Hybrid Algorithm Computation Methodology to Accelerate Sound Source Localization

Christian Ibala, Fernando Escobar, Xin Chang, and Carlos Valderrama

***Abstract*—Latest algorithm solutions for sound source localization have dramatically increased their computation burden. This paper proposes a novel hybrid algorithm with the ability to adapt its search spectrum based on target movement; we also show some studies and metrics to analyze its implementation constraints and remark its advantages over previous solutions. The proposed algorithm provides a good balance between processing power and real-time execution; our approach combines the General Cross Correlation (GCC) with the Delay and Sum Beamforming (DSB) algorithm, such that, less than 50% of the DSB computation is necessary to locate the sound source.**

***Index Terms*—DSB, GCC, algorithm interface, localization, FPGA**

I. Introduction

The auditory system of living creatures provides a vast amount of information about their environment, which allows them to perform, for instance, sound source localization [1].

Using several closely positioned microphones, that is, a microphone array, it is possible to listen to sound coming from one specific direction, while reducing noise and interference sound coming from others. This signal processing technique is called beamforming. Rapid advancement in adaptive beamforming applications such as sonar and radar algorithms has greatly increased the computation and communication demands on beamforming arrays; this is particularly important for applications that require autonomous and real-time execution [2]. Parallel processing applied to beamforming not only reduces execution time, power consumption and costs but also increase scalability and dependability [3]. In most cases, a parallel architecture guarantees the highest throughput at the cost of bigger area and resource utilization. Architecture complexity increases with requirements such as flexibility and adaptability. Amdahl's law provides more details on parallelism constraint [4]; however, the primary determinant of performance nowadays is power efficiency, rather than insufficient area or frequency [5]. In general, power efficiency can be achieved by reducing either switching activity or resource utilization [6]. Other approaches to improve power consumption are discussed in [7, 8].

Since sequential implementation of beamforming algorithms with many microphones presents a significant computational challenge in real-time processing, our contributions in this work can be summarized as follows:

We combine the advantages of two well-known algorithms: the GCC (Generalize Cross Correlation) and the DSB (Delay and Sum Beamforming). On one hand, the DSB algorithm is easy to implement and accurate for localization, but requires large computational power; however, the GCC is a faster algorithm, capable of providing, at least, the direction of arrival (angle) of the sound source. By using this angle we can restrain the DSB search area to accelerate its computation.

Consider Figure 1 where there is an example of a FOV for a sound source located at 2.5 meters above the microphone array at 45 degrees from the center. By using our hybrid algorithm the sound source is located faster since the FOV is reduced, as shown in Figure 2. The scale at the right side of Figure 1 is the Steered Response Power (SRP) which will be detailed later.

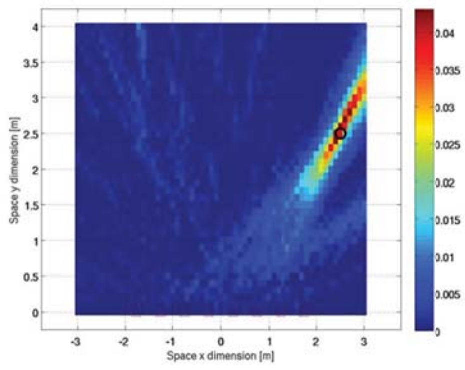


Figure 1. Steered Power Response (SRP) with one sound source, obtained with the DSB algorithm.

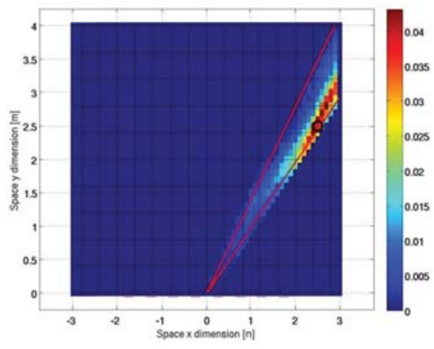


Figure 2. Steered Power Response (SRP) with one sound source, obtained with the GCC-DSB algorithm.

C. Ibala is with Department of Electronics and Computer Engineering, University of Limerick, Ireland (e-mail: sibala@acm.org)

F. Escobar, X. Chang and C. Valderrama are with Electronics and Microelectronics Department, Faculty of Engineering, University of Mons, Belgium (e-mails: {fernando.escobar,carlos.valderrama}@umons.ac.be)

Copyright © 2012 by Department of Microelectronics & Computer Science, Technical University of Lodz

The remainder of this paper will be organized as follows: In Section II we discuss the specification of the hardware used to capture the signals. Section III presents an explanation of the beamforming algorithms. In Section IV, the algorithm's workload is explained and profiled and compared. In Section V we formulate the problem and explain our contribution. In Section VI we discuss the statisctics obtained; we also show full plots of the algorithm's profiles and present an estimate of resource utilization when implementing the DSB algorithm. In Section VII we advice on further work and conclude this one.

II. Hardware Acquisiton System

Although the microphone's characteristics do not have any direct implication in the computational complexity of a beamforming algorithm, they play a crucial role in the architecture requirements, in terms of interfacing and throughput. We briefly describe our system's specifications.

A. Microphone Characteristics

The microphone is a sensor for capturing the small changes in air pressure (sound) and converting them into an electrical signal [9]. The microphones used to record signals are the Miniature Electro-Mechanical System Microphone (MEMS). For instance, the ADMP421 MEMS [10, 12] is an omni-directional microphone, that is, equally sensitive to sound coming from all directions regardless of its orientation. Figure 2.a shows the functional block of the ADMP421. It includes an Analog to Digital Converter (ADC) for signal sampling at a rate of 48 KHz and generates a 2.4MHz Sigma-Delta (Δ-Σ) fourth-order, digitally modulated data by using a Pulse Density Modulator (PDM). The Power Management and the Channel Select are both hot swapping modules. A sub-cardioid response microphone shown in Figure 2.b would be enough to reproduce this work.

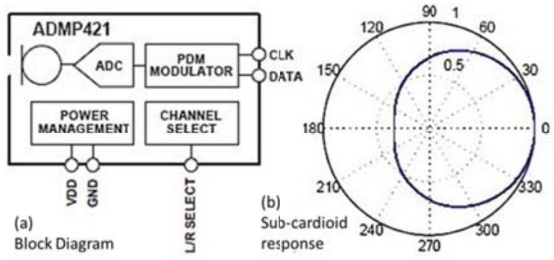


Figure 2. ADMP421 micro-MEMS.

B. Acquisition Hardware Specification

Figure 3 represents the overall acquisition and processing system used. Instead of a Digital Signal Processor, a Field Programmable Gate-Arrays (FPGA) is used providing the required flexibility to implement the algorithms while supporting MEMS scalability and data synchronization. On the application software side, the FTDI (Future Technology Device international) [11] is a USB to Parallel FIFO interface, connects the FPGA to a PC. The FTDI is only necessary for signal integrity verification purpose. Therefore the data rate through the USB has not implication to our application.

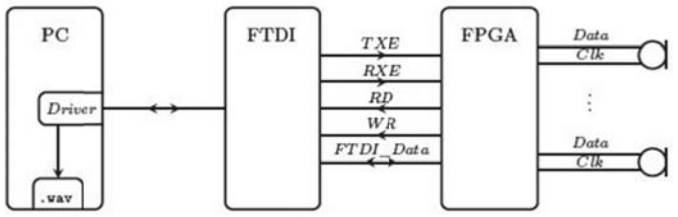


Figure 3. Acquisition sytem block diagram.

The FPGA front-end requires a Δ-Σ demodulator to recover data coming from each micro-MEM at a frequency of 2.4 MHz. Demodulation can be perfomed by using a FIR (Finite Impulse Response), CIC (Cascaded Integrator Comb) or Kalman filter, among others [13]. In this work, a CIC filter is employed. It achieves downsampling (decimation) without using complex hardware structures such us multipliers [14]. A downsampling factor allows reducing sampling frequency from 2.4 Mhz to 48 KHz.

