XXVIII Encontro da Sociedade Brasileira de Acústica
3 a 5 de outubro de 2018
Porto Alegre - RS

MICROPHONE ARRAY PROCESSING OF PULSE-DENSITY MODULATED BITSTREAMS

Ipenza, Sammy Carbajal[1]; Masiero, Bruno S.[2]

(1) Dept. of Communications, Campinas State University, sipenza@decom.fie.unicamp.br
(2) Dept. of Communications, Campinas State University, masiero@unicamp.br

RESUMO

Atualmente, os microfones digitais modulados por densidade de pulso (PDM) são amplamente utilizados em aplicações comerciais, já que esta é uma maneira eficiente de transmitir informação de áudio para processadores digitais em dispositivos móveis. No entanto, esses microfones digitais requerem custosos filtros de decimação de alta ordem para converter o fluxo PDM para a modulação por código de pulso (PCM) usada por estes processadores. Além disso, o estado-da-arte dos algoritmos de processamento digital de arranjos assumem que os sinais recebido dos microfones sempre está em uma representação em banda-base. A implementação destes algoritmos em sistemas embarcados, onde os recursos de processamento são críticos, ou em circuitos integrados para aplicações específicas (ASIC), onde a energia consumida e área também são críticas, pode se tornar muito dispendiosa devido ao uso de dezenas de filtros de decimação para converter os sinais de PDM para PCM. Este trabalho explora as vantagens e desvantagens de realizar o processamento de arranjo de microfones diretamente nos sinais modulados em densidade de pulsos ao invés de modulados por código de pulsos. É mostrado que o ruído de quantização, que é moldado para as frequências mais altas nos sinais PDM, não influencia no desempenho da detecção da fonte do arranjo de microfone e que o processamento no domínio PDM pode consumir menos recursos de software ou hardware do que o processamento convencional no domínio PCM.

Palavras-chave: PDM, microfones digitais, modulação sigma-delta, processamento de arranjo, beamforming.

ABSTRACT

Nowadays, pulse-density modulated (PDM) digital microphones are widely used on commercial applications as they have become a popular way to deliver audio to digital processors on mobile applications. However, these digital microphones require costly high-order decimation filters to translate PDM bitstreams to baseband multi-bit signals on pulse-code modulation (PCM). In addition, the state-of-the-art of beamforming and array processing algorithms take for granted that the microphones signals are already converted to PCM representation. The implementation of microphones array algorithms in embedded systems, where processing resources are critical, or in application specific integrated circuits (ASIC), where power and area are also critical, may become very expensive because of the use of the tens of decimation filters to convert PDM bitstreams to PCM signals. This work explores the advantages and disadvantages of processing PDM bitstreams instead of PCM signals for beamforming applications. It is shown that the quantization noise, which is shaped to the higher frequencies on PDM signals, does not influence in the microphone array source detection performance and that the array processing in the PDM domain can consume fewer software or hardware resources than conventional PCM beamforming processing.

Keywords: PDM, digital microphones, sigma-delta modulation, array processing, beamforming.

DOI: 10.17648/sobrac-87156

1. INTRODUCTION

In a digital microphone, a digital transductor converts the audio signal to an electrical signal, then an internal sigma-delta modulator converts the electrical analog signal to a digital pulse-density modulated (PDM) bitstream. For audio applications, the PDM bitstream is delivered at a sampling rate typically in the 1 MHz to 3 MHz range, while the audio or baseband signal is supposed to be in the 20 Hz to 20 kHz range. The modulator's order depends on the vendor and they are generally 2^{nd} or higher order modulators. The modulator shapes the quantization noise at higher frequencies while the audio signal remains in the baseband range. This quantization noise shaping is perfomed by analog feedback and oversampling stages within the modulator with the intention of increasing the signal-to-noise ratio (SNR) at baseband frequencies [1].

In order to get a pulse-code modulated (PCM) audio signal at a lower sampling rate, it is required to pass the PDM bistream through a decimation filter [2]. This decimation filter is commonly a Finite Impulse Response (FIR) filter as a linear phase response is required on audio applications [3].

