "fullness" or "muddiness" of a piece of music.

- **Long-term dynamic range** - The ratio of maximum to minimum level over a thirty second interval. It corresponds to the "mood" of a piece.

If there is a large discrepancy between the ratio of the short- and long-term dynamic range of the input to that of the output, the music does not sound natural. When a DRC compresses or limits a piece of music, the long-term dynamic range is unaffected by $T_r$; it is determined by the static compression ratio adjustment. The short-term dynamic range depends on $T_r$.

Consider a piece of music that is heavily compressed using long recovery time. The gain changes very slowly after a high peak and for normal programs the gain is constant. Thus, there is no change in the dynamic range (i.e. we have implemented a volume control). The gain must vary to produce effective compression. Blesser shows that the effective compression ratio (as a percentage, 100% corresponding to maximum compression) can be written as

$$CR_e = 1 + \frac{(CR_s - 1)T}{kT_r}$$

for $T < kT_r$, where $T$ is the time interval between peaks; $CR_s$ is the static compression ratio; $CR_e$ is the effective compression ratio and $k$ is some constant. The relationship shows that the recovery time function has only one degree of freedom—one parameter determines all of its properties. Clearly this is undesirable.

To overcome this difficulty, Blesser split $T_r$ into two parts, a short-term $T_s$ and a long-term $T_r$. The short-term $T_s$ controls the recovery of a percentage of the gain change,
CHAPTER 2. BACKGROUND

$P_*$, caused by the last peak. The long-term $T_r$ controls the recovery of the remaining percentage, $1 - P_*$. He also includes a hold time ($T_h$) that keeps the gain constant (for $T_h = 100\text{ms}$) until most of the musical event that caused the gain change has passed. If a new peak arrives during the hold time, the recovery is again postponed. The hold time allows the compression section to decide if it really "wants" to recover.

The essence of good dynamic range compression is the proper adjustment of $P_*$. Empirically, Blesser found that making $P_*$ a piecewise linear function of $T$ (the time between peaks) produced good results. He also allowed the peak-to-average loudness ratio of the program to control $T_r$ so that a program's original dynamic range determined how much compression was used. It is not revealed how this is accomplished in the actual design.

Blesser also includes an adaptive $T_r$ in the limiting section of his DRC design. He sets another threshold 2dB below the limit threshold ($L_{th}$). The frequency with which peaks enter this window (i.e. between $L_{th}$ and $L_{th}-2$dB) is used to adjust $T_r$. For a single peak, the DRC reduces the gain and then immediately returns it to its previous value. For repetitive peaks, the DRC reduces the gain for the first peak and then observes that succeeding peaks lie within the window. $T_r$ then increases and the gain remains constant. $T_r$ varies from 150ms to more than 30s in limiting.

Mapes-Riordan and Leach [11] developed a digital peak limiter that recovers in fixed amounts at zero crossings of the signal. They explain that this "makes the recovery time inversely proportional to frequency, thus minimizing the problems associated with a fixed recovery time." Thus, distortion of low-frequency signals caused by too short a recovery time is minimized while program "holes" and "dropouts" caused by short duration peaks are minimized. They report good subjective results with this design.

McNally [12] proposes a scheme for adapting $T_r$ (Figure 2.8). $T_r$ is adapted based on the peak and RMS level of the input signal so that a short $T_r$ is used for isolated peaks but with signals of higher average level, the $T_r$ is extended to avoid excessive limiting or
2.5 Measurement Methods

In the past, DRC designs have used three methods of signal measurement for gain control: peak detection, average signal level or RMS signal level. Each of these methods has perceptual consequences and an effect on the attack and recovery times of the DRC gain.

2.5.1 Peak Detection

Peak level detection provides a simple method of achieving a short $T_a$ and a long $T_r$. It ensures that the output peak level never exceeds the maximum input level of the channel. Most peak measurement systems are constructed as shown in Figure 2.9. In this design, $R_a$ controls $T_a$ and $R_r$ controls $T_r$. Typically, $R_r \gg R_a$ so $T_a \approx R_a C$ and $T_r \approx R_r C$.

A digital implementation of a peak detector is shown in Figure 2.10. Here, $T A$ controls $T_a$ and $T R$ controls $T_r$. The non-linear element that receives the difference of the full-wave rectified input ($x[n]$) and the delayed output ($y[n]$) simulates an ideal diode using
CHAPTER 2. BACKGROUND

Figure 2.9: Block Diagram of Peak Detector

Figure 2.10: Block Diagram of Digital Peak Detector
if $z[n] - y[n] > 0$

$$q[n] = z[n] - y[n]$$

else

$$q[n] = 0$$

In their design of a digital peak limiter, Mapes-Riordan and Leach [11] use the peaks between zero crossings and compute a gain to reduce a peak below the maximum input to the channel if the peak exceeds the limit threshold.

Using peak level for gain control has the disadvantage that the ear does not respond to the peak level of a signal. The ear's response resembles a RMS or average level measurement [5]. Thus, peak level control causes gain changes to be perceived as being unrelated to program content.

2.5.2 Average and RMS Level

Some designs have used average or RMS levels for gain control. Generally this improves the perceptual performance of a DRC because the gain control is better related to the perceived signal level [1]. However, to approximate the ear's response, averaging times that are much longer than the attack times required to suppress transients are used. Thus, transients exceeding the maximum channel input may appear on the output of the DRC.

These methods also have the disadvantage that $T_r = T_a$ for most simple implementations. Figure 2.11 shows a digital implementation of a RMS level measurement. To implement an average measurement, the squaring operation on the input is replaced with a full-wave rectifier (i.e. absolute value).
2.6 Summary

Previous research has been done in the design and implementation of DRC's. This chapter presented general background on a number of designs and introduced the terminology required to understand the operation of a DRC.

Most recent DDRC designs have used custom hardware. Typically, they have been re-implementations of traditional designs. These designs have made a number of compromises that will be addressed in our improved DDRC design that is presented in the next chapter.
Chapter 3

Design of an Improved DDRC

3.1 General

Previous DRC designs have been implemented via analog or digital methods. We selected DSP as the implementation method because it provides powerful processing options like adaptation that are difficult to implement using other methods. Also, DSP-based designs are immune to temperature variations; they do not depend on component tolerances and are generally more stable than analog implementations.

A feedforward topology was selected because it offers a number of advantages over other options (Section 2.2). The side chain linearity requirement of this topology does not pose any serious problems in a DSP-based implementation.

The four region static characteristics described in Section 2.3 will be used. These characteristics are reasonably easy to implement and they have been used successfully in many compressor/limiter designs in the past [12],[3],[18].

Because musical signals are non-stationary, it is not possible to select a set of fixed parameters (e.g. T, and the time constants for level measurements) that will provide good performance for all input signals (or even different portions of the same input signal).
With fixed parameters, a compromise must always be made. To improve performance, our design is adaptive.

The input signal and peak level are buffered so adaptation can be based on the future (with respect to the output) statistics of the input signal. The level measurement and $T_r$ adapt based on input signal statistics. Novel attack characteristics are also used to improve performance. Figure 3.1 shows a block diagram of the adaptive DDRC. The remainder of this chapter describes the high-level design of the adaptive $T_r$ section, the adaptive level measurement and the attack portion of the adaptive DDRC design.

3.1.1 Statistics Selection

A set of input signal statistics to control the adaptation must be selected. Ideally, a set of perceptually relevant, orthogonal input signal statistics should be used. This is impossible to derive except for very simple inputs so an approximation to the ideal set of statistics must be used. In a real-time implementation, the statistics must be computed rapidly. Thus, a set of easy to compute statistics that have some perceptual bearing on the signal or that are quantities we must control will govern the adaptation of $T_r$ and the level measurement.

After some background experimentation and consideration of the issues involved in real-time computation, the statistics listed below were selected for use in the adaptive DDRC. Further explanation of the rationale behind the selection of these statistics is presented in later sections of this chapter.

1. Peak Level: The peak level between zero crossings. This is a quantity that we wish to control. Zero crossings provide a convenient block size for the determination of peak level.

2. Average Level: This is the local level of the signal. It is related to the perceived signal level. An averaging time of 100ms will be used.
CHAPTER 3. DESIGN OF AN IMPROVED DDRC

Figure 3.1: Block Diagram of the Adaptive DDRC
3. Crest Factor ($C_f$): This is the ratio of the peak level to the RMS level of the signal. Our design will approximate this using,

$$C_f^{-1} = \frac{\bar{p}}{p[k]}$$

where $p[k]$ is the present peak level between zero crossings and $\bar{p}$ is the average input level. This quantity is easier to compute in real-time and it is always fractional since $p[k] \geq \bar{p}$. This statistic provides information about the "peakiness" of the original signal (i.e. the signal's short-term dynamic range).

4. Peak Variation ($P_v$): This value is computed as,

$$P_v = p[k] - \frac{1}{N} \sum_{n=1}^{N} p[k + n]$$

where $p[k]$ is the present peak level. It provides a rough measure of the isolation of a peak level by indicating how the current peak level compares to the future average peak level (Figure 3.2).

The peak variation is computed using forward averages. The input samples and peak level are buffered so that statistics based on the future signal properties (with respect to the output) can be computed.

Although the RMS signal level is better related to the perceived signal level [5], the DDRC will use the average signal level because it is easier to compute and does not suffer the underflow problems associated with squaring fractional input samples. It is possible to compute all of these statistics in real-time using relatively simple computations. These measurements will be used to adapt $T_0$ and the level measurement.

3.1.2 Design Method

To control the adaptation of the DDRC, adaptation rules (or equations) will be developed. The adaptation must be computed in real-time, so it is imperative that the rules be simple.
Figure 3.2: Calculation of Peak Variation
CHAPTER 3. DESIGN OF AN IMPROVED DDRC

It is very difficult to specify the required performance mathematically—this makes it impossible to use conventional adaptation methods like least mean square algorithm [17]. Thus, heuristic methods will be employed.

Two approaches for the development of rules were explored. We first attempted to classify all input signals based on the set of statistics presented above. Each statistic was split into three ranges. If we assume (pretend) that the statistics presented above are orthogonal, this is equivalent to partitioning a four dimensional measurement “space” and specifying a set of parameters (to control the adaptation) for each region. Although, this method provides an organized methodical approach to the adaptation design, it requires the consideration of too many classes. Thus, this method was not used.

A second approach proved more successful. This method can be split into five steps:

1. Specify the problems to be solved by the adaptation.
2. Consider how adjustment of the available parameters can be used to compensate for these situations.
3. Determine the input statistics that indicate these problems.
4. Relate these statistics to the parameter adjustments required to solve the problems.
5. Develop simple (heuristic) rules based on the relationships derived in step 4.

This approach is used to develop the adaptation rules for $T_r$ and the adaptive level.

3.2 Recovery Time Adaptation

Non-adaptive DRC's have the best performance when they use short $T_a$'s and long $T_r$'s. A long $T_r$ ($> 250 ms$) is required for low modulation distortion and to maintain constant short-term dynamic range [3]. However, this set of parameters allows isolated transients
to cause "holes" in the musical program by rapidly reducing the gain and then recovering slowly. If a short $T_r$ is used, repetitive high-level peaks will cause excessive distortion by applying a rapidly changing gain to the output signal (i.e. modulation distortion). Most DRC's use a compromise $T_r$ of 200 to 250ms to balance these difficulties.

If $T_r$ is adaptive, a compromise will not be necessary and improved performance will result. The goals of adapting $T_r$ are summarized below:

1. Maintain the relative (short-term) dynamic range of the output signal whenever possible by using a long $T_r$ (e.g. $T_r = 200ms$).

2. Use a short $T_r$ following isolated high-level peaks to eliminate "holes" in the musical program.

3. For repetitive peaks, use a long $T_r$ (in effect, a hold time) to avoid excessive modulation distortion caused by frequent gain changes.

The adaptation algorithm will use these rules to adapt $T_r$ based on the input statistics. Like most DRC's our design will use a first-order exponential recovery.

$T_r$ should be a function of the "peakiness" (relative peak size) of the input signal and the "isolation" of these peak levels. $C_f$ is a measure of the relative peak size of a signal (in relation to the average level). $P_o$ is a measure of the isolation of a peak level. The input signal properties reflected in each of these statistics and their relation to $T_r$ is considered below.

- **Crest Factor** ($C_f$): A large $C_f$ indicates a signal with large peaks relative to its average level. For this signal a short $T_r$ will avoid "holes" in the program. A small $C_f$ indicates a signal with low "peakiness". For this class of signal, a long $T_r$ will reduce modulation distortion and will not introduce "holes" in the program.

- **Peak Variation** ($P_o$): A positive $P_o$ indicates that the future average peak level is less that the present peak level. For the situation, a short $T_r$ is required because
the present peak is isolated. A negative $P_v$ indicates that the average peak level is increasing; this calls for a long $T_r$ to reduce gain modulation (in effect a hold time). A $P_v = 0$ indicates no change in the present a future average peak level. This requires a long $T_r$.

From this discussion, we see that $T_r$ is inversely related to both $C_f$ and $P_v$.

3.2.1 Adaptation Equation

To design the rules for the control of $T_r$, the priority of these statistics must be determined. $P_v$ is not important if the signal has a low peak level. $C_f$ is a more important statistic in the adaptation of $T_r$ because it provides information about the relative size of the signal peaks. If an input signal has a high $C_f$, additional information is required to adapt $T_r$. If the peak is isolated, $T_r$ should be short to avoid causing a "hole" in the program. Conversely, if the peak is not isolated, the gain should recover slowly (in effect a hold time) to reduce gain modulation. $P_v$ provides the information about the isolation of the peak level. Because the DDRC will be operating from low-noise digital sources, excessive background noise will not be a problem and the noise masking properties of the recovery time are not a consideration.

Typically, the peak and RMS levels of musical signals differ by 10dB [7] (i.e. $C_f \approx 3.2$). For low $C_f$ signals, a fixed $T_r$ will be used because the signal has low "peakiness"—a fixed $T_r$ can be used without causing holes in the program. For higher $C_f$ signals (i.e $C_f > 3.2$), $T_r$ is adapted based on $C_f$ and $P_v$. A linear adaptation equation is used to simplify real-time computation. By using $C_f^{-1}$ in the adaptation, the inverse relationship between $T_r$ and $C_f$ can be obtained. $C_f^{-1}$ is always fractional—an important consideration for implementation on the DSP56000 which uses fractional arithmetic. The adaptation rule used in the design is
if $C_f < 3.2$

$$T_r \text{ is constant}$$

else

$$T_r \propto k_1(C_f^{-1}) - k_2(P_v) + c$$

In this equation, $k_1, k_2 \geq 0$. A negative sign in front of $k_2 P_v$ provides the inverse relationship between $T_r$ and $P_v$ as described above. The constant $c$ controls the starting point of the adaptation while $k_1$ and $k_2$ control the range of the adaptation. The values used for these parameters are derived in Chapter 4. The range of the $T_r$ adaptation will typically be $50ms \leq T_r \leq 300ms$ or so. This range will be refined through listening tests.

### 3.3 Level Adaptation

Previous DDRC designs have used either peak or RMS signal level for gain control \cite{12},\cite{18},\cite{11}. As discussed in Chapter 2, peak detection is a simple-to-compute measure that ensures the channel will never be overdriven. However, the peak signal level has a weak relation to the perceived signal level. This can lead to undesirable results when compressing/limiting signals \cite{3}.

The average or RMS signal level is better measure of the perceived level of the signal. However, these measures do not respond quickly to transients because of their inherent averaging. Thus, the peak output signal level may exceed the maximum input to the channel (the saturation level) for isolated transients.

In our design, the peak and average signal levels will be adaptively combined so that, whenever possible, the average signal level is used for gain control. Using the average level will provide improved perceptual performance because the gain will be better related to the perceived signal level. Peak level or a linear combination of peak and average level will be used when necessary to ensure that the peak level does not exceed the saturation level. In summary, the goals of this design are:
• Whenever possible, use a signal measure for gain control that reflects the perceived level of the signal.

• Suppress output transients that will cause the peak output level to exceed the channel saturation level.

• Provide a smooth measure of the signal level to the gain section.

3.3.1 Adaptation Equation

The first design goal can be satisfied if the average level is used to control the gain (whenever possible). That is, if the output will not exceed the saturation level, the average level is used for gain control. If transients will cause the output to saturate the channel, the peak level will be used for gain control. This satisfies the second goal.

