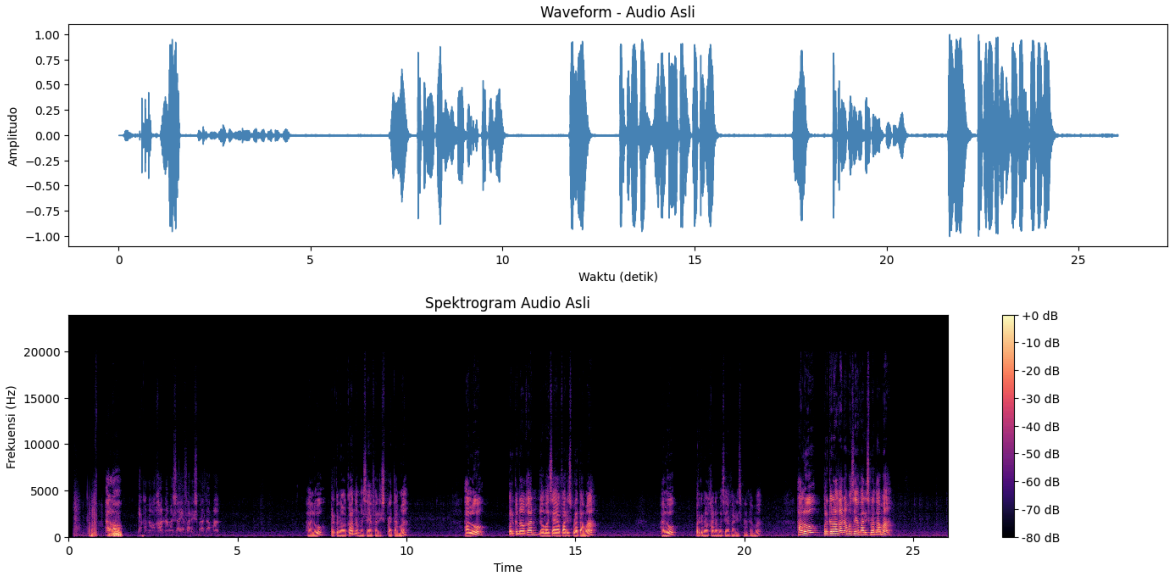
Tugas Sistem dan Multimmdia IF4021 - Multimedia Information Processing Nama : Faris Pratama Nim : 122140021

Tugas 2.1 Rekaman dan Analisis Suara Multi-Level

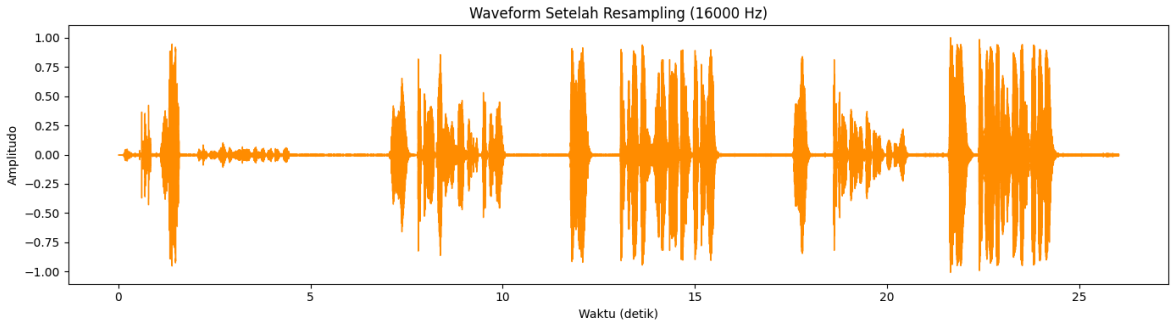
In [ ]:
```
#122140021

import os
import numpy as np
import librosa
import librosa.display
import soundfile as sf
import matplotlib.pyplot as plt

# ============================================================
# ====================== LOAD AUDIO ==========================
# ============================================================
file_audio = "audiofaris25.wav"  # Ganti dengan nama file kamu

if not os.path.exists(file_audio):
    print(f"❌ File '{file_audio}' tidak ditemukan di direktori ini.")
    raise SystemExit

# Membaca audio beserta sampling rate-nya
audio_data, sample_rate = librosa.load(file_audio, sr=None)

print(f"✅ Audio berhasil dimuat: {file_audio}")
print(f"Sample Rate Asli : {sample_rate} Hz")
print(f"Durasi Audio     : {len(audio_data)/sample_rate:.2f} detik")

# ============================================================
# ============ TAMPILKAN WAVEFORM & SPEKTROGRAM ==============
# ============================================================
plt.figure(figsize=(14, 10))

# --- Waveform ---
plt.subplot(3, 1, 1)
librosa.display.waveshow(audio_data, sr=sample_rate, color='steelblue')
plt.title("Waveform - Audio Asli")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")

# --- Spektrogram ---
spektrogram = librosa.stft(audio_data)
spektrogram_db = librosa.amplitude_to_db(np.abs(spektrogram), ref=np.max)

plt.subplot(3, 1, 2)
librosa.display.specshow(spektrogram_db, sr=sample_rate, x_axis='time', y_axis='
plt.colorbar(format="%+2.0f dB")
plt.title("Spektrogram Audio Asli")
plt.ylabel("Frekuensi (Hz)")

plt.tight_layout()
plt.show()

# ============================================================
# ================  RESAMPLING KE 16 kHz ====================
```

```python
# ============================================================
sr_target = 16000

# Konversi sampling rate
audio_resample = librosa.resample(audio_data, orig_sr=sample_rate, target_sr=sr_

# Simpan hasil resampling
output_file = "resampled_audio.wav"
sf.write(output_file, audio_resample, sr_target)

print(f"\n💾 Audio hasil resampling disimpan: '{output_file}'")
print(f"Sample Rate Baru : {sr_target} Hz")

# Visualisasi hasil resampling
plt.figure(figsize=(14, 4))
librosa.display.waveshow(audio_resample, sr=sr_target, color='darkorange')
plt.title(f"Waveform Setelah Resampling ({sr_target} Hz)")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")
plt.tight_layout()
plt.show()

# ============================================================
# ================== PERBANDINGAN DASAR ======================
# ============================================================
durasi_lama = len(audio_data) / sample_rate
durasi_baru = len(audio_resample) / sr_target

print(f"\n📊 Perbandingan Durasi:")
print(f"- Durasi Sebelum Resampling : {durasi_lama:.2f} detik")
print(f"- Durasi Sesudah Resampling : {durasi_baru:.2f} detik")
```

```
✅ Audio berhasil dimuat: audiofaris25.wav
Sample Rate Asli : 48000 Hz
Durasi Audio     : 26.02 detik
```




```
💾 Audio hasil resampling disimpan: 'resampled_audio.wav'
Sample Rate Baru : 16000 Hz
```




```
📊 Perbandingan Durasi:
- Durasi Sebelum Resampling : 26.02 detik
- Durasi Sesudah Resampling : 26.02 detik
```