III. Sound Source Localization and Beamforming Principles

A. Beamforming Principle

One of the most important functionalities of microphone arrays is to extract the speech of interest from its observation corrupted by noise, reverberation, and competing sound sources. This is done by aiming the beam towards the desired sound source [16].

Beamforming's basic idea is to sum up the contribution of each microphone; as a result, signals from this so called "look-direction" are reinforced while signals from all the other directions are attenuated [16].

The response of a linear array of N sensors, with a uniform inter-element spacing d, is known as directivity pattern D. D is a function of the direction ϕ and frequency f. The far-field directivity pattern is given by equation (1):

$$D(f,\phi) = \sum_{n=0}^{N} w_n(f) \cdot e^{\frac{j2\pi nd\cos\phi}{\lambda}} \tag{1}$$

- $w_n(f)$ is a complex weight associated to the n^{th} sensor;
- ϕ being the angle measured from the array axis in the horizontal plane;
- λ is the wavelength;
- $r = |r|$ is the radial distance from the sound source to the microphone aperture;
- f is the working frequency;
- d is the distance between two consecutive microphones.

The far-field directivity pattern applies to planar wave fronts respecting equation (2), with |r| as the sound source minimum distance to the sensors.

$$|r| > \frac{2(Nd)^2}{\lambda} \tag{2}$$

A simple horizontal directivity for equally weighted sensors $w_n(f) = 1/N$ is shown by the bold line of Figure 5, illustrating the directional nature of the array response. From the directivity pattern, we see that the sensor array is capable of enhancing a signal arriving from a certain direction with respect to signals arriving from all other directions.

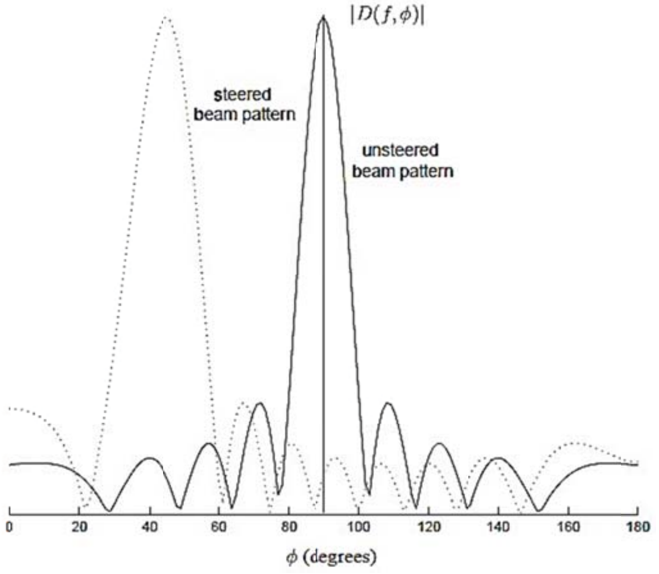


Figure 5. Unsteered and steered (ϕ=45°) directivity patterns (f = 1kHz, N = 10, d = 0.15m) [19].

In general, the complex weighting $w_n(f)$ can be expressed by its amplitude and phase as shown by equation (4):

$$w_n(f) = a_n(f) \cdot e^{j\phi_n(f)} \tag{3}$$

where $a_n(f)$ and $\phi_n(f)$ are frequency dependent amplitude and phase. By modifying the amplitude $a_n(f)$ we can modify the shape of the directivity pattern. Similarly, by modifying the phase $\phi_n(f)$, we can control the angular location of the response main lobe. Thus, the response of the array can be controlled to enhance the signal arriving from a specific direction.

B. Sound Localization Algorithms

There are two major groups of microphone-array processing algorithms: time-invariant and adaptive [9]. The first group is fast and simple to get a real-time implementation. Acoustic adaptive algorithms are able to automatically adapt their response to different weightings or time-delays, however, they require more CPU power and are complex to implement. Thus, only the DSB and GCC-PHAT will be used for this work.

1) GCC-PHAT

The Generalized Cross Correlation (GCC) algorithm returns an angle ϕ which is the sound source direction of arrival (DOA). To compute ϕ, the GCC uses the estimation of the temporal shift between two microphones that lead to the maximum cross-correlation function between them as in equation (4):

$$\Delta_{ij} = arg_k \max R(k) \tag{4}$$

The cross correlation between two microphones is computed by taking the inverse Fourier transform of the product of each microphone's FFT (Fast Fourier Transform) and the conjugate of the other's FFT as in equation (5).

$$R(k) = IFFT\ \left(FFT\big(f(t)\big).FFT^*\big(g(t)\big)\right) \tag{5}$$

$FFT\big(f(t)\big)$ is the Fourier transform of the signal at the microphone i, and $FFT^*\big(g(t)\big)$ is the Fourier transform conjugate of the signal at the microphone j. In order to correct the effect of phase, and improve robustness against noise and other undesired effects, there is a correction called *PHAT,* i.e. phase transform that can be applied. Equation (5) then becomes:

$$R(k) = IFFT\left(\frac{FFT\big(f(t)\big).FFT^*\big(g(t)\big)}{\left|FFT\big(f(t)\big).FFT^*\big(g(t)\big)\right|^{\beta}}\right) \tag{6}$$

Equation (6) is defined as the GCC-PHAT; β is a coefficient factor in the interval of]0, 1[. The IFFT is performed to return back to time domain and extract the corresponding value of index k. The value of k can be computed by taking the index of the maximum value from the GCC-PHAT. Using the far field approximation, the cosine of the angle of arrival, measured by microphones i and j can be computed as follows:

$$\cos\phi_{ij} = \frac{k_{ij}.v_s}{f_s.d_{ij}} \tag{7}$$

wheref_s is the sampling frequency, d_{ij} is the distance between microphone i and j, and v_s is the sound speed. Because every pair of microphones can provide one angle, the results from them can be combined as:

$$\phi = \frac{1}{C_N^2}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\phi_{ij} \tag{8}$$

Note that the GCC computation load is associated to the number of cross-correlation it needs to compute. That number is modeled by equation (9). P takes the value of 2 and N is the number of microphones:

$$C_N^P = \frac{N!}{P!\,(N-P)!} \tag{9}$$

2) DSB - SRP (Time domain approach)

The DSB is a beamforming algorithm which uses a predefined Field Of View (FOV) and resolution to compute its Steered Response Power (SRP), which will be explained below. An FOV is the region in the space where the sound source is susceptible to be found and the resolution is the measurement of the smallest distinguishable region, as shown in Figure 6.

The FOV size and shape is application dependent. Figure 6 shows an example of an FOV region of size 150x150 cm^2 split into 9 small squares of size 50x50 cm^2. The number of small squares (NOSS) is related to the FOV resolution as defined as in equation (10)

$$NoSS = \frac{FOV}{Resolution} = \frac{L \cdot H}{\Delta x \cdot \Delta y} \tag{10}$$

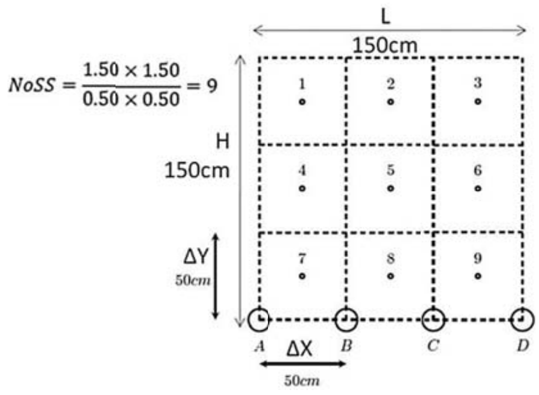


Figure 6. 2D 3x3 FOV with 150cm x 150cm size and a 50cm x 50cm resolution.