To realize these characteristics, two thresholds, the peak control threshold ($V_{pc}$) and the average control threshold ($V_{ac}$) are set as shown in Figure 3.3. For signals with low peak level (between zero crossings), the average level is used for gain control. For signals with high peak level, the peak level is used for gain control. Between these thresholds, a
linear combination of the average and peak levels is used as the signal "level"

\[ \text{level} = A p[k] + B \bar{x} \]

Typically, we will set \( V_{pc} = L_{th} \) so that the peak level is used for limiting. Because the average level of a signal (\( \bar{x} \)) is always less than the peak level, the thresholds must be spaced apart by a reasonable amount (e.g. 10 to 15dB) to ensure that the transition from average to peak control (and vice-versa) occurs smoothly (i.e. there is not an abrupt change in level). Thus, \( V_{ac} \approx V_{pc} - 15dB \).

This design is also very flexible. The weights \((A, B)\) and thresholds \((V_{pc}, V_{ac})\) can be adjusted to realize most conventional DRC designs.

### 3.4 Attack Design

Most conventional DRC designs attack at any point on the input signal. No effort is made to perform rapid gain reductions at points where they introduce lower distortion. Our design attacks instantaneously at the zero crossing preceding a transient in an effort to reduce the distortion introduced by rapid gain reductions.

Mapes-Riordan and Leach [11] presented a digital peak limiter design with instantaneous attack at the zero crossing preceding a transient. Their design also recovers in "fixed dB amounts" at zero crossings. They conducted tests for intermodulation distortion (using the sum of sinusoids with incommensurate periods as an input) and found that their limiter (with typical settings) introduced less total harmonic distortion (THD) than a peak clipper. The limiter output spectra for their limiter consisted only of difference frequency components while the clipper spectra contained harmonic and difference frequency components. They reported good subjective results for their design.

Our design strategy pursues a similar course to that of Mapes-Riordan and Leach. Blesser [3] reports that an instantaneous attack time will not produce good subjective
results. However, our design will only have instantaneous attack when the DRC switches
into limiting and the peak level is used for gain control. Otherwise, our design will have
an attack time that is not instantaneous because a linear combination of the average
signal level and the peak level or the average level is used for gain control.

We conducted preliminary tests to determine whether instantaneous gain reductions
at zero crossings introduced less distortion than instantaneous gain reductions at any
other point on a signal. To simplify the tests, gain reductions at peaks were chosen
for the comparison because they offered the greatest variation in performance from gain
reductions at zero crossings. Sinusoidal input signals were used for all tests because
they have clearly defined peaks and the location of the gain reductions will not vary
significantly.

3.4.1 THD and Spectral Tests

Two objective tests were conducted to assess the effects of gain changes at peaks and
zero crossings.

Total Harmonic Distortion: THD was used to measure the distortion performance.
Although this measure does not indicate the perceptual performance of any method, it
does provide an "objective yardstick" to roughly gauge the performance of each method.
THD is computed as

\[ THD = \frac{\sum_{k=0}^{N-1} (A(\omega_k) - B(\omega_k))^2}{\sum_{k=0}^{N-1} A^2(\omega_k)} \times 100\% \] (3.1)

where \( A(\omega_k) \) are the components of the magnitude spectra for the reference signal (no
gain change) and \( B(\omega_k) \) are the components of the magnitude spectra for the gain-reduced
signal.

Spectral Test: The spectra for gain reductions of 3, 6 and 9 dB for each gain reduction
method were plotted. The frequency domain characteristics of each method was exam-
To obtain the frequency domain results a 4096-point sinusoid with a single gain reduction at the peak of zero crossing closest to the 2048th sample was Hamming windowed and a 4096-point FFT was taken. A sampling rate of 10kHz was used.

Single gain reductions of 3, 6 and 9dB were applied to each signal. These gain changes are typical of the gain reductions that may be employed in a DDRC. Two test signal frequencies (312.5Hz and 1562.5Hz) were selected. The FFT provides the best results for these frequencies because the signal fits into the window with an integral number of half-periods. A Hamming window is used to taper the ends of the data sequence and reduce any DC offset introduced by non-integral numbers of full periods.

Figures 3.4 and 3.5 show the spectra for the 9dB gain reductions (f=312.5 and 1562.5 Hz). Results for the other gain reductions are similar. Table 3.1 shows the THD for these tests.

As expected both of these methods have identical THD and the THD is proportional
Figure 3.5: Spectra of High-frequency Test Signal

<table>
<thead>
<tr>
<th>Gain Red. (dB)</th>
<th>Peak</th>
<th>Zero Crossing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>312.5 Hz</td>
<td>1562.5 Hz</td>
</tr>
<tr>
<td>-3.0</td>
<td>1.21%</td>
<td>1.21%</td>
</tr>
<tr>
<td>-6.0</td>
<td>4.83%</td>
<td>4.83%</td>
</tr>
<tr>
<td>-9.0</td>
<td>10.52%</td>
<td>10.55%</td>
</tr>
</tbody>
</table>

Table 3.1: THD for Peak and Zero Crossing Gain Reductions
to the size of the gain reduction. In all cases, peak gain reduction introduces more high frequency harmonics while zero crossing gain reduction introduces more low frequency harmonics. The slight differences in THD are caused by the time differences in the location of the gain change. As the input frequency approaches the Nyquist rate, the samples where peak gain and zero crossing gain reduction occur converge. This is reflected in the spectra—they move together as the frequency increases.

### 3.4.2 Listening Tests

For the listening tests, a 4096 sample signal with the envelope shown in Figure 3.6 was generated. A number of these signals were concatenated together to form a test signal with a large number of periodic gain changes at peaks or zero crossings.

All listening tests were conducted using only a 9dB gain reduction. The test set-up is shown in Figure 3.7. A pure tone with no gain reductions was used as a reference in all listening tests.

Informal listening tests showed that the gain reductions are easily heard. For the low frequency tests \(f=312.5\), all listeners commented that the peak gain changes sounded more "clicky" than the zero crossing gain changes. This is a result of the high frequency content that is evident in the spectral plots. The low frequency distortion introduced by
zero crossing gain changes did not seem to have any perceptual effect. It is possible that the strong fundamental masks the lower frequency distortion components.

For the high frequency tests \((f=1562.5)\), the gain reductions could be clearly heard. However, there was not an audible difference between the gain-reduced signals. The 3.4kHz reconstruction filter does not allow all the high-frequency distortion introduced by peak gain changes to be heard. This should make it easier to hear the low frequency distortion introduced by zero crossing gain reduction. Yet, the listeners reported no significant differences.

From these results, we can see that zero crossing gain reduction is superior to peak gain reduction because it introduces less high frequency distortion which gives better subjective performance. This distortion is frequency dependent— as the input frequency approaches the Nyquist rate, the samples where peak and zero crossing gain reduction occur converge.

### 3.4.3 Attack Implementation

Based on the results of the tests conducted above, our design attacks at the zero crossing preceding an input signal peak. This method introduces less perceivable distortion.

In operation, the DDRC detects zero crossings of the input signal and stores the peaks
between zero crossings in the peak level buffer (Figure 3.1). A zero crossing detector at the output is used to keep the output of the peak level synchronized with the input buffer. The output zero crossing information is also provided to the attack/recovery block. If there is a zero crossing and the input level from the adaptive level computation exceeds the previous output level from the attack/recovery block, the gain will attack. Otherwise the gain will recover.

3.5 Summary

Three problems of conventional designs: recovery time, level measurement and attack characteristics have been addressed in our improved DDRC design. This chapter explained the high-level design of the features in our adaptive DDRC that solve these problems.

To obtain low modulation distortion without introducing “holes” in the program, the adaptive DDRC adapts the recovery time based on the input signal crest factor and the isolation of peak levels. Rapid gain reductions (attack) occur instantaneously at zero crossings preceding a transient. This introduces less distortion than conventional methods. The DDRC gain is controlled by an adaptive signal level that uses the peak level for limiting and the average signal level for compression and expansion. This provides improved perceptual performance because the average signal level is better related to the perceived level than the peak signal level.
Chapter 4

Implementation

4.1 General

Having completed the high-level design of the adaptive DDRC, we now turn to the real-time implementation. The DDRC will be implemented on a Motorola DSP56000 Application Development System (DSP56000ADS). This system consists of a single DSP56000 with \( 8k \times 24 \) bit of RAM and the necessary support circuitry. To simplify testing, a Motorola supplied A/D and D/A sub-system (DSP56ADC16) will be used. A direct digital input to the DSP56000 from a digital source will not be used.

The DSP56000 is a powerful DSP chip that uses 24 bit values internally to reduce round-off error. It also features two 56 bit accumulators and high-speed serial A/D and D/A input/output ports. These features make the DSP56000 ideally suited for audio signal processing. Although it is possible to use other number representations (e.g. floating point), the DSP56000 architecture is optimized for fixed-point fractional twos complement arithmetic. Thus, it is much faster to use fractional arithmetic (and scaling, if necessary). The adaptive DDRC will use 16 bit input and output samples.

Because the adaptive DDRC algorithm execution time is variable, an interrupt driven implementation is used. The DSP56000 uses a base clock rate of 20.48MHz (\( T_{\text{clock}} \approx \) \( 40 \))
49ns). Most simple DSP56000 instructions execute in two clock cycles (approximately 98ns). The DDRC will operate at a sampling rate (= interrupt rate) of 44.1kHz—the standard sampling rate for Compact Disk players. Thus, the entire adaptive DDRC algorithm must execute between two interrupts in $\frac{20.48\text{MHz}}{44.1\text{kHz}} = 464$ clock cycles or less.

The DDRC design can be split into input and output sections (Figure 3.1). The input section consists of a high-pass filter, a zero crossing detector, a peak level detector/buffer, an average level filter and an input buffer. The output section operates on delayed input and peak level samples. It performs the level adaptation, $T_r$ adaptation, attack/recovery decision and computes the static characteristics. Figure 4.1 shows an overview flow chart of the adaptive DDRC algorithm.

The remainder of this chapter covers the “block-by-block” implementation of the adaptive DDRC. Complete code listings are contained in Appendix A. A memory map is in Appendix B.

4.2 Input High-pass Filter

The input highpass filter (HPF) removes any DC component of the input signal. Stikvoort [18] included a filter in his design so input DC offsets would not be modulated by the gain control signal and cause “plops” in the output signal. Because the offset voltage and signal level are indistinguishable, a DC offset may also corrupt the average level measurement.

Mapes-Riordan and Leach [11] included an input HPF in their digital peak limiter to restrict the lowest input signal frequency so the input signal buffer would not overflow between zero crossings. In our design, the input signal buffer must store the input samples between zero crossings, but we assume that the input is bandlimited to greater than 20Hz (with a possible DC offset). Thus, the HPF will only be required to block DC. This
Chapter 4. Implementation

Input Section

1. Read input
2. Write output
3. High pass filter input
4. Detect input
5. Zero crossings
6. Buffer input samples

Output Section

7. Detect output
8. Zero crossings
9. Compute adaptive level
10. Attack/recovery
11. Compute adaptive $T_r$
12. Compute static characteristics (i.e., gain)
13. Compute output sample

Figure 4.1: Overview Flow Chart of Adaptive DDRC Algorithm
assumption allows the use of a low-order filter which saves computation time.

To minimize the computation time, our design will use a first-order HPF. The filter (Figure 4.2) has a -3dB cutoff frequency of 15Hz. The simulated frequency response is shown in Figure 4.3. Notice that the filter has infinite rejection at DC because $a_0 = -a_1$. The real-time implementation uses parallel updated pointers and orders coefficients in memory to speed execution (see gain.asm listing in Appendix A).
4.3 Zero Crossing Detector and Buffering

The input section uses two circular buffers to hold the input samples and the peak signal levels detected between input signal zero crossings (zc's). A modulo addressing mode and parallel pointer updating are used to provide fast input and peak level buffering.

4.3.1 Zero Crossing Detector

To detect zc's in real-time, an algorithm that uses multiplications and as few comparisons as possible was developed. Multiplications execute much faster than comparisons on the DSP56000. The zc detector uses the current and previous input samples ($z[n]$ and $z[n-1]$) to determine if the present input sample is a zc:

$$\text{if } z[n]z[n - 1] \leq 0 \text{ then}$$

$$\text{input is a zero crossing}$$

$$\text{else}$$

$$\text{input is not a zero crossing}$$

For $z[n] = 0$ this algorithm detects two zc's and gives a peak value of zero. This may cause the peak value to be less than the average level for a signal segment (i.e. the samples between two zc's) — the DDRC algorithm is designed so that this condition will not cause any problems. A previous zc detection algorithm that used three input samples to detect zc's did not suffer from this fault. However, it executed much more slowly than the two-sample method presented above.

4.3.2 Buffers

The circular peak and input buffer sizes are determined by the maximum (tolerable) delay through the DDRC and the number of peak levels (detected between zc's) used in the $P_o$ calculation. For the input signal buffer of size $M$, the output sample is $M - 1$ samples
behind or one sample ahead of the current input sample. Whenever there is an input zc, the current peak input level is stored and an input peak pointer is incremented. A similar zc detector at the output end of the input sample buffer increments the output peak pointer. The input buffer is filled with small positive values at initialization, so the difference between the input and output peak pointers always equals the number of zc's in the input buffer.

The input buffer must store the samples between two zc's of the lowest input frequency to the DDRC. For a sampling rate of $f_s = 44.1kHz$ and a minimum input frequency of $f_{min} = 20Hz$, the buffer must be

$$M = \left\lfloor \frac{f_s}{2f_{min}} \right\rfloor = 1103$$

samples long. The minimum size of the peak buffer is also $M$. If the highest input frequency is $f_{max} = f_s/2$ (i.e. the Nyquist rate), there will be an input zero crossing every sample. Thus, the number of zc's is equal to the number of input samples.

Recall that the $P_v$ calculation uses $N$ future peak values—each detected between a zc on the input. The input buffer must hold at least $N + 1$ zc's (i.e. $N$ future zc's and the present zc) of $f_{min}$. Thus, the input buffer must be extended to at least

$$M = (N + 1) \left\lfloor \frac{f_s}{2f_{min}} \right\rfloor = (N + 1)(1103)$$

samples long. The DDRC currently uses $N = 5$ (see Section 4.4). This means that the delay through the DDRC is

$$T_{pd} = (5 + 1)(1103) \left( \frac{1}{f_s} \right)$$

$$\simeq 150ms$$

This is a tolerable delay. The delay time is fixed so synchronization is not a problem and in most applications users will never hear the unprocessed input to the DDRC. For a delay of this length, the average measurement must also be delayed so that is does not act too far ahead of the input signal.
Presently, the DDRC uses a fixed delay of $1125(1/44100.0) \approx 25.5\text{ms}$. This is constrained by the memory available on the DSP56000ADS (i.e. money!), the restrictions of the ADS memory configurations [9] and the starting addresses that may be used for modulo addressing [15]. If more high speed memory were purchased, this would not be a limitation. This means that $P_v$ calculations for some input signals (those with less than $(N + 1)$ zc's in the input buffer) will be incorrect because the future peak values will be wrong. For this short delay, the average level does not require buffering. We conducted some informal listening tests and found that this limitation on the $P_v$ calculation did not seriously compromise the DDRC's performance.

4.4 Level Measurement

Two signal measurements, the peak level and the average level, are computed by the input section of the DDRC. These levels are adaptively combined as explained in Section 3.3.1. The peak level is used primarily for peak limiting; the average level is used for compression/expansion.

4.4.1 Peak Detector

The peak detector measures the peak level between zc's and stores it in the peak buffer. Zc's are detected as described above. They provide a convenient block size for the determination of peak level. Peak detection between zc's is also convenient because the DDRC attacks instantaneously at zero crossings. The peak detector has an (effective) instantaneous attack because it immediately updates the peak level.

At the output of the peak buffer, the peak output pointer is the address of the peak level between the present and next zc. This "future" peak level allows the DDRC (when the gain is controlled by the peak level) to reduce the gain at the zc before a transient. The output peak level remains constant between zero crossings. Figure 4.4 shows how
the contents of the peak and input buffers are related. The input buffer contains high pass filtered samples from the source and the peak buffer contains peak levels between zero crossings of the input signal.

4.4.2 Average Level

The average level measurement is used as an approximation to the perceived signal level. A first-order IIR filter is used because it is computationally efficient and has no overshoot. An IIR structure provides sufficient averaging in a low order design (Figure 4.5).

Good averaging characteristics require that the time constant of the filter \( \tau \) be much longer than the period of the lowest input frequency. The minimum input frequency \( f_{\text{min}} = 20 \text{Hz} \) has a period of 50ms. Thus, for good averaging we require \( \tau > > 50\text{ms} \).

Cabot [5] reports that "the ear responds more to the power of the signal averaged over some moderate time period (typically one-quarter of a second) than to the average level." This implies that we ought to use \( \tau = 250\text{ms} \). However, for a value of \( \tau \) in this range, the peak and average levels may differ by so much that the transition from average to peak (via level adaptation) will not occur smoothly. To achieve a balance, we will use \( \tau = 100\text{ms} \). This will provide reasonable averaging and should respond to program level changes rapidly so that the transition from average to peak level control will occur smoothly. If required, this value can be refined through listening tests.

