International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-1, November 2019

Unit Testing of ETSI DECT Codec

Arvind Vishnubhatla

Abstract*: *The low complexity communication codec (LC3Plus) plus performs audio quality tests which are both subjective and objective ranging from narrowband to fullband are tested at clean and distorted channels. The relevant DECT scenarios for transcoding and VOIP are investigated. In this paper we take LC3Plus from ETSI and test the correctness by compiling the program under linux environment.
A careful planning of the test stimuli helps create useful test cases for `testing the LC3Plus encoder and decoder. Testing is performed meticulously to verify the correctness of the standard.

***Keywords:* speech, codec, LC3Plus, stimuli , testing, linux**

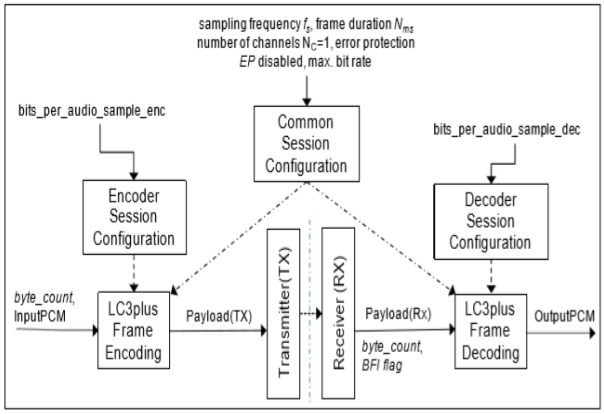
I. INTRODUCTION

The low complexity communication codec(LC3Plus) plus performs audio quality tests which are both subjective and objective ranging from narrowband to fullband are tested at clean and distorted channels. The relevant DECT scenarios for transcoding and VOIP are investigated. In this paper we take LC3Plus from ETSI and test the correctness by compiling the program under linux environment.
A careful planning of the test stimuli helps create useful test cases for `testing the LC3Plus encoder and decoder.Testing is performed meticulously to verify the correctness of the standard.
References [1]-[7] discuss various facets like session description protocol ,Real time transport protocol for real time applications Profile for audio and video conferences, Session description protocol ,Media RTP payload formats and file storage formats. Sound quality assessment is also discussed.

Revised Manuscript Received on November 30, 2019.
* Correspondence Author
Arvind Vishnubhatla*,ECE,Gokaraju Rangaraju Institute of engineering and technology,Hyderabad,India,vainfo66@gmail.com

II. PROPOSED METHODOLOGY

A. Block Diagram



B. Code Extraction

The DECT speech source code is available as ts_103634v010101p0.zip.
Now unzip the contents using the command
Archive: ts_103634v010101p0.zip
creating: LC3plus_ETSI_src_v11482_20190725/
creating: LC3plus_ETSI_src_v11482_20190725/testvec/
inflating:
LC3plus_ETSI_src_v11482_20190725/testvec/md5_dec.txt
…….
…….
Now the directory structure looks like the one shown below
[root@vlsibtechclient26 dect]# ls
LC3plus_ETSI_src_v11482_20190725
ts_103634v010101p0.zip
package.info ts_103634v010101p.pdf
[root@vlsibtechclient26
LC3plus_ETSI_src_v11482_20190725]# ls
conformance Readme.txt **src** testvec tools
The code is now available in the src directory.
[root@vlsibtechclient26fixed_point]# ls
adjust_global_gain_fx.c per_band_energy_fx.c
al_fec.c plc_apply_fx.c
apply_global_gain_fx.c plc_classify_fx.c
ari_codec.c plc_damping_scrambling_fx.c
attack_detector_fx.c plc_lpc_scaling_fx.c
basic_op plc_main_fx.c
basop_mpy.c plc_noise_substitution_fx.c
basop_mpy.h plc_phecu_f0_refine_first_fx.c
basop_util.c plc_phecu_fec_hq_fx.c
basop_util.h plc_phecu_lf_peak_analysis_fx.c
codec_exe.c
plc_phecu_peak_locator_fx.c

constants.c
constants.h
cutoff_bandwidth.c
dct2_fx.c
dct4_fx.c
dec_entropy.c
dec_lc3.c
defines.h
detect_cutoff_warped_fx.c
enc_entropy.c
enc_lc3.c
estimate_global_gain_fx.c
fft.c
functions.h
imdct_fx.c
lc3.c
lc3.h
levinson_fx.c
license.h
ltpf_coder_fx.c
ltpf_decoder_fx.c
makefile
mdct_fx.c
mdct_shaping_fx.c
msvc
noise_factor_fx.c
noise_filling_fx.c
olpa_fx.c
pc_apply_fx.c
pc_classify_fx.c
pc_main_fx.c
pc_update_fx.c
plc_phecu_setf0hz_fx.c
plc_phecu_tools_fx.c
plc_tdac_fx.c
plc_tdc_inverse_odft_fx.c
plc_tdc_lagwin_fx.c
plc_tdc_main_fx.c
plc_tdc_pre_emphasis_fx.c
plc_update_aft_imdct_fx.c
plc_update_fx.c
plc_xcorr_fx.c
pvq_dec_fx.c
pvq_enc_fx.c
pvq_index_fx.c
quantize_spec_fx.c
reorder_bitstream_fx.c
resamp12k8_fx.c
residual_coding_fx.c
residual_decoding_fx.c
rom_basop_util.c
rom_basop_util.h
scale_signal24_fx.c
setup_dec_lc3.c
setup_dec_lc3.h
setup_enc_lc3.c
setup_enc_lc3.h
sns_compute_scf_fx.c
sns_interpolate_scf_fx.c
sns_quantize_scf_fx.c
tinywavein_c.h
tinywaveout_c.h
tns_coder_fx.c
tns_decoder_fx.c

