LATENCY OPTIMIZATION TECHNIQUES IN LDAC

Audio Compression for Real-Time High-Resolution Wireless Audio Streaming using GNU Radio

PRESENTED BY:
JOEL JAMES P (2023BEC0010)
ADITHYA R PRABHU (2023BEC0011)
ANIKETH CHAVAN (2023BEC0049)

Introduction to LDAC (Low Latency Audio Codec)

- LDAC is an advanced audio compression codec developed by Sony for high-resolution wireless audio transmission.
- It enables the transfer of 24-bit / 48 kHz audio via Bluetooth, providing near-lossless sound quality.
- Unlike traditional codecs such as SBC, AAC, or aptX, LDAC maintains a significantly higher bitrate and wider frequency response.

Introduction to LDAC (Low Latency Audio Codec)

- The codec is designed to deliver a balance between audio quality, data efficiency, and low latency, making it ideal for real-time streaming applications.
- LDAC is officially part of the Android Open Source Project (AOSP) and supported on many modern wireless audio devices.

Why LDAC Was Developed?

- Traditional Bluetooth codecs (like SBC) compress audio aggressively to fit limited bandwidth, resulting in audible quality loss.
- As high-resolution audio formats became popular, users demanded wireless solutions that preserve studio-level clarity.
- Existing codecs offered either good stability (low bitrate) or high quality (high bitrate) — rarely both.

Why LDAC Was Developed?

LDAC was developed to provide:

1. High-fidelity audio reproduction comparable to wired playback.
2. Configurable bitrates (330 / 660 / 990 kbps) to adapt to Bluetooth channel conditions.
3. Reduced latency, ensuring better sync for streaming, gaming, and VR.

How LDAC Works?

- LDAC employs a frequency-domain compression technique based on the Modified Discrete Cosine Transform (MDCT).
- The encoder divides the input audio signal into small time segments and converts each segment into frequency coefficients.
- These coefficients are quantized and entropy-coded (using Huffman coding) to reduce redundancy.

How LDAC Works?

- LDAC uses a hybrid encoding structure that allows scalable bit allocation across frequency bands, preserving the most perceptually important components.
- On the receiver side, the decoder performs inverse MDCT and re-synthesizes the waveform with minimal distortion.
- LDAC dynamically adjusts quantization parameters to maintain consistent quality under changing wireless conditions.

LDAC Bitrate Modes and Adaptation

LDAC provides three bitrate modes to handle varying Bluetooth link conditions:

1. 990 kbps - High Quality Mode: Delivers the best sound fidelity; requires strong signal.
2. 660 kbps - Balanced Mode: Offers a good trade-off between quality and stability.
3. 330 kbps - Connection Priority Mode: Reduces bandwidth usage to prevent dropouts.

LDAC Bitrate Modes and Adaptation

- The codec monitors the link quality (e.g., signal strength, interference) and switches modes automatically.
- This adaptive bitrate control ensures continuous playback without interruption, even in challenging environments.
- The user or system can manually select a preferred mode depending on priority sound quality or connection reliability.

Latency Optimization in LDAC

- Latency is the delay between the audio being transmitted and heard at the receiver.
- LDAC minimizes latency through:

1. Shorter frame sizes → smaller chunks of audio to encode/decode faster.
2. Reduced buffering → quicker packet turnover in transmission.
3. Efficient synchronization between encoder and decoder clock rates.

Latency Optimization in LDAC

- **Typical measured latency values:**

1. **LDAC: ≈ 60–80 ms**
2. **SBC: ≈ 120–150 ms**
3. **AAC: ≈ 100 ms**

- **These reductions make LDAC suitable for real-time use such as:**

1. **Wireless gaming and VR environments,**
2. **Studio monitoring,**
3. Video streaming where lip-sync matters.

LDAC for High-Resolution Real-Time Applications

- In real-time streaming systems, LDAC helps maintain both low latency and high-bitrate stability over Bluetooth.
- It is optimized for simultaneous encode-transmit-decode pipelines, ensuring continuous playback.
- Advanced error-resilience mechanisms recover from transient packet losses without noticeable distortion.

