Embedded Self Organizing Systems (Vol 9. No 3. 2022) (pp.58-61)

Embedded Selforganizing Systems

Special Issue Topic: “International Symposium on Computer Science and Educational Technology“

FPGA based Speech Separation using IPD Features

André Böhle
Professorship of Computer Engineering
Chemnitz University of Technology
09111 Chemnitz, Germany
andbo@hrz.tu-chemnitz.de

René Schmidt
Professorship of Computer Engineering
Chemnitz University of Technology
09111 Chemnitz, Germany
rene.schmidt@informatik.tu-chemnitz.de

Wolfram Hardt
Professorship of Computer Engineering
Chemnitz University of Technology
09111 Chemnitz, Germany
cera@cs.tu-chemnitz.de

***Abstract*-- The problem of speaker separation is an established field in science and goes back to the cocktail party problem defined in 1953. For decades, methods have been improved and developed, but the computational complexity is rarely considered just as the possibility to use hardware acceleration mechanisms. For this reason, this paper addresses the research question: how speaker separation can be realized on embedded systems by exploiting parallelization and intelligent hardware/software partitioning. For this purpose, a concept is described which uses an FPGA for parallelization to separate a speech signal from an intended direction providing a constant throughput rate. The implementation results show the independence of FPGA resources except BRAM size, proving the scalability of the concept, just as the real-time capabilities.**

Keywords--Speech and Signal Processing, FPGA

I. INTRODUCTION

The processing of human voice characterizes a broad field of digital signal processing and covers a wide range of research areas. One of these research areas is the separation of speech signals, which goes back to the cocktail party problem defined by Colin Cherry already in 1953 [1]. The application areas vary widely and spread over speech enhancement, speech recognition enhancement, music processing, generation of three-dimensional sound images, as well as information recovery in audio tracks [2]. As a mathematical basis of speech separation, the problem is described as a linear system of equations:

$$x(t) = A(t) * s(t) + V(t) \qquad (1)$$

where x represents the speech mixture, the matrix A is known as the mixing matrix or spatial impulse response, V describes the noise, and s is representing the intended source signal. One way of solving the problem is founded in the research area of Auditory Scene Analysis, which is described in detail by Bregman [3]. Based on Bregman's description, the computational auditory scene analysis (CASA) was established using computer-aided modelling methodlogies. CASA models the human auditory system and attempts to mimic the human ability to hear selectively. Speech Separation methods using two microphones only are belong to the field of binaural audition.

The fundamental idea of most methods is based on the fact that acoustic signals propagate as waves and thus vary over timen and place. Consequently, at the same time at two different measurement locations in space, there is a time difference of the signal just as a difference in intensity. From this, the features of the interaural phase difference (IPD) and the interaural intensity difference (IID) are derived [4].

II. STATE OF THE ART

The IPD as well as the IID have their origin in Auditory Epipolar Geometry (AEG) [5]. The basis of the model are two microphones placed arbitrary in the room. The resulting mathematical model is as follows:

$$\begin{cases} H_L^{AEG}(\theta, f) = 1 \\ H_R^{AEG}(\theta, f) = e^{-j\phi(\theta)} = e^{-j2\pi f \tau_{AEG}(\theta)} \end{cases} \qquad (2)$$

Here θ describes the angle of incidence and f the frequency. From the model, it can be seen that the differentiation is done by the time lag τ, since $H_L^{AEG}(\theta, f)$ was chosen as a random reference value (cf. [6]).

The herby established AEG is subsequently extended as Revised Epipolar Geometry (RAEG) [7]. The RAEG places the microphones at the edge of a circle representing the cross-section of a robot head. However, this did not lead to a change in the mathematical model, only in an alternative determination of τ taking into account the longer path over radians. A further development of the model by the application of the Scattering Theory whereby a sphere is taken as a basis for the placement of the microphones [8].

