ANALYSIS OF ZERO-LEVEL SAMPLE PADDING OF VARIOUS MP3 CODECS

By

JOSH BERMAN

B.S., University of Colorado, Denver, 2013

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado, in partial fulfillment

of the requirements for the degree of

Masters of Science

Recording Arts

2015

This thesis for the Master of Science Degree by

Josh Berman

has been approved by the

Recording Arts Program

By

Lorne Bregitzer

Jeff Smith

Catalin Grigoras, Chair

11/20/2015

Berman, Josh (M.S. Recording Arts)

Analysis of Zero-Level Sample Padding of Various MP3 Codecs

Thesis directed by Assistant Professor Catalin Grigoras

ABSTRACT

As part of the MP3 compression process, the codec used will often pad the beginning and end of a file with “zero-level samples”, or silence. The number of zero-level samples (ZLS) varies by codec used, sample rate, and bit depth of the compression. Each re-compression of a file in the MP3 format will typically add more silence to the beginning and/or end of the file. By creating multiple generations of files using various audio editors/codecs, we hope to be able to determine the generation of MP3 compression of the files based solely off of the number of ZLS at the beginning and end of the file.

The form and content of this abstract are approved. I recommend its publication.

Approved: Catalin Grigoras

ACKNOWLEDGEMENTS

I'd like to thank my family, first and foremost, for being so awesome and supportive throughout my education. Without you, none of this would have happened. Another big thanks to all of the great teachers I've had throughout the years.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

1.1 A Brief History

The MP3 format, developed in the early 1990's by the Fraunhofer Institute along with the Motion Picture Experts Group, is a very popular method of digital audio compression. The motivation behind it's development was to create a style of compression that could reduce the size of an audio file drastically while still retaining as much quality as possible. In the early 90's hard drive space and Internet bandwidth made full resolution audio files quite impractical. An average length 3-minute song can take up to 50 Megabytes of storage, which even by today's standards can be impractical. The MP3 codec takes advantage of limitations in human hearing to remove information that we can't hear from a lossless audio file. The result is up to 90% smaller file sizes with minimal compromise to audible quality depending on the bit rate used. Over time, various MP3 codecs have been developed (Fraunhofer, LAME, etc) and each has it's own method of coding and decoding MP3 data. Regardless of the exact codec used, MP3 files have become the standard for recording and distributing audio.

Each codec will potentially interpret the same file in different ways. This is due to the nature of MP3 compression. The MP3 standard states certain things that make an MP3 file readable by any decoder, however each decoder may decode the information differently.

1.2 Zero-Level Sample Padding

Part of the coding/decoding process for MP3 files is to pad the beginning and end of the files with silence, or "Zero-Level Samples" (ZLS). Depending on the codec used and specs of the file, more or less zeroes will be added. Typically, each additional re-compression will add more zeroes to the beginning and end of the file. See Figure 1 showing the expected number of zeroes for each generation MP3 compression. This hypothesis is what is being tested in this study.

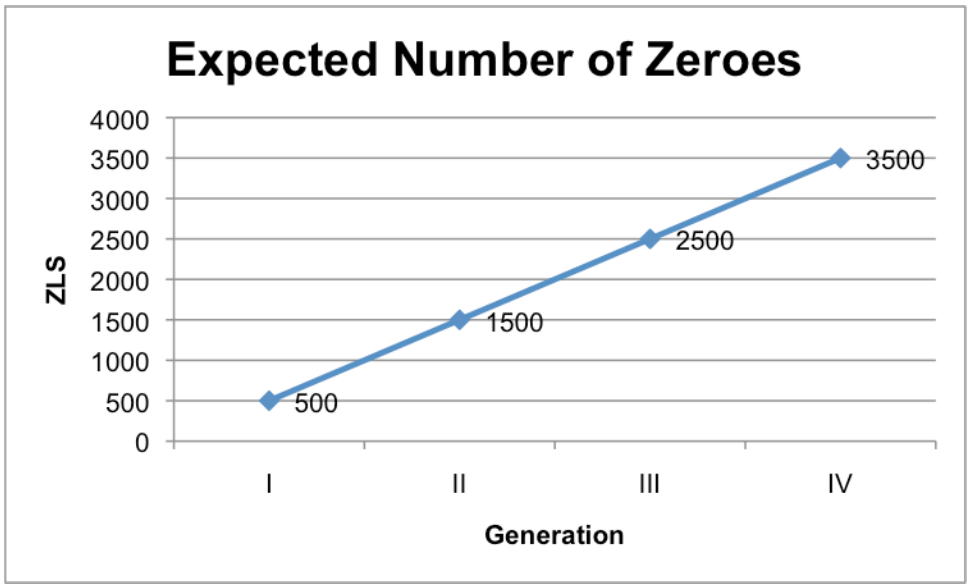


Figure 1 - Expected number of ZLS for each additional MP3 compression

The exact number of zeroes will vary, but this graph shows a general trend we would expect to see.

1.3 Purpose of the Study

The purpose of this study is not to determine the cause/reasoning behind zero-level sample padding, but rather to provide a comprehensive sample of files and their respective padding.

With this study, we hope to be able to find out how many generations of MP3 compression a file has undergone, solely based on the zero-level samples at the beginning and end of a file and on each channel. The default format for most handheld recorders is

MP3, so we expect to see some amount of ZLS on an authentic, unedited file. If a file were to be loaded into an audio editor, manipulated, and re-saved, we would expect that the codec used by the audio editor would add more zeroes to the file, telling us that the file was re-compressed. By collecting files created by various handheld recorders then re-compressed with various programs/codecs, the purpose is to develop a database to reference when analyzing a file for authenticity. For example, if "File A" was claimed to be recorded with "Recorder A" and has 2000 zeroes at the beginning of the file, yet all sample recordings from "Recorder A" only have ~500 zeroes at the beginning, this is a sign that the file was likely re-compressed.

CHAPTER II

MATERIALS AND METHODS

The procedure of this study was to take various handheld audio recorders and record samples of audio, re-compress that audio using different codecs, and analyze the number of zeroes on the file for each generation. The handheld recorders used were:

(5) Tascam DR-07's

(1) Olympus DM-520

(1) Zoom H1

(1) Marantz PMD620

(1) Philips LFH0882

(2) Olympus WS-700's

Each recorder recorded ten (10) files. The files were all of varying loudness:

(3) Recordings of loud music

(3) Recordings of moderate speech

(4) Recordings of silence

The total number of files came to 110 (10 recordings on 11 recorders). The next process was to recompress each file a number of times. Most of the programs used could both encode and decode data, meaning that they can read and create MP3 files. The process for such programs was as follows:

1. Convert each original MP3 file to uncompressed .wav PCM files
2. Convert new .wav PCM files back to MP3
3. Repeat for a total of four (4) generations of .wav files

4. Read the number of zeroes at the beginning and end of each .wav file on each channel.

Various programs/codecs used were strictly coders, not decoders, meaning that could only create MP3 files from an uncompressed .wav file and not read them/convert to .wav. The process for such programs was as follows:

1. Convert each original MP3 file to .wav PCM using dBpoweramp
2. Convert new .wav PCM files to MP3 using the code in question
3. Repeat for a total of four (4) generations of .wav file
4. Read the number of zeroes at the beginning and end of each .wav file on each channel.

Some programs were purely decoders, meaning that they could read an MP3 file and save it as a .wav, but not convert to MP3. The process for such files was as follows:

1. Convert each original MP3 file to uncompressed .wav file using the decoder in question
2. Convert new .wav files back to MP3 using Adobe Audition CC2015
3. Repeat for a total of four (4) generations of .wav file
4. Read the number of zeroes at the beginning and end of each .wav file on each channel.

The reason for converting each MP3 file to .wav before reading the ZLS is because each program potentially uses a different decoder to read the files. This means that opening the same MP3 file in different programs will potentially show a different level of zero-level samples. By using each decoder to create uncompressed .wav files, we essentially “print” the number of zeroes imparted on the file by that decoder. Opening

this new .wav file in any program will show the same number of zeroes because no decoder is being used to read it. This way, all of the .wav files could be read/interpreted by the same MATLAB script without MATLAB adding (or subtracting) more zeroes. All files were recorded at 16bit/128kbps mp3 with the exception of the files from the Phillips LFH0882 which does not have the capability to record at 128kbps. These files were recorded at 16bit/192kbps instead.

The following programs were used to create the sample files:

Table 1 - Programs/Codecs Used

Program	Program Version	MP3 Codec Used
Audacity	2.1.1	LAME 3.98.3
Adobe Audition CS3	CS3	Fraunhofer
Adobe Audition CC2014	CC2014	Fraunhofer
Adobe Audition CC2015	CC2015	Fraunhofer
Blade Encoder	0.82	Blade 0.82
dBpoweramp	15.3	LAME 3.99
ffmpeg	2.7.1	LAME 3.99.5
fpMP3 Encoder	1.0.0.2	fpMP3 1.0.0.2
Gogo Encoder	3.13	Gogo 3.13
Apple iTunes	12.3.0.44	Fraunhofer
Apple Logic Pro	9.1.8	Fraunhofer
LAME (Mac Terminal)	3.99.5	LAME 3.99.5
Mad Decoder	0.15.2b	Mad 0.15.2b
Mpg123 Decoder	1.22.0	Mpg123 1.22.0

Avid Pro Tools	11.3.1	Fraunhofer
SoX (Mac Terminal)	14.4.2	LAME 3.99.5
NCH Software Switch	4.85	LAME 3.97
Steinberg Wavelab (LAME)	8	LAME 3.98.4
Steinberg Wavelab (Fraunhofer)	8	Fraunhofer 4.1.3
Xing Encoder	1.5.0.5	Xing 1.5.0.5

The only settings modified in the programs was the bitrate. This was done in order to keep each new file at 16bit/128kbps CBR. All other settings were left at the default.

While there were 110 original files, file "H1_L3.mp3" appeared to be corrupt when opening in certain programs and displayed an abnormally high amount of zeroes. This outlier was omitted when testing the following programs:

- Adobe Audition CS3
- Avid Pro Tools
- Switch

CHAPTER III

RESULTS

Legend for Data Tables/Graphs:

Term	Definition
Initial Average	Average of both left and right channels at the **beginning** of the file.
Final Average	Average of both left and right channels at the **end** of the file.
ZLS	"Zero Level Samples", number of zero level samples.
Generation	Number of the .wav file generation.

3.1 Audacity

Generation	I	II	III	IV
Initial Average	3.2	3.5	3.1	3.4
Initial Standard Deviation	3.1	3.4	3.4	3.5
Final Average	0	0	0	0
Final Standard Deviation	0	0	0	0

Table 3 .1: Audacity

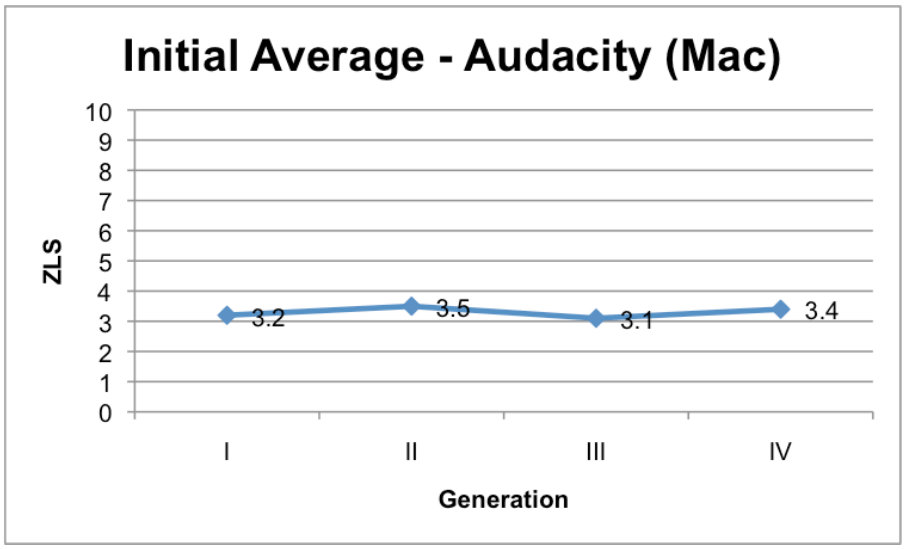


Figure 3.1a: Initial Average – Audacity

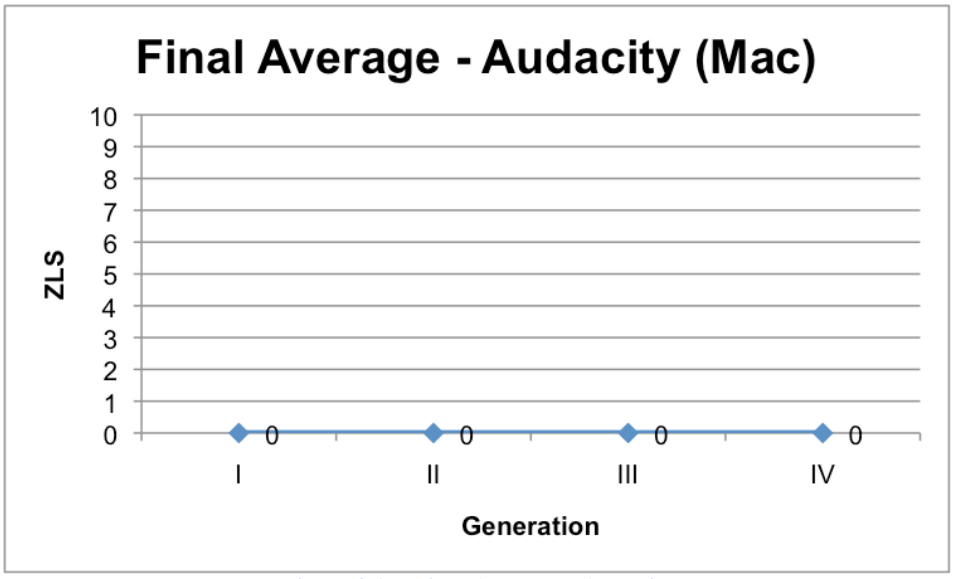


Figure 3.1b: Final Average – Audacity

3.2 ADOBE AUDITION CS3 (OUTLIER OMITTED)

Generation	I	II	III	IV
Initial Average	1211.9	1973.8	3035.3	4191
Initial Standard Deviation	901.5	879.9	909.9	901.4
Final Average	.2	1604.5	4385.9	7099.3
Final Standard Deviation	2.4	1976	3353.6	4625