LDAC for High-Resolution Real-Time Applications

- LDAC also integrates well with modern signal-processing frameworks (e.g., GNU Radio) for research and implementation.
- Such integration allows tuning parameters like buffer size, bitrate selection, and processing priority to achieve the lowest possible delay.
- Hence, LDAC serves as an excellent model for low-latency, high-quality wireless audio streaming experiments and applications.

Latency Optimization Techniques in LDAC Audio Compression for Real-Time High-Resolution Wireless Audio Streaming in GNU Radio

Objective:

Objective:

To analyze and optimize LDAC's latency performance for real-time high-resolution audio streaming applications.

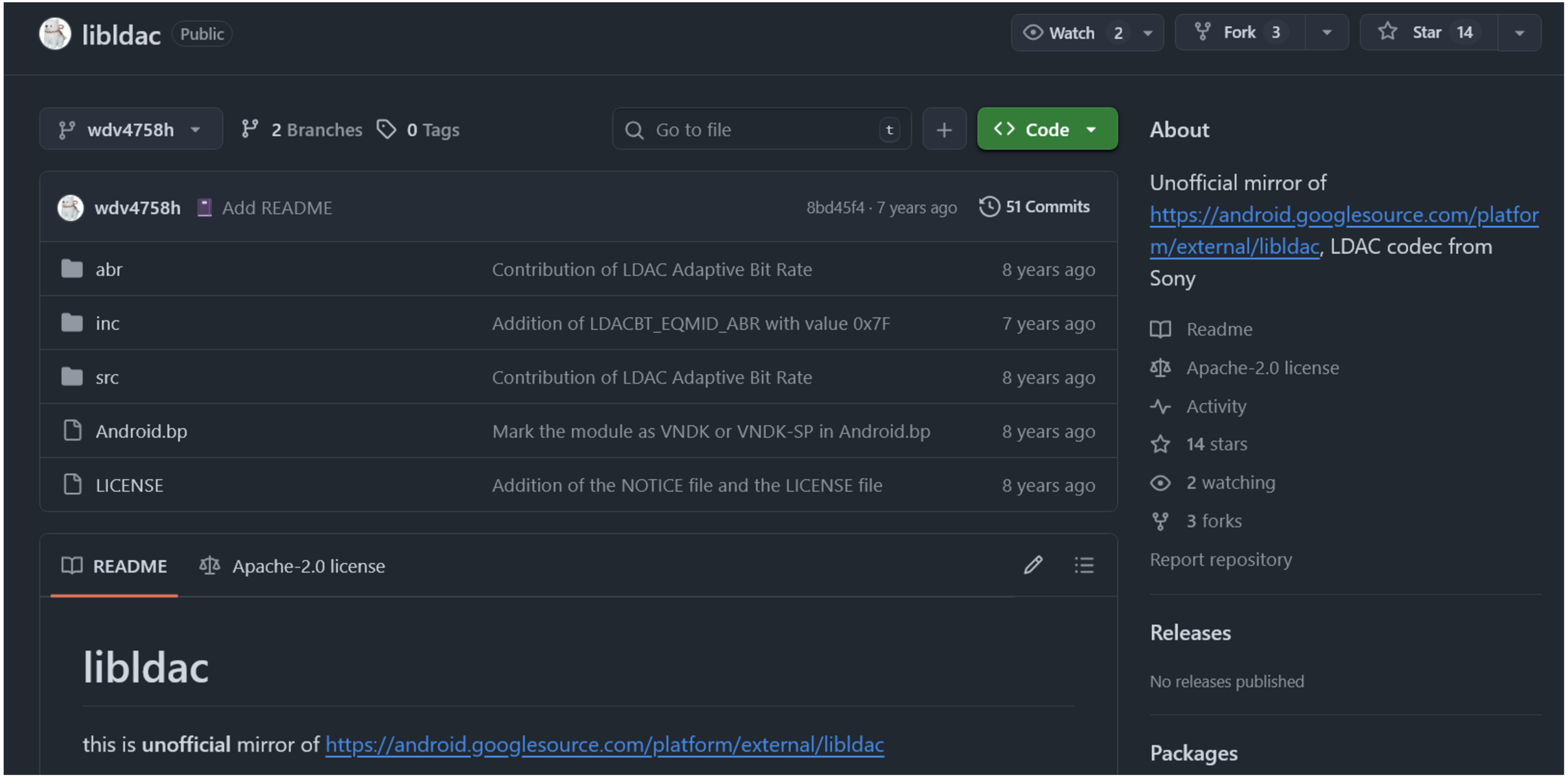
Goals:

