AN13793

FLAC Porting and Codec Performance Evaluation based on i.MX RT685

Rev. 0 — 30 November 2022 Application note

Document information

Information	Content
Keywords	FLAC
Abstract	This document introduces the FLAC codec library porting on i.MX RT6xx, compares the performance differences between CM33 and HiFi4 DSP, uses the GNU profiler tool to analyze time-consuming functions in the FLAC library, and provides recommendations for further performance optimization.

1 Introduction

This document:

- Introduces the Free Lossless Audio Codec (FLAC) codec library porting on i.MX RT6xx.
- Compares the performance differences between CM33 and HiFi4 DSP.
- Uses the GNU profiler tool to analyze time-consuming functions in the FLAC library.
- Provides recommendations for further performance optimization.

2 Introduction to FLAC

FLAC is an audio format similar to MP3, but lossless, meaning that audio is compressed in FLAC without any loss in quality. The FLAC encode falls into the following four steps:

1. Blocking
2. Inter-Channel Decorrelation
3. Modeling
4. Residual coding

For more details and version differences, see FLAC. The following evaluations are based on the FLAC Rev. 1.3.4.

3 Evaluation environment

3.1 i.MX RT685

- Next-generation Cortex-M33 control processor core is running at a frequency up to 300 MHz.
- Two coprocessors are for the Cortex-M33:
  - PowerQuad hardware accelerator for (fixed and floating-point unit) DSP functions.
  - CASPER Crypto coprocessor to enable hardware acceleration for various functions required for certain asymmetric cryptographic algorithms.
- Highly Optimized Cadence Tensilica HiFi 4 DSP Processor Core is running at frequencies of up to 600 MHz.
  - Hardware floating-point unit, up to four single-precision IEEE floating-point MACs per cycle.
- Cortex-M33 built-in Memory Protection Unit (MPU) supporting eight regions.
- Up to 4.5 MB of system SRAM accessible by both CPUs and all DMA engines.

3.2 i.MX RT600 EVK

The i.MX RT600 EVK (MIMXRT685-EVK) features the advanced implementation of the NXP Arm Cortex-M33 core, combined with the highly optimized Cadence Tensilica HiFi 4 DSP processor core. Its features make it ideal for ML/AI, voice, and audio applications.

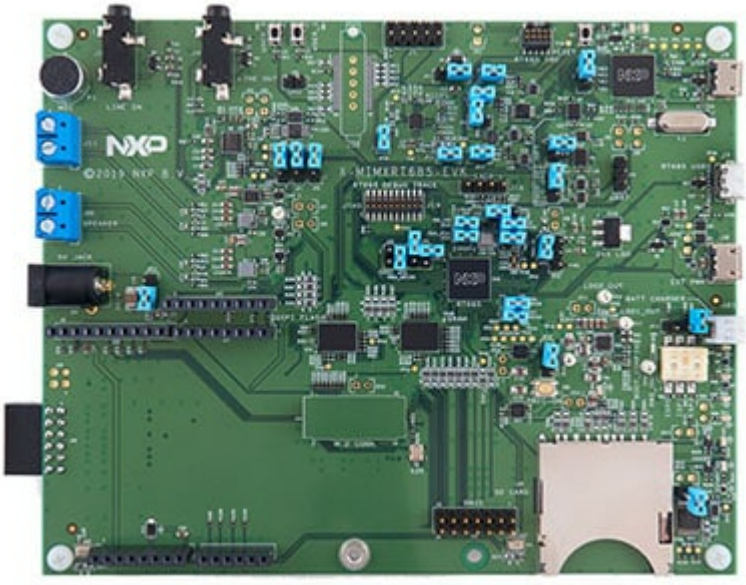

Figure 1. Hardware platform

4 FLAC porting

We can download the latest version of FLAC from GitHub. The FLAC library contains two part source code which are **encoder** and **decoder**. Here takes version 1.3.4 as an example.

Since we must evaluate on the CM33 and DSP cores separately, we must port FLAC to two different IDE tools. IAR and Xplorer are slightly different. We use IAR IDE to perform a porting demonstration and give some porting advise on Xplorer in Section 4.6. To maintain the integrity of the source code, we use a macro definition FLAC_EMBEDDED to differentiate the two versions during the pre-compilation phase.

4.1 FLAC library directory structure

The FLAC library is implemented in both C and CPP. FLAC is getting bigger and bigger due to iterations of versions and evaluation on different hardware platforms. In addition to the core codec source code, it also includes documentation, test code, compilation scripts, and so on. For our embedded devices, only basic codec functions are required. As shown in Figure 2, three directories are more important and the rest can be ignored:

1. **Src/libFLAC directory**: The core code location. We use the C version of the source code. The directory consists of source code and include header files. The libFLAC had optimized for some hardware platforms (for example, SSE2, SSSE3, Neon, VSX, and so on) with platform intrinsic assemblies. For example, fixed_intrin_sse2.c indicates that the fixed.c function is optimized for the SSE2 (Intel Platform). Because Arm CM33 and HiFi4 DSP are not optimized by official, we can only use the basic version (function name without suffix).
2. **Examples/c directory:** The demo code location. There are two versions in this directory (C and CPP). We still use the C version here. There are main.c both at encode and decode subdirectory which are good reference for porting.
3. **Include directory**: In addition to the libFLAC/include directory, there are also some header files located in subdirectory **FLAC** and **share**, which must be included.

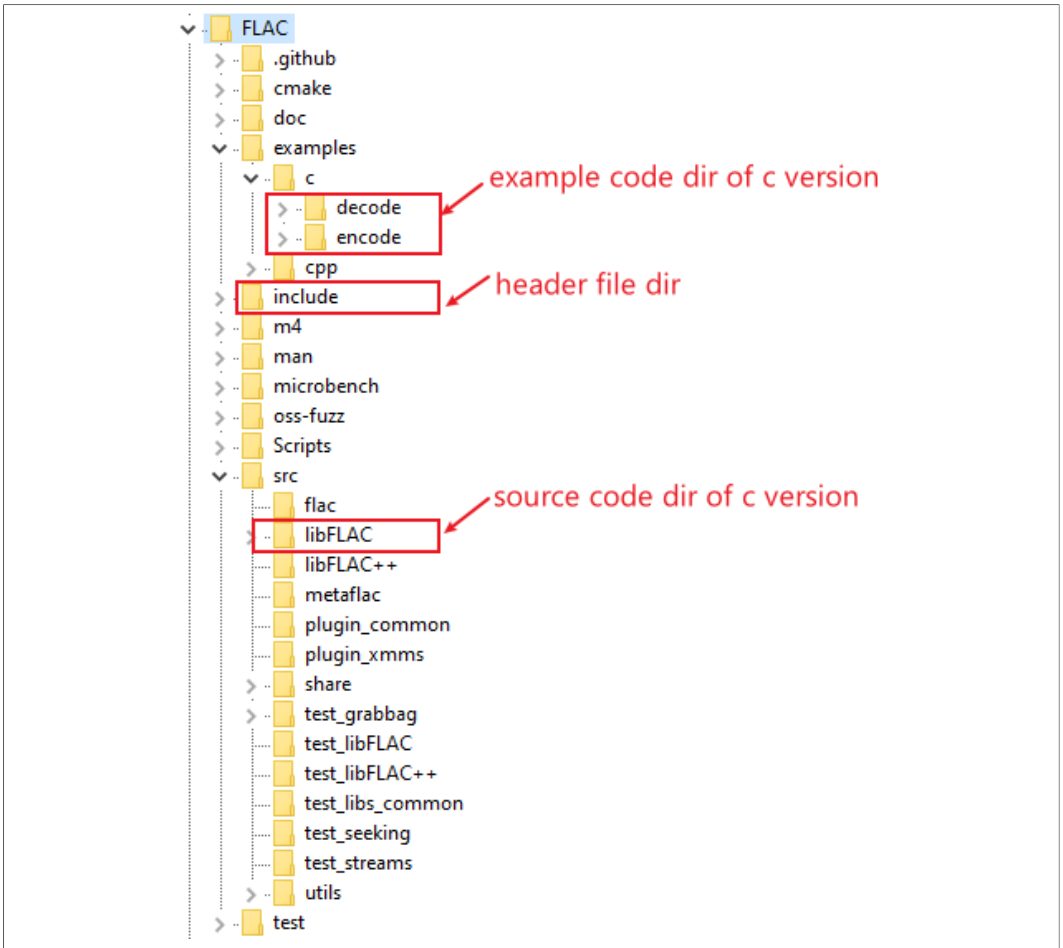

Figure 2. FLAC source directory structure

Table 1 lists dependent files required for FLAC porting.

