08-6326-RN-ZCH66
SEPTEMBER 17, 2007
3.4

Release Notes for aacPlus Decoder on ARM ELINUX

ABSTRACT:

Release Notes for aacPlus Decoder on ARM11 ELINUX

KEYWORDS:

Multimedia codecs, AAC

APPROVED:

Shang Shidong

Revision History

VERSION	DATE	AUTHOR	CHANGE DESCRIPTION
0.1	07-Jul-2005	Ganesh Kumar	Initial Draft
1.0	08-Jul-2005	Ashok Kumar	Added Review Comments
1.1	01-Sep-2005	C Ganesh Kumar	Document updated for Parametric stereo
2.0	02-Sep-2005	Ashok Kumar	Review and rework
2.1	28-Nov-2005	Anand	Updated makefile changes and build procedure changes for RVDS 2.2
3.0	06-Feb-2006	Lauren Post	Draft version using new format
3.1	14-Mar-06	Gaurav Goel	Support of LP-SBR and HQ-SBR Optimizations has been added
3.2	22-Mar-2006	Surendra Jain	Updated Known Problems
3.3	13-Oct-2006	ShyamKrishnan M	Raw bit stream support
3.4	17-Sep-2007	Bing Song	Update for 4 libraries to 1 library
4.0	28-Apr-09	Lyon Wang	Use AAC LC lib for AAC LC part decoder
4.1	15-May-09	Lyon Wang	Add ARM9 support

Table of Contents

Introduction

1.1 Purpose

The purpose of this document is to provide information on the package contents, instructions on building library and test applications and test execution on ARM11 ELINUX, RVDS and Linux x86.

1.2 Scope

The scope is restricted to information on the package contents and instructions for building and testing. This document does not provide architecture or details about the APIs provided in the package. Performance data will be provided in another document as detailed in the Requirements Book.

1.3 Audience Description

The reader is expected to have basic understanding of Audio Signal processing and aacPlus decoding.

1.4 References

1.4.1 Standards

- ISO/IEC 13818-7:1997 Information technology -- Generic coding of moving pictures and associated audio information -- Part 7 (popularly known as *MPEG-2 AAC*)
- ISO/IEC 13818-4:1997 Information technology -- Generic coding of moving pictures and associated audio information -- Part 4 (compliance testing)
- ISO/IEC 14496-3:2001(E) Information technology -- Coding of audio-visual objects -- Part 3: Audio.
- ISO/IEC 14496-4:2000, Information technology -- Coding of audio-visual objects -- Part 4: Conformance testing.
- ISO/IEC 14496-3:2001 Amendment 1 -- WD Text for Backward Compatible Bandwidth Extension for General Audio Coding.
- Coding Technology aacPlus Decoder Certification Document, Version 2.3

1.4.2 General References

- Ted Painter and Andreas Spanias, “Perceptual Coding of Digital Audio”, Proc. IEEE, vol-88, no.4, April 2000
- H.S.Malvar, “Lapped transforms for efficient subband/transform coding”, IEEE trans. ASSP, June 1990.

- Seymour Shlien, “The Modulated Lapped Transform, Its Time-Varying Forms and Its Applications to Audio Coding Standards.”
- “A Tutorial on MPEG/Audio compression” by Davis Pan
- Martin Wolters, Kristofer Kjorling2, Daniel Homm and Heiko Purnhagen, “A closer look into MPEG-4 High Efficiency AAC”, AES 115th Convention, 2003.

1.4.3 Freescale Multimedia References

- AAC Plus Enhanced Application Programming Interface - aacplus_dec_api.doc
- AAC Plus Enhanced Requirements Book - aacplus_dec_reqb.doc
- AAC Plus Enhanced Test Plan - aacplus_dec_test_plan.doc
- AAC Plus Enhanced Release notes - aacplus_dec_release_notes.doc
- AAC Plus Enhanced Test Results – aacplus_dec_test_results.doc
- AAC Plus Enhanced Performance Results – aacplus_dec_perf_results.doc
- AAC Decoder API doc Appendix C on Header data extraction – aac_dec_api.doc
- AAC Plus Enhanced Interface Header – aacplus_dec_interface.h
- AAC Plus Enhanced Application Code - aacplus_main.c