Table 3.2: Adobe Audition CS3 (Outlier Omitted)

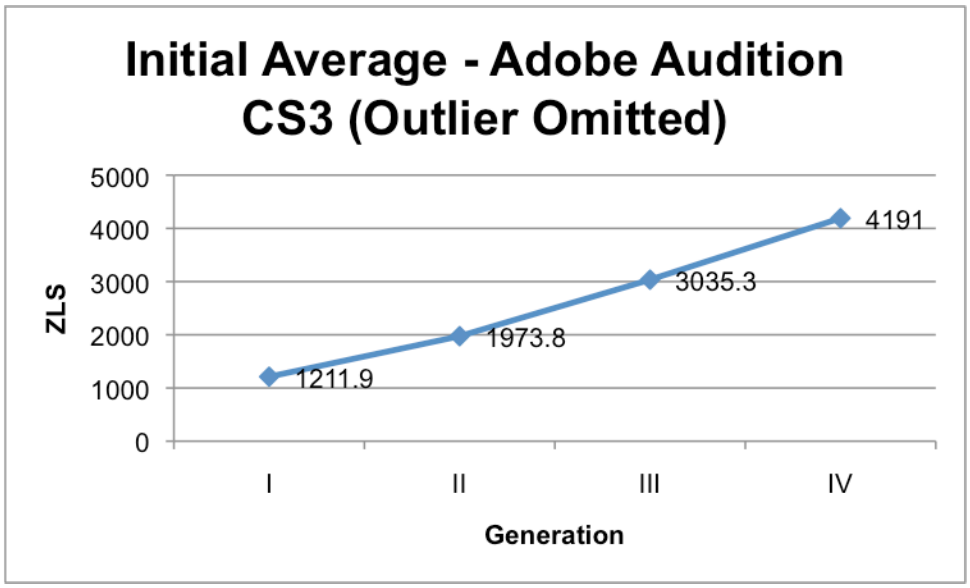


Figure 3.2a: Initial Average – Adobe Audition CS3 (Outlier Omitted)

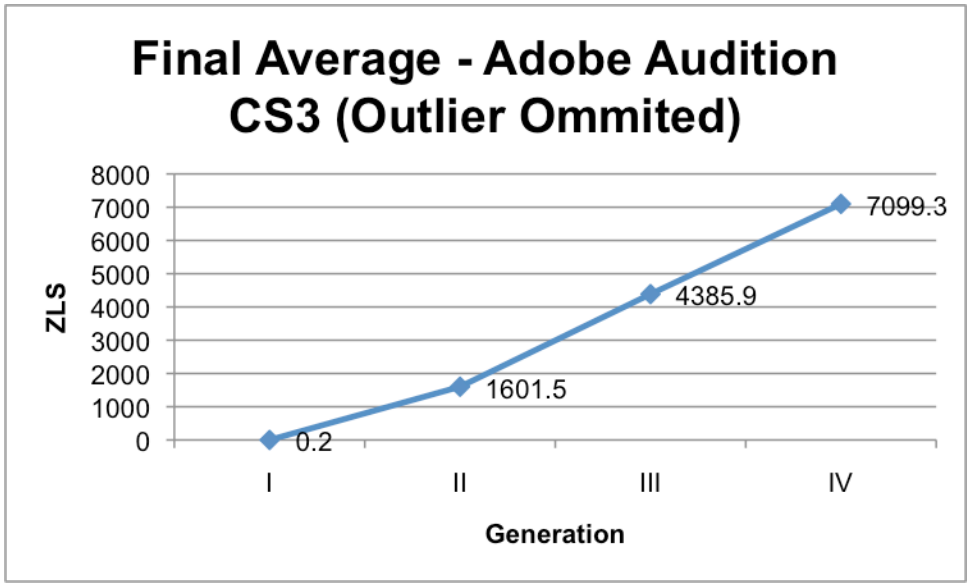


Figure 3.2b: Final Average – Adobe Audition CS3 (Outlier Omitted)

3.3 ADOBE AUDITION CC2014

Generation	I	II	III	IV
Initial Average	1206.9	848.8	843	837.7
Initial Standard Deviation	898.9	788.5	785.1	785.1
Final Average	.2	0	0	0
Final Standard Deviation	2.4	.2	.2	.2

Table 3.3: Adobe Audition CC2014

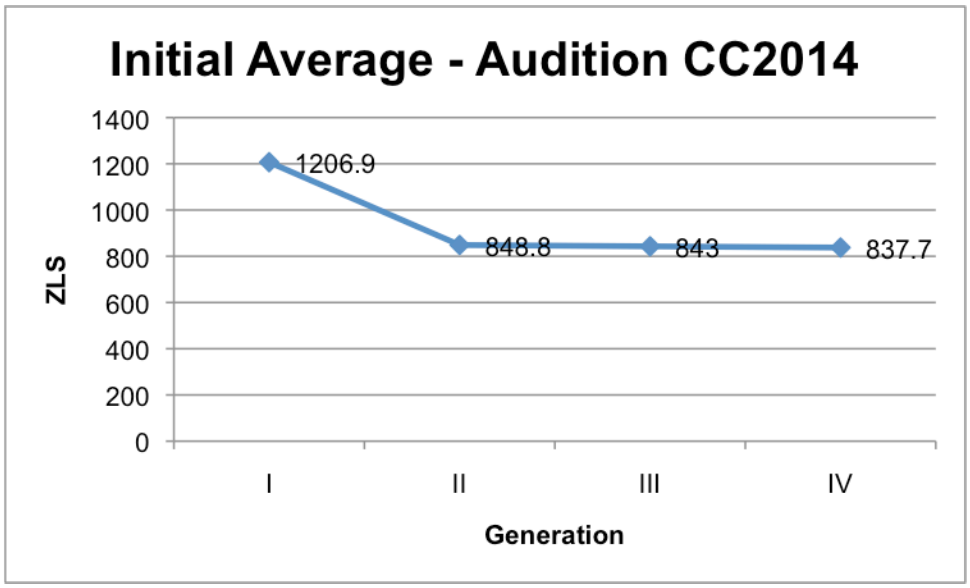


Figure 3.3a: Initial Average – Adobe Audition CC2014

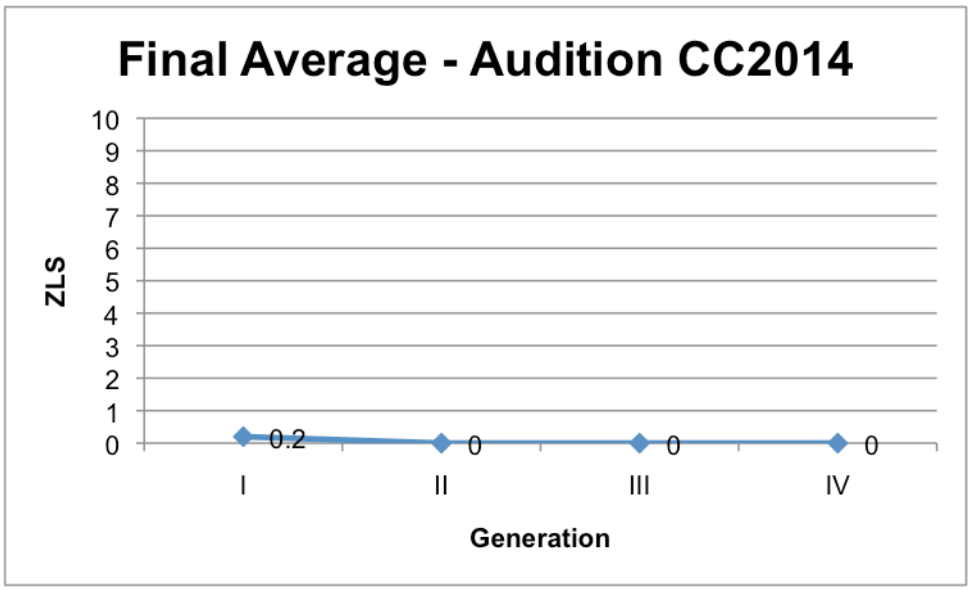


Figure 3.3b: Final Average – Adobe Audition CC2014

3.4 ADOBE AUDITION CC2015

Generation	I	II	III	IV
Initial Average	1206.9	848.8	843	837.7
Initial Standard Deviation	898.9	788.5	785.1	785.1
Final Average	.2	0	0	0
Final Standard Deviation	2.4	.2	.2	.2

Table 3.4: Adobe Audition CC2015

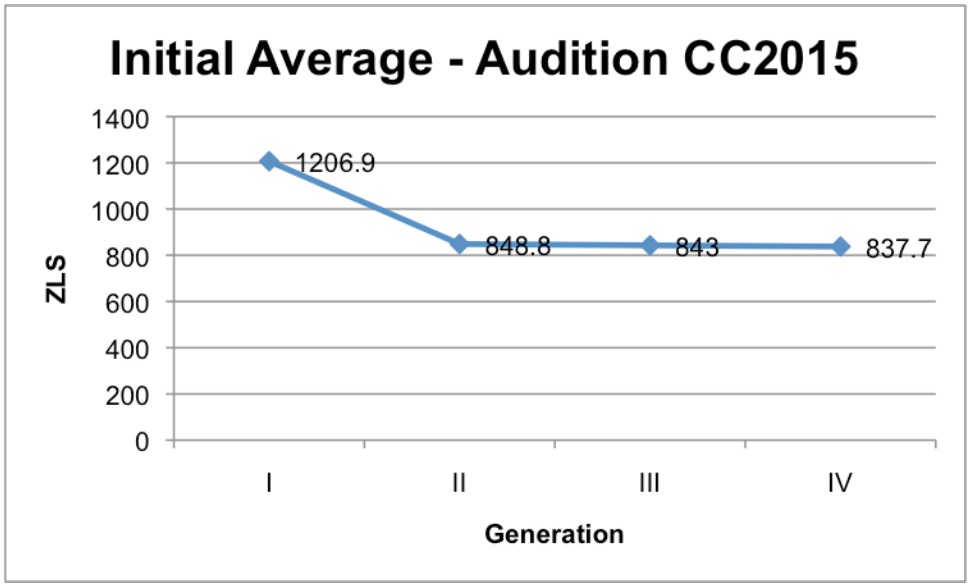


Figure 3.4a: Initial Average – Adobe Audition CC2015

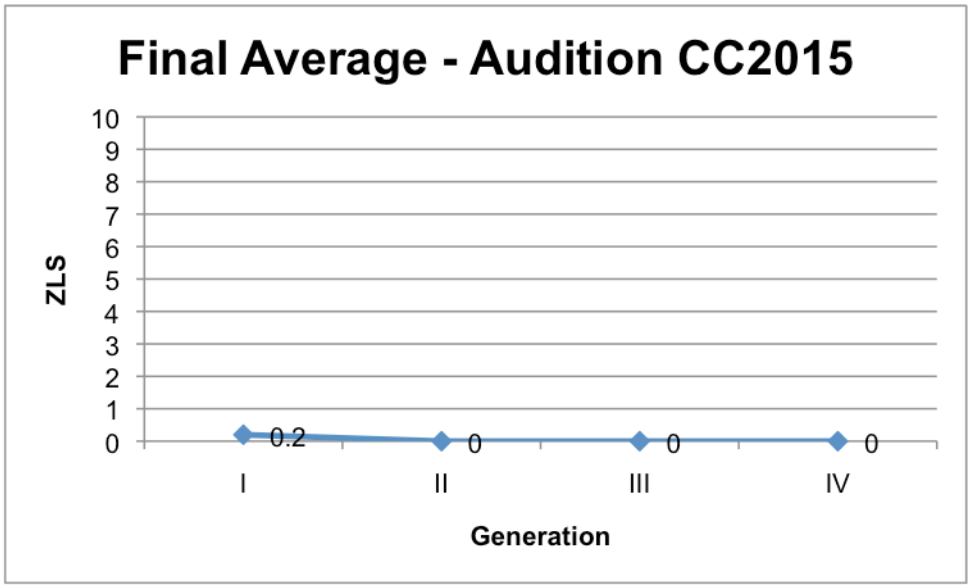


Figure 3.4b: Final Average – Adobe Audition CC2015

3.5 BLADE ENCODER

Generation	I	II	III	IV
Initial Average	1206.9	2109.2	3097.7	4106.7
Initial Standard Deviation	898.9	954.8	953.6	956.4
Final Average	0	0	0	0
Final Standard Deviation	.3	0	0	0