The point in the FOV with the highest Steered Response Power (SRP) is where the sound source is located. Before computing the SRP, the signal amplitude must be corrected according to its phase by applying a β-PHAT operation. The DSB β-PHAT is modeled by equation (11).

$$W(f) = \frac{X(f)}{|X(f)|^{\beta}} \tag{11}$$

X(f) is the signal spectrum and β is a constant in the interval]0, 1[. if $\beta = 1$ the magnitude influence is totally removed. The modified spectrum W(f) will then be used as an input to the IFFT using equation (12) before computing the SRP.

$$w(n) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} W(f)e^{j2\pi ft}\,\mathrm{df} \tag{12}$$

This work propose that IFFT module be implemented using the FFT architecture by inverting the imaginary and real part as shown in Figure 7. This computation methodology increase modules re-usability. The FFT module will be detailed later.

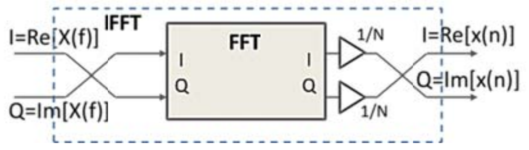


Figure 7. Computing the IFFT with FFT processing element.

The SRP at the point i (with $0 < i < N_{OSS}$) is defined as in equation (13).

$$SRP_i = \sum_{t=1}^{Ns}[(\sum_{n=1}^{N} w_{in}x_n(t-\tau_{in}))^2 - \sum_{n=1}^{N} w_{in}^2 x_n^2(t-\tau_{in})] \tag{13}$$

where $x_n(t)$ is the sample recorded by the n_{th} microphone at time t, w_{in} is the weight of the point i relative to microphone n and is computed as in equation (14).

$$w_{in} = \frac{1/d_{in}}{\sum_{n=1}^{N} 1/d_{in}} \tag{14}$$

d_{in} is the distance of the i^{th} small square (SS) to the n^{th} microphone. It is computed as in (15):

$$d_{in} = \sqrt{(x_i - x_n)^2 + (y_i - y_n)^2} \tag{15}$$

The pairs (x_i, y_i) and (x_n, y_n) are respectively the coordinates of point i and the microphone n.

3) DSB-SRP (Frequency domain approach)

Another approach to compute the SRP using the principle of *Delay and Sum Beamforming* is encountered in the frequency domain. Since the GCC-PHAT algorithm provides a cross-spectrum between the signals from two microphones, it is expected that this value is highest when the signals are in phase i.e. at their corresponding delay index on the GCC-PHAT output.

The difference between this approach and the previous one is that, for each point in the FOV, it is only required to compute the theoretical delay between all pairs of microphones; using this index, the corresponding energy value in the cross-spectrum output (GCC-PHAT) is extracted. The energy from all pairs of microphones is extracted using theoretical delays and added up for each point in the FOV.

Compared to the time-domain approach, this algorithm skips the steps from equation (11) to (14) which have a huge computational cost.

If d_i and d_j are the distances from point p to microphones i and j, we denote d_{ij} as the difference between them. The theoretical arrival delay between them is given by equation (16):

$$\tau_{ij} = \frac{d_{ij} * f_s}{v_s} \tag{16}$$

Bearing in mind equation (6), the SRP is therefore computed as in equation (17)

$$SRP_i = \sum_{k=1}^{N}\sum_{j=k+1}^{N} R(\tau_{kj})\Big|_{k \neq j} \tag{17}$$

IV. Acquisition Block Diagram and Algorithm Computational Comparison

Figure 8 shows a proposed block diagram to implement and specially analyse both the GCC and DSB algorithms in terms of their computational complexity. The branch where the output is an "incidence angle", represents the GCC and the one with "source localization" output represents the DSB.

Both algorithms share the demodulator (Δ-Σ), framing, Voice Activity Detector (VAD). They also share the Fast Fourier Transform (FFT) module, since it is required for the GCC and for the β-PHAT operation in DSB as shown in Equation (11). Each module will be presented briefly before introducing our hybrid algorithm.

For the sake of clarity and comparison, Figure 9 shows a block diagram of the DSB-SRP algorithm; it is important to not that this algorithm skips the computation of equations (13) and (14), but requires the execution of the GCC-PHAT algorithm first. As will be seen later in the paper, our proposal profits from the GCC-PHAT angle detection to reduce the FOV and then compute the SRP.

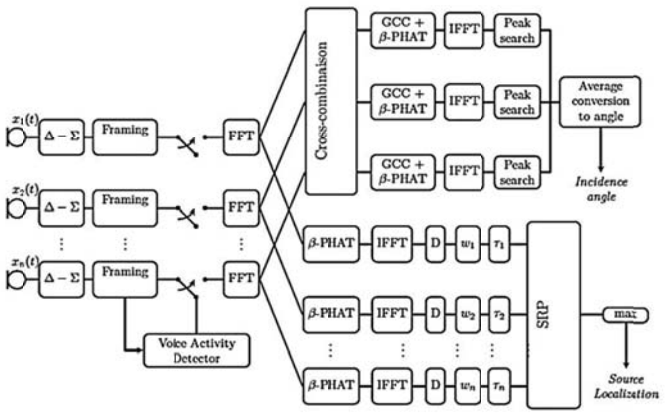


Figure 8. GCC (upper branch) and DSB-Time domain (lower branch) functional block diagrams. Both algorithms can be computed in parallel.

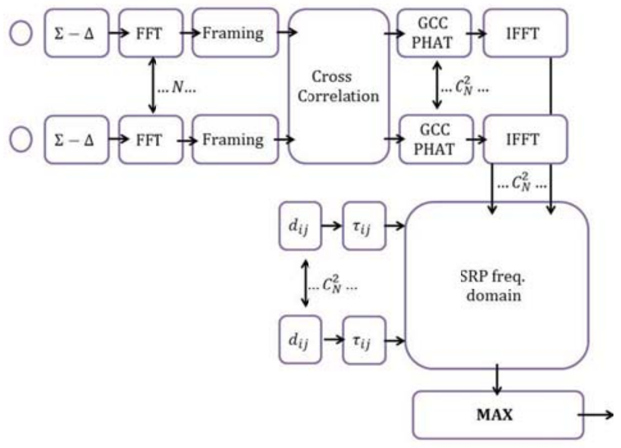


Figure 9. DSB algorithm in the Frequency Domain. A whole GCC is required before starting the computation of the *Steered Power Response*.

The demodulated CIC signal is transmitted to the framing; then, the VAD modules which decide if the content of the signal is a noise or a sound source to trigger the start of the localization algorithms. Otherwise that frame is ignored as shown by the switch in Figure 8.

A. Framing and Voice Activity Detector

The framing module cuts the audio signal in frames of (256 or more samples) each sample being 2 bytes long. The VAD module is used to compute noise mean and variance over the 10 first frames. These values are used to determine the mean (18), variance (19) and threshold (20) equations, each incoming frame are compared to that threshold. If the computed value is lower than the threshold we assume that we are in presence of a noisy frame; otherwise there is a sound source.

$$\mu = \frac{\sum_{i=0}^{Ns} x_i}{Ns} \tag{18}$$

$$\sigma^2 = \frac{\sum_{i=0}^{Ns} x_i^2}{Ns} - \mu^2 \tag{19}$$

where μ is the mean, σ^2 is the variance, Ns is the number of samples in the frames, x_i is the value of sample i. Sound detection threshold can be computed as in equation (20) (where cst is a constant taken with a value≥ 3):

$$N_{tresh} = \mu + cst \cdot \sigma \tag{20}$$

B. FFT Implementation Algorithm

After detecting a sound source in the signal frame, the VAD triggers FFT computation which is modelled by equation (21).

