Kompresi Data Audio Lossless format FLAC menjadi Audio Lossly format MP3 dengan Algoritma Huffman Shift Coding

JURNAL

Diajukan untuk memenuhi persyaratan memperoleh gelar Sarjana

Program Studi Ilmu Komputasi

Universitas Telkom

Disusun Oleh :

Luthfi Firmansah

1107110084

PROGRAM STUDI ILMU KOMPUTASI

UNIVERSITAS TELKOM BANDUNG

2015

KOMPRESI DATA AUDIO *LOSSLESS* FORMAT FLAC MENJADI AUDIO *LOSSY* FORMAT MP3 DENGAN ALGORITMA *HUFFMAN SHIFT CODING*

***Luthfi Firmansah*[1]*; Erwin Budi Setiawan*[2]**
[1.2] *Prodi S1 Ilmu Komputasi, Fakultas Informatika, Universitas Telkom*
Jalan Telekomunikasi No.1, Dayeuh Kolot, Bandung 40257
e-mail: ituphie@gmail.com [1], *erwinbudisetiawan@telkomuniversity.ac.id* [2]

ABSTRAK

Saat ini, musik sudah menjadi suatu kebutuhan. Teknologi di bidang musik sendiri sudah berkembang pesat. Bisa di buktikan dengan perbaikan kualitas audio dari generasi ke generasi. Seperti halnya data gambar maupun video, data audio juga memerlukan kompresi untuk isu storage dan keperluan pengaksesan secara real time. Serta dukungan player pada umumnya yang masih sedikit support terhadap beberapa file audio lossless. Salah satu algoritma yang dapat digunakan adalah Huffman, dengan perkembangan algoritma nya yaitu Huffman Shift Coding. Dalam algoritma Huffman Shift Coding mampu mengubah setiap simbol yang dimiliki suatu data audio baik lossy maupun losless. Dari metode Huffman Shift Coding yang sudah pernah diujikan, rasio kompresi rata - rata diatas 50%. Namun hasil kompresi dari data audio tersebut tidak bisa diputar ulang, kecuali dukungan dari aplikasi kompresi itu sendiri. Dalam penelitian yang dilakukan, penulis mencoba membuat suatu aplikasi kompresi yang dapat merubah data audio yang sifatnya lossless menjadi lossy yang dimana nantinya hasil keluaran dari proses kompresi akan menjadi lebih kecil dengan tujuan untuk menghemat isu storage, dan juga tetap dapat diputar di banyak player pemutar musik yang banyak beredar saat ini.

Kata kunci : kompresi, Huffman *Coding*, audio, *lossless audio*, *lossly audio*

ABSTRACT

Nowadays, music has become a necessity. Technology in the field of music itself has been growing rapidly. It can be attested by the improvement of the quality of audio from generation to generation. As well as images and video data, audio data also require compression for storage and for the purposes of accessing the issues in real time. As well as supported players in general are still a little support to some lossless audio files. One algorithm that can be used is Huffman, with the development of its algorithm is called Huffman Shift Coding. Huffman Shift Coding able to change any symbol held on audio data either lossy or losless. Huffman Shift Coding method that has been tested, average compression ratio - 50% above average. But the results of the compression of audio data cannot be played back, but the support of compression application itself. In this research, the author tries to make a compression application that can change the nature of lossless audio data into a lossy, where later the output of the compression process will be smaller in order to save storage issues, and also still can be played in various music players that is currently available.