McNally [12] showed that the time constant of this filter (Figure 4.5) can be written as

\[
\tau = \frac{T_s \sqrt{1 - a}}{a}
\]

\((T_s \text{ is the sampling period})\) providing

\[
\tau > 2T_s
\]

That is, \( \tau > 2/44100 \approx 45\mu s \)—a condition that will be met. Solving this equation for \( a \)
CHAPTER 4. IMPLEMENTATION

Figure 4.4: Input and Peak Buffer Operation
results in
\[ a = \frac{1}{2} \left( \frac{T_s}{T} \right)^2 \left[ -1 + \left( 1 + 4 \left( \frac{T}{T_s} \right)^2 \right)^{1/2} \right] \]

For \( T_s = 1/44100 \) and \( \tau = 0.100 \) we find \( a = 2.2673316 \times 10^{-4} \). This value of \( a \) is used in the adaptive DDRC implementation. \( a \) is written into RAM so the value may be easily changed to meet different requirements.

### 4.5 Level Adaptation

As described previously, the adaptive level section combines the peak and average level measurements so (1) the peak output level is always below the saturation level of the channel and (2) the gain signal is better related to the perceived signal level (whenever possible). Figure 4.6 shows (in block diagram form) how this scheme (described in Section 3.3.1) is implemented in the adaptive DDRC.

Recall, that below the average control threshold \( V_{ae} \), the average signal level is used for gain control. Above the peak control threshold \( V_{pc} \), the peak level (detected between \( zc \)'s) is used for gain control. Between these two thresholds, a linear combination of the
Figure 4.6: Block Diagram of Adaptive Level Scheme

peak and average level control the gain. The sum of $A$ and $B$ is always less than or equal to 1.0—this ensures that the adaptive level output is always fractional. The equations for the adaptation are

- $p[k] < V'_{ac} \Rightarrow A = 1 - B = 0, B = 1$
- $V'_{ac} \leq p[k] \leq V'_{pc} \Rightarrow A = 1 - B, B = (1/\Delta)(p[k] - V'_{pc})$
- $p[k] > V'_{pc} \Rightarrow A = 1 - B, B = 0$

where $\Delta = V'_ac - V'_pc$, $V'_ac = 10V_{ac}/20$ and $V'_pc = 10V_{pc}/20$ ($V_{ac}$ and $V_{pc}$ are both specified in dB).

To implement these equations on the DSP56000, all values must be fractional. Both $V'_{ac}$ and $V'_{pc}$ will be fractional since $V_{pc}, V_{ac} \leq 0$ dB. $1/\Delta$ will be non-fractional, so it must be scaled. A maximum value of $\Delta$ must be found so a scaling factor ($2^i$) can be determined to make $\Delta/2^i \leq 1.0$. If we choose $\epsilon = V_{pc} - V_{rc} > 10$ dB and $V_{pc} > -50$ dB a scaling factor of $2^8 = 512$ can be used (Appendix C). These restrictions on $\epsilon$ and $V_{pc}$ are required to ensure that there is not an unnecessary loss of accuracy in the adaptive level calculations. This scaling factor gives results that are accurate to 14 bits. This is adequate for these computations.
CHAPTER 4. IMPLEMENTATION

To increase the speed of these computations, pre-computed (and scaled) values of $1/\Delta$ and $V_{pc}/\Delta$ are used. Also, since we always have $A = 1 - B$, only $B$ must be computed. Whenever possible, variables used in sequence are ordered in memory and fetched via parallel updated pointers. The entire level adaptation algorithm is implemented as shown if Figure 4.7.

The values of $V_{pc}'$ and $V_{ac}'$ are user adjustable since they are written into RAM. Typically, $V_{pc}$ will be set equal to the limit threshold ($L_{th}$) so that the DDRC will use the peak level for limiting. $V_{ac}$ will be set 10-15dB below $V_{pc}$ to ensure a smooth transition from average to peak level gain control.

4.6 Dynamic Characteristics

As described in Section 3.2.1, the recovery time ($T_r$) of the adaptive DDRC is controlled by the crest factor ($C_f$) and peak variation ($P_v$) of the input signal. $T_r$ is adapted to achieve low modulation distortion and avoid causing "holes" in the musical program. $T_r$ will be in the range $50\text{ms} \leq T_r \leq 200\text{ms}$. Our design uses a first order exponential recovery characteristic. Gain attack occurs instantaneously at the zero crossing preceding a transient.

4.6.1 Recovery Time Adaptation

The adaptive DDRC design uses a first order recovery filter (Figure 4.8). By considering the unit sample response, it can be shown that the relationship between $T_r$ and $a$ (the filter parameter) is

$$a = \exp \left( -\frac{1}{T_r/T_s + 1} \right)$$

where $T_s$ is the sample period. (Note: This could have also been solved using the relationship derived by McNally [12] as in Section 4.4.2.) Figure 4.9 shows a graph of this
CHAPTER 4. IMPLEMENTATION

Figure 4.7: Flow Chart of Adaptive Level Implementation
relationship. Notice that there is a very small range of \( a \) over which \( T_r \) can adapt within the range \( 50\, ms < T_r < 200\, ms \). Also, notice that increasing \( a \) implies a longer \( T_r \).

As described earlier, the filter parameter \( (a) \) is adapted using

\[
a = k_1 C_f - k_2 P_v + c
\]
\( P_v \) is computed using

\[
P_v = p[k] - \frac{1}{5} \sum_{n=1}^{5} p[k+n]
\]

where \( p[k] \) is the present peak value between zero crossings. The value \( N = 5 \) is a compromise. We want to measure the average future peak level so that the DDRC will adapt based on a future trend—not just one or two future peak levels. However, \( N \) must not be too large or isolated future peak levels will not affect the average at all. \( N = 5 \) provides a good compromise and allows the \( P_v \) calculation to execute in a reasonable number of clock cycles. The multiply and accumulate operation (\textit{mac}) of the DSP56000 is used in conjunction with parallel pointer updates to "speed up" the \( P_v \) computation.

For the real-time implementation, we found that \( C_f^{-1} \) took too long to calculate because it required a division. \( C_f^{-1} \) was replaced with a quantity we named \textit{crest difference} (\( C_d \)). \( C_d \) provides similar information to \( C_f^{-1} \), but it can be computed much faster:

\[
C_f^{-1} = \frac{\bar{x}}{p[k]} = 1 + \frac{\bar{x} - p[k]}{p[k]}
\]

\[
C_d = 1 + \bar{x} - p[k]
\]

We also found (in later experiments) that the peak level between zc's was not always greater than the average signal level—this should have been apparent earlier! Also, the peak level between zc's provides no information about the previous peak level used for gain control. This information is important when deciding how to adapt \( T_r \). For these reasons, the peak level between zc's (\( p[k] \)) was replaced by the recovered output level (\( l[k] \)) in the \( C_d \) calculation. Now we have,

\[
C_d = 1 - l[k] + \bar{x}
\]

\( l[k] \) is always greater than or equal to \( \bar{x} \) and it is always fractional. Thus, \( C_d \) is always fractional.
Parameter Selection

The parameters for the adaptation equation

\[ a = k_1 C_d - k_2 P_v + c \]

must be chosen to keep \( a \) within the desired range of recovery times. Let the threshold that determines whether \( T_r \) is adaptive or constant be \( Q \). Then, \( T_r \) is constant for \( C_d > Q \) and adaptive for \( C_d < Q \).

\( Q \) is set so that \( T_r \) will adapt when the DDRC goes into peak limiting. Thus, \( T_r \) should adapt when the peak level is greater than the peak control threshold (i.e. \( p[k] > V_{pc} \)). As described previously, we set \( V_{pc} = L_{th} = -15dB \). Typically, the peak level of a signal is 10-15dB greater than its RMS level [7]. If we consider that average and RMS level are roughly equivalent, the RMS level can be used in the \( C_d \) calculation in place of the average signal level. Thus, for a typical signal we expect

\[ 1 - 10^{\frac{L_{th}}{20}} + 10^{\frac{L_{th}-10}{20}} \leq C_d \leq 1 - 10^{\frac{L_{th}}{20}} + 10^{\frac{L_{th}-10}{20}} \]

\[ 0.854 \leq C_d \leq 0.878 \]

when \( p[k] > V_{pc} \). For our design we selected \( Q = 0.86 \). The adaptation equation is only used for \( 0 \leq Q \leq 0.86 \). For \( 0.86 < Q < 1.0 \), a constant value of \( a=0.999886641 \) gives a constant \( T_r = 200ms \).

The values for the other adaptation parameters are determined by considering the conditions that will lead to the maximum and minimum values of \( T_r \). When \( T_r \) is adaptive

\[ a = k_1 C_d - k_2 P_v + c \]

where \( 0 \leq C_d \leq 0.86, -1.0 \leq P_v \leq 1.0 \).

\( T_r \) should be a minimum (\( \Rightarrow a \) is minimum) when \( C_d = 0 \) (i.e. \( I[k] >> \bar{z} \)) and \( P_v = 1.0 \) (i.e. the current peak is isolated). \( T_r \) should be a maximum (\( \Rightarrow a \) is maximum) when
$C_d = Q = 0.86$ (i.e. $T_r$ has just become adaptive) and $P_v = -1.0$ (i.e. future peaks are large). Thus,

$$\begin{align*}
min(a) &= k_1(0) - k_2(1.0) + c \\
&= -k_2 + c \\
max(a) &= k_1(0.86) - k_2(-1.0) + c
\end{align*}$$

(4.1) (4.2)

where \(min(a) = 0.999546794\) (\(T_r = 50\text{ms}\)) and \(max(a) = 0.999886641\) (\(T_r = 200\text{ms}\)).

Solving equation 4.1 for \(c\) and substituting it into equation 4.2 we find

$$max(a) - min(a) = 0.86k_1 + 2k_2$$

We have three variables and only two equations, so one parameter must be chosen. If we select \(k_1 = k_2\), \(P_v\) will have twice as much influence on the adaptation as \(C_d\) does since it has roughly twice the range of \(C_d\). Making this substitution (i.e. \(k_1 = k_2 = k\)), results in

$$k = \frac{max(a) - min(a)}{2.86} = 1.18827 \times 10^{-4}$$

These values of \(k_1 = k\) and \(k_2 = k\) will serve a starting points and can be refined via listening tests. Substituting this value of \(k_2\) into equation 4.1, results in

$$c = min(a) + k_2 = 0.999665621$$

4.6.2 Attack/Recovery Implementation

The first order recovery filter used in the adaptive DDRC is a peak hold type. That is, the output \((l[k])\) is controlled such that the larger of the adaptive level \((x[k])\) or the recovered output level \(a[l[k-1]]\) is written to the output. Since the peak level is updated at \(z_c\)'s, gain attack occurs instantaneously at \(z_c\)'s. Figure 4.10 shows a block diagram of the attack/recovery section (including $T_r$ adaptation) design used in the real-time implementation of the adaptive DDRC.
4.7 Static Characteristics

As discussed previously (Section 2.3), the static characteristics specify the steady state performance of the DDRC. They describe the instantaneous relationship between the input level and the DDRC gain (both in dB). The adaptive DDRC uses four region level independent static characteristics.

There are three basic considerations when implementing static characteristics:

1. Speed: The static characteristics should be computed as rapidly as possible.

2. Accuracy: The gain value should be as accurate as possible.

3. Storage: The computations should use as little memory as possible.

The computation of the static characteristics can be split into four steps: (1) determine the input region (i.e. expansion, no-action, compression or limiting) (2) compute the input level in dB (3) perform the gain calculation and (4) convert the gain from dB to a linear...
value. Using the equations presented in Section 2.3, these computations are implemented as shown in Figure 4.11.

The computation of the static characteristics is now reduced to real-time calculation of logarithms and exponents. Since the these functions must be computed on a binary processor (the DSP56000), base two logarithms and exponents can be used to reduce the computation time. To increase the accuracy provided by low-order approximations, a previous design used binary floating point (BFP) arithmetic [12].

BFP arithmetic uses an approximation between 0.5 and 1.0 (i.e. \(2^{-1}\) and \(2^{0}\)) along with normalization/denormalization to compute a base two logarithm or exponent. The logarithm of any number \(x\) can be written as

\[
\log_2 x = \log_2 m + E
\]

where \(E\) is the exponent and \(m\) is the mantissa. Assuming \(0 \leq x \leq 1.0\) is a binary number\(^1\), \(E\) is the number of left shifts necessary to set the most significant bit (MSB) of \(x\). This procedure is known as normalization. The normalized value is \(m\), \(0.5 \leq m \leq 1.0 - 2^{-23}\). \(\log_2(m)\) is found using an approximation.

The computation of the \(2^x\) is similar in nature. If \(y = \log_2(m) + E\) where \(m\) and \(E\) are defined as above and \(-1.0 \leq f = \log_2 m \leq 0\) (since \(0.5 \leq m \leq 1.0\)) and \(E\) is a negative number.

\(^1\)The DSP56000 uses 24 bit twos complement fractional values so \(0 \leq x \leq 1 - 2^{-23}\)
integer, then

\[ z = 2^t \]

\[ = (2^t)(2^E) \]

\(2^t\) is found via an approximation. \(2^E\) is simply a right shift for a binary number. The right-shifting operation *denormalizes* \(2^t\).

BFP arithmetic provides better accuracy for low order approximations because a smaller range is approximated. For the computation of \(\log_2(x)\) this approach greatly increases the accuracy because of the large slope as \(x \to 0\). The disadvantage of this approach is the normalization/denormalization operation. Two clock cycles per bit are required to implement bit-shifting in software.

The accuracy required for the gain calculations is difficult to specify. The input samples (from the source) are 16-bits. This provides an upper limit on the accuracy. The lower limit is determined by subjective analysis. McNally [12] states that coarse errors in the static characteristics may be subjectively acceptable "if gain changes are sufficiently small and occur infrequently." However, if short attack and recovery times are used, rapid gain changes may generate *zipper noise*—a subjectively displeasing effect caused by a fast changing, severely quantized gain value.

McNally used a \(256 \times 8\) bit lookup table for the mantissa in his \(\log_2(m)\) calculations. BFP arithmetic was used and the exponent was limited to \(0 \leq E \leq 7\) to save computation time. To achieve good subjective results, he employed an adaptive filtering scheme to smooth "cogs" in the gain signal and reduce zipper noise. We will consider 8 bit accuracy for the mantissa as our lower limit. Our design will aim for an accuracy between 8 and 16 bits in the mantissa computation. If necessary, McNally's adaptive smoothing scheme can be used to reduce zipper noise.

Two basic methods, lookup tables and polynomial approximations, were examined for the real-time calculation of base two logarithms and exponents. In all approximations
considered, the peak error is minimized. Two basic lookup table types are examined in this section, uniform tables and non-uniform tables. Although interpolated lookup tables are not presented here, they were also under consideration for real-time implementation. They were not used because they required too much computation time. Chebyshev polynomial approximations were also examined.

4.7.1 Uniform Lookup Tables

Lookup tables sequentially store the value of a function at a number of sample points across the (BFP) input range. Lookup table approximation error can be attributed to two sources:

1. **Quantization Error**: error resulting from the finite precision of the samples stored in the table (i.e. The table uses fixed point entries).

2. **Interpolation Error**: error caused by the finite number of table entries.

These two sources of error are dependent: enough precision must be used so that adjacent table entries are distinct. Otherwise the effective table size will be reduced by overlapping entries. The DSP56000 uses 24 bit fractional twos complement arithmetic so the total error is bounded by the worse case interpolation error for any look-up table.

A lookup table can be considered as a quantizer. The input is split into ranges separated by decision levels, $y[k]$. In each range, a representation level, $z[k]$, is selected to represent all values within the range. The value $q[k] = Q[f(z[k])]$\(^2\) is stored in the look-up table. For our application, the input 24 bit binary value. This value can be considered continuous when compared to the interpolation error introduced for the table sizes under consideration.

\(^2\)\(Q[\cdot]\) represents the quantization operation.
TABLE 4.1: Peak Error for BFP Look-up Tables

<table>
<thead>
<tr>
<th>Table Size</th>
<th>$\log_2(z)$ Error</th>
<th>$2^z$ Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>$2.815 \times 10^{-3}$</td>
<td>$1.353 \times 10^{-3}$</td>
</tr>
<tr>
<td>512</td>
<td>$1.408 \times 10^{-3}$</td>
<td>$6.767 \times 10^{-4}$</td>
</tr>
<tr>
<td>1024</td>
<td>$7.043 \times 10^{-4}$</td>
<td>$3.384 \times 10^{-4}$</td>
</tr>
<tr>
<td>2048</td>
<td>$3.523 \times 10^{-4}$</td>
<td>$1.692 \times 10^{-4}$</td>
</tr>
</tbody>
</table>

To generate an output value ($z$), for an input ($x$) the decision rule

$$z = Q[f(x[k])] = q[k]$$

iff $y[k-1] \leq z \leq y[k]; 0 \leq k \leq N$ is used. Considering lookup tables as quantizers provides a flexible analysis framework because no restrictions are placed on the location of representation or decision levels.

