Design Space Exploration Framework for Tensilica-Based Digital Audio Processors in Hearing Aids

Jens Karrenbauer, Lukas Gerlach, Guillermo Payá-Vayá and Holger Blume
Institute of Microelectronic Systems
Leibniz University Hannover
Appelstr. 4, 30167 Hannover, Germany
{karrenbauer, gerlach, guipava, blume}@ims.uni-hannover.de

***Abstract*—Choosing a suitable processor architecture for a hearing aid is a difficult task. Various aspects have to be taken into account, like power consumption and silicon area. Also, the computational performance and flexibility of an architecture are essential. Therefore, a wide variety of design goals must be weighted against each other before a final decision for the architecture can be made. In this paper, several configurable audio processors are evaluated, using five commonly known acoustic beamforming algorithms. In order to reduce the exploration time, this paper presents a partly automated design space exploration framework. The hearing aid algorithms are implemented in fixed-point representation to reduce the computational complexity. This framework includes a fixed-point analysis and an automated reference code generation using MATLAB tools. With the Xtensa Xplorer, different configurations of the Tensilica-based processor architecture are profiled. Finally, a case study is presented to show the usability of the proposed framework.**

***Index Terms*—Design Space Exploration, Tensilica Hifi, Tensilica Fusion, MATLAB, Hearing Aid Processor, Low-Power, Acoustic Beamforming**

I. Introduction

Even in difficult acoustic scenarios with background noise and dialogues, the human ear can extract and understand individual conversations. When a hearing-impaired person cannot distinguish between these, it is called the cocktail party problem. Regarding recent studies of the World Health Organization (WHO), around 466 million people all over the world have a severe hearing loss [1]. A modern digital hearing aid could help to reduce the effect of the hearing loss with different monaural or binaural algorithms. One example algorithm to reduce these effects is an acoustic beamformer [2] [3]. This algorithm steers a beam to the desired direction of speech while attenuating the background noise. Therefore, a digital signal processor (DSP) processes the input signals of multiple omnidirectional microphones placed on one (monaural) or two (binaural) hearing aids. Due to the limited power-budget, chip area, and low processing time of a hearing aid, the choice of a suitable hardware architecture for the hearing aid algorithms is a difficult task.

This paper aims to evaluate different digital audio processor configurations, based on five beamforming algorithms of different computational complexity. To keep the design and evaluation time low, a design space exploration framework is proposed. Therefore, MATLAB fixed-point code is analyzed and converted into a hardware-specific fixed-point C-code. The different hardware configurations are profiled with the built-in profiling tools of the Xtensa Xplorer using the generated C-code.

The remainder of the paper is organized as follows: Section II gives an overview of acoustic beamforming algorithms and how they create a focused beam from a set of at least two omnidirectional microphones. In Section III, the basic processor architecture and the different configurations are briefly introduced. Furthermore, Section IV describes the proposed framework for the design space exploration and in the evaluation in Section V a case study to showing the usability of the framework is presented. The paper is concluded in Section VI.

II. Acoustic Beamforming Algorithms

In order to get a good overview of the performance of the different hardware configurations, different beamforming algorithms are used in this paper. The block diagrams of each of them are presented in Fig. 1. A beamformer can generate a directed microphone characteristic from different omnidirectional microphones. Therefore, at least two microphones at different locations are needed. When assuming the distance to the sound source is significantly larger than the distance between the microphones, the angle of incidences for the noise and the desired speech is the same. Therefore, if the distance d between the microphones is known, it is possible to calculate the origin of a signal based on the transit-time difference between them.