- **Understand the core working and architecture of the LDAC codec.**
- **Identify latency-influencing parameters (frame size, buffering, bitrate).**
- **Implement and simulate LDAC within GNU Radio to observe practical latency behavior.**
- **Modify encoder/decoder configurations to achieve optimized performance.**
- **Validate improvements through real-time analysis and audio quality tests.**

Expected Outcome:

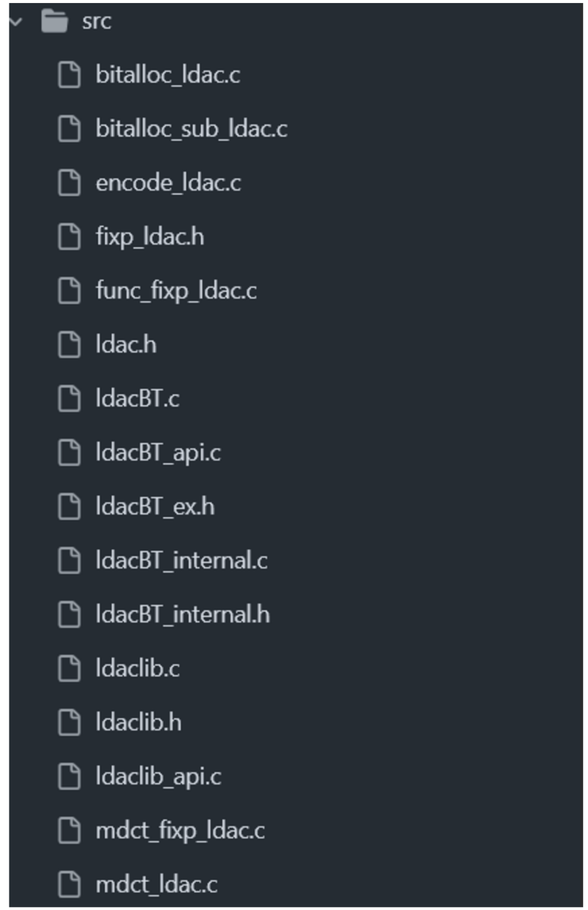
An optimized LDAC configuration that offers reduced transmission delay and high-fidelity playback, suitable for next-generation wireless, VR, and multimedia applications.