Table 3.5: Blade Encoder

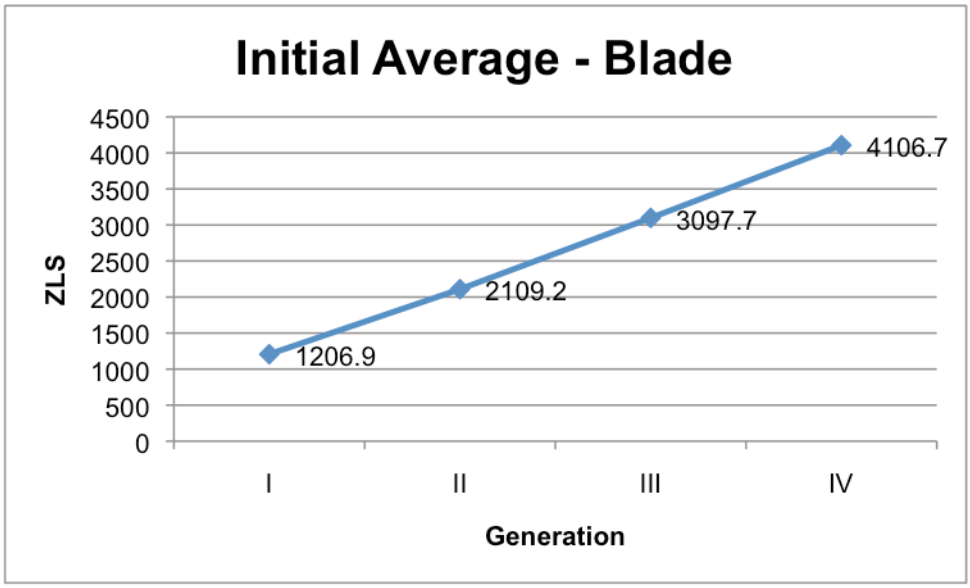


Figure 3.5a: Initial Average – Blade Encoder

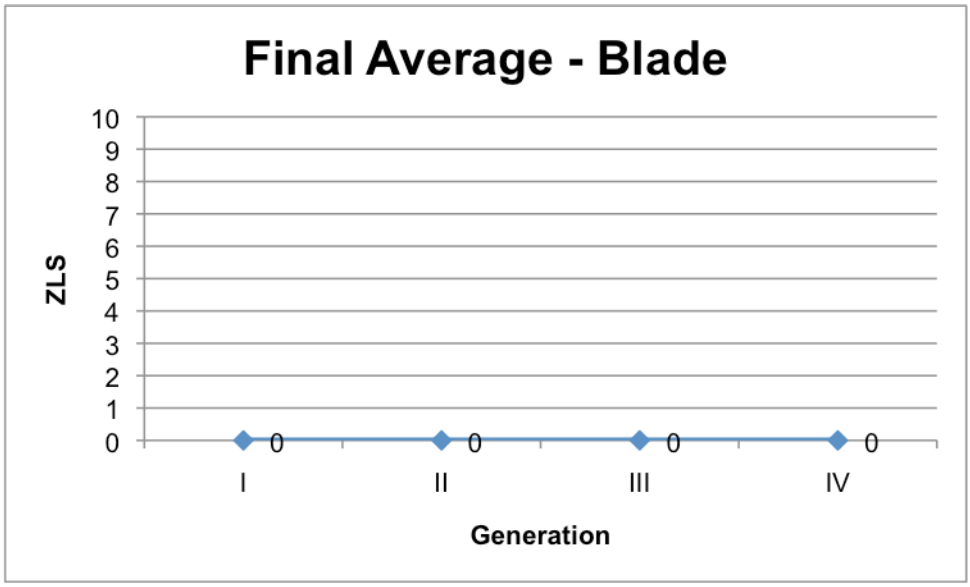


Figure 3.5b: Final Average – Blade Encoder

3.6 DBPOWERAMP

Generation	I	II	III	IV
Initial Average	1206.9	1206.9	1054.8	1037.1
Initial Standard Deviation	898.9	989.9	907.6	905.3
Final Average	.1	.1	.1	.1
Final Standard Deviation	.3	.3	.3	.1

Table 3.6: dBpoweramp

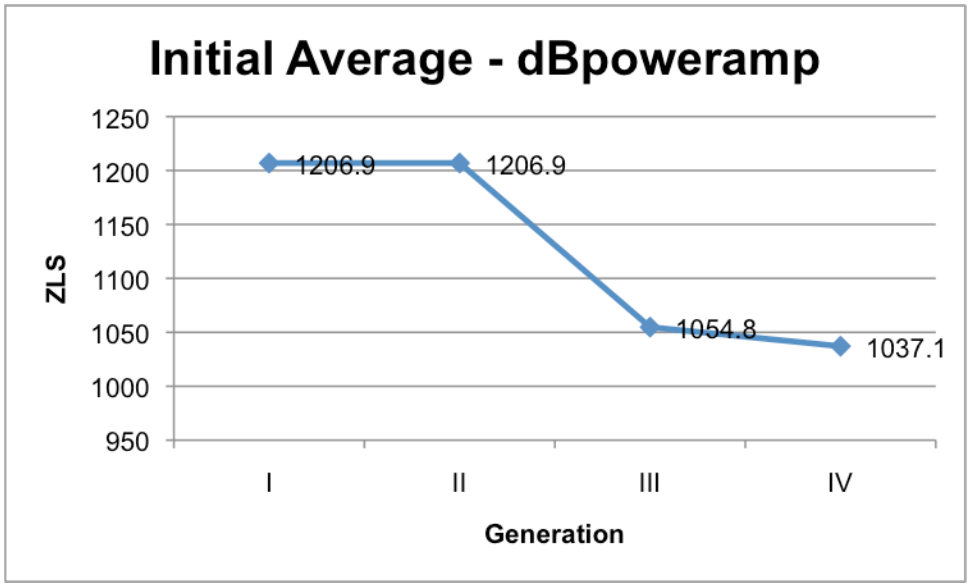


Figure 3.6a: Initial Average – dBpoweramp

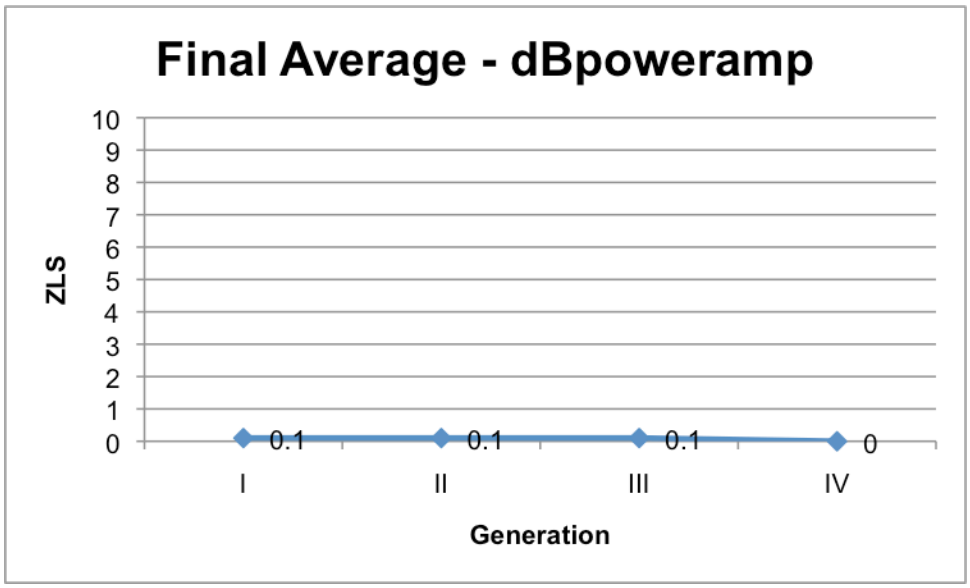


Figure 3.6b: Final Average – dBpoweramp

3.7 FFMPEG

Generation	I	II	III	IV
Initial Average	825.1	347	21.2	7.1
Initial Standard Deviation	806.9	378.8	42.2	10.7
Final Average	0	.1	361.4	498
Final Standard Deviation	.1	.5	133.6	235

Table 3.7: ffmpeg

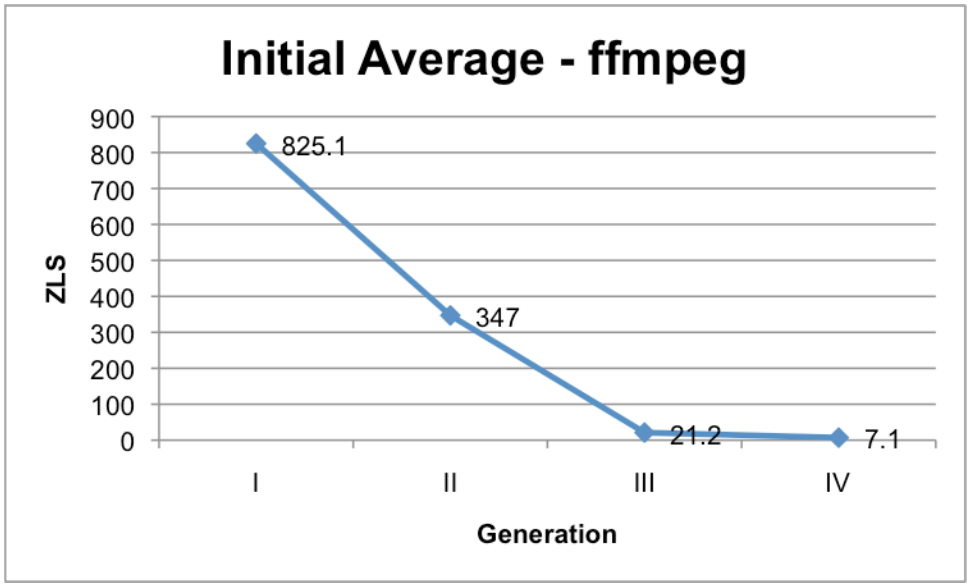


Figure 3.7a: Initial Average - ffmpeg

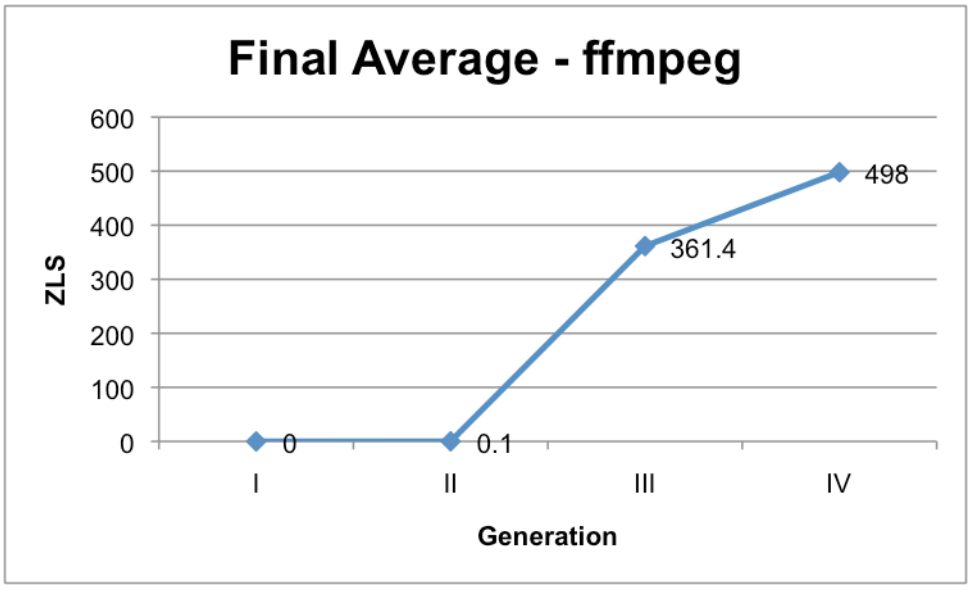


Figure 3.7b: Final Average - ffmpeg

3.8 FPMP3

Generation	I	II	III	IV
Initial Average	1206.9	694.6	375.7	112.2
Initial Standard Deviation	898.9	711.9	445.8	195.7
Final Average	.1	0	0	0
Final Standard Deviation	.3	0	0	0

Table 3.8: fpMP3

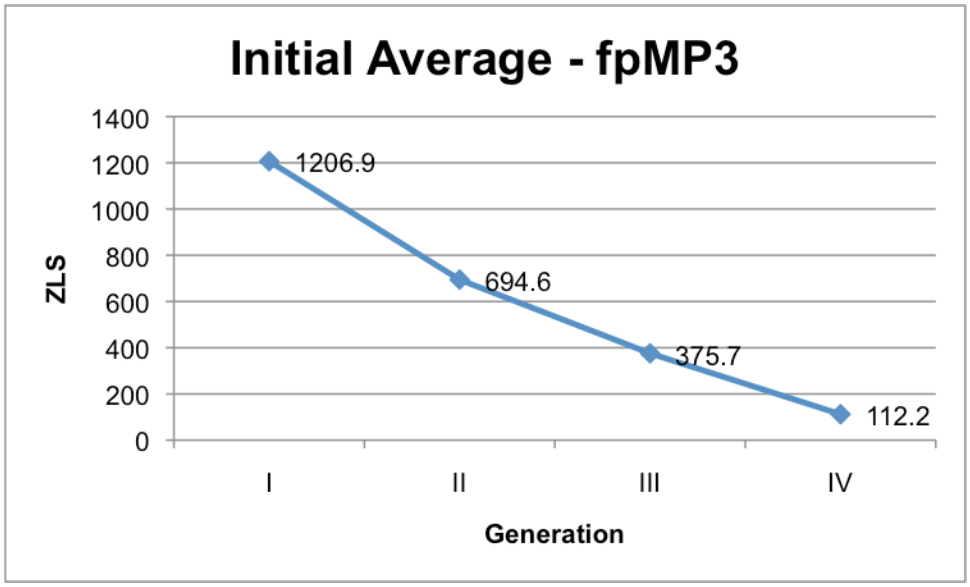


Figure 3.8a: Initial Average – fpMP3

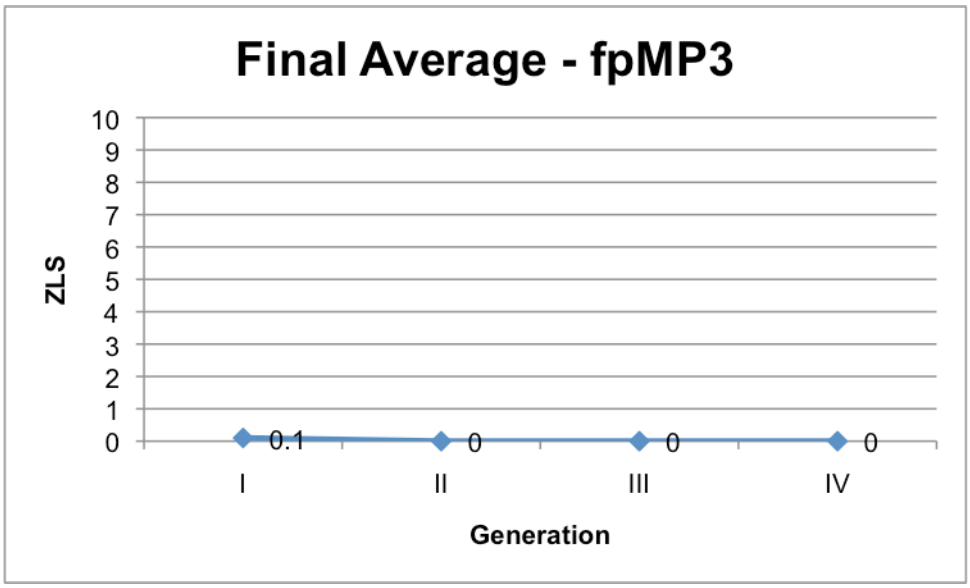


Figure 3.8b: Final Average – fpMP3

3.9 GOGO

Generation	I	II	III	IV
Initial Average	1206.9	694.6	375.7	112.2
Initial Standard Deviation	898.9	711.9	445.8	195.7
Final Average	.1	0	0	0
Final Standard Deviation	.3	0	0	0

