Technion - Israel Institute of Technology
Department of Electrical Engineering

Real-Time Digital Watermarking System for Audio Signals Using Perceptual Masking

Yuval Cassuto, Michael Lustig and Shay Mizrachy

Signal and Image Processing Lab, Faculty of EE, Technion IIT , Haifa, Israel.
www-sipl.technion.ac.il

Abstract

Recent development in the field of digital media raises the issue of copyright protection. Digital watermarking offers a solution to copyright violation problems. The watermark is a signature, embedded within the data of the original signal, which in addition to being inaudible to the human ear, should also be statistically undetectable, and resistant to any attempts to remove it. In addition, the watermark should be able to resolve multiple ownership claims (known as the deadlock problem), which is achieved by using the original signal (i.e., the unsigned signal) in the signature detection process.

In order to meet the above demands, a frequency-masking scheme using a psycho-acoustic model is used to ensure a maximal, yet inaudible, additive signature. The algorithm description is as follows: The audio signal is divided into segments. For each segment a local key is calculated and summed-up with an owners key (independent of the segment) to initiate a pseudo-random noise sequence for the segment. The noise is colored by a filter, which is calculated according to the psycho-acoustic model. The original signal is watermarked by adding the colored noise to it.

The project described hereafter contains Matlab^{TM} simulation, a GUI C++ application (insertion & detection) and a real-time embedding scheme on the Texas Instruments TMS320C5410 DSP, using Spectrum Digital XDS510 EVM. The host-target communication was based on the JTAG connection, which caused a bottleneck for data transfer. In order to match real-time data transfer requirements the TIGER 5410/PC supporting Host-Port-Interface (using PC-ISA bus) is used.

The system supports the following modes:

- Off-line mode: a file located in the host-PC is watermarked to create a new one. This is done using DSP target board in order to improve embedding time.
- Digital playback mode: an unsigned file located in the host-PC is delivered to the target board, watermarked and played via the board's D/A converter to the speaker.
- Live mode: a sound is sampled, watermarked and played through a speaker. In the same time, the original samples are stored in the host-PC in order to support future ownership claims.

Abbreviations

AWGN	Additive White Gaussian Noise
A/D	Analog to Digital converter (ADC)
C54	The TMS320C54x digital signal processor.
CCS	Code Composer Studio
CD	Compact Disc
CODEC	Coder Decoder
CPU	Central Processing Unit , the DSP core.
D/A	Digital to Analog converter (DAC)
dB	decibels
DMA	Direct Memory Access
DOS	Disk Operation System
DSP	Digital Signal Processor
DSPLIB	Digital Signal Processing Library
DWM	Digital Watermark
EXP	Exponentiation
EVM	Evaluation Module
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
GUI	Graphic User-friendly Interface
HPI	Host Port Interface
IFFT	Inverse Fast Fourier Transform
ISA	Industry Standard Architecture, IBM- PC/AT Bus
LFSR	Linear Feedback Shift Register
LOG	Logarithm
MCBSP	Multi-channel Buffered Serial Port
MSB	Most Significant Bit
PC	Personal Computer
PN-26	26 bit Pseudo Noise generator
PRG	Pseudo Random Generator
RSA	Rivest, Shamir and Adelman, a public key cryptography algorithm
RTDX	Real Time Data eXchange
SNR	Signal to Noise Ratio
SPL	Sound Pressure Level
TI	Texas Instruments
WAV	Waveform Audio file format

1 Introduction

In recent years we see a growing trend of converting analog media to digital one. This is true for still images, video, and audio signals. It is commonly agreed that the quality of the digital systems are by far better than their analog equals. Manipulating the digital data is more flexible and simpler, duplication and distribution of digital media is easier, and there is no information or quality loss in these processes.

Nevertheless, the increase in the use of digital media raises the problem of copyright protection issues. Copying the media does not reduce its quality, and the created copy is an exact clone of the original. The rapid evolvement of the Internet and the growing development in computer networking has escalated this copyright problem. There are several possible solutions to the copyright problem. Among these possibilities we can mention encryption of the media and blocking the duplication option in devices like audio CD players and recorders. In this paper we focus on a different approach called digital watermarking, and apply it to audio signals. Digital watermarking is used as one of the means to identify ownership of the media. It inserts the copyright data into the media itself in a way that the changes in the data would be unnoticeable, while maintaining standardization.

Digital watermarking systems have two components - the signature embedding system and the detection system. The embedding systems are usually based on adding a signature to the original signal. The detection systems detect the existence of the signature in the media. Most of the methods used to watermark digital audio media are based on perceptual masking phenomena of the human ear [1]. This masking phenomenon occurs in frequency as well as in time domain. The frequency masking is masking of weak spectral components by stronger ones. Temporal masking is the masking of low-level signals that occur before or after high-level signals. In recent years standards for audio compression using perceptual masking were developed. The most common one is the ISO/IEC MPEG [6]. By using an appropriate model of the human hearing system, a signature can be created, such that when playing it at the same time with the media, the signature would not be heard, and the original signal quality is not degraded [1].

The assumption is that the original signal is kept by the owners for future ownership claims. The signature is based on a key, which is known to all (public key) and on the original signal (private key). In order to produce the signature, one needs the public key and the original signal. This way, only the owner can reproduce the signature that exists in the watermarked signal.

The algorithm we have used here works as follows: The audio signal is divided into segments. For each segment, a deterministic pseudo noise signal is calculated. The noise is then colored by a filter, whose coefficients are calculated according to the psycho-acoustic model. Finally, the filtered noise is added to the original signal to get a watermarked signal. The work described hereafter includes a basic MATLAB simulation of the embedding mechanism [3], a C++ implementation in DOS of the embedding mechanism [4], a Windows^{TM} application of the embedding and detection mechanisms, and a real time embedding scheme using the TMS320C5410 [5].

A real time embedding implementation is needed for several applications, e.g.:

- Selling music files (such as WAV) over the Internet, and while sending the file, embedding it with a signature containing the owner's key, transparently to the customer. Thus not compelling additional waiting time.

- Embedding the signature to live broadcasts (e.g., concerts, Internet radio, and on-line conferences).

Examination of the algorithm shows that among the most computation consuming blocks are the FFT, noise filtering, and creating the pseudo-noise (PN) sequence. An apt platform for these operations is a DSP. The TMS320C5410 was chosen because it combines high performance, low cost and portability.

It is important to note that no detection mechanism was implemented in the real-time system. The reason is that, in contrary to signature embedding, detection is naturally done off-line, and the complexity of a robust detection is very high.

The system works in three modes: The “offline mode”, in which we already have a digital media and we want to watermark the media. The “Digital playback mode”, in which we want to play a digital media, and while played, signing it with a watermark in real-time. The “Live mode”, in which live analog audio is sampled, watermarked, and played while saving the original samples in the HOST-PC, to deal with future ownership claims.

2 Digital watermarking algorithm

2.1 Digital signature requirements

There are mandatory requirements, which DWM system must satisfy:

- The signature must be embedded within the data itself, meaning it will not be located in a header, external bit stream or another file, otherwise it will be possible to discard it.
- Knowledge of the algorithm does not enable the removal of the signature.
- In order to retain audio quality, the signature should be inaudible when embedded into the audio signal.
- Undetectable signature will ensure inability to delete it by any attacker. A signature, which can be divulged by averaging, correlation, spectral analysis etc., may be removed or distorted, and therefore is unsafe.
- The signature needs to be resistant against intentional and unintentional distortions, such as additive noise, D/A and A/D conversions, and lossy compression. Thus, it is required that defecting the signature will necessarily damage the audio signal quality.
- The offered watermark has to identify the owner and solve the "Dead-Lock" problem, which arises when several parties claim ownership. This issue is discussed in the next chapter.

