# ARCHITECTURE OF A SINGLE CHIP ACOUSTIC ECHO AND NOISE CANCELLER USING CROSS SPECTRAL ESTIMATION

*Marco Liem, O. Manck*
Institute of Computer Engineering and Microelectronics, Berlin University of Technology, Cortologic AG
E-mail: marco@liem.de

## ABSTRACT

This paper describes the architecture of a single chip acoustic echo canceller and noise filter for full duplex hands free communications.

To our knowledge the CENS90200 is the first ASIC to eliminate the acoustic echo by using cross spectral estimation. This approach is much more robust against the environmental noise typically found in an hands free (e.g. automotive) environment than the usually employed class of "Normalized Least Mean Square" (NLMS) algorithms.

Additionally a noise filter reduces the background noise by using spectral minimal statistics and diffusive gain factors.

These advanced algorithms have substantial greater computational performance requirements than their traditional counterparts. A highly portable architecture is presented that meets these requirements at low clock to sample frequency ratios. It combines directly hardware mapped algorithms and a purpose build processor optimized for spectral signal processing, thus making a low cost, single chip implementation possible.

## 1. INTRODUCTION

A typical operation environment of an acoustic echo canceller (AEC) and noise filter (NF) is shown in Figure 1, where a speaker at the "far end" communicates over a network with a person using a hands free (car) kit at the "near end".

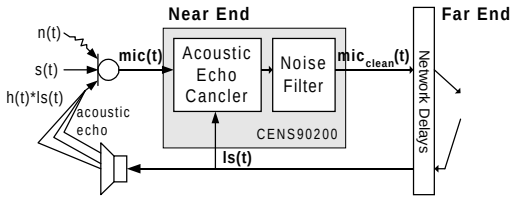


**Fig. 1.** Typical operation environment

There a loudspeaker plays back the signal ls(t) received from the far end speaker. This generates acoustic echoes whose characteristics are determined by the unknown acoustic properties h(t) of the loudspeaker-room-microphone system. Therefore the microphone signal mic(t) does not only contain the desired speech signal s(t), but also the unwanted acoustic echoes h(t)★ls(t) and the environmental noise n(t).

$$\mathbf{mic(t) = s(t) + h(t)\bigstar ls(t) + n(t)} \quad (1)$$

The main task of an AEC is to determine **h(t)**. All other AEC chips known to us use a NLMS type of stochastic gradient descent, which has a very low computational complexity, but is also sensitive to disturbances from **s(t)** and **n(t)** [1,2,3].

## 2. AEC AND NF ALGORITHMS

This section gives a brief outline of the algorithms used by the chip. First, the computation of the outgoing microphone signal $\mathbf{mic_{clean}(t)}$ is described, followed by the echo path and noise estimation algorithms.

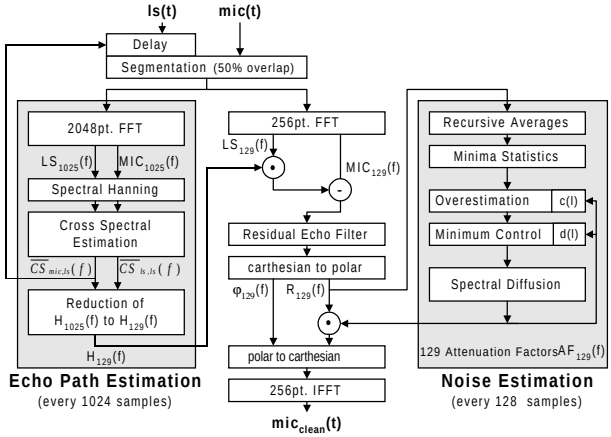


**Fig. 2.** Overview of the AEC and NF algorithms

### 2.1. Computation of the outgoing microphone signal

All signal processing of the chip takes place in the frequency domain. The incoming signals **mic(t)** and **ls(t)** are converted into their corresponding periodograms $MIC_{129}(f)$ and $LS_{129}(f)$ by a *single* 256 point Fast Fourier Transform (FFT).

The estimated echo path $H_{129}(f)$ is used to calculate and subtract the estimated acoustic echo $H_{129}(f)LS_{129}(f)$ from $MIC_{129}(f)$. The attenuation factors $AF_{129}(f)$ from the noise estimation are then applied to filter out the environmental noise. An attenuation factor $AF_{129}(f_n)$ reflects the noise portion of the signal **mic(t)** at the discrete frequency $f_n$, i.e. $AF_{129}(f_n) = 1$, if no noise is detected at $f_n$.