Table 3.9: Gogo

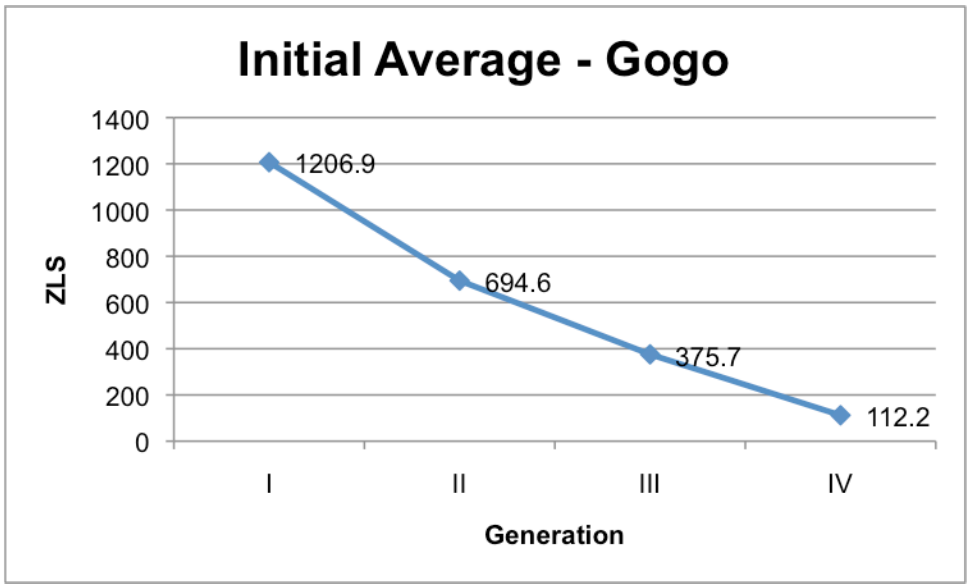


Figure 3.9a: Initial Average – Gogo

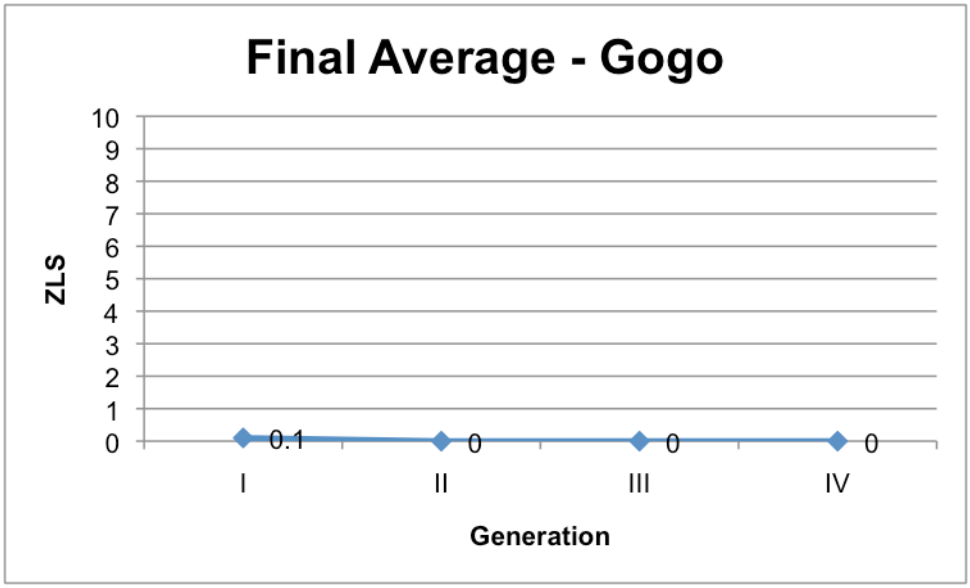


Figure 3.9b: Final Average – Gogo

3.10 APPLE ITUNES

Generation	I	II	III	IV
Initial Average	502.9	460.3	441.7	433.8
Initial Standard Deviation	549.7	515	503.7	498.3
Final Average	0	0	0	0
Final Standard Deviation	0	0	0	0

Table 3.10: iTunes

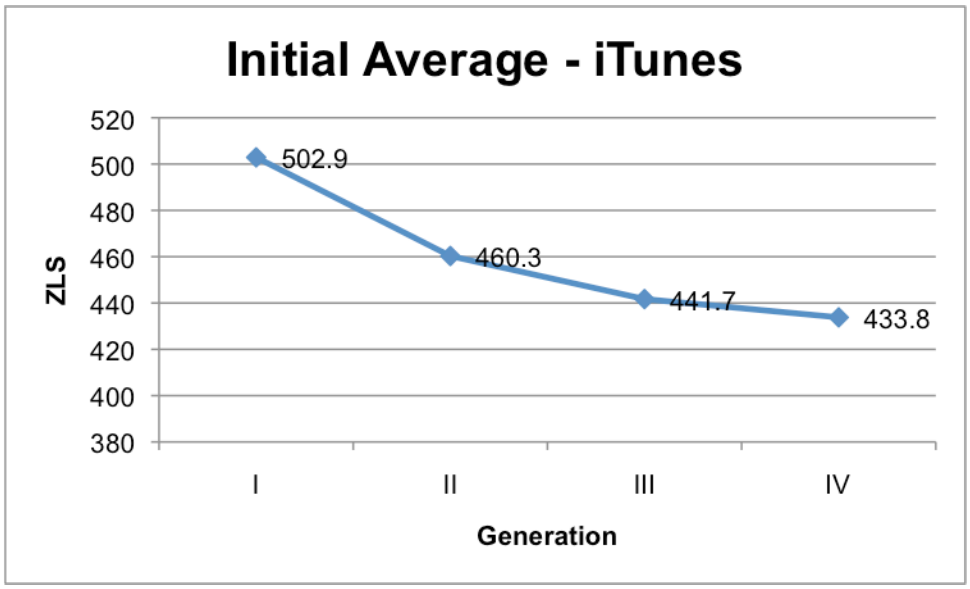


Figure 3.10a: Initial Average – iTunes

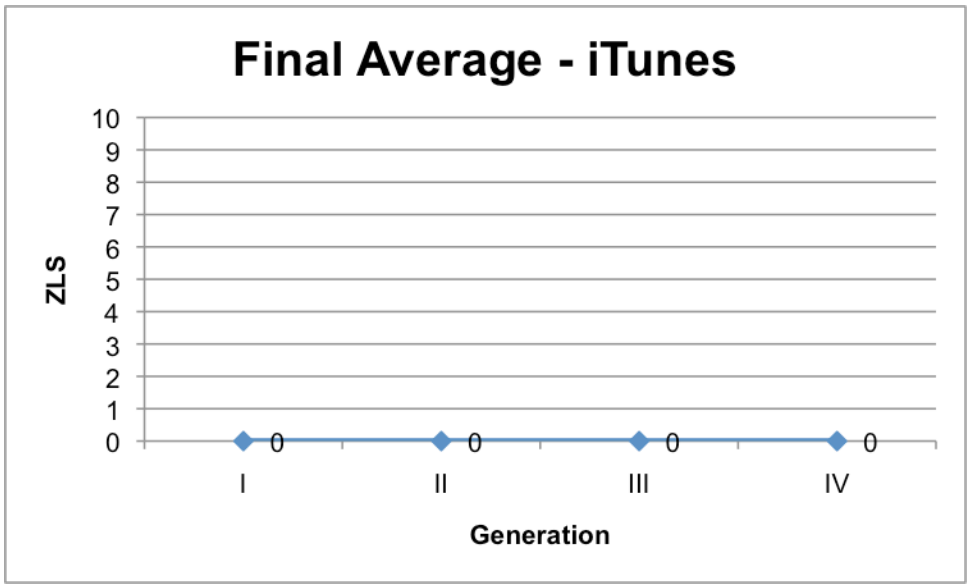


Figure 3.10b: Final Average – iTunes

3.11 APPLE LOGIC PRO

Generation	I	II	III	IV
Initial Average	786.2	691.6	669.7	658.9
Initial Standard Deviation	782.5	731.5	717.6	709.6
Final Average	492.2	4.4	.4	.9
Final Standard Deviation	22.4	31.7	6.1	11

Table 3.11: Apple Logic Pro

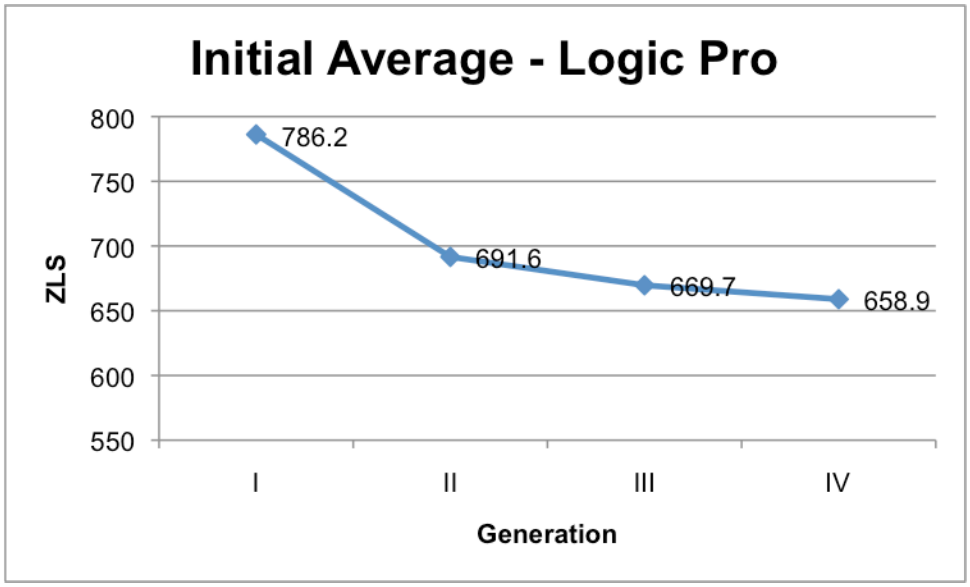


Figure 3.11a: Initial Average – Apple Logic Pro

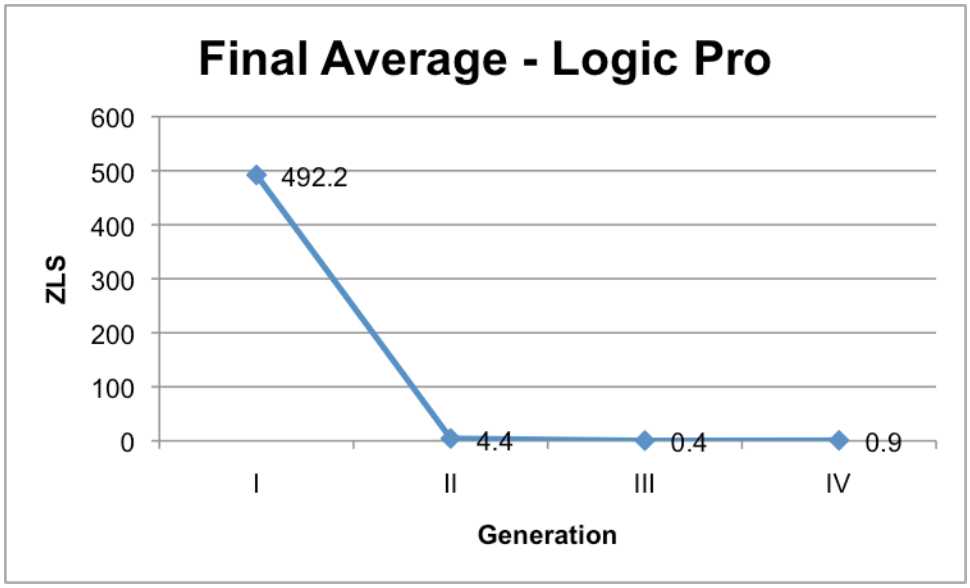


Figure 3.11b: Final Average – Apple Logic Pro

3.12 LAME (Mac Terminal)

Generation	I	II	III	IV
Initial Average	787.4	723.5	694.4	672.6
Initial Standard Deviation	783.2	745.2	723.9	715
Final Average	0	0	0	0
Final Standard Deviation	.3	.1	.1	.3