1.5 Definitions, Acronyms, and Abbreviations

TERM/ACRONYM	DEFINITION
AAC	Advanced Audio Coding
aacPlus	AAC low complexity plus SBR decoder
aacPlus V2	AAC low complexity plus SBR plus parametric stereo decoder
ADIF	Audio_Data_Interchange_Format
ADTS	Audio_Data_Transport_Stream
API	Application Programming Interface
ARM	Advanced RISC Machine
FSL	Freescale
HE-AAC	High Efficiency AAC
IEC	International Electro-technical Commission
ISO	International Organization for Standardization
LC	Low Complexity
MDCT	Modified Discrete Cosine Transform
MPEG	Moving Pictures Expert Group
PCM	Pulse Code Modulation
PNS	Perceptual Noise Substitution
PS	Parametric Stereo
SBR	Spectral Band Replication

RVDS	ARM RealView Development Suite
UNIX	Linux PC x/86 C-reference binaries
TBD	To be decided

1.6 Document Location

/fsl_mad_multimedia_codec/docs/aacplus_dec

2 Release History

RELEASE NUMBER	DELIVERABLES	FEATURES
2.0	• Documentation • Application Interface Header file for the decoder (aacplus_dec_interface.h) • Example Application (aacplus_main.c, aacplus_portio.c and aacplus_aux.c) • RVDS generated library to be linked using RVDS tools (libaac_dec_arm11_RVDS.a, libsbr_dec_arm11_RVDS.a, • libffr_bit_arm11_RVDS.a, libffr_lib_arm11_RVDS.a) • RVDS generated library to be linked using gnu tool set for the board (libaac_dec_arm11_ELINUX.a, libsbr_dec_arm11_ELINUX.a, libffr_bit_arm11_ELINUX.a, libffr_lib_arm11_ELINUX.a) • Makefiles to build library and test application for RVDS/ELINUX/UNIX • Source code of decoder • Test vectors	• Initial Release
2.1	Same	• Shared Library Support • Upgrade to Latest Release • Upgrade to RVDS 2.2
2.3	Same	• Stereo support
2.4	Same	• Support for Low Power SBR • Optimized High Quality SBR with PS
2.5	Same	• Raw bit stream support

2.6	• Documentation • Application Interface Header file for the decoder (aacplus_dec_interface.h) • Example Application (aacplus_main.c, aacplus_portio.c and aacplus_aux.c) • RVDS generated library to be linked using RVDS tools (lib_aacplus_dec_arm11_lervds.a) • RVDS generated library to be linked using gnu tool set for the board (lib_aacplus_dec_arm11_elinux.a) • Makefiles to build library and test application for RVDS/ELINUX/UNIX • Source code of decoder • Test vectors	• Just one library
3.00.00	• Use AAC LC lib for AAC LC part decoder for AAC Plus	• BLN_MAD-MMCODECS_AACPD_ARM_03.00.00 • Should use with BLN_MAD-MMCODECS_AACD_ARM_03.00.00

Table 1. Details of the release

2.1 Assumptions and Known Problems

- Unsupported relocation warning gets generated while cross linking of RVDS generate decoder library with sample test application using GNU tool set.
- Test vectors al17_* do not pass for adts inputs
- PS Conformance fails when we include assembly function “applyRotation_asm_1”. Hence for this release, equivalent C code is included. Once the issue with the assembly code is resolved, the PS performance figures are expected to improve by 5-8%.

2.2 Contacts

Please report any problems to Freescale customer representative.