Keyword : compression, Huffman Coding, audio, lossless audio, lossly audio

1. Pendahuluan

Teknologi sudah berkembang sangat pesat, kebutuhan manusia untuk hiburan sudah sangat meningkat dari segi kualitas. Termasuk di bidang audio. Kualitas audio yang baik, menjadi salah satu kepuasan tersendiri bagi para penikmatnya. Banyak sekali format audio yang tersedia sampai saat ini. Salah satu format audio yang cukup populer adalah FLAC dan MP3. Dari segi kualitas, jelas FLAC lebih baik dibandingkan dengan MP3 [5]. Namun, pada kenyataannya penikmat audio digital itu sendiri masih banyak menggunakan standar audio MP3 (MPEG-1 Audio Layer 3). Dimana format ini cukup populer dari tahun 90-an hingga saat ini. Karena dianggap sudah cukup bagus kualitasnya, sehingga masih banyak penikmat *audio digital* tetap mempergunakan format MP3 ini. Jika kita bandingkan dengan format MP3, format FLAC ini memang memerlukan ruang yang cukup besar. Jika suatu data audio berkualitas CD Audio menggunakan *sampling rate* 44,1 kHz, 16 *bit per sample*, 2 kanal (stereo), maka total media penyimpanan data audio per detik adalah sekitar 176.400 *byte* sehingga untuk durasi 60 detik (1 menit) diperlukan 10,584 MB. Jika rata-rata durasi dalam satu lagu sekitar 4 menit, maka dibutuhkan tempat sekitar 40 *Megabyte* untuk menyimpan data audio tersebut. Ini tentunya sangat memboroskan media penyimpanan *hard disk [4]*.

Maka diperlukan kompresi dengan teknik dan algoritma tertentu. Untuk merubah dari *lossless* menjadi *lossy* tersebut, digunakan algoritma *Huffman* yang banyak digunakan untuk kompresi lainnya. Algoritma *Huffman* bekerja dengan cara melakukan pengkodean dalam bentuk *bit* untuk mewakili data karakter. Algoritma ini kurang maksimal jika ada banyak variasi simbol. Untuk mengoptimalkan algoritma *Huffman* untuk kompresi ini bisa digunakan algoritma *Huffman Shift Coding* yang akan membagi simbol awal menjadi beberapa blok. Sehingga pada tugas akhir ini akan diimplementasikan algoritma algoritma *Huffman Shift Coding* tersebut untuk melakukan kompresi *file audio lossless* menjadi *file audio lossly* namun tetap dengan memperhitungkan kualitas suara agar hasil kompresi nya tetap baik untuk di dengar.

1.1 Perumusan Masalah

Rumusan masalah pada Tugas Akhir ini adalah sebagai berikut :

1. Bagaimana menciptakan dan mengembangkan aplikasi menggunakan algoritma *Huffman Shift Coding* untuk melakukan kompresi dari *file audio lossless* menjadi *lossly*.
2. Bagaimana tingkat kompresi dengan menggunakan algoritma *Huffman Shift Coding* tersebut dari hasil kompresi yang di lakukan.
3. Bagaimana aplikasi yang dibuat tersebut mampu menghasilkan keluaran yang nanti nya dapat diputar atau di *playback* dengan aplikasi pihak ketiga.

1.2 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah sebagai berikut :

1. Menghasilkan aplikasi untuk kompresi dengan hasil kompresi yang baik dengan penerapan algoritma *Huffman Shift Coding.*
2. Mengetahui tingkat kompresi dari *file audio* dari hasil yang di dapatkan dengan penerapan algoritma *Huffman Shift Coding*.

1.3 Batasan Masalah

Batasan masalah yang diberikan dalam Tugas Akhir ini sebagai berikut :

1. Data audio FLAC yang digunakan untuk kompresi adalah data *lossless audio* yang memiliki banyak detil instrumen musik.
2. Data audio yang digunakan adalah data audio yang sudah terkompresi dan tidak melakukan *ripping cd* secara langsung. Data yang diambil adalah data yang di unduh dari beberapa *source* penyedia FLAC.
3. Data audio yang diujikan memiliki kemiripan atau originalitas yang cukup tinggi dengan penentuan kualitas nya diatas 70%.
4. Hasil kompresi menggunakan standar *lossly dengan bitrate* 64 kpbs karena memiliki ukuran yang cukup kecil dan masih dianggap bagus untuk suatu kualitas audio.
5. Alokasi media penyimpanan untuk perhitungan jumlah akhir dari data asli (*lossless)* menjadi data keluaran (*lossy)* sebesar 500 *megabyte* atau setara 0,5 *gigabyte*.
6. Proses kompresi dalam aplikasi masih berada dalam *form* yang berbeda dan hanya dapat dilakukan satu kali proses kompresi dengan satu data yang diujikan.
7. *Detail* selama proses kompresi tidak ditunjukan dengan fungsi *timer*, namun diakhiri dengan suatu *pop-up* atau pesan peringatan dari aplikasi tersebut yang menandakan bahwa proses kompresi sudah selesai dilakukan.
8. Penentuan nama *file* dilakukan dengan manual, tidak mengikuti nama asli dari *file* audio yang diujikan