The decimation filter design depends on the desired audio output quality, improving with a thinner passband ripple and a higher stop-band attenuation. Unfortunetaly, the number of FIR filter taps increases when either the passband ripple decreases, stop-band attenuation increases or transition band decreases. For audio applications, where a passband ripple less than 0.1dB and a stopband attenuation greater than 80dB is usually required, the number of taps may become greater than 2500 in case a single-stage decimation filter structure is implemented [4]. Because of the larger number of taps required to implement a single-stage decimation filter, Cascade Integrator-Combinator (CIC) filters in a multistage structure [2] are commonly used on filter design because their simpler architecture does not require multipliers, only adders [5].

Beamformer implementations using many PDM microphones will require greater hardware or software resources because a decimation filter is required for each microphone input to convert the PDM bitstream to PCM representation. Because of the quantity of resources required for beamforming applications, this work proposes alternative beamformer implementation methods which does not require a decimation filter for each PDM bistream input. These methods are based on processing the PDM bitstreams without decimate them, performing the processing at a higher sample rate. In these proposed methods the PDM bitstream is passed directly to frequency-domain via Fast-Fourier transform (FFT) blocks, then these signals are delayed and summed accordingly.

For comparison purposes, delay-and-sum beamformers (DAS) were implemented in hardware and software using these alternative beamforming methods. Then the performance in their software and hardware implementations is compared with the conventional[1] implementation methods [6][7].

2. CONVENTIONAL DELAY-AND-SUM BEAMFORMER IMPLEMENTATIONS

Delay-and-sum (DAS) beamformer is the oldest and simplest array signal processing algorithm [8]. The underlying idea is to delay each microphone input by an appropiate time and add them together. In this sense, the audio signal arriving in a determined direction at the array

[1]Please note that the word "conventional" is used to refer to traditional PCM implementations and not to a particular type of algorithm, the conventional (or Barlett) beamformer.

DOI: 10.17648/sobrac-87156

will be reinforced with respect to other signals arriving from other directions and noise. The delay-and-sum algorithm can be implemented in time-domain [9] [10], delaying each microphone input and adding them together; or in frequency-domain using FFT blocks. In this section will be presented the mathematical basis of the frequency-domain implementation methods.

2.1 One-dimensional FFT beamformer

Given an array of M microphones with time-domain outputs and denoting the mth microphone output as $y_m(t)$.

Lets $Y_m(\omega)$ denote the Fourier transform of the mth microphone output $y_m(t)$. Then the spectrum of the delay-and-sum beamformer's output would be

$$Z(\omega) = \sum_{m=0}^{M-1} w_m Y_m(\omega) \exp(-j\omega\Delta_m), \tag{1}$$

where Δ_m is the delay in the mth microphone output $y_m(t)$.

In practice, however, $Y_m(\omega)$ can not be computed because it would require integrating over all time. So, in order to analyze the time-domain signal in a limited time frame, it is introduced the concept of short-time Fourier transform

$$Y_m(t,\omega) = \int_t^{t+D} \tilde{w}(t-\tau) y_m(\tau) e^{-j\omega\tau} d\tau, \tag{2}$$

where D is the time frame after the t instant. Here, $\tilde{w}(t)$ denotes a finite-duration window defined over $[0, D]$. Therefore, Equation 2 can be rewritten as

$$Y_m(t,\omega) e^{j\omega t} = \int_0^{D} \tilde{w}(\tau) y_m(t+\tau) e^{-j\omega\tau} d\tau. \tag{3}$$

This expression can be interpreted as an approximation of the spectrum at time t in a frame of length D. Then, the spectrum of the delay-and-sum beamformer's output will be

$$Z(t,\omega) = \sum_{m=0}^{M-1} w_m Y_m(t,\omega) e^{j\omega t} \exp(-j\omega\Delta_m). \tag{4}$$

Even thought Equation 4 is limited in time, it still require to integrate over the time-domain which is not possible for discrete-time signal processing. Provided that $y_m[n]$ is the nth sample of $y_m(t)$ signal sampled at $f_s = 1/T$ rate so that $t = nT$, Equation 2 can be expressed in the discrete-time-domain as

$$Y_m[n,\omega] = \sum_{l=n}^{n+D_s-1} \tilde{w}[l-n] y_m[l] e^{-j\omega T l}, \tag{5}$$