3 List of Deliverables

3.1 Documentation

Base directory: /fsl_mad_multimedia_codec/

Subdirectory	Files
docs/aacplus_dec	aacplus_dec_api.doc aacplus_dec_reqb.doc aacplus_dec_test_plan.doc aacplus_dec_test_results.doc aacplus_dec_perf_results.doc aacplus_dec _release_notes.doc

3.2 Public Headers

Base directory: /fsl_mad_multimedia_codec/

Subdirectory	File
ghdr	aacplus_dec_interface.h
	aacd_dec_interface.h

3.3 Test Application Source

Base directory: /fsl_mad_multimedia_codec/

Subdirectory	Files
test/aacplus_dec	“Makefile” makefile for building RVDS, UNIX and ELINUX board executables.
test/aacplus_dec/c_src	*.c, application code.
utils/aacplus_dec	scripts for running batch processing

3.4 Library Source

Base directory: /fsl_mad_multimedia_codec/

Subdirectory	Files
src/aacplus_dec	Makefile “Makefile” for building RVDS, UNIX, and ELINUX libraries. lib_aacplus_dec_arm11_RVDS.a – Special options for simulator testing for High Quality SBR lib_aacplus_dec_arm11_ELINUX.a - static library for board lib_aacplus_dec_arm11_ELINUX.so – shared library for board for High Quality Mode lib_aacplus_dec_UNIX.a – library for Linux x/86 – c reference code
src/aacplus_dec/FFR_bitbuf	aacPlus bit buffer management source code

lib	
src/aacplus_dec/FFR_sbrdeclib	Contains the SBR 'C' source codes(Formally a C++ code converted to 'C' But the file names retain CPP extension). This library contains the code for High Quality (HQ) SBR.
src/aacplus_dec/FFRlib	Contains the 'C' library files used by SBR (Formally a C++ code converted to C. But the file names retain CPP extension)
src/aac_dec	Optimized AAC decoder source code
src/aacplus_dec/c_source	*.c, AACPlus decoder source code
src/aacplus_dec/asm_source	*.s AACPlus assembler source
src/aacplus_dec/include	*.h, AACPlus decoder library header files

3.5 Common Makefiles

Base Directory: /fsl_mad_multimedia_codec/

Subdirectory	**Files**
build/Makefile.init	This is a common makefile included in the codec library makefile for building the libraries. This file includes common options used by all codecs. Following flags can be overwritten or added to in the codec library makefile 1. Path to toolchain tools (TC_ROOT) 2. GNU header file path (HEADER_PATHS) 3. GNU library path (LIB_PATHS) 4. GNU Compiler/Assembler Options (GNU_CFLAGS, GNU_AFLAGS) 5. Endian Flags 6. Optimization Flags(OPTIM_LEVEL, OPTIM_TYPE) 7. Common options for RVDS,UNIX and ELINUX (CFLAGS,AFLAGS) 8. Build specific flags 9. Source directory of 'C' code 10. Source directory of 'assembly(.s)' code 11. Object directory for .o files 12. RVDS Compilation Tools 13. Codec header path 14. Arguments for librarian for UNIX builds 15. SHARED_ELINUX builds for libraries that must be linked using the toolchain because of external library includes.

build/ Makefile_test.init	This is the common makefile included in the codec test makefile for building the test application. This file includes the common options used by the all the codecs. Following flags can be overwritten or added to in the codec test makefile 1. Toolchain path depending on the build option 2. Compiler Flags 3. Linker flags 4. Paths for c_source, exe and object directories 5. Codec header files' INCLUDES path 6. Endian Flags 7. CODEC_LIB generation

3.6 Test Vectors

Base Directory: multimedia_vectors/test_vectors

The test vectors are provided in another location from the library and test source.

Subdirectory	**Files**
aacplus_dec/testvectors/mpeg_aac_conformance	*.adif, *.adts *.mp4 – AAC test vectors
aacplus_dec/testvectors/mpeg_sbr_conformance	*.adif, *.adts *.mp4 – SBR test vectors

4 Software Setup & Tools used

- ARM RVDS 3.0 (build 441) should be installed in the PC.
- Freescale Linux OS Release L26.1.17 must be running on the evaluation board.
- Intel based Red Hat Linux Machine must have the **devtek** toolchain installed on it.
  - **devtek Toolchain** `gcc 4.1.1 glibc 2.4 nptl 6`
- 'Cygwin' **Version** CYGWIN_NT-5.1, a freely downloadable linux emulator is installed in PC - **http://www.cygwin.com/**.
- 'make' utility available for targeted platforms

5 Build Procedure

All the required makefiles are provided under individual directories. The library can be built for windows / target processor (ARM1136J-S). The details for the build procedure are described below.