On this basis the derivation of the definition for the IPD determination according to (cf. [7]) is carried out:

$$IPD_{AEG}(\theta, f) = 2\pi f \tau_{AEG}(\theta) = \frac{2\pi f a}{c} \cos\theta \qquad (3)$$

These theories provide the basis for the active direction pass filter of Nakadai et al. [9]. This uses the previously described theoretical basis to separate a speech signal from a previously determined direction. A video-based approach is used for direction determination, which detects people in 3D space and thus determines potential speaker positions. The intended angle of incidence is determined on this basis. For speaker separation, the reference value of the IPDs and an

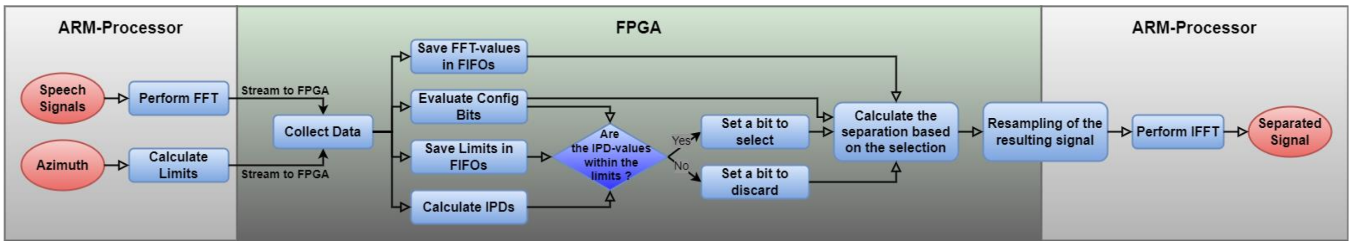

Fig. 1. Process flow of the speech separation

upper and a lower limit value is determined, since the IPD determination becomes less accurate with increasing angles.

When using a robot, the scattered theory can be applied and also the IIDs can be included as reference values. However, for this paper, the general case using AEG will be used as the first impact, focusing on IPDs. The resulting research question is how an AEG-based speaker separation system can be implemented on the hardware side.

III. CONCEPT

From hardware perspective, a ZedBoard utilizing an AMD Xilinx Zynq®-7000 All Programmable SoC is used providing flexible possibilities of hardware software partitioning. Focus of this study is the analysis of Hardware acceleration possibilities for speech separation following the methodology defined by Nakadai et al. [9]. Due to this reason, no separation quality evaluation is conducted in this study. Furthermore, IID features are not considered in this study since the IID reference value calculation expects a robot head. Due to this reason, the speech separation is based on the analysis of the IPD features of the recorded speech signals from software perspective. By evaluating the IPD features, it is possible to extract frequency bins coming from a certain direction from the speech signals. These frequency bins are collected and combined, resulting in a separated speech signal. To support uniform linear microphone arrays (ULA), a minimal ULA of three microphone is considered in this study.

For the calculation of the IPDs defined by Equation 3, the speech signals must be transformed into the frequency domain by applying the FFT. Afterwards, the phase is calculated applying the atan2 function using the real and imaginary parts for each frequency bin as inputs. Finally, the phase difference of the two signals is computed by subtracting the phases per frequency bin resulting in the IPD-value of these two signals.

The IPD values and the direction of arrival, represented by the azimuth angle, are needed for speech separation. Thereby, the azimuth is used to determine the passband for frequency bin selection following the methodology defined by Nakadai et al. [9]. The passband is limited by two angles, the so-called upper and lower limits, which are calculated using the passband function. In the next step, it is checked for each signal combination whether the calculated IPD values are within the limits of the passband. If this is the case, the corresponding frequency bin is selected and collected for signal reconstruction.

After all IPD values are examined and all frequency bins are collected, the result signal is generated from the collected frequency bins. Since the error susceptibility of the IPDs rises with increasing frequency, downsampling should be supported. This allows the use of narrow-band speech models for speech recognition which minimizes the influence of the erroneous frequency ranges. Therefore, a resampling of the result signal is performed changing the sample rate to 8 kHz representing analogous telephone quality. Finally, the result signal is transformed from the frequency domain back to the time domain applying the IFFT. The signal produced after the IFFT contains the information coming from the sound source with respect to the direction indicated by the given azimuth, which completes the speech separation.