- Analisis Waveform – Audio Asli *

Tampak fluktuasi amplitudo yang bervariasi sepanjang durasi sekitar 26 detik. Bagian dengan amplitudo rendah menunjukkan segmen hening atau jeda antar frasa, sementara bagian dengan amplitudo tinggi menandakan puncak energi suara, kemungkinan pada bagian refrain atau kalimat utama. Rekaman tampak bersih tanpa clipping (tidak melebihi batas ±1).

- Analisis Spektrogram – Audio Asli * Distribusi frekuensi menunjukkan energi kuat di bawah 5000 Hz, terutama pada rentang 300–3000 Hz, yang merupakan area dominan bagi frekuensi vokal manusia. Warna cerah (merah–kuning) menandakan intensitas tinggi di bagian-bagian vokal utama, sedangkan warna gelap (ungu–biru) menggambarkan area frekuensi rendah atau bagian hening. Hal ini menunjukkan bahwa rekaman memiliki keseimbangan antara energi rendah dan tinggi, khas rekaman vokal dengan sedikit noise di latar belakang.

- Analisis Setelah Resampling (16.000 Hz) *

Waveform – Setelah Resampling: Bentuk gelombang tetap identik dengan versi asli, menandakan bahwa struktur amplitudo dan durasi tidak berubah secara signifikan. Resampling hanya menurunkan jumlah sampel per detik (sampling rate) dari nilai asli ke 16 kHz, tanpa mengubah panjang waktu (durasi audio tetap ±26 detik).