2.2 The deadlock problem

The "Dead-Lock" problem arises when two or more parties claim ownership of a media-file. Assume that any watermark detection is done using the original signal (before signature embedding). The method is to use a signature, which is the output of a pseudo random generator (PRG). The seed of the PRG is composed of a public (X1) key, which represents the owner, and a local (X2) key, which is calculated using the original signal by a one-way hash function. The attacker is facing a problem of finding X2 without having the original file, in order to remove the embedded signature. The 'solution' from the attacker's point of view, is to take the watermarked file of the legal owner, which is the sum of the original signal and the legal signature W1, embed it with a signature representing his own owner key W2, and claim ownership on the file, using the legal owner's watermarked file as his original. Verifying the attacker's signature W2 will succeed, but the legal owner will show that the attacker's watermarked file includes his signature W2 too, while the attacker's signature W2 is not detected in his original signal. This way, a definite resolution of ownership is given, and the multiple-claim problem is solved.

2.3 Frequency masking

Frequency masking refers to spectral components masking. When two spectrally close signals occur simultaneously, the stronger signal can cause the weaker one to be inaudible to the human ear. The masking threshold of the masking signal depends on its frequency, SPL (Sound Pressure Level), and the tonal/atonal characteristics of the masked and the masking signals: wideband signals (atonal) can mask tonal signals more easily then vice versa.

The human ear acts like a frequency analyzer, and is able to detect tones in the range of 20 Hz – 20kHz. It is possible to model the human hearing system as a group of 25 band-pass filters, with bandwidth that increases with proportion to their central frequency. These bands are called critical bands, and the bandwidth of every one is denoted as one Bark (there is a nonlinear transformation between Barks and Hertz). The Bark unit is an analytical approximation, which evolved from empirical results of psycho acoustic experiments. These experiments proved that a signal located in the same critical band of a stronger signal, would be unperceived.

2.4 Using Frequency masking for data embedding

The frequency masking effect may be used to embed external data (specifically a signature) in a signal with sufficient magnitude but without any degradation in audio quality [1]. The method described below, relies on the psycho acoustic model of ISO-MPEG Audio for a sampling rate of 44.1 KHz [6]. The following are the stages in applying frequency masking:

I **Instantaneous spectrum calculation -** The audio signal is divided into segments of 512 samples each. Every segment is weighted using a Hann window, and then transformed to frequency domain using an FFT algorithm. The spectral components are normalized such that the maximal spectral point in every segment is considered as the peak SPL constant from the psycho acoustic model tables.

II **Tonal components identification -** Discrimination of tonal (sinusoidal-like) and non-tonal (noise-like) components is required, due to the different masking pattern of each class. Tonal components are local maxima of the spectrum, which satisfy certain criteria regarding their magnitude, relative to adjacent components (figure 1 top left and top right). Non-tonal components are the sum of all spectral magnitudes in a critical band, excluding the contributions of the tonal components. The center frequency of the non-tonal component is set to the result of a weighted average of all the magnitudes in the critical band (figure 1 top right). A single non-tonal component exists in every critical band. The number of tonal components is signal dependent and changes from segment to segment.

III **Decimation of masked components -** After finding all the tonal and non-tonal components, a decimation process is launched. The components, which are decimated, are those below the absolute hearing threshold and the tonal components, which are less than 0.5 Bark away from stronger tonal components. The decimation process reduces computation complexity by eliminating weak un-contributing components, and increases the accuracy of the model by avoiding referring to two close components, as independent, when what happens in practice is a masking of the weak tonal by the strong one.

IV **Masking threshold computation -** The discrete tonal and non-tonal components are assigned each with a masking curve. This curve represents the "amount" of masking effect for all frequencies, caused by a particular component. These empiric curves take their maxima in the spectral point of the component, and decay as the masking component gets farther from the masked frequency. As mentioned before, different masking behavior of tonal and non-tonal components is expressed in different masking curves (figure 1 bottom left). Summing all masking curves and adding the overall contribution to the threshold in quiet, gives the total masking threshold for the segment (figure 1 bottom right graph). Any signal, with a lower spectrum than this threshold, will not be heard by a human hearing system.

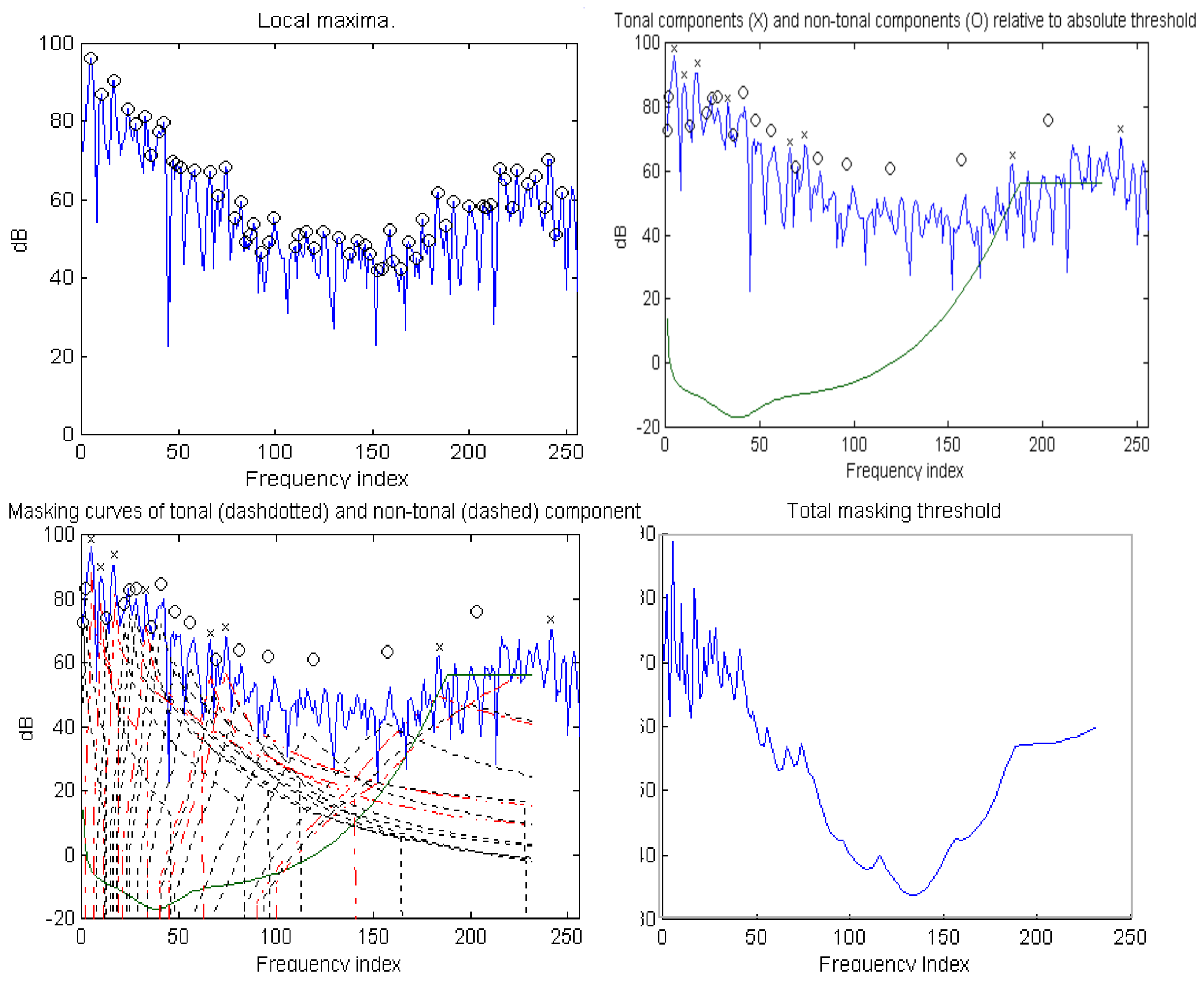

Figure 1: Frequency masking threshold calculation

2.5 The signature embedding system

The following is a layout of the signature construction mechanism (figure 2).