For a uniform BFP lookup table, $z[k]$ and $y[k]$ are uniformly spaced. The input value is a fractional fixed-point mantissa ($0.5 \leq m \leq 1.0$). Since $\log_2(z)$ is a monotonic non-decreasing function for $0 < z \leq 1.0$, the worst case peak error occurs in the interval where the slope is greatest. Thus, the error for the $\log_2(z)$ look-up table is

$$\epsilon = |\log_2 y[0] - \log_2 \left( y[0] \left( 1 - \frac{1}{2^N} \right) + \frac{y[N]}{2^N} \right) |$$

$$\epsilon = |1 - \log_2 \left( 0.5 \left( 1 - \frac{1}{2^N} \right) + \frac{1}{2^N} \right) |$$

Table 4.1 shows the peak error for a $\log_2(z)$ BFP lookup table.

By a similar analysis, the peak error for a $2^z$ BFP look-up table is

$$\epsilon = |2^y[N] - 2^z[N]|$$

where $y[N] = 0$ and $z[N] = y[0]/2^N = \log_2(0.5)/2^N$. Thus,

$$\epsilon = |1 - 2^{-1/2^N}|$$  \hfill (4.3)
Table 4.1 shows the error for some typical table sizes.

For both tables, the BFP number representation causes the error to repeat in powers of two. Within each interval (i.e. power of two) the error increases as the input moves from the low slope to high slope end of the interval. For both table types, doubling the table size halves the peak error (i.e. the accuracy increases by one bit). The error is highly non-linear across and within decision intervals. For the $2^z$ lookup table, quantization error makes a significant contribution to the error at the lower extreme of the input range. However, the error does not exceed the bound set by Equation 4.3.

4.7.2 Non-uniform Lookup Tables

Nonuniform tables have nonuniformly spaced decision and/or representation levels. These tables are designed to some optimization criterion. We will not consider tables with non-uniform decision and representation levels because they are difficult to address in real-time. However, tables with only non-uniform representation levels will be examined because they are addressed as uniform tables. Since peak error was selected as the error metric—tables with non-uniform representation levels that minimize peak error will be examined.

Both $\log_2(z)$ and $2^z$ are monotonic non-decreasing functions. Thus, the maximum peak error within any decision interval may be minimized by selecting

$$z[k] = f^{-1}\left(\frac{f(y[k]) + f(y[k-1])}{2}\right)$$

For both $2^z$ and $\log_2(z)$ lookup tables, the worst case error occurs where the slope of the function is the greatest. Thus, for $\log_2(z)$ the worst case error is

$$\epsilon_1 = \left|f(y[0]) - f(x[1])\right|$$

$$= \left|f(y[0]) - \frac{f(y[k]) + f(y[k-1])}{2}\right|$$

$$= \left|\frac{f(y[0]) - f(x[1])}{2}\right|$$
CHAPTER 4. IMPLEMENTATION

<table>
<thead>
<tr>
<th>Table Size</th>
<th>( \log_2(z) ) Error</th>
<th>( 2^x ) Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>( 2.8123 \times 10^{-3} )</td>
<td>( 1.3520 \times 10^{-3} )</td>
</tr>
<tr>
<td>512</td>
<td>( 1.4075 \times 10^{-3} )</td>
<td>( 6.7644 \times 10^{-4} )</td>
</tr>
<tr>
<td>1024</td>
<td>( 7.0410 \times 10^{-4} )</td>
<td>( 3.3833 \times 10^{-4} )</td>
</tr>
<tr>
<td>2048</td>
<td>( 3.5213 \times 10^{-4} )</td>
<td>( 1.6920 \times 10^{-4} )</td>
</tr>
</tbody>
</table>

Table 4.2: Error for Non-Uniform Look-up Tables for \( \log_2(z) \) and \( 2^x \)

Notice also that \( \epsilon_1 = |f(y[1]) - f(x[1])| \) since the peak error is equalized across the interval. For \( 2^x \), a similar analysis show that the maximum peak error is

\[
\epsilon_N = \frac{|f(y[N]) - f(y[N-1])|}{2}
\]

Again we may also write \( \epsilon_N = |f(y[1]) - f(x[1])| \) because the peak error is the same at each end of the interval. Table 4.2 shows the maximum peak error for BFP \( \log_2(z) \) and \( 2^x \) tables.

For a non-uniform \( \log_2(z) \) BFP table, there is not a significant improvement in the error when compared to the same size uniform table. The non-uniform \( 2^x \) BFP table shows only a very small reduction (approximately 1.5%) in error compared to the uniform table. If tables are selected for the final design a non-uniform (representation level) table should be used because it provides slightly lower peak error than a uniform (representation level) table. Both of these tables are simple to address.

4.7.3 Polynomial Approximations

There are many methods available for approximating an arbitrary function with a polynomial. However, a majority of these methods require that \( n + 1 \) samples be used to obtain an \( n^{th} \)-order approximation. This form of a solution matches exactly at the \( n \) sample points, but may deviate far from a best-fit solution at points between the samples.
CHAPTER 4. IMPLEMENTATION

This behavior is especially evident for high-order approximations.

For the approximation of \( \log_2(x) \) and \( 2^x \) we want a method that can approximate a large number of samples using a low-order polynomial and minimize the peak error. This can be accomplished via a minimax approximation—an approximation that minimizes the maximum error. This can be shown to be equivalent to the Chebyshev Criterion:

Of all polynomials, \( p_n(x) \) of degree \( n \), having a leading coefficient equal to 1, the polynomial

\[
\frac{T_n(x)}{2^{n-1}}
\]

has the smallest least upper bound for its absolute value in the interval \(-1 \leq x \leq 1\) (where \( T_n(x) \) is a Chebyshev polynomial) [8].

Note that for cases where the approximation will be for \( x \geq 0 \), it is necessary to work with the shifted Chebyshev polynomials:

\[
T^*_n(x) = T_n(2x - 1)
\]

which are defined over the range \( 0 \leq x \leq 1.0 \). Chebyshev polynomials can be also be stretched to provide approximations over arbitrary ranges greater than \( 0 \leq x \leq 1.0 \) or \(-1 \leq x \leq 1 \). Normally the calculation of Chebyshev polynomial coefficients is a complicated procedure [8] that involves the evaluation of a number of Fast Fourier Transforms. However, a powerful symbolic algebra package (MAPLE\(^3\)) was used to simplify this process.

MAPLE was used to find the minimum-order Chebyshev approximation to each function for a specified accuracy. Approximations over the BFP input ranges: \( 0.5 \leq x \leq 1.0 \) for \( \log_2(x) \) and \(-1.0 \leq y \leq 0 \) for \( 2^x \) were found. Table 4.3 shows the peak error for \( 2^{nd} \), \( 3^{rd} \), and \( 4^{th} \)-order Chebyshev approximations. The coefficients for these approximations

\(^3\)A program developed at the University of Waterloo.
CHAPTER 4. IMPLEMENTATION

<table>
<thead>
<tr>
<th>Order</th>
<th>( \log_2(z) )</th>
<th>( 2^x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>( 5.5830 \times 10^{-3} )</td>
<td>( 1.2628 \times 10^{-3} )</td>
</tr>
<tr>
<td>3</td>
<td>( 7.2527 \times 10^{-4} )</td>
<td>( 5.4737 \times 10^{-5} )</td>
</tr>
<tr>
<td>4</td>
<td>( 1.0019 \times 10^{-4} )</td>
<td>( 1.8895 \times 10^{-6} )</td>
</tr>
</tbody>
</table>

Table 4.3: Peak Error for BFP Chebyshev Polynomial Approximations

are tabulated in Appendix D. To give an idea of the nature of these curve fits, Figure 4.12 shows plots of the BFP approximations to \( \log_2(z) \).

The 3\(^{rd}\)- and 4\(^{th}\)-order BFP approximations give acceptable results—for \( \log_2(z) \) the 3\(^{rd}\)-order approximation gives 10.5-bits accuracy while the 4\(^{th}\)-order approximation gives 13-bits accuracy. Notice that the approximations are not exactly equiripple because \( \log_2(z) \) is non-linear.

For \( 2^x \), the 3\(^{rd}\)-order approximation provides 14-bits accuracy and the 4\(^{th}\)-order curve fit gives 19-bits of accuracy. The 2\(^{x}\) approximations are closer to equiripple because of

![Figure 4.12: Error for Chebyshev Approximation over BFP Range for \( \log_2(z) \)](image-url)
Figure 4.13: Error for Two Chebyshev Approximations to $\log_2(x)$ for BFP range

the lower variation in slope over the input range. These results are excellent.

To improve the results for the $3^{rd}$-order approximation to $\log_2(x)$, two separate approximations can be applied across the input range. Figure 4.13 shows a plot of the error for such an approximation. The "break point" in the approximation was chosen to provide roughly equiripple error with $3^{rd}$-order approximations for each segment of the input range. The coefficients for these approximations are tabulated in Appendix D. This approximation provides a peak error of $5.2125 \times 10^{-5}$ or approximately 14-bits accuracy. An approximation of this type can be implemented with only a small increase in computation time—we only need to determine which range the input lies and specify a pointer to the desired polynomial coefficients.

Clearly, the $3^{rd}$-order BFP Chebyshev approximation to $2^x$ is acceptable. If 10-bits accuracy can be tolerated then a $3^{rd}$-order approximation for $\log_2(x)$ could also be used. If this does not provide sufficient accuracy then two $3^{rd}$-order approximations since they provide better accuracy than the $4^{th}$-order approximation for a roughly equal increase in computation time.
It should be noted that for a fixed point implementation on the DSP56000 the coefficients will be quantized and there will be round-off error in the multiplications, particularly for high powers of \( z \). This may cause slightly poorer results for the polynomial approximations.

### 4.7.4 Real-time Implementation

**Approximation Method**

If the approximation methods are compared for accuracy, we see that they all provide at least 10 bits of accuracy. The 3\(^{rd}\)-order Chebyshev approximation for \( 2^z \) provides 14-bits accuracy. The same range of accuracy (\( \approx 14\)-bits) can be achieved for \( \log_2(z) \) if two 3\(^{rd}\)-order approximations are applied across the BFP input range.

The look-up tables require much more memory than the polynomial approximations. Because it is convenient to use 24-bit entries (DSP56000 word size), look-up tables will use substantial portion of the \( 6k \times 24 \) bit memory available on the (stock) DSP56000ADS. Chebyshev polynomial approximations require very little storage, just \((k+1)24\) bit values for a \( k^{th}\)-order approximation.

Table 4.4 shows the estimated number of clock cycles to compute the approximation given a normalized input value. (De)Normalization is not considered since it is used by all approximations. All possible "tricks" are used to get the best performance from each method.

Clearly, the 3\(^{rd}\)-order BFP Chebyshev polynomial approximation provides that lowest computation time for the desired accuracy. It also uses the least memory. This method was selected for real-time implementation.
Normalization and Denormalization

The normalization and denormalization of input/output values are two distinctly operations: normalization can be generalized as a search operation. We want to find the largest bit that is set in a binary number—that is, we have no prior knowledge of which bit is set. Denormalization is a right shifting an amount specified by the exponent. This operation can be quickly done using a small look-up table and multiplication where the exponent provides the address of the required shift constant.

Numerous search methods may be applied to normalize a number. Two simple algorithms, exhaustive search and binary search were investigated. Theoretically, binary search should execute much faster than exhaustive search: $N/\log(N)$ faster for a search through $N$ elements [16]. However, in practice the additional overhead required for set-up reduces the practical advantage of a binary search over an exhaustive search.

An exhaustive search is easily implemented via bit-shifting. To simplify the implementation, the DSP56000 instruction set provides the norm instruction [15]. A conditional jump instruction (jnn) that redirects program execution depending on the state of the normalization flag in the SR and a low-overhead loop instruction (rep) are also provided.

Two implementations of the exhaustive search were tested: a norm—jnn loop and a rep #23—norm loop. Table 4.5 shows the best case, worst case and average (estimated) execution times for these implementations.

<table>
<thead>
<tr>
<th>BFP Approximation Type</th>
<th>clock cycles</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>best case</td>
<td>worst case</td>
</tr>
<tr>
<td>interpolated table 54 (N=256)</td>
<td>66 (N=2048)</td>
<td></td>
</tr>
<tr>
<td>non-uniform table 24 (N=256)</td>
<td>30 (N=2048)</td>
<td></td>
</tr>
<tr>
<td>3rd-order Chebyshev</td>
<td>26 (two approx.)</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4: Computation Time for Selected Approximation Methods
A binary search uses the divide-and-conquer paradigm. At each iteration the location of the desired element is resolved to half of the previous possible locations. Tests on the DSP56000 showed that the set-up time for each execution of the search was significant. Thus, a combined binary/exhaustive search process was used. A single iteration of the binary search is done to "narrow down" the location of the MSB that is set to within 12-bits. Then, an exhaustive search is applied to find the MSB that is set within these 12-bits. Figure 4.14 shows a flow chart of the algorithm use to implement this search technique. Two methods, a norm—jnn loop and a rep #11—norm loop, were applied to the exhaustive search portion of the algorithm. The execution times for these algorithms are shown in Table 4.5.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Execution Times (clk cyc)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>best</td>
</tr>
<tr>
<td>norm1</td>
<td>norm/jnn loop</td>
<td>8</td>
</tr>
<tr>
<td>norm2</td>
<td>rep#23/norm loop</td>
<td>58</td>
</tr>
<tr>
<td>norm3</td>
<td>binary-norm/jnn loop</td>
<td>34</td>
</tr>
<tr>
<td>norm4</td>
<td>binary-rep#11/norm loop</td>
<td>48</td>
</tr>
</tbody>
</table>

Table 4.5: Execution Times for Search Algorithms on DSP56000

We see that norm4 provides the best performance. It also has the advantage of offering roughly the same execution time for the best and worst cases. This simplifies the analysis of the overall execution time of the DDRC. A complete implementation of the static characteristics is shown in Figure 4.14.
CHAPTER 4. IMPLEMENTATION

Figure 4.14: Flow Chart of Static Characteristics Implementation
4.8 Summary

The adaptive DDRC was implemented in real-time on a Motorola DSP56000ADS. It operates at the standard (Compact Disk) sampling rate of 44.1kHz with 16 bit input and output samples. The design is interrupt driven—the entire DDRC algorithm must execute between two interrupts in $\frac{1}{44100} = 22.67\mu s$ or 464 clock cycles of the DSP56000.

The design uses a 150ms delay to adapt based on future statistics (with respect to the output) of the input signal. Our present design is constrained to 25ms delay because of memory and DSP56000 architecture limitations. For this short delay, the $P_c$ calculation will be in error for very low frequency input signals. This problem is easily solved by purchasing more memory.

Numerous parameters for the design of the input/output section components were derived in this section. A time constant of 100ms was selected for the average level filter. This value is a compromise between the need for good averaging and a reasonable response time to program level changes. The static characteristics are computed using Chebyshev polynomial approximations and BFP arithmetic. To "speed-up" normalization, a combined binary/exhaustive search algorithm was used.
Chapter 5

Testing

5.1 General

This chapter describes the testing and evaluation of the adaptive DDRC. In previous chapters, the variables used to adapt the dynamic characteristics (Table 5.1) and the static characteristics (Table 5.2) of the DDRC have been presented. The values used for the static parameters are typical of those that would be used for a compressor/limiter. The adaptive parameter values were developed theoretically and refined through preliminary (informal) listening tests. Unless specified otherwise, the parameter values shown in these tables will be used for all tests conducted in this chapter.