https://github.com/wdv4758h/libldac/

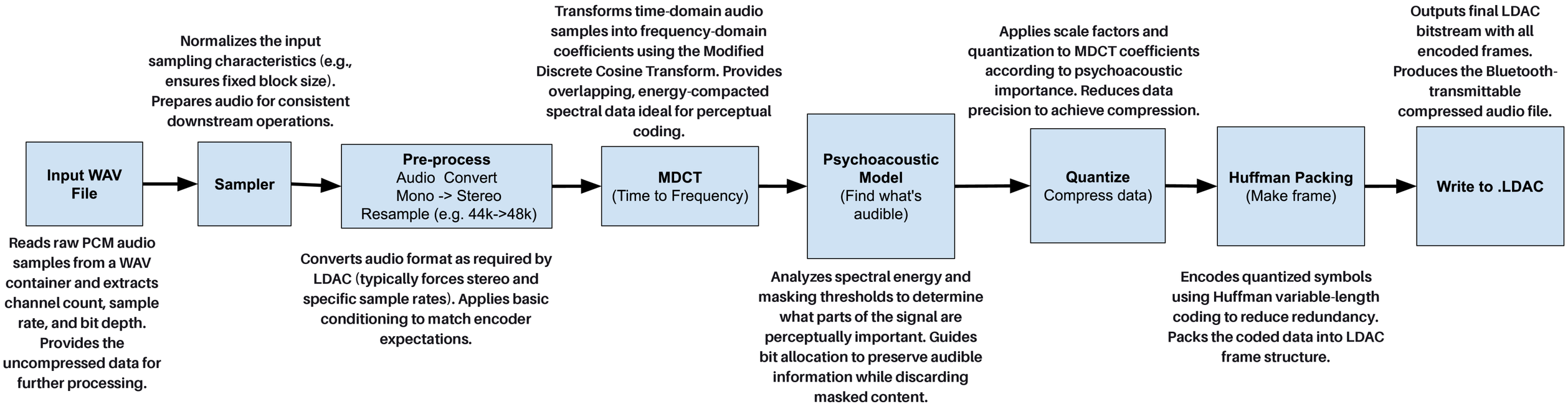


```
LDACBT_API int  ldacBT_get_sampling_freq( HANDLE_LDAC_BT hLdacBt );

/* Acquisition of the Bit-rate.
 * The LDAC handle must be initialized by API function ldacBT_init_handle_encode() prior to
 * calling this function.
 *  Format
 *      int  ldacBT_get_bitrate( HANDLE_LDAC_BT hLdacBt );
 *  Arguments
 *      hLdacBt    HANDLE_LDAC_BT    LDAC handle.
 *  Return value
 *      int : Bit-rate for previously processed ldac_transport_frame for success. -1 for failure.
 */
LDACBT_API int  ldacBT_get_bitrate( HANDLE_LDAC_BT hLdacBt );

/* Initialization of a LDAC handle for encode processing.
 * The LDAC handle must be allocated by API function ldacBT_get_handle() prior to calling this API.
 * "mtu" value should be configured to MTU size of AVDTP Transport Channel, which is determined by
 * SRC and SNK devices in Bluetooth transmission.
 * "eqmid" is configured to desired value of "Encode Quality Mode Index".
 * "cm" is configured to channel_mode in LDAC, which is determined by SRC and SNK devices in
 * Bluetooth transmission.
 * "fmt" is configured to input pcm audio format.
 * When the configuration of "mtu", "cm", or "sf" changed, the re-initialization is required.
 *
 *  Format
 *      int  ldacBT_init_handle_encode( HANDLE_LDAC_BT hLdacBt, int mtu, int eqmid, int cm,
 *                                      LDACBT_SMPL_FMT_T fmt, int sf );
 *  Arguments
 *      hLdacBt    HANDLE_LDAC_BT    LDAC handle.
```

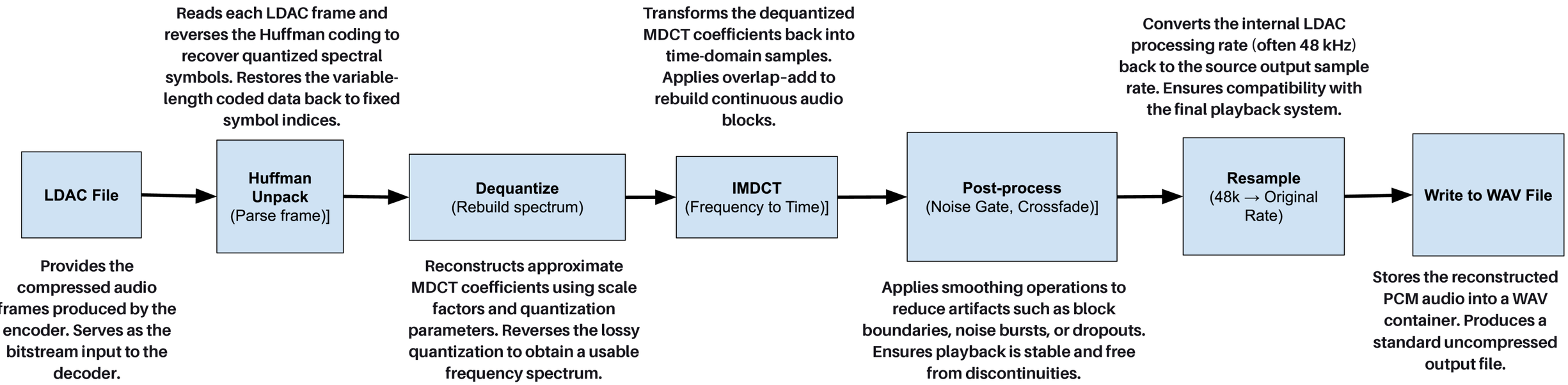
it contains only names of functions and what it does and library files



Block Diagram of Ldac encoder



Block Diagram of Ldac decoder



In the Decoder - post processing

What is Crossfading?