IV. IMPLEMENTATION

Since a Xilinx ZedBoard Zynq-7000 is used for the implementation, where the FFT, IFFT and the calculation of the passband limits are performed on the ARM-processor, while the speech separation using IPD features and the resampling are implemented on the FPGA. The breakdown of the individual process steps and the process flow are shown in Fig. 1.

The ARM-processor and the FPGA are connected based on the ARM AMBA® AXI interconnect. The access to the interconnect is controlled via the AXI interface, whereby an AXI 64-bit interface is used for data transmission in both directions. This leads to the highest possible data throughput and the AXI interface provides the basis for pipelining.

After the FFT values of the three microphone signals as well as the limits of the passband have been calculated, they are streamed to the FPGA via the AXI interface. The data transmission is predefined and performed in three cycles, whereby a component called Collector receives all data and assigns it to the signals for further processing. In this process, three 64-bit vectors are transmitted as group and the data is divided among them as shown in Fig. 2.

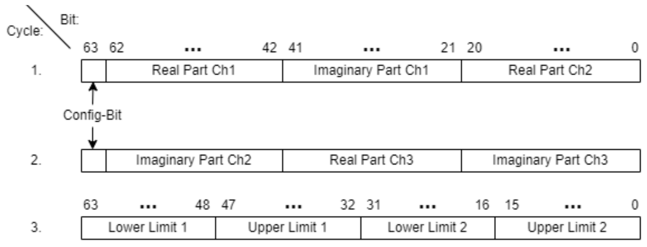

Fig. 2. Structure of the 64-bit vectors

After every transmitted vector group, the configuration bits are separated, the channel values as well as the limits are buffered in FIFOs, and the IPDs between each channel are calculated. All these steps occur simultaneously with each other in different processes. The configuration bits are used to decide whether both limits or only the upper or lower limit should be considered and whether resampling should take place. This mechanism provides the needed flexibility for ULA processing. The IPDs are calculated on the basis of the

Xilinx CORDIC IP-core, which is configued as atan2 function using the real and imaginary parts of the signals of the various channels as input. Subsequently, the phases of each channel combinations are subtracted and scaled down to 32 bit signal vectors using the VHDL resize function, resulting in the IPD-value. Finally, the IPD-values between channels 1 & 2, 1 & 3, and 2 & 3 are determined in this way.

After calculating the IPDs, the next step is to pass them, along with the passband limits from the FIFOs to a component that checks whether the IPD values are within the passband. Depending on the settings determined by the config bits the passband verification takes place. If the IPD value is within the limits, the bits of a 6-bit vector for the respective channel combination are set to the value 1. This vector thus provides information about which frequency bins of which channels are used for the signal reconstruction.

In the last step, the 6-bit vector and the channel values buffered in the FIFOs are used to generate the result signal. For this purpose, the real and imaginary parts of the individual channels are selected for whose channel combination the respective bit of the 6-bit vector corresponds to 1. Subsequently, all real parts are added up and divided by the number of channels, which in this case is three, corresponding to the three microphones used. The summative combination of the separated signals represents a signal enhancement by beamforming in zero direction. The real part created by this combination is finally resized to 32 bits. The same is done analogously with the imaginary parts. At this point there are two possibilities how to proceed, depending on the configuration by the configuration bits, which determines whether the resampling will happen or not. If no resampling is performed, the value for the real part and that for the imaginary part are combined in a 64-bit vector and written to memory using the AXI interface. Otherwise resampling is performed and after this the values are written to the memory in the same way as mentioned before. The resampling used here serves to downsample the signal from 44.1 kHz to 8 kHz. For this purpose it is necessary to interpolate the signal by a factor of 80 and to decimate it by a factor of 441. These two steps can be merged into one by alternately combining five and six frequency bins, respectively. Therefore, the average of the five to six frequency bins is taken, because the energy of the signal must be preserved. However, only the addition of the frequency bins takes place on the FPGA, while the division by the quantity is performed on the ARM-processor, because it is very resource intensive on the FPGA (cf. [10]).