Table 1. FLAC core source code list

No.	Function name	Function introduction	Modification required for porting	Files that Decode needs	Files that Encode needs
1	Bitmath.c	Provides the silog2 calculation function		√	√
2	Bitreader.c	Read the Bit-stream interface		√	√
3	Bitwriter.c	Write the Bit stream interface			√
4	Cpu.c	Provides an interface function for querying the CPU version model			√
5	Crc.c	CRC verifies the interface function		√	√
6	Fixed.c	Fixed-order polynomial fitting interface function		√	√

Table 1. FLAC core source code list...*continued*

No.	Function name	Function introduction	Modification required for porting	Files that Decode needs	Files that Encode needs
7	Format.c	FLAC formats the correlation functions		√	√
8	Lpc.c	Linear prediction computes related interface functions		√	√
9	Md5.c	Provides the MD5 verification interface function		√	√
10	Memory.c	Dynamic memory application-related functions		√	√
11	Metadata_object.c	Generate the interface functions for the FLAC metadata			√
12	Stream_decoder.c	Decode the core functions	√	√	√
13	Stream_encoder.c	Coding core functions	√		√
14	Stream_encoder_framing.c	Encode to form a frame function			√
15	Window.c	Window function functions, haming, hann, tukey			√

Modify the code in `Stream_decoder.c` and `Stream_encoder.c` for porting.

4.2 Use FatFs to replace C library file system

By analyzing the FLAC source code, we can find that the example/C routine is based on standard C library function interfaces, such as, fopen, fwrite, and fread. The main flow of the two routines is as follows:

1. In the encode routine:
   a. libFLAC reads the input WAV audio file and parses the WAV audio header 44 bytes to set the encoder initialization parameters.
   b. The data is divided into blocks and looped sent to encoder.
   c. The compressed audio file is output by callback function.
2. In the decode routine:
   a. libFLAC reads the bit stream from the FLAC file and recognizes each frame through the synchronization flag.
   b. Decoder decides the corresponding decompression method by parsing compression method flag which stored in each frame header.
   c. It generates PCM data output and saves them into a WAV file.

The encoder and decoder of FLAC has two sets of APIs: filed-based (green arrows in Figure 3) and stream-based (orange arrows in Figure 3). Because we must get the final codec file as output, we use a file-based approach to porting. File system in LIBFLAC

library is changed from C-library file system interface to the FatFs interface, such as, replacing `fopen` with `f_open`, replacing `fwrite` with `f_write`. Figure 3 shows the block diagram.

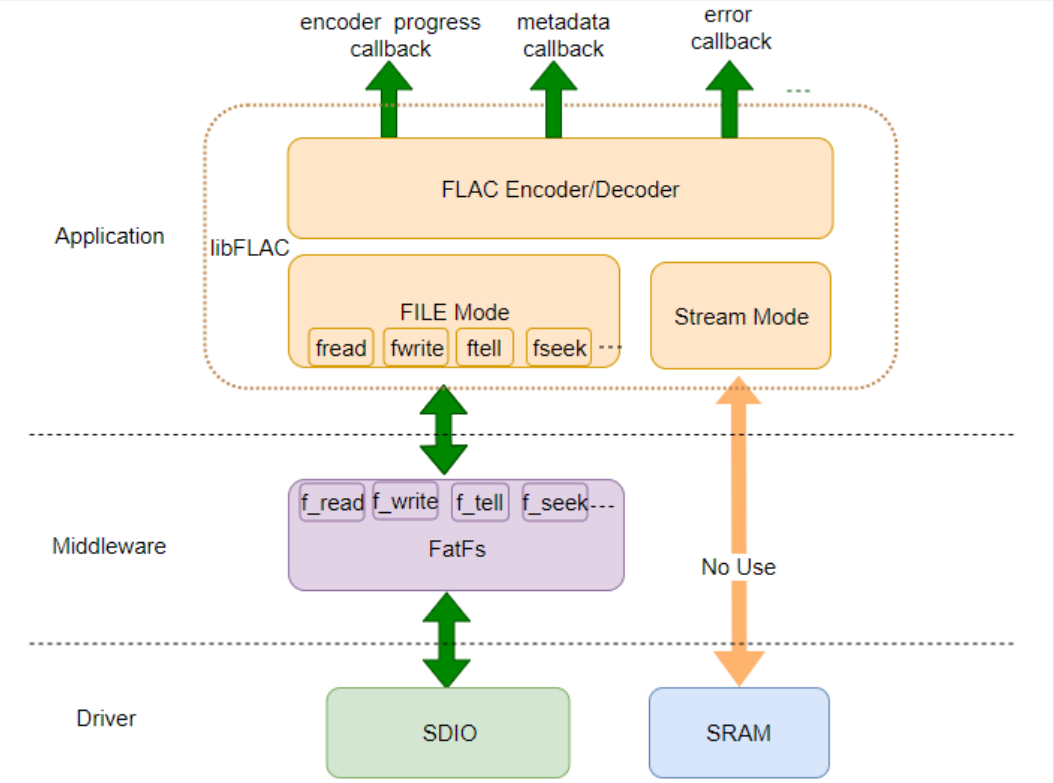

Figure 3. FLAC data flow diagram after joining FatFs

4.3 SD card driver porting

For SD CARD driver, we use the SDIO hardware interface of RT6xx with 4-bits transfer mode. To read and write the SD card on the PC, we use the **FatFs** open source file system. For FatFs porting, only implement the basic SD card read, write sector, and get registers, and so on, in diskio.c. For porting details, see the SDK example: `i.MX_RT600\SDK\boards\evkmimxrt685\sdmmc_examples\sdcard_fatfs`.

4.4 Add macros definitions

There are many macro definitions in the FLAC source code, which can help us switch to different hardware platform conveniently. Table 2 describes the macros definitions to add in the porting.

Table 2. Macro definition table to be added

Macros definition to be added	Description
FLAC__HAS_OGG=0	We use Native FLAC instead of Ogg FLAC here, so we turn off OGG.
PACKAGE_ VERSION="1.3.4"	Manually define the package version string

Table 2. Macro definition table to be added*...continued*

Macros definition to be added	Description
HAVE_LROUND	Indicates that IAR and Xplorer support the lround function and do not use the build_in implementation
HAVE_STDINT_H	Indicates that IAR and Xplorer support stdint.h header files
FLAC_EMBEDDED	FLAC uses a ported embedded interface

4.5 Increase stack area size

In addition, note that libFLAC uses dynamic memory applications (calling system malloc(), free() functions). The heap of our project must be set large enough. Otherwise, libFLAC fails at initialization. For stack, set it to 8 kB, and for heap 430 kB (encode) and 100 kB (decode) are required. For detailed memory size, see Section 6.3.

