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.

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
package.info
ts_103634v010101p0.zip

[root@vlsibtechclient26]
ls
LC3plus_ETSI_src_v11482_20190725
package.info
src

Now the directory structure looks like the one shown below [root@vlsibtechclient26 fixed_point]# ls
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
codec_exe.c

codec_exe.c

codec_exe.c

codec_exe.c

B. Code Extraction

The DECT Extraction 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 fixed_point]# ls
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

codec_exe.c

Revised Manuscript Received on November 10, 2019.
* Correspondence Author
Arvind Vishnubhatla* ECE, Gokaraju Rangaraju Institute of engineering and technology, Hyderabad, India, vainfo66@gmail.com
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_f.c
Compiling al_fec.c
Compiling apply_global_gain_f.c
Compiling ar1_coder.c
Compiling attack_detector_f.c
Compiling basop_mpy.c

D. Flow

The directory structure now looks like the one shown below:
adjust_global_gain_f.c plc_update_f.c
al_fec.c plc_band_energy_f.c
apply_global_gain_f.c plc_apply_f.c
ar1_coder.c plc_classify_f.c
attack_detector_f.c plc_damping_scrambling_f.c
basic_op plc_lpc_scaling_f.c
basop_mpy.c plc_main_f.c
basop_mpy.h plc_noise_substitution_f.c
basop_util.c plc_phecu_f0_refine_first_f.c
basop_util.h plc_phecu_f0_refine_first_f.c
build plc_phecu_iff_peak_analysis_f.c

codec_exe.c plc_phecu_peak_locator_f.c
constants.c plc_phecu_setf0hz_f.c
constants.h plc_phecu_tools_f.c
cutoff_bandwidth.c plc_tdac_f.c
dct2_f.c plc_tdc_inverse_odft_f.c
dct4_f.c plc_tdc_lagwin_f.c
dec_entropy.c plc_tdc_main_f.c
dec_lc3.c plc_tdc_pre_emphasis_f.c
defines.h plc_update_aft_imp_f.c
detect_cutoff_warped_f.c plc_update_f.c
detect_cutoff_warped_f.c plc_update_f.c
enc_entropy.c plc_xcorr_f.c
enc_lc3.c plc_xcorr_f.c
estimate_global_gain_f.c pvq_dec_f.c
ff.c pvq_enc_f.c
functions.h quantize_spec_f.c
imdct_f.c reorder_bitstream_f.c
lc3.c resamp12k8_f.c
lc3.h residual_coding_f.c
levinson_f.c residual_decoding_f.c
license.h rom_basop_util.c
ltcf_coder_f.c rom_basop_util.h
ltcf_decoder_f.c scale_signal24_f.c
makefile setup_dec_l.c3
mdct_f.c setup_dec_l.c3
mdct_shaping_f.c setup_enc_l.c3
msvc setup_enc_l.c3
noise_factor_f.c sns_compute_scf_f.c
noise_filling_f.c sns_interpolate_scf_f.c
olfa_c.f sns_quantize_scf_f.c
pc_apply_f.c tinywavein_c.h
pc_classify_f.c tinywaveout_c.h
pc_main_f.c tns_coder_f.c
pc_update_f.c tns_decoder_f.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
levinson_f.c rom_basop_util.c
license.h rom_basop_util.h
ltcf_coder_f.c scale_signal24_f.c
ltcf_decoder_f.c setup_dec_l.c3
makefile setup_dec_l.c3
mdct_f.c setup_enc_l.c3
mdct_shaping_f.c sns_compute_scf_f.c
msvc sns_interpolate_scf_f.c
noise_factor_f.c sns_quantize_scf_f.c
olfa_c.f tinywavein_c.h
pc_apply_f.c tinywaveout_c.h
pc_classify_f.c tns_coder_f.c
pc_main_f.c tns_decoder_f.c
===================
```

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]**
The test vectors are as shown below:
[root@vlsibtechclient26 bitstream_tst]# ls
At the output of the execution we see the following output
[root@vlsibtechclient26 decoded_tst]# ls
At the end of decoding we see the following output
[root@vlsibtechclient26 bitstream_tst]# ls
The test results reveal that

REFERENCES


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.

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 , 2002-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

Published By:
Blue Eyes Intelligence Engineering & Sciences Publication