Tugas 2.2 Noise Reduction dengan Filtering

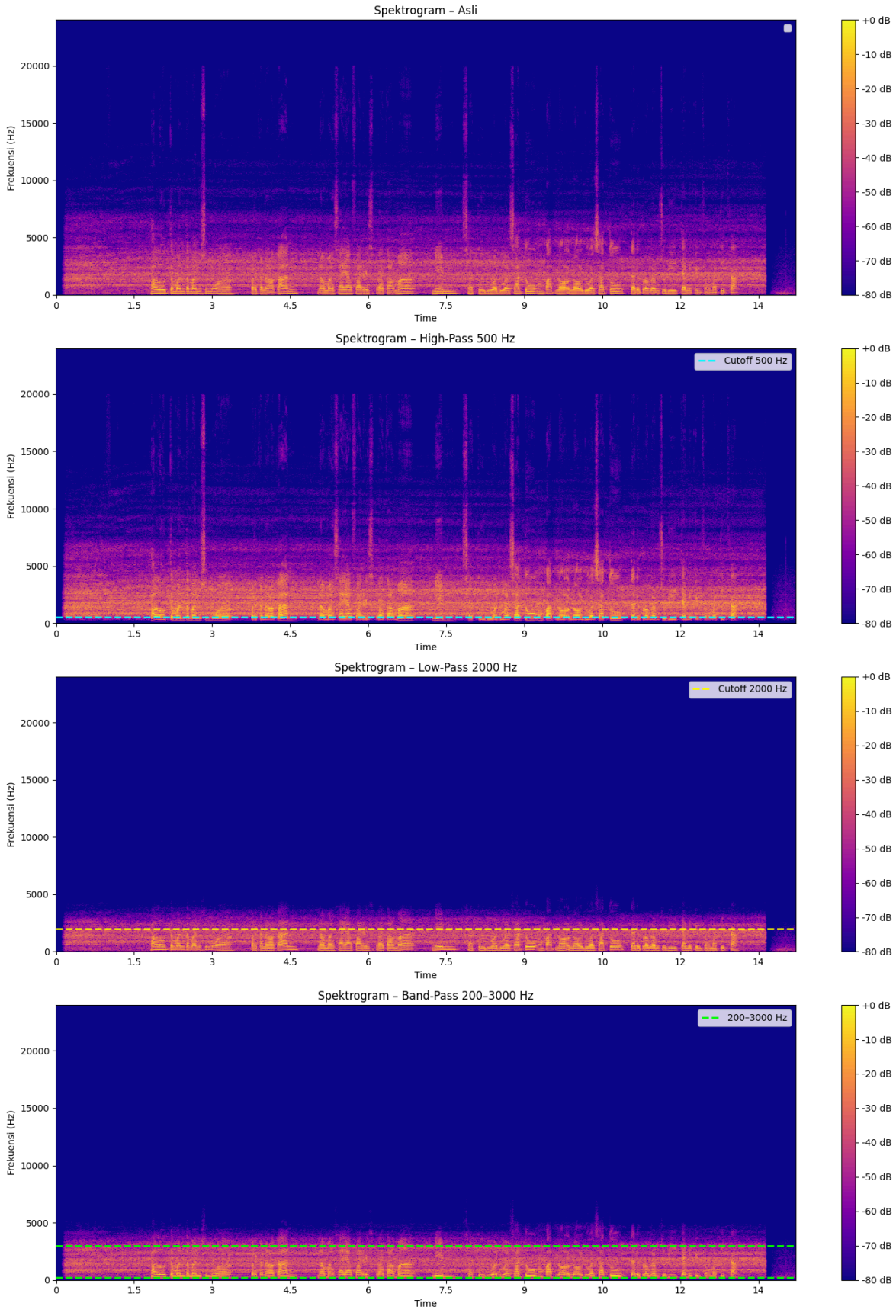
In [12]:
```python
# 122140021

import numpy as np
import librosa
import librosa.display
import matplotlib.pyplot as plt
from scipy.signal import butter, lfilter
import os

# ============================================================
# ===================== LOAD FILE AUDIO ======================
# ============================================================
nama_file = "audiofaris10.wav"  # ubah sesuai file kamu

if not os.path.exists(nama_file):
```

```
    print(f"❌ File '{nama_file}' tidak ditemukan. Letakkan di folder yang sama
    raise SystemExit

# Muat sinyal audio beserta sampling rate-nya
sinyal, fs = librosa.load(nama_file, sr=None)
durasi = len(sinyal) / fs

print(f"✅ Berhasil memuat file: {nama_file}")
print(f"Sampling rate  : {fs} Hz")
print(f"Durasi rekaman : {durasi:.2f} detik")

# ============================================================
# ============ DEFINISI FUNGSI FILTER BUTTERWORTH ============
# ============================================================
def buat_filter(data, batas, fs, jenis="low", order=5):
    """
    Membuat filter digital Butterworth (low/high/band-pass).
    - data  : array audio
    - batas : frekuensi cutoff (angka tunggal / list [low, high])
    - fs    : sampling rate
    - jenis : 'lowpass', 'highpass', 'bandpass'
    - order : orde filter (default 5)
    """
    nyquist = 0.5 * fs

    if jenis == "bandpass":
        if not isinstance(batas, (list, tuple)) or len(batas) != 2:
            raise ValueError("Gunakan list [low_cut, high_cut] untuk bandpass fi
        norm_cutoff = [b / nyquist for b in batas]
    else:
        norm_cutoff = batas / nyquist

    b, a = butter(order, norm_cutoff, btype=jenis, analog=False)
    hasil = lfilter(b, a, data)
    return hasil

# ============================================================
# ================== PROSES FILTERING ========================
# ============================================================
# Nilai cutoff tiap filter
HP_CUTOFF = 500           # high-pass → hilangkan frekuensi rendah
LP_CUTOFF = 2000          # low-pass  → hilangkan frekuensi tinggi
BP_CUTOFF = [200, 3000]   # band-pass → area suara manusia

# Terapkan semua filter
audio_terfilter = {
    "Asli": sinyal,
    f"High-Pass {HP_CUTOFF} Hz": buat_filter(sinyal, HP_CUTOFF, fs, "highpass"),
    f"Low-Pass {LP_CUTOFF} Hz": buat_filter(sinyal, LP_CUTOFF, fs, "lowpass"),
    f"Band-Pass {BP_CUTOFF[0]}-{BP_CUTOFF[1]} Hz": buat_filter(sinyal, BP_CUTOFF
}

# ============================================================
# ================ TAMPILKAN SPEKTROGRAM =====================
# ============================================================
plt.figure(figsize=(15, 5 * len(audio_terfilter)))

for i, (judul, data) in enumerate(audio_terfilter.items(), 1):
    # Ubah ke domain frekuensi
    stft_data = librosa.stft(data)
```

```
    stft_db = librosa.amplitude_to_db(np.abs(stft_data), ref=np.max)

    # Plot
    plt.subplot(len(audio_terfilter), 1, i)
    librosa.display.specshow(stft_db, sr=fs, x_axis="time", y_axis="hz", cmap="p
    plt.colorbar(format="%+2.0f dB")
    plt.title(f"Spektrogram – {judul}")
    plt.ylabel("Frekuensi (Hz)")

    # Tambahkan garis cutoff
    if "High-Pass" in judul:
        plt.axhline(y=HP_CUTOFF, color="cyan", linestyle="--", linewidth=2, labe
    elif "Low-Pass" in judul:
        plt.axhline(y=LP_CUTOFF, color="yellow", linestyle="--", linewidth=2, la
    elif "Band-Pass" in judul:
        plt.axhline(y=BP_CUTOFF[0], color="lime", linestyle="--", linewidth=2)
        plt.axhline(y=BP_CUTOFF[1], color="lime", linestyle="--", linewidth=2, l

    plt.legend(loc="upper right")

plt.tight_layout()
plt.show()

print("\n✅ Semua filter berhasil diterapkan dan divisualisasikan.")
```

```
✅ Berhasil memuat file: audiofaris10.wav
Sampling rate  : 48000 Hz
Durasi rekaman : 14.21 detik
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_10964\640807968.py:94: UserWarning: No
artists with labels found to put in legend.  Note that artists whose label start
with an underscore are ignored when legend() is called with no argument.
  plt.legend(loc="upper right")
```



✅ `Semua filter berhasil diterapkan dan divisualisasikan.`

- Penjelasan

Pada rekaman tersebut terdengar noise berupa dengungan frekuensi rendah (hum) yang muncul di bawah 500 Hz, kemungkinan berasal dari kipas, getaran, atau sumber listrik. Berdasarkan hasil spektrogram, band-pass filter dengan rentang 200–3000 Hz