4.6 Porting to DSP differences

Since the correctness verification is done on CM33, DSP does not need to transplant FatFs file system. Replace FatFs interface to SRAM API. For API design, see Section 5.5.

5 Performance evaluation method

To evaluate the performance of the FLAC library on RT685, we first choose any audio to judge the correctness of the FLAC porting. If the porting succeeds, diversity audio files are used to compress and decompress on the CM33 and HiFi4 cores to compare the performance differences.

Note:

The performance difference refers to the difference in time taken by the two cores to run FLAC at the maximum frequency.

5.1 Porting correctness verification

Because FLAC example default uses the file Mode, it is convenient to verify on PC after saving as a file. We put an audio file to encode,decompress it, and compare the decompressed file with the original wav file. The specific steps are as follows:

1. Put the raw.wav audio file into the SD card.
2. The FLAC preset compress level (level 0-level 8. 0 means the lowest compression ratio, that is, the largest file after compression. 8 is the highest compression ratio, that is, the smallest file after compression. For more details, see Table 4) is used to compress raw.wav respectively. The compressed file is named as level_0.flac,... level_8.flac, also written to the SD card.
3. Loop these 9 files level_0.flac... level_8.flac. Use the FLAC library to decompress to level_0.wav... level_8.wav. They are also stored in the FatFs file system of the SD card.
4. Insert the SD card into the PC and use the **Beyond Compare** software to compare *raw.wav* and nine files, *level_0.wav ... level_8.wav*. If they are consistent with the original audio, the porting was successful.

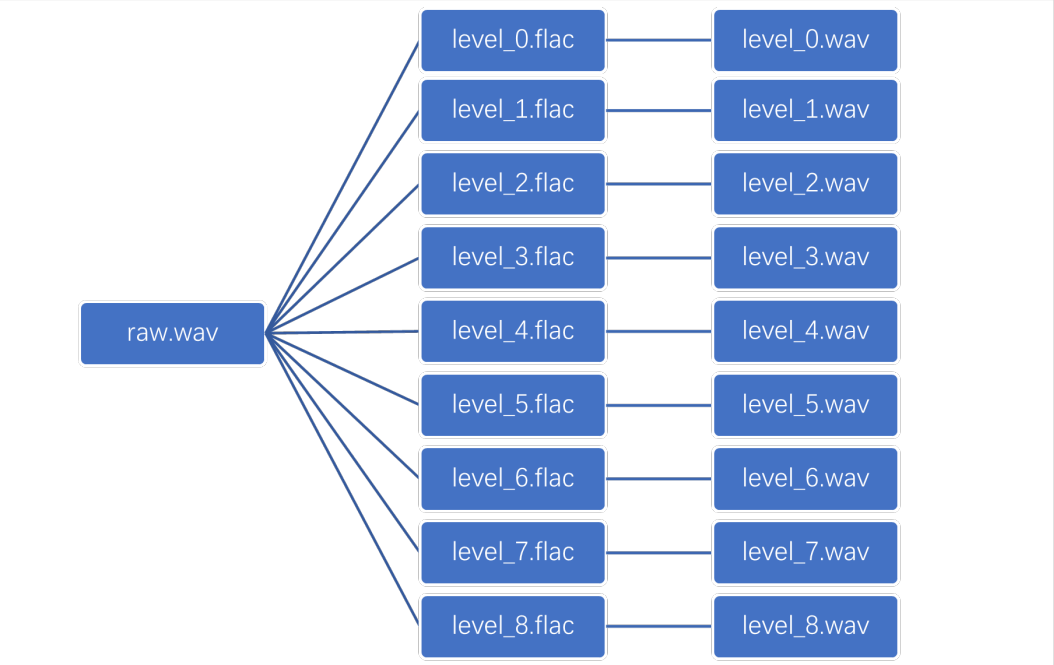

Figure 4. Migrate correctness verification logic diagrams

5.2 Test audio source selection

To fully validate and evaluate the FLAC, we use three types of audio sources:

1. Sine wave sweep
2. English speech
3. Music

At the same time, there are differences in sample rate and channel. The test audio information is as follows:

Table 3. Test audio information

No.	Raw file name	Signal	SR	CH	Resolution	Duration(s)	File size (kB)	
1	16 kHz_mono_ sine_10s.wav	Sine wave	16 kHz	mono	16 bit	10	313	Sine sweep signal. Because the test audio has 16 kHz and 48 kHz, according to the sampling law, a sinusoidal signal with a linear increase of 0 - 8 kHz is used
2	48 kHz_mono_ sine_10s.wav	Sine wave	48 kHz	mono	16 bit	10	938	
3	16 kHz_mono_ speak_10s.wav	Speak	16 kHz	Mono	16 bit	10	313	English speech. The left and right channel data of dual-channel audio in No.4 and No.6 are consistent.
4	16 kHz_stereo_ speak_10s.wav	Speak	16 kHz	Stereo	16 bit	10	626	
5	48 kHz_mono_ speak_10s.wav	Speak	48 kHz	Mono	16 bit	10	938	
6	48 kHz_stereo_ speak_10s.wav	Speak	48 kHz	Stereo	16 bit	10	1876	

Table 3. Test audio information*...continued*

No.	Raw file name	Signal	SR	CH	Resolution	Duration(s)	File size (kB)	
7	48 kHz_mono_ music_10s.wav	Music	48 kHz	Mono	16 bit	10	940	Music (Chinese song). No.8 represents similar but not identical data of two channels. No.9 represents different audio of two channels (left channel Music + right channel Sine wave).
8	48 kHz_stereo_ music_10s.wav	Music	48 kHz	Stereo	16 bit	10	1878	
9	48 kHz_stereo_ music_sine_10s.wav	Music + Sine	48 kHz	Stereo	16 bit	10	1878	

5.3 Method for evaluating compression performance

To evaluate compression performance, perform the following steps:

1. Put the six test audios into the SD card, as shown in Table 3.
2. Use nine different compression levels to compress them in turn. The encoded files are named myEncode0.flac, myEncode1.flac.... myEncode8.flac.
3. Verify that the file on PC and record the compressed file size.

Table 4 lists the nine compression levels predefined by libFLAC. For specific parameter description, see the FLAC source code.

Table 4. FLAC library predefined compression levels

Compress level	do_ mid_ side_ stereo	loose_ mid_ side_ stereo	max_ lpc_ order	qlp_ coeff_ precision	do_ qlp_ coeff_ prec_ search	do_ escape_ coding	do_ exhaustive_ model_ search	min_ residual_ partition_order	max_ residual_ partition_order	rice_ parameter_search_ dist	*apodization	Block size
0	FALSE	FALSE	0	0	FALSE	FALSE	FALSE	0	3	0	tukey (5e-1)	1152
1	TRUE	TRUE	0	0	FALSE	FALSE	FALSE	0	3	0	tukey (5e-1)	1152
2	TRUE	FALSE	0	0	FALSE	FALSE	FALSE	0	3	0	tukey (5e-1)	1152
3	FALSE	FALSE	6	0	FALSE	FALSE	FALSE	0	4	0	tukey (5e-1)	4096
4	TRUE	TRUE	8	0	FALSE	FALSE	FALSE	0	4	0	tukey (5e-1)	4096
5	TRUE	FALSE	8	0	FALSE	FALSE	FALSE	0	5	0	tukey (5e-1)	4096
6	TRUE	FALSE	8	0	FALSE	FALSE	FALSE	0	6	0	subdivide_tukey (2)	4096
7	TRUE	FALSE	12	0	FALSE	FALSE	FALSE	0	6	0	subdivide_tukey (2)	4096