The tests are split into two basic types: objective and subjective. Objective tests are used to confirm that the DDRC operates as designed. The majority of these tests are conducted using the DSP56000 simulator (SIM56000). This program exactly simulates a DSP56000 and runs the same compiled and linked .lod files as the evaluation board. The simulator can read and write a number of input/output files and provide an exact count of the clock cycles required for execution of the algorithm. At any point, simulations can be stopped and restarted. However, the simulator has the disadvantage of slow operation (e.g. it takes 2.25 hours to process 186ms of audio); this limits the length of tests that
### Table 5.1: Parameters for DDRC Adaptation Control

<table>
<thead>
<tr>
<th>Section</th>
<th>Parameter</th>
<th>Function</th>
<th>Typ. Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level Measurement</td>
<td>a</td>
<td>average level</td>
<td>$(\tau = 100 \text{ms})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>filter parameter</td>
<td>$2.2673316 \times 10^{-4}$</td>
</tr>
<tr>
<td>Recovery Adaptation</td>
<td>$k_1$</td>
<td>$C_d$ weight</td>
<td>$1.18827 \times 10^{-4}$</td>
</tr>
<tr>
<td></td>
<td>$k_2$</td>
<td>$P_v$ weight</td>
<td>$1.18827 \times 10^{-4}$</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>adaptive recovery</td>
<td>0.99966562</td>
</tr>
<tr>
<td></td>
<td></td>
<td>starting point</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a</td>
<td>fixed $T_r$ filter</td>
<td>0.99988664</td>
</tr>
<tr>
<td></td>
<td></td>
<td>parameter</td>
<td>$(T_r = 200 \text{ms})$</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>adaptive $T_r$ threshold</td>
<td>0.86</td>
</tr>
<tr>
<td>Level Adaptation</td>
<td>$V_{pc}$</td>
<td>avg. control threshold</td>
<td>$V_{pc} - 10 \text{dB}$</td>
</tr>
<tr>
<td></td>
<td>$V_{pc}$</td>
<td>peak control threshold</td>
<td>$-15 \text{dB} (=L_{th})$</td>
</tr>
</tbody>
</table>

### Table 5.2: Parameters for Static Characteristics

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
<th>Typ. Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_{th}$</td>
<td>limit threshold</td>
<td>-15dB</td>
</tr>
<tr>
<td>$C_{th}$</td>
<td>compression threshold</td>
<td>-35dB</td>
</tr>
<tr>
<td>$E_{th}$</td>
<td>expansion threshold</td>
<td>-50dB</td>
</tr>
<tr>
<td>$R_L$</td>
<td>limit ratio</td>
<td>1/100</td>
</tr>
<tr>
<td>$R_C$</td>
<td>compression ratio</td>
<td>1/3</td>
</tr>
<tr>
<td>$R_E$</td>
<td>expansion ratio</td>
<td>2/1</td>
</tr>
</tbody>
</table>

---

**CHAPTER 5. TESTING**

73
can be conducted on the simulator.

Subjective or listening tests were conducted to determine the sonic performance of the adaptive DDRC relative to (almost) conventional designs. Through specific combinations of parameters, the adaptive DDRC can be configured to perform like conventional (non-adaptive) designs. The subjective tests will also show whether any subjectively displeasing artifacts are introduced by the adaptation. In these tests, no attempt will be made to arrive at the optimum set of adaptation parameters since this evaluation is beyond the scope of the present research.

5.1.1 Test Set-up

Simulator: Input files containing fractional samples (i.e. \(-1 < z[n] < 1\)) are generated with the desired waveforms using programs written in C. These are "ideal" signals containing no noise and having precisely controlled characteristics. The simulator (SIM56000) executes the DDRC algorithm and reads these input files. Output files of specified variables (e.g. the recovered signal level) are generated.

Development System: The test set up for real-time tests is shown in Figure 5.1. An adjustable gain mixer (Appendix E) was constructed to mix the left/right outputs of the CD player and provide a balanced input to the A/D converter. For a majority of the tests, a test CD (Denon Audio Technical CD #38C39-7147) was used. This CD provides standard test signals for distortion measurements. Sinusoids at various input levels (including 0dB peak level) are also provided. These signals proved invaluable for setting the input level to the A/D (i.e. the gain of the two channel mixer). It is imperative that the input level to the A/D be properly adjusted so that the DDRC receives the maximum input dynamic range.
5.2 Objective Tests

5.2.1 Execution Time

As discussed previously, for the adaptive DDRC to operate in real-time, the algorithm must execute within one sample period \( T_s \) (where \( T_s \approx 22.67\mu s \)). The ADS56000 operates at a base clock rate of 20.48MHz. Thus, there are 464 clock cycles available to execute the entire algorithm. Table 5.3 shows the number of clock cycles required for each component of the algorithm and the percentage for each component expressed relative to the maximum execution time of 464 cycles. All features of the DSP56000 (i.e. parallel pointer updating, parallel data fetching and pointer addressing) are used to speed execution.

From this table, we see that the worst case path through the algorithm requires more time than is available. The worst case path through the algorithm will be followed rarely (possibly never). Some quick listening tests confirmed that the DDRC operated properly. There was no evidence of blocks of samples being skipped because the algorithm did not
<table>
<thead>
<tr>
<th>Section</th>
<th>Component</th>
<th># clock cycles</th>
<th>% total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>interrupts and I/O</td>
<td>16</td>
<td>3.4</td>
</tr>
<tr>
<td></td>
<td>HPF/setup pointers</td>
<td>26</td>
<td>5.6</td>
</tr>
<tr>
<td></td>
<td>peak/zc detect</td>
<td>50</td>
<td>10.8</td>
</tr>
<tr>
<td></td>
<td>avg. level</td>
<td>12</td>
<td>2.6</td>
</tr>
<tr>
<td>Output</td>
<td>zc detect</td>
<td>18</td>
<td>3.9</td>
</tr>
<tr>
<td></td>
<td>adaptive level</td>
<td>56</td>
<td>12.1</td>
</tr>
<tr>
<td></td>
<td>attack/recovery</td>
<td>48</td>
<td>10.3</td>
</tr>
<tr>
<td></td>
<td>adaptive recovery</td>
<td>66</td>
<td>14.2</td>
</tr>
<tr>
<td></td>
<td>region</td>
<td>28</td>
<td>6.0</td>
</tr>
<tr>
<td></td>
<td>normalize</td>
<td>54</td>
<td>11.6</td>
</tr>
<tr>
<td></td>
<td>log2()</td>
<td>16</td>
<td>3.4</td>
</tr>
<tr>
<td></td>
<td>compute gain</td>
<td>64</td>
<td>13.8</td>
</tr>
<tr>
<td></td>
<td>exp2()</td>
<td>16</td>
<td>3.4</td>
</tr>
<tr>
<td></td>
<td>denormalize</td>
<td>12</td>
<td>2.6</td>
</tr>
<tr>
<td></td>
<td>output sample</td>
<td>4</td>
<td>0.9</td>
</tr>
<tr>
<td>Totals</td>
<td></td>
<td>486</td>
<td>104.7</td>
</tr>
</tbody>
</table>

Table 5.3: Execution Times for DDRC Algorithm Components
complete between interrupts. We could leave the algorithm as is. However, to ensure a reliable design, the HPF filter was eliminated from the input stage. This reduces the worst case execution time to 464 cycles. As discussed previously, this may have some adverse affects on the sonic quality of the DDRC if the input has a DC offset.

5.2.2 Static Characteristics

Tests on the static characteristics were conducted to determine the accuracy of the gain computation. This computation includes the normalization of the input value, the computation of polynomial approximations, calculation of the static characteristics and the denormalization of the result.

Initially, tests were conducted in real-time on the ADS56000 using the test setup shown in Figure 5.1 except a 1kHz input from a signal generator was used in place of the CD player signal source. A test program that reads an input sample and writes it to the output (intioevb.asm) was run of the ADS56000. The input level was set so that the output was just below (digital) clipping. At this point, the input level to the A/D was 1.596Vrms (measured differentially across the A/D input with a floating DVM). The output level was 1.042Vrms. These levels are the 0dB reference voltages for the input and output.

To test the static characteristics the adaptive DDRC program was run. The input level was varied and the output level was recorded. The adaptive DDRC was configured so that the adaptive level section was disabled and the peak input level was used for level control. The results of this test are tabulated in Table 5.4 and plotted (along with the ideal static characteristics) in Figure 5.2. These results show that for our test setup, the measured and theoretical static characteristics (over the range we were able to measure) differ by no more than 2dB. Our test setup had severe noise problems, we could not measure below -40dB input level because the signal was “buried” in noise (i.e. the test setup has a noise floor of approximately -45dB). Clearly, the noise introduced into our
CHAPTER 5. TESTING

<table>
<thead>
<tr>
<th>Input Level (dB)</th>
<th>Output Level(dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>-28.18</td>
</tr>
<tr>
<td>-1.02</td>
<td>-28.18</td>
</tr>
<tr>
<td>-1.50</td>
<td>-28.18</td>
</tr>
<tr>
<td>-2.80</td>
<td>-28.18</td>
</tr>
<tr>
<td>-3.78</td>
<td>-28.18</td>
</tr>
<tr>
<td>-5.88</td>
<td>-28.40</td>
</tr>
<tr>
<td>-8.88</td>
<td>-28/40</td>
</tr>
<tr>
<td>-17.25</td>
<td>-29.38</td>
</tr>
<tr>
<td>-22.74</td>
<td>-31.70</td>
</tr>
<tr>
<td>-25.68</td>
<td>-32.76</td>
</tr>
<tr>
<td>-29.90</td>
<td>-34.42</td>
</tr>
<tr>
<td>-36.48</td>
<td>-38.42</td>
</tr>
<tr>
<td>-40.00</td>
<td>-40.18</td>
</tr>
</tbody>
</table>

Table 5.4: Measured Static Characteristics

test setup by the surrounding equipment (computers, power supplies etc.) makes it very difficult to assess the accuracy of the static characteristics.

To obtain more accurate results, tests were conducted using the DSP56000 simulator. The adaptive DDRC was configured as described above and 1024 input levels from approximately -75dB to 0dB were applied to the input (512 uniformly spaced samples from -75dB to -40dB and 512 uniformly spaced samples from -40dB to 0dB). The results of this test are plotted in Figure 5.3. Figure 5.4 shows the error in these gain computations compared to floating point theoretical calculations.

The error plot (Figure 5.4) shows that the static characteristics of the adaptive DDRC differ by less than 0.008dB from the theoretical static characteristics over the usable input
CHAPTER 5. TESTING

Figure 5.2: Measured Static Characteristics

Figure 5.3: Static Characteristics Measured Using Simulator
range (i.e. before the output level underflows 16 bits). Clearly this is accurate enough since no one can perceive an error this small. The error plot shows increasing error as the input level falls below -80dB. This is a result of exponent underflow (i.e. $E < -23$) in the gain calculation. In the no action region of the characteristics, the only error is caused by the use of fixed point arithmetic since the gain is set to $1 - 2^{-23}$. In all other regions of the static characteristics (limiting, expansion and compression), the error is a combination of the polynomial approximation error, fixed point arithmetic error and error introduced by scaling.

5.2.3 Level Measurement

To check the operation of the average level measurement, the step response of the first-order averaging filter was measured. A unit step that changed from 0V to 1.0V at the first sample was used as the input. The output voltage at the 4091st sample (for a 4096-point

Figure 5.4: Error for Static Characteristics
input step) was \( V_{\text{out}}[4091] = 0.6043706 \). Thus, the time constant of the filter is

\[
\tau = \frac{-4091(T_s)}{\ln(1 - 0.6043706)} \approx 100 \text{ms}
\]

This is as designed.

The peak level detector was tested by applying a 1kHz, 4096-point, 0.25V peak level sinusoid with some modified peak levels (Figure 5.5) to the input of the adaptive DDRC. Figure 5.6 shows the output peak level for a portion of the input waveform where the peak input levels have been altered. Notice that all peak levels are greater than \( V_{pc} \). (Recall that \( V_{pc} = -15\text{dB} \) or 0.17783 volts.) This causes the input signal peak level to be used for gain control.

Clearly, the peak detector operates as desired: the peak level is updated at the zero crossing preceding the peak. During attack, the peak level is constant between zero crossings (so that peaks are properly suppressed). Recovery begins once the present peak level falls below the recovered signal level.

A simple test was also conducted to confirm the operation of the adaptive level mea-
surement. Using the simulator, the characteristics shown in Figure 3.3 were reproduced within the errors introduced by scaling and fixed point arithmetic (Section 4.5).

5.2.4 Dynamic Characteristics

Tests were also conducted to confirm the operation of the attack and (adaptive) recovery times. Typically, these tests are conducted using sinusoidal tone bursts with level changes at zero crossings [6]. However, a waveform of this type does not demonstrate the instantaneous attack capabilities of the adaptive DDRC (i.e. the gain changes at the zero crossings before the transient).

The dynamic characteristics of the adaptive DDRC operate in three different modes. Let the peak level between the present output zero crossing be $p[k]$. Recall that $\tau$ is the time constant of the filter used to measure the average signal level. Also, let $T_{rf}$ be the recovery time of the adaptive recovery filter. When the adaptive signal level is controlled by the average level (i.e. $p[k] < V_{pc}$), we have $T_a = \tau$ and $T_r = f(\tau, T_{rf})$. When the adaptive signal level is controlled by the peak level between zero crossings (i.e. $p[k] \geq V_{pc}$), we have $T_a = f(\tau, T_{rf})$ and $T_r = \tau$.
$p[k] > V_{pc}$, $T_a$ is instantaneous and the gain attacks at the zero crossing preceding the transient. For this case, $T_r = T_{rf}$. If $V_{pc} \leq p[k] \leq V_{pc}$ the adaptive signal level is a linear combination of $p[k]$ and the average signal level. Then, $T_a$ and $T_r$ will lie between the two extremes presented above.

We want to test the instantaneous attack time and the adaptive recovery time. Thus, the input signal must be such that $p[k] > V_{pc}$ for all $k$. For these tests, the sinusoidal signal presented above (Figure 5.5) was used. This signal has isolated transients that will demonstrate the unique attack and recovery characteristics of the adaptive DDRC. To obtain a reasonable number of non-zero output samples for a 4096-point input signal, the input and peak buffers were set to 256 samples. This size ensures that the $P_v$ calculation will always be done using "valid" peaks for a 1kHz input waveform (Section 4.3.2).

Figure 5.7 shows the DDRC gain response to the input signal. When the peak level is used to control the gain, the attack is instantaneous and it precedes the transient. The gain is held constant between zero crossings during attack so all peaks between the zero crossings are properly suppressed.
To test the adaptive recovery operation, the peak variation ($P_v$) and crest difference ($C_d$) measurements were examined. Again, the input signal shown in Figure 5.5 was used. Figure 5.8 shows the $P_v$ for this input signal. $P_v$ is constant when the DDRC attacks since it is only computed during recovery. The initial 277 samples of the $P_v$ are zero (256 while the peak buffer empties and 21 while the DDRC attacks). As expected, when the future average peak level is large, $P_v < 0$. For a smaller future average peak level, $P_v > 0$. For a constant peak level (i.e. a periodic signal), $P_v \approx 0$.

The $P_v$ for this input waveform (Figure 5.5) also demonstrates a disadvantage of the simple two-sample zero crossing detection method. For the periodic part of the input signal, we expect $P_v = 0$. However, the sinusoidal input signal has periodically occurring 0V samples. This causes the peak detector to store a peak level of zero. When this peak value of zero is time shifted through the $P_v$ calculation, $P_v = 0.25 - 1.0/5 = 0.05$ when the peak value of zero is in the future. When the present peak level is zero, $P_v = 0 - 1.25/5 = -0.25$. For a single 0V sample, this transient lasts for only $6(T_s) = 136.02\mu s$ which is not long enough to be audible. Thus, this small anomaly will not have any affect on the
sonic performance of the DDRC.

$C_d$ for this input waveform is shown in Figure 5.9. Compared to the recovered signal level (Figure 5.9), the average signal level is small since the simulation is only 90ms long. Thus, $C_d \simeq 1 - l[k]$. We see that where peaks are large, $C_d$ is large ($C_d = 0.75$). Where the peaks are small, $C_d$ is smaller. Since $C_d < 0.86$, $T_r$ is always adaptive.

$T_r$ for this same input waveform (Figure 5.5) is shown in Figure 5.10. $T_r$ was found by post-processing the recovery filter coefficient. The first 277 samples are not shown because they are constant at the initialized value of zero. Clearly, $T_r$ lies within specified range of $50ms < T_r < 200ms$. Also, when the input signal (Figure 5.5) has a high peak level (relative to the average level), $T_r$ is short ($\approx 80ms$). $T_r$ increases as the gain recovers and the signal level ($l[k]$) decays towards the average signal level. Because the level ($l[k]$) will take a minimum of 50ms to recover, $T_r$ does not adapt too quickly. The anomaly in the $P_s$ calculation (discussed above) does not have any serious affect on the recovery time computation.
5.2.5 Distortion Measurements

To perform dynamic range control, the adaptive DDRC uses time-varying non-linear processing. These operations will generate harmonic and intermodulation distortion products. Because, the DDRC performs all calculations in the digital domain, it is possible for these components to alias into the passband of the corresponding lowpass filter. (In an analog system, these components are removed by low pass filtering.) This problem is most severe in the level measurement section where the input is the full bandwidth audio signal [12]. To assess the effects of the distortion on the DDRC performance a harmonic distortion and two intermodulation distortion tests were conducted.