Finally the outgoing signal $\mathbf{mic_{clean}(t)}$ is reconstructed from the cleared spectrum by an inverse FFT in a combination with an overlap-save scheme.

The pass-through time of **mic(t)** in samples is determined by the length $L$ of the FFT window, the overlap $O$ between two consecutive windows and the processing time $PT$.

$$t_{pass} = L/2 + (L-O)/2 + PT \quad (2)$$

The chip uses $L = 256$, $O = 128$ and $PT = 53$.

**2.2. Cross spectral estimation of the acoustic echo path h(t)**

Classical cross spectral estimation [4] is used to provide a frequency domain estimate $H_{1025}(f)$ of the unknown acoustic echo path **h(t)**:

$$\begin{aligned} \overline{CS}_{mic,ls}(f) &= \overline{MIC_{1025}(f) \cdot LS_{1025}{}^{*}(f)} \\ \overline{CS}_{mic,ls}(f) &= \overline{LS_{1025}(f) \cdot LS_{1025}{}^{*}(f)} \\ H_{1025}(f) &= \frac{\overline{CS}_{mic,ls}(f)}{\overline{CS}_{ls,ls}(f)} \end{aligned} \quad (3)$$

$MIC_{1025}(f)$, $LS_{1025}(f)$ are the periodograms of the microphone and loudspeaker signals, respectively and the asterix denotes complex conjugation.

The periodograms are updated every 1024 samples from 2048 sample long segments of **mic(t)** and **ls(t)** using a *single* FFT. A "Hann" window is applied to each periodogram by reducing every frequency bin by the arithmetical average of its two neighbors. Recursive averaging yields the cross spectral estimates $\overline{CS}_{mic,ls}(f)$ and $\overline{CS}_{ls,ls}(f)$ from which $H_{1025}(f)$ is directly derived.

$H_{1025}(f)$ could theoretically be used to estimate the acoustic echo as $H_{1025}(f) \cdot LS_{1025}(f)$. However this echo estimate couldn't be applied to the periodogram $MIC_{129}(f)$ of the outgoing microphone signal because the two periodograms have different resolutions (1025 vs. 129 discrete frequencies). It is not feasible to increase the resolution of $MIC_{129}(f)$ by increasing the FFT length, as this would lead to a longer pass-through delay (see equation (2) ). On the other hand, the echo path estimate $H_{1025}(f)$ should be based on an FFT as long as possible to reduce the bias of the echo path estimation.

The straightforward solution would be to convert $H_{1025}(f)$ into its time domain representative $h_{2048}(t')$ and convolute this with **ls(t)** to estimate the acoustic echo. However, this convolution would involve around 1100 read accesses to **ls(t)** and $h_{2048}(t')$ for every incoming sample. This bandwidth cannot be provided even when separate memory blocks would be used for $h_{2048}(t')$ and **ls(t)**, given the clock to sample frequency ratio of 1536 and a two cycle memory access scheme.

Therefore the resolution of $H_{1025}(f)$ has to be reduced from 1025 to 129 frequency bins. This is achieved by convoluting $H_{1025}(f)$ with the truncated "leakage" function $lf(i)$ for every of the 129 destination frequency bins of $H_{129}(f)$.

$$H_{129}(f_n) = \sum_{i=-32}^{32} H_{1025}(f_n + \frac{i}{8}) \cdot lf(i), \quad n = 0..129 \quad (4)$$

The loudspeaker signal **ls(t)** has to be aligned with its acoustic echo in the microphone signal **mic(t)** to reduce the bias of the cross spectral estimate $\overline{CS}_{mic,ls}(f)$. The optimal alignment is achieved by delaying the loudspeaker signal up to 1024 samples in relation to the microphone signal, so that the peak of the time domain cross correlation function of **mic(t)** and **ls(t)** is at zero lag [4,sec.9.3.3]. This cross correlation function is updated every 8192 samples with an 2048 point IFFT from $\overline{CS}_{mic,ls}(f)$.

**2.3. Noise estimation using spectral minimal statistics and spectral diffusion**

The Noise filter processes only the 129 radiuses $R(f)$ of the microphone spectrum. It assumes that the amount of noise in each $R(f_n)$ is represented by the minimum $R_{Min}(f_n, T)$ of $R(f_n)$ during the preceding period $T$, as during noise free speech each $R(f_n)$ would become zero from time to time.

The noise estimates obtained by this assumption are generally too low, especially for a higher overall noise level $l$. Therefore they are increased by an overestimation function $c(l)$. These overestimated noise values $c(l) \cdot R_{Min}(f_n, T)$ need to be limited according to a minimum control function $d(l)$ to prevent the suppression of low energy phonemes. The overall effect of the non-linear feedback $c(l)$ and $d(l)$ is to balance noise suppression against the generation of undesired musical tones.