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-j2\pi f t} dt \tag{21}$$

where x(t) and X(f) are the signal input and output and f the frequency. For computation speed, a Fast Fourier Transform (FFT) which is a fast DFT algorithm that reduces the computing burden from N^2 to $\mathrm{N} \cdot \log_2 \mathrm{N}$ is used. Since FFT processors using a radix-4 architecture have fewer multiplications than processors using radix-2 [15], they are prefered in order to reduce the memory access rate and arithmetic workload, hence, power consumption.

It is important to remark a few assumptions that have been adopted for this work; on one hand, a far-field approximation is used. (cf equation (2)), and the microphone array is linear.

As the sound source is far enough from the microphone array, the difference between the signal received by the n^{th} microphone $\mathrm{x_n}$ and the center of the array is a pure delay [17]. That delay in time units can be expressed as in equation (22):

$$\tau_n = \frac{f_s \cdot d_n \cos\phi}{c} \tag{22}$$

where f_s is the sample frequency, d_n is the extra distance travelled by the sound wave to reach the n^{th} sensor compared to the reference sensor, ϕ is the wave front incident angle, and c being the sound speed.

- No multipath contribution is taken into account.
- The only noise sources are the microphones themselves and stationary noise.

C. GCC-PHAT Compuational Load

Because the cross-correlation is performed between pairs of microphones, then, from equation (9), P equals 2 and N the number of microphones in the array e.g. 4, 8, 16, 32, 64. For the previous list, the number of cross-correlation will vary as described by $\mathrm{C}_{\mathrm{N}}^{\mathrm{P}}$, e.g. 6, 28, 120, 496, 2016. This leads to a CPU power increase with the number of microphones. To reduce the CPU power requirements with the same amount of microphones, the configuration can be changed from one microphone array to a two smaller microphone sub-arrays of equal amount of units. The number of cross-correlations will then respectively vary as 2, 12, 56, 240, 982 which represents more than 50% computation reduction at the cost of 2 to 3 degree loss of accuracy of the sound source localization, according to our tests.

We analyzed the blocks listed in TABLE I from Figure 8; TABLE I shows the computation burden and the number of sequential operations (memory access, addition, multiplication, division and square root) required to obtain the direction of arrival of the sound source denoted above as ϕ. We include all operations required by equation (6), (7) and (8), except for the IFFT, which is performed in both algorithms as shown in Figure 8. This computation was perform with the audio frames of $N_s = 512$ samples and with N = 4 microphones.

TABLE I
GCC COMPUTATION BURDEN

	GCC + PHAT ($\beta = 1$) + Peak Search + Angle detection	
Blk_{read}	$(C_n^p \cdot N_s) + C_n^p \cdot N_s$	6144
$Mult$	$(C_n^p \cdot N_s \cdot complex) + (2C_n^p \cdot N_s) + (1)$	18433
Add	$(2 \cdot N_s) + (2 \cdot C_n^p \cdot N_s) + (C_n^p \cdot N_s) + (C_n^p - 1)$	10245
Div	$(2 \cdot C_n^p \cdot N_s) + (1)$	6145
$Sqrt$	$(C_n^p \cdot N_s)$	3072

Assuming that all operations are performed in one clock cycle except the division and square root (8 clock cycles using Xilinx IP Cores), the number of clock cycles per frame (NCCF) is given by equation (23).

$$NCCF = Mult + Add + Blk_{read} + 8.(Div + Sqrt) \qquad (23)$$

Using equation (23) and TABLE I the throughput of the GCC with clock running at certain speed can be computed. With $\mathrm{C}_{\mathrm{N}}^{\mathrm{P}}$ being the total number of cross-correlations, TABLE II gives the GCC throughput computed per frame sequentially. Parallelized operations of TABLE I could reduce the GCC throughput.

TABLE II
GCC SEQUENTIAL THROUGHPUT

GCC (Clock Speed)	200 MHz	400 MHz	600 MHz
Throughput	0.55 ms	0.28 ms	0.14ms

D. DSB Compuational Load

The DSB algorithm combines accurate sound source localization with the flexibility of having pre-computed coefficients if necessary. Those coefficients (weight and delay) could then be stored in an FPGA BRAM or in an external memory.

TABLE III presents the computation weight of the DSB for different value of the *NOSS*; here, a purely sequential execution is assumed.

TABLE III
DSB COMPUTATION BURDEN

	Localization	$N_{oss} = 256$	$N_{oss} = 32$	$N_{oss} = 25$	$N_{oss} = 12$
Blk_{read}	$[(N) + (N_s \cdot N) + (N)] \cdot N_{oss} + N_{oss} + [(2N_s \cdot N)] \cdot N_{oss}$	1575168	196896	153825	73836
$Mult$	$[(N_s \cdot N) + (N_s \cdot N) + (N_s) + 1] \cdot N_{oss} + [(2N_s \cdot N)] \cdot N_{oss}$	2228480	278560	217625	104460
Add	$[(N_s \cdot (N-1)) + (N_s \cdot (N-1)) + (N_s) + (N_s)] \cdot N_{oss} + [(N_s \cdot N)] \cdot N_{oss}$	1572864	196608	153600	73728
Div	$[2N_s \cdot N] \cdot N_{oss}$	1048576	131072	102400	49152
$Sqrt$	$[(N_s \cdot N)] \cdot N_{oss}$	524288	65536	51200	24576

TABLE IV shows the DSB throughput for a NoSS = 256 at different clock speeds.

TABLE IV
DSB SERIAL THROUGHPUT WITH NOSS = 256

DSB (Clock Speed)	200 MHz	400 MHz	600 MHz
Throughput	90 ms	45 ms	30 ms

For a sound source localization using a sampling frequency of 44.1 KHz and a size frame of 512, to achieve a real-time application, the localization must be performed in less than 11.6 ms. TABLE IV shows that from 200 MHz to 600 HMz the throughput time is superior to 11.6 ms. Based on that challenge this work will present an hybrid algorithm to reduce the NOSS number.

The GCC computation burden only depends on the number of microphones whereas in the DSB, it's dependent on the number of microphones, the FOV size and resolution. Multiple sound source tracking is possible with DSB but not with GCC. Globally we can say that the GCC algorithm is faster than the DSB, but the localization made with the DSB is more accurate (GCC just provides the angular direction). A possible way to improve the localization using GCC is to use two sub microphone arrays. But this method is error sensitive, because a small error on the estimation of both angles, results in a false estimation of the sound source position.

V. Contribution

Our contribution in this work is to accelerate DSB computation by using the GCC. To illustrate the problem we will use a 4-microphone array as represented in Figure 10. The search region limited by the angles $\phi \pm \varepsilon$ can be obtained by applying first the GCC algorithm, thus reducing the search spectrum and the number of DSB computations for the same resolution.

The basic idea of this approach is to create a reduced detection zone by drawing two lines; one above and the other below the GCC detected angle with an inclination $\pm\varepsilon$ chosen by the user Figure 10 shows one case of this approach, the remainders are mathematically detailed below. Our algorithm can be described as follows:

For every angle ϕ returned by the GCC, we will assume that:

$$0 \le \phi \le 180$$

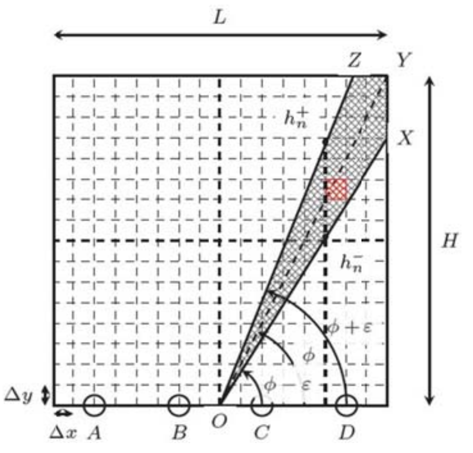


Figure 10. FOV of the GCC-DSB.