Table 3.12: LAME 3.99.5 (Mac Terminal)

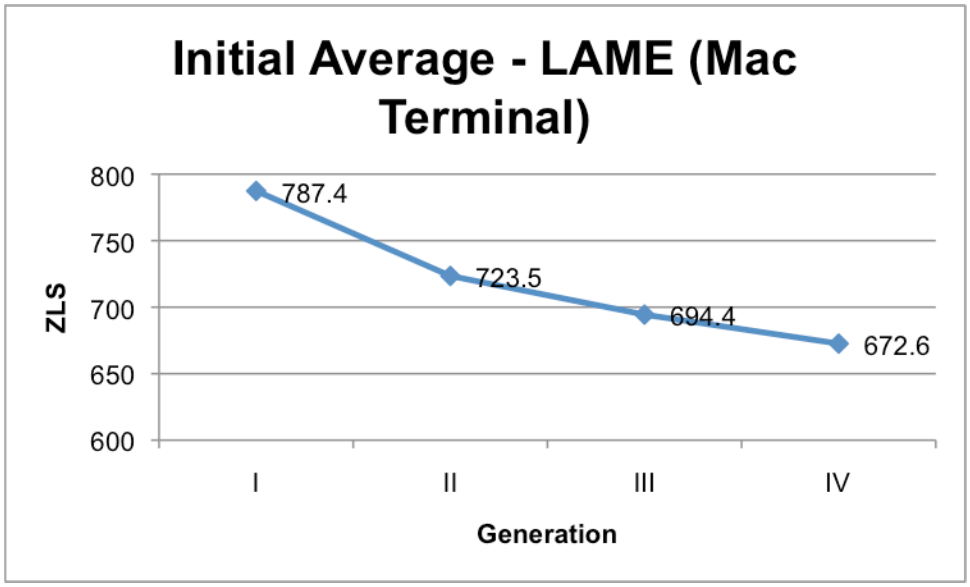


Figure 3.12a: Initial Average – LAME (Mac Terminal)

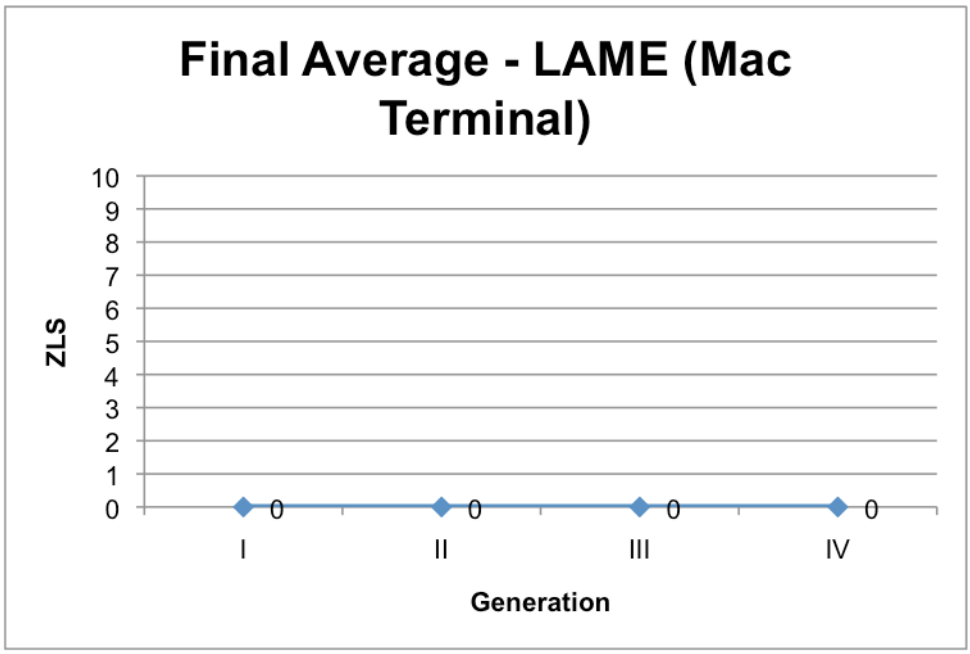


Figure 3.12b: Final Average – LAME (Mac Terminal)

3.13 MAD

Generation	I	II	III	IV
Initial Average	1	1	1	1
Initial Standard Deviation	0	0	0	0
Final Average	0	.7	1.1	1.4
Final Standard Deviation	.2	2.9	4	4.7

Table 3.13: Mad

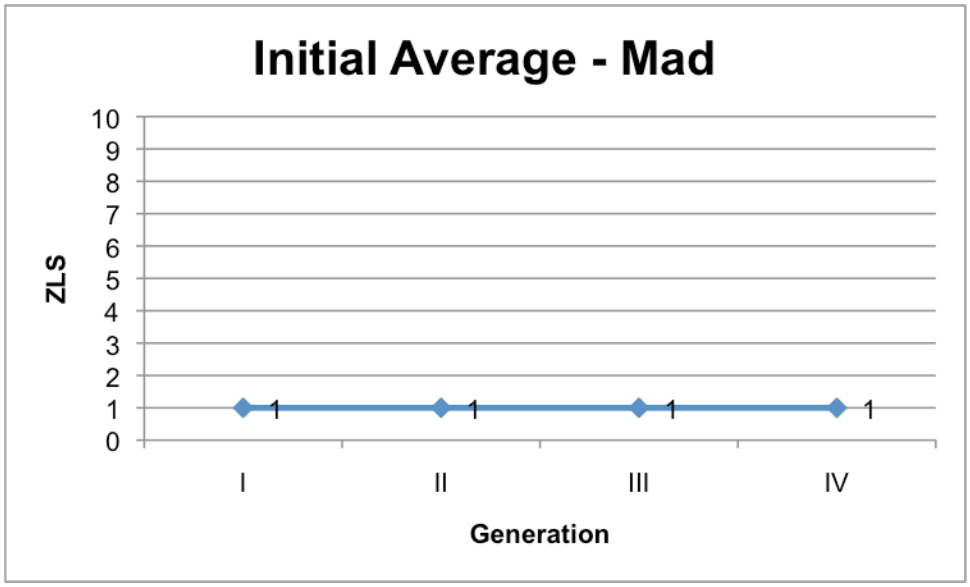


Figure 3.13a: Initial Average – Mad

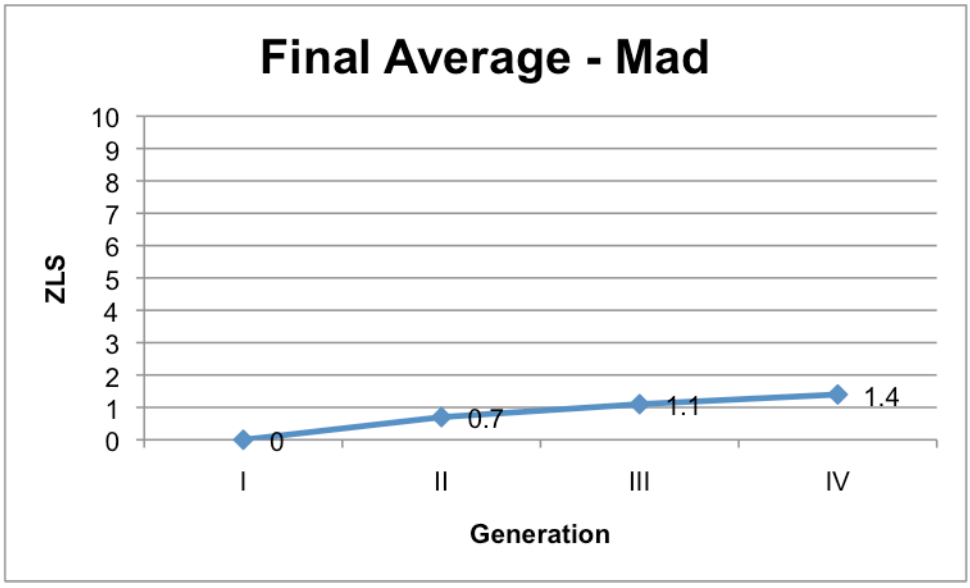


Figure 3.13b: Final Average – Mad

3.14 MPG123

Generation	I	II	III	IV
Initial Average	881.8	1994	3048.8	4144.5
Initial Standard Deviation	874.2	905.1	924.8	927.2
Final Average	.5	324.5	1178.5	2124.8
Final Standard Deviation	2.1	262.3	351.7	421.8

Table 3.14: mpg123

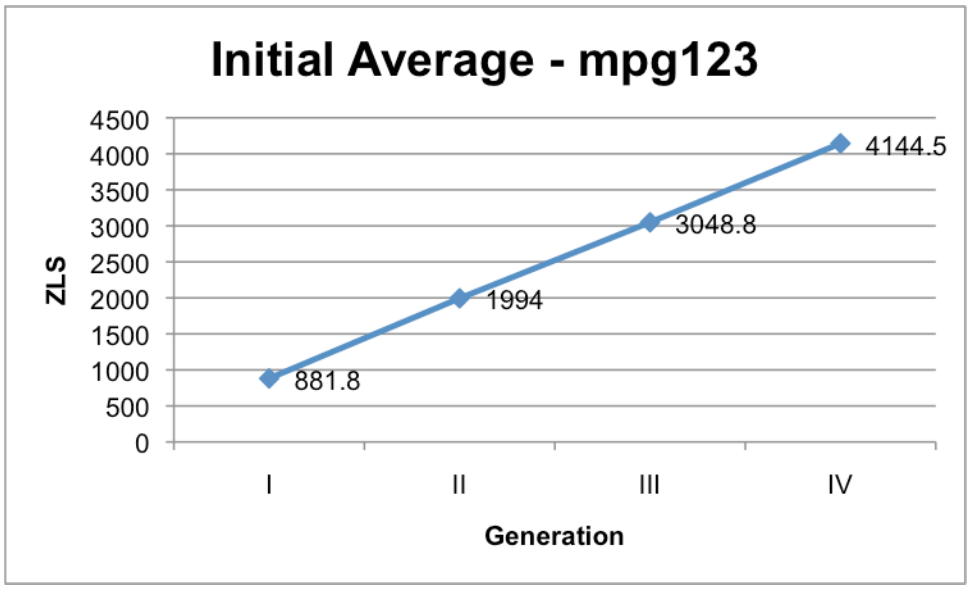


Figure 3.14a: Initial Average – mpg123

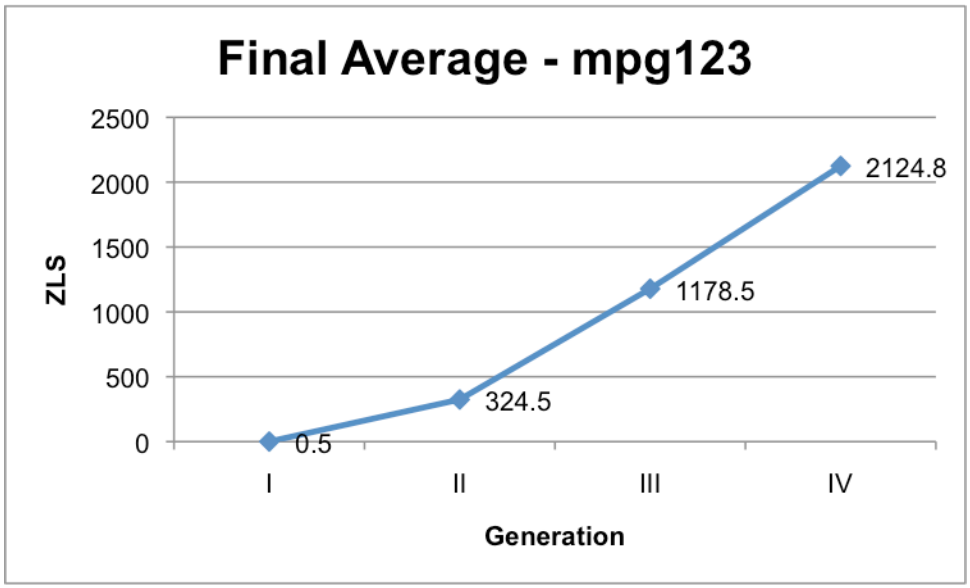


Figure 3.14b: Final Average – mpg123

3.15 AVID PRO TOOLS (OUTLIER OMITTED)

Generation	I	II	III	IV
Initial Average	793.5	1060.1	1537.4	2168.5
Initial Standard Deviation	784.3	817.8	932	924.6
Final Average	2314.1	3165.2	3630.4	4845
Final Standard Deviation	1688.1	1727.1	2516.2	2792

Table 3.15: Avid Pro Tools (Outlier Omitted)

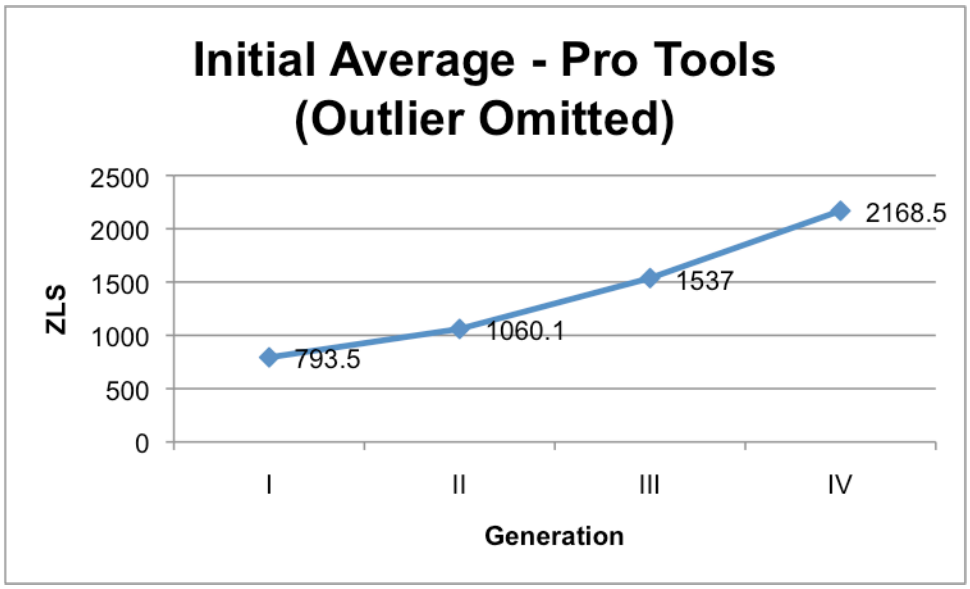


Figure 3.15a: Initial Average – Pro Tools (Outlier Omitted)