After the values have been written to the memory, the outstanding division is performed first at the software level, depending on the initial configuration. If no resampling was applied, this step will be skipped. Finally, the IFFT is performed, resulting in the separated speech signal.

V. RESULTS

The implemented design has been synthesized on Xilinx Vivado 2015.4. The utilization results are shown in Table I dividing the process flow of the speech separation in seven process steps. The first step represents the main process (P_{Main}) linking and controlling the individual components of the implementation. The second process step represents the data acquisition (P_{Acq}) followed by the FFT values buffering (B_{FFT}) and the limit values buffering (B_{Limit}), storing the dedicated values in FIFOs. Furthermore, the two subsequent processing steps addressing the IPD calculation ($P_{CalcIPD}$) and the frequency bins selection (P_{Select}) The last step covers the processes for the calculation and the resampling of the separated signal (P_{Res}).

TABLE I
UTILIZATION (IN NUMBERS) OF THE FPGA RESOURCES AFTER SYNTHESIS FOR EACH PROCESS STEP (PS)

PS	*Lookup Tables*	*Flip Flops*	*Block RAM*	*DSPs*
P_{Main}	171	344	0	0
P_{Acq}	8	195	0	0
B_{FFT}	129	288	5	0
B_{Limit}	86	172	2	0
$P_{CalcIPD}$ (atan2-IP-core)	25707 (25074)	24442 (24252)	0 (0)	0 (0)
P_{Select}	177	15	0	0
P_{Res}	829	1075	0	2
Total	**27107**	**26531**	**7**	**2**

The results show, that the main part of the resource usage is occupied by the Xilinx IP-core providing the atan2 calculation. Besides the atan2 and the related IPD calculation, the Lookup tables (LUTs) and Flip Flops (FFs) are mainly needed for the calculation of the result signal and its resampling. Furthermore, block RAM is necessary for the FIFOs to store the FFT values and limits until they are processed. The digital signal processor (DSP) is designed for performing mathematical functions like "add", "subtract", "multiply" and "divide" very quickly. In particular, many additions and multiplications take place when the selected frequency bins are combined to form the separated signal, which is why the DSPs are used to generate the real and imaginary parts of the separated speech signal. Overall, the hardware solution shown in Figure 1 requires approximately 60 % of the LUTs, 30 % of the FFs, 10 % of the block RAM, and 3 % of the DSPs provided by the Xilinx ZedBoard Zynq-7000. This shows that the implementation leaves room for further computations on the hardware side and that the speech separation can also be used in other projects.

In addition, the hardware solution works on the principle of pipelining, as can be seen from Figure 1, and requires a fixed processing time for the individual processing steps leading to a constant clock cycle count. The required clock cycles to fill the pipeline are listed for the corresponding tasks in Table II. The process steps for buffering are not listed in Table II reasoned by the parallel processing of P_{Main} and B_{FFT} as well as $P_{CalcIPD}$ and B_{Limit}. Furthermore, the constant processing time of the buffering, with one clock cycle only, does not influence the overall processing time and can be ignored in the overall processing time analysis. For this reason, only the remaining processing steps have been considered in Table II.

TABLE II
CLOCK CYCLES COUNT FOR EACH PROCESS STEP (PS).

PS	*Clock Cycles with resampling*	*Clock Cycles without resampling*
P_{Acq}	3	3
$P_{CalcIPD}$ (atan2-IP-core)	39 (37)	39 (37)
P_{Select}	2	2
P_{Res}	21	4
Total	**65**	**48**

Similarly to the resource usage, most of the processing time is needed for the calculation of the atan2 values, followed by the calculation and resampling of the separated signal. The configuration bits allow different settings, whereas the setting of the limits of the passband has no influence on the clock cycle count, as this only adjusts the selection criterion for the frequency bins. Depending on the configured resampling usage, the resampling process adds 17 clock cycles to the processing time. After the pipeline is filled, a new value of the separated signal is generated after a constant number of clock cycles. If no resampling takes place, a new value is generated every three clock cycles. In case resampling is performed, a new value is generated every 15 clock cycles if five frequency bins are combined and every 18 clock cycles for combining six frequency bins.