There are two special case angles, which are the upper border of the FOV. These two angles denoted as δ_1 and δ_2 are defined as follows:

$$\delta_1 = \arctan\left(\frac{H}{L/2}\right)$$

$$\delta_2 = 180° - \arctan\left(\frac{H}{L/2}\right)$$

- **If $(\boldsymbol{\phi} - \boldsymbol{\varepsilon} \le 0)$,the FOV is delimited by:**
$$\{0{,}0\};\left\{\left(\tfrac{L}{2}\right),0\right\};\left\{\left(\tfrac{L}{2}\right),\left(\tfrac{L}{2}\cdot\tan(\phi+\varepsilon)\right)\right\};$$

- **If $(\boldsymbol{\phi} - \boldsymbol{\varepsilon} > 0$ and $\boldsymbol{\phi} + \boldsymbol{\varepsilon} \le \ddot{a}_1)$ the FOV is:**
$$\{0{,}0\};\left\{\left(\frac{L}{2}\right),\left(\frac{L}{2}\cdot\tan(\phi-\varepsilon)\right)\right\};\left\{\left(\frac{L}{2}\right),\left(\frac{L}{2}\cdot\tan(\phi+\varepsilon)\right)\right\}$$

- **If $(\boldsymbol{\phi} - \boldsymbol{\varepsilon} < \ddot{a}_1$ and $\boldsymbol{\phi} + \boldsymbol{\varepsilon} > \ddot{a}_1)$ the FOV is:**
$$\{0{,}0\};\left\{\left(\tfrac{L}{2}\right),\left(\tfrac{L}{2}\cdot\tan(\phi-\varepsilon)\right)\right\};\left\{\tfrac{L}{2},H\right\};\{H\cdot\operatorname{cotan}(\phi+\varepsilon)\,,H\}$$

- **If $(\boldsymbol{\phi} - \boldsymbol{\varepsilon} > \ddot{a}_1$ and $\boldsymbol{\phi} + \boldsymbol{\varepsilon} < 90°)$ the FOV is:**
$$\{0{,}0\};\{H\cdot\operatorname{cotan}(\phi-\varepsilon)\,,H\};\{H\cdot\operatorname{cotan}(\phi+\varepsilon)\,,H\}$$

- **If $(\boldsymbol{\phi} - \boldsymbol{\varepsilon} < \ddot{a}_2$ and $\boldsymbol{\phi} + \boldsymbol{\varepsilon} > \ddot{a}_2)$ the FOV is:**
$$\{0{,}0\};\{-H\cdot\operatorname{cotan}(\phi-\varepsilon)\,,H\};\left\{\left(\tfrac{-L}{2}\right);H\right\}\left\{\left(\tfrac{-L}{2}\right),\left(\tfrac{-L}{2}\right)\cdot\tan(\phi+\varepsilon)\right\}$$

- **If $(\boldsymbol{\phi} - \boldsymbol{\varepsilon} > \ddot{a}_2$ and $\boldsymbol{\phi} + \boldsymbol{\varepsilon} < 180°)$ the FOV is:**
$$\{0{,}0\};\left\{\left(\tfrac{-L}{2}\right),\left(\tfrac{-L}{2}\right)\cdot\tan(\phi-\varepsilon)\right\};\left\{\left(\tfrac{-L}{2}\right),\left(\tfrac{-L}{2}\right)\cdot\tan(\phi+\varepsilon)\right\}$$

-

- **If $(\boldsymbol{\phi} - \boldsymbol{\varepsilon} < 180°$ and $(\boldsymbol{\phi} + \boldsymbol{\varepsilon} > 180°)$ the FOV is:**
$$\{0{,}0\};\left\{\left(\tfrac{-L}{2}\right),\left(\tfrac{-L}{2}\right)\cdot\tan(\phi-\varepsilon)\right\};\left\{\left(\tfrac{-L}{2}\right),0\right\}$$

The number of test cases cover above is dependent of the angle range cover by ϕ and the precision (Ɛ) of the GCC localization.

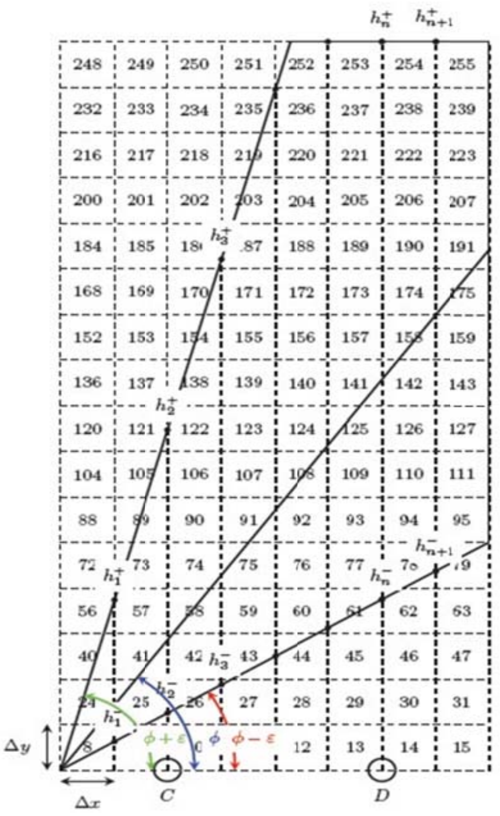


Figure 11. Region search description.

A. Memory Region Access

The FPGA internal memory structure can be read by implementing a simple counter. Figure 11 shows that the numbers encapsulated in the cone between h_n^- and h_n^+ cannot be determined easily. The accesses to those values in our hybrid algorithm are complex and governed by equations (24) and (25).

$$h_n^- = n \cdot \Delta x \cdot \tan(\phi - \varepsilon) \quad (24)$$

$$h_n^+ = n \cdot \Delta x \cdot \tan(\phi + \varepsilon) \quad (25)$$

The h_n^- and h_n^+ are the lowest and highest line, they represents respectively the lines passing by the points {O,X} and {O,Z} in Figure 10.

In Figure 11 0the region between these two points h_n^- and h_n^+ represents the addresses that need to be computed by algorithm 1 using equation (23) and (24) [21]. Every Small Square of the FOV has a number assigned to it. With that number additional information such as his weight and his shift can be retrieved. In our approach the GCC is computed in middle of the microphone array FOV. Besides the GCC computation, two correctives factors, X_{offset} and Y_{offset} are required. Note that algorithm 1 shows only the steps required when ϕ < 90°. The Division by Δx and Δy in Algorithm 1 shows that the unitary resolution or a power of 2 will be a better choice to avoid costly division implementation.

```
if ($\varphi$ <90)
    $cnt\_wr\_blk = 0$;
    for($i = 0; i = i + \Delta x; i \le \frac{L}{2\Delta x} - 1$)
        $h_n^+ = (i+1) \cdot k_1$;
        if ($h_n^+ > H$)
            $max_j = \frac{H - \text{ceil}(h_n^-)}{\Delta y}$;
        else
            $max_j = (i+1) \cdot k$;
        for($j = 0; j = j + \Delta y; j < \text{ceil}(max_j)$)
            $ptx = i\Delta x$;
            $pty = j\Delta y + ik_2$;
            $nb = ptx + x_{offset} + [\text{floor}(\frac{pty}{\Delta y})] \cdot y_{offset}$;
            $cnt\_wr\_blk = cnt\_wr\_blk + 1$;
else
    for($i = 0; i = i + \Delta x; i \le \frac{L}{2\Delta x} - 1$)
        $h_n^+ = (i+1) \cdot k_1$;
        if ($h_n^+ > H$)
            $max_j = \frac{H - \text{ceil}(h_n^-)}{\Delta y}$;
        else
            $max_j = (i+1) \cdot k$;
        for($j = 0; j = j + \Delta y; j < \text{ceil}(max_j)$)
            $ptx = i\Delta x$;
            $pty = j\Delta y + ik_2$;
            $nb = x_{offset} - ptx + [\text{floor}(\frac{pty}{\Delta y})] \cdot y_{offset}$;
            $cnt\_wr\_blk = cnt\_wr\_blk + 1$;
```

Algorithm 1. Algorithm to interface the GCC with the DSB.