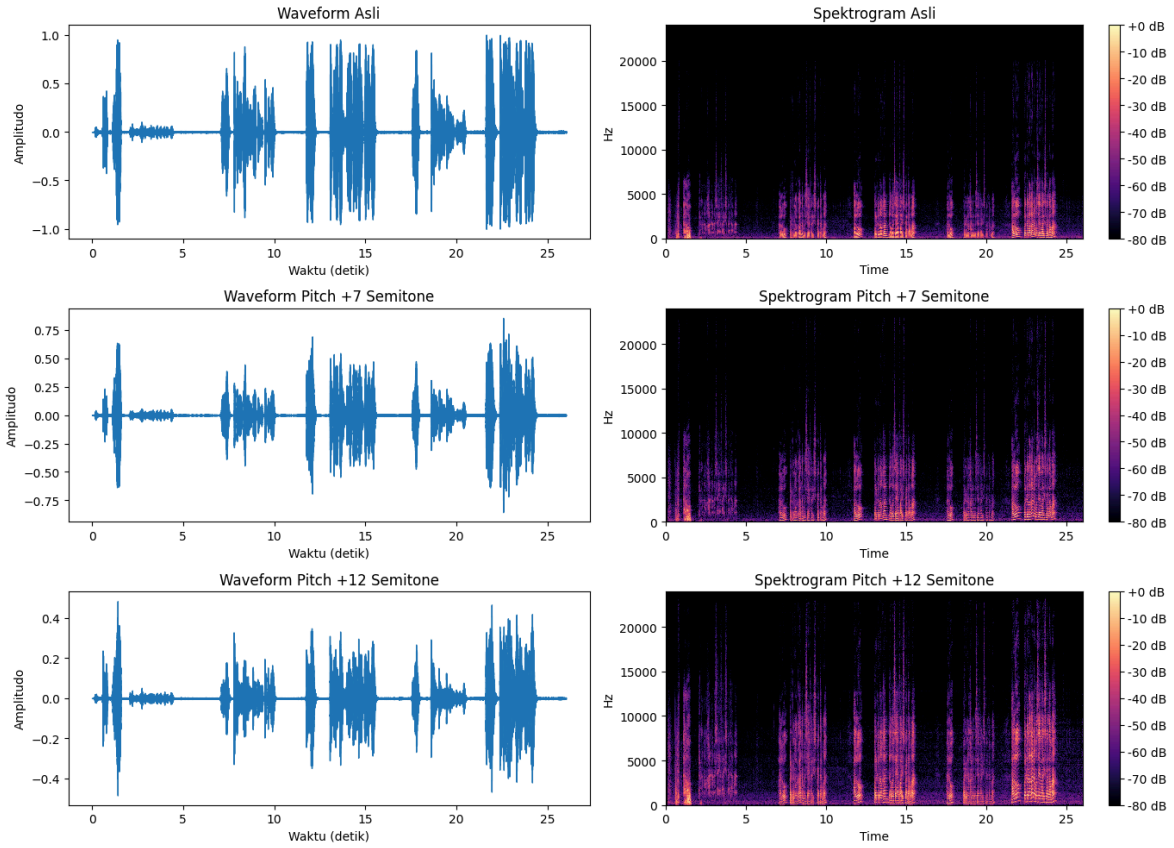
merupakan filter yang paling efektif untuk mengurangi noise tersebut. Filter ini mampu menghilangkan suara rendah yang tidak diinginkan sekaligus menekan noise frekuensi tinggi, tanpa mengganggu rentang utama frekuensi suara manusia. Nilai cutoff terbaik berada pada 200 Hz hingga 3000 Hz, karena dapat mengurangi gangguan tanpa membuat suara menjadi mendem. Setelah proses filtering dilakukan, noise latar belakang berkurang secara signifikan, dan suara terdengar lebih jernih serta ucapan menjadi lebih jelas, meskipun sedikit kehilangan komponen frekuensi tinggi.

Tugas 2.3 Pitch Shifting dan Audio Manipulation

In [15]:
```python
# 122140021

import librosa
import librosa.display
import matplotlib.pyplot as plt
import numpy as np
import soundfile as sf

# ============================================================
# ================ LOAD AUDIO FILE ASLI ======================
# ============================================================
file_audio = 'audiofaris25.wav'

try:
    data, sample_rate = librosa.load(file_audio, sr=None)
    print(f"✅ Berhasil memuat '{file_audio}' (Sample Rate: {sample_rate} Hz)")
except FileNotFoundError:
    print(f"❌ File '{file_audio}' tidak ditemukan!")
    raise SystemExit

# ============================================================
# ========= PROSES PITCH SHIFT (Efek Chipmunk) =============
# ============================================================
# List nada naik (dalam satuan semitone)
shift_values = [7, 12]
hasil_shift = {}

for n in shift_values:
    shifted = librosa.effects.pitch_shift(data, sr=sample_rate, n_steps=n)
    hasil_shift[f"Naik {n}"] = shifted
    print(f"🎶 Proses pitch shift +{n} semitone selesai.")

# ============================================================
# =========== VISUALISASI WAVEFORM & SPEKTROGRAM ===========
# ============================================================
plt.figure(figsize=(14, 10))

# --- (a) Waveform Asli ---
plt.subplot(3, 2, 1)
librosa.display.waveshow(data, sr=sample_rate)
plt.title("Waveform Asli")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")

# --- (b) Spektrogram Asli ---
plt.subplot(3, 2, 2)
spec_asli = librosa.amplitude_to_db(np.abs(librosa.stft(data)), ref=np.max)
```

```
librosa.display.specshow(spec_asli, sr=sample_rate, x_axis="time", y_axis="hz",
plt.title("Spektrogram Asli")
plt.colorbar(format="%+2.0f dB")

# --- (c) Waveform Pitch +7 ---
plt.subplot(3, 2, 3)
librosa.display.waveshow(hasil_shift["Naik 7"], sr=sample_rate)
plt.title("Waveform Pitch +7 Semitone")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")

# --- (d) Spektrogram Pitch +7 ---
plt.subplot(3, 2, 4)
spec_7 = librosa.amplitude_to_db(np.abs(librosa.stft(hasil_shift["Naik 7"])), re
librosa.display.specshow(spec_7, sr=sample_rate, x_axis="time", y_axis="hz", cma
plt.title("Spektrogram Pitch +7 Semitone")
plt.colorbar(format="%+2.0f dB")

# --- (e) Waveform Pitch +12 ---
plt.subplot(3, 2, 5)
librosa.display.waveshow(hasil_shift["Naik 12"], sr=sample_rate)
plt.title("Waveform Pitch +12 Semitone")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")

# --- (f) Spektrogram Pitch +12 ---
plt.subplot(3, 2, 6)
spec_12 = librosa.amplitude_to_db(np.abs(librosa.stft(hasil_shift["Naik 12"])),
librosa.display.specshow(spec_12, sr=sample_rate, x_axis="time", y_axis="hz", cm
plt.title("Spektrogram Pitch +12 Semitone")
plt.colorbar(format="%+2.0f dB")

plt.tight_layout()
plt.show()

# ============================================================
# =========== GABUNGKAN HASIL PITCH +7 DAN +12 =============
# ============================================================
gabungan_chipmunk = np.hstack([hasil_shift["Naik 7"], hasil_shift["Naik 12"]])

nama_output = "hasil_chipmunk_gabungan.wav"
sf.write(nama_output, gabungan_chipmunk, sample_rate)
print(f"\n💾 File gabungan berhasil disimpan sebagai '{nama_output}'")
```

```
✅ Berhasil memuat 'audiofaris25.wav' (Sample Rate: 48000 Hz)
🎶 Proses pitch shift +7 semitone selesai.
🎶 Proses pitch shift +12 semitone selesai.
```