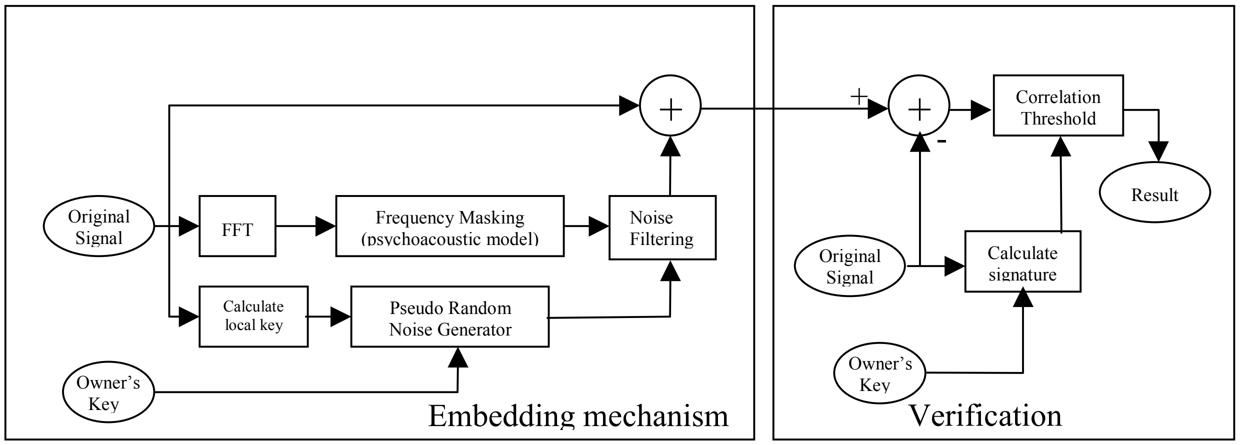

Figure 2: Schematic description of the embedding and detection algorithms

The signal is divided into segments, each segment overlaps with its neighboring segments, in order to reduce discontinuity in the spectral characteristics of the signal. Every segment is fed to two separate paths. In the first path, a signal dependant key (private key) is calculated, and in the other path, a spectral threshold is calculated using the psycho acoustic model (discussed in 2.4). Using the private (local) key, together with a public (owner) key, a pseudo-noise signal is generated. The proper way to merge the two keys into a PRG seed is using a certain cryptographic function such as RSA [8]. Nevertheless the cryptographic safety of the signature is of little interest in the present work. Therefore, the key merging function that was chosen is a shifted sum of the two keys. The extra computational cost of using a safe cryptographic function instead is insignificant because the seed is calculated only once per segment. The PRG is a linear feedback shift register (LFSR) of length 26, which generates a maximum length sequence, with a period of 2^{26}-1. It is important to initialize the PN-26 with a nonzero seed, in order to prevent an all zero sequence.

A pseudo noise sample is calculated by averaging every eight uniformly distributed values of the PRG. From the central limit theorem we get that the statistically independence of the values ensures a normal (Gaussian) distributed samples. In a Gaussian distributed process, statistical dependence can be measured using the correlation function. After having both the pseudo noise signal, and the masking spectral threshold, we use the frequency mask to filter the noise, and change its white flat spectrum, to a spectrum with the shape of the frequency mask. The last stage of the algorithm is normalizing the colored noise to be lower than the signal in frequency domain, meaning adding a constant to the log-space spectrum of the noise, to get a single point interception between the graphs (figure 3).

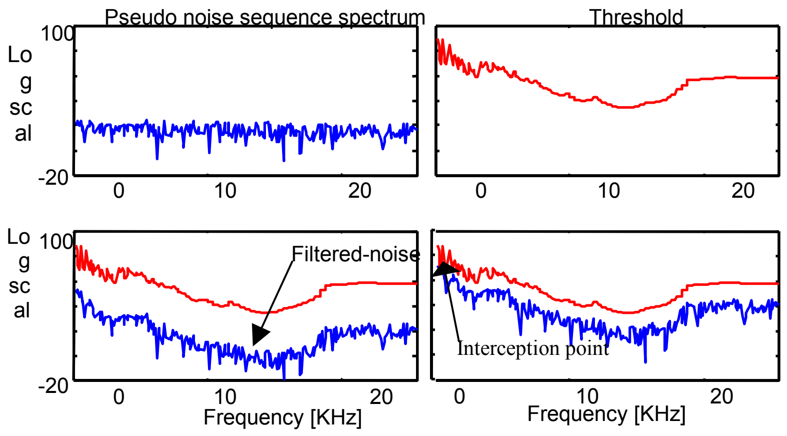

Figure 3 : Noise filtering and scaling

2.6 Signature detection

The possible results of any detection system may be:

- Hit – The system declares that a signature exists, when the input is watermarked with the right signature.
- Correct rejection – The system decides that correct signature does <u>not</u> exist, when the input is <u>not</u> watermarked or watermarked with a different signature.
- Miss – The system decides that correct signature does <u>not</u> exist, when the input is watermarked with the right signature.

- False Alarm - The system declares a signature exists, when the input is not watermarked or watermarked with a different signature.

The False Alarm is the most severe problem because it demonstrates low reliability of the system, and may be used to refute legitimate signatures.

The equivalence between watermarking and natural additive noise suggests that distinguishing the signature from the original signal is difficult, and limits the attacks to actions, which distort the watermarked signal with an intention to prevent detection. Suppose r(n) , $0 \le n \le N-1$ is a segment of N samples, and our goal is to check whether this segment contains a signature. Assume the exact alignment of the tested segment and the original segment is known. This assumption is reasonable, because a cross-correlation function calculation between the tested signal and the original will give the right alignment. In this case, r(n) can be expressed as r(n) = s(n) + d(n) , $0 \le n \le N-1$, where s(n) is identical to the original and d(n) contains noise only, or noise with a signature. Note that the signature is a colored pseudo-noise, which is added to the original signal, while the noise was added to the watermarked signal, via intentional noising or as a result of processing. The detection mechanism is based on the fact, that during verification we have access to the original signal, and to both keys X1 and X2, which are needed for producing the signature. Checking two hypotheses performs the detection. Since s(n) is known, we get:

H_0 : u(n) = r(n) – s(n) = v(n) - no signature found

H_1 : u(n) = r(n) – s(n) = w'(n) + v(n) - signature found

w'(n) is the possibly altered signature , and v(n) is noise. The right hypothesis is chosen by calculating the correlation between u(n) and the original signature w(n) :

$$Corr(u,w) = \frac{\sum_{j=0}^{N-1} u(j) \cdot w(j)}{\sqrt{\sum_{j=0}^{N-1} u(j)^2 \cdot \sum_{j=0}^{N-1} w(j)^2}}$$

and then comparing with a predefined threshold T. The implicit assumption that v(n) is white Gaussian noise with zero mean does not necessarily hold, but simulations show good performance even with more general additive noise, such as MP3 compression. Another measure used for verification is the Similarity:

$$Sim(u,w) = \frac{\sum_{j=0}^{N-1} u(j) \cdot w(j)}{\sum_{j=0}^{N-1} w(j)^2}$$

The similarity value is not bounded to [-1,1] as the correlation value, and can be any value in the range $(-\infty,+\infty)$. The correlation and similarity measures are differently sensitive to certain distortions, and a good decision must be taken on the basis of both values. The main task of the detection system is to set the appropriate thresholds (correlation and similarity), to maximize the Hit rate, while retaining a zero number of

False Alarms. This matter is widely discussed in [3]. The scheme of the detection algorithm is presented in figure 2.