B. Memory Size After the GCC

The size of BLOCKRAM to contain the new FOV produced by algorithm 1 cannot be known in advance as it is related to the angle ϕ. Therefore the system needs to be evaluated to determine the largest possible BLOCKRAM.

The only accurate procedure to determine the GCC NOSS is to test for all possible angles $\phi \in [0..180]$ the highest NOSS for a given precision Ɛ and set the memory size accordingly. Figure 12 shows results for 3 values of Ɛ equal to 5, 10 and 15; for every Ɛ around ϕ = [70°…110°] the NOSS is the highest and a symmetry can be seen around ϕ = 90°. Angle Ɛ can be reduced by improving GCC accuracy, leading to a smaller GCC NOSS (See Figure 10). Therefore the DSB computation speed is increased.

C. Hybrid Algorithm Parrallelism

Recall from Figure 8 the block diagram of the GCC and DSB acquisition chain; with this new algorithm, Figure 13 shows the new execution chain. Figure 13 shows that the DSB can interpret the result from the GCC as shown in Algorithm 1 described above.

Figure 13 also shows that the entire computation of the DSB cannot be executed in parallel with the GCC as the DSB require the GCC angle to start its computation. As a positive note, the DSB can start his computation as the FOV BLOCKRAM from the GCC is being filled.

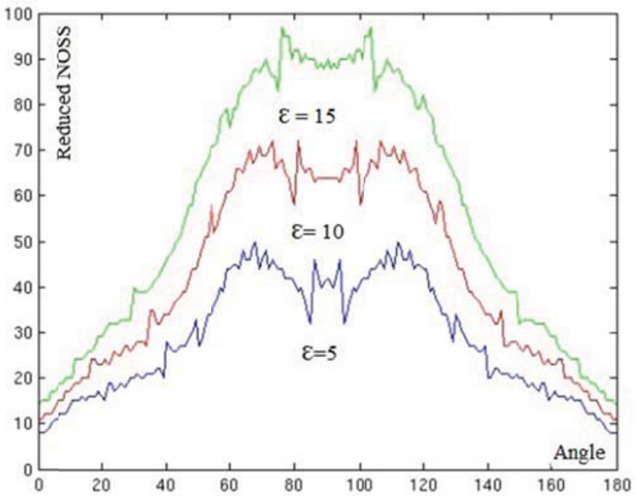


Figure 12. Number of small squares for different values of epsilon.

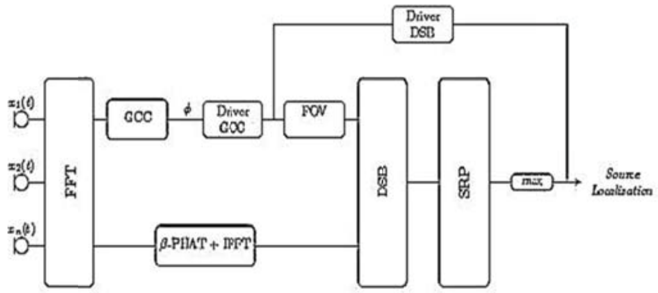


Figure 13. GCC-DSB and DSB –DSB block diagram.

In Figure 13 the aim of the DSB drivers is to use the first DSB localization point to define a reduced region around this detection, if the moving speed of the sound source is known in advance see Figure 14. The next region where the sound source should be tracked can be foretold as shown in Figure 14. The region size depends of the distance traveled by the sound source between two frames as described in equation (26).

$$X_{shift} = Y_{shift} = vT_{trpt} \tag{26}$$

where v is the speed of the sound source (which can be a human walking as he talks) and T_{trpt} is the throughput, the time between two frame computations. The DSB-DSB approach is only possible if a sound source is detected over multiple consecutives audio frames and that the sound source remain in the region we thought it would be. Therefore in Figure 14 the transition 3 is an illegal transition. The DSB-DSB would be a great improvement over GCC-DSB if it was possible to implement it standalone as all the logic resources use for the GCC computation will no longer be necessary.

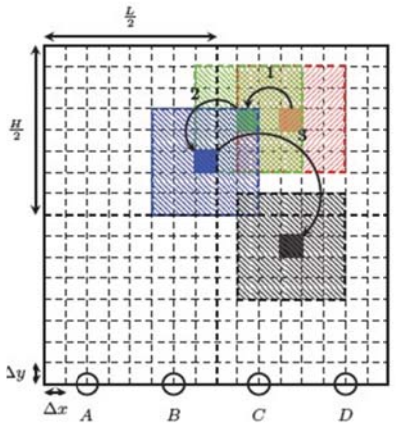


Figure 14. Improvement of the GCC-DSB using DSB –DSB.

TABLE V
DSB AND GCC-DSB COMPUTATION BURDEN

	Localization	$N_{oss} = 256$ (DSB)	$N_{oss} = 32$ (DSB+GCC) $\phi = 53.5$	$N_{oss} = 25$ (DSB+GCC) $\phi = 43.5$	$N_{oss} = 12$ (DSB+GCC) $\phi = 20$
Blk_{read}	$[(N) + (N_s \cdot N) + (N)] \cdot N_{oss} + N_{oss} + [(2N_s \cdot N)] \cdot N_{oss}$	1575168	203040	159969	79980
$Mult$	$[(N_s \cdot N) + (N_s \cdot N) + (N_s) + 1] \cdot N_{oss} + [(2N_s \cdot N)] \cdot N_{oss}$	2228480	296993	236058	122893
Add	$[(N_s \cdot (N-1)) + (N_s \cdot (N-1)) + (N_s) + (N_s)] \cdot N_{oss} + [(N_s \cdot N)] \cdot N_{oss}$	1572864	206853	163845	83973
Div	$[2N_s \cdot N] \cdot N_{oss}$	1048576	137217	108545	55297
$Sqrt$	$[(N_s \cdot N)] \cdot N_{oss}$	524288	68608	54272	27648

VI. Results and Discussion

This section will present the computation methodology used to accelerate the DSB localization and approximations made to reduce design resources utilization, improve implementation flexibility for real-time applications

MATLAB and C are used as verification engine and to pre-compute twiddle factor for the FFT, weights coefficients equation (14) and the delay in sample equation (16). TABLE VI clearly shows that pre-loading this constants on the FPGA, can improve the design speed and reduce logic used. No FPGA multiplications or divisions are necessary in pre-computed values.

TABLE VI
WEIGHT AND SHIFT COMPUTATION BOARD ML505

Resources	Preloaded Constants		Non-Preloaded	
Slice Registers	8	1%	8192	28%
SLICE LUTs	8	1%	6227	21%
RAM	2	3%	7	11%
DSP48	0	0%	142	9%
Estimate Speed	450 MHz		80.64 MHz	

A. GCC- DSB Compared with DSB

TABLE V shows that the number of computations decreases drastically with the NOSS, therefore our contributions become competitive as shown in TABLE V comparing a NOSS (256) DSB to a combined GCC-DSB (NOSS=32) with a ϕ = 53.5° the computation reduction is around 80%. TABLE V also shows that for small angles the computation point's decrease which will be the opposite if the FOV was rectangular. This last result is FOV geometry dependent.

Comparing TABLE V (DSB+GCC, NOSS = 32) and the TABLE III (DSB, NOSS = 32) representing the DSB-DSB algorithm the difference of throughput is given by TABLE II which is quite negligeable, the advantage of the DSB-DSB as state above will then be only ressource utilization.

