A 0.964mW digital hearing aid system

Peng Qiao
and Henk Corporaal
Eindhoven University of Technology
Department of Electrical Engineering
Eindhoven, The Netherlands

Menno Lindwer
Silicon Hive
Eindhoven, The Netherlands

***Abstract*—This paper concerns the design and optimization of a digital hearing aid application. It aims to show that a suitably adapted ASIP can be constructed to create a highly optimized solution for the wide variety of complex algorithms that play a role in this domain. These algorithms are configurable to fit the various hearing impairments of different users. They pose significant challenges to digital hearing aids, having strict area and power consumption constraints. First, a typical digital hearing aid application is proposed and implemented, comprising all critical parts of today's products. Then a small area and ultra low-power 16-bit processor is designed for the application domain. The resulting hearing aid system achieves a power reduction of $> 56\times$ over the RISC implementation and can operate for > 300 hours on a typical battery.**

I. Introduction

HEARING impairment issues are becoming serious in today's world. At present about 12% of the human population suffers from hearing problems[1] and are potential users of hearing aids.

This paper concerns the design and optimization of a typical hearing aid system on an ASIP (Application-Specific Instruction-set Processor). The design target is to meet the current requirement on area and power consumption, that is, a chip area smaller than $25\,\text{mm}^2$ which can run at least 50 hours on one 1.35 V zinc battery[2]. Through a series of domain-specific software and hardware optimizations, a power reduction of $56\times$ was achieved over the original implementation on an embedded RISC processor. This paper is organized as follows: First, the related work will be presented in Chapter II. some critical blocks of the proposed hearing aid system will be discussed in Chapter III. Chapter IV provides the basic prototype processor of this design and initial power analysis. The application-specific software and hardware optimizations are discussed in Chapter V, followed by conclusions in Chapter VI.

II. Related work

Digital hearing aid systems consist of different configurable signal processing blocks. Significant research has been devoted to the signal processing domain for hearing aids and many different algorithms were suggested. [3] have listed the recent achievements of different blocks and suggest that the critical signal processing blocks of today's high-end digital hearing aid include adaptive feedback cancellation, adaptive beamfomer,

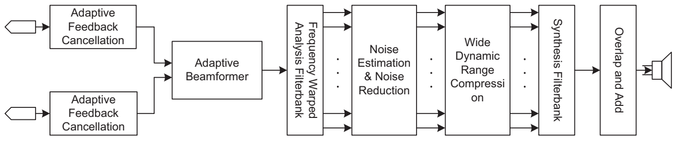
Adaptive Feedback Cancellation
Adaptive Feedback Cancellation
Adaptive Beamformer
Frequency Warped Analysis Filterbank
Noise Estimation & Noise Reduction
Wide Dynamic Range Compression
Synthesis Filterbank
Overlap and Add

Fig. 1: Block diagram of digital hearing aid system

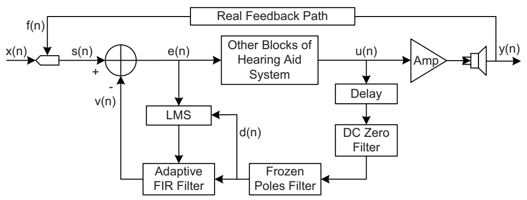

Fig. 2: Block diagram of adaptive feedback cancellation

filterbank, adaptive noise reduction and wide dynamic range compression.

Based on different application requirements and resource budget, the blocks and algorithms for these blocks vary in different hearing aid systems. Almost all studies on hearing aid processors focus on a subset of the blocks and algorithms and adopt different implementation strategies. For instance, [4] proposed a DSP only for filterbank and beamformer, which has power consumption of 0.8 mW; [5] implemented a filterbank on DSP with 0.316 mW power consumption. It is hard to make a direct comparison between these developments. This paper proposes an ultra low-power ASIP for all typical blocks in a high-end hearing aid system for the first time.

III. Proposed hearing aid system

Among different types of hearing aids, such as behind-the-ear (BTE), in-the-ear (ITE), in-the-canal (ITC) and completely-in-the-canal (CIC), this work focuses on the signal processing for a BTE type hearing aid, with two microphones. It contains all the critical blocks. Fig. 1 shows the block diagram of such a hearing aid system. In the following sections, the most critical blocks will be introduced in more detail.

A. *Feedback cancellation (FBC)*

Feedback is caused by the re-amplification of output from the speaker. A feedback path model (see Fig. 2) of the environment must be established in order to cancel the feedback signal before re-processing[2].

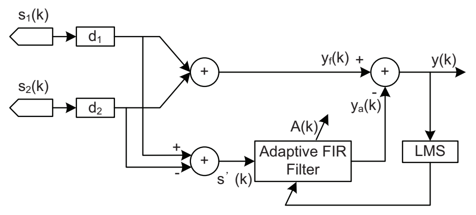

Fig. 3: Block diagram of adaptive beamformer

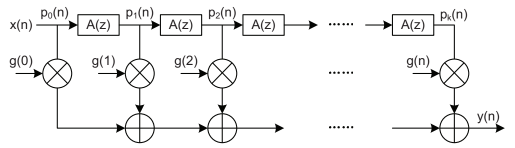

Fig. 4: Block diagram of frequency warped FIR filter

B. Beamformer (BMF)

In hearing aids, it is generally assumed that the signal source is in front of the listener. Generalized sidelobe canceller[6] (see Fig. 3) is a typical adaptive beamformer algorithm in hearing aids. It suppresses signals from the sides and rear using an array of two omni-directional microphones.

C. Frequency warped filterbank (FWFB)

The human ear embodies a critical-band frequency scale having increasing bandwidth with increasing frequency. This block uses a warped FIR filter to approximate the frequency resolution of the ear[7]. It is obtained by replacing the unit delay in a digital filter with first-order all-pass filters (see Fig. 4). In this design, the signal is divided into 17 frequency bands.

D. Noise Reduction (NR)

This block uses different signal characteristics of speech and noise to differentiate intended speech in a noisy environment. It is composed of two parts: noise estimation[8] and noise reduction gain computation[9] (see Fig. 5).

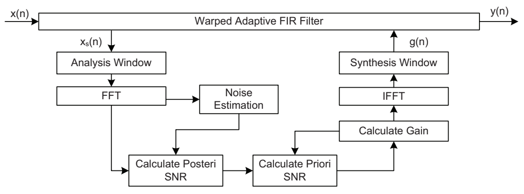

Fig. 5: Block diagram of adaptive NR combined with FWFB

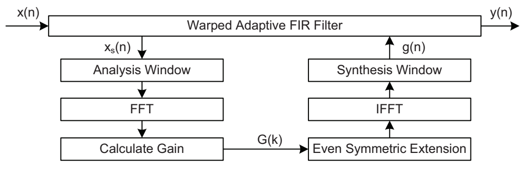

Fig. 6: Block diagram of WDRC combined with FWFB

E. Wide dynamic range compression (WDRC)

The ears of hearing-impaired people have a smaller dynamic range than those of healthy people. This block (see Fig. 6) applies different gains for different input levels to map the normal dynamic range to the reduced range[7].

IV. Processor prototype and basic design

The algorithms discussed in section III were implemented both in floating point and in fixed-point, in order to validate the performance and power consumption of the proposed system. The average output error of the fixed-point implementation was shown to be <1.5%. The fixed-point versions were mapped on ARM 946E-S as reference.

The input data are processed every block of 16 samples to reduce the computation load. The performance calculating one block of samples running on the ARM 946E-S is 130755 cycles. [10] provides its area of 0.488 mm^2 and average power consumption P_u of 0.142 mW/MHz (TSMC C65LP, including caches). Taking into account the cycle count for one block, the power consumption for ARM 946E-S, running the hearing aid application in real-time is calculated as follows:

$$P = (\frac{T_{cycle\_count}}{10^6} \times \frac{F_s}{N_{block}} \times P_u) > 18.6\,\mathrm{mW} \qquad (1)$$

where $F_s = 16\,\mathrm{kHz}$ is the sampling frequency.

Batteries of type 13 with 350 mWh capacity are typical for BTE hearing aids. The power consumption of a complete hearing aid system mainly comes from AD/DA converter and the processor core. The power consumption of state-of-the-art A/D converters can be obtained from [11], as being 0.21 mW, at a 16-bit sampling rate of 16 KHz and 90 dB maximum SNR. As D/A is a part of A/D[11], the upper bound of the total power consumption of two A/Ds and one D/A is assumed to be 0.63 mW.

The starting point of the design space exploration for optimized ASIP is a processor from Silicon Hive called Pearl, a standard 2-issue 32-bit VLIW processor. Pearl has two register files with 16 registers each, one data memory and a program memory. The process started with reducing Pearl's data width to 16-bits. In order to accelerate fixed-point processing and enhance precision, specific function units and 32-bit and 40-bit registers for intermediate results were used.

The power consumption of the ASIP was established using operation trace level power estimation tools and Synopsys PrimePower, while running the actual algorithm.

V. Design optimization

This chapter discusses further software/hardware optimization on the application and processor.

A. Software Optimization

1) Circular Buffer: In adaptive filters, previous outputs are always needed as the delayed inputs for adaptive algorithm (the LMS algorithm in this design):

```
for {$i = 0; i < BS; i++$}
  for {$j = 0; j < Order; j++$}
    if {$j < BS - i$}
```

```
        $delayed\_input[j] = out_{old}[BS + i + j]$;
    else
        $delayed\_input[j] = out_{current}[i + j - BS]$;
```

Two buffers for the previous and current outputs and a conditional branch are needed. In addition, the delay line involves an iterative shift operation after each input data. The processor's modulo-add operation facilitates software implementation of circulator buffers. The adaptive filter can be re-written as:

```
for {$i = 0; i < BS; i++$}
    for {$j = 0; j < Order; j++$}
        $delayed\_input[c\_index1] = out[c\_index2]$;
        $c\_index1 = OP\_modadd(c\_index1, 1, BS + Order)$;
        $c\_index2 = OP\_modadd(c\_index2, 1, Order)$;
```

The absence of conditional code reduces the number of basic blocks, exposing additional instruction-level parallelism.

2) FFT Optimization: The N-point FFT algorithm consists of $\log N$ stages of butterfly calculations and a re-order stage for the bit-reversed output from the butterflies. However, re-order stage can be removed in our application because each output is processed independently.

Furthermore, as the last two butterfly stages of FFT and the first two butterfly stages of IFFT only involve the twiddle factor $W_N^0 = 1$ and $W_N^{N/4} = -j$, the load operations of these stages can be saved by substitution.

3) Loop Optimization: Loop optimization plays an important role in improving memory performance, making effective use of parallel processing capabilities, and reducing overhead associated with executing loops. The loop optimization methods used in this work are loop merging and loop unrolling.

4) Data Distribution: The processor is extended with an additional local data memory and associated load/store unit. The samples and weights arrays of FIR/IIR filter etc. are assigned to different memories so that the loading can be performed in parallel.

TABLE I: Result of software optimizations

	Application	**Cycle Count**
Beamformer	original	9729
	optimized	1964
Feedback cancellation	original	6318
	optimized	1997
Wide dyn. range compr.	original	16223
	optimized	8506
Noise reduction	original	11095
	optimized	8728
Overall system	original merged	15873
	overall optimized	13124

Table I shows the optimization results per block, and after merging into a single loop. Note that the dual microphones imply two feedback cancellation blocks and wide dynamic range compression and noise reduction can share FFT results.

B. Hardware optimization

1) Custom Instructions: Custom instructions result in compact code as well as a lower number of instruction fetches, decodes, and register accesses. On the other hand function units for custom instructions may increase area and may have little relevance in other applications.

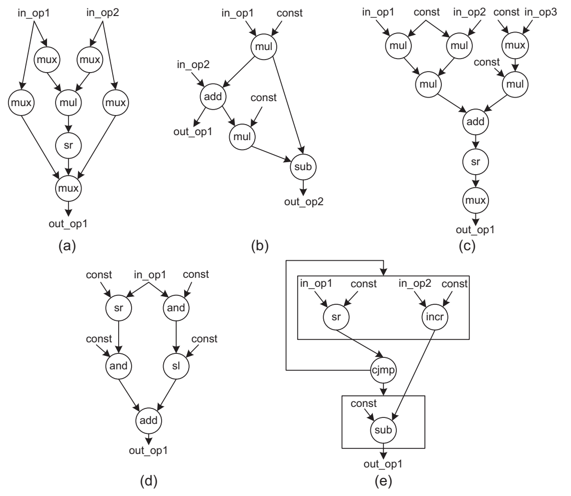

Fig. 7: DFGs of the 5 Custom Instructions: (a) OP_sfpmul (b) OP_warp_unit (c) OP_psnr (d) OP_bitslice (e) OP_norm

In a common representation, a custom instruction is a subgraph of the dataflow graph (DFG) corresponding to the code[12]. The number of subgraphs for a DFG is exponential in terms of the number of the nodes. In this paper, the following constraints were applied:

- Number of operands: Only operations with at most 3 inputs and 2 outputs were selected due to the limitation of number of ports of register files.
- Load/store architecture: DFGs with memory accesses were not considered.
- Scope of operations: We do not considers DFGs that cross the boundary of basic blocks.

Subsequently, the selection was reduced based on the frequency of execution and complexity. The complexity is obtained by analysis of the critical path in DFG. Eventually, 5 custom instructions were selected. Fig. 7 shows the corresponding DFGs.

1. OP_warp_unit: contains the operations for the all-pass filter unit in the frequency warped FIR filter, executed $960\times$ per block.

2. OP_sfpmul: signed fixed-point multiplication with shift, executed $131\times$ per block.

3. OP_bitslice: extracts the specified bit range from a 32-bit input, executed $136\times$ per block.

4. OP_norm: Counts the redundant sign bits of a 16-bit input for fixed-point division. , executed $17\times$ per block.

5. OP_psnr: Calculates the priori SNR based on 3 input operands, executed $17\times$ per block.

Having the selected custom function units, some design space exploration was required to determine allocation of these new units to issue slots. Consequently, the cycle count reduces to 10564 with these custom instructions and the additional area is $0.04384\,\mathrm{mm}^2$. It is quite small compared to the $25\,\mathrm{mm}^2$ area constraint.

2) Reduce instruction width: Instruction width relates directly to both static and dynamic the power consumption of the program memory. After the previous optimization steps, the instruction width is 182 bits. The instructions contain select bits for registers, busses, operations, and immediate bits for constant numbers. Thus, instruction width is reduced by:

1. Removing infrequently used hardware resources, e.g. reduce the size of register files.

2. Reducing width of immediate operands and overlaying them with the select bits of registers.

Both these methods will increase the execution time a little. Table II shows the trade-off and optimization result.

TABLE II: Results of reducing instruction width

Processor	Width	Cycle Count	Power (mW)
Use overlayed imm. bits (v1)	160	10650	1.393
Half the size of reg. files (v2)	150	11221	1.407

3) Loop Cache: Loop cache is a small compiler-driven single-ported register file between program memory and processor. If we define utilization of loop cache as $util$, power consumption of program memory as P_{pmem} and power consumption of loop cache as P_{lc}, the total power consumption of the ASIP becomes,

$$P_{ASIP} = (\frac{T}{10^6} \times \frac{F_s}{N_{block}} \times (P_u - util \times P_{pmem} + P_{lc})\,\mathrm{mW} \quad (2)$$

In order to find a trade-off between area increase and power saving, processor 'v1' from last section is used as reference. In this processor, 58% of power is consumed by the 160-bit program memory. Through careful comparison and selection, a loop cache with size of 32 was adopted.

Table III shows the power consumption of different processor architectures, utilizing the loop cache. Processor 'v3' has a better performance considering both power consumption and execution time.

TABLE III: Loop cache and instruction width reductions

Processor	LC Size	Cycle Count	Power (mW)
v3 (v1 with loopcache)	32	10952	1.210
v4 (v2 with loopcache)	32	11429	1.215

4) Varying VLIW width: Above experiments were performed on a 3-issue processor. As a next step, the number of issue slots can be varied, keeping loop cache size at 32 and overlayed immediate bits. There should be at least two issue slots for parallel access to the two data memories. Table IV shows the power consumption of these alternative processors.

5) Initial synthesis and voltage scaling results: Because execution time of 'v3' is significantly higher than 'v5', yet similar to 'v6', 'v3' is likely to benefit most from voltage scaling. Thus 'v3' was selected for detailed power and area characterisation. When synthesizing 'v3' for TSMC C65G at 20 MHz, the area (including memories, after layout) is $0.49\,\mathrm{mm}^2$. When running at 11 MHz, the minimum clock frequency for real-time operation, the supply voltage can be reduced from 1 V to 0.8 V. At this frequency and voltage, the power consumption of 'v3' on the hearing aid application is 0.334 mW, a reduction of $> 56\times$ over the RISC implementation of section IV.

TABLE IV: Comparison for different VLIW widths

Processor	Cycle Count	Power (mW)
v5(2 issue slots)	15154	1.154
v3(3 issue slots)	10952	1.210
v6(4 issue slots)	9551	1.326

The total power consumption is calculated as follows:

$$P_{v4\_total} \approx P_{AD/DA} + P_{v4} = 0.964\,\mathrm{mW} \quad (3)$$

This system is expected to operate > 364 hours on a standard 350 mWh battery.

VI. Conclusion and future work

This paper for the first time studies the performance of a single ASIP to efficiently execute all the critical blocks of today's high-end hearing aids. A complete hearing aid, based on this ASIP could achieve >300 hours of autonomy and area of $0.49\,\mathrm{mm}^2$, which is well within the specifications of the market (50 hours of autonomy). In future work, additionally vectorization of the algorithms, more aggressive voltage scaling will be studied to further improve system efficiency. Also, based on the result of this study, we aim to find some heuristics and design an automated flow from algorithms to silicon.

References

[1] E. C. Dijkmans, "Hearing instruments go digital," in *European Solid-State Circuits Conf.*, 1997, pp. 16–28.

[2] J. M. Kates, *Adaptive Signal Processing: Application to Real-World Problems*, J. Benesty and A. Y. Huang, Eds. Springer, 2003.

[3] V. Hamacher *et al.*, "Signal processing in high-end hearing aids: State of the art, challenges, and future trends," *EURASIP Journal on Applied Signal Processing*, vol. 18, pp. 2915–2929, 2005.

[4] E. Chau *et al.*, "A subband beamformer on an ultra low-power miniature DSP platform," in *Proc. ICASSP*, vol. 3, Orlando, FL, May 13–17, 2002, pp. 2953–2956.

[5] R. Muscedere *et al.*, "A low-power two-digit multi-dimensional logarithmic number system filterbank architecture for a digital hearing aid," *EURASIP Journal on Applied Signal Processing*, vol. 18, pp. 3015–3025, 2005.

[6] L. Griffiths and C. Jim, "An alternative approach to linearly constrained adaptive beamforming," *IEEE Transactions on Antennas and Propagation*, vol. 30, no. 1, pp. 27–34, 1982.

[7] J. M. Kates, "Principles of digital dynamic-range compression," *Trends In Amplification*, vol. 9, no. 2, pp. 45–76, 2005.

[8] H. G. Hirsch and C. Ehrlicher, "Noise estimation techniques for robust speech recognition," in *Proc. ICASSP*, vol. 1, Detroit, MI, May 9–12, 1995, pp. 153–156.

[9] P. Scalart and J. V. Filho, "Speech enhancement based on a priori signal to noise estimation," in *Proc. ICASSP*, vol. 2, 1996, pp. 629–632.

[10] [Online]. Available: http://www.arm.com/products/processors/selector.php

[11] H. Y. Yang, "A time-based energy-efficient analog-to-digital converter," PhD thesis, Massachusetts Institute of Technology, Feb. 2006.

[12] P. Yu and T. Mitra, "Scalable custom instructions identification for instruction-set extensible processors," in *Proc. CASES*, Washington DC, USA, 2004, pp. 69 – 78.