A fixed beamformer is a simple algorithm to solve this task. Fig. 1a shows a block diagram of such a beamformer. Basically, the signals from the rear microphone are subtracted from the front microphone. However, the signals from the rear microphone are delayed before subtracting. Based on this delay (τ), it is possible to attenuate different signals at different angles in the microphone characteristic. To block noise from

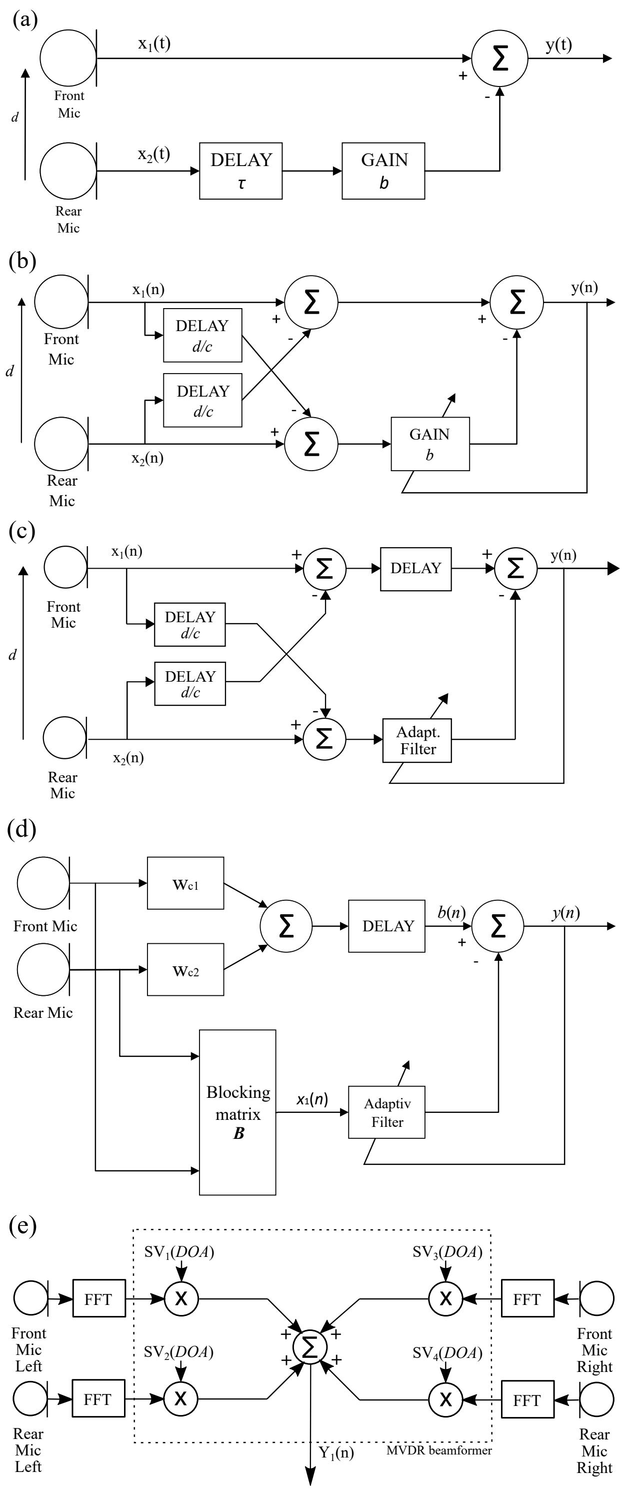

Fig. 1: Block diagrams of different beamforming algorithms: (a) Fixed beamformer [2] (b) Adaptive Gain beamformer [4] [5] (c) Adaptive Filter beamformer [6] (d) General Sidelobe Canceler (GSC) beamformer [7] (e) Minimum Variance Distortionless Response (MVDR) beamformer [3] [8].

the back of the head, the delay (τ) should be set similar to the time, which the sound needs to travel between the two microphones. With this configuration sound arriving from 180° degree will at first arrive at the rear microphone. While the sound propagates further to the front microphone, the delay unit delays the rear signal. The two signals arrive at the same time at the subtraction unit and wipe each other out. With this configuration, the fixed beamformer creates a cardioid microphone characteristic, where signals from the back are canceled. Setting the delay time (τ) to zero, signals arriving from 90° are annihilated. With delay time in between, it is also possible to create super- or hyper-cardioid characteristics. By the gain b following the delay unit, it is possible to balance different amplification levels of the two microphones, e.g. based on production margins.

As the name fixed beamformer implies, the microphone characteristic of such a beamforming algorithm is once configured and can never be changed during the run time. For changing the characteristic during run time, more advanced algorithms are used. They require more complex hardware to do the processing in real-time. In addition to the fixed beamforming algorithm in Fig. 1, an adaptive gain [4] [5], an adaptive Filter [6] a General Sidelobe Canceler (GSC) [7] and a Minimum Variance Distortionless Response (MVDR) [3] [8] beamforming algorithm are used as a reference. The fixed beamformer is quite a simple algorithm that can be implemented only using a FIR filter with eight taps and a single MAC unit. The other algorithms are more advanced. The adaptive gain beamformer (Fig. 1b), for example, creates two different characteristics and calculates a cross-correlation between them subsequently. A Least Mean Square (LMS) or Normalized LMS is used to select a suitable gain.

The adaptive filter beamformer (Fig. 1c) is based on the same concept as the adaptive gain filter, but the gain unit is replaced by an adaptive filter with 16 filter taps instead of one gain. The FIR filter allows the beamformer to attenuate different sound sources from different directions independently in each frequency band. To compensate for the delay of the filter, an additional delay filter with four taps is added. To adapt the coefficient of the FIR filter, a NLMS or a Recursive Least Squares (RLS) algorithm is used. The RLS is not only using the current value, but also the values of the previous iteration. With the additional computational complexity, the filter converges faster.

In the GSC beamformer (Fig. 1d), the signal processing is separated into two different paths. In the upper signal path, the microphone signals are weighted equally and added afterwards. The sum is then delayed to match the processing time of the lower path. In this path, a blocking matrix in combination with an adaptive filter is used to model the noise. This model is then subtracted from the weighted signal to remove the noise. In [9] the improvement of speech intelligibility is shown by the different beamformers. The fixed beamformer gives the least improvement, the adaptive filter beamformer a little more, the adaptive gain beamformer even more, and the GSC beamformer offers the most speech intelligibility. However, these beamformers work under the assumption that the desired signal is always in the range of 0° to 90° and, therefore, only attenuate signals in the range of 90° to 180°. These algorithms neglect interference signals coming from the front.

The MVDR beamformer (Fig. 1e) is the most advanced algorithm among the given choices. It can steer a beam to focus a speaker in between 0° and 90° using binaural information. Therefore, the beforehand guessed Direction Of Arrival (DOA) is used to multiply the inputs with a complex-valued Steering Vector (SV). This is the only algorithm among the beamformers operating in the frequency domain. As a consequence, all operations shown in the figure are using complex-valued operands, which adds a higher computational effort or special hardware requirements to the algorithm. In addition, this beamformer is a frame-based algorithm, which operates on a window of 256 samples. The FFTs, depicted in Fig. 1e, are not taken into account in later comparisons because they are a pre-processing step and not part of the actual beamforming algorithm. In a real hearing aid application, this beamformer is a part of an algorithmic chain, which includes the classification and estimation of the DOA, as presented in [3]. Because all the other parts are also operating in the frequency domain, the FFT is done at the beginning and the input of the beamformer is already in the frequency domain. The FFT blocks are only shown in the block diagram to indicate that the beamformer does not operate in the time domain.

The different complexity of the beamforming algorithms adds several dimensions to the design space. However, this is necessary to get an overview of the performance of the architecture with a different computational load. Nevertheless, it also increases the elaboration time for the exploration because the algorithms have to be adapted to run on the different hardware configurations.

III. Processor Architecture Organization

Selecting a suitable processor architecture for a hearing aid application, different factors have to be taken into account. It is a trade-off between power consumption, chip area, performance, and flexibility. A general-purpose processor (GPP) offers high flexibility, but therefore it has to implement lots of different hardware units, which increase the used chip area. On the opposite, a dedicated full custom hardware unit has the best performance and power consumption but nearly zero flexibility because all the units are specially designed for a particular task. The Application-Specific Instruction-set Processor (ASIP) combines the benefits of both architectures [10]. A Tensilica LX7 [11] is an example of such an ASIP. It consists of a 32-bit reduced instruction set computer (RISC) architecture using a 5 stage pipeline. Using the Xtensa Xplorer, a Tensilica/Cadence designer tool [12], it is possible to modify the hardware configurations. For example, the size of the local memory or number of register files is configurable. Moreover, different arithmetic units, like a divider or a multiplier, can be added and adjusted. Also, a single- or double-precision floating-point unit can be added. As mentioned before, all the hearing aid algorithms are implemented in fixed-point representation to save not only the computational overhead but also the cost of the additional hardware. Therefore, floating-point units are never used in this work, but the impact of different arithmetic units is evaluated later in this work.

TABLE I
COMPARISON OF SOME HARDWARE SPECIFICATION OF DIFFERENT TENSILICA CONFIGURATIONS AND EXTENSIONS.

	Basic	Hifi		Fusion	
	LX7	4	F1	G3	G6
Load (bits)	1x32	2x32	1x32	2x128	2x256
Store (bits)	1x32	1x32	1x32	1x128	1x256
ALU ops/cycle*	1	4	2	4	8
MACs/cycle*	-	4	1	4	8
Issue Slots	1	4	2	4	4
SIMD*	-	2	-	4	8
Pipeline Stages	5	5	5	7	7

*32-bit operands

Furthermore, custom instructions can be generated using the Tensilica Instruction Extension (TIE). These instructions are written in a Verilog like language. Not only instructions but also register files or hardware tables can be constructed by TIE. Regarding the size of these registers, Single Instruction Multiple Data (SIMD) implementations are possible. If two 32-bit wide operands are placed into one 64-bit wide register, a single 64-bit multiplication instruction can execute two 32-bit multiplications at the same time. Also, a Very Long Instruction Word (VLIW) implementation is possible with the Flexible Lenght Instructions eXtension (FLIX). In a VLIW approach, more than one operation can be executed in parallel. After building the configuration, the Xtensa Xplorer generates a corresponding compiler, which supports the different hardware modifications of the architecture, even with high-level languages like C. All these different configuration options increase the design space.

Moreover, Tensilica/Cadence offers different extensions to add to the basic configuration of the LX7. These extensions are a collection of pre-written TIEs and FLIX. For the hearing aid purpose, mainly the cores from the Hifi and Fusion family are interesting because they are designed especially for signal processing while remaining a relatively small core size. The later shown case studies focus mainly on the the Hifi4 [13], the Fusion F1 [14], the Fusion G3 [15] and the Fusion G6 [15]. Table I shows some specifications of the basic core and the extensions. The configurations mainly differ in the bit-width of the arithmetic units and the size of the load/store units. Furthermore, the different configurations have a different amount of added special registers. Particularly special is the configuration of the Hifi 2 extension [13]. The native bit width of the added arithmetic units is reduced to 24-bit instead of 32-bit as for the other configurations. Through the smaller word length, the hardware units can be made smaller, but also the calculation results are more inaccurate. In this work, a word length of 32-bit will be used for the evaluation, to achieve

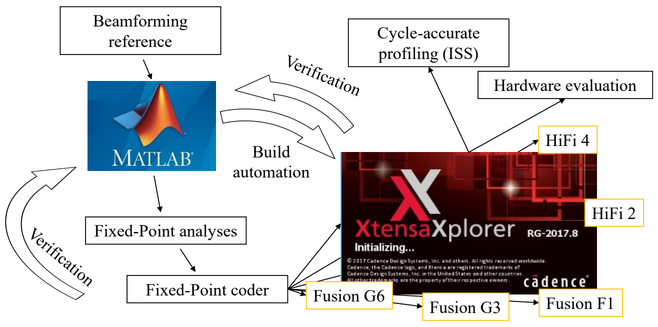

Fig. 2: Proposed Design Space Exploration Framework, using commercial tools like MATLAB and Xtensa Xplorer.

more precise results for the hearing aid algorithms. Therefore, the Hifi2 is not taken into account in the rest of the paper.

The additional extensions add even more complexity to the design space. Furthermore, through different data- or hardware-dependencies, the palatalization of the different beamforming algorithms is not optimal. Moreover, not every instruction can be issued in every issue slot. In the Hifi4, for example, most of the special arithmetic units for processing the audio data are located in issue slots two and three. In contrast, the load and store units are only available at slots zero and one. These effects have to be considered while analyzing the performance of each configuration. Therefore, a simple estimation of the performance of each algorithm is not possible and the algorithms have to be compiled and scheduled on each different hardware configuration, using the given Instruction Set Simulator (ISS) for each configuration. Because of this, the exploration time is increased enormously. To keep the effort of an exploration as low as possible, a design space exploration framework is necessary.

IV. Design Space Exploration Framework

Due to the already shown vast possibilities of different configuration options of the Tensilica cores and the multiple reference algorithms, a design space exploration framework is suggested in this Section. With the help of this framework, it is possible to explore the different benefits of the cores. The proposed framework is shown in Fig. 2. As stated in Section II, the reference algorithms are five different types of beamforming algorithms, all in fix-point representation. These algorithms are all written in MATLAB scripts. At first, a fixed-point analysis is performed to adapt the hardware capabilities of the individual hardware configurations, like the wordlength or the size of the arithmetic units. Therefore, fixed-point objects (fi-objects) have to be created. Fi-objects can model, for example, the wordlength or the fractional point of a variable. Furthermore, the behavior of the arithmetic operations of the hardware can be configured, like the rounding or overflow behavior.

After finishing these configurations, a fixed-point analysis can be performed with these objects and the MATLAB toolboxes [16]. If enough input samples are available, MATLAB can execute the algorithms with floating-point numbers to analyze the range of the computations. Thereafter, a suitable fractional format is suggested. In order to have a valid suggestion, a wide variety of inputs have to be analyzed to match as many scenarios as possible. If the range of input is too small, later on, overflows or range extensions can happen. This might cause wrong results. Therefore, MATLAB also offers to include a certain safety margin. After completing the different fixed-point configurations, the fixed-point beamforming algorithms can be run by MATLAB. To verify that the fixed-point configurations perform accurate enough, the results of these fixed-point executions are compared to the floating-point results. The metrics used in this framework are the absolute and mean differences between the two implementations. In the next step, the fixed-point MATLAB code is automatically translated into C fixed-point code, using the MATLAB Coder.

This generated code represents the configured fixed-point options and handles all the necessary operations to match the configured formats, e.g. the shift before and after certain mathematical operations like an addition. Afterwards, the code is imported into the workspace of the Xtensa Xplorer. To verify that the fixed-point models of the hardware are done correctly in MATLAB, the code is compiled and simulated with the given tools. The results from these simulations are then again compared to the MATLAB floating-point results, using the mean and absolute difference. If the arithmetic units were modeled correctly, these results should not differ more than a few digits. These inaccuracies between the models are based on slightly different overflow and rounding behavior.

Therefore, the same audio files are used as input for the beamforming algorithms as for the MATLAB reference. The results of the Xtensa Xplorer are written into a file and imported to MATLAB. Thus, the comparison can be computed automatically. After the verification of the algorithms, they can be profiled using the given instruction-set simulators (ISS) for the different hardware configurations.

V. Evaluation

A case study is done to evaluate the effectiveness of the suggested framework. Therefore, different hardware configurations are profiled using the five beamforming algorithms. For profiling, different metrics are considered. At first, this evaluation takes the chip area and power consumption into account. The Xtensa Xplorer offers a rough estimation based on a certain technology and operating frequency. Fig. 3 shows the results of an estimation with a 40 nm TSMC technology at 100 MHz. Unfortunately, the tools do not take the switching activities of the profiled application into account for the power estimations. Therefore, a default switching module is used for the calculations. Furthermore, it is also not possible to configure the target frequency lower than 100 MHz. As a result, the estimations performed in this case study used this operating frequency. Even so, in a real hearing aid application, the frequency will much likely be lower than that. However, in this early stage of the design space exploration, this estimation is enough. It will save much time compared with a complete

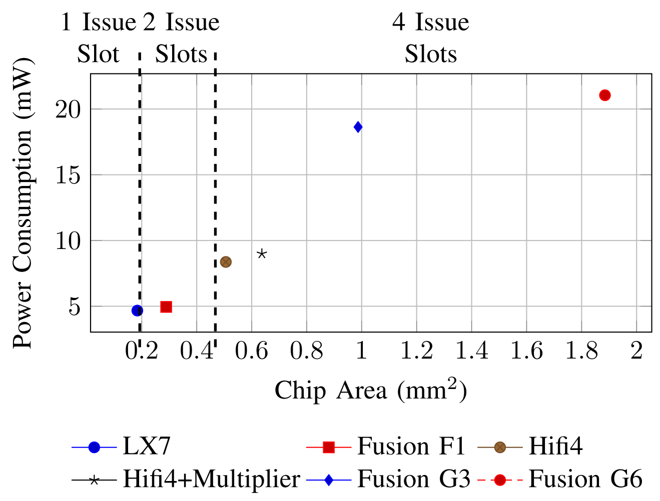

Fig. 3: Comparison of the chip area, power consumption and number of issue slots of the different Tensilica configurations

synthesis, and the estimation errors are assumed to be the same on the different configurations. Because this is out of the scope of a quick design space exploration, it is not done here. However, this estimation already shows the relation between the increasing chip area and the rising power consumption of the different configurations. The configurations with higher computational power and bigger load/store units, like the Fusion G6, are much bigger than the simple configurations as the LX7. The power consumption of the chips is important because the battery life of a hearing aid depends directly on this. In addition to the already presented configurations from Section III, another one is added, here called "Hifi4+Multiplier". In this configuration, an additional multiplier unit is added to the base configuration of the LX7 before adding the Hifi4 coprocessor to look at the impact of this modification. It will increase the area and power consumption but also add some computational benefits, as shown later.

Figure 4 shows some example results of a design space exploration in three different spiderweb diagrams, displaying the metrics number of cycles, hardware utility, and the required real-time frequency. On the axis of the spiderwebs, five different hardware configurations are denoted. The individual colored lines in every spiderweb diagram represent the performance of a specific beamforming algorithm for each metric. The number of cycles is the first metric presented in the most left diagram. This metric depicts how many cycles a hardware configuration needs to process an input sample with a certain beamformer. The fewer cycles a hardware configuration needs for processing a sample, the better is the performance of this configuration. The second diagram shows the utilization of the different hardware configurations. It is measured in Instructions Per Cycle (IPC). To fairly compare the different issue slot architectures, the results are given in percentage, whereas utilization of a 50% means that a four issue slot architecture, for example, schedules a mean of two instructions per cycle. The optimum of this metric would be a utilization

TABLE II
PROFILING RESULTS OF THE BASIC LX7 CONFIGURATION

Beamformer	No. of Cycles	Hardware Utility	Req. Freq.
fixed	536	38,5%	8,5 MHz
adaptive Gain	1582	37,5%	25,3 MHz
adaptive Filter	3822	38,5%	61,2 MHz
GSC	3315	38,5%	53,0 MHz
MVDR	1734	41,5%	17,0 MHz

of 100% because, in that case, the hardware configuration is utilized completely in every cycle. Furthermore, the last spiderweb diagram on the right-hand side of the figure shows the required frequency of the configuration to perform one process step of the beamformer in real-time. As sampling frequency 16 kHz is assumed, a typical sampling frequency for hearing aids. The lower the required real-time frequency is, the less power the hardware configuration will need later. Therefore, the optimum of this diagram is the lowest frequency possible.

The results depicted in Fig. 4 are all normalized to a processing step for one sample, even though some algorithms like the MVDR beamformer are working frame-based. With this simplification, the results are more comparable and the impact of a higher utilization is also represented in the number of cycles, for example. The profiling results of the basic LX7 configuration are not included in the figure, as they are displayed in Table II, using the same metrics. These results are listed separately because the execution time of the configurations would be out of the scope of the Fig. 4. The LX7 needs about ten times more cycles than the F1 to process one sample of the adaptive filter beamformer. With this example, the framework shows that the increasing hardware cost can also give a real benefit in computation time. Even though the size of the F1 configuration is about 50% higher than the LX7, the already gained mean speedup without any optimizations is about one order of magnitude.

The adaptive gain beamformer is a good example to analyze the impact of the before mentioned additional multiplier added to the LX7 before adding the Hifi4 extension. The configuration "Hifi4+Multiplier" needs fewer cycles for processing one sample, maintaining the same utilization resulting in a lower fundamental operating frequency. Conversely, for the other beamforming algorithms, this effect is not visible. This shows that an analysis of the target application for the hardware configuration is crucial. A hardware modification might not be beneficial for every algorithm or every configuration.

Another outcome of the exploration is the execution time of the MVDR beamformer. Even though the beamformer is the most advanced and quite computational complex algorithm, it has a better performance than the other flexible beamformer in nearly every metric. Given that there is no overhead for the processor to get the samples of two hearing aids, this algorithm seems to be more parallelizable than the others. Only the GSC beamformer seems to be more utilizable for some hardware configurations. For the G3 and G6, the utilization

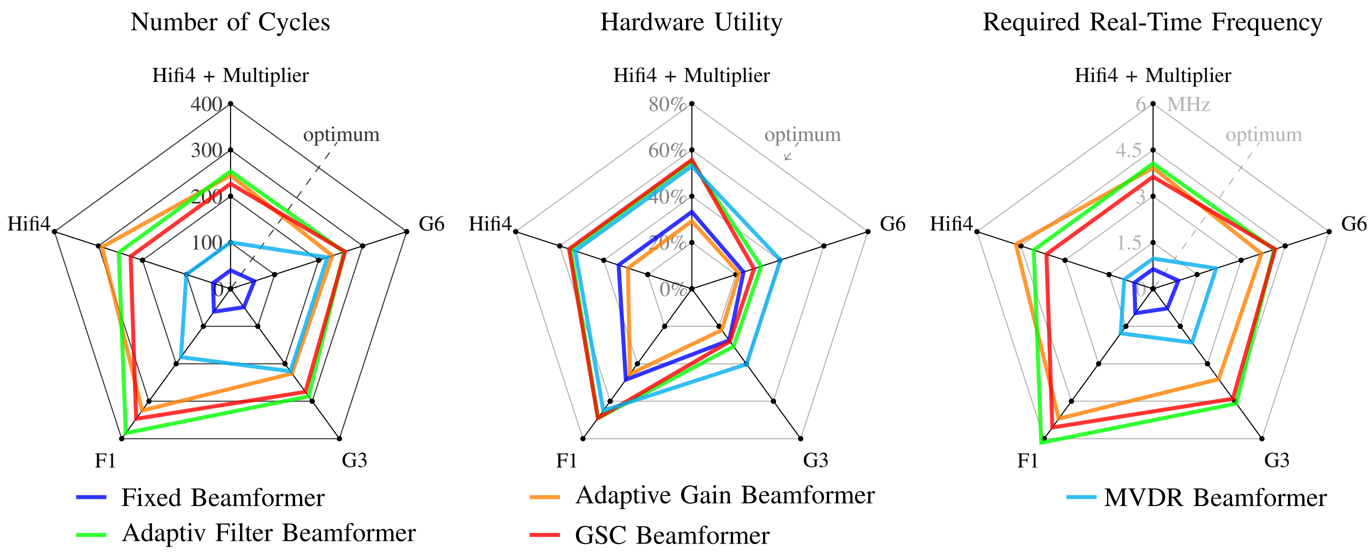

Fig. 4: Results of a case study, comparing the different metrics: number of cycles, the hardware utility and the required real-time frequency using five different complex beamforming algorithms to process one sample on five different hardware configurations

of the hardware architecture is improved using the MVDR beamformer, also resulting in a better performance.

This case study revealed some unexpected behaviors in the interactions of the hardware configurations and the beamforming algorithms. These specific results have not been expected before and demonstrate the need for a fundamental design space exploration. It is also shown that the proposed framework significantly improves and partly automated the research.

VI. Conclusion

Due to the vast possibilities, a design space exploration for digital hearing aids is a difficult task. To perform a fundamental research, five commonly used beamforming algorithms and several Tensilica based hardware configurations are taken into account. However, due to the vast configuration possibilities of these combinations, an enormous design space is created. This paper presents a design space exploration framework to speed up the investigation of the established design space. The framework is based on commercial tools like MATLAB and the Xtensa Xplorer to make the exploration partly automated. For example, this is done with an automated fixed-point analysis and automated C code generation. Besides, the presented case study points out the influence of varying hardware parameters on essential metrics, such as energy consumption, chip area, or execution times. This evaluation shows the benefits of the proposed framework.

Acknowledgment

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2177/1 - Project ID 390895286.

References

[1] World Health Organisation, "Deafness and hearing loss." [Online]. Available: http://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss
[2] J. M. Kates, *Digital hearing aids*. San Diego: Plural Pub, 2008.
[3] K. Adiloğlu, H. Kayser, R. M. Baumgärtel, S. Rennebeck, M. Dietz, and V. Hohmann, "A binaural steering beamformer system for enhancing a moving speech source," *Trends in hearing*, vol. 19, 2015.
[4] G. W. Elko and A.-T. N. Pong, "A steerable and variable first-order differential microphone array," in *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE Comput. Soc. Press, 21-24 April 1997, pp. 223–226.
[5] F.-L. Luo, J. Yang, C. Pavlovic, and A. Nehorai, "Adaptive null-forming scheme in digital hearing aids," *IEEE Transactions on Signal Processing*, vol. 50, no. 7, pp. 1583–1590, 2002.
[6] S. S. Haykin, *Adaptive filter theory*, 3rd ed., ser. Prentice-Hall information and system sciences series. Upper Saddle River, NJ: Prentice-Hall, 1996.
[7] O. L. Frost, "An algorithm for linearly constrained adaptive array processing," *Proceedings of the IEEE*, vol. 60, no. 8, pp. 926–935, 1972.
[8] A. Lacroix, A. Venetsanopoulos, M. Brandstein, and D. Ward, "Microphone arrays: Signal processing techniques and applications," in *Digital signal processing: Mathematical and computational methods, software development and applications*, ser. /Woodhead Publishing series in electronic and optical materials], J. M. Blackledge, Ed. Oxford: WP Woodhead Publ, 2013, pp. 19–37.
[9] L. Gerlach, G. Paya-Vaya, S. Liu, M. Weisbrich, H. Blume, D. Marquardt, and S. Doclo, "Analyzing the trade-off between power consumption and beamforming algorithm performance using a hearing aid asip," in *2017 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*. IEEE, 17.07.2017 - 20.07.2017, pp. 88–96.
[10] N. Werner, G. Payá-Vayá, and H. Blume, "Case study: Using the xtensa lx4 configurable processor for hearing aid applications," *ICT Open*, November 2013.
[11] Cadence Design Systems, Inc., "Xtensa lx7 processor," 2016.
[12] ——, "Xtensa processor developer's toolkit," 2014.
[13] ——, "Tensilica hifi dsp family," 2019.
[14] ——, "Tensilica fusion f1 dsp," 2016.
[15] ——, "Tensilica fusion g dsp family," 2017.
[16] The MathWorks, Inc., "Matlab and matlab coder 2019b," Natick, Massachusetts, United States, 2019.