5.1 Library for ELINUX and RVDS

To build the ELINUX and RVDS libraries, run 'make' on 'Makefile' from library directory. The makefile shall create the required directory to hold the object files. The makefile can be used if you want to build the library only. The same makefile can used to build libraries for both board, Unix/Linux and RVDS with different build options. The following options are available to build the library. By Default High Quality SBR libraries and Test Application will be build.

Options

a) **BUILD options**:

a. **BUILD= ARM11ELINUX** : This is the default option and builds both static libraries 'lib_aacplus_dec_arm11_elinux.a'' , for testing on the board.
BUILD= ARM9ELINUX : This is the default option and builds both static libraries 'lib_aacplus_dec_arm9_elinux.a'' , for testing on the board

b. **BUILD= ARM11ELINUXDLIB:** This is for creating . shared library 'lib_aacplus_dec_arm11_elinux.so' which is linked by gcc linker, should run after previous command
BUILD= ARM9ELINUXDLIB: This is for creating . shared library 'lib_aacplus_dec_arm9_elinux.so' which is linked by gcc linker, should run after previous command

c. **BUILD= ARM11LERVDS**: This option builds the static libraries'lib_aacplus_dec_arm11_lervds.a ' for testing on RVDS (Armulator).
BUILD= ARM9LERVDS: This option builds the static libraries'lib_aacplus_dec_arm9_lervds.a ' for testing on RVDS (Armulator).

Eg: make BUILD= ARM11ELINUX
make BUILD= ARM9ELINUX
make BUILD= ARM11LERVDS
make BUILD= ARM9LERVDS

b) **ENDIAN options for RVDS**:

- **TARGET_ENDIAN=LITTLE**: This is the default option and sets the endian-ness to 'little'

c) **clean options**:

- **clean**: Deletes all the object files and RVDS,UNIX and ELINUX libraries.

Note: Make appropriate changes in file 'Makefile.init' at directory '/fsl_mad_multimedia_codec/*build/*' for the location of toolchains.

The libraries that are built is saved as 'lib_aacplus_dec_arm11_lervds.a' for RVDS build and 'lib_aacplus_dec_arm11_elinux.a' for the board build. These libraries are saved in the current directory (the same directory in which the source and assembly directories are listed).

Target	Compilation Environment	Build Options	Library Name
Board	PC (Using Cygwin)	BUILD= ARM11ELINUX	lib_aacplus_dec_arm11_elinux.a
		BUILD= ARM11ELINUXDLIB	lib_aacplus_dec_arm11_elinux.so
RVDS	PC (Using Cygwin)	BUILD= ARM11LERVDS	lib_aacplus_dec_arm11_lervds.a

5.2 Library for UNIX

To build the UNIX library 'lib_aacplus_dec_x86_unix.a', run 'make BUILD=UNIX' from library directory. The makefile shall create the required directory to hold the object files. The makefile can be used if you want to build the library only.

Target	Compilation Environment	Build Options	Library Name
Unix/ Linux	Unix/Linux machine	BUILD=UNIX	lib_aacplus_dec_x86_unix.a

5.3 Test Application for ELINUX and RVDS

To build the test application, run 'make' on 'Makefile' from the test directory. This makefile can create executables for testing on both board and RVDS for ARM11. The executables test_aacplus_dec_arm11_lervds for RVDS and test_aacplus_dec_arm11_elinux for board are stored under test/exe directory. The makefile shall create the required directory structure to hold the object files and executables. The following commands should be invoked so as to build the executables.