💾 `File gabungan berhasil disimpan sebagai 'hasil_chipmunk_gabungan.wav'`

🔍 PENJELASAN PROSES PITCH SHIFTING

Parameter yang digunakan: • n_steps = +7 dan +12 digunakan untuk menaikkan nada masing-masing sebesar 7 dan 12 semitone. Nilai +12 berarti satu oktaf lebih tinggi dari suara asli, menghasilkan efek khas suara chipmunk. • sr atau sampling rate tetap menggunakan nilai asli dari file agar durasi suara tidak berubah.

Perbedaan dalam representasi visual: • Waveform: bentuk gelombang masih mirip dengan versi asli, tetapi amplitudo sedikit berubah akibat peningkatan frekuensi dan perubahan fase. • Spektrogram: terdapat pergeseran energi ke frekuensi yang lebih tinggi, terlihat dari warna cerah (merah–kuning) yang naik ke bagian atas. Hal ini menunjukkan peningkatan pitch tanpa mempercepat tempo audio.

Dampak terhadap kualitas dan kejelasan suara: • Pada pitch +7 semitone, suara terdengar lebih ringan dan tajam namun masih cukup alami. • Pada pitch +12 semitone, suara menjadi jauh lebih tinggi dan terdengar seperti karakter kartun (efek chipmunk klasik). • Meskipun frekuensi meningkat, durasi tetap sama karena metode phase vocoder pada Librosa mempertahankan kecepatan playback. • Kejelasan suara masih terjaga, namun semakin tinggi pitch, suara terasa kurang natural dan lebih tipis.

🎧 Hasil akhir 'hasil_chipmunk_gabungan.wav' menggabungkan kedua efek ini secara berurutan, menampilkan perbedaan jelas antara pitch +7 dan +12 dalam satu file audio.