In consequence, the hardware solution is fully pipelined and independent of the number of values processed leading to a constant resource utilization. The processing time remains constant as long as the configuration does not change. Consequently, due to a constant processing time, speech separation is possible in real time. Finally, the described implementation and results of an AEG-based speaker separation system proofing scalable realization possibilities fulfilling real-time requirements.

VI. DISCUSSION

The results show a high utilization of LUTs in the implemented design leading to a challenge for optimization on smaller FPGAs. However, by using larger FPGAs it could be considered to transfer the FFT and the IFFT calculation to the FPGA as well. In this respect, the FFT values of the three channels can be calculated in parallel, leading to a better piplining usage and increasing throughput. The transfer of the FFT to the FPGA will also bring higher communication costs, since two times more data have to be transferred to the FPGA. This is reasoned by the symmetric result of the FFT calculation. Additionally, it has to be considered that the FFT calculation on the FPGA is accompanied by higher buffering costs for the necessary limits until the FFT calculation is completed, which leads to higher resource utilization. With the IFFT calculation on FPGA side, the resources needed is increasing even more since the value normalization has to be done on the FPGA before. Depending on the settings made by the configuration bits, the division resulting from the resampling has to be executed on the FPGA as well. A division on an FPGA is very complex and requires additional processing time and resources (cf. [10]). In addition, the representation of the resulting values after the division have to be taken into account ensuring the needed accuracy accompanied by the needed bit vector size. Based on these problems, further investigation can be done in the hardware/software partitioning area to decide if it is useful to compute the FFT and especially the IFFT on the FPGA. Independently of the calculation of the FFT and IFFT, it is possible to make further optimizations on the hardware solution so that the resource utilization and the required clock cycles are reduced.

VII. CONCLUSION

In this paper, a FPGA based speech separation implementation is presented utilizing IPD calculation and supporting ULA processing with three channels. Through an optimized hardware/software partitioning approach and effective piplining the system throughput has been maximized leading to a independent and constant resource utilization. The processing time depends only on the FFT window size used, but is constant for a fixed FFT window size representing real-time capabilities. In future work the influence of the FFT calculation transfer to the FPGA on the resource usage and the processing time has to be investigated enabling further system improvements. Additionally, the increase of the ULA channel count has to be evaluated.

REFERENCES

[1] E. C. Cherry, “Some experiments on the recognition of speech, with one and with two ears,” The Journal of the acoustical society of America, vol. 25, no. 5, pp. 975–979, 1953.

[2] E. Vincent, N. Bertin, R. Gribonval, and F. Bimbot, “From blind to guided audio source separation: How models and side information can improve the separation of sound,” IEEE Signal Processing Magazine, vol. 31, no. 3, pp. 107–115, 2014.

[3] A. S. Bregman, Auditory scene analysis: The perceptual organization of sound. MIT press, 1994

[4] J. Blauert, The technology of binaural listening. Springer, 2013

[5] K. Nakadai, T. Lourens, H. G. Okuno, and H. Kitano, “Active audition for humanoid,” in AAAI/IAAI, 2000, pp. 832–839.

[6] S. Argentieri, P. Danes, and P. Soueres, “A survey on sound source localization in robotics: From binaural to array processing methods,” Computer Speech & Language, vol. 34, no. 1, pp. 87–112, 2015.

[7] K. Nakadai, H. G. Okuno, and H. Kitano, “Epipolar geometry based sound localization and extraction for humanoid audition,” in Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180), vol. 3. IEEE, 2001, pp. 1395–1401.

[8] R. O. Duda and W. L. Martens, “Range dependence of the response of a spherical head model,” The Journal of the Acoustical Society of America, vol. 104, no. 5, pp. 3048–3058, 1998.

[9] K. Nakadai, H. G. Okuno, and H. Kitano, “Robot recognizes three simultaneous speech by active audition,” in 2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422), vol.1. IEEE, 2003, pp. 398–405.

[10] Floating-Point Operator v7.1 LogiCORE IP Product Guide, AMD Xilinx, 12 2020.