Table 4. FLAC library predefined compression levels*...continued*

Compress level	do_mid_side_stereo	loose_mid_side_stereo	max_lpc_order	qlp_coeff_precision	do_qlp_coeff_prec_search	do_escape_coding	do_exhaustive_model_search	min_residual_partition_order	max_residual_partition_order	rice_parameter_search_dist	*apodization	Block size
8	TRUE	FALSE	12	0	FALSE	FALSE	FALSE	0	6	0	subdivide_tukey(3)	4096

5.4 Method for evaluating decompression performance

To evaluate decompression performance, perform the following steps:

1. Compress as described in Section 5.3.
2. Decode these nine FLAC files, myEncode0.flac, myEncode1.flac... myEncode8.flac, respectively. After decoding, they are named mydecode0.wav, mydecode1.wav... mydecode8.wav.
3. Insert the SD card into the PC to compare these nine *wav* files with the source file *xx.wav* using the HEX method of beyond compare. The files are exactly the same to prove the lossless compression of that FLAC.

5.5 Remove the time-consuming impact of reading and writing files

Since the above FLAC codec tests require reading and writing SD card files, MCU reads and writes SD cards slowly. It may affect the final test results. After the porting verification is passed, read the entire audio file to the shared SRAM in advance (for memory partition, see Section 5.7) and write the compressed/decompressed file directly to SRAM (the correctness of the file is no longer verified here and only the time required is evaluated), thereby evaluating the time required for pure FLAC codec.

The audio file is placed in the SRAM area. A set of interface areas for SRAM reading and writing is designed to replace the f_write, f_read, f_tell, f_size, and other interfaces of the previous FatFs. The interfaces of SRAM are shown as follows:

```
void virtual_fs_api_resetReadPr(void);
uint8_t virtual_fs_api_readbuf(uint8_t* data, uint32_t len);
uint8_t virtual_fs_api_copyMusic2Sram(char *inputfile);
uint8_t virtual_fs_api_readbuf_autoMovePr(uint8_t *data,
 uint32_t len);
uint8_t virtual_fs_api_seek(uint32_t absolute_byte_offset);
uint32_t virtual_fs_api_tell(void);
uint32_t virtual_fs_api_size(void);
bool virtual_fs_api_eof(void);
```

5.6 Time consuming statistical method

The accuracy of the time statistics directly affects the conclusion, so the choice of timer is important. We use OS Event Timer of RT6xx. It has a 64-bit Gray code counter, so overflow is not required to be considered. We do not need to deal with the problem of interrupt accumulation Count.

Moreover, the timer can be directly accessed by CM33 and HiFi DSP. The code of CM33 and HiFi4 is basically the same. Note that CM33 uses OSTIMER0 and HiFi4 DSP uses OSTIMER1. The core code is as follows:

```
static uint64_t gs_encodeStartTimestamp;  // starttimeStamp
static uint64_t gs_encodeEndTimestamp;    //  endtimeStamp

CLOCK_AttachClk(kHCLK_to_OSTIMER_CLK); // use HCLK 250MHz clc
// Switch clc source need reset os event timer (recount from
 zero).
*(uint32_t *)(0x40020000 + 0x40) = 0x01 << 27;  // PSCCTL0_SET
*(uint32_t *)(0x40020000 + 0x70) = 0x01 << 27;  //
 PSCCTL0_CLROSTIMER_Init(OSTIMER0);
...
gs_encodeStartTimestamp =
 OSTIMER_GetCurrentTimerValue(OSTIMER0);
EncodeDemo(inputfilename, outputfilename, iter);
gs_encodeEndTimestamp = OSTIMER_GetCurrentTimerValue(OSTIMER0);
PRINTF ("Timer consume:%.3fms\n", (gs_encodeEndTimestamp-
gs_encodeStartTimestamp)/250000.0);
```

5.7 Memory region partition

The i.MX RT6xx has a large memory of 4.5 MB, which satisfies the condition of putting the entire audio file into SRAM. Before encoding and decoding, pre-store the audio file to be compressed/decompressed to the Music area. It is at the highest address of 4.5 MB SRAM through the SDIO interface of the CM33 core.

For the FLAC test on CM33 core, only run one CM33 code. Because the CM33 core can read the contents of the SD card through the SDIO, the CM33 can be exclusive the SRAM area except for Music. For 0x0000 0000~0x0007, FFFF is the default shared reserved memory of the SDK. The modification is not required here. The final memory division of CM33 operation is as follows:

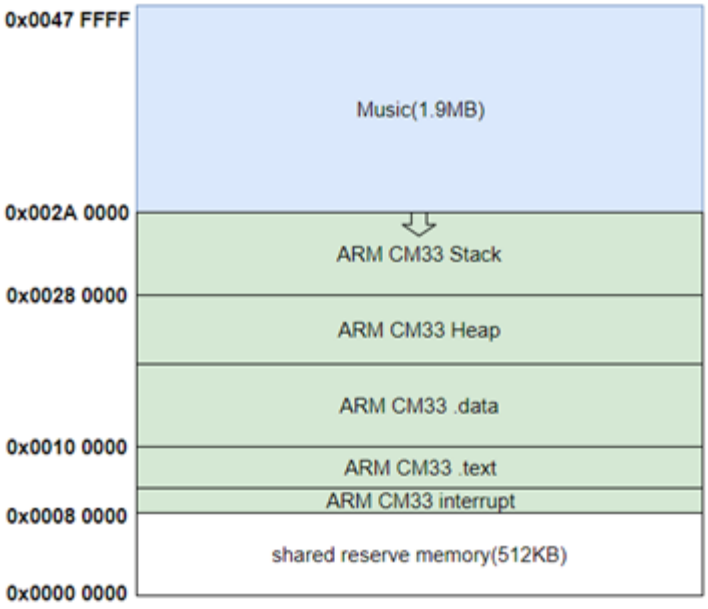

Figure 5. CM33 running a codec memory partition

For the FLAC test on HiFi 4 DSP core, since the HiFi 4 DSP cannot be started alone, it relies on CM33 to start the HiFi DSP and initialize necessary clocks, and so on. So we actually need to run two sets of code. The CM33 is responsible for reading audio files to the Music area and launching HiFi4, so the occupied space is much smaller than HiFi4. The final memory partition is as follows:

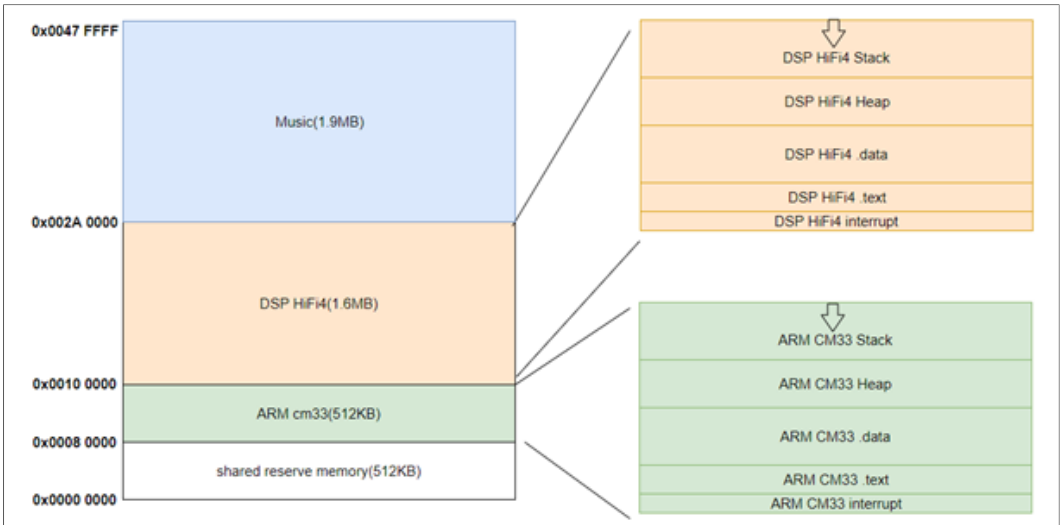

Figure 6. DSP running the codec memory partition

6 Performance results

Table 5 shows the test environment and configurations.