The resulting noise estimates are algebraically converted into attenuation factors $AF_{pre}(f)$ and spectrally diffused which greatly reduces the remaining musical tones.

$$\begin{aligned} AF_{pre}(f) &= 1 - \frac{Noise\,Estimate(R(f))}{R(f)} \\ AF_{129}(f) &= AF_{pre}(f) + c\frac{\partial^2 AF_{pre}(f)}{df^2} \end{aligned} \quad (5)$$

Finally the attenuation factors are used to remove the estimated noise from the microphone signal.

$$R_{clean}(f) = R(f) AF_{129}(f) \quad (6)$$

Detailed descriptions of the acoustic echo cancellation and noise filtering algorithm can be found in [5,6].

## 3. SYSTEM REQUIREMENTS

**3.1. Design goals**

The primary design goals were to achieve maximum portability and a low clock frequency.

The algorithmic portion (core) of the design can be integrated as an IP Block into almost any design regardless of the underlying process technology. This is achieved by using a fully synchronous design style and by minimizing the dependency on hard macros. The only hard macro type used by the core are single ported static rams (spsram). Every access to a spsram lasts two complete clock cycles. Furthermore, all inputs and outputs of a spsram are directly buffered by logic registers. These measures decouple the timing of the spsram completely from that of the synchronous core, making it possible to use almost any type of spsram without modifying the core. However they also bring with them a low memory throughput (0.5 accesses/clock cycle) and a long access latency (4 clock cycles).

These are serious limitations given that one sample period lasts only 1536 clock cycles. This clock to sample frequency ratio was dictated by the avoidance of an non-portable analogue PLL in conjunction with the availability of cheap crystals below 20MHz for the standardized sample frequencies $f_s$ of 8000hz/11025Hz. The resulting clock frequencies are 12.288 MHz and 16.9344 MHz, respectively.

**3.2. CENS concepts vs. traditional DSP architectures**

The requirements detailed above prohibit the use of a traditional DSP architecture which operates on accumulator registers and fetches most operands from memory. Although this optimization for streaming multiply and accumulate (MAC) operations makes

it ideal for the FFT algorithm, the other, much less regular portions of the NF/AEC algorithms would be executed extremely slowly, as their intermediate results would have to be kept in the slow memory.

More advanced DSP architectures alleviate this issue by adding more registers with an associated ALU, in effect creating a processor with two computation units. However most of the time one of these units tends to be idle, because it is almost impossible to interleave a MAC based algorithm such as an FIR filter with an algorithm utilizing the other execution unit(s).

Therefore the architecture of the CEN90200 distributes the execution of the algorithm to three units which can operate independently from each other: The FFT unit contains the MAC module of the chip. The noise filter is also hardware mapped as its execution time also directly effects the input to output delay. A processor specifically optimized for spectral signal processing (SPU) executes the remaining operations, including the AEC.

Thus all parts of the algorithm which require streaming memory accesses and/or MAC operations are hardwired and thus can make maximum use of pipelining to "hide" the slow spsram timing. The less regular parts of the algorithm are decoupled from the slow memory by special features of the SPU.

## 4. SYSTEM ARCHITECTURE

### 4.1. Overview

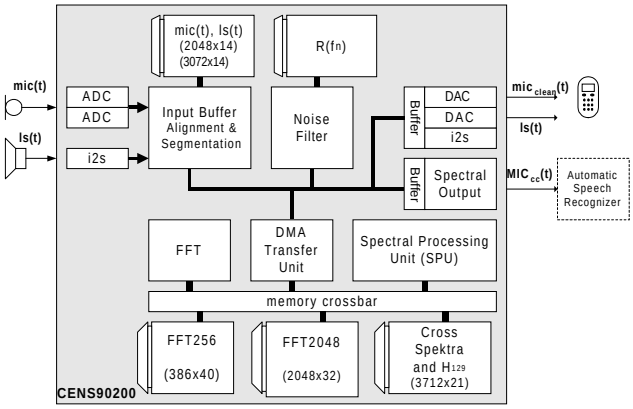


**Fig. 3.** Architecture of the CENS90200 ASIC

The two incoming audio signals **mic(t)** and **ls(t)** are sampled by the two integrated 14 Bit analogue to digital converters (ADCs) or digitally received via an I2S interface. The input buffer aligns **ls(t)** with **mic(t)** and divides them into segments of 256 and 2048 samples each.