The logic resources necessary to generate the interface between GCC and the DSB are negligible but require an intellectual propriety (IP) core to compute operations such as tangent.

TABLE VII compared with TABLE IV shows that above 200 MHz the real-time constrainst pose in Section IV.D is resolved with our Hybrid algorithm as {5.9 ms and 3.93 ms} are in inside the range [0..11.6] ms.

TABLE VII
GCC-DSB TROUGHPUT COMPARED TO TABLE IV

GCC-DSB (Clock Speed) MHz	200	400	600
Throughput(ms) NOSS = 32	11.75	5.9	3.93

Another profiling approach for both algorithms was done to see the advangage of one over the other under a sequential implementation. All algorithms were described in MATLAB but since most complex functions such as FFTs, matrix multiplication and trigonometric computation, are optimized, it was not possible to use MATLAB's profiler. We decided to describe them in terms of the amount of multiply-accumulates they require to execute.

First we evaluated the effect of grid size, which affects the NOSS and therefore the computation load; we also changed the number of microphones and obtained the results shown in Figure 15, (time domain DSB), and Figure 16 (frequency domain DSB); they show the multiply-accumulate load of the DSB when increasing the number of microphones. The dashed line is for the smallest grid resolution (4 cm x 4cm) and is the most expensive in terms of the amount multiply-accumulates required since there are more points to evaluate. The importance here, is to see that the increment of resolution affects more than the increase of microphones. Note, from the axis scale, that the time domain approach is more expensive than the frequency one, in terms of computational load.

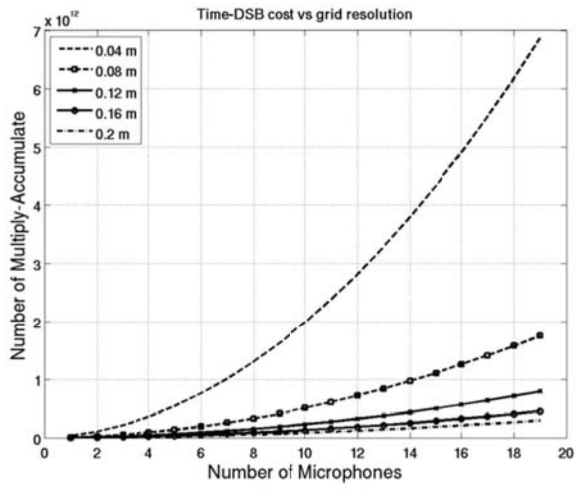


Figure 15. Time domain DSB Computation load for different FOV resolution and number of microphones.

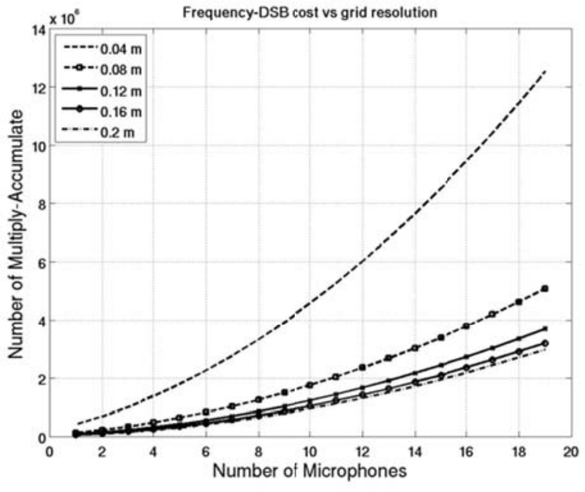


Figure 16. Frequency domain DSB Computation load for different FOV resolution and number of microphones.

For the following graphs, time domain outputs are shown in red whilst frequency domain ones are shown in blue. Black lines at the top account for the DSB behaviour in the time domain, whilst the ones on the bottom, for the frequency domain one.

Multiply accumulate load for the entire DSB algorithm compared to one for the GCC-DSB, yields Figure 17. We can see that the DSB grows at a higher rate than the GCC-DSB; this means that we could increase the number of microphones at low computational cost with the new approach since the increment is not as big as for the DSB. We could conclude from this graph that, under the same implementation device, be it FPGA, CPU, GPU, etc, the GCC-DSB algorithm would be able to handle signals from more microphones than the DSB, at lower computational cost.

Figure 18 shows the amount of Multiply-Accumulate vs Frame Size, for the DSB in the time and frequency domains, compared to the GCC-DSB also in both domains, at different resolutions. When the frame size is small, there is a difference of around 1 order of magnitude between DSB vs GCC-DSB (frequency domain, i.e. lower lines); however, as the frame size increases, this difference is reduced significantly; therefore, even when the GCC-DSB is still faster, its performance compared to the DSB will be barely perceptible for small frames. For the time domain approach, performance difference is kept irrespective of the frame size.

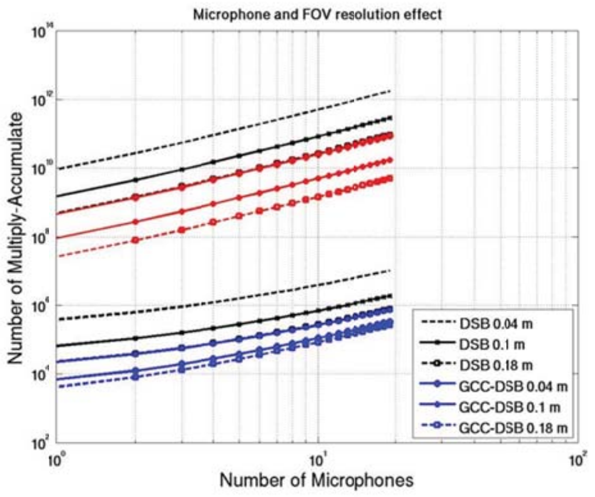


Figure 17. DSB vs DSB-GCC M.A. load Lines on the upper part are computed using the time domain approach whereas the lower ones are for the frequency domain.

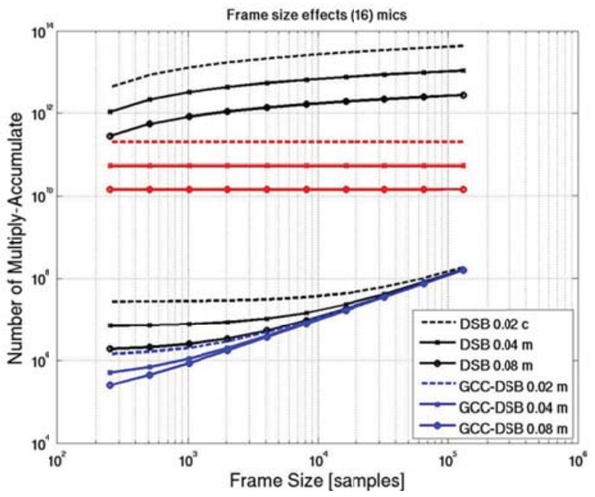


Figure 18. Effects of varying the frame size on both, time (upper lines) and frequency (lower lines) domain, approaches.

Finally, Figure 19 shows a comparison of the multiply-accumulate load between the DSB when varying the value of epsilon; an increase of around one order of magnitude in terms of multiply accumulates, is obtained when changing epsilon from 5 degrees up to 50 degrees. For this plot we assumed the sound source is located at 90 degrees from the center of the array, since the increase range of epsilon bigger. For the time domain approach (upper lines), this speedup is kept irrespective of the grid resolution, while for the frequency domain one, the speed up decreases with less resolution (bigger small squares). Increase in computational load between 5 and 10 degrees is small for the time, as well as, for the frequency domain approaches.