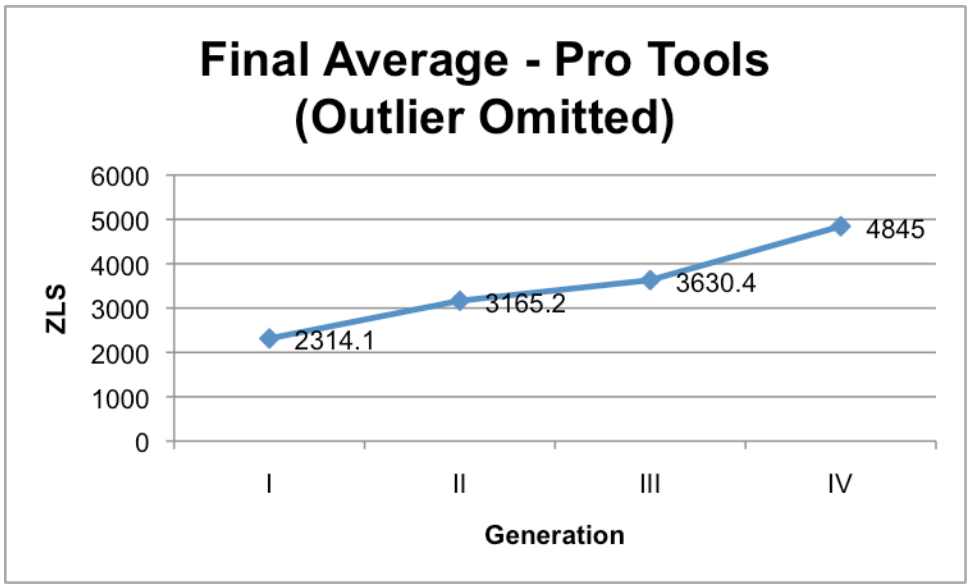


Figure 3.15a: Final Average – Pro Tools (Outlier Omitted)

3.16 SOX

Generation	I	II	III	IV
Initial Average	1206.9	2111.3	3178.9	4261.5
Initial Standard Deviation	898.9	968.4	967	964.4
Final Average	0	0	0	0
Final Standard Deviation	0	0	0	0

Table 3.16: SoX

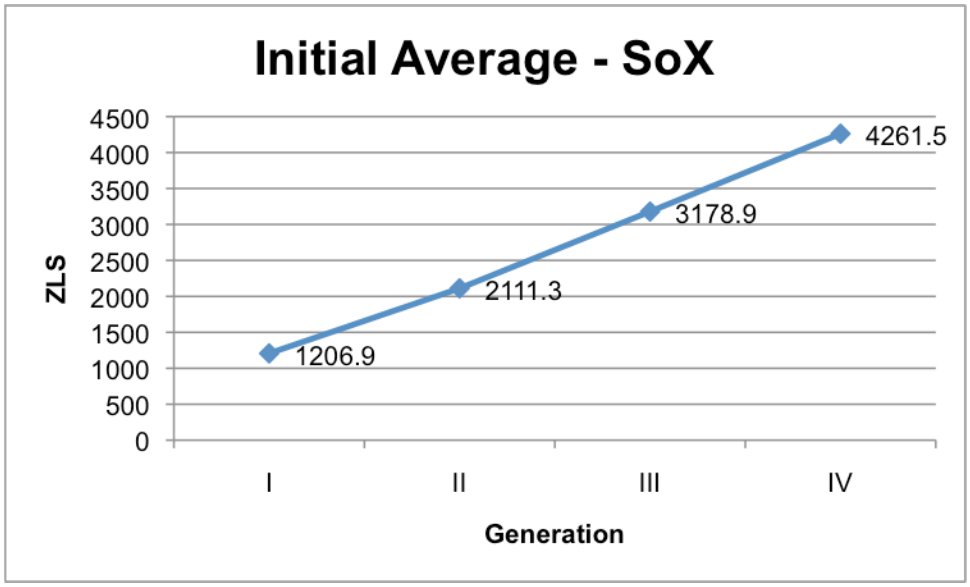


Figure 3.16a: Initial Average - SoX

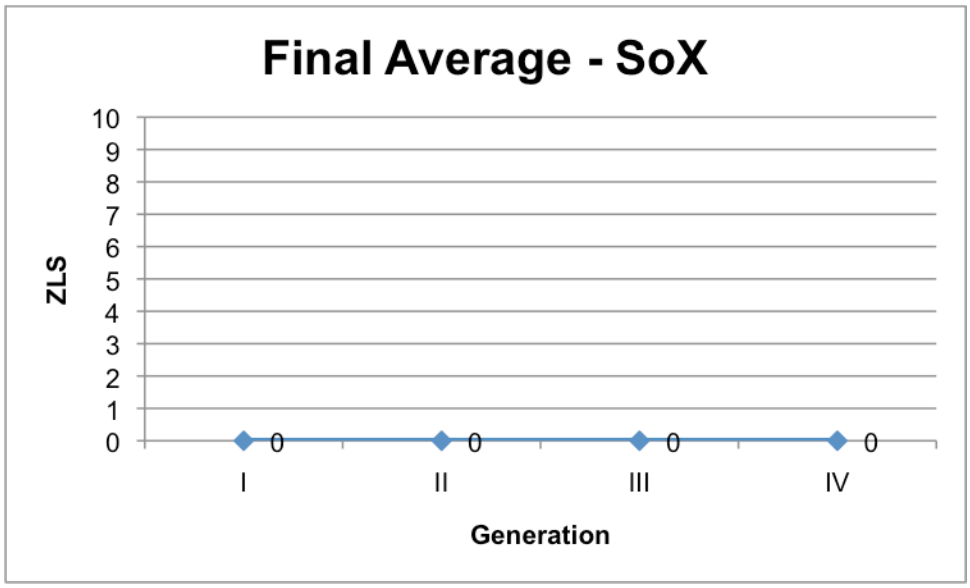


Figure 3.16b: Final Average - SoX

4.17 SWITCH (OUTLIER OMITTED)

Generation	I	II	III	IV
Initial Average	1.6	1	1.1	5.1
Initial Standard Deviation	2	0	.4	21.2
Final Average	.1	0	0	0
Final Standard Deviation	1.1	0	0	0

Table 3.17: Switch (Outlier Omitted)

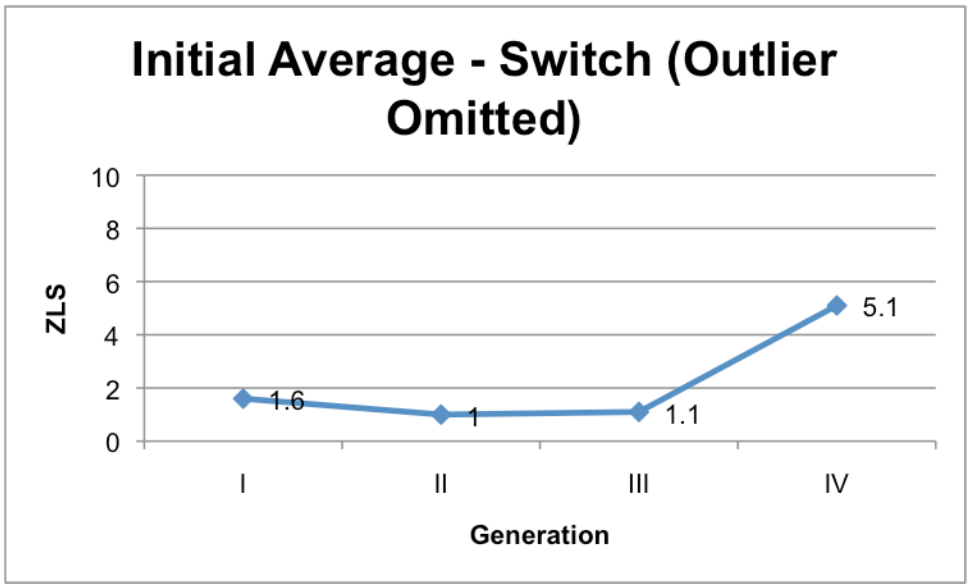


Figure 3.17a: Initial Average – Switch (Outlier Omitted)

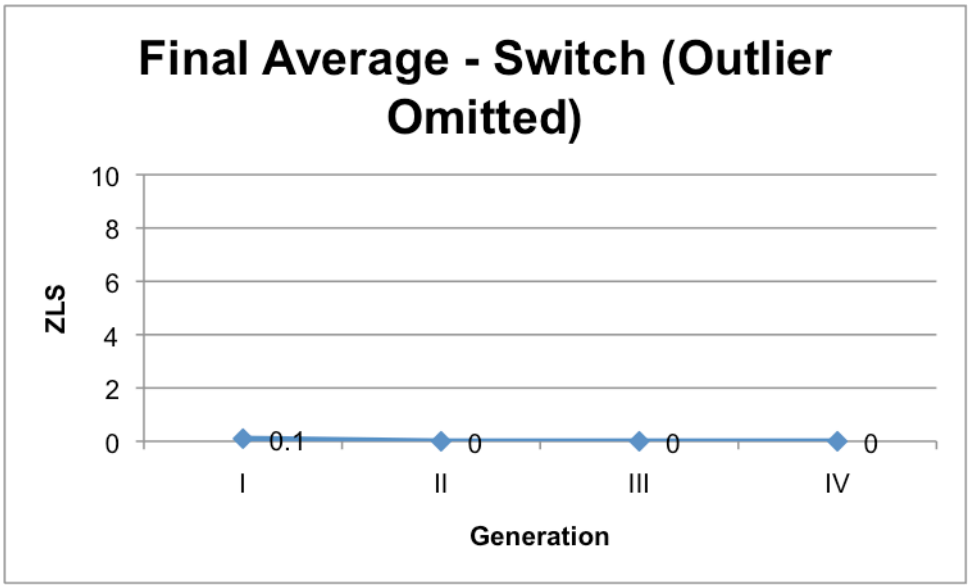


Figure 3.17b: Final Average – Switch (Outlier Omitted)

3.18 WAVELAB (FRAUNHOFER)

Generation	I	II	III	IV
Initial Average	1256.5	2171.7	3284.8	4475.9
Initial Standard Deviation	886	883.5	917.1	938.7
Final Average	1.6	500.7	1393.6	2383.8
Final Standard Deviation	7.1	247.1	397	469.6

Table 3.18: Wavelab (Fraunhofer)

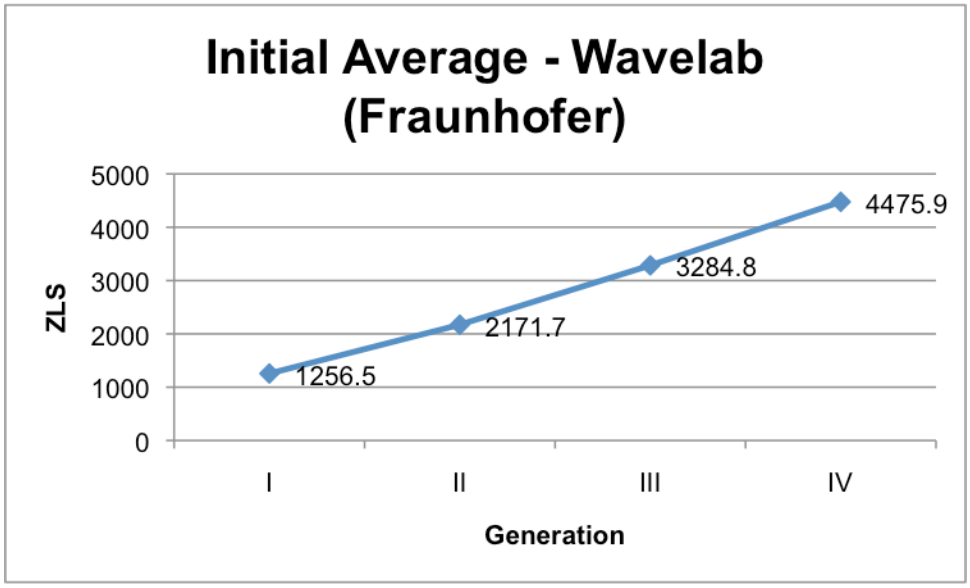


Figure 3.18a: Initial Average – Wavelab (Fraunhofer)