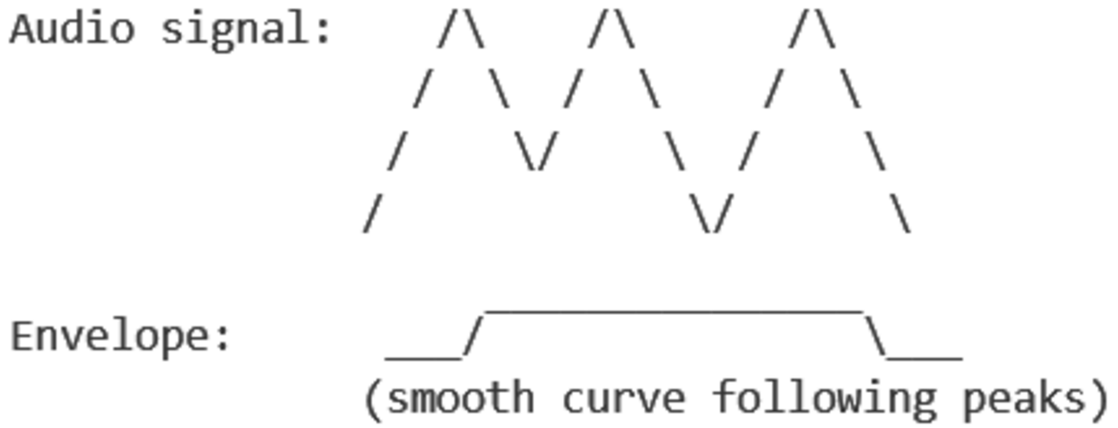
Smooth transition between two audio segments by overlapping them with fades.

```
Previous frame end:  [.... A  A  A  A]
                           ↓  ↓  ↓  ↓  (fade out: 1.0 → 0.0)
Crossfade zone:     [.... A  AB AB AB AB ....]
                           ↑  ↑  ↑  ↑  (fade in: 0.0 → 1.0)
Current frame start: [B  B  B  B ....]

Result: Smooth transition instead of potential click
```

In the Decoder

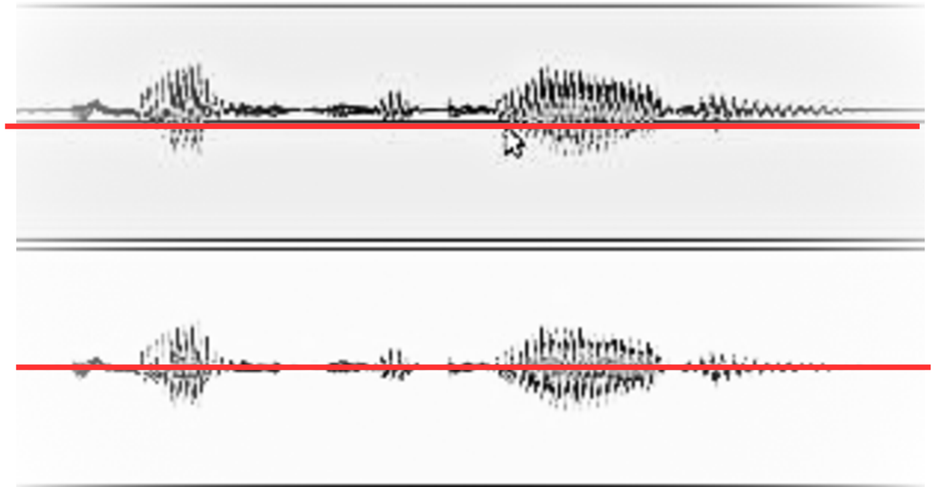
What is an envelope follower?