Table 5. Test environment table

Item	CM33	DSP
Run the clock frequency	300 MHz	600 MHz
The location of code execution	SRAM on Chip	SRAM on Chip
IDE Version	IAR for Arm 9.20.4	Xtensa Xplorer V 9.0.18
SDK Version	SDK_2.12.0	DSP Configuration V 8.0.15
IDE optimization level	-Ofast	-O3
Additional compilation options	Library -Full -FPU	-mcoproc -LNO:simd

Note:

IAR is used here as the IDE of the CM33 core. IAR is chosen because after actual testing, it is found that IAR runs fastest when compared with MCUXpresso and Keil under the same code conditions.

6.1 Compression performance

For the nine files in Section 5.2, we compress on two cores, CM33 and HiFi4, respectively. Then the following nine tables are obtained. Each table consists of nine rows excluding the title row. Each row represents a compression level. The meaning of each column is:

1. **FLAC compression level**: Range 0-8, FLAC predefined compression levels (see Table 4 for details).
2. **Source file size (kB)**: The size of the original WAV file to be compressed.
3. **After encode file size (kB)**: The compressed FLAC file size.
4. **Compression ratio**: After encode file size/source file size * 100 %.
5. **CM33 core compression time (write to the SD card version, unit s)**: The time from the beginning of compression to the completion of compression of the CM33 core (this is the version of the FLAC file written to the SD card, which is used to obtain the compressed file to verify the correctness of compression).
6. **CM33 core compression time (write to the SRAM version, unit s)**: The time from the beginning of compression to the completion of compression of the CM33 core (the version in which the FLAC file is written to the SRAM, which is used to remove the influence of writing to the SD card to accurately evaluate the compression time).
7. **Time consume of pure write data to SD card (unit s)**: calculation method = file writing to SD card version compression time - file writing SRAM version compression time.
8. **HiFi4 core compression time (write to SRAM version, unit s)**: The time from the start of compression to the completion of compression of the HiFi4 core (the version in which the FLAC file is directly written to the SRAM).
9. **HiFi4 time consuming/CM33 time consuming**: Used to compare the difference in compression time between HiFi4 and CM33.
   Calculation formula:
   *Result = HiFi4 encode time consuming / CM33 encode time consuming * 100 %*
   For example, in the first row of Table 6, result = 0.060 / 0.242 * 100 % ≈ 25 %. It means that the compression time of HiFi4 is only 1/4 of that of CM33.

Table 6. Compression performance @16kHz_sine_mono_16bit.wav

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time-write to SD card (s)	CM33 core compression time-write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time-write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	313	279	89 %	1.716	0.242	1.474	0.060	25 %
1	313	279	89 %	1.627	0.236	1.391	0.060	26 %
2	313	279	89 %	1.532	0.236	1.296	0.060	26 %
3	313	134	43 %	1.478	0.837	0.641	0.258	31 %
4	313	123	39 %	1.534	0.997	0.537	0.311	31 %
5	313	123	39 %	1.549	0.999	0.550	0.311	31 %
6	313	118	38 %	2.275	1.780	0.495	0.553	31 %
7	313	115	37 %	2.865	2.421	0.444	0.759	31 %
8	313	100	32 %	4.039	3.642	0.397	1.121	31 %

Table 7. Compression performance @48kHz_sine_mono_16bit.wav

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time- write to SD card (s)	CM33 core compression time- write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time- write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	938	642	68 %	2.933	0.592	2.341	0.172	29 %
1	938	642	68 %	3.204	0.593	2.611	0.172	29 %
2	938	642	68 %	4.036	0.594	3.442	0.172	29 %
3	938	296	32 %	3.533	2.385	1.148	0.756	32 %
4	938	296	32 %	4.257	2.850	1.407	0.909	32 %
5	938	296	32 %	4.302	2.853	1.449	0.911	32 %
6	938	271	29 %	6.156	5.182	0.974	1.640	32 %
7	938	271	29 %	8.358	7.097	1.261	2.254	32 %
8	938	242	26 %	11.938	10.668	1.270	3.333	31 %

Table 8. Compression performance @16kHz_speak_mono_16bit.wav

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time- write to SD card (s)	CM33 core compression time- write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time- write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	313	228	73 %	1.464	0.230	1.234	0.059	26 %
1	313	228	73 %	1.491	0.232	1.259	0.059	26 %
2	313	228	73 %	1.400	0.233	1.167	0.059	25 %
3	313	228	73 %	1.882	0.848	1.034	0.260	31 %
4	313	227	73 %	1.666	1.013	0.653	0.315	31 %
5	313	226	72 %	1.822	1.029	0.793	0.315	31 %
6	313	223	71 %	2.575	1.827	0.748	0.560	31 %
7	313	222	71 %	3.273	2.486	0.787	0.775	31 %
8	313	221	71 %	4.509	3.774	0.735	1.141	30 %

Table 9. Compression performance @16kHz_speak_stereo_16bit.wav

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time- write to SD card (s)	CM33 core compression time- write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time- write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	626	453	72 %	1.871	0.387	1.484	0.100	26 %
1	626	228	36 %	1.224	0.334	0.890	0.078	23 %
2	626	228	36 %	1.254	0.388	0.866	0.083	21 %
3	626	453	72 %	2.980	1.613	1.367	0.500	31 %

Table 9. Compression performance @16kHz_speak_stereo_16bit.wav*...continued*

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time- write to SD card (s)	CM33 core compression time- write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time- write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
4	626	227	36 %	2.744	1.939	0.805	0.607	31 %
5	626	226	36 %	3.486	2.725	0.761	0.855	31 %
6	626	223	36 %	5.942	5.111	0.831	1.590	31 %
7	626	222	35 %	7.867	7.114	0.753	2.219	31 %
8	626	221	35 %	11.662	10.982	0.680	3.319	30 %

Table 10. Compression performance @48kHz_speak_mono_16bit.wav

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time- write to SD card (s)	CM33 core compression time- write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time- write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	938	569	61 %	2.806	0.576	2.230	0.165	29 %
1	938	569	61 %	2.811	0.577	2.234	0.165	29 %
2	938	569	61 %	2.784	0.577	2.207	0.165	29 %
3	938	551	59 %	4.209	2.420	1.789	0.764	32 %
4	938	549	59 %	4.686	2.908	1.778	0.923	32 %
5	938	548	58 %	5.079	2.912	2.167	0.925	32 %
6	938	547	58 %	7.780	5.282	2.498	1.654	31 %
7	938	545	58 %	9.439	7.294	2.145	2.292	31 %
8	938	544	58 %	13.563	11.113	2.450	3.383	30 %