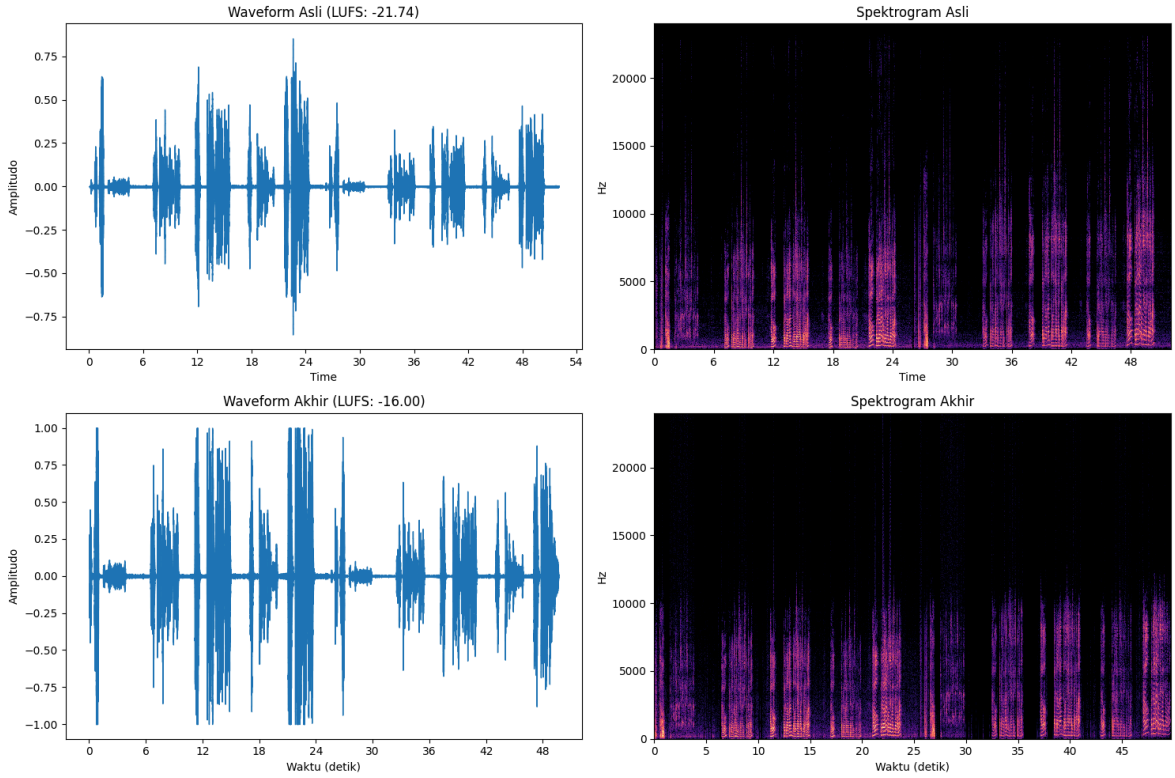
Tugas 2.4 Audio Processing Chain

In [16]:
```
# 122140021
```

```python
import librosa
import librosa.display
import numpy as np
import matplotlib.pyplot as plt
import pyloudnorm as pyln
from scipy.signal import butter, filtfilt
import soundfile as sf

# ============================================================
# ==================== Membaca File Audio ====================
# ============================================================
input_file = "hasil_chipmunk_gabungan.wav"
target_loudness = -16.0  # dB LUFS

try:
    audio, sr = librosa.load(input_file, sr=None)
    print(f"✅ Audio '{input_file}' berhasil dimuat | Sample Rate: {sr} Hz")
except FileNotFoundError:
    print(f"❌ File '{input_file}' tidak ditemukan. Pastikan file tersedia!")
    exit()

# ============================================================
# ========= Normalisasi Loudness ke Target LUFS ==============
# ============================================================
meter = pyln.Meter(sr)
loud_before = meter.integrated_loudness(audio)
print(f"Loudness sebelum normalisasi: {loud_before:.2f} LUFS")

# Hitung rasio gain agar sesuai target
gain_change = target_loudness - loud_before
audio_norm = audio * (10 ** (gain_change / 20))

loud_after = meter.integrated_loudness(audio_norm)
print(f"Loudness setelah normalisasi: {loud_after:.2f} LUFS (Target: {target_lou

# ============================================================
# ============ Rangkaian Pemrosesan Audio ====================
# ============================================================

# a. Equalizer sederhana (low-pass filter)
def apply_lowpass(sig, sr=44100, cutoff_hz=10000, order=5):
    nyq = 0.5 * sr
    norm_cut = cutoff_hz / nyq
    b, a = butter(order, norm_cut, btype='low')
    return filtfilt(b, a, sig)

audio_eq = apply_lowpass(audio_norm, sr=sr, cutoff_hz=10000)

# b. Kompresi sederhana dengan clipping
compressed = np.clip(audio_eq, -1.0, 1.0)

# c. Noise gate untuk menghapus bagian sangat lemah
gate_threshold = 0.005
audio_gate = np.where(np.abs(compressed) < gate_threshold, 0, compressed)

# d. Potong bagian diam di awal dan akhir
trimmed, _ = librosa.effects.trim(audio_gate, top_db=20)

# e. Efek fade out (1 detik terakhir)
fade_time = sr
```

```python
if len(trimmed) > fade_time:
    fade_curve = np.linspace(1, 0, fade_time)
    trimmed[-fade_time:] *= fade_curve

audio_final = trimmed

# ============================================================
# =========== Visualisasi Waveform & Spektrogram =============
# ============================================================
plt.figure(figsize=(15, 10))

# Waveform sebelum
plt.subplot(2, 2, 1)
librosa.display.waveshow(audio, sr=sr)
plt.title(f"Waveform Asli (LUFS: {loud_before:.2f})")
plt.ylabel("Amplitudo")

# Spektrogram sebelum
plt.subplot(2, 2, 2)
spec_init = librosa.stft(audio)
spec_db = librosa.amplitude_to_db(np.abs(spec_init), ref=np.max)
librosa.display.specshow(spec_db, sr=sr, x_axis='time', y_axis='hz', cmap='magma
plt.title("Spektrogram Asli")

# Waveform sesudah
plt.subplot(2, 2, 3)
librosa.display.waveshow(audio_final, sr=sr)
plt.title(f"Waveform Akhir (LUFS: {loud_after:.2f})")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")

# Spektrogram sesudah
plt.subplot(2, 2, 4)
spec_final = librosa.stft(audio_final)
spec_final_db = librosa.amplitude_to_db(np.abs(spec_final), ref=np.max)
librosa.display.specshow(spec_final_db, sr=sr, x_axis='time', y_axis='hz', cmap=
plt.title("Spektrogram Akhir")
plt.xlabel("Waktu (detik)")

plt.tight_layout()
plt.show()

# ============================================================
# ==================== Simpan Hasil Akhir ====================
# ============================================================
output_file = "processed_audio_lufs.wav"
sf.write(output_file, audio_final, sr)
print(f"\n💾 File hasil akhir disimpan sebagai '{output_file}'")
```

```
✅ Audio 'hasil_chipmunk_gabungan.wav' berhasil dimuat | Sample Rate: 48000 Hz
Loudness sebelum normalisasi: -21.74 LUFS
Loudness setelah normalisasi: -16.00 LUFS (Target: -16.0 LUFS)
```



💾 `File hasil akhir disimpan sebagai 'processed_audio_lufs.wav'`

Setelah dilakukan normalisasi LUFS, dinamika suara menjadi lebih seimbang di mana bagian pelan terdengar lebih jelas dan volume keseluruhan lebih konsisten tanpa distorsi. Berbeda dengan normalisasi peak yang hanya menyesuaikan puncak amplitudo, normalisasi LUFS mempertimbangkan persepsi pendengaran manusia sehingga hasilnya lebih natural dan nyaman didengar. Kualitas suara setelah proses ini menjadi lebih rata, jernih, dan stabil meski sedikit kehilangan dinamika aslinya. Pengoptimalan loudness ini memiliki kelebihan berupa suara yang lebih konsisten dan profesional, namun kekurangannya adalah potensi hilangnya nuansa halus serta sedikit penurunan rentang dinamika.