In the past many methods have been developed for the characterization of distortion [6]. Typically, these measures cannot gauge the subjective effects of distortion. However, a distortion analysis based on simple input signals can provide a means of comparison with other devices. Also, by examining the input and output spectra, insight into the nature of the device characteristics can be gained. Also, it can provide a basis for optimizing the parameters of the device under test.
To determine the distortion of the adaptive DDRC, a total harmonic distortion (THD) test and two intermodulation distortion (IMD) tests were conducted. Our original plan was to use test signals provided on the Denon Test CD and perform the tests in real-time. However, our test setup had a noise floor of approximately -45dB. This made it difficult to make accurate measurements. Thus, simulations were used for both tests.

For both tests, peak controlled gain was used. This method of gain control introduces the most distortion since the peak level (peaks between zero crossings) is not as smooth as the average signal level. Also, the output reaches steady state much faster when the peak level is used for gain control—this allows shorter duration simulations. Both simulations were run for 8192 samples. Spectral plots were made using an FFT of the last 4096 samples to ensure that transients (e.g., the buffers emptying at initialization) did not affect the results.

For the THD test, a 1.0V peak (i.e., full-scale input), 1kHz sinusoid was used. Figures 5.11 and 5.12 show the input and output spectra for the THD test. From these results, the THD was computed to be (using Equation 3.1) 0.000031%.
CHAPTER 5. TESTING

For the IMD test, a standard IEC test signal

\[ x(t) = 0.80 \sin(2\pi f_1 t) + 0.20 \sin(2\pi f_2 t) \]

where \( f_1 = 250\, \text{Hz} \) and \( f_2 = 8020\, \text{Hz} \) was used. Figures 5.13 and 5.14 show the input and output spectra for these tests. The IMD was computed (using Equation 3.1) to be 0.000525%.

Both of these tests result in distortion levels that are inaudible. For the THD test, this is expected. The adaptive DDRC should have no steady state distortion for a single sinusoidal input because the device simply computes a gain based on the constant peak level and applies it to the input signal. Since the peak level is constant, the gain is constant and the output is simply the input at a lower level. The small amount of distortion that is present is a result of the inaccuracies in the gain computations and the use of fixed point arithmetic and scaling in the algorithm.

The IMD test results in a higher, but still inaudible distortion level. Again, this low distortion results from the almost constant peak level of the input signal. The small
CHAPTER 5. TESTING

Figure 5.13: Input Spectrum for IMD Test

Figure 5.14: Output Spectrum for IMD Test
amount of distortion can be attributed to the same sources that are responsible for the harmonic distortion.

To obtain results that would better reflect the adaptive DDRC performance to “real world” (musical) input signals, we conducted an additional IMD test. The input signal (Figure 5.15) used was one employed by Mapes-Riordan and Leach [11] in their tests on a digital peak limiter. Their signal was 1024 points long. We concatenated eight such signals (8192 samples long) and applied this as an input to the adaptive DDRC. Figures 5.16 and 5.17 show the input and output spectra for the last 4096 samples of each signal. For these signals, we obtained a distortion level of 0.438%. If all 8192 input and output samples are used in the distortion calculations (i.e. the initialization and gain attack are included in the analysis), a distortion of 0.450% is obtained. The spectra for the 8192 point analysis was very similar to the 4096 point spectra. This test signal results in higher distortion levels than the previous IMD test because it has a larger variation in peak level.

These distortion levels may be audible. For their tests on a digital peak limiter
CHAPTER 5. TESTING

Figure 5.16: Input Spectrum for Second IMD Test

Figure 5.17: Output Spectrum for Second IMD Test
with gain attack and recovery at zero crossings, Mapes-Riordan and Leach [11] report a distortion of 5.4% with $T_r=0$. For long recovery times (on the order of those used in our adaptive DDRC) and a 6dB limiting level, they report distortions of “less than 1%.” These results compare favourably with our results. They also indicate that good subjective results are obtained with these long recovery times.

### 5.3 Subjective Tests

Signal processing may sometimes generate annoying distortion or artifacts that will not be apparent in the objective tests. To uncover these problems, listening tests are usually performed— they are the definitive measurement of performance.

An important requirement of these experiments is one of presenting a totality of stimuli with maximal randomness of order to protect against the possible effects of presentation order and with sufficient redundancy for averaging out variability in subjects decisions ([10] - Appendix F). However, the large number of tests that are required to average out variability make listening tests very time consuming.

For our evaluation of the adaptive DDRC performance, we did not conduct exhaustive testing; instead, seven subjects were used to get an idea of how the performance of the adaptive DDRC compares to conventional DDRC designs. Comparative listening tests were used exclusively: listeners compared processed signals to unprocessed signals and rated their relative quality on a five point opinion scale (Table 5.5).

Preliminary listening tests showed that there was “clicking” present in the attack portion of some processed signals. This artifact was most evident when the average signal level was used for gain control. Clicking was present for both the adaptive and conventional designs. Through a number of tests, we determined that this artifact was not a result of input buffer overflow or (input) dc level modulation [18]. We concluded this artifact was zipper noise [12] caused by the combination of attack at zc's and the rise
CHAPTER 5. TESTING

Score Quality Comments
5 Excellent sounds the same as the original
4 Good compressed, but still of high quality
3 Fair compressed with occasional annoying distortion
2 Poor compressed with annoying distortion
1 Unsatisfactory sound quality is unacceptable

Table 5.5: Five-point Opinion Scale

time of the average level measurement.

When the average signal level is used for gain control, an increase in level results in a exponentially increasing average level measurement. This results in a sequence of gain attacks at zc’s. This is perceived as a sequence of clicks—zipper noise! McNally’s adaptive gain smoothing scheme [12] could be used to eliminate the zipper noise. However, our algorithm currently uses 100% of the available processing time. Thus a faster processor or an improved DDRC algorithm (faster) would be required to allow this addition. Because this artifact has a known cause and cure, listeners were asked to ignore it in their evaluation of the Adaptive DDRC’s sound quality.

Three high-quality test signals (compact disk “DDD” recordings) were selected for use in the listening tests. These selections represent typical wide dynamic range listening material.

- (A) Larry Coryell
  Scheherazade
  1. The Sea and Sindbad’s Ship
  ©1983 Nippon Phonogram Co., Ltd., Tokyo
  (solo guitar)

- (B) Tears for Fears
CHAPTER 5. TESTING

The Seeds of Love
Badman's Song
©1989 Phonogram Ltd., London
(piano, electric guitar and heavy drums)

• (C) W.A. Mozart
Violin Concerto No. 5 in A Major (KV 219)
1. Allegro Aperto
©1988 Enigma Classics
(string ensemble and solo violin)

Seven processing methods that were developed in the preliminary round of testing were used. (Note: Unless specified otherwise, the adaptation parameters in Table 5.1 are used. Recall that \( \tau \) is the time constant of the average level filter).

1. Adaptive DDRC (\( \tau =30\text{ms} \))
2. Adaptive DDRC (\( \tau =100\text{ms} \))
3. Peak Level Controlled Gain
4. Average Level Controlled Gain
5. Adaptive Level and Fixed \( T_r=200\text{ms} \)
6. Adaptive DDRC (Extended \( T_r \) range 50ms to 250ms)
7. No Processing

Each signal was processed using all of the above methods to give a total of 21 test signals. Between 45s and 60s of each unprocessed/processed signal pair were recorded in random order on a high-quality cassette. The use of a cassette slightly degrades the unprocessed examples. However, all testing is relative so this minor degradation is not of
any consequence. In our opinion, the recorded versions of the processed signals suffered very little degradation.

Listeners were given the following list of terms to use when commenting on performance they felt was degraded:

- **holes**: the program is momentarily blanked out following a transient
- **muddy**: the program is noticeably modulated by its level (IM distortion)
- **dense**: the output program is heavily compressed. The original balance of the program components is lost or degraded.
- **strident**: the timbre of the program is altered. Some instruments sound "rough" or too "bright."
- **fuzzy**: harmonic distortion introduces additional harmonics

Listeners were allowed to use their own descriptive terms provided they explained them.

Seven listeners were used to obtain the results shown in Table 5.6. To obtain the mean opinion score results, all scores for each processing option were averaged. The raw data for this test is contained in Appendix F.

As expected, listeners judged the unprocessed signal to be of the highest (relative) quality. Signals compressed using the average signal level were judged to be of the next highest quality. This was also expected because average level gain control is better related to the perceived signal level. Thus, the signal is not as heavily compressed. The adaptive DDRC with $\tau = 30\,\text{ms}$ was ranked third in quality by the listeners. The subjects preferred this processing to the peak gain controlled example and either of the other adaptive examples (extended $T_r$ and $\tau=100\,\text{ms}$). The fixed $T_r$ example was judged to be the worst quality.

These results indicate that the adaptive DDRC has the potential for better performance than conventional designs that use only the peak signal level for gain control. The
Table 5.6: Mean Opinion Scores

<table>
<thead>
<tr>
<th>Test</th>
<th>Processing</th>
<th>Score</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Adapt $\tau=30\text{ms}$</td>
<td>4.00</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Adapt $\tau=100\text{ms}$</td>
<td>3.69</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>Peak Control</td>
<td>3.83</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>Average Control</td>
<td>4.38</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>Adapt (fixed $T_r$)</td>
<td>3.50</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>Adapt (extend $T_r$)</td>
<td>3.81</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>None</td>
<td>4.95</td>
<td>1</td>
</tr>
</tbody>
</table>

The comments made by listeners also provide some insight into the quality of the processed signals. Listeners reported that $\approx 29\%$ of the peak controlled gain processed signals contained holes. However, the other processed signals were reported as having roughly equivalent "hole counts" ($\approx 14\%$). This means that either listeners were not reporting all the holes (some listeners did not indicate hearing any) or that the $T_r$ adaptation algorithm (parameters) require refinement. When signals were rated as being low quality, they were usually reported as being strident or muddy. This results from poor control of the recovery time. These results also indicate that further tests are required to improve the adaptation parameter set.
5.4 Summary

The results for the subjective tests indicate that the Adaptive DDRC operates as designed. The Adaptive DDRC has very low total harmonic distortion. The intermodulation distortion results compare favourably with a previous design that was reported to have good subjective performance [11]. Concern expressed about the two sample zero crossing detection method used in the design was unwarranted. Simulations show that this has a negligible effect on performance.

Listening tests were conducted using seven listeners with seven processing methods and three different musical input signals. One Adaptive DDRC example was ranked third in average quality—behind the unprocessed signal and the average level controlled gain example. However, two adaptive DDRC examples were ranked below the peak controlled gain example. Clearly, the adaptive DDRC shows promise for improved performance. However, extensive listening tests are required to arrive at the "best" parameter set.
Conclusions and Recommendations

This thesis describes the design and real-time implementation of an Adaptive DDRC. Three features of this design offer improved performance over previous DDRC designs.

1. Attack at Zero Crossings: The Adaptive DDRC attacks at the zero crossing preceding a transient. This reduces the perceived distortion introduced by gain attacks.

2. Adaptive Recovery Time: The recovery time is adjusted based on two input signal statistics (peak variation and crest difference) to avoid introducing "holes" into the output program, yet achieve low modulation distortion. The recovery time adaptation characteristics are controlled by user adjustable parameters.

3. Adaptive Level Measurement: For peak levels below the average control threshold, the average level controls the gain. When the peak level exceeds the peak control threshold (= limit threshold) the gain is controlled by the peak signal level. Between these thresholds, a linear combination of the peak and average signal levels
is used to provide a smooth transition from peak to average-control (and vice-versa). This arrangement gives improved perceptual performance because the average signal level (which is better related to the perceived signal level than the peak signal level) is used for gain control whenever possible.

Our test results show that the Adaptive DDRC performs well. The total harmonic distortion is insignificant. Intermodulation distortion measurements are comparable to those for another DDRC design [11] that is reported to have good subjective performance. Listening tests using seven listeners, three test signals and seven processing methods were conducted. In these tests, one configuration of the Adaptive DDRC received better mean opinion scores than conventional designs (peak level gain control, non-adaptive recovery time). Although these tests cannot be considered exhaustive, they do indicate that the Adaptive DDRC has promise.

Some versions of the Adaptive DDRC (different parameter sets) were rated below a conventional design (peak controlled gain). This indicates a need for more extensive testing to obtain the “best” set of adaptation parameters. With this “best” parameter set, our design offers the control of a peak limiter (i.e. the peak level never exceeds the limit threshold) with perceptual performance almost on par with a DDRC that uses average level gain control.

The present Adaptive DDRC implementation uses all of the available processing time (at a sampling rate of 44.1kHz). To achieve this execution time, an input HPF was eliminated. This degrades performance since input dc offsets can corrupt level measurements and cause “plops” in the output signal [18].

Under some conditions, the output signal contains an annoying artifact known as *zipper noise*. This artifact is caused by successive attacks at zero crossings when the average level is used for gain control. In his DDRC design, McNally [12] implemented an adaptive smoothing scheme to remedy this problem. The lack of processing time precluded the implementation of this smoothing scheme. A new implementation on a
faster processor should be investigated so listening tests with the HPF and adaptive smoothing schemes can be conducted.
Bibliography


Appendix A

Code Listings
ddrc.lod: main.lnk gain.lnk coeff.lnk rshftab.lnk static.lnk
lnk56000 -Bddrc.lod -M gain coeff rshftab static main

# main program
# main.lnk: ioequ.asm input.asm main.asm
# asm56000 -B -Lmaint main

# ddrc algorithm [interrupt handler]
# gain.lnk: gain.asm fshft.asm input.asm ioequ.asm
# asm56000 -B -Lgaint gain

# polynomial approximation coefficients
# coeff.lnk: coeff.asm
# asm56000 -B -Lcoefft coeff

# lookup table for fast right shifts (denormalization)
# rshftab.lnk: rshftab.asm
# asm56000 -B -Lrshftabt rshftab

# static compression parameters
# static.lnk: static.asm
# asm56000 -B -Lstatict static
Main module of adaptive DDRC program.

This program executes an endless loop and services interrupts from the SSI. The SSI interrupt service routine is the DDRC algorithm. This module is based on the program intioevb.asm by C. Thompson.

Start of Main Program

; Mask all Interrupts
ori #$03,MR

; Initialize SSI and BCR
mov ep #$0,x:M_BCR ;zero wait states to all memory
mov ep #$3000,x:M_IPR ;SSI Rx interrupt priority
mov ep #$4000,x:M_CRA ;set SSI word_length = 16
mov ep #$b200,x:M_CRB ;word long frame sync, RX interrupts enab.,
;external clock and frame sync
mov ep #$01FF,x:M_PCC ;turn on SSI port

; Set up ADS board in case of force
C:\TODD\56K\JUL10\MAIN.ASM 2

; break instead of force reset
;----------------------------------------
movec #1,sp ; init stack pointer
movec #0,dr ; clear loop flag/interrupt mask bits
;----------------------------------------
; Warm restart and 'one-time'
; initialization for DDRC
;----------------------------------------
_init move #logcof,r4 ; point to polynomial coefficients
move #(8-1),m4 ; mod 8 addressing
move #in_buf,r3 ; circular input buffer
move #(bufsz-1),m3 ; mod(buvsiz) addressing
move #-1,n3

;# input buffer is initialized with small +ve values
;# it initially contains no zero crossings
move #pk_buf,r2
move #(bufsz-1),m2 ; for INPUT section with
move #-1,n2 ; mod(buvsiz) addressing

move #pk_buf,r6
move #(bufsz-1),m6 ; for OUTPUT section with
move #1,n6 ; mod(buvsiz) addressing

;# set r7 for mod(buvsiz) addressing
move #(bufsz-1),m7
move #-1,n7

move #-1,n5
;----------------------------------------
; fill input buffer with small +ve
; samples.
;----------------------------------------
move #$000001,a ; a <- 2^-23
rep #(bufsz) ; fill input buffer with 2^-23
move a,x:(r3)+
clr a ; clear initial output value
;----------------------------------------
; zero fill peak buffer.
;----------------------------------------
rep #(bufsz) ; zero fill peak buffer
move a,x:(r2)+
;----------------------------------------
; Unmask all Interrupts
;----------------------------------------
andi #$FC,MR
;----------------------------------------
C:\TODD\56K\JUL10\MAIN.ASM 3

; Loop & wait for Rx Interrupt
self jmp self
!!! jmp rdwrite

; I/O Interrupt Handler

org p:
xref rdwrite

; Interrupt exception handlers for SSI RX-TX overrun/underrun

clear the exception flag for the SSI transmitter and return

txcept movep x:M_SR,a0 ; read status
      movep al,x:M_TX ; send something
      nop ; place a BREAKPOINT here
      rti ; for debugging exceptions

; clear the exception flag for the SSI receiver and return

rxcept movep x:M_SR,a0 ; read status
      movep x:M_RX,al ; receive something
      nop ; place a BREAKPOINT here
      rti ; for debugging exceptions

; Install interrupt handlers
; in vector table

org p:I_SSIIRD ; SSI interrupt vector-receive
jsr rdwrite

org p:I_SSIIRDE ; SSI exception vector-receive
jsr rxcept

org p:I_SSIITDE ; SSI exception vector-transmit
jsr txcept

endsec
end
Adaptive DDRC algorithm

This module is an interrupt service routine. In response to a
SSI Rx interrupt, it reads an input sample, writes an output
sample (computed last time the algorithm was executed) and
calculates the output sample for the next SSI Rx interrupt.