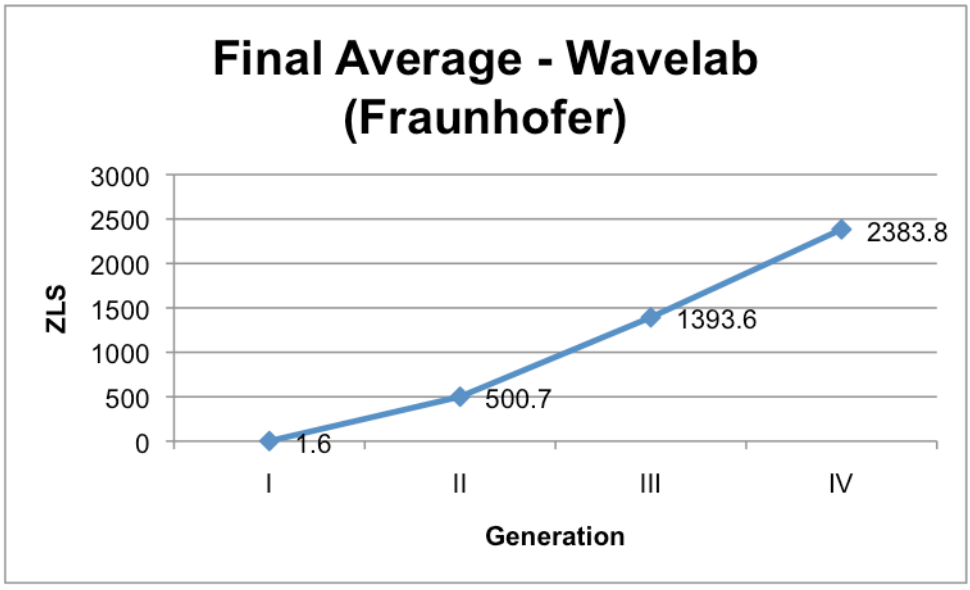


Figure 3.18b: Final Average – Wavelab (Fraunhofer)

3.19 WAVELAB (LAME)

Generation	I	II	III	IV
Initial Average	1256.5	2258.2	3326.5	4436
Initial Standard Deviation	886	907.2	875.8	870.2
Final Average	1.6	1.5	3.2	4.6
Final Standard Deviation	7.1	6.9	13.5	18.1

Table 3.19: Wavelab (Lame)

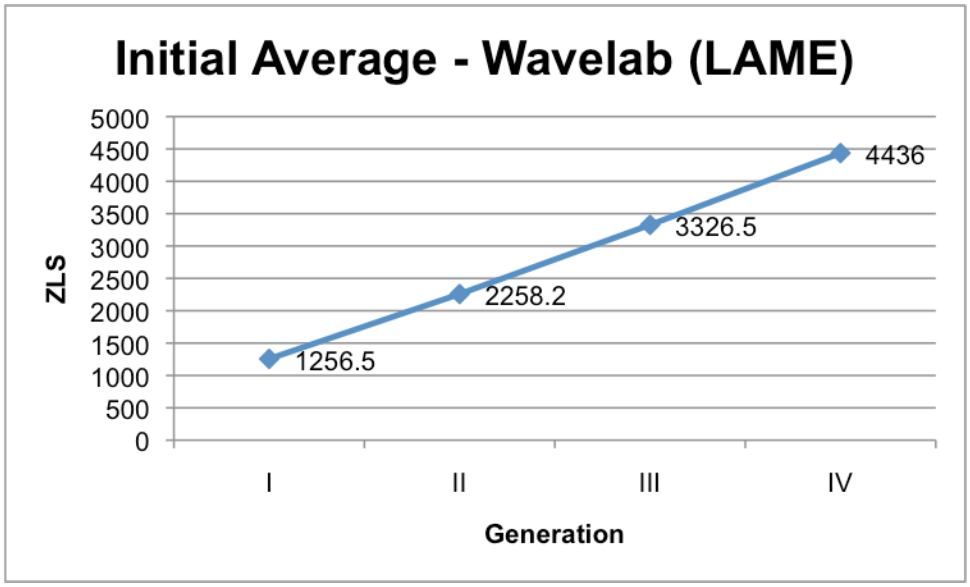


Figure 3.19a: Initial Average – Wavelab (LAME)

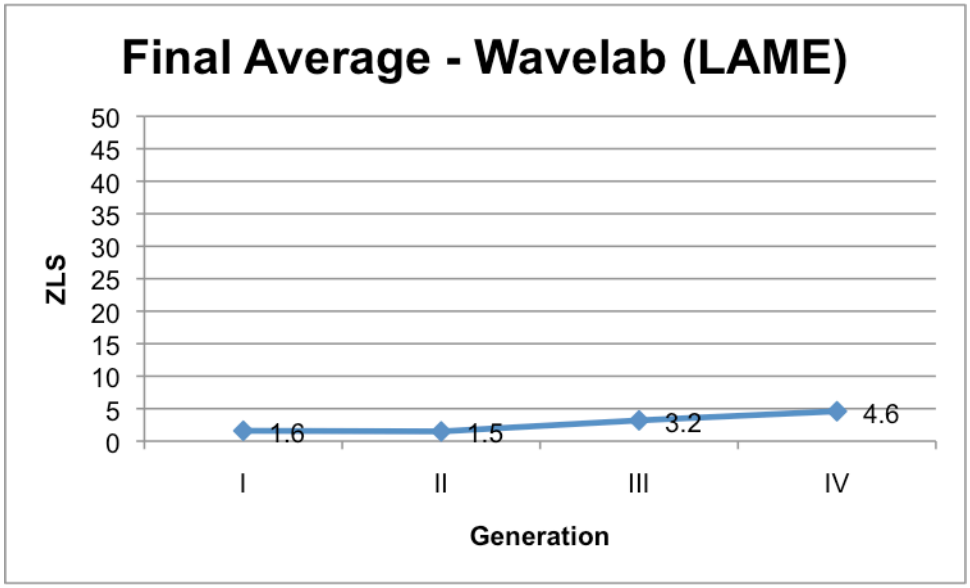


Figure 3.19b: Final Average – Wavelab (LAME)

3.20 XING

Generation	I	II	III	IV
Initial Average	1206.9	1842.8	2824.9	3831.5
Initial Standard Deviation	898.9	807.6	866.1	877.1
Final Average	.1	0	.1	0
Final Standard Deviation	.3	0	.4	.2

Table 3.20: Xing

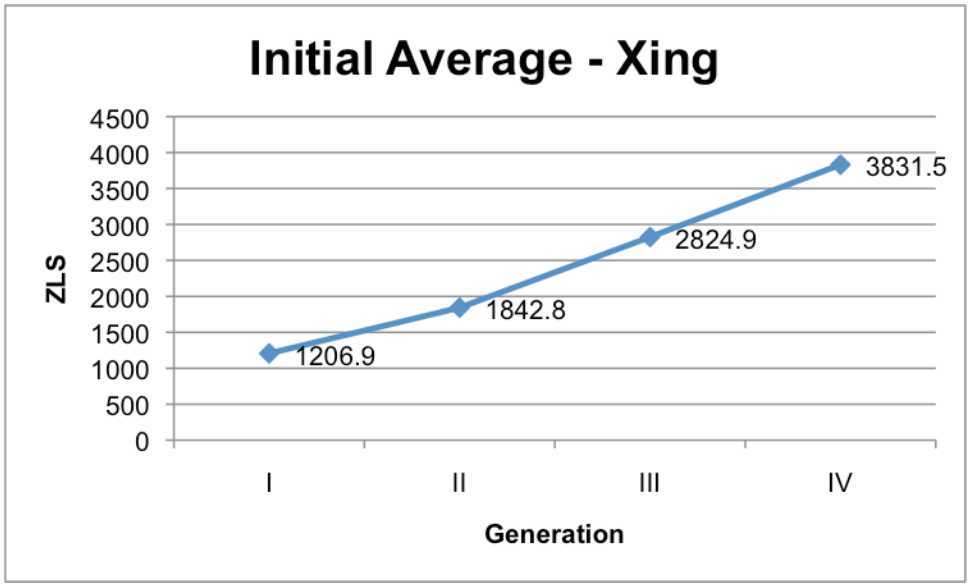


Figure 3.20a: Initial Average – Xing

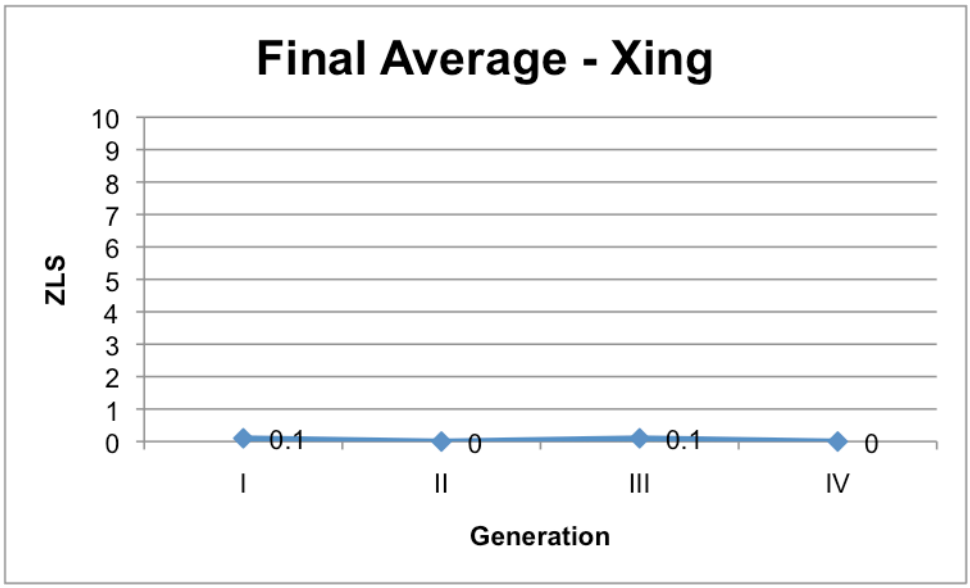


Figure 3.20b: Final Average – Xing

3.21 ALL LAME PROGRAMS

This section compiles all programs that used LAME. The chart "Initial Average – All LAME Programs" and "Final Average – All LAME programs" show all LAME programs superimposed.

Generation	I	II	III	IV
Audacity	825.1	347	21.2	7.1
Wavelab LAME	1256.5	2258.2	3326.5	4436
Switch	1.6	1	1.1	5.1
ffmpeg	825.1	347	21.2	7.1
dBpoweramp	1206.9	1206.9	1054.8	1037.1
LAME	787.4	723.5	694.4	672.6
SoX	1206.9	2111.3	3178.9	4261.5
Average	**755.4**	**950.2**	**1182.9**	**1489**

Table 3.21: Initial Average of all LAME programs

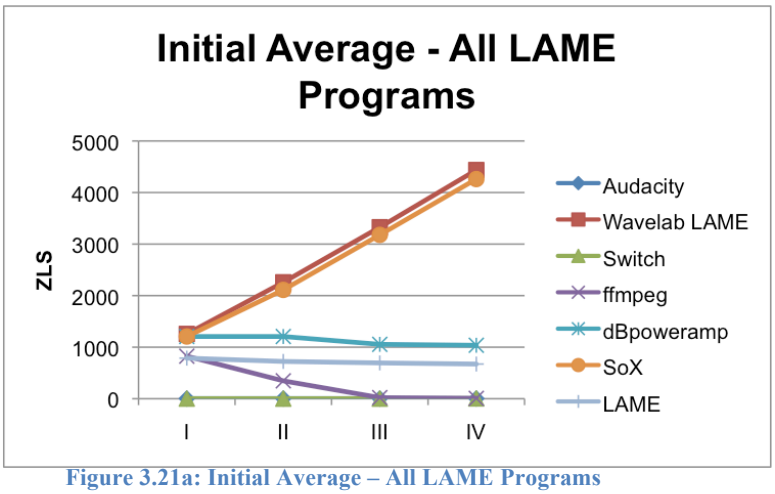


Figure 3.21a: Initial Average – All LAME Programs

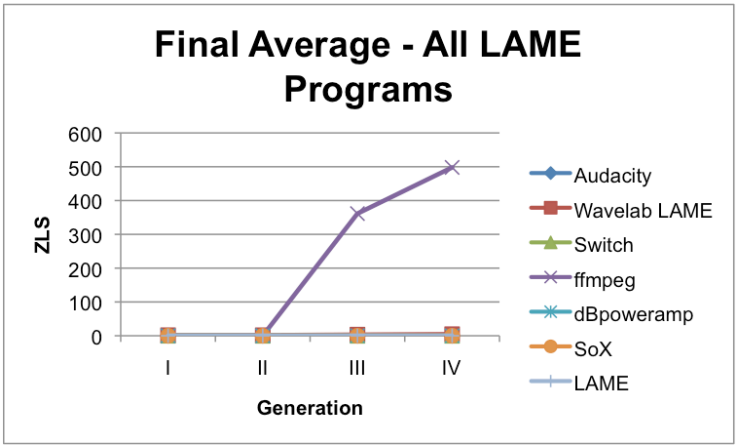


Figure 3.21b: Final Average – All LAME

3.22 ALL FRAUNHOFER PROGRAMS

This section compiles all programs that used Fraunhofer. The chart "Initial Average – All Fraunhofer Programs" and "Final Average – All Fraunhofer programs" show all Fraunhofer programs superimposed.