9. Menganalisis hasil keluaran kompresi yang dilakukan untuk penghematan *storage* yang dimiliki.

2. Tinjauan Pustaka

2.1 Audio Digital

Suara yang kita dengar sehari-hari adalah merupakan gelombang *analog*. Gelombang ini berasal dari suatu tekanan udara yang ada di sekeliling kita, yang dapat kita dengar dengan bantuan gendang telinga. Layaknya gambar, suatu audio digital dapat dipecah menjadi angka-angka yang kemudian dapat diolah [5]. Digitalisasi suatu data audio dilakukan dengan mengukur suatu tegangan di banyak titik dalam suatu rentan waktu, menejermahkan setiap pengukuran ke dalam bentuk angka dan kemudian menuliskan angka-angka tersebut pada sebuah file. Proses ini disebut dengan *sampling.*

2.2 Kelebihan Audio Digital

Kelebihan *audio digital* adalah kualitas reproduksi suara yang sempurna. Kualitas reproduksi yang sempurna yang dimaksud adalah kemampuannya untuk menggandakan sinyal audio secara berulang-ulang tanpa mengalami penurunan kualitas suara.

Kelebihan lain dari *audio digital* adalah ketahanan terhadap *noise* (sinyal yang tidak diinginkan). Pada saat transmisi data dan pemrosesan dengan komponen-komponen elektrik, pada sinyal analog sangat mudah sekali terjadi gangguan-gangguan berupa *noise*. . Suara desis pada kaset rekaman merupakan salah satu contoh terjadinya *noise* berupa gangguan pada frekuensi tinggi [2].

2.3 Format FLAC

Nama FLAC adalah singkatan dari *Free Lossless Audio Compression*. FLAC dirancang secara khusus untuk kompresi audio dan juga mendukung streaming dan arsip data audio.

FLAC adalah gagasan dari Josh Coalson yang dikembangkan pada tahun 1999 yang kemudian ia memulai proyek FLAC nya di situs yang terkenal yaitu *Sourceforge* dengan merilis implementasi dari format FLAC tersebut. Sejak itu, banyak pengembang memberikan kontribusi nya untuk meningkatkan implementasi dari FLAC ini [8]. Format FLAC memanfaatkan tingginya korelasi antar *sample* pada data audio. FLAC menggunakan prediksi linear untuk mengkonversi *sample* menjadi deretan angka yang kemudian disebut residu. FLAC menghasilkan rasio kompresi sebesar 50% hingga 60%.

2.4 Pengertian Kompresi

Kompresi adalah pengubahan data kedalam bentuk yang memerlukan *bit* yang lebih sedikit, biasanya dilakukan agar data dapat disimpan atau dikirimkan dengan lebih efisien. Kebalikan dan proses kompresi, yaitu dekompresi. Dekompresi sendiri merupakan proses untuk mengembalikan data baru yang telah dihasilkan oleh proses kompresi menjadi data awal. Data yang telah dilakukan kompresi dapat digunakan jaringan yang lebih rendah yang sesuai dengan kapasitas data yang telah dilakukan proses kompresi [5].

Teknik kompresi terdiri dari 3 kategori, antara lain : sumber, *entropy*, dan *hybrid*. Untuk kompresi kategori sumber yaitu jenis audio *lossly*, terjadi beberapa bagian kompenen dari data yang hilang akibat dari proses kompresi. Untuk kategori *entropy* adalah jenis audio *lossless*, yang berarti tidak ada data yang hilang selama proses kompresi berjalan. Sedangkan untuk kategori *hybrid*, merupakan kombinasi dari jenis audio *lossy* dan *lossless.*

2.5 Algoritma *Huffman Shift Coding*

Pada *Huffman Shift Coding*, simbol dibagi menjadi beberapa blok dengan ukuran yang sama. Biasanya ukuran blok tersebut adalah $2^k - 1$ simbol, dimana k adalah bilangan bulat positif. Jika k=1, maka *Huffman Shift Coding* sama dengan *Standart Huffman Shift Coding*.