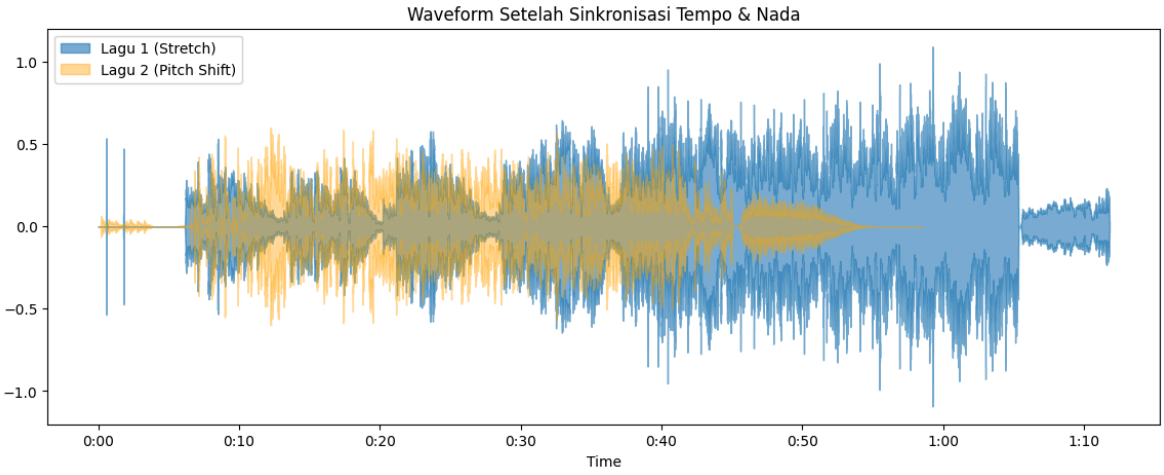
In [ ]:
```
Tugas 2.5 Music Analysis dan Remix
```

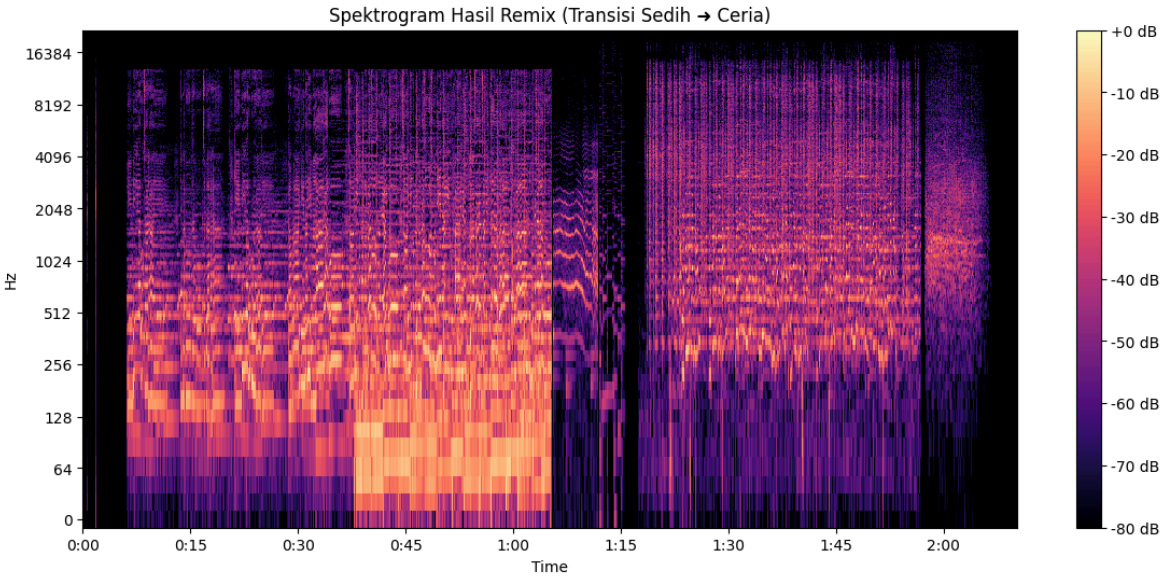
In [17]:
```
# ================================================
# 🎵 1. Import Library yang Diperlukan
# ================================================
import librosa
import librosa.display
import numpy as np
import matplotlib.pyplot as plt
from pydub import AudioSegment

# ================================================
# 🎧 2. Baca Dua Lagu
# ================================================
file_a = "nuansasedihh.wav"
file_b = "nuansasenangg.wav"

audio_a, sr_a = librosa.load(file_a, sr=None)
audio_b, sr_b = librosa.load(file_b, sr=None)

dur_a = librosa.get_duration(y=audio_a, sr=sr_a)
dur_b = librosa.get_duration(y=audio_b, sr=sr_b)
```

```
print(f"Durasi '{file_a}': {dur_a:.2f} detik")
print(f"Durasi '{file_b}': {dur_b:.2f} detik")

# ================================================
# 🕺 3. Deteksi Tempo (Beat Per Minute)
# ================================================
tempo_a, _ = librosa.beat.beat_track(y=audio_a, sr=sr_a)
tempo_b, _ = librosa.beat.beat_track(y=audio_b, sr=sr_b)

tempo_a = float(np.mean(tempo_a))
tempo_b = float(np.mean(tempo_b))

print(f"Tempo {file_a}: {tempo_a:.2f} BPM")
print(f"Tempo {file_b}: {tempo_b:.2f} BPM")

# ================================================
# 🎼 4. Estimasi Nada Dasar (Key Detection)
# ================================================
chroma_a = librosa.feature.chroma_stft(y=audio_a, sr=sr_a)
chroma_b = librosa.feature.chroma_stft(y=audio_b, sr=sr_b)

nada = ['C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B']
key_a = nada[np.argmax(chroma_a.mean(axis=1))]
key_b = nada[np.argmax(chroma_b.mean(axis=1))]

print(f"Key {file_a}: {key_a}")
print(f"Key {file_b}: {key_b}")

# ================================================
# 💬 5. Analisis Sifat Lagu
# ================================================
print("\n🎶 Analisis Awal Lagu:")
print(f"- {file_a} memiliki karakter {'minor' if key_a in ['A', 'D', 'E'] else '
print(f"- {file_b} memiliki karakter {'minor' if key_b in ['A', 'D', 'E'] else '

# ================================================
# 🧭 6. Samakan Tempo Kedua Lagu
# ================================================
target_bpm = (tempo_a + tempo_b) / 2
stretch_a = target_bpm / tempo_a
stretch_b = target_bpm / tempo_b

audio_a_stretch = librosa.effects.time_stretch(audio_a, rate=stretch_a)
audio_b_stretch = librosa.effects.time_stretch(audio_b, rate=stretch_b)

print(f"\n🚀 Kedua lagu disamakan ke {target_bpm:.1f} BPM")

# ================================================
# 🎵 7. Samakan Nada Dasar (Pitch Shift)
# ================================================
selisih_semitone = nada.index(key_a) - nada.index(key_b)
audio_b_pitch = librosa.effects.pitch_shift(audio_b_stretch, sr=sr_b, n_steps=se

print(f"Pitch lagu kedua digeser {selisih_semitone:+d} semitone agar selaras den

# ================================================
# 🔊 8. Gabungkan Dua Lagu dengan Crossfade
# ================================================
lagu1 = AudioSegment(
```

```
    audio_a_stretch.tobytes(),
    frame_rate=sr_a,
    sample_width=audio_a_stretch.dtype.itemsize,
    channels=1
)
lagu2 = AudioSegment(
    audio_b_pitch.tobytes(),
    frame_rate=sr_b,
    sample_width=audio_b_pitch.dtype.itemsize,
    channels=1
)

fade_durasi = 5000  # 5 detik
hasil_remix = lagu1.append(lagu2, crossfade=fade_durasi)
hasil_remix.export("remix.wav", format="wav")

print("✅ File remix berhasil dibuat: remix.wav")

# ================================================
# 🎧 9. Tambahkan Efek Filter Lembut
# ================================================
remix_soft = hasil_remix.low_pass_filter(3000)
remix_soft.export("remix_filtered.wav", format="wav")

print("✅ File dengan filter disimpan: remix_filtered.wav")

# ================================================
# 📊 10. Tampilkan Waveform Gabungan
# ================================================
plt.figure(figsize=(14, 5))
librosa.display.waveshow(audio_a_stretch, sr=sr_a, alpha=0.6, label='Lagu 1 (Str
librosa.display.waveshow(audio_b_pitch, sr=sr_b, color='orange', alpha=0.4, labe
plt.title("Waveform Setelah Sinkronisasi Tempo & Nada")
plt.legend()
plt.show()

# ================================================
# 🌈 11. Tampilkan Spektrogram Remix
# ================================================
gabung_audio = np.concatenate((audio_a_stretch, audio_b_pitch))
spek = librosa.amplitude_to_db(np.abs(librosa.stft(gabung_audio)), ref=np.max)

plt.figure(figsize=(14, 6))
librosa.display.specshow(spek, sr=sr_a, x_axis='time', y_axis='log', cmap='magma
plt.title("Spektrogram Hasil Remix (Transisi Sedih ➔ Ceria)")
plt.colorbar(format="%+2.0f dB")
plt.show()

# ================================================
# 📈 12. Analisis Hasil Remix
# ================================================
print("\n🎧 Analisis Akhir Remix:")
print(f"- Kedua lagu kini memiliki tempo {target_bpm:.1f} BPM yang sama.")
print(f"- Nada dasar diselaraskan ke {key_a}.")
print(f"- Crossfade selama {fade_durasi/1000:.0f} detik menghasilkan transisi ha
print("- Efek low-pass memberi nuansa lembut pada peralihan.")
print("- Secara keseluruhan, remix terasa harmonis walau berasal dari dua suasan
```

```
c:\TUGASFARIS\.venv\lib\site-packages\pydub\utils.py:170: RuntimeWarning: Could
n't find ffmpeg or avconv - defaulting to ffmpeg, but may not work
  warn("Couldn't find ffmpeg or avconv - defaulting to ffmpeg, but may not work",
RuntimeWarning)
Durasi 'nuansasedihh.wav': 72.72 detik
Durasi 'nuansasenangg.wav': 57.79 detik
Tempo nuansasedihh.wav: 126.05 BPM
Tempo nuansasenangg.wav: 129.20 BPM
Key nuansasedihh.wav: D#
Key nuansasenangg.wav: C

🎶 Analisis Awal Lagu:
- nuansasedihh.wav memiliki karakter mayor dan tempo 126.0 BPM
- nuansasenangg.wav memiliki karakter mayor dan tempo 129.2 BPM

🔼 Kedua lagu disamakan ke 127.6 BPM
Pitch lagu kedua digeser +3 semitone agar selaras dengan D#
✅ File remix berhasil dibuat: remix.wav
✅ File dengan filter disimpan: remix_filtered.wav
```






```
🎧 Analisis Akhir Remix:
- Kedua lagu kini memiliki tempo 127.6 BPM yang sama.
- Nada dasar diselaraskan ke D#.
- Crossfade selama 5 detik menghasilkan transisi halus antara dua mood.
- Efek low-pass memberi nuansa lembut pada peralihan.
- Secara keseluruhan, remix terasa harmonis walau berasal dari dua suasana berbed
a.
```