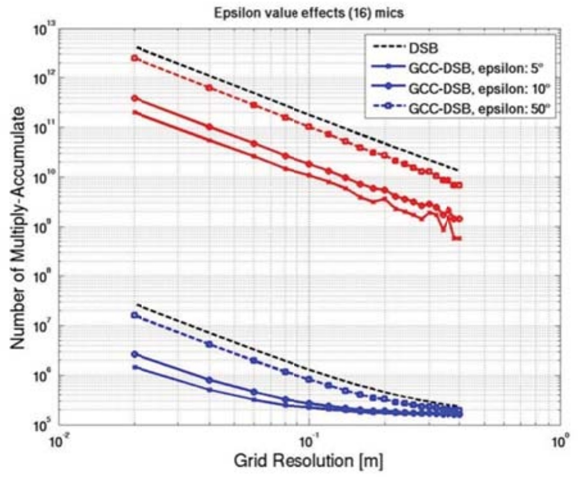


Figure 19. Effects of changing the value of epsilon.

VII. Conclusion and Further Work

This work has shown, at the algorithmic level, that it was possible to improve the speed of sound source localization with our hybrid approach, GCC-DSB compared to the DSB (TABLE VII). At the architectural level we showed there is the possibility to merge the acquisition chain of both algorithms (see Figure 8). This hybrid approach opens the doors to, speed improvements in real applications such as beamforming.

The GCC-DSB proposed algorithm performs better than the DSB independently of the domain in which signals are processed. The analysis employed showed improvements more susceptible in the frequency, to variables such as number of microphones, frame size and grid resolution; nonetheless, the time domain approach is much heavier to compute.

Since the frequency DSB is based on the GCC-PHAT outputs, the area cost of the hybrid algorithm is expected to be small. A few specialized modules for computing tangents and arccosines are required though. More detail in terms of area consumption, power efficiency, accuracy and execution speed is required to thoroughly establish the contribution of the presented algorithm.

FPGA implementation and related design to consider beamforming is proposed as future work; with the speed up gained from the hybrid proposal, the beamforming algorithm, as shown in Figure 20 can be implemented. If the exact location of a sound source is found faster beamforming can also execute faster, therefore allowing it to be applied on moving targets with better results.

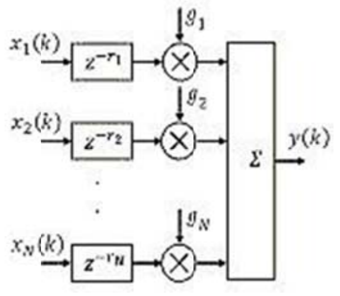


Figure 20. Structure of the Delay and Sum Beamformer [16].

References

[1] Jean-Marc Valin, François Michaud, and Jean Rouat, "Robust Localization and Tracking of Simultaneous Moving Sound Sources Using Beamforming and Particle Filtering," Robotics and Autonomous Systems, vol. 55, no. 3, pp. 216-228, 2007.

[2] SINHA P. and GEORGE A.D. and KIM K., Parallel algorithms forrobustbroadband MVDR beamforming, Journal of Computational Acoustics, 2002. Vol 10, Page 69-96

[3] PRIYABRATA SINHA, ALAN D. GEORGE and KEONWOOK KIM, “ParallelAlgorithmsforRobust Broadland MVDR Beamforming” http://www.hcs.ufl.edu/pubs/JCA2001.pdf

[4] C. CAVE, R. WASSER, Estimating Parallel Processing Speed Multiplier”, http://www.visisoft.us/PDF_Files/EstimatingSpeedMultipliers.pdf.

[5] J.C.H Eric S Chung, Peter A Milder and K. Mai, “Single-Chip Heterogenous computing: Does the Future Include Custom Logic FPGAs GPGPUs” Proceedings of IEEE/ACM International Symposium on Microarchictecture (MICRO), Atlanta, GA, December 4-8, 2010.

[6] Fahad Qureshi, Syed Asad Alam and Oscar Gustafsson, “4K-Point FFT Algorithms based on optimized twiddle factormultiplication for FPGAs” The Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia),Sanghai, Sept 22-24 2010.

[7] Sanjay Thatte, John Blaine, “How to Manage Power Consumption in Advanced FPGAs,” Xcell Journal Xilinx Fall 2002.

[8] Hichem Belhadj, Vishal Aggrawal, Ajay Pradhan and Amal Zerrouki “Power-Aware FPGA Design” 2009. http://www.actel.com/documents/Power_Aware_WP.pdf.

[9] Ivan Tashev, Sound Capture and Processing, Wiley, Ed, 2009.

[10] Analog Device, "Datasheet ADMP421 REV D,".

[11] FTDI UM245 USB-Parallel FIFO Development Module Datasheet Document Reference no: FT_000202. Version 1.04 2009-12-10

[12] Analog Device “Datasheet Application Note 1003”

[13] Saman S. Abeysekera and Charayaphan Charoensak "Efficient Realization of Sigma-Delta (Σ-Δ) Kalman Lowpass Filter in Hardware Using FPGA" Hindawi Publishing Corporation Eurasip Journal on Applied Signal Processing. Volume 2006, Article ID 52736, Pages 1-11 DOI 10.1155/ASP/2006/52736

[14] Altera, "Understanding CIC Compensation Filters”

[15] Weidong Li and Lars Wanhammar “Efficient Radix-4 and Radix-8 Butterfly Elements” www.es.isy.liu.se/publications/papers_and_reports/

[16] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Jacek Dmochowski, "On Microphone Array Beamforming From a MIMO Acoustic Signal Processing Perspective," Audio, Speech, and Language Processing, vol. 15, no. 3, pp. 1053 - 1065, March 2007.

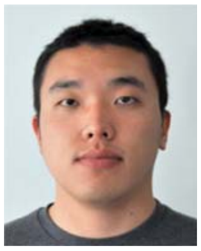
[17] Qi Li, Manli Zhu, and Whei Li, "A portable USB-Based Microphone Array Device For Robust Speech Recognition," in IEEE International Conference on Acoustics, Speech and Signal Processing, Taipei, 2009, pp. 1301 – 1304.

[18] Christian Serge Ibala, J. Vachaudez Carlos Valderrama, “Novel interface drivers to combine real-time localization and tracking“ Submitted to Applied Electronic 2012 Czech Republic 5-7 September 2012.

[19] Ian McCowan, “Robust Speech Recognition using Microphone Arrays”, PhD Thesis, Queensland University of Technology, Australia, 2001.

Christian Serge Ibala received his MSc in 2000, he went to work for Cadence Scotland from 2000 to 2002 then from 2003 to 2009 worked for Xilinx Ireland. In 2008 start his PhD at the University of Limerick (Ireland) he is working toward finishing it in collaboration with the University of Mons (Belgium). His research interest includes reconfigurable architecture, Digital signal processing and systems digital design and validation.

Fernando Escobar was born in 1985. He received bachelor's and MSc and degrees in Electronic Engineering from Universidad de Los Andes, Bogotá, Colombia, in 2008 and 2011 respectively. He is currently a PhD student in the Department of Electronics and Microelectronics, University of Mons, Mons, Belgium. His research interests include high level modelling using Hardware Description Languages, SystemC and MATLAB, among others; his areas of expertise are computer architecture, embedded systems, networks on chip and digital design.



Xin Chang was born in 1988. He received MSc degree in Embedded Computing from University of Turku (Finland) in 2012. He is currently working as a research assistant in the Department of Electronic and Microelectronics, University of Mons. His research interests include on-chip interconnection, high-level synthesis, multiprocessor system-on-chip and signal processing.

Carlos Valderrama's research interests are power processing, consumption and management. He is active in the area of embedded applications, wireless smart sensors for logistics, and signal processing for biomedical and telecommunication applications, among others. His main research activities are methodologies and tools for the design of multi-core architectures and SoC platforms for embedded applications. He is currently member of several scientific committees of international conferences (DAC, FPL, RAW, IDT, ReConfig and Iberchip among others). His research activity is supported by several publications and books chapters, and tutorials.