Options

1) **BUILD options**:
   - **BUILD=ARM11ELINUX**: This is the default option and builds the executable 'test_aacplus_dec_arm11_elinux ', for the board.
     BUILD=ARM9ELINUX: This is the default option and builds the executable 'test_aacplus_dec_arm11_elinux ', for the board
   - **BUILD=ARM11LERVDS**: This option builds the executable 'test_aacplus_dec_arm11_lervds' for the RVDS (Armulator).
     BUILD=ARM9LERVDS: This option builds the executable 'test_aacplus_dec_arm9_lervds' for the RVDS (Armulator).
     Eg: make BUILD=ARM11ELINUX (for board)

make BUILD=ARM11LERVDS (for Armulator)

2) **ENDIAN options for RVDS**:
   - **TARGET_ENDIAN=LITTLE**: This is the default option and sets the endian-ness to 'little'

3) **LIBRARY options**:
   - **LIB_TYPE= STATIC**: This option builds the ELINUX test application linked with the ELINUX static libraries 'lib_aacplus_dec_arm11_elinux.a'. If nothing is specified, the executable links with shared library 'lib_aacplus_dec_arm11_elinux.so'.

     Eg: make LIB_TYPE=STATIC

4) **clean options**:
   - **clean**: Deletes all the object files and RVDS,UNIX ELINUX executables.

Note:

- In 'Makefile_test.init' at directory '/fsl_mad_multimedia_codec*/build/*', the paths for the compiling and linking tools are hard coded for the current set-up. These paths may not be the same in the user's directory set up. Hence, the 'Makefile_test.init' should be modified to point to the directories where the linking and compilation tools are present before building the application for board.
- To build AAC plus test application, the AAC library is needed for linking, lib_aac_dec_arm11_elinux.so lib_aac_dec_arm11_elinux.a or lib_aac_dec_arm11_lervds.a may lib_aac_dec_arm9_elinux.so lib_aac_dec_arm9_elinux.a or lib_aac_dec_arm9_lervds.a may needed according to each build option.

The following table summarises the build options,

Target	Compilation Environment	Build Options	Executable Name
Board	Redhat Linux Machine	BUILD= ARM11ELINUX	test_aacplus_dec_arm11_elinux
RVDS	PC (Using Cygwin)	BUILD= ARM11LERVDS	test_aacplus_dec_arm11_lervds

5.4 Test Application for UNIX

To build the UNIX test application 'test_aacplus_dec_x86_unix', run 'make BUILD=UNIX from test directory. The makefile shall create the required directory to hold the object files and will build the UNIX libraries in the library directory and create 'lib_aacplus_dec_x86_unix.a' in the library directory and link the libraries to build the final executable test_aacplus_dec_x86_unix in test directory.

make BUILD=UNIX clean (to clean)

make BUILD=UNIX (for Unix/Linux machine)

Target	Compilation Environment	Build Options	Executable Name
Unix/ Linux	Unix/Linux machine	BUILD=UNIX	test_aacplus_dec_x86_unix

6 Test Application Execution

6.1 ELINUX

It can be used to decode adif/adts files as follows

aacplus_dec_arm11(9)_elinux ***<input_file>*** ***<output_file>*** ***<downsample_flag>***

where ***input_file –*** *adts/adif input file*

output_file – *output file name without extension(.wav extension will be appended to it by the test code)*

downsample_flag – flag to indicate use downsample or not. 1 means use downsample.

6.2 RVDS

It can be used to decode adif/adts files as follows

aacplus_dec_arm11(9)_lervds ***<input_file>*** ***<output_file>*** ***<downsample_flag>***

where ***input_file –*** *adts/adif input file*

output_file – *output file name without extension(.wav extension will be appended to it by the test code)*

downsample_flag – flag to indicate use downsample or not. 1 means use downsample.

6.3 UNIX Reference

It can be used to decode adif/adts files as follows

aacplus_dec_arm11_unix ***<input_file>*** ***<output_file>*** ***<downsample_flag>***

where ***input_file –*** *adts/adif input file*

output_file – *output file name without extension(.wav extension will be appended to it by the test code)*

downsample_flag – flag to indicate use downsample or not. 1 means use downsample.

7 Conformance Testing

7.1 AAC/SBR/PS Conformance testing

Scripts provided with package can be used for conformance testing of the decoder. These scripts compare the characteristics of the .wav file generated from the 'DUT' with the reference output generated by the reference decoder provided, to decide if the vector has been decoded correctly.