;***************************************************************************
gain ident 0,1
opt cex,cre,w
page 140,66,4,4,5
section gain

;Include files for the DDRC routine

include 'ioequ'
include 'input'
include 'fshft'
org y:
xref logcof
xref unflow
xref rshstop
xref rshbot
xref limit
org x:
xref lthresh
org y:$047E

;!!! temporary storage for testing only !!!
temp3 dc $000000
temp4 dc $000000

; Start of DDRC routine

;org p:$80
xdef rdwrite ; address installed in vector table

; Real-time I/O:
; Read A/D data from the SSI RX register
; and write data to the SSI TX register.
; Both registers operate synchronously so
; TX empty flag need not be checked. TX is
; always empty due to reception of new A/D word.

rdwrite movep x:M_RX,x0 ; A/D input-->x0
movep a,x:M_TX ; write output-->M_TX
INPUT SECTION OF ROUTINE

* HIGHPASS FILTER THE INPUT SIGNAL. (1ST ORDER FILTER)

ENTRY:  x0 - input sample
         r3 - pointer to input buffer
         n3 - offset to previous input sample (-1)

EXIT:   x0 - filtered output sample
         r1 - pointer to avg. level filter coefficients

** FIND THE PEAK LEVEL BETWEEN ZERO CROSSINGS AND STORE IT IN THE PEAK LEVEL BUFFER. **

ENTRY:  x0 - filtered input sample (z[n])
         r2 - pointer to peak buffer
         r3 - pointer to input buffer
         r5 - pointer to peak value (c_peak)
         n3 - offset to previous input sample (-1)

EXIT:   y0 - input level squared (z[n]^2)
         r5 - pointer to peak level

REG'RS:  a,b,x0,x1,y1 - temporary storage

***************************************************************************

;***********************************************************************
;***************************************************************************
;***************************************************************************
;***************************************************************************
C:\TODD\56K\JUL10\GAIN.ASM

mpy x0,x1,a
cmp y1,a
jgt _notzc

_iszc move x1,x:(r2)+
jmp _avgl

_notzc cmp x1,b
jmi _avgl
move y0,y:(r5)

***************************************************************************
; Compute the average signal level (Output is q[n]).
; Entry: y0 - abs(input sample)
; r5 - pointer to previous st level
; r1 - pointer to filter coefficients (ifltcof)
; Exit: r5 - points to avg. signal level
***************************************************************************
_avgl move y:(r5)+,y1 ; update r5 (y1 <- junk)
move x:(r1)+,x0 ;x0<-tavl
move y:(r5),y1 ;yl<-q[n-1]
mpy x0,y0,a x:(r1)+,x1 ;a<-tavl*|z[n]|, xl<-omtavl
mac x1,y1,a ;a<-tavl*z[n]+omtavl*q[n-1]
move a,y:(r5) ;store st_avg

==========================================================================
= Output Section of Routine
==========================================================================

_outzc move x:(r3)+,x0 ;x0 <- z[n-N+1] (current output sample)
move x:(r3)-,xl ;xl <- z[n-N+2]
mpy x0,x1,a ; a <- z[n-N+1]*z[n-N+2]
tst a ; IF (z[n-N+1]*z[n-N+2]>0)
jgt _level ; goto _level
; ELSE
move x:(r6)+,a ; update output peak pointer (r6) ; (a <- junk )

;******************************************************************************

; Compute adaptive level
; Entry: r5 - points to loc'n past stl (y-memory)
; r6 - points to current peak value (x-memory)
; Exit: b - contains the adaptive level
;******************************************************************************

_level move y:(r5)+,x0 ; x0 <- s (short-term level)
move x:(r6),x1 ; x1 <- peak (adapt via peak)
move y:(r5)+,a ; a <- 1.0
move y:(r5)+,b ; b <- Tpc
cmp x1,b ; IF (1 < Tac)
jml _dobe b ; goto _dobe

clr a y:(r5)+,b ; a <- 0, b <- Tpc
cmp x1,b y:(r5)+,y0 ; IF (1 > Tpc), y0 <- 1/d*scale
jml _dobe ; goto _dobe

mpy x1,y0,a y:(r5)+,b
add a,b
move b,y1
MSHL y1,x1,(9),b ; unscale result
move b0,a ; a <- (1 - Tpc)/d

_;dobee move #one,b ; b <- 1.0
sub a,b x:(r6),y0 ; b <- 1 - A, y0 <- p
move a,x1 ; x1 <- A, y1 <- B
move b,y1 ; y1 <- B
mpy x0,x1,b ; b <- s*A
mac y0,y1,b ; b <- s*A + p*B

;!!! move x0,b ; !!! testing !!! avg level
;!!! move x:(r6),b ; !!! testing !!! peak level

;******************************************************************************

; Make attack/recovery decision, attack if (x[n]-y[n-1])>0 
; Entry: b - adaptive signal level, x[n]
; r3 - pointer to input buffer (@z[n-N/2])
; r1 - pointer to previous recovered output sample, y[n-1] .
; r6 - pointer to peak buffer (output section)
; Exit: a - "recovered" or "attacked" level, y[n]
;******************************************************************************

_attack move x:(r1),y0 ; y0 <- y[n-1]
cmp y0,b ; if (x[n] - y[n-1])<0
C:\TODD\56K\JUL10\GAIN.ASM

jm i _recov ; goto _recov

/* Attack ... \[n-1] <= \[n]
move b,a ; a <- \[n]
imp _arend

;**********************************************************************
/*
*/ Compute adaptive recovery filter coefficient.
*/
/*
* Entry: r5 - points to loc'n containing 1.0
* r6 - points to peak level
*/
/*
* Exit: x0 - contains recovery filter coefficient
*/
/*
;**********************************************************************
/* compute Cd - relative peak measure
_recov move #radapt,r5 ; pointer to coefficients
move x:(r1),b ; b <- prev recovered level (\[n-1])
move y:(r5)+,a ; a <- 1.0
sub b,a ; a <- 1 - p
move y:stlev,b ; b <- s
add b,a ; a <- 1 - p + s = Cd

;* compute peak variation (Pv)
;* r7 set for mod(bufsiz) addressing
lwa (r6)+n6,r7 ; temp ptr to peak buffer (n6=1)
clin b a,x0 ; b <- 0, x0 <-Cd (store Cd)
move x:(r7)+,x1 ; x1 <- pk[n+1] (ie. future peak)
move y:(r5)+,y1 ; y1 <- 1/N
rep #N ; compute avg of future peak levels
mac x1,y1,b x:(r7)+,x1 ; b <- b + (1/N)sum[p[n+k]]
move x:(r6)+,a ; a <- p[n]
sup b,a ; a <- p[n] - (1/N)sum[p[n+k]]

;* To adapt or not to adapt? What is the condition.
;* x0 contains Cd
move y:(r5)+,x1 ; x1 <- fixed a
move y:(r5)+,b ; b <- Q
cmp x0,b ; IF (Cd > Q)
jmi _rlev ; goto _rlev (ie. a=fixa)
jmp _rlev ; !!! testing !!! no adaptation

;* compute adaptive recovery coefficient
;* a contains Pv
move a,x1 y:(r5)+,y1 ; x1 <- Pv, y1 <- k1
mpy x0,y1,a y:(r5)+,y1 ; a <- k1*Cd, y1 <- -k2
mac x1,y1,a y:(r5)+,y1 ; a <- k1*Cd - k2*Pv, y1 <- c
add y1,a ; a <- k1*CD - k2*Pv + c
move a,x1 ; x1 <- adaptive recov. coefficient

;**********************************************************************
Recover if \((x[n]-y[n-1])<0\)

Entry: \(x_l\) - contains the filter coefficient, \(a\)
\(y_0\) - contains the prev. recovered level, \(y[n-1]\)
\(x_l\) - points to the prev. recovered level, \(y[n-1]\)

Exit: \(a\) - contains recovered signal level \((l[k])\)

**--------------------------------------------------------------------------**

\(_{\text{rlev}}\) mpy \(x_l, y_0, a\); \(a \leftarrow y[n-1] a = y[n]\)

\(*\) store output for next time 'thru

\(_{\text{arend}}\) move \(a, x:\{x_l\}\); \(y[n-1] \leftarrow y[n]\)

**--------------------------------------------------------------------------**

Determine the input region

Entry: \(a\) - input value.

Exit: \(x_l\) - pointers to static parameters (+7).
\(A\) - gain for no-action region \((\text{jmp to } _{\text{gend}}\text{ only})\)

Reg'x: \(x_l\) - temporary storage of linear (ms) thresholds.
\(n_l\) - offset for static parameter structures.

**--------------------------------------------------------------------------**

\(_{\text{region}}\) move \(#\text{limit},x_l\); point to static parameters
move \(#\text{step},n_l\); step value for static data

nop
move \(y:\{x_l\}+n_l,x_l\); get \(Lth\) and update \(x_l\)

\(\text{cmp} \ x_l,a \ y:\{x_l\}+n_l,x_l\); get \(Eth\) and update \(x_l\)
\(\text{jpl} \ _{\text{norm}}\); IF \(\text{input}>Lth\) THEN \(\text{limit}; \text{goto } _{\text{norm}}\)

\(\text{cmp} \ x_l,a \ y:\{x_l\}+n_l,x_l\); get \(Cth\) and update \(x_l\)
\(\text{jmi} \ _{\text{norm}}\); IF \(\text{input}<Eth\) THEN \(\text{expand}; \text{goto } _{\text{norm}}\)

\(\text{cmp} \ x_l,a \ y:\{x_l\}+n_l,x_l\); get dummy thresh. and update \(x_l\)
\(\text{jpl} \ _{\text{norm}}\); IF \(\text{input}>Cth\) THEN \(\text{compr}; \text{goto } _{\text{norm}}\)

**--------------------------------------------------------------------------**

\(-\) input level is in no-action region

**--------------------------------------------------------------------------**

move \(#\text{one},a\); no-action region has gain of 1.0
\(\text{jmp} \ _{\text{gend}}\); goto end of gain computation

**--------------------------------------------------------------------------**

Normalize an input value supplied in \(x_0\) and correct the pointer
to the static data.
**Entry:**  
a - input value  
r1 - static data pointer (+7)  

**Exit:**  
B - normalized input value  
r0 - integer part or exponent  
r1 - points to static data (gain) for input region.  

**Reg'r:**  
x1 - left shift constant  
x0 - temporary input storage  
rl - pointer adjustment value  

---

**_norm**  
move #7, rl  
move a, x0  
lua (rl)-nl, rl  
move #0, x0  
tfr a, b  
>kl, xl  
cmp xl, a  
jl _doshft  
mpy x0, xl, a  
mov a0, b1  

**_doshft**  
rep #(shft-1)  
norm r0, b  

---

**Compute log2() of value supplied in B using a third order polynomial approximation (a1*x^3+a2*x^2+a3*x+a4)/8.**

**Entry:**  
B - normalized input value (i.e. 0.5 <= B < 1.0)  
r4 - points to scaled polynomial coefficients (initialized in main.asm - circular)  

**Exit:**  
A - log2(B)/8  

**Reg'r:**  
x1 - contains x^3  
x0 - contains x  
yl - contains x^2  
y0 - temp. storage for coefficients  

**_log2**  
move b, x0  
mpyr x0, x0, a  
move a, y1  
mpyr y1, x0, a y:(r4)+, y0  
move a, x1  
mpyr y0, x1, a y:(r4)+, y0  
mac y0, y1, a y:(r4)+, y0  
mac y0, x0, a y:(r4)+, y0  
add y0, a
Compute the gain in \( \log_2() \) "dB".

**Entry:**
- \( r0 \) - exponent (integer)
- \( A \) - \( \log_2(m)/16 \) (fractional)
- \( r1 \) - points to array of static parameters

**Exit:**
- \( x0 \) - fractional part of gain (16-bits accuracy)
- \( n0 \) - integer part of gain (table offset)

**Reg'r:**
- \( x1 \) - temp storage
- \( y0 \) - fractional part of gain (16-bits accuracy)

**Comment Notation:**
- \( E \) - integer exponent (two's complement)
- \( m \) - fractional mantissa
- \( S \) - gain slope
- \( Xth \) - \( \log_2() \) threshold for input region
- \( Adj \) - gain adjustment (non-zero for limiting only)

**Modified:**
- Apr 28/91 - changed scaling to fix error and get extra bit of accuracy.

---

Evaluate \( E*S \)

1. Move \( r0 \), \( x1 \) to \( x1 \), \( y1 \)
2. MSHR \( x1, y1, (scale+2), b \)
3. Move \( b0, x1 \)
4. Move \( y: (x1)+, y1 \)
5. Mpy \( x1, y1, b \) to \( a, x1 \)

Evaluate \( \log_2(m)*S \)

1. Mpyr \( x1, y1, a \)
2. Asr \( a \)
3. Asr \( a \)
4. A <- \( \log_2(m)*S/128 \)

Sum exponent, mantissa and unscale result

1. Add \( b, a \) to \( y: (x1)+, x1 \)
2. Mpy \( x1, y1, b \) to \( y: (x1), x1 \)
3. Asr \( b \)

Sub \( b, a \)

4. Add \( x1, a \)

Clr \( b \) to \( a1, x0 \)

5. Move \( a0, b1 \)

---

\( a <- S*(E + \log_2(m))/128 \)

\( x1 <- Xth/16 \)

\( b <- (S/4)*(Xth/16) \)

\( x1 <- Adj/128 \)

\( b <- S*Xth/128 \)

\( a <- S*(\log_2(m)+E-Xth)/128 \)

\( a <- S*0 \), \( x0 <- result/128 \)

\( (\text{extra shift to recover int. part}) \)

\( b <- \text{frac}_{-}\text{part} \)
Check integer part of exponent for underflow
move #m24,x1 ; x1 <- -23
move #zero,a0 ; zero fractional part for compare
move #unflow,n0 ; use shift coefficient of zero
cmp x1,a0 ; if (int_part - -23) < 0
jmi _dnorm ; goto _dnorm

Recover integer portion of result
asl b ; combined with above MSHL
move a1,r0 ; store (int_part-1) in r0
move #$8000000,x1 ; set-up to adj. sign of fract. part
lua (r0)+,n0 ; pipeline delay
; store corrected int_part
; (used for de-normalization)

Recover fractional part of result ( bl contains fract_part )
asr b ; get bits into correct places
or x1,b ; set MSB (all results are -ve)
move bl,x0 ; set-up for exp2() calculation

Compute a third order polynomial approximation to $2^x$ for a fractional input x.
Entry: x0 - fractional part of gain (db)
r4 - points to polynomial coefficients
Exit: A - $2^x$
Reg'rs: x1 - contains $x^3$
x0 - contains x
y1 - contains $x^2$
y0 - temp. storage for coefficients

Do right shift as specified by integer part of log2().
(i.e. un-normalize result)
Entry: a - $2^x$ for fractional part of input
Exit: a - right shifted result

Reg'x x0 - temp storage for shift constant
y0 - temp storage of input value
r0 - table index
n0 - integer part part result (= table offset)

******************************************************************************
dnorm move #rshtop,r0 ;get table address
move a,y0 ;y0<-2^-(frac(gain))
move y:(r0+n0),x0 ;get shift constant from table
mpy x0,y0,a ;shift right
;result is in A
_gend move a,yl ;yl <- gain

; Apply Gain to Output Sample

move x:(r3),xl ;xl <- z[n-N+1]
mpyr xl,yl,a ;output = a <- z[n-N+1]*gain
asl a

; End of DDRC Algorithm

rti
endsec
end
C:\TODD\56K\JUL10\STATIC.ASM

;***************************************************************************
;* User Adjustable Parameters for Static Characteristics.
;***************************************************************************
static ident 0,2
  title 'Static Parameters'
  opt cex,cre,w
  page 160,66,4,4,5

section static

; Thresholds in dB
LTH equ -15.0
CTH equ -35.0
ETH equ -50.0

; Ratios are output/input in dB
LRatio equ 1.0/100.0
CRatio equ 1.0/3.0
ERatio equ 2.0/1.0

; Internal Variables (DO NOT ALTER!!!)