DOI: 10.17648/sobrac-87156

where $Y_m[n,\omega]$ is the discrete short-time Fourier transform of the $y_m(t)$ at the instant $t = nT$ and over the frame time $D = D_s T$ provided that D_s is an integer. Here, $\tilde{w}[n]$ is also the discrete-time version of the window function $\tilde{w}(t)$ over $[0, D_s]$. Then the discrete short-time Fourier transform of the beamformer's output equals to

$$Z[n,\omega] = \sum_{m=0}^{M-1} w_m Y_m[n,\omega] e^{j\omega T n} \exp(-j\omega\Delta_m)\,. \tag{6}$$

If the frequency-domain is discretized so that $\omega T = 2\pi v/D_s$ for $v = 0,...,D_s-1$, Equation 5 can be rewritten as

$$Y_m[n,v] \exp\left\{ j\frac{2\pi v}{D_s} n \right\} = \sum_{l=0}^{D_s-1} \tilde{w}[l] y_m[n+l] \exp\left\{ -j\frac{2\pi v}{D_s} l \right\}. \tag{7}$$

So the beamformer's output will be

$$Z[n,v] = \sum_{m=0}^{M-1} w_m Y_m[n,v] \exp\left\{ j\frac{2\pi v}{D_s}(n - \Delta_m/T) \right\}. \tag{8}$$

Then, if it is defined $\tilde{y}_m[n,l] = \tilde{w}[l] y_m[n+l]$, the DFT of $\tilde{y}_m[n,l]$ will be

$$\tilde{Y}_m[n,v] = \sum_{l=0}^{D_s-1} \tilde{y}_m[n,l] \exp\left\{ -j\frac{2\pi v}{D_s} l \right\}, \qquad v = 0,...,D_s-1\,. \tag{9}$$

Finally, Equations 7 and 8 can be rewritten as

$$Y_m[n,v] \exp\left\{ j\frac{2\pi v}{D_s} n \right\} = \tilde{Y}_m[n,v]\,, \qquad v = 0,...,D_s-1\,, \tag{10}$$

$$Z[n,v] = \sum_{m=0}^{M-1} w_m \tilde{Y}_m[n,v] \exp\left\{ -j\frac{2\pi v}{D_s}\frac{\Delta_m}{T} \right\}, \qquad v = 0,...,D_s-1\,. \tag{11}$$

Equation 11 can be implemented in hardware as shown in Figure 1. At first, it is required to transform PDM bitstreams $x_m[n]$ to PCM representation $y_m[n]$. Then each PCM audio signal is passed through a windowing function $\tilde{w}[n]$ to yield $\tilde{y}_m[n]$. Each windowed signal is then passed through a FFT block to obtain $\tilde{Y}_m[n,v]$. These frequency-domain signals should be multiplied by a weighting factor $W_m(v) = w_m \exp\left\{ -j\frac{2\pi v}{D_s}\frac{\Delta_m}{T} \right\}$ which depends on the desired direction of arrival. Finally, the weighted outputs are summed together and transformed to time-domain by an inverse Fourier transform (IFFT).

DOI: 10.17648/sobrac-87156

Figure 3a shows the normalized power of an uniform linear array implemented with *one-dimensional FFT beamformer* method using 50 microphones ($M = 50$). Three audio sources of 1 kHz, 3 kHz and 5 kHz are located at 20, 60 and 110 degrees respectively.