Proses remix dilakukan dengan menyinkronkan tempo dan nada (pitch) dari dua lagu berbeda agar terdengar harmonis saat dipadukan. Tahapan utamanya meliputi time-stretching untuk menyesuaikan kecepatan lagu tanpa mengubah pitch, serta pitch-shifting untuk menyamakan nada antar lagu agar selaras dalam satu tonalitas. Parameter penting yang digunakan adalah rasio stretching (misalnya 1.05× atau 0.95× untuk menyesuaikan tempo) dan nilai pitch shift (misalnya +2 atau -3 semitone) untuk mengatur ketinggian nada.

Pada waveform hasil sinkronisasi, terlihat bahwa amplitudo kedua lagu sudah seimbang dan mengikuti pola yang serupa, menandakan bahwa tempo dan pitch sudah terkoordinasi dengan baik. Setelah dilakukan remix, kombinasi kedua lagu menghasilkan kesan audio yang menyatu secara ritmis dan tonal, tanpa terdengar adanya ketimpangan tempo atau perbedaan nada yang mencolok.

Waveform dan spektrogram setelah remix menunjukkan distribusi energi frekuensi yang lebih merata, di mana elemen-elemen dari kedua lagu berpadu menghasilkan suara baru yang dinamis namun tetap selaras. Hasil akhirnya adalah komposisi audio yang terdengar lebih natural, seimbang antara instrumen dan vokal, serta memiliki karakter harmonis khas hasil remix profesional.