C. Algorithm

Now we compile the code using the command
make -f makefile
A snapshot of the compilation process is as shown below:
Compiling adjust_global_gain_fx.c
Compiling al_fec.c
Compiling apply_global_gain_fx.c
Compiling ari_codec.c
Compiling attack_detector_fx.c
Compiling basop_mpy.c
…..
….
Linking LC3plus

D. Flow

The directory structure now looks like the one shown below:

adjust_global_gain_fx.c
al_fec.c
apply_global_gain_fx.c
ari_codec.c
attack_detector_fx.c
basic_op
basop_mpy.c
basop_mpy.h
basop_util.c
basop_util.h
build
codec_exe.c
constants.c
constants.h
cutoff_bandwidth.c
dct2_fx.c
dct4_fx.c
dec_entropy.c
dec_lc3.c
defines.h
detect_cutoff_warped_fx.c
enc_entropy.c
enc_lc3.c
estimate_global_gain_fx.c
fft.c
functions.h
imdct_fx.c
lc3.c
lc3.h
LC3plus
levinson_fx.c
license.h
ltpf_coder_fx.c
ltpf_decoder_fx.c
makefile
mdct_fx.c
mdct_shaping_fx.c
msvc
noise_factor_fx.c
noise_filling_fx.c
olpa_fx.c
pc_apply_fx.c
pc_classify_fx.c
pc_main_fx.c
pc_update_fx.c
per_band_energy_fx.c
plc_apply_fx.c
plc_classify_fx.c
plc_damping_scrambling_fx.c
plc_lpc_scaling_fx.c
plc_main_fx.c
plc_noise_substitution_fx.c
plc_phecu_f0_refine_first_fx.c
plc_phecu_fec_hq_fx.c
plc_phecu_lf_peak_analysis_fx.c
plc_phecu_peak_locator_fx.c
plc_phecu_setf0hz_fx.c
plc_phecu_tools_fx.c
plc_tdac_fx.c
plc_tdc_inverse_odft_fx.c
plc_tdc_lagwin_fx.c
plc_tdc_main_fx.c
plc_tdc_pre_emphasis_fx.c
plc_update_aft_imdct_fx.c
plc_update_fx.c
plc_xcorr_fx.c
pvq_dec_fx.c
pvq_enc_fx.c
pvq_index_fx.c
quantize_spec_fx.c
reorder_bitstream_fx.c
resamp12k8_fx.c
residual_coding_fx.c
residual_decoding_fx.c
rom_basop_util.c
rom_basop_util.h
scale_signal24_fx.c
setup_dec_lc3.c
setup_dec_lc3.h
setup_enc_lc3.c
setup_enc_lc3.h
sns_compute_scf_fx.c
sns_interpolate_scf_fx.c
sns_quantize_scf_fx.c
tinywavein_c.h
tinywaveout_c.h
tns_coder_fx.c
tns_decoder_fx.c

The compilation has created an executable **LC3plus** shown in bold.

III. RESULT ANALYSIS

To test the system a perl script is used as shown below
input md5_dec.txt **testvecCheck.pl**
plc_fer.dat
md5_bin.txt Readme.txt
A snapshot of testvecCheck.pl is shown below:

```
====================================
LC3plus ETSI Testvector script V0.0.1
====================================
====================
WHAT IS THIS SCRIPT?
====================
```