These segments are then processed by the FFT, SPU and noise filter units. The SPU and the FFT unit access their memories over a crossbar in a unified address space, to allow an exchange of data with zero overhead.

The resulting $\mathbf{mic_{clean}(t)}$ and the unchanged **ls(t)** signals are outputted simultaneously by two integrated 14 Bit digital to analogue converters (DACs) and an digital I2S interface.

Additionally a periodogram of $\mathbf{mic_{clean}(t)}$ is transmitted every 128 samples by a synchronous serial interface. It consists of 129 radiuses and 129 phases, each having 16 bit resolution. This spectral output can be used directly by most automatic speech recognition algorithms. The pre-processing of the CENS allows a simple ASR implementation on a low cost micro controller.

### 4.2. Architectures of the FFT unit and the noise filter

Both units are hardwired implementations of the corresponding algorithms.

The FFT unit uses a decimation in frequency ("Sande-Turkey") type of algorithm to compute the (inverse) discrete Fourier transform. A block exponent is utilized to increase precision. A 256/2048 point complex (I)FFT is executed in a maximum of 15,885/170,817 clock cycles. These execution times include the in place bitrevers reordering.

The noise filtering unit executes most stages of the NF algorithm (minimum statistics to spectral diffusion) in parallel, with a different Radius $R(f_n)$ in every stage. It requires a maximum of 12298 clock cycles to process all 129 radiuses of $R(f)$.

### 4.3. Architecture of the spectral processing unit

The spectral processing unit performs all (spectral) computations not covered by the FFT or noise filtering units. It is basically a RISC style three address machine with a load/store architecture with special features for spectral signal processing.

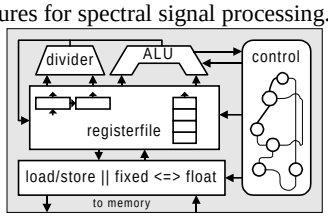


**Fig. 4.** The spectral processing unit (SPU)

An arithmetical-logical unit (ALU) operates on any three registers selected from a register file. Each operation lasts one clock cycle and can be executed in parallel to a multi cycle division. The inclusion of a dedicated divider reflects the prominence of divisions in spectral algorithms. On average, the divider is active during almost half of the total time the SPU is operating.

The load/store unit can simultaneously transfer up to two values between a memory location and the register file. During a transfer a fixed to floating point conversion (and vice versa) can be performed. This allows the cross spectrums to be stored in a 21 bits wide memory. Otherwise their dynamic range would require a width of 36 bit.

The register file contains 24 directly addressable registers with widths ranging from 36 to 20 bits. Eight registers can be paired into four FIFO register pairs (e.g. r0a, r0b). Loading a new value into r0a will update r0b with r0a. This is very useful in iterations, where the results of the previous iterations are accessed (e.g. in spectral hanning):

$$iterate\{\ res(i) = f(res(i-1), res(i-2), ...)\ \} \qquad (7)$$

Without this FIFO structure the iterations would require either more cycles due to register transfers or very long "unrolled" loops.

Thirty-two coefficient registers are *indirectly* addressable by a dedicated register. This address register can be incremented by a value during every access to a coefficient registers. This makes multiple convolution very efficient, making it possible to perform the spectral reduction with a single pass through $H_{1025}(f)$ (see equation (4)). The coefficient registers are set to $lf(i)$. The indirect addressing scheme directly generates the eight shifted sequences of $lf(i)$ required to compute eight $H_{129}(f_n)$ in parallel.

The control unit stores and executes the algorithms. The instructions of the algorithms are not fetched from a ROM, but hardwired in finite state machines (FSMs). Each state corresponds to one instruction executed by the SPU during a clock cycle. An instruction consists of individual commands for *all* execution units of the SPU. Multiple commands for one execution unit may be stored in a single state, one command is then selected based on flags set by the preceding instructions. This concept brings the following benefits:

- Extremely high protection of the algorithm against disclosure, even from the company manufacturing and testing the chips
- Parallel, independent operation of all units (100% orthogonal command set). The SPU can simultaneously initiate in every clock cycle: An arithmetical operation, a division, an immediate load of a constant, a load/store with a float to fixed conversion and a conditional (subroutine) jump
- No cycles lost due to program flow control: Zero cycle conditional jumps, subroutine calls, and deeply nested loops
- No ROM hard macro used, the 2 cycle memory access scheme doesn't affect the arithmetical performance

The size of the control unit's FSMs compares favorably to a ROM having less than *half* the required width:

| | FSMs | ROM |
|---|---|---|
| total number of states / entries | 423 | 512 |
| combined width of all outputs | >80 | 40 |
| size [nand2 equivalents] | 4100 | 4370 |

The obvious drawback of the FSM based architecture of the control unit is that a new algorithm would require considerably more new masks than a new ROM, and also a repeated run of the logic synthesis, P&R and associated verification scripts.