3 Implementation on the TMS320C5410 DSP

3.1 Core implementation

3.1.1 Pseudo-random noise generation

The pseudo random noise is the signature. The noise sequence is initiated by the sum of two keys: the owner key, which is known to all and the local key, which is calculated from the samples in the original signal segment. The creation of the local key is done using a one-way hash function:

$$Local\_Key_{(i)} = (\sum_{n=0}^{383} |X_i(n)|) \bmod 256$$

As a result, the private key is always in the range of [0 , 255] . The operation modulo 256 was chosen because a power of 2 modulo is equivalent to taking the lower bits of the number, and can be implemented as a bitwise AND, which takes one cycle. The local key and the owner key are used to create a third key, which initiates the PRG. The key synthesis function that is used is:

$$Key = 256 \bullet Owner\_Key + Local\_Key$$

In this case the key generated is unique for the owner key chosen.

The pseudo random noise is generated by a PN-26 sequence. The idea is to use a shift register with a linear feedback as described in the figure 4:

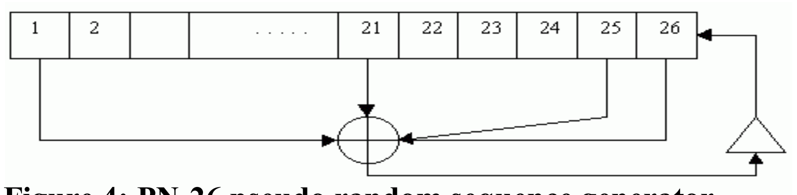

Figure 4: PN-26 pseudo random sequence generator

The register is initialized with the binary value of the combined key. The result is a sequence with zero mean, and an autocorrelation function, which approaches an impulse (figure 5).

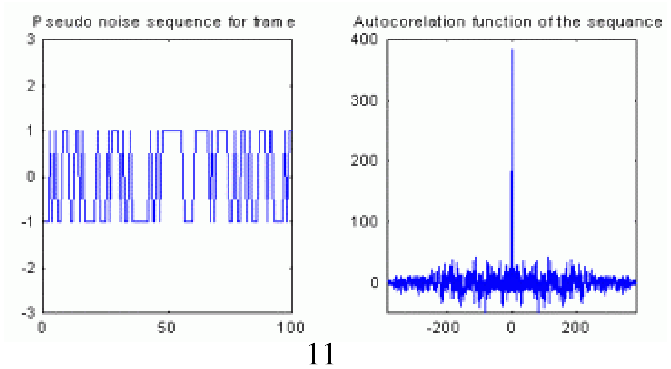

Figure :5 Pseudo random sequence and autocorrelation function

Every eight iterations of the generator are averaged in order to get a normal distribution. Eight was chosen because dividing by a power of two is implemented by a shift operation that takes only one CPU cycle. The number eight is a reasonable compromise between processing speed, and achieving normal distribution as shown in figure 6:

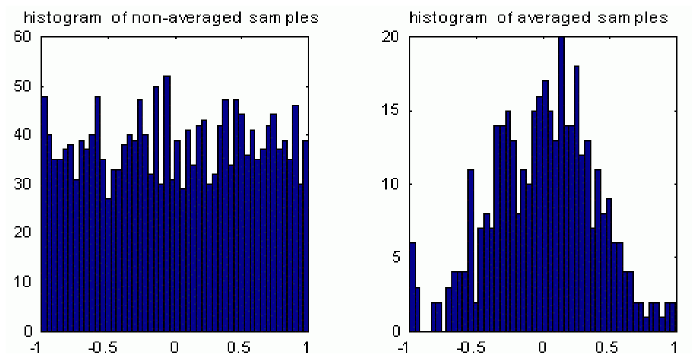

Figure 6: Noise histogram before and after averaging

3.1.2 Spectral analysis of the audio segment

In the spectral analysis we perform a FFT operation on a Hann windowed segment, and then find the absolute value of the spectrum by using square and square root functions.

The FFT operation - The FFT operation is done using the DSPLIB [7] optimized function. The problem is that in order to prevent overflow, the results of each FFT phase are scaled by a factor of two. In our case the signal is scaled down by a factor of 512 at the end of the FFT process, which reduces the accuracy of the spectrum analysis and consequently, hampers the building of the masking threshold. To overcome this problem and to increase accuracy without increasing computation time, before performing the FFT we do a preprocessing of the segment. The aim is to ensure that regardless of the signal level in the input of the FFT routine, the entire dynamic range of the FFT computation will be used. This is done by finding the maximum absolute value of the segment, then calculating a power of 2 scale factor in order to left justify the highest '1' bit of this number, meaning how many shifts are required to get a '1' bit in the word's MSB. Finally, scaling by this factor is performed on the entire segment. At the end of the embedding process, the signature is normalized back by the same factor so that the level of the overall signature stays the same. This operation enhances the accuracy of the FFT considerably especially in low-level signals.

Square root operation - After performing the FFT we find the absolute value of the spectrum. Calculating the absolute value of a complex vector is a time consuming operation. Although the square operation is implemented in the C54 as one operation, the calculation of the square root is rather complicated. We implemented the square root with an iterative algorithm described in figure 7.

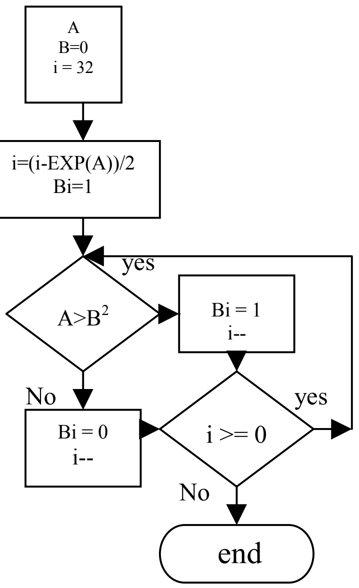

Figure 7 : Iterative implementation of fixed point square root

A is the number that we want to square root, B is the result, Bi is the i-th bit of B and the operation EXP (A) returns the number of leading zero bits before the most significant '1' in A. EXP (A) is implemented in the C54 as one CPU operation. The advantage of this method is that the number of iterations is dependant on the magnitude of A. Because the results of the FFT are usually small numbers, the square root operation is very fast in average (5-10 iterations). The second advantage is that the operation is more accurate than other methods, like the Taylor polynomial approximation for example. Retaining high accuracy in intermediate stages of the algorithm was a main consideration, because of the additive nature of the quantization error, imposed by the fixed-point representation. A segment absolute spectrum run-time result is shown in figure 9.

3.1.3 Masking threshold calculation

Fixed-point issues - The psycho-acoustic model represents energy levels in decibels. This yields to a vast use of logarithm and exponent operations. The number representation suitable for these operations is floating-point and not fixed point. Due to these facts, special design considerations had to be taken. In the following description we use the term "Qx.y" as a fixed-point representation where "x" is the number of integer bits and "y" is the number of fractional bits. The first consideration was the number representation, three different representations were used to cope with the wide range of results: the Q.15, Q3.12 and Q9.6. Where higher accuracy was needed, 32 bit Q16.15 and Q23.8 representations were used. Another consideration was to minimize transformations between log space and linear space, every transformation reduces accuracy considerably, so results of both linear and log space had to be kept in the on chip memory. The third consideration was optimizing the use of that memory, to maintain high-speed performance, in a strict memory allocation regime.

Logarithm calculation - The psycho-acoustic model tables and formulas are represented in decibels. This suggests, logarithmic scale operations on the calculated elements. Our implementation of the logarithm, uses the Taylor polynomial approximation of the function ln(1+x) . We performed then, a linear transformation on the model equations and tables to use a natural-base logarithm instead of the ten-base used in the model, in order to save constant multiplications and improve performance. A run-time segment spectrum in log-scale is shown in figure 9.

Logarithm result representation - The range of results from ln(x) operated on a signed Q.15 16 bit number is (-10.39,0). This range requires a Q4.11 representation, which positive part is unused. To optimize the output bit representation, the results are justified to the range of (-4.39,6), meaning the maximum spectral component will be +6 in Q3.12 representation (figure 9 top right graph, the maximum components is set to +6). The added value is subtracted back after the filtering of the noise.

