Application Note

Akademie věd České republiky
Ústav teorie informace a automatizace AV ČR, v.v.i.

UTIA Evaluation Board v1.7-v1.8 Beamforming Demo

Zdeněk Pohl, Lukáš Kohout
zdenek.pohl@utia.cas.cz, kohoutl@utia.cas.cz

Revision history

Rev.	Date	Author	Description
01	2.7.2019	Z.P.	Document creation
02	5.9.2019	Z.P.	Fixed address of data in packet structure

Contents

Acknowledgement

This work has been supported from project SILENSE, project number ECSEL 737487 and MSMT 8A17006.

1 Description

This document provides documentation to hardware evaluation board version v1.7-v1.8 developed in UTIA (UTIA evBoard v1.7, v1.8), hardware interfaces in FPGA and accelerated DSP chain for implementation of beamforming application. To view measured data, the PC application was also developed and its operation is described in this document. The hardware platform consists of three basic components: TE0720 FPGA SoM module, TE0706 carrier board and UTIA evBoard v1.7, see

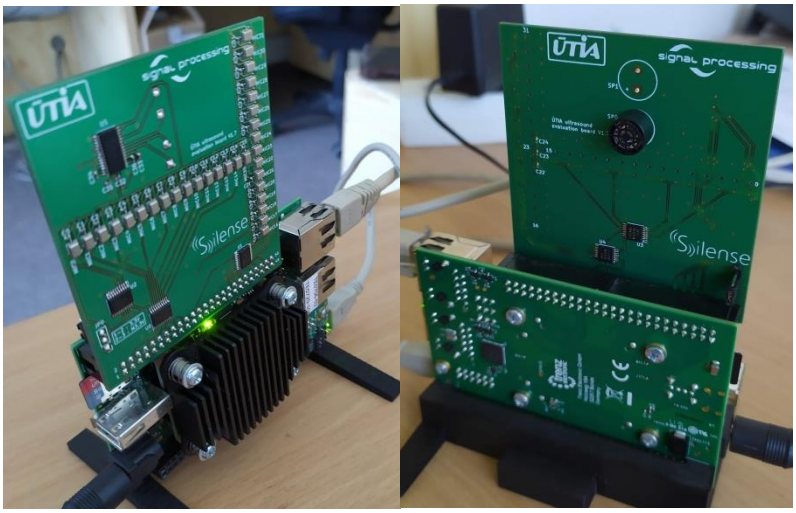


Figure 1.

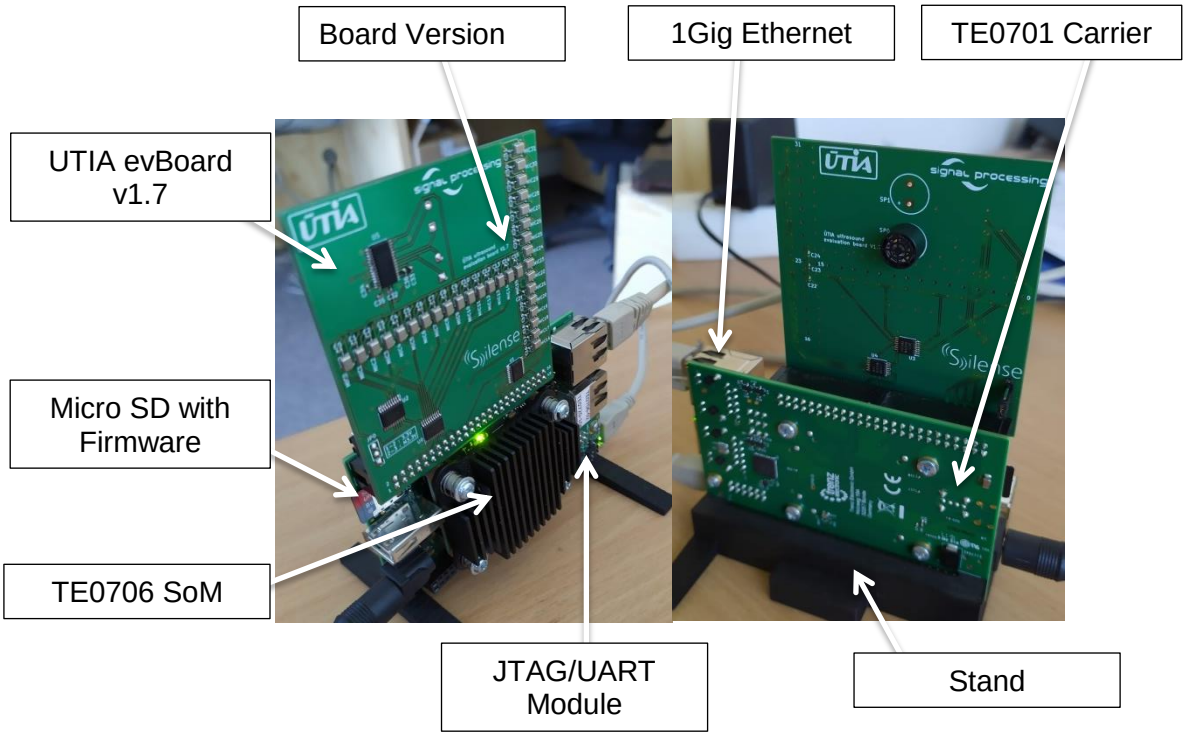


Figure 1: UTIA evBoard with FPGA module and carrier board.

2 Board Features

The evaluation board version 1.7 implements following features:

a) Linear microphone array consisting of 32 digital microphones.
b) Microphone frequency range 0-80kHz
c) Array maximal frequency 40 kHz. Distance between microphone acoustic ports = 3.8 mm
d) Dual 40 kHz piezo US speaker with wide output beam to cover maximal area
e) Common clock distribution to all microphones
f) US speaker driving circuit
g) Connector for mating with TE0706 carrier board J5 connector.
h) IMPORTANT: UTIA evBoard v1.7 is compatible only with FPGA modules supporting 3.3V IO.

3 How to Run

Prerequisites:

1. UTIA evBoard v1.7 assembled with TE0706 carrier with TE0720-1CF/1QF/2IF Zynq SoM.
   IMPORTANT: When assembling UTIA evBoard to TE0706 carrier make sure J5 pins are correctly aligned.
2. 5V power source for TE0706.

3. (optional) Mini USB cable for Virtual UART connection to PC via JTAG/UART module.
4. UTP Cable for connection to PC data visualization. We recommend to use Gigabit Ethernet interface for best performance. (board uses UDP multicast to pass beamformer output)
5. (optional) Board stand mounted to support assembled boards in upright position. Stand also features tripod mount possibility.
6. IMPORTANT: Carrier board jumpers J10,J11,J12 must have all shorts placed to 2-3 position.
7. The DHCP server must be running on local network where the board is connected. Alternatively, the UART connection to board can be used to setup Ethernet interface manually.
8. Micro SD card with firmware.

Updating/Writing firmware to Micro SD card:

1. Plug the Micro SD card to PC reader (it may be needed to use SD card adapter)
2. Format the card to FAT32 filesystem. Alternatively, the FAT32 can be only first partition on the card with size at least 128 MByte.
3. Copy content of firmware package to the SD card root.
4. Disconnect card and remove it from reader.

Running the application:

1. Insert Micro SD card with firmware to J4 slot on TE0706.
2. Plug the power source. Board should start booting automatically. If the boot process does not start automatically, the reset button S2 on carrier board TE0706 can be used to initiate booting.
3. The beamformer application should start automatically after linux boot. In case of need the application can be also started manually by following steps:
    a. Using serial console:
        i. Use optional Mini USB cable and connect J4 connector on JTAG/UART module to PC.
        ii. Use serial console application in PC (putty for example) and connect to board using parameters: 115200bps, parity none, stop bits 1, data 8bit, flow none
        iii. log in using following credentials: login: root , password: root
    b. Using ssh server:
        i. Find IP address assigned to board by DHCP server.
        ii. Use ssh client on PC to connect the board.
        iii. Use login: root , password: root

    Run user application by following command:

    ```
    > /media/card/evBoard_das_ffbf_hvplane.elf
    ```

    IMPORTANT: If another instance of the application is running, it must be terminated before executing this command. Otherwise the application will crash.

4. The board features FTP server which can be used for accessing captured data files or beamformer configuration file:
    a. Configuration is stored in `evboard_ffbf.cfg` file at location:

    ```
    > /media/card/evboard_ffbf.cfg
    ```

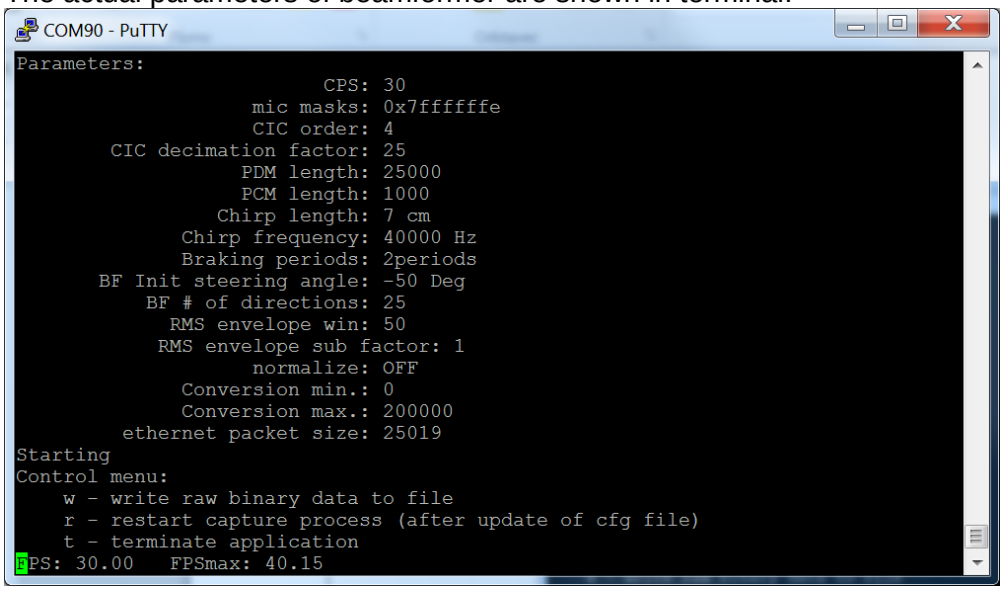
b. Captured data can be found in the following file after they are written (must be manually initiated while the application is running):

```
> /media/card/test.bin
```

NOTE: Provided package contains folder evBoard_config_files where examples of configuration files can be found. Choose one and rename it while copying to the board.

5. While the application is running:

a. The actual parameters of beamformer are shown in terminal:



The application updates actual frame rate (FPS) for beamformer output, at the same time it presents maximal framerate (FPSmax) which describes the maximal FPS which can be reached in this configuration of the beamformer (Real FPS may be lower because of lower chirp per second (CPS) request in configuration file.)

b. From terminal, the beamformer application can be controlled by entering following command characters:
   i. **w –** immediately writes raw binary data of one captured chirp to file `test.bin`.
   ii. **r –** restarts the application. It is intended to be used after change in configuration file. Application re-reads new parameters from file and restarts.
   iii. **t –** terminates the beamformer application

4 Beamformer FPGA Design

This section provides overview of beamformer HW accelerated solution of Zynq FPGA. The parameters of individual hardware cores which can be modified by the changes to configuration file are discussed.

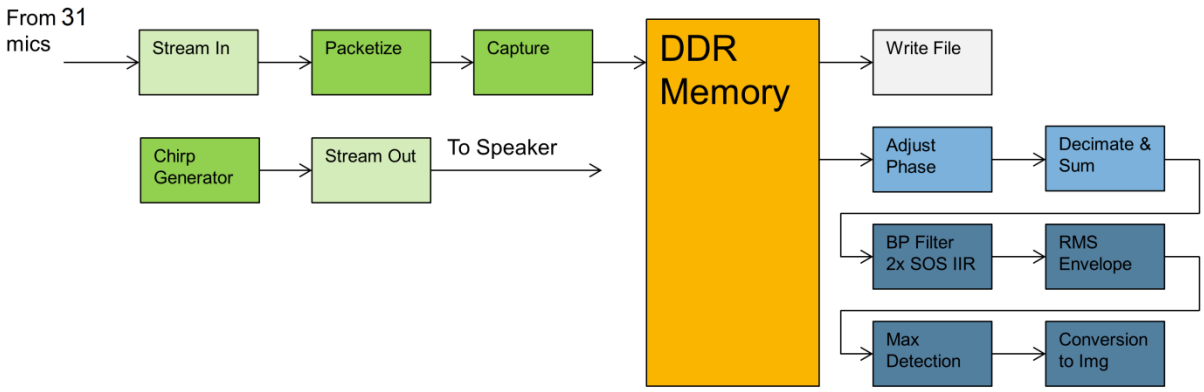


Figure 2: FPGA implementation of the beamformer accelerators

The design shown in Figure 2 contains following blocks:

1. Packetize – Not configurable. The block adds end markers to microphone data stream. With the data divided to blocks the consequent capture can use DMA unit to store data to DDR memory.
2. Capture – Not configurable. Stores blocks of data to DDR memory using DMA unit.
3. Chirp Generator – This core generates chirp. The frequency of the chirp, length of the chirp and number of reverse phase periods at the end of each chirp can be configured. The length of the chirp is defined in cm. For example if 7 cm chirp is requested, the wave length is 8 mm @40 kHz thus 8 periods at 40 kHz will be generated. If 2 braking periods are requested, the result will be 8 periods of which 2 will have reverse phase to cancel ringing.
4. Adjust Phase – Configurable. Adjusts phase of microphone signals depending on the steering direction of the beamformer. We can define initial angle and number of sectors to which the viewfield will be divided. In default settings, the initial angle is -50.0 deg and 25 beamformer directions. That means the view field will be from -50 deg to 50 deg and the angular step will be 4,17 deg. Beamformer output is evaluated sequentially for each direction. Thus the execution time for 50 directions is 2x longer than for 25 directions.
5. Decimate and sum – Configurable. By the parameters of the sum part we can define microphones which will be used for beamforming (by mask applied to microphone signals). The parameters of decimator are order and subsampling/decimation factor. We recommend to keep default values for decimator (order = 4, decimation factor 25).

6. BP Filter – Not configurable. Applies the bandpass filtering to DAS beamformer output. Magnitude and phase response of used filter consisting of 2 Second Order Stages (SOS) is shown in Figure 3. The filter uses floating point arithmetic.

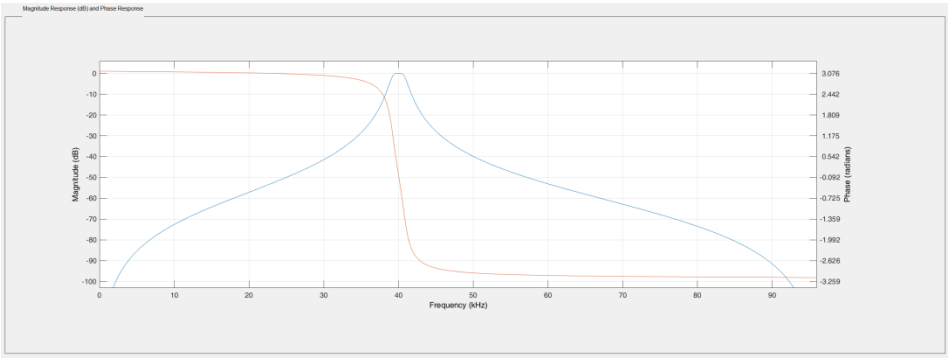


Figure 3: Magnitude and phase response of used BP filter

7. RMS Envelope – Configurable. The core computes RMS using running window and optionally can also subsample the output. The configurable parameters are window length and subsampling factor applied to output envelope. Default values are window length 50 and no subsampling (i.e. subsampling factor = 1).
8. Max Detection – Not configurable. The core detects position and value of global maximum in output data. Output is index of beamformer direction, index of sample in that direction and RMS envelope value at the point of maxima.
9. Conversion to Img – Configurable. The core can be operated in two modes:
    a. Normalization turned off – in this mode minimal value and maximal value are parameters used to convert RMS envelope data to value between 0-255. Values lower or equal to minimum are converted to 0, values bigger or equal to maximum are converted to 255. All values in between are linearly mapped to range 0-255.
    b. Normalization is turned on – in this mode, the exponential function is used to compensate strength of the recorded ultrasound depending on distance. The function:

$$y = a.e^{b}$$

where default parameters a = 667013 and b = -0.00185358 are configurable by the configuration file.

5 Beamformer Application Configuration File

The beamformer application was implemented to load its configuration from file `'evboard_ffbf.cfg'` in the host filesystem folder `'/media/card'`. The individual parameter values must be present in the file in given order and within the correct ranges. Writing other values than listed in the following table may crash the application.

Row #	Default Value	Value Range	Description
1	30	1-60	Desired number of chirps per second generated and captured, if too high it limits maximal PCM length
2	0x3ffffffe	-	Mask which enables individual microphones. Microphones masked out will not be added by the beamformer. Mic32 is not connected to IO and thus bit 31 of the mask must be always 0. When reducing number of mics please take into account that mic8 and mic23 are always middle speakers for horizontal and vertical beamformer respectively
3	4	1-4	Order of CIC decimation filter
4	25	25 (other values not supported at this moment)	Decimation factor. Mics are providing 1bit @ 4,8MHz. Decimation 25 results in subsampling of data to 192kHz and bit width increase to approx. 24bit IMPORTANT: Do not change this value.
5	2000	1-192000	Number of PCM samples to be captured – will be limited by application if requested CPS is too high. Value of 192000 will record one second of mic data. Thus CPS value must be equal 1.
6	5	1-30	Chirp length in cm. Length of generated chirp will be rounded to full periods of given chirp frequency value. Example: at 40 kHz one period wave length is 8mm, 5 cm chirp request will generate 6 periods. More chirp length creates more blurry output. Lower value creates weaker chirp.
7	40000	unsigned int	Desired chirp frequency. wavegenerator core creates rectangular chirp signal at desired frequency. Keep in mind that resonant frequency of used speaker is peaking around 40 kHz. Choosing another frequency will result in limited output power.
8	2	0-3	Number of speaker active braking periods to cancel ringing. Number must be lower than number of periods in chirp. We do recommend

			to use value 2 as value 3 has no effect and value 4 is already making reverse phase response of the membrane.
9	-50.0	float 0…-90.0	Beamformer angle from array axis. When -50.0 is set, the beamformer assumes its view angle to be from -50 to 50 degrees with respect to array axis.
10	25	1-100	Number of directions computed by the beamformer. The view angle will be divided to given number of sectors. For example value 3 for angle -50 to 50 says three beams will be computed using beam directions: [-50,0,50] Deg
11	50	1-100	Window length for RMS envelope computation
12	1	see decription	Subsampling of RMS envelope output, value 1 = subsampling OFF, value 2 = drop every odd sample, etc. We do recommend to use lower value than 1/10th of window length
13	0	unsigned int	Minimal value for conversion to CV_8UC1, used when normalize flag is off. Output is computed as min-max saturation and after that scaled to unsigned char range.
14	200000	unsigned int	Maximal value for conversion to CV_8UC1, used when normalization is off. Value of 200000 means that this value or higher will be scaled to output at 255.
15	1	0-1	Normalization: 0 – off, 1 – on
16	667013	float	First coefficient of normalization function fitted by calibration process: The calibration curve: Y = 667013.e^-0.00185358
17	-0.00185358	float	Second coefficient of normalization function fitted by calibration process: The calibration curve: Y = 667013.e^-0.00185358
18	0.0	float	Reserved value
19	0.0	float	Reserved value

6 Beamformer UDP Multicasting

UTIA evBoard multicasts its output to multicast group 239.0.01:6001. Every beamformer output is sent as one packet with header. Thus, to send all beamformer data we generate two packets one with horizontal and one with vertical data. Zynq ARM core is using the same endianness as Intel architecture – no conversions to network byte order are performed.

Packet structure:

Address	Size	Type	Name	Description
0x0000	1	unsigned char	group_id	Counter which identifies one set of measurements
0x0001	1	unsigned char	packet_data_id	0 – horizontal beamformer output 1 – vertical beamformer output Other – reserved for future measurements
0x0002	2	short	height	number of rows, rows are equivalent to beamformer directions
0x0004	2	short	width	number of columns = number of PCM samples in every direction, may be subsampled by the RMS envelope subsampling
0x0006	4	float	fi0	initial angle of beamformer
0x000A	1	unsigned char	fps	frames per second
0x000B	4	unsigned	max_val	Value of global maximum, computed in HW before normalization, can be used for calibration
0x000F	2	unsigned short	max_row	Row index of global maximum, computed before normalization, used for calibration
0x0011	2	unsigned short	max_col	Column index of global maximum, computed before normalization, used for calibration
0x0013	height*width	unsigned char	data	Beamformer data output array, pixel format is CV_8UC1

Example code for parsing header from packet which works on Intel architecture, it is assumed that packet data are in `msgbufh` array of `unsigned char` type:

```
int packet_group_id = msgbufh[0];
int packet_data_id = msgbufh[1];
int rows = (msgbufh[3] << 8) + msgbufh[2];
int cols = (msgbufh[5] << 8) + msgbufh[4];
float fi0 = *((float*)&msgbufh[6]);
unsigned char cps = msgbufh[10];
unsigned max_val = (msgbufh[14] << 24) + (msgbufh[13] << 16) + (msgbufh[12] << 8) +
msgbufh[11];
unsigned max_row = (msgbufh[16] << 8) + msgbufh[15];
unsigned max_col = (msgbufh[18] << 8) + msgbufh[17];
```

7 PC Application bf_view2

For visualization of multicasted beamformer data, the application for PC was developed using OpenCV 3.4.1 library. The application implements these functionalities:

1. Opens and reads UDP multicasts from the board.
2. Computes beamformer data transformation to interpolated polar images.
3. (optionally) Captures data from USB webcam to add also corresponding visual information.
4. Implements coding of displayed data to video file.

The application is parsing the UDP packets coming from the UTIA evBoard v1.7, v1.8

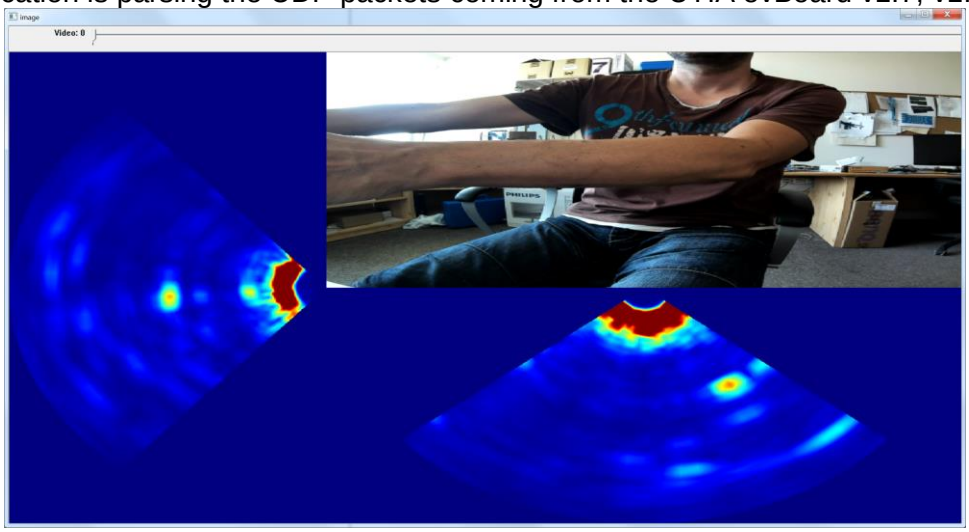


Figure 4: PC application bf_view2 GUI.

8 Using Matlab Scripts to Analyze Raw Captured Data

The folder Matlab of the package contains scripts and workspace variables needed to show captured raw data from microphones as a waveforms. To show captured waveforms follow these steps:

1. Raw data capture:
    a. Start the board and the beamformer application from Virtual UART or SSH console.

b. While the application is running enter ‘w’ in application UI at the moment you want to capture raw chirp data.
c. Connect to board using FTP and copy file ‘test.bin’ from board filesystem location: `/media/card` to PC

2. We assume, that location of the unpacked package is in the root of drive o: .

3. PC location of the ‘`test.bin`’ should be `o:\Matlab\testvec`
4. Open Matlab and change its working directory to `o:\Matlab`.
5. Call:

```
>> load('utia_evboard_17_30mic.mat')
```

the command loads variable `p_utia_evb25` which contains parameters needed to process the raw data. The file sets the parameters of the microphone array etc.

6. Use command:

```
>> pcm = show_evboard_waveforms_b('testvec/test.bin',
p_utia_evboard17_30mic);
```

After short time, 30 figures each containing captured data from one microphone converted to PCM at 192 kHz sampling are displayed.

7. The output variable `pcm` is array with PCM samples for every microphone and `chirp` contains chirp waveform subsampled to PCM sample time. Its waveform shape is affected by the subsampling. More precise result may be obtained from raw data without subsampling.

9 License

The package is provided by UTIA AV CR as is free of charge. For sources please contact authors.

10 Content of the package

```
evBoard_v17-18_appnote_files.zip
     - bf_view2                bf_view2 application, including:
                               source code, required dll libraries
                               from OpenCV 3.4.1 library
     - Doc                     Folder for this document
```

- evBoard_config_files	folder for example configuration files
- firmware	contains SD card content to be loaded to the root of first FAT32 partition on the SD card
- Matlab	Matlab scripts to process raw data captured by evBoard

11 References

[1] Silense project web pages: http://www.silense.eu