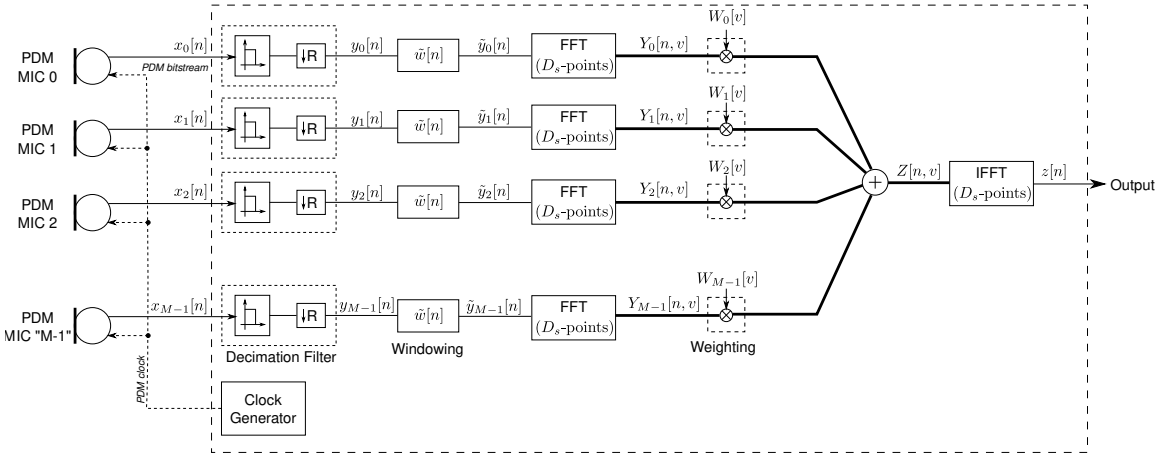


Figure 1: One-dimensional FFT beamformer implementation method.

2.2 Two-dimensional FFT beamformer

In the same way that *one-dimensional FFT beamformer*, if it is used an uniform regular array whose spatial origin occurs at phase begin so that $\Delta_m = \alpha_x md$; where d is the space between microphones, $\alpha_x = \cos(\theta)$ and θ is the angle of arrival; the argument of the exponential in Equation 11 can be written as

$$\frac{2\pi v}{D_s}\frac{\Delta_m}{T} = \frac{2\pi u}{M}m\,, \qquad m = 0,...,M-1\,, \tag{12}$$

where $u = 0,...,M-1$. Thus Equations 12 and 9 can be replaced in Equation 11 as

$$Z[n,u,v] = \sum_{m=0}^{M-1}\sum_{l=0}^{D_s-1} w_m \tilde{y}_m[n,l]\exp\left\{-j\frac{2\pi v}{D_s}l\right\}\exp\left\{-j\frac{2\pi u}{M}m\right\}. \tag{13}$$

Finally, if it is defined $x[n,m,l] = w_m\tilde{y}_m[n,l] = w_m\tilde{w}(l)y_m[n+l]$, Equation 13 can be written as a two dimensional DFT

$$Z[n,u,v] = \mathrm{DFT}\left\{\mathrm{DFT}\left\{x[n,m,l]\right\}\right\}, \qquad m = 0,...,M-1\; l = 0,...,D_s-1\,. \tag{14}$$

Equation 14 can be implemented in hardware as shown in Figure 2. At first, it is required to transform PDM bitstreams $x_m[n]$ to PCM representation $y_m[n]$. Then each PCM audio signal is passed through a windowing function $\tilde{w}[n]$ to yield $\tilde{y}_m[n]$. Each windowed signal is then passed through a FFT block to obtain $\tilde{Y}_m[n,v]$. The outputs of those FFT blocks are passed through another FFT block. Then FFT's output is passed through a steerer block which, given a desired angle of arrival θ, filters only the samples meeting the u and v relation defined by Equation 12. Finally, the steerer's output is passed through an inverse Fourier transform (IFFT) block.

DOI: 10.17648/sobrac-87156

Figure 3b shows the normalized power of a 50-microphones uniform linear array implemented with the *two-dimensional FFT beamformer* method. Three audio sources of 1 kHz, 3 kHz and 5 kHz are located at 20, 60 and 110 degrees respectively. The stepped response shown in the normalized power diagram is due to the small size of the 2nd FFT ($N = 50$) which limits the beamformer resolution. This resolution can be improved increasing N so that it would be always greater or equal than the number of microphones ($N \geq M$).