Logarithm of a sum - Finding the value of non-tonal components, requires a logarithm calculation of the sum of spectral components values. Summing 16 bit numbers may results in overflow, so obviously a shift back is required in order to use 16 bit log function. The fact that a shift right is equivalent to a power of 2 division, enables us to correct it by adding a multiple of log (2) to the result, and maintain an accurate result, using a single word log function (figure 8).

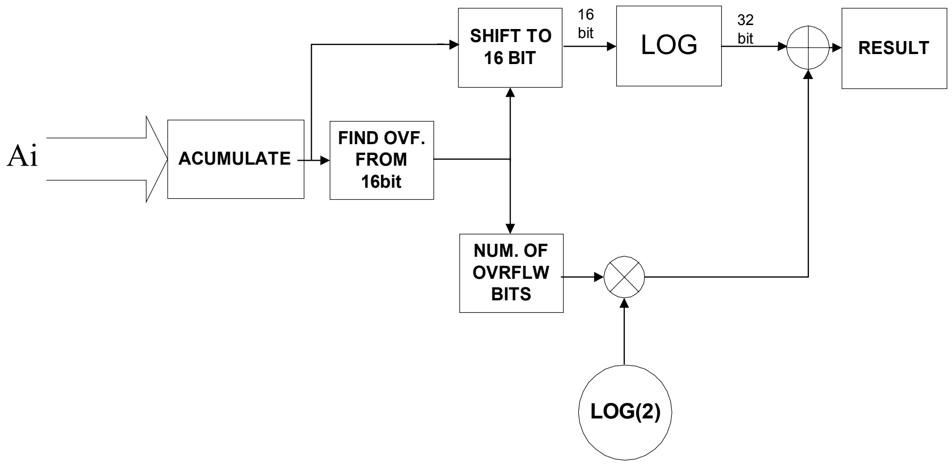

Figure 8 : Log calculation of a sum

Component detection - First, the tonal components are detected by a pass over spectrum. A list of the tonal component location is built. The non-tonal components are the weighted average of the components that are not tonal, in every critical band. The average is done in linear space, and then the result is transformed to log space. A list of the non-tonal components' locations is built too. A runtime detection of tonal and non-tonal components is presented in figure 9.

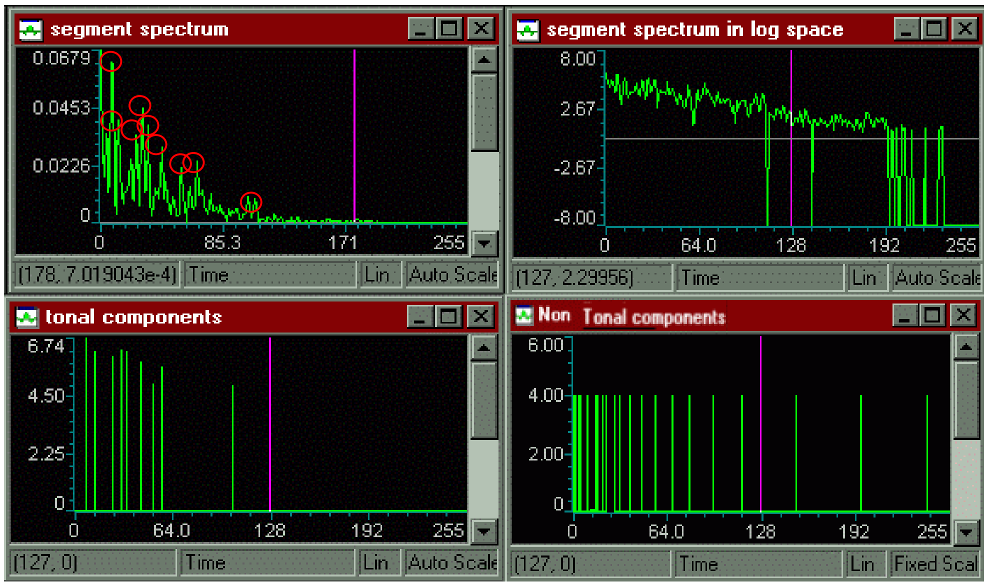

Figure 9: Spectral analysis and component detection - runtime results (tonal component markers were added externally)

Global masking threshold - Finding the global masking threshold is the most CPU time-consuming block; it takes almost 25% of the total embedding time. For each of the tonal and non-tonal components in the list, a masking curve is calculated. All the curves are then summed and added to the threshold in quiet, to create the global masking threshold. The masking curve is approximately a linear function in Bark units. The region around a component is divided into four sections relative to the location of the component. The curve has a different slope in each of the sections. The sections are located from 3 Barks lower, to 8 Barks higher than the component. Outside these boundaries the curve is zero. This range is not constant in frequency scale; moreover, it is frequency dependent in a non-linear form. For example, for the 9th component (775[Hz] frequency), The range (-3, +8) [Barks] will include the range from the 5th index to the 32nd index (in frequency units: 344[Hz] to 2670[Hz]) and for the 34th component, the same range in barks (-3, +8) will include the 22nd till 150th indexes (in frequency units: 1809[Hz] to 12919[Hz]). (See figure 10)

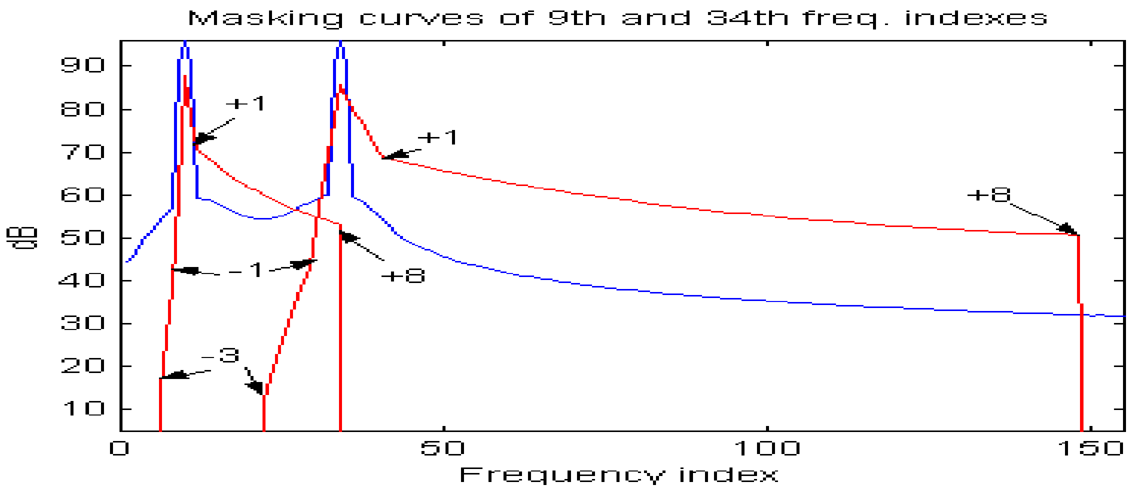

Figure 10 : The masking curve depends on the Bark distance and the frequency of the masking component

In order to calculate a curve, checks should be performed to find the masking domain of the specific component's curve. Our implementation does not use any conditional operations and thus is optimal. A table containing the frequency indexes for each component was built (for ex. The 9th entry holds the indexes 4, 7, 10, 31 corresponding to the frequency indexes in the relative Bark distances –3, -1, +1, +8 from the 9th component). For each component the curve was calculated in the appropriate range that was found in the right table entry.
When analyzing the behavior of the masking curve, one can see that in the range (0,+8)[Barks] the curve function is monotonically descending. In the implementation, if the curve function result goes below a certain threshold the current curve calculation stops, and the program moves to the next component. We use the fact that as the frequency distance increases, the power of tonal and non-tonal components decreases, consequently, the masking curves are lower and their contribution to the global masking threshold is negligible.
The summing of the contributions from all the components is done in linear space, while the curves are computed in log space; this yields to a necessity of using exponent operations.

