IMPLEMENTATION OF A CAR HANDSFREE SPEECH ENHANCEMENT APPLICATION ON A TI TMS320C54x DSP

Jamil CHAOUI, Sebastien de GREGORIO, Daravith KHO, Stephane SINTES, Yves MASSE
Texas Instruments France - Wireless Business Unit
BP5 - 06271 Villeneuve Loubet Cedex FRANCE
j-chaoui1@ti.com, s-de-gregorio@ti.com, d-kho@ti.com, s-sintes@ti.com, y-masse@ti.com

ABSTRACT

This paper describes the implementation of a car handsfree speech enhancement application on a TI TMS320C54x DSP. This fixed-point DSP family is especially suited for wireless applications and it is shown how, by taking full advantage of the DSP architecture and instruction set, advanced wireless speech processing applications can be efficiently implemented on these devices. The performances of the complete speech enhancement application in a car environment are presented, showing that even with a fixed-point arithmetic implementation, high performances close to a floating point implementation can be achieved.

1. INTRODUCTION

In car handsfree applications, the driver speech to be transmitted is mainly corrupted by the ambient noise coming from the car, especially at high speeds. Due to the growth of wireless cellular phone usage and the customer requirement for better speech quality, it becomes mandatory to provide a high conversation comfort during handsfree communications. Therefore, handsfree car kits must include powerful DSPs, allowing to perform efficient speech enhancement algorithms.

At Texas Instruments, a speech enhancement algorithm for car handsfree applications have been developed and implemented on TI TMS320C54x fixed-point DSP. This DSP is especially suited for all wireless applications and in particular to advanced speech processing applications such as speech coding, handsfree speech enhancement, acoustic echo cancellation and speech recognition.

In this paper, we will first present the speech enhancement algorithm. We will then explain why the TI TMS320C54x DSP architecture is well suited for implementing this kind of algorithm. The performances of the complete DSP application in handsfree car environment will then be described.

2. ALGORITHM DESCRIPTION

Let s(t) and n(t) denote the original speech and the car noise respectively. x(t) is the noisy speech signal, collected by the microphone: x(t) = s(t)+n(t).

Let S,X,N represent the spectra of these signals. It is well known that the clean speech S can be estimated from the noisy signal X by applying a time-varying filter H(f) in the frequency domain [1][2]. Therefore, we can write

$$S(f) = H(f)X(f) \quad (1)$$

The time-varying gain filter H can be expressed as a function of the signal to noise ratio in each frequency band, as follows:

$$H(f) = G\left[SNR(f)\right] \quad (2)$$

where SNR is the signal-to-noise ratio computed in each frequency band, expressed by

$$SNR(f) = \frac{X^2(f)}{N^2(f)} \quad (3)$$

The block diagram in Figure 1 shows the main processing modules of the whole speech enhancement application.

The signal is sampled at 8KHz, and processed by blocks of 32ms length (256 samples), with a 50% overlapping between adjacent blocks. Therefore, the processing is done every 16ms.

A hamming windowing is first performed. The spectrum of the frame is estimated with a classical FFT algorithm. The SNR in each frequency band is then computed with (3). The spectral filter H is then estimated with (2).

The input frame is then filtered with H in the frequency domain, by multiplying these 2 signals, following (1).

The cleaned signal is then converted back to the time domain, using a classical inverse FFT algorithm.

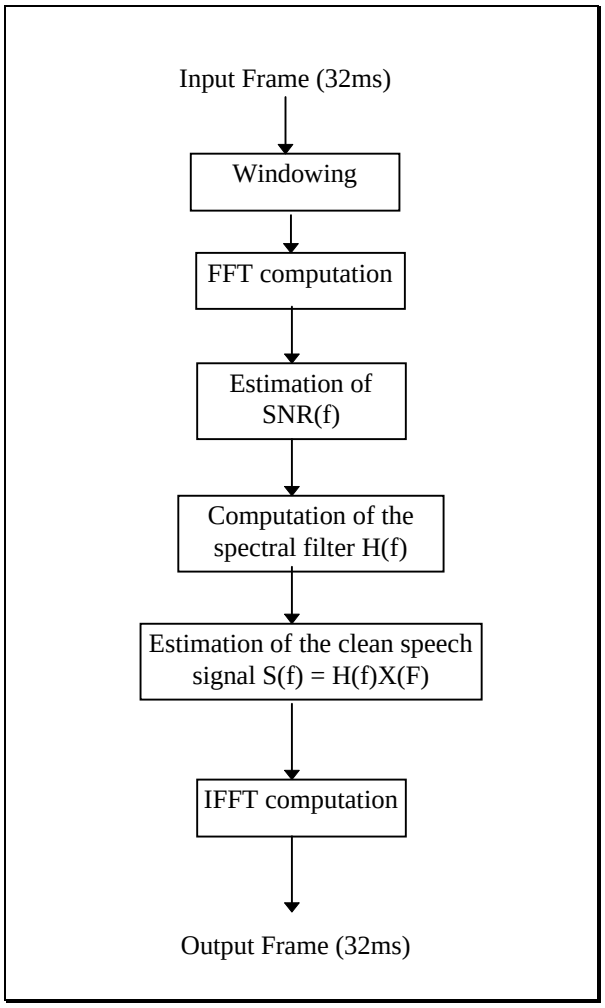

Figure 1
Block diagram

3. IMPLEMENTATION ON TI TMS320C54x DSP

The algorithm has been implemented on a TI TMS320C54x fixed-point DSP [3]. These DSPs combine high-performance (up to 100MIPS at 2.5V CPU core voltage), very low power consumption (0.6 mA/MIPS at 2.5V), a large degree of parallelism, and a specialized instruction set aimed at efficiently implementing a variety of complex algorithms and wireless applications.

These DSPs are based on an advanced Harvard architecture (Figure 2), built around 4 major internal buses: 1 program bus, 2 read data buses and 1 write bus. Therefore, 2 read and 1 write operation can be performed in a single cycle. Some instructions allow to perform memory store or memory loads in parallel with arithmetic computation.

Therefore, advanced speech applications requiring intensive computations, such as speech enhancement algorithms, can be implemented in a very efficient way, by taking fully advantage of the powerful architecture of these DSPs.

3.1 FFT Implementation

Thanks to the bit-reversal addressing provided by the TMS320C54x DSP family, a highly efficient FFT/IFFT implementation can be achieved. This addressing mode is indeed generating addresses in a bit-reversed order, so that no extra cycles are required to arrange the FFT coefficients. Moreover, thanks to the high parallelism of the CPU architecture, the FFT butterfly can be performed in only 9 machine cycles.

It may be demonstrated that the overall cycle number for the complete FFT can be approximated with:

$$C_{FFT} \approx \frac{N}{2} * \log_2(N) * Nb\_cycles\_butterfly$$

Where *N* is the FFT length and *Nb_cycles_butterfly* is the number of cycles in the inner butterfly loop.

Therefore, in our application,

$$C_{FFT} \approx \frac{256}{2} * \log_2(256) * 9 \approx 9276 \text{ cycles}$$

3.2 SNR Estimation

The SNR is estimated by dividing the power spectrum of the noisy speech by the power spectrum of the noise estimated during non-speech periods. This division is performed thanks to a TMS320C54x DSP dedicated instruction called SUBC, performing a conditional subtract operation.

The SUBC operation subtracts the content of a source accumulator from the content of a 16-bit data-memory operand. If the result is greater than 0, this result is shifted 1 bit left, added to 1, and stored into the accumulator. Otherwise, the source accumulator is shifted 1-bit left.

Given a 16-bit positive dividend and divisor, repeating the SUBC instruction 16 times produces a 16-bit quotient in the low accumulator part and a 16-bit remainder in the high accumulator part. Therefore, a 16-bit fractional division can be performed in 16 cycles.

Therefore, for the complete application, the overall cycle number can be approximated with

$$C_{SNR} \approx 128 * 16 = 2048$$

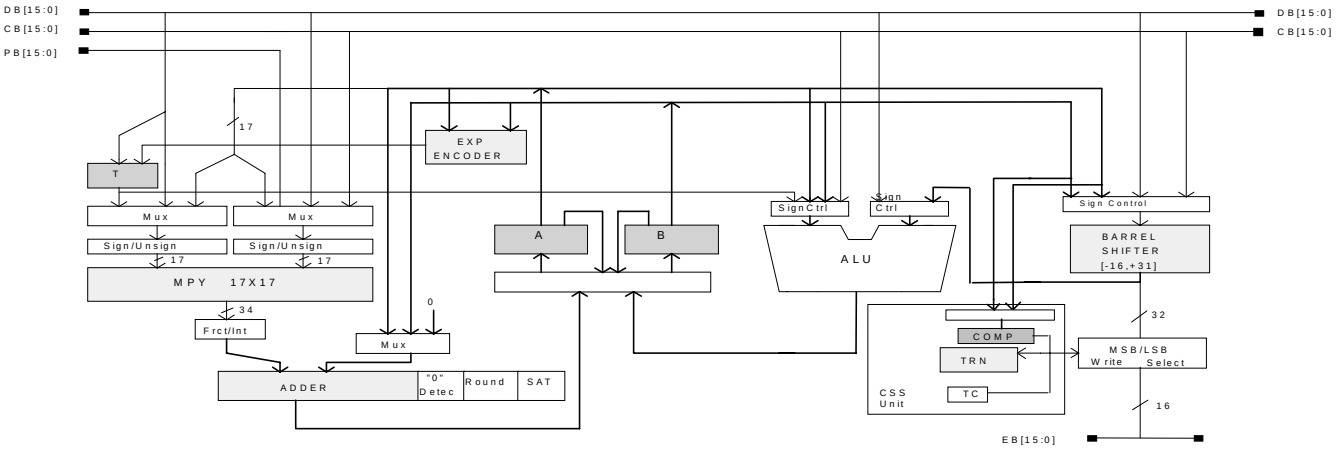

Figure 2
TMS320C54x CPU Architecture

3.3 Computation of the spectral filter H(f)

As the TMS320C54x is also providing a dedicated instruction (POLY) to compute efficiently the output value of a given polynomial, the spectral filter H is not tabulated but directly estimated with a 4^{th}-order polynomial approximation.

POLY instruction is performing 2 operations in parallel:

1) Loading a memory location in the high part of an accumulator B,
2) Multiply the high part of the other accumulator A by the content of a temporary register T, adds the resulting product to the high part of B, rounds the result and stores it into accumulator A.

In simple terms, it may be said that this instruction is performing the following operation in a single cycle:

$$P = P*x + b$$

Therefore, repeating this operation n times will provide the computation of the output of a n-th order polynomial, in the form:

$$P_n(x) = \left(\left(\left(a_n x + a_{n-1}\right)* x + a_{n-2}\right)* x + \ldots . a_1\right)* x + a_0$$

For the complete application, the overall cycle number required by this part can be approximated with:

$$C_H \approx 128 * 4 = 512$$

3.4 Spectral filtering of the noisy signal

Thanks to the 2 read buses, multiplying H(f) and X(f) can be performed in a single cycle. Therefore, the whole filtering of the original signal X through H can be done in 2 cycle per frequency band.

For the complete application, the overall cycle number required by this part can be approximated with:

$$C_{HX} \approx 128 * 2 = 256$$

3.5 Overall application benchmarks

The complete speech enhancement application is performed in 48000 cycles, which represents a CPU load of only 3 MIPS $\left(= \frac{48000 \text{ cycles}}{16\text{ms}}\right)$. Therefore, for a 100 MIPS device, the DSP will only be active during 3% of the frame length.

The following diagram shows the sharing of these cycles between the main processing blocks:

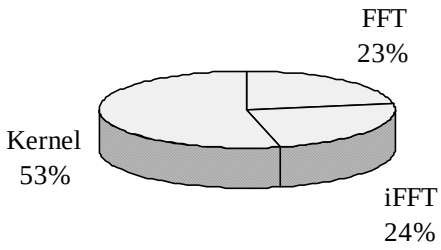

Figure 3
Cycle sharing between the different blocks

Therefore, from this figure, it could be noticed that the FFT/IFFT represents about half of the total number of cycles of the speech enhancement algorithm.

As the speech enhancement application might be executed on a DSP also performing wireless digital cellular baseband processing, special care has also been given to reduce as much as possible the memory consumption. Program size is less than 3K*16bits. Data RAM requirement is 1.6K*16bits, shared between scratch RAM (1K*16bits) and static RAM (0.6K*16bits). Static RAM

blocks must be kept between two consecutive frames. At the opposite, scratch RAM can be used by other applications when the speech enhancement algorithm has completed to process a given frame.

4. PERFORMANCES

The performances of the algorithm DSP implementation have been measured on a database, recorded in real car environment with a typical GSM car handsfree microphone. The software is running on a TMS320C541 DSP, providing 5K*16bits on-chip RAM and 28K*16bits on-chip ROM.

Figure 4 and Figure 5 show the performance of the algorithm on a car speech signal recorded with a male speaker driving at 110 km/h on a highway.

Measurements on these files have shown that the noise power is reduced by more than 10dB in noise periods, and that the signal to noise ratio is enhanced by about 9dB in speech periods. No musical noise can be heard, even when SNR is very low. Subjective tests with naive listeners have shown that speech distortion is considered as very small and not annoying for the conversation.

The adaptation of the algorithm is also very fast. The noise power is reduced by about 10dB after less than 1s of noise.

It can also be seen on this curves that the overall level of the speech is very slightly attenuated by the speech enhancement processing. Precise measurements have shown that the speech attenuation is less than 1dB.

Because of the FFT block processing, the delay introduced by the algorithm on the total mobile round-trip delay is 32ms. This delay is fulfilling ETSI specifications on handsfree processing [4], allowing an maximum additional delay of 39ms for the whole handsfree processing.

In order to study the influence of the fixed-point arithmetic precision loss, the performances of the DSP software have also been compared with the performances of a Matlab floating point implementation of the same algorithm. These tests have shown that the performances are very similar. In particular, the noise power reduction during noise periods is only differing of 0.5dB between both implementations. Informal subjective listening tests have also shown that human hear cannot distinguish between the Matlab and the DSP implementations.

5. CONCLUSION

In this paper, we have described the implementation of a speech enhancement algorithm on a TI TMS320C54x DSP. We have shown how the architecture of this DSP is well suited for implementing this kind of algorithm in a very efficient way, minimizing at the same time CPU loading, memory usage and code size.
We have also detailed the performances of the complete application, in a car handsfree environment.

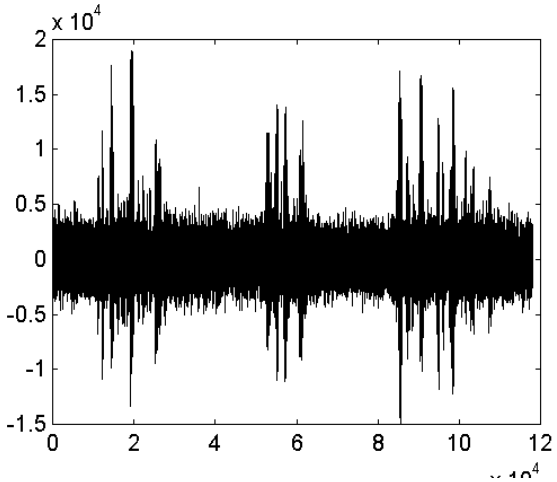

Figure 4
Original File

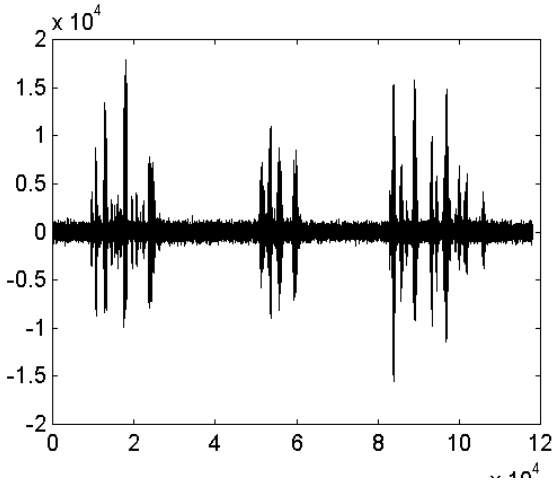

Figure 5
Processed File

6. REFERENCES

[1] Boll, "*Suppression of acoustic noise in speech using spectral subtraction*" IEEE Transaction on Acoustic Speech and Signal Processing, April 1979

[2] R.J. Mac Aulay and M.L.Malpass, "*Speech Enhancement using a soft-decision noise suppression filter subtraction"*, IEEE Transaction on Acoustic Speech and Signal Processing, April 1980.

[3] "*TMS320C54x User's Guide*" - 1996

[4] *"ETSI GSM Specification 3.50 Version 5.0.1"*, ETSI TC-SMG, ETS 300 903, November 1996