This is the LC3plus ETSI testvector script. It checks whether the output of your LC3plus build produces the expected results and matches the precalculated MD5 hashes for a number of operating points. It is meant to be used for the fixed-point version of LC3plus only.
The following configurations are tested:
Samplingrate [Hz] | Bitrate [bps] | EP Mode [0 = off, 4 = highest protection]

```
------------------+---------------+-----------------------------------
------
 8000            | 32000         | 0, 4
 16000           | 32000         | 0, 4
 24000           | 48000         | 0, 4
 32000           | 48000         | 0, 4
 44100           | 64000         | 0, 4
 48000           | 64000         | 0, 4
################################################
##
my $EXE_TST = './../src/fixed_point/LC3plus'; # Path to test
LC3plus executable
my $md5_bin = './md5_bin.txt';
my $md5_dec = './md5_dec.txt';
my $MY_MD5  = 'md5sum'; # System dependent MD5 call
my $epf = '-epf ./plc_fer.dat';
################################################
##
use strict;
use File::Basename;
use File::Path 'rmtree';
use POSIX;
my $VERSION = 'V0.0.1';
my $timestamp = strftime "%m_%d_%Y_%H_%M_%S",
localtime;
my $tmp_folder = "lc3plus_testvectors".$timestamp;
my $inputFile = "./input/thetest";
my $output_folder_stream_tst =
$tmp_folder."/bitstream_tst";
my $output_folder_decoded_tst =
$tmp_folder."/decoded_tst";
my $report =
"lc3plus_testvectors_report_".$timestamp.".txt";
my $fh;
my $quiet = '>/dev/null 2>&1';
my $testvectors_fail = 0;
my @SR = (8000, 16000, 24000, 32000, 44100, 48000);
my @BR_8 = (32000);
my @BR_16 = (32000);
my @BR_24 = (48000);
my @BR_32 = (48000);
my @BR_441 = (64000);
my @BR_48 = (64000);
----------
……
```

The input test vectors are as shown below:

```
[root@vlsibtechclient26 input]# ls
thetest16.wav  thetest32.wav  thetest48.wav
thetest24.wav  thetest44.wav  thetest8.wav
```

[root@vlsibtechclient26 testvec]# ./testvecCheck.pl
Script started...
Creating files with test executable...
...done!
The testvector check test was passed!

The testvector directory is as shown

```
cd _testvectors09_28_2019_09_49_55
```

At the output of the execution we see the following output

```
[root@vlsibtechclient26 bitstream_tst]# ls
thetest16_32000_EP0.lc3plus  thetest44_64000_EP0.lc3plus
thetest16_32000_EP4.lc3plus  thetest44_64000_EP4.lc3plus
thetest24_48000_EP0.lc3plus  thetest48_64000_EP0.lc3plus
thetest24_48000_EP4.lc3plus  thetest48_64000_EP4.lc3plus
thetest32_48000_EP0.lc3plus  thetest8_32000_EP0.lc3plus
thetest32_48000_EP4.lc3plus  thetest8_32000_EP4.lc3plus
```

At the end of decoding we see the following output

```
[root@vlsibtechclient26 decoded_tst]# ls
thetest16_32000_EP0.wav  thetest32_48000_EP0.wav
thetest48_64000_EP0.wav
thetest16_32000_EP4.wav  thetest32_48000_EP4.wav
thetest48_64000_EP4.wav
thetest24_48000_EP0.wav  thetest44_64000_EP0.wav
thetest8_32000_EP0.wav
thetest24_48000_EP4.wav  thetest44_64000_EP4.wav
thetest8_32000_EP4.wav
```

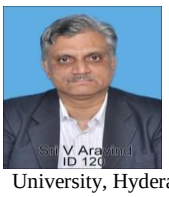
IV. CONCLUSION

A systematic unit testing is performed on the ETSI DECT speech encoder and decoder with proper test stimuli to exhaustively test the given code. The test results reveal that the testing is performed satisfactorily.

REFERENCES

1. IETF RFC 3264: "An Offer/Answer Model with Session Description Protocol (SDP)", Rosenberg, J. and H. Schulzrinne, June 2002.
2. IETF RFC 3550: "RTP: A Transport Protocol for Real-Time Applications", STD 64, Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, July 2003.
3. IETF RFC 3551: "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, Schulzrinne, H. and S. Casner, July 2003.
4. IETF RFC 4855: "Media Type Registration of RTP Payload Formats", Casner, S., February 2007.
5. IETF RFC 4566: "SDP: Session Description Protocol", Handley, M., Jacobson, V., and C. Perkins, July 2006. NOTE: Available at https://www.rfc-editor.org/info/rfc4566.
6. IETF RFC 4867: "RTP Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs", Sjoberg, J., Westerlund, M., Lakaniemi, A., April 2007.
7. European Broadcasting Union: "Sound Quality Assessment Material recordings for subjective tests".

AUTHORS PROFILE



Arvind Vishnubhatla
Areas of Interest: Signal Processing , Image Processing , Real time Embedded Systems , Photonics
Education
- Ph.D., Jawaharlal Nehru Technological University, Hyderabad
- MS, Northern Illinois University,Dekalb IL
- BE, Thapar institute of Engineering and Technology, PATIALA

Academic/Industrial Experience
1. Professor, ECE, Gokaraju Rangaraju institute of Engineering and Technology , 2009-Present
2. Associate Professor,ECE, Gokaraju Rangaraju institute of Engineering and Technology , 2002-2009
3. Project Manager, Ericsson Communications India Limited, 2000-2002
4. Senior Software Engineer, **Motorola, Location (Network Solution Sector, Schaumburg, IL)**, 1999-2000
5. Senior Software Engineer, **Motorola, Location(Bangalore)**, India, 1993-1997
6. Software Engineer, **Bharat Heavy Electricals Ltd(Electronics Division. Bangalore)** 1989-1992