To summarize, the inherent parallelism of the SPU allows to fully "hide" the program flow control and accesses to memory/constants for a wide range of algorithms. At the same time the simple, fully orthogonal programmer's model of a RISC like machine simplifies their implementation and modification. This makes the SPU a good middle ground between a fully hardwired algorithm and a ROM controlled DSP.

## 5. MEASUREMENTS AND RESULTS

The delay imposed on the microphone signal **mic(t)** is always 245 samples periods long. Only 53 sample periods are caused by computations, the remaining 192 are a direct result of the overlap-save scheme applied to the incoming and outgoing signals (see equation (2)).

The following performance measurements were made using the analogue in/outputs with a sampling frequency of 11025Hz: The acoustic echo cancellation for real recordings made in an automotive environment is typically around 15db. This is an average achieved using only speech as **ls(t)** and without any form of attenuation of **s(t)**. The noise reduction was measured by filtering human speech signals which had various noises artificially added to them:

| Noise Type | **F16** | **Factory** | **Pink** |
|---|---|---|---|
| Input SNR [db] | -6 | -9.96 | -10.33 |
| Output SNR [db] | 0.78 | -2.75 | -1.92 |
| SNR improvement [db] | 6.78 | 7.21 | 8.41 |

The smallest clock to sample frequency ratio the chip can handle is about 1450. A standard DSP (Texas Instruments C54x class) with a zero wait state memory requires approximately a ratio of 1700 for the same type of AEC/NF algorithm, but with the following limitations: The length of the DSP's AEC FFT is only 1024 instead of 2048 points, no echo delay calculation is performed and the resulting $H_{513}(f)$ is neither reduced to 129 radiuses nor used to subtract the echo.

A comparison with others AEC/NF ASICS [1,2,3] is given below:

| **Name** | **CENS 90200** | **CS 6422** | **MSM 7731-02** | **PSB 2170** |
|---|---|---|---|---|
| fclk [Mhz] | 12.29 / 16.93 | 20.48 | 19.2 | ≈17 / 34,56 |
| i [mA @V] | 34 / 43 @3.3 | 60@5 | 35 @3.0 | ≈30/50@3.3 |
| fs [kHz] | 8 / 11.025 | 8 | 8 | 8 |
| max AEC length [ms] | 128 + 128 / 92.8 + 92.8 | 63.5 | 59 | 50-129 / 96 |

The CENS90200 has been mass produced in a 0.35µm 3 layer metal CMOS process. The die size is 36mm$^2$ and the primitive logic gate count is 81,516 nand2 gate equivalents.

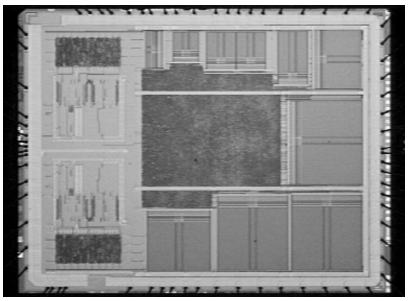

**Fig. 5.** Die photo of the CENS90200

## 6. CONCLUSION

A highly portable and efficient architecture was presented which makes a low cost, single chip implementation of advanced spectral AEC/NF algorithms possible.

## 7. ACKNOWLEDGEMENTS

We would like to thank Dr. Hyoung-Gook Kim for supplying his test signals for the noise filtering performance measurements.

## 8. REFERENCES

[1] Cirrus Logic: "CS6422 Enhanced Full-Duplex Speakerphone IC" (Datasheet), July 2001

[2] OKI Semiconductor: "MSM7731-02 Voice Signal Processor" (Datasheet), May 2001

[3] Infinion Technologies: "Acoustic Echo Canceller ACE PSB2170 Version 2.1" (Datasheet), 1999

[4] G. Jenkins, D. Watts: "Spectral Analysis and its applications", Holden-Day, San Francisco, 1968

[5] H-G. Kim, Klaus Obermayer: "Modified Spectral Subtraction using Diffusive Gain Factors", 7th International Workshop on Acoustic Echo and Noise Control, Darmstadt, 2002

[6] Dietmar Ruwisch: "Verfahren und Vorrichtung zur Elimination von Lautsprechersignalen aus Mikrophonsignalen", Patent De 100 43 064 A1, 2000