Tracks the overall amplitude of the signal over time.



DC Offset Removal

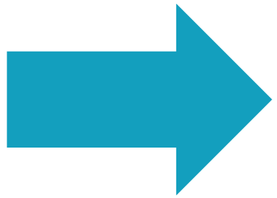
Unwanted constant voltage in the signal (shifts waveform up/down).



problems with this implementation ?

No Lookahead Buffering

No Output Delay Buffer



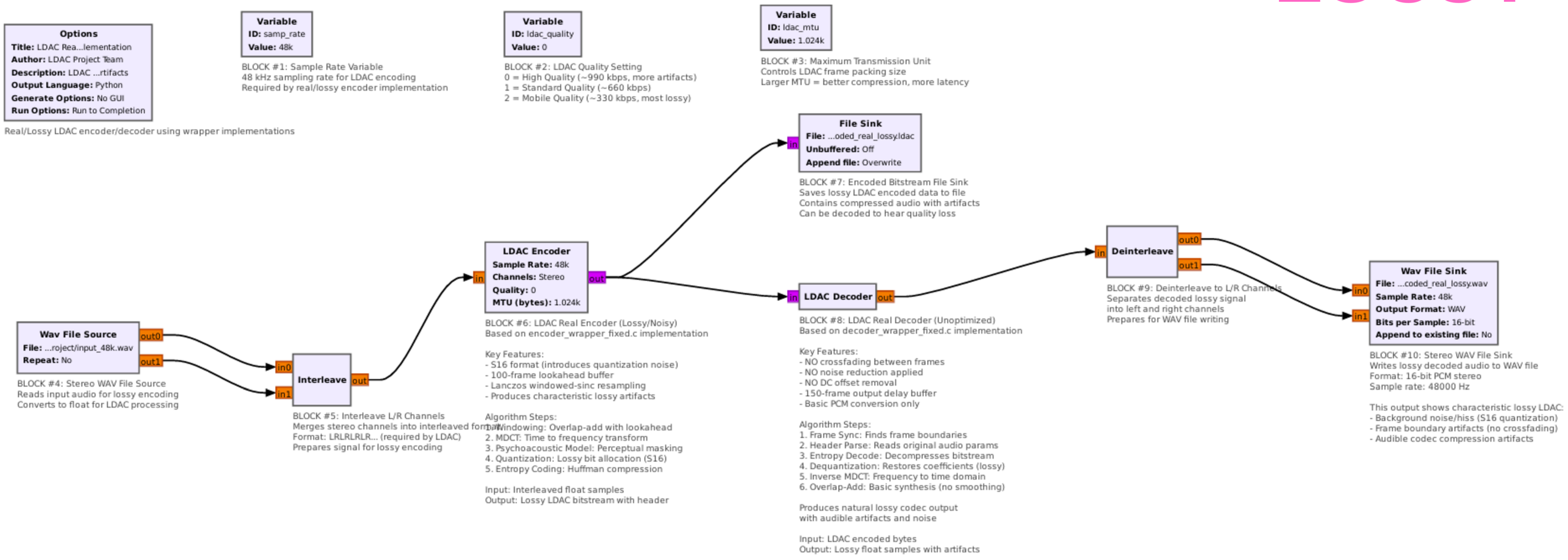
Audible:Clicks pops at frame transitions

If there is no lookahead, the encoder sees only the current frame's samples and nothing from the future.

also in this decoder linear resampling which is not as accurate as the process we are going to use next