; Gain slopes
LS equ LRatio-1
CS equ CRatio-1
ES equ ERatio-1

; Thresholds converted to log2()
LTH2 equ LTH/(20*@L10(2.0))
CTH2 equ CTH/(20*@L10(2.0))
ETH2 equ ETH/(20*@L10(2.0))

; Gain adjustment for limiting (gain already applied by compression)
GA equ CS*(LTH2-CTH2)

; Linear data.
; linear threshold: linear threshold (mean square values).
; gain slope: S/4
; log2(threshold): dB thresholds converted to log2()

org y:$04A1
xdef limit
limit dc @pow(10.0,LTH/20.0) ;linear threshold
C:\TODD\56K\JUL10\STATIC.ASM

; xxx changed GA/256 to GA/128 May 22/91 TS
; (gain slope)/16
; log2(threshold)/16
; gain adjust. for limiting

dc     LS/4.0
dc     LTH2/16

expand dc @pow(10.0,ETH/20.0)
dc     ES/4.0
dc     ETH2/16
dc     0.0

compr dc @pow(10.0,CTH/20.0)
dc     CS/4.0
dc     CTH2/16
dc     0.0

noact dc 0.0
dc 0.0
dc 0.0
dc 0.0
endsec
end
**User Adjustable Parameters**

.times equ 4095 ; # of times for gain loop !!! testing only !!!

;------------------------------------------------- ; Average Level Filter Parameters ;-------------------------------------------------
tavl equ 0.000226733 ; avg. level coefficient (tau=100ms)
*tavl equ 0.00007558 ; avg. level coefficient (tau=30ms)
omtavl equ 1-tavl ; coefficient for st_level filter

;------------------------------------------------- ; Adaptive Level Thresholds ;-------------------------------------------------
; TPC = peak control threshold > -50 dB
; TAC = average control threshold (TPC - TAC > 10dB)

TPC equ -15
TAC equ -25

;------------------------------------------------- ; Adaptive Recovery Parameters ;-------------------------------------------------
; fixa = fixed recovery time (low Cd signals)
; Q = (crest difference threshold)^-1
; k1 = crest difference coefficient
; k2 = peak variation coefficient
; c = adaptation equation constant
; N = # of future samples for Pv calculation

*fixa equ 0.999886641 ; fixed Tr = 200 ms
*k1 equ 0.000118827 ; max Tr = 200 ms
*k2 equ 0.000118827 ; min Tr = 50 ms
*c equ 0.9993665621

fixa equ 0.999909309 ; fixed Tr = 250 ms
k1 equ 0.000285002 ; max Tr = 250 ms
k2 equ 0.000285002 ; min Tr = 25 ms
c equ 0.999379205

Q equ 0.86
N equ 5
pvc equ 1.0/(1.0*N) ; 1/N ... coefficient for Pv samples

;*****************************************************************************
; INTERNAL PARAMETERS - DO NOT ALTER !!!
;*****************************************************************************
avmern equ $04Bl
xdef bufsiz
bufsz equ 1150 ; size of input and peak buffers
adaptive level parameters

Tac  equ  @pow(10.0,TAC/20.0) ; linear adaptive level thresholds
Tpc  equ  @pow(10.0,TPC/20.0) ; delta
dlt  equ  (Tac-Tpc) ; scaling coeff. for adaptive level

do     equ  512.0  ; scaling coeff. for adaptive level

Storage Locations for Testing

org  x:avmem
inval  dc  $000000 ; input value
temp1  dc  $000000 ; temp storage !!! testing only !!!
temp2  dc  $000000

org  y:$0000
stor1  dc  $000000
stor2  dc  $000000
stor3  dc  $000000
stor4  dc  $000000
stor5  dc  $000000
stor6  dc  $000000

Input and Gain buffers

org  x:$0 ; input signal buffer
inbuf  dsm  bufsiz
xdef  in_buf
pk_buf  dsm  bufsiz
xdef  pk_buf

State variable for input HPF
Coefficients for IIR averaging filters

org  x:$4c0 ; HP state variables
hpstate  dc  $000000 ; x(n-1) (output)
ifltc0f  dc  tav1 ; ST Avg. filter coefficients
dc  omtav1
recovery  dc  $000000 ; recovery filter - delayed output

Coefficients for input HPF
Peak detection stuff
Adaptive level stuff

org  y:$4c0
*hpc0ff  dc  0.99893 ; a0 (HP filter coefficients)
*x  dc  -0.99893 ; a1
*x  dc  0.9978628 ; -b1
C:\TODD\56K\JUL10\INPUT.ASM 3

hpcovf    dc $7fffffff ; no filter
dc 0.0
dc 0.0

peak     dc $000000 ; peak between zero-crossings
dc $000000

stlev    dc $7fffffff ; short-term average signal level
dc Tpc ; peak control threshold
dc 1/(dscale*dlc) ; scaled delta^-1

ladapt   dc $7fffffff ; approx. 1.0
dc Tac ; st average control threshold
dc Tpc ; peak control threshold

radapt   dc $7fffffff ; approx. 1.0
dc pvc ; coefficient for Pv samples (1/N)
dc fixa ; fixed recovery coefficient

dc Q ; adaptive recovery threshold

dc -1.0*k2 ; Pv coefficient

dc c ; constant for adaptation equation

FLICT[l] :****************************************************************************

* Output Section Constants
*
FLICT[l] :****************************************************************************

C[l]----------------------------- Constants for determination of the region

sstep  equ 4 ; step size thru static data table
one   equ $7fffffff ; approx. 1.0

FLICT[l] :------------------------------

C[l]----------------------------- Constants for log2() computation

shift   equ 12 ; bits to shift for binary search
k1  equ @cvi(@pow(2,shift-1)) ; test value & shift constant

FLICT[l] :-----------------------------

C[l]----------------------------- Constants for gain computation

scale  equ 4 ; log2(scaled_factor)
m24   equ $FPPF88 ; used for underflow check
zero  equ $000000 ; used to zero fractional part
C:\TODD\56K\JUL10\RSHFTAB.ASM 1

;*****************************************************************************
;* Lookup table for FAST right shifts
;*****************************************************************************
rshftab ident 0,3

title 'Fast Right Shift Lookup Table'

opt cex,cre,w

page 160,66,4,4,5

section shift_table

org y:$0468

xdef unflow

xdef rshbot

xdef rshtop

unflow dc $000000 ; underflow table

rshbot dc @pow(2,-23) ; 2^-23
dc @pow(2,-22)
dc @pow(2,-21)
dc @pow(2,-20)
dc @pow(2,-19)
dc @pow(2,-18)
dc @pow(2,-17)
dc @pow(2,-16)
dc @pow(2,-15)
dc @pow(2,-14)
dc @pow(2,-13)
dc @pow(2,-12)
dc @pow(2,-11)
dc @pow(2,-10)
dc @pow(2,-9)
dc @pow(2,-8)
dc @pow(2,-7)
dc @pow(2,-6)
dc @pow(2,-5)
dc @pow(2,-4)
dc @pow(2,-3)
dc @pow(2,-2)
dc @pow(2,-1)

rshtop dc $ffffffff ; approx. 2^0

endsec

end
FAST SHIFT MACROS

Macro definitions for generating fast right and left shift constants, KR and KL, and performing the right and left shifts.

Let

- \( s \) = the source register
- \( m \) = the multiplier register
- \( n \) = the number of bits to be shifted
- \( acc \) = the destination accumulator

where \( s, m \) can be one of \( X0, X1, Y0, Y1 \) and \( acc \) can be A or B

Taken from: "Fractional and Integer Arithmetic Using the DSP56000 Family of General-Purpose DSP's" pg 9

MSHR macro

\[
\text{move} \quad \#\text{pow}(2,-n),m \\
\text{mpy} \quad s,m,acc \\
\text{endm}
\]

; four input variables
; load the multiplier register
; shift right \( n \) bits

MSHL macro

\[
\text{move} \quad \#\text{cvi}(\text{pow}(2,n-1)),m \\
\text{mpy} \quad s,m,acc \\
\text{endm}
\]

; four input variables
; load the multiplier register
; shift left \( n \) bits
;**********************************************************************
;* Coefficients for 3rd-order polynomial approximation to log2() and exp2()
;**********************************************************************
coeff ident 0,4
  title 'Polynomial Coefficients for log2 and exp2'
  opt cex,cre,w
  page 132,68,4,4,5

section coefficients
;**********************************************************************
;* Coefficients for 3rd order polynomial approximation to log2().
;**********************************************************************
org y:$0480
xdef logcof
logcof dc 1.24356684/8.0  ;a1
dc -4.15703264/8.0  ;a2
dc 6.05895642/8.0  ;a3
dc -3.14494063/8.0  ;a4

;**********************************************************************
;* Coefficients for 3rd order polynomial approximation to 2^x
;**********************************************************************
xdef expcof
expcof dc 0.03954285  ;a1
dc 0.23088672  ;a2
dc 0.69314016  ;a3
dc 0.99994464  ;a4

endsec
end
; Equates for DSP56000 I/O registers and ports
;ioequ ident 1,0
;
;& Equates for I/O Port Programming
;
; Register Addresses
M_BCR equ $FFFE ;Port A Bus Control Register
M_PCC equ $FFE1 ;Port C Control Register

; Equates for SSI
;
; Register Addresses
M_RX equ $FFEF ;Serial Rx Data Register
M_TX equ $FFEF ;Serial Tx Data Register
M_CRA equ $FFEC ;SSI Control Register A
M_CRB equ $FFED ;SSI Control Register B
M_SR  equ $FFEE ;SSI Status Register

; Equates for Exception Processing
;
; Register Addresses
M_IPR equ $FFFF ;Interrupt Priority Register

; Equates for DSP56000 interrupts
;
C:\TODD\56K\JUL10\IOEQU.ASM 2

I_IRQB   equ   $0010  ;_IRQB
I_SSIRD  equ   $000C  ;SSI Rx Data
I_SSIRDE equ   $000E  ;SSI Rx Data with exception
I_SSIDTD equ   $0010  ;SSI Tx Data
I_SSITDE equ   $0012  ;SSI Tx Data with exception
Appendix B

Memory Map

P:$7FF

P:$14A

DDRC algorithm
(interrupt service routine)

P:$080

P:$070

Initialization Code

P:$040

P:$000

Program Memory
APPENDIX B. MEMORY MAP

X: $FFFF

peak buffer

X: $C70

hpf parameters
temporary storage

X: $800

input buffer

X: $4C3

 temporary storage

X: $4B1

input buffer

X: $47D

input and adaptation parameters

Y: $7FF

Y: $4D0

input and adaptation parameters

Y: $4C0

static characteristics

Y: $4B0

right shift table

Y: $4A1

polynomial coefficients

Y: $4A0

short-term average buffer (unused)

Y: $488

Y: $487

Y: $480

Y: $48D

Y: $000

Y data memory

X data memory
Appendix C

Level Adaptation Scaling Factor

In this appendix, the restrictions on the locations of the peak control threshold ($V_{pc}$) and the average control threshold ($V_{ac}$) are developed. The maximum value of the slope of the linear combination portion of the characteristic depends on the separation of the two thresholds. The scaling factor required for these calculations must also be found. It is desirable that this scaling factor be as small as possible.

Maximize

$$|\Delta| = \left| \frac{1}{V'_{ac} - V'_{pc}} \right|$$

with respect to $V'_{ac}$ and $V'_{pc}$. This can be accomplished by minimizing

$$|\Delta| = \left| \frac{V'_{ac} - V'_{pc}}{10^{\frac{V_{ac}}{10}} - 10^{\frac{V_{pc}}{10}}} \right|^{1/20}$$

Let $V_{ac} = z$ and $V_{pc} = y$. We must always have $y > z$ and $z, y > -100dB$ (i.e. the minimum input level). This can be expressed as

$$-100 < y < 0$$

$$-100 < z < y - \epsilon$$
APPENDIX C. LEVEL ADAPTATION SCALING FACTOR

where \( \epsilon = y - x > 0 \). Now we may write

\[
|\Delta| = 10^{y/10} \left| 10^{-\epsilon/20} - 1 \right|
\]

Now \( |\Delta| \) may be minimized by minimizing \( 10^{y/20} \) and \( 10^{-\epsilon/20} - 1 \). \( 10^{y/20} \) is a minimum for \( y = -100 \).

\[
10^{-100/20} = 10^{-5}
\]

\( 10^{-\epsilon/20} - 1 \) is a minimum for the smallest value of \( \epsilon \). Notice that \( \epsilon = 0 \Rightarrow \) minimum value is 0. After much trial and error, we set \( \min(y) = -50dB \) and \( \min(\epsilon) = 10dB \). This gives

\[
\max(|\Delta|) = \frac{1}{\min(\Delta)} = 462.5
\]

This maximum \( \Delta \) must be scaled down by \( 2^9 \) so that it is fractional. The calculations will be accurate to \( 23 - 9 = 14 \) bits (only positive twos complement values are used for the thresholds). This implies that the lower limit of the average control threshold is

\[
20 \log_{10}(2^{-14}) \approx -84dB.
\]

In summary, the restrictions on the thresholds are

\[
V_{pc} > -50dB
\]

\[
\epsilon > 10dB
\]

\[
V_{ac} > -84dB
\]

\[
V_{ac} = V_{pc} - \epsilon
\]
Appendix D

Polynomial Approximation Coefficients

The coefficients for polynomial approximations over the BFP input range $0.5 \leq z \leq 1.0$ are tabulated below. The coefficients are all tabulated as

$$f(z) = \sum_{i=0}^{i=N} a_i z^i$$

where $N$ is the order of the approximation.

<table>
<thead>
<tr>
<th>Order</th>
<th>$a_0$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>-2.66403001</td>
<td>4.01872957</td>
<td>-1.35900725</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>3</td>
<td>-3.14494063</td>
<td>6.05895642</td>
<td>-4.15703264</td>
<td>1.24356684</td>
<td>—</td>
</tr>
<tr>
<td>4</td>
<td>-3.50561466</td>
<td>8.09923378</td>
<td>-8.39760912</td>
<td>5.08408893</td>
<td>-1.28017403</td>
</tr>
</tbody>
</table>

Table D.1: BFP Chebyshev Polynomial Coefficients
APPENDIX D. POLYNOMIAL APPROXIMATION COEFFICIENTS

**Table D.2: BFP Chebyshev Polynomial Coefficients**

<table>
<thead>
<tr>
<th>Order</th>
<th>$a_0$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.99870893</td>
<td>0.66909732</td>
<td>0.17157244</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>3</td>
<td>0.99994464</td>
<td>0.69134017</td>
<td>0.23088672</td>
<td>0.03954285</td>
<td>—</td>
</tr>
<tr>
<td>4</td>
<td>0.99999809</td>
<td>0.69305067</td>
<td>0.23943921</td>
<td>0.05322683</td>
<td>0.00684199</td>
</tr>
</tbody>
</table>

**Table D.3: BFP Chebyshev Polynomial Coefficients (two polynomials)**

<table>
<thead>
<tr>
<th>Range</th>
<th>$a_0$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0.5 \leq x &lt; 0.71875$</td>
<td>-3.383159930</td>
<td>7.199881431</td>
<td>-5.955824132</td>
<td>2.177818986</td>
</tr>
<tr>
<td>$0.71875 \leq x &lt; 1.0$</td>
<td>-2.883159931</td>
<td>5.093505086</td>
<td>-2.983556684</td>
<td>0.7732426917</td>
</tr>
</tbody>
</table>
Appendix E

Mixer Schematic

all opamps are LF356
supply is ±12 volts dc
Appendix F

Listening Test Results

This appendix contains the raw listening test results. In the table below, each listener is identified by his/her initials. The tests are indicated by a number/letter combination where the letter indicates the musical signal used and the number indicates the processing. The numbers and letters for each signal and processing option are listed in Section 5.3.
### APPENDIX F. LISTENING TEST RESULTS

#### Table F.1: Listening Test Results

<table>
<thead>
<tr>
<th>Test Number</th>
<th>Subject</th>
<th>Trial Averages</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>JVH</td>
<td>PC</td>
</tr>
<tr>
<td>1A</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>1B</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>1C</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>2A</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>2B</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2C</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3A</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3B</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3C</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>4A</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>4B</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>4C</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5A</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5B</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>5C</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>6A</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>6B</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>6C</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>7A</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>7B</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>7C</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>