Exponent calculation - Because of the wide range of inputs [-8, +6], the Taylor polynomial approximation alone is not suitable as an implementation. At this range, fifteen coefficients should be used, which slows the operation. Moreover, fixed-point representation does not allow coefficients of different magnitude scale, and therefore, only 8 coefficients can be represented at Q3.12 representation, the others are two small. If we consider a lookup table, for this range we will need 2^{21} bytes of memory. We combined the two methods and used the following property of the exponential function:

$$exp(a+b) = exp(a) \cdot exp(b)$$

Our method is: divide the number to its integer and fractional part. The exponent of the integer part is calculated using a lookup table, and the fractional part is calculated using the Taylor polynomial approximation with only five coefficients. The two results are then multiplied, and saved in a Q9.6 representation. This operation is fast and consumes a reasonable amount of memory.

3.1.4 Signature filtering

As mentioned, the pseudo-random noise filtering is needed in order that the amplitude of the noise in every frequency will match the calculated masking threshold. Due to the symmetry of the absolute value of the FFT result, the global masking threshold that was calculated from 512 samples has actually 256 values. Those values tell, for each frequency, how much noise can be added to the signal, without the human ear hearing it. This means that the masking threshold is the frequency response of the FIR filter that should be used to filter the pseudo random noise. There are two possible methods of filtering the noise:

I **Time domain filtering -** By performing the IFFT operation on the global masking threshold we will get the 512 coefficients of the filter. Performing a convolution will result in colored noise that matches the global masking threshold spectrum. The number of operations needed are: one IFFT operation (O(nlogn)), to get the filter coefficients and one convolution operation ($O(n^2)$), for filtering. The gain of the filtered noise must be raised to nearly match the gain of the mask, so another FFT operation is done (O(nlogn)) to find the gain. This sums up to three FFT operations and one convolution with a 512 coefficients filter.

II **Frequency domain filtering -** The complex spectrum of the noise is calculated. Filtering is done, by multiplying the mask vector with the noise complex spectrum. Applying the IFFT operation gives the filtered noise samples. The number of operations needed are: one FFT of the noise (O(nlogn)), multiplying the mask vector with the noise spectrum(O(n)), and one IFFT operation (O(nlogn), to get the filtered noise samples. This sums up to three FFT operations and one complex vector multiplication.

Obviously, filtering in frequency domain is faster than in time domain (for FFT/IFFT of 512). The problem is, that because we do not zero pad the noise, the frequency domain multiplication is equivalent to a cyclic convolution operation instead of a linear convolution. The justification is that because we filter white noise, it doesn't really matter that we perform a cyclic convolution, because the beginning and the end of the noise segment are not correlated. Additionally, we can still be able to reproduce the same signature from the original samples, and the noise spectrum matches the psycho-acoustic model. The runtime results of filtering in frequency domain are presented in figure 11.

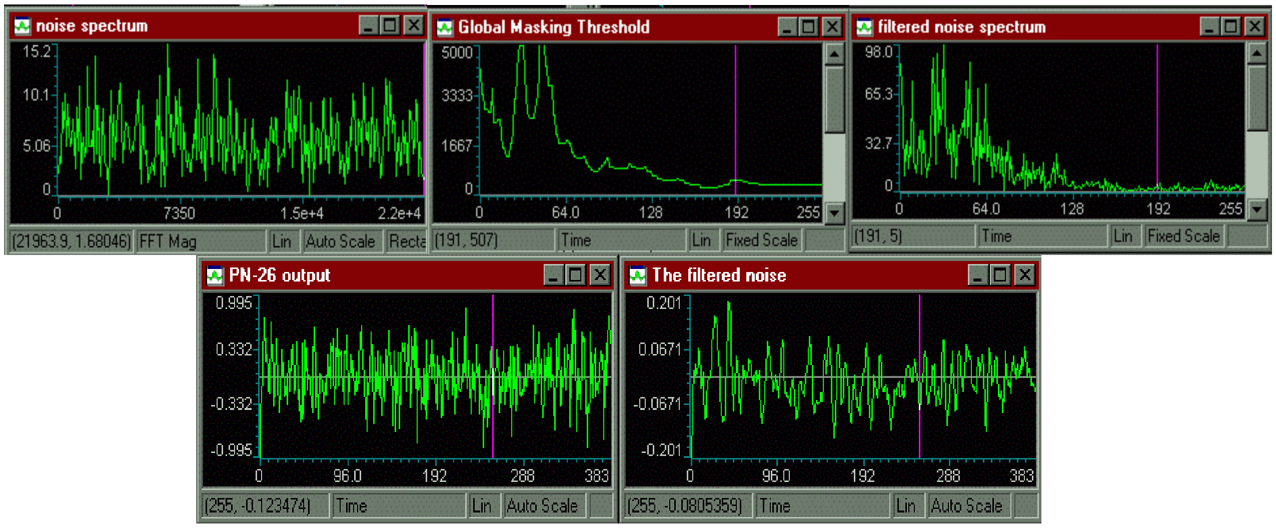

Figure 11 : Noise filtering – runtime results

3.1.5 Signature normalization to signal level

The filtered noise gain must be raised in order that its spectrum will nearly match the spectrum gain of the mask, that way we get the strongest signature we can, without being heard. To do that, the minimal difference between the mask and the filtered noise spectrum in log units must be computed. In order to spare unnecessary transformation between log-space and linear-space, the minimal log-space difference was found by calculating the minimal ratio in linear-space. We make a pass over the filtered noise and mask, for each index compute a ratio by dividing the mask components with the filtered noise components, and then find the minimum ratio. The range of the resulting ratio is so great that we had to implement a 32 bit exponent function and keep the result in 32-bit representation Q14.17. The filtered noise is then transformed by an IFFT operation to time domain and then multiplied by the above ratio. Because the ratio is kept in a 32-bit representation and the multiplying unit in the C54 performs only 16x16 bits multiplications, a special treatment was taken. The scale factor is left shifted so that its most significant '1' bit will be left justified. The filtered noise vector is then multiplied by the left-justified scale factor in a 16x16 bit operation. The result is shifted according to the justification done and stored in a Q.15 16bit format. In the C54 the “EXP” and “NORM” commands are used to left justify the scale factor. The multiplication and storing with shift is done efficiently by the “ST || MPY” with shift command, in a pipelined one cycle per multiplication loop.

3.2 Real-Time application

3.2.1 Hardware and peripherals

EVM implementation - The initial implementation of the system was on the SPECTRUM-DIGITAL XDS-510 EVM. A primary feasibility test was done to verify real-time embedding. The test was done to measure one segment process time. The sample rate used was 44.1[KHz] and each segment has 384 samples, so a real-time application should process each segment in less than 8.7[mSec]. A segment was processed repeatedly on the DSP, and an I/O pin level was swapped between segments. An external oscilloscope was connected to the I/O pin and measured the processing time. The measured processing time upper bound was 7.5[msec] and further optimized to 4.8[msec]. The layout of the feasibility test is shone in figure 12.

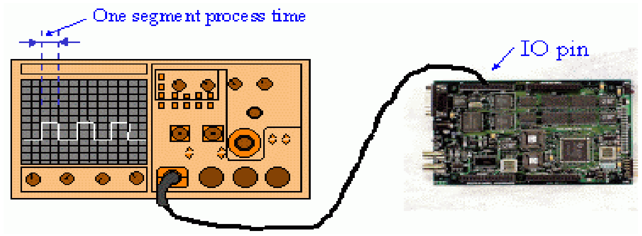

Figure 12 : Real-Time feasibility test layout

A HOST application was written in Visual Basic. The application used the RTDX capabilities to transfer audio samples to the C54, and to transfer watermarked samples from it. The EVM was connected to the HOST-PC via the JTAG emulation port. The transfer rate was not sufficient for a real-time transfer rate of 44.1 K-words per second. This led to the need of HPI connected hardware.