Table 11. Compression performance @48kHz_speak_stereo_16bit.wav

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time- write to SD card (s)	CM33 core compression time- write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time- write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	1876	1132	60 %	5.126	1.000	4.126	0.282	28 %
1	1876	570	30 %	3.029	0.843	2.186	0.220	26 %
2	1876	570	30 %	3.196	1.017	2.179	0.237	23 %
3	1876	1099	59 %	8.330	4.678	3.652	1.476	32 %
4	1876	550	29 %	6.121	4.174	1.947	1.327	32 %
5	1876	549	29 %	9.849	7.997	1.852	2.537	32 %
6	1876	547	29 %	17.049	15.110	1.939	4.725	31 %
7	1876	546	29 %	22.882	21.087	1.795	6.596	31 %

Table 11. Compression performance @48kHz_speak_stereo_16bit.wav*...continued*

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time-write to SD card (s)	CM33 core compression time-write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time-write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
8	1876	545	29 %	34.503	32.521	1.982	9.873	30 %

Table 12. Compression performance @48khz_mono_music_10s.wav

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time-write to SD card (s)	CM33 core compression time-write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time-write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	940	699	74 %	4.419	0.598	3.821	0.168	28 %
1	940	699	74 %	3.242	0.595	2.647	0.168	28 %
2	940	699	74 %	3.185	0.595	2.590	0.168	28 %
3	940	667	71 %	4.489	2.448	2.041	0.770	31 %
4	940	658	70 %	5.135	2.956	2.179	0.932	32 %
5	940	658	70 %	5.776	2.948	2.828	0.934	32 %
6	940	658	70 %	7.586	5.330	2.256	1.665	31 %
7	940	648	69 %	9.393	7.357	2.036	2.312	31 %
8	940	648	69 %	14.396	11.233	3.163	3.410	30 %

Table 13. Compression performance @48khz_stereo_music_10s.wav

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time-write to SD card (s)	CM33 core compression time-write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time-write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	1878	1411	75 %	7.925	1.029	6.896	0.289	28 %
1	1878	1387	74 %	6.877	1.100	5.777	0.286	26 %
2	1878	1381	74 %	5.993	1.272	4.721	0.308	24 %
3	1878	1351	72 %	9.105	4.724	4.381	1.489	32 %
4	1878	1308	70 %	11.754	6.743	5.011	2.123	31 %
5	1878	1307	70 %	14.731	10.588	4.143	3.344	32 %
6	1878	1306	70 %	24.373	20.131	4.242	6.270	31 %
7	1878	1288	69 %	32.455	28.198	4.257	8.801	31 %
8	1878	1288	69 %	47.913	43.703	4.210	13.192	30 %

Table 14. Compression performance @48khz_stereo_music_sine_10s.wav

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time-write to SD card (s)	CM33 core compression time-write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time-write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	1878	1023	54 %	5.585	1.011	4.574	0.279	28 %
1	1878	1023	54 %	4.524	1.018	3.506	0.256	25 %
2	1878	1023	54 %	4.680	1.213	3.467	0.280	23 %
3	1878	912	49 %	8.972	4.660	4.312	1.472	32 %
4	1878	905	48 %	10.673	6.636	4.037	2.078	31 %
5	1878	905	48 %	13.314	10.471	2.843	3.301	32 %
6	1878	887	47 %	23.087	19.970	3.117	6.224	31 %
7	1878	877	47 %	31.792	27.891	3.901	8.724	31 %
8	1878	858	46 %	47.353	43.080	4.273	13.099	30 %

From Table 6 to Table 14, the following conclusions can be drawn:

About FLAC compression:

1. As seen in Table 6 to Table 14, we can clearly see that if gradually increase the compression level, the compression ratio gradually increases (the value decreases) and the compression time gradually increases.
2. As seen in Table 6 to Table 14, by comparing *CM33 core compression time (write to SD card version)* and *CM33 core compression time (write to SRAM version)*, we can see that the time taken varies widely, which also validates our conjecture. If accurate evaluation of FLAC encode is required, it is necessary to remove the influence of reading and writing SD. At the same time, we also found that the time consumption for CM33 to write to the SD card is unstable. Even for files of the same size (such as level 0~2 in Table 7), there is a difference of several hundred milliseconds. It is preliminarily judged that it is caused by the storage mechanism of the SD card file system (ECC, read and write balance).
3. As seen in Table 6 and Table 7, the sample rate of the audios to be compressed have been increased from 16 kHz to 48 kHz. Compared with the CM33/HiFi4, encode time almost increases by three times. Therefore, it is proved that the increase in sample rate is linearly positively correlated with FLAC Encode time.
4. As shown in Table 6, Table 8, Table 7, Table 10, and Table 12, it can be concluded that only audio content differs (such as sine wave, speech, and Music. All are mono). The encoding time in the same level is almost the same (Compression time difference is less than 5 %). But the compression ratio is different. Obviously the compression ratio of the sine wave is better than the speech. Because the sine wave signal is easier to be predicted and get smaller residuals, the final Rice-encoded value is smaller.
5. As shown in Table 8 to Table 11, it can be concluded that for the same audio only the difference between mono and stereo, the time-consuming of FLAC encode is not twice. The higher the compression rate of stereo, the more Complex Linear Predictive Coding (LPC) will be used. The stereo audio is more time-consuming and more than twice as long as mono audio.

6. As shown in Table 11 and Table 13, for stereo audio, two channels with different data take more time than two channels with the same audio. The time-consumption increase is about 30 %.
7. As shown in Table 13 and Table 14, for stereo audio with different data in the two channels, only the audio content is different. The final compression time consumption is almost same. It is consistent with 4.
8. Through the compression ratio of level 3 in Table 9 and Table 11, there are both abnormal performance. The compression ratio is not as good as the lower compression level 1 and 2. By comparing Table 4, it can be found out that the *do_mid_side_stereo* is disabled in level 3. So enable this option when encode stereo audios.

The performance of RT685 can be summarized:

1. As shown in Table 6 to Table 14, We can see that the encode performance of HiFi4 is better than that of CM33 (HiFi4 runs at 600 MHz, compared with CM33 runs at 300 MHz). The performance increases more than twice.
2. For the CM33 core runs at 300 MHz:
   a. The low-quality audio, such as 16kHz_mono_16bit (Table 6 and Table 8): All compression levels (level 0-8) can achieve real-time compression (here, 1 s of original audio can be compressed within 1 s, that is, it is considered to be real-time compression).
   b. The 48 kHz_mono_16bit and 16kHz_stereo_16bit audio (Table 7, Table 9, Table 10, and Table 12): Except for the highest level 8 cannot achieve real-time compression, other compression levels lower than 8 can be satisfied.
   c. The high-quality audio, such as 48kHz_stereo_16bit (Table 11, Table 13, and Table 14): Compression ratios of level 6 and above cannot be compressed in real-time.
3. For the HiFi4 DSP core runs at 600 MHz.
   - Except for the highest compression level of 48kHz_stereo_16bit which stereo is not same is slightly difficult. It can achieve real-time all compression levels of compression rate for other audio. The HiFi4 DSP is higher efficiency in processing audio data than CM33. The time consumption is only 25 % - 30 % of CM33.

6.2 Decompression performance

For the nine files in Section 5.2, after be compressed in Section 6.1, we can get 9*9=81 FLAC files. We decompress them on two cores, CM33 and HiFi4, respectively, and the following nine tables are obtained. Each table consists of nine rows excluding the title row. Each row represents a compression level. The meaning of each column is:

1. **Compression level**: Range 0-8, FLAC predefined compression level.
2. **FLAC source file size (kB)**: It is the FLAC file size which is compressed in corresponding compression level.
3. **After decode file size (kB)**: It is the size of the decompressed WAV file.
4. **Consistent with the original .wav file**: Correspond the extracted WAV file to the original test file in Section 5.2. If it is consistent, it is Yes. Otherwise, No.
5. **CM33 core decode time (write to the SD card version, unit s)**: The time from the start of decompression to the completion of decompression of the CM33 core. It is used to obtain the decompressed file to verify the correctness of compression.
6. **CM33 core decode time (write to the SRAM version, unit s)**: The time from the start of decompression to the completion of decompression of the CM33 core. It is

used to remove the influence of writing the SD card to evaluate the decompression time.