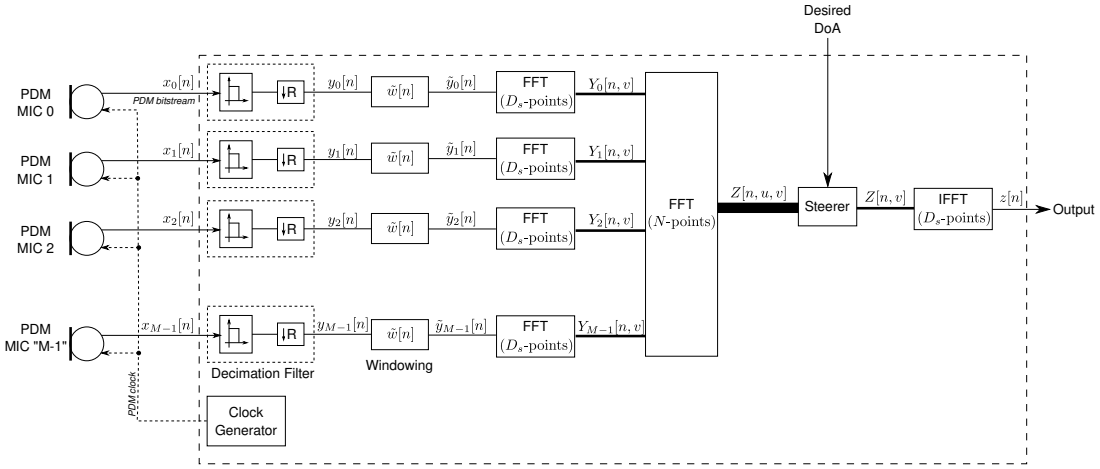
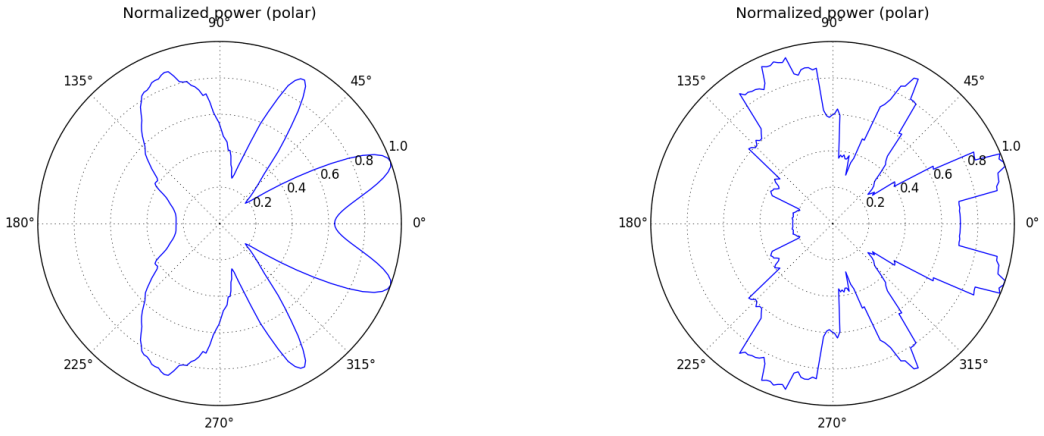


Figure 2: Two-dimensional FFT beamformer implementation method.



(a) One-dimensional FFT method

(b) Two-dimensional FFT method

Figure 3: Normalized power of an uniform linear array of 50 microphones ($M = 50$). Three audio sources of 1 kHz, 3 kHz and 5 kHz are located at 20, 60 and 110 degrees respectively.

3. PROPOSED DELAY-AND-SUM BEAMFORMER IMPLEMENTATIONS

In this section are proposed two frequency-domain methods that do not require decimation filters. These methods are merely described here and results are presented and discussed in Section 4.

3.1 One-dimensional bitstream FFT beamformer

Due to the fact that a PDM bitstream has the same information than its converted PCM signal but with quantization noise shaped at higher frequencies, a PDM bitstreams can be treated as

DOI: 10.17648/sobrac-87156

a baseband signals with a higher sampling rate $f_s' = Rf_s$, where f_s is the required sampling rate in the beamformer's output and R is the decimation rate. Therefore, the *one-dimensional FFT beamformer* can be modified so that decimation would be performed in frequency-domain at the end of the beamformer, before the inverse-FFT is applied, as shown in Figure 4. This implementation method will be called as *one-dimensional bitstream FFT beamformer*.

Because the input sampling rate is higher than in a conventional DAS beamformer, it is required a frame length $D_s' = RD_s$ i.e. R times larger than the required in conventional implementation methods.

Figure 6a shows the normalized power of an uniform linear array implemented with an *one-dimensional bitstream FFT beamformer* method using 50 microphones ($M = 50$). Three audio sources of 1 kHz, 3 kHz and 5 kHz are located at 20, 60 and 110 degrees respectively.