TIGER 5410/PC implementation - The Tiger development board has a PC-ISA bus interface. The TMS320C5410 HPI port is mapped to the ISA interface and therefore can be directly accessed by the HOST-PC. The fast HPI port enables the HOST-PC to read and write data from the DSP internal memory without interfering DSP processing. This connection was used to transfer audio samples. The board has additional bi-directional status and control registers, which can be accessed by both the HOST and the DSP. These registers were used to synchronize the HOST and the DSP. The connection to the “analog world” was the on board CS4216 audio CODEC. The CODEC is mapped to the C54 MCBSP. Using the C54 DMA, sampled analog data was transferred to and from the DSP. Both the HPI connection and the MCBSP with the DMA, operate independently, without occupying the DSP-CPU [2]. The scheme of the system is presented in figure 13:

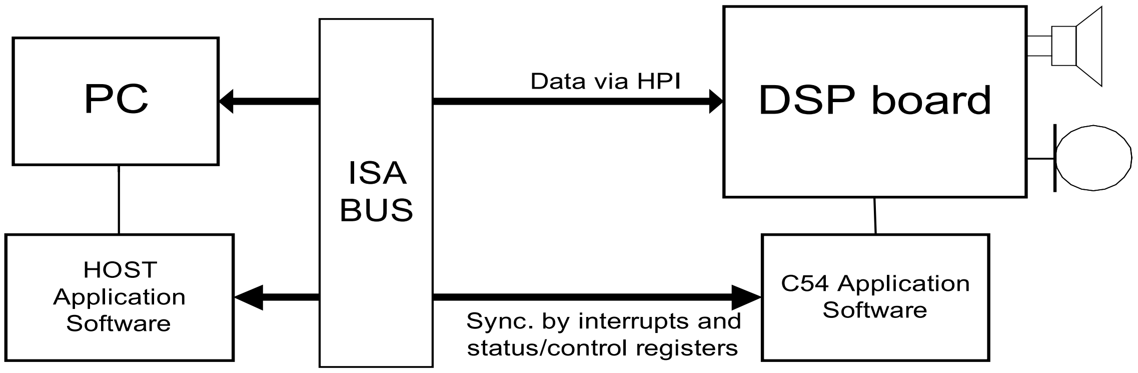
PC
ISA BUS
Data via HPI
DSP board
HOST Application Software
Sync. by interrupts and status/control registers
C54 Application Software

Figure 13: The scheme of the real-time embedding system

3.2.2 Modes of operation

A HOST-PC and a DSP application was developed supporting the three modes of operation (see introduction). Double buffering and read-write synchronization of both DSP-PC and DSP-CODEC paths were implemented.

The Offline-mode - In this mode a WAV file in the HOST-PC, is transferred to the DSP, watermarked, and sent back to the HOST-PC. The HOST-PC reads and writes data directly to the internal memory of the C54 using the HPI. The DSP has to be synchronized with the HOST-PC in both reading and writing.Figure 14 illustrates the control signals used.

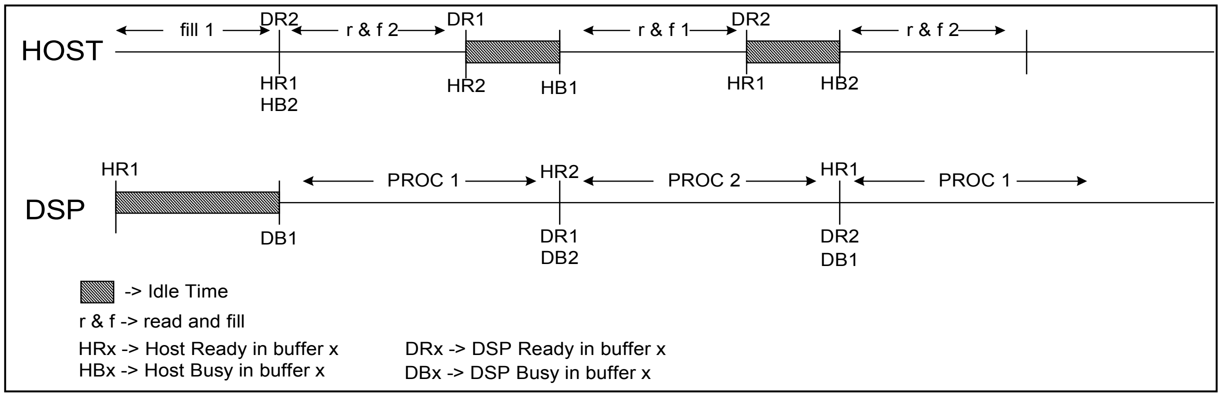

Figure 14 : Offline-mode synchronization control signals

The Digital playback mode - In this mode a WAV file in the HOST-PC is transferred to the DSP, watermarked, and played on the DSP board D/A converter. In this mode the D/A converter sets the real-time timing of the system. The DSP has to be in synchronization with both D/A converter and HOST-PC. Figure 15 illustrates the control signals used.

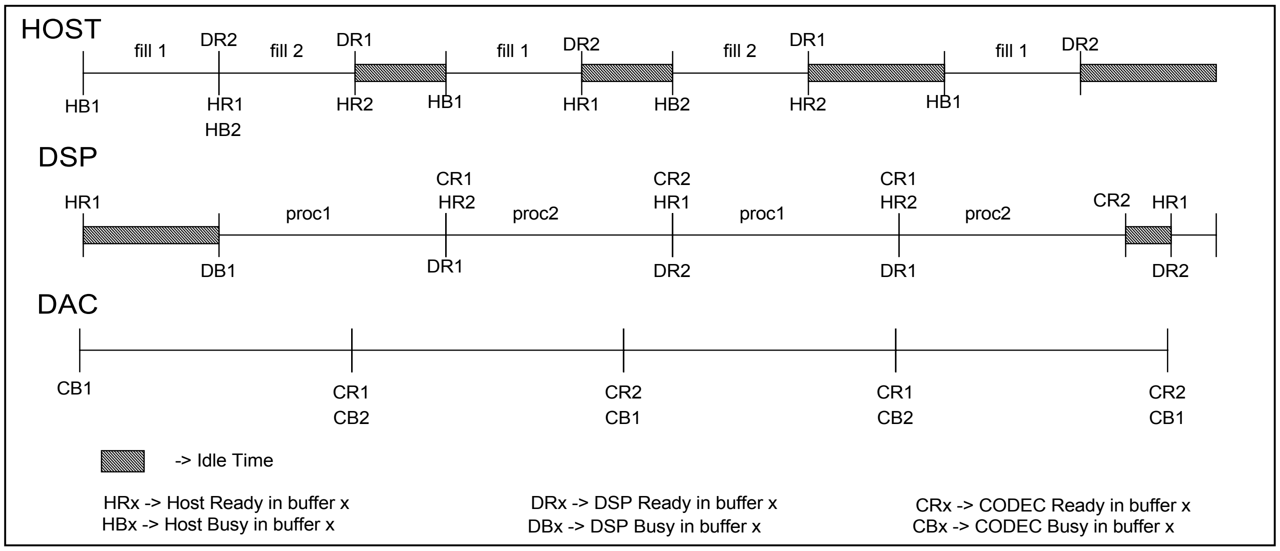

Figure 15 : Digital playing mode synchronization control signals

The Live mode - In this mode a live signal is sampled using the on board A/D converter, watermarked, and played in real-time using the on board D/A converter. The samples of the original signals are transferred to the HOST-PC and saved for future detection. Input and output synchronization has to be maintained for both DSP-HOST and DSP-CODEC paths. Figure 16 illustrates the control signals used.