7. **Time consume of pure write data to SD card (unit s)**: calculation method = CM33 core decompression time (write to SD card version) - CM33 core decompression time (file write to SRAM version).
8. **HiFi4 core decode time (write to SRAM version, unit s)**: The time from the start of decompression to the completion of decompression of the HiFi4 core.
9. **HiFi4 time consuming/CM33 time consuming**: Used to compare the difference in compression time between HiFi4 and CM33.
   Calculation formula:
   Result = HiFi4 decode time consuming / CM33 decode time consuming * 100 %.
   For example, in the first row of the following table, result = 0.025 / 0.099 * 100 % ≈ 25 %. It means that the decompression time of HiFi4 is only 1/4 of that of CM33.

Table 15. Decompression performance @16kHz_mono_sine_16bit.wav

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original. wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	279	313	Yes	2.144	0.099	2.045	0.025	25 %
1	279	313	Yes	2.084	0.092	1.992	0.025	27 %
2	279	313	Yes	1.876	0.095	1.781	0.025	26 %
3	134	313	Yes	1.762	0.094	1.668	0.025	26 %
4	123	313	Yes	1.765	0.101	1.665	0.027	27 %
5	123	313	Yes	1.776	0.096	1.680	0.027	28 %
6	118	313	Yes	1.767	0.095	1.672	0.025	26 %
7	115	313	Yes	1.767	0.096	1.671	0.027	28 %
8	100	313	Yes	1.866	0.098	1.768	0.026	26 %

Table 16. Decompression performance @48kHz_mono_sine_16bit.wav

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original. wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	642	938	Yes	6.635	0.247	6.388	0.066	27 %
1	642	938	Yes	7.126	0.250	6.876	0.066	27 %
2	642	938	Yes	5.764	0.247	5.517	0.066	27 %
3	296	938	Yes	5.485	0.241	5.244	0.059	25 %
4	296	938	Yes	5.588	0.238	5.350	0.059	25 %
5	296	938	Yes	5.574	0.241	5.333	0.059	25 %
6	271	938	Yes	5.677	0.261	5.416	0.059	23 %

Table 16. Decompression performance @48kHz_mono_sine_16bit.wav*...continued*

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original. wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
7	271	938	Yes	5.599	0.241	5.358	0.059	25 %
8	242	938	Yes	5.397	0.239	5.158	0.059	25 %

Table 17. Decompression performance @16kHz_mono_speak_16bit.wav

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original. wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	228	313	Yes	1.782	0.094	1.688	0.025	26 %
1	228	313	Yes	1.776	0.091	1.685	0.025	27 %
2	228	313	Yes	1.797	0.094	1.703	0.025	26 %
3	228	313	Yes	1.961	0.096	1.865	0.026	27 %
4	227	313	Yes	1.991	0.104	1.887	0.028	27 %
5	226	313	Yes	2.075	0.102	1.973	0.028	27 %
6	223	313	Yes	2.160	0.106	2.054	0.029	27 %
7	222	313	Yes	2.126	0.113	2.013	0.037	32 %
8	221	313	Yes	1.941	0.116	1.825	0.036	31 %

Table 18. Decompression performance @16kHz_stereo_speak_16bit.wav

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original. wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	453	626	Yes	3.790	0.163	3.627	0.043	26 %
1	228	626	Yes	4.432	0.137	4.295	0.039	28 %
2	228	626	Yes	4.298	0.139	4.159	0.037	26 %
3	453	626	Yes	3.738	0.176	3.562	0.045	25 %
4	227	626	Yes	3.691	0.144	3.547	0.043	30 %
5	226	626	Yes	4.306	0.145	4.161	0.041	28 %
6	223	626	Yes	3.570	0.143	3.427	0.041	28 %
7	222	626	Yes	3.487	0.156	3.331	0.049	31 %
8	221	626	Yes	3.966	0.153	3.813	0.048	31 %

Table 19. Decompression performance @48kHz_mono_speech_10s.wav

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original .wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	569	938	Yes	5.613	0.245	5.369	0.061	25 %
1	569	938	Yes	5.766	0.242	5.524	0.061	25 %
2	569	938	Yes	5.642	0.245	5.397	0.061	25 %
3	551	938	Yes	5.873	0.257	5.616	0.064	25 %
4	549	938	Yes	5.944	0.273	5.671	0.070	26 %
5	548	938	Yes	5.571	0.271	5.300	0.070	26 %
6	547	938	Yes	5.280	0.276	5.004	0.071	26 %
7	545	938	Yes	5.772	0.322	5.450	0.091	28 %
8	544	938	Yes	6.003	0.300	5.703	0.090	30 %

Table 20. Decompression performance @48kHz_stereo_speak_16bit.wav

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original. wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	1132	1876	Yes	11.240	0.451	10.789	0.115	25 %
1	570	1876	Yes	11.506	0.371	11.136	0.104	28 %
2	570	1876	Yes	11.406	0.360	11.046	0.098	27 %
3	1099	1876	Yes	12.766	0.493	12.273	0.124	25 %
4	550	1876	Yes	11.800	0.396	11.404	0.114	29 %
5	549	1876	Yes	11.256	0.392	10.864	0.108	28 %
6	547	1876	Yes	11.692	0.391	11.301	0.108	28 %
7	546	1876	Yes	10.812	0.420	10.392	0.128	30 %
8	545	1876	Yes	11.526	0.419	11.107	0.128	30 %

Table 21. Decompression performance @48khz_mono_music_10s.wav

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original .wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	699	940	Yes	5.860	0.253	5.607	0.061	24 %
1	699	940	Yes	6.318	0.251	6.067	0.061	25 %

Table 21. Decompression performance @48khz_mono_music_10s.wav*...continued*

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original .wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
2	699	940	Yes	5.564	0.275	5.289	0.061	22 %
3	667	940	Yes	5.271	0.275	4.997	0.068	25 %
4	658	940	Yes	5.298	0.292	5.006	0.076	26 %
5	658	940	Yes	5.325	0.290	5.035	0.076	26 %
6	658	940	Yes	5.542	0.293	5.250	0.076	26 %
7	648	940	Yes	5.318	0.319	4.999	0.104	33 %
8	648	940	Yes	5.318	0.322	4.997	0.104	32 %

Table 22. Decompression performance @48khz_stereo_music_10s.wav

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original. wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	1411	1878	Yes	10.611	0.463	10.148	0.116	25 %
1	1387	1878	Yes	10.387	0.498	9.889	0.136	27 %
2	1381	1878	Yes	10.451	0.495	9.956	0.135	27 %
3	1351	1878	Yes	10.420	0.517	9.903	0.131	25 %
4	1308	1878	Yes	10.482	0.580	9.902	0.167	29 %
5	1307	1878	Yes	10.489	0.583	9.906	0.167	29 %
6	1306	1878	Yes	10.468	0.582	9.886	0.167	29 %
7	1288	1878	Yes	11.308	0.644	10.664	0.225	35 %
8	1288	1878	Yes	10.934	0.642	10.293	0.223	35 %