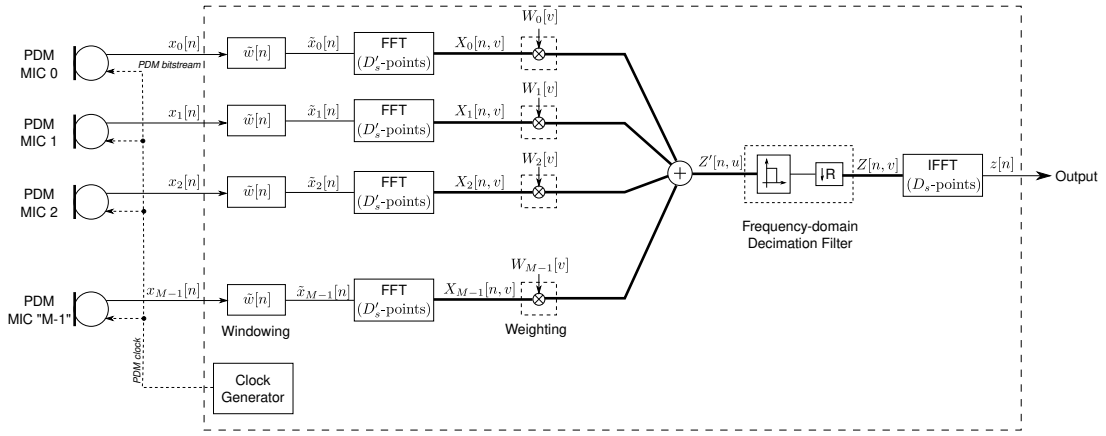


Figure 4: One-dimensional bitstream FFT beamformer implementation method.

3.2 Two-dimensional bitstream FFT beamformer

The *two-dimensional bitstream FFT beamformer* may be derived in the same way than the one-dimensional bitstream FFT beamformer, just performing the decimation filtering in the frequency-domain at the end of the beamformer as shown in Figure 5.

Figure 6b shows the normalized power of an uniform linear array implemented with a *two-dimensional bitstream FFT beamformer* method using 50 microphones ($M = 50$). Three audio sources of 1 kHz, 3 kHz and 5 kHz are located at 20, 60 and 110 degrees respectively.

4. RESULTS

4.1 Performace metrics

The metrics to measure the performance of each implementation method will change depending on either the beamformer is implemented in hardware or software. In software implementation case, the metric to measure the implementation's performance would be the *processing time*; in hardware implementation case though, the metric would be the *hardware logic utilization*.

In this sense, a delay-and-sum beamformer was implemented in hardware and software as a uniform linear array of 8 microphones ($M = 8$) with distance between microphones $d = 8.5$ mm, output sampling rate $f_s = 48$ kHz, decimation filter with a $5th$-order CIC without compensation