The scripts provided are

1. Convert – To convert the **.mp4** files to **.adif** or .**adts** files
2. callDUT – To specify the decoder to be used
3. toStereo – To convert single channel vectors to stereo vectors
4. mpegAACConformance – Script for AAC conformance testing
5. mpegSBRConformance - Script for SBR conformance testing
6. parametricstereoConformance - Script for PS conformance testing

These scripts internally use the executables provided along with this package.

1. ./adiftomp4 – to convert mp4 files to adts/adif format
2. ./mp4audec_mc – reference AAC decoder
3. ./ctrms – Used by toStereo script
4. ./ssnrcd – Used by mpegAACConformance script for SNR,RMS difference and cepstrum distortion comparisons.
5. ./pnsConformance - Used by mpegAACConformance script for PNS conformance
6. ./sbrConfToolHQ – Used by mpegSBRConformance script for SBR conformance
7. ./sbrConfToolPS– Used by parametricstereoConformance script for PS conformance
8. ./run - Used for calling any one of the above.Sets the environment variables. BSFORMAT field must be set to adif or adts

The input vectors are provided as .mp4 files in the following directories
test_vectors/aacplus_dec/mpeg_aac_conformance - **for AAC**
test_vectors/aacplus_dec/mpeg_sbr_conformance – **for SBR**
test_vectors/aacplus_dec/ps-conformance – **for PS**

NOTE: - As the scripts provided are linux based it is advisable to install CYGWIN so that the scripts can be run on windows without any modification.

Please change directory to "test_util/scripts" directory to run all the scripts

The following steps are to be followed to check for the decoder functionality.

1. **./run convert**
   Converts the .mp4 files to adts/adif depending on BSFORMAT in run script.
2. **./run aacConformance**

Runs the reference decoder and 'DUT' for all the vectors specified and performs AAC conformance test..It create a *.log* file and a summary in *.results* file with date and time stamp on it.

3. **/run mpegSBRConformance**
   Runs the reference decoder and 'DUT' for all the vectors specified and performs SBR conformance test. It creates a *.log* file and a summary in *.results* file with date and time stamp on it.
4. **./run parametricstereoConformance**
   Runs the reference decoder and 'DUT' for all the vectors specified and performs PS conformance test. It create a *.log* file and a summary in *.results* file with date and time stamp on it
5. ./run all
   Run aacConformance (MPEG-4), mpegSBRConformance, mpegPSConformance and certification. It create a *.log* file and a summary in *.results* file with date and time stamp on it. This .log file can send to CT to do certification.

7.2 Conformance testing of PCM files generated from board

Once the PCM files are generated in specified output directory the conformance test can be Done on ***LINUX (native)*** using the scripts under /scripts_compare fold. These scripts use the following executables which is provided in the code/certification/bin directory

aacConformance
pcm2wav – Utility to convert PCM files to WAV files (needed by the conformance tool)
mp4audec_mc – AAC reference decoder to generate reference outputs
pnsConformance – Conformance tool to test PNS test vectors
ssnrcd – Tool to find segmental SNR and cepstral distortion between two wav files
ctrms – Tool to generate RMS difference between two wav files

mpegSBRConformance
pcm2wav – Utility to convert PCM files to WAV files (needed by the conformance tool)
sbrConfToolHQ – Tool to test the SBR Conformance

parametricstereoConformance
pcm2wav – Utility to convert PCM files to WAV files (needed by the conformance tool)
sbrConfToolPS – Tool to test the PS Conformance

The paths of these executables should be such that it selects from the bin directory. The script takes care of generating the intermediate directories to store the wave files generated by the reference decoder and DUT .However the path of the PCM files needs to be exported in the command line to enable the script to take the PCM files as input. Also the path of the reference MP4 files (to be used by ***sbrConfToolHQ*** *in* **mpegSBRConformance** *script and sbrConfToolPS in* **parametricstereoConformance** *for generating reference wav files)* needs to be given appropriately.

The scripts compares the generated reference wav files with wav files obtained from aacPlus decoder and performs the conformance test. The results are logged in a ***log*** and ***results*** with date and time stamp in its name.

The results file can be viewed to see the final results.
Same procedure can be applied to test adts files by exporting the *BSFORMAT* to adts on the board command prompt.