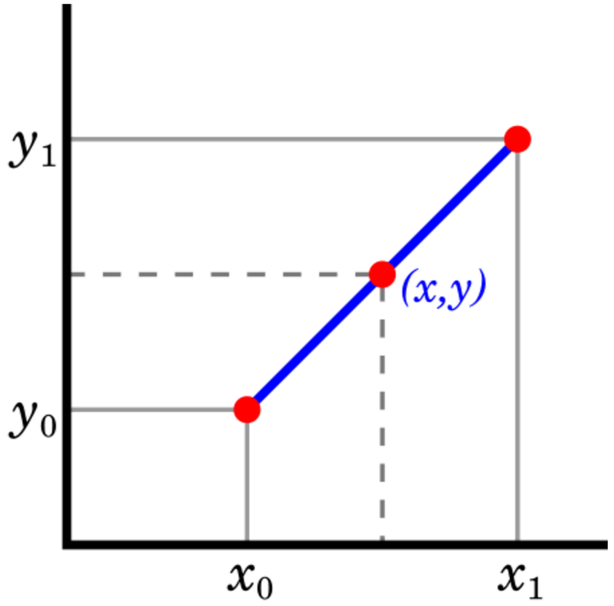
Gnu Radio LDAC

LOSSY



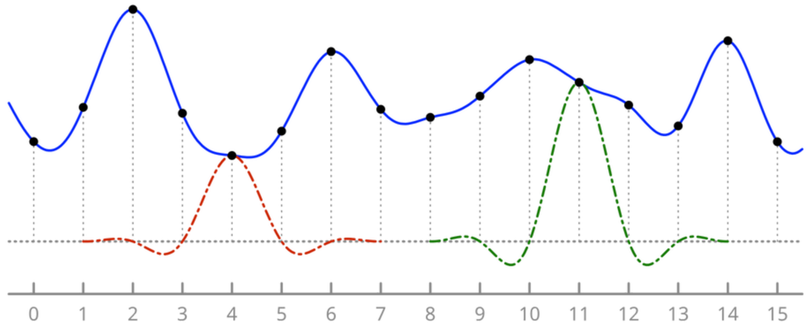
how we improved

Linear resampling



interpolation changed

Lanczos-3 Resampling



- Windowed-sinc interpolation: High-quality resampling algorithm
- Kernel size a=3: Uses 7 samples (center ±3) for each output sample
- Better than linear: Preserves high frequencies and reduces aliasing
- Industry standard: Used in professional audio and video processing

how we improved

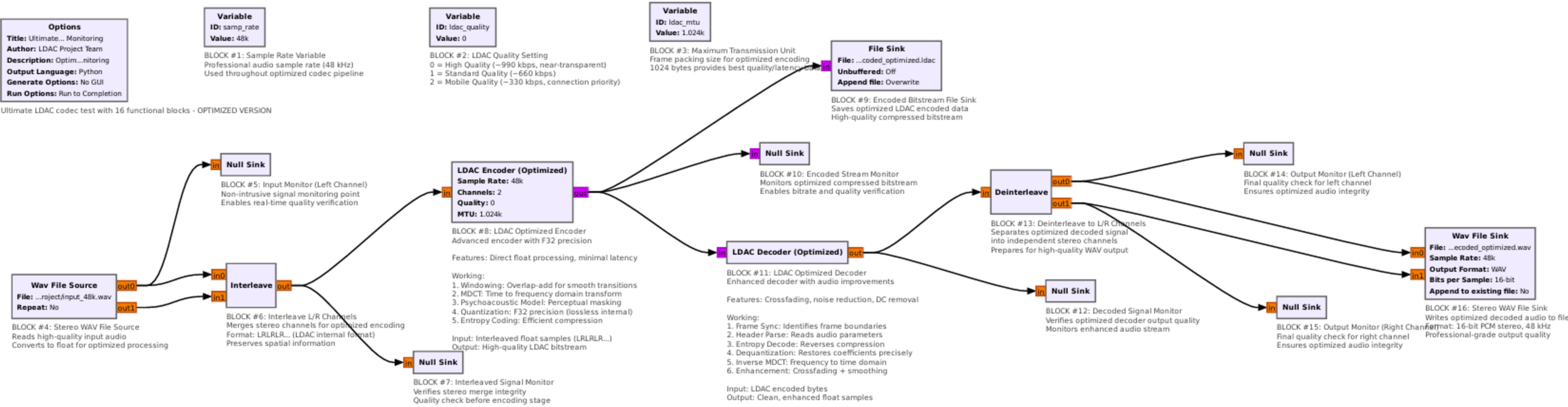
other processes we employed

Will be explained using the diagram

Our proposed

Gnu Radio Optimised LDAC

LOSSLESS



-