Simbol dari blok pertama akan dikodekan persis menggunakan *Huffman Coding* standar. Ketika mengkodekan simbol dari blok pertama, ikut pula dikodekan simbol hipotesis yang frekuensi kemunculannya sama dengan jumlah frekuensi kemunculan simbol-simbol dari blok yang lainnya. Yang membedakan antara satu blok dengan blok selanjutnya adalah penambahan *prefix* yang kemudian hasil pengkodean simbol hipotesis tersebut digunakan untuk menandai setiap blok.

Dengan algoritma *Huffman Shift Coding* ini, bisa meningkatkan penggunaan waktu yang lebih sedikit dan rata - rata panjang kode yang lebih efisien. Secara garis besar, berikut algoritma kompresi *Huffman Shift Coding* ini bekerja :

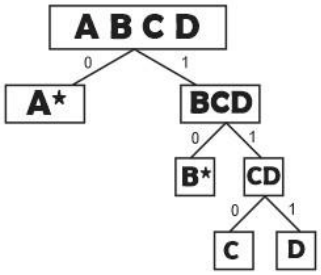
1. *Source Symbol* disusun sehingga kemungkinan yang muncul dari yang terbesar sampai yang terkecil.
2. Jumlah total dari *source symbol* dibagi menjadi beberapa blok simbol dengan ukuran yang sama.
   Simbol dari blok pertama akan dikodekan menggunakan *Huffman Coding Standart.* Ketika mengkodekan simbol dari blok pertama, ikut pula dikodekan simbol

hipotesis yang frekuensi kemunculannya sama dengan jumlah frekuensi kemunculan simbol-simbol dari blok yang lainnya. Yang membedakan antara satu blok dengan blok selanjutnya adalah penambahan satu atau lebih kode *prefix* hasil dari pengkodean simbol hipotesis untuk menandai tiap blok.

3. Simbol hipotesis dari *Huffman Coding* kita anggap sebagai C.
4. Kode simbol ke-I dari blok k adalah (𝑘−1 digabungkan dengan simbol ke-I dari *Huffman Coding* blok pertama.

Terdapat perbedaan dalam pembentukan *tree* dari Huffman Shift Coding ini. Seperti yang ditunjukan dalam gambar berikut.

Nilai :
A = 0,4
B = 0,3
C = 0,2
D = 0,1



2.6 *FFMPEG*

Agar hasil keluaran dari proses kompresi dapat diputar, dibutuhkan *plugin* khusus. Salah satunya adalah penggunaan *FFMPEG. FFMPEG* adalah program komputer yang dapat merekam, mengkonversikan dan streaming audio dan video digital dalam berbagai format. *FFMPEG* merupakan aplikasi *command line* yang terdiri dari kumpulan pustaka perangkat lunak bebas. Proyek ini dimulai oleh Fabrice Bellard, dan saat ini dikelola oleh Michael Niedermayer. [7]

Dalam pembuatan aplikasi ini, *FFMPEG* sendiri membantu dalam proses kompresi dimana nanti nya hasil keluaran dari proses kompresi tersebut masih dapat diputar. Setiap *byte* serta *metadata* juga dirubah oleh bantuan *FFMPEG* ini. *FFMPEG* juga mendukung *stream* dari *layer 3* dimana ini seperti *layer* standar yang dimiliki oleh MP3.

3. Perancangan Sistem

3.1 Deskripsi Perancangan Sistem

Pada tugas akhir ini, akan dirancang sebuah aplikasi yang mampu melakukan kompresi data atau juga dikenal sebagai pemadatan data audio yang mempunyai tujuan memperkecil ukuran data sehingga selain dapat menghemat media penyimpanan dan memudahkan *transfer* dalam jaringan seperti Internet misalnya. Untuk proses kompresi nya sendiri, khusus hanya pada *file* audio berjenis FLAC (*Free Lossless Audio Codec*) dan keluaran yang ditentukan adalah MP3 yang hanya mendukung jumlah kanal maksimum 2 buah kanal (*mono* dan *stereo*).

Pada umumnya, suatu *file* audio merupakan data digital yang berupa representasi atas *bit* ‘0’ dan ‘1’. Seringkali dalam sebuah *file* terjadi perulangan data atau *redundancy*. Semua metode kompresi melakukan pemadatan terhadap data berulang tersebut.