Table 23. Decompression performance @48khz_stereo_music_sine_10s.wav

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original. wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	1023	1878	Yes	10.552	0.452	10.100	0.114	25 %
1	1023	1878	Yes	10.357	0.475	9.882	0.114	24 %
2	1023	1878	Yes	10.365	0.453	9.913	0.114	25 %
3	912	1878	Yes	10.367	0.481	9.887	0.121	25 %

Table 23. Decompression performance @48khz_stereo_music_sine_10s.wav*...continued*

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original. wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
4	905	1878	Yes	10.441	0.499	9.942	0.130	26 %
5	905	1878	Yes	10.390	0.500	9.890	0.130	26 %
6	887	1878	Yes	10.489	0.495	9.994	0.128	26 %
7	877	1878	Yes	10.416	0.529	9.887	0.157	30 %
8	858	1878	Yes	10.472	0.531	9.941	0.157	30 %

As shown in Table 15 to Table 23, the following conclusions can be drawn:

About FLAC decompression:

1. We can see that the decompression time of compressed audio from the same file at different compression levels is similar the same.
2. Compare with the encode, it can be seen that the FLAC compression level only has a large impact on the encoding time, but has little effect on the decoding. The higher the compression level is, the more time it takes to find the best compression algorithm (higher order). The decoder only needs to directly decompress according to the given compression algorithm parameters.
3. FLAC algorithm is designed to facilitate decompression. This feature facilitates the implementation of decoders on different grades of hardware and is helpful to the promotion of FLAC.

The performance of RT685 can be summarized as follows:

1. As shown in Table 15 to Table 23, we can clearly see that the decode performance of HiFi4 is significantly better than that of CM33 (HiFi4 runs at 600 MHz, compared with CM33 runs at 300 MHz) and the performance increases more than twice.
2. As shown in Table 15 to Table 23, we can see that for the CM33 runs at 300 MHz and HiFi4 DSP runs at 600 MHz. All the FLAC audio can be decompressed within 0.6 s. The decompression time of HiFi4 is 1/3~1/4 of that of CM33.

6.3 FLAC memory resource consumption

The following evaluation of FLAC codec consumption of MCU resources is based on IAR (see Table 5 for detailed configurations). Table 24 shows the results. We can see after the FLAC compression is greater than level 2, the Heap consumption increases a lot due to the linear prediction algorithm (LPC).

Table 24. Resource consumption table

Category	ENCODER		DECODER
Compress level	0-2	3-8	0-8
RO TEXT	95’578 bytes		35’638 bytes
RO DATA	14’832 bytes		8’328 bytes
R/W DATA	556 bytes		552 bytes
Heap	158’576 bytes	431’360 bytes	99’784 bytes

Table 24. Resource consumption table*...continued*

Category	ENCODER		DECODER
Compress level	0-2	3-8	0-8
Stack	6’232 bytes		5’272 bytes

7 GNU profiler analysis

7.1 Introduction to GNU profiler

Profiler is a tool of GNU. It allows you to learn where your program spent its time. It can easily help us find the most time-consuming function or unexpected function calling relationships. It helps us debug and locate problems.

Tensilica Xtensa supports hardware profiling method to collect profile information. The theory is to run the program on the hardware platform, then periodically sample through an Xtensa timer. Finally, the debugger sends the generated profiling data to the PC through the OCD Deamon tool (depending on the **libgdbio** library. Therefore, the gdbio LSP must be used to link the executable file). To perform a hardware profiling analysis of Tensilica, the following three steps are required:

- Compiler and linker add configurations to enable profilling
- Add the necessary profile code and run
- Parse raw gmon.out to readable file with gprof

For more details, see *GNU Profiler User's Guide.pdf* in the Xtensa software installation directory.

7.2 Profiling file analysis

Through the three steps in Section 7.1, we can get the following parsed *profile.txt*. It includes Flat profile, Call Graph, annotated source, and other parts.

In the *Flat profile*, each function is sorted in decreasing order of time-consuming, with the first being the most time-consuming. So we can see that for the FLAC encoder program, the `FLAC__lpc_compute_autocorrelation()` function is the most time-consuming. Then, we have a clear direction for the nextstep optimization. For example, it is a good idea to put time-consuming functions into the ITCM region of HiFi4 or rewrite these functions using HiFi4's special instruction set.

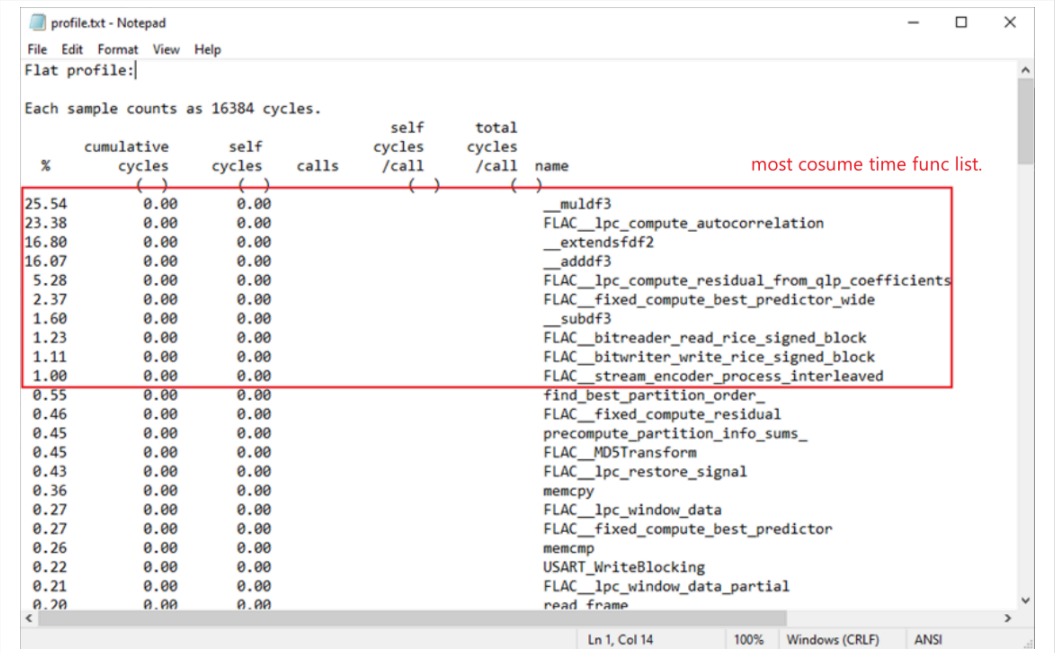

Figure 7. Screenshot of the profile *Flat profile* section

8 Conclusion

1. The open source library FLAC is proved to be lossless codec.
2. For FLAC decompression, both CM33 and HiFi4 DSP cores of RT685 can easily competent.
3. For FLAC compression, both CM33 and HiFi4 DSP cores of RT685 can achieve full-level real-time compression for most audio. They can achieve a slightly lower level of real-time compression for a small part of high-quality audio.
4. The encoding and decoding time of i.MX RT685 HiFi 4 DSP@600MHz is only about 25-30 % of that of CM33@300MHz.
5. Compared with decompression, the different compression levels of compression have a great impact on the running time. When using the project, it is necessary to select the optimal compression level to balance time and space.

9 References

1. FLAC website
2. GNU gprof
3. GNU Profiler User's Guide Version 2.34
4. MIMXRT600-EVK Schematic (Rev E2)
5. *RT600 User Manual* (document UM11147)

10 Revision history

Revision number	Date	Substantive changes
0	30 November 2022	Initial release

11 Legal information

11.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

11.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

11.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents

For more information, please visit: http://www.nxp.com

Date of release: 30 November 2022
Document identifier: AN13793