Generation	**I**	**II**	**III**	**IV**
Audition CS3	1211.9	1973.8	3035.3	4191
Audition CC2014	1206.9	848.8	843	837.7
Audition CC2015	1206.9	848.8	843	837.7
iTunes	502.9	460.3	441.7	433.8
Logic Pro	786.2	691.6	669.7	658.9
Pro Tools	793.5	1060.1	1537	2168.5
Wavelab (Fraunhofer)	1256.5	2171.7	3284.8	4475.9
Average	**995**	**1150.7**	**1522.1**	**1943.4**

Table 3.22: All Fraunhofer Programs

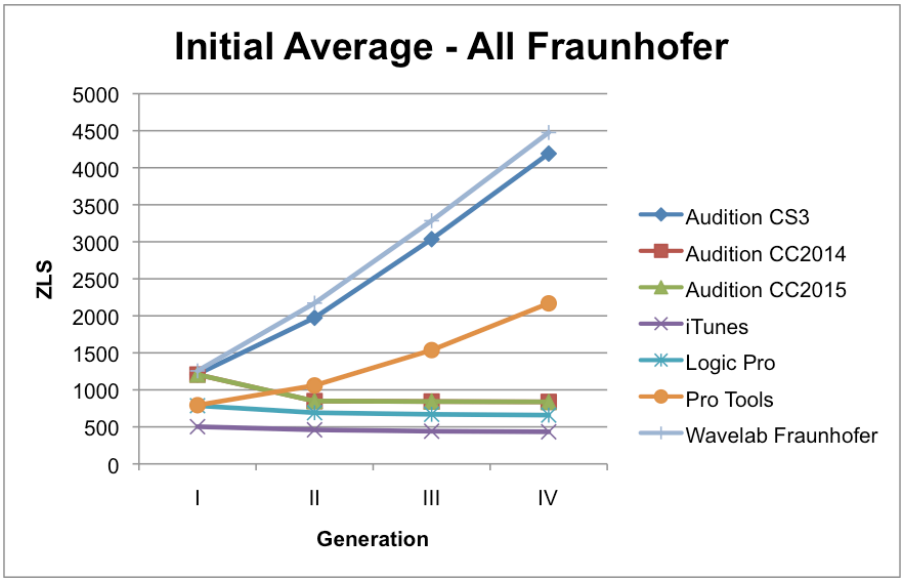


Figure 3.22a: Initial Average – All Fraunhofer Programs

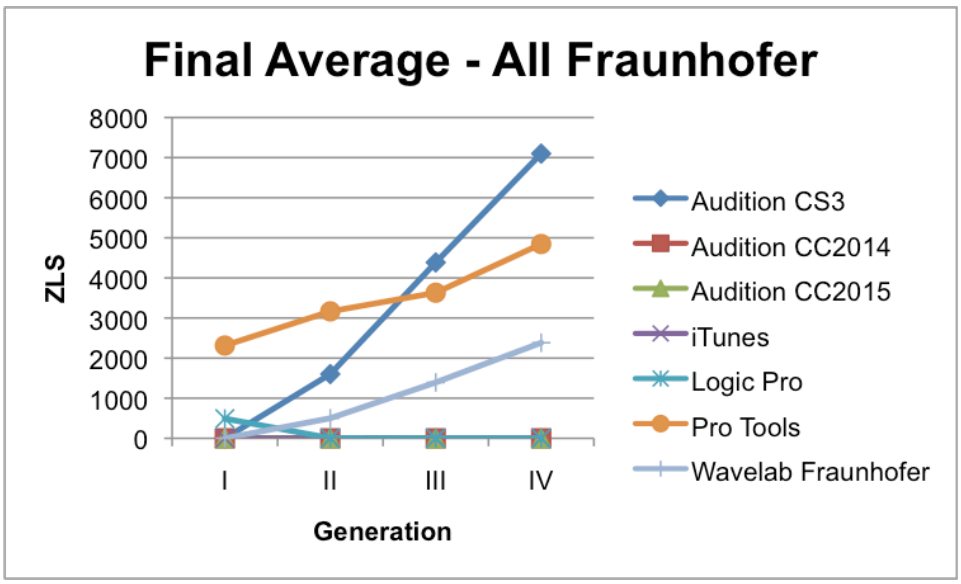


Figure 3.22b: Final Average – All Fraunhofer Programs

Chapter IV

DISCUSSION

While the original hypothesis of this study was that the number of zero-level samples at the beginning and end of each file would increase with each additional compression, that was found to not always be the case. While some programs showed the general upward trend of adding more zeroes after each additional compression, many did not. The ones that did not varied from having no zeroes on the files whatsoever to starting with a substantial amount then going down after each recompression.

Programs/Codecs that behaved as expected (number of ZLS grew after each recompression):

- Wavelab (LAME)
- Wavelab (Fraunhofer)
- Pro Tools
- Audition CS3
- Blade
- Mpg123
- Xing
- SoX

Programs that did not behave as expected (number of ZLS did not grow after each recompression):

- ffmpeg
- Switch
- Audacity

- Logic Pro
- dBpoweramp
- iTunes
- Audition CC2014
- Audition CC2015
- Mad
- fpMP3
- Gogo
- LAME (Mac Terminal)

Only 8 out of the 20 programs (40%) of the programs behaved as expected vs. 12 out of 20 (60%) which did not. Among the programs that did not behave as expected, some had a high initial number of zeroes then decreased, whereas some had a nominally small number of zeroes throughout each re-compression.

The programs that had a high initial number of zero-level samples and decreased throughout were:

- ffmpeg
- Logic Pro
- dBpoweramp
- iTunes
- Adobe Audition CC2014
- Adobe Audition CC2015
- fpMP3

- Gogo
- LAME (Mac Terminal)

The programs that had a nominally small number of zeroes throughout were:

- Switch
- Audacity
- Mad

One possible reason behind the zero-level samples is for gapless playback information[1]. Depending on the exact codec and specs used, the encoder and decoder will add a number of samples. This is used as a buffer by many different audio players to ensure "gapless playback". However, in more recent revisions of MP3 codecs, developers have been able to remove this buffer to some extent. As LAME states in their FAQ:

"Starting with LAME 3.55, we have a new MDCT/filterbank routine written by Takehiro Tominaga with a 48 sample delay. With even more rewriting, this could be reduced to 0."

Other more popular codecs such as Fraunhofer also state that they are working to reduce the encoder and decoder delays[2].

When using LAME directly in the MAC terminal, a message is displayed when decoding MP3 files which clearly states that the zero-level samples are being skipped:

```
input:  DM520_L1.mp3  (44.1 kHz, 2 channels, MPEG-1 Layer III)
output: DM520_L1.wav  (16 bit, Microsoft WAVE)
skipping initial 529 samples (encoder+decoder delay)
Frame#  1164/1164   128 kbps   MS
```

MP3 Generation 1 Files Being Decoded to WAV Generation 1

```
input:  DM520_L1.mp3  (44.1 kHz, 2 channels, MPEG-1 Layer III)
output: DM520_L1.wav  (16 bit, Microsoft WAVE)
skipping initial 1105 samples (encoder+decoder delay)
skipping final 576 samples (encoder padding-decoder delay)
Frame#  1164/1164   128 kbps  L  R
```

MP3 Generation 2 Files Being Decoded to Wav Generation 2

```
input:  DM520_L1.mp3  (44.1 kHz, 2 channels, MPEG-1 Layer III)
output: DM520_L1.wav  (16 bit, Microsoft WAVE)
skipping initial 1105 samples (encoder+decoder delay)
skipping final 576 samples (encoder padding-decoder delay)
Frame#  1164/1164   128 kbps  L  R
```

MP3 Generation 3 Files Being Decoded to WAV Generation 3

```
input:  DM520_L1.mp3  (44.1 kHz, 2 channels, MPEG-1 Layer III)
output: DM520_L1.wav  (16 bit, Microsoft WAVE)
skipping initial 1105 samples (encoder+decoder delay)
skipping final 576 samples (encoder padding-decoder delay)
```

MP3 generation 4 Files Being Decoded to WAV Generation 4

Note the phrases “Skipping initial 529 samples (encoder+decoder delay)” and “ Skipping final 567 samples (encoder padding-decoder delay)”. According to Mark Taylor’s LAME FAQ, available at http://lame.sourceforge.net/tech-FAQ.txt, most MP3 codecs have a roughly 528 sample decoder delay and 528 sample encoder delay. At the time of the writing of that FAQ, LAME was working to reduce this delay further. As stated by the current version (3.99.5) much of that delay is now ignored when decoding, meaning the resulted .wav files will have no additional zero-level samples added by the codec. This does not, however, mean that files decoded with LAME will have no zeros. The data collected shows that the files will have a very similar number of zeros as the original file did. In other words, regardless of how many generations are created with LAME 3.99.5, the file will always have a similar number of zeros. The four generations created in this stuffy showed a slightly decreasing amount of zeros in each generation on average, but most individual files showed no decrease.

4.2 Inter-Program Variance

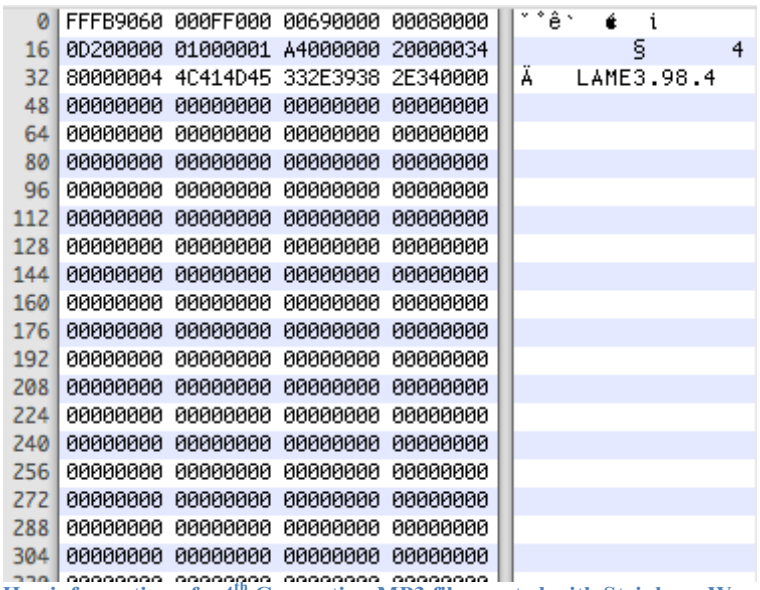
While one would expect multiple programs that use the same MP3 library to produce identical results, this was found to not always be the case. Take, for example, SoX and ffmpeg. Both programs were executed in the Mac terminal and use LAME 3.99.5. The two programs produced different results. It is clear that more than just the MP3 codec used factors into the amount of zero-level sample padding added.

Chapter V

FUTURE RESEARCH / FRAMEWORK

Any future research on this topic would be useful in developing a more comprehensive database with which to compare evidence files. The most popular codecs used today are LAME and Fraunhofer, but each has many different revisions with different behaviors. An interesting topic for further research would be ZLS variance within different versions of the same codec. This way, if we know which version of a codec was used to compress a file (sometimes displayed in the file metadata) we can more accurately estimate how many times the file was recompressed. An interesting topic of research would be to study each version of a certain codec to see how its behavior changes over time.

The current framework for audio authentication includes more than just zero-level sample analysis. One of the more commonly used methods for audio authentication is a metadata/structure analysis. This analysis includes looking at the metadata of an audio file to see if it is consistent with an authentic file. A proposed addition to this current framework is to combine these two analyses (zero-level sample and metadata analysis). By looking at the metadata of a file we are more likely able to determine the codec and version used to compress the file. For example, files encoded with LAME will typically have “LAME” and the version number in the header of the file.



0	FFFB9060 000FF000 00690000 00080000	ˇ°ê` ¢ i
16	0D200000 01000001 A4000000 20000034	§ 4
32	80000004 4C414D45 332E3938 2E340000	Ä LAME3.98.4
48	00000000 00000000 00000000 00000000	
64	00000000 00000000 00000000 00000000	
80	00000000 00000000 00000000 00000000	
96	00000000 00000000 00000000 00000000	
112	00000000 00000000 00000000 00000000	
128	00000000 00000000 00000000 00000000	
144	00000000 00000000 00000000 00000000	
160	00000000 00000000 00000000 00000000	
176	00000000 00000000 00000000 00000000	
192	00000000 00000000 00000000 00000000	
208	00000000 00000000 00000000 00000000	
224	00000000 00000000 00000000 00000000	
240	00000000 00000000 00000000 00000000	
256	00000000 00000000 00000000 00000000	
272	00000000 00000000 00000000 00000000	
288	00000000 00000000 00000000 00000000	
304	00000000 00000000 00000000 00000000	

Figure 5: Hex information of a 4th Generation MP3 file created with Steinberg Wavelab

Chapter VI

CONCLUSION

In the past examiners may have used the number of zero-level sample on an MP3 file to estimate whether or not the file has been re-compressed multiple times. According to the results of this study, many of the new versions of codecs no longer add a significant amount of zeroes to the file. As a result of this development, it is not necessarily accurate for all codecs to use the number of zero-level samples on the file to determine the number of re-compressions. It can be effective for some codecs, but not all. Depending on the codec used, we can determine that a file was likely recompressed *at least* once, but not always exactly how many times it was edited.

Take, for example, a file in question which has no zero-level sample padding at the beginning or end of the file. It is unlikely that the file is directly from the recorder, as none of the original files had no zero-level samples straight from the recorder. We cannot, however, make an accurate estimate as to how many times the file was re-compressed unless we know which codec and which version were used. Based on this sample data alone, we know that a file with no zero-level samples at the beginning could be a 4th generation compressed with LAME 3.99.5, a 3rd generation compressed with Logic Pro, or a number of other possibilities. Most of the recorders used use the Fraunhofer codec[3], but the exact version is not given. Based on the fact that most of the original files had some number of zero-level samples at the beginning of the files we can guess that they are using an older version of the codec, but we cannot be sure.

By considering the findings of this study with the already existing parts of the audio authentication framework, we can improve our methods and ensure more accurate authentication results in the future.

REFERENCES/BIBLIOGRAPHY

[1]Taylor, Mark. "LAME Technical FAQ." *LAME Technical FAQ*. LAME, June 2000. Web. 14 Oct. 2015. <http://lame.sourceforge.net/tech-FAQ.txt>.

[2]Allamanche, Eric, Ralf Gieger, Jürgen Herre, and Thoma Sporer. "MPEG-4 Low Delay Audio Coding Based on the AAC Codec." *Audio Engineering Society* (1999). Web. 14 Oct. 2015.

[3]Product manuals available on manufacturer websites

Sripada, P. (2006). *MP3 Decoder in Theory and Practice* Masters Thesis Blekinge Institute of Technology, Sweden

Raissi, R. (2002). *The Theory Behind MP3*