Seperti diketahui, jenis algoritma kompresi terbagi atas *lossless compression* dan *lossy compression.* Pada *lossly compression* ada data yang hilang tetapi tidak banyak setelah data dikompresi. Data hasil kompresi *lossy compression* jika dikembalikan maka hasilnya tidak akan sama persis dengan data orisinil. Berbeda dengan *lossy compression,* pada *lossless compression* tidak ada data yang hilang setelah proses kompresi dan data dapat dikembalikan seperti data semula.

3.2 Algoritma Kompresi

Algoritma atau *encoding* Huffman sebenarnya merupakan algoritma kompresi yang dapat diterapkan pada semua jenis baik untuk *file* biner maupun *file* teks. Algoritma ini efektif dengan rasio kompresi yang rendah jika terdapat banyak *redundancy data* atau perulangan data yang sama pada *file* audio yang digunakan.

File FLAC tersebut biasanya selalu berukuran besar untuk durasi waktu main yang lama. Dari data audio yang sudah dikumpulkan dengan jenis *sample rate* 48.000 kHz dengan jumlah kanal 2 atau *stereo* dan bits *per sample* nya 24 *bit* untuk durasi selama 1 detik saja memerlukan kapasitas sebesar 48.000 x 2 x 24 = 2.304.000 *bit* per detik = 288.000 *byte* per detik. Bila kita ambil rata - rata durasi dari data audio yang memiliki durasi sekitar 4 menit, maka media penyimpanan yang diperlukan adalah 288.000 x 4 x 60 = 69.120.000 byte atau setara dengan 69 *Megabyte.* Apabila kita memiliki media penyimpanan sebesar 500 *megabyte,* maka dapat dihitung *disk space* setelah disimpan satu *data* audio *format* FLAC adalah = 500 - 69 = 431 *megabyte.* Maka jika kita memiliki media penyimpanan yang ingin di *transfer data* audio FLAC, maka hanya dapat menampung kurang lebih 7 *file* saja.

4. Algoritma dan Implementasi

4.1 Implementasi Sistem

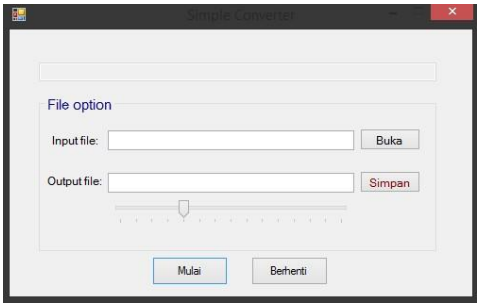
Dalam pembuatan sistem ini, digunakan aplikasi *Visual Studio 2013* yang berjalan di sistem operasi *Windows*. *Visual Studio* sendiri hingga sekarang dikembangkan oleh *Microsoft*. Untuk menjalankan program tersebut, tentu membutuhkan media untuk menjalankan nya. Baik secara perangkat keras (*hardware*) maupun perangkat lunak (*software*) nya.

4.2 Cara Penggunaan Program

Untuk menjalankan program tersebut, dapat dilakukan melalui aplikasi *Visual Studio 2013* yang sudah terinstall di komputer atau pada program *AudioConverter*.exe yang terletak di tempat tujuan kita menyimpan hasil akhir dari *source code* yang dibuat. Berikut tampilan dari antarmuka aplikasi tersebut.

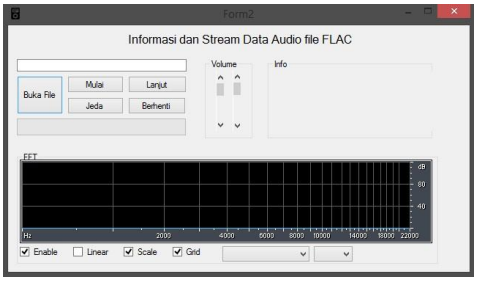
Gambar 4.1 Tampilan Utama

Untuk memulai proses kompresi, *user* dapat menekan tombol  "Mulai Convert" untuk mulai melakukan proses kompresi setiap *file* FLAC satu persatu. Akan ada *form* khusus yang menampilkan tentang aplikasi untuk kompresi. Agar hasil kompresi yang kita proses dapat diputar kembali tanpa harus di *decompress*, maka dalam aplikasi ini didukung dengan *plugin FFMPEG* yang tersimpan dalam *folder "bin"* di project yang kita buat. Bentuk *plugin* tersebut sudah menjadi *execute* yang dengan mudah kita lakukan eksekusi dalam program. *Plugin* tersebut dapat dengan mudah kita dapatkan dari situs resmi *FFMPEG*. Berikut tampilan dari *form* untuk melakukan kompresi.



Gambar 4.2 *Form* untuk melakukan kompresi

Dalam aplikasi yang dibuat ini, terdapat juga fitur untuk menganalisa suatu *file* FLAC. Dimana nantinya, *file* tersebut akan menampilkan beberapa informasi serta dapat diputar untuk kemudian di analisa. Fitur ini dibuat di *form* berbeda dengan menu utama, dan dapat dijalankan apabila kita men-klik  "Analisa File". Saat *form* tersebut muncul, menu utama tidak dapat digunakan dan akan terlihat seperti gambar berikut ini.



Gambar 4.3 Menu Form kedua untuk analisa *file audio*

4.3 Pengujian Sistem

Untuk mengetahui hasil pengujian program kompresi *file audio* ini dengan Algoritma Huffman Shift coding tersebut yang telah di implementasikan, dengan itu dilakukan pengujian untuk beberapa file audio format FLAC. FLAC yang diuji cobakan mempunyai ukuran yang bervariasi dan sudah dilakukan pengecekan terhadap file FLAC tersebut dengan tingkat originalitas yang tinggi. Diharapkan bahwa *output*-an yang dihasilkan tetap dapat diputar dengan baik. Pengecekan tersebut dilakukan dengan aplikasi *Spectro* yang memiliki tampilan *spectrogram* serta mampu membaca seberapa besar original suatu data audio dengan hasil rekaman asli nya.

Tabel 4.1 Identitas Data Audio *format* FLAC yang diujikan

No	Nama File Audio	Originality (%)	Ukuran (*megabyte*)
1	Critical Acclaim.flac	72	38,22
2	Almost Easy.flac	73	28,99
3	Scream.flac	74	36,00
4	Dear God.flac	71	46,99
5	Shake It Off - Taylor Swift.flac	73	27,02
6	Jealous - Nick Jonas.flac	73	27,4
7	Centuries - Fall Out Boy.flac	76	29,53
8	Animals - Maroon 5.flac	72	28,03
9	Time Of Our Lives – Pitbull.flac	72	27,89
10	Ghost - Ella Henderson.flac	73	26,68
11	Elastic Heart – Sia.flac	71	31,08
12	Heroes (We Could Be) – Alesso.flac	79	28,15
13	I Lived – OneRepublic.flac	71	28,24
14	Prayer in C - Lilly Wood.flac	71	22,77
15	She Knows - Ne-Yo.flac	76	28,34

Tabel 4.2 Identitas dari hasil kompresi yang dilakukan menjadi *format* MP3

No	Nama File Audio	Ukuran Hasil Ouput (*megabyte)*	Lama Kompresi (detik)
1	Critical Acclaim.mp3	2,39	12
2	Almost Easy.mp3	1,77	9
3	Scream.mp3	2,19	11
4	Dear God.mp3	2,99	15
5	Shake It Off - Taylor Swift.mp3	1,66	8
6	Jealous - Nick Jonas.mp3	1,68	9
7	Centuries - Fall Out Boy.mp3	1,73	9
8	Animals - Maroon 5.mp3	1,74	9
9	Time Of Our Lives - Pitbull.mp3	1,74	10
10	Ghost - Ella Henderson.mp3	1,94	8
11	Elastic Heart - Sia.mp3	1,96	11
12	Heroes (We Could Be) - Alesso.mp3	1,59	8
13	I Lived - OneRepublic.mp3	1,78	9
14	Prayer in C - Lilly Wood.mp3	1,43	7
15	She Knows - Ne-Yo.mp3	1,67	9

Tabel 4.3 Perbandingan Proses Kompresi berulang kali

File Audio format FLAC	Ukuran Setelah Pengujian (byte)		
	Percobaan 1	Percobaan 2	Percobaan 3
Critical Acclaim.flac	2,511,339	2,511,339	2,511,33
Animals - Maroon 5.flac	1,827,558	1,827,558	1,827,55
Elastic Heart - Sia.flac	2,060,152	2,060,152	2,060,15

5. Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan penelitian yang dilakukan sebelumnya maka dapat diambil kesimpulan sebagai berikut :

1. Kecepatan proses tidak hanya bergantung dari jumlah *file* audio yang di proses, tetapi juga bergantung dari ukuran dari *file* tersebut dan juga dari seberapa banyak proses yang dijalankan bersamaan saat aplikasi ini dijalankan
2. Proses kompresi dari aplikasi tersebut bersifat statis. Karena walaupun digunakan dan diujikan dengan *file* audio yang sama

berulang kali, tetap memiliki hasil keluar dengan ukuran yang sama.

3. Hasil *output* dari aplikasi ini, dapat diputar di kembali dengan fitur yang terdapat dalam aplikasi ini bahkan aplikasi pihak ketiga sekalipun karena memiliki standar *stream* layaknya *format* audio MP3 pada umumnya.
4. *Range* perubahan dari *file* audio yang diujikan, memiliki persentase perubahan sebesar 93,79% dan hasil keluaran nya menjadi 6,21% dari ukuran awal *file* audio FLAC yang diujikan.

5.2 Saran

Untuk pengembangan dari aplikasi ini agar lebih baik, maka diberikan beberapa saran sebagai berikut :

1. Diharapkan proses kompresi dapat dilakukan di menu utama dan dapat dilakukan dengan jumlah data yang banyak tanpa harus *input* satu per satu.
2. Kualitas audio yang diciptakan dari hasil kompresi dapat memiliki kualitas yang lebih baik namun tetap dengan penghematan media penyimpanan.
3. Diharapkan hasil dari proses kompresi hanya menghasilkan satu *file* saja agar lebih memudahkan untuk pengguna dalam pencarian data yang sudah terkompresi.
4. Apabila diperlukan, fitur - fitur tambahan yang dilakukan di dalam penelitian ini dapat diterapkan dalam satu aplikasi kompresi yang diciptakan.

DAFTAR PUSTAKA

[1] Adhitama, Gagarin. (2009). *Perbandingan Algoritma Huffman dengan Algoritma Shannon-Fano*. Institut Teknologi Bandung, Bandung.

[2] Angga, *Teknik Kompresi Lossless dan Lossy pada Audio*, 23 Januari 2013 [Online]. Available : http://bangkitagp.wordpress.com/2013/01/23/teknik-kompresi-lossless-dan-lossy-pada-audio/ [Diakses 18 Oktober 2014]

[3] Bagus, Galang Prasetyo (2013). *Kompresi File Audio Wave Menggunakan Algortima Huffman Shift Coding*. Universitas Brawijaya, Malang.

[4] Benjamin, A. (2010). *Music Compression Algorithms and Why You Should Care*. Alexander Benjamin.

[5] Daryanto, T. (2005). *Sistem Multimedia dan Aplikasinya*. Yogyakarta: Graha Ilmu.

[6] Hacker, S. *Inside the MP3 Codec* [Online]. Available : http://www.mp3-converter.com/mp3codec/huffman_coding.htm [Diakses 22 Oktober 2014]

[7] Rongshan Yu, X. L. (2004). *A Scalable Lossy to Lossless Audio Coder For MPEG-4 Lossless Audio Coding*. National University of Singapore, Singapore.

[8] Solomon, D. (2012). *Data Compression The Complete Reference : Fourth Edition*. Northridge, California: David Solomon.

[9] Wilson, R. (2007). *Rancang Bangun Perangkat Lunak Komposer Musik Menggunakan MATLAB*. Fakultas Teknik Universitas Indonesia, Depok.

[10] Xiph.Org Foundation. "FLAC Format". [Online]. Available : https://xiph.org/flac/ [Diakses 2 Oktober 2014]

[11] Yenny. (2004). *Kompresi File Wave Dengan Algoritma Huffman*. STMIK Mikroskil, Medan.