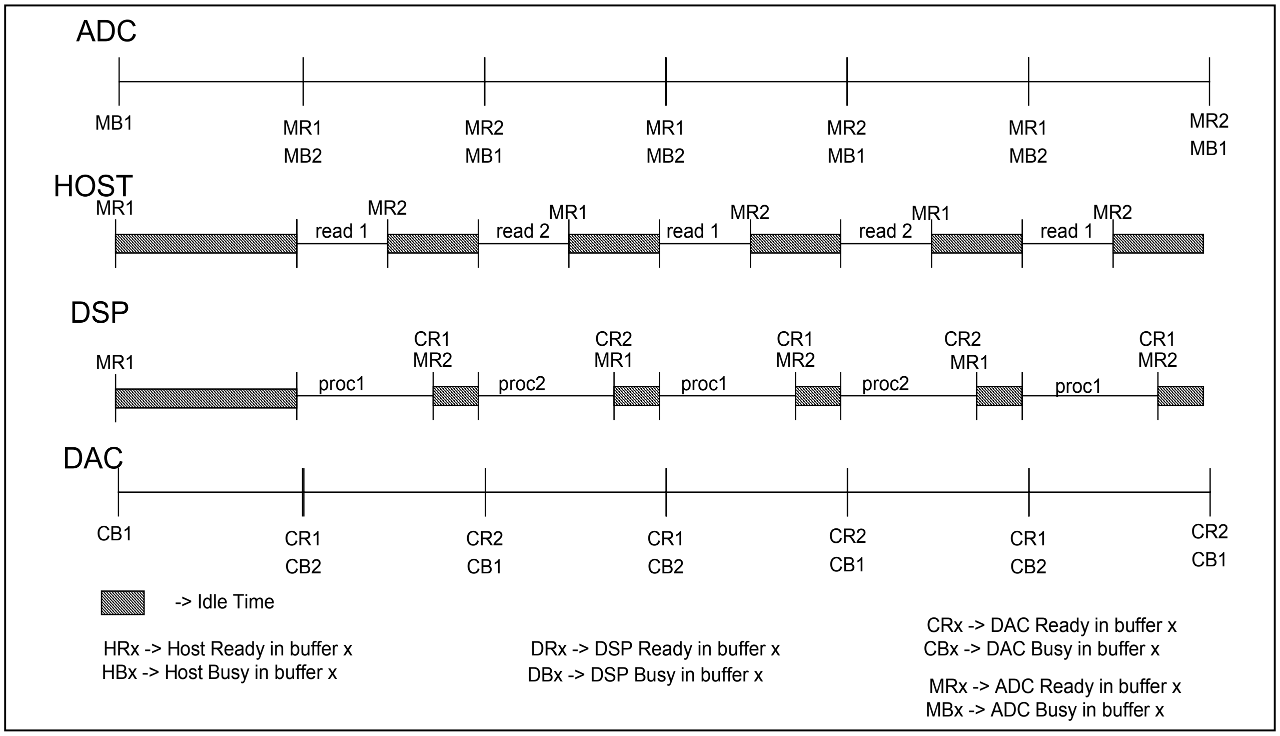
ADC
MB1
MR1
MB2
MR2
MB1
MR1
MB2
MR2
MB1
MR1
MB2
MR2
MB1
HOST
MR1
read 1
MR2
read 2
MR1
read 1
MR2
read 2
MR1
read 1
MR2
DSP
MR1
proc1
CR1
MR2
proc2
CR2
MR1
proc1
CR1
MR2
proc2
CR2
MR1
proc1
CR1
MR2
DAC
CB1
CR1
CB2
CR2
CB1
CR1
CB2
CR2
CB1
CR1
CB2
CR2
CB1
-> Idle Time
HRx -> Host Ready in buffer x
HBx -> Host Busy in buffer x
DRx -> DSP Ready in buffer x
DBx -> DSP Busy in buffer x
CRx -> DAC Ready in buffer x
CBx -> DAC Busy in buffer x
MRx -> ADC Ready in buffer x
MBx -> ADC Busy in buffer x

Figure 16 : Live mode synchronization control signals

4 Summary

The digital watermarking algorithm was studied and a simulation was written in MATLAB^{TM}. The simulation was used to conduct extensive performance tests to verify inaudibility of the signature and its robustness against distortions. A windows application for signature embedding and detection was developed [4], but the embedding processing time was not satisfactory for a real-time application. Embedding one segment on an Intel Pentium^{TM} 400[MHz] processor, took eight times the segment duration in 44.1 [KHz] sampling rate.

In order to support real-time signature embedding application an implementation on the Texas Instruments DSP TMS320C54x was considered. A fixed-point algorithm was developed, to meet the model specifications. The algorithm was implemented on a TMS320C5410 EVM using TI's Code Composer Studio. The core-algorithm was written in assembly language and the management and flow control, in C language. Initial functionality and signature quality was tested using the CCS tools: the graph displays, probe points, and profiling. A performance test was done and real-time capabilities were proven as a concept. Unfortunately, a bottleneck in Host-DSP communication, prevented sufficient data transfer rate to meet real-time.

The TIGER 5410/PC platform was chosen to solve the data-transferring problem through the PC-ISA bus that is connected to the C5410 HPI port. A full functioning application was developed for both PC and DSP. For each of the three modes of operation necessary control mechanisms were added, such as HOST-DSP synchronization, double buffered I/O to HOST-PC, and double buffered analog I/O to and from the on board CS4216 Audio Codec. The application is activated via a GUI.

The project was presented in ICASPP 2000 in Istanbul in TI's booth, and in the "3rd European DSP Education and Research Conference" in Paris.

The application was designed in a way that it can be easily upgraded to a commercial product. The fast processing allows implementing stereo embedding by adding minor software modifications. The algorithm contains independent blocks. This fact enables parallel execution and a possibility to extend the algorithm to use a more cryptographically "safe" noise generator. Moreover, multi-channel embedding can be implemented by a multiprocessor system.

Our TMS320C5410 based system achieved a very high performance; embedding time of a segment was reduced to almost **half** of its playing time at a rate of 44.1[KHz]. In this fast processing rate the embedded signature showed very high quality. The signature could not be heard at 21[dB] signal to signature ratio. A similar SNR of 21[dB] of AWGN is heard and reduces the quality of the signal considerably. These results on the low-cost, low-power processor are a considerable improvement relative to the previously discussed PC implementation (more than 10 times faster).

5 Acknowledgement

The authors would like to thank the Signal and Image Processing Lab staff, for the help and support during the whole year and a half of working on the project.
The authors are also thankful of the support provided by TI through the Elite program.

6 References

[1] Mitchell D. Swanson, Mei Kobayashi, and Ahmed H. Tewfik, "Multimedia Data-Embedding and Watermarking Technologies," Proc. of the IEEE, Vol. 86, No. 6, June 1998, pp. 1064-1086.

[2] TMS320C54x DSP volume I – CPU and Peripherals Reference Set, Texas Instruments SPRU131D (1997).

[3] Shay Mizrachi , "Robust Detection of Digital Watermarking Signature in Audio Signals" An M.Sc thesis , Dept. of EE, Technion IIT, Haifa, Sep. 2000 (In Hebrew)

[4] Tal Mizrahi, Eran Borenstein, George Leifman, "Digital Watermarking of Audio Signals, Using the Psycho-Acoustic Model", Project report SIPL, Department of EE, Technion IIT, Haifa (In Hebrew).

[5] Y. Cassuto, M. Lustig, G. Leifman, T. Mizrahi, E. Borenstein, S. Mizrachi, N. Peleg, "Real Time Implementation for Digital Watermarking in Audio Signals Using Perceptual Masking", The 3rd European DSP Education and Research Conference, ESIEE, Noisy Le Grand, Paris (September 2000).

[6] ISO/IEC, JTC1/SC29/WG11 MPEG, "Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s – Part 3: Audio", IS11172-3 1992 ("MPEG-1").

[7] Optimized DSP Library for C Programmers on the TMS320C54x, Application Report SPRA480A (January 2000)

[8] R.L Rivest, A. Shamir, and L. Adleman. "A method for obtaining digital signatures and public key cryptosystems", Communications of the ACM, 21 (1978) pp. 120-126.