DOI: 10.17648/sobrac-87156

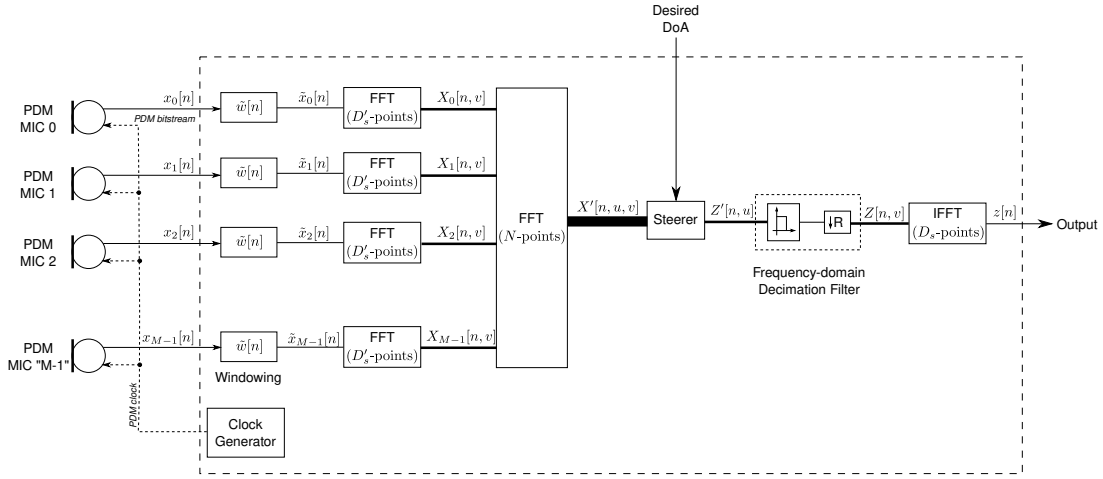
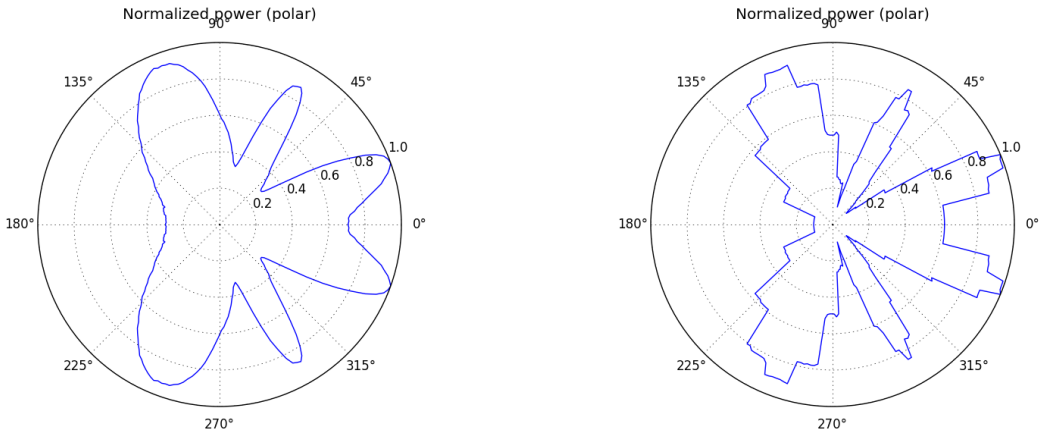


Figure 5: Two-dimensional bitstream FFT beamformer implementation method.



(a) One-dimensional bitstream FFT method

(b) Two-dimensional bitstream FFT method

Figure 6: Normalized power of an uniform linear array of 50 microphones ($M = 50$). Three audio sources of 1 kHz, 3 kHz and 5 kHz are located at 20, 60 and 110 degrees respectively.

filter structure [5], decimation rate $R = 64$, frame length $D_s = 128$ and rectangular windowing.

4.2 Software implementation

The processing time is a metric used to compare methods in a software implementation. However, the processing time would also depend on the beamformer application. So, it was measured the processing time of the beamformers working on the following scenarios:

- *Spatial filtering*, the beamformer is steered at a specific direction-of-arrival, working as spatial filter.
- *Direction-of-arrival detection*, the beamformer is steered sequentially in all directions in order to detect the direction-of-arrival, working as a sonar. In this case, the beamformer was steered in 256 equally spaced points around the 360 degrees range.

Frequency-domain methods were implemented in Python using Numpy and Scipy libraries,

DOI: 10.17648/sobrac-87156

their processing time were then measured on the referred scenarios as shown in Table 1. In this comparison is shown that on the spatial filtering scenario the *one-dimensional bitstream FFT beamformer* has the best performance because decimation is not required in their inputs. However, the *one-dimensional bitstream FFT beamformer* has the worst performance in direction-of-arrival (DoA) detection because FFT needs to be calculated for 256 equidistant points around the 360 degrees range. It is also shown that on the DoA detection scenario the *two-dimensional FFT beamformer* has the best performance even thought this method requires decimation in the sensor inputs. Finally, it is also shown that even thought decimation in the inputs is not required in the *two-dimensional FFT bitstream beamformer*, it does not have a good performace on any of the testing scenarios because of its large FFT processing time.

Table 1: Processing time comparison of one frame of 128-samples length at 48 kHz sampling rate, total frame time of 2.67 ms.

Implementation Method	Input Decimation Filter (ms)	FFT (ms)	Spatial Filtering (ms)	DoA Detection (ms)
1D FFT beamformer	17.21	0.24	18.54	80.87
2D FFT beamformer	17.17	1.59	19.26	25.57
1D bitstream FFT beamformer	-	2.03	6.93	3225.02
2D bitstream FFT beamformer	-	233.85	234.37	240.90

4.3 Hardware Implementation

The beamformer described in Section 4.1 was implemented on a FPGA using conventional and proposed methods. Table 2 summarizes the hardware logic utilization on each implementation of the mentioned beamformer. In this comparison is shown that the *one-dimensional* and *two-dimensional FFT beamformer* implementation methods require fewer resources than the *one-dimensional* and *two-dimensional bitstream FFT beamformer* methods, the last ones require 60% more hardware logic. It is shown also that even though the *one-dimensional* and *two-dimensional bitstream FFT beamformer* methods do not require decimation filters in the input, they have more hardware logic utilization because their FFT logic is almost the double.

Table 2: Comparison of hardware logic utilization. All quantities are expressed in number of hardware logic modules.

Implementation Method	Decimation Filter	1st FFT	2nd FFT	Weighting	Steerer	IFFT	Total
1D FFT beamformer	3038.0	22463.2	-	768.0	-	2807.9	29077.1
2D FFT beamformer	3038.0	22463.2	2968.5	-	96.0	2807.9	31373.6
1D bitstream FFT beamformer	-	47175.2	-	768.0	-	2807.9	50751.1
2D bitstream FFT beamformer	-	47175.2	2968.5	-	96.0	2807.9	53047.6

DOI: 10.17648/sobrac-87156

5. CONCLUSIONS

This article analyzes the conventional beamformer implementation methods and proposes new approaches as alternatives to them. At first, both conventional and proposed methods were implemented in software; then they were implemented in hardware, specifically in a FPGA board.

It has been shown that the *one-dimensional bitstream FFT beamformer* method being implemented in software for spatial filtering scenarios has better processing time even though its hardware implementation is more costly. Also it has been shown that the conventional *two-dimensional FFT beamformer* is the most efficient software implementation for direction-of-arrival detection applications.

Finally, even though this research has been focused to analyze the different possible implementation of delay-and-sum beamformers, the results are also valid for others frequency-domain beamformers such as Capon's beamformers [11].

References

[1] J.M. de la Rosa and R. Río. *CMOS Sigma-Delta Converters: Practical Design Guide*. Wiley - IEEE. Wiley, 2013. ISBN 9781118568439.

[2] L. Milic. *Multirate Filtering for Digital Signal Processing: MATLAB Applications: MATLAB Applications*. Premier reference source. Information Science Reference, 2009. ISBN 9781605661797.

[3] Julius O. Smith. *Introduction to Digital Filters with Audio Applications*. W3K Publishing, 2007. ISBN 978-0-9745607-1-7.

[4] Jon Dattorro. The implementation of recursive digital filters for high-fidelity audio. *J. Audio Eng. Soc*, 36(11): 851–878, 1988.

[5] Eugene Hogenauer. An economical class of digital filters for decimation and interpolation. 29:155 – 162, 05 1981.

[6] Don H. Johnson and Dan E. Dudgeon. *Array Signal Processing: Concepts and Techniques*. Simon & Schuster, Inc., New York, NY, USA, 1992. ISBN 0130485136.

[7] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009. ISBN 0131988425, 9780131988422.

[8] H. Krim and M. Viberg. Two decades of array signal processing research: the parametric approach. *IEEE Signal Processing Magazine*, 13(4):67–94, Jul 1996. ISSN 1053-5888. doi: 10.1109/79.526899.

[9] Jelmer Tiete, Federico Domínguez, Bruno Silva, Laurent Segers, Kris Steenhaut, and Abdellah Touhafi. Soundcompass: A distributed mems microphone array-based sensor for sound source localization. *Sensors*, 14 (2):1918–1949, Jan 2014. ISSN 1424-8220. doi: 10.3390/s140201918.

[10] Kim Youngkey, Kang Jungoo, and Lee Myunghan. Developing beam-forming devices to detect squeak and rattle sources by using fpga. In *Inter-noise*, pages 1–6, 2014.

[11] J. Capon. High-resolution frequency-wavenumber spectrum analysis. *Proceedings of the IEEE*, 57(8): 1408–1418, Aug 1969. ISSN 0018-9219. doi: 10.1109/PROC.1969.7278.

